IMPARIAMO A PROGRAMMARE IN

assembler

di Valter Di Dio

In questa puntata ci occuperemo dello spostamento di grosse quantità di dati da una parte all'altra della memoria. È questo uno dei problemi che più comunemente si presenta ai programmatori, infatti spostare anche un solo Kappa di memoria dal basic impegna la macchina per un tempo a volte notevole. Sull'Apple occorrono circa 9 secondi, sul VIC o sul 64 pochi di più; ma se dobbiamo ricopiare una intera pagina grafica (8k) non possiamo arrestare il lavoro per oltre un minuto (1' e 12" sull'Apple)!

Il linguaggio macchina in questo caso diventa indispensabile; bastano infatti 40 centesimi di secondo per copiare 8192 locazioni di memoria senza l'uso di tecniche particolari. Programmi specializzati impiegherebbero infatti meno di un decimo di secondo, molto meno di quanto serva al Basic per riconoscere ed eseguire la CALL.

Spostamento di dati

Una delle cose che i computer fanno più spesso è quella di muovere dei dati, singoli o a blocchi, da una parte all'altra della memoria. Proprio per questo motivo i progettisti tendono a realizzare macchine in cui la possibilità di trasferimento dei dati sia la più comoda e la più veloce possibile.

Nel 6502 esistono due possibilità, diciamo base, di trasferimento a seconda che il blocco dei dati superi o meno i fatidici 256 byte. Al disotto di tale valore è infatti possibile utilizzare l'indirizzamento indicizzato dai registri X o Y che consente una gestione del trasferimento molto veloce ed efficiente in termini di spazio e impegno "fisico" del programmatore. Una semplice routine per trasferire un numero N di dati dalla locazione START alla DEST potrebbe essere il seguente:

```
        LDX # N
loop    LDA START, X
        STA DEST, X
        DEX
        BNE loop
        RTS
```

Nel caso invece di blocchi dati grandi quanto si voglia, si sente la mancanza di un registro a sedici bit che ci permetterebbe di utilizzare la stessa routine di prima. Resta comunque vero che molto raramente si devono trasferire grosse parti di memoria per cui l'incremento medio del tempo di esecuzione non risente molto di questa mancanza.

Una routine di MOVE, così si chiama in genere su tutte le macchine, che permette qualsiasi tipo di spostamento, è quella riportata nella figura a pié di pagina.

La routine è scritta con l'assembler LISA per Apple II; vediamola in particolare commentando anche le istruzioni del LISA (Lap 83 Interactive Symbolic Assembler).

La prima colonna di numeri a sinistra corrisponde alle locazioni di memoria del programma in linguaggio macchina definitivo, l'inizio viene comunicato al compilatore col comando OBJ $300; la seconda colonna comincia solo più in basso e contiene i codici esadecimali prodotti dal compilatore e insieme alla prima colonna formano il programma in linguaggio macchina vero e proprio. Purtroppo il LISA, a differenza di altri assembler, nella stampa non separa tra loro i codici oggetto, con un risultato estetico piuttosto penoso.

La terza colonna, quella che inizia per uno e prosegue in ordine crescente di uno, contiene il numero di riga delle istruzioni per il LISA (ricordo ancora che gli Assemblatori sono dei compilatori, usano quindi dei comandi in input che analizzano per generare il corrispondente modulo oggetto il programma in linguaggio macchina).

Tutte le righe che iniziano per ; sono dei commenti (REM), altri commenti si possono mettere in fondo ad una riga valida separandoli sempre con il punto e virgola.

Dalla riga 15 alla 20 troviamo una serie di cose strane. Sono le istruzioni di assegnazione. Servono al compilatore per sapere cosa mettere al posto dei nomi che noi diamo alle locazioni. Le assegnazioni più

```
0300          1          ORG $300
0300          2          OBJ $300
0300          3  ;
0300          4  ;************************
0300          5  ;
0300          6  ;     ROUTINE DI MOVE
0300          7  ;
0300          8  ; Copia da destinazione
0300          9  ; in poi tutto quello
0300         10  ; che si trova tra start
0300         11  ;  ed end.
0300         12  ;
0300         13  ;************************
0300         14  ;
0300         15  STARTL  EPZ $6
0300         16  STARTH  EPZ $7
0300         17  DESTL   EPZ $8
0300         18  DESTH   EPZ $9
0300         19  ENDL    EPZ $19
0300         20  ENDH    EPZ $1A
0300         21  ;
0300         22  ;
0300 A000    23  MOVE    LDY #0
0302 B106    24  LOOP    LDA (STARTL),Y
0304 9108    25          STA (DESTL),Y
0306         26  ;
0306         27  ; incrementa start
0306         28  ;
0306 E606    29          INC STARTL
0308 D002    30          BNE NEXTD
030A E607    31          INC STARTH
030C         32  ;
030C         33  ; incrementa destinazione
030C         34  ;
030C E608    35  NEXTD   INC DESTL
030E D002    36          BNE ANCORA
0310 E609    37          INC DESTH
0312         38  ;
0312         39  ; Ancora ?
0312         40  ;
0312 A506    41  ANCORA  LDA STARTL
0314 C519    42          CMP ENDL
0316 A507    43          LDA STARTH
0318 E51A    44          SBC ENDH
031A 90E6    45          BCC LOOP
031C         46  ;
031C         47  ; NO!
031C         48  ;
031C 60      49          RTS
             50          END

***** END OF ASSEMBLY
```

comuni sono la EQU che assegna una locazione qualsiasi al nome che si trova alla sua sinistra e la EPZ che è identica salvo che la locazione è in pagina zero (un solo byte quindi).

Alla riga 23 inizia il vero programma, la label MOVE non è citata nel resto del programma ma potrebbe esistere altrove se, ad esempio, il programma facesse parte di un sistema operativo (cosa del resto vera in quanto è stata estratta pari pari dal Monitor dell'Apple).

Il programma inizia con l'azzeramento del registro Y e questo dovrebbe già far presagire l'uso dell'indirizzamento diretto indicizzato, come sappiamo è infatti l'unico modo di scorrazzare liberamente per tutta la memoria. Alla riga 24 eccolo subito lì: carica nell'accumulatore il contenuto della locazione il cui indirizzo si trova in STARTL e STARTH ($6 e $7 come da EPZ). In queste due locazioni dovremo quindi scrivere l'indirizzo iniziale del blocco dati che vogliamo trasferire, prima di chiamare la MOVE.

Subito dopo, con lo stesso metodo, scarichiamo il contenuto dell'accumulatore nella locazione di destinazione, il cui indirizzo dovremo sempre aver scritto, prima, nelle locazioni $8 e $9.

Righe 29, 30 e 31 si incrementa di uno il puntatore START; 35, 36, 37 lo stesso per DEST.

Occorre ora controllare se si è raggiunto l'ultimo dato da copiare; dal momento che la routine deve poter muovere qualsiasi tipo di dato non si può usare un "tappo" cioè un valore speciale che non compare tra i dati, si controlla allora l'indirizzo dell'ultimo trasferimento effettuato con quello depositato in precedenza nelle locazioni $19 e $1A chiamate ENDL e ENDH.

Se guardate attentamente le righe da 41 a 45 noterete che il metodo usato per questo controllo è alquanto strano. Non preoccupatevi, non siete rincretiniti tutto d'un tratto, quello che state osservando è un tipico esempio di come strizzando a dovere il cervello si possa risparmiare memoria e tempo di esecuzione in un programma destinato ad un uso "pesante".

Risultato dell'operazione Confronto			
Operandi	N	Z	C
A,X,Y < M	1'	0	0
A,X,Y = M	0	1	1
A,X,Y < M	0'	0	1

Figura 1 - il flag di segno è posto in accordo con i numeri con segno, gli altri valgono per i numeri senza segno (0 - 255).

Vediamo prima la versione "normale" di questo pezzetto poi analizzeremo in dettaglio, e con l'uso di una tabella, il funzionamento di quella di Steve Wozniak papà dell'Apple.

```
ancora  LDA STARTL
        CMP ENDL
        BNE loop
        LDA STARTH
        CMP ENDH
        BNE loop
        RTS
```

Questo è semplice! Confronta la parte bassa di START con la parte bassa di END se non sono uguali salta a loop per continuare, se sono uguali controlla anche la parte alta se è uguale anche quella allora abbiamo finito altrimenti si continua. La lunghezza di questo programma è di dodici byte (RTS escluso).

Vediamo ora come funziona la routine di Wozniak.

```
*300L

0300-   A0 00     LDY   #$00
0302-   B1 06     LDA   ($06),Y
0304-   91 08     STA   ($08),Y
0306-   E6 06     INC   $06
0308-   D0 02     BNE   $030C
030A-   E6 07     INC   $07
030C-   E6 08     INC   $08
030E-   D0 02     BNE   $0312
0310-   E6 09     INC   $09
0312-   A5 06     LDA   $06
0314-   C5 19     CMP   $19
0316-   A5 07     LDA   $07
0318-   E5 1A     SBC   $1A
031A-   90 E6     BCC   $0302
031C-   60        RTS

:A
```

Disassemblato Apple della routine per trasferire un blocco di memoria. L'indirizzo di partenza va messo in $6, $7 la fine in 19, 1A la destinazione in $8, $9.

Dopo aver caricato la parte bassa esegue lo stesso il confronto ma, invece di effettuare subito il salto prosegue nei confronti. E il risultato precedente? Semplice, l'istruzione Compare effettua una sottrazione tra l'Accumulatore e la Memoria; se il risultato è zero viene settato il flag Z, ma se l'Accumulatore è minore della Memoria preleva dal Carry il prestito necessario alla sottrazione per cui le possibilità dopo un CMP sono quelle di figura 1.

Se allora STARTL e quindi Acc. è minore di ENDL il Carry è pulito perché è stato usato, se invece STARTL è > o = ad ENDL il Carry resta ad 1.

Veniamo ora alla seconda fase: questa volta invece di usare la CMP che modificherebbe nuovamente il Carry, si usa direttamente la SBC = Sottrai la Memoria dall'Accumulatore con Carry! I casi sono due, o il Carry è già vuoto, perché STARTL era minore di ENDL e allora in ogni caso resta vuoto anche in presenza di una eventuale richiesta di prestito da parte della sottrazione, oppure il Carry era settato perché STARTL era strettamente maggiore di ENDL e allora dobbiamo controllare se anche STARTH sia maggiore di ENDH. Ancora una volta ce lo dice il Carry; se infatti STARTH è minore di ENDH la sottrazione avrà utilizzato il Carry restato in precedenza, se invece STARTH è maggiore o uguale a ENDH (condizione di arresto) il Carry non viene usato e resta uguale a uno.

Il salto viene perciò eseguito solo in caso di Carry vuoto (BCC, Branch on Carry Clear) altrimenti ci si arresta (RTS).

Questo programma è più corto di due byte ed effettua un test con diramazione (da 5 a 6 µS) in meno del precedente. C'è comunque da dire che il programma precedente effettuava tutti e due i controlli solo una volta su 256 mentre questo deve eseguire tutte e due le CMP (anche se una travestita da SBC) ad ogni passaggio. Se qualcuno pensa che un risparmio di due byte sia poca cosa, cerchi di far entrare un programma per la gestione di una interfaccia RS-232 in soli 256 byte e in modo che possa lavorare in qualsiasi zona della memoria, naturalmente su ROM. MC

Quando i nomi contano.

HEWLETT PACKARD

calcolatrici professionali serie 10
computer tascabili serie 40
computer portatili serie 70
personal computer serie 80
personal computer tecnici serie 200

bit computers

rivenditore autorizzato HEWLETT PACKARD

Completa assistenza hardware e software, corsi e libri HP
Offerte promozionali di lancio, credito personale, leasing

bit computers

Computer shop: Roma, via F. Satolli, 55/57/59
(p.zza Pio XI) - tel. 06/6386096 - 6386146

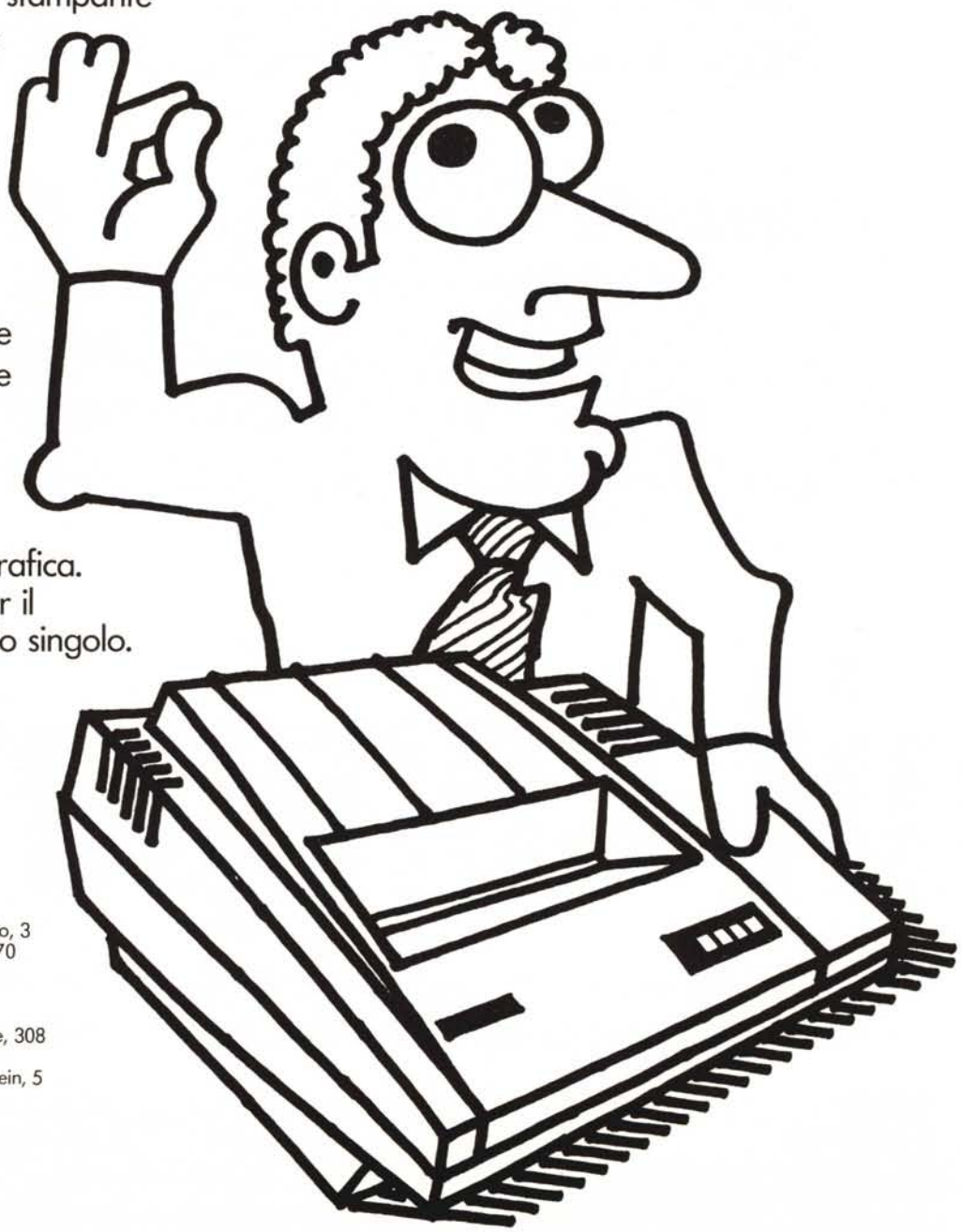
Sede centrale: Roma, v. Flavio Domiziano, 10
(EUR) - tel. 06/5126700 - 5138023

Agenzie:
LATINA: via Armando Diaz, 14 - telef. 0773/495285
LATINA: corso della Repubblica, 200 - telef. 0773/497301
CISTERNA DI LATINA: via Aversa, 11 - telef. 06/9696973
VITERBO: via Giacomo Matteotti, 73 - telef. 0761/38669
GAETA: lungomare Caboto, 74 - telef. 0771/470168
TARQUINIA: via S. Lucia Filippini, 17 - telef. 0766/856212

SE HAI UN PERSONAL...

"SPIRIT" E' LA TUA STAMPANTE CIOE' LA SUA.

"SPIRIT" è una nuova stampante seriale a 80 colonne.
È stata particolarmente studiata per raggiungere elevati livelli di qualità al costo più basso di mercato.
Facilmente collegabile a qualsiasi Personal e Micro Computers, include tra le prestazioni base la possibilità di stampa normale e grafica.
È già predisposta per il trattamento del foglio singolo.



MANNESMANN TALLY

20094 Corsico (MI) - Via Cadamosto, 3
Tel. (02) 4502850/855/860/865/870
Telex 4500934
00137 Roma - Via I. Del Lungo, 42
Tel. (06) 8278458
10099 San Mauro (TO) - Via Casale, 308
Tel. (011) 8225171
40050 Monteveglio (BO) - Via Einstein, 5
Tel. (051) 965208