Spediz. in abbonamento postale GR. II/70 L. 2.000
(...)

45 CORSO PRATICO COL COMPUTER

421974

è una iniziativa
FABBRI EDITORI
in collaborazione con
BANCO DI ROMA
e **OLIVETTI**

F4 F5 F6 F7 F8

BATTERY LOW

diretto da GIANNI DEGLI ANTONI

Direttore dell'opera
GIANNI DEGLI ANTONI

Comitato Scientifico
GIANNI DEGLI ANTONI
Docente di Teoria dell'informazione, Direttore dell'Istituto di Cibernetica dell'Università degli Studi di Milano

UMBERTO ECO
Ordinario di Semiotica presso l'Università di Bologna

MARIO ITALIANI
Ordinario di Teoria e Applicazione delle Macchine Calcolatrici presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

MARCO MAIOCCHI
Professore Incaricato di Teoria e Applicazione delle Macchine Calcolatrici presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

DANIELE MARINI
Ricercatore universitario presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

Curatori di rubriche
MARCO ANELLI, DIEGO BIASI, ANDREA GRANELLI, ALDO GRASSO, MARCO MAIOCCHI, DANIELE MARINI, GIANCARLO MAURI, CLAUDIO PARMELLI

Testi
ADRIANO DE LUCA, VIRGINIO SALA, Eidos (BRUNO MOTTA, CARMINE STRAGAPEDE), Etnoteam (ADRIANA BICEGO)

Tavole
Logical Studio Communication
Il Corso di Programmazione e BASIC è stato realizzato da Etnoteam S.p.A., Milano
Computergrafica è stato realizzato da Eidos, S.c.r.l., Milano
Usare il Computer è stato realizzato in collaborazione con PARSEC S.N.C. - Milano

Direttore Editoriale
ORSOLA FENGHI

Redazione
CARLA VERGANI
LOGICAL STUDIO COMMUNICATION

Art Director
CESARE BARONI

Impaginazione
BRUNO DE CHECCHI
PAOLA ROZZA

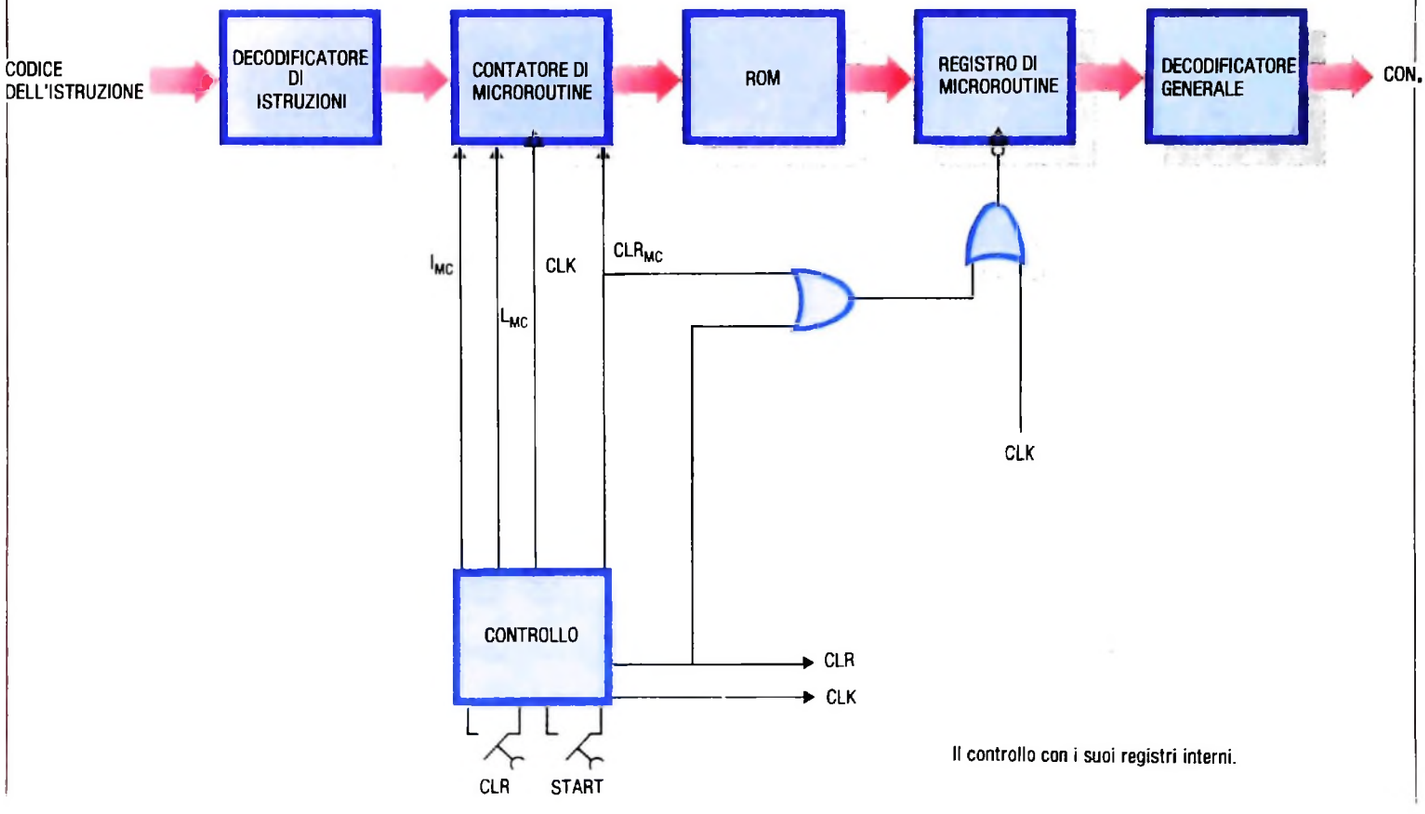
Programmazione Editoriale
ROSANNA ZERBARINI
GIOVANNA BREGGÈ

Segretarie di Redazione
RENATA FRIGOLI
LUCIA MONTANARI

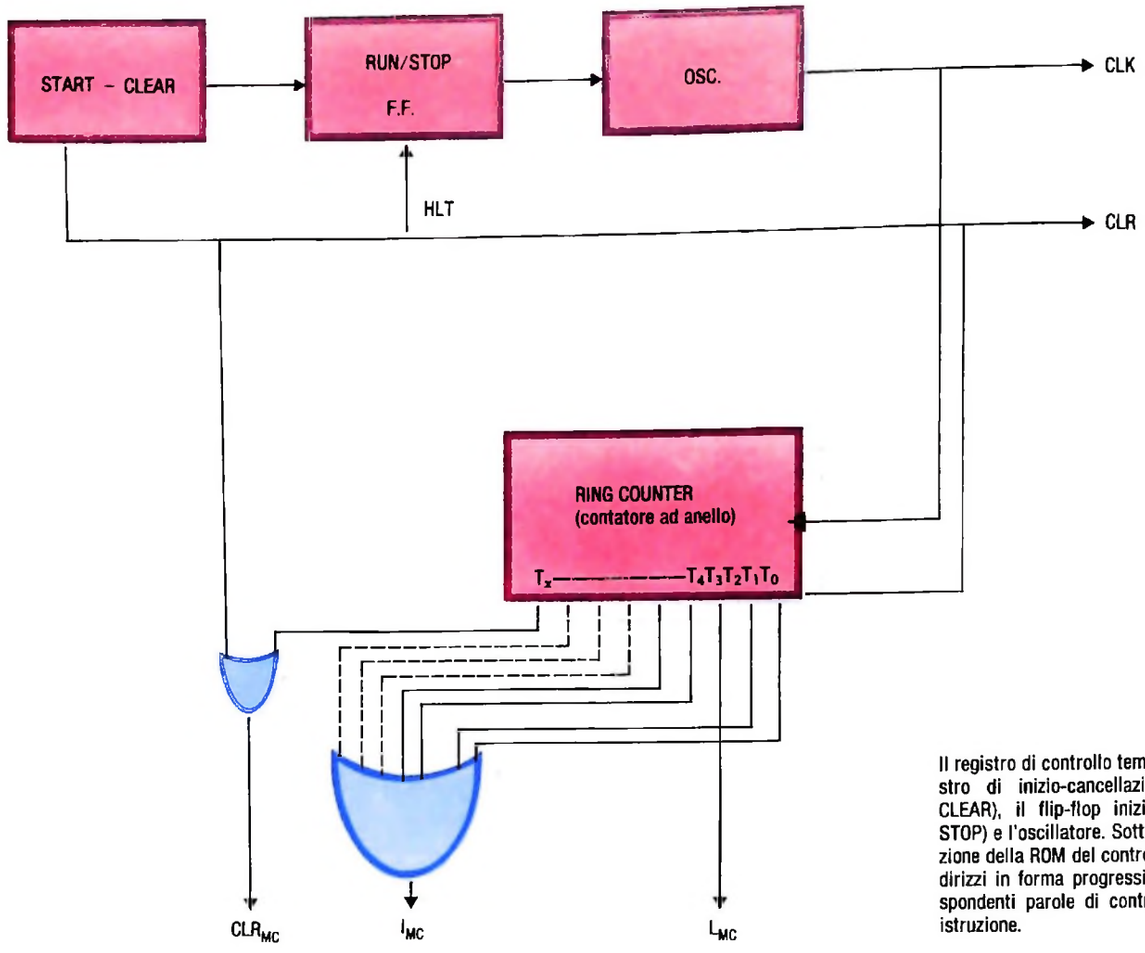
CONTROLLO (II)

Analizziamo la temporizzazione degli eventi logici all'interno di un microprocessore.

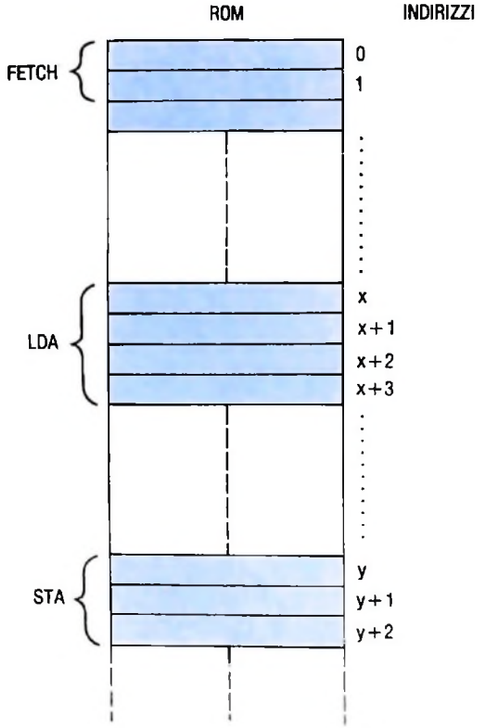
Riprendiamo l'analisi del Controllo (figura in basso) esaminando la temporizzazione, la parte più critica del sistema.
Partiamo dall'analisi interna del blocco Controllo (figura in alto a pagina seguente). Anche qui vediamo che esiste una catena formata dal sistema di START-CLEAR (Inizia-Cancella), dal flip-flop RUN/STOP (funziona-arresto) e dall'oscillatore OSC. In basso c'è il RING COUNTER (contatore ad anello) che naturalmente emetterà una serie di CLOCK esattamente uguali alle fasi dell'istruzione più lunga.
Vediamo come funziona tutto il sistema incominciando dall'analisi del contenuto della ROM.
Nella ROM del controllo (figura in basso a pagina seguente) sono immagazzinate le varie parole di controllo in forma lineare: infatti, come possiamo vedere dalla figura, ci sono, da una parte gli indirizzi (in ordine progressivo) in corrispondenza con le varie parole di controllo, dall'altra, il tipo di istruzione.
L'indirizzo della ROM viene fornito dal Registro Contatore di microroutine che a sua volta è controllato dal Controllo.
Dal Controllo (figura in alto a pagina seguente) escono 5 comandi, tre dei quali vanno al contatore di microroutine. Essi sono:
— CLR_{MC} che può essere attivato per due motivi: uno perché si è premuto il tasto di cancellazione, secondo perché il RING COUNTER è arrivato al termine. Quando il CLR_{MC} è attivo il Registro Contatore di microroutine si azzera e punta quindi all'inizio della ROM cioè al FETCH.
A ogni fase un segnale I_{MC} provvede automaticamente ad incrementare l'indirizzo.
— I_{MC} nella fase T_2 carica internamente l'indirizzo di inizio



Il controllo con i suoi registri interni.



Il registro di controllo tempi, con il registro di inizio-cancellazione (START-CLEAR), il flip-flop inizio-stop (RUN-STOP) e l'oscillatore. Sotto, l'organizzazione della ROM del controllo con gli indirizzi in forma progressiva e le corrispondenti parole di controllo per ogni istruzione.



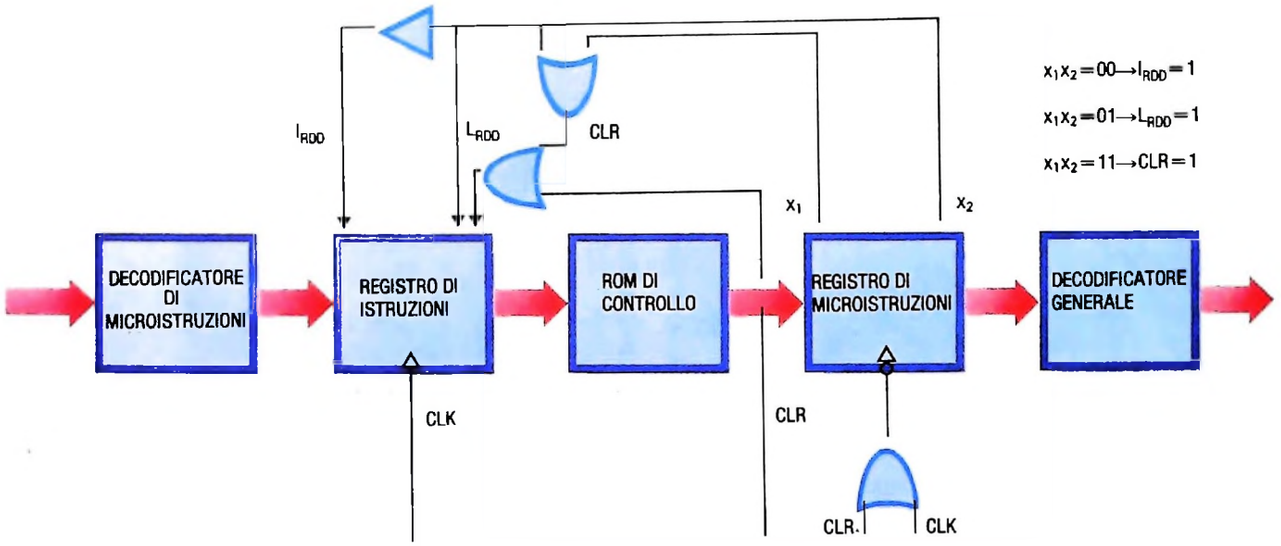
dell'istruzione nella ROM. A ogni CLOCK l'indirizzo s'incrementa fino ad esaurire l'istruzione.

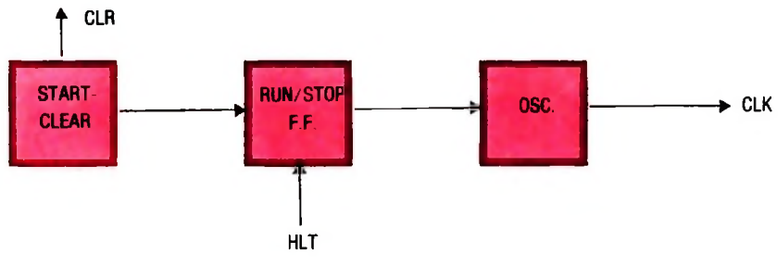
— Gli ultimi due comandi CLK e CLR (orologio e cancellazione) vanno, invece, al sistema.

Questo Controllo ha il pregio di essere abbastanza semplice, però ha il grande difetto di essere molto lento. La lentezza è dovuta alle "fasi vuote" causate dalle istruzioni composte da una quantità di fasi minore del numero massimo del RING COUNTER (contatore ad anello).

L'indirizzo della ROM corrispondente all'istruzione è dato dal registro di entrata Decodificatore di istruzione. Questo registro ha la particolare funzione di trasformare il codice dell'istruzione in "indirizzo" della ROM. Questa trasformazione è necessaria, ed è dovuta al fatto che, mentre l'assegnazione del codice a un'istruzione è vincolata a tutta una serie di restrizioni, come abbiamo potuto vedere, le parole di controllo invece sono immagazzinate in memoria in ordine progressivo.

Analizziamo adesso il Controllo nel tempo. Inizialmente quando si parte con un FETCH il Registro Contatore di microroutine è stato azzerato con il comando CLR_{MC} per cui il valore zero alla sua uscita punta alla prima posizione della memoria della ROM.

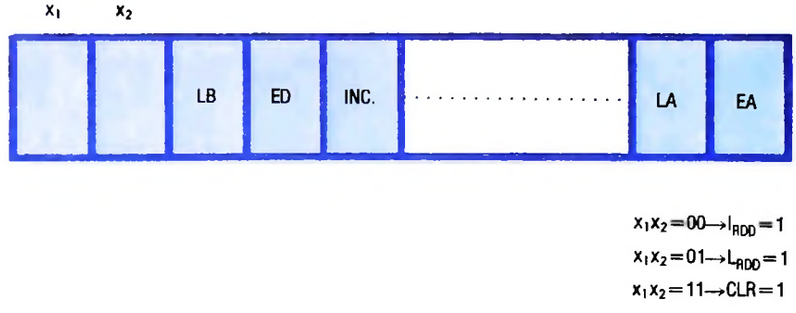




La struttura complessiva del controllo. Sopra, l'insieme dei registri per la trasformazione delle istruzioni in parole di controllo, il circuito di retroalimentazione e controllo automatico della gestione dell'istruzione. Sotto, il registro di START-CLEAR, il flip-flop RUN-STOP e l'oscillatore.

La parola di controllo prelevata da questo indirizzo si presenta all'entrata del Registro di microroutine e viene immagazzinata nello stesso registro alla transizione negativa del comando CLR_{MC}.
La parola di controllo, ormai fissa nel Registro di microroutine, entra nel Decodificatore generale e infine la parola completa di controllo si presenta a tutto il sistema.

Nella fase che segue si attiva I_{MC} che incrementa il Contatore di microroutine e, sempre nella stessa fase, alla transizione negativa del CLK la nuova parola si immagazzina nel Registro di microroutine.
Vediamo ora un sistema che non presenta lo stesso inconveniente del precedente, cioè la lentezza (figura in alto).
La prima cosa che notiamo è che è sparito il RING COUN-



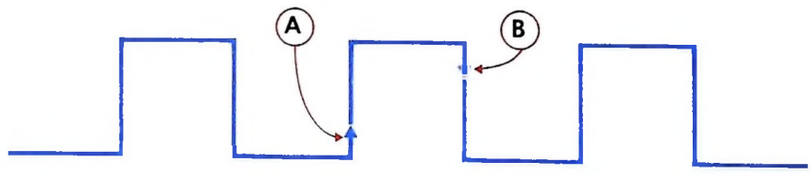
La parola di controllo con l'aggiunta dei due bit di controllo X_1 e X_2

$x_1x_2=00 \rightarrow I_{RDD}=1$ (incremento del Registro di microroutine)
$x_1x_2=01 \rightarrow L_{RDD}=1$ (carica il Registro di microroutine con il contenuto del Registro di istruzioni)
$x_1x_2=11 \rightarrow CLR=1$ (azzera il contenuto del Registro di microroutine)

Forma in cui sono riservati per il controllo i due bit X_2 e X_1 della nuova parola di controllo.

TER e che l'unico registro rimasto è il doppio bottone di inizio e cancellazione.
La nuova forma di realizzazione del controllo consiste nel creare una parola di controllo che oltre a svolgere la funzione normale che già conosciamo, possa anche avere degli attributi in più che permettono di eliminare il RING COUNTER.
Analizziamo la parola di controllo nuova (figura in basso a pagina precedente) e vediamo che esistono due bit in più (X_2 e X_1) che sono riservati per il controllo nella forma della figura in alto di questa pagina.
Quando un'istruzione entra nel registro Decodificatore di istruzioni questo trasforma il codice dell'istruzione nell'indirizzo corrispondente nella ROM. Alla translazione positiva del CLK l'indirizzo viene immagazzinato nel Contatore di microroutine. La ROM a sua volta presenta la prima parola di controllo modificata per l'aggiunta di due elementi nuovi X_1 X_2. Alla translazione negativa del CLK (figura in basso) la parola di controllo viene immagazzinata nel Registro di microroutine.
All'uscita di questo registro i segnali X_1 X_2, attraverso un piccolo circuito combinatorio, ritornano al Registro di microroutine trasformati nei nuovi segnali I_{RDD}, L_{RDD} e CLR.

Prendiamo come esempio il FETCH di una istruzione che, come sappiamo, è composto da due fasi. Nella prima fase la parola di controllo avrà, oltre ai comandi necessari, i due comandi X_1 $\bar{X}_2$ = 00 che nel circuito combinatorio attivano solo I_{RDD} = 1, mentre l'indirizzo nel registro s'incrementa e punta alla seconda fase. Nella fase successiva viene completato il FETCH e nel Decodificatore di microistruzioni è presente il codice dell'istruzione da eseguire. I segnali X_1 X_2 = 01 attivano L_{RDD} = 1 per cui al CLK il nuovo indirizzo viene caricato nel Registro di indirizzo.
Le fasi di incremento vengono ripetute fin quando non viene esaurita l'istruzione e si ha X_1 X_2 = 11. Questi segnali attivano CLR = 1 per cui si riparte con un nuovo FETCH fino al termine del programma attraverso l'istruzione HLT che blocca il sistema.
Come ultima osservazione sulla UAMICRO III notiamo che anche in questa versione l'avviamento del micro è fatto attraverso la sequenza dei due pulsanti CLR (per cancellare e iniziare il primo FETCH) e RUN (per iniziare tutto il processo). Tale scelta, che si diversifica da quanto avviene nei normali microprocessori in commercio, è stata fatta per rendere più chiara la comprensione del sistema di funzionamento.



(A) = l'indirizzo dell'istruzione viene caricato nel Contatore di microroutine

(B) = la parola di controllo viene caricata nel Registro di microroutine

Come viene effettuato, nel tempo, il caricamento dell'indirizzo nel contatore di microroutine (sopra) e della parola di controllo nel registro di microroutine (sotto).

MACCHINE A STATI FINITI COME RICONOSCITORI

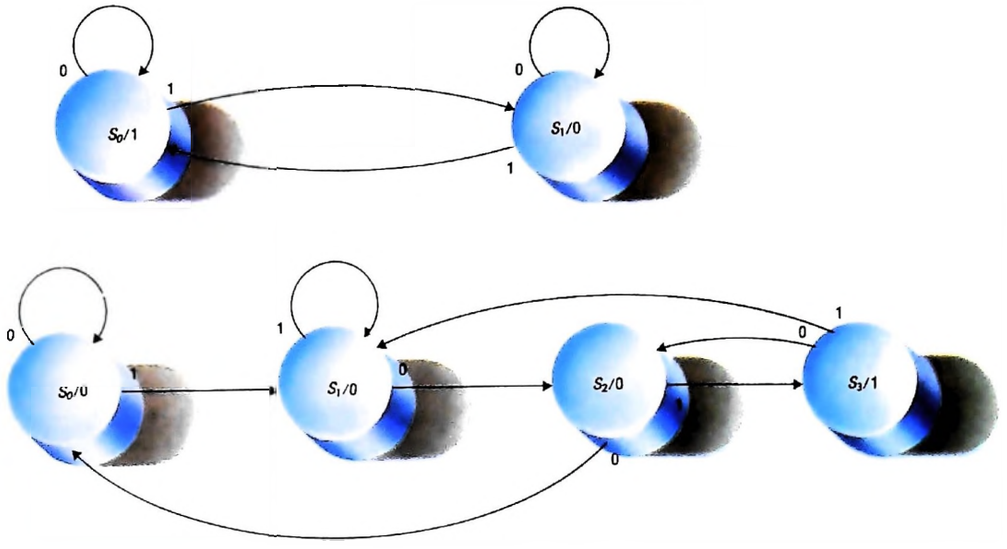
Una macchina a stati finiti in un certo senso "ricorda" ciò che riceve in ingresso: si possono quindi costruire macchine a stati finiti che riconoscono particolari configurazioni in ingresso.

Prendiamo una macchina M semplicissima, con due soli stati: S_0, lo stato iniziale, e S_1. La macchina ha un funzionamento elementare: se riceve in ingresso uno 0 rimane nello stato in cui si trova, qualunque esso sia, mentre se riceve in ingresso un 1 effettua una transizione di stato. In S_0 stampa in uscita un 1, in S_1 stampa in uscita uno 0. A che cosa può servire questa macchina? Se osservate bene il grafo di transizione di M, riportato qui in basso, noterete che stampa un 1 in uscita solo quando ha ricevuto in ingresso 0, 2, 4, 6 ... volte un 1; stampa uno 0 quando ha ricevuto in ingresso 1, 3, 5, 7 ... volte 1. Basta dunque osservare l'ultima cifra che M stampa, per sapere se la successione di caratteri in ingresso conteneva un numero pari o dispari di 1. M, dunque, potrebbe essere utilizzata per una forma molto semplice di controllo "di parità".

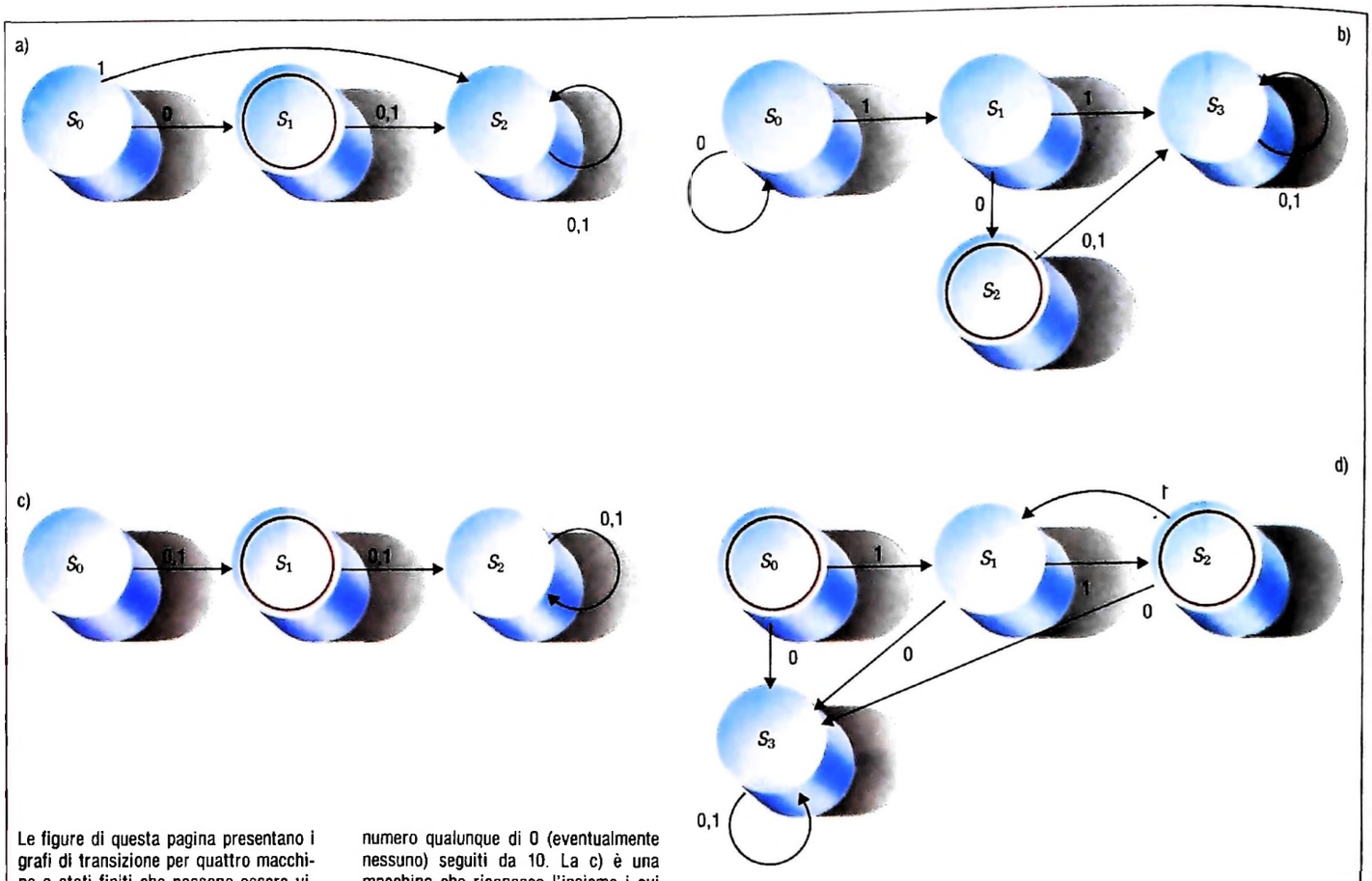
La macchina M presenta un caso elementare di macchina a stati finiti utilizzabile come riconoscitore: M è in grado di tener conto dell'andamento del suo input e può essere utilizzata per verificare se l'input gode o meno di una certa proprietà. L'altro grafo di transizione riportato in basso presenta un caso un poco più complesso: la macchina $M1$ si ferma stampando un 1 solo se la successione di caratteri che riceve in ingresso termina con una terna 101; in tutti gli altri casi si ferma in uno stato che ha come uscita uno 0.

Che cosa può riconoscere una macchina a stati finiti?

Questi ultimi esempi ci hanno portati vicino al mondo che ci interessa, quello dei calcolatori: ormai la scelta di 0 e 1 dovrebbe suonare ovvia. Sappiamo già che, comunque, con opportune stringhe formate da questi due unici simboli siamo in grado di esprimere tutto quello che può servirci; ed è questo il repertorio che può comprendere un calcolatore digitale. Per semplificarci ulteriormente la strada, nei prossimi dia-



Una macchina per il controllo della parità, in alto quando ha ricevuto un numero pari di 1 la sua uscita è 1, mentre è 0 quando ha ricevuto un numero dispari di 1. Il grafo di transizione in basso rappresenta invece una macchina a stati finiti che dà in uscita 1 solo quando riceve in ingresso la stringa 101 e in ogni altro caso dà in uscita 0.



Le figure di questa pagina presentano i grafi di transizione per quattro macchine a stati finiti che possono essere viste come riconoscitori. La a) è una macchina che riconosce l'insieme formato da un solo 0. La b) riconosce l'insieme i cui elementi sono stringhe di un numero qualunque di 0 (eventualmente nessuno) seguiti da 10. La c) è una macchina che riconosce l'insieme i cui elementi sono uno 0 singolo e un 1 singolo. La d) riconosce l'insieme i cui elementi sono formati da un qualunque numero pari (0 compreso) di 1.

grammi di transizione useremo una convenzione in più: non indicheremo più l'uscita, nell'etichetta di ogni nodo, ma ci limiteremo a rappresentare con due cerchi concentrici i nodi che hanno uscita 1; gli altri nodi si sottintende abbiano uscita 0. Escludiamo ogni altra possibilità. Senza bisogno di esplicitarlo più, d'ora in poi considereremo sempre S_0 come lo stato iniziale. Chiameremo, inoltre, "stato finale" un nodo che abbia uscita 1.

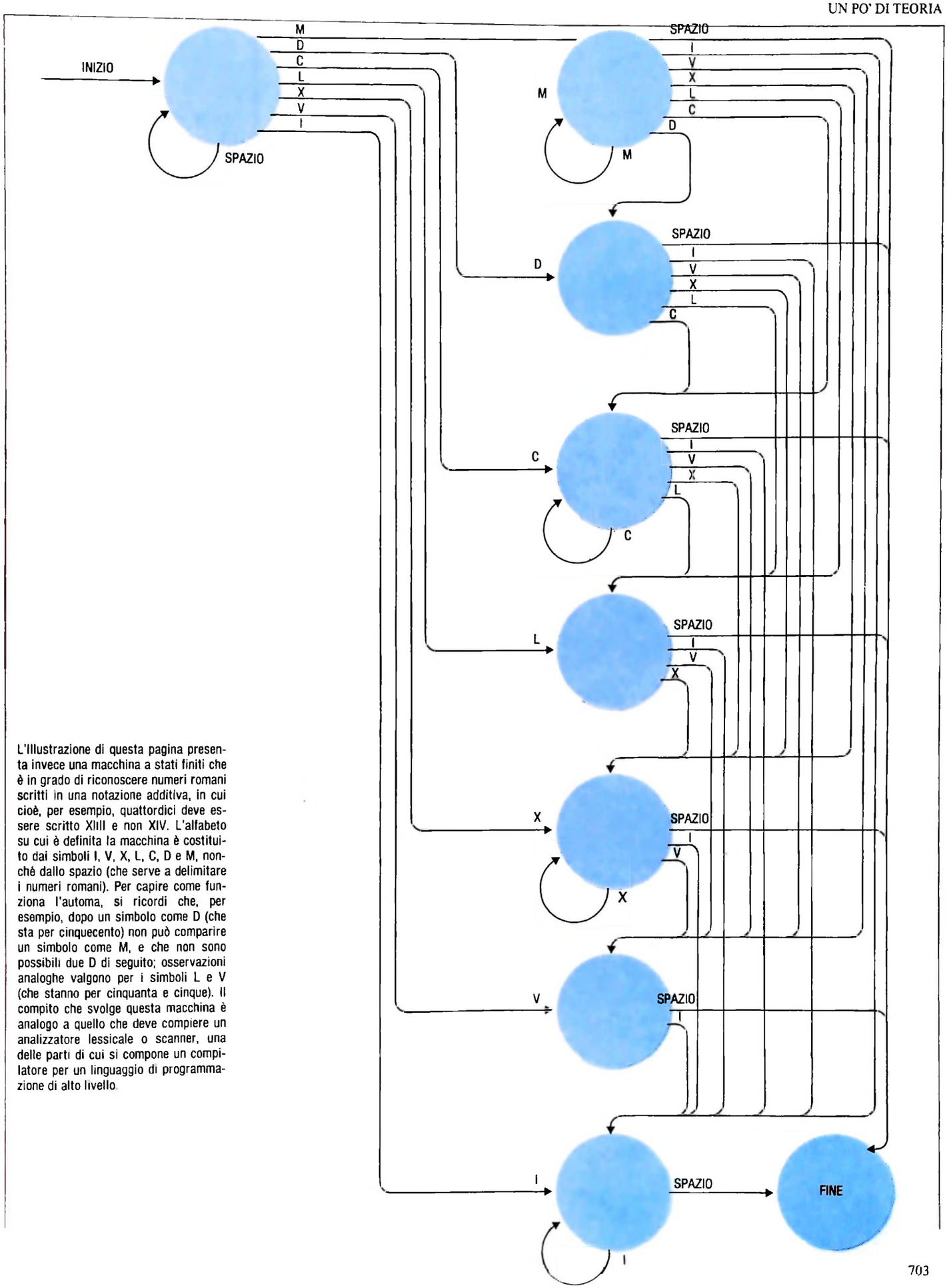
I diagrammi di transizione riportati in questa pagina si attengono già a questa convenzione. Rappresentano quattro macchine in grado di riconoscere particolari configurazioni di ingresso. Provate a cercare di capire quali configurazioni possa riconoscere ciascuna di quelle macchine, prima di leggerne la spiegazione nella didascalia.

Una volta scoperto che gli automi finiti possono essere utilizzati come riconoscitori di configurazioni, viene abbastanza spontaneo domandarsi se è possibile caratterizzare quali tipi di configurazioni possano essere riconosciute da un automa finito.

Qualunque tipo di configurazione? Solo qualcuna? Oppure esiste una legge generale che ci permetta di stabilire chiaramente il campo d'azione degli automi finiti?

In effetti gli automi finiti non sono in grado di riconoscere qualsiasi configurazione, ma è possibile caratterizzare la natura delle configurazioni riconoscibili. Una configurazione è riconoscibile da un automa finito se è un'espressione regolare e, viceversa, qualunque configurazione venga riconosciuta da un automa finito è una espressione regolare. La risposta sarebbe perfetta se avessimo già detto che cos'è un'espressione regolare, ma per questo dobbiamo tornare un momento ancora alla teoria degli insiemi.

Consideriamo un insieme *I* di simboli, un *alfabeto*, i cui simboli possiamo chiamare per comodità *lettere*. Una *parola* su *I* è una successione finita di lettere: se l'alfabeto fosse costituito dalla sola lettera a, le parole possibili sarebbero solo a, aa, aaa, aaaa, e via dicendo. Fra le parole possibili si considera sempre, per qualunque alfabeto, anche la *parola vuota* (la parola che non è costituita di alcuna lettera), indicata con la lettera greca lambda maiuscola, Λ. (Attenzione a non confondere la parola vuota con l'insieme vuoto: l'insieme vuoto è un insieme che non contiene parole, mentre Λ è una parola, vuota sì, ma pur sempre parola, e l'insieme che contiene come unico elemento Λ non è un insieme vuoto, ma un insieme con un elemento, la parola vuota.) Oltre a due nozioni che abbiamo già visto, quella di unione e quella di prodotto fra insiemi, ce ne serve una terza: quella di *chiusura* (o *stellatura*)



L'Illustrazione di questa pagina presenta invece una macchina a stati finiti che è in grado di riconoscere numeri romani scritti in una notazione additiva, in cui cioè, per esempio, quattordici deve essere scritto XIIII e non XIV. L'alfabeto su cui è definita la macchina è costituito dai simboli I, V, X, L, C, D e M, nonché dallo spazio (che serve a delimitare i numeri romani). Per capire come funziona l'automa, si ricordi che, per esempio, dopo un simbolo come D (che sta per cinquecento) non può comparire un simbolo come M, e che non sono possibili due D di seguito; osservazioni analoghe valgono per i simboli L e V (che stanno per cinquanta e cinque). Il compito che svolge questa macchina è analogo a quello che deve compiere un analizzatore lessicale o scanner, una delle parti di cui si compone un compilatore per un linguaggio di programmazione di alto livello.

dell'alfabeto *I*, indicata con *I**, che è l'insieme costituito dalla parola vuota e da tutte le parole che si possono formare concatenando un numero finito di parole di *I*. Con queste nozioni possiamo definire la classe degli *insiemi regolari* su *I* mediante una definizione di tipo ricorsivo:

a) ogni insieme finito di parole su *I* è un insieme regolare;
b) se *A* e *B* sono insiemi regolari su *I*, sono insiemi regolari su *I* anche la loro unione $A \cup B$ e il loro prodotto *AB*;
c) se *A* è un insieme regolare su *I*, lo è anche la sua chiusura *I**.

Gli insiemi regolari, possono essere rappresentati mediante *espressioni regolari*, formate a partire dalle lettere dell'alfabeto di partenza. Ancora una volta abbiamo una definizione ricorsiva:

a) l'insieme vuoto e la parola vuota sono espressioni regolari su *I*;
b) ogni lettera dell'alfabeto *I* è un'espressione regolare su *I*;
c) se *a* e *b* sono espressioni regolari su *I*, allora sono espressioni regolari su *I* *ab*, $a + b$ e a^*

ab è la semplice concatenazione di *a* e *b*; la scrittura $(a + b)$ indica un'espressione formata da una *a*, oppure da una *b* oppure da una *a* e una *b*; la stellatura di un'espressione è la concatenazione di quell'espressione con se stessa un numero finito di volte.
Le espressioni regolari possono rappresentare gli insiemi regolari: i simboli di insieme vuoto e di parola vuota rappresentano gli insiemi corrispondenti; le singole lettere dell'alfabeto rappresentano ciascuna l'insieme che contiene come unico elemento quella lettera; se *a* e *b* sono espressioni che rappresentano gli insiemi *A* e *B* rispettivamente, l'espressione ottenuta per concatenazione di *a* e *b* (*ab*) rappresenta l'insieme prodotto *A* x *B*; l'espressione disgiunzione di *a* e *b* ($a + b$) rappresenta l'insieme unione $A \cup B$; la stellatura di *a* rappresenta la chiusura A^*.

Espressioni regolari

La definizione astratta del concetto di espressione regolare lascia forse un po' perplessi. Vediamo di concretizzarla con qualche esempio chiarificatore. Il nostro alfabeto I conterrà, in questi esempi, le sole due lettere 1 e 0. Sappiamo, dalla definizione, che l'insieme vuoto 0, la parola vuota Λ, 1 e 0 sono espressioni regolari. Anche 10*, per esempio, è un'espressione regolare. Ma che cosa rappresenta? Tutte le parole su I che incominciano con un 1 e proseguono con un numero finito (eventualmente nullo) di 0. L'espressione regolare 10*, dunque, rappresenta l'insieme delle parole su I della forma:

1 10 100 1000 10000 100000 ...

e via dicendo. Quando nell'espressione regolare compaiono espressioni separate dal segno +, significa che possono comparire l'una o l'altra o ambedue.
L'espressione regolare 1 + 0, per esempio, rappresenta l'insieme che ha come elementi le parole:

1, 0, 10

e nulla più. È, in effetti, l'insieme unione degli insiemi di parole rappresentati da 1 e 0 (che non sono altro che i singoletti di 1 e 0, cioè gli insiemi che hanno come unico elemento 1 e 0, rispettivamente).

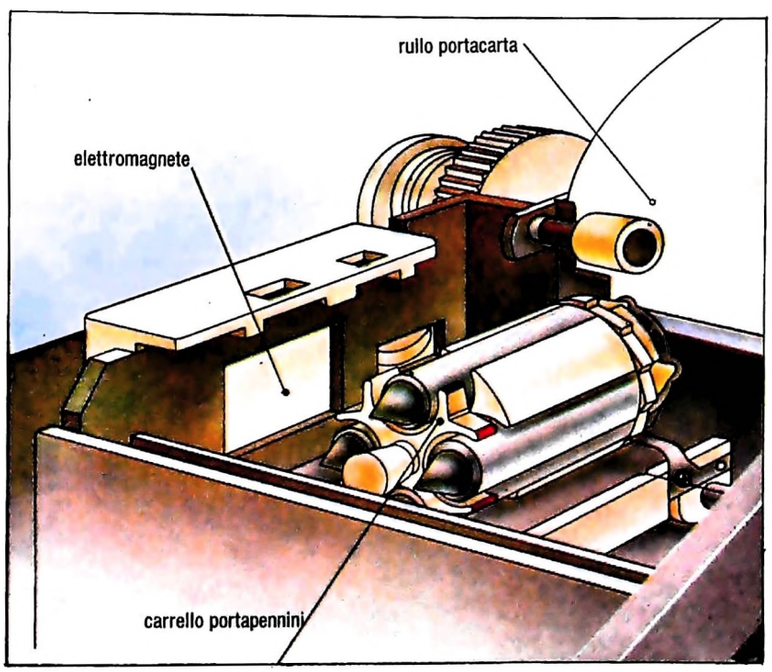
Il teorema di Kleene e i limiti delle macchine a stati finiti

Una macchina a stati finiti, dunque, può riconoscere solo espressioni regolari sul suo alfabeto di input, e vale anche il viceversa: per qualunque espressione regolare su un alfabeto, esiste una macchina a stati finiti che ha quell'alfabeto come alfabeto di input e riconosce esattamente quell'espressione regolare. Questa affermazione è dimostrabile: costituisce, in effetti l'enunciato del *teorema di Kleene* (dimostrato negli anni Cinquanta dal logico americano Stephen C. Kleene).
Il teorema caratterizza formalmente le capacità di riconoscimento di una macchina a stati finiti: che cosa rimane escluso? Dagli esempi riportati in queste pagine si può notare che una macchina a stati finiti può riconoscere molte configurazioni interessanti, ma non è difficile trovare qualche classe di configurazioni che non possa essere catturata da un'espressione regolare: per fare solo un esempio, l'insieme delle parole della forma $a^n b^n$ (cioè formate da *n* lettere *a* seguite da *n* lettere *b*, per *n* maggiore o uguale a zero) non può essere riconosciuto da alcuna macchina a stati finiti. Non è un insieme regolare, e non può dunque essere rappresentato da una espressione regolare.
Anche quest'ultima affermazione può essere dimostrata rigorosamente, ma non è importante farlo in questa sede. Si può comunque verificarla abbastanza intuitivamente, perché basta riflettere sul fatto che parole di questo tipo non possono essere costruite con le operazioni permesse dalla definizione di insieme regolare. Il problema, ovviamente, sta tutto nella limitata capacità di memoria di una macchina a stati finiti, che non ha modo di conservare traccia della storia passata dei suoi input e di effettuare confronti estesi, come quelli necessari per identificare l'uguale numero di 1 e 0 in una espressione della forma $1^n 0^n$.
Per questo gli automi finiti debbono essere estesi con qualche dispositivo di memoria ulteriore.

Lezione 44

I colori del microplotter

Nelle precedenti lezioni dedicate al microplotter abbiamo imparato a realizzare lettere di differenti grandezze e a posizionare il pennino sulla carta, in modo da poter scrivere su differenti righe.
Impareremo ora a comandare la rotazione dei pennini del microplotter, in modo da poter cambiare il colore della scrittura.
Il portapennini è costituito da un supporto con 4 spazi ciascuno dei quali può raccogliere un pennino. I quattro spazi portano una piccola strisciolina colorata alla base, in modo che sia possibile associare a ciascuno di essi il pennino del corrispondente colore. Ciò è importante perché, evidentemente, il microplotter non può riconoscere il colore dei pennini, e semplicemente si aspetta che ciascun colore si trovi in una specifica posizione.



Il comando da inviare per selezionare il cambio del pennino è C, seguito da un numero che identifica il colore; come al solito tale comando sarà inviato mediante l'istruzione LPRINT.
Se i pennini colorati sono stati messi a posto correttamente, l'associazione tra numeri e colori sarà la seguente:

0 NERO
1 BLU
2 VERDE
3 ROSSO

Così, l'istruzione provocherà la rotazione del portapennini in modo che la punta

```
10 LPRINT "C3"
```

scrivente sia quella con l'inchiostro rosso.
Supponiamo di voler alterare l'ultimo programma costruito, che scriveva parole con caratteri di differente grandezza, in modo che:

- il numero di parole stampate sia solo 10
- ogni parola venga scritta con un colore differente.

Allora dovremo inserire nel programma un'istruzione di cambio del colore, ancora una volta con la costruzione del comando mediante operazioni di concatenazione di stringhe.
Faremo quindi variare una variabile I da 0 a 9 per selezionare la dimensione dei caratteri, e per ogni valore di I sceglieremo il colore individuato da

I MOD 4

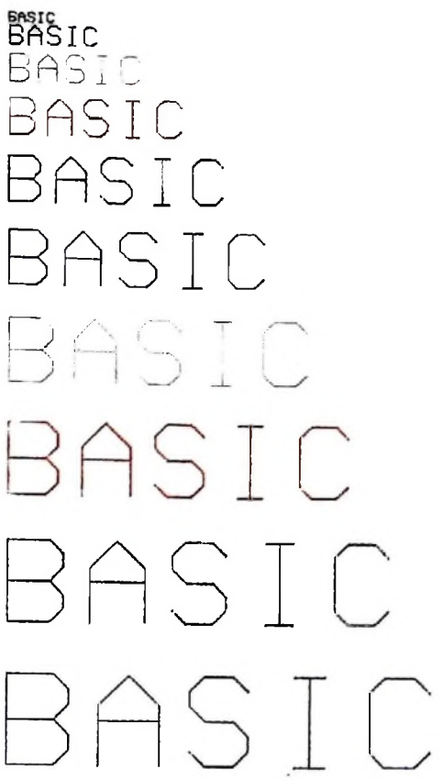
L'operatore MOD (modulo), infatti, fornisce il resto della divisione tra i due operandi, e, quando I vale 0 dà risultato 0, per I=1 dà 1, per I=2 dà 2, per I=3 dà 3, per I=4 torna a dare 0, per I=5 fornisce nuovamente 1, e così via, fornendo quindi un valore compreso tra 0 e 3, che corrisponde proprio all'indicazione dei pennini disponibili.
Come al solito, comporremo il comando mediante operazioni su stringhe:

```
35 LET C$="C"+STR$(I MOD 4)
```

che inseriremo nel programma precedente, per ottenere il seguente:

```
10 LPRINT CHR$(18) 'Graphic mode
30 FOR I=0 TO 9
35 LET C$="C"+STR$(I MOD 4)
38 LPRINT C$
40 REM Costruisce il comando
50 LET S$="S"+STR$(I)
60 LPRINT S$
65 LPRINT "PBASIC"
67 LET P$="R-"+STR$(30*(I+1))+",-"+STR$(
10*(I+1)+5)
68 LPRINT P$
70 NEXT I
```

L'esecuzione del programma ci fornirà il seguente risultato in cui le varie scritte di diverso formato si susseguono, come previsto, nei vari colori:



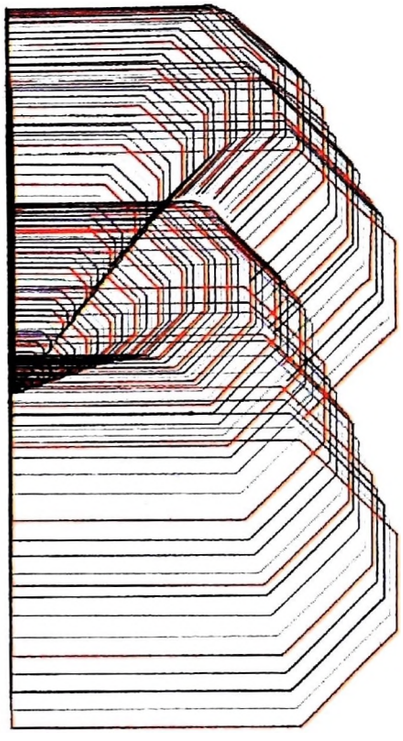
Ancora, un simpatico esempio di applicazione prevede lo stesso programma, questa volta applicato a tutti i 64 formati dei caratteri, allineando tutti i caratteri sul margine sinistro, ma senza cambiare linea.
Si tratta, in sostanza, dopo avere tracciato la stringa, di ritornare alla posizione di partenza con un solo movimento orizzontale, senza far scorrere la carta.
Per fare ciò, introduciamo un nuovo comando, M, che funziona esattamente come R, ma che chiede che l'indicazione delle coordinate del punto d'arrivo siano fornite in modo assoluto, indipendentemente dal punto in cui il pennino si trova nell'istante di esecuzione del comando. Non occorre quindi indicare, come avevamo fatto con il comando R, di quanto il pennino si deve spostare rispetto al punto in cui si trova quando il comando viene inviato, ma indicare direttamente il punto di spostamento, facendo riferimento al fatto che le condizioni in cui il pennino si trova all'inizio dell'esecuzione del programma sono considerate come coppia (0,0).
Così

```
68 LPRINT "M0,0"
```

riposiziona il pennino all'"origine" delle righe e delle colonne, indipendentemente dal punto in cui esso si trova.
Il seguente programma esegue l'operazione indicata scrivendo il solo carattere B in tutte le sue dimensioni.

```
10 LPRINT CHR$(18) 'Graphic mode
30 FOR I= 0 TO 63
35 LET C$="C"+STR$(I MOD 4)
38 LPRINT C$
40 REM Costruisce il comando
50 LET S$="S"+STR$(I)
60 LPRINT S$
65 LPRINT "PB"
68 LPRINT "M0,0"
70 NEXT I
```

Il risultato è il seguente:



Cosa abbiamo inparato

In questa lezione abbiamo visto:

- il comando C per cambiare il colore di stampa del microplotter;
- l'operatore MOD, che fornisce il resto della divisione tra due operandi;
- il comando M per spostare in MODO ASSOLUTO (cioè senza riferimento alla posizione corrente) il pennino sulla carta.

ESEMPI FAMOSI DI APPLICAZIONI IN LISP

Dopo aver esaminato le strutture del linguaggio, vediamone alcune applicazioni.

Finora abbiamo visto i rudimenti del linguaggio Lisp, e abbiamo esaminato le sue caratteristiche orientate all'elaborazione simbolica e al trattamento del linguaggio.
Ci sono esempi piuttosto famosi di applicazioni in Lisp su tale argomento e noi ne esamineremo qui un paio: una dovuta a J. Weizenbaum e una dovuta a T. Winograd.

Eliza, di Weizenbaum

Nel 1966 un ricercatore del Massachusetts Institute of Technology, Joseph Weizenbaum, costruì un programma in Lisp che simulava un dialogo con uno psicanalista. Il programma fu chiamato Eliza, dal nome della protagonista della commedia *Pigmalione* di George Bernard Shaw: nella commedia Eliza, una giovane fioraia, povera e incolta, veniva educata da un colto professore all'uso corretto del linguaggio e della pronuncia, con lo specifico obiettivo di mostrare che la capacità di usare il linguaggio è determinante per la valutazione delle persone, tanto da poter portare alla corte reale una persona come Eliza, purché in grado di parlare correttamente. Il programma ricalca la storia di Eliza, in quanto mostra una eccezionale potenza apparente legata solo al fatto di riuscire a cavarsela in un dialogo in linguaggio naturale. Il programma fu scritto in Lisp, ed è per questo che riteniamo interessante riportarne qui le caratteristiche.
Il programma innescò numerose diatribe che, allo stato attuale della psicanalisi, possono essere ancora considerate interessanti e che comunque presentano numerosi aspetti divertenti, sulla pura base del gustoso pettegolezzo. Una ver-

```
Da molto tempo non ho a che fare con mio fratello
Giorgio.
MI RACCONTI PIU' COSE SULLA SUA FAMIGLIA.
Siamo cinque fratelli, e Giorgio e' il maggiore.
PROSEGUA.
Anche quando ero un ragazzino Giorgio non aveva
alcuna considerazione per i miei diritti.
I SUOI DIRITTI.
Si'. Non mi dava un momento di pace.
MI SEMBRA CHE LEI SIA MOLTO SICURO DI CIO'.
```

Un breve esempio di colloquio dal programma Doctor. Le parti in maiuscolo sono i contributi del calcolatore.

```
QUAL E' IL SUO PROBLEMA?
Picchio frequentemente il mio cane.
IL SUO CANE.
Povero Fido. Piange e guaisce quando lo picchio.
NON CREDO DI CAPIRE COMPLETAMENTE CIO' CHE
VUOL DIRE.
Il fatto e' che non sono sicuro se odio o no Fido.
VERAMENTE NON SA SE ODIA O NO FIDO?
Che cosa sta pensando?
PERCHE' MI FA QUESTA DOMANDA?
Voglio conoscere il suo punto di vista.
PERCHE' LA PREOCCUPA IL MIO PUNTO DI VISTA?
Perche' la stimo.
STIAMO DISCUTENDO DI LEI,NON DI ME.
Mi scusi.
COSA SENTE QUANDO SI SCUSA?
```

Ancora un brano di colloquio con Doctor.

sione modificata di Eliza fu il programma Doctor, più specificamente orientato alla psicoterapia "non direttiva" (cioè più basata sull'autonoma ricerca del paziente attraverso il dialogo, che non sulle indicazioni fornite dal terapeuta), sostenuta dallo psicoterapeuta Carl R. Roger; una versione più provocatoria e divertente fu invece Parry, che, scambiando i ruoli del calcolatore e dell'utente, simulava un comportamento paranoico, a diversi livelli di gravità: quest'ultimo lavoro è dovuto a Kenneth Colby.

Il programma Doctor

Per comprendere l'efficacia del programma, esaminiamo due brani di colloquio: il primo è rappresentato nella pagina precedente, il secondo in questa pagina (ovviamente i brani sono tradotti dall'inglese in italiano, e quindi non completamente fedeli nelle specifiche fasi. Le parti in maiuscolo erano generate dal programma, mentre le altre erano introdotte dall'utente).
Cosa c'è di intelligente dietro questo programma? Quale potenza è racchiusa in un programma Lisp che interpreta le frasi fornitegli in linguaggio naturale e che risponde a tono? In realtà si tratta di una raffinata finzione, che tuttavia vale la pena di esaminare in qualche dettaglio e che mette comunque in evidenza la potenza del linguaggio adottato.
Di fatto Eliza o Doctor non hanno alcuna capacità di comprendere che cosa viene loro detto; semplicemente sono due programmi in grado di esaminare le frasi che a loro vengono proposte, più o meno come è stato fatto negli esempi di Lisp precedentemente forniti, ma con maggior raffinatezza sulla struttura delle stesse.

Il programma ha al suo interno una tabella contenente un certo insieme di "parole chiave" (o "locuzioni chiave"), che usa come elementi guida per dirigere il colloquio. Scandendo una frase sottopostagli, esamina se una di tali parole chiave è presente e, sulla base di ciò, costruisce una risposta: se per esempio trova un elemento come mio padre, mio fratello ecc., allora genera in modo fisso una frase del tipo MI PARLI DI PIÙ DELLA SUA FAMIGLIA; se invece non trova alcun elemento a cui appigliarsi, risponde con generiche VADA AVANTI, oppure NON SONO SICURO DI CAPIRE ESATTAMENTE CIÒ CHE VUOLE DIRE.
Talvolta addirittura, individuando qualche elemento "interessante" in una frase, lo memorizza per poterlo sfruttare successivamente: così, se per esempio una frase detta è del tipo talvolta penso di essere un fallito può causare, a fronte di una frase che non presenta alcuna parola chiave, una risposta del tipo PRIMA LEI MI AVEVA DETTO CHE A VOLTE PENSA DI ESSERE UN FALLITO, che dà effettivamente l'impressione all'interlocutore di avere di fronte a sé un individuo raziocinante.
Eliza aveva lo scopo di mostrare un certo insieme di possibilità sulla rappresentazione del linguaggio e sull'interfaccia uomo-macchina; di fatto esso scatenò polemiche tra gli addetti ai lavori, e in particolare tra Weizenbaum e alcuni psicanalisti che ritenevano di poter realmente usare il prodotto nella terapia.

Il programma Parry

La versione "capovolta" di Eliza, che fa interpretare al calcolatore la parte del paziente, ha a disposizione diversi "livelli"

```
Da quanto tempo si trova in clinica?
DA CIRCA SETTE GIORNI.
Si sente depresso?
NO.
Prende qualche medicinale?
NO.
Che tipo di problema ha?
LE PERSONE MI INNERVOSISCONO.
In che modo?
MI OSSERVANO.
Perche' l'osservano?
PROBABILMENTE PERCHE' IL MIO ASPETTO NON PIACE.
Lei ha un aspetto strano?
PERCHE' MI DOMANDA DEL MIO ASPETTO?
Perche' lei mi ha detto che la gente l'osserva.
SI',QUALCHE VOLTA.
```

Un dialogo con Parry a livello di "depressione".

```
Perche' e' in clinica?
NON C'E' MOTIVO CHE IO SIA QUI.
Chi l'ha portata?
I POLIZIOTTI.
Che problemi ha con la polizia?
I POLIZIOTTI NON FANNO IL LORO DOVERE.
E che cosa fanno allora?
ARRESTANO LA GENTE PERBENE.
Talvolta arrestano gente colpevole.
LA LEGGE DOVREBBE PUNIRE LA DELINQUENZA ITALIANA
Lei e' nato in Italia?
SONO INGLESE, ANCHE SE SONO NATO IN QUEL PAESE.
Conosce delinquenti italiani?
STO ALLA LARGA DA QUELL'AMBIENTE.
Ne ha paura?
LA MAFIA E' TREMENDA.
Ha qualche rapporto con la mafia?
LORO MI CONOSCONO.
Come mai?
MI SPIANO.
Perche' la spiano?
VOGLIONO FREGARMI CON PROVE FALSE.
```

Ancora Parry in azione: qui a un secondo livello di "mania di persecuzione".

di paranoia; a un livello di "depressione" vengono scatenati dialoghi come quello riportato nella figura sopra (in maiuscolo gli interventi del calcolatore); mentre a un secondo livello di "mania di persecuzione" si ottengono dialoghi come quello della figura sotto, e così via.
La tecnica è la stessa: molte apparenze e poca sostanza.

Il "Robot" di Winograd

Non dobbiamo però pensare che tutte le cose mostrate non abbiano dietro uno spazio concreto di rappresentazione della coscienza e di possibilità che un calcolatore "capisca" ciò che gli viene detto: a riprova di ciò sta il programma costruito come tesi di dottorato di ricerca da Terry Winograd, altro importante personaggio statunitense legato alla storia delle applicazioni in Lisp e dell'Intelligenza artificiale.
Winograd mise a punto un programma che realizzava su uno schermo video un "mondo" costituito di solidi geometrici colorati: cubi, scatole, coni, piramidi, e così via.
Un "braccio" visibile sullo schermo era il robot con cui interloquire; l'utente forniva comandi in linguaggio naturale al robot, che li eseguiva o, qualora si presentassero casi di ambiguità, faceva adeguate domande per chiarire il problema; i

Solleva la piramide verde.
(Il braccio si spostava sullo schermo, prelevava la piramide verde, e la sollevava)
OK.
Mettila sopra il cubo rosso.
(Il braccio depositava la piramide sul cubo rosso, posto che questo avesse una superficie libera di appoggio, cioè non avesse sopra qualcos'altro; si noti la presenza del pronome che identifica la piramide; in realtà, in inglese, il riconoscimento dei pronomi è più facile, non essendo questi aggregati come suffisso a un verbo, come accade invece in italiano).
OK.

Il "robot" di Winograd in azione nel mondo dei blocchi.

comandi erano indicazioni di prelevare oggetti, spostarli, identificarli, e così via.
La figura sopra riporta un dialogo tipico.
Supponiamo che nel "mondo" fosse presente un cubo verde sopra un parallelepipedo blu; allora poteva avere luogo il seguente dialogo:

Metti la piramide rossa sul parallelepipedo blu.
NON CAPISCO COSA INTENDI CON "SUL": GENERICAMENTE "AL DI SOPRA" O "A CONTATTO CON"?
A contatto con.
(Il braccio si sposta, preleva il cubo che si trova sul parallelepipedo blu, lo deposita altrove, preleva la piramide rossa che mette sul parallelepipedo)
OK.

Era prevista anche una forma di apprendimento:

Chiamiamo "PILA" un CUBO con sopra un CUBO, con sopra una piramide.
OK.

Ci sono pile?
(supponendo che il "mondo" non ne presenti alcuna)
NO.
Costruiscine una.
(il braccio si sposta e, prelevando i vari pezzi dai vari punti in cui sono presenti, con tutti gli spostamenti necessari, costruisce la figura richiesta, mostrando di avere compreso e memorizzato nome e significato del termine "PILA").

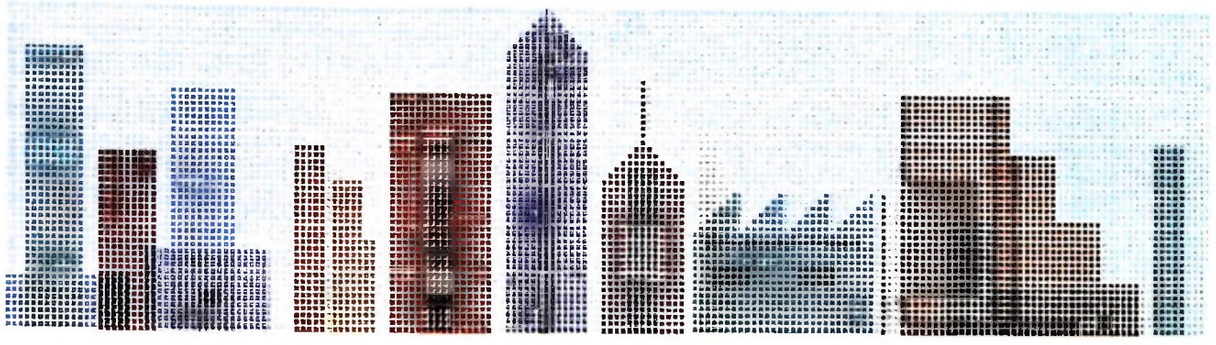
Addirittura, a richieste di compiere operazioni non possibili come il mettere un cubo su una piramide, si dichiara "incompetente":

Metti un cubo sulla piramide gialla.
NON SO COME FARE.

Il Lisp è ancora oggi uno dei principali punti di riferimento per applicazioni di Intelligenza artificiale, anche se diversi nuovi linguaggi di programmazione sono stati realizzati per questo tipo di applicazioni.

PLOTTER E GRAFICA INTERATTIVA

Utilizzando il plotter riproduciamo su carta quanto realizzato su video.



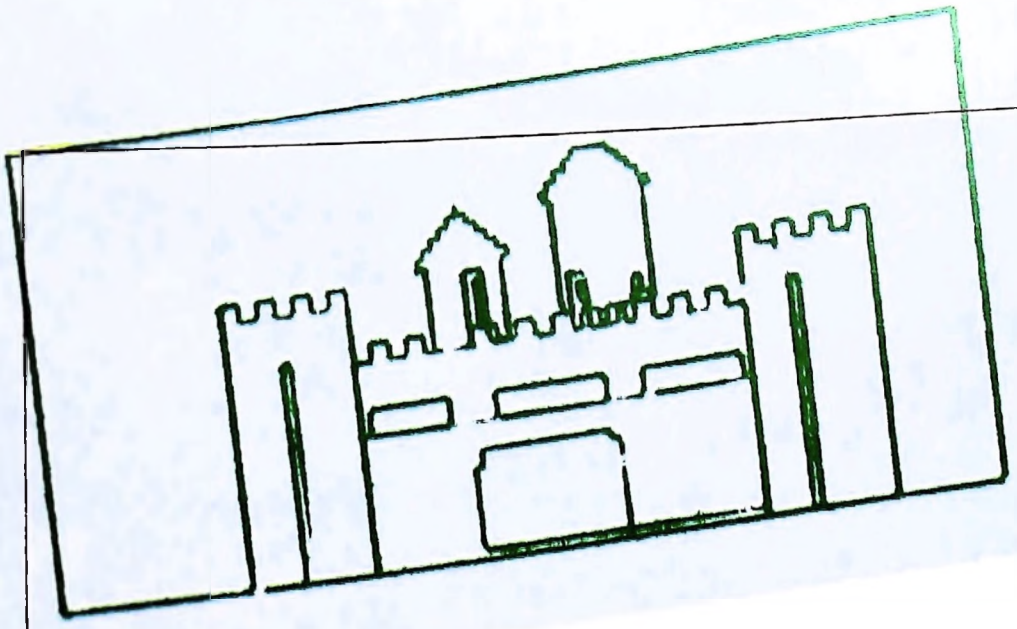
In alcuni articoli precedenti si sono affrontate le più semplici problematiche relative alla grafica interattiva.
Si è fatto a tale scopo ricorso a due programmi che sono stati indicati con il nome Pen e Pen2.
Le caratteristiche principali di Pen in sintesi sono:
1) possibilità di utilizzare lo schermo dell'M10 come se si trattasse di una lavagna;
2) possibilità di disegnare in "tempo reale" rendendo inoltre disponibile una correzione immediata di quanto disegnato.
Ovvia estensione a quanto trattato in Pen è stato il programma Pen2 che offriva la possibilità di memorizzare, su di un Data File di nome predefinito, la sequenza delle coordinate relative ai segmenti introdotti.
Successivamente, mediante la primitiva LINE del BASIC M10, si rendeva possibile rivedere il disegno creato, con ovvio vantaggio sul tempo impiegato per la visualizzazione.
Questo programma avrebbe potuto rappresentare, se opportunamente espanso, il primo stadio per la costruzione di un Editor Interattivo a due dimensioni.
Tale programma applicativo potrebbe essere utilizzato per creare disegni da utilizzare per scopi diversi (per esempio per una presentazione grafica di un programma) semplicemente leggendo tramite una procedura (che potrebbe essere estratta direttamente dal programma Pen2 stesso) i dati che sono stati immessi nei Data File.
Sulla base di questo discorso si vuole ora affrontare più approfonditamente il problema fino ad arrivare, in seguito, alla stesura di un vero e proprio "pacchetto" didattico-applicativo per la grafica interattiva a due dimensioni.
A tale scopo proponiamo in questa lezione un ampliamento del programma Pen2.

Il programma Pen3

Con questo programma, il plotter s'inserisce nel discorso della grafica interattiva come uno strumento che offre la possibilità di riprodurre su carta quanto realizzato su video.
Lo scopo del programma è, naturalmente, didattico e si propone principalmente d'indicare uno dei possibili metodi per la costruzione in tempo reale di comandi grafici per questa periferica.

La struttura

Il listato è sostanzialmente da interpretarsi come un rifacimento del programma Pen2, del quale utilizza la struttura, e



In questa e nella pagina precedente, due esempi di riproduzione su carta, mediante plotter, di disegni ottenuti sul video del computer. Il programma PEN3 introdotto in queste pagine si propone di indicare uno dei metodi possibili per la costruzione in tempo reale di comandi grafici per il plotter, per inserire utilmente questo strumento nel campo della grafica interattiva.

a cui sono stati aggiunti i comandi relativi alla realizzazione di un disegno su plotter.
Si è resa più fluida la struttura di lettura dei comandi forniti da tastiera mediante l'uso dell'istruzione GOSUB alle linee 24 e 25 (è bene ricordare, a questo proposito che questa istruzione ha una importanza fondamentale per la programmazione strutturata, dal momento che permette di separare parti di programma e di unirle poi mediante le opportune chiamate, nonché di rientrare all'istruzione successiva a quella di chiamata).
Poiché si deve controllare in due parti completamente diverse del programma (disegno su schermo e su plotter) se il Data File che si vuole aprire è presente o meno, si è resa necessaria la costruzione di un'apposita procedura indipendente che viene chiamata alle linee 130 e 510. La procedura restituisce un valore di controllo (comunemente indicato con il nome di "flag") nella variabile WF che indica la presenza (se ha valore 1) o l'assenza (valore 0) di errori nell'apertura del Data File.
La cancellazione del Data File, che si può ottenere mediante la pressione del tasto "c" in modalità MEMO COORDINATE, viene ora effettuata con una specifica istruzione del Basic dell'M10:

```
KILL "nomefile.tipofile"
```

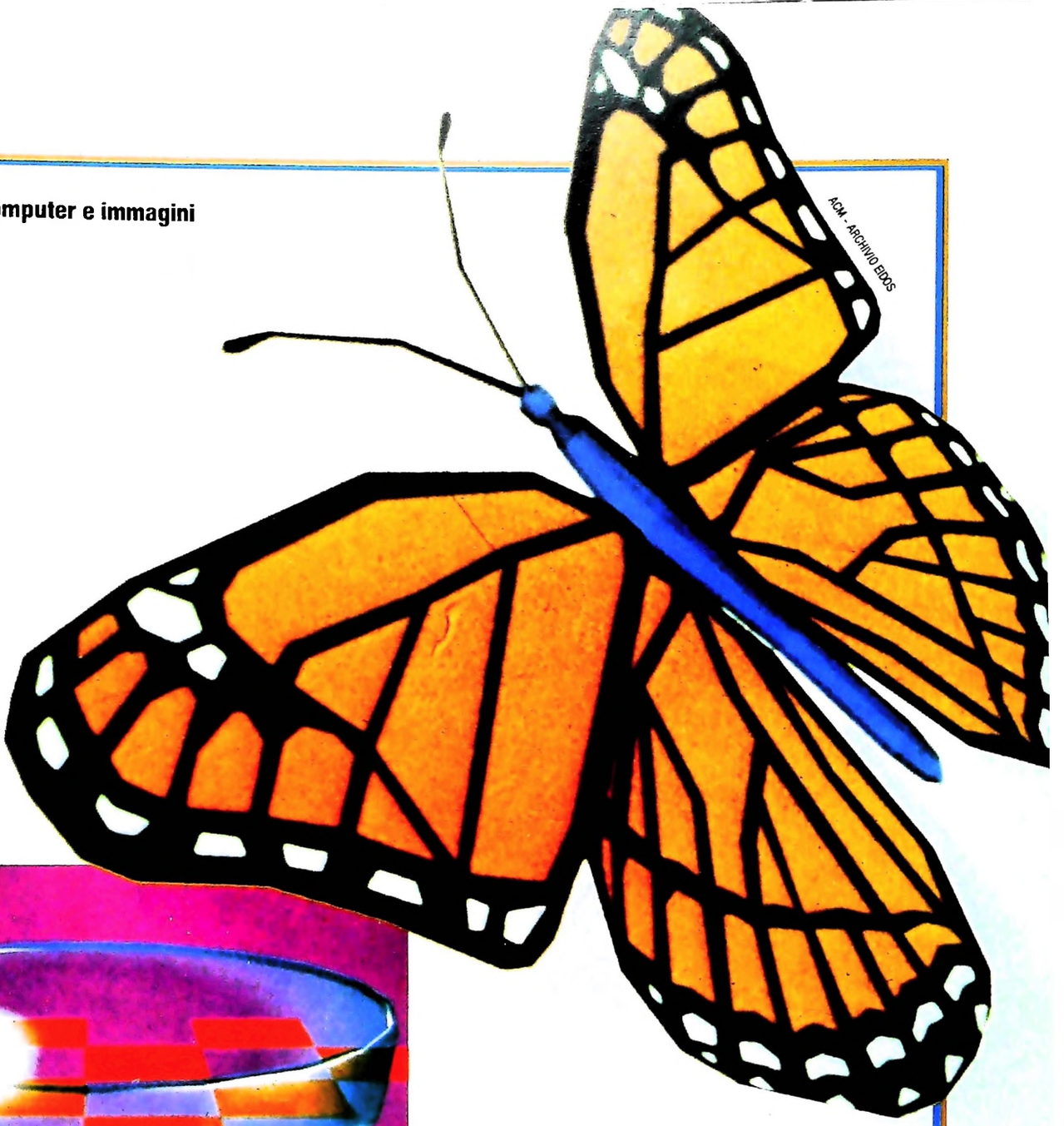
dove nomefile = nome del file che si vuole cancellare (M10 riconosce un massimo di 6 lettere significative) e tipofile = tipo del file considerato (DO per file di testo e BA per programmi in BASIC).
Si è inoltre prevista una standardizzazione del comando relativo all'uscita dal programma: ora è sufficiente premere il tasto q per terminare l'esecuzione (l'abbandono di un programma mediante il comando di QUIT è in pratica di uso comune in moltissimi programmi).
La parte che è stata aggiunta al programma già esistente è quella indicata nelle linee 500-620.

Dopo aver creato un disegno mediante i comandi di MEMO COORDINATE si effettua copia su plotter di tale oggetto mediante la sequenza di comandi:
1) f (si entra in ambiente di gestione del Data File);
2) p (si esegue la copia sul plotter del disegno).

La procedura di disegno su plotter

L'esecuzione dei comandi indicati sopra attiva la procedura che ha inizio alla linea 500. Si esegue un controllo sulla presenza o meno dei dati cercati e si pone il plotter in modalità grafica (linee 500-525).
A questo punto con il comando H si riporta la penna a sinistra all'origine del sistema di riferimento del plotter e poi, mediante (M x, y) là si posiziona all'inizio del disegno (che per come è stato configurato lo schermo ha sempre coordinate x=8, y=106 per il plotter).
A questo punto si deve costruire la stringa di comandi per il microplotter.
Per la costruzione di tale sequenza si utilizzano le già viste istruzioni STR$ e LEN.
Mediante la STR$ si convertono in stringhe i valori numerici che vengono letti da Data File e vengono sommati fra di loro in modo tale da costituire un'unica stringa preceduta dal comando relativo (nel nostro caso D che indica Draw).
Poiché la lunghezza massima accettata da M10 per una stringa è di 255 caratteri, si procede a un controllo sulla lunghezza di quella che si sta creando tenendo conto che un numero può essere costituito anche da 3 cifre (a tale scopo si pone a 240 la soglia di controllo). Al raggiungimento dalla massima lunghezza della stringa di comando, si effettua il plottaggio dei primi segmenti "codificati". Si prosegue in questo modo fino a quando si sono esaurite le coordinate del disegno memorizzato nel Data File.
Si plotta a questo punto l'ultima stringa di comandi e si rientra in ambiente operativo principale.

Computer e immagini



ACM - ARCHIVIO EIDOS

ACM - ARCHIVIO EIDOS

Lo strumento computer è ormai il mezzo privilegiato per la riproduzione di immagini sia che si limiti alla creazione di semplici disegni al tratto, sia che traduca la creatività di un artista.
Proponiamo in questa pagina due esempi di immagini realistiche realizzate col computer. Il grado di riproduzione è molto elevato, grazie a nitidezza e precisione dei contorni. Il computer sembra diventare il mezzo per dare all'espressione artistica una nuova orientazione. È evidente il realismo della farfalla; nel caso del calice è stato ottenuto un notevole effetto di trasparenza e riflessione.

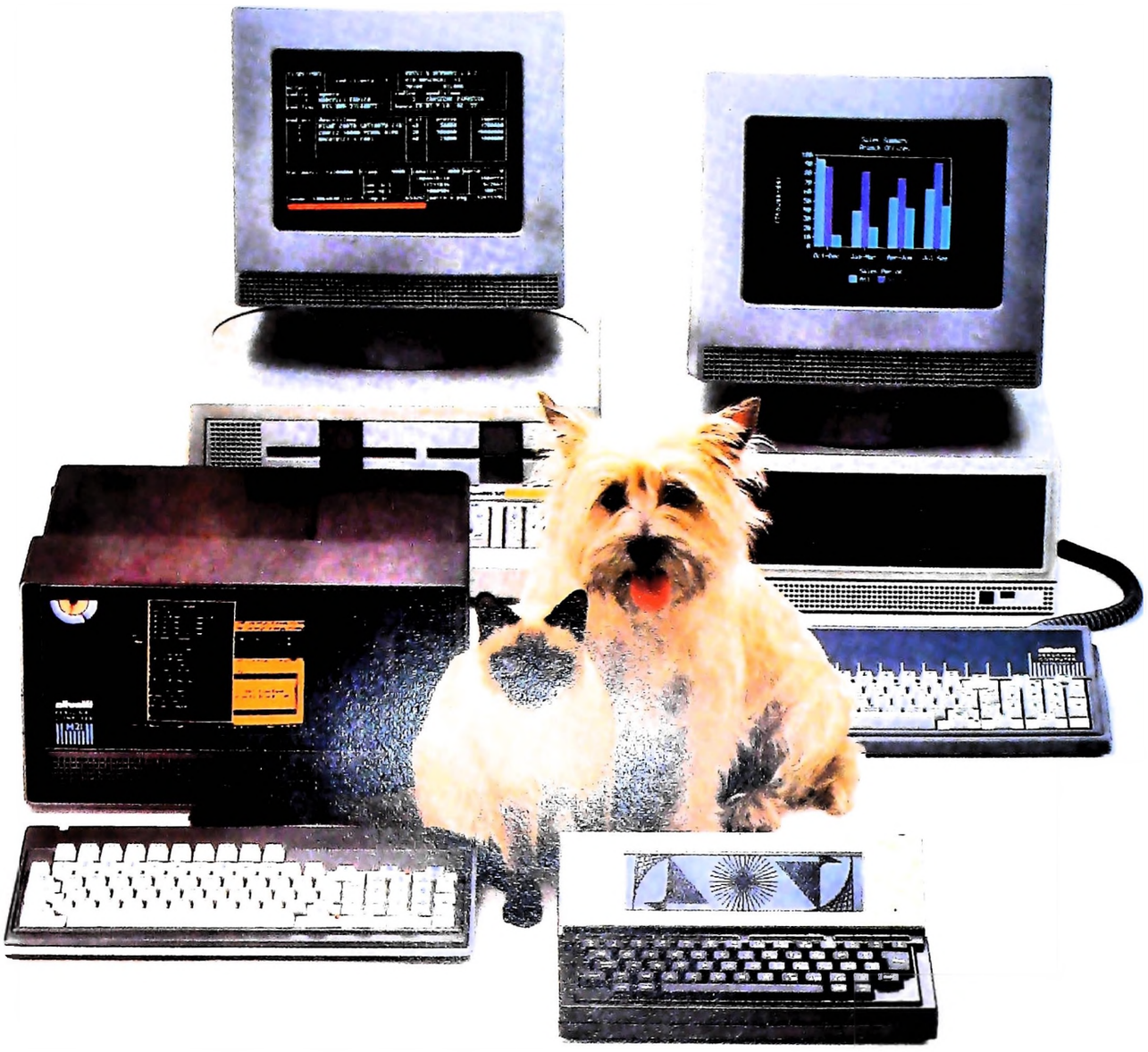
Espansioni al programma e conclusioni

È possibile, e certamente utile, prevedere la possibilità di creare e leggere un Data File il cui nome viene introdotto dall'utente.
Questo per fornire la possibilità di creare un numero di disegni limitato solo dalle capacità della memoria di massa disponibile; ora infatti rimane memorizzato solo l'ultimo dei disegni creati, poiché la cancellazione del Data File avviene automaticamente alla creazione del successivo.
A causa della risoluzione notevolmente più elevata del plotter rispetto a quella dello schermo, il disegno viene riprodotto su plotter ingrandito due volte ma potrebbe essere utile prevedere una trasformazione simile a quella di scala per poter agire a piacimento sulle dimensioni del disegno.
Questo programma si propone di essere il primo di una serie che ha lo scopo di portare alla realizzazione di un pacchetto in grado di utilizzare il plotter per più sofisticate funzioni: il pilotaggio interattivo delle penne e dei colori e la possibilità di simulare, con le ovvie limitazioni di dimensioni, la funzione di Digitizer permettendo così di utilizzare questa versatile periferica come elemento di input dei dati.

```
5  REM**INIZIALIZZAZIONE E MENU**
7  CLEAR 1500
8  CLS
10 PRINT@0,"*PEN3*M>ATITA,C>LS,F>ILE,Q>UIT,    MOV.
11 MM=0:X=2:Y=10
12 CL=1:LL$=A$:A$=""
13 A=X:B=Y
14 REM**LETTURE DA TASTIERA
15 A$=INKEY$:IF A$=""THEN 15
17 REM
18 IFASC(A$)=28THENX=X+1:GOTO 28
19 IFASC(A$)=29THENX=X-1:GOTO 28
20 IFASC(A$)=30THENY=Y-1:GOTO 28
21 IFASC(A$)=31THENY=Y+1:GOTO 28
22 IFA$="F"THEN CL=0:GOTO 28
23 IFA$="M"THEN CL=1:GOTO 28
24 IFA$="S"THEN GOSUB 1000:GOTO 10
25 IFA$="C"THEN GOSUB 800:GOTO 10
26 IFA$="Q" THEN CLS:END
27 GOTO 12
28 '*********************************
29 '***CONTROLLO SULLE COORDINATE******
30 IF X<3 THEN X=3
31 IF X>236 THEN X=236
32 IF Y<10 THEN Y=10
33 IF Y>60 THEN Y=60
45 IF CL=0THEN 60
46 '*********************************
48 '***DISEGNO E MOVIMENTO MATITA******
50 LINE(A-1,B-1)-(A+1,B+1),0,B F
52 LINE(X-2,Y-2)-(X+1,Y+1),1
53 IF MM=1 THEN GOSUB 300
54 PSET(X,Y,1)
55 GOTO 12
58 '*********************************
60 '***GESTIONE FILE SUPPORTO**********
100 PRINT@ 0,"GESTIONE FILE M>EMO,D>ISEGNA,P>
LOTTER "
110 RP$=INKEY$:IF RP$=""THEN 110
120 IF RP$="M" THEN 250 ELSE IF RP$="P" THEN 500 ELSE
IF RP$<>"D" THEN 110
122 '*********************************
125 '***DISEGNA IL FILE CREATO*********
130 GOSUB 1270
140 IF WF=1 THEN 10
149 '***IL CONTENUTO E' ORA DISEGNATO**
150 INPUT# 1,X1,Y1
160 FOR CI=1 TO 2000
170 IF EOF(1) THEN CLOSE 1:GOTO 10
180 INPUT#1,X2,Y2
190 LINE(X1,Y1)-(X2,Y2)
200 X1=X2:Y1=Y2
210 NEXT CI
220 CLOSE 1
230 '*********************************
250 '***GESTIONE MEMO DEI DATI*********
260 PRINT@0,"MEMO COORDINATE***C>ANCELLA FILE,S>
TOP"
270 MM=1
275 OPEN "RAM:DRAW" FOR OUTPUT AS 1
280 GOTO 48
300 IF ASC(A$)>27 AND ASC(A$)<32 THEN 330 ELSE
RETURN
320 '*********************************
330 '********MEMORIZZA I DATI**********
340 IF A$<>LL$ THEN PRINT# 1,A;B
350 RETURN
360 '*********************************
500 '***DISEGNO SU PLOTTER*************
510 GOSUB 1270
520 IF WF=1 THEN 10
525 LPRINT CHR$(18) '**PLOTTER GRAFICO*
527 LPRINT"H"
528 LPRINT"M 8,106"
530 P$="D"
540 INPUT #1,PX,PY
550 P$=P$+STR$(PX*2)+","+STR$(126-PY*2)
560 FOR CT=1 TO 2000
570 IF EOF(1) THEN CLOSE 1:GOTO 610
580 INPUT #1,PX,PY
590 P$=P$+","+STR$(PX*2)+","+STR$(126-PY*2)
595 IF LEN(P$)>240 THEN LPRINT P$:GOTO 530
600 NEXT CT
610 LPRINT P$
615 LPRINT CHR$(17)
620 GOTO 10
630 '*********************************
800 '***CANCELLA IL FILE***************
805 IF MM<>1 THEN CLS ELSE CLOSE 1:CLS:KILL "DRAW.DO"
810 RETURN
820 '*********************************
1000 '***FINE DEL PROGRAMMA*************
1010 IF MM=1 THEN PRINT# 1,X,Y
1020 CLOSE 1
1030 RETURN
1040 '*********************************
1270 'CONTROLLO SE ESISTE IL FILE******
1280 WF=0
1300 OPEN "RAM:DRAW"FOR APPEND AS 1
1420 CLOSE 1
1460 OPEN "RAM:DRAW" FOR INPUT AS 1
1480 IF EOF(1) THEN PRINT@200,"ATTENZIONE, DATI
INESISTENTI          ":WF=1:CLOSE 1
1510 RETURN
```

Anche in leasing con Olivetti Leasing

LA FAMIGLIA DEI PERSONAL COMPUTER OLIVETTI



FRIENDLY & COMPATIBLE

Questa famiglia di personal compatibili tra loro e con i più diffusi standard internazionali, non ha rivali per espandibilità e flessibilità. Prestazioni che su altri diventano opzionali, sui personal computer Olivetti sono di serie. Per esempio M24 offre uno schermo ad alta definizione grafica, ricco di 16 toni o di 16 colori e con una risoluzione di 600x400 pixel; mentre la sua unità base dispone di 7 slots di espansione, fatto questo che gli consente di accettare schede di espansione standard anche se utilizza un microprocessore a 16 bit reali (INTEL 8086). Ma ricchi vantaggi offrono anche tutti gli altri modelli.

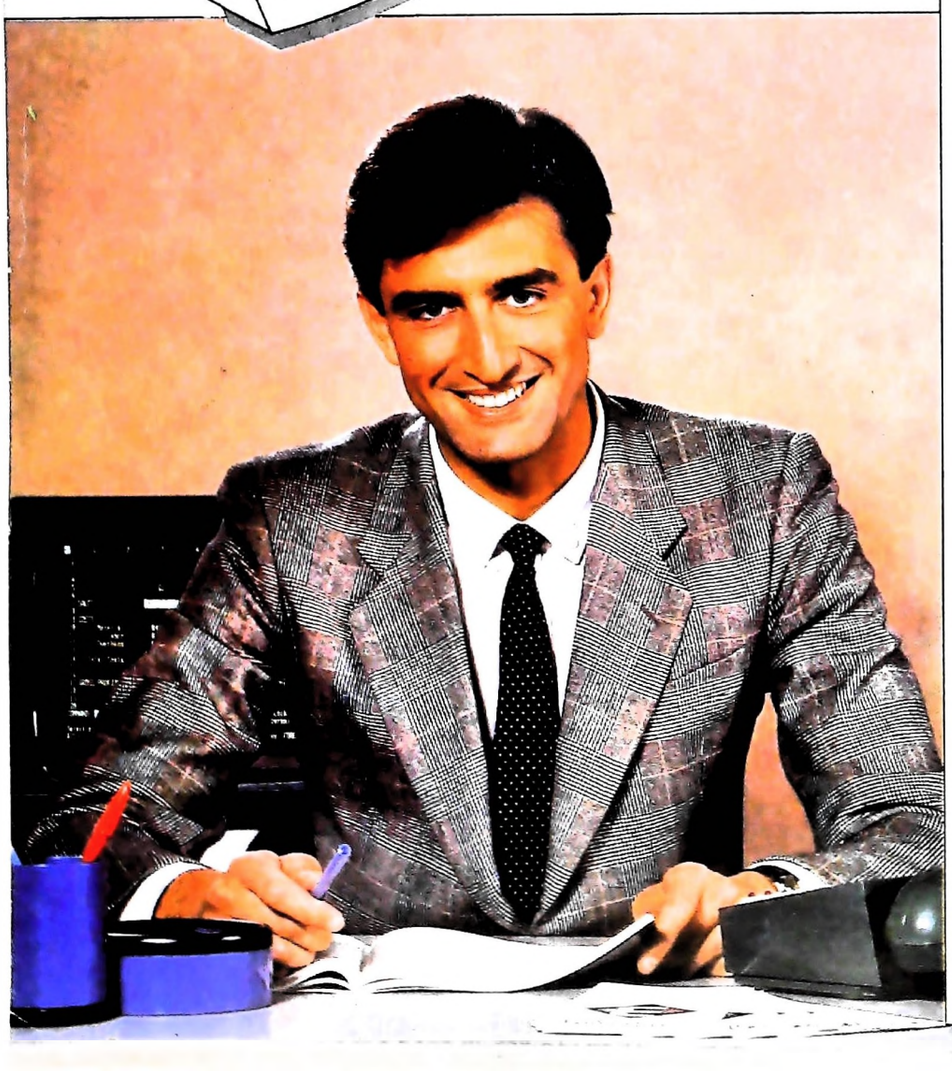
Basti pensare che tutte le unità base includono sia l'interfaccia seriale che quella parallela. Oppure basti pensare all'ampia gamma di supporti magnetici: floppy da 360 a 720 KB o un'unità hard disk (incorporata o esterna) da 10 MB.
La loro compatibilità, inoltre, fa sì che si possa far uso di una grande varietà di software disponibile sul mercato. Come, ad esempio, la libreria PCOS utilizzabile anche su M24. Come le librerie MS-DOS®, CP/M-86® e UCSD-P System®, utilizzabili sia da M20 che da M21 e M24.

olivetti

Per maggiori informazioni, inviare il coupon a: Olivetti
Divisione Personal Computer Via Meravigli 12, 20123 Milano.
NOME
INDIRIZZO
CITTÀ
TELEFONO

UN NUOVO MODO DI USARE LA BANCA.

Consulenza



GLI INVESTIMENTI CON VOI E PER VOI DEL BANCO DI ROMA.

Il Banco di Roma non si limita a custodire i vostri risparmi. Vi aiuta anche a farli meglio fruttare. Come? Mettendovi a disposizione tecnici e analisti in grado di offrirvi una consulenza di prim'ordine e di consigliarvi le forme di investimento più giuste. Dai certificati di deposito ai titoli di stato, dalle obbligazioni alle azioni, il Banco di Roma vi propone professionalmente le varie opportunità del mercato finanziario. E grazie ai suoi "borsini", vi permette anche di seguire, su speciali video, l'andamento della Borsa minuto per minuto.

Se desiderate avvalervi di una gestione qualificata per investire sui più importanti mercati mobiliari del mondo, i fondi comuni del Banco di Roma, per titoli italiani ed esteri, vi garantiscono una ampia diversificazione.

Inoltre le nostre consociate Figeroma e Finroma forniscono consulenze per una gestione personalizzata del portafoglio e per ogni altra esigenza di carattere finanziario.

Veniteci a trovare, ci conosceremo meglio.

BANCO DI ROMA
CONOSCIAMOCI MEGLIO.