a cura di Fabio Marzocca

L'ASSEMBLER DEL PC-1500

Come più volte annunciato sui numeri scorsi, pubblichiamo questo mese i risultati a cui siamo pervenuti dopo lunghe ricerche fra i bit del microprocessore che svolge funzioni di CPU nel PC-1500. L'elenco delle istruzioni è ancora incompleto, sebbene con esso sia già possibile realizzare routine in linguaggio macchina di notevole complessità.

Siamo comunque certi che la pubblicazione del set parziale di codici stimolerà i lettori alla ricerca ed allo studio delle istruzioni mancanti pervenendo così, in breve tempo, alla stesura della tabella completa di questo "strano" assembler di cui sembra che la Sharp sia morbosamente gelosa.

Premessa

Come già detto sul nº 9 di MC, in occasione della prova del PC-1500, questo pocket dispone di un microprocessore a 8 bit dati e 16 bit indirizzi. Quindi la quantità di memoria direttamente indirizzabile raggiunge i 64 Kbyte; la parola di codice è lunga 8 bit e con le loro combinazioni è possibile realizzare istruzioni a 1, 2 o 3 byte.

Già durante la prova del PC-1500, rilevammo la presenza di comandi quali PEEK, POKE e CALL, chiaramente dedicati ad un uso del computer con il linguaggio macchina. Da questo punto sono partiti i lavori di ricerca i quali hanno portato alla scoperta di altre istruzioni, della gestione della memoria da parte del microprocessore, delle mappe in cui è divisa la ROM e, infine, dei codici operativi delle istruzioni assembler.

È appena il caso di ricordare i notevoli vantaggi della programmazione assembler. Operando in Basic, il computer è costretto ad interpretare una per una le istruzioni, traducendole in linguaggio macchina, e quindi eseguire una serie di routine e cicli che svolgono le operazioni indicate dall'istruzione con un notevole dispendio di tempo; in assembler, invece, l'esecuzione è rapidissima in quanto viene by-passata tutta la fase di interpretazione. Inoltre, con tale linguaggio è possibile operare direttamente sul contenuto di ogni locazione di memoria modificando, per esempio, una variabile utilizzata nel programma Basic, oppure intervenendo opportunamente sul registro del display o della stampante, e così via.

Dec.	Hex.	Mnemonico	Azione prodotta
000	00	ADD, A,$\overline{L}$	Acc = Acc + $\overline{L}$
2	2	ADD A,L	Acc = Acc + L
4	4	LD A, L	Acc←L
5	5	LD A, (HL)	Acc←(HL)
8	8	LD H, A	H←Acc
9	9	AND (HL)	Acc.←Acc. AND (HL)
10	A	LD L, A	L←Acc.
11	B	OR (HL)	Acc.←Acc. OR (HL)
13	D	XOR (HL)	Acc.←Acc. XOR (HL)
14	E	LD (HL), A	(HL)←Acc.
16	10	ADD A, $\overline{C}$	Acc.=Acc+$\overline{C}$
18	12	ADD A, C	Acc.=Acc.+C
20	14	LD A, C	Acc.←C
21	15	LD A, (BC)	Acc.←(BC)
24	18	LD B, A	B←Acc.
25	19	AND (BC)	Acc.←Acc. AND (BC)
26	1A	LD C, A	C←Acc.
27	1B	OR (BC)	Acc.←Acc. OR (BC)
29	1D	XOR (BC)	Acc.←Acc. XOR (BC)
30	1E	LD (BC), A	(BC)←Acc.
32	20	ADD A, $\overline{E}$	Acc.=Acc.+$\overline{E}$
34	22	ADD A, E	Acc.=Acc.+E
36	24	LD A, E	Acc.OE
37	25	LD A, (DE)	Acc.←(DE)
40	28	LD D, A	D←Acc.
41	29	AND (DE)	Acc.←Acc. AND (DE)
42	2A	LD E, A	E←Acc.
43	2B	OR (DE)	Acc.←Acc. OR (DE)
45	2D	XOR (DE)	Acc.←Acc. XOR (DE)
46	2E	LD (DE), A	(DE)←Acc.
64	40	INC L	L = L + 1
65	41	AINC HL	(HL)←Acc. HL=HL+1
66	42	DEC L	L = L - 1
67	43	ADEC HL	(HL)←Acc.; HL=HL-1
68	44	INC HL	HL = HL + 1
69	45	INCA HL	Acc.←(HL); HL=HL+1
70	46	DEC HL	HL = HL - 1
71	47	DECA HL	Acc.←(HL); HL=HL-1
72	48	LD H, data	Carica in H il byte dati successivo
74	4A	LD L, data	Carica in L il byte dati successivo
76	4C	CP H, data	Confronta con H il byte dati successivo
78	4E	CP, L data	Confronta con L il byte dati successivo
80	50	INC C	C = C + 1
81	51	AINC BC	(BC)←Acc.; BC=BC+1
82	52	DEC C	C = C - 1
83	53	ADEC BC	(BC)←Acc.; BC=BC-1
84	54	INC BC	BC = BC + 1
85	55	INCA BC	Acc.←(BC); BC=BC+1
86	56	DEC BC	BC = BC - 1
87	57	DECA BC	Acc.←(BC); BC=BC-1
88	58	LD B, data	Carica in B il byte dati successivo
90	5A	LD C, data	Carica in C il byte dati successivo
92	5C	CP B, data	Confronta con B il byte dati successivo
94	5E	CP C, data	Confronta con C il byte dati successivo
96	60	INC E	E = E + 1
97	61	AINC DE	(DE)←Acc.; DE=DE+1
98	62	DEC E	E = E - 1
99	63	ADEC E	(DE)←Acc.; DE=DE-1
100	64	INC DE	DE = DE + 1
101	65	INCA DE	Acc.←(DE); DE=DE+1
102	66	DEC DE	DE = DE - 1
103	67	DECA DE	Acc.←(DE); DE=DE-1
104	68	LD D, data	Carica in D il byte dati successivo
106	6A	LD E, data	Carica in E il byte dati successivo

Figura 1 - Il set parziale di istruzioni

È chiaro, a questo punto, quanto possa essere importante per un utilizzatore disporre dei codici macchina e della possibilità di poter far girare sul suo computer un programma in assembler.

Tecniche d'impiego e due nuove istruzioni Basic

Altre due nuove istruzioni Basic? Beh, ormai non è più una novità ed anche questa volta si tratta di 2 istruzioni particolarmente interessanti.

CSAVE M "program"; ind.1, ind. 2
CLOAD M "program"

Già dalla loro struttura se ne intuisce l'utilizzazione: si tratta infatti di due comandi che permettono la registrazione e la lettura da cassetta di un programma in linguaggio macchina. La parola tra virgolette rappresenta l'etichetta alfanumerica di individuazione della routine (non può essere omessa in CSAVE M), mentre "ind.1" e "ind. 2" sono gli indirizzi di inizio e fine della routine da salvare. Nella fase di CLOAD M la routine verrà caricata nelle stesse locazioni di memoria specificate da CSAVE M.

Vediamo ora quali sono le procedure da seguire per poter scrivere in linguaggio macchina. In primo luogo dovremo individuare una porzione di memoria libera per l'allocazione della routine: STATUS 2 rappresenta, in tal senso, l'indirizzo della prima locazione libera, mentre STATUS 3 contiene l'ultimo indirizzo libero; quindi STATUS 2 - STATUS 3 ci indica quanti byte abbiamo a disposizione per il nostro programma. Attenzione a non creare interferenze con il programma Basic: allungando infatti quest'ultimo c'è rischio di andare ad invadere la zona in cui è scritto il programma in linguaggio macchina. È consigliabile, perciò, allocare la nostra routine assembler sempre nelle ultime posizioni di memoria.

Per poter scrivere il programma in memoria, faremo uso dell'istruzione POKE addr, B1, B2,... dove "addr" è l'indirizzo di partenza e B1, B2, ecc. sono i byte delle istruzioni, i quali possono essere scritti in decimale oppure, se preceduti dal carattere &, in esadecimale. L'istruzione POKE può

```
5500: 58 00 5A 00
5504: 48 40 4A C5
5508: BE 55 20 94
550C: 41 14 41 05
5510: 2A 41 05 62
5514: 6E 00 99 07
5518: 41 05 2A 6E
551C: FF 99 17 9A
5520: 6A 09 15 51
5524: 62 93 05 9A
```

Figura 2 - Codice oggetto del programma Renumber.

scrivere contemporaneamente tanti byte fino a riempire il buffer d'ingresso (circa una ventina), ma conviene scriverne 10 alla volta, incrementando così di 10 in 10 l'indirizzo "addr".

Per lanciare l'esecuzione del programma, useremo invece l'istruzione

CALL addr.

dove "addr" è l'indirizzo della prima istruzione della routine.

Queste operazioni possono essere indif-

Dec.	Hex.	Mnemonico	Azione prodotta
108	6C	CP D, data	Confronta con D il byte dati successivo
110	6E	CP E, data	Confronta con E il byte dati successivo
128	80	ADD A, $\overline{H}$	Acc. = Acc. + $\overline{H}$
130	82	ADD A, H	Acc. = Acc. + H
132	84	LD A, H	Acc.←H
139	8B	FJNZ, data	Se l'operazione immediatamente precedente è diversa da 0, salta a *PC + data*
142	8E	FJMP, data	Salta a *PC + data*
144	90	ADD A, $\overline{B}$	Acc. = Acc. + $\overline{B}$
146	92	ADD A, B	Acc. = Acc. + B
147	93	BJME, data	Se il contenuto del registro E non è negativo, salta a *PC - data*
148	94	LD A, B	Acc.←B
153	99	BJNZ, data	Se l'operazione immediatamente precedente è diversa da 0, salta a *PC - data*
154	9A	RET	Ritorno incondizionato
158	9E	BJMP, data	Salta a *PC - data*
160	A0	ADD A, $\overline{D}$	Acc. = Acc. + D
161	A1	SBC A, addr	Acc. = Acc. - (addr.) - bit carry
162	A2	ADD A, D	Acc. = Acc. + D
163	A3	ADC A, addr	Acc. = Acc. + (addr) + bit carry
164	A4	LD A, D	Acc.←D
165	A5	LD A, addr	Acc.←(addr.)
167	A7	CP A, addr	Confronta il contenuto dell'accumulatore con il contenuto della locazione specificata dai due byte successivi (Hi, Lo)
169	A9	AND addr	Acc.←Acc. AND (addr.)
171	AB	OR addr	Acc.←Acc. OR (addr)
173	AD	XOR addr	Acc.←Acc. XOR (addr)
174	AE	LD addr, A	(addr)←Acc.
177	B1	SBC A, data	Acc. = Acc. - data - bit carry
179	B3	ADC A, data	Acc. = Acc. + data + bit carry
181	B5	LD A, data	Carica nell'accumulatore il byte dati successivo
183	B7	CP A, data	Confronta il byte dati successivo con l'accumulatore
185	B9	AND data	Acc.←Acc. AND data
186	BA	JMP addr	Salta alla locazione specificata dai due byte successivi
187	BB	OR data	Acc.←Acc. OR data
189	BD	XOR data	Acc.←Acc. XOR data
190	BE	CALL addr	Chiamata a subroutine incondizionata
209	D1	RRA	Shift a destra dei bit dell'accumulatore attraverso il carry
213	D5	RRCA	Shift a destra dei bit dell'accumulatore
217	D9	RLCA	Shift a sinistra dei bit dell'accumulatore
219	DB	RLA	Shift a sinistra dei bit dell'accumulatore attraverso il carry
221	DD	INC A	Acc. = Acc. + 1
222	DE	DEC A	Acc. = Acc. - 1
3	3	ADD A, (HL)	Acc. = Acc. + (HL)
19	13	ADD A, (BC)	Acc. = Acc. + (BC)
35	23	ADD A, (DE)	Acc. = Acc. + (DE)

ferentemente eseguite sia in modo PRO che in modo RUN.

Le routine che verranno scritte in linguaggio macchina, saranno inoltre incancellabili, a meno che non si sovrappongano ad esse nuovi codici, oppure il programma BASIC. Quindi, dopo aver scritto in memoria tramite la POKE, neanche un ALL RESET cancellerà le nostre routine.

Il set di istruzioni

In figura 1 è rappresentato il set parziale delle istruzioni, in cui è stato usato un codice mnemonico simile a quello dello Z-80.

Le istruzioni sono quasi tutte standard, contenute cioè anche in altri tipi di assembler: esamineremo perciò soltanto quei codici la cui definizione mnemonica è risultata influenzata dalla particolarità delle operazioni da essi eseguite.

Iniziamo con i codici di incremento e decremento di una coppia di registri. Per ogni coppia HL, BC, DE esistono 3 diversi tipi di incremento e decremento; vediamo quelli relativi ad HL con ovvia estensione anche a BC e DE:

AINC HL: prima di incrementare la coppia di registri HL, il contenuto dell'accumulatore viene caricato nella locazione di memoria specificata da HL. Può essere utile nei cicli, risparmiando per ogni ciclo l'istruzione LD (HL), A.

INCA HL: prima di incrementare la coppia di registri HL, viene caricato nell'accumulatore il contenuto della locazione indirizzata da HL.

INC HL: incrementa HL senza ulteriori operazioni.

Le stesse specifiche valgono per ADEC HL, DECA HL e DEC HL e per le altre due coppie di registri.

Le istruzioni indicate con *CP r, data* confrontano il contenuo del byte dati seguente con il registro r, senza modificare l'accumulatore o il registro, ma intervenendo semplicemente sui bit dei flag. Queste istruzioni si rivelano particolarmente utili come test di condizione per i cicli DO...UNTIL; il codice A7 indica invece che il confronto è eseguito fra l'accumulatore ed il contenuto della locazione di memoria specificata dai due byte seguenti.

Attenzione: contrariamente a quanto avviene per lo Z-80 ed altri microprocessori, quando si deve specificare un indirizzo (2 byte) bisognerà scrivere prima il byte HI, seguito da quello LO. Es.:

CP A &4A &C5

confronta il contenuto dell'accumulatore con il byte immagazzinato nella locazione &4AC5.

Le istruzioni di salto vanno analizzate con particolare cura. Per quanto riguarda JMP addr. e CALL addr., esse si comportano come le analoghe dello Z-80. Riferiamoci invece ora a BJME, BJNZ, FJNZ, FJMP, BJMP: in esse la lettera B sta per back (dietro) e F per forward (avanti), quindi l'istruzione:

BJNZ, data

```
&5500

58 00       LD B,0
5A 00       LD C,0
48 40       LD H,&40
4A C5       LD L,&C5
BE          CALL
55 20       &5520
94          LD A,B
41          AINC HL
14          LD A,C
41          AINC HL
05          LD A,(HL
2A          LD E,A
41          AINC HL
05          LD A,(HL
62          DEC E
6E 00       CP E,0
99 07       BJNZ, 7
41          AINC HL
05          LD A,(HL
2A          LD E,A
6E FF       CP E,&FF
99 17       BJNZ,23
9A          RET

&5520

6A 09       LD E,9
15          LD A,(BC
51          AINC BC
62          DEC E
93 05       BJME, 5
9A          RET
```

Figura 3 - Disassemblato del programma Renumber.

salterà alla locazione *PC-data* se il flag di zero non è settato. Per questi tipi di indirizzamento, occorrerà ricordare che quando il microprocessore analizza il byte "data", il suo program counter si trova già alla locazione successiva, e ciò va tenuto in conto nell'assegnazione del byte di salto.

FJMP e BJMP sono salti incondizionati, avanti o indietro, del numero di passi specificati dal byte successivo.

L'istruzione *BJME, data* è molto singolare: il suo significato è Back Jump Minus on register E, e provoca un salto di "data" passi indietro se il contenuto del registro E non è negativo.

Il programma Renumber

Il primo "impegno" che il nostro PC-1500 si è trovato a dover affrontare in linguaggio macchina è stato, ovviamente, una routine di renumber. Precisiamo subito che la routine in oggetto rinumera solo le righe, e non gli indirizzi degli statement GOTO, GOSUB e THEN: questo ostacolo può comunque essere facilmente aggirato etichettando opportunamente il programma ed indirizzando quindi i salti con assegnazioni del tipo:

GOSUB "etichetta"

In figura 2 è rappresentato il listing in linguaggio macchina da caricare nelle locazioni da &5500 a &5528 tramite l'istruzione POKE. Questi indirizzi sono riferiti alle macchine che hanno a disposizione l'espansione da 4K RAM, altrimenti sarà sufficiente rilocare la routine altrove, cambiando opportunamente l'indirizzo della CALL in essa contenuta. Con la CE-151 inserita, perciò, si hanno ancora a disposizione 5179 byte per il programma Basic, senza timore di sovrapposizioni; andando oltre questo limite, si provocherà l'alterazione della routine RENUMBER.

Dopo aver caricato il programma, quindi, basterà lanciare un CALL &5500 per veder rinumerare il nostro programma Basic quasi istantaneamente. Se non ci sono problemi di spazio in memoria, la routine può rimanere sempre caricata nella macchina ed essere richiamata all'occorrenza, in quanto, come già detto, è insensibile ai comandi NEW, NEW 0 e ALL RESET.

La tecnica di rinumerazione adottata è la stessa di quella descritta sul n° 14 di MC in occasione del renumber in Basic: si iniziano a caricare i valori 0 0 e 10 nelle locazioni 16581 e 16582 (con il modulo 8K RAM questi indirizzi vanno cambiati) per proseguire poi di 10 in 10 fino alla fine del programma.

Ricordiamo che questa routine occupa solo 40 byte, riducibili comunque nel caso in cui vengano scoperti nuovi codici che permettano operazioni più immediate. Volendo, infine, salvare su nastro il programma di linguaggio macchina, imposteremo le istruzioni:

CSAVE M "RENUMBER"; &5500, &5529

Conclusioni

Si apre così una nuova pagina sull'impiego del PC-1500: la programmazione in assembler. Vogliamo comunque precisare, per i puristi della materia, che in verità si tratta di "linguaggio macchina" e non "linguaggio assembler" in quanto vengono introdotti direttamente i codici esadecimali e non quelli mnemonici. La differenza risiede nel fatto che il processo di assemblaggio viene eseguito dall'operatore e non da un apposito programma assemblatore.

Continuiamo, in ogni modo, a non capire quali potrebbero essere stati i reconditi motivi che hanno spinto la Sharp a tenere nascosta questa eccezionale caratteristica del PC-1500 tenendo conto, fra l'altro, che in tempi più o meno lunghi sarebbe stato comunque decodificato il linguaggio macchina del microprocessore.

Perplessità a parte, invitiamo tutti i lettori ad approfondire le ricerche in merito visto che, inoltre, sembrerebbe che i primi a scoprire i codici delle istruzioni del PC-1500 siamo stati proprio noi, nel senso che non ci è finora capitato di leggere nulla di simile neppure su riviste estere. MC

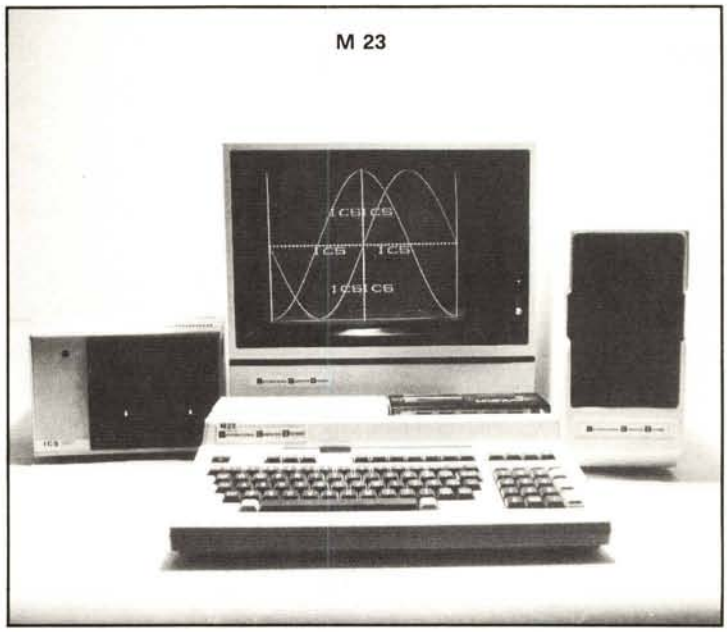
INTERNATIONAL COMPUTER SYSTEMS

Uffici di Roma: Via della Balduina, 85-89 – Tel. 34.81.85 - 34.92.760-660 – Telex 611091 CRMC Stabilimento: Via Nettunense, 49 – 00042 Anzio – Tel. 98.46.206

In Italia come in tutto il mondo la gamma dei nostri elaboratori sta ricevendo l'adesione degli esperti di informatica e degli utilizzatori. Per ragioni che sono le più valide: rigore tecnologico, fabbricazione professionale e sforzo costante di creare degli autentici sistemi di informatica al costo più basso. La International Computer Systems garantisce la distribuzione dei prodotti migliori direttamente dagli stabilimenti produttivi situati in Giappone, Irlanda, Italia.

M23 mark III – M23 mark V

Piccolo. Leggero. Potente.
Si impara a programmarlo in tre giorni!



Configurazioni a scelta con floppy da 5 o da 8 pollici monitor a fosfori verdi o a colori (RGB) da 14 pollici.
Scheda grafica a colori optionale.

Unità centrale
Un microprocessore ZILOG Z 80A con un clock a 4 MHz gestisce le risorse del sistema.
Un 2° micro APU effettua tutti i calcoli matematici.
Una memoria RAM da 128 Kbytes è a disposizione utente.
Due interfacce seriali RS232 programmabili e un'interfaccia parallela permettono il collegamento con l'esterno.
Questo insieme dà all'unità centrale la potenza richiesta per una larga gamma di applicazioni.

Unità minifloppy
Due minifloppy da 5'' (328 Kbytes ciascuno), semplice faccia, doppia densità, gestiti da un'interfaccia interna DMA (accesso diretto memoria).

Unità floppy 8''
Due Driver doppia faccia , doppia densità (1,1 MB ciascuno), con possibilità di formattazione in tutti i formati IBM.

Tastiera
Un blocco alfanumerico standard con maiuscole e minuscole.
Un blocco numerico separato con i comandi del cursore.
Un blocco di 14 funzioni programmabili.
Le sue numerose funzioni permettono una grande flessibilità di utilizzo.

Schermo
25 righe per 80 colonne maiuscole e minuscole in visione normale o ''negativa''.
32 caratteri semigrafici permettono la costruzione di tabelle o di grafici.

SYSTEM SOFTWARE

● Relocatable assembler ● Editor ● Debugger ● Relocatable loader ● Library file editor

● Subroutines in Assembler possono essere richiamate all'interno di programmi in BASIC o in Fortran ● EBASIC – Interprete esteso occupa circa 32 Kbytes ● CBASIC – Compilatore compatibile con Ebasic consente di aumentare di 5/6 volte la velocità di esecuzione ● MBASIC – A doppia precisione (13 cifre) per calcoli tecnici e matriciali ● TBASIC – Per trasmissione dati e collegamento con altri computers ● FORTRAN IV – Per calcoli tecnico-scientifici ● COBOL – Corrispondente a livello ANSI 74 ● UCSD PASCAL ● L'SGL è un linguaggio grafico che permette, eventualmente anche con monitor a colori, di eseguire disegni estremamente complessi utilizzando la libreria BASIC con delle subroutines per le funzioni più comuni.

Vasta scelta di software applicativo gestionale-scientifico

PIPS, un linguaggio facile da imparare, sfrutta al massimo le capacità della macchina

Il PIPS, software unico, sviluppato per uso gestionale, è molto più vicino alla mente umana dell'Assembler, del Fortran, del Basic. Il PIPS permette a tutti di usare un potente computer con facilità. Il PIPS lavora utilizzando oltre 100 comandi. La gestione dei dati avviene tramite la semplice selezione di questi comandi. Per ricercare dei dati si imposta il comando CS. Per sortare si imposta SORT. Per funzioni grafiche si imposta GR. E così via. Vari programmi e funzioni possono essere ottenute a seconda dell'ordine con cui si selezionano i comandi. Il PIPS elimina la necessità di programmi specialistici. Alcuni tipi di lavoro richiedono soltanto di digitare i comandi nel loro ordine, per ottenere i risultati richiesti!

M 243 – M 343 Una famiglia di micro da 8 e da 16 bit multiutente con multiprogrammazione

L'**M 243** e l'**M 343** sono il culmine di anni di esperienza combinati con la più sofisticata tecnologia. Sono microcomputers completamente nuovi che si adattano perfettamente ai più disparati tipi di applicazioni, Offrono possibilità di ampliamento in memoria centrale con schede; in memoria di massa con dischi floppy da 5'' e da 8'' e dischi rigidi Winchester. Oltre ad avere inserite interfacce di qualsiasi tipo e a poter essere utilizzati come terminali intelligenti di computers più potenti, sono dotati di uno schermo completamente grafico ad altissima definizione anche a colori e permettono la gestione di più posti dilavoro in multi-programmazione.

Unità Centrale
Un microprocessore a 8 bit Z80A gestisce le risorse del sistema nel M 243.
Un microprocessore a 16 bit 8086 è invece utilizzato nel modello M 343.
Un 2° processore logico effettua tutte le operazioni logiche sui numeri fino a 32 bit in virgole flottanti.
Un counter/timer programmabile da software controlla la successione delle operazioni.
Un orologio in tempo reale, con batteria tampone, fornisce la data e l'ora e permette di avviare, tra l'altro, dei programmatori ad ore prestabilire.
Una memoria RAM da 192 Kbytes a 1 Mbytes è a disposizione utente. Tale memoria consente la presenza di più posti lavorocompleti in multiprogrammazione.
Quattro canali seriali RS232 programmabili da 50 a 19.200 Baud e un canale parallelo permettono il collegamento con l'esterno.

M5 – Home Computer
Il micro più piccolo della nostra famiglia

Si collega al televisore a colori di casa ed ad un registratore a cassette

Unità centrale
Z 80A – RAM 4 k + 16 k video RAM espandibile con cassetta fino ad altri 32 k.
Uscita per stampante parallela.
Uscita per TV color.
Uscita per monitor e altoparlante.
Optionali n. 2 Joypads per video game.
Tastiera con 52 tasti a 4 funzioni (maiuscoli, minuscoli, istruzioni basic e semigrafica).
Cassetta elettroniche con basic, pips e vasta scelta di video games.

INSTALLAZIONE IN TUTTA ITALIA CON LE SEGUENTI PROCEDURE

- Contabilità generale magazzino fatturazione.
- Contabilità generale e semplificata per commercialisti.
- Contabilità generale a booking per Agenzie di Viaggi.
- Trattamento testi e mailing list merge universale.
- Contabilità finanziaria per scuole ed enti pubblici.
- Paghe e stipendi per scuole.
- Gestione magazzini componenti o ricambi.
- Gestione biblioteche.
- Gestione iscritti ordini professionali.
- Calcolo strutture per zone sismiche.
- Gestione laboratori di analisi cliniche.

STAMPANTI 80-132-220 COLONNE ANCHE GRAFICHE A MATRICE DI 9 AGHI ED A MARGHERITA.

PLOTTER A 8 COLORI.
CONVERTITORI ANALOGICI/DIGITALI E D/A.

Cercansi distributori per zone libere