i trucchi del CP/M

a cura di Claudio Rosazza

Basic & Assembler
II parte

Nella scorsa puntata abbiamo analizzato i problemi che sorgono nel caso si vogliano realizzare programmi applicativi misti, cioè costituiti da nuclei in linguaggio evoluto, nel caso particolare Basic, e subroutine Assembler.

Il problema è stato affrontato tenendo in considerazione l'uso del compilatore Basic della Microsoft che consente di produrre dei moduli rilocabili direttamente compatibili con quelli prodotti dall'assemblatore e linkabili assieme attraverso il linker senza particolari problemi.

Avevamo accennato che il problema si sarebbe complicato se avessimo usato anziché il compilatore, l'interprete Basic.

In effetti essendo l'interprete Basic un .COM, cioè un programma direttamente eseguibile e già definito in tutte le sue parti sia come tipo di istruzioni macchina, sia come locazioni di lavoro, non si può fare affidamento sul Linker per l'accorpamento dinamico ed automatico dell'eventuale subroutine Assembler.

Quest'ultima, infatti, andrà realizzata in modo da poter girare in una ben determinata area di memoria e dovrà essere caricata precedentemente all'interprete in modo tale che il Basic possa fare le dovute CALL a delle locazioni determinate trovando la subroutine in questione.

A questo punto occorre analizzare preventivamente l'organizzazione di memoria di un computer utilizzante il sistema operativo CP/M, in modo da poter decidere la posizione di memoria migliore dove poter allocare la subroutine Assembler.

In figura 1 viene schematizzata l'organizzazione di memoria tipica di un calcolatore che utilizzi il CP/M ed abbia 64K di Ram disponibili. In alto sono raffigurati gli indirizzi bassi della memoria crescenti verso il basso fino all'indirizzo FFFFH che corrisponde alla massima locazione di lavoro indirizzabile dallo Z80.

Partendo dalla locazione 0 troviamo dapprima un'area di 255 locazioni e cioè FF in esadecimale, denominata Page 0.

Questa area è riservata al CP/M e viene principalmente usata per la memorizzazione di alcuni parametri interni al sistema operativo e come dispositivo di interfacciamento software verso i programmi che fanno uso delle funzioni interne del CP/M.

A partire dalla locazione 100H fino ad una locazione denominata simbolicamente TPA troviamo uno spazio di memoria definito come Work-Area. In quest'area di memoria vengono collocati i programmi che si intendono far girare su quel calcolatore.

Di seguito a partire dalla locazione denominata TPA alla locazione denominata BIOS troviamo un'area di memoria occupata da un nucleo chiamato BDOS.

Questa è la parte essenziale del sistema operativo CP/M e contiene in forma indipendente dall'hardware tutte le funzioni intrinseche del CP/M stesso.

Infine, dalla locazione denominata BIOS fino alla massima locazione disponibile troviamo un'area occupata dal BIOS che è il nucleo hardware dipendente del CP/M; quello in pratica che consente di poter adattare il sistema operativo CP/M su hardware diversi. Nel caso che il calcolatore abbia meno di 64K Ram disponibili, il Bios finirà ugualmente alla massima locazione disponibile e la minor memoria disponibile andrà a scapito della Work-Area descritta in precedenza che sarà ovviamente più piccola.

```
        .Z80
        ASEG
        ORG     100H
;
CONIO   EQU     6
LIST    EQU     5
BDOS    EQU     5
;
BEGIN   EQU     ØAØØØ   (da controllare in
                        funzione del parti-
                        colare tipo di cal-
                        colatore)
;
BLKTRF: LD      HL,10EH
        LD      DE,BEGIN
        LD      HL,JEX-ZUBØ
;
        LDIR
        JP      Ø
;
        .PHASE  BEGIN
;
ZUBØ:   LD      E,(HL)
        LD      C,CONIO
        CALL    BDOS
        RET
;
ZUB1:   PUSH    HL
ZUB1B:  LD      E,ØFFH
        LD      C,CONIO
        CALL    BDOS
        CP      Ø
        JR      Z,ZUB1B
        LD      E,Ø
        LD      D,A
        POP     HL
        LD      (HL),D
        INC     HL
        LD      (HL),E
        RET
;
ZUB2:   LD      E,(HL)
        LD      C,LIST
        CALL    BDOS
        RET
;
JEX     EQU     $
;
        END
```

Listato Assembler delle subroutine di I/O per il terminale e la stampante.

Esiste inoltre un modulo denominato CCP e facente parte del CP/M che viene allocato nella Work-Area. Tale modulo provvede all'interfacciamento verso la console quando il calcolatore si trova ad operare sotto comandi impliciti CP/M e cioè in A>. Il motivo per cui viene allocato nella Work-Area è dato dal fatto che il CCP è necessario fin tanto che ci si trova nell'ambito dei comandi CP/M; non appena si entra in un programma applicativo sia esso interpretato o compilato il CCP non è più necessario e l'area da esso occupato può essere effettivamente utilizzata come Work-Area. C'è da tener presente che il CCP viene automaticamente ricaricato nella sua locazione di lavoro ogni volta che si fa ritorno al CP/M entrando in A> e cioè ogni volta che il programma applicativo restituisce il controllo al CP/M ed ogni volta che si forza tale operazione da CP/M digitando C.

Nel momento in cui trovandoci in A> digitiamo Mbasic con l'intento di caricare in memoria l'interprete Basic, il CP/M provvede a leggere il file su disco contenente l'interprete e lo carica in Ram nella Work-Area a partire dalla locazione 100H. L'area rimanente fino alla locazione TPA viene denominata Work-Area-Basic ed è effettivamente l'area disponibile per l'utente per i programmi e i dati; il messaggio iniziale del Basic indicante un certo numero di Byte Free corrisponde appunto alla grandezza della Work-Area-Basic disponibile per l'utente.

Dalla figura 2 risulta la situazione della memoria dopo aver caricato il Basic interprete e risulta evidente che un eventuale inserimento di una subroutine Assembler non può essere fatto altro che nella Work-Area-Basic.

E qui sorgono i primi problemi in quanto dovremmo infilare un programma in codice macchina, risultato dell'assemblatore, in un'area che il Basic stesso si riserva per sé come area di lavoro e considera quindi intoccabile ed intoccata dall'esterno.

A questo punto cerchiamo di capire come il Basic riesce ad individuare la grandezza della sua Work-Area.

La chiave di Volta si trova in Page 0 ed in particolare nelle locazioni 6 e 7 dove il CP/M tiene memorizzato l'indirizzo della locazione denominata TPA. Il meccanismo è a questo punto semplice: dopo che il CP/M ha caricato il Basic nella Work-Area effettua un Jump alla locazione 100H

dove l'interprete inizia una serie di setting interni fra i quali quello corrispondente alla lettura delle locazioni 6 e 7 e successiva sottrazione del massimo indirizzo occupato dal Basic stesso; il risultato è precisamente la grandezza della Work-Area-Basic.

È possibile, però, forzare il Basic a non leggere le locazioni 6 e 7, ma a prendere come TPA una locazione fornita dall'utente tramite un comando od una istruzione di programma.

Il comando che forza l'interpretazione esterna del TPA corrisponde a /M:&HXX XX digitato di seguito al richiamo del Basic in ambiente CP/M, dove XXXX è la locazione che si desidera far interpretare al Basic come TPA.

Digitando quindi MBASIC /M:&HA000

si forzerà il Basic ad interpretare come TPA l'indirizzo A000 espresso in esadecimale.

La corrispondente forzatura può essere fatta anche all'interno di un programma con la istruzione CLEAR nel modo seguente

10 CLEAR, &HA000

Operando dopo questa istruzione un PRINT FRE (0) si noterà la differenza nella segnalazione della memoria disponibile per l'utente. Appare ora evidente che operando una tale forzatura si rende possibile l'inserimento di una subroutine Assembler nella parte alta (come locazioni di memoria) della Work-Area-Basic in quanto dopo lo spostamento del TPA a conoscenza del Basic l'area compresa fra questo punto e il TPA effettivo non viene né riconosciuta né alterata da parte dell'interprete. Ma qui sorge un secondo problema e ce lo crea il CCP in quanto è caricato nella parte alta della Work-Area immediatamente a ridosso del TPA. Ora ipotizzando di caricare anche la subroutine Assembler a ridosso del TPA per lasciare al Basic successivamente il massimo di Work-Area possibile, è necessario in ogni caso dopo il caricamento della subroutine Assembler restituire il controllo al CP/M per consentire all'utente di richiamare il Basic. Questo, però, comporta il restore automatico da parte del CP/M del CCP nell'area ad esso assegnata con conseguente distruzione della subroutine Assembler. Risulta quindi evidente che l'unico modo per poter allocare le subroutine Assembler senza problemi è quello di porle nella Work-Area al di sotto del CCP e di forzare il Basic a modificare il suo TPA ponendolo al disotto delle subroutine Assembler (fig. 3).

Per poter conoscere il TPA della macchina sul quale si sta lavorando occorre usare un semplice programma Basic (non fatevi venire la voglia di andare a guardare le locazioni 6 e 7 con il DDT perché il DDT modifica il contenuto di queste locazioni ripristinandolo solo quando restituisce il controllo al CP/M):

```
10 X1%=PEEK (6)
20 X2%=PEEK (7)
30 X=(X2% *256)+X1%
40 PRINT HEX (X)
```

Il numero che questo programma visualizza sullo schermo è il TPA di quella particolare macchina sulla quale si è fatto girare il programma ed è espresso in notazione esadecimale. La prima locazione usata dal CCP è data dalla differenza fra il TPA e la lunghezza del CCP che per il CP/M 2.2 è di 806 espresso in esadecimale. Risulta quindi chiaro che l'ultima locazione utilizzata dalle subroutine Assembler dovrà essere al di sotto della prima locazione del CCP; il TPA interno del Basic si forzerà quindi al di sotto della prima locazione utilizzata dalle subroutine Assembler.

Come esempio di lavoro riportiamo il listato delle subroutine usate la volta scorsa ovviamente adattate per l'uso con l'interprete. Si nota una struttura sostanzialmente identica salvo l'aggiunta di una routine iniziale per lo spostamento del pacchetto nella locazione di lavoro assegnata. Occorre precisare che la costante BEGIN indica la locazione di lavoro delle subroutine Assembler e va calcolata in modo tale che l'ultima locazione della subroutine sia al disotto della prima locazione del CCP. A questo proposito occorre per tentativi assemblare ed andare a controllare nel punto PRN che l'ultima locazione o meglio JEX sia inferiore al valore calcolato da TPA-806H.

Il comando per assemblare è quello descritto nella precedente puntata e cioè dato il sorgente Assembler di nome SUB

M80 SUB, SUB=SUB (return)

Raggiunta la condizione di lavoro di cui sopra si provvederà a trasformare il modulo rilocabile .REL in un modulo eseguibile .COM mediante il programma L80 con il seguente comando

L80 SUB, SUB/N/E (return)

che produrrà un file SUB.COM

Figura 1

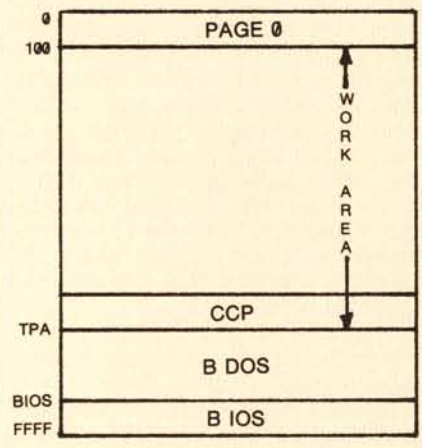


Figura 2

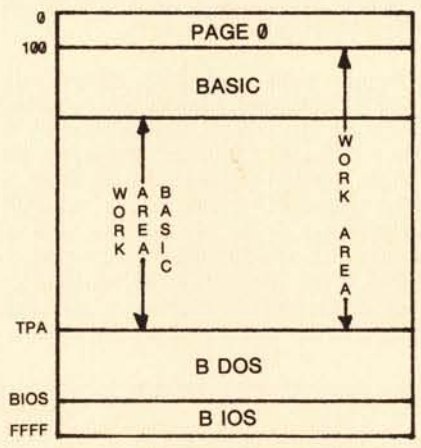
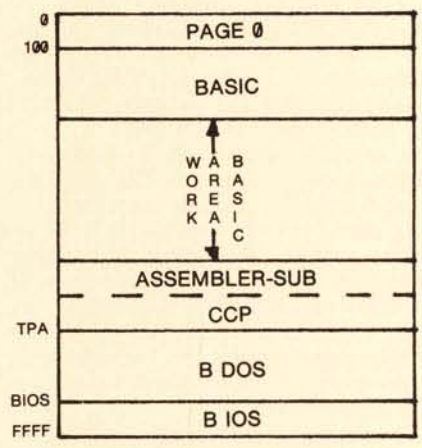


Figura 3



Richiamando questo file come un programma direttamente dal CP/M esso verrà caricato in Ram nelle locazioni da noi scelte e quindi il controllo ripassa al CP/M tornando in A> senza apparentemente aver prodotto nulla.

A questo punto richiamando il Basic forzando con /M: HXXXX il TPA interno siamo pronti per utilizzare allo stesso modo che con il compilatore le subroutine in Assembler.

Esiste comunque un ulteriore problema. Nell'uso dell'interprete effettuare una CALL ZUB1 (ZZ%) senza aver prima definito il valore di ZUBI comporta un ritorno immediato al CP/M. I valori di ZUB1, 2,3 sono le locazioni di ingresso delle rispettive subroutine in Assembler e vanno ricavati dal file SUB.PRN ed inserite nel programma Basic nel seguente modo:

```
1 DEFINT Z : ZUB0 = HYYYY:ZUB1 =
HYYYY:ZUB2 = HYYYY
```

dove YYYY vanno sostituiti con le locazioni di ingresso alle subroutine espresse in esadecimale.

Nella prossima ed ultima puntata riguardante il Basic e l'Assembler adotteremo un ulteriore "trucco" per poter utilizzare come Work-Area-Basic anche lo spazio del CCP altrimenti perduto. MC