128 da zero

di Andrea de Prisco

MMU: scottanti rivelazioni

MMU sta per Memory Management Unit. Nella sua accezione più classica, una MMU serve per tradurre un indirizzo logico in un indirizzo fisico. Molte volte, per motivi di praticità, sono demandati alle MMU altri compiti sempre inerenti la gestione della memoria. Con questo articolo vedremo cosa fa la MMU all'interno del 128, oltre naturalmente a gestire i banchi di memoria discussi alcuni numeri fa.

Cos'è una MMU

Prima di entrare nel merito *centovettotesimo* desideriamo dare alcuni chiarimenti circa l'accezione classica di MMU. Come descritto in Appunti di Informatica di MC numero 53, i calcolatori *veri* dispongono del meccanismo della memoria virtuale. Ciò al fine di ottimizzare l'utilizzo della memoria fisica, da parte dei vari processi in esecuzione.

Semplicisticamente parlando, per ogni programma in esecuzione non è mantenuto in memoria centrale tutto il codice e tutti i dati, ma solo un sottoinsieme di questi necessari per l'elaborazione in quel momento. Se un determinato dato o un pezzo di codice è richiesto, ma non è contenuto in memoria, il sistema provvede a prelevarlo dalla memoria secondaria (dischi) eventualmente scaricando qualcos'altro per «fare posto». A causa di questo fatto, lo spazio di indirizzamento logico di un processo è in generale diverso dai veri e propri indirizzi di memoria e quindi (generalmente a tempo di esecuzione) si rende necessario un meccanismo di traduzione {indirizzo logico}/{indirizzo fisico}, per poter accedere al dato necessario. Se ad esempio il calcolatore in questione implementa la sua memoria virtuale a pagine, un processo in esecuzione potrebbe riferire un dato contenuto nella pagina logica 3, posizione 100. Dal momento che tale pagina logica, sempreché sia presente in memoria, potrebbe essere locata nella pagina fisica 5, l'indirizzo effettivo per prelevare il dato sarà pagina 5 locazione 100.

Per attuare questa traduzione nel più breve tempo possibile (ogni accesso alla memoria deve essere tradotto) tale compito è interamente demandato ad una unità specializzata interposta tra processore e memoria denominata appunto MMU. Essa riceve l'indirizzo logico dal processore, esegue immediatamente la traduzione in indirizzo fisico, richiede il dato alla memoria e lo invia al processore che continua l'elaborazione normalmente, non essendosi accorto di nulla (per *lui* è stato un normale accesso in memoria).

Oltre a questo, una MMU che si rispetti si occupa anche di smistare indirizzamenti a periferiche I/O memory mapped, a segnalare eventuali fault di pagina o di segmento, a gestire le interruzioni (questo assieme al processore).

FF04	PCR D	D50B	VR
FF03	PCR C	D50A	P1 H
FF02	PCR B	D509	P1 L
FF01	PCR A	D508	PØ H
FF00	CR	D507	PØ L
		D506	RCR
		D505	MCR
		D504	PCR D
		D503	PCR C
		D502	PCR B
		D501	PCR A
		D500	CR

Figura 1 - I 12 registri della MMU.

L'MMU del 128

L'utilizzo di una MMU nel 128 non è certamente necessaria per i motivi sopra esposti. Trova la sua ragion d'essere dato che, tutti ormai lo sanno, il processore di questo è capace di indirizzare solo 64 k, mentre la memoria disponibile tra ram e rom è molta di più. Come nel caso dei *veri calcolatori*, avremo che un riferimento logico ad una cella di memoria è dato dalla coppia (banco, posizione) mentre l'indirizzamento fisico... beh, quello proprio non è identificabile dato che la memoria fisica del 128 è sparsa per tutta la macchina sottoforma di due banchi ram da 64 k l'uno, 16 k rom del sistema operativo, 32 k rom del Basic + monitor, generatore dei caratteri, memory mapped I/O ecc.

Purtroppo, a livello hardware, non è possibile che un programma locato in un banco possa fare riferimenti ad altri banchi, se non comandato alla MMU una commutazione di banco. Fortunatamente al livello di sistema operativo ciò non accade essendo disponibili delle apposite routine che permettono di accedere a qualunque locazione di qualsiasi banco semplicemente effettuando opportune chiamate (cfr. MC n. 57, 128 da zero).

12 registri

Per impartire ordini alla MMU, che come vedremo non si occupa solo dei banchi nudi e crudi, si utilizzano 12 re-

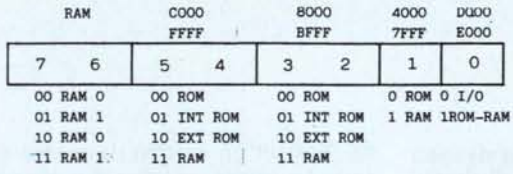


Figura 2 - Registro CR ($D500 - $FF00).

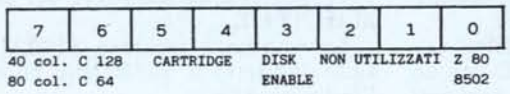


Figura 3 - Registro MCR ($D505).

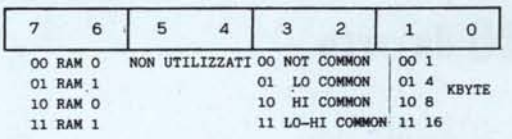


Figura 4 - Registro RCR ($D506).

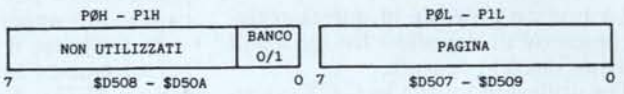


Figura 5 - Registri P0-P1.

gistri mappati a partire dall'indirizzo esadecimale $D500 del banco 15 (figura 1). In quella zona, come più volte ripetuto, è mappato l'I/O della macchina compreso quindi i registri per il suono, per i CIA, per il video ecc. I primi 5 registri della MMU sono inoltre mappati a partire dall'indirizzo esadecimale $FF00 di ogni banco: ciò per evitare che, una volta commutato un determinato banco di memoria, non sapremmo più come tornare indietro.

Del primo registro, $FF00 o $D500, ne abbiamo già parlato nel numero 57, ed è lì che vi rimandiamo per maggiori chiarimenti. Esso è la vera cloche di comando della memoria dato che settando o resettendo i suoi bit si può impostare qualsiasi configurazione, anche non prevista dal comando BANK del Basic. In figura 2 è mostrato tale registro e il significato dei suoi bit.

I registri 1-4, locati a $D501-$D504 del banco 14, e disponibili solo in lettura anche a $FF01..04, servono per impostare delle preconfigurazioni di memoria, quelle che più useremo, richiamabili semplicemente accedendo ai registri copia corrispondenti. Ovvero, una volta settate le nostre configurazioni preferite a partire da $D501, per effettuare una commutazione sarà sufficiente accedere in scrittura nel registro copia corrispondente ($FF01..04) ed essere così *catapultati* nella nuova configurazione di memoria.

Il primo dei registri non disponibili sottoforma di copia è registro Modo di Configurazione (MCR) ed è raffigurato in figura 3. In esso possiamo leggere alcune informazioni a dire il vero non troppo interessanti: ad esempio se all'accensione il tasto 40/80 colonne era premuto o meno. Il bit 0 sembra l'unico degno di nota dato che controlla quale processore è attualmente al lavoro (Z80 o 8502).

A partire dall'indirizzo $D506 le cose si fanno sempre più interessanti. Con questo primo registro (Registro Configurazione Ram, figura 4) è possibile configurare la memoria secondo *altri punti di vista.* Ad esempio possiamo cambiare la ram visibile dal Video Interface Chip (40 colonne) impostando il banco 1. È così possibile effettuare rapidi swap di schermo, sia in bassa che in alta risoluzione (oppure swap di sprite...) semplicemente allocando lo stesso spazio di memoria video sia nel banco 0 che nel banco 1. Per effettuare lo swap sarà sufficiente comunicare alla MMU quale banco deve essere visibile dal VIC e il gioco è fatto.

Sempre nel registro RCR troviamo la possibilità di definire aree comuni ai due banchi in testa o in coda, di dimensioni pari a 1,4,8 o 16 k byte. Per default, come detto sempre alcuni numeri fa, l'area di memoria comune assomma a 1 k, allocato a inizio memoria. Grazie a questo artificio, un programma *giacente* in una zona di memoria comune può ordinare commutazioni di configurazione alla MMU senza perdere il controllo del flusso.

Puntatori di pagina

Tramite la MMU del 128 è possibile definire pagine (non banchi, attenzione) 0 e 1 in qualsiasi punto della memoria del 128. Come si sa, il processore 8502 permette alcuni modi di indirizzamento solo in pagina 0 mentre lo stack di sistema è sempre allocato in pagina 1. Ad esempio, per spostare grosse aree di memoria, chiunque abbia usato solo un po' il linguaggio macchina, conoscerà il modo di indirizzamento:

```
LDA ($PP), Y
```

dove $PP è un indirizzo in pagina 0. Chi invece il linguaggio macchina lo usa spesso e volentieri, avrà notato come le locazioni libere in pagina 0 sono sempre poche (sono quasi tutte adoperate dal sistema) e occorre ricorrere a vari artifizi per rubarne qualcuna in più. Utilizzando opportunamente la MMU possiamo tagliare la testa al toro definendo una nuova pagina 0, ad esempio a partire dall'indirizzo $1000 e disporre così di 256 locazioni di tale tipo, tutte libere per noi. Ovvero, dopo aver impostato opportunamente la MMU, scrivendo:

```
STA $03
```

immetteremo il contenuto dell'accumulatore nella locazione $1003 e un accesso del tipo:

```
LDA ($03), Y
```

equivale a un LDA ($1003), Y addirittura non disponibile normalmente.

Come detto prima, è possibile fare lo stesso giochetto anche per lo stack (pagina 1) nel qual caso potremmo implementarne uno nuovo in qualsiasi punto della memoria. Ciò può essere utile non tanto come nuovo stack, ma come indirizzamento rapido di una qualsiasi area di memoria. Ad esempio per azzerare 256 byte a partire da $1000 allochiamo lì in nostro nuovo stack e, caricato nell'accumulatore il valore 0, non ci resta che dare 256 PHA per essere accontentati. Si noti che un «PHA» è ben più rapido di un normale «STA 1000, X» dato che il primo richiede 3 cicli di clock il secondo 5. In figura 5 sono mostrati i registri interessati, per la pagina 0 e 1. Le rispettive parti basse indicano la pagina riferita come pagina 0 o 1, mentre delle parti alte interessa solo il bit meno significativo nel quale indicheremo se ci riferiamo al banco 0 o 1 della ram.

Attenzione a rimettere a posto stack, stack pointer e pagina 0 dopo l'uso: avremmo sicuramente effetti catastrofici dimenticandocene.

Per finire, in figura 6, è mostrato il registro Versione, nel quale possiamo leggere (!) quanti banchi ram possiede il nostro 128 e, addirittura!, la versione della nostra MMU. Il massimo.

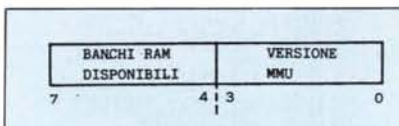


Figura 6- Registro VR ($D50B).

Kernal di I/O

di Asnaghi Adriano - Mestre (VE)

La routine presente in questa sezione permette di definire i file sui quali l'utente intende operare.

Per utilizzare un file in C128 occorre passare alle routine specializzate il nome del file, i parametri di modo, il banco dove prelevare il nome stesso ed il banco dove prelevare (SAVE) o mettere (LOAD) il file stesso. Questi lavori sono eseguiti dalle seguenti routine.

1) Definisce il BANCO per il NOME e l'AREA di memoria.

La routine è locata a $FF68 con salto a $F73F. I parametri da fornire sono:

Accumulatore - indice di configurazione del banco per l'area di memoria;

Reg. Y - indice di configurazione del banco per il nome del file.

L'utente può anche fare a meno di utilizzare la routine se memorizza nelle locazioni di pagina zero $C6 e $C7 rispettivamente il banco per il nome del file e quello per l'area di memoria.

2) Definisce i PARAMETRI del file.

La routine è locata a $FFBA con un salto a $F738. Essa utilizza i seguenti parametri:

Accumulatore - numero logico del file;

Reg. X - indirizzo device;

Reg. Y - indirizzo secondario.

I valori possono essere memorizzati direttamente dall'utente nelle posizioni $B8, $BA e $B9 di pagina zero.

Per l'indirizzo di device si ha la seguente assegnazione:

0-3 tastiera, tape, user-port (RS-232), screen;

4-7 printer;

8-11 disk drive.

Per l'indirizzo secondario si devono consultare i manuali relativi ai dispositivi usati. Per esempio un indirizzo secondario 0 per la stampante specifica il modo upper/graphic, ecc.

3) Definisce il NOME del file.

La routine è locata a $FFBD con un salto a $F731. Essa utilizza i seguenti parametri:

Accumulatore - lunghezza del nome del file;

Reg. X - indirizzo basso del nome del file;

Reg. Y - indirizzo alto del nome del file.

I valori possono essere memorizzati direttamente dall'utente nelle posizioni $B7, $BB e $BC di pagina zero.

4) APRE il file.

La routine è locata a $FFC0 con un salto indiretto a $031A che contiene l'indirizzo $EFBD. Nessun parametro è richiesto.

Finché questa routine non è eseguita, non è possibile aprire canali di Input o Output.

5) Apre un canale di INPUT o di OUTPUT.

Esiste una routine per l'INPUT locata a $FFC6 con salto indiretto a $031E che contiene l'indirizzo $F106 ed una per l'OUTPUT locata a $FFC9 con salto relativo a $0320 che contiene l'indirizzo $F14C. I parametri da passare sono:

Reg. X - numero logico del file.

Le routine ritornano un indicatore di errore nel Carry. Se esso è 0 l'operazione è avvenuta correttamente.

6) LEGGE un carattere dal canale di INPUT.

La routine è locata a $FFCF con salto indiretto a $0324 che contiene l'indirizzo $EF06.

La routine legge un carattere del canale di input definito con la routine in 5). Nessun parametro deve essere passato. La routine ritorna il carattere letto nell'Accumulatore.

7) SCRIVE un carattere nel canale di OUTPUT.

La routine è locata a $FFD2 con salto indiretto a $0326 che contiene l'indirizzo $EF79.

La routine manda il carattere al canale di output definito con la routine in 5). Il carattere da scrivere deve essere passato nell'Accumulatore.

Se si desidera invece mandare un carattere direttamente sul video, alla posizione corrente del cursore, basta utilizzare la routine locata a $C72D.

8) CHIUDE il file.

La routine è locata a $FFC3 con salto indiretto a $031C che contiene l'indirizzo $FF18.

La routine chiude il file logico il cui valore è specificato nell'Accumulatore.

La routine ritorna un indicatore di errore nel Carry. Se esso è a 0 l'operazione si è conclusa correttamente.

9) CARICA un file in MEMORIA.

La routine è locata a $FFD5 con salto a $F265. Prima di chiamare questa routine devono essere definiti i banchi per il nome del file e per l'area di memoria, il numero logico del file ed il nome del file stesso.

I parametri da passare alla routine sono:

Reg. X - indirizzo basso dell'area di memoria dove verrà caricato il file;

Reg. Y - indirizzo alto dell'area di memoria dove verrà caricato il file.

Nella figura A pubblichiamo un esempio di caricamento:

10) SALVA la memoria su di un FILE.

La routine è locata a $FFD8 con salto a $F53E. Come per la routine precedente, anche questa deve essere chiamata dopo aver definito il banco per l'area di memoria, per il nome del file, ecc.

I parametri da passare alla routine sono:

Accumulatore - indirizzo di pagina zero di due byte contenenti rispettivamente l'indirizzo basso e alto indicanti l'indirizzo dell'inizio dell'area da salvare;

Reg. X - indirizzo basso della fine dell'area da salvare;

Reg. Y - indirizzo alto della fine dell'area da salvare.

Nella figura B possiamo vedere un esempio di salvataggio.

MC

A

```
LOADF   JMP ENTRY  ; skip nome file
;
NOMEF   .DC 'PROVA,P'
;
ENTRY   LDA #$00    ;area in bank 0
        TAX         ;nome file in bank 0
        JSR $FF68   ;set bank
        LDA #$01    ;numero logico del file
        LDX #$08    ;indirizzo device (8=disk)
        LDY #$00    ;indirizzo secondario (0=read)
        JSR $FFBA   ;set parametri del file
        LDA #$07    ;lunghezza nome file
        LDX #>NOMEF ;indirizzo basso area nome
        LDY #<NOMEF ;indirizzo alto area nome
        JSR $FFBD   ;set nome del file
        LDX #$00    ;indirizzo basso area memoria
        LDY #$40    ;indirizzo alto area memoria
        LDA #$00    ;flag per LOAD (VERIFY=01)
        JSR $FFD5   ;carica a $4000
        RTS         ;rientra
```

B

```
SAVEF   JMP ENTRY  ; skip nome file
;
NOMEF   .DC 'PROVA,P'
;
ENTRY   LDA #$00    ;area in bank 0
        TAX         ;nome file in bank 0
        JSR $FF68   ;set bank
        LDA #$01    ;numero logico del file
        LDX #$08    ;indirizzo device (8=disk)
        LDY #$00    ;indirizzo secondario (0=read)
        JSR $FFBA   ;set parametri del file
        LDA #$07    ;lunghezza nome file
        LDX #>NOMEF ;indirizzo basso area nome
        LDY #<NOMEF ;indirizzo alto area nome
        JSR $FFBD   ;set nome del file
        LDA #$00    ;indirizzo basso inizio area memoria
        STA *$FA    ;e salva
        LDA #$1C    ;indirizzo alto inizio area memoria
        STA *$F8    ;e salva
        LDX #$00    ;indirizzo basso fine area memoria
        LDY #$20    ;indirizzo alto fine area memoria
        LDA #$FA    ;pointer pagina zero
        JSR $FFD8   ;salva da $1C00 a $2000
        RTS         ;rientra
```

Gruppo Distributori Associati

IL MIGLIOR SERVIZIO AL MIGLIOR PREZZO
PRODOTTI CON GARANZIA UFFICIALE

Sede: S. Martino Siccomario (Pavia) - S.S. dei giovi (ang. via Gabba) - ☎ 0382 - 49.94.39

COMPUTERS:

OLIVETTI

- **M24** - 640K RAM, 2 FDD 360K, VIDEO, TASTIERA. **L. 3.450.000**
- **M24** - 640K RAM, 1 UNITÀ FDD 360 K, 1 HD 20 Mb, VIDEO, TASTIERA. **L. 4.450.000**
- **M24 SP** - 640K RAM, 1 FDD 360K, 1 HD 20 Mb, VIDEO , TASTIERA. **L. 5.500.000**
- **M28** - 512K RAM, 1 FDD 1,2 Mb, 1 HD 20 Mb, VIDEO, TASTIERA. **L. 6.990.000**
- **M28** - 512K RAM, 1 FDD 1,2 Mb, 1 HD 20 Mb, VIDEO, TASTIERA, STREAMER DA 20 Mb. **L. 7.980.000**
- **M28** - 512K RAM, 1 FDD 1,2 Mb, 1 HD 40 Mb, VIDEO, TASTIERA, STREAMER DA 20 Mb. **L. 9.100.000**

Box espansione 1 floppy da 369K autoalimentato × M28 **L. 862.000**

Streamer interno da 20 Mb × M28 **L. 1.912.000**

Streamer da 20 Mb in box esterno autoalimentato × M28 **L. 2.378.000**

EPSON

- **PC** - 256 K RAM, 2 FDD 360K, VIDEO, TASTIERA **L. 2.245.000**
- **PC/C** - 256 K RAM, 2 FDD 360K, VIDEOGRAFICO a colori, TASTIERA **L. 2.830.000**
- **PC/HD** - 256 K RAM, 1 FDD 360K, HD 20 Mb, VIDEO, TASTIERA **L. 3.493.000**
- **PC/HDC** - 256 K RAM, 1 FDD 360K, HD 20 Mb, VIDEOGRAFICO a colori, TASTIERA **L. 4.077.000**
- **PC+** - 640 K RAM, 2 FDD 360K, 8086, VIDEO GRAFICO, TASTIERA **L. 3.078.000**
- **PC+/HD** - 640 K RAM, 1 FDD 360K, 1 HD 20 Mb, VIDEO GRAFICO, TASTIERA **L. 4.310.000**
- **PC/AXM** - 80286 6-8-10 Mhz, 640K, 1 FDD 1,2 Mb, VIDEO, TASTIERA **L. 4.310.000**
- **PC/AX HD2** - 80286 6-8-10 Mhz, 640K, 1 FDD 1,2 Mb, 1 HD 20 Mb, VIDEO, TASTIERA **L. 5.724.000**
- **PC/AX HD4** - 80286 6-8-10 Mhz, 640K, 1 FDD 1,2 Mb, 1 HD 40 Mb, VIDEO, TASTIERA **L. 7.074.000**

12 MESI DI GARANZIA INTEGRALE

AMIGA 2000

- **AMIGA** - 1024 K RAM, VIDEO a colori, TASTIERA, MOUSE **L. 2.550.000**
- **DRIVE 3,5"** - ESTERNO TECNOLOGIA TOSHIBA **L. 349.000**
- **Dischetti** da 3,5" doppia faccia (min. 50 pezzi) **L. 3.800**

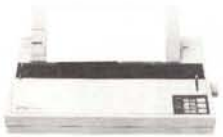
Oltre 700 programmi disponibili con arrivi settimanali dalla germania e dagli Stati Uniti!
Ogni sabato pomeriggio dimostrazione in sede.
(Richiedeteci la lista programmi).

STAMPANTI:

EPSON

- **LX 86 F/T** - 80 colonne, 120 CPS, grafica, parallela. **L. 626.000**

- **FX 800** - 80 colonne, 200 CPS, grafica, parallela. **L. 810.000**
- **EX 800** - 80 colonne, 250 CPS, grafica, parallela. **L. 1.047.000**



- **FX 1000** - 136 colonne, 200 CPS, grafica, parallela. **L. 918.000**
- **EX 1000** - 136 colonne, 250 CPS, grafica, parallela. **L. 1.490.000**

- **LQ 800 F/T** - 80 colonne, 180 CPS, grafica, parallela, 60 CPS NLQ **L. 1.350.000**
- **LQ 1000 F/T** - 136 colonne, 180 CPS, grafica, parallela **L. 1.598.000**
- **LQ 2500 F/T** - 136 colonne, 270 CPS, grafica, parallela. **L. 1.980.000**

ACCESSORI PER STAMPANTI EPSON

- **KIT COLORE** per serie EX-800 / EX-1000 **L. 150.000**
- **KIT COLORE** per serie LQ-2500 **L. 150.000**
- **INTERFACCIA SERIALE** per serie FX-800 / FX-1000 **L. 150.000**
- **INSERITORE** fogli singoli per LX-86 **L. 210.000**
- **INSERITORE** fogli singoli per FX-800 **L. 380.000**
- **INSERITORE** fogli singoli per FX-1000 **L. 470.000**
- **INSERITORE** fogli singoli per EX-800 **L. 380.000**
- **INSERITORE** fogli singoli per EX-1000 **L. 470.000**

NASTRI ORIGINALI PER STAMPANTI

- **EPSON** LX-86 **L. 10.900**
- **EPSON** FX-800 **L. 8.000**
- **EPSON** FX-1000 **L. 10.500**
- **EPSON** FX-85 **L. 8.000**
- **EPSON** FX-105 **L. 10.500**
- **EPSON** EX-800 **L. 18.500**
- **EPSON** EX-1000 **L. 18.500**
- **EPSON** LQ-800 **L. 16.000**
- **EPSON** LQ-2500 **L. 24.000**
- **EPSON** LQ-1500 **L. 11.000**

confezioni da 12 pezzi

- **TALLY** MT 130/140/180/290 **L. 11.800**
- **TALLY** MT 80 **L. 10.500**
- **COMMODORE** 803 **L. 10.500**

IBM COMPATIBILI

- **F2** - XT 512K, 2 FDD 360K, TASTIERA, MONITOR PHILIPS **L. 1.690.000**
- **F20** - XT 512K, 1 FDD 360K, 1 HD 20 Mb, TASTIERA, MONITOR PHILIPS **L. 2.490.000**

HARD-DISK

• MONTAGGIO INTERNO PER PC/XT: •

KIT 10 Mb SLIM **L. 850.000**
KIT 20 Mb SLIM **L. 980.000**
KIT 30 Mb SLIM **L. 2.340.000**

• MONTAGGIO ESTERNO PER PC/XT: •

KIT 10 Mb **L. 1.450.000**
KIT 20 Mb **L. 1.570.000**
KIT 33 Mb **L. 2.850.000**

• MONTAGGIO INTERNO PER AT: •

KIT 20 Mb **L. 1.460.000**
KIT 30 Mb **L. 1.725.000**
KIT 40 Mb **L. 1.950.000**

CARD-DISK

- **MINISCRIBE** 20 Mb **L. 1.390.000**

GARANZIA 12 MESI

DESIDERO RICEVERE GRATUITAMENTE
IL VOSTRO CATALOGO COMPLETO

Cognome .....

Nome .....

Indirizzo .....

Professione .....

Firma .....

ASSISTENZA TECNICA SPECIALIZZATA • VENDITA RATEALE O LEASING • VENDITA PER CORRISPONDENZA

CONDIZIONI DI VENDITA: Il pagamento potrà essere effettuato in forma anticipata a mezzo vaglia telegrafico o assegno circolare o in contrassegno tramite posta o corriere. Le spese sono a carico del destinatario.
La spedizione è prevista entro 15 gg.

I PREZZI SONO IVA ESCLUSA

Ogni mese in edicola

le riviste con disco programmi per IBM e compatibili, Apple e Commodore.

Lei possiede un computer IBM
o compatibile? Oppure un Apple II?
Oppure, ancora, un Commodore 64 o un 128?
Allora acquisti subito in edicola PcDisk, AppleDisk o CommoDisk.
Si tratta di una novità eccezionale, e cioè di tre riviste i cui contenuti sono composti da recensioni, articoli di fondo d'interesse generale per gli utenti di personal computer, ma anche dalla descrizione dettagliata dei programmi registrati sul disco allegato alla rivista. E non si tratta di "programmini", bensì di programmi utili, il cui costo tradizionale sarebbe di decine o, in alcuni casi, anche di centinaia di migliaia di lire ciascuno.
AppleDisk, CommoDisk, PcDisk sono in vendita in tutte le edicole d'Italia a 15.000 lire l'una (CommoDisk a 13.000 lire). Se il suo edicolante ne fosse sprovvisto, prenoti presso di lui il prossimo numero.
Perderlo sarebbe un peccato!

AppleDisk - PCDISK - COMMODISK - sono pubblicazioni:
Editronica - 20122 Milano - c.so Montorte, 39 - Tel. 02-702429