ASSEMBLER 8086 8088

di Pierluigi Panunzi

terza parte

Il set di istruzioni
Istruzioni di controllo

Continuiamo la nostra analisi delle istruzioni di controllo del flusso di un programma parlando delle istruzioni di salto condizionato: si tratta di una trentina di istruzioni (almeno a livello codici mnemonici) che consentono di effettuare un salto all'etichetta indicata se si è verificata una condizione indicata dall'istruzione stessa.

Le istruzioni di controllo JUMP condizionate

Si tratta come detto di un certo numero di istruzioni senza dubbio molto utili in programmazione, se non indispensabili, e che si incontrano a decine se non a centinaia in un programma che si rispetti.

Ad occhio e croce in generale in un programma la parte del leone la fanno le istruzioni di MOV: una stima del 40% di presenza di tali istruzioni tra tutte quelle di un programma non dovrebbe essere lontana dalla realtà, anche se nessuno, a quanto ci consta, ha mai fatto studi in merito.

Comunque il secondo posto spetta senza dubbio ai salti condizionati, che certo non sono così belli a vedersi come le raffinate istruzioni "if-then-else" o "repeat-until" dei vari linguaggi ad alto livello, ma si sa, l'Assembler è un linguaggio a bassisimo livello, nettamente «non strutturato» e perciò ricco di istruzioni che fanno saltare qua e là, con un aggrovigliamento di idee e flussi di istruzioni: gli americani hanno battezzato il tutto con il termine colorito di "spaghetti-like programming".

Dicevamo che i salti in esame sono "condizionati" allo stato attuale di cinque flag principali di cui è dotato il microprocessore 8086/88 e cioè il flag di Zero (ZF), il Carry (CF), il flag di Parità (PF), il flag di Overflow (OF) ed il flag di Segno (SF).

In particolare, oltre alle istruzioni di salto sullo stato di "on" e quelle sullo stato di "off" dei cinque flag visti, per un totale di già dieci istruzioni differenti, il nostro microprocessore prevede due coppie di istruzioni legate allo stato di due flag insieme ed infine esiste un'altra coppia di istruzioni di salto condizionato allo stato contemporaneo di ben tre flag.

Vediamo nella tabella A un riassunto delle istruzioni di salto condizionato, laddove vediamo che praticamente quasi tutte le istruzioni hanno un "sinonimo", alcune hanno un nome solo ed un paio addirittura tre sinonimi: l'uso di uno o l'altro dei codici mnemonici è lasciato semplicemente al gusto del programmatore.

Prima di passare all'analisi delle singole istruzioni, innanzitutto diciamo, anche se non ce ne sarebbe bisogno, che tutte indistintamente si basano sullo stato "attuale" del o dei flag a cui si riferiscono, intendendo con il termine "attuale" il fatto che sono le istruzioni precedenti ad alterare lo stato del o dei flag (ad esempio le operazioni matematiche), stato che viene conservato laddove vengano eseguite istruzioni che "non alterano lo stato dei flag": a tal proposito tutte le istruzioni di salto condizionato appartenenti a questo gruppo non alterano minimamente lo stato dei flag, per cui possono essere poste in "cascata" anche più istruzioni di salto condizionato.

Ricordiamoci dunque che un'istruzione di MOV (tanto per fare un esempio) NON altera lo stato di alcun flag, neanche se si trattasse di un'istruzione del tipo

```
MOV AX, 0
```

come invece accadeva in microprocessori ad 8 bit della passata generazione.

Altra piccola considerazione riguarda la sintassi di queste istruzioni. Indicando con "Jxx" una qualsiasi istruzione di questo gruppo, la sintassi è:

```
Jxx ETICHETTA
```

dove ETICHETTA è per l'appunto l'etichetta a cui si deve saltare, la quale però deve sottostare alla condizione assai stringente di essere posta in memoria al massimo entro 127 byte in avanti o 128 byte all'indietro rispetto all'istruzione chiamante: si tratta perciò in ogni caso di "short jump", per i quali cioè nella codifica dell'istruzione appare non l'offset dell'etichetta, ma un "displacement", un byte che ci dice di quanto ci si deve spostare in avanti o indietro.

Nel caso che invece l'etichetta si trovi al di là delle barriere di +127 o −127 byte allora è usuale ricorrere ad un espediente semplicissimo.

Supponiamo di avere la situazione seguente in un nostro programma:

```
...
JZ LONTANA ;etichetta lontana!
MOV AX,ALFA
...
```

dove LONTANA è appunto un'etichetta posta al di là di 128 byte (e questo fatto viene segnalato dall'assem-

blatore, mica che ci dobbiamo mettere a contare i byte!!).

Il frammento di programma corretto allora diventa in questo modo:

```
        ...
        JNZ SOTTO
        JMP LONTANA
SOTTO:  MOV AX,ALFA
        ...
```

dove cioè si scrive il test opposto che va a scavalcare un'istruzione di JMP, che come sappiamo può arrivare dappertutto all'interno di un segmento: sì, lo sappiamo, è brutto a vedersi, ma egualmente efficace e richiede appena due byte in più...

Passiamo ora ad analizzare tutte le istruzioni della tabellina per vedere come funzionano.

codice	"jump if"
JE,JZ	equal, zero
JNE,JNE	not equal, not zero
JC,JB,JNAE	carry, below, not above nor equal
JNC,JNB,JAE	not carry, not below, above or equal
JP,JPE	parity, parity even
JNP,JPO	not parity, parity odd
JO	overflow
JNO	not overflow
JS	sign
JNS	not sign
JBE,JNA	below or equal, not above
JNBE,JA	not below nor equal, above
JL,JNGE	less, not greater nor equal
JNL,JGE	not less, greater or equal
JLE,JNG	less or equal, not greater
JNLE,JG	not less nor equal, greater

Tabella A

JE, JZ: si salterà all'etichetta se il flag di zero (ZF) è settato: in genere si troverà dopo un'istruzione di "CoMPare", ad esempio:

```
CMP AX,1000H
JE DOVEVUOI
```

oppure dopo un'istruzione in cui si testa se un registro è nullo

```
OR DI,DI
JZ ENULLO
```

Nel primo caso abbiamo usato il codice mnemonico "JE" dal momento, che volevamo vedere se AX è uguale a 1000H, mentre nel secondo caso abbiamo usato "JZ" perché testavamo se DI era nullo: nulla ci impedisce però di usare l'uno o l'altro in qualunque situazione, a seconda dei propri gusti o simpatie ed in alcuni casi (come quelli visti) per migliorare la "leggibilità".

JNE, JNZ: vale evidentemente quanto detto nel caso precedente, ma ovviamente in questo caso il salto avviene se il flag di zero (ZF) risulta resettato.

JC, JB, JNAE: in questo caso il flag testato è il Carry (CF), che deve essere settato se vogliamo effettuare il salto. I tre tipi di codici mnemonici derivano ancora una volta dalla migliore "leggibilità" che consentono; in particolare con JC vediamo subito che testiamo lo stato del Carry, ad esempio dopo un'operazione di shift, mentre con "JB" e "JNAE" (rispettivamente "Jump if Below" e "Jump if not Above nor Equal") si intende evidenziare il fatto che si testa se tra due quantità una è "below", cioè "sotto" oppure "not above nor equal" e cioè "né sopra né uguale" ad un'altra, intendendo in questo caso (attenzione!) che le due quantità devono essere considerate "unsigned" e cioè non dotate di segno (che ricordiamo essere in genere il bit più significativo).

JNC, JNB, JAE: analoga al caso precedente, ma relativa allo stato "off" ("0") del flag di Carry CF): in questo caso si salterà se una quantità "unsigned" è "non al di sotto" oppure «sopra o uguale» ad un'altra.

JP, JPE: il salto in questo caso viene effettuato se risulta settato il flag di Parity (PF).

Ricordiamo che il flag di parità, viene settato (e cioè la parità è "Even") se il numero di bit posti ad uno di un certo operando è pari: ad esempio nel frammento di programma

```
        ...
        MOV BX,3
        OR BX,BX
        JP SALTA
        ...
```

il registro BX contiene "3" e cioè la rappresentazione binaria contiene due "uni"; l'or con se stesso setterà dunque il flag PF, con il che si salterà a "SALTA".

JNP, JPO: in questo caso il salto avviene se la parità di un operando precedente era "odd" "dispari" e cioè era dispari il numero di "uni" nella rappresentazione binaria dell'operando: ciò comportava il reset del flag di parità.

JO: il salto avverrà se è settato il flag di Overflow (OF). Ricordiamo a questo punto (rimandando alle scorse puntate per i dettagli) che il flag di Overflow viene settato se, per effetto di una certa operazione, il nuovo bit più significativo risulta diverso da quello che era prima di effettuare l'operazione: ciò serve nelle operazioni con numeri dotati di segno per controllare che ad esempio la somma di due numeri negativi sia ancora un numero negativo, mentre nel caso in cui essa risulti positiva voleva dire che il byte o la word in cui essa era contenuta non era capace di mantenerne il valore corretto con il segno esatto, perché ... sono finiti i bit a disposizione (ecco perché si parla di "overflow"). In questo caso, ricordiamolo pure, non si intende minimamente di trabocco di moltiplicazione e divisioni, che ci pensano già loro (come visto nelle puntate precedenti) a gestirlo.

JNO: è il caso opposto del precedente, in cui cioè il flag di overflow non è settato.

JS: in questo caso il salto avviene se il flag di segno (SF) è settato: in generale ciò accade se il bit più significativo dell'operando su cui si era effettuata un'operazione precedentemente risulta settato, ad indicare che si tratta di un numero negativo.

Ricordiamo che il fatto che un numero sia negativo perché il suo MSB è settato vale solo se "noi" lo intendiamo negativo: il valore OFFH vale "– 1" se lo consideriamo in notazione "complemento a 2" e cioè come numero negativo, mentre vale "255" se letto come "unsigned" e cioè senza considerarne il segno, utilizzando cioè l'MSB come bit significativo.

Il microprocessore lavorerà in entrambi i casi nello stesso modo, ma lascerà a noi il compito di interpretare il risultato.

JNS: è evidentemente la condizione opposta alla precedente, riferita allo stato di "off" ("O") del flag SF.

JBE, JNA: ecco il primo caso di salto condizionato allo stato di due flag, in questo caso il carry (CF) e lo Zero (ZF): il salto avviene laddove almeno uno dei due flag (oppure entrambi i flag) è settato, mentre viceversa non si ha il salto se entrambi i flag sono resettati.

Si tratta evidentemente dell'"OR" tra i due flag e spiega il perché dei due codici mnemonici: "Below or Equal" richiede o la condizione "Below" (espressa dal CF settato) oppure la condizione "Equal" (espressa da ZF settato) oppure entrambe.

Il sinonimo è evidente: "sotto o uguale" equivale a "non sopra", il tutto sempre relativo a quantità non dotate di segno.

JNBE, JA: è la condizione opposta alla precedente per la quale si salta all'etichetta se entrambi i flag ZF e CF so-

codice	condizione	*
JE, JZ	ZF = 1	-
JNE, JNE	ZF = 0	-
JC, JB, JNAE	CF = 1	U
JNC, JNB, JAE	CF = 0	U
JP, JPE	PF = 1	-
JNP, JPO	PF = 0	-
JO	OF = 1	-
JNO	OF = 0	-
JS	SF = 1	S
JNS	SF = 0	S
JBE, JNA	(CF or ZF) = 1	U
JNBE, JA	(CF or ZF) = 0	U
JL, JNGE	(SF xor OF) = 1	S
JNL, JGE	(SF xor OF) = 0	S
JLE, JNG	((SF xor OF) or ZF) = 1	S
JNLE, JG	((SF xor OF) oR ZF) = 0	S

(*) U = quantita' "Unsigned"
S = quantita' "Signed"
- = indifferente

Tabella B

no nulli e cioè se la quantità "unsigned" è "non sotto né uguale" oppure "sopra" rispetto ad un'altra quantità.

JL, JNGE: ecco il secondo caso di istruzione condizionata allo stato di due flag, in questo caso il flag di segno (SF) e l'overflow (OF): si parla dunque di quantità dotate di segno (e cioè positive o negative, perché noi vogliamo leggerle così) ed in questo caso il salto avverrà se e solo se i due flag in esame sono differenti e cioè quando il loro XOR vale "1".

In questo caso si parla di quantità "Less" o "Not Greater nor Equal" (rispettivamente "minore" o "né maggiore né uguale") tra quantità dotate di segno: per chiarire il tutto facciamo un esempio

```
...
MOV AX,ALFA
CMP AX, BETA
JL MINORE
...
```

e vediamo a seconda dei valori di ALFA e BETA cosa succede.

Se entrambi sono positivi la CMP manterrà il flag di segno resettato mentre (dato che la CMP effettua la sottrazione "ALFA − BETA" senza riportare il risultato ma settando i flag se necessario) setterà il flag di overflow solo se ALFA è minore di BETA (dato che il risultato della sottrazione è negativo mentre in partenza i numeri erano positivi): solo in questo caso SF = 0 e OF = 1 con il che il salto viene effettuato. Il caso in cui ALFA è maggiore di BETA comporta infatti SF = 0 e OF = 0 in quanto la sottrazione genera un numero positivo, come positivo era il primo operando.

Se ALFA è positivo e BETA è negativo già sappiamo che il salto non deve avvenire: vediamone la ragione in termini dei flag. In questo caso SF è nullo in quanto ALFA è positivo e la sottrazione genera un risultato ancora positivo, il che comporta il reset dell'overflow (risultato positivo da un operando positivo): i due flag sono dunque uguali e il salto non si ha.

Vediamo ora i casi in cui ALFA è negativo.

Se anche BETA è negativo allora il flag di segno è settato comunque ed ora bisogna distinguere i due casi in cui il valore assoluto di ALFA sia maggiore o minore di quello di BETA: supponiamo che ALFA vale −5 e BETA valga −10.

Già sappiamo che in questo caso ALFA non è minore di BETA e perciò il salto non deve avvenire: verifichiamolo.

Il flag di segno è come detto posto ad "1" in quanto ALFA è negativo e la sottrazione genera un valore positivo, scatenando l'overflow: i flag sono uguali e perciò non si salta!

Se invece ALFA vale −5 e BETA vale −4 allora si dovrà saltare: infatti il flag di segno è settato, ma in questo caso non si genera overflow in quanto la sottrazione dà un risultato negativo.

L'ultimo caso da analizzare è che ALFA sia negativo con BETA positivo: in questo caso sappiamo che si dovrà effettuare il salto, cosa che verifichiamo subito.

ALFA è negativo e perciò SF = 1; la sottrazione "ALFA − BETA" genererà un numero negativo per cui l'overflow se ne sta lì buono buono, consentendo dunque il salto in quanto differente da SF.

JNL, JGE: a questo punto possiamo dire che si tratta semplicemente del caso opposto al precedente e che prevederà il salto in tutti quei casi in cui i flag SF e OF risultano uguali, ovverosia quando il loro XOR risulta nullo.

JLE, JNG: è questa con la successiva coppia di istruzioni, una coppia di salti condizionati allo stato di tre flag: SF e OF come nel caso precedente ed in più ZF.

Si tratta infatti del caso "Less or Equal" oppure "Not Greater" e che perciò racchiude le condizioni "Less" viste nel caso precedente con la condizione "Equal" che ricordiamo essere data semplicemente da ZF = 1.

In parole povere il salto avverrà in questo caso se è verificata una delle due condizioni "minore" o "uguale" oppure entrambe: ciò in termini booleani significa l'OR tra la condizione di "minore" e quella di "uguale", rispettivamente date da (SF xor OF) = 1 e ZF = 1.

In definitiva si ha il salto se

```
( (SF xor OF) or ZF ) = 1
```

il che avviene quando o (SF xor OF) = 1 o ZF = 1 o entrambe, come volevasi dimostrare.

JNLE, JG: invece è questo il caso in cui il salto avviene se entrambe le due condizioni precedenti (SF xor OF) e ZF sono entrambe false e cioè se SF risulta uguale ad OF (entrambi nulli o entrambi ad "1", fatto che annulla lo XOR) ed inoltre se ZF = 0.

In parole povere ciò si ha se il primo operando non è minore del secondo operando (si vedano le considerazioni precedenti...), ma non è neanche uguale al secondo operando: in parole ancora più povere il salto si ha se il primo operando era "maggiore" ("Greater") del secondo, fatto che ci fa tornare in paro i conti!

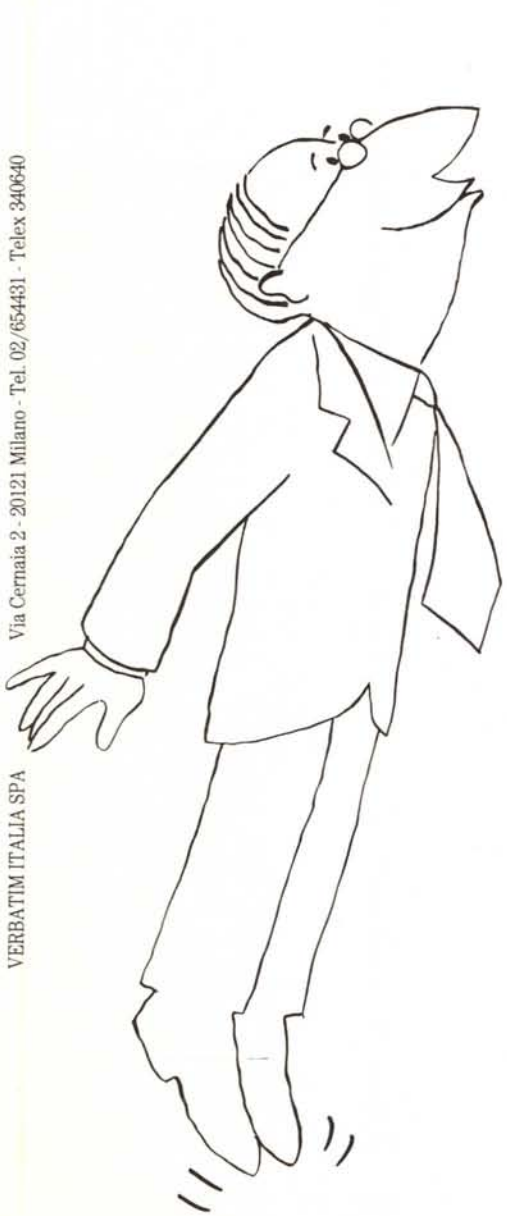
Terminata dunque questa carrellata pensiamo di fare cosa gradita ai lettori riassumendo nella tabella B le condizioni per le quali l'istruzione indicata (o i suoi sinonimi) comportano un salto.

Abbiamo inoltre aggiunto una colonna in cui le lettere "U" e "S" rappresentano il fatto che il test è valido per quantità rispettivamente "Unsigned" e "Signed".

Termineremo nella prossima puntata l'analisi delle istruzioni di controllo dei cicli: a risentirci dunque.

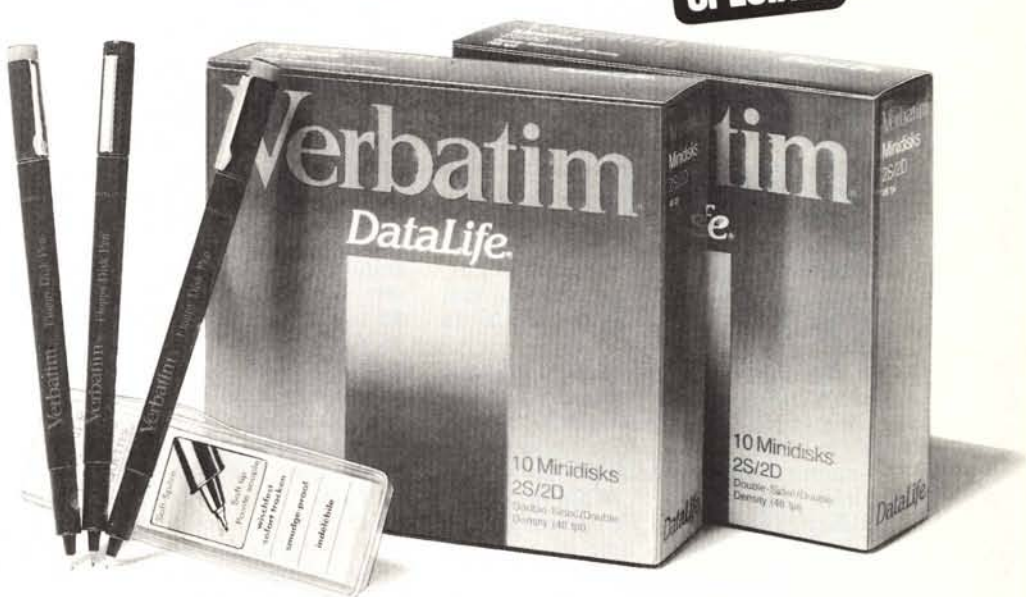
MC

AVETE MAI COMPERATO DUE CONFEZIONI VERBATIM 5¼" PER SCRIVERE A MANO SUL VOSTRO FLOPPY?



Due confezioni DataLife® 5¼" 2S/2D in una...
Comperando due confezioni di DataLife® 5¼" 2S/2D da 20 dischetti, 10 per confezione, troverai tre floppy disk pen.
Tre penne speciali che Ti permetteranno di scrivere, per la particolare flessibilità della punta, sul jacket senza danneggiare il floppy disk.
Un'idea Verbatim®.

OFFERTA SPECIALE



Verbatim
A Kodak Company

VERBATIM ITALIA SPA
Via Cernaia 2 - 20121 Milano - Tel. 02/654431 - Telex 340640