ASSEMBLER 8086 8088

di Pierluigi Panunzi

quarta parte

Il set di istruzioni
Istruzioni di controllo

Siamo arrivati dunque alla quarta parte del nostro discorso relativo alle istruzioni di controllo del flusso di programmazione: abbiamo visto nelle scorse puntate come si effettuano i salti condizionati ed incondizionati. Nonché come si gestiscono le chiamate a subroutine, stavolta con un meccanismo solamente «incondizionato».
Terminiamo dunque l'analisi delle istruzioni di controllo, studiando ora tre istruzioni relative alla gestione dei tanto amati cicli di istruzioni, sia condizionati che incondizionati, nei quali la parte del leone, come vedremo subito, viene fatta dal registro CX.

Istruzioni di controllo - LOOP

Si tratta di un'istruzione molto usata programmando in Assembler proprio perché racchiude in sé le due operazoni necessarie al controllo di un «loop» di programma.

A beneficio dei programmatori alle prime armi, rammentiamo che si parla di «loop» o «ciclo di istruzioni» o «ciclo iterativo» allorché abbiamo a che fare con un certo numero di istruzioni (un «blocco») che deve essere ripetuto un certo numero di volte.

In termini grafici abbiamo una situazione del tipo riportato in figura A, dove un ruolo fondamentale è svolto dal cosiddetto «contatore di controllo del loop», nel nostro caso identificato dalla variabile «I».

Perciò, supponendo che il «blocco di istruzioni» deve essere ripetuto «M» volte, ecco che bisogna «inizializzare» (come si dice in gergo) la variabile I ad M e poi testare se ha raggiunto il valore nullo nel qual caso l'elaborazione prosegue per i fatti suoi.

Invece nel caso in cui tale limite non sia stato ancora raggiunto (è ad esempio il caso della prima iterata, cioè al primo passaggio) allora l'elaborazione viene dirottata verso quel piccolo bloccco in cui si decrementa il contatore I, dopodiché si ritorna ad elaborare il blocco principale di istruzioni.

Tutto questo può essere riportato nell'Assembler dell'8086/88, semplicemente considerando come contatore I il registro CX e dove al posto del blocco di test e di decremento c'è l'istruzione LOOP in esame.

In particolare l'istruzione LOOP ha innanzitutto la seguente sintassi:

```
LOOP etichetta
```

dove «etichetta» è il nome simbolico della locazione di memoria a cui si deve saltare allorché il loop debba essere eseguito un'altra volta.

Così come succede per le istruzioni di salto condizionato, anche per l'istruzione LOOP l'etichetta di arrivo deve essere posta entro +/− 128 byte dall'istruzione stessa e più precisamente tra −128 e +127 byte, dal momento che (oltre all'op-code pari a OE2H), l'istruzione prevede un altro unico byte per il ben noto «displacement». Passando ad analizzare in dettaglio il funzionamento, c'è da dire che per effetto di questa istruzione dapprima viene decrementato il contenuto del registro CX e poi ne viene testato il contenuto: se è diverso da 0 allora si avrà il salto all'etichetta indicata nell'istruzione, mentre altrimenti (siamo alla «chiusura» del loop) l'esecuzione passerà all'istruzione successiva.

Come di consueto riportiamo un semplice schemino che riporta le operazioni effettuate dal microprocessore quando esegue un'istruzione LOOP.

```
Istruzione LOOP
---------------
(CX) = (CX) - 1
if CX <> 0
 then goto "etichetta"
```

Istruzioni di controllo i LOOP condizionati

Oltre alla situazione vista con la figura precedente, ci si può imbattere in altri casi in cui l'uscita dal loop di istruzioni non deve avvenire solo per effetto dell'annullamento del contenuto di un certo contatore, ma anche per il sopraggiungere di una certa condizione, generata da una o più istruzioni all'interno del «blocco»: nel caso dell'Assembler 8086/88 la condizione che può far uscire dal loop è unicamente lo stato del flag di Zero (ZF).

In particolare, con l'istruzione LOOPE («LOOP if Equal») ed il suo «sinonimo» LOOPZ («LOOP if Zero»),

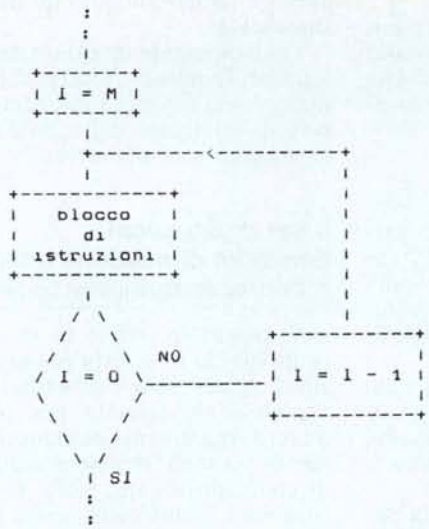

Figura A

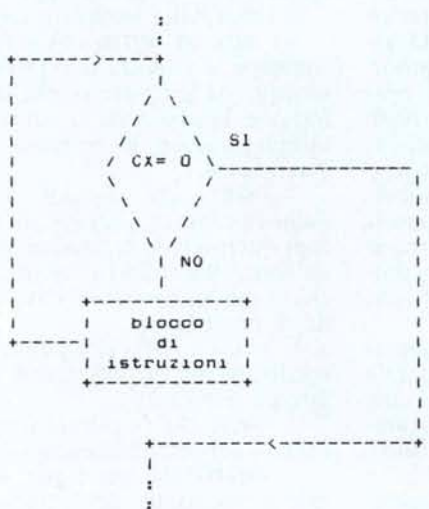

Figura C

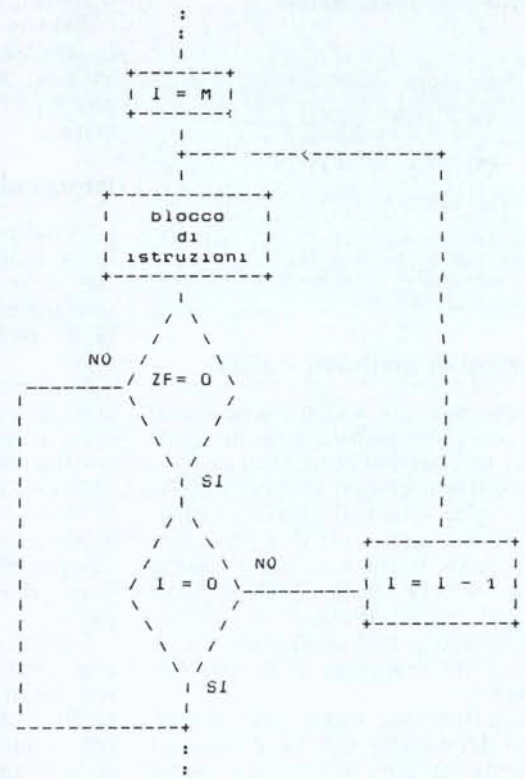

Figura B

nonché con le istruzioni duali LOOPNE («LOOP Not Equal») ed il suo sinonimo LOOPNZ («LOOP if Not Zero») si prevede la possibilità di testare lo stato del flag di Zero (ZF) ed effettuare di nuovo un'altra iterata, nel caso in cui la condizione sia verificata.

Le «LOOPE» e «LOOPZ» (esattamente identiche come codice operativo e perciò funzionamento, di codice operativo pari a OE1H) fanno eseguire una nuova iterata se il registro CX è ancora maggiore di 0 oppure se il flag di Zero è settato, mentre le istruzioni «LOOPNE» e «LOOPNZ» (analogamente identiche a tutti gli effetti ed aventi il codice operativo OEOH) saltano all'etichetta indicata se il contatore CX è diverso da 0 oppure se il flag di Zero è resettato.

Analogamente al caso dell'istruzione LOOP, possiamo rappresentare la situazione con la figura B, relativa al caso in cui l'uscita dal loop si ha, oltre che per l'azzeramento del contatore I, anche perché il flag di Zero (ZF) è settato.

Prima di proseguire nell'analisi, vogliamo fare un piccolo passo a ritroso ritornando per un istante all'istruzione LOOP: in questo caso sappiamo che l'uscita dal loop avviene allorché il registro CX si annulla, dopo essere stato decrementato ogni iterata.

Il fatto importante è che comunque il flag di Zero (ZF) non viene assolutamente toccato, sia nel caso in cui CX sia diverso da zero, sia nel caso in cui diventi proprio zero: questo per dire che lo stato del flag di Zero precedente all'istruzione LOOP viene mantenuto inalterato.

Tornando all'analisi delle istruzioni di «LOOP condizionato» abbiamo detto che l'uscita dal loop si ha per due condizioni: la prima è l'azzeramento del contatore (registro CX) e l'altra lo stato del flag di Zero (ZF), quest'ultima a seconda del tipo di istruzione.

Ecco che anche in questo caso valgono gli stessi ragionamenti fatti per la «LOOP» e cioè anche in questo caso le varie «LOOPE», «LOOPNE», «LOOPNZ» e «LOOPZ» non alterano in alcun modo lo stato «precedente» del flag di Zero, ma viceversa da esso dipendono. Anche in questo caso, la sintassi delle quattro istruzioni più volte citate è la seguente:

```
LOOPx etichetta
```

dove «x» sta per le lettere «E», «NE», «NZ» e «Z», rispettivamente «Equal», «Not Equal», «Not Zero» e «Zero» e dove «etichetta» è il nome simbolico della locazione di memoria a cui si salta nel caso che il loop debba essere eseguito un'altra volta.

Ancora una volta l'etichetta non può distare di più di 128 byte «all'indietro» (cioè per indirizzi decrescenti), né più di 127 «in avanti» e cioè verso indirizzi crescenti.

Per comodità del programmatore riportiamo la tabellina relativa alle

istruzioni in esame, che ne riporta in dettaglio il funzionamento.

```
Istruzioni LOOPE o LOOPZ
------------------------
(CX) = (CX) - 1
if (CX <> 0) and (ZF = 1)
 then goto "etichetta"

Istruzione LOOPNE o LOOPNZ
--------------------------
(CX) = (CX) - 1
if (CX <> 0) and (ZF <> 1)
 then goto "etichetta"
```

Istruzioni di controllo - JCXZ

Anche questa è un'istruzione molto utile per l'implementazione di loop: mentre le istruzioni precedenti gestivano i cosiddetti blocchi «REPEAT UNTIL» e cioè «ripeti il blocco di istruzioni fino a che si verifica una certa condizione», in questo caso possiamo gestire blocchi detti «WHILE DO» nei quali viene effettuato il test per primo e solo se tale condizione è vera allora viene eseguito il bloccco di istruzioni.

Nel nostro caso il test sarà sul contenuto del registro CX ed il salto ad una certa etichetta avverrà solamente se il contenuto di CX è nullo (vedi figura C).

In particolare la sintassi dell'istruzione è la solita:

```
JCXZ etichetta
```

dove ancora una volta «etichetta» può trovarsi solamente all'interno del range dato da 128 byte «all'indietro» e 127 byte «in avanti», sempre dovuto al fatto che dopo l'opcode di un byte (pari a OE3H) l'Assembler si aspetta un solo byte di «displacement».

In tabella riportiamo la schematizzazione (sempre molto utile quando si programma!) di quanto esegue l'istruzione in esame:

```
Istruzione JCXZ
---------------
if CX = 0
 then goto "etichetta"
```

Come si vede l'istruzione è stringatissima ma parimenti efficace in quanto racchiude in sé il test del registro CX ed il salto all'etichetta nel caso che il suo contenuto sia nullo: dimenticavamo di aggiungere prima del test il salvataggio dello stato attuale del flag di Zero ed il suo successivo ripristino, sia che si salti all'etichetta, sia che si prosegua all'istruzione successiva.

Già, perché anche in questo caso l'istruzione JCXZ non altera in alcun modo lo stato del flag di Zero (ZF), pur effettuando operazioni che normalmente lo altererebbero.

Evidentemente ciò risulta molto utile nel caso in cui istruzioni precedenti abbiano alterato il flag di Zero, che poi dovrà essere testato successivamente.

Istruzioni di controllo - NOP

Potrebbe sembrare strano, ma parliamo anche dell'istruzione NOP, anche se in realtà non effettua alcuna operazione, cosi come già si può intuire dal nome che sta per «No OPeration».

È in pratica un'istruzione innocua, che consente di generare, laddove servissero, dei semplici cicli di ritardo, considerato che la NOP in questione «dura» 3 cicli di clock.

Non ci sarebbe niente altro da aggiungere, oltre al fatto che il suo codice operativo e pari a 90H, ma segnaliamo viceversa alcune piccole annotazioni.

Innanzitutto il suo codice operativo non è stato scelto tra quelli non utilizzati da altre istruzioni ben più importanti: è chiaro che scegliere un proprio codice operativo per la NOP avrebbe impegnato inutilmente un codice, sfruttabile viceversa dai processori successivi (cosa che infatti succede, come vedremo: i microproccessori 80186, 80286 e 80386 infatti utilizzano per le loro istruzioni aggiuntive proprio i «buchi» lasciati dal capostipite, il nostro 8086/88).

In particolare, andando a studiare il dettaglio, il codice 90H corrisponde all'istruzione «XCHG AX,AX» che, come è facile vedere, effettua lo scambio tra il registro AX e se stesso, senza alterare i flag...

Semmai programmando in Assembler sarà l'assemblatore stesso ad inserire qua e là questa NOP laddove aveva predisposto due byte per un'istruzione che alla resa dei conti ne richiedeva uno solo. A tale scopo consigliamo di rivedere quanto detto a proposito dell'istruzione «JMP» che può essere sia a due byte (opcode + displacement ad un byte) oppure a tre byte (opcode + displacement a 16 bit), rispettivamente «JMP» di tipo «SHORT» e di tipo «LONG».

Altre volte in cui l'assemblatore inserisce una NOP è quando si fa riferimento in un'istruzione ad un'etichetta che l'Assembler ancora non ha incontrato: ad esempio scrivendo

```
          MOV SI.OFFSET TABELLA
          ...
          ...
          ...
TABELLA   DB ....
          ...
```

l'Assembler quasi sicuramente aggiungerà un «90H», dovuto ad un «ripensamento».

Evidentemente in questo caso è perfettamente inutile fornire al programmatore una tabellina indicante le operazioni effettuate dalla NOP, causa... la mancanza di operazioni!

Il set di istruzioni Istruzioni di gestione interrupt e coprocessori (prima parte)

Visto che lo spazio ce lo consente, proponiamo in questa puntata un inizio di analisi delle rimanenti istruzioni dell'8086/88, lasciate per ultime in quanto leggermente più complesse che non le normali istruzioni aritmetiche e di controllo sin qui viste: si tratta di istruzioni in un certo senso collegate all'hardware ed a particolari pin del microprocessore, pin ai quali è collegato qualcosa a cui il «nostro» deve rispondere in maniera opportuna.

Si tratta delle istruzioni seguenti:

— **«INT»** ed **«INTO»** che servono per simulare la risposta a determinati interrupt, via software o meglio per effettuare la chiamata a subroutine di utilità generale, in maniera veloce e «standard».

— **«IRET»** che consente il ritorno dalla routine di gestione di un interrupt esterno o dalla routine attivata da un'istruzione «INT» al programma che il microprocessore stava eseguendo in precedenza.

— **«CLI»** e **«STI»** che permettono rispettivamente di disabilitare ed abilitare gli interrupt.

— **«HLT»** che fa entrare il microprocessore nel particolare stato di «halt».

— **«WAIT»** che serve per sincronizzare l'esecuzione dell'istruzione successiva con l'accadere di un evento esterno.

— **«ESC»** che serve a segnalare all'8086/88 che l'istruzione successiva è un'istruzione riservata al coprocessore matematico 8087 e perciò deve attendere il termine dell'esecuzione prima di continuare nel programma.

— **«LOCK»** che serve nei casi in cui il microprocessore lavora in ambiente multi-processor, dove cioè ci sono più microprocessori a gestire le stesse risorse del sistema (memoria, I/O, ecc.).

Come si vede si tratta di istruzioni un po' più complesse, che richiedono ulteriori approfondimenti: a partire dalla prossima puntata le analizzeremo singolarmente.

Inizieremo con la gestione degli interrupt da parte del nostro microprocessore, dal momento in cui viene ricevuto al momento in cui si ridà il controllo al programma che era stato interrotto.

NEWEL s.r.l.

20155 MILANO - Via Mac Mahon, 75
tel. 02/32.34.92 - 32.70.226

BANCA DATI MODEM-SHOP - NUOVA NEWEL
E IN FUNZIONE TUTTI I GIORNI DALLE 13 ALLE 9
ALLO 02/3270226-300/1200 BAUD

DISTRIBUTORE ATARI PRODEST

NEGOZIO AL PUBBLICO E VENDITA PER CORRISPONDENZA
Attualità elettroniche e Microcomputers

DISTRIBUTORE Commodore

DIGITALIZZATORE VIDEO
Amiga 500 - 1000 - 2000
Sistema base

L. 125.000

MIDI INTERFACCE
L. 69.000

KRIVE L. 209.000
per il tuo Amiga 500 - 1000

È il più compatto drive esistente

AMIGA Sconti

ATARI

1040 STF	L. 990.000
1040 STF + MONITOR SM 125	L. 1.190.000
1040 STF + MONITOR SC 1224	L. 1.390.000
520 STM + DRIVE SF 314 (360Kb)	L. 690.000
520 STM + DRIVE MB 114 (720 Kb)	L. 790.000
520 STM + DRIVE MB 214 (2 × 720Kb)	L. 990.000
MB 114 (DRIVE 720 Kb)	L. 295.000
MB 214 (DOPPIO DRIVE 2 × 720Kb)	L. 450.000
MODULATORE ESTERNO × 1040	L. 95.000
PRO DRAW (TAVOLETTA GRAFICA)	L. 690.000
PUCK (MOUSE MAGNETICO × PRO DRAW)	L. 160.000
PRO SOUND DESIGNER (DIGIT. SUONO)	L. 120.000
MEGA ST 2 (2Mb RAM) + SM 125	L. CHIEDERE
MEGA ST 4 (4Mb RAM) + SM 125	L. CHIEDERE
ATARI LASERPRINTER	L. CHIEDERE
ATARI PC + MONITOR MONOCROMATICO	L. 950.000
ATARI PC + MONITOR A COLORI	L. 1.250.000

PLUS 2 è l'espansione di memoria da 512 Kb
Disponibile per Amiga 500 L. 127.500

NEWEL DISTRIBUTORE

IL PC1640 ECD AMSTRAD

Finalmente non dovrete più scegliere tra prezzo e prestazioni grafiche.
Con le avanzate capacità grafiche e le possibilità di espansione del PC1640 ECD non dovrete spendere un capitale per deliziare i vostri occhi.

Video a colori, tastiera,
Mouse e software 75-DOS e GEM inclusi

versione
a singolo FLOPPY DISK
da 360 KBYTE **L. 1.599.000 + IVA**

versione
con DISCO RIGIDOL. 2.599.000
da 20 MEGABYTE **+ IVA**

versione
doppio FLOPPY DISK
da 360 KBYTE **L. 1.849.000 + IVA**

Nessun altro PC Vi offre tanto per così poco

• Compatibile con EGA, Hercules, MDA e CGA • Altissima risoluzione, video a colori avanzato • 640K RAM • Microprocessor 8086 a 8 MHz • Versioni a singolo o doppio drive e a disco rigido • 3 slot per schede di espansione a dimensione intera • e naturalmente, completamente compatibile con "chi voi sapete"!

AMIGA 2000
a L. 2.000.000

SCONTI QUANTITÁ

AMIGA 500
a L. 780.000

Il sistema SYNTETYC permette di effettuare la digitalizzazione tramite una qualsiasi fonte sonora. Il pacchetto permette la manipolazione di quattro piste indipendenti sulle quali il possibile intervento è total. **125.000**

VIDEOSOUND

Digitalizzatore video-audio in un unico sistema hardware. Ha le stesse caratteristiche del VID e SYNTETYC.
Per Amiga 500 - 1000 - 2000 **L. 187.000**

THE NEW FINAL CARTRIGE III
per 64/128 (modo 64) **L. 70.000**
L'evoluzione continua!!!
Eccovi l'ultima release della mitica cartuccia notevolmente migliorata e modificata. Turbo, la favolosa rutine dello speeddos su cartuccia fino a 10 volte più veloce sia in lettura che in scrittura!!! 8 Tasti funzione programmati, 24 K ram extra per i prog. in Basic. Un favoloso Sprottettore di programmi tipo O.M.A. incorporato, dischi e cassette IN UN SOLO FILE!!! (+ boot se necessita) inoltre ha incorporato il GAME KILLER evita la collisione degli sprite, ed ha ben 40 comandi Basic Turbo a disposizione ... HARDCOPY "HR". Premendo un solo tasto potrete fare l'hardcopy del video in 12 gradazioni di grigio!!!
ECCEZIONALE!!!

NOVITÀ HARDWARE PER COMMODORE 64/128 - STARDOS NEW! L. 30.000
Eccezionale novità un velocizzatore che supera persino la velocità dello speeddos attiva i tasti funzione ecc. In una sola Eprom Kit da inserirsi nel C64 con manuale in ital. Non necessita di elaborazioni al drive né del cavo parallelo.
In dotazione anche un disco copiatori velocissimo!!!

O.M.A. PLUS (BANDID II) 64/128 & 1280 **L. 60.000**
Eccovi l'ultima rivoluzionaria cartuccia sprotettore di programmi, trasferisce **IN UN UNICO FILE** ricassettabili e il 99,99% del software protetto!!! Da nastro a disco da disco a disco, da disco a nastro, da nastro a nastro **IN TRE MINUTI E SEGUE TUTTO IL LAVORO!!!**

OC 118N SLIM

A SOLE LIRE 230.000

Il DISK DRIVE per il tuo Commodore 64/128
1) COMPATIBILE al 100% (stesso DOS Commodore);
2) Tipo SLIM LINE, con alimentatore esterno compreso);
3) DOPPIO connettore seriale;
4) GARANZIA totale (12 mesi, ricambi e mano d'opera);
5) Libretto ISTRUZIONI in italiano;
6) DEVIATORE esterno per cambiare numero del DRIVE.

DISTRIBUTORE PER MILANO E PROVINCIA

INTERFACCIA SERIALE × MODEM 64
L . 30000

OFFERTA STAMPANTI
SMITH CORONA 80 cps
BIDIREZIONALE, CARTA LETTERA
a L. 200.000 (GRAFICA IBM)

OFFERTE VALIDE NOVEMBRE 87 = PREZZI IVA ESCLUSA