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

(...)

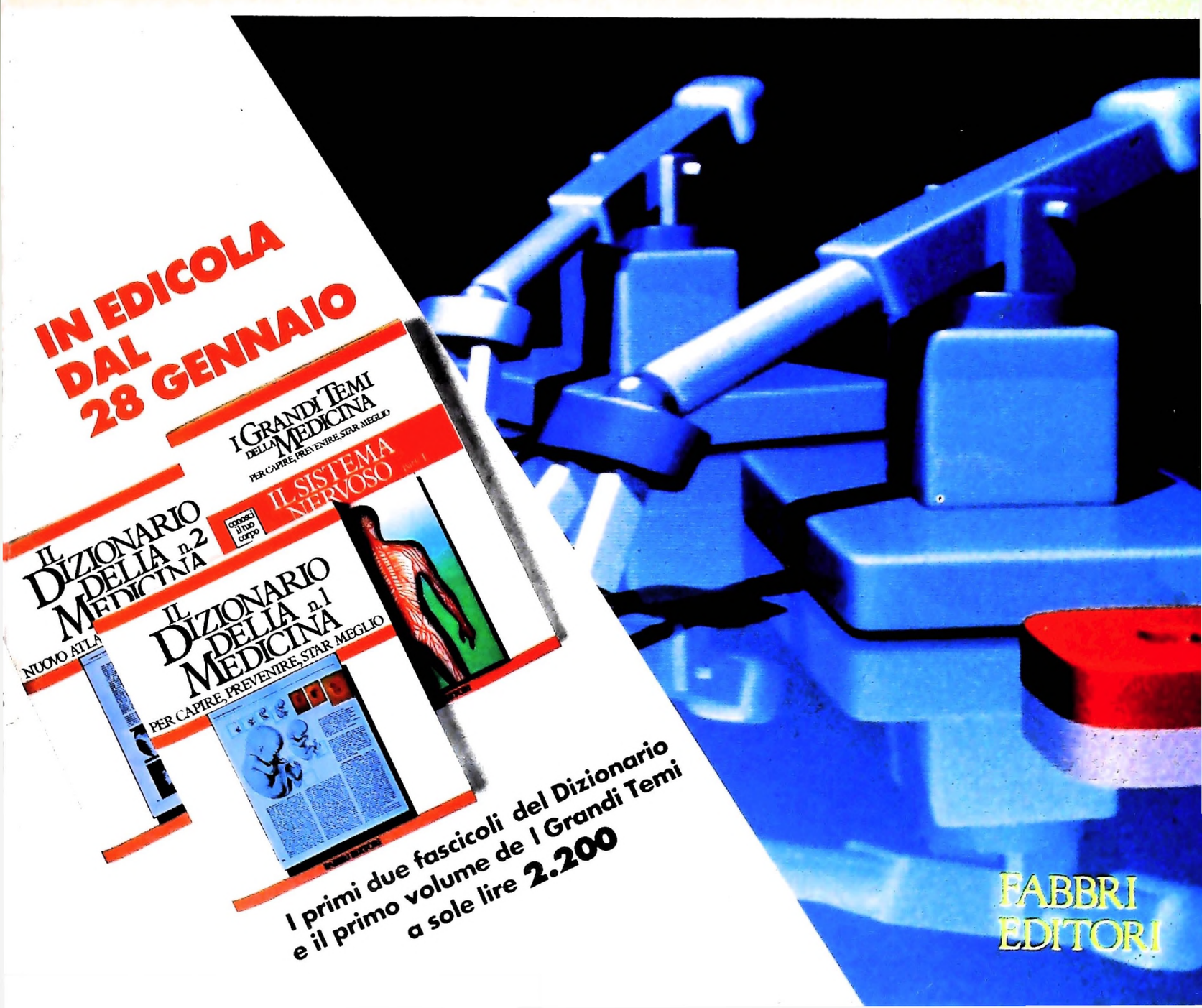
42 CORSO PRATICO COL COMPUTER

421941

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

F4 F5 F6 F7 F8 BATTERIA CARICA

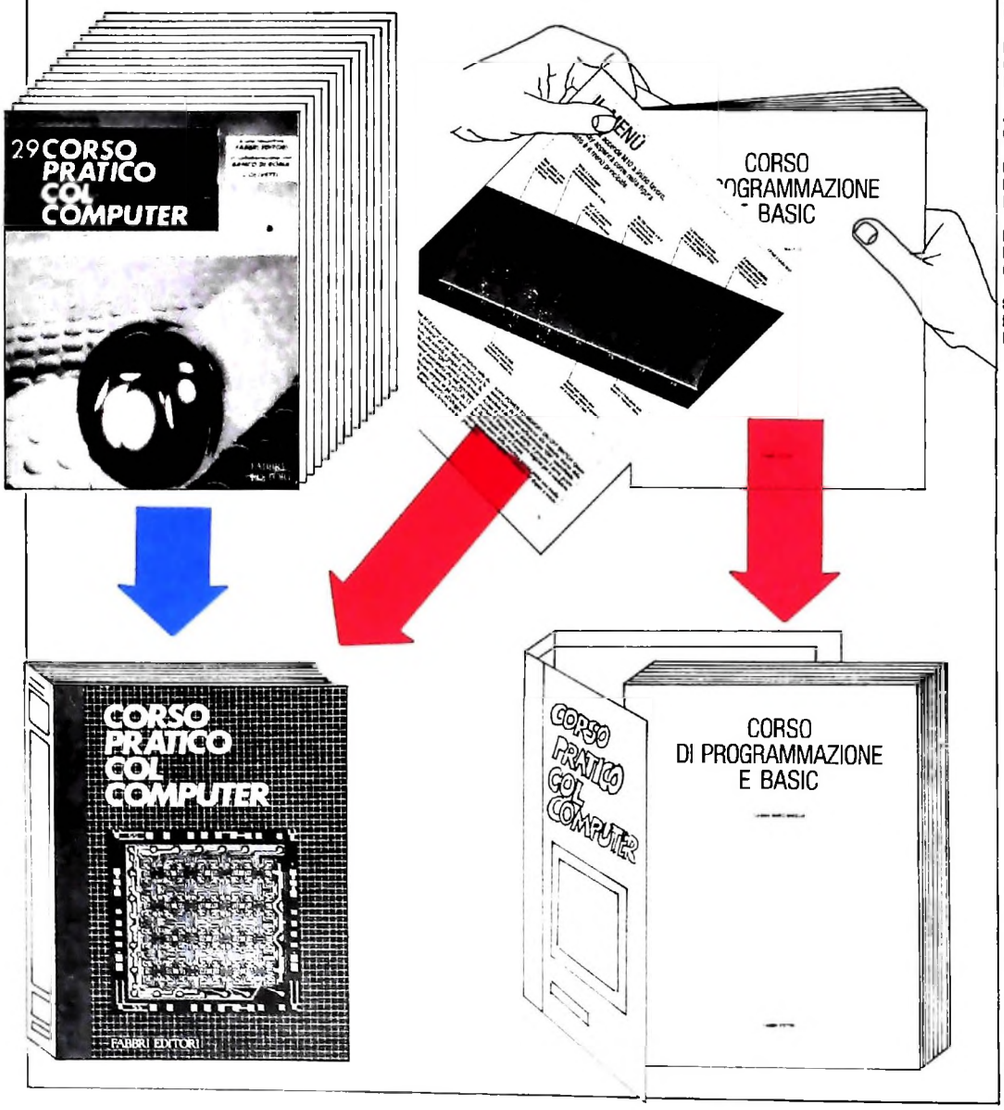
diretto da GIANNI DEGLI ANTONI



AVVISO AI LETTORI

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

- i fascicoli dal n. 29 al n. 42;
- 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 è stata 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. 40;
- ricordiamo che il frontespizio del volume fa parte del fascicolo n. 29 e il sommario del fascicolo n. 42.



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 dei 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,
UGO SALVI, 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

FASI

Conoscere la differenza tra i diversi tipi di memoria è importante per una scelta ragionata e obiettiva di un sistema operativo ottimale.

Nell'articolo precedente abbiamo detto che nell'indirizzare la memoria non sono tanto importanti la quantità e la qualità delle istruzioni, quanto la tecnica usata per realizzarle, cioè la *microprogrammazione*, in quanto è assolutamente indispensabile conoscerla se si vuol capire come funziona internamente un microprocessore.
Facciamo alcuni esempi.
Vediamo l'esecuzione dell'istruzione LDAA (carica l'accumulatore "A" con il contenuto della memoria) analizzando la variazione della *parola di controllo* attraverso le varie fasi. Per capire meglio questa parte è necessario riprendere in esame l'architettura dei registri A.R., P.C., INDEX e S.P. che abbiamo presentato alle pagine 526, 527 quando abbiamo parlato dell'UAMICRO III.

Fase di FETCH

Incominciamo con la fase di FETCH che è uguale per tutte le istruzioni (figura 1). La ricerca dell'istruzione (FETCH) si sviluppa in tre fasi: nella prima fase T_0 l'istruzione viene prelevata dalla memoria con E_{PC} e VMA attivi e caricata nel registro I.R. con L_{IR}. Nella fase T_1 si incrementa il contenuto del P.C, nella terza fase T_2 non c'è azione esterna, ma solo tempo di elaborazione interna. Alcuni microprocessori usano questa fase per effettuare un'azione di rinfresco per le memorie dinamiche, come lo Z80, per esempio.
Trascorsa la fase di FETCH incomincia la fase di ESECUZIONE, e questa varia secondo le istruzioni e la forma di indirizzamento della memoria scelta.
Vediamo adesso lo sviluppo delle varie forme di indirizzamento prendendo come base l'istruzione LDAA.
IMMEDIATA
La forma *immediata* vuol dire, per l'istruzione LDAA naturalmente, caricare l'accumulatore "A" con il byte che segue il codice operativo dell'istruzione (figura 2).
In questo caso nella fase T_3 ritorniamo alla memoria per prendere il dato e caricarlo nell'accumulatore indicato, mentre nella fase che segue, T_4, s'incrementa il P.C.
DIRETTA
L'istruzione in memoria è formata da due byte come la precedente, però mentre prima il 2° byte conteneva un dato, ora contiene un indirizzo alla pagina Zero.
Vediamo come avviene tutto questo.
Nella fase T_3 (figura 3) con CL_{RAR} si cancella la parte sini-

①

PHASE	CON	
T_0	E_{PC} VMA L_{IR}	PC → BUS DIREC, OPER. → IR
T_1	I_{PC}	PC+I → PC
T_2		DECODE INSTR.

Come variano i comandi nelle tre fasi del FETCH.

②

PHASE	CON	
T_3	E_{PC} VMA LA	PC → BUS DIREC, DATA → AC.
T_4	I_{PC}	PC+I → PC

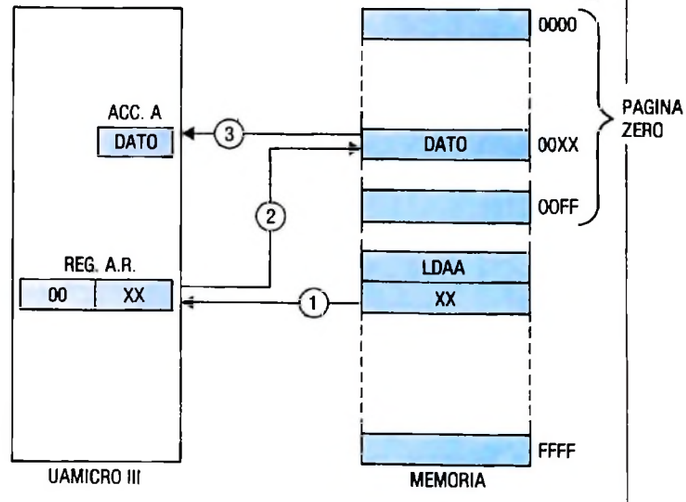
Come variano i comandi nell'istruzione IMMEDIATA.

③

PHASE	CON	
T_3	E_{PC} VMA L_{AR} CL_{RAR}	PC → BUS DIREC, OPERANDO → AR
T_4	E_{AR} VMA L_A I_{PC}	AR → BUS DIREC, DATO → AC, PC+I → PC

Come variano i comandi nell'istruzione DIRETTA.

④



Come viene eseguita l'istruzione DIRETTA. La parte alta del registro A.R. è uguale a zero, per cui punta alle prime 256 località di memoria. Le varie fasi sono numerate in ordine cronologico.

stra del registro ausiliario A.R. (la parte alta dell'indirizzo), che punta così alla pagina Zero (figura 4). Con E_{PC} e VMA si prende il byte in memoria e con L_{AR} lo si carica nella parte destra dell'A.R. per completare l'indirizzo.
Nella fase T_4 con E_{AR} il contenuto di A.R. passa al bus di indirizzi, con VMA si abilita l'operazione, con L_A si carica il dato nell'accumulatore, con I_{PC} si incrementa il P.C.

ESTESA

In questa forma i due byte che seguono il codice operativo dell'istruzione contengono l'indirizzo del dato. L'operazione si svolge così (figure 5 e 6):

— T_3 → si preleva il primo byte dell'indirizzo e lo si carica nella parte destra di A.R.;
— T_4 → s'incrementa il P.C.;
— T_5 → si preleva il 2° byte e lo si deposita nella sinistra di A.R.;
— T_6 → s'incrementa il P.C., si emette l'indirizzo completo da A.R., e con VMA attivo si preleva il dato in memoria e lo si deposita nell'accumulatore "A".

INDEXATA

Carica l'accumulatore "A" con il contenuto della memoria, il cui indirizzo è dato dalla somma del contenuto del registro INDEX più il 2° byte dell'istruzione (figure 7 e 8):

— T_3 → si carica il contenuto del byte che segue il codice operativo dell'istruzione nell'accumulatore B;
— T_4 → si trasferisce il contenuto di B in INDEX, si somma tramite il comando S_{IN}, con il contenuto dello stesso registro e s'incrementa il P.C.;
— T_5 → si preleva il contenuto della memoria con l'indirizzo dato dalla somma in INDEX e si deposita nell'accumulatore "A".

Vediamo adesso l'interessantissima istruzione di JMS (salto a sottoprogramma) che nella Uamicro III si può indirizzare in due modi: ESTESA e INDEXATA. Le due forme sono molto simili per cui svilupperemo solo la prima, lasciando al lettore il compito di sviluppare la seconda forma (figura 9).

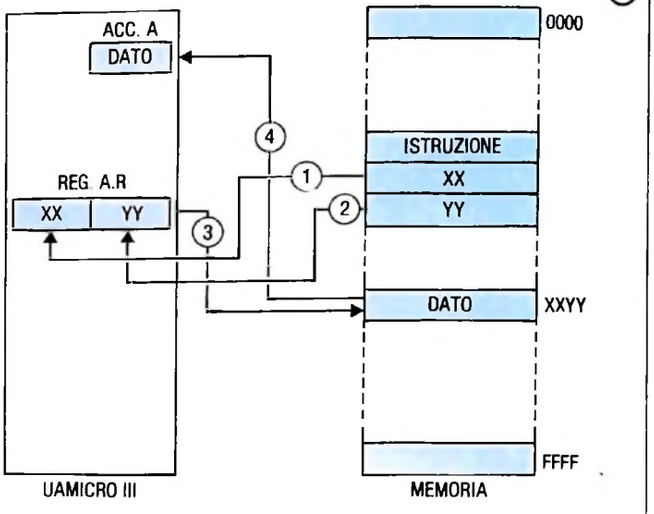
— T_3 → si carica il 1° byte dell'indirizzo (la parte alta dell'indirizzo del sottoprogramma) nella sinistra del registro A.R.;
— T_4 → s'incrementa il P.C.;
— T_5 → si carica il 2° byte (la parte bassa dell'indirizzo) nella parte destra del registro A.R.;
— T_6 → s'incrementa il P.C.;
— T_7 → il contenuto dello S.P è trasferito nel bus di indirizzi per cui punta alla prima località vuota dello STACK. La parte destra del P.C. è trasferita nel bus di dati (attraverso l'INTER. BUS) (figura 10);
— T_8 → s'incrementa lo S.P.;
— T_9 → la parte sinistra del P.C. occupa la seconda località dello STACK (figura 11);
— T_{10} → s'incrementa lo S.P.;
— T_{11} → la parte destra dell'INDEX occupa il terzo posto dello STACK (figura 12);
— T_{12} → si incrementa lo S.P.;
— T_{13} → la parte sinistra dell'INDEX occupa il quarto posto dello STACK (figura 13);
— T_{14} → s'incrementa lo S.P.;

5

PHASE	CON	
T_3	E_{PC} VMA L_{AR}	PC → BUS DIREC, OPERANDO → AR
T_4	I_{PC}	PC+I → PC
T_5	E_{PC} VMA L_{ARR}	PC → BUS DIREC, OPERANDO → AR
T_6	I_{PC} E_{AR} VMA LA	PC+I → PC, AR → BUS DIREC, DATO → ACC.

Come variano i comandi nell'istruzione ESTESA.

6



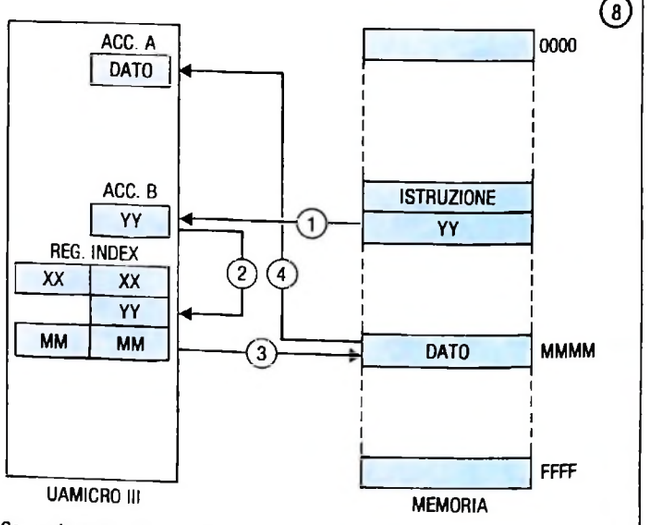
Come viene seguita l'istruzione ESTESA.

7

PHASE	CON	
T_3	E_{PC} VMA L_B	PC → BUS DIRECT, OPERANDO → AC B
T_4	I_{PC} E_B S_{IN} LIN	B + INDEX → INDEX, PC + I → PC
T_5	E_{IN} VMA L_A	INDEX + XX → BUS DIRECT + DATO → AC

Come variano i comandi nell'istruzione INDEXATA.

8



Come viene eseguita l'istruzione INDEXATA.

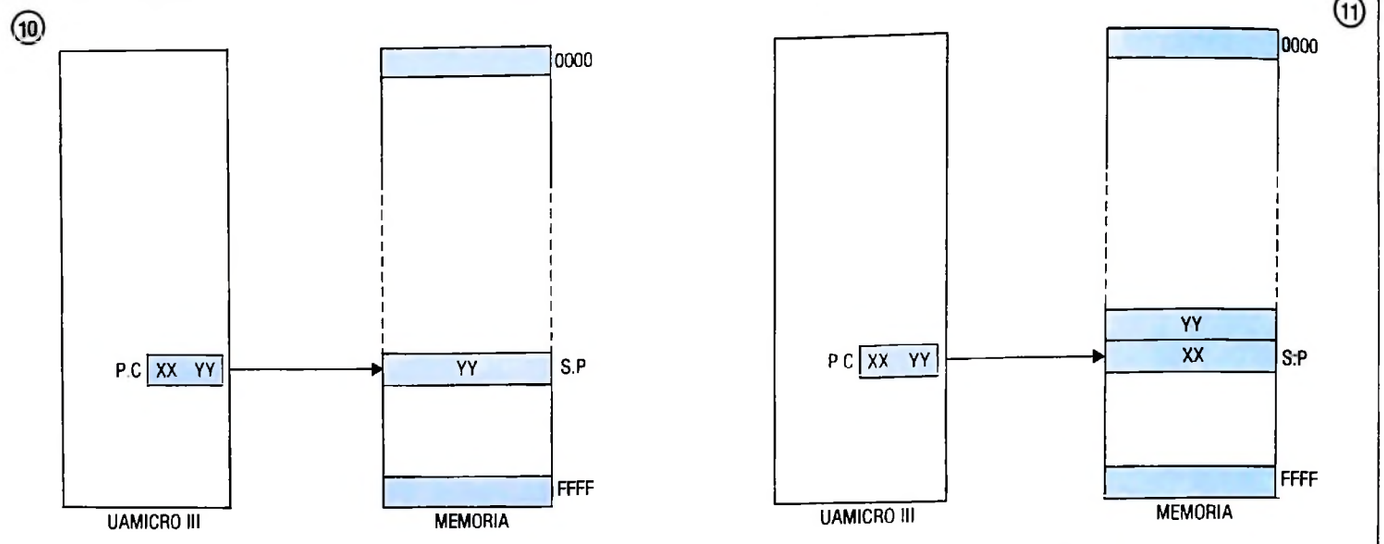
(9)

T_3	E_{PC} VMA LARR	PC → BUS DIREC, OPERANDO → AR
T_4	I_{PC}	PC+1 → PC
T_5	E_{PC} VMA LAR	PC → BUS DIREC, OPERANDO → AR
T_6	I_{PC}	PC+1 → PC
T_7	E_{PCC} E_{SP}W/R VMA	SP → BUS DIREC, PC → BUS DATA
T_8	I_{SP}	SP+1 → SP
T_9	EE_{INN} E_{SP} W/R VMA	SP → BUS DIREC, INDEX → BUS DATA
T_{10}	I_{SP}	SP+1 → SP
T_{11}	E_{INN} E_{SP} W/R VMA	SP → BUS DIREC, A → BUS DATA
T_{12}	I_{SP}	SP+1 → SP
T_{13}	EE_{INN} E_{SP} W/R VMA	SP → BUS DIREC, INDEX → BUS DATA
T_{14}	I_{SP}	SP+1 → SP
T_{15}	E_A E_{SP}W/R VMA	SP → BUS DIREC, A → BUS DATA
T_{16}	I_{SP}	SP+1 → SP
T_{17}	E_B E $_{SP}$W/R VMA	SP → BUS DIREC, B → BUS DATA
T_{18}	I_{SP}	SP+1 → SP
T_{19}	E_{SF} E_{SP} W/R VMA	SP → BUS DIREC, STATUS FLAG → BUS DATA
T_{20}	I_{SP}	SP+1 → SP
T_{21}	EE_{AR} L_{PCC}	AR → PC
T_{22}	EE_{ARR} LL_{PCC}	AR → PC

Come variano i comandi nella istruzione JMS (salto al sottoprogramma) in forma ESTESA.

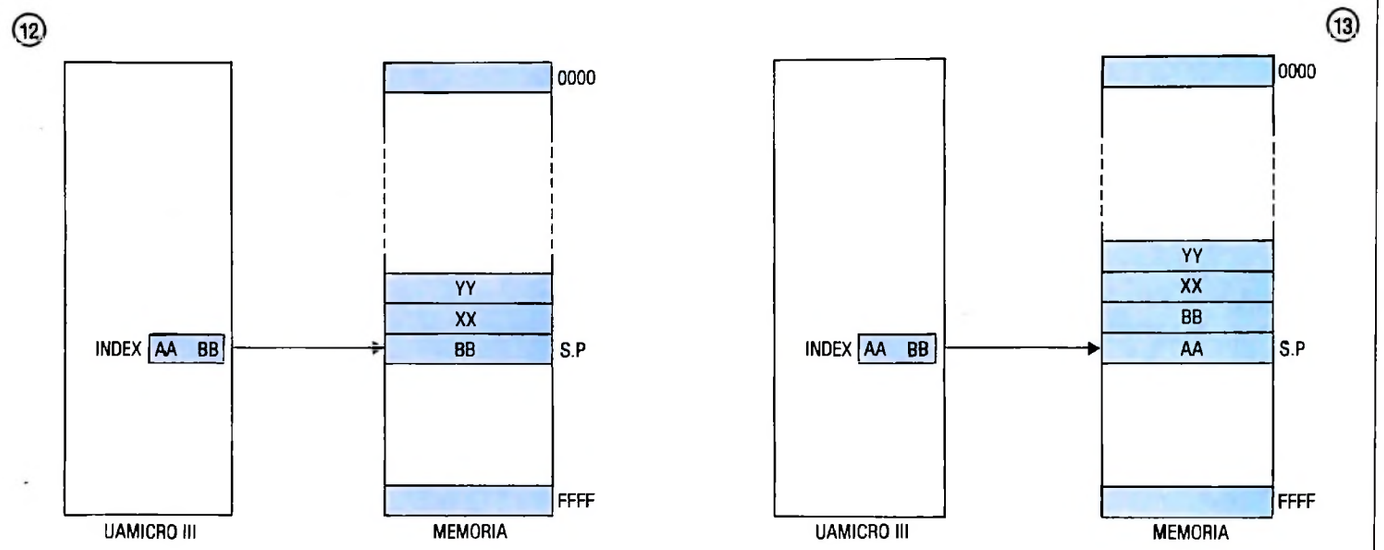
— T_{15} → il contenuto dell'accumulatore "A" occupa il quinto posto dello STACK (figura 14);
— T_{16} → s'incrementa lo S.P.;
— T_{17} → il contenuto dell'accumulatore "B" occupa il sesto posto dello STACK (figura 15);
— T_{18} → s'incrementa lo S.P.;
— T_{19} → il contenuto dello STATUS FLAG occupa il settimo posto dello STACK (figura 16);
— T_{20} → s'incrementa lo S.P.;
— T_{21} → il contenuto della parte destra di A.R. è trasferito nella parte destra del P.C.;
— T_{22} → il contenuto della parte sinistra di A.R è trasferito nella parte sinistra del P.C.

Come possiamo vedere da quest'ultima operazione, il passaggio dei dati da A.R. ai registri interni P.C., INDEX e S.P avviene per 16 bit.
Questo è possibile vederlo anche nella figura dell'Uamicro III, che abbiamo presentato in un precedente articolo, a pagina 525, dove l'INTER. BUS, in corrispondenza di questi 4 registri, diventa di 16 bit.
Quando al termine dell'esecuzione del sottoprogramma s'incontra l'istruzione RTS, inizia il processo inverso per cui prima si decrementa lo S.P. per portarlo all'ultima posizione occupata da un dato, cioè dov'era il contenuto dello STATUS FLAG che viene ricaricato nel suo registro e così di seguito, fino a completare tutti i registri del microprocessore.



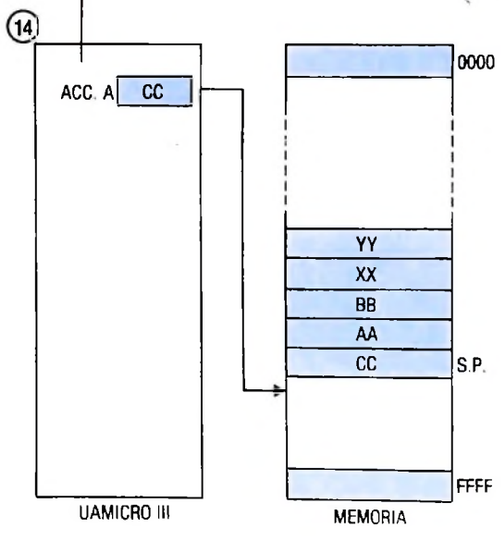
Il valore più basso del P.C. passa alla 1ª casella dello Stack.

Il valore più alto del P.C. passa alla 2ª casella dello Stack.

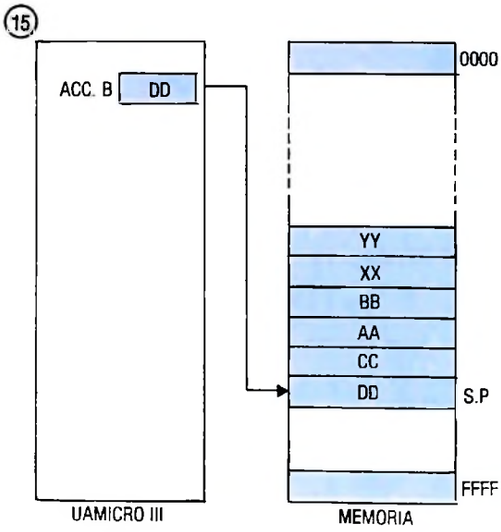


Il valore più basso dell'Index passa alla 3ª casella dello Stack.

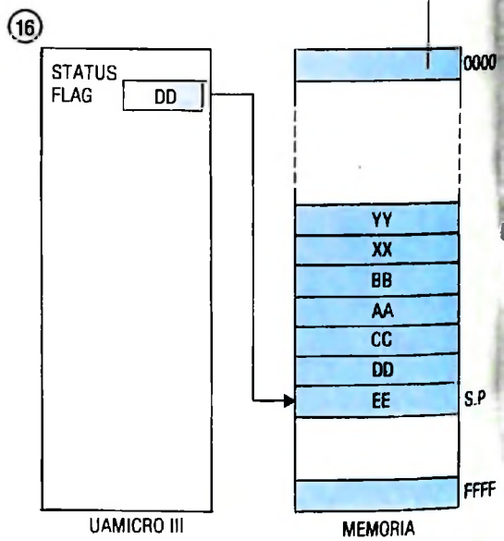
Il valore più alto dell'Index passa alla 4ª casella dello Stack.



Il valore "A" passa alla 5ª casella dello Stack.



Il valore "B" passa alla 6ª casella dello Stack.



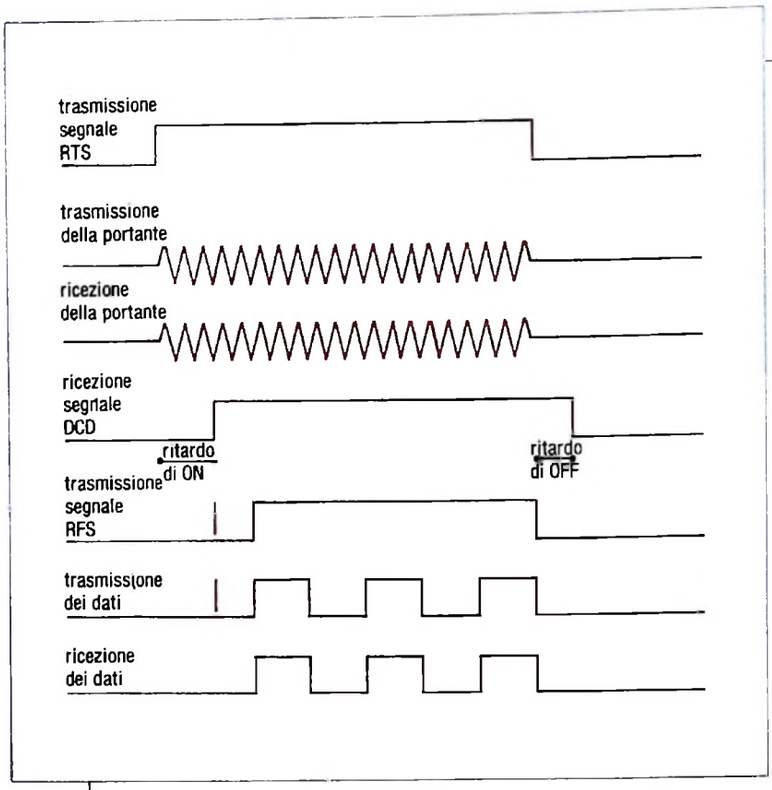
Il valore dello Status Flag passa alla 7ª casella dello Stack.

FUNZIONI DEL MODEM

L'applicazione sulle linee a due fili e gli effetti significativi della trasmissione su quattro fili.

Linea a due fili

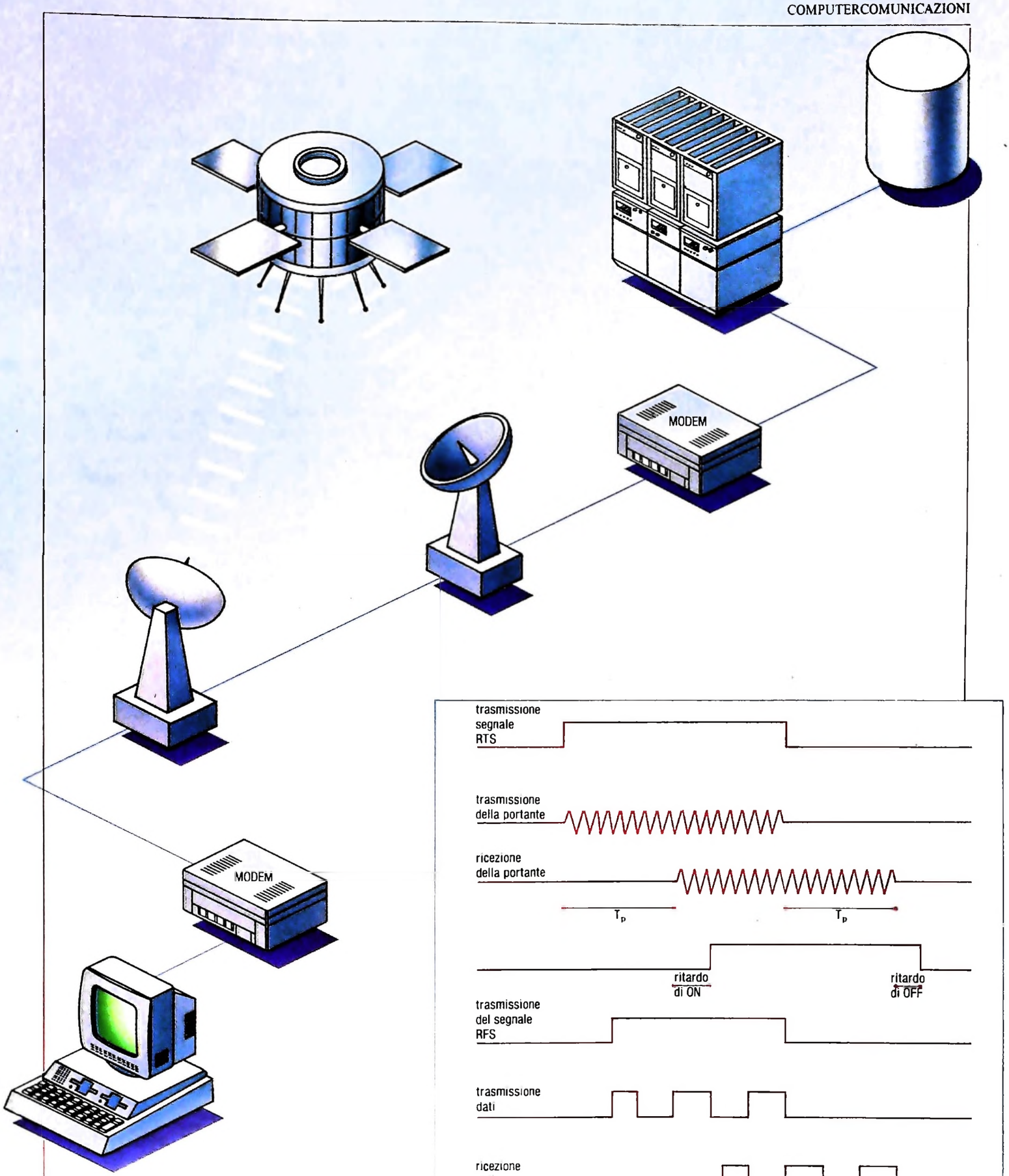
Supponiamo che si stia operando su di una linea a due fili con un terminale bufferizzato ed esaminiamo la sequenza degli eventi che avvengono sulla configurazione della figura della pagina seguente in seguito a un susseguirsi di domande e risposte.
L'operatore registra una transazione e la invia sulla linea. L'elaboratore riceve il messaggio, lo elabora e prepara una risposta che a sua volta viene trasmessa al terminale lungo lo stesso canale di comunicazione.
Se il terminale è dotato di memoria di transito (buffer), l'operatore registrerà i dati della transazione sul terminale e alla fine premerà il tasto di trasmissione.
Se il terminale funziona in modo non controllato, ciò farà alzare (portare a valore logico 1) il segnale RTS sull'interfaccia V24/RS232; in caso contrario (cioè se il terminale funziona in modo controllato), esso non alzerà il detto segnale fino a quando l'elaboratore non interrogherà il terminale (tecnica di *polling*).
Il segnale RTS (richiesta di trasmettere) avvisa il modem che il terminale vuole trasmettere e il modem si approntera alla trasmissione inviando sulla linea telefonica la "portante" che, rilevata dal modem in ricezione, gli permetterà di sintonizzarsi e sincronizzarsi sul segnale. Il modem ricevente può ora demodulare i dati della portante e segnalare quindi questa sua condizione al suo terminale alzando il segnale DCD (rilevamento portante). Il tempo necessario per portare in ON (acceso) il DCD è poi il tempo che occorre effettivamente al modem in ricezione per riconoscere la presenza della portante in arrivo.
Per dare al modem ricevente il tempo per sintonizzarsi sulla portante, le operazioni del modem in trasmissione sono ritardate da un apposito circuito costruito in modo tale da creare un ritardo più lungo del tempo necessario per porre in ON il segnale DCD.
Dopo che è trascorso questo intervallo, il modem in ricezione restituisce il segnale RFS (pronto a inviare) al terminale in trasmissione comunicandogli così che può dar inizio alla trasmissione dati.
Il terminale invia il suo blocco di dati, che viene modulato sulla portante dal modem in trasmissione. La portante modulata viaggia lungo la linea ed è demodulata dal modem in ricezione, il quale passa i dati al terminale che interfaccia.
Quando il terminale in trasmissione ha completato la trasmissione del blocco di dati, abbassa il segnale RTS (richiesta di invio) e quindi fa sì che il modem in trasmissione faccia cadere sia la portante che il segnale RFS. All'estremo in ricezione, il modem vede sparire sia i dati che la portante e dopo un breve tempo di ritardo, il modem in ricezione abbassa il segnale DCD. Questo ritardo è detto ritardo di *off* (spento) del DCD e dipende da un circuito interno che consente al modem di continuare a superare eventuali momentanee cadute di portante senza essere costretto a notificare al terminale dati la perdita della stessa.
Trascorsi alcuni millisecondi, l'elaboratore "capirà" di aver ricevuto un messaggio da elaborare. Questo intervallo viene chiamato *tempo di reazione dell'elaboratore.*
L'elaboratore quindi elabora il messaggio e prepara una risposta da inviare al terminale.
Il tempo che trascorre dalla fine del tempo di reazione all'inizio della trasmissione del messaggio di risposta viene chiamato *tempo di elaborazione* (oppure anche *computer turnaround time*, che significa tempo di permanenza del messaggio nell'elaboratore).
Per far restituire la risposta al terminale, si ripete l'intero processo nella direzione opposta, invertendo i ruoli del trasmittente e del ricevente.
Il circuito di ritardo costruito nel modem in trasmissione viene chiamato ritardo READY FOR SENDING per la V24 o ritardo CLEAR TO SEND per la RS 232 ed è il ritardo del pronto a trasmettere o più comunemente *turnaround time* del modem. Nel corso dell'intera sequenza, come visto, si verificano due *turnaround time* del modem per ogni transazione.
In realtà sul sistema esisterà probabilmente qualche tipo di meccanismo per l'individuazione e la correzione automatica dell'errore in modo da verificare e riconoscere ogni blocco di dati trasmesso, e poterne così assicurare la corretta ricezione. Ciò significa che la sequenza di eventi si svolge nel modo seguente: l'operatore registra una transazione sul terminale e preme il tasto di trasmissione che fa trasmettere i dati lungo la linea dopo che è trascorso il *turnaround time* del modem. Quando l'elaboratore riceverà il messaggio, ne controllerà la validità e risponderà con un segnale di riconoscimento trasmesso appena trascorso il *turnaround time* del modem collegato all'elaboratore stesso. Quindi il viceversa.



Tempificazione dell'attività dei modem in un sistema a due fili con un ritardo di propagazione nullo (a sinistra). A lato: diagramma dei tempi con significativo ritardo di propagazione. In questo caso, il modem ricevente sarà sincronizzato per tutto il tempo che intercorre perché i dati lo raggiungano.

Trasferimento blocco a blocco

È assai frequente che i sistemi remoti di registrazione dei dati trasmettano un gran numero di blocchi dati in modo consecutivo da un punto all'altro. In tal caso, in condizione di assenza di errore, gli eventi si susseguono nel modo seguente: il terminale in trasmissione invia un blocco di dati che verrà controllato dal terminale in ricezione per scoprire gli eventuali errori: se il blocco dei dati supera il controllo, esso viene riconosciuto e il terminale in ricezione ordina all'altro terminale di trasmettere il blocco successivo. Ricomincia così la sequenza di operazioni appena vista con la trasmissione e il riconoscimento del blocco successivo, sequenza che verrà ripetuta fino a esaurimento dei dati da trasmettere.
Se la rete di trasmissione ha due fili, si dovranno subire i due *turnaround time* dei modem per ogni blocco trasmesso e l'efficienza della trasmissione dipenderà quindi dalla lunghezza dei blocchi, dai *turnaround time* dei modem e da tutti gli altri ritardi del sistema. Ecco l'elenco dei ritardi che si possono verificare in un sistema di trasmissione:
A) *ritardo di propagazione* (propagation delay) è il tempo richiesto per portare il segnale dall'una all'altra estremità della linea e dura un tempo ben definito per i segnali elettrici che viaggiano lungo una linea di trasmissione dati. Sulle linee terrestri questo ritardo è di circa 6-9 microsecondi per kilometro (in funzione del mezzo trasmissivo usato) e sui collegamenti tramite satellite è di circa 250-300 millisecondi per ogni percorso terra-satellite-terra.
Dalla figura della pagina accanto si può vedere come si modifica la sequenza di eventi indicata nella figura qui sopra, quando sulla linea si verificano apprezzabili ritardi di propagazione.
In questo diagramma deve trascorrere tutto il ritardo di propagazione (Tp) prima che appaia sull'altro estremo qualsiasi

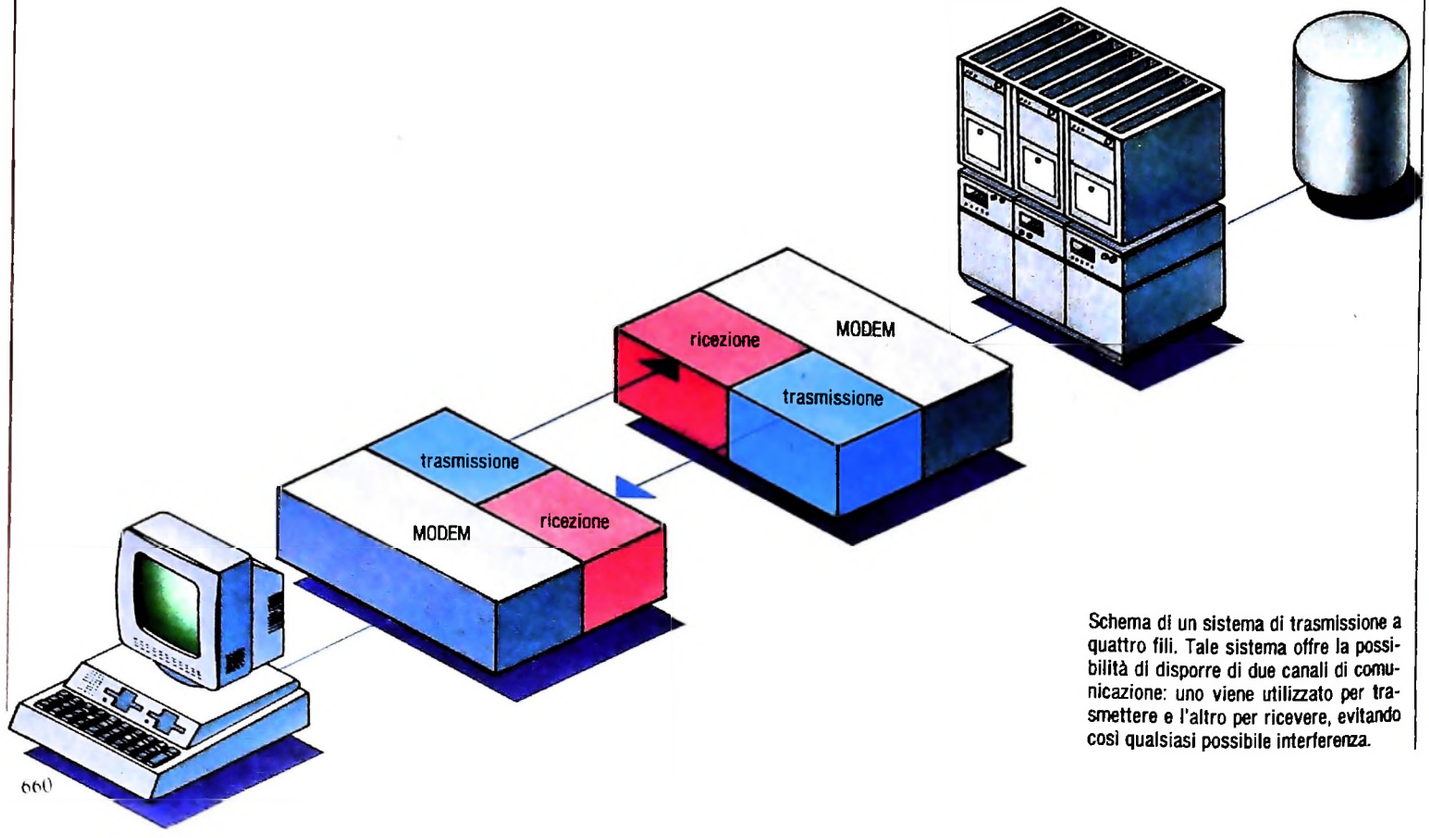

MODEM
MODEM

trasmissione
segnale
RTS
trasmissione
della portante
ricezione
della portante
T_p
T_p
ritardo
di ON
ritardo
di OFF
trasmissione
del segnale
RFS
trasmissione
dati
ricezione
dati
T_p

segnale avviato sulla linea. In particolare si noti come la portante in ricezione sia ritardata di Tp rispetto alla portante in trasmissione.
Il diagramma mostra che il segnale RFS viene restituito al terminale in trasmissione prima che la portante abbia raggiunto il terminale in ricezione. Ciò non costituisce un problema finché il modem in ricezione può sincronizzarsi in un tempo inferiore al ritardo RFS. In altre parole, il modem ricevente sarà sincronizzato per tutto il tempo in cui i dati lo raggiungeranno.
B) *ritardo del modem* (modem delay) è il tempo che passa tra il momento in cui il segnale digitale viene presentato all'interfaccia V24/RS-232 e il momento in cui la portante modulata appare sulla linea.
Quando il modem demodula il segnale in entrata si verifica un ritardo analogo.
C) *tempo di reazione* (reaction time) del terminale o dell'elaboratore a ogni capo della linea è il tempo richiesto perché il terminale o l'elaboratore stesso capisca di aver ricevuto un dato e che quindi deve inviare un segnale di riconoscimento o che deve svolgere una qualche altra azione sui dati.
D) *altri ritardi* possono essere generati nella rete di trasmissione dal tipo di componenti impiegati per costruire la rete.

La trasmissione punto a punto su quattro fili

Il *turnaround time* del modem può avere un effetto significativo sul *throughput* (quantità dei dati elaborati nell'unità di tempo) dei dati e sul tempo di risposta del sistema.
Per minimizzare questi effetti, possiamo costruire la rete usando linee di trasmissione a quattro fili secondo lo schema riportato qui sotto.
La linea a quattro fili offre effettivamente due canali di comunicazione e ciò permette di trasmettere su un canale e ricevere sull'altro. In tal modo ogni modem può tenere la portante in continuità essendo le portanti su canali diversi e queste non interferiranno l'una con l'altra.
In questo schema il terminale che desidera trasmettere dati può mandarli nel modem che li invierà direttamente sulla linea senza perdere il tempo di turnaround.
Spesso si configura la rete in questo modo perché la maggior parte delle linee a grande distanza delle reti telefoniche esce dalla rete stessa a quattro fili. E poiché la rete offre questa possibilità vale la pena sfruttarla con la massima efficienza (in alcuni sistemi, gli elaboratori e i terminali sono costruiti in modo tale da poter accettare una risposta immediata dal modem, poiché devono eseguire la sequenza già vista, prima del segnale RTS. Per ovviare a questo inconveniente si inserisce un falso ritardo all'interno del sistema in modo da evitare problemi per il terminale anche se il modem può tenere sempre attiva la portante).
Una linea a quattro fili come quella della figura in basso ci dà la possibilità di utilizzare in full-duplex la linea: in tal modo l'insieme "modem e linea a quattro fili" è in grado di ricevere e trasmettere dati simultaneamente. Tuttavia l'intero sistema "terminale, modem/linea/modem ed elaboratore" sarà un sistema full-duplex solo se il terminale e l'elaboratore sono capaci di trasmettere e ricevere simultaneamente. Capita invece spesso che si colleghino terminali half-duplex ad elaboratori con linee a quattro fili: in tal caso l'intero sistema è un sistema in half-duplex.



Schema di un sistema di trasmissione a quattro fili. Tale sistema offre la possibilità di disporre di due canali di comunicazione: uno viene utilizzato per trasmettere e l'altro per ricevere, evitando così qualsiasi possibile interferenza.

Lezione 41

Usare il microplotter

Nella lezione precedente abbiamo visto che cosa è un plotter, e come si può predisporre il microplotter PL10 per l'M10. Vediamo ora come è possibile pilotarne l'uso per ottenere stampe o effetti grafici nella programmazione in BASIC.

L'istruzione LPRINT

In BASIC l'istruzione che permette di inviare comandi al microplotter è LPRINT, che abbiamo già visto per l'invio di dati alla stampante.
Questa istruzione viene usata con due differenti modalità:

- quando si opera in TEXT MODE, per indicare che cosa deve essere visualizzato dal microplotter, esattamente come se questo fosse una normale stampante, nello stesso modo in cui abbiamo utilizzato l'istruzione PRINT per visualizzare le informazioni sul video;
- quando si opera in GRAPHIC MODE, per inviare al microplotter comandi che permettono di muovere i pennini, descrivere immagini o stampare un testo.

Il microplotter, appena viene acceso, è naturalmente predisposto a funzionare in "text mode", cosicché istruzioni come

```
10 PRINT "MESSAGGIO"
```

provocano la visualizzazione sulla carta di

```
MESSAGGIO
```

nel colore nero, colore su cui avviene automaticamente il posizionamento al momento dell'accensione.
Per far passare il funzionamento dalla modalità "testo" a quella grafica, è necessario inviare opportuni comandi al microplotter, sempre mediante l'istruzione LPRINT.
Ovviamente ci troviamo di fronte a un problema: come è possibile infatti fare sì che il microplotter capisca quali caratteri devono essere interpretati come elementi di stringhe da visualizzare e quali come comandi diretti proprio a lui per modificarne il comportamento? È semplice: allo scopo vengono adottati caratteri "invisibili", che non sono presenti, in genere, sulle tastiere dei calcolatori, e viene usata la funzione BASIC di nome CHR$.
Ogni carattere viene rappresentato all'interno del calcolatore mediante un opportuno codice numerico. Per esempio, come si può vedere dalla tabella dei caratteri ASCII (ASCII è la sigla di American Standard Code for Information Interchange, ed è il nome di una delle possibili forme convenzionali di rappresentazione interna dei caratteri), la lettera A è rappresentata con il valore 65. La funzione CHR$ trasforma un codice numerico nel carattere corrispondente, cosicché la scrittura

```
10 LPRINT "A"
20 LPRINT CHR$ (65)
```

Rappresentazione dei caratteri stampabili

Ogni carattere, all'interno del calcolatore, è rappresentato con un opportuno codice numerico. Il codice adottato dall'M10 è detto ASCII (American Standard Code for Information Interchange); la seguente tabella riporta il codice di rappresentazione interna di tutti i caratteri stampabili.

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

provoca la visualizzazione di

A
A

con la trasformazione, cioè, del codice 65 nella lettera A.
Tuttavia sono disponibili altri caratteri non visualizzabili sul video o addirittura

non presenti come tasti sulla tastiera.
Per esempio, se voi volete inserire in una stringa BASIC il carattere di "ritorno a capo" , non sarete in grado di farlo con la semplice pressione del tasto RETURN, che viene considerato come "fine istruzione"; potete però usare un'istruzione che richieda, mediante la funzione CHR$, di inserire il carattere che corrisponde alla codifica interna 10, che è proprio l'"a capo".

È possibile operare in modo analogo per inviare comandi al microplotter: per esempio, la scrittura

```
10 LPRINT CHR$ (18)
```

invia al microplotter il carattere la cui codifica interna è 18, che non corrisponde ad alcun carattere stampabile, ma che ha l'effetto di far passare il microplotter in "graphic mode".
Alcuni di questi caratteri non stampabili hanno significato solo se inviati al microplotter.
Riportiamo in tabella l'elenco di tutti i caratteri "speciali" che possono essere usati quando si voglia inviare un comando al microplotter; si noti che, al di fuori di quello che fa passare da "graphic mode" a "text mode", tutti gli altri possono essere usati solo quando si opera in modo "testo", essendo disponibili comandi differenti per il modo "grafico".

Codici di controllo per il microplotter

Quando si opera in modo "testo" è possibile fare compiere al microplotter alcune operazioni inviandogli caratteri "invisibili" che è in grado di interpretare come comandi a sé destinati; riportiamo qui di seguito i codici di tali caratteri, che devono essere inviati mediante la funzione di conversione CHR$.
Si noti che tra essi il codice 17 corrisponde alla richiesta di riportare il microplotter in modo "testo", ed è l'unico che può essere usato quando il microplotter è in modo "grafico".

CHR$(08) — Backspace; sposta il portapennino indietro di uno spazio, e può essere usato, per esempio, in un testo sottolineato.

CHR$(10) — Line Feed; fa avanzare il carrello di una linea.

CHR$(11) — Line-Feed; sposta il carrello indietro di una linea.

CHR$(13) — Ritorno Carrello; sposta il carrello sul margine sinistro.

CHR$(17) — modo Testo; questo comando seleziona il microplotter a funzionare in modo Testo.

CHR$(18) — modo Grafico; questo comando seleziona il microplotter a funzionare in modo Grafico nel quale tutta una serie di comandi sono validi.

I comandi in modo "Grafico"

Una volta che siamo passati in modo grafico, l'istruzione LPRINT potrà inviare un comando per volta racchiuso tra virgolette.
Tale comando sarà caratterizzato da:
- una prima lettera che indica il comando
- una serie di informazioni che lo specifica.

Per esempio, per visualizzare una stringa di caratteri una volta che siamo in modo grafico, non potremo più operare come avevamo fatto precedentemente: infatti la scrittura

```
10 LPRINT CHR$ (18) 'Graphic mode
20 LPRINT "MESSAGGIO"
```

interpreta la M di messaggio come specificazione di comando di "movimento del pennino" (lo vedremo meglio in seguito), e si aspetta di trovare dietro indicazioni su tale movimento; la parola "MESSAGGIO" non viene affatto visualizzata.
Per ottenere ciò dovremo anteporre alla stringa "MESSAGGIO" il carattere P che richiede proprio la stampa:

```
20 LPRINT "PMESSAGGIO"
```

e che fa ottenere la visualizzazione di

```
MESSAGGIO
```

Alcuni di questi comandi hanno un effetto immediato cioè provocano uno spostamento del pennino, oppure il tracciamento di una linea, e così via; altri hanno l'effetto di "cambiare" lo stato del microplotter, con il risultato di influenzare l'effetto dei comandi che seguiranno (per esempio, sarà possibile definire la dimensione dei caratteri da visualizzare, che verrà "adottata" da tutti i comandi successivi che richiederanno le stampe di caratteri, fino a un nuovo cambiamento di dimensione).
Vedremo nelle prossime lezioni quali altri comandi siano disponibili per pilotare il microplotter in "graphic mode".

Cosa abbiamo imparato

In questa lezione abbiamo visto:

- l'istruzione LPRINT per comunicare con il microplotter;
- la funzione CHR$ per comunicare "comandi" al microplotter;
- come porre il microplotter in modo "grafico";
- come stampare stringhe con il comando P in modo grafico.

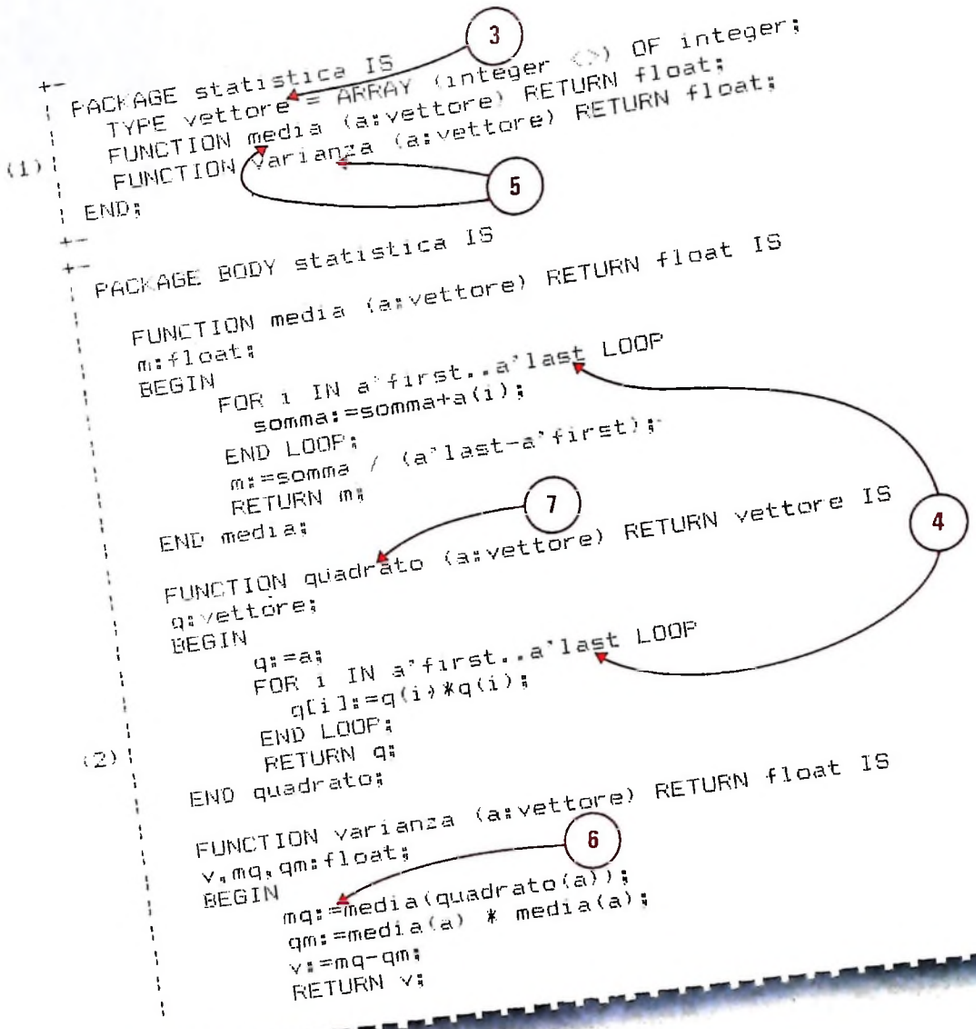
Il LINGUAGGIO ADA

Un linguaggio ad alto livello con caratteristiche di portabilità e orientato a gestire problemi di "concorrenza".

Negli anni Settanta il Dipartimento della Difesa degli Stati Uniti calcolò che la maggior parte delle spese per il software, veniva impiegata per il "trasporto" (la traduzione) di programmi da una macchina a un'altra. Se per esempio un programma, già esistente, scritto in FORTRAN avesse dovuto funzionare su una macchina priva del compilatore FORTRAN, la sua traduzione in un altro linguaggio avrebbe comportato una perdita di tempo molto vicina a quella necessaria per riscriverlo "ex novo".
Si stabilì inoltre che all'interno dello stesso Dipartimento si utilizzavano più di trecento tra linguaggi e "dialetti". Nacque da queste considerazioni la necessità di disporre di un linguaggio che potesse imporsi come "standard" e che avesse la caratteristica fondamentale della completa indipendenza di un programma dalla struttura interna della macchina utilizzata per l'esecuzione. Inoltre il sistema informativo del Dipartimento era orientato per lo più ad applicazioni militari, e quindi un altro requisito richiesto al linguaggio era l'affidabilità del sistema nel far fronte a situazioni esterne e a risolverle in tempo "reale", quindi, la sua capacità nel gestire la "concorrenza" nell'esecuzione di più programmi.
Il Dipartimento della Difesa indisse una gara per la definizione di un linguaggio che rispondesse alle caratteristiche su esposte.
Il vincitore è risultato un linguaggio ad alto livello con caratteristiche di portabilità e orientato a gestire problemi di "concorrenza".
Questo linguaggio è stato dedicato alla prima programmatrice della storia: Ada, contessa di Lovelace, figlia di Lord Byron, e assistente di Babbage, costruttore nei primi decenni del 1800 del precursore dei calcolatori. ADA è un linguaggio che pur non avendo alla base idee particolarmente innovative e rivoluzionarie, riunisce in un amalgama i pregi dei linguaggi più validi.
Inoltre ADA è un linguaggio orientato allo sviluppo di applicazioni (normalmente piuttosto grandi, per poter sfruttare a pieno le sue potenzialità) in modo industriale.
Tali applicazioni hanno in generale la proprietà di contenere al loro interno delle attività che sono comuni a più elaborazioni. Per esempio, è molto probabile che più programmi debbano fare calcoli statistici: è inutile e pericoloso replicare questo codice in tutti i programmi, e ancora peggio lasciare che ogni programmatore si scriva la propria routine, con il risultato di avere risultati diversi a fronte di dati in ingresso uguali.
ADA offre la possibilità di scrivere nel codice utilizzabile da tutti gli utenti e in particolare tutti i programmi possono includere al loro interno le funzioni e le procedure interne a una struttura chiamata *package*.
Il package permette, quindi, di costruire una "libreria" di programmi alla quale tutti possono accedere.
Da un punto di vista sintattico un programma ADA è abbastanza simile a un programma Pascal: infatti ADA può essere visto, in prima approssimazione, come un'estensione e una sofisticazione del Pascal.

Caratteristiche generali

Prima di esaminare in dettaglio le caratteristiche del linguaggio ADA diamo un'occhiata al programma (figura a pagina seguente), che chiarisce il concetto package già visto.
Si nota immediatamente che il package è diviso in due parti: la prima parte (parte 1) è la parte "specifica" nella quale vengono definiti i tipi, le funzioni, le procedure che compongono il package; da notare che questa parte contiene solo le definizioni senza alcuna notizia di come vengono poi effettivamente sviluppate le varie parti. Questo modulo ha una notevole importanza descrittiva e di documentazione del prodotto.
Nella seconda parte (parte 2) vengono sviluppate le dichiarazioni fatte nella prima: è da sottolineare che le due parti possono essere moduli di compilazione separati con l'evidente vantaggio che la correzione di errori, per esempio nella parte body, comporta la ricompilazione della stessa e non di tutto il programma.
Analizziamo in dettaglio il programma:
— nella definizione del tipo vettore (3) notiamo una prima fondamentale differenza dal Pascal: infatti non è indicato il numero di elementi dell'array "vettore" che può, quindi essere variato dinamicamente.
— Il ciclo di FOR con gli attributi first e last (4) permette l'uso delle funzioni media e quadrato con un array di numero di elementi qualunque perché il ciclo viene indicizzato dal primo all'ultimo elemento dell'array stesso.
— Come abbiamo già visto le funzioni media e varianza (5)



```
PACKAGE statistica IS
  TYPE vettore = ARRAY (integer <>) OF integer;
  FUNCTION media (a:vettore) RETURN float;
  FUNCTION varianza (a:vettore) RETURN float;
END;

PACKAGE BODY statistica IS

  FUNCTION media (a:vettore) RETURN float IS
  m:float;
  BEGIN
       FOR i IN a'first..a'last LOOP
         somma:=somma+a(i);
       END LOOP;
       m:=somma / (a'last-a'first);
       RETURN m;
  END media;

  FUNCTION quadrato (a:vettore) RETURN vettore IS
  q:vettore;
  BEGIN
       q:=a;
       FOR i IN a'first..a'last LOOP
         q(i):=q(i)*q(i);
       END LOOP;
       RETURN q;
  END quadrato;

  FUNCTION varianza (a:vettore) RETURN float IS
  v,mq,qm:float;
  BEGIN
       mq:=media(quadrato(a));
       qm:=media(a) * media(a);
       v:=mq-qm;
       RETURN v;
```

(1) (2) 3 4 5 6 7

Prima e seconda parte del package: nella prima, vengono definiti i tipi, le funzioni, le procedure; nella seconda, vengono sviluppate le dichiarazioni fatte.

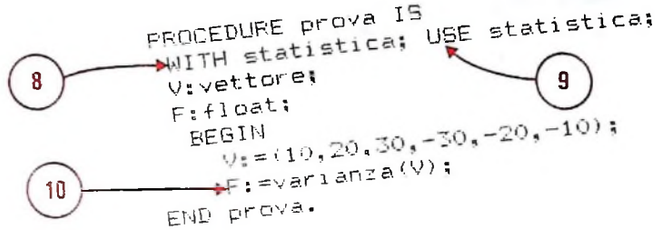
appaiono la prima volta in fase dichiarativa senza nessuna informazione sulla loro implementazione: si sa solo che entrambe hanno bisogno di un parametro del tipo "vettore" definito come array di interi e che restituiscono un valore floating point.

— L'assegnamento mq:= media (quadrato (a)) (6) assegna alla variabile mq il risultato della funzione media alla quale viene passato come parametro il risultato della funzione quadrato (che è un vettore con gli elementi uguali al quadrato degli elementi dell'array a passato come parametro).

— È da notare, inoltre, che la funzione quadrato (7) non è dichiarata nella parte specifica, e non a caso; infatti questa funzione è interna al package "statistica" e non è visibile, e quindi utilizzabile, da altri programmi (a differenza di media e varianza).

Le funzioni dichiarate nella parte "specifica" del package "statistica" sono, come abbiamo visto, utilizzabili da tutti gli altri programmi.

Per esempio un programma del tipo:



```
PROCEDURE prova IS
WITH statistica; USE statistica;
V:vettore;
F:float;
  BEGIN
    V:=(10,20,30,-30,-20,-10);
    F:=varianza(V);
END prova.
```

8 9 10

dichiara con il termine WITH (8) l'incapsulamento al suo interno del package "statistica", compilato separatamente, e con il termine USE (9), ADA offre la possibilità di utilizzare gli stessi nomi definiti in "statistica"; l'omissione di USE comporta la necessità di utilizzare la notazione DOT, cioè l'istruzione

F:=media (V)

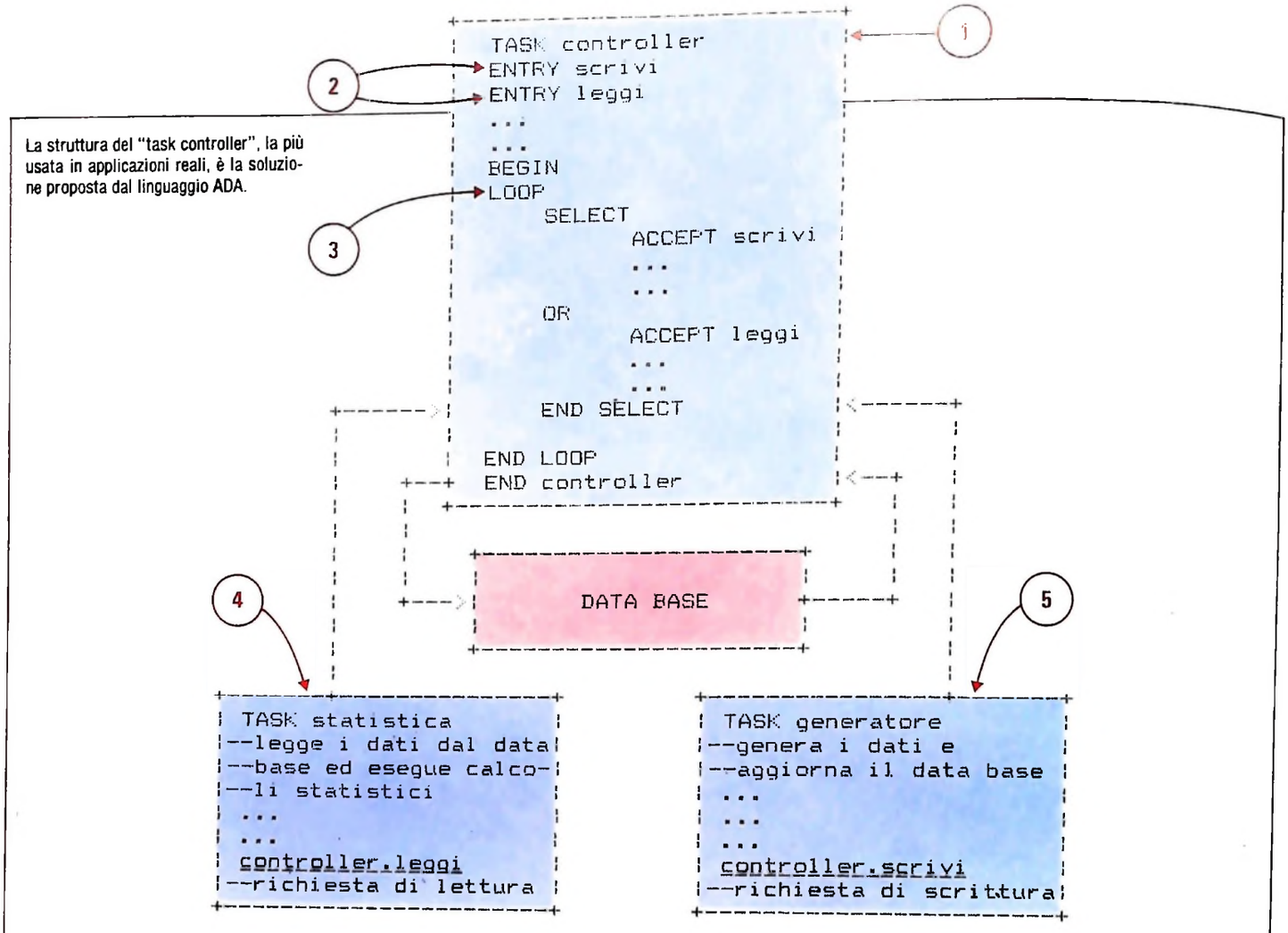
andrebbe scritta:

F:=statistica. media (V)

appesantendo il programma.
L'assegnamento F:=media (V) richiama la funzione "media" e assegna il suo risultato a F (10).
Vediamo le caratteristiche generali del linguaggio ADA:
— Strutture dati.
Il linguaggio ADA è basato su una definizione di tipi come il Pascal anche se con delle estensioni, abbiamo già parlato, per esempio, della definizione di array dinamici.
— Strutture di controllo.
Anche le strutture di controllo sono abbastanza simili al Pascal con delle estensioni orientate a gestire i problemi di concorrenza (le vedremo più dettagliatamente in seguito).
— Visibilità.
Sono visibili e utilizzabili da parte di tutti gli utenti le funzioni e le procedure sviluppate in altri package ma dichiarate nelle parti "specifica". Non è quindi possibile arrivare a funzioni "interne" di un package (vedi la funzione quadrato dell'esempio precedente).
All'interno delle funzioni e delle procedure valgono le stesse regole di visibilità delle risorse (scope-rules) del PASCAL.
— Sviluppo Top-Down.
Essendo ADA un linguaggio indirizzato allo sviluppo di sistemi di notevoli dimensioni, ne facilita la costruzione mettendo a disposizione tutti gli strumenti per la programmazione "per raffinamenti successivi".
Le caratteristiche più importanti sono: la possibilità di compilazioni separate e l'uso da parte di tutti dei "package" già scritti.

Gestione della concorrenza

Spesso la complessità di un'applicazione porta alla necessità di suddividere l'applicazione stessa in più attività.
L'esecuzione di queste attività non deve essere, però, sequenziale perché l'evoluzione di ognuna di esse è strettamente connessa a quella delle altre.
Un esempio tipico è quello della gestione di un buffer, che deve da un lato "ricevere" caratteri da un "alimentatore" (per esempio una periferica di ingresso dati) e dall'altro deve "cederli" a un "lettore" (per esempio un programma che elabora le informazioni lette).
In questo caso è necessario che esistano due processi (ovvero due programmi in esecuzione), che provvedono rispettivamente a fornire caratteri (processo "produttore") e a prelevarli (processo "consumatore").
Le operazioni di registrazione e di lettura devono avvenire in momenti separati, onde evitare che i due processi operino su una situazione ambigua (il consumatore non può prelevare ciò che non è ancora stato "prodotto").
Si dice in questo caso che i due processi "concorrono" alla stessa "risorsa": l'uno infatti cerca di accedere al buffer per registrare caratteri e l'altro per prelevarli, ma la risorsa in ogni singolo istante può essere attribuita solo all'uno o all'altro, ma mai a entrambi contemporaneamente.
Poiché d'altrô lato i due processi sono indipendenti l'uno dall'altro e in particolare rispondono a stimoli "esterni" (una periferica di ingresso attivata innesca il processo produttore, un'operazione di lettura da un programma innesca il processo consumatore) occorre individuare un meccanismo che, essendo estraneo ai due, garantisca la corretta attribuzione della risorsa.
Questo è un problema tipico della costruzione in particolare di sistemi operativi, nei quali la risorsa che più tipicamente deve essere condivisa è la CPU. Questo problema comporta anche la gestione dell'attesa del processo o dei processi che richiedono una risorsa "occupata".
Infatti nel caso in cui per esempio il processo consumatore venga innescato mentre il buffer è attribuito al processo produttore, il primo deve restare in attesa fino a quando la risorsa viene liberata.
Questo tipo di problemi ha trovato differenti soluzioni nella storia dei linguaggi di programmazione: vedremo qui la soluzione proposta dal linguaggio ADA.
In certe applicazioni (ormai le più numerose) è anche necessario che il sistema riesca ad affrontare ed effettuare le giuste azioni in "tempo reale" a fronte di eventi imprevisti.
Chiariamo questi concetti con un esempio: il controllo di una centrale elettrica.
In condizioni normali il sistema controlla i vari organi periferici, effettua calcoli statistici, risponde alle interrogazioni degli operatori (questa attività corrisponde proprio all'idea di "processi concorrenti").
In questa situazione più programmi sono contemporaneamente in esecuzione, cioè più processi sono attivi (associeremo sempre a "processo" l'idea della dinamicità e dell'evoluzione, in particolare un processo è un'entità "viva" che concorre all'uso delle risorse della macchina tra le quali, la più importante, è la CPU) e tutti concorreranno, in base alla loro priorità, all'utilizzo della CPU (utilizzeremo il termine task come sinonimo di processo).
Analizziamo l'esempio (molto semplificato) riportato nella figura della pagina seguente.
Anche se non è obiettivo dell'esempio entrare nel dettaglio, la struttura del task "controller" (1) è molto usata in applicazioni reali.
In essa si possono notare le dichiarazioni "ENTRY leggi" ed "ENTRY scrivi" (2) che definiscono quali "servizi" sono messi a disposizione degli altri task. Il corpo del task è costituito da un loop di attesa richieste (3).
Il primo dei processi (tra statistica e generatore) che effettua la richiesta viene servito perché la SELECT non è deterministica e i due processi richiedenti hanno la stessa priorità.



La struttura del "task controller", la più usata in applicazioni reali, è la soluzione proposta dal linguaggio ADA.

Inoltre essendo la SELECT condizionata su una OR, mentre il controller soddisfa una richiesta, un'eventuale altra richiesta non può essere soddisfatta: le due azioni sono cioè "mutuamente esclusive".

Se un processo legge dei dati da un data base e contemporaneamente un altro processo modifica quelli appena prelevati, i risultati dell'elaborazione effettuata con i dati "vecchi" non sono in sintonia con il data base modificato e possono portare a gravi errori.

È evidente, quindi, che per avere una situazione consistente del data base bisogna evitare che un processo vada a scrivere quando un altro sta leggendo e viceversa: questo è realizzato nell'esempio affidando il ruolo di controllore del data base a un unico task che accoglie e risolve le richieste degli altri processi facendo quindi da tramite tra i vari processi e il data base.

Abbiamo quindi, nell'esempio un task "controller" che sostanzialmente è sempre in attesa di accettare lavoro dal task statistica (4) o da quello generatore (5).

Se il processo statistica chiede di leggere dei dati dal data base, il task controller esegue la parte di codice relativa.

La soluzione del problema della concorrenza alla base dati è dunque quella di attribuire una funzione di supervisione a un processo che vaglia tutte le richieste di accesso alla risorsa condivisa e garantisce che solo una alla volta sia soddisfatta.

Il task controller è dunque continuamente in attesa di richieste: quando uno dei due task effettua la richiesta e il task controller è libero e disponibile a soddisfarla si realizza il cosiddetto RENDEZ-VOUS, cioè il task che ha effettuato la richiesta cede il controllo al task controller, che provvede a soddisfarla e ripassa quindi il controllo al task che l'ha richiamato.

Inoltre se contemporaneamente il programma generatore fa pervenire la sua richiesta al controller questo non può soddisfarla e il generatore deve quindi sincronizzarsi a sua volta su un evento (la liberazione da parte del controller).

Abbiamo visto nell'esempio della centrale elettrica una situazione di funzionamento normale; infatti se si presenta una situazione anomala, per esempio un trasformatore si brucia o una linea non riesce a supportare le richieste, il controllo deve passare immediatamente a un processo che deve tempestivamente far fronte alla nuova situazione.

Nel linguaggio ADA l'utente ha la possibilità di scrivere i programmi che devono essere invocati a fronte di eventi esterni: è l'utente che decide come trattare e risolvere delle sollecitazioni esterne.

I programmi che vanno in esecuzione al verificarsi di particolari condizioni hanno la massima priorità e il processo che in quel momento è possessore della CPU deve lasciarla per fargli posto.

ATTIVITÀ D'UFFICIO E AUTOMAZIONE

L'introduzione del computer negli uffici ha trasformato non solo le mansioni, ma anche l'ambiente di lavoro: si sta andando verso un ufficio senza carte.

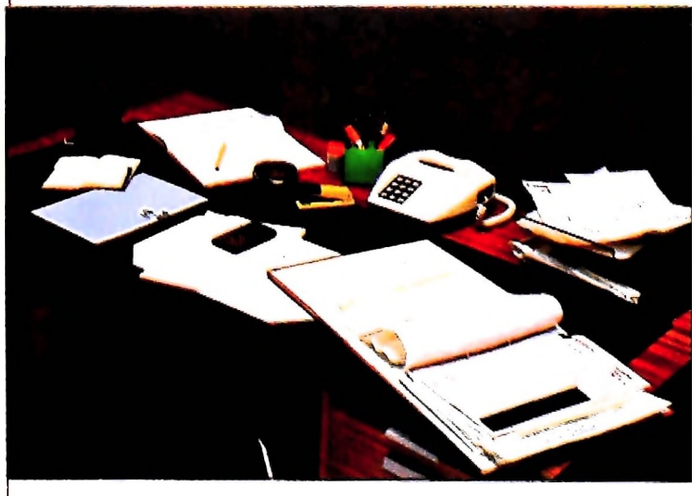
Da oltre un decennio la penetrazione degli strumenti elettronici all'interno del lavoro di ufficio è in costante aumento; a ciò si accompagna un incremento in senso qualitativo della efficacia e della flessibilità di tali strumenti. È possibile verificare questo con facilità: basta osservare il numero di documenti, scontrini, ricevute e fatture, prodotti con tecnologia elettronica, che in una giornata di lavoro o di acquisti cade nel nostro raggio di azione. In precedenza, soltanto aziende con fatturato notevole si potevano permettere l'acquisto di un computer per il disbrigo delle faccende di amministrazione e per la gestione di magazzini.
La diffusione dei microelaboratori ha reso ora possibile la meccanizzazione di contabilità e gestione di archivi anche in aziende a carattere familiare.
Se le piccole aziende e attività mostrano un netto interesse per l'utilizzo di una macchina che svolga un servizio sostanzialmente limitato, l'attuale tendenza nelle aziende maggiori indica una strada differente: è in atto una trasformazione del lavoro d'ufficio con l'integrazione del calcolatore in tutte le fasi. Si parla quindi di automazione dell'ufficio, di ufficio del futuro o ufficio senza carta. L'assenza di carta è naturalmente da vedersi come un ulteriore passo verso l'obiettivo finale di affidare il lavoro di routine a un calcolatore, in modo da elevare la qualità del lavoro nell'ufficio. Il calcolatore inoltre sarà in grado di fornire all'impiegato o al manager il dovuto supporto anche in attività di tipo prevalentemente umano come creare, comunicare, pensare e prendere decisioni.

Situazione e tendenze

L'automazione del lavoro di ufficio è attualmente uno dei campi in cui è più accesa la concorrenza fra le industrie alla ricerca di ampi spazi di mercato. La presenza di un alto numero di industrie ha portato a una progressiva e rapida evoluzione dei prodotti proposti per questo settore. Le tecnologie e le idee più innovative sviluppate nei laboratori di ricerca delle Università, e in particolare nel campo dell'Intelligenza Artificiale, hanno trovato una naturale collocazione nell'area del lavoro di ufficio. Specificatamente gli studi sulle interfacce uomo-macchina, di cui si è parlato in un precedente articolo, hanno portato allo sviluppo di calcolatori, il cui capostipite è lo STAR della Xerox, particolarmente semplici da utilizzare.
Ciò ha permesso un approccio più intuitivo e naturale alle tecniche informatiche anche da parte di utenti tradizionalmente abituati a usare strumenti come carta e matita.
L'utilizzo poi di strumenti informatici nel lavoro di ufficio ha trasformato alcune figure professionali, mutandone le caratteristiche ed il tipo di competenze: a una segretaria, per esempio, è sempre più spesso richiesta la conoscenza degli elaboratori e per una dattilografa è divenuto importante essere in grado di utilizzare un word-processor.

Computer e uffici

Di fronte a una tendenza di questo tipo, è importante esaminare quali siano le funzioni che sono ritenute fondamentali in un sistema integrato per l'automazione degli uffici. Sulla base dell'analisi che segue, sarà possibile avere un buon punto di vista per valutare i sistemi attualmente in uso. In una pubblicazione del 1980, *The future of the office of the future*, H. Morgan schematizzava in questo modo le attività che in un ufficio vengono espletate da impiegati e manager.
1) Atti di comunicazione (sia interpersonale che tra la propria e le altre aziende).
2) Acquisizione, schedatura, immagazzinamento e reperimento di informazione.
3) Analisi di dati, loro trasformazione e, su tale base, attività decisionale.
4) Assistenza personale.
5) Organizzazione del lavoro.
Ciascuna di queste attività può essere realizzata essendo assistiti da un calcolatore, o più generalmente da strumenti elettronici. L'espressione "meccanizzazione degli uffici" intende indicare la presenza in un ufficio di ausilii elettronici che permettano lo svolgimento di uno o più dei compiti suddetti. Quando si parla di "automazione degli uffici", invece, si vuole intendere la presenza nell'ufficio stesso di un sistema che integri in sé le funzioni descritte, e che diventi l'unico strumento di lavoro.



L'introduzione del computer negli uffici sta rivoluzionando il sistema di lavoro e l'ambiente stesso. La scrivania 'cosparsa' di carte è sostituita da una scrivania elettronica che, collegata con una rete Ethernet, consente di lavorare in qualunque località. I documenti possono essere redatti semplicemente introducendo le lettere da tastiera e controllando simultaneamente sia il formato sia il testo (in alto). I vari tipi di grafici, si ottengono partendo da un insieme di forme base (al centro). L'andamento della produttività è facilmente controllabile raccogliendo dati da fonti diverse in modo rapido e preciso (in basso).

Analizzeremo qui di seguito e in successivi articoli ciascuno dei punti indicati.

La comunicazione

Si tratta della funzione che un impiegato/manager svolge con maggior frequenza e che assorbe la gran parte del tempo di lavoro. Negli uffici, le comunicazioni rappresentano la linfa vitale, e la presenza di linee scorrevoli di comunicazione può essere determinante ai fini del successo o della stabilità di un'azienda. Il lato tecnologico della comunicazione è stato finora appannaggio di telefono, macchine per la scrittura, la riproduzione e la distribuzione di lettere di ogni forma (circolari, documenti interni...), avvisi, promemoria, telex ecc.
Lo sviluppo negli ultimi anni di reti locali e geografiche di connessione fra elaboratori ha consolidato alcune tecniche di comunicazione elettronica, in cui le linee telefoniche giocano ancora un ruolo importante, e condotto alla creazione del termine telematica per identificare la relativa disciplina: i migliori risultati, tra gli altri, si sono ottenuti nella creazione di sistemi che gestissero le comunicazioni fra i nodi di una rete di elaboratori in modo da realizzare la funzionalità della posta elettronica.

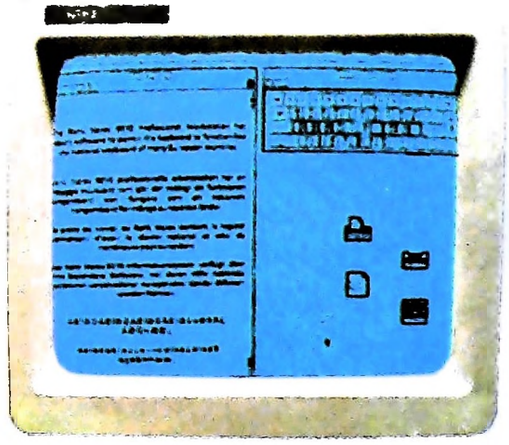
Reti di elaboratori e posta elettronica

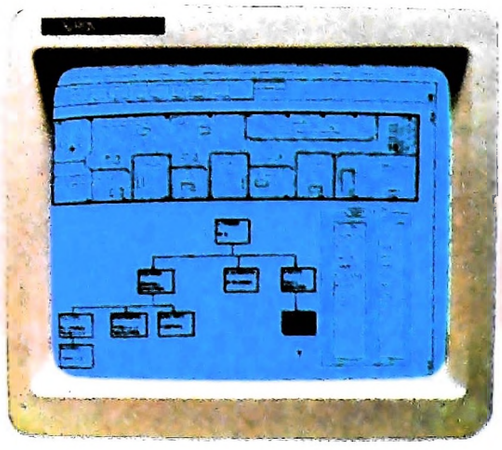
Il successo della telematica è stato immediato e notevole: negli Stati Uniti e in Francia le esperienze a riguardo sono ormai molteplici. Si tratta di paesi, in particolare gli USA, dove il numero di "personal" installati nelle abitazioni è altissimo e in continua ascesa è anche il numero di servizi telematici offerti ai cittadini dallo stato o da aziende private. Una delle esperienze più consolidate per ciò che riguarda la posta elettronica si ha negli Stati Uniti con la rete ARPA. Essa in realtà interessa anche paesi europei. In origine non era stata pensata come rete di comunicazione personale tra gli utenti dei calcolatori connessi alla rete stessa, ma l'uso in tal senso si è affermato in modo massiccio nell'ambiente degli informatici americani, che ne usufruiscono come e più del servizio postale, a parte ovviamente l'invio di pacchi.
All'interno della rete di elaboratori interconnessi, la posta elettronica viene trattata come un generico trasferimento di file (archivi elettronici), con alcuni requisiti in più, per ottenere che, una volta raggiunto il calcolatore del destinatario, il messaggio venga "parcheggiato" e al destinatario umano ne sia notificato l'arrivo; se egli è in quello stesso istante collegato, l'evento gli verrà indicato, altrimenti ciò si verificherà al primo successivo collegamento.
Si tratta quindi di una fedele riproduzione concettuale dell'attività di inviare una comunicazione da un terminale a un altro, dove per terminale si può intendere un qualsiasi oggetto o persona in grado di ricevere e/o inviare una comunicazione. I vantaggi in realtà sono molti, e in particolare in un ambiente di lavoro. In primo luogo, trattandosi di un servizio che si appoggia su elaboratori connessi in rete, per la

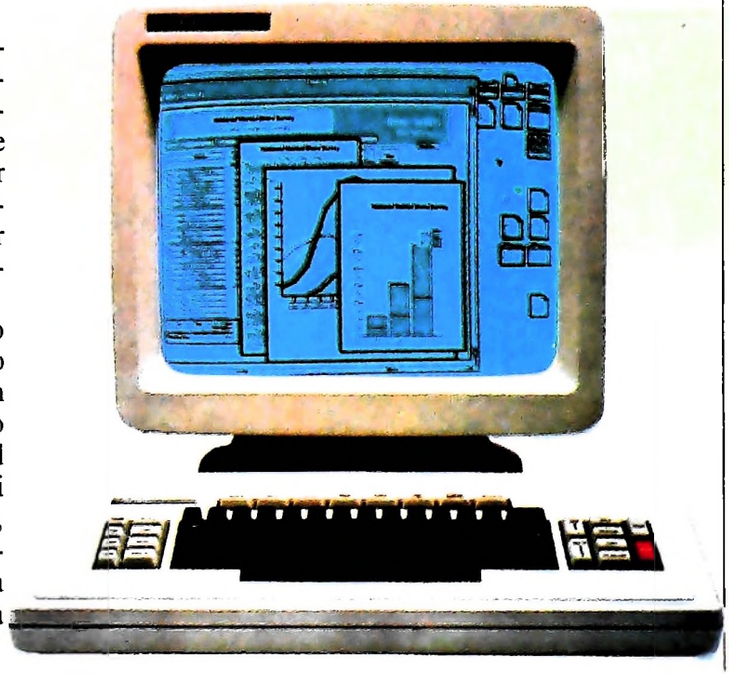
composizione del documento che s'intende creare e poi eventualmente inviare a terminali opportuni, è possibile utilizzare i programmi di composizione di testi che il proprio elaboratore mette a disposizione. In secondo luogo, il mittente può decidere di "duplicare" il messaggio che ha composto semplicemente indicando al sistema un insieme a sua scelta di destinatari: esso penserà all'inoltro, eliminando quindi ogni fotocopia, ribattitura, copie e buste dalla scrivania. L'inoltro sarà quasi istantaneo.

Quando il telefono pesa

Il fattore più importante rispetto a un ambiente di lavoro, in cui il telefono è fortemente in uso, è che la comunicazione tramite messaggio elettronico è asincrona, ovvero non dipende dal tempo.
Infatti, non è necessario che le due o più persone coinvolte nella comunicazione siano immediatamente disponibili a comunicare contemporaneamente.
Il mittente manda il messaggio quando gli è comodo, ben sapendo che il destinatario lo leggerà appena gli sarà possibile farlo, anche se in quel momento è occupato o assente. Questo evita ogni difficoltà o nervosismo da mancata conversazione, telefono occupato, difficoltà nel prendere la linea, ed è utilissimo sul lavoro, dove la maggior parte delle comunicazioni e dei messaggi ha carattere tecnico o di servizio, e non è necessaria un'interazione in prima persona. Per il destinatario naturalmente il fattore dell'asincronicità è di grande comodità, in quanto una telefonata interrompe quasi completamente l'attività che si sta svolgendo, richiedendo in modo imposto l'attenzione del ricevente; il telefono assume così la veste di strumento per la comunicazione interpersonale, dove il fattore umano ha peso, mentre assurge a regola di buona conversazione elettronica (in realtà sempre valida) il fatto di inviare una forma di "ricevuto" al mittente della missiva, che potrebbe restare nel dubbio.
I messaggi elettronici possono essere letti e a essi si può rispondere quando si ritiene utile farlo, senza interferenze. Essi vengono immagazzinati nell'ordine di arrivo e possono essere elaborati con le funzionalità che il calcolatore ospite mette a disposizione. Possono essere ordinati per data o per mittente o con altro criterio, poi conservati, cancellati, e infine è possibile riutilizzarli nella risposta, usando il text-editor del proprio sistema, per ottenere citazioni o per chiedere precisazioni.
L'esperienza di utenti della rete ARPA ha mostrato che l'uso del telefono si riduceva di molto, e veniva usato soprattutto per raggiungere persone non connesse alla rete stessa. In un ambiente d'ufficio, l'utilità di tale strumento, oltre a quello di essere complementare alla posta ordinaria e di sostituire il telegrafo, risulta netta per ciò che riguarda le comunicazioni all'interno dell'azienda. Avvisi di riunioni, seminari, note, eventi avvenuti, qualsiasi informazione ritenuta utile può essere inviata a una persona del proprio o di altri uffici, così da essere certi che non verrà persa tra altre carte, o ignorata senza precisa volontà.



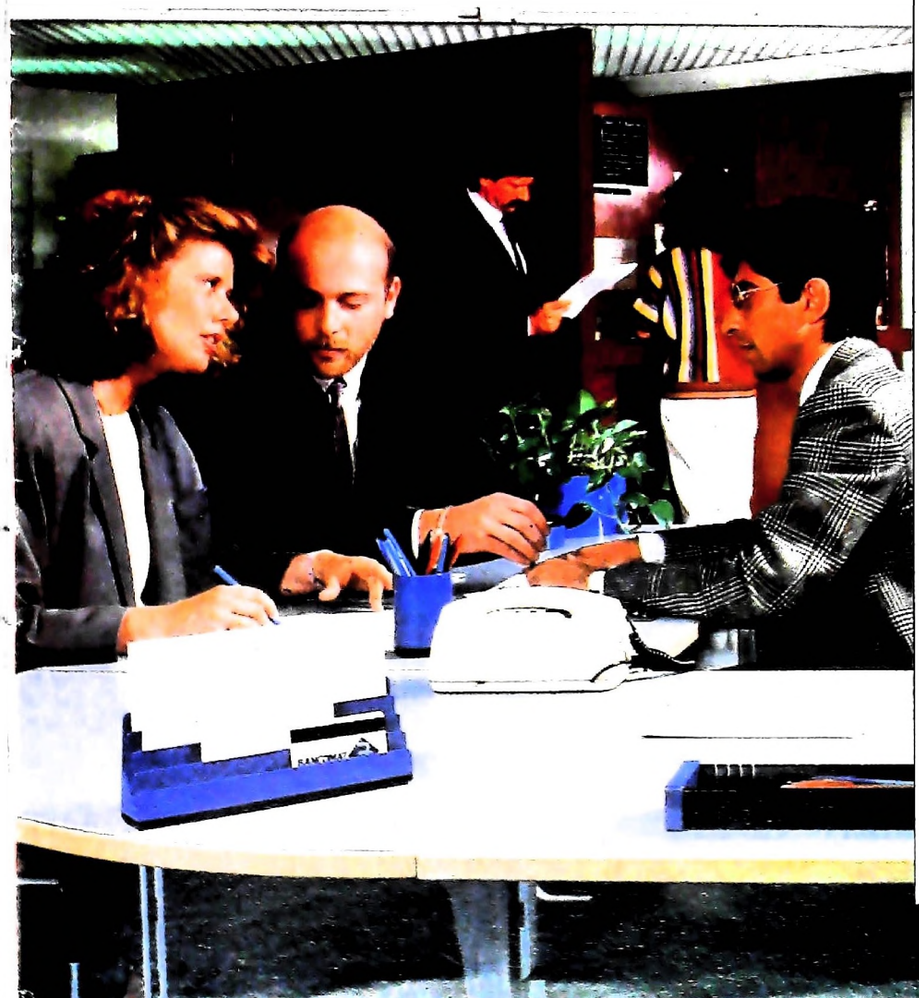




Sommario del Terzo Volume

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.

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 comunicare via telefono per spedire e ricevere informazioni. In grado di funzionare a batteria oppure collegato all'impianto elettrico, M10 mette ovunque a disposizione la sua potenza di memoria, il suo display orientabile a cristalli liquidi capace anche di elaborazioni grafiche, la sua tastiera professionale arricchita da 16 tasti funzione.

Ma M10 può utilizzare piccole periferiche portatili che ne ampliano ancora le capacità, come il microplotter per scrivere e disegnare a 4 colori, o il registratore a cassette per registrare dati e testi, o il lettore di codici a barre. E in ufficio può essere collegato con macchine per scrivere elettroniche, con computer, con stampanti. Qualunque professione sia la vostra, M10 è in grado, dovunque vi troviate, di offrirvi delle capacità di soluzione che sono davvero molto grandi. M10: il più piccolo di una grande famiglia di personal.

PERSONAL COMPUTER OLIVETTI M10

L'UFFICIO DA VIAGGIO

Anche in leasing con Olivetti Leasing.

olivetti

Per informazioni inviate a mezzo contrassegno da Olivetti M10 Punto di Vendita" a: Olivetti, Divisione Personal Computer, Via Meravigli 12 20123 Milano

NOME/COGNOME

VIA N.

CAP/CITTÀ

TELEFONO