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

19 CORSO PRATICO COL COMPUTER

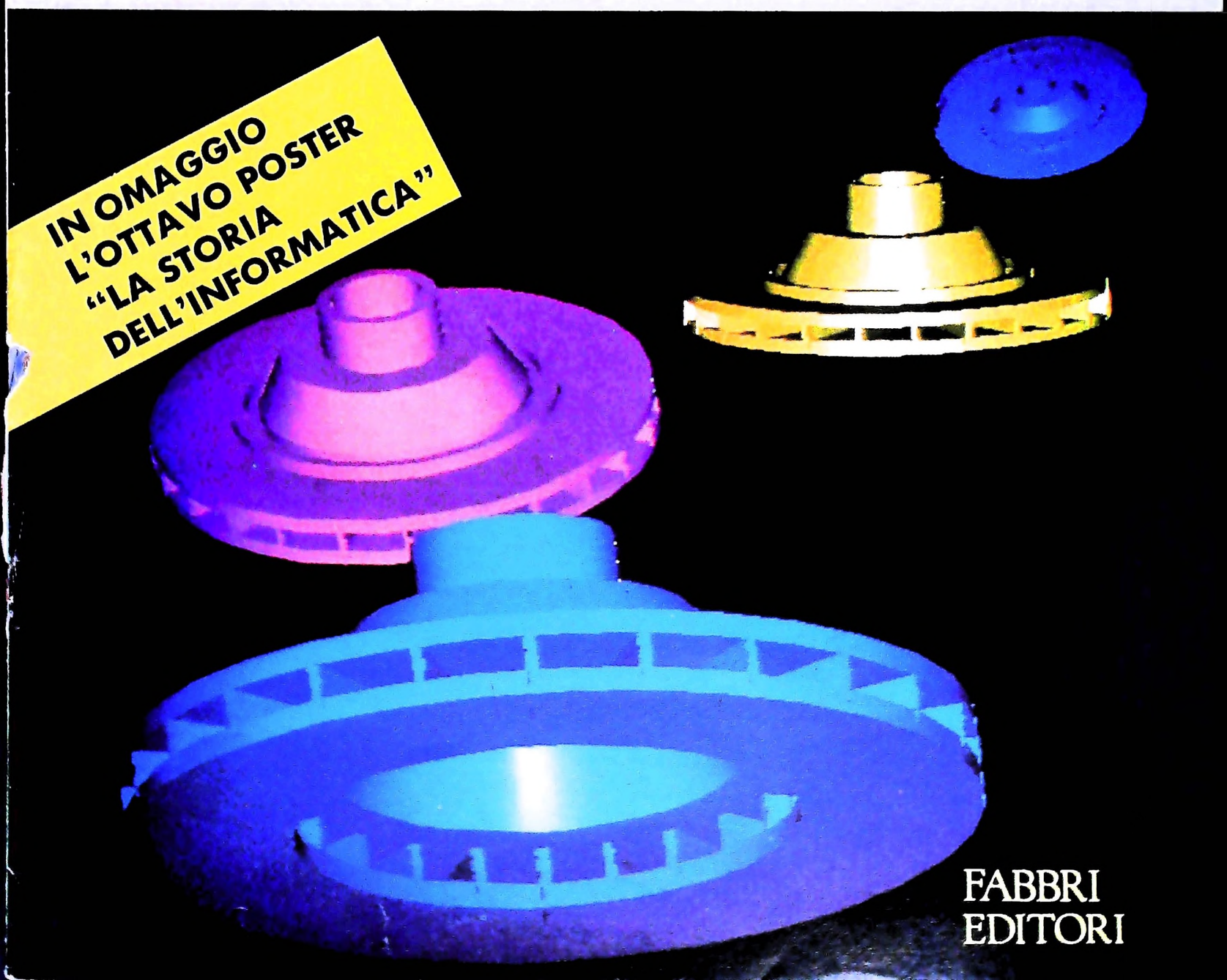
421719

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

IN OMAGGIO L'OTTAVO POSTER "LA STORIA DELL'INFORMATICA"



FABBRI EDITORI

Cari lettori,
con questo numero sospendiamo per la pausa estiva l'uscita del
CORSO PRATICO COL COMPUTER
Arrivederci in edicola il 30 agosto con il n. 20.
Buone vacanze!

La redazione

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
TULLIO CHERSI, 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, GIANCARLO MAURI, GIANLUIGI ROCCA, Eidos (TIZIANO BRUGNETTI), 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

Coordinatore settore scientifico
UGO SCAIONI

Redazione
MARINA GIORGETTI
LOGICAL STUDIO COMMUNICATION

Art Director
CESARE BARONI

Impaginazione
BRUNO DE CHECCHI
PAOLA ROZZA

Programmazione Editoriale
ROSANNA ZERBARINI
GIOVANNA BREGGÈ

Segretarie di Redazione
RENATA FRIGOLI
LUCIA MONTANARI

AVVISO AI LETTORI

Con il n. 20 sarà in edicola la copertina per rilegare il secondo volume del "Corso pratico col computer". Prenotatela dal vostro edicolante

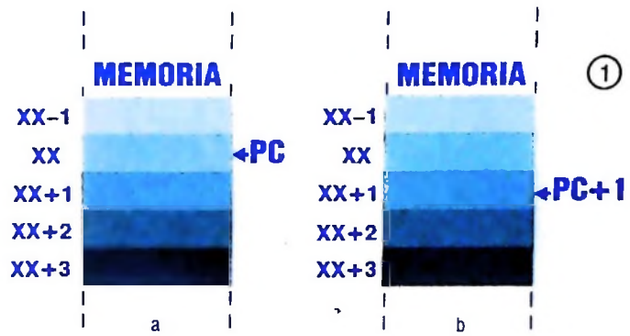
Corso Pratico col Computer - Copyright © sul fascicolo 1984 Gruppo Editoriale Fabbri, Bompiani, Sonzogno, Etas S.p.A., Milano - Copyright © sull'opera 1984 Gruppo Editoriale Fabbri, Bompiani, Sonzogno, Etas S.p.A., Milano - Prima Edizione 1984 - Direttore responsabile GIOVANNI GIOVANNINI - Registrazione presso il Tribunale di Milano n. 135 del 10 marzo 1984 - Iscrizione al Registro Nazionale della Stampa n. 00262, vol. 3, Foglio 489 del 20.9.1982 - Stampato presso lo Stabilimento Grafico del Gruppo Editoriale Fabbri S.p.A. Milano - Diffusione Gruppo Editoriale Fabbri S.p.A. via Mecenate, 91 - tel. 50951 - Milano - Distribuzione per l'Italia: A. & G. Marco s.a.s., via Fortezza 27 - tel. 2526 - Milano - Pubblicazione periodica settimanale - Anno I - n. 19 - esce il giovedì - Spedizione in abb. postale - Gruppo II/70. L'Editore si riserva la facoltà di modificare il prezzo nel corso della pubblicazione, se costretto da mutate condizioni di mercato.

L'UAMICRO I - PARTE II

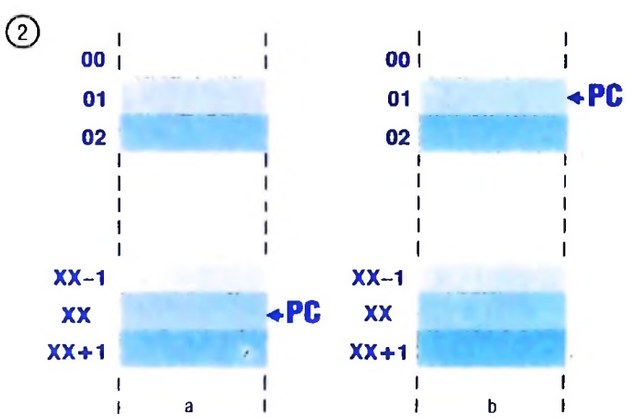
Prosegue l'analisi dei componenti dello schema a blocchi del microprocessore UAMICRO I.

P.C. (Contatore di Programma)

Il P.C. è un registro capace di contare da 0 fino a 255, cioè 256 locazioni in totale perché anche l'indirizzo 0 è una locazione; in notazione esadecimale si va da 00H fino a FFH (con la lettera H si indica appunto che il numero è esadecimale). La sua funzione è quella di emettere indirizzi validi; vedremo più avanti quando un indirizzo, sia di memoria come di qualsiasi elemento esterno esistente è considerato valido. Ciò vuol dire che se mettiamo il programma, per esempio, a partire dalla casella di memoria 00H e gli facciamo puntare il nostro P.C., cioè mettiamo il P.C uguale a zero, quando diamo il comando di START (inizio), il microprocessore incomincia proprio da quella locazione ad eseguire il programma, autoincrementandosi quando preleva (FETCH) un valore dalla memoria.
Il nome dei comandi del BUS di controllo che arrivano al P.C. è IPC che vuol dire INCREMENTA il P.C.; cioè quando IPC = 1 al contenuto del P.C. in quel momento, all'arrivo del CLOCK, viene sommato un 1, in modo che già punti alla locazione di memoria successiva (figura 1). Questa caratteristica soddisfa in pieno la prima delle strutture di controllo, cioè la sequenziale. Con CLR = 1 il P.C. viene azzerato per cui all'uscita si ha 00H (figura 2). Con EPC = 1 il contenuto del P.C. passa al BUS in forma asincrona.



Come il Contatore di Programma (P.C.) punta alla memoria. In a e b, rispettivamente, il valore di P.C. prima e dopo l'incremento.



Come viene azzerato il P.C. In a e b, rispettivamente, posizione del P.C. prima del CLR e dopo il CLR.

I.R. (Registro di Istruzione)

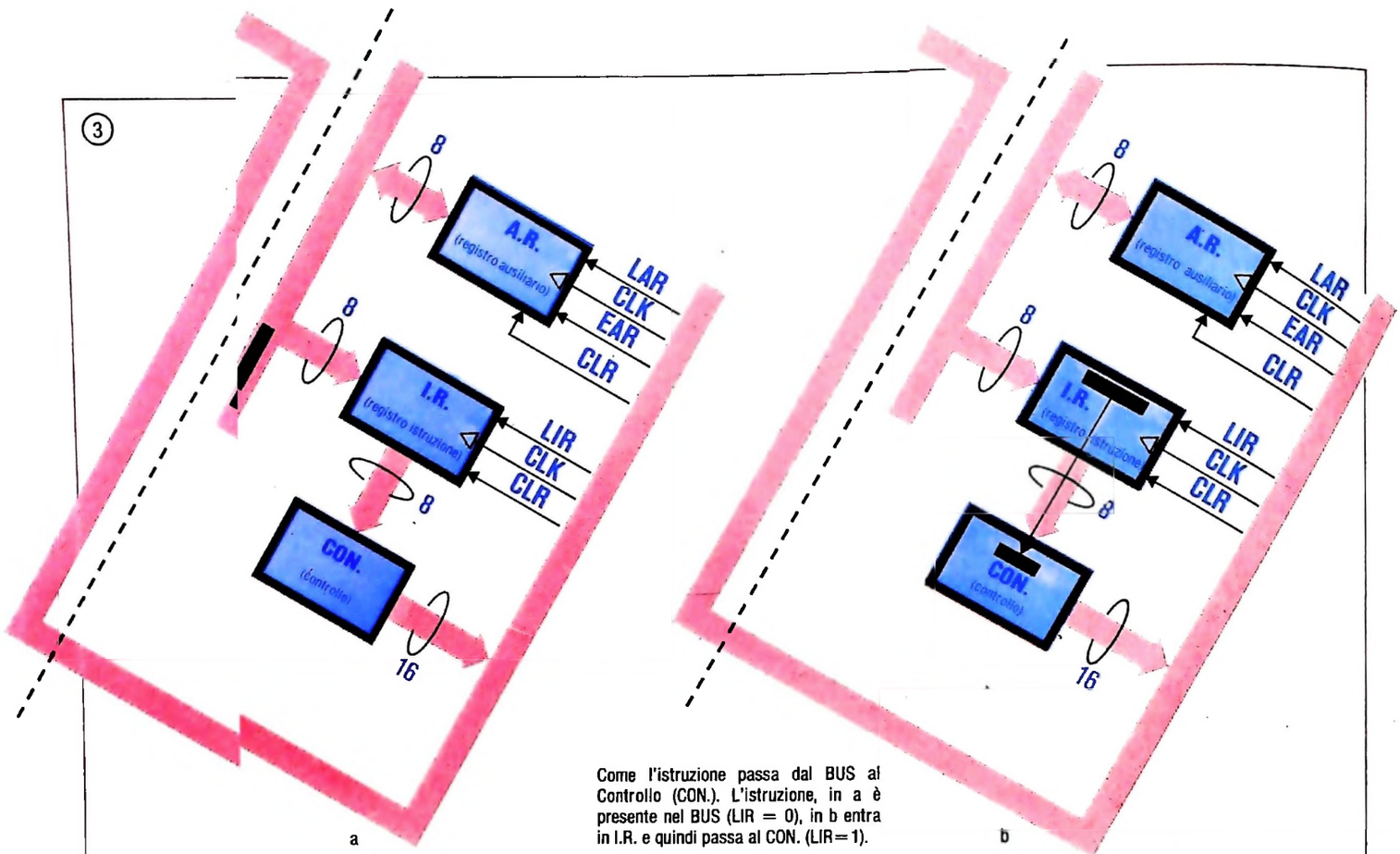
Il registro I.R. è un registro che in realtà fa parte del sistema di controllo di tutto il microprocessore. In questa prima versione della famiglia UAMICRO si è tenuto separato per rendere più facile la comprensione della sua funzione. Quando abbiamo il comando LIR=1, all'arrivo della transizione positiva del CLOCK, il contenuto del BUS, che all'inizio di qualsiasi programma deve essere sempre il codice di un'istruzione e come tale viene trattato, entra nel registro I.R. e, in forma asincrona, passa al registro CON. (Controllo) dove viene opportunamente elaborato (figura 3). Con il comando CLR=1 il contenuto del Registro I.R. viene azzerato.

A.R. (Registro Ausiliario)

I registri ausiliari, sempre numerosi in qualsiasi microprocessore, sono delle piccole memorie interne che servono come deposito momentaneo dei dati per una loro corretta elaborazione. Quasi sempre sono inaccessibili all'utente, che non sa né della loro esistenza né conosce la forma di accedere a essi. Con il comando LAR=1, il contenuto del BUS passa internamente al Registro Ausiliario, alla transizione positiva del CLOCK, mentre con EAR=1 il suo contenuto passa al BUS in forma asincrona. Con il comando CLR=1 il contenuto di A.R. diventa zero.

CON. (Unità di Controllo)

Il controllo è senza dubbio il registro più importante di tutto il microprocessore. Esso trasforma il microprocessore in una macchina automatica per l'elaborazione dei dati.
Vediamo come funziona. Prima di iniziare qualsiasi processo il CON., attraverso il segnale CLR del BUS di controllo, az-



Come l'istruzione passa dal BUS al Controllo (CON.). L'istruzione, in a è presente nel BUS (LIR = 0), in b entra in I.R. e quindi passa al CON. (LIR=1).

in RESET ai registri I.R., A.R. e P.C., per cui i zera, cioè dà u P.C. presenteranno degli zeri al BUS di indi- registri A.R. ando saranno attivati attraverso i rispettivi co- rizzi e dati qu ABLE; mentre il registro I.R. presenterà degli mandi di EN lo stesso.

zeri al control ircuito di controllo viene generato anche il se- Nello stesso c CK che, come detto altre volte, sincronizza tut- gnale di CLO ni siano esse interne oppure esterne al micro- te le operazio remo modo di vedere più avanti come questa processore; a ossibile.

operazione è l trollo che esce dal CON. è composto da 16 li- Il BUS di con delle quali corrisponde una funzione ben pre- nee, a ognuna ata con le seguenti abbreviazioni che sono poi cisa, denomin troviamo nei registri:

le stesse che ri A, LA, SU, EU, LB, EB, IPC, EPC, LAR, CLR, CLK, E , W/R, VMA.

EAR, LIR, LM questi comandi già conosciamo il significato, Di molti di c lo vedremo più avanti. Questa parola di con- mentre di altr ogni transizione positiva del CLOCK e il suo trollo varia a na come e quale registro o quali registri saran- valore determ

no attivati.

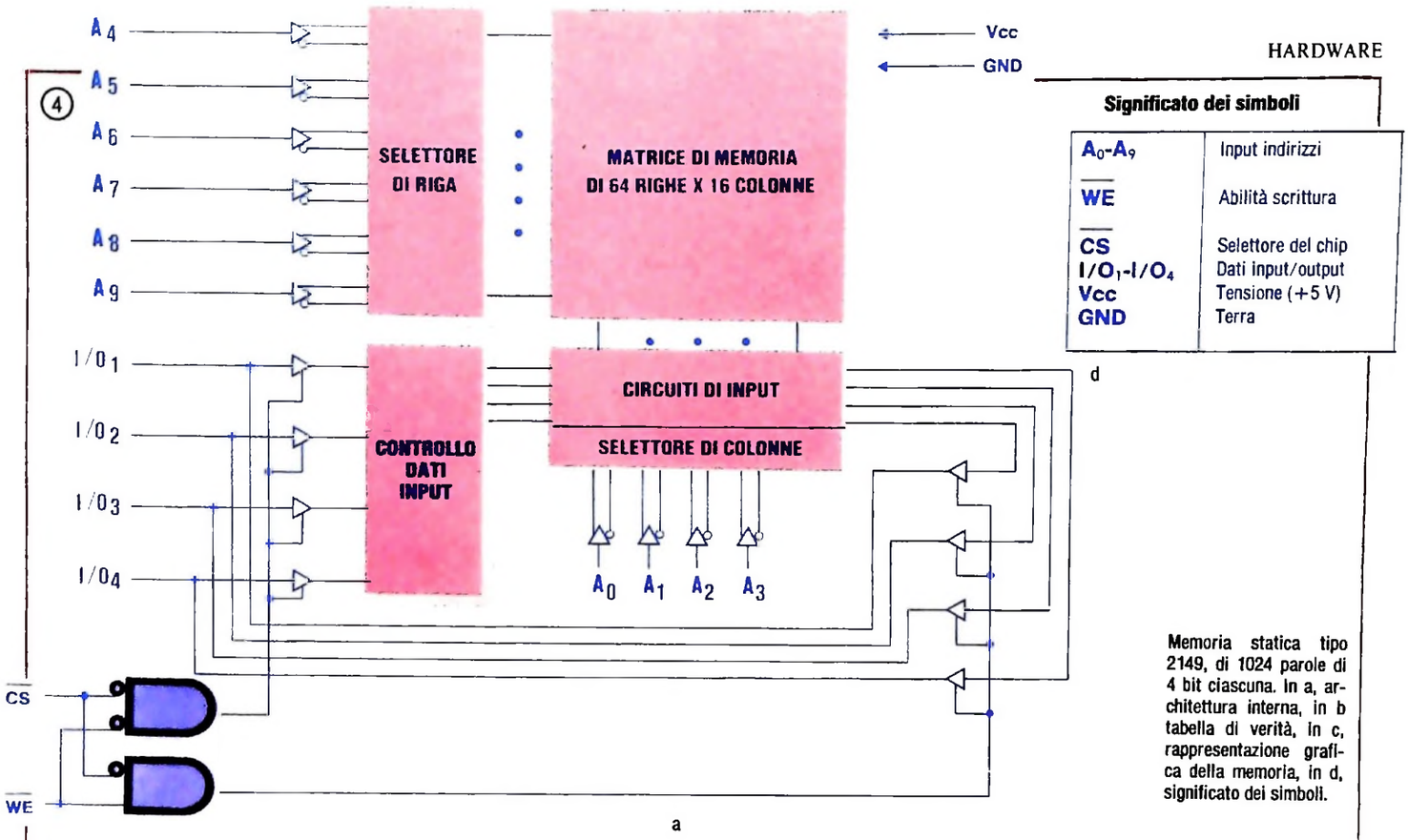
M.A.R. (Re gistro di Indirizzo di Memoria)

istro esterno al microprocessore che analizzia- È il primo reg gistro è necessario solo quando ci troviamo in mo. Questo re icroprocessori con un unico BUS per dati e in- presenza di m rima di trattare il funzionamento del M.A.R. è dirizzi. Però, p opportuno, anche per quanto riguarda il microprocessore in generale, fare un breve accenno alla forma in cui sono realizzati, in genere, i circuiti di memoria. Prendiamo, per esempio, una memoria di 1024x4 cioè 1024 parole di 4 bit, tipo 2149 (figura 4). Analizziamo prima di tutto l'architettura della memoria e vediamo che la collocazione delle celle interne è fatta a forma di matrice, cioè come una specie di reticolo in cui in ogni punto di incrocio delle righe e colonne c'è una cella.

Con le linee A4 →A9 vengono selezionate le linee orizzontali, e quindi con 5 linee abbiamo a disposizione 64 righe (2^5= 64), mentre con A0 →A3 abbiamo a disposizione 16 colonne (2^4 = 16), per un totale di 1024 celle di memoria.

L'entrata $\overline{CS}$ abilita, quando è ATTIVA NEGATIVA, il circuito; questo presuppone che, prima di tutto, l'indirizzo sia valido e, inoltre, sia proprio in quel blocco di memoria entro cui vogliamo operare.

Il segnale $\overline{WE}$ vuol dire, quando $\overline{WE}$ = 0, che si vuol scrivere in memoria, mentre con $\overline{WE}$ = 1 si vuole leggere; ciò è facilmente comprensibile se analizziamo attentamente il circuito della figura 4 prendendo in esame i piedini $1/0_1 \rightarrow 1/0_4$ con i rispettivi BUFFER a tre stati. Vediamo adesso il funzionamento del circuito integrato, tenendo presente anche il fattore tempo. Prendiamo in considerazione dapprima la fase di scrittura (figura 5). In questa fase, come pure in quella di lettura, la prima cosa che deve essere pronta e stabile è l'indirizzo, dopo un certo tempo che in genere varia da 10 a 50 ns (nanosecondi) bisogna abilitare il $\overline{CS}$ (ATTIVO NEGATIVO) del circuito o dei circuiti in cui si vuole operare. Que-



a

Significato dei simboli

d

A_0-A_9	Input indirizzi
$\overline{WE}$	Abilità scrittura
$\overline{CS}$	Selettore del chip
I/O_1-I/O_4	Dati input/output
Vcc	Tensione (+5 V)
GND	Terra

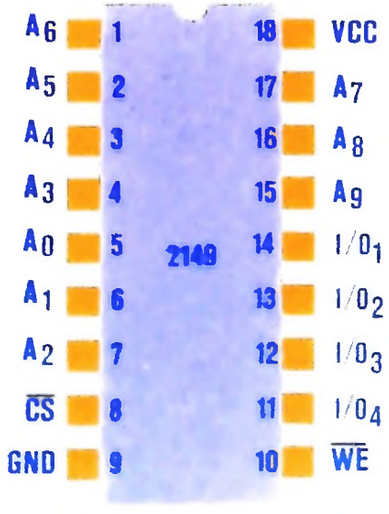
Memoria statica tipo 2149, di 1024 parole di 4 bit ciascuna. In a, architettura interna, in b tabella di verità, in c, rappresentazione grafica della memoria, in d, significato dei simboli.

Tabella di verità

$\overline{CS}$	$\overline{WE}$	MODO	I/O
H	X	Non selezionato	Alto Z
L	L	Scrive	DIN
L	H	Legge	DOUT

b

c



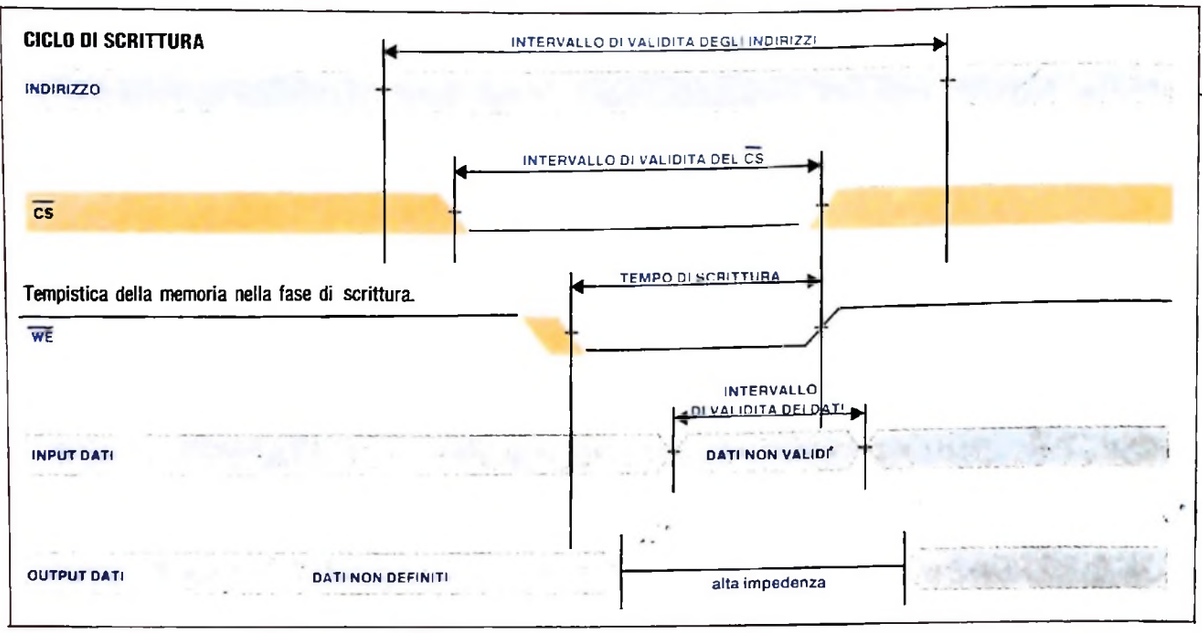
st'ultima operazione si realizza, quando è necessario, con un altro circuito chiamato DECODIFICATORE che vedremo più avanti. Dopo questa abilitazione, se siamo in fase di scrittura, bisogna abilitare il $\overline{WE}$ in forma ATTIVA NEGATIVA, però bisogna fare attenzione che non è nella transizione negativa che i dati vengono scritti in memoria, ma nella transizione positiva, per cui i dati devono essere stabili al limite solo in quest'ultima transizione. Per la lettura, invece, (figura 6) il comando $\overline{WE}$=1 e i dati della cella prescelta saranno pronti e giusti solo dopo un tempo chiamato "tempo di accesso" (vedi la tabella sui tempi di lettura).

Dopo questa constatazione, risulta chiaro che non è possibile usare lo stesso BUS contemporaneamente sia per i dati sia per gli indirizzi. Il registro M.A.R. risolve appunto questo problema in quanto in esso viene immagazzinato l'indirizzo, che è il primo ad essere generato e comparire nel BUS, quando abbiamo il comando LM=1 e alla transizione positiva del CLOCK, naturalmente.

Dopo questa operazione, l'indirizzo presente nel registro M.A.R. passa immediatamente alla memoria e agli elementi esterni e vi resta fisso fino al termine dell'operazione. Una volta liberato il BUS dall'indirizzo, in esso possono circolare i dati sia per la lettura come per la scrittura.

Dopo questa premessa, il funzionamento del registro M.A.R. risulta semplice: con il comando LM=1, sincrono, alla transizione positiva del CLOCK, il contenuto del BUS viene immagazzinato nel Registro e passa, asincronamente, alla MEM e al DEC (Decodificatore), dopo di che il BUS è libero per altri usi.

⑤

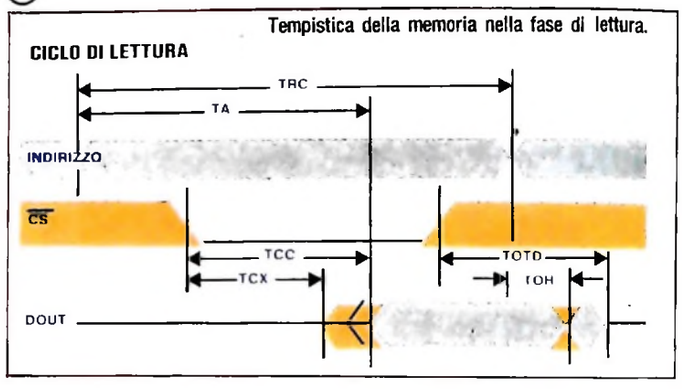


Tempistica della memoria nella fase di scrittura.

Tempi di lettura (in nanosecondi) nelle varie operazioni

PARAMETRO	SIMBOLO	2149 MIN	2149 MAX	UNITÀ
Tempo ciclo di lettura	TRC	35		NS
Tempo di accesso	TA		35	NS
Selezione del chip e output valido	TCO		15	NS
Selezione del chip e output attivo	TCX	5		NS
Terzo stato degli output dalla deselezione	TOTD		10	NS
Tempo di stasi degli output dal cambio di indirizzo	TOH	5		NS

⑥



Tempistica della memoria nella fase di lettura.

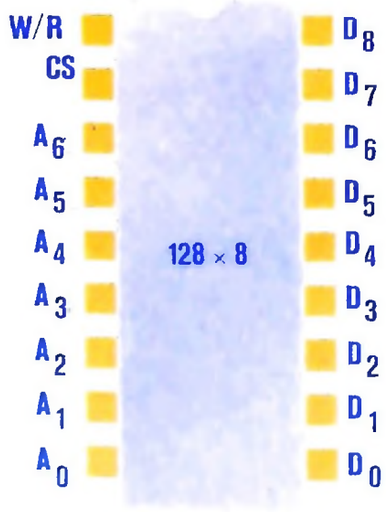
DEC (Decodificatore)

Questo circuito è, a volte, necessario in base al tipo e alla quantità di elementi esterni necessari da gestire. Il decodificatore ha la funzione di interpretare l'indirizzo e di abilitare i $\overline{CS}$ giusti. Nel nostro caso, essendo il sistema molto semplice ed embrionale, l'indirizzo che entra provoca l'abilitazione di una delle due uscite CS0 o CS1, con CS0 = 1 si abilita il registro "O" mentre con CS1 = 1 si abilita la memoria. Il segnale VMA che entra nel registro DEC è un segnale di controllo emesso dal CON del microprocessore per assicurare ai circuiti esterni che l'informazione presente nel BUS, o nel M.A.R. nel nostro caso, è un indirizzo sempre valido.

MEM (Memoria)

La memoria comprende 128 parole di 8 bit (figura 7). Come in ogni memoria, da un lato abbiamo le entrate degli indirizzi dall'altra l'uscita o l'entrata dei dati, i comandi di controllo sono il CS ATTIVO POSITIVO dove entra il segnale di controllo CS1 e il W/$\bar{R}$ che ha due significati: con W/$\bar{R}$=1 la memoria viene scritta, cioè il dato presente nel BUS passa alla memoria; con W/$\bar{R}$=0 il dato dalla memoria, localizzato dall'indirizzo, passa al BUS e da qui al microprocessore.

⑦



La configurazione della memoria 128x8 (cioè di 128 parole da 8 bit ciascuna) dell'UAMICRO I.

ENUNCIATI

Una breve escursione nel mondo della logica ci fa scoprire un'altra applicazione dell'algebra di Boole.

Il linguaggio è sicuramente un fenomeno di grande complessità. È possibile esaminarlo a livelli molto diversi, da quello morfologico, della struttura delle parole, a quello sintattico, della concatenazione delle "parole" e delle frasi, da quello semantico, del significato, a quello estetico, e via dicendo.
Gli studiosi di logica si sono interessati, fondamentalmente, a un aspetto limitato, ma significativo, del funzionamento del linguaggio: il rapporto tra la forma e la verità del discorso. Monsieur de La Palisse è famoso — a ragione o a torto — per aver legato il nome a frasi come "un uomo è un uomo" o "se viene, viene", che brillano per la loro scarsissima ricchezza di informazione (almeno se prese nel significato letterale). Ora, una frase come "se viene, viene" può essere trasformata in migliaia di frasi analoghe, parimenti lapalissiane: "se mangia, mangia", "se corre, corre", e via dicendo. Che queste frasi siano tutte banalmente vere, dipende non dal significato dei verbi "venire", "mangiare", "correre", ma dalla sola "forma" delle frasi, che è sempre del tipo "se A, A".
Questa forma può essere ogni volta riempita con un contenuto diverso, dando alla variabile A ogni possibile valore: la frase rimane sempre vera. Ecco un caso, banale certo, di rapporto fra la forma di una frase e la sua verità.

A	~	A
1		0
0		1

A	B	A ∧ B
1	1	1
1	0	0
0	1	0
0	0	0

A	B	A ∨ B
1	1	1
1	0	1
0	1	1
0	0	0

A	B	A → B
1	1	1
1	0	0
0	1	1
0	0	1

A	B	A ↔ B
1	1	1
1	0	0
0	1	0
0	0	1

Una tabella di verità è una rappresentazione grafica dei valori di verità degli enunciati, che facilita il calcolo dei valori di verità degli enunciati composti. Sono date qui le tabelle di verità dei cinque connettivi logici fondamentali: nelle prime colonne sono date le combinazioni dei possibili valori degli enunciati di partenza e nell'ultima vengono forniti i corrispondenti valori dell'enunciato risultante.

$$((A \vee B) \wedge (C \rightarrow B) \leftrightarrow (A \wedge B)$$

A	B	C	A ∨ B	C → B	(A ∨ B) ∧ (C → B)	A ∧ B	((A ∨ B) ∧ (C → B)) ↔ (A ∧ B)
1	1	1	1	1	1	1	1
1	1	0	1	1	1	0	1
1	0	1	1	0	0	0	1
1	0	0	1	1	1	0	0
0	1	1	1	1	1	0	0
0	1	0	1	1	1	0	0
0	0	1	0	0	0	0	1
0	0	0	0	1	0	0	1

$$(A \wedge (B \vee C)) \leftrightarrow (A \wedge B) \vee (A \wedge C)$$

A	B	C	B ∨ C	A ∧ (B ∨ C)	A ∧ B	A ∧ C	(A ∧ B) ∨ (A ∧ C)	(A ∧ (B ∨ C)) ↔ ((A ∧ B) ∨ (A ∧ C))
1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	0	1	1
1	0	1	1	1	0	1	1	1
1	0	0	0	0	0	0	0	1
0	1	1	1	0	0	0	0	1
0	1	0	1	0	0	0	0	1
0	0	1	1	0	0	0	0	1
0	0	0	0	0	0	0	0	1

Fra tutti gli enunciati, ve ne sono alcuni "a statuto speciale": le tautologie e le contraddizioni. Le tautologie sono enunciati sempre veri, per tutte le combinazioni di valori degli enunciati componenti; le contraddizioni sono enunciati sempre falsi. Le tautologie sono "verità logiche": vediamo a destra la tabella di verità della tautologia che esprime la validità della distributività della congiunzione rispetto alla disgiunzione.
Per valutare un enunciato complesso si seguono le regole dell'algebra: si risolvono prima le parentesi più interne, poi le più esterne. La tabella di verità a sinistra dà la valutazione dell'enunciato $((A \vee B) \wedge (C \rightarrow B)) \leftrightarrow (A \wedge B)$: si valutano prima $(A \vee B)$ e $(C \rightarrow B)$, poi $(A \vee B) \wedge (C \rightarrow B)$, poi $(A \wedge B)$ e si combinano gli ultimi due risultati per valutare tutto l'enunciato complesso.

Enunciati

Fermiamo la nostra attenzione al livello delle frasi. Qui non ci interessa il significato, quindi parleremo piuttosto di "enunciati", intendendo riferirci con questo vocabolo, come fanno normalmente i logici, alla struttura sintattica solamente e, più precisamente, alla struttura sintattica delle frasi dichiarative (non ci occupiamo, cioè, delle frasi esclamative o delle interrogative), come "Giovanni è partito per Roma", "Luigi sta leggendo un libro" e simili.
Un enunciato (una frase dichiarativa) può essere o vero o falso: "Cristoforo Colombo raggiunse l'America" è un enunciato vero, "Napoleone Bonaparte fu incoronato zar di tutte le Russie" è un enunciato falso. Non ci importa sapere come facciamo a determinare se un enunciato elementare è vero o falso: questa può essere una questione empirica, da risolvere volta per volta con metodi diversi. Ora ci interessa vedere che cosa succede quando combiniamo, in un discorso, più enunciati elementari.
"Giovanni è andato al cinema" e "Luigi ha mangiato un gelato" sono enunciati elementari, che possiamo combinare, in un discorso di tutti i giorni, con una congiunzione "e": "Giovanni è andato al cinema e Luigi ha mangiato un gelato". Questo enunciato complesso è a sua volta vero o falso, e la sua verità o falsità dipende dalla verità o falsità dei due enunciati semplici che lo compongono. In particolare, è vero se (e solo se) sono veri ambedue gli enunciati componenti. E questo vale per tutte le situazioni analoghe: se A e B sono due enunciati qualunque, l'enunciato "A e B" è vero se e solo se sono veri sia A sia B.
Nel linguaggio di tutti i giorni, usiamo molte particelle diverse per collegare due frasi: "ma", "o", "se ... allora", "tuttavia", "sebbene", e via dicendo. Dal punto di vista logico, possiamo vedere tutte queste particelle come "funzioni" che associano ai valori di verità (vero o falso) di due enunciati un terzo valore di verità, quello dell'enunciato composto, realizzato grazie al loro intervento. Ritorniamo alla nostra "e". L'enunciato A può avere solo i due valori "falso" e "vero", che possiamo indicare con F e V o, meglio ancora, con 0 e 1. Lo stesso dicasi per B. Si possono presentare in definitiva solo quattro casi:

A è 1, B è 1
A è 1, B è 0
A è 0, B è 1
A è 0, B è 0

Che cosa fa la particella "e"? Ci dà un enunciato "A e B" che vale 1 (è vero) se e solo se A è 1 e B è 1; negli altri casi, ci dà un enunciato che vale 0 (è falso). In notazione funzionale: e (1, 1) = 1; e (1, 0) = e (0, 1) = e (0, 0) = 0. In questo senso si dice solitamente che la "e" è una *funzione di verità*.

Funzioni di verità

Una funzione di verità è una funzione che prende come argomenti valori di verità e dà come valori ancora valori di verità.

Le più semplici, fra le funzioni di verità, sono quelle a un solo argomento. Ce ne possono essere solo quattro: una che lascia inalterati i valori che prende in ingresso, una che per ciascun valore di verità ci dà il suo opposto, una che dà sempre in uscita il valore 1, una che dà sempre valore 0. La più interessante è la seconda, che corrisponde sostanzialmente alla particella "non" del linguaggio quotidiano: se A è vero, "non A" è falso, e viceversa.

Le altre tre funzioni non hanno un corrispettivo nel linguaggio ordinario.

Le funzioni di verità a due argomenti, come la "e", possono essere, in tutto, sedici (potete provare a contare le combinazioni possibili, se non ci credete) e alcune hanno un corrispettivo nel linguaggio ordinario. La "o" per esempio, è la funzione di verità che dà valore vero se e solo se almeno uno dei due enunciati che collega è vero: "Vado al cinema o torno a casa" è vero se è vero "vado al cinema" oppure è vero "torno a casa"; è vero anche se sono veri ambedue gli enunciati componenti. In altre parole, "A o B" vale 1 se almeno uno, fra A e B, vale 1.

Nel linguaggio ordinario, esiste un senso più ristretto della congiunzione "o", che esclude la verità contemporanea dei due enunciati componenti: è la cosiddetta "o esclusiva", che usiamo in frasi come "è vivo o è morto", "vincerà o perderà". In questi casi l'enunciato complesso è vero se e solo se uno solo dei due enunciati componenti è vero, è falso in ogni altro caso.

I logici normalmente prendono in considerazione due altre costruzioni: "se ... allora" e "se e solo se". La seconda non crea difficoltà: è la funzione di verità che dà all'enunciato complesso valore di verità 1 quando i due enunciati componenti hanno lo stesso valore di verità, e valore 0 quando i due

Alcuni teoremi della logica

$((A \wedge A) \leftrightarrow A)$	(legge di idempotenza di $\wedge$)
$((A \vee A) \leftrightarrow A)$	(legge di idempotenza di $\vee$)
$((A \wedge B) \leftrightarrow (B \wedge A))$	(legge di commutatività di $\wedge$)
$((A \vee B) \leftrightarrow (B \vee A))$	(legge di commutatività di $\vee$)
$((A \wedge (B \wedge C)) \leftrightarrow ((A \wedge B) \wedge C))$	(legge di associatività di $\wedge$)
$((A \vee (B \vee C)) \leftrightarrow ((A \vee B) \vee C))$	(legge di associatività di $\vee$)
$((A \wedge (A \vee B)) \leftrightarrow A)$	(legge di assorbimento di $\wedge$)
$((A \vee (A \wedge B)) \leftrightarrow A)$	(legge di assorbimento di $\vee$)
$((A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C)))$	(legge di distributività di $\wedge$ su $\vee$)
$((A \vee (B \wedge C)) \leftrightarrow ((A \vee B) \wedge (A \vee C)))$	(legge di distributività di $\vee$ su $\wedge$)
$\neg (A \wedge \sim A)$	(legge di non-contraddizione)
$(A \vee \sim A)$	(legge del terzo escluso)
$(A \leftrightarrow A)$	(legge di identità)
$((A \wedge B) \leftrightarrow \neg (\neg A \vee \sim B))$	(I legge di De Morgan)
$((A \vee B) \leftrightarrow \neg (\neg A \vee \sim B))$	(II legge di De Morgan)
$(\neg (A \wedge B) \leftrightarrow (\neg A \vee \sim B))$	(III legge di De Morgan)
$((A \rightarrow B) \rightarrow (\neg B \rightarrow \sim A))$	(I legge di contrapposizione)
$((A \rightarrow (B \rightarrow C)) \rightarrow ((A \wedge B) \rightarrow C))$	(legge di importazione)
$(((A \wedge B) \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C)))$	(legge di esportazione)
$((A \rightarrow B) \leftrightarrow (\neg A \vee B))$	(legge di Filone Megarico)
$((A \rightarrow B) \leftrightarrow \neg (A \wedge \neg B))$	(legge di Crisippo)
$((A \rightarrow C) \rightarrow ((A \wedge B) \rightarrow C)$	(legge di rafforzamento dell'antecedente)
$((A \rightarrow B) \rightarrow (A \rightarrow (B \vee C))$	(legge di indebolimento del conseguente)
$((A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C))$	(legge di scambio dell'antecedente)

Le tabelle di verità offrono un metodo meccanico per decidere se un enunciato è una tautologia o meno. Si dice che l'insieme delle tautologie è un insieme decidibile. Nella tabella sono elencate alcune tautologie importanti: come si può vedere, una tautologia non è necessariamente una banalità come "se corre, corre". In alcuni casi bisogna ricorrere alla tabella di verità, per stabilire se un enunciato è effettivamente una tautologia o no.

Algebra booleana di insiemi	Calcolo proposizionale
U (insieme universale)	**V** (vero)
∅ (insieme vuoto)	**F** (falso)
a, b, c, ... (insiemi, sottoinsiemi, elementi)	***p, q, r, ...*** (proposizioni)
$a \cup b$ (unione: la totalità di ***a*** e ***b***)	$p \vee q$ (disgiunzione: o ***p*** solo o ***q*** solo o entrambi sono veri)
$a \cap b$ (intersezione: elementi comuni ad ***a*** e ***b***)	$p \wedge q$ (congiunzione: ***p*** e ***q*** sono entrambi veri)
$a = b$ (identità: ***a*** e **b** sono lo stesso insieme)	$p \leftrightarrow q$ (equivalenza: **p** è vero se e solo se è vero ***q***)
$\bar{a}$ (complemento: tutti gli elementi di **U** che non sono in ***a***)	$\sim p$ (negazione: ***p*** è falso)
$a \varepsilon b$ (inclusione: ***a*** è un sottoinsieme di ***b***)	$p \rightarrow q$ (implicazione: se ***p*** è vero, ***q*** è vero)

I diagrammi di Venn, che abbiamo visto in un articolo precedente, sono utili anche nella rappresentazione degli enunciati: l'interno di un cerchio è, per esempio, l'insieme delle condizioni che rendono vero A, l'esterno è l'insieme delle condizioni che rendono vero non A, cioè che rendono A falso. La tabella qui accanto presenta le corrispondenze fra l'algebra degli insiemi e quella degli enunciati, visualizzabili facilmente con i diagrammi di Venn.

enunciati componenti hanno valori diversi. "T è un triangolo rettangolo se e solo se T è un triangolo con un angolo retto" è vero se "T è un triangolo rettangolo" e "T è un triangolo con un angolo retto" sono ambedue veri o ambedue falsi; è falso in ogni altro caso.

"Se ... allora" è una costruzione più complessa. È la costruzione del periodo ipotetico, e in genere siamo portati a pensare che fra l'antecedente e il conseguente di un periodo ipotetico debba esistere un rapporto di significato. Qui tuttavia, vogliamo prescindere dal significato e far riferimento solo alla sintassi, pertanto dobbiamo staccarci un po' dall'uso quotidiano.

Per un logico, l'enunciato "Se A, allora B" è vero se B è vero oppure A è falso, in altre parole:

per A = 1, B = 1, "se A, allora B" = 1
per A = 1, B = 0, "se A, allora B" = 0
per A = 0, B = 1, "se A, allora B" = 1
per A = 0, B = 0, "se A, allora B" = 1.

Allora, una frase come "Se Roma è la capitale dell'Olanda, allora Cristoforo Colombo ha scoperto l'America" è vera, perché l'antecedente è falso e il conseguente è vero. Come interpretazione del linguaggio comune può risultare un po' più forzata, o almeno strana, ma non crea difficoltà in un contesto rigoroso come quello matematico (per il quale è nata la logica moderna).

Ancora un'algebra di Boole

A partire da enunciati semplici, si possono costruire enunciati sempre più complessi, collegando ripetutamente gli enunciati mediante particelle logiche come "o", "e", "se ... allora", e via dicendo. I logici usano normalmente i simboli $\sim$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, per indicare rispettivamente "non", "e", "o", "se ... allora", "se e solo se" e dimostrano che queste cinque funzioni di verità sono sufficienti per costruire qualunque altra funzione di verità. (Le funzioni a più di due argomenti si possono ottenere come combinazioni di varie funzioni a due argomenti solamente.)

In realtà, questi cinque "connettivi", come vengono anche chiamati, sono *più* che sufficienti: ne bastano infatti due soli, per esempio $\sim$ e $\vee$. Gli altri tre connettivi possono infatti essere ottenuti come combinazione di questi due.

Consideriamo ora tre connettivi: $\sim$, $\wedge$ e $\vee$. Sappiamo già che possiamo interpretarli come funzioni (o operazioni) che fanno corrispondere valori di verità a valori di verità. Ricordate l'algebra booleana a due soli elementi vista nel capitolo precedente? Bene, provate ora a sostituire il simbolo · con il connettivo $\wedge$, il + con il connettivo $\vee$ e interpretare il $\sim$ come complemento: funziona tutto perfettamente. Se interpretiamo 1 e 0 come valori di verità, la somma come disgiunzione, il prodotto come congiunzione e il complemento come negazione, scopriamo una nuova "algebra di Boole", l'algebra degli enunciati.

Lezione 18

I sottoprogrammi

Nello sviluppo top down di un programma, il modo di procedere è quello di costruire innanzitutto l'intelaiatura generale, lasciando eventuali "buchi" di funzioni da svolgere e poi procedere allo stesso modo per ciascuna delle parti mancanti, fino all'ottenimento del programma completo.
Questa struttura logica del modo di procedere può essere calata nell'organizzazione fisica di un programma mediante l'uso di SOTTOPROGRAMMI.
Un sottoprogramma è una porzione di programma che svolge un compito ben determinato e che può venire richiamato da un programma principale (o da un altro sottoprogramma), svolgendo il compito richiesto e quindi restituendo il controllo a chi lo aveva richiamato.
Facciamo un esempio. Si voglia costruire un programma che applichi il teorema di Pitagora e calcoli l'ipotenusa di un triangolo rettangolo, dati i cateti. Una prima approssimazione del programma può essere:

- leggi i valori dei cateti
- controlla che siano corretti
- IF corretti THEN
  - •• calcola ipotenusa

  ELSE
  - •• segnala errore

Invece di pensare alle varie indicazioni fornite come a "promemoria" di ciò che deve essere sviluppato, possiamo vederle come vere e proprie istruzioni di un linguaggio di programmazione di altissimo livello, non disponibile sui calcolatori, ma che noi ci possiamo mettere a disposizione; con questo atteggiamento, il programma precedente può essere organizzato in una forma simile a quella del linguaggio Pascal, come segue:

```
PROGRAM Pitagora;
 PROCEDURE controllalati;
    BEGIN
        •••
        END;
 PROCEDURE calcolaipotenusa;
    BEGIN
        •••
    END;
 BEGIN programma principale
  read(C1,C2)
  controllalati
  IF corretti THEN
     calcolaipotenusa
   ELSE
     writeln("lati errati")
END.
```

Completata questa diciottesima lezione del Corso di Programmazione e BASIC, siete in grado di eseguire gli esercizi

MERGET. DO
MERGEP. BA

contenuti nella cassetta "5 esercizi di programmazione", lato B.

I titoli seguiti dal suffisso DO corrispondono a testi, quelli seguiti da BA a programmi in BASIC.
Caricateli secondo le modalità che avete appreso.

Proviamo a leggerlo.
Il programma principale effettua una lettura dei due valori dei cateti nelle variabili C1 e C2 (istruzione PASCAL "read(C1,C2)"); quindi viene invocato un sottoprogramma di nome "controllalati".
Quando, nel corso dell'esecuzione del programma, tale invocazione viene incontrata, l'esecuzione del programma viene sospesa, il controllo passa al sottoprogramma richiesto, cioè alla parte indicata con "PROCEDURE controllalati", la quale viene eseguita.
Al termine dell'esecuzione il controllo viene restituito al programma principale, proprio nel punto in cui la procedura era stata invocata.
L'esecuzione prosegue allora con il controllo della variabile "corretti", che evidentemente deve essere stata posta a "vero" o a "falso" dal sottoprogramma appena chiamato, a indicare che i lati erano o no corretti: se corretti è uguale a vero, viene invocato, con la stessa modalità, il sottoprogramma "calcolaipotenusa", altrimenti viene visualizzato un messaggio d'errore con l'istruzione Pascal "writeln" ("scrivi una riga").
Le procedure "controllalati" e "calcolaipotenusa" sono diventate, dal punto di vista del programma principale, vere e proprie istruzioni, specificamente orientate al tipo di problema, e il nostro problema è stato decomposto in una gerarchia di sottoproblemi, che a loro volta hanno generato una gerarchia di livelli di linguaggio di programmazione.
I due sottoprogrammi, nell'esempio, non sono stati ancora dettagliati: si tratta per il momento di scatole vuote da riempire; questa tecnica è tipica dello sviluppo top down; infatti, nel risolvere il problema di livello più alto, diamo per scontate le soluzioni dei sottoproblemi e ci troviamo ora a dover risolvere non più un problema di una certa complessità, ma due sottoproblemi molto più semplici.

I sottoprogrammi in BASIC

Il linguaggio BASIC mette a disposizione una coppia di istruzioni per realizzare sottoprogrammi:

- GOSUB. che trasferisce il controllo all'istruzione d'inizio di un sottoprogramma, e che quindi sarà collocata nel programma chiamante;
- RETURN, che restituisce il controllo al programma chiamante, all'istruzione successiva alla chiamata e che quindi deve essere l'ultima istruzione del sottoprogramma.

Vediamo come realizzare il programma dell'esempio precedente:

```
10 ' Calcolo ipotenusa con teorema di Pitagora
20 INPUT "Cateti";C1,C2
30 ' Controllo correttezza dei dati
40 GOSUB 100
50 ' Se lati corretti C=1, altrimenti C=0
60 ' Se C=1 calcola ipotenusa
70 IF C=1 THEN GOSUB 200 ELSE PRINT "Lati errati"
80 END
```

Il programma costruito invoca due sottoprogrammi che si trovano ai numeri di linea 100 e 200 rispettivamente.
Tali programmi non sono stati ancora costruiti; possiamo costruirli lasciandoli ancora vuoti:

```
100 ' Sottoprogramma controllo cateti
180 RETURN
200 ' Sottoprogramma calcolo ipotenusa
250 RETURN
```

Ora il programma, pur non essendo completo, è eseguibile e se noi riempiamo i due sottoprogrammi di istruzioni fittizie che ci indicano le operazioni fatte, possiamo già verificarne il corretto funzionamento della struttura generale; ad esempio, con le istruzioni:

```
110 LET C=1
120 PRINT "Valori corretti"
```

e

```
210 PRINT "Ipotenusa calcolata"
```

possiamo verificare con l'esecuzione il corretto funzionamento:

```
Ok
run
Cateti? 5,4
Valori corretti
Ipotenusa calcolata
Ok
```

Ora possiamo preoccuparci dei soli sottoprogrammi.
Completiamo il primo come segue:

```
110 LET C=1
120 IF C1<=0 OR C2<=0 THEN LET C=0
```

e chiediamone la lista parziale con il comando:

```
LIST 100-180
100 ' Sottoprogramma controllo cateti
110 LET C=1
120 IF C1<=0 OR C2<=0 THEN LET C=0
180 RETURN
Ok
```

Completiamo il secondo come segue:

```
210 PRINT "Ipotenusa: ";SQR(C1^2+C2^2)
```

ed eseguiamo il programma completo:

```
run
Cateti? 12,9
Ipotenusa:  15
Ok
```

I sottoprogrammi che abbiamo costruito sono evidentemente troppo piccoli e troppo semplici per giustificare l'uso di un tale strumento, e dobbiamo senza dubbio considerare il programma sviluppato come un esempio didattico: in un caso come questo avremmo potuto tranquillamente inserire direttamente i controlli e il calcolo nel programma principale.
Un'ultima osservazione: i sottoprogrammi BASIC sono parte integrante del programma BASIC in cui si trovano e quindi ne condividono i numeri di linea, le variabili, e così via. Infatti, i nostri sottoprogrammi avevano la visibilità di C1 e C2, così come il programma principale aveva la visibilità di C.
È bene fare attenzione a questi aspetti, per evitare di commettere l'errore di usare in un sottoprogramma una variabile che è già usata nel programma principale con un altro significato.
Ancora, è bene che un sottoprogramma sia accessibile solo da una istruzione iniziale e si possa abbandonare solo con un RETURN finale, cioè abbia un solo punto d'ingresso e un solo punto di uscita (infatti si potrebbe pensare di inserire più di un RETURN): ciò è più coerente con la visione di sottoprogramma come istruzione di alto livello e permette di tenere sotto controllo la complessità dei programmi.

Cosa abbiamo imparato

Nella presente lezione abbiamo visto:

- lo sviluppo top down mediante sottoprogrammi
- il concetto di procedura in Pascal
- l'esecuzione di programmi parzialmente specificati, per verificare la correttezza del programma mano a mano che questo viene sviluppato
- un nuovo modo di usare il comando LIST, specificando i numeri di linea di inizio e di fine della lista richiesta

SUL LINGUAGGIO DELLE IMMAGINI

I vari tipi di interfaccia di comunicazione uomo-calcolatore.

Nei precedenti articoli si sono visti gli sviluppi della ricerca nel campo della Intelligenza Artificiale e si è constatata l'esistenza di un problema di comunicazione delle informazioni e dei concetti tra l'uomo e le macchine. Si è anche rilevato che l'informazione di tipo visuale ha caratteristiche che la rendono più facilmente comprensibile ed assimilabile di quella testuale, e che questo tipo di comunicazione è in grado di trasmettere, anche attraverso una sola immagine, un numero di contenuti molto alto. Un esempio può essere visto nella figura 1 (pagina seguente), in cui si nota come, nonostante in ogni quadro siano presentati pochi punti, la sequenza di questi comunichi l'idea di movimento. La comunicazione degli stessi contenuti avrebbe richiesto, tramite il linguaggio scritto, l'uso di diverse frasi per specificare sia ciò che avviene, sia il contesto dell'avvenimento.
La disponibilità di display grafici e di tecniche di visualizzazione e organizzazione delle immagini (esposte in un precedente articolo), oltre a fornire uno strumento con cui comunicare le informazioni, permette di diminuirne la complessità o, alternativamente, aumentare la quantità trasmessa di queste come unità. La disponibilità, inoltre, di un linguaggio grafico con cui rappresentare delle entità potrebbe ridurre la necessità di imparare i diversi linguaggi, spesso inconsistenti fra loro, che sono caratteristici di applicazioni diverse. In questo articolo verranno discussi i vari tipi di interfaccia di comunicazione con il calcolatore esistenti e ne verrà analizzato in particolare uno di recente sviluppo.

Tipi di interfaccia

Le interfacce, ambiente di comunicazione fra l'uomo e il computer, possono essere classificate in tre gruppi:

a) le interfacce a icone (immagini);
b) le interfacce a comandi;
c) le interfacce a menù.

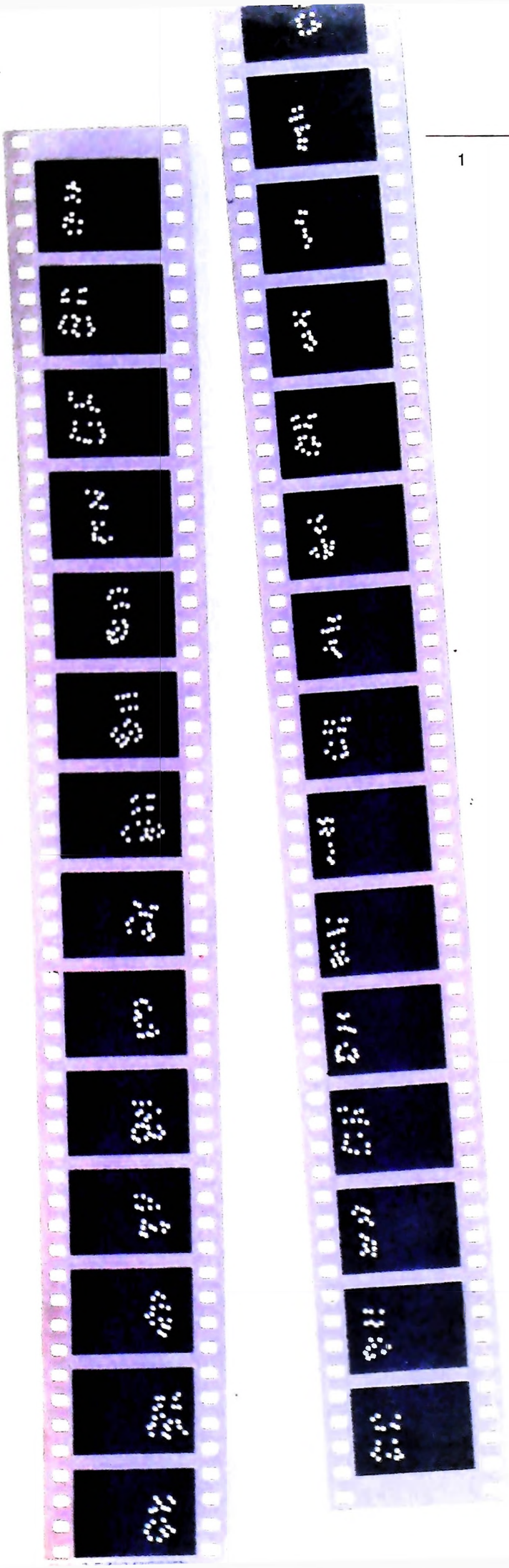
Si va diffondendo, presso gli istituti di ricerca e i centri universitari, una sempre maggiore tendenza all'utilizzo di comunicazione ad icone (immagini) a causa della loro maggiore naturalezza di interazione e facilità di uso rispetto alle comuni interfacce (a comandi e a menù) in cui la trasmissione di messaggi avviene in modo testuale. Questo tipo di interfaccia risulta più semplice da utilizzare in quanto richiede all'utente, per operare una scelta, di riconoscere l'oggetto scelto e di puntare a questo, il che risulta essere un compito più semplice di quello normalmente richiesto, cioè ricordare un comando e scriverlo. Il riconoscimento di un'icona può anche essere soltanto approssimato: infatti, è rilevante la distinzione dell'icona con il significato ricercato all'interno di un gruppo di altre con significati diversi, piuttosto che la sua comprensione completa. Nelle comuni interfacce a comandi, per ottenere un risultato, è necessario scrivere il comando corrispondente, di cui bisogna ricordare la sintassi precisa ed una serie di opzioni che, dovendo essere per necessità compatte, tendono ad essere criptiche e innaturali. Un esempio può essere tratto dall'utilizzo di un sistema operativo, UNIX, che, sebbene molto evoluto, manca di un'interfaccia colloquiale (ad icone). Nel linguaggio di questo sistema operativo, per ottenere la stampa (in un formato determinato) di una lettera che si trova su un file chiamato *Messaggio*, bisogna scrivere:

```
nroff -me -Tr32 Messaggio | lpr
```

e se, per esempio, si volesse fare altro durante la stampa, il comando andrebbe modificato con una aggiunta finale:

```
nroff -me -Tr32 Messaggio | lpr &
```

Come si vede, il comando tende ad essere incomprensibile per un non iniziato ed esiste una difficoltà effettiva nell'imparare ogni comando con tutte le sue opzioni e la sintassi associata. Le interfacce a menù, in cui all'utente viene richiesto di scegliere l'operazione da effettuare all'interno di un gruppo di sentenze (frasi o comandi) che gli vengono presentate, non comportano la necessità di imparare comandi specifici, come in quelle del tipo precedente; impongono tuttavia all'utente di imparare una nomenclatura con un preciso significato (diversi per ogni applicazione) e tendono a diventare pesanti nel continuo trasferirsi da un menù all'altro. Come si può dunque notare, l'uso di icone per la realizzazione di in-



1

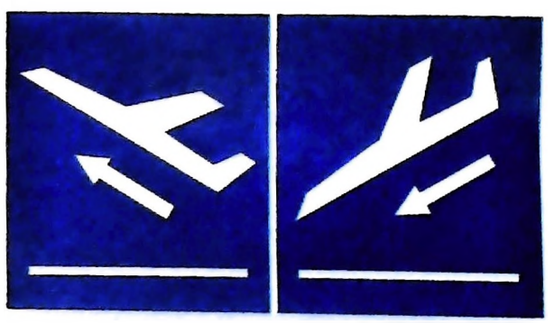
Disegno ricavato da una serie di fotogrammi ripresi mentre una coppia di ballerini danzava al buio. Nella sequenza i ballerini sono individuati da 12 sorgenti luminose attaccate alle spalle, ai gomiti, ai polsi, alle anche, alle ginocchia e alle caviglie. Si ottiene un'efficace 'descrizione' del movimento.

terfacce presenta una maggiore naturalezza unita ad una maggiore semplicità, come dimostrato dall'entusiasmo che un utente inesperto dimostra se messo di fronte ad un calcolatore con interfaccia di questo tipo.

Le interfacce a icone: interpretazione e classificazione

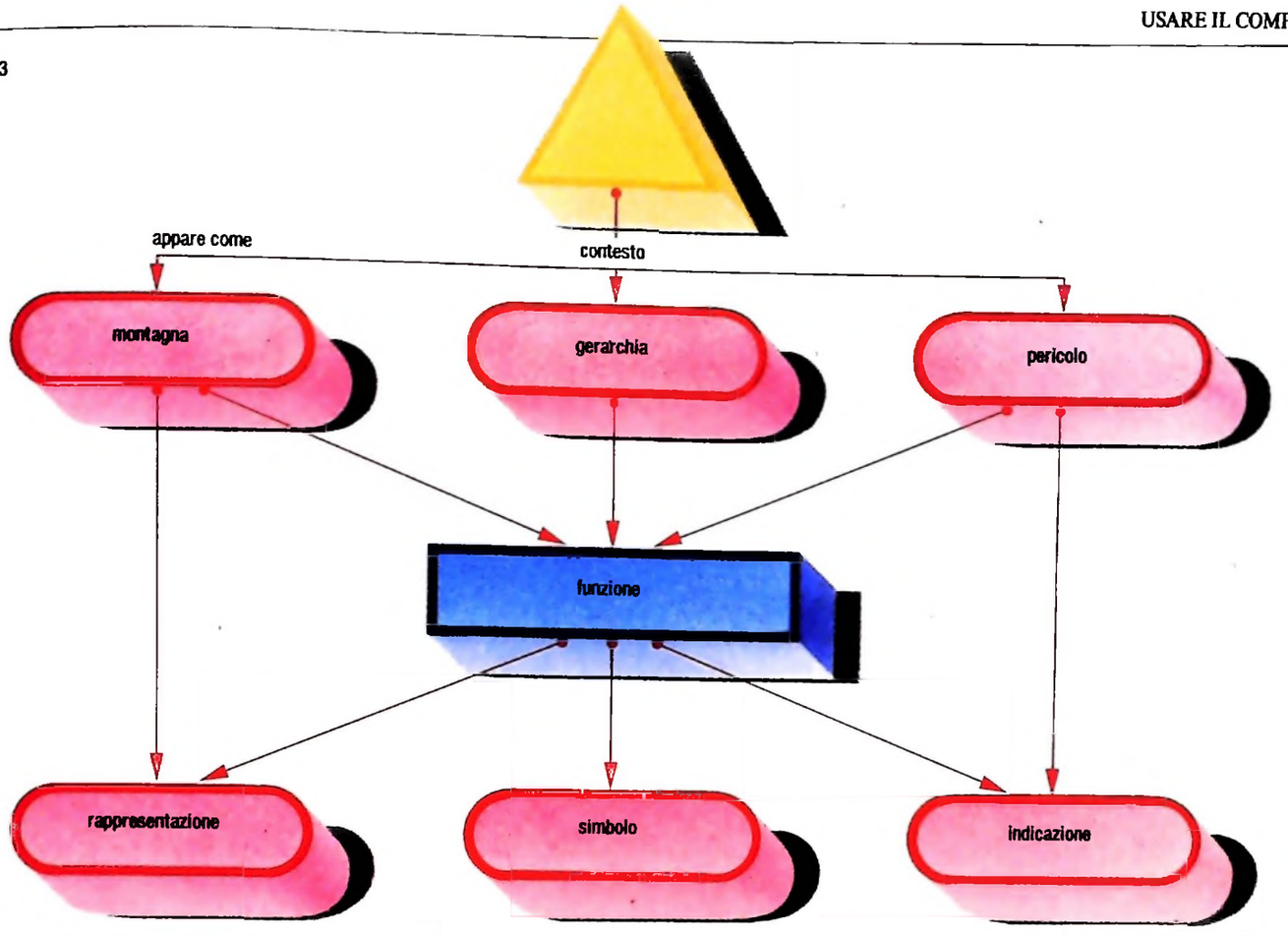
Per una corretta interpretazione di una icona è necessaria, in generale, l'identificazione di tre componenti:
1) la rappresentazione (codifica) della immagine;
2) l'associazione di questa con una didascalia;
3) il contesto in cui essa appare.
La *codifica* dell'immagine deve mettere in grado l'utente di riconoscere, senza fatica, l'oggetto che la icona rappresenta o comprendere il significato che la stessa porta con sé.
La *didascalia* permette di riconoscere ed identificare la icona all'interno di un gruppo di icone dello stesso tipo, e fornisce dunque una informazione ridondante in certi casi.
Il *contesto* fornisce gli elementi, eventualmente mancanti, necessari per togliere ambiguità nel riconoscimento della icona.
A partire da questi punti, una icona può essere generalmente interpretata ma, anche se l'uso delle immagini è considerato un mezzo per superare barriere culturali e di linguaggio, vi possono essere delle difficoltà. Il disegno e la definizione di un'icona è infatti influenzato sia da ciò che il disegnatore percepisce come rilevante, sia da altri fattori come la sua educazione, la sua cultura e l'ambiente in cui vive che possono risultare non universali e portare dunque ad una incom-

2



Due icone che segnalavano la partenza e l'arrivo dei voli in un aeroporto USA. Bisognò sostituirle (v. fig. 4) per la sensazione di 'pericolo' che comunicavano.

3



La stessa icona (quindi lo stesso segno) assume significati diversi a seconda del contesto in cui si trova (montagna, gerarchia o pericolo).

contesto	montagna	gerarchia	pericolo
funzione	rappresentazione	simbolo	indicazione

4

Due icone più rassicuranti collocate al posto di quelle in figura 2: a sinistra gli arrivi (simboleggiati dall'aereo), a destra le partenze (il viaggiatore).

prensione della icona stessa o del messaggio associato. Oltre a questo, una icona può essere male interpretata a causa di messaggi subliminali, percepiti a livello inconscio, che essa stessa può trasportare.

Un esempio di ciò può essere visto nella figura 2, che rappresenta alcune icone usate come segnali che si trovavano negli aeroporti americani per indicare l'arrivo e la partenza dei voli. Queste icone dovettero essere sostituite con quella di figura 4, a causa della sensazione di disastro aereo più che di sicuro atterraggio che esse suscitavano nei passeggeri. Altri problemi possono insorgere se il disegnatore inserisce nella immagine troppi particolari che rendono difficile e ritardano la comprensione della stessa da parte dell'utente oltre a essere di difficile rappresentazione su schermi a bassa risoluzio-

5

Il calice di vetro, simbolo immediato di fragilità, in uso sugli imballaggi di merci delicate (a esempio elettroniche).

6

Un simbolo meno immediato, quello del pericolo radioattivo, che è collocato dovunque esistano sorgenti radioattive.

ne. Il fatto inoltre che una icona, diversamente dalle parole, debba essere creata ex novo aggiunge altre difficoltà, non esistendo un vocabolario definito cui fare riferimento per selezionare immagini con un significato preciso. Un primo aiuto alla soluzione di problemi di questo tipo potrebbe essere dato dalla classificazione delle icone in classi distinte a seconda del significato che esse intendono comunicare: ciò comporta la comprensione della funzione che le immagini devono avere.

Arnheim, nel suo articolo *Visual Thinking* e seguenti, riconosce tre funzioni principali di un'immagine: *di rappresentazione, di simbolo* e *di indicazione*; in altre parole è il contesto che permette di assegnare un'immagine ad una di queste tre classi (figura 3).

Una immagine può avere funzione di *rappresentazione* quando essa intende essere un esempio di un oggetto, non una copia ma una esemplificazione delle sue caratteristiche rilevanti; di *simbolo* se cerca di comunicare un concetto e non l'oggetto stesso rappresentato (come può essere visto in figura 5 in cui si cerca di comunicare il concetto di fragilità); di *indicazione* se la sua funzione è quella di indicare l'oggetto stesso e non le sue caratteristiche. Icone con funzione rappresentativa sono semplici da definire, e ciò porta a differenti rappresentazioni con lo stesso significato, e molto semplici da interpretare.

Icone di tipo simbolico sono altrettanto facili da comprendere, ma di più difficile disegno in quanto devono focalizzarsi non sugli oggetti stessi ma sul significato che questi devono comunicare. Le icone di tipo indicativo, un esempio può essere visto in figura 6, sono invece di comprensione non immediata e il loro significato deve essere imparato in quanto non si appoggiano per la comunicazione di un messaggio su oggetti predefiniti.

Una caratteristica particolare di questo tipo di icone è che, nonostante le difficoltà iniziali di comprensione, il loro significato, una volta appreso, risulterebbe essere più persistente nel tempo.

La definizione di una icona

Alla appropriata definizione di un'icona concorrono almeno tre elementi interdipendenti. È necessario stabilire prima di tutto e senza alcuna ambiguità il messaggio che deve essere comunicato. Oltre a ciò, deve essere preso in considerazione il contesto in cui la icona deve apparire poiché questo influenza (figura 3) la sua interpretazione.

La realizzazione effettiva della immagine deve considerare il fatto che essa verrà usata sia su display a colori che in bianco e nero e su schermi a bassa e ad alta risoluzione.

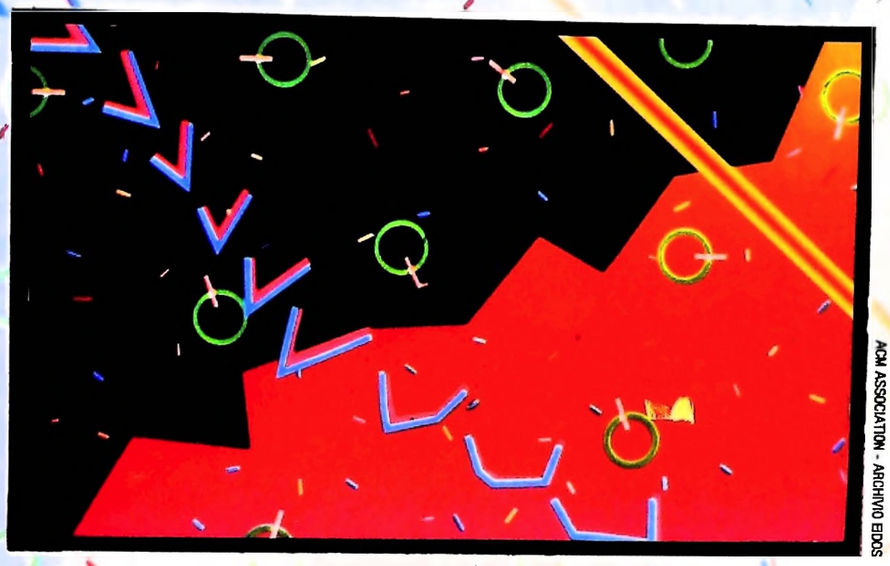
Per superare questi problemi è necessario fare in modo che il colore non comunichi nessun significato particolare e, come nelle più recenti realizzazioni di cartine topografiche, fare in modo che le linee del disegno si intersechino a quarantacinque gradi così da evitare l'effetto scala tipico dei display a bassa risoluzione. Le tecniche di organizzazione delle immagini tratte dal campo pubblicitario possono inoltre aiutare quando si definiscano gruppi di icone. Infine, è necessario verificare quanto risulta comprensibile il messaggio che si vuole comunicare. Se si sviluppano diverse icone con lo stesso significato, il criterio di scelta fra di queste dovrebbe procedere attraverso la eliminazione della soluzione peggiore e non la scelta di quella migliore, in quanto, per un utente, è più rilevante la difficoltà di comprensione di un messaggio e non la sua eventuale comprensione perfetta.

Conclusioni

L'uso di icone sembra dunque rappresentare un miglioramento nella capacità di visualizzazione e di facile comprensione di messaggi nella comunicazione uomo-macchina; in un prossimo capitolo, vedremo come questa tecnica, mutuata dagli istituti di ricerca, sia migrata a campi commerciali e daremo uno sguardo a macchine che fanno un uso intensivo di immagini (icone) nelle loro interfacce.

LE TRASLAZIONI E LE TRASFORMAZIONI DI SCALA

Le possibilità di manipolare la posizione e le dimensioni di una figura.



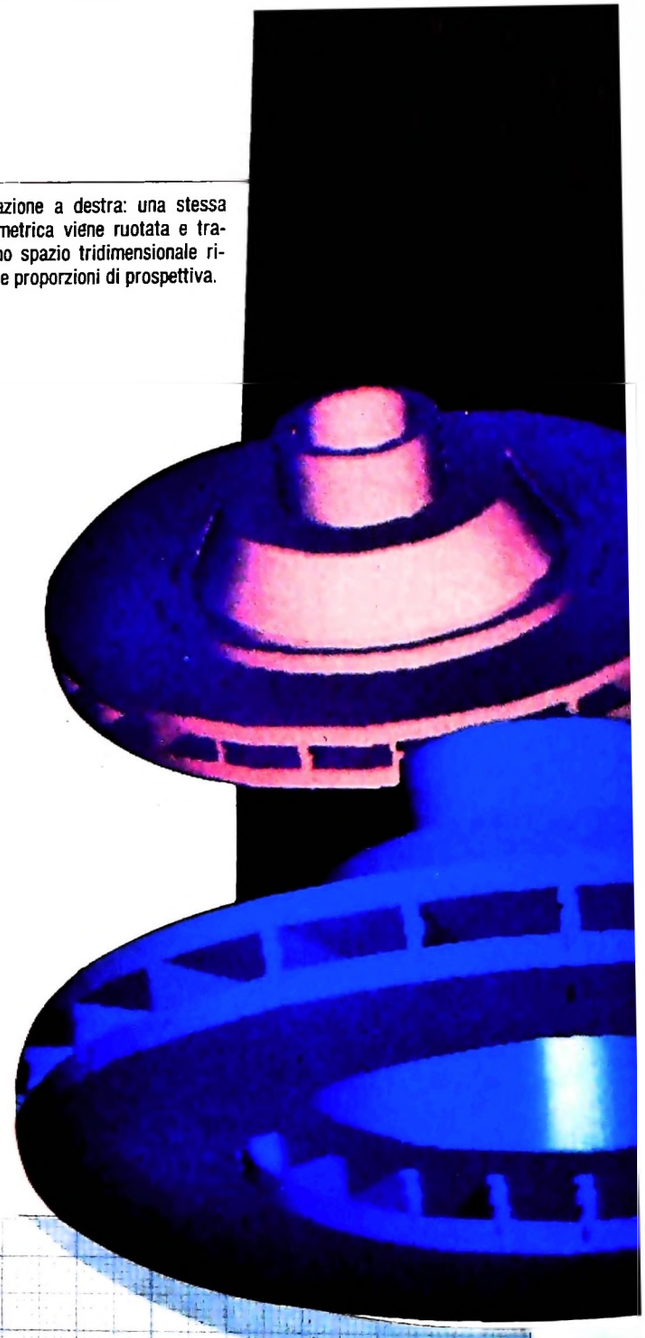
ACM ASSOCIATION - ARCHIVIO EIDOS

Una stessa forma geometrica viene ruotata e traslata in uno spazio bidimensionale.

Introduzione

Le trasformazioni geometriche costituiscono un importante e vasto capitolo della geometria.
Felix Klein, un famoso matematico tedesco, in un discorso tenuto ad Erlangen nel 1872, propose di classificare i diversi rami della geometria secondo le classi di trasformazioni considerate, in modo da identificare quali siano le proprietà più generali che rimangono invariate applicando una trasformazione qualsiasi. L'impulso che ricevette la ricerca in geometria da questo approccio diviene oggi particolarmente fruttifero per la possibilità di utilizzarne i risultati nella elaborazione automatica di figure e nella generazione di loro immagini trasformate.
Desideriamo perciò aprire una serie di articoli aventi l'obiettivo di fornire quelle competenze sufficienti per capire i problemi legati alle trasformazioni geometriche e a fornire quei concetti base che permettano di risolvere i relativi problemi di programmazione.
In questo e in alcuni articoli futuri, intendiamo quindi analizzare le trasformazioni più tipicamente interessanti per la computergrafica. Ci occuperemo subito delle trasformazioni che descrivono i moti rigidi, le quali lasciano invariate le grandezze degli oggetti considerati e, successivamente, delle trasformazioni proiettive, nate secoli fa dall'esigenza di rappresentare oggetti solidi (tridimensionali) sul piano (bidimensionale) e studiate in particolare da Leonardo da Vinci e da Albrecht Dürer; infine, daremo uno sguardo a trasformazioni forse di scarsa utilità pratica, ma dotate di grande fascino: le anamorfosi, giochi grafici usati fin dal XVI secolo per realizzare immagini misteriose che potevano venire decodificate e riconosciute con accorgimenti particolari.
Una necessità che si presenta quasi subito all'utente di un sistema grafico computerizzato è quella di poter manipolare le posizioni e le dimensioni delle figure. Per esempio, può servire ingrandire una figura per evidenziarne un dettaglio, ridurne le dimensioni per visualizzarla interamente sullo schermo, ruotarla nel piano attorno ad un punto o ad un asse prefissato, oppure trasportarla da una zona all'altra dello schermo. Queste prestazioni sono ottenibili su M10 via software. È possibile infatti scrivere una serie di subroutine che ricevono in entrata le informazioni relative ad una certa immagine e forniscono in uscita l'immagine trasformata. Due aspetti particolarmente rilevanti vanno subito evidenziati nella trat-

tazione delle trasformazioni:
1) una trasformazione è rappresentabile mediante una singola entità matematica, che viene chiamata matrice (si veda il riquadro dal titolo "Vettori e matrici");
2) due o più trasformazioni (per esempio una traslazione, una trasformazione di scala e una rotazione) possono essere "concatenate" e dare origine ad un'unica trasformazione che genera gli stessi effetti prodotti dall'applicazione in sequenza delle originarie.
In questa lezione ci occuperemo delle traslazioni e delle trasformazioni di scala nel piano, ossia in due sole dimensioni, mentre affronteremo le rotazioni nella lezione successiva.



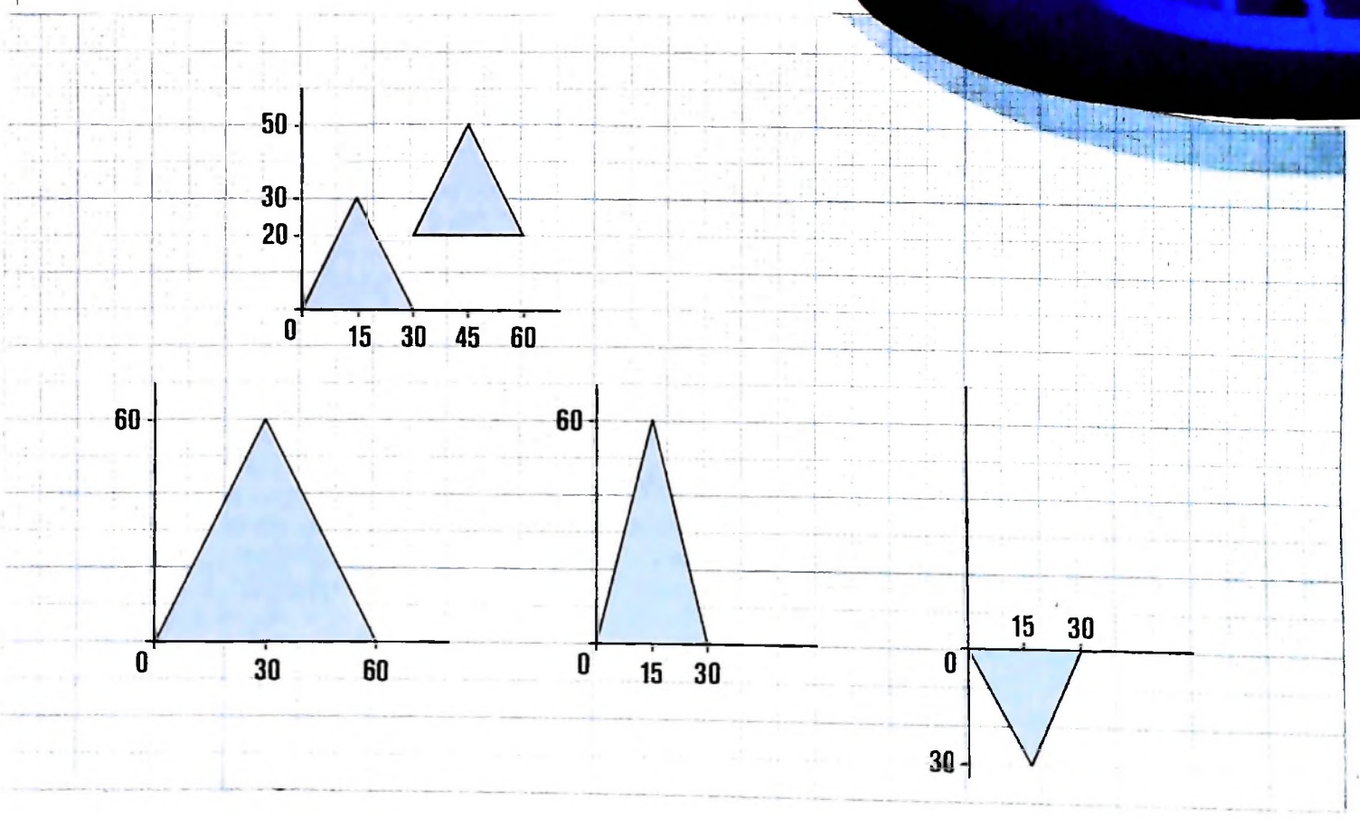
Nell'illustrazione a destra: una stessa forma geometrica viene ruotata e traslata in uno spazio tridimensionale rispettando le proporzioni di prospettiva.

Traslazione

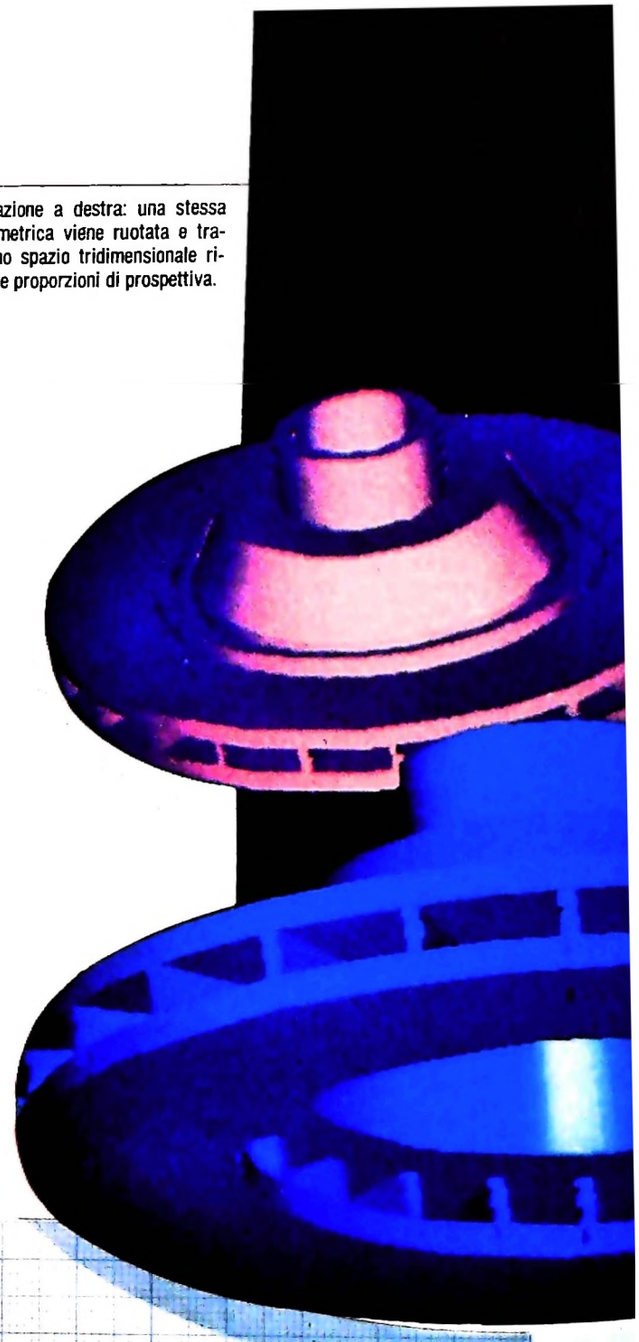
Per traslazione si intende lo spostamento di una figura da una zona all'altra dello schermo. La traslazione avviene applicando ad ogni punto della figura le formule di traslazione appropriate. Se la figura è costituita da segmenti di linee rette, allora è sufficiente applicare le formule di traslazione ai soli vertici di ogni segmento. Le formule sono:

$$x' = x + Tx \quad ; \quad y' = y + Ty$$

in cui x' e y' sono le nuove coordinate del punto trasformato; x e y sono le coordinate del punto di partenza, mentre Tx e Ty rappresentano di quanto si dovrà spostare, rispettivamente lungo i due assi di riferimento, il punto considerato. Per esempio, prendiamo il triangolo definito dai tre vertici (0,0), (30,0), (15,30). Trasliamolo di 30 unità verso destra e di 20



tazione delle trasformazioni:
1) una trasformazione è rappresentabile mediante una singola entità matematica, che viene chiamata matrice (si veda il riquadro dal titolo "Vettori e matrici");
2) due o più trasformazioni (per esempio una traslazione, una trasformazione di scala e una rotazione) possono essere "concatenate" e dare origine ad un'unica trasformazione che genera gli stessi effetti prodotti dall'applicazione in sequenza delle originarie.
In questa lezione ci occuperemo delle traslazioni e delle trasformazioni di scala nel piano, ossia in due sole dimensioni, mentre affronteremo le rotazioni nella lezione successiva.



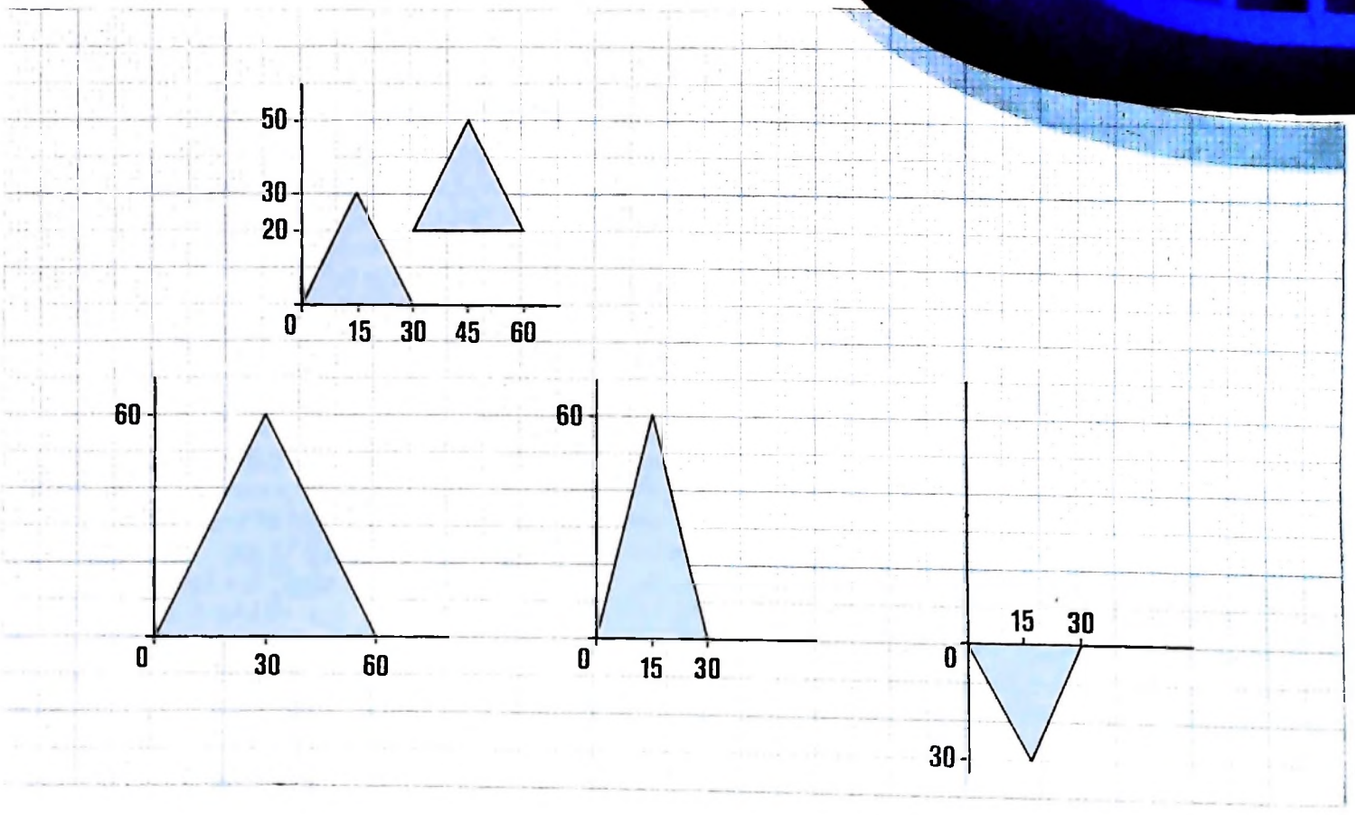
Nell'illustrazione a destra: una stessa forma geometrica viene ruotata e traslata in uno spazio tridimensionale rispettando le proporzioni di prospettiva.

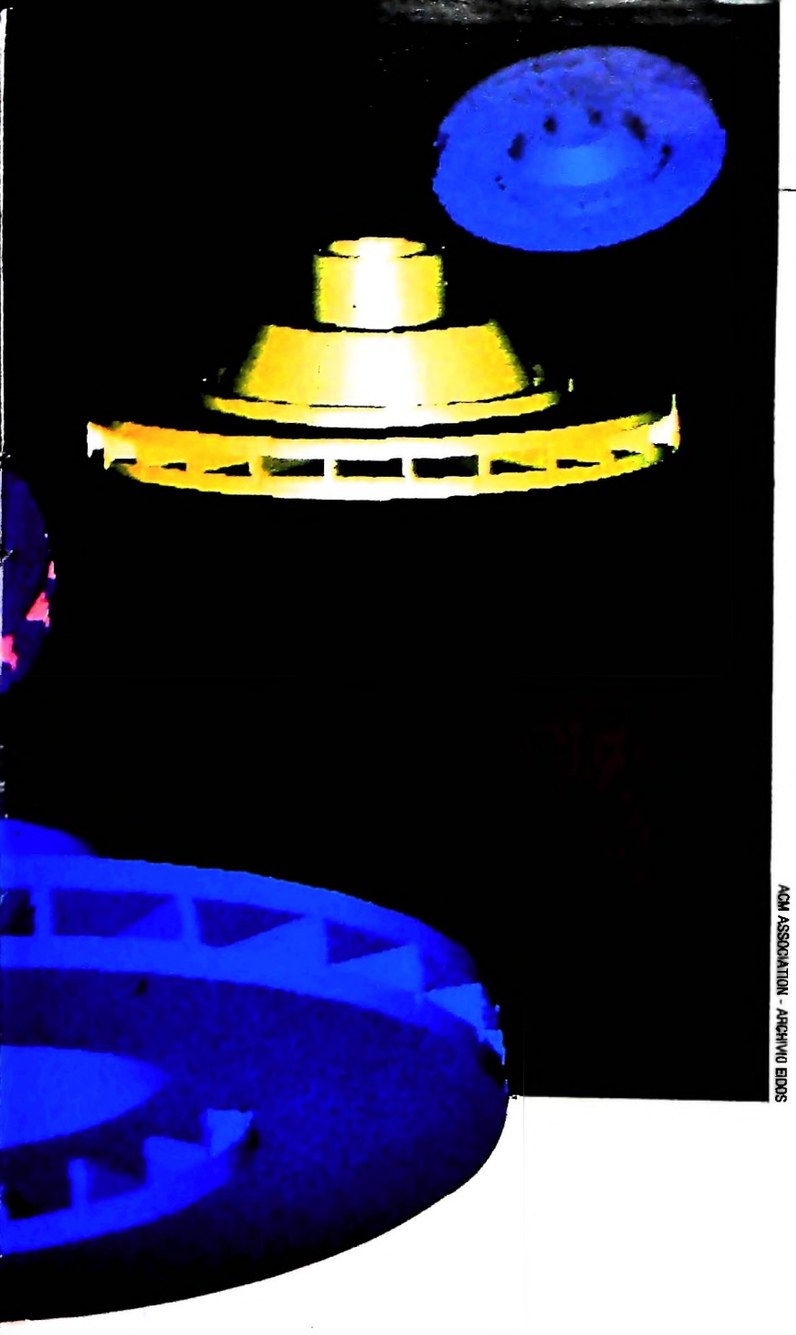
Traslazione

Per traslazione si intende lo spostamento di una figura da una zona all'altra dello schermo. La traslazione avviene applicando ad ogni punto della figura le formule di traslazione appropriate. Se la figura è costituita da segmenti di linee rette, allora è sufficiente applicare le formule di traslazione ai soli vertici di ogni segmento. Le formule sono:

$$x' = x + Tx \quad ; \quad y' = y + Ty$$

in cui x' e y' sono le nuove coordinate del punto trasformato; x e y sono le coordinate del punto di partenza, mentre Tx e Ty rappresentano di quanto si dovrà spostare, rispettivamente lungo i due assi di riferimento, il punto considerato. Per esempio, prendiamo il triangolo definito dai tre vertici (0,0), (30,0), (15,30). Trasliamolo di 30 unità verso destra e di 20





AGM ASSOCIATION - ARCHIVIO EIDOS

verso l'alto: varranno allora Tx = 30 e Ty = 20. I nuovi vertici saranno (30,20), (60,20), (45,50).

Trasformazione di scala

Effettuare una trasformazione di scala (*scaling*, in inglese) significa modificare le dimensioni di una figura: rimpicciolendola, ingrandendola o deformandola. Le formule di trasformazione sono:

$$x' = x \cdot Sx \quad ; \quad y' = y \cdot Sy$$

Possono essere utilizzate per vari scopi: per esempio, se si pone Sx = Sy = 2, per raddoppiare le dimensioni della figura; se invece Sx e Sy sono diversi, si ottiene una distorsione dell'immagine, che potrà essere "allargata" o "stirata" lungo le direzioni dei due assi; per ottenere una immagine speculare è invece necessario utilizzare valori negativi di Sx o di Sy. Vediamo qualche esempio: il triangolo visto precedentemente (0,0), (30,0), (15,30), se sottoposto alle trasformazioni di scala con Sx = Sy = 2, diviene il triangolo (0,0), (60,0), (30,60); se invece Sx = 1 e Sy = 2, diventa (0,0), (30,0), (15,60); se infine Sx = 1 e Sy = –1, diventa (0,0), (30,0), (15,–30).

Rappresentazione matriciale

Tutte le trasformazioni bidimensionali possono essere espresse in forma di matrice 3x3 (si veda il riquadro).
La generica traslazione di un vettore [x y] in un nuovo vettore [x' y'] è rappresentata dalla seguente espressione:

$$[x'\ y'\ 1] = [x\ y\ 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix}$$

La matrice 3x3 rappresenta la traslazione e tutte le traslazioni sono rappresentate da matrici di questo tipo, ossia con la stessa disposizione degli elementi. L'aggiunta del terzo elemento unitario al vettore [x y] è per renderlo conformabile con la matrice 3x3, ossia per poter effettuare la moltiplicazione "righe per colonne".
La generica trasformazione di scala di un vettore [x y] in un nuovo vettore [x' y'] è rappresentata dall'espressione:

$$[x'\ y'\ 1] = [x\ y\ 1] \quad \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La matrice 3x3 rappresenta una trasformazione di scala, e tutte le trasformazioni di scala sono rappresentate da matrici di questo tipo, ossia con la stessa disposizione degli elementi. Risolvendo queste due equazioni matriciali, ovvero effettuando i prodotti righe per colonne, si verifica che sono esattamente equivalenti alle formule di trasformazione viste in precedenza.

Concatenazione di matrici di trasformazione

La formulazione matriciale è particolarmente adatta per concatenare fra loro vari tipi di trasformazioni. Per esempio, consideriamo la seguente sequenza: vogliamo scalare un vettore con Sx = Sy = 3 e poi traslarlo con Tx = 20 e Ty = 10. Avremo:

$$[x'\ y'\ 1] = [x\ y\ 1] \cdot \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[x''\ y''\ 1] = [x'\ y'\ 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 20 & 10 & 1 \end{bmatrix}$$

Vettori e matrici

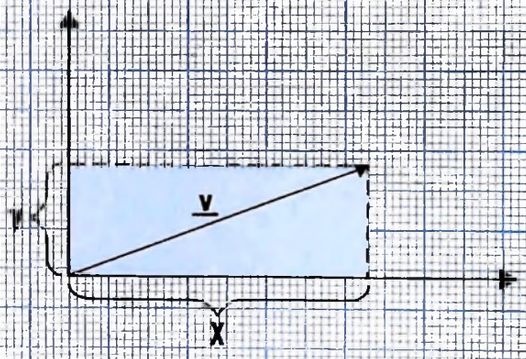
Chi si interessa di computergrafica, deve avere nel suo bagaglio di cognizioni matematiche i concetti fondamentali di "vettore" e di "matrice", con le relative operazioni algebriche ad essi applicabili. Vediamo allora rapidamente come possiamo introdurli.

Vettori

Un generico vettore viene rappresentato graficamente da una freccia che parte nell'origine e formalmente dalla notazione v=[x y], in cui x e y rappresentano le sue proiezioni ortogonali sugli assi coordinati, chiamate "componenti del vettore".

Abbiamo visto come tutti i pixel dello schermo siano identificabili con una coppia di punti (x, y) dette coordinate. Il formalismo vettoriale è quindi qualcosa di simile. Ossia, ogni pixel viene ancora indicato dagli stessi due numeri, ma essi vengono ora intesi come la lunghezza delle componenti ortogonali del vettore spiccato dall'origine con punta nel pixel considerato. Per sommare algebricamente due vettori, si sommano fra loro le rispettive componenti:

$$[x\ y] + [z\ w] = [x+z\ \ y+w]$$



Scomposizione di un vettore v sulle sue componenti ortogonali x e y

Matrici

In molti problemi ci si imbatte in tabelle di numeri o di funzioni dei quali interessa evidenziare che sono dotati di un doppio ordinamento. Per rappresentare tabelle di questo tipo, dette "matrici", si usa il seguente simbolismo:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & \\ \vdots & \vdots & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

I numeri o le funzioni a_{ij} che compaiono nella matrice vengono chiamati "elementi" della tabella. Gli indici i e j dell'elemento a_{ij} indicano rispettivamente la riga e la colonna alle quali appartiene l'elemento. Chiaramente i assume i valori compresi fra 1 ed m, mentre j quelli fra 1 ed n. Se m=n, ossia se il numero di righe è uguale al numero di colonne, si dice che la matrice è "quadrata di ordine n", altrimenti si dice che è di tipo m righe per n colonne, abbreviato in m x n.

In particolare allora risulta che un vettore è una matrice di tipo 1 x 2. Due matrici si dicono "conformabili" se il numero di colonne della prima è uguale al numero di righe della seconda, ossia se sono del tipo m x n ed n x p. In particolare le matrici quadrate dello stesso ordine sono tutte fra loro conformabili. Noi ci serviremo soltanto di matrici quadrate e di vettori. La somma algebrica fra due matrici è la matrice che ha per elementi la somma algebrica dei corrispondenti elementi delle matrici addende.

Prodotto fra matrici

È possibile soltanto fra matrici conformabili. Il prodotto fra matrici è detto "righe per colonne", dato che si effettua la somma dei prodotti degli elementi di ogni riga per il rispettivo elemento di una colonna della seconda.

Vediamone più chiaramente un esempio utilizzando due matrici quadrate di ordine 2:

$$\begin{bmatrix} 2 & 4 \\ 3 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2x1 + 4x2 & 2x0 + 4x1 \\ 3x1 + 2x2 & 3x0 + 2x1 \end{bmatrix} = \begin{bmatrix} 10 & 4 \\ 7 & 2 \end{bmatrix}$$

È essenziale l'ordine con cui si moltiplicano le matrici, infatti, se A e B sono per esempio due matrici quadrate, in generale vale che A B è diverso da B A. Questo fatto si esprime dicendo che il prodotto righe per colonne di due matrici non è commutativo. Provate, per verifica, a cambiare l'ordine delle due matrici dell'esempio precedente e a calcolarne il prodotto.

Il risultato [x' y' 1] costituisce semplicemente un passaggio intermedio dell'intera sequenza di trasformazione; possiamo di conseguenza eliminarlo sostituendo la prima equazione nella seconda:

$$[x''\ y''\ 1] = [x\ y\ 1] \cdot \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 20 & 10 & 1 \end{bmatrix}$$

Le due matrici 3x3 sono completamente indipendenti dal vettore [x y 1] da trasformare, per cui possiamo semplificare la situazione eseguendo subito la moltiplicazione fra le due matrici 3x3, e ottenere così l'equazione complessiva:

$$[x''\ y''\ 1] = [x\ y\ 1] \cdot \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 20 & 10 & 1 \end{bmatrix}$$

Quindi il prodotto di due matrici rappresenta la loro "concatenazione" e il risultato dell'intera trasformazione non cambia se, invece della sequenza delle due, viene applicata la matrice di trasformazione globale ottenuta.

Prossimamente vedremo anche alcune applicazioni pratiche di queste teorie.

PERSONAL COMPUTER OLIVETTI M10

L'UFFICIO DA VIAGGIO

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

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

olivetti

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

NOME COGNOME

VIA N

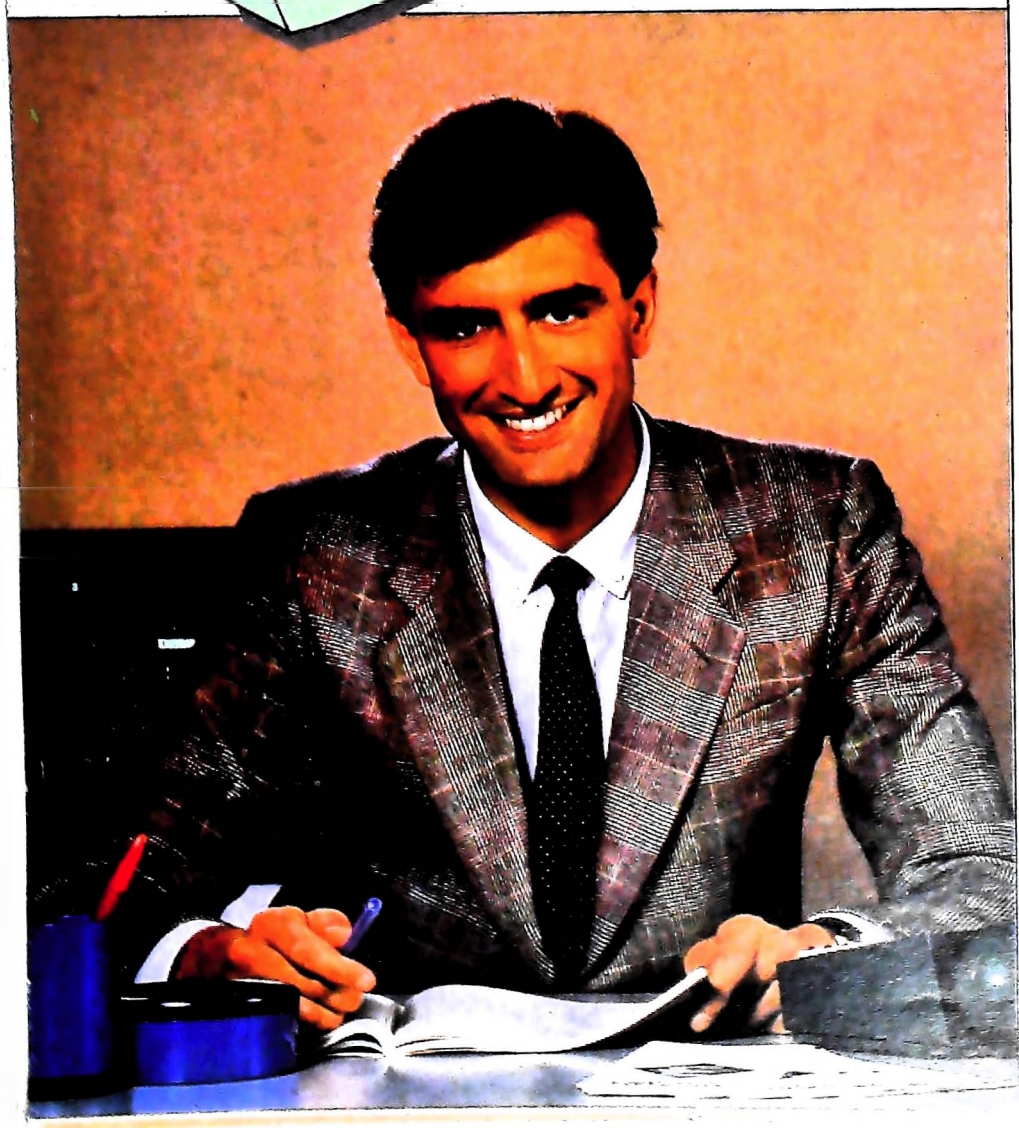
CAP CITTÀ

TELEFONO

Anche in leasing con Olivetti Leasing.

UN NUOVO MODO DI USARE LA BANCA.

Consulenza



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

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

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

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

Veniteci a trovare. ci conosceremo meglio.

BANCO DI ROMA

CONOSCIAMOCI MEGLIO.