IL MIO COMPUTER

CORSO PRATICO DI INFORMATICA: MICRO, MINI E PERSONAL COMPUTER



ISTITUTO GEOGRAFICO DE AGOSTINI

IL MIO
COMPUTER

IL MIO COMPUTER

CORSO PRATICO DI INFORMATICA: MICRO, MINI E PERSONAL COMPUTER

volume terzo

ISTITUTO GEOGRAFICO DE AGOSTINI

Direzione: Achille Boroli - Adolfo Boroli
Direzione editoriale: Mario Nilo; *settore fascicoli:* Jason Vella

Redazione dell'edizione italiana a cura di:
Edizioni Remo Sandron - Firenze
Redazione dell'edizione inglese:
Richard Pawson (*Editor*); David Tebbutt (*Consultant Editor*); Chris Cooper (*Executive Editor*); David Whelan (*Art Director*); Gareth Jefferson, Roger Ford (*Collaborators*)
Designed and Produced by Bunch Partworks Limited
Traduzione a cura di: Daniel Quinn

Coordinamento grafico: Otello Geddo
Coordinamento fotografico a cura del Centro Iconografico dell'Istituto Geografico De Agostini

Titolo originale dell'opera: The home computer course
© Orbis Publishing Limited, Londra, 1983.
© Istituto Geografico De Agostini S.p.A., Novara, 1984.
Stampato in Italia - IGDA Officine Grafiche, Novara, 1984.

Sommario

481 Cosa imparare per il futuro?
484 La scelta migliore
486 Ant Attack: impariamo da un gioco
488 L'algebra di Boole
490 Novità Commodore
493 Software in vetrina
494 Il BASIC dello Spectrum
496 Gli aspetti di base
500 Bill Gates
501 Il computer per insegnare
504 Funzioni e strutture di controllo
506 Il WordStar
509 La scelta di un buon monitor
512 Sommare... con logica
514 I Microdrive Sinclair
516 I banchi di memoria
519 Il successo della Atari
521 Software per divertimento
523 Il chip Motorola 68000
524 I primi passi
526 Processi logici semplificati
529 Software in vetrina
530 Spectrum terza versione
532 Unità a dischi Commodore
534 L'ABC del... BBC!
536 I programmi monitor
540 La Acorn Computers
541 Esami al computer
543 Unità a dischi Atari
546 La logica del programmatore
548 Dissaldare
550 Metodi di stampa
553 Software in vetrina
554 Word processing sul BBC!
556 Analisi del testo
559 Il software della Imagine
561 L'incredibile Hulk
564 Unità a dischi BBC
566 Banco di prova
568 Software in vetrina
569 Il Personal dell'IBM
572 Le mappe di Karnaugh
574 Il BASIC dei Commodore
576 Alcune utili convenzioni
580 Entra in scena la Xerox
581 Facile accesso
584 Unità a dischi della Dragon
586 Dalla mappa al circuito
588 Software in vetrina
589 L'AIM 65 della Rockwell
592 L'ufficio computerizzato
594 Le leggi elettroniche
596 Byte per byte
599 Le menti della Commodore
601 Contabilità e computer
604 Incompatibilità dei dischi
606 Una risposta logica
609 Software in vetrina
610 Il Dragon 64
612 Le immagini viewdata
615 Largo alla memoria
618 L'interno dei componenti
620 La Oric
621 Lo standard MSX
624 Contatti... logici
626 La conversione dei segnali
628 Software in vetrina
629 Lo Spectravideo 318
632 Tiriamo le somme
634 Qualche grado... in Basic
636 Linea di partenza
639 La giapponese SORD
641 Sopravviva il migliore
644 Mezza misura
646 Strade alternative
648 Software in vetrina
649 La Grafpad
652 Ordini e magazzini
654 Gradi di precisione
656 Sventola una bandiera
660 La Virgin Games
661 I sistemi operativi
664 Memoria sotto controllo
666 Un po' di parità
668 Software in vetrina
669 Novità Atari
672 Controllo sulle scorte
674 Facciamo il punto
676 Modi d'indirizzamento
680 La Tandy Corporation
681 Solchi di luce
684 I file sequenziali
686 Un convertitore per display
688 Software in vetrina
689 Soluzioni agli esercizi
690 Lo Sharp PC-5000
692 Resoconto finale
694 La grafica del Commodore
696 Etichette e cicli
700 La Zilog
701 Tutto sotto controllo
704 Disegniamo al computer
706 L'accesso ai file
708 I flip-flop RS
710 Poligono di tiro
712 Gli sprite del Commodore
715 Software in vetrina
716 Subrotine trasportabili
719 La Digital Research

Indice per settore

Panorami

481 Cosa imparare per il futuro?
493 Software in vetrina
506 Il WordStar
529 Software in vetrina
553 Software in vetrina
554 Word processing sul BBC!
561 L'incredibile Hulk
568 Software in vetrina
581 Facile accesso
588 Software in vetrina
609 Software in vetrina
612 Le immagini viewdata
621 Lo standard MSX
628 Software in vetrina
641 Sopravviva il migliore
648 Software in vetrina
661 I sistemi operativi
668 Software in vetrina
681 Solchi di luce
688 Software in vetrina
701 Tutto sotto controllo
715 Software in vetrina

Hardware

484 La scelta migliore
490 Novità Commodore
509 La scelta di un buon monitor
514 I Microdrive Sinclair
523 Il chip Motorola 68000
530 Spectrum terza versione
532 Unità a dischi Commodore
543 Unità a dischi Atari
550 Metodi di stampa
564 Unità a dischi BBC
569 Il Personal dell'IBM
584 Unità a dischi della Dragon
589 L'AIM 65 della Rockwell
604 Incompatibilità dei dischi
610 Il Dragon 64
629 Lo Spectravideo 318
649 La Grafpad
669 Novità Atari
690 Lo Sharp PC-5000
710 Poligono di tiro

Software

486 Ant Attack: impariamo da un gioco
501 Il computer per insegnare
521 Software per divertimento

541 Esami al computer
592 L'ufficio computerizzato
601 Contabilità e computer
632 Tiriamo le somme
652 Ordini e magazzini
664 Memoria sotto controllo
672 Controllo sulle scorte
684 I file sequenziali
692 Resoconto finale
706 L'accesso ai file

Le chiavi dell'Informatica 

488 L'algebra di Boole
512 Sommare... con logica
526 Processi logici semplificati
546 La logica del programmatore
572 La mappe di Karnaugh
586 Dalla mappa al circuito
606 Una risposta logica
626 La conversione dei segnali
646 Strade alternative
666 Un po' di parità
686 Un convertitore per display
708 I flip-flop RS

Programmazione BASIC 

494 Il Basic dello Spectrum
504 Funzioni e strutture di controllo
534 L'ABC del... BBC!
574 Il BASIC dei Commodore
634 Qualche grado... in Basic
654 Gradi di precisione
694 La grafica del Commodore
704 Disegniamo al computer
712 Gli sprite del Commodore

Linguaggio macchina 

496 Gli aspetti di base
516 I banchi di memoria
536 I programmi monitor
556 Analisi del testo
576 Alcune utili convenzioni
596 Byte per byte
615 Largo alla memoria
636 Linea di partenza
656 Sventola una bandiera
676 Modi d'indirizzamento
696 Etichette e cicli
716 Subrotine trasportabili

I pionieri dell'Informatica 

500 Bill Gates
519 Il successo della Atari
540 La Acorn Computers
559 Il software della Imagine
580 Entra in scena la Xerox
599 Le menti della Commodore
620 La Oric
639 La giapponese SORD
660 La Virgin Games
680 La Tandy Corporation
700 La Zilog
719 La Digital Research

Laboratorio 

524 I primi passi
548 Dissaldare
566 Banco di prova
594 Le leggi elettriche
618 L'interno dei componenti
624 Contatti... logici
644 Mezza misura
674 Facciamo il punto
689 Soluzioni agli esercizi



PAUL CHAVE

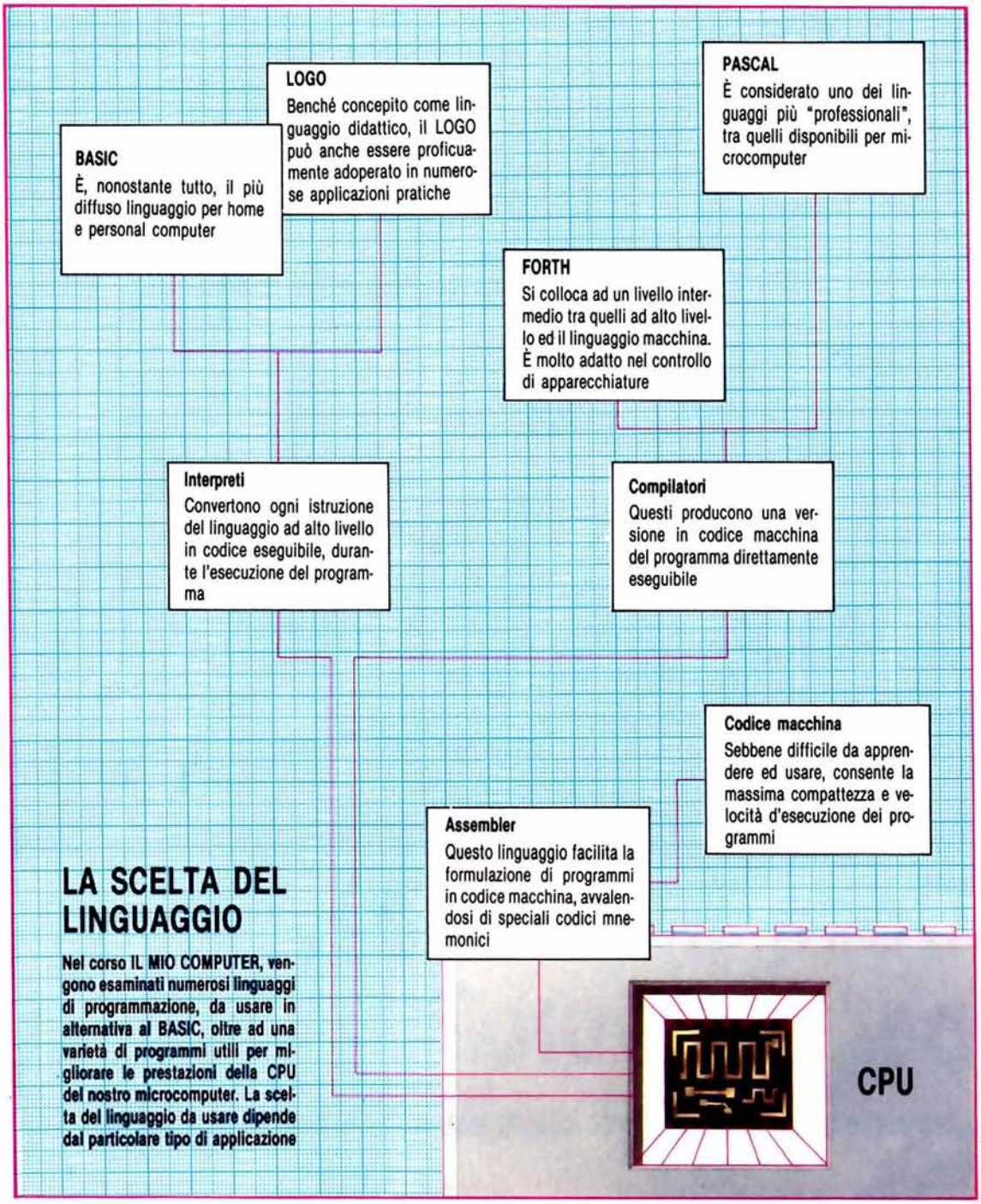
Cosa imparare per il futuro?

Una serie di consigli utili per coloro che desiderano accostarsi all'informatica

Tenersi aggiornati sui più recenti sviluppi dell'informatica è già sufficientemente arduo per gli addetti ai lavori (programmatori, analisti, ecc.), benché ciò faccia parte del loro lavoro; ma come deve comportarsi un dilettante, o un semplice appassionato?
Certo, può entrare nel primo negozio che capita ed acquistare un qualche economico microcomputer; ma già in questa categoria la scelta non è facile, se non si è ben consigliati. E poi? Forse l'apparecchio dispone di più linguaggi, ma qual è il più adatto? E, ancora, quali programmi conviene acquistare, tra quelli offerti dal mercato? Non parliamo poi degli accessori: dischetti o nastri? Una stampante seriale o una parallela?

L'evoluzione in questo settore è stata talmente rapida che è veramente difficile sapersi orientare, nonostante la ormai immancabile presenza di computer in uffici, fabbriche, scuole e, perfino, nelle automobili. Per molti, la soluzione più ovvia consiste nell'iscriversi ad uno dei tanti corsi di programmazione, tenuti da istituti privati. Generalmente, oltre ad uno specifico linguaggio di programmazione, viene anche data una panoramica generale sull'informatica. Tuttavia, questo genere di corsi è, necessariamente, di carattere troppo generico per chi desideri imparare ad usare uno specifico modello di computer.
I manuali operativi dei microcomputer posso-

La società del domani
Nel breve arco di dieci anni, i computer si sono diffusi ad ogni livello, grazie a tecnologie che consentono una produzione di massa, a basso costo, di questi apparecchi. Tenere il passo con un così rapido progresso è già difficile per gli operatori del settore, ma soprattutto per chi si avvicina per la prima volta al complesso campo dell'informatica. Un corso d'apprendimento, graduale e ben strutturato, può essere una soluzione per non rimanere esclusi dalla vita di domani



no offrire una vasta gamma d'informazioni, ma difficilmente vi si troveranno dei confronti con le prestazioni di altre marche di computer. Probabilmente, per chi ha già una propria occupazione e desidera approfondire le proprie conoscenze sui computer, la migliore soluzione consiste nel seguire un programma d'apprendimento, a casa propria. Contemporaneamente, può essere utile frequentare un corso serale presso qualche istituto, per confrontare e mettere in pratica le nozioni apprese.

Lo scopo del programma d'apprendimento non deve essere limitato unicamente all'uso di un solo linguaggio di programmazione e di un particolare home computer, ma deve spaziare, invece, su tutto l'arco delle applicazioni di computer nella vita quotidiana, sulle periferiche disponibili e sui programmi in commercio. Inoltre, la programmazione deve riguardare gli apparecchi di maggior diffusione e non soltanto un particolare tipo di computer. Infine, è necessaria anche una parte dedicata alla teoria: dagli elementi di logica formale alla storia del computer, dai sistemi numerici alla formulazione di modelli computerizzati.

Nel concepire la struttura de IL MIO COMPUTER, per esempio, tutti questi elementi sono stati tenuti in buon conto: partendo da nozioni elementari, applicabili agli home computer più diffusi, e dal linguaggio BASIC, anch'esso il più diffuso, il lettore viene gradualmente portato ad apprendere anche

l'uso di altri linguaggi ad alto livello per microcomputer (PASCAL, LOGO, FORTH, C, ecc.), come pure il linguaggio macchina relativo ai più diffusi microprocessori.
Conoscere il linguaggio macchina è essenziale per comprendere a fondo il funzionamento dei computer e dei linguaggi a più alto livello, al punto che potremmo addirittura dedicarci alla formulazione di un nuovo linguaggio, maggiormente adatto alle nostre esigenze.
Inoltre, dal punto di vista teorico-pratico, una buona conoscenza dei principali metodi adottati nella struttura dei dati e nella gestione degli archivi (siano essi su nastro o su dischetto) consente di sfruttare al massimo anche le limitate risorse dei più piccoli home e personal computer.
Parallelamente, un'ampia e particolareggiata panoramica sui "pacchetti" software reperibili in commercio (word processor, lavagne elettroniche, database, generatori di programmi, ecc.) permette di impratichirsi con le caratteristiche e le prestazioni di ciascuno di essi e, all'occorrenza, di saper scegliere quello più adatto ad una particolare situazione.
Per completare questo vasto arco di materie è anche prevista una sezione de IL MIO COMPUTER dedicata all'elettronica, grazie alla quale si apprende il funzionamento dei vari circuiti di un computer e come intervenire, ovviamente nei casi più elementari, sui vari circuiti e connessioni. Vengono esaminate le caratteristiche dei più diffusi home e personal computer, assieme alle applicazioni che questi apparecchi possono avere sia nell'ambito domestico, sia in quello scolastico, sia, infine, nel campo del lavoro.
Un corso così concepito, rappresenta un valido strumento per coloro che desiderano crearsi un'approfondita conoscenza sui computer, allo scopo di migliorare la propria posizione di lavoro, non disponendo del tempo necessario per frequentare un istituto specializzato. Comodamente, a casa propria e nelle ore più opportune, si possono apprendere le materie fondamentali per restare al passo coi tempi.
Tra le pagine de IL MIO COMPUTER si possono trovare molte delle risposte ai quesiti più frequenti, ad esempio: quale impatto avrà, sui rapporti umani, l'avvento delle più sofisticate tecnologie computerizzate? Si avranno ripercussioni anche sull'atteggiamento politico delle nazioni che si troveranno ai vertici dell'evoluzione tecnologica? Quale sarà il ruolo dei genitori e degli insegnanti, a seguito del radicale cambiamento apportato dai computer nel settore didattico? In genere, trovare una risposta a queste domande non è facile: la stampa e gli altri mezzi di comunicazione trattano sporadicamente argomenti come questi, lasciando perplessa e dubbiosa la maggior parte della gente. Una cosa è certa: mai come oggi, l'informazione, ma soprattutto la capacità di accedere, di elaborare e di selezionare il maggior numero di informazioni, è stata così importante. Simultaneamente alla crescente diffusione di apparecchi in grado di espletare simili funzioni, a costi sempre minori, va aumentando la richiesta di persone ben preparate, adatte a manovrarli con dimestichezza e cognizione di causa. Molti scrittori, in passato, hanno paventato la formazione di gruppi privilegiati, all'interno delle società future, delle specie di "superuomini", identificabili, volendo, con i futuri superesperti dell'informatica.
Ebbene, riteniamo che tali supposizioni debbano essere sdrammatizzate o quantomeno ridimensionate: i computer sono e devono restare uno strumento capace di alleggerire le fatiche umane. Saranno appunto la preparazione e la maturità che ciascuno di noi ha oggi la possibilità di formarsi ad equilibrare ed arginare i tentativi di far predominare la macchina sull'uomo.

UN SALTO DI QUALITÀ

La Sinclair ha "divorziato" dalla CPU Z80, adoperata in tutti i modelli precedenti, per passare ad un microprocessore a 32 bit, il 68000 della Motorola. Il QL Sinclair, oltre a questa innovazione, è dotato di 128 Kbyte di RAM (estendibili a 512 Kbyte) e due Microdrive incorporati. Si tratta di una generazione totalmente nuova di computer per una fascia intermedia di utilizzazione, che sta tra l'home ed il personal computer



IAN McKINNELL

La scelta migliore

Vantaggi e svantaggi dei più comuni sistemi di memorizzazione di massa

ATTENZIONE!

I dischetti vengono irreparabilmente danneggiati dai campi magnetici: accostandoli ad oggetti anche apparentemente innocui, come un telefono o un altoparlante, ad esempio, si può causare la perdita di tutti i dati!

I microcomputer sono strumenti molto versatili per l'elaborazione di dati, ma l'utilità di simili apparecchi viene meno se non si dispone di un adeguato metodo per conservare le informazioni, una volta spento il computer. Chi possiede un home computer si rende conto ben presto delle forti limitazioni imposte dalle cartucce ROM o dai nastri magnetici "a cassetta", per l'archiviazione dei dati.

Cartucce

Questo metodo è di scarsa utilità per il programmatore. La maggior parte delle cartucce contengono un certo tipo di PROM (Programmable Read Only Memory, ossia Memoria Programmabile a Sola Lettura) sulla quale è registrato un gioco o, magari, un programma applicativo. Queste unità servono, però, soltanto in lettura e non è possibile modificarne il contenuto. Esistono, comunque, dei tipi di cartucce dotate di memorie riprogrammabili, ossia le EEPROM (Electrically Erasable PROM, ovvero PROM cancellabili elettricamente). In questo caso, tali memorie, chiamate anche "non volatili", vengono impiegate in modo simile alla RAM, ma con il vantaggio, non indifferente, di conservare i dati anche quando viene tolta l'alimentazione o quando esse vengono estratte dal connettore del computer. Un altro sistema, analogo nei risultati, impiega dei chip di RAM, costruiti secondo la tecnologia CMOS, alimentati autonomamente mediante piccoli accumulatori interni alla cartuccia.

Il principale svantaggio, sia delle cartucce EEPROM che di quelle a RAM CMOS è il prezzo: crearsi una piccola dotazione di queste cartucce costa quanto acquistare un'unità a dischetti.



IAN McKINNELL

L'unità a dischi del Micro BBC
Prima di poter usare l'unità a dischetti, occorre montare sul computer la ROM che contiene il sistema operativo per dischi (DOS).
Al contrario, nelle unità a dischi "intelligenti" il DOS è contenuto nelle memorie dell'unità stessa

Nastri a cassetta

L'uso dei nastri a cassetta venne inizialmente diffuso per offrire un'economica alternativa alle prime costosissime unità a dischetti, ma conserva, ancora oggi, una vasta popolarità, in quanto quasi tutti possiedono già un normale registratore a cassette ed i nastri sono facilmente reperibili. In genere basta un normale registratore domestico, ma alcune ditte costruttrici (come nel caso della Commodore e dell'Atari) consentono di usare soltanto apparecchi appositamente progettati per i propri computer.

I programmi ed i dati vengono conservati, sui nastri, in "file" (archivi) sequenziali, usando particolari toni acustici per rappresentare i valori binari 0 ed 1. Normalmente, i circuiti di controllo del computer provvedono a registrare sul nastro una serie di informazioni, quali, ad esempio, il nome dell'archivio o del programma ed, eventualmente, l'indirizzo della locazione di memoria relativo al particolare blocco di istruzioni o dati, prima di memorizzare il resto. Il trasferimento avviene, un bit alla volta, per formare una sequenza di byte, generalmente raggruppati in blocchi di 256 byte. Molti microcomputer prevedono, inoltre, un sistema di controllo automatico degli errori, chiamato "checksum". Questo sistema permette di rilevare eventuali inesattezze, in fase di lettura, nei programmi o nei dati.

I comandi, tipici di un sistema basato sui nastri a cassette, sono SAVE (per memorizzare informazioni sul nastro) e LOAD (per leggere informazioni dal nastro). Negli apparecchi più sofisticati sono anche disponibili dei comandi per ottenere l'elenco dei file presenti sul nastro.

La facilità d'uso e l'economicità dei nastri a cassette sono, però, controbilanciate da alcuni svantaggi:

1. Nella maggioranza dei casi, occorre eseguire manualmente parte delle operazioni (riavvolgimento del nastro, aggiustamento del livello di volume, ecc.).
2. Poiché le informazioni vengono conservate in forma sequenziale, la ricerca di un particolare file (eccezion fatta per il sistema automatico "Hobbit" e per il registratore a microcassette dell'Epson HX-20) non è affatto rapida. Può migliorare se il registratore dispone di

un contagiri, altrimenti la procedura è piuttosto laboriosa e lenta. Inoltre, la conservazione sequenziale dei dati è scarsamente efficiente per la maggior parte dei programmi applicativi, nei quali, spesso, occorre estrarre, elaborare e riscrivere soltanto una piccola porzione di dati, e non tutto l'intero archivio. Non è dunque disponibile il sistema cosiddetto "ad accesso casuale", indispensabile per una valida gestione di database.

3. Il trasferimento dei dati, sui nastri, avviene con una velocità (o lentezza?) compresa, tipicamente, tra 300 e 1200 bit al secondo. Pertanto, considerati anche i due punti precedenti, un sistema a cassette risulta decisamente lento. L'archiviazione o la lettura di programmi anche di modeste dimensioni (5 Kbyte, ad esempio) possono richiedere diversi minuti. Oltretutto, una simile lentezza non invoglia a produrre copie di sicurezza dei programmi o dei dati, con tutti i rischi che ciò comporta.

4. Le informazioni registrate su nastri, a lungo andare, possono deteriorarsi, per via della normale usura del supporto magnetico.

5. A causa delle notevoli diversità, tra registratori di marche diverse, è talvolta impossibile effettuare scambi di programmi su cassette tra possessori di un medesimo modello di microcomputer.

I floppy disk

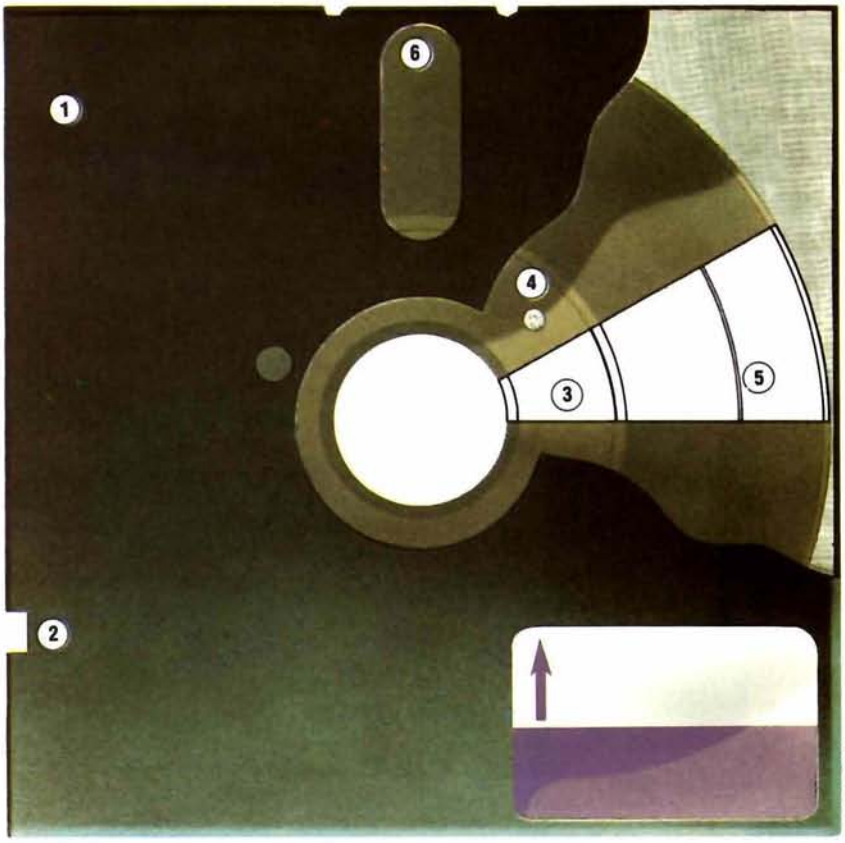
Se paragonato con i due sistemi precedenti, quello dei dischetti, o floppy disk, presenta senz'altro un numero minore di inconvenienti. Le unità a dischetti sono di costruzione complessa e delicata, ma soprattutto costosa. Ed anche gli stessi dischetti non sono certo economici, costando mediamente dalle 4 alle 8 volte di più di un nastro a cassetta. D'altro canto, questo sistema offre elasticità d'uso, un discreto grado di affidabilità ed una velocità operativa dalle 50 alle 200 volte superiore a quella dei nastri.

Tutti i microcomputer forniti di unità a dischi sono anche dotati di un qualche tipo di sistema operativo per dischi (o brevemente DOS, da Disk Operating System) i cui comandi permettono di accedere a tutta una serie di operazioni sui dischetti. Una fra queste è la cosiddetta "formattazione" del dischetto: prima di venir usato per le normali operazioni di archiviazione, un dischetto deve essere "preparato", registrandovi sopra alcune informazioni di sistema, quali, ad esempio, la separazione tra le varie tracce. Il numero di tracce impiegato varia da un fabbricante all'altro di computer ed è normalmente compreso, per ogni facciata, tra 35 ed 80. Ogni traccia è ulteriormente suddivisa in settori (il cui numero, per traccia, varia proporzionalmente alla distanza dal bordo esterno del dischetto). A sua volta, ciascun settore è suddiviso in blocchi, generalmente di 256 byte.

Il DOS "ricorda" dove vengono memorizzati tutti i dati, mediante l'impiego di una mappa di disponibilità, o BAM (Block Availability Map) e di un elenco dei nomi relativi ai file (vedere, a tal proposito, quanto detto a pag. 324).

La possibilità di usufruire delle informazioni concernenti la distribuzione dei dati sul dischetto, e quella di posizionare opportunamente la testina di scrittura/lettura consentono di usare un accesso del tipo "casuale". Ciò significa che un particolare dato può essere scritto (o letto) all'interno di un file, senza dover necessariamente scrivere (o leggere) l'intero archivio.

1. CUSTODIA PROTETTIVA
2. FESSURA DI PROTEZIONE
3. SETTORE
4. FORO DI RIFERIMENTO
5. TRACCIA
6. FESSURA DI ACCESSO



DAVID WEEKS

Attualmente, le dimensioni più comuni per i dischetti sono 5 1/4 pollici (minifloppy) ed 8 pollici (floppy disk). La capacità dei dischetti varia, ovviamente, a seconda delle dimensioni, ma anche di altri elementi, quali la "densità" delle informazioni registrate (che può essere singola, doppia o addirittura quadrupla nei sistemi più recenti), il fatto che vengano utilizzate ambedue le facciate del disco o una sola ed, infine, le caratteristiche intrinseche del sistema operativo. Indicativamente, i minifloppy possono contenere da 100 a 400 Kbyte, anche se alcune particolari marche di computer riescono ad utilizzare al massimo la superficie del disco, portando la capacità ad oltre 1 Megabyte. Per i floppy da 8 pollici, invece, si va da 400 Kbyte a 1,2 Megabyte. Anche in questo caso viene fatto largo uso di sigle per esprimere sinteticamente le diverse caratteristiche del disco: SS sta per Single Sided (Singola faccia), DS per Double Sided (Doppia faccia), SD per Single Density (Singola densità) e DD per Double Density (Doppia densità).

Cosa c'è in un floppy disk?

I floppy disk sono realizzati in Mylar, uno speciale materiale plastico rivestito, a sua volta, con uno strato magnetizzabile di ossidi metallici.

Il dischetto vero e proprio è racchiuso in una custodia plastica, di forma quadrata ed una fessura oblunga, su ciascun lato, consente alla testina di lettura/scrittura di avvicinarsi alla superficie del disco.

I floppy devono essere maneggiati con cura, onde evitare danni alla superficie del disco

Ant Attack: impariamo da un gioco

Un gioco di enorme successo, sviluppato con criteri decisamente fuori dal comune

Gli autori e le case produttrici di software, in ogni parte del mondo, tentano qualsiasi strada per ottenere la protezione legale contro la riproduzione "piratesca" dei loro prodotti. La vigente legislazione, infatti, non offre alcuna reale garanzia, come accade invece, per esempio, nel caso della produzione letteraria o industriale. Sandy White, l'autore di questo gioco, è ricorso ad uno stratagemma molto ingegnoso: non potendo brevettare il programma in sé, egli ha brevettato l'algoritmo (ossia la formula matematica) usato per realizzare il gioco.

Che cosa differenzia, quindi, l'*Ant Attack* della Quicksilva, da tutti gli altri giochi che, normalmente, non richiedono l'impiego di complessi algoritmi?

Ant Attack, in primo luogo, non deriva dal tipo di giochi diffusi nelle "sale giochi", prodotti, in gran parte, da case specializzate quali l'Atari o la Taito. Infatti, *Ant Attack* è stato concepito da uno studente dell'Edinburgh College of Arts, dichiaratosi totalmente estraneo al mondo delle sale giochi. Senza alcuna precedente esperienza nella stesura di software ricreativo, Sandy White ha condotto una sola "ricerca di mercato", cioè ha fatto una piccola inchiesta, presso amici e conoscenti, sulle loro preferenze in merito.

Sorprendentemente, la Sinclair Research, alla quale White aveva fiduciosamente inviato una versione su videonastro a scopo dimostrativo, non mostrò alcun interesse per il programma.

La prima caratteristica insolita di questo gioco è rappresentata dal fatto che il giocatore può scegliere il sesso del protagonista dell'avventura, dopo di che il gioco diviene immediatamente incalzante ed inizia con un'accorata invocazione d'aiuto, diretta all'"eroe".

Costui (o costei!) dispone di una limitata quantità di munizioni per debellare le gigantesche formiche, che lo (o la) inseguono attraverso un complicatissimo labirinto. Purtroppo, non si sa se volutamente o meno, i proiettili o le bombe non causano sempre i medesimi effetti sull'avversario, lasciando una certa insicurezza sull'esito del combattimento. Gli spostamenti del protagonista, in senso orario o antiorario, avvengono mediante la pressione del tasto M e Symbol Shift dello Spectrum. Com'è noto, la tastiera di questo apparecchio non è tra le migliori e talvolta l'imprecisione dei movimenti ha conseguenze veramente frustranti per il giocatore.

Il programma, evidentemente, è stato creato prima che venisse prodotta l'Interfaccia 2 della Sinclair che, permettendo l'uso di due joystick, faciliterebbe enormemente il controllo sui movimenti.

Oltre che per spostare l'immagine grafica del protagonista, il giocatore dispone anche di comandi per sparare le munizioni (scegliendo tra quattro gittate) e per selezionare uno tra quattro "punti di vista", tutti riferiti, ovviamente, alla posizione del protagonista.

1



1. Mio eroe!

Nella prima scena del gioco, la "vittima" si trova appena al di là della cinta di mura di una città. Il protagonista (che può essere maschio o femmina) viene accolto da un'accorata invocazione d'aiuto

2. Attenzione alle formiche

Le giganti e bellicose formiche, fortunatamente, non possono scalare i gradini. Il protagonista può dunque mettersi in salvo e, da quella posizione, sparare le proprie munizioni, ma occorre tener d'occhio anche l'orologio: il tempo a disposizione è limitato

2



per concessione della SOFT

IAN McKINNELL

È proprio questa parte del programma che lo rende unico nel suo genere: gli effetti grafici tridimensionali sono pressoché istantanei, nel passaggio da una scena all'altra, e superano di gran lunga le prestazioni di qualsiasi altro gioco, che non impegni più di 48 Kbyte di memoria. La possibilità di modificare il punto di vista è essenziale allo svolgimento del gioco e l'autore del programma, comprensibilmente, non è disposto a svelare tanto facilmente i segreti che permettono una così efficiente gestione delle immagini. La sola indicazione deducibile è che il campo di gioco non viene codificato, come sarebbe logico pensare, in una matrice di 128 × 128 × 6 elementi, che non offrirebbe un'adeguata elasticità di programmazione.

Ma vediamo quale sia lo svolgimento del gioco, ambientato nella città di Antescher (detto per inciso, il nome deriva dalla combinazione di "Ant" = "formica" in inglese, ed Escher, che è il nome dell'artista noto per le sue ingegnose rappresentazioni di strutture impossibili a realizzare). Il protagonista, che inizialmente si trova al di fuori delle mura della città, si avventura in essa nel tentativo di salvare il personaggio che ha lanciato invocazioni d'aiuto. Sul percorso sono collocati numerosi ostacoli, da scalare o da aggirare. La città viene rappresentata in proiezione isometrica e non tiene conto, quindi, di alcuna prospettiva reale.

Durante le varie fasi del gioco, sullo schermo viene riprodotta soltanto una piccola porzione della città, come se si trattasse di una "finestra" mobile (assieme al protagonista). Lo scroll (ossia lo spostamento dell'immagine nelle quattro direzioni) è molto efficace ed eccellente è pure l'effetto globale di animazione.

La città è popolata da una gran quantità di grosse formiche i cui morsi, benché non immediatamente letali, possono condurre alla morte del protagonista. Se una formica si avvede della presenza del protagonista, inizia un accanito inseguimento e le possibilità sono due: riuscire a "seminare" l'inseguitore oppure ricorrere alle poco affidabili munizioni (facendo attenzione a non colpire qualche muro posto nelle vicinanze, altrimenti si salta in aria!).

Nella scena iniziale del gioco, l'immagine della vittima da salvare appare appena al di là dell'ingresso alla città ma, col passar del tempo, diviene sempre più difficile individuarla e raggiungerla. Esistono più "piani" o "livelli" nel gioco, ma è consentito passare da uno all'altro solamente in modo graduale; se la vittima si trova ad un livello superiore e non ci sono, nelle vicinanze, scale o strutture da scalare, il protagonista si trova nei guai. Una soluzione può essere quella di attirare l'attenzione delle formiche, catturarne una e montare su di essa per passare al livello superiore.

Il gioco termina quando il protagonista e la vittima riescono, finalmente, ad uscire incolumi dalla città.



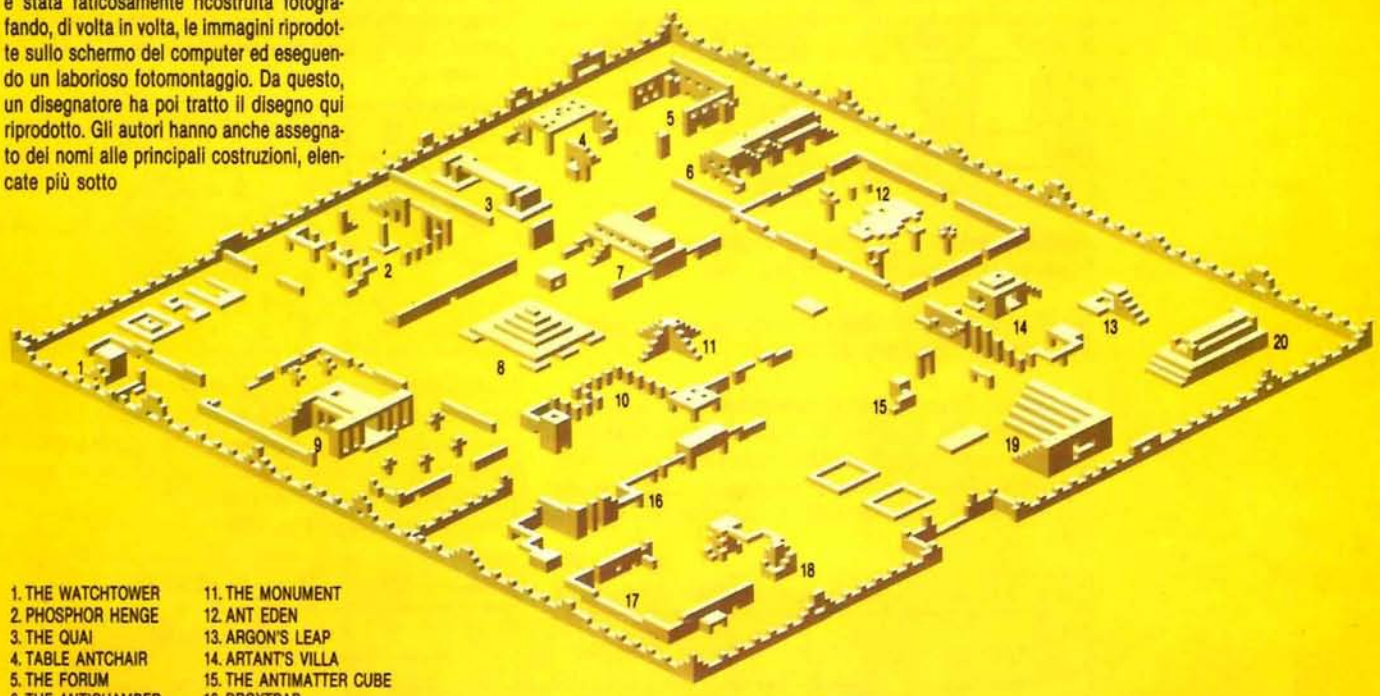
IAN McKINNELL

L'autore di Ant Attack

Questo gioco è stato il primo programma ricreativo scritto da Sandy White, appena 23enne quando iniziò a commercializzarlo. White è diplomato in scultura presso l'Edinburgh College of Arts ed una sua amica, Angela Sutherland, ha collaborato alla creazione delle strutture che formano la città di Antescher

UN LABIRINTO NEL DESERTO

Questa mappa dell'intera città di Antescher è stata faticosamente ricostruita fotografando, di volta in volta, le immagini riprodotte sullo schermo del computer ed eseguendo un laborioso fotomontaggio. Da questo, un disegnatore ha poi tratto il disegno qui riprodotto. Gli autori hanno anche assegnato dei nomi alle principali costruzioni, elencate più sotto



1. THE WATCHTOWER
2. PHOSPHOR HENGE
3. THE QUAI
4. TABLE ANTCHAIR
5. THE FORUM
6. THE ANTICHAMBER
7. SKAZ YANDOR
8. THE PYRAMID
9. THE ANCIENT
10. OXYMINE
11. THE MONUMENT
12. ANT EDEN
13. ARGON'S LEAP
14. ARTANT'S VILLA
15. THE ANTIMATTER CUBE
16. DROXTRAP
17. ADRIANT'S WALL
18. BONZAI WALK
19. THE SQUARENA
20. THE CRYPT

per concessione della YOUR SPECTRUM

KEVIN JONES

L'algebra di Boole

I computer ragionano con una logica basata su due soli elementi: lo zero e l'uno

L'algebra di Boole, un ramo della matematica che concerne la logica "vero/falso", è la base teorica sulla quale viene realizzata l'intera architettura interna di un computer. I concetti e le regole di questa materia sono pochi e facilmente comprensibili.
Vedremo come queste poche nozioni si applichino, nella pratica, ai circuiti dei calcolatori. Le regole, nell'algebra di Boole, si basano su tre semplici operazioni logiche, chiamate AND (= "e" congiunzione), OR (= "oppure") e NOT (= negazione), dal significato del tutto simile a quello impiegato nel normale linguaggio. Consideriamo la seguente frase:

Se è Domenica **e** la giornata è bella, allora Marco va a passeggiare.

Il fatto che Marco vada a passeggiare dipende da due condizioni, che devono verificarsi contemporaneamente: essere Domenica ed essere una bella giornata. In altri termini, si può dire che l'azione viene intrapresa soltanto se sono vere ambedue le condizioni. Esistono quattro possibili combinazioni, che si possono riassumere in una "tavola della verità" per l'operazione logica AND:

È BELLO	È DOMENICA	MARCO VA A PASSEGGIO
FALSO	FALSO	FALSO
FALSO	VERO	FALSO
VERO	FALSO	FALSO
VERO	VERO	VERO

Analogamente, possiamo descrivere la funzione dell'operazione logica OR:

Se Guido **oppure** Mario vanno al mare, allora Marco può andare al mare.

Di nuovo, le condizioni determinanti, affinché Marco possa andare al mare, sono due, ma questa volta è sufficiente che sia vera o l'una o l'altra. Possiamo così costruire una tavola della verità anche per l'operazione logica OR:

GUIDO VA AL MARE	MARIO VA AL MARE	MARCO VA AL MARE
FALSO	FALSO	FALSO
FALSO	VERO	VERO
VERO	FALSO	VERO
VERO	VERO	VERO

La terza operazione logica (NOT) è addirittura la più semplice di tutte. Si consideri la frase:

Se **non** è buio, allora usciamo di casa.

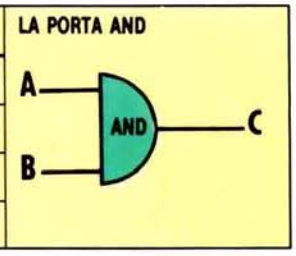
Questa volta esiste un'unica condizione da considerare: che non sia buio e le possibilità da riassumere nella tavola della verità sono soltanto due:

È BUIO	USCIAMO
FALSO	VERO
VERO	FALSO

Le porte logiche

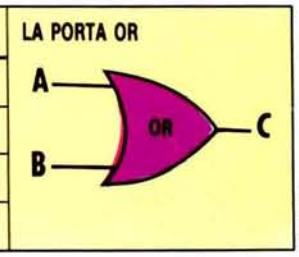
Abbiamo, in parte, già accennato al funzionamento delle porte logiche (vedere a pag. 68 e pag. 92) spiegando come i segnali elettrici, interpretati come livelli logici "1" o "0", consentano di applicare l'algebra di Boole ai circuiti di un computer.
Riassumiamo brevemente, sotto forma di tavole della verità, il funzionamento dei vari tipi di porte logiche.
La tavola della verità ed il diagramma di una porta AND a due ingressi (A e B) ed un'uscita (C) sono:

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1



La funzione della porta AND può essere così riassunta a parole: «L'uscita vale 1, se valgono 1 ambedue gli ingressi, altrimenti vale 0».
La notazione matematica relativa a questa operazione è A.B (A punto B).
Vediamo una porta OR:

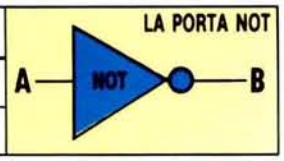
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1



LIZ DIXON

A parole, possiamo dire: «L'uscita vale 1 se i due ingressi, o uno solo di essi, vale 1». In notazione matematica espresso con A+B.
Diversamente dalle precedenti, la porta NOT ha un solo ingresso:

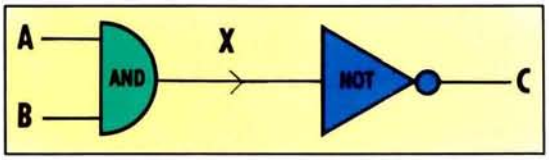
A	B
0	1
1	0



ed il funzionamento si può descrivere con: «Il valore in uscita è opposto al valore d'ingresso». In simbologia matematica: $\overline{A}$.

Combinazioni di porte logiche

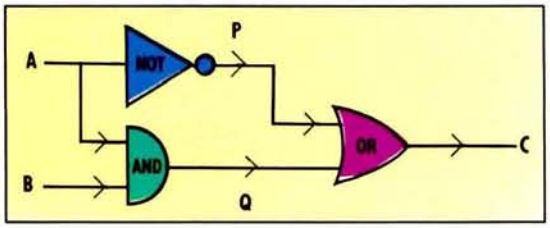
Così come è possibile concatenare diverse singole frasi per comporre un discorso, similmente le porte logiche vengono collegate l'una all'altra per svolgere funzioni più complesse, fino a quelle necessarie al funzionamento di un computer. Ad ogni configurazione di porte, così concepita, si può associare una tavola della verità che ne riassuma le funzioni. Ad esempio, consideriamo il seguente circuito logico:



In esso, vi sono due ingressi, A e B, ed una sola uscita, C. Per facilitare la costruzione della tavola della verità, identifichiamo con X l'uscita della prima porta ed otteniamo:

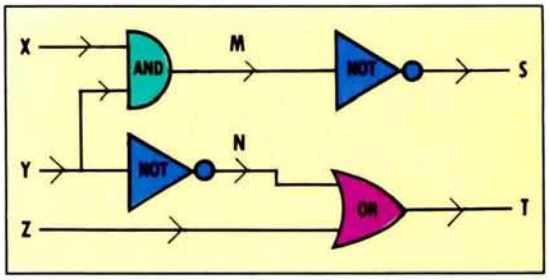
A	B	X	C
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

L'uscità della porta AND viene applicata ad una porta NOT, ottenendo i risultati riportati nella colonna C.
Ecco un circuito un poco più complesso, assieme alla relativa tavola della verità. Si noti che vi sono solo due ingressi e che il numero di combinazioni è, pertanto, di nuovo quattro. La seconda parte della tavola (colonne P, Q e C) è una variazione della tavola OR.



A	B	P	Q	C
0	0	1	0	1
0	1	1	0	1
1	0	0	0	0
1	1	0	1	1

L'uso di tavole della verità non si limita, però, ai circuiti con due soli ingressi ed una sola uscita.
Ecco lo schema di un circuito con tre ingressi e due uscite:

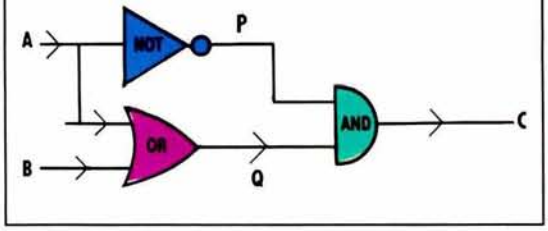


Il numero di combinazioni sale così ad otto:

X	Y	Z	M	N	S	T
0	0	0	0	1	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	0	0	0
1	1	1	0	1	0	1

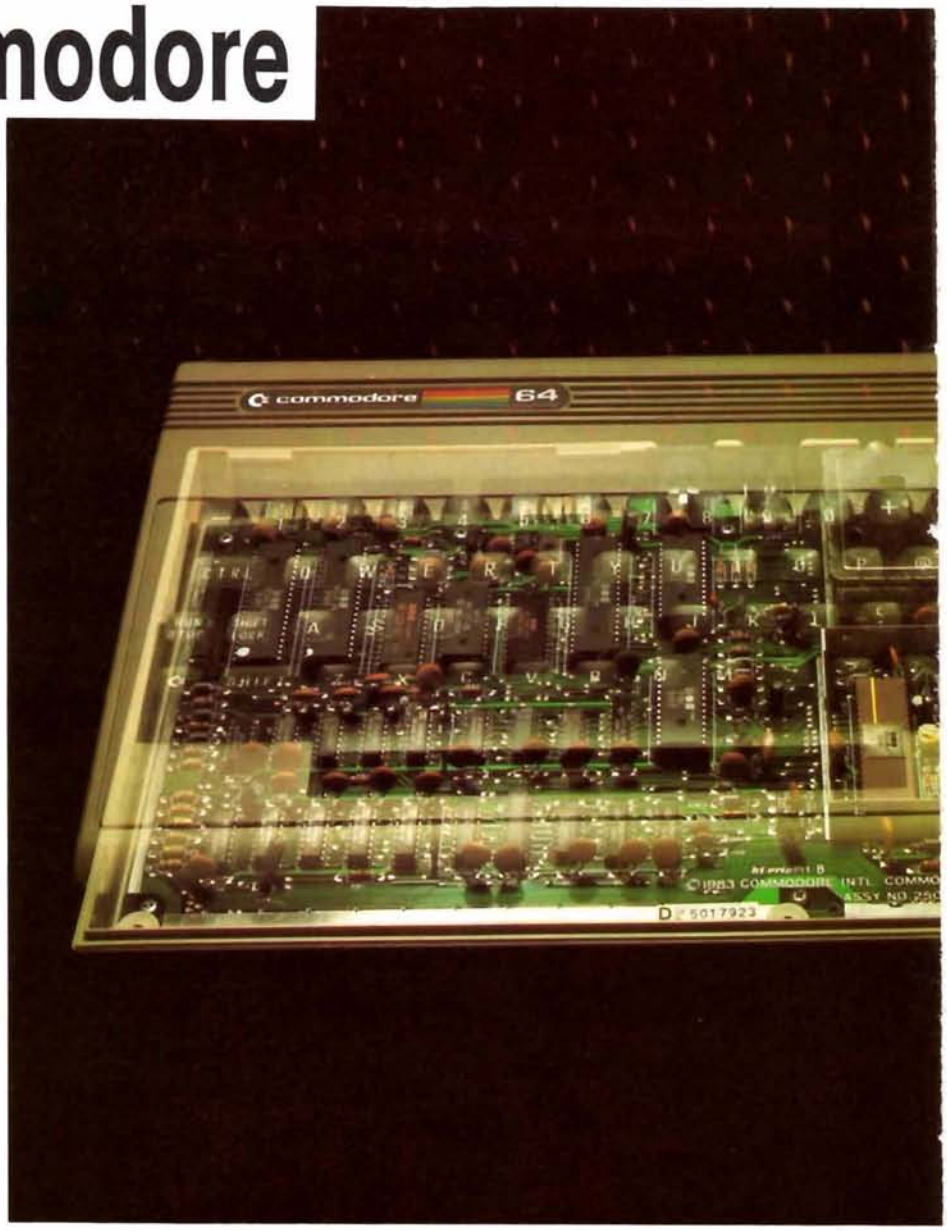
ESERCIZIO 1

1. Si costruisca una tavola della verità per la seguente situazione: "Gianni può guidare un'auto se possiede una patente o se è accompagnato da un possessore di patente".
2. Si costruisca una tavola della verità per la seguente situazione: "Si può caricare un programma nel computer se disponiamo di un registratore o di un'unità a dischi e se il programma non è scritto per un altro computer".
3. Si costruisca una tavola della verità per il seguente circuito logico:



Novità Commodore

La Commodore ha lanciato recentemente una nuova versione del famosissimo C64: un confronto tra i due modelli



La Commodore, dopo le prime esperienze con la serie PET, si è specializzata anche nella produzione di home computer, un ramo nel quale ha incontrato enormi favori da parte del pubblico. Apparecchi quali il Vic-20 ed il Commodore 64 fanno ormai parte della storia, segnando l'avvento del computer a grandissima diffusione. Il successo è stato indubbiamente determinato dall'aver offerto, a prezzi decisamente concorrenziali, apparecchi dotati di funzioni grafiche ed audio eccezionali (se paragonate a quelle di apparecchi dello stesso genere), adattissimi sia ad applicazioni didattiche che ricreative. Uno degli ultimi "nati", in casa Commodore, porta la sigla SX-64 e costituisce il "fratello" portatile del C64.

Ma procediamo con ordine, confrontando le caratteristiche del Vic-20, del C64 ed infine dello SX-64.

Una prima considerazione da fare riguarda il chip impiegato nei tre apparecchi: si tratta del famosissimo 6502, per il Vic-20, del meno noto 6510 per gli altri due. A livello di istruzioni, in codice macchina, questi due chip sono identici: la sostanziale differenza consiste nella capacità del 6510 (dovuta ad una diversa struttura interna) di poter gestire più di un banco di memoria. Infatti, una delle maggiori

IL PORTATILE SX-64

Questo apparecchio, prodotto ultimamente dalla Commodore, è pressoché identico al C64 (stessa unità centrale, capacità grafiche e sonore). Tuttavia, è alloggiato in un comodo contenitore, dall'eccellente design, che lo rende facilmente trasportabile da un luogo all'altro.

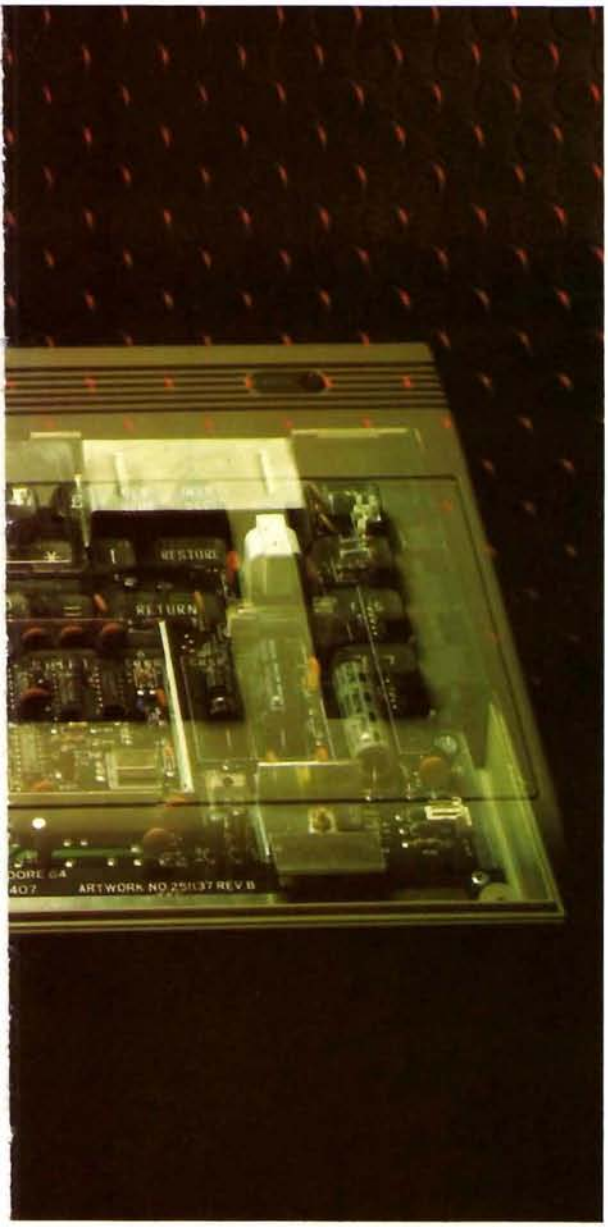
Nella versione di base è incorporato un monitor a colori da 5 pollici ed un'unità a dischetti. La tastiera è comodissima, è dotata di un ottimo "tocco" e su ciascun tasto è raffigurato anche il corrispondente simbolo grafico. Durante il trasporto, la tastiera si aggancia all'involucro principale dell'apparecchio, formando un tutt'uno molto solido e compatto.

Un aspetto molto importante da considerare è la completa compatibilità di software (sia su dischetti che su cartucce ROM) tra questo modello ed il Commodore 64



SX-64 per concessione della COMMODORE

IAN McKINNELL



IAN McKINNELL

limitazioni di tutti i chip ad 8 bit è proprio quella di poter accedere ad un massimo di 64 Kbyte di memoria. Normalmente, parte dei 64 Kbyte sono necessariamente occupati dalle memorie ROM, lasciando (come nel caso del Vic-20) ben poco spazio ai programmi applicativi.
Il principale campo d'applicazione del Vic-20, considerata la scarsità di RAM (salvo impiegare le schede d'espansione, che aumentano la capacità a 32 Kbyte) è quello dei giochi e della didattica. Il C64, immesso sul mercato poco dopo il Vic-20, è dotato, invece, di ben 64 Kbyte di RAM e possiede anche una pregevole grafica "a sprite" e la possibilità di generare effetti sonori, ivi compresi interi brani musicali a tre voci. La maggior quantità di memoria del C64, inoltre, permette di usare proficuamente anche le unità a dischetti, oltre ai normali nastri "a cassetta". Questo secondo apparecchio, benché adattissimo, come il Vic-20, alla didattica ed ai giochi, viene anche impiegato per piccole attività commerciali, avendo un costo molto accessibile.
Tre altri chip distinguono il C64 dagli apparecchi della medesima categoria: sono quelli che sovrintendono al funzionamento degli apparecchi periferici (il 6526), alla gestione delle funzioni grafiche del video (il 6566) ed alla sofisticata generazione dei suoni (il 6581).
Il chip 6526 è anche chiamato CIA (Complex Interface Adaptor, ossia adattatore complesso di interfacce) ed è una versione migliorata dei PIA (Peripheral Interface Adaptor, ovvero, adattatore di interfacce per periferiche), finora più diffusi. Esso contiene, oltre alle normali linee programmabili di I/O, dei circuiti temporizzatori e dei registri "a scorrimento", per la conversione dei segnali da seriali a paralleli. Inoltre, questo chip è dotato anche di un orologio con sveglia incorporata, ma non viene spiegato, nei manuali, come accedervi dal BASIC.
Il chip 6566 gestisce la grafica del Commodore, ed è un parente stretto del Video Interface Chip (Chip d'interfaccia video), dal quale prende nome il Vic-20.
Infine, alla generazione degli effetti audio provvede il chip 6581, detto anche SID (da Sound Interface Device, ossia dispositivo d'interfaccia audio). Con un poco di pratica, si riescono ad ottenere dei suoni molto fantasiosi, per allietare, magari, le varie fasi di un gioco.

Il software per i Commodore

Una vasta quantità del software per i Commodore 64 e SX-64 è stata ereditata dai modelli precedenti, il PET ed il Vic-20. Il BASIC è pressoché identico in tutta la gamma di apparecchi, al pari di molte caratteristiche del sistema operativo dei dischetti.
Per le applicazioni commerciali esiste un'ampia scelta di programmi per l'elaborazione di testi (word processor), alcuni dei quali dotati di funzioni per il controllo ortografico (appli-

UNA "PAGINA" DEI COMMODORE

Costruita con il programma Magpie, un generatore di applicazioni che permette la creazione di programmi per la gestione di banche dati



IAN McKINNELL

COMMODORE 64

DIMENSIONI

mm. 404 x 216 x 75

PESO

gr. 1820

CPU

6510

CLOCK

1 MHz

MEMORIA

64 Kbyte di RAM dei quali 39 Kbyte per programmi in BASIC. 20 Kbyte di ROM (incluso il generatore di caratteri)

VIDEO

25 linee di 40 caratteri. In bassa risoluzione, sono selezionabili 16 colori da tastiera per i caratteri, lo sfondo ed il bordo delle immagini. Risoluzione massima: 320 x 200 pixel. Si possono usare contemporaneamente fino ad 8 sprite

INTERFACCE

Per 2 joystick, per periferiche parallele e seriali, per monitor, per TV. Porta seriale RS232 (adattatore escluso), ingresso ed uscita audio

LINGUAGGI FORNITI

BASIC

ALTRI LINGUAGGI DISPONIBILI

FORTH, LOGO, Assembly 6502

TASTIERA

Tipo macchina da scrivere, dotata di tasti programmabili

DOCUMENTAZIONE

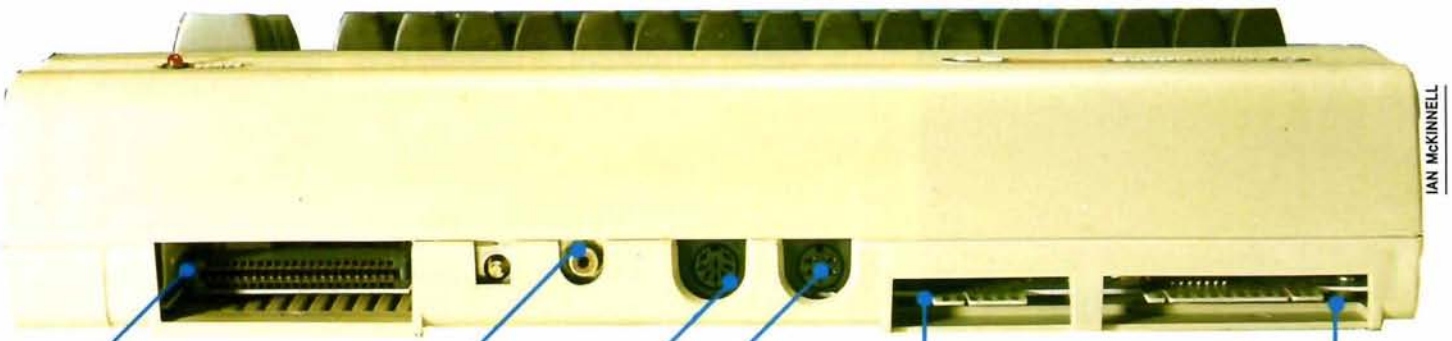
I manuali sono piuttosto soddisfacenti, ma per chi desideri approfondire la propria conoscenza sulle notevoli capacità di questo apparecchio, esistono in commercio numerosi testi dedicati appositamente all'argomento

PRO

Una buona quantità di memoria RAM. Grafica a sprite. Generatori di suoni molto sofisticati. Tastiera di buona qualità. Si presta bene anche per piccole applicazioni commerciali

CONTRO

Occorre usare un'apposito registratore (quelli normali non sono adatti). Il BASIC è carente di alcuni comodi comandi. La scelta dei livelli di risoluzione non è molto ampia



IAN McKINNELL

Connettore per cartucce

Quando si inserisce una cartuccia ROM, la parte di memoria (fino a 16 Kbyte) che essa sostituisce viene automaticamente selezionata dall'apparecchio. Inoltre, qualora i primi nove byte della ROM contengano una particolare sequenza, il computer provvede ad una partenza automatica del programma contenuto nella cartuccia

Uscita video TV

Diversamente dal Vic-20, nel C64 è presente un modulatore RF, per pilotare direttamente un normale televisore domestico

Presa Audio/Video

Su questa presa è disponibile un segnale composito, per pilotare un monitor a colori (sebbene non del tipo RGB). Esiste anche un'uscita audio, per riprodurre su di un impianto hi-fi i suoni generati. Infine, è presente un ingresso per miscelare suoni esterni con quelli generati dal computer

Bus seriale

Questa è una speciale interfaccia, progettata dalla Commodore, per pilotare gli apparecchi periferici (stampanti ed unità a dischi). Il protocollo di trasmissione è simile allo standard IEEE-48, ma viene impiegato un canale seriale al posto di quello parallelo

Connettore per registratore

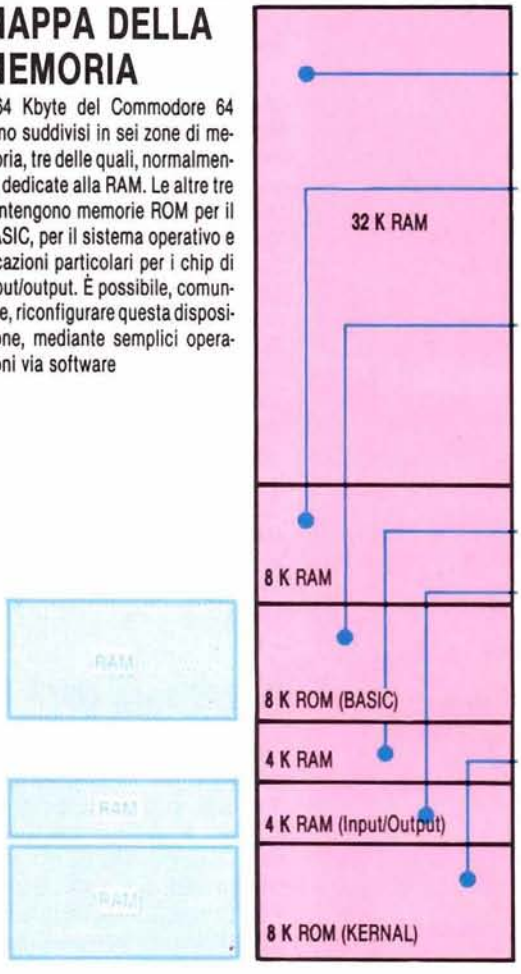
Si possono collegare soltanto registratori a cassette appositamente progettati dalla Commodore. Molti acquirenti, comunque, passano presto alle più comode unità a dischetti

Porta per espansioni

Lo scopo di questo connettore è duplice: può servire da porta seriale RS232, benché sia necessario in questo caso servirsi di un adattatore, oppure come porta parallela per l'uso di apparecchi adatti alla sperimentazione

MAPPA DELLA MEMORIA

I 64 Kbyte del Commodore 64 sono suddivisi in sei zone di memoria, tre delle quali, normalmente, dedicate alla RAM. Le altre tre contengono memorie ROM per il BASIC, per il sistema operativo e locazioni particolari per i chip di input/output. È possibile, comunque, riconfigurare questa disposizione, mediante semplici operazioni via software



Zona 1
Contiene sempre 32 Kbyte di RAM

Zona 2
Normalmente contiene 8 Kbyte di RAM, che viene però disabilitata, inserendo una cartuccia ROM da 8 Kbyte

Zona 3
Normalmente, contiene 8 Kbyte di ROM, occupata dal BASIC. Disabilitata se si impiega una cartuccia da 16 Kbyte. È possibile anche rimpiazzarla con memoria RAM

Zona 4
Contiene sempre 4 Kbyte di RAM

Zona 5
Normalmente contiene le locazioni dedicate ai chip di input/output, ma è possibile trasferirvi il generatore di caratteri ROM con operazioni via software

Zona 6
All'accensione, questa zona contiene 8 Kbyte di ROM, nella quale risiede il Kernal (un sistema operativo di minime dimensioni). Usando il linguaggio macchina, si può rimpiazzare questa zona con 8 Kbyte di RAM

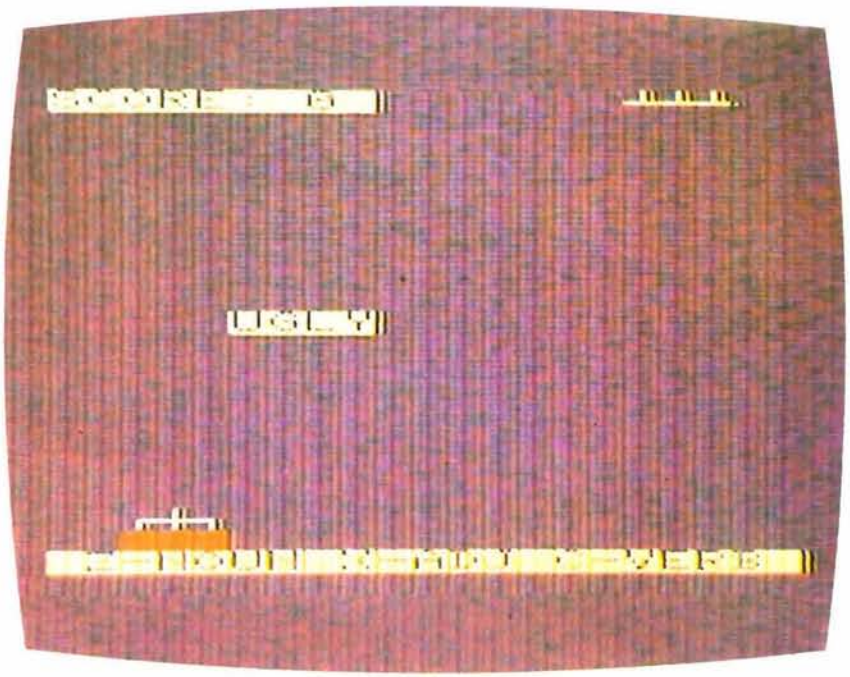
cabili, però, nella sola lingua inglese): l'Easy-Write e EasySpell (sviluppati dalla stessa Commodore), il VisaWrite/VisaSpell, il Paperclip 64 ed il Wordcraft 40. Quest'ultimo differisce dagli altri in quanto il formato del testo visualizzato sullo schermo viene riprodotto con assoluta fedeltà nella stampa. Gli altri programmi, invece, intercalano ai caratteri del testo alcuni codici di controllo (usati per specificare le caratteristiche del formato di stampa) ed i testi prodotti possono essere stampati soltanto usando il programma che li ha generati.

Anche per le cosiddette "lavagne elettroniche" (spreadsheet) la scelta è vasta: da programmi elementari, dal costo di poche decine di migliaia di lire, a sofisticati "pacchetti", meno economici, ma dalle prestazioni notevoli. Uno di questi è il CalcResult, nel quale vengono sfruttate appieno le capacità grafiche a colori, anche per la generazione automatica di istogrammi. I dati vengono raccolti in varie "pagine" della lavagna e possono essere comodamente confrontati o sommati tra loro.

Software in vetrina

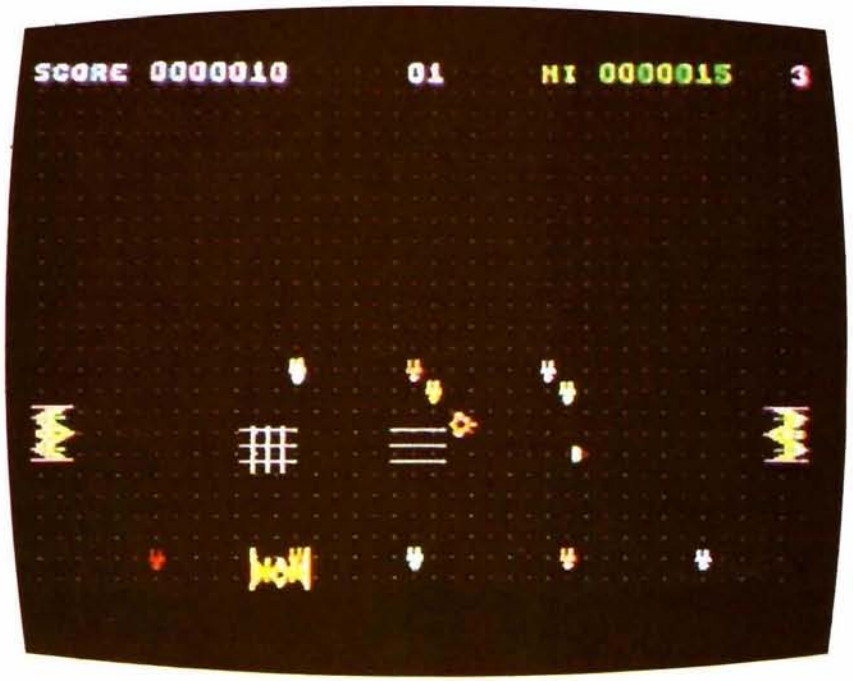
Una selezione tra i numerosissimi programmi disponibili per i computer più diffusi

In commercio esistono migliaia di programmi, dai più economici giochetti per home computer, ai più sofisticati sistemi per applicazioni commerciali. Mentre questi ultimi, per la complessità delle loro funzioni, vengono esaminati singolarmente in altre parti del corso, per i primi può bastare una concisa descrizione: iniziamo col presentare due giochi, uno per il Commodore 64 ed uno per il Vic-20.



English Invaders per il Vic-20

Questo gioco appartiene al genere degli "Invasori spaziali", ma ne costituisce una variante molto curiosa: i proiettili lanciati dagli alieni sono... parole! Infatti, il gioco è stato concepito, sia pure ad un livello molto elementare, per divertire, ma allo stesso tempo per insegnare. Il giocatore, per difendersi, deve indicare se il "proiettile" è un verbo, un sostantivo od un aggettivo. Il fatto che la lingua usata (come si arguisce dal nome del gioco) sia l'inglese potrebbe stimolare chi, tra i più giovani, la sta studiando a scuola. Un passatempo utile, tutto sommato!



Quintic Warrior per il Commodore 64

Prodotto dalla Quicksilva, la stessa software house che distribuisce l'*Ant Attack* (vedere a pag. 486), questo programma appartiene alla categoria di giochi cosiddetti "a griglia". Il gioco è accattivante, anche se gli "alieni" sembrano tanti piccoli diapason. Sono previsti più livelli di gioco, ben equilibrati tra difficoltà e divertimento. Colpire gli alieni che procedono verso la navetta del giocatore (la cui immagine è riportata nella parte inferiore dello schermo) ed evitare i raggi laser sparati dal nemico è già abbastanza impegnativo, ma si aggiunge un altro imprevisto: gli alieni possono anche colpire la navetta alle spalle! *Quintic Warrior* è un gioco che può divertire a lungo, senza stancare.

Il BASIC dello Spectrum

Esistono molti dialetti del BASIC: esaminiamo le caratteristiche di quello per lo Spectrum

Iniziamo dai nomi delle variabili, una fonte inesauribile di confusione nel passaggio da un dialetto all'altro del BASIC. Nel BASIC della Sinclair, per le variabili alfanumeriche (o "stringhe"), è consentito usare nomi di una sola lettera, indipendentemente se questa è maiuscola o minuscola. Vale a dire che le stringhe A$ ed a$ condividono la stessa porzione di memoria. Le variabili alfanumeriche ad indice (chiamate anche "matrici") seguono le medesime regole delle variabili semplici: una volta DIMensionata la variabile H$, ad esempio, qualsiasi successiva menzione di questa variabile, nel resto del programma, si riferirà alla variabile indicizzata H$.
Ciò dipende dal fatto che il BASIC Sinclair considera tutte le stringhe come delle matrici, alcune delle quali dimensionate esplicitamente, altre no.
Al contrario, i nomi delle variabili numeriche non sono soggetti ad altrettante limitazioni: devono iniziare con una lettera, ma possono avere una lunghezza a piacere. Il nome può contenere degli spazi, lettere maiuscole o minuscole o un misto delle due. Alcuni nomi, validi, di variabili numeriche sono:

```
qwert, ub40, Il Mio Computer
```

che, per quanto concerne il computer, equivalgono esattamente a:

```
QWERT, UB40, IL MIO COMPUTER
```

I nomi delle matrici devono essere di una sola lettera. Tuttavia, si può tranquillamente adoperare la stessa lettera, simultaneamente in un programma, sia per una variabile numerica ad indice che per una variabile semplice: ossia, la variabile ad indice v(8) e la variabile semplice v sono distinte fra loro. Nei cicli FOR...NEXT è consentito solo l'uso di nomi di variabili numeriche composte da una sola lettera: pertanto, FOR V=1 TO 9 è formalmente corretto, mentre FOR conto=1 TO 9 non lo è.
Una delle principali differenze tra il BASIC Sinclair e gli altri consiste nel diverso modo di trattare le stringhe. Vediamo per primo il caso dell'istruzione DIM. Nel Sinclair, una linea composta da DIM a$(12) provoca l'attribuzione, alla variabile a$, di 12 byte consecutivi di memoria, inizialmente contenenti il carattere spazio. Ciascun byte può essere individuato mediante l'indice della variabile, mentre per riferirsi collettivamente ai 12 byte, si usa semplicemente a$ (senza indice), la cui lunghezza, nel caso in esame, è sempre di 12 caratteri. Supponiamo di scrivere:

```
DIM a$(12):LET a$="123456789"
```

la variabile a$ conterrà i nove caratteri assegnati, seguiti da tre spazi (totale = 12). Scrivendo invece:

```
DIM a$(12):LET a$="ABCDEFGHIJKLMN"
```

la variabile conterrà soltanto i primi dodici caratteri "ABCDEFGHIJKL", ignorando quelli in eccesso a 12. Se adesso scriviamo:

```
LET a$(2 TO 5)="1234"
```

il contenuto di a$ viene modificato in modo da contenere "A1234FGHIJKL", dimostrando quanto efficace sia il BASIC Sinclair nella manipolazione delle stringhe: ogni carattere di una stringa è accessibile mediante l'impiego di un indice. L'uso della DIM differisce sostanzialmente, in questo caso, da quello comune agli altri BASIC. Infatti, generalmente, scrivere DIM a$(12) significa creare 12 diverse stringhe a$(1), a$(2), ecc., ciascuna delle quali ha una lunghezza massima imposta dal particolare BASIC (normalmente 256 caratteri). Se ad uno degli elementi della variabile non viene assegnato alcun valore, allora la sua lunghezza è 0 e contiene un valore nullo, ossia " ".
Ciò significa che, negli altri BASIC, la manipolazione delle stringhe può avvenire soltanto ricorrendo a nuove specifiche funzioni (LEFT$, MID$, RIGHT$, ad esempio).



IAN McKINNELL

IL SUPERBASIC

La principale diversità tra il SuperBASIC, progettato appositamente per il nuovissimo apparecchio della Sinclair, il QL, ed il BASIC impiegato per tutti i modelli precedenti (ZX80, ZX81 e ZX Spectrum), consiste nell'abbandono del sistema "a tasto singolo" per l'immissione dei comandi e delle istruzioni del linguaggio.
Questa insolita caratteristica, parzialmente dovuta alla difficoltà d'impiego delle economiche tastiere della serie ZX, implica l'uso di un apposito tasto per modificare il significato dei tasti

Sinteticamente:

ALTRI BASIC	BASIC SINCLAIR	EFFETTO
B$=LEFT$(A$,N)	B$=A$(TO N)	B$ = N caratteri a sinistra di A$
B$=RIGHT$(A$,N)	B$=A$(LEN A$—N+1 TO)	B$ = N caratteri a destra di A$
B$=MID$(A$,P,N)	B$=A$(P TO P+N—1)	B$ = N dalla posizione P in poi

La funzione INSTR, che serve ad individuare la posizione di una particolare sequenza di caratteri all'interno di una stringa, nel Sinclair si può ottenere, ma non in modo altrettanto diretto, ossia:

```
LET S=INSTR(A$,"stringadiprova")
```

diviene:

```
LET Y$=A$:LET Z$="stringadiprova":GOSUB 9900: LET S=POSN

9900 LET ZL=LET Z$:LET SL=LET Y$—ZL:LET POSN=0
9910 FOR K=1 TO SL
9920 IF Y$(K TO K+ZL—)=Z$ THEN LET POSN=K:LET K=SL
9930 NEXT K:RETURN
```

Si noti che, in questa subroutine, la variabile Y$ viene impiegata come se fosse indicizzata, nonostante non sia stata precedentemente DIMensionata. Il BASIC Sinclair, in tal caso, considera la stringa Y$ di lunghezza variabile, composta da una serie di singoli caratteri, e ad una sola dimensione. È possibile assegnare più di una dimensione ad una variabile alfanumerica, specificando un secondo valore nell'istruzione DIM: ad esempio, DIM Y$(8,7) significa una variabile composta da 8 elementi, ciascuno dei quali ha una lunghezza di 7 caratteri (l'ultimo valore, nella DIM, specifica il numero di caratteri).
Da quanto detto, è dunque necessario prestare molta cura a come vengono impiegate le variabili alfanumeriche nel BASIC Sinclair. Anche linee di programma apparentemente semplici possono, infatti, nascondere un trabocchetto. Ad esempio, se a$ è una variabile semplice, allora LET a$="" assegna ad a$ una stringa nulla (ossia "") e la sua lunghezza è zero. Ma se si è usata, precedentemente nel programma, una DIM a$(7), allora scrivere LET a$="" riempie a$ con 7 caratteri spazio, e la sua lunghezza è 7! Pertanto, se a questo punto tentassimo una verifica del tipo:

```
IF a$ = "" THEN PRINT "stringa vuota"
```

il test fallirebbe, poiché a$ non è affatto vuota: contiene sette spazi!
Desiderando effettuare verifiche di questo genere, è conveniente definire appositamente una stringa, dimensionata opportunamente e quindi contenente soltanto spazi. Ossia:

```
100 DIM a$(12,34)
110 DIM b$(7,56)
120 DIM N$(56)
150 REM N$ viene usata per i confronti
:
580 if b$(3)=N$( TO 56) THEN PRINT "vuota"
590 if a$(11)=N$( TO 34) THEN PRINT "vuota"
:
```

In questo caso, N$ è esclusivamente usata come pietra di paragone, senza la quale i controlli alle linee 580 e 590 fallirebbero, o andrebbero riscritti come:

```
580 if b$(3)="                    " THEN PRINT "vuota"
585 REM      56 spazi tra i doppi apici
```

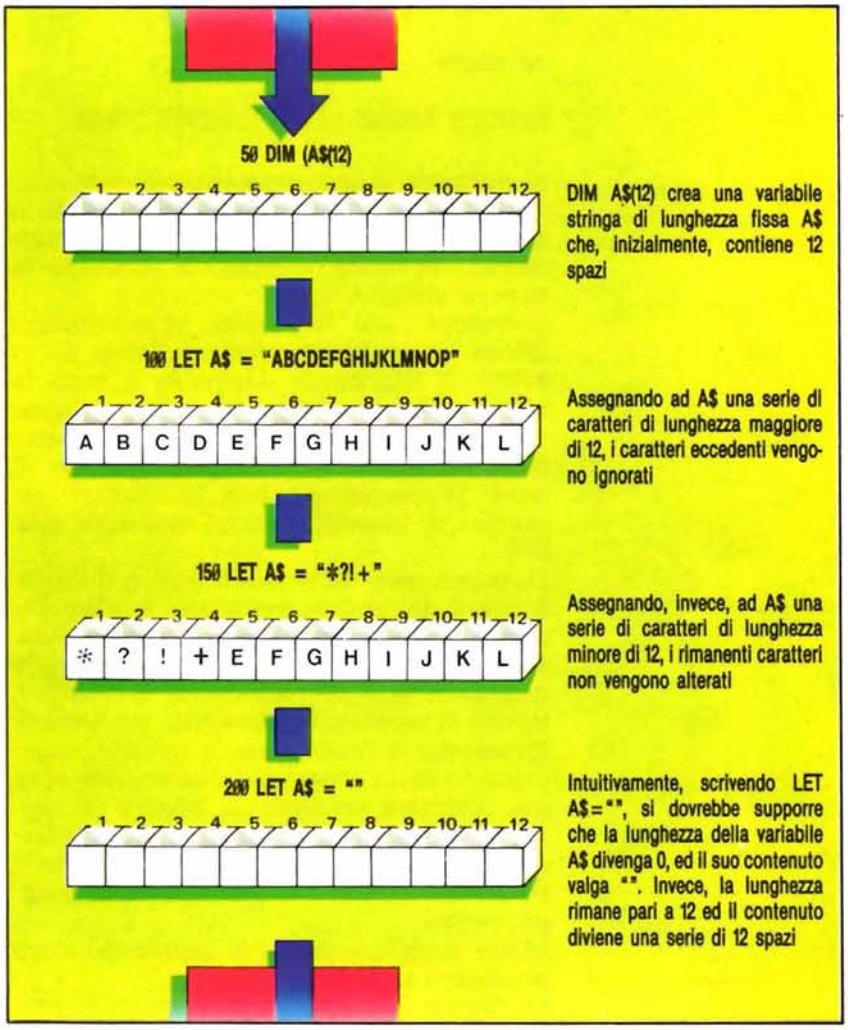
Un simile metodo può facilmente portare a degli errori.
Un altro metodo, da usare però con molta cautela, potrebbe esser quello di assegnare alle variabili una dimensione superiore di un'unità rispetto al valore necessario, cosicché si potrebbe riscrivere la linea 590, ad esempio, come:

```
590 IF a$(11)=a$(12) THEN PRINT "vuota"
```

Ciò suppone, ovviamente, che l'ultimo carattere di a$ non venga mai utilizzato altrimenti nel programma.
Ultimo particolare degno di nota è il valore del primo indice di una matrice. In molti BASIC, tra i quali il Sinclair, il primo elemento utilizzabile ha indice pari ad 1, in altri pari a 0 (ossia è consentito assegnare a$(0) = "stringa").
Nella prossima lezione verranno affrontate altre caratteristiche del BASIC dello Spectrum Sinclair.

Le dimensioni nel BASIC Sinclair

L'effetto dell'istruzione DIM è alquanto diverso da quello che si ottiene nella maggior parte degli altri dialetti del BASIC ed occorre tenerne conto, convertendo un programma stilato per apparecchi diversi dai Sinclair



KEVIN JONES

Gli aspetti di base

In questa prima lezione, affrontiamo alcuni concetti basilari del linguaggio macchina 6502 e Z80

Il linguaggio macchina può presentarsi così:

```
INSTK: SBC $D9FA,X; Uscita segnale dal port
```

oppure così:

```
DE23 FD FA D9
```

o così:

```
11011110 00100011 11111101 11111010 11011001
```

Altre volte, invece:

```
1240 LET ACC = ACC-FLAG(X)
```

od anche:

```
PERFORM AGGIUSTA-FLAG THROUGH CICLO1
```

Nonostante queste numerose forme esteriori, il linguaggio macchina vero e proprio non assomiglia a niente di tutto ciò, trattandosi, in realtà, soltanto di sequenze di segnali elettrici.
Comunque, ciò che viene comunemente inteso come linguaggio macchina è, in effetti, il linguaggio Assembly e tutte le differenti forme appena illustrate non sono che particolari adattamenti, più o meno facili da usare, di un unico concetto di base: rappresentare una particolare sequenza di segnali elettrici mediante simboli.
La prima cosa da tenere a mente, dunque, è che il linguaggio macchina è semplicemente un altro linguaggio di programmazione per computer ed è bene ricordare quanto detto in precedenza per il BASIC: il lavoro di codifica, legato ad un particolare linguaggio, è l'ultimo passo nel complesso sviluppo di un programma ed occorre sempre compiere un'accurata analisi preliminare del problema che intendiamo risolvere, eventualmente tenendo in buon conto le possibilità offerte dal particolare linguaggio scelto.
Molto sinteticamente: il contenuto deve precedere la forma.
La ragione per la quale il linguaggio macchina è chiamato così deriva dalla precisa corrispondenza logica tra le istruzioni mnemoniche del linguaggio e le effettive operazioni svolte dal microprocessore impiegato.
Il linguaggio macchina è indispensabile per tutte quelle applicazioni nelle quali è indispensabile mantenere un rigoroso controllo sulle varie fasi di operazioni molto critiche, quali, ad esempio, il funzionamento di un'apparecchiatura industriale.
Ma il maggior pregio del linguaggio macchina è indubbiamente la velocità d'esecuzione, molto più alta di quella offerta dai linguaggi interpretati. Ciò non significa, però, un risparmio di tempo nella fase di analisi, di codifica, di verifica e di correzione degli errori che, al contrario, possono richiedere molto più impegno che non impiegando linguaggi ad alto livello. Tutto sommato, questi ultimi sono stati creati proprio per facilitare il lavoro dei programmatori!
A livello puramente teorico, la CPU (l'unità centrale di processo del computer) si comporta come un interruttore, che controlla il flusso di corrente nei vari elementi del computer, ossia la memoria, l'unità aritmetica/logica (ALU) e le porte d'ingresso/uscita dei dati.
Premendo un tasto sulla tastiera del computer, si immette una certa informazione, ma dal punto di vista della CPU, non si è che immessa una certa sequenza di impulsi elettrici. La CPU legge la sequenza, la invia ad una parte di memoria, ottenendo in risposta una nuova sequenza di segnali, che vengono, infine, trasmessi al circuito video, che provvede a riprodurre sullo schermo il carattere corrispondente al tasto.
Espresso in questi termini, può sembrare che un computer non sia altro che una macchina per scrivere, dove le connessioni meccaniche vengono sostituite da impulsi elettrici, ma la realtà non è così semplice, poiché l'effetto finale delle operazioni può esser modificato, appunto, programmando la CPU.
Per quanto ci riguarda, comunque, le funzioni della CPU che ci interessano sono classificabili in: operazioni logiche, opera-

zioni aritmetiche, operazioni sulla memoria ed operazioni di controllo.

Le operazioni della CPU

Le operazioni aritmetiche che la CPU può eseguire si riducono, in realtà, alla somma ed alla sottrazione. Quest'ultima, per di più, non è che una variante della somma, nella quale il numero da sottrarre viene cambiato di segno e sommato all'altro operando.
Ovvero:

7+5=12 equivale a (+7) sommato a (+5)

mentre

7—5=2 equivale a (+7) sommato a (—5)

La moltiplicazione e la divisione vengono effettuate operando ripetutamente delle somme e delle sottrazioni sugli operandi.
La CPU può fare, quindi, le quattro operazioni fondamentali, nonostante essa le riconduca tutte a semplici somme o sottrazioni.
Le operazioni logiche, ai nostri fini attuali, possono esser considerate come degli elementari confronti tra due numeri, non solo in termini quantitativi, ma anche, se non soprattutto, nella configurazione dei bit che li compongono.
Che 7 sia maggiore di 5 è facilmente verificabile: perché sottraendo 5 da 7 il risultato è ancora un numero positivo. La CPU è capace di simili confronti, ma è anche capace di stabilire, per esempio, che 189 e 102 hanno il medesimo valore nella colonna delle centinaia.
Per adesso questo fatto ci lascia abbastanza indifferenti, ma vedremo in seguito l'enorme utilità di questo genere di confronto.
La CPU è capace, essenzialmente, di eseguire due operazioni sulla memoria: copiare il contenuto di una locazione di memoria in un "registro" interno alla stessa CPU, e, viceversa, copiare il contenuto di un registro interno in una locazione di memoria.
Queste due funzioni sono necessarie e sufficienti per la gestione della memoria del computer.
Le operazioni di controllo consistono, praticamente, in una serie di decisioni relative alla sequenza con la quale effettuare le operazioni precedentemente descritte. Non è necessario, per il momento, comprendere nei minimi particolari queste funzioni: per adesso, basti sapere che la CPU è in grado di prendere tali decisioni.
La CPU è quindi in grado di: effettuare le operazioni aritmetiche, confrontare numeri, spostare i dati nella memoria e decidere la sequenza di tali operazioni: questo sintetico elenco descrive in modo completo ciò che un computer deve saper fare per portare a termine un qualsivoglia processo di elaborazione.
La corretta sequenza delle operazioni, ovviamente, è impostata mediante un programma per il quale, almeno fino ad oggi, è necessario l'intervento umano.

Se non fossimo ancora convinti che le quattro funzioni appena descritte siano veramente sufficienti alla realizzazione di un elaboratore, potremmo considerare quali siano, ad esempio, quelle richieste da un programma applicativo in BASIC.
In ogni programma BASIC vengono usate delle variabili, che altro non sono che particolari zone di memoria alle quali è stato assegnato, per comodità, un nome.
La maggior parte dei programmi effettua delle operazioni aritmetiche sulle variabili, alle quali seguono, generalmente, dei confronti con i valori di altre variabili. In base al risultato di ciascun confronto, viene decisa la più opportuna sequenza di istruzioni da usare. Generalmente, i dati vengono immessi, dall'operatore, mediante la tastiera ed i risultati visualizzati sullo schermo.
Salvo per quanto concerne l'ingresso e l'uscita dei dati, la descrizione non contiene che le quattro funzioni della CPU, esaminate in precedenza. Se poi accettiamo l'idea che, per la CPU, le operazioni di input/output vengano trattate esattamente come trasferimenti da una zona di memoria all'altra, allora il quadro risulta completo. Di conseguenza, l'esecuzione di un programma può essere descritta come un flusso pilotato di informazioni: immissione, elaborazione ed emissione dei risultati.
A questo punto, però, occorre comprendere cosa sia e come funzioni la memoria di un computer. Benché l'argomento sia già stato trattato (vedere a pag. 58 e pag. 96), è utile ricorrere ad un esempio molto semplice.

Un esempio... logico

Si immagini di avere un circuito composto da una lampadina, un interruttore ed una pila: se l'interruttore viene chiuso, la lampadina si illumina (condizione ON) rimanendo accesa finché non si gira nuovamente l'interruttore, oppure la pila non si scarica completamente (condizione OFF).
Poniamo che l'interruttore venga collocato all'ingresso di una ditta e la lampadina nella stanza del direttore. Il primo dipendente che entra in ditta, gira l'interruttore,

accendendo la lampadina, l'ultimo che esce la spegne. Si può paragonare il circuito ad un sistema di memorizzazione di un'informazione, che in questo caso è rappresentata dalla condizione ON/OFF della lampadina.
In effetti, la memoria di un computer si comporta in modo molto analogo: tutte le informazioni si riducono, in definitiva, alla presenza od all'assenza di elettricità in un circuito.
Sviluppiamo adesso l'esempio, considerando che il circuito sia composto da quattro lampadine ed altrettanti interruttori indipendenti.
Ad ogni dipendente viene assegnata una particolare sequenza di accensione (ad esempio X, al suo arrivo, deve chiudere gli interruttori 1 e 2 ed aprire il 3 ed il 4, mentre Y deve chiudere il 4 ed aprire tutti gli altri; Z, invece, deve aprire il 2 ed il 4 e chiudere gli altri).
Il direttore, a questo punto, oltre a sapere che c'è almeno un dipendente in ditta, può anche individuare quali siano presenti e quali no.
Supponiamo che la posizione OFF di ciascun interruttore corrisponda a "0", e la posizione ON ad "1". Il codice del dipendente X è dunque 1100, quello di Y è 0001 e quello di Z è 1010. Se il direttore vede, nella fila di lampadine, il codice 0001, sa che è entrato il dipendente Y.
Abbiamo stabilito un codice di comunicazione.
Quante combinazioni si possono ottenere con quattro interruttori lo si può facilmente calcolare. Ogni interruttore ha due posizioni, pertanto: $2\times2\times2\times2=16$.
Ovvero:

0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111
1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

Da qualcosa di concreto (gli interruttori) siamo passati ad un concetto astratto (il codice) e se ci spingiamo un poco oltre, arriviamo a convertire questi codici in numeri.
Nella normale numerazione decimale, esistono 10 simboli unici (dallo 0 al 9): è dalla combinazione di essi che possiamo rappresentare i numeri maggiori di 9, ed ogni volta che raggiungiamo il 9 in una colonna, ci serviamo di una nuova colonna, a sinistra, per continuare.
Se adesso immaginiamo di avere soltanto due simboli (lo zero e l'uno), possiamo facilmente contare fino ad 1, ma poi è necessario passare alla colonna successiva e la combinazione 10 corrisponde, ora, al numero 2 decimale. Proseguendo, troviamo che a 3 corrisponde 11, a 4 corrisponde 100, mentre a 5 corrisponde 101 e così via.
In altre parole, se nel sistema decimale il valore di ciascuna colonna viene moltiplicato, rispettivamente, per 1, 10, 100 e via dicendo, nel nostro sistema il fattore di moltiplicazione è, invece, 1, 2, 4, 8...
Ciò significa che, quando noi scriviamo il numero 152, in realtà, questo significa:

$(1 \times 100) + (5 \times 10) + (2 \times 1) = 152$

mentre se scriviamo 101 in binario:

$(1 \times 4) + (0 \times 2) + (1 \times 1) = 5$

Indubbiamente, ragionare secondo questo sistema di numerazione, chiamato "binario", risulta scomodo: è impensabile cimentarsi nel calcolo manuale di una radice quadrata, ad esempio.
Eppure, al microprocessore del computer un simile compito non fa nessun spavento. Anzi, il sistema di numerazione binaria è l'unico che una CPU sia in grado di comprendere: le informazioni, infatti, vengono trasferite mediante segnali elettrici che possono assumere soltanto un livello alto (equivalente a 1) od un livello basso (equivalente a 0).

Adesso possiamo tornare al nostro esempio fatto di lampadine e d'interruttori, azionati dai dipendenti dell'ipotetica ditta. La decisione più ovvia, per scegliere un metodo di codifica, consiste nell'associare a ciascuna combinazione di lampadine accese e spente un numero binario: in questo caso, al dipendente X corrisponde il numero binario 1100 (ossia 12 in decimale), al dipendente Y il numero 0001 (ossia 1 decimale), mentre al dipendente Z il numero 1010 (che vale 10 in decimale). Adesso, osservando via via le combinazioni che appaiono sul pannello delle lampadine, si può risalire a numero decimale corrispondente e cercare, nell'elenco dei dipendenti, il nominativo associato a quel particolare numero.
Si può quindi dire che una serie d'informazioni è stata "memorizzata", usando delle correnti elettriche.
Ovviamente, questo è un puro e semplice esempio teorico, di scarsissima utilità pratica; tuttavia, ci ha permesso di passare, senza grossi traumi, da un fatto pratico (l'ingresso di un dipendente) ad un concetto teorico, quale la numerazione binaria.
Nei computer, tutti i dati (nomi, quantità ecc.) vengono tradotti in particolari combinazioni di segnali elettrici, proprio come nell'esempio delle lampadine, benché a velocità decisamente superiori.

Nella prossima lezione, verrà spiegato come la numerazione binaria serva per rappresentare le informazioni in un computer.

UNA MARCIA IN PIÙ

Questi semplici programmi dimostrano la maggiore velocità del codice macchina rispetto al BASIC. L'intero set dei caratteri viene visualizzato impiegando ambedue i linguaggi. (Nello Spectrum, invece dei caratteri, vengono usati quadratini di colore)

BBC MICRO

```
100 REM*********************************************
110 REM**              BBC PROVA L/M              **
120 REM*********************************************
200 MODE 4:*TV 254
300 GOSUB 30000
700 FOR P=1920 TO 6079
800 K=K+1:IF K)2679 THEN K=1920
900 ?(HIMEM+P)=?(K+47232)
1000 NEXT P
1100 PRINT TAB(13);"QUESTO ERA IL BASIC"
1200 INPUT"PREMERE RETURN PER LA VERSIONE IN
     LINGUAGGIO MACCHINA";A$:CLS
1300 FOR L=0 TO 15:FOR B=0 TO 255 STEP L
1400 ?(SA)=LS:?(SA+1)=HS
1500 ?(FA)=LF:?(FA+1)=HF
1600 DUMMY=USR(PSTRT)
1700 VDU 30
1800 NEXT B,L
1900 STOP
30000 REM** CARICATORE LINGUAGGIO MACCHINA  **
30010 K=1919:PSTRT:=PAGE+8:VSTR:=HIMEM+1920
30020 HS=INT(VSTR/256):LS=VSTR-256*HS
30030 LF=LS+56:HF=HS+2:SA=114:FA=116
30100 DATA 50,169,32,197,112,48,4,240,2,133
30110 DATA 112,165,112,32,227,255,230,114,208
30120 DATA 2,230,115,165,116,197,114,208,7,165
30130 DATA 117,197,115,208,1,96,230,112,169
30140 DATA 128,197,112,208,224,169,32,133,112
30150 DATA 208,218,96,96,96
30160 READ ZZ
30170 FOR BY=PSTRT TO PSTRT+ZZ
30180 READ LM:?(BY)=LM
30190 NEXT BY
30200 RETURN
30299 REM*********************************************
30300 REM**         SALVARE IL PROGRAMMA          **
30301 REM**          PRIMA DI ESEGUIRLO           **
30302 REM*********************************************
30303 REM**        NON LISTARE LA LINEA 100       **
30305 REM**    DOPO L'ESECUZIONE DEL PROGRAMMA    **
30307 REM*********************************************
```

SPECTRUM

```
10 REM*****************************************
11 REM**          SPECTRUM PROVA L/M          **
12 REM**     NON LISTARE LA LINEA 10 DOPO     **
13 REM**      L'ESECUZIONE DEL PROGRAMMA      **
99 REM*****************************************
150 LET PTR=23635
160 LET SA=PEEK(PTR)+(256*PEEK(PTR+1))+7
200 BORDER 2
350 DATA 1,0,3,17,0,88,33,0,0,237,176,201
400 FOR I=0 TO 11
500 READ LM
600 POKE SA+I,LM
700 NEXT I
1000 LET OF=0
1100 FOR I=0 TO 1 STEP 0
1200 POKE SA+7,OF
1300 LET DUMMY=USR SA
1400 LET OF=OF+13
1500 IF OF)=256 THEN LET OF=OF-256*INT(OF/256)
1600 NEXT I
1700 STOP
1799 REM*****************************************
1800 REM**        SALVARE IL PROGRAMMA         **
1801 REM**         PRIMA DI ESEGUIRLO          **
1802 REM*****************************************
```

COMMODORE 64

```
100 REM*****************************************
101 REM**          COMMODORE 64 PROVA L/M       **
102 REM*****************************************
200 PRINT CHR$(147)
300 PRINT "  ATTENDERE PREGO"
400 GOSUB 60000
500 PRINT CHR$(147);CHR$(5):CC=0
600 FOR P=SM TO FM
700 POKE P,CC:POKE P+OF,CL
800 CC=CC+1:IF CC)255 THEN CC=0
900 NEXT P
1000 PRINT TAB(13);"QUESTO ERA IL BASIC"
1100 INPUT "PREMERE RETURN PER LA VERSIONE
     IN LINGUAGGIO MACCHINA";A$
1200 FOR LP=1 TO 9:FOR B=0 TO 255 STEP LP
1300 POKE SA,LS:POKE SA+1,HS
1400 POKE FA,LF:POKE FA+1:HF
1500 POKE BA,B:POKE CH,0:SYS AA
1600 NEXT B,LP
1700 STOP
60000 REM**      CARICATORE LINGUAGGIO MACCHINA  **
60010 SM=256*PEEK(648):OF=55296-SM:FM=SM+999
60020 BD=53280:SC=BD+1::CS=8:CB=6:CL=0
60030 POKE BD,CB:POKE SC,CS
60040 LS=0:HS=PEEK(648):LF=232:HF=HS+3:SA=231
60050 FA=253:BA=250:CH=2
60100 DATA 850,885,169,0,170,165,250,133,2,165
60110 DATA 2,129,251,230,251,208,2,230,252,165
60120 DATA 253,197,251,208,7,165,254,197,252
60130 DATA 208,1,96,230,2,208,22,9,240,223
60150 READ AA,ZZ
60160 FOR BY=AA TO ZZ
60170 READ LM:POKE BY,LM
60180 NEXT BY
60200 RETURN
60300 REM*****************************************
60301 REM**         SALVARE IL PROGRAMMA          **
60302 REM**          PRIMA DI ESEGUIRLO           **
60303 REM*****************************************
```

Bill Gates

Il creatore della Microsoft, una delle case produttrici di software più note al mondo

La storia della Microsoft è tipica, nel campo dei microcomputer: nata dall'associazione di alcuni appassionati, si è sviluppata in una società con capitali ingentissimi. Venne fondata, nel 1972, da Bill Gates, allora un semplice dilettante, oggi giovanissimo presidente della società.
Bill Gates ebbe i suoi primi contatti con l'informatica quando frequentava le scuole superiori, a Seattle (USA), dove l'associazione dei genitori e degli insegnanti, dimostrando una certa lungimiranza, aveva dotato l'istituto di un terminale collegato ad un minicomputer DEC PDP-11. In seguito, Gates, laureatosi nel frattempo presso l'università di Harvard, intraprese un'attività commerciale con un suo vecchio compagno di scuola: Paul Allen. La ditta che essi fondarono si chiamava Traff-O-Data e si occupava del controllo automatizzato del traffico nella città di Seattle per conto delle autorità locali. Ciò avveniva proprio mentre nascevano i primi microprocessori, quali l'Intel 4004 e, successivamente, il chip 8008.



per concessione della MICROSCOPE
TONY SLEEP

Gates, che aveva sviluppato una discreta pratica sul PDP-11, pensò di adattare il BASIC di quella macchina su di un computer, basato sul nuovo chip 8080. Non disponendo di un sistema di sviluppo, il progetto venne realizzato interamente "sulla carta". Comunque sia, il programma funzionò al primo colpo, quando Gates ebbe l'opportunità di usare un microcomputer presso l'Altair, una nota azienda del Nuovo Messico. Era nato il MBASIC, un linguaggio ad un livello tale che ancora oggi è difficile superare.
In quel momento nacque la Microsoft, che riuscì in brevissimo tempo ad acquisire un'ottima fama come "software house", in grado, all'occorrenza, anche di progettare sistemi operativi per microcomputer. La stessa IBM, che in passato aveva sempre sviluppato il software "tra le proprie mura", si rivolse a Gates, affinché studiasse un sistema operativo per il nuovo Personal IBM. Inizialmente, Gates rifiutò l'offerta, indicando nella Digital Research (la nota produttrice del sistema operativo CP/M) l'organizzazione più adatta per la creazione del software richiesto. Tuttavia, la IBM, dopo un certo tempo, si rivolse nuovamente alla Microsoft, che questa volta accettò l'offerta. I linguaggi fino ad allora sviluppati per i computer ad 8 bit (PASCAL, FORTRAN e MBASIC) vennero riscritti in versione da 16 bit. Ad essi venne affiancata una nuova versione di BASIC, il GW BASIC, dotato di una vasta gamma di comandi specifici per la grafica e la generazione di brani musicali.
Contemporaneamente, Gates si accorse che l'Unix, un sistema operativo multi-utente sviluppato dalla Bell Laboratories, poteva essere adattato ai microcomputer dotati di chip a 16 o 32 bit, creando così lo Xenix. Sia la Tandy Corporation che l'Apple, hanno acquistato i diritti di usare questo nuovo sistema operativo e pare che anche nel software del Mackintosh, prodotto dalla Apple, vi sia lo zampino della Microsoft.
Tra l'altro, la Microsoft ha fornito delle versioni riadattate del MBASIC per alcuni dei computer portatili prodotti in Giappone, fra i quali il Tandy 100, il NEC PC 8201 e l'Olivetti M10 e, ultimamente, ha notevolmente contribuito alla formulazione dello standard MSX, che dovrebbe consentire un facile trasferimento di programmi e dati fra i sistemi operativi MS-DOS e Xenix.

UNO STANDARD INDUSTRIALE

Il BASIC venne sviluppato presso il Dartmouth College (USA), da J. Kemeny e T. Kurtz, nel 1965, ben sette anni prima dell'avvento dei microprocessori. Benché da allora siano state concepite numerose versioni di questo linguaggio, il MBASIC della Microsoft rappresenta ancora oggi un vero e proprio standard industriale.
La Microsoft si sta largamente affermando anche nel campo dei sistemi operativi, con il MS-DOS, fortissimo concorrente del famosissimo CP/M.
Uno degli ultimi prodotti di questa gigantesca software house è l'MS-WINDOWS, un programma gestionale molto sofisticato, usato in abbinamento con un "mouse", destinato a divenire un "best-seller"

Il computer per insegnare

Le esperienze condotte in Gran Bretagna offrono notevoli spunti per la didattica computerizzata

Nel 1980 il governo inglese lanciò un progetto teso ad incoraggiare l'apprendimento dell'uso del computer nelle scuole primarie e secondarie. Conosciuto come MEP (Microcomputers in Education Project, ossia "Progetto per l'uso didattico dei Microcomputer"), il progetto doveva avere una durata di 6 anni ed un costo complessivo di circa 50 miliardi di lire, da ripartire tra le 25.000 scuole (primarie, medie e superiori) britanniche. I microcomputer più adatti al progetto furono considerati il Micro BBC ed il 380Z della Research Machines. Il finanziamento stanziato per il progetto si è subito dimostrato insufficiente, costringendo le scuole a ricorrere ai loro fondi interni.

Nel 1983, a metà strada del progetto, il Ministro per l'Informazione Tecnologica poté affermare che tutte le scuole secondarie del paese (4550 nella sola Inghilterra) e più della metà di quelle primarie possedevano almeno un computer. Questo risultato è stato ottenuto grazie anche ai contributi delle Associazioni dei Genitori e di finanziatori esterni ma, soprattutto, degli studenti stessi. In principio, l'iniziativa del Governo, invece di stimolare un aggiornamento dei programmi di studio e di accrescerne la qualità, sembrò esasperare i problemi esistenti. La necessità di acquistare materiali didattici costosi accrebbe la frustrazione sia degli insegnanti che degli allievi.

Il numero di microcomputer, disponibili in ciascuna scuola, non è ancora sufficiente e, mediamente, ogni studente non trascorre più di 15 minuti al giorno davanti al computer; un arco di tempo appena sufficiente per avvicinarsi ad un programma d'apprendimento su computer e certamente del tutto inadeguato per esplorare le potenzialità didattiche sia delle macchine che del software. In questa ottica, si può facilmente comprendere perché, nelle famiglie inglesi, l'acquisto di un home computer sia principalmente motivato dall'intenzione di offrire un potente strumento educativo ai propri figli, di supporto a quello presente nella scuola.



ALAN ADLER

Una buona partenza

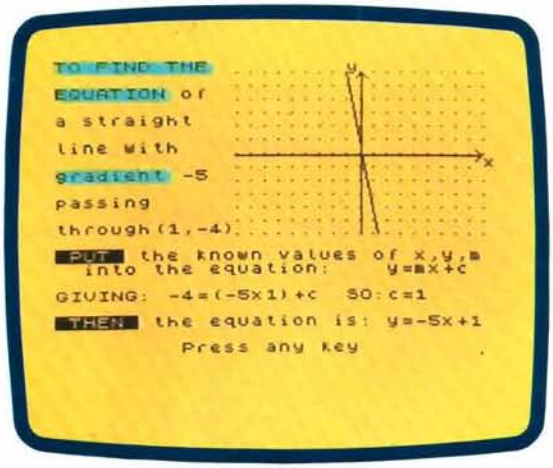
Molti possessori di home computer ritengono, giustamente, di poter offrire maggiori prospettive ai propri figli, acquistando programmi di software didattico. Per i ragazzi più giovani è stato prodotto molto materiale di grande aiuto nelle prime fasi dell'apprendimento e la realizzazione di software, adatto anche a studenti delle scuole superiori italiane, non si farà attendere molto

Sezione "Ripasso"

I pacchetti software, sviluppati per aiutare lo studente nel ripasso di specifiche materie d'esame, hanno un obiettivo piuttosto limitato. Ciascuno di essi è dedicato ad una particolare materia (ad esempio, la matematica o la letteratura), trattata in modo approfondito. Il primo esempio in figura prende in esame equazioni matematiche di vario tipo (lineari, quadrate e sistemi d'equazioni) e viene usato sullo Spectrum, mentre il pacchetto di geometria, progettato per la stessa macchina, offre maggiori prestazioni. Gli ultimi due esempi propongono problemi che fanno riferimento alla Legge di Ohm ed alla progettazione di amplificatori di segnali

Non c'è dubbio che il software didattico abbia un ruolo determinante come valido sussidio all'insegnamento. Scrivere programmi di "ripasso" è un lavoro abbastanza semplice: non occorre una grande abilità di programmazione e le strutture impiegate assomigliano, in molti casi, a quelle dei giochi impostati sull'uso di immagini video. Questo tipo di software ha molti punti in comune con

"Equazioni e disuguaglianze" della Rose Software



i libri di testo tradizionali. Come per i libri di testo, il nome dell'autore riveste una notevole importanza e, inoltre, il metodo di trattare gli argomenti, impiegato nei libri, viene adottato e sviluppato su questo nuovo mezzo didattico.

La differenza fondamentale, comunque, risiede nel concetto di interazione. Secondo la didattica tradizionale, l'apprendimento degli studenti deriva da due fonti principali: l'insegnante ed il libro di testo. Il rapporto tra insegnante e studente è altamente interattivo, anche se distribuito su un arco di tempo limitato (specialmente nelle classi più numerose). Non meraviglia, quindi, il fatto che gli esperti in problemi didattici abbiano esplorato nuove strade che permettano agli studenti di sviluppare un rapporto più creativo ed interattivo con le materie di studio.

Le prime esperienze, nel campo dell'apprendimento automatizzato, sono dovute ai "laboratori linguistici", sviluppati durante gli anni '60 ed applicati all'insegnamento delle lingue straniere, chiamate anche "materie imitative". Ogni studente, al quale viene fornito un registratore ed un testo preregistrato, studia seguendo un ritmo di lavoro idoneo alle proprie capacità. L'insegnante, potendo accedere al canale audio di ciascun studente, controlla lo svolgimento della lezione e, se necessario, interviene direttamente. L'obiettivo dei laboratori linguistici, comunque, era quello di limitare al minimo indispensabile gli interventi dell'insegnante.

Anche i corsi d'insegnamento basati su computer hanno come scopo principale, appunto, quello di essere autonomi dall'opera dell'insegnante. In pratica, il software "di ripasso" deve essere un sistema duttile: deve essere, cioè, completamente autonomo, non deve contenere imprecisioni e, soprattutto, deve essere progettato per pilotare lo studente attraverso la materia di studio in modo facile e spontaneo. Di conseguenza, è necessario che il sistema di elaborazione sia estremamente facile da usare per lo studente senza che egli debba ricorrere all'intervento di un programmatore o di un operatore specializzato.

Più oltre nel corso esamineremo i contenuti del software didattico, ma è bene far subito una prima grossa distinzione tra le materie scientifiche (nelle quali si ha a che fare con conoscenze e fenomeni reali quantificabili) e le materie letterarie (dove le analisi hanno carattere individuale e sono legate a valutazioni puramente soggettive). Nel caso del software di ripasso, è necessaria un'ulteriore distinzione: le materie scientifiche sono, per lo più, caratterizzate da moltissime illustra-

"Geometria" della Rose Software



zioni (anche quando il microcomputer usato possiede scarse possibilità grafiche). Al contrario, i programmi per le materie letterarie si basano essenzialmente su routine per l'analisi dei testi.

Anche in Italia esistono numerosi programmi per microcomputer, orientati all'insegnamento dei primi elementi di aritmetica, della lettura e della compitazione, per le scuole elementari, ma il vero e proprio software didattico è ancora in fase di sperimentazione, soprattutto in attesa di precise direttive ministeriali.

Alle due grandi categorie di programmi (quelli basati su semplici meccanismi a "domanda e risposta" e quelli nei quali, in base alla risposta, viene determinato il successivo livello di spiegazione) si aggiunge una terza branca: quella dei programmi di simulazione sperimentale. In questo caso le tecniche di programmazione utilizzate sono molto più complesse, poiché i programmi devono riprodurre, matematicamente e graficamente, le interazioni tra varie forze fisiche. Un software di questo genere viene adottato con entusiasmo sia dagli insegnanti di scienze, nelle scuole, sia nel settore della ricerca industriale, in quanto permette un più alto livello di sperimentazione, specialmente in quelle pro-

ve troppo pericolose o troppo costose per esser fatte nella realtà.
Seguendo il parere degli esperti in didattica, il software è stato suddiviso in tre grandi fasce, a seconda dell'età degli allievi. Vediamone le caratteristiche salienti.

1ª fascia: fino agli 8 anni

Molti pacchetti software progettati per i ragazzi più giovani riguardano lo sviluppo delle capacità di riconoscimento delle forme e dei modelli. L'aritmetica, a livello elementare, o la scomposizione delle parole vengono trattate mediante "oggetti" da riconoscere e da confrontare, in modo da rendere familiari i simboli utilizzati in queste materie (ad esempio, i numeri e le lettere alfabetiche). Molti programmi, che fanno parte di questa area didattica, fanno uso di procedure simili a giochi, con lo scopo di tener maggiormente desta l'attenzione dei bambini. Alcuni pacchetti richiedono però la presenza di un adulto o di un ragazzo più grande.
Particolare successo riscuotono i programmi che insegnano ai bambini a leggere l'orologio, a contare, a sommare e sottrarre (un interessante metodo presenta una bilancia animata i cui piatti si inclinano dall'una o

"D.C." della SciCAL Software



dall'altra parte a seconda del peso posto su di essi), a costruire brevi frasi ed a sillabare parole di uso comune.

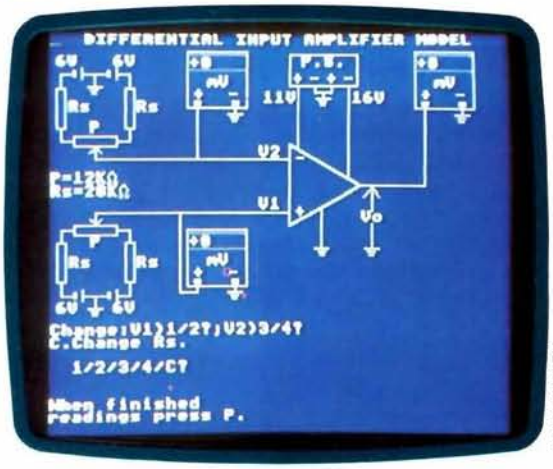
2ª fascia: dai 9 ai 14 anni

Il software sviluppato per questa fascia d'età ha un'impostazione diversa dal precedente: si allontana lo studente da un apprendimento basato sul gioco per avvicinarlo ad uno studio più disciplinato, spesso riproponendo argomenti già trattati in classe, ma in modo tradizionale. Affinché l'interesse dei ragazzi rimanga vivo e costante, sono stati compiuti notevoli sforzi per dotare i programmi di incentivi e premi, in risposta ai progressi fatti nello studio delle varie materie. Un artificio, che riscuote un notevole successo, prevede la comparsa sullo schermo di un videogioco, purché il ragazzo riesca a completare una parte del programma d'apprendimento in un tempo prestabilito ed ottenendo risultati superiori ad un valore prefissato.
L'aritmetica, la sillabazione delle parole e l'appropriato uso del linguaggio sono ancora gli argomenti di maggior interesse per i ragazzi in questo gruppo d'età. Ma, oltre a questi, sono disponibili numerosi pacchetti di carattere storico o geografico ed altri che insegnano la teoria musicale. Altri ancora hanno come oggetto semplici simulazioni.

3ª fascia: 15 anni e oltre

In questa fascia, per il particolare livello delle materie trattate, il software didattico produce i risultati migliori. Naturalmente, sono disponibili pacchetti che permettono di approfondire le conoscenze di base delle varie materie, ma si preferisce usare, in genere, un software concentrato su specifici argomenti di studio. Questi non sono direttamente collegati ad un particolare tema d'esame ma, più spesso, ad un insieme di corsi di studio proposti da una particolare commissione d'esame. A questi livelli, i programmi presentano, solitamente, un "compendio" o riassunto dei temi affrontati negli anni precedenti. Per materie come la letteratura, vengono offerti dei compendi di analisi stilistiche o di contenuto, trattando i vari argomenti proprio come farebbe un insegnante.
È ragionevole supporre che gli studenti appartenenti a questa fascia d'età siano sufficientemente automotivati, quindi le procedure dei programmi non adottano più meccanismi artificiosi, atti ad accaparrarsi l'attenzione del ragazzo. Si cerca, al contrario, di presentare gli argomenti in modo interessan-

"Ampli" della SciCAL Software



IAN McKINNELL

te usando grafica statica o animata, magari corredata da brevi brani musicali. Per questo gruppo d'età, all'estero, è disponibile un'ampia varietà di software didattico. Esamineremo più avanti ed in modo più approfondito la gamma di pacchetti software prodotti per i più popolari home computer.

Funzioni e strutture di controllo

Quali vantaggi offre, in merito, il BASIC della Sinclair

Alcune funzioni del BASIC della Sinclair, contrariamente alle loro equivalenti negli altri dialetti, non richiedono l'uso di parentesi per gli argomenti. Ad esempio, si può scrivere LEN X$ oppure LEN (X$). Conviene usare le parentesi se il significato di un'espressione è dubbio o ambiguo.
La funzione CODE è l'equivalente Sinclair della funzione ASC ( ). Tuttavia, occorre prestare attenzione al fatto che la gamma di caratteri Sinclair segue lo standard ASCII solo per i valori compresi tra 32 e 122. Cosicché, mentre PRINT CHR$(7) nella maggior parte dei dialetti BASIC provoca l'emissione di un "bip" sonoro, nel BASIC Sinclair questa istruzione corrisponde alla visualizzazione di un messaggio d'errore.
La funzione VAL rientra nello standard, ma scrivere, ad esempio, VAL("a45") provoca l'interruzione del programma, in quanto l'argomento della funzione non è numerico. Nella maggior parte degli altri dialetti BASIC questa istruzione "ritornerebbe" con il valore 0. Naturalmente, a ciò si può ovviare scrivendo una subroutine di verifica, che esamini il codice del primo carattere dell'argomento: se CODE A$(1) <48 oppure CODE A$(1) >57, allora A$ non risulta numerico e non è adatto ad essere l'argomento della funzione VAL.
Il VAL del Sinclair possiede, tuttavia, l'insolita capacità di calcolare espressioni numeriche. Quindi, scrivendo:

```
LET A4 = "6*12": PRINT VAL A$
```

il computer visualizza 72, ossia il risultato dell'espressione 6 × 12. Nella maggior parte dei dialetti BASIC, invece, una simile linea di programma visualizza il valore 6.
Vediamo come sfruttare questa caratteristica tipica del VAL Sinclair. Un semplice esempio è illustrato da questo programma che disegna un grafico a blocchi:

```
100 DIM S$ (31)
200 LET S$ = " ********************************"
300 INPUT "INSERIRE UNA FUNZIONE DI X";F$
400 PRINT "Y = ";F$:PRINT
500 FOR X = 1 TO 10
600 PRINT S$(TO INT(VALF$))
700 NEXT X
800 PRINT " ======================="
900 PRINT"00000000011111111112222222223"
950 PRINT"12345678901234567890123456789O"
```

Quando si lancia il programma, si può inserire una qualsiasi espressione algebrica, contenente la variabile X, come ad esempio (2*X+3/X), ottenendo sul video uno schematico grafico a blocchi della funzione. In questo programma l'asse delle y è orizzontale, quello delle x verticale e la grafica è a dimensione di pixel. Con poche righe di programma in più, è possibile anche determinare la massima ampiezza dei valori della x e della y, in modo da aggiustare automaticamente il fattore di "scala", stampare gli assi ed, infine, visualizzare il grafico in alta risoluzione; il programma avrebbe un notevole effetto.
Le istruzioni GOSUB e GOTO, da questo punto di vista, sono abbastanza simili a VAL. Infatti, il loro argomento, che nel normale BASIC sarebbe un numero di linea, può essere anche un'espressione matematica. Questa caratteristica presenta numerosi vantaggi. È possibile, ad esempio, associare dei nomi alle subroutine del programma, assegnando a delle variabili i relativi numeri di linea, e quindi "chiamare" le subroutine (GOSUB) usando il loro nome. Il seguente esempio chiarisce questo procedimento:

```
100 LET INIZIO = 1000
200 LET STAMPA = 2000
300 LET CALCOLA = 3000
400 GOSUB INIZIO
500 GOSUB CALCOLA
600 GOSUB STAMPA
700 STOP
1000 REM *SUBROUTINE INIZIO*
2000 REM *SUBROUTINE STAMPA*
3000 REM *SUBROUTINE CALCOLA*
```

Ciò rende il listato di più facile interpretazione. Se sostituiamo a GOSUB INIT l'istruzione GOSUB (VAR1+N*VAR2) o una qualsiasi espressione numerica valida, l'espressione verrà calcolata ed il risultato verrà trattato come un numero di linea. Nel BASIC Sinclair, se l'argomento dell'istruzione GOTO o GOSUB è un numero di linea che non esiste, allora il controllo passa al successivo numero di linea valido. Se, ad esempio, scriviamo GOTO 17 e la linea 17 non esiste, allora il controllo passa alla linea 18, senza che compaia alcuna segnalazione d'errore, come avviene invece per la maggior parte dei dialetti BASIC. La capacità del BASIC Sinclair di trattare questi "salti calcolati", permette di aggirare i problemi dovuti alla mancanza delle istruzioni ON...GOTO e ON...GOSUB. In un altro dialetto BASIC si potrebbe scrivere:

```
2360 ON D GOSUB 100,200,300,400,500
```

In questa linea di programma, se il valore di D è 1, il programma salta alla linea 100, se D=2, salta alla linea 200 e così via. Nel BASIC Sinclair occorre scrivere:

```
2360 GOSUB (100*D)
```

con identici risultati. Si noti che in entrambe le versioni è necessario prevedere la possibilità che il valore di D non sia intero, che sia minore di 1 o maggiore di 5.
Il BASIC dello Spectrum non possiede strutture del tipo REPEAT...UNTIL oppure WHILE...WEND, ma è possibile simularle in più modi. La struttura REPEAT è un ciclo chiuso che ha inizio con la parola REPEAT e termina con la parola UNTIL seguita da un'istruzione di condizione. Se questa condizione risulta vera, allora il ciclo termina ed il controllo passa all'istruzione che segue UNTIL. Sullo Spectrum questa struttura di controllo può essere simulata nel modo seguente:

```
100 DATA "A","B","C","D","E","*","F","G","*"
200 FOR L = 1 TO 1
300 READ X$
400 PRINT X$
500 IF X$<>"*" THEN LET L = 0
550 NEXT L
600 PRINT "END OF DATA"
```

Il problema di una struttura REPEAT...UNTIL è che, siccome la condizione di uscita dal ciclo viene esaminata alla fine del ciclo stesso esso viene sempre eseguito almeno una volta Per eliminare questo inconveniente, si ricorre alla struttura WHILE...WEND.

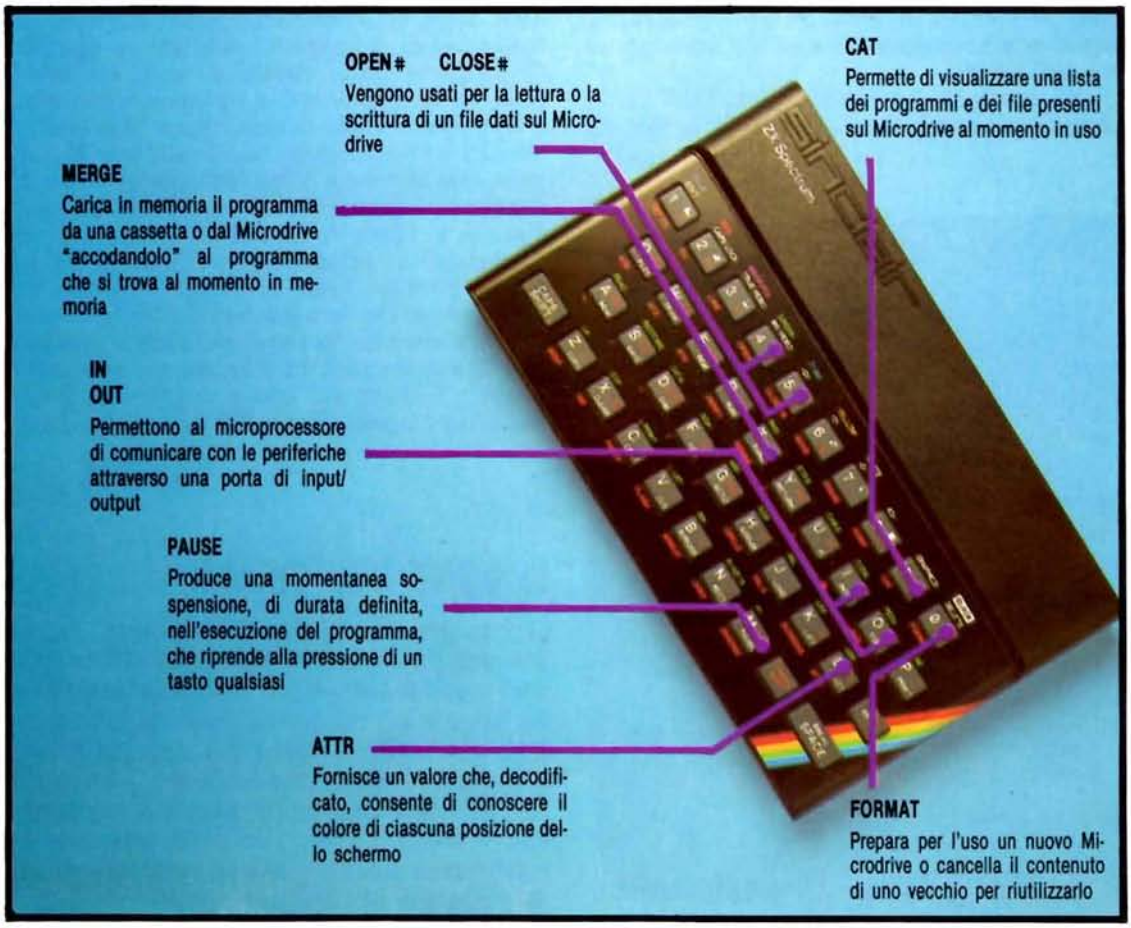
Anche questa è un ciclo chiuso, che ha inizio con l'istruzione WHILE (seguita da una condizione) e che termina con l'istruzione WEND. Finché la condizione risulta falsa, le istruzioni di programma comprese tra WHILE e WEND vengono ripetutamente eseguite. Quando la condizione risulta vera, il controllo passa dall'istruzione WHILE a quella che segue l'istruzione WEND (le istruzioni che si trovano all'interno del ciclo chiuso verranno saltate). Il programma risultante sarebbe:

```
100 DATA "A","B","C","D","E","*","F","G","*"
200 WHILE X$<>"*" DO
300 READ X$
400 PRINT X$
500 WEND
600 PRINT "END OF DATA"
```

Questa struttura viene simulata, su un computer Sinclair, con l'uso di IF...THEN...GOTO. Alternativamente:

```
 50 LETX$ = ""
100 DATA "A","B","C","D","E","*","F","G","*"
200 LET TEST = (X$<>"*")
250 FOR L = 1 TO TEST
300 READ X$
400 PRINT X$
500 LET L = (X$ = "*")
550 NEXT L
600 PRINT "END OF DATA"
```

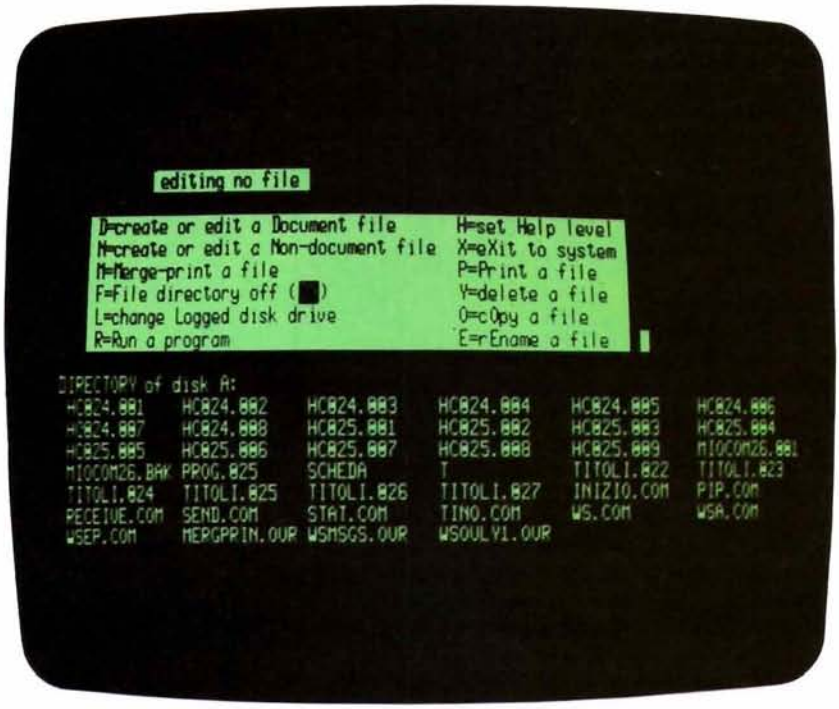
Con questo, abbiamo esaminato le principali variazioni del dialetto BASIC Sinclair. Esistono altre piccole differenze, ma di minima importanza.



Il WordStar

Questo programma, prodotto dalla Micropro, è uno dei più noti e diffusi word processor per micro computer

Nella sua forma più semplice, un word processor non è altro che una sofisticata macchina da scrivere. Il video sostituisce il foglio di carta ed il tasto DELETE la gomma. Per un uso commerciale, la caratteristica più significativa di un word processor è la capacità di archiviare documenti (costituiti da singole righe, frasi, paragrafi o intere pagine) e permettere, in un secondo momento, che essi vengano inseriti in un nuovo testo. Ciò è particolarmente utile nella stesura di contratti e di altri documenti nei quali una o più parti rimangono per lo più invariate.
Un documento può essere quindi creato, modificato, archiviato e richiamato: un altro importante aspetto è l'impaginazione di un testo. Esistono tre modi principali per definire l'impaginazione di un testo: con allineamento a sinistra, con allineamento a destra o con la "giustificazione". Nel caso di un testo stampato col metodo della giustificazione, gli spazi tra le parole e le lettere vengono sistemati in modo che tutte le linee presentino una stessa lunghezza e siano ben allineate su entrambi i lati.
Un programma di buona qualità, per l'elaborazione di testi, deve essere in grado di trovare una specifica parola o frase all'interno di un intero documento e di sostituirla, all'occorrenza, con un'altra parola o frase. Questa caratteristica è assai utile quando una parola è stata ripetutamente scritta in modo errato oppure quando si desidera "personalizzare" il testo di una circolare.
In termini tecnici, il WordStar della Micropro è semplicemente un word processor controllato da menu ed un text editor dotato di chiari comandi per la redazione e l'impaginazione di un testo. Il programma funziona col sistema operativo CP/M e richiede l'uso del tasto CONTROL, onde poter distinguere tra i codici di comando ed una normale battuta di un carattere. Nel WordStar, l'elenco dei comandi disponibili può essere visualizzato costantemente sullo schermo (occupando, in questo caso, le prime dieci righe dello schermo), eliminato o visualizzato solo in parte. I comandi, che conferiscono particolari caratteristiche di stampa al testo, vengono evidenziati sullo schermo (all'interno dello stesso testo) ma non vengono riportati in stampa. Gran parte della gestione degli archivi è affidata direttamente al sistema operativo.
Una volta caricato in memoria e "lanciato", il WordStar presenta all'utente il Menu Principale, che permette di selezionare la funzione desiderata: si può creare o redigere un testo sia che si tratti di un documento che di un "non documento" (questo termine viene usato per distinguere i normali testi dai listati dei programmi, che non devono contenere caratteri di controllo). Si possono fondere assieme due file (inviandoli in stampa o meno), selezionare il drive sul quale conservare i file, cancellare o copiare un file oppure cambiar-



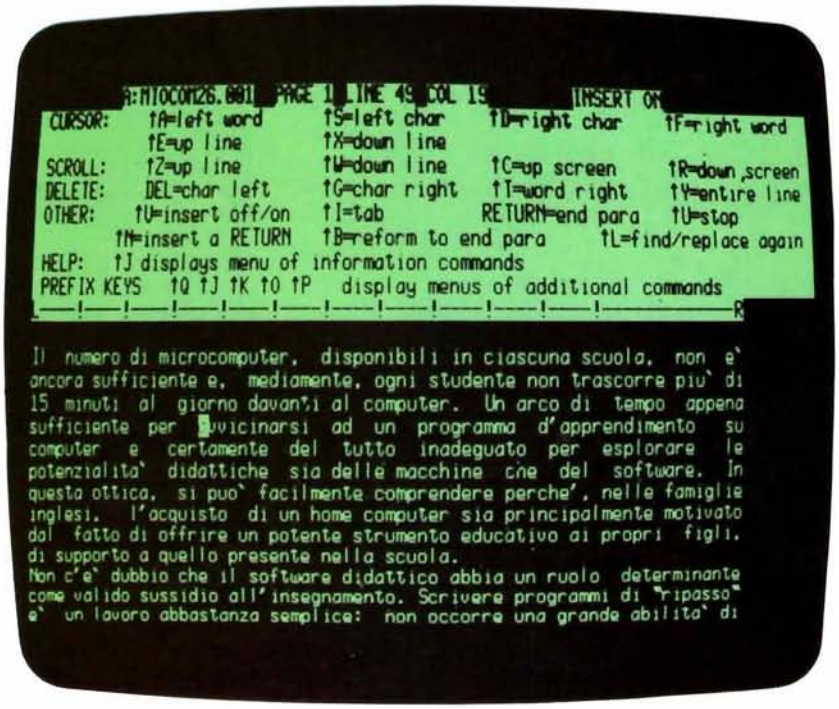
IL MENU PRINCIPALE

La prima immagine che il WordStar visualizza è il menu principale che presenta le varie funzioni di base tra le quali scegliere. L'utente può creare o redigere sia un file "documento", usando tutti i parametri di impaginazione standard (come l'invio a capo della parola finale di una riga, l'allineamento a destra, ecc...), che un file "non-documento" (ad esempio: il listato di un programma). Altre funzioni sono copiare un file, assegnargli un nuovo nome o cancellarlo. In breve, questa prima parte del programma tratta routine gestionali

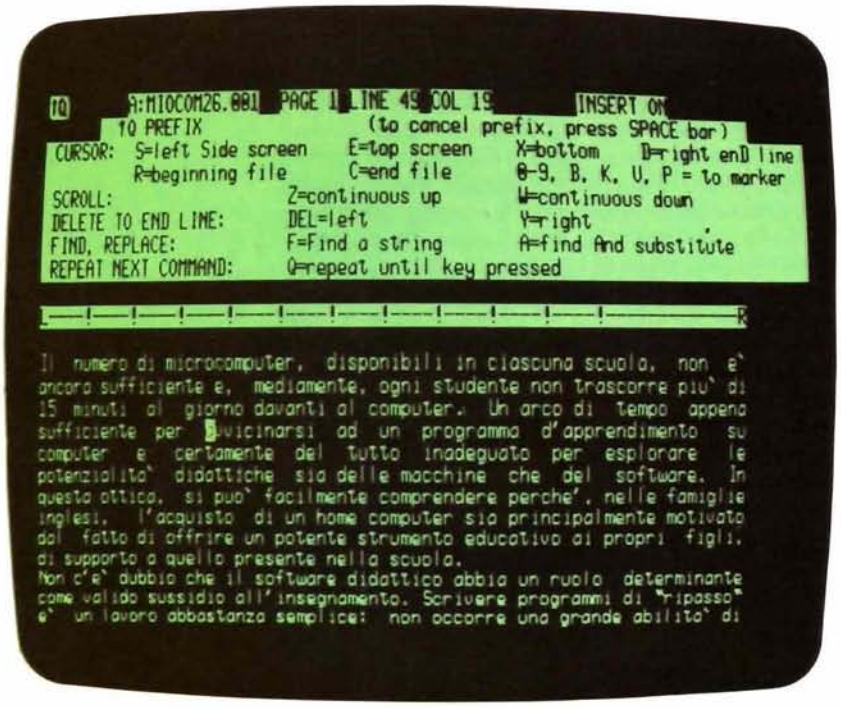
gli nome, abbandonare il sistema operativo ed infine selezionare il livello di assistenza desiderato.
Nel menu principale, ogni scelta avviene mediante la battuta di un singolo tasto (non c'è bisogno di usare il tasto CONTROL). Ciò provoca la presentazione di un sottomenu specifico, relativo alla funzione richiesta. L'ultima delle opzioni citate (la selezione del livello d'assistenza) consente di scegliere la quantità d'informazioni contenute nei vari sottomenu: ad un minor livello corrisponde un minor numero di linee riservate al menu e, di conseguenza, un maggior spazio a disposizione per la redazione del testo. La maggior parte dei comandi iniziali si ricordano molto facilmente, essendo composti da un singolo carattere. Ad esempio, D permette di creare un file documento, mentre P serve per la stampa di un testo.
La prima domanda che il computer rivolge all'utente è il nome da attribuire al file ch'egli si accinge a creare. Fornito questo, il sistema visualizza il menu di lavoro, contenente alcune scelte possibili, come il movimento del cursore, l'inserimento o la cancellazione dei caratteri, assieme a brevi frasi che chiariscono l'uso dei codici di controllo a singolo carattere, usati per richiamare altre derivazioni del menu principale (ad esempio, il menu per l'impaginazione). Il WordStar non dà per scontato che la macchina sulla quale verrà usato (il computer "oggetto"), sia dotata di tasti per il controllo del cursore.
Lo spostamento del cursore, di una posizione alla volta, si ottiene premendo contemporaneamente il tasto CONTROL ed i tasti E,S,D ed X (sulle macchine dotate di tasti di controllo per il cursore, questi comandi risultano duplicati). Il WordStar è dotato di funzioni, per il controllo del cursore assai efficaci: è in grado di sistemare singoli caratteri, parole, linee o interi paragrafi spostandoli automaticamente in tutte le direzioni. Anche i comandi per la cancellazione di singole lettere o di intere parole sono altrettanto elastici: è possibile cancellare un singolo carattere, sia sotto il cursore sia immediatamente alla sua destra; si può cancellare un'intera parola posta all'immediata destra del cursore o un'intera riga (indipendentemente dalla posizione del cursore). Le funzioni di controllo consentono anche di ottenere uno scorrimento del testo sul video, senza modificare il posizionamento del cursore. Questa procedura è estremamente utile quando si desidera esaminare una parte del testo, per poi tornare a quella in fase di stesura.
Il metodo d'inserimento dei caratteri è selezionabile: si può decidere se inserire un carattere alla posizione del cursore, provocando l'eventuale scorrimento a destra del resto dello scritto, oppure se sovrapporre (ossia sostituire) i caratteri dello scritto con quelli immessi da tastiera. Questa funzione (INSERT ON) viene abilitata o disabilitata agendo ripetutamente sul comando CONTROL V.
Il programma provvede automaticamente all'impaginazione ed all'invio a capo delle parole che superano il margine destro della riga, durante la stessa immissione del testo.
Tutte le funzioni sin qui descritte sono di uso comune nella creazione e redazione di documenti, ma il WordStar possiede anche altre particolarità. Può automaticamente centrare lo scritto (ad esempio, un titolo) sulla pagina, provocare passaggi ripetuti di stampa (sulla stessa riga) per creare scritte in neretto, stampare caratteri al di sopra e al di sotto della linea, sovrastampare la linea precedente ed infine sottolineare e cancellare singoli caratteri. Tutte queste funzioni sono utilizzabili sia durante l'inserzione di un testo che durante la sua redazione.
La funzione di "ricerca e sostituzione" viene abilitata mediante il menu Q, che contiene anche le funzioni supplementari di posizionamento del cursore e di cancellazione. L'operatore può richiedere la ricerca di una parola o frase all'interno di un testo, specificando anche se operare una sostituzione con una nuova parola o frase.

IL SOTTOMENU "DI LAVORO"

Durante l'immissione del testo, nella parte superiore dello schermo compare un breve elenco dei comandi relativi allo spostamento del cursore, alla cancellazione di lettere o parole, ecc. Come tutti i sottomenu, anche questo può essere eliminato, lasciando maggior spazio all'area di scrittura.
Il comando J consente l'accesso ad una serie di spiegazioni molto utili se non si ha il manuale a portata di mano

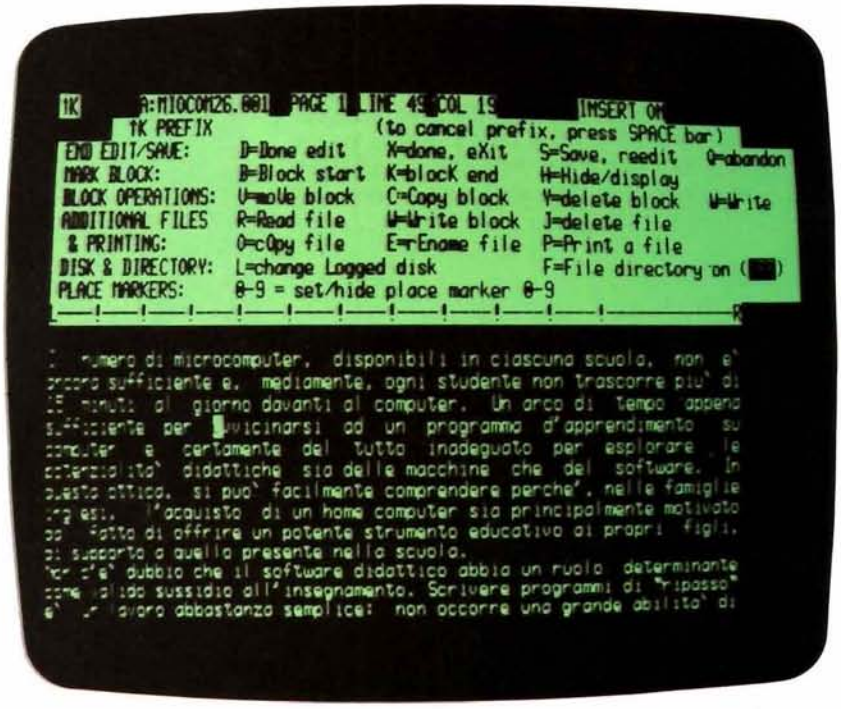


Il sottomenu K presenta le varie possibilità offerte dal WordStar per lo spostamento, la copia o l'eliminazione di interi blocchi di testo, opportunamente "marcati" o addirittura di testi precedentemente archiviati. Altri tre sottomenu del WordStar, il J, l'O ed il P, aiutano, rispettivamente, nella creazione o redazione di un documento, nell'impaginazione e sistemazione di un documento ed, infine, nella definizione del suo formato di stampa. Complessivamente, il WordStar rappresenta un pacchetto software completo e di qualità professionale. Pochissime sono le funzioni che questo word processor non è in grado di eseguire efficacemente. Sebbene a prima vista possa sembrare complicato e di difficile impiego, può invece essere adoperato anche da un principiante, grazie agli esaurienti menu. Un gran numero di microcomputer, che sono in grado di adottare questo word processor, possiedono tasti di funzione programmabili come standard. In commercio, comunque, sono reperibili speciali tastiere, adattabili alla maggior parte dei computer più diffusi, che facilitano l'immissione dei comandi del WordStar: ogni tasto è identificato da una sintetica descrizione della funzione svolta.



IL SOTTOMENU Q

I menu secondari del WordStar vengono richiamati usando un codice di controllo a singolo carattere che risulta quindi costituito da sigle estremamente sintetiche. Il menu Q, ad esempio, è il sintetico modo con cui ci si riferisce al menu di Comando Rapido. Il Q menu fornisce all'utente una lista dei tasti di controllo del cursore e delle efficaci funzioni di ricerca e sostituzione



IL SOTTOMENU K

Alcune sigle composte da un singolo carattere sono piuttosto oscure; K, ad esempio, sta per "blocco". Questo controlla l'archiviazione e l'utilizzazione dei file, lo spostamento di blocchi di testo ed alcune strutture di accesso ai file. È da notare che questo menu ripresenta alcune delle routine gestionali presenti anche nel menu principale

La scelta di un buon monitor

La qualità dell'immagine video è importante per evitare danni alla vista

Riguardo alla nostra salute, il video è una delle più importanti periferiche collegate al computer. In molti casi viene usato un normale televisore o un economico apparecchio in bianco e nero acquistato appositamente e, benché la qualità delle immagini sia accettabile, non è certamente quanto di meglio si possa ottenere.

Infatti, il segnale inviato allo schermo dalla memoria video, posta all'interno del computer, viene sottoposto a numerosi passaggi di codifica e decodifica. Indipendentemente dalla buona qualità dei circuiti coinvolti in queste operazioni, il segnale presenta sempre qualche difetto e l'immagine finale risulta spesso offuscata, difficile da leggere o caratterizzata da un fastidioso sfarfallio.

Il segreto per ottenere immagini di alta qualità consiste nell'eliminare le distorsioni di segnale e, poiché queste sono prodotte dai circuiti di modulazione e demodulazione, la soluzione più ovvia è di rimuoverli completamente ed adottare un sistema alternativo.

I monitor, appunto, sono concepiti con un simile criterio: costruiti sempre attorno a dei cinescopi televisivi, sono privi però di modulatori e demodulatori e costituiscono quindi un sistema semplificato in grado di produrre immagini più nitide, più luminose e stabili.

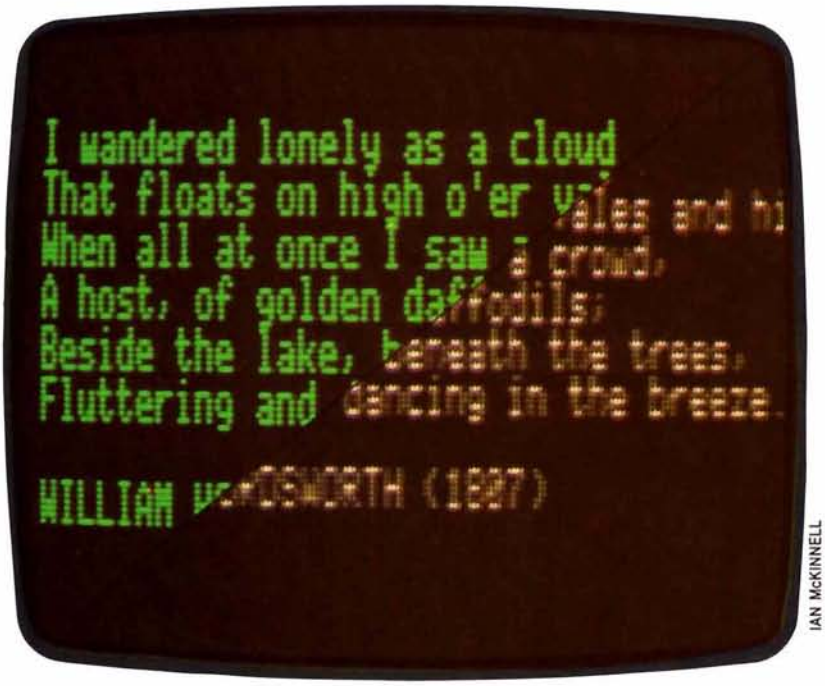
Poiché i monitor sono privi di demodulatori di tipo televisivo, è necessario, dunque, che il computer fornisca un tipo di segnale adatto, da applicare direttamente al monitor. Sia gli home che i personal computer di più recente fabbricazione offrono più tipi d'uscita video, tra i quali quello per monitor.

La generazione di un'immagine sullo schermo televisivo si sviluppa secondo una procedura composta da più passaggi, che si susseguono secondo tempi ben definiti. Il problema deriva dal fatto che il movimento di scansione del fascio elettronico, sulla superficie del tubo, viene generato all'interno del monitor e risulta inaccessibile da parte del circuito di controllo interno al computer.

Il fascio di elettroni di un monitor acceso, ma non ancora collegato ad un segnale, "spazzola" l'intero schermo 60 volte al secondo, producendo così un'area visiva equamente illuminata. Quando, però, si collega un segnale opportunamente modulato, esso provoca l'accensione e lo spegnimento, con pari frequenza, di ogni punto dello schermo: un qualsiasi fenomeno di instabilità, durante questo rapidissimo processo, provoca fastidiosi sfarfallii, con serie conseguenze per la vista dell'operatore.

L'intero processo è regolato da impulsi di sincronizzazione, prodotti, assieme al segnale di luminosità, direttamente dal computer. Gli impulsi di sincronizzazione sono di due tipi: il primo per ciascuna linea dell'immagine ed il secondo per l'immagine presa nel suo complesso. Al termine di ciascun ciclo completo, il monitor riceve un breve segnale per informare che la scansione dell'intera immagine è stata completata. A questo punto il fascio di elettroni (e quindi il punto luminoso che esso produce) deve tornare nell'angolo in alto a sinistra dello schermo da dove inizierà un nuovo ciclo di scansione.

Un procedimento simile avviene alla fine di ogni linea: il monitor riceve un segnale per indicare che il "pennello" ha completato una linea e che deve quindi essere riportato sul margine sinistro dello schermo, da dove inizia la scansione della linea successiva.



IAN McKINNELL

IMMAGINI A CONFRONTO

La figura qui riportata mostra la sensibile differenza tra la qualità delle immagini ottenibili su un monitor monocromo progettato appositamente per computer (in questo caso il Monitor III della Apple) e su un televisore domestico di alta qualità, usati entrambi per visualizzare le immagini prodotte da un pacchetto di word processing

Ci sono vari tipi di monitor, ma la principale distinzione è tra i monocromi e quelli a colori. I monitor monocromi sono piuttosto semplici, concettualmente, ed il solo importante tipo di segnale impiegato viene detto "composito". In esso gli impulsi di sincronizzazione vengono combinati al livello di brillantezza in un unico segnale che poi i circuiti del monitor scompongono per dar vita all'immagine.

I monitor a colori

Un meccanismo simile è stato adottato in molti monitor a colori ma, poiché il segnale è assai più complesso, questi apparecchi risultano più simili a dei televisori a colori, coi quali hanno in comune gran parte dei circuiti. I principali sistemi a colori sono il PAL, il SECAM e lo NTSC, gli stessi nomi adottati nell'emittenza televisiva. Questi sistemi rappresentano i diversi metodi di codifica dei tre colori "additivi" (rosso, verde, blu) e dei due impulsi di sincronizzazione.
In alternativa, i vari segnali possono essere inviati al monitor separatamente. Sebbene vi siano sottili differenze nei meccanismi di trasmissione usati, questi sistemi vengono genericamente chiamati RGB (= Red, Green, Blue, ossia rosso, verde e blu). Il segnale più semplice è del tipo TTL (logica transistor-transistor) nel quale i colori possiedono solo due stati, on e off, e la loro combinazione dà vita agli usuali otto colori utilizzati dai sistemi viewdata come il Prestel e il Ceefax.

LARGHEZZA DI BANDA

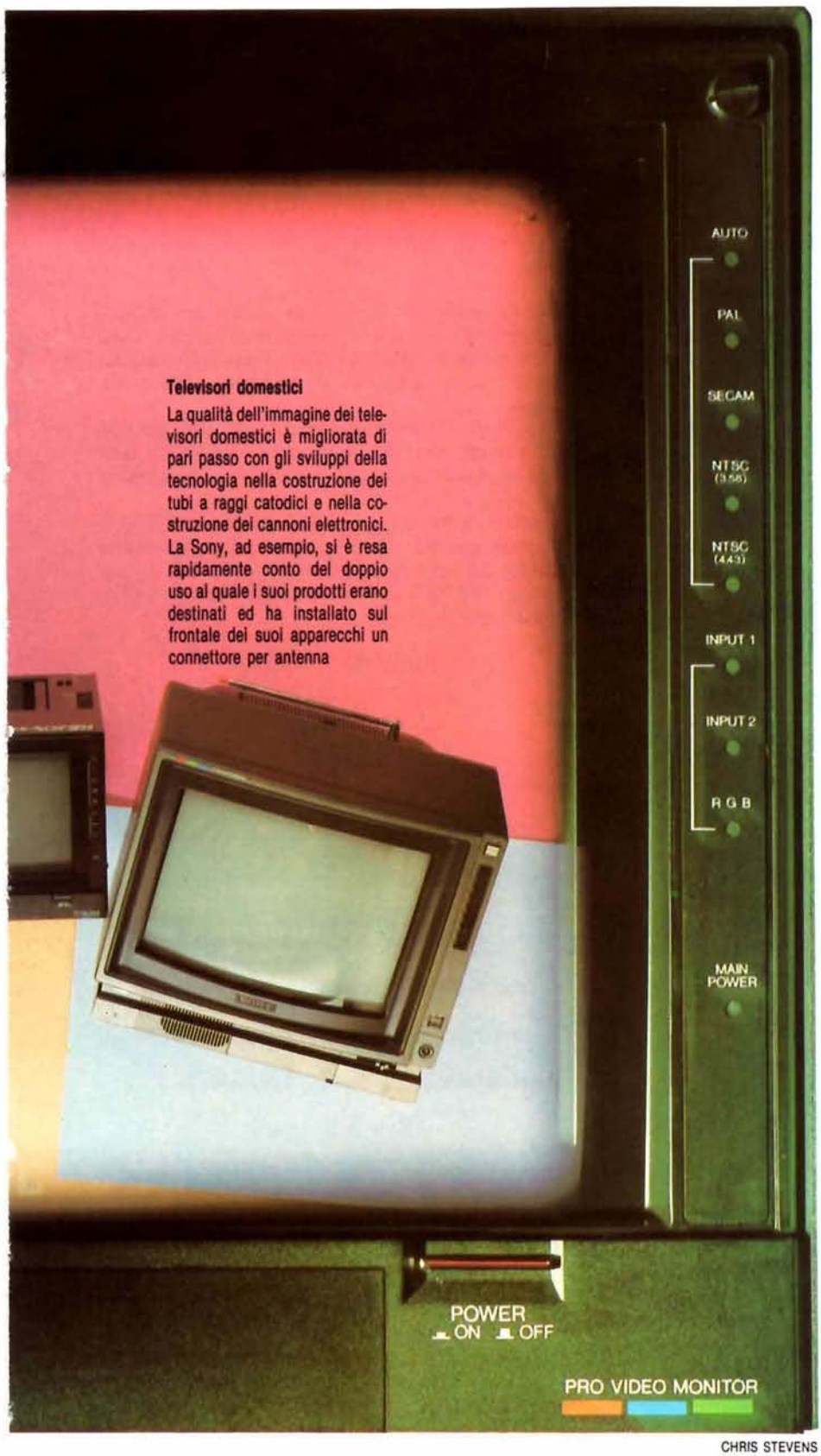
Nell'acquisto di un monitor, la larghezza di banda è un elemento da tenere in grande considerazione: indica quanto piccolo è il punto elettronico generato dalla circuiteria. In linea di massima, più piccolo è il punto, migliore è la resa, ma entrano in gioco anche altre considerazioni.
In primo luogo occorre stabilire la minima larghezza di banda accettabile per il computer. Per calcolarla, si moltiplica il massimo numero di caratteri, su ciascuna riga dello schermo, per il numero massimo di righe, ottenendo il numero totale di caratteri. Questo valore viene quindi moltiplicato per la larghezza (in punti) della matrice sulla quale sono costruiti i caratteri ed, infine, per l'altezza (sempre in punti) della matrice.
Il risultato è il numero totale di punti da rappresentare sullo schermo che, generalmente, è compreso tra 10.000 e 1.500.000. Per esempio, in un normale schermo 24 x 80, con una matrice 7 x 9, il valore è 110.960. Tutti questi punti vengono illuminati, dal cannone elettronico, ben 60 volte al secondo (in alcune nazioni il ritmo è di 50 volte al secondo). Moltiplicando il risultato ottenuto precedentemente per questo valore, si ottiene un numero tra 600.000 e 90.000.000, che rappresenta quante volte al secondo ogni singolo punto debba essere illuminato.
Trattandosi di una frequenza, il valore va espresso in Hertz (Hz), nel caso specifico in MegaHertz (MHz).
La larghezza di banda dei monitor in commercio è, in genere, di 5, 7, 10, 12 e 15 MHz. Il prezzo s'impenna notevolmente per i monitor da 20MHz ed è addirittura inaccessibile ai più per monitor da 100MHz.
Ad una maggiore larghezza di banda corrisponde una maggiore definizione, generalmente preferibile. Una bassa larghezza di banda produce caratteri sbavati e di difficile lettura, mentre, con una larghezza di banda troppo alta, i contorni di ciascun punto sono talmente precisi da ostacolare la continuità di un'immagine



Monitor monocromi
I monitor più semplici e meno costosi sono quelli monocromi. Ad un prezzo accessibile, questi apparecchi mettono a disposizione schermi a fosfori colorati ed una definizione discretamente alta. Sono adeguati per le necessità di word processing e di altre applicazioni commerciali, ma vengono usati anche per semplici immagini grafiche

Monitor a tre colori
Il CUB della Microvitec è un eccellente esempio di monitor a colori di basso costo utilizzabile con un home computer. Sebbene la sua capacità di risoluzione non sia altissima, è perfettamente adeguato per un gran numero di applicazioni, comprese quelle grafiche

Versatilità dal Giappone
Il monitor TM90PSM è in grado di ricevere ed adattarsi a segnali di diverso tipo. È quindi particolarmente adatto a quegli utenti che vogliano utilizzare il loro monitor per differenti applicazioni. Oltre a svolgere la funzione di output visivo per computer, può essere collegato ad un video registratore a cassette o a dischi



Televisori domestici

La qualità dell'immagine dei televisori domestici è migliorata di pari passo con gli sviluppi della tecnologia nella costruzione dei tubi a raggi catodici e nella costruzione dei cannoni elettronici. La Sony, ad esempio, si è resa rapidamente conto del doppio uso al quale i suoi prodotti erano destinati ed ha installato sul frontale dei suoi apparecchi un connettore per antenna

CHRIS STEVENS

È possibile ottenere un maggior numero di sfumature, qualora si possa attribuire una "intensità" a ciascun colore. In questo caso, benché le intensità vengano graduate in "passi discreti" (digitali), i monitor vengono chiamati "analogici" o "lineari".

Molti monitor a colori sono disponibili tanto in versione TTL che analogica, con lievi differenze di prezzo.

Il monitor TM90PSM è un interessante e flessibile apparecchio prodotto dalla industria JVC. Le sue dimensioni sono piuttosto ridotte (lo schermo è da 10 pollici) e sebbene la sua ampiezza di banda non sia particolarmente alta (il che pone dei limiti alla risoluzione massima), questo monitor è in grado di ricevere quasi ogni tipo di input, dal comune segnale composito monocromo, al TTL, all'analogico RGB, fino ad uno qualsiasi dei 4 segnali compositi a colori PAL, SECAM, NTSC 3,58 e NTSC 4,43. Possiede internamente un meccanismo di selezione capace di analizzare il segnale in arrivo e di predisporre quindi il monitor nel modo adatto.

Questo tipo di monitor sta diventando sempre più comune e può essere utilizzato come terminale generico per video registratori a cassette o a dischi e per altri apparecchi.

PERSISTENZA E COLORE

Nella scelta di un monitor monocromo, è importante considerare il tipo di fosfori impiegato (la sostanza che rende luminescente lo schermo) o, per meglio dire, il colore e la persistenza delle immagini sullo schermo offerti dal particolare tipo di fosfori. Il colore necessita di pochi commenti, mentre la persistenza è un concetto che occorre approfondire. La persistenza consiste nell'intervallo di tempo che intercorre tra l'illuminazione delle particelle di fosforo da parte del "pennello" elettronico, ed il momento nel quale esse perdono luminosità. Questa caratteristica viene raramente quantificata, e si parla spesso di "breve" o "lunga" persistenza.

I fosfori impiegati nelle televisioni ed in quasi tutti i monitor a colori hanno generalmente una persistenza breve, come del resto hanno anche i monitor monocromi. In alcune applicazioni, però, sia per la vista dello spettatore che per l'immagine generale, è preferibile impiegare schermi a lunga persistenza, che eliminano, almeno in parte, l'effetto di sfarfallio.

Una di queste applicazioni, forse la più nota, è negli schermi dei radar, dove l'immagine rimane visualizzata per qualche frazione di secondo, senza la necessità di ricorrere a complicate memorie di altro genere.

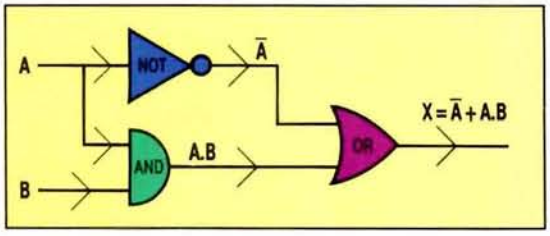
Comunque, nei terminali dotati di penne ottiche, è indispensabile che la persistenza sia molto breve, altrimenti l'impiego delle penne ottiche diviene impossibile. Infatti, queste leggerebbero non solo l'immagine vera e propria, ma anche le "rimanenze" di essa, dopo che, in effetti, l'immagine non esiste più! Ciò confonderebbe completamente il computer, che non saprebbe più calcolare con esattezza la posizione della penna rispetto allo schermo.

Esiste una vasta scelta nel colore dei fosfori impiegati per lo schermo dei monitor ed essa dipende soprattutto dal particolare tipo di applicazione cui è destinato l'apparecchio. Si possono produrre monitor con schermi di qualsiasi colore. Quelli con fosfori bianchi, a breve persistenza, sono i più economici, ma molto diffusi sono anche quelli con fosfori verdi oppure giallo-ambra, molto più rilassanti per la vista. Il blu viene spesso impiegato nei terminali in dotazione presso le agenzie di viaggi o gli uffici delle linee aeree, mentre il rosso è maggiormente impiegato negli schermi dei radar

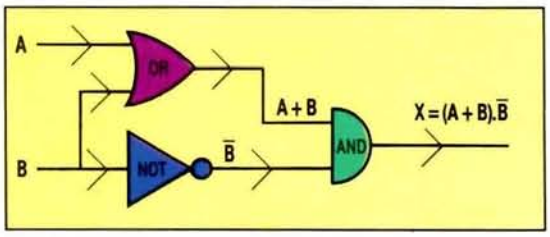
Sommare... con logica

L'aritmetica usata dai computer non è difficile

Approfondiamo, in questa lezione, alcuni aspetti dell'algebra di Boole applicata ai circuiti logici di un computer. Questa felice combinazione consente di svolgere le quattro operazioni aritmetiche fondamentali, dalle quali derivano, poi, tutte le altre.
Abbiamo visto come esprimere, secondo la notazione di Boole, le relazioni tra ingressi ed uscita delle porte logiche elementari: AND (A.B), OR (A+B) e NOT ($\overline{A}$). Possiamo usare queste notazioni per rappresentare circuiti molto più complessi. Ad esempio, l'espressione $X = \overline{A} + A.B$ corrisponde al circuito:



È da notare che l'ordine col quale vengono eseguite le operazioni AND ed OR è molto importante. Per convenzione, la funzione AND è prioritaria rispetto alla OR ed alla NOT. Desiderando modificare questo ordine di precedenza, si ricorre all'uso di parentesi. Ad esempio: $X = (A+B).\overline{B}$. In questa espressione, prima viene eseguita una OR sui due ingressi e sul risultato, infine, viene operato una AND con l'inverso di B. Ecco il circuito equivalente:



Per disegnare il circuito, partendo da un'espressione di Boole, conviene sempre partire dall'uscita e procedere all'indietro.

Or inclusivo ed esclusivo

In termini logici, la parola "oppure" (OR in inglese) può assumere due diversi significati. L'aspetto finora considerato si riassume in:

Un evento OPPURE un altro OPPURE ambedue.

Un secondo aspetto si riassume invece con:

Un evento OPPURE un altro ma non ambedue.

Per chiarire gli esempi: in un elenco di possessori di veicoli a due ruote, possiamo figurare come ciclisti OPPURE come motociclisti OPPURE in ambedue le categorie (OR inclusivo).
Viceversa (facendo un altro esempio), possiamo avere una statura alta OPPURE una bassa, ma non ambedue contemporaneamente (OR esclusivo).
L'operazione logica OR esclusivo (XOR) è uno strumento molto utile e la si ottiene mediante una combinazione delle altre operazioni logiche. La tavola della verità dell'operazione XOR è:

INGRESSI		USCITE
A	B	A⊻B
0	0	0
0	1	1
1	0	1
1	1	0

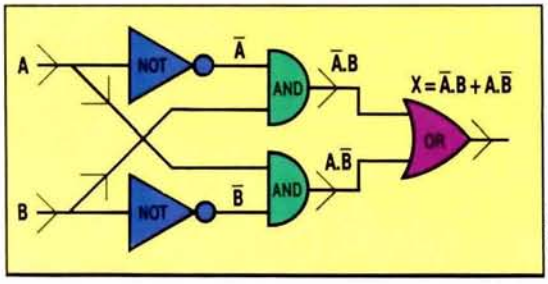
Riferendosi alla tabella, nella seconda e terza riga si ottiene un risultato pari ad uno se

su NOT(A) si opera un AND con B
OPPURE (OR)
su A si opera un AND con NOT(B)

l'espressione di Boole equivalente è:

$$X = \overline{A}.B + A.\overline{B}$$

Un circuito in grado di operare come OR esclusivo può essere il seguente:



LIZ DIXON

Questo è un circuito a cinque porte. In segui-

to vedremo come semplificarlo, impiegando solo quattro porte.

L'aritmetica e le porte logiche

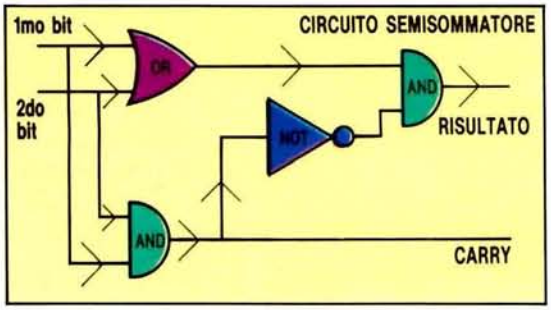
Benché i computer siano in grado di eseguire tutte le funzioni aritmetiche, la circuiteria interna è, in realtà, capace di fare soltanto delle somme.
Tutte le altre operazioni si ottengono mediante ingegnose combinazioni di porte logiche (i sommatori) e di funzioni controllate dal software.
Riassumiamo brevemente il procedimento di somma, nella notazione binaria:

	8	4	2	1
	0	1	0	1
	0	1	1	1
	1	1	0	0
Riporti	1	1	1	

Prendiamo una delle colonne (la seconda da destra, ad esempio). I dati "in ingresso" sono i due bit da sommare e l'eventuale riporto dalla colonna precedente, mentre quelli "in uscita" sono il bit del risultato e quello da riportare nella colonna successiva. Il circuito "sommatore" completo è piuttosto complesso, perciò cominciamo col descrivere il funzionamento del "semisommatore". Questo dispositivo differisce da quello "completo" semplicemente per il fatto che ignora i riporti da una colonna all'altra.
La tavola della verità per un semisommatore è:

INGRESSI		USCITE	
Primo bit	Secondo bit	Carry (Riporto)	Risultato
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Si nota subito che il Carry (Riporto) vale 1 solo se ambedue i bit d'ingresso sono ad 1 (funzione AND), mentre il Risultato deriva da un'operazione OR sui due ingressi, salvo il caso in cui il Carry valga 1.
Il circuito equivalente è:

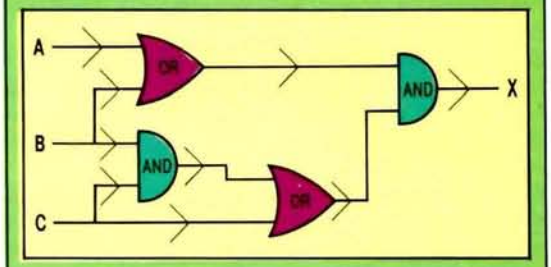


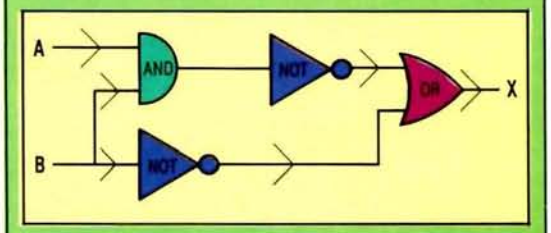
ESERCIZIO 2

1) Si disegni il circuito logico per le seguenti espressioni Booleane:
   a) $X = (A + B).C$
   b) $X = A.B + (A + C)$
   c) $X = \overline{A}.B + (A + B)$
   d) $X = \overline{A.B}.(A + B)$

2) Si scriva l'espressione Booleana corrispondente, nel circuito semisommatore, sia per il Carry che per il Risultato, usando A e B come ingressi.

3) Si scrivano le espressioni Booleane equivalenti ai due seguenti circuiti:





Risposte all'Esercizio 1 di pag. 489

1)

FATTO ESAME GUIDA	ACCOMPAGNATO DA UNO CON PATENTE	ABILITATO A GUIDARE
FALSO	FALSO	FALSO
FALSO	VERO	VERO
VERO	FALSO	VERO
VERO	VERO	VERO

2)

REGISTRATORE DISPONIBILE	UNITÀ DISCHI DISPONIBILE	VERSIONE PER ALTRO COMPUTER	PROGRAMMA ESEGUIBILE
FALSO	FALSO	FALSO	FALSO
FALSO	FALSO	VERO	FALSO
FALSO	VERO	FALSO	VERO
FALSO	VERO	VERO	FALSO
VERO	FALSO	FALSO	VERO
VERO	FALSO	VERO	FALSO
VERO	VERO	FALSO	VERO
VERO	VERO	VERO	FALSO

3)

A	B	P	Q	C
0	0	1	0	0
0	1	1	1	1
1	0	0	1	0
1	1	0	1	0

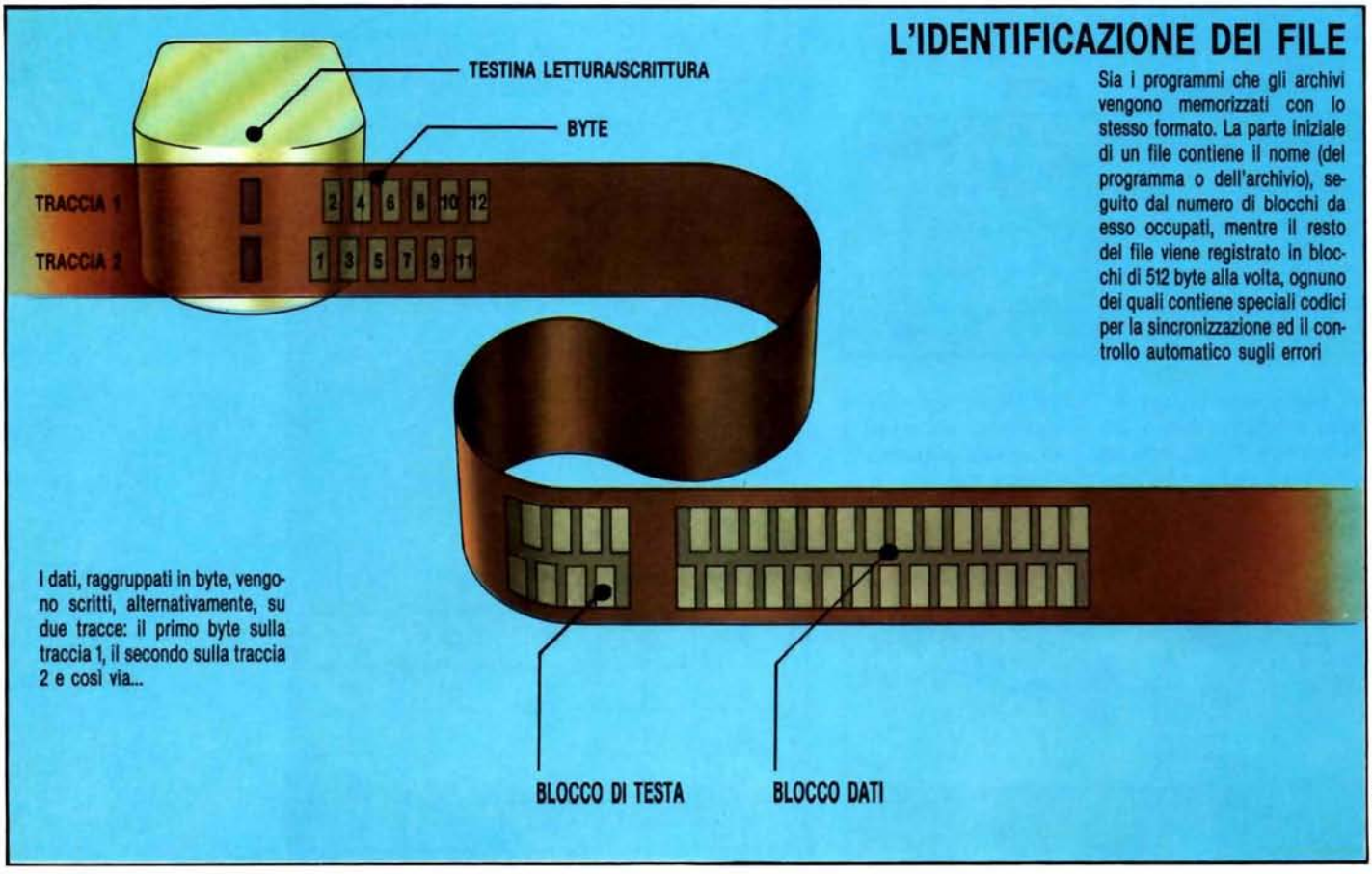
I Microdrive Sinclair

I principi di funzionamento di questa economica memoria di massa

Nel Microdrive della Sinclair viene impiegato un anello di nastro magnetico (del tipo usato per i videoregistratori) largo 2 mm. e lungo circa 8 metri. Un giro completo dell'anello, racchiuso in una speciale cartuccia, impiega circa 7 secondi.
Benché considerato (ed usato) come se si trattasse di un'unità a dischi, in realtà il Microdrive appartiene, dunque, alla categoria dei dispositivi noti col nome di "stringy floppy" o "floppy tape".
I dati vengono registrati sotto forma digitale e non analogica, come avviene normalmente con i registratori. Il nastro magnetico viene inizialmente "formattato" (ossia "preparato" da un apposito programma), creando due tracce, nelle quali i dati vengono memorizzati secondo un metodo a "zig zag" (vedere illustrazione).
Questo sistema, che richiede l'impiego di due testine di lettura/scrittura, permette di conservare le informazioni con una densità ed una velocità doppie rispetto ai nastri con una sola traccia. A sua volta, ogni traccia è suddivisa in blocchi di 256 byte, ognuno dei quali contiene una descrizione dei dati memorizzati ed è preceduto da 27 byte d'identificazione.
Il nome "settore" si riferisce ad un blocco, completo dei 27 byte preliminari. Il nastro del Microdrive può contenere fino a 200 settori, ossia, la sua densità è di circa 500 byte per pollice. Se un file è composto da più di 512 byte, esso occupa più settori e l'eventuale spazio inutilizzato, nell'ultimo settore, non è utilizzabile per un altro file. In teoria, quindi, ogni Microdrive può contenere 100 Kbyte di dati, ma in pratica la capacità varia tra gli 85 ed i 90 Kbyte. Il tempo medio occorrente per individuare e leggere un programma oscilla tra i 10 ed i 15 secondi.

L'interfaccia ZX

Per collegare i Microdrive allo Spectrum è necessaria l'Interfaccia ZX 1. Questa consente di utilizzare fino ad otto Microdrive con-



temporaneamente, oltre ad offrire una porta seriale RS232, e permettere un collegamento in rete locale di più Spectrum e l'uso di una stampante ZX.
Nelle ROM dell'Interfaccia 1 si trova anche il software necessario al funzionamento ed alla gestione di tutti questi apparecchi. L'Interfaccia 1 ed i Microdrive sono degli accessori molto interessanti, le cui prestazioni si avvicinano notevolmente a quelle di un sistema dotato di vere unità a dischi, ma ad un prezzo decisamente più economico. Gli unici svantaggi sono l'assenza di un vero e proprio accesso "casuale" ai dati e le incertezze dovute all'affidabilità, a lungo termine, dei nastri magnetici. Ciò pone dei limiti, pertanto, ad applicazioni che vadano molto oltre la didattica o l'hobbystica. Per di più, attualmente, la quantità di software in commercio, montato su Microdrive, è decisamente scarsa.



IAN McKINNELL

Lo "stringy floppy"
Le unità Microdrive sono dispositivi abbastanza interessanti, combinando l'economicità e la semplicità dell'archiviazione su nastro con una velocità d'operazione simile a quella dei dischetti. Per il funzionamento, occorre collegarli all'Interfaccia 1 della Sinclair

I COMANDI DEL MICRODRIVE

I comandi necessari al funzionamento del Microdrive sono:

FORMAT CAT SAVE* VERIFY* LOAD*
MERGE* ERASE OPEN# PRINT# INPUT#
INKEY$# CLOSE# MOVE

In ciascuno dei comandi che seguono, M seleziona il Microdrive, N il numero identificativo del Microdrive al quale accedere (1-8) ed S il numero di canale associato al file.

FORMAT
Serve per la preparazione di un Microdrive prima dell'uso o per cancellare e riutilizzare un Microdrive. Il suo formato è:

FORMAT "M";N;"NOME"

oppure

FORMAT M$;N;C$

dove NOME è il nome assegnato alla cartuccia (massimo 10 caratteri), M$ = "M" e C$ contiene il NOME.

CAT
Questo comando elenca sullo schermo (o su diverso dispositivo) i nomi dei file contenuti in una cartuccia.

CAT N

oppure

CAT #S;N

L'elenco è formato dal nome della cartuccia seguito da un massimo di 50 nomi di file e dal numero di Kbyte ancora disponibili.

SAVE*
Permette di memorizzare programmi o archivi dati. Il formato è:

1. SAVE* "M";N;"NOMEFILE"
2. SAVE* "M";N;"NOMEFILE" SCREEN$
3. SAVE* "M";N;"NOMEFILE" DATA A( )
4. SAVE* "M";N;"NOMEFILE" LINE X

per memorizzare, rispettivamente:

1. un file
2. un file consistente di SCREEN* (schermi)
3. un file di dati A( )
4. un programma la cui linea di partenza è la X

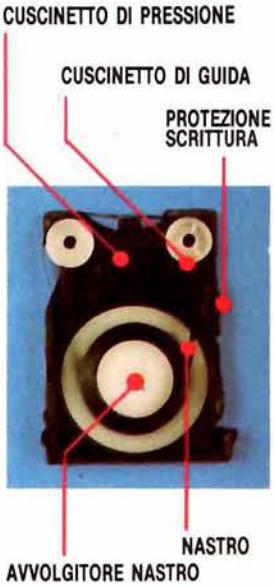
VERIFY*, MERGE* ed ERASE*
Questi comandi hanno lo stesso formato di 1. SAVE* appena visto. VERIFY* confronta il contenuto di un file con quanto è presente in memoria, generando un messaggio d'errore in caso di differenza. MERGE* accoda "NOMEFILE" al programma in memoria. ERASE* serve per cancellare il file "NOMEFILE".

LOAD*
Serve per leggere un file dal nastro. Il formato è identico a 1. SAVE e 2. SAVE visti precedentemente.

OPEN#, PRINT#, INPUT#, INKEY$#, CLOSE# e MOVE
Servono alla gestione degli archivi di dati. Questi vengono conservati sequenzialmente, ma è possibile simulare alcune proprietà dell'accesso "casuale". Il flusso dei dati viene regolato "aprendo" (OPEN) e "chiudendo" (CLOSE) un canale di lettura/scrittura. Ad esempio:

OPEN#S;"M";N;"NOMEFILE"

associa al numero S il file "NOMEFILE" sul Microdrive N. A questo punto si può scrivere nel file (PRINT#S), o si può leggere da esso (INPUT#S, INKEY$#S). Il comando MOVE si usa per trasferire file all'interno di un Microdrive, da una cartuccia all'altra (o ad altro dispositivo individuabile con un numero di canale).
Quando il lavoro su di un file è concluso, occorre "chiudere" il canale (CLOSE#S).



La cartuccia del Microdrive
Per assicurare una maggiore affidabilità, il nastro magnetico impiegato nelle cartucce dei Microdrive è del tipo adoperato per i videoregistratori. La superficie del nastro è trattata con una sostanza che facilita lo scorrimento sulla testina di lettura/scrittura. Come i nastri a cassetta ed i dischetti, anche questo tipo di cartucce è dotato di una protezione, che disabilita, se necessario, la scrittura sul nastro

I banchi di memoria

La memoria dei computer è suddivisa in varie sezioni o "banchi"

Nella prima lezione sul linguaggio macchina, abbiamo visto come una serie di informazioni possa essere memorizzata sotto forma di segnali elettrici: un gruppo di lampadine e di interruttori è servito per codificare certi eventi in una numerazione di tipo binario.
Nel particolare esempio, le possibili combinazioni di lampadine accese o spente, sedici in tutto, si potevano rappresentare mediante un numero compreso tra 0 e 15. Disponendo, invece, di 8 interruttori ed altrettante lampadine, otterremmo 256 diverse combinazioni

(2×2×2×2×2×2×2×2=256)

rappresentabili con un numero compreso tra 0 e 255.
La memoria di un computer è paragonabile ad una serie di interruttori, organizzati in gruppi di otto alla volta: ciascun gruppo corrisponde ad un byte.
In generale, la CPU elabora le informazioni un byte alla volta, ossia, essa può sommare, confrontare e memorizzare soltanto dei numeri compresi tra 0 e 255. Ciò può sembrare una forte limitazione alle capacità aritmetiche di un computer, ma in realtà, considerata anche l'elevata velocità di funzionamento, il problema non causa alcuna conseguenza apprezzabile. Semplicemente, come del resto avviene con carta e penna, i numeri di grosse dimensioni vengono elaborati una cifra alla volta.
Un byte è composto da otto bit, le più piccole unità d'informazione.
Un bit può assumere due stati logici: OFF (spento) oppure ON (acceso) e ciò in binario equivale, rispettivamente, al valore 0 ed al valore 1.
Per individuare con esattezza la posizione di ciascun bit, all'interno di un byte, si adopera una convenzione, che consiste nel numerare i bit da 0 a 7, partendo da destra. Ossia, se il byte contiene il valore 00000001, si dice che il bit0 è 1 (oppure ON, oppure SET), mentre gli altri bit sono 0 (oppure OFF, o RESET, o CLEAR).
Cosicché, nel byte 01001000, il bit3 è SET, il bit6 è SET, il bit4 è OFF, il bit7 è 0, il bit0 è CLEAR e così via. Inoltre, il bit0 è anche chiamato il "bit meno significativo" o LSB (dall'inglese Least Significant Bit), mentre il bit7 è il "bit più significativo" o MSB (dall'inglese Most Significant Bit).
La memoria di un computer, pertanto, la si può concepire come una lunga striscia di carta quadrettata, larga otto quadretti e lunga, magari, migliaia di quadretti: ogni riga di otto quadretti corrisponde ad un byte ed ogni quadretto ad un bit. La memoria è utile soltanto se si è in grado di localizzare un particolare dato, quindi ad ogni byte è associato un "indirizzo".
L'indirizzo non è che il numero progressivo del byte, iniziando da 0: desiderando scrivere un valore nel byte 43, si comincia perciò dal byte 0 e si contano i successivi byte fino al byte numero 43.
Non esiste nessun altro riferimento: l'indirizzo è unicamente determinato dal numero progressivo attribuito.
In effetti, ogni byte di memoria è costituito da otto microscopici circuiti elettronici, paragonabili ad interruttori transistorizzati.
Esiste, tuttavia, un inconveniente nell'impiegare un simile metodo d'indirizzamento: la CPU è rapidissima nel contare da 1 a 100 (qualche frazione di millisecondo), ma per raggiungere, ad esempio, 20.000 il tempo richiesto inizia ad essere eccessivo. L'ostacolo viene aggirato dividendo la memoria in "pagine", proprio come in un libro.

Tornando all'esempio della carta quadrettata, possiamo supporre di tagliarla trasversalmente in più strisce, ciascuna composta da una serie di 100 byte. Adesso, la pagina 0 inizia col byte 0 e termina col byte 99, la pagina 1 inizia col byte 100 e termina col byte 199 e via discorrendo.
Per localizzare un particolare byte (il 3518, ad esempio) non è necessario iniziare il conteggio dal byte 0: si nota subito dall'indirizzo del byte che esso si trova nella pagina 35. Basta quindi contare 35 pagine e, successivamente, contare 18 byte dall'inizio della 35ma pagina.
Questo sistema, in un computer, risulta molto conveniente, poiché la suddivisione dell'indirizzo in due parti (la pagina e la posizione del byte all'interno di essa) consente di risparmiare notevoli quantità di tempo nella localizzazione delle informazioni. La posizione relativa del byte viene anche chiamata "offset" (=scarto).

Rimane un'ultima considerazione da fare: la CPU non ragiona usando i numeri decimali, bensì quelli binari, perciò una numerazione di pagine che vada di 100 in 100 (decimale) non è molto conveniente. In binario, infatti, il valore decimale 99 vale 01100011, 100 vale 01100100, 199 vale 11000111. Si potrà notare che non esiste un metodo immediato per apprezzare il passaggio da 99 a 100 o da 199 a 200, come esiste invece nella numerazione decimale. Per il nostro esempio, abbiamo scelto di numerare le pagine di 100 in 100 perché ciò è più comodo per un essere umano, abituato al sistema decimale. Lavorando in un sistema binario, occorre necessariamente scegliere una misura di pagina più idonea. La misura generalmente adottata è 256 (in decimale): la pagina 0 inizia col byte 0 e termina col byte 255, la pagina 1 inizia col byte 256 e termina col byte 511 e così via. Per apprezzare la convenienza di tale numerazione si convertano gli indirizzi in binario:

Pagina 0: byte 00000000 - byte 11111111

Pagina 1: byte 00000001 - byte 11111111

Se consideriamo gli otto bit più a destra come offset (posizione del byte nella pagina), il numero della pagina si individua facilmente usando i bit che partono dall'ottavo e procedono verso sinistra: ad ogni "salto di pagina" corrisponde lo "scatto" di un bit.

Riassumendo, l'indirizzo di un byte di memoria è costituito da due byte: il primo rappresenta la "pagina", il secondo l'offset. Se adesso riportiamo in binario la suddivisione delle pagine otteniamo:

Pagina 0: inizia da 00000000 00000000
finisce a 00000000 11111111

Pagina 1: inizia da 00000001 00000000
finisce a 00000001 11111111

Pagina 10: inizia da 00000010 00000000
finisce a 00000010 11111111

Pagina 11: inizia da 00000011 00000000
finisce a 00000011 11111111

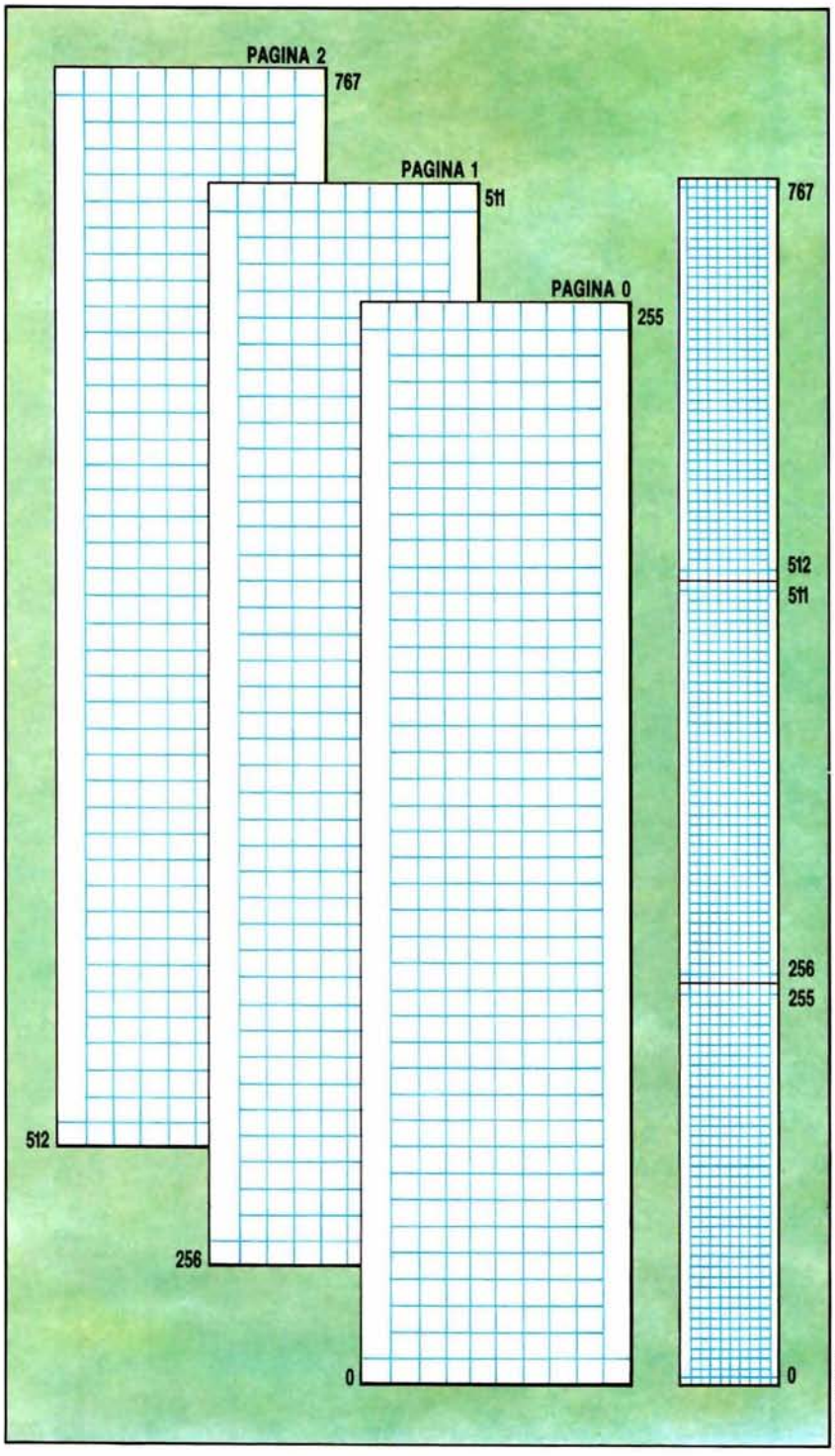
(dove il numero di pagina è espresso anch'esso in binario)

Per la CPU, che può operare soltanto su di un byte alla volta, questo sistema risulta molto conveniente: per accedere ad un byte di memoria, il microprocessore esamina per prima la pagina (1 byte) e, una volta selezionata questa, esamina l'offset (1 byte).

Nella pagina seguente, presentiamo dei programmi utili alla conversione di valori da decimale in binario ed in esadecimale (un comodo sistema di numerazione che tratteremo più avanti nel corso).

INDIRIZZAMENTO DELLA MEMORIA

L'indirizzamento basato sulla suddivisione della memoria in "pagine" di 256 byte risulta il più comodo per la CPU di un computer. Tutti gli indirizzi, infatti, possono essere espressi sotto forma di due byte: il primo identifica la pagina di memoria, il secondo individua, all'interno di essa, il particolare byte al quale si desidera accedere



KEVIN JONES

FACILE CONVERSIONE

I tre programmi a fianco per il Micro BBC, per il Commodore 64 e per lo Spectrum, servono per convertire i numeri decimali nella forma binaria

COMMODORE 64

```
10 REM*************** COMMODORE 64 ****************
20 S$="          ":X$="0123456789ABCDEF"
50 REM S$ CONTIENE 9 SPAZI
60 PRINT CHR$(147)
70 PRINT "PER STAMPARE NUMERI DECIMALI"
80 PRINT "E I LORO EQUIVALENTI BINARI"
90 PRINT:PRINT "PER USCIRE DIGITARE 0":PRINT
100 FOR K=1 TO 1
110 FOR L=1 TO 1
120 INPUT "DIGITA UN NUMERO INTERO POSITIVO";A$
130 NU=VAL(A$)
140 IF NU=0 THEN PRINT "FINE PROGRAMMA":STOP
150 IF INT(NU)<>ABS(NU) THEN L=0
160 IF NU>65535 THEN PRINT NU;" E' TROPPO GRANDE":L=0
170 NEXT L
200 NM=NU:H$="":GOSUB 2000
210 PRINT NU;TAB(5);N$;
220 IF RIGHT$(A$,1)="+" THEN GOSUB 4000
230 PRINT H$:PRINT:PRINT
240 K=0:NEXT K
250 END
1000 REM*************** BYTE BINARIO ****************
1010 B$=""
1020 FOR D=8 TO 1 STEP -1
1030 N1=INT (N/2)
1040 R=N-2*N1
1050 B$=MID$(STR$(R),2)+B$
1060 N=N1
1070 NEXT D
1080 RETURN
2000 REM************ CONVERSIONE BINARIA ************
2010 IF NM<256 THEN N=NM:GOSUB 1000:N$=S$+B$:RETURN
2020 HI=INT(NM/256):LO=NM-256*HI
2030 N=HI:GOSUB 1000:N$="= "+B$
2040 N=LO:GOSUB 1000:N$=N$+" "+B$
2050 RETURN
3000 REM***************** BYTE HEX *****************
3010 HB=INT(N/16):LB=N-16*HB
3020 B$=MID$(X$,HB+1,1)+MID$(X$,LB+1,1)
3030 RETURN
4000 REM************** CONVERSIONE HEX **************
4010 IF NM<256 THEN N=NM:GOSUB 3000:H$="= "+B$:RETURN
4020 HI=INT(NM/256):LO=NM-256*HI
4030 N=HI:GOSUB 3000:H$="= "+B$
4040 N=LO:GOSUB 3000:H$=H$+" "+B$
4050 RETURN
```

MICRO BBC

Copiare il precedente listato apportando le seguenti modifiche:

```
60 CLS:@%=5
210 PRINT TAB(0);NU;TAB(5);N$;
1050 B$=STR$(R)+B$
```

Questo programma non sfrutta a pieno le capacità che il BBC possiede per le rappresentazioni dei numeri

SPECTRUM

```
10 REM*************** SPECTRUM ***************
20 LET S$="          "
30 LET X$="0123456789ABCDEF"
50 REM S$ CONTIENE 9 SPAZI
60 CLS
70 PRINT "PER STAMPARE NUMERI DECIMALI"
80 PRINT "E I LORO EQUIVALENTI BINARI"
90 PRINT:PRINT "PER USCIRE DIGITARE 0":PRINT
100 FOR K=1 TO 1
110 FOR L=1 TO 1
120 INPUT "DIGITA UN NUMERO INTERO
    POSITIVO";A$
130 LET NU=VAL(A$)
140 IF NU=0 THEN PRINT "FINE PROGRAMMA":STOP
150 IF INT(NU)<>ABS(NU) THEN LET L=0
160 IF NU>65535 THEN PRINT NU;" E' TROPPO
    GRANDE":LET L=0
170 NEXT L
200 LET NM=NU:LET H$="":GOSUB 2000
210 PRINT NU;TAB(5);N$;
220 IF A$(LEN A$)="+" THEN GOSUB 4000
230 PRINT H$:PRINT:PRINT
240 LET K=0:NEXT K
250 END
1000 REM********** BYTE BINARIO ************
1010 LET B$=""
1020 FOR D=8 TO 1 STEP -1
1030 LET N1=INT (N/2)
1040 LET R=N-2*N1
1050 LET B$=STR$(R)+B$
1060 LET N=N1
1070 NEXT D
1080 RETURN
2000 REM******** CONVERSIONE BINARIA ********
2010 IF NM<256 THEN LET N=NM:GOSUB 1000:
     LET N$=S$+B$:RETURN
2020 LET HI=INT(NM/256):LET LO=NM-256*HI
2030 LET N=HI:GOSUB 1000:LET N$="= "+B$
2040 LET N=LO:GOSUB 1000:LET N$=N$+" "+B$
2050 RETURN
3000 REM************** BYTE HEX **************
3010 LET HB=INT(N/16):LET LB=N-16*HB
3020 LET B$=X$(HB+1)+X$(LB+1)
3030 RETURN
4000 REM********** CONVERSIONE HEX **********
4010 IF NM<256 THEN LET N=NM:GOSUB 3000:
     LET H$="= "+B$:RETURN
4020 LET HI=INT(NM/256):LET LO=NM-256*HI
4030 LET N=HI:GOSUB 3000:LET H$="= "+B$
4040 LET N=LO:GOSUB 3000:LET H$=H$+" "+B$
4050 RETURN
```

Se al numero immesso viene fatto seguire un "+" (ad esempio 6435+), oltre alla forma binaria, viene anche visualizzata quella esadecimale

Il successo della Atari

Come molte altre grandi ditte del settore, la Atari è frutto dell'ingegno di un solo uomo

Il rivoluzionario avvento dei videogiochi, nei quali chi siede di fronte ad un televisore diventa attivo partecipe al gioco, deriva da un gioco molto semplice: il *Pong*. L'ideatore di questo gioco è Nolan Bushnell.
Bushnell e i suoi due soci, Ted Dabney e Larry Bryan, investirono circa 200.000 lire a testa nel lancio di *Pong*, che, inizialmente, venne distribuito nelle sale giochi di Sunnyvale, in California. La grandissima fortuna commerciale della nota casa americana Atari proviene in gran parte dalla decisione di acquistare i diritti per l'invenzione di Bushnell.
La Atari ha dominato il mercato dei videogiochi per tutti gli anni '70, fino a quando l'interesse del pubblico non si è spostato verso gli home computer. La commercializzazione dei giochi è simile, per molti aspetti, a quella di dischi musicali: occorre individuare il potenziale "divo" e promuoverlo su larga scala. Non a caso, la Atari è proprietà della Warner Communication International, un nome ben noto nel campo dello spettacolo.
Gli introiti della Atari, nel settore delle macchinette da sale giochi, hanno avuto una impennata alla fine degli anni '70, ma durante il 1983 hanno subito un calo del 25%, causando forti perdite.
Il gioco più noto, commercializzato dalla Atari ma prodotto dalla Taito, è lo *Space Invaders*, divenuto quasi un fenomeno sociale e progenitore di una fertile messe di varianti coronate anch'esse da un enorme successo.
Le cose sono cambiate, però, quando sul mercato sono arrivati gli home computer. Questi apparecchi, rispetto ai videogiochi tradizionali, offrono due vantaggi: si possono usare gli stessi giochi da "sala giochi" (senza spenderci una fortuna) ed, al tempo stesso, si possiede un elaboratore utilizzabile in tutt'altre applicazioni.
Inizialmente, la Atari è corsa ai ripari convertendo i giochi di maggior successo in versioni adatte ai microcomputer, in genere su cartucce ROM. Ma non essendo queste riprogrammabili, i magazzini si sono ben presto riempiti di una vasta quantità di materiale inutile, ossia di giochi su cartucce dallo scarso successo.



per concessione della ATARI

Nolan Bushnell
Gran parte della fortuna iniziale la Atari la deve a Nolan Bushnell, inventore del Pong, il primo videogioco commercializzato su larga scala. Il gioco consisteva in una elementare simulazione del Ping Pong e le "racchette" venivano manovrate dai progenitori degli attuali joystick

Una fortuna in declino

Si è trattato, evidentemente, di un errore nella politica commerciale della casa americana. Pare che, a conti fatti, ben 14 camion carichi di giochi su cartucce (dal prezzo oscillante tra le 12.000 e le 60.000 lire) siano stati scaricati in un'enorme fossa nel deserto del Nevada.
Un'altra fonte di ingenti perdite, per la Atari, è stato il fenomeno dilagante della riproduzione illegale dei programmi.
Nuove tecnologie hanno permesso la costruzione di apparecchiature per la trasmissione di dati via cavo, mediante linee telefoniche, o addirittura, via etere. Nel 1983, per esempio, la Romox Corporation ha messo in commercio un apparecchio, chiamato il "Romox Programming Terminal", composto da un disco rigido da 15 Megabyte e da circuiti predisposti alla comunicazione, mediante linee telefoniche, con una grande banca dati di giochi. Essendo dotato, inoltre, di un programmatore per EPROM fornito dei connettori per cartucce più diffusi, il Romox è divenuto una fonte inesauribile di riproduzioni non autorizzate di software.
Alternativamente, gli utenti di home computer statunitensi si possono abbonare a speciali servizi telefonici, tipo il "Gameli-

Le consolle per videogiochi
Il successo dei videogiochi domestici è in netto declino. Infatti, questi apparecchi, pur essendo dotati di tutte le componenti di un microcomputer, non possono svolgere altre funzioni che non il gioco. Per il funzionamento sono necessarie delle cartucce ROM (in genere una per gioco)



ne" ed usare i giochi di proprio gradimento, pagando una tariffa oraria.
Esistono molti servizi di questo genere negli USA e, presto, anche in Europa si diffonderanno delle reti nazionali ed internazionali per lo scambio o l'utilizzo di banche dati contenenti giochi.
Due delle maggiori reti informatiche statunitensi, Compuserve e The Source, offrono tra i propri servizi anche l'accesso ad una vasta quantità di software ricreativo. Gli utenti di tali servizi non hanno che da collegare il proprio home o personal computer alla banca dati mediante un modem, collegato a sua volta alle normali linee telefoniche.
La Coleco, una famosa ditta produttrice di videogiochi, si è unita alla AT&T (American Telephone & Telegraph) onde fornire un servizio ricreativo interattivo per il tempo libero. La Atari, analogamente, ha tentato di concludere un simile accordo con la Activision, per l'immissione di programmi di gioco, mediante l'uso di linee telefoniche, nell'Ataritel, un progetto, fino ad ora, coperto dal più stretto riserbo. Finché la Atari farà parte del gruppo commerciale che fa capo alla Warner Communications non vi saranno problemi, ma che cosa accadrebbe se alla Atari — che sembra contare moltissimo sul progetto Ataritel — venisse meno l'accesso alla vastissima rete di comunicazione Warner Amex Cable Communications, non è facile da prevedere.
La Atari, comunque, continua la produzione dei noti home computer, nei quali vengono adoperati speciali chip per la gestione della grafica, degli effetti audio e dei canali d'ingresso/uscita.
Tutti gli apparecchi finora prodotti sono basati sul microprocessore 6502 ed esiste in commercio una vasta scelta di software di ottima qualità. Ultimamente, è stata realizzata una scheda aggiuntiva, dotata di un microprocessore Z80, aprendo le porte al sistema operativo CP/M della Digital Research.
Gli interessi della Atari si estendono anche al settore software: molti dei programmi di maggior successo commerciale vengono adattati agli apparecchi Atari e vi è una costante ricerca di programmatori, o, per meglio dire, di programmi nuovi da inserire tra le offerte indirizzate agli acquirenti dei computer.

IL "MAJOR HAVOC"

Durante gli anni '70, la Atari si era specializzata negli apparecchi da sale giochi, come quella qui raffigurata. L'avvento degli home computer ha costretto, però, a rivedere la strategia di commercializzazione di tali prodotti



Software per divertimento

I programmi generatori di giochi aprono un nuovo orizzonte sul vasto mercato del software ricreativo

IL "GAMES DESIGNER"



Il formato di base

Gli sprite appaiono in un lento ed ordinato movimento su uno sfondo omogeneo

Effetto finale

Sul formato di base sono state apportate diverse modifiche, l'unico elemento rimasto intatto è il gatto



La creazione degli sprite

Con semplici operazioni si possono definire nuove forme: sulla destra si ha un'immagine ingrandita dello sprite, mentre in basso lo sprite è visualizzato in dimensioni reali



Menu di configurazione

Permette di selezionare i parametri relativi al movimento delle varie "navette spaziali", i colori dello sfondo e dei vari oggetti, la forma del "nemico", ed i vari effetti sonori



Menu del movimento

In base alle varie indicazioni riprodotte sullo schermo, è possibile definire un modello, o una combinazione di più modelli, relativi al movimento delle forze nemiche



IAN McKINNELL

I programmi di giochi per home computer si possono distinguere in due grosse categorie: quelli cosiddetti "d'avventura", nei quali lo svolgimento non è necessariamente accompagnato da una sofisticata grafica, e quelli, invece, dove la grafica, sfruttata al massimo, costituisce il cardine di tutta l'azione.
Si tratta, in questo caso, del genere di giochi al quale appartengono i vari *Space Invaders*, *Asteroids*, ecc., tutti più o meno simili tra loro. Infatti, le due caratteristiche fondamentali di tutti i giochi di questo tipo sono la presenza di un "posto di combattimento", manovrato dal giocatore, e di una serie di "oggetti" da colpire e distruggere. Pertanto, il progetto di questi giochi si riduce alla definizione di una serie di parametri, che influiscono sui due elementi appena citati, ad esempio: la posizione del posto di combattimento, le caratteristiche delle armi impiegate, la frequenza degli attacchi nemici e via dicendo. Se disponessimo di un programma che permettesse di intervenire su tali parametri, potremmo facilmente creare un'infinità di variazioni "personalizzate".
Un primo esempio di programmi "generatori" di questo tipo è il *Games Designer* (realizzato per lo Spectrum con 48 Kbyte dalla Software Studios/Quicksilva). Il *Games Designer* offre una scelta tra otto schemi di gioco, nei quali è possibile variare tutti i parametri principali, sebbene ciò non significhi, in realtà, creare di sana pianta un nuovo gioco. Per facilitare le operazioni, il programma è pilotato da menu, dai quali l'operatore seleziona le caratteristiche desiderate.
Operata la prima scelta, ossia uno fra gli otto schemi di gioco, sul video compaiono le indicazioni necessarie per variare le caratteristiche specifiche, iniziando dalla forma e dal

colore degli "sprite" usati per rappresentare i "protagonisti" del gioco. La forma degli sprite, costruita su di una matrice 12 × 12 (ossia 2 × 2 caratteri) viene selezionata tra quelle già definite all'interno del programma, pur avendo la facoltà di modificarla a piacimento. Il formato di 2 × 2 caratteri offre una vasta gamma di variazioni sulla quale sbizzarrirsi. Anche la definizione degli sprite è pilotata da menu: a sinistra, sullo schermo, compaiono i comandi disponibili, mentre sulla parte destra è riprodotta un'immagine ingrandita dello sprite. Per maggiore convenienza, sulla parte inferiore dello schermo viene visualizzato lo sprite in dimensioni reali (così come apparirà nel gioco).



IAN McKINNELL

Avventure a volontà

Il generatore di giochi "The Quill" permette di creare originali giochi d'avventura. Il suo funzionamento è molto simile a quello dei programmi per la gestione di database: i vari elementi del gioco vengono prima definiti, poi ad essi vengono associati vari attributi che formano la struttura del gioco. Alcuni dei giochi d'avventura oggi in commercio sono stati creati proprio impiegando questo generatore

Il passo successivo è quello di stabilire come i vari sprite debbano muoversi sullo schermo e ciò avviene durante la fase chiamata "configurazione", nella quale si definiscono anche altre specifiche del movimento. Sempre in base alle opzioni presentate da un menu, si possono selezionare dei modelli di movimento.

A questo punto si passa a stabilire la frequenza degli "attacchi" nemici, definendo, al tempo stesso, gli effetti speciali sia audio che video.

Terminata la definizione di tutti i parametri, l'ultima operazione, prima di lanciarsi nel nuovo gioco, è quella di memorizzare su nastro tutte le caratteristiche, in modo da poterle riutilizzare più volte. Il *Games Designer*, per chi desiderasse generare giochi con lo scopo, poi, di venderli, ha un punto debole. Infatti, sul nastro non viene memorizzato il gioco in forma eseguibile, ma soltanto i parametri definiti nel corso della creazione, e ciò significa che non lo si può utilizzare se non si possiede anche il programma generatore.

Appartenente a tutt'altro genere è invece il programma *The Quill* (letteralmente "la penna d'oca") che, oltre ad essere orientato alla creazione di giochi d'avventura, permette di generare giochi completi ed eseguibili, facilmente commercializzabili. Addirittura, un capitolo del manuale è proprio dedicato ad una serie di utili consigli per creare giochi in forme adatte alla vendita. L'unica condizione, imposta dagli autori di *The Quill* a chi desideri commercializzare giochi creati con questo programma, è quella di menzionare, in qualche punto del gioco, che esso è stato creato con *The Quill*. Questo è un fatto abbastanza insolito, considerando che, in casi simili, viene generalmente imposta una "royalty".

Benché fondamentalmente concepito come un programma generatore di giochi d'avventura sul tipo di *Dungeons and Dragons*, la sua struttura ed il suo funzionamento assomigliano molto a quelli dei programmi per la gestione di database commerciali. Il programma si compone di tre parti distinte: un database (ossia l'archivio dei parametri), un programma che consente la modifica dei parametri del gioco ed, infine, un programma "interprete", per rendere il gioco eseguibile.

Così come abbiamo ridotto, concettualmente, i giochi del genere precedente (*Space Invaders* e simili) alla definizione di due caratteristiche fondamentali, anche nei giochi d'avventura, in sostanza, il principio di base può essere riassunto nella creazione di un "labirinto" a più dimensioni (non soltanto limitate nello spazio). Cosicché il giocatore, oltre a dover trovare l'uscita dal labirinto, deve anche scegliere opportunamente quali oggetti, tra un elenco di quelli disponibili in una certa fase del gioco, "portarsi dietro". L'utilità degli oggetti varia a seconda delle circostanze e, per rendere più avvincente l'avventura, la quantità degli oggetti "trasportabili" viene spesso limitata, imponendo quindi molta oculatezza nella loro scelta.

La prima fase, nella creazione di un gioco d'avventura, consiste nel redigere un copione "di massima", nel quale viene descritto lo scenario ed i vari percorsi all'interno di esso. Il manuale di questo particolare generatore, *The Quill*, fornisce un esempio abbastanza dettagliato sul modo di procedere, per creare un'avventura ambientata in un labirinto di sei "stanze", ciascuna con i propri trabocchetti e pericoli di vario genere.

Il giocatore dispone di dieci diversi oggetti, alcuni indispensabili durante particolari fasi del gioco. I parametri del gioco presentato come esempio non sono già incorporati nel programma, cosicché s'inizia immediatamente a far pratica con l'inserimento delle varie caratteristiche, come se si trattasse di un gioco creato di sana pianta.

Esistono delle limitazioni nei parametri che si possono impiegare, ma esse sono talmente ampie (252 "luoghi" e 210 "oggetti" diversi) da non rappresentare, in realtà, un vero ostacolo allo sviluppo di complesse avventure.

I giochi creati con questo generatore hanno un livello pari a quelli normalmente in commercio (alcuni dei quali creati proprio con *The Quill*), tuttavia esiste una notevole limitazione per quanto riguarda la grafica. Infatti, non è prevista la definizione grafica di immagini con le quali accompagnare le varie fasi del gioco.

Il chip Motorola 68000

Le caratteristiche di un microprocessore di sicuro successo

Il chip 68000 ha una forte rassomiglianza con un suo predecessore, il Motorola MC 6800, ancora largamente impiegato in molte apparecchiature di controllo. Ciò significa che, pur avendo prestazioni nettamente superiori, il 68000 può essere facilmente interfacciato con tutta una serie di apparecchiature già presenti sul mercato. Tra queste, si possono citare le schede d'ingresso/uscita basate sui chip 6821 (noti come PIA o Peripheral Interface Adaptor), i video dotati di chip 6845 (comunemente indicati con la sigla CRTC, ossia Cathode Ray Tube Controller), i temporizzatori basati sull'impiego di timer programmabili 6840 ed, infine, schede per le unità a dischi.

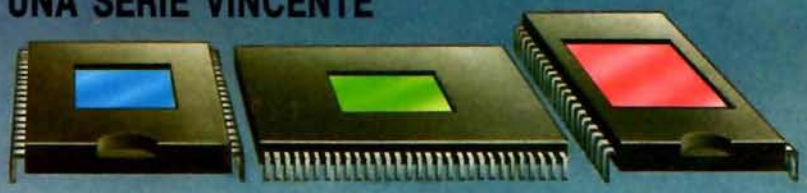
Una caratteristica che gioca a netto favore del 68000 è l'ampiezza sia del bus dei dati che di quello degli indirizzi. Questi, oltre ad essere fisicamente separati tra loro, dispongono di un piedino di connessione per ciascun segnale, a differenza di quanto accade con i chip a 16 bit del tipo 8086, 8088 e Z8000. In questi chip, per limitare il numero di piedini, i segnali vengono "multiplexati", ossia, mediante un opportuno sistema di sincronizzazione, vengono alternativamente applicati ad un limitato numero di piedini.

Il microprocessore 68000, dunque, riesce a lavorare a frequenze assai maggiori rispetto ad altri chip consimili e, qualora venga impiegato con memorie RAM altrettanto veloci (sull'ordine dei 50 o 90 nanosecondi di risposta), i tempi "morti", ossia sprecati in inutili attese, vengono ridotti al minimo se non addirittura eliminati. Il chip più veloce, nella famiglia dei 68000, è il 68000L12, col quale è possibile raggiungere frequenze di lavoro che si aggirano sui 14 Megahertz.

La Sinclair Research, per il QL, ha impiegato il 68008. Internamente questo chip non differisce molto dagli altri della stessa serie, ma, per renderlo compatibile con i precedenti sistemi ad 8 bit, il suo bus dei dati è, appunto, composto da 8 bit, invece dei normali 16. Essendo il numero di segnali così limitato, il chip viene montato in comuni involucri con 40 piedini.

La Motorola sta approntando, comunque, un microprocessore ancora più potente: il 68020, a ben 32 bit, contenuto in un involucro con 96 piedini, la forma del quale non è stata ancora ben definita. Parallelamente, è in via di sviluppo il 68881, uno speciale processore per funzioni matematiche a virgola fluttuante, dotato di 8 registri (ciascuno composto da 80 byte), che dovrebbe aumentare considerevolmente la quantità di dati numerici elaborabili dal computer.

UNA SERIE VINCENTE



Il 6502

Sviluppato dalla MOS Technology, il processore 6502 è divenuto, insieme con lo Z80, uno standard industriale. Esso utilizza un bus degli indirizzi a 16 bit ed un bus dei dati ad 8 bit. Una peculiarità del 6502 è la struttura dei registri interni: esiste un solo accumulatore, ma l'intera pagina 0 può essere sfruttata suddividendola in registri

Il 68000

La Motorola ha riprogettato completamente il vecchio 6800, producendo il 6809, ma non lo ha fatto in tempo utile per accaparrarsi una consistente fetta di mercato. Ciò si è rivelato, in fin dei conti, un fatto positivo, poiché ha stimolato il progetto del 68000, un microprocessore a 16/32 bit. Molto più potente di altri chip della stessa categoria, il 68000 si può facilmente interfacciare con i chip della serie 6800

Lo Z80

Virtualmente molto più potente del 6502, col quale condivide grosso modo la stessa architettura, lo Z80 è dotato di un gruppo di registri molto più ampio: 12 registri generici ad 8 bit e 2 speciali registri a 16 bit per l'indirizzamento. Anche l'insieme delle istruzioni è molto più vasto. Il grandissimo pregio dello Z80 deriva dalla possibilità d'impiegare il sistema operativo CP/M della Digital Research

KEVIN JONES

La serie 68000 comprende anche un certo numero di chip per le funzioni di ingresso/uscita molto simili, ma con migliori prestazioni, a quelli della generazione precedente.

Dal punto di vista dei programmatori, il 68000 offre molti vantaggi in più rispetto agli altri chip a 16 bit, principalmente dovuti alla simmetria dei registri degli indirizzi e dei dati ed alla notevole ricchezza di istruzioni.

Cio nonostante, nessuno è perfetto. In primo luogo, esiste una distinzione tra i registri dei dati e quelli degli indirizzi, ambedue a 32 bit, benché le operazioni su di essi siano effettuate con istruzioni identiche; ciò significa, in pratica, che spesso occorre spostare i dati da un registro degli indirizzi ad uno dei dati, elaborarli, e quindi riportare il risultato nel registro degli indirizzi. Sarebbe stato meno complicato se la Motorola avesse potuto evitare questo fastidioso inconveniente.

In secondo luogo, esistono inutili ripetizioni nell'insieme delle istruzioni, principalmente dovute a sovrapposizioni nel complicato sistema d'indirizzamento. Ciò, agli effetti pratici, non causa alcun problema. Significa soltanto che è possibile ottenere una stessa operazione partendo da presupposti diversi.

Il 68000 sembra destinato ad avere un grande successo e sono già molti i microcomputer ad averlo adottato. Ad esempio, oltre al QL, già citato, viene montato anche sul Lisa e sul MacIntosh della Apple.

I primi passi

Talvolta è possibile operare piccoli interventi sull'hardware, senza ricorrere a laboratori specializzati

In questa sezione esaminiamo come molti interventi, necessari per aggiungere periferiche ed accessori ad un microcomputer, possano essere fatti in proprio. Iniziamo dallo spiegare come vengono fatte le saldature elettriche.
La saldatura è un metodo per unire due oggetti metallici, usando una lega metallica facilmente malleabile (ossia con temperatura di fusione molto bassa). I due oggetti da saldare assieme vengono prima di tutto scaldati ad una temperatura abbastanza elevata (circa 280 gradi centigradi), sufficiente a fondere lo stagno. A questo punto, viene aggiunto lo stagno e, subito dopo, rimossa la fonte di calore. Raffreddandosi, i due oggetti metallici rimangono connessi.
Il saldatore, ossia la fonte di calore, non è altro che una punta metallica, generalmente riscaldata mediante una resistenza elettrica. Avvicinando il saldatore, ben caldo, allo stagno, questo si fonde. Uno dei pericoli maggiori, nelle saldature, è quello di non saper dosare adeguatamente il calore: una temperatura troppo alta rischia di danneggiare i componenti elettronici, mentre una troppo bassa può causare delle saldature "fredde", nelle quali la connessione, se non fisicamente, è elettricamente instabile.



La pinza strappafili

Per rimuovere la parte isolante dei cavi elettrici, una pinzetta strappafili può tornar molto comoda. Ne esistono di vari tipi, ma anche le più economiche sono dotate di un qualche dispositivo per regolarle sullo spessore del filo, producendo un lavoro pulito e ben fatto

Il saldatore

Questo attrezzo deve esser scelto con cura, perché ne esistono più tipi, a seconda dell'applicazione. I componenti elettronici possono venir danneggiati da un eccessivo calore, quindi è bene orientarsi su saldatori di bassa potenza (sui 15-16 Watt), possibilmente dotati di una punta fine, adatta alle dimensioni dei componenti. Anche lo stagno viene venduto in più misure, tra le quali occorre sceglierne una adeguata al lavoro che si intende fare (un diametro di mm. 1,5 può andar bene per la maggior parte dei casi)

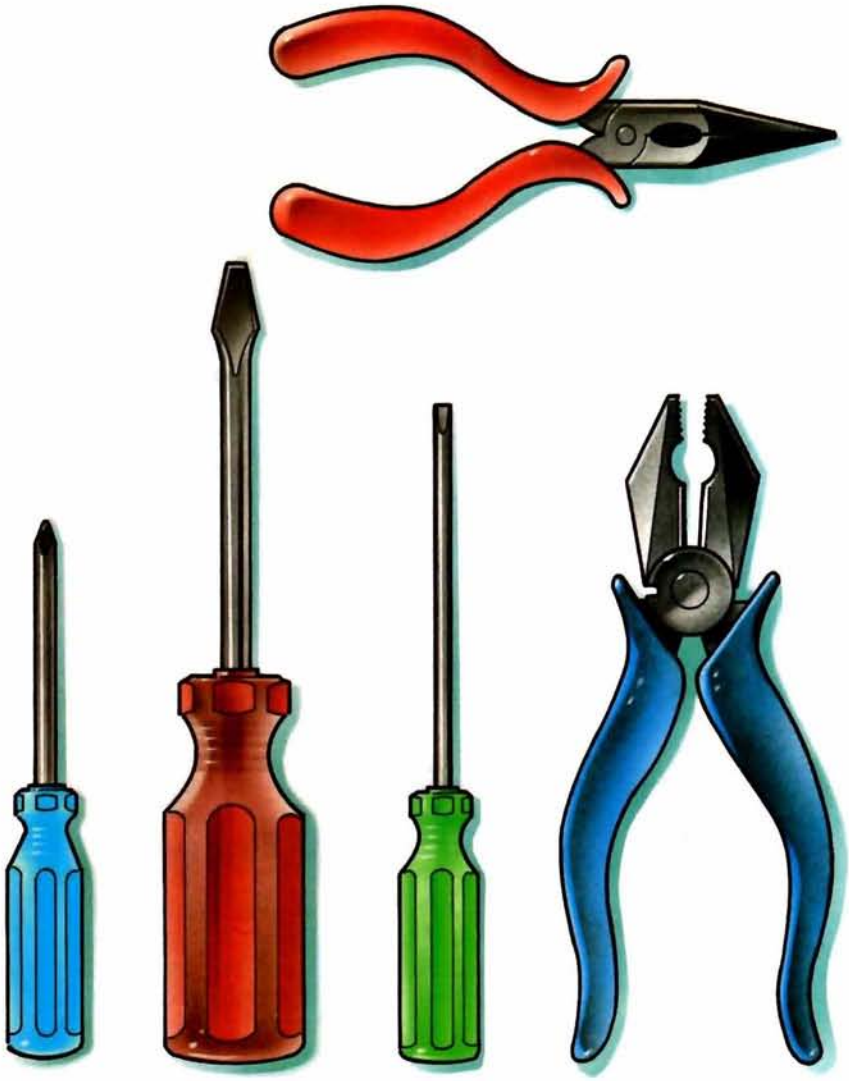


LA "TERZA MANO"

Le dimensioni dei componenti elettronici sono talvolta così piccole, da rendere difficoltoso qualsiasi intervento. Un utilissimo strumento da laboratorio è una pinza, montata su un solido piedistallo. In commercio se ne trovano di diversi tipi, adatti anche per piccoli lavori di modellismo



IAN McKINNELL



I cacciaviti

Di questo comunissimo attrezzo, come si sa, ne esistono due tipi, a seconda che la punta sia piatta o "a croce". Generalmente, chi si applica in piccole riparazioni domestiche ne possiede di più forme e dimensioni

Le pinze

Per i lavori di elettronica più semplici, bastano due tipi di pinze: quelle comuni, piuttosto tozze e robuste, e quelle più minute, con i becchi appuntiti

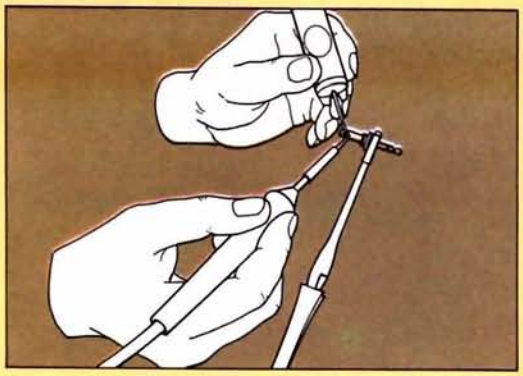


La pinza tagliafili

Questo attrezzo è utile per rimuovere tutte le eventuali sporgenze ed eccedenze di filo, a seguito di una saldatura. Ne esiste anche una versione con la lama disposta in senso trasversale rispetto al manico

MICHAEL BROWNLOW

LA SALDATURA DI UNO SPINOTTO



La sbucciatura e la stagnatura

La prima fase consiste nello sbucciare la pellicola isolante. Conviene abbondare in questa operazione e tagliare in seguito la parte di filo eccedente. Si appoggia il saldatore, ben caldo, sul filo elettrico applicando quindi lo stagno che fondendosi, riveste il filo di un fine strato. Attenzione a non eccedere con la quantità di stagno. Questa preparazione facilita le successive fasi. Dopo aver fissato saldamente lo spinotto su di un sostegno, si ripete l'operazione sul punto dove andrà collegato il filo elettrico. Una piccola parte di stagno, fondendosi, coprirà la superficie della saldatura. Adesso, mantenendo calda la zona della saldatura, si appoggia il filo stagnato. Si rimuove il saldatore facendo ben attenzione a non staccare le due parti



Rimozione degli eccessi

L'ultima operazione è quella di eliminare la parte di filo eccedente, per evitare che eventuali sporgenze possano andare in contatto con altre parti elettriche, causando un corto circuito (con tutte le conseguenze del caso!).
Per questa operazione conviene servirsi di apposite forbici o pinzette tagliafilo

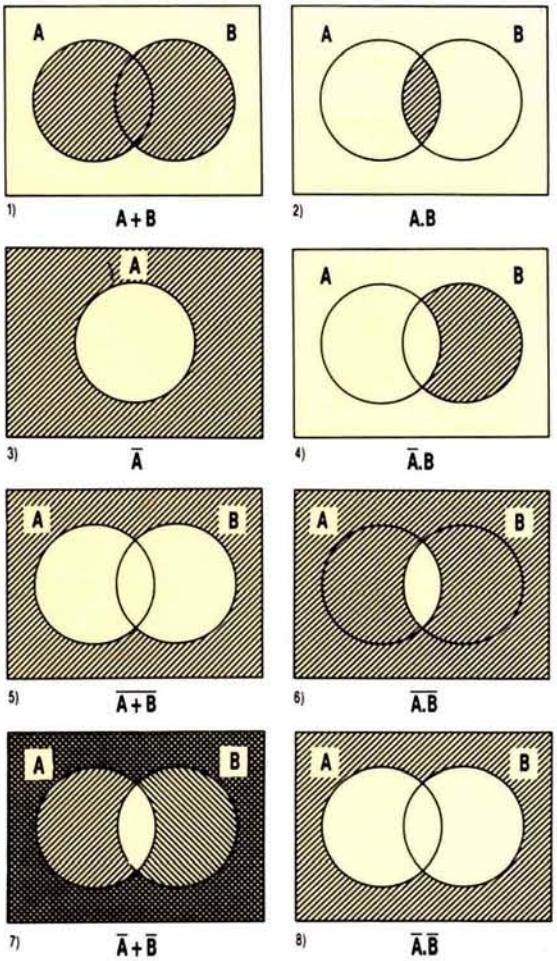


KEVIN JONES

Processi logici semplificati

Come ridurre le espressioni dell'algebra di Boole in forme meno complesse e più adatte al progetto di circuiti logici

I diagrammi di Venn costituiscono un valido metodo di rappresentazione grafica per semplificare le espressioni algebriche di Boole, consentendo di rappresentarle con semplici disegni. L'area all'interno di un rettangolo (che simboleggia 1 = Identità o Insieme Universale) rappresenta tutte le possibili combinazioni di valori degli ingressi ed i cerchi disegnati all'interno di esso rappresentano alcune delle combinazioni. Ecco alcuni diagrammi di Venn:



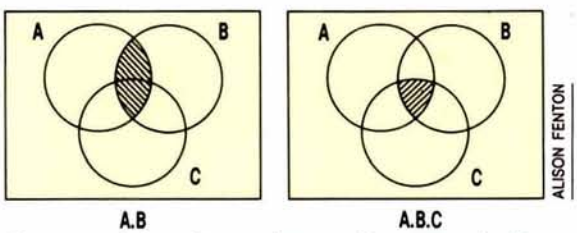
Un confronto tra i diagrammi 5 e 7 rivela immediatamente che NOT(A OR B) non equivale affatto a NOT(A) OR NOT (B). Analogamente, i diagrammi 6 ed 8 dimostrano che NOT(A AND B) non equivale a NOT(A) AND NOT(B).

Per meglio comprendere le operazioni AND ed OR nei diagrammi di Venn, si consideri A.B come l'area comune tra la superficie A e la superficie B, mentre A+B come la combinazione delle superfici A e B. Esistono alcune semplici regole nell'algebra di Boole: si provi a verificarle mediante i diagrammi di Venn (lo 0 qui rappresenta l'insieme "vuoto", ossia una combinazione impossibile):

1) $A.\overline{A} = A$
2) $A.\overline{A} = 0$
3) $A.0 = 0$
4) $A.1 = A$
5) $A.(\overline{A}+B) = A$
6) $A.(\overline{A}+B) = A.B$

Le leggi dell'algebra di Boole

Il concetto di "dualità" facilita molto la semplificazione delle espressioni e si basa sulla simmetria delle operazioni AND ed OR. Per formare il duale di una qualsiasi relazione, si cambiano tutte le AND in OR e viceversa. Al tempo stesso, si convertono tutti gli 0 in 1 e viceversa. Per esempio, si prenda la quinta relazione vista nel precedente elenco. Il duale di questa relazione è A + A.B = A. Anche questa relazione è vera ed introduce il concetto di "assorbimento duale". Osservando un diagramma di Venn, si vede chiaramente che la quantità A.B è completamente contenuta nell'area A e si può dunque dire che essa è assorbita da A. Ciò si può estendere anche al caso di tre variabili, ad esempio: A.B + A.B.C = A.B. Ecco i relativi diagrammi di Venn:



ALISON FENTON

Per esercizio, si provino a disegnare i diagrammi di Venn per le duali di tutte le cinque relazioni prima elencate.
Ritorniamo ora ai primi diagrammi. Da un confronto tra i diagrammi 5 ed 8 si può notare che la seguente relazione è sempre vera: $\overline{A+B} = \overline{A}.\overline{B}$. E così anche: $\overline{A.B} = \overline{A} + \overline{B}$.

Queste due relazioni, che passano sotto il nome di leggi di de Morgan, trovano applicazione anche nei casi di tre variabili ($\overline{A+B+C} = \overline{A}.\overline{B}.\overline{C}$ e $\overline{A.B.C} = \overline{A}+\overline{B}+\overline{C}$). Inoltre, queste leggi si possono applicare per gradi successivi:

$\overline{(A+B).C}$
$= \overline{A}.\overline{B}.\overline{C}$ (applicando le leggi di de Morgan al contenuto delle parentesi)
$= \overline{A+B+C}$ (ricombinando il risultato sempre in base a dette leggi)

In comune con la normale algebra, quella di Boole ha altre tre regole. Una è la legge "associativa", che permette lo spostamento delle parentesi:

$(A.B).C = A.(B.C) = A.B.C$
$(A+B)+C = A+(B+C) = A+B+C$

L'ordine col quale vengono scritte le lettere può esser variato grazie alla legge "commutativa":

$A.B = B.A$
$A+B = B+A$

Mentre la legge "distributiva" permette di sviluppare le parentesi:

$A.(B+C) = A.B+A.C$

Esempi di semplificazione

1. Semplificare $\overline{(A+\overline{B}+\overline{A}.B)}.B$
   $= (\overline{A}.\overline{\overline{B}}+\overline{\overline{A}}.B).B$ (de Morgan)
   $= \overline{A}.\overline{B}.B+\overline{A}.B.B$ (legge distributiva)
   $= 0+\overline{A}.B$ ($\overline{B}.B = 0, B.B = B$)
   $= \overline{A}.B$
2. Semplificare $\overline{A}.\overline{B}+\overline{A}.B+A.B$
   $= \overline{A}.(\overline{B}+B)+A.B$ (legge distributiva)
   $= \overline{A}+A.B$ ($\overline{B}+B = 1$)
   $= \overline{A}+B$ (duale relazione 6)
3. Semplificare $\overline{\overline{A}+B}+\overline{\overline{A}+\overline{B}}+\overline{A}.B$
   $= \overline{\overline{A}}.\overline{B}+\overline{\overline{A}}.\overline{\overline{B}}+\overline{A}.B$ (de Morgan)
   $= A.\overline{B}+A.B+\overline{A}.B$ ($\overline{\overline{A}} = A$)
   $= A.(\overline{B}+B)+\overline{A}.B$ (legge distributiva)
   $= A+\overline{A}.B$ ($\overline{B}+B = 1$)
   $= A+B$ (duale relazione 6)

Una porta XOR semplificata

Nella precedente lezione abbiamo visto l'esempio di una porta logica OR Esclusivo (XOR) non semplificata. Vediamo come ridurla efficacemente in base a quanto esposto finora. La tavola della verità di una porta XOR è:

INGRESSI		USCITA
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Dalla tavola della verità avevamo già dedotto che $C = \overline{A}.B+A.\overline{B}$. In questo caso si può semplificare ben poco, ma esaminiamo il problema da un altro punto di vista. Dalla tavola della verità si deduce che C varrà 1 se A e B non sono ambedue 0 o ambedue 1. Esprimendo ciò in termini di Boole:

$C = \overline{A.B+\overline{A}.\overline{B}}$

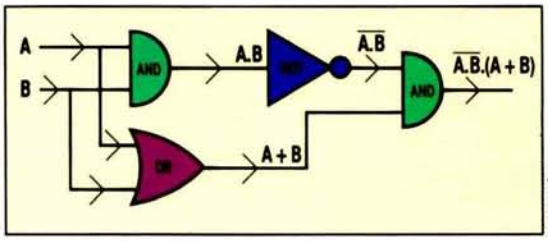
Adesso, applicando ripetutamente la legge di de Morgan, si può ottenere:

$C = \overline{(A.B)}.\overline{(\overline{A}.\overline{B})}$

ed, infine:

$C = \overline{A.B}.(A+B)$

Questa espressione richiede solo quattro porte:



LIZ DIXON

Un sommatore completo

Precedentemente, abbiamo visto il metodo per la somma binaria e progettato un semplice circuito logico adatto a questa funzione, chiamandolo "semisommatore". Indicando con X il primo ingresso e con Y il secondo, possiamo vedere dalla tavola della verità del semisommatore (vedere a pag. 513), che la somma (o risultato) S si può rappresentare con l'espressione: $S = \overline{X}.Y+X.\overline{Y}$, che, applicando la legge di De Morgan, si semplifica in: $S = \overline{X.Y}.(X+Y)$. Il Carry (= Riporto) non è che $C = X.Y$.
Ma noi sappiamo che in una somma binaria, in effetti, i bit da sommare sono tre su ciascuna colonna. Ossia, oltre ai due bit dei sommandi, c'è il Riporto dalla colonna precedente. Per ottenere un sommatore completo, quindi, dobbiamo prendere in considerazione tre ingressi e due uscite. Indicando con P il Riporto dalla colonna precedente, la tavola della verità per il sommatore completo diviene:

INGRESSI			USCITE	
P	X	Y	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Prendendo il caso di S = 1, si può dedurre l'espressione:

$S = \overline{P}.\overline{X}.Y + \overline{P}.X.\overline{Y} + P.\overline{X}.\overline{Y} + P.X.Y$

applicando le regole viste, si può semplificare in:

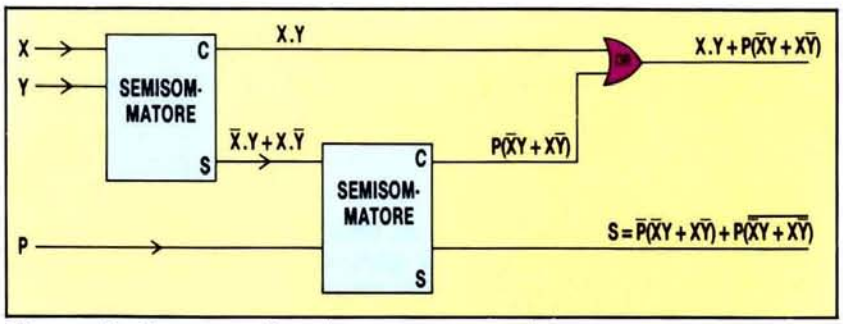
$S = \overline{P}.(\overline{X}.Y + X.\overline{Y}) + P.(\overline{X}.\overline{Y} + X.Y)$ (legge distributiva)

$S = \overline{P}.(\overline{X}.Y + X.\overline{Y}) + P.\overline{(X.Y + X.\overline{Y})}$ (de Morgan)

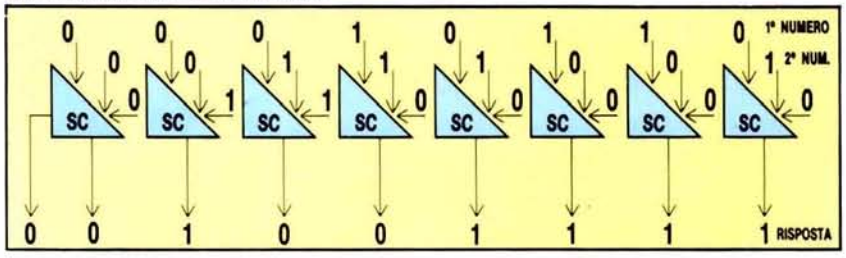
Analogamente, possiamo formulare un'espressione per C:

$C = \overline{P}.X.Y + P.\overline{X}.Y + P.X.\overline{Y} + P.X.Y$
$C = X.Y.(\overline{P} + P) + P.(\overline{X}.Y + X.\overline{Y})$ (legge distributiva)
$C = X.Y + P.(\overline{X}.Y + X.\overline{Y})$ ($\overline{P} + P = 1$)

Si noti che $\overline{X}.Y + X.\overline{Y}$ è il risultato della somma eseguita con un semisommatore, pertanto il circuito di un sommatore completo si può costruire con due semisommatori:



L'esempio che segue dimostra come una serie di otto sommatori completi, all'interno di una ALU, possa servire per la somma di due numeri binari ad otto bit.



ESERCIZIO 3

1) Si semplifichino queste espressioni:
   a) $A.(\overline{A} + \overline{B})$
   b) $X + Y.(X + Y) + X.(\overline{X} + Y)$
   c) $P.Q + \overline{P}.Q + \overline{P}.\overline{Q}$
   d) $\overline{X} + \overline{Y.Z} + \overline{Z}.Y$

2) Un antifurto per auto è dotato di un interruttore generale e di un interruttore su ciascuna delle due portiere. L'allarme suona se una delle due porte viene aperta oppure azionando l'interruttore generale. Si costruisca una tavola della verità con i tre ingressi (Portiera A, Portiera B, Interruttore Generale) e l'uscita (allarme ON oppure OFF). Si usi poi la tavola per dedurre l'espressione Booleana equivalente e si progetti il circuito corrispondente.

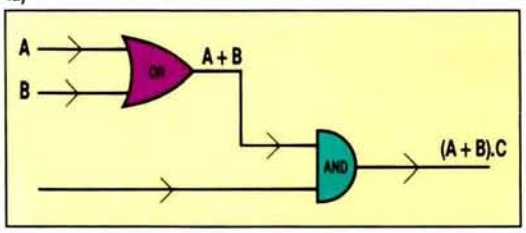
3) La lampada di corridoio può essere accesa mediante tre interruttori posti in tre luoghi diversi. Disegnare un circuito logico adatto.

4) Ci troviamo perduti nel deserto, assieme a due altre persone: una delle due dice sempre la verità, l'altra racconta sempre bugie. Per sopravvivere è necessario stabilire chi dica il vero. Esistono molte domande che potremmo porre alle due persone a questo fine. Si costruisca una tavola della verità per esaminare le possibili risposte. Ecco una traccia iniziale:
«Tu dici sempre la verità?»

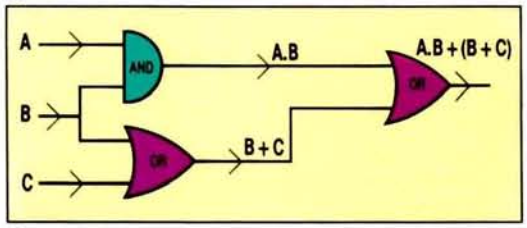
		RISPOSTE POSSIBILI	
		SI	NO
IDENTIFICAZIONE DELLA PERSONA	BUGIARDA	1	0
	SINCERA	1	0

Risposte all'Esercizio 2, pag. 513

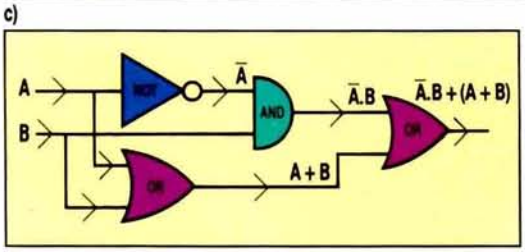
1a)



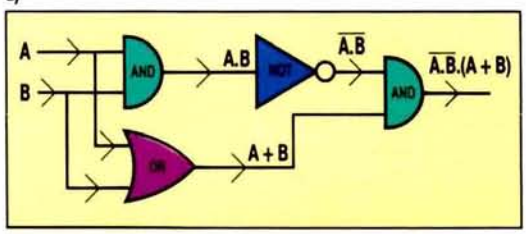
b)



c)



d)



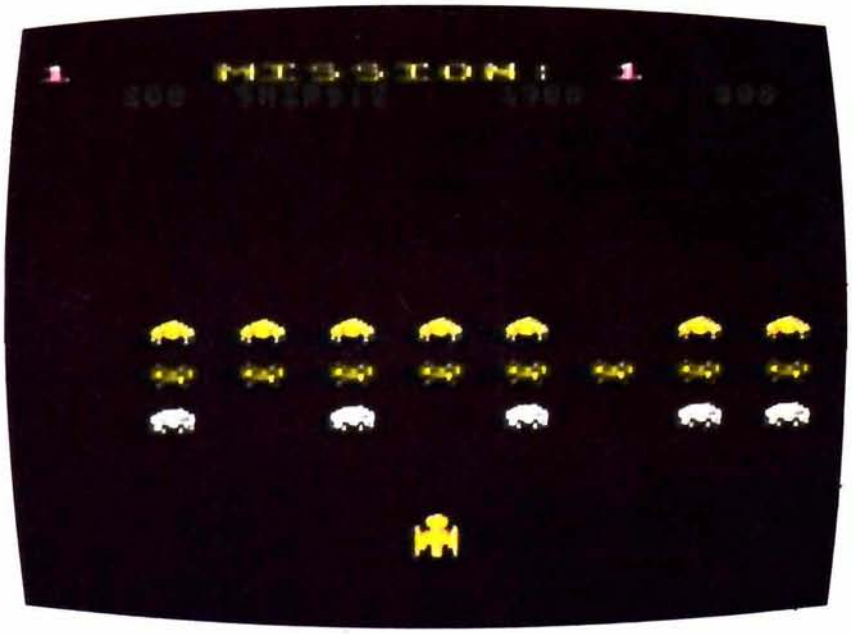
2a) $C = \underline{A.B}$
b) $S = \overline{A.B}(A + B)$

3a) $X = \underline{(A + B)}.(B.C + C)$
b) $X = \overline{A.B} + \overline{B}$

Software in vetrina

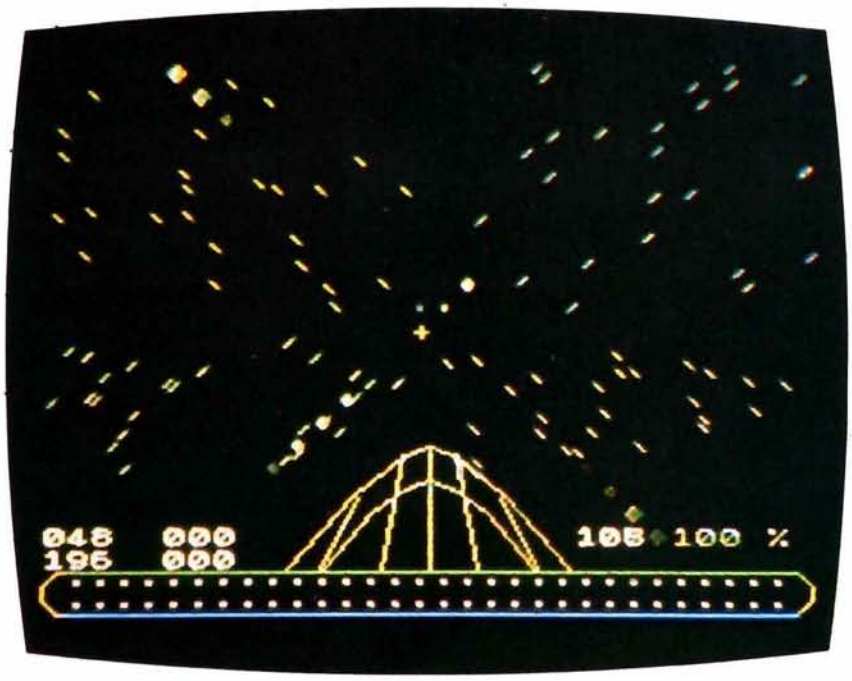
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Continua la panoramica su alcuni dei giochi reperibili in commercio, adatti alle marche di home computer maggiormente diffuse. È questa la volta di due divertenti programmi, appartenenti ambedue al genere spaziale, ma di concezione totalmente diversa. Il primo gioco è per l'Atari, il secondo per lo Spectrum.



Gorf per l'Atari

Il gioco fa parte del vasto genere *Space Invaders*, ma con diverse varianti. È suddiviso in quattro fasi distinte. Nella prima, il giocatore deve difendere il proprio pianeta, parzialmente protetto da un campo di forza. I robot attaccanti lanciano bombe antigravitazionali, ma, evitate queste ed eliminati tutti i robot, si procede alla fase successiva. Il compito, qui, è di distruggere due astronavi nemiche e di parare gli attacchi delle navette *kamikaze* e dei *Gorf*. Nella terza fase, il campo di battaglia è molto simile ad una tela di ragno, generata dalla nave spaziale nemica, la Flagship, in avvicinamento. Nell'ultima fase occorre distruggere la Flagship.



Il Black Planet per lo Spectrum

Siamo nel XXI Secolo ed il giocatore è *Starmagon*, il capo di una pattuglia spaziale. La Terra ha esteso i propri domini nelle più lontane galassie, ma i pirati, la cui base è il "Pianeta Nero" (Black Planet), un mondo senza sole, imperversano sugli innocenti mercanti spaziali. Per trovare il pianeta nero occorre individuare una chiave, le cui componenti sono sparpagliate su sette diversi pianeti, che il giocatore deve, pertanto, visitare. Il gioco si sviluppa in un viaggio "grafico" attraverso questi mondi fantastici, ciascuno dei quali ha i propri pericoli.

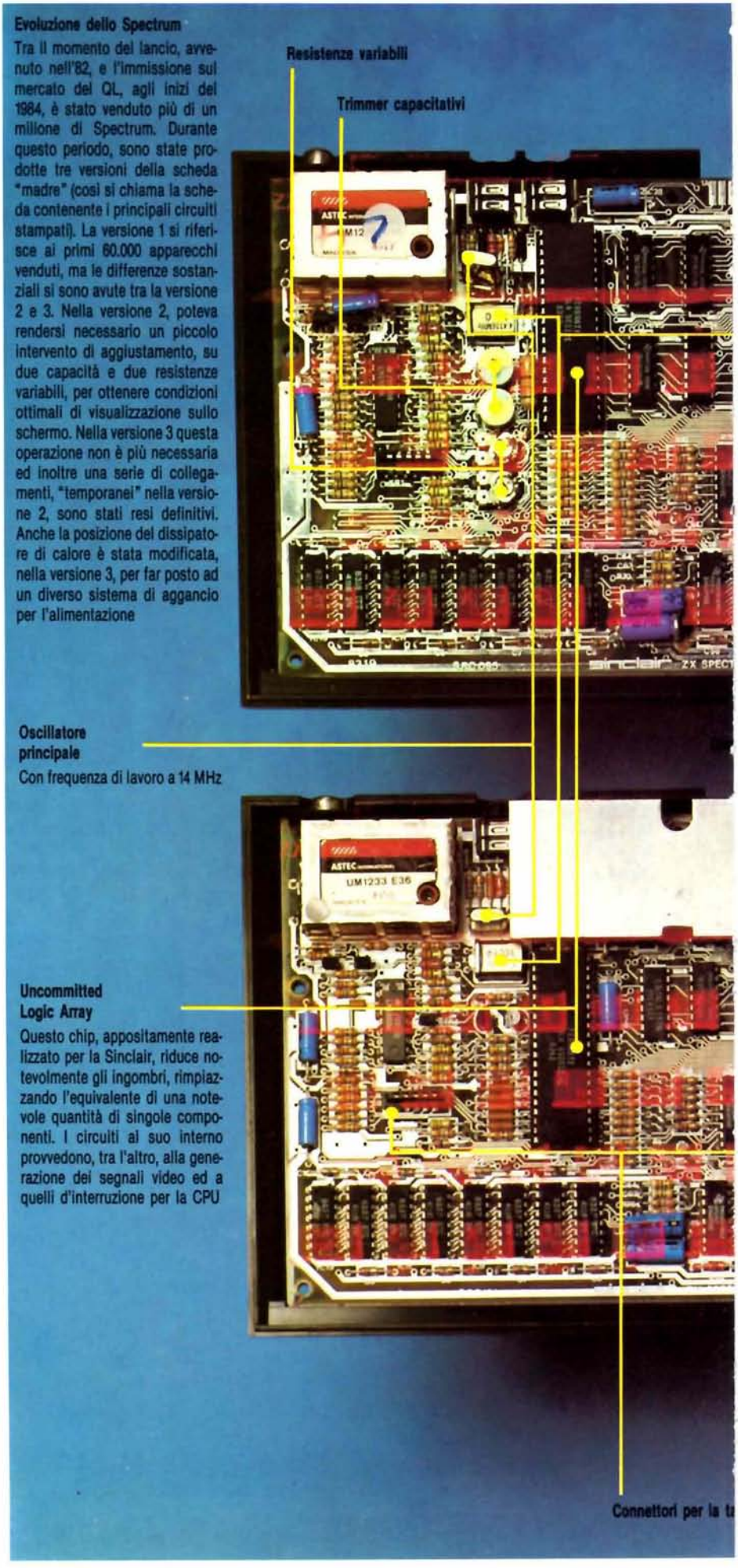
Spectrum terza versione

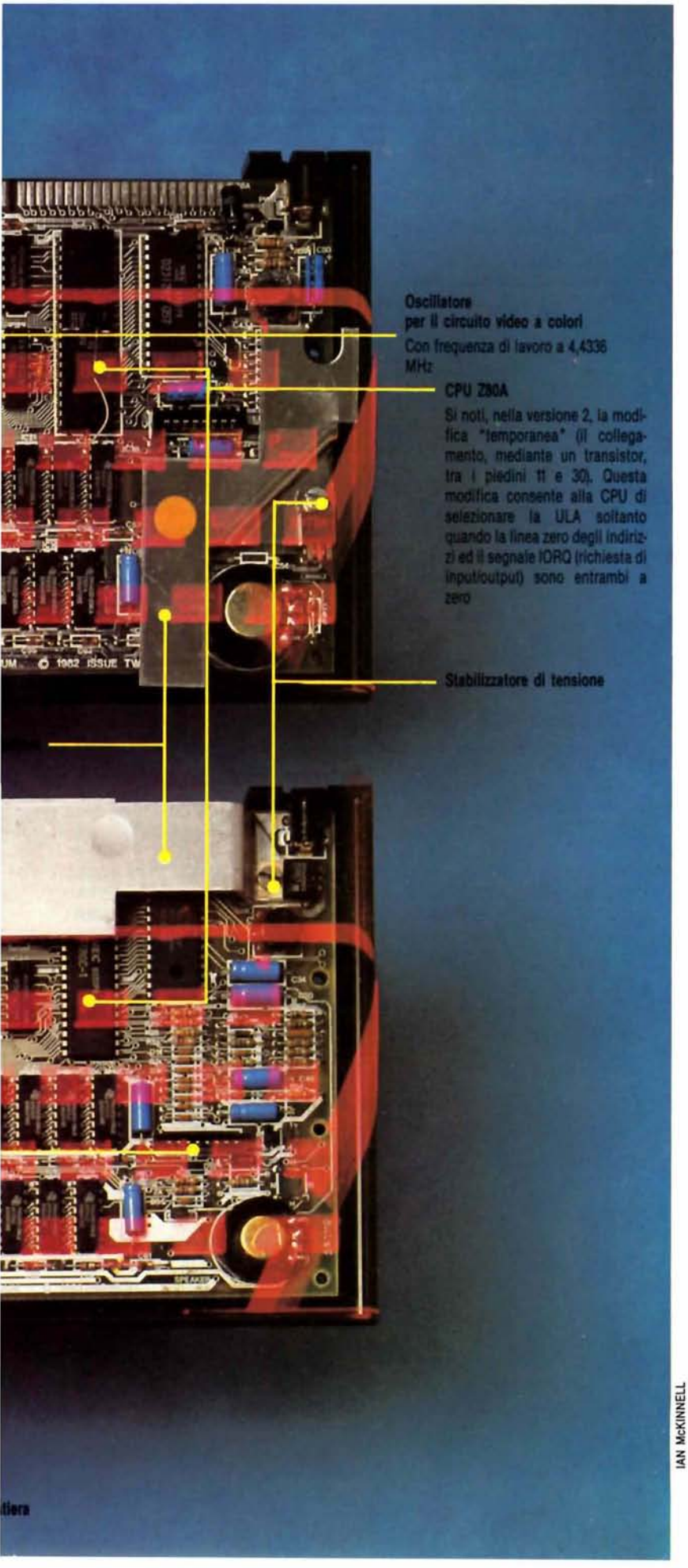
Alcuni principali inconvenienti sono stati eliminati, ma...

Pochi produttori di computer possono vantarsi di aver venduto oltre un milione di unità nel breve arco di 15 mesi: la Sinclair ha ottenuto questo strabiliante risultato con lo Spectrum, un home computer con numerosi pregi (non ultimo il costo molto contenuto), ma anche con qualche inconveniente.
Uno tra questi è dovuto alle caratteristiche della tastiera, che, nonostante la disposizione QWERTY dei tasti (comune alla maggior parte dei computer), non è molto comoda: i tasti sono "gommosi" al tatto e non certo adatti a lunghe sessioni di lavoro.
Il "bus" del sistema consente il collegamento, anche contemporaneo, alla stampante ZX (originariamente progettata per lo ZX81) ed alle interfacce ZX 1 e ZX 2. L'apparecchio dispone anche di prese MIC e EAR, che permettono di archiviare programmi e dati su nastri a cassette. Il sistema basato sul registratore, tuttavia, non è sempre molto affidabile. Benché il successo delle operazioni di scrittura e di lettura sia segnalato sullo schermo da delle strisce blu e gialle, per evitare la fastidiosa ripetizione di più tentativi di scrittura occorre disconnettere la presa EAR. Altrettanto fastidiosa è la mancanza di un tasto di RESET: ciò significa dover sfilare e rinfilare la presa d'alimentazione, ogni volta che sia necessario riazzerare il computer, a seguito di un blocco nel programma. Fortunatamente, esistono in commercio delle piccole unità accessorie, che incorporano sia dei selettori per le operazioni di archiviazione che un tasto di RESET.
La politica aziendale della Sinclair sembra contare molto sull'iniziativa di molte ditte, da essa indipendenti, che producono una vastissima gamma di accessori.
Tuttavia, per lo Spectrum, numerose periferiche sono state progettate "in casa": i Microdrive e le due interfacce ZX 1 e ZX 2, ad esempio, che consentono di disporre di una considerevole memoria di massa (circa 680 Kbyte in tutto). L'interfaccia seriale RS232, inoltre, permette di creare una rete locale con un massimo di 64 Spectrum.
Per quanto concerne il software, la Sinclair ha totalmente lasciato ad altre ditte lo sviluppo e la commercializzazione di programmi, limitandosi ad apporre un "sigillo di approvazione" su determinati programmi.

Evoluzione dello Spectrum

Tra il momento del lancio, avvenuto nell'82, e l'immissione sul mercato del QL, agli inizi del 1984, è stato venduto più di un milione di Spectrum. Durante questo periodo, sono state prodotte tre versioni della scheda "madre" (così si chiama la scheda contenente i principali circuiti stampati). La versione 1 si riferisce ai primi 60.000 apparecchi venduti, ma le differenze sostanziali si sono avute tra la versione 2 e 3. Nella versione 2, poteva rendersi necessario un piccolo intervento di aggiustamento, su due capacità e due resistenze variabili, per ottenere condizioni ottimali di visualizzazione sullo schermo. Nella versione 3 questa operazione non è più necessaria ed inoltre una serie di collegamenti, "temporanei" nella versione 2, sono stati resi definitivi. Anche la posizione del dissipatore di calore è stata modificata, nella versione 3, per far posto ad un diverso sistema di aggancio per l'alimentazione





IAN McKINNELL

SPECTRUM SINCLAIR

DIMENSIONI

mm. 233 x 144 x 30

CPU

Z80A

CLOCK

3,5 MHz

MEMORIA

16 Kbyte di RAM (estendibile a 48 Kbyte). 16 Kbyte di ROM contengono il BASIC

VIDEO

Lo schermo è suddiviso in 24 linee di 32 caratteri. La grafica ad alta risoluzione permette l'uso di 192 x 256 punti. 16 caratteri grafici sono predefiniti, mentre l'utente può definirne fino a 21 totalmente nuovi. Sono disponibili 8 colori, il lampeggio e due livelli d'intensità

INTERFACCE

Connettore per TV, per registratore a cassette. Porta di connessione con il bus del sistema

LINGUAGGI FORNITI

BASIC ed Assembly Z80

ALTRI LINGUAGGI DISPONIBILI

FORTH, PASCAL, LISP, LOGO, PROLOG

TASTIERA

40 tasti "a membrana". Mediante l'uso di speciali tasti di SHIFT, ogni tasto può svolgere fino ad 8 funzioni diverse

DOCUMENTAZIONE

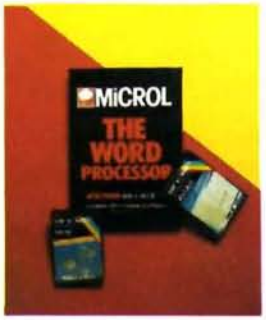
Il computer è dotato di un manuale introduttivo e di un manuale operativo, contenente anche una guida al BASIC della Sinclair

PRO

Il bassissimo costo dell'apparecchio lo rende ideale per i principianti. La quantità di software in commercio è veramente notevole sia nel settore dei giochi che in quello delle applicazioni più serie

CONTRO

Il maggior inconveniente, più volte menzionato, è la tastiera: un compromesso tra qualità e costo. L'uso dello schermo non segue i metodi adottati per la maggior parte degli home computer, suscitando, agli inizi, qualche perplessità



La biblioteca dello Spectrum

Oltre alla vastissima gamma di programmi ricreativi, esiste una considerevole scelta di "pacchetti" applicativi anche per usi sia domestici che commerciali, molti dei quali ad un prezzo ragionevolissimo. Disponendo dei Microdrive, infatti, lo Spectrum può essere usato, sia pure con certe limitazioni, anche per applicazioni "serie". La Psion, una delle case di software più esperte negli apparecchi Sinclair, ha prodotto molti programmi applicativi, per la maggior parte contraddistinti dalla sigla VU: VU-CALC (una lavagna elettronica), VU-FILE (un programma di gestione archivi) e VU-3D (per la gestione grafica). Inoltre, nonostante la difficoltà d'impiego della tastiera, è stato prodotto anche un word processor relativamente sofisticato, che consente di memorizzare fino a dieci pagine di testo in formato A4

Spectrum terza versione

Alcuni principali inconvenienti sono stati eliminati, ma...

Pochi produttori di computer possono vantarsi di aver venduto oltre un milione di unità nel breve arco di 15 mesi: la Sinclair ha ottenuto questo strabiliante risultato con lo Spectrum, un home computer con numerosi pregi (non ultimo il costo molto contenuto), ma anche con qualche inconveniente.

Uno tra questi è dovuto alle caratteristiche della tastiera, che, nonostante la disposizione QWERTY dei tasti (comune alla maggior parte dei computer), non è molto comoda: i tasti sono "gommosi" al tatto e non certo adatti a lunghe sessioni di lavoro.

Il "bus" del sistema consente il collegamento, anche contemporaneo, alla stampante ZX (originariamente progettata per lo ZX81) ed alle interfacce ZX 1 e ZX 2. L'apparecchio dispone anche di prese MIC e EAR, che permettono di archiviare programmi e dati su nastri a cassette. Il sistema basato sul registratore, tuttavia, non è sempre molto affidabile. Benché il successo delle operazioni di scrittura e di lettura sia segnalato sullo schermo da delle strisce blu e gialle, per evitare la fastidiosa ripetizione di più tentativi di scrittura occorre disconnettere la presa EAR. Altrettanto fastidiosa è la mancanza di un tasto di RESET: ciò significa dover sfilare e rinfilare la presa d'alimentazione, ogni volta che sia necessario riazzerare il computer, a seguito di un blocco nel programma. Fortunatamente, esistono in commercio delle piccole unità accessorie, che incorporano sia dei selettori per le operazioni di archiviazione che un tasto di RESET.

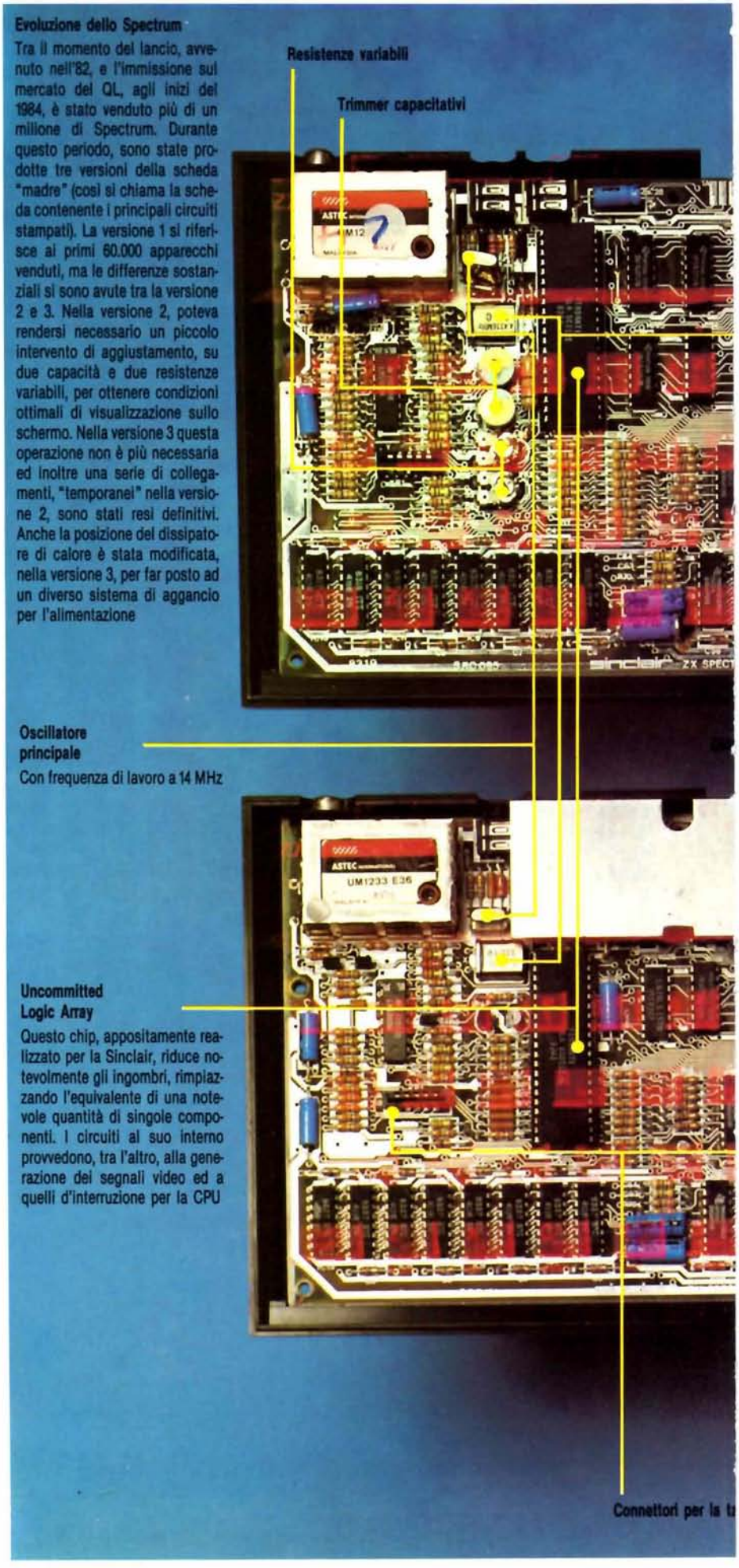
La politica aziendale della Sinclair sembra contare molto sull'iniziativa di molte ditte, da essa indipendenti, che producono una vastissima gamma di accessori.

Tuttavia, per lo Spectrum, numerose periferiche sono state progettate "in casa": i Microdrive e le due interfacce ZX 1 e ZX 2, ad esempio, che consentono di disporre di una considerevole memoria di massa (circa 680 Kbyte in tutto). L'interfaccia seriale RS232, inoltre, permette di creare una rete locale con un massimo di 64 Spectrum.

Per quanto concerne il software, la Sinclair ha totalmente lasciato ad altre ditte lo sviluppo e la commercializzazione di programmi, limitandosi ad apporre un "sigillo di approvazione" su determinati programmi.

Evoluzione dello Spectrum

Tra il momento del lancio, avvenuto nell'82, e l'immissione sul mercato del QL, agli inizi del 1984, è stato venduto più di un milione di Spectrum. Durante questo periodo, sono state prodotte tre versioni della scheda "madre" (così si chiama la scheda contenente i principali circuiti stampati). La versione 1 si riferisce ai primi 60.000 apparecchi venduti, ma le differenze sostanziali si sono avute tra la versione 2 e 3. Nella versione 2, poteva rendersi necessario un piccolo intervento di aggiustamento, su due capacità e due resistenze variabili, per ottenere condizioni ottimali di visualizzazione sullo schermo. Nella versione 3 questa operazione non è più necessaria ed inoltre una serie di collegamenti, "temporanei" nella versione 2, sono stati resi definitivi. Anche la posizione del dissipatore di calore è stata modificata, nella versione 3, per far posto ad un diverso sistema di aggancio per l'alimentazione



Unità a dischi Commodore

La Commodore adotta il criterio delle unità a dischetti "intelligenti", dotate del proprio sistema operativo

Il maggior problema, nella produzione di unità a dischi "intelligenti", è dovuto ai notevoli costi che comporta. Poco dopo il lancio del notissimo Vic-20, la Commodore ha realizzato un'unità a dischetti, abbastanza economica, derivata da quelle impiegate per il PET: la Vic-1540. L'uso di queste unità, con il Commodore 64, presenta poche difficoltà salvo la necessità di eseguire due POKE, una prima dell'accesso ai dischetti, una dopo. Tuttavia, grazie ad una nuova versione del sistema operativo per dischi (DOS), che ha portato alla realizzazione dell'unità Vic-1541, questo fastidioso problema è risolto. La 1541 è totalmente compatibile sia col Vic-20 che con il Commodore 64.

Il 1541 è controllato mediante un microprocessore 6502, due chip d'interfaccia VIA (Versatile Interface Adapters) 6522, 2 Kbyte di RAM ed 8 Kbyte di ROM, che contengono il DOS. Quest'ultimo possiede notevoli doti, consentendo di programmare operazioni piuttosto complesse per quanto concerne l'archiviazione di programmi (archivi PRG), di file sequenziali (archivi SEQ) e di file ad accesso casuale. Il tutto corredato da apposite procedure di controllo degli errori. Il collegamento con il computer avviene mediante una versione seriale dell'interfaccia IEEE488, simile alla versione parallela (usata per il controllo delle altre periferiche dei Commodore), ma dall'impiego molto più flessibile. Quest'interfaccia permette di collegare "in cascata" più periferiche. Ad esempio, l'unità a dischi può "scaricare" un file su stampante, mentre il computer si occupa, contemporaneamente, di un altro programma. Ciò è possibile associando dei numeri identificativi sia agli archivi sia ai dispositivi esterni del computer.

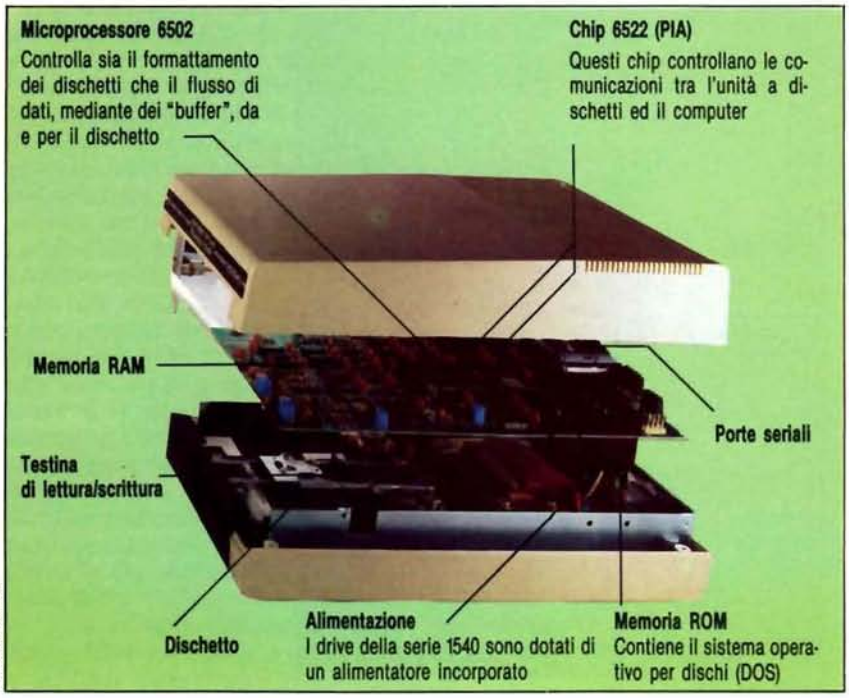
I dischetti sono "formattati" in singola faccia, con 35 tracce concentriche. Quelle più esterne contengono 21 settori, quelle più interne soltanto 17. A sua volta, ogni settore contiene un blocco di 256 byte di dati, oltre ad informazioni di gestione interna. Un dischetto contiene, dunque, 683 blocchi (664 dei quali accessibili all'utente), con una capacità totale di circa 170 Kbyte. Il DOS gestisce la collocazione dei dati sui dischetti utilizzando una Block Availability Map (Mappa di Disponibilità dei Blocchi) o BAM ed una Directory (elenco degli archivi). La BAM è memorizzata sulla traccia 18, settore 0, e consiste di 144 byte, ciascuno dei quali indica quali blocchi siano liberi e quali già occupati. La Directory, invece, è conservata nel settore 1 della stessa traccia e può contenere fino ad un massimo di 144 nomi di archivi, assieme ad informazioni sula loro lunghezza in blocchi. Sia la BAM che la Directory vengono aggiornate ogni volta che si esegue una scrittura o una cancellazione di dati sul dischetto.

Il prezzo abbastanza elevato di queste unità a dischetti è ampiamente ripagato dalla flessibilità che l'intero sistema acquista grazie al loro impiego. Infatti, pur offrendo una discreta quantità di memoria di massa, non viene intaccata la quantità di RAM disponibile al computer, dato che il sistema operativo è completamente esterno ad esso. Unico svantaggio è la velocità di trasferimento, che pur essendo circa 50 volte superiore a quella dei registratori a nastri, non è paragonabile a quella ottenuta con apparecchi di marca diversa. I manuali, comunque, non forniscono esatte indicazioni su questo parametro. Desiderando ampliare la memoria di massa, si può acquistare una speciale interfaccia che consente di collegare le periferiche usate in passato per il PET.

Il disk drive Commodore

L'unità 1541, collegabile sia al Vic-20, sia al Commodore 64, è incorporata nel più recente SX-64.

Si tratta di un'unità a dischetti "intelligente", dotata di un proprio microprocessore e di un proprio sistema operativo, in modo da non appesantire in alcun modo il funzionamento del computer. Il microprocessore è identico a quello dell'unità centrale: un 6502 della MOS Technology



IAN McKINNELL

I comandi del DOS Commodore

Il sistema operativo per dischi (DOS) della Commodore è dotato di numerosi comandi, che permettono al programmatore di sfruttare sia le normali forme di archiviazione sequenziali, sia il comodissimo metodo di accesso "casuale". I comandi vengono riportati qui di seguito. Il numero 8 identifica, negli esempi, il canale associato all'unità a dischi.

SAVE
Memorizza su disco un programma o una serie sequenziale di dati, assegnandogli il nome specificato (fino a 16 caratteri) e l'estensione PRG.
Il formato è:

SAVE "NOMEFILE",8

LOAD
Ha il formato:

LOAD "NOMEFILE",8

e serve per copiare il file specificato dal dischetto alla memoria RAM del sistema. Il comando

LOAD "NOMEFILE",8,1

ricopia il contenuto del file esattamente nell'identica posizione di memoria dalla quale era stato originariamente estratto.

LOAD "$",8

copia la Directory del disco in memoria, permettendo di visualizzarla, con un comando LIST, come se si trattasse di un programma BASIC.
L'ordine delle informazioni è il seguente:

Nome del dischetto
2 caratteri identificativi del dischetto
Fino a 144 nomi di file
Tipo (PRG e SEQ) di ciascun file
Lunghezza, in blocchi, di ciascun file
Numero di blocchi ancora liberi

VERIFY
Il formato è:

VERIFY "NOMEFILE",8

e serve per controllare, mediante un confronto, se un file archiviato su dischetto corrisponde esattamente a quello in memoria, avvisando di eventuali errori. Viene usato per verificare la corretta memorizzazione dei file.

OPEN
Stabilisce un canale di comunicazione, identificato da un "numero logico di file" (LFN) compreso tra 1 e 255. Si possono "aprire", contemporaneamente, fino a 10 canali di comunicazione. La OPEN, inoltre, stabilisce un "indirizzo secondario" (SA), che indica il modo di funzionamento del dispositivo al quale si accede. L'unico valore accettato, per i dischetti, è il 15, che accede ad un canale "privilegiato", riservato ai comandi.
Il formato è:

OPEN LFN,8,SA

CLOSE
Segue il formato

CLOSE LFN

e serve per terminare l'accesso al canale di comunicazione identificato da LFN. I file logici dovrebbero sempre essere "chiusi" (con una CLOSE), quando non sono più necessari.

PRINT#, INPUT# e GET#
La PRINT# funziona esattamente come la normale PRINT, salvo il fatto che i dati vengono emessi, sequenzialmente, sul file logico determinato dall'istruzione OPEN associata, anziché sullo schermo.
Il formato è:

PRINT#LFN,"DATI" oppure
PRINT#LFN,A$,B$...

Analogamente, INPUT# e GET# leggono file sequenziali (del tipo SEQ). INPUT# deve essere usata soltanto se i dati sono separati fra loro con dei punti e virgola (altrimenti non viene operata alcuna distinzione e la serie di dati viene trattata come un'unica stringa). GET#, viceversa, legge un solo byte alla volta, indipendentemente dal suo contenuto. Questa funzione è molto comoda se non si conosce, a priori, il formato dei dati conservati su disco. Ecco alcuni esempi sul formato:

INPUT#LFN,A$,B$,...
GET#LFN,A$,B$,...

Quando la PRINT# viene usata congiuntamente ad un file logico "aperto" con una OPEN sul canale dei comandi (ad esempio: OPEN LFN,8,15), allora essa si trasforma in un potente mezzo di comunicazione con il sistema operativo dei dischetti.
Il formato è:

PRINT#LFN,8,15,"stringa del comando"

Questa particolare funzione serve anche per utilizzare i dischetti usando file con accesso casuale (del tipo REL).

Comandi di sistema

Ecco un elenco di comandi che si possono impartire mediante le PRINT# ed OPEN operate sul canale "privilegiato" dei comandi:

NEW
Formatta ed assegna il nome ad un dischetto
Costruisce la relativa BAM ed una Directory vuota
Assegna 2 caratteri identificativi al dischetto (DI)
Formato: N:NOMEDISCO,DI

INITIALIZE
Confronta la BAM su disco con quella in RAM
Formato: I

VALIDATE
Riorganizza la BAM, riscrivendola su disco
Formato: V

RENAME
Cambia il nome di un file nella Directory
Formato: R:NUOVONOME=VECCHIONOME

SCRATCH
Cancella i file specificati dal disco e dalla Directory
Formato: S:NOMEFILE1, NOMEFILE2,...

COPY
Copia il contenuto di un file in un altro, sullo stesso dischetto
Formato: C:NOMECOPIA + NOMEORIG
Unisce più file sequenziali (SEQ) in uno solo file (concatenazione)
Formato: C:NOMEFILE=NOME1, NOME2, ...

Verifica degli errori
Sul pannello frontale dell'unità a dischetti 1451 sono presenti due lampadine "spia" a LED: una verde (per l'alimentazione) ed una rossa, che ha le seguenti funzioni:

accesa = è in corso un'operazione di lettura o scrittura

spenta = nessuna operazione in corso

lampeggiante = il DOS ha rilevato un errore

Per stabilire la fonte di un errore, occorre leggere lo speciale "canale d'errore" del DOS. Quello che segue è un semplice programma adatto a questo scopo. Il significato di ciascun codice d'errore è riportato nel manuale del computer:

```
10 REM **VERIFICA ERRORE
   SU DISCO**
20 OPEN 15,8,15
30 INPUT#15,EN,EM$,ET,ES
40 PRINTCHR$(147)
50 PRINT "ERRORE N.";EN
60 PRINT EM$
70 PRINT "TRACCIA:";ET
80 PRINT "SETTORE:";ES
90 CLOSE 15:END
```

L'ABC del... BBC

Le attrattive del dialetto del BASIC adottato per il Micro BBC

Il BASIC viene frequentemente criticato per essere un linguaggio non strutturato, che incoraggia (o non impedisce) l'instaurarsi di cattive abitudini di programmazione nei principianti. In particolar modo, il BASIC consente la stesura di programmi funzionanti, ma magari "rappezzati" alla meno peggio e costellati da un'infinità di GOTO.
L'impiego della struttura IF...THEN...ELSE permette, entro un certo grado, l'eliminazione dei più comuni trabocchetti dovuti ad un uso indiscriminato di istruzioni GOTO. Le poche linee di programma che seguono, ad esempio:

```
1500 IF VALORE > 0 THEN GOTO 1800
1600 PRINT "VALORE NON ACCETTABILE"
1700 GOTO 1900
1800 PRINT "VALORE ACCETTABILE"
1900 NEXT L
```

si possono più convenientemente esprimere con:

```
1500 IF VALORE > 0 THEN PRINT "VALORE ACCETTABILE"
     ELSE PRINT "VALORE NON ACCETTABILE"
1600 NEXT L
```

dove, grazie all'uso di ELSE, si eliminano completamente le GOTO.
L'istruzione GOSUB, normalmente, ha per argomento un valore numerico (un numero di linea), causando due inconvenienti. Il primo è che una frase del tipo GOSUB 1000 niente ci dice sullo scopo della subroutine situata alla linea 1000 e seguenti. Il secondo inconveniente è che, salvo casi speciali, una modifica alla numerazione delle linee di programma o una fusione tra due programmi diviene alquanto complicata.
La GOSUB e la GOTO sono istruzioni molto lente, in quanto il BASIC deve compiere una ricerca lineare della linea di programma alla quale esse si riferiscono, ogni volta che le incontra.
Il BASIC del Micro BBC offre una risposta a questi due problemi: con esso è possibile definire funzioni e procedure che eliminano quasi del tutto i due inconvenienti appena visti. Ambedue le strutture sono simili a delle subroutine, ma, anziché col numero di linea, esse vengono indicate mediante dei nomi. Ciò rende enormemente più chiaro il listato di un programma ed elimina i problemi di numerazione o ri-numerazione del programma. Inoltre, le funzioni e le procedure vengono eseguite con maggiore rapidità che non le GOTO e le GOSUB.
Le procedure e le funzioni iniziano, rispettivamente, con DEF PROC e DEF FN, cui segue un nome e (normalmente, ma non necessariamente) un elenco di parametri. Per esempio:

```
1200 DEF FNcalc(a,b,c) = (a—b)*c/100 e
2500 DEF PROCoperaz(w,x$,y$,z)
```

Nella definizione, per i parametri vengono usati dei nomi di variabili, ma quando, nel programma, vengono fatte delle chiamate alla funzione o alla procedura, ai parametri vengono sostituiti dei valori numerici (o alfanumerici, a seconda del caso) del medesimo tipo indicato. Per esempio:

```
250 risultato = FNcalc(prezzo,costo,12) oppure
545 PROCoperaz (6,nome$,"Bianchi",valore(12))
```

I valori dei parametri, "passati" alla funzione o alla procedura, vengono sostituiti alle variabili indicate nella definizione, producendo gli appropriati risultati. Si noti che una funzione può essere impiegata, all'interno di un'espressione, come se si trattasse di una normale variabile o quantità numerica, mentre una procedura viene adoperata come se si trattasse di un nuovo comando BASIC.
L'uso dell'istruzione LOCAL, all'interno di una definizione, permette di circoscrivere l'uso di una variabile, evitando un comune errore di programmazione. Si consideri, ad esempio:

```
100 FOR K = 1 to 10: GOSUB 500:NEXT K:END
500 FOR K = 1 to 5: PRINT "*****":NEXT:RETURN
```

La variabile K viene usata come contatore nel ciclo principale (linea 100) ed, allo stesso tempo, nella subroutine (linea 500). Questo genere di errori, dai risultati imprevedibili, è molto difficile da individuare, particolarmente nei programmi di una certa lunghezza. Nel BASIC del BBC, comunque, questo pericolo si può evitare:

```
100 FOR K=1 TO 10:PROCstelle:NEXT:END
500 DEF PROCstelle
520 LOCAL K
540 FOR K = 1 TO 5:PRINT "*****":NEXT
560 ENDPROC
```

L'istruzione LOCAL (linea 520) indica che K è una nuova variabile, indipendente da quella usata in altre parti del programma ed il cui valore non influisce su quest'ultima. (Da notare che, nell'esempio, PROCstelle non possiede parametri).
Anche la struttura REPEAT...UNTIL, insolita per il BASIC, è assai comoda: essa permette di costruire dei cicli, per la ripetizione di un gruppo di istruzioni, che si concludono soltanto quando l'espressione dopo la UNTIL è soddisfatta. Per esempio:

```
200 DATA 12,234,31,45,65,0,76,81
250 REPEAT
```

TRA LE RIGHE

È bene numerare i programmi BASIC con multipli di 10: ciò permette, eventualmente, l'aggiunta di ulteriori linee senza dover variare l'intera numerazione

```
300 READ numero: somma = somma + numero
350 UNTIL numero = 0
400 PRINT "La somma è :";somma
```

Questo formato, molto più chiaro, è meno soggetto a quei banali errori nei quali si incorre usando dei cicli basati sulle GOTO o sulle frasi FOR...NEXT.
Il BASIC del BBC è dotato, inoltre, di tre comode istruzioni per la ricerca degli errori (debugging): TRACE, ON ERROR... e ERL. TRACE provoca la visualizzazione del numero di linea attualmente in corso d'esecuzione; ON ERROR GOTO (o GOSUB) consente di far deviare l'esecuzione del programma, in caso d'errore, ad una subroutine per l'individuazione ed il trattamento dell'errore; ERL, infine, è una variabile "di sistema", ossia il suo valore viene modificato direttamente dal sistema operativo all'insorgere di un errore. Il programmatore può esaminarne il contenuto per stabilire il numero di linea nella quale si è verificato l'errore.
Vi sono ulteriori aspetti del BASIC adottato per il Micro BBC che meritano attenzione: il comando VDU, le chiamate al sistema operativo, e via dicendo. Esse formano oggetto di alcune lezioni trattate più oltre.

LA GRAFICA COL BBC

Sul Micro BBC, i comandi del BASIC direttamente interessati alla grafica sono:

MODE

Questo serve per selezionare uno dei sette modi grafici disponibili:

Modo	Grafica	Colori	Testo
0	640 x 256	2	80 x 32
1	320 x 256	4	40 x 32
2	160 x 256	16	20 x 32
3	text only	2	80 x 25
4	320 x 256	2	40 x 32
5	160 x 256	4	20 x 32
6	text only	2	40 x 25
7	Teletext	2	40 x 25

Il BBC Modello A dispone soltanto dei modi 4, 5, 6 e 7, mentre nel Modello B sono accessibili tutti e sette. Nei modi da 0 a 6, le caratteristiche dei caratteri si possono alterare mediante il comando VDU. Nel modo 7, le forme dei caratteri sono prestabilite e corrispondono al formato Teletext e non al normale codice ASCII.

COLOUR

Permette la selezione dei colori dello sfondo e del testo, tra i 16 disponibili, a seconda del modo grafico scelto. Il formato è

COLOUR N

dove N è compreso tra 0 e 15 per i colori dei testi, e tra 128 e 143 per i colori dello sfondo. I colori prescelti dal valore di N, comunque, variano a seconda del modo grafico impiegato. Sul manuale sono presenti delle comode tabelle di riferimento.

VDU

Questo comando è comodissimo. VDU A equivale a PRINT CHR$(A). Analogamente, VDU A,B,C equivale a PRINT CHR$(A);CHR$(B);CHR$(C).
Ciò semplifica enormemente la visualizzazione di speciali caratteri sullo schermo.
Questo comando serve anche per la definizione di nuovi caratteri.

CLG

Serve per "ripulire" l'area di schermo dedicato alla grafica, riportando il cursore nell'angolo in alto a sinistra dello schermo.

CLS

Serve per "ripulire" l'intero schermo (grafico e non), riportando il cursore nell'angolo superiore sinistro dello schermo.

DRAW

Nei modi grafici 0, 1, 2, 4 e 5 serve per tracciare linee sullo schermo. Il formato è

DRAW X,Y

Le coordinate X ed Y definiscono il punto "d'arrivo" della linea. Il punto iniziale può essere definito sia dalla posizione finale della linea precedente sia mediante il comando MOVE.

GCOL

Serve a definire i colori di sfondo e di rilievo per la grafica. Il formato è

GCOL N,M

dove N definisce come deve essere impiegato il colore, mentre M definisce il colore "logico" da impiegare (come per il comando COLOUR). Il valore di N ha i seguenti effetti:

0 - Tracciamento nel colore indicato da M
1 - OR logico di M con il colore presente sullo schermo
2 - AND logico di M con il colore presente sullo schermo
3 - OR Esclusivo, logico, di M con il colore presente sullo schermo
4 - Inversione del colore presente sullo schermo

MOVE

Serve per posizionare il cursore alle coordinate specificate. Il formato è

MOVE X,Y

L'effetto è identico a quello di DRAW, ma non viene tracciata alcuna linea.

PLOT

Questo comando è molto flessibile e serve per svariate funzioni grafiche, tra le quali l'accensione o lo spegnimento di un punto sullo schermo, il tracciamento di linee o di triangoli. Il formato è

PLOT K,X,Y

dove K, che assume valori da 0 a 255, definisce il tipo di grafica da usare (si vedano le relative tabelle sul manuale d'uso).

POINT

Con questo comando si può "leggere" il valore relativo al colore di un particolare punto dello schermo. Il formato è

VARNUM = POINT(X,Y)

dove VARNUM è una normale variabile numerica.

I programmi monitor

Esaminare la memoria spetta ai programmi monitor, ma occorre la notazione esadecimale

A questo punto del corso sul linguaggio macchina è bene approfondire la questione della notazione usata per rappresentare i numeri al computer. Il sistema decimale (quello normalmente adoperato nella vita di tutti i giorni) non presenta alcun ostacolo, quello binario lo abbiamo affrontato a pagina 498.

Non dovrebbe essere necessario ricordare che questi due sistemi non sono che un modo diverso di rappresentare la medesima entità: il numero.

La situazione è simile a quella che si affronta nei vari linguaggi umani, quando il numero cinque può esser chiamato cinq, pente, fünf o five: il numero rimane sempre lo stesso, cambia soltanto il sistema di rappresentarlo. Esiste una corrispondenza univoca tra i numeri espressi nella lingua italiana e quelli, ad esempio, espressi in tedesco. L'aritmetica conduce ai medesimi risultati, quale che sia la lingua usata per esprimere le singole componenti di un'espressione.

Una circostanza molto simile si ha passando da un sistema numerico all'altro: il numero cinque non cambia, sia esso rappresentato con il simbolo 5 o con 101b (la "b" serve ad indicare che il numero è espresso in binario). L'unico motivo di preferenza, nell'adottare un sistema piuttosto di un altro, è dettato dall'abitudine o dalla convenienza.

Il sistema decimale è conveniente, per gli esseri umani, perché è quello più comunemente diffuso, ma non è certo l'unico: negli orologi digitali, ad esempio, si nota facilmente che la numerazione delle ore è alquanto bizzarra: in parte decimale, in parte modulo 60 (60 minuti in un'ora, 60 secondi in un minuto) ed in parte modulo 24 (ci sono 24 ore in un giorno).

Addirittura, prima del 1971, il sistema monetario in gran Bretagna era in parte modulo 12 (le pence in uno scellino) e modulo 20 (gli scellini in una sterlina)!

Quando si parla di computer, è naturale introdurre per primo il concetto di numerazione binaria, poiché è facilmente confrontabile con quanto avviene, a livello logico, nella circuiteria interna, dove le correnti assumono o il valore 1 (on) o quello 0 (off). Se dovessimo trattare sempre numeri binari di soli otto bit, con un poco di pratica riusciremmo in breve tempo a convertire con destrezza da decimale in binario e viceversa. Purtroppo, però, gli indirizzi di memoria in particolare e la gran parte di numeri utili, in generale, sono abitualmente troppo grandi per esser rappresentati con un singolo byte, cosicché i progettisti ed i programmatori hanno sentito la necessità di inventare un nuovo sistema, per così dire, "intermedio". In realtà sono stati inventati più sistemi, ma quelli principali restano l'"esadecimale" e l'"ottale". Il primo, oggi diffusissimo sui microcomputer, ha come base 16, mentre il secondo, usato in passato per i computer delle prime generazioni, ha per base 8.

L'uso dei numeri esadecimali

Nell'esaminare le differenze tra il sistema decimale e quello binario, abbiamo notato che dalla scelta di una particolare base di numerazione derivano due precise conseguenze: una è che la base costituisce il numero di simboli necessari per il particolare sistema, e l'altra è che la base è il fattore di moltiplicazione nel passaggio da una colonna all'altra di un numero a più cifre. Ad esempio, nel sistema decimale i simboli sono dieci (da 0 a 9) e, nel passare da una posizione all'altra, verso sinistra, il simbolo viene moltiplicato per 10.

Nel sistema esadecimale, pertanto, occorrono 16 simboli: vengono impiegati i numeri da 0 a 9 e, non essendo questi sufficienti, le lettere da A ad F.

Contare in esadecimale è dunque relativamente semplice. Arrivati al 9 si passa ad A (10 decimale), B (11 decimale) e via dicendo fino ad F (15 decimale). Ciò esaurisce l'arco dei numeri esadecimali composti da una sola cifra, così il numero successivo ad F è 10 (la dizione corretta è "uno-zero"), che corrisponde a 16 decimale.

Nel sistema decimale, quindi, vi sono le colonne delle Unità, delle Decine, delle Centinaia, e delle Migliaia. Per analogia, nel sistema esadecimale (o più brevemente: "hex"), si hanno le colonne delle Unità, dei Sedicesimi, dei Duecentocinquantaseiesimi, e così via.

Un vantaggio, rispetto al sistema binario, appare subito evidente: l'arco di numeri rappresentabili con quattro bit corrisponde esattamente a quello rappresentabile con una singola cifra hex (ossia da 0 a 15 decimale). Alcuni esempi per rendere più chiaro il concetto:

Decimale	Binario	Hex
0	00000000	0
1	00000001	1
2	00000010	2
3	00000011	3

.....

7	00000111	7
8	00001000	8
9	00001001	9
10	00001010	A
11	00001011	B
12	00001100	C
13	00001101	D
14	00001110	E
15	00001111	F
16	00010000	10
17	00010001	11
.....		
24	00011000	18
25	00011001	19
26	00011010	1A
27	00011011	1B
.....		
31	00011111	1F
32	00100000	20
33	00100001	21

Il campo numerico di un byte (composto da otto bit) è dunque rappresentabile con due cifre hex:

da	0 a 255	in decimale
da	00000000 a 11111111	in binario
da	0 a FF	in esadecimale

Per convertire un numero hex in binario, è sufficiente esprimere ciascuna cifra hex nell'equivalente numero binario a quattro bit. Se un byte viene espresso mediante 2 cifre hex, allora la cifra hex a sinistra corrisponde ai quattro bit a sinistra; analogamente accade per le cifre a destra. In termini più precisi, metà byte (ossia quattro bit) viene chiamato un "nybble". Il nybble a sinistra, dunque, rappresenta il semy-byte più significativo, mentre quello a destra il meno significativo. Ecco un esempio:



È estremamente importante familiarizzare con il sistema numerico esadecimale, poiché facilita in modo impressionante la manipolazione di quantità numeriche ad otto bit, così comuni nei microcomputer (senza contare che gli indirizzi di memoria, normalmente usati nel codice macchina, si ricordano molto più facilmente in forma esadecimale che non decimale). Ammettiamo che l'abituarsi ad un nuovo sistema richieda un po' di tempo, ma una volta imparato il sistema hex, ci si chiederà spesso come potremmo farne a meno!
In fondo a questa lezione, vengono presentati alcuni programmi (per il Micro BBC, per il Commodore 64 e per lo Spectrum) che consentono di esaminare il contenuto delle varie locazioni di memoria. Questi programmi, che abbiamo chiamato "Mempeek", per prima cosa chiedono l'indirizzo iniziale dal quale partire nell'esame (ossia il numero della prima locazione da visualizzare). La seconda domanda riguarda il numero di locazioni che si desiderano esaminare. Per fare un esempio, potremmo chiedere di iniziare l'esame dalla locazione numero 1953, e visualizzare i quattro byte che seguono: a sinistra, sullo schermo, comparità il numero decimale 1953, accanto ad esso il valore contenuto dalla locazione 1953, della 1954, della 1955 ed, infine della 1956.
Ciò che dobbiamo ricordarci, comunque, è che anche se noi vediamo sullo schermo che il byte 1956 contiene il valore decimale 175, ciò significa, in realtà, che la zona di memoria individuata dal numero 1956 contiene una serie di otto livelli elettrici. Se un livello di 0 volt indica 0 ed un livello di 5 volt indica 1, allora il byte 1956 contiene, in binario, il valore 10101111. Questo numero, espresso in notazione decimale è, appunto, 175.
Conviene ricordare che spesso, parlando dei computer, si è portati ad usare una certa imprecisione. Ricorrere, perciò, ad esempi di carattere strettamente fisico può aiutare a meglio comprendere la materia.
Il contenuto di un byte visualizzato sullo schermo non è, in effetti, quello "vero": ciò che vediamo è una rappresentazione simbolica, ottenuta mediante dei caratteri associati all'effettiva serie di correnti elettriche. Ossia, abbiamo codificato i segnali in una forma binaria, e abbiamo convertito questa in forma decimale, associando a ciascun numero un carattere del codice ASCII (l'American Standard Code for Information Interchange). È, questo, un codice internazionalmente accettato ed adottato per la maggioranza dei microcomputer: a ciascun numero compreso tra 0 e 127 viene associato il simbolo di un carattere alfanumerico. Pertanto, al numero 65 decimale corrisponde una "A", al 66 una "B", e così via. Tra i caratteri non alfabetici, il codice 32 corrisponde allo spazio, il 42 all'asterisco, il 13 al ritorno carrello e via dicendo. I caratteri ASCII visualizzabili hanno un codice compreso tra 32 e 127, quelli al di fuori di questo arco o sono caratteri "di controllo" o non vengono usati o, ancora, vengono usati per funzioni specifiche, non standard. Ecco perché i programmi che presentiamo, quando incontrano un codice esterno al campo menzionato, visualizzano un semplice punto. Nella prossima lezione di questo corso, forniamo una tabella completa dei caratteri ASCII con codice compreso tra 0 e 127.
Un'esplorazione del "set" (insieme) di caratteri ASCII è particolarmente utile come base per la comprensione del linguaggio macchina. In primo luogo perché rafforza il concetto secondo il quale l'interpretazione del contenuto di una cella di memoria è un fatto puramente soggettivo: si può egualmente asserire che un byte contiene un numero, un indirizzo o un carattere codificato o, ancora, un'istruzione. Si tratta pur sempre di un dato che occorre interpretare. In secondo luogo, perché offre una visione maggiormente comprensibile del contenuto della memoria, spe-

cialmente di quelle parti che contengono effettivamente i codici dei caratteri alfanumerici, o di quelle contenenti sezioni del sistema operativo.
I sistemi operativi, ad esempio, contengono tutta una serie di messaggi e di avvisi d'errore (ad esempio: READY, oppure SYNTAX ERROR nel BASIC, ecc.). Tutto ciò che deve essere comunicato all'operatore, viene conservato, nella stragrande maggioranza dei casi, usando il codice ASCII. A qualcuno, questa rivelazione può far riflettere sulle reali capacità intellettive di un computer e su quanto queste siano diverse da quelle umane. Il nostro cervello non conserva le informazioni usando simili formati: ogni pensiero viene formulato sul momento ed espresso mediante l'associazione di parole appropriate.

LE MAPPE DI MEMORIA

Una mappa di memoria non è che una semplice rappresentazione schematica dell'uso che viene fatto della memoria disponibile.
Alcune zone di essa vengono adoperate sempre per le stesse operazioni.
Nel Commodore 64, ad esempio, i byte da 0 a 1024 sono sempre usati dal BASIC come area "di lavoro". Altre zone di memoria possono invece variare in dimensioni, a seconda della lunghezza del programma o della particolare fase d'esecuzione di esso. I limiti tra le varie aree possono quindi essere "fissi" (indicati con linee continue nei diagrammi che seguono) o "fluttuanti" (indicati col tratteggio). Nella mappa di memoria del Commodore, i limiti della RAM di video sono fissi (dal byte 1024 al 2048), mentre quelli relativi alla zona di memoria occupata dalle variabili del BASIC possono variare, a seconda del numero e della lunghezza delle variabili usate ad un dato momento.
I programmi nella pagina a fianco possono servire per determinare i limiti delle varie zone. Il Commodore possiede sei "puntatori" ai limiti fluttuanti (chiamati anche "variabili di sistema") e più sotto viene dato un esempio di come venga calcolato l'indirizzo di una particolare area di memoria. Il BASIC del BBC possiede quattro variabili di sistema, mentre lo Spectrum ne ha cinque.
Si ricordi, ad ogni buon conto, che la mappa di memoria di un computer non è che la rappresentazione statica di un qualcosa che varia continuamente ed i limiti fluttuanti possono cambiare di momento in momento.
Nella pagina a fianco vengono anche offerti dei suggerimenti per fare alcuni esperimenti coi puntatori.

Le variabili di sistema del Commodore

43,44 Inizio del testo dei programmi BASIC
45,46 Inizio dell'area delle variabili BASIC
47,48 Inizio delle variabili a indice BASIC
49,50 Fine delle variabili a indice
51,52 Inizio zona delle variabili alfanumeriche
53,54 Fine zona delle variabili alfanumeriche

Esempio

Si usi il programma Mempeek riportato a pag. 539 per esaminare alcune locazioni. Sullo schermo compare qualcosa di simile a:

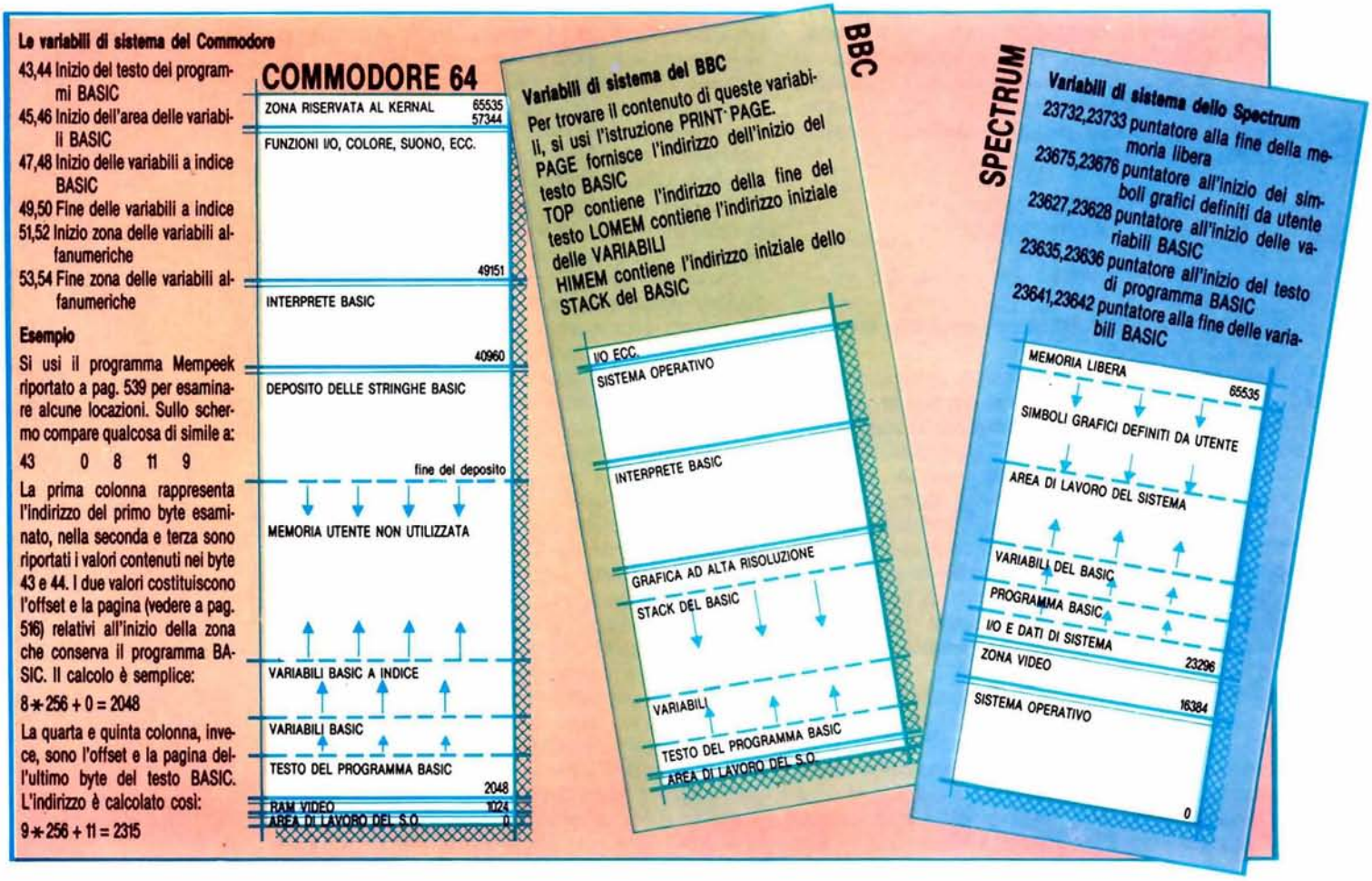
43 0 8 11 9

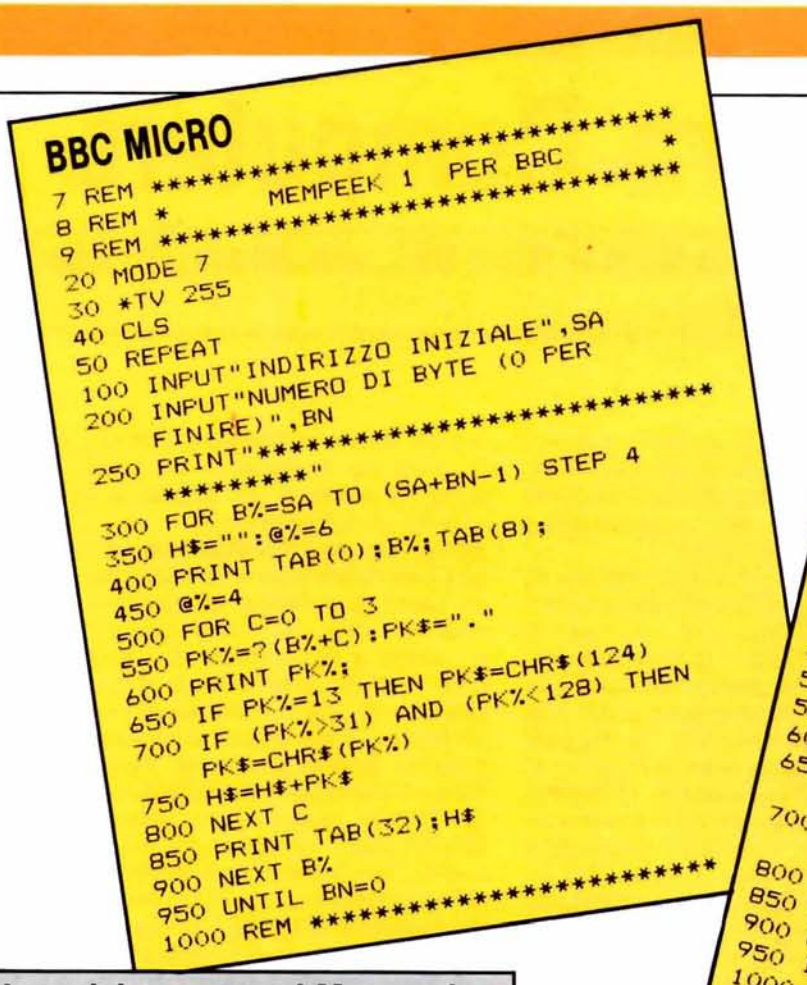
La prima colonna rappresenta l'indirizzo del primo byte esaminato, nella seconda e terza sono riportati i valori contenuti nei byte 43 e 44. I due valori costituiscono l'offset e la pagina (vedere a pag. 516) relativi all'inizio della zona che conserva il programma BASIC. Il calcolo è semplice:

8 * 256 + 0 = 2048

La quarta e quinta colonna, invece, sono l'offset e la pagina dell'ultimo byte del testo BASIC. L'indirizzo è calcolato così:

9 * 256 + 11 = 2315





BBC MICRO

```
7 REM ********************************
8 REM *        MEMPEEK 1  PER BBC     *
9 REM ********************************
20 MODE 7
30 *TV 255
40 CLS
50 REPEAT
100 INPUT"INDIRIZZO INIZIALE",SA
200 INPUT"NUMERO DI BYTE (0 PER
    FINIRE)",BN
250 PRINT"*************************
    *********"
300 FOR B%=SA TO (SA+BN-1) STEP 4
350 H$="":@%=6
400 PRINT TAB(0);B%;TAB(8);
450 @%=4
500 FOR C=0 TO 3
550 PK%=?(B%+C):PK$="."
600 PRINT PK%;
650 IF PK%=13 THEN PK$=CHR$(124)
700 IF (PK%>31) AND (PK%<128) THEN
    PK$=CHR$(PK%)
750 H$=H$+PK$
800 NEXT C
850 PRINT TAB(32);H$
900 NEXT B%
950 UNTIL BN=0
1000 REM ************************
```



SPECTRUM

```
7 REM************************
8 REM* MEMPEEK 1 PER  SPECTRUM *
9 REM************************
30 DIM H$(4)
50 FOR L = 0 TO 1 STEP 0
100 INPUT"INDIRIZZO INIZIALE";SA
200 INPUT"NUMERO DI BYTE (0 PER
    FINIRE)";BN
250 PRINT"*********************
    **********"
300 FOR B=SA TO (SA+BN-1) STEP 4
350 LET H$="...."
400 PRINT B;TAB 7;
500 FOR C=0 TO 3
550 LET PK=PEEK(B+C)
600 PRINT PK;" ";
650 IF (PK>31) AND (PK<128) THEN
    LET H$(C+1)=CHR$ PK
700 IF PK=13 THEN LET H$=(C+1)=
    "■"
800 NEXT C
850 PRINT TAB 26;H$
900 NEXT B
950 IF BN=0 THEN LET L=2
1000 NEXT L
1050 REM************************
```

L'uso dei programmi Mempeek

Dopo aver trascritto il programma Mempeek adatto al proprio computer, si ricordi di memorizzarlo (comando SAVE) e, soltanto dopo un'attenta verifica, lo si lanci con il comando RUN.

Per prima cosa, il programma chiede l'indirizzo iniziale ed il numero di locazioni da esaminare. Ambedue i valori devono essere numeri interi positivi compresi tra 0 e 65535. Per terminare il programma, si risponde 0 alla domanda sul numero di locazioni da esaminare. Supponiamo di voler iniziare dalla locazione 230. Sullo schermo apparirà qualcosa di simile a:

```
INDIRIZZO INIZIALE? 230
NUMERO DI BYTE (0 PER FINIRE)? 8
********************************
230  193   32   65   49   .A1
234  129   64   93   98   .@]b
INDIRIZZO INIZIALE?
```

Nella colonna più a sinistra è riportato l'indirizzo iniziale, in notazione decimale, seguito da quattro valori, che rappresentano (sempre in decimale) il contenuto delle quattro locazioni consecutive. Nell'ultima colonna sono riportati i caratteri ASCII equivalenti ai quattro valori. Qualora un valore sia esterno al campo ASCII, viene usato un semplice punto.

Dopo aver fatto un po' di pratica, può essere interessante andare a curiosare nelle varie zone di memoria, magari scoprendo dove e come vengano conservati, ad esempio, i messaggi d'errore del BASIC o del sistema operativo. In questa ricerca è utile consultare sia le mappe di memoria riportate nella pagina a fianco, sia il manuale dell'apparecchio.

Se si riescono ad individuare le "variabili di sistema", che indicano i limiti delle varie aree di memoria, potrebbe essere interessante aggiungere qualche linea di REM, nel programma BASIC, per vedere quali ripercussioni si abbiano nei puntatori. Oppure si può variare il contenuto della zona di memoria riservata alle variabili, aggiungendo al programma, ad esempio:

```
3 DIM Z$(254)
4 LET X$ = ""
5 FOR M = 1 TO 255: LET X$ = X$ + "*": NEXT M
```

COMMODORE 64

```
7 REM*******************************
8 REM* MEMPEEK 1 PER COMMODORE 64 *
9 REM*******************************
30 PRINT CHR$(147)   :REM PULIZIA
   SCHERMO
40 PRINT CHR$(142)   :REM MODO
   MAIUSCOLO
50 FOR LP=0 TO 1 STEP 0
100 INPUT"INDIRIZZO INIZIALE";SA
200 INPUT"NUMERO DI BYTE (0 PER
    FINIRE)";BN
250 PRINT"************************"
300 FOR B=SA TO (SA+BN-1) STEP 4
350 H$=""
400 PRINT B;TAB(8);
500 FOR C=0 TO 3
550 PK = PEEK(B+C):PK$="."
600 PRINT TAB(8+5*C);PK;
650 IF PK=0 THEN PK$=CHR$(122)
700 IF (PK>31) AND (PK<128) THEN
    PK$=CHR$(PK)
750 H$=H$+PK$
800 NEXT C
850 PRINT TAB(32);H$
900 NEXT B
950 IF BN = 0 THEN LP=1
1000 NEXT LP
1050 REM************************
```

La Acorn Computers

Chris Curry: dal piccolo MK14 al Micro BBC, uno dei migliori microcomputer per la didattica



Chris Curry

Herman Hauser



Il fondatore della Acorn, Chris Curry, è un ex-dipendente e grande amico di Sir Clive Sinclair. Curry entrò nella Sinclair Radionics nel 1965, quando Sinclair gli offrì un posto come ingegnere addetto allo sviluppo.
Uno dei progetti al quale Curry lavorò fu quello che portò alla produzione della calcolatrice Executive, nel 1971. Per i successivi cinque anni, egli si dedicò totalmente alla realizzazione di piccole calcolatrici tascabili, considerate da molti le progenitrici dei moderni home computer. Nel 1975, la Sinclair Radionics cessò la propria attività e Curry si unì a Sinclair in un'operazione chiamata *Science of Cambridge*, che si sarebbe occupata di realizzare delle scatole di montaggio per apparecchi elettronici.
Uno dei prodotti che ebbe gran successo fu un orologio digitale da polso, con calcolatrice incorporata. Ma Curry cominciò ad interessarsi molto alle prime schede a microprocessore, prodotte negli Stati Uniti, e ne realizzò un prototipo, chiamato MK14 (da Microprocessor Kit con 14 chip). La scheda montava un microprocessore della National Semiconductor, 256 byte di memoria RAM, una piccola memoria contenente un elementare programma monitor, un display a LED ed una sezione d'alimentazione.
Ben presto, la ditta cominciò ad esser tempestata da telefonate e da richieste di consigli da parte di hobbysti ed appassionati d'elettronica, tanto che Curry decise di assumere un giovane neolaureato di Cambridge, Herman Hauser. Poco tempo dopo, le idee di Curry e quelle di Sinclair presero direzioni diverse, e Curry pensò di creare una ditta in proprio, portando con sé Hauser. La ditta venne chiamata *Cambridge Processor Unit*, le cui iniziali, guarda caso, erano CPU!
Inizialmente, l'attività del piccolo ufficio di Cambridge si limitava a consulenze sul settore dell'elettronica e dell'informatica.
Il successo dell'MK14, e le esperienze condotte negli USA, dimostravano chiaramente che ciò che il pubblico desiderava era un computer, di piccole dimensioni, completo di un linguaggio BASIC. Avendo sviluppato un'elementare versione di questo linguaggio per uno dei propri clienti, Curry pensò di adattarlo ad un nuovo apparecchio da mettere in commercio. Nacque così l'Atom e la ditta di Curry divenne la Acorn. Il microcomputer era stato commercializzato con l'intento di catturare l'interesse del settore didattico, ma la gran parte degli istituti scolastici ritenne che la versione di BASIC fornita differisse troppo da quella, ormai diffusa, della Microsoft. Tuttavia, l'apparecchio trovò un suo discreto mercato tra gli hobbysti.
La Acorn non si dette per vinta e realizzò un altro computer orientato alla didattica: il Proton.
Quando però, nel 1981, il Proton era nelle fasi preliminari di produzione, Curry venne a sapere che la BBC aveva indetto un concorso per la fornitura di un microcomputer da adottare per una serie di trasmissioni didattiche sull'informatica. Immediatamente, Curry si precipitò alla BBC per fare una dimostrazione con un nuovo prototipo di computer, basato sul chip 6502.
La BBC era alla ricerca di un apparecchio facile da usare per dei principianti, ma, al tempo stesso, espandibile fino a divenire un computer dalle prestazioni semiprofessionali. Inoltre, il prezzo doveva essere estremamente contenuto. La concorrenza, in particolare quella della Sinclair Research, fu enorme, ma la Acorn vinse l'appalto. Entrò così in produzione il Micro BBC, al ritmo di 12.000 apparecchi al mese.



per concessione della ACORN

Esami al computer

Nei paesi anglosassoni il settore del software didattico è sviluppatissimo: che cosa e come s'impara?

In Gran Bretagna gli studenti possono farsi un ripasso delle materie scolastiche acquistando uno dei tanti programmi "di ripasso", di cui solo pochi richiedono la presenza di un programmatore. La maggior parte del software di questo tipo consiste in una serie di domande, pertinenti ad una certa materia, e lo studente si limita ad inserire delle risposte. Il programma avverte automaticamente se queste sono più o meno corrette. Questo metodo, anche se piuttosto discutibile sotto il profilo didattico, può essere però adattato ad un gran numero di materie per integrare o addirittura sostituire, in qualche raro caso, il libro di testo. Simili programmi consentono di ottenere risultati soddisfacenti, ma si deve riconoscere che, anche quelli più costosi, sono piuttosto noiosi. Tuttavia, le prossime generazioni di software di ripasso certamente si avvarranno ampiamente della grafica a colori, disponibile sulla maggior parte degli home computer.



Le lingue straniere

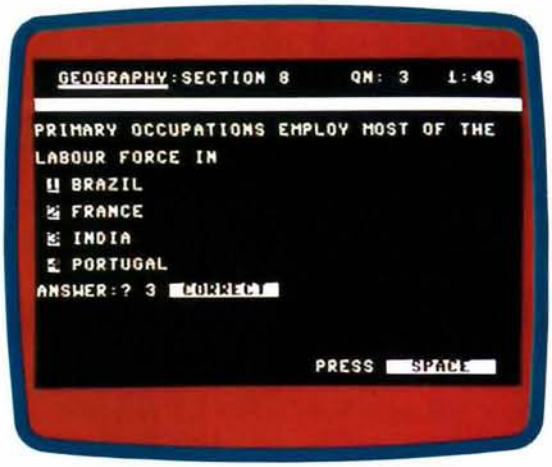
Esistono dei pacchetti software di ripasso, adatti a funzionare su numerose macchine più comunemente adottate nella didattica (quali il Micro della BBC, lo ZX Spectrum ed anche gli apparecchi Atari), concepiti per lo studio delle lingue straniere. Una ditta inglese, la Cosmos Software, ha prodotto su cassetta una serie di pacchetti linguistici utilizzabili sullo Spectrum e sul Modello B della BBC. I programmi (per il Francese, il Tedesco e lo Spagnolo) seguono tutti il medesimo principio di funzionamento e tutti prevedono due livelli d'insegnamento. Il livello A è costituito da una lista di parole o di brevi frasi (composte da non più di 59 caratteri), presentate tanto in lingua madre (in questo caso l'inglese!), che nella lingua di studio, divise in gruppi come: "Famiglia", "Cibo", "Animali" e simili; il livello B introduce gli aggettivi, gli avverbi, le coniugazioni e i tempi dei verbi. Lo studente può anche creare nuovi elenchi di vocaboli, che possono essere salvati su cassetta e quindi richiamati. Ogni cassetta del pacchetto è suddivisa in 16 lezioni ed ogni lezione può contenere un massimo di 250 elementi. Questo limite è valido anche per gli elenchi di parole create dallo studente. Egli può scegliere tra più metodi d'istruzione: uno presenta le parole sia nella propria lingua che in quella straniera, un secondo consiste in un "autotest" nel quale le parole che compaiono sul video appartengono ad una sola delle due lingue, il terzo, infine, è una prova "a tempo".



per concessione della PILOT SOFTWARE

La lingua inglese

L'insegnamento della lingua del proprio paese si differenzia notevolmente da quello di una lingua straniera. L'obiettivo è nel primo caso di insegnare ad usare nel modo migliore la propria lingua. Il computer è quindi utile perché si dimostra particolarmente adatto all'insegnamento della grammatica. Tuttavia la capacità di scrivere riassunti, perifrasi, componimenti e temi dipende più dalle doti individuali, che dalla conoscenza delle regole grammaticali, quindi in questo caso i pacchetti di ripasso per computer non risultano di aiuto immediato. La *Lingua inglese* della Commodore, venduta su cassette, si basa su materiale didattico fornito dalle scuole internazionali di corrispondenza e fa parte di una più vasta gamma di prodotti progettati per il Vic-20 (con espansione di memoria). In esso



vengono curati con particolare attenzione la grammatica e l'uso delle parole. Il menu principale offre varie possibilità di scelta: Composizione (caratterizzata da vari esercizi sulla scelta e sull'uso delle parole), Sillabazione, Grammatica e Livello di comprensione. Questo pacchetto è corredato da un manuale che ne spiega, con adeguati esempi, il funzionamento e presenta alcuni esercizi di ricapitolazione e di spiegazione.

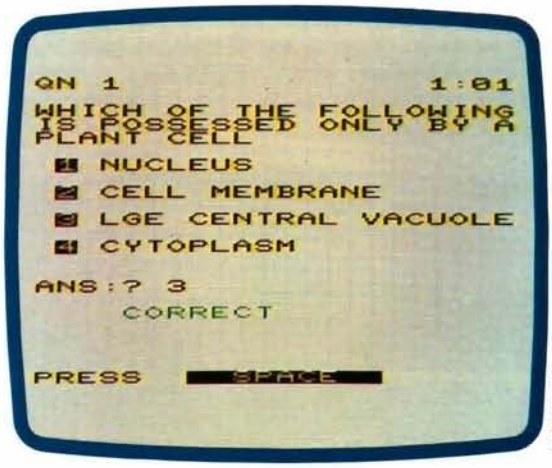


Letteratura inglese

Sfortunatamente per i produttori di software, i programmi scolastici di letteratura si basano su particolari opere letterarie e la lista dei loro titoli è piuttosto nutrita. "Orgoglio e Pregiudizio", un programma prodotto dalla Software Sussex, costituisce un buon esempio dell'enorme quantità di lavoro necessaria per questo tipo di software. Scritto per il 380Z della Research Machine (ma con opportune modifiche, adattabile a tutti i computer funzionanti con il CP/M) e per la serie 3000 e 4000 della Commodore, questo pacchetto è contenuto in tre dischetti e descrive i personaggi, le tematiche, la trama ed il contesto sociale trattati nell'omonimo romanzo. Viene utilizzato, assieme ad altri prodotti di carattere letterario e storico della stessa Casa, come complemento al programma scolastico delle scuole inferiori britanniche, benché sembri più adatto agli studenti dei corsi superiori. Sebbene il metodo sia basato su scelte multiple, per proporre le domande, ciascun quesito è corredato da ampie spiegazioni.

Storia e geografia

Lo studio della Storia, della Geografia e dell'Economia implica l'interpretazione di fatti certi ed è spesso basato su uno studio mnemonico della materia: si adatta bene, quindi, ad un ripasso eseguito con il computer. La Sussex Software ha pubblicato più di una dozzina di pacchetti adatti a questo scopo, per il 380Z ed il Commodore 4000, ma esistono anche versioni adatte per lo Spectrum della Sinclair (commercializzate con la sigla Akadimias) e, presto, anche per il Micro della BBC.
La Commodore ha prodotto un "package" storico chiamato *Storia del XX Secolo*. Pubblicato in collaborazione con la Ivan Berg Software (coeditrice dei prodotti per il Vic-20) e sviluppato in base al materiale editoriale fornito dalla Hodder e Stoughton, questo pacchetto, suddiviso in 8 sezioni, è uno strumento di ripasso completo della storia politica, economica e sociale del XX Secolo.



LIZ HEANEY

Matematica e scienze

Lo studio della Matematica e delle Scienze, a livello delle scuole inferiori, non dà adito ad interpretazioni personali. Queste sono forse le materie che maggiormente si adattano alle metodologie dei programmi di ripasso per computer. Per questo settore, è stata prodotta una gran quantità di software, che riguarda tanto gli aspetti generali delle singole materie, che argomenti specifici (ad esempio, la geometria e la trigonometria sono comprese nel corso di studi della matematica).
Il pacchetto software di matematica della Commodore, suddiviso in due sezioni, è un ottimo esempio di corso di ripasso per la matematica generale. Simili programmi sono disponibili anche dalla Rose Software (per lo Spectrum) e dalla SciCAL (per il Micro BBC). Poiché molti argomenti matematici e scientifici possono essere meglio chiariti con semplici diagrammi, viene ampiamente sfruttata la grafica dei vari computer.

Unità a dischi Atari

Una descrizione dei comandi usati dal DOS Atari per la gestione degli archivi su dischi

I computer Atari 400 e 800 sono probabilmente stati i primi ad essere progettati appositamente per uso domestico. Quasi contemporaneamente, la Atari produsse l'unità a dischetto singolo 810, che accresceva le capacità del computer, ma che non ebbe molto successo. Dopo l'introduzione dell'Atari 600, però, l'unità a dischetti 810 è tornata d'attualità.

I dischetti impiegati sono da 5 1/4 pollici, a singola faccia e singola densità. L'unità viene collegata al computer attraverso una speciale porta. È possibile collegare tra loro fino a quattro 810 "in cascata", numerandoli da 1 a 4 a seconda della posizione di un interruttore di codifica del drive, posto posteriormente a ciascuna unità. La 810 è dotata di un proprio microprocessore interno, ma non può essere considerato una vera e propria unità "intelligente" perché una parte del DOS II Atari, il Sistema Operativo per Dischi normalmente usato, deve essere caricato nella RAM, prima di accedere all'impiego dei dischetti. Il DOS occupa approssimativamente 5 Kbyte di RAM utente e lascia liberi appena 8 Kbyte di RAM su macchine a 16 Kbyte, come l'Atari 600, per il quale saranno presto disponibili, schede di espansione che porteranno la capacità della RAM a 64 Kbyte.

Il DOS Atari, registrato su dischetto, viene consegnato insieme con l'unità a dischetti e si compone di tre file distinti, ma collegati tra loro. Essi sono: il DOS.SYS, che contiene il Sistema di Gestione File e le istruzioni di comando che vengono caricate in RAM quando la macchina è in funzione; il DUP.SYS, un file di servizio per il disco che contiene il menu DOS ed alcune istruzioni di comando DOS; l'AUTORUN.SYS, contenente un file che, in risposta ad un particolare comando, viene automaticamente caricato in RAM ed inviato in esecuzione, permettendo di richiamare il menu DOS e quella parte di DOS che si trova nella RAM.

Per accedere al DOS con la cartuccia BASIC inserita, l'unità a dischetti deve venir accesa prima di dare corrente al computer e di inserire il dischetto del DOS nel drive.

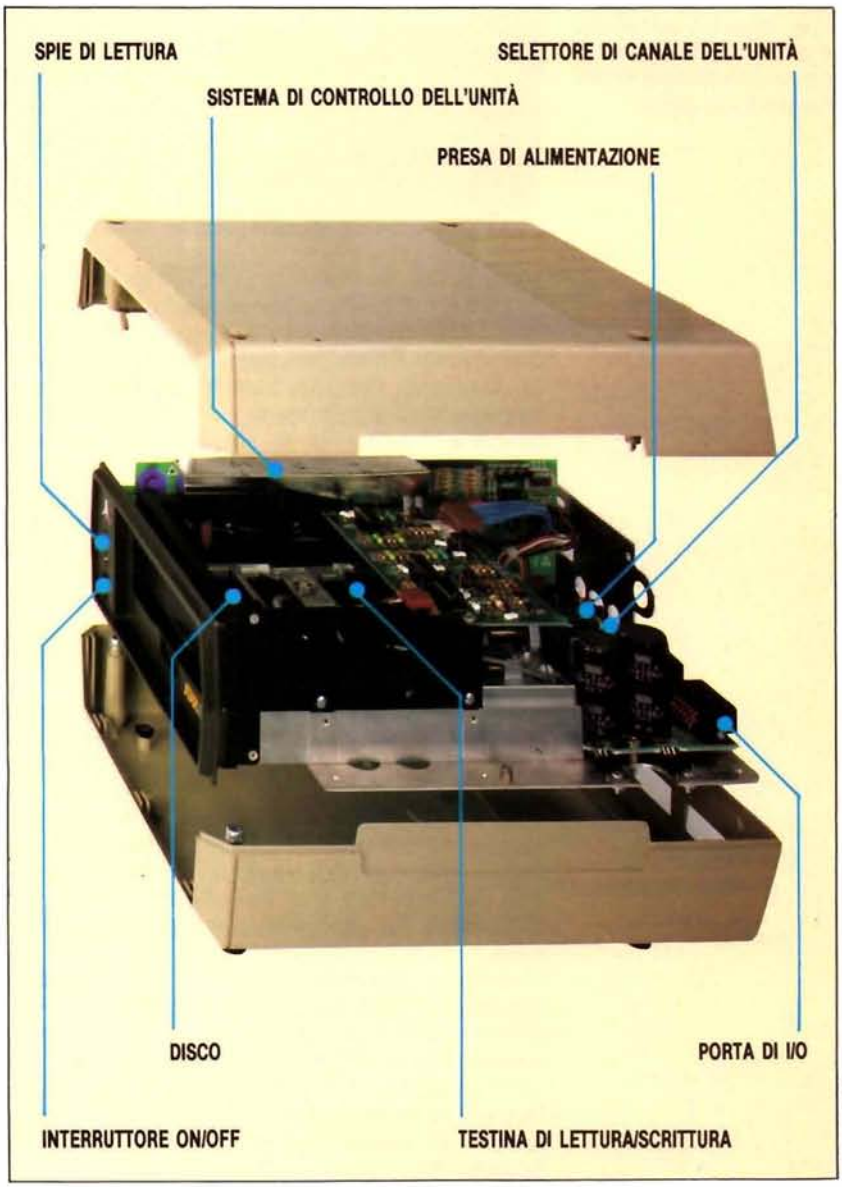
L'accensione del computer fa sì che una parte del DOS.SYS venga "ricopiata" nella RAM. Per richiamare il menu generale del DOS, occorre digitare la sigla DOS e premere RETURN. Se non è innestata la cartuccia, il menu DOS viene richiamato automaticamente.

Il menu offre la scelta tra 15 comandi per la gestione dei dischi. Questo sistema facilita le operazioni su disco, ma presenta un grave inconveniente: usa una parte della memoria a disposizione dell'utente, cancellando qualsiasi programma o dato registrato in quell'area. Per questa ragione i programmi in fase di scrittura o di revisione devono essere salvati su disco prima di richiamare il menu e, solo quando sono completate le varie operazioni, ricaricati in memoria. Il DOS è in grado di eseguire questa procedura automaticamente, se necessario, ma rimane pur sempre un metodo contorto di operare.

Il DOS esegue una verifica automatica dei dati memorizzati grazie ai vari comandi di scrittura e ciò riduce la velocità di trasferimento dei dati a 2,4 Kbyte per secondo. Se si desidera una maggior velocità di trasferimento è possibile disattivare la procedura di

L'Atari 810

I maggiori inconvenienti dell'Atari 810 sono la sua lentezza nel trasferimento dei dati e la sua limitata capacità d'immagazzinamento di 88 Kbyte per disco. Tuttavia il Sistema Operativo per Dischi (DOS) è abbastanza sofisticato



IAN McKINNELL

A bordo di "caccia"

Lo Zaxxon è un gioco, prodotto dalla Atari, che ha riscosso un gran successo fra il pubblico. Il giocatore diviene il pilota di jet da "caccia" in combattimento contro altri aerei o contro basi al suolo. L'immagine sul video rappresenta la visuale che si avrebbe sedendo al posto del pilota



Il "Blue Max"

Un secondo gioco di combattimento prodotto dall'Atari trasporta il giocatore in una differente epoca storica. Il "Blue Max", venduto su dischetto, pone il giocatore al posto di pilotaggio di un biplano da "caccia" della 1ª Guerra Mondiale. Questo gioco non ha la pretesa di essere una simulazione di volo e la sua dote principale è l'ottima qualità della grafica



per concessione della SOFT

verifica con il comando POKE 1913,80. La velocità passa a 4,8 Kbyte per secondo. Il comando POKE 1913,87 ripristina la procedura diverifica. Il comando FORMAT DISK permette di formattare un dischetto vergine, inserito nell'unità selezionata tracciando su di esso 40 tracce, ciascuna suddivisa in 18 settori. Ogni settore contiene 128 byte, tre dei quali sono riservati al Sistema di Gestione File. Il DOS assegna 8 settori alla Directory del disco ed un settore alla Tabella del Contenuto di Volume (equivalente alla RAM) e riserva 4 settori per l'uso del DOS; restano quindi disponibili per la memorizzazione di dati o programmi complessivamente 707 settori di 125 byte ciascuno, ovvero 8.375 byte (circa 86 Kbyte).

I comandi LOAD, SAVE e simili possono essere usati per immagazzinare programmi o file dati, specificando il disco sul quale si trovano. I file possono essere immagazzinati anche in modo da essere letti con accesso sequenziale o casuale (random), un byte alla volta. Quando i file vengono scritti o modificati, la Directory e la Tabella del Contenuto di Volume vengono automaticamente aggiornate.

L'unità a dischetti 810 è stata progettata intorno al 1979 e, benché il DOS II Atari abbia caratteristiche uniche nel suo genere, questo apparecchio è oggi già superato. La limitata capacità dei dischetti (solo 86 Kbyte), la perdita di 8 preziosi Kbyte della memoria utente ed, infine, il prezzo rendono questo apparecchio poco attraente.

I COMANDI PER IL DISCHETTO ATARI

Per le successive descrizioni dei comandi, occorre dare una descrizione standard del formato dei comandi. Eccola:

COMANDO "DN:NOMEFILE.EST"

COMANDO rappresenta il comando DOS, N è il numero di selezione dell'unità a dischetti (da 1 a 4), NOMEFILE può contenere fino ad 8 caratteri che identificano il file (il primo deve essere alfabetico); EST rappresenta l'eventuale estensione del file (può essere usata per indicare il tipo di file memorizzato). Quando questa estensione è obbligatoria, ciò è indicato con FSP. Con alcuni comandi, inoltre, è consentito l'uso di simboli "jolly" come "?" (al posto di un singolo carattere) e "*" (in sostituzione di un qualsiasi numero di caratteri validi situati dopo questo simbolo). Per selezionare i comandi offerti dal menu DOS, occorre digitare sulla tastiera la lettera corrispondente, seguita da RETURN. Per brevità, nei seguenti esempi, NMF equivale a D1:NOMEFILE.EST.

A. DISK DIRECTORY

Seguito da RETURN permette di visualizzare un elenco di tutti i nomi dei file contenuti nel disco inserito nell'unità 1, la loro estensione ed il numero di settori occupati da ciascun file.

B. RUN CARTRIDGE

Questa scelta trasferisce il controllo del computer alla cartuccia innestata, di solito la cartuccia BASIC.

C. COPY FILE

Permette di copiare un file da un dischetto ad un altro (o sullo stesso dischetto, ma con un nome diverso). Ad esempio:

D1:NOMEFILE.EST, D2:NOMEFILE.EST

copia il NOMEFILE.EST dall'unità a dischetti 1 alla 2. Similmente:

D1:NOMEFILE.EST, D2:NOMEFILE.BAK

crea una copia di "back-up" (di sicurezza) di un file sullo stesso dischetto. Il nome del nuovo file deve essere diverso dall'originale: in questo caso è stata cambiata l'estensione.

D. DELETE FILE

NMF (RETURN) cancella uno o più file specificati. Se devono essere cancellati tutti i file del dischetto, si userà il simbolo jolly "*" in sostituzione dei nomi di file e delle estensioni.

E. RENAME FILE

Questa scelta permette di cambiare il nome di un particolare file ed aggiorna la Directory. Ad esempio:

D1:VECCHIO.NOM, NUOVO.NOM

Si possono usare simboli jolly per cambiare l'estensione di un gruppo di file.

F. LOCK FILE

NMF (RETURN) è una protezione software che impedisce la modifica o la cancellazione accidentale di un file, a meno che il dischetto non venga sottoposto ad una nuova formattazione. I file protetti sono preceduti, nella Directory, dal simbolo "*".

G. UNLOCK FILE

NMF (RETURN) elimina la protezione di un particolare file o, se usato con simboli jolly, di tutto un gruppo di file.

H. WRITE DOS FILES

Permette di memorizzare il DOS su di un dischetto appena formattato (o riformattato).

I. FORMAT DISK

Permette di formattare un dischetto.

J. DUPLICATE DISK

Vengono date le istruzioni per copiare il contenuto di un intero dischetto da un'unità all'altra oppure, se si dispone di una sola unità, su di un altro dischetto, trasferendo temporaneamente il contenuto del dischetto originale in memoria.

K. BINARY SAVE

NMF, SSSS, EEEE (RETURN) memorizza il contenuto di una specifica area di memoria, di solito un programma in codice macchina. SSSS è l'indirizzo di memoria iniziale e EEEE quello finale, entrambi scritti in notazione esadecimale a 4 cifre.

L. BINARY LOAD

NMF (RETURN) legge un file salvato con BINARY SAVE, trasferendolo all'interno delle locazioni dalle quali era stato prelevato per eseguire il salvataggio.

M. RUN AT ADDRESS

Alla comparsa del segnale di "prompt" (attesa di comando) inserire l'indirizzo di memoria, in esadecimale, dal quale ha inizio il programma letto con il BINARY LOAD. La pressione del tasto RETURN avvia il programma.

N. CREATE MEM.SAV

Permette di creare un file chiamato MEM.SAV. Quando viene richiamato il menu DOS, il DOS automaticamente salva il contenuto della memoria che verrebbe altrimenti cancellato dalla presenza del menu. Per ristabilire le condizioni precedenti, viene selezionata la scelta RUN CARTRIDGE.

O. DUPLICATE FILE

NMF (RETURN) permette di copiare un file da un disco ad un altro sulla stessa unità a dischetti. È consentito l'uso di simboli jolly.
Gli altri comandi che gestiscono i file di programma ed i file dati sono:

SAVE LOAD LIST ENTER RUN OPEN# CLOSE# PRINT# INPUT# NOTE# POINT# PUT# GET# STATUS# X10

FILE DI PROGRAMMA:

SAVE FSP

Permette di salvare un particolare programma su disco sotto forma di "token" (simboli in codice).

LOAD FSP

Carica un particolare programma in forma di "token" nella memoria utente a partire dalle locazioni con indirizzo più basso.

LIST FSP, LN1,LN2

Permette di immagazzinare un programma BASIC in formato ATASCII (la versione Atari dello ASCII standard). La specificazione FSP da sola permette di memorizzare l'intero programma. LN1 ed LN2 rappresentano i numeri di linea usati per indicare le linee iniziale e finale della porzione di programma che deve essere memorizzata. Usato assieme ad ENTER, questo comando permette di fondere 2 programmi.

ENTER FSP

permette di trasferire in memoria utente un file precedentemente immagazzinato con il comando LIST e lo fonde ad un altro programma già presente in memoria. Se nei 2 programmi ci sono numeri di linea uguali, le linee del programma trasferito in memoria per ultimo sostituiscono quelle già presenti.

RUN FSP

Trasferisce in memoria un programma espresso in "token" e lo invia automaticamente in esecuzione.

FILE DI DATI:

OPEN#

Questo comando controlla l'accesso a speciali canali chiamati Blocchi di Controllo di I/O (IOCB) e li collega, sotto particolari condizioni, ad un'appropriata periferica, in questo caso un'unità a dischetti ed un FSP, nel seguente modo:

OPEN# IOCB, AC1, AC2, FSP

dove IOCB indica il canale di I/O (1-5), AC1 è il codice ausilario 1 (che definisce il tipo di operazioni di I/O in conformità con una tabella riportata nel manuale DOS) e AC2 è sempre 0 per il dischetto. I seguenti comandi si riferiscono ai Blocchi di Controllo di I/O aperti (OPEN) nel modo esposto in precedenza.

CLOSE# IOCB

Permette di liberare un particolare IOCB dalle condizioni per lui selezionate in precedenza. Gli IOCB interessati dal comando CLOSE# divengono "inaccessibili".

PRINT#

Permette di inviare un dato numerico (x,y) o una stringa (A$) su uno specifico IOCB; ad esempio:

PRINT# IOCB,X,Y oppure PRINT# IOCB,A$

INPUT#

Permette di leggere un dato numerico o una stringa in arrivo su un particolare IOCB; ad esempio:

INPUT# IOCB, X,Y, oppure PRINT# IOCB,A$

NOTE#

Questo comando viene selezionato prima di immagazzinare un dato con PRINT#. Registra in un particolare record il numero del settore ed il numero dei byte che indicano dove verrà registrato sul disco il byte successivo; la lista risultante può essere immagazzinata sotto forma di tabella in un altro file e permetterà, tramite il comando POINT#, di accedere ai dati con procedura "random" (ad accesso casuale) e, se lo si desidera, ad un byte alla volta, per esempio:

NOTE IOCB, A, B

dove A indica il numero di settore (1-719) e B il numero del byte (1-124).

POINT#

Trasferisce nella memoria utente uno specifico byte, immagazzinato prima usando il comando NOTE#; ad esempio:

POINT# IOCB, A, B

PUT#

Invia un singolo byte su uno specifico IOCB. Per esempio:

PUT# IOCB, N

dove N può assumere valori compresi tra 1 e 255.

GET#

Legge un singolo byte immagazzinato con il comando PUT#. Ad esempio:

GET# IOCB, N

STATUS# IOCB, ERRORE

Assegna una variabile, in questo caso ERRORE, al corrente codice d'errore per l'ultima operazione di I/O eseguita su quel particolare IOCB. Il numero può quindi essere confrontato con la tabella degli errori del manuale DOS dell'Atari.

X10 CN,#IOCB, AC1, AC2, FSB

Permette di duplicare alcune funzioni del menu DOS usando il Numero di Comando (CN) che realizza il collegamento con la funzione richiesta. Il manuale DOS dell'Atari presenta una lista dei Numeri di Comando e delle funzioni ad essi associate.

PREZZO E PROGRESSO

A partire dal 1973 il costo di un computer è diminuito di 500 volte per ogni bit di memoria. Se ciò fosse accaduto per una Rolls Royce, ad esempio, il suo prezzo attuale sarebbe di circa 2000 lire e percorrerebbe 40.000 km con un litro di benzina!

La logica del programmatore

Le operazioni logiche AND, OR sono usate anche nella programmazione

I due operatori logici AND ed OR sono molto utili sia nel codice macchina che in BASIC. Essi vengono comunemente usati per collegare due o più condizioni all'interno di un'unica istruzione condizionale. Proviamo, ad esempio, a calcolare quale sarà il risultato del seguente programma BASIC:

```
10 FOR L = 1 TO 5
20 FOR J = 1 TO 5
30 IF L = 3 AND J = 2 THEN PRINT L,J
40 NEXT L
50 NEXT J
60 END
```

Il programma eseguirà regolarmente i due cicli "nidificati", ma visualizzerà i valori di L e J solo se L=3 e J=2. Quindi il risultato che apparirà sullo schermo sarà il seguente:

```
3  2
```

La funzione OR può essere usata allo stesso modo. Se sostituiamo la linea 30 con la seguente:

```
30 IF L=3 AND J=2 OR J=4 THEN PRINT L,J
```

l'output sul video sarà il seguente:

```
1  4
2  4
3  2
3  4
4  4
5  4
```

Il BASIC esegue l'operazione AND prima di quella OR. L e J verranno visualizzati se L=3 e J=2, oppure se J=4. L'ordine di precedenza può essere cambiato usando delle parentesi. Quale sarà il risultato del programma se la linea 30 viene sostituita con la seguente?

```
30 IF L=3 AND (J=2 OR J=4) THEN PRINT L,J
```

Isolamento dei bit in un registro

Molti microcomputer fanno uso di particolari "registri" (locazioni di memoria) per controllare varie funzioni della macchina. Ciascun bit all'interno di questi registri può controllare un differente aspetto di una particolare operazione. Nel Commodore 64, ad esempio, c'è un registro ad 8 bit che controlla l'accensione e lo spegnimento degli sprite. Ciascun bit del registro è collegato ad uno degli 8 sprite disponibili. Se un bit viene posto ad 1, il corrispondente sprite appare sullo schermo. Viceversa, se il bit è posto a 0, lo sprite non appare sullo schermo. In BASIC è possibile ottenere una qualsiasi combinazione di sprite, semplicemente depositando nell'opportuno registro, con il comando POKE, il numero decimale equivalente alla sequenza degli otto bit. Questa procedura, tuttavia, non tiene conto del contenuto del registro, precedente all'invio del comando POKE, e può causare la scomparsa accidentale di uno o più sprite prima visibili. Per risolvere questo problema è necessario adottare una tecnica che consenta di isolare dagli altri i bit da modificare. Per chiarire il funzionamento di questa procedura, supponiamo che, inizialmente, siano accesi solo gli sprite 0, 1, 5, 6. Il registro di controllo presenterà questa configurazione:

N. degli Sprite	7	6	5	4	3	2	1	0
Bit corrispondenti	0	1	1	0	0	0	1	1

Accendiamo ora lo sprite 4, leggendo il contenuto del registro con il comando PEEK, eseguendo un OR del numero letto dal registro con il valore 16 (00010000 in binario) e depositando nel registro, con il comando POKE, il risultato ottenuto.

N. degli Sprite	7	6	5	4	3	2	1	0
Byte originale	0	1	1	0	0	0	1	1
OR con	0	0	0	1	0	0	0	0
Nuovo Byte	0	1	1	1	0	0	1	1

Usando il formato POKE reg, (PEEK(reg) OR 16), possiamo quindi modificare il bit 4 del registro, senza danneggiare gli altri. Analogamente, per spegnere lo sprite 4, occorre nuovamente intervenire con il comando PEEK, ma operando questa volta un AND rispetto a 293 (11101111 in binario).

N. degli Sprite	7	6	5	4	3	2	1	0
Nuovo Byte	0	1	1	1	0	0	1	1
AND con	1	1	1	0	1	1	1	1
Byte originale	0	1	1	0	0	0	1	1

Si noti che il numero 239 si calcola facilmente sottraendo 16 da 255. Con la sequenza POKE reg, (PEEK(reg) AND 239) abbiamo quindi riportato il registro nella sua condizione iniziale. Queste tecniche vengono utilizzate nella programmazione in codice macchina dove le modifiche ai registri di controllo sono una componente fondamentale dei programmi.

Risposte all'Esercizio 3

1)

a) $A.(\overline{A}+\overline{B})$
$= A.\overline{A}+A.\overline{B}$ (legge distributiva)
$= A.\overline{B}$ ($A.\overline{A} = 0$)

b) $X+Y.(X+Y)+X.(\overline{X}+Y)$
$= X+Y+X.(\overline{X}+Y)$ (relazione 5)
$= X+Y+X.Y$ (relazione 6)
$= X+Y$ (assorbimento)

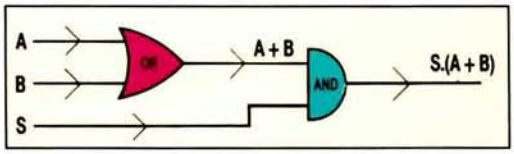
c) $P.Q+\overline{P}.Q+\overline{PQ}$
$= P.Q+\overline{P}.(Q+\overline{Q})$ (legge distributiva)
$= P.Q+\overline{P}$ ($Q+\overline{Q} = 1$)
$= \overline{P}+Q$ (duale relazione 6)

d) $\overline{\overline{X}+Y.\overline{Z}+\overline{Z}.Y}$ (de Morgan)
$= \overline{\overline{X}}.\overline{Y.\overline{Z}}.\overline{\overline{Z}.Y}$ (legge distributiva)
$= X.Y.Z.(Z+\overline{Y})$ ($\overline{\overline{X}} = X$, de Morgan)
$= X.Y.Z.Z+X.Y.Z.\overline{Y}$ (legge distributiva)
$= X.Y.Z+0$ ($Z.Z = Z$; $Y.\overline{Y} = 0$)

2) La tavola della verità per il sistema d'allarme è la seguente:

INGRESSI			USCITA
A	B	S	Allarme
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Allarme $= \overline{A}.B.S+A.\overline{B}.S+A.B.S$
$= \overline{A}.B.S+A.S.(\overline{B}+B)$ (legge distributiva)
$= \overline{A}.B.S.+A.S$ ($B+B = 1$)
$= S.(A+\overline{A}.B)$ (legge distributiva)
$= S.(A+B)$ (duale relazione 6)



3) Se i tre interruttori sono x, y, z e la luce della stanza è P, allora la tavola della verità è:

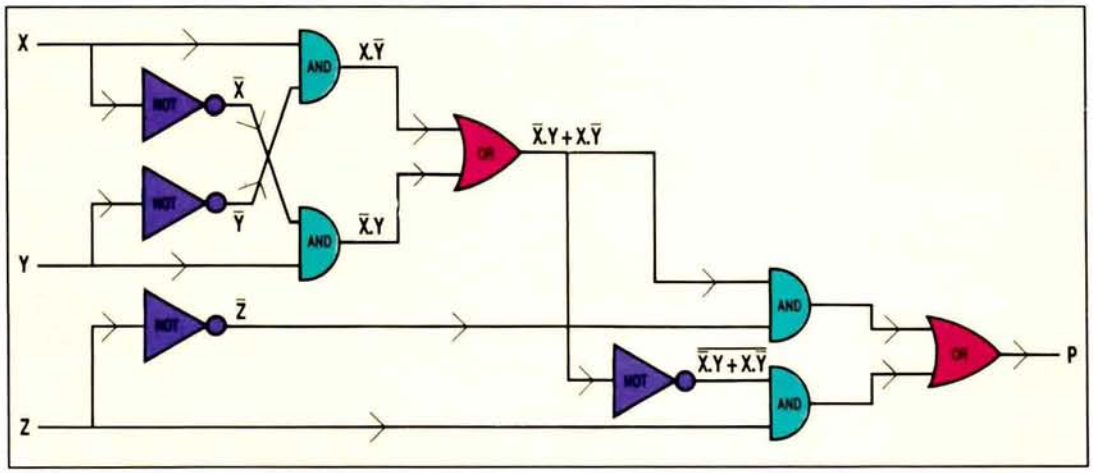
INGRESSI			USCITA
X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$P = \overline{X}.\overline{Y}.Z.+\overline{X}.Y.\overline{Z}.+X.\overline{Y}.\overline{Z}+X.Y.Z$
$= Z.(\overline{X}.\overline{Y}+X.Y)+\overline{Z}.(\overline{X}.Y+X.\overline{Y})$ (legge distributiva)
$= Z.(\overline{\overline{X}.Y+X.\overline{Y}})+\overline{Z}.(\overline{X}.Y+X.\overline{Y})$ (de Morgan)

4) La tavola della verità mostra che la domanda «Dici la verità?» ci è di scarsa utilità, poiché tanto un'asserzione vera che una falsa daranno lo stesso risultato. La tavola della verità ha la stessa forma della funzione $X.Y+X.Y$ che, semplificata, dà Y. Cioè, la risposta dipende da un'unica variabile, non due, ossia non c'è alcuna differenza tra asserzioni vere e false. Tuttavia, se poniamo la domanda «I cani hanno le ali?», la tavola della verità sarà la seguente:

		RISPOSTA POSSIBILE	
		SI	NO
POSSIBILE IDENTITÀ DI CHI RISPONDE	BUGIARDO	1	0
	SINCERO	0	1

Questa è la tavola della verità per la funzione $X.\overline{Y}+\overline{X}.Y$, ed equivale anche ad una tavola per OR-Esclusivo. In questo caso è possibile identificare chi risponde il vero.



Dissaldare

Impariamo a rimuovere una ROM ed a montare, per essa, uno zoccolo di tipo speciale

Nella scorsa lezione abbiamo visto come si fanno le saldature, adesso vediamo come dissaldare componenti da un circuito senza provocare danni. Se operiamo soltanto su cavetti e spinotti le precauzioni non sono molte: alla peggio, si può tagliare un pezzo di cavo, gettare lo spinotto danneggiato e ricominciare con un componente nuovo. Ma cosa accade quando occorre sostituire un chip o cambiarne la posizione? Anche i semplici transistor possiedono tre piedini o terminali: per rimuovere un transistor da un circuito stampato, è necessario scaldare contemporaneamente i tre piedini per sciogliere i punti di saldatura e quindi estrarre il transistor. In alternativa, si può scaldare e liberare un piedino alla volta. Ma se l'intervento contemporaneo su tre piedini crea preoccupazioni, come comportarsi di fronte ai 40 piedini di un comune microprocessore ad 8 bit?

Sostituzione di un chip

Invece di sostituire un chip a caso o di aggiungere chip RAM per espandere le capacità di memoria di un computer, progettiamo un intervento leggermente più ambizioso, su di uno ZX81: dotarlo di uno zoccolo per ROM, esterno al contenitore, sul quale inserire la ROM del BASIC o altre ROM a nostro piacimento.
Si tratta, in sostanza, di rimuovere la ROM interna e di portare i collegamenti all'esterno, mediante un cavo piatto. Per questo esperimento è stato scelto lo ZX81, perché il linguaggio "residente" può essere sostituito con il FORTH, pure contenuto in un chip di ROM. Oltre al nuovo linguaggio, su questa ROM è incorporato anche un sistema operativo multiuso, che permette a più programmi di funzionare simultaneamente ed in modo indipendente l'uno dall'altro. È questo un ottimo risultato per una macchina così piccola, anche se implica l'occupazione di almeno 2 Kbyte di RAM e sarà, quindi, necessario inserire un modulo aggiuntivo di memoria RAM.
Questo piccolo progetto permette di esercitarci nella manipolazione dei componenti di un computer e dimostra quanto sia importante lavorare con ordine e precisione. Vedremo in seguito come eseguire una verifica del lavoro fatto usando un multimetro. Questo apparecchio misura la corrente, il voltaggio, la resistenza ed è uno strumento indispensabile per il controllo e la verifica di circuiti e componenti.

IL MATERIALE OCCORRENTE

Se dobbiamo sostituire un chip del nostro computer o semplicemente cambiarne la posizione, oppure se dobbiamo aggiungerne uno nuovo, il montaggio di uno zoccolo su cui sia possibile innestarlo costituisce una buona esercitazione. I chip montati su zoccoli si possono rapidamente sostituire. Se le sostituzioni avvengono con una certa frequenza, conviene utilizzare degli zoccoli sofisticati, detti "a pressione d'innesto zero" (ZIF), grazie ai quali si riduce al minimo il rischio di danneggiare i piedini. Oltre agli utensili già descritti (vedere a pag. 524), occorre acquistare una bobina di "calza" di rame per asportare lo stagno delle saldature, oppure un dissaldatore ad aspirazione. L'esercitazione che abbiamo scelto implica la rimozione della ROM BASIC di uno ZX81 e la sua sostituzione con uno zoccolo standard da cui parte un cavetto a nastro che viene collegato ad un supporto ZIF posto al di fuori della macchina. Lo scopo dell'intervento è sostituire il BASIC dello ZX81 con una delle ROM FORTH multiuso (progettate da David Husband), lasciando però la possibilità di tornare al BASIC in un momento successivo. Per eseguire questo lavoro sono inoltre necessari i seguenti materiali: una basetta di Veroboard (o di circuito "prestampato"), un lungo cavetto a nastro con 28 linee di connessione (entrambi possono essere acquistati presso un qualsiasi rivenditore di materiale elettronico) ed una lamina di polistirolo espanso, da usare come supporto isolante

Istallazione della ROM

Per prima cosa aprire la custodia dello ZX81 ed individuare la ROM BASIC. La custodia è tenuta assieme da 5 viti autofilettanti, tre delle quali si trovano sotto i piedini di gomma della macchina. Il piedino che non nasconde una vite si trova vicino agli spinotti MIC e EAR. Staccare con attenzione i piedini di gomma e svitare le tre viti a croce poste sotto di essi, poi le altre due visibili. Sollevare la parte inferiore della custodia per scoprire la faccia inferiore della scheda dei circuiti (PCB). Questa può essere rimossa svitando le tre viti a croce visibili: due si trovano vicino al connettore a lama, la terza presso il dissipatore (la lamina d'alluminio posta al di sotto della tastiera). Girando la scheda vediamo che il chip ROM si trova in alto e leggermente a sinistra del connettore per la tastiera

STRUMENTI PER DISSALDARE

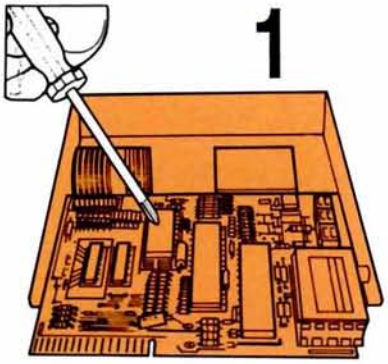
I principali strumenti per rimuovere una saldatura sono due: il meno costoso è una sottilissima calza di rame, al cui interno si trova un'"anima" di sostanza dissaldante. Funziona sulla base del principio fisico della "capillarità". La saldatura viene disciolta dal calore e risale all'interno della calza. Sono disponibili calze di rame con diverse sezioni a seconda delle dimensioni della saldatura da asportare. Via via, la parte usata deve essere eliminata e non può venire utilizzata una seconda volta. Il secondo strumento dà risultati molto più soddisfacenti: assomiglia ad una piccola pompa per biciclette, ma funziona nel modo opposto, cioè aspira aria dal beccuccio, invece di emetterne. Il suo costo è superiore a quello della calza di rame, ma, oltre ad essere molto più funzionale, la sua durata è praticamente illimitata. Come vedremo anche in seguito, questi strumenti sono indispensabili per eseguire interventi di rimozione

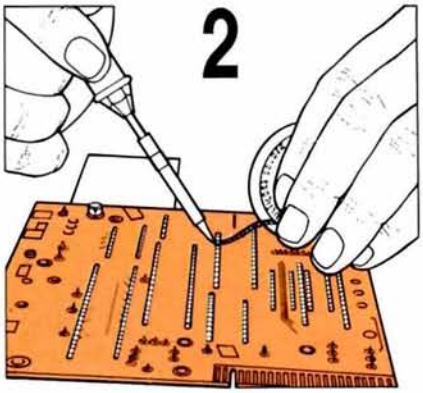


La dissaldatura

Riscaldare la saldatura finché la lega metallica non si scioglie, avvicinare la piccola pompa aspirante e premere il pulsante di richiamo. Il metallo sciolto verrà aspirato nella camera interna della pompa







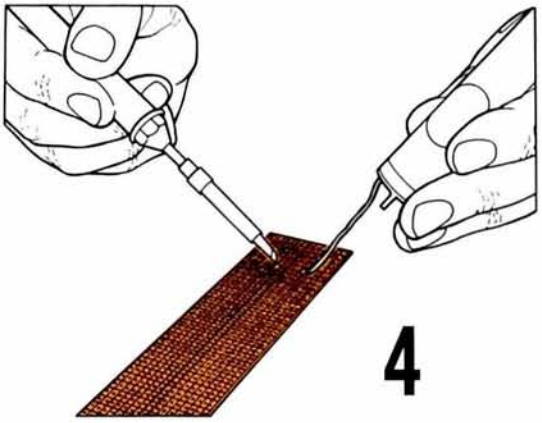
Estrazione della ROM

Il segreto per eseguire una buona dissaldatura, come per la saldatura, è applicare sufficiente calore. Premere la calza di rame sulla connessione con la punta del saldatore fino ad ottenere la liquefazione del "fondente": probabilmente scaturirà un sottile filo di fumo blu. A questo punto la calza rimuove per capillarità la saldatura. Tagliare infine la piccola porzione di calza saturata



Cavo a nastro

Sono disponibili cavi a nastro di varie dimensioni. In questa operazione abbiamo bisogno di un cavo a nastro con 28 linee di connessione (cioè 28 cavetti indipendenti) o di due cavi a 14 linee. Togliere 1 cm. di rivestimento a ciascuno dei 28 cavetti e stagnare i conduttori interni. Iniziare il lavoro dall'angolo del chip contrassegnato da una piccola tacca semicircolare e procedere alla saldatura tenendo il cavo a nastro rivolto verso il basso da entrambe le parti. Tutti i chip presentano tacche o punti che indicano in quale ordine devono essere allineati

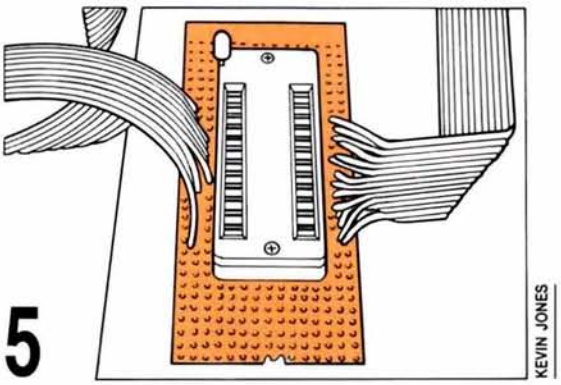


Collegamento

Tagliare a misura le due parti di cavo a nastro, togliere il rivestimento e stagnare i 28 conduttori interni. Assicurarsi che i conduttori provenienti dalla estremità marcata del chip vadano all'estremità corrispondente dello zoccolo, quindi saldarli sul Veroboard. Fare attenzione a non incrociare tra loro i conduttori, alterando così le rispettive posizioni

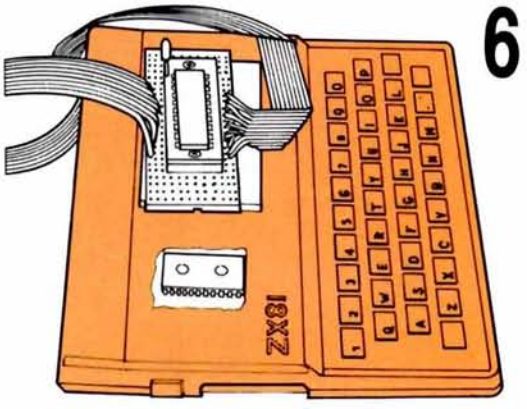
Veroboard e Veroblock

Questi due particolari prodotti sono stati progettati per sostituire schede a circuiti stampati. Il Veroboard è una lamina di plastica rigida che presenta una configurazione standard di fori disposti a matrice; su ogni riga della matrice è disposta una sottile lamina di rame. Il contatto tra due punti di connessione può essere interrotto tagliando la lamina di rame, mentre è possibile creare un collegamento tra lamine poste su righe diverse. Il Veroblock è una versione più sofisticata dello stesso oggetto; viene incapsulato (sigillato) in un blocco di plastica che permette di realizzare connessioni senza saldature



Montaggio dello zoccolo

Il Veroboard è il materiale che meglio si adatta a questo tipo di lavoro. Presenta 50 punti di contatto raccolti in circa 25 cm. e può essere spezzato con le dita per ottenere la dimensione voluta. Mettere lo zoccolo nel centro della piastrina isolata e saldare un piedino d'angolo per mantenerlo in posizione, quindi eseguire il lavoro di saldatura prima su un lato e poi sull'altro, facendo attenzione che le saldature non presentino sbavature



Rifiniture

Dopo aver eseguito tutte le saldature, controllarle una per una, sia quelle sulla scheda dei circuiti del computer che le nuove eseguite sulla scheda figlia, per verificare che non ci siano contatti accidentali. Quando si effettuano piccole saldature su connessioni vicine tra loro, spesso si creano punti di contatto o sbavature che causano cortocircuiti. Controllare quindi tutti i collegamenti con una lente d'ingrandimento e, nei casi dubbi, raschiare lo spazio compreso tra due saldature sospette con un punteruolo o un altro strumento di metallo appuntito

ATTENZIONE!

La garanzia di un computer non ancora scaduta può essere annullata se la custodia dell'apparecchio viene aperta da persone diverse dal fabbricante o da un suo agente delegato

Metodi di stampa

Prima o poi sorge la necessità di riprodurre su carta le informazioni elaborate: ecco alcune possibilità

Le stampanti sono apparecchi di primaria importanza per chi desideri produrre copie su carta (dette anche "hard copy") di listati di programma o di parti d'un testo. Purtroppo una buona stampante a matrice d'aghi può costare addirittura il doppio o il triplo di un home computer, per non parlare delle stampanti "a margherita", che raggiungono prezzi astronomici. Esistono, tuttavia, apparecchi che adoperano metodi alternativi a quello, più comune, chiamato "ad impatto".

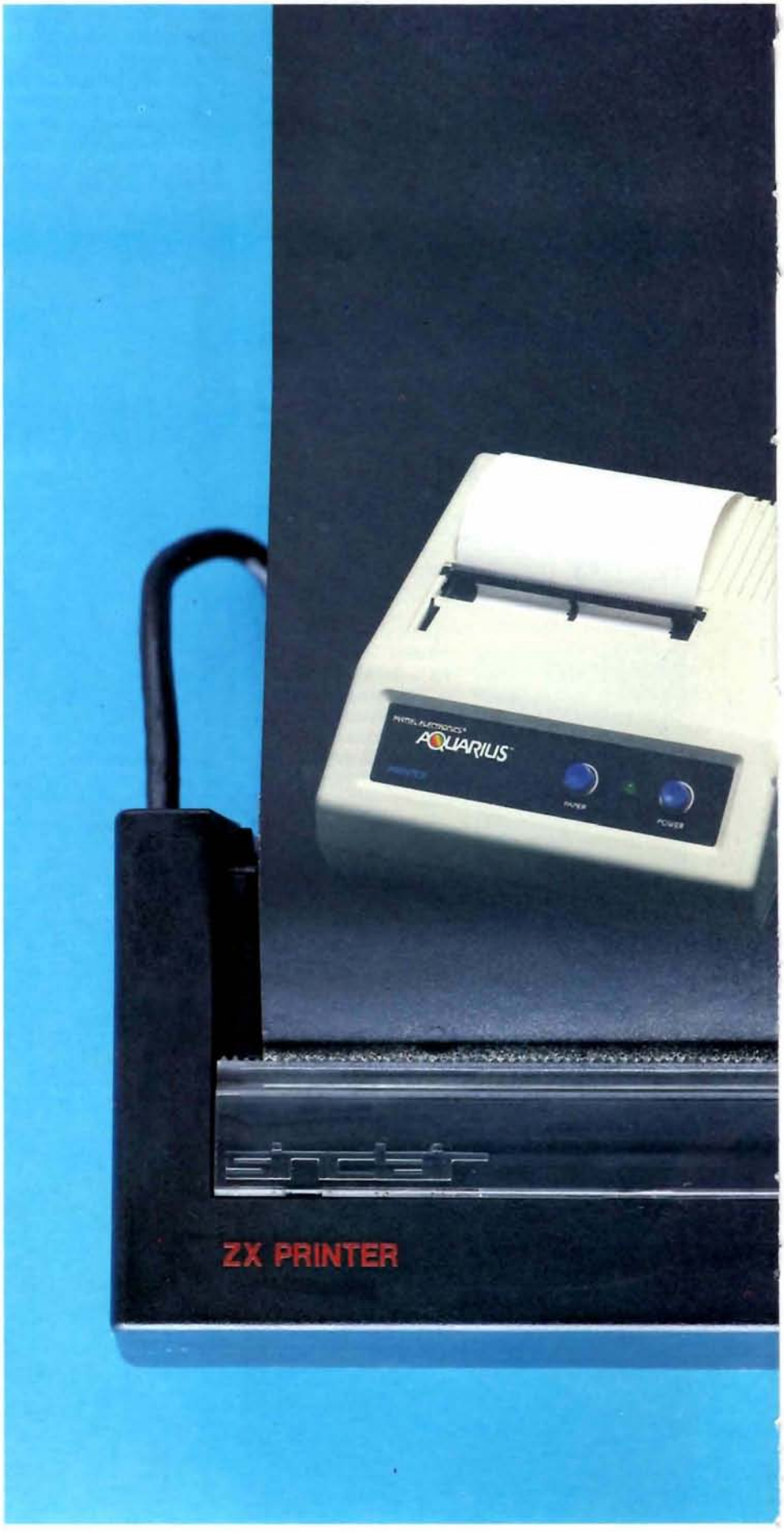
Il nome di questo metodo deriva dal fatto che viene fatto uso di un nastro e di una testina a matrice d'aghi (o di parti metalliche sagomate) per stampare ad impatto, appunto, i caratteri sulla carta. Le prime stampanti di tipo diverso da quelle ad impatto furono applicate ai registratori di cassa ed ai terminali portatili, quali ad esempio la serie Silent 700 della Texas, nei quali viene usato il calore, per produrre immagini su di un particolare tipo di carta sensibile alle variazioni termiche.

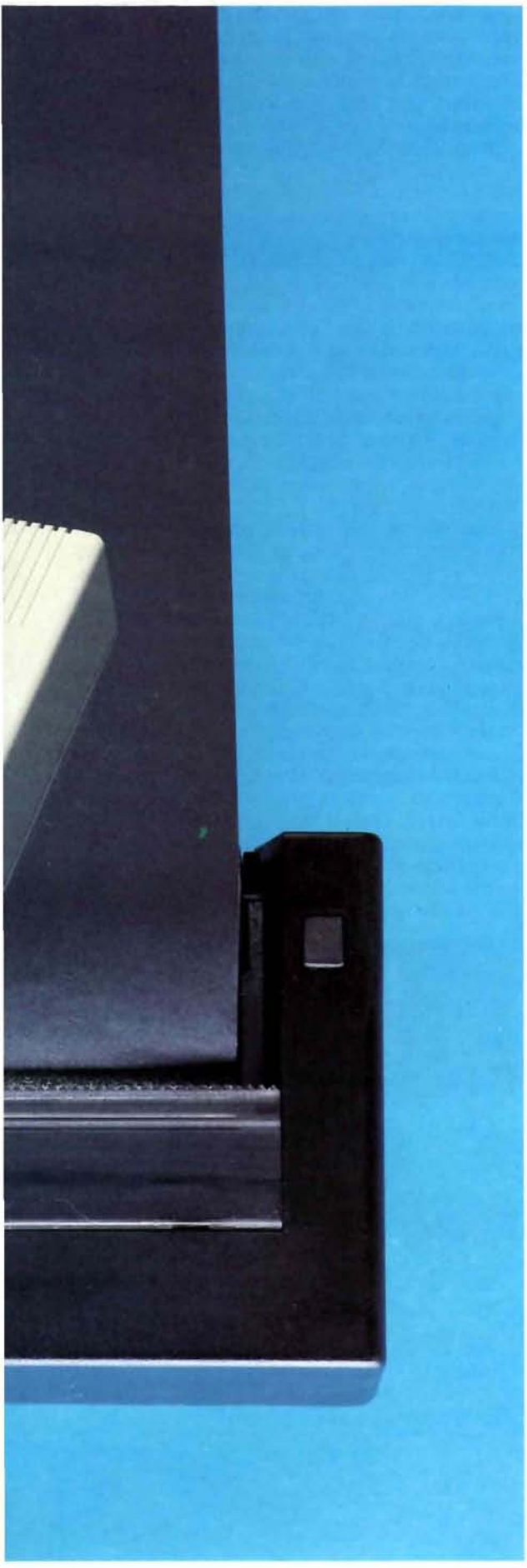
Così come le stampanti a matrice d'aghi presentano una serie verticale di aghi posti su una testina che si muove orizzontalmente sulla pagina, le stampanti termiche possiedono una serie verticale di elementi, soggetti a riscaldamento. Quando deve essere stampato un punto, il corrispondente elemento della testina termica viene rapidamente riscaldato e la carta su cui esso poggia cambia colore. Il contrasto tra l'immagine stampata e lo sfondo è discreto, mentre la qualità dei caratteri di stampa lascia molto a desiderare. La caratteristica più significativa di questi apparecchi, è la quasi totale silenziosità.

Oltre che per i terminali della Texas, le stampanti termiche sono state adottate anche dalla Apple, che ha prodotto una piccola stampante chiamata Silentype, ed, in passato, dalla Mattel, che vendeva un apparecchio, con caratteristiche simili al precedente, da abbinare all'home computer Aquarius.

Viceversa, la meccanica delle stampanti elettrostatiche è relativamente rumorosa e, se si esclude il primo modello della Centronics chiamato Microprinter P1, questi apparecchi non hanno ottenuto un gran successo fra il pubblico, finché Sir Clive Sinclair non ha adottato questo sistema per la stampante ZX, progettata per lo ZX81 e lo Spectrum.

Le stampanti elettrostatiche sono caratterizzate da una testina di stampa a cavo singolo,



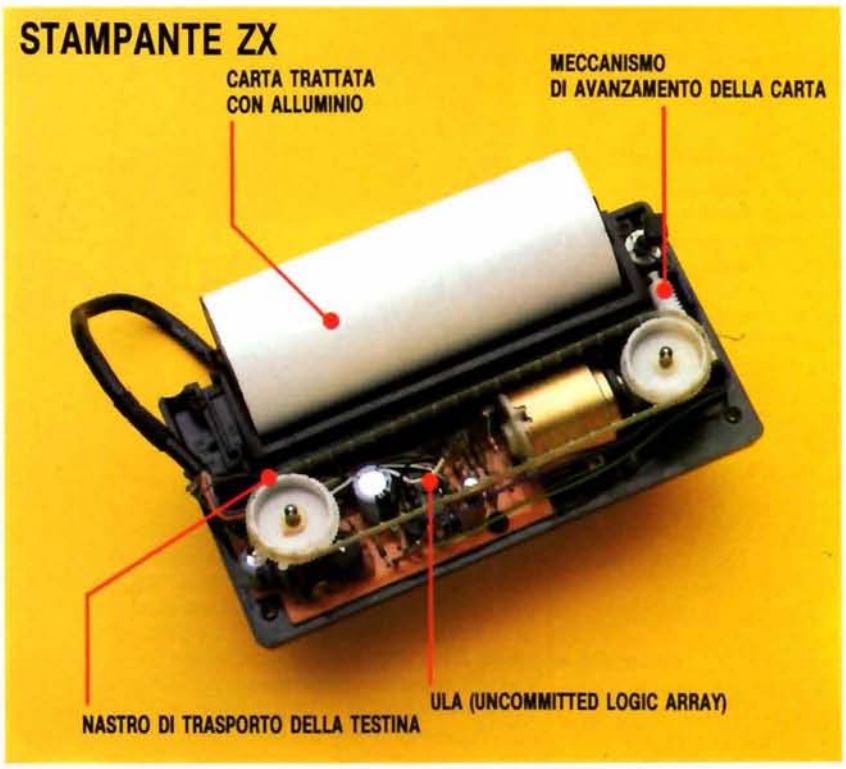


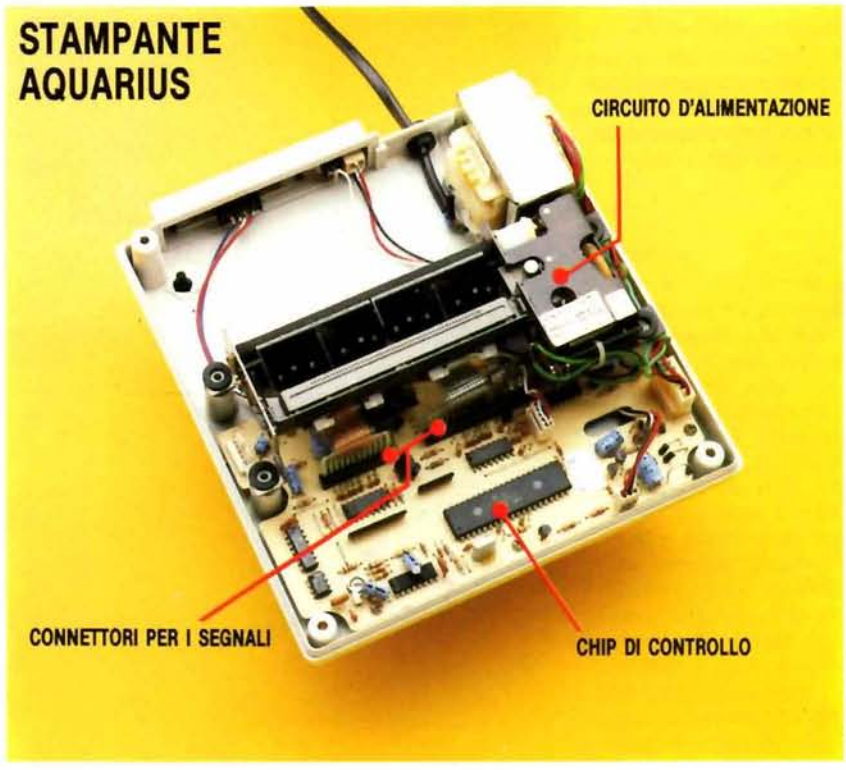
Stampante ZX

Presentata come accessorio per lo ZX81, la Stampante ZX è anche compatibile con lo Spectrum. È una periferica a basso costo che permette agli utenti di home computer di stampare testi o listati di programma. Il prezzo è assai interessante, ma l'uso di carta elettrostatica rappresenta un notevole svantaggio

Stampante Aquarius

L'Aquarius ha adottato un metodo di stampa termico, ma sono oggi disponibili anche economiche stampanti a quattro colori che utilizzano penne a sfera. La stampante Aquarius, come la ZX, presenta il difetto di adoperare carta speciale





CHRIS STEVENS

che si muove su una speciale carta a doppia lamina. I caratteri di stampa sono composti da una serie di punti, prodotti da scintille, che bruciano una porzione puntiforme della lamina metallica di rivestimento e mettono allo scoperto la carta nera sottostante. Sinclair ha perfezionato questo sistema introducendo una seconda testina posta su una cinghia di trascinamento ad anello. Occorrevano tuttavia ben otto passaggi della testina per stampare una riga di caratteri completa. Ma, poiché la stampante ZX stampa solo 32 caratteri per ogni riga, la velocità risulta accettabile.

Il principale inconveniente di questi due tipi di stampante è che entrambi usano carta speciale piuttosto costosa e disponibile solo in rotoli, il che provoca talvolta difficoltà di trascinamento. Per ottenere immagini di buona qualità con le stampanti termiche è fondamentale acquistare il giusto tipo di carta: in caso contrario la stampa sbiadisce col passare del tempo o con l'esposizione al calore. La carta elettrostatica è ancora più delicata e, se viene toccata con mani umide, la patinatura che la riveste si scioglie e i caratteri stampati appaiono offuscati e tendono a sbiadire. In entrambi i casi è consigliabile fotocopiare le stampe eseguite e, fortunatamente, con la carta argentata si ottengono fotocopie di ottima qualità.

Nonostante questi difetti, entrambi i tipi di stampante permettono ai fabbricanti di computer di piccole dimensioni di dotare le proprie macchine di vari sistemi di stampa. Il metodo a matrice d'aghi permette di stampare copie dirette delle immagini presenti sul video: in questo modo è possibile riprodurre su carta testi e grafica senza ulteriori spese e la qualità relativamente bassa dell'immagine finale non crea grossi problemi.

Recentemente, tuttavia, il primato di questi sistemi di stampa è stato insidiato dai plotter a penne di 4 colori prodotti dalla Tandy e dalla Sharp. Questi possono tracciare un'eccellente gamma di caratteri a pieno formato e un'ottima grafica su normale carta, praticamente allo stesso costo delle stampanti termiche ed elettrostatiche.



Il grande fratello

Tra le realizzazioni di stampanti termiche, l'apparecchio meglio concepito e più elegante è forse il Brother EP-22. Per una cifra molto ragionevole si acquista non solo una stampante termica a 75 colonne, ma una vera e propria macchina da scrivere portatile. Progettato per funzionare sia con carta termica che con nastro inchiostrato, come una normale stampante "a impatto", questo apparecchio può memorizzare fino a 2.000 caratteri. Non esistono funzioni di word processing vere e proprie, tuttavia si ha la possibilità di correggere gli ultimi 16 caratteri battuti, visualizzati da un display a cristalli liquidi (LDC). La qualità di stampa è paragonabile a quella che si ottiene con le economiche stampanti ad aghi. Infatti, non esistono veri e propri caratteri "discendenti" ma, come stampante portatile associata, magari, a computer come l'Epson HX-20 ed il Modello 100 della Tandy, si possono ottenere risultati molto soddisfacenti



IAN McKINNELL

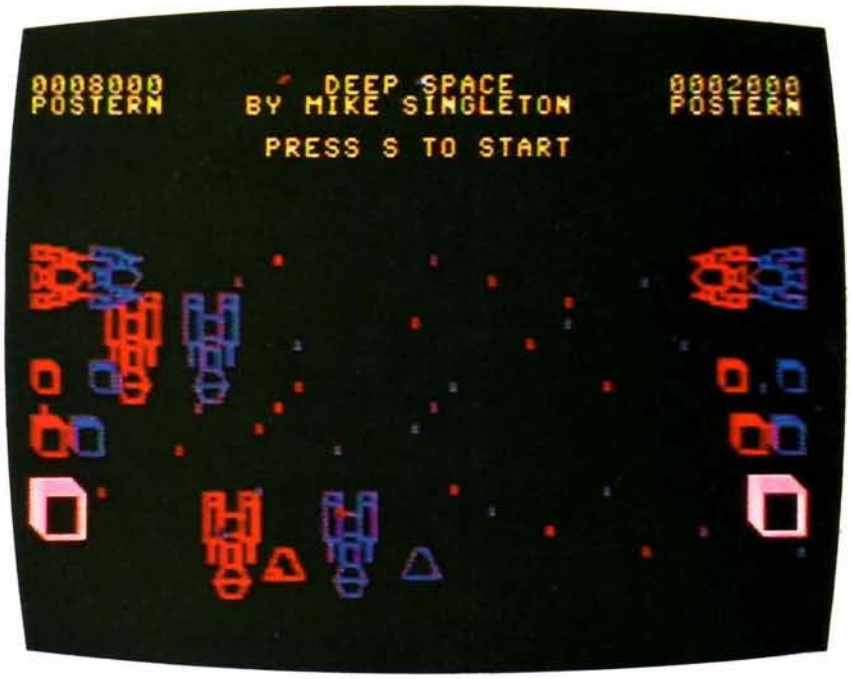
Stampante termica Tandy

Più flessibile delle due stampanti di basso costo già descritte, la stampante termica TP-10 della Tandy è compatibile con tutti i micro della gamma Tandy

Software in vetrina

Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Con l'intento di fornire un ampio panorama sulla vastissima produzione di giochi per home computer, presentiamo altri due programmi. Appartengono ambedue al genere dei giochi spaziali e sono stati scritti per il Commodore 64 ed il Vic-20.



Spazio profondo 3D per il Commodore 64

In questo gioco, il cui svolgimento non esce molto dai normali canoni dei videogame spaziali, la principale caratteristica è il tentativo di creare un effetto tridimensionale, ricorrendo all'uso di speciali occhiali colorati, forniti assieme alla cassetta. Gli spostamenti della base di comando, che il giocatore deve difendere, sembrano avvenire, oltre che lateralmente, anche "in avvicinamento" ed "in allontanamento" rispetto all'osservatore.



Cosmic Fire-Birds per il Vic-20

Prodotto dalla Solar Software, questo gioco assomiglia al *Phoenix*, diffusissimo nelle sale giochi e, per certi versi, è uno dei progenitori di tutti i videogiochi. Al comando di una potente astronave, ci troviamo a combattere contro alieni chiamati, appunto, Cosmic Fire-Birds (uccelli di fuoco cosmici). La dinamicità dei loro movimenti è tale da impegnare anche i più provetti giocatori: giravolte, scarti, evoluzioni ed improvvise "picchiate". Decisamente un gioco d'azione, che sfrutta a fondo le capacità grafiche del Vic-20.

Word processing sul BBC!

Due utilissimi programmi di videoscrittura di facile impiego, per il Micro BBC

Uno dei word processor di maggior successo, per il Micro della BBC, è il Wordwise, venduto su chip, da inserire nello zoccolo ROM del computer. Per lanciare il programma, dopo aver innestato la ROM, basta digitare sulla tastiera le lettere *W. Il Wordwise comprende il chip di ROM, le istruzioni per l'istallazione, un manuale ed una cassetta che contiene il testo del manuale in forma visualizzabile sullo schermo del computer.

Il Wordwise è di facile impiego anche per un principiante, senza dover consultare continuamente il manuale. Per cominciare a lavorare basta premere il tasto ESCAPE, che avvia il processo di redazione di testi (Edit) ed iniziare la battitura. Il Wordwise sfrutta i tasti di funzione del BBC per muovere o copiare parti del testo.

Avendo acquisito una certa familiarità con le funzioni principali del programma, si possono esplorare le caratteristiche più sofisticate di questo sistema di videoscrittura. Per esempio: saltare all'inizio o alla fine di una linea o di una pagina usando i tasti Shift e di controllo del cursore oppure sovrapporre una parola ad una già esistente, con l'aiuto di un tasto di funzione.

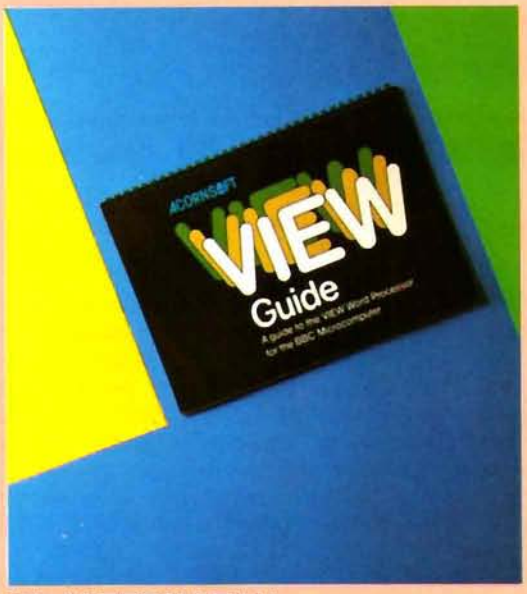
Il formato del testo, sullo schermo, è di 40 battute, ed i caratteri sono comodamente leggibili anche usando un normale televisore come monitor. L'aspetto finale del testo può essere esaminato sullo schermo, prima della stampa, anche in un formato ad 80 colonne, usando un particolare tasto di funzione. Questa utile possibilità consente di apportare eventuali modifiche al formato del testo, prima di passare alla stampa definitiva. In questa fase intermedia, il manuale può tornare utile per esaminare l'effetto dei vari comandi di controllo. Ad esempio, il comando LM, seguito da un numero, seleziona la posizione del margine sinistro; LL definisce la lunghezza della linea, IN la rientranza del margine all'inizio di un paragrafo, PL la lunghezza della pagina e così via. Questi comandi si ricordano molto facilmente e vengono

Un sistema ideale

Un buon sistema di word processing deve prevedere: una comoda tastiera, un monitor (o televisore), una serie di dischetti, un'unità a dischetti per l'archiviazione, una stampante ed un buon rifornimento di carta per la stampa



KEVIN JONES



LIZ HEANEY

IN ALTERNATIVA

Il View, prodotto dalla Acorn per il Micro della BBC, è un altro word processor venduto su di un chip di ROM. Offre il vantaggio di partire automaticamente all'accensione della macchina, oppure può essere richiamato mediante un comando del BASIC.
Il View sfrutta appieno le caratteristiche del generatore video BBC e può presentare i testi con un formato a 76, 64, 34 o 16 colonne, adattando automaticamente la dimensione dei caratteri. Questo effetto viene prodotto "amplificando" l'intera immagine video: lo schermo si trasforma quindi in una finestra che si sposta su uno schermo virtuale di dimensioni molto maggiori. Il comando Mode, che produce questi cambiamenti, funziona a livello comandi del View, con la stessa procedura usata per l'accesso alle operazioni su dischetto, a quelle di stampa ed a quelle per la ricerca e la sostituzione di caratteri o parole.
Quest'ultima operazione, oltre alla normale funzione di ricerca di una stringa, esegue un'operazione che il manuale chiama "ricerca selvaggia". Ovvero, la ricerca segue la normale procedura, ma all'interno della stringa un punto interrogativo sostituisce i caratteri dubbi. Quindi, se la parola da cercare è "ca?", la procedura di ricerca individuerà tutte le parole di 4 lettere che cominciano per "ca..." (cane, care, case, ecc...). Naturalmente bisogna fare molta attenzione quando si definisce la stringa per una ricerca selvaggia, perché i caratteri ambigui possono avviare un processo di sostituzione troppo generico. Un altro comando particolarmente utile (specie per quei giornalisti pagati in base alla lunghezza degli articoli) è quello che permette di contare le parole che compongono un testo, una funzione non sempre presente sui word processor.
L'altro protocollo operativo del View, chiamato Modo Screen (Video), è suddiviso, a sua volta, in due gruppi di comandi: i comandi "immediati", che controllano l'inserzione e la cancellazione dei caratteri, il movimento del cursore e di parti del testo e le altre richieste "on-off"; ed i comandi di "memorizzazione", che servono per tener conto dei parametri selezionati nella creazione e nella redazione di un documento, per l'impaginazione, per l'intestazione ed altre procedure simili. Nel Modo Video, il View utilizza al meglio i 10 tasti di funzione del Micro BBC, ciascuno dei quali presenta tre diversi usi a seconda che venga usato da solo oppure insieme coi tasti Shift o Control.

abilitati premendo particolari tasti di funzione, ma divengono invisibili sia in fase di visualizzazione del testo definitivo che nella stampa.
Se nessuno dei comandi a disposizione viene selezionato, il Wordwise assume un formato di impaginazione standard: 70 caratteri per ogni linea, 66 linee per ciascuna pagina e 5 spazi per il margine sinistro. È quindi possibile produrre un documento ben impaginato ancor prima di conoscere i comandi di formattazione. Inoltre si possono inserire, come se si trattasse di comandi, anche alcuni codici di controllo, che permettono di selezionare il formato dei caratteri (corsivo, espanso, ecc.) della stampante, purché quest'ultima, ovviamente, offra tali possibilità di scelta. Vedremo più oltre come le chiavi di funzione possano essere usate per facilitare l'inserimento di questi codici di controllo.
Il Wordwise dispone di un menu mediante il quale è possibile memorizzare, rileggere e stampare le parti di un testo. Il menu compare sul video quando il Wordwise viene avviato ed offre otto diverse opzioni. Le prime quattro riguardano il caricamento e la memorizzazione di un testo su o da un dischetto o una cassetta.
L'opzione 5 permette di trovare e/o sostituire specifiche parti di testo. Ad esempio, in un articolo si potrebbe voler sostituire la parola "opzione" con "scelta". Questa procedura può svilupparsi sia in modo globale che selettivo. Nel primo caso, la parola viene sostituita automaticamente tutte le volte che ricorre nel testo, mentre nel secondo si può decidere, caso per caso, se effettuare o meno la sostituzione. L'opzione 6 permette di inviare il testo in stampa, mentre la 7 consente di visualizzarne sullo schermo la versione finale pronta per la stampa. L'opzione 8 salva il testo con l'impaginazione standard selezionata dal Wordwise.
Per passare dal menu al modo "edit", occorre premere il tasto ESCAPE ed il testo appare con il cursore posizionato dove era stato lasciato in precedenza. Poiché lo schermo mostra il testo solo su 40 colonne, è spesso necessario saltare dal modo edit all'opzione 7 del menu per avere una visione complessiva dell'impaginazione del testo.
Un'interessante caratteristica è che i tasti di funzione, usati dal Wordwise per muovere, riprodurre o cancellare il testo e per compiere altre operazioni, possono essere ridefiniti, per scopi differenti, grazie ai tasti CTRL e SHIFT: per produrre un nuovo paragrafo, per esempio, o per cancellare un'intera linea o per inserire nel testo un codice di controllo per la stampante. Molto utili sono anche i comandi "stella" (*) del BBC, che si possono usare, durante la visualizzazione del menu, per selezionare un particolare tipo di stampante, un'unità a nastro o a dischetti oppure per riportare la macchina al BASIC, premendo "*B".
Il Wordwise è uno strumento molto utile per il Micro della BBC. Inoltre, è facile da istallare e da usare. Sono disponibili word processor più sofisticati come il View, ma con il Wordwise si ottengono risultati rispondenti alla maggior parte delle normali esigenze.

Analisi del testo

Come vengono memorizzate ed interpretate le linee dei programmi BASIC sui vari apparecchi

Molti credono che il computer, durante l'immissione di un programma BASIC, non esegua alcuna operazione, finché non venga impartito il comando RUN. Niente di più falso: fin dal momento in cui viene acceso, il computer passa automaticamente sotto il controllo di un complesso programma, il Sistema Operativo. Questo, in genere, è permanentemente memorizzato in un chip all'interno della macchina. Il suo compito è "semplicemente" quello di far funzionare la macchina: gestisce la visualizzazione delle immagini sullo schermo, comunica con la stampante e con l'unità a dischetti, interpreta i caratteri immessi da tastiera e così via. Il Sistema Operativo interpreta tutto ciò che costituisce l'"input", come dati da far elaborare ai suoi sottoprogrammi interni.
Per certi versi, l'interprete BASIC può essere considerato uno di questi sottoprogrammi. La sua funzione è quella di esaminare il testo di un programma BASIC e di tradurre le istruzioni ivi contenute in codice eseguibile. Quando scriviamo un programma, l'interprete lo riconosce come tale in quanto ciascuna linea inizia con un numero (chiamato, appunto, numero di linea). Salvo alcune eccezioni, il codice di ciascun carattere di una linea di programma viene depositato, uno per byte, in una specifica area di memoria RAM. Quando battiamo RUN l'interprete BASIC, come un qualunque altro programma, inizia un lavoro di elaborazione sui dati forniti (in questo caso le istruzioni BASIC).
Tutt'al più, se vi è un errore tra le istruzioni, visualizza una frase del tipo SYNTAX ERROR oppure OVERFLOW ERROR o altre simili. Ma non si sognerà mai di comunicarci un avviso del tipo PARADOSSO TEMPORALE o DISCONTINUITÀ FILOSOFICA.
In effetti, il BASIC memorizza il testo del programma carattere dopo carattere, eccezion fatta per le parole chiave BASIC. Ogni qual volta incontra una stringa di lettere (o caratteri o numeri oppure configurazioni di tensioni, se vogliamo) che compongono una parola chiave del BASIC, l'interprete sostituisce a quella parola un numero di codice, composto da un solo byte, chiamato "token" (letteralmente: gettone). Questa tecnica permette un notevole risparmio di memoria. Si pensi, ad esempio, alla parola chiave RESTORE, che occuperebbe, altrimenti, ben sette byte. Inoltre, l'uso di token rende molto più agevole il lavoro di conversione del programma BASIC eseguito dall'interprete.
Non esiste una tabella standard dei codici usati per i token ma, in generale, i valori impiegati sono numeri più grandi di 127. I codici ASCII per i caratteri stampabili (vedere la tavola a pag. 557) sono tutti compresi tra 32 e 127, quindi se un byte nell'area di memoria nella quale è memorizzato il testo di un programma contiene un numero superiore a 127, l'interprete sa di dover interpretare il numero come token per una parola chiave, ed attiva il sottoprogramma corrispondente alla specifica parola chiave.
Ci si potrebbe chiedere, allora, perché, quando viene stampato il listato di un programma, non vengano visualizzati questi codici, ma, viceversa, vengono stampate tutte le parole chiave. La risposta è abbastanza ovvia: durante l'esecuzione del comando LIST, l'interprete esamina ciascun byte dell'area riservata al testo, prima di visualizzarlo. Se il valore è superiore a 127, lo considera un token. Poiché in una speciale zona della memoria è registrata una tabella, contenente le rappresentazioni ASCII di tutte le parole chiave del BASIC, in base al codice numerico del token si risale alla parola associata. A questo punto, l'interprete visualizza la parola chiave e non il token. Questo procedimento è facilmente dimostrabile su un Commodore 64 (questa dimostrazione risulta meno diretta sul BBC o sullo Spectrum). Scriviamo, usando il Modo minuscolo, la seguente linea:

```
100 rem ********************* h *********************
```

Usando il comando LIST 100 si ottiene:

```
100 rem ********************* left$ *********************
```

Ciò accade perché sui computer Commodore il valore ASCII di "h", in Modo minuscolo, è 200, cosicché, nell'eseguire il LIST, l'interprete trova il valore di 200 e lo interpreta come il token per la parola chiave LEFT$. Se invece scriviamo:

```
100 rem ********************* H *********************
```

ed eseguiamo LIST 100, avremo:

```
100 rem ********************* H *********************
```

È importante ricordarsi che alcuni caratteri, di solito i caratteri grafici, hanno codici ASCII più grandi di 127 e che essi vengono riconosciuti soltanto se racchiusi tra virgolette. Altrimenti saranno trattati come token.
Vediamo adesso come vengono immagazzinate nella memoria le linee di programma BASIC. Generalmente, i primi tre o quattro

byte di una linea di programma BASIC, nell'apposita sezione di memoria riservata al testo, contengono il numero di linea e la lunghezza della linea. Il numero di ciascuna linea di programma viene immagazzinato, per quanto non sotto forma ASCII (altrimenti il numero di linea 61030 occuperebbe ben cinque byte). Al contrario, viene sempre immagazzinato sotto forma di due byte interi. In questo formato, i numeri da 0 a 255 (che possono essere immagazzinati in un unico byte di 8 bit) vengono immagazzinati come un byte 0 seguito da un byte che contiene il numero. I numeri più grandi di 255 vengono registrati esattamente come indirizzi impaginati (vedere a pag. 516): il valore del primo byte viene moltiplicato per 256 e sommato al valore del secondo byte. Il numero 1000, ad esempio, viene conservato come 3232 (3*256+232=1000). Questi due byte si trovano sempre nella stessa posizione in ogni linea memorizzata nell'area di testo (essi saranno o no i primi due byte della linea a seconda del tipo di computer usato).

L'informazione sulla lunghezza della linea viene inserita in un singolo byte nel BBC ed in due byte nello Spectrum. Essa rappresenta semplicemente il numero di byte della linea (compresi i due byte del numero di linea e lo stesso byte che indica la lunghezza della linea). Se conosciamo l'indirizzo di memoria del primo byte di una linea di programma BASIC e sommiamo ad esso il contenuto del byte della lunghezza della linea, otteniamo l'indirizzo del primo byte della linea di programma successiva. Poiché il numero più grande che un byte può contenere è 255, la lunghezza massima di una linea di programma BASIC sul BBC sarà quindi di 255 caratteri. Possiamo usare il programma Mempeek, presentato a pagina 539, per curiosare nella memoria del nostro computer: potremo così verificare direttamente come e dove vengono conservate le linee dei programmi BASIC.

Sul Commodore il byte della lunghezza della linea è sostituito da due byte chiamati "indirizzo di collegamento". Questo costituisce l'indirizzo reale in forma di due byte del primo byte della linea successiva del programma.

È interessante notare che nel BBC e nello Spectrum l'indirizzo iniziale della nuova linea viene calcolato sommando all'ultimo indirizzo la lunghezza della linea (questa procedura è più lenta, ma si risparmia un byte!). Sul Commodore, invece, l'indirizzo viene im-

Tavola di conversione

Il Codice Americano Standard per lo Scambio d'Informazioni (ASCII) attribuisce a ciascun carattere un valore compreso tra 0 e 127. I codici da 0 a 31 non corrispondono a caratteri visualizzabili, ma vengono invece usati per inviare segnali di controllo a periferiche come il video e la stampante. Il significato di questi codici di controllo può subire variazioni notevoli, da macchina a macchina, come si vede dalla tavola riprodotta a fianco. In alcuni computer, quali il Commodore e lo Spectrum, molti dei codici rimangono inutilizzati (sono quelli individuati da un pallino nero). I codici da 32 a 127 si riferiscono ai caratteri visualizzabili e i codici ASCII standard di questo gruppo sono comuni (ma con qualche eccezione) alla maggior parte dei computer

CODICE ASCII	STANDARD ASCII	COMMODORE	SPECTRUM	Micro BBC
0	NUL — Nessun effetto	●	●	Nessun effetto
1	SOH — Inizio testata	●	●	Prossimo carattere alla stampante
2	STX — Inizio testo	●	●	Attiva la stampante
3	ETX — Fine testo	●	●	Disattiva la stampante
4	EOT — Fine trasmissione	●	●	Separa cursori testo/grafica
5	ENQ — Richiesta	Tasto clr bianco	●	Unisce cursori testo/grafica
6	ACK — Ricevuto	●	PRINT	Abilita driver video
7	BEL — Campanello	●	EDIT	Emette un "bip"
8	BS — Backspace	Disabilita tasto CBM	Cursore a sinistra	Cursore indietro una posizione
9	HT — Tabulazione orizzontale	Abilita tasto CBM	Cursore a destra	Cursore avanti una posizione
10	LF — Line feed	●	Cursore in basso	Cursore in basso una posizione
11	VT — Tabulazione verticale	●	Cursore in alto	Cursore in alto una posizione
12	FF — Form Feed	●	Cancella	Cancella area testo
13	CR — Carriage return	RETURN	ENTER	RETURN
14	SO — Shift out	Modo minuscolo	Numero	Modo pagina on
15	SI — Shift in	●	●	Modo pagina off
16	DLE — Data link escape	●	INK	Cancella area grafica
17	DC1 — Device control 1	Cursore in basso	PAPER	Definizione colore testo
18	DC2 — Device control 2	Reverse	FLASH	Definizione colore grafico
19	DC3 — Device control 3	Cursore "a casa"	BRIGHT	Definizione colore logico
20	DC4 — Device control 4	Cancella	INVERSE	Ripristina colore logico
21	NAK — Negazione Ricevuto	●	OVER	Disabilita driver video
22	SYN — Sync. idle	●	AT	Selezione schermo
23	ETB — Fine blocco	●	TAB	Riprogramm. carattere
24	CAN — Cancella	●	●	Definizione finestra grafica
25	EM — Fine supporto	●	●	Plot m,x,y
26	SUB — Sostituzione	●	●	Ripristina finestre
27	ESC — Escape	●	●	Nessun effetto
28	FS — Separatore file	Tasto clr rosso	●	Definizione finestra per testo
29	GS — Separatore gruppo	Cursore a destra	●	Definizione origina grafica
30	RS — Separatore record	Tasto clr verde	●	Sposta cursore testo
31	US — Separatore unità	Tasto clr blu	●	Sposta cursore testo a x,y

CODICE ASCI	ASCII	CODICE ASCII	ASCII
32	Spazio	80	P
33	!	81	Q
34	"	82	R
35	#	83	S
36	$	84	T
37	%	85	U
38	&	86	V
39	'	87	W
40	(	88	X
41	)	89	Y
42	*	90	Z
43	+	91	[
44	,	92	\
45	—	93	]
46	.	94	^
47	/	95	_
48	0	96	|
49	1	97	a
50	2	98	b
51	3	99	c
52	4	100	d
53	5	101	e
54	6	102	f
55	7	103	g
56	8	104	h
57	9	105	i
58	:	106	j
59	;	107	k
60	<	108	l
61	=	109	m
62	>	110	n
63	?	111	o
64	@	112	p
65	A	113	q
66	B	114	r
67	C	115	s
68	D	116	t
69	E	117	u
70	F	118	v
71	G	119	w
72	H	120	x
73	I	121	y
74	J	122	z
75	K	123	{
76	L	124	¦
77	M	125	}
78	N	126	~
79	O	127	Delete

magazzinato tale e quale (usando un byte in più, ma guadagnando in velocità). Ciò dimostra, ancora una volta, che non esiste una soluzione "ideale" per la costruzione di un computer: ogni progettista adotta soluzioni diverse, ricorrendo talvolta a dei compromessi tra velocità e risparmio. Un simile problema s'incontra quando si scrivono programmi BASIC su computer dalla memoria limitata (ad esempio, il Vic-20 non espanso o lo ZX81). Spesso è necessario rinunciare a prestazioni brillanti, se vogliamo far entrare tutto il programma in memoria.
Ricordiamo, infine, che nell'area di testo, per ciascuna linea di programma, vengono usati dei codici di inizio e di fine linea. Sul Micro della BBC ogni linea comincia con un byte che contiene il numero 13 (codice ASCII per il Ritorno Carrello), mentre sullo Spectrum questo valore è usato come marcatore di fine linea. Ogni linea BASIC nel Commodore, invece, termina con un byte di valore 0.

LA MEMORIZZAZIONE DEI PROGRAMMI BASIC

Ogni macchina ha un proprio modo di immagazzinare le linee dei programmi BASIC. Esaminiamo le tecniche individuali mostrate per le linee di testo date più sotto

BBC MICRO

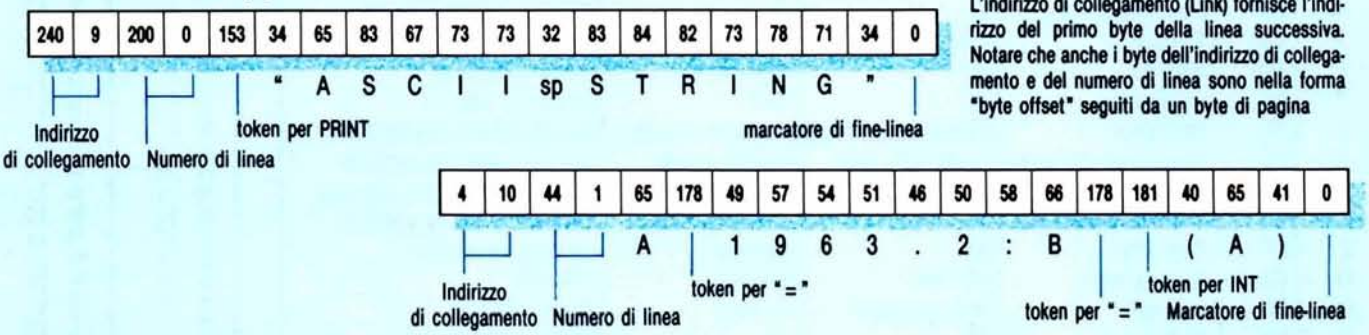
200 PRINT "STRINGA ASCII"
300 A = 1963.2:B = INT(A):A$ = "C"



In questo esempio le parole chiave BASIC vengono sostituite da token a singolo byte. Tutti gli altri caratteri sono immagazzinati come codici ASCII. I byte del marcatore di inizio linea, del numero di linea e della lunghezza della linea vengono aggiunti dall'interprete

COMMODORE 64

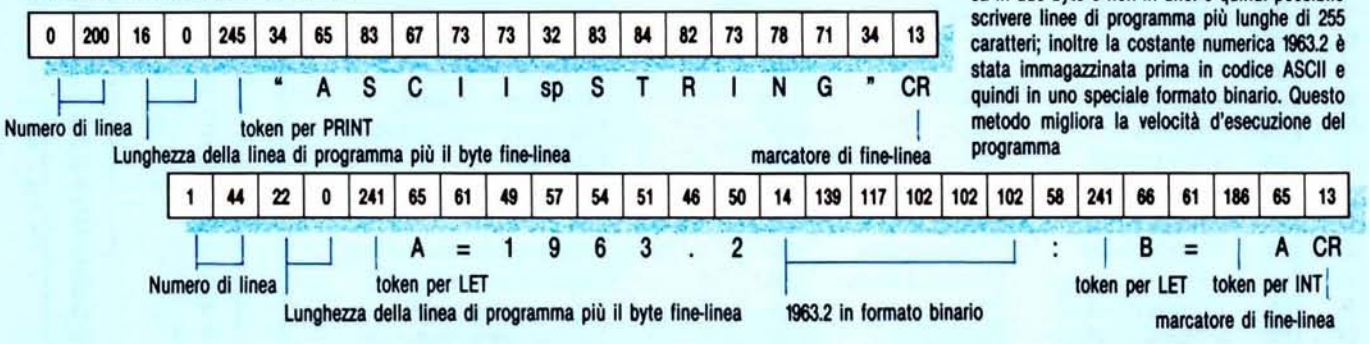
200 PRINT "STRINGA ASCII"
300 A = 1963.2:B—INT (A)



L'indirizzo di collegamento (Link) fornisce l'indirizzo del primo byte della linea successiva. Notare che anche i byte dell'indirizzo di collegamento e del numero di linea sono nella forma "byte offset" seguiti da un byte di pagina

SPECTRUM

200 PRINT "STRINGA ASCII"
300 LET A = 1963.2:LET B = INT A



Notare che la lunghezza della linea viene espressa in due byte e non in uno: è quindi possibile scrivere linee di programma più lunghe di 255 caratteri; inoltre la costante numerica 1963.2 è stata immagazzinata prima in codice ASCII e quindi in uno speciale formato binario. Questo metodo migliora la velocità d'esecuzione del programma

Il software della Imagine

Fondata da due soli giovani intraprendenti, la Imagine conta oggi più di cento dipendenti

Come tutte le industrie di successo, la Imagine ha raggiunto risultati di prestigio nel suo campo specifico grazie ad uno staff di esperti di notevoli capacità. I fondatori, David Lowson e Mark Butles, possiedono, rispettivamente, una notevole esperienza tecnica ed un grande acume commerciale.

La Imagine si è specializzata nella produzione di software per la gamma Commodore. Quattro dei loro giochi per i Commodore (il Bewitched, il Catcha Snatcha, il Wacky Waiters e l'Arcadia) sono stati per un certo periodo in testa alle classifiche di vendita. Grazie a questa stretta collaborazione tra la Commodore e la Imagine, una équipe di questa Software House è stata invitata a visitare la sede della Commodore a Norristow, in Pennsylvania, per visionare in anteprima le macchine 264 e V364. Questi apparecchi sono destinati soprattutto ai "patiti" dei giochi. A seguito della visita alla Imagine sono stati commissionati due giochi, che saranno commercializzati sotto il marchio Commodore: un risultato molto apprezzabile, per una ditta così giovane.

Nonostante ciò la Imagine, nel proprio paese, non si è mai trovata d'accordo con la Sinclair. Inizialmente, quando Mark Butler e Dave Lowson lasciarono la Bug-Byte Software, dove si erano conosciuti, ebbero alcuni contatti con la Sinclair, ma non raggiunsero con essa nessun accordo operativo. Sembra che i programmi di gioco, prodotti dalla Imagine, possano essere facilmente adattati ai nuovi modelli di computer con semplici modifiche alle locazioni di memoria usate: può darsi, quindi, che alcuni programmi verranno adattati al QL della Sinclair.

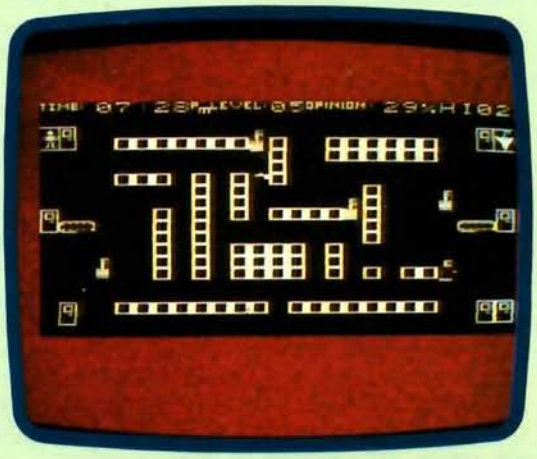
Il QL è basato sul microprocessore 68000 della Motorola e già oggi molti programmatori con esperienza di lavoro su questo chip operano nelle ditte di software professionale o lavorano nei dipartimenti informatici delle università. La Psion è la Software House responsabile per lo sviluppo dei 4 programmi che corredano il QL: conosciute con un anno di anticipo le caratteristiche tecniche della macchina, la Psion ha potuto simularne il funzionamento su di un Minicomputer Vax, per sviluppare in tempo i propri prodotti.

La Imagine ha cercato a lungo uno "staff" di capaci programmatori, familiari con il chip 68000, ma i risultati sono stati abbastanza deludenti. Benché le risposte siano state moltissime, solo pochi programmatori avevano un'esperienza adeguata ed alcuni non avevano nemmeno mai scritto un intero programma di giochi. Questi non erano certo gli uomini adatti a sviluppare una nuova generazione di programmi!

Sembra che, attualmente, la Imagine stia puntando verso il mercato del software commerciale, un settore più remunerativo e più competitivo. Pare che questa scelta sia dovu-

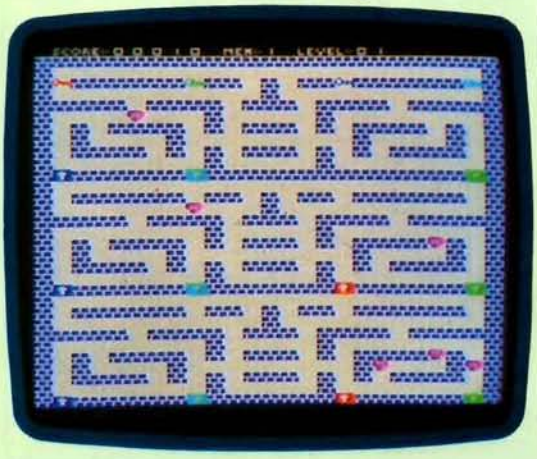
Catcha Snatcha

Questo particolare gioco del tipo "caccia/labirinto" è stato scritto per il Vic-20. Il protagonista è un detective dei grandi magazzini col compito di scoprire i ladri dei prodotti in mostra e di rintracciare i genitori dei bambini che si sono perduti nel grande magazzino



Labirinto stregato

Questo gioco presenta una particolarità: all'interno del labirinto ci sono delle porte che il giocatore deve aprire per avere accesso a tutti i settori. Il gioco è complicato dal fatto che non tutte le porte possono essere aperte: la posizione delle porte viene variata a caso ed occorre evitare i fantasmi che si aggirano nei corridoi



per concessione della SOFT

ta ai contatti instauratisi tra questa compagnia e la Apple. Dopo alcune difficoltà iniziali, la Imagine si è ben presto adattata ai progetti software richiesti dalla Apple, che, ultimamente, sono molto orientati ad un uso interattivo del video: finestre e "icone" controllate da un "topo". Questa impostazione è molto simile al software dei giochi, anch'essi corredati da sofisticati effetti grafici e da forme di immissione dati diverse dalla tradizionale tastiera.

La tecnologia ad 8 bit della Apple non era nuova alla Imagine poiché, sin dagli inizi, parte del lavoro veniva sviluppato su degli Apple IIe, complementati da dei computer Sage IV. Dave Lawson ha sviluppato una certa quantità di software di base, risultato utilissimo nella stesura dei programmi di giochi. Spesso, i programmi non vengono scritti utilizzando direttamente i computer ai quali essi sono destinati, ma si ricorre invece ad una compilazione "incrociata" ("cross compilation"). Ossia, si usa un diverso computer per la compilazione, molto potente e veloce, ottenendo programmi nel codice macchina, adatto, però, al computer "finale". Il Sage IV possiede un disco RAM di alta capacità assai utile quando si programma in Assembler. Per quanto riguarda i tempi di sviluppo, è più economico acquistare una macchina come il Sage, piuttosto che costringere i programmatori alle lunghe attese dovute alla lenta compilazione dei programmi su di un home computer. Per il Sage è disponibile il p-System della UCSD (Università della California a San Diego), un'implementazione del linguaggio PASCAL, che consente di simulare comodamente il funzionamento del Lisa e del MacIntosh della Apple. Come molte altre Software House, anche l'Imagine si sta oggi interessando molto al linguaggio per computer chiamato "C", che si sta dimostrando uno dei più versatili, grazie anche alla facilità con la quale i programmi possono essere trasferiti da un computer all'altro. La struttura modulare di questo linguaggio ne fa uno strumento ideale per sviluppare sistemi software.

Pochi sono oggi i programmatori esperti di "C" e di computer Sage. Per questa ragione l'Imagine sta "setacciando" le università e le Software House alla ricerca di esperti in questi campi.

Questa è la Imagine!

La forza di ogni Software House poggia sulle idee dei propri progettisti di programmi. Associatisi con il preciso scopo di sviluppare i giochi "Psyclapse" e "Bandersnatch", i "cervelli" della Imagine sono, a partire da sinistra, Ian Weatherburn, Mike Glover, John Gibson e Eugene Evans



per concessione della IMAGINE

La formula vincente

Che cosa trasforma un prodotto software in un best-seller? Molto dipende dalla sua originalità. Quando Butler e Lawson dettero vita all'Imagine, avevano un'idea piuttosto chiara dei prodotti che la compagnia nella quale lavoravano in precedenza stava commercializzando con successo. Essi cercarono di calarsi nella psicologia dei più accaniti utenti di giochi per computer. Si posero domande di questo tipo: "Se questo fosse il mio primo computer, che cosa vorrei da questo gioco? Perché ci giocherei? Per quanto tempo? Mi piacerebbero degli effetti sonori o dei "flash" grafici in questa fase del gioco?" Questo è certo il miglior modo di affrontare i problemi della programmazione dei giochi.

La Imagine è cresciuta da allora, ed attualmente impiega uno staff di 8 programmatori grafici, completamente dedicato all'animazione e allo sviluppo della trama dei nuovi giochi, che vengono poi esaminati da un diverso gruppo di programmatori.

Sfortunatamente per tutte le Software House, ma in particolare per quelle produttrici di giochi, quanto più un programma è valido tanto maggiore è la quantità di copie, riprodotte illegalmente, che invadono il mercato. Il fenomeno di "pirateria", dalla semplice "copia" fatta per l'amico alla produzione di massa organizzata, arreca ingentissimi danni a tutto il settore. È stato stimato che per ogni cassetta di giochi venduta legalmente ne vengono fatte almeno sette copie illegali, la maggior parte impiegando semplicemente due registratori. Sono state adottate misure preventive per eliminare o almeno ridurre questa pratica, ma esse sono costose e questo costo extra dovrebbe poi essere caricato sull'acquirente. L'Imagine ha senza dubbio subito forti perdite, a causa delle copie illegali, tuttavia i suoi indici di vendita sono ancora molto alti.

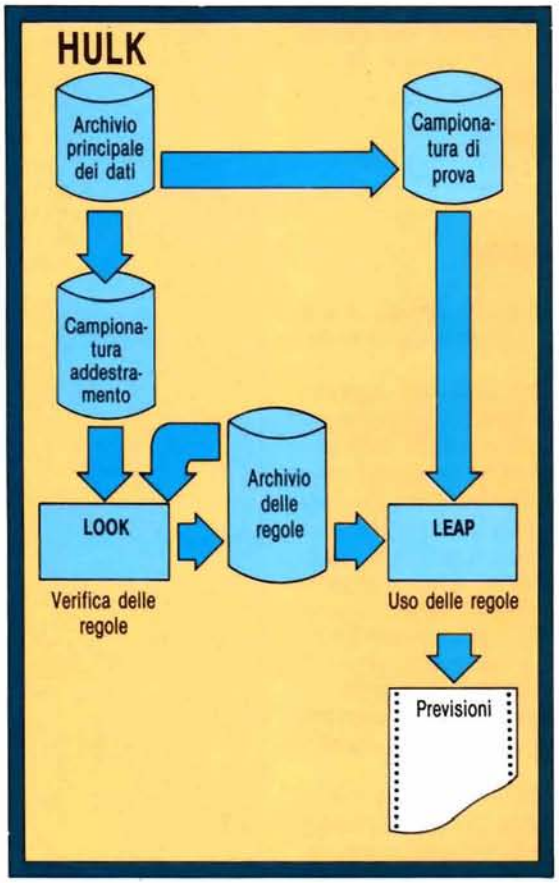
Molto presto le cartucce o i dischi rimpiazzeranno completamente le cassette ma già stanno comparendo metodi rivoluzionari per la distribuzione del software: i programmi potrebbero venire trasmessi da un computer centrale e caricati sugli home computer attraverso i cavi telefonici di supporto alla televisione via-cavo. Una Software House che intenda espandersi in questi anni deve tener presente questi sviluppi; per questa ragione Mark Butler si è interamente dedicato alla progettazione di nuovi tipi di giochi facendo uso di sintesi vocale e di altre nuove tecnologie come, ad esempio, il disco a laser.

L'incredibile Hulk

Possono i computer sostituire l'intervento di persone esperte e qualificate?

Col nome di "sistema esperto" si indica una sorta di banca dati concernente uno specifico settore della conoscenza umana, gestita da sofisticati programmi, che consentono di accedere in maniera organizzata alle informazioni contenute nell'archivio.
Consideriamo, ad esempio, la formulazione di una diagnosi medica. L'abilità di un medico consiste nel saper collegare i sintomi accusati dal paziente a quanto riportato sui libri di testo e riscontrato in anni di esperienza. In effetti, anche se la realtà è un poco più complessa, la diagnosi rappresenta un'attenta valutazione di una situazione e di tutta una serie di eventi ad essa collegati. I fattori principali di questa procedura sono l'abilità di ricordare informazioni in modo organizzato, l'abilità di collegare un particolare evento ad altri accaduti precedentemente e l'abilità di sopperire, laddove gli elementi siano scarsi o non perfettamente concordanti, con esperienze passate. I primi due elementi, l'organizzazione e la classificazione di dati sono la specialità dei computer, mentre nel terzo l'uomo è il più esperto. Se si riesce, però, a sostituire la componente intuitiva (tipica dell'uomo) con un'efficiente analisi di correlazione statistica, allora è forse possibile ottenere un sistema computerizzato dalle prestazioni incredibilmente superiori a quelle umane.
Simili sistemi sono particolarmente adatti nei settori dove il giudizio umano è determinante, mancando le basi di una teoria vera e propria, come accade, appunto, nella formulazione di una diagnosi. Altre applicazioni si possono prevedere in campi diversi quali la legislazione fiscale o la determinazione dei requisiti necessari per l'assistenza pensionistica: le informazioni, sia pure pubblicamente note e già strutturalmente organizzate, sono spesso troppo complesse per l'applicazione da parte della gente comune. Poiché la conoscenza, intesa come una raccolta di re-





LIZ DIXON

Il sistema HULK

In un sistema esperto completo, l'archivio di base (detto anche "della conoscenza") contiene tutta una serie di regole espresse nel formato SE... ALLORA, che vengono applicate dal "motore delle inferenze", quando l'utente interroga il sistema, attraverso il modulo d'interrogazione. La raccolta di conoscenze, specifica ad una particolare materia, viene immessa da esperti mediante un "modulo di acquisizione", incorporato nel sistema. Il sistema HULK, scritto per i microcomputer, costituisce una versione ridotta di un sistema esperto vero e proprio: i due programmi che lo compongono, LOOK e LEAP, consentono di impostare e verificare le regole che si suppongono adatte ad una previsione particolare



LIZ HEANEY

Un progetto pratico

Il sistema HULK, concepito e realizzato da Richard Forsyth, un giovane laureato inglese, è stato progettato in soli quindici giorni, ma sono poi occorsi più di sei mesi per raffinare le tecniche d'interazione con il mondo esterno (in particolare con l'operatore). È probabile che ne venga prodotta una versione adatta al QL Sinclair

gole logiche, è di capitale importanza in questo tipo di sistemi, ad essi viene spesso dato l'appellativo di sistemi "fondati su regole" o "fondati sulla conoscenza".
I sistemi esperti stanno già dimostrando la loro superiorità, rispetto alle capacità umane, in più settori, tra i quali quello medico, precedentemente citato, e quello della ricerca delle sostanze minerali. Gli studi, attualmente in corso, sull'Intelligenza Artificiale (più brevemente AI, da Artificial Intelligence) sono particolarmente orientati alla produzione di sistemi di larga applicazione, con conseguenze determinanti sui metodi adoperati per il progetto dei sistemi hardware. In sostanza, il criterio tradizionale:

DATI + ALGORITMO = PROGRAMMA

viene sostituito da:

CONOSCENZA + INFERENZA = SISTEMA

che rappresenta una notevole svolta nella filosofia dei sistemi automatizzati, dalle enormi conseguenze pratiche.
Le altre componenti di un sistema esperto completo, oltre al nucleo delle informazioni (la conoscenza) sono: il "motore delle inferenze", che crea nuove regole per l'interpretazione dei dati in base alle regole ed alle informazioni già esistenti; il "modulo di acquisizione delle conoscenze", attraverso il quale la macchina raccoglie conoscenze in base alla propria "esperienza" ed a quella degli esperti umani; l'"interfaccia utente", che permette ai non esperti di accedere e di interrogare il sistema. Vediamo in maggior dettaglio le prime due di queste componenti: il "motore delle inferenze" ed il "modulo di acquisizione delle conoscenze".
Un "archivio di conoscenze" (in inglese viene usato il termine "knowledge base") contiene fatti (o asserzioni) e regole. I fatti possono mutare rapidamente (anche, ad esempio, durante la stessa consultazione). Le regole, invece, costituiscono un insieme di meccanismi, di maggior validità temporale, in base ai quali vengono generate delle ipotesi basate su fatti già acquisiti. La principale differenza tra un normale archivio dati (database) ed un archivio di conoscenze (knowledge base) consiste nella maggiore creatività di quest'ultimo. Generalmente, i fatti contenuti in un database sono entità certe e passive: un particolare dato viene archiviato e richiamato senza che su di esso venga operato alcun processo modificativo. Nei knowledge base, viceversa, le informazioni vengono automaticamente modificate ed interpretate (e talvolta addirittura integrate) in base all'esperienza ed ai dati già contenuti nell'archivio.
Le regole espresse nel formato SE... ALLORA (note come "regole produttive"), sono quelle preferite da questi sistemi. Ad esempio:

SE la squadra che gioca in casa ha perso l'ultima partita in casa E la squadra ospite ha vinto con due goal la sua ultima partita

PREVISIONI DEL TEMPO

Gli archivi dei dati, nel sistema HULK, si possono creare normalmente usando il BASIC, e sono pertanto di facile accesso. Il file qui riportato a titolo d'esempio è abbastanza tipico.

Al sistema viene fornita la seguente ipotesi sui dati:

DOMANI = 1

significando che si è interessati soltanto ai campioni nei quali l'ultimo valore è 1 (nella nostra codifica, ciò equivale alla data del giorno precedente uno nel quale sia piovuto). L'intento è quello di formulare, mediante l'uso di HULK, una regola che ci consenta di prevedere quali saranno le condizioni atmosferiche di domani, in base a quelle odierne.
A questo punto vengono introdotte delle ipotesi di regole, ciascuna delle quali viene verificata e valutata sia separatamente, sia globalmente rispetto alle altre. In breve tempo siamo giunti all'individuazione di tre regole per la previsione delle condizioni meteorologiche:

1) Se oggi vi sono stati più di mm. 2 di pioggia

E

2) Se oggi c'è stato il sole per un periodo inferiore alle 3 ore e mezzo

E

3) Se la massima temperatura di domani è inferiore di almeno 6 gradi, rispetto a quella minima di oggi,

ALLORA

si ha un 83% di probabilità che domani piova (purché, ovviamente, i dati usati per la prova siano sufficientemente rappresentativi dei valori di tutti i periodi dell'anno).

Benché quello che segue possa sembrare un programma BASIC, in realtà si tratta soltanto di un formato comodo da usare con il sistema HULK.

```
1 METEOROLOGIA,30,5
```

La linea 1 non è che una descrizione dell'archivio. Contiene il nome (METEOROLOGIA), il numero di campioni (30) ed il numero di elementi per ogni campione (5). Nell'esempio, i dati si riferiscono ad un arco di tempo di un mese, in una città notoriamente piovosa: Londra. Oltre al numero identificativo del giorno (G01, G02,...), sono riportati: la temperatura minima, la massima, la quantità di pioggia (in millimetri), le ore di sole ed, infine, una variabile Booleana (1 = nel giorno successivo è piovuto, 0 = non è piovuto).

```
100 TMINIMA
200 TMASSIMA
300 PIOGGIA
400 SOLE
500 DOMANI
```

Le linee 100-500 assegnano i nomi ai gruppi di dati che seguono:

```
1001 G01,54,110,175,32,1
1002 G02,42,125,041,62,1
1003 G03,76,112,077,11,1
1004 G04,27,105,018,43,0
1005 G05,3i,120,000,95,0
1006 G06,44,106,000,55,0
1007 G07,48,094,000,51,1
1008 G08,68,092,055,48,1
1009 G09,64,102,048,41,1
1028 G28,58,154,000,20,1
1029 G29,67,088,064,42,0
1030 G30,45,096,000,68,1
```

Le linee da 1001 a 1030 contengono i dati, con un campione per linea. Ogni campione inizia con un titolo identificativo della particolare linea di dati. Si noti che i valori, per il sistema HULK, devono essere interi. Pertanto, onde ottenere una certa precisione, abbiamo moltiplicato tutti i valori per 10, il che non influisce sui risultati.

```
DOMANI = 1
RICHIESTA DELLA REGOLA [1]
LA REGOLA È: [PIOGGIA > 20]

TAVOLA DI CONTINGENZA
```

	SUCCESSI	INSUCCESSI
REGOLA VERA	12	2
REGOLA FALSA	5	11

```
TASSO DI SUCCESSO = 76,3%

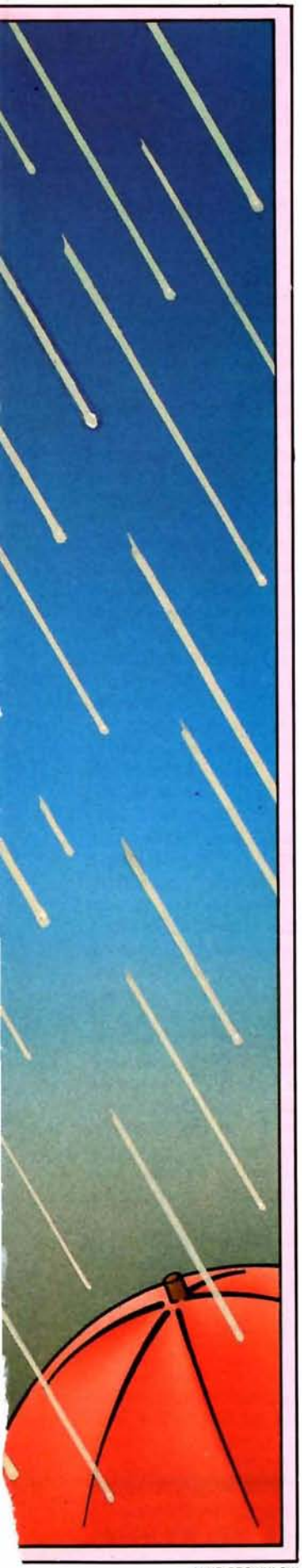
ELEMENTI PER CAMPIONE
```

PRIMA	PIOGGIA > 20	1.00
DOPO	PIOGGIA > 20	0.85

```
SI CONSIGLIA DI [MANTENERE] LA REGOLA OK
LA REGOLA È STATA [INCLUSA] NELL'ARCHIVIO
```

Elementi per campione
Questa è una valutazione circa il numero di campioni che confermano la validità della regola





MIKE BROWNLOW

giocata in casa, ALLORA la probabilità di un pareggio è moltiplicata per 1,088.

Questo genere di regole non è inglobato nel codice del programma, ma costituisce, invece, il tipo di dati sul quale lavora il "motore delle inferenze", che si può paragonare ad un "interprete" ad alto livello.
Esistono due principali strategie ad alto livello per la formulazione di inferenze: la "concatenazione in avanti" e quella "all'indietro". La prima coinvolge un metodo di esame dei dati con lo scopo di formulare ipotesi accettabili, mentre la seconda è orientata a trovare conferme, nei dati, a supporto di ipotesi già formulate.
I sistemi esperti più sofisticati adoperano ambedue queste strategie. Qualunque sia il criterio adottato, comunque, una procedura d'inferenza elabora sempre dei dati non necessariamente certi ed assodati. Fin dagli inizi dell'informatica, infatti, sono falliti tutti i tentativi di far rientrare le caratteristiche del mondo in cui viviamo in rigidi schemi precostituiti. Al contrario, le ricerche sui sistemi esperti tentano di fornire uno strumento adattabile a tutte le circostanze mutevoli del mondo reale.
Le teorie, basate sul concetto di "incertezza", sono molte e talvolta molto complicate, ma in genere sono tutte tese a sostituire frasi quali : SE X È VERO, ALLORA Y È VERO, con inferenze statistiche più caute, del tipo: SE X È VERO NEL 65% DEI CASI, ALLORA LE POSSIBILITÀ CHE SIA VERO Y STANNO TRA IL 50% ED IL 70%. La cosa che più stupisce, in questa materia, è che anche gli schemi logici apparentemente più strani sembrano funzionare bene. Una ragione di ciò potrebbe essere che l'organizzazione delle informazioni, in questo settore, conta molto più dei valori numerici associati a dette informazioni. Molti knowledge base offrono la possibilità di giungere alle medesime conclusioni attraverso percorsi totalmente diversi ed i valori numerici, che rappresentano il grado di credibilità, assumono importanza soltanto per selezionare i percorsi migliori da seguire.
Recentemente, è stato posto in commercio del software progettato per facilitare la creazione e la gestione di knowledge base o "archivi di conoscenza", ma l'unico programma alla portata dei possessori di home e personal computer è, per il momento, HULK (dalle iniziali di Help Uncover Latent Knowledge, ossia "aiuto per svelare la conoscenza latente") scritto per il Micro BBC.
HULK permette la costruzione e la verifica di tutta una serie di regole, da poter in seguito usare per previsioni o classificazioni. Il programma risulta particolarmente utile se si hanno dei gruppi di misurazioni, caratterizzate da più variabili, e se desideriamo trovare il grado di correlazione fra di esse, onde poter formulare previsioni ragionevoli su eventi simili.
Supponiamo, ad esempio, che il proprietario di una vigna archivi su di un computer una dettagliata serie di misurazioni (altezza, colore delle foglie, zona di piantagione, ecc.) di svariate centinaia di viti. Al momento della vendemmia, vengono archiviate le informazioni relative alla produzione di ciascuna pianta. Il sistema, sulla base di ciò, permette di sviluppare alcune regole sul miglior metodo di coltivazione da usare in futuro.
Un altro esempio può riguardare il Totocalcio: sulla base dei risultati ottenuti dalle varie squadre nei precedenti campionati, nelle diverse situazioni di gioco, si potrebbero ottenere delle ragionevoli "proiezioni" su di un particolare incontro. In ambedue i casi, le informazioni da fornire al sistema sono facilmente ottenibili, ma il lavoro necessario ad estrarre delle "regole", da tutta la messe dei dati, sarebbe certamente impossibile, se condotto manualmente. Le potenziali applicazioni di HULK sono innumerevoli: basta partire con un sufficiente numero di dati sui quali lavorare.
HULK è composto da due programmi: LOOK (da Logical Organizer Of Knowledge, ossia "organizzatore logico della conoscenza") e LEAP (da Likelihood Estimator And Predictor, ovvero "valutatore e proiettore di probabilità"). Con questi, si può sviluppare un insieme di regole che caratterizzano un gruppo di dati, precedentemente memorizzati su dischetto, per poi applicarlo ad una quantità di dati ancora incompleta, onde ottenere delle previsioni (ad esempio, i risultati di un incontro di calcio non ancora giocato).
L'archivio di conoscenze, creato da HULK, cresce mediante l'apporto dell'utente: alcune regole vengono suggerite dall'operatore e verificate dal sistema ed è lo stesso utente a decidere quali scartare come meno significative, rispetto all'andamento globale.
Per usare il programma LOOK è sufficiente un insieme di dati, ma con due insiemi il lavoro è più completo: uno viene chiamato l'insieme "di addestramento" (per la prova delle regole), l'altro, invece, "di prova" (per confermare l'utilità delle regole ed introdurre situazioni nuove non previste).
Al programma LOOK viene "dato in pasto" il primo insieme, assieme alle regole proposte. Ad una ad una, esse vengono provate, ottenendo una specie di votazione, sul grado di utilità di ciascuna regola ai fini della previsione dei risultati. Spetta all'utente la decisione finale di adottare o no la particolare regola.
LOOK non è un programma di acquisizione propriamente detto, ma una sorta di "filtro", che permette di selezionare soltanto quelle ipotesi o congetture degne di considerazione. È l'utente che svolge la funzione di apprendimento, mentre il computer si occupa di tutto il lavoro di confronto, di calcolo e di valutazione.
Il programma LEAP viene successivamente usato per applicare le regole ad una nuova campionatura di dati, ottenendo previsioni sui probabili valori di quelli non ancora noti.
Benché non rappresenti un vero e proprio sistema esperto, HULK può servire egregiamente per apprendere questa nuova applicazione dei computer.

Unità a dischi BBC

Una breve descrizione del Disk Filing System (DFS) per il Micro BBC e dei suoi comandi

Virtualmente, il Micro BBC può essere collegato ad una qualsiasi unità a dischi da 5 pollici e 1/4, e pare che non vi siano ostacoli nell'impiego di unità ad 8 pollici. Sfortunatamente, il sistema operativo per dischetti, chiamato DFS (Disk Filing System) non viene ceduto assieme al computer, ma va acquistato separatamente, sotto forma di chip ROM da inserire in un apposito zoccolo. Esistono diverse versioni del DFS, anche non prodotte dalla Acorn, dotate di minime varianti, ma il prezzo non varia molto.

Poiché il DFS è contenuto nella ROM, molti comandi relativi alla gestione dei dischi non occupano la memoria centrale del sistema. Il DFS, comunque, dispone di alcuni programmi "di utilità", una sorta di estensioni al sistema operativo, memorizzati su dischetto. Questi sono relativi al particolare tipo di unità impiegata e servono per il formattamento dei dischi e la verifica dei dati ivi memorizzati.

Le unità a dischi vengono collegate al computer mediante un cavo piatto a 34 poli, che fa capo alla presa indicata con "DISK DRIVE", nella parte inferiore dell'involucro del computer. Questo cavo trasporta tutti i segnali indirizzati ai dischetti o da essi provenienti, sia che si tratti di una singola unità sia di due. A tutte le funzioni di controllo provvede un chip, che ha per sigla 8271. Esso converte tutti i segnali dalla forma parallela, emessa dal computer, a quella seriale, necessaria al trasferimento e viceversa. I formati di formattamento attualmente disponibili sono quattro: 40 o 80 tracce in singola faccia, 40 o 80 tracce in doppia faccia, tutti in singola densità. Ciascuna traccia è composta da 10 settori, ognuno dei quali contiene 256 byte. Ne risulta una capacità totale di circa 100 Kbyte per la versione a 40 tracce e di circa 200 Kbyte per quella ad 80 tracce.

Le unità, o "drive", sono identificate da un numero: 0 per la prima unità ed 1 per la seconda. Se però vengono usate unità per dischetti a doppia faccia, allora si attribuisce un numero ad ogni lato del dischetto, con numerazione alterna: 0 e 2 per le due facce del primo dischetto, 1 e 3 per il secondo.

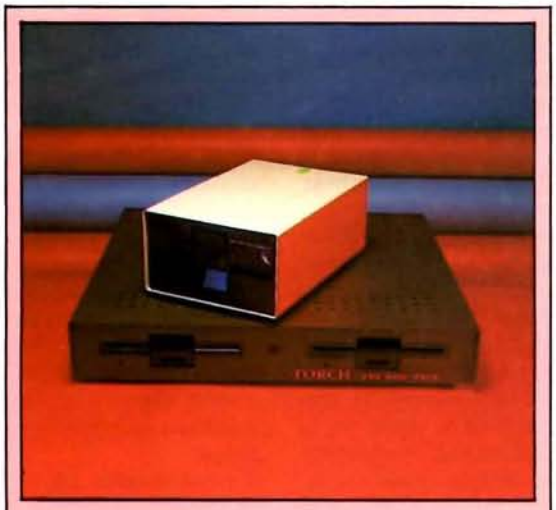
L'elenco dei file contenuti sul dischetto (o sulla facciata) occupa 2 Kbyte e si trova nei primi due settori della prima traccia. Questo "catalogo" dei file contiene informazioni relative sia al dischetto in generale, sia ai singoli archivi memorizzati, permettendo di creare fino a 27 diverse "directory". Ciò consente di raggruppare comodamente i file per tipo o categoria. Non esiste alcuna mappa di disponibilità (BAM), mentre è presente un comando *COMPACT, che serve ad individuare eventuali blocchi inutilizzati ed a riorganizzare in modo ottimale la conservazione dei dati, per sfruttare tutto lo spazio libero. La procedura per la memorizzazione di archivi dati e di programmi è pressoché identica, sia che venga usata l'unità a dischi sia l'unità di registrazione a cassette. In effetti, il computer seleziona automaticamente il funzionamento coi dischetti (qualora la ROM del sistema operativo sia inserita). In tal caso, per usare il registratore, occorre impartire il comando *TAPE, mentre, per tornare ai dischetti, il comando è *DISK.

Il sistema operativo mette a disposizione una ricca gamma di comandi, piuttosto sofisticati, tra i quali anche uno di "assistenza" (*HELP), utile quando non si ha il manuale sottomano, ed occorre conoscere il formato di uno degli altri comandi.

Il trasferimento dei dati da e per i dischetti ha una velocità considerevole e, tutto sommato, il sistema operativo del BBC offre delle prestazioni superiori al normale standard degli home computer. L'unico svantaggio è che ciascuna faccia del disco può contenere, al massimo, 31 nomi di file: considerando la capacità totale, questa è una limitazione abbastanza forte.



Il Signore... delle stringhe!
L'unità "Hobbit" (una via di mezzo tra i sistemi a dischetti e quelli a nastri) è compatibile col Micro BBC, rappresentando un'alternativa economica ai più costosi dischetti. Tutte le funzioni (riavvolgimento, avanzamento, ecc) sono controllate via software



Le unità a dischi BBC
Per sfruttare appieno le prestazioni del Micro BBC, è essenziale disporre di un'unità a dischetti. Le due qui illustrate sono quelle più diffuse: l'Acorn 100 e la Torch Z80. Per l'uso dei dischetti, è necessario acquistare un chip ROM che contiene il sistema operativo

IAN McKINNELL

I COMANDI PER I DISCHI DEL BBC

In tutti i casi ove occorre specificare il file al quale si riferisce un comando è usata la notazione:

COMANDO:DV.DR.NOMEFILE

dove: COMANDO è il particolare comando in esame; DV è il numero identificativo del drive (da 0 a 3); DR è il carattere identificativo della directory (da A ad X oppure $); e NOMEFILE è il nome del file in questione, composto da un massimo di 7 caratteri (non sono permessi '#', '*', '.', ':' all'interno di un nome). Se non vengono specificati il numero di drive o la directory (o nessuno dei due), il sistema seleziona automaticamente l'unità 0 e la directory $, ma ciò si può modificare intervenendo coi comandi *DIR e *LIB. Il sistema operativo prevede l'impiego di caratteri "jolly" (ossia "#" e "*") in alcuni dei comandi, affinché questi abbiano effetto su più di un file alla volta.
Nelle descrizioni che seguono, la notazione relativa ai file viene abbreviata con <FSP>.

Oltre ai normali comandi per registratore a cassette, attivati all'accensione dell'apparecchio, il sistema operativo su ROM contiene i seguenti comandi per i dischetti:

***FORM40, *FORM80 e *VERIFY**
Questi comandi "di utilità" sono conservati sul dischetto di sistema e servono per inizializzare i dischetti (con 40 o 80 tracce) e per verificarli.

***ACCESS**
*ACCESS <FSP> L protegge il file da eventuali cancellature o sovrascritture accidentali. *ACCESS <FSP> rimuove la protezione.

***BACKUP, *DESTROY ed *ENABLE**
*BACKUP DVsorgente DVdestinazione copia l'intero contenuto del dischetto "sorgente" nel dischetto "destinazione" (distruggendo il vecchio contenuto di quest'ultimo). *DESTROY <FSP> cancella il file specificato. Se nella specifica del file <FSP> sono contenuti dei caratteri "jolly", allora si può usare un solo comando per cancellare un intero gruppo di file. Poiché l'effetto di questo comando è "distruttivo", per attivarlo occorre prima digitare *ENABLE.

***BUILD <FSP>**
Il comando crea un file ASCII, nel quale viene copiato tutto ciò che si batte alla tastiera, fino alla pressione del tasto ESCAPE.

***CAT DV**
Visualizza il catalogo del dischetto.

***COMPACT DV**
Riorganizza la distribuzione dei dati sul disco in modo da utilizzare tutto lo spazio disponibile.

***COPY**
Copia un file (o un gruppo di essi, se si usano i caratteri "jolly") da un dischetto ad un altro.

***DELETE <FSP>**
Elimina il file specificato dalla directory di un disco.

***DIR DR**
Permette di modificare la selezione automatica della directory. Tutti i file, successivamente, vengono memorizzati nella nuova directory.

***DRIVE DV**
Permette di modificare la selezione automatica dell'unità a dischetti.

***DUMP <FSP>**
Visualizza il contenuto di un file in formato esadecimale.

***EXEC <FSP>**
Permette di usare il testo di un file ASCII, creato con il comando BUILD, come se si trattasse di caratteri digitati da tastiera.

***HELP**
Visualizza un elenco parziale dei comandi disponibili per i dischetti ed il loro formato. *HELP UTILS visualizza i restanti comandi.

***INFO <FSP>**
Visualizza informazioni ausiliarie sulle caratteristiche dei file, normalmente non fornite dal comando *CAT (collocazione in memoria, indirizzi d'esecuzione, lunghezza in byte e collocazione sul disco).

***LIB:DV:DR**
Permette di specificare quale directory usare come "libreria privilegiata", consentendo, successivamente, di usare il comando abbreviato *NOMEFILE, per il caricamento e l'esecuzione di un programma.

LIST <FSP>
Visualizza il file ASCII specificato, completo di numeri di linea.

***LOAD <FSP>**
Legge dal dischetto il file specificato, collocandolo nella memoria centrale del computer.

***OPT 1**
Per abilitare la visualizzazione delle informazioni normalmente fornite dal comando *,9x69INFO, ogni volta che si accede ad un file, si impartisce il comando *OPT 1 1. Per disattivare la funzione, usare *OPT 1 0.

***OPT 4**
Consente di modificare la procedura di avviamento automatico, che avviene all'accensione o alla pressione del tasto (SHIFT) BREAK. *OPT4 0 disabilita l'avviamento automatico; *OPT 4 1 esegue un LOAD del file !BOOT; *OPT 4 2 esegue un RUN di !BOOT e *OPT 4 4 un EXEC di !BOOT.

***RENAME <vecchia FSP> <nuova FSP>**
Serve per cambiare il nome ad un file, spostandolo in una nuova directory. Non trasferisce i file da un disco ad un altro.

***RUN <FSP>**
Serve per caricare ed eseguire un programma non contenuto nella "libreria" attualmente in uso.

***SAVE**
Copia la zona di memoria specificata sul dischetto attualmente selezionato. Il formato completo è:

*SAVE "NOME" SSSS FFFF EEEE RRRR

oppure

*SAVE "NOME" SSSS + LLLL EEEE RRRR

dove SSSS è l'indirizzo iniziale della zona da copiare, FFFF quello finale, EEEE l'indirizzo da usare per l'esecuzione ed RRRR è l'indirizzo iniziale da usare, successivamente, in fase di lettura. LLLL è la lunghezza (in byte) del blocco di memoria, in alternativa ad FFFF. Tutti questi valori vanno espressi in esadecimale. RRRR ed EEEE possono essere omessi, in tal caso equivarranno a SSSS.

***SPOOL <FSP>**
Consente di memorizzare un programma BASIC in formato ASCII, anziché nello speciale formato compresso non leggibile.

***TITLE "NOME DISCO"**
Sostituisce il nome del dischetto con quello specificato.

***TYPE <FSP>**
Visualizza il contenuto di un file ASCII, ma senza numerare le linee.

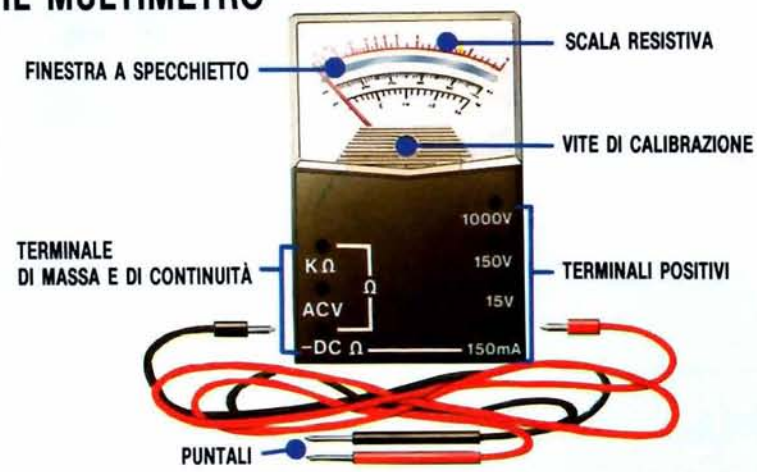
***WIPE <FSP>**
Identico a *DESTROY, ma non richiede il preventivo uso di *ENABLE.

Banco di prova

Prima di dar tensione ad un apparecchio elettronico appena costruito o riparato, occorre sempre verificare che non vi siano né cortocircuiti, né discontinuità elettriche

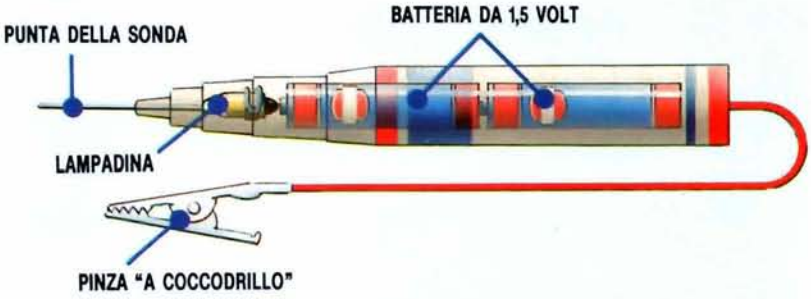
La prima verifica da eseguire, su di un cablaggio di componenti elettronici, è quello sull'efficienza delle saldature. Una giunzione fatta ad una temperatura troppo bassa può sembrare accettabile, esternamente, ma al suo interno la conduzione elettrica potrebbe essere interrotta. Smuovendo delicatamente i terminali della giuntura essi non devono separarsi (se ciò accade, la saldatura è comunque da rifare).
Adesso si può passare all'uso di un "tester" di qualche tipo. Nelle versioni più semplici, un provacircuiti è composto da una sonda appuntita, una pila, una lampadina a basso voltaggio ed un cavetto "di ritorno", dotato di un morsetto del tipo a "coccodrillo". Il tutto forma un circuito elettrico, nel quale la lampadina s'illumina solo se la connessione in corso di verifica non è interrotta.
Uno strumento di gran lunga più utile, comunque, è il multimetro. Selezionando, su di esso, la funzione di Ohmmetro, non solo si può misurare la resistenza elettrica di un materiale, ma anche se vi è continuità nella conduzione elettrica. Il nome dell'unità di misura "Ohm", deriva da quello del fisico tedesco Georg Ohm (1787-1854), che per primo scoprì il fenomeno della resistenza elettrica. La resistenza di un materiale varia proporzionalmente alla sua sezione trasversale, ma essa può essere modificata artificialmente, ricorrendo all'uso di componenti chiamati, appunto, "resistenze". Ai nostri fini, la saldatura deve offrire una resistenza completamente trascurabile alla corrente generata dal tester, cosicché la lampadina s'illuminerà completamente, o l'ago dell'Ohmmetro devierà completamente verso lo 0 della scala. Con qualsiasi altro risultato (o in caso di dubbio), è consigliabile rifare la saldatura.
Oltre alla funzione di Ohmmetro, un multimetro serve anche per misurare le correnti (in Ampère) e le tensioni (in Volt). Queste due unità elettriche sono in stretta connessione tra loro: la differenza di potenziale tra due punti di un circuito, tra i quali scorre una corrente di un Ampère e nel quale è dissipata la potenza di un Watt, è di un Volt.

IL MULTIMETRO

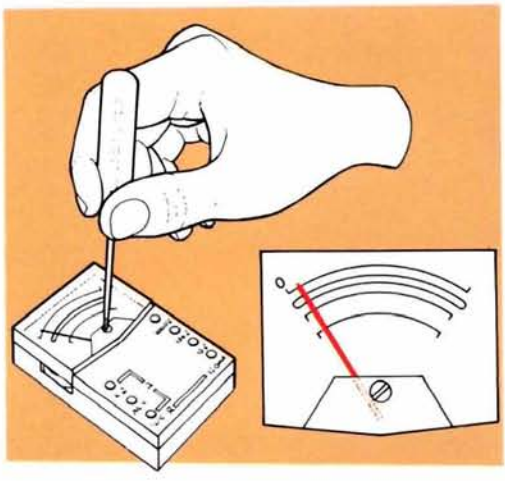


I Multimetri servono per misurare diverse quantità elettriche: la resistenza (in Ohm), la corrente (in Ampère) e la tensione o differenza di potenziale (in Volt). La visualizzazione della lettura avviene su di una scala ad ago tradizionale (come quella di una bilancia, per intendersi) o con i più moderni display digitali LCD (a cristalli liquidi).
Il costo di questi ultimi è ovviamente superiore a quelli di tipo tradizionale.

PROVACIRCUITI

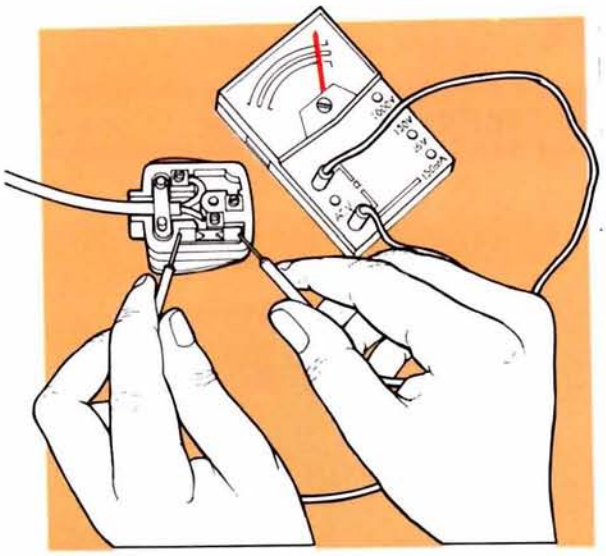


LA CALIBRAZIONE DELLO STRUMENTO



La calibrazione dello strumento
Le scale ad ago, come quella illustrata qui a fianco, devono essere dotate di un qualche sistema di calibrazione o taratura. In genere, nei multimetri, è presente una piccola vite, montata sul fulcro dell'indice. Una superficie riflettente, situata sulla scala di lettura, consente misure molto accurate, eliminando l'effetto dovuto alla parallasse. Per le misure resistive, inoltre, esiste anche un comando di taratura fine

LA CONTINUITÀ

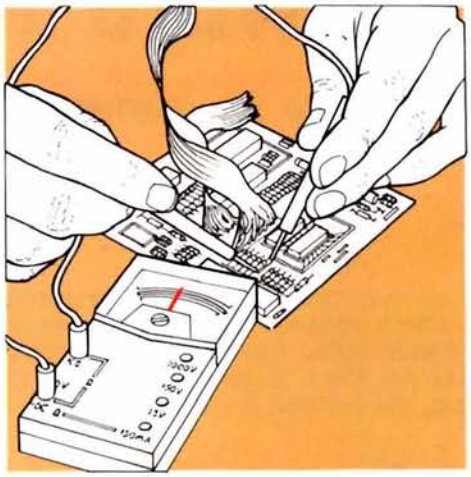


Continuità e resistenza

Uno dei più frequenti esempi di mancanza di continuità, in un circuito elettrico, avviene quando "salta" il fusibile di un'apparecchiatura. Il fusibile è appositamente progettato con lo scopo di proteggere i circuiti di un apparecchio da una corrente troppo alta (dovuta, per esempio, ad un corto circuito). Grazie a questo piccolo dispositivo, facilmente sostituibile, si possono salvare i circuiti di apparecchi molto costosi.

Un multimetro serve oltre che per eseguire verifiche sulla continuità elettrica di un circuito, alla misura della resistenza di un componente

LA RESISTENZA



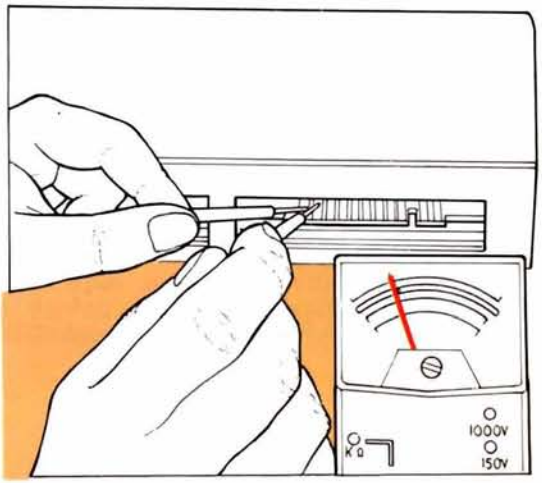
Connettore a pressione

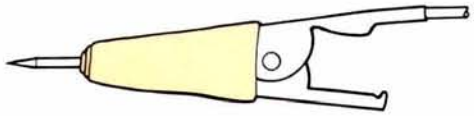
Nei computer, i connettori del tipo illustrato sono molto comuni e vengono impiegati per trasferire i segnali dal computer alle periferiche e viceversa. Alcuni apparecchi hanno un semplice connettore di questo genere. La comodità consiste nel fatto che essi formano parte integrante del circuito stampato sul quale vengono montati tutti gli altri componenti



Il voltaggio

Un utile esperimento con il multimetro può essere quello di individuare su quali piedini del connettore di un computer sia presente la tensione d'alimentazione. Consultando attentamente il manuale dell'apparecchio, si possono rilevare diverse tensioni (in genere i +5 volt ed i +9 volt). Occorre collegare il puntale negativo (quello nero) ad un punto di riferimento con potenziale 0 (massa)



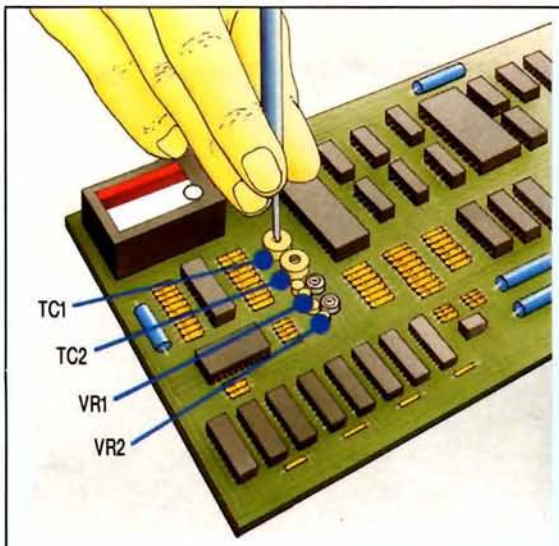


Pinza "a coccodrillo"

Una normale pinzetta a coccodrillo può esser resa più utile, nell'elettronica, se vi si fissa una piccola punta metallica (ad esempio, un chiodino) ed isolando il tutto con un rivestimento in gomma o nastro isolante

ATTENZIONE!

Generalmente, gli interventi sui computer, non operati da personale autorizzato, fanno decadere la garanzia sull'apparecchio. Si consultino attentamente le condizioni di garanzia relative al proprio computer, prima di procedere a qualsiasi tipo d'intervento



TARATURA DELLO SPECTRUM

Come abbiamo detto a pag. 530, i possessori della prima versione dello Spectrum possono aggiustare la qualità delle immagini riprodotte sullo schermo del computer. Le due resistenze variabili (dette anche "trimmer") VR1 e VR2 controllano l'emissione dei segnali rosso-verde e blu-giallo, rispettivamente. I trimmer capacitativi TC1 e TC2, invece, controllano la brillantezza e l'intensità del colore.

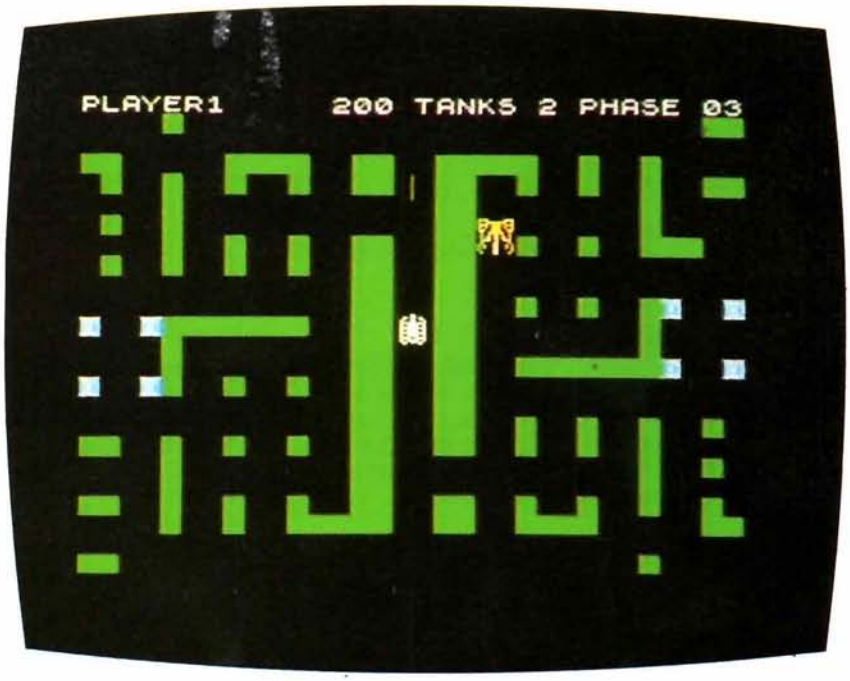
Occorre aprire l'involucro dell'apparecchio (attenzione a quanto detto qui a fianco sulla garanzia), svitando le 5 viti che fissano il coperchio.

I possessori della versione 3 dello Spectrum non possono operare né questa né altre modifiche. Per il riconoscimento delle varie versioni, si veda a pag. 530

Software in vetrina

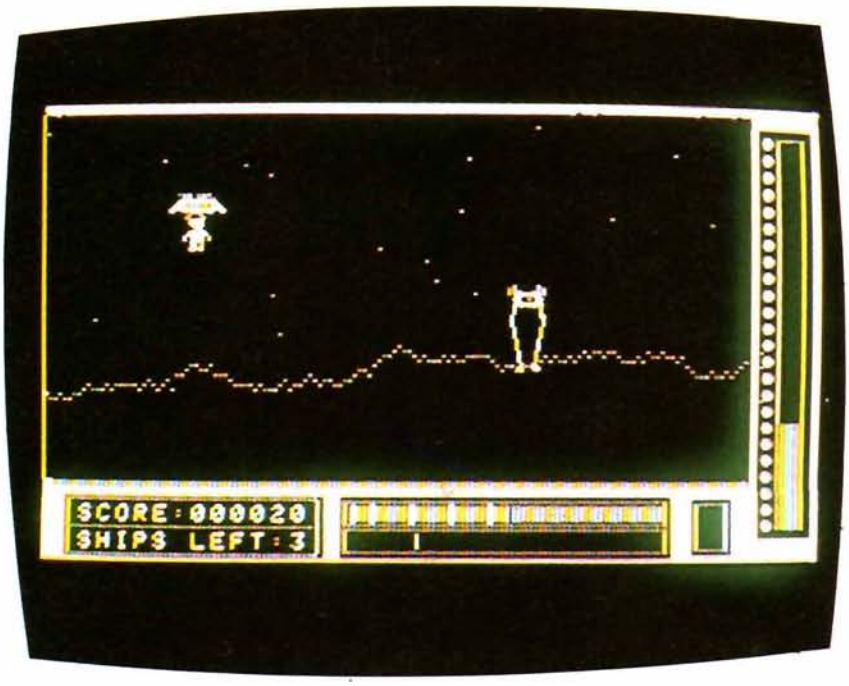
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Per gli appassionati di giochi, presentiamo una nuova selezione di computer game, adatti, questa volta, agli home computer Spectrum ed Atari. Il genere è quello, molto ampio, dei labirinti (e giochi derivati).



Brain Damage per lo Spectrum

Il Brain Damage (letteralmente "Danno al cervello") si svolge in un labirinto. Sullo schermo appare una prospettiva dall'alto del labirinto, situazione comune alla gran parte dei giochi di questa categoria. Occorre eliminare gli avversari, prima di essere da loro avvistati ed abbattuti. I nemici, in questo gioco, si chiamano il Panzer Electron, il Centurione ed il Predatore. La grafica non è particolarmente avvincente, e ciò dipende in parte dalla natura del gioco, ma l'azione diviene sempre più veloce e la mira degli avversari (purtroppo!) sempre più precisa. Il programma è prodotto dalla Silversoft (GB).



Lunar Leeper per l'Atari

Il gioco è scritto per il computer Atari, versione da 48 Kbyte e dotato di unità a dischetti. Le fasi di gioco sono due. Nella prima, si tratta di atterrare felicemente su di una base lunare, per soccorrere degli umanoidi in difficoltà. Ciò va fatto evitando gli attacchi dei predatori spaziali, che tentano di divorare la navetta di soccorso. La seconda parte del gioco si svolge all'interno di un pericoloso labirinto di caverne dalle pareti mobili. La grafica è buona e l'animazione è piuttosto scorrevole. Non è un gioco facile, tanto che spesso, sottovalutando la forza d'inerzia cui è soggetta la navetta spaziale (e quindi perdendola) non si riesce neanche ad arrivare alla seconda fase del gioco!

Il Personal dell'IBM

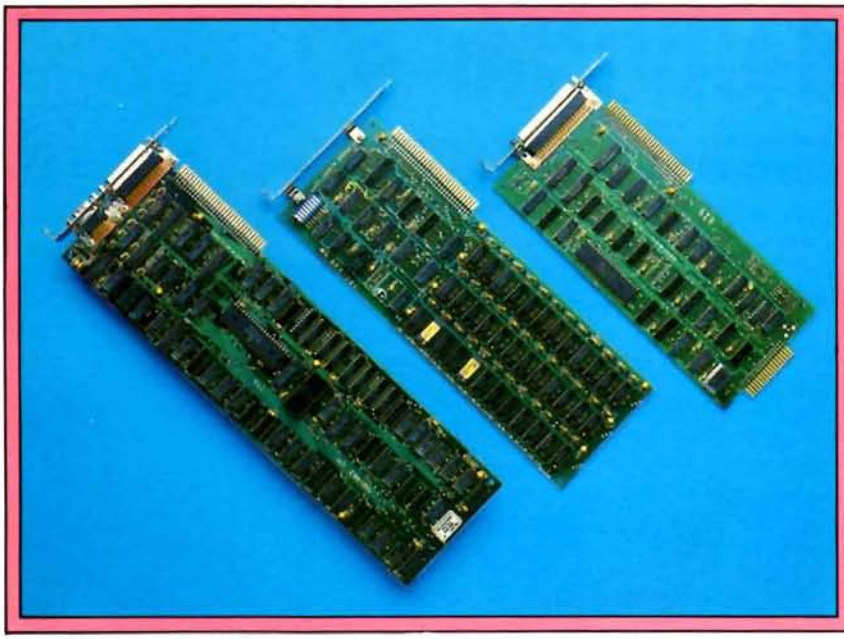
Il gigante IBM si è mosso con lentezza verso i microcomputer, ma con il tradizionale livello di qualità

Nel campo dei grandi elaboratori e dei minicomputer, la IBM non eccelle certo quale produttrice di apparecchiature particolarmente innovative. Il software IBM, sebbene ampiamente e dettagliatamente documentato, potrebbe essere sicuramente migliorato in termini di facilità d'impiego. Comunque, questa grande ditta americana è nota per la robustezza del proprio hardware ed il Personal IBM non fa eccezione, neanche per il prezzo, considerato che i prodotti IBM non sono propriamente economici. La IBM giustifica questo stato di cose spiegando che sul prezzo incide largamente l'alto livello di assistenza tecnica offerta.
Ciò lascia un poco perplessi, dato che per l'assistenza tecnica occorre stipulare un apposito contratto, che è gravato di un canone annuo...
Tuttavia, nonostante il prezzo, le vendite del Personal IBM sono considerevoli anche in Italia, dove, almeno inizialmente, il fenomeno PC IBM si è avviato con lentezza. Se il successo di un computer si può giudicare dalla quantità delle "imitazioni" (si pensi a quanto accaduto in passato per l'Apple), la IBM può ritenersi ben soddisfatta.
Le caratteristiche del PC IBM non sono tecnicamente parlando, particolarmente superiori a quelle della concorrenza: il microprocessore impiegato è un 8088, che viene considerato un chip a 16 bit. Tuttavia, in questo microprocessore tanto il bus dei dati che quello degli indirizzi sono "multiplexati" (ossia sono disponibili alternativamente sugli stessi piedini). Anche se ciò significa un risparmio di piedini sul contenitore, un simile sistema rallenta le operazioni della CPU.
Infatti, la velocità del chip 8088 è superiore di un 25% appena rispetto a quella di una CPU ad 8 bit.
Per sfruttare al massimo le potenzialità dell'apparecchio, è necessario dotare la versione base di adeguate estensioni. Prima fra tutte



CHRIS STEVENS

Design ergonomico
L'estetica e la configurazione del PC riflettono la vasta esperienza maturata dalla IBM nel settore dei macchinari per ufficio. Le tre parti principali (la tastiera, l'unità principale ed il monitor) sono separati fra loro e ciò permette di disporli per la massima comodità di lavoro. Nella versione base, il monitor è monocromo, ma, volendo, ne esiste una versione a colori



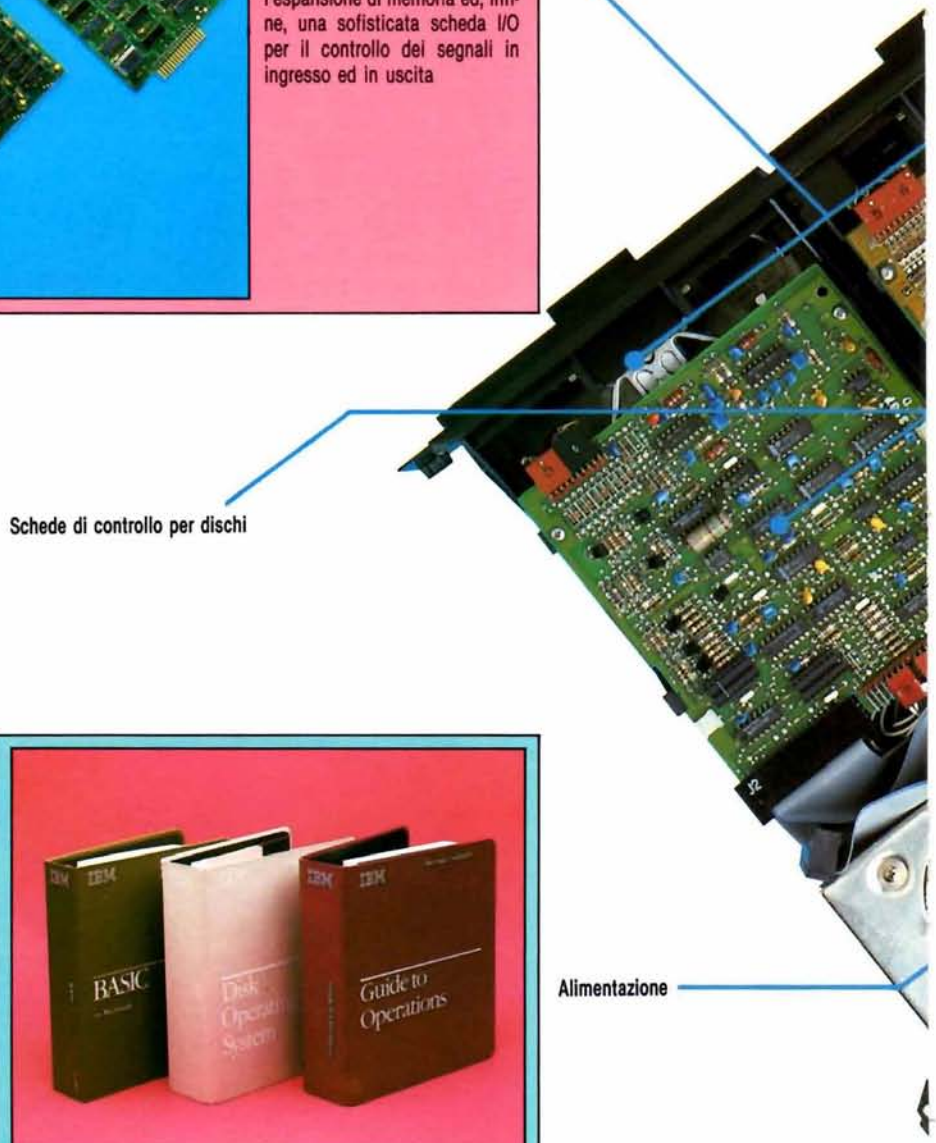
Le schede d'espansione

Nella versione di base, il PC IBM non regge il confronto con altri apparecchi della concorrenza, molto più economici. Ma l'aggiunta delle schede d'espansione, qui illustrate, innalza il livello del computer agli standard necessari nel campo commerciale. Da sinistra a destra: la scheda per grafica a colori, quella per l'espansione di memoria ed, infine, una sofisticata scheda I/O per il controllo dei segnali in ingresso ed in uscita

la memoria, che altrimenti risulta insufficiente per programmi di una certa dimensione. In secondo luogo occorre aggiungere almeno una scheda di I/O per la comunicazione con l'esterno, il cui costo non è indifferente.
Gli effetti grafici che si possono ottenere sono fantastici, ma occorre un'altra scheda aggiuntiva. Di questa esistono due versioni: a colori o monocroma. In ambedue, comunque, si tratta principalmente di un notevole blocco di memoria, dove viene memorizzata l'immagine da riprodurre, e di opportuni circuiti che generano i segnali video adatti. Le schede sono controllate dalla CPU principale, sebbene siano da poco disponibili anche delle schede video dotate di un proprio microprocessore.
Il PC possiede cinque connettori per espansioni, pertanto le schede accessorie, piuttosto che essere piccole ed economiche, tendono ad essere grandi e molto care, raccogliendo in sé quante più funzioni possibile.
Siccome le prestazioni dell'apparecchio, senza qualche espansione, sono piuttosto limitate, è necessario preventivare al momento dell'acquisto anche la spesa per le espansioni.
Naturalmente, esistono più versioni dell'apparecchio, con alcune schede già incorporate o, come nel caso della XT, addirittura dotate di un disco fisso a grande capacità. A parità di prezzo, però, viene la tentazione di orientarsi verso l'acquisto di apparecchi quali il sofisticato Lisa della Apple.
Qualcuno, evidentemente infatuato delle mode più recenti, considera il PC un computer "portatile", ma i suoi 14 chili di peso "minimo" non confermano questa convinzione. In questa versione, le normali unità a dischetti sono sostituite da una singola unità a doppia faccia, lasciando spazio sufficiente per un monitor da 9 pollici. Inoltre, la tastiera, leggermente più piccola, viene agganciata al contenitore del computer, diventandone il coperchio.

Unità a dischetti

La versione base è dotata di una sola unità a singola faccia singola densità, ma la si può sostituire con un'unità a doppia faccia doppia densità



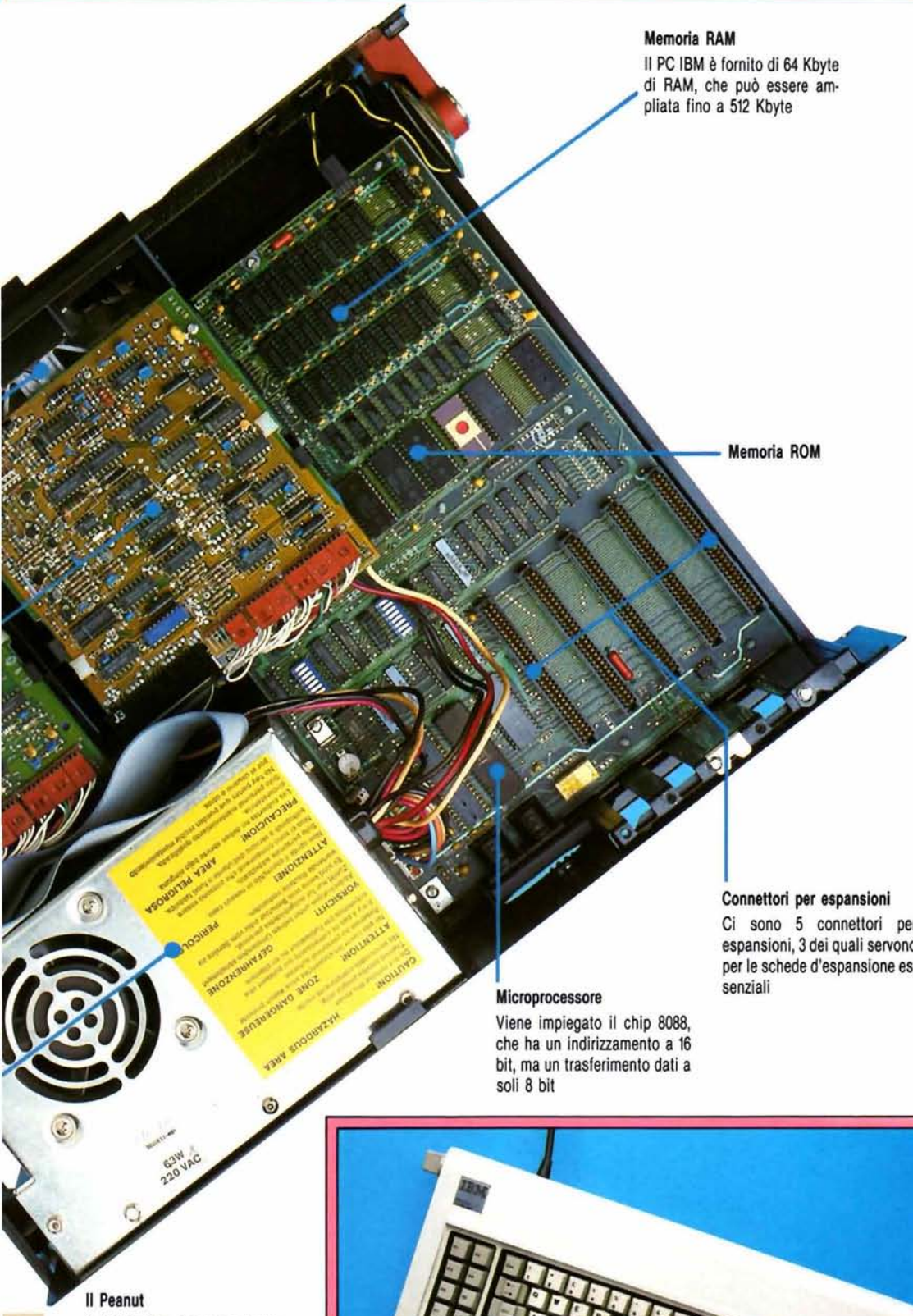
Schede di controllo per dischi

Alimentazione

IL SOFTWARE PER IL PC IBM

Uno dei principali motivi per l'acquisto di un IBM è la vasta scelta di ottimo software disponibile. Ciò spiega anche perché una grandissima fetta di fabbricanti di microcomputer tende a rendere le proprie macchine IBM-compatibili. Il sistema operativo MS-DOS è stato sviluppato dalla Microsoft, ma è possibile anche usare il CP/M-86 della Digital Research o l'UCSD-p.
Esiste anche una certa varietà di linguaggi disponibili: COBOL, FORTRAN, PASCAL e BASIC.
Per chi ama divertirsi con un giocattolo costoso, esistono anche molti giochi per il PC, sviluppati da software house statunitensi





Memoria RAM
Il PC IBM è fornito di 64 Kbyte di RAM, che può essere ampliata fino a 512 Kbyte

Memoria ROM

Connettori per espansioni
Ci sono 5 connettori per espansioni, 3 dei quali servono per le schede d'espansione essenziali

Microprocessore
Viene impiegato il chip 8088, che ha un indirizzamento a 16 bit, ma un trasferimento dati a soli 8 bit

PC IBM

DIMENSIONI
mm. 140 x 500 x 400

CPU
8088 della Intel

CLOCK
4,7 MHz

MEMORIA
64 Kbyte di RAM (estendibile fino a 512 Kbyte). 40 Kbyte di ROM

VIDEO
25 linee di 80 caratteri

INTERFACCE
Parallela Centronics, cinque connettori per espansioni

LINGUAGGI FORNITI
BASIC

ALTRI LINGUAGGI DISPONIBILI
PASCAL, COBOL, FORTRAN e altri

TASTIERA
79 tasti, di ottima fattura

DOCUMENTAZIONE
All'altezza di quanto ci si aspetta dalla IBM, che fornisce sempre descrizioni ampie e dettagliate

PRO
L'impiego del computer è facile, e la qualità del software disponibile è veramente ottima. Dotando l'apparecchio di adeguate espansioni si ottengono prestazioni notevoli

CONTRO
Principalmente il prezzo, che è decisamente superiore, a parità di prestazioni, rispetto alla concorrenza. Anche il software risulta più caro

Il Peanut
Il PC Junior, soprannominato Peanut (nocciolina) durante la fase di sviluppo, è una versione notevolmente ridotta del computer PC. La principale innovazione è il sistema di collegamento tra la tastiera e l'unità centrale, che non avviene via cavo, bensì via raggi infrarossi (come i telecomandi dei TV color). Il PC Junior dispone di due slot per cartucce



La tastiera
Come è logico aspettarsi da una delle massime case produttrici di apparecchi per ufficio, la tastiera del PC IBM è virtualmente perfetta. È separata dal corpo principale del computer, in modo da collocarla per il massimo comfort dell'utente. Ha un profilo molto basso e l'inclinazione è regolabile. Anche la disposizione dei tasti, montati con una leggera curvatura, rende questa tastiera una delle migliori in commercio

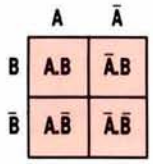
CHRIS STEVENS

Le mappe di Karnaugh

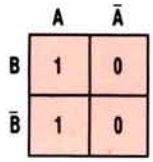
Le espressioni Booleane si possono semplificare ricorrendo a questa particolare forma descrittiva

Le "mappe di Karnaugh" (altrimenti note come le mappe-k) non sono che delle estensioni dei diagrammi di Venn, da noi già descritti in una lezione precedente (vedere a pagina 527).
Esse consentono di rappresentare graficamente delle espressioni di logica. Le mappe di Karnaugh possono assumere forme leggermente diverse, come conseguenza del numero di variabili (o lettere) coinvolte nel processo di semplificazione.
In genere la loro comodità è più evidente quando le variabili sono due, tre o quattro.

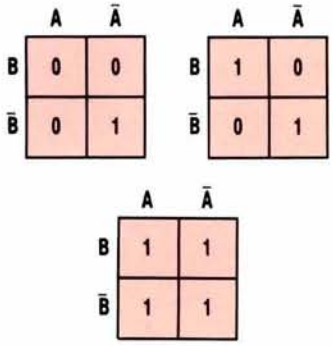
Due variabili: ogni quadrato di una mappa-k (ve ne sono $2^2 = 4$) rappresenta una funzione AND, come appare qui sotto:



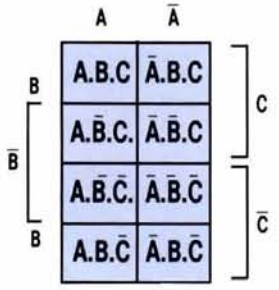
Per rappresentare l'espressione $AB+A\bar{B}$, si collocano degli 1 nei relativi quadrati:



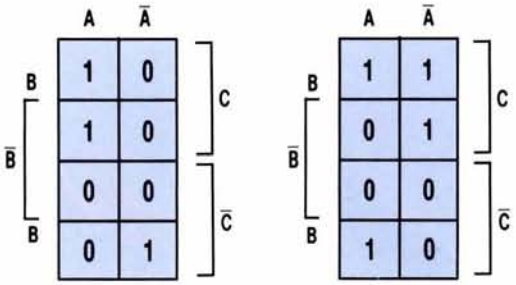
Ecco altri tre esempi di mappe-k, per rappresentare, rispettivamente: $A\bar{B}$, $AB+A\bar{B}$, ed $AB+\bar{A}B+A\bar{B}$:



Tre variabili: in questo caso, il numero di quadrati aumenta di due ($2^3 = 8$). La mappa-k di base è:



Ecco due espressioni, $AC+\bar{A}B\bar{C}$ e $AB+\bar{A}C$, rappresentate in forma di mappe-k:

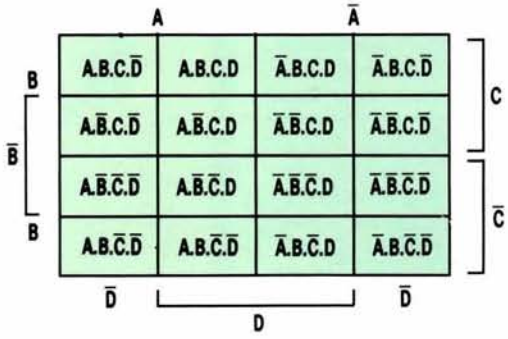


$ABC+A\bar{B}C+\bar{A}B\bar{C}$
$= AC(B+\bar{B})+\bar{A}B\bar{C}$
$= AC+\bar{A}+B\bar{C}$

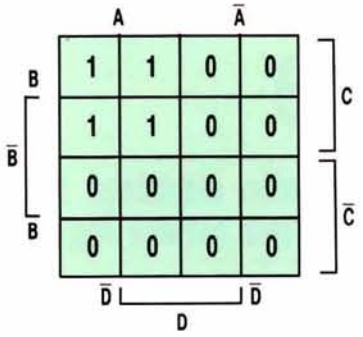
$ABC+AB\bar{C}+\bar{A}BC+\bar{A}\bar{B}C$
$= AB(C+\bar{C})+\bar{A}C(B+\bar{B})$
$= AB+\bar{A}C$

Si noti che ambedue le espressioni si sono semplificate usando la regola Booleana, in base alla quale operando un OR tra un insieme A e la sua negazione $\bar{A}$, il risultato è 1, ossia l'insieme Universale o Identità.

Quattro variabili: a questo punto le mappe cominciano a complicarsi (abbiamo $2^4 = 16$ quadrati), ma l'interpretazione di esse è ancora abbastanza facile. La mappa base è:

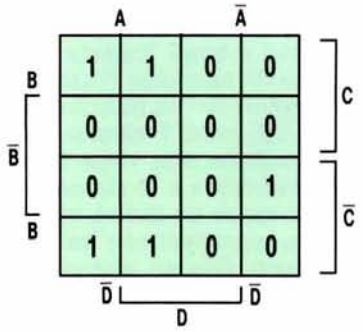


Ecco una mappa-k, con relativa espressione semplificata:



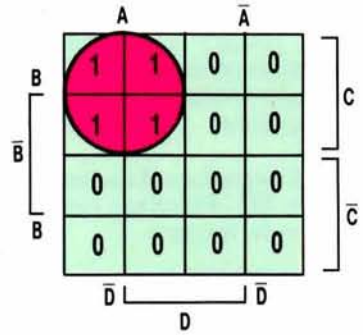
$$
\begin{aligned}
&ABC\bar{D}+ABCD+A\bar{B}C\bar{D}+A\bar{B}CD\\
&= ABC(\bar{D}+D)+A\bar{B}C(\bar{D}+D)\\
&= ABC+A\bar{B}C\\
&= AC(B+\bar{B})\\
&= AC
\end{aligned}
$$

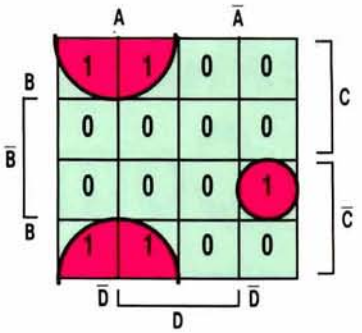
Ed ecco un nuovo esempio:



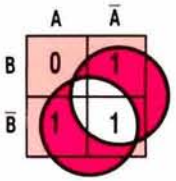
$$
\begin{aligned}
&ABC\bar{D}+ABCD+\bar{A}\bar{B}\bar{C}\bar{D}+AB\bar{C}\bar{D}+AB\bar{C}D\\
&= ABC(\bar{D}+D)+\bar{A}\bar{B}\bar{C}\bar{D}+AB\bar{C}(\bar{D}+D)\\
&= ABC+\bar{A}\bar{B}\bar{C}\bar{D}+AB\bar{C}\\
&= AB(C+\bar{C})+\bar{A}\bar{B}\bar{C}\bar{D}\\
&= AB+\bar{A}\bar{B}\bar{C}\bar{D}
\end{aligned}
$$

Se osserviamo la disposizione degli 1, nella mappa, si può scoprire la ripetizione di modelli. Nel primo esempio, in tutti i quadrati relativi alle espressioni che contengono AC, vi è un 1. Nel secondo esempio, ciò accade per le espressioni che contengono AB. Ecco come semplificare le espressioni, operando soltanto sulle mappe-k:



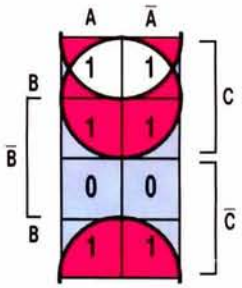


Con un poco di pratica, si riescono ad individuare facilmente i gruppi di 1, utili alla semplificazione. Si consideri l'espressione: $\bar{A}B+A\bar{B}+\bar{A}\bar{B}$.



Usando una mappa-k a due variabili, possiamo individuare due gruppi: uno relativo a tutte le NOT(B), l'altro a tutte le NOT(A), il che semplifica l'espressione a: $\bar{A}+\bar{B}$. Ma questa si può semplificare ancora, usando la legge di de Morgan, ottenendo: $\overline{A.B}$. Era possibile intravedere questa soluzione dalla mappa-k?
Ecco adesso un esempio, più complesso, a 3 variabili:

$$ABC+A\bar{B}C+\bar{A}BC+AB\bar{C}+\bar{A}B\bar{C}+\bar{A}\bar{B}C$$



Il gruppo di quattro 1, in alto, rappresenta tutti i casi nei quali C è vero. La riga più alta e quella più bassa rappresentano i casi possibili in cui B è vero. Da qui, la semplificazione: $B+C$.
Nella prossima lezione, continua l'esplorazione delle mappe di Karnaugh e vedremo come trasporle in circuiti elettrici.

Esercizio 4:
Si disegnino le mappe-k per semplificare la seguente espressione Booleana:
a) $\bar{A}.B.C+\bar{A}.\bar{B}.C+\bar{B}.\bar{C}+\bar{A}.B.\bar{C}$
b) $A.\bar{B}.\bar{C}+\bar{A}.B.\bar{C}+A.B.\bar{C}$

Il BASIC dei Commodore

Questa versione di BASIC non corrisponde allo standard Microsoft, tuttavia è comoda e dotata di una semplice, ma ottima logica

Nel BASIC di tutte le macchine Commodore sono consentiti nomi di variabili composti da più caratteri, dei quali, però, sono significativi soltanto i primi due. Ciò significa che FATTURA e FASCIA sono nomi ammessi, ma equivalenti, quindi il risultato di:

```
100 FATTURA = 17:FASCIA = 2*FATTURA
200 PRINT FATTURA,FASCIA
```

è:

```
34   34
```

Ciò vale per tutti i tipi di variabili: a virgola fluttuante (ad esempio NUMERO), intere (NUMERO%), stringhe (NUMERO$) e matrici (NUMERO(62,47)). I tipi delle variabili sono convenzionali, ma sul Vic-20 non si possono usare variabili intere, poiché il BASIC non è dotato del modulo per l'aritmetica intera. Tuttavia, il tipo di variabile intera esiste, per compatibilità con gli altri apparecchi.
Una fastidiosa conseguenza di questo sistema è che può facilmente accadere di includere sigle riservate nei nomi di variabili (ad esempio START equivale alla parola riservata ST).
Le variabili a matrice possono avere fino a 255 dimensioni e sono limitate unicamente dalla quantità di memoria RAM disponibile. Il primo elemento di una matrice ha indice 0, pertanto DIM EX(6) crea 7 elementi: EX(0), EX(1)... EX(6). Se una matrice non viene DIMensionata, il BASIC la considera a 10 elementi. Qualora tentassimo di usare un indice maggiore, l'errore verrebbe segnalato con un BAD SUBSCRIPT (= indice errato). Questo metodo è comodo, ma un buon programmatore dimensiona sempre le matrici, sia per chiarezza del listato, sia per una migliore gestione della memoria da parte dell'interprete. Raggruppare tutte le DIM all'inizio del programma può servire a rendere l'esecuzione del programma più spedita.
Per come è strutturata la gran parte dei BASIC, un metodo per velocizzare l'esecuzione è quello di inizializzare le variabili, in ordine d'importanza, all'inizio del programma. Una linea come:

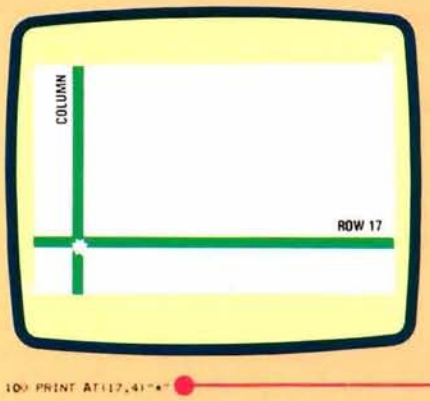
```
10 DIM A$(10,24),K,L,PUNTI
```

apparentemente non ha alcun effetto, ma consente di piazzare le variabili K, L e PUNTI all'inizio della tabella delle variabili, facilitandone la ricerca.
Se facciamo un confronto con lo standard Microsoft, il BASIC della Commodore ha qualche funzione in meno e qualcuna in più. L'omissione più evidente riguarda la INKEY$, mentre le aggiunte più significative sono TIME$ e STATUS.
La funzione di scansione della tastiera, INKEY$, è ottenuta con GET, d'impiego pressoché identico. Un esempio tipico è:

```
150 GET GT$:IF GT$ = " " THEN 150
```

dove l'esecuzione si arresta, in attesa che

IL POSIZIONAMENTO DEL CURSORE

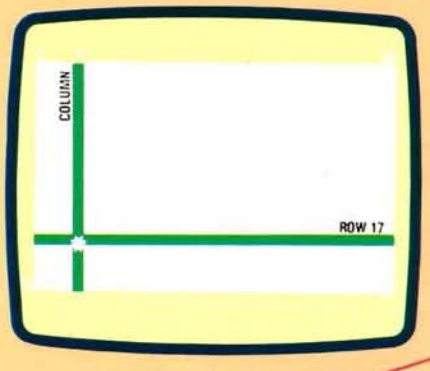


La possibilità di specificare il posizionamento del cursore mediante stringhe alfanumeriche, sui Commodore, consente una facile gestione della grafica

Se il BASIC della Commodore avesse il comando PRINT AT, forse la procedura sarebbe più immediata

```
100 PRINT AT(17,4)"*"
```

Ma poiché tale comando non esiste, è necessario ricorrere ad un altro sistema:



si inizializza POSIZIONE$ con un carattere HOME e 24 caratteri CURSORE IN BASSO. Poi si inseriscono opportuni parametri nell'espressione

Se il posizionamento del cursore è una funzione richiesta con frequenza, allora conviene scrivere una piccola subroutine adatta allo scopo. In questo caso, a RIGA e COLONNA vengono assegnati i valori richiesti, prima della chiamata alla subroutine

```
50 POSIZIONE$="■■■■■■■■■■■■■■■■■■■■■■■■"
100 PRINT LEFT$(POSIZIONE$,17)TAB(4-1)"*"

50 POSIZIONE$="■■■■■■■■■■■■■■■■■■■■■■■■"
100 RIGA=17:COLONNA=4:GOSUB 1000:PRINT "*"
500 END
1000 PRINT LEFT$(POSIZIONE$,RIGA)TAB(COLONNA-1);:RETURN
```

LIZ DIXON

venga premuto un qualsiasi tasto. Quando ciò avviene, GT$ contiene il carattere corrispondente al tasto. Tuttavia, ciò può non essere sempre vero, poiché la GET scandisce il buffer della tastiera e non quest'ultima direttamente. Quindi, se il buffer contiene un carattere, la GET lo acquisisce immediatamente, e non sospende l'esecuzione. Ecco una dimostrazione di questo fenomeno:

```
 50 FOR K=1 TO 100:PRINT K:NEXT K
 60 PRINT "PREMERE QUALSIASI TASTO"
150 GET GT$:IF GT$=" " THEN 150
200 PRINT "IL TASTO PREMUTO ERA ";GT$
```

Se si lancia il programma, sullo schermo compaiono i numeri da 1 a 100, poi l'esecuzione si arresta, in attesa di un carattere da tastiera. Ma se durante la visualizzazione dei numeri si preme qualche tasto (il cui codice viene depositato temporaneamente nel buffer), non ha luogo nessuna pausa, ed il programma passa direttamente alla linea 200. Ciò può essere particolarmente fastidioso nei giochi, spesso basati sull'impiego di GET. Per ovviare, basta azzerare il buffer subito prima della GET:

```
149 POKE KBPTR,0
```

dove KBPTR contiene l'indirizzo del contatore di caratteri della tastiera (sul Commodore 64 questo è 198).
La GET, normalmente, non visualizza in modo lampeggiante il cursore. Ciò si può ottenere con POKE FLASH,0 (dove FLASH contiene l'indirizzo della locazione RAM riservata a questo scopo: 204 per il Commodore 64).
Mediante la variabile TIME$ (abbreviata in TI$) si accede all'orologio del sistema. All'accensione esso viene azzerato, dopodiché indica le ore, i minuti ed i secondi trascorsi dall'accensione, fino al valore massimo "235959". Il programmatore può inserire qualsiasi orario valido, usando TI$ come una normale variabile, ad esempio TI$="000000" oppure TI$="084503".
Associata a TI$ è la variabile numerica TI, che contiene il tempo trascorso dall'accensione in 60mi di secondo (chiamati "jiffy"). L'arco di valori si estende da 0 a 5183999 (=60*60*60*24−1). Su TI non si interviene direttamente, ma solo attraverso TI$.
La STATUS (abbreviata in ST) è una variabile cosiddetta "di sistema". Quando il BASIC incontra un errore di I/O, in essa viene depositato il codice corrispondente. Un impiego tipico è questo:

```
330 IF ST>0 THEN GOSUB 30000: REM GESTIONE DEGLI
    ERRORI DI I/O SUI FILE
:
:
30000 REM MESSAGGI D'ERRORE
30100 IF ST=16 THEN PRINT"ERRORE FATALE DI LETTURA"
```

CMD è un utile membro delle istruzioni del Commodore. Il suo effetto consiste nel ridirezionare l'output, che altrimenti andrebbe sullo schermo, verso qualche altro dispositivo. Ad esempio:

```
OPEN 4,4:CMD 4:LIST
```

serve per LISTare il programma su stampante, anziché sullo schermo. Ottenuta la stampa, occorre richiudere il canale di I/O:

```
PRINT #4:CLOSE 4
```

CMD è utile anche per copiare su stampante il contenuto dello schermo. Se un sottoprogramma alla linea 3000 provoca la visualizzazione di un certo messaggio sullo schermo (mediante PRINT e non con delle POKE!), si può ottenerne la stampa con:

```
OPEN 4,4:CMD 4:GOSUB3000:PRINT#4:CLOSE4
```

Benché il punto interrogativo (?) si possa usare come abbreviazione del comando PRINT, non si può abbreviare PRINT# con ?#. Si usa invece pR (p seguito da SHIFT R).
L'abbreviazione dei comandi sugli apparecchi Commodore è di gran lunga precedente alla corrispondente funzione dei computer Sinclair. Quasi tutte le parole chiave si possono abbreviare usando i primi due caratteri delle istruzioni (il primo minuscolo il secondo maiuscolo). Se due istruzioni hanno i due caratteri in comune, si usa anche il terzo. Ad esempio: READ, RESTORE e RETURN, vengono abbreviati in rE, reS e reT.
Sia l'editor che il sistema operativo delle macchine Commodore sono programmi compatti e di facile impiego, un punto di forza dell'intera gamma. Già presente nei PET, questa filosofia viene ancora adottata per gli apparecchi più recenti. Per apportare modifiche ad una linea di programma, ad esempio, basta LISTare la linea e spostare il cursore nella posizione voluta.
Terminata la correzione, si batte il tasto RETURN. Più semplice di così non può davvero essere.
Un altro vantaggio non indifferente di questo ottimo editor, è la possibilità di copiare facilmente intere linee di programma. Supponiamo di dover immettere queste due linee:

```
100 IF INT (NUMERO/INDICE−TASSO)=5 THEN 3000
200 IF INT (NUMERO/INDICE−TASSO)=7 THEN 3800
```

Si scrive la prima linea e si batte RETURN. Poi, si sposta il cursore sul numero di linea, lo si cambia in 200, si cambia 5 con 7 e 3000 con 3800. Si batte RETURN ed il gioco è fatto: la linea 100 rimane inalterata, ma adesso abbiamo anche la linea 200! Questo processo può essere ripetuto quante volte si vuole, facilitando l'immissione di parti ripetitive del programma.
Senza contare che, se le espressioni sono di una certa complessità, il rischio di commettere errori è minimo.
Questo è soltanto uno dei piccoli trucchi disponibili sull'editor dei Commodore, ma serve bene a dimostrare la facilità e l'immediatezza del suo impiego.

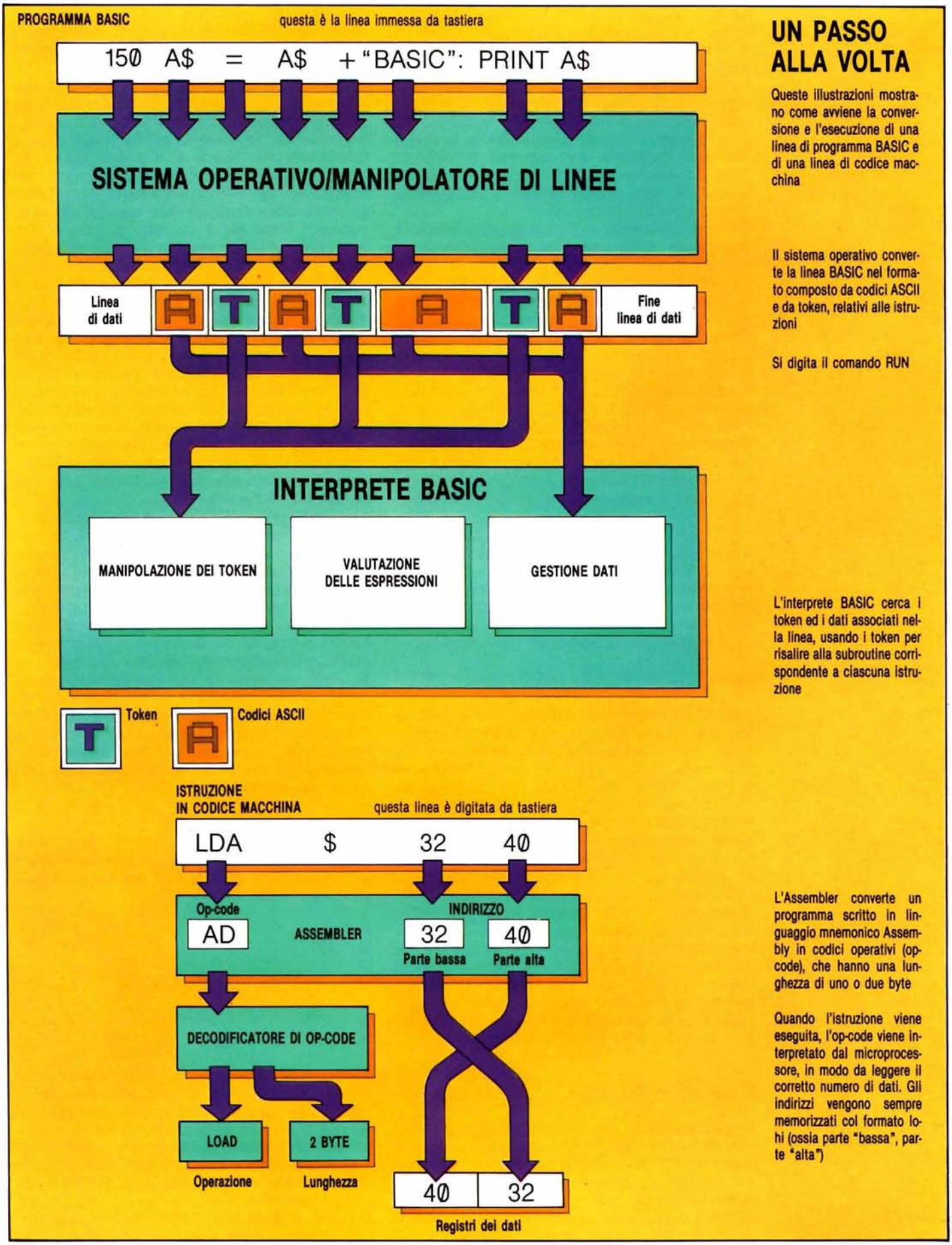
Alcune utili convenzioni

Volendo imparare il codice macchina occorre abituarsi all'uso di token, di puntatori e di op-code

Quando si esegue un programma, la prima azione del sistema operativo è quella di esaminare i puntatori all'inizio del testo del programma BASIC per stabilire dove risieda il codice da eseguire. Questo, che può sembrare un laborioso metodo di lavoro, è necessario per dare flessibilità a tutto il sistema. Si ricordi che il sistema operativo è un programma permanente, conservato generalmente nelle ROM, ed i dati in esso contenuti (quali, ad esempio, gli indirizzi di memoria) sono altrettanto permanenti. Si supponga, per un momento, che vengano prodotte diverse versioni di un computer in un breve arco di tempo. Nella prima può essere conveniente, magari, che l'indirizzo iniziale del testo BASIC sia 2048. Nella seconda versione, però, è necessario spostare questo indirizzo a 4096. Ciò impedirebbe di adoperare il medesimo sistema operativo sulle due macchine, perché il testo non risiede nelle stesse locazioni di memoria. Di conseguenza, andrebbe prodotta una nuova serie di ROM per ciascuna versione, aumentando i costi e rischiando, inoltre, che i programmi sviluppati per la versione 1 non siano compatibili con la versione 2. Ma se il sistema operativo contiene soltanto gli indirizzi dei puntatori al testo, questo problema non si presenta affatto. Le locazioni di questi puntatori possono rimanere invariate, all'interno della piccola zona di RAM riservata al sistema (tipicamente 1 Kbyte). Mantenere fissa la posizione di questo blocco di memoria (nella prima pagina della RAM, ad esempio) e costruirci attorno tutto il sistema non rappresenta un grosso ostacolo per i progettisti, mentre lo sarebbe dover fissare definitivamente l'area di RAM dedicata al testo BASIC.

La pratica corrente

Una diffusa convenzione riguarda il formato col quale gli indirizzi vengono conservati in memoria e viene indicata col termine "lo-hi" (ossia basso-alto). Se i byte 43 e 44 devono contenere l'indirizzo 7671 (pagina 29, offset 247), allora il byte 43 contiene 247 (offset o byte "lo"), mentre il byte 44 contiene 29 (pagina o byte "hi").
Inizialmente, questo metodo induce una certa confusione, ma non nella CPU, che anzi lo preferisce. Dopo tutto, è anche logico che la parte "bassa" dell'indirizzo sia nel byte con indirizzo più "basso" e, analogamente, lo stesso avvenga per la parte "alta" dell'indirizzo.
Ripetendo l'esempio, ma usando questa volta i corrispondenti valori esadecimali, si può apprezzare la comodità di questo secondo sistema rispetto al decimale. D'ora in poi, indicheremo dei valori hex (esadecimali) usando il suffisso $. I byte che servono da deposito sono dunque $2B e $2C, mentre l'indirizzo da conservare è $1DF7. Pertanto, $2B contiene $F7 (la parte "bassa"), mentre $2C contiene $1D (la parte "alta"). Si noti come la scomposizione in byte sia più omogenea usando la notazione hex. Va notata una curiosa eccezione a questa convenzione, nel Micro BBC e nello Spectrum. Qui i numeri di linea dei programmi BASIC vengono conservati col formato inverso, cioè "hi-lo". Anche se detti valori sono parametri e non propriamente degli indirizzi, rimane pur sempre un metodo abbastanza ibrido.
Un'altra convenzione, concernente gli indirizzi, è che spesso viene indicata soltanto la parte bassa. Ad esempio, si può dire che nel Commodore, il puntatore all'area riservata al testo BASIC si trova all'indirizzo 43. Con ciò si intende, in effetti, la coppia di byte 43 e 44, trattandosi di un puntatore che deve necessariamente avere due byte.
Altri elementi da considerare sono i "token" (vedere a pag. 556). Il significato di questi codici, per il programmatore, è duplice: essi costituiscono un'abbreviazione di un comando o istruzione BASIC (quali PRINT o RESTORE) composti da più caratteri, ma essi usano anche un offset. Un comando BASIC, anche se indicato da una sola parola, implica tutta una serie di operazioni. Il comando PRINT, ad esempio, implica l'individuazione, nella memoria, di una serie di caratteri da visualizzare e l'invio di un carattere alla volta al circuito di visualizzazione. Queste varie funzioni sono compito delle diverse subroutine all'interno dell'interprete BASIC. Quando esso incontra il token per il comando PRINT, lo usa per calcolare l'indirizzo della corrispondente subroutine.
Supponiamo di avere una versione di BASIC nella quale esistano soltanto tre comandi: INPUT, PRINT e STOP. A questi vengono associati, rispettivamente, i token: $80, $81 e $82. Inoltre, supponiamo che le corrispondenti subroutine inizino agli indirizzi: $D010, $EA97 e $EC00. Questi indirizzi sono depositati in una tabella di sei byte, nel formato lo-hi, a partire dall'indirizzo $FA00.



UN PASSO ALLA VOLTA

Queste illustrazioni mostrano come avviene la conversione e l'esecuzione di una linea di programma BASIC e di una linea di codice macchina

Il sistema operativo converte la linea BASIC nel formato composto da codici ASCII e da token, relativi alle istruzioni

Si digita il comando RUN

L'interprete BASIC cerca i token ed i dati associati nella linea, usando i token per risalire alla subroutine corrispondente a ciascuna istruzione

L'Assembler converte un programma scritto in linguaggio mnemonico Assembly in codici operativi (op-code), che hanno una lunghezza di uno o due byte

Quando l'istruzione viene eseguita, l'op-code viene interpretato dal microprocessore, in modo da leggere il corretto numero di dati. Gli indirizzi vengono sempre memorizzati col formato lohi (ossia parte "bassa", parte "alta")

KEVIN JONES

Adesso, quando il nostro ipotetico interprete incontra un token ($81 ad esempio), sottrae da esso $80, moltiplica il valore ottenuto per 2 e somma il risultato a $FA00. In questo caso, l'indirizzo che ne deriva è $FA02, la posizione nella tabella dove è conservato l'indirizzo della subroutine PRINT. Si provi a ripetere il procedimento con gli altri token e si vedrà che esso funziona con estrema precisione.
L'uso di token è diffuso in quasi tutti gli interpreti, ma spesso anche i compilatori ricorrono a sistemi simili.
Un sistema basato sull'impiego di token individua una delle principali differenze tra il linguaggio macchina ed i cosiddetti linguaggi ad "alto livello". I comandi del BASIC sono ragionevolmente comprensibili, anche se appartengono alla lingua inglese. Ma l'insieme di codici ASCII, di puntatori e di token è materia pertinente alla CPU, che ci si muove a suo completo agio.
L'ultimo aspetto da considerare, nella manipolazione della memoria, è il concetto di "contesto". Abbiamo già visto che l'area di memoria riservata al testo BASIC è costellata da tutta una serie di codici: caratteri in formato ASCII, numeri in formato binario, token per i comandi, ecc. (vedere a pagina 558).
Tutti questi codici sono compresi nell'arco di numeri binari da 00000000 a 11111111 (o da $00 a $FF in esadecimale, da 0 a 255 in decimale), depositati in singoli byte della memoria ed interpretati in base al loro "contesto". Nell'area riservata al testo BASIC di un Commodore, ad esempio, la seguente linea BASIC:

```
200 rem ••••••••••••••••••••••left$••••••••••••••••••••••
```

potrebbe avere tre diversi byte con il medesimo valore 200. Uno di essi potrebbe essere nella parte bassa dell'indirizzo di LINK, un altro nella parte bassa del numero di linea ed il terzo, infine, nel token che rappresenta l'istruzione left$. Il valore è identico, ma è l'interpretazione che cambia. Ed essa è funzione della posizione di ciascuno dei tre byte all'interno della linea di programma.
In effetti, ciò si ricollega a quanto detto nell'introduzione al corso di linguaggio macchina: tutte le informazioni, in un computer, vengono conservate sotto forma di codice numerico. Alcune di queste forme ci sono già familiari (come il codice ASCII), altre meno (come i token). Di altre ancora non si è nemmeno parlato (i codici operativi del linguaggio macchina). Passiamo dunque a trattare proprio quest'ultimo argomento.

I codici operativi

I programmi in codice macchina non sono che sequenze, più o meno lunghe, di valori conservati in locazioni di memoria consecutive. Alcuni di questi valori rappresentano delle istruzioni, altri, invece, costituiscono dati sui quali deve operare il microprocessore. Come accennato precedentemente, è il contesto che permette di separare gli uni dagli altri. Dobbiamo quindi considerare, in primo luogo, il formato dei codici operativi che formano i programmi.
Un'istruzione in codice macchina inizia con un codice identificativo dell'operazione da svolgere. Questo codice è chiamato anche, brevemente, "op-code" o anche "opc". La sua lunghezza può essere di uno o di due byte. L'op-code può essere autosufficiente, nel senso che l'operazione da esso svolta non richiede la specifica di alcun dato, ma, più frequentemente, l'op-code è seguito da uno o due byte di dati. I dati composti da un solo byte sono generalmente delle costanti numeriche o dei codici ASCII, mentre se la lunghezza è di due byte, si tratta probabilmente di un indirizzo (sempre espresso nel formato lo-hi).
Siamo appena giunti faticosamente ad una definizione, ed ecco che saltano subito fuori le diversità tra i vari apparecchi: il Micro BBC usa il chip 6502A, il Commodore 64 usa il chip 6510 (molto simile al 6502, per cui tratteremo, in seguito, solo quest'ultimo), mentre lo Spectrum impiega lo Z80A della Zilog. Questi chip sono stati realizzati pressappoco durante lo stesso periodo (agli inizi degli anni '70), quindi l'architettura interna dei due microprocessori non differisce molto. Purtroppo, però, l'insieme delle istruzioni e lo stesso funzionamento dei due tipi di chip hanno rarissimi punti in comune. Ad esempio, tutti gli op-code del 6502 sono di un solo byte, e la lunghezza dei dati è di uno o di due byte.
Nello Z80, viceversa, gli op-code possono essere composti da due byte, seguiti a loro volta, eventualmente, dai byte dei dati.
Quando l'op-code di una particolare operazione viene letto dalla CPU, il sistema di decodifica permette di comunicare alla CPU il tipo di operazione associata al codice e l'eventuale lunghezza dei dati che devono seguire l'op-code. Ad esempio la seguente sequenza di byte, per il 6502:

```
A9  0E  8D  01  4E  60  44  52  41  54
```

rappresenta tre istruzioni, seguite da quattro byte di codice ASCII. Per nostra comodità possiamo riscriverli così:

```
A9 0E
8D 01 4E
60
44
52
41
54
```

dove meglio si apprezza che l'op-code A9 è seguito da un byte di dati, l'op-code 8D è seguito da due byte di dati, mentre l'op-code 60 non necessita di alcun dato. Se al microprocessore inviamo il valore A9, quando esso è in attesa di un op-code, allora tutto procede regolarmente. Le informazioni derivate da ciascun op-code indicano quanti byte di dati seguono e dove si trova l'op-code successivo. Ma, qualora inviassimo invece il secondo

byte, ossia 0E, il microprocessore lo interpreterebbe come op-code e la sequenza verrebbe così interpretata come:

```
0E 8D 01
4E 60 44
52
```

ossia: l'op-code 0E richiede 2 byte di dati, l'op-code 4E richiede 2 byte di dati, ma 52, che non è un op-code, causa l'equivalente di un SYNTAX ERROR del BASIC. Ciò dimostra quanto sia importante comunicare con precisione l'indirizzo iniziale di un programma alla CPU.
Inoltre, dimostra anche quanto la programmazione in codice macchina sia ostica per chi è alle prime armi: la macchina, dopo tutto non fa che eseguire ciecamente i codici da noi forniti. Se l'ordine non è corretto (e nei programmi di una certa dimensione possono facilmente sfuggire banali errori di trascrizione) il risultato sarà un fiasco completo, senza nemmeno darci la benché minima indicazione su quale codice abbiamo sbagliato!
Parte di queste difficoltà si possono superare usando, durante la fase di sviluppo dei programmi, una notazione mnemonica composta da lettere, anziché da numeri, convertendo il tutto in valori numerici soltanto nell'ultimo stadio. Questa notazione mnemonica non è altro che il linguaggio Assembly del particolare microprocessore. Il processo di conversione, dal codice mnemonico a quello numerico si chiama "assemblare" un programma. Si noti che vi è una corrispondenza univoca tra i codici mnemonici e quelli numerici.
Se adesso riscriviamo il frammento di programma, usato in precedenza, in linguaggio Assembly 6502, otteniamo:

```
0000 A9 0E      LDA #$0E
0002 8D 01 4E   STA $4E01
0005 60         RTS
```

La medesima sequenza di operazioni, in Assembly Z80, risulta invece:

```
0000 3E 0E      LD A, $0E
0002 32 01 4E   LD ($4E01),A
0005 C9         RET
```

La prima colonna mostra l'indirizzo esadecimale delle locazioni di memoria per il primo byte di ciascuna linea. La successiva colonna contiene uno, due o tre byte e rappresenta il codice macchina. Nell'ultima colonna, infine, è riportato, in Assembly, l'equivalente del codice macchina per ciascuna riga.
Si noti il diverso formato hi-lo e lo-hi adoperato per gli indirizzi, a seconda che si usi l'Assembly o il codice macchina.
Comunque, non ci si soffermi troppo ad esaminare ogni singola istruzione, l'importante è afferrare il concetto generale del linguaggio Assembly e delle differenze dovute all'impiego dei due chip citati.
Nella prossima lezione, approfondiremo la nostra conoscenza con gli op-code e con l'architettura interna del microprocessore.

CONVERTITORE ESADECIMALE

Per far sì che il programma Mempeek, presentato a pagina 539 visualizzi il contenuto dei byte in forma esadecimale, anziché decimale, si apportino le seguenti modifiche:

MICRO BBC

Aggiungere:

```
3000 DEF PROCHXPRINT(DECNUM)
3100 LOCAL X$
3200 X$="0123456789ABCDEF"
3300 HB=INT(DECNUM/16):LB=DECNUM-HB*16
3400 B$=MID$(X$,HB+1,1)+MID$(X$,LB+1,1)+"   "
3500 PRINT B$;
3600 ENDPROC
```

e cambiare la linea 600 in:

```
600 PROCHXPRINT(PK%)
```

SPECTRUM

Aggiungere:

```
10 LET X$="0123456789ABCDEF"
3000 REM **** SUBROUTINE PER ESADECIMALE ****
3100 LET HB=INT(PK/16):LET LB=PK-HB*16
3200 LET B$=X$(HB+1)+X$(LB+1)+"   "
3300 PRINT B$;
3400 RETURN
```

e cambiare la linea 600 in:

```
600 GOSUB 3000
```

COMMODORE 64

Aggiungere:

```
10 LET X$="0123456789ABCDEF"
3000 REM **** SUBROUTINE PER ESADECIMALE ****
3100 HB=INT(PK/16):LB=PK-HB*16
3200 B$=MID$(X$,HB+1,1)+MID$(X$,LB+1,1)+"   "
3300 PRINT B$;
3400 RETURN
```

e cambiare la linea 600 in:

```
600 GOSUB 3000
```

Queste modifiche consentono la visualizzazione dei valori secondo il formato esadecimale, ma l'indirizzo iniziale e quello finale devono essere forniti in notazione decimale

Entra in scena la Xerox

L'escalation di una grossa corporazione: dalle fotocopiatrici alla rete informatica Ethernet

Agli inizi degli anni '70, la Xerox avviò un ambizioso progetto a lungo termine per la realizzazione di quello che allora sembrava un sogno: dotare l'ufficio moderno di una fonte d'informazioni, allacciata più o meno, come la normale corrente od il telefono. Al gruppo di ricerca, la cui base operativa si trovava a Palo Alto, in California (USA), venne data "carta bianca", ossia la più completa libertà d'azione.

Il trasferimento a Palo Alto, così distante dalla casa "madre", nel New Hampshire, fu particolarmente felice.

Vicino al "campus" dell'Università di Stanford, nel quale esisteva un produttivo dipartimento informatico di ricerca sull'Intelligenza Artificiale, il Centro di Ricerca della Xerox a Palo Alto (spesso indicato con la sigla PARC, da Palo Alto Research Center), riuscì a carpire più validi elementi da inserire nel proprio organico.

Il PARC divenne un polo attorno al quale gravitava tutta la cultura informatica del momento. Il linguaggio usato al centro era un tale misto di termini gergali, da essere incomprensibile ai "non addetti". A molti dei prodotti Xerox, durante la fase di sviluppo, vennero affibbiati nomignoli curiosi. Ad esempio, per la serie di apparecchi 820 venne usato "Worm" (baco), forse con la segreta speranza che esso avrebbe presto divorato la Mela (l'Apple)!

Il principale obiettivo del gruppo di lavoro era quello di realizzare una rete locale (Local Area Network o LAN). Oggi questa sigla si è diffusa enormemente, ma, quando la Xerox effettuava i primi esperimenti, alla fine degli anni '60, il concetto era addirittura rivoluzionario. I maggiori problemi incontrati riguardavano il collegamento fisico tra i terminali e l'unità centrale: la massima lunghezza dei cavi non superava la ventina di metri; del resto, i commutatori delle linee telefoniche non consentivano una velocità di trasmissione superiore ai 9600 baud. Ancora troppo poco.

L'obiettivo era di creare una rete informativa, basata sull'impiego di microcomputer, con la possibilità di accedere a banche dati di vaste proporzioni e di scambiare dati tra i vari utenti. Era il concetto di base del LAN Ethernet.

In questo sistema, i collegamenti vengono fatti con normali cavi coassiali, consentendo un traffico di 10 milioni di bit al secondo, non solo di dati, ma anche di suoni e di informazioni grafiche digitalizzate. Inoltre, distanze fino ai 500 metri non richiedono l'impiego di ripetitori amplificati. L'allacciamento di un nuovo terminale non presenta problemi: basta un connettore, consentendo così la massima flessibilità d'uso.

La rete fisica di per sé è passiva: i dati, di qualunque genere, vengono fatti circolare nella rete e trascodificati da opportuni dispositivi collegati a ciascun apparecchio. Ciascun dispositivo è in grado di stabilire se una particolare informazione è destinata all'apparecchio ad esso collegato o no.

Alla metà degli anni '70, la rete Ethernet era funzionante e la Xerox pensò che se fosse riuscita a coinvolgere altri fabbricanti del settore, l'Ethernet sarebbe potuta divenire uno standard di comunicazione tra computer.

La IBM non si dimostrò interessata, mentre la Digital Equipment Corporation (DEC), e più tardi la Intel (produttrice di una serie di speciali chip adatti alla comunicazione), aderirono al progetto.

La rete Ethernet venne realizzata a scopo sperimentale all'interno di un complesso commerciale, in Svezia. Successivamente, sono state installate con successo altre numerose reti. Di fatto, il sistema Ethernet è divenuto uno standard, come era nelle speranze della Xerox: la ICL in Gran Bretagna, la Siemens in Germania, la Olivetti in Italia hanno tutte adottato questo sistema di comunicazione.

Oggi tutti i nuovi prodotti della Xerox, dalle fotocopiatrici ai computer ed alle stampanti laser (un'altra invenzione del PARC), possono essere collegati all'Ethernet.



CHRIS STEVENS

Un precursore del Lisa

Uno dei maggiori trionfi del PARC è stato la realizzazione di STAR, un sistema di programmazione basato sul linguaggio SMALLTALK. Lo STAR opera combinando programmi e dati nello stesso file: un sistema molto simile a quello in seguito adottato dal Lisa della Apple. Effettivamente, l'équipe che ha sviluppato il Lisa proviene in gran parte dal PARC



Facile accesso

Un esame del Micronet 800, uno dei primi servizi per lo scambio di software a disposizione di chi ha un home computer



IAN McKINNELL

Il Micronet 800 è uno dei tanti servizi offerti dal sistema di video-informazione britannico Prestel. Per le sue particolari caratteristiche e per il fatto che si tratta di uno dei primi esperimenti del genere in Europa, vale la pena di analizzarlo attentamente.

Lo scopo del Micronet è di consentire l'accesso, da parte dei possessori di home e personal computer, ad una vasta "banca" di software e ad un sistema di "posta elettronica". Un aspetto molto interessante è che, in molti casi, i programmi del circuito Micronet possono essere usati e copiati (alcuni gratis) sui dischetti del proprio apparecchio. Tuttavia, gli abbonati devono dotarsi dell'apposita apparecchiatura e del software adatto al collegamento.

Il primo dispositivo, indispensabile quando si vuole allacciare un computer ad una banca dati attraverso le linee telefoniche, è un modem (modulatore/demodulatore). Questo apparecchio provvede a convertire, nella forma più opportuna, i segnali in arrivo e quelli in partenza. I modem sono disponibili in due versioni: ad accoppiamento diretto e ad accoppiamento acustico. La principale differenza tra i due è che il primo si collega direttamente alla linea telefonica con uno spinotto,

Collegarsi al Micronet

Per l'allacciamento al sistema Micronet occorre possedere un modem adatto ed un gruppo di programmi (su ROM o su dischetto) appositamente concepiti per il proprio apparecchio. È necessario, inoltre, abbonarsi al servizio. Nell'illustrazione è mostrato un Micro BBC collegato al Micronet. La pagina sullo schermo è quella iniziale, con il menu principale delle opzioni

mentre il secondo si collega alla cornetta del telefono ed i segnali vengono inviati sotto forma di suoni udibili.
Benché venga consigliato l'uso dei modem prodotti dalla Prism Technology per comunicare con il sistema Prestel, è sufficiente che il modem abbia una velocità di trasmissione di 1.200 baud ed una velocità di ricezione di 75 baud. I modem "acustici", in genere, possiedono il solo formato 75/1.200 baud, ma il vantaggio è che non occorre alcuna modifica all'impianto telefonico e, addirittura, si possono usare anche i telefoni pubblici (purché bastino i gettoni...).

La Prism Technology produce tre versioni di modem. I modelli 1000 e 2000, che si adattano alla maggior parte degli home computer (il software viene ceduto dal Micronet), consentono lo scambio di dati anche tra diversi computer (con velocità di trasmissione e di ricezione a 1.200 baud), oltre che con il Prestel. Nel modello 2000 la selezione di velocità avviene via software. Il terzo modello, il VTX 5000, è progettato appositamente per lo Spectrum Sinclair. Oltre ad offrire le medesime caratteristiche dei modelli 1000 e 2000, ossia tipo "full-duplex" (trasmissione e ricezione simultanea), la velocità è selezionabile tra

1. Indice generale

Questo è l'elenco generale delle varie voci disponibili. La selezione avviene mediante un codice numerico, provocando la visualizzazione della pagina richiesta

2. Posta elettronica

Uno dei servizi maggiormente consultati è il servizio di corrispondenza elettronico. Spesso, i messaggi lasciati dagli utenti sono a dir poco misteriosi

1



2



3. Giochi gratis

Alcuni giochi, come quello mostrato, si possono usare o copiare gratuitamente, per altri viene addebitato l'importo sul "conto corrente" dell'utente

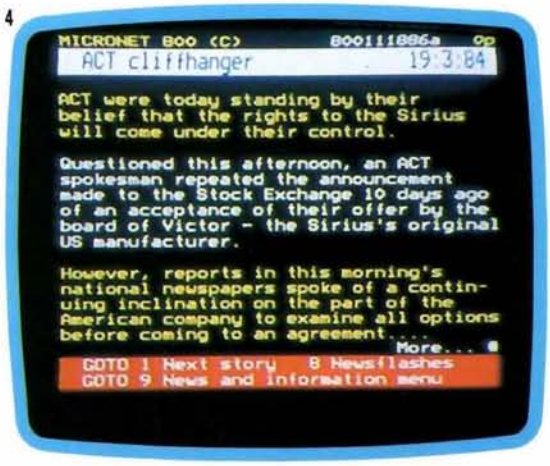
3



4. Notiziario

Le informazioni riportate nella pagina delle notizie, in genere tutte pertinenti all'informatica, vengono aggiornate quotidianamente

4



5. Carlo e Diana

Tra le tante immagini della "Galleria", una delle opzioni di Micronet, appare anche quella, stilizzata, del principe Carlo e consorte

5



6. Meteorologia

L'abbonamento a Micronet comprende anche quello all'intero sistema Prestel, che contiene informazioni di carattere molto più generale

6



LIZ HEANEY

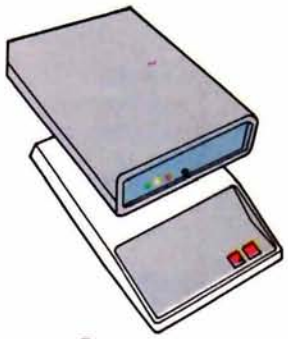
75/1.200 baud oppure 1.200/1.200 baud. In più, questo apparecchio è dotato di una ROM, che contiene tutto il software necessario.
I possessori di altre marche di computer, però, devono sobbarcarsi la spesa, oltre che per il modem, anche per l'acquisto del software. Questo è venduto su dischetto oppure, per il Micro BBC, su di un chip ROM. Il funzionamento è pressoché identico in tutti i casi, quindi l'esempio d'impiego con il Micro BBC, qui riportato, si adatta anche agli altri apparecchi.
I programmi di decodifica, scritti dalla Scicon, una delle più note software house britanniche, sono necessari per adattare i protocolli di scambio di ciascun tipo di computer a quelli adottati dal sistema Prestel. Detto per inciso, agli abbonati al Micronet è consentito il libero accesso a tutta la banca dati Prestel, che contiene enormi quantità d'informazioni di vario genere.

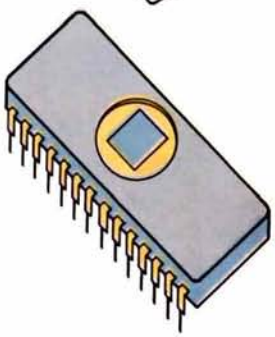
Ingresso al sistema

Tutti i sistemi del tipo "viewdata" sono pilotati da menu: all'utente viene presentata una serie di opzioni, ognuna delle quali porta ad un successivo sottomenu, fino alla "pagina" desiderata. La fase di accesso al sistema (chiamata "logging on") consiste in una serie iniziale di "quadri", destinati all'identificazione del computer chiamante, mediante due codici individuali: uno, di 10 caratteri, è "personale", l'altro è il codice di abbonamento Prestel. Quest'ultimo viene usato per addebitare (nei casi previsti) la spesa di acquisti operati mediante il servizio Prestel. Il sistema è concepito in modo da evitare che "occhi indiscreti" carpiscano tale codice: se l'utente digita una serie di asterischi, al posto del codice, la richiesta viene riproposta, ma questa volta i caratteri immessi non vengono visualizzati sullo schermo.
Ottenuto l'accesso al sistema, l'utente può spaziare per l'intero database, prelevando o depositando messaggi, trasferendo giochi o programmi applicativi sul proprio computer o dal proprio apparecchio nel sistema. Infatti, essendo il Micronet un sistema a due vie, si può acquistare software da esso, ma lo si può anche vendere.
La struttura di un database pilotato da menu deve essere, necessariamente, gerarchica ed il Micronet non fa eccezione, utilizzando un'impostazione pilotata da menu e sottomenu. Quando l'utente riesce a collegarsi con successo al Micronet mediante un modem, la prima serie di opzioni comprende una decina di titoli del tipo "Novità", "Vendite e scambi", "Posta elettronica/telex", ecc. Le possibili "uscite" da questo menu sono due: una conduce alle informazioni sull'uso del sistema (indicata con HELP), l'altra ad una delle varie opzioni di Prestel.
La principale attrattiva del Micronet consiste nella possibilità di distribuire il software; vediamo come ciò avviene. L'opzione da selezionare, in questo caso, è "Telesoftware", che provoca la visualizzazione di un elenco dei computer per i quali vi è del software disponibile, di una specie di "Hit Parade" dei programmi più richiesti, di una breve spiegazione su come acquistare e vendere software per mezzo del Micronet e di altre informazioni.
Selezionando uno dei tipi di computer, appare una nuova "pagina" di opzioni: singoli programmi di gioco, di utilità, di carattere commerciale, novità, best seller, con l'indicazione di quali, fra questi, sono gratis e quali a pagamento.
Scegliendo la voce giochi, ad esempio, viene fornito un elenco di titoli, con una succinta descrizione del funzionamento.
La selezione di un titolo, da questo elenco, porta al "Downloading menu", ossia alle opzioni disponibili per il caricamento dei programmi. Si può scegliere di copiare un programma sul proprio computer, oppure di usarlo restando collegati al Micronet (con esultanza della società telefonica...).
Una parte dei programmi, circa un centinaio, sono offerti gratuitamente, ma, essendo per la maggior parte prodotti da appassionati e dilettanti, non c'è da aspettarsi niente di particolarmente eccezionale. I prezzi del software a pagamento variano considerevolmente, essendo fissati dagli stessi autori.
Poiché la maggioranza del software disponibile è a pagamento, forse la definizione di servizio pubblico mal si adatta a ciò che risulta essere, in pratica, una vera e propria rete di vendita (assimilabile alla vendita per corrispondenza). In effetti, la Prism Technology pensa di allargare la rete di distribuzione, utilizzando il sistema americano Romox, per la produzione in serie di chip ROM preprogrammati.
Le altre funzioni del Micronet assomigliano a quelle, più generali, del Prestel: c'è un servizio di "posta elettronica", nel quale depositare o leggere messaggi, e svariate pagine di pubblicità e di notizie, specifiche ai prodotti per l'informatica.
Ciascuna "pagina" del sistema Micronet è associata ad un numero d'identificazione, cosicché, conoscendo tale numero, è possibile richiamar la pagina direttamente, anziché percorrere l'intero iter attraverso menu e sottomenu. Ciò consente un notevole risparmio di tempo (e di bollette telefoniche), oltre alla noia ed alla frustrazione che derivano dal ripercorrere più e più pagine.
Considerando la qualità e la quantità dei servizi offerti, il costo dell'abbonamento al Micronet non è certo esoso: solo un paio di sterline (circa 3.000/4.000 lire) alla settimana. Ovviamente, vanno considerate le spese d'impianto (modem e programmi) e quelle telefoniche.
Reti di distribuzione simili al Micronet saranno presto disponibili in tutti i paesi europei, ma già fin da adesso (se il costo della teleselezione internazionale non scoraggia) anche un utente italiano potrebbe allacciarsi al Micronet.

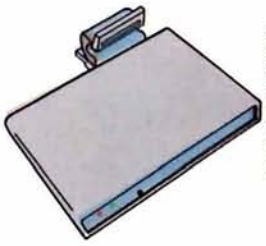


Accoppiatori acustici
È l'apparecchio più economico per il collegamento dei microcomputer tra loro o tra uno di essi ed il sistema Micronet. L'apparecchio in sé non basta: occorre il software adatto a farlo funzionare





Modem
I modem 1000 e 2000, prodotti in Gran Bretagna dalla Prism, si collegano direttamente alla linea telefonica. In questo caso, però, i costi sono maggiori



TONY DUNCAN-SMITH

Per lo Spectrum
Il VTX 5000 è stato particolarmente concepito per il computer della Sinclair, e contiene una ROM con tutti i programmi necessari al collegamento con il Micronet. L'apparecchio forma un piedistallo per lo Spectrum, collegandosi direttamente al connettore per espansioni

Unità a dischi della Dragon

Anche le prestazioni dei Dragon migliorano, grazie all'impiego di comodi disk drive



IAN McKINNELL

Linguaggio a scelta

Il primo gruppo di programmi su dischetto per i Dragon, previsto per funzionare con il sistema operativo OS9, comprende linguaggi ad alto livello, quali il PASCAL ed il C



Un software sofisticato

Grazie al sistema operativo OS9, è possibile adoperare software commerciale di buon livello. Ad esempio, è disponibile un ottimo word processor, dotato di verificatore ortografico (per adesso, buono soltanto per la lingua inglese)

I dischetti del Dragon

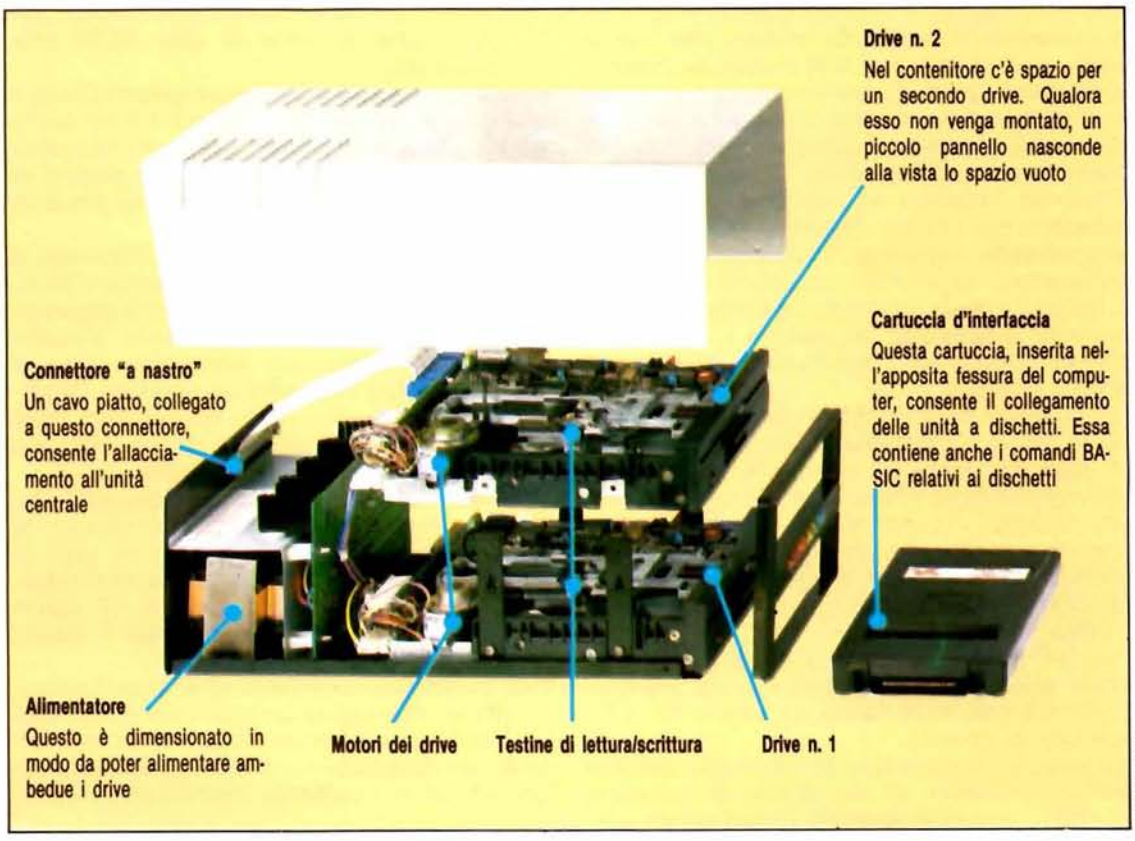
Le unità a dischetti per il Dragon possono contenere due drive. All'interno è alloggiato anche l'alimentatore. Per il collegamento al computer viene adoperato un cavo piatto, che fa capo ad un'interfaccia, inserita come una normale cartuccia nell'apposita fessura del computer. Ogni dischetto può contenere fino a 175 Kbyte. Si può usare sia il sistema operativo normale, che il molto più sofisticato OS9

Le unità della Dragon usano dischetti da 5 pollici e un quarto, a singola densità. La formattazione viene fatta suddividendo la superficie magnetica in 40 tracce, ciascuna composta da 18 settori, ognuno dei quali contiene 256 byte. Pertanto, la capacità totale di ciascun dischetto è di 40 × 18 × 256 = 184.320 byte o 180 Kbyte. Il DOS occupa parte di questa memoria per informazioni di sistema (identificazione dei file, tabelle, ecc.), lasciando circa 175 Kbyte liberi per l'utente.

La directory (l'elenco dei file conservati sul disco) si visualizza con un apposito comando. Compaiono, oltre al nome di ciascun file, il suo "tipo", la sua lunghezza ed il numero di byte ancora liberi. I file possono appartenere a quattro "tipi" diversi: programmi BASIC, dati, codici binari o file di copia (backup). Rispettivamente, per indicare il tipo di file si aggiunge, in coda al nome, una delle seguenti sigle: .BAS, .DAT, .BIN e .BAK. I manuali del Dragon non specificano il sistema adottato per la conservazione fisica delle informazioni, su dischetto, ma, non esistendo un comando di "riorganizzazione", è logico supporre che venga usata una BAM (Block Availability Map, una mappa sulla disponibilità dei blocchi).

Stranamente, sul Dragon non esiste alcun metodo per l'accesso casuale (random), un byte alla volta, ai dati memorizzati su disco, mentre questa funzione è molto utile nei programmi che leggono o scrivono brevi porzioni di informazioni (ad esempio, quelli per la gestione di database). In cambio, nel DOS esiste un metodo denominato "simulazione di accesso casuale", che pare svolgere una simile funzione, salvo richiedere qualche linea BASIC in più nei programmi. Il DOS del Dragon è un ottimo complemento del BASIC fornito.

Le unità a dischi Dragon, consideratone il rapporto prezzo/capacità, sono collocabili in una fascia media: infatti non sono troppo care, ma nemmeno molto capaci. È prevista, comunque, la realizzazione di unità a doppia densità, basate sull'impiego di 80 tracce, aumentando considerevolmente l'attuale capacità. Per il Dragon 64 è disponibile un sistema operativo opzionale, su RAM, la cui sigla è OS9. Questo occupa, è vero, ben 16 Kbyte della RAM, ma presenta ottime qualità: con esso è possibile usare linguaggi molto interessanti, quali il PASCAL, il C, il COBOL ed il BASIC strutturato, sfruttando al massimo le caratteristiche della CPU 6809.



IAN McKINNELL

I COMANDI DEL DOS

Negli esempi, viene usato un formato più o meno comune a tutti i comandi:

COMANDO"D:NOMEFILE.TIP"

dove D seleziona una delle unità (o drive) ed è compreso tra 1 e 4; NOMEFILE è lungo fino ad otto caratteri ed identifica un particolare file; .TIP è il "tipo" di file, che può essere specificato dall'operatore (se non lo è, il sistema operativo assume .BAS). Come abbreviazione del suddetto formato, negli esempi viene usata la sigla FSP.

DRIVE N

Questo comando seleziona l'unità da usare. N è compreso tra 1 e 4.

DSKINIT

Serve per formattare un dischetto nuovo:

DSKINIT D,S,T

dove D seleziona il drive, S seleziona la facciata (1 o 2) e T il numero di tracce da usare (40 oppure 80). Possedendo una sola unità basta impartire il comando DSKINIT, seguito dal tasto ENTER.

DIR D

Provoca la visualizzazione della directory del drive D, ad esempio:

```
DIR
NOMEPROG.BAS 1654
CONVERTI.BIN 1389
ANAGR.DAT 2581
NOMEPROG.BAK 1654
167322 FREE BYTES
```

COMANDI PER FILE DI PROGRAMMI

SAVE

Questo programma memorizza su disco programmi o file in formato binario. SAVE FSP serve per i programmi (l'estensione .BAS è aggiunta automaticamente). Il formato per i file binari è:

SAVE FSP,SSSS,EEEE,XXXX

dove SSSS è l'indirizzo iniziale (in decimale) del blocco di memoria; EEEE è l'indirizzo finale; XXXX è l'indirizzo di esecuzione.

LOAD

Serve per leggere programmi o file binari dal disco:

LOAD FSP

Se il file è del tipo binario, si può anche specificare, aggiungendo ,SSSS al comando, un nuovo indirizzo iniziale per il file.

RUN FSP

Carica ed esegue il programma BASIC specificato.

CHAIN FSP

Carica ed esegue il programma specificato, senza cancellare i valori delle variabili in memoria. Utile quando due programmi devono condividere dati comuni. L'aggiunta di ,SSSS ha lo stesso effetto spiegato nel caso di LOAD.

FREE D

Visualizza il numero di byte ancora disponibili sul disco specificato.

COPY

Copia un file attribuendogli un nuovo nome, ad esempio:

COPY VECCHIO TO NUOVO

Se non vengono specificati drive diversi, l'operazione avviene su un solo dischetto.

RENAME

Cambia il nome (ma non l'estensione) di un file. I nomi indicati non possono riferirsi a file su dischi differenti.

MERGE FSP

Questo comando "accoda" un programma a quello attualmente in memoria. Se vi sono accavallamenti nei numeri di linea, prevalgono quelli del programma in memoria.

KILL FSP

Cancella il file specificato dal disco.

PROTECT

Questo comando abilita una "protezione via software" contro il cancellamento accidentale del file (eccetto quando si usi DSKINIT). Il formato è:

PROTECT ON FSP

per abilitare la protezione (nella directory compare una P su sfondo inverso). PROTECT OFF FSP disabilita la protezione.

BACKUP

Serve per duplicare l'intero contenuto di un dischetto (DA) su un altro dischetto (DB):

BACKUP DA TO DB,S,T

dove S e T corrispondono a quanto detto per il comando DSKINIT. Se si possiede una sola unità, il comando BACKUP (ENTER) visualizza le istruzioni necessarie a fare la copia con un solo drive.

VERIFY

ON abilita il comando, OFF lo disabilita. Serve per controllare automaticamente l'esatta corrispondenza di un file salvato su disco con quello attualmente in memoria.

COMANDI PER FILE DI DATI

FWRITE

Il comando crea un file per dati, scrivendovi il contenuto di variabili. Per ciascuna FWRITE viene aperto un canale di comunicazione associato al nome del file, permettendo anche l'accodamento di nuovi dati. Il formato è:

FWRITE "NOMEFILE";VAR

NOMEFILE è il file da (eventualmente) creare e sul quale scrivere i dati. VAR è il nome della variabile che contiene i valori da scrivere. La virgola ed il punto e virgola sono usati per terminare le stringhe alfanumeriche nei file. Il seguente formato:

FWRITE "NOMEFILE",FROM SB, FOR TB;VAR

scrive la variabile VAR nel file NOMEFILE, iniziando dal byte SB, per una lunghezza TB. Il massimo numero di file aperti contemporaneamente è 10.

CLOSE D

"Chiude", sul drive specificato, il canale "aperto" da una FWRITE, una FREAD o una FLREAD.

CREATE "NOMEFILE",FL

Crea un file dati, chiamato NOMEFILE, di lunghezza FL.

FREAD

Il formato di questo comando somiglia a quello di FWRITE, salvo che esso serve per leggere la variabile VAR dal file. Nel secondo formato, la lettura inizia dal byte SB, dopodiché il puntatore viene incrementato a SB+TB.

FLREAD

Il formato è identico a quello di FREAD, ma in questo caso le virgole ed i punti e virgola vengono considerati come parte della variabile e non come caratteri "terminatori".

EOF

Viene usato per indicare la condizione di "fine file". Ad esempio, in:

EP=EOF("NOMEFILE")

il valore di EP è 0, finché non viene eseguita la lettura dell'ultimo dato. Allora EP viene automaticamente posto ad 1.

LOC "NOMEFILE"

Questo comando visualizza la posizione del puntatore di lettura al file specificato (che rappresenta il numero del prossimo byte da leggere).

SWRITE

Con questo comando si possono scrivere due stringhe, A$ e B$ per esempio, nel settore S, traccia T del dischetto. La massima lunghezza di ciascuna stringa è 128 byte. Il formato è:

SWRITE D,T,S,A$,B$

SREAD

Esegue la funzione inversa di SWRITE. Il formato rimane identico. I nomi delle variabili possono essere diverse da A$ e B$.

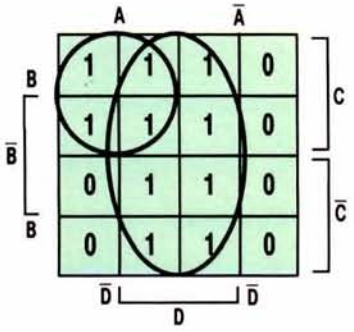
Dalla mappa al circuito

Come si semplifica il progetto dei circuiti logici, mediante l'impiego delle mappe di Karnaugh

"Quattro variabili": quando le espressioni Booleane implicano l'uso di quattro variabili, le mappe-k, ma anche le stesse espressioni, appaiono particolarmente difficili. Ma, applicando i semplici concetti, esposti in precedenza a proposito delle mappe-k, a due o tre variabili, le difficoltà si appianano.
Supponiamo, per esempio, di dover semplificare la seguente espressione:

$A\overline{B}CD + ABCD + \overline{A}BCD + A\overline{B}\overline{C}D + \overline{B}CD + A\overline{B}\overline{C}D + \overline{A}\overline{B}\overline{C}D + B\overline{C}D$

Innanzitutto, occorre una mappa-k a quattro variabili. Ma, benché vi siano solo otto elementi nell'espressione, dobbiamo riempire 10 caselle della mappa (ciascuno dei termini $\overline{B}CD$ e $B\overline{C}D$ genera due casi distinti). La mappa-k è:



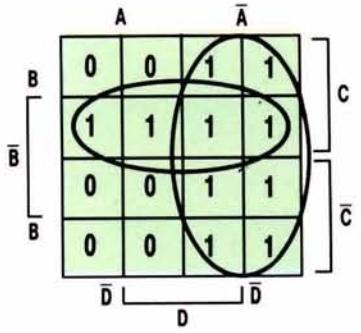
Da essa possiamo notare che il gruppo centrale degli 1 rappresenta ogni possibile caso che implichi la D, mentre il gruppo di quattro 1 in alto a sinistra è relativo a tutte le possibili combinazioni che coinvolgono A AND C. Cosicché, l'espressione si semplifica in A AND C OR D (ossia: A.C+D).
Talvolta, un'espressione va leggermente rielaborata, prima di dedurre da essa una mappa-k, come nel seguente caso:

$\overline{A+B+C} + \overline{A}.B + \overline{B+C}$

Qui dobbiamo prima applicare la legge di de Morgan (vedere a pag. 526), riformulando l'espressione in:

$\overline{A}.\overline{B}.\overline{C} + \overline{A}.B + \overline{B}.C$

dalla quale deriva la mappa-k:



Dalla mappa si può arrivare a semplificare l'espressione in: $\overline{A} + \overline{B}.C$. Ma applicando ancora la legge di de Morgan, si ottiene:

$\overline{A.(\overline{B}+C)}$

Il progetto di circuiti

Vediamo un caso pratico. Tutti conoscono la tiritera «Trenta dì conta Novembre, con April, Giugno e Settembre...». L'obiettivo è di creare un circuito con quattro ingressi (il codice dei mesi) ed un'uscita, che varrà 1 se il mese ha trenta giorni. Ogni mese è codificato in forma binaria, da Gennaio = 0001 a Dicembre = 1100.
La tavola della verità del circuito è:

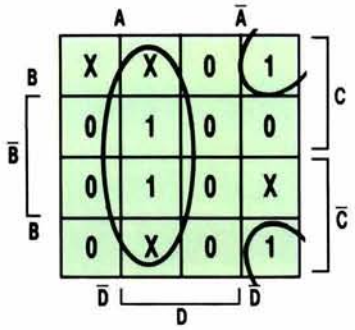
MESI	INGRESSI				USCITA
	A	B	C	D	S
	0	0	0	0	X
GENNAIO	0	0	0	1	0
FEBRAIO	0	0	1	0	0
MARZO	0	0	1	1	0
APRILE	0	1	0	0	1
MAGGIO	0	1	0	1	0
GIUGNO	0	1	1	0	1
LUGLIO	0	1	1	1	0
AGOSTO	1	0	0	0	0
SETTEMBRE	1	0	0	1	1
OTTOBRE	1	0	1	0	0
NOVEMBRE	1	0	1	1	1
DICEMBRE	1	1	0	0	0
	1	1	0	1	X
	1	1	1	0	X
	1	1	1	1	X

I segni X corrispondono ai valori non consentiti in ingresso e diamo per scontato che tali valori non verranno immessi. Quando nella

tavola risulta S=1, possiamo formulare la seguente espressione Booleana, basata sui bit in ingresso:

$S = \overline{A}.B.\overline{C}.\overline{D}+\overline{A}.B.C.\overline{D}+A.\overline{B}.\overline{C}.D+A.\overline{B}.C.D$

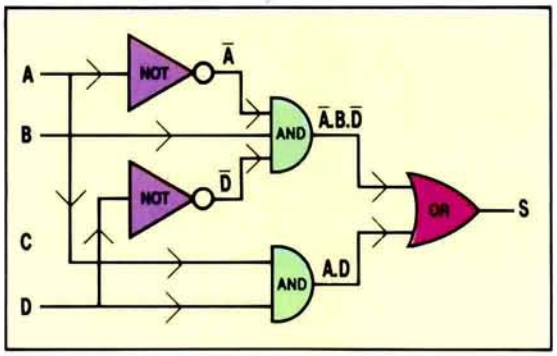
Riportando il tutto su una mappa-k, assieme alle condizioni "impossibili" (X), abbiamo:



Dalla quale possiamo derivare la forma semplificata:

$A.D+\overline{A}.B.\overline{D}$

Ecco, quindi, che possiamo costruire il nostro circuito per i mesi di trenta giorni:



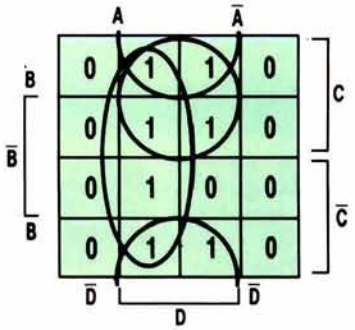
Un esempio con i numeri dispari: ci viene chiesto di realizzare un circuito nel quale, partendo da un arco di valori binari da 0000 a 1111 (da 0 a 15 in decimale), l'uscita valga 1 solo se il numero in ingresso è dispari e maggiore di 2.
Per prima cosa, si costruisce la tavola della verità per tutte le possibili condizioni:

NUMERO DECIMALE	INGRESSI A	B	C	D	USCITA S
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

Da questa, si può derivare la relativa espressione Booleana, per tutti i casi nei quali S è vero (=1):

$S = \overline{A}.\overline{B}.C.D+\overline{A}.B.\overline{C}.D+\overline{A}.B.C.D+A.\overline{B}.\overline{C}.D+$
$A.\overline{B}.C.D+A.B.\overline{C}.D+A.B.C.D$

La mappa di Karnaugh per espressione, risulta:



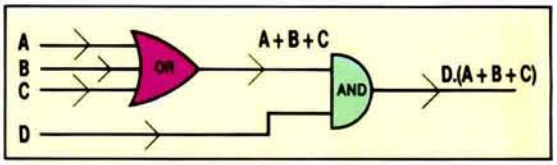
Dalla mappa, si possono isolare tre gruppi, rappresentati dall'espressione:

$S = A.D+C.D+B.D$

che può essere ulteriormente semplificata, mediante la legge distributiva, in:

$S = D.(A+B+C)$

Di conseguenza, lo schema del circuito è:



Nella prossima lezione, verranno tirate le somme di quanto esposto finora, assieme ad altri esempi di realizzazioni.

Esercizio 5

1. Semplificare la seguente espressione Booleana, usando le mappe di Karnaugh:

a) $\underline{A.B.C}+A.\underline{\overline{B}.\overline{C}}+\overline{A}.\underline{\overline{C}}+\overline{A}.B.C+A.B.\overline{C}$

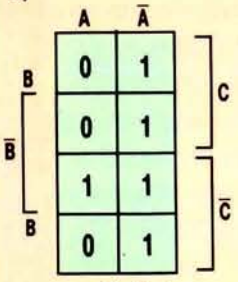
b) $\underline{B+C}+\underline{B.\overline{C}}+A.C$

c) $A.\overline{B}.D+\overline{A}.D+A.B.C.D+A.B.\overline{C}+\overline{A}.B.\overline{C}.\overline{D}$

2. Occorre progettare un circuito che accetti, in ingresso, valori binari compresi tra 0 e 7 inclusi. L'uscita deve essere 1 se il numero in ingresso è dispari oppure è un multiplo di tre (ossia: 3 o 6). Dopo aver disegnato la tavola della verità ed ottenuta la relativa espressione semplificata, si disegni lo schema del circuito necessario alla funzione richiesta.

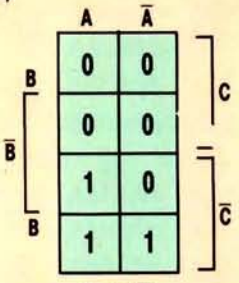
Risposte agli esercizi di pag. 573

1a)



$\overline{A}+\overline{B}.\overline{C}$

b)

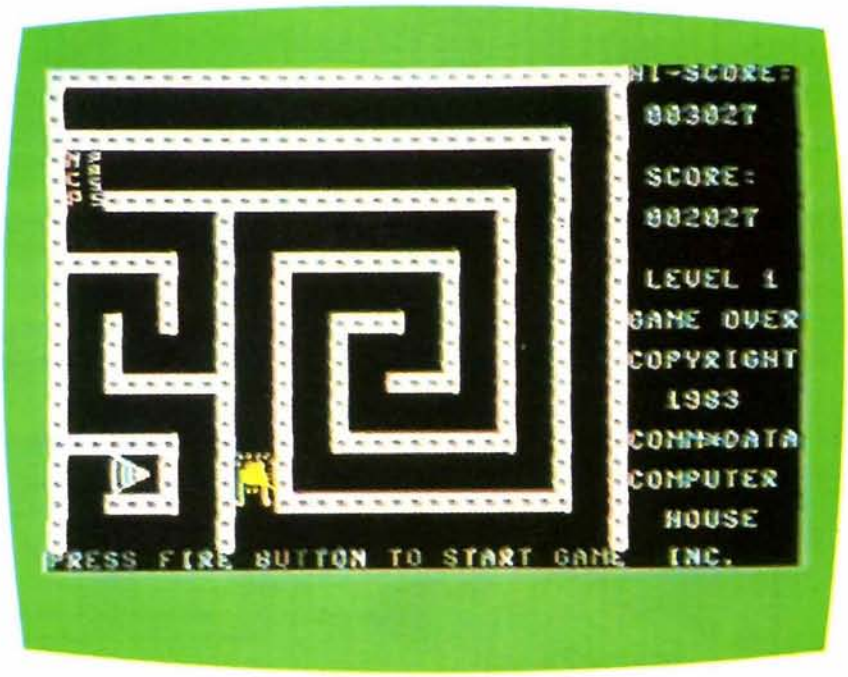


$(A+B).\overline{C}$

Software in vetrina

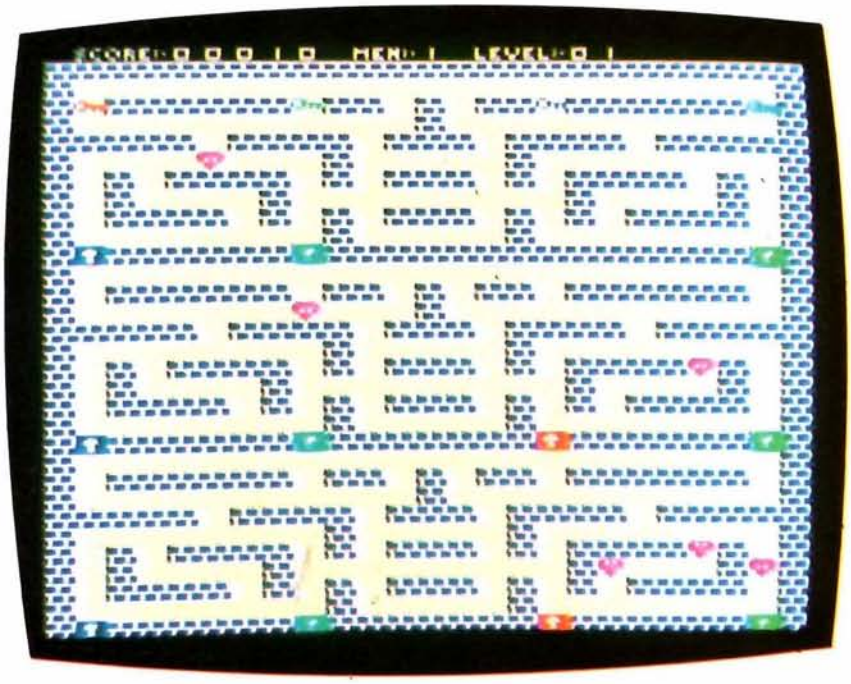
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Tra i numerosissimi programmi di gioco, quelli appartenenti al genere dei "labirinti" riscuotono un enorme successo e ne esistono molte varietà.
Tratteremo qui di due programmi di questo tipo: uno per il Commodore 64, l'altro per il Vic-20.



Escape MCP per il Commodore 64

Già il titolo, traducibile con *Fuga da MCP*, suggerisce che l'ispirazione del gioco proviene dal film della Disney *Tron*. Infatti, nel labirinto il protagonista viene rincorso da MCP, ossia Master Control Program (programma di controllo principale), come nel film. Senza dubbio la grafica del Commodore può offrire effetti migliori di quelli prodotti da questo programma. Tuttavia, il gioco è piacevole e, pur essendo abbastanza difficile sfuggire all'inseguitore, è piuttosto avvincente. Il programma è prodotto dalla Rabbit Software.



Bewitched per il Commodore Vic-20

È un gioco molto divertente, prodotto dalla Imagine. Il successo del gioco consiste nel saper uscire da un labirinto, nel quale esistono quattro porte (ciascuna di un colore diverso dalle altre). Per aprire le porte, occorre impossessarsi delle chiavi, il cui colore corrisponde a quello delle varie porte, situate nella parte alta dello schermo. Per rendere più difficile, e quindi più avvincente, il gioco, alcune porte possono "bloccarsi", pur avendo la chiave giusta. Per riuscire nell'intento, occorre una discreta pratica, garantendo ore ed ore di divertimento.

L'AIM 65 della Rockwell

Un computer particolarmente orientato alla sperimentazione ed alla didattica

Nella versione più semplice, l'AIM 65 consiste in una scheda di circuiti, senza alcun contenitore. Tuttavia, dal punto di vista operativo, l'apparecchio è completo e possiede molte attrattive, tra le quali una piccola stampante incorporata, fatto insolito negli apparecchi cosiddetti "di sviluppo", (ossia orientati alla didattica). In genere, gli apparecchi di sviluppo non sono dotati di una tastiera di dimensioni regolari, né di un comodo display a LED, che consente di visualizzare ben 20 caratteri alla volta.
Benché non essenziali al funzionamento, queste tre caratteristiche differenziano l'AIM dagli altri computer del genere, facilitandone l'uso. Per quanto concerne i canali di ingresso/uscita e la capacità di memoria, invece, l'AIM rientra abbastanza nello standard, avendo 4 Kbyte di RAM e 12 Kbyte di ROM. Paragonato ai computer che appartengono alla fascia degli home e dei personal, nei quali una RAM di 16 Kbyte è il minimo, la capacità dell'AIM sembra quasi ridicola, ma occorre ricordarsi che si tratta di un apparecchio per la didattica, nel quale, anche con così poca memoria, si ottengono risultati pregevoli. Comunque, esiste la possibilità, mediante schede aggiuntive, di ampliare la RAM fino a 32 Kbyte.
Poiché nell'AIM è adoperato il ben noto microprocessore 6502, i confini di ciascuna zona di memoria sono fissi: i primi 512 byte della RAM sono riservati al sistema (dalla locazione $0000 alla $00FF ci sono gli "pseudo registri" del 6502, mentre dalla locazione $0100 alla $01FF c'è lo "stack" del sistema). Poiché non occorre che i decodificatori degli indirizzi includano la parte alta dell'indirizzo (la "pagina"), che sarebbe, comunque, sempre uguale a 00000000, le istruzioni di conseguenza sono più sommarie. Questa è una delle caratteristiche distintive del chip 6502, che lo hanno reso così famoso tra i fabbricanti di computer.
La quantità di memoria ROM dell'AIM è superiore a quella normalmente trovata in altri apparecchi didattici e sono disponibili 5 zoccoli per aggiungere altri chip di ROM, a seconda delle esigenze. La ROM è collocata nella parte "alta" della mappa di memoria,



CHRIS STEVENS

Un sistema completo con l'AIM
Considerato che l'uso principale di questo computer è nei laboratori didattici (o comunque nel settore sperimentale), l'aspetto esteriore di un sistema completo non è particolarmente attraente. Esiste, in commercio, un involucro adatto a contenere la scheda principale e la tastiera, ma molti utenti preferiscono prima investire in qualche espansione di memoria. La scatola azzurra è l'alimentatore del sistema, mentre quella più a sinistra è il "box" delle espansioni, collegabile all'unità centrale mediante un cavo piatto



Scheda d'espansione per la memoria
Questo contenitore per schede, o "box", permette l'aggiunta di un massimo di otto schede di estensione (ma ne esistono versioni adatte a contenere fino a 16 schede). Le dimensioni delle schede sono standard ed il connettore è del tipo "Euro", dotato di 96 piedini. La memoria può essere ampliata ad un massimo di 128 Kbyte tra RAM e ROM. Una delle schede aggiuntive ha una porta standard IEEE in più rispetto al sistema (per il collegamento con floppy disk, stampanti o altre periferiche)

così come imposto dall'uso del chip 6502. È disponibile una considerevole quantità di linguaggi e di programmi "di utilità", ma tutti dipendono dal programma "monitor", che risiede nella ROM, occupando circa 4 Kbyte (dalla locazione $E000 alla $EFFF). Questo programma contiene un "editor" (per la scrittura e la memorizzazione di linee di testo), un "single-stepper", (ossia un programma che consente l'esecuzione "passo per passo" dei programmi, permettendo di esaminare e modificare il contenuto della memoria durante l'esecuzione di un programma), oltre alle normali funzioni di esame e modifica dei registri e delle locazioni di memoria.
Tra i chip di "firmware" (così si chiama il software su ROM) disponibili, ve n'è una che contiene un discreto BASIC, ma l'interesse dei possessori dell'AIM, maggiormente orientati alla sperimentazione, va in genere ai programmi di controllo per processi industriali, e tra i linguaggi adatti allo sviluppo di tali programmi si può usare, sull'AIM, il PL65 (una via di mezzo tra l'ALGOL ed il PL/I).
Un altro linguaggio è l'INSTANT PASCAL, che, a differenza di altre versioni di PASCAL, consiste in un interprete e non in un compilatore. Il PASCAL, comunque, lo si può usare soltanto aggiungendo opportune espansioni di memoria, poiché l'occupazione di RAM è notevole. Infine, un altro linguaggio disponibile, degno di nota, è il FORTH, per il quale non necessita alcuna espansione di memoria.
Considerando che gli apparecchi di sviluppo vengono principalmente usati in applicazioni di controllo su macchinari, l'AIM è dotato di un'ampia gamma di canali di ingresso/uscita, anche nella versione base.
Oltre al connettore per le espansioni, sul quale si trovano tutti i segnali di sistema (compresi i bus dei dati e degli indirizzi, i segnali di clock e le linee d'alimentazione), l'AIM 65 è dotato di due chip 6522 (o VIA, da Versatile Interface Adaptor, ossia adattatori versatili d'interfaccia), uno dei quali impiegato per il controllo delle interfacce per la stampante e per il registratore. L'altro chip, utilizzabile dall'utente in base alle proprie necessità, è collegato ad un apposito connettore per applicazioni ausiliarie, rappresentando un doppio canale di ingresso/uscita bidirezionale. Per il controllo e la gestione della tastiera viene usato uno speciale chip, il 6532, contraddistinto dalla sigla RIOT (RAM I/O Timer, ossia temporizzatore d'ingresso/uscita per RAM).
Poiché l'uso dell'AIM è rivolto alla sperimentazione, è molto probabile che non venga usato con linguaggi ad alto livello, bensì con moduli ROM in codice macchina, preparati



La tastiera
La tastiera, contrariamente a quanto avviene in altri apparecchi "di sviluppo", è di formato regolare e facilita l'immissione di istruzioni tanto in BASIC, che in codice macchina. Il tasto PRINT, visibile in alto a destra, viene usato assieme al tasto CTRL per attivare o disattivare la stampantina incorporata nell'AIM 65. Se questa è attiva, tutto ciò che compare sul display viene riprodotto sulla stampante

CHRIS STEVENS

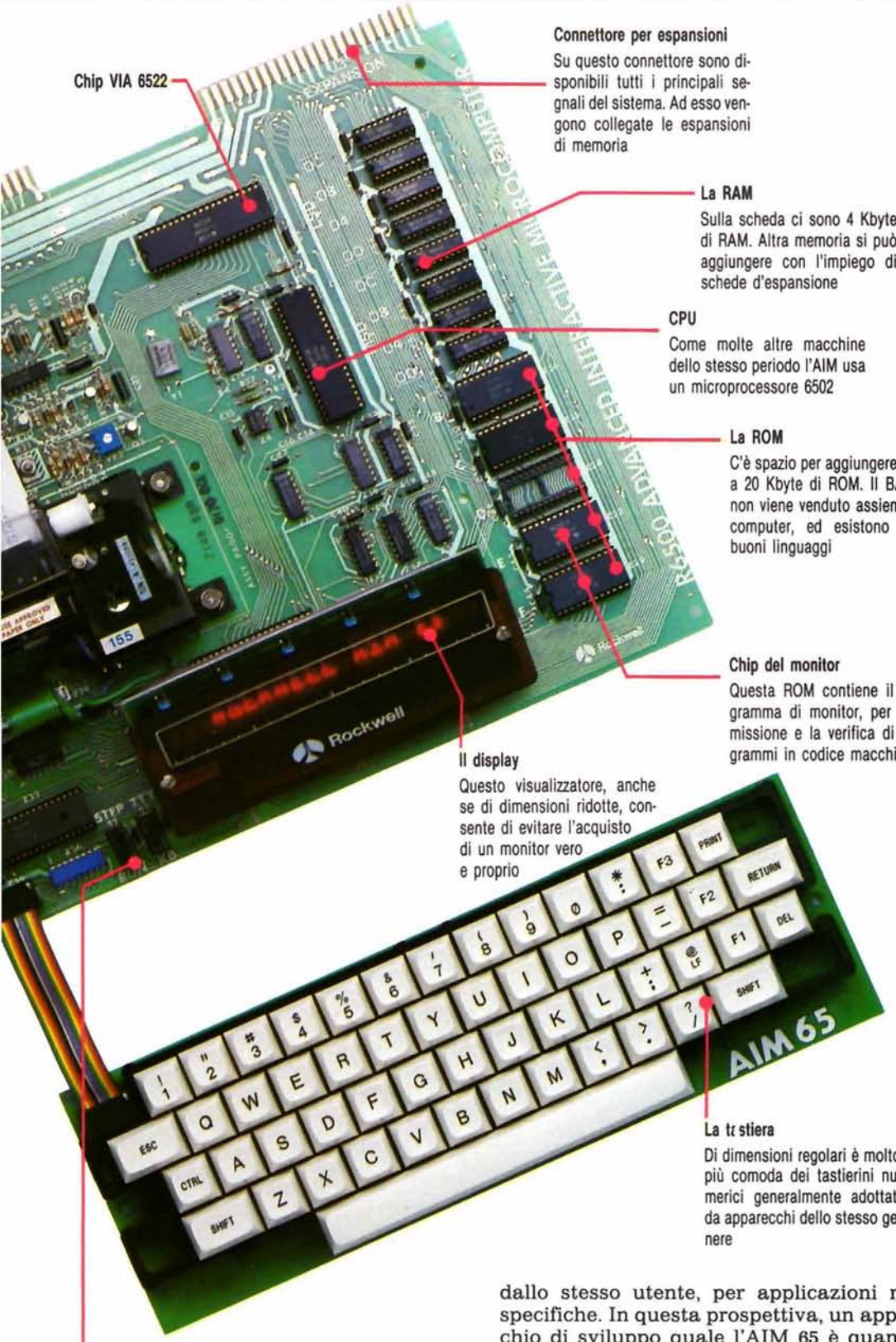
Connettore ausiliario
Mediante questo connettore si accede alle due porte ausiliarie dell'AIM

Connettore per l'alimentazione
Per l'apparecchio occorre una tensione di +5 volt, mentre la stampante funziona a 12 volt

Stampante
La stampa avviene su 20 colonne e la carta impiegata è del tipo per calcolatrice

Tasto di reset
Per riavviare il computer

Decodificatore della tastiera
Viene usato uno speciale chip, il 6532



dallo stesso utente, per applicazioni molto specifiche. In questa prospettiva, un apparecchio di sviluppo quale l'AIM 65 è quanto di meglio si possa chiedere, ad un prezzo molto ragionevole.

In definitiva, l'AIM 65 è un computer di facile e flessibile impiego, di costruzione solida, ben concepita e certamente molto adatta nel campo della didattica sul controllo di processi automatizzati. Non è particolarmente adatto, invece, per chi intendesse adoperarlo per le applicazioni più tipicamente appartenenti al campo degli home computer.

AIM 65

DIMENSIONI

mm. 292 x 267 x 60

CPU

6502 della Intel

CLOCK

—

MEMORIA

4 Kbyte di ROM (estendibile fino a 20 Kbyte) e 4 Kbyte di RAM (estendibile a 64 Kbyte)

VIDEO

Per un monitor occorre una scheda aggiuntiva. Il sistema è dotato, comunque, di un display a LED per 20 caratteri

INTERFACCE

Due porte bidirezionali per periferiche, con due segnali di controllo ciascuna, più il bus di sistema

LINGUAGGI DISPONIBILI

Un MINI ASSEMBLER ed un EDITOR sono incorporati, ma sono disponibili il FORTH, l'INSTANT PASCAL, il BASIC ed il PL/65

TASTIERA

Di dimensioni e formato tradizionali, con 53 tasti

DOCUMENTAZIONE

I manuali d'installazione e quelli d'impiego sono di una chiarezza unica, per chi conosce l'inglese

PRO

L'AIM 65 è un apparecchio molto versatile, particolarmente adatto alla sperimentazione ed al controllo di apparecchiature elettromeccaniche. La documentazione è ottima

CONTRO

Considerata la fascia d'uso, non vi sono particolari degni di nota. Non è disponibile una gran quantità di software commerciale ad alto livello, ma, del resto, non è questa la primaria funzione dell'apparecchio

L'ufficio computerizzato

L'enorme diffusione di computer sempre più economici consente di avvalersi di nuovi sistemi contabili

Sotto la voce "software commerciale" si può raccogliere un vastissimo numero di applicazioni. Oltre ad un'interminabile serie di variazioni, i programmi di questo genere differiscono anche nel livello delle prestazioni. La prima grossa distinzione, in base al livello, può essere tra il software fornito su disco e quello, invece, venduto su cassetta, adatto alla fascia dei computer di più piccole dimensioni. Poiché le unità a disco sono più veloci ed efficienti, i migliori programmi ne prevedono l'impiego, anche se, talvolta, il prezzo dei drive incide sensibilmente sull'acquisto delle apparecchiature.

La seconda grossa distinzione è tra i programmi adatti alla multiutenza (ossia in sistemi dotati di più terminali) e quelli, invece, eseguibili da un solo apparecchio per volta. Iniziamo il nostro esame dal software più semplice, per trattare in seguito le caratteristiche di sistemi più complessi.

Ad usare il software commerciale, sia esso per home computer o per sistemi a multiutenza, sono soprattutto utenti digiuni di programmazione. Pertanto, i programmi sono generalmente concepiti secondo un criterio "interattivo", ossia da un sistema nel quale il funzionamento è facilitato dal frequente impiego di menu e di messaggi esplicativi.

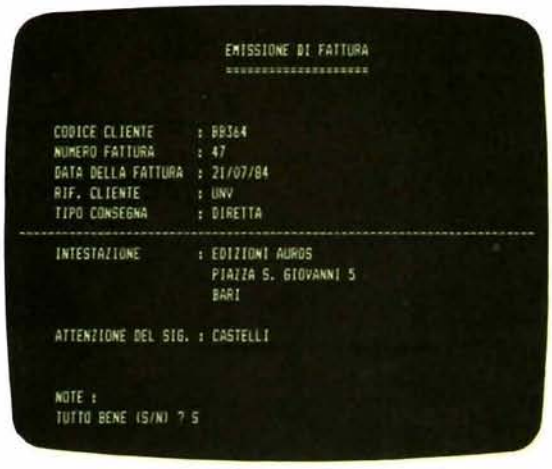
La selezione di una voce da un menu conduce ad un successivo sottomenu e così via finché non si giunge alla funzione richiesta. A questo punto, sullo schermo compaiono dei messaggi che sollecitano l'immissione, in forma organizzata, di particolari dati ed informazioni. Ad esempio, può trattarsi dell'immissione di una "prima nota" contabile. Il concetto noto sotto il nome di "user-friendly" (traducibile in italiano soltanto in modo approssimativo con "uso facilitato"), è il segreto che ha portato al successo i migliori programmi per applicazioni commerciali.

Nei programmi contabili, la materia può essere grosso modo suddivisa in tre settori: la gestione degli acquisti, quella delle vendite e quella della contabilità generale, che riassume l'andamento di un'attività commerciale.

In tutti i programmi per computer, siano essi per applicazioni commerciali o no, vengono impiegati degli archivi (file). Ogni file è composto da più record ed ogni record da più campi. La distinzione tra questi elementi, nei programmi di contabilità, è facilmente individuabile: una gestione delle vendite, ad esempio, adopera un file che contiene un record per ciascun cliente. Le varie informazioni relative ad ogni cliente (ragione sociale, indirizzo, codice fiscale, ecc.) costituiscono i campi del record.

Ovviamente, un programma di contabilità deve svolgere operazioni di calcolo, quindi deve contenere dei sottoprogrammi per l'immissione di dati numerici, ed altri che elaborano in modo opportuno tutte le quantità numeriche.

Nei sistemi basati sull'impiego di unità a





Moduli separati

Spesso, i programmi di contabilità per apparecchi dotati di registratore a cassette costituiscono dei moduli indipendenti l'uno dall'altro: il programma di fatturazione consente l'emissione di fatture, ma l'aggiornamento del magazzino non avviene automaticamente ed occorre usare un secondo programma. I programmi "integrati" (normalmente disponibili solo nelle versioni per apparecchi a dischi), prevedono l'aggancio automatico tra contabilità, fatturazione e magazzino

Lo stretto necessario

I "package" di contabilità su nastri a cassetta devono necessariamente funzionare all'insegna della stringatezza, per essere di qualche utilità nonostante le limitate capacità di memoria e l'impossibilità di adoperare archivi ad accesso casuale. Spesso, quindi, detti programmi servono più come traccia indicativa sull'andamento generale di un'azienda, che come efficaci programmi ai quali affidarsi in modo completo

disco, i programmi possono prevedere registrazioni corredate da una considerevole quantità di dettagli, riferiti tanto ai clienti che ai fornitori. Lo scopo da raggiungere è comunque quello di registrare in forma completa e permanente, quanto più possibile su di una transazione operata dalla ditta. Tuttavia, esistono dei limiti alla quantità di dettagli memorizzabili, anche sui sistemi più costosi e dotati di memorie di massa capienti. Una delle prime domande, che un potenziale acquirente di un programma dovrebbe porsi, è se il software è capace di gestire il volume di affari prodotto dalla propria attività.

Non è da trascurare il fatto che più dati vengono memorizzati, tanto maggiore è il tempo necessario per il riordino di essi. Ecco perché alcuni si orientano verso programmi di contabilità impostati sul metodo delle "partite aperte", nei quali vengono conservati i dati relativi soltanto alle transazioni non ancora concluse (ossia con debiti o crediti sospesi).

Un altro criterio, adottato per economizzare sulla memoria, consiste nel conservare soltanto i "dare/avere" progressivi, dei vari conti e sottoconti. Il sistema diviene meno elastico, a causa della minore quantità di informazioni disponibili, ma più facile da usare. Occorre dunque accettare un compromesso tra efficienza (misurata come quantità di dettagli memorizzati) e la capacità dell'apparecchio che si possiede.

Si può facilmente prevedere fin da ora, che il software basato sull'impiego di unità a cassette non offre mai le prestazioni ottenibili, invece, grazie all'uso dei dischetti: emissione automatica delle fatture, degli estratti conto e di tabulati statistici, ecc.

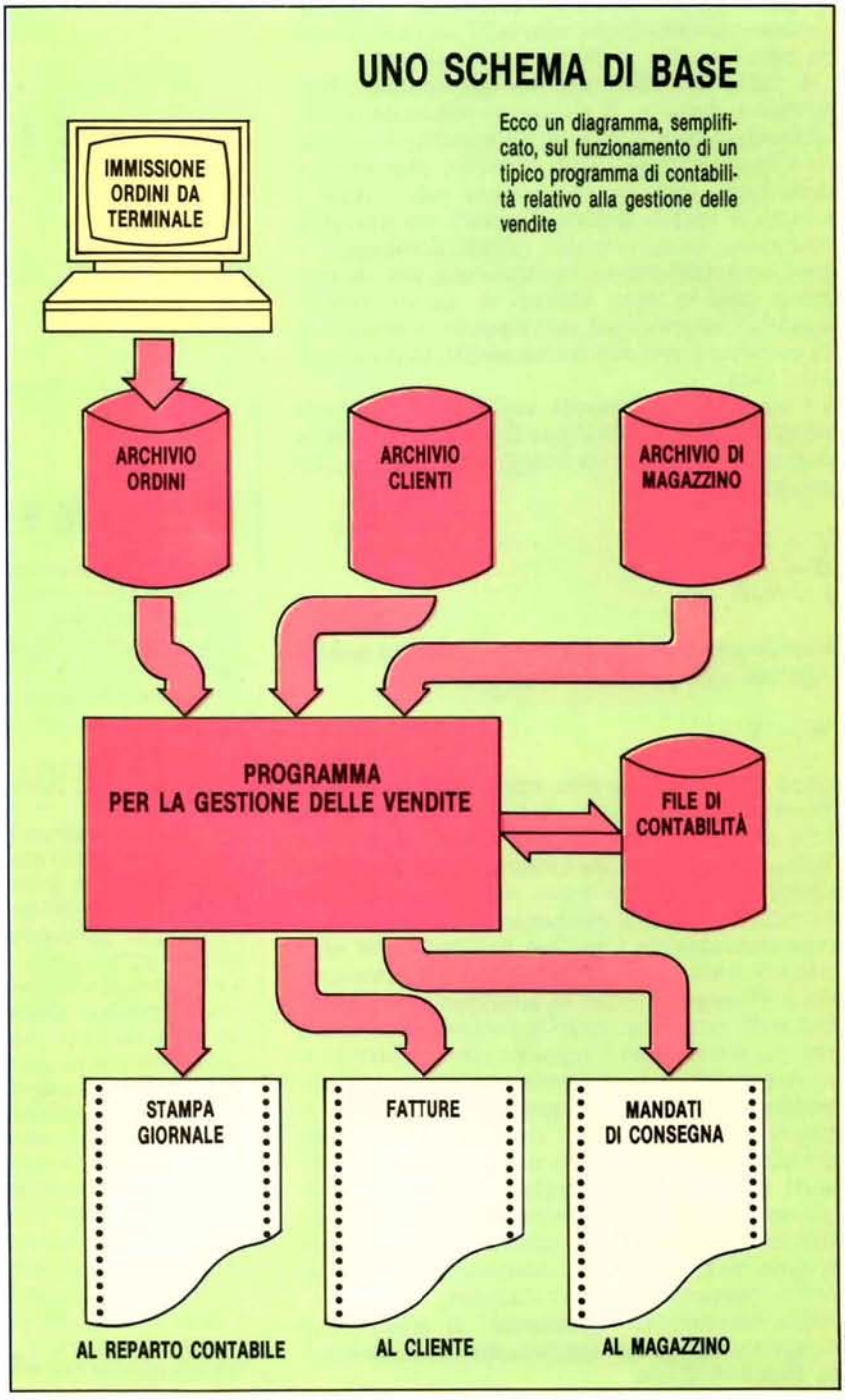
Per comprendere i meccanismi dei programmi di contabilità, senza essere esperti né di ragioneria né di programmazione, conviene sempre riferirsi ad un concetto di "flusso" del denaro. Semplificando molto il problema, le informazioni di vitale importanza sono due: sapere quanto, in termini economici, produce un'attività e quali sono i costi generati per il funzionamento di tale attività. Uno dei più semplici sistemi manuali per ottenere questi dati è un normale "giornalmastro" ossia un registro dotato di un numero di colonne adeguato ad identificare le differenti voci di "uscita" e di "entrata". Dovendo annotare, per legge, anche gli importi I.V.A. relativi alle transazioni, nelle contabilità semplificate (manuali), si usano direttamente i registri I.V.A. degli acquisti e delle vendite.

Tali registri devono, almeno in teoria, essere aggiornati quotidianamente.

Trasponendo tali procedure in programmi eseguibili al computer, è logico supporre che nei sistemi nei quali è adoperato un registratore a cassette, il formato e la quantità di dati, oggetto delle registrazioni, deve essere necessariamente ridotto al minimo essenziale. Non è detto, però, che simili contabilità "ridotte", non si prestino egregiamente per attività di piccole dimensioni, nelle quali, più che una gran ricchezza di dettagli, tabulati, opzioni ecc. è più necessario tenere ordine nelle registrazioni, tenendo al minimo le possibilità di commettere errori.

Generalmente, infatti, nei programmi di contabilità di questo tipo il "Piano dei Conti" è di ampiezza piuttosto limitata, come il numero di registrazioni di "prima nota" registrabili. Ciò spesso impone una gestione a periodi mensili, al termine dei quali vengono, più o meno automaticamente, riportati i saldi al periodo successivo. Inoltre, le stampe disponibili sono limitate, magari, alla lista delle registrazioni di prima nota ed ai saldi dei vari conti.

Nelle prossime parti di questa sezione, verranno esaminati più a fondo le caratteristiche dei vari sistemi contabili computerizzati.



Le leggi elettriche

Una delle leggi fondamentali è quella di Ohm che lega voltaggio, corrente e resistenza

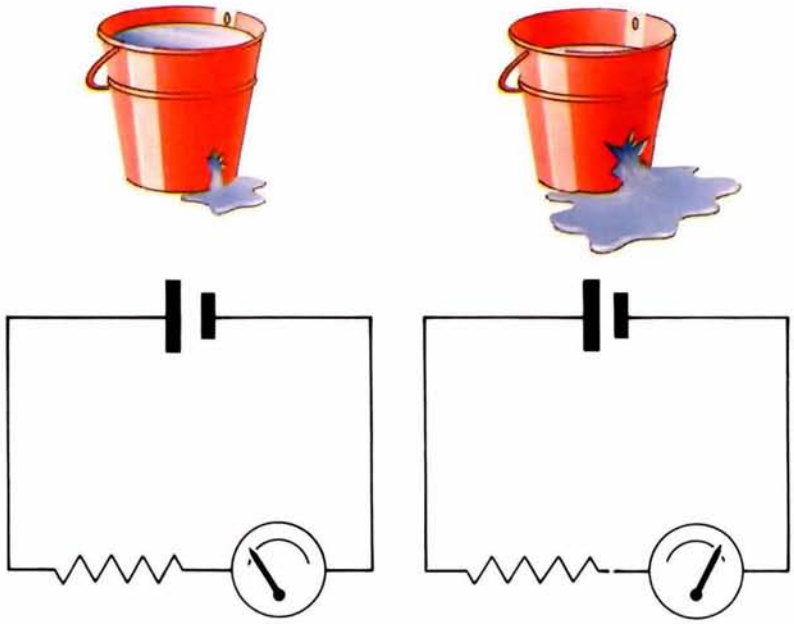
Nella lezione precedente, abbiamo esplorato alcuni fenomeni che servono al funzionamento delle apparecchiature elettriche. Vediamo adesso una relazione che lega alcune di esse in una formula di facile applicazione.
Un fisico del 19mo secolo, Georg Ohm, fu il primo a dedurre il rapporto esistente tra la differenza di potenziale, l'intensità di corrente e la resistenza di un circuito, riuscendo a formulare una legge in base alla quale è possibile legare assieme queste tre quantità elettriche. In un circuito, infatti, il voltaggio è pari alla resistenza moltiplicata per la corrente che in esso scorre. A questo punto, semplici espressioni aritmetiche consentono di calcolare uno dei tre elementi, in base agli altri due.
Le equazioni risultanti, nelle quali vengono adoperate le lettere V per il voltaggio, I per la corrente ed R per la resistenza, sono molto semplici:

$V = R \times I$
$R = V/I$
$I = V/R$

Una conseguenza della legge di Ohm è quella relativa alla potenza di un circuito:

$W = V \times I$

dove W si riferisce alla potenza, misurata in Watt, dissipata da un circuito, sottoposto ad una differenza di potenziale V (misurata in Volt) ed una corrente I (misurata in Ampère). Questa formula è utile anche nell'ambito domestico, perché consente di dimensionare opportunamente i fusibili dell'impianto elettrico. Supponiamo di avere uno scaldabagno da 3 kilowatt. Poiché la tensione di rete è di 220 volt, possiamo calcolare che la corrente che scorre in esso è leggermente superiore a 13 Ampère (per l'esattezza 13,6363...). Aggiungendo alla nostra supposizione che non vi siano altri apparecchi collegati allo stesso circuito dell'impianto (tra l'altro con 3 Kilowatt la linea è già sufficientemente "sotto sforzo"), possiamo calcolare il valore del fusibile da applicare. Un fusibile da 13 Ampère è troppo basso e salterà immediatamente: occorre usarne uno da 15 Ampère.
Nelle successive "esperienze" di laboratorio vengono viste altre applicazioni pratiche della legge di Ohm.

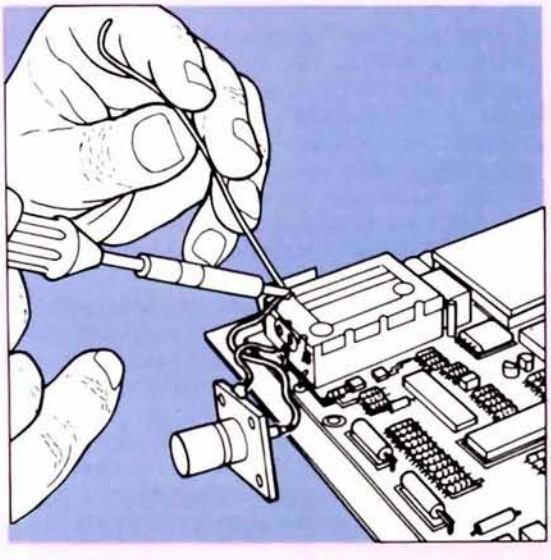


CORRENTE E RESISTENZA

Ogni componente di un circuito elettrico offre resistenza al passaggio della corrente, così come la maggiore o minore ampiezza di una tubazione consente un flusso altrettanto maggiore o minore d'acqua. Analogamente, in un circuito elettrico, maggiore è la resistenza, minore è il flusso di corrente, a parità di voltaggio. La resistenza elettrica di un materiale dipende dalla sua composizione fisica: nelle connessioni tra i componenti vengono spesso usati fili di rame, perché la resistenza elettrica di questo materiale è pressoché trascurabile, consentendo una bassa dispersione d'energia

UNA MODIFICA ALLO SPECTRUM

La piccola modifica, che ora si suggerisce, consente di ottenere un segnale video composito dallo Spectrum della Sinclair e di poter, quindi, usare un monitor ad alta risoluzione. Si apre il contenitore del computer, e si individua il modulatore RF (un voluminoso componente protetto da una custodia metallica). Il modulatore ha due collegamenti: si trovi quello più vicino alla parte posteriore dell'apparecchio e si saldi il capo di un filo elettrico ad esso, saldando l'altro capo alla "massa" di una presa per monitor. Si saldi poi un altro cavetto, collegandone un capo all'involucro metallico del modulatore e l'altro al perno centrale della presa per il monitor



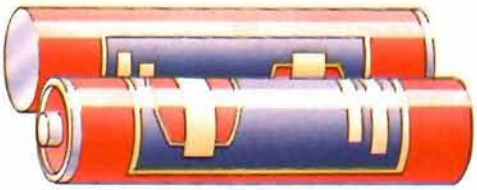
I COMPONENTI DI UN CIRCUITO

Lo schema di un circuito elettronico può sembrare incomprensibile a prima vista, ma si tratta soltanto di associare a ciascun simbolo il componente corrispondente. A scopo dimostrativo, riportiamo il caso di un circuito per interfono, descrivendo ciascun componente ed associandolo al relativo simbolo dello schema elettrico. Accanto ai simboli, nello schema, sono spesso riportate delle sigle (ad esempio: R1, R2, C1, ecc.), che servono ad identificare i valori di ciascun componente. Le linee che collegano un componente all'altro rappresentano semplicemente i fili elettrici (o le tracce di un circuito stampato)

1. Interruttori

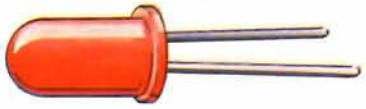
Ve ne sono di diversi tipi, ma i principali sono due: quelli "a scatto" e quelli "a pulsante". Nei primi il contatto viene aperto (o chiuso) e così rimane fino a nuova operazione. Nei secondi, il contatto viene aperto (o chiuso) soltanto finché si preme il pulsante





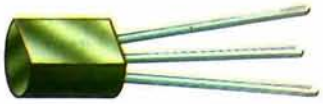
2. Diodi LED (fotoemittenti)

I diodi sono il tipo più semplice di semiconduttori e consentono il fluire della corrente soltanto in una direzione. Alcuni semiconduttori, incapsulati in materiale plastico trasparente, emettono una certa quantità di luce al passaggio della corrente e sono molto utili come "spie"



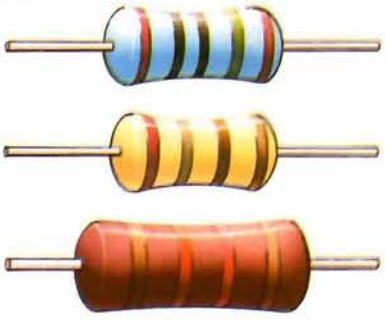
3. I transistor

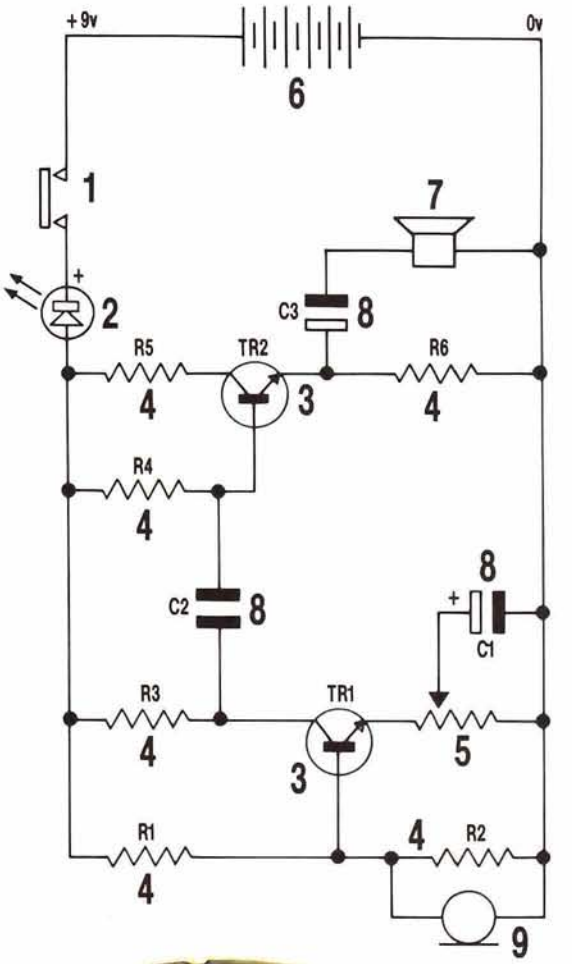
A seconda di come vengono utilizzati nel circuito, i transistor possono agire da interruttori o da amplificatori di segnale. La loro invenzione, nel 1947, ha rivoluzionato l'industria elettronica

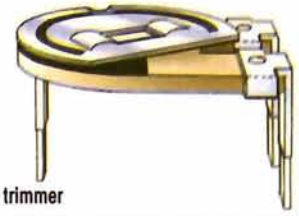


4. Resistenze

Introducendo dei componenti che offrono resistenza al passaggio della corrente, è possibile controllare il flusso di essa nelle varie parti di un circuito. Le resistenze vengono vendute in valori standard, che vanno da pochi ohm alle decine di Megaohm. Il valore è indicato mediante un codice a colori







5. I trimmer

Non tutte le resistenze hanno un valore fisso: spesso è utile poterlo variare per controllare, ad esempio, il volume di un apparecchio. In genere si tratta di una striscia semicircolare di materiale resistivo (spesso è usato il carbone), sulla quale scivola una "slitta" metallica. La resistenza totale varia a seconda della posizione della "slitta"

6. Pile a secco

Circuiti di modeste dimensioni, che presentano un limitato consumo di energia, possono essere alimentati da comuni pile a secco

7. Altoparlanti

Gli altoparlanti sono parenti molto prossimi dei microfoni: lavorano grazie agli stessi principi, ma per scopi opposti. In un altoparlante, un sottile diaframma vibra in base alla corrente applicata ad un elettro magnete, producendo vibrazioni dell'aria circostante, ossia suoni



8. Condensatori

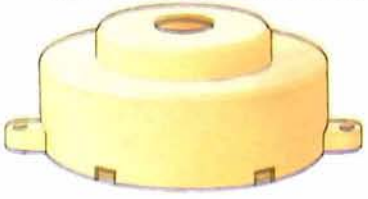
I condensatori (o capacità) sono dei componenti in grado di conservare una carica elettrica. Un condensatore si "carica" se ad esso viene applicata una differenza di potenziale (voltaggio) e rimane in tale condizione (anche se viene tolta l'alimentazione), finché i suoi due terminali non vengono a contatto o il circuito assorbe questa corrente accumulata





9. Microfoni

I microfoni funzionano in modo opposto agli altoparlanti: le onde sonore fanno vibrare una membrana che, collegata ad un elettromagnete, converte il suono in segnali elettrici, utilizzabili da un circuito



KEVIN JONES

Byte per byte

I primi passi con il linguaggio Assembly 6502 e Z80

Nelle lezioni precedenti, abbiamo visto come le linee di programma BASIC vengano memorizzate con una combinazione di caratteri ASCII e di "token".
Ci siamo accorti che, in fondo, il BASIC è un linguaggio ad alto livello per modo di dire: consiste, essenzialmente, in una sequenza di istruzioni e ciascuna istruzione consiste in una parola di comando (immediatamente sostituita con un token, ossia da qualcosa di appena superiore ad un op-code del codice macchina), seguita da una sequenza di dati relativi alla particolare istruzione.
Il fatto che le parole impiegate, ed i relativi dati, siano vicini ad un linguaggio naturale e che le istruzioni siano visibilmente separate da numeri di linea o da virgole, fa sembrare un programma BASIC più di alto livello a noi umani, che non all'interprete BASIC. Ne consegue che il codice macchina, se presentato in una forma adatta, può essere facilmente comprensibile.
L'operazione "cosmetica" viene effettuata dal programma "Assembler", nel quale vengono usati i codici mnemonici del linguaggio ASSEMBLY (quali, ad esempio: LDA o ADC) al posto di codici operativi, numerici, corrispondenti e, invece di indirizzi e valori esadecimali, vengono impiegate delle "label" (ossia "etichette") del tipo DEPOS1, FLAG2.
Siccome il microprocessore non comprende il linguaggio ASSEMBLY, occorre un programma che converta il tutto in codici numerici (l'Assembler), salvo il programmatore non desideri, non senza una certa dose di masochismo, operare la conversione manualmente.
La funzione di un Assembler è quella di trasporre, ad una ad una, tutte le istruzioni ed i dati in altrettanti codici operativi e valori numerici, a differenza di un interprete BASIC, nel quale ad ogni comando corrisponde tutta una serie di codici. Il linguaggio ASSEMBLY è molto più comprensibile del codice macchina, non essendo altro che una diversa forma di esso, più comprensibile al programmatore.
In generale, quindi, un programma viene scritto in ASSEMBLY e quindi "assemblato", ma il codice risultante dall'operazione, salvo rari casi, interessa soltanto in quanto direttamente eseguibile dal computer e non per altri motivi.
Come detto più volte, il microprocessore, in sostanza, non fa che manipolare il contenuto della memoria.
Queste operazioni avvengono nella RAM, ma in modo particolare in una speciale zona di memoria, generalmente interna allo stesso microprocessore, che viene, per comodità, suddivisa in "registri". Questi sono delle locazioni "privilegiate", in quanto su di esse il computer può operare tutta una serie di operazioni.

Il registro accumulatore

Il più importante registro di un microprocessore è l'"accumulatore". Essendo direttamente collegato all'Unità Logica ed Aritmetica viene adoperato con maggior frequenza degli altri registri.
Per manipolare un dato occorre prima depositarlo nell'accumulatore (operazione che si chiama "caricamento" dell'accumulatore). In linguaggio ASSEMBLY per il 6502, ciò si ottiene con l'operazione LDA, mentre, per lo Z80, con l'operazione LD A.
Estrarre un dato dall'accumulatore è altrettanto importante: nel 6502 si usa l'operazione STA. Nello Z80 questa operazione viene considerata alla stessa stregua della precedente, ossia un trasferimento di dati, cosicché l'operazione è ancora indicata con LD A, ma con un diverso formato, come vedremo in seguito.
Supponiamo, dunque, di voler scrivere un programma in ASSEMBLY per copiare il contenuto di un byte in quello immediatamente successivo.
Ecco ciò che scriviamo:

6502	Z80
LDA $09FF	LD A,($09FF)
STA $0A00	LD ($0A00),A

In ambedue i casi, il contenuto di $09FF, qualunque esso sia, viene copiato in $0A00. La locazione $09FF può contenere qualsiasi valore compreso tra $00 ed $FF: esso viene "caricato" nell'accumulatore e quindi copiato nella locazione $0A00. Nell'ASSEMBLY 6502 non è chiaro, a prima vista, che LDA e STA operano sul "contenuto" delle locazioni di memoria, ma viene palesemente distinto il verso del trasferimento.
Nell'ASSEMBLY Z80 è vero il contrario, ma vale sempre una convenzione nel formato delle istruzioni:

OPCODE DESTINAZIONE, SORGENTE

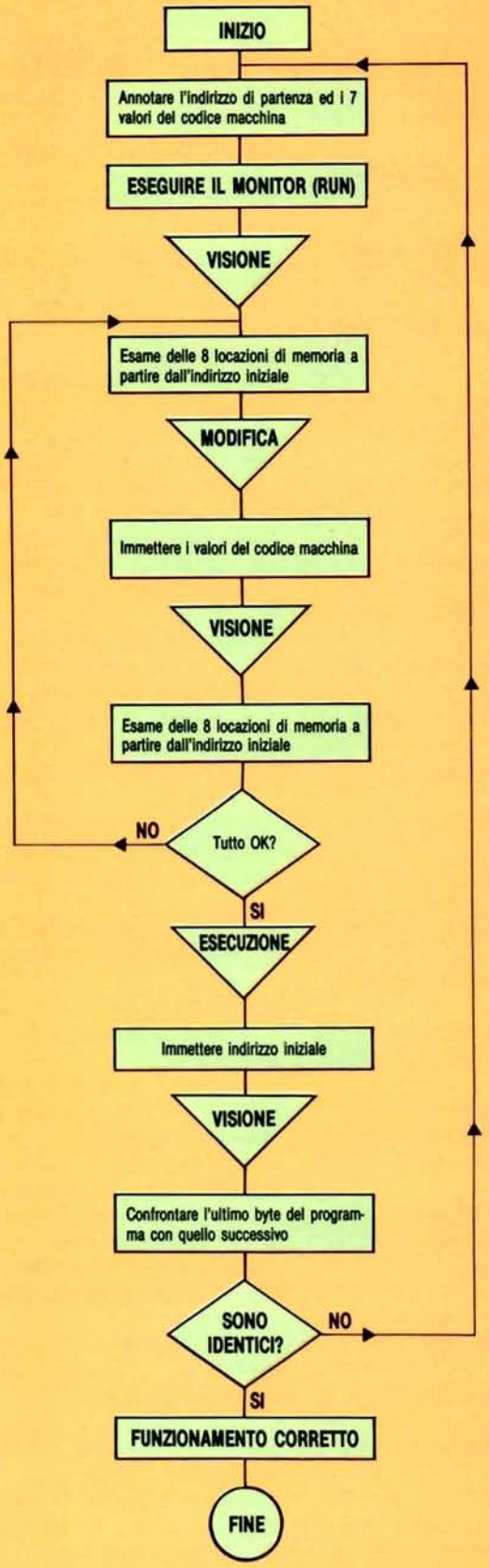
e, qualora le operazioni si riferiscano al contenuto di locazioni di memoria, la destinazione o la sorgente viene racchiusa tra parentesi.

Le poche istruzioni presentate formano un programma completo, secondo logica, ma le useremo come si trattasse di un sottoprogramma, accodandovi l'equivalente dell'istruzione RETURN del BASIC, ossia: RTS (per il 6502) e RET (per lo Z80).
Volendo usare la nostra subroutine, dobbiamo necessariamente convertirla in codice macchina, depositarla in qualche parte di memoria, e farla eseguire dal microprocessore. Possiamo ottenere queste ultime due operazioni con un programma "monitor" (vedere a pag. 558), ma dobbiamo prima effettuare la conversione e decidere dove memorizzare il programma.
Queste sono operazioni abbastanza insolite per un programmatore abituato al BASIC, dove l'effettiva collocazione del programma non riveste alcun interesse, dato che lo si esegue con un semplice RUN.
Un programma in codice macchina può essere memorizzato ed eseguito ovunque, nella RAM, ma alcune locazioni sono preferibili ad altre e queste variano a seconda della marca di computer; ecco dunque il perché delle molteplici versioni che seguono:

6502		
Indirizzo di memoria	**Codice macchina**	**Linguaggio ASSEMBLY**
COMMODORE 64		
$0350	AD 56 03	LDA $0356
$0353	8D 57 03	STA $0357
$0356	60	RTS
BBC MICRO		
$0070	AD 76 00	LDA $0076
$0073	8D 77 00	STA $0077
$0076	60	RTS
Z80		
16K SPECTRUM		
$7FA0	3A A6 7F	LD A,($7FA6)
$7FA3	32 A7 7F	LD ($7FA7),A
$7FA6	C9	RET
48K SPECTRUM		
$FFA0	3A A6 FF	LD A,($FFA6)
$FFA3	32 A7 FF	LD ($FFA7),A
$FFA6	C9	RET

Si noti che ciascuna versione del programma non fa che copiare il contenuto dell'ultima locazione di memoria, occupata dallo stesso programma, in quella successiva. Nel programma per lo Spectrum con 48 Kbyte, ad esempio, è copiato il contenuto di $FFA6 in $FFA7. Si noti, ancora, che gli indirizzi compaiono secondo il formato hi-lo (per nostra comodità) nel listato ASSEMBLY, mentre nel formato lo-hi nel codice macchina.
Nell'ASSEMBLY Z80 viene usato, in entrambi i casi, il codice mnemonico LD, sebbene la direzione del trasferimento sia diversa: l'opcode 3A copia i dati nell'accumulatore, mentre l'opcode 32 li estrae.

USO DEL PROGRAMMA MONITOR



Il programma alla pagina seguente permette la **Modifica**, la **Visione** e l'**Esecuzione** di quanto contenuto nella memoria del computer. Dopo aver selezionato una di queste funzioni, viene chiesto un indirizzo esadecimale. Questo serve come:
1. Indirizzo di partenza per le modifiche, oppure
2. Indirizzo di partenza per la visualizzazione, oppure
3. Indirizzo dal quale far partire l'esecuzione di un programma in codice macchina.

Digitando una X, anziché un numero, riporta il monitor a "livello comandi", dove una pressione del tasto Q termina l'esecuzione del programma.

Avendo selezionato **Modifica**, e fornito l'indirizzo iniziale, questo viene visualizzato nuovamente sullo schermo, seguito da un punto interrogativo: si digiti il nuovo valore esadecimale da depositare in memoria, terminando con la pressione del tasto RETURN. Automaticamente, il programma passa all'indirizzo successivo, ed il processo si ripete finché non si digita una X anziché un valore numerico, provocando il ritorno al "livello comandi".

Caricamento in memoria

Per depositare il programma in codice macchina, relativo a ciascun apparecchio, si prenda nota dell'indirizzo di partenza e dei valori dei sette byte di codice macchina (per il Commodore 64, ad esempio, sono rispettivamente: $0350, AD, 56, 03, 8D, 57, 03, 60): questi andranno forniti al programma monitor.
1. **Lanciare** il programma Monitor e procedere alla **Visione** del contenuto dei byte che iniziano all'indirizzo di partenza (da $0350 a $0357 sul Commodore)
2. Selezionare **Modifica**, immettere l'indirizzo ed i valori precedentemente annotati, che formano il programma in codice macchina
3. Selezionare **Visione** per controllare che detti valori siano stati correttamente riportati in memoria
4. Selezionare **Esecuzione** ed immettere l'indirizzo di partenza. Non accade nulla di visibile
5. Selezionare **Visione** ed ispezionare le locazioni di memoria: si troverà che il contenuto dell'ultimo byte del programma è stato copiato in quello successivo

PROGRAMMA MONITOR SPECTRUM

```
48 REM*************************************
49 REM*                                   *
50 REM*          IMC - MONITOR 1          *
51 REM*  --- VERSIONE PER LO SPECTRUM ---  *
52 REM*       (SALVARE IL PROGRAMMA       *
53 REM*        PRIMA DI ESEGUIRLO)        *
54 REM*                                   *
55 REM*************************************
100 GOSUB 1000  : REM  *INIT*
200 CLS
300 PRINT " **COMANDI DELL'IMC MONITOR 1**"
400 FOR P=1 TO LT:PRINT O$(P):NEXT P
500 FOR Z = 0 TO 1 STEP 0
550 GOSUB 2000   : REM *IMMISSIONE*
600 GOSUB (4500+CM*500)
650 NEXT Z
700 STOP
750 REM +++ FINE PROGRAMMA PRINCIPALE +++
1000 REM **** SUBROUTINE INIT ****
1050 LET LT=4:DIM C$(LT):DIM O$(LT,24):DIM
     I$(16)
1100 LET I$="0123456789ABCDEF"
1150 LET C$="MVEQ":LET C1=-48:LET C2=
     10-CODE(C$(1))
1200 LET O$(1)="          M - MODIFICA ALLA
     MEMORIA"
1220 LET O$(2)="          V - VISIONE DELLA
     MEMORIA"
1240 LET O$(3)="          E - ESECUZIONE DEL
     CODICE"
1260 LET O$(4)="          Q - FINE LAVORO"
1300 RETURN
2000 REM **** SUBROUTINE IMMISSIONE ****
2100 FOR P = 0 TO 1 STEP 0
2150 PRINT:PRINT"COMANDO ??"
2190 IF INKEY$<>"" THEN GO TO 2190
2200 LET A$=INKEY$:IF A$="" THEN GO TO 2200
2250 FOR J=1 TO LT
2300 IF A$=C$(J) THEN LET CM=J:LET J=LT:LET
     P=2
2350 NEXT J:NEXT P:IF A$="Q" THEN RETURN
2400 PRINT O$(CM)
2450 FOR P=0 TO 1 STEP 0
2500 INPUT"INDIRIZZO IN HEX (X=FINE)";A$
2550 GOSUB 5200  : REM CONTROLLA E AGGIUSTA
2600 NEXT P:IF A$="X" THEN LET CM=0
2650 RETURN
3000 REM *** SUBROUTINE CONVERSIONE BYTE ***
3010 LET HB=INT(N/16):LET LB=N-HB*16
3020 LET B$=I$(HB+1)+I$(LB+1)
3030 RETURN
3100 REM **** SUBROUTINE DEC - HEX ****
3110 IF NN<256 THEN LET N=NN* GOSUB 3000:LET
     H$ = "00"+B$:RETURN
3120 LET HI=INT(NN/256):LET LO=NN-256*HI
3130 LET N=HI:GOSUB 3000:LET H$=B$
3140 LET N=LO:GOSUB 3000:LET H$=H$+B$
3150 RETURN
4000 REM **** SUBROUTINE HEX - DEC ****
4050 LET RX=1:LET DN=0:LET HL=LEN(H$): IF
     (HL<1) OR (HL>4) THEN LET DN=-1:
     RETURN
4100 FOR H=HL TO 1 STEP -1
4150 LETD$=H$(H)
4200 LET V=CODE(D$)+C1*(D$>="0" AND D$<="9")+
     C2*(D$>="A" AND D$<="F")
4250 IF V>15 THEN LET DN=-1:LET H=1:NEXT H:
     RETURN
4300 LET DN=DN+V*RX:LET RX=RX*16
4350 NEXT H:RETURN
4500 REM *** SUBROUTINE FITTIZZIA ***
4550 RETURN
5000 REM *** SUBROUTINE DI MODIFICA ***
5020 FOR P=0 TO 1 STEP 0
5040 PRINT A$;:INPUT "NUOVO VALORE HEX (X=
     FINE) ?";V$
5050 PRINT V$
5060 GOSUB 5340: REM **VERIFICA ED ESEGUI**
5080 NEXT P:RETURN
5200 REM *** SUBR. CONTROLLA E AGGIUSTA ***
5220 IF A$="X" THEN LET P=2:RETURN
5240 LET LL=LEN(A$):IF LL>4 THEN RETURN
5260 LET H$=A$:GOSUB 4000
5280 IF DN>=0 THEN LET P=2:LET N=DN
5300 LET A$=A$+"    ":IF LL<4 THEN LET A$=
     "0000" (TO 4-LL) + A$
5320 RETURN
5340 REM *** SUBR. VERIFICA ED ESEGUI ***
5360 IF V$="X" THEN LET P=2:RETURN
5380 LET H$=V$:GOSUB 4000
5400 IF (DN<0) OR (DN>255) THEN RETURN
5420 POKE NN,DN
5440 LET NN=NN+1:IF NN>65535 THEN LET
     P=2:RETURN
5460 GOSUB 3100 : REM ** DEC - HEX **
5480 LET A$=H$+ "    ":RETURN
5500 REM *** SUBRUOTINE VISIONE ***
5520 FOR P=0 TO 1 STEP 0
5540 INPUT "NUMERO (DECIMALE) DI BYTE (X=
     FINE)";N$:IF N$="X" THEN LET P=2:NEXT P:
     RETURN
5560 LET BN=VAL(N$):IF (BN>0) AND (BN+
     NN<65536) THEN LET P=2
5580 NEXT P
5600 FOR B=NN TO (NN+BN-1) STEP 4
5620 LET L$="":LET NN=B:GOSUB 3100
5640 PRINT H$;TAB(6);
5660 FOR C=0 TO 3
5680 LET N=PEEK(B+C):LET K$="."
5700 GOSUB 3000 : REM ** DEC - HEX ***
5720 PRINT TAB(6+4*C);B$;
5740 IF N=0 THEN LET K$="■"
5760 IF (N>31) AND (N<128) THEN LET K$=
     CHR$(N)
5780 LET L$=L$+K$
5800 NEXT C
5820 PRINT TAB(26);L$
5840 NEXT B:RETURN
6000 REM *** SUBROUTINE ESECUZIONE ***
6050 RANDOMIZE USR(NN):RETURN
6500 REM ** SUBROUTINE DI FINE LAVORO **
6550 PRINT TAB(5);"■■■■ FINE DEL
     PROGRAMMA ■■■■ "
6600 LET Z=2:RETURN
```

BBC MICRO

```
39 REM*************************************
40 REM*          IMC  MONITOR 1           *
41 REM* --------- VERSIONE BBC ----------  *
42 REM* MODIFICARE LA VERSIONE SPECTRUM    *
43 REM* COME SEGUE:                       *
44 REM*                                   *
45 REM* SOSTITUIRE: CODE(   CON: ASC(     *
46 REM*                                   *
47 REM* AGGIUNGERE, TOGLIERE O MODIFICARE *
48 REM* LE SEGUENTI LINEE:                *
49 REM*                                   *
50 REM*************************************
60 *TV 255
70 MODE 7
200 PRINT CHR$(147);CHR$(142)
600 ON CM GOSUB 5000,5500,6000,6500
1050 LT=4:DIM C$(LT),O$(LT)
1150 C$(1)="M":C$(2)="V":C$(3)="E":C$(4)="Q":
     C1=48:C2=ASC(C$(1))-10
2190 --------- TOGLIERE QUESTA LINEA ------
3020 B$=MID$(I$,HB+1,1)+MID$(I$,LB+1,1)
4150 D$=MID$(H$,H,1)
4500 --------- TOGLIERE QUESTA LINEA ------
4550 --------- TOGLIERE QUESTA LINEA ------
5050 --------- TOGLIERE QUESTA LINEA ------
5300 A$=A$+ "    ":IF LL<4 THEN A$=
     LEFT$("0000",4-LL)+A$
5420 ?(NN)=DN
5680 N=?(B+C):K$="."
5740 IF N=13 THEN K$=CHR$(255)
6050 CALL NN:RETURN
6600 Z=2:RETURN
```

COMMODORE 64

```
39 REM*************************************
40 REM*        IMC  MONITOR 1             *
41 REM* --- VERSIONE COMMODORE 64 -------  *
42 REM* MODIFICARE LA VERSIONE SPECTRUM    *
43 REM* COME SEGUE:                       *
44 REM*                                   *
45 REM* SOSTITUIRE TUTTE LE LET P=2 CON    *
46 REM*             P=1                   *
47 REM*                                   *
48 REM* SOSTITUIRE: CODE(   CON: ASC(     *
49 REM*                                   *
50 REM*
51 REM* AGGIUNGERE, TOGLIERE O MODIFICARE *
52 REM* LE SEGUENTI LINEE:                *
53 REM*                                   *
54 REM*************************************
200 PRINT CHR$(147);CHR$(142)
600 ON CM GOSUB 5000,5500,6000,6500
1050 LT=4:DIM C$(LT),O$(LT)
1150 C$(1)="M":C$(2)="V":C$(3)="E":C$(4)="Q":
     C1=48:C2=ASC(C$(1))-10
2190 --------- TOGLIERE QUESTA LINEA ------
3020 B$=MID$(I$,HB+1,1)+MID$(I$,LB+1,1)
4150 D$=MID$(H$,H,1)
4500 --------- TOGLIERE QUESTA LINEA ------
4550 --------- TOGLIERE QUESTA LINEA ------
5050 --------- TOGLIERE QUESTA LINEA ------
5300 A$=A$+ "    ":IF LL<4 THEN A$=
     LEFT$("00 00",4-LL)+A$
5740 IF N=0 THEN K$=CHR$(122)
6050 SYS(NN):RETURN
6600 Z=1:RETURN
```

Questo programma vi permetterà di far apparire i contenuti della memoria, cambiarli, ed eseguire un programma già registrato in linguaggio (o codice) macchina

Le menti della Commodore

In pochi mesi, milioni di home computer venduti in tutto il mondo

Il segreto del grande successo della Commodore risiede nella perfetta organizzazione del sistema produttivo internazionale. Le forniture dei semiconduttori necessari all'assemblaggio degli apparecchi non costituiscono un problema, ma, anzi, un punto di forza per la Commodore, che si serve di ditte "imparentate". La Commodore infatti è un'organizzazione molto vasta e riesce talvolta ad ottenere, grazie alle enormi quantità impiegate, componenti elettronici a prezzi eccezionalmente bassi, rispetto alla concorrenza.
Molta della fortuna economica della Commodore proviene dal primo computer prodotto, il Personal Electronic Transactor, meglio noto come PET. Il progettista di questo apparecchio, Chuck Peddle, ha preso, a suo tempo, tre importanti decisioni. La prima fu di adoperare il microprocessore 6502; la seconda fu di orientarsi verso il BASIC della Microsoft; la terza fu quella di dotare l'apparecchio di un buon programma "editor", che facilitò la stesura di programmi, rispetto ai computer della concorrenza (spesso poco più che schede sperimentali, con le quali "giocavano" gli appassionati, agli inizi degli anni '70). Ancora oggi, comunque, nei computer considerati della generazione "successiva" (anche il PC IBM!) non viene fornito in dotazione uno "screen editor" altrettanto buono.
La Commodore è partita molto avvantaggiata nel campo dei microcomputer. Infatti, la MOS Technology, la ditta produttrice in esclusiva del chip 6502, era di proprietà della Commodore e, poiché sia l'Apple che il TRS-80 della Tandy erano basati su tale chip, essa poteva vantare un'ottima posizione di controllo. Tuttavia, la nascita del PET non fu certo indolore: Jack Tramiel, presidente della Commodore, insisteva, contro ogni parere di Chuck Peddle, che le memorie per l'apparecchio provenissero anch'esse dalla MOS Technology. Ciò provocò una tale frattura tra i due, che Peddle abbandonò la Commodore.
Il PET rappresenta ancora oggi un bel pezzo di macchinario. Originariamente, il suo involucro era metallico (come del resto la maggior parte dell'arredamento per uffici di un tempo). Successivamente, venne adoperata una robusta plastica, più al passo coi tempi. L'ultima versione, denominata serie 8000, è capace di gestire ben 22 Megabyte di disco fisso. Nonostante l'avvento del chip 8080, dello Z80, dei più recenti microprocessori a 16 bit e del fatto che, commercialmente parlando, l'apparecchio sia "vecchio", esso ottiene ancora un pregevole successo di mercato.
Ciò può anche essere dovuto alla politica adottata dalla Commodore nei confronti della clientela: la Commodore, la Apple e la Tandy sono state le prime ditte a produrre personal computer dal costo accessibile e dall'impiego molto semplice. La Commodore, però, è per certi versi rimasta "fedele" ai propri clienti, non apportando, nel corso degli anni, sostanziali modifiche all'hardware, tanto che la gran parte del software scritto per il PET "gira" ancora sulle macchine più recenti e la versione 2.0 del BASIC Microsoft è usata ancora diffusamente oggi, come ai tempi di Chuck Peddle.
Sfortunatamente per gli hobbysti più esperti, la versione 2.0 del BASIC Microsoft è stata sviluppata quando la grafica e gli effetti sonori erano un vero lusso sui microcomputer. Di conseguenza, benché il Vic-20 ed il Commodore 64 siano dotati di un'ottima grafica e di una eccellente generazione di effetti sonori, il BASIC montato su queste macchine è totalmente sprovvisto di adeguati comandi per queste funzioni, obbligando a laboriose contorsioni, mediante PEEK e POKE. Il problema non si presenta, tuttavia, se si usano i programmi su cartucce, di cui esiste una vasta scelta.
Una delle ragioni che ha salvato la Commodore dal naufragio commerciale, subìto da una gran parte dei concorrenti gettatisi nel marasma dei programmi di gioco, è forse stata il costante approvvigionamento di componenti. La Texas e la Mattel si sono ritirate dal mercato degli home computer e la Atari è parsa boccheggiare più di una volta nel corso degli ultimi anni. La Commodore, riuscendo a tener bassi i costi di produzione, è stata capace di offrire prodotti economici, pur avendo un buon margine di profitto. La modernità degli impianti di produzione e la notevole esperienza accumulata in due decenni di lavoro nel campo, hanno senz'altro contribuito a questa solidità, assieme alla possibilità di rifornirsi di componenti a prez-



per concessione della MICROSCOPE



per concessione della VNU

Jack Tramiel
La guida della CBM è stata, fin dalla sua nascita, nelle capaci mani di Jack Tramiel, fondatore e presidente della società. Egli si è ritirato dagli affari alla fine del 1983 e molti sentiranno la sua assenza dalla scena

Chuck Peddle
È l'uomo che ha inventato il PET ed il microprocessore 6502. Ha lasciato la Commodore per fondare una propria ditta, produttrice del computer Sirius, a 16 bit

zo concorrenziale. Pare che, secondo alcune stime approssimative, il costo di produzione di un Commodore 64 si aggiri sul centinaio di migliaia di lire.
La fortuna non ha sempre arriso alla Commodore. Anzi, per ben due volte questo impero ha vacillato fortemente, rischiando il fallimento. Jack Tramiel, un giovane polacco approdato negli Stati Uniti dopo esser scampato ad Auschwitz, formò la CBM (Commodore Business Machines), nel 1955. La sede era a Toronto, in Canada, e, inizialmente, la ditta si occupava modestamente dell'assemblaggio di macchine da scrivere, su licenza di una ditta cecoslovacca. Si dice che la sigla sia stata scelta per la somiglianza con quella già molto nota della IBM.
Nel 1975, dopo venti anni di attività nel settore delle apparecchiature per ufficio, la CBM si trovò in grande difficoltà, a causa dell'invasione di calcolatrici tascabili, principalmente giapponesi. Ma Tramiel, temuto quanto rispettato per i suoi metodi di condurre le trattative, riuscì a sopravvivere. Nel 1976 intravide le vaste potenzialità commerciali dei microcomputer e riuscì ad assicurarsi la collaborazione di Chuck Peddle. Nel giro di un solo decennio il valore della CBM è cresciuto di almeno 50 volte.
Un magnifico esempio di riconversione industriale; ma, se esiste un punto debole nell'intera struttura, questo è forse nello sviluppo di nuovi prodotti. La filosofia adottata da Tramiel è di «vendere alle masse, non alle classi», ed è sua ferma convinzione che il pubblico acquista sempre ciò che rappresenta il miglior compromesso tra prestazioni e costo. Pertanto, può darsi che impostando una produzione massiccia su apparecchi economici, ostacoli l'assunzione di nuove tecnologie, più avanzate. Alla fine del 1982, addirittura, l'intero staff della Commodore addetto allo sviluppo ha lasciato la ditta, che da allora si appoggia a terzi, acquistando i frutti delle ricerche effettuate da altre compagnie. Sono prospettati accordi con ditte giapponesi per la produzione di unità a dischi e di hardware per i computer della "quinta generazione".
Nel 1984, la Commodore si è dedicata alla produzione di due nuovi apparecchi, basati sul nuovo processore 7501: il 264 ed il V364. Quest'ultimo è dotato di un modulo di sintetizzazione vocale, un aspetto senz'altro molto innovativo nel settore.

LE PIETRE MILIARI DELLA COMMODORE



1982
Esce il modello **CBM 700** per le applicazioni tipicamente commerciali



1984
Viene lanciato il modello **SX64**, la versione portatile del Commodore 64, dotato di unità a dischetti e di monitor a colori incorporati

1977
Nasce il **Commodore PET**, il primo personal computer prodotto in massa. Ancora oggi ha un discreto successo, nelle versioni aggiornate





1979
Entra in produzione il Vic-20. Nonostante alcune forti limitazioni e la grande concorrenza ha tuttora uno strepitoso successo



1981
Nel **CBM 8032** una grande innovazione rispetto ai modelli precedenti: lo schermo ad 80 colonne



1983
È l'anno del **Commodore 64**, che migliora notevolmente le prestazioni del precedente home computer Vic-20



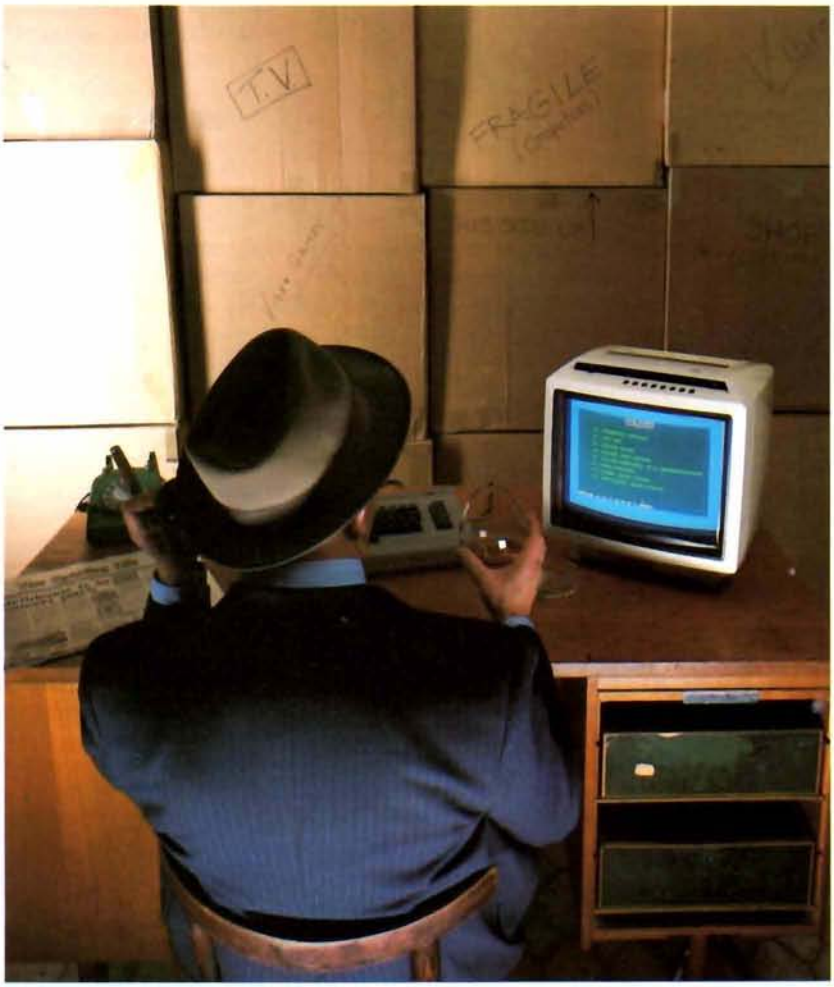
1980
Viene lanciato il **SuperPET** (o CBM 9000): un tentativo di produrre un apparecchio commerciale derivato dal PET originale

IAN McKINNELL, CHRIS STEVENS e per concessione della COMMODORE

Contabilità e computer

Anche un home computer può tenere in ordine i conti di una piccola attività commerciale

Per risolvere un qualsiasi problema contabile, un sistema semplificato deve essere in grado di scomporre le varie fonti d'entrata e di uscita dell'attività commerciale. Ad un commerciante può non interessare un riassunto totale di quanto è stato incassato o speso in un dato periodo, quanto, invece, un'analisi delle varie voci che compongono i dati finali. Per esempio, sorge la necessità di sapere quanto viene speso per affitti, viaggi, piccole spese, cancelleria e così via.
Per soddisfare questo tipo di esigenza, quindi, il programma deve dare all'utente la possibilità di registrare le entrate e le uscite sotto varie "voci". Il Mastro Generale, o, con termine più diffuso, il "Piano dei Conti" è lo strumento contabile che raggruppa tutte le voci in "conti" e "sottoconti" e costituisce il cuore di qualsiasi sistema di gestione contabile.
Esiste una discreta gamma di programmi per la gestione della contabilità, ma una prima distinzione è tra quelli basati sull'impiego di unità a dischi e quelli, invece, basati sull'uso di registratori a cassette. In quest'ultimo caso, ovviamente, possono presentarsi limitazioni piuttosto rilevanti, soprattutto per la quantità dei conti relativi ai movimenti ed al numero delle stesse registrazioni contabili. In linea di massima, il "Piano dei Conti" dei programmi di questo genere può prevedere una settantina di "voci" (mentre nei programmi che utilizzano dischetti il numero può essere dell'ordine di qualche centinaio). Spesso, inoltre, il Piano dei Conti è pre-strutturato, ossia il numero associato a ciascun conto è, in alcuni casi, obbligato. Una simile organizzazione permette al programma di decidere più facilmente quali operazioni siano o meno consentite su ciascun sottoconto. La pre-strutturazione impone però alcune limitazioni alla struttura del mastro che vogliamo creare.
I numeri da 01 a 19, ad esempio, possono rappresentare i conti economici e si riferiscono alle entrate provenienti da vendite e alle uscite per acquisti (di prodotti per la rivendita o di scorte). Tuttavia, il commerciante potrebbe aver bisogno di usare solo quattro dei 19 titoli di conto commerciale disponibili. Un singolo conto, ad esempio il numero 01, potrebbe essere chiamato "Incassi" e tutte le entrate potrebbero venir inserite (il termine tecnico è "accreditate") in questo conto. Il conto 01 conterrebbe quindi i totali di tutte le entrate.
D'altra parte, il suddetto commerciante potrebbe aver bisogno di tener separati i totali, ad esempio, di 5 diverse categorie di prodotti venduti: ciò significherebbe aprire 5 conti d'entrata (da 01 a 05). Sarebbe quindi necessario sviluppare alcune procedure di verifica che permettano di registrare ciascuna vendita nello specifico conto.
I numeri di conto da 20 a 49 potrebbero riferirsi ai profitti ed alle perdite. Questa sezione comprende tutte le differenti voci di spesa che l'attività commerciale comporta. Il commerciante dovrà creare titoli di conto del tipo: affitti, stipendi, elettricità, spese bancarie, interessi bancari, cancelleria, pubblicità e telefono.
I numeri di conto da 50 a 79 potrebbero, infine, esser riservati ai conti necessari al bilancio contabile. Questi evidenziano in ogni momento lo stato finanziario globale dell'attività commerciale. Le diverse voci comprenderanno, ad esempio, i conti patrimoniali (che definiscono il valore di tutte le



MARCUS WILSON-SMITH

Un aiuto in... cassa!
Anche i piccoli home computer, che fanno uso di cassette, possono tenere una contabilità di dimensioni ridotte. La maggior parte dei pacchetti software commerciali per queste macchine presenta obiettivi piuttosto limitati, specialmente per quanto concerne il numero di "conti", ossia delle voci di entrata e di uscita, disponibili

proprietà riferibili all'attività commerciale), l'attuale conto bancario (o i conti, se sono più di uno), il conto I.V.A. ed il conto di controllo dei creditori (che mostra quanto è dovuto a tutti i fornitori), ecc.
In un sistema a mastro sviluppato manualmente, il commerciante deve aggiornare il brogliaccio delle vendite e degli acquisti, che contiene il sunto totale giornaliero degli incassi e delle spese. Il menu di un semplice programma per computer è normalmente abbastanza conciso, ma chiaro:

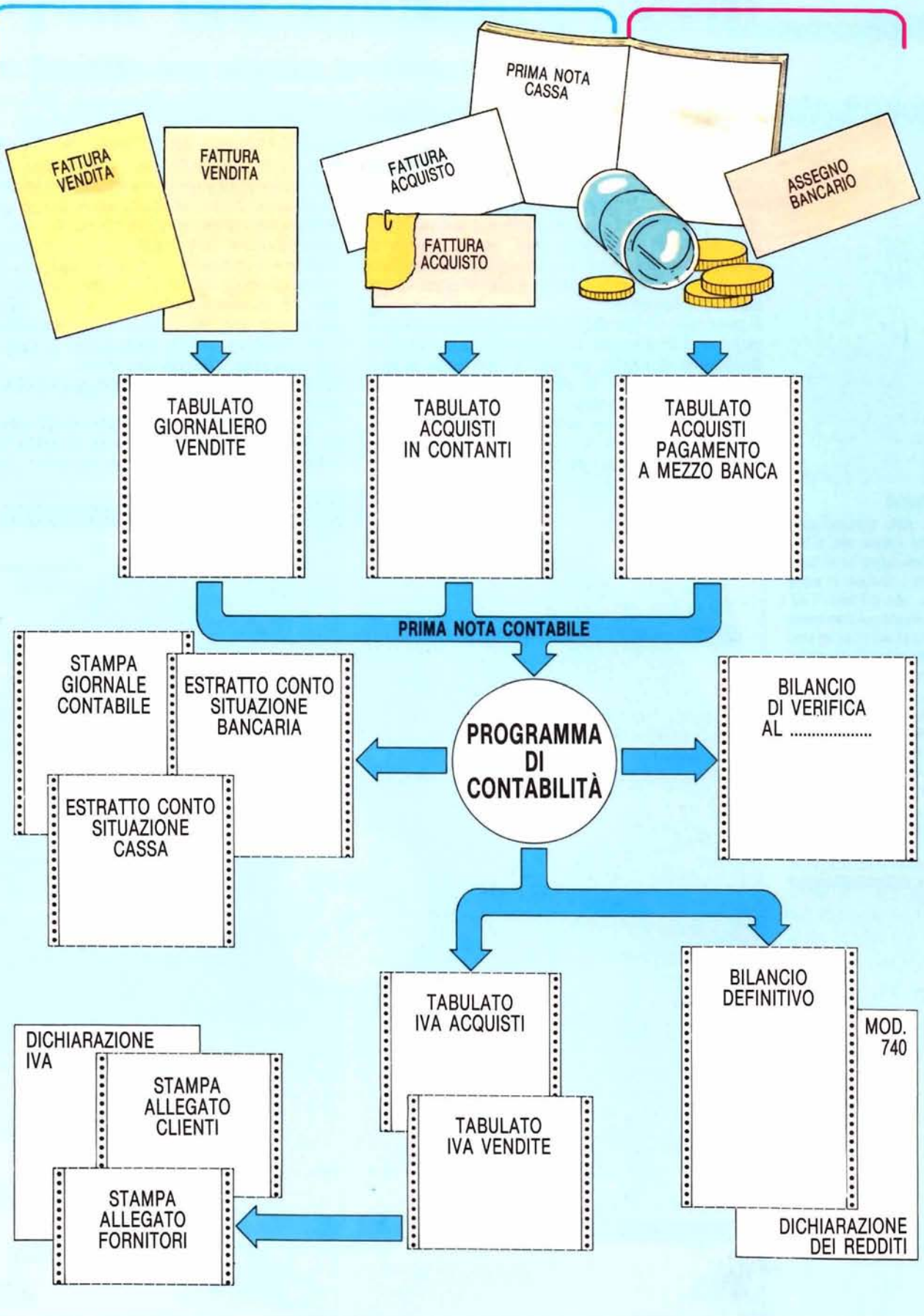
1) Immissione Prime Note
2) Stampa delle registrazioni
3) Stampa dei registri IVA
4) Stampa statistiche

L'opzione 1, "Immissione Prime Note" è stata progettata per registrare un evento contabile (un incasso o una spesa), addebitandone l'importo ai conti interessati. I dati indispensabili sono la data di registrazione, il numero del documento cui tale movimento si riferisce, la percentuale I.V.A. applicabile ed, ovviamente, l'imponibile.
A seconda del livello di completezza del programma, gli importi relativi all'I.V.A. vengono suddivisi in gruppi, in base all'aliquota. Anche la causale del movimento, in certi casi, può essere ricavata da una "tabella" (permettendo così una gestione automatica degli scarichi), oppure indicata per esteso per ciascuna registrazione.
L'estensione e la quantità delle informazioni collegate a ciascun conto variano enormemente da programma a programma. In alcuni, il numero di caratteri a disposizione per l'intestazione del conto è limitato a poco più di una dozzina, consentendo soltanto di indicare, ad esempio, la ragione sociale del cliente o del fornitore. In altri, invece, esiste la possibilità di includere anche altri dati: l'indirizzo completo, il codice fiscale o la partita I.V.A., e, magari, un codice per usi speciali (ad esempio, il massimo credito da accordare ad un cliente). Ovviamente, occorre sempre tener presente che la quantità di memoria disponibile impone limiti abbastanza precisi: un maggior numero di informazioni per ciascun conto si traduce in un minor numero di conti e viceversa.
Altre funzioni, quali la stampa dei registri I.V.A., sono condizionate dalle leggi in vigore: la quantità ed il tipo di informazioni presentate deve corrispondere esattamente a precise norme di legge. Altre ancora sono invece puramente facoltative dal punto di vista fiscale, ma possono risultare estremamente comode, ad esempio: un elenco dei debitori, magari suddiviso per periodi a 30, a 60 ed a 90 giorni oppure un raffronto tra gli incassi di un mese e quello successivo.
Naturalmente, i programmi che sfruttano le prestazioni, molto più flessibili, di unità a dischetti, possono sbizzarrirsi in più larga misura sul livello generale di sofisticazione di tutte le funzioni viste fin qui. Il numero di conti e sottoconti è, in genere, composto da almeno 6 cifre, due delle quali, secondo i canoni tradizionali, individuano il conto (ad esempio, quello dei Clienti), mentre i restanti quattro consentono di distinguere ben 9.999 sottoconti. Ognuno di questi, grazie ad un apposito sottoprogramma, può essere richiamato per esaminare i progressivi del "dare" e dell'"avere", assieme al "saldo". In effetti, il Piano dei Conti può diventare qualcosa di molto simile all'archivio di un database: da esso possono venir estratte informazioni utili anche per altri scopi. Ad esempio, può esser prevista una stampa automatica degli estratti conto dei soli Clienti che abbiano un pagamento in sospeso, magari completa di invito a regolare la propria posizione con sollecitudine! Oppure può esser comodo, dato che le informazioni sono già contenute nell'archivio, far stampare al computer una serie di etichette con l'indirizzo di gruppi selezionati di clienti, ai quali, per esempio, inviare una particolare offerta promozionale.
Ma torniamo all'argomento principale: la contabilità. Per offrire un quadro sintetico sulle funzioni di un programma di contabilità, si può operare una prima grande distinzione. Esistono "package" di contabilità generale "integrati" (ossia collegati con i moduli di fatturazione, di magazzino e, magari, di gestione degli ordini). Esistono invece programmi di contabilità non "integrati", nei quali lo scarico di magazzino, ad esempio, deve essere fatto semi-manualmente, ricorrendo ad un altro programma. Questo secondo metodo, benché sia certamente più scomodo del primo, permette sia di acquistare i vari programmi separatamente, costruendo "pezzo per pezzo" un sistema completo, ma con un investimento di capitale progressivo sia di poter sfruttare anche computer di piccole dimensioni, dunque più economici. Se, poi, i programmi vengono venduti non soltanto nella forma "oggetto" (ossia in quella eseguibile, ma non "listabile"), ma anche corredati del listato del programma, allora i più intraprendenti possono anche cimentarsi ad aggiungere nuove procedure, adatte a particolari necessità individuali. Una volta che si conosca il formato degli archivi (ossia, in termini più professionali, il "tracciato dei record"), infatti, non è difficile studiare e realizzare qualche modifica o qualche programma aggiuntivo. L'importante è prestare enorme attenzione a non interferire con la regolarità fiscale delle registrazioni contabili.
A questo proposito, vale la pena di ricordare che le leggi vigenti impongono la tenuta di una precisa documentazione in merito alle procedure adottate per i sistemi contabili computerizzati, come del resto sul tipo e lo scopo degli archivi su supporto magnetico.
Questa deve contenere una dettagliata descrizione dei procedimenti e dei criteri adoperati nei programmi e, possibilmente, un listato degli stessi, con tutte le "chiavi" necessarie alla corretta interpretazione di tutti i dati elaborati. Indicazioni su queste norme si possono trovare, ad esempio, nel D.P.R. n. 600 del 29 settembre 1973.

Facciamo i conti...

In un sistema contabile semplificato, come quello illustrato qui di fianco, le operazioni sono ridotte al minimo essenziale. Il primo elemento, di primaria importanza per qualunque tipo di contabilità, è la registrazione di "prima nota", ossia l'annotazione di ciascun evento contabile, completa di tutti i dettagli richiesti dalle norme fiscali: il numero della registrazione, la data, i conti coinvolti e l'importo dell'operazione.
In aggiunta alle stampe necessarie ai fini fiscali (quali, ad esempio, quella dei registri I.V.A.) un programma di contabilità può prevederne altre di estrema utilità: estratti conto dei clienti "in dare", o la situazione di cassa, o, ancora, la situazione del conto corrente bancario. Inoltre, può essere previsto un sottoprogramma per la compilazione di un bilancio di verifica, oltre a quello definitivo di fine anno

SCHEMA DI FUNZIONAMENTO DI UN PROGRAMMA DI CONTABILITÀ



TONY DUNCAN SMITH

Incompatibilità dei dischi

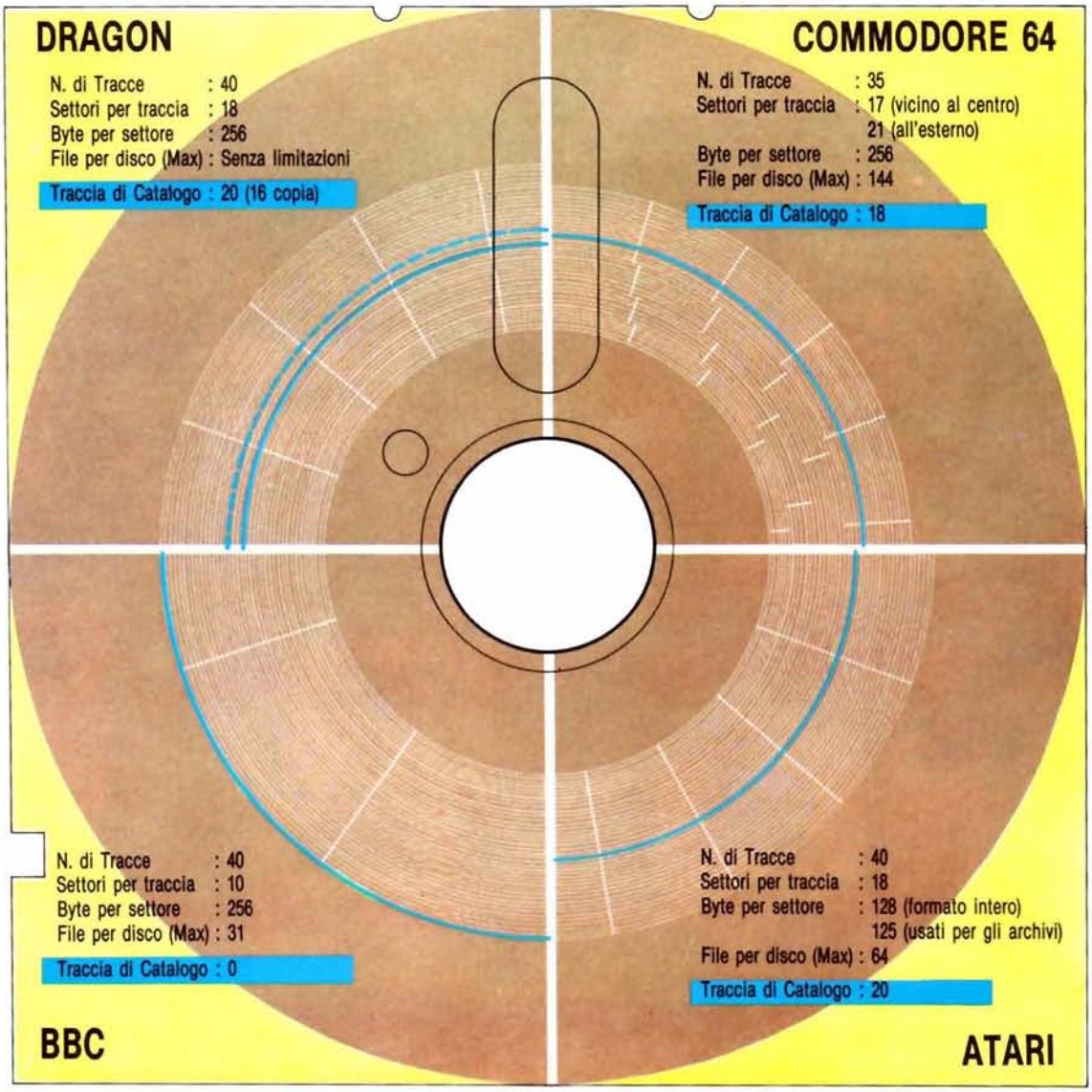
Raramente i dischetti adoperati da un computer sono utilizzabili da altri

Quando si acquista una scatola di dischetti nuovi questi non possono essere usati così come sono: occorre prima "formattarli". Sebbene i principi di base della formattazione siano sempre gli stessi, esistono varianti da un computer all'altro, che impediscono lo scambio di dischetti tra computer di marche diverse, anche se i dati registrati sono teoricamente leggibili.

Il processo di formattazione è paragonabile a ciò che si fa quando si tracciano delle linee su un foglio di carta, prima di scrivervi sopra. Per poter usare il dischetto, il computer traccia su di esso una serie di tracce concentriche, suddivise in settori a "fetta di torta". Per i dischetti a 5 pollici e 1/4 vengono generalmente usati due formati: 40 oppure 80 tracce. Alcune unità sono fornite di due testine di lettura/scrittura contrapposte, che consentono l'impiego di entrambe le facce del dischetto. Altre, invece, scrivono e leggono esclusivamente su di una sola faccia. Questi procedimenti di formattazione vengono indicati col termine di "soft-sectoring", in quanto i settori vengono creati dallo stesso programma di formattazione. Viceversa, nell'hard-sectoring sul dischetto viene praticata, in fase di lavorazione, una serie di piccoli fori sul margine interno del disco.

Questa tecnica è da tempo pressoché scomparsa.

L'ultima variabile che influisce sulla quantità di dati immagazzinabili è la densità dei dati sulla superficie magnetica: esistono dischetti



Quattro formati

Sebbene usino dischetti dello stesso tipo, il Micro BBC, il Dragon, il Commodore 64 e l'Atari formattano i dischetti in modo differente l'uno dall'altro. Il numero dei file che possono essere immagazzinati sul disco dipende dal sistema operativo. Da notare che l'elenco dei file presenti sul dischetto (la traccia di catalogo), si trova sulla traccia più esterna nei dischetti Atari e sulla traccia centrale negli altri. La posizione centrale è di solito la migliore perché la testina percorre, mediamente, un percorso più breve per spostarsi dalla traccia di catalogo alla traccia sulla quale si trovano i dati: l'intera operazione di accesso al disco risulta quindi più veloce

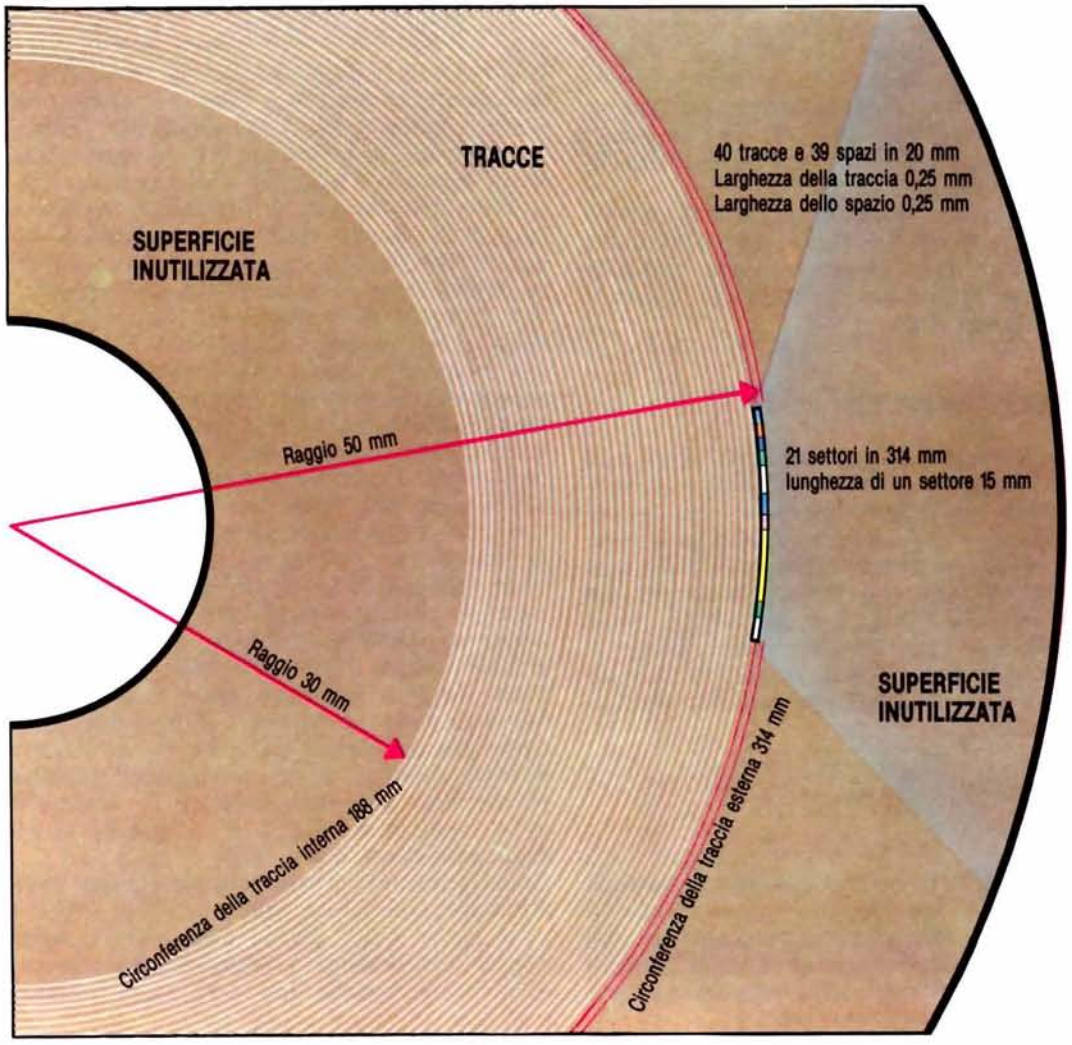
a "singola densità" e a "doppia densità". Tutto ciò spiega perché la capacità dei floppy disk possa variare da circa 90 Kbyte, nei dischetti a "singola faccia/singola densità", fino a 1,2 Megabyte nei dischetti a "doppia faccia/doppia densità".

Una volta formattato, un dischetto può essere usato per la registrazione di programmi e di informazioni. Quest'operazione non avviene in modo molto semplice: tutti i dati si trovano all'interno di file la cui gestione spetta al DOS (sistema operativo). Questo mantiene un elenco (Directory) di tutti i file del disco e utilizza i settori liberi per immagazzinare nuove informazioni. Fisicamente, un file può essere letteralmente "sparpagliato" tra quelli che già si trovano sul disco: il DOS usa un indice dei settori occupati da ciascun file, per concatenare i vari frammenti assieme.

Fino a poco tempo fa la formattazione era spesso molto diversa da un apparecchio all'altro, e le possibilità di utilizzare su una macchina un dischetto formattato su un'altra erano molto scarse. Fortunatamente, nei micro più recenti questa barriera sembra essere molto ridotta. Molte macchine adottano sistemi operativi standard come il CP/M e lo MS-DOS, quindi i file vengono organizzati allo stesso modo.

Tracce e settori

I dati vengono scritti sul disco in tracce concentriche. Ciascuna traccia è suddivisa in settori. Un settore contiene un blocco di dati al quale il DOS, automaticamente, accoda dei codici di controllo sugli errori e i dati di identificazione necessari per la gestione dei file e dei record



SETTORI DI TESTA

Segnale 1 di sincronizzazione
Permette di confrontare la velocità di lettura della testina con la velocità di rotazione

Numero di traccia
Identifica la traccia

Numero di settore
Identifica il settore

Checksum 1
Serve per il controllo degli errori nei settori di testa

Spazio 1
Divide i settori di testa dal blocco dati

Segnale 2 di sincronizzazione

Collegamento file
Individua il settore successivo del corrente file

Blocco dati
128, 256 o 512 byte di dati utente a seconda del sistema operativo in uso

Checksum 2
Serve per il controllo degli errori nei dati utente

Spazio 2
Separa un settore dal successivo

SETTORE BLOCCO DATI

Una risposta logica

Un riassunto degli argomenti fin qui appresi

È il momento di raccogliere le fila di quanto è stato trattato fin qui. I principi della logica si applicano, nella progettazione dell'hardware dei computer, per ottenere macchine capaci di eseguire particolari lavori. Abbiamo esaminato il progetto di un circuito semisommatore che, associato ad altri circuiti dello stesso tipo, permette di sommare numeri binari (vedere a pag. 528). Un tale circuito imita il metodo di addizione tipico degli uomini, prevedendo anche il passaggio dei riporti da una colonna alla successiva.
Nella progettazione di questo circuito abbiamo utilizzato tre elementi di base, chiamati porte logiche. Le funzioni logiche di queste porte sono deducibili dai nomi loro assegnati (AND, OR e NOT) e si possono ricavare dalle rispettive tavole della verità. La tavola della verità è un semplice sistema che permette di determinare l'uscita di un circuito in dipendenza da ogni possibile combinazione dei segnali applicati agli ingressi. Due ingressi generano quattro possibili combinazioni (2^2), tre ingressi ne danno otto (2^3) e così via.
Abbiamo visto, inoltre, come sia possibile collegare tra loro le porte logiche per ottenere alcuni particolari risultati. Anche questi circuiti più complessi possono essere descritti mediante tavole della verità.
In particolare, abbiamo progettato un circuito OR-Esclusivo a 5 porte, la cui uscita vale 1 (condizione vera) soltanto quando uno solo dei cinque ingressi è in condizione logica 1 (vedere a pag. 512).

Algebra Booleana

La combinazione degli elementi logici può essere formulata su carta mediante una serie di simboli, similmente a quanto accade nell'algebra normale. La branca della matematica che studia la rappresentazione della logica è conosciuta come Algebra Booleana, dal nome del matematico inglese George Boole (1815-64), che per primo ne ha definito i principi. Ciascuno dei tre elementi di base della logica viene rappresentato con un simbolo particolare:

A AND B	A.B
A OR B	A+B
NOT A	$\bar{A}$

Esistono leggi che regolano la manipolazione dei numeri in matematica e delle lettere in algebra, così come ci sono leggi speciali che regolano la semplificazione delle espressioni logiche. Le leggi dell'Algebra Booleana sono riassunte nella seguente tabella:

RELAZIONI SPECIALI	
Relazione	**Duale**
A.A = A	A+A = A
A.$\bar{A}$ = 0	A+$\bar{A}$ = 1
A.0 = 0	A+1 = 1
A.1 = A	A+0 = A
A.(A+B) = A	A+A.B = A
A.($\bar{A}$+B) = A.B	A+$\bar{A}$.B = A+B
LEGGE DI DE MORGAN	
1) $\overline{A+B} = \bar{A}.\bar{B}$	
2) $\overline{A.B} = \bar{A}+\bar{B}$	
LEGGE ASSOCIATIVA	
A.(B.C) = (A.B).C = A.B.C	
A+(B+C) = (A+B)+C = A+B+C	
LEGGE COMMUTATIVA	
A.B = B.A	
A+B = B+A	
LEGGE DISTRIBUTIVA	
A.(B+C) = A.B+A.C	

Usando queste regole è possibile semplificare le espressioni logiche e ridurre il numero di porte necessarie per il circuito finale. Oltre a questo metodo algebrico, per la semplificazione dei circuiti logici abbiamo anche trattato l'uso delle mappe di Karnaugh (vedere a pag. 572). Queste mappe rappresentano un significativo progresso: non possono sostituire completamente le semplificazioni algebriche, tuttavia permettono di ridurre la complessità del lavoro. Le mappe di Karnaugh, che in pratica sono un'estensione dei diagrammi di Venn (vedere a pag. 506), permettono di riunire a gruppi di due, quattro o otto, le espressioni estratte dalle tavole della verità. I gruppi ottenuti rappresentano espressioni Booleane più elementari e, pertanto, costituiscono una semplificazione delle espressioni stesse. In pratica ci si serve di una combinazione di mappe-K e di algebra per ottenere circuiti più efficienti.
Abbiamo anche approfondito l'uso delle operazioni logiche AND e OR nella programmazione BASIC. Abbiamo visto come questi comandi possano essere utilizzati per associare più condizioni nelle istruzioni BASIC IF...THEN e come queste operazioni logiche permettano al programmatore di "accendere" o "spegnere" singoli bit all'interno di un registro (vedere a pag. 526).
Come punto di arrivo di questa prima fase del corso, abbiamo usato tutte le nostre conoscenze sulle tavole della verità, sulle mappe di Karnaugh, sull'algebra Booleana e sulle porte logiche per progettare circuiti particolari (vedere a pag. 586).

Esercizi di riepilogo

1) Ad un gruppo "pop" è assicurato il successo di un brano musicale, se produrrà un buon "video" da trasmettere assieme al pezzo musicale e (AND) la casa discografica accluderà al disco un regalo gratuito, o (OR) se il gruppo parteciperà ad una trasmissione televisiva. Tracciare una tavola della verità che presenti tutti i possibili risultati. Se più eventi hanno identiche probabilità di verificarsi, quali saranno le possibilità del gruppo di ottenere un successo?

2) Una scuola sta organizzando un club di scacchi e le tessere del club devono essere ancora assegnate. Il comitato organizzatore decide che i tesseramenti verranno effettuati secondo i seguenti criteri:

I) Un rappresentante degli insegnanti
II) Uno studente del quarto o del quinto anno che segue il corso di matematica e quello di scienze biologiche
III) Uno studente del sesto anno che segue il corso di matematica oppure quello di scienze biologiche

Progettare un distributore automatico di tessere che funzioni premendo i pulsanti che meglio definiscono il futuro membro del club. I pulsanti sono:

A) Studente del quarto anno
B) Studente del quinto anno
C) Studente del sesto anno
D) Rappresentante degli insegnanti
E) Studente di matematica
F) Studente di scienze biologiche

Progettare un diagramma di circuito per la macchina.

3) Un microcomputer possiede un registro con indirizzo 23148 (decimale), usato per controllare lo schermo. Il bit 0 è quello meno significativo del registro, il bit 7 quello più significativo. Lo schermo può essere selezionato nel modo di alta risoluzione ponendo i bit 4 e 5 a 1. Poiché gli altri bit nel registro vengono usati per controllare altre funzioni, è necessario che il loro valore non venga alterato. Scrivere i comandi BASIC per:

a) Portare il video nel modo di alta risoluzione
b) Riportare il video alla risoluzione standard

4) La serratura della camera di sicurezza di una banca si basa su interruttori a codice. Il direttore, il suo delegato ed il capo cassiere posseggono tutti i codici. La porta della camera di sicurezza può essere aperta da due dei tre codici, in conformità alle seguenti restrizioni:

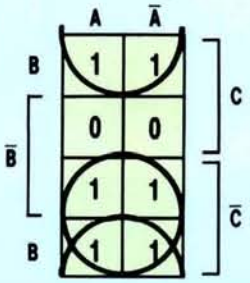
a) Tra le 9 e le 17 dei giorni feriali, la camera può essere aperta solo in presenza del capo cassiere
b) Nelle altre ore la camera può essere aperta solo dal direttore assieme ad una delle altre due persone

Progettare un circuito che controlli il sistema di apertura.

5) In un sistema di trasmissione di dati digitali in binario, che deve garantire un alto grado di affidabilità, ciascun bit viene inviato contemporaneamente su tre canali paralleli che, in arrivo, vengono esaminati da un apparecchio verificatore. Se non ci sono stati errori di trasmissione, tutti e tre i bit saranno uguali. In caso di errore, il verificatore correggerà l'errore su un canale e produrrà un unico output sulla base degli input che presentano lo stesso valore di bit. Progettare un circuito per il verificatore.

Risposte all'esercizio a pag. 587

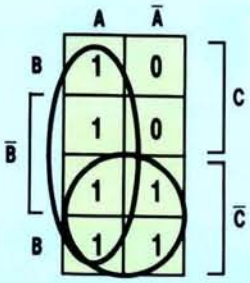
1a)



$= \bar{C} + B$
(oppure $= \overline{\bar{C}.\bar{B}}$)

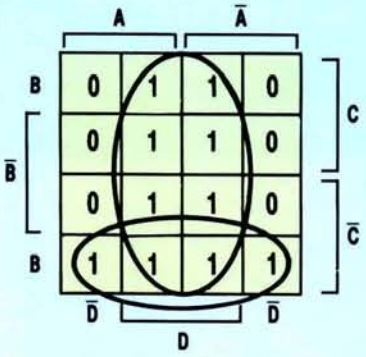
b)

$\overline{\bar{B} + C} + B.\bar{C} + A.C$
$= \bar{B}.\bar{C} + B.\bar{C} + A.C$ (de Morgan)



$= A + \bar{C}$
(oppure $= \bar{A}.C$)

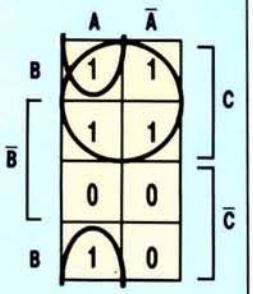
c)



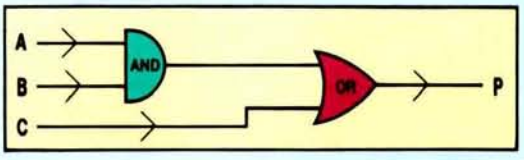
$= D + B.\bar{C}$

2) La tavola della verità è:

Decimale	A	B	C	P
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



Dalla mappa-k otteniamo l'espressione $P = C + A.B$.
Il circuito risultante sarà:



LIZ DIXON

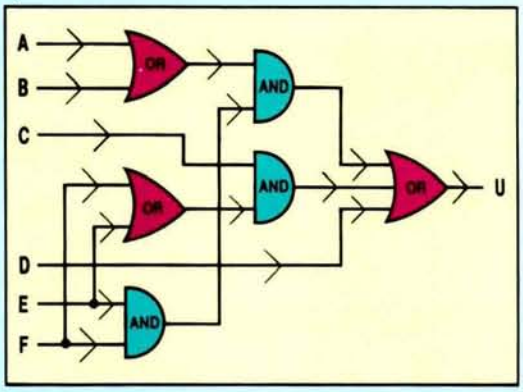
Risposte all'Esercizio di Riepilogo

1)

Buon Video	Regalo	Trasmissione TV	Successo del disco
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Probabilità di successo del disco = 5/8 = 62.5%

2)



3a) Per attivare l'alta risoluzione

POKE23148,PEEK(23148)OR48

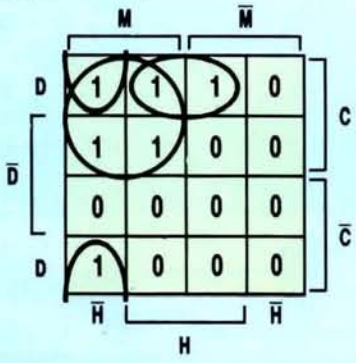
b) Per disattivare l'alta risoluzione

POKE23148,PEEK(23148)AND207

4) La tavola della verità è:

Direttore	Delegato	Capo Cassiere	Ore 9-17	Porta aperta
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

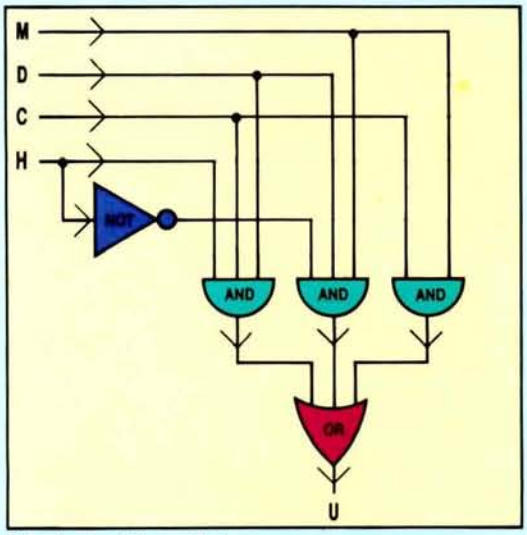
Dalla mappa di Karnaugh:



Otteniamo l'espressione: U = M.C + M.D.H + D.C.H.
Per verificare se a questo punto la situazione è corretta, poniamo le parole intere al posto delle sigle. L'espressione afferma che la porta può essere aperta da:

I) Il Direttore ed il Capo Cassiere in ogni momento
II) Il Direttore ed il Delegato fuori dall'orario
III) Il Delegato ed il Capo Cassiere tra le 9 e le 17

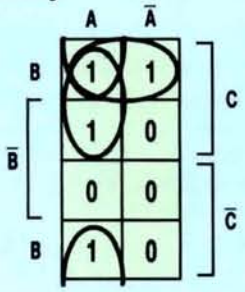
Il circuito corrispondente sarà:



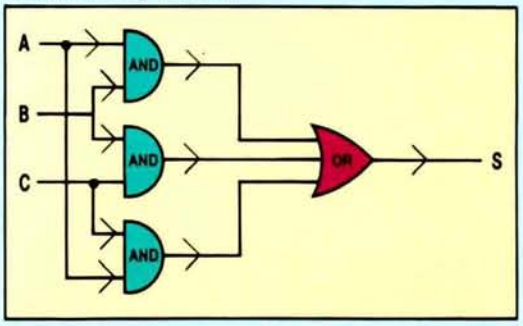
5) La tavola della verità è:

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Dalla mappa di Karnaugh:



Otteniamo l'espressione: S = A.B + A.C + B.C
Il circuito corrispondente è:

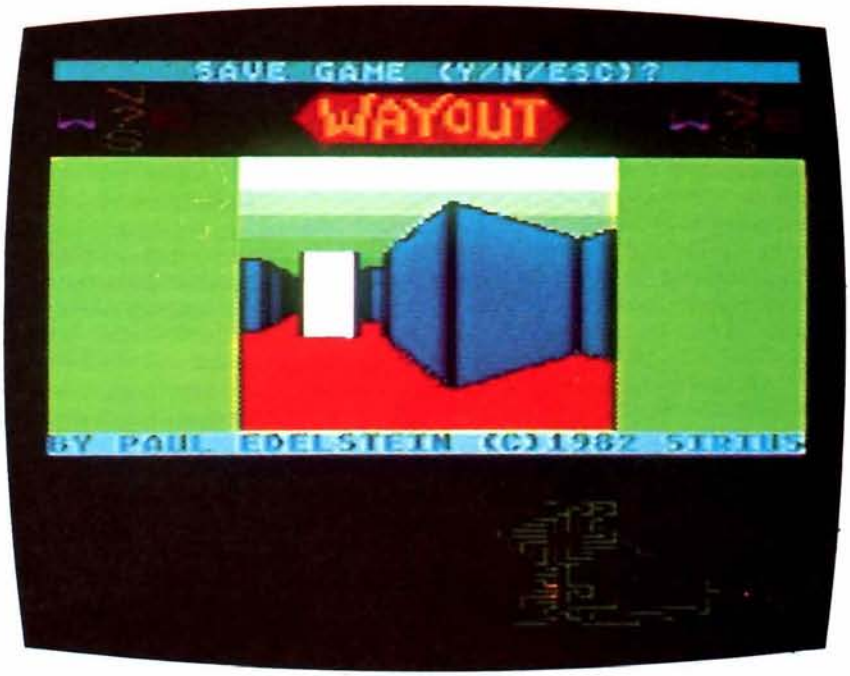


LIZ DIXON

Software in vetrina

Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Due giochi, uno per l'Atari ed uno per lo Spectrum, che fanno parte della folta schiera dei "labirinti". Come si può vedere anche da questi nuovi esempi, è questo un genere che offre una vasta scelta, anche per i più esigenti.



Way out per l'Atari

Anche chi odia, per principio, i labirinti, rimane affascinato dalle caratteristiche di questo avvincente gioco, scritto per l'Atari 800, dotato di unità a dischi.
In primo luogo, le scene del labirinto sono visualizzate con un effetto tridimensionale piuttosto efficace. Al giocatore viene offerta una scelta fra i 26 labirinti disponibili. La differenza tra l'uno e l'altro consiste, essenzialmente, nella lunghezza e nel percorso. L'avversario, in questo gioco, è un curioso personaggio "cleptomane" (in effetti si chiama Cleptangle), che può rubare la bussola, la mappa del labirinto o altri oggetti che il giocatore ha faticosamente conquistato. Per ogni labirinto, viene tenuto un punteggio "massimo", assieme alle iniziali del giocatore che lo ha conseguito.



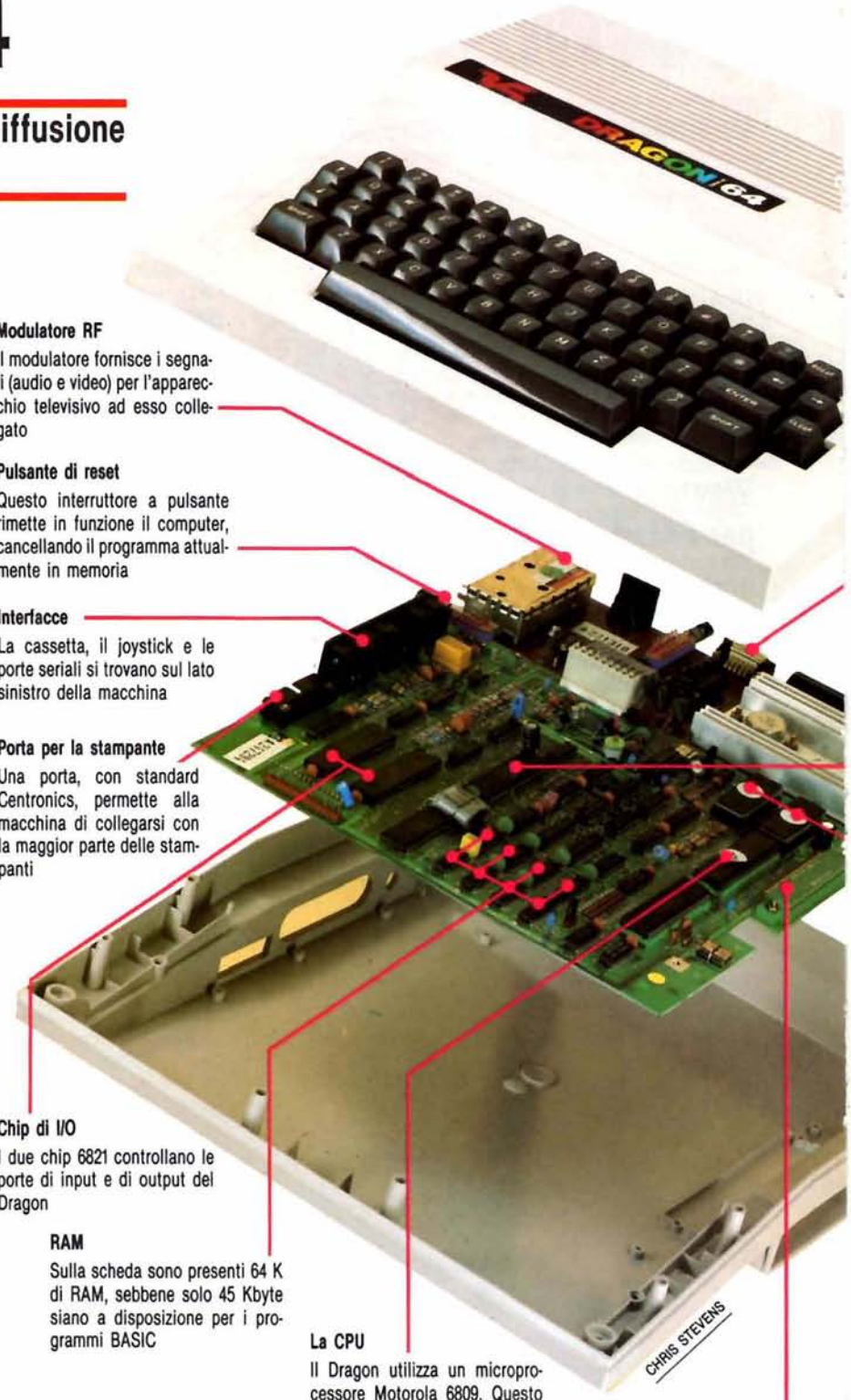
Manic Miner per lo Spectrum da 48 Kbyte

In questo gioco, l'abilità del giocatore viene messa alla prova dai tentativi di riuscire a far emergere, dalle profondità di una miniera, un personaggio, che ha scovato un'incredibile vena aurifera. Per raggiungere l'esterno (e quindi la ricchezza) occorre avanzare per 20 intricate gallerie, in ciascuna delle quali sono in agguato innumerevoli pericoli: frane improvvise, voragini da superare, ecc. Durante il percorso, inoltre, è necessario raccogliere cinque chiavi, senza le quali è impossibile accedere alle varie gallerie. L'animazione è molto buona, ed il gioco non stanca, anche se usato più volte.
È prodotto dalla "Bug-Byte", una software house specializzata in giochi per lo Spectrum.

Il Dragon 64

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

Il Dragon 64 ha una tastiera simile a quelle del precedente apparecchio 32, ma, benché di qualità superiore, essa è priva di alcuni tasti di funzione, tra i quali Escape, Tab e Control.
All'accensione, il Dragon 64 entra automaticamente nel "modo 32" e può quindi utilizzare tutto il software Dragon. Per il BASIC sono a disposizione fino a 30 Kbyte di memoria libera. Il comando EXEC 48000, comunque, deseleziona la ROM BASIC e mette a disposizione altri 32 Kbyte di memoria libera (il BASIC viene trasferito nella parte alta della memoria, lasciando così un totale di 45 Kbyte liberi). Tutti e 64 i Kbyte di memoria sono disponibili solo utilizzando altri sistemi operativi oppure programmi in codice macchina.
La scheda a circuiti stampati del 64 è molto diversa da quella del 32: il microprocessore è lo stesso, ma i chip sono stati spostati per dar spazio alla nuova porta seriale ed alla maggiore memoria. Pertanto, i possessori del Dragon 32 che volessero aumentare le capacità del proprio apparecchio, non hanno la possibilità di aggiungere semplicemente qualche pezzo, ma devono sostituire l'intera scheda. La CPU del 64 è un Motorola 6809, un prodotto che è comparso troppo tardi sul mercato per poter ottenere il successo del 6502 e dello Z80. Il chip video 6847, della medesima serie, è in grado di produrre immagini tanto su un apparecchio televisivo che su un monitor, dotando il Dragon di un modo di visualizzazione del tutto particolare.
Il Dragon, infatti, prevede uno schermo suddiviso in 32 × 26 caratteri, su sfondo sia verde che arancione. Un display così limitato può non creare problemi all'utente del Dragon 32, ma rappresenta una limitazione per chi intendesse impiegare il Dragon 64 per usi commerciali. Il nuovo software basato su dischetti, in particolare il sistema operativo professionale OS9, risente notevolmente delle difficoltà causate dalla ristretta area video.
Il Dragon possiede un certo numero di "modi" di alta risoluzione per la grafica, che vanno da una risoluzione di 128 × 96 pixel in quattro colori ad un massimo di 256 × 192 pixel, ma in due colori. Se confrontato con altre macchine, il Dragon presenta molti limiti, soprattutto perché la qualità dell'immagine può risultare molto bassa. Tuttavia, questo apparecchio è in grado di far girare programmi di gioco tipo "sala giochi", e, grazie ad un software appositamente concepito, può visualizzare scritte, in maiuscolo e minuscolo, su 24 righe per 52 colonne, un formato utile per una lavagna elettronica o per programmi di word processing.



Memoria ed espansioni

Il Dragon 64 utilizza un dialetto del Microsoft BASIC, in una buona versione estesa, da 16 Kbyte. Fra le sue caratteristiche ci sono una serie di potenti comandi per gestire il suono e la grafica. Il Dragon può immagazzinare in memoria, simultaneamente, un buon numero



Unità a dischetti Dragon
Può avere uno o due drive, ciascuno con 175 Kbyte di capacità. Il DOS, sebbene limitato, è adeguato al livello dell'apparecchio ed è incorporato nella cartuccia d'interfaccia



Presa per l'alimentazione

Controllo dell'immagine
Questo chip 6847 produce le immagini per il televisore ed il monitor del Dragon

Dissipatore
Il trasformatore per l'alimentazione del Dragon si trova in un contenitore separato, ma il sistema ha bisogno di un grosso dissipatore per eliminare il calore generato dai circuiti

Il BASIC
Queste due EPROM contengono il BASIC, una versione Microsoft da 16 Kbyte

Dragon 32
Il Dragon 32 viene ancora venduto con un certo successo, ma molti dei suoi possessori tentano di accrescerne le capacità e la flessibilità per portarle al livello del 64



IAN McKINNELL

di immagini grafiche, per poi spostarsi rapidamente da una all'altra, fornendo quindi un semplice strumento per creare animazioni grafiche in BASIC. Volendo, lo spazio di memoria solitamente riservato alla grafica può essere adoperato per programmi BASIC e dati.

L'apparecchio è corredato da una serie completa di interfacce, comprendenti due convertitori da analogico a digitale e una porta standard Centronics, che permette di collegarlo a quasi tutti i tipi di stampante parallela. La nuova porta seriale può essere collegata a stampanti (comprese quelle del tipo a margherita) o ad altri computer e apparecchiature. Potrebbe essere interessante sviluppare il sistema, collegandolo ad un terminale professionale, il che permetterebbe di utilizzare software molto più sofisticato.

Il Dragon può essere interfacciato con un normale registratore a cassette. Appositi comandi BASIC possono avviare o fermare il registratore dall'interno di un programma, oppure inviare suoni all'altoparlante dell'apparecchio televisivo usato come monitor. Inoltre, il BASIC è in grado di gestire archivi su cassette, grazie ad uno specifico gruppo di comandi. Abbiamo già parlato delle unità a dischetti del Dragon (vedere a pag. 584). La relativa cartuccia d'interfaccia contiene un sistema operativo di ridotte dimensioni, ma ben funzionante, ed una serie di estensioni al BASIC.

Il Dragon, considerate le attuali caratteristiche della visualizzazione, si dimostra un apparecchio assai interessante e di buone capacità più per attività di tipo hobbystico e didattico, che per applicazioni di genere commerciale.

OS9, un sistema operativo "multi-tasking" e multi-utente

Il microprocessore 6809 del Dragon 64, i 64 Kbyte di memoria e le unità a dischetti mettono questa macchina in grado di utilizzare il sistema operativo professionale OS9. Questo è forse il migliore sistema operativo per computer basati sul processore 6809. Esso fornisce alcune notevoli facilitazioni e la possibilità di utilizzare una vasta gamma di sofisticati programmi per attività commerciali. La Dragon già da tempo aveva intenzione di rendere disponibile l'OS9 per il 64. Sfortunatamente, l'alto costo dell'OS9 e del relativo software, anche se non superiore al prezzo di altri pacchetti di uso commerciale, ha limitato la popolarità di questo sistema tra gli utenti di home computer.

Il sistema operativo OS9 possiede alcune delle caratteristiche tipiche dell'UNIX, un sistema operativo per microcomputer in uso sui micro gestionali di maggiori dimensioni. Due sono le caratteristiche più significative di questo sistema: la possibilità di far girare più di un programma alla volta (multi-tasking) e quella di dar modo a più operatori di lavorare contemporaneamente con la macchina. Per conseguire questi risultati, l'OS9 organizza i file di programma ed i dati in una struttura formale e non secondo una comune Directory di file all'interno di ciascun dischetto. Ciascun file viene dotato di parole chiave e di codici d'accesso. Grazie a questo tipo di organizzazione, soltanto determinate persone possono accedere a particolari programmi e informazioni.

Molti utenti di home computer non sfruttano appieno le possibilità offerte dall'OS9 (del resto, ciò farebbe... "sbuffare" il Dragon, che è pur sempre un home computer!). Tuttavia, questo apparecchio permette di effettuare con poca spesa esperimenti con un sistema molto sofisticato ed offre la possibilità di stilare programmi in linguaggi alternativi, quali il "C" ed il PASCAL

DRAGON 64

DIMENSIONI

mm. 380 x 330 x 90

CPU

6809

MEMORIA

64 Kbyte di RAM (45 Kbyte per i programmi BASIC) 16 Kbyte di ROM

VIDEO

In funzionamento non grafico (text), 16 righe di 32 caratteri, soltanto in maiuscolo. Esistono dei caratteri grafici predefiniti. Otto colori disponibili. La risoluzione in funzionamento grafico va da 128 x 96 pixel (quattro colori) a 256 x 192 pixel (due colori)

INTERFACCE

Interfaccia per TV, per monitor a video composito, per registratore a cassette, per due joystick. Porta seriale e porta parallela per stampante. Connettore per cartucce ROM ed espansioni

UNITÀ A DISCHI

Sono collegabili fino a due drive da 175 Kbyte, sotto il controllo sia del sistema operativo Dragon che dell'OS9

LINGUAGGI DISPONIBILI

BASIC, FORTH, ASSEMBLY 6809. Con il sistema operativo OS9: PASCAL, BASIC strutturato, il linguaggio C ed altri ancora

TASTIERA

È di buona qualità, con 53 tasti

DOCUMENTAZIONE

I manuali del Dragon non sono particolarmente generosi in fatto di informazioni utili e, talvolta, alcune delle omissioni sono senz'altro fastidiose

PRO

Il Dragon 64 possiede elementi comuni a computer molto più sofisticati: una memoria ampia, da 64 Kbyte, ed una vasta gamma di interfacce. Il sistema operativo in dotazione è buono e la possibilità di usare l'OS9 è molto apprezzabile

CONTRO

Il Dragon 64 risente molto delle limitazioni imposte dalla ridotta quantità di caratteri visualizzabili sullo schermo, specialmente per chi intendesse usarlo per applicazioni commerciali. La velocità della CPU, che lavora ad una frequenza di 1 MHz, è appena sufficiente ad alcune delle funzioni offerte dal sistema operativo OS9

Le immagini viewdata

Esistono economici programmi per creare e visualizzare pagine simili a quelle dei sistemi viewdata

Oltre al Prestel, ci sono altri due sistemi pubblici di viewdata in Gran Bretagna: il Ceefax e l'Oracle. Poiché essi vengono irradiati via etere e non via cavo, in realtà costituiscono dei veri e propri sistemi "teletext". Molte grandi industrie utilizzano oggi dei propri sistemi viewdata, diffusi internamente all'azienda, tali da permettere ai dipendenti un costante aggiornamento su generiche informazioni sulla stessa azienda o su informazioni specializzate, quali, per esempio, i prezzi di mercato più recenti. Analogamente, i comuni di alcune città britanniche hanno istituito un sistema locale di viewdata, gratuito, che comprende un notiziario, informazioni sui servizi a disposizione dei cittadini ecc.

I sistemi viewdata vengono utilizzati sempre più frequentemente anche in settori commerciali di piccole dimensioni, ed anche in alcune catene di negozi. In tutti questi settori del commercio può essere adottato un sistema di informazioni pubblicitarie che si basa su un viewdata. È quindi possibile creare pagine d'informazioni viewdata e quindi utilizzarle sul proprio sistema privato oppure venderle a eventuali acquirenti. Sul sistema Prestel, ad esempio, si può inserire una pagina di pubblicità, ad un prezzo che oscilla sulle 70.000 lire. Tale costo può essere in seguito recuperato mediante una specie di "tassa", corrisposta dall'utente, ogni volta che egli accede alla pagina. Questa tecnica pubblicitaria è altamente remunerativa, sia grazie al meccanismo ora descritto, che per i risultati ottenuti dall'inserzione pubblicitaria.

Un intraprendente gruppo di studenti del Galles ha proposto ad un grande magazzino, nel periodo natalizio, l'istallazione di un piccolo sistema viewdata che presentava ai

Creazione di una pagina

Esaminiamo le varie fasi di creazione di una pagina viewdata. Poniamo che una agenzia di viaggi ci abbia commissionato una pagina di pubblicità per una vacanza in Grecia, che andrà a far parte di un sistema a carosello posto nella vetrina dell'agenzia.

La prima fase, prima ancora di lavorare sulla tastiera, consiste nel disegnare un progetto della pagina su carta quadrettata, di 40 x 24 quadrati e considerare ciascun quadratino come uno spazio di carattere sul video. Sul foglio disegneremo una tipica scena di vacanza (la spiaggia, il mare, il sole ecc...) e su di essa scriveremo le informazioni riguardanti il prezzo, la località e così via



Pagina 100

In questa immagine compaiono mare e spiaggia. Il mare è stato creato premendo, all'inizio di ciascuna linea, il tasto di funzione "blu" seguito dal tasto di funzione "sfondo". Si è proceduto allo stesso modo per la spiaggia



Pagina 101

Creiamo ora una piccola collina sovrapponendo i caratteri a mosaico verde a quelli del mare. Nello stesso modo abbiamo inserito la parte superiore di un ombrellone in color viola



Pagina 102

Il disegno sta prendendo forma: la collina è fatta, abbiamo aggiunto una striscia rossa all'ombrellone ed abbiamo iniziato a lavorare sul sole. Le lettere di grande formato sono composte da 6 caratteri "mosaici"



Pagina 103

L'immagine è quasi completa. Le scritte evidenziate, il sole e l'ombrellone sono già delineati e sul mare è stata disegnata una barca a vela. Per rendere più caratteristica la scena, abbiamo posto un tempio greco sulla sommità della collina



Pagina 104

La pagina è finita. Come ultima cosa abbiamo inserito il testo "Due settimane in Grecia" composto da caratteri a doppia altezza: il prezzo della vacanza appare in realtà lampeggiante. Le onde del mare sono state create usando alcuni segni "meno" (—) di color azzurro

IAN McKINNELL

clienti pagine pubblicitarie. I ragazzi avevano progettato le varie pagine sul proprio microcomputer, usando un semplice programma "editor" per viewdata e quindi hanno collocato il computer nella vetrina principale del grande magazzino: qui la macchina automaticamente mostrava a rotazione le varie pagine pubblicitarie. Facendosi pagare le pagine pubblicitarie e l'affitto del micro i giovani hanno raccolto una somma non indifferente.

Chiunque abbia un minimo d'immaginazione, potrebbe trovare utili (e remunerative) applicazioni di un simile sistema: basta avere un microcomputer, un po' di talento artistico e, naturalmente, il software adatto.

Un programma di questo genere si chiama "Viewtext" ed è stato concepito per il Micro BBC. Per chi sia particolarmente interessato ai viewdata, questo computer si dimostra superiore alla maggior parte degli altri micro, poiché possiede, definita internamente, una propria serie di caratteri viewdata. Sebbene tale programma sia uno dei più economici, esso possiede sia un "editor" per creare le pagine viewdata, che un programma di controllo della visualizzazione "a carosello", per un massimo di 22 pagine.

Il funzionamento dell'editor non differisce molto dai comuni sistemi di videoscrittura: il cursore viene posizionato mediante tasti di controllo, dopo di che è possibile inserire il testo. Per selezionare le diverse caratteristiche del viewdata, i colori per il testo e per lo sfondo, i caratteri lampeggianti e a doppia altezza, è sufficiente premere gli speciali tasti di funzione, posti nella parte alta della tastiera del BBC. Ad esempio, per ottenere la parola CIAO in caratteri gialli e lampeggianti, basterà premere i tasti f3 (per il colore giallo) e f8 (per il lampeggio) e quindi scrivere la parola.

A questo punto è necessario ricordare due particolari che riguardano il funzionamento di un editor viewdata. Innanzitutto, le caratteristiche (quali il lampeggio o il colore del testo) vanno perduti quando ci si sposta sulla linea successiva: il testo ritorna a piccole lettere bianche su sfondo nero. Inoltre ciascuna caratteristica selezionata occupa uno spazio di carattere invisibile sullo schermo. Quindi nell'esempio precedente ci dovrebbero essere due spazi bianchi (vuoti) prima della parola CIAO. Se cancelliamo questi spazi o scriviamo su di essi, le caratteristiche ad essi associate scompariranno dal video.

La grafica viewdata

Siamo a questo punto in grado di creare pagine, di solo testo, adatte a molte applicazioni viewdata. Tuttavia, per creare diagrammi o materiale pubblicitario è necessario utilizzare la grafica, oltre che i testi. La grafica viewdata è costituita da una serie di caratteri grafici, ciascuno dei quali formato da un massimo di 6 pixel disposti su una matrice di 2 colonne per 3 righe. Il metodo di costruzione delle immagini è molto simile a quello usato per creare un mosaico e per questa ragione i caratteri vengono chiamati "alfamosaici". La qualità dell'immagine ottenuta è piuttosto scadente, se confrontata con i risultati grafici della maggior parte degli home computer; ciononostante, con un po' di fantasia è possibile realizzare qualsiasi tipo d'immagine.



L'editor Viewtext

Il Viewtext comprende un editor che permette di disegnare le pagine teletext. In questo esempio, la pubblicità per una vacanza in Grecia è stata modificata sia nel testo che nelle immagini

I pacchetti viewdata utilizzano questo tipo di grafica perché ogni pagina può disporre di un solo Kbyte di memoria, mentre un'immagine grafica ad alta risoluzione su un home computer può occupare fino a 20 Kbyte. Considerando la velocità di trasmissione del Prestel, il tempo impiegato per ricevere un'intera pagina ad alta risoluzione sarebbe eccessivo.

L'uso dei caratteri alfamosaici, mediante l'editor Viewtext, non è particolarmente facile: questi speciali caratteri vengono selezionati premendo il tasto di funzione f9 e quindi premendo un tasto di una lettera minuscola o di un numero sulla tastiera. Ogni tasto produce un carattere grafico differente, ma, sfortunatamente, non c'è alcun rapporto diretto tra ciò che sta scritto sul tasto ed il carattere grafico che appare sullo schermo: occorre quindi consultare continuamente una tabella del manuale per trovare il tasto voluto. Questo modo di procedere è molto faticoso: ad esempio, la creazione delle pagine grafiche che compaiono in queste pagine richiede circa due ore di lavoro. Editor più sofisticati permettono di costruire i caratteri accendendo o spegnendo ciascuno dei 6 pixel che compongono la matrice del carattere: questo è un procedimento molto più rapido.

Limitatamente ai caratteri alfamosaici, esistono due particolari funzioni: la "Separate Graphics", nella quale i pixel che compongono un carattere risultano leggermente separati l'uno dall'altro, e la "Hold Graphics", nella quale gli spazi vuoti, che si creano quando viene selezionata una nuova caratteristica, vengono soppressi. Ad esempio, per eliminare gli spazi bianchi che comparirebbero tra un carattere rosso ed uno verde vicini tra loro, prima si dovrà battere il tasto di funzione "Hold Graphics" e quindi il tasto per il colore verde.

VIEWDATA AD ALTA RISOLUZIONE

Molti sistemi viewdata fanno uso di una grafica alfamosaica, che consente di risparmiare memoria, ma a danno della qualità delle immagini, più scadente. Un tipo di grafica alternativa per i viewdata, con la quale si ottengono immagini più realistiche, è chiamata "alfa-geometrica".

Oltre alle lettere, ai numeri ed ai caratteri a mosaico, la grafica alfa-geometrica consente di tracciare, con semplici comandi linee, cerchi e figure simili. Il procedimento è molto simile a quanto avviene con i comandi BASIC per la grafica ad alta risoluzione, disponibili sulla maggior parte dei micro.

Con la grafica ad alta risoluzione, lo schermo viene suddiviso in una matrice di 320 x 240 pixel, diversamente dalla griglia 40 x 24 adottata in quella a caratteri alfamosaici. L'alta risoluzione permette di creare figure particolareggiate, ma il tempo per trasmettere una pagina "via telefono" può superare i due minuti.

Un sistema commerciale oggi disponibile in Gran Bretagna è il terminale MUPID: un microcomputer specializzato, in grado di visualizzare sia grafica alfamosaica che alfageometrica. Le sue originali caratteristiche permettono di produrre immagini animate su una singola pagina, colori sfumati e ombreggiature.

Il sistema alfageometrico è ideale per creare cartine geografiche o figure particolareggiate







Abbiamo fin qui descritto le principali caratteristiche degli editor viewdata; alcune versioni di maggior costo permettono di velocizzare il processo di creazione delle pagine e sono molto utili per le applicazioni commerciali, ma non indispensabili per un utente dilettante. Per esempio, programmi più costosi permettono di muovere o di copiare parti di una pagina viewdata, consentendo di procedere molto più speditamente.

Dopo aver creato una pagina viewdata, è opportuno salvarla su disco o su nastro, assegnarle un numero di pagina ed inserirla nel nostro sistema viewdata. Il pacchetto Viewtext può contenere, simultaneamente, fino a 22 pagine all'interno della memoria di un BBC. Terminata la creazione di un numero sufficiente di pagine, si può avviare il programma di visualizzazione "a carosello", che presenta le pagine secondo un ciclo ripetitivo: il tempo di permanenza sul video di ciascuna immagine viene stabilito in fase di sviluppo. Sulla linea superiore di ogni pagina viewdata appaiono informazioni come: la data e l'ora, il numero di pagine inserite nel sistema, il nome del servizio viewdata, ecc... Il programma di visualizzazione chiede l'impostazione di queste informazioni, prima di avviare la presentazione. Programmi più sofisticati permettono lo scambio di pagine con altri sistemi, mediante un modem ed una linea telefonica. Essi, oltre a disporre del sistema a carosello, permettono di costruire un database simile al Prestel, con pagine di menu che consentono di passare liberamente da una pagina all'altra senza seguire uno schema prefissato. Simili programmi sono più costosi dei precedenti e fanno uso di dischetti per l'immagazzinamento delle pagine di informazioni.

Largo alla memoria

Per i sottoprogrammi in codice macchina occorre trovare un'apposita area di memoria RAM

In questa fase del corso, la conversione manuale di un programma in codice macchina è un compito abbastanza semplice e molto utile dal punto di vista didattico. Ma quando si sono ben capiti i meccanismi che stanno alla base del procedimento via via che i programmi in ASSEMBLY aumentano di dimensioni, è consigliabile acquistare un programma Assembler adatto alla macchina di cui si fa uso.

Nel breve programma in codice macchina presentato in precedenza (vedere a pag. 597), ci sono alcuni punti che meritano un poco d'attenzione. Nel programma viene usato uno dei "registri" della CPU, per manipolare la memoria. Inoltre, abbiamo deciso in quale parte della memoria immagazzinare il codice macchina ed abbiamo fatto in modo che il microprocessore lo eseguisse. Queste sono le fasi della programmazione in codice macchina che creano maggiori difficoltà ad un principiante: è quindi utile analizzarle con maggior attenzione. Iniziamo con il problema di dove immagazzinare il codice macchina.

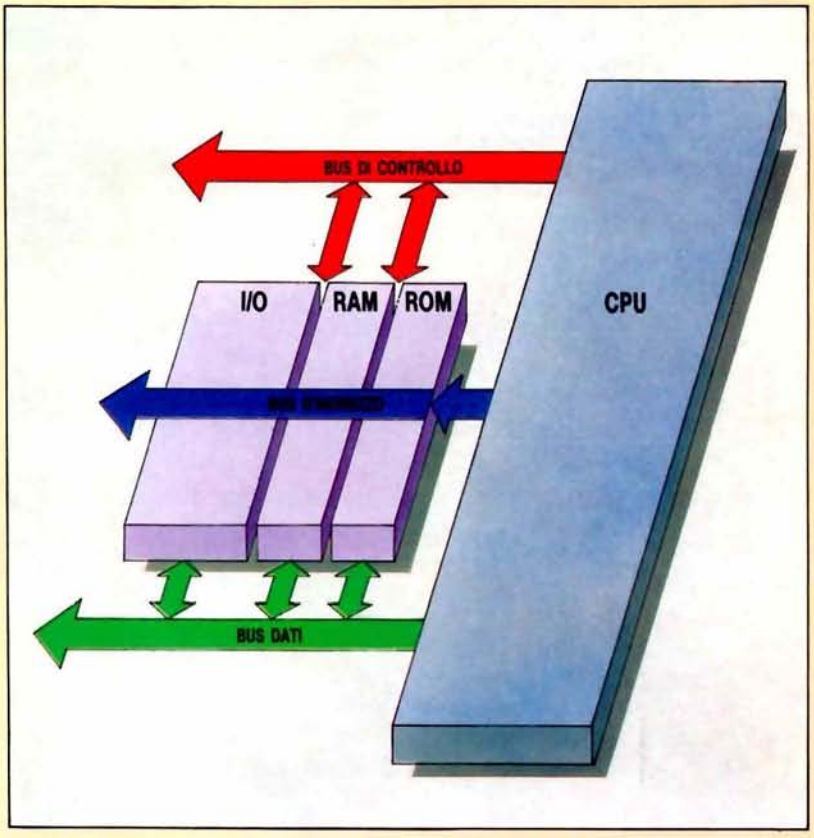
Per la CPU la sola differenza esistente tra due byte di memoria successivi risiede nel fatto che essi facciano parte della memoria di lettura/scrittura (RAM), oppure della memoria di sola lettura (ROM). I chip ROM contengono programmi di sistema ed informazioni, protetti contro sovrascritture accidentali (o volute): ecco perché tali dati possono essere solo letti. Teoricamente, a parte queste aree di memoria, niente impedisce di caricare un programma in un qualsiasi settore della memoria, ma vi sono alcune considerazioni pratiche di cui tener conto.

Sia la CPU che il sistema operativo usano alcuni settori della RAM per depositarvi temporaneamente i dati. Se carichiamo un programma in quelle aree, esso viene danneggiato da tali registrazioni oppure (evento più probabile) la CPU interpreterà il nostro codice macchina come uno dei suoi stessi dati.

Inoltre, i programmi BASIC possono occupare una grossa fetta della RAM, sia in forma di testi di programma, che come aree di deposito per le variabili. È quindi abbastanza pericoloso interferire con queste aree ed il programmatore si trova ad affrontare un dilemma apparentemente insolubile.

Una soluzione può essere quella di non usare del tutto il BASIC, lasciando quindi libera gran parte della RAM. Tuttavia, spesso conviene scrivere almeno una parte di programma in BASIC ed usare sottoprogrammi in codice macchina solo per quelle funzioni nelle quali il BASIC si dimostra troppo lento (ad esempio, nelle animazioni su video). Se il BASIC e i programmi in codice macchina devono coesistere nella RAM, occorre dunque riservare a questi ultimi una zona di memoria sufficiente: possiamo modificare l'occupazione dell'area riservata al testo dei programmi BASIC, oppure trovare settori della RAM non usati da alcune delle procedure di sistema.

Spostare i confini dell'area BASIC è molto



STRUTTURA DI UN PICCOLO SISTEMA

L'architettura di un tipico computer, in forma schematica, comprende la memoria ed una CPU. La prima è composta da chip ROM (che contengono programmi di sistema), da chip RAM e da particolari chip dedicati alle operazioni di input/output.

I segnali di controllo e dei dati fluiscono tra la CPU ed il sistema attraverso i "bus" di sistema. Questi sono dei gruppi di connessioni, in grado di trasportare uno o più byte di dati simultaneamente. I bus possono essere "unidirezionali", (come il bus degli indirizzi) o "bidirezionali" (come il bus dei dati). Il bus di controllo trasmette attraverso il sistema informazioni di commutazione che aprono e chiudono porte logiche per dirigere il flusso dei dati. Il bus degli indirizzi serve alla CPU per selezionare un particolare byte della memoria, consentendo quindi ai dati di fluire attraverso il bus dati ad 8 bit

KEVIN JONES

semplice sul Micro della BBC, perché gli indirizzi sono contenuti nelle variabili di sistema PAGE, TOP, LOMEN e HIMEN. PAGE, per esempio, punta all'inizio dell'Area di Testo dei programmi BASIC, che di solito corrisponde all'indirizzo $1200. Se inseriamo l'istruzione:

```
PAGE=PAGE+500
```

allora il sistema operativo immagazzinerà i programmi BASIC 500 byte "più in alto" nella memoria, lasciando quindi un'area di 500 byte libera per i nostri programmi in codice macchina. È possibile ottenere lo stesso risultato, su altre macchine, mediante istruzioni POKE (vedere le mappe di memoria a pag. 498). In alternativa, è possibile recuperare spazio spostando verso il basso il confine superiore dell'Area di Testo dei programmi BASIC. Sullo Spectrum, il comando CLEAR seguito da un indirizzo produce proprio questo risultato. Il solo limite imposto alla quantità di spazio sottratta al BASIC in questo modo è che si lasci memoria sufficiente per il programma BASIC. Piccole aree libere possono essere trovate nel sistema operativo RAM: un tipico esempio di ciò è il "buffer" usato per il trasferimento dati con il registratore a cassette del Commodore 64. Esso contiene 192 Kbyte di RAM (da $033C a $03FB), usati dal sistema operativo solo quando viene impiegata l'unità a cassetta. Spesso, questo spazio è sufficiente per le necessità del codice macchina.

All'interno dei programmi BASIC è possibile trovare altri blocchi di memoria liberi, benché di dimensioni minori: nelle linee REM, ad esempio. Nell'Area di Testo di programma BASIC questa linea:

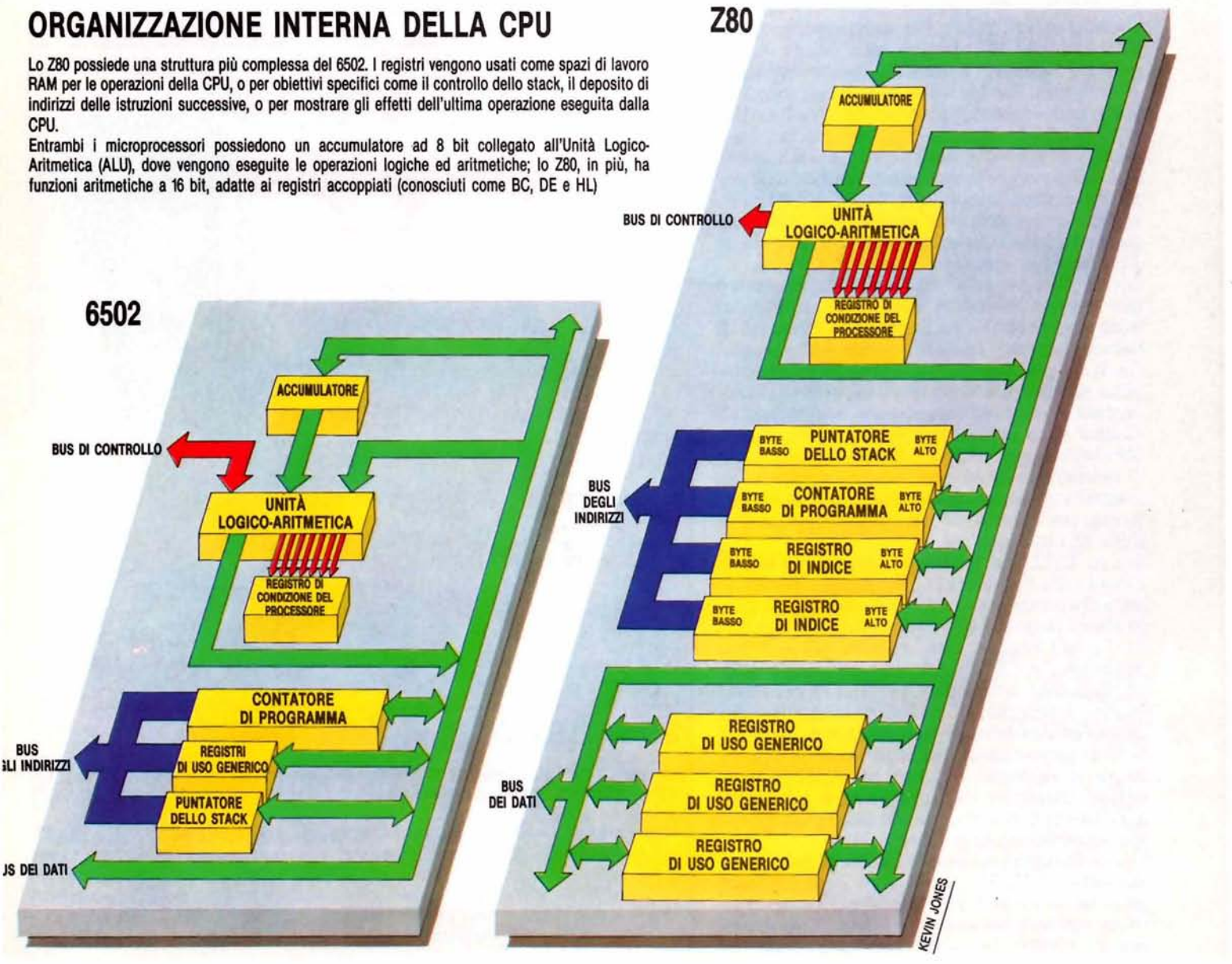
```
10 REM **********************************************
```

è composta da 25 byte consecutivi, tutti di valore $2A (il codice ASCII per "*"). Questi byte non vengono mai esaminati dal sistema

ORGANIZZAZIONE INTERNA DELLA CPU

Lo Z80 possiede una struttura più complessa del 6502. I registri vengono usati come spazi di lavoro RAM per le operazioni della CPU, o per obiettivi specifici come il controllo dello stack, il deposito di indirizzi delle istruzioni successive, o per mostrare gli effetti dell'ultima operazione eseguita dalla CPU.

Entrambi i microprocessori possiedono un accumulatore ad 8 bit collegato all'Unità Logico-Aritmetica (ALU), dove vengono eseguite le operazioni logiche ed aritmetiche; lo Z80, in più, ha funzioni aritmetiche a 16 bit, adatte ai registri accoppiati (conosciuti come BC, DE e HL)



operativo, né dall'interprete BASIC, poiché per essi il comando REM significa: "Ignora il resto di questa linea". Pertanto, è possibile caricare un sottoprogramma in codice macchina nei byte che contengono gli asterischi: in questa posizione esso non verrà mai "disturbato". Il principale vantaggio di questo metodo apparentemente disordinato (ma spesso usato nei programmi per lo ZX81 espanso), è che il codice macchina viene salvato e riletto assieme al programma principale in BASIC. Con altri metodi, invece, è necessario salvare il codice macchina separatamente dal programma BASIC. L'unico inconveniente è che, quando si fa un listato di questa linea, i byte in codice macchina vengono interpretati come caratteri ASCII e ciò può danneggiare l'immagine della linea sul video. Ecco perché sono state inserite delle "Avvertenze" nei programmi di dimostrazione a pag. 499 per il Micro della BBC e per lo Spectrum.

La versione Commodore 64 di quel programma carica il sottoprogramma in codice macchina nel buffer del registratore, mentre le versioni BBC e Spectrum lo caricano nelle rispettive linee di commento (REM) iniziali: da qui l'avvertimento di non listare queste versioni del programma.

Dopo aver scritto un programma in linguaggio ASSEMBLY, averlo assemblato in codice macchina e caricato nell'area RAM scelta, allora possiamo procedere alla sua esecuzione usando i comandi BASIC CALL (solo per il BBC), SYS (solo per il Commodore 64) o USR (per tutte e tre le macchine). Ciascuno di questi comandi è seguito dall'indirizzo del primo byte del programma in codice macchina, dovunque esso sia immagazzinato. Tutti e tre i comandi hanno lo stesso significato per l'interprete: "Esegui il programma in codice macchina partendo dall'indirizzo dato e ritorna ad eseguire l'istruzione BASIC successiva quando viene incontrato il codice operativo RET o RTS". La procedura è quindi simile a quella del comando GOSUB in BASIC.

Nella precedente lezione abbiamo presentato un programma per copiare il contenuto di un byte in un altro byte, usando l'accumulatore. Una tale procedura illustra l'importanza della CPU: i dati ed il controllo del sistema devono passare dalla memoria alla CPU e quindi tornare indietro alla memoria. Mentre in BASIC possiamo scrivere LET X = Y (che significa "copia il contenuto di Y direttamente in X), in linguaggio ASSEMBLY dobbiamo trasferire i dati dalla memoria alla CPU e quindi di nuovo dalla CPU alla memoria. I registri della CPU (vedere tabella) sono speciali byte di RAM, nei quali i dati provenienti dalla memoria vengono immagazzinati e manipolati. Sia lo Z80 che il 6502 posseggono un registro chiamato "accumulatore", cui si fa riferimento nella maggior parte delle istruzioni del linguaggio ASSEMBLY e nel quale vengono effettuate la maggior parte delle operazioni aritmetiche e logiche.

Supponiamo di voler sommare i due numeri $42 e $07 (il simbolo $, come si ricorderà, significa che il numero è esadecimale). Questa operazione viene eseguita semplicemente depositando uno dei due numeri nell'accumulatore ed aggiungendovi l'altro: la loro somma si "accumula" nello stesso registro. Tradotto in istruzioni ASSEMBLY:

Z80	6502
LD A,$42	LDA #$42
ADC A,$07	ADC #$07

Qui le istruzioni Z80 si riferiscono entrambe ai numeri che devono essere caricati e sommati, mentre nella versione 6502 i numeri sono preceduti dal simbolo #, che li identifica come numeri e non come indirizzi. Quindi: LDA #$65 significa "carica il numero $65 nell'accumulatore", mentre LDA $65 significa "carica nell'accumulatore il contenuto del byte che ha come indirizzo $65". Similmente l'istruzione "somma" ADC (valida tanto per lo Z80 che per il 6502) significa in questo caso "somma il numero seguente a quello contenuto nell'accumulatore". I numeri $42 e $07 sono chiamati "dati immediati" ed LDA #$42 può essere letto come "carica nell'accumulatore il dato immediato #$42".

Eseguite queste due istruzioni, l'accumulatore contiene la somma dei due numeri, ma in forma non "invisibile", almeno direttamente. Perciò, occorre trasferire il contenuto dell'accumulatore in un byte RAM, dove potrà essere comodamente esaminato. Il programma deve concludersi con un'istruzione RETURN ed il programma completo è:

Z80	
Codice Macchina	**Linguaggio Assembler**
A7	AND A
3E 42	LD A,$42
CE 07	ADC A,$07
32 ?? ??	LD (BYTE1),A
C9	RET
6502	
18	CLC
A9 42	LDA #$42
69 07	ADC #$07
8D ?? ??	STA BYTE1
60	RTS

Non ci occuperemo per ora del significato delle prime istruzioni in entrambi i programmi, ma è importante notare che la quarta istruzione contiene il simbolo BYTE1 e non un indirizzo. Il valore di BYTE1 varia da macchina a macchina: per questa ragione qui usiamo semplicemente un simbolo, che viene in seguito sostituito con un valore esadecimale.

Scegliamo adesso un'area in cui immagazzinare il codice macchina, assegnando quindi a BYTE1 l'indirizzo del primo byte successivo alla fine del programma. Tale indirizzo va inserito in forma "alto-basso" all'interno del codice macchina. Infine usiamo il programma Monitor a pag. 598 per caricare ed eseguire il codice macchina e per esaminare il byte nel quale dovrebbe essere immagazzinato il risultato $49.

Accumulatore

È il registro più importante della CPU. Le operazioni aritmetiche, logiche e quelle di trasferimento dei dati avvengono usando questo registro

Unità logico-aritmetica

È costituita da un sommatore binario e da porte logiche, che permettono l'accesso ai bit individuali del registro ed al bus dei dati. Controllata in modo appropriato, questa unità permette l'esecuzione di addizioni, di sottrazioni e di operazioni Booleane

Registro di stato del processore

Ogni volta che la CPU esegue un'operazione, ciascun bit di questo registro riflette alcuni aspetti di essa. Ad esempio, se il risultato di essa è 0, oppure se c'è un riporto a seguito di una somma

Contatore di programma

Esso "punta" all'indirizzo di memoria nel quale è immagazzinato il successivo op-code, da far eseguire alla CPU. La funzione BASIC USR (indirizzo) fa sì che l'indirizzo specificato venga caricato direttamente nel contatore di programma, provocando l'esecuzione del codice esistente a tale indirizzo

Puntatore dello stack

Questo punta all'indirizzo del successivo byte di RAM a disposizione per le funzioni interne della CPU. Ogni volta che la CPU deposita un dato o un indirizzo nello stack, il puntatore dello stack viene aggiornato di conseguenza

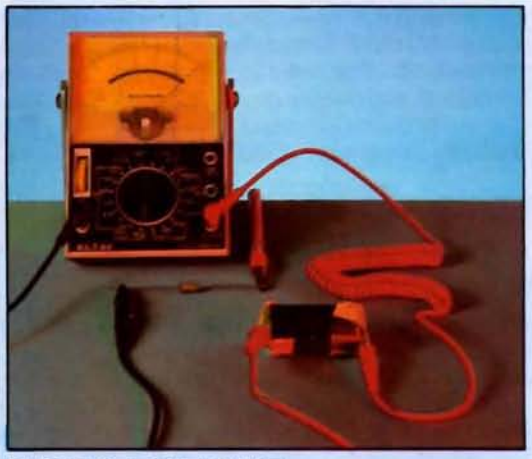
Registri di Indice e di Uso Generico

I registri di indice vengono usati dalla CPU per indirizzare più efficientemente la memoria, mentre i Registri per Uso Generico (General Purpose Registers) sono usati dalla CPU per attività generali di calcolo e di confronto

L'interno dei componenti

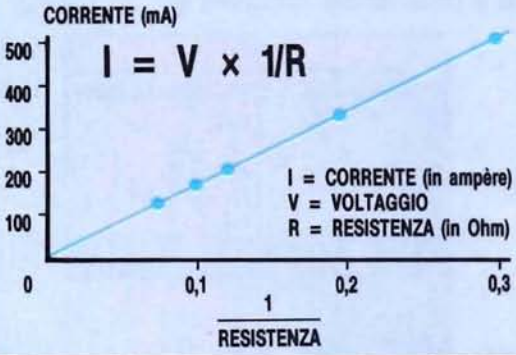
Resistenze, condensatori, diodi e transistor: le parti elementari di ogni circuito elettronico

Anche i più complessi circuiti integrati si basano e vengono in larga parte fabbricati con componenti elettrici assai semplici, che si possono distinguere in due grandi famiglie: i componenti "attivi" e quelli "passivi".
Questi ultimi sono estremamente semplici: il loro compito è unicamente quello di bloccare o attenuare i segnali elettrici che li attraversano. Le resistenze ed i condensatori sono due esempi di componenti passivi.
I componenti attivi sono, nel complesso, molto più ingegnosi. Possono, ad esempio, amplificare il segnale che li attraversa. Un transistor ne è un buon esempio: quando riceve un piccolo segnale, lo amplifica trasformandolo in un segnale di ampiezza maggiore.
I componenti passivi vengono fabbricati con materiali abbastanza comuni. A parte i materiali impiegati per gli involucri (varie resine e plastiche), che avvolgono, ad esempio, una resistenza, il componente stesso viene fabbricato con rame, carbone e acciaio. Tutti questi materiali sono conduttori. I componenti attivi, al contrario, sono tutti fabbricati con silicio o germanio. Questi due elementi presentano la particolare proprietà di non comportarsi né come conduttori, come l'acciaio ed il rame, né come resine e plastiche isolanti, ma come una via di mezzo tra queste due cose. In particolari condizioni questi materiali sono in grado di condurre elettricità, mentre in altre circostanze non sono in grado di farlo: per questa ragione sono stati chiamati "semiconduttori".
I componenti attivi basati su materiali semiconduttori svolgono le loro funzioni in modo assai interessante. Ad esempio, non si può affermare con certezza che, applicando un certo voltaggio ad un semiconduttore, si avrà come risultato il passaggio attraverso di esso di una corrente elettrica di un dato valore, né, in effetti, che si produrrà un qualsiasi passaggio di corrente. Tutti i conduttori, tuttavia, si basano su un principio molto semplice. La Legge di Ohm (vedere a pag. 694) descrive come un qualsiasi conduttore reagirà al segnale elettrico che lo attraversa. Il transistor è il più importante di questi componenti per l'elaborazione: grazie ad esso i computer possono memorizzare ed elaborare le informazioni. In un prossimo articolo useremo transistor per la costruzione di alcune semplici porte logiche di cui abbiamo già parlato nella sezione "Le chiavi dell'Informatica".



DIMOSTRAZIONE DELLA LEGGE DI OHM

Si può verificare la Legge di Ohm con l'aiuto di qualche resistenza, una pila, alcuni fili elettrici ed un multimetro. Le resistenze dovrebbero avere una potenza di un watt o più, con valori compresi tra 3,3 e 15 Ohm. Predisporre il multimetro per la lettura della corrente. Un fondo scala di 400 mA (milliampère) è il più adatto per la nostra dimostrazione. Collegare ora l'apparecchio agli altri componenti del circuito come mostrato nella figura. Inserire via via tutte le resistenze e controllare la lettura sul multimetro. Se, dopo aver moltiplicato i vari valori di corrente letti sul multimetro per uno diviso il valore dei singoli resistori, tracciamo un diagramma dei risultati ottenuti, otterremo una linea retta dove I è direttamente proporzionale a 1/R a voltaggio costante. Quindi I = V x 1/R, cioè V = IR che è la formula della Legge di Ohm



LIZ HEANEY

I DIODI



I diodi sono gli equivalenti elettrici delle valvole più elementari. Un diodo offre una resistenza molto bassa alla corrente elettrica in una direzione (solitamente di pochi Ohm) mentre offre una resistenza molto alta nella direzione opposta (anche molti milioni di Ohm). Questa discordanza con la legge di Ohm si verifica perché il diodo non è un normale conduttore, bensì un semiconduttore, fabbricato con materiali come il silicio ed il germanio. In alcuni diodi, incapsulati in sostanze trasparenti, si riesce effettivamente a vedere il semiconduttore. La polarità di un diodo è individuata, in genere, da una striscetta rossa ad una delle estremità

SPECTRUM

```
20 DIM b$(3,7)
25 DIM c$(4,7)
30 INPUT "Colori delle bande"
40 INPUT "banda n. 1",b$(1)
50 IF b$(1) = "oro" OR b$ =
   (1)"argento"  THEN GO TO 30
60 INPUT "banda n. 2",b$(2)
70 INPUT "banda n. 3",b$(3)
80 PRINT
90 FOR i=1 TO 3
100 RESTORE 180
110 FOR j=0 TO 9
120 READ c$(1),c$(2),c$(3),c$(4)
130 IF c$(1)=b$(i) THEN GO SUB 1000
140 NEXT j
150 NEXT i
160 PRINT "  ohm"
170 STOP
180 DATA "nero","0","0",".0","marrone",
    "1","1","0","rosso","2","2",".00",
    "arancio","3","3","000"
190 DATA "giallo","4","4","0000",
    "verde","5","5",".00000","blu","6",
    "6",".000000"
200 DATA "viola","7","7",".0000000",
    "grigio","8","8",".0","bianco","9",
    "9",".0"
1000 FOR k=1 TO 7
1020 IF c$((i+1),k)<>" " THEN PRINT
     c$((i+1),k);
1025 NEXT k
1030 RETURN
```

BBC

```
10 REM conversione codice resistenze
20 DIM B$(3),C$(3)
30 PRINT "Colori delle bande:"
40 INPUT "BANDA NUMERO 1",B$(1)
50 IF B$(1)="ORO" OR B$(1)="ARGENTO"
   THEN GOTO 30
60 INPUT "BANDA NUMERO 2",B$(2)
70 INPUT "BANDA NUMERO 3",B$(3)
80 PRINT
90 FOR I=1 TO 3
100   RESTORE
110   FOR J=0 TO 9
120     READ C$(0),C$(1),C$(2),C$(3)
130     IF C$(0)=B$(I) THEN PRINT C$(I);
140   NEXT J
150 NEXT I
160 PRINT "  ohm"
170 END
180 DATA NERO,0,0,,MARRONE,1,1,0, ROSSO,
    2,2,00, ARANCIO,3,3,000
190 DATA GIALLO,4,4,0000,VERDE,5,5,
    00000,BLU,6,6,000000
200 DATA VIOLA,7,7,0000000, GRIGIO,8,
    8,, BIANCO ,9,9,,
```

Per il Commodore 64, sostituire le linee 40, 60 ed 80 con:

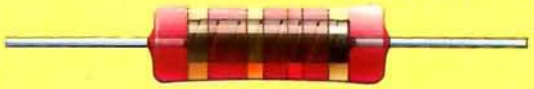
```
40 INPUT "BANDA NUMERO 1";B$(1)
60 INPUT "BANDA NUMERO 2";B$(2)
70 INPUT "BANDA NUMERO 3";B$(3)
```

LE RESISTENZE

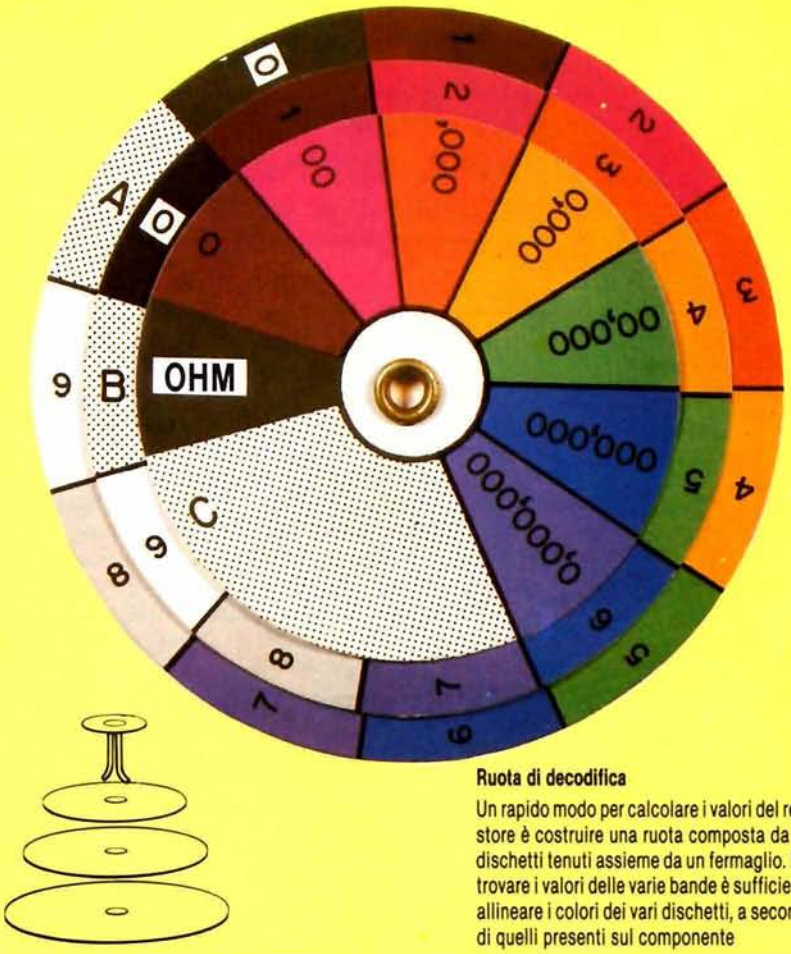
RESISTENZA A FILO



RESISTENZA A FILM DI CARBONE



Le resistenze sono forse i componenti elettronici più semplici: i due tipi più comuni sono "a filo" ed "a film" di carbone. Le resistenze a filo sono composte da un lungo filo metallico, strettamente avvolto ad un cilindro di materiale isolante e ricoperto da un rivestimento anch'esso isolante. La corrente deve fluire lungo tutto il filo e ciò produce una sensibile diminuzione di corrente. Le resistenze a film di carbone funzionano sullo stesso principio, ma il percorso che la corrente deve seguire è costituito da una spirale incisa su un film di carbone stratificato su un cilindro di materiale isolante. Entrambi i tipi di resistenze vengono contrassegnati con bande colorate, che ne indicano il valore, e la lettura di queste bande è abbastanza semplice. Le prime due bande indicano un valore che va moltiplicato per un multiplo di dieci (indicato dalla terza banda), per ottenere il valore della resistenza in Ohm. La banda finale (oro oppure argento) indica la tolleranza del componente e viene anche usata per stabilire in quale direzione devono essere lette le altre bande. Le bande dovranno sempre essere lette tenendo la banda oro (o argento) alla propria destra. Per non dover ricordare a memoria il valore dei vari colori, si può usare un breve programma, come quello listato qui a fianco



Ruota di decodifica

Un rapido modo per calcolare i valori del resistore è costruire una ruota composta da tre dischetti tenuti assieme da un fermaglio. Per trovare i valori delle varie bande è sufficiente allineare i colori dei vari dischetti, a seconda di quelli presenti sul componente

I CONDENSATORI



I condensatori oppongono resistenza alla corrente alternata. Le correnti alternate ad alta frequenza scorrono attraverso i condensatori più facilmente di quelle a frequenza più bassa: questi componenti sono quindi assai utili per "ripulire" o filtrare un segnale elettrico e molte schede per microprocessori sono costellate da condensatori, che svolgono appunto questa funzione. I condensatori sono costituiti da due lamine metalliche, separate da un sottile strato isolante, composto da vari sali speciali (per gli alti voltaggi) o da ceramica (per bassi voltaggi).
Poiché la capacità aumenta proporzionalmente alla superficie delle lamine, queste possono avere anche notevoli dimensioni e, per ovviare a questo problema, esse vengono letteralmente "arrotolate" su se stesse





I TRANSISTOR

Il transistor è il componente più complesso tra quelli qui illustrati: è un semiconduttore e la sua invenzione sta alla base della moderna elettronica. Il transistor svolge due principali funzioni: come amplificatore può modulare, con una piccola corrente di controllo, una corrente molto maggiore in uscita. Come interruttore, invece, può usare un flusso di corrente per interrompere o meno un secondo flusso. Questo secondo tipo di funzionamento, come interruttore elettronico, sta alla base di tutti i circuiti elettronici digitali ed è fondamentale per il funzionamento dei computer.
Come i diodi, i transistor vengono fabbricati con materiale semiconduttore ma presentano al loro interno due giunzioni, anziché una sola. Dall'involucro fuoriescono tre piedini, chiamati base, collettore ed emettitore. La funzione del transistor dipende dal modo col quale vengono collegati, nel circuito, questi piedini. A seconda del tipo di collegamento, infatti, il transistor lavorerà come amplificatore o come interruttore. Diversamente dalle resistenze e dai diodi, non ci sono metodi standard per identificare i piedini dei transistor. Di solito, occorre riferirsi alle indicazioni fornite dal fabbricante per ciascun tipo di transistor, identificato da una particolare sigla

KEVIN JONES

La Oric

Nata come concorrente della Sinclair, questa giovane ditta ha incontrato non pochi problemi



Nel 1982, nella Oric confluì la notevole esperienza di quattro persone: quella di Barry Muncaster e di Peter Harding, rispettivamente amministratore e direttore alle vendite, quella del Dott. Paul Johnson, esperto progettista hardware e quella, infine, di Paul Kaufmann, esperto softwarista. L'Oric-1, il computer prodotto da queste quattro menti, è simile allo Spectrum della Sinclair, ma è basato sul chip 6502, e viene venduto in 2 versioni: con 16 Kbyte o con 48 Kbyte. Rispetto allo Spectrum, è dotato di una tastiera migliore, di un contenitore più elegante e di superiori capacità grafiche e sonore. Il BASIC è un dialetto Microsoft standard, simile a quello adottato dal Vic-20 e dall'Apple. L'Oric-1 possiede anche un'interfaccia standard per stampante Centronics, che permette l'uso di normali stampanti ad 80 o 132 colonne.
Nonostante la validità del prodotto, la Oric ha dovuto affrontare non poche difficoltà iniziali, tra le quali ritardi nella distribuzione delle macchine ed errori di progettazione. Le prime macchine avevano problemi con il caricamento di programmi da registratore e le immagini video erano piuttosto instabili. Le ROM del Basic contenevano un certo numero di piccoli, ma fastidiosi difetti, che ostacolavano la programmazione. Inoltre, le tecniche adottate per la grafica, tutte tese al risparmio di memoria, rendevano difficoltosa la programmazione della macchina. I problemi della Oric crebbero quando la Casa fu costretta ad aumentare i prezzi, portandoli al di sopra di quelli dello Spectrum.



Vecchi e nuovi Oric

L'Oric-1, dall'aspetto molto futuristico, ha resistito 15 mesi prima che la Oric decidesse di dotarlo di un nuovo involucro rosso e nero e di ripresentarlo col nome di Atmos: esso ha ora una tastiera standard ed una ROM BASIC modificata. La stampante/plotter a 4 colori è stata anch'essa riprogettata per adattarla al nuovo design

Il risultato finale di questa situazione fu che la macchina incontrò notevoli problemi di vendita e ci furono difficoltà a trovare software da utilizzare su di essa. La Tansoft lavorò duramente per rendere più competitivo questo computer, fornendolo di linguaggi come l'ASSEMBLY ed il FORTH; finalmente oggi l'Oric possiede una rispettabile biblioteca di software.
La Oric ha preannunciato una vasta gamma di accessori, tra cui: unità a dischetti, modem e stampanti. Di questi a tutt'oggi è apparsa sul mercato solo la stampante/plotter a 4 colori. Dopo questo incerto avvio commerciale, la Oric si è lentamente riorganizzata. Le vendite all'estero hanno avuto una crescita impressionante. Delle 170.000 macchine prodotte nel 1983, oltre il 50% è stato esportato. L'Oric ha incontrato un particolare successo in Francia dove è stato proclamato il miglior micro del 1983, perché il suo output monitor RGB, diversamente da quello degli altri micro inglesi, è compatibile con gli apparecchi televisivi francesi. Per il mercato giapponese è stato progettato un modello con particolari modifiche che ha avuto un ottimo successo. Cinque mesi dopo il lancio sul mercato della sua macchina, la Oric ha cercato di rimediare ai molti problemi presenti sull'Oric-1 in un modo assai originale. Ha prodotto il modello Oric Atmos che presenta una tastiera di tipo standard e miglioramenti della ROM BASIC, ma questo apparecchio non è altro che la stessa macchina, dotata di una nuova custodia. Dopo l'annuncio che è previsto l'impiego di un'unità a dischetti a tre pollici della Hitachi, a completamento del sistema, l'Atmos Oric comincia ad essere considerato come un micro a dischetti di discreta qualità, più vicino al livello dell'Acorn Electron e del Commodore 64 che dello Spectrum.

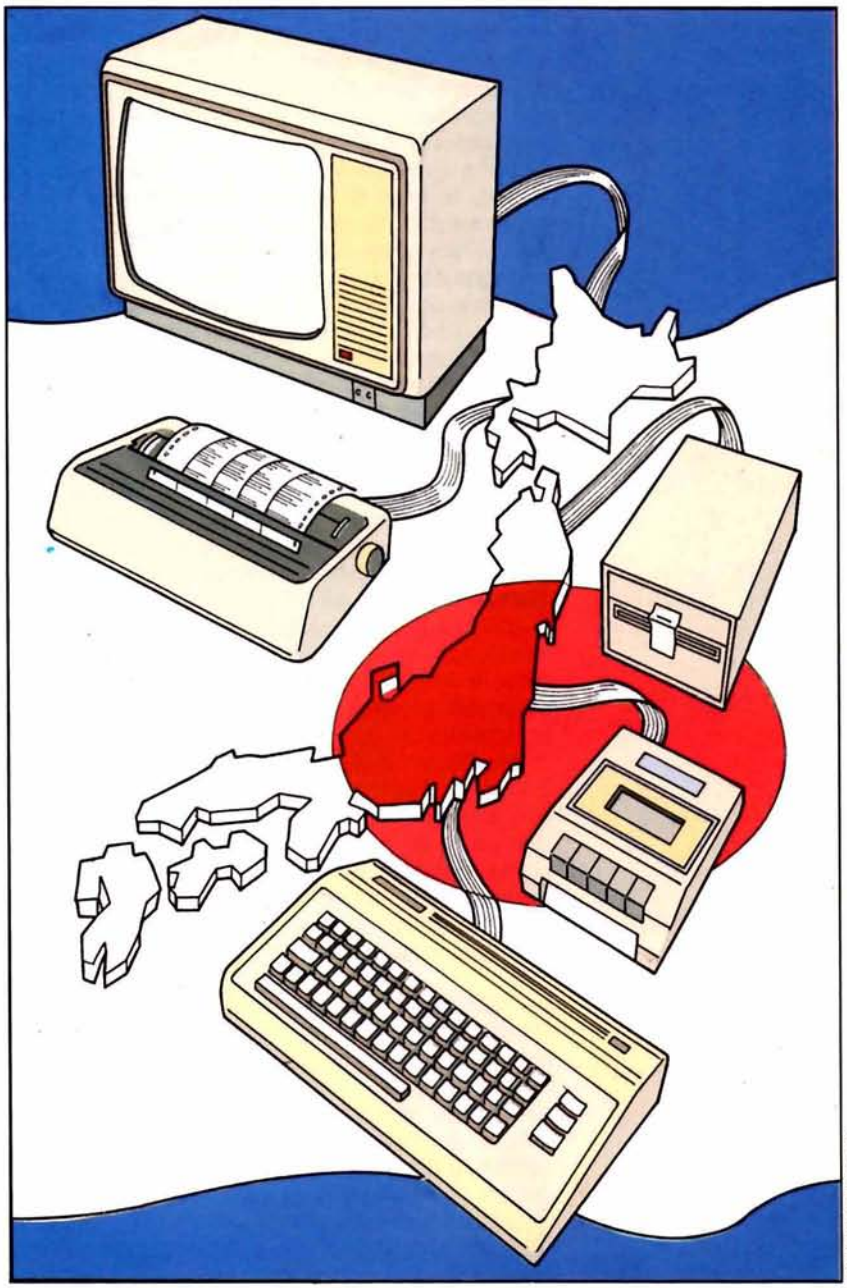
Lo standard MSX

Un tentativo di abbattere le barriere dovute all'incompatibilità tra i diversi microcomputer

Per comprendere perché vi siano differenze così marcate, tra i vari tipi di computer, è necessario analizzare lo sviluppo, quanto mai anomalo, dell'industria dei microcomputer. I primi microprocessori che abbiano lasciato un'impronta significativa nel settore dei microcomputer sono stati il chip 8080 della Intel ed il 6800 della Motorola. Gran parte degli attuali problemi di incompatibilità, stranamente, deriva dalla notevole versatilità di questi primi microprocessori. Ad esempio, il grande numero di porte d'ingresso/uscita, disponibile su ambedue i microprocessori, ha concesso ai fabbricanti la massima libertà nella scelta di quelle da usare per il collegamento con i dispositivi del computer (circuiti per la gestione del video, ad esempio). Di tale libertà si è abusato ampiamente; basti pensare che i primi home computer sono stati fabbricati nei luoghi più impensati, come vecchie autorimesse e laboratori di dilettanti. Non è stato possibile, dunque, standardizzare gli indirizzi di memoria delle porte di ingresso o di uscita: il costruttore A poteva assegnare l'indirizzo 255 alla porta parallela per la stampante, mentre il costruttore B poteva dare a quella stessa porta l'indirizzo di memoria 254.

La situazione, che era ed è tutt'ora abbastanza caotica, si è aggravata con l'introduzione di più sofisticati circuiti atti a migliorare le capacità grafiche o ad aggiungere speciali effetti sonori. Tutti i programmi basati sull'impiego di questi nuovi circuiti, su di un particolare microcomputer, raramente possono essere utilizzati su altri apparecchi, senza che vengano apportate modifiche sostanziali. Se, fin dal principio, ci fosse stata un'industria di grandi dimensioni capace di imporre criteri di progettazione standard, oggi la situazione sarebbe ben diversa. Viceversa, il "fenomeno microcomputer" è dovuto al sorgere di numerosi piccoli costruttori, ciascuno dei quali ha adottato un proprio stile di progettazione e propri criteri di fabbricazione. I computer, oltre a presentare notevoli differenze di carattere fisico ed elettronico, si sono diversificati anche nei linguaggi di programmazione adottati spesso in versioni incompatibili con altre macchine. Per il BASIC, il linguaggio più diffuso sui piccoli computer, non è stato fissato alcun criterio di standardizzazione, tanto più che, fino al 1970, negli ambienti specializzati veniva considerato un linguaggio di secondaria importanza, adatto solo ai principianti. A partire dalla fine degli anni '70, però, i computer si sono sviluppati ad una velocità spaventosa: progetti pionieristici (come quello della Apple, per esempio) cominciarono ad introdurre prestazioni di alto livello, particolarmente la gestione della grafica a colori. Per rendere queste innovazioni accessibili agli utenti, è stato necessario aggiungere al BASIC nuovi comandi ed istruzioni, provocando una incontrollata proliferazione di nuovi "dialetti" del BASIC, l'uno diverso dall'altro.

Questi sviluppi, più che accrescere le possibi-



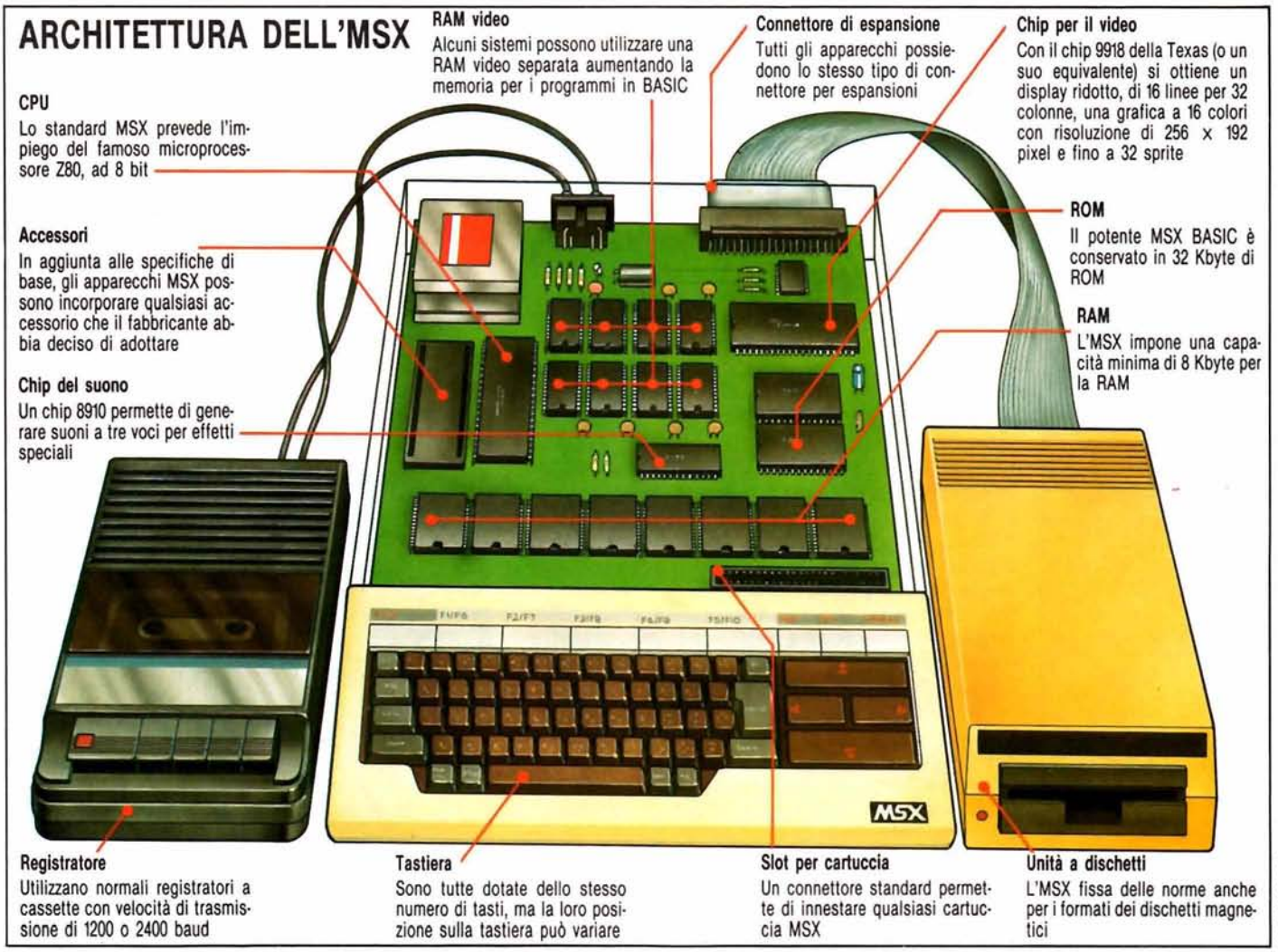
TONY DUNCAN-SMITH

La strada del successo
Lo standard MSX è la via imboccata dai giapponesi per conquistare il mercato mondiale degli home computer. Se avrà successo, il Giappone dominerà in questo settore, rinnovando quanto accaduto in passato nel campo dell'hi-fi e delle macchine fotografiche

lità di scelta da parte degli acquirenti, hanno indotto un notevole senso di frustrazione non solo negli utenti e nei fabbricanti di questi apparecchi, ma anche nei progettisti di software. Il possessore di un SORD incontrerà enormi difficoltà nel modificare uno qualsiasi tra i tanti giochi che oggi vanno per la maggiore, se, ad esempio, il programma è stato scritto originariamente per lo Spectrum.
Un fabbricante di computer, che si appresti a lanciare un nuovo apparecchio, sa benissimo che non sarà disponibile una vasta scelta di software per il nuovo prodotto, finché esso non avrà conquistato una buona posizione sul mercato. Quando il software per una particolare macchina scarseggia, anche le vendite subiscono una battuta d'arresto.
Infine, le case di software sanno che i loro programmi verranno acquistati esclusivamente dai possessori dei computer per i quali i programmi sono stati scritti. Supponiamo che una software house abbia progettato un nuovo gioco e che si appresti a commercializzarlo. Una apposita ricerca di mercato dimostra che il potenziale di vendita di un simile prodotto risulta di notevoli proporzioni, purché il prezzo non superi, ad esempio, le 15.000 lire e se ne vendano almeno 65.000 copie. Ma raramente un programma adatto ad un solo tipo di computer può essere assorbito dal mercato in simili quantità: occorre quindi che la casa di software sviluppi una seconda versione, adatta ad un altro apparecchio. Ciò comporta però un maggior investimento e, di conseguenza, un aumento del prezzo di vendita, di nuovo a scapito del successo commerciale del prodotto. È questo il dilemma che attanaglia molti produttori di software!
Il problema della compatibilità del software non è stato ignorato dall'industria dei computer: nei paesi occidentali si guarda ancora con un certo sospetto alla standardizzazione; i progettisti di computer dell'Estremo Oriente invece preferiscono adottare procedure sistematiche e standardizzate in quanto, a loro modo di vedere, ciò garantisce maggiori profitti.
La ASCII/Microsoft, una ditta nata dagli sforzi congiunti della Microsoft Corporation e della ASCII (una casa editrice giapponese di grande successo), sta tentando di portare un po' di ordine in tutto questo caos. La ASCII, da tempo, sviluppa e pubblica anche software commerciale e quando la Microsoft decise di conquistarsi uno spazio nell'impenetrabile mercato giapponese fu giudicata il partner ideale per una collaborazione: essa possedeva le necessarie capacità tecniche, mentre la ASCII garantiva una profonda conoscenza del mercato giapponese.
La Microsoft Corporation si è fatta un nome creando una versione di BASIC, chiamata MBASIC, divenuto, per certi versi, uno standard "de facto" ed adottato in tutto il mondo dai costruttori di computer. Nonostante quest'enorme diffusione, non è ancora possibile garantire che un programma scritto in MBASIC funzioni su tutte le macchine che adottano questo linguaggio. Infatti, ciascun apparecchio è pur sempre progettato in modo differente dagli altri.
L'obiettivo della ASCII/Microsoft era la creazione, in collaborazione con le industrie "leader" giapponesi, di uno standard che fosse riconosciuto a livello internazionale. Il prodotto finale della collaborazione è stato chiamato MSX Standard e fissa una serie di caratteristiche essenziali, sia per quanto concerne l'hardware, sia per quanto riguarda il linguaggio di programmazione BASIC.

Lo standard MSX

L'MSX BASIC è molto simile al Microsoft MBASIC, ma è dotato di significativi miglioramenti, onde utilizzare appieno la gestione grafica e le capacità di produrre effetti audio. Le nuove istruzioni comprendono: SCREEN, per definire il funzionamento del video, le dimensioni degli sprite, il "click" dei tasti, la velocità di scambio dati con i nastri a cassetta e le opzioni relative alla stampante; LOCATE, per posizionare un carattere sullo schermo; COLOR, per selezionare uno dei 16 colori disponibili; PUT SPRITE, per definire le caratteristiche o attributi di uno sprite; CIRCLE, per tracciare cerchi ed ellissi; DRAW, per disegnare figure; LINE, per tracciare linee tra due punti; PAINT, per riempire le figure con un particolare colore. L'istruzione KEY permette di assegnare interi comandi ai tasti di funzione. Altre istruzioni permettono di inserire dei valori nella RAM video (VPOKE) e nei registri dei chip per gli effetti sonori (SOUND); altre ancora servono per controllare il motore del registratore (MOTOR).
L'MSX, tuttavia, oltre a definire uno standard per il software, fa specifico riferimento anche ad alcune componenti hardware: la CPU deve essere uno Z80 che funziona a 3,85 MHz; ci devono essere almeno 32 Kbyte di ROM per immagazzinare il software MSX e almeno 8 Kbyte di RAM (non viene posto alcun limite massimo né alla ROM né alla RAM). Un computer MSX deve possedere un chip di controllo del video Texas Instruments TMS9918A (o un suo equivalente) ed un chip di controllo del suono a tre voci AY-3-8910. La gestione del video deve prevedere 32 colonne per 24 linee oppure 40 colonne per 24 linee. Attualmente non è previsto un formato ad 80 colonne. Inoltre, è stabilita una risoluzione grafica di 256 × 192 pixel. C'è uno standard anche per le interfacce per registratore, considerate dall'MSX le principali unità di immagazzinamento per programmi e dati. Il computer deve usare il sistema di codifica FSK a 1200/2400 bit per secondo. La tastiera, evidenziando l'origine nipponica dell'MSX, non solo deve avere una configurazione standard, comprendente i tasti di funzione, ma deve anche consentire l'uso del Katakana (l'alfabeto "angolare" giapponese), l'Hiragana (l'alfabeto "corsivo"), caratteri grafici standardizzati e, a richiesta, il Kanji (il sistema di scrittura cinese).



STEVE CROSS

Deve essere previsto un connettore per cartucce ROM, uno per i joystick ed anche un bus di ingresso/uscita a 50 piedini.
Unificati sono anche il formato dei dischetti (con scelta tra i 3 1/2 pollici della Sony, i 5 1/4 e gli 8 pollici) ed il sistema operativo, chiamato MSX-DOS. Questo è funzionalmente equivalente all'MS-DOS e permette di leggere i file dati di quest'ultimo. Sembra, addirittura, che l'MSX-DOS sia anche compatibile con il popolarissimo CP/M 2.2.
Tutto ciò significa che, aderendo allo standard, ogni programma scritto su un computer MSX ed immagazzinato su un disco di qualsiasi formato, potrà girare su un qualunque altro computer MSX e sfruttare appieno le sue capacità di suono e di grafica. I vantaggi sia per il fabbricante che per l'utente appaiono evidenti. Tuttavia, il sistema MSX presenta anche degli svantaggi, fra i quali è comune a tutti gli standard la scarsa elasticità nei confronti delle future innovazioni nello specifico settore. Se, ad esempio, venisse prodotto un nuovo chip di controllo video con caratteristiche superiori, i programmi MSX non potrebbero trarne vantaggio e lascerebbero campo libero ai prodotti concorrenti dotati di software capace di utilizzare le potenzialità del nuovo chip.
Il secondo inconveniente è che l'MSX è basato sui microprocessori ad 8 bit (di cui lo Z80 è il rappresentante di maggior successo) e questi hanno vita limitata: strutturalmente incapaci di indirizzare direttamente più di 64 Kbyte di memoria, non possono gestire dati più grandi, in valore numerico, di 256.
Sotto questa luce, lo standard MSX potrebbe essere interpretato come l'estremo tentativo di prolungare la vita dello Z80.
L'MSX permette quindi di formulare alcune ipotesi per il futuro. È difficile credere che tra 5 o 10 anni il mercato dei microcomputer sarà ancora dominato da microprocessori ad 8 bit. Se lo standard MSX darà risultati positivi, probabilmente i progettisti di computer si convinceranno che la standardizzazione dovrà essere una componente fondamentale per qualsiasi futuro sviluppo della tecnologia dei computer. Tuttavia l'MSX renderà ancora grandi servigi a quei fabbricanti le cui macchine sono basate sullo Z80 (la sua parabola è ormai discendente, ma non si può negare che questo componente abbia rivestito un ruolo fondamentale nella nascita e nello sviluppo della standardizzazione). Per il momento l'MSX è stato adottato da 16 case costruttrici tra cui: le giapponesi Yamaha, JVC, Hitachi, Sony, Sanyo, National, Pioneer, Canon, Fujitsu, Mitsubishi; l'americana Spectravideo e la coreana Daewoo.

Progetto standardizzato

Per creare programmi e accessori compatibili con tutti i sistemi MSX, il progetto di un micro MSX deve attenersi a precise norme. Una volta soddisfatti i requisiti principali, illustrati in figura, i progettisti possono aggiungere altri accessori e periferiche

Contatti... logici

Passiamo alla pratica con qualche semplice circuito logico

I transistor svolgono due funzioni essenziali: si comportano come amplificatori di un segnale oppure permettono o meno il passaggio di una corrente in conseguenza di un secondo flusso di corrente. È questa capacità di comportarsi come interruttori elettronici, che fa del transistor un componente di base dei computer. Collegandone assieme più d'uno, è possibile costruire dei circuiti capaci di immagazzinare configurazioni di bit, interpretati dal computer come numeri binari. Altri circuiti basati sui transistor sono le porte logiche che permettono di sommare tra loro sequenze di segnali ecc.

Se costruiamo le porte logiche descritte nella pagina seguente, ci accorgiamo che, qualora i computer fossero basati sul solo impiego di transistor, sarebbero molto costosi e di grosse dimensioni: così infatti erano i primi computer. Per divenire macchine di piccole dimensioni e di prezzo ragionevole, i computer avevano bisogno di una nuova tecnologia: quella dei circuiti integrati. Questi sono composti da centinaia di transistor, inseriti in un minuscolo chip di silicio, contenuto, a sua volta, in un piccolo involucro di plastica nera. Oggi i circuiti integrati più semplici (chiamati "chip TTL") sono comunemente in vendita: quattro porte AND raccolte in un chip costano poche centinaia di lire. I chip per eseguire lavori più complessi, come contatori binari o altri, costano poche migliaia di lire.

La procedura seguita per assemblare più circuiti su uno stesso chip viene spesso chiamata "integrazione su scala molto larga" (la sigla inglese è VLSI, per Very Large Integration Scale). I chip VLSI sono costituiti da molte centinaia di componenti fittamente stipati e in grado di eseguire operazioni molto complesse. Il microprocessore è uno dei chip che generano un'immagine televisiva; controlla le interfacce e produce quella vasta gamma di effetti sonori, comuni nei microcomputer. I principi sono gli stessi che abbiamo usato per le tre principali porte logiche: cambia solo il numero dei componenti utilizzati e la complessità dei circuiti.

I chip VLSI sono fabbricati con tecnologie di diverso tipo, scelte in base al più conveniente compromesso tra costi, prestazioni e consumo energetico. La tecnica più usata, nella produzione dei computer, è quella MOS (metallo/ossido/silicio), mentre per molti apparecchi portatili a pile vengono usati chip CMOS (MOS Complementari), più lenti ma più economici dal punto di vista energetico. In una prossima lezione, verrà descritta la costruzione di un semisommatore, in base a quanto esposto in via teorica a pag. 513.

Transistor come amplificatori

Un transistor si compone di tre parti: base, collettore ed emettitore. Se applichiamo una corrente alla base, tra collettore ed emettitore ci sarà passaggio di corrente. Un cambiamento nel voltaggio della corrente applicata alla base produrrà una conseguente variazione della resistenza offerta dal collettore alla corrente che lo attraversa. Se applichiamo una forte corrente al collettore ed induciamo in essa variazioni mediante una piccola corrente applicata alla base, l'emettitore produrrà un potente segnale che varia al variare della corrente che attraversa la base. Il transistor è stato quindi utilizzato per amplificare un piccolo segnale in ingresso



Transistor come interruttori

L'uso di un transistor come interruttore implica il pieno sfruttamento delle proprietà della base, del collettore e dell'emettitore. Il transistor permette il passaggio, dal collettore all'emettitore, solo di una corrente con un particolare voltaggio. C'è quindi un livello di soglia, raggiunto il quale, la corrente che scorre dal collettore all'emettitore diviene insensibile alle piccole variazioni prodotte dalla corrente che attraversa la base



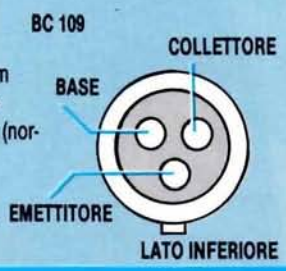
COSTRUZIONE DI PORTE LOGICHE

Questi semplici circuiti chiariscono come sia possibile sfruttare la capacità dei transistor di funzionare come interruttori. I circuiti possono essere costruiti uno alla volta, usando la stessa serie di componenti ed un'apposita basetta sperimentale.

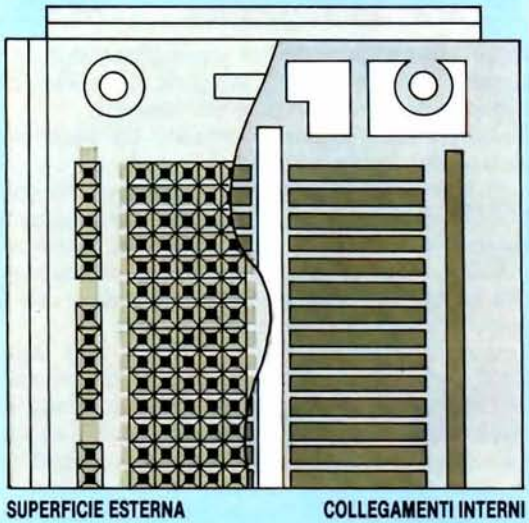
Nei seguenti esempi gli ingressi dei circuiti sono costituiti da pulsanti dotati di LED. In un computer gli ingressi applicati a circuiti simili a questi sono i segnali provenienti da altri circuiti. Una volta costruito e capito il funzionamento di queste porte, potremo dedicarci alla costruzione di circuiti più complessi, collegando l'uscita di una porta logica all'ingresso di una seconda porta

ELENCO DEI COMPONENTI

1 Basetta sperimentale
2 Transistor BC109
2 LED rossi
1 LED verde
3 Resistenze da 500 ohm
2 Resistenze da 15 ohm
2 Interruttori a pulsante (normalmente aperti)
1 Batteria a 9 volt
1 Morsetto per batteria
Alcuni cavetti elettrici

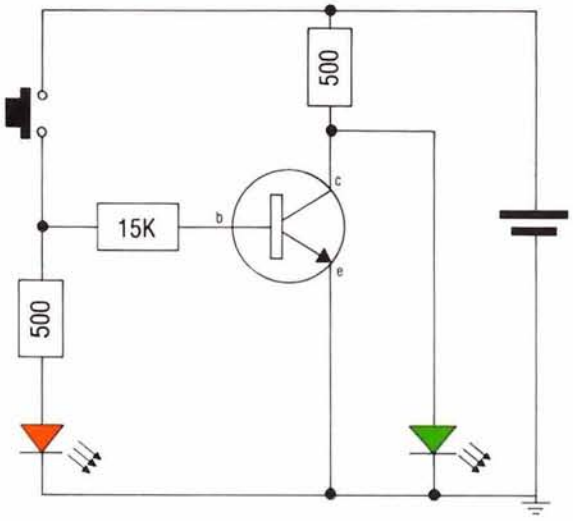


BASETTE SPERIMENTALI



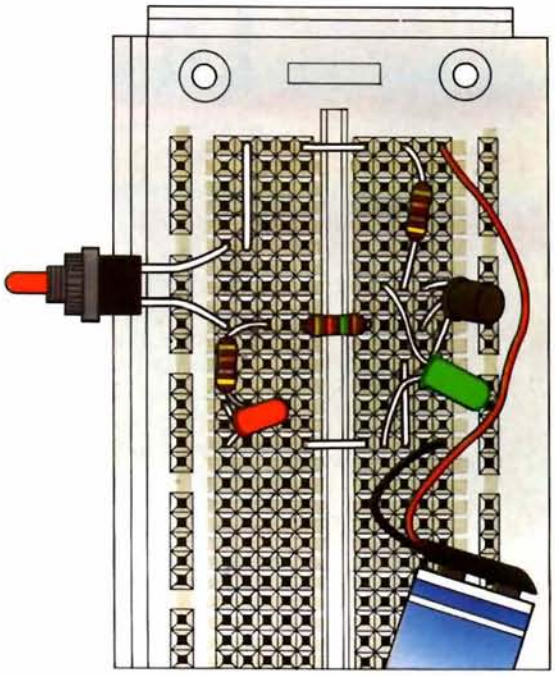
Le basette sperimentali permettono di eseguire esperimenti con circuiti come quelli mostrati nella pagina seguente senza dover saldare tra loro i componenti. La superficie esterna di uno di questi telaietti presenta una serie di fori, disposti a matrice, che garantiscono un perfetto contatto tra i vari componenti innestati su di essa. I fori metallici di innesto funzionano infatti come conduttori e i 5 fori che compongono ciascun gruppo sono collegati elettricamente tra loro. Grazie a questa matrice, risulta facile costruire semplici circuiti usando corti cavetti elettrici per collegare i vari gruppi di fori. La scheda in figura è molto più grande di quella necessaria per costruire le porte logiche

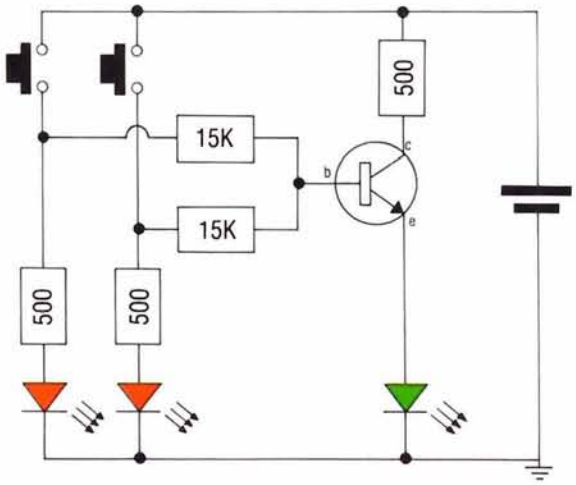
KEVIN JONES



La porta NOT

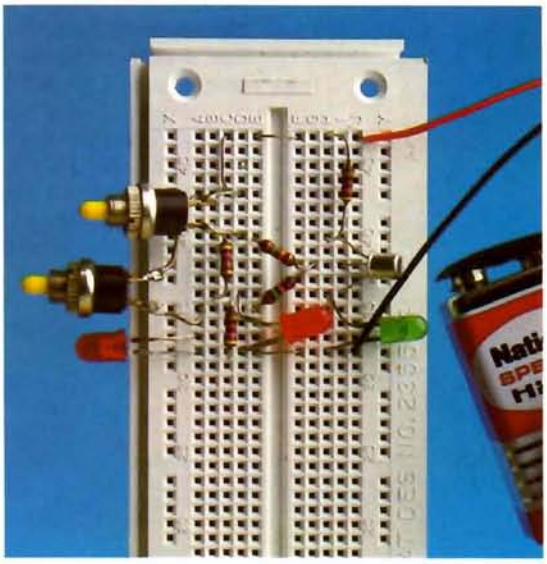
È la più semplice delle porte logiche: ha un ingresso, indicato dall'interruttore a LED rosso, ed una sola uscita, il LED verde. Quando l'interruttore è aperto, la corrente non può attraversare la base per raggiungere l'emettitore del transistor. Ciò induce un'alta resistenza sul collettore e fa sì che la corrente venga deviata sul circuito che ha come output il LED verde. Perciò, quando l'interruttore non viene premuto (cioè l'input è 0), il LED verde risulta acceso (cioè l'output è 1). Quando si preme il pulsante, una corrente attraversa la base del transistor eliminando la resistenza offerta dal collettore. La corrente ora attraversa il transistor e non raggiunge il LED. Perciò quando il pulsante viene premuto (l'input è 1), il LED risulta spento (cioè l'output è 0)

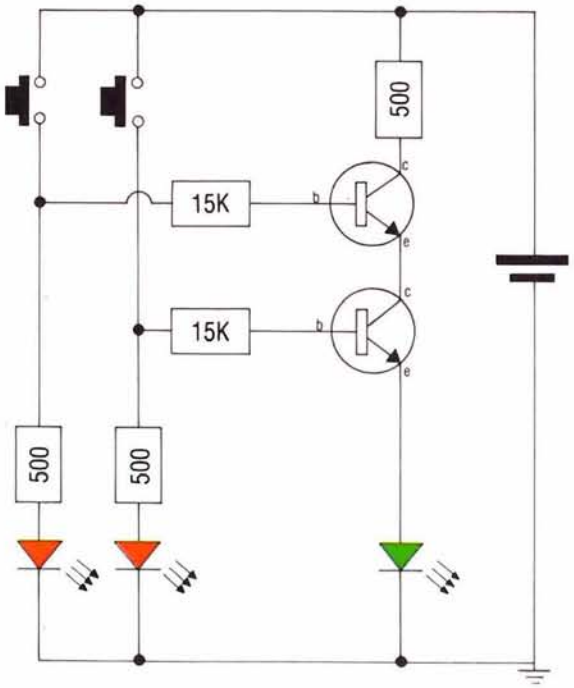

KEVIN JONES



La porta OR

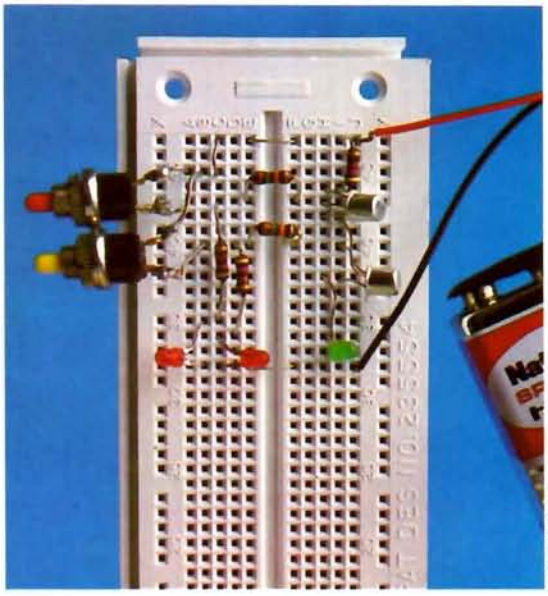
Il circuito usato per la porta NOT può essere facilmente adattato per costruire una porta OR. Come primo cambiamento, dobbiamo posizionare il LED di output (verde) in modo che venga raggiunto dalla corrente che attraversa il transistor. Ci sono due ingressi in una porta OR, ciascuno con interruttore e LED propri. Quando entrambi gli interruttori inviano un segnale alla base, il transistor verrà attraversato dalla corrente ed il LED verde risulterà acceso





La porta AND

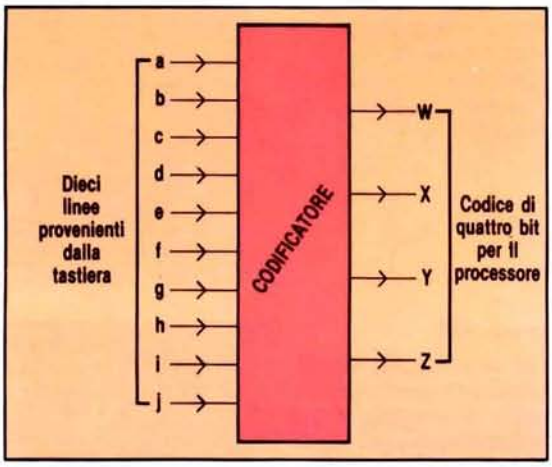
Per costruire una porta AND dobbiamo inserire due transistor sul circuito di uscita, ciascuno con un proprio ingresso. Solo quando entrambi gli interruttori sono premuti (cioè il loro input è 1) ambedue i transistor vengono attraversati dalla corrente che raggiungerà il LED verde accendendolo


IAN McKINNELL

La conversione dei segnali

Il funzionamento ed il progetto di codificatori e decodificatori

Talvolta, per ridurre al minimo il numero di linee utilizzate da una periferica, viene adottata una procedura di codifica. Un classico esempio di circuito codificatore è quello generalmente collegato alla tastiera. Teoricamente, questa avrebbe 64 linee di output, ciascuna delle quali emette un segnale, premendo il tasto ad essa corrispondente. Poiché i tasti vengono premuti uno alla volta, ciascuno dei 64 possibili segnali di output può essere codificato sotto forma di numero binario a 6 bit. In altre parole, sono sufficienti 6 linee per comunicare al microprocessore quale tasto è stato battuto. In effetti, vengono utilizzate altre due linee: una necessaria alla procedura per il controllo di parità, l'altra per segnalare quando viene premuto, contemporaneamente a un altro tasto, lo Shift o il Control. Per meglio chiarire il concetto, consideriamo una tastiera molto più semplice, dotata solo di 10 tasti, che permetta di digitare un numero compreso tra 0 e 9. Con un codice binario a tre bit esistono soltanto otto possibili combinazioni, quindi il codificatore deve essere in grado di accettare 10 linee di input e di produrre 4 linee di output.



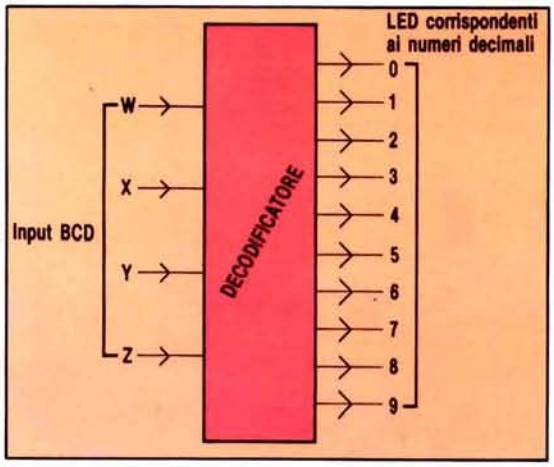
Poiché può essere attiva una sola linea alla volta, la tavola della verità per il codificatore è la seguente:

Decimale	Ingressi										Uscite			
	a	b	c	d	e	f	g	h	i	j	W	X	Y	Z
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	0	0	0	0	1	0	1
6	0	0	0	0	0	0	1	0	0	0	0	1	1	0
7	0	0	0	0	0	0	0	1	0	0	0	1	1	1
8	0	0	0	0	0	0	0	0	1	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	1	1	0	0	1

I circuiti decodificatori eseguono una procedura inversa rispetto a quella dei codificatori: invece di avere un gran numero di ingressi ed un piccolo numero di uscite, questi dispositivi accettano un piccolo numero di segnali in ingresso (provenienti dal processore sotto forma di codici binari) e inviano in uscita un gran numero di segnali, adatti a controllare, per esempio, il funzionamento di un plotter o di una stampante. Entrambi i circuiti descritti vengono comunemente usati per controllare i movimenti della testina di lettura dei dischetti e per selezionare i canali di output sulla base dei codici assegnati alle varie periferiche.

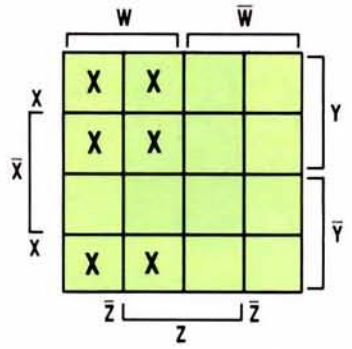
Progetto di decodificatore

Vediamo adesso come sia possibile progettare un semplice decodificatore, usando le porte logiche AND, OR e NOT. Consideriamo il seguente problema: un sistema di elaborazione ha bisogno di un decodificatore per convertire numeri decimali codificati in binario (BCD) in una forma che permetta di accendere uno dei dieci LED corrispondenti al valore decimale del codice. Il circuito deve dunque comportarsi in modo inverso rispetto all'esempio del codificatore visto in precedenza.

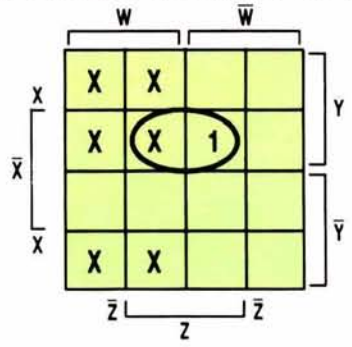


I codici BCD sono una rappresentazione binaria a 4 bit delle cifre decimali comprese tra 0 e 9, quindi il decodificatore deve possedere quattro linee di ingresso. Poiché su queste può presentarsi una qualsiasi combinazione di input, le possibili configurazioni in ingresso sono 16. Prendiamo in considerazione solo le prime 10 combinazioni, attribuendo alle altre la condizione (X) di input non valido. La Tabella 1 rappresenta la tavola di verità. Consideriamo adesso ciascuno dei 10 output

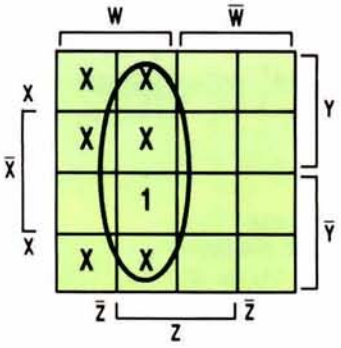
separatamente. A prima vista, sembrerebbe che l'espressione Booleana per ogni output debba essere composta da 4 termini: W, X, Y e Z, come in Tabella 2. Tuttavia, sono possibili alcune semplificazioni. Dal momento che le condizioni di "input non valido" sono le stesse per ciascuno dei 10 output, possiamo collegare tra loro i termini per semplificarli. La seguente è una mappa di Karnaugh a 4 variabili sulla quale sono state segnalate le 6 condizioni di input non valido.



Esaminando ciascuna uscita, in successione, ci accorgiamo che è possibile fare qualche semplificazione. Ad esempio, l'output della cifra 3: riportando l'espressione Booleana sulla mappa-k, ci accorgiamo che è possibile tracciare un'ellisse intorno ad essa:



Pertanto, l'espressione può essere semplificata a: X.Y.Z. Prendiamo adesso il caso dell'uscita 9:



Si può tracciare un'ellisse che includa tre delle condizioni di input non valido e che rappresenti l'espressione Booleana W.Z. Molti degli altri termini di uscita possono venir semplificati allo stesso modo. Se eseguiamo una verifica, vedremo che i termini di output semplificati corrispondono a quelli mostrati in Tabella 3.

Il lavoro di progettazione termina con la costruzione del diagramma del circuito sulla base delle 10 espressioni Booleane. Poichéciascun input viene usato sia nella sua forma normale che in quella negativa, è più semplice costruire il circuito partendo dalle 8 linee parallele che rappresentano i termini W, X, Y e Z. Ciascuna delle 10 uscite verrà sviluppata creando delle biforcazioni sulle linee pertinenti, usando porte AND. Il diagramma completo del circuito di decodifica è mostrato nella figura in basso.

Esercizio 6

1) Progettare un codificatore a tre ingressi, capace di produrre un'uscita di valore 1 per i valori in ingresso 011, 101, 110 o 111. L'uscita deve essere 0 per tutte le altre combinazioni d'ingresso.
a) Tracciare la tavola della verità per il codificatore
b) Generare e semplificare un'espressione Booleana per l'output
c) Disegnare il circuito del codificatore

Tabella 1

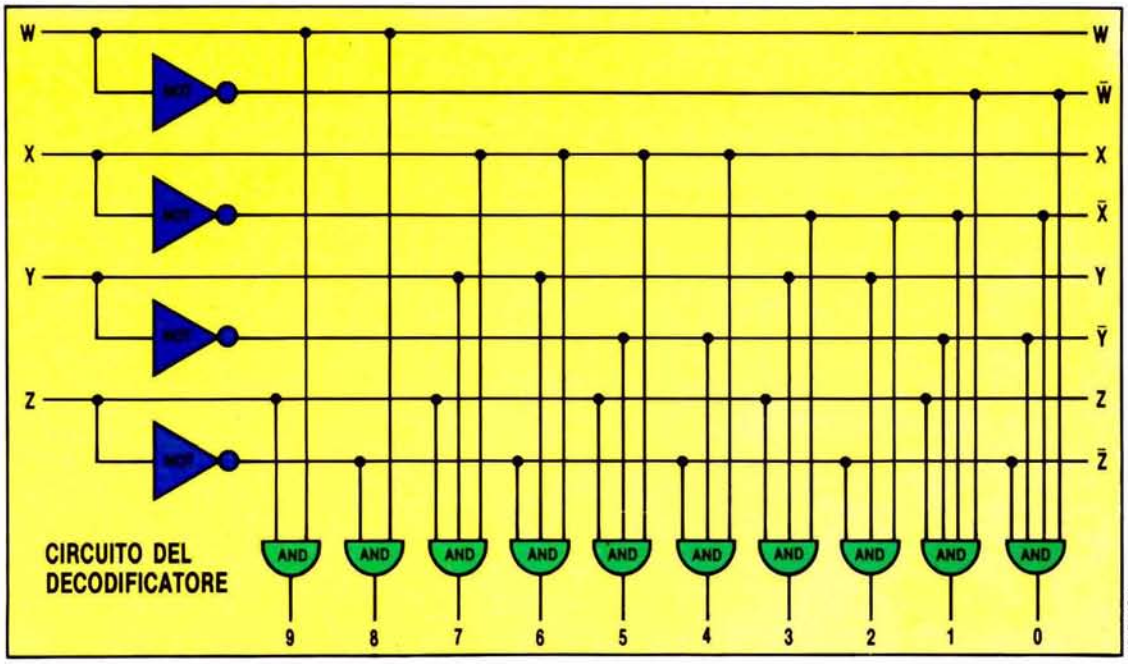
Ingressi				Numeri BCD
W	X	Y	Z	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Tabella 2

Numeri BCD	Espressione Booleana
0	$\bar{W}.\bar{X}.\bar{Y}.\bar{Z}$
1	$\bar{W}.\bar{X}.\bar{Y}.Z$
2	$\bar{W}.\bar{X}.Y.\bar{Z}$
3	$\bar{W}.\bar{X}.Y.Z$
4	$\bar{W}.X.\bar{Y}.\bar{Z}$
5	$\bar{W}.X.\bar{Y}.Z$
6	$\bar{W}.X.Y.\bar{Z}$
7	$\bar{W}.X.Y.Z$
8	$W.\bar{X}.\bar{Y}.\bar{Z}$
9	$W.\bar{X}.\bar{Y}.Z$

Tabella 3

Numeri BCD	Espressione Booleana
0	$\bar{W}.\bar{X}.\bar{Y}.\bar{Z}$
1	$\bar{W}.\bar{X}.\bar{Y}.Z$
2	$\bar{X}.Y.\bar{Z}$
3	$\bar{X}.Y.Z$
4	$X.\bar{Y}.\bar{Z}$
5	$X.\bar{Y}.Z$
6	$X.Y.\bar{Z}$
7	$X.Y.Z$
8	$W.\bar{Z}$
9	$W.Z$



LIZ DIXON

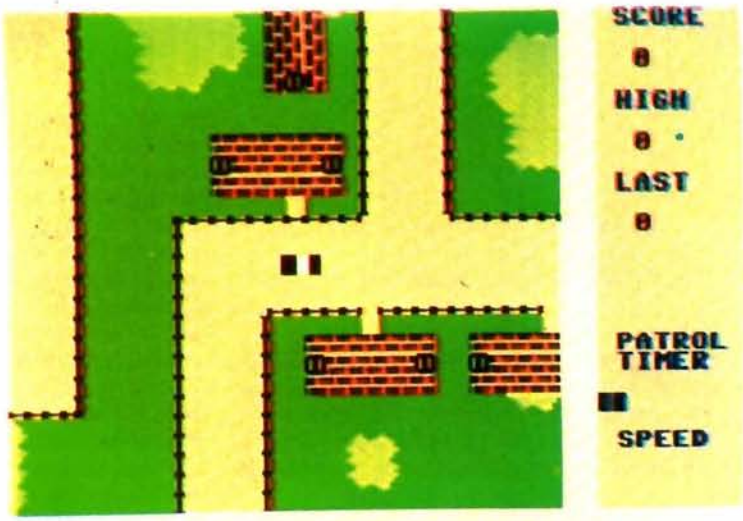
Software in vetrina

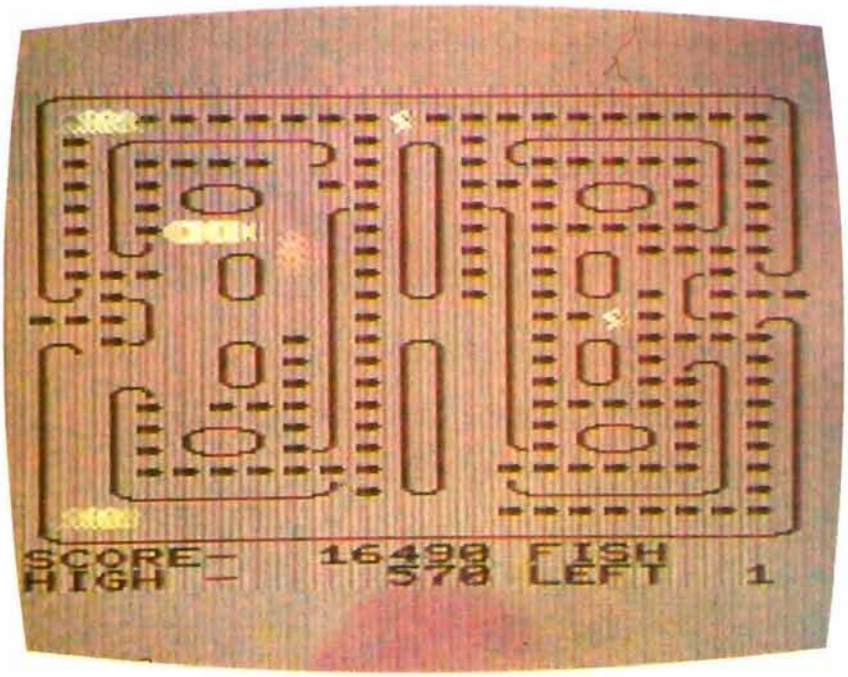
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

La panoramica sui giochi prosegue con altri due programmi, concepiti per il Commodore 64 ed il Vic-20. Si tratta, ancora una volta, di giochi basati sui labirinti, un settore che va per la maggiore.

Siren City per il Commodore 64

Questo è forse uno tra i più originali giochi del genere. Il giocatore assume il ruolo di un poliziotto americano, in pattuglia tra le strade di New York. Vengono offerti 10 livelli di difficoltà: dal semplice gironzolare con l'auto intorno agli isolati, a frenetiche scorribande nel tentativo di catturare insidiosi malviventi. La grafica è molto accurata (la prospettiva è dall'alto). Il pulsante di "fuoco" del joystick serve per compiere improvvise svolte ed è particolarmente utile negli inseguimenti.
Il gioco è prodotto dalla Interceptor Software.





Pakakuda per il Vic-20

Rispetto al precedente, questo gioco è forse meno emozionante e vivace: appartiene alla foltissima schiera dei vari "Pacman", pertanto la trama, salvo minime variazioni, è la solita. Il protagonista, in questo caso, "campa" divorando pesci ed anguille. L'avversario è costituito da grosse piovre. Anche queste possono essere "mangiate", purché si abbia fatta una buona scorta di anguille!
Tutto sommato, il Pakakuda lascia con un certo sapore di "déjà vu".

Lo Spectravideo 318

Uno dei primi apparecchi conformi alle specifiche dello standard MSX

Lo Spectravideo 318 è un economico home computer di qualità pari al Micro della BBC ed al Commodore 64. Possiede tre sintetizzatori di suono, grafica ad alta risoluzione con sprite, un joystick inserito sulla custodia ed un porta cartucce.

La tastiera si avvicina alle specifiche dello standard MSX (vedere a pag. 621): come nello Spectrum Sinclair, i tasti sono ricavati da una lamina di gomma, tuttavia sono ben spaziati e di uso molto confortevole. Il controllo del cursore è affidato ad un joystick incorporato, situato a fianco della tastiera: una specie di piastra circolare sostituisce i più comuni 4 tasti a freccia e, facendo pressione su di essa, il cursore si muove in alto, in basso, a destra o a sinistra. In alternativa, è possibile inserire sulla piastra una piccola manopola che facilita lo spostamento del cursore, anche durante la correzione dei programmi.

Il BASIC dello Spectravideo è il più recente di una serie di interpreti prodotti dalla Microsoft. Questa versione è assai simile all'MSX BASIC, un'estensione del BASIC GW usato su macchine come l'IBM PC. La stesura dei programmi in BASIC è facilitata grazie ad un efficiente programma di editor e ad altre facilitazioni (quali la numerazione automatica delle linee di programma). Purtroppo mancano comandi "strutturati" (WHILE...WEND, REPEAT...UNTIL, ecc.) presenti invece nei BASIC più recenti. Tuttavia, esiste la struttura IF...THEN...ELSE, indispensabile per scrivere programmi chiari ed efficienti. Comunque, questo BASIC permette di sfruttare al meglio le caratteristiche grafiche dello Spectravideo.

Il video è dotato di una discreta risoluzione (256 × 192 pixel) in 16 colori, anche se il colore scelto influenza i punti a piccoli gruppi. Queste caratteristiche possono sembrare piuttosto limitate rispetto a quelle presenti su macchine come il Micro BBC, ma la resa grafica dipende più dalle modalità del controllo video, che non dalle caratteristiche tecniche della macchina. Il 318 possiede tutte le estensioni grafiche del Microsoft BASIC (singoli comandi per tracciare punti, linee, rettangoli, cerchi, archi ed ellissi), il comando PAINT permette di colorare una qualsiasi figu-



IAN McKINNELL

La tastiera Spectravideo

La tastiera del 318 possiede caratteristiche tecniche aderenti allo standard MSX. La sua struttura è simile a quella dello Spectrum, ma la buona spaziatura e qualità dei tasti ne fanno uno strumento confortevole. Oltre a possedere tutti i comuni tasti Control, Escape, Tab e la Barra Spaziatrice, presenta 5 tasti di funzione (ciascuno con doppio funzionamento), i tasti STOP e SELECT e vari tasti di editing come INS, DEL e COPY. Possiede anche una serie di caratteri grafici (rettangoli, punti, linee) che vengono abilitati premendo contemporaneamente il tasto di una lettera ed il tasto LEFT GRAPH o RIGHT GRAPH. I simboli grafici, relativi a ciascun tasto, sono rappresentati in modo chiaro sulla tastiera

ra chiusa. Per una più accurata definizione, si possono utilizzare comandi come VPOKE e VPEEK, che permettono di scrivere e di leggere direttamente nell'area di memoria riservata al video. È anche disponibile un piccolo linguaggio grafico, usato con il comando DRAW per tracciare figure più complesse e diagrammi.

Il comando GET permette di prelevare una porzione rettangolare di video, trasferirla in una matrice BASIC e quindi, tramite il comando PUT, riprodurla nuovamente sullo schermo. Grazie a questi comandi è possibile ottenere facilmente figure armoniose, semplici animazioni o effetti speciali come la riproduzione "in negativo" di un'immagine. Infine il 318 possiede una buona grafica a sprite. Il chip di visualizzazione 9929 permette al programmatore di progettare proprie animazioni, quali: figure di persone, invasori spaziali o missili.

L'istruzione ON SPRITE GOSUB permette di richiamare un sottoprogramma detto "rivelatore di eventi": l'animazione si sviluppa normalmente ma, quando due sprite entrano in contatto, il BASIC passerà il controllo ad una particolare routine che prende in esame la collisione. Questa procedura permette, ad esempio, di rivelare una collisione tra un missile ed una astronave, liberando il programmatore dalla necessità di verificare ripetutamente tutte le possibili collisioni. La buona qualità del suono è ormai una caratteristica essenziale ed il chip impiegato nello Spectravideo possiede, oltre a tre voci, una serie di effetti speciali che permettono di ottenere risultati di ottimo livello, benché in BASIC ciò sia abbastanza difficoltoso. Il segnale audio viene inviato all'apparecchio televisivo collegato al computer.

Lo Spectravideo è dotato delle seguenti interfacce: due porte per joystick, una porta per cartucce, una per cassette ed un connettore di espansione. Gli accessori disponibili sono numerosi, ma piuttosto costosi. Inoltre, per collegarli all'apparecchio, occorre acquistare un apposito "mini-espansore", che consente l'allacciamento di una sola scheda aggiuntiva alla volta, oppure una seconda scheda d'espansione, detta "super-espansore", che permette, invece, di collegare contemporaneamente fino a 7 periferiche (stampanti, unità a dischetti, modem ed anche memoria aggiuntiva) mediante un sistema ad innesto simile a quello dell'Apple II. Gli appassionati di videogiochi possono acquistare un accessorio veramente interessante, l'adattatore per giochi Coleco, che permette di usare cartucce di giochi televisivi prodotte dalla Coleco.

Il 318 adotta esclusivamente un registratore a cassette fabbricato dalla stessa Spectravideo. Questa restrizione rende il sistema ancora più costoso, ma garantisce un alto livello di affidabilità ed una maggior velocità di immagazzinamento e di richiamo delle informazioni.



IAN McKINNELL

Il joystick dello Spectravideo

Incorporato nella macchina sostituisce i quattro tasti a freccia. In assenza della manopola, si esercita pressione sulla piastra circolare, posta sulla destra della tastiera, per muovere il cursore nelle quattro direzioni. Inserendo la manopola sulla piastra, è possibile orientare il cursore ancor più efficacemente. La presenza di un joystick incorporato è qualcosa di più di un'innovazione estetica: esso infatti permette di muovere il cursore diagonalmente



Porta per cartucce

Le cartucce vengono inserite in un alloggiamento fisso, situato nella parte superiore del contenitore

ROM

Il BASIC dello Spectravideo è conservato in due ROM da 16 Kbyte

Connettore per espansioni

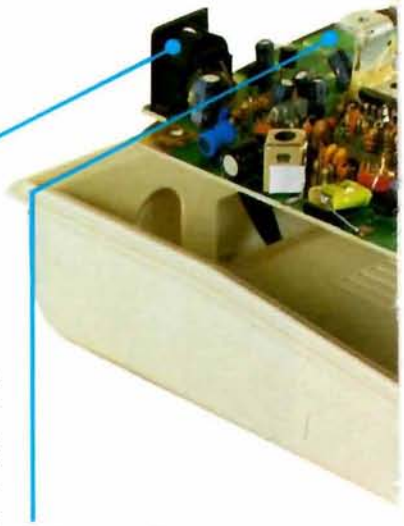
Ad esso vengono collegate una serie di espansioni e di accessori

CPU

Il microprocessore è il notissimo Z80

Uscita per il monitor

Un modulatore RF, esterno, permette il collegamento con un apparecchio televisivo. Selezionando il modulatore appropriato, lo Spectravideo può essere usato con apparecchi televisivi caratterizzati da standard diversi



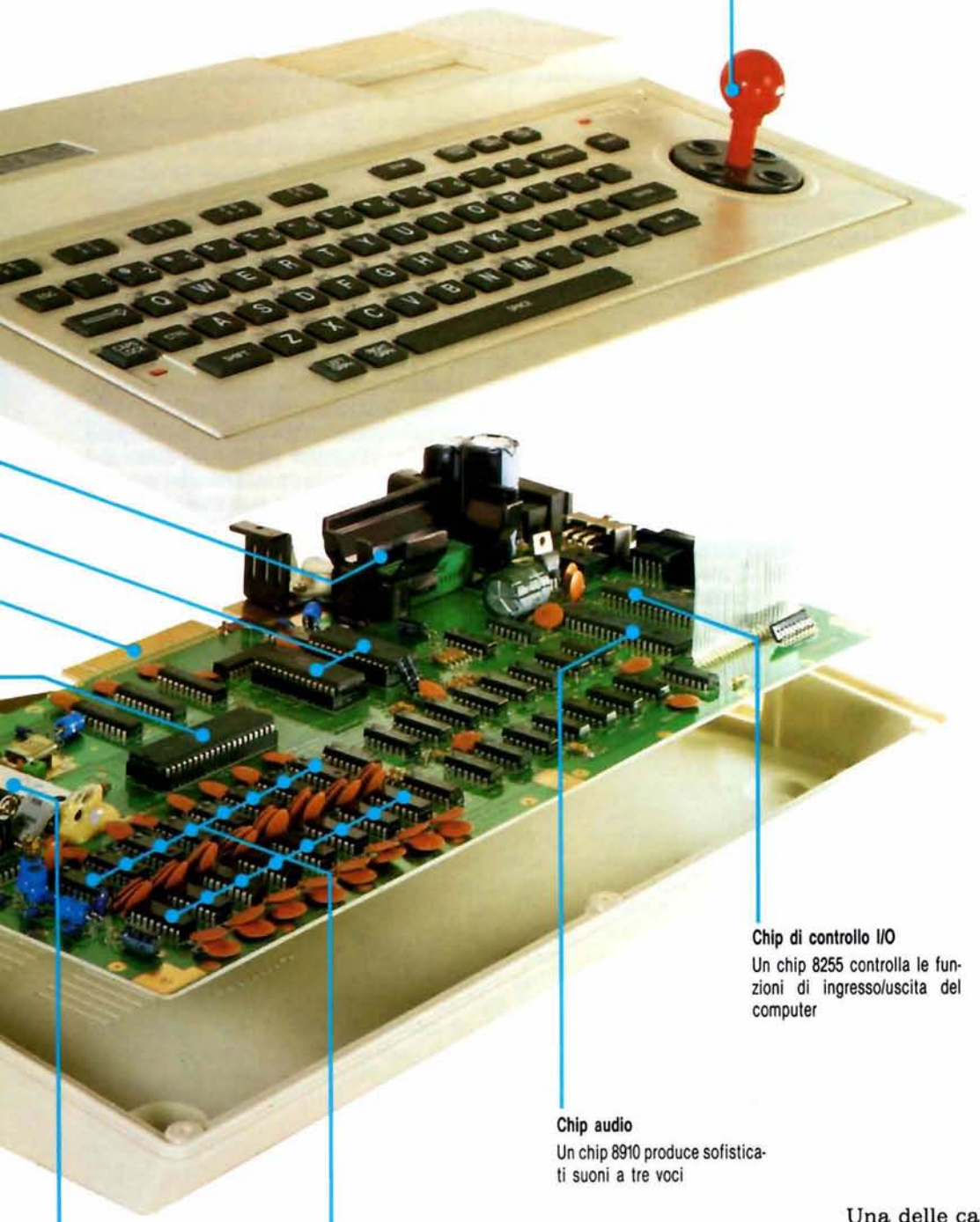
Porta per cassette

Uno speciale connettore "a innesto" consente il collegamento dello Spectravideo all'apposito registratore a cassette



Il fratello maggiore

Un'interessante alternativa al 318, è lo Spectravideo 328. Questa versione più sofisticata possiede una tastiera tipo macchina da scrivere, 80 Kbyte di RAM ed un word processor incorporato. Il suo costo è relativamente basso ed è una macchina progettata per il mercato dei computer commerciali



Joystick incorporato
Una piastrina circolare, sulla quale viene innestato il joystick, sostituisce i comuni tasti a freccia

Chip di controllo I/O
Un chip 8255 controlla le funzioni di ingresso/uscita del computer

Chip audio
Un chip 8910 produce sofisticati suoni a tre voci

Chip di visualizzazione
Un dissipatore (aggiunto in un secondo momento) nasconde un chip di visualizzazione 9929. Simili frettolosi interventi compaiono spesso sulle prime versioni delle macchine

RAM
16 chip RAM forniscono 32 Kbyte di RAM

SPECTRAVIDEO 318

DIMENSIONI

mm. 410 x 220 x 80

CPU

Z80A

MEMORIA

32 Kbyte di RAM (di cui circa 12 Kbyte per programmi in BASIC). 32 Kbyte di ROM

VIDEO

24 linee di 40 caratteri. Per la grafica in alta risoluzione: 256 x 192 pixel, 16 colori e possibilità di usare sprite. Come opzione, si può avere uno schermo ad 80 colonne

INTERFACCE

Connettore per espansioni, connettore per cartucce, due connettori per joystick

LINGUAGGI DISPONIBILI

BASIC

TASTIERA

A tasti mobili su unica base di gomma, ma ben distanziati e comodi all'uso. Un joystick incorporato

DOCUMENTAZIONE

Potrebbe essere migliore. Probabilmente, molte case indipendenti produrranno libri e riviste che sopperiranno alle carenze dei manuali

PRO

Le caratteristiche, molte delle quali conformi allo standard MSX, un BASIC eccellente, grafica a sprite, una buona tastiera ed un joystick incorporato. Discreta espandibilità

CONTRO

Grossa carenza, per il momento, di software. L'interfaccia per stampante è venduta a parte. Il computer funziona soltanto con uno speciale registratore

Una delle caratteristiche più interessanti del 318, è la sua capacità di espansione. Dotando l'apparecchio del super espansore, di una RAM a 64 Kbyte, della scheda per 80 colonne video e dell'unità a dischetti, si otterrà un piccolo computer CP/M per usi commerciali, in grado di utilizzare software commerciale professionale. Tuttavia, sarà difficile trovare programmi compatibili con il formato disco dello Spectravideo. Questo problema, comune agli apparecchi nuovi e relativamente poco conosciuti, tenderà a scomparire se il 318 otterrà un buon successo di mercato.

Tiriamo le somme

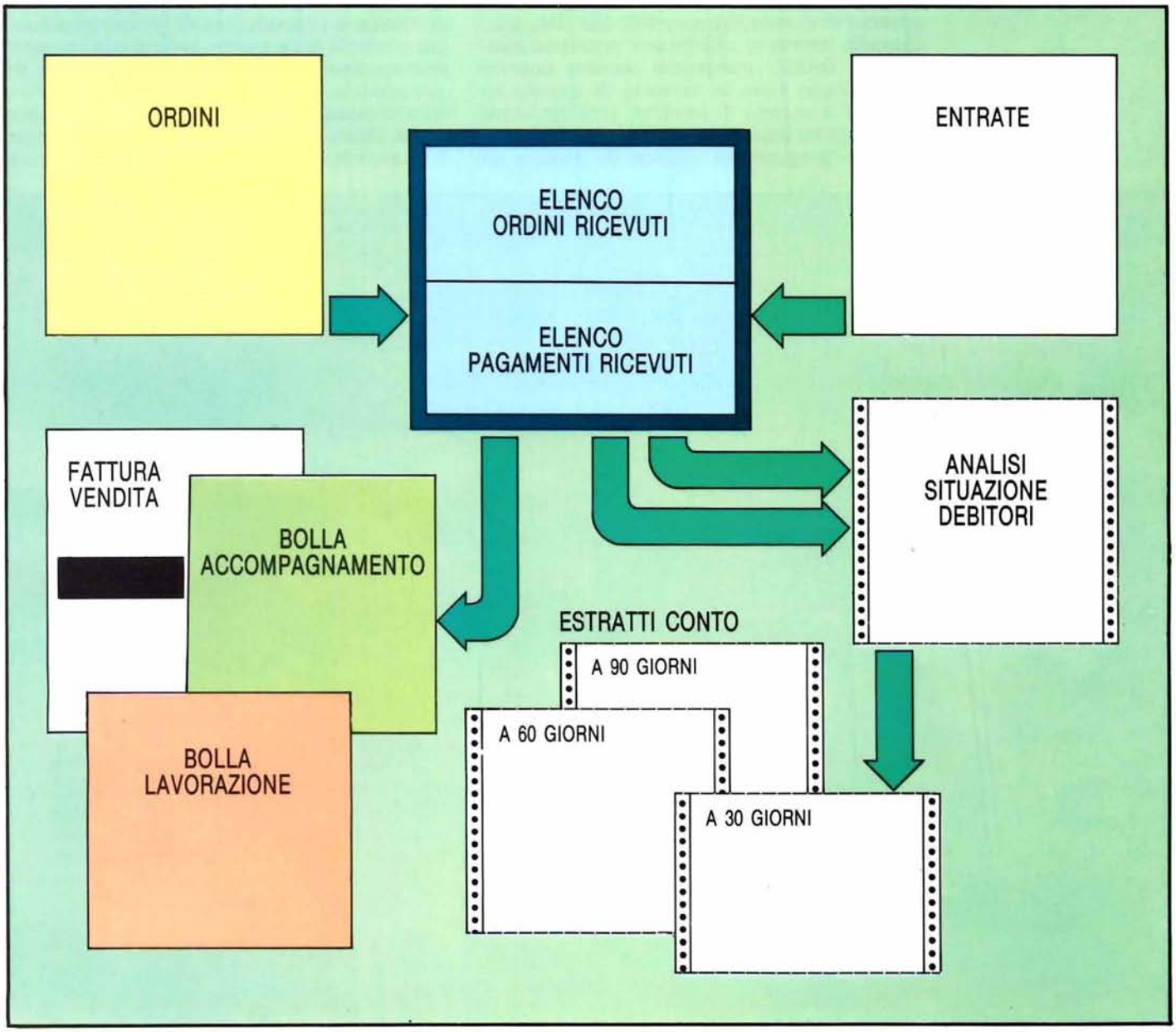
Nei programmi di contabilità è utile disporre di tutta una serie di stampe e tabulati

Oltre a registrare dati sugli acquisti e sulle vendite di beni e servizi, i "package" di contabilità dovrebbero anche offrire una certa quantità di resoconti stampati. Questi possono essere di due tipi: i resoconti gestionali (destinati ad un uso interno all'azienda) e quelli di carattere fiscale (Registri IVA, Bilanci, ecc.).
Tuttavia, come per tutti i sistemi di elaborazione dati, le informazioni di partenza devono già trovarsi all'interno del sistema, affinché quest'ultimo possa produrre dei resoconti.
I programmi scritti per computer dotati di solo registratore a cassette talvolta non hanno la possibilità di gestire archivi tali da produrre la stessa gran quantità di resoconti, solitamente ottenibili da programmi contabili che sfruttano, invece, unità a dischetti.
Una delle stampe ottenibili su tutti i sistemi di un certo livello è, invece, quella relativa alla "Lista delle Registrazioni", che costituisce una valida documentazione di tutte le transazioni effettuate.
Ovviamente, i sistemi migliori prevedono la tenuta di un archivio completo delle transazioni effettuate, per ciascun periodo contabile, relative ad ogni singolo conto (sia per i clienti, che per i fornitori), permettendo di stampare estratti conto (ed eventualmente fatture) da rimettere ai clienti, in cui vengono mostrati gli importi ancora scoperti ed una descrizione delle transazioni intercorse. Per i fornitori si possono ottenere una "nota d'avviso", che riassume la posizione contabile, ed anche indicazioni su ordinazioni che intendiamo fare.
Inoltre, può esser comodo disporre di una stampa del piano dei conti ordinata alfabeticamente (per quanto concerne Clienti e Fornitori) ed una, più generale, in ordine progressivo di codice del conto.
Altre opzioni di stampa riguardano resoconti analitici su periodi settimanali o mensili, bilanci di verifica, estratti sulla situazione di cassa o di ciascun conto corrente bancario, e via dicendo.
I resoconti relativi ai debitori ed ai creditori sono un ottimo esempio di un utilissimo strumento gestionale, che è piuttosto difficile realizzare con una procedura manuale. Al contrario, un sistema computerizzato, dotato di un programma ragionevolmente sofisticato, è in grado di estrarre questo tipo di informazioni rapidamente e senza alcun intervento supplementare da parte dell'utente. Tali resoconti possono illustrare gli importi dovuti o in arretrato, esaminati a seconda della loro data di scadenza (30, 60 o 90 giorni). In effetti, il procedimento grazie al quale si ottiene facilmente questa raccolta di informazioni è molto semplice: il programma non fa che leggere le date registrate assieme ad ogni singola vendita (o acquisto) e le riorganizza in ordine progressivo, in un nuovo archivio, selezionandole automaticamente nelle tre categorie di scadenza (a 30, a 60 o a 90 giorni). Quando l'importo viene registrato come "pagato", esso viene automaticamente eliminato dal procedimento di analisi.
La stampa del bilancio di verifica è un'opzione comune alla maggior parte dei sistemi e permette di stampare ordinatamente, per ciascun conto e sottoconto, il relativo totale dei progressivi "dare/avere". Dal momento che nella quasi totalità dei pacchetti contabili viene fatto uso del sistema "a partita doppia", la somma dei totali in "dare" ed in "avere", in qualsiasi momento, deve sempre risultare zero.
Qualora ciò non avvenisse, occorre riesaminare tutte le operazioni, alla ricerca di eventuali doppie od errate trascrizioni di un importo (ad esempio, l'aver scritto 45.000 invece di 54.000).
Sulla base del bilancio di verifica l'azienda può ricavare il resoconto dei profitti e delle perdite ed il bilancio definitivo d'esercizio. In alcuni casi, quest'ultima operazione è automatica, ma semplificata: viene semplicemente fatta la somma delle entrate e quella delle uscite, sottraendo, infine, il secondo totale dal primo.
Una simile procedura è adeguata alle necessità di piccole contabilità, ma non produce situazioni corrispondenti al reale, qualora si debba tener conto di altri fattori, che influiscono sui profitti e sulle perdite e per i quali è sempre necessario intervenire in modo semiautomatico.
Non tutti i resoconti di verifica possono essere ottenuti, infatti, con un così ridotto impegno da parte dell'utente, come nel caso del bilancio di verifica o dell'elenco dei debitori e dei creditori. Questi, come un paio degli altri resoconti ottenibili (ad esempio, la lista delle registrazioni), sono dei puri e semplici "archivi di deposito", all'interno dei quali ogni dato riceve una rudimentale impaginazione ed un'intestazione, per venir poi inviato in stampa. Altri resoconti sono molto più selettivi e sviluppare i criteri per la selezione dei dati può risultare assai complesso, indipendentemente dal pacchetto di programmi in uso.
In sistemi relativamente sofisticati, alcune

delle operazioni possono seguire un criterio selettivo. È infatti possibile assegnare, a ciascun conto o sottoconto, un particolare codice aggiuntivo, che ne indichi la categoria (ad esempio: 02xxx = produzione, 03xxx = vendita e 04xxx = acquisti materie prime, ecc.); allora potremo utilizzare detti codici per scoprire il costo di gestione relativo a ciascuna sezione. Ci saranno delle spese comuni (e quindi dei conti comuni) a tutte e tre le sezioni. Ma sfruttando la suddivisione vista poc'anzi, sarà possibile produrre un bilancio consuntivo o di previsione, che raggruppi i conti in base ai codici di sezione. Questo tipo di analisi può essere di estrema utilità nella compilazione di budget e per migliorare l'organizzazione interna dell'azienda.
In altri "packge" il codice impiegato per questo tipo di analisi fa parte dello stesso codice di conto: in tal caso, i due caratteri iniziali del codice individuano il gruppo (o categoria), mentre i tre (o quattro) caratteri seguenti identificano il particolare sottoconto. In tal modo, è possibile ottenere una stampa "riepilogativa" con gli importi totali di ciascun conto (ad esempio: Clienti, Fornitori, Banche, ecc.), oppure una stampa molto più dettagliata, relativa ad uno o più conti. Va purtroppo detto che, nonostante il numero di pacchetti di contabilità attualmente disponibile sia alquanto vasto, è quasi impossibile trovare già pronta in commercio una raccolta di programmi "ideale". Tutti, per un verso o per l'altro, ricorrono a dei compromessi (dovuti principalmente alla limitata quantità di memoria del computer), che li rendono adatti soltanto a particolari applicazioni.
Per quanto concerne, poi, opzioni di stampa più sofisticate (ad esempio statistiche di vendita, ripartizione dei costi, ecc.) il discorso si fa ancora più complesso e spesso occorre rivolgersi ad esperti programmatori per avere la redazione di appositi moduli aggiuntivi da collegare agli altri normali programmi di contabilità.

Resoconto istantaneo

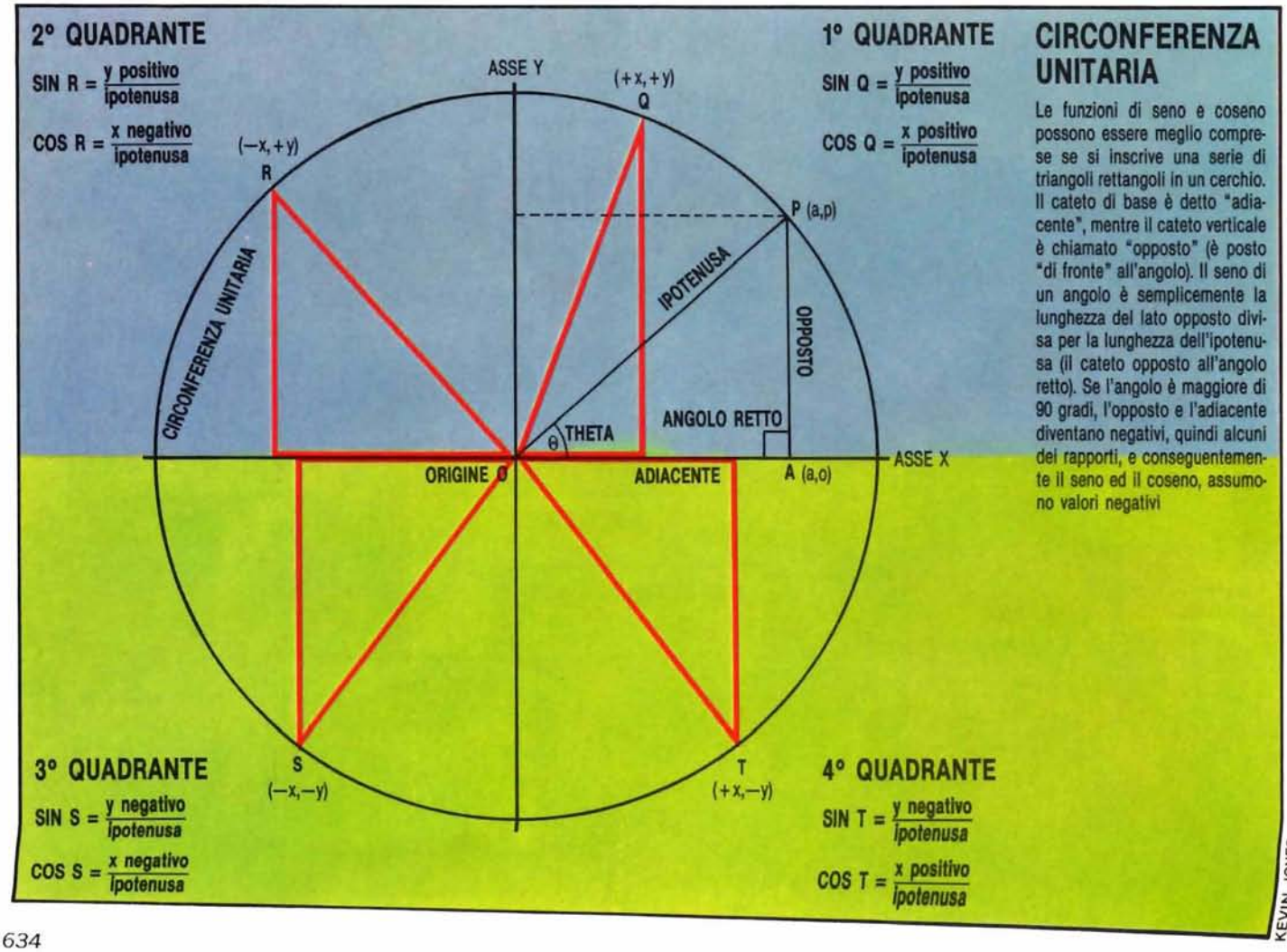
Un sistema computerizzato consente un accesso quasi istantaneo ad una vasta gamma di utili resoconti. In questo caso il sistema offre informazioni su tutte le transazioni, fornendo automaticamente anche un elenco dei clienti debitori. Sistemi più sofisticati possono prevedere l'uso di moduli prestampati per l'emissione di fatture, di bolle d'accompagnamento e via dicendo



Qualche grado... in BASIC

Alcune delle funzioni matematiche del BASIC possono sembrare di scarso interesse, ma...

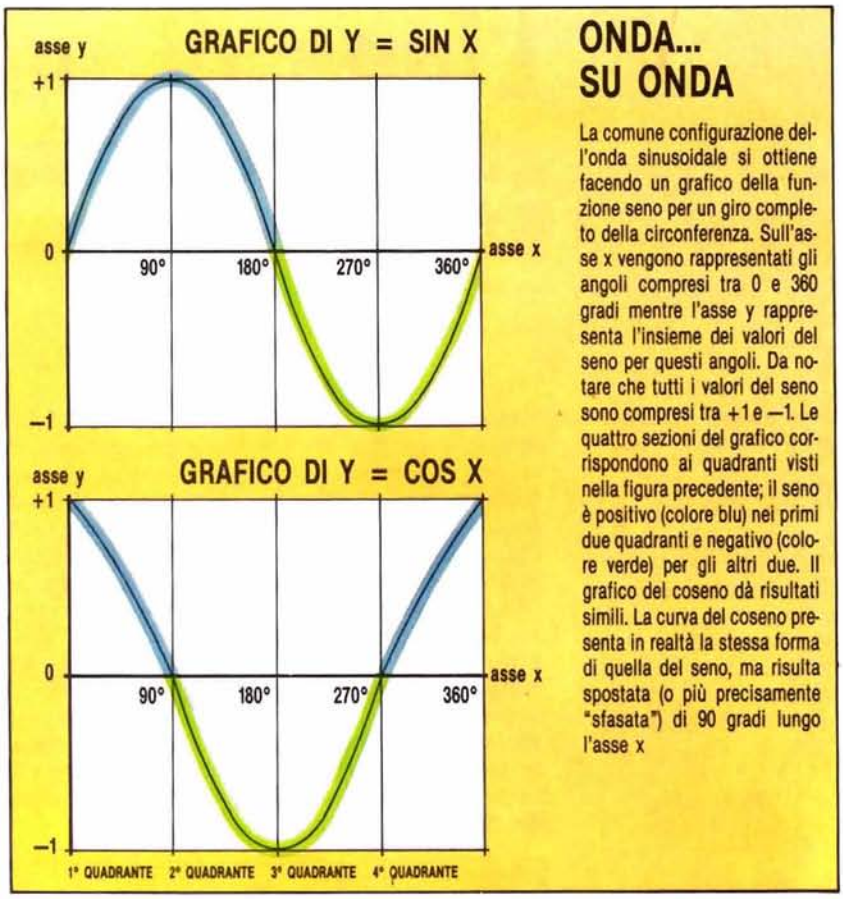
Di quale tipo di matematica si servono i programmatori? Ciò dipende, ovviamente, dal tipo di programma in corso di stesura. Le varie versioni BASIC sono caratterizzate da numerose istruzioni e funzioni per la gestione della grafica video (PLOT, CIRCLE, FILL, LINE, COLOUR, INK, PAPER, ecc.) che permettono di eseguire con una certa facilità operazioni di traslazione e rotazione di una figura sullo schermo. Tuttavia in alcuni casi è necessario ricorrere all'uso di funzioni trigonometriche quali: seno, coseno e tangente. Fortunatamente, il BASIC è fornito di una buona gamma di queste funzioni (SIN, COS, TAN, ecc.). Quando invece si affrontano problemi statistici, il BASIC purtroppo mostra notevoli limiti. Quasi tutte le versioni di questo linguaggio mancano di funzioni statistiche per la manipolazione dei dati. Se vogliamo scrivere un programma capace di predire chi vincerà le prossime elezioni o quanti ragazzi con gli occhi azzurri riusciranno a superare un particolare esame, allora dovremo sviluppare da soli le necessarie funzioni statistiche. Se scriviamo programmi di gioco per imparare a battere a macchina nei quali i tempi di risposta dell'utente sono importanti, ancora una volta troviamo grossi limiti in molte versioni BASIC, che mancano spesso di funzioni di temporizzazione affidabili. Nell'arco di più lezioni affronteremo gli aspetti della matematica del BASIC ed approfondiremo tre argomenti principali: la trigonometria, la statistica e le procedure di temporizzazione. Gli studenti delle scuole secondarie spesso si domandano quale rapporto esista tra la trigonometria ed il mondo reale. La risposta a questo quesito è che la trigonometria rappresenta l'anello di congiunzione tra la geometria euclidea, che si occupa della manipola-



zione di punti, linee e curve, e l'algebra, che consente di giungere a soluzioni matematiche trattando variabili con valori incogniti. Prendiamo ad esempio la figura geometrica della parabola. Questa è una curva dalle molte utili proprietà, che si possono evidenziare mediante semplici strumenti come un goniometro, una riga, un po' di carta millimetrata ed una matita. Ancor più utile è capire che questa curva può essere rappresentata con la formula algebrica: $y = x^2$, che ci permette di calcolare i valori di ciascun punto della curva senza doverla effettivamente disegnare.
Il computer si presta particolarmente bene alla risoluzione di problemi che implicano un'elaborazione algebrica: appare quindi chiara l'estrema utilità delle funzioni trigonometriche.
L'obiettivo di questa prima lezione sull'argomento è quello di offrire una breve sintesi dei concetti trigonometrici e mostrare come essi possano essere facilmente codificati in programmi BASIC. Iniziamo dalle funzioni trigonometriche fondamentali: il seno ed il coseno.

Calcolo dei rapporti

Il coseno ed il seno sono due funzioni che rappresentano il rapporto esistente tra i lati di un triangolo rettangolo. Inoltre, essi sono anche in diretto rapporto con la circonferenza. Ogni triangolo rettangolo può essere "inscritto" in una circonferenza, che viene chiamata "unitaria", qualora essa possieda un raggio di una "unità". La sua reale misura non ha importanza perché ciò che conta è il rapporto. L'illustrazione mostra una linea che è stata ruotata di 35 gradi. La posizione di partenza, dalla quale viene calcolata la rotazione coincide, per convenzione, con l'asse orizzontale e la rotazione avviene in senso antiorario. L'asse orizzontale è chiamata "asse x" e l'angolo di rotazione "Θ", che viene scritto usando la lettera greca Θ. Se tracciamo una linea dalla circonferenza all'asse x con angolo Θ rispetto a quest'ultimo, otteniamo un triangolo rettangolo.
Il cos Θ (cos è l'abbreviazione di coseno) è definito come il rapporto tra la lunghezza del cateto "adiacente" del triangolo (il lato che giace sull'asse x) e quella dell'ipotenusa (il raggio della circonferenza unitaria). Consideriamo una circonferenza con raggio di 15 mm. e misuriamo con precisione il cateto adiacente del triangolo inscritto in essa: la sua lunghezza sarà di 12,28728 mm. Dividendo questo valore per 15 otterremo 0,819152, che equivale al coseno di 35 gradi. Possedendo una calcolatrice dotata del tasto COS (potremmo usare in alternativa una tavola dei coseni, se ne siamo capaci), si provi ad inserire 35 COS: il risultato dovrebbe essere 0,819152044. Questo rapporto non cambia qualunque sia la dimensione della circonferenza unitaria e quella del triangolo rettangolo in essa inscritto.
Anche se il raggio della circonferenza fosse di un chilometro o di un anno luce, il cateto del triangolo che giace sull'asse x sarebbe sempre approssimativamente lo 0,82 della misura del raggio.
Possiamo anche tracciare altri valori di Θ nel cerchio unitario in figura. È possibile vedere che per ogni valore di Θ, il valore di COS Θ non sarà mai maggiore di 1 né minore di 0. Per valori di Θ maggiori di 90 gradi, COS Θ avrà, tuttavia, valore negativo. Ciò accade perché COS Θ implica il valore della coordinata x (il punto sull'asse x che corrisponde al punto di intersezione tra l'ipotenusa e la circonferenza unitaria). Una convenzione matematica definisce che al punto di origine della rotazione sull'asse x venga assegnato il valore 0. Qualunque punto alla sua sinistra ha valore negativo. Per la stessa ragione il coseno degli angoli tra 180 e 270 gradi sarà anch'esso negativo, mentre quello degli angoli tra 270 e 360 sarà di nuovo positivo.
La funzione seno di un angolo è assai simile al coseno, ma si riferisce ai valori dell'asse y (l'asse verticale). Se Θ è 0, il cateto adiacente del triangolo avrà lunghezza uguale all'ipotenusa. La coordinata di P sull'asse x sarà sempre 1 (perché 1/1 = 1), ma la coordinata sull'asse y sarà 0. Per tutti i valori di Θ fino a 90 gradi SIN Θ (sin è l'abbreviazione di seno) è compreso tra 0 e 1.
Nel secondo quadrante del cerchio, SIN Θ è ancora positivo, ma decrescerà da 1 a 0, con l'avvicinarsi di Θ ai 180 gradi. Tutti i valori di Θ compresi tra 180 e 360 gradi (escluso quest'ultimo), sono negativi.



ONDA... SU ONDA

La comune configurazione dell'onda sinusoidale si ottiene facendo un grafico della funzione seno per un giro completo della circonferenza. Sull'asse x vengono rappresentati gli angoli compresi tra 0 e 360 gradi mentre l'asse y rappresenta l'insieme dei valori del seno per questi angoli. Da notare che tutti i valori del seno sono compresi tra +1 e −1. Le quattro sezioni del grafico corrispondono ai quadranti visti nella figura precedente; il seno è positivo (colore blu) nei primi due quadranti e negativo (colore verde) per gli altri due. Il grafico del coseno dà risultati simili. La curva del coseno presenta in realtà la stessa forma di quella del seno, ma risulta spostata (o più precisamente "sfasata") di 90 gradi lungo l'asse x

Linea di partenza

In ASSEMBLY esistono delle pseudo-istruzioni, che facilitano il lavoro di programmazione

Dove localizzare le istruzioni in codice macchina

BBC MICRO

A livello comandi, impartire: PRINT.~PAGE. Questo comando definisce l'indirizzo esadecimale di partenza del nostro spazio riservato. Adesso inserire: PAGE = PAGE+N, dove N rappresenta il numero decimale di byte che si vogliono riservare



Nella precedente lezione di questa serie, abbiamo scritto un semplice programma in codice macchina per sommare due numeri nell'accumulatore ed immagazzinare il risultato in memoria. Nella stesura del programma, abbiamo affrontato alcuni punti di notevole importanza per la programmazione in codice macchina. Supponiamo che esso venga "caricato" in memoria a partire dall'indirizzo $000 e che il risultato contenuto nell'accumulatore debba essere registrato all'indirizzo $0009 (N.B. Questo è solo un esempio teorico e qualsiasi tentativo reale di utilizzare queste locazioni produrrà un'irrimediabile interruzione del programma). Le due versioni del programma sono:

Indirizzo di Locazione	Codice Macchina	Linguaggio ASSEMBLY
Z80		
0000	A7	AND A
0001	3E 42	LD A,$42
0003	CE 07	ADC A,$07
0005	32 09 00	LD BYTE1,A
0008	C9	RET
6502		
0000	18	CLC
0001	A9 42	LDA #$42
0003	69 07	ADC #$07
0005	8D 09 00	STA BYTE1
0008	60	RTS

Si noti che nella quarta istruzione (che permette di immagazzinare il contenuto dell'accumulatore nell'indirizzo $0009 in entrambi i programmi, non è stato usato un valore numerico per l'indirizzo di destinazione, ma, al contrario, un nome "simbolico": BYTE1. Nella versione in codice macchina vediamo però che l'op-code per il trasferimento del contenuto dell'accumulatore è seguito dal valore 09 00, che rappresenta, nel formato "basso-alto", i due byte dell'indirizzo $0009.

È questo un altro aspetto del processo di traduzione (o assemblaggio) dal linguaggio ASSEMBLY al codice macchina. Solitamente, si usano codici mnemonici di comando di facile interpretazione (ad esempio, STA e RET) al posto di codici esadecimali del tipo 8D e C9, per rendere più comoda l'interpretazione dei listati.

Analogamente, viene fatto largo uso di parole simboliche come BYTE1 *al posto di indirizzi esadecimali o valori numerici.*

Questa procedura assomiglia molto all'inizializzazione delle variabili in un programma BASIC. Il motivo è lo stesso in entrambi i casi: il programma risulta più leggibile, si riducono le possibilità d'errore ed il programma diventa più facile da manipolare. Per esempio, cambiando l'istruzione che assegna il valore costante alla variabile, all'inizio di un programma, il nuovo valore viene automaticamente utilizzato in tutte le restanti operazioni.

Questo è un concetto di facile comprensione quando parliamo di programmazione BASIC, ma nel nostro programma ASSEMBLY quale sarà l'istruzione equivalente al comando BASIC LET BYTE1 = $0009?

Al momento, dato che stiamo "assemblando" il programma manualmente, non c'è alcun comando di questo tipo.

Se, invece, avessimo la possibilità di usare un programma assemblatore (per la codifica automatica), allora potremmo inserire una speciale istruzione di assegnazione all'inizio del programma (presentiamo qui la versione Z80 del programma, anche se queste direttive di assemblaggio vanno bene per entrambi i processori):

Z80			
0000		BYTE1	EQU $0009
0000	A7		AND A
0001	3E 42		LD A,$42
0003	CE 07		ADC A,$07
0005	32 09 00		LD BYTE1,A
0008	C9		RET

Il nome BYTE1 viene allora inserito in una propria colonna, conosciuta come campo di identificazione (label field, in inglese), di cui parleremo più avanti. Nel campo op-code viene inserito un nuovo mnemonico EQU, che significa "equivale a", e nel campo operandi viene inserito il valore da assegnare a BYTE1 (in questo caso $0009).

Va sottolineato che, sebbene EQU appaia nel campo op-code ed assomigli ad un codice operativo, essa non fa parte dei codici operativi del linguaggio ASSEMBLY e non compare né nel set di istruzioni dello Z80 né in quello del 6502.

Mnemonici di questo tipo vengono chiamati "pseudo op-code" o "direttive di assemblaggio". EQU informa l'assemblatore che ad ogni occorrenza di BYTE1 deve essere sostituito il

valore specificato dopo la stessa EQU ($0009 nel nostro esempio).
L'output di un assemblatore è costituito, solitamente, da un listato ASSEMBLY completo e da un programma in codice macchina composto semplicemente da una serie di byte esadecimali. Il codice macchina può essere salvato sotto forma di file ed utilizzato successivamente, oppure caricato subito in memoria per l'esecuzione.
L'assemblatore, oltre alla codifica del programma, può eseguire altre utili operazioni, quale, ad esempio, inserire gli indirizzi di locazione in ciascuna linea del programma. Lo pseudo-op ORG provvede a questa operazione e viene aggiunto al programma in questo modo:

Z80			
0000			ORG $A000
A000		BYTE1	EQU $0009
A000	A7		AND A
A001	3E 42		LD A,$42
A003	CE 07		ADC A,$07
A005	32 09 00		LD BYTE1,A
A008	C9		RET

Si noti che l'indirizzo di locazione inserito nella prima linea del programma è $0000 mentre l'indirizzo della linea successiva è $A000, ciò riflette l'effetto dell'istruzione ORG. Notiamo inoltre che non compaiono byte in codice macchina sulle linee che contengono pseudo-op, in quanto questi ultimi non fanno parte dell'insieme vero e proprio di istruzioni e non possono venire tradotti in codice macchina. Dato che gli pseudo-op sono caratteristici dell'assemblatore e non fanno parte della gamma di istruzioni della CPU essi potranno differire da un assemblatore all'altro. EQU ad esempio, viene spesso sostituito da "=" e ORG da ".=". Il risultato finale, tuttavia, non cambia e noi continueremo ad usare EQU e ORG nella nostra trattazione.
Fin da principio, abbiamo usato un altro pseudo-op: il simbolo "$", è questa una direttiva che permette all'assemblatore di interpretare il valore che segue come un numero esadecimale. Similmente, il simbolo "#", introdotto nella scorsa lezione, identifica i "dati immediati": ciò significa che il valore seguente rappresenta una quantità assoluta e non un puntatore o un simbolo.
Estendendo questo concetto potremmo in pratica considerare lo stesso linguaggio ASSEMBLY come una serie di pseudo-op. Al limite, nulla ci impedisce di inventare mnemonici personalizzati per la serie di istruzioni in codice macchina, purché venga conservata una corrispondenza "uno a uno" con la serie stessa.
In questo corso, tuttavia, continueremo ad usare gli mnemonici già visti, cioè quelli dei linguaggi ASSEMBLY concepiti dai fabbricanti di microprocessori. Tralasciamo per il momento gli pseudo-op e torniamo ad esaminare altri punti significativi del nostro programma. In particolare confrontiamo la traduzione delle istruzioni LDA ed LD A data qui di seguito, con quella dei programmi precedenti. Prima abbiamo scritto:

AD ?? ??	LDA $????	(6502)
3A ?? ??	LD A,($????)	(Z80)

che significa "carica nell'accumulatore il byte con indirizzo ????". La traduzione in codice macchina dell'op-code "carica l'accumulatore", è $AD per il 6502 e $3A per lo Z80. Confrontiamo questo con la seconda linea dell'attuale programma, nella quale viene usata l'istruzione "carica l'accumulatore con il valore immediato $42".
In questo caso gli op-code sono $A9 per il 6502 e $3E per lo Z80. Perché sono diversi? Il motivo è semplice: anche se stiamo eseguendo lo stesso tipo di operazione in entrambi i casi (il trasferimento di un dato nell'accumulatore), la provenienza del dato non è la stessa e la CPU considera queste come istruzioni diverse tra loro. Pertanto, sono necessari differenti op-code.
Nella prima versione, il dato veniva caricato a partire da un byte di indirizzo noto, non sapevamo però nulla del contenuto di questo byte. La CPU doveva semplicemente copiare il suo contenuto nell'accumulatore. I tre byte in codice macchina, AD $$ $$ e 3A $$ $$, venivano decodificati dalla CPU nel seguente modo: "interpretare i due byte che seguono questo op-code come l'indirizzo assoluto di origine dei byte".
Nella seconda versione, il dato che deve essere caricato nell'accumulatore si trova effettivamente nel byte che segue l'op-code perciò i due byte di codice macchina, A9 42 e 3E 42, vengono così decodificati dalla CPU: "interpreta il byte che segue questo op-code come l'origine del dato". Qualcosa nell'op-code (A9 o 3E) ordina alla CPU di caricare nell'accumulatore il byte seguente. Poiché il contatore di programma contiene sempre l'indirizzo dell'istruzione successiva, la CPU può calcolare l'indirizzo di origine del byte e quindi eseguire l'operazione: "carica nell'accumulatore il byte contenuto nell'indirizzo successivo all'op-code".
Tutto ciò rafforza l'affermazione che le operazioni eseguite dalla CPU seguono di solito procedure molto elementari. Un intero gruppo delle sue operazioni (circa 1/5 del repertorio completo) è composto da operazioni per il trasferimento di dati da un byte indirizzato ad uno dei suo registri interni. Tutte queste operazioni "primitive" implicano un unico lavoro: trasferire dati dalla memoria ai registri della CPU, e ciò che permette di distinguere un'operazione dall'altra è il tipo di formato con cui viene presentato l'indirizzo del byte di origine.
Addentrarsi in questa miriade di operazioni interne della CPU può creare, inizialmente, un po' di confusione, ma questo difficile impegno ci consentirà poi di armonizzare le attuali conoscenze con quelle che acquisiremo in seguito.

SPECTRUM

A livello comandi, impartire:

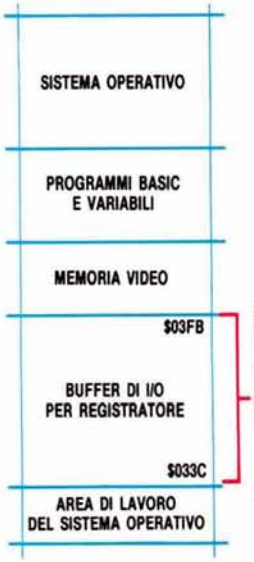
```
LET RTOP = PEEK(23730) + 256 *
  PEEK(23731)
LET RTOP = RTOP—N
PRINT "RTOP = ";RTOP
```

dove N è il numero di byte che vogliamo riservare per il nostro programma. L'area riservata inizia a 1+RTOP



COMMODORE 64

Usare il buffer del registratore (da $033C a $03FB)



Attenzione!

Si devono mettere a punto questi puntatori di memoria subito dopo aver acceso il computer, quando cioè non ci sono ancora programmi BASIC in memoria

ESERCIZIO IN ASSEMBLY

1) Nel riquadro qui di fianco, forniamo una versione Assembly di un semplice programma. Si traduca il programma in codice macchina, determinando gli indirizzi dei vari codici

2) Quale istruzione manca nel programma?

3. Quale effetto ha il programma sui registri della CPU e sulla RAM?

4) Cosa significa la dizione "dato immediato"? Quali altri tipi di dati esistono?

5) Nel programma, BYTE1 rappresenta un indirizzo. A quale pagina della RAM si riferisce?

N.B. I valori usati nel programma sono puramente indicativi. Per poter usare il programma, gli indirizzi devono essere scelti accuratamente, in base al tipo di computer

INDIRIZZO DI LOCAZIONE	CODICE MACCHINA	LINGUAGGIO ASSEMBLY		
		6502		
		START	EQU	$A000
		BYTE1	EQU	$45
		BYTE2	EQU	$38
			ORG	START
			LDA	#BYTE1
			CLC	
			ADC	#BYTE1
			STA	BYTE1
			ADC	#BYTE2
			STA	BYTE2
		Z80		
		START	EQU	$A000
		BYTE1	EQU	$45
		BYTE2	EQU	$38
			ORG	START
			LD	ABYTE1
			AND	A
			ADC	A,BYTE1
			LD	(BYTE1),A
			ADC	A,BYTE2
			LD	(BYTE2),A

FUNZIONAMENTO DELLE ISTRUZIONI IN CODICE MACCHINA



Istruzione 1: CARICA (LOAD) l'accumulatore col contenuto dell'INDIRIZZO 1
Istruzione Z80: LD A,(INDIR1)
Istruzione 6502: LDA INDIR1

Istruzione 2: SOMMA (ADD) col contenuto di INDIRIZZO 2
Istruzione Z80: ADC A, (INDIR1)
Istruzione 6502: ADC INDIR1

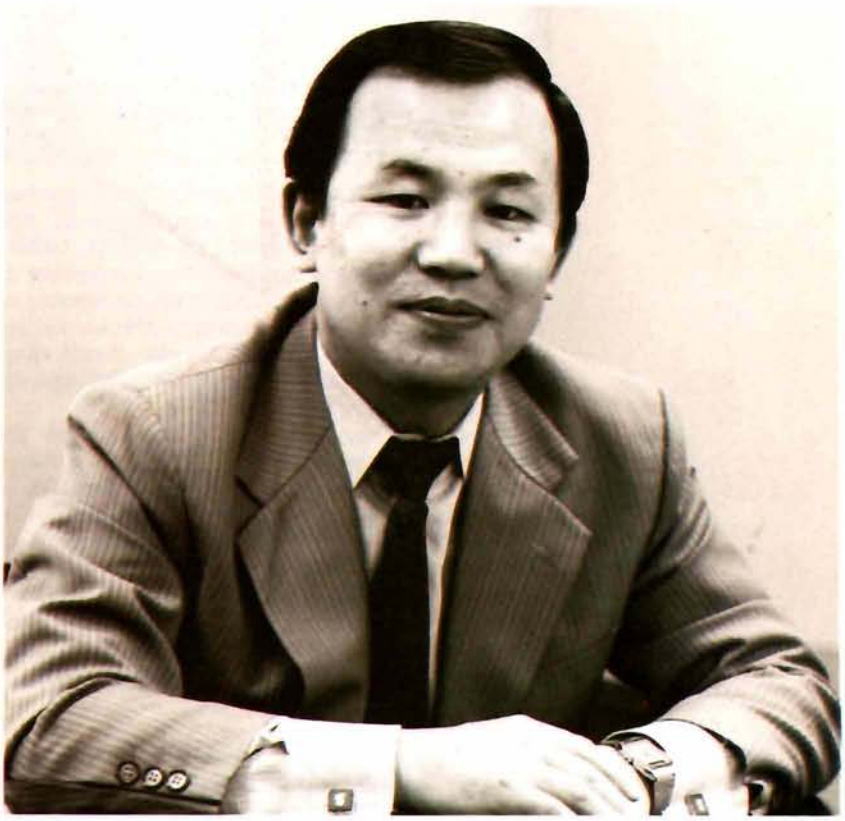
Istruzione 3: DEPOSITA (STORE) il contenuto dell'accumulatore in INDIRIZZO 2
Istruzione Z80: LD (INDIR2),A
Istruzione 6502: STA INDIR2

L'effetto di istruzioni per il trasferimento di dati, quali LDA INDIR1 o LD (INDIR2),A, è sempre quello di copiare il contenuto della locazione "sorgente" nella locazione "destinataria". Pertanto, il precedente contenuto di quest'ultima viene perduto, mentre il contenuto della locazione sorgente rimane inalterato

La giapponese SORD

La SORD non è un colosso, ma i suoi apparecchi sono spesso all'avanguardia

La SORD deriva il suo originale nome da una combinazione delle parole SOftware e haRDware. È un nome che ben si adatta a questa azienda che ha sempre mostrato attenzione agli sviluppi di ambedue questi settori.
Gli inizi sono stati piuttosto tormentati. Il suo presidente, Takayoshii Shiina, lasciò l'università per lavorare con la Rikei Industries, una fabbrica di non grande successo, negli anni '60. Qui egli si interessò alla riorganizzazione della politica di marketing della compagnia. L'assunzione di Shiina fu un evento quasi unico nella storia del Giappone (dove la gente lavora tutta la vita per gli stessi datori di lavoro): infatti, durante il colloquio di assunzione, Shiina dichiarò che avrebbe svolto quel lavoro solo per pochi anni perché aveva l'intenzione di dar vita ad una propria attività commerciale.
Nel 1970, con un suo amico, era pronto a fondare una nuova società: nacque così la SORD con un capitale iniziale di 65.000 jen, pari a circa 5 milioni di lire odierni. Shiina tuttavia continuò a lavorare per la Rikei sino alla fine del 1970. Le prime attività della SORD furono alcuni contratti per lavori di programmazione e lo sviluppo di un tester logico di basso costo.
Nel 1973, la Sord dette inizio alla produzione di apparecchi, e nel 1974 comparvero le prime unità a dischetti che utilizzavano un'interfaccia progettata dalla SORD. A questa presto seguì l'SMP-80/20, uno dei primi computer giapponesi basati sull'Intel 8080.
L'SMP-80/20 è stato un prodotto di grande successo. Ma Shiina aveva progetti molto più ambiziosi e nel 1976 si consociò alla Toppan, una delle più importanti industrie giapponesi di stampaggio, per un 20% del capitale. Questa iniezione di denaro permise alla SORD di sviluppare un considerevole supporto software da associare al suo crescente numero di prodotti informatici, e nel 1981 la Compagnia lanciò sul mercato il PIPS.
Il PIPS era un pacchetto software molto progredito per quel tempo ed uno dei primi esempi al mondo di software integrato: il suo contributo è stato essenziale per il consolidamento della posizione della SORD nel mercato dei computer. Il PIPS fonde le funzioni di lavagna elettronica, word processor e database in un pacchetto di programmi che anche i principianti possono imparare ad adoperare in poche ore.
Il suo grandissimo successo in Giappone può essere capito solo tenendo conto delle condizioni del mercato dei computer domestici di quel periodo. In Giappone i computer venivano venduti senza software di supporto, ed il software applicativo per attività commerciali, comunemente adottato in Europa e negli Stati Uniti, era sconosciuto.
Le ragioni di questo ritardo erano due: innanzi tutto i giapponesi preferiscono non importare dall'estero ciò che possono produrre da soli (quindi solo minime quantità di software americano avevano raggiunto il Giappone); inoltre, essi sono poco portati per le lingue, e le difficoltà con i manuali in inglese sarebbero state enormi. La scarsità di software applicativo del mercato nipponico ha quindi aperto vasti spazi di mercato alla SORD. Essa cominciò a sviluppare la serie dei computer M200 basata sullo Z80 ed in seguito la serie M23. Con la successiva serie M343 fu possibile soddisfare le necessità di multiutenza. La serie M5 fu prodotta per venire incontro al mercato dei videogiochi casalinghi ed oggi viene commercializzata la serie M68 (che incorpora i processori Z80 e Motorola 68000), il piccolo computer commerciale M243 e l'M285, un apparecchio a 32 bit, che utilizza il software VAX-11 per applicazioni CAD (progettazione computerizzata).
Nessun'altra compagnia al mondo presenta



per concessione della SORD

Takayoshü Shüna



PIPS

Il PIPS è l'unico pacchetto della SORD per attività commerciali. È un programma di impiego generico, che permette di gestire la maggior parte delle procedure commerciali: dalla contabilità alla presentazione grafica. Un sintomo del modo tutto giapponese di affrontare la commercializzazione dei prodotti informatici è che il PIPS lo si può ottenere gratuitamente, ma solo su alcuni apparecchi SORD

una così vasta gamma di prodotti sul mercato, ciascuno dei quali è corredato di un proprio software specifico. La SORD, tuttavia, ha una grossa limitazione: considera i propri prodotti come un sistema hardware e software a sé stante e quindi chiuso in se stesso. Per questo motivo i suoi utenti non hanno la possibilità di adottare altri tipi di software sui loro apparecchi.

Ultimamente la compagnia ha cercato di ovviare a questo problema adottando il sistema operativo SB-80 come opzione per la sua serie M23 basata sullo Z80. L'SB-80 equivale al sistema operativo CP/M 2.2 della Digital Research e permette per la prima volta di utilizzare il software CP/M sui computer SORD. Un secondo passo della SORD in questa direzione è stato l'adozione del Sistema-p della UCSD (Università di San Diego in California).

Agli utenti che limitano i propri interessi al software della SORD, vengono offerte alcune potenti versioni di BASIC, il PIPS ed un word processor simile al Wang.

Oltre ai già citati CP/M e Sistema-p, la SORD ha seguito il cammino già battuto da altri fabbricanti di micro come la DEC, mettendo a disposizione dei suoi clienti sistemi basati in larga misura su software prodotto in casa. Nel frattempo, i maggiori produttori di computer hanno capito che la standardizzazione e la compatibilità fra prodotti di Case diverse stanno alla base dei futuri sviluppi di mercato. Purtroppo la SORD si sta adeguando molto lentamente a questa nuova impostazione.

Essa continua a commercializzare sistemi hardware innovativi, ma gli esperti affermano che la loro documentazione è piuttosto scadente. Negli ultimi due anni la compagnia ha compiuto un notevole sforzo per corredare tanto il PIPS che il suo word processor con migliore documentazione. Purtroppo anche l'hardware presenta molti limiti da questo punto di vista e poche sono le Case di software che producono programmi in grado di funzionare sugli apparecchi della SORD.

I prossimi anni saranno probabilmente molto critici per la SORD, che dovrà affrontare la concorrenza di industrie giapponesi di maggiori dimensioni, ma anche della IBM. Shiina ripone grande fiducia nelle industrie di piccole dimensioni e nella intraprendenza individuale: in effetti la sua compagnia ha riscosso un notevole successo anche in campo internazionale. Riuscirà però a mantenere la posizione raggiunta nel competitivo mondo dei computer anche nel prossimo futuro?

Modelli SORD

Gli apparecchi più noti della SORD sono l'home computer M5 ed il portatile commerciale M23P. Quest'ultimo ha introdotto nuovi standard per gli apparecchi portatili come i dischetti Sony ed un display a cristalli liquidi (LCD) da 80 colonne



IAN McKINNELL

Sopravviva il migliore

La gamma di microprocessori disponibili è vastissima, ma sono pochi i chip ad aver veramente successo

I computer basati su chip sono stati inventati quasi per caso. Infatti, nel 1972, la Datapoint richiese alla Intel, una ditta produttrice di circuiti integrati, di sviluppare un chip che fosse in grado di sostituire la gran quantità di integrati TTL (Logica Transistor-Transistor) utilizzati nei terminali dei computer di allora. Il prodotto realizzato dalla Intel venne chiamato 8008: era in grado di elaborare dati di otto bit alla volta e sarebbe stato un "sostituto logico" ideale per terminali della Datapoint, se non avesse sofferto di un'eccessiva lentezza. La Datapoint decise, quindi, di non adottarlo, ma le potenzialità dell'8008, come CPU per elaboratori per impieghi generici, furono ben presto messe in luce da progettisti e da amatori, che contribuirono in modo significativo alla nascita del computer da ufficio.

Tuttavia, i limiti dell'8008, in termini di velocità e potenza d'elaborazione, divennero ben presto palesi e la Intel cominciò a progettare un chip più avanzato. Nacque così l'8080, che rapidamente divenne il nuovo dominatore del mercato.

Quasi contemporaneamente, anche la Motorola lanciò un processore ad 8 bit chiamato 6800 e poco dopo un vero colpo di fortuna aprì la strada al prodigioso successo di un terzo chip, lo Z80. Nel 1974 Gary Kildall, oggi presidente della Digital Research, sviluppò per la Intel un sistema operativo chiamato CP/M, che permetteva ai computer basati sull'8080 di gestire le unità a dischetti Shugart di recente introduzione. Il sistema operativo non venne adottato dalla Intel, che giudicava il software esistente più che sufficiente per i grossi sistemi di elaborazione di quel periodo.

Tuttavia, i computer di minori dimensioni stavano divenendo sempre più popolari ed il CP/M rendeva molto più agevole la gestione degli archivi su questi apparecchi. Questo fatto ha permesso all'8080 di dominare il mercato dei micro per molti anni. Furono fatti vari tentativi per dotare il 6800 di un sistema operativo per unità a dischi di pari qualità, ma il successo conquistato dall'8080 non subì alcuna flessione.

Il fulmineo sviluppo del mercato dei microcomputer scatenò una frenetica concorrenza tra i fabbricanti di chip che dovettero, però, fare i conti con la riluttanza del pubblico ad accettare un qualsiasi nuovo prodotto che non fosse in grado di offrire significativi vantaggi. Inoltre, gli investimenti profusi nella progettazione hardware e nella produzione software ostacolarono l'adozione di un qualsiasi nuovo microprocessore non compatibile coi precedenti.



Una scelta di base
Per molti home computer è stato adottato un microprocessore 6502 (come per il Micro BBC) o uno Z80 (per lo Spectrum). Il Dragon, è uno dei pochi ad utilizzare un chip diverso, il 6809

Durante lo sviluppo dello Z80, la Zilog, costituita da un gruppo di progettisti che aveva in precedenza lavorato sull'8080 per la Intel, intuì che il set di istruzioni poteva essere ampliato. In altre parole, non erano ancora state sfruttate tutte le possibili combinazioni di uno e zero che l'8080 era in grado di interpretare come istruzioni. Utilizzando combinazioni binarie non usate dal chip Intel, la Zilog progettò un microprocessore in grado di operare come l'8080 (e quindi sfruttare tutto il software esistente per questo chip), offrendo, al tempo stesso, prestazioni di gran lunga superiori.

Oltre a questa innovazione, la Zilog ottenne anche un altro importante risultato commerciale: il chip della Intel faceva affidamento tanto su un chip di temporizzazione che su un chip di controllo del sistema. Ebbene, lo staff della Zilog riuscì a fondere in un singolo chip tutta la logica necessaria ad un compu-

GENERAZIONI DI CHIP

I microprocessori si sono evoluti a partire da due sorgenti principali: alcuni derivano dai primi microprocessori della Intel, altri dal chip 6800 della Motorola. Questo schema mostra le modalità del loro sviluppo ed alcune delle macchine su cui sono stati usati. Molti dei chip meno noti sono stati adottati da apparecchi di minor successo. La Apple III è forse l'unico apparecchio per usi commerciali che utilizza il 6502. L'Olivetti M20 è l'unico ad impiegare lo Z8000. In entrambi i casi, l'insolita scelta del microprocessore e la conseguente mancanza di software ha limitato il successo della macchina. Alcune macchine di grande successo, come l'IBM PC, hanno grandemente contribuito alla popolarità dei chip da esse adottati

INTEL

4004

8008

Progettato per sostituire un gran numero di circuiti integrati TTL, il 4004 era un chip molto semplice in grado di gestire dati solo in gruppi di 4 bit. La Intel ben presto passò ad un processore ad 8 bit producendo l'8008. Gli hobbysti e gli ingegneri ne apprezzarono le potenzialità e cominciarono ad adottarlo per i primi microcomputer

'71

'72

'73

'74

'75

'76

'77

'78

'79

'80

'81

'82

'83

'84

MOTOROLA

6800

È il principale concorrente dell'8080, con pari prestazioni, ma con una filosofia di progettazione completamente diversa. Da questo scontro si sono sviluppate due scuole: l'una ha preferito l'Intel 8080, l'altra ha optato per il modo di operare del Motorola 6800

6809

Il 6809, probabilmente il più potente chip a 8 bit mai realizzato, è stato sviluppato dal 6800. Tuttavia, la sua tardiva comparsa ne ha limitato il successo ed è quindi stato adottato da poche macchine



DRAGON 32

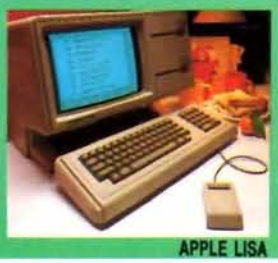
68000

Il successo di questo famosissimo chip a 16 bit della Motorola è stato mitigato dalla scarsa disponibilità di software a basso costo e dalla notevole diffusione dell'8088. Tuttavia, la Sinclair ha scelto una sua versione ridotta, il 68008, per il microcomputer QL

68008



SINCLAIR QL



APPLE LISA

MOS TECHNOLOGY

6502



COMMODORE PET

La MOS Technology ha sviluppato un proprio chip a 8 bit che, sebbene incompatibile col 6800, è in gran parte un suo discendente. Il suo basso prezzo lo ha reso molto apprezzato dagli hobbysti e dai progettisti: infatti è stato utilizzato su micro di grande successo della prima generazione come il PET e l'Apple. Ancora oggi è un prodotto molto popolare sui micro casalinghi ed è stato usato su apparecchi come l'Oric ed il BBC



APPLE III



IBM PC

8080

Con lo sviluppo del sistema operativo CP/M, questo è stato il primo chip usato per produrre home computer ed apparecchi per ufficio. Una versione aggiornata dell'8080, realizzata dalla stessa Intel, è stato il chip 8085. Compatibile con l'8080, l'8085 è anche disponibile in una versione CMOS a bassa potenza adottata su portatili come il Modello 100 della Tandy e l'Olivetti M10

8085



TANDY TRS-80 MODELLO 100

8086

8088

L'8088 è una versione ridotta, che può usare chip di supporto di vecchio tipo: questa caratteristica ed il suo utilizzo sul Sirius e sull'IBM-PC lo hanno reso il più famoso chip a 16 bit. L'8086 è capace di maggiori prestazioni, ma è totalmente compatibile con l'8088 ed inizia soltanto oggi ad essere usato sulla maggior parte degli apparecchi a 16 bit

ZILOG

Z80

Un gruppo di progettisti lasciò la Intel per fondare la Zilog e produrre questa versione avanzata, ma totalmente compatibile, dell'8080



TANDY TRS-80 MODELLO 1

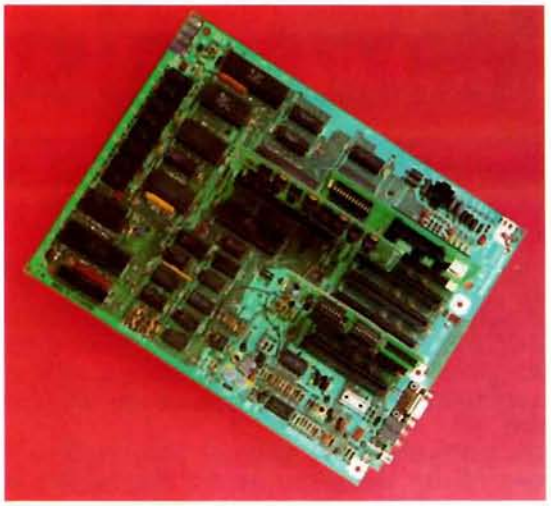
Z8000

Il primo chip a 16 bit della Zilog non ha riscosso grande popolarità sul mercato dei computer universali. Il secondo chip a 16 bit prodotto dalla Zilog è stato lo Z800 che la Zilog definisce totalmente compatibile con lo Z80 (e quindi anche con l'8080)



OLIVETTI M20

ter basato su microprocessore. Sebbene questo componente fosse relativamente costoso, la sua capacità di sostituire numerosi altri chip suscitò grande interesse nei fabbricanti di computer.
Il 6800, pur non avendo riscosso un successo pari a quello dell'8080, ha tuttavia goduto di una certa popolarità presso alcuni progettisti e programmatori. La Motorola, in seguito, sempre basandosi sul 6800, ha progettato un microprocessore ad 8 bit altamente sofisticato, chiamato 6809. Sfortunatamente per la Motorola, proprio quando il 6809 cominciava a dare buoni risultati di mercato, la MOS Technology lanciò una versione migliore del 6800, chiamata 6502. Questo è divenuto il più popolare tra tutti i microprocessori della serie 6500: tutti gli elementi di questa serie utilizzano la stessa gamma di istruzioni, ma si differenziano per potenza e prestazioni.
Il 6502 si fonda su una filosofia di progetto affine a quella del Motorola 6800, ma non è compatibile con quest'ultimo: né dal punto di vista hardware, né da quello software. Mentre invece lo Z80 incorpora l'intera gamma di istruzioni dell'8080 e, con appropriate modifiche, può sostituirlo all'interno di un sistema di elaborazione.
Il 6502 offre una gamma di istruzioni tali da appagare le necessità di qualsiasi programmatore abituato ad operare col 6800, ma con prestazioni assai migliori ed esigenze d'interfacciamento leggermente minori. Il software non è però compatibile, né si può operare una sostituzione tra i due chip, senza apportare sostanziali modifiche ai circuiti. In base a queste considerazioni, difficilmente il 6502 avrebbe potuto conservare a lungo la sua posizione di preminenza, se non fosse stato adottato dalla Apple per i suoi computer di immenso successo.
Quando l'Apple apparve sul mercato, i microcomputer da ufficio erano per la maggior parte concepiti sull'impiego del bus S-100. Ossia, era prevista una "scheda madre" per l'invio di segnali ad una serie di schede destinate a svolgere funzioni specifiche. Un sistema S-100 necessitava quindi di un dispositivo di alimentazione e delle seguenti schede: una scheda madre, una CPU, una scheda di memoria, una VDU e, verosimilmente, di una scheda per la stampante e di una separata per l'unità a dischetti. È facile intuire che un simile sistema è costoso e che sarebbe uscito sconfitto dal confronto con un sistema a scheda unica come quello dell'Apple.
Il prezzo relativamente basso di questo apparecchio era già un grande risultato, ma la principale innovazione introdotta da Steve Wozniak e dal suo gruppo di lavoro fu il progetto di un programma, chiamato VisiCalc. Questo programma divenne molto popolare tra gli uomini d'affari, che scoprirono di poterlo usare per sviluppare ipotesi finanziarie in modo più facile e rapido che a mano. Il successo riscosso dal VisiCalc ha permesso alla Apple di vendere un gran numero di computer e ciò ha fatto del 6502 una pietra miliare nel campo dei progetti per microprocessori. La Commodore ha adottato il 6502 nel PET e nei suoi successori.



Sempre meno chip
La realizzazione di circuiti integrati sempre più sofisticati ha permesso di ridurre il numero di chip necessari su una scheda di circuiti. Quando la Apple ha perfezionato l'Apple II, il numero di chip principali impiegati sulla nuova versione "IIe" era addirittura dimezzato

Il successo del 6502 si è ampliato ancor più quando la Acorn, in Gran Bretagna, ha prodotto il Micro BBC, anch'esso basato su questo chip. Inizialmente la BBC aveva deciso di adottare uno Z80, ma nessun fabbricante inglese riuscì a svilupparne una versione adeguata nei limiti di tempo stabiliti.
Mentre il chip 6502 diveniva il processore di maggior prestigio nella progettazione dei computer a 8 bit, cominciarono ad apparire sul mercato i primi computer a 16 bit. La Intel progettò per questi computer l'8088 e l'8086, mentre la Motorola produsse il 68000 e la Zilog lo Z8000. Tutti e tre i progetti a 16 bit avevano i loro pregi, ma nessuno era compatibile con i rispettivi predecessori a 8 bit. Fortunatamente per la Intel, la Microsoft e la Digital Research svilupparono in breve tempo dei sistemi operativi per l'8086 e per l'8088 (rispettivamente il CP/M 86 e l'MS-DOS), mentre la Zilog e la Motorola non ebbero la stessa fortuna. L'adozione da parte della IBM dell'8088 nel suo computer PC ha dato un ulteriore impulso al successo del chip della Intel.
La lotta per la conquista del mercato dei chip a 16 bit già si profila come una ripetizione delle battaglie commerciali per i chip a 8 bit. L'Intel 8086 e la sua versione ridotta 8088 stanno divenendo standard come già accadde per lo Z80 ed il 6502: ciò dipende dal supporto software fornito dai sistemi operativi MS-DOS e CP/M 86 e dalla loro adozione su micro di successo come l'IBM ed il Sirius. Il chip Z8000 della Zilog è stato usato per un solo microcomputer: l'M20 della Olivetti. Evidentemente, dopo notevoli sforzi per dotare la macchina di software adeguato, è stata fatta una palese "marcia indietro", realizzando una scheda a innesto con un chip 8086, per permettere all'apparecchio di utilizzare il software MS-DOS ed il CP/M 86.
La Zilog ha progettato un altro chip a 16 bit, ma in grado di sfruttare il software scritto per lo Z80. Si tratta dello Z800, che tuttavia pare non riscuotere molto successo in un mercato ancora dominato dallo Z80 e dal 6502.

Mezza misura

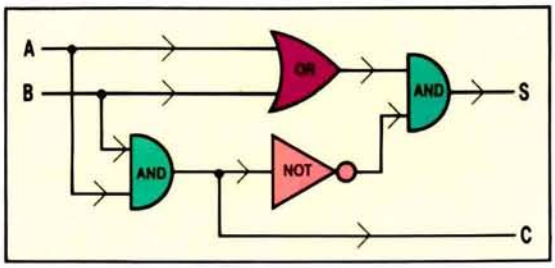
Realizziamo porte logiche con i circuiti integrati al posto dei transistor

Il semisommatore esaminato in "Le chiavi dell'Informatica" (vedere a pag. 513), è un piccolo dispositivo, costituito da una serie di semplici porte, usato per sommare tra loro due bit.
Un semisommatore possiede due ingressi (i due bit da sommare) ed ha due uscite: la somma ed il bit di riporto.
La tavola della verità che descrive questa situazione è la seguente:

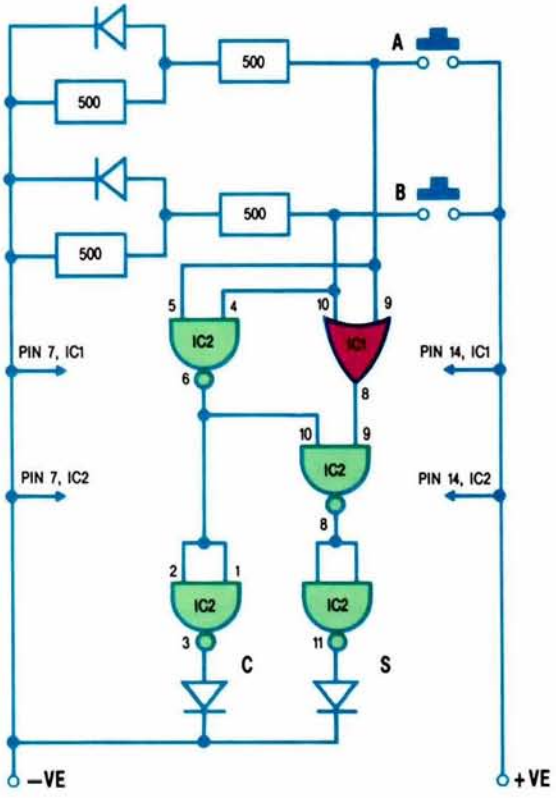
A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

L'uscita S (somma) rappresenta il risultato della somma dei due bit in ingresso: quando entrambi sono 1, la somma è 10 in binario. Questo risultato non può essere contenuto in un unico bit di output, è perciò necessario un secondo bit che prende il nome di bit di riporto (o Carry).
Nei computer ad otto bit, più che un semisommatore, occorre un circuito capace di sommare due parole di otto bit ed esso può essere costruito utilizzando 16 semisommatori. I primi due bit vengono sommati dal primo semisommatore ed il bit di somma costituisce il primo bit del risultato finale. Il bit di riporto viene sommato al risultato della seconda somma ed il bit di riporto proveniente da questa addizione, viene aggiunto alla terza somma e così via fino ad ottenere il risultato finale.
Anche per un semplice semisommatore sono necessari circa una decina di transistor, se usiamo i circuiti integrati nella scorsa lezione. Tuttavia, per pochi soldi, si possono acquistare circuiti presentati che contengono gruppi di quattro porte logiche AND, NAND, OR, NOR, in un unico contenitore. La realizzazione di un semisommatore risulta così facilitata.
Il circuito logico del semisommatore è mostrato più avanti: si tratta della forma più semplice del circuito e utilizza tre tipi di porte logiche: OR, AND e NOT. Poiché i circuiti logici che stiamo per usare contengono ciascuno un solo tipo di porta, il seguente circuito è stato semplificato in modo da usare il minor numero di porte di diverso tipo: quattro porte NAND ed un'unica porta OR. Il numero di circuiti integrati è così ridotto a due.
Questo circuito è più complesso di quello a singola porta, presentato a pag. 624, quindi dovremo prestare particolare attenzione alla disposizione dei componenti sulla scheda e ai collegamenti tra essi.
La costruzione del circuito si rivela più complessa che non il suo funzionamento, ma tale operazione sarebbe stata ancora più complessa se avessimo usato componenti "discreti", quali transistor e resistenze.
In effetti, nei computer vengono usati circuiti integrati simili ma molto più complessi internamente, onde elaborare una notevole quantità di ingressi ed uscite contemporaneamente.

CIRCUITO LOGICO



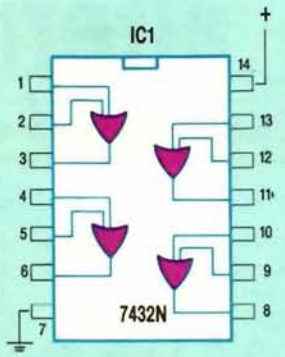
CIRCUITO ELETTRONICO

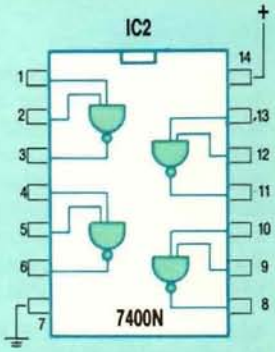


ELENCO DEI COMPONENTI

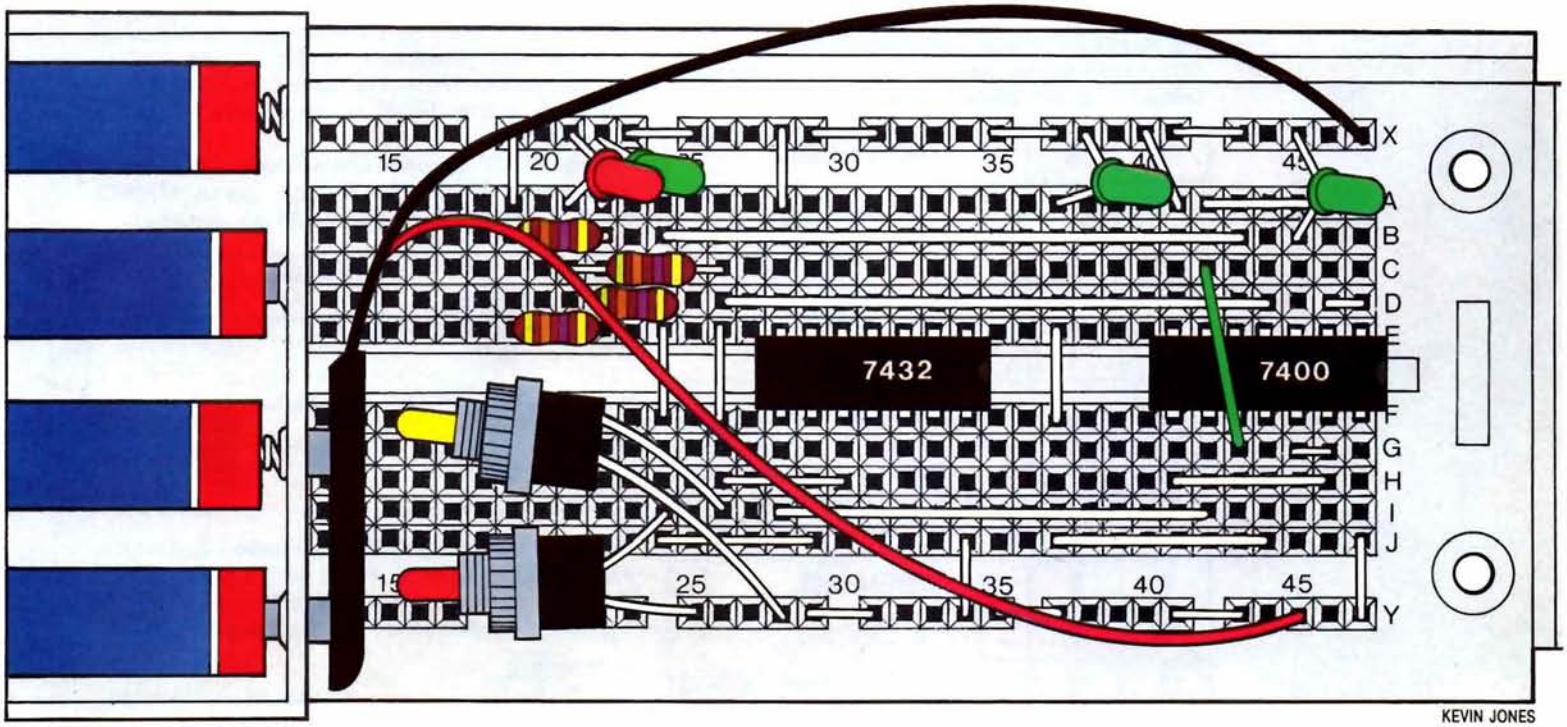
4 resistenze da 500 ohm, 1/4 di watt
4 LED
2 interruttori a pulsante normalmente aperti
1 circuito integrato 7400N
1 circuito integrato 7432N
4 pile HP7 o equivalenti da 1,5 volt
1 telaietto porta-pile
1 attacco per pile
1 basetta per esperimenti
Cavetti elettrici

PIEDINI DI CONNESSIONE





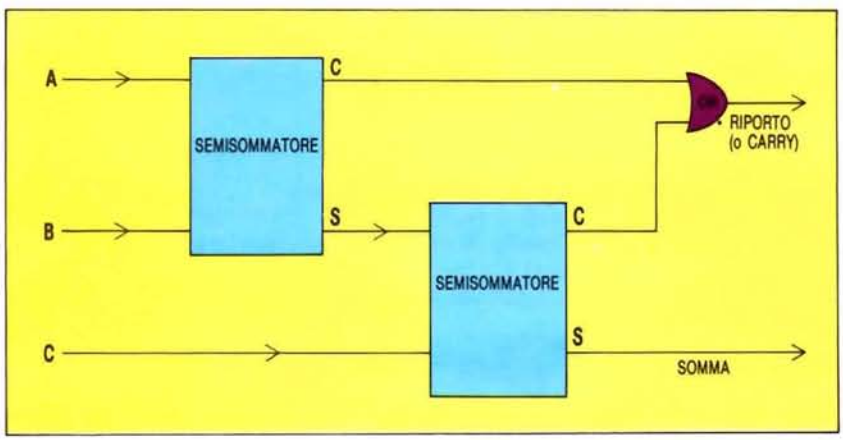
I due circuiti integrati qui impiegati sono chip TTL a 14 piedini. Il 7400N contiene 4 porte NAND, mentre il 7432N contiene 4 porte OR. Questo schema mostra la disposizione delle porte sul chip: occorre collegare tra loro i piedini (pin) in modo corretto. Ad un'estremità del chip c'è una piccola tacca di riferimento per la numerazione dei pin. La tensione d'alimentazione, necessaria al funzionamento dell'integrato, viene applicata ai piedini 7 e 14



KEVIN JONES

Assemblaggio sulla scheda

La fase successiva alla progettazione del circuito elettronico è l'assemblaggio dei componenti sulla basetta da esperimenti. Occorre montare i componenti e collegarli tra loro attenendosi il più possibile al circuito originale: quanto più il progetto sarà chiaro, tanto più facile sarà costruirlo e maggiori saranno le probabilità che esso funzioni al primo colpo



Sommatore completo

Come esercizio, cerchiamo di trasformare il nostro circuito semisommatore in un sommatore completo. Quest'ultimo circuito non solo somma insieme due bit, ma somma anche il bit di riporto proveniente dalla precedente "colonna". Il modo più semplice per creare un sommatore completo è collegare due semisommatori come mostrato in figura. Il segnale di somma proveniente dal primo semisommatore potrebbe essere collegato come ingresso al secondo. Il Carry generato dal primo sommatore potrebbe essere posto in OR con l'uscita di riporto del secondo sommatore, ottenendo il segnale da inviare al LED che rappresenta il riporto

Strade alternative

Due nuove porte logiche: la NAND e la NOR

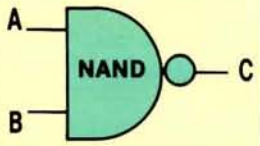
Dato che possiamo risolvere tutti i problemi logici usando porte AND, OR e NOT, che bisogno c'è di studiare altri tipi di porte? Usando queste nuove porte, da sole o assieme ad altre, potremo ridurre i costi del circuito semplificandone la struttura ed ottenendo soluzioni formalmente più eleganti. Tutti i problemi logici possono essere risolti utilizzando una delle seguenti tecniche:

a) porte AND, OR e NOT assieme
b) solo porte NAND
c) solo porte NOR
d) combinazione delle tecniche precedenti

Esaminiamo ora il funzionamento di queste due nuove porte, partendo da un esame delle relative tavole della verità:

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

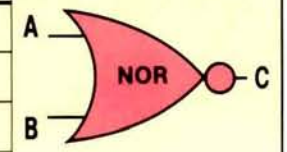
PORTA NAND



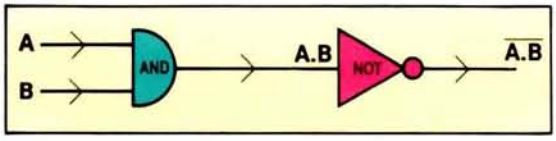
NAND è un'abbreviazione per Not AND e, confrontando questa tavola della verità con quella della porta AND (vedere a pag. 488), ci si accorgerà che nella colonna delle uscite tutti gli 1 sono stati cambiati in 0 e viceversa.

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

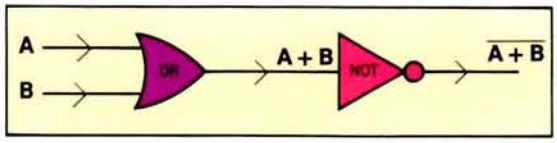
PORTA NOR



Analogamente, NOR sta per Not OR ed un confronto tra la colonna delle uscite di questa tavola e la tavola di una porta OR (vedere a pag. 488) mostra di nuovo che tutti gli 1 e gli 0 sono stati "negati".
Non ci sono simboli speciali per le operazioni NAND e NOR nell'algebra Booleana: possiamo tuttavia rappresentare ciascuna funzione mediante i simboli AND, OR e NOT che già abbiamo incontrato. Una porta NAND equivale al seguente semplice circuito:

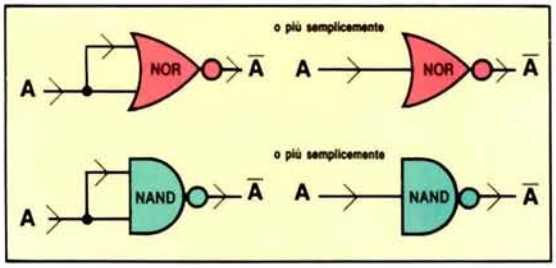


ed una porta NOR ad una porta OR seguita da una porta NOT:



Uso di NAND e NOR

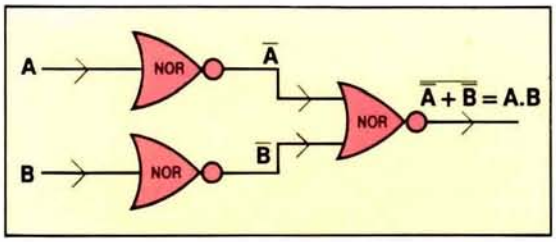
Proprio come è possibile disegnare circuiti AND/OR/NOT equivalenti alle porte NAND e NOR, così possiamo rappresentare ciascuna di queste tre porte principali come una serie di porte NOR o una serie di porte NAND.
"Porte NOT": la negazione è ottenibile sia in una porta NOR che in una porta NAND, collegando tra loro entrambi gli ingressi:



"Porte AND": in termini di algebra Booleana, l'uscita di una porta AND, con ingressi A e B, è A.B. Possiamo tuttavia trasformare questa espressione dandole una forma più utile:

$A.B = \overline{\bar{A}.\bar{B}}$ (come $A = \bar{\bar{A}}$)
$= \overline{\bar{A}+\bar{B}}$ (legge di de Morgan)

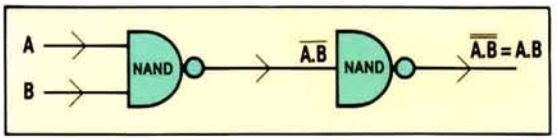
Possiamo quindi realizzare il circuito facendo passare NOT(A) e NOT(B) attraverso una porta NOR:



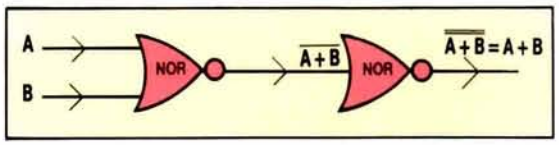
Inoltre è possibile creare una porta AND usando porte NAND. L'output di una porta NAND è $\overline{A.B}$.
Se questo output viene negato, otterremo:

$\overline{\overline{A.B}} = A.B$

Così, il circuito equivalente è:



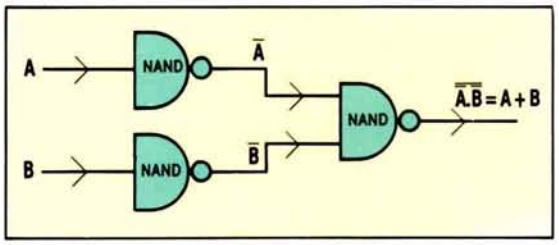
"Porte OR": poiché il collegamento "in serie" di due porte NAND equivale ad una porta AND, collegando tra loro due porte NOR otterremo un circuito equivalente ad una porta OR;



L'uscita di una porta OR è A+B. Con le regole dell'algebra Booleana, questa espressione può essere trasformata in una forma NAND:

$A+B = \overline{\overline{A}+\overline{B}} = \overline{\overline{A}.\overline{B}}$

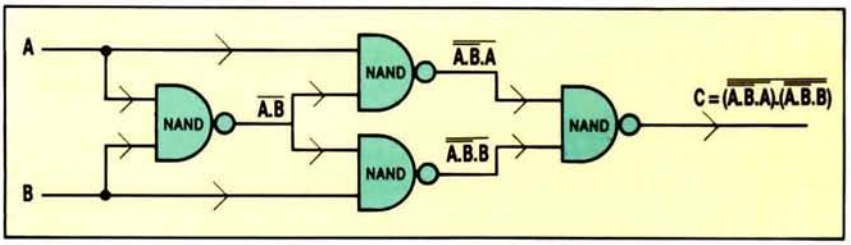
e di conseguenza il corrispondente circuito in cui vengono usate porte NAND è:



Se vogliamo costruire un circuito usando solo elementi NAND o NOR, possiamo ancora seguire i metodi di semplificazione che già abbiamo incontrato ma, innanzi tutto, dobbiamo trasformare l'espressione Booleana finale in una forma appropriata. Per i circuiti che incorporano porte NAND, useremo quelle regole dell'algebra Booleana che ci permettono di creare un'espressione composta da una serie di porte AND collegate da OR ed utilizzeremo ripetutamente la legge di de Morgan finché l'espressione apparirà interamente in forma NAND. Per circuiti in forma NOR useremo regole analoghe, come mostrato nell'esempio. Per meglio chiarire l'uso di tali regole, esaminiamo nuovamente la porta OR Esclusivo (XOR) (vedere a pag. 527). L'uscita di un XOR può essere definita dall'espressione $C = \overline{A.B}.(A+B)$.
Prendiamo questa espressione e convertiamola in modo che il circuito per la porta XOR possa essere costruito esclusivamente con porte NAND. Innanzi tutto, trasformiamo l'espressione in modo da ottenere gruppi di porte AND collegate da OR.

$C = \overline{A.B}.(A+B)$
$= (\overline{A.B}.\overline{A})+(\overline{A.B}.B)$ (proprietà distributiva)
$= (\overline{A.B}.A).\overline{(A.B}.B)$ (legge di de Morgan)

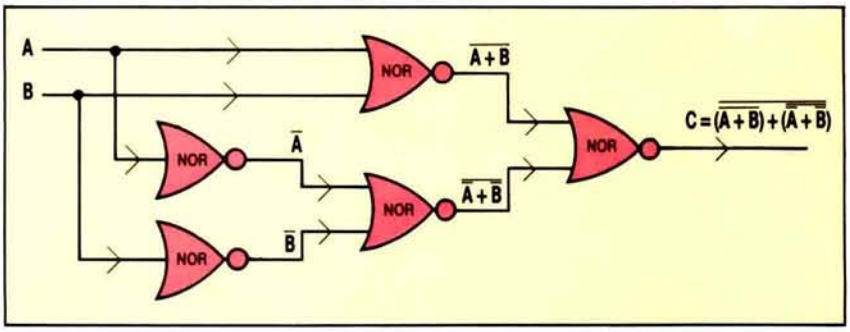
Per disegnare lo schema del circuito sulla base di un'espressione complessa come quella appena vista, conviene partire dall'uscita e tracciare il circuito procedendo "a ritroso", verso gli ingressi. Per capire come è stato concepito il seguente circuito, cerchiamo di seguirlo muovendoci dall'uscita agli ingressi:



Per la forma NOR, possiamo nuovamente partire dall'espressione originale, semplificata, per la porta XOR e trasformarla in una serie di porte OR collegate da AND. Questa prima operazione può essere eseguita applicando la legge di de Morgan sulla parte a sinistra:

$C = \overline{A.B}.(A+B) = (\overline{A}+\overline{B}).(A+B) = \overline{(\overline{A}+\overline{B})+(\overline{A}+\overline{B})}$

La trasformazione di quest'espressione in uno schema di circuito viene fatta partendo dall'output e procedendo in direzione dell'input.



Risposte all'Esercizio 6 a pag. 627

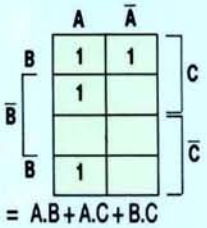
1a)

Ingressi			Uscita
A	B	C	P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

b) L'espressione Booleana per P è:

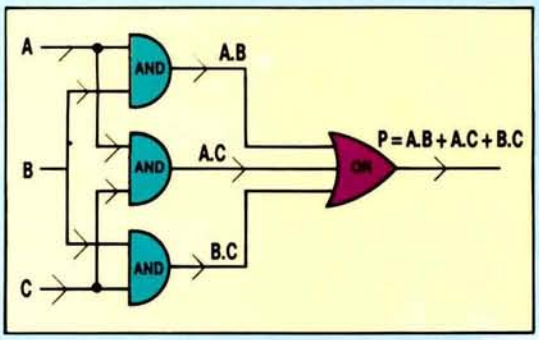
$P = \overline{A}.B.C+A.\overline{B}.C+A.B.\overline{C}+A.B.C$

Questa espressione può essere semplificata usando una mappa di Karnaugh:



Quindi $P = A.B+A.C+B.C$

c) Il circuito è:

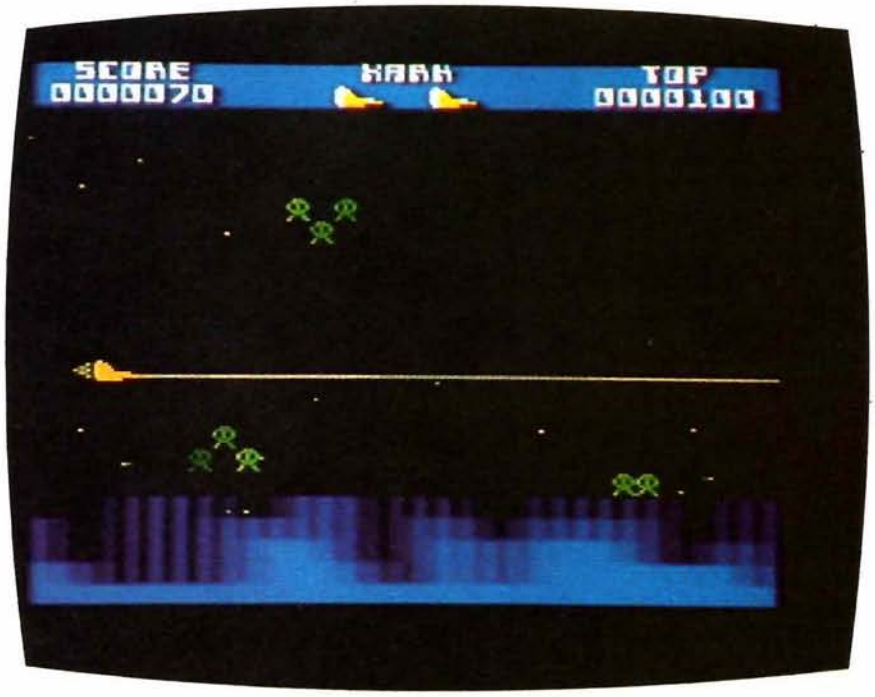


LIZ DIXON

Software in vetrina

Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Presentiamo altri due giochi: questa volta appartengono al settore delle battaglie spaziali: qualche spunto per giocare, ma anche per creare qualche nuovo gioco con le proprie conoscenze di programmazione.



Xark per lo Spectrum

Come in tutti i giochi del tipo *Defender*, il giocatore si immedesima nel ruolo dell'ultimo difensore di un pianeta (la Terra?) attaccato da astronavi aliene: dalle sue capacità dipende la sopravvivenza futura della stirpe umana. Considerando che per il programma è necessaria la versione a 48 Kbyte dello Spectrum, la grafica potrebbe essere certamente più sofisticata. L'azione si svolge abbastanza rapidamente, rendendo il gioco sufficientemente avvincente. Purtroppo, come tutti i giochi di questo tipo, ci si stanca ben presto della situazione ripetitiva in cui ci si trova.



Repton per l'Atari

Come il precedente, anche questo gioco è del tipo *Defender*, anzi, ne è addirittura una copia esatta, ma scritto per l'Atari dotato di dischetto. Lo svolgimento del gioco è quello, appunto, di difendere il pianeta da un'invasione spaziale, abbattendo tutte le astronavi nemiche, prima che gli alieni sopraffacciano il "difensore", ossia il giocatore. La grafica e la velocità di gioco sono molto buone, ma per chi abbia giocato con il *Defender* originale (per l'Apple), questo appare un surrogato un po' meno divertente. Si possono raggiungere punteggi elevati, ma a costo di ore ed ore di manovre al joystick, accessorio necessario in questo caso.

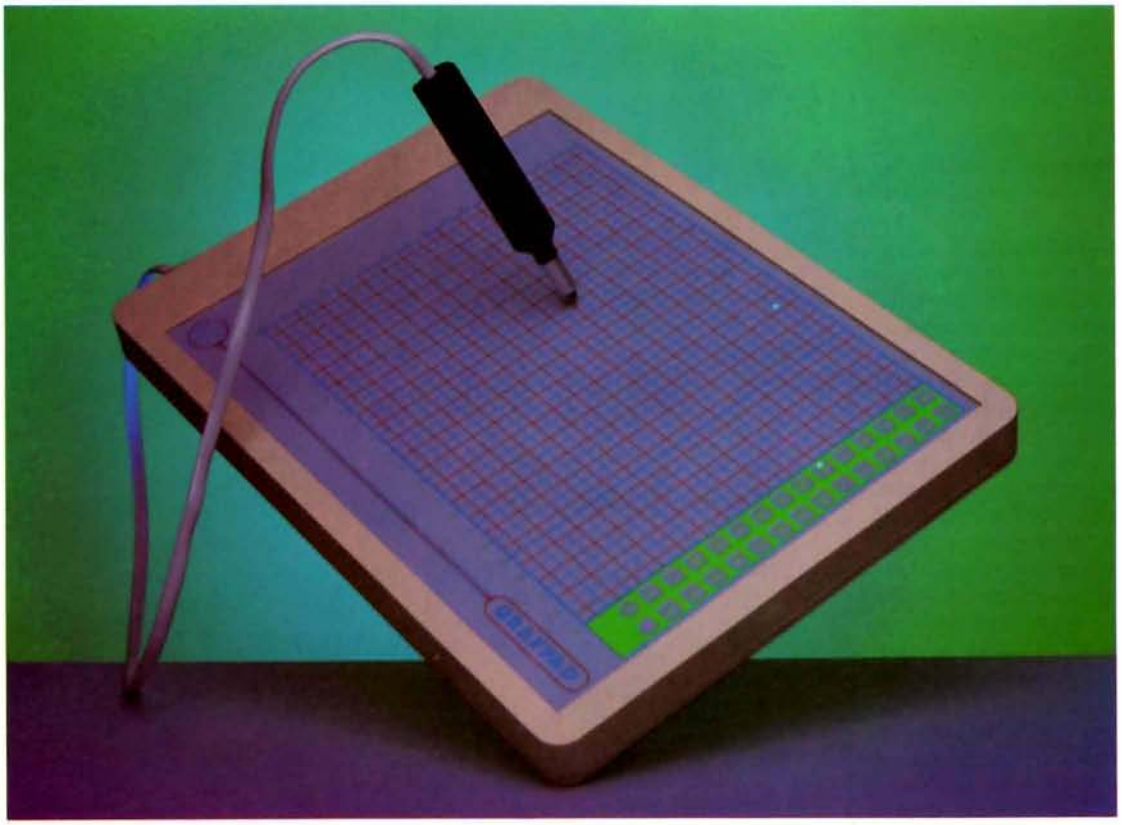
La Grafpad

Uno strumento utile e divertente, per l'elaborazione di immagini mediante il computer

Le tavolette grafiche sono forse le periferiche per microcomputer più versatili ed utili: vengono usate per il disegno e la progettazione, sia per "schizzi" a mano libera sia per la progettazione di circuiti elettronici e per il disegno di mappe. Oltre a permettere queste elementari applicazioni grafiche, tali tavolette sono un valido e rapido strumento per la selezione di comandi. Esiste quasi sempre, infatti, un più o meno piccolo pannello di comando, a lato della tavoletta, sul quale sono riportate o semplicemente le lettere dell'alfabeto o, addirittura, interi comandi. Per attivarne uno, basta selezionarlo, mediante lo "stylus" (la speciale penna in dotazione), su di un riquadro a lato della tavoletta.

Questi apparecchi sono stati, in un primo momento, realizzati più per progettisti ed ingegneri, che non per il largo pubblico, ma oggi il loro prezzo è talmente contenuto, che anche gli utenti di home computer possono acquistarsi una tavoletta grafica. La Grafpad, qui esaminata, è uno dei primi apparecchi relativamente economici, dalle buone caratteristiche. È disponibile in più versioni: per il Micro della BBC (vedere illustrazione), per il Commodore 64 e per lo Spectrum Sinclair. La Grafpad si compone di tre elementi: la tavoletta vera e propria, una penna ad essa collegata ed il software di controllo. Tanto la tavoletta che la penna vengono collegate al BBC attraverso una porta utente. La superficie della "lavagna" è suddivisa in una griglia rettangolare, di 20 quadrati per 16, ed un pannello di comando (cioè una piccola area separata sulla quale sono stampate le lettere dell'alfabeto). Il pannello di comando può essere usato per gestire alcune proprietà del software, riducendo quindi l'uso della tastiera. La superficie di lavoro della tavoletta è rivestita da uno strato di protezione in perspex. È possibile progettare i propri "cartoncini operativi" dotandoli di particolari comandi e griglie.

All'interno della tavoletta c'è una matrice di 320 × 256 fili elettrici, distanziati tra loro di circa 1,2 mm. La punta della penna agisce come un piccolo interruttore: quando si preme sul perspex di rivestimento, un chip ULA ispeziona, in successione, ciascun filo elettrico finché non individua la posizione della penna. Questo esame si ripete 2.000 volte al secondo e permette di individuare con estre-



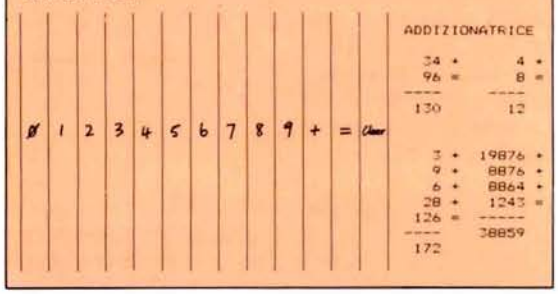
Idee grafiche

Le versatili prestazioni della Grafpad possono essere adoperate ricorrendo a programmi sviluppati per proprio conto, oppure usufruendo di quello venduto assieme all'apparecchio

UNA CALCOLATRICE

Il semplice programma che segue, scritto per il Micro BBC, serve per usare la Grafpad come una semplice addizionatrice: sul ripiano della tavoletta viene fissato un diagramma contenente i numeri ed i simboli necessari e le operazioni vengono impostate agendo semplicemente con la speciale penna

```
10 REM PROGRAMMA DIMOSTRATIVO PER LA GRAFPAD
20 REM
30 REM simulazione di un'addizionatrice
40 REM
50 MODE 1
60 PRINT "ADDIZIONATRICE"
70 HIMEM=&29FF
80 XADJUST=7
90 REM
100 REM preparazione tabella delle coordinate
110 REM
120 DIM B(15):FOR I=1 TO 15:B(I)=I*25:NEXT I
130 R1=0:R=0
140 REM
150 REM lettura programma operativo per Grafpad
160 REM
170 *LOAD "PADREAD" 2A00
180 PEN%=&2A00
190 REM
200 REM INIZIO PROGRAMMA PRINCIPALE
210 REM
220 REM attende la pressione della penna
230 CALL PEN%
240 'X%='X%-XADJUST:IF 'X%<0 THEN 'X%=0
250 IF ?UX<>0 THEN 230
260 REM emette un 'bip' alla pressione
270 SOUND 1,-15,120,1
280 X='X%
290 REM converte la posizione (tra 0 e 12)
300 I=0
310 IF X>B(I) THEN I=I+1:GOTO 310
320 REM interpreta i codici da 0 a 12
330 IF I<10 THEN R=R*10+I
340 IF I=10 THEN PRINT R;" + ";R1=R1+R:R=0
350 IF I=11 THEN PRINT R;" = ";PRINT"      -----":R1=
    R1+R:PRINT R1:R1=0:R=0:PRINT:PRINT
360 IF I=12 THEN PRINT:PRINT"AZZERATO":PRINT:R=0:R1=0
370 REM attende che la penna sia sollevata
380 CALL PEN%
390 IF ?UX=0 THEN 380
400 REM chiusura del ciclo per la successiva lettura
410 GOTO 230
```




```
ADDIZIONATRICE

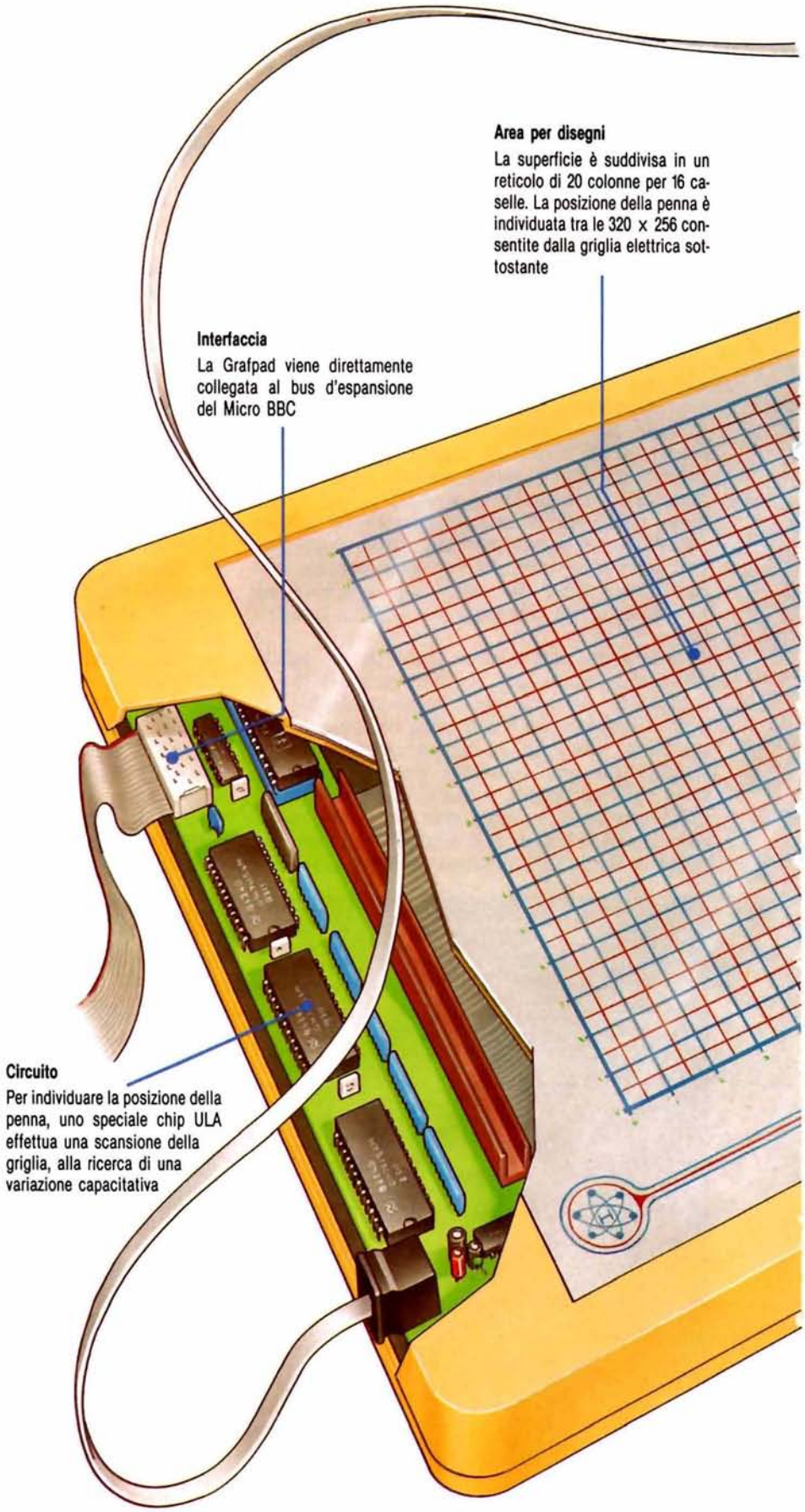
  34 +       4 +
  96 =       8 =
-----      ----
 130        12

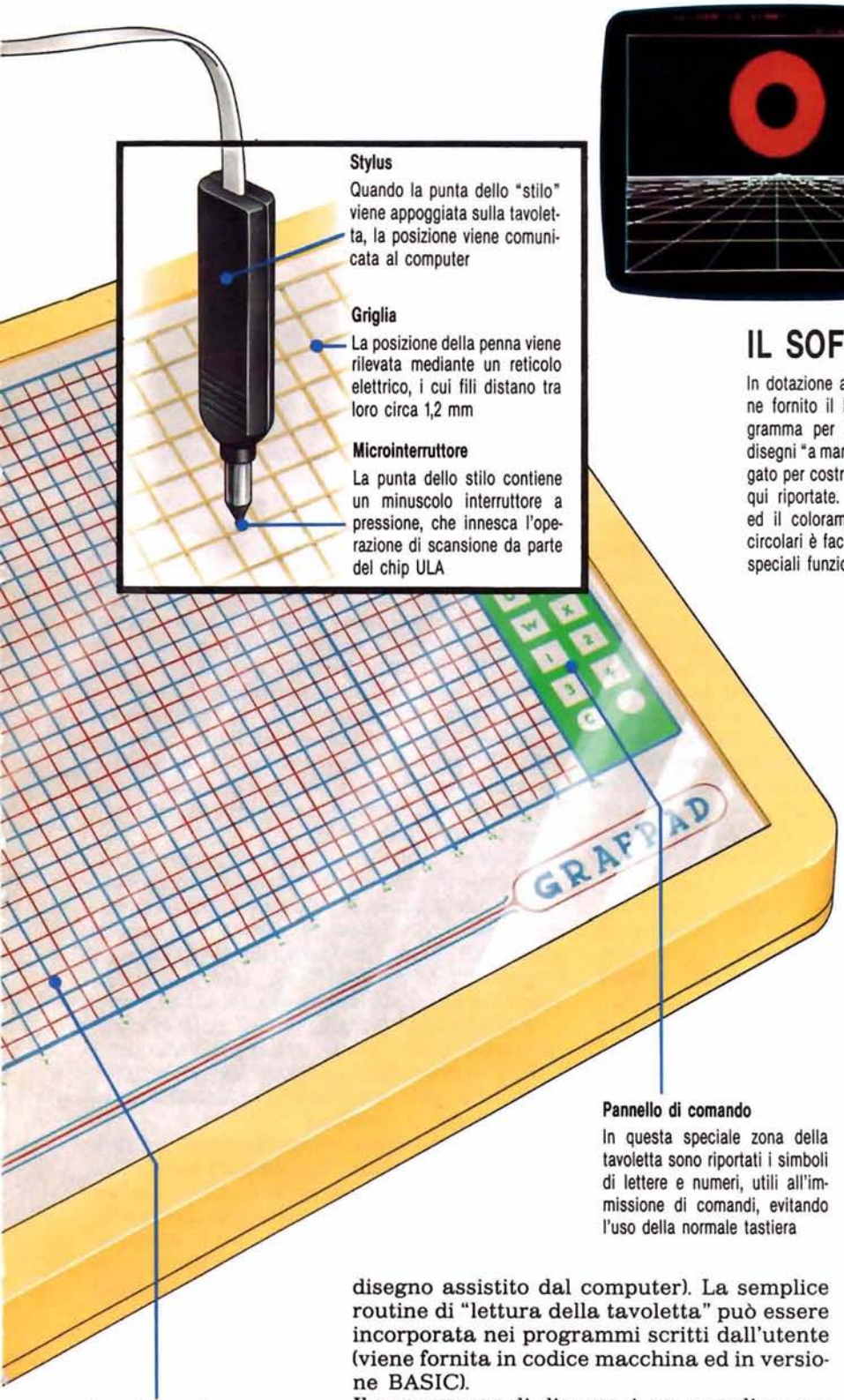
   3 +   19876 +
   9 +    8876 +
   6 +    8864 +
  28 +    1243 =
 126 =   -----
-----    38859
 172
```

ma rapidità la posizione della penna. Quest'ultima dovrebbe essere usata tenendo le dita sulla fascetta metallica, posta appena al di sopra della punta.

Quando la penna viene appoggiata sulla tavoletta, il computer riceve un segnale di "contatto" e una serie di informazioni sulle coordinate del pennino: questi dati vengono interpretati dal software. Può quindi apparire sullo schermo un cursore a forma di croce, nella posizione corrispondente, oppure può essere abilitato un particolare comando. È in questo caso che comincia a palesarsi l'economicità della Grafpad. La penna può essere individuata solo in una delle 320 x 256 posizioni di griglia e questa limitazione crea difficoltà nel disegno di figure di piccole dimensioni o dai contorni molto precisi. Le dimensioni della tavoletta sono piuttosto ridotte e l'area di lavoro sensibile alla penna ha le dimensioni di un foglio di carta formato A4.

La Grafpad è corredata da tre programmi: una banale routine dimostrativa, un semplice programma di disegno e un sofisticato pacchetto CAD (Computer Aided Design, ossia





Stylus

Quando la punta dello "stilo" viene appoggiata sulla tavoletta, la posizione viene comunicata al computer

Griglia

La posizione della penna viene rilevata mediante un reticolo elettrico, i cui fili distano tra loro circa 1,2 mm

Microinterruttore

La punta dello stilo contiene un minuscolo interruttore a pressione, che innesca l'operazione di scansione da parte del chip ULA

Pannello di comando

In questa speciale zona della tavoletta sono riportati i simboli di lettere e numeri, utili all'immissione di comandi, evitando l'uso della normale tastiera

Strato di protezione

A scopo di protezione, la superficie è ricoperta da un rivestimento in perspex, sul quale possono venir fissati disegni, diagrammi, ecc.



IL SOFTWARE

In dotazione alla Grafpad viene fornito il PROG2, un programma per la creazione di disegni "a mano libera", impiegato per costruire le immagini qui riportate. Il tracciamento ed il coloramento di sezioni circolari è facilitato, mediante speciali funzioni



disegno assistito dal computer). La semplice routine di "lettura della tavoletta" può essere incorporata nei programmi scritti dall'utente (viene fornita in codice macchina ed in versione BASIC).

Il programma di disegno è un semplice programma grafico paragonabile ai principali pacchetti software per disegno artistico, compresi quelli che non prevedono l'uso di una tavoletta grafica; ne possiede tutte le caratteristiche più importanti: linee, quadrati, cerchi, triangoli e forme "libere". Permette anche di colorare figure chiuse. Tuttavia mancano caratteristiche più sofisticate, quali la capacità di copiare o spostare parti di un disegno. La versione BBC può visualizzare solo quattro colori contemporaneamente ed i tempi di risposta sembrano piuttosto lenti.

Il programma CAD è una semplice dimostrazione di alcuni dei principi alla base di questi apparecchi. Innanzi tutto, permette la creazione di una serie di caratteri utilizzabili nei propri disegni, come, ad esempio, per impieghi elettronici, i simboli dei transistor, delle resistenze, oppure delle porte logiche, ecc. Queste operazioni sono assai simili a quelle previste in un vero e proprio programma CAD, sebbene il software Grafpad non sia adatto ad uso professionale di un certo livello. Altre utili proprietà sono: la possibilità di inserire etichette esplicative sui grafici, ruotare e mettere in scala i disegni, ingrandire un particolare settore del video e posizionare con grande precisione piccoli particolari. Manca una certa flessibilità nella correzione degli errori. Inoltre, nonostante la presenza del piccolo pannello di comando, molti comandi possono essere inseriti solo da tastiera e ciò rende il funzionamento globale dell'apparecchio piuttosto macchinoso.

La Grafpad è una versatile periferica, che ha un buon rapporto prezzo/prestazioni. Per mantenere il prezzo a livelli ragionevoli, l'area di lavoro, la risoluzione grafica e l'affidabilità non sono state sviluppate al massimo ed anche il software di corredo può lasciare perplessi, ma l'apparecchio suscita senz'altro l'interesse di coloro che, con una certa pratica di programmazione, sono capaci e disposti a sviluppare propri programmi di supporto. Nonostante queste limitazioni, principalmente dovute al tentativo di contenere il prezzo di vendita, una maggiore diffusione di tavolette grafiche di questo tipo permetterà anche ai meno esperti di esplorare nuove applicazioni e potrà dare un notevole impulso allo sviluppo di una grafica più sofisticata sugli home computer.

Ordini e magazzini

Come vengono usati i computer per una gestione ordinata ed efficiente del magazzino

In attività commerciali ben organizzate, dove si hanno le idee chiare su quali saranno le richieste della clientela e si conoscono alla perfezione le giacenze di magazzino, raramente si verificano errori per eccesso o per difetto negli approvvigionamenti. Entrambi gli eventi dipendono, infatti, da scarsità d'informazione, ma a ciò può essere ovviato mediante un sistema computerizzato di gestione del magazzino.

Per ottenere una buona gestione del magazzino il computer deve mettere a disposizione una precisa serie di informazioni: la situazione generale del magazzino, una sua valorizzazione, la velocità di rotazione di particolari linee di prodotti o di singoli articoli, quando sarà utile fare una nuova ordinazione, ecc.

La gestione computerizzata del magazzino ha come primo obiettivo la visualizzazione dei movimenti di magazzino. Tali movimenti possono venir suddivisi nelle seguenti categorie: uscite di magazzino in risposta ad ordini di vendita, entrate di magazzino (cioè merci acquistate dai fornitori), articoli in giacenza in attesa di ordinazioni e articoli giacenti, ma già "prenotati" da un ordine.

A queste quattro categorie deve essere aggiunta la possibilità di aggiornare la quantità dei singoli articoli presenti in magazzino in risposta a beni rifiutati dai clienti o restituiti ai fornitori. Inoltre, l'inventario di magazzino spesso mette in evidenza discrepanze tra le informazioni in nostro possesso e ciò che in realtà si trova sugli scaffali.

Il sistema deve anche riflettere le variazioni dei valori di magazzino, così, oltre a registrare le quantità e a visualizzare i movimenti di magazzino, il programma deve gestire le variazioni dei prezzi.

I sistemi di magazzino si dividono in due categorie piuttosto diverse tra loro, a seconda che siano stati sviluppati per il commercio (la distribuzione o la vendita al dettaglio), oppure per ditte manifatturiere. Nel secondo caso, il sistema di magazzino dovrà tener conto di ciò che viene prelevato dal magazzino, sotto forma di materia prima necessaria al processo di lavorazione, e di ciò che viene successivamente aggiunto, quale prodotto finito. Molti sistemi di magazzino basati su microcomputer tentano di soddisfare le esigenze di entrambi i tipi di attività. In queste pagine concentriamo l'attenzione sulle attività commerciali di distribuzione e vendita al dettaglio.

Poiché la gestione del magazzino è collegata ai più disparati aspetti di attività commerciale, di solito i sistemi di magazzino vengono integrati da un certo numero di altri programmi ("integrazione" significa che due o più pacchetti applicativi potranno scambiarsi vicendevolmente valori e informazioni). Un tipico sistema completamente integrato potrebbe essere collegato alla gestione degli ordini di acquisto e quelli di vendita, alla fatturazione, e alla contabilità generale.

L'integrazione presenta un buon numero di vantaggi. Consideriamo, ad esempio, un'attività commerciale che adotti un sistema di magazzino collegato al sistema di gestione degli ordini di vendita: se questi due sistemi sono in grado di comunicare tra loro, gli archivi di magazzino vengono automaticamente aggiornati, quando viene inserito un ordine di vendita. Inoltre, se il sistema di gestione degli ordini di vendita può ricevere dall'archivio di magazzino la descrizione completa degli articoli e dei loro prezzi, in base al codice di ciascun articolo, l'operatore spreca meno tempo ad inserire dati alla tastiera, facendo anche diminuire la possibilità di commettere errori.

Il punto di partenza di ogni sistema di magazzino è naturalmente l'archivio "anagrafico" di magazzino. A ciascun articolo viene associato un opportuno codice (in genere numerico). Questo codice viene usato dai vari programmi come chiave di ricerca, consentendo la realizzazione sia di sistemi gestionali relativamente semplici, sia di sistemi assai complessi. Ovviamente, come già visto per i sistemi di contabilità, poter sfruttare unità a dischetti facilita e rende più rapido il funzionamento dei programmi.

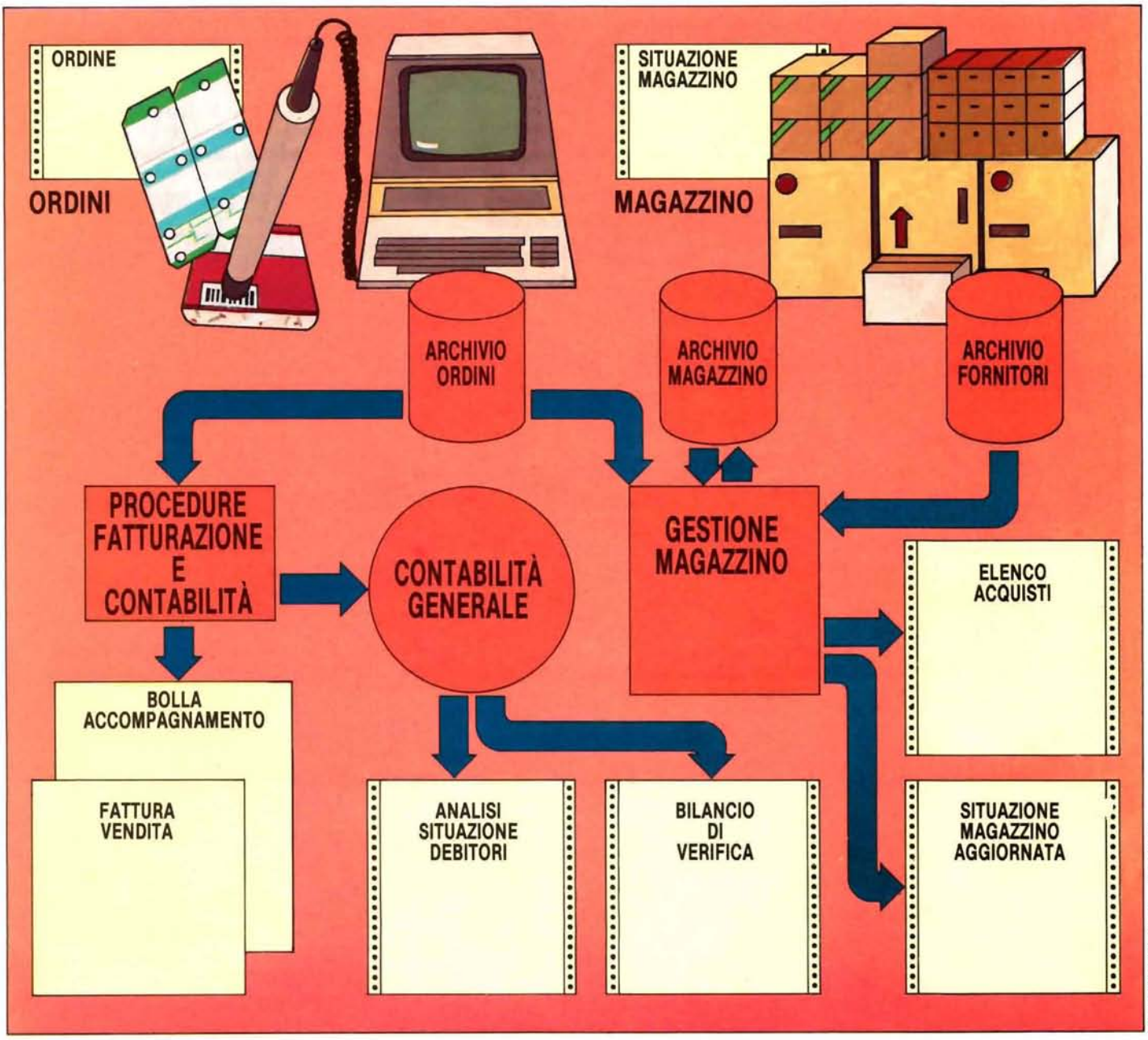
In alcuni programmi vengono messi a disposizione otto caratteri alfanumerici per il codice dell'articolo di magazzino, più un codice numerico a due cifre, per ciascun gruppo di prodotti. Ciò significa che, oltre ad aver accesso ad ogni singolo articolo (in base al proprio codice), è anche possibile ottenere dei tabulati relativi a ciascun gruppo di articoli: una comodità non trascurabile. Se ad un articolo di magazzino viene assegnato un codice di lunghezza inferiore alle otto cifre, poniamo 445, in genere il sistema provvede automaticamente ad "allineare" il numero a destra, cosicché i seguenti codici: 445, 00445 e 00000445, vengono interpretati dal sistema come identici fra loro.

Questo concetto è di estrema importanza, tant'è vero che in alcuni programmi viene offerta la possibilità di scegliere, ovviamente una volta per tutte, tra due sistemi di codifica, uno con allineamento a destra, l'altro con allineamento a sinistra.

L'allineamento a destra è maggiormente adatto ai sistemi che impiegano codici ordinati numericamente, mentre quello a sinistra meglio si presta quando vengono adottati sistemi di codifica più complessi, con caratteri alfanumerici, ad esempio: PX445/44. Questa impostazione permette di inserire codici di diversa lunghezza per differenti prodotti ed è necessaria in quei sistemi nei quali l'utente desidera utilizzare il codice di prodotto per indicare alcune caratteristiche della linea di magazzino come la forma, la dimensione, il colore ecc.
In sistemi più costosi, si può sfruttare un sistema di codifica più complicato, definito in termini di gruppi di magazzino. Un gruppo di magazzino può essere composto da un qualsiasi insieme di informazioni relative al magazzino, collegate tra loro da un comune processo di elaborazione o da comuni esigenze di tabulazione. Si differenzia dal sistema di codifica allineato a sinistra, in quanto ciascun gruppo di magazzino può essere gestito separatamente secondo proprie regole di elaborazione.
Da tutto ciò risulta che la struttura del codice di un sistema di magazzino deve essere abbastanza flessibile onde permettere agli utenti di identificare e suddividere convenientemente gli articoli che compongono il magazzino.
I sistemi più semplici utilizzati sugli home computer meno costosi sono di solito meno flessibili a causa dei limiti di memoria e d'archiviazione. Se si adoperano degli home computer dotati del solo registratore a cassette, un massimo tipico è di 350 articoli di magazzino (anche se c'è chi riesce ad ottenere molto di più), mentre con unità a dischetti questo limite è molto più ampio ed esso dipende quasi esclusivamente dalla capacità dei dischetti.

Gestione di magazzino

I sistemi di gestione integrata di vendite e di magazzino, nelle attività commerciali di larghe dimensioni, possono ottenere grandi vantaggi da un sistema automatizzato: le giacenze di magazzino sono costantemente aggiornate assieme alla fatturazione e all'intera situazione contabile.

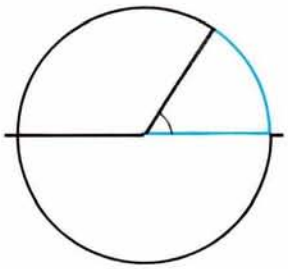


Gradi di precisione

È bene verificare la precisione del BASIC prima di usare le funzioni trigonometriche

Poiché il BASIC è dotato delle due funzioni SIN e COS, il calcolo della posizione di un punto, dopo averlo ruotato per un certo numero di gradi, è relativamente semplice. Il valore di COS Θ dà la posizione sull'asse x, mentre quello di SIN Θ dà la posizione sull'asse y. Tuttavia, nell'uso di queste comode funzioni, occorre ricordare che la maggior parte dei BASIC misura gli angoli in radianti e non in gradi. Inoltre, per valori molto prossimi a 0 o ad 1, dette funzioni possono non essere precise. Per prima cosa affrontiamo la differenza tra radianti e gradi.
Se tracciamo una porzione di cerchio (ossia un "arco") di lunghezza pari al raggio della circonferenza, l'angolo al centro misura, per definizione, un radiante (vedere illustrazione). Se il raggio della circonferenza è unitario, altrettanto unitaria è la lunghezza dell'arco. La formula per trovare il perimetro di una circonferenza è $2\pi r$: da ciò consegue che vi sono 2π radianti in una rotazione completa. In altri termini, poiché in un cerchio vi sono 360°, 2π equivale a 360°.
Ecco, dunque, alcuni semplici rapporti tra gradi e radianti:

Misura in radianti



360° = 2π radianti
180° = π radianti
90° = $\pi/2$ radianti
1° = $\pi/180$ = 0,0174 radianti

Un programma BASIC per il calcolo del coseno di un angolo deve quindi, prima di applicare la funzione, convertire il valore in radianti.
Si provi:

```
10 INPUT "ANGOLO IN GRADI";A
20 LET B# = A*0.0174
30 LET C# = COS(B#)
40 PRINT "IL COSENO DI ";A; "GRADI È ";C#
50 END
```

Il simbolo #, usato nel programma, contraddistingue le variabili a "doppia precisione", che vengono spiegate più avanti. Con semplici modifiche, si ottiene il calcolo del seno degli angoli compresi tra 0° e 360°, sotto forma di tabulato. Se questi valori vengono poi impiegati per tracciare una curva, riferendoli agli assi x ed y, il risultato è un'onda sinusoidale, la cui forma è ben nota agli appassionati di hi-fi (vedere a pagina 635). Ogni punto di questa curva corrisponde ad un'intersezione del raggio con ciascun punto del cerchio, riportando le coordinate sotto forma di grafico. In altre parole, abbiamo definito un cerchio usando un metodo alternativo, rispetto a quello matematico.
Sono pochi i BASIC nei quali si può scegliere, per le funzioni trigonometriche, radianti o gradi. Se il nostro BASIC prevede soltanto l'uso di radianti, possiamo ovviare definendo una funzione speciale, che operi la conversione. Ad esempio:

```
10 REM FUNZIONE DEFINITA DA UTENTE PER LA CONVERSIO-
   NE IN RADIANTI
20 DEF FNDSIN(D#) = SIN(D#*0.017453293)
30 INPUT "ANGOLO IN GRADI";D#
40 PRINT "IL SENO DI ";D#;" GRADI È";FNDSIN(D#)
50 END
```

La linea 20 definisce una funzione DSIN, che utilizza una sola variabile, a doppia precisione, D#. La parte a destra della definizione stabilisce come elaborare il valore della variabile (in questo caso: il seno dell'angolo in

Errori di arrotondamento nel BASIC

```
1 REM VERIFICA SU VALORI NUMERICI MOLTO PICCOLI
10 LET X# = .000033333333333#
20 PRINT "ITERAZIONE","DOPPIA  PR.","SINGOLA PR."
30 PRINT
40 FOR I = 1 TO 40
50 LET X# = X# / 10
60 LET X! = X#
70 PRINT I,X#,X!
80 NEXT I
90 END
```

ITERAZIONE	DOPPIA PR.	SINGOLA PR.
1	.0000033333333333	3.33333E-06
2	.00000033333333333	3.33333E-07
3	3.333333333D-08	3.33333E-08
4	3.333333333D-09	3.33333E-09
5	3.333333333D-10	3.33333E-10
6	3.333333333D-11	3.33333E-11
7	3.333333333D-12	3.33333E-12
8	3.333333333D-13	3.33333E-13
9	3.333333333D-14	3.33333E-14
10	3.333333333D-15	3.33333E-15
11	3.333333333D-16	3.33333E-16
12	3.333333333D-17	3.33333E-17
13	3.333333333D-18	3.33333E-18
14	3.333333333D-19	3.33333E-19
15	3.333333333D-20	3.33333E-20
16	3.333333333D-21	3.33333E-21
17	3.333333333D-22	3.33333E-22
18	3.333333333D-23	3.33333E-23
19	3.333333333D-24	3.33333E-24
20	3.333333333D-25	3.33333E-25
21	3.333333333D-26	3.33333E-26
22	3.333333333D-27	3.33333E-27
23	3.333333333D-28	3.33333E-28
24	3.333333333D-29	3.33333E-29
25	3.333333333D-30	3.33333E-30
26	3.333333333D-31	3.33333E-31
27	3.333333333D-32	3.33333E-32
28	3.333333333D-33	3.33333E-33
29	3.333333333D-34	3.33333E-34
30	3.333333333D-35	3.33333E-35
31	3.333333333D-36	3.33333E-36
32	3.333333333D-37	3.33333E-37
33	3.333333333D-38	3.33333E-38
34	0	0
35	0	0
36	0	0
37	0	0
38	0	0
39	0	0
40	0	0

gradi). Per sfruttare questa nuova funzione nel resto del programma, basta indicarne il nome, facendo seguire (racchiuso tra parentesi) il valore da usare nel calcolo. La linea che contiene la definizione deve essere eseguita prima di poter usare la funzione.
Il problema con la funzione seno, in BASIC, è che, per valori di Θ tendenti a 0 il risultato non è sempre preciso. Ciò può apparire strano, ma dipende dalla limitata precisione del BASIC per quanto concerne sia i valori numerici molto grandi, sia quelli molto vicini allo zero. Quindi, se l'angolo Θ è molto piccolo (ad esempio 1.0E-36, ossia 1 × 10 elevato alla meno 36_{ma}) il valore del seno risultante è talmente piccolo, che il BASIC lo considera 0, anche quando non lo è affatto! Prima di fidarsi ciecamente della funzione SIN, si provi il piccolo programma riportato nella tabella della pagina precedente.
Sono riportati anche i valori ottenuti con un "lancio" del programma usando il BASIC Microsoft. Questo interprete non ha problemi fino a valori di Θ minori a 1.0E-38 (ossia: zero, virgola, 37 zeri, uno).
È bene ricordarsi che esistono dei limiti ai valori calcolati dalle funzioni del BASIC, prima di profonderne l'impiego nei nostri programmi.
A questo proposito è bene rammentare che, generalmente, se una variabile non viene definita altrimenti, essa è a "singola precisione" (dotata di 7 cifre significative). Quindi, X o SOMMA sono a singola precisione. Se vogliamo evidenziare questo fatto, possiamo usare il simbolo !, come in X! e SOMMA!, mentre possiamo usare variabili a "doppia precisione" (con ben 17 cifre significative) semplicemente aggiungendo un simbolo # al nome della variabile, ad esempio X# o SOMMA#. Le variabili "intere" (ossia capaci di contenere soltanto numeri interi) sono indicate usando il simbolo %, come in X% o SOMMA%.
Questa lezione termina con la presentazione di due brevi programmi che consentono di stabilire cifre significative siano contenute nei diversi tipi di variabili nella particolare versione di BASIC impiegata. Se questa è una versione Microsoft, viene anche prodotto un tabulato su stampante. Il primo programma esegue una verifica sui numeri molto piccoli, il secondo, invece, sui numeri molto grandi.
La stampa relativa al primo programma evidenzia che quando si raggiungono valori molto piccoli (minori di 3.3 × 10E-38), il BASIC tende ad arrotondarli allo zero. Per valori molto grandi, invece, può presentarsi un errore del tipo "OVERFLOW", traducibile con "traboccamento" (il numero è troppo grande anche per il computer!) e può esser necessario scrivere appositi moduli aritmetici, se nei programmi l'ordine di grandezza necessario è molto alto.

Numeri piccoli nel BASIC

```
1 REM VERIFICA SU VALORI NUMERICI MOLTO PICCOLI
10 LET X# = .000033333333333#
20 PRINT "ITERAZIONE","DOPPIA  PR.","SINGOLA PR."
30 PRINT
40 FOR I = 1 TO 40
50 LET X# = X# / 10
60 LET X! = X#
70 PRINT I,X#,X!
80 NEXT I
90 END
```

ITERAZIONE	DOPPIA PR.	SINGOLA PR.
1	.0000033333333333	3.33333E-06
2	.00000033333333333	3.33333E-07
3	3.3333333333D-08	3.33333E-08
4	3.3333333333D-09	3.33333E-09
5	3.3333333333D-10	3.33333E-10
6	3.3333333333D-11	3.33333E-11
7	3.3333333333D-12	3.33333E-12
8	3.3333333333D-13	3.33333E-13
9	3.3333333333D-14	3.33333E-14
10	3.3333333333D-15	3.33333E-15
11	3.3333333333D-16	3.33333E-16
12	3.3333333333D-17	3.33333E-17
13	3.3333333333D-18	3.33333E-18
14	3.3333333333D-19	3.33333E-19
15	3.3333333333D-20	3.33333E-20
16	3.3333333333D-21	3.33333E-21
17	3.3333333333D-22	3.33333E-22
18	3.3333333333D-23	3.33333E-23
19	3.3333333333D-24	3.33333E-24
20	3.3333333333D-25	3.33333E-25
21	3.3333333333D-26	3.33333E-26
22	3.3333333333D-27	3.33333E-27
23	3.3333333333D-28	3.33333E-28
24	3.3333333333D-29	3.33333E-29
25	3.3333333333D-30	3.33333E-30
26	3.3333333333D-31	3.33333E-31
27	3.3333333333D-32	3.33333E-32
28	3.3333333333D-33	3.33333E-33
29	3.3333333333D-34	3.33333E-34
30	3.3333333333D-35	3.33333E-35
31	3.3333333333D-36	3.33333E-36
32	3.3333333333D-37	3.33333E-37
33	3.3333333333D-38	3.33333E-38
34	0	0
35	0	0
36	0	0
37	0	0
38	0	0
39	0	0
40	0	0

Numeri grandi nel BASIC

```
1 REM VERIFICA PER VALORI NUMERICI MOLTO GRANDI
10 LET X# = 3.333333333333334#
20 PRINT "ITERAZIONE","DOPPIA PR.","SINGOLA PR."
30 PRINT
40 FOR I = 1 TO 40
50 LET X# = X# * 10
60 LET X! = X#
70 PRINT I,X#,X!
80 NEXT I
90 END
```

ITERAZIONE	DOPPIA PR.	SINGOLA PR.
1	33.33333333333334	33.3333
2	333.3333333333334	333.333
3	3333.333333333334	3333.33
4	33333.33333333334	33333.3
5	333333.3333333334	333333
6	3333333.333333334	3.33333E+06
7	33333333.33333334	3.33333E+07
8	333333333.3333334	3.33333E+08
9	3333333333.333334	3.33333E+09
10	33333333333.33334	3.33333E+10
11	333333333333.3334	3.33333E+11
12	3333333333333.334	3.33333E+12
13	33333333333333.34	3.33333E+13
14	333333333333333.4	3.33333E+14
15	3333333333333334	3.33333E+15
16	3.3333333333333334D+16	3.33333E+16
17	3.3333333333333334D+17	3.33333E+17
18	3.3333333333333334D+18	3.33333E+18
19	3.3333333333333334D+19	3.33333E+19
20	3.3333333333333334D+20	3.33333E+20
21	3.3333333333333334D+21	3.33333E+21
22	3.3333333333333334D+22	3.33333E+22
23	3.3333333333333334D+23	3.33333E+23
24	3.3333333333333334D+24	3.33333E+24
25	3.3333333333333334D+25	3.33333E+25
26	3.3333333333333334D+26	3.33333E+26
27	3.3333333333333334D+27	3.33333E+27
28	3.3333333333333334D+28	3.33333E+28
29	3.3333333333333334D+29	3.33333E+29
30	3.3333333333333334D+30	3.33333E+30
31	3.3333333333333335D+31	3.33333E+31
32	3.3333333333333335D+32	3.33333E+32
33	3.3333333333333334D+33	3.33333E+33
34	3.3333333333333335D+34	3.33333E+34
35	3.3333333333333335D+35	3.33333E+35
36	3.3333333333333335D+36	3.33333E+36
37	3.3333333333333335D+37	3.33333E+37
38	1.7014118346046937D+38	1.70141E+38
39	1.7014118346046937D+38	1.70141E+38
40	1.7014118346046937D+38	1.70141E+38

Sventola una bandiera

Il ruolo della Flag di Carry nelle operazioni aritmetiche in codice macchina

L'istruzione di somma, sia nel linguaggio ASSEMBLY Z80 che in quello 6502, è ADC (Add with Carry = somma aggiungendo il Carry) che è di fondamentale importanza nella programmazione in codice macchina, come del resto il concetto di "riporto" o "Carry". Consideriamo la somma di due valori esadecimali all'interno dell'accumulatore:

```
   A7      =      10100111
+  3E      =   +  00111110
-------------------------
   E5      =      11100101
```

Poiché l'accumulatore è un registro ad otto bit, tanto i numeri da sommare che il risultato devono essere compresi tra $00 e $FF (come nell'esempio), altrimenti essi "sconfinano". Ciò non significa, comunque, che non si possano sommare valori più grandi. Consideriamo la seguente somma, che contravviene ai limiti appena citati:

```
   FF      =      11111111
+  FF      =   +  11111111
-------------------------
   1FE     =      111111110
```

Gli addendi, in questo caso, sono i massimi valori ottenibili con un byte di 8 bit e, ad esser precisi, per la somma occorrerebbe un accumulatore di 9 bit.
La soluzione però è facilmente individuabile: basta un solo bit in più, per manipolare il risultato (non gli addendi) dell'operazione di somma e ciò soltanto quando è richiesto un riporto.

Il registro di condizione del processore

Il bit aggiuntivo è chiamato "bit di Carry" ed è conservato in ciò che viene definito il "registro di condizione del processore" (abbreviato in PSR, dall'inglese Processor Status Register), anch'esso a 8 bit. Questo registro è collegato sia all'accumulatore che all'unità logico-aritmetica, in modo da riflettere il risultato di certe operazioni aritmetiche o logiche compiute sull'accumulatore. Il contenuto del PSR può essere considerato un semplice numero, ma è in genere molto più conveniente vederlo come una serie di otto singole "flag" (letteralmente "bandiere", ma più propriamente segnalatori d'evento), ognuna delle quali si riferisce ad un particolare effetto dell'ultima operazione eseguita sull'accumulatore.
In questo caso, i valori delle flag possono essere soltanto due: 1 oppure 0 (acceso/spento, on/off), il che si adatta perfettamente alla logica dei computer.
Se il risultato di una somma eseguita sull'accumulatore "sconfina", la flag di Carry assume il valore 1, altrimenti viene "resettata" (ossia posta a zero).
Alcune operazioni, quali il caricamento dell'accumulatore o il trasferimento di dati, non modificano affatto il valore del Carry. Quando in seguito affronteremo le varie istruzioni del codice macchina, faremo spesso riferimento al PSR, per esaminare gli effetti che esse hanno sulle flag, ma per adesso ci limitiamo al flag di Carry.
In generale, quando si addizionano due valori di otto bit, non sappiamo a priori se il risultato è maggiore o minore di $FF: occorre quindi riservare al risultato due byte di RAM. Si considerino di nuovo gli esempi fatti:

Numeri hex		Flag di Carry		Numeri binari
A7	=			10100111
+ 3E	=		+	00111110
00E5	=	0		11100101



FF	=			11111111
+ FF	=		+	11111111
01FE	=	1		111111110



In ambedue i casi abbiamo usato due byte per rappresentare il risultato.
Nel primo caso la flag di Carry è posta a zero (non c'è riporto), mentre nel secondo è posta ad 1, ed il secondo byte del risultato è pari a $01.
Per esser certi di conservare correttamente il risultato di una somma, quindi, occorre copiare il contenuto dell'accumulatore nella porzione "bassa" della locazione di memoria, e poi copiare il valore del Carry nella parte

alta. Non esiste, purtroppo, una singola istruzione per far ciò, ma la ADC serve egregiamente allo scopo, poiché somma il valore che segue l'istruzione al valore del Carry e a quello dell'accumulatore.
Ciò significa, però, che dobbiamo conoscere il valore del Carry prima dell'operazione, ecco dunque perché nella scorsa lezione sono state usate le istruzioni CLC ed AND A. La prima, nel 6502, significa "azzera la flag di Carry"; la seconda, nello Z80, equivale a "esegui un AND dell'accumulatore con se stesso". Benché non particolarmente progettata per azzerare solamente il Carry, l'effetto è proprio questo.
Avendo azzerato il Carry prima della somma, resta da conservare il valore che esso contiene dopo la somma, ciò si può facilmente ottenere sommando $00 all'accumulatore: nulla accade se il Carry è zero, ma viene sommato il valore $01 se il Carry è pari ad 1.
Tutto quanto è stato esposto fin qui ci porta a formulare un primo metodo per le operazioni aritmetiche su singoli byte:

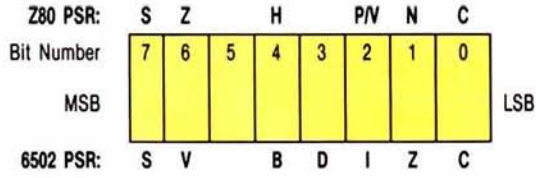
1) Azzerare la flag di Carry
2) Caricare nell'accumulatore un valore
3) Sommare il secondo valore
4) Conservare il risultato nella parte bassa della locazione di memoria per il risultato
5) Caricare nell'accumulatore la parte alta della locazione
6) Sommare $00 all'accumulatore
7) Depositare il contenuto dell'accumulatore nella parte alta della locazione per il risultato

In linguaggio ASSEMBLY:

IN COMUNE PER I DUE PROCESSORI		
Etichetta	**Direttiva**	**Operando**
BYTE1	EQU	$FF
BYTE2	EQU	$FF
LOBYTE	EQU	$A000
HIBYTE	EQU	$A001
	ORG	$A020

Z80		6502	
Opcode	**Operando**	**Opcode**	**Operando**
LD	A,$00	LDA	#$00
LD	(HIBYTE),A	STA	HIBYTE
AND	A	CLC	
LD	A,BYTE1	LDA	#BYTE1
ADC	A,BYTE2	ADC	#BIYTE2
LD	(LOBYTE),A	STA	LOBYTE
LD	A,(HIBYTE)	LDA	HIBYTE
ADC	A,$00	ADC	#$00
LD	(HIBYTE),A	STA	HIBYTE
RET		RTS	

Si rammenti che i valori qui indicati per LOBYTE, HIBYTE ed ORG sono teorici: essi vanno adattati al particolare tipo di computer in uso. Si noti che le prime istruzioni servono ad azzerare il contenuto di HIBYTE, per evitare che esso contenga dati a caso (magari frutto di una precedente operazione).
Non occorre far lo stesso con LOBYTE, poiché in esso viene depositato un valore, prima di qualsiasi lettura.
Vale la pena di soffermarsi nuovamente sulle differenze tra il linguaggio ASSEMBLY 6502 e quello dello Z80.
Nel primo caso le istruzioni sono abbastanza chiare, con l'uso del simbolo # per indicare i valori "immediati".
Nel secondo caso, invece, si può essere fuorviati dall'apparente somiglianza di tutte le istruzioni di caricamento (viene sempre usata la LD), e dall'assenza del simbolo # (vengono invece, usate parentesi) per indicare il verso dei trasferimenti. Cosicché: LD A,BYTE1 significa "carica l'accumulatore con il valore immediato BYTE1", mentre LD A, (HIBYTE) significa "carica l'accumulatore col contenuto della locazione HIBYTE".
Nel listato completo di un programma in ASSEMBLY, eventuali dubbi possono essere fugati verificando il valore esadecimale del codice di ciascuna istruzione. Tuttavia, con un regolare esercizio, ci si abitua rapidamente ai vari codici mnemonici.
Infine, va fatto notare che il PSR contiene altre flag, oltre a quella del Carry, che verranno esaminate più avanti nel corso:



PSR BIT	Z80	6502	PSR BIT
7	(S) = SEGNO	(S) = SEGNO	7
6	(Z) = ZERO	(V) = OVERFLOW	6
5	non usato	non usato	5
4	(H) = HALFCARRY	(B) = BREAK	4
3	non usato	(D) = MODO BCD	3
2	(P/V) = PARITÀ/OVERFLOW	(I) = INTERRUPT	2
1	(N) = SOTTRAZIONE	(Z) = ZERO	1
0	(C) = CARRY	(C) = CARRY	0

Per i nostri scopi attuali, le flag di maggiore importanza sono quella di Carry, quella del Segno e quella di Zero. Abbiamo visto che, a seguito di una somma, il Carry contiene il valore del riporto.
Nella flag del Segno viene sempre copiato l'ottavo bit dell'accumulatore e la flag di Zero viene posta uguale ad 1 se il risultato dell'operazione è zero e viceversa.

Risposte all'esercizio sul linguaggio ASSEMBLY di pag. 638

1) I programmi assemblati sono riportati qui a fianco. Si noti che i simboli BYTE1 e BYTE2 vengono adoperati tanto come dati immediati, quanto come indirizzi. In questo secondo caso, però, occorre assemblarli nel formato a due byte.

2) In ambedue i programmi manca l'istruzione "ritorno da subroutine". Nella versione 6502 occorre aggiungere la seguente linea:

A00D 60 RTS

mentre nella versione Z80:

A00D C9 RET

3) Prima viene caricato in accumulatore il valore $45. Ad esso viene sommato $45, cosicché l'accumulatore contiene il risultato $8A, che viene depositato nella RAM all'indirizzo $0045. A questo punto, all'accumulatore viene sommato il valore immediato $38, ed il risultato finale, ossia $C2($45+$45+$38) è depositato nella locazione RAM $0038.

4) Il termine "dato immediato" si riferisce al valore che segue il codice della relativa istruzione. Nelle istruzioni esaminate, (quali: LDA #$9C e LD A, $E4) i valori $9C e $E4 sono i dati immediati da caricare nell'accumulatore e formano parte integrante dell'istruzione di caricamento. Qualora un dato dovesse essere sconosciuto, al momento della redazione del programma, allora ad esso dovrebbe essere riservata una particolare locazione di RAM a cui l'istruzione di caricamento dovrebbe riferirsi.

5) Il valore di BYTE1 è dato come $45, ma in realtà si tratta di $0045, che, ovviamente, appartiene alla "pagina zero" della memoria.

Esercizio

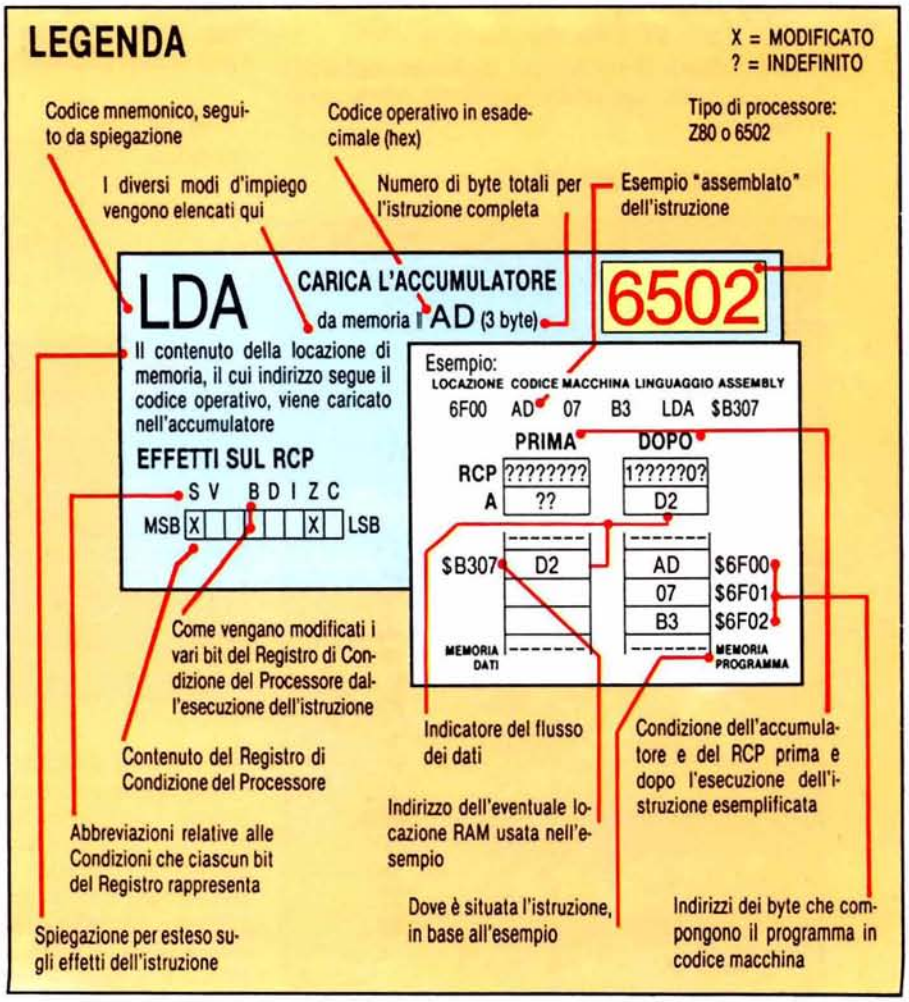
Può essere utile esaminare il contenuto del registro di condizione del processore (PSR) e sarebbe conveniente visualizzarlo in formato binario, anziché decimale o esadecimale. Ecco una subroutine di conversione da decimale a binario, per lo Spectrum.
L'esercizio consiste nell'inserirla come parte del programma monitor, presentato a pag. 598.

Basic & Basic

Sul Commodore 64, cambiare la linea 7050 in

7050 B$ = MID$(STR$(R),2) + B$

Indirizzo locazione	Codice Macchina	Linguaggio ASSEMBLY		
6052				
0000		START	EQU	$A000
0000		BYTE1	EQU	$45
0000		BYTE2	EQU	$38
0000			ORG	START
A000	A9 45		LDA	#BYTE1
A002	18		CLC	
A003	69 45		ADC	#BYTE1
A005	8D 45 00		STA	BYTE1
A008	69 38		ADC	#BYTE2
A00A	8D 38 00		STA	BYTE2
Z80				
0000		START	EQU	$A000
0000		BYTE1	EQU	$45
0000		BYTE2	EQU	$38
0000			ORG	START
A000	3E 45		LD	A,BYTE1
A002	A7		AND	A
A003	CE 45		ADC	A,BYTE1
A005	32 45 00		LD	(BYTE1),A
A008	CE 38		ADC	A,BYTE2
A00A	32 38 00		LD	(BYTE2),A



LDA — CARICA L'ACCUMULATORE — 6502

modo immediato ‖ A9 (2 byte)

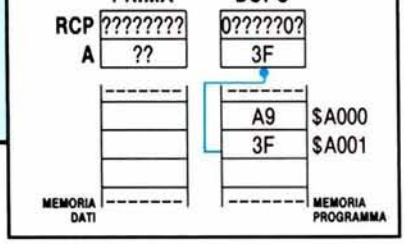
Il valore del byte che segue il codice operativo viene caricato nell'accumulatore

EFFETTI SU RCP

	S	V		B	D	I	Z	C
MSB	X						X		LSB

Esempio:

LOCAZIONE	CODICE MACCHINA		LINGUAGGIO ASSEMBLY	
A000	A9	3F	LDA	#$3F



LD A, — CARICA L'ACCUMULATORE — Z80

modo immediato ‖ 3E (2 byte)

Il valore del byte che segue il codice operativo viene caricato nell'accumulatore

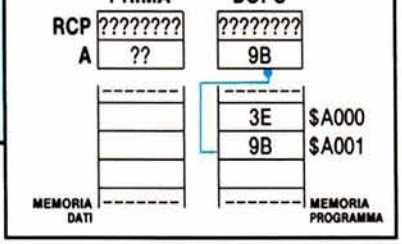
EFFETTI SUL RCP

	S	Z		H		V	N	C
MSB									LSB

NESSUN EFFETTO

Esempio:

LOCAZIONE	CODICE MACCHINA		LINGUAGGIO ASSEMBLY	
A000	3E	9B	LD	A,$9B



LDA — CARICA L'ACCUMULATORE — 6502

dalla memoria ‖ AD (3 byte)

Il contenuto della locazione di memoria il cui indirizzo è specificato nell'istruzione viene copiato nell'accumulatore

EFFETTI SUL RCP

	S	V		B	D	I	Z	C
MSB	X						X		LSB

Esempio:

LOCAZIONE	CODICE MACCHINA			LINGUAGGIO ASSEMBLY	
6F00	AD	07	B3	LDA	$B307



LD A,() — CARICA L'ACCUMULATORE — Z80

da memoria ‖ 3A (3 byte)

Il contenuto della locazione di memoria, il cui indirizzo segue il codice operativo, viene caricato nell'accumulatore

EFFETTI SUL RCP

	S	Z		H		V	N	C
MSB									LSB

NESSUN EFFETTO

Esempio:

LOCAZIONE	CODICE MACCHINA			LINGUAGGIO ASSEMBLY	
6F00	3A	E9	F4	LD	A,($F4E9)



STA — CONSERVA L'ACCUMULATORE — 6502

in memoria ‖ 8D (3 byte)

Il contenuto dell'accumulatore viene copiato nella locazione, il cui indirizzo segue il codice operativo

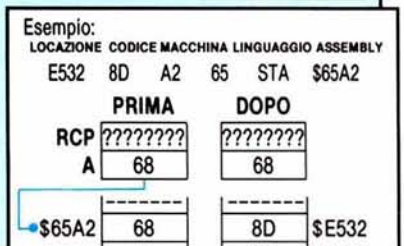
EFFTTI SU RCP

	S	V		B	D	I	Z	C
MSB									LSB

NESSUN EFFETTO

Esempio:

LOCAZIONE	CODICE MACCHINA			LINGUAGGIO ASSEMBLY	
E532	8D	A2	65	STA	$65A2



LD( ),A — CONSERVA L'ACCUMULATORE — Z80

in memoria ‖ 32 (3 byte)

Il contenuto dell'accumulatore viene copiato nella locazione, il cui indirizzo segue il codice operativo

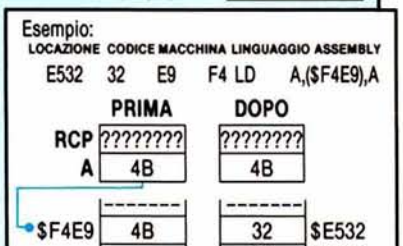
EFFETTI SUL RCP

	S	Z		H		V	N	C
MSB									LSB

NESSUN EFFETTO

Esempio:

LOCAZIONE	CODICE MACCHINA			LINGUAGGIO ASSEMBLY	
E532	32	E9	F4	LD	A,($F4E9),A



ADC — SOMMA CON IL CARRY — 6502

modo immediato ‖ 69 (2 byte)

Il contenuto dell'accumulatore viene sommato al valore che segue il codice operativo ed al contenuto del bit di Carry (riporto)

EFFETTI SUL RCP

	S	V		B	D	I	Z	C
MSB	X	X					X	X	LSB

Esempio:

LOCAZIONE	CODICE MACCHINA		LINGUAGGIO ASSEMBLY	
840B	69	A2	ADC	#$A2



ADC A, — SOMMA CON IL CARRY — Z80

modo immediato ‖ CE (2 byte)

Il contenuto dell'accumulatore viene sommato al valore che segue il codice operativo ed al contenuto del bit di Carry (riporto)

EFFETTI SUL RCP

	S	Z		H		V	N	C
MSB	X	X		X		X		X	LSB

Esempio:

LOCAZIONE	CODICE MACCHINA		LINGAGGIO ASSEMBLY	
840B	CE	B9	ADC	A,$B9



La Virgin Games

Una ditta con grande esperienza nel campo musicale si occupa oggi di software

Agli inizi degli anni '70 il boom degli home computer stava appena cominciando, ma già un nutrito stuolo di giovani intraprendenti, specialmente negli USA ed in Gran Bretagna, si dedicava alla produzione di software su cassette, di cui c'era una forte domanda. Chiunque sapesse programmare qualche originale gioco in BASIC si procurava un duplicatore di nastri a cassetta ad alta velocità ed iniziava a vendere per corrispondenza, mettendo piccoli annunci sulle riviste specializzate.

Oggi, le cose non sono più così semplici. Per quanto bravo possa essere un programmatore, occorre una fervida fantasia e molta originalità per farsi strada. Le software house operano secondo modalità piuttosto diverse. La Imagine Software, per esempio (vedere a pag. 559), impiega uno staff di programmatori che traducono in programmi le idee dei loro capi.

Nick Alexander, il ventottenne direttore della Virgin Games, asserisce che in genere i buoni programmatori hanno poche idee originali. Ciò può anche esser vero, poiché la logica, il metodo e la precisione (doti tipiche in un buon programmatore), non fanno mostra di sé, in genere, negli individui creativi. Per questo motivo, alla Virgin Games, il numero di programmatori è ridotto al minimo essenziale. La politica adottata è quella di offrire un servizio di consulenza ed assistenza per correggere e migliorare le caratteristiche dei numerosissimi programmi che giungono alla Virgin Games da giovani appassionati. Nasce così un connubio tra il lavoro di esperti programmatori e di fertili autori di giochi, dotati di buone idee.

I risultati economici offerti agli autori dalla Virgin Games possono sembrare molto al di sotto della media (circa il 7,5% di diritti d'autore, contro il 25% delle altre software house). Alexander si giustifica affermando che, se si considera che gran parte degli utili vengono di nuovo investiti nella promozione e nella distribuzione dei prodotti, il risultato finale può essere, quantitativamente, superiore a quello offerto da altre ditte.

La Virgin, in campo promozionale e commerciale, la sa decisamente lunga. Infatti, Richard Branson, il titolare di questa Casa, ha lavorato per molti anni nella produzione musicale. In effetti, la Virgin lancia gli autori di software in modo simile a quanto accade per le "star" musicali: le cassette dei giochi non contengono solo un'accurata biografia dell'autore, ma perfino una sua foto!

Quando la Virgin apparve sulla scena, nel febbraio del 1983, ricevette subito circa 500 offerte da parte di autori di giochi. Oggi possiede circa 46 titoli di successo per otto diversi tipi di home computer. In testa alle classifiche sono i giochi per lo Spectrum, poi vengono quelli per il Micro BBC e, infine, quelli per il Commodore 64.

Uno degli ultimi "divi del computer", lanciati dalla Virgin è Martin Wheeler, un ragazzo di 15 anni, che ha scritto due ottimi programmi per lo Spectrum: *Dr. Franky and the Monsters* e *Sorcerers*. Wheeler ha scritto i programmi in codice macchina, sviluppando una grafica a dir poco eccezionale.

Alexander ritiene che quanto accaduto una decina di anni fa nel settore musicale, si stia ripetendo oggi nel campo degli home computer e può darsi che, prima o poi, questi due aspetti della cultura si incontrino, proprio grazie ai più giovani.

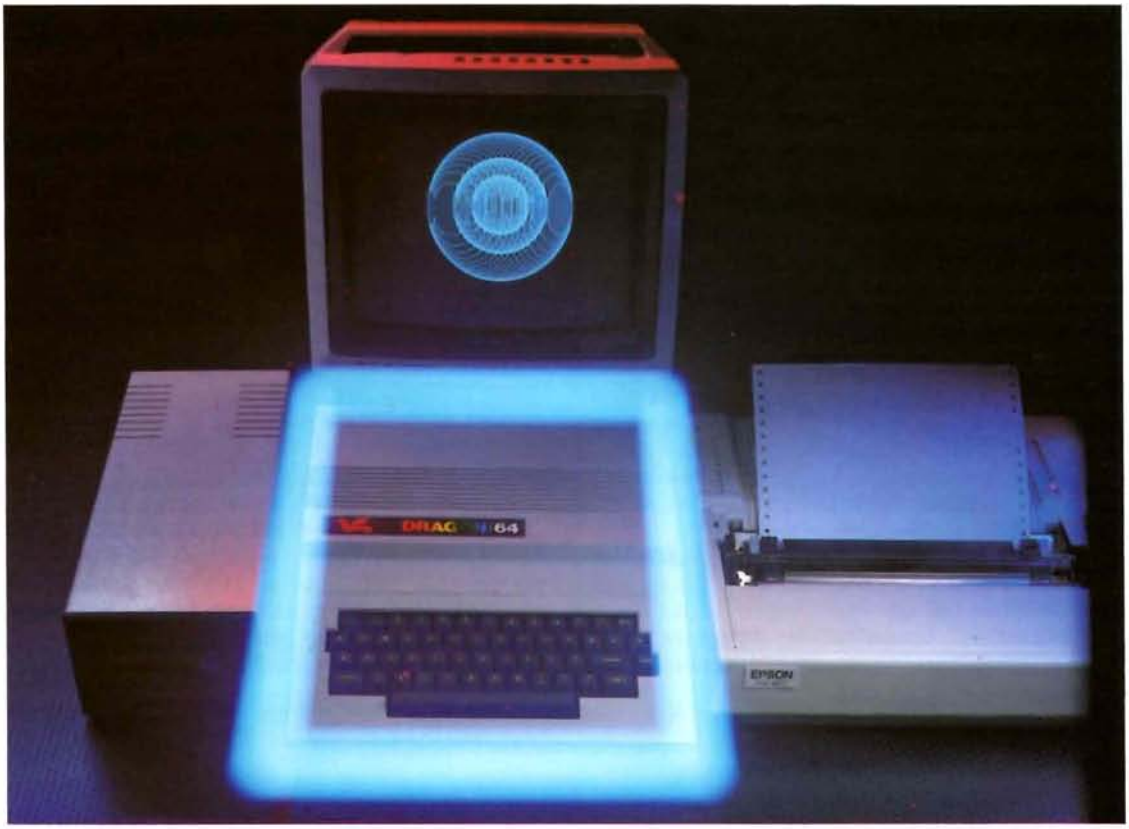


Nick Alexander

Richard Branson

I sistemi operativi

Il lavoro dei sistemi operativi è spesso sottovalutato, ma è di fondamentale importanza



PAUL CHAVE

Il Manager
Ogni computer possiede una forma di sistema operativo, che, come un buon manager, sovrintende e controlla tutte le operazioni svolte dal computer e dalle sue periferiche

Con i linguaggi ad alto livello il programmatore può ignorare le operazioni interne della CPU e realizzare software facilmente trasportabile da un computer all'altro. Infatti, posto che il linguaggio usato non sia una versione molto particolare, le istruzioni funzioneranno regolarmente anche su altri apparecchi dotati dello stesso linguaggio.
È l'interprete, o il compilatore, che durante l'elaborazione delle istruzioni che formano il programma, assegna opportunamente le varie porzioni di memoria occorrenti. L'interprete ed il compilatore non sono, in realtà, che dei programmi e come tali devono essere caricati nella memoria del computer prima di poter convertire il programma "sorgente" in codice eseguibile. Fanno eccezione i BASIC memorizzati su ROM, che sono immediatamente accessibili.
Tuttavia, al funzionamento di un computer occorre molto più che un semplice programma di conversione da codice sorgente a codice oggetto: è necessario un programma che sovrintenda a tutte le operazioni basilari del sistema quali, ad esempio, la visualizzazione sullo schermo di un carattere premuto sulla tastiera. In qualche parte della memoria deve esserci un programma che provvede ad una continua scansione della tastiera, "a caccia" di eventuali tasti premuti dall'operatore. Occorre individuare di quale tasto si tratti e, successivamente, comunicare ai circuiti video la corretta sequenza dei punti da illuminare sullo schermo, per produrre il corrispondente carattere. Tutte le funzioni simili a questa sono dette "trasparenti" per l'utente, in quanto avvengono senza che l'operatore debba in qualche modo occuparsene.
Un'altra funzione "trasparente" è associata, per esempio, al comando CLOAD: il trasferimento dal nastro magnetico alla memoria del computer avviene senza che l'operatore debba nemmeno conoscere il formato dei segnali letti dal registratore: a tutto ciò pensa il sistema operativo.
Il sistema operativo, dunque, è un programma "di sottofondo", sempre in funzione, che governa tutte le operazioni. In certi casi la distinzione tra sistema operativo e BASIC non risulta semplice, soprattutto negli apparecchi in cui ambedue i programmi sono

memorizzati su ROM. Nella sua forma più elementare, dunque, una ROM contiene tutto il software di base occorrente per il sistema, ma non i programmi applicativi (word processor, giochi, ecc.), generalmente realizzati ed immessi dall'utente. Una parte della ROM conterrà l'interprete BASIC, un'altra conterrà, invece, il programma necessario alla redazione dei listati, ossia l'"editor"; un'altra parte ancora conterrà tutto il software "di sistema" (chiamato anche "monitor") per la gestione della tastiera, del video, del registratore, ecc.

Il termine "monitor", che non deve essere confuso con quello che indica l'apparecchio di visualizzazione, è più o meno equivalente a "sistema operativo". Nella sua forma più elementare, un monitor non fa altro che accettare istruzioni in codice macchina, depositarle in memoria ed eseguirle. Ma se le operazioni cui esso sovrintende hanno una maggiore complessità, allora si preferisce indicarlo col termine "sistema operativo".

All'estremo opposto troviamo i computer per applicazioni commerciali, nei quali vengono impiegate unità a dischi in cui il sistema operativo è molto più sofisticato e completo. Prima di esaminare gli apparecchi che appartengono alla fascia intermedia, come l'Apple, vediamo brevemente le caratteristiche del sistema operativo di un computer dotato unicamente di dischetti.

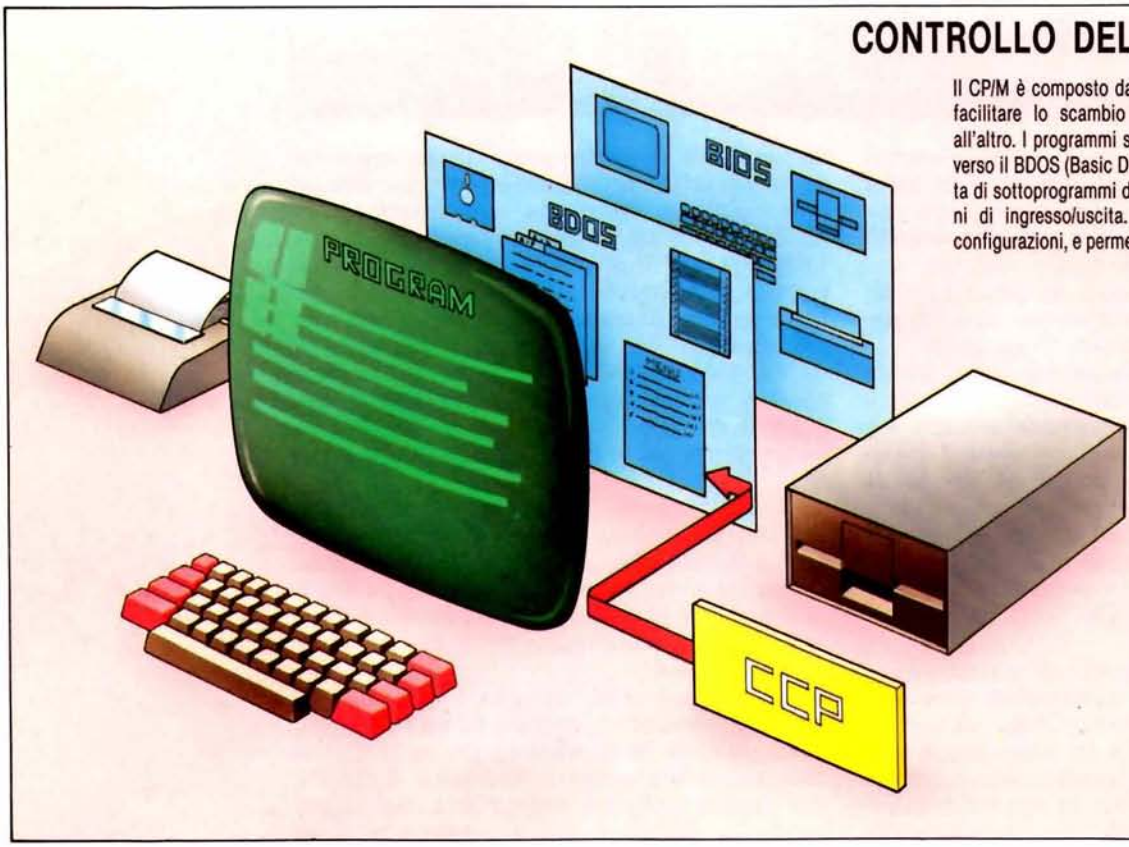
Generalmente, in apparecchi di questo tipo, le ROM non contengono che qualche elementare routine "di servizio", oltre ad un "bootstrap loader". Quest'ultimo contiene quelle poche istruzioni in codice macchina che, all'accensione del computer, servono alla CPU per accedere ai dischi e per trasferire il sistema operativo nella memoria centrale.

Il sistema operativo, in questo caso, deve provvedere a funzioni molto più complesse di quelle necessarie a un computer di minori pretese, basato su di un sistema operativo su ROM: deve possedere i comandi necessari ad accedere e a manipolare gli archivi su disco (i files). Devono esserci ad esempio comandi per elencare, cancellare o variare i nomi dei files presenti sul dischetto o per copiare un archivio da un dischetto all'altro.

Di contro, i sistemi operativi di apparecchi più semplici possiedono solo qualche comando utile al trasferimento dei programmi da e verso il registratore a cassette. Un sistema operativo di buon livello tiene nota della localizzazione, sul dischetto o sul nastro, di ogni particolare gruppo di informazioni, permettendo in tal modo rapide ricerche; nella maggior parte dei sistemi operativi più elementari, invece, può essere necessario scorrere l'intero nastro alla ricerca di un programma.

I microcomputer della fascia intermedia, dei quali il Micro BBC è un ottimo esempio, sono capaci di gestire altrettanto bene sia le unità a cassette che quelle a dischetti, usando pressappoco gli stessi comandi. Il sistema operativo risiede nella memoria ROM, ma, trattandosi di una ROM tenuta separata dal resto della memoria, si può paragonarlo ad un sistema operativo a dischetti. Le funzioni comprendono le operazioni di trasferimento degli archivi, molto più funzionali di quelle disponibili su apparecchi come lo Spectrum.

CONTROLLO DELLE OPERAZIONI



Il CP/M è composto da più strati, progettati in modo da facilitare lo scambio di programmi da un computer all'altro. I programmi scritti dall'utente funzionano attraverso il BDOS (Basic Disk Operating System), una raccolta di sottoprogrammi d'impiego generico per le operazioni di ingresso/uscita. Il BDOS è comune a tutte le configurazioni, e permette una quasi totale portabilità del software scritto sotto CP/M. Quando il BDOS deve eseguire una specifica operazione di input/output, ricorre ai sottoprogrammi contenuti nel BIOS, che, invece, è "hardware-dipendente" (per usare un termine del gergo). Ciò significa che, per ogni particolare tipo di computer, il BIOS deve essere opportunamente adattato: a questa operazione, normalmente, provvedono i fabbricanti dei computer.

La terza ed ultima parte del CP/M è la CCP (Command Control Program, ossia programma di controllo dei comandi), che accetta ed esegue i comandi impartiti, mediante la tastiera, dall'operatore

KEVIN JONES

Possiamo quindi considerare il sistema operativo come un particolare programma, che si colloca tra la CPU (e le sue periferiche) ed i linguaggi di programmazione o il software applicativo redatto dall'operatore.

Portabilità

La possibilità di usare su altri computer il software scritto su di un particolare apparecchio si indica col termine "portabilità". In merito alla portabilità vi sono alcuni aspetti da considerare: il primo è che esistono microprocessori molto diversi tra loro che richiedono, quindi, l'uso di un diverso codice macchina. Ad esempio, l'istruzione in codice macchina per sommare il contenuto di due locazioni di memoria ha un formato per il 6502 (il chip usato per l'Apple) ed uno completamente diverso per lo Z80 (quello usato nello Spectrum). Tuttavia, il compito di convertire le istruzioni di un linguaggio ad alto livello spetta all'interprete oppure al compilatore. Ciò implica la realizzazione di interpreti e compilatori adatti a ciascun tipo di CPU.
Un secondo inconveniente è che le cose non sono affatto semplici nemmeno quando due computer adoperano la stessa CPU: le mappe di memoria possono non corrispondere nei due apparecchi, i codici usati per il movimento del cursore sullo schermo possono non coincidere ed altrettanto può accadere per le operazioni di input/output.
Per aggirare questo problema, sono stati sviluppati dei sistemi operativi che permettono lo scambio di software tra computer dotati della stessa CPU e del medesimo sistema operativo, come, ad esempio, tra due sistemi basati sul chip Z80 e forniti entrambi di unità a dischetti. Il più noto di tali sistemi operativi è il CP/M (Control Program/Microcomputers, ossia programma di controllo per microcomputer).
Il CP/M consiste, in un miglioramento degli elementari monitor dedicati a specifici chip, in modo da permettere una considerevole portabilità di software. Infatti, i programmi scritti "sotto" CP/M o MS-DOS possono girare su altri computer dotati del medesimo sistema operativo, posto che non venga fatto uso, ad esempio, di particolari funzioni grafiche non presenti su uno dei due computer.
La complessità e la flessibilità dei sistemi operativi per dischi varia considerevolmente, ma i più comuni (quali il CP/M e l'MS-DOS) sono composti, essenzialmente, di tre parti: l'interprete dei comandi, il sistema operativo per dischi vero e proprio (BDOS) ed una sezione dedicata alle funzioni basilari di ingresso/uscita (BIOS). Delle tre parti, quella che riguarda maggiormente la portabilità è il BIOS. La sezione BIOS, infatti, contiene tutti i sottoprogrammi che governano le comunicazioni tra l'unità centrale e le periferiche, e quindi deve essere adattata al particolare tipo di computer. A ciò provvedono, di solito, i fabbricanti. Tutti i programmi che adottano il CP/M comunicano con il sistema mediante il BIOS, che si occupa, ad esempio, di accettare un carattere dalla tastiera, di visualizzarlo, di accedere ai dischi, ecc.
Il BDOS contiene tutti i sottoprogrammi generici per l'accesso e la gestione delle periferiche. Il BDOS ed il BIOS assieme corrispondono grosso modo al programma monitor o ai sistemi operativi presenti sugli apparecchi di minori dimensioni.
L'interprete dei comandi è quella parte del sistema operativo che accetta e provvede ad eseguire i comandi digitati sulla tastiera. Comandi tipici sono quelli che servono per trasferire un file dalla memoria al disco o viceversa, per elencare i nomi dei file memorizzati su di un dischetto e via dicendo.
Siccome il lavoro del sistema operativo non si vede ma, anzi, come accennato precedentemente, è "trasparente", si tende spesso a sottovalutarlo, nonostante sia di fondamentale importanza per il funzionamento dell'intero sistema. I programmatori più esperti, tuttavia, riescono a sfruttare molte delle funzioni offerte dai sistemi operativi quali il CP/M, mediante programmi scritti in codice macchina. Facendo ciò, il software risulta molto più efficiente e di più rapida esecuzione di quello realizzato in linguaggi di alto livello. Il CP/M, nato per essere usato sui chip ad 8 bit 8080 e Z80, è ancora oggi uno dei più diffusi. Nel settore dei chip a 16 bit ha, però, un forte concorrente: l'MS-DOS, sviluppato dalla Microsoft, la ditta statunitense specializzata nella realizzazione di linguaggi ad alto livello per microcomputer.



IAN McKINNELL

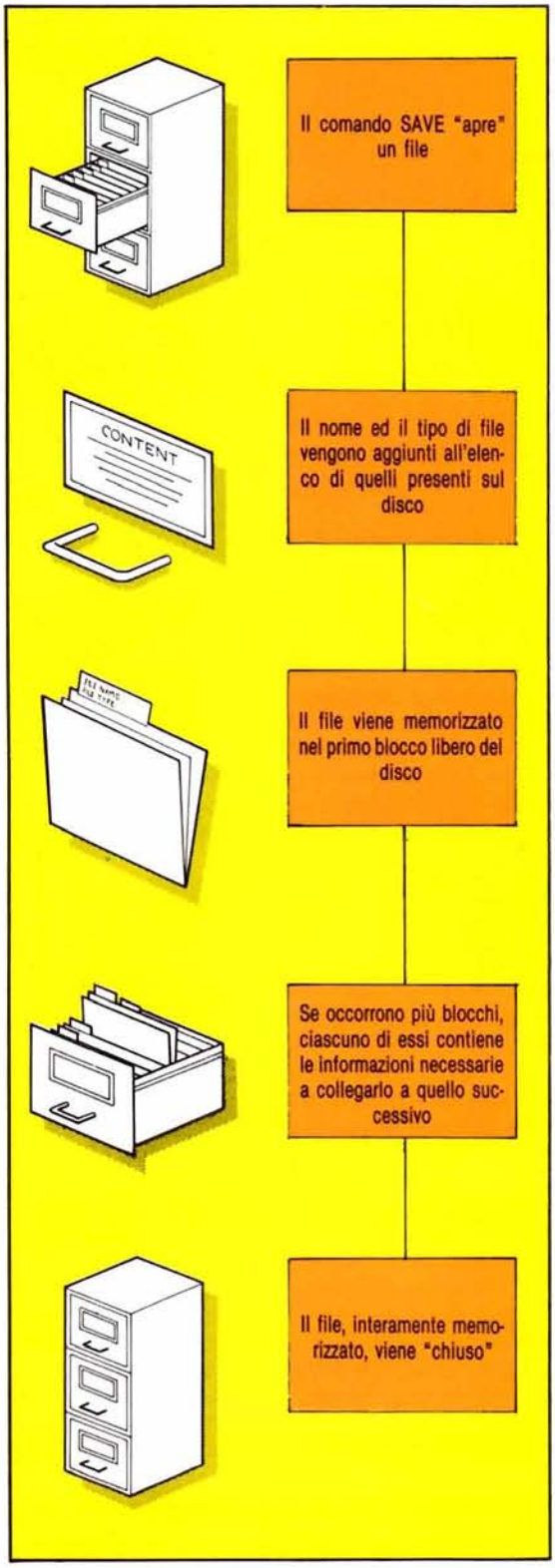
La chiave del successo
L'Osborne-1 deve gran parte del proprio successo al fatto di esser dotato del sistema operativo CP/M. Il computer, inizialmente, veniva venduto assieme ad alcuni dei migliori programmi sviluppati sotto CP/M: il Wordstar, il SuperCalc e l'MBASIC

Memoria sotto controllo

Memorizzare un programma è concettualmente simile allo staccare un foglio da un blocco per appunti

La memorizzazione di un file binario

Un file binario non è che la copia esatta di una porzione della memoria RAM. Questa può contenere un programma, oppure i codici che compongono un'immagine o un testo. La memorizzazione in sé è un processo abbastanza semplice, quasi esclusivamente attuato dal sistema operativo: basta infatti indicare il nome col quale intendiamo identificare il file. Il sistema operativo fa sì che il nuovo file non vada a sovrapporsi ad altri già esistenti



"File" è il termine impiegato, nell'informatica, per indicare un gruppo di dati (informazioni o programmi) conservati su disco o su nastro. Letteralmente, la parola significa "inserto" o "pratica", con evidente riferimento agli archivi basati su vecchi metodi manuali.

Per condurre ordinatamente la nostra esistenza quotidiana, è necessario mantenere una traccia, più accurata possibile, delle esperienze, delle transazioni monetarie e degli appuntamenti che ci coinvolgono. Molta gente adopera dei taccuini o delle agende per appuntamenti ed indirizzi, oppure le matrici degli assegni per tener conto dei pagamenti fatti. L'efficienza di un sistema informativo dipende largamente dalla rapidità con la quale si possono ritrovare notizie e dati concernenti una particolare transazione, un particolare nominativo o un preciso evento. Se si tratta di un'impresa commerciale in via di espansione, la gestione di un archivio dati può diventare molto onerosa e complessa, ma da essa dipende gran parte del successo commerciale: un'informazione errata o mal interpretata può causare danni incalcolabili! I buoni amministratori sanno quanto sia importante organizzare e strutturare adeguatamente l'archivio in base al tipo e alla quantità di informazioni da elaborare.

Tali principi si applicano anche agli archivi gestiti da apparecchi computerizzati, con il non indifferente vantaggio di una rapidità e precisione impossibili da ottenersi con metodi manuali. Al cuore di un sistema di gestione computerizzata c'è il sistema operativo del computer, che lavora con estrema precisione, purché le informazioni e le istruzioni fornite siano quelle appropriate. Ne consegue che un sistema informativo computerizzato è efficiente in misura direttamente proporzionale alla sofisticatezza delle strutture dati offerte dal sistema operativo e al modo col quale quest'ultimo viene impiegato.

I metodi usati nei microcomputer per la gestione dei file derivano in massima parte da quanto progettato a suo tempo per i computer delle vecchie generazioni, e poi per i minicomputer. Tali sistemi possono essere suddivisi in tre categorie: i file binari, quelli sequenziali e quelli ad accesso casuale o "random"; vediamoli ad uno ad uno.

I file binari

Un file binario consiste, essenzialmente, in una fedele copia di una porzione di memoria del computer. Un buon esempio di questo è

LA DIRECTORY DEL DISCO

Qui a fianco è mostrato quanto si ottiene con il comando STAT del CP/M, che fornisce una quantità di informazioni superiore ai normali comandi di elencazione

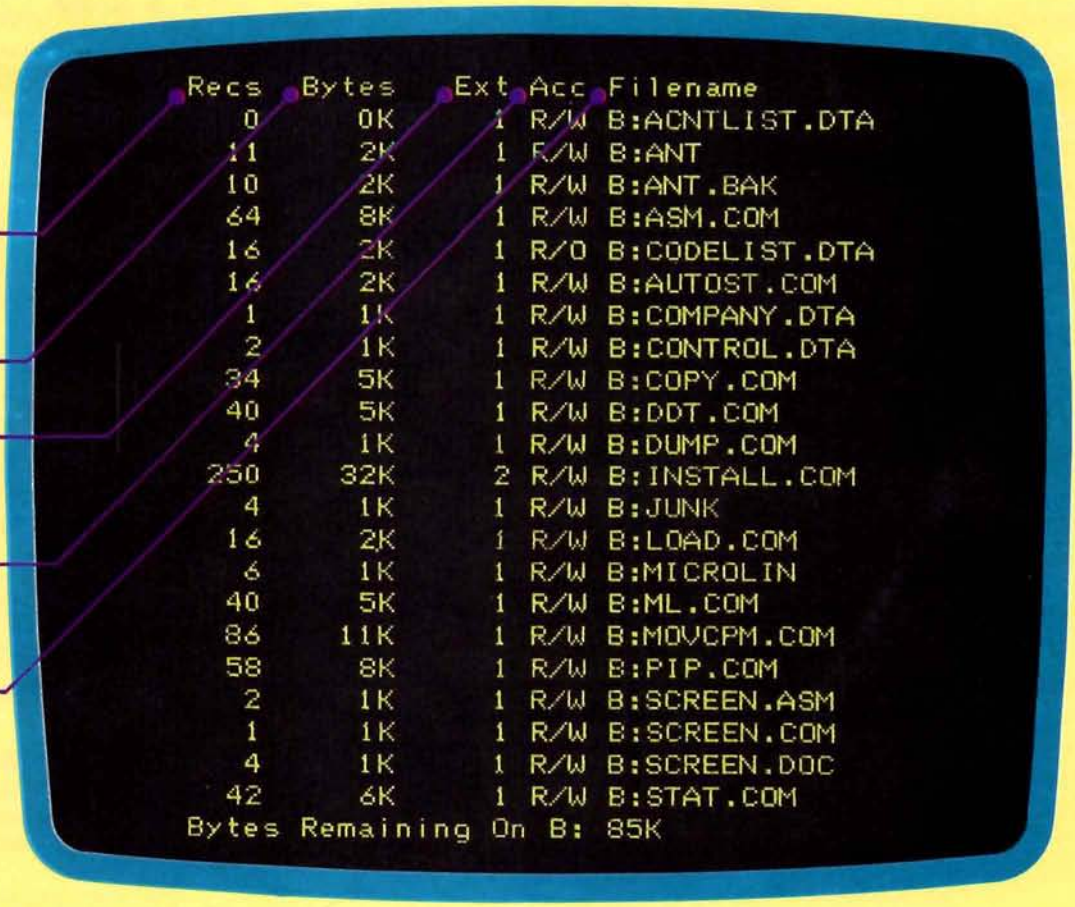
Recs
Il numero di record in un file può variare. Nel CP/M, ogni record è lungo 128 byte

Bytes
La lunghezza del file espressa in Kbyte

Ext
Il numero degli "extent" (estensioni) indica quanti blocchi di 16 Kbyte occupa il file

Acc
Tipo di accesso: il file può essere contrassegnato come a sola lettura (R/O) o a lettura/scrittura (R/W)

Filename
Il nome del file, per intero, inizia con una lettera, seguita da due punti (A:, B:, ecc.), che indica il drive. Subito dopo il nome assegnato (ad esempio: AUTOST), seguito dall'estensione (ad esempio: .COM), che indica il tipo di informazioni contenute nel file



```
Recs   Bytes   Ext Acc Filename
   0      0K     1 R/W B:ACNTLIST.DTA
  11      2K     1 R/W B:ANT
  10      2K     1 R/W B:ANT.BAK
  64      8K     1 R/W B:ASM.COM
  16      2K     1 R/O B:CODELIST.DTA
  16      2K     1 R/W B:AUTOST.COM
   1      1K     1 R/W B:COMPANY.DTA
   2      1K     1 R/W B:CONTROL.DTA
  34      5K     1 R/W B:COPY.COM
  40      5K     1 R/W B:DDT.COM
   4      1K     1 R/W B:DUMP.COM
 250     32K     2 R/W B:INSTALL.COM
   4      1K     1 R/W B:JUNK
  16      2K     1 R/W B:LOAD.COM
   6      1K     1 R/W B:MICROLIN
  40      5K     1 R/W B:ML.COM
  86     11K     1 R/W B:MOVCPM.COM
  58      8K     1 R/W B:PIP.COM
   2      1K     1 R/W B:SCREEN.ASM
   1      1K     1 R/W B:SCREEN.COM
   4      1K     1 R/W B:SCREEN.DOC
  42      6K     1 R/W B:STAT.COM
Bytes Remaining On B: 85K
```

un programma memorizzato su nastro o disco, mediante un comando SAVE. Si immagini la memoria RAM riservata ai programmi come un semplice blocco per appunti. Se volessimo conservare un disegno o degli appunti, non faremmo che staccare la pagina dal blocco, riponendola, magari, in una cartellina di facile consultazione. I file binari servono allo stesso scopo. Impartendo un comando SAVE, il sistema operativo provvede a ricopiare, un byte alla volta, tutta una porzione di RAM sulla memoria di massa (disco o nastro), assegnando al file il nome da noi scelto. Sui dischi, il programma viene memorizzato in più blocchi, alla fine di ciascuno dei quali vengono depositate le informazioni necessarie al collegamento col blocco successivo. L'ultimo blocco termina con un codice speciale, che indica la condizione di "fine file" (End of File, o spesso, più brevemente, EOF). Per ritornare alla nostra analogia con un sistema manuale, tutto ciò equivale a staccare la pagina dal blocco, a riporla e ad aggiungere il nome della "pratica" tra quelli riportati sulla targhetta.

Durante l'immissione del listato di un programma BASIC, come abbiamo visto a pag. 558, ogni linea viene "compressa", ricorrendo ad una particolare forma di codifica (i "token"): i comandi e le istruzioni del BASIC occupano, cosìsoltanto un byte. Questo sistema consente di risparmiare una notevole quantità di memoria. Poiché la memoria RAM può contenere codici di tutti i generi (BASIC, codice macchina o ASCII), un comando SAVE può tornare comodo, su alcuni apparecchi, anche per conservare intere "videate" di informazioni o di immagini, composte da codici ASCII. Successivamente, con un comando LOAD, è possibile richiamare tali informazioni per ripresentarle sullo schermo. Inoltre, alcune versioni di BASIC (o particolari comandi del sistema operativo), consentono anche di memorizzare l'intero listato di un programma BASIC nel formato ASCII. Ciò è utile sia che si debba elaborare ulteriormente il listato usando un sofisticato word processor, sia che si voglia trasmettere il programma via modem ad un altro computer.

Lo svantaggio dei file binari è che essi costituiscono un intero blocco di codici: non è possibile richiamarne in memoria solo una parte. Inoltre, la grandezza massima di un file binario è limitata dalla quantità di memoria RAM disponibile.

Nella prossima lezione esamineremo le caratteristiche dei file sequenziali e di quelli ad accesso casuale.

Un po' di parità

Siamo adesso in grado di progettare circuiti logici più complessi

Prima di passare a realizzazioni più impegnative, esaminiamo le caratteristiche di un'altra importante porta logica, quella associata allo OR esclusivo (XOR). A pagina 527 abbiamo già presentato, di sfuggita, questa porta, ma non ne abbiamo ancora fornito né la tavola della verità, né l'equazione Booleana associata. Eccole:

Tavola della verità			Simbolo circuitale
A	B	C	
0	0	0	
0	1	1	
1	0	1	
1	1	0	

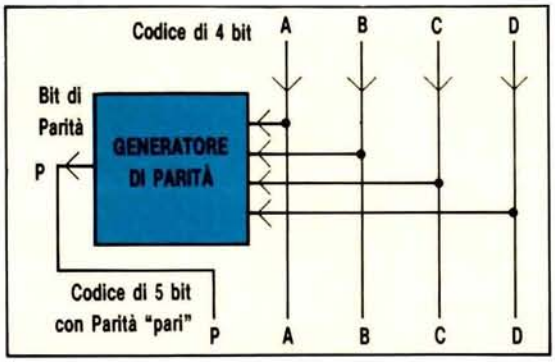
Simbolo Booleano = ⊕

Dalla tavola della verità, possiamo vedere che l'uscita C può essere espressa in due maniere:

a) $C = A \oplus B = \overline{A}.B + A.\overline{B}$
b) $\overline{C} = A \oplus B = \overline{A}.\overline{B} + A.B$

La seconda espressione è stata formulata considerando i casi nei quali C è "non-uno" (ossia: zero). Questa porta logica è di grande utilità nella nostra prima applicazione.

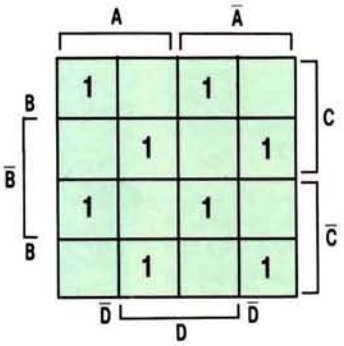
Un generatore di bit di Parità



Il concetto di parità è importantissimo nel progetto di sistemi di trasmissione. Il bit di parità (vedere il precedente diagramma) di un codice binario viene "accodato" ad esso, in modo da ottenere un numero pari di 1. Analogamente, esiste la convenzione "parità dispari", nella quale il numero di 1 nel codice viene, appunto, reso dispari dal bit di parità. Il bit di parità, pari o dispari che sia, serve per verificare che la trasmissione di un dato sia avvenuta correttamente. Il circuito che svilupperemo deve accettare 4 bit in ingresso e generare l'appropriato bit di parità. Con piccole modifiche, il circuito può essere anche impiegato nel controllo di dati in arrivo. La tavola della verità del circuito è riportata a margine, mentre la mappa-k risulta:

A	B	C	D	P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Tavola della verità per il Generatore di Bit di Parità



La disposizione simmetrica degli elementi, sfortunatamente, non permette alcuna semplificazione, in quanto non consente raggruppamenti. L'espressione risultante, per l'uscita P, è:

$P = \overline{A}.\overline{B}.\overline{C}.D + \overline{A}.\overline{B}.C.\overline{D} + \overline{A}.B.\overline{C}.\overline{D} + \overline{A}.B.C.D + A.\overline{B}.\overline{C}.\overline{D} + A.\overline{B}.C.D + A.B.\overline{C}.D + A.B.C.\overline{D}$

Raggruppando i termini in azzurro e quelli in rosso, possiamo semplificare in:

$P = (\overline{A}.\overline{B} + A.B).(\overline{C}.D + C.\overline{D}) + (A.\overline{B} + \overline{A}.B).(\overline{C}.\overline{D} + C.D)$

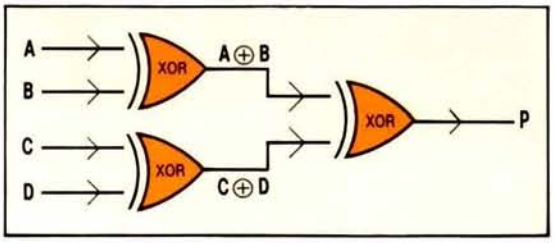
Adesso, ricorrendo alla definizione di porta XOR, data all'inizio della lezione, si può ulteriormente semplificare l'espressione:

$P = \overline{(A \oplus B)}.(C \oplus D) + (A \oplus B).\overline{(C \oplus D)}$

Considerando ciascun termine racchiuso tra parentesi, come un ingresso della porta XOR, otteniamo infine:

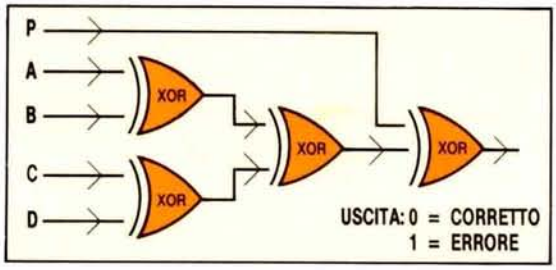
$P = (A \oplus B) \oplus (C \oplus D)$

ed il circuito corrispondente è formato da una serie di XOR collegate l'una all'altra ("in cascata" per dirla in gergo):



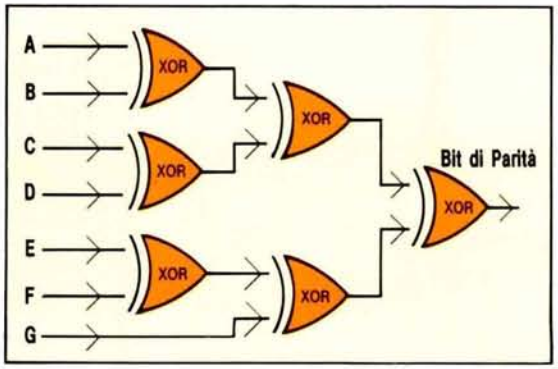
Semplicemente aggiungendo ancora una porta XOR, otteniamo un circuito verificatore di parità per dati in arrivo:

Codice di 5 bit in arrivo



In pratica, la maggior parte dei computer adopera, nelle trasmissioni, il codice internazionale ASCII, ossia un codice ad otto bit, dei quali sette riservati all'informazione ed uno al bit di parità. La realizzazione di un circuito generatore di parità, adatto per il codice ASCII, è molto facile:

Codice di 7 bit



Un codificatore di priorità

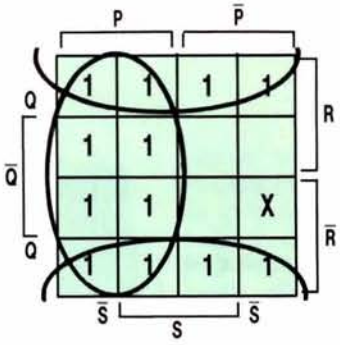
In molti computer viene impiegato un sistema di "priorità delle interruzioni (interrupt)", per controllare il flusso dei dati da e per le periferiche. In tali sistemi, le operazioni della CPU vengono interrotte da un segnale esterno (generato dalla periferica che "chiede attenzione"). Tuttavia, occorre prevedere il caso in cui due periferiche inviino contemporaneamente il segnale d'interruzione. In sostanza si tratta di assegnare una graduazione di "priorità" a ciascuna periferica, in modo che la CPU si dedichi per prima a quella più "importante".

Per un circuito in grado di controllare quattro periferiche sono necessarie due linee d'uscita. Inoltre, useremo una terza linea, che servirà ad indicare che è stato generato un "interrupt". Indichiamo con P, Q, R ed S le quattro periferiche (P ha la priorità massima, S quella minima). Chiamiamo le uscite che identificano la periferica A e B, mentre indichiamo quella concernente il segnale d'interrupt con Z. Possiamo così costruire la tabella della verità del codificatore (con X sono indicate le condizioni "indifferenti"):

P	Q	R	S	A	B	Z
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

Per facilitare la comprensione di tale tabella, si osservi l'ultima riga. In questo caso, P ha generato l'interrupt, e, poiché P è la periferica con maggiore priorità, non ci curiamo di altri interrupt di livello minore. Le tre uscite del codificatore vanno esaminate una alla volta. Iniziando da A, la mappa-k risulta:

Uscita A



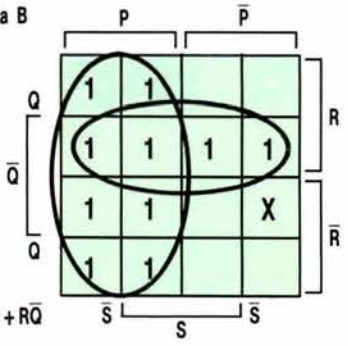
A=P+Q

Il caso "indifferente", per quanto concerne l'uscita di A, è rappresentato con una X, ma i casi "indifferenti" per gli ingressi sono trattati in modo diverso. Se prendiamo il caso nel quale P vale 1, mentre Q, R ed S sono "indifferenti", dobbiamo riempire le caselle della mappa-k dove P vale 1 (otto in tutto). Dalla mappa-k si ottiene l'espressione semplificata:

A = P + Q

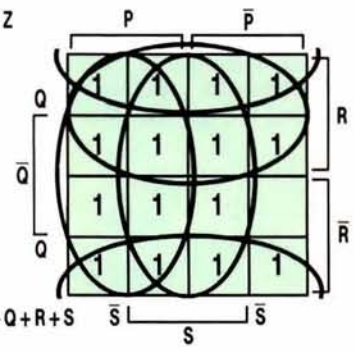
Analogamente, per B e Z, otteniamo le mappe-k:

Uscita B



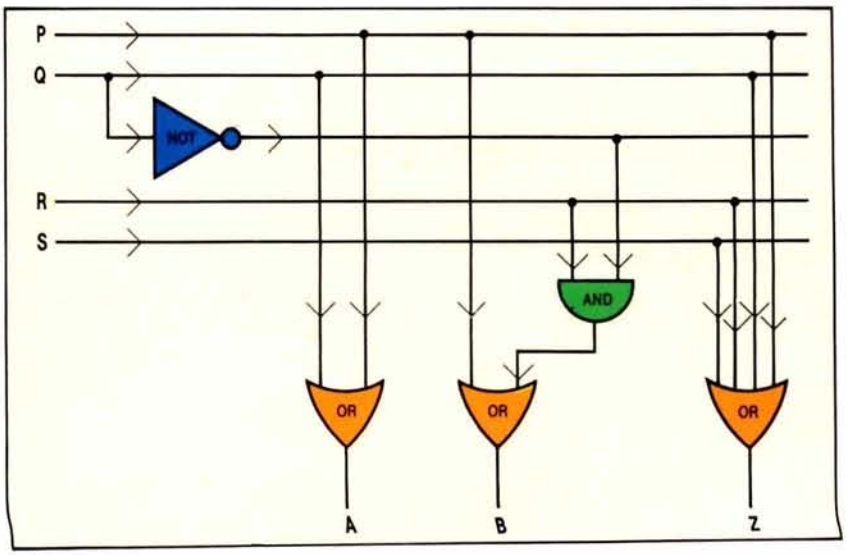
B=P+RQ̄

Uscita Z



Z=P+Q+R+S

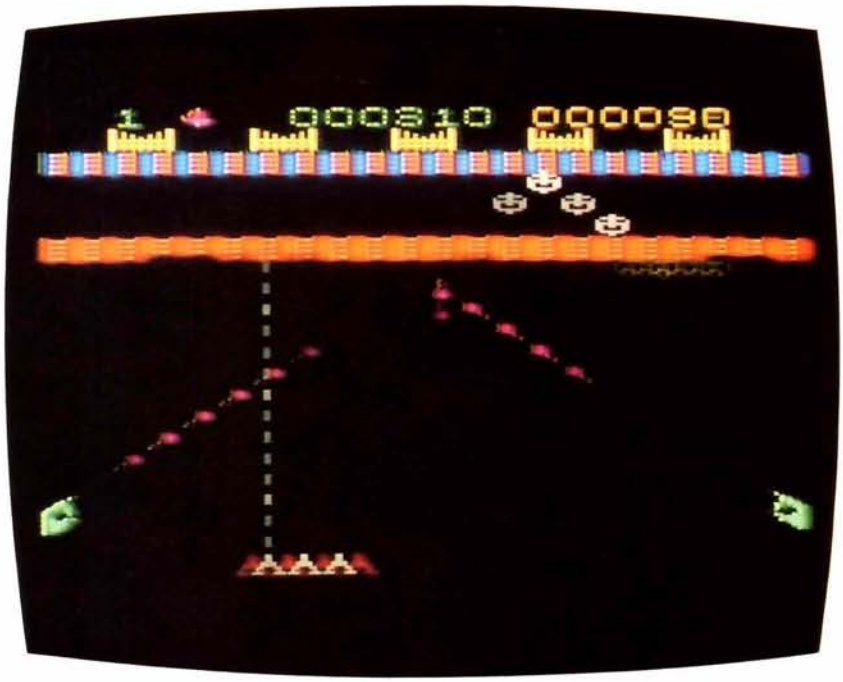
Infine, usando le tre espressioni, si giunge a formulare il circuito:



Software in vetrina

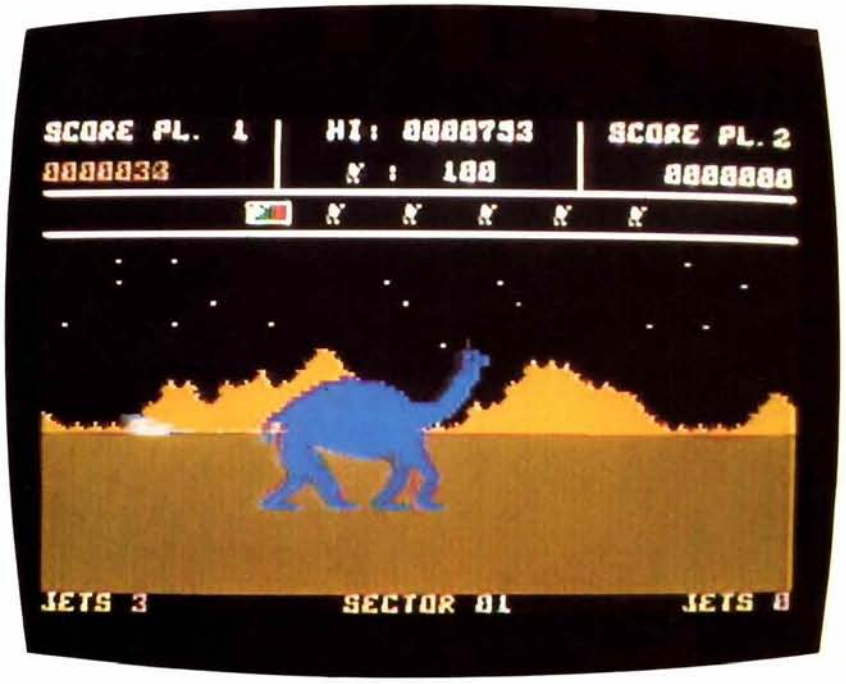
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Gli "adventure" sono dei giochi per home computer di grandissimo successo. Il giocatore si trova ad impersonare il protagonista di fantastiche storie, il cui esito non è mai scontato. Proprio per questa caratteristica di intervento diretto nello svolgimento degli eventi, c'è chi considera questi giochi migliori della passiva lettura di un libro.
Il primo dei giochi qui presentati è appunto del tipo adventure.



Four gates to freedom per il Vic 20

Lo scopo del gioco è quello di liberare un gruppo di scienziati, prigionieri di un esercito di bellicosi personaggi, sul pianeta Xzinos. L'azione si svolge in più fasi: superando le varie "prove", il giocatore raccoglie informazioni sufficienti per avvicinarsi alla meta. L'impostazione di questo gioco è abbastanza interessante, perché si discosta dai normali canoni dei giochi d'avventura.



Attack of Mutant Camels per il Commodore 64

Il giocatore, dalla propria astronave, deve arginare l'attacco di una schiera di "cammelli mutanti". Facile a dirsi, ma, dato che i cammelli non si lasciano abbattere tanto facilmente, l'impresa è ardua. Concettualmente semplice, questo gioco è irresistibile anche selezionando il livello per principianti.

Novità Atari

Due nuovi modelli della nota casa statunitense con caratteristiche molto competitive

La Atari è una delle più note ditte produttrici di home computer. Indirizzatasi fin dagli inizi verso il settore dei videogiochi, detiene ancora oggi uno dei primi posti nella commercializzazione di piccoli apparecchi dotati di un'ottima grafica.
Dovendo competere, però, con una concorrenza veramente agguerrita, la Atari ha ritenuto opportuno riprogettare la propria linea di apparecchi, introducendo due nuovi modelli: 600XL e 800XL. Essendo compatibili con i precedenti modelli 400 ed 800, la quantità di software disponibile è molto ampia: dai giochi alle applicazioni commerciali.
Fortunatamente, per ambedue gli apparecchi è stata adottata la tastiera del precedente modello 800, in quanto quella del 400, del tipo "a sfioramento", risulta inadeguata per applicazioni che vadano al di là dei semplici giochi. La tastiera dell'Atari 800, al contrario, è una delle migliori del settore.
In entrambi gli apparecchi la tastiera ha 62 tasti, che includono quattro tasti di funzione: START, SELECT, OPTION e RESET, situati nella parte destra della tastiera. Esiste anche un tasto HELP, adoperato da alcuni programmi per visualizzare informazioni di assistenza sullo schermo. Oltre all'intero gruppo di caratteri ASCII, ci sono 29 tasti con simboli grafici. Una caratteristica fastidiosa, ereditata dall'Atari 800, riguarda i tasti per il controllo del cursore: per usarli occorre premere, simultaneamente, anche il tasto CONTROL. A parte questa stranezza, del resto abbastanza trascurabile, la tastiera risulta molto comoda.
L'Atari 400 possiede 16 Kbyte, mentre l'Atari 800 ne ha 48, ma entrambi sono dotati di un connettore per espansioni, raggiungibile scoperchiando l'apparecchio. Nei modelli XL, al contrario, il connettore è collocato esternamente all'involucro; la differenza principale è la quantità di memoria di cui sono dotati al momento dell'acquisto: il 600XL ha 16 Kbyte, espandibili a 64, mentre l'Atari 800XL viene venduto con 64 Kbyte.
I due modelli XL sono dotati di un'interfaccia, chiamata "Expander", che assomiglia ad un connettore per "bus", adatta al collegamento con un'ampia gamma di accessori. È presente anche uno slot per cartucce ROM. Il precedente Atari 800 era fornito di due slot per cartucce, ma, poiché non esisteva, in pratica, alcun programma che sfruttasse il secondo slot, questo è stato soppresso nella linea XL. Anche il numero di joystick applicabili (4 nella vecchia serie) è stato ridotto a due.



IAN McKINNELL

Suoni e Luci

Tutti i computer Atari possono essere collegati a normali televisori domestici e tutti, salvo il 400, possono essere collegati anche a speciali monitor. In tutti i casi, la visualizzazione è di buona qualità: i caratteri hanno una forma ben leggibile, ed il contrasto tra scritta e sfondo è abbastanza gradevole. Tra le varie combinazioni di colore, per lo scritto e per lo sfondo, quella normale prevede i testi in bianco su sfondo blu. Lo schermo è suddiviso in 24 linee di testo, di 40 caratteri ciascuna.
I computer Atari sono stati tra i primi ad usare la grafica a sprite. Tale grafica è stata soprannominata dalla Atari "Grafica Giocatore-Missile" ed è controllata mediante uno speciale chip: il GTIA. I giocatori sono oggetti creati mediante raggruppamenti di pixel: una volta definita la sagoma di un'immagine, i valori di ciascun pixel vengono depositati in una particolare zona della memoria, chiamata "tabella delle forme". Si possono creare fino a quattro giocatori, ad ognuno dei quali è associato un missile. Il giocatore può essere colorato in più tinte e la sua visualizzazione è controllata modificando i valori nella "ta-

I gemelli Atari

Gli home computer 600XL ed 800XL si assomigliano moltissimo, ma il 600XL possiede solo 16 Kbyte di memoria, mentre l'800XL ha ben 64 Kbyte. Entrambi gli apparecchi hanno una tastiera di buona qualità ed un'ottima grafica. Essendo compatibili con i precedenti modelli Atari, la gamma di software disponibile è molto ampia



L'unità a dischetti

Un accessorio senz'altro molto utile per l'800XL è il disk drive, che ha una capacità di 127 Kbyte. Il 600XL può utilizzare i dischetti soltanto se ne viene espansa la memoria. Un gran numero di programmi è disponibile soltanto nella versione su disco

bella delle forme". La "Grafica Giocatore-Missile" non è estremamente facile da usare, ma gli effetti che si ottengono sono spettacolari.
I nuovi apparecchi Atari possiedono ben 11 modi grafici diversi e 256 colori (per esser più precisi: 16 colori con 16 sfumature). Il numero di colori visualizzabili simultaneamente, però, varia a seconda del modo grafico impiegato, in quanto dipende dalla quantità di memoria libera. Più alta è la risoluzione, minore è il numero di colori. La massima risoluzione ottenibile sul 600XL e sull'800XL è di 320 × 192 pixel.
Anche la generazione di effetti audio è controllata da uno speciale chip. Sono disponibili quattro "voci" indipendenti, ognuna delle quali ha un'estensione di 3 ottave e mezzo. In BASIC, le voci si controllano col comando SOUND, oppure depositando (mediante delle POKE) opportuni valori negli appositi registri di memoria. È possibile intervenire su diverse caratteristiche del suono: oscillazione, distorsione, volume e altezza. Se viene adoperato il comando SOUND, è possibile avviare una sola voce per volta, creando fastidiosi ritardi nelle armonizzazioni. Gli effetti migliori si ottengono mediante piccoli sottoprogrammi in codice macchina, oppure usando delle POKE.
Contrariamente alla vecchia serie, gli Atari della linea XL sono dotati di un BASIC incorporato. La versione di BASIC non è delle migliori (mancano istruzioni quali CIRCLE, PRINT@ e PRINT USING ed altre utili funzioni), tuttavia è possibile acquistare una cartuccia ROM con il BASIC Microsoft ed un BASIC esteso. Anche tutta la linea di periferiche Atari è stata riprogettata, aggiungendo maggiori possibilità di espansione. La più utile delle periferiche è forse la scatola d'espansione, che si collega all'interfaccia Expander. Il "box" d'espansione può contenere più schede d'interfaccia, due porte seriali RS232 ed un bus parallelo. La Atari ha progettato anche una scheda dotata di un microprocessore Z80, del sistema operativo CP/M e di un modulo che consente la visualizzazione su 80 colonne.



La stampante Atari 1027
Il funzionamento si basa sull'impiego di una testina, come quella di molte macchine da scrivere elettriche. I caratteri sono di buona qualità, ma il funzionamento è molto lento.
Esistono altre due stampanti Atari: una con penne a sfera a colori, l'altra è una normale stampante ad aghi, che, per qualità di stampa è inferiore alle altre, ma è molto più veloce. Questi tipi di stampante sono gli unici utilizzabili dalla serie XL, mancando un'interfaccia standard per stampante





Porta per cartucce
La gamma XL dispone di un solo slot per cartucce ROM

Chip per la grafica
Le ottime funzioni grafiche degli Atari sono ottenute mediante due speciali chip, chiamati ANTIC e GTIA



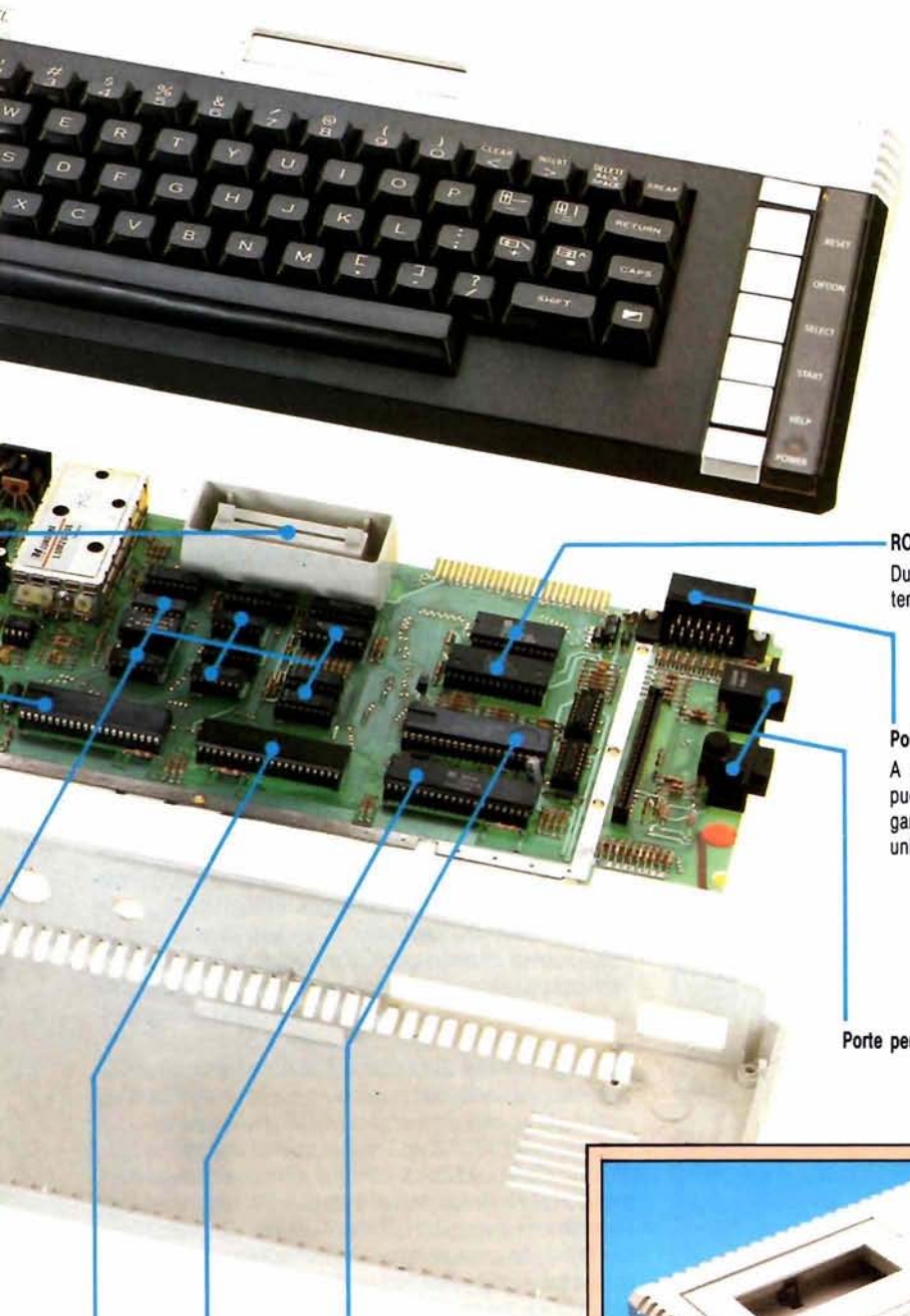
RAM
Questi chip forniscono i 16 Kbyte di RAM





La sfera e la bacchetta
Per il controllo dei giochi sono disponibili due accessori Atari: uno è il normale joystick (i joystick Atari sono ormai divenuti uno standard industriale), l'altro è la "sfera", che permette un controllo molto dinamico

La linea Atari XL sembra destinata al successo per la buona qualità di entrambi i modelli. Anche la notevole quantità di software, direttamente ereditata dai precedenti modelli, può contribuire al successo commerciale degli XL.



ROM
Due chip di memoria ROM contengono il BASIC

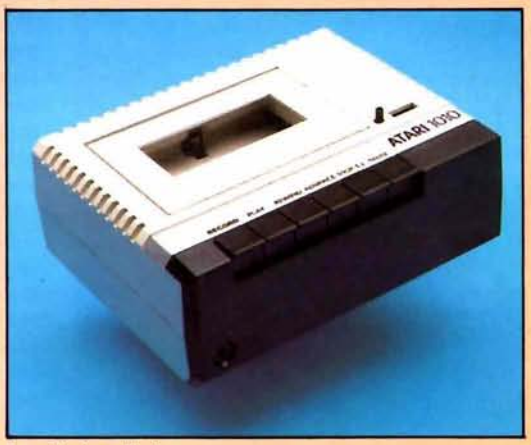
Porta per periferiche
A questo connettore a 13 poli può essere collegata un'ampia gamma di periferiche (stampanti, unità a dischi, ecc.)

Porte per joystick

Effetti audio
Questo speciale chip, chiamato POKEY, gestisce la generazione di effetti sonori

Chip di I/O
Un 6520 si occupa del controllo dei segnali in ingresso ed in uscita

CPU
Gli Atari sono progettati usando il noto chip 6502



Il registratore Atari
Gli Atari possono essere collegati soltanto ad un apposito registratore a cassette della stessa marca, che benché piuttosto caro, offre alcuni apprezzabili vantaggi. Infatti, essendo progettato per gli Atari, è molto più affidabile; inoltre, vengono usate due tracce: una serve per la memorizzazione dei programmi, l'altra per la registrazione di suoni. Ciò risulta conveniente se si desidera aggiungere un commento parlato ai programmi

ATARI 600XL ATARI 800XL

DIMENSIONI

mm. 380 x 170 x 40 (600XL)
mm. 380 x 220 x 40 (800XL)

CPU

6502 a 2 MHz

MEMORIA

16 (o 64) Kbyte di RAM 24 Kbyte di ROM contengono il BASIC

VIDEO

24 linee di 40 caratteri, o fino a 320 x 192 punti per la grafica ad alta risoluzione con sprite. 16 colori disponibili in 16 sfumature

INTERFACCE

Due porte per joystick, porta per periferiche, porta per espansioni, slot per cartucce

LINGUAGGI DISPONIBILI

BASIC, FORTH, LOGO, PILOT e ASSEMBLY 6502

DOCUMENTAZIONE

I manuali della Atari non sono mai stati di altissimo livello, soprattutto per la parte che concerne gli aspetti più tecnici. Ciò può derivare dal fatto di considerare i propri apparecchi più destinati al gioco, che non ad utenti desiderosi di avventurarsi nella programmazione. Esistono, tuttavia, numerose ed ottime pubblicazioni che forniscono utili consigli e suggerimenti

PRO

Sugli home computer Atari girano i migliori programmi di gioco disponibili sul mercato, che sfruttano appieno la fenomenale grafica di questi apparecchi

CONTRO

Principalmente, il costo. Su di esso incide il fatto che non è possibile usare un normale registratore a cassette e, inoltre, gli alti prezzi del software. L'uso della grafica è un po' ostico, se paragonato a quello disponibile su altri apparecchi

Controllo sulle scorte

Altre caratteristiche essenziali nei programmi per la gestione di un magazzino

Nella scorsa lezione abbiamo esaminato alcuni aspetti generali riguardo alla gestione computerizzata di un magazzino. In particolare, abbiamo visto come l'uso di un'opportuna codifica faciliti il richiamo delle informazioni relative a ciascun articolo. Tuttavia, un buon sistema computerizzato deve essere molto più duttile ed offrire anche altre forme di accesso, utili alla gestione aziendale.

L'ammontare delle informazioni che possiamo archiviare, ovviamente, dipende dal tipo di memoria di massa disponibile (nastri o dischetti) e dalla dimensione della memoria RAM del proprio apparecchio.

Così come avviene nei "database", anche nei programmi per la gestione di magazzino occorre trovare un compromesso tra consumo di memoria ed il numero di "campi" che compongono ciascun record. Nella maggior parte delle applicazioni, bastano sette diversi campi: codice dell'articolo, descrizione, livello di riordino, prezzo d'acquisto, prezzo di vendita e unità di misura.

Tutto sotto controllo

Un package di enorme successo in Gran Bretagna è lo "Stock Recording System", scritto per il Dragon 64. Si tratta di un gruppo di programmi che girano con lo speciale sistema operativo OS9, un derivato del sistema operativo UNIX



IAN McKINNELL

Alcune grandezze possono essere "derivate". Ad esempio, una misura del margine di profitto globale può esser desunta sottraendo dal prezzo di vendita quello di costo. L'unità di misura è un elemento comodo per determinare, ad esempio, il peso o l'ingombro dell'articolo, quando occorre preparare una spedizione. Inoltre, per alcuni articoli può essere conveniente usare il peso, mentre per altri basterà indicare "scatola".

Una delle principali caratteristiche, nel progetto di archivio di magazzino, è che i dati sono per lo più "dinamici" (ossia variano continuamente). Nella contabilità, al contrario, gran parte delle informazioni sono di carattere "statico".

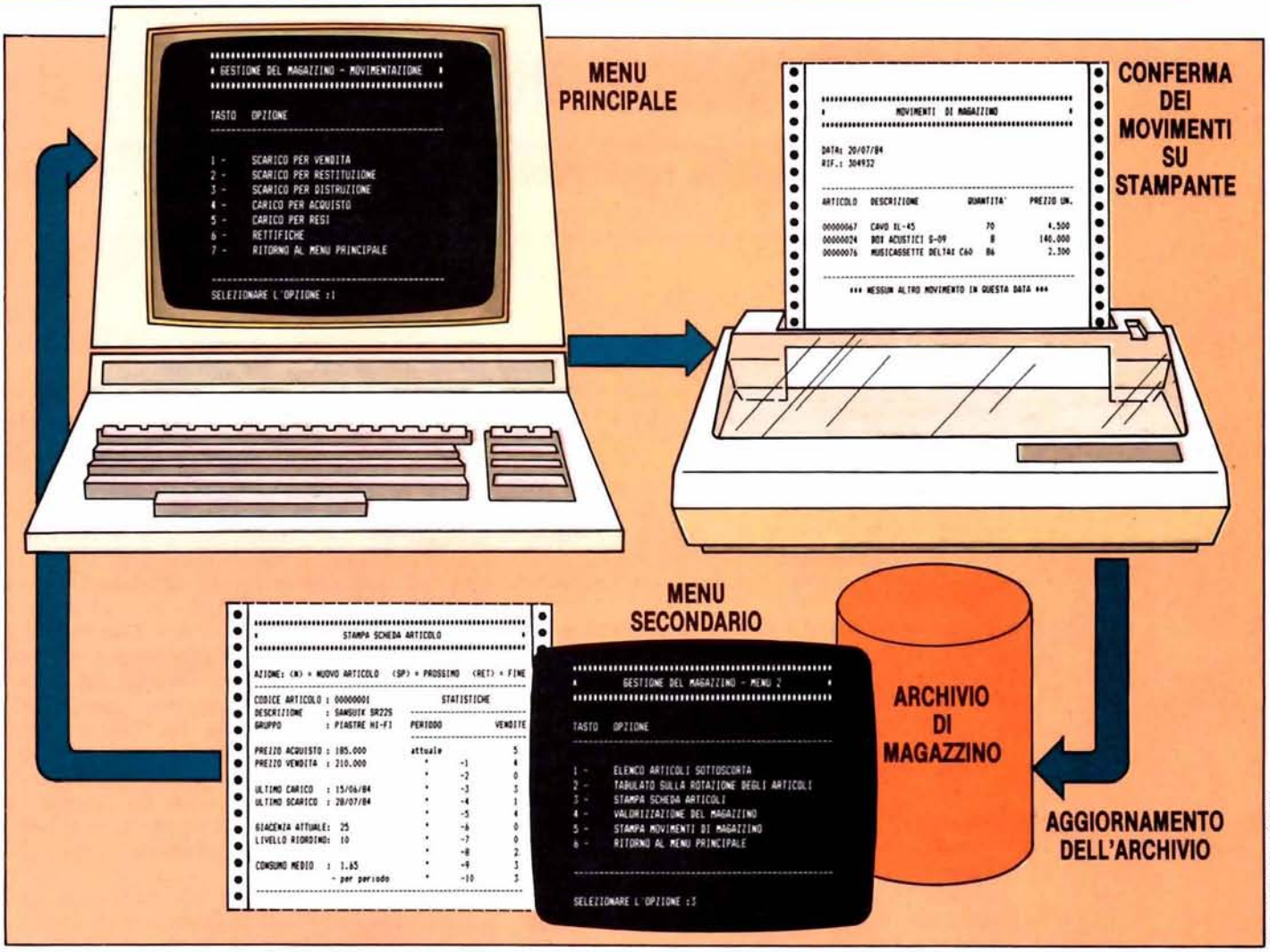
Infatti, sebbene alcune parti essenziali di ciascun record rimangano inalterate per lungo tempo (il codice e la descrizione dell'articolo), altre variano di continuo in base ai movimenti ed alla rotazione delle scorte.

Se facciamo un confronto con un sistema contabile, può capitare di dover modificare l'intestazione di un conto relativo ad un cliente o ad un fornitore (cambia il numero di telefono o l'indirizzo). Per apportare queste variazioni, che non possono essere "calcolate" dal computer, esiste un apposito programma.

Nella gestione di un magazzino, invece, sia il prezzo di acquisto che quello di vendita variano ogni volta che viene consegnata una nuova partita. Una scelta molto logica, quindi, consiste nel far dedurre automaticamente questi due valori, in base alla fattura del fornitore, quando i nuovi articoli vengono caricati in magazzino, anziché conservarli staticamente assieme alla descrizione. Ovviamente, occorre sempre un programma che consenta di variare manualmente una o più voci di ciascun articolo.

Vediamo, comunque, di procedere con ordine. Si parte innanzitutto da un'analisi di ciò che rappresenta l'input (i dati forniti dall'operatore) e l'output (i risultati delle informazioni elaborate) del sistema. I tre elementi più importanti sono: la procedura per l'immissione dei movimenti di magazzino, i particolari di ciascuna transazione, ed il sistema di interrogazione.

Prima di descrivere la procedura seguita dal programma per la movimentazione di una vendita (lo scarico da magazzino di un bene venduto), è opportuno soffermarsi su di un'importante qualità, che distingue un programma di buona qualità da uno mediocre: la verifica dei dati immessi. Nei sistemi più



sofisticati, ossia di migliore qualità, ogni dato immesso dall'operatore viene attentamente controllato sia nel tipo (non devono esserci lettere in un codice o in un valore numerico), che, quando possibile, nella validità numerica (un prezzo negativo, ad esempio). In molti sistemi l'utente viene guidato, nell'immissione dei dati, da appropriati messaggi.

Ma sono utili anche altri tipi di controllo. Il programma, ad esempio, dovrebbe impedire all'operatore l'immissione di un movimento di carico o di scarico, se non è stato indicato il numero di riferimento della bolla di accompagnamento obbligatoria; oppure dovrebbe verificare l'esistenza, in magazzino, dell'articolo specificato, prima di permettere alcun movimento su di esso e magari visualizzarne la descrizione per conferma.

La gestione di un magazzino si presta molto bene alla formulazione di statistiche e di tabulati di vario genere, considerando i diversi tipi di informazione presenti negli archivi. Il sottomenu relativo alla procedura d'interrogazione, per esempio, potrebbe contenere le seguenti opzioni:

STAMPA DELLE GIACENZE ATTUALI, TABULATO ARTICOLI SOTTOSCORTA, ELENCO CRONOLOGICO DEI MOVIMENTI, ELENCO DEI MOVIMENTI PER CODICE ARTICOLO, ELENCO ARTICOLI CON BASSA ROTAZIONE, ecc.

Una transazione di vendita influenza il contenuto di tutte le stampe appena citate. Se, infatti, chiediamo un tabulato degli articoli "sottoscorta" (presenti in magazzino, ma in quantità inferiore a quella minima da noi fissata), essa deve riflettere la situazione aggiornata fino all'ultimo carico o scarico avvenuto. Il programma che gestisce la movimentazione deve quindi operare un controllo sulle giacenze e segnalare quelle che cadono al di sotto del livello minimo di riordino.

La stampa di un inventario di magazzino, limitato agli articoli che abbiano una rotazione inferiore ad un minimo prefissato, può essere molto utile per decidere se convenga o no ordinare nuovamente un particolare articolo.

Inoltre, un buon package di magazzino deve poter fornire in qualsiasi momento una valorizzazione delle giacenze. Spesso non ci si limita alla sola valutazione LIFO (= Last In First Out), ma anche a "costo medio" e a "ultimo costo".

Infine, anche per la valutazione del livello di riordino esistono metodi più o meno sofisticati: da quello empirico, basato sull'esperienza o sulla pratica (ad esempio sul quantitativo minimo di ordinazione accettato dal fornitore) a quelli quasi totalmente automatici, basati sull'impiego di complesse formule, che consentono di ottimizzare le scorte e, di conseguenza, il guadagno dell'impresa.

Gestione delle scorte

Un sistema ben impostato deve effettuare dei continui controlli sulle giacenze attuali e sui livelli degli articoli, durante l'immissione dei movimenti di magazzino. Oltre ad evitare banali errori d'immissione (un articolo che non esiste), ciò consente di ottenere, in qualsiasi momento, una situazione in "tempo reale" del magazzino. Altre opzioni riguardano la stampa di tabulati, molto utili quando si tratta di riordinare le scorte

Facciamo il punto

Un conciso panorama sugli argomenti esaminati, prima di procedere oltre

Abbiamo iniziato questa sezione del corso esaminando gli attrezzi necessari (vedere a pagina 524) per apportare modifiche o riparazioni alle apparecchiature. Certo, possiamo usare un normale cacciavite piatto per le viti "a croce", ma è probabile che quest'ultime si danneggino; possiamo usare una pinza per svitare un dado, ma una chiave dell'apposita misura rende il lavoro più semplice e sicuro. A lungo andare, risulta più economico spendere qualcosa in più, ma avere gli strumenti adatti.

Lo stesso discorso vale per la saldatura dei componenti. Una saldatura fatta bene, con un buon saldatore, resiste più a lungo e raramente è causa di fastidiosi guasti. Ricordarsi di appoggiare lo stagno sul filo da saldare e non direttamente sulla punta del saldatore: questa pratica garantisce una maggior durata della punta ed una saldatura più resistente.

La giuntura va scaldata bene ed il saldatore rimosso soltanto quando lo stagno è ben fuso su tutte le parti di essa. Poche regole, ma se osservate bene assicurano il successo. Per quanto concerne la dissaldatura, per la sostituzione di componenti, vedere quanto detto a pagina 548.

Abbiamo trattato i concetti fondamentali dell'elettronica digitale (sia in *Laboratorio* che ne *Le chiavi dell'informatica*), introducendo i principali componenti ed esaminando come agiscono. La resistenza è il componente elettronico più semplice ed anche quello più frequentemente usato: basta dare un'occhiata all'interno di un qualsiasi apparecchio elettronico.

Anche i condensatori sono molto usati: nei computer servono per filtrare i segnali, in modo da rimuovere disturbi e interferenze. Ogni volta che un segnale viene cambiato, amplificato o altrimenti usato, inevitabilmente si introducono dei disturbi; i condensatori servono appunto per ripulire convenientemente i vari segnali di un circuito.

Comunque, il componente più importante che abbiamo trattato è il transistor, essenziale in tutti i circuiti dove occorra manipolare in qualche modo dei segnali elettrici. I transistor servono come amplificatori, ma nei computer la loro funzione principale è quella di operare come interruttori. È ciò che accade all'interno delle porte logiche.

A scopo sperimentale, abbiamo costruito le tre porte più semplici: la NOT, la OR e la AND, impiegando solo qualche piccolo componente (anche se al loro interno, i circuiti integrati dei computer sono molto più complessi dei semplici transistor).

Le porte logiche non sono particolarmente utili prese a sé, ma possono essere combinate in circuiti logici più complessi, in grado di elaborare dei dati. Nell'ultima lezione, ad esempio, abbiamo realizzato un semisommatore, sfruttando le porte logiche di due circuiti integrati.

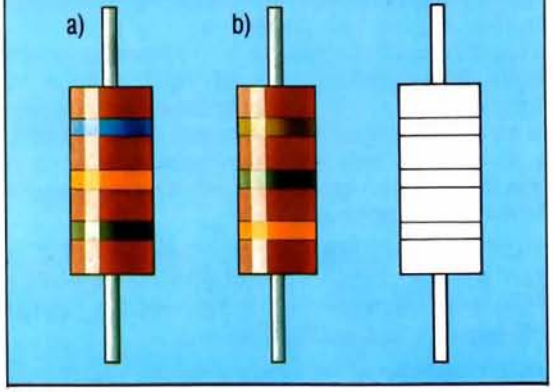
I circuiti integrati sono i componenti elettronici più complessi: quelli adoperati per i nostri esperimenti appartengono alla categoria dei TTL (Logica Transistor Transistor) e sono realizzati con una tecnologia a piccola scala d'integrazione (ogni chip contiene soltanto l'equivalente di qualche transistor). Ma, col progredire della tecnica, è stato possibile inserire un numero sempre maggiore di transistor in un solo chip, come quelli adoperati nei primi computer. La tecnologia a media scala d'integrazione ha consentito la realizzazione di circuiti integrati contenenti migliaia di transistor.

Infine, l'integrazione a larga (LSI) ed a larghissima scala (VLSI) ha permesso di costruire chip che contengono un'intera CPU. Siamo giunti ad un punto tale che complicatissimi circuiti, composti da decine di migliaia di componenti, sono racchiusi in una superficie di pochi millimetri quadri.

Avendo imparato tutto ciò, prima di avventurarsi in nuove imprese, suggeriamo di realizzare i circuiti riportati nella pagina a fianco. Non sono complessi e richiedono ben pochi componenti: solo qualche transistor e qualche circuito integrato.

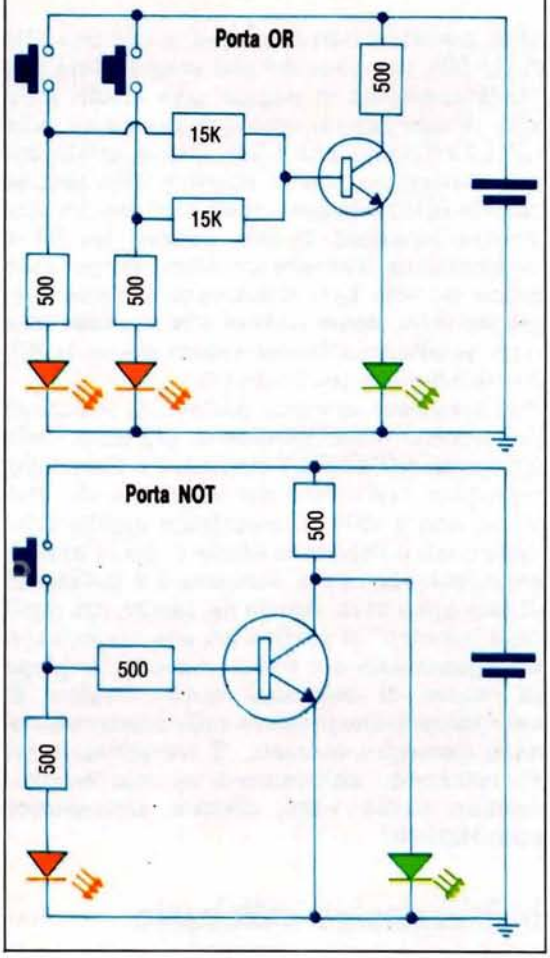
1) LE RESISTENZE

Il valore delle resistenze è indicato mediante delle piccole fascette colorate. Qual è il valore delle prime due resistenze? Quali colori troveremmo su di una resistenza da 150 Ohm?



2) PORTA LOGICA NOR

A pagina 625 abbiamo costruito, con dei transistor, una porta NOT, una OR ed una AND. Come possiamo realizzare, in modo analogo, una porta NOR? Riportiamo qui di seguito i circuiti OR e NOT. Una soluzione può essere quella di collegarli assieme, ma, studiando attentamente la porta NOR, possiamo trovare una scorciatoia...



3) CONVERTITORE DA DECIMALE A BINARIO

Si costruisca un circuito che converta da decimale a binario. Per non complicarlo troppo, limitiamo i valori in ingresso a numeri binari di due bit (da zero a tre in decimale). Il circuito dovrebbe avere quattro interruttori, contrassegnati da 0 a 3. Quando se ne preme uno, l'equivalente numero binario deve apparire dalla combinazione di due LED. Una tavola della verità, parziale, è la seguente:

INTERR. ZERO	INTERR. UNO	INTERR. DUE	INTERR. TRE	BIT ALTO	BIT BASSO
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

4) BCD O NON BCD?

Nella sezione "Le chiavi dell'informatica" abbiamo già studiato il sistema di numerazione decimale codificata in binario (più brevemente: BCD). Il BCD è un sistema intermedio tra quello decimale e quello binario: ogni cifra decimale viene convertita in binario ed i quattro bit risultanti vengono depositati in un mezzo byte. Per esempio, 53 decimale è codificato in 01010011; 5 è rappresentato da 0101 e 3 da 0011. Ciò significa che viene adoperato soltanto l'arco di valori binari compresi tra 0000 e 1001 (ossia da 0 a 9 in decimale). I valori da 1001 a 1111 non sono consentiti nel codice BCD.
Si realizzi un circuito in grado di verificare se il numero binario ai suoi quattro ingressi è un codice BCD valido oppure no. Il circuito dovrebbe avere quattro pulsanti (B0, B1, B2 e B3) che servono ad immettere il codice binario. Le uscite sono costituite da due LED: uno verde che si illumina se il codice è valido, uno rosso che si accende in caso contrario. Ecco la tavola della verità:

Equivalente Decimale					BCD valido	BCD non valido
0	0	0	0	0	1	0
1	0	0	0	1	1	0
2	0	0	1	0	1	0
3	0	0	1	1	1	0
4	0	1	0	0	1	0
5	0	1	0	1	1	0
6	0	1	1	0	1	0
7	0	1	1	1	1	0
8	1	0	0	0	1	0
9	1	0	0	1	1	0
10	1	0	1	0	0	1
11	1	0	1	1	0	1
12	1	1	0	0	0	1
13	1	1	0	1	0	1
14	1	1	1	0	0	1
15	1	1	1	1	0	1

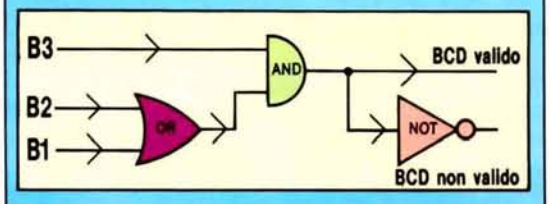
Da questa tavola della verità, possiamo vedere che il segnale "valido" è dato da $\overline{B3} + \overline{B2}.\overline{B1}$, mentre il segnale "non valido" è il NOT di tale espressione:

$$\overline{\overline{B3} + \overline{B2}.\overline{B1}}$$

che si semplifica in:

$$\overline{B3} \cdot \overline{\overline{B2}.\overline{B1}}$$
$$B3.(B2 + B1)$$

Cosicché il circuito che desideriamo ottenere può essere semplificato, necessitando di soli tre ingressi. Ecco un diagramma logico per tale circuito:



Le risposte sono nella prossima lezione di "Laboratorio".

Modi d'indirizzamento

L'accesso ai dati, nel linguaggio ASSEMBLY, può avvenire secondo diversi metodi

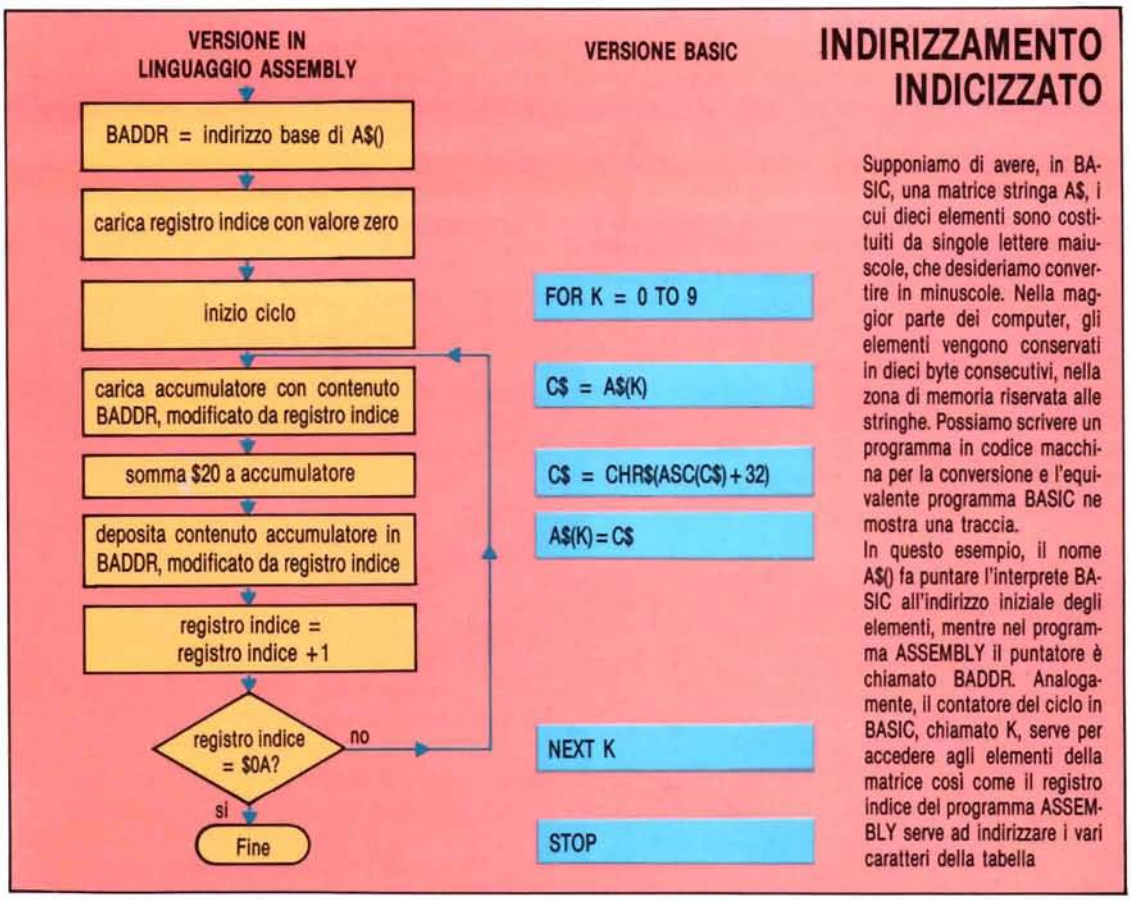
Qualsiasi istruzione del linguaggio ASSEMBLY si riferisce, esplicitamente o implicitamente, al contenuto della memoria e, poiché ciascun byte si differenzia dagli altri in base al proprio indirizzo, ogni istruzione dell'ASSEMBLY deve dunque riferirsi ad almeno un indirizzo. Tale riferimento può essere del tutto palese, come in LDA $E349 (che significa: carica l'accumulatore con il contenuto dell'indirizzo $E349): in questo caso sia l'accumulatore, sia l'indirizzo, sono citati in modo non ambiguo ed il tipo di rapporto tra queste due entità è molto chiaro.
In altri casi, però, il riferimento all'indirizzo può essere del tutto oscuro: RET, che significa "ritorna al programma principale dopo una chiamata ad un sottoprogramma", ne è un buon esempio. Questa istruzione pare non riferirsi ad un indirizzo, ma se completiamo la spiegazione con "riprendendo l'esecuzione dalla locazione successiva a quella della chiamata", allora il riferimento è più ovvio. I due esempi citati mostrano i due estremi di ciò che viene definito "metodo d'indirizzamento".
Fino ad ora, abbiamo incontrato due metodi d'indirizzamento: quello "immediato" (come in LD A,$45 oppure in ADC #31), e quello "diretto assoluto", come in STA $58A7 oppure in LD ($696C),A. Ma quelli esposti non sono i soli metodi d'indirizzamento, ne esistono altri.

Indirizzamento in pagina zero

Conosciuto anche come "indirizzamento abbreviato", questo metodo è spesso usato quando un'istruzione si riferisce ad un indirizzo compreso tra $0000 e $00FF. Siccome tutti gli indirizzi di questo arco di valori hanno il byte "alto" pari a $00, essi vengono indicati come appartenenti alla "pagina zero". Nelle relative istruzioni, quindi, basta usare due soli byte: uno per l'istruzione ed uno per il valore basso dell'indirizzo. Quando la CPU incontra un indirizzo composto da un solo byte, dà per scontato che esso si trovi in pagina zero. Il vantaggio derivante da questo metodo è una maggiore rapidità d'esecuzione.
I due microprocessori Z80 e 6502 differiscono notevolmente nell'uso dell'indirizzamento in pagina zero. Nel linguaggio ASSEMBLY 6502, qualsiasi istruzione che si riferisca alla RAM (LDA, per esempio) può essere usata con l'indirizzamento in pagina zero e tutti i 256 byte di tale pagina sono a disposizione della CPU. Nell'ASSEMBLY Z80, invece, esiste una sola istruzione che si riferisce alla pagina zero (la RST) usata per "chiamare" una tra otto precise locazioni di tale pagina. La RST è un'istruzione talmente specifica, che per essa basta un solo byte (l'indirizzo è parte integrante dello stesso codice) e la sua esecuzione è rapidissima. Incontreremo spesso la RST, per queste sue particolarità.
Può sembrare assurdo parlare di istruzioni più o meno veloci quando la più lenta delle istruzioni ASSEMBLY richiede un tempo d'esecuzione nell'ordine dei microsecondi. Tuttavia, non è difficile incontrare applicazioni nelle quali il risparmio anche di pochi millisecondi può segnare il successo o il fallimento di una procedura. Anche nei giochi, nei quali ogni "quadro" di grafica ad alta risoluzione, tridimensionale e a colori, richiede l'impiego di milioni di istruzioni, poche frazioni di secondo possono incidere sulla scorrevolezza delle immagini animate. Il risparmio di un microsecondo, all'interno di un ciclo iterativo ripetuto 64.000 volte, diventa decisamente apprezzabile!

Indirizzamento indicizzato

Il modo "indicizzato" è di vitale importanza nella programmazione in linguaggio ASSEMBLY, poiché permette la costruzione di strutture a forma di matrice e di tabelle, utilissime nella maggior parte dei programmi. Questo sistema offre molta elasticità nell'elaborazione dei dati.
Gli elementi essenziali, nell'indirizzamento indicizzato, sono l'indirizzo "base" e l'"indice". Supponiamo di voler realizzare una tabella contenente i codici ASCII in una serie di byte adiacenti. L'indirizzo "base" di questa tabella corrisponde a quello del primo byte. Fissato ciò, possiamo accedere a qualsiasi codice della tabella, ricavandone l'indirizzo dalla sua posizione rispetto alla base. Ne risulta che il primo byte è in posizione zero, il secondo in posizione uno, e via dicendo. La posizione relativa alla base è chiamata "indice" e l'indirizzo assoluto di memoria di cia-



LIZ DIXON

scun elemento si calcola sommando l'indice all'indirizzo base. Se realizziamo un ciclo iterativo in ASSEMBLY, nel quale il contatore corrisponde all'indice della tabella, allora possiamo accedere, in sequenza, a tutti gli elementi. Ciò equivale, in BASIC, ad un ciclo FOR...NEXT per la lettura di una matrice.
Una volta ancora, esistono notevoli diversità tra lo Z80 e il 6502, anche nell'indirizzamento indicizzato. Il 6502 possiede due registri di un solo byte, chiamati X e Y, che possono essere usati come indici di una tabella. Con un solo byte, tuttavia, la lunghezza della tabella è limitata ad un massimo di 256 byte. Nello Z80, invece, esistono due registri, chiamati IX e IY, a due byte, nei quali viene depositato l'indirizzo base. Esistono poi speciali istruzioni per accedere agli elementi della tabella, specificando semplicemente uno "scarto" dall'indirizzo base. Dato che si tratta di registri a due byte, non vi sono limiti alla lunghezza della tabella.

Indirizzamento indiretto

Questo metodo si basa sull'impiego di "puntatori", un concetto già incontrato precedentemente nel corso, quando abbiamo parlato dei confini fluttuanti nelle mappe di memoria (vedere a pagina 538). Per meglio comprendere come funzioni questo sistema, ricorriamo ad un esempio. Supponiamo di dover prendere un treno: giunti alla stazione, per sapere verso quale binario incamminarsi, consultiamo il tabellone degli orari. Il numero del binario, in questo caso, rappresenta il "puntatore". Se per un motivo qualsiasi il binario venisse cambiato (a seguito di un ritardo del treno, ad esempio), basta tener d'occhio il tabellone per avere il nuovo "puntatore".
Col metodo dell'indirizzamento indiretto è possibile usare istruzioni nelle quali è specificato l'indirizzo della locazione che contiene un puntatore e quindi operare sul contenuto della cella di memoria "puntata". I vantaggi di questo metodo sono enormi, specialmente se viene usato congiuntamente all'indirizzo



IAN McKINNELL

Un "puntatore" ai binari

Un esempio molto pratico dell'uso di puntatori, nella vita quotidiana, è offerto dai tabelloni delle stazioni, che riportano informazioni sui treni in partenza e in arrivo.
Per sapere da quale binario parte un treno, consultiamo il tabellone. In pratica, il numero del binario è il "puntatore" al treno. Un metodo simile è usato nel linguaggio ASSEMBLY, per accedere a tabelle di dati conservati in memoria: una particolare locazione (il puntatore) contiene l'indirizzo della locazione di memoria ove è contenuto il dato. Questo metodo di indirizzamento è utilissimo in quasi tutti i programmi

indicizzato. Supponiamo di voler scrivere un sottoprogramma in ASSEMBLY, per la ricerca di un codice in una tabella. Ritornando al programma principale, il sottoprogramma deve fornire l'indice dell'elemento trovato. Supponiamo, inoltre, di avere più tabelle in memoria, ma di voler usare un solo sottoprogramma di ricerca. Se questo è realizzato in modo da estrarre l'indirizzo "base" di ciascuna tabella da un puntatore, basterà depositare l'indirizzo "base" della tabella che vogliamo consultare nel puntatore, prima di "chiamare" il sottoprogramma.
In generale, nei programmi vengono adoperati dei metodi misti d'indirizzamento, piuttosto che un singolo metodo. L'istruzione LDA del 6502, per esempio, può essere usata nei seguenti modi:

Op-code	Operando	Modo
LDA	#$34	Immediato
LDA	$A2	Pagina Zero e diretto
LDA	$967F	Assoluto e diretto
LDA	$A2,X	Pagina Zero, indicizzato da X
LDA	$967F,X	Assoluto, indicizzato da X
LDA	$967F,Y	Assoluto, indicizzato da Y
LDA	($A2,X)	Indiretto, pre-indicizzato da X
LDA	($A2),Y	Indiretto, post-indicizzato da Y

Questa istruzione del 6502 si presta molto bene come esempio, poiché riassume un po' tutti i metodi possibili. Una simile tabella può sconcertare e confondere, ma quando avremo fatto pratica con i vari metodi d'indirizzamento, ci apparirà molto più semplice. Infatti, fino ad ora, abbiamo fatto esempi sull'impiego di LDA e ADC usando solo i due metodi più semplici: quello immediato e quello assoluto.
La tabella, inoltre, risponde ad un quesito lasciato in sospeso alla fine dell'ultima lezione: ci eravamo chiesti come poter distinguere il metodo d'indirizzamento se il codice mnemonico di un'istruzione è sempre lo stesso. Possiamo notare che la differenza risiede nel formato dell'operando. L'unico dubbio che può affacciarsi, nella lettura di un listato, è se LDA SYMB1 si riferisca alla pagina zero o sia invece un indirizzo assoluto. Questo dilemma lo risolve, per noi, il programma assemblatore, ma dovendo eseguire la codifica a mano, occorrerà verificare se SYMB1 è stato definito come singolo byte o come un valore a due byte.
In genere, quando si dispone di un programma assemblatore, si possono sorvolare tutti i dettagli che concernono la lunghezza degli op-code, il numero di byte di un'istruzione e via dicendo, concentrandosi invece sulle tecniche di programmazione. È molto importante comprendere la meccanica del codice macchina, ma l'uso di un buon assembler è il miglior modo per imparare a programmare, combinando l'efficienza del codice macchina con molte delle caratteristiche tipiche dei linguaggi ad alto livello.

Risposte all'esercizio di pagina 658

Il programma monitor di pagina 598 è stato scritto modularmente, cosicché l'aggiunta di una nuova opzione al menu è abbastanza facile:

VERSIONE SPECTRUM

1) Aggiustare la parte di preparazione del programma, aggiungendo o modificando le seguenti linee:

```
1050 LET LT=5:DIM C$(LT):DIM 0$(LT,24):DIM X$(16)
1150 LET C$="MVEQB":LET C1=-48:LET
     C2=10-CODE(C$(1))
1280 LET 0$(5)="  B-BINARIO"
```

2) La routine di immissione, alla linea 2000, ha già provveduto a ridurre l'indirizzo iniziale nel formato esadecimale a quattro cifre in A$. L'equivalente decimale è contenuto in DN, cosicché la routine per la visualizzazione in binario è:

```
7000 REM **DISPLAY HEX&BIN**
7020 FOR P=DN TO (DN+15)
7040 LEN NM=P:GOSUB 3100:PRINT H$
7060 LET N=PEEK(P):LET NM=N
7080 GOSUB 3000:PRINT B$:" ";
7100 GOSUB 7300:PRINT B$
7120 IF P=65535 THEN LET P=DN+15
7140 NEXT P
7200 RETURN
7300 REM **BYTE BINARIO**
7310 LET B$ = ""
7320 FOR D=8 TO 1 STEP-1
7330 LET N1=INT(N/2)
7340 LET R=N-2*N1
7350 LET B$=STR(R)+B$
7360 LET N=N1
7370 NEXT D
7380 RETURN
```

VERSIONE BBC/COMMODORE

Trascrivere la versione per lo Spectrum, apportando le seguenti modifiche:

1) Apportare le stesse modifiche alla parte di preparazione, LT e 0$(), come spiegato sopra. Aggiungere C$(5)="B" alla linea 1150
2) La linea 600 deve adesso contenere l'indirizzo della nuova routine, quindi:
   600 ON CM GOSUB 5000,5500,6000,6500,7000
3) Sul solo BBC, cambiare la linea 7060 in:
   7060 N=?(P):NM=N
4) Sul solo Commodore, cambiare la linea 7350 in:
   7350 B$=MID$(STR(R),2)+B$

La Tandy Corporation

Nel corso degli ultimi anni questa ditta americana ha conquistato una larga fetta di mercato

La Tandy Corporation, attraverso le proprie organizzazioni di rivendita (la Tandy e la American Radio Shack) possiede quasi 400 "computer centre" ed oltre 5.500 punti vendita sparsi in 76 nazioni diverse. Possiede inoltre 29 fabbriche che producono le apparecchiature vendute sotto il nome Tandy o Radio Shack.

La Tandy iniziò la propria attività in un campo totalmente diverso da quello dell'elettronica. Infatti, la ditta venne costituita nel 1929 da Norton Hinkley e David Tandy per trattare forniture di cuoio per scarpe ad alcune ditte di Beaumont, nel Texas.

L'espansione nel settore dell'elettronica avvenne nel 1963, quando il figlio di Tandy acquistò una buona parte delle azioni di un'azienda di Boston: la Radio Shack. Questa si occupava del commercio di componenti elettronici e di piccole apparecchiature per radioamatori. Benché commerciasse principalmente per corrispondenza ed avesse ben 9 negozi a Boston e dintorni, gli affari non andavano affatto bene. In soli quattro anni, Charles Tandy riuscì a capovolgere la situazione: da un bilancio iniziale chiuso con una perdita netta di 4 milioni di dollari, ad un utile di oltre 20 milioni di dollari.

Il successivo passo significativo avvenne quando, nel 1970, Tandy rilevò una catena di negozi che rappresentarono lo sbocco verso una vastissima clientela nel settore elettrico ed elettronico. Le cose si sono sviluppate a tal punto che il catalogo della Tandy per il 1984 contiene, oltre ai quasi 400 prodotti per computer, più di 2.600 articoli (dalle resistenze agli hi-fi).

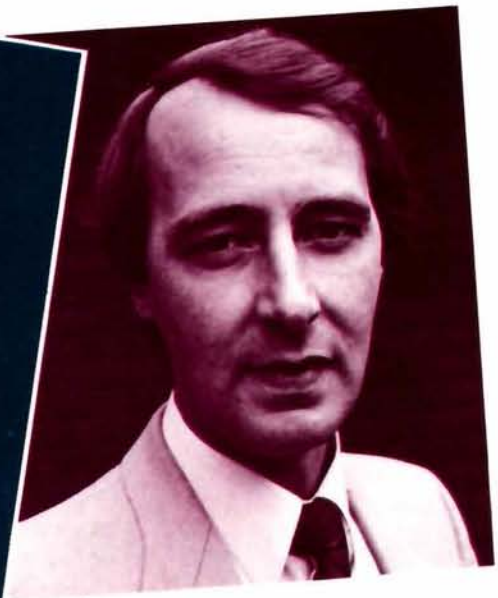
La Tandy è arrivata in Europa (e più precisamente in Gran Bretagna) nel 1973, assicurandosi in breve tempo una posizione di rilievo nel commercio di apparecchiature elettriche. Nel 1978, quando venne lanciato il microcomputer TRS-80 Modello I, la Tandy aveva ben 120 punti vendita nel solo Regno Unito, diventati poi 227 nel 1983. Ventisei di questi negozi sono specializzati in computer, software e periferiche.

Il glorioso Modello I ha fatto della Tandy una delle migliori produttrici di computer. Dotato di un microprocessore Z80, di almeno 4 Kbyte di memoria RAM e di una grafica a bassa risoluzione (il tutto su scheda singola), questo computer è stato un vero successo. Oggi sono disponibili anche unità a dischetti ed il sistema operativo CP/M.

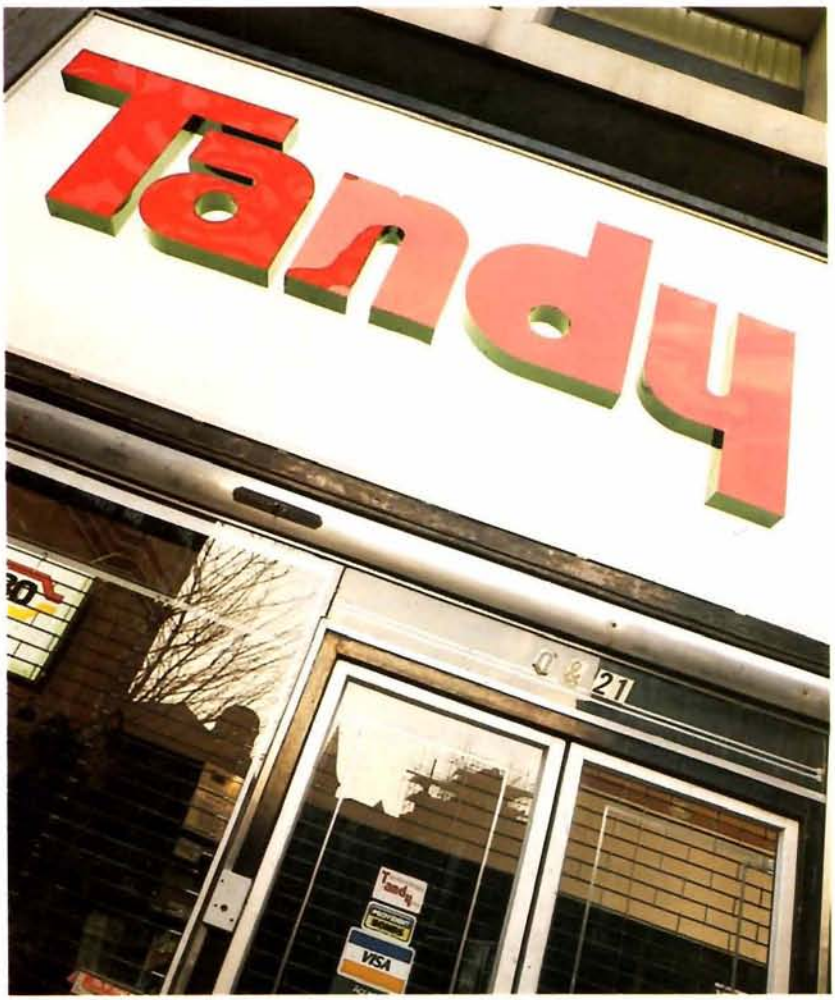
La Tandy si è costantemente tenuta al passo coi tempi, ma non è riuscita a ripetere il grande successo ottenuto con il Modello I. È stato prodotto un Modello II e, ultimamente, i Modelli 12 e 16 (entrambi a 16 bit) ed il modello 2000, compatibile con il Personal IBM. Inoltre, è stato realizzato un Modello III, a cavallo tra la fascia degli home computer e quelli per applicazioni commerciali, ben



John Sayers



Vince Moore



IAN McKINNELL

Una fitta rete

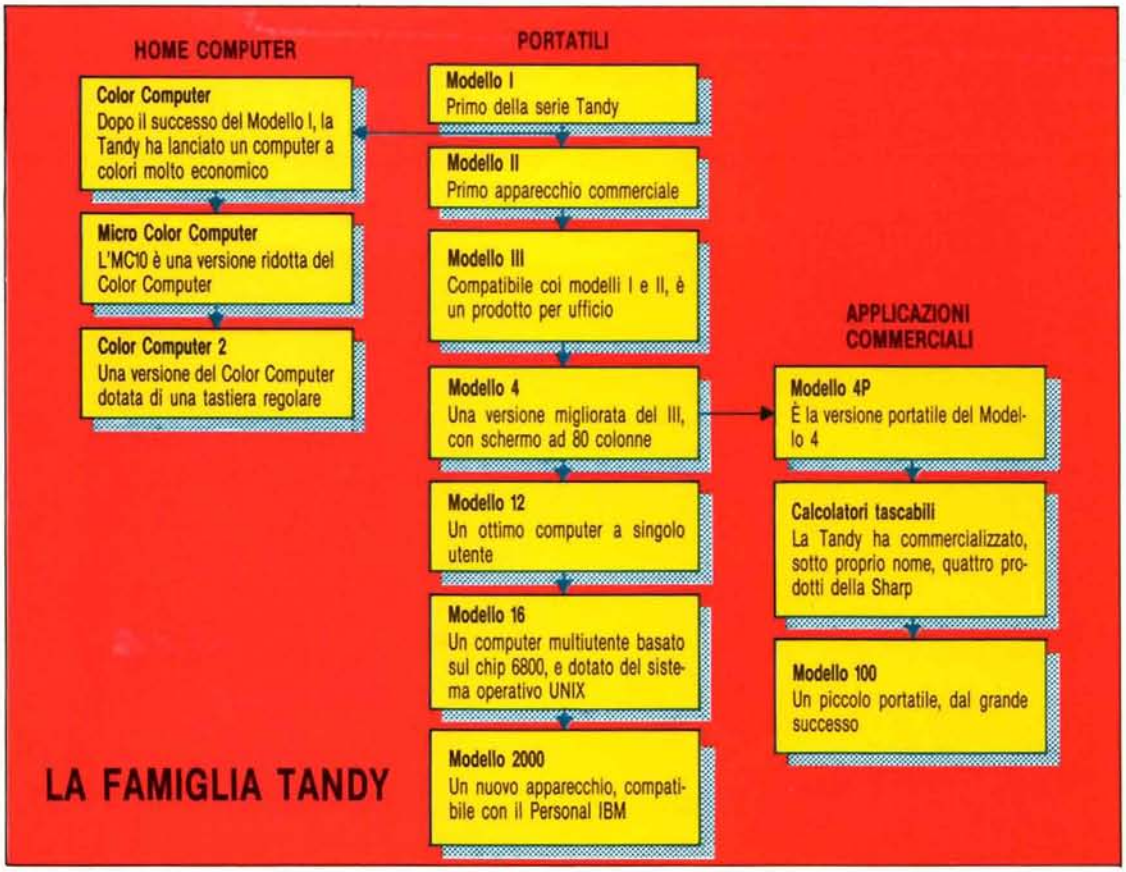
Parte della forza della Tandy proviene da una vastissima rete di distribuzione, sparsa in 76 paesi del mondo. Molti dei negozi sono veri e propri "computer centre", specializzati in prodotti per l'informatica

presto rimpiazzato dal Modello 4 (e dal gemello portatile 4P). Questi ultimi apparecchi sono compatibili con i precedenti Modelli I e III, un fatto raro nel mondo dei microcomputer. Di conseguenza, la quantità di software disponibile per tutta la serie di apparecchi è davvero impressionante.

La Tandy ha anche prodotto un home computer basato su di un altro microprocessore, il 6809 (lo stesso impiegato per il Dragon), chiamato Tandy Color Computer. Le vendite sono state abbastanza soddisfacenti, specialmente negli USA dove l'apparecchio è stato ribattezzato "CoCo". Ne esiste anche una versione ridotta, il Micro Color Computer, a dimostrare che la Tandy non intende abbandonare il settore.

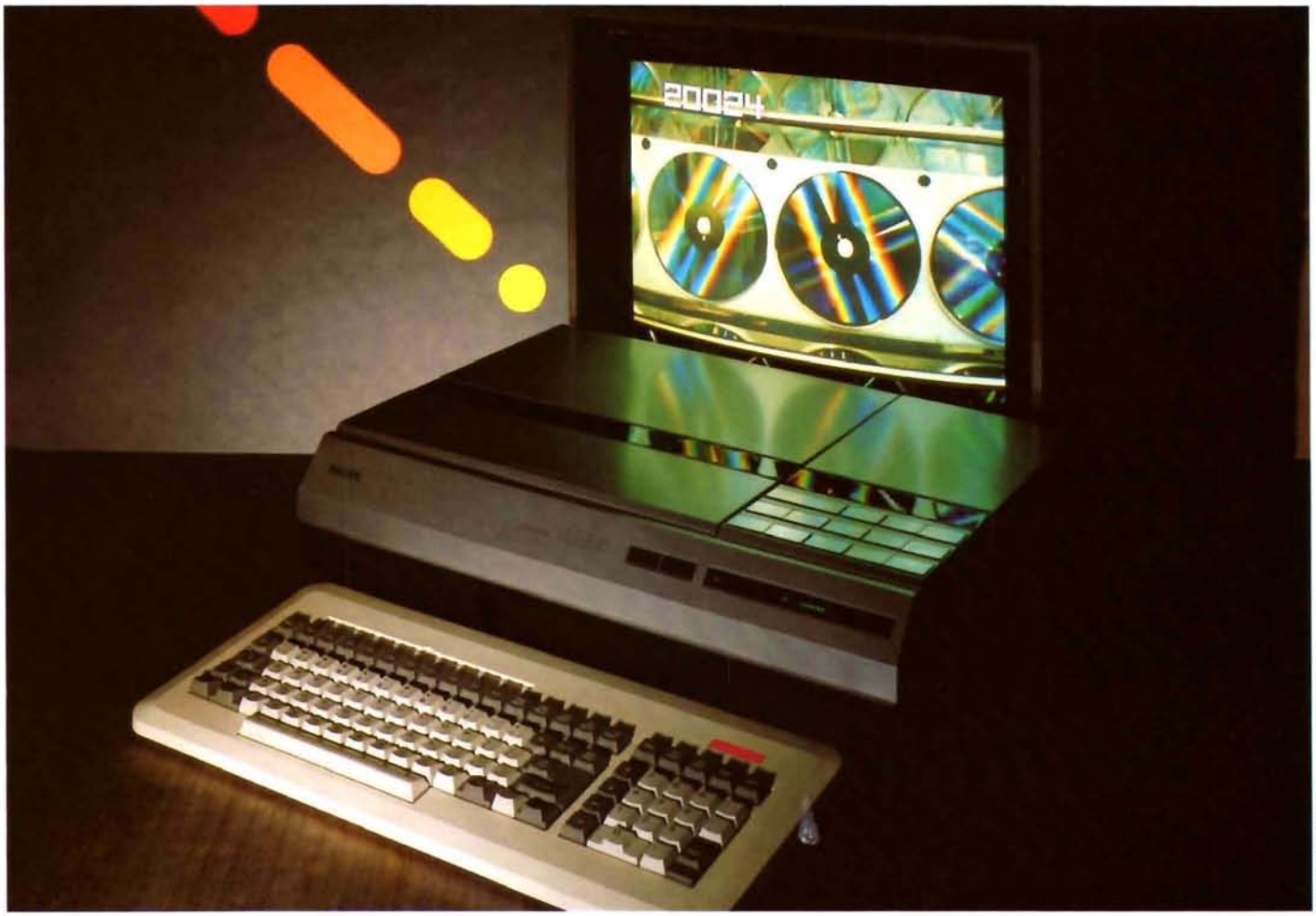
Forse, i prodotti attualmente più interessanti sono i piccoli portatili. Congiuntamente alla giapponese Sharp, la Tandy ha prodotto tutta una serie di piccoli computer tascabili che lavorano in BASIC. Ultimamente, rivolgendosi ad un'altra ditta nipponica, la Kyocera, ed alla ben nota Microsoft, la Tandy ha prodotto il Modello 100, un computer portatile, alimentato a pile, completo di BASIC, di un word processor e di altri utili programmi.

Agli inizi del 1984, molte grosse ditte del settore hanno subito una preoccupante crisi, ma la Tandy sembra resistere solidamente, continuando a produrre sempre nuovi apparecchi sia negli USA, a Fort West, che in Giappone, dove opera la TC Electronics Corporation, una filiale della Tandy.



Solchi di luce

Presto avremo in casa i videoriproduttori a dischi laser, affiancati dal personal



IAN McKINNELL

I giradischi a laser, considerati fino a pochi anni orsono come apparecchi di lusso, costano oggi più o meno quanto un videoregistratore, offrendo, tuttavia, prestazioni di gran lunga superiori. Nei videoregistratori, le immagini televisive sono composte da un'ininterrotta serie di "quadri" dinamici e l'unico modo per ritrovarne uno in particolare è quello di far scorrere il nastro in "avanti veloce", basandosi sul contagiri dell'apparecchio. Sui videodischi laser, al contrario, le immagini vengono immagazzinate sotto forma di configurazioni individuali, consentendo il pressoché immediato ritrovamento di ciascuna di esse. La locazione di ogni configurazione sul disco può essere descritta in termini di tracce e di settori, pertanto un microprocessore si rivela utilissimo per accedere in modo rapido e ordinato all'archivio delle immagini. Inoltre, mediante il microprocessore, si possono esaminare sia singole immagini "fisse", sia sequenze al rallentatore.

La conquista tecnologica più importante, nei dischi laser, non è stata tanto quella di ottenere dal microprocessore un totale controllo sia sulla rotazione costante ad alta velocità della piastra, sia sul preciso posizionamento della testina di riproduzione sulla superficie del disco, ma quella di riuscire ad avere un'elevata densità nell'immagazzinamento dei dati. Infatti, incidere migliaia di immagini su un disco da 12 pollici non è affatto facile.

Parlando della grafica ad alta risoluzione, adottata sui microcomputer, abbiamo visto che un'immagine televisiva è costituita da singoli punti luminosi (pixel): maggiore è il numero di pixel sullo schermo, migliore è la qualità dell'immagine. Di conseguenza, la conservazione di immagini ad alta risoluzione occupa una notevole quantità di memoria.

Laser da...salotto

Il prezzo dei riproduttori a dischi laser dovrebbe presto avvicinarsi a quello degli home computer e sarà abbastanza comune trovare, nelle abitazioni, entrambi questi apparecchi, collegati tra loro. Con dischi e software appropriati, sarà possibile accedere a grandi database per immagini e a sofisticati programmi di aggiornamento, come pure a divertenti giochi d'avventura

Una nuova memoria

Dischi laser a 11 pollici e dischi compatti di minori dimensioni possono essere usati per immagazzinare sia informazioni video che audio, come oggi avviene per i dati digitali archiviati su cassette o dischetti. L'unico inconveniente è che i dischi non possono essere riutilizzati in scrittura (tecnicamente si comportano come memorie ROM). Ciò significa che attualmente non è possibile memorizzare programmi o dati su un disco laser, come sui dischetti magnetici: il software deve essere acquistato in forma "preconfezionata"



IAN McKINNELL

Il Micro BBC, ad esempio, ha una risoluzione massima di 640 × 256 pixel per immagine ed uno schermo intero occupa esattamente 20 Kbyte di memoria. Se un'immagine televisiva avesse pari risoluzione, allora immagazzinare una sequenza di un solo secondo (a 25 immagini per secondo) richiederebbe 25 × 20 Kbyte di memoria! Un minuto di un programma televisivo occuperebbe 30 Megabyte (30 milioni di byte) ed un telefilm di media lunghezza avrebbe bisogno di più di un Gigabyte di memoria (1000 Megabyte). In una simile ipotesi, occorrerebbero circa 6.000 dischetti a singola faccia e singola densità (il tipo usato sulla maggior parte degli home computer).

In base a queste cifre, registrare digitalmente un programma televisivo di mezz'ora su dischetti magnetici sarebbe un'idea da scartare, a tutto vantaggio dei tradizionali videoregistratori a nastro.

Il problema è stato risolto diminuendo drasticamente la superficie occupata da ciascun dato. Infatti, nei registratori a laser, i dati vengono impressi mediante un fascio laser sottile quanto un capello su un disco metallico rivestito da un sottile strato translucido. La lettura dei dati dal disco avviene impiegando un raggio laser a bassa potenza. Dunque, i raggi laser vengono usati come vere e proprie "puntine" di lettura/scrittura di massima precisione. Nessun'altra tecnica permette l'immagazzinamento di un così alto numero di informazioni in uno spazio così ristretto.

Il formattamento dei dischi laser è una combinazione tra la tecnica adottata sugli apparecchi ad alta fedeltà e quella usata per le unità a dischetti dei computer. I solchi di un comune disco musicale formano una spirale continua e la conformazione delle pareti dei solchi riproduce le forme d'onda del suono registrato. Nei dischetti dei computer non esistono solchi: le informazioni (sequenze di livelli 0 ed 1) vengono scritte, mediante induzione magnetica, in tracce concentriche. Nei dischi laser la registrazione viene effettuata con un processo ottico suddividendo il disco in tracce, ma con andamento a spirale. I dischi laser non sono riutilizzabili in scrittura, poiché le configurazioni di 0 e di 1 (che costituiscono la rappresentazione digitale delle immagini) vengono incise sulla superficie del disco mediante microscopiche bruciature puntiformi (pit), ottenute col laser. Ciascun pit è largo mezzo micron (0,0005 mm.) e profondo un decimo di micron (0,0001 mm.), pertanto un centimetro quadrato di disco può contenere ben 400 milioni di pit.

Pur essendo stupefacente, un tale livello di miniaturizzazione è appena sufficiente per soddisfare le necessità di immagazzinamento di sequenze video. Un disco di 35 cm. di diametro può contenere fino a 54.000 immagini video per faccia, cioè circa 36 minuti di registrazione. I calcoli approssimativi sull'occupazione delle immagini, fatti in precedenza, si basavano sulla risoluzione grafica dei computer in bianco e nero, ma un disco laser deve immagazzinare anche le informazioni sul colore di ciascun punto dell'immagine oltre a possedere una traccia audio. Un'immagine a colori con la corrispondente traccia audio può quindi occupare 100 Kbyte di memoria: per 54.000 immagini occorrono dunque 5.400.000 Kbyte, cioè 5,4 Gigabyte.

Avendo rimosso gli ostacoli dovuti alle limitazioni di memoria, i dischi a laser offrono immense possibilità applicative. Uno dei principali vantaggi è l'eliminazione del maggior inconveniente nell'elaborazione dei dati: la raccolta e l'immissione dei dati. Infatti, le informazioni sono in genere facilmente ottenibili, ma è pur sempre necessario che un operatore si sieda al terminale ed inserisca fisicamente nel sistema le rappresentazioni in codice dei dati. È questa una procedura lunga, noiosa ed anche costosa. Se, invece, basta puntare una telecamera sui dati e far sì che essa registri le informazioni su disco, occorrerà soltanto che l'operatore si dedichi a numerare le immagini, con evidente risparmio di tempo e fatica.

Esistono registratori a disco laser di varia qualità: sono già in vendita piccoli apparecchi domestici utilizzabili per proiettare film, più o meno simili ai normali video registratori, ma con il vantaggio di una qualità dell'immagine di gran lunga superiore sia nelle sequenze a velocità normale o al rallentatore che nelle riproduzioni statiche. Alcuni apparecchi permettono anche di selezionare singole immagini battendo il loro numero di codice su una tastierina manuale e utilizzando una procedura di ricerca. Il registratore oscurerà lo schermo per un momento e poi visualizzerà l'immagine richiesta.

È possibile ottenere questo risultato solo quando le singole immagini possono venir selezionate da un apposito programma per computer. Registratori in grado di compiere questa operazione vengono già prodotti dalla Pioneer e dalla Philips, ma sono ancora piuttosto costosi e destinati esclusivamente ad usi professionali. Il sistema più semplice adotta interfacce IEEE o RS232 e permette al computer di selezionare una particolare immagine in base al suo numero. Il programma

del computer può anche creare un proprio archivio di indici per selezionare le immagini secondo speciali criteri.
La Philips ha addirittura incorporato, nei modelli più recenti, un microcomputer in grado di caricare un programma adatto ad un particolare disco sia da una cartuccia EPROM innestata nell'apparecchio, sia dallo stesso disco a laser. Ogni disco contiene due tracce audio ed una traccia video; ciò permette di avere, ad esempio, una colonna sonora in due lingue diverse. Tuttavia, se la seconda traccia audio non è necessaria, può essere utilizzata per immagazzinare un programma per il computer.
Abbiamo quindi una banca di 54.000 immagini di buona qualità controllate dal computer. La fase finale consiste nella fusione delle immagini registrate sul disco laser con il testo proveniente o controllato dal computer. Il risultato finale sarà uno strumento completamente nuovo: il video interattivo. L'operatore ed il software possono gestire l'immagine televisiva sia sequenze che immagini statiche richiamando le immagini sul disco in un qualunque ordine.
L'applicazione più ovvia è un database iconografico, nel quale l'utente rivolge domande al computer, ottenendo specifiche informazioni prelevate dal database, e guida il riproduttore alla creazione di una particolare sequenza video. Questo genere di apparecchio potrà essere sfruttato con prestazioni nettamente superiori ai sistemi basati su microfilm.
Il video interattivo può costituire uno strumento impareggiabile nel campo della didattica. Con opportuni programmi di controllo, le immagini possono susseguirsi ad un ritmo e ad un livello di complessità accuratamente commisurato allo spettatore, al quale vengono via via rivolte domande atte a determinare il grado di comprensione. È inoltre possibile produrre filmati che lo stesso spettatore dirige e modifica: l'intera trama ed il finale di un film potrebbero svilupparsi in modo diverso a seconda dei desideri del "regista". Gli appassionati di giochi d'avventure per micro potrebbero trovare queste nuove possibilità molto stimolanti.
Lo sviluppo di questo mercato presenta naturalmente molti problemi. A parte il costo dei componenti e della fabbricazione vera e propria dei dischi laser, nuove capacità dovranno essere sviluppate nella progettazione e nella produzione dei dischi interattivi sia per quanto riguarda il software per computer che per l'elaborazione e la realizzazione delle immagini. Molte piccole compagnie stanno cominciando a porsi questi obiettivi ed una multinazionale, la Computer Assisted Televideo (CAT), si è già affermata come compagnia leader in questo mercato. Essa fornisce ai clienti un servizio completo: scelta ed installazione dei vari componenti del sistema, progettazione e produzione di dischi, sviluppo del relativo software. La CAT ha in progetto di seguire quasi ogni applicazione del video interattivo, ma gli attuali alti costi limitano la maggior parte del suo lavoro alla produzione di programmi di addestramento per attività commerciali di grandi dimensioni.
Quando, tra non molto, saranno disponibili sistemi completi ad un costo accessibile anche per uso domestico, chi andrà più al cinema?

Video interattivi

Molti registratori a dischi laser, dotati di interfaccie IEEE e RS232, vengono posti sotto il controllo di microcomputer. Un tipico sistema potrebbe essere formato da un computer che gestisca l'archivio dei riferimenti alle immagini immagazzinate su un disco laser. L'utente seleziona le informazioni che lo interessano mediante il computer, che, a sua volta, guida il riproduttore a dischi laser nella ricerca e nella visualizzazione di una particolare immagine in base ad un numero d'identificazione. Il sistema mostrato in figura adotta un comune metodo di combinazione su schermo dell'output del computer con le immagini di un disco laser. Un sistema più semplice potrebbe usare due schermi separati



STEVE CROSS

I file sequenziali

Come creare ed usare i file sequenziali sul nostro microcomputer

Un file binario (o programma) è un caso semplificato di un file sequenziale. Quando impartiamo il comando SAVE "NOMEPROG", il sistema operativo esegue alcune operazioni necessarie alla creazione di un file: innanzi tutto apre una linea di comunicazione con il registratore o l'unità a dischetti scrivendovi un'"etichetta" col nome del file ed alcune informazioni sul suo contenuto. Poi, il sistema operativo ricopia, all'interno del file, il blocco di memoria che contiene il programma da memorizzare. Infine, viene accodato un marcatore di "fine file" e viene chiuso il collegamento tra il computer ed il registratore o l'unità a dischetti.
I comandi BASIC usati nella creazione di un file sequenziale, variano da una versione all'altra, ma devono tutti effettuare lo stesso tipo di operazioni. Prendiamo, ad esempio, il Commodore 64:

```
100 R$ = "Questo è un record del file"
150 OPEN 5,1,2, "DATAFILE,SEQ,WRITE"
200 PRINT #5,R$
250 CLOSE 5
```

Il programma inizia con il comando OPEN 5,1,2, che istruisce il sistema operativo ad aprire un canale di comunicazione, chiamato canale 5, con la periferica numero 1 (cioè il registratore). Il sistema operativo può creare file sequenziali su un certo numero di diverse periferiche e può accedere contemporaneamente a più di un file: per questa ragione il canale che collega il computer alla periferica (e quindi ad un particolare file) deve possedere un'etichetta d'identificazione. Il numero 2, nell'esempio, è specifico per il Commodore.
Il comando OPEN è seguito dal nome del file. Sul Commodore il nome del file viene indicato dalla parola "DATAFILE", seguita dal tipo o estensione del file (SEQ, che significa file sequenziale) e dal metodo di accesso (WRITE indica che il file viene aperto per la scrittura).
I file sequenziali possono essere aperti (OPEN) tanto per la lettura che per la scrittura, ma non per entrambe le procedure contemporaneamente. Questo comando fa sì che la testina di lettura/scrittura provveda a contrassegnare l'inizio del file con il nome del file ed alcune informazioni di sistema.
Il comando PRINT #5, alla linea 200, invia dati al file. PRINT mantiene il suo usuale significato, ma il segno "#" indica che l'informazione non deve essere inviata al video, ma ad uno specifico canale di uscita. Il contenuto di R$ viene quindi inviato sul canale 5 al file sequenziale chiamato "DATAFILE". Il file contiene adesso una registrazione: la stringa "Questo è un record del file". Infine, CLOSE 5 chiude il canale 5 e accoda un marcatore di fine file al contenuto dello stesso file.
Le informazioni, all'interno di un file, sono organizzate in modo tale da poter essere lette in una fase successiva, per esempio:

```
500 OPEN 3,1,2, "DATAFILE,SEQ,READ"
550 INPUT #3,A$
600 CLOSE 3
650 PRINT A$
```

Sia in questo frammento di programma, sia in quello precedente, viene impiegato il comando OPEN, che qui significa però: "trova l'inizio del file chiamato DATAFILE e preparati alla sua lettura". Il numero di canale assegnato al file è diverso dal precedente (ma avremmo anche potuto usare lo stesso valore o qualsiasi altro, purché lo si ripeta in tutte le istruzioni che concernono le operazioni sul file). L'indirizzo della periferica è lo stesso in entrambi i casi perché il file è stato immagazzinato sulla periferica 1 cioè il registratore. Il nome del file e la sua estensione sono gli stessi perché identificano il file a cui si vuol accedere, ma è diverso il metodo di accesso: READ al posto di WRITE.
Questo programma usa il comando INPUT #3,A$, come già spiegato per PRINT, il simbolo # fa sì che l'immissione dei dati, anziché dalla tastiera, avvenga leggendo il file. Il primo record del file viene letto ed inviato sul canale 3 all'interno della variabile A$, proprio come il comando PRINT #5,R$ aveva inviato il primo record sul canale 5 alla variabile R$.
I file sequenziali possono dunque immagazzinare dati, sotto forma di record, senza particolari limitazioni alle dimensioni del file. D'altra parte, però, il contenuto di un file sequenziale può essere letto solo a partire dall'inizio, record dopo record: non è possibile aprire il file posizionandosi su un particolare record né è consentito saltare un record, rileggerlo, cancellarlo, inserirlo o accodarlo.



BOB HALL

Ordine di registrazione

I file sequenziali sono stati sviluppati quando non esistevano che i nastri magnetici: infatti l'ordine con cui i record vengono inviati al file è l'ordine che essi conserveranno per sempre. Il solo modo per riordinare o manipolare un file sequenziale è di crearne una nuova versione, proprio come l'ordine delle canzoni di una musicassetta può essere cambiato solo trasferendone il contenuto su di un'altra cassetta

IMMAGAZZINAMENTO E RICERCA

```
199 REM +++++++++ Versione C64 ++++++++++
200 REM +       SCRITTURA DEI FILE        +
201 REM +++++++++++++++++++++++++++++++++
220 GOSUB 1500
240 FOR K=65 TO 90
260 Z$=CHR$(K)+X$:C$=D$+"SCRITTURA "
    +CHR$(K)
280 GOSUB 2000
300 NEXT K
399 REM +++++++++++++++++++++++++++++++++
400 REM +        LETTURA DEI FILE         +
401 REM +++++++++++++++++++++++++++++++++
420 FOR L=0 TO 1 STEP 0:FOR M=1 TO 1
440 PRINT D$;"ACCESSO AI RECORD"
460 INPUT "STRINGA DI RICERCA (*=FINE)";N$
480 L$=LEFT$(N$,1):IF L$="*" THEN GOTO 5000
500 IF L$ <"A" OR L$> "Z" THEN M=0
520 NEXT M
540 Z$=L$+Y$
560 GOSUB 3000
580 PRINT TAB(5) "RECORD ";N;"(PREMERE UN
    TASTO)"
600 GET GT$:IF GT$="" THEN 600
620 NEXT L
999 END
1499 REM ================================
1500 REM =  SUBROUTINE INIZIALIZZAZIONE  =
1501 REM ================================
1520 D$=CHR$(147):PRINT D$,CHR$(8);CHR$(142)
1540 X$=",S,W":Y$=",S,R"
1600 RETURN
1999 REM ================================
2000 REM =    SUBROUTINE DI SCRITTURA    =
2001 REM ================================
2020 PRINT C$:INPUT "QUANTI RECORD";R
2040 OPEN 8,8,2,Z$
2060 IF R=0 THEN PRINT#8,"*":CLOSE8:RETURN
2080 FOR I=1 TO R
2100 PRINT C$:PRINT "RECORD N.";I
2120 INPUT "TESTO....";R$
2140 PRINT#8,R$
2160 NEXT I
2180 PRINT#8,"*":CLOSE8
2499 RETURN
2999 REM ================================
3000 REM =    SUBROUTINE DI LETTURA      =
3001 REM ================================
3020 PRINT D$;"RICERCA DI ";L$;" PER ";N$
3040 OPEN 8,8,2,Z$
3060 FOR I=1 TO 100000
3080 INPUT#8,R$
3100 PRINT R$
3120 IF R$="*" THEN N=0:I=100000
3140 IF R$=N$ THEN N=I:I=100000
3160 NEXT I:CLOSE8:N$=""
3180 RETURN
5000 REM ======= FINE PROGRAMMA ==========
5020 PRINT CHR$(9);"FINE DEL PROGRAMMA":STOP
```

220: Inizializzazione

260-280: Creazione dei file da "A" a "Z"

420-450: Immissione del record da cercare, individuazione della lettera iniziale, identificazione del file che la contiene

560: Ricerca in tale file

580-600: Visualizza il risultato della ricerca

1520: Cancellazione del video, selezione del modo "maiuscole"

2040; Apertura (OPEN) del file per la scrittura (WRITE)

2060: Controllo per file pieno, scrittura del carattere "*", chiusura (CLOSE) del file

2120-2140: Scrittura del record e suo inserimento nel file

2180: Scrittura del carattere "*" come ultimo record, chiusura del file

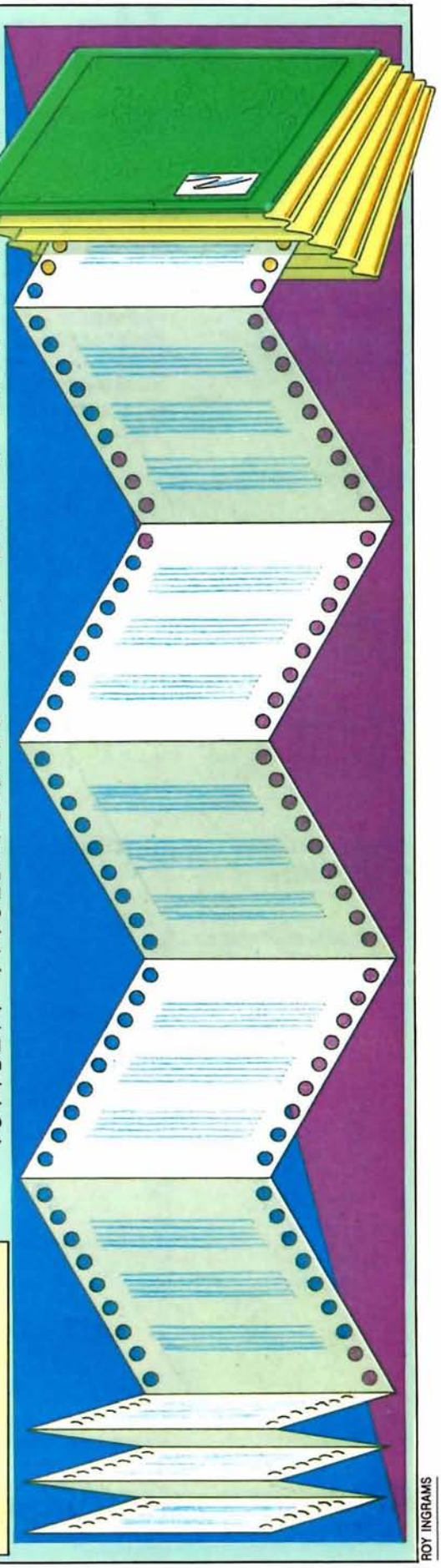
3040: Apertura del file per la lettura (READ)

3120: Test per l'ultimo record, abbandono della ricerca

3140: Verifica se il record è stato trovato o abbandono della ricerca

3160: Chiusura (CLOSE) del file

Questo programma chiarisce l'uso dei file sequenziali su disco e crea un semplice indice alfabetico comprendente 26 file, uno per ciascuna lettera dell'alfabeto. All'interno di ogni file è possibile scrivere record che iniziano con la lettera identificativa del file, oppure nessun record. Si può quindi eseguire la ricerca di un qualsiasi record. Viene individuato il file appropriato ed il suo contenuto verrà visualizzato fino all'individuazione del record cercato. Se la ricerca fallisce, sul video compare il messaggio "Record 0".
Il programma utilizza il sistema operativo a disco per accedere al file appropriato ed eseguire la ricerca del record: il file viene quindi letto in modo sequenziale. Questa procedura permette di ridurre il tempo di ricerca; se si trattasse di un unico grande file, la ricerca sarebbe notevolmente lenta



ROY INGRAMS

Basic & Basic

BBC Micro
Eseguire le stesse variazioni dello Spectrum per le linee 260 e 540. Sostituire PRINT#8, INPUT#8 e CLOSE8 con PRINT#C8, INPUT#C8 e CLOSE#C8.

```
 600 GT$ = GET$
1520 *DISK
1530 MODE 7
1540 D$ = CHR$(12):PRINT D$ "USARE MAIUSCOLE"
1550 PRINT "-PREMERE QUALSIASI TASTO":GT$ = GET$
2040 C8 = OPENOUT(Z$)
3040 C8 = OPENIN(Z$)
5020 PRINT "FINE DEL PROGRAMMA"
```

Spectrum con Microdrive
Inserire le LET nelle assegnazioni
Sostituire PRINT D$;... con CLS:PRINT...
Sostituire PRINT C$ con CLS:PRINT C$
Sostituire OPEN 8,8,2,Z$ con OPEN #8;"m";1;Z$
Sostituire CLOSE8 con CLOSE #8
Cancellare la linea 1540

```
 260 Z$ = CHR$(K):LET C$ = "SCRITTURA" + Z$
 480 LET L$ = N$(1):IF L$ = "*" THEN GOTO 5000
 540 LET Z$ = L$
 600 PAUSE 0
1520 CLS:LET F2 = PEEK 23658:POKE 23658,8
3080 INPUT #8;R$
5020 POKE 23658,F2:PRINT "FINE DEL PROGRAM-
     MA":STOP
```

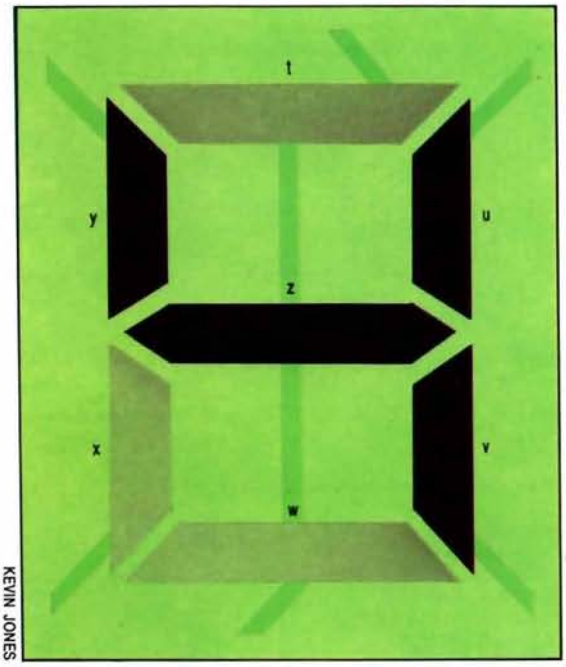
Un convertitore per display

Come pilotare i display a sette segmenti, usati negli orologi e nelle calcolatrici

Per il progetto del nostro circuito partiamo dal presupposto che la rappresentazione binaria dei numeri decimali adotti il codice conosciuto come Binary Decimal Coded (decimale codificato in binario), più brevemente BCD. Ciò significa che ciascun numero decimale viene immagazzinato come un gruppo di 4 cifre binarie, come mostrato nella seguente tabella.

Decimale Codificato in Binario	Decimale
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
da 1010 a 1111	(non usati)

I display a sette segmenti
Con soli sette segmenti si possono rappresentare i dieci numeri decimali.
Il visualizzatore a LCD è composto da 7 barrette commutabili individualmente, etichettate con lettere comprese tra la "t" e la "z"



KEVIN JONES

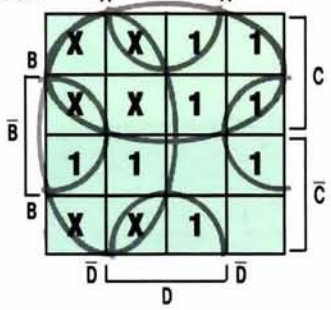
Un display a sette segmenti visualizza i numeri decimali, accendendo opportunamente alcuni diodi fotoemittenti (nel caso dei LED) oppure modificando la polarità dei segmenti interessati nei display a cristalli liquidi (LCD). L'illustrazione mostra la lettera alfabetica assegnata a ciascun segmento (la barretta in alto è indicata dalla lettera "t", quella più in basso dalla "w" e così via). Utilizzeremo quindi le varie combinazioni di segmenti per formare i numeri da 0 a 9.
Queste sono tutte le informazioni che ci occorrono per costruire la tavola della verità del nostro convertitore. Grazie alle illustrazioni presentate a fianco possiamo comprendere quali segmenti verranno accesi, in base ai valori in ingresso al circuito. Ad esempio, se l'input binario è 0100 (corrispondente al decimale 4), il circuito accenderà le barrette u, v, y, z. Similmente, l'input 1000 (8 in decimale) provoca l'accensione di tutte le barrette. Quindi, la tavola della verità per il nostro circuito è:

Decimali	Ingressi				Uscite						
	A	B	C	D	t	u	v	w	x	y	z
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
	1	0	1	0	X	X	X	X	X	X	X
	1	0	1	1	X	X	X	X	X	X	X
	1	1	0	0	X	X	X	X	X	X	X
	1	1	0	1	X	X	X	X	X	X	X
	1	1	1	0	X	X	X	X	X	X	X
	1	1	1	1	X	X	X	X	X	X	X

Sfortunatamente non esiste un procedimento semplice per analizzare questa tavola della verità e di conseguenza ciascuna uscita (t, u, v, w, x, y e z) deve essere semplificata separatamente. Possiamo costruire una mappa di Karnaugh per ogni uscita, inserendo su ciascuna mappa le condizioni di "non validità" (il simbolo X). Queste, in alcuni casi, possono essere utili alla semplificazione delle espressioni. Queste 7 mappe k, una per ogni segmento, sono mostrate nella pagina seguente. Dopo aver racchiuso in un cerchio i vari gruppi, possiamo ottenere altrettante espressioni semplificate. In alcuni casi è possibile raggiungere un'ulteriore semplificazione mediante la fattorizzazione.
La prima mappa k, ad esempio, si riferisce agli ingressi che accendono il segmento "t" (usato in tutti i numeri eccetto 1 e 4). Dopo aver semplificato le espressioni, potremo tracciare il circuito definitivo.

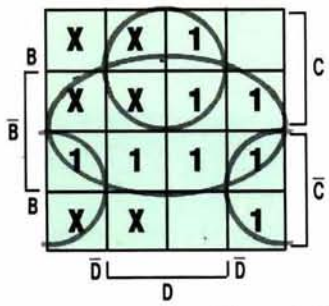


Per t



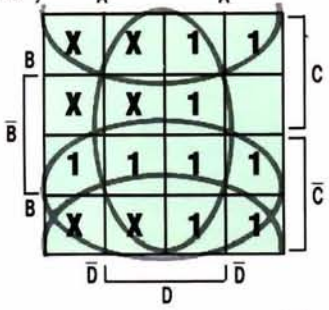
$t = C + A + \bar{B}.\bar{D} + B.D$

Per u



$u = \bar{B} + C.D + \bar{C}.\bar{D}$

Per y

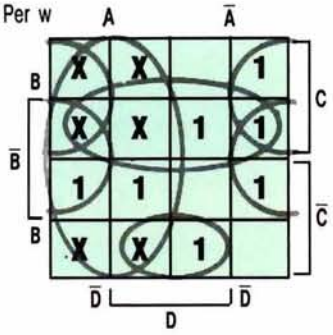


$v = D + \bar{C} + B$

Codice a sette segmenti

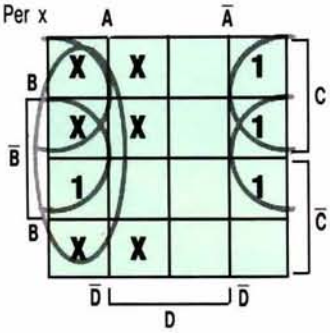
In base a questi diagrammi possiamo determinare quali segmenti concorrono alla formazione di ciascun numero. Il numero 1, ad esempio, impiega soltanto i segmenti "u" e "v". La tavola della verità illustrata nella pagina precedente mostra le uscite necessarie per formare tutti i numeri dallo 0 al 9

Per w



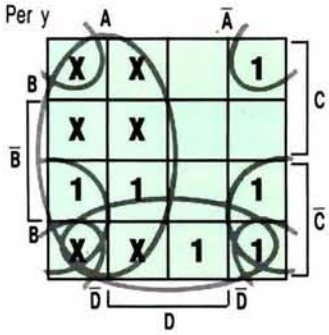
$w = A + C.\bar{D} + \bar{B}.C + \bar{B}.\bar{D} + B.\bar{C}.D.$

Per x



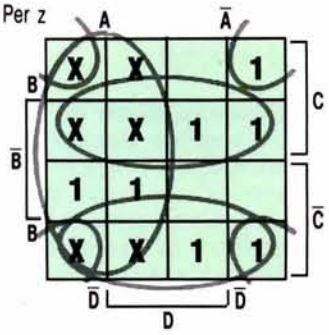
$x = C.\bar{D} + A.\bar{D} + \bar{B}.\bar{D}$
$= (A + \bar{B} + C).\bar{D}$

Per y

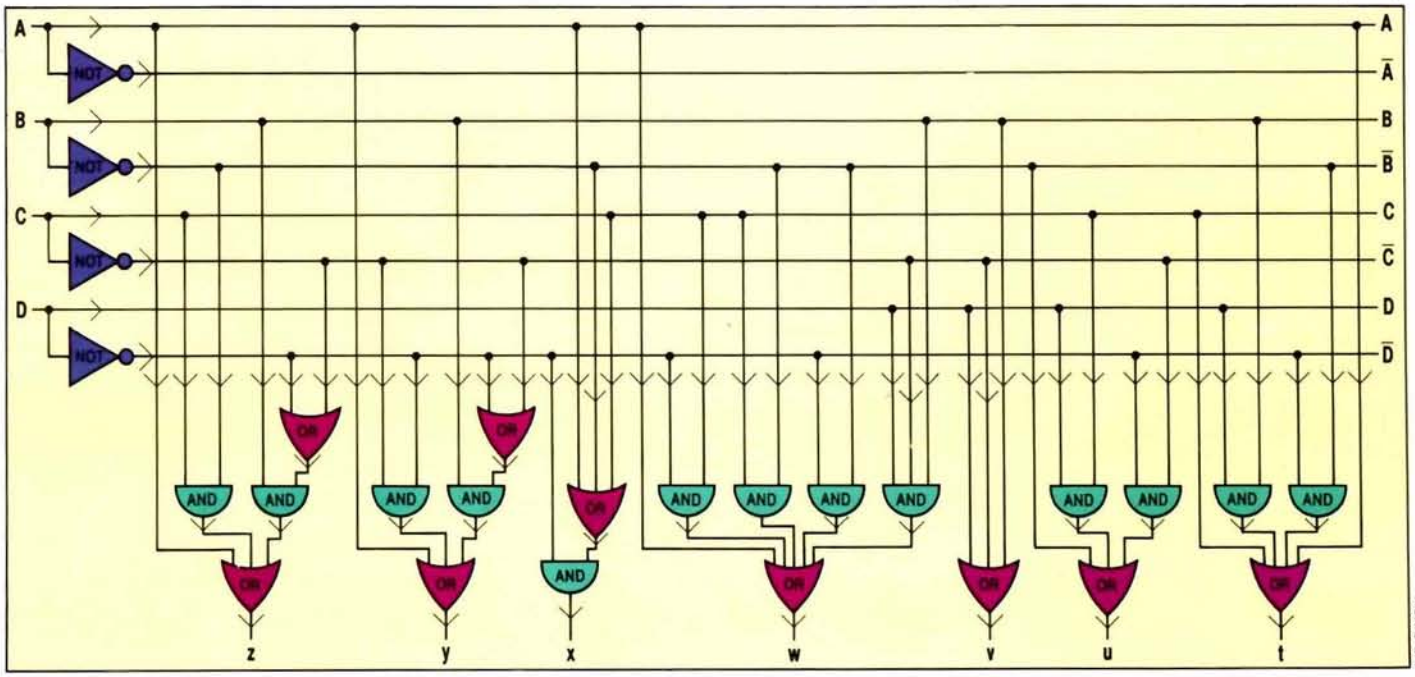


$y = A + B.\bar{C} + \bar{B}.\bar{D} + \bar{C}.\bar{D}$
$= A + B.(\bar{C} + \bar{D}) + \bar{C}.\bar{D}$

Per z



$z = A + \bar{B}.C + B.\bar{C} + B.\bar{D}$
$= A + \bar{B}.C + B.(\bar{C} + \bar{D})$

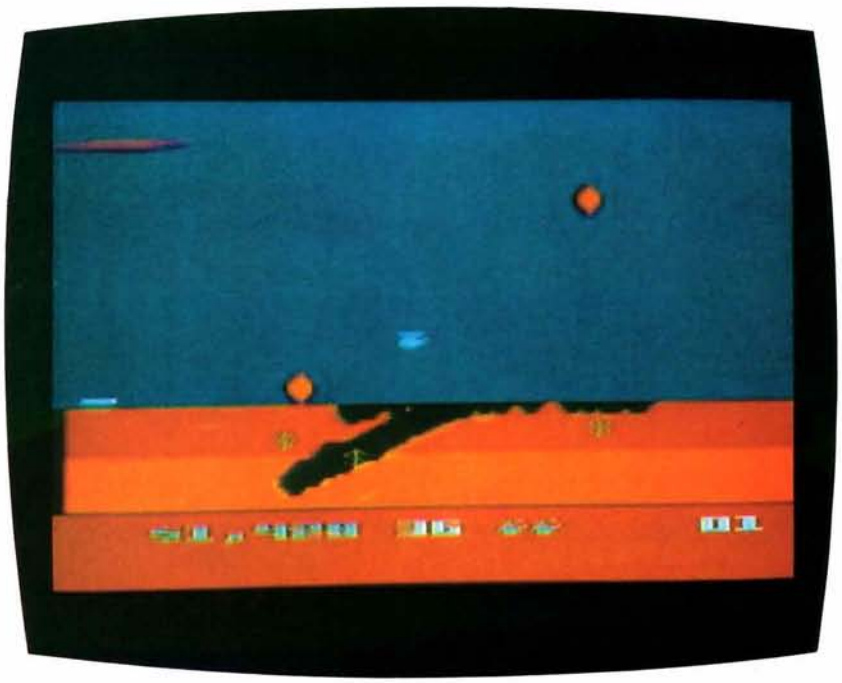


Il circuito che abbiamo progettato controlla un display con una sola cifra, ma, poiché la maggior parte dei circuiti di questo tipo prevedono 8 o 10 cifre, può sembrare necessario costruire un circuito per ogni cifra. In realtà, si può adoperare un solo circuito di decodifica, sfruttando un sistema chiamato "multiplex". Questo particolare metodo consente di accendere o spegnere ciascuna cifra dell'intero display in una sequenza talmente rapida da far sembrare attive tutte le cifre contemporaneamente.

Software in vetrina

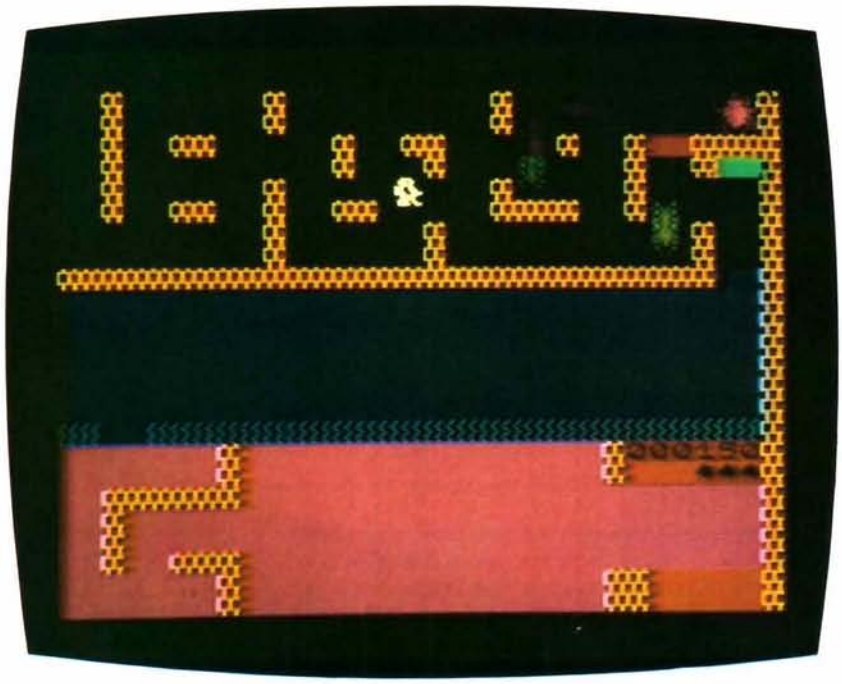
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Presentiamo altri due programmi che appartengono, solo alla lontana, alla categoria degli "adventure games": in questi due giochi, infatti, il protagonista non controlla gli avvenimenti come nelle vere e proprie "avventure". Il divertimento tuttavia è assicurato.



Buried Bucks per l'Atari

Il giocatore pilota un elicottero con lo scopo di raccogliere quanto più denaro possibile (il titolo del gioco significa letteralmente "Dollari sepolti"). In primo luogo occorre individuare dove siano sotterrati i soldi (in genere sotto tonnellate di terra) e, per far ciò occorre scavare lunghe gallerie; ma, proprio quando sembra di aver raggiunto il tesoro, ecco comparire degli aeroplani che provvedono a riempire di terra gli scavi! Anche altri ostacoli attendono l'intrepido pilota, per rendere più avvincente il tutto. Il gioco è stato realizzato dalla Analogue Software.



Styx per lo Spectrum 48K

Si tratta di un duello contro le forze del male e l'obiettivo è quello di scendere nell'Ade per distruggere il capo degli inferi, chiamato "Dark One". Col solo aiuto di una spada laser, si devono uccidere enormi ragni che infestano le gallerie, attraversare lo Stige (nel quale, ovviamente, vive ogni sorta di creature letali) e combattere contro gli spiriti maligni. Il gioco è divertente, ma a lungo andare diventa facile superare i numerosi ostacoli, a scapito della vivacità dell'azione. È un prodotto della Bug-Byte.

Soluzioni agli esercizi

Confrontiamo le nostre risposte con quelle riportate qui sotto

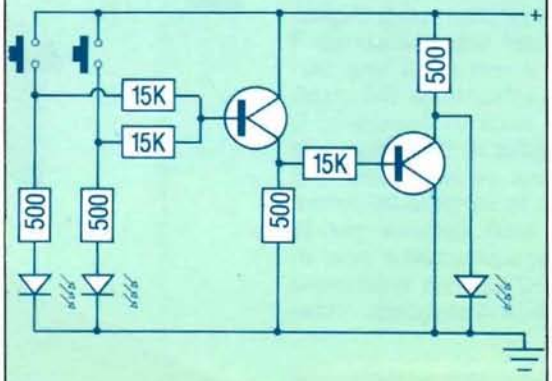
Esistono svariati modi di risolvere un problema, quindi, quelli qui proposti non sono gli unici.

1) Resistenze

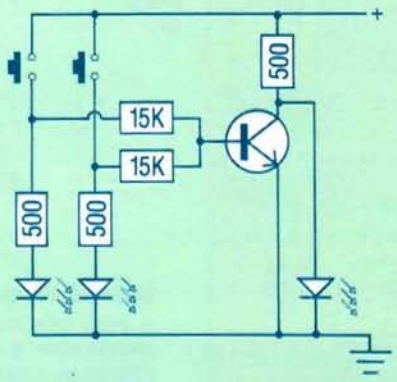
I valori delle resistenze illustrate sono: a) 6400 K-Ohm e b) 150 K-Ohm. Le bande colorate di una resistenza da 150 Ohm sono: marrone-verde-marrone

2) Porta NOR

Il metodo più ovvio per realizzare una porta NOR consiste nel combinare assieme i due circuiti OR e NOT come mostrato in figura:

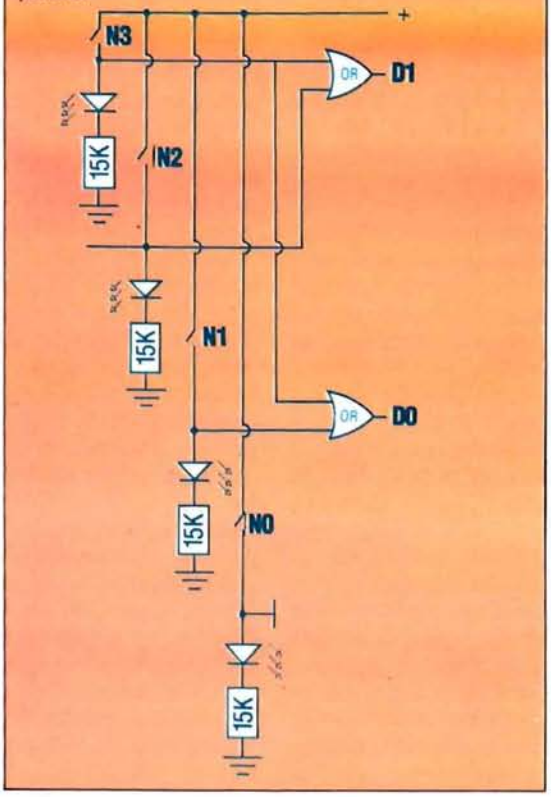


Tuttavia, la procedura più breve, consiste nell'usare il circuito per una porta OR, prelevando il segnale di uscita sul collettore del transistor, anziché dall'emettitore, in modo simile al circuito della porta NOT:



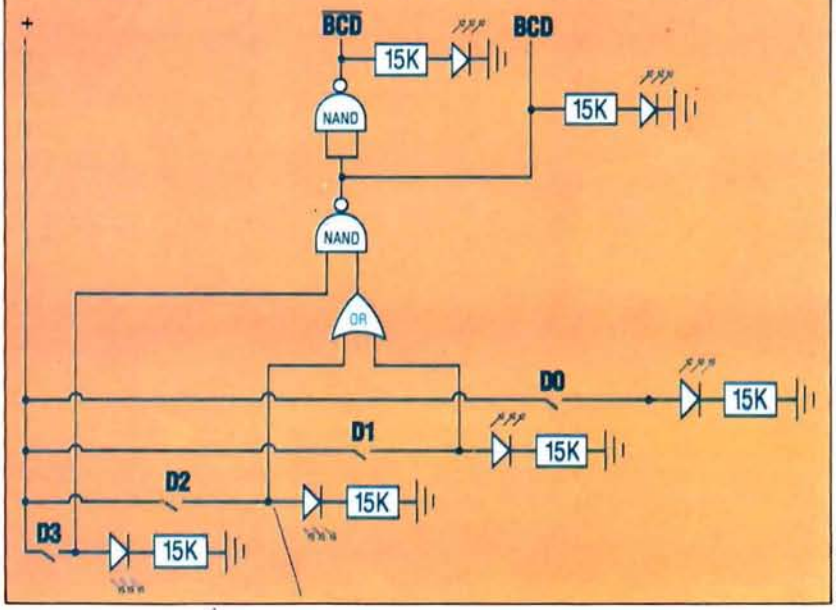
3) Convertitore da decimale in binario

Occorre un solo circuito integrato, ossia un chip TTL con quattro porte OR



4) Convalidatore per codice BCD

I codici BCD validi sono compresi nella gamma che va da 0000 a 1001 mentre i codici non validi sono compresi tra 1010 e 1111. Ecco un modo per realizzare il circuito del convalidatore:



LIZ DIXON

Lo Sharp PC-5000

Uno dei portatili più sofisticati, dotato di memoria a bolle

I computer portatili si presentano sotto svariate forme, ma la linea che recentemente ha riscosso maggior successo è stata quella dei piccoli portatili compatti, quali il Modello 100 TRS-80 e l'Epson HX-20. Questi apparecchi sono dotati di uno schermo LCD e possono funzionare anche a pile: lo Sharp PC-5000 è il modello più costoso della categoria, ma occorre tener conto che la sua memoria di massa è realizzata con la nuova tecnologia delle memorie a bolle.

Lo sviluppo della memoria a bolle, che risale ad alcuni anni fa, ha causato un notevole scompiglio nell'industria dei computer, senza però riscuotere il successo previsto; poche case produttrici, infatti, sono state capaci di superare i problemi di velocità ed affidabilità che questa nuova tecnologia comporta. La Sharp sembra invece averli risolti. La memoria a bolle consuma poca energia e permette di immagazzinare grandi quantità di dati in uno spazio molto limitato: l'ideale, dunque, per un portatile.

Il PC-5000 sembra una piccola macchina da scrivere elettrica; sollevato il coperchio, che ruota su due piccoli cardini, appare una comoda tastiera, dotata di otto tasti di funzione definibili dall'utente e di quattro tasti per il controllo del cursore, allineati nella parte alta della tastiera. Il coperchio incernierato contiene lo schermo a LCD: è piuttosto pesante e viene mantenuto in posizione da un meccanismo che permette di dare allo schermo l'angolazione più confortevole. Sopra la tastiera c'è un pannello con tre piccole "spie" LED (alimentazione, batteria scarica, memoria a bolle) e un alloggiamento per la cartuccia di memoria a bolle.

La parte posteriore di questo pannello è predisposta per l'alloggiamento della stampante (accessorio opzionale). La stampante ha una forma a parallelepipedo, che si innesta perfettamente nell'apposito vano del portatile. Si tratta di una stampante termica, con una velocità di 37 caratteri al secondo la cui stampa è di buona qualità, sebbene la carta termica non sia, di per sé, molto elegante.

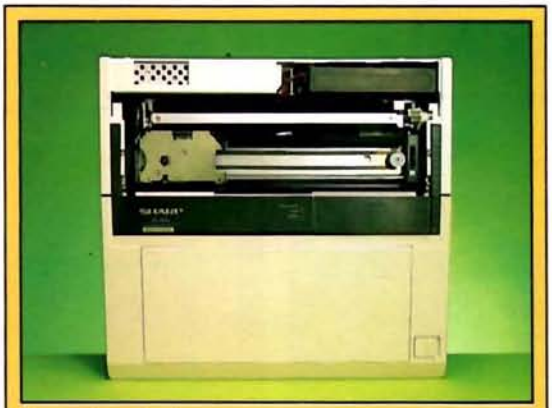
Lo Sharp PC-5000 possiede 128 Kbyte di memoria utente e 192 Kbyte di ROM, incluso il Microsoft GW BASIC, la stessa versione usata sull'IBM PC. La CPU è un microprocessore Intel 8088 (lo stesso del PC IBM) ed il sistema operativo adottato è l'MS-DOS. Il computer indirizza la memoria a bolle come se fosse un'unità a due dischetti (A e B) di tipo tradizionale. La Sharp produce anche una vera e propria unità a due dischetti, i cui codici d'indirizzo sono C e D, da innestare sul retro del PC-5000; l'unità a dischetti, però, non funziona a pile.

Questo apparecchio è corredato di diversi pacchetti software prodotti dalla Sorcim, tra cui: il SuperCalc, il SuperWriter ed il SuperComm, un programma per comunicazioni. I programmi sono memorizzati su di una cartuccia a bolle e vengono selezionati dal menù di sistema premendo i tasti di funzione. Il significato di ciascun tasto di funzione può essere specificato da una serie di etichette che appaiono sull'ultima linea dello schermo.

La Sharp è stata per anni famosa per la buona qualità sia di progettazione che di fabbricazione e con il PC-5000 ha realizzato un potente computer dalle dimensioni veramente contenute.



L'unità a due dischetti
Questo apparecchio contiene due unità a dischetti da 320 Kbyte, che funzionano sotto MS-DOS. Viene innestato sul retro del PC-5000, ma non può essere alimentato a pile. Il software, scritto per altre macchine MS-DOS, deve essere adattato per il display ad 8 linee di questo computer



Stampante opzionale
Guardando all'interno del PC-5000 si può vedere come la stampante sia inserita nell'apposito alloggiamento. Essa si adatta al contenitore e viene collegata alla scheda di sistema mediante un cavetto a nastro. Ha una velocità di 37 caratteri per secondo e produce stampe di buona qualità su carta termica

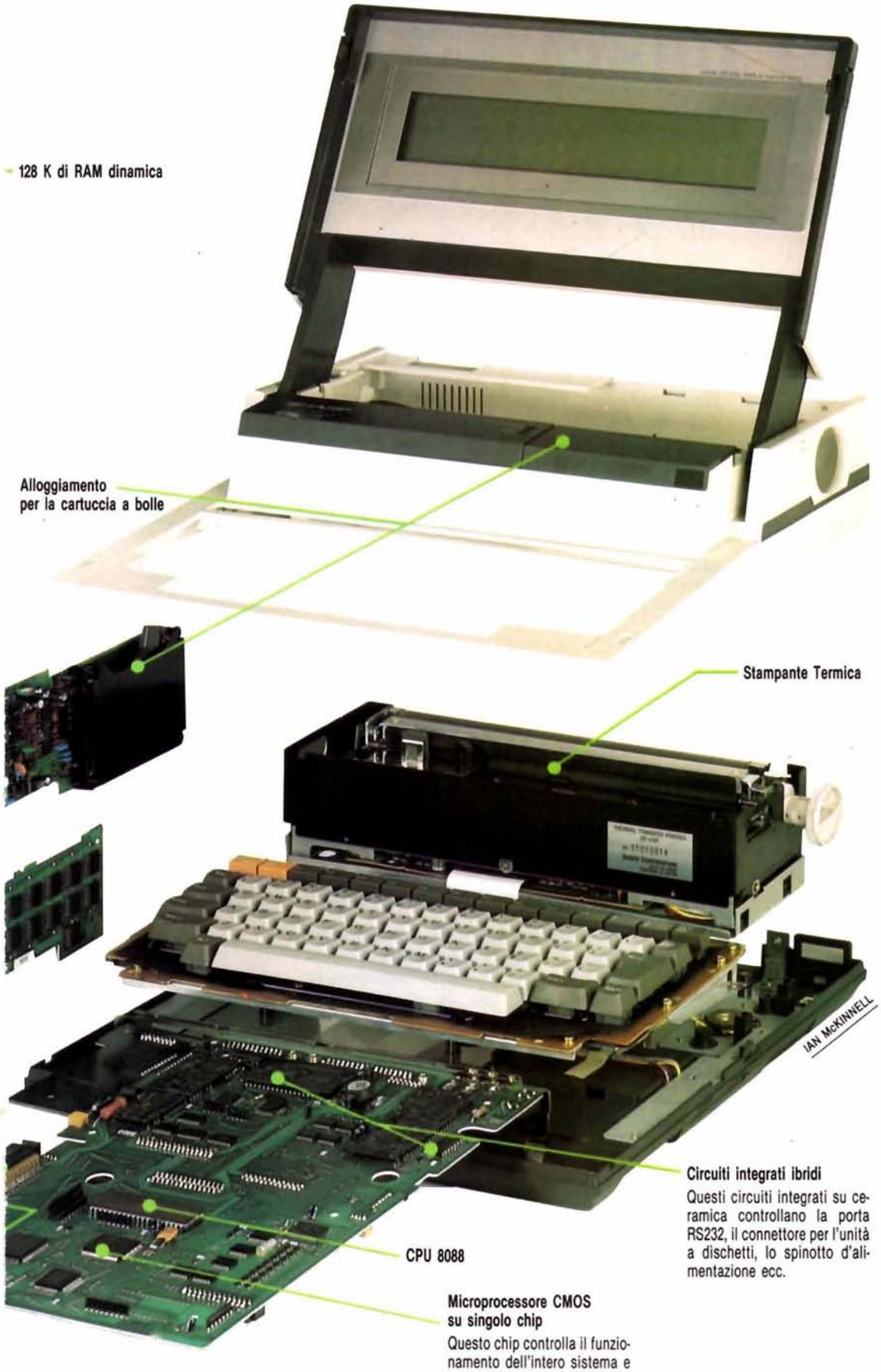
IAN McKINNELL



Scheda di controllo della memoria a bolle
Questa scheda, pur funzionando come una scheda di controllo di un'unità a dischetti, controlla gli ingressi e le uscite del connettore per le cartucce a bolle

Scheda ROM
Questa scheda possiede 128 K di ROM nelle quali si trova l'MS-DOS, il generatore di caratteri per il video e per la stampante, alcuni sistemi I/O e di comunicazione ed il PISCS. Quest'ultimo è un chip che traduce i segnali provenienti dalle cartucce a bolle in un codice comprensibile all'MS-DOS, come se si trattasse di normali unità a dischetti

Connettori per espansione
A questi connettori vengono collegati moduli di espansione RAM oppure software registrato su cartucce ROM



SHARP PC-5000

DIMENSIONI

mm. 300 x 318 x 90

PESO

5,7 kg (stampante compresa)

CPU

8088 Intel a 16 bit

MEMORIA

128 Kbyte di RAM
192 Kbyte di ROM

VIDEO

A cristalli liquidi (LCD) con 80 caratteri su 8 linee. Risoluzione grafica di 640 x 80 punti. Simboli grafici e caratteri internazionali incorporati

INTERFACCE

Per registratore a cassette, per dischetti, per modem. Interfaccia RS232, alimentatore da rete, porte per espansioni ROM/RAM

LINGUAGGI FORNITI

BASIC GW della Microsoft

TASTIERA

Tipo macchina da scrivere, con 57 tasti più 8 di funzione, controllo del cursore e tre altri tasti speciali. La funzione dei tasti programmabili può apparire sull'ultima riga del display

DOCUMENTAZIONE

I manuali in dotazione spiegano ampiamente come far funzionare il PC-5000 ed il sistema operativo MS-DOS. Il manuale del BASIC, che forse è un po' troppo tecnico per i principianti, è stato redatto dalla Microsoft

PRO

Il PC-5000 offre parecchie prestazioni di microcomputer di dimensioni molto maggiori. La tecnica della memoria a bolle costituisce una grande innovazione rispetto alla concorrenza

CONTRO

L'unico appunto da fare, considerato l'alto livello qualitativo dell'apparecchio, può essere la relativa lentezza della memoria a bolle per chi sia abituato a lavorare coi dischetti. Tuttavia, questo è un fatto trascurabile rispetto alle prestazioni

Resoconto finale

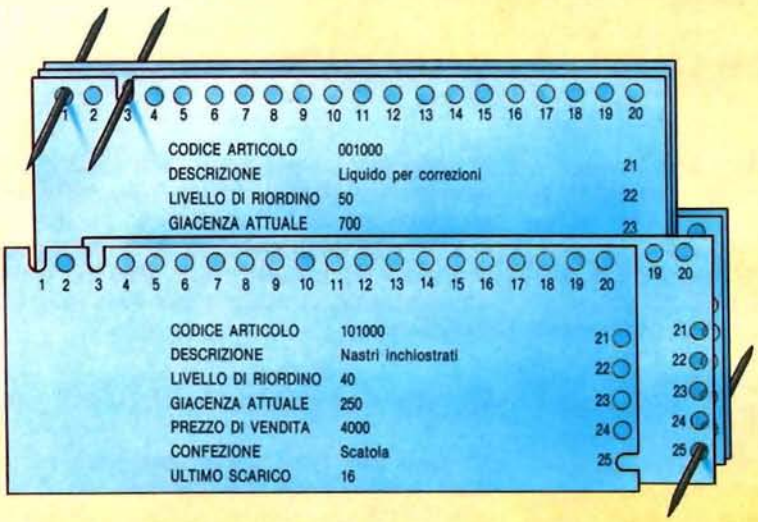
Nell'elaborazione dei dati e nella loro analisi il computer è veramente imbattibile specialmente in velocità

Ai responsabili per l'organizzazione di un magazzino occorre tutta una serie di informazioni su ciascun articolo in giacenza. È necessario avere una precisa descrizione di tutti i prodotti immagazzinati e dei loro movimenti.
Questo risultato si può ottenere in due modi: interrogando il sistema computerizzato attraverso il terminale video, oppure tramite resoconti stampati.
Un buon programma deve prevedere tutta una serie di opzioni per la gestione delle informazioni e per la loro visualizzazione in modo razionale.
Un resoconto informativo fondamentale è quello riferito agli articoli a lenta rotazione. Poiché a tutti i movimenti di carico e di scarico del magazzino è associata la data dell'operazione, il programma possiede già gli elementi necessari per produrre un resoconto cronologico su particolari gruppi di articoli. Basta che il sistema esamini l'archivio dei movimenti confrontando la registrazione della data. Poiché il significato del termine "lenta rotazione" varia a seconda del tipo di attività (e ciò che è lento in un caso può non esserlo in un altro), il resoconto deve permettere all'utente di specificare con precisione il parametro da assegnare a questo termine.
Questo obiettivo è contemplato, ad esempio, nel programma "Dragon Data": indicando una data (ad esempio, 150485 per 15 Aprile 1985) il sistema dispone di un termine di confronto in base al quale estrarre (e visualizzare) tutte le informazioni relative agli articoli che non sono stati movimentati da una certa data in poi. Questa procedura risulta estremamente utile e versatile: si possono ottenere più resoconti, riferiti a periodi diversi, semplicemente indicando una nuova data. L'unica limitazione, del resto ovvia, è che la data deve riferirsi ad un periodo interno all'intervallo delle registrazioni, altrimenti comparirà un messaggio del tipo: "Nessun articolo all'interno del criterio di selezione".
Questo genere di resoconti può presentare una limitazione, qualora il programma consideri soltanto gli articoli di magazzino che non sono stati soggetti a nessuna transazione. Infatti, se un articolo avesse appena un paio di movimenti durante l'arco di svariati mesi, esso andrebbe qualificato come a lenta rotazione e pertanto incluso nel resoconto.
Un sistema più sofisticato potrebbe prevedere una maggiore flessibilità.
Due sono le vie per raggiungere questo obiettivo: il sistema potrebbe disporre di un ulteriore parametro di selezione, diverso da quello della data (ad esempio, un livello minimo di transazioni). Se il numero di transazioni è al di sotto di quello specificato l'articolo in questione verrebbe incluso nella lista degli articoli a lenta rotazione. In alternativa, il sistema stesso potrebbe leggere e confrontare i movimenti avvenuti su tutte le linee di prodotti e produrre un elenco relativo alle 50 linee più lente, quindi alle 50 successive e così via.

Le videate

In genere sullo schermo vengono visualizzate le informazioni relative ad un solo articolo alla volta. Di conseguenza, se molti sono gli articoli a lenta rotazione, il passaggio da una pagina all'altra della lista provoca enormi perdite di tempo.
In casi come questo, la possibilità di ottenere tabulati stampati, di più rapida e facile consultazione, può essere addirittura indispensabile.
L'analista ed il programmatore del software dovrebbero avere sempre ben presente questo tipo di problemi durante lo sviluppo di pacchetti commerciali per la gestione del magazzino.
Le videate relative ai movimenti di magazzino offrono molte altre informazioni importanti per l'utente: il valore di magazzino (che permette all'utente di conoscere il capitale vincolato ai beni in giacenza), il consumo medio mensile, le giacenze attuali di magazzino e la data dell'ultima vendita.
Grazie a queste informazioni l'utente può operare una più oculata rotazione degli articoli in magazzino, magari promuovendo le vendite di un determinato articolo, se necessario, mediante un forte sconto.
La possibilità di ottenere informazioni basate su particolari criteri di selezione (ad esempio, la data) è di vitale importanza quando si devono analizzare ed elaborare dati sull'impostazione dinamica di un magazzino. Ovviamente, oltre a stampare un dettagliato elenco di tutti i movimenti per ciascun articolo, occorre che il sistema sia in grado di

SCHEDE PERFORATE



Articoli a lenta rotazione

Il Sistema di Gestione di Magazzino della Dragon Data si avvale di un database per generare una serie di resoconti sulla situazione delle giacenze. Tenere grandi scorte di articoli a lenta rotazione implica dei costi non indifferenti, occorre quindi tenere la situazione sotto controllo. Nell'esempio riportato a fianco, viene effettuata una ricerca, all'interno del gruppo 101, degli articoli senza movimenti da più di 15 giorni. Ne risulta che le giacenze di nastri inchiostrati sono molto alte e la rotazione di vendita è molto lenta: è necessario ridurre rapidamente le scorte di questo articolo

Sul margine

Un sistema a schede perforate costituisce una semplice forma di database. Sul bordo delle schede sono praticati dei fori che, all'occorrenza, possono venir "aperti". I fori si possono considerare come numeri binari: un foro significa 0, un'apertura significa 1. Ad esempio, le schede possono riferirsi ad un archivio di magazzino: i fori da 1 a 3 indicano i gruppi di prodotto, quelli da 4 a 6 rappresentano il codice di listino, cosicché i fori da 1 a 6 indicano, per intero, il codice dell'articolo. I fori da 21 a 25 indicano il numero di giorni trascorsi dall'ultimo scarico dell'articolo (nell'esempio, il foro 25 corrisponde a 15 giorni)



KEVIN JONES

indicare con precisione le operazioni effettuate su specifici gruppi di articoli durante un particolare lasso di tempo.

Il programma per il Dragon, ad esempio, prevede questo genere di analisi, invitando l'operatore ad inserire i valori minimo e massimo di una serie di articoli di magazzino e la data iniziale e finale del periodo da esaminare.

Oltre a vagliare tutte le transazioni effettuate su una particolare serie di articoli, l'utente può anche richiedere un'analisi comparata con gli articoli raggruppati in base al tipo di transazione (fatture di vendita, rettifiche di magazzino e via dicendo). Uno dei menu del Dragon Data offre tre modalità di selezione dei movimenti in base alla data: solo quelle del corrente periodo, tutte le transazioni, le transazioni avvenute in uno specifico periodo. Inoltre, è anche consentito, mediante un sottomenu, di suddividere la selezione per tipo di transazione. Questo sottomenu presenta anche l'opzione "99 Tutti i tipi di transazione", che permette una stampa generale di tutti i tipi di transazione che rispondono ai parametri specificati.

Il programma dà all'utente la facoltà di associare gli articoli di magazzino a particolari gruppi di prodotto: ciò consente di prendere in esame i gruppi di prodotto e la loro descrizione e di conoscere le scorte di magazzino di un particolare gruppo di prodotti. I resoconti su schermo sono utili, ma hanno il difetto di non essere permanenti.

Per quanto riguarda la stampa di tabulati, la possibilità che il Dragon Data ha di ottenere resoconti ricalca per molti aspetti il formato adottato nella presentazione su schermo.

Molti sono i motivi per cui il programma per il Dragon è considerato un sistema di magazzino piuttosto ridotto, nonostante le relative sofisticate procedure analitiche. Uno fra questi è che il package è stato sviluppato come un programma "a sé stante": non è quindi in grado di collegarsi ad altri programmi commerciali, che potrebbero sfruttare convenientemente le informazioni contenute nei suoi archivi. Ad esempio, non è possibile agganciare la gestione del magazzino con quella degli ordini. Considerando lo stretto legame tra questi due aspetti gestionali, un sistema che obblighi a riempire questa lacuna con metodi semimanuali è decisamente sconsigliabile. Diviene decisamente laborioso, infatti, controllare se la disponibilità di magazzino riesce a coprire le ordinazioni ricevute dai clienti, che magari riguardano un elevato numero di articoli tutti diversi tra loro.

Con ciò, abbiamo offerto una panoramica sull'uso degli home computer per l'elaborazione dei dati nelle piccole attività commerciali. Dovrebbe adesso risultare chiaro che, sebbene questi pacchetti software possano offrire, in teoria, sistemi integrati, in realtà i programmi sono spesso orientati verso aspetti particolari dell'attività commerciale quali, ad esempio, le registrazioni di prima nota, ma ne trascurano altri (la gestione degli ordini, l'inventario, la formulazione di bilanci e così via).

Simili programmi sono forse più utili per fornire informazioni supplementari agli operatori commerciali, che non costituire il nucleo sul quale impostare la propria attività. Nonostante ciò, i pacchetti commerciali per home computer possono senz'altro accrescere l'efficienza in attività commerciali di ridotte dimensioni.



La grafica del Commodore

Mettiamo in azione le portentose capacità grafiche del Commodore

Uno dei pregi del Commodore 64 è la possibilità di scrivere giochi del tipo "arcade", in BASIC. Ciò è possibile grazie alla presenza degli sprite, ossia simboli ad alta risoluzione definibili dall'utente e facilmente manipolabili sullo schermo. La memoria del Commodore è dotata di speciali registri, che controllano le caratteristiche degli sprite (ad esempio: il colore, le dimensioni e la posizione sullo schermo). Depositando, con il comando POKE, opportuni valori in questi registri, il programmatore può facilmente controllare il movimento degli sprite. Il gioco *Caccia ai Sottomarini*, che verrà sviluppato nell'arco di più lezioni, utilizza quattro sprite, degli otto disponibili sul Commodore 64, per visualizzare la nave, il sottomarino, le bombe di profondità lanciate dalla nave e l'esplosione del sottomarino.

Il gioco può quindi essere progettato in fasi successive: ciascuna sezione verrà scritta sotto forma di subroutine, richiamata dal programma principale. Questo tipo di struttura permette di individuare gli errori e rende molto più semplice aggiungere nuove sezioni al programma.

Il funzionamento del gioco è probabilmente noto a molti: il giocatore è al comando di una nave, a caccia di sottomarini che si muovono sullo schermo a velocità e profondità diverse, mentre la nave lancia contro di essi cariche di profondità. Il punteggio del giocatore aumenta ogni volta che un sottomarino viene colpito ed il valore dell'obiettivo annientato dipende dalla sua velocità e profondità: più il sottomarino è veloce e maggiore è la sua profondità, più alto è il punteggio. Il valore dei sottomarini che sfuggono alla caccia viene sottratto al punteggio del giocatore. La nave viene guidata da tastiera, usando i tasti Z e X per muoverla orizzontalmente verso sinistra e verso destra. Le cariche di profondità vengono sganciate battendo il tasto M; sullo schermo appare anche un segnatempo che permette di giocare per tre minuti. Alla fine del tempo, il computer chiede al giocatore se vuole fare un'altra partita e viene tenuta una registrazione dei più alti punteggi ottenuti.

La prima regola da osservare, quando si progetta un gioco di animazione in BASIC, è mantenere il corpo centrale del programma il più breve possibile. Il programma Caccia ai Sottomarini utilizza delle subroutine per realizzare la maggior parte delle procedure che compongono il gioco. Le sole procedure controllate direttamente dal programma principale sono: l'aggiornamento del contatempo, la rilevazione di un input dalla tastiera ed i movimenti della nave e dei sottomarini.

Il progetto di massima del programma è abbastanza generico e può essere adottato su qualsiasi tipo di computer, mentre molti dei dettagli variano a seconda del computer utilizzato. In questa prima fase del progetto esamineremo la routine che crea lo scenario del gioco.

Scena del gioco

Ci sono due modi per visualizzare caratteri sullo schermo del Commodore 64: il primo usa l'istruzione PRINT, il secondo permette di depositare, mediante il comando POKE, dei numeri nelle appropriate locazioni dell'area di memoria video. Noi useremo entrambi questi metodi per creare la scena del gioco. Lo schermo del Commodore è composto da 25 righe di 40 caratteri ciascuna: in altre parole ci sono 1.000 posizioni di schermo a disposizione. Ad ogni posizione è associata una coppia di numeri: il primo indica al computer quale carattere visualizzare nella particolare posizione, il secondo indica quale colore usare per il carattere. Ci sono quindi due blocchi di memoria di 1.000 locazioni ciascuno: uno per i codici dei caratteri e l'altro per i codici di colore. L'area sulla quale sono registrati i codici dei caratteri è detta "memoria video" ed è compresa tra le locazioni 1024 e 2023, mentre la memoria di colore ha come locazione iniziale 55296 e finale 56295.

Ogni carattere possiede un proprio codice ed un elenco completo è riportato nel manuale per il Commodore 64.

Il Commodore può visualizzare fino a 16 colori, i cui codici sono:

0	nero	8	arancione
1	bianco	9	marrone
2	rosso	10	rosso chiaro
3	azzurro	11	grigio 1
4	porpora	12	grigio 2
5	verde	13	verde chiaro
6	blu	14	blu chiaro
7	giallo	15	grigio 3

Oltre ai blocchi di memoria appena citati, ci sono altre due locazioni di particolare interesse: la 53280 (che controlla il colore del contorno) e la 53281 (che controlla il colore dello schermo). Per progettare il paesaggio marino del nostro gioco, le sei righe superiori dello schermo devono essere blu chiaro per il cielo, mentre il resto dello schermo ed il contorno saranno blu scuro per rappresentare il mare, eccetto le due righe in fondo allo schermo che rappresenteranno il fondo marino.

Per assegnare alle righe dello schermo il colore blu chiaro e al bordo il blu scuro, il programma dovrà contenere le due seguenti POKE:

```
POKE 53281,14:POKE 53280,6
```

La settima riga dello schermo parte dalla locazione 1264, la penultima dalla locazione 1944. Possiamo colorare il mare inserendo con una POKE il codice video per uno spazio inverso (cioè un carattere spazio colorato in contrasto) nelle locazioni comprese tra 1246 e 1943 e depositando il codice di colore 6 nelle corrispondenti locazioni di memoria di colore. Il modo più semplice per associare una locazione della memoria di colore con la corrispondente locazione di memoria video è sommare il numero 54272 alla locazione di memoria video.
Il codice video per un carattere "spazio" è 32. Un codice video per un carattere inverso può essere calcolato sommando 128 al normale codice video, perciò il valore per uno "spazio" inverso sarà 32+128=160. Le seguenti linee di programma utilizzano un semplice ciclo FOR...NEXT per colorare il mare:

```
FOR I=1264 TO 1943
POKE I,160:POKE I+54272,6
NEXT I
```

Il fondale marino si ottiene con due righe composte da un carattere quadrettato (codice video = 102), colorato in marrone. Anche in questo caso si ricorre ad un ciclo FOR...NEXT:

```
FOR I=1944 TO 2023
POKE I,102:POKE I+54272,9
NEXT I
```

Anche l'istruzione PRINT è comoda per produrre immagini sullo schermo. Il colore e la posizione del cursore si possono definire nella stessa istruzione PRINT usando gli speciali caratteri di controllo del Commodore oppure adoperando i codici CHR$. Noi useremo quest'ultimo metodo perché si dimostra più facile da leggere nel listato del programma. Una lista completa dei codici CHR$ è riportata nell'Appendice F del manuale del Commodore. Vediamo ora quali sono i codici relativi al colore e alla posizione del cursore:

CHR$(5)	bianco
CHR$(144)	nero
CHR$(147)	cancella lo schermo e posiziona il cursore nell'angolo in alto a sinistra
CHR$(19)	posiziona il cursore nell'angolo in alto a sinistra
CHR$(17)	muove il cursore verso il basso
CHR$(145)	muove il cursore verso l'alto
CHR$(157)	muove il cursore verso sinistra
CHR$(29)	muove il cursore verso destra

Come parte della nostra routine di creazione dello schermo, PUNTEGGIO e PUNTEGGIO MASSIMO devono essere visualizzati sulla prima linea dello schermo. CHR$(19) serve a portare il cursore all'inizio della prima linea. Il seguente comando fa stampare il punteggio iniziale in nero:



Comincia la caccia...
Il programma che stiamo creando è un classico gioco grafico per computer, nel quale si bombardano sottomarini in profondità. La grafica a sprite del Commodore 64 viene usata per dare animazione alla nave ed ai sommergibili

IAN McKINNELL

```
PRINT CHR$(19);CHR$(144);"PUNTI 000"
```

Anche il PUNTEGGIO MASSIMO deve comparire sulla prima linea, ma nell'angolo destro. La funzione SPC permette di inserire un certo numero di spazi. Possiamo modificare il comando PRINT per visualizzare anche il PUNTEGGIO MASSIMO:

```
PRINT CHR$(19);CHR$(144);"PUNTI 000";SPC(16);"MAX. 000"
```

La preparazione dello schermo verrà eseguita da una subroutine, con inizio alla linea 1000. Essa contiene anche una POKE, che abilita la ripetizione automatica di ogni tasto, se tenuto premuto. Questa subroutine può essere verificata mediante le seguenti linee di programma:

Dopo aver trascritto il programma, è bene memorizzarlo su cassetta o su dischetto, prima di eseguirlo.

```
10 GOSUB 1000:REM PREPARAZIONE SCHERMO
20 END
1000 REM * * * PREPARAZIONE SCHERMO * * *
1010 PRINT CHR$(147):REM PULIZIA SCHERMO
1020 :
1030 REM * * * COLORAZIONE DEL MARE * * *
1040 POKE 53281,14:POKE 53280,6
1050 FOR I=1264 TO 1943
1060 POKE I,160:POKE I+54272,6
1070 NEXT
1080 :
1090 REM * * * FONDO MARINO * * *
1100 FOR I=1944 TO 2023
1110 POKE I,102:POKE I+54272,9
1120 NEXT
1130 POKE 650,128:REM RIPETIZ. AUTOM.
1140 :
1150 REM * * * PUNTEGGIO * * *
1160 PRINT CHR$(19);CHR$(144);"PUNTI 000";SPC(16);"MAX. 000"
1170 RETURN
1180 :
1190 :
```

Etichette e cicli

Le basi per realizzare cicli iterativi in linguaggio Assembly

Prima di cominciare ad usare nel modo migliore le varie procedure d'indirizzamento della CPU (soprattutto gli indirizzi indicizzati) dobbiamo imparare come realizzare i cicli o "loop". Senza la conoscenza di questa fondamentale struttura, ci troveremmo nelle stesse condizioni di un programmatore BASIC che ha una nozione generale dell'insieme delle istruzioni, ma non conosce il comando FOR...NEXT. Nel linguaggio Assembly non ci sono strutture automatiche come FOR...NEXT (anche se lo Z80 ne possiede una molto simile), ma è possibile costruire dei cicli chiusi del tipo IF...THEN GOTO. Occorrono, per questi, comandi capaci di produrre decisioni o esprimere condizioni e di cambiare l'ordine di esecuzione delle istruzioni che compongono il programma.

Nel linguaggio Assembly la capacità decisionale si incentra sugli "indicatori di condizione", contenuti nel registro di stato del processore: Essi evidenziano gli effetti sull'accumulatore dell'ultima operazione eseguita. Tutti questi indicatori possono essere utilizzati nella formazione di una decisione ma, per adesso, ne esamineremo soltanto due: lo zero (Z) ed il riporto o "carry" (C).

Il valore di questi indicatori può essere utilizzato per stabilire se il processore deve eseguire la successiva istruzione del programma oppure saltare ad un'altra: il processore, normalmente, esegue le istruzioni sequenzialmente, salvo l'indirizzo contenuto nel registro contatore di programma venga modificato. Questo registro contiene sempre l'indirizzo della successiva istruzione in codice macchina da eseguire. Quando il processore ne inizia l'esecuzione, carica l'op-code dell'istruzione dal byte indicato dall'indirizzo che si trova nel contatore di programma. Un nuovo indirizzo, calcolato sommando al precedente la lunghezza dell'istruzione in via d'esecuzione, viene depositato nel contatore, puntando così all'op-code dell'istruzione successiva. Se l'istruzione in esecuzione ordina al contatore di puntare ad un indirizzo posto altrove nel programma, allora si verificherà un "salto".

Sul 6502, l'istruzione BEQ provoca un cambiamento nel contatore di programma se l'indicatore zero ha valore 1. BCS produce lo stesso risultato qualora sia il carry a valere 1. Sullo Z80, queste istruzioni sono rispettivamente JR Z e JR C. Questi quattro op-code vengono chiamati "istruzioni di salto", poiché rappresentano un punto di diramazione nel flusso del programma. Il loro operando è un numero a singolo byte che viene sommato all'indirizzo nel contatore di programma per ottenere un nuovo indirizzo. Vediamo ora cosa accade quando viene eseguito il seguente programma:

ORG $5E00			
6502		Z80	
5E00	ADC #$34	ADC	A,$34
5E02	BEQ $03	JR	Z,$03
5E04	STA $5E20	LD	($5E20),A
5E07	RTS	RET	

Se l'istruzione ADC (locazione $5E00) produce un risultato di 0 nell'accumulatore (caso improbabile ma, come vedremo più avanti, possibile), allora le istruzioni BEQ e JR Z (indirizzo $5E02) faranno sì che $03 venga sommato al contenuto del contatore di programma. L'esecuzione passerà dunque all'istruzione di ritorno, situata all'indirizzo $5E07 (viene saltata quella con indirizzo $E04).

A prima vista questo procedimento può sembrare errato. Infatti, se l'istruzione a $5E02 fa sì che $03 venga sommato al contatore di programma, l'indirizzo nel contatore non dovrebbe essere $5E05? Occorre però ricordare che il contatore di programma punta sempre all'istruzione successiva e non a quella in corso d'esecuzione. Pertanto, quando inizia l'esecuzione dell'istruzione a $5E02, il contatore di programma conterrà l'indirizzo $5E04, la locazione dell'istruzione seguente. Sommando $03 a $5E04, abbiamo proprio $5E07, ossia l'indirizzo dell'istruzione che verrà eseguita subito dopo.

È importante sottolineare che il processore non è in grado di verificare se gli indirizzi a cui punta sono corretti. Se, inavvertitamente, indicassimo $02 nell'istruzione di salto, il contatore di programma verrà incrementato (se l'accumulatore contiene 0) di $02 ed il processore accetterà $5E06 come indirizzo dell'op-code dell'istruzione successiva. Nel nostro programma, $5E06 contiene il valore $5E, che è il byte alto dell'operando contenuto nell'istruzione a $5E04. Il processore non è tuttavia in grado di valutare se questa istruzione è giusta o meno: riterrà quindi $5E un op-code valido se lo eseguirà trattando i byte che seguono $5E06 come operandi dell'istruzione. Come risultato, il programma fallisce e potrà essere necessario "resettare" il computer. Simili errori di calcolo sono i più comuni nella programmazione in codice macchina.

Nella programmazione in linguaggio Assembly, il calcolo dei salti ha pochi inconvenienti, in quanto eseguito direttamente dall'assemblatore. Cosicché, invece di fornire un valore esadecimale come operando dell'istruzione di salto, si indica l'indirizzo simbolico dell'istruzione a cui saltare. Questa procedura, inol-

tre, rende molto più chiari e comprensibili i listati. L'assemblatore traduce l'indirizzo simbolico in un indirizzo assoluto, calcola lo spostamento necessario per arrivare all'indirizzo cui saltare e deposita il valore di spostamento nell'istruzione in codice macchina. L'indirizzo simbolico viene chiamato "etichetta" ed è analogo al numero di linea di un programma BASIC.
Esaminiamo con più attenzione l'uso delle etichette. Un'etichetta (o label) non è che una stringa alfanumerica scritta all'inizio di una linea di programma Assembly. L'assemblatore considera l'etichetta come un valore simbolico a due byte, che indica l'indirizzo del primo byte dell'istruzione. Possiamo quindi riscrivere il programma nel modo seguente:

ORG $5E00		
	6502	Z80
5E00	ADC #$34	ADC A,$34
5E02	BEQ EXIT	JR Z,EXIT
5E04	STA $5E20	LD ($5E20),A
5E07 EXIT	RTS	RET

L'istruzione con indirizzo $5E02 può essere così interpretata: "Se (IF) il valore dell'accumulatore è 0, saltare (THEN GOTO) all'indirizzo rappresentato dall'etichetta EXIT". Il programma appare ora molto più comprensibile e di conseguenza si riducono le probabilità di commettere errori nel calcolare l'indirizzo d'arrivo del salto.
Possiamo ora utilizzare le etichette e le istruzioni di salto per creare un ciclo chiuso:

ORG $5E00		
	6502	Z80
5E00 START	ADC #$34	ADC A,$34
5E02	BNE START	JR NZ,START
5E04	STA $5E20	LD ($5E20),A
5E07 EXIT	RTS	RET

Si noti l'uso della nuova etichetta START e delle due nuove istruzioni di salto. Sia BNE che JR NZ significano: "Salta se l'accumulatore è DIVERSO da 0". Vediamo adesso quale sarà il risultato di questo programma: innanzi tutto sommerà $34 all'accumulatore. Se il risultato è diverso da 0, il programma salta indietro all'indirizzo $5E00 cioè l'indirizzo rappresentato dall'etichetta START. $34 sarà nuovamente aggiunto all'accumulatore ed il risultato dirà se deve essere compiuto un nuovo salto. Questo ciclo chiuso si ripete finché si verifica la condizione di salto. Quando il contenuto dell'accumulatore diviene uguale a zero, grazie all'istruzione ADC, l'esecuzione non passerà più all'indirizzo $5E02, ma verrà invece eseguita l'istruzione successiva, con indirizzo $5E04.
Questa procedura differisce ben poco dalla IF...THEN GOTO... del BASIC. Tuttavia, resta da capire come il valore contenuto nell'accumulatore possa divenire 0, se esso viene incrementato di $34 ad ogni iterazione del ciclo. La ragione è semplice: l'accumulatore è un registro a singolo byte e, se il risultato della somma è un numero a due byte, l'indicatore di riporto del registro di stato del processore viene posto ad 1 e nell'accumulatore troviamo il byte basso del risultato. Se l'accumulatore contiene, ad esempio, $CC, sommando $34 otteniamo il numero a due byte $0100. Come risultato, l'indicatore di riporto è posto a 1, nell'accumulatore finisce il byte basso del risultato, cioè $00, e di conseguenza l'indicatore di zero assume il valore 1.
Fatte queste considerazioni, possiamo ora riscrivere il programma con una diversa condizione di salto, inserendo lo stato dell'indicatore di riporto al posto dell'indicatore di zero.

ORG $5E00		
	6502	Z80
5E00 START	ADC #$34	ADC A,$34
5E02	BCC START	JR NC,START
5E04	STA $5E20	LD ($5E20),A
5E07 EXIT	RTS	RET

In questa versione l'istruzione all'indirizzo $5E02 significa: "se l'indicatore di riporto vale 0, salta all'etichetta START". Quando il valore contenuto nell'accumulatore, al quale viene ripetutamente aggiunto $34, supera $FF, l'indicatore di riporto passa da 0 ad 1 e non si verifica più il salto all'etichetta START.

Contatori di ciclo

Può sembrare che i salti condizionati soltanto dallo stato dell'indicatore di zero o di quello di riporto offrano limitate possibilità, invece essi consentono un'ampia gamma di opportunità decisionali, come avremo presto occasione di apprezzare. Ciò che manca adesso al nostro repertorio sono i "contatori di ciclo". Supponiamo di voler contare quante volte un ciclo viene ripetuto prima che si verifichi la condizione di uscita oppure di volere uscire dal ciclo dopo un certo numero di iterazioni. Il primo obiettivo è facilmente raggiungibile usando un registro di indice della CPU come contatore ed un'istruzione di incremento per aggiornare il contatore:

6502		
0000	ORG	$5DFD
5DFD	LDX	#$00
5DFF START	INX	
5E00	ADC	#$34
5E02	BBC	START
5E04	STX	$5E20
5E07 EXIT	RTS	

Z80		
0000	ORG	$5DFA
5DFD	LD	IX,$0000
5DFE START	INC	IX
5E00	ADC	A,$34
5E02	JR	NC,START
5E04	LD	($5E20),IX
5E08 EXIT	RET	

L'introduzione di questa nuova struttura ha prodotto diversi cambiamenti nel programma. Primo fra tutti, un nuovo indirizzo ORG, poiché abbiamo aggiunto delle istruzioni all'inizio del programma. Le istruzioni producono effetti molto simili tanto sul 6502 quanto sullo Z80, pur avendo lunghezza diversa. Ovviamente, i vari indirizzi del programma adesso differiscono nelle due versioni.

Inoltre, sono state usate nuove istruzioni di caricamento (LDX, LD IX) e di immagazzinamento (STX, LD(),IX) per depositare il valore iniziale $00 nel registro di indice della CPU. Il registro X del 6502 è a singolo byte, mentre il registro IX dello Z80 possiede due byte. I registri "indice" svolgono speciali funzioni, ma sono essenzialmente RAM di CPU (come del resto lo stesso accumulatore) ed in questo caso vengono utilizzati come accumulatori addizionali con funzione di contatori di ciclo. Quando si verifica l'uscita dal ciclo iterativo, il contenuto del registro X del 6502 viene depositato all'indirizzo $5E20. Nello Z80, il byte basso del registro IX (due byte) verrà immagazzinato all'indirizzo $5E20 e quello alto all'indirizzo $5E21.

Infine, un'istruzione completamente nuova ha sostituito la ADC all'inizio del ciclo: INX e INC IX sono entrambe istruzioni d'incremento che sommano $01 al contenuto del registro indice. Questa procedura permette di aggiornare il contatore ogni volta che il ciclo viene eseguito.

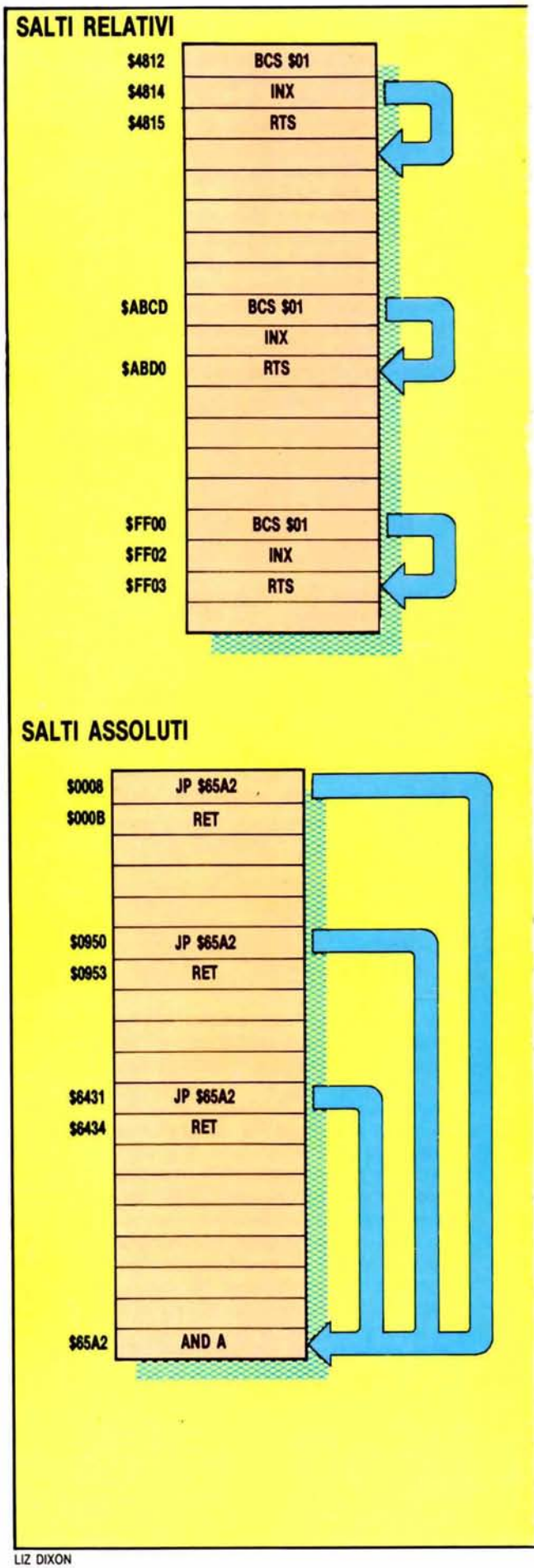
Il programma può essere così interpretato: "Assegna il valore zero al contatore di ciclo, inizia l'esecuzione del ciclo incrementando il contatore, somma $34 all'accumulatore e salta indietro all'inizio del ciclo se l'indicatore di riporto vale 0, altrimenti immagazzina il valore contenuto nel contatore di ciclo all'indirizzo $5E20". Un'ulteriore modifica accresce la sua utilità e funzionalità:

6502		
0000	ORG	$5DFA
5DFA	LDX	#$00
5DFC START	STA	$5E22,X
5DFF	INX	
5E02	ADC	#$34
5E02	BBC	START
5E04	STX	$5E20
5E07 EXIT	RTS	

Z80		
0000	ORG	$5DF7
5DF7	LD	IX,$5E00
5DFB START	LD	(IX+$22),A
5DFE	INC	IX
5E00	ADC	A,$34
5E02	JR	NC,START
5E04	LD	($5E20),IX
5E08 EXIT	RET	

Entrambe le versioni del programma (6502 e Z80) producono lo stesso risultato: creano, a partire dalla locazione $5E22, una tabella dei valori contenuti di volta in volta nell'accumulatore durante l'esecuzione del programma. Il valore del contatore di ciclo, pari al numero di byte contenuti nella tabella, viene depositato nella locazione $5E20, prima di eseguire l'istruzione di ritorno (RTS o RET).

Nella versione 6502 questa operazione si ottiene con l'istruzione STA $5E22,X che significa: "somma il contenuto del registro X all'indirizzo di base $5E22 ed immagazzina il contenuto dell'accumulatore all'indirizzo così ottenuto". L'istruzione STA, in questo caso, utilizza il modo di indirizzamento assoluto, diretto e indicizzato, vale a dire che il registro X viene usato come un indice per modificare l'indirizzo di base $5E22. Poiché il registro X inizialmente contiene il valore $00 e successivamente viene incrementato ad ogni iterazione del ciclo, il primo valore dell'accumulatore viene depositato all'indirizzo $5E22, il valore successivo a $5E23 e così via. Terminato il ciclo, STX



LIZ DIXON

La maggior parte delle istruzioni di salto, come BCS ("salta se l'indicatore di riporto vale 1") e JR NZ ("salta se l'accumulatore è diverso da zero"), viene eseguita in base alla condizione del registro di stato del processore. Il tipo di salto compiuto da queste istruzioni è detto "relativo", poiché viene sommato uno scarto (positivo o negativo) all'indirizzo della successiva istruzione da eseguire. L'alternativa è il salto "assoluto", nel quale l'indirizzo di destinazione viene specificato per intero.
Nell'esempio, l'istruzione BCS $01 produce sempre un salto relativo in avanti di un byte (condizionato però dallo stato dell'indicatore di riporto) indipendentemente dall'indirizzo di locazione al quale si trova il codice macchina. L'istruzione BCS $01 è sempre seguita dall'istruzione INX, anch'essa un'istruzione a byte singolo. Quando l'indicatore di riporto passa da 0 ad 1, BCS esegue un salto di un byte, all'istruzione che segue la INX

In questo esempio, l'istruzione JP $65A2 produce sempre un salto "incondizionato" (ossia, indipendente dalla condizione dei flag di stato del processore). Il suo compito è quello di dirottare l'esecuzione del programma all'indirizzo che appare come operando (in questo caso $65A2). Il funzionamento di questa istruzione non è legato alla sua posizione nel programma (come accade invece per i salti relativi) e l'esecuzione prosegue sempre dall'indirizzo specificato nel salto.
Entrambi i tipi di salto hanno dei pro e dei contro: un fattore decisivo, nella scelta tra un salto relativo ed uno assoluto, è la necessità che il programma risultante debba essere "rilocabile" o meno. Capita spesso, nel linguaggio Assembly, di scrivere una routine, assemblandola ad un particolare indirizzo ORG e trovarsi poi a riutilizzarla in un altro programma, ma partendo da una ORG diversa. Se nella routine tutti i salti sono relativi, allora il cambiamento dell'indirizzo di ORG (che influisce sugli indirizzi di tutte le istruzioni) non creerà alcun problema ed il programma continuerà a girare normalmente. Se però viene cambiata la ORG in una routine in cui compare un salto assoluto, l'indirizzo del salto può anche non aver più significato per il programma o, addirittura, essere completamente esterno ad esso. I salti relativi consentono quindi la "rilocazione", quelli assoluti no

provvede a memorizzare il valore finale del contatore di ciclo alla locazione $5E20.
Nella versione Z80 il registro IX viene impiegato contemporaneamente come puntatore alla tabella dei valori e come contatore di ciclo. L'istruzione LD IX,$5E00 carica l'indirizzo di base $5E00 nel registro IX e quindi il byte basso contiene $00. L'istruzione LD(IX+$22),A significa: "somma $22 all'indirizzo contenuto in IX ed immagazzina il contenuto dell'accumulatore all'indirizzo così ottenuto". Poiché IX inizialmente contiene il valore $5E00 (in seguito incrementato ad ogni iterazione del ciclo), il primo valore dell'accumulatore viene depositato all'indirizzo $5E22, il valore successivo a $5E23 e così via. Il byte basso di IX registra il numero di iterazioni del ciclo che, quando il ciclo termina, viene immagazzinato all'indirizzo $5E20. L'istruzione LD(IX+$22),A costituisce un esempio del modo di indirizzamento assoluto, diretto e indicizzato più complesso di quello della versione 6502, ma molto più potente.
Nella prossima lezione vedremo come utilizzare in pratica le strutture di programmazione in linguaggio Assembly, viste fin qui.

Esercizi

In questa lezione sono stati affrontati argomenti molto importanti e piuttosto complessi. Soltanto con una seria esercitazione pratica si riescono ad afferrare i principi dei vari modi d'indirizzamento.
Si usi il programma CHAMP per assemblare e memorizzare i vari frammenti di programma presentati in questa lezione. Per eseguire le brevi routine, occore usare il modo <DEBUG>, che consente di esaminare le locazioni di memoria modificate dal programma. È buona norma, prima di eseguire i programmi, inizializzare queste locazioni con una costante riconoscibile, ad esempio $FF. Questo accorgimento permette in seguito di individuare i cambiamenti nelle locazioni di memoria utilizzate dal programma. Per questa operazione possiamo usare i comandi Alter o Move nel modo <DEBUG>.
Ricordare, come sempre, che gli indirizzi nei programmi presentati sono soltanto degli esempi e che è necessario scegliere indirizzi adeguati alla macchina che si usa.

Caricamento e salvataggio del programma CHAMP
Per prudenza, conviene sempre farsi, su una cassetta diversa, una copia dei programmi di cui possediamo soltanto un originale. Lo stesso vale per il CHAMP. Fatto ciò, si potranno rimuovere dall'originale le fascette di protezione in scrittura. Nelle seguenti istruzioni, i comandi LOAD si riferiscono al nastro CHAMP ed i comandi SAVE al nastro di copia

BBC Modello B
1) LOAD "CHAMP"
2) SAVE "CHAMP", eseguire un RUN e tornare al BASIC
3) *SAVE "CHAMP" M/C" 1000,4600

Commodore 64
1) LOAD "CHAMP"
2) SAVE "CHAMP", eseguire un RUN e passare al modo <DEBUG>
3) Battere [W][RETURN], seguito da [S] per SAVE
4) Indirizzo iniziale 1000, indirizzo finale 4600; nome del file "CHAMP M/C"

Spectrum
1) LOAD "CHAMP"
2) Tornare al BASIC, poi: SAVE "CHAMP" LINE 1
3) SAVE "CHAMP M/C" CODE 27000,9231

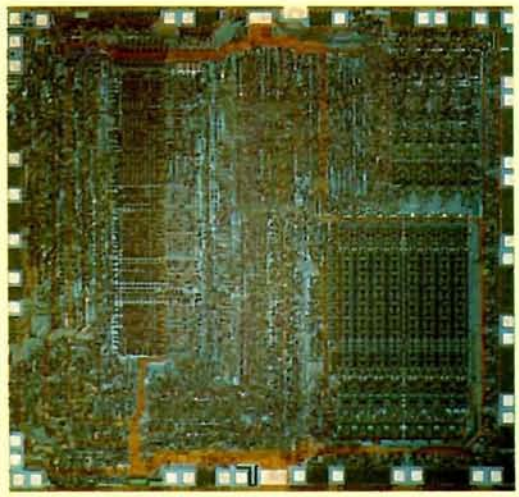
Le istruzioni di salto condizionato, come abbiamo visto, dipendono dal contenuto del registro di stato del processore. L'aggiunta dell'opzione di visualizzazione in binario al programma Monitor (vedere alle pagine 598 e 678) permette un esame del registro di stato prima e dopo l'esecuzione di un'istruzione, per osservare le conseguenze sugli indicatori (flag). Purtroppo, non esistono istruzioni di Assembly, né per il 6502 né per lo Z80, che consentano di memorizzare i valori contenuti nel registro di stato. Occorre quindi ricorrere ad un piccolo stratagemma:

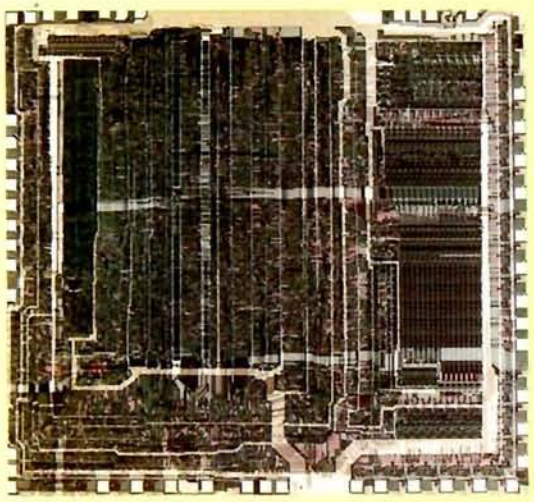
Z80	
3E00 F5	PUSH AF
3E01 F5	PUSH AF
3E02 E1	POP HL
3E03 22 lo hi	LD (STORE1),HL
3E06 F1	POP AF
6502	
E00 48	PHA
3E01 08	PHP
3E02 48	PHA
3E03 08	PHP
3E04 68	PLA
3E05 8D lo hi	STA STORE1
3E08 68	PLA
3E09 8D lo' hi'	STA (1+STORE1)
3E0D 68	PLP
3E0D 68	PLA

Questa serie di istruzioni provvede a depositare nel byte indicato con STORE1 (il cui indirizzo va scelto in modo opportuno per il computer usato), i valori contenuti nel registro di stato. Il contenuto dell'accumulatore viene memorizzato in (1+STORE1). Occorre collocare questa sequenza di istruzioni immediatamente prima e dopo l'istruzione della quale si vogliono conoscere gli effetti. Si ricordi che le istruzioni riguardanti STORE1 occupano 3 byte, se il codice macchina viene inserito manualmente. Dopo aver eseguito il programma, sarà possibile, col programma Monitor, visualizzare il contenuto di STORE1 e STORE1+1.
Il blocco di comandi appena visto potrebbe anche essere usato come una subroutine, evitando di riscriverlo più volte in un programma. In Assembly esiste un'istruzione equivalente al GOSUB del BASIC ma il suo uso, in questo caso, complicherebbe le cose. Infatti, essa utilizza lo "stack" e ciò interferirebbe con l'uso dello stack da parte delle istruzioni PLA, PUSH, PHP, ecc. (istruzioni che verranno spiegate più ampiamente in seguito). È significativa la differenza di lunghezza tra i codici del 6502 e dello Z80: essa è dovuta ai registri a due byte dello Z80 ed alle istruzioni ad essi associate.

La Zilog

Una forte concorrente nella produzione di microprocessori





Chip... al microscopio
Queste microfotografie evidenziano la complessità dei circuiti di un microprocessore. A sinistra è mostrata un'immagine ad alto ingrandimento del famosissimo Z80 della Zilog, a destra un chip più recente della serie Z8000. Notare che lo Z8000, essendo un chip a 16 bit, presenta una maggior densità di circuiti e molte più connessioni sul margine del chip

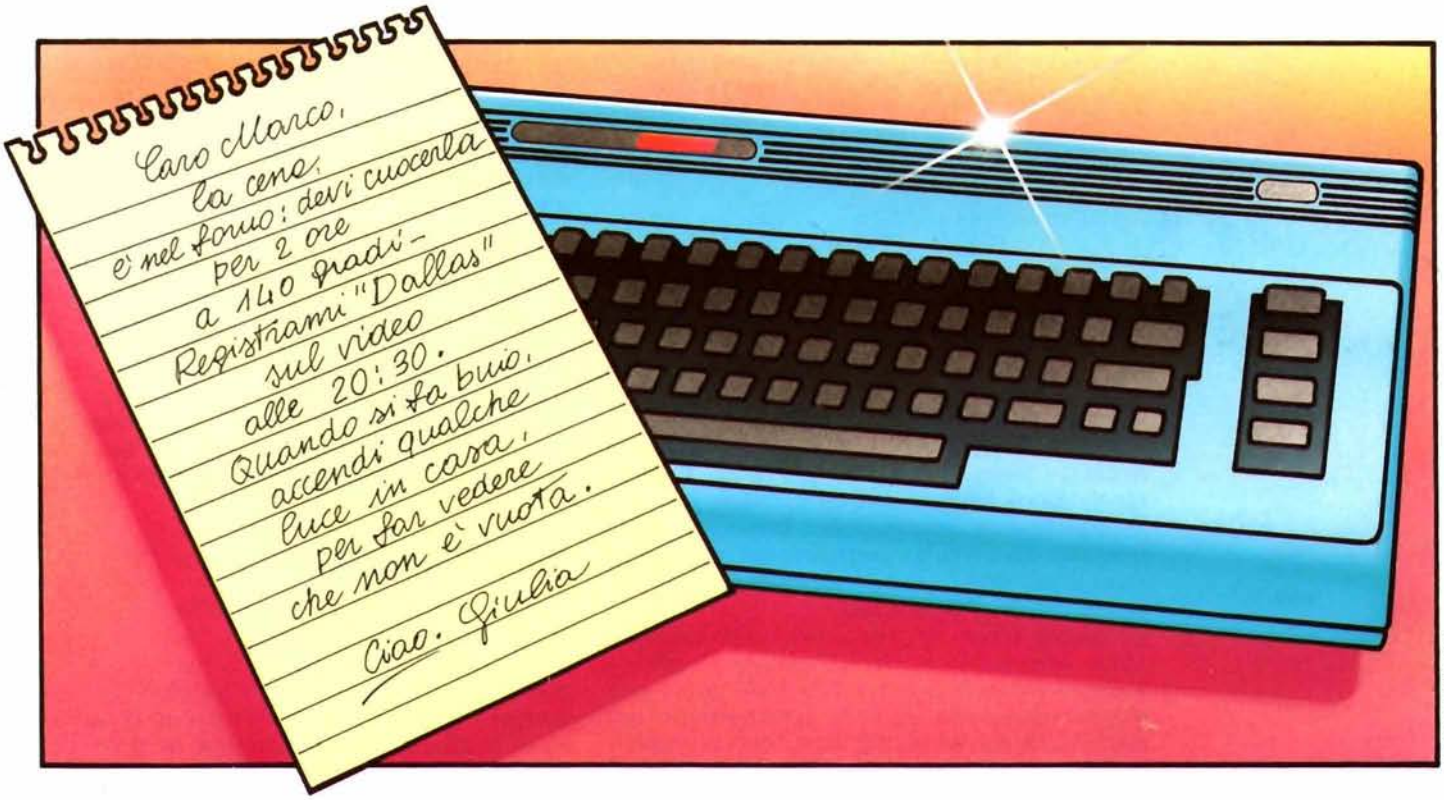
La storia della Zilog inizia nei primi anni '70, quando Frederico Faggin e Masatoshi Shima, che lavoravano per l'industria di microchip Intel, decisero di fondare una società in proprio. Entrambi si erano occupati della progettazione del microchip 8080A (considerato il primo "computer su un singolo chip") e sfruttarono l'esperienza acquisita nella loro precedente attività per realizzare una nuova versione di questo microprocessore.
L'8080 aveva già conquistato una vasta popolarità presso i progettisti di sistemi di elaborazione e gli hobbysti ed ovviamente Faggin e Shima decisero di rendere il nuovo chip compatibile con l'8080. L'approfondita conoscenza di questo componente permise loro di allargarne il "set" di istruzioni (la serie di comandi in codice macchina contenuti nel microprocessore) introducendo registri supplementari, op-code a due byte ed altre novità tecniche.
Il microprocessore sviluppato, chiamato Z80, presentava notevoli miglioramenti rispetto al chip 8080.
Questa innovazione nel campo hardware coincise con un radicale cambiamento nel settore software.
Nel 1972, Gary Kildall e John Torode svilupparono un nuovo programma chiamato Control Program/Monitor o CP/M che permetteva ad un microprocessore di controllare le unità a dischetti di recente introduzione. Poiché Kildall era un consulente della Intel, il programma era stato progettato per poter funzionare con l'8080 e con l'8085.
Il CP/M divenne rapidamente il più importante sistema di gestione disco per i microcomputer e la Zilog, grazie al suo nuovo e potente microprocessore compatibile con l'8080, si trovò nella condizione ideale per sfruttare nel modo migliore l'ondata di successo del software CP/M.
La fortuna della Zilog, negli anni seguenti, è stata alterna. Sebbene lo Z80 sia stato venduto in grandi quantità (ne sono stati prodotti circa un milione di esemplari), il tentativo di sviluppare il chip per il mercato dei 16 bit non ha condotto ai risultati sperati.
Il primo tentativo di processore a 16 bit è stato lo Z8000.
Questo nuovo componente, dotato di un'ampia gamma d'istruzioni e di un gran numero di registri, è molto potente, ma assai complesso da programmare e presenta numerosi altri inconvenienti. Innanzi tutto, per quanto sia compatibile con lo Z80000 a 32 bit (o Z80K) di imminente lancio, non lo è con lo Z80 e non poteva quindi utilizzare la grande e differenziata produzione software sviluppata negli anni precedenti per lo Z80.
Per questa ragione le ordinazioni dei fabbricanti intenzionati ad estendere le capacità elaborative dei loro apparecchi ai 16 bit, furono piuttosto limitate.
Dopo questo relativo insuccesso nel mercato dei microcomputer, la Zilog è tornata alla progettazione e si appresta a lanciare lo Z800, un processore a 16 bit compatibile con lo Z80. Le cose stanno andando meglio in altri campi, dopo l'annuncio della Commodore di dotare una nuova serie di apparecchi gestionali con il microprocessore Z8000.



Frank de Weeger, presidente della Zilog

Tutto sotto controllo

È possibile tenere a bada gli elettrodomestici di casa col proprio piccolo computer?



MIKE BROWNLOW

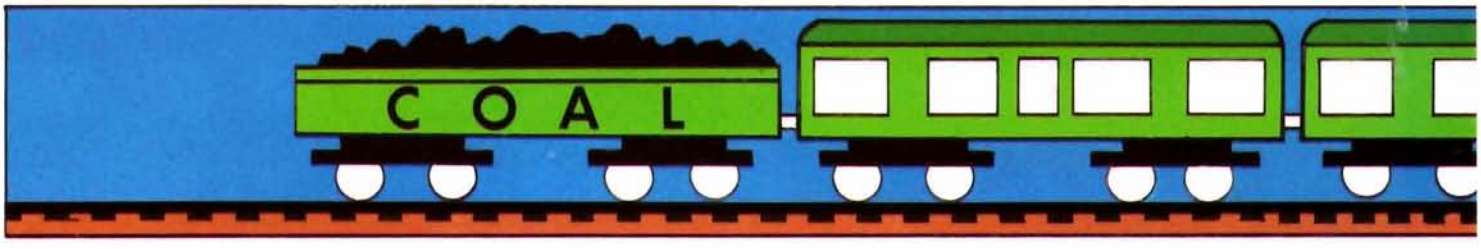
Programmare un microcomputer ad eseguire piccoli lavori domestici (come aprire le tende al mattino o innaffiare le piante quando si è in vacanza) può essere un passatempo piacevole e stimolante. L'hobby di progettare e costruire periferiche viene spesso considerato un semplice gioco ma può anche dimostrarsi, a lungo andare, un'attività remunerativa. Tra non molto, robot ed altri apparecchi controllati da computer diverranno parte integrante della nostra vita (come lo sono divenuti gli stessi computer nel giro di pochi anni), quindi capacità acquisite oggi potrebbero in futuro dimostrarsi estremamente utili. Non a caso alcuni "colossi" dell'informatica, come Apple e Atari, sono nati in comuni garage di appassionati d'elettronica.

Un sistema controllato da computer è costituito da una o più periferiche pilotate, ovviamente, dal computer. Occorre inoltre un meccanismo di collegamento per l'invio dei segnali di controllo alle periferiche ed il software che permetta al computer di gestire l'intero sistema. Ma non basta: il computer deve anche poter verificare gli effetti causati dai segnali trasmessi, per poter effettuare rettifiche di precisione. Questo processo prende il nome di "feedback" e senza di esso il computer non è che un autista cieco.

Tutti i sistemi guidati da computer vengono controllati mediante segnali elettrici. Sfortunatamente, le deboli tensioni elettriche usate all'interno del computer non sono adatte ad essere impiegate direttamente: anche una piccola lampadina a pile assorbe una potenza molto superiore di quella erogabile da un qualsiasi componente del computer. Il procedimento per convertire queste deboli tensioni in segnali più potenti (a volte, quanto la stessa tensione di rete) avviene in più stadi.

Controllo a distanza

Collegando varie apparecchiature ad un computer, si possono eseguire in modo automatico innumerevoli operazioni. Il computer può intervenire in tempi prestabiliti oppure reagire ad eventi come la diminuizione di temperatura o l'accensione di un allarme



Collegamenti di controllo

Per molti home computer, soprattutto per il BBC e lo ZX Spectrum, sono disponibili interfacce complete per funzioni di controllo. In questi apparecchi vengono di solito impiegati dei relais. Il computer può accendere e spegnere una specifica periferica e può sapere quando un sensore è attivo o no



IAN McKINNELL

Il primo di questi concerne la memoria: per inviare segnali all'esterno il computer deve dedicare a questo compito una specifica sezione di memoria (chiamata "Porta utente"). I messaggi provenienti dal microprocessore vi vengono depositati come se si trattasse di normale memoria RAM, ma ad essi viene riservato un trattamento particolare, per consentire che le informazioni immagazzinate possano essere lette elettronicamente dall'esterno del micro, senza interferire in alcun modo con la sua attività.

In alcuni micro la porta utente è venduta come accessorio a parte, mentre in altri è incorporata nell'apparecchio. Tale porta può essere usata per pilotare direttamente dei diodi LED; ma, se gli obiettivi sono più ambiziosi, sono necessari appositi circuiti di potenza oltre ad un'alimentazione adatta ai relè impiegati. I "relè" sono interruttori elettromagnetici, in grado di accendere e spegnere correnti relativamente grandi, pur essendo controllati da piccole tensioni. Sono quindi lo strumento ideale per questo tipo di applicazioni. Uno dei primi obiettivi per gli appassionati di sistemi di controllo computerizzato dovrebbe quindi essere la realizzazione di un sistema a relè, col quale controllare le più svariate periferiche: motori elettrici, pompe ad acqua, luci, campanelli, segnali acustici, trenini elettrici o addirittura modellini radiocomandati.

Per la maggior parte delle applicazioni sperimentali, bastano piccoli relè, adatti al controllo di apparecchi a pile. Tuttavia, in casi particolari, occorrono relè di maggiori dimensioni, capaci di lavorare con la corrente di rete e, poiché quest'ultima è molto pericolosa (sia per lo sperimentatore che per i circuiti del computer), è consigliabile adoperare prodotti commerciali ben collaudati. Chi lavora con bassi voltaggi può acquistare o fabbricarsi da sé semplici circuiti dotati di commutatori a relé, da collegare al computer.

Con simili dispositivi, il computer può controllare l'accensione o lo spegnimento di molti altri apparecchi domestici oppure, adoperandolo come un timer, accendere la televisione ad una particolare ora o far lampeggiare le luci per spaventare eventuali ladri.

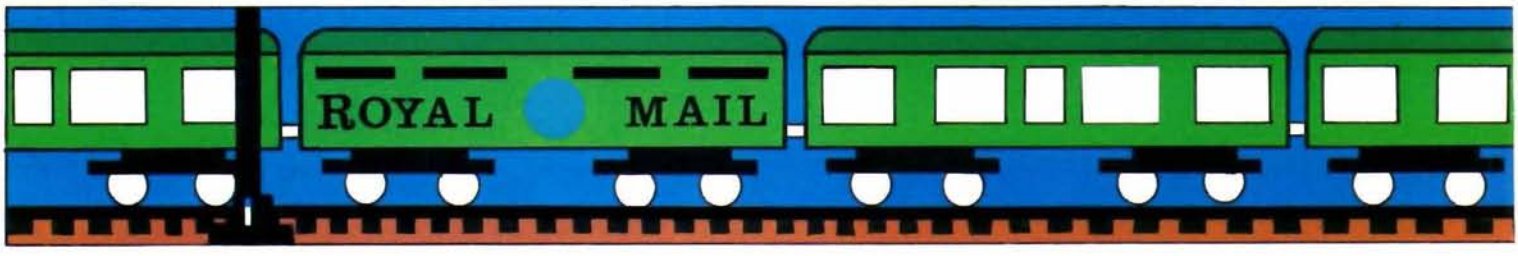
Attualmente, il collegamento tra un computer e le unità poste sotto il suo controllo, è possibile soltanto realizzando un apposito impianto di cavi elettrici; ma già alcune industrie stanno sperimentando nuove apparecchiature che permetteranno di eliminare questo inconveniente, sfruttando convenientemente l'impianto elettrico già esistente nell'abitazione, per inviare segnali alle varie periferiche. Un tipico sistema sarà quindi composto da un computer e da unità secondarie sparse per la casa ed alimentate da normali spine di corrente. Il computer potrà controllare ciascuna periferica inviando messaggi "individuali". Qualsiasi elettrodomestico può essere trasformato in unità secondaria e quindi controllato dal computer.

Segnali di feedback

Abbiamo già detto che, per ottenere un controllo veramente efficace, il sistema deve possedere un meccanismo di "feedback". Nel controllo dell'impianto d'illuminazione, ad esempio, si potrebbe introdurre un dispositivo di feedback, che permetta al computer di sapere quando fuori fa buio e di accendere le luci di casa in risposta a questa informazione; oppure un sistema che informi il computer quando una persona entra in una stanza, in modo da accendere automaticamente la luce. Se il computer controlla l'impianto di riscaldamento, è necessario un feedback per conoscere la temperatura di una stanza e per mantenerla costante.

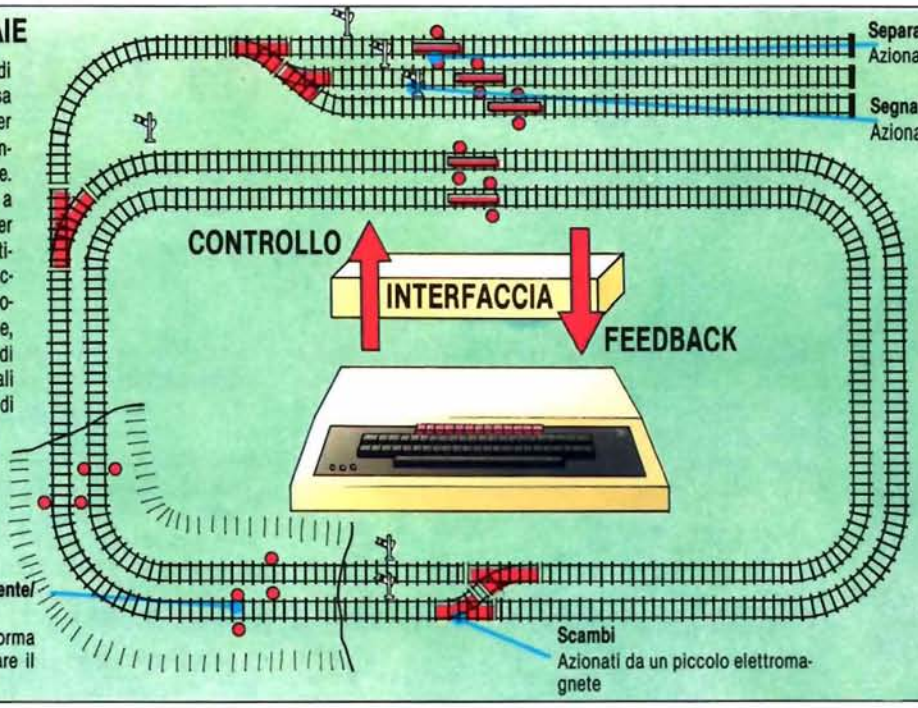
I segnali di feedback sono di due tipi. Alcuni possono indicare solo una condizione acceso/spento, senza alcun stato intermedio. Un segnale di questo tipo potrebbe essere prodotto da un interruttore che informa se una finestra è aperta o chiusa o se qualcuno ha suonato il campanello di casa. Le porte utente sono in grado di interpretare questo tipo di "segnale acceso/spento" direttamente.

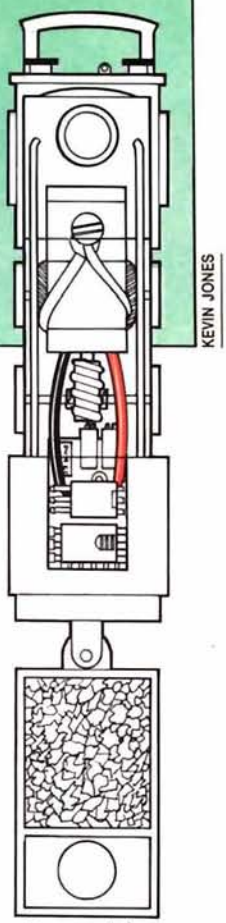
Un secondo tipo di feedback, più utile ma più complesso, produce un "segnale analogico". Un simile segnale composto da tutta una serie di valori è utile per misurare il calore o



COMPUTER... SU ROTAIE

La realizzazione di un sistema di controllo computerizzato, sia essa per un modellino di ferrovia o per un'intera casa, richiede la conoscenza delle stesse tecniche di base. Occorre adottare una procedura a ciclo chiuso, nella quale il computer invia segnali di controllo ad una particolare periferica attraverso l'interfaccia. Questi segnali controllano servomotori, luci e così via. Le periferiche, a loro volta, inviano informazioni di feedback, generate da sensori quali cellule foto-sensibili o rilevatori di passaggio. Tali segnali di ritorno permettono al computer di tenere sotto costante controllo l'intero sistema





KEVIN JONES

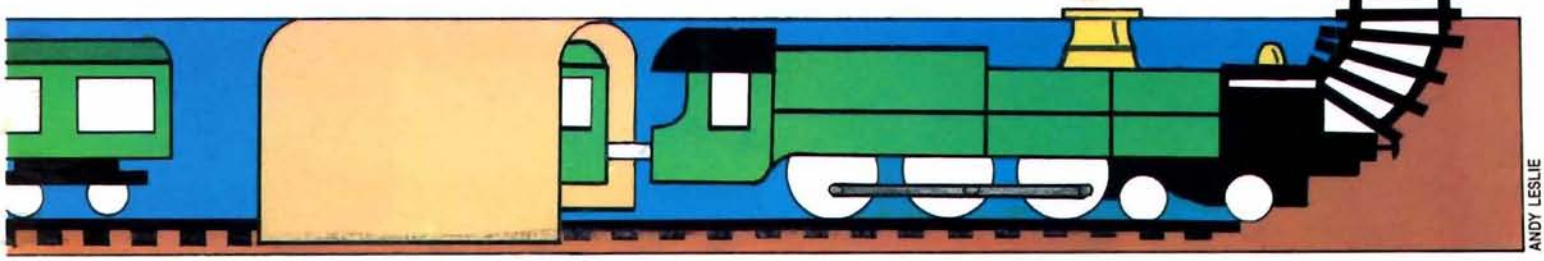
il peso di un oggetto, il suo spostamento o la sua rotazione, oppure anche il voltaggio di una batteria. Uno strumento capace di interpretare questo tipo di segnali viene chiamato "convertitore analogico/digitale" (in sigla A/D) perché riceve dall'oggetto sottoposto a controllo una serie variabile di valori (segnali analogici) e li converte in segnali digitali, comprensibili da parte del computer.

La presenza di feedback offre al sistema di controllo un vasto campo di applicazioni. Se un motore fa girare una ruota, il computer può calcolare la rotazione effettuata in un dato tempo. Tuttavia, se poniamo un peso su di essa o se le batterie che azionano il motore sono quasi scariche, la risposta del computer potrebbe essere falsata. Un sensore ottico potrebbe segnalare al computer quando la ruota ha compiuto una rotazione completa e quest'ultimo potrebbe immagazzinare le informazioni ricevute, per ottenere una misurazione più veritiera.

Alcuni tipi di motori elettrici, costruiti appositamente per usi di controllo, sono dotati di un meccanismo di feedback incorporato. Quando il computer invia loro il comando di eseguire un particolare movimento, il motore entra in azione e si ferma solo dopo aver eseguito l'istruzione ricevuta. I motori di questo tipo si dividono in due categorie: i "motori passo-passo" ed i "servomotori". I primi possono ruotare continuamente, come un normale motore, ma possono anche essere fermati in una qualunque posizione (motori di questo tipo sono usati nelle unità a dischetti). Tuttavia, la loro potenza è limitata e possono manovrare soltanto piccoli pesi. I servomotori, invece, sono molto più potenti, ma possono ruotare soltanto di un piccolo angolo, solitamente circa 90 gradi. Spesso svolgono un movimento ripetitivo simile a quello di uno stantuffo. Per poter lavorare con i computer, entrambi i tipi di motore richiedono particolari unità di controllo, che vengono prodotte solo per i computer più diffusi. I servomotori sono usati in molti bracci robotizzati; di questi in commercio ne esistono alcuni, piccoli, interfacciabili con gli home computer, anche se il loro prezzo è piuttosto alto.

L'ultima categoria di apparecchi controllati da computer che prendiamo in esame prevede il controllo mediante un voltaggio variabile; ad esempio, un piccolo motore elettrico, in grado di ruotare a differenti velocità, a seconda del voltaggio applicato. In tal caso si impiega un convertitore "digitale/analogico", che opera all'opposto di un convertitore A/D: trasforma cioè i segnali digitali usati dal computer in voltaggi variabili.

Per la realizzazione di sistemi controllati da computer è necessario disporre di molto tempo e saper combinare buone idee e conoscenze tecniche. Spesso i risultati sono inferiori a quelli dei prodotti commerciali, ma la soddisfazione ed il divertimento sono assicurati!



ANDY LESLIE

Disegniamo al computer

Ciò che si ottiene con poche e semplici linee di BASIC è strabiliante!



Tracciare il grafico di un'espressione matematica è abbastanza facile. Basta prendere un gruppo di valori di una delle due variabili dell'espressione e calcolare i corrispondenti valori dell'altra. Dalla semplice espressione $Y=X^2$ calcoliamo la seguente tabella:

X	−5	−4	−3	−2	−1	0	1	2	3	4	5
Y	25	16	9	4	1	0	1	4	9	16	25

Adesso, tracciando i punti su un foglio di carta millimetrata e congiungendoli con una linea, otteniamo la ben nota curva che rappresenta graficamente l'espressione $Y=X^2$. Questa curva si può considerare come la traccia lasciata da un punto che si sposti secondo l'espressione $Y=X^2$. La traccia viene anche chiamata "luogo" e, manipolando e combinando tra loro luoghi differenti, si possono facilmente realizzare strutture grafiche sbalorditive.

Prendiamo ora in esame una curva nota a tutti, il cerchio. La formula matematica usata per rappresentare un cerchio è assai più semplice di quella vista in precedenza. Sia la X che la Y vengono definite da una terza variabile (o parametro), ed assegnando ad essa tutta una data serie di valori, si possono calcolare le varie coppie di X e Y. Matematicamente, la circonferenza si esprime con la seguente equazione:

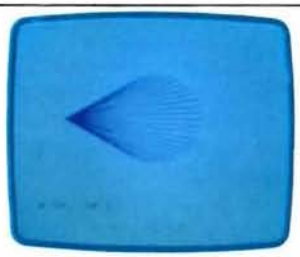
$$X = R \text{ SEN } (I) \quad Y = R \text{ COS } (I)$$

Se calcoliamo una serie di valori per X ed Y, facendo crescere l'angolo I da 0 a 360 gradi (cioè per un'intera circonferenza), otteniamo il luogo di un cerchio. Il seguente programma, per lo Spectrum, crea un cerchio. (Consultare il riquadro "Basic & Basic" per altri dialetti BASIC).

```
10 REM Disegno di un cerchio
20 LET xm = 256: LET ym = 176: LET xc = int (xm/2): LET yc = INT
   (ym/2)
30 LET r = 50
40 LET s = PI/20
50 FOR i = 0 TO 2*PI STEP s
60 INK 2: PLOT xc + r*SIN (i), yc + r*COS (i)
70 NEXT i
```

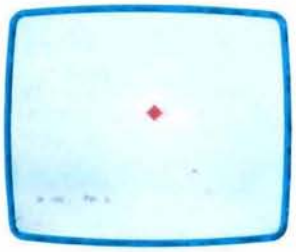
Si noti che, cambiando la grandezza dello STEP tra i punti tracciati, è possibile variare la definizione del cerchio e la velocità con cui viene tracciato. La maggior parte dei dialetti BASIC si dimostra piuttosto lenta nel tracciare cerchi composti da singoli punti. È spesso preferibile, quindi, ricorrere ad un altro metodo, che consiste nel congiungere un certo

La prima immagine viene realizzata tracciando una serie di linee che congiungono un punto fisso a ciascun punto del cerchio. I seguenti due programmi posizionano il punto fisso rispettivamente al centro ed a sinistra del cerchio



Il seguente programma usa un punto in movimento anziché fisso e traccia contemporaneamente due luoghi, congiungendo con linee i punti corrispondenti delle due circonferenze. La figura ottenuta è composta da due cerchi concentrici collegati da numerose linee

```
10 REM fiore
20 LET xm=256: LET ym=176:
   LET xc=INT (xm/2): LET yc=INT (ym/2)
30 LET r=50
40 LET s=PI/20
50 FOR i=0 TO 2*PI STEP s
60 INK 2: PLOT xc+r*SIN (i), yc+r*COS (i):
   DRAW xc-(xc+r*SIN (i)),yc-(yc+r*COS (i))
70 NEXT i
```



```
10 REM Cerchio & Punto fisso
20 LET xm=256: LET ym=176:
   LET xc=INT (xm/2): LET yc=INT (ym/2)
30 LET r=50
40 LET s=PI/20
50 For i=0 TO 2*PI STEP s
60 INK 2: PLOT xc+r*SIN (i), yc+r*COS (i):
   DRAW xc-100-(xc+r*SIN (i)), yc-(yc+r*COS (i))
70 NEXT i
```

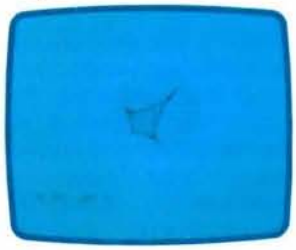
```
10 REM Cerchi concentrici
20 LET xm=256: LET ym=176:
   LET xc=INT (xm/2): LET yc=INT (ym/2)
30 LET r=50
40 LET s=PI/20
50 FOR i=0 TO 2*PI STEP s
60 LET x=xc+(r+30)*SIN (i):
   LET y=yc+(r+30)*COS (i)
70 LET p=xc+r*SIN (i): LET q=yc+r*COS (i)
80 INK 2: PLOT x,y: DRAW p-x,q-y
90 NEXT i
```



Il programma qui mostrato permette di disegnare centinaia di figure, modificando uno o entrambi i luoghi. Una variazione può essere quella di scambiare SEN e COS in una delle formule, oppure si potranno elevare a potenze SEN o COS (o moltiplicarli tra loro) per ottenere effetti diversi

```
10 Cerchi deformati
20 LET xm=256: LET ym=176:
   LET xc=INT (xm/2): LET yc=INT (ym/2)
30 LET r=50
40 LET s=PI/20
50 FOR i=0 TO 2*PI STEP s
60 LET x=xc+(r+30)*COS (i):
   LET y=yc+(r+30)*SIN (i)
70 LET p=xc+r*SIN (i): LET q=yc+r*COS (i)
80 INK 2: PLOT x,y: DRAW p-x,q-y
90 NEXT i
```





Le variazioni possibili sono infinite. Da un comune testo di matematica si possono estrarre formule per realizzare vari tipi di curve, ma è molto più divertente inventarne di nuove, programmandole da soli. Con poche e semplici modifiche al programma, può essere lo stesso computer a generare nuove sagome, in modo casuale. Il programma finale, pur non arrivando a riprodurre tutte le possibili configurazioni, genera effetti molto interessanti

```
10 REM Cerchio & SIN
20 LET xm=256: LET ym=176:
   LET xc=INT (xm/2): LET yc=INT (ym/2)
30 LET r=50
40 LET s=PI/20
50 FOR i=0 TO 2*PI STEP s
60 LET x=xc+SIN (i)*80:
   LET y=yc+(r+30)*SIN (i)
70 LET p=xc+r*SIN (i): LET q=yc+r*COS (i)
80 INK 2: PLOT x,y: DRAW p-x,q-y
90 NEXT i
```





```
10 REM Cerchio & SIN*COS
20 LET xm=256: LET ym=176:
   LET xc=INT (xm/2): LET yc=INT (ym/2)
30 LET r=50
40 LET s=PI/20
50 FOR i=0 TO 2*PI STEP s
60 LET x=xc+r*SIN (i)*COS (i):
   LET y=yc+r*SIN (i): LET q=yc+r*COS (i)
70 LET p=xc+r*SIN (i): LET q=yc+r*COS (i)
80 INK 2: PLOT x,y: DRAW p-x,q-y
90 NEXT i
```

```
10 REM Figure generate casualmente
15 RANDOMIZE
20 LET xm=256: LET ym=176:
   LET xc=INT (xm/2): LET yc=INT (ym/2)
30 LET r=60: LET s=PI/20
35 CLS
40 LET c=INT (RND*4)+1: LET d=INT (RND*4)+1:
   LET e=INT (RND*4)+1: LET f=INT (RND*4)+1
50 FOR i=0 TO 2*PI STEP s
60 LET x=xc+r*SIN (i/c)*COS (i*d):
   LET y=yc+r*SIN (i/e)*COS (i*f)
70 LET p=xc+r*SIN (i): LET q=yc+r*COS (i)
80 PLOT x,y: DRAW p-x,q-y
90 NEXT i
100 IF INKEY$="" THEN GO TO 100
110 GO TO 35
```



numero di punti della circonferenza con tratti lineari. Se i punti usati sono relativamente numerosi, si ottiene un accettabile compromesso tra velocità d'esecuzione e linearità del disegno. Questa procedura può essere usata anche per tracciare archi ed ellissi. Variando il valore di I si otterranno archi con diversa curvatura. Per le ellissi si dovranno assegnare due valori diversi da R, nelle due formule per X e Y. Il cerchio è certamente la figura geometrica più adatta per creare attraenti figurazioni grafiche.

Basic & Basic

I programmi presentati possono girare sullo Spectrum con 16 o con 48 Kbyte, ma possono essere facilmente modificati per altre macchine, purché dotate di grafica ad alta risoluzione (preferibilmente almeno 256 x 176 pixel), un BASIC a virgola fluttuante con le funzioni SIN e COS ed un'istruzione di comando per tracciare singoli punti e linee rette.

Le prime correzioni da fare riguardano XM e YM, i valori massimi di X e Y che il nostro computer è in grado di tracciare. A seconda delle funzioni usate, anche altre costanti del programma come R e S dovranno essere cambiate. Si deve quindi stabilire l'appropriato modo grafico e selezionare un colore per la figura. Infine, occorrerà un comando che permetta di congiungere le coordinate assegnate a x, y, p e q con delle linee rette. Sullo Spectrum si usa il comando PLOT, seguito da DRAW. L'uso di quest'ultimo comando, sullo Spectrum, è piuttosto complicato perché si riferisce sempre all'ultimo punto tracciato, mentre in questo caso dobbiamo tracciare linee in una particolare posizione assoluta. Molti micro possiedono una funzione per tracciare linee in posizione assoluta, semplificando molto la realizzazione del programma.

MICRO BBC. Tutti i modi grafici del BBC impiegano una griglia grafica 1280 x 1024 ed il MODO 0 permette di ottenere ottimi risultati. Il comando GCOL seleziona il colore del disegno mentre MOVE e DRAW tracciano le linee.

DRAGON 32/64. Il PMODE 4 del Dragon utilizza una griglia di 256 x 192 punti, adatta a questi programmi. Si usi SCREEN 1,0 o SCREEN 1,1 per selezionare i colori di sfondo verde o giallo. Per tracciare le linee può essere usato il comando LINE (LINE (X,Y)—(P,Q),PSET).

COMMODORE 64/VIC-20. Queste macchine possiedono una buona grafica ad alta risoluzione, ma non sono dotate di comandi adeguati. Quindi, per far girare questi programmi, sarà necessario crearsi dei propri comandi grafici, oppure usare una cartuccia di estensione al BASIC come il Simon's BASIC.

LINX. Il Linx è particolarmente adatto a questo tipo di programmazione, poiché ha una grafica ad otto colori di 256 x 248 punti che, come sullo Spectrum, viene abilitata automaticamente. Useremo INK per selezionare il colore del disegno ed i comandi MOVE e DRAW per tracciare le linee.

ORIC 1/ATMOS. L'istruzione HIRES abilita il video grafico 204 x 200 dell'Oric. Le linee possono essere tracciate usando CURSET, per posizionarsi sul loro punto di origine (x,y), e DRAW per tracciare la linea. Il DRAW dell'Oric è un comando relativo, quindi il suo formato è: DRAW p-x,q-y.

QUALCHE ALTRO SPUNTO

1) Tornando al semplice programma a ciclo chiuso che disegna un cerchio, abbiamo visto come modificarlo per tracciare archi ed ellissi. Si trovi il modo per disegnare spirali.
2) Proviamo ad usare altri comandi come SQR e TAN per creare dei "luoghi". Queste funzioni vanno usate con grande prudenza, perché sono inclini a generare numeri inadatti allo scopo. Tuttavia, dovremmo essere in grado di ottenere buoni risultati.
3) Si producano versioni "animate" dei programmi. Usando delle matrici, per registrare le ultime 5 linee tracciate, dovremmo essere capaci di realizzare un gruppo di cinque linee che si muovono all'interno di due "luoghi" concentrici.
4) Perché non creare figure basate su tre "luoghi"? Definiamone due semplici (ad esempio un cerchio ed una linea retta), in modo da ottenere un'immagine "pulita".

L'accesso ai file

Come impiegare i file sequenziali nei programmi e superarne le limitazioni

Un file sequenziale è un compatto blocco di dati registrati su disco o nastro e, come tale, impone alcune limitazioni alle procedure di accesso e di aggiornamento. Per rintracciare una particolare informazione, è necessario "leggere" tutti i dati che la precedono, mentre per aggiornare il file occorre farne una nuova copia, fino al dato da cambiare, inserire le modifiche nel nuovo file ed infine, portare a termine la copia del file originale.

È importante sottolineare che le informazioni all'interno del file devono essere organizzate in modo logico: è il programmatore che, in base al tipo di applicazione, sceglie il modo di procedere più conveniente. Se un file contiene un testo, sarà probabilmente composto da una sequenza di codici ASCII, seguiti da un marcatore di "fine file". Tuttavia, se il file deve contenere un database (ad esempio, un catalogo di libri), allora i dati dovranno essere organizzati secondo un preciso criterio. Il sistema più usato è quello di suddividere il file in "record" e "campi". Ciascun libro in catalogo avrà il proprio record, composto da un certo numero di campi (uno per il titolo del libro, uno per l'autore, e via dicendo). All'interno di un file sequenziale, queste suddivisioni vanno contrassegnate con caratteri speciali, inseriti tra un campo e l'altro.

Di solito, è il carattere di "ritorno carrello" (codice ASCII n. 13) che funge da marcatore tra i campi ed i record. Poiché tutti i record avranno lo stesso numero di campi, il programma individuerà facilmente dove termina un record e ne comincia un altro.

Una volta creato, il file sequenziale può essere sottoposto alle seguenti operazioni di base: ricerca, modifica o cancellazione di record ed immissione di nuovi record. I diagrammi della pagina seguente mostrano come queste procedure operano sul file. Poiché quest'ultimo può essere letto soltanto in modo sequenziale e non è possibile cambiare liberamente i dati in esso contenuti, le procedure leggono il file originale, creandone via via una nuova copia. Nel punto e nel momento opportuno, le modifiche vengono quindi inserite nel nuovo file, che rimpiazzerà quello originale al termine dell'operazione. Il vecchio file viene eliminato o conservato come copia di riserva.

Queste semplici tecniche sono alla base di tutte le routine di gestione dei file sequenziali. Ovviamente, il sistema operativo deve essere in grado di tener aperti contemporaneamente due diversi file, leggendo i dati da uno e scrivendoli nell'altro. Questo è possibile nella maggior parte dei sistemi a disco, mentre per quello che riguarda i micro basati sull'impiego di cassette, ciò avviene soltanto in quelli che possono collegarsi simultaneamente a due registratori. Le interfacce per il Grundy Newbrain e per il PET Commodore, ad esempio, prevedono questa possibilità. Negli apparecchi dotati di una sola unità a cassetta, i file devono avere dimensioni tali da poter essere trasferiti interamente in memoria e qui elaborati.

Questo sistema di gestire i file sequenziali presenta anche un interessante effetto secondario: dopo un qualsiasi intervento sul file (aggiunta, cancellazione o correzione) si ottengono due copie: una "vecchia", che contiene il file non aggiornato e una "nuova" o "corrente", con i cambiamenti introdotti. Spesso, nei programmi commerciali, entrambe le copie vengono conservate, perché se il file corrente viene accidentalmente danneggiato, si ha modo di utilizzare la copia non aggiornata (mancante solo dell'ultima serie di modifiche), senza perdere tutto quanto il lavoro. Molti sistemi gestionali conservano fino a tre "generazioni" di ciascun file: il file corrente detto "figlio" ed i due precedenti chiamati rispettivamente "padre" e "nonno".

Queste tecniche si possono applicare a file di dimensioni tali da non entrare interamente nella memoria del computer, tanto più che, in genere, è soltanto una piccola porzione dell'archivio quella interessata ad una singola modifica.

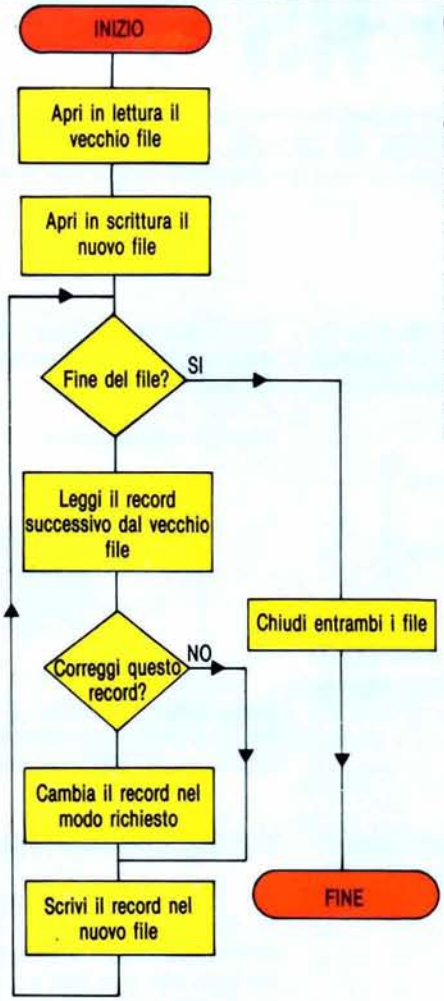
Al contrario, i file di minore estensione vengono trasferiti integralmente in memoria ed elaborati ad alta velocità, prima che il nuovo file, modificato, venga registrato su disco o nastro.

Questo modo di procedere presenta un grosso rischio: poiché le modifiche apportate divengono permanenti solo dopo che l'informazione è stata scritta su cassetta o su disco, un'interruzione accidentale del programma o nel funzionamento del computer, provoca la perdita dei dati. Pertanto, nei nostri programmi, è indispensabile assicurarsi che vengano create frequenti copie d'archivio dei file e che il file corrente venga memorizzato prima che il programma termini.

Con l'esperienza ci si accorgerà che queste precauzioni, sebbene un po' complicate, sono in realtà assai utili. Su molti sistemi di piccole dimensioni è possibile lavorare solo con file a struttura sequenziale. Quando esamineremo i file ad accesso casuale, incontreremo tecniche che integrano i file seriali e che forniscono semplici e rapide procedure di accesso e di aggiornamento.

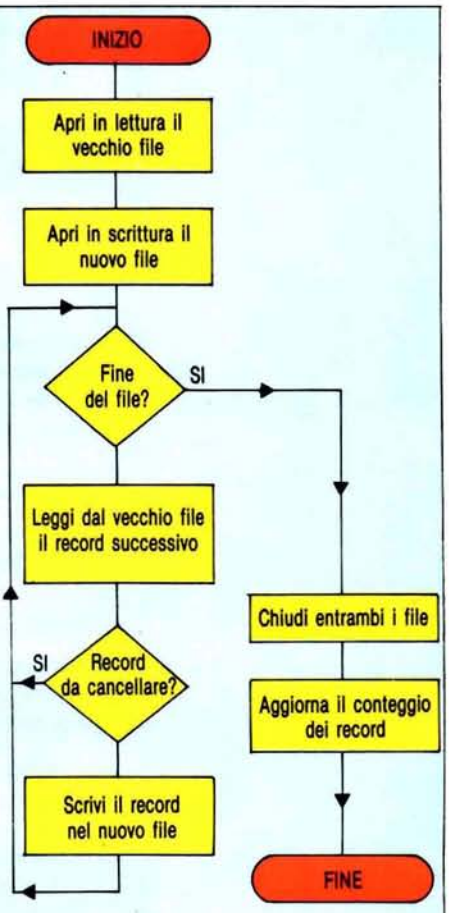
CORREZIONE DI UN RECORD

Ecco come avviene la correzione (ossia una modifica al contenuto) di un record. Innanzi tutto si devono copiare in un nuovo file tutti i record che lo precedono. Raggiunto il record voluto, lo si trasferisce in RAM dove viene modificato dal programma. Il record così aggiornato viene quindi scritto nel nuovo file, sul quale vengono poi scritti i successivi record del file "padre". Con un solo passaggio attraverso il file è possibile aggiornare un qualsiasi numero di record



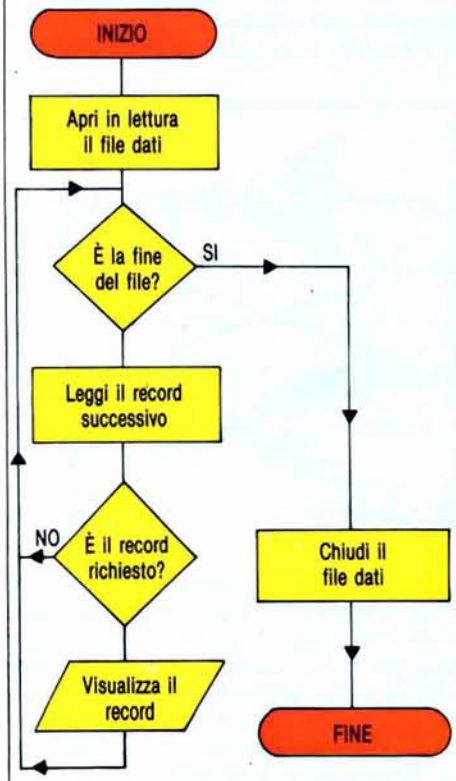
CANCELLAZIONE DEI RECORD

I record possono essere cancellati da un file leggendo e copiando fino ad incontrare il record che si vuol cancellare. Quest'ultimo viene letto (ma non copiato) nel nuovo file. Infine, anche il resto del file viene letto e copiato nel nuovo file. Con un solo passaggio è possibile cancellare più record. Dopo la cancellazione è necessario aggiornare immediatamente il conteggio dei record



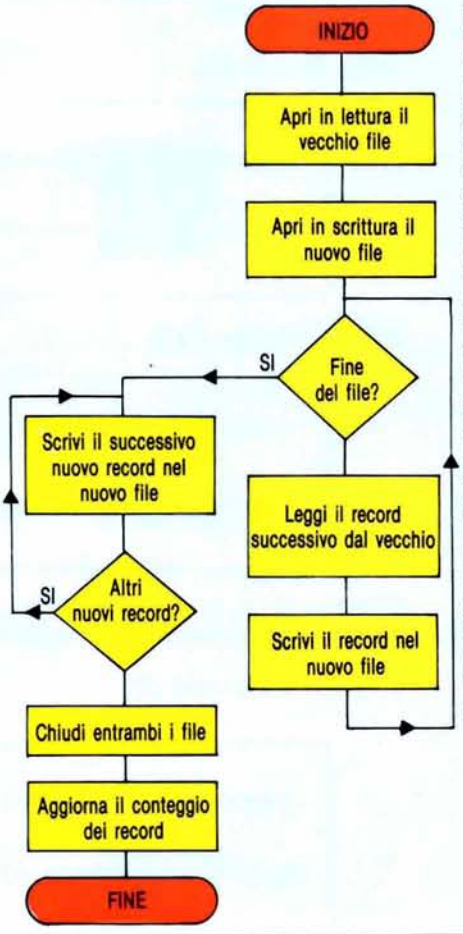
RICERCA DEI RECORD

I file sequenziali mal si adattano alle procedure di estrazione dei record dal file. La ricerca di un record implica, infatti, la lettura dell'intero file e ciò richiede molto tempo. Durante la ricerca di un particolare record, il programma è costretto ad esaminare ciascun record, fino ad incontrare quello richiesto. Se si vogliono trovare più record, il programma li leggerà in sequenza, ma rispettando la loro disposizione all'interno del file. Spesso, per questo motivo, i file sequenziali vengono ordinati secondo un particolare criterio (ad esempio, in ordine alfabetico), prima di essere memorizzati



IMMISSIONE DI NUOVI RECORD

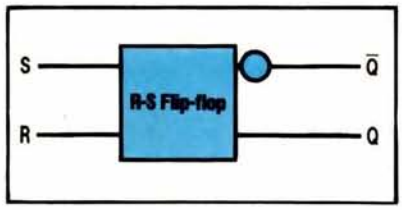
I record possono essere aggiunti ad un file in due modi. Alcuni BASIC possiedono un comando APPEND che permette di aggiungere il nuovo record direttamente alla fine del file. Nei sistemi che non dispongono di questo comando è necessario leggere l'intero file e creare una nuova copia. Quindi scrivere i nuovi record alla fine del file ed infine chiudere entrambi i file. In ambedue i casi si dovrà aggiornare il conteggio dei record. Se tale valore viene memorizzato all'interno dello stesso file, la routine di aggiornamento dovrebbe verificare che esso venga immediatamente scritto nel file. Questa verifica impedirà di perdere quanto appena inserito



I flip-flop RS

Esaminiamo il funzionamento di questi particolari circuiti elettronici

Esistono molti tipi di flip-flop, ma tutti funzionano in base ai medesimi principi. Il flip-flop R-S è dotato di due ingressi e di due uscite.

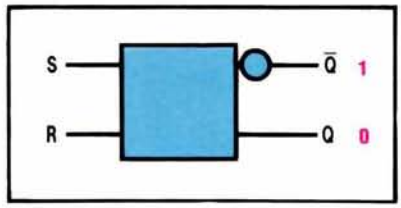


Il circuito è realizzato in modo che le uscite Q e $\overline{Q}$, siano sempre l'una di valore opposto all'altra. Cioè:

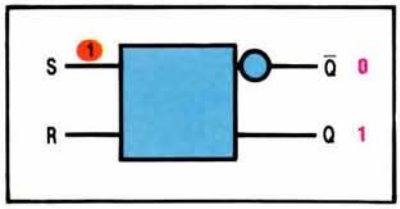
se Q=1 allora $\overline{Q}$=0 (stato di SET)
se Q=0 allora $\overline{Q}$=1 (stato di RESET)

Supponiamo che il flip-flop si trovi inizialmente in stato di RESET. Un segnale inviato sulla linea S farà sì che il circuito "salti" (flip) allo stato di SET.

1) Stato iniziale (RESET)

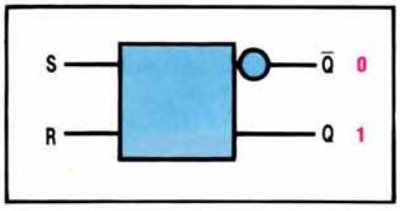


2) Segnale sulla linea SET (S)



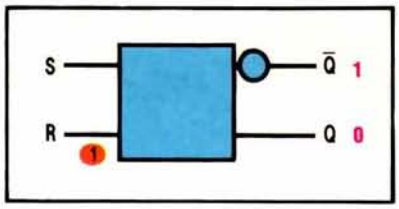
Quando il segnale sulla linea S cessa, il circuito resta stabilmente nello stato di SET.

3) Il circuito rimane stabile (SET)



Un segnale inviato sulla linea R, riporta indietro (flop) il circuito al suo stato primitivo di RESET.

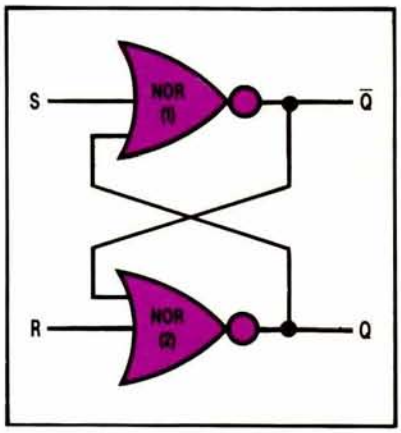
4) Segnale sulla linea RESET



Dopo aver descritto il funzionamento di un flip-flop R-S, esaminiamo brevemente gli elementi logici del circuito.

Il circuito flip-flop R-S

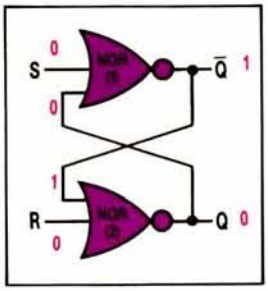
Un flip-flop può essere costruito con varie tecniche, ad esempio collegando due porte NAND oppure, come nell'esempio in figura, collegando due porte NOR in modo che l'uscita di ciascuna porta sia collegata ad uno degli ingressi dell'altra. Questa sorta di "linea di ritorno" dei segnali logici consente al flip-flop di avere una "memoria".



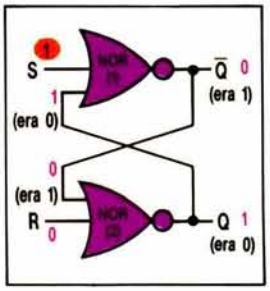
Consideriamo adesso le funzioni SET e RESET del flip-flop e vediamo in qual modo la combinazione di porte NOR permetta di ottenerle. Se supponiamo che, inizialmente, il flip-flop si trovi in RESET e che non vi sia alcun segnale in ingresso, allora esso si troverà in condizioni di stabilità. (Ricordiamo che una porta NOR produce un output di 1 se entrambi gli input sono 0). Un segnale sulla linea S altera

questa situazione di stabilità e provoca il passaggio a 0 dell'uscita "non Q" ($\overline{Q}$). Questo cambiamento modifica il segnale in ingresso alla seconda porta NOR dalla "linea di ritorno" e cambia in 1 l'uscita Q di questa porta. Tale "salto", a sua volta, farà sì che, se un segnale uguale al precedente è presente anche sulla prima porta NOR (1), entrambi gli input della porta NOR (1) risultino uguali ad uno. Quindi anche l'output della porta NOR (1) risulterà essere 0 ed il circuito si troverà nuovamente in uno stato stabile, ad esempio SET.

1) Stato iniziale (RESET)

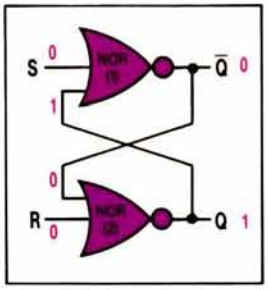


2) Un segnale sulla linea SET

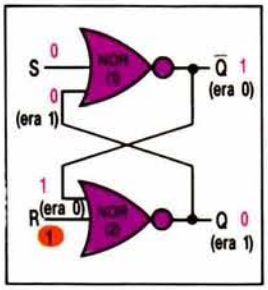


Il circuito permane in uno stato stabile anche quando cessa il segnale sulla linea S. Se applichiamo un segnale alla linea R, il circuito entra nuovamente in uno stato di instabilità e, dopo una ripetizione del processo appena descritto, il circuito ritorna stabile, ma in stato di RESET.

3) Il circuito risulta stabile

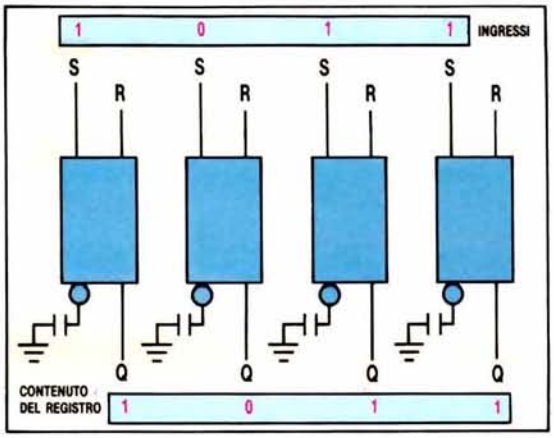


4) Un segnale sulla linea RESET

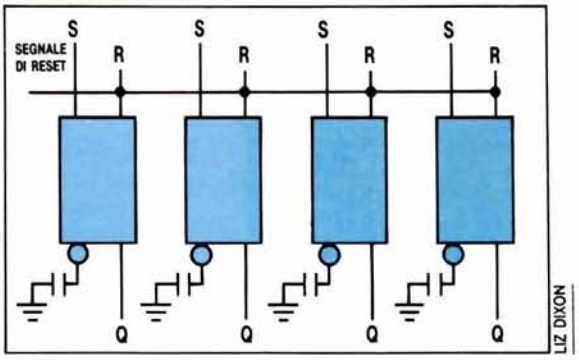


I registri

Il microprocessore di un microcomputer è dotato di una nutrita serie di registri (l'accumulatore, i registri di istruzione e di indice ecc.), che di norma contengono numeri di otto bit, ossia valori binari compresi tra 0 e 255. Poiché questi registri devono ricevere e memorizzare informazioni in forma binaria, è abbastanza logico che siano costituiti da una serie di 8 flip-flop. Per sostenere questa affermazione, esaminiamo la procedura seguita da un registro a quattro bit per ricevere e memorizzare dei valori numerici. Se vogliamo immagazzinare il numero 1011, è necessario che la configurazione binaria venga trasmessa sulla linea S.



Si noti che in questo circuito non viene usata l'uscita "non Q". Siccome la configurazione binaria in ingresso viene inviata sulle linee S dei quattro flip-flop, le loro linee Q presenteranno un'uscita corrispondente. Tuttavia, per scrivere un secondo numero, ad esempio 0110, al posto di quello già immagazzinato nel registro, non è sufficiente inviare la sua configurazione binaria sulle linee S dei flip-flop. In effetti, limitandoci a questa operazione, nel registro avremmo come risultato il numero 1111. Gli 1 in posizione esterna, nel numero ottenuto, derivano dal primo numero memorizzato. Per eliminare questo problema è necessario azzerare ciascun flip-flop prima di immagazzinare il secondo numero. Dato che tutti i flip-flop devono essere azzerati contemporaneamente, conviene eseguire quest'operazione collegando le quattro linee R ed inviando un singolo segnale di azzeramento.



Nella prossima lezione prenderemo in esame altri circuiti sequenziali: i flip-flop di tipo D ed i J-K.

Esercizio 7

1) Che cos'è un flip-flop, noto anche come "bistabile"?
2) Quando accendiamo un computer, un flip-flop si trova inizialmente nel seguente stato:

   $Q=0$, $\overline{Q}=0$, $S=0$, $R=0$

   a) È in condizioni di stabilità?
   b) Se no, quale stato assumerà il flip-flop?
   c) Può lo stato del flip-flop cambiare rispetto a quello assegnatogli con la precedente risposta? (Suggerimento: iniziare dall'altra porta)
   d) Che cosa occorre per esser certi che, quando accendiamo un computer, tutti i registri si trovino in una condizione nota a priori?

Poligono di tiro

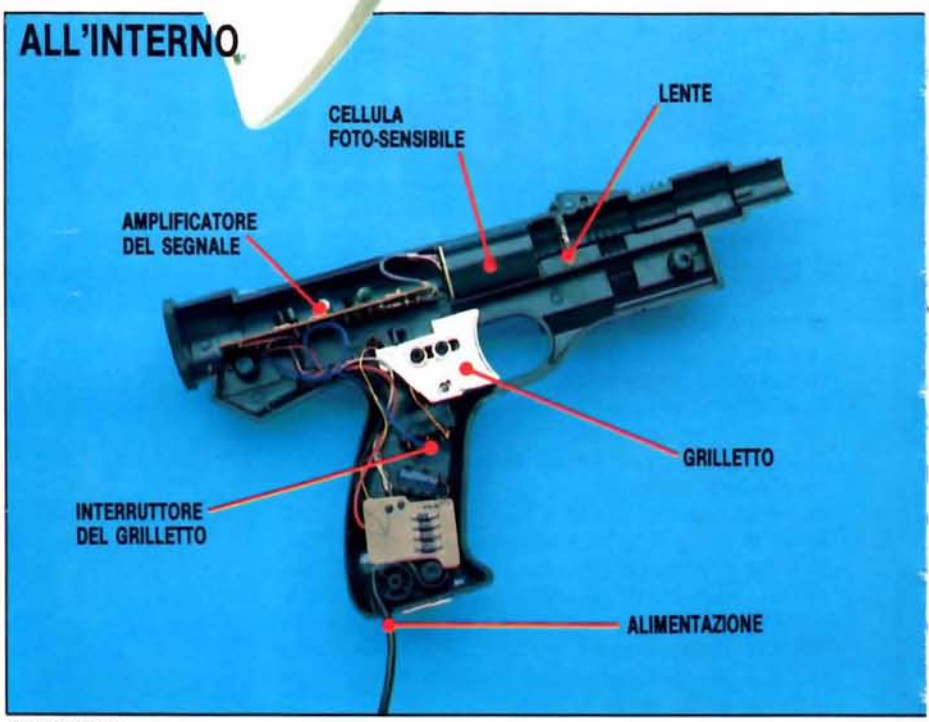
Lo Stack Light Rifle (SLR) aggiunge più realismo ai giochi basati sul tiro al bersaglio computerizzato

Il componente principale dello "Stack Light Rifle System" è la pistola a bersaglio elettronico, collegata al computer da un lungo cavetto. Sul cavo da inserire nel computer è presente, a seconda delle versioni, un connettore adatto al particolare apparecchio. Per lo Spectrum ZX, il connettore contiene due chip e qualche semplice componente, per poter interfacciare i circuiti elettronici principali, posti all'interno del fucile, con il computer. Innestando un "calcio" sulla pistola, una "canna" lunga ed un falso mirino telescopico, è possibile rendere l'arma più precisa e trasformarla in un fucile.

Il circuito elettronico della pistola è costituito da un diodo rivelatore di luce, da un piccolo amplificatore e da un buffer. La luce che penetra attraverso la canna viene focalizzata da una piccola lente di plastica sul foto-diodo, in grado di rivelare i cambiamenti d'intensità dell'immagine presente sullo schermo. Il segnale, intensificato dall'amplificatore, viene quindi "tosato", per essere trasformato in un impulso digitale ed inviato al computer passando attraverso l'interruttore collegato al grilletto. L'area dello schermo soggetta a scansione in un dato momento è quella inquadrata dalla canna del fucile e quando il computer riceve un segnale dal Light Rifle, confronta il valore contenuto nei suoi registri di scansione con la posizione sullo schermo del bersaglio e, se i due valori corrispondono, assegna un "centro" al giocatore.

Sono oggi disponibili versioni del Light Rifle adatte allo Spectrum, al Vic-20 della Commodore e al Commodore 64, benché tutte funzionino secondo lo stesso principio. La Stack ha corredato il Light Rifle di tre giochi su cassetta e, sebbene alcune case di software indipendenti abbiano realizzato giochi che sembrano adattarsi correttamente a questo tipo di periferica, in realtà sono poche quelle che hanno prodotto o modificato programmi in modo da funzionare con questo nuovo sistema. Un secondo fattore che può limitare le potenzialità di vendita del Light Rifle è il fatto che la Stack non fornisce routine "pilota", che consentano all'utente di sviluppare programmi propri. Questa omissione, sommata alla mancanza di dettagli tecnici sul funzionamento del sistema, fa capire che il Light Rifle non può essere considerato una vera e propria alternativa al classico joystick.





CHRIS STEVENS



Il Light Rifle si basa sugli stessi principi operativi delle penne ottiche, ma è stato progettato per essere tenuto a circa 3 metri dallo schermo televisivo e non a contatto con esso: un lungo tubo scuro (la "canna" del fucile) ed una lente permettono di filtrare gran parte della luce ambientale. L'azione combinata di questi due elementi fornisce un ragionevole grado di precisione e consente al giocatore di sparare anche stando seduto. I giochi di corredo sono piuttosto limitati e non consentono di sfruttare appieno le potenzialità del sistema: sia la grafica che lo sviluppo del gioco sono poco entusiasmanti.
Uno dei maggiori problemi nella programmazione sia delle penne che delle carabine ottiche, è che i programmi devono possedere una notevole efficienza. Ad esempio, in tutti i programmi forniti dalla Stack, il gioco si ferma quando viene premuto il grilletto. Ciò avviene perché il metodo comunemente usato per le penne ottiche, ossia una continua scansione dello schermo, rallenterebbe eccessivamente il gioco. Quindi, quando si spara un colpo col Light Rifle, è necessario che sia il software a stabilire se il bersaglio sullo schermo è allineato con la canna del fucile, dopodiché il gioco può riprendere in base al risultato. In teoria, la parte di programma necessaria ad analizzare la nuova porzione di video sulla quale è puntata la carabina non dovrebbe essere molto lunga, ma non sempre questa condizione viene rispettata.
Non esiste ancora una versione del Light Rifle per il Micro BBC, ma, essendo questo dotato di un chip video già predisposto all'impiego di una penna ottica, i compiti del software risulterebbero molto semplificati. Il Commodore 64 possiede un sistema simile, mentre lo ZX Spectrum, sul quale è stato sperimentato il Light Rifle, manca di questa caratteristica e ciò viene evidenziato dal tempo impiegato per determinare la posizione del fucile quando si preme il grilletto.

UNO SPARO... NEL BUIO



Nel Light Rifle la cellula foto-sensibile rivela il passaggio del punto prodotto dalla scansione dell'immagine televisiva. Il software controlla ininterrottamente il registro del contatore di scansione del computer ed è quindi costantemente aggiornato sulla posizione del fascio di scansione. Quando si preme il grilletto del Light Rifle esso segnala di aver individuato il fascio di scansione ed il programma converte il valore presente nel contatore di scansione nelle coordinate video X, Y del punto mirato dalla carabina

STAGIONE DI CACCIA



"High Noon" è il migliore dei tre programmi dimostrativi della Stack. L'animazione è discreta ed il "pistolero" può voltarsi rapidamente e sparare. Sia "Grouse Shoot" che "Shooting Gallery" offrono un singolo bersaglio mobile, da centrare prima che esca dal perimetro di gioco. Il movimento dei bersagli è a scatti, e, dopo lo sparo, il gioco si blocca fastidiosamente per qualche attimo

IAN McKINNELL

Gli sprite del Commodore

Come definire caratteri e simboli speciali usando le notevoli risorse grafiche del Commodore 64

La realizzazione di caratteri grafici propri, sul Commodore 64, non è un processo semplice: non ci sono comandi specifici nel BASIC in dotazione, quindi l'intera operazione deve essere eseguita usando i comandi PEEK e POKE per inserire o cambiare i valori contenuti in memoria.

L'area di memoria per il set di caratteri del Commodore 64 è costituita da una sezione di ROM, che inizia alla locazione di memoria 53248. Ciascun carattere appare sullo schermo come una configurazione di punti disposti in una matrice di 8 × 8. Per definire ciascuna di queste configurazioni sono necessari 64 bit, ovvero otto byte. Gli otto byte compresi fra le locazioni 53248 e 53255 definiscono il carattere "@", che è il primo della serie. Il suo codice video è 0 per cui, se inseriamo (POKE) tale valore in uno dei byte della RAM video, sul video compare questo carattere. I successivi otto byte, da 53256 a 53263, definiscono la "A" (codice video 1) e così via.

Poiché si tratta di memoria ROM, non è possibile modificare queste configurazioni direttamente: è necessario trasferirle parzialmente o totalmente nella RAM per apportarvi le variazioni desiderate. Successivamente, si potrà utilizzare il nuovo set di caratteri RAM sullo schermo, sostituendolo a quello normale.

Il set di caratteri ROM condivide i propri indirizzi di memoria con le unità di input/output (ossia i registratori a cassetta e le unità a dischetti). Solitamente, la CPU 6510A adopera questa zona di memoria per le operazioni di ingresso/uscita, ma può essere riprogrammata affinché la consideri come una zona di memoria riservata al set dei caratteri. Ciò sembrerà strano dato che, normalmente, non è compito della CPU estrarre le definizioni dei caratteri per inviarle allo schermo. Infatti, questo incarico viene normalmente demandato ad un chip ausiliario controllato dalla CPU. Il contenuto della locazione 1 definisce il tipo di operazione di I/O ed il bit 2 di tale locazione si comporta come un interruttore, guidando la CPU nella scelta del set di caratteri. Se questo bit è posto a zero, la CPU ne deduce che la zona di memoria è dedicata, attualmente, alle operazioni di I/O. Gli altri bit della locazione 1 svolgono altre funzioni di controllo sul sistema, assai simili a quella appena citata: è quindi necessario prestare molta attenzione a non alterarne lo stato, quando modifichiamo il valore del bit 2. Il modo migliore per eseguire questa operazione è usare gli operatori logici AND e OR.

Supponiamo che il contenuto della locazione 1 sia:

Bit	7	6	5	4	3	2	1	0
	0	1	1	0	1	1	1	1

e che si voglia portare a zero il bit 2. Potremmo raggiungere il nostro scopo calcolando il valore decimale di 01101111 ed inserendo il risultato nella locazione 1, ma ciò sarebbe possibile solo se già conoscessimo il precedente valore contenuto nella locazione 1, cioè 01101111. Il miglior sistema per modificare soltanto il bit 2 consiste nell'usare le istruzioni AND e PEEK. Il seguente comando permette di conoscere (PEEK) il valore iniziale della locazione 1, lo pone in AND rispetto a 251 (11111011) ed infine deposita (POKE) il risultato ottenuto nella locazione 1:

```
POKE 1,PEEK(1) AND 251
```

L'effetto di questa serie di comandi è mostrato nella seguente tabella:

Bit	7	6	5	4	3	2	1	0	
	0	1	1	0	1	1	1	1	= Valore iniziale
AND	1	1	1	1	1	0	1	1	= 251 in binario
	0	1	1	0	1	0	1	1	Risultato dell'operazione logica AND

Indipendentemente dal valore iniziale del bit 2, l'operazione "AND rispetto a zero" dà sempre come risultato zero, mentre ponendo in "AND rispetto ad uno" tutti gli altri bit della locazione, non cambieremo il loro valore iniziale. Il numero binario 11111011 (251 in decimale) viene chiamato "maschera" ed è stato qui usato come "maschera-AND".

Il bit 2 può essere posto a 1, senza modificare gli altri bit, usando questa sequenza di comandi:

```
POKE 1, PEEK(1) OR 4
```

Bit	7	6	5	4	3	2	1	0	
	0	1	1	0	1	0	1	1	= Valore iniziale
OR	0	0	0	0	0	1	0	0	= 4 in binario
	0	1	1	0	1	1	1	1	Risultato dell'operazione logica OR

Questa procedura fa sì che il BASIC non vada a scrivere sopra il nostro set di caratteri. Completato il trasferimento in RAM, la CPU può essere nuovamente abilitata a selezionare le unità di I/O.

L'operazione finale consiste nel costringere il chip di controllo del video a usare il nostro set di caratteri, anziché quello situato nella ROM. I bit 0 e 3 della locazione 53272 puntano all'indirizzo iniziale del set di caratteri e la seguente tabella mostra come il Commodore 64 interpreti i valori di questi bit puntando ai loro specifici indirizzi di memoria:

Valore decimale dei bit 0-3	Bit 3,2,1,0	Locazione puntata
0	0000	0
2	0010	2048
4	0100	4096
6	0110	6144
8	1000	8192
10	1010	10240
12	1110	12288
14	1110	14336

Il bit zero, in questo registro, è insignificante, mentre i bit 4 e 7 controllano altre funzioni e non devono essere modificati. Useremo quindi il numero binario 11110000 (240 in decimale) come "maschera-AND" e 00001110 (14 in decimale) come "maschera-OR" per ottenere che il registro punti all'indirizzo 14336, la locazione di avvio del nostro set di caratteri:

```
POKE 53272, (PEEK(53272) AND 240) OR 14
```

Ora, utilizzando un ciclo FOR...NEXT, trasferire in RAM il set di caratteri ROM. Durante questa operazione la CPU non è in grado di rilevare gli "interrupt" trasmessi dalle unità di I/O. La tastiera, ad esempio, invia un interrupt ogni sessantesimo di secondo, permettendo così alla CPU d'interpretare le battute dei tasti. Questi interrupt vengono prodotti dal sistema di temporizzazione. Se la CPU venisse disturbata da segnali provenienti da un'unità di I/O, durante il trasferimento dei caratteri, molto probabilmente il computer si bloccherebbe e sarebbe necessario spegnere e riaccendere l'apparecchio. Fortunatamente questo meccanismo può essere disattivato azzerando il bit 0 della locazione 56334 (gli altri bit di questa locazione devono restare immutati). Useremo quindi la sequenza:

```
POKE 56334,PEEK(56334) AND 254
```

Il trasferimento avrà inizio solo dopo che il sistema d'interruzione è stato disattivato e la CPU costretta ad esaminare il set di caratteri in ROM.
Supponiamo di voler trasferire i 64 caratteri compresi tra "@" e "?" (codice video da 0 a 63): dovremo copiare in un appropriato blocco RAM le 512 locazioni (8×64=512) che hanno come indirizzo iniziale 53248. Possiamo usare svariate aree di memoria e, nel nostro esempio, abbiamo scelto un blocco con indirizzo iniziale 14336, che di solito si trova nell'area di programma del BASIC. Possiamo proteggere l'area scelta modificando il valore contenuto in 65 (il puntatore al limite superiore della memoria):

```
POKE 56,32
```

Creazione di nuovi caratteri

Ogni carattere viene disegnato su una matrice di 8 × 8 punti. Ciascuna riga della matrice viene interpretata come un numero binario (i punti illuminati corrispondono agli 1, quelli che non appaiono sul video a 0) ed occupa un byte di memoria: per l'intero carattere occorrono dunque otto locazioni consecutive di memoria. L'indirizzo iniziale di un carattere può essere calcolato moltiplicando l'indirizzo iniziale dell'intero blocco di memoria per il codice video di quel particolare carattere:

Prima locazione = 14336 + 8 × (codice video)

Conosciuti gli indirizzi dei byte che definiscono un carattere, possiamo depositare (POKE) al loro interno i nuovi valori, modificando così la configurazione di punti che apparirà sul video quando useremo il nuovo carattere. A questo punto è sufficiente battere il tasto appropriato, perché il carattere appaia sul video. Nel seguente programma dimostrativo vengono ridefiniti i 4 caratteri [,],£. e ↑ (codici 27-30) per creare l'immagine di un uomo, che viene poi animata inviando in successione sul video due sue diverse versioni.

```
130 :
140 REM ***COPIA DEI CARATTERI ROM***
150 POKE56,32
    :REM AGGIUSTA LIMITE SUPERIORE
    MEMORIA
160 POKE56334,PEEK (56334) AND254
    :REM DISABILITA LE INTERRUPT
165 POKE1,PEEK (1) AND251
    :REM SELEZIONA I CARATTERI ROM
170 FOR I=0 TO 511
    :REM COPIA
180 POKE14336+I,PEEK (53248+I)
    :REM 64
190 NEXTI
    :REM CARATTERI
200 POKE1,PEEK (1) OR4
    :REM SELEZIONA MEMORIA I/O
210 POKE56334,PEEK (56334) OR1
    :REM RIABILITA LE INTERRUPT
220 POKE 53272,(PEEK(53272)AND240) OR14
    :REM PUNTATORE AL SET DI CARATTERI
230 REM ***COPIA COMPLETATA***
240 :
250 :
300 REM ***CREAZIONE DELL'OMINO***
310 FOR I=14552TO14552+31
    :REM LETTURA DELLE DATA
320 READ A:POKEI,A:NEXT I
    :REM PER I CARATTERI
330 PRINTCHR$ (147)
    :REM PULIZIA DELLO SCHERMO
340 POKE55338,14:POKE55378,14
    :REM COLORAZIONE AZZURRA
350 POKE1066,27:POKE1106,28
    :REM CON BRACCIA IN ALTO
360 FORI=1TO500:NEXT  I
    :REM CICLO DI RITARDO
370 POKE1066,29:POKE1106,30
    :REM CON BRACCIA IN BASSO
380 FORI=1TO500:NEXT I
    :REM CICLO DI RITARDO
390 GOTO350
480 :
490 :
500 REM ***DATI PER BRACCIA IN ALTO***
510 DATA129,153,189,153,66,60,60,60
520 DATA60,60,36,36,66,66,195,0
530 REM ***DATI PER BRACCIA IN BASSO***
540 DATA0,24,60,24,0,126,189,189
550 DATA189,189,36,36,36,36,102,0
```

128	64	32	16	8	4	2	U	DECIMAL
1	0	0	0	0	0	0	1	129
1	0	0	1	1	0	0	1	153
1	0	1	1	1	1	0	1	189
1	0	0	1	1	0	0	1	153
0	1	0	0	0	0	1	0	66
0	0	1	1	1	1	0	0	60
0	0	1	1	1	1	0	0	60
0	0	1	1	1	1	0	0	60
0	0	1	1	1	1	0	0	60
0	0	1	1	1	1	0	0	60
0	0	1	0	0	1	0	0	36
0	0	1	0	0	1	0	0	36
0	1	0	0	0	0	1	0	66
0	1	0	0	0	0	1	0	66
1	1	0	0	0	0	1	1	195
0	0	0	0	0	0	0	0	0

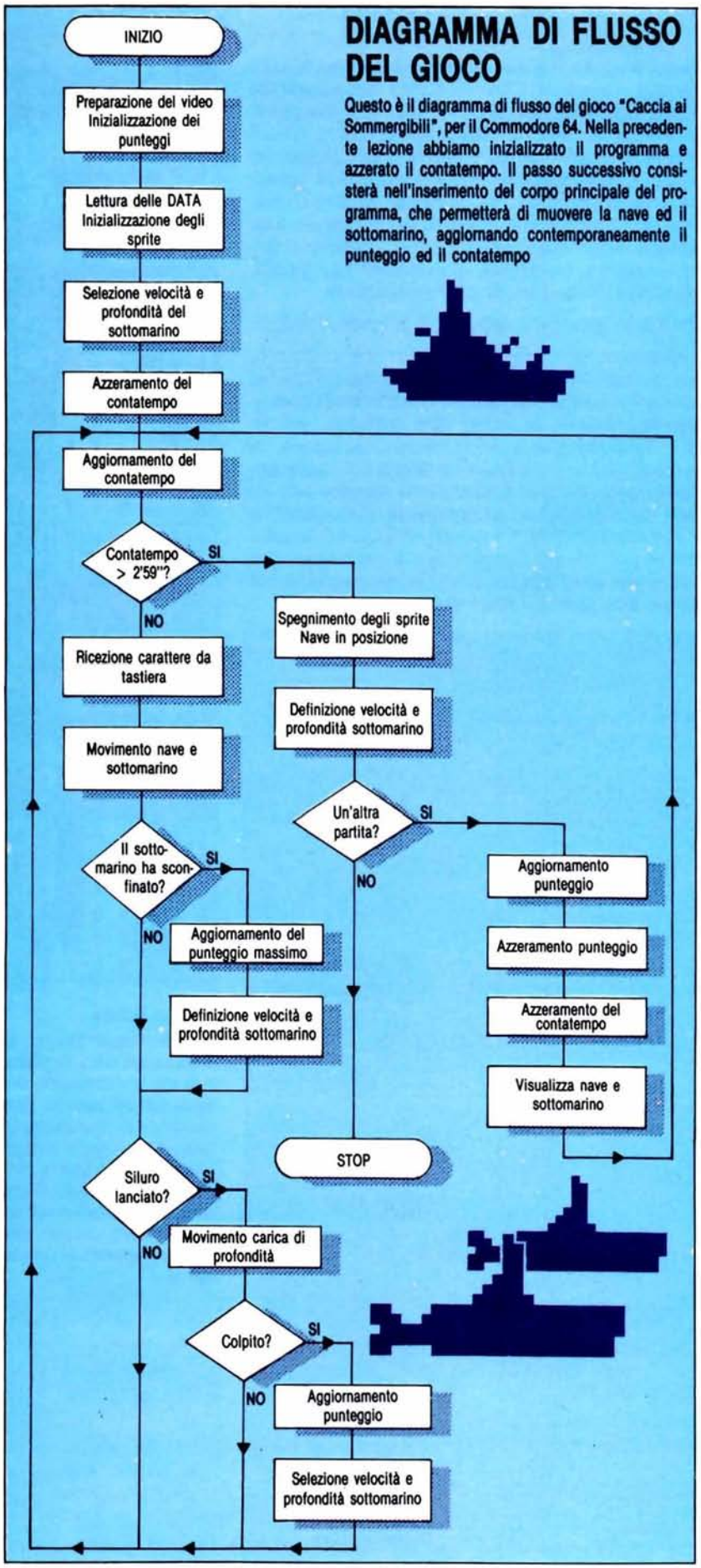
128	64	32	16	8	4	2	U	DECIMAL
0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	24
0	0	1	1	1	1	0	0	60
0	0	0	1	1	0	0	0	24
0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	126
1	0	1	1	1	1	0	1	126
1	0	1	1	1	1	0	1	189
1	0	1	1	1	1	0	1	189
1	0	1	1	1	1	0	1	189
0	0	1	0	0	1	0	0	36
0	0	1	0	0	1	0	0	36
0	0	1	0	0	1	0	0	36
0	0	1	0	0	1	0	0	36
0	1	1	0	0	1	1	0	102
0	0	0	0	0	0	0	0	0

Immagine animata

I caratteri vengono disegnati su di una matrice di 8 × 8 codificata in otto byte consecutivi. Ciascuna riga del carattere viene interpretata come un numero binario a singolo byte nel quale gli 1 rappresentano i punti e gli 0 indicano gli spazi vuoti. Questi numeri binari devono essere poi convertiti in decimale, per essere usati nei programmi del Commodore 64

DIAGRAMMA DI FLUSSO DEL GIOCO

Questo è il diagramma di flusso del gioco "Caccia ai Sommergibili", per il Commodore 64. Nella precedente lezione abbiamo inizializzato il programma e azzerato il contatempo. Il passo successivo consisterà nell'inserimento del corpo principale del programma, che permetterà di muovere la nave ed il sottomarino, aggiornando contemporaneamente il punteggio ed il contatempo



OSSERVAZIONI SUL PROGRAMMA

Il Commodore 64 ha un temporizzatore incorporato, molto utile per scandire il tempo nei programmi BASIC, dotato di 6 cifre, come un orologio digitale. Queste cifre indicano le ore (00-23), i minuti (00-59) ed i secondi (00-59). Il temporizzatore può essere manipolato in BASIC mediante la variabile TI$. Il valore di TI$ indica il tempo trascorso da quando il computer è stato acceso e può essere azzerato in qualsiasi momento. Il programma mostra come funziona il temporizzatore.

```
10 REM ***TEMPORIZZATORE***
20 PRINT CHR$(147 : REM RIPULISCE LO SCHERMO
30 TI$ = "000000" : REM AZZERA IL TEMPORIZZATORE
40 PRINT CHR$(145);TI$ : REM VISUALIZZA LA CIFRA CONTENUTA NEL TEMPORIZZATORE
50 : REM CHR$(145) = CURSORE IN ALTO
60 GOTO 40 : REM RIPETI IL TUTTO
```

Il programma si ripete all'infinito, visualizzando il contenuto del temporizzatore, finché non viene premuto RUN/STOP. Il gioco "Caccia ai Sommergibili", nel quale viene utilizzato un contatempo, termina quando sono trascorsi tre minuti: per il contatempo, quindi, è impiegata soltanto quella parte di TI$ che si riferisce ai minuti ed ai secondi. Grazie alle funzioni per le stringhe, possiamo utilizzare la variabile TI$ nel seguente modo:

RIGHT$(TI$,2)
↓
TI$ = HH(MM)(SS)
↑
MID$(TI$,3,2)

Le due cifre dei secondi possono essere "estratte" con l'istruzione RIGHT$(TI$,2) e quelle dei minuti con l'istruzione MID$(TI$,3,2).
Il corpo principale del programma inizia alla linea 200 e termina alla linea 390. Possiamo adesso richiamare in memoria la subroutine descritta nella scorsa lezione (e memorizzata su nastro), per aggiungervi le seguenti linee di programma:

```
140 TI$ = "000000"
150 :
160 :
200 REM ***SEZIONE PRINCIPALE***
205 :
210 REM ***TEMPORIZZATORE***
220 PRINT CHR$(19);:TAB(14)CHR$(5)
    "TEMPO ";MID$(TI$,3,2);":"; RIGHT$(TI$,2)
225 IF VAL(TI$)>259 THEN 400: REM FINE DEL GIOCO
:
390 GOTO 200: REM RIAVVIA LA SEZIONE PRINCIPALE
400 END
```

La linea 140 azzera il contatempo all'avvio del programma. La linea 220 visualizza il tempo in minuti e secondi, separati da ":". TAB(14) permette di lasciare 14 spazi a sinistra della scritta "TEMPO" e visualizza il contatempo in posizione centrata. CHR$(5) assegna ai caratteri il colore bianco. La linea 225 trasforma TI$ in una quantità numerica: ciò permette di vagliarne il contenuto. Se il tempo ha superato i 2 minuti e 59 secondi, il gioco termina.

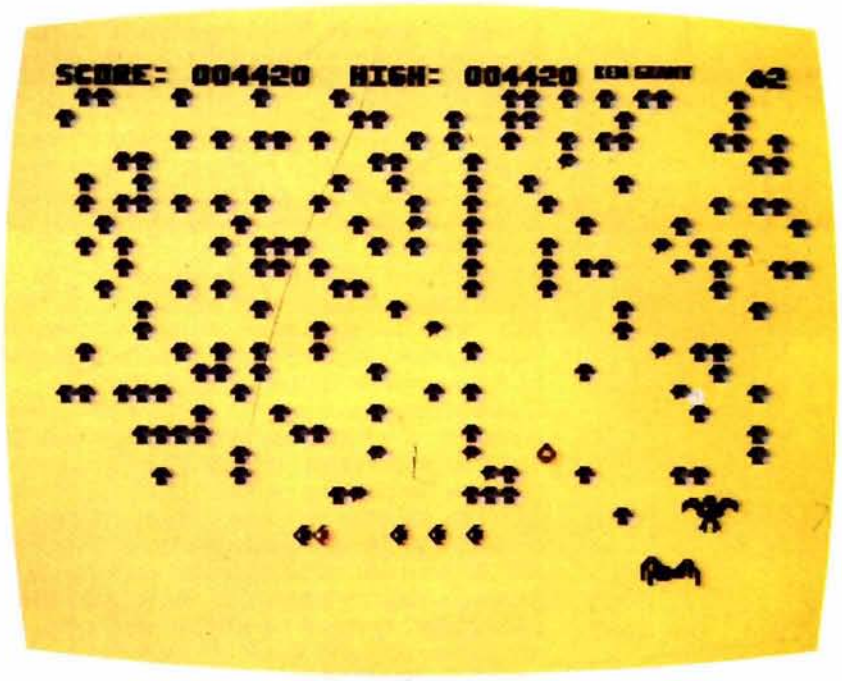
Software in vetrina

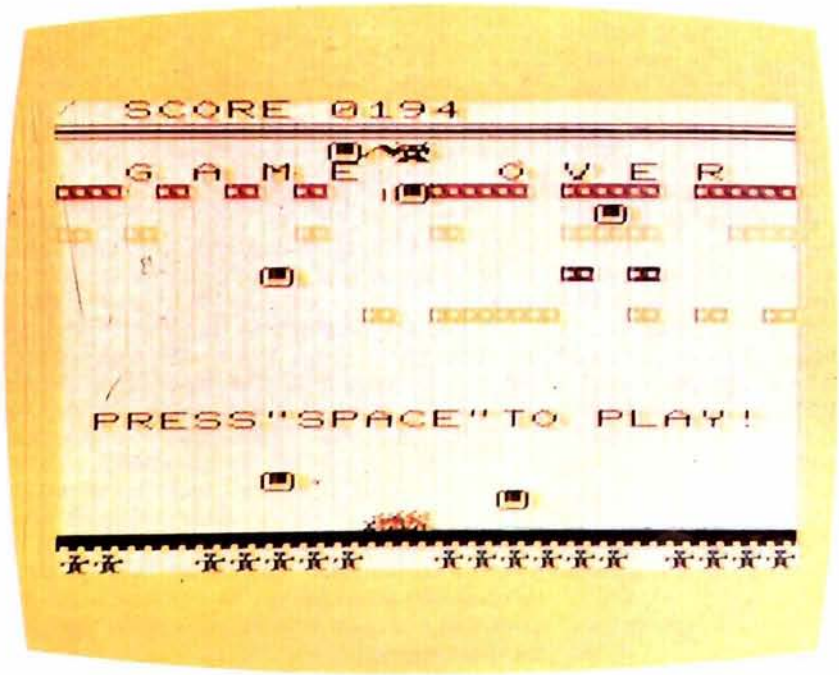
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Azione mozzafiato in due giochi per i Commodore. La trama non è del tutto nuova per gli appassionati dei giochi "arcade", tuttavia il divertimento non mancherà.

Exterminator per il Commodore 64

Per gli affezionati di *Centipede* (uno dei giochi più "gettonati" nelle sale giochi), questo programma costituisce una variante di sicura attrazione. Oltre a piccole creature tipo Centipede, il giocatore deve affrontare (sullo schermo, fortunatamente): un'aquila impazzita, una gigantesca tarantola, micidiali zanzare ed un immenso scorpione. La grafica a sprite del Commodore viene messa davvero a buon frutto, con alcuni effetti assai realistici. Ogni animale, inoltre, emette un suono caratteristico. Exterminator è prodotto dalla "Bubble Bus".





Antimatter splatter per il Vic 20

Il "cattivo" della situazione è Traal, che intende distruggere la terra con un diluvio di bombe antimateria. L'unica arma di cui il giocatore è dotato, per difendere i propri consimili, è un "cannone positronico". Forse lo svolgimento del gioco, che assomiglia in parte al notissimo "invasori spaziali", e la difficoltà nel distruggere le troppo veloci bombe antimateria potrà frenare l'entusiasmo dei giocatori meno provetti. Per il gioco, prodotto dalla Rabbit Software, è necessario un joystick.

Subroutine trasportabili

L'uso di simboli e di etichette facilita la stesura di sottoprogrammi di applicazione generica

Poiché l'Assembly è un semplice linguaggio di programmazione composto da una serie di comandi "primitivi", interpretati dalla CPU, capita spesso di riscrivere più volte parti di programma, le cui funzioni diamo invece per scontate adoperando linguaggi ad alto livello. Ci riferiamo, per esempio, alle procedure di input/output o alle routine aritmetiche su due byte.
Per facilitare il lavoro di programmazione, è consigliabile crearsi una biblioteca, su nastro, disco o carta, delle routine adoperate con maggior frequenza, per poterle riutilizzare nella stesura di nuovi programmi.
Questo sistema presenta però due grossi problemi. Il primo riguarda la difficoltà di scrivere importanti sezioni di programma, talvolta lunghe, in modo sufficientemente generico da poterle inserire in programmi di diverso tipo senza troppe correzioni o modifiche. Il secondo consiste nello scrivere utili routine, non rigidamente relegate in una particolare zona di memoria, ma che, al contrario, possano venir rilocate (ossia assemblate in un'altra parte della memoria, senza alterarne la funzione originaria).
Entrambi i problemi non sono che aspetti particolari del più vasto problema della "trasportabilità del software", ben noto ai programmatori BASIC.
Questi problemi possono venire risolti o ricorrendo all'impiego di variabili (per trasferire valori dal programma alle subroutine), oppure adoperando variabili "locali" nelle subroutine (rendendole indipendenti dal più vasto contesto del programma), o, ancora, evitando l'uso di grandezze assolute (costanti numeriche o stringhe) e di numeri di linea del programma.
Nella programmazione Assembly ci siamo abituati a considerare le locazioni di memoria come un equivalente delle variabili BASIC: i programmi elaborano il contenuto di queste locazioni, qualunque esso sia, così come accade con le variabili del BASIC. Per maggior chiarezza, fino ad ora, ci siamo riferiti alle locazioni di memoria usando il loro indirizzo assoluto: d'ora in poi occorrerà però adottare una metodica che consenta una maggiore elasticità. Useremo quindi, al posto degli indirizzi e dei valori assoluti, le varie forme simboliche offerte dagli pseudo-opcode, quali equivalenti sia delle variabili, che dei numeri di linea del programma. Consideriamo, ad esempio, il seguente programma:

6502			Z80		
DATA1	EQU	$12	DATA1	EQU	$12
DATA2	EQU	$79	DATA2	EQU	$79
	LDA	DATA1	LD	A,(DATA1)	
LOOP	ADC	DATA2	ADC	A,(DATA2)	
	BNE	LOOP	JR	NZ,LOOP	
	RTS		RET		

Il listato contiene due tipi di simboli (due valori ed un'etichetta), entrambi usati come operandi nelle istruzioni Assembly. Grazie ad essi il frammento di programma è di applicazione generale e facilmente rilocabile. Le sole quantità assolute sono i valori di DATA1 e DATA2 ed essi possono venire inizializzati in un'altra parte del programma piuttosto che all'inizio della routine.
Ci sono altri pseudo-opcode che non abbiamo ancora trattato: in particolare DB (Define Byte), DW (Define Word) e DS (Define Storage) che, come ORG ed EQU, possono differire da un programma assemblatore all'altro. Questi pseudo-opcode permettono di inizializzare ed allocare locazioni di memoria, come nel seguente esempio:

			ORG	$D3A0
D3A0	5F	LABL1	DB	$5F
D3A1	CE98	LABL2	DW	$98CE
D3B3		LABL3	DS	$10
D3B3		DATA1	EQU	LABL3

SYMBOL TABLE
LABL1 = D3A0:LABL2 = D3A1:LABL3 = D3A3
DATA1 = D3A3
ASSEMBLY COMPLETE—NO ERRORS

Nella parte finale di questo listato (prodotto da un programma assemblatore), incontriamo per la prima volta una "tavola dei simboli" (SYMBOL TABLE), composta dai simboli definiti nel programma e dai valori che essi rappresentano. Si notino alcune importanti novità nel listato: innanzi tutto, nella linea che comincia con LABL1 viene usato lo pseudo-opcode DB. Inoltre, ORG ha assegnato a LABL1 l'indirizzo $D3A0 e ciò è confermato dalla tavola dei simboli. In questo esempio, DB fa sì che il valore $5F venga depositato nel byte indirizzato da LABL1 (corrispondente all'indirizzo $D3A0), come possiamo verificare esaminando la seconda colonna di codice macchina del listato.

Il simbolo LABL2 rappresenta l'indirizzo $D3A1. Tuttavia, DW ha l'effetto di depositare una "parola" (cioè due byte consecutivi) nella memoria, quindi il valore $98CE viene immagazzinato nelle locazioni $D3A1 e $D3A2 in forma basso-alto, come si vede nella colonna di codice macchina. Poiché DW converte automaticamente i suoi operandi nella forma basso-alto, viene spesso usato per inizializzare le locazioni di memoria con opportuni indirizzi. LABL2 (locazione $D3A1) potrebbe essere il deposito per un "puntatore" all'indirizzo $98CE.

Una terza particolarità del listato è l'istruzione DS $10, che ha l'effetto di sommare $10 al contatore di programma. Ciò appare più chiaramente nella tavola dei simboli che nel listato. LABL3 rappresenta la locazione $D3A3 (la locazione che segue la precedente istruzione), sebbene nel listato il suo valore sembri essere $D3B3. Questo è in realtà l'indirizzo di locazione dell'istruzione successiva all'istruzione DS, quindi DS $10 ha riservato un blocco di 16 byte (da $D3A3 a $D3B2 incluso) fra le due istruzioni. Questa procedura equivale all'inserimento di lunghe linee REM in un programma BASIC, per la creazione di uno spazio libero nell'area di testo di programma da utilizzare per successive POKE e PEEK (vedere a pagina 615).

Infine, l'ultima istruzione (EQU) serve per assegnare ad un simbolo il valore di un altro: in questo caso DATA1 assume il valore $D3A3 (cioè il valore di LABL3). Ciò può tuttavia creare una certa confusione. LABL3 è la rappresentazione simbolica dell'indirizzo di locazione $D3A3, quindi l'istruzione DATA1 EQU LABL3 significa: "il simbolo DATA1 deve avere lo stesso significato e valore del simbolo LABL3". Il fatto che l'istruzione DB abbia reso il contenuto di $D3A3 uguale a $5F non ha alcuna rilevanza per il significato dei simboli LABL3 e DATA1. Una delle maggiori difficoltà, nello studio della programmazione in linguaggio Assembly, è proprio quella di mantenere ben distinti il significato di "locazione" e quello di "contenuto" di essa.

Problemi molto simili si incontrano in BASIC con le variabili ed il loro contenuto.

L'istruzione DB non è un doppione di EQU: LABL1 significa "locazione $D3A0" e DB $5F ha inizializzato questo byte con il valore $5F ma, sebbene il valore di LABL1 sia adesso definito, i contenuti della locazione che simbolizza possono essere cambiati in qualsiasi momento (ad esempio, immagazzinando in una fase successiva del programma i valori contenuti nell'accumulatore). Similmente, il valore del simbolo DATA1 viene determinato dall'istruzione EQU e non può essere cambiato dall'esecuzione del programma.

Infine, LABL3 punta al primo byte di un'area dati di 16 byte, il cui contenuto può essere cambiato nel programma, mentre LABL3 è invariabile.

Questa analisi introduce, ma non esaurisce, la possibilità di nuovi pseudo-opcode. Esaminiamo questa nuova versione del frammento precedente:

			ORG	$D3A0
D3A0	4D4553	LABL1	DB	'MESSAGGIO1'
D3A9	CE98	LABL2	DW	$98CE
D3BB		LABL3	DS	$10
D3BB		DATA1	EQU	LABL3

SYMBOL TABLE
LABL1 = D3A0:LABL2 = D3A9:LABL3 = D3AB
DATA1 = D3AB
ASSEMBLY COMPLETE—NO ERRORS

L'operando dell'istruzione DB è stavolta la stringa "MESSAGGIO 1" e l'assemblatore ha inizializzato le locazioni che vanno da $D3A0 a $D3AB con i valori ASCII dei caratteri che compongono la stringa; ciò può essere dedotto da un esame della colonna degli indirizzi di locazione del listato e parzialmente confermato dalla colonna di codice macchina. I tre byte $D3A0, $D3A1 e $D3A2 contengono rispettivamente $4D, $45 e $53 che sono i codici esadecimali delle lettere M, E e S.

Questa procedura è assai utile non solo perché libera il programmatore dal noioso lavoro di traduzione in codice ASCII di messaggi o di singoli caratteri, ma anche perché rende la lettura del listato molto più semplice e contribuisce alla possibilità di ottenere immagini video dai programmi in linguaggio Assembly. Quest'ultima opportunità è particolarmente significativa perché fino a questo momento ci siamo limitati ad immagazzinare risultati in memoria e ad esaminarli usando il programma Monitor (vedere a pag. 598). Più

ESERCIZI

1) Il primo frammento di programma riportato nel testo usa lo pseudo-opcode DS per riservare un blocco di 10 byte di memoria partendo dall'indirizzo rappresentato dall'etichetta LABL1. Scrivere un programma in Assembly capace di immagazzinare in quel blocco ed in ordine decrescente i valori da $0F a $00, un numero per byte. Dovranno essere impiegati un ciclo e tecniche di indirizzamento ad indice, per le quali sarà necessario utilizzare le istruzioni DEX (decrementa il registro X) o DEC IX (decrementa IX). Il ciclo dovrebbe ripetersi finché il valore del registro di indice non si sia ridotto a 0; a questo punto si useranno le istruzioni di salto BNE o JR NZ.

2) Usando le tecniche dell'esercizio precedente, scrivere un programma per trasferire il messaggio immagazzinato in LABL1 mediante lo pseudo-opcode DB (vedere il secondo frammento di programma riportato nel testo) in un blocco di memoria che parte dall'indirizzo immagazzinato in LABL2, mediante lo pseudo-opcode DW. Se l'indirizzo $98CE non si adatta al computer in uso, sarà necessario cambiare l'inizializzazione. Il programma dovrà però funzionare con qualsiasi indirizzo e con messaggi di qualunque lunghezza. Per realizzare ciò, il programma deve usare il numero di caratteri che compongono il messaggio come un contatore di ciclo, oppure deve essere in grado di individuare la fine del messaggio. Nel secondo caso si potrà inserire, ad esempio, un asterisco come carattere finale di ogni messaggio.

avanti nel corso tratteremo anche della gestione del video. Comunque, se riflettiamo sui consueti modi di immagazzinamento in memoria dei dati e consideriamo che le immagini video registrate in memoria non sono altro che aree di memoria, allora possiamo già intuire come sia possibile creare immagini sul video, dall'interno di un programma.
La caratteristica fondamentale della nuova istruzione DB è che essa conferisce a LABL1 lo stato di una variabile di stringa BASIC. Quando scriviamo in BASIC:

```
200 LET A$="MESSAGGIO 1"
```

stiamo creando un puntatore all'inizio di un blocco di byte contenenti i codici ASCII per le lettere M, E, S e così via. Ogni qual volta l'interprete BASIC incontra un richiamo ad A$, và ad esaminare la sua tavola dei simbóli per trovare la locazione a cui esso punta, cioè la locazione di partenza del contenuto di A$. In modo simile, nel nostro programma in linguaggio Assembly, possiamo trattare LABL1 come l'equivalente di A$, dato che abbiamo già scritto una sezione di programma che ci permette di adoperare una tavola, usando l'indirizzamento ad indice.
Gli pseudo-opcode ci consentono quindi di eliminare dal programma gli indirizzi ed i valori assoluti, sostituendoli con comodi simboli e consentendo la stesura di moduli "rilocabili". Il passo successivo riguarda le modalità di accesso a questi moduli rilocabili, da parte del programma principale. In altre parole, abbiamo bisogno di un'istruzione in codice macchina equivalente al comando GOSUB del BASIC.
Le istruzioni JSR e CALL, rispettivamente nel 6502 e nello Z80, rispondono alle nostre necessità. Entrambe richiedono un indirizzo assoluto (che può essere un'etichetta) come operando e permettono di depositarne il valore nel "contatore di programma". Dopo una JSR o una CALL, la successiva istruzione che verrà eseguita è la prima della subroutine "chiamata". L'esecuzione della subroutine prosegue fino ad incontrare la prima istruzione di "ritorno", rispettivamente RTS e RET. Questo comando causa il ripristino, nel contatore di programma, dell'indirizzo immediatamente successivo a quello dell'istruzione JSR o CALL, provocando così l'esecuzione dell'istruzione che segue JSR o CALL. L'interprete BASIC adotta un identico meccanismo per eseguire il comando GOSUB. Cerchiamo ora di capire come vengono reinseriti i vecchi valori del contatore di programma quando viene eseguita l'istruzione RETURN. I comandi JSR e CALL depositano nello stack il contenuto del contatore di programma (vedere la figura a pag. 616) prima di sostituirlo con l'indirizzo della subroutine, quindi lo recuperano inserendolo nuovamente nel contatore di programma. In una successiva fase del corso vedremo che cosa sia lo stack ed affronteremo i problemi connessi alla sua gestione.

ISTRUZIONI DEL CODICE MACCHINA

BEQ — SALTA SE ZERO — relativo ‖ F0 (2 byte) — 6502

Al contenuto del contatore di programma viene sommato lo scarto specificato nel secondo byte dell'istruzione

EFFETTI SUL RCP

	S	V		B	D	I	Z	C
MSB								LSB

NESSUN EFFETTO

Esempio:

LOCAZIONE	CODICE MACCHINA	LINGUAGGIO ASSEMBLY
8F00	F0 16	BEQ $16

		PRIMA		DOPO
PC	lo	02		18
	hi	8F		8F

F0	$8F00
16	

PC = Program Counter (Contatore di Programma)

MEMORIA PROGRAMMA

JR Z — SALTO RELATIVO SE ZERO — relativo ‖ 28 (2 byte) — Z80

Al contenuto del contatore di programma viene sommato lo scarto specificato nel secondo byte dell'istruzione

EFFETTI SUL RCP

	S	Z	H		V	N	C
MSB							LSB

NESSUN EFFETTO

Esempio:

LOCAZIONE	CODICE MACCHINA	LINGUAGGIO ASSEMBLY
8F00	28 16	JR Z,$16

		PRIMA		DOPO
PC	lo	02		18
	hi	8F		8F

28	$8F00
16	

PC = Program Counter (Contatore di Programma)

MEMORIA PROGRAMMA

INX — INCREMENTA REGISTRO X — implicito ‖ E8 (1 byte) — 6502

Al contenuto del registro X viene sommato 1

EFFETTI SUL RCP

	S	V		B	D	I	Z	C
MSB	X						X	LSB

Esempio:

LOCAZIONE	CODICE MACCHINA	LINGUAGGIO ASSEMBLY
F391	E8	INX

	PRIMA	DOPO
PSR	????????	0??????1?
X	FF	00

E8	$F391
16	

MEMORIA PROGRAMMA

INC IX — INCREMENTA REGISTRO IX — implicito ‖ DD23 (2 byte) — Z80

Al contenuto del registro IX viene sommato 1

EFFETTI SUL RCP

	S	Z	H		V	N	C
MSB							LSB

NESSUN EFFETTO

Esempio:

LOCAZIONE	CODICE MACCHINA	LINGUAGGIO ASSEMBLY
F391	DD23	INC IX

		PRIMA		DOPO
PC	lo	FF		00
	hi	E7		E8

DD	$F391
23	

PC = Program Counter (Contatore di Programma)

MEMORIA PROGRAMMA

LIZ DIXON

La Digital Research

Il sistema operativo per microcomputer CP/M, prodotto da questa casa, ha riscosso un fenomenale successo

Gary Kildall, uno dei creatori del microprocessore 8080 della Intel, sviluppò la prima versione del sistema CP/M nel 1974, come supporto a un compilatore per il PL/M (il primo linguaggio ad alto livello prodotto dalla Intel). Nel 1975, egli aggiunse al sistema un programma di editor (ED), un assemblatore (ASM) ed un debugger (DDT), ma la Intel ne rifiutò la commercializzazione. Forse questo fu per Kildall un vero colpo di fortuna! Infatti associatosi con Dorothy McEwan, Kildall iniziò a pubblicare articoli su riviste specializzate ed a vendere personalmente il CP/M, che, a sua volta, contribuì grandemente al "boom" di queste riviste.

Il CP/M ha notevolmente ridotto uno dei maggiori problemi dei primi microcomputer: la compatibilità. I più diffusi computer del tempo (il PET, l'Apple ed il Tandy) adottavano sistemi operativi a dischi, incompatibili l'uno con l'altro; pertanto, le software house indipendenti erano costrette a produrre programmi adatti ad un solo apparecchio, tralasciando gli altri (salvo riscrivere completamente i programmi). Il CP/M provocò una vera e propria rivoluzione: grazie alla sua immediata popolarità fu adottato da quasi tutti i produttori e divenne in breve tempo uno "standard". Molti fabbricanti di computer basati sul chip 8080 della Intel o sullo Z80 della Zilog, adottarono volentieri il CP/M per le sue semplici procedure di accesso a tutte le unità di ingresso/uscita. La disponibilità di software crebbe parallelamente al successo del sistema, inducendo un numero sempre maggiore di fabbricanti ad adottarlo.

Inizialmente, il Control Program for Microprocessor (Programma di Controllo per Microprocessore) venne distribuito, sotto licenza, a pochi e selezionati clienti. La sigla CP/M in un primo momento significava "Programma di Controllo Monitor", ma questa definizione venne presto cambiata. Già nel 1976, Kildall era sommerso dalle ordinazioni ricevute. Si dimise quindi dalla carica di ordinario di scienza dei computer (presso il Collegio Navale di Monterey) e fondò la Digital Research a Pacific Grove, in California.

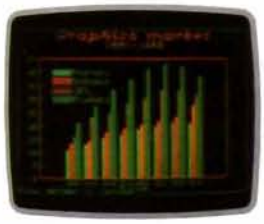
Mentre il CP/M continuava a mietere successi, la Digital Research si dedicò ai sistemi multi-utente e produsse l'MP/M. Questo sistema avrebbe dovuto essere del tutto compatibile con il CP/M, ma le prime versioni non riscossero il successo ottenuto dal suo predecessore. Né la ripartizione dell'area utente né le altre procedure di sistema si sviluppavano in modo lineare e, in alcuni casi, la gestione dei file differiva da quella del CP/M. L'enorme espansione del mercato dei microprocessori, inoltre, provocava una sensibile riduzione nel loro prezzo e l'interesse per i sistemi multi-utente calò decisamente. La stessa sorte è toccata, così, anche alla seconda versione dell'MP/M.

Nel 1981, la Digital Research ha ottenuto nuovi finanziamenti, grazie a compartecipazioni di capitale di altre compagnie, trasformandosi in una vera e propria multinazionale con forti rappresentanze in Europa (ha aperto infatti uffici in Gran Bretagna, Germania e Francia).



Grafica LOGO

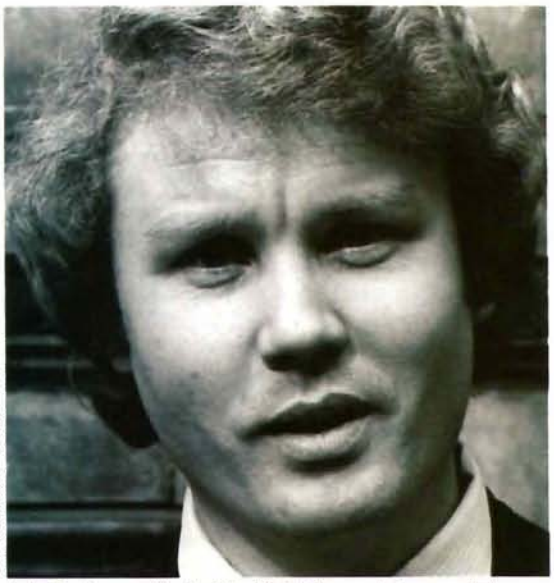
La Digital Research ha lavorato anche sui linguaggi ed ha sviluppato una propria versione del LOGO. Come per tutti i LOGO di buona fattura, la grafica è uno dei suoi punti di forza



IAN McKINNELL

Grafica gestionale

Il GSX è un pionieristico pacchetto software le cui applicazioni grafiche possono essere utilizzate su diversi tipi di apparecchi. Quello mostrato in figura è il pacchetto di grafica gestionale



per concessione della DIGITAL RESEARCH INC.

John Rowley, presidente della Digital Research Incorporated



Gary Kildall



per concessione della DIGITAL RESEARCH INC.

La sede della Digital Research Incorporated nel Massachusetts, USA.

Nello stesso periodo, la Digital Research combatteva una strenua battaglia per assicurarsi un contratto con la IBM: il contratto riguardava lo sviluppo di un sistema operativo per il Personal Computer di quella Casa. La contesa terminò a favore della Microsoft, ma la Digital Research non si dette per vinta. Da allora, ha prodotto due nuove versioni del CP/M: la prima adatta ai microprocessori 8088/8086 e molto simile all'MS-DOS, la seconda, chiamata Concurrent CP/M, orientata alla multiprogrammazione. Con questa versione, infatti, è possibile far girare simultaneamente programmi diversi. L'operatore può quindi eseguire tre diverse operazioni contemporanee, ad esempio: usare una lavagna elettronica, stampare dei tabulati ed inviare messaggi con la «posta elettronica», saltando a piacere dall'una all'altra funzione. Inoltre, alcune versioni del Concurrent CP/M permettono di visualizzare assieme, mediante "finestre", un'intera immagine ed un suo particolare. Nuove versioni di questo sistema sono in grado di eseguire, senza modifiche, la maggior parte del sofware scritto per il PC IBM.

Una decisione strategica, presa non soltanto dalla Digital Research, ma anche da molte altre case di software, è quella di scrivere tutti i nuovi prodotti software usando il "C", un linguaggio di programmazione particolarmente interessante per la sua "trasportabilità". I programmi scritti in C possono essere usati su microprocessori diversi, operando semplicemente una ricompilazione, anche se molti ritengono ingombrante il prodotto finale codificato. Alcuni programmatori, infatti, preferiscono ancora lavorare in Assembly su ciascun processore. Tuttavia, poiché il noto ed apprezzato sistema operativo UNIX è stato scritto in C, l'attrazione verso questo linguaggio sembra essere irreversibile.

La Digital Research ha giustamente intuito che la vera trasportabilità si ottiene solamente con linguaggi ad alto livello ed attualmente offre una discreta varietà di linguaggi per una vasta gamma di micro.

Per la fascia di prodotti più economici, la Digital Research ha organizzato una Divisione Vendite che commercializzerà il Personal BASIC, il Personal CP/M ed una propria versione del LOGO.

Il Personal CP/M, come il CP/M-86, è progettato per essere memorizzato in ROM e, grazie ad un accordo con la Zilog, sarà presto disponibile su un chip Z80. La Digital Research descrive questo prodotto come "microware" (per sottolineare la combinazione tra microprocessore e software) ed è sicura di poter prolungare la vita di un gran numero di programmi per CP/M, che stanno ormai invecchiando, "svendendoli" ai possessori di home computer.

Altri due prodotti di sicuro avvenire sono il VIC ed il GSX. Il primo è un economico programma visivo, che facilita la produzione di "videate" uniformi nei programmi, indipendentemente dal pacchetto applicativo utilizzato. L'idea è buona, in quanto programmi di varia natura utilizzano spesso gli stessi dati e questi ultimi possono venir così trasferiti da un'applicazione all'altra. Il VIP, quindi, assomiglia concettualmente al sistema adottato per il Lisa ed il Macintosh della Apple, ma occupa meno memoria. Esso potrà essere utilizzato su qualunque computer con più di 50 Kbyte di RAM e con dischetti da almeno 150 Kbyte.

In campo grafico, il GSX permetterà sviluppi simili a quelli ottenuti con il CP/M per i dischetti. Adotta infatti una serie standard di istruzioni grafiche utilizzabili su apparecchiature di diverso tipo: potrà quindi essere usato senza alcuna modifica su schermi sia in bianco e nero sia a colori, su una stampante ad aghi e su un plotter.