i trucchi del CP/M

a cura di Claudio Rosazza

Basic & Assembler

I^a parte

Stilare programmi applicativi avvalendosi di linguaggi evoluti, siano essi interpreti o compilatori, significa quasi sempre scendere ad un compromesso.

Il compromesso consiste nell'accettare una maggiore flessibilità e versatilità di programmazione unita ad una buona dose di diagnostica sintattica e logica, a scapito ovviamente dei restanti elementi caratterizzanti un programma: velocità ed occupazione di memoria.

Scrivere, quindi, un programma in Basic significa fruire dei vantaggi di cui sopra a scapito della velocità di esecuzione e dell'efficienza in termini di occupazione rispetto ad un programma scritto in Assembler.

Scrivere, però, un programma in Assembler richiede molta esperienza e molto più tempo sia per la stesura organica del programma, sia per il debug dello stesso.

In alcuni casi è utile realizzare dei sistemi misti in cui le parti di programma la cui caratteristica peculiare richiesta è la velocità vengono scritte in Assembler ed il resto della procedura in linguaggio evoluto.

A volte, però, non è solo l'esigenza di velocità che spinge a scrivere parte della procedura in Assembler, ma anche l'esigenza di dover bypassare certe funzioni interne del linguaggio evoluto non desiderate.

Per esempio nei Basic Microsoft esistono alcune fastidiose funzioni di filtro nel trattamento dei dati da e verso la console e verso la stampante; nel Basic interprete 4.51, per esempio, esiste un maledetto contatore interno incrementato volta per volta dall'invio di caratteri verso la console che quando arriva al valore definito nell'istruzione WIDTH, che stabilisce la larghezza massima del video in numero di caratteri, automaticamente provoca l'invio di una sequenza CR-LF verso la console provocando lo spostamento del cursore all'inizio della riga successiva. Il contatore viene azzerato solamente in questo caso o nel caso si forzi il ritorno a capo del cursore con un PRINT non terminato da punto e virgola.

Tutto ciò è teoricamente corretto ed in certi casi anche utile poiché consente lo sfruttamento delle funzioni TAB e POS, ma cade pesantemente in crisi nel caso che il nostro calcolatore abbia un terminale staccato dalla CPU, o comunque in tutti quei casi in cui il Basic a nostra disposizione non abbia delle funzioni intrinseche per l'indirizzamento del cursore sul video e per lo svolgimento di particolari funzioni quali cancellazioni, inserimenti di righe o caratteri Etc.

In quest'ultimo caso, infatti, l'invio di caratteri di controllo verso il terminale tramite, per esempio, l'istruzione PRINT CHR$, provoca l'esecuzione della funzione prescelta sul video, ma il Basic incrementa il suo contatore di posizione in funzione del numero di caratteri inviati, registrando una situazione in molti casi non corrispondente alla realtà in quanto, per esempio, quella sequenza di caratteri inviati al terminale ha provocato lo spostamento del cursore in un'altra posizione dello schermo.

In questa situazione scrivendo stringhe sul video senza forzare il ritorno a capo e spostando continuamente il cursore utilizzando le dovute sequenze di controllo, scopriremo che ad un certo punto il Basic d'arbitrio forzerà il ritorno a capo del cursore, magari spezzando una parola, perché in quel momento il famigerato contatore ha raggiunto il valore massimo consentito. Nella versione 5.0 e successive del Basic Microsoft tale inconveniente è stato eliminato dando la possibilità all'utente di definire tramite l'istruzione WIDTH un valore di larghezza pari a 255. Tale valore viene interpretato all'interno del Basic stesso come infinito ed il contatore viene azzerato solamente in caso di forzamento volontario a capo del cursore da parte del programma.

Rimane comunque il problema che il Basic, quale ne sia la versione, opera alcune funzioni di filtro nei confronti di alcuni caratteri particolari quali HT e FF ed altri, che spesso possono dare seri problemi, a volte insormontabili, nelle gestione di periferiche esterne al sistema.

Occorre precisare che il discorso fin qui tenuto vale, seppur con problematiche diverse, anche nei confronti della stampante.

In questa prima parte di trattazione, prendendo spunto dall'esigenza di bypassare le funzioni interne del Basic descritte prima, esporremo la realizzazione di una subroutine in Assembler che consenta di realizzare le seguenti funzioni:

A - Invio di un carattere al terminale
B - Lettura di un carattere da tastiera
C - Invio di un carattere alla stampante.

In questa sede analizzeremo l'interfacciamento software fra programmi Basic compilati e subroutine Assembler, mentre nella prossima puntata analizzeremo il problema con il Basic interprete; paradossalmente, infatti, il problema è più facilmente affrontabile con il compilatore e più tardi capirete anche il perché. Il collegamento fra un programma Basic ed una subroutine Assembler viene realizzato tramite l'istruzione CALL.

Tale istruzione, nel caso del compilatore, deve essere seguita da un nome di massimo sei lettere che sarà il nome da riportare come label iniziale nella subroutine Assembler.

L'istruzione CALL prevede di essere utilizzata in due modi: il primo si riferisce al richiamo di una subroutine Assembler senza il passaggio bidirezionale di dati ed il secondo con passaggio reciproco di dati.

Nel primo caso la subroutine dovrà svolgere una funzione che non necessita di un passaggio di dati, bensì il fatto stesso di richiamare "quella" subroutine ne definisce univocamente tutti i parametri. Ad esempio se vogliamo realizzare una subroutine che cancelli l'intero video ed ammettendo che il carattere di controllo che svolga questa funzione sia FF (valore ASCII pari a 12) sarà sufficiente realizzare una subroutine chiamata ad esempio CS che spedisca semplicemente verso il terminale il carattere di valore 12 ritornando il controllo poi al Basic. Operando da Basic in qualsiasi punto del programma l'istruzione:

```
CALL CS
```

verrà svolta la suddetta funzione senza necessità di passaggio di dati poiché tutti i

parametri sono già stati definiti all'interno della subroutine stessa.

Nel caso che sia invece necessario spedire un particolare carattere verso il terminale deciso all'interno del Basic la CALL, oltre che chiamare la corrispondente subroutine, dovrà passare anche il carattere che intende spedire verso il terminale.

Il passaggio dei parametri avviene automaticamente da parte del Basic utilizzando i registri interni dello Z80 ed in particolare nel caso che il parametro sia uno solo il registro utilizzato è HL. Inserendo nel programma l'istruzione:

CALL COUT (ZZ%)

dove COUT è il nome della subroutine che provvede a spedire un carattere verso il terminale e ZZ% è una variabile intera contenente il valore ASCII del carattere da inviare; il Basic passerà alla subroutine Assembler nel registro HL l'indirizzo di memoria dove è memorizzato il contenuto di ZZ%.

Per produrre una subroutine in Assembler occorre avere a disposizione un Editor per poter scrivere il programma sotto forma di testo ASCII, un Assemblatore che generi il modulo cosiddetto rilocabile ed un linker che generi il modulo finale incorporando il programma principale scritto in Basic con la parte in Assembler.

Per quanto riguarda l'Editor va bene un qualsiasi Editor, possibilmente full screen; se usate il Word-Star create i vostri programmi con il comando N. Per quanto riguarda l'assemblatore ed il linker vanno bene quelli forniti dalla Microsoft con il compilatore Basic e cioè l'M80 e l'L80.

A questo punto dovrebbe risultare chiaro il perché sia facile generare programmi misti Basic-Assembler se la parte in Basic risulta compilata.

Sia il Bascom, infatti, sia l'M80 producono dei moduli che utilizzano un particolare codice detto rilocabile, dove ogni istruzione macchina Z80 è già definita dal punto funzionale, ma non riguardo alla locazione di lavoro. Il Linker provvede a mescolare i moduli rilocabili producendo un codice oggetto direttamente seguibile.

Val la pena di precisare che il Linker tratta i moduli rilocabili allo stesso modo sia che provengano da una compilazione Basic sia da un programma Assembler.

Il programma Assembler di seguito riportato contiene tre subroutine denominate rispettivamente ZUBØ, ZUB1, ZUB2.

ZUBØ provvede all'invio di un carattere verso il terminale, ZUB1 provvede alla lettura di un carattere da tastiera e ZUB2 provvede all'invio di un carattere verso la stampante.

Tutte e tre le subroutine prevedono l'utilizzo di una variabile Basic intera per il passaggio dei dati.

Dopo aver redatto il testo del programma ed essersi accertati che l'END finale sia terminato con un Return, si può uscire dall'editor salvando il file sotto forma di file ASCII con nome SUB.MAC.

```
;
        .Z80
        CSEG
        ORG     100H
;
        PUBLIC  ZUBØ,ZUB1,ZUB2
;
;
CONIO   EQU     6
LIST    EQU     5
BDOS    EQU     5
;
;
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
;
        END
```

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

A questo punto si provvede alla generazione del .REL con il seguente comando:
M80 SUB, SUB=SUB < return >
dove M80 è M80. COM e SUB è il testo del programma con nome SUB.MAC. Dopo qualche decina di secondi di lavoro su disco l'M80 replicherà con il messaggio NO Fatal Error(s); in caso contrario ricontrollate attentamente la stesura del testo.

Dando il precedente comando l'M80 genera due nuovi file denominati rispettivamente SUB.REL e SUB.PRN.

Mentre SUB.REL non è ispezionabile in nessun modo poiché trattasi del modulo da linkare assieme al prodotto della compilazione Basic, il file SUB.PRN contiene sotto forma di testo il programma originale con a fianco la transcodifica in codice macchina Z80; inoltre in caso di segnalazione di errori contiene le evidenziazioni delle righe nelle quali sono stati ravvisati errori sintattici o incongruenze logiche.

Ammettendo di avere il programma in Basic di nome PROG.BAS, dopo averlo salvato con l'opzione A e cioè:

SAVE "PROG.BAS", A <RETURN>

si procede alla compilazione dando l'usuale comando:
BASCOM PROG, PROG=PROG/N/Z <return>
dove Bascom è il compilatore Basic e Prog è il programma da compilare.

Il compilatore genererà un modulo rilocabile di nome PROG.REL che potrà essere associato a quello in Assembler con il comando:

L80 PROG,SUB,PROG/N/E " <return>

Il linker produrrà un modulo direttamente eseguibile denominato PROG. COM.

L'uso delle subroutine è molto semplice e richiede l'utilizzo di una variabile intera di comodo per il passaggio dei dati.

Riferendoci a quanto detto prima, se nella variabile ZZ% associamo il valore decimale 65 e formiamo la seguente istruzione Basic:

CALL ZUBØ (ZZ%)

il carattere con valore decimale ASCII 65, e cioè la A verrà inviato verso il teminale, diversamente se generiamo l'istruzione:

CALL ZUB2 (ZZ%)

l'effetto sarà identico tranne per il fatto che il carattere anziché verso il terminale verrà inviato verso la stampante.

Operando, invece, l'istruzione:

CALL ZUB1 (ZZ%)

la subroutine Assembler restituirà il controllo al Basic solo dopo che sia stato premuto un tasto sulla tastiera e contemporaneamente al ritorno in Basic in ZZ% troveremo il codice ASCII del carattere premuto.

Occorre precisare che queste funzioni bypassando completamente il Basic oltre ad evitare manipolazioni sui caratteri sia in uscita sia in ingresso da parte del Basic stesso, non alterano lo stato dei contatori interni di posizione del Basic e non consentono al Basic l'interpretazione sia dei caratteri in ingresso sia in uscita. Ciò si traduce ad esempio nell'impossibilità di riconoscere il ^C se accettato tramite la subroutine ZUB1.

Un programma del genere:

```
10 CALL ZUB1 (ZZ%)
20 CALL ZUBØ (ZZ%)
30 GOTO 10
```

accetta un carattere da tastiera e ne fa l'eco su video all'infinito. In questo caso se premete ^C il programma invierà verso il video il carattere con valore ASCII corrispondente (cioè 3) senza eseguire l'usuale funzione di Break del Basic.

Per ovviare a questo inconveniente occorre modificare il programma come segue:

```
10 CALL ZUB1 (ZZ%)
20 IF ZZ%=3 THEN STOP
30 CALL ZUBØ (ZZ%)
40 GOTO 10
```

Passa da Computer City. Non sarai certo il primo.

La certezza di trovarci le marche che contano, le grandi protagoniste, da cui discendono i prodotti piú affidabili nel campo dei computers.
L'emozione di applicare i computers ad un'impresa affascinante come quella di rendere sempre piú efficiente la gestione della tua azienda. Questo ti dà Computer City: una vasta rete di centri specializzati nei piccoli computers,i piú adatti alle tue necessità, dove la vendita viaggia con l'assistenza di un personale esperto e qualificato, di cui ti puoi fidare. Il passo è fatto, da quando c'è Computer City.

computer city
Parla la tua lingua.