L'ASSEMBLER dello Z80

di Pierluigi Panunzi

Le istruzioni di salto

Dopo aver visto nelle scorse puntate un gran numero di istruzioni "statiche", nel senso che non portavano altro "sconvolgimento" che la lettura di un certo dato dalla memoria, ci occuperemo questa volta di un gruppo di istruzioni molto importanti, in quanto consentono di alterare la naturale sequenza di un programma: stiamo parlando delle cosiddette "istruzioni di salto".

In presenza di tali istruzioni lo Z80, invece di eseguire la istruzione successiva a quella eseguita per ultima, salta ad un altro punto della memoria dove troverà ad aspettarlo (se tutto va bene!) un altro pezzo di programma.

Analizzeremo dunque vari tipi di istruzioni di salto: dapprima ci occuperemo dei salti assoluti (condizionati e non), poi dei salti cosiddetti relativi (anche questi condizionati e non) ed infine di alcune particolari istruzioni di salto indiretto e di gestione dei cicli di istruzioni.

Come dice il loro nome e come è facilmente intuibile, le istruzioni di salto consentono al programma di saltare da un punto all'altro della memoria oppure, ma è la stessa cosa, permettono al programmatore di far eseguire determinate parti di un certo programma invece di altre, per il fatto che si è verificata una certa condizione, o per il semplice motivo che è necessario effettuare il salto.

Le istruzioni di salto per lo Z80 si dividono in tre gruppi: al primo appartengono i salti, condizionati e non, assoluti, al secondo i salti relativi, anche questi condizionati e non, mentre al terzo appartengono quelli indiretti. Cominciamo allora dal primo gruppo per dire che il loro attributo di "assoluti" deriva dal fatto che l'istruzione stessa specifica l'indirizzo di memoria al quale deve saltare il programma: in particolare si tratta di istruzioni formate da tre byte, in cui il primo è il codice operativo ("Opcode") e gli altri due rappresentano l'indirizzo di salto.

Soffermiamoci su questo indirizzo con un esempio: iniziamo con l'istruzione "JP 1234H", che impone al programma di saltare all'indirizzo (esadecimale) 1234H.

Supponendo che l'istruzione si trovi all'indirizzo (sempre esadecimale, ma ora non lo diremo più) 0500H, automaticamente lo Z80 si posizionerà sull'indirizzo 1234H, praticamente ignorando 0C31H celle di memoria.

Non perdiamo l'abitudine a fare piccoli calcoli in esadecimale! Il valore citato si ottiene semplicemente effettuando la sottrazione tra 1234H e 0500H, che dà 0C34H: ora però da questo valore dobbiamo sottrarre i due byte rimanenti dell'istruzione di salto che stiamo analizzando ed infine dobbiamo ancora sottrarre 1 in quanto stiamo calcolando il "numero di byte" tra l'istruzione di partenza e quella di arrivo.

Chiudiamo questa parentesi, in quanto generalmente non dobbiamo fare questi calcoli: ci serviva soltanto per far vedere l'"entità" di un certo salto (nel nostro caso "assoluto"), entità che può assumere praticamente qualsiasi valore tra 0000H ed FFFFH, sottintendendosi così che con un'istruzione di salto assoluto si può raggiungere qualsiasi punto della memoria, senza alcun tipo di "impedimento" o "protezione" che dir si voglia.

Detto ciò possiamo analizzare le modalità di esecuzione dell'istruzione: in particolare l'istruzione vista fa saltare sempre all'indirizzo dato.

Esistono poi le istruzioni di "salto assoluto condizionato", per le quali giocano un ruolo molto importante i "flag", dei quali abbiamo parlato abbondantemente nella puntata relativa alle operazioni logico-aritmetiche dello Z80.

Non a caso ci eravamo soffermati sui tre flag C, Z e S (rispettivamente il Carry, il flag di Zero e quello di Segno): questa volta non ci occuperemo del quarto flag (N), ma viceversa parleremo dell'altro flag (P, flag di Parità).

In particolare, nel caso di un'istruzione di salto condizionato bisogna far bene attenzione al valore di questi quattro flag ed in particolar modo a quello strettamente legato all'istruzione di salto.

Dato che i flag sono quattro (ripetiamo: Z, C, P, S) e dato che ognuno di questo può avere due "stati" (ne riparliamo più oltre...) ecco che otteniamo 8 istruzioni di salto, che sono:

```
JP  Z,nnnn
JP NZ,nnnn
JP  C,nnnn
JP NC,nnnn
JP PO,nnnn
JP PO,nnnn
JP PE,nnnn
JP  P,nnnn
JP  M,nnnn
```

dove "nnnn" è un indirizzo assoluto esadecimale, formato da 4 cifre esadecimali (come ad esempio 1000H, D4F3H, 777AH, ecc.)

Analizziamo la prima istruzione, che si legge "salta, se zero, all'indirizzo nnnn": come dice la frase, il salto avviene se e solo se dalle istruzioni precedenti, ed in particolare "la" precedente, si ha che il flag "Zero" è settato.

Se ciò non accade, e questo vale in *tutti* i casi di salto condizionato, e cioè se la condizione non è verificata, allora l'istruzione di salto viene semplicemente ignorata ed il programma passa all'istruzione successiva.

Nella seconda istruzione dell'elenco il salto avviene se e solo se il flag di Zero è 0 (ad esempio se il risultato di una certa operazione "non" era zero, secondo l'ormai consueto gioco di parole).

Analogamente si ha per la terza e la quarta istruzione, che si riferiscono stavolta al flag di Carry.

La quinta e la sesta istruzione si riferiscono invece al flag di Parità: vediamo ora più in dettaglio il significato di questo flag.

Dato un certo byte, che come sappiamo bene è formato da 8 bit, si intende per parità la somma dei bit "1" costituenti il byte stesso: il flag o bit di parità è concepito in modo tale da rendere sempre "dispari" il numero totale di "1" (8 del byte, più uno dato dal flag stesso).

Facciamo un esempio, anzi un paio!

Supponiamo di avere il byte 34H: esprimendo tale valore in notazione binaria avremo 00111000.

Ora andiamo a contare il numero di uni del nostro byte: è facile! Sono tre. Ora il flag di parità sarà in questo caso 0 in quanto solo così il totale degli uni rimane dispari. Viceversa, consideriamo il byte F0H: scritto in binario abbbiamo 11110000. In totale abbiamo dunque quattro uni: avendo a disposizione l'ulteriore bit dato dal flag di Parità, questo deve valere "1" rendendo così dispari (pari a 5) la somma dei bit pari ad "1".

Nel primo degli esempi (in cui P vale 0) si dice che la parità è "dispari" ("Odd" in inglese), invece nel secondo esempio la parità si dice "pari" ("Even" in inglese) ed il flag P è pari ad 1.

Tornando perciò alle istruzioni di salto condizionato, le "JP PO,nnnn" e "JP PE,nnnn" si leggono "salta se la parità è dispari..." e "salta se la parità è pari...".

Evidentemente sarebbe meglio leggere le istruzioni in inglese "jump on parity odd to nnnn" e "jump on parity even to nnnn" in quanto solo così abbiamo la corrispondenza con le sigle PO e PE. A loro volta, comunque si leggano, le istruzioni significano: la prima, "salta se P=0 " e la seconda "salta se P=1".

A questo punto però si impone una fermata: specie per chi è alle prime armi, non è molto facile distinguere subito bene concetti come parità pari, dispari e collegarli a PO, PE e ai valori di P: proponiamo dunque un semplicissimo metodo mnemonico (inventato dall'autore dell'articolo, ma facilmente scovabile!), che per il fatto di essere mnemonico è, sì, molto semplice, ma altresì non trova alcun riscontro "matematico". Eccolo dunque: per raccapezzarcisi sulle JP PO e JP PE basta semplicemente fissare l'attenzione sulla prima (per l'altra si va evidentemente ad esclusione!), che è "JP PO".

Le lettere "PO" come visto significano "Parity Odd": per un caso fortuito la chiave è nella paroletta "Odd" dove c'è praticamente tutto! C'è la "O" che ci ricorda che il bit di parità è "0 " ed in più ci sono ben due "d" che ci indicano che la parità è "d"-ispari!! Semplice ma a suo modo efficace!

Ripetiamo che dobbiamo solo pensare alla sigla PO e alla parola "Odd": inevitabilmente il discorso opposto non si può ottenere da PE e tantomeno da "Even" (forse, machiavellicamente, si potrebbero contare le cifre che compongono le "parolette": è un caso anche questo, ma "Odd" è formato da tre lettere, mentre "Even" da quattro!!). E poi dicono che la matematica non è un'opinione...

Comunque lasciamo al lettore la scelta di quale metodo usare per associare meccanicamente o coscientemente i vari concetti relativi alla parità: altra scelta è anche quella di ignorare completamente il nostro "trucchetto ignobile"...

Tornando dunque alle ultime due istruzioni di salto condizionato, abbiamo le "JP P,nnnn" e "JP M,nnnn": rispettivamente si riferiscono al segno risultante dalle operazioni precedenti. Se il bit di flag S è 1, il segno negativo (il segno, in inglese è "Minus") e viceversa, se S è 0, il segno è positivo (in inglese "Plus"). Ecco che la prima istruzione effettuerà il salto se il segno risultante era "Plus", mentre la seconda lo effettuerà se il segno risultante è "Minus".

I salti relativi

È questo un tipo di istruzione che si trova anche in altri microprocessori ad 8 bit, quale ad esempio il 6502, ma che viceversa non era presente nell'8080, il processore dello Z80. Come dice il nome, si trattava di salti "relativi all'indirizzo" di memoria in cui si trovano: con queste istruzioni si può saltare ad una certa istruzione che si trova, rispetto a quella di salto, nelle "immediate vicinanze" intendendo con questo termine una distanza di al massimo 127 byte in più oppure 128 byte in meno, rispetto alla locazione in cui si trova nell'istruzione di salto.

Come si vede, dunque, c'è una differenza sostanziale con le istruzioni di salto precedenti: in questo caso infatti l'indirizzo di arrivo *non* può assolutamente spaziare tra 0000H ed FFFFH, come avveniva invece nel caso precedente.

La spiegazione di ciò ed in particolare del valore massimo del "range" di indirizzi raggiungibili, risiede nel fatto che questo tipo di istruzioni è a due byte in cui il primo è al solito l'"opcode", mentre il secondo byte rappresenta un "valore esadecimale complemento a 2". Consigliamo a questo punto i lettori di fare riferimento alla puntata apparsa sul n. 36 di MCmicrocomputer: questa volta diciamo solo che secondo tale "logica", un byte può rappresentare un valore per l'appunto compreso tra -128 e +127.

In questo caso però, rispetto alle istruzioni di salto assoluto, abbiamo meno possibilità, fondamentalmente per una scelta dei costruttori; le istruzioni di salto relativo sono:

```
JR   e
JR   C,e
JR NC,e
JR   Z,e
JR NZ,e
```

dove "e" (in inglese "extension", estensione) rappresenta appunto il secondo byte.

La prima istruzione della tabellina rappresenta un salto relativo incondizionato, mentre le altre sono rispettivamente condizionate allo stato del flag di Carry (se è settato o no) e del flag di Zero (se è settato o no).

Non ci dilunghiamo oltre in quanto, a parte le considerazioni sull'indirizzo, non vi è altro di nuovo rispetto alle precedenti istruzioni di salto.

Le istruzioni di Salto Indiretto

Appartengono a questo piccolo gruppo tre istruzioni di salto indiretto:

```
JP (HL)
JP (IX)
JP (IY)
```

nelle quali l'indirizzo non è indicato dall'istruzione stessa, ma viceversa è contenuto nella coppia di registri HL oppure nei registri indice IX e IY: dato che in ogni caso si tratta di registri a 16 bit (perciò 2 byte), abbiamo in questo caso le stesse possibilità di indirizzamento delle istruzioni assolute.

Vantaggio notevole di queste istruzioni è che l'indirizzo può essere "calcolato" dal programma stesso: in genere vengono sfruttate quando da una parte del programma possiamo saltare ad un certo numero di parti della memoria, parti i cui indirizzi iniziali sono ad esempio posti in un'apposita tabella.

Un'istruzione per i loop

Chiudiamo questa puntata riguardante i salti con un'istruzione molto potente, la

```
DJNZ e
```

la quale consente un'agevolissima gestione dei cicli o loop che dir si voglia.

In termini tecnici, si tratta di un'istruzione "multipla", nel senso che non si limita semplicemente ad effettuare un salto, seppur condizionato o no.

In questo caso l'istruzione decrementa il contenuto del registro B ed effettua un salto relativo (nell'istruzione compare infatti una "e") se il contenuto di B risultante non è zero; ecco dunque l'utilizzaziione della DJNZ nei loop: basta porre nel registro B il numero di volte che il ciclo deve essere effettuato.

Facciamo dunque riferimento ad una situazione del genere:

```
...
programma
...
LD B,numero di volte
...
(etichetta)
blocco di istruzioni
DJNZ etichetta
```

In questo caso abbiamo un programma in cui ad un certo punto poniamo in B un valore (compreso tra 0 e 255): ora si avrà che il "blocco di istruzioni" verrà eseguito tante volte quanto è il valore posto inizialmente nel registro B.

Infatti l'istruzione DJNZ decrementerà il registro B e fino al suo successivo annullamento effettuerà il salto all'"etichetta".

Dimenticavamo di dire che il nome dell'istruzione DJNZ deriva da "Decrement and Jump on Not Zero" e cioè "decrementa (il registro B) e salta sulla condizione di non-zero".

Chiudiamo dunque questa rassegna sulle istruzioni di salto dicendo che in nessuno dei casi visti si ha un'alterazione di alcuni flag: come dire che lo stato dei flag viene integralmente mantenuto, anche in quest'ultimo caso in cui il flag di Zero viene effettivamente utilizzato, ripetiamo, perché è importante, che lo stato dei flag rimane quello che si aveva "prima" dell'esecuzione dell'istruzione di salto, qualsiasi essa sia.

Nella prossima puntata ci occuperemo della gestione delle subroutine e vedremo che ritorneranno "a galla" molti dei concetti espressi questa volta. MC

dcp mondadori

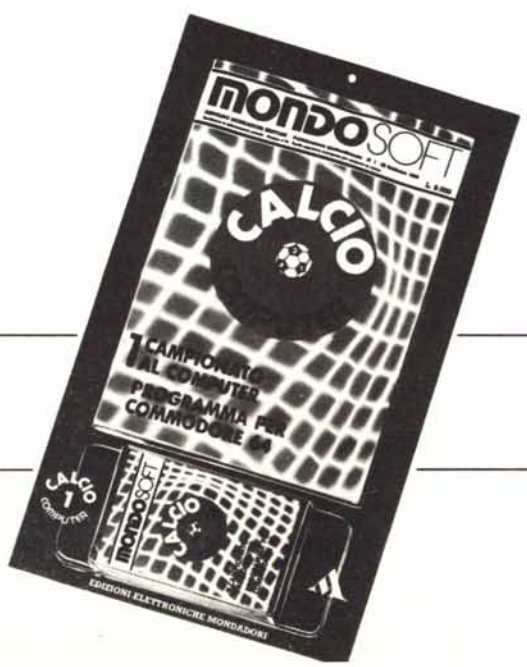
SHIFT RUN PLAY GOAL!

Calcio Computer, *8 programmi software per Commodore 64, che di volta in volta ti insegneranno a compilare la schedina (e possibilmente vincere), elaborare i tuoi pronostici sulla classifica di campionato, raccogliere e consultare i dati delle squadre e dei giocatori di serie A e B.*

Con ***Calcio Computer*** *si inaugura* ***Mondosoft,*** *il nuovo periodico per home computer della Mondadori.*

Ogni quattordici giorni in edicola con un nuovo programma su cassetta corredato da un fascicolo a colori.

Mondosoft, *per esplorare i vari "mondi" del software.*



MONDOSOFT

Il periodico che leggi con il computer.