CADEL

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

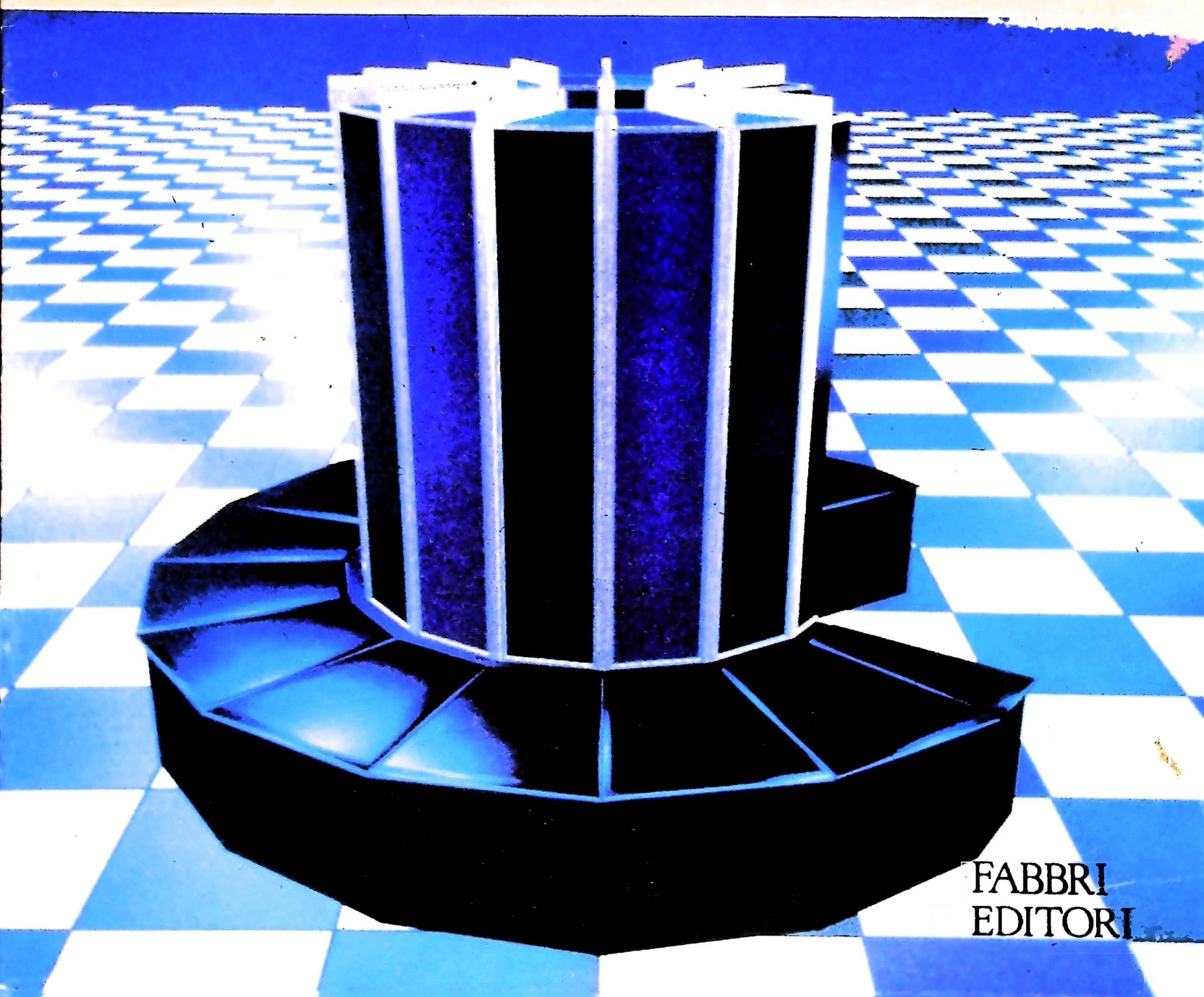
28 CORSO PRATICO COL COMPUTER

421800

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

F4 F5 F6 F7 F8

diretto da GIANNI DEGLI ANTONI

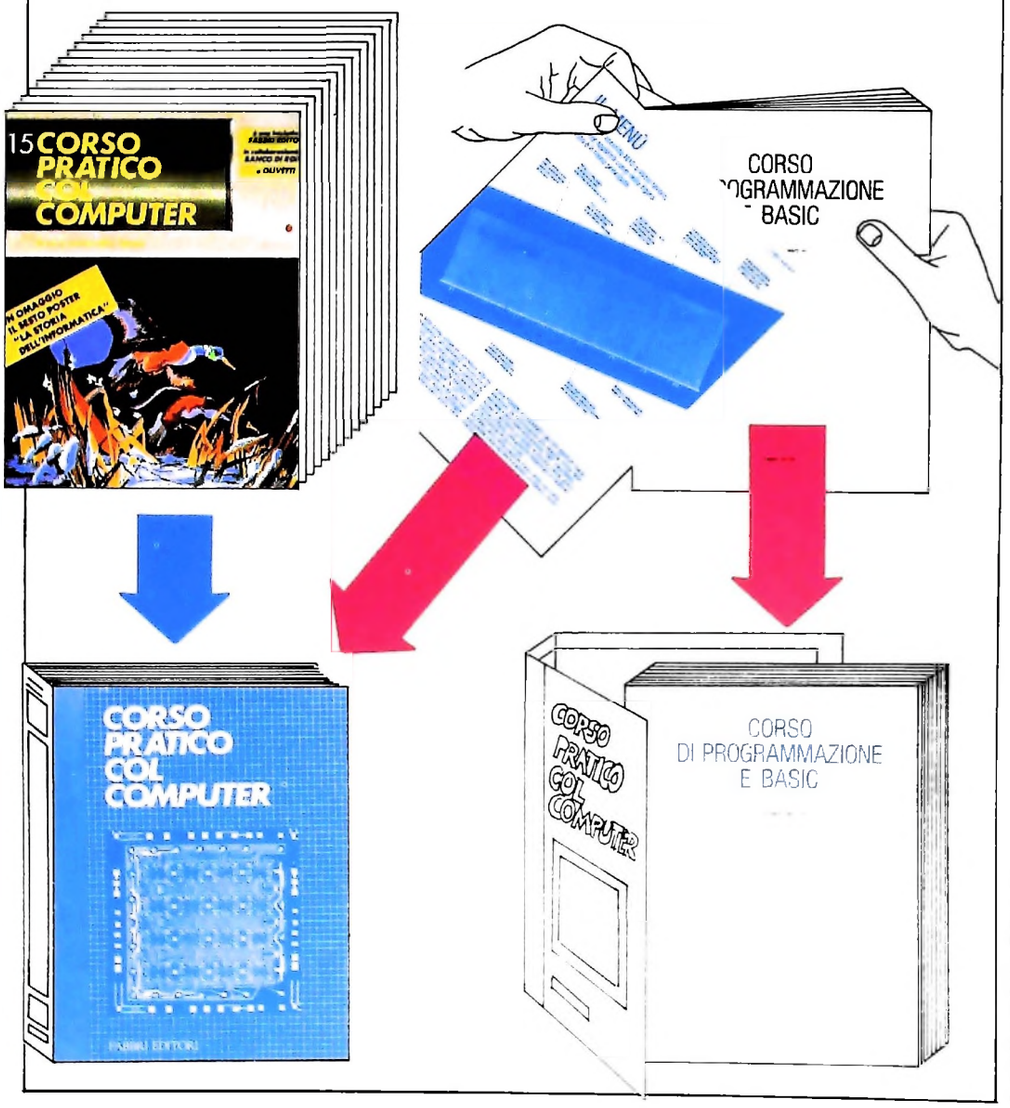


FABBRI
EDITORI

AVVISO AI LETTORI

Con questo fascicolo si conclude il secondo volume del "Corso pratico col computer". Per rilegare il volume si useranno:

- i fascicoli dal n. 15 al n. 28;
- occorrerà staccare l'inserto centrale di 4 pagine relativo al "Corso di programmazione e BASIC"; tutti gli inserti contenuti nei fascicoli, fino al n. 72, dovranno essere conservati per essere rilegati nel volume "Corso di Programmazione e BASIC", la cui copertina sarà messa in vendita con il fascicolo n. 30;
- i risguardi, inseriti nella copertina del volume;
- la copertina del volume, che è stata messa in vendita con il n. 22;
- ricordiamo che il frontespizio del volume fa parte del fascicolo n. 15 e il sommario del fascicolo n. 28.



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, ENNIO PROVERA,
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 BREGGÉ

Segretarie di Redazione
RENATA FRIGOLI
LUCIA MONTANARI

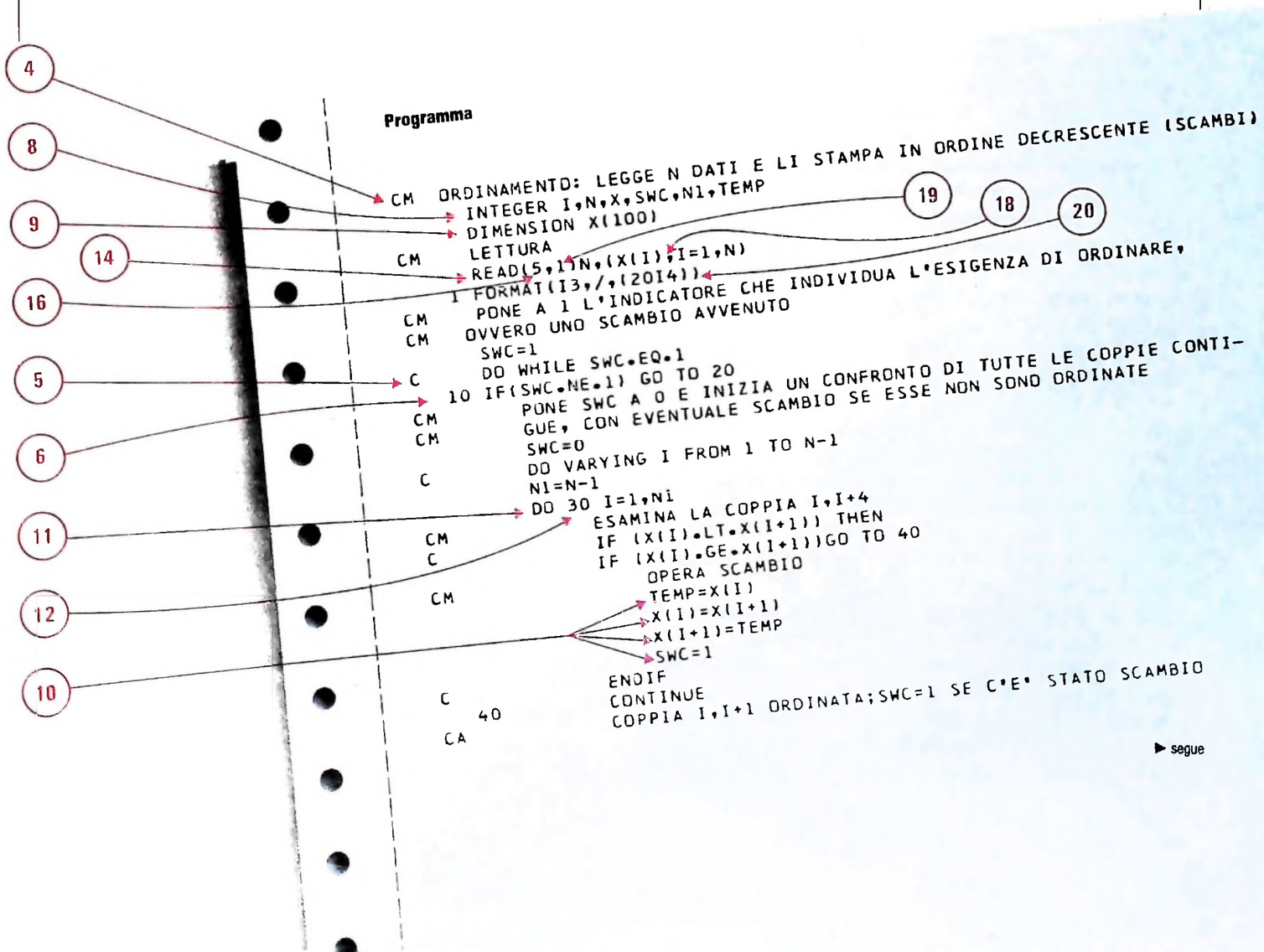
IL LINGUAGGIO FORTRAN

Un linguaggio adatto essenzialmente a obiettivi di calcolo che, pur risentendo dell'età, resta ancora un valido strumento per costruire programmi efficenti e veloci.

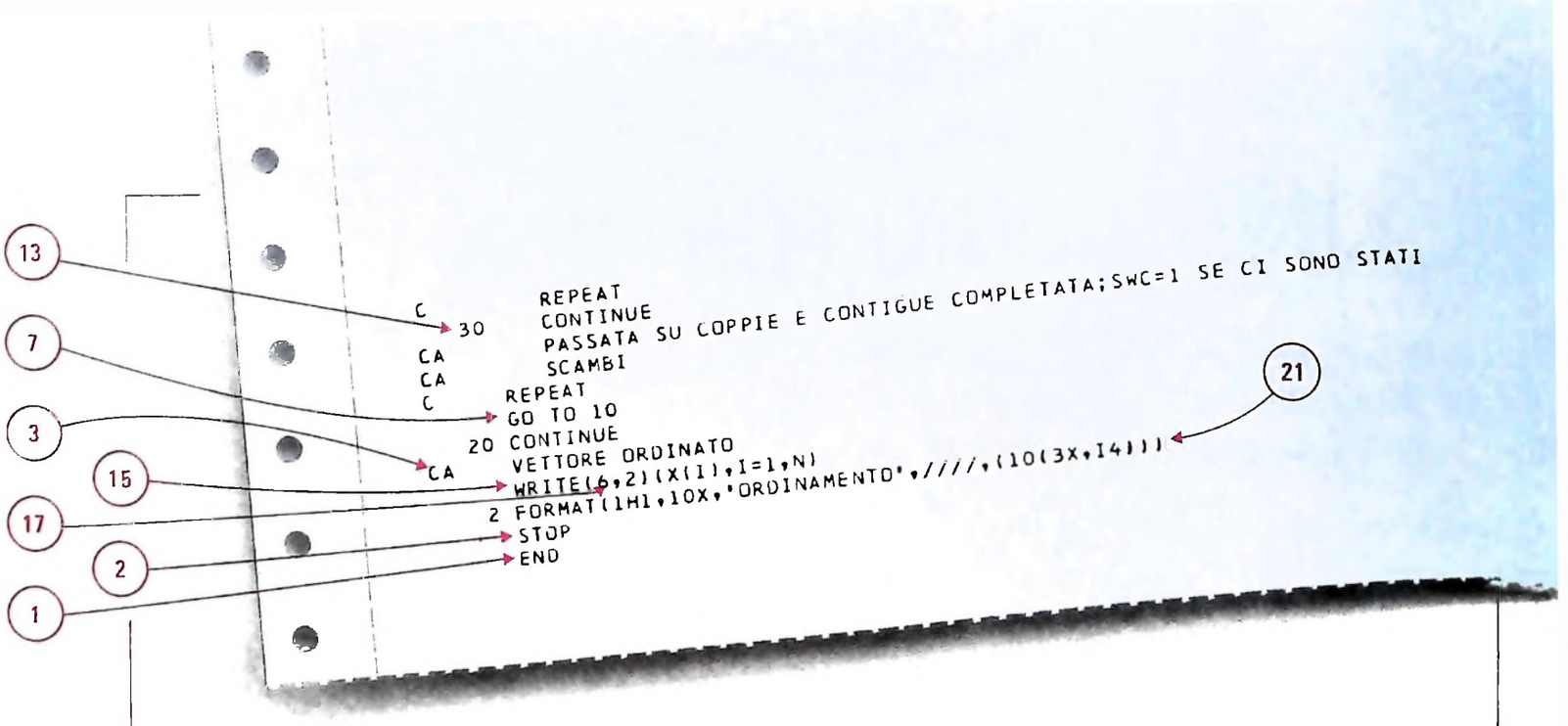
Il linguaggio FORTRAN deriva il suo nome da FORmulas TRANslation (cioè traduttore di formule matematiche), che ne evidenzia l'obiettivo di calcolo a cui è essenzialmente rivolto. Si tratta di uno dei linguaggi più anziani, essendo stato definito da J. W. Backus intorno alla metà degli anni Cinquanta. Dopo la prima versione successive evoluzioni sono state messe a punto, tanto da essere disponibili diversi compilatori per FORTRAN II, FORTRAN IV, FORTRAN V, FORTRAN 77, e così via.
Il linguaggio FORTRAN presenta molti compilatori di ottime qualità ed efficacia, così da ottenere programmi tradotti di eccezionale velocità di esecuzione.

Caratteristiche generali

Il FORTRAN è un linguaggio di programmazione con binding statico: l'insieme di variabili, il loro tipo, i sottoprogrammi dichiarati nel programma vengono predisposti dal compilatore prima dell'esecuzione e non possono essere modificati durante l'esecuzione del programma.
Prima di esaminare con maggiore dettaglio le varie caratteristiche del linguaggio, diamo una rapida occhiata al programma illustrato in figura, che effettua l'ordinamento degli elementi di un array per interscambi.
Innanzitutto osserviamo che il programma si presenta come



▶ segue



Un programma esemplificato in FORTRAN: provvede all'ordinamento degli elementi di una matrice mediante interscambi.

una successione di istruzioni non numerate che terminano con una END (1): quest'ultima frase serve a indicare al compilatore il termine fisico del programma, e non ha nulla a che vedere con il termine dell'esecuzione, che è invece invocato dall'istruzione STOP (2).
Il linguaggio FORTRAN non mette a disposizione strutture di controllo della programmazione strutturata, che nel programma sono state realizzate mediante istruzioni di controllo, che permettono, eventualmente al verificarsi di una condizione, di guidare l'esecuzione dirigendola a un'istruzione "etichettata" con un numero; una ricca collezione di commenti (qualunque riga iniziante per C, come in (3, 4), illustra tali realizzazioni; per esempio, la struttura WHILE in (5) è realizzata da un'istruzione di salto condizionato (6)

```
IF(SWC.NE.1)GO TO 20
```

che ha un significato equivalente all'istruzione BASIC

```
IF SWC<>1 THEN 20
```

e dall'istruzione di salto all'indietro GO TO 10 (7).
Si noti la dichiarativa INTEGER (8), equivalente all'istruzione DEFINT BASIC, che permette di dichiarare le variabili elencate come di tipo intero; in assenza di dichiarazioni specifiche, il compilatore assume come intere tutte le variabili che hanno un nome che inizia con una lettera compresa tra I e N (iniziali di "integer").
Si osservi anche l'istruzione DIMENSION (9), completamente equivalente alla DIM del BASIC (ma che permette solo allocazione statica, con il numero di elementi dell'array specificato nel programma).
Gli assegnamenti (10) sono molto simili a quelli che si possono riscontrare in BASIC, senza alcuna parola chiave.
L'unica struttura di controllo disponibile in FORTRAN è l'iterazione enumerativa, aperta dall'istruzione DO (11) che indica:

- l'etichetta dell'ultima istruzione del blocco da iterare (12)
- la variabile indice
- la coppia di estremi inferiore e superiore della iterazione
- un eventuale "passo" opzionale.

Spesso si preferisce "chiudere" un ciclo DO con una istruzione CONTINUE (13) che non ha alcun effetto, e che serve come punto di appoggio dell'"etichetta" numerica.
Il FORTRAN presenta istruzioni di entrata e uscita complesse e raffinate: accanto alle istruzioni vere e proprie di lettura (READ (14)) e di scrittura (WRITE (15)), che specificano anche la "periferica" da usare ("5" indica la periferica standard d'entrata (16) e "6" quella standard di uscita (17)) e la lista delle variabili da leggere o da visualizzare (con la possibilità di indicare implicitamente iterazioni enumerative nella lettura (18)), deve esser specificato il formato con cui i dati da leggere sono forniti, o quelli da visualizzare devono essere disposti; le istruzioni fanno infatti riferimento all'etichetta (19) di un'istruzione FORMAT (20), che fornisce proprio tali informazioni.
Per esempio il FORMAT in (21) specifica di saltare a nuova pagina (1H1), di lasciare quindi 10 spazi (10X), di visualizzare la stringa 'ORDINAMENTO', di andare a capo quattro volte (////), quindi di visualizzare 10 elementi, ciascuno dei quali sia formato da 3 spazi (3X) e da un intero allineato a destra su un campo di 4 spazi (I4).

L'esecuzione del programma con i dati

Dati

```
40
 10  15  30  18  65   1  19   6  14  20  31  62  86  90   7 100  77  88  17  36
 24  12  51 110   8 190 230 217 712   0 161 220  89 320  55  33 111 222 450 611
```

fornisce infatti i risultati:

Risultati

```
                ORDINAMENTO

  712   611   450   320   230   222   220   217   190   161
  111   110   100    90    89    88    86    77    65    62
   55    51    36    33    31    30    24    20    19    18
   17    15    14    12    10     8     7     6     1     0
```

Astrazione nei dati

Il FORTRAN non mette a disposizione dell'utente la possibilità di definire propri tipi di dati: sono disponibili esclusivamente INTEGER, REAL, COMPLEX (per numeri complessi), DOUBLE PRECISION (per rappresentare i valori con molte cifre significative), LOGICAL (per i valori "vero" e "falso") e i caratteri (spesso senza neppure avere la possibilità di dichiarazione di tipo CHARACTER, ma solo usando variabili di altri tipi e specificando che si tratta di caratteri nel FORMAT di lettura o scrittura).
Il FORTRAN mette a disposizione come unica struttura di dati l'array, spesso con il limite di 7 indici.
Non è prevista una struttura record.

Strutture di controllo

Il FORTRAN non mette a disposizione strutture di controllo diverse dal succitato DO; esse devono quindi essere "sintetizzate" mediante istruzioni di tipo IF e GOTO.
Solo nelle versioni più recenti, come nel FORTRAN 77, sono stati introdotti le strutture IF..THEN..ELSE e i vari tipi di iterazione.

Sottoprogrammi

Il FORTRAN permette la definizione di sottoprogrammi che possono essere scritti e compilati a parte (si ricordi l'importanza di questa possibilità per la riduzione dei tempi di compilazione !), nel qual caso hanno a disposizione variabili completamente diverse da quelle del programma principale.
Due sono i possibili modi con cui un programma e un sottoprogramma possono "comunicare":

- attraverso un'area dati comune, dichiarata tale con un'apposita istruzione COMMON (in tal caso il programma chiamante depositerà i valori da elaborare in tale area, e preleverà dalla stessa i risultati lasciati dal sottoprogramma)
- mediante parametri: il sottoprogramma indica, nella sua definizione, un insieme di variabili fittizie su cui intende operare; all'atto della chiamata, il programma principale specifica al sottoprogramma quali sono le variabili su cui effettivamente devono essere eseguite le operazioni. Di fatto il sottoprogramma riceve l'informazione degli "indirizzi" di memoria delle variabili su cui operare; tale modo di comunicare i parametri è detto *per referenza* o, con espressione inglese, *by reference*.

L'esempio della pagina seguente illustra un programma FORTRAN che usa un sottoprogramma, riportato sotto.

Conclusioni

Il FORTRAN è un linguaggio che risente dell'età: definito quando l'informatica non aveva ancora raggiunto l'attuale grado di maturità, mostra molte ineleganze e difficoltà per il programmatore; tuttavia resta ancora uno strumento molto diffuso grazie alla facilità con la quale consente di costruire programmi efficienti, veloci, in particolare laddove devono essere eseguiti calcoli complessi.

Programma principale

```
CM  LETTURA DI 3 N-UPLE A(I),B(I),C(I) E CALCOLO DEL MASSIMO DELLE LORO
CM  MEDIE
      INTEGER N,I
      REAL A,B,C,MA,MB,MC,MAX
      DIMENSION A(100),B(100),C(100)
      COMMON N
      READ(5,1)N
    1 FORMAT(I3)
      READ(5,2)(A(I),I=1,N),(B(I),I=1,N),(C(I),I=1,N)
    2 FORMAT(10F8.2)
CA    VETTORI A,B,C LETTI
CM    CALCOLO MEDIE
      CALL MEDIA(A,MA)
      CALL MEDIA(B,MB)
      CALL MEDIA(C,MC)
CA    MA,MB,MC CONTENGONO RISPETTIVAM. LE MEDIE DEI VALORI DI A,B E C
CM    CALCOLO MASSIMO TRA MA,MB,MC
      MAX=MA
C     IF MB.GT.MAX THEN MAX=MB ENDIF
      IF (MB.GT.MAX)MAX=MB
      IF  MC.GT.MAX THEN MAX=MC ENDIF
      IF (MC.GT.MAX)MAX=MC
      WRITE(6,3)MAX
    3 FORMAT(1X,'MASSIMO=',F8.2)
      STOP
      END
```

Sottoprogramma

```
      SUBROUTINE MEDIA(X,MX)
CM  CALCOLA IN MX IL VALOR MEDIO DELLE COMPONENTI DEL VETTORE X
      COMMON N
      DIMENSION X(100)
      INTEGER N,I
      REAL X,MX
CM    CALCOLO SOMMA X(I)
      MX=0.
C     DO VARYING I FROM 1 TO N
      DO 1 I=1,N
        MX=MX+X(I)
      REPEAT
    1 CONTINUE
CM    CALCOLO MEDIA
      MX=MX/N
      RETURN
      END
```

Un programma FORTRAN che richiama una subroutine, riportata in calce al programma stesso.

Definizioni implicite in FORTRAN

Il FORTRAN è molto permissivo: in genere non costringe il programmatore a dichiarare tutte le proprie intenzioni; per esempio, non è necessario dichiarare le variabili; esse vengono incontrate nel testo del programma e quindi allocate. Anche se ciò può sembrare un fatto di comodità, i rischi a cui si va incontro sono enormi.

È famoso il caso del fallimento di una missione spaziale americana, che perse un satellite nello spazio a causa di un errore di un programma di controllo; in esso, infatti, un ciclo del tipo

```
      DO 10 I=1,100
      ....
10    CONTINUE
```

che richiedeva di effettuare 100 iterazioni di una sequenza era stato erroneamente scritto come:

```
      DO 10 I=1.5
      .....
10    CONTINUE
```

Non essendo necessario dichiarare le variabili, e non essendo presi in considerazione gli spazi, l'istruzione DO è stata interpretata dal compilatore come l'assegnamento di 1.5 alla variabile di nome DO10I:

```
      DO10I =1.5
```

senza che nessun errore potesse venire segnalato.

Il danno causato da tale errore è da misurare in termini di molti miliardi.

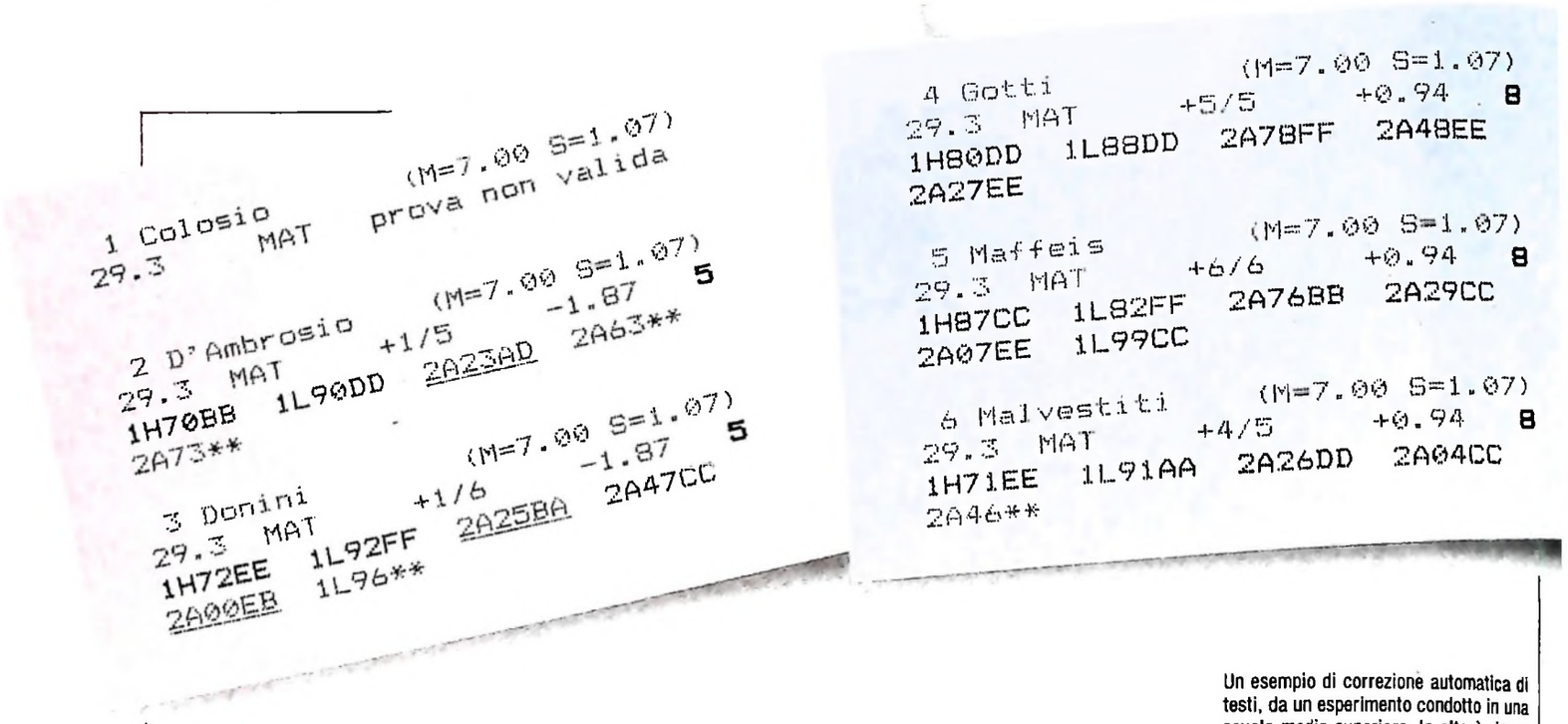
L'INFORMATICA E LA VALUTAZIONE SCOLASTICA

Uno strumento per rendere il più possibile automatico il processo di valutazione: rilevazione e analisi dei risultati.

Entro l'ambito di tutto il processo didattico, è abbastanza evidente la posizione speciale del problema della valutazione. Parecchi studi sono stati fatti in proposito e molte pagine sono state scritte.
Noi crediamo che l'impostazione corretta abbia le sue radici in quello che possiamo chiamare un "atteggiamento cibernetico". La cibernetica è la scienza che studia i sistemi (chiamati appunto "cibernetici") "orientati" e "adattivi", cioè orientati al raggiungimento di una data situazione o stato, e capaci di mantenerlo, adattandosi alle condizioni esterne (naturalmente entro certi limiti).

L'esempio più facile di sistema cibernetico è un impianto di riscaldamento dotato di termostato: tale impianto riesce a mantenere costante la temperatura di un dato ambiente nel modo noto, cioè il termostato rileva la temperatura attuale, la confronta con quella prevista e invia alla caldaia segnali di accensione o di spegnimento in base al confronto effettuato. Ebbene noi crediamo che la classe scolastica possa e debba essere considerata in un certo senso un sistema cibernetico, un sistema che evolve, o dovrebbe evolvere, lungo una linea ideale, quella dell'apprendimento.
La valutazione rappresenta appunto il momento cruciale del

Costruzione e gestione di una banca di test alla facoltà di Scienze dell'Università di Milano. I dati dalle schede, su cui gli studenti segnano le risposte, sono introdotti nell'elaboratore mediante un lettore ottico.



Un esempio di correzione automatica di testi, da un esperimento condotto in una scuola media superiore. In alto è riportata una serie di schede, qui accanto una scheda è esaminata in dettaglio.

confronto tra la situazione attuale e quella prevista lungo la linea dell'apprendimento, onde trarre le indicazioni per le decisioni conseguenti. in modo da mantenere l'evoluzione della classe costantemente lungo la linea stabilita.

Atteggiamento cibernetico e informatica

Chiaramente l'atteggiamento cibernetico è più ampio dell'informatica e più fondamentale.
L'informatica tuttavia può offrire ottimi strumenti per la sua realizzazione in una classe.
Soprattutto offre strumenti per rendere il processo della valutazione il più possibile automatico, cioè sganciato, è il caso di dirlo, dall'umore dell'insegnante, dai suoi difetti, dai suoi limiti inevitabili.
Diversi sono i tipi di impiego di un elaboratore applicato come sussidio per la valutazione.
Il più classico, diremmo il più tipico, è l'impiego per la rilevazione e l'analisi dei risultati.
Riportiamo un esempio sperimentato da parecchi anni in una scuola media superiore: la correzione automatica di test.
Gli esercizi sono scritti su fogli indipendenti, un esercizio per foglio, e distribuiti a mano dall'insegnante agli studenti: si tratta di esercizi tutti diversi tra loro con risposte chiuse a scelta multipla.
Ogni studente ne riceve da tre a sei a seconda delle disponibilità relative a un dato argomento.
L'allievo cerca di risolverli e consegna un foglio su cui ha segnato soltanto il numero di codice degli esercizi avuti e le relative risposte, che ritiene corrette e che indica con una lettera A, B, C, D, E, F.
L'allievo, se non è sicuro della soluzione, o comunque se non riesce a risolvere un esercizio, può anche lasciare una (o più) risposte in bianco.
L'insegnante introduce questi dati nell'elaboratore, il quale controlla le risposte, le valuta e attribuisce un voto allo studente in base a criteri stabiliti da chi ha curato il programma. Un esempio di questa correzione automatica è dato nell'illustrazione di questa doppia pagina.

Punteggi normalizzati

Vogliamo fermarci a commentare in particolare tre dati che risultano stampati nell'esempio che viene esaminato nell'illustrazione.
M è il valore della media dei voti della prova (e nel caso evidenziato a destra vale 7) e S è la "deviazione standard", cioè, secondo l'uso della statistica, la radice quadrata dello scarto quadratico medio dei voti singoli; in questo caso vale 1.07.
M alto o basso significa che in generale, cioè in media, la prova è andata più o meno bene.
S grande significa che i voti sono molto distribuiti lungo la scala, cioè che esistono parecchi voti alti ma anche parecchi voti bassi. S piccolo, al contrario, significa che i voti sono concentrati attorno al voto medio.
Questi due elementi servono per eseguire una trasformazione dei voti singoli in una distribuzione che potremmo chiamare "normalizzata".
Normalizzare una distribuzione di voti significa trasformarla in una nuova distribuzione che abbia una media e una deviazione standard stabilite a piacere.
Il punteggio −1.87 corrisponde al voto 5 espresso mediante "punti zeta", cioè una distribuzione che abbia M=0 e S=1. Si chiamano "punti zeta" perché la media è zero.

7 Micheletti (M=7.00 S=1.07)
29.3 MAT +2/5 -0.94 6
1H89EE 1L81AA 2A75** 2A06DD
2A28CD

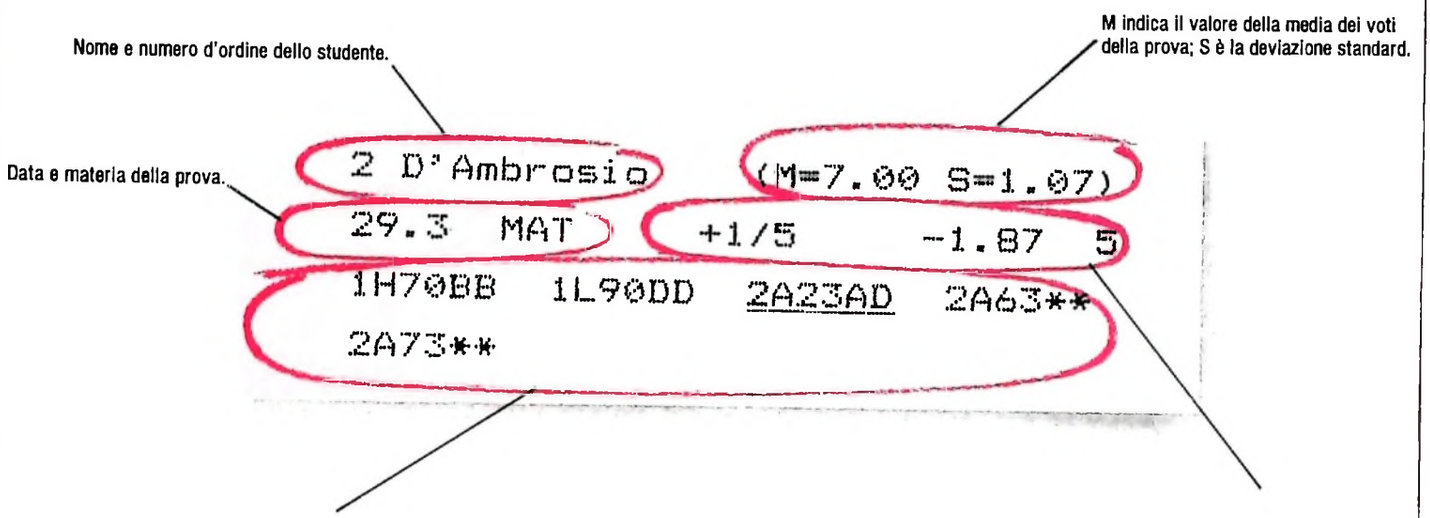
8 Micheli (M=7.00 S=1.07)
29.3 MAT +3/5 +0.00 7
1H88DD 1L84AA 2A71FF 2A61CC
2A09DA

9 Mistrini (M=7.00 S=1.07)
29.3 MAT +2/5 -0.94 6
1H78FF 1L83BB 2A42BB 2A70**
2A60ED

10 Morena (M=7.00 S=1.07)
29.3 MAT +3/5 +0.00 7
1L80DD 1H73DD 2A20BB 2A05CC
2A74EA

11 Paris (M=7.00 S=1.07)
29.3 MAT +6/6 +0.94 8
1H82CC 1L85EE 2A44DD 2A01DD
2A24FF 1L95FF

12 Pedrinelli (M=7.00 S=1.07)
29.3 MAT +4/5 +0.94 8
1L94CC 1H86DD 2A68** 2A22CC
2A45DD



La regola della trasformazione dei voti è la seguente:

$$(X - M)/S = (X' - M)/S'$$

dove:
X è il voto normale.
X' è il voto trasformato.
M è la media della distribuzione data.
M' è la media della nuova distribuzione.
S è la deviazione standard della distribuzione data.
S' è la deviazione standard della nuova distribuzione.
Dalla formula precedente si ricava:

$$X' = (X - M)*S'/S + M'$$

I punteggi normalizzati servono essenzialmente a due scopi: anzitutto danno una classificazione relativa del singolo rispetto al complesso della prova. Per esempio nella prova vista la media era 7 per cui il voto 6 corrisponde a un punteggio negativo (-0.94) cioè ad una prova mediocre, relativamente alla classe. Inoltre possono servire a confrontare con maggiore aderenza alla realtà alunni di classi diverse o di una stessa classe, ma di materie diverse, perché vengono eliminate le caratteristiche relative ai singoli docenti, cioè che uno sia più "largo" o più "stretto", e che usi solo due o tre voti oppure tutta la scala di 10.
L'uso del calcolatore permette di eseguire automaticamente i calcoli relativi alle trasformazioni offrendo così all'insegnante la possibilità, senza dover fare i conti, di avere nuovi e più ricchi elementi per una valutazione.

Applicazioni più sofisticate dell'elaboratore per la valutazione

Quello che abbiamo presentato e commentato piuttosto diffusamente è un primo esempio, tutto sommato abbastanza semplice, di impiego dell'elaboratore come strumento per la valutazione; le prestazioni però possono essere ulteriormente arricchite. Anzitutto l'elaboratore può essere usato per la costruzione e la gestione di una banca di test.
Ci riferiamo anche in questo caso a un'esperienza in atto, precisamente nell'Università di Milano alla Facoltà di Scienze. Nella memoria dell'elaboratore è accumulato un certo numero di esercizi, suddivisi per argomento.
Il docente interessato alla prova sceglie il tipo e il numero di esercizi da assegnare a ogni allievo e il relativo grado di difficoltà massimo, stabilendo quindi il numero di tabulati da stampare, uno per allievo.
Gli esercizi sono identici per ogni studente, però vengono stampati dall'elaboratore sui tabulati in ordine pseudocasuale in modo che, essendo in numero da 20 a 30, sia piuttosto difficile la copiatura tra allievi.
L'elaboratore contrassegna ciascun tabulato con un codice di identificazione che memorizza insieme con l'ordine degli esercizi stampati sul tabulato stesso.
Lo studente, durante lo svolgimento della prova, cerca di rispondere ai singoli quesiti scrivendo su una scheda, in opportuni simboli, il codice del tabulato e le risposte a ogni esercizio. Le risposte, anche in questo caso, sono chiuse e a scelta multipla.
Una "risposta", tra l'altro, corrisponde alla non risposta.
La valutazione avviene poi in modo automatico; i dati sulle schede sono introdotti nell'elaboratore attraverso un lettore ottico.
Una prima elaborazione fornisce per ogni esercizio la frequenza delle risposte corrette, quella delle risposte scorrette e quella delle risposte nulle (cioè delle domande cui non è stata data la risposta).
Questo permette di aggiornare il grado di difficoltà dei singoli esercizi e più in generale fornisce l'andamento della prova dal punto di vista dei particolari esercizi.
L'elaboratore fornisce pure l'entità dei punteggi grezzi dei singoli allievi: un punto per ogni esercizio corretto, più un punto per ogni esercizio corretto con grado di difficoltà massimo, meno un punto per ogni esercizio scorretto con grado di difficoltà minimo. L'operatore, cioè il docente, stabilisce quindi il voto massimo e il voto minimo in trentesimi e l'elaboratore trasforma i punteggi grezzi in voti normalizzati, stampando per ogni studente il numero di risposte corrette, scorrette o nulle e il voto.
Un altro miglioramento ottenibile da un elaboratore è la variazione pseudocasuale dei dati di problemi numerici.
Gli esercizi sono memorizzati come "testi base"; l'elaboratore, quando deve assegnarli, sceglie anzitutto la variabile o le variabili da considerare come incognite e da calcolare, quindi attribuisce, alle altre variabili, dati numerici costruiti con opportuni algoritmi in modo pseudocasuale entro intervalli stabiliti.
Calcola poi la soluzione numerica del problema e la stampa, sempre in ordine pseudocasuale, nel tabulato dello studente accanto ad altre soluzioni errate, che differiscono in modo opportuno da quella corretta.
Con questo sistema è possibile praticamente assegnare lo stesso tipo di esercizi ad un'intera classe, mentre la variazione dei dati numerici ostacola la copiatura tra allievi.

Uso della strategia tutoriale nel problema della valutazione

Esiste un'ulteriore possibilità di applicazione dell'elaboratore come strumento in aiuto alla valutazione e precisamente la conduzione diretta di una prova di valutazione.
L'allievo ha un rapporto immediato con l'elaboratore, nel quale è impiegata una strategia molto simile a quella tutoriale, con domande, risposte e relative ramificazioni.
L'elaboratore presenta una serie di esercizi o di quesiti ai quali lo studente deve fornire le risposte.
A seconda del valore delle risposte sono previste delle ramificazioni che differenziano l'andamento successivo della prova. Ciò significa che esistono esercizi per così dire riassuntivi, rispondendo correttamente ai quali l'allievo effettua un percorso abbreviato.
Se al contrario lo studente sbaglia, le difficoltà vengono suddivise in esercizi più elementari.
Esistono naturalmente altre possibilità, come per esempio il memorizzare le risposte dello studente con automatizzazione della correzione, la possibilità di fornire dei suggerimenti individualizzati per il recupero degli errori in rapporto ai risultati degli esercizi.
Anzi uno sviluppo completo di questa strategia consente di associare sempre, allo svolgimento dei test, le parti teoriche per il ricupero immediato degli errori.
Si tratta di una strategia ricca e molto interessante.

Conclusioni

Come si vede le possibilità aperte dall'informatica nel campo della valutazione sono notevoli.
Ma ne resta una che riteniamo fondamentale: l'autovalutazione. Meccanizzare la valutazione non significa soltanto alleggerire il lavoro dell'insegnante e consentire al tempo stesso un approfondimento delle conoscenze relative, significa anche rendere lo studente più partecipe del processo didattico, dandogli la sensazione diretta, e il meno possibile traumatica, dello stato in cui si trova, momento per momento.
Mediante un corretto impiego dell'elaboratore lo studente può essere condotto ad un'autovalutazione, cioè ad una analisi autonoma dei risultati raggiunti e di quelli che gli mancano, cioè del suo grado di apprendimento.
Ma questo è tutto un nuovo modo di guardare alla didattica, un modo totalmente rivoluzionario, al quale dedicheremo il prossimo e ultimo articolo relativo alla scuola.

Esempi di uso di strutture a lista

L'uso di strutture a lista è particolarmente indicato nella soluzione di problemi che richiedono frequenti aggiornamenti di dati mantenendo l'ordinamento degli stessi.
In tali casi l'organizzazione dei dati in una lista consente di:
- inserire nuovi dati con semplici operazioni di accodamento, simulando l'ordinamento da un punto di vista logico tramite i valori dei puntatori;
- eliminare dati rendendoli "irraggiungibili" da qualsiasi puntatore.

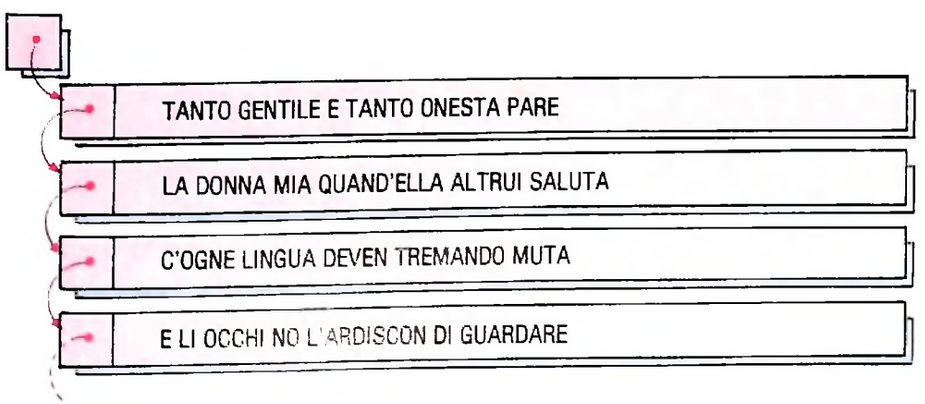
In breve quello che si realizza è una organizzazione dei dati "logica" che non corrisponde a quella fisica.
Le operazioni richieste a tale scopo sono molto semplici, algoritmicamente, e soprattutto molto veloci rispetto alle operazioni necessarie per mantenere i dati "fisicamente" ordinati.
Un impiego classico di strutture a lista è nella realizzazione di editor, ovvero di strumenti per la gestione di testi, che consentono quindi di:
- inserire nuove stringhe all'interno del testo;
- cancellare porzioni di testo;
- sostituire porzioni di testo con altre differenti;
- aggiungere in coda parti di testo a quelle già esistenti.

Se per esempio abbiamo costruito un testo del tipo:

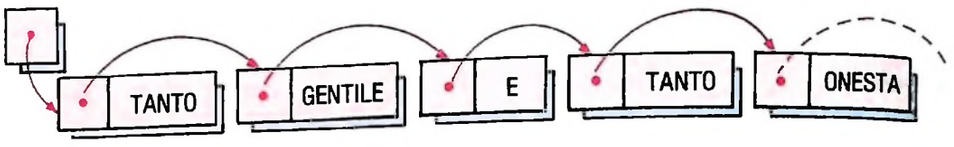
Tanto gentile e tanto onesta pare
la donna mia quand'ella altrui saluta
c'ogne lingua deven tremando muta
e li occhi no l'ardiscon di guardare

possiamo pensare alla possibilità di correggere errori eventualmente commessi nella digitazione, così come possiamo desiderare di completare il testo.
Possiamo allora pensare di organizzare il testo secondo un "modello di descrizione" dello stesso che lo veda come:
- un insieme di "righe";
- un insieme di parole;
- un insieme di caratteri.

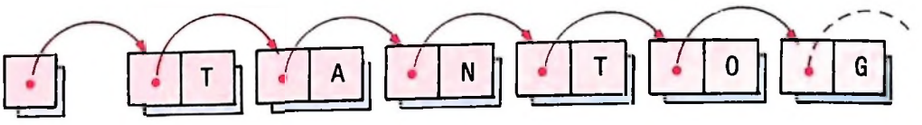
A seconda del modello scelto possiamo costruire un editor che organizzi il testo in una lista i cui elementi contengano nel campo "informazione" rispettivamente le righe, le parole o i caratteri che lo costituiscono.
Otterremo così la seguente struttura dati nel caso dell'editor a linea:



che consente di operare su linee intere, consente cioè di inserire, cancellare, sostituire una riga alla volta.
Se vogliamo invece poter operare su parole organizzeremo la struttura dati nel modo seguente:



E ancora per operare a livello del singolo carattere frammenteremo il testo in una lista di caratteri:



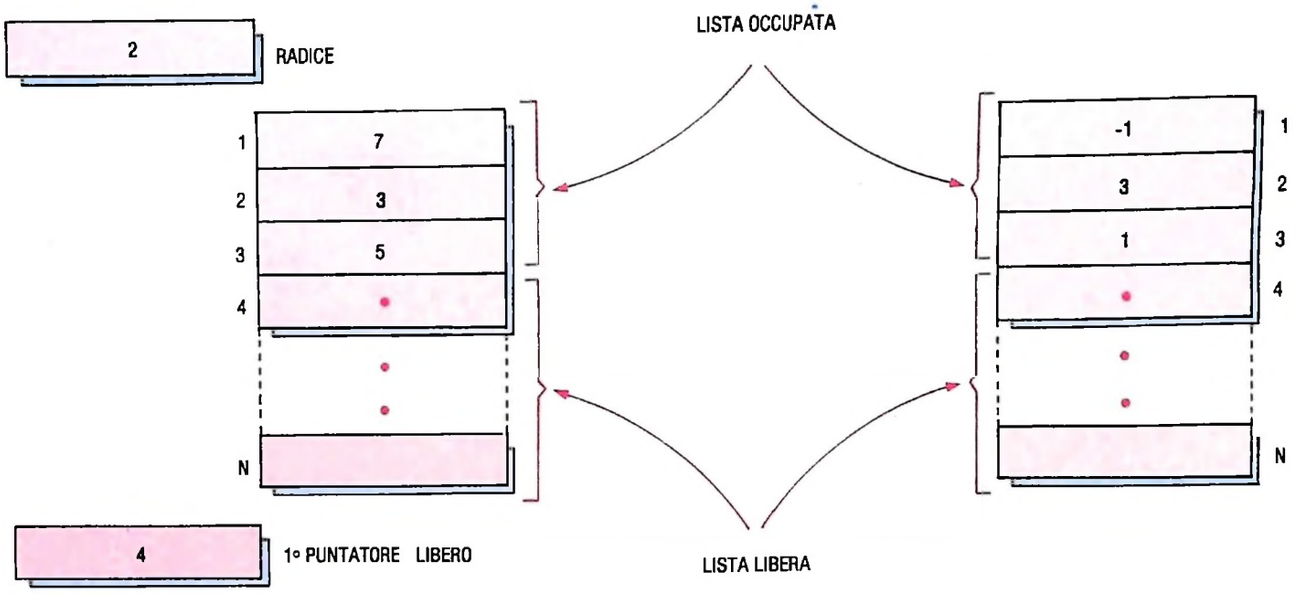
Tecniche di gestione della memoria nel caso di liste simulate

Come abbiamo visto l'uso di liste è particolarmente indicato per problemi di aggiornamento di dati, che comportino inserimenti di dati nuovi e cancellazione di quelli esistenti.
La soluzione di tali problemi è facilitata quando si possa ricorrere a un linguaggio che consenta un'allocazione dinamica delle variabili. In questo caso infatti l'uso della memoria disponibile è ottimizzato grazie alla possibilità di allocare e rilasciare la memoria a richiesta.
Ogni nuova informazione inserita infatti richiede uno spazio di memoria che viene sottratto alla memoria "libera" disponibile, così come ogni operazione di cancellazione produce una restituzione alla memoria "libera" dello spazio precedentemente occupato dall'informazione eliminata.
Nel caso di linguaggi che non ammettano la gestione di variabili dinamiche, la gestione della memoria resta a carico del programmatore.
Cerchiamo di chiarire meglio il problema. Immaginiamo di avere costruito un programma che consenta di inserire ed eliminare dati da una lista simulata con un array. Tale array da un punto di vista logico può essere visto come un'area di memoria suddivisa in due porzioni che rappresentano rispettivamente:
- la lista dei dati inseriti;
- la memoria "libera" disponibile per eventuali successivi inserimenti.

L'inizio della memoria libera è individuato da un puntatore che indica il primo elemento libero dell'array.
A ogni nuovo inserimento, la nuova informazione viene inserita nella prima locazione di memoria libera e il puntatore a essa viene aggiornato.
Consideriamo adesso il problema dell'eliminazione di dati dalla lista. Tale operazione è in realtà molto più semplice di quella di inserimento: supponiamo per esempio di cancellare il 5 dalla lista della figura, basterà far puntare l'indice associato al 3 al 7 invece che al 5. In realtà così il dato non viene cancellato fisicamente, ma viene reso logicamente inesistente.

Riprendiamo la lista completa del programma che effettua inserimenti ordinati in una lista.

```
10 INPUT "Quanti valori";N
20 DIM V(N),P(N)
30 GOSUB 100 'Inserimento
40 GOSUB 1000 ' Ordinamento
50 END
60 '
70 '
100 ' Inserimento
102 ' Crea radice
105 INPUT "Valore";V(1)
107 LET P(1)=-1
110 LET R=1 ' Valore iniziale puntatore radice
115 LET I=2
120 ' While I<=N do
123 IF I>N GOTO 165
125 ' Begin
130 INPUT "Valore";V(I)
140 GOSUB 250 'Posizionamento indici
150 LET I=I+1
155 ' End
160 GOTO 120
165 '
170 RETURN
180 '
190 '
250 'Posizionamento indici
255 LET J=R
260 ' Exec until intesta,ins,inscoda
270 ' Event instesta if V(I)<V(J) and J=R
280 IF V(I)<V(J) AND J=R GOTO 330
285 ' Event ins if V(I)<V(J)
290 IF V(I)<V(J) GOTO 400
295 'Event inscoda if P(J)=-1
298 IF P(J)=-1 GOTO 370
300 LET K=J
305 LET J=P(J)
310 ' Thencase
320 GOTO 260
325 ' Thencase
330 ' When instesta
340 LET R=I
350 LET P(I)=J
360 GOTO 450
370 ' When inscoda
375 LET P(I)=P(J)
380 LET P(J)=I
390 GOTO 450
400 ' When ins
410 LET P(I)=J
420 LET P(K)=I
450 ' Endexec
460 RETURN
570 '
580 '
1000 ' Scansione per ordinamento
1010 IF R=-1 GOTO 1050
1020 LET J=R
1025 ' Repeat
1030 PRINT V(J)
1040 LET J=P(J)
1050 ' Until J=-1
1060 IF J<>-1 GOTO 1025
1070 RETURN
```



In questo modo però avremo una porzione di memoria disponibile sempre inferiore rispetto al numero di informazioni "utili" presenti nella lista: alcune locazioni di memoria saranno infatti occupate da informazioni logicamente cancellate.
Una soluzione a tale problema è data dalle tecniche dette di GARBAGE COLLECTION che indicano letteralmente il recupero della "spazzatura".
Il concetto di fondo è che ogni elemento di array che contenga un valore logicamente cancellato venga restituito alla memoria libera, in modo da poter essere successivamente riutilizzato.
In questo caso l'operazione di eliminazione di un elemento dalla lista risulterà più complesso di come era stato precedentemente descritto. L'elemento eliminato deve infatti risultare il primo elemento dell'array disponibile. Per fare ciò dovremo:

- aggiornare il puntatore alla memoria libera facendolo puntare all'elemento eliminato;
- posizionare l'indice di tale elemento perché punti a quello che era precedentemente il primo libero.

Abbiamo così evidenziato un'ulteriore esigenza: la memoria libera infatti dovrà essere anch'essa organizzata a lista e la suddivisione mostrata dalla figura sarà ancora una volta l'immagine "logica" dell'array profondamente differente da quella fisica.

Cosa abbiamo imparato

In questa lezione abbiamo visto:

- esempi di utilizzo di strutture a lista per la realizzazione di editor;
- tecniche di gestione della memoria (GARBAGE COLLECTION) con l'uso di liste.

MODI DI TRASMISSIONE

Esaminiamo le caratteristiche della trasmissione parallela e della trasmissione seriale: quest'ultima è la più utilizzata.

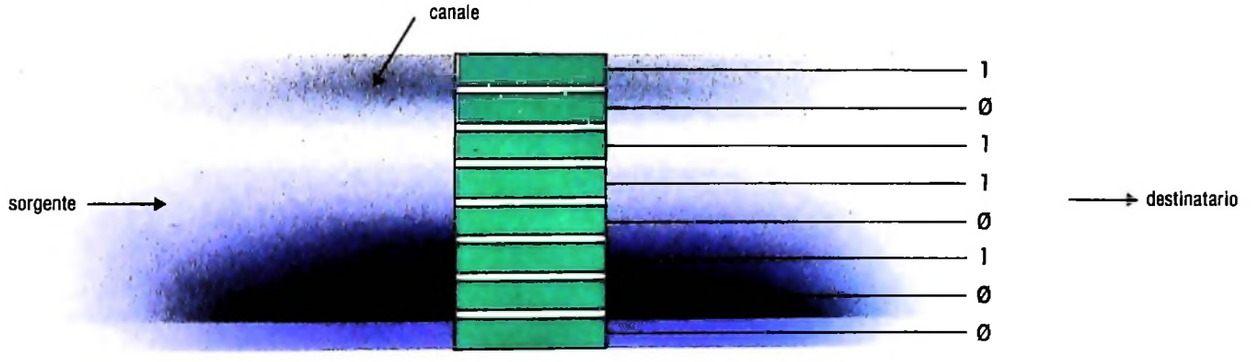
La trasmissione dati avviene in forma binaria, normalmente affidata alle linee di comunicazione in modo *parallelo* o in modo *seriale*.

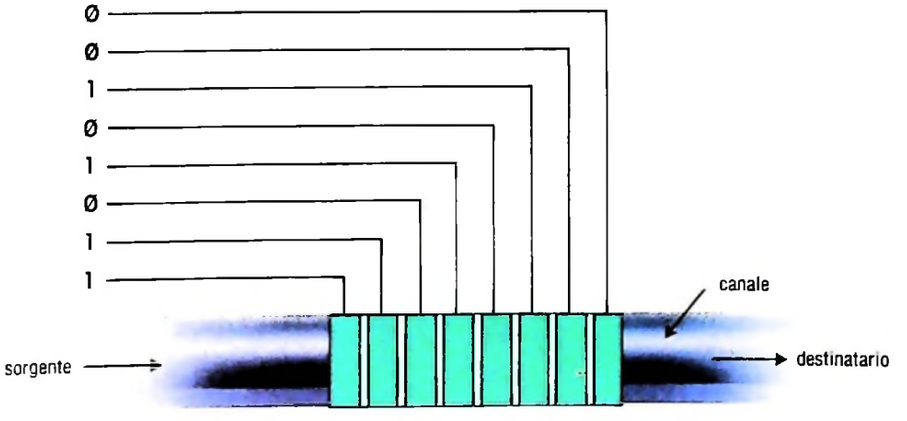
Nella *trasmissione parallela* tutti i bit del carattere codificato sono trasmessi simultaneamente. Questo significa che ogni livello del codice viene immesso su un canale a esso dedicato. La figura in basso mostra come tutti i bit di un carattere lascino l'origine simultaneamente e come nello stesso modo arrivino a destinazione. Quindi con il termine "trasmissione parallela" si fa riferimento al fatto che i bit del carattere sono trasmessi in parallelo mentre i caratteri vengono trasmessi uno dopo l'altro (serialmente).

La trasmissione parallela viene per lo più impiegata per le comunicazioni locali e per il trasferimento di dati tra un elaboratore e le sue periferiche poiché in questo caso si raggiungono velocità di trasferimento molto elevate. Non viene invece normalmente utilizzata nelle trasmissioni dati, perché sulle lunghe distanze il costo diventa esorbitante.

Di gran lunga più utilizzata è la *trasmissione seriale*, dove i bit del carattere sono trasmessi l'uno dopo l'altro lungo il canale di comunicazione: è un po' come se si trattasse d'un convoglio ferroviario lungo i binari. Il destinatario assembla poi in caratteri il flusso dei bit in arrivo.

La differenza tra i due modi di trasmissione è la seguente: in quello seriale il sorgente invia un bit seguito da un intervallo di tempo, poi il secondo bit e così via finché tutti i bit non siano stati trasmessi. Questo significa che l'operazione impiega n unità di tempo per trasmettere n bit (8 nel caso ASCII).

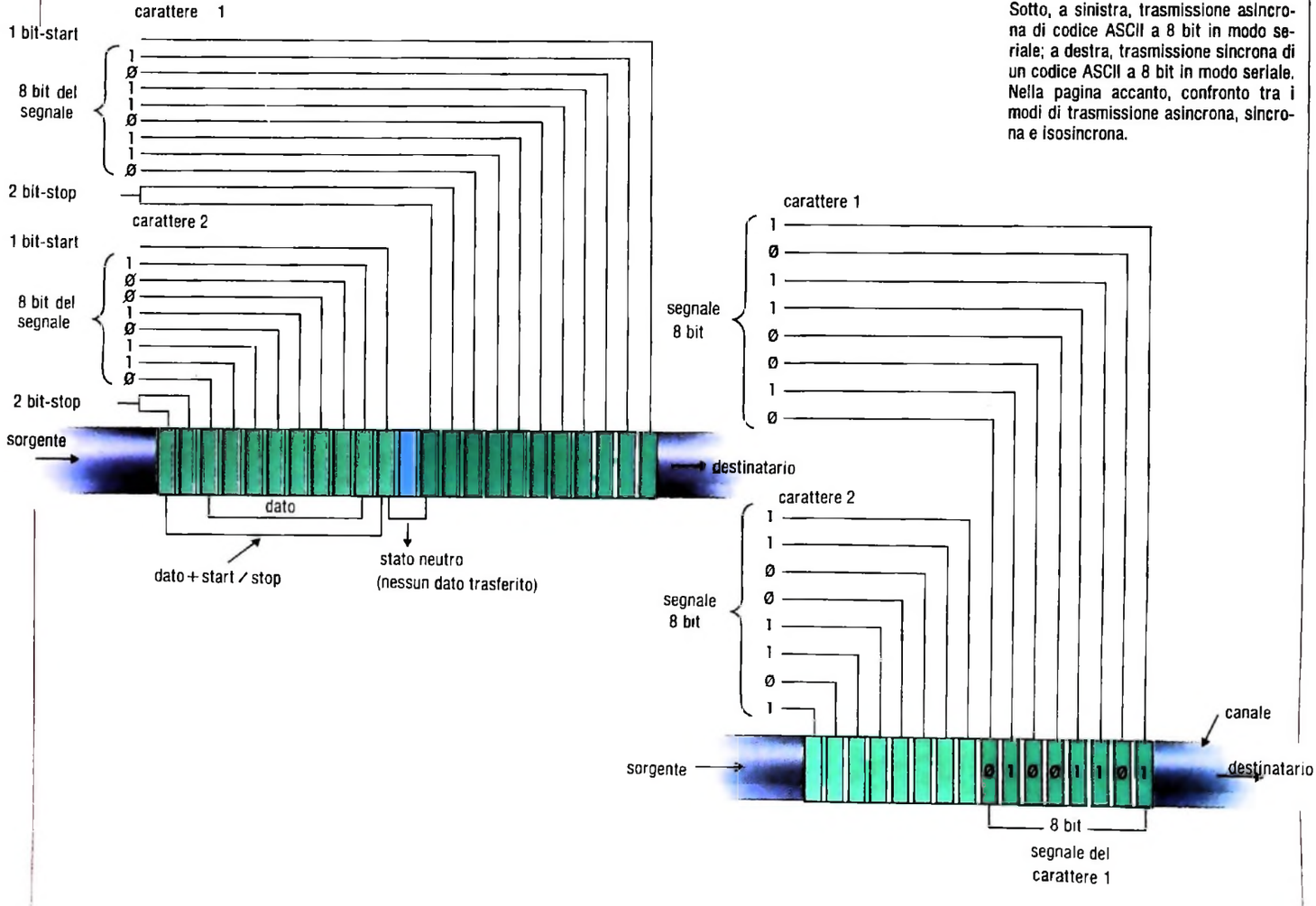




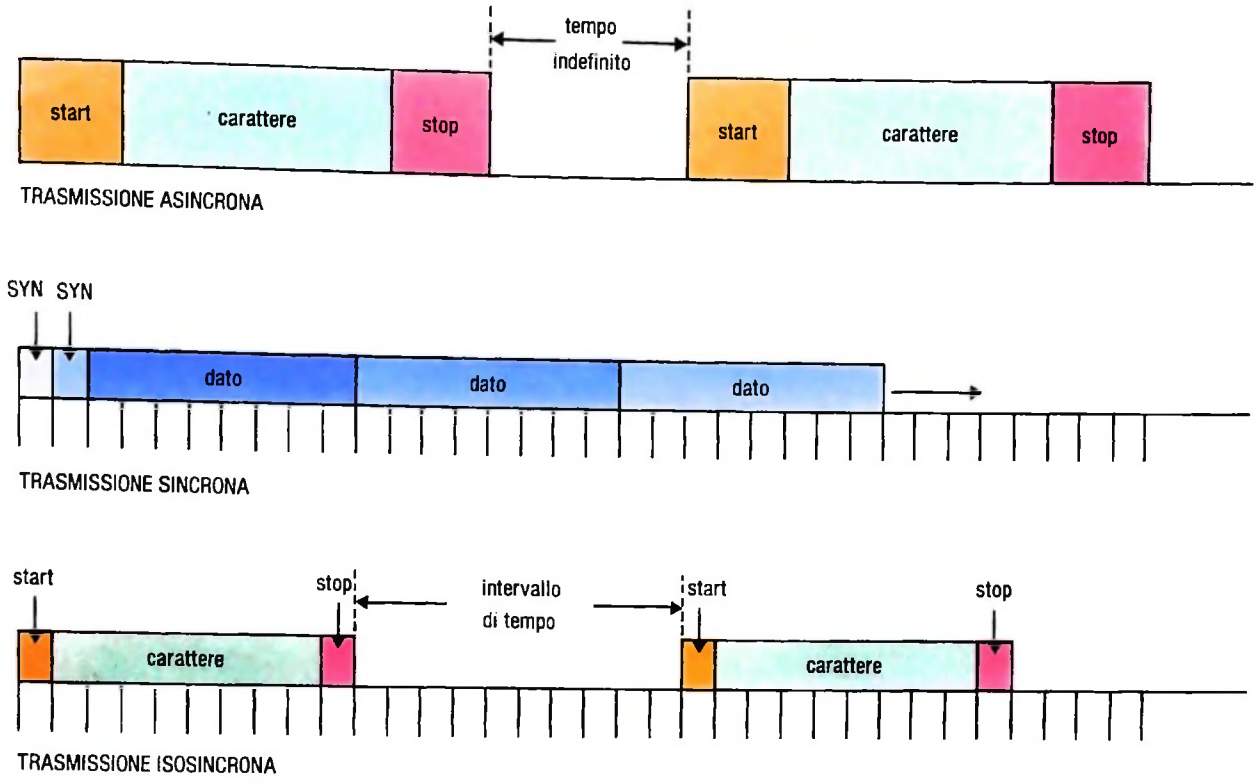
Sopra, trasmissione parallela di un codice ASCII a 8 bit. A lato, trasmissione seriale di un codice ASCII a 8 bit.

In quella parallela, invece, vengono inviati n bit, poi l'intervallo di tempo, quindi altri n bit e così via. In questo caso quindi la trasmissione di n bit (8 nel caso ASCII) richiede una sola unità di tempo.
La trasmissione seriale è a sua volta suddivisa in tre categorie: *asincrona*, *sincrona*, *isosincrona*.
La *trasmissione seriale asincrona* (detta spesso "start/stop"), viene usata in quei sistemi in cui i caratteri sono inviati uno alla volta senza che ci debba essere necessariamente una relazione fissa tra un carattere e il successivo.
I caratteri possono essere inviati a intervalli irregolari: per esempio un carattere per secondo, oppure un carattere, seguito da 10 secondi di pausa.
Per informare il destinatario dell'arrivo e quindi dell'assemblaggio di un carattere, ciascuno dei caratteri trasmessi è preceduto da un bit di start e seguito (al termine degli n bit che lo compongono) da 1 o 2 bit di stop. (Il codice ASCII contempla ambedue le modalità di stop, figura a sinistra). Quindi nella trasmissione seriale asincrona ciascun carattere viene sincronizzato dai suoi bit di start/stop.

In conclusione, il bit di start è il segnale utilizzato per informare il terminale destinatario di cominciare a codificare secondo la propria struttura il segnale in arrivo, mentre quello di stop, che segue i bit costituenti il carattere, informa il destinatario che il carattere è stato interamente ricevuto e inizializza nuovamente il terminale per il seguente bit di start; quindi la sincronizzazione del terminale viene ristabilita ogni volta alla ricezione di ciascun carattere.
La *trasmissione seriale sincrona* viene utilizzata per trasmettere, ad alte velocità, blocchi di caratteri. Sia il sorgente sia il destinatario operano simultaneamente e vengono risincronizzati dopo la trasmissione di alcune migliaia di bit.
La sincronizzazione viene raggiunta tramite la trasmissione di un predeterminato gruppo di caratteri "SYNC" dal terminale sorgente a quello di destinazione, in modo che quest'ultimo possa determinare l'intervallo di tempo tra i vari bit (figura a destra).
Il sorgente, quindi, invia un lungo "treno" di dati che possono contenere migliaia di bit. Il destinatario, essendo a conoscenza del codice utilizzato, conta l'appropriato numero di



Sotto, a sinistra, trasmissione asincrona di codice ASCII a 8 bit in modo seriale; a destra, trasmissione sincrona di un codice ASCII a 8 bit in modo seriale. Nella pagina accanto, confronto tra i modi di trasmissione asincrona, sincrona e isosincrona.



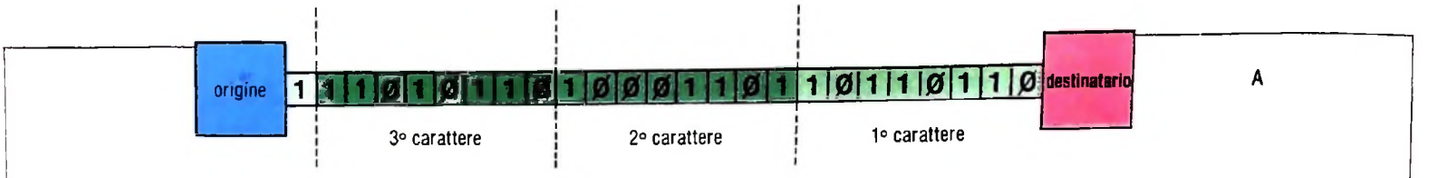
bit e lo codifica nel carattere corrispondente per poi passare al conteggio dei bit formanti il secondo carattere e così via.
La trasmissione seriale sincrona risulta quindi più efficiente in quanto esistono pochissimi bit di controllo in proporzione al numero totale di bit trasmessi; infatti la sincronizzazione occupa da 16 a 32 bit mentre il flusso dei bit del blocco trasmesso può essere composto da migliaia di bit.
D'altra parte, se interviene un errore durante una trasmissione asincrona, l'effetto dello stesso può al massimo alterare o distruggere un carattere, dato che ciascun carattere si autosincronizza tramite i suoi bit di start/stop; lo stesso errore, in una trasmissione sincrona, può causare la "distruzione" dell'intero blocco del messaggio, avendone interrotta la sincronizzazione.
La *trasmissione seriale isosincrona* combina le caratteristiche della trasmissione asincrona e di quella sincrona: ciascun carattere ha bit di start e bit di stop e la sorgente e il destinatario sono sincronizzati.
Il tempo di sincronizzazione, intervallo tra successivi bit, è un multiplo intero della lunghezza di un bit di codice. Cioè, tutti i periodi di non trasmissione consistono in un intervallo di tempo equivalente a uno o a più di un carattere. La figura in alto illustra le interrelazioni e le differenze fra le tre diverse modalità di trasmissione seriale.
In conclusione, i modi di trasmissione sono *seriale* (bit dopo bit lungo il canale) e *parallela* (tutti i bit costituenti il carattere inviati simultaneamente). A sua volta la trasmissione seriale può essere suddivisa in:
asincrona: dove i bit di un carattere sono spediti indipendentemente dal tempo e ciascun carattere è preceduto da un bit di start e 1 o 2 bit di stop;
sincrona: dove la stazione sorgente e quella destinataria sono sincronizzate tra loro e la trasmissione è caratterizzata dall'invio di blocchi formati da migliaia di bit;
isosincrona: dove i caratteri sono muniti di bit di start e stop e le due stazioni sono sincronizzate tra loro.

Efficienza della trasmissione

La figura in alto della pagina seguente confronta l'efficienza della trasmissione dati con modalità sincrona e asincrona. Dalla figura A che schematizza la trasmissione sincrona, si vede che si impiega quasi tutta la capacità della linea per portare informazioni. La figura B riporta il carico dei bit di start e di stop su ogni carattere e il tempo variabile che intercorre tra i caratteri.
Nel caso del codice ASCII (il più usato nelle trasmissioni dati) si devono quindi trasmettere un totale di 10 bit per ogni carattere (8 bit di parola + 1 bit di start + 1 bit di stop). Supponendo che i caratteri siano intervallati da un tempo zero e che si debbano trasmettere 300 caratteri, otteniamo:

300 caratteri × 8 bit per carattere	= 2400 bit
300 caratteri × 2 bit start/stop	= 600 bit
Totale bit trasmessi	= 3000 bit

L'efficienza è quindi data dal rapporto:
2400 bit di informazione / 3000 bit trasmessi = 80%.
Nel caso di trasmissione sincrona, supposto di trasmettere lo stesso blocco di 300 caratteri in codice ASCII e che il blocco sia preceduto da 3 caratteri SYN si ottiene:



TRASMISSIONE SINCRONA

TRASMISSIONE ASINCRONA

Confronto tra trasmissione sincrona e trasmissione asincrona.

300 caratteri × 8 bit per carattere = 2400 bit
3 caratteri SYN × 8 bit per carattere = 24 bit
Totale bit trasmessi = 2424 bit

L'efficienza è quindi:
2400 bit di informazione / 2424 bit trasmessi = 99%.

Si nota quindi che l'efficienza massima (avendo supposto tempo zero tra carattere e carattere) che si può ottenere con la trasmissione asincrona è inferiore di circa il 19% a quella della trasmissione sincrona.

In generale la trasmissione sincrona usa più efficacemente il canale di comunicazione di quanto non faccia la trasmissione asincrona. Un canale capace di trasmettere 4800 bit al secondo può trattare 600 caratteri ASCII in modo sincrono, ma potrebbe trasferire solo 480 caratteri al secondo in modo asincrono (supponendo che il segnale di stop sia composto da 1 bit).

La trasmissione asincrona presenta però il grande vantaggio di consentire l'impiego di apparecchi di ricetrasmissione non costosi e poco sofisticati che consentono, per esempio, all'operatore di battere i caratteri inviandoli direttamente in linea uno per volta alla velocità che desidera perché non deve rispettare limiti di tempo tra un carattere e l'altro.

Il bit di parità

Come si è già visto, in ogni carattere ASCII i bit da 1 a 7 contengono l'informazione che si desidera trasmettere mentre il bit 8 non trasmette una nuova informazione, ma conferma quella appena trasmessa e viene identificato con il nome *bit di parità*.

Tale bit ha lo scopo di consentire (almeno entro certi limiti) di individuare gli errori. Il bit può essere *pari* (*even*) o *dispari* (*odd*).

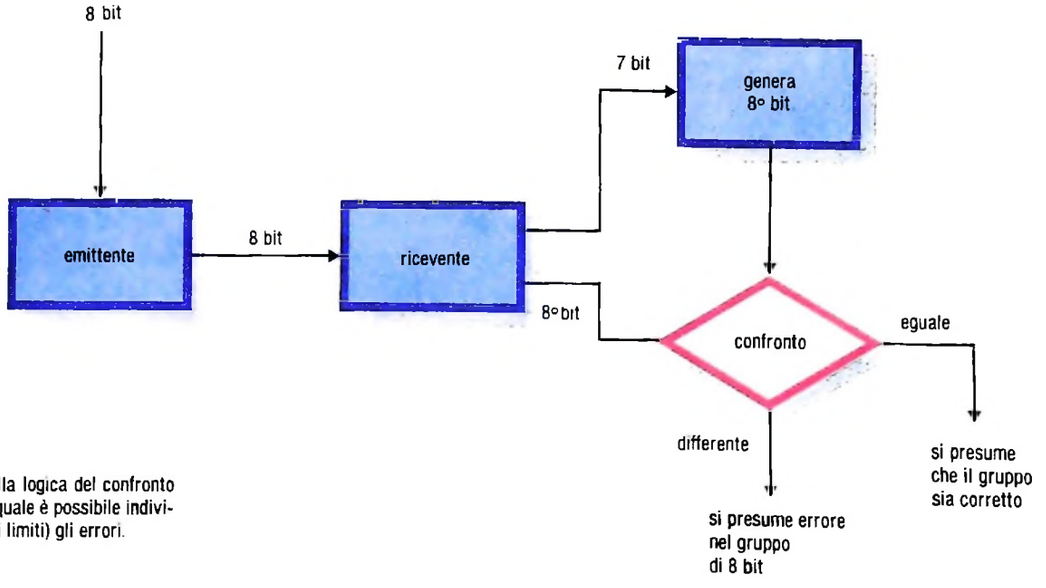
Nel caso pari, il bit 8 assumerà il valore necessario per rendere pari la somma degli "uno" del carattere codificato (compreso il bit di parità), nel caso di parità dispari, il bit 8 assumerà un valore tale da rendere dispari il numero di "uno" del carattere.

Prendiamo per esempio la lettera V e la lettera W e assumendo parità pari:

la lettera "V" è codificata come 0110101; dato che il numero degli "uno" è 4 (quindi pari) il bit 8 assumerà valore 0;

la lettera "W" è codificata come 0001101; il numero degli "uno" è 3 (dispari) quindi il bit 8 assumerà valore 1.

La figura qui sotto visualizza come opera il bit di parità.

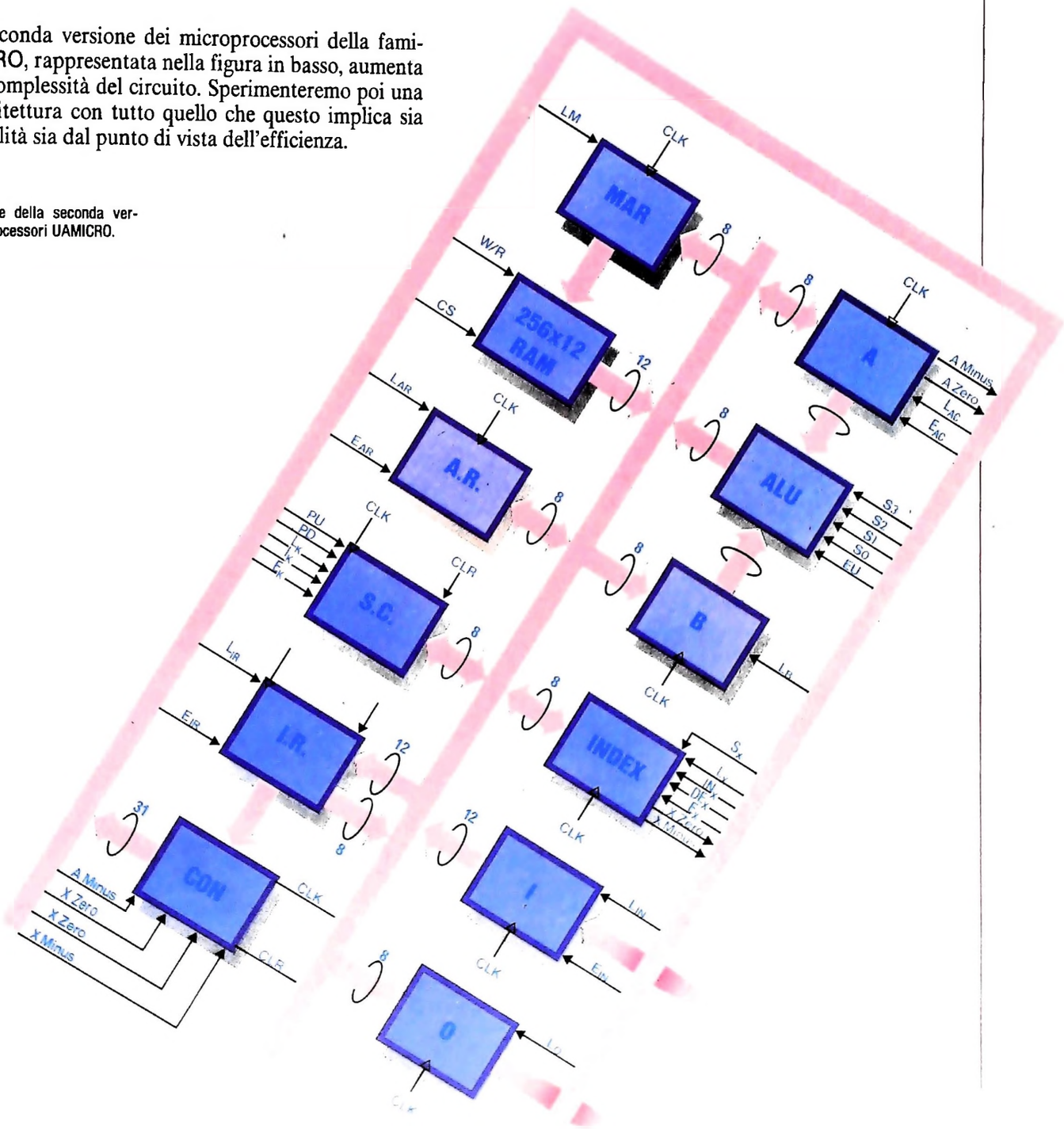


Procedimento della logica del confronto di parità, con la quale è possibile individuare (entro certi limiti) gli errori.

UAMICRO II

Istruzione di salto, possibilità di richiamo in cascata dei sottoprogrammi, architettura con registri di entrata e di uscita sono alcune delle caratteristiche di questo microprocessore.

In questa seconda versione dei microprocessori della famiglia UAMICRO, rappresentata nella figura in basso, aumenta il grado di complessità del circuito. Sperimenteremo poi una diversa architettura con tutto quello che questo implica sia per funzionalità sia dal punto di vista dell'efficienza.

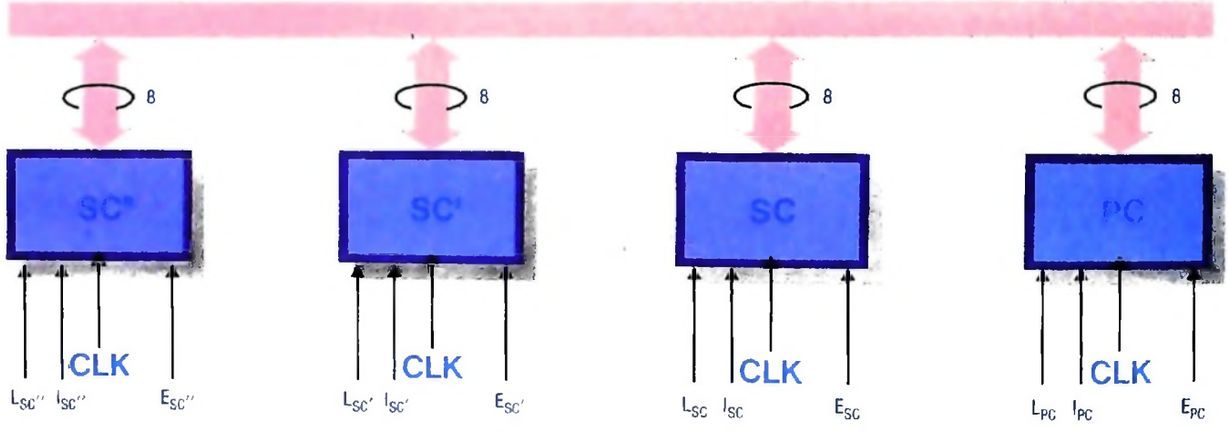


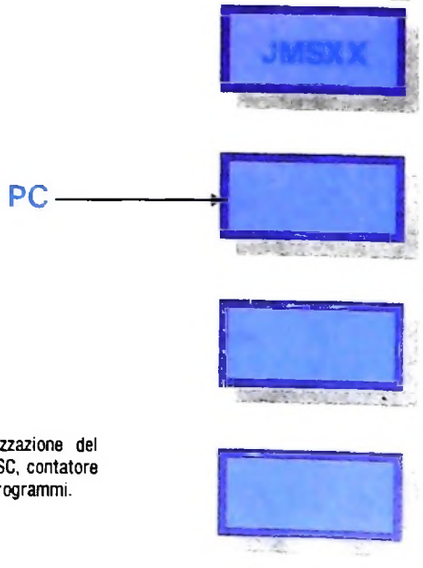
Struttura generale della seconda versione dei microprocessori UAMICRO.

Ecco le specifiche dell'UAMICRO II
— Bus unico per indirizzi e dati.
— Parola di 12 bit in cui compaiono contemporaneamente sia le istruzioni (4 bit), sia gli operandi (8 bit). Questa nuova tecnica ci permette di aumentare la velocità: nella versione precedente con le istruzioni MRI dovevamo infatti recuperare dalla memoria l'istruzione e poi ritornare a prendere l'operando, effettuando due trasferimenti per completare l'istruzione, adesso in una sola volta vengono prelevati istruzione e operando. La maggiore velocità consente una maggiore quantità di memoria che, in questo caso, deve avere una parola di 12 bit, e una maggiore quantità di linee di bus.
— Istruzione di salto (JUMP) che permette, anche di lanciare sottoprogrammi o ripetere iterativamente una sequenza di istruzioni.
— Architettura ispirata alla versione Von Neumann completa per cui sono presenti anche i registri di entrata e uscita con i rispettivi comandi.
— Possibilità di richiamo "in cascata" dei sottoprogrammi: vedremo meglio più avanti questa forma (annidamento dei sottoprogrammi).
Incominciamo a esaminare i vari registri.

SC (Contatore di sottoprogrammi)

Questo registro è un registro complesso composto da un PC (*Program Counter*, contatore di programma), uguale a quello della versione precedente, e altri elettronicamente uguali, ma con caratteristiche differenti. Per poter capire esattamente le funzioni di questo registro abbiamo bisogno di approfondire un po' alcuni concetti basilari.
— Sottoprogrammi
La necessità di disporre di sottoprogrammi capaci di svolgere funzioni specifiche cambiando soltanto i dati o i parametri iniziali è stata riconosciuta fin dall'inizio dello sviluppo dei calcolatori. Bisognava poi risolvere il problema di come sfruttare tali sottoprogrammi. Nel corso degli anni sono stati messi a punto alcuni metodi di utilizzazione che risultano di un certo interesse. Supponiamo di eseguire un programma e di trovarci, a un certo punto, di fronte alla necessità di estrarre la radice quadrata di un numero. Invece di proseguire nella scrittura di una sequenza di istruzioni che eseguano tale operazione, è più pratico poter disporre di tale sequenza in un altro posto della memoria in modo che sia sempre accessibile. In altre parole, non è necessario riscrivere la stessa





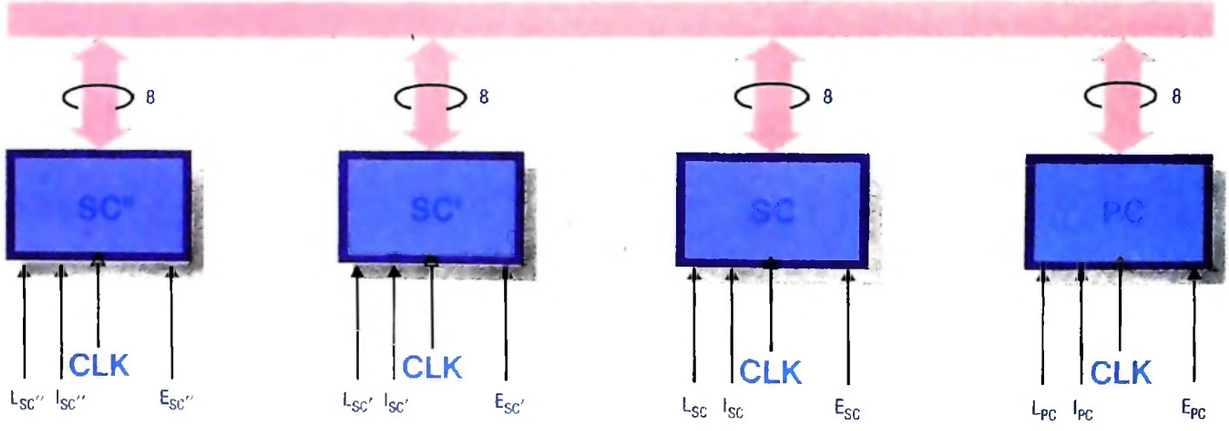
Schematizzazione del registro SC, contatore di sottoprogrammi.

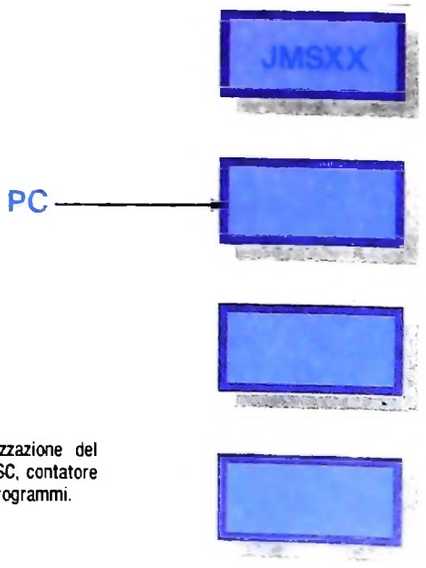
sequenza ogni volta che se ne ha bisogno, basta scriverla una volta e richiamarla poi tutte le volte che è necessario. Il problema sorge però nelle sequenze di istruzioni richiamate più volte che vengono dette sottoprogrammi. Quando stiamo eseguendo un programma impegniamo per una ragione o per l'altra i registri interni del microprocessore, principalmente il PC. I valori in esso contenuti, necessari per la corretta esecuzione del programma, devono essere conservati da qualche parte per permettere al sottoprogramma di usare il PC e gli altri registri interni. Naturalmente, quando il sottoprogramma ha esaurito la sua funzione e ha dato un certo risultato, bisogna ripristinare tutto come prima, ricaricando i registri con i valori precedenti per poter riprendere il processo. L'operazione di salvataggio dei contenuti dei registri (più particolarmente il registro PC per la UAMICRO II) e il loro ripristino può essere fatto in più modi, ma richiede tempo di gestione. Ne prenderemo in considerazione solo due: il primo

Ecco le specifiche dell'UAMICRO II
— Bus unico per indirizzi e dati.
— Parola di 12 bit in cui compaiono contemporaneamente sia le istruzioni (4 bit), sia gli operandi (8 bit). Questa nuova tecnica ci permette di aumentare la velocità: nella versione precedente con le istruzioni MRI dovevamo infatti recuperare dalla memoria l'istruzione e poi ritornare a prendere l'operando, effettuando due trasferimenti per completare l'istruzione, adesso in una sola volta vengono prelevati istruzione e operando. La maggiore velocità consente una maggiore quantità di memoria che, in questo caso, deve avere una parola di 12 bit, e una maggiore quantità di linee di bus.
— Istruzione di salto (JUMP) che permette, anche di lanciare sottoprogrammi o ripetere iterativamente una sequenza di istruzioni.
— Architettura ispirata alla versione Von Neumann completa per cui sono presenti anche i registri di entrata e uscita con i rispettivi comandi.
— Possibilità di richiamo "in cascata" dei sottoprogrammi: vedremo meglio più avanti questa forma (annidamento dei sottoprogrammi).
Incominciamo a esaminare i vari registri.

SC (Contatore di sottoprogrammi)

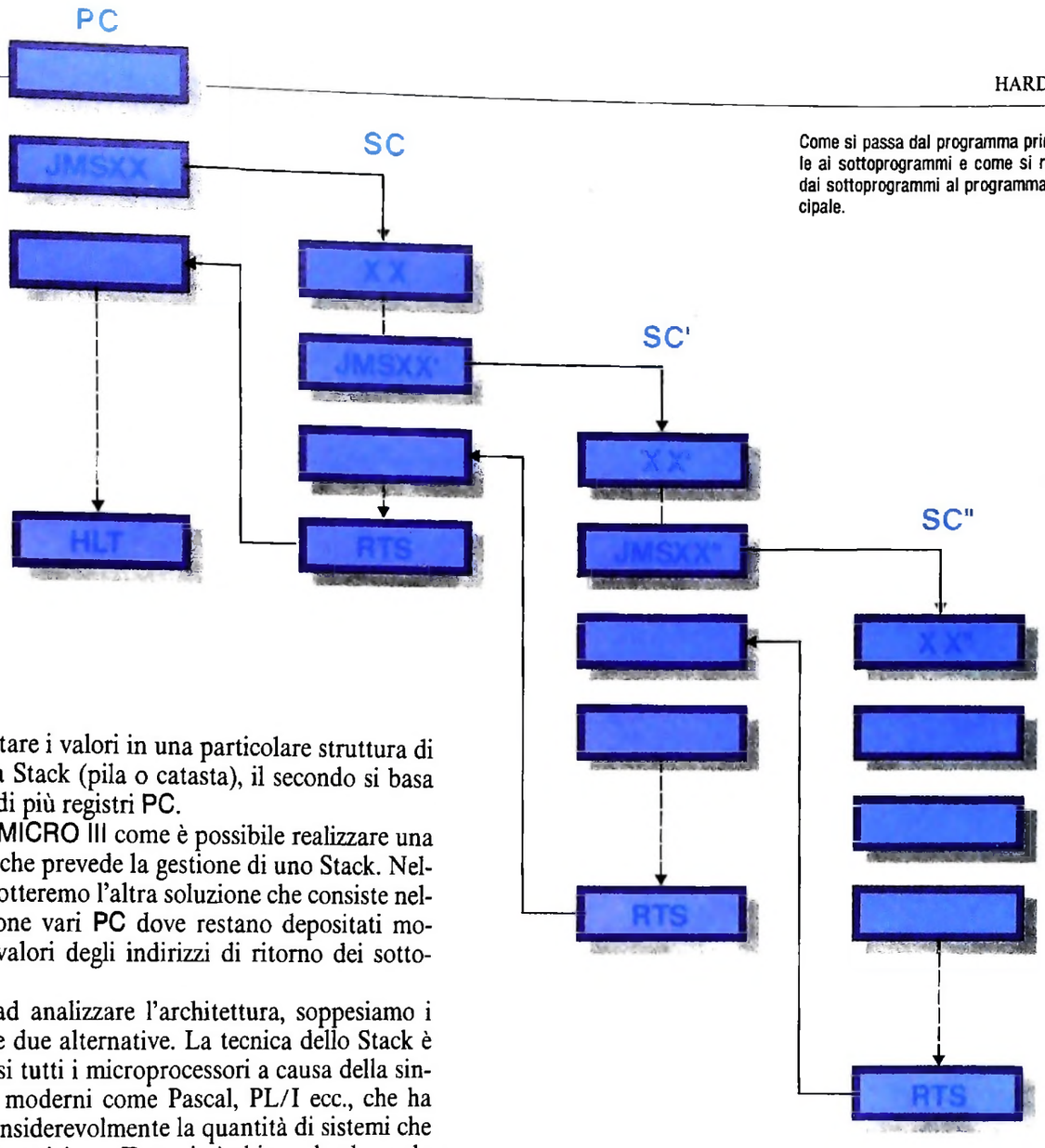
Questo registro è un registro complesso composto da un PC (*Program Counter*, contatore di programma), uguale a quello della versione precedente, e altri elettronicamente uguali, ma con caratteristiche differenti. Per poter capire esattamente le funzioni di questo registro abbiamo bisogno di approfondire un po' alcuni concetti basilari.
— Sottoprogrammi
La necessità di disporre di sottoprogrammi capaci di svolgere funzioni specifiche cambiando soltanto i dati o i parametri iniziali è stata riconosciuta fin dall'inizio dello sviluppo dei calcolatori. Bisognava poi risolvere il problema di come sfruttare tali sottoprogrammi. Nel corso degli anni sono stati messi a punto alcuni metodi di utilizzazione che risultano di un certo interesse. Supponiamo di eseguire un programma e di trovarci, a un certo punto, di fronte alla necessità di estrarre la radice quadrata di un numero. Invece di proseguire nella scrittura di una sequenza di istruzioni che eseguano tale operazione, è più pratico poter disporre di tale sequenza in un altro posto della memoria in modo che sia sempre accessibile. In altre parole, non è necessario riscrivere la stessa





Schematizzazione del registro SC, contatore di sottoprogrammi.

sequenza ogni volta che se ne ha bisogno, basta scriverla una volta e richiamarla poi tutte le volte che è necessario. Il problema sorge però nelle sequenze di istruzioni richiamate più volte che vengono dette sottoprogrammi. Quando stiamo eseguendo un programma impegniamo per una ragione o per l'altra i registri interni del microprocessore, principalmente il PC. I valori in esso contenuti, necessari per la corretta esecuzione del programma, devono essere conservati da qualche parte per permettere al sottoprogramma di usare il PC e gli altri registri interni. Naturalmente, quando il sottoprogramma ha esaurito la sua funzione e ha dato un certo risultato, bisogna ripristinare tutto come prima, ricaricando i registri con i valori precedenti per poter riprendere il processo. L'operazione di salvataggio dei contenuti dei registri (più particolarmente il registro PC per la UAMICRO II) e il loro ripristino può essere fatto in più modi, ma richiede tempo di gestione. Ne prenderemo in considerazione solo due: il primo



Come si passa dal programma principale ai sottoprogrammi e come si ritorna dai sottoprogrammi al programma principale.

consiste nel depositare i valori in una particolare struttura di memoria, chiamata Stack (pila o catasta), il secondo si basa sulla disponibilità di più registri PC.
Vedremo nella UAMICRO III come è possibile realizzare una architettura simile che prevede la gestione di uno Stack. Nella UAMICRO II adotteremo l'altra soluzione che consiste nell'avere a disposizione vari PC dove restano depositati momentaneamente i valori degli indirizzi di ritorno dei sottoprogrammi.
Prima di passare ad analizzare l'architettura, soppesiamo i pro e i contro delle due alternative. La tecnica dello Stack è la più usata in quasi tutti i microprocessori a causa della sintassi dei linguaggi moderni come Pascal, PL/I ecc., che ha fatto aumentare considerevolmente la quantità di sistemi che usano lo Stack a disposizione. Tuttavia è chiaro che dovendo andare continuamente in memoria per depositare o ritirare i contenuti dei registri interni tutta l'operazione viene rallentata, mentre con il sistema dei vari PC tale operazione è velocissima. In definitiva la scelta di una forma o dell'altra dipende dalla velocità di esecuzione che si vuole e dalla quantità di annidamenti giudicata necessaria.
Nella figura della pagina a lato vediamo la configurazione interna del sistema SC: i vari elementi interni sono tutti e quattro esattamente uguali, il nome diverso è dovuto alle diverse funzioni a cui sono preposti. Sia per il registro PC sia per gli altri, i comandi E_{PC}, E_{SC}, $E_{SC'}$, $E_{SC''}$, sono comandi di abilitazione che già conosciamo, i comandi I_{PC}, I_{SC}, $I_{SC'}$, e $I_{SC''}$ sono comandi di incremento sincroni e L_{PC}, $L_{SC'}$ e $L_{SC''}$ sono comandi di caricamento anch'essi sincroni. Analizziamo il loro funzionamento.
Quando stiamo elaborando il programma principale è il registro PC quello che emette gli indirizzi e gestisce tutte le operazioni (figura in alto), quando invece ci troviamo davanti, per la prima volta, alla istruzione JMS XX che vuol dire "salta alla locazione di memoria con indirizzo XX e di lì inizia a elaborare il sottoprogramma esistente" il controllo passa al registro SC, previamente caricato con il valore XX, mentre nel PC resta immagazzinato l'indirizzo seguente all'istruzione JMS XX del programma principale al quale bisogna ritornare al termine del sottoprogramma per riprenderne l'esecuzione. Per tornare al programma principale e segnalare anche il termine del sottoprogramma è disponibile un'istruzione apposita denominata RTS, che vuol dire "Ritorno dal sottoprogramma". Ogni volta che il registro CON incontra questa istruzione sposta di un registro a sinistra i comandi di esecuzione.
Naturalmente è possibile da un sottoprogramma chiamare un altro sottoprogramma: questa operazione viene chiamata di annidamento. Nel nostro caso al massimo possiamo "annidare" tre sottoprogrammi, però nulla impedisce di aumentare la quantità di registri SC dando quindi la possibilità di annidamento. Questa architettura è, come abbiamo detto, una architettura molto veloce: naturalmente il prezzo di questo vantaggio si paga con la necessità di avere più registri e con il controllo necessario a gestirla.

Sommario del Secondo Volume

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.

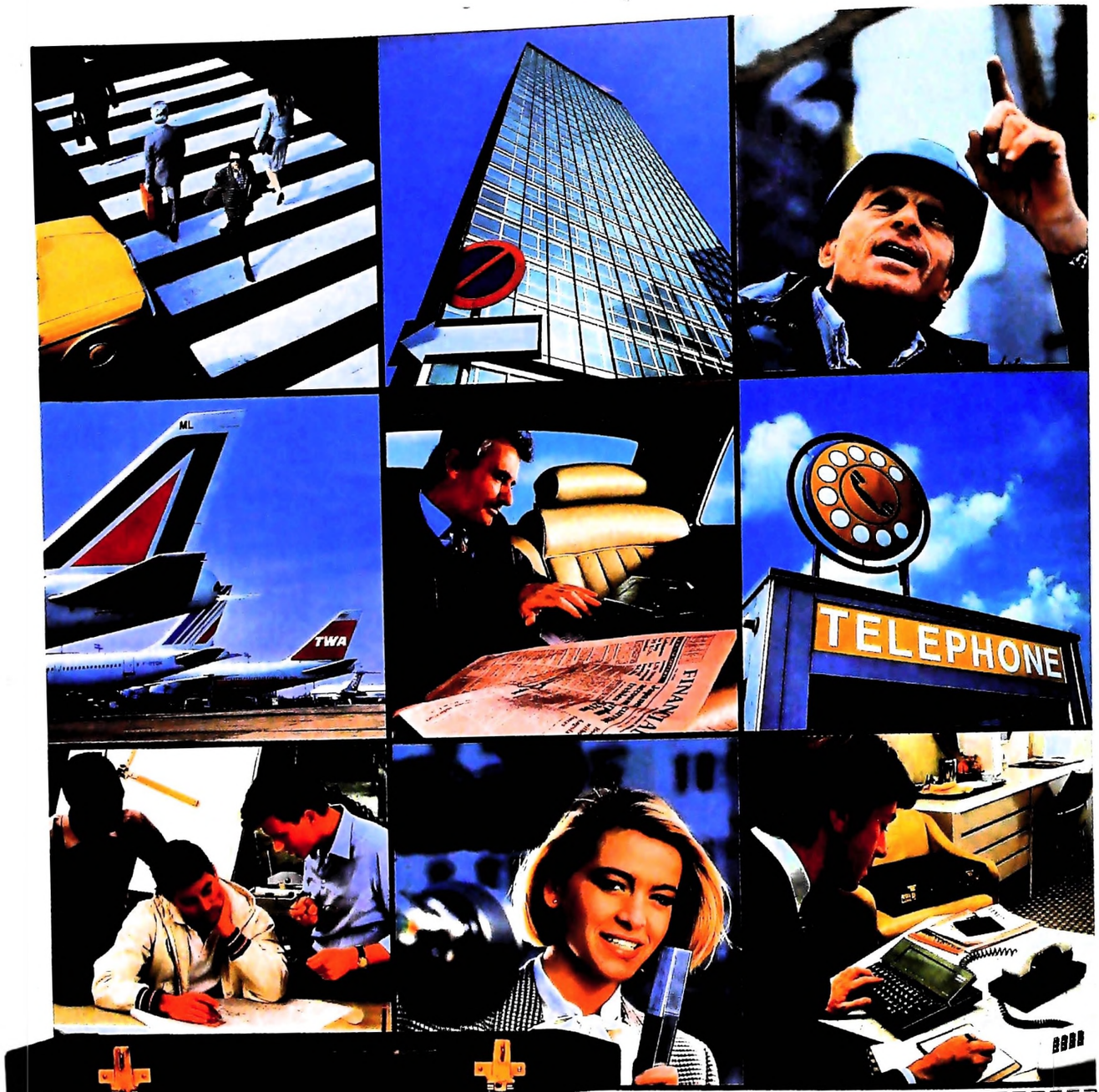
Più pratico, perché consente l'utilizzo del sistema di prelievo automatico Bancomat e l'ottenimento della carta di credito.

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 ai negozi contrassegnati da "Olivetti M10 Punto di Vendita" o inviare il coupon a: Olivetti, Divisione Personal Computer, Via Meravigli 12, 20123 Milano

NOME/COGNOME

VIA N

CAP/CITTÀ

TELEFONO