

128 da zero

di Andrea de Prisco

Banking da linguaggio macchina

Dopo aver visto il mese scorso come il Commodore 128 mette a disposizione tutta la sua memoria (ram e rom) al livello del linguaggio di programmazione Basic, questo mese ci addentreremo maggiormente nel merito mostrando come siano possibili analoghi «smanettamenti» anche da linguaggio macchina.

Il Monitor

Per programmare in linguaggio macchina occorre un buon monitor: ovviamente i fosfori in questo caso non c'entrano proprio nulla. Ci serve un monitor per il linguaggio macchina, che stavolta mamma Commodore s'è ricordata di infilare dentro alle rom di sistema, come aveva già fatto tanti anni fa col dottor Pet. Un monitor di linguaggio macchina è dunque uno strumento indispensabile per poter manipolare facilmente le celle di memoria o per essere più precisi i contenuti di queste. Il monitor del 128 fa anche qualcosa in più: assembla e disassembla codici mnemonici e porzioni di memoria, converte con estrema facilità numeri da una base all'altra

non senza permettere una ragionevole interazione tra computer e la più importante delle periferiche, l'unità a dischi, almeno per quanto riguarda directory e comandi DOS.

Tutto ciò è comunque ben illustrato sul manuale fornito con la macchina al quale naturalmente vi rimandiamo qualora non aveste mai usato questa feature.

Il registro CR della MMU

Ogni manipolazione della configurazione della Memoria coinvolge sempre la MMU del 128 il cui acronimo sta appunto per Memory Management Unit. Per dialogare con questa unità, come avviene anche per gli altri processori interni al 128, si usano dei

registri mappati in alcune celle di memoria.

Noi faremo riferimento, per tutto quest'articolo, al registro CR (già nominato il mese scorso) locato all'indirizzo esadecimale \$FF00. Sempre come già detto, tale registro è ovviamente accessibile da qualsiasi configurazione di memoria dato che deve essere sempre possibile passare da un banco ad un altro: se in qualche banco questo non fosse accessibile, una volta selezionato quel banco non si potrebbe più «venir via» non potendo dialogare con la MMU.

Di ciò si evince che per passare da un banco ad un altro, da linguaggio macchina basta infilare qualcosa nel registro CR per ottenere il voluto. Quasi.



128 da zero

Il problema infatti è un pò più complesso: infatti anche il flusso di controllo del programma in corso verrebbe catapultato indesideratamente sul banco desiderato. Facciamo un esempio: immaginiamo di aver scritto un programma in linguaggio macchina nel banco 0 della memoria. Sempre per ipotesi poniamo il caso in cui a un certo punto ci serve il contenuto della locazione 4000 del banco 1, interamente riempito di dati (ovvero non-programmi). La situazione è mostrata in figura 1: dopo qualche operazione si pone \$7F nel registro CR per cambiare banco prima di prelevare col

LDA la locazione 4000. Abbiamo combinato un bel pasticcio: infatti i cambiamenti di banco non riguardano solo gli accessi ai dati da parte delle istruzioni in linguaggio macchina ma quanto effettivamente il processore preleva per eseguire. Nella fattispecie, l'aver eseguito la sequenza:

```
LDX # $7F
STX $FF00
```

corrisponde in pratica (solo quelle due istruzioni) ad aver effettuato un salto, un JMP, all'altro banco di memoria, con le catastrofiche conseguenze che possiamo supporre: sicuramen-

te il blocco del sistema fino a nuovo Reset. Questo perchè siamo saltati in mezzo ai dati (leggi: numeri a casaccio per il processore) e, si sa, non tutti i numeri compresi tra 0 e 255 sono codici operativi di istruzioni di macchina.

Considerato poi che la situazione di cui sopra è tutt'altro che irrealistica, se non ci fosse una soluzione sarebbe davvero un bel casotto. Fortunatamente di soluzioni ce ne sono due, una hardware e l'altra software.

Spegnete pure il saldatore, non dobbiamo fare nessuna modifica è tutto compreso nel prezzo. Hardware nel senso che ci riferiremo al modo come è stato costruito il 128 e Software rifacendoci ad opportune routine di sistema operativo. In ogni caso farina di mamma Commodore.

Prima soluzione

Alla base di tutti i cambiamenti di banco, oltre al registro CR esiste un altro componente altrettanto importante: il primo K di memoria RAM comune sia a RAM 0 che a RAM 1: le due pagine di memoria disponibili su 128. Ciò vuol dire essenzialmente che quanto mostrato in figura 1, se l'avessimo effettuato nelle prime 1024 locazioni di memoria non avrebbe provocato nulla di paranormale.

La figura 2 mostra come stanno i fatti: si nota come la cosiddetta Common RAM sia indipendente dal banco selezionato. Allocando le nostre routine nel primo K di memoria (figura 3), non si hanno tali problemi: di fatto la Commodore stessa usa questa zona per effettuare le sue interazioni tra pagine, a livello di sistema operativo e/o Basic 7.0.

L'unico problema è dato dal fatto che in questo primo K non troviamo molto spazio libero per i nostri programmi essendo quasi interamente occupato da variabili e sottoprogrammi di sistema operativo. In altre parole, se ci serve qualche buco qua e là otteniamo facilmente il nostro scopo, ma se servono grosse zone di memoria non possiamo non ricorrere al metodo software che andiamo subito ad illustrare.

Soluzione software

Lasciamo dunque il nostro primo K di memoria e scriviamo comodamente il nostro programma in linguaggio

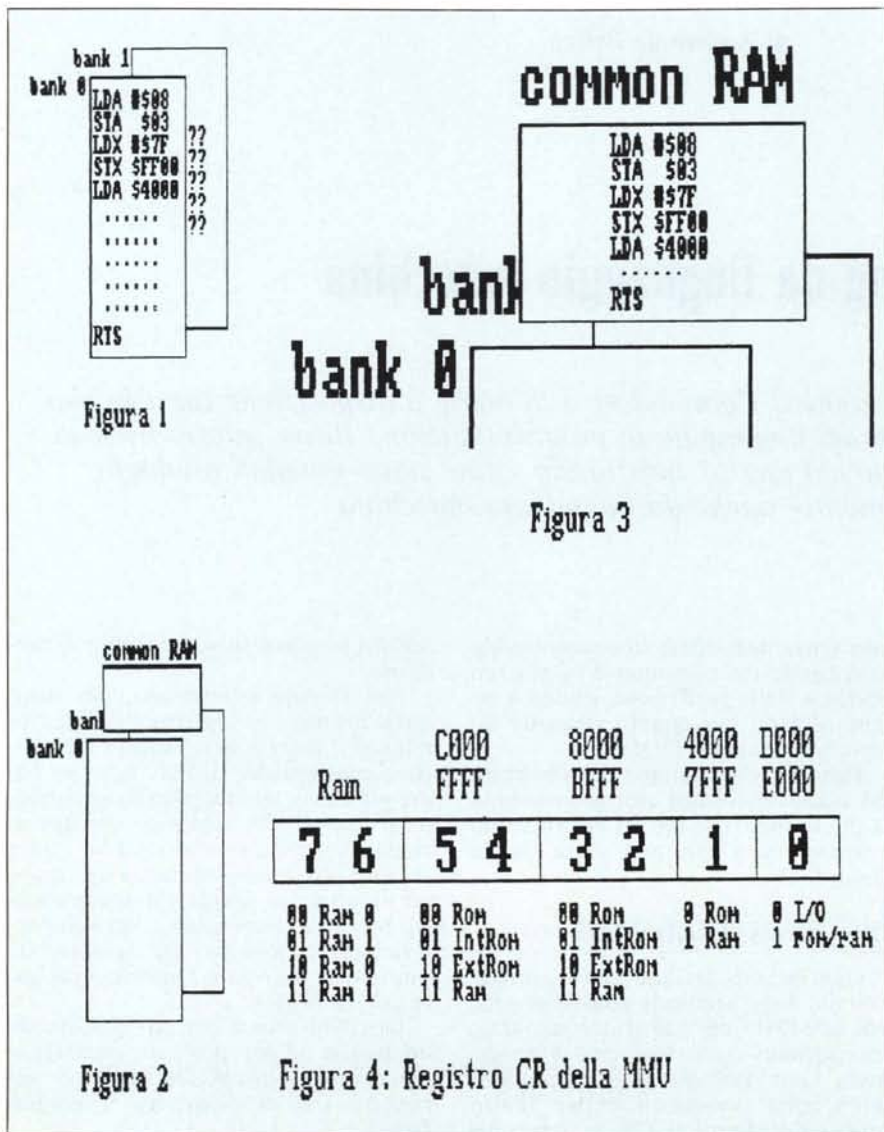


Tabella 2
Corrispondenza tra numero
banco e valore di CR

n.banco	valore cr
0	\$3F
1	\$7F
2	\$BF
3	\$FF
4	\$16
5	\$56
6	\$96
7	\$D6
8	\$2A
9	\$6A
10	\$6A
11	\$EA
12	\$06
13	\$0A
14	\$01
15	\$00

Tabella 1

Tabella riassuntiva delle routine del kernel per la manipolazione dei banchi di memoria

Routine	Descrizione	Indirizzo	Parametri IN	Parametri OUT
GETCFG	Trasforma il n.ro del banco nel corrispondente valore di CR	\$F7EC 63438	X = # banco	A = Valore CR
JSRFAR	Chiama dal banco 15 una subroutine posta in qualsiasi banco	\$02CD 717	\$02 = # banco \$03 = Hi addr. \$04 = Lo addr. \$05 = status \$06 = registro A \$07 = registro X \$08 = registro Y	\$05 = status \$06 = registro A \$08 = registro Y \$09 = stack pointer
JMPFAR	Salta dal banco 15 ad una locazione posta in qualsiasi banco	\$02E3 739	\$02 = # banco \$03 = Hi addr. \$04 = Lo addr. \$05 = status \$06 = registro A \$07 = registro X \$08 = registro Y	
INDFET	Legge il contenuto di una cella posta in qualsiasi banco	\$F7D0 63440	A = Puntatore al puntatore al byte X = # banco Y = offset	A = Byte cercato
INDSTA	Scrive un byte in una cella posta in qualsiasi banco	\$F7DA 63450	A = Byte da scrivere \$2B9 = Puntatore al puntatore alla cella X = # banco Y = offset	
INDCMP	Confronta l'accumulatore con una cella posta in qualsiasi banco	\$F7E3 63459	\$2CB, A, Puntatore al puntatore alla cella A = carattere da comparare X = # banco	Status register

Listato 1

```
. FF7EC BD F0 F7 LDA $F7F0,X
. FF7EF 60      RTS
```

Listato 2

```
. 002CD 20 E3 02 JSR $02E3
. 002D0 85 06    STA $06
. 002D2 86 07    STX $07
. 002D4 84 08    STY $08
. 002D6 08      PHP
. 002D7 68      PLA
. 002D8 85 05    STA $05
. 002DA BA      TSX
. 002DB 86 09    STX $09
. 002DD A9 00    LDA #$00
. 002DF 8D 00 FF STA $FF00
. 002E2 60      RTS
. 002E3 A2 00    LDX #$00
. 002E5 B5 03    LDA $03,X
. 002E7 48      PHA
. 002E8 E8      INX
. 002E9 E0 03    CPX #$03
. 002EB 90 F8    BCC $02E5
. 002ED A6 02    LDX $02
. 002EF 20 68 FF JSR $FF6B
. 002F2 8D 00 FF STA $FF00
. 002F5 A5 06    LDA $06
. 002F7 A6 07    LDX $07
. 002F9 A4 08    LDY $08
. 002FB 40      RTI
```

Listato 3

```
. FF7D0 8D AA 02 STA $02AA
. FF7D3 BD F0 F7 LDA $F7F0,X
. FF7D6 AA      TAX
. FF7D7 4C A2 02 JMP $02A2

. 002A2 AD 00 FF LDA $FF00
. 002A5 8E 00 FF STX $FF00
. 002A8 AA      TAX
. 002A9 B1 66    LDA ($66),Y
. 002AB 8E 00 FF STX $FF00
. 002AE 60      RTS
```

macchina dove ci pare (o quasi). Per interagire con gli altri banchi possiamo usare alcune routine di sistema operativo atte allo scopo.

La tabella 1 riassume tali routine che, come vedremo, sono tutte piuttosto semplici sia da capire che da usare. A tale scopo ci riferiremo anche ai listati in linguaggio macchina che le descrivono, presenti in queste pagine.

La prima routine serve per trasformare il numero di un banco (0-15) nel corrispondente valore da «pok-are» nel registro CR della MMU. Per usarla è sufficiente mettere in X il banco desiderato ed effettuare un JSR all'indirizzo \$F7EC del banco 15. Al ritorno da questa, troveremo nel registro A il valore corrispondente. Dando uno sguardo al listato 1 possiamo notare quanto sia banale tale trasformazione che non è altro che la lettura di un valore in una tabella posta a partire dall'indirizzo \$F7F0.

La seconda routine, mappata nel primo K (guardacaso) a partire dall'indirizzo esadecimale \$02CD permette di chiamare una subroutine posta in qualsiasi banco a condizione però che la chiamata avvenga dal banco 15 (a tal proposito fra un po' dirò qualcosa). Prima di utilizzarla è necessario caricare un po' di celle di memo-

ria con il desiderato, nella fattispecie metteremo nella cella \$02 il numero del banco desiderato (0-15), nelle due successive l'indirizzo dove è posta la nostra subroutine, nelle celle \$05 e seguenti rispettivamente il registro di stato e i registri A-X-Y coi quali desideriamo che venga eseguita la subroutine. Settati tali parametri, possiamo eseguire dal nostro banco (qualsiasi) JSR \$02CD per ottenere quanto voluto. I risultati dell'esecuzione della subroutine li ritroveremo nelle celle \$05-\$09 come mostrato in tabella 1.

Proviamo ora a commentare quanto succede al momento della chiamata a \$02CD, tenendo sott'occhio il listato 2. La prima operazione è una chiamata di subroutine all'indirizzo \$02E3 (per la cronaca questo è l'indirizzo della prossima routine che mostreremo). Essenzialmente li succede che le celle contenenti indirizzo e status vengono infilate nello stack (vedremo tra poco perché); di seguito a questo è effettuata la trasformazione da numero di banco desiderato a corrispondente valore di CR (JSR \$FF6B corrisponde a saltare al listato 1, anche se con un passaggio in più) per poi aggiornarlo ovvero switch-are banco; infine sono caricati i registri A-X-Y coi corrispondenti valori immessi precedentemente



128 da zero

alla chiamata nelle celle \$06-\$08.

L'RTI di fondo quasi per magia effettua il salto alla subroutine desiderata, nonostante non vi sia stato alcun interrupt: è solo che alla Commodore sono molto furbi. Infatti l'RTI non fa altro che scaricare dallo stack un primo valore per porlo come status, di seguito a questo preleva i due successivi valori e, interpretati come un indirizzo di 16 bit, salta alla locazione così ottenuta.

Badaben-badaben-badaben che la subroutine a cui siamo diretti prima o poi terminerà con un RTS (se no, che subroutine è?) e la domanda da porci

è naturalmente: dopo tutti questi spopolamenti dove mai torneremo?

Semplice, a \$02D0 ovvero all'istruzione successiva al JSR \$02E3 che abbiamo effettuato qualche centinaio di parole fa. In effetti ancora non è finita: occorre mettere in \$06-\$08 il contenuto dei registri A-X-Y, in \$05 lo status e in \$FF00 il valore 0 per ripristinare il banco chiamante, 15.

Apriamo, prima di continuare, una piccola parentesi: per la scrittura di articoli specifici come questi, è d'obbligo una massiccia documentazione in tema prima di cimentarsi in tali «prodezze». Nel caso di 128 da zero,

dato che di programmer's references guide della Commodore, fino a questo momento manco a parlarne, il «riferimento» è un libro edito dalla Abacus Software denominato Tricks and Tips. Peccato che se non si sta veramente attenti a quello che c'è scritto si finisce per confondersi le idee più che senza libri affatto. Tanto per citarne una, la routine appena commentata è spacciata per «chiamata di subroutine da qualsiasi banco in qualsiasi banco» con tanto di cella \$09 atta a contenere il banco chiamante... Oppure i 128 americani sono diversi... oppure è diverso il 128 del sottoscritto...

Bene, tornando alla tabella 1, la successiva routine permette di saltare a una qualsiasi locazione posta in qualsiasi banco. Inutile dirvi che non commenteremo il listato dato che esso corrisponde alla porzione \$02E3-\$02FB che abbiamo già mostrato. Passiamo oltre: tocca a INFET, mappata a partire dall'indirizzo \$F7D0 (vedi listato 3) che permette di leggere il contenuto di una cella posta in qualsiasi banco. Come al solito tutta la complessità sta nel settare i parametri, tra cui il puntatore al puntatore (!) al byte cercato da porre nel registro A. Ovvero ci scegliamo due celle contigue in pagina zero e posto in esse l'indirizzo voluto, scriviamo in A l'indirizzo della prima di queste due celle. Come è consuetudine del linguaggio macchina, indirizzi a 16 bit vengono spezzati mettendo la parte bassa nella prima delle due celle e la parte alta nella seconda. Caricheremo poi in X il banco desiderato e in Y un eventuale offset per accessi indicati. Al ritorno da questa routine troveremo in A il byte cercato.

Di fattura assai simile, la successiva routine, mappata all'indirizzo \$F7DA permette di scrivere un byte in una cella posta in qualsiasi banco. Metteremo come prima in X il banco desiderato, in Y un eventuale offset mentre in A il byte da scrivere e nella cella \$02B9 (come prima in A) il puntatore al puntatore alla cella. Semplice, no?

Stiamo finendo: l'ultima routine permette di confrontare l'accumulatore col contenuto di una cella posta in qualsiasi banco. Il risultato lo otterremo naturalmente nello status register come per qualsiasi altra comparazione semplice. Come prima in \$02C8 va messo il puntatore al puntatore alla cella e in X il banco desiderato.

Listato 4				Listato 5			
. FF7DA	48	PHA		. FF7E3	48	PHA	
. FF7DB	BD F0 F7	LDA \$F7F0,X		. FF7E4	BD F0 F7	LDA \$F7F0,X	
. FF7DE	AA	TAX		. FF7E7	AA	TAX	
. FF7DF	68	PLA		. FF7E8	68	PLA	
. FF7E0	4C AF 02	JMP \$02AF		. FF7E9	4C BE 02	JMP \$02BE	
. 002AF	48	PHA		. 002BE	48	PHA	
. 002B0	AD 00 FF	LDA \$FF00		. 002BF	AD 00 FF	LDA \$FF00	
. 002B3	8E 00 FF	STX \$FF00		. 002C2	8E 00 FF	STX \$FF00	
. 002B6	AA	TAX		. 002C5	AA	TAX	
. 002B7	68	PLA		. 002C6	68	PLA	
. 002B8	91 FF	STA (\$FF),Y		. 002C7	D1 FF	CMP (\$FF),Y	
. 002BA	8E 00 FF	STX \$FF00		. 002C9	8E 00 FF	STX \$FF00	
. 002BD	60	RTS		. 002CC	60	RTS	

Per chi non sa

Brevemente dedicheremo questo riquadro a coloro i quali non conoscono (ancora) il linguaggio macchina del 6502 e gentile famiglia, tra cui il processore del 128 l'8502. Ovviamente non faremo un corso d'Assembler neppure alla lontana, ci limiteremo semplicemente a spiegare le istruzioni che il processore è in grado di eseguire.

LDA, LDX, LDY, STA, STX, STY, servono rispettivamente per caricare un byte in uno dei tre registri dell'8502 (LD sta per load) o per scaricare il contenuto di uno di questi tre registri in una cella di memoria (ST sta per store).

PHP, PHA, PLP, PLA servono per immettere nello o togliere dallo stack il puntatore al top dello stack stesso o il contenuto del registro A. Esistono poi istruzioni per eseguire trasferimenti tra registri come TSX, TXS, TAX, TXA, TAY, TYA che spostano i contenuti dei registri A, X, Y e S detto anche status register.

A questi si aggiungono operazioni per

incrementare un registro (INX, INY) o per il decremento (DEX, DEY) o riferite a una qualsiasi cella di memoria (INC, DEC). È possibile confrontare un registro con un dato (CMP, CPX, CPY, BIT) così come effettuare un salto a seconda di una condizione verificatasi precedentemente la quale ha settato o resettato uno dei bit (C, Z, N, V) dello status register. Tali operazioni di salto condizionato sono: BCC, BCS, BVC, BVS, BNE, BEQ, BMI, BPL.

A queste aggiungiamo un paio di operazioni per eseguire somma e sottrazione di byte (ADC, SBC), operazioni logiche come l'and, l'or inclusivo, l'or esclusivo (AND, ORA, EOR), shift a destra e a sinistra di una posizione (ASL, LSR), rotazione di byte (ROR, ROL), manipolazione del registro di stato (SEC, CLC, SED, CLD, SEI, CLI, CLV), salto incondizionato (JMP), interrupt da programma (BRK), salto a sottoprogramma (JSR) e relativo ritorno (RTS), ritorno da routine di manipolazione delle interruzioni (RTI) e per ultima operazione NOP che non fa un bel nulla.

GVH

linea

computer

GVH - Via della Beverara, 39 - 40131 Bologna - Tel. 051/370687- 360526 - Telex 511375 GVH I

COMPATIBILI IBM®

* IBM è un marchio registrato della International Business Machines Corp.



P 14 T - Personal computer CPU 8088.2/veloce (8087 opz) Main board turbo clock 4.77-8MHz con installati 256K espandibili 640K 8 slot. Controller per 2 disk drive. Installati n. 1 disk drive a trazione diretta da 360Kb meccanica slim. Altoparlante interno. Contenitore metallico, con coperchio apribile, look AT. Tastiera ergonomica con 10 tasti funzione. Modello 5060. Alta affidabilità con elettronica capacitiva. Approvata norme FCC. Corredata di scheda CX 20 monocromatica alta risoluzione uscita parallela oppure a scelta di scheda CX 25 grafica video colore+parallela (precisare nell'ordinazione). Montato collaudato con garanzia GVH di 12 mesi **L. 1.100.000**

TURBO!!



K 4 T - Caratteristiche generali come il P 14 ma in versione Kit, da assemblare. Contenitore standard look AT tempo di montaggio ±2 ore. Con accessori ed istruzioni per il montaggio (in versione kit non viene fornita la scheda CX 20 o CX 25) **L. 899.000**

MONITOR



CDM 1200 - Video monocromatico 12", fosfori verdi (gn) o arancio (or) a scelta. Ingresso video composito, regolazione luminosità e contrasto, 256 caratteri diversi, 25 righe da 80 colonne, schermo antiriflesso inclinato per una più facile lettura, basso consumo, commutatore bassa/alta sensibilità. Definizione orizzontale e verticale: 1000 linee. **L. 199.000**



MD 3 - Video colori 14", 16 colori RGB, 14 MHz, ingresso TTL, risoluzione da 640 pixel orizzontale/200 linee verticale. Regolazione luminosità, colore e contrasto, 256 caratteri diversi, 25 righe da 80 colonne, tubo a 90x per una durata maggiore, 70 watt max di consumo, connettore a 9 pin. Commutatore per fosfori: verde/arancio/16 colori. **L. 699.000**

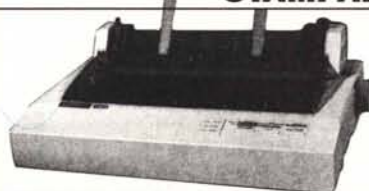


KLM 1211 - Center video monocromatico 12", ingresso TTL, presa a 9 pin compatibile IBM, banda passante 25 MHz. Risoluzione centrale 1100 linee. Risoluzione laterale 900 linee. Regolazione luminosità e contrasto. **L. 180.000**

PHILIPS CM 8833 - Video colori 14", con audio incorporato. Schermo antiriflesso. Due ingressi RGB (Scart) e RGBI, un ingresso video composito. Banda passante maggiore 12 MHz. Tubo a 90x. 16 colori riprodotti. Risoluzione orizz.: 600 pixels. Risoluzione vert.: 285 linee. Numero di caratteri: 2000 (80x25). **L. 649.000**

PHILIPS BM 7513 - Monitor a fosfori verdi 12", tubo a 90x. Schermo antiriflesso. Ingresso TTL. Risol. orizz.: 920 pix. Risol. vert.: 350 pix. Numero di caratteri: 2000 (80x25). **L. 189.000**

STAMPANTI



CITIZEN 120D - Stampante 80 colonne, velocità 120 CPS, NLQ 25 CPS, protocollo IBM/EPSON, interfaccia parallela standard Centronics compatibile intercambiabile ad innesto. Garanzia di 2 anni. **L. 570.000**



CITIZEN MSP 15 - Stampante Citizen ad aghi, 136 colonne, velocità 160 CPS, NLQ 40 CPS, protocollo IBM/EPSON. Profilo piatto. Alta affidabilità. **L. 1.100.000**

SCHEDE DI ESPANSIONE PER PC



CX 20 - Scheda grafica video monocromatico (Hercules) con connettore standard RGB/TTL. Provista di porta parallela per stampante. Risoluzione 720x348. **L. 153.900**

CX 25 - Scheda grafica video colori con connettore standard RGB e connettore con uscita videocomposita. Risoluzione 640x200 (bianco/nero), 320x200 (16 colori). **L. 153.900**

CX 26 - Scheda EGA (Enhanced Graphic Adapter). Adattatore per video colori ad alta risoluzione: 640x200 16 colori; 640x350 64 colori. Uscita parallela per stampante. **L. 480.000**

CX 30 - Scheda multi I/O, con 2 porte seriali (una sulla scheda), una porta parallela, orologio calendario; connettore per joystick. Cavetto per una porta seriale. **L. 140.600**

CX 40 - Scheda EPROM/PROM Writer con 4 porte. Programma le EPROM, con relativo software permette di verificare lo stato della EPROM, di visualizzare e/o modificare il contenuto, scrivere e caricare da buffer o da files su disco. Consente di programmare in sequenza 4 EPROM differenti. **L. 361.000**

CX 50 - Scheda seriale RS 232. Permette di collegare le periferiche con standard per comunicazioni; come MODEM, MOUSE, ecc. **L. 63.000**

CX 52 - Scheda Controller per 2+2 drive con cavo per 2 drive. **L. 68.000**

CX 70 - Scheda 576Kb RAM (senza RAM). Espans. di memoria RAM da 576Kb. **L. 69.500**

MB 4 - Main Board Turbo 640Kb (scheda madre), con 8 slot e 256Kb di memoria RAM già installati. Doppio clock 4,77-8 MHz. **L. 310.000**

PARTI STACCATE

LH 4 - Disk drive a trazione diretta 360K slim. Made in Japan. TEAC. **L. 218.000**

LH 5 - Disk drive a trazione diretta 360K slim. Made in Japan. Toshiba. **L. 235.000**

HD 20 - Hard disk 20MB; (NEC) controller Western digital. Garan. 1 anno. **L. 1.150.000**

HD 10 - Hard disk 10MB; (NEC), controller Western digital. Garan. 1 anno. **L. 950.000**

MS 808 - Joystick 3 pulsanti, potenziometro preset. **L. 32.000**

MP 303 - Modemphone 303. Bell 103/CCITT V21 compatibile, 300 BPS. Telefono incorporato, con uscita RS 232 per collegamento PC approvato norme FCC. **L. 199.000**

GM 4 - Genius Mouse, Endecoder ottici, per PC XT/AT compatibili; 3 pulsanti per il disegno, massima traccia disegnable 200 mm/sec, connettore: D-25P (standard). Applicazioni software: D base III, Multiplan, Wordstar, Autocad, ed altri programmi compatibili. **L. 185.000**

TASTIERA T-5060 - Pratica, ergonomica, con 10 tasti funzione. Gestita da microprocessore, con 84 tasti compatibile XT/AT, 3 led indicatori (Caps lock, Num lock, Scroll lock), cavo di collegamento flessibile, inclinazione regolabile, ripetizione automatica del tasto premuto. Comando soffice del tasto. Approvata norme FCC. **L. 110.000**

PX - Alimentatore da 150 Watt. Interruttore laterale. Alta affidabilità. **L. 134.000**

CX 4 - Contenitore per PC, tipo standard. Con coperchio apribile; con accessori. Look XT. Usato per la versione Kit. **L. 59.000**

CX 5 - Come sopra ma con look AT. **L. 65.000**

CX 6 - Come sopra ma in versione profess. Con coperchio a slitta. **L. 75.000**

ACCESSORI

CP 25 - Cavo per stampante parallela; lunghezza 1,8 mt. **L. 9.900**

CS 25 - Cavo per stampante seriale; lunghezza 1,8 mt. **L. 9.900**

SK 12 - Dischetti 5" 1/4, SS DD Bulk (minimo 100 pezzi). **L. 1.300**

SK 14 - Dischetti 5" 1/4, DS DD Bulk (minimo 100 pezzi). **L. 1.400**

NOVITA' IN ARRIVO - TELEFONATE!!

P 14 T MINI - Versione da tavolo occupa meno spazio con + accessori.

COMPACT - Portatile XT con schermo 9 pollici incorporato.

COMPAT - Compatibile AT. Main board baby portatile con monitor 9".

MODEM HAYES (R) - Compatibili CCITT 0±1.200 bps. Si monta su slot del P14-K4 o qualunque altro.

PREZZI IVA ESCLUSA

DISTRIBUTORI ESCLUSIVI DI ZONA: Bologna, BOTTEGA ELETTRONICA, Via Battistelli 6/C - Cerignola (FG), DISCOTECA OMNIA, Via Foligno 22/B - Forlì, PLAYER, Via F.lli Valpiani 6/A - Lecce, CAMEL, Via Marinosci 13 - Milano, CRC ITALIA, Via Dario Papa 4/1 - Modena, ELECTRONIC CENTER, Via Malagoli 36 - Napoli, POWER, C.so Secondigliano 397 - Pertosa (SA), E.C. computer, Via S. Maria delle Grazie - Salerno, ELETTRONICA HOBBY, Via L. Cacciatore 56 - GENERAL COMPUTER, C.so Garibaldi 56 - Treviso, ELB TELECOM, Via Montello 13 abc.

CENTRI ASSISTENZA TECNICA: Bari, AD SERVICE s.n.c., Via De Samuele Cagnazzi 12/A, Tel. 080/349902 - Bologna, GVH, Via Beverara 39, Tel. 051/370687 - TECNILAB, Via S. Croce 24, Tel. 051/236530 - Forlì, PLAYER, Via F.lli Valpiani 6/A, Tel. 0543/36300 - Milano, CRC ITALIA, Via Dario Papa 4/1, Tel. 02/6071515 - Napoli, AMATISTA & Co., Via G. Carata 4, Tel. 081/7804511 - Salerno, SACO ELETTRONICA, Via Maganario 65, Tel. 089/394901. Riparazioni entro 48 ore.