a cura di Fabio Marzocca

Il comportamento tenuto dalla Sharp, nei confronti del suo più recente pupillo PC-1500, si sta dimostrando quantomeno singolare e misterioso: nella documentazione che accompagna questo computer, infatti si direbbe che il colosso industriale giapponese abbia fatto di tutto per nascondere all'utente le reali possibilità operative del PC-1500, non citando utili istruzioni invece presenti, e mantenendo un grosso velo di riserbo sul set di istruzioni del suo nuovo microprocessore. Per quanto riguarda le 6 istruzioni nascoste, ne abbiamo già parlato su MC n° 9; questo mese pubblichiamo alcune notizie molto utili, frutto di un'indagine profonda sulla ROM del PC-1500, ed un interessante renumber del lettore Luca Ridarelli di Roma. Invitiamo fra l'altro tutti i lettori ad inviarci le loro personali esperienze in merito a questo problema.

RENUMBER

di Luca Ridarelli (Roma)

Per comprendere a fondo il funzionamento del programma di Renumber riportato in figura 1 è necessario dare un'occhiata a come il PC-1500 gestisce la propria memoria e a questo proposito può essere di aiuto lo schema riassuntivo di figura 2. Gli indirizzi che vanno da $0000 a $4000 (esadecimale) sono lasciati liberi e sono probabilmente destinati all'espansione da 16K RAM annunciata recentemente dalla SHARP ma purtroppo non ancora disponibile sul mercato a causa del costo troppo elevato delle RAM HM6116LP impiegate nella PC-1500. La memoria RAM vera e propria parte invece dall'indirizzo $4000 per arrivare, nella versione senza espansioni, a $4800, in quella 4K a $5800 e in quella 8K a $6800. Le locazioni $7000 - $7FFF sono adibite alla gestione delle scritte sul display mentre le ROM del sistema operativo e del BASIC sono indirizzate da $BFFF fino a $BFFFF. Altri 16K di memoria e precisamente le locazioni $7FFF-$BFFF, hanno la funzione di 'agganciare' le ROM delle periferiche permettendo di espandere ulteriormente il sistema. Il BASIC del PC-1500, al pari dei sistemi operativi più avanzati, memorizza le linee di programma in forma condensata a partire dalla locazione $40C5 (e non da $4000 poiché quest'area è riservata alle funzioni dei tasti programmabili F1, F2 ecc.) in su. Una linea BASIC, come tutti sappiamo, viene immessa da tastiera nella forma:

10 PRINT "PC-1500" <ENTER>

dove 10 è il numero della linea, PRINT "PC-1500" l'istruzione e <ENTER> il tasto che autorizza l'immissione della linea in memoria. Al contrario di quanto si potrebbe pensare il computer non memorizza la linea in questa forma, ma la condensa e ritraduce allo scopo di risparmiare spazio e soprattutto tempo nella fase di compilazione in linguaggio macchina durante l'esecuzione del programma. Ciò significa, ad esempio, che la linea:

10 PRINT "PC" <ENTER>

viene trasformata dal PC-1500 in:

00 0A 07 F0 97 22 50 43 22 0D

I primi due byte vanno accoppiati e il numero risultante dà il numero di riga, in questo caso 000A = 10; il terzo indica dove trovare l'inizio della prossima linea e cioè dopo 7 byte esatti; il quarto e il quinto byte rappresentano il codice mediante il quale il computer riconosce l'istruzione PRINT (la lista dei codici relativi alle istruzioni del BASIC è ricavabile utilizzando il MINI-DEBUG presentato sul n. 11 di Microcomputer). I byte che seguono, tranne l'ultimo, costituiscono il testo che deve essere inviato al display e sono codificati in normalissimo ASCII. Il byte che chiude la linea è sempre presente e rappresenta il codice per il tasto ENTER.

Come si può vedere dal diagramma di flusso di figura 3, il programma di renumber fa uso esclusivamente dei primi tre byte di ogni linea e più precisamente riscrive i primi due, corrispondenti come già detto al numero di linea vero e proprio, e legge il terzo per localizzare e raggiungere la riga seguente. Nella riga 60160 troviamo prima di tutto un'etichetta "x" la quale ha la funzione di sostituire la noiosa procedura di lancio del programma del tipo RUN 60160 con un semplice DEF X; seguono, sulla stessa riga tre variabili M, H e L. "M" indica in quale locazione di memoria trovare l'inizio della prima linea di programma da rinumerare, in questo caso il valore è 16581 perché la PC-1500 comincia a scrivere sempre da questo punto in poi. "H" e "L" corrispondono al primo numero di riga e infatti la loro somma è H + L = 10. La linea 60165 si occupa della rinumerazione vera e propria scrivendo i numeri di riga H e L nella locazione M tramite un semplice POKE. Oltre a questo viene individuata, tramite la variabile I, la lunghezza in byte della linea rinumerata allo scopo di localizzare la linea successiva. La linea 60170 aumenta il valore delle variabili M, H e L per consentire la rinumerazione della linea seguente. La riga 60175 controlla che il programma non rinumeri se stesso verificando ogni volta che il numero di linea non superi 60160.

Al passo 60180 viene instaurato un loop necessario per assicurare un'esecuzione completa. In questo "renumber" sono state utilizzate esclusivamente variabili semplici del tipo "L" o "I" al posto di variabili composte come "INC" o "LO" poiché la PC-1500 per gestire variabili semplici non utilizza la memoria destinata ai programmi ma una speciale area mentre nel caso delle variabili composte è costretta ad "invadere" l'area normalmente utilizzata dal BASIC sottraendo byte preziosissimi ai pochi disponibili. Per concludere un suggerimento: per evitare che il programma di "renumber" crei delle linee con lo stesso numero o che addirittura rinumeri se stesso, non cambiate in alcun modo l'ordine o la numerazione delle sue linee.

```
60160 "X":M=16581:H=0:L=10

60165 POKE M,H,L:I=PEEK(M+2)

60170 M=M+I+3:L=L+10:IF L>255
      LET L=L-256:H=H+1

60175 IF PEEK M=235 END

60180 GOTO 60165
```

Figura 1

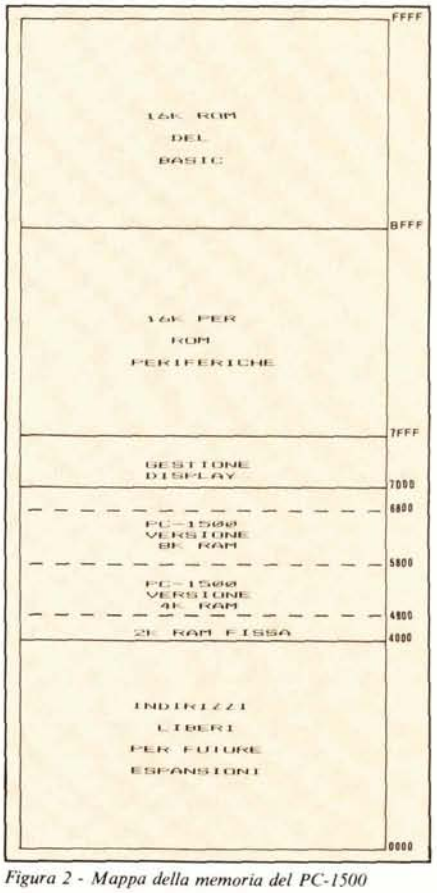

Figura 2 - Mappa della memoria del PC-1500

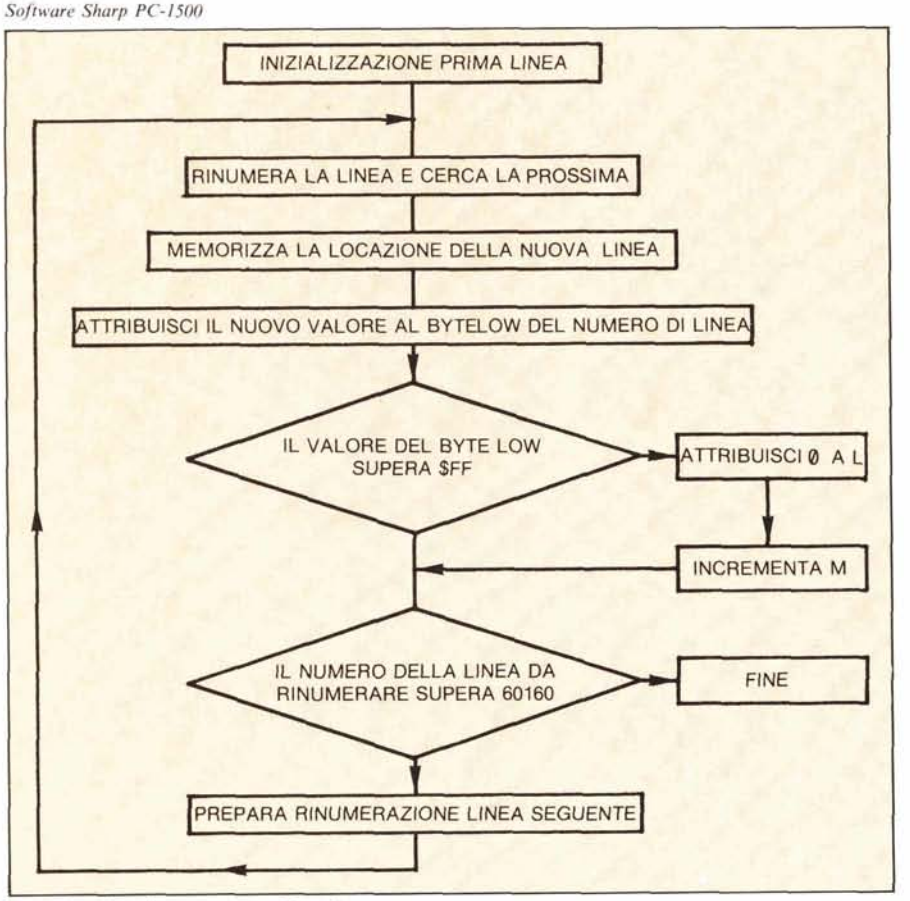

Figura 3 - Flow-chart programma Renumber

80	ASC	81	OR
88	MEM	91	TIME
92	INKEY$	93	PI
96	ASC	97	STR$
98	VAL	99	CHR$
100	LEN	101	DEG
102	DMS	103	STATUS
104	POINT	107	SQR
109	NOT	110	PEEK#
111	PEEK	112	ABS
113	INT	114	RIGHT$
115	ASN	116	ACS
117	ATN	118	LN
119	LOG	120	EXP
121	SGN	122	LEFT$
123	MID$	124	RND
125	SIN	126	COS
127	TAN	128	AREAD
129	ARUN	130	BEEP
131	CONT	132	CURSOR
133	USING	134	GRAD
135	CLEAR	136	CLS
138	CALL	139	DIM
140	DEGREE	141	DATA
142	END	144	LIST
145	INPUT	146	GOTO
147	GCURSOR	148	GOSUB
150	IF	151	PRINT
152	LET	153	RETURN
154	NEXT	155	NEW
156	ON	157	OPN
158	OFF	159	GPRINT
160	POKE#	161	POKE
162	PAUSE	164	RUN
165	FOR	166	READ
167	RESTORE	168	RANDOM
170	RADIAN	171	REM
172	STOP	173	STEP
174	THEN	175	TRON
176	TROFF	177	TO
179	WAIT	180	ERROR
181	LOCK	182	UNLOCK

Figura 4 - Elenco codici istruzioni del PC-1500 (configurazione senza stampante)

INTERPRETANDO L'INTERPRETE...

L'interprete Basic all'interno del PC-1500 è situato a partire dalla locazione $C000 (49152 decimale) ed occupa complessivamente 16 Kbyte. Negli 8K che vanno invece da A000 a BFFF viene invece caricato, al momento dell'accensione, il programma residente nella ROM della stampante. Dall'analisi di alcune parti di questo programma interprete siamo giunti a scoprire gli indirizzi di qualche interessante routine. Alla locazione $D080, ad esempio, si trova la routine che permette l'azzeramento di tutte le variabili; si tratta, infatti, delle istruzioni macchina che vengono eseguite per il comando CLEAR. Per verificare ciò, sarà sufficiente digitare CALL &D080, e controllare quindi il contenuto delle variabili. All'indirizzo $D00D invece inizia una routine che viene utilizzata dai comandi RUN e NEW: si tratta di una routine che provvede alla inizializzazione a zero del program counter (STATUS 4÷255).

Il programma interprete è stato inoltre utile nella ricerca del codice macchina del microprocessore a 8 bit che svolge le funzioni di CPU nel PC-1500. Nel momento in cui scriviamo, i codici per i quali abbiamo un'interpretazione certa sono i seguenti: 6F, BE, 9A.

Il primo fra questi, 6F (111 in decimale) è forse anche il più anomalo; è un'istruzione a 3 byte e si rappresenta nella forma:

<B1>...6F, <B3>

nella quale <B1> deve essere sempre la prima istruzione della routine chiamata. L'effetto di questo codice è quello di sommare B3 a B1 ogni volta che l'istruzione viene eseguita. Vediamo un esempio. Dopo aver dato un NEW, provate a digitare la seguente linea:

POKE 18000, 0, 111, 2, 154 <ENTER>

In questo modo abbiamo registrato, nelle locazioni da 18000 a 18003, un breve programma in linguaggio macchina (sull'istruzione 154 torneremo dopo). Facciamo ora eseguire al PC-1500 le quattro istruzioni, richiamando la routine:

CALL 18000

e andiamo a vedere cos'è successo in memoria:

PEEK 18000

All'indirizzo 18000 ci sarà ora 2, mentre le altre locazioni saranno rimaste invariate. Dando ora un altro CALL 18000 verificheremo che il contenuto di 18000 sarà diventato 4, e così via. Praticamente questa istruzione può venire impiegata in un ciclo LOOP...UNTIL, nel quale B1 rappresenta la variabile di controllo e B3 lo STEP del ciclo.

Il codice BE (190 decimale) rappresenta invece l'istruzione di chiamata a subroutine in modo incondizionato (CALL). È anch'essa un'istruzione a 3 byte, operante nella forma:

BE, <B2>, <B3>

in cui B2 rappresenta il byte d'indirizzo HI della locazione chiamata, e B3 il byte LO. Verifichiamo anche questo codice con un esempio. Abbiamo detto che all'indirizzo D080 è presente la routine che esegue il comando CLEAR; immettiamo perciò il seguente programma:

POKE 18000, &BE, &D0, &80, 154 <ENTER>

Carichiamo ora qualche dato nelle variabili: A=1, B=2, C=3 e lanciamo il programma in linguaggio macchina:

CALL 18000

Andando ora ad esaminare il contenuto delle variabili A, B, e C si noterà che queste sono state tutte azzerate; l'istruzione BE ha infatti provveduto a richiamare la subroutine D080, che equivale al comando CLEAR.

A questo punto avrete ormai capito che il codice 9A (154 decimale) corrisponde ad un ritorno incondizionato da subroutine (RET): i vostri programmi in linguaggio macchina, perciò, dovranno tutti terminare con questa istruzione.