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

CIAD CL

40 CORSO PRATICO COL COMPUTER

421925

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

F4 F5 F6 F7 F8

diretto da GIANNI DEGLI ANTONI

BATTERY LOW

DAL 18 GENNAIO 1985

STORIA D'ITALIA EINAUDI.

IN EDICOLA

A FASCICOLI SETTIMANALI.

SCOPRI UN PAESE STRAORDINARIO.

IL TUO.

2 fascicoli di Storia
+ 2 fascicoli di Documenti
152 pagine – L. 3.000

FABBRI EDITORI

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
ADRIANO DE LUCA (Professore di Architettura del Calcolatori all'Università Autonoma Metropolitana di Città del Messico), GOFFREDO HAUS, MARCO MAIOCCHI, DANIELE MARINI, GIANCARLO MAURI, CLAUDIO PARMELLI, ENNIO PROVERA

Testi
ADRIANO DE LUCA, CLAUDIO PARMELLI,
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

AVVISO AI LETTORI

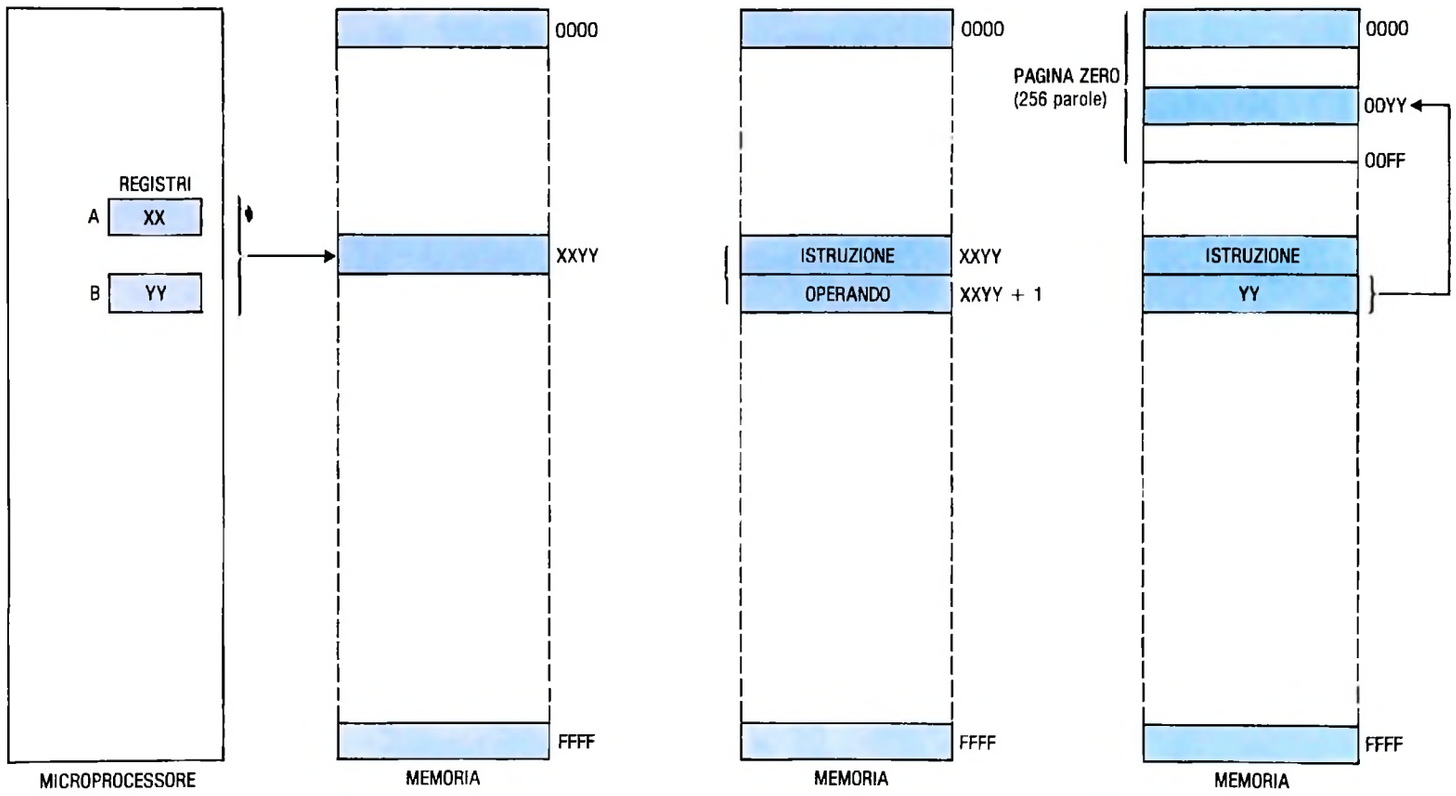
È in edicola la copertina per rilegare il terzo volume del "Corso pratico col computer".

FORME PER INDIRIZZARE LA MEMORIA

Come utilizzare gli indirizzi di memoria e le istruzioni della UAMICRO III.

Le forme attuali per indirizzare la memoria sono le seguenti:
attraverso registri interni;
immediata;
diretta;
estesa;
implicita;
relativa;
indexata;
indiretta;

Spieghiamo come vengono realizzate.
Attraverso registri interni. Questa forma (figura in basso a sinistra) molto usata per i microprocessori della famiglia 8080 e Z80, consiste nel depositare l'indirizzo in due registri interni e poi incrementarlo e decrementarlo, secondo le necessità, per andare a prendere l'operando in memoria.
Le istruzioni che usano queste forme di indirizzamento si compongono di un solo byte.
Immediata. Usata da tutti i microprocessori, indica (figura al



A sinistra: come s'indirizza la memoria attraverso registri interni: l'indirizzo si compone della somma di A e B. Al centro: nell'istruzione immediata l'operando segue l'istruzione. A destra: con l'istruzione diretta, nelle prime 256 località di memoria solo il byte meno significativo cambia; quello più significativo resta fisso a zero.

centro della pagina precedente) che il secondo byte dell'istruzione è l'operando su cui dovrà agire l'istruzione stessa. L'istruzione occupa complessivamente due byte.

Diretta. Questa forma (figura a destra della pagina precedente) è molto interessante e consiste in questo: poiché il secondo byte dell'istruzione è di 8 bit (come già sappiamo) e quindi al massimo può indirizzare 256 località, noi possiamo depositare i dati su cui dovremo operare nelle prime 256 posizioni di memoria. Risulta allora semplice usare un solo byte di indirizzo per andare in quelle località di memoria. Questa forma è molto usata dalla famiglia 6800 e 6500. (Molti fabbricanti chiamano Pagina Zero *le prime* 256 località di memoria, cioè quella parte di memoria indirizzabile con un solo byte, essendo l'altro forzatamente a zero).
Istruzione e operando occupano due byte.

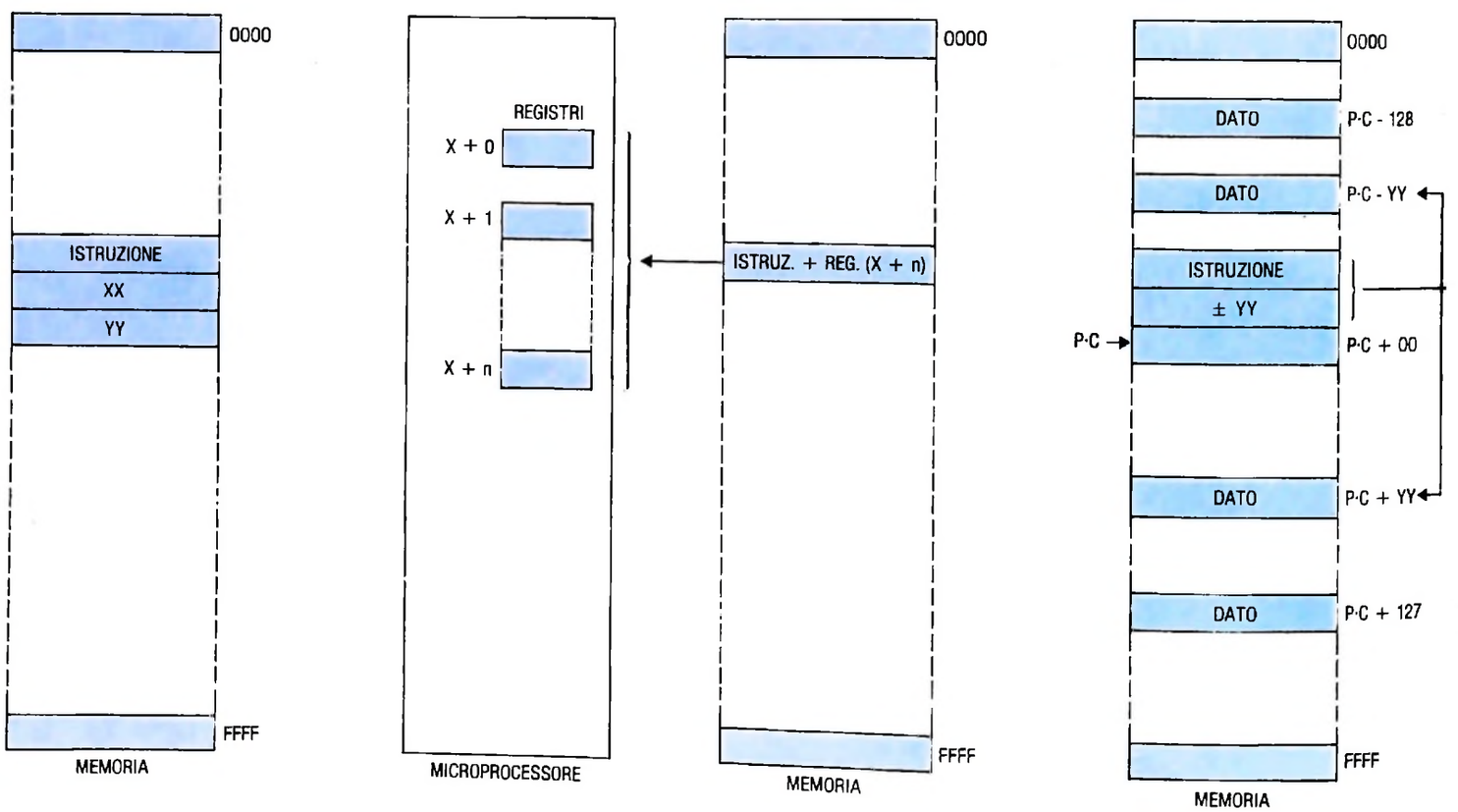
Estesa. Questa forma (figura in basso a sinistra) permette di accedere a qualsiasi parte della memoria, però occorrono 3 byte: uno per il codice operativo dell'istruzione e due per l'indirizzo.

Implicita. Anche questa forma è corta (figura in basso al centro) perché è fatta da un solo byte, che contiene sia il codice operativo dell'istruzione che l'operando, il quale è quasi sempre un registro interno.

Relativa. Questa forma si usa per effettuare salti condizionati e l'indirizzo di arrivo del salto è dato dalla somma algebrica (per cui si può incrementare o decrementare) dell'indirizzo in cui si trova (Program Counter) e del contenuto del 2° byte. Il secondo termine può essere negativo se il salto è all'indietro, come nei "loop". Poiché l'operando è composto da un solo byte, i posti raggiungibili sia in avanti come all'indietro, rispetto al punto in cui siamo, sono —128, e +127 (figura in basso a destra), cioè in totale 256 compreso lo zero.

Indexato. Significa che l'indirizzo dell'operando è dato dalla somma del contenuto del registro index più lo spostamento offset dato dal secondo byte dell'istruzione. In questo modo possiamo muoverci dal punto indicato dall'index in avanti fino a 255 località. Esistono microprocessori che permettono la somma algebrica dell'offset per cui è possibile anche tornare indietro (figura a sinistra della pagina accanto).
L'istruzione è composta da due byte: uno per il Codice Operativo e uno per l'offset.

Indiretta. Questa forma vuol dire che l'indirizzo dell'operando è dato dal contenuto di due posizioni adiacenti di memoria indirizzate dal 2° e 3° byte dell'istruzione (figura a destra della pagina accanto). L'istruzione è composta da tre byte: uno per il Codice Operativo e due per l'indirizzo.



A sinistra: come si presenta in memoria un'istruzione estesa. Al centro: come si presenta un'istruzione implicita: nello stesso byte ci sono il codice operativo dell'istruzione e il numero del registro interno dove è depositato l'operando. A destra: come s'indirizza con un'istruzione relativa: al valore P.C. viene sommato algebricamente il valore + - YY.

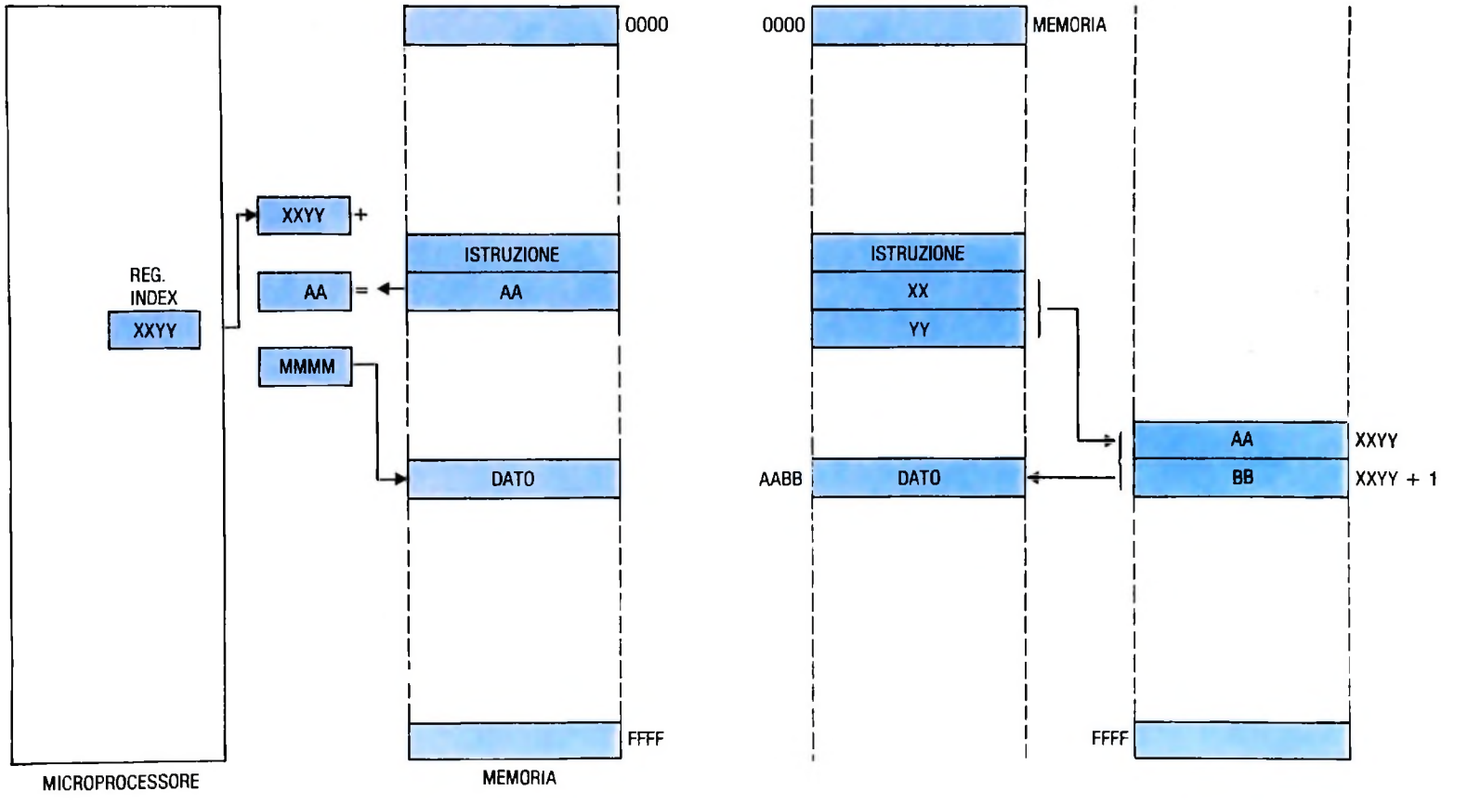
Istruzioni della Uamicro III

Per l'Uamicro III le forme scelte per indirizzare sono: immediata, indiretta, estesa, indexata, implicita e relativa. Anche in questo caso non è la quantità o la qualità delle istruzioni che conta, perché molto dipende dall'uso che si vuol fare del microprocessore.
Vedremo ora come si sviluppa la "microprogrammazione" che permette la realizzazione di alcune istruzioni tra quelle presentate nella tabella della pagina seguente, in alto.

Fasi

L'esecuzione di una qualsiasi istruzione si compone di diverse fasi, che in un microprocessore vanno da un minimo di cinque a più di venti; ma due sono sempre presenti: *scrittura* e *lettura* della memoria (figura della pagina seguente, a sinistra). In queste due fasi le funzioni di ENABLE sono date alla transizione negativa del clock, mentre quelle di LOAD alla transizione positiva. La nostra Uamicro, come si può vedere dalla figura, è del tipo clock unico: chiariamo bene questo concetto per non cadere in facili errori.

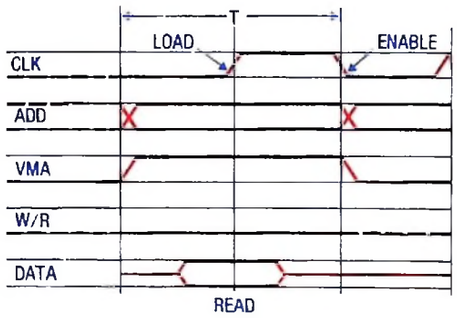
Qualsiasi microprocessore essendo un insieme di registri, nella maggior parte sincroni, ha bisogno naturalmente di un orologio che scandisce i tempi di azione. Ci sono due funzioni strettamente legate all'orologio e sono la funzione relativa ai registri interni, e quella relativa alla memoria o agli elementi esterni in genere. Per questa ragione la frequenza dell'orologio ha significato solo se si specifica a quale delle due funzioni si riferisce. Nella parte *b* della figura della pagina seguente, a destra, per esempio è illustrata la fase di lettura di un microprocessore a clock unico; nella parte *a* al primo clock si abilita il comando di READ, mentre nel terzo si ha la lettura vera e propria.
Tutte e due le forme sono usate nei microprocessori commerciali per cui se vogliamo sapere il tempo necessario per eseguire un'istruzione bisogna tenere presente questa differenza di funzionamento.
Le due tecniche finiscono per fare le stesse cose però in modo diverso e anche con cicli attivi diversi. Consideriamo ancora la figura citata: in *b* i segnali VMA e READ sono *attivi positivi* e la lettura viene fatta alla transizione positiva del clock. In *a* il $\overline{\text{READ}}$ e $\overline{\text{MREQ}}$ (equivalenti al $\overline{\text{VMA}}$) sono *attivi negativi* e vengono abilitati alla metà di T_1 mentre i dati vengono letti nella metà di T_3.

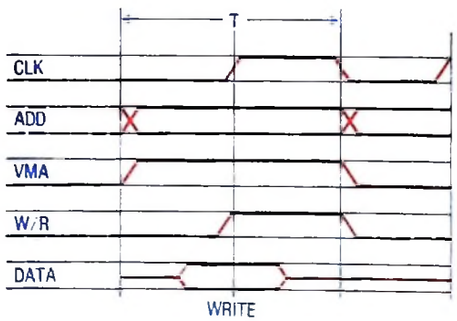


A sinistra: come s'indirizza la memoria con un'istruzione indexata: al valore del registro Index viene sommato il valore AA del 2° byte dell'istruzione (offset) e il risultato MMMM è preso come indirizzo del dato. A destra: come s'indirizza la memoria con un'istruzione indiretta; l'indirizzo viene depositato in due posizioni di memoria adiacenti, da cui viene prelevato con l'indirizzo XXYY per la parte alta e con l'indirizzo XXYY + 1 per la parte bassa.

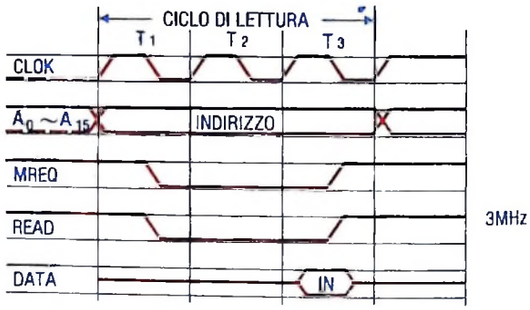
	INMED	DIRECT	EXTEND	INDEX	IMPLIED	
LDAA	X	X	X	X		M → A
LDAB	X	X	X	X		M → B
STAA		X	X	X		A → M
STAB		X	X	X		B → M
NEG			X	X		00 → M → M
NEG A					X	00 → A → A
NEG B					X	00 → B → B
TAB					X	A → B
TBA					X	B → A
⋮						
AND A	X	X	X	X		A·M → A
AND B	X	X	X	X		B·M → B
ADD A	X	X	X	X		A + B → A
ADD B	X	X	X	X		B + M → B
ABA					X	A + B → A
ADCA	X	X	X	X		A + M + C → A
EDRA	X	X	X	X		A + M → A
⋮						
⋮					RELATIVE	
BRA					X	C = 0 BRANCH
BCC					X	C = 1
BEQ					X	Z = 1
⋮						
JMP			X	X		
IMS						

Alcune istruzioni dell'Uamicro III (a lato): la 'X' indica che l'istruzione può indirizzare nella forma sopraindicata. Sotto, a sinistra: fase di lettura e scrittura della memoria. A destra: *a*, ciclo di lettura di un microprocessore a clock multiplo (3MHz); *b* ciclo di lettura di microprocessore a clock unico (1MHz).

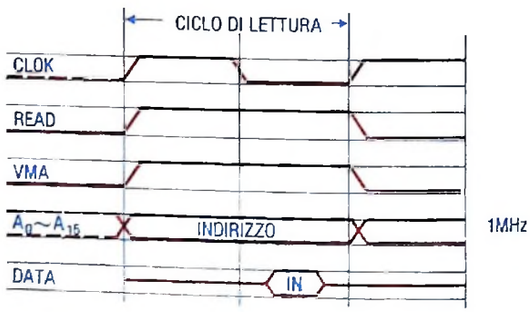




a)



b)



BASIC-LISP: UN INTERPRETE PER M10 (II)

Analizziamo le funzioni SET e SETQ.

Manipolazione di s-espressioni con LISP

L'apostrofo è utilizzato per "virgolettare" una s-espressione in BASIC-LISP; come conseguenza, l'interprete LISP non elaborerà ciò che segue, in quanto si sta dichiarando una costante. Infatti (DIV 35 7) è una funzione che trova risultato nel numero 5 e '(DIV 35 7) è una lista di tre atomi: DIV,35 e 7. L'apostrofo rappresenta l'abbreviazione della funzione QUOTE e l'espressione '(DIV 35 7) viene internamente rappresentata come (QUOTE(DIV 35 7)).
Come le variabili del BASIC, anche gli atomi del LISP possono assumere valori. Gli atomi che sono stati associati a un valore sono chiamati *atomi legati*; viceversa quelli che non sono associati a un valore sono identificati come *atomi slegati*. Come le variabili normali, il valore che un atomo può assumere può essere qualsiasi oggetto LISP: una lista, un numero o addirittura un altro atomo. Non ci sono atomi "stringa" e atomi "interi", un singolo atomo può essere associato a entrambi i valori in tempi differenti. Inizialmente tutti gli atomi non definiti sono slegati e possono essere associati a un valore mediante la funzione SET e SETQ.
Per esempio si supponga di voler rappresentare attraverso l'atomo FRATELLI la lista (CARLO MICHELE). Digitando (SET'FRATELLI' (CARLO MICHELE)), come pure (SETQ FRATELLI' (CARLO MICHELE)) si ottiene l'effetto desiderato, infatti chiedendo il valore di FRATELLI si ottiene la lista (CARLO MICHELE) (in basso).
Analizziamo la differenza tra SET e SETQ: la funzione SET valuta se l'atomo è legato così che lo stesso deve essere racchiuso da apostrofi se si vuole evitarne l'elaborazione (fate la prova), mentre SETQ non effettua elaborazioni. Si noti come SETQ può lavorare con parecchi assegnamenti alla volta.
Sia SET che SETQ assolvono a due compiti: assegnano il valore del loro secondo argomento al primo e ritornano il va-

```
$(SET'FRATELLI'(CARLO MICHELE))
(CARLO MICHELE)

$(SETQ SORELLE'(DANIELA ANASTASIA))
(DANIELA ANASTASIA)

$FRATELLI
(CARLO MICHELE)

$SORELLE
(DANIELA ANASTASIA)

$(SETQ RAGAZZE SORELLE)
(DANIELA ANASTASIA)

$RAGAZZE
(DANIELA ANASTASIA)
```

```
$(SETQ RAGAZZE' SORELLE)
SORELLE

$RAGAZZE
SORELLE

$SORELLE
(DANIELA ANASTASIA)

$DANIELA
:DANIELA ATOMO SLEGATO

$(SETQ UNO 1 DUE 2 TRE 3)
3

$DUE
2
```

lore dell'ultimo argomento. Questo assegnamento è conosciuto con il termine "side-effect", quasi tutte le funzioni del BASIC-LISP ritornano un valore, ma solo poche di esse hanno un side-effect. Per esempio la funzione (ADD 1 1) non ha side-effect; tale funzione ritorna il valore 2 ma nulla viene mutato, mentre (SETQ B'C) in (SETQ A(SETQB'C)) assegna C a B e ritorna C, il quale è assegnato ad A dal primo SETQ. Il risultato è quello di assegnare C sia ad A che a B.
La funzione EVAL provvede a un'elaborazione extra in aggiunta a quella già svolta; per esempio se, dalla tabella, si richiede la funzione (EVAL RAGAZZE) tale funzione ritorna (DANIELA ANASTASIA).

La funzione CAR ritorna il primo elemento di una lista: (CAR' (A B C)) dà come risultato A. La funzione CDR ritorna la lista di tutti gli elementi della lista eccetto il primo: (CDR' (A B C)) dà come risultato (B C).
La funzione DELETE prende un atomo di una lista come argomento e ritorna ai livelli superiori una copia della lista con tutte le occorrenze dell'atomo cancellate.
Nei linguaggi LISP completi si può cancellare qualsiasi s-espressione da una lista; il BASIC-LISP permette invece di cancellare solo atomi.
La funzione CONS prende una lista e un nuovo elemento e ritorna la lista con il nuovo elemento aggiunto.

```
$(EVAL RAGAZZE)
(DANIELA ANASTASIA)

$(DELETE'A'(A B C))
(B C)

$(SETQ A-LIST'(A(A B)C A)
(A(A B)C A)

$(DELETE'A A-LIST)
((A B)C)

$A-LIST
(A(A B)C A)

$(CONS'A'(B C))
(A B C)

$(CAR'((A B)(C D)))
(A B)

$(CDR'((A B)(C D)))
((C D))

$(CAR(CDR'(CARLO MICHELE
DANIELA ANASTASIA) MICHELE
```

```
**********************************************************************
*                                                                    *
*                          BASIC-LISP                                *
*                          by C.V.P.                                 *
*                                                                    *
**********************************************************************
```

```
338 ST(A+1)=TT
    :ST(A+2)=1
    :C=LM(E)
    :A=A+1
340 IF C<>N THEN PT(LM(C)-N)=ST(A)
    :A=A+1
    :C=PL(C)
    :GOTO340
345 A=A-ST(A)-1
    :TT=PL(E)
350 IF TT<>N THEN X=LM(TT)
    :GOSUB265
    :TT=PL(TT)
    :GOTO350
355 C=LM(E)
    :A=A-ST(A)
360 IF C<>N THEN PT(LM(C)-N)=ST(A)
    :A=A+1
    :C=PL(C)
    :GOTO360
365 A=A-ST(A)-1
    :GOTO330
500 C=0
    :IF AL=N THEN IFC=0 THEN A=A+1
    :ST(A)=0
    :GOTO510 ELSE 510
505 X=LM(AL)
    :GOSUB265
    :C=C+1
    :A=A+1
    :ST(A)=X
    :IF PL(AL)<>N THEN AL=PL(AL)
    :GOTO505
510 A=A+1
    :ST(A)=0
    :RETURN
4000 IF ST(A)<>1 THEN ER=2
     :GOTO25000
```

```
4005 A=A-1
     :IF ST(A)=N THEN X=N
     :A=A-1
     :RETURN
4006 IF ST(A)<2001 AND ST(A)>0
THEN X=LM(ST(A))
     :A=A-1
     :RETURN
4007 ER=4
     :GOTO 25000
4010 IF ST(A)<>1 THEN ER=2
     :GOTO25000
4015 A=A-1
     :IF ST(A)=N THEN X=N
     :A=A-1
     :RETURN
4017 IF ST(A)<2001 AND ST(A)>0
THEN X=PL(ST(A))
     :A=A-1
     :RETURN
4020 ER=4
     :GOTO25000
4025 IF ST(A)<>2 THEN ER=2
     :GOTO25000
4030 A=A-1
     :T2=AS
     :AS=PL(AS)
     :LM(T2)=ST(A-1)
     :PL(T2)=ST(A)
     :A=A-2
     :X=T2
     :RETURN
4035 IF ST(A)<>2 THEN ER=2
     :GOTO25000
4040 A=A-1
     :IF ST(A-1)<N OR ST(A-1)>4000
THEN ER=3
     :GOTO25000
4045 PT=(ST(A-1)-N)=ST(A)
     :A=A-2
     :RETURN
4050 X=LM(AL)
     :RETURN
4060 WW=0
     :FOR J=1 TO ST(A)
     :A=A-1
     :IF ST(A)>4000 AND ST(A)<5001
THEN WW=WW+FP(ST(A)-4000)
     :NEXT ELSE ER=5
     :GOTO25000
4065 A=A-1
     :GOSUB10000
     :RETURN
4070 IF ST(A)<>2 THEN ER=2
     :GOTO25000
4075 A=A-1
     :IF ST(A)<4001 OR ST(A)>5000
OR ST(A-1)<4001 OR ST(A-1)>5000
THEN ER=5
     :GOTO25000
4080 WW=FP(ST(A-1)-4000)-FP(ST(A)
-4000)
     :A=A-2
     :GOSUB10000
     :RETURN
4085 WW=1
     :FOR J=1 TO ST(A)
     :A=A-1
     :IF ST(A)>4000 AND ST(A)<5001

THEN WW=WW*FP(ST(A)-4000)
     :NEXT ELSE ER=5
     :GOTO25000
4090 A=A-1
     :GOSUB10000
     :RETURN
4095 IF ST(A)<>2 THEN ER=2
     :GOTO25000
4100 A=A-1
     :IF ST(A)<4001 OR ST(A)>5000
THEN ER=5
     :GOTO25000
4105 A=A-1
     :IF ST(A)<4001 OR ST(A)>5000
THEN ER=5
     :GOTO25000
4110 IF FP(ST(A+1)-4000)=0 THEN
ER=7
     :GOTO25000
4115 WW=FP(ST(A)-4000)/FP(ST(A+1)
-4000)
     :A=A-1
     :GOSUB10000
     :RETURN
4120 IF LM(AL)>=N AND LM(AL)<4000
THEN X=LM(PL(AL))
     :GOSUB265
     :PT(LM(AL)-N)=X ELSE ER=3
     :GOTO25000
4125 AL=PL(AL)
     :IF AL=N THEN ER=2
     :GOTO25000 ELSE AL=PL(AL)
     :IF AL=N THEN RETURN ELSE 4120
4130 IF ST(A)<>1 THEN ER=2
     :GOTO25000
4135 A=A-1
     :IF ST(A)>=N AND ST(A)>=N AND
ST(A)<5000 THEN X=T
     :A=A-1
     :RETURN ELSE X=N
```

```
     :A=A-1
     :RETURN
4150 C=LM(AL)
     :X=LM(C)
     :GOSUB265
     :IF X=N THEN AL=PL(AL)
     :IF AL=N THEN RETURN ELSE 4150
4155 AL=PL(C)
4160 X=LM(AL)
     :GOSUB265
     :IF PL(AL)=N THEN RETURN ELSE
AL=PL(AL)
     :GOTO 4160
4165 AL=PL(C)
4170 IF ST(A)<>2 THEN ER=2
     :GOTO25000
4175 A=A-1
     :IF ST(A-1) THEN X=T ELSE X=N
4180 A=A-2
     :RETURN
4190 PL(E)=AS
     :AS=E
     :X=LM(AL)
     :PT(X-N)=AL
     :IF LM(PL(AL))=N THEN LM(AL)=LB
     :RETURN ELSE IF LM(LM(PL(AL)))
=LB OR LM(LM(PL(AL)))=NB THEN PT
(X-N)=LM(PL(AL))
     :RETURN ELSE LM(AL)=LB
     :RETURN
4200 IF ST(A)=0 THEN X=N
     :A=A-1
     :RETURN ELSE X=AS
     :F=ST(A)
     :A=A-F
     :FOR J=1 TO F
     :IF ST(A)=0 THEN ER=4
     :GOTO25000 ELSE G=AS
     :AS=PL(AS)
     :LM(G)=ST(A)
     :A=A+1
     :NEXT
     :PL(G)=N
     :A=A-ST(A)-1
     :RETURN
4220 A=A-1
     :IF ST(A)=N THEN X=T ELSE X=N
4225 A=A-1
     :RETURN
4230 IF ST(A)<>1 THEN ER=2
     :GOTO25000 ELSE A=A-1
4235 IF ST(A)>4000 AND ST(A)<5000
THEN X=T ELSE X=N
4240 A=A-1
     :RETURN
4245 IF ST(A-1)>4000 AND ST(A-1)
<5000 THEN FOR J=1
TO ST(A)-1
     :A=A-1
     :IF ST(A-1)>4000 AND ST(A-1)
<5000 THEN IF FP(ST(A)-4000)
     (FP(ST(A-1)-4000) THEN X=T
     :NEXT
     :A=A-2
     :RETURN ELSE 4252 ELSE 4250
4250 ER=5
     :GOTO25000
4252 X=N
     :A=A-2
     :RETURN
4255 IF ST(A-1)>4000 AND ST(A-1)
<5000 THEN FOR J=1 TO ST(A)-1
     :A=A-1
     :IF ST(A-1)>4000 AND ST(A-1)
<5000 THEN IF FP(ST(A)-4000)
     )FP(ST(A-1)-4000) THEN X=T
     :NEXT
     :A=A-2
     :RETURN ELSE 4261 ELSE 4260
4260 ER=5
     :GOTO25000
4261 X=N
     :A=A-2
     :RETURN
4265 IF AL<>N THEN X=LM(AL)
     :GOSUB265
     :IF X<>N THEN AL=PL(AL)
     :GOTO4265
4270 RETURN
4275 IF AL<>N THEN X=LM(AL)
     :GOSUB265
     :IF X=N THEN AL=PL(AL)
     :GOTO4275
4280 RETURN
4285 X=E
     :RETURN
4290 IF ST(A)<>1 THEN ER=2
     :GOTO25000 ELSE A=A-1
     :X=ST(A)
     :GOSUB 210
     :X=0
     :A=A-1
     :RETURN
4295 IF ST(A)<>1 THEN ER=2
     :GOTO25000 ELSE A=A-1
     :X=ST(A)
     :GOSUB265
     :A=A-1
     :RETURN
4300 IF ST(A)<>1 THEN ER=2
     :GOTO25000
4305 A=A-1
```

Lezione 39

File su cassetta

Abbiamo visto nelle lezioni precedenti la costruzione di archivi di informazioni residenti in RAM.
La possibilità di costruire file in RAM è molto favorevole rispetto alla necessità di ricorrere a supporti di memoria secondaria, quali le cassette o i dischetti, che, come vedremo, pongono problemi di operabilità. Tuttavia, poiché la memoria di M10 ha una capacità limitata, in tutti i casi in cui la quantità di informazioni da registrare sia sufficientemente grande, siamo costretti a ricorrere a un supporto diverso.
In questa lezione vedremo come è possibile costruire un archivio residente su cassetta.
A questo scopo costruiremo un programma che chiede i nominativi e i numeri telefonici all'utente e li registra quindi in un archivio su cassetta e, quando il caricamento è completato, rilegge il file e ne stampa il contenuto su stampante.

Come collegare il computer a un registratore a cassetta

Prima di passare alla costruzione del programma, vediamo di che cosa abbiamo bisogno per disporre di un archivio su cassetta.
Innanzitutto dovremo disporre di un registratore a cassetta e del cavo opportuno per collegarlo a M10.
Il registratore scelto deve disporre delle tre prese:

- MIC, per ingresso del microfono;
- EAR, per uscita auricolare;
- REM, per controllo remoto

che verranno collegate alle tre spinette del cavo di collegamento.
Tale cavo dispone di un connettore rotondo, che deve essere inserito nella presa per l'uscita da M10, collocata sul retro ed evidenziata dalla parola TAPE.
Esso dispone inoltre, all'altro lato del filo di collegamento, di tre spinette, rispettivamente bianca, rossa e nera, che vanno così collegate:

- la spinetta bianca all'uscita EAR; si tratta della presa attraverso cui il registratore comunica dati a M10
- la spinetta nera alla presa REM; si tratta della presa attraverso cui M10 comanda la partenza e l'arresto del motore del registratore;
- la spinetta rossa alla presa MIC; si tratta della presa attraverso cui il registratore riceve dati da M10.

In questo modo possiamo sia inviare dati alla cassetta, sia riceverli.
Per registrare basterà a questo punto premere i tasti RECORD e PLAY, esattamente come nel caso di una normale registrazione di musica. Quando poi vorremo rileggere il file memorizzato, dovremo riavvolgere il nastro fino al suo punto di inizio e premere quindi il tasto PLAY, come nel caso normale di riascolto.
Per facilitare l'operazione di ricerca della posizione iniziale del file sarà utile ricorre-

re a un registratore che disponga anche del contatore di nastro: ci sarà così più facile ritrovare la posizione di inizio della registrazione.
Attenzione: la registrazione è molto delicata, e dovremo posizionare con cura (anche mediante vari tentativi) il volume (o anche il tono) del nostro registratore.
In genere una posizione a 3/4 tra il minimo e il massimo, verso il massimo, risulta adeguata.

Il programma

Come vedremo in seguito, le conoscenze di BASIC che ci consentiranno di costruire un file su cassetta sono in realtà molto semplici. L'aspetto che rende più complesso il problema è invece quello di operare correttamente sul registratore, ricordando di volta in volta le operazioni da eseguire.
Per questo motivo porremo cura, nella costruzione del programma, di inserire opportuni messaggi nei momenti più delicati per guidare l'utente nelle operazioni di predisposizione del registratore.
Veniamo dunque alla costruzione del programma. Poiché, da un punto di vista algoritmico, il problema è molto semplice e inoltre è già stato illustrato in precedenza, costruiremo il programma direttamente in BASIC, curando però di seguire un atteggiamento TOP DOWN sia nei passi di costruzione sia nella definizione della struttura del programma stesso.

```
10 ' Programma di caricamento e lettura di un
file su cassetta
20 GOSUB 100 ' Caricamento
30 GOSUB 1000 ' Messaggi all'utente
40 GOSUB 1500 ' Stampa
50 END
```

Incominciamo con il modulo principale:
Nel programma è stato inserito un modulo che provvederà a guidare l'utente nella fase più delicata dell'uso del registratore e cioè quella in cui il file è stato registrato ed è necessario riposizionare le testine del registratore sulla posizione iniziale di registrazione per poterlo rileggere.

Il modulo di caricamento

Riportiamo direttamente il modulo BASIC, che già abbiamo avuto occasione di presentare:

```
100 ' Caricamento archivio
110 CLS
120 PRINT "CARICAMENTO ARCHIVIO"
130 OPEN "CAS:AGENDA" FOR OUTPUT AS #1
140 ' Legge nominativo
150 INPUT "Nominativo";N$
160 ' While not stop do
```

```
170 IF N$="STOP" THEN GOTO 250
180 INPUT "Numero telefonico";T
190 PRINT #1,N$;",";T
200 INPUT "Nominativo";N$
210 GOTO 160
250 ' Endwhile
260 CLOSE #1
270 RETURN
```

Il modulo sopra riportato è del tutto simile a quello già usato per costruire un archivio residente in RAM. Le istruzioni impiegate sono esattamente le stesse, con una sola differenza rappresentata dall'istruzione OPEN. Infatti il nome del file in questo caso ha un prefisso differente, che specifica il tipo di supporto su cui il file risiede, in questo caso appunto la cassetta.

Il modulo di interfaccia utente

Se eseguiamo il modulo di caricamento precedente con un registratore collegato opportunamente, vedremo il nastro avanzare in corrispondenza delle operazioni di trasferimento dati da M10 al registratore.
A caricamento completato, per rileggere il file sarà necessario riportarsi nella posizione iniziale.
Ciò richiede evidentemente un tempo di attesa nel quale l'utente possa comandare il registratore affinché riavvolga il nastro fino alla posizione opportuna.
Cercheremo allora di costruire il programma in modo che:

- avverta l'utente che la registrazione è terminata e che per proseguire è necessario riavvolgere il nastro;
- resti in attesa fino a quando l'utente gli comunica che può proseguire.

A questo scopo il programma chiederà all'utente di comandare il proseguimento del programma digitando un tasto qualunque.
Useremo con questo obiettivo la funzione BASIC INKEY$, che restituisce il carattere digitato alla tastiera. Se nessun carattere viene fornito restituisce una stringa nulla. In tal modo potremo fermare il programma ripetendo la lettura da tastiera fino a quando viene fornito un carattere.
Vediamo dunque il programma:

```
1000 ' Messaggi all'utente
1010 CLS
1020 PRINT "RIAVVOLGERE IL NASTRO FINO A INIZIO
FILE"
1030 PRINT
1040 PRINT "PREMERE TASTO PLAY"
1050 PRINT
1060 PRINT "Battere un tasto qualunque per
continuare"
1070 ' Repeat
1080 LET A$=INKEY$
```

```
1090 IF A$="" THEN GOTO 1070
1100 ' Until viene digitato un tasto
1120 RETURN
```

Il modulo di stampa

Non resta ora che costruire il modulo finale che legge i nominativi e i numeri telefonici e li stampa.
Ancora una volta l'algoritmo è molto semplice, in quanto consiste della iterazione di lettura e stampa fino alla fine del file. Riportiamo perciò direttamente il modulo BASIC, che va osservato soprattutto per gli aspetti di linguaggio che consentono di effettuare le stampe:

```
1500 ' Modulo di stampa
1510 OPEN "CAS:AGENDA" FOR INPUT AS #1
1520 LPRINT "NOMINATIVO","NUMERO TELEFONICO"
1530 ' While not eof(AGENDA) do
1540 IF EOF(1) THEN GOTO 1600
1550 INPUT #1,N1$,T1
1560 LPRINT N1$,T1
1570 GOTO 1530
1600 ' Endwhile
1610 PRINT "fine stampa"
1620 CLOSE #1
1630 RETURN
```

Come si vede l'uso della stampante è estremamente semplice: comporta l'impiego dell'istruzione LPRINT con il formato dell'istruzione PRINT per la visualizzazione di dati da display. Il programma così scritto però funziona solo nel caso in cui la stampante usata sia collegata sull'interfaccia parallela di M10. Nel caso infatti in cui si ricorresse a una stampante con interfaccia seriale, collegata sulla linea RS232, i dati a essa inviati sarebbero visti più o meno come un file e richiederebbero conseguentemente un trattamento analogo a quello dei file su cassetta e su RAM.

Cosa abbiamo imparato

In questa lezione abbiamo visto:

- come si costruisce un archivio su cassetta;
- l'uso dell'istruzione OPEN per dichiarare un file residente su cassetta piuttosto che in RAM;
- come si opera sul registratore rispettivamente per registrare e leggere un file su cassetta;
- l'uso della funzione INKEY$ per "fermare" un programma fino a richiesta di proseguimento da parte dell'utente;
- l'uso dell'istruzione LPRINT per visualizzare dati su stampante.

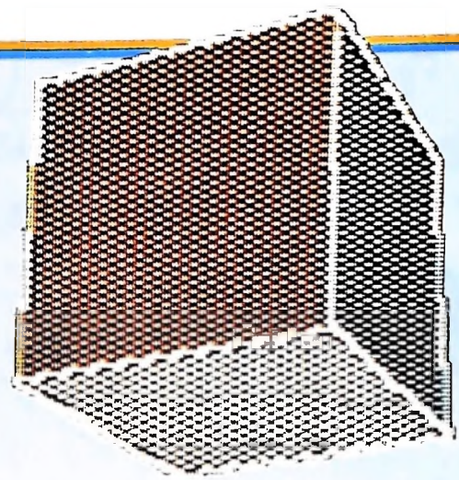
```
     :X=ST(A)
     :IF X)=N AND
X(5000 GOSUB 225
     :X=0
     :A=A-1
     :RETURN ELSE ER=3
     :GOTO25000
4310 IF ST(A)=0 OR ST(A-1)=N
THEN X=N
     :A=A-ST(A)-1
     :RETURN ELSE X=AS
     :FOR J=A-ST(A) TO A-1
     :Y=ST(J)
     :IF Y=0 OR Y)2000 AND Y()N
THEN ER=4
     :ST(A)=Y
     :GOTO25000
4312 IF Y()N THEN Z=AS
     :AS=PL(AS)
     :LM(Z)=LM(Y)
     :Y=PL(Y)
     :GOTO4312
4313 NEXT
4314 A=ST(A)-1
     :PL(Z)=N
     :RETURN
4315 IF ST(A)()2 THEN ER=2
     :GOTO25000
4320 A=A-1
     :IF ST(A)(4001 OR ST(A))5000
THEN ER=5
     :GOTO25000
4325 A=A-1
     :IF ST(A)(4001 OR ST(A))5000
THEN ER=5
     :GOTO25000
4330 WW=FP(ST(A)-4000)^FP(ST(A+1)
-4000)
     :GOSUB10000
     :A=A-1
     :RETURN
4399 IF LM(AL)(3000 OR LM(AL))4000
THEN ER=1
     :GOTO4447 ELSE T2=PT(LM(AL)-N)
     :IF T2)2000 OR T2=0 THEN ER=1
     :GOTO4447 ELSE IF LM(T2)()LB
AND LM(T2)()NB THEN ER=1
     :GOTO4447
4400 PRINT
     :PRINT"(DEFUN ";
     :X=LM(AL)
     :A$=CHR$(13)
     :GOSUB230
     :PRINT" (";
     :X=LMN(T2)
     :GOSUB230
     :PRINT" ";.
T2=PL(T2)
     :X=LM(T2)
     :J1=1
     :X1(J1)=X
     :GOSUB225
     :J=0
     :J2=0
4405 T2=PL(T2)
     :IF T2()N THEN PRINT"
     :PRINTTAB(3);"";
     :X1(J2)=-2
     :X=LM(T2)
     :GOSUB4410
     :GOTO440 ELSE PRINT"))";
     :X=0
     :RETURN
4410 IF X)4000 THEN PRINTFP(X-4000)
;CHR$(24),
     :RETURN
4415 IF X)=N THEN PRINTOB(X-N);
     :RETURN
4420 IFLM(X)=QU THEN PRINT"'";
     :X=LM(PL(X))
     :GOSUB225
     :RETURN
4425 J=J+1
     :T1(J)=X
     :D=LM(X)
     :B=D-N
     :IF B=40 OR B=41 OR B=31 THEN
4445 ELSE IF B()6 AND B()9 AND
B()10  AND B()14 AND B()20 AND B()
21 THEN PRINT"(";ELSE 4435
4430 X=T1(J)
     :X=LM(X)
     :GOSUB4410
     :X=T1(J)
     :J=J-1
     :X=PL(X)
     :IF X=N THEN PRINT")";
     :RETURN ELSE J=J+1
     :T1(J)=X
     :PRINT" ";
     :GOTO4430
4435 T1(J)=PL(T1(J))
     :PRINTTAB(X1(J2)+2)"(";
     :J2=J2+1
     :X1(J2)=POS(0)
     :X=0
     :GOSUB4415
     :PRINT""
4440 X=LM(T1(J))
     :PRINTTAB(X1(J2)+2);
     :GOSUB4410
     :X=T1(J)
```

```
     :J=J-1
     :X=PL(X)
     :IF X=N THEN J2=J2-1
     :PRINT")";
     :RETURN ELSE PRINT""
     :J=J+1
     :T1(J)=X
     :GOTO4440
4445 T1(J)=PL(T1(J)
     :PRINTTAB(X1(J2)+2)"(";
     :J2=J2+1
     :X1(J2)=POS(0)
     :X=D
     :GOSUB4415
     :PRINT" ";
     :X=LM(T1(J))
     :GOSUB4410
     :PRINT""
     :T1(J)=PL(T1(J))
     :GOTO4440
4447 E=0
     :LM(E)=LM(AL)
     :GOTO25000
4450 IF ST(A)<>2 THEN ER=2
     :GOTO25000 ELSE A=A-1
     :IF ST(A)>2000 THEN ER=4
     :GOTO25000 ELSE A=A-1
     :IF ST(A)<N OR ST(A)>4000
THEN ER=3
     :GOTO25000 ELSE J=ST(A+1)
     :D=ST(A)
     :X=AS
     :Z=N
4455 IF J<>N THEN IF LM(J)=D THEN
4460 ELSE Z=AS
     :AS=PL(AS)
     :LM(Z)=LM(J) ELSE IF Z=N
THEN X=N
     :RETURN ELSE PL(Z)=N
     :RETURN
4460 J=PL(J)
     :GOTO4455
4500 PRINT
     :PRINT"; premi ENTER per
cominciare";
     :GOSUB90
     :OPEN"lisp"FOR APPEND AS 1
4505 PRINT#1,FE;",";PE;",";AS;",";
     :FOR J=2 TO FE
     :PRINT#1,FP(J);",";
     :NEXT
     :FOR J= 49 TO PE
     :PRINT#1,OB(J);","
     :NEXT
 ;PT(J);",";
     :FOR J=1 TO AS
     :PRINT#1,LM(J);",";PL(J);","
     :NEXT
     :X=0
     :CLOSE 1
     :RETURN
4600 PRINT""
     :PRINT";premi ENTER per
cominciare";
     :GOSUB90
     :OPEN"lisp"FOR INPUT AS 2
4605 INPUT#2,FE,PE,AS
     :FOR J=2 TO FE
     :INPUT#2,FP(J)
     :NEXT
     :FOR J=49 TO PE
     :INPUT#2,OB(J),PT(J)
     :FOR J=1 TO AS
     :INPUT#2,LM(J),PL(J)
     :NEXT
     :X=0
     :CLOSE 2
     :RETURN
4650 X=0
     :A=A-1
     :IF PE>48 THEN PRINT""
     :PRINT"; ";OB(PE);" cancellato
dalla lista OB";
     :PT(PE)=0
     :OB(PE)=""
     :PE=PE-1
4655 RETURN
4700 TT=LM(AL)
     :E=PL(AL)
     :AL=E
4705 X=TT
     :GOSUB265
     :IF X<>N THEN AL=E
     :GOSUB4800
     :GOTO4705 ELSE RETURN
4750 TT=LM(AL)
     :E=PL(AL)
     :AL=E
4755 X=TT
     :GOSUB265
     :IF X=N THEN AL=E
     :GOSUB4800
     :GOTO4755 ELSE RETURN
4800 IF AL<>N THEN X=LM(AL)
     :GOSUB265
     :AL=PL(AL)
     :GOTO4800
4805 RETURN
10000 FOR J=1 TO FE
     :IF FP(J)=WW THEN 10010
```

```
10005 NEXT
      :FE=FE+1
      :FP(FE)=WW
      :X=FE+4000
      :RETURN
10010 X=J+4000
      :RETURN
25000 X=ST(A)
      :J1=1
      :X1(J)=X
      :IF A$()CHR$(13) THEN PRINT""
25001 A$=CHR$(13)
      :ON ER GOTO 25002,25003,25004,
25005,25006,25007,25008
25002 PRINT"; ";
      :X=LM(E)
      :GOSUB230
      :PRINT" nome funzione
non valido";
      :GOTO25050
25003 PRINT""
      :PRINT"; nr. di arsomenti
improprio in SUBR o NSUBR";
      :GOTO25050
25004 PRINT""
      :PRINT"; ";
      :GOSUB225
      :PRINT" atomo non valido";
      :GOTO25050
25005 PRINT""PRINT"; ";
      :GOSUB225
      :PRINT" lista non valida"
      :GOTO25050
25006 PRINT""
      :PRINT"; ";
      :GOSUB230
      :PRINT" numero non valido";
      :GOTO25050
25007 PRINT""
      :PRINT"; ";
      :X=V
      :GOSUB230
      :PRINT" atomo slegato";
      :GOTO 25050
25008 PRINT""
      :PRINT"; divisione per zero";
      :GOTO25050
25050 X=0
      :ON ERROR GOTO 25051
      :P=1/0
25051 PRINT""
      :RESUME30
26000 IF A$()CHR$(13)
THEN PRINT""
26001 IF PE)90
THEN PRINT"
lista OB piena"
      :PE=90
      :I$=""
      :GOTO27100
26005 IF FE)50 THEN PRINT";
FP pieno"
      :FE=50
      :I$=""
      :GOTO27100
26010 IF AS=N THEN PRINT";
memoria lista piena"
      :GOTO27100
26013 IF ERR/2+1=9
THEN IF A)350 OR
J1)15 OR J2)15 OR J)15
THEN PRINT";
stack overflow"
      :GOTO27000
26015 PRINT"; errore"
27000 RESUME30
27100 PRINT";
premi ENTER per
reinizializzare,
qualsiasi tasto
per continuare "
      :GOSUB90
      :IF A$=CHR$(13)
THEN PRINTCHR$(15)
      :RUN ELSE 27000

50000 DATANIL,3000,T,3001,SETQ,
6003,EQ,5012,CAR,5001,CDR,5002

50001 DATACOND,6004,DEFUN,6005,
ATOM,5011,LIST,5013,APPEND,5020

50002 DATAADD,5005,SUB,5006,MUL,
5009,CONS,5003,NUMBERP,5015

50003 DATAGREATERP,5016,LESSP,5017,
EVAL,5007,PRINTF,6009,AND,6007

50004 DATAOR,6008,DELETE,5021,SET,
5004,DIV,5010,NOT,5014,POWER,5019

50005 DATAPRINT,5008,PATOM,5018,
READ,6002,QUOTE,6001,LAMBDA,6006

50006 DATANLAMBDA,6006,SAVE,6010,
LOAD,6011,RPAREN,3044,LPAREN,3043

50007 DATAQT,3045,CR,3046,SP,3047,
DOWHILE,6013,DOUNTIL,6014,%,6012

50008 DATA(,0,),0,',0,CR,0," ",
0,FREE,4001
```

Immagini tridimensionali

L'innovazione forse più interessante apportata dai computer nel campo dell'immagine è rappresentata dalla possibilità di generare figure tridimensionali a partire da una base matematica.
In questo semplice esempio un cubo è visto da due angolazioni diverse. Con lo stesso principio, cioè partendo da un modello matematico, è possibile produrre immagini di straordinario realismo.
Tali immagini tuttavia richiedono computer dalla potenza di calcolo enorme. Per questa ragione essi vengono utilizzati esclusivamente in cinematografia e nella produzione di pubblicità televisiva.
In particolare alla Digital Production e alla Lucas Film sono state ottenute immagini di elevatissima qualità.



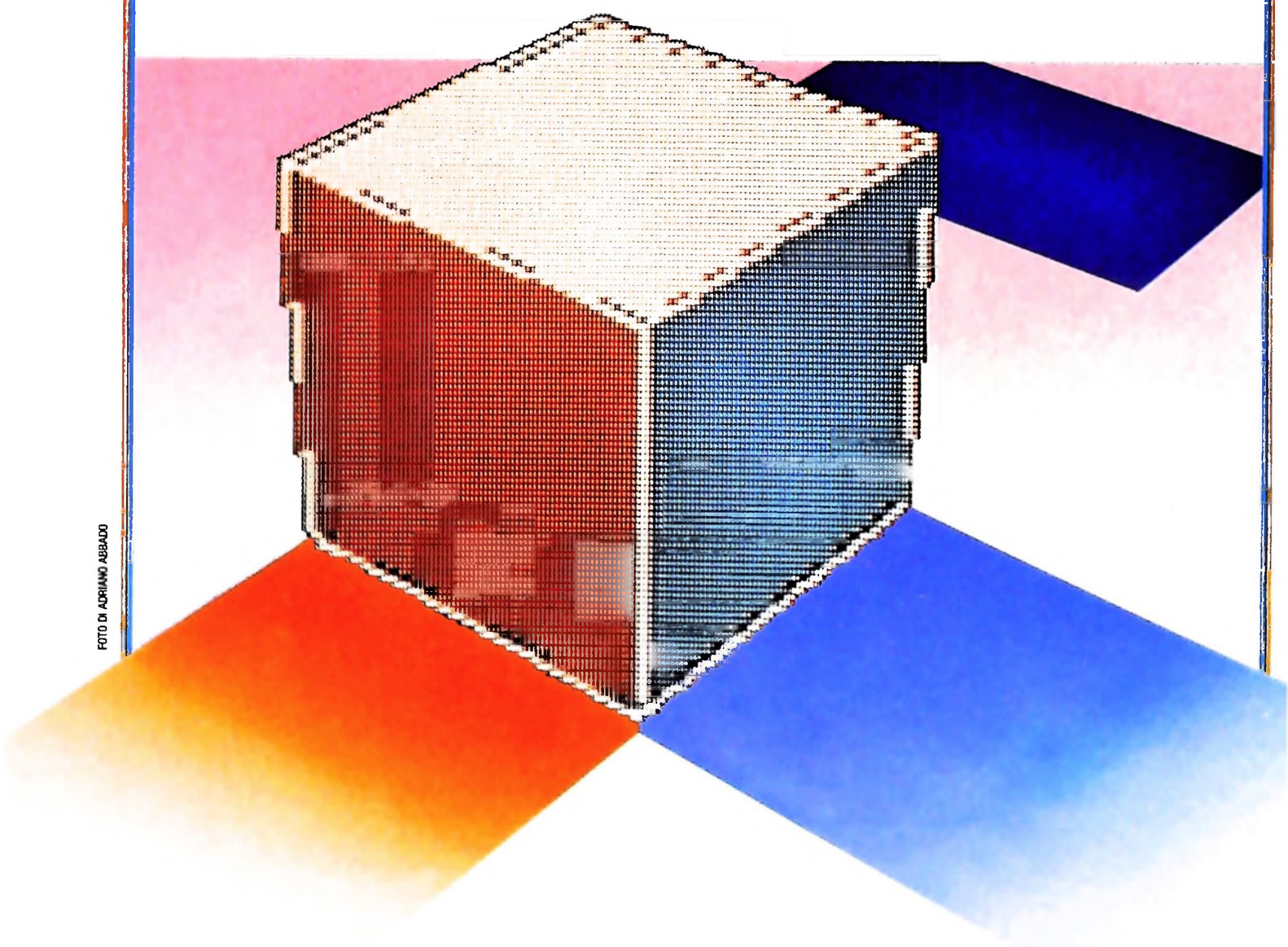


FOTO DI ADRIANO ABBADO

GLI ATTRIBUTI DEL SOFTWARE

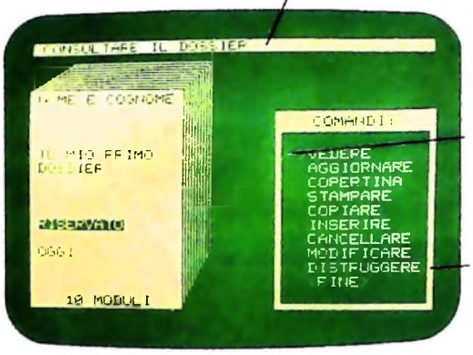
L'assenza di errori non è sufficiente per garantire la buona qualità di un programma.

Spesso si pensa che la qualità dei programmi sia legata alla correttezza, cioè all'assenza di errori: di fatto la qualità è una caratteristica che si articola in maniera molto più complessa, nella quale entrano a far parte anche aspetti di assistenza commerciale e di disponibilità di manuali di facile consultazione che contemplino ogni possibile esigenza di approfondimento specifico.
Tra i principali attributi che contribuiscono a determinare la qualità dei programmi, riportiamo quelli più significativi nella seguente tabella.

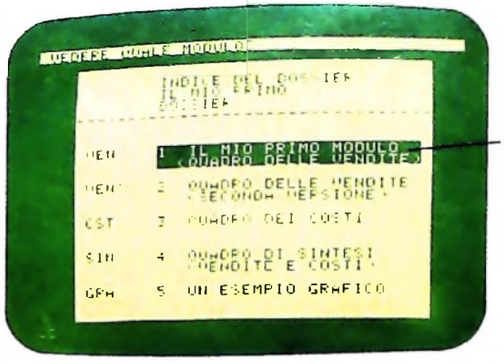
Gli attributi della qualità dei programmi

ATTRIBUTO	SIGNIFICATO
Correttezza	Consiste nell'assenza di errori nel programma, rispetto a quanto dichiarato nelle specifiche di prodotto e nei manuali d'uso; è rilevante per ogni prodotto software, ma in particolare per quelli destinati a un largo mercato, per cui non è pensabile una manutenzione dopo la vendita, con correzione degli errori per le copie già vendute.
Installabilità	È la facilità di operazioni per installare il prodotto software sul calcolatore che molto spesso richiede procedure complesse di personalizzazione del prodotto, a seconda delle caratteristiche di configurazione del calcolatore in uso (numero e tipo di periferiche, dimensioni di memoria ecc.); questo attributo è rilevante per i prodotti di larga diffusione, ove l'installazione deve essere fatta dall'utente finale stesso.
Configurabilità	Legato all'attributo precedente, consiste nella possibilità di adattarsi a un calcolatore nelle sue possibili differenti configurazioni, adattando le caratteristiche di prestazione alla periferia e alla memoria disponibili; così, per esempio, l'installazione di un programma può chiedere che venga usata una configurazione con video in bianco e nero, piuttosto che a colori se questo non è disponibile, eliminando i sottoprogrammi destinati alla gestione dei colori, con risparmio di memoria.
Adeguatezza	È uno degli attributi più scontati, ma difficile da ottenere; consiste nell'adeguatezza del prodotto software nel suo complesso, sia dal punto di vista delle funzioni che esso mette a disposizione all'utente, sia dal punto di vista delle procedure d'uso che impone: se queste sono in contrasto con le normali procedure di lavoro dell'ambiente che usa il prodotto, questo verrà rifiutato dagli utenti stessi. Per esempio, è sufficiente che i risultati richiesti da un addetto che si trova a uno sportello vengano forniti su stampa da un'apparecchiatura lontana dallo stesso, che costringa a un continuo andirivieni tra il posto di lavoro e la stampante.
Facilità d'uso	Si tratta di un attributo importante per quei prodotti che sono orientati a persone non specificamente competenti di informatica e di calcolatori; spesso è legato alla naturalezza dei comandi disponibili per invocare le varie funzioni, alla presenza di "menù" che guidino le scelte dell'utente, alla disponibilità di comandi di "help", cioè di comandi che forniscano istruzioni su come il prodotto deve essere usato ecc.

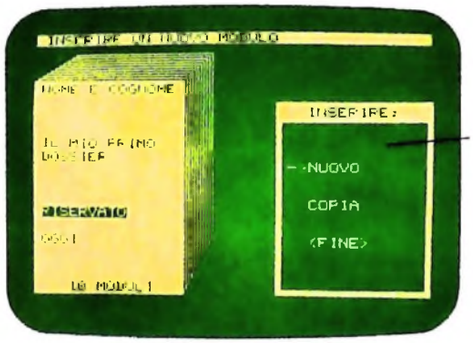
Facilità d'uso



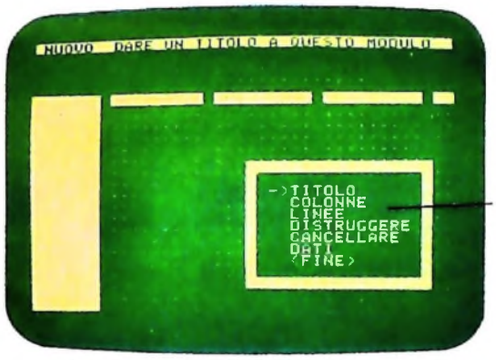
L'utente ha a disposizione un dossier, e sceglie di vederne il contenuto.



Tra i vari moduli che compongono il dossier, l'utente sceglie il primo.



Se l'utente chiede di inserire un nuovo modulo, un menù di secondo livello offre diverse scelte possibili.



Se il modulo da inserire è nuovo, un terzo menù compare, con nuove possibilità.

Un significativo esempio di "facilità d'uso", riferito al "foglio elettronico" DOSSIER® che permette a un utente di definire tabelle di dati per le quali vengono calcolati automaticamente totali, percentuali, medie ecc., anche con possibilità grafiche. L'utente "entrato nel prodotto" è completamente guidato al suo uso da menù e messaggi esplicativi, che gli rendono superfluo lo studio approfondito del prodotto e la consultazione di manuali.

Manutenibilità

È importante per il costruttore di software, avere la possibilità di "tenere a lungo in vita" un prodotto, potendone correggere errori scoperti dopo il rilascio, o migliorarne le caratteristiche , o ampliarne le funzioni offerte; in genere la manutenibilità è legata a diversi fattori, tra cui le caratteristiche architetturali del programma (che si presti a essere ampliato) e le caratteristiche dell'ambiente di produzione (documentazione aggiornata, programmi facilmente reperibili, test riusabili ecc.).

Prestazioni

Consistono nella misura della velocità di elaborazione. Le prestazioni richieste da un programma dipendono dal tipo di applicazione: è sufficiente che una stampa di un consuntivo contabile prodotto una volta all'anno sia disponibile da un giorno all'al-

tro, mentre è necessario che le interazioni tra utente e programma a un video forniscano la risposta in pochi secondi. Si pensi a un programma che controlli il funzionamento di una centrale nucleare che deve rispondere a specifici eventi nel tempo di pochi millisecondi, e anche meno. Quando si parla si prestazioni, si sottintendono spesso anche le caratteristiche di occupazione della memoria da parte del programma; anche in questo caso l'occupazione è più o meno rilevante a seconda dell'applicazione, essendo poco influente nel caso di grandi calcolatori dotati di "memorie virtuali", e molto influente, per esempio, su calcolatori a bordo di aerei o di satelliti artificiali.

Robustezza

Consiste nella capacità del programma di rispondere a un cattivo uso da parte dell'utente, senza che per questo vengano causati danni ai dati già memorizzati; si ottiene in genere enfatizzando i controlli su tutti i comandi forniti dall'utente.

Integrità

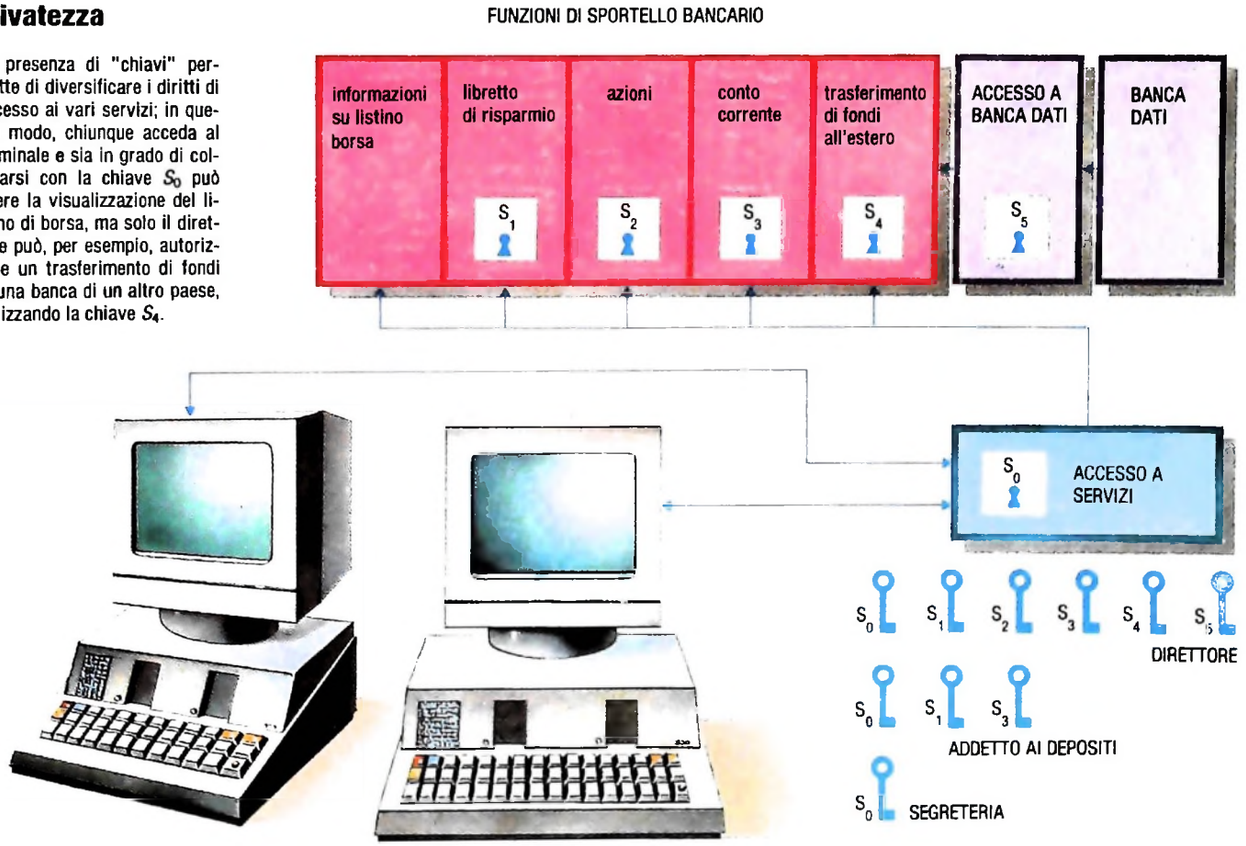
Questo attributo esprime la capacità di un programma di mantenere intatti e congruenti i propri dati, sia per eventi casuali (cadute di tensione o malfunzionamenti delle periferiche), che per errori nel programma (per esempio ponendo dei "filtri" che controllino la correttezza dei dati prima di ogni registrazione).

Privatezza

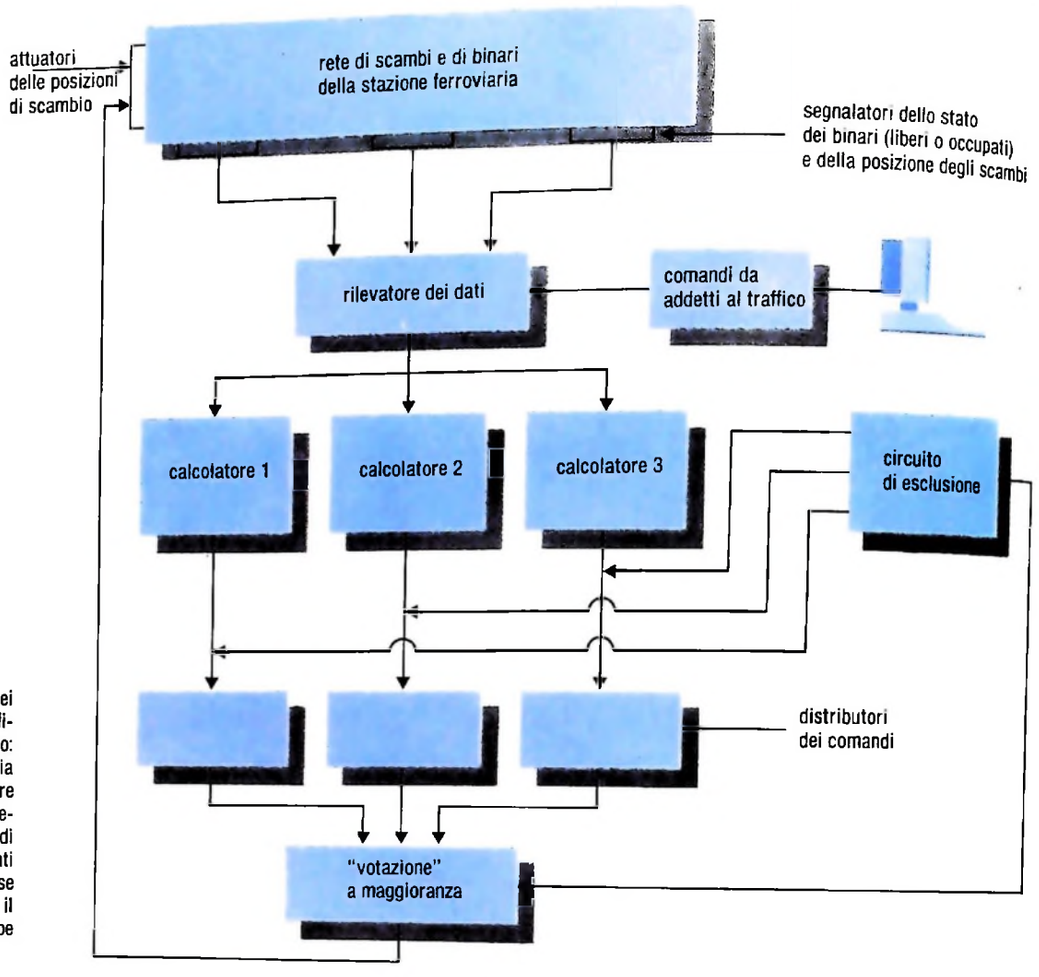
È la capacità di un programma di far accedere alle informazioni solo chi ne è autorizzato con l'uso di "parole d'ordine" (password) note solo all'utente autorizzato o attraverso dispositivi fisici analoghi (chiavi, tessere magnetiche ecc.). In altri casi, specialmente quando i dati si trovano inseriti in una rete di calcolatori, si usano tecniche aggiuntive, come la cifratura delle informazioni (i dati vengono cioè anagrammati o comunque codificati con speciali procedimenti che li rendono incomprensibili a chi non conosce la regola di decrittazione).

Privatezza

La presenza di "chiavi" permette di diversificare i diritti di accesso ai vari servizi; in questo modo, chiunque acceda al terminale e sia in grado di collegarsi con la chiave S_0 può avere la visualizzazione del listino di borsa, ma solo il direttore può, per esempio, autorizzare un trasferimento di fondi in una banca di un altro paese, utilizzando la chiave S_4.



Affidabilità



L'affidabilità è molto importante in quei casi in cui sia coinvolta la sicurezza fisica delle persone che usano l'impianto: nella figura, una stazione ferroviaria gestita da calcolatore mostra come tre unità svolgono le stesse attività di rilevazione dello stato della stazione e di attuazione dei comandi di movimenti della stessa; se una delle tre mostrasse risultati diversi da quelli delle altre, il circuito di "rotazione" provvederebbe alla sua esclusione.

Affidabilità

È la capacità di un sistema di non guastarsi ed è legata sia alle caratteristiche di correttezza del software che a quelle di robustezza dell'hardware. Viene misurata in termini di tempo medio che ci si aspetta tra due errori o guasti che causino l'arresto del sistema; tale misura viene indicata con la sigla MTBF (Mean Time Between Failure); l'affidabilità è ottenuta spesso mediante ridondanza hardware. Apparecchiature di controllo di processi delicati vedono il calcolatore triplicato. A ogni passo di elaborazione dello stesso programma sulle tre unità centrali uno speciale dispositivo confronta i tre risultati, occupandosi dell'esclusione di quel calcolatore che eventualmente fornisse un risultato difforme dagli altri due, e segnalando l'errore.

Disponibilità

Legato all'attributo precedente, misura la capacità di un sistema di fornire il servizio per cui è stato previsto senza interrompersi. Se un sistema si guasta in una sua qualche parte, è opportuno che tale parte venga esclusa, e con essa non siano più disponibili un certo insieme di funzioni, ma che il sistema continui a erogare i servizi essenziali, anche se con funzioni degradate. Ciò è importante non solo in quelle applicazioni il cui malfunzionamento può causare gravi danni, ma anche in quei casi in cui la manutenzione e la riparazione sono praticamente impossibili (come per i calcolatori a bordo di satelliti artificiali). La disponibilità è legata a diversi fattori, tra cui: la presenza di software "diagnostico" (programmi, che nei tempi morti di elaborazione esercitano i vari circuiti del sistema con dei dati prefissati, per accertarne il corretto funzionamento), la capacità di recupero (recovery, capacità di arrestare l'elaborazione per un guasto individuato e di mantenere integrità dei dati), la possibilità di ripartenza (restart, capacità di far ripartire il programma dopo le procedure di recovery, senza complesse procedure di inizializzazione del sistema complessivo).

UN NUOVO MODO DI USARE LA BANCA.

Conto corrente più

TANTI PENSIERI IN MENO CON IL CONTO CORRENTE "PIU'" DEL BANCO DI ROMA.

Essere cliente del Banco di Roma vuol dire anche essere titolari del conto corrente "più". Un conto corrente più rapido: perché già nella maggior parte delle nostre filiali trovate gli operatori di sportello che vi evitano le doppie file.

Più comodo, perché potete delegare a noi tutti i vostri pagamenti ricorrenti: dai mutui all'affitto, dalle utenze alle imposte.

Più pratico, perché consente l'utilizzo del sistema di prelievo automatico Bancomat e l'ottenimento della carta di credito.

Inoltre un servizio utilissimo, soprattutto per imprenditori e commercianti denominato "esito incassi", consente di avere comunicazione dell'eventuale insolvenza entro solo cinque giorni dalla scadenza. Una opportunità veramente speciale.

Più sicuro, perché con una minima spesa potrete assicurarvi contro furti e scippi mentre vi recate in banca o ne uscite.

Veniteci a trovare, ci conosceremo meglio.

BANCO DI ROMA
CONOSCIAMOCI MEGLIO.

Anche in leasing con Olivetti Leasing.

PERSONAL COMPUTER OLIVETTI M10

L'UFFICIO DA VIAGGIO

Olivetti M10 vuol dire disporre del proprio ufficio in una ventiquattrore. Perché M10 non solo produce, elabora, stampa e memorizza dati, testi e disegni, ma è anche capace di collegarsi via telefono per spedire o ricevere informazioni.

Qualunque professione sia la vostra, M10 è in grado, dovunque vi troviate, di offrirvi delle capacità di soluzione davvero molto grandi. M10: il più piccolo di una grande famiglia di personal.

olivetti

Per informazioni rivolgersi a negozi contrassegnati da "Olivetti M10 Punto di Vendita" o inviare questo coupon a: Olivetti, Divisione Personal Computer, Via Meravigli 12, 20123 Milano.

NOME/COGNOME

VIA/N

CAP/CITTÀ

TELEFONO