CADEL

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

34 CORSO PRATICO COL COMPUTER

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

F4 F5 F6 F7 F8 BATTERY LOW

diretto da GIANNI DEGLI ANTONI

FABBRI EDITORI

Direttore dell'opera
GIANNI DEGLI ANTONI

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

UMBERTO ECO
Ordinario di Semiotica presso l'Università di Bologna

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

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

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

Curatori di rubriche
ADRIANO DE LUCA (Professore di Architettura 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,
Etnoteam (ADRIANA BICEGO)

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

Direttore Editoriale
ORSOLA FENGHI

Redazione
CARLA VERGANI
LOGICAL STUDIO COMMUNICATION

Art Director
CESARE BARONI

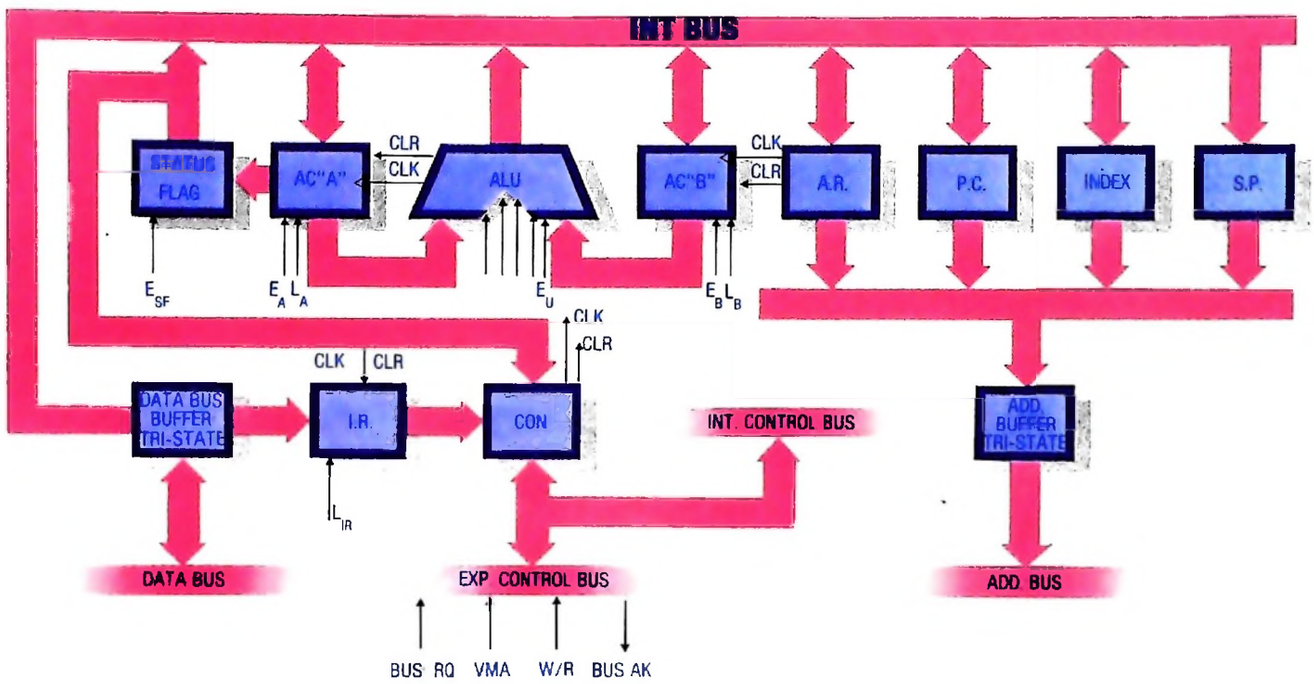
Impaginazione
BRUNO DE CHECCHI
PAOLA ROZZA

Programmazione Editoriale
ROSANNA ZERBARINI
GIOVANNA BREGGÈ

Segretarie di Redazione
RENATA FRIGOLI
LUCIA MONTANARI

UAMICRO III

È l'ultima versione della serie UAMICRO. Ha un'architettura simile a quella dei microprocessori a 8 bit, con però alcune varianti.



Schema a blocchi della UAMICRO III con i registri interni, i buffer dei collegamenti con i bus esterni e il bus interno (INT-BUS).

Vediamo com'è fatto internamente l'UAMICRO III (figura in alto): ci sono tre bus, uno di indirizzi di 16 bit per permettere di gestire sessantaquattromila porzioni di memoria, comprese le porzioni dedicate a porti di entrata e uscita. Il secondo è un bus di dati, di 8 bit, che permette la definizione di 256 istruzioni diverse. È sempre possibile inoltre collegare più parole di 8 bit per avere un numero più grande: infatti la presenza di altri elementi interni come il registro A.R. di 16 bit, il Carry (riporto) ecc. consentono questa possibilità. Per ultimo si ha il bus di controllo che, nel nostro caso, prevede solo le linee:

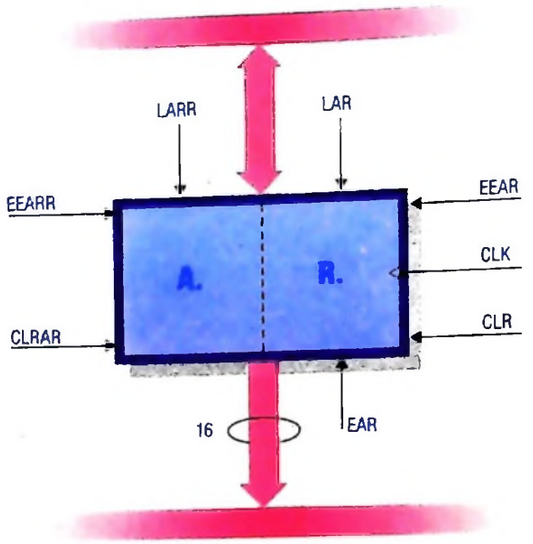
$\overline{VMA}$ = Uscita attiva-negativa che convalida le linee di indirizzo.

$\overline{W}/R$ = Uscita attiva-negativa per la scrittura, attiva-positiva per la lettura.

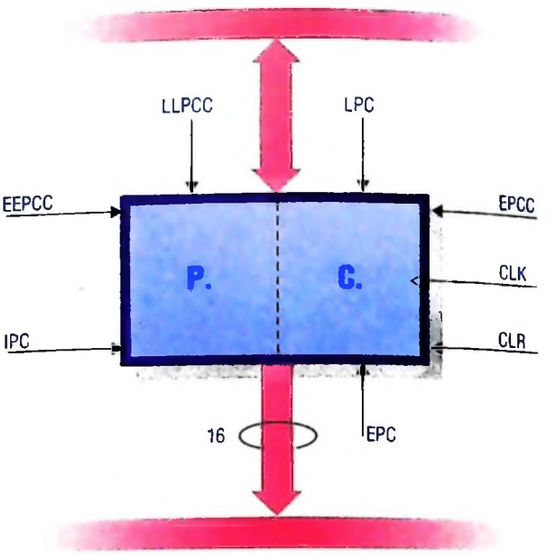
$\overline{BUSRQ}$ = Entrata attiva-negativa per la richiesta di controllo dei bus per operazioni in DMA o multiprocessi.

$\overline{BUSAK}$ = Uscita attiva-negativa che conferma, da parte del microprocessore, il suo stato di isolamento dai bus di dati e di indirizzi attraverso i suoi buffer di tre stati.

È importante notare il protocollo di comunicazione generato da questi due ultimi comandi. Questa prassi è molto comune e funziona in questa forma: il primo comando $\overline{BUSRQ}$ è un comando di richiesta di controllo dei bus da parte di un altro microprocessore o un elemento come il DMA (sistema capace di indirizzare direttamente alla memoria) collegato agli stessi bus, mentre il comando $\overline{BUSAK}$ è la risposta del microprocessore o dell'elemento che in quel momento sta usando i bus. Questo prima di tutto completa l'esecuzione dell'istruzione in corso e poi mette in terzo stato i buffer dei bus,



Il registro A.R. (Registro Ausiliario) con i suoi comandi di controllo.



Il registro P.C. (Contatore di Programma) con i suoi comandi di controllo.

compreso $\overline{VMA}$ e $\overline{W}$/R, per cui l'elemento richiedente non entra in funzione fin quando non riceve la risposta $\overline{BUSAK}$.
Per il momento abbiamo tralasciato altri comandi di controllo per non appesantire eccessivamente la comprensione delle funzioni principali: di essi ci occuperemo quando tratteremo i microprocessori commerciali.
È importante notare anche la quantità di bus esistenti. Abbiamo visto nelle versioni precedenti delle UAMICRO come l'uso di un singolo bus di dati e indirizzi rendeva più semplice l'architettura anche se poi veniva appesantita esternamente per la necessità di logica di controllo aggiuntiva. In realtà questa configurazione è quella che ha trovato più riscontro nelle architetture dei microprocessori a 8 bit.
La disponibilità di diversi bus, sia interni che esterni, risponde ad alcune necessità tecniche: prima fra tutte la velocità di esecuzione, assicurata dall'architettura parallela derivante dalla presenza di bus diversi per dati e indirizzi, e secondariamente la necessità di mettere in terzo stato il microprocessore verso l'esterno, che obbliga alla realizzazione di un bus interno indipendente.
Incominciamo ora l'analisi dei registri interni, alcuni già conosciuti, come i due accumulatori A e B, al primo dei quali è associato un nuovo registro STATUS FLAG (Registro di stati) che serve a rivelare le condizioni per i salti condizionati.
I quattro registri A.R. (registro ausiliario), P.C. (Contatore di programma), INDEX (Registro indice) e S.P. (Puntatore di STACK) sono registri di 16 bit. Il registro A.R. è composto da due parole di 8 bit con i comandi che funzionano nel seguente modo (in alto a sinistra):

LARR = Carica gli 8 bit di sinistra di A.R. con il contenuto dell'INT-BUS.
LAR = Carica gli 8 bit di destra di A.R. con il contenuto dell'INT-BUS.
EAR = Abilita il registro A.R. nel BUS di indirizzi.
CLRAR = Azzera gli 8 bit di sinistra di A.R.
EEARR = Abilita gli 8 bit di destra di A.R. nell'INT-BUS.
EEAR = Abilita gli 8 bit di destra di A.R. nell'INT-BUS.

P.C. (Contatore di programma)

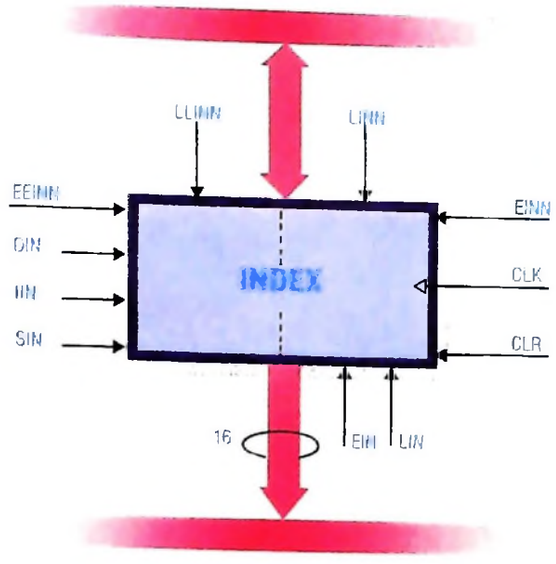
Il contatore di programma P.C. (in alto a destra) ha i seguenti comandi:

IPC = Incrementa il contenuto del P.C.
LPC = Carica il contenuto dell'INT-BUS negli 8 bit di destra del P.C.
EPC = Abilita il P.C. nel bus di indirizzi.
LLPCC = Carica il contenuto dell'INT-BUS negli 8 bit di sinistra del P.C.
EPCC = Abilita gli 8 bit di destra del P.C. nell'INT-BUS.
EEPCC = Abilita gli 8 bit di sinistra del P.C. nell'INT-BUS.

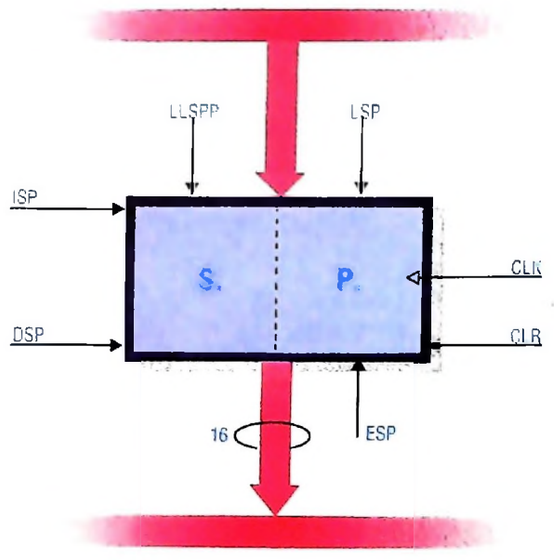
Index (Registro indice)

Il registro INDEX (in alto a sinistra) ha i seguenti comandi:

SIN = Somma al contenuto di INDEX il contenuto dell'INT-BUS.
EINN = Abilita il contenuto degli 8 bit di destra di INDEX nell'INT-BUS.



Il registro Index (Registro Indice) con i suoi comandi di controllo.



Il registro S.P. (Puntatore di Stack) con i suoi comandi di controllo.

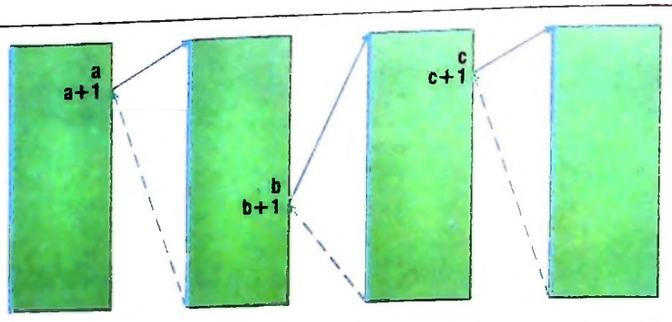
EEINN = Abilita il contenuto degli 8 bit di sinistra di INDEX nell'INT-BUS.
IIN = Incrementa il contenuto di INDEX.
DIN = Decrementa il contenuto di INDEX.
EIN = Carica il contenuto dell'INT-BUS in INDEX.
LIN = Carica il contenuto dell'INT-BUS in INDEX.
LINN = Carica il contenuto dell'INT-BUS negli 8 bit di destra di INDEX.
LLINN = Carica il contenuto dell'INT-BUS negli 8 bit di sinistra di INDEX.

S.P. (Puntatore di STACK)

Il registro S.P. (in alto a destra) ha i seguenti comandi:

ISP = Incrementa il contenuto di S.P.
DSP = Decrementa il contenuto di S.P.
ESP = Abilita S.P. nel BUS di indirizzi.
LSP = Carica il contenuto dell'INT-BUS negli 8 bit di destra del S.P.
LLSPP = Carica il contenuto dell'INT-BUS negli 8 bit di sinistra del S.P.

Dei primi tre registri A.R., P.C. e INDEX già conosciamo le funzioni anche se, in questa versione la loro architettura si è arricchita di più elementi che li rendono più efficienti.
Vediamo ora di approfondire la funzione del registro S.P. di 16 bit. Quando nella Uamicro II abbiamo trattato il problema della gestione dei sottoprogrammi abbiamo detto che esisteva un'altra forma, chiamata dello STACK, usata da quasi tutti i microprocessori. Vediamo in cosa consiste e come risolve il nostro problema.
Prima di tutto facciamo un breve riepilogo della problematica creata con l'introduzione dell'uso dei sottoprogrammi. Quando c'è il passaggio dal programma principale a un sottoprogramma bisogna salvare i contenuti dei registri interni. Nella UAMICRO II questo problema veniva risolto per hardware creando più registri uguali, guadagnando in velocità di esecuzione a fronte di un aumento di costo. Con la soluzione a STACK, che tratteremo adesso, si perde velocità però si guadagna in costo ed estensibilità. La parola STACK vuol dire pila e viene usata per dare l'idea di come funziona tutto il meccanismo. Proviamo, infatti, a immaginare quello che succede in cucina con una pila di piatti: man mano che vengono lavati e asciugati vengono accatastati l'uno sull'altro, e al momento di usarli l'ultimo che è stato posto in cima alla pila è il primo ad essere prelevato.
I problemi inerenti all'uso dei sottoprogrammi sono due: il primo è quello che abbiamo già visto e cioè la salvaguardia dei registri e il loro ripristino, il secondo è la possibilità di annidamento. Su quest'ultimo punto si sono già dati dei cenni che ora riprendiamo.
Nella figura in alto con a rappresentiamo il programma principale: alla chiamata al sottoprogramma nel punto "a", viene salvato nello STACK l'indirizzo "a+1" più i contenuti dei registri, cioè il contenuto degli accumulatori A, B, lo STATUS FLAG ed eventualmente altri.
Lo STACK è una zona di memoria, che occupa in genere le posizioni di indirizzo più alto (in basso a sinistra), il cui indirizzo iniziale, chiamato TOP, si trova nel registro S.P.
Quando c'è una chiamata al sottoprogramma nel Top dello STACK si deposita una parte dell'indirizzo di ritorno (8 bit), s'incrementa il registro S.P. e si deposita la seconda parte

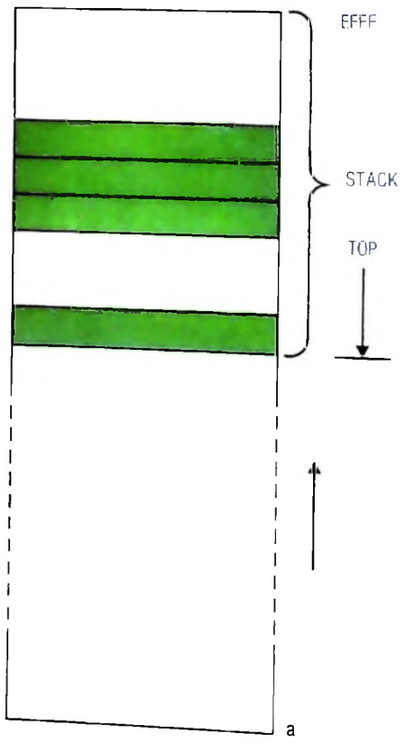


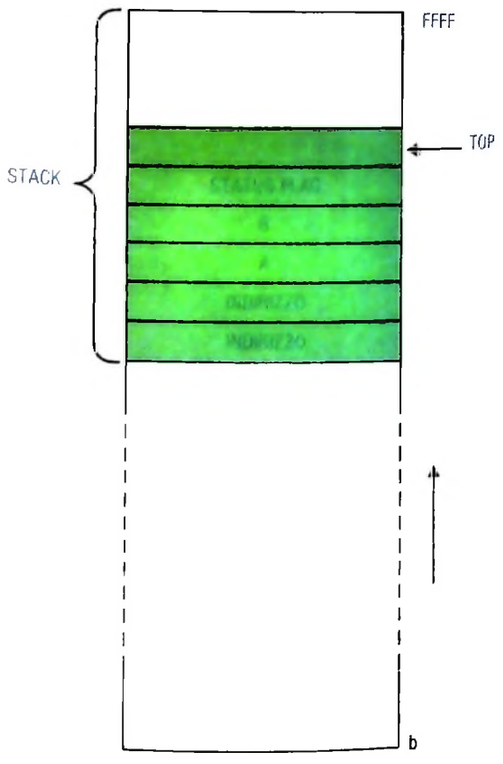
Esempio di passaggio dal programma principale ai sottoprogrammi e ritorno al programma principale.

dell'indirizzo; poi si incrementa ancora il registro S.P. e si deposita il contenuto di A, poi quello di B e infine lo STATUS FLAG.
A questo punto il registro S.P. viene incrementato ancora una volta e il nuovo indirizzo indica così il nuovo TOP dello STACK (in basso a destra).
Ritorniamo ora alla figura in alto e continuiamo l'analisi: quando nel sottoprogramma b si presenta la necessità di andare al sottoprogramma c, nello STACK si ripete l'operazione precedente e così fino a d. Quando ci troviamo in quest'ultimo caso, se non ci sono altre chiamate a sottoprogramma, il processo viene invertito ed è così che viene fuori l'analogia con la pila dei piatti.

C'è anche un termine tecnico per definire questo comportamento: LIFO (Last In First Out), cioè "l'ultimo entrato è il primo che esce".
Nel nostro caso il ritorno dal sottoprogramma funziona nel seguente modo: prima si decrementa il registro S.P. che punta così a una casella più profonda del TOP, (che era vuoto), di lì si estrae il contenuto e lo si deposita nel registro corrispondente; poi si ripete l'operazione con il contenuto della posizione che segue e, infine, si preleva l'indirizzo di ritorno completo, che viene depositato nel P.C. A questo punto quella posizione è il nuovo TOP dello STACK. Le operazioni vanno ripetute per ogni chiamata a sottoprogramma e per ogni ritorno da sottoprogramma.

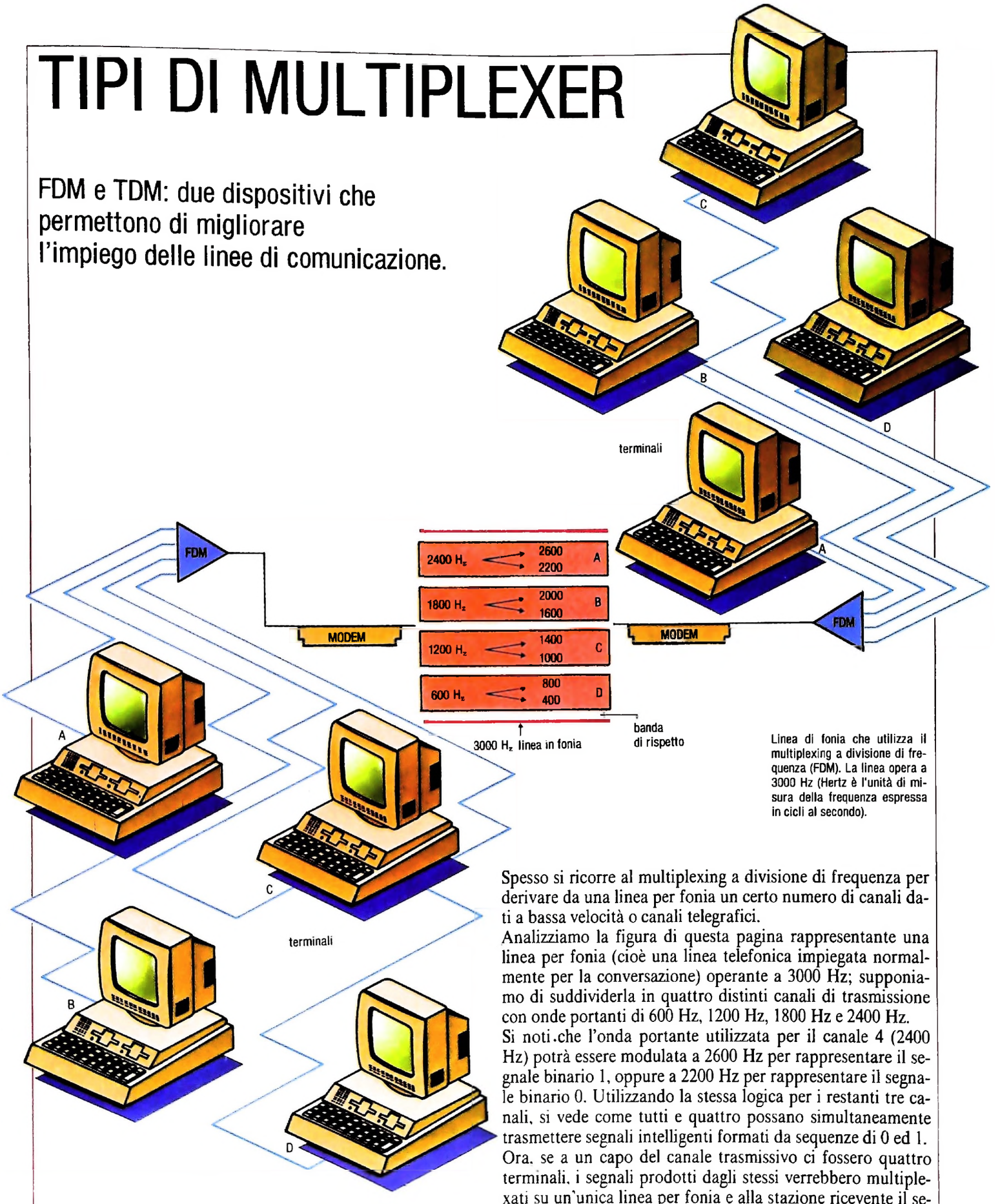
Creazione della zona riservata allo STACK nella memoria e l'ubicazione del TOP (a). Configurazione dello STACK e sistema con cui vengono depositati i contenuti dei registri nelle sue locazioni (b).





TIPI DI MULTIPLEXER

FDM e TDM: due dispositivi che permettono di migliorare l'impiego delle linee di comunicazione.



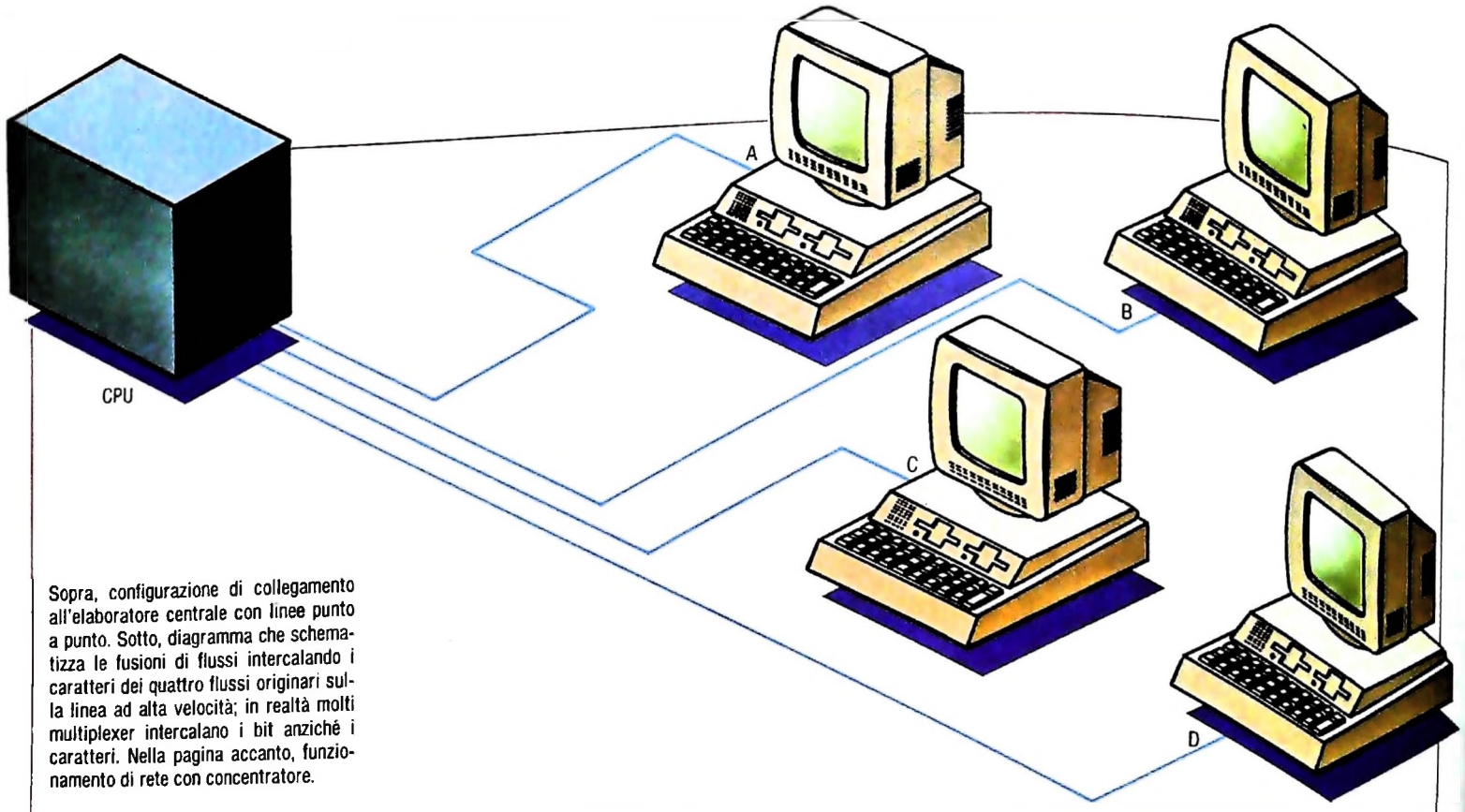
Linea di fonia che utilizza il multiplexing a divisione di frequenza (FDM). La linea opera a 3000 Hz (Hertz è l'unità di misura della frequenza espressa in cicli al secondo).

Spesso si ricorre al multiplexing a divisione di frequenza per derivare da una linea per fonia un certo numero di canali dati a bassa velocità o canali telegrafici.

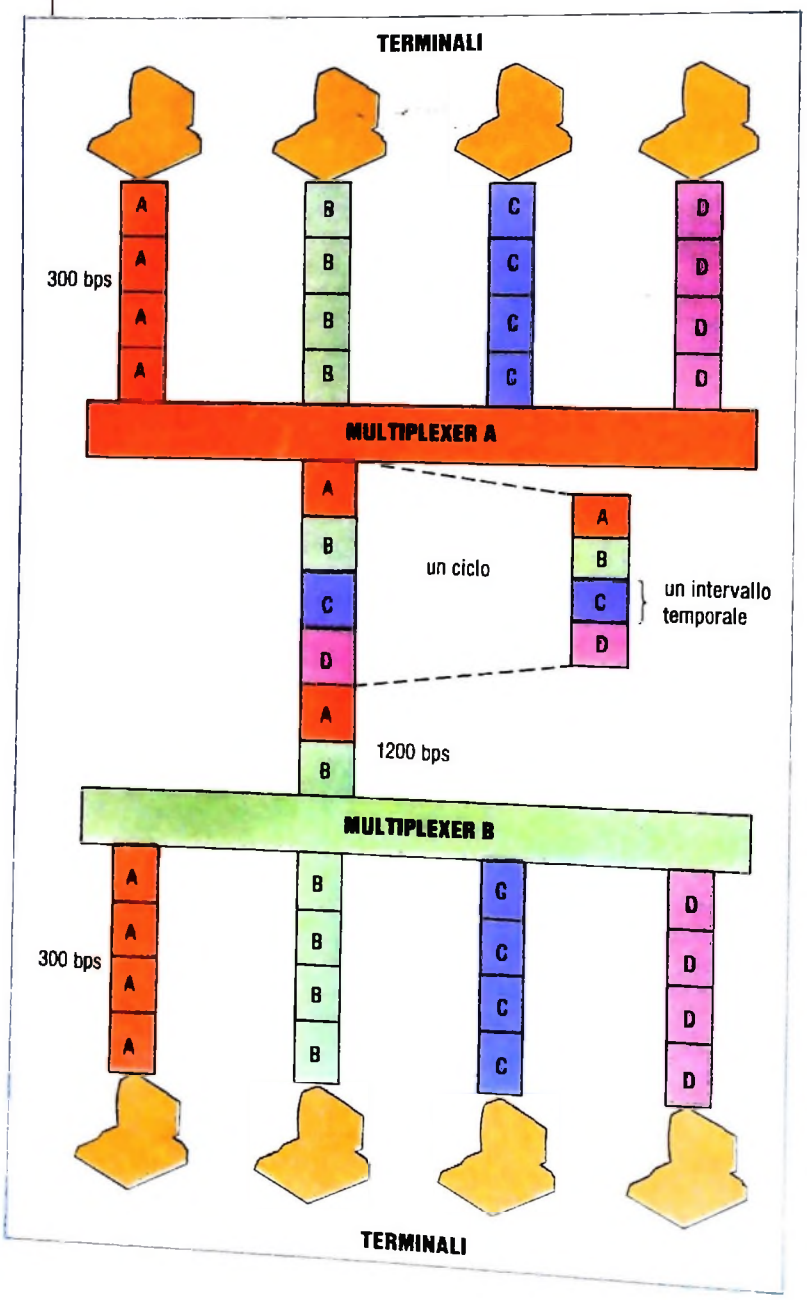
Analizziamo la figura di questa pagina rappresentante una linea per fonia (cioè una linea telefonica impiegata normalmente per la conversazione) operante a 3000 Hz; supponiamo di suddividerla in quattro distinti canali di trasmissione con onde portanti di 600 Hz, 1200 Hz, 1800 Hz e 2400 Hz.

Si noti che l'onda portante utilizzata per il canale 4 (2400 Hz) potrà essere modulata a 2600 Hz per rappresentare il segnale binario 1, oppure a 2200 Hz per rappresentare il segnale binario 0. Utilizzando la stessa logica per i restanti tre canali, si vede come tutti e quattro possano simultaneamente trasmettere segnali intelligenti formati da sequenze di 0 ed 1.

Ora, se a un capo del canale trasmissivo ci fossero quattro terminali, i segnali prodotti dagli stessi verrebbero multiplexati su un'unica linea per fonia e alla stazione ricevente il se-



Sopra, configurazione di collegamento all'elaboratore centrale con linee punto a punto. Sotto, diagramma che schematizza le fusioni di flussi intercalando i caratteri dei quattro flussi originari sulla linea ad alta velocità; in realtà molti multiplexer intercalano i bit anziché i caratteri. Nella pagina accanto, funzionamento di rete con concentratore.



gnale risultante verrebbe demultiplexato e i relativi segnali indirizzati ai rispettivi terminali riceventi. In conclusione la tecnica FDM (Multiplexer a Divisione di Frequenza) frazione la banda di frequenza disponibile sul canale telefonico in tanti intervalli di frequenza (compresi nella banda di partenza) ciascuno dei quali assegnato a un singolo terminale che accede al multiplexer.

Con questo tipo di multiplexing, l'efficienza della trasmissione viene ridotta in quanto si richiedono delle "bande di rispetto" tra gli intervalli di frequenza assegnati per evitare interferenze tra segnali adiacenti. Tali bande di rispetto utilizzano ovviamente una parte della capacità della linea.

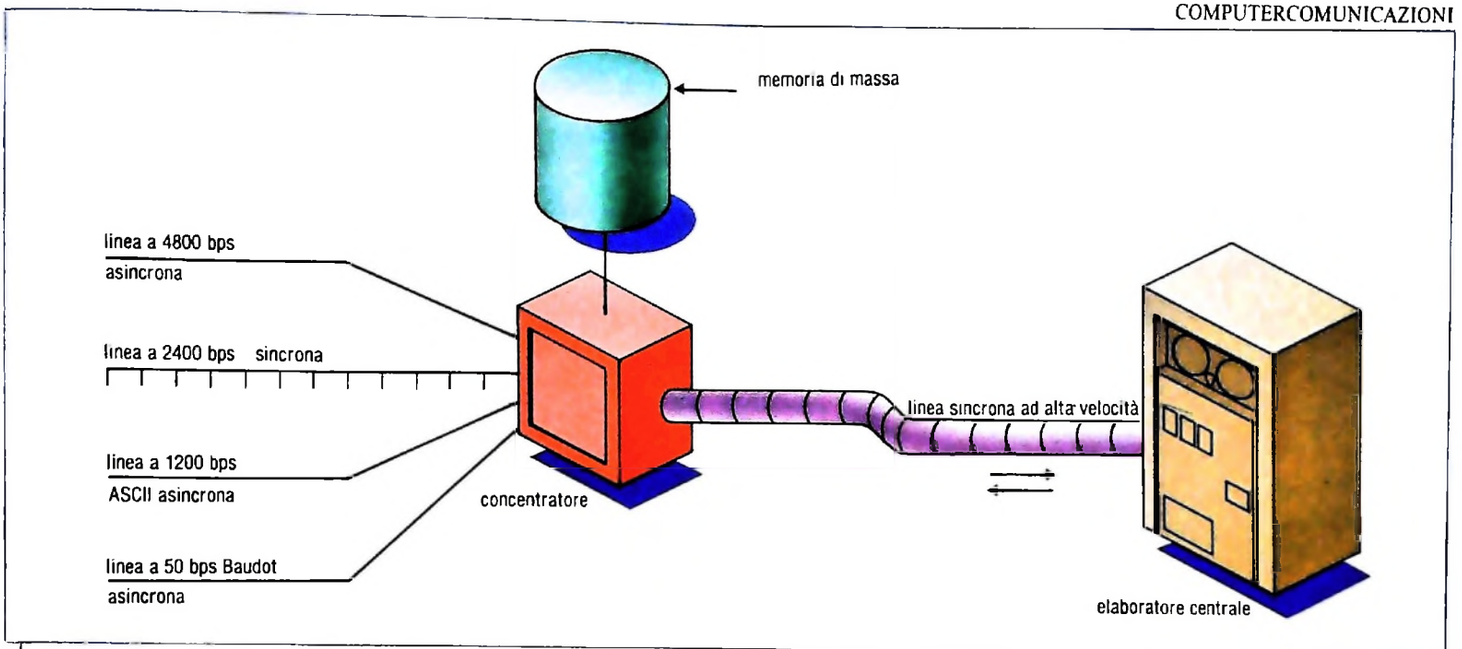
Per descrivere i principi di funzionamento del "multiplexing" a divisione di tempo utilizziamo un esempio.

Si supponga che quattro ricercatori dislocati in quattro città diverse desiderino collegarsi con l'elaboratore centrale del proprio istituto. Se si vuole installare un terminale a 300 bit al secondo in ognuna delle quattro abitazioni, lo si potrebbe collegare all'elaboratore centrale con linee punto a punto come indicato nella figura in alto. Questo consentirebbe a ogni terminale di accedere all'elaboratore senza limitazioni di tempo, ma le linee impiegate risulterebbero costose.

Con il "multiplexer" si possono fondere i quattro flussi di dati a bassa velocità in un unico flusso che può essere trasmesso su un canale ad alta velocità. Per trasmettere dati provenienti da terminali diversi su un'unica linea occorre fondere, intercalare e collegare tra di loro i dati.

La figura a lato riporta i flussi di dati a bassa velocità in arrivo da ogni terminale verso il multiplexer B, che rileva i caratteri da ciascuna linea, li intercala e li trasmette, lungo il canale ad alta velocità, al multiplexer A che a sua volta demultipla i flussi di caratteri intercalati e ricostruisce i flussi originali di dati a bassa velocità.

Con questo genere di configurazioni, si può vedere che i multiplexer sono trasparenti per la rete e che ogni terminale "vede" un collegamento punto a punto con l'elaboratore.

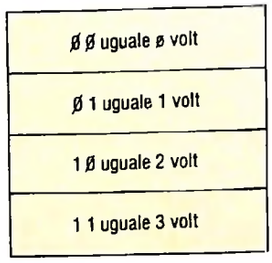


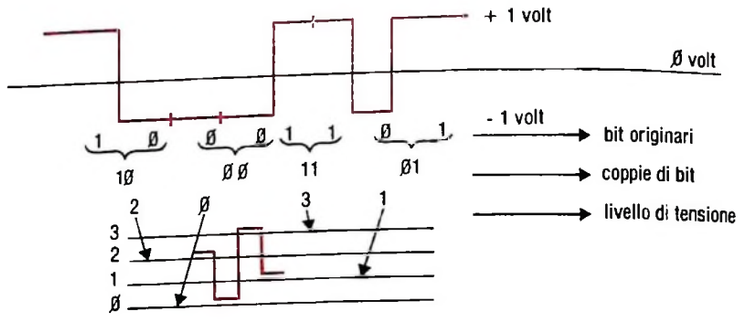
Concentratori

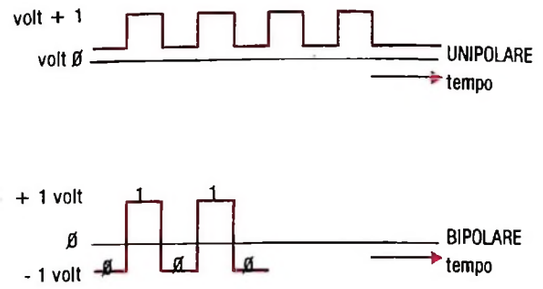
Un *concentratore* o *elaboratore delle comunicazioni* è un dispositivo basato su un elaboratore che spesso possiede anche una memoria di massa. Il concentratore è simile al multiplexer in quanto combina i dati provenienti da più terminali su una linea ad alta velocità per la trasmissione a un elaboratore centrale, ma ne differisce in quanto è in grado di modificare la forma dei flussi di dati prima di fonderli su una linea ad alta velocità (figura in alto).
Il concentratore stesso è di solito collegato da un lato all'elaboratore centrale per mezzo di una linea sincrona ad alta velocità e dall'altro direttamente con i terminali. Gestendo la rete di comunicazioni, esso rileva una parte del carico di lavoro che altrimenti graverebbe sull'elaboratore centrale.
Per gestire le linee di trasmissione dati è necessario disporre non tanto di istruzioni di grande potenza quanto piuttosto di un veloce ciclo interno, e può accadere così che minicalcolatori di basso costo possano gestire il traffico dei dati quanto un elaboratore costoso.
Il concentratore, se è dotato di memorie di massa e di memoria centrale sufficiente, può quindi funzionare come dispositivo per memorizzare e inviare messaggi: assemblerà i messaggi completi o i blocchi di messaggi provenienti dai vari terminali della rete, li archivierà nella sua memoria e li invierà all'elaboratore centrale nel momento voluto.
Eseguirà conversioni di codice, conversioni di velocità e conversioni di formato. Dotato di "intelligenza", userà meglio le linee ad alta velocità comprimendo i dati prima della trasmissione, rimuovendo le informazioni ridondanti o convertendo le informazioni digitali in binario puro piuttosto che tradurle in caratteri ASCII e così via.
Il concentratore può anche migliorare l'utilizzazione della linea mediando statisticamente il traffico della rete sulla linea ad alta velocità. Poiché il concentratore è dotato di capacità di calcolo e di memoria, esso può accordare il traffico sulla linea di trasmissione in modo da evitare che il tempo della linea ad alta velocità e il tempo del terminale siano necessariamente corrispondenti. Da ciò deriva che è possibile collegare e far lavorare più terminali sulla stessa linea ad alta velocità perché in tal modo è sufficiente che la linea stessa sia in grado di sostenere un traffico appena un po' più elevato.
Il concentratore può inoltre aumentare l'affidabilità della rete anche se questo dipende dall'applicazione. In alcuni casi serve come centro raccolta dei dati della rete, dati che verranno poi trasmessi sotto forma di blocchi interi di messaggi all'elaboratore centrale o *host computer*.
Se per una qualsiasi ragione l'elaboratore centrale o la rete non dovessero funzionare, il concentratore può continuare a raccogliere e assemblare i dati senza che l'incidente si ripercuota sulla rete periferica. Quando poi l'elaboratore centrale o la linea principale ritorneranno attivi, invierà i dati all'elaboratore centrale nel modo consueto.
Un caso del tutto particolare è il *calcolatore di gestione delle linee* o *front-end processor* che segue lo stesso principio generale di funzionamento del concentratore, ma è installato nello stesso luogo in cui si trova l'elaboratore principale. Un *front-end processor* gestisce la rete di comunicazioni per conto dell'elaboratore centrale e gli trasmette messaggi completi. Il collegamento fisico tra i due elaboratori può essere costituito da una linea dati seriale ad alta velocità o da un'interfaccia parallela elaboratore-elaboratore.

Modem e interfacce

Concetti di base dei Circuiti di Trasmissione
Ci sono quattro termini molto importanti nelle trasmissioni dati e cioè *baud*, *dibit*, *unipolare*, *bipolare*.
Il *baud* è l'unità di velocità del segnale telegrafico ed è ottenuto dal reciproco della lunghezza (in secondi) della più breve pulsazione usata per creare un carattere.







Come si ottengono quattro stati con 0 o 1 presi a due a due (in alto a sinistra). Dibit rappresentati da differenti voltaggi (in alto a destra). Schema di funzionamento dei segnali unipolare e bipolare (qui sopra).

La lunghezza di un impulso per il codice Baudot con una telescrivente a 60 parole al minuto è di 0.022 secondi; perciò si ottiene 1/0.022 = 45.45 baud.
Un semplice metodo per incrementare il rapporto di bit trasmessi lungo una linea è quello di combinare coppie di bit adiacenti in *dibit* (*di* = due). L'obiettivo è quello di spedire ciascun dibit come un elemento di segnale separato. Dato che le leggi della teoria dell'informazione non possono essere annullate, il processo di trasmissione simultanea di due bit implica quattro (2^2) differenti stati di segnale. Un modo per ottenere i quattro stati è rappresentato nella figura in alto a sinistra. La figura in alto a destra mostra come, variando la tensione tra 0 e 3 volt, un dispositivo trasmittente possa inviare due bit simultaneamente.
Un carattere ASCII trasmesso con modalità seriale asincrona richiederà 10 o 11 bit per essere trasmesso; se il rapporto di trasmissione è di 1200 bit per secondo (bps), ciascuno di questi bit richiederà 833 μ secondi e quindi saranno necessari 8.330 μ secondi per carattere (con carattere a 10 bit).
Utilizzando una configurazione a dibit allo stesso rapporto di prima, lo stesso carattere richiederà un tempo pari a 4165 μ secondi (la metà del precedente), in quanto ogni 833 μ secondi verranno trasmessi due bit invece di uno.
I tipi di segnale riconosciuti da una linea sono:
unipolare che descrive la condizione della linea dove il segnale è commutato dalla tensione più bassa (0 volt) a quella più alta (1 volt) ad intervalli fissati o arbitrari; l'accensione e lo spegnimento di una lampadina è un processo unipolare;
bipolare che descrive la condizione della linea nella quale il segnale è alternativamente invertito a causa dell'applicazione sia di potenziali positivi che di quelli negativi (figure a lato). Le opposte polarità sono contraddistinte dagli stati 0 e 1. La trasmissione bipolare è il metodo di base utilizzato al giorno d'oggi sulle linee di trasmissione.

Modulazione
I segnali intelleggibili nei circuiti di trasmissione subiscono molti cambiamenti nella forma elettrica durante la trasmissione a lunga distanza. I segnali dei dati e di telescrivente devono essere trasformati in segnali audio rappresentativi prima della trasmissione a causa della banda passante limitata dei circuiti telefonici. Una conversione finale riporta il segnale nella sua forma originale.
I segnali di trasmissione sono trasportati nei sistemi telefonici nello stesso modo in cui una lettera è portata dalla posta. La persona che imposta la lettera e la persona che la riceve corrispondono ai terminali di un sistema di trasmissione, e la lettura costituisce il messaggio da comunicare. Il sistema postale è il mezzo di comunicazione. Ma il sistema postale non è una cosa unica; esso è costituito da molti componenti, quali i diversi uffici postali attraverso cui la lettera passa.
I postini, gli autocarri, i treni e gli aerei che trasportano la lettera sono i portatori, necessari al trasferimento dell'informazione, ma non legati al testo della lettera.
Le *portanti elettroniche (carriers)* sono simili ai portatori postali. Le portanti non sono i segnali di informazione, ma vengono usate nel trasferimento delle forme d'onda dei segnali mediante la *modulazione* e la *demodulazione*.
La modulazione è un processo d'impressione di una forma d'onda sopra la portante per la trasmissione. Raramente viene trasmessa la forma d'onda di comunicazione originale: l'informazione è contenuta nella forma d'onda modulata.
La demodulazione è il procedimento di separazione dell'informazione della portante dopo la trasmissione. A questo punto la portante è normalmente eliminata o messa da parte in quanto non è più utilizzabile.
Una portante è una forma d'onda ad ampiezza, frequenza e fase costanti che può essere modulata cambiando l'ampiezza, la frequenza o la fase. Le forme d'onda delle portanti sono normalmente di frequenza molto più alta di quella del segnale da trasmettere.

Un po' di grafica (III)

Nelle due precedenti lezioni abbiamo imparato ad affrontare il problema di tracciare la curva dei bioritmi di una persona, e quindi a usare l'istruzione PSET per disegnare sullo schermo figure come rette, curve ecc.
In particolare, relativamente al problema dei bioritmi, avevamo inquadrato completamente lo schermo del nostro M10, con l'evidenziazione di un piccolo campione delle tre curve e con il tracciamento di una scala dei tempi che individuava i 30 giorni di un mese, con un'opportuna scala numerica.
A questo punto possiamo finalmente occuparci di tracciare le curve vere e proprie:

- facciamo variare una variabile K da 0 a 29 (cioè per 30 giorni)
- per ogni giorno facciamo variare i pixel da 1 a 7 con una variabile H (ricordiamo, infatti, che l'asse del mese era composto da 210 pixel, che individuavano 30 giorni di 7 pixel ciascuno)
- per ogni coppia H, K calcoliamo la posizione del pixel come J=K*7+H
- per ogni J così ottenuto, tracciamo una funzione sinusoidale del tipo y=sin(x), che vada "sopra" e "sotto" l'asse tracciato; per fare sì che sia centrata sull'asse, facciamo in modo che si sposti da 0 a 60, e, poiché una funzione come quella usata assume valori da 0 a 1, useremo un'espressione del tipo:

$$y=30-30*\sin(x)$$

  in modo che, quando sin(x) vale 1 allora y assume il valore 0, mentre quando sin(x) vale -1, y assume il valore 60.
- ciascuna di queste funzioni deve
- avere un periodo pari a 23, 28 o 33 giorni (cioè partire da zero in un certo istante, "alzarsi" fino ad un massimo, "abbassarsi" fino ad andare sotto l'asse, quindi ritornare a zero, il tutto in corrispondenza a 23, 28 o 33 giorni)
- essere "opportunamente" in fase, ovvero fare in modo che il loro "movimento" parta non dal giorno 1, ma sia da considerare già partito da un certo numero di giorni, come precedentemente calcolato; ciò, essendo la funzione sin periodica di 2 pi greco, si ottiene approssimativamente con:

$$\sin(6.28*(J+F*7)/(7*periodo)$$

essendo F il numero di giorni da cui la fase è partita. Quindi:

```
150 FOR K=0 TO 29
160 FOR H=1 TO 7
165 LET J=(K*7+H)
166 LET Y=30-30*SIN(6.28*(J+F*7)/(7*23))
167 LET Z=30-30*SIN(6.28*(J+S*7)/(7*28))
168 LET W=30-30*SIN(6.28*(J+I*7)/(7*33))
200 NEXT H
210 NEXT K
```

A questo punto possiamo finalmente inserire le nostre istruzioni PSET, che ci permetteranno finalmente di visualizzare le tre curve del bioritmo; ricordiamoci però che, mentre la prima deve "accendere" tutti i pixel, la seconda, per poterla distinguere dalla precedente, dovrà "accenderne" uno ogni due e la terza, per lo stesso motivo, uno ogni tre. Quindi:

```
170 PSET(J,Y)
172 IF J MOD 2 = 0 THEN PSET(J,Z)
173 IF J MOD 3 = 0 THEN PSET(J,W)
```

E con questo possiamo considerare di avere completato finalmente il programma, che possiamo vedere nella sua lista completa:

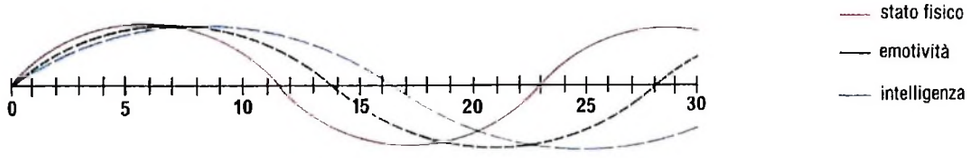
```
10 INPUT"Inserire data di nascita(gg,mm,aa)"
;G,M,A
20 GOSUB 500 'Calcolo n.giorni
30 LET X=N
40 INPUT"Inserire mese richiesto(mm,aa)";M,A
50 LET G=1
60 GOSUB 500 'Calcolo n.giorni
70 LET P=N-X
80 CLS
85 'Calcolo dei giorni di inizio dal termine del
l'ultimo
86 'ciclo per ogni ritmo
90 LET F=P MOD 23
100 LET S=P MOD 28
105 LET I=P MOD 33
110 'Traccia i grafici
111 PRINT"                                        F"
112 PRINT"                                        S"
113 PRINT"                                        I"
114 PRINT
115 PRINT"     5     10     15     20     25     30"
116 FOR L=225 TO 235
117 PSET(L,4)
118 IF L MOD 2 = 0 THEN PSET(L,12)
119 IF L MOD 3 = 0 THEN PSET(L,20)
120 NEXT L
122 FOR K=1 TO 210
130 PSET(K,30)
135 IF K MOD 7 = 0 THEN PSET(K,31)
138 IF K MOD 35= 0 THEN PSET(K,32)
140 NEXT K
150 FOR K=0 TO 29
160 FOR H=1 TO 7
165 LET J=(K*7+H)
166 LET Y=30-30*SIN(6.28*(J+F*7)/(7*23))
167 LET Z=30-30*SIN(6.28*(J+S*7)/(7*28))
168 LET W=30-30*SIN(6.28*(J+I*7)/(7*33))
170 PSET(J,Y)
172 IF J MOD 2 = 0 THEN PSET(J,Z)
173 IF J MOD 3 = 0 THEN PSET(J,W)
```

```
200 NEXT H
210 NEXT K
490 END
500 'Calcola il n. di giorni da una data di rife
rimento
510 IF M-3>=0 THEN LET M=M+1:GOTO 530
520 LET A=A-1:LET M=M+13
530 LET N=INT(365.25*A)+INT(30.6*M)+G
540 RETURN
```

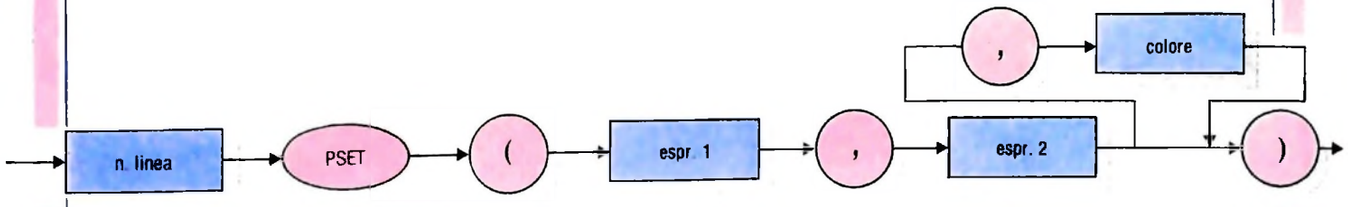
E finalmente possiamo "godercene" le esecuzioni.
Iniziamo con un'esecuzione campione che ci faccia verificare che, se chiediamo il bioritmo corrispondente al mese della nascita, effettivamente le tre curve partono insieme in quel giorno, e hanno periodo corretto:

```
run
Inserire data di nascita(gg,mm,aa)? 1,1,
1980
Inserire mese richiesto(mm,aa)? 1,1980
```

Effettivamente otteniamo la seguente esecuzione:



Carta sintattica di PSET



L'istruzione PSET accede direttamente a un pixel dello schermo e lo "accende". I pixel sono numerati da 0 a 239 a partire da sinistra a destra e da 0 a 63 a partire dall'alto in basso dello schermo.
Nella carta sintattica,

<espr. 1> indica la colonna
<espr. 2> indica la riga.

Ambedue devono rispettare i limiti di variabilità di colonne e righe.
<colore> è un parametro opzionale che:

— se vale 1 "accende" il pixel corrispondente
— se vale 0 "spegne" il pixel corrispondente.

Se viene omesso, viene assunto a 1.

Dall'immagine che compare sullo schermo emergono un certo insieme di piccole osservazioni sulle scelte effettuate durante la costruzione del programma:

- la visualizzazione dell'asse, con la segmentazione in giorni e la scala con ulteriori segmentazioni di 5 in 5 giorni è molto funzionale; senza questa sarebbe stato impossibile valutare correttamente la disposizione delle curve;

- la differenziazione delle curve è pure stata una scelta corretta, in quanto dopo una loro intersezione non sarebbe stato così facile distinguerle l'una dall'altra; inoltre sarebbe comunque necessario poterle distinguere tra loro per conoscere a che cosa ciascuna si riferisce;

- felice la scelta di indicare nell'angolino i tre campioni delle curve, in modo da non dover ricordare le convenzioni adottate;

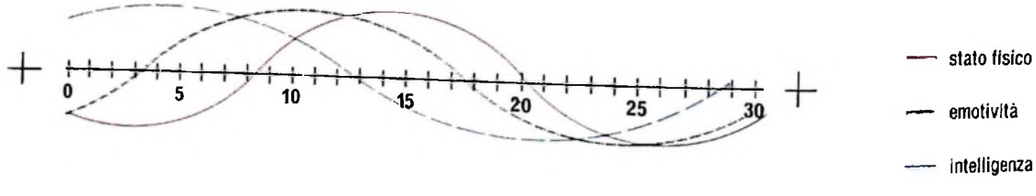
- opportuna la scelta di far variare le curve da 0 a 60 come estensione di righe di pixel, per migliorare la leggibilità;

- corretto il programma che:
  - per il bioritmo del giorno di nascita fa partire le curve correttamente
  - mostra chiaramente i due periodi 23 e 28, e fa intuire che il periodo della terza curva è 33 (anzi, misurando sulla metà della curva, vediamo chiaramente un semiperiodo pari a 16 e mezzo).

Ancora, con date diverse,

```
run
Inserire data di nascita(gg,mm,aa)? 10,5
,1962
Inserire mese richiesto(mm,aa)? 10,1984
```

ci fa ottenere:



Che cosa abbiamo imparato

In questa lezione non abbiamo introdotto elementi innovativi, ma abbiamo completato un programma e osservato criticamente quali sue caratteristiche positive sono state introdotte.

Dall'immagine che compare sullo schermo emergono un certo insieme di piccole osservazioni sulle scelte effettuate durante la costruzione del programma:

- la visualizzazione dell'asse, con la segmentazione in giorni e la scala con ulteriori segmentazioni di 5 in 5 giorni è molto funzionale; senza questa sarebbe stato impossibile valutare correttamente la disposizione delle curve;

- la differenziazione delle curve è pure stata una scelta corretta, in quanto dopo una loro intersezione non sarebbe stato così facile distinguerle l'una dall'altra; inoltre sarebbe comunque necessario poterle distinguere tra loro per conoscere a che cosa ciascuna si riferisce;

- felice la scelta di indicare nell'angolino i tre campioni delle curve, in modo da non dover ricordare le convenzioni adottate;

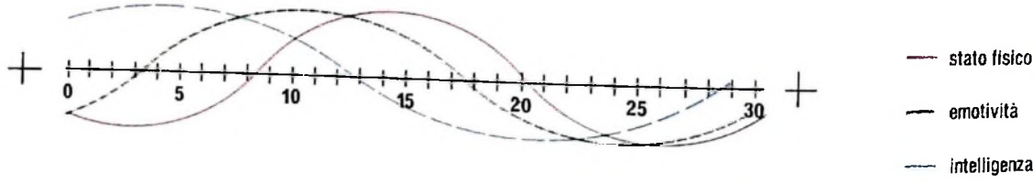
- opportuna la scelta di far variare le curve da 0 a 60 come estensione di righe di pixel, per migliorare la leggibilità;

- corretto il programma che:
  - per il bioritmo del giorno di nascita fa partire le curve correttamente
  - mostra chiaramente i due periodi 23 e 28, e fa intuire che il periodo della terza curva è 33 (anzi, misurando sulla metà della curva, vediamo chiaramente un semiperiodo pari a 16 e mezzo).

Ancora, con date diverse,

```
run
Inserire data di nascita(gg,mm,aa)? 10,5
,1962
Inserire mese richiesto(mm,aa)? 10,1984
```

ci fa ottenere:



Che cosa abbiamo imparato

In questa lezione non abbiamo introdotto elementi innovativi, ma abbiamo completato un programma e osservato criticamente quali sue caratteristiche positive sono state introdotte.

CONTABILITÀ CASALINGA

Un programma di grande utilità che, fornendo rendiconti dettagliati e generali, permette il controllo costante del bilancio familiare.

Quante volte si è cominciato a trascrivere su di un'agenda, o su di un quaderno, le spese sostenute giornalmente e quante volte, al momento di tirare le somme, ci si è resi conto di aver mischiato gli importi, di aver dimenticato qualcosa, o di non riuscire a rendersi conto dell'influenza che ogni singola spesa aveva sul totale generale?
Ecco un programma che risolve questi inconvenienti in maniera semplice: basta ricordarsi di "scaricare" nella memoria dell'M10 le spese sostenute nella giornata associandole alla corrispondente categoria ed ecco, in qualsiasi momento, il saldo generale, quello dettagliato e il grafico di incidenza.

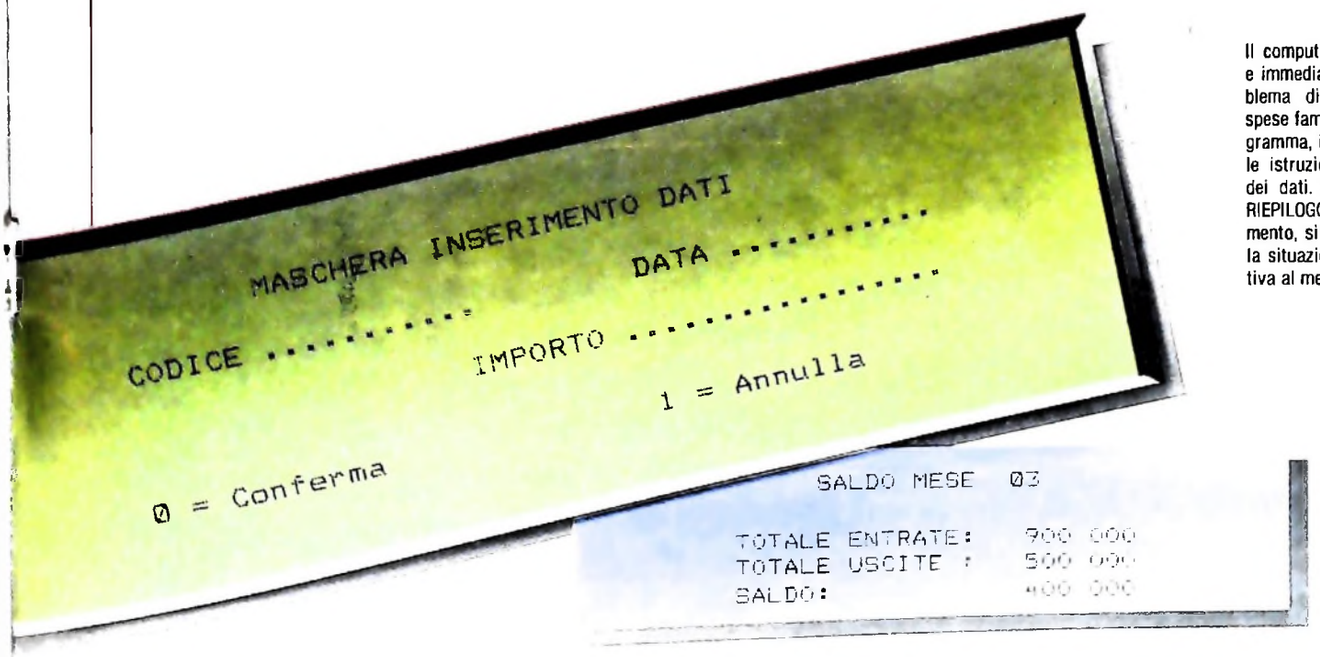
Utilizzo

Il programma è suddiviso in:
1) REGISTRAZIONE USCITE
2) REGISTRAZIONE ENTRATE
3) RIEPILOGO
Il sottoprogramma RIEPILOGO è a sua volta costituito da:
a) saldo mensile;
b) saldo dettagliato per codice con grafico;
c) stampa completa degli archivi.

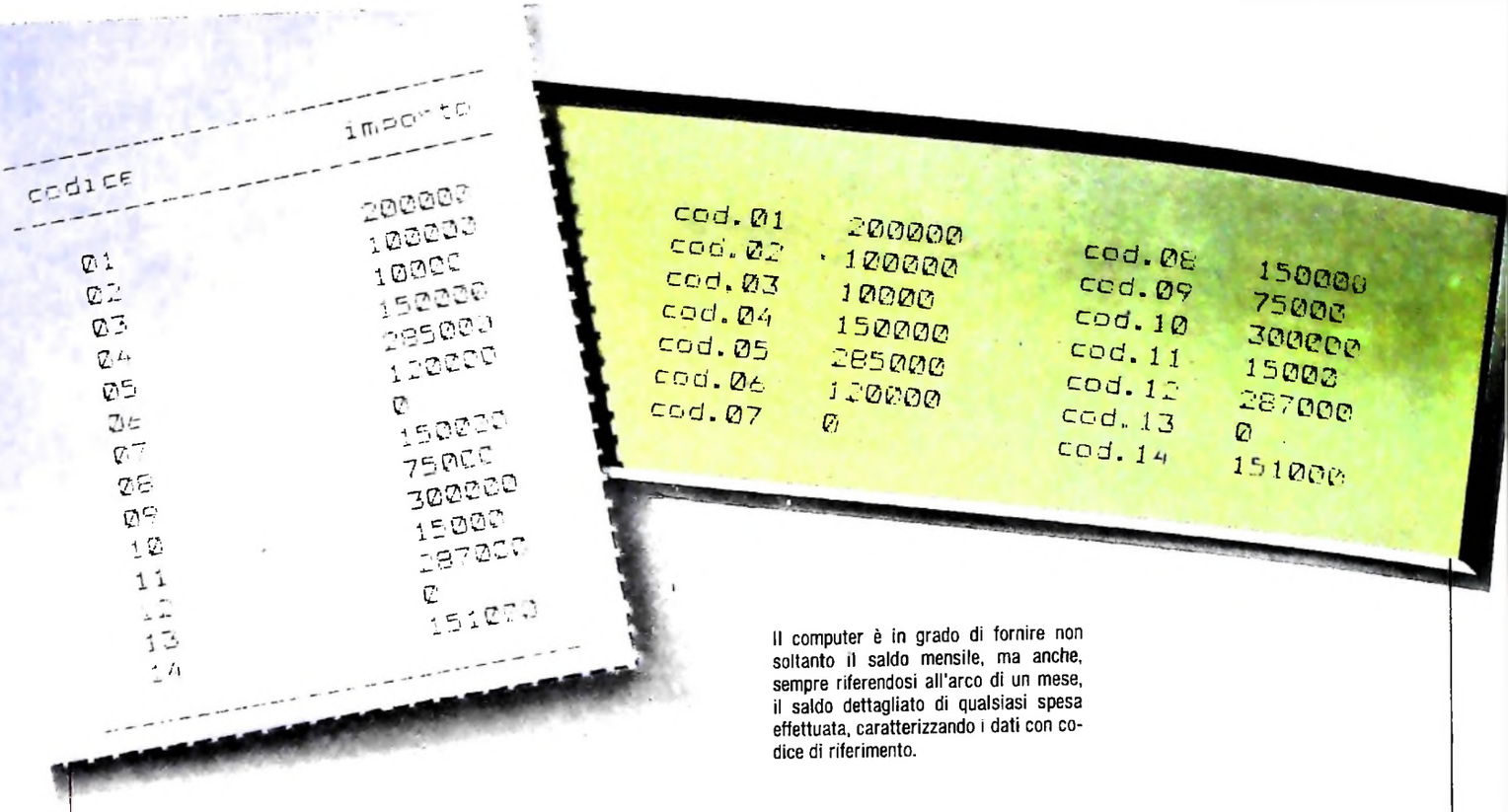
REGISTRAZIONE USCITE
Permette la memorizzazione in un archivio sequenziale (USCITE.DO) delle spese sostenute suddivise in quattordici categorie rappresentate da altrettanti codici numerici:

01	ALIMENTARI		CARBURANTE, BOLLO
02	ABBIGLIAMENTO	08	LIBRI, GIORNALI, RIVISTE
03	AFFITTO, SPESE COND., RISCALDAMENTO	09	SCUOLA
		10	VISITE MEDICHE, FARMACIA
04	GAS	11	SVAGHI
05	LUCE	12	ARREDAMENTO
06	TELEFONO	13	ELETTRODOMESTICI
07	AUTOMOBILE (MANUT.),	14	ALTRE SPESE

Naturalmente le categorie possono essere sostituite con altre più opportune senza dover intervenire sul programma in quanto lo stesso opera direttamente sui codici.
Al momento della selezione del sottoprogramma, il display visualizzerà la maschera di inserimento dati (figura in basso); premendo il tasto ENTER apparirà il cursore direttamente posizionato sul campo CODICE. Inserire ora l'opportuno codice relativo alla spesa sostenuta (il codice è composto da due cifre da 01 a 14) seguito dal tasto ENTER.
Il cursore verrà ora a trovarsi in corrispondenza del campo DATA; inserire la data corrispondente formata da 6 cifre



Il computer rende più agevole e immediato il "faticoso" problema di registrazione delle spese familiari. Stabilito il programma, il display visualizzerà le istruzioni per l'inserimento dei dati. Nel sottoprogramma RIEPILOGO, in qualsiasi momento, si troverà così indicata la situazione del bilancio relativa al mese richiesto.



Il computer è in grado di fornire non soltanto il saldo mensile, ma anche, sempre riferendosi all'arco di un mese, il saldo dettagliato di qualsiasi spesa effettuata, caratterizzando i dati con codice di riferimento.

(MMGGAA) seguita dal tasto ENTER. Per effetto di tale manovra il cursore si posizionerà sul campo IMPORTO. A questo punto si dovrà quindi inserire l'importo relativo alla spesa effettuata e premere il tasto ENTER.
Possiamo confermare l'informazione prodotta premendo il tasto 0 (zero) oppure annullarla premendo il tasto 1.
Inserendo come codice 99 si causerà la chiusura del file, il rilascio della maschera di inserimento e il ritorno al menù principale.

REGISTRAZIONE ENTRATE

Permette la memorizzazione in un archivio sequenziale (ENTRAT.DO) delle entrate suddivise in 5 codici:

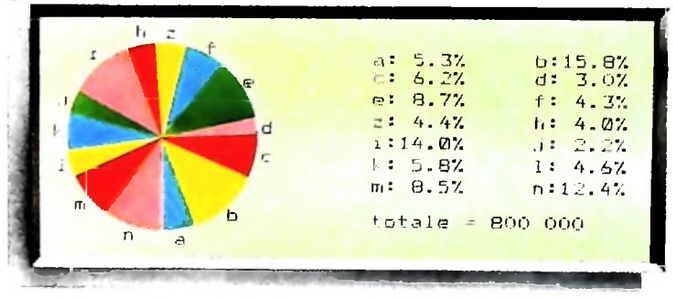
01 SALARIO CAPOFAMIGLIA
02 SALARIO CONIUGE
03 ALTRI SALARI
04 AFFITTI
05 INTERESSI ATTIVI

In questo caso il numero delle voci e la loro natura si possono aumentare e variare senza intervenire sul programma.
Selezionando il sottoprogramma verrà visualizzata una maschera simile alla precedente e le operazioni da effettuare saranno identiche alle precedenti.

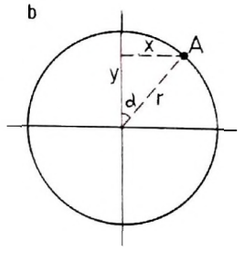
RIEPILOGO

a) Saldo mensile.
Il sottoprogramma, dopo aver richiesto per quale mese eseguire il saldo (da inserirsi sotto forma di numero di 2 cifre), accede agli archivi creati ed effettua le opportune somme fornendo come risultato una videata (figura a pagina precedente). La pressione di un qualsiasi tasto permetterà il ritorno al menù principale.
b) Saldo dettagliato per codice con grafico.
Anche in questo caso verrà richiesto per quale mese effettuare il conteggio; il sottoprogramma accederà quindi agli archivi sommando, codice per codice, gli importi memorizzati. Al termine dell'operazione verrà richiesto di scegliere su quale dispositivo dirottare la stampa: D per display o S per stampante. L'opportuna selezione fornirà il relativo output che assumerà le forme delle figure in alto.
La conseguente pressione di un tasto richiamerà l'esecuzione della routine "grafico a torta". Tale grafico ha la funzione di visualizzare le percentuali dei 14 codici di spesa sul totale generale. La scelta di 14 codici anziché 15 o 18 o 20 è determinata dalle dimensioni del display: infatti tenendo conto che sull'ottava riga viene visualizzato il totale generale, rimangono sette righe utili e, utilizzando due codici per riga, risulta

a



b



$x = r \cdot \sin(\alpha)$
$y = r \cdot \cos(\alpha)$

Il "grafico a torta" visualizza l'incidenza percentuale di ogni singola spesa sul totale delle spese mensili. Per ottenerlo ci si basa sul principio trigonometrico, per cui l'ascissa di un punto sulla circonferenza è data dalla misura del raggio per il seno dell'angolo compreso, l'ordinata dal raggio per il coseno dell'angolo compreso.

14. Il grafico è formato da un cerchio (torta) i cui settori di ampiezza variabile (fette) sono associati a ciascun codice (figura a, in basso).

c) Stampa completa degli archivi.

Effettua la stampa in ordine di inserimento delle informazioni depositate nei due archivi. È possibile in qualsiasi momento cancellare completamente gli archivi (utilizzando la funzione KILL"..."); gli stessi verranno automaticamente ricreati dal programma al primo inserimento.

Il programma

Cominciamo dalla routine generante il grafico a torta.

La subroutine 3000-3070 effettua la somma totale delle spese, sommando i totali T(I) di ciascun codice I (infatti N=14).

La subroutine 4000 calcola le coordinate dei punti appartenenti alla circonferenza che, congiunti con il centro, formeranno i settori relativi (figura b, in basso).

La subroutine 5000-5040 calcola la posizione in cui visualizzare le lettere (a,...,n) relative ai 14 settori. Da notare, nella linea 5020, la presenza del simbolo \ che rappresenta una divisione tra interi, per cui gli operandi sono arrotondati al numero intero più vicino prima di effettuare la divisione e anche il quoziente viene troncato al valore intero più vicino.

La subroutine 6000-6060 ha la funzione di visualizzare le percentuali relative a ciascuna categoria di spesa. Alla riga 6040 il simbolo MOD (divisione modulo) fornisce quel numero intero che è il resto di una divisione tra numeri interi (per esempio: 10 MOD 3 = 1).

Le rimanenti routine non presentano difficoltà di programmazione tali da essere trattate esplicitamente.

```
1 '  ********************************************
2 '  *     CONTABILITA' CASALINGA                *
3 '  *           by CVP                          *
4 '  ********************************************
5 P$=CHR$(27)+"P"
 :Q$=CHR$(27)+"Q"
10 GOTO 8100
47 '*********************************
48 '*      routine grafico a torta    *
49 '*********************************
50
55 N=14
100 CLS
   :P2=3.1416#2
105 GOSUB 3000
107 PRINT@297,"totale= ";SX;
110 GOSUB 1000
120 ANG=0
   :GOSUB 4000
   :LA=0
130 FOR I=1 TO N-1
140 ANG=ANG+P2*T(I)/SX
   :GOSUB 4000
150 GOSUB5000
   :NEXT I
   :ANG=P2
   :GOSUB 5000
165 GOSUB 6000
170 A$=INKEY$
   :IFA$=""THEN 170 ELSE GOTO 8100
1000 RAD=24
    :ANG=0
    :GOSUB 8000
1010 GOSUB 7000
    :ANG=ANG+.3
    :IF ANG<P2 THEN 1010
    :ELSE RAD=24
    :ANG=0
    :GOSUB 7000
    :RETURN
2000 X=RAD*SIN(ANG)+45
    :Y=RAD*COS(ANG)+32
    :PSET(X,Y)
    :RETURN
3000 SX=0
    :FOR I=1 TO N
3050 SX=SX+T(I)
    :NEXT I
3070 CLS
    :RETURN
4000 X=24*SIN(ANG)+45
    :Y=24*COS(ANG)+32
    :LINE(45,32)-(X,Y)
    :RETURN
5000 A=LA+(ANG-LA)/2
    :LA=ANG
5010 X=33*SIN(A)+45
    :Y=33*COS(A)+32
5020 PL=(Y\8)*40+(X\6)
5022 IF PL>320 THEN PL=PL-40
5025 A$=CHR$(96+I)
5030 PRINT@PL,A$;
5040 RETURN
6000 PL=16
    :FM$="": ##.#%"
6010 FOR I=1TON
    :W=T(I)/SX*100'
6020 PRINT@0+PL,USING FM$;CHR$(96+I);W;
6030 PL=PL+12
6040 IF I MOD 2=0 THEN PL=PL+16
6050 NEXT I
6060 RETURN
7000 OX=X
    :OY=Y
    :GOSUB 8000
    :LINE(OX,OY)-(X,Y)
    :RETURN
8000 X=24*SIN(ANG)+45
    :Y=24*COS(ANG)+32
    :RETURN
8100 CLS
8105 PRINT@8,"CONTABILITA' CASALINGA"
    :LINE(0,8)-(239,8)
8110 PRINT@82,P$;"1";Q$;" - REGISTRAZIONE USCITE"
8115 PRINT@122,P$;"2";Q$;" - REGISTRAZIONE ENTRATE"
8120 PRINT@162,P$;"3";Q$;" - RIEPILOGO"
8123 PRINT@202,P$;"4 - FINE LAVORO";Q$
8125 PRINT@252,"seleziona"
8130 A$=INKEY$
    :IF A$="" OR VAL(A$)<1 OR VAL(A$)>4 THEN 8130
8140 ON VAL(A$) GOTO 8500,8600,8700,13000
8490 '*********************************
8492 '*       routine uscite          *
8494 '*********************************
8500 CLS
    :GOSUB 12000
8505 OPEN"cam:uscite"FOPAPPENDAS1
8510 PRINT@89,;
    :LINEINPUT C$
    :IF C$="99" THEN 8550
8515 PRINT@107,;
    :LINEINPUT D$
    :PRINT@180,;
    :LINEINPUT L$
```

```
8520 A1$=INKEY$
    :IF A1$="" THEN 8520
8523 IFA1$()"0" AND A1$()"1" THEN PRINT@242,
 "devi digitare 0 oppure 1!"
    :GOTO 8520
8525 IF A1$="0" THEN 8540
8530 C$=""
    :D$=""
    :L$=""
    :PRINT@89,"..."
    :PRINT@107,"......."
    :PRINT@180,"......."
8535 GOTO 8510
8540 PRINT#1,C$;",";D$;",";L$;",";
8545 PRINT@242,"0=Conferma              1=Annulla"
    :GOTO 8530
8550 CLOSE1
    :GOTO8100
8597 '*********************************
8598 '*      routine entrate          *
8599 '*********************************
8600 CLS
    :GOSUB 12000
8605 OPEN"ram:entrat"FORAPPENDAS1
8610 GOTO 8510
8690 '*********************************
8695 '*      riepilogo mensile        *
8700 '*********************************
8705 GOTO 15000
8710 CLS
    :PRINT@125,"Inserire mese desiderato:";
    :LINEINPUT D2$
8715 OPEN"ram:uscite"FORINPUTAS1
    :K=0
8720 IF EOF(1) THEN 8750
8725 INPUT#1,C$,D$,L$
8730 D1$=MID$(D$,3,2)
    :IF D1$()D2$ THEN 8720
8735 L=VAL(L$)
8740 LT=LT+L
8745 GOTO 8720
8750 IF K=0 THEN LU=LT
    :LT=0
    :CLOSE1
8752 IF K=1 THEN GOTO 8755
8753 OPEN"ram:entrat"FORINPUTAS 1
    :K=1
    :GOTO 8720
8755 LE=LT
    :LT=0
    :CLOSE1
8760 CLS
8765 PRINT@53,"SALDO MESE ";D2$
8770 PRINT@122,P$;"TOTALE ENTRATE:";Q$;"   ";LE
8775 PRINT@162,P$;"TOTALE USCITE :";Q$;"   ";LU
8780 PRINT@202,P$;"SALDO:";Q$;"          ";(LE-LU)
8785 A$=INKEY$
    :IF A$="" THEN 8785 ELSE 8100
8797 '*********************************
8798 '*   routine saldo dettaglio     *
8799 '*********************************
8800 DIM T(14)
8801 CLS
8802 PRINT@42,"Inserire mese desiderato:";
    :LINEINPUT D2$
8805 OPEN"ram:uscite"FORINPUTAS1
8810 IF EOF(1) THEN 8840
8815 INPUT#1,C$,D$,L$
8817 D1$=MID$(D$,3,2)
    :IF D1$()D2$ THEN 8810
8820 C=VAL(C$)
    L=VAL(L$)
8825 T(C)=T(C)+L
8830 GOTO 8810
8840 CLOSE1
    :CLS
    :PRINT@82,"Display o Stampante? (D/S)";
    :LINEINPUT G$
8841 IF G$="D" OR G$="d" THEN GOTO 10000
8842 LPRINT"--------------------------"
8845 LPRINT" codice          totale"
8850 LPRINT"--------------------------"
8860 FOR I=1 TO 14
    :LPRINT TAB(3);I;TAB(18);T(I)
    :NEXT I
8870 LPRINT"--------------------------"
8880 CLS
    :PRINT@120,"Per il grafico a torta premere
enter"
8890 A$=INKEY$
    :IF A$="" THEN 8890
8895 GOTO 50
9000 '*********************************
9005 '*routine stampa completa archivi*
9010 '*********************************
9020 CLS
    :PRINT@120,"Stampa degli archivi in ordine di
inserimento"
    :K1=0
9030 OPEN"ram:uscite"FORINPUTAS 1
    :LPRINT"   ARCHIVIO USCITE"
9031 LPRINT"-----------------------------"
9032 LPRINT" data      codice      importo"
9033 LPRINT"-----------------------------"
9034 LPRINT""
9035 IF EOF(1) THEN 9070
9040 INPUT#1,C$,D$,L$
9045 D1$=LEFT$(D$,2)
    :D2$=MID$(D$,3,2)
    :D3$=RIGHT$(D$,2)
9050 D4$=D1$+"-"+D2$+"-"+D3$
    :LPRINT D4$;TAB(13);C$;TAB(22);L$
9060 GOTO 9035
9070 CLOSE1
    :IF K1=1 THEN 8100
9080 LPRINT" "
    :LPRINT" "
    :LPRINT"    ARCHIVIO ENTRATE"
9090 OPEN"ram:entrat"FORINPUTAS1
9100 K1=1
    :GOTO 9031
10000 CLS
10010P=0
10020for i=1 to 14
10030if i=8 then p=20
10040@P,"cod.";i;"   ";t(i)
10050P=P+40
                :next i
10060a$=inkey$
                 :if a$="" then 10060
10070goto 50
12000 CLS
12010 LINE(0,8)-(234,8)
    PRINT"          MASCHERA INSERIMENTO DATI"
12020 PRINT@82,"CODICE .."
    :PRINT@102,"DATA ......"
    :PRINT@172,"IMPORTO ......."
12030 PRINT@242,"0=Conferma              1=Annulla"
12050 A2$=INKEY$
    IFA2$=""THEN12050
12060 RETURN
13000 CLS
    :PRINT@124,"FINE LAVORO"
    :END
15000 CLS
    :PRINT@15,"RIEPILOGO"
    :LINE(0,8)-(239,8)
15010 PRINT@82,"1 - Saldo mensile"
15020 PRINT@122,"2 - Saldo dett. per codice
con grafico"
15030 PRINT@162,"3 - Stampa completa degli archivi"
15040 PRINT@202,"4 - Ritorno menu principale"
15050 PRINT@252,"SELEZIONA"
15060 A$=INKEY$
    :IF A$="" OR A$<"1" OR A$>"4" THEN 15060
15070 ON VAL(A$) GOTO 8710,8800,9000,8100
```

IL LINGUAGGIO PASCAL (III)

Proseguiamo nell'analisi di questo moderno linguaggio esaminandone le caratteristiche relative alla capacità di definire tipi di dati.

Il Pascal presenta, nella storia dei linguaggi di programmazione, l'aspetto innovativo di permettere all'utente stesso la definizione dei propri tipi di dati, cioè dei valori che una variabile dichiarata di un certo tipo può assumere. Ricordiamo che il tipo di una variabile è un suo attributo che identifica l'insieme dei valori che questa può assumere, l'insieme delle operazioni che su di essa possono essere fatte, e il loro comportamento.
In Pascal esiste una dichiarativa apposita per definire un tipo di dati e una, diversa dalla precedente, per richiedere l'allocazione delle variabili. Per esempio:
TYPE giorno = 1..31;
VAR d: giorno;
specificano che "giorno" è il nome di un tipo di dati, che contempla tutti i valori interi tra 1 e 31, e che "d" è il nome di una variabile di tale tipo.
In Pascal si distinguono due classi fondamentali di tipi di dati: quelli SEMPLICI e quelli STRUTTURATI. Mentre i primi definiscono insiemi di valori ciascuno dei quali è costituito da una sola costante (per esempio gli interi), i secondi individuano valori ciascuno dei quali è composto da un insieme di valori elementari (come per esempio un array).

Tipi semplici

Ci sono due modi per definire tipi semplici in Pascal: per ENUMERAZIONE e per SUBRANGE.
ENUMERAZIONE. Il modo più naturale per definire quali costanti identificano valori di un certo tipo è di elencarle:

TYPE giorno = (lun, mar, mer, gio, ven, sab, dom);

definisce un insieme di valori che, mnemonicamente, ci ricorda i giorni della settimana; mentre in un tradizionale linguaggio di programmazione il programmatore avrebbe dovuto fare un'operazione mentale di rappresentazione e usare, per esempio, valori interi per rappresentare lo stesso concetto (pensando a 1 come lunedì, a 2 come martedì e così via), qui l'operazione di rappresentazione interna della costante è effettuata dal compilatore, e lun, mar,... diventano vere e proprie COSTANTI che il programma può usare. Così
VAR x:giorno;
permette di scrivere istruzioni come
x: = gio;
IF x = sab THEN... e così via.

Una definizione enumerativa induce un naturale ordinamento nelle costanti introdotte, cosicché, riferendoci all'esempio, risulta che
lun<mar<mer...<dom
La definizione per enumerazione è potente, ma scomoda per la sua complessità. Vi sono quindi a disposizione dei TIPI PREDEFINITI, cioè già definiti al nostro posto. Sono:
- INTEGER (gli interi);
- REAL (i reali);
- BOOLEAN (i "booleani", ovvero i valori logici "vero" e "falso");
- CHAR (i caratteri).

SUBRANGE. La parola inglese "subrange" indica un intervallo in un insieme ordinato; poiché le definizioni enumerative inducono un ordinamento, è possibile prendere delle "porzioni" di tali insiemi per definir nuovi tipi. Per esempio:
TYPE giorno-lavorativo=lun...ven;
individua un tipo per SUBRANGE, in cui sono permessi tutti i valori da lun a ven compresi. Una variabile di tale tipo non potrà assumere i valori sab e dom.

Tipi strutturati

Il Pascal mette a disposizione "costruttori" di tipo strutturato che permettono di definire i tipi strutturati a partire da quelli semplici. I principali costruttori sono:
- ARRAY
- RECORD
- FILE
- SET

ARRAY. La struttura array consiste in un insieme di elementi, tutti dello stesso tipo, accessibili mediante il valore di un "indice". Per esempio:
TYPE mese=ARRAY [1..31] OF giorno;
VAR x,y: mese;
definisce il tipo "mese" in termini di un array di 31 elementi, numerati da 1 a 31, ciascuno dei quali è di tipo "giorno"; le due variabili x e y sono dichiarate di tipo mese, ed è possibile scrivere istruzioni come:
x: = y;
x [4]: = mar;
ove si vede che è permesso assegnare un intero array (y) a un altro dello stesso tipo (x), ovvero accedere al singolo elemen-

IL LINGUAGGIO PASCAL (III)

Proseguiamo nell'analisi di questo moderno linguaggio esaminandone le caratteristiche relative alla capacità di definire tipi di dati.

Il Pascal presenta, nella storia dei linguaggi di programmazione, l'aspetto innovativo di permettere all'utente stesso la definizione dei propri tipi di dati, cioè dei valori che una variabile dichiarata di un certo tipo può assumere. Ricordiamo che il tipo di una variabile è un suo attributo che identifica l'insieme dei valori che questa può assumere, l'insieme delle operazioni che su di essa possono essere fatte, e il loro comportamento.
In Pascal esiste una dichiarativa apposita per definire un tipo di dati e una, diversa dalla precedente, per richiedere l'allocazione delle variabili. Per esempio:
TYPE giorno = 1..31;
VAR d: giorno;
specificano che "giorno" è il nome di un tipo di dati, che contempla tutti i valori interi tra 1 e 31, e che "d" è il nome di una variabile di tale tipo.
In Pascal si distinguono due classi fondamentali di tipi di dati: quelli SEMPLICI e quelli STRUTTURATI. Mentre i primi definiscono insiemi di valori ciascuno dei quali è costituito da una sola costante (per esempio gli interi), i secondi individuano valori ciascuno dei quali è composto da un insieme di valori elementari (come per esempio un array).

Tipi semplici

Ci sono due modi per definire tipi semplici in Pascal: per ENUMERAZIONE e per SUBRANGE.
ENUMERAZIONE. Il modo più naturale per definire quali costanti identificano valori di un certo tipo è di elencarle:

TYPE giorno = (lun, mar, mer, gio, ven, sab, dom);

definisce un insieme di valori che, mnemonicamente, ci ricorda i giorni della settimana; mentre in un tradizionale linguaggio di programmazione il programmatore avrebbe dovuto fare un'operazione mentale di rappresentazione e usare, per esempio, valori interi per rappresentare lo stesso concetto (pensando a 1 come lunedì, a 2 come martedì e così via), qui l'operazione di rappresentazione interna della costante è effettuata dal compilatore, e lun, mar,... diventano vere e proprie COSTANTI che il programma può usare. Così
VAR x:giorno;
permette di scrivere istruzioni come
x: = gio;
IF x = sab THEN... e così via.

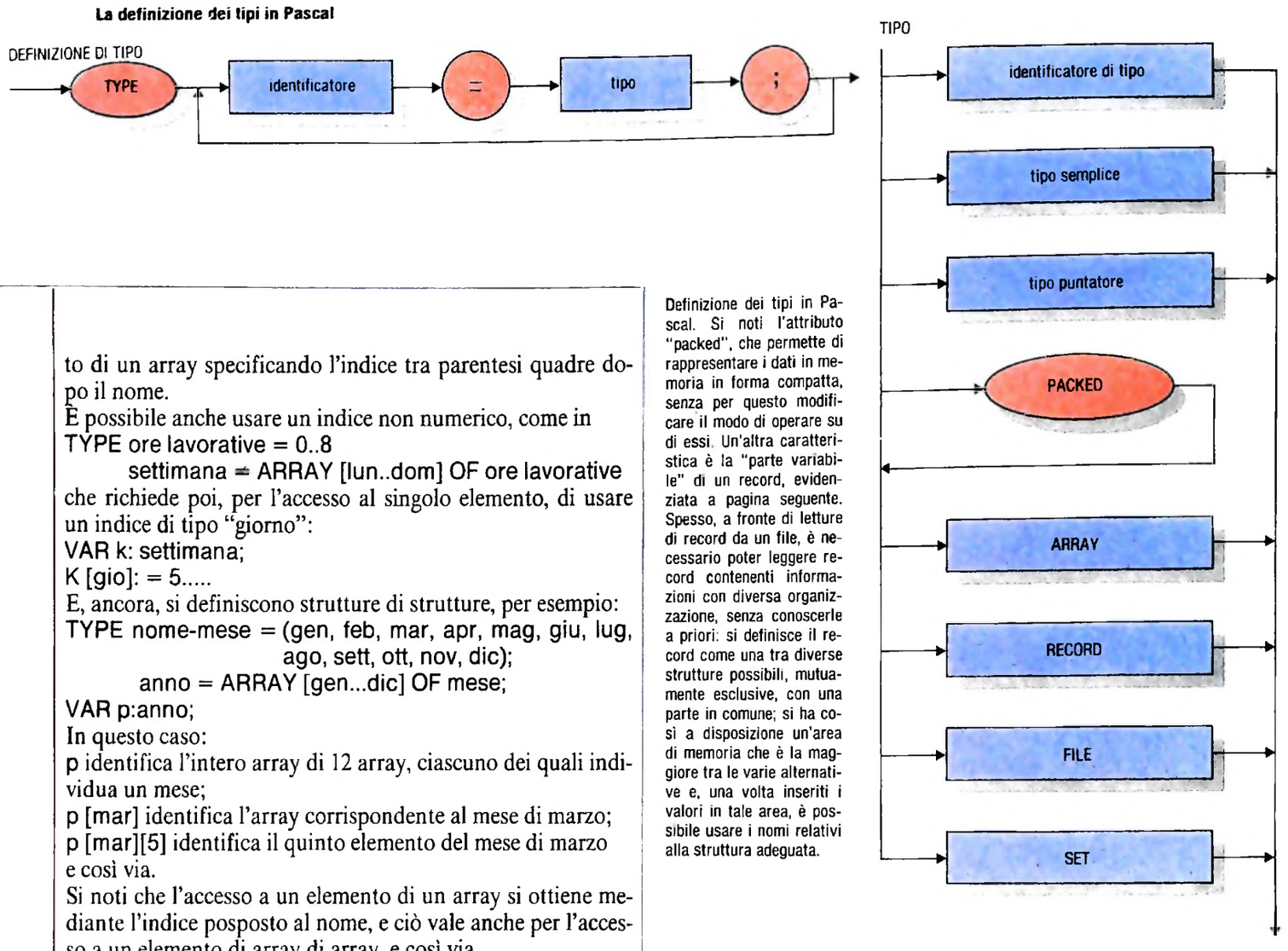
Una definizione enumerativa induce un naturale ordinamento nelle costanti introdotte, cosicché, riferendoci all'esempio, risulta che
lun<mar<mer...<dom
La definizione per enumerazione è potente, ma scomoda per la sua complessità. Vi sono quindi a disposizione dei TIPI PREDEFINITI, cioè già definiti al nostro posto. Sono:
- INTEGER (gli interi);
- REAL (i reali);
- BOOLEAN (i "booleani", ovvero i valori logici "vero" e "falso");
- CHAR (i caratteri).

SUBRANGE. La parola inglese "subrange" indica un intervallo in un insieme ordinato; poiché le definizioni enumerative inducono un ordinamento, è possibile prendere delle "porzioni" di tali insiemi per definir nuovi tipi. Per esempio:
TYPE giorno-lavorativo=lun...ven;
individua un tipo per SUBRANGE, in cui sono permessi tutti i valori da lun a ven compresi. Una variabile di tale tipo non potrà assumere i valori sab e dom.

Tipi strutturati

Il Pascal mette a disposizione "costruttori" di tipo strutturato che permettono di definire i tipi strutturati a partire da quelli semplici. I principali costruttori sono:
- ARRAY
- RECORD
- FILE
- SET

ARRAY. La struttura array consiste in un insieme di elementi, tutti dello stesso tipo, accessibili mediante il valore di un "indice". Per esempio:
TYPE mese=ARRAY [1..31] OF giorno;
VAR x,y: mese;
definisce il tipo "mese" in termini di un array di 31 elementi, numerati da 1 a 31, ciascuno dei quali è di tipo "giorno"; le due variabili x e y sono dichiarate di tipo mese, ed è possibile scrivere istruzioni come:
x: = y;
x [4]: = mar;
ove si vede che è permesso assegnare un intero array (y) a un altro dello stesso tipo (x), ovvero accedere al singolo elemen-



Definizione dei tipi in Pascal. Si noti l'attributo "packed", che permette di rappresentare i dati in memoria in forma compatta, senza per questo modificare il modo di operare su di essi. Un'altra caratteristica è la "parte variabile" di un record, evidenziata a pagina seguente. Spesso, a fronte di letture di record da un file, è necessario poter leggere record contenenti informazioni con diversa organizzazione, senza conoscerle a priori: si definisce il record come una tra diverse strutture possibili, mutuamente esclusive, con una parte in comune; si ha così a disposizione un'area di memoria che è la maggiore tra le varie alternative e, una volta inseriti i valori in tale area, è possibile usare i nomi relativi alla struttura adeguata.

to di un array specificando l'indice tra parentesi quadre dopo il nome.
È possibile anche usare un indice non numerico, come in

```
TYPE ore lavorative = 0..8
     settimana = ARRAY [lun..dom] OF ore lavorative
```

che richiede poi, per l'accesso al singolo elemento, di usare un indice di tipo "giorno":

```
VAR k: settimana;
K [gio]: = 5.....
```

E, ancora, si definiscono strutture di strutture, per esempio:

```
TYPE nome-mese = (gen, feb, mar, apr, mag, giu, lug,
                  ago, sett, ott, nov, dic);
     anno = ARRAY [gen...dic] OF mese;
VAR p:anno;
```

In questo caso:
p identifica l'intero array di 12 array, ciascuno dei quali individua un mese;
p [mar] identifica l'array corrispondente al mese di marzo;
p [mar][5] identifica il quinto elemento del mese di marzo
e così via.
Si noti che l'accesso a un elemento di un array si ottiene mediante l'indice posposto al nome, e ciò vale anche per l'accesso a un elemento di array di array, e così via.
RECORD. Il record definisce una struttura di dati composta da un certo insieme di componenti di tipo diverso, accessibili mediante il loro nome. Per esempio:

```
TYPE data = RECORD
                giorno:1..31;
                mese:gen..dic;
                anno:1900..2000
            end;
VAR x, y: data;
```

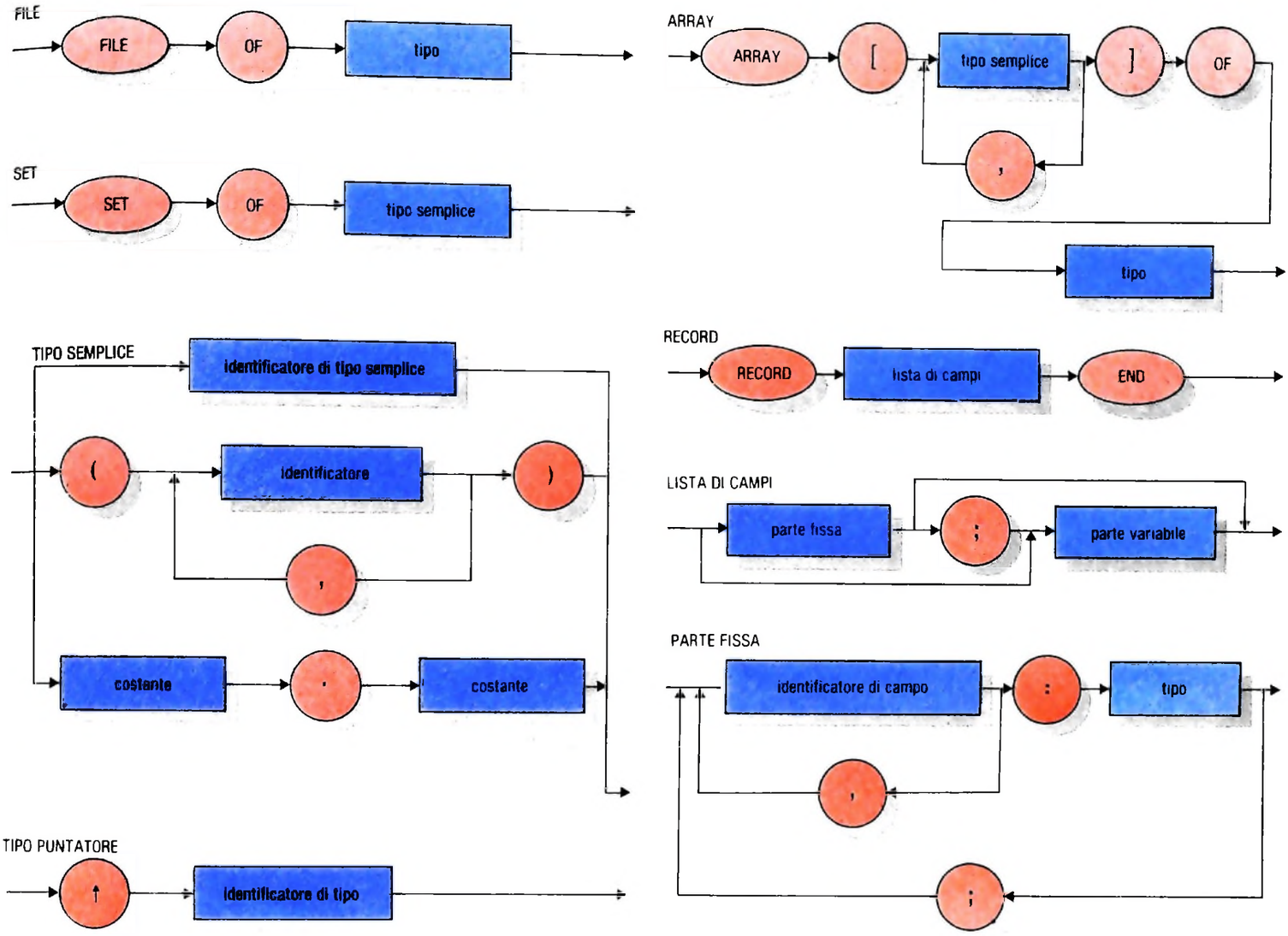
definiscono un record di tre campi e due variabili x e y di quel tipo. È quindi possibile scrivere istruzioni come

```
x:=y;
x. mese:=y. mese;
x. anno:=1984;
```

tra le quali la prima assegna alla variabile x, pensata come terna di variabili semplici l'intera terna costituita dalla variabile y, la seconda assegna al campo mese della variabile x il campo mese della variabile y e l'ultima assegna il valore 1984 al campo anno della variabile x.
Si osservi il modo con cui è possibile accedere al campo di un record: la giustapposizione del nome del campo alla destra del nome della variabile separato con un ".".

Come al solito, è possibile avere strutture composte, e il campo di un record può essere a sua volta un record o un array, e così via. Le modalità di accesso ai vari componenti più elementari seguono le solite regole estremamente uniformi: la specificazione di un tipo per gli array e il nome del campo per i campi dei record, eventualmente applicati ripetutamente uno dopo l'altro. Per esempio nella struttura

```
TYPE nome=ARRAY[1..20] OF char;
     genere=(romanzo, manuale, saggio, poesia);
     libro=RECORD
                autori:ARRAY[1..4] OF nome;
                editore: nome;
                titolo:ARRAY [1..40] OF char;
                tipo:genere
                collocazione:RECORD
                                armadio: 'A'..'Z';
                                scaffale:1..9
                                posizione:1..100
                             END;
           END;
```



VAR x,y,z:libro;
definisce la struttura di un "libro" dal punto di vista di una biblioteca, e tre variabili di tale tipo; è possibile ora scrivere:
x che rappresenta un intero libro;
x. autori che rappresenta l'insieme degli (eventuali) 4 autori del libro;
x. autori [1] che rappresenta il primo dei 4 autori;
x. autori [1][1] che individua l'iniziale del primo autore;
x. collocazione. scaffale che individua lo scaffale in cui è collocato il libro... e così via.

FILE. Il file è una struttura di dati costituita da elementi dello stesso tipo, in numero non noto a priori, accessibili in ordine strettamente sequenziale, e solo per leggerli uno dopo l'altro, oppure solo per scriverli uno dopo l'altro.
Questa struttura è specificamente basata sulla necessità di usare file SEQUENZIALI come supporto esterno di memoria, come per esempio le cassette magnetiche, e le sue caratteristiche si adattano specificamente a questo scopo.
Per disporre le informazioni della nostra biblioteca su un file, avendo già la definizione del tipo "libro", scriviamo

TYPE biblioteca = FILE OF libro;
var miabibl:biblioteca;
La variabile "miabibl", nella realtà della struttura fisica, è realizzata come un'area di memoria capace di contenere un solo elemento del file (in questo caso, di tipo "libro"); qui è caricato il primo elemento disponibile del file fisico che si trova sul supporto esterno (come la cassetta), se si opera in lettura, o le informazioni da registrare nel primo posto libero del supporto esterno, se si opera in scrittura. Quindi, la variabile di tipo file non occupa tutta la memoria corrispondente al file, ma solo quella necessaria per costruire una "finestra" di comunicazione con il supporto fisico.
Il Pascal dispone di procedure predefinite per operare con i file:
— reset che predispone un file a essere usato in lettura;
— rewrite che predispone il file a essere usato in scrittura;
— read che opera la lettura;
— write che opera la scrittura;
e le funzioni:
— eof che individua la fine dei dati in un file;

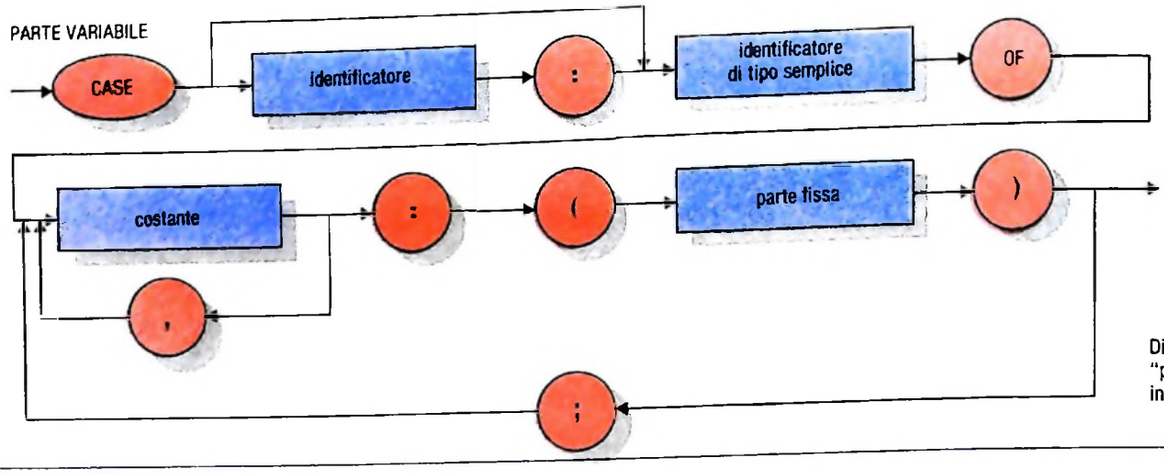


Diagramma sintattico della "parte variabile" di un record in Pascal.

— eoln che individua la fine di una riga, quando si forniscono dati da un terminale, con numerose righe.
SET. La struttura/set definisce un insieme di elementi di tipo semplice. Per esempio la definizione:

```
TYPE turni=SET OF lun...vern;
VAR t:turni;
```

definisce una variabile t di tipo set che può contenere come valore un qualsiasi sottoinsieme degli elementi da lun a ven; è possibile, per esempio, effettuare assegnamenti come

```
t:=[lun,gio];
```

per dire che lun e gio sono gli elementi che compongono l'insieme t; inoltre è interessante l'uso dell'operatore IN:

```
IF ven IN t THEN...
```

se ven è uno degli elementi che sono presenti in t, allora la condizione risulta vera, altrimenti risulta falsa.

I puntatori (pointer)

Oltre ai costruttori esaminati, esiste anche il POINTER, che difficilmente può essere classificato con i criteri esaminati. Un pointer è un tipo di dato che specifica che una variabile deve avere come valore l'indirizzo di un'altra variabile. Per esempio:

```
TYPE punt=^integer;
VAR x:punt;
```

specifica che x è una variabile che può contenere solo indirizzi di variabili di tipo intero.
Il Pascal mette a disposizione una procedura "new", che permette di allocare dinamicamente le variabili durante l'esecuzione del programma: la scrittura

```
new (x);
x^:=1000;
```

richiede che venga allocata una variabile il cui indirizzo deve essere inserito in x; poiché x può "puntare" solo a interi, la variabile allocata sarà di tipo intero; inoltre, la seconda istruzione usa x per accedere alla variabile puntata (mediante la notazione x^), alla quale assegna il valore 1000.
Questo tipo di dati è significativo nella costruzione delle "strutture di dati dinamiche", come liste, code, alberi ecc. Per esempio, una lista può essere definita come

```
TYPE lista=RECORD;
        valore:integer;
        prossimo:puntatore;
     END;
  puntatore=^lista;
VAR x:puntatore;
```

È disponibile una costante NIL che indica che un puntatore è vuoto; così

```
x:=NIL
```

inizializza la lista come vuota,

```
new(x)
```

"attacca" un elemento a x,

```
x^valore:=100
x^prossimo:=NIL
```

inseriscono nell'elemento allocato il valore 100 e l'indicazione che la lista è terminata; operazioni come

```
new(x^prossimo)
```

permettono di attaccare alla lista nuovi elementi.

Osservazioni finali

Il Pascal, che presenta caratteristiche estremamente sofisticate per la modellazione delle entità di problema in termini di tipi di dati, può considerarsi un punto fermo nella storia dei linguaggi di programmazione.
È significativa l'assenza di file "ad accesso diretto"; questa è legata al fatto che tali tipi di file sono spesso definiti nelle loro modalità operative sulla base delle esigenze dell'utente medio, piuttosto che su di un modello di struttura preciso e raffinato dal punto di vista della definizione; la possibilità di accedere a file in modo diretto, senza dover scorrere tutto il file è una necessità per garantire l'efficienza a un programma. La maggior parte dei linguaggi Pascal disponibili a livello di programmazione industriale mettono a disposizione primitive non standard che permettono di accedere a un record che si trova in una certa posizione (file ad accesso "random" o "diretto"), o a un record che contiene, in un opportuno campo detto "chiave", un valore specificato (file "indexed"). Tali primitive non sono inserite forzando la logica del linguaggio, ma vengono messe a disposizione come procedure, del tutto simili a la read e la write, predefinite, anche se non facenti parte dello standard.

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 ulteriori informazioni inviate questo coupon a Olivetti M10 Punto di Vendita c/o Ing. C. Olivetti & C. S.p.A. Divisione Personal Computer, Via Meravigli 12, 20123 Milano

NOME COGNOME

VIA N

CAP CITTÀ

TELEFONO