L'ASSEMBLER dello Z80

di Pierluigi Panunzi

Terza parte

Come segnalato al termine della scorsa puntata, in questa termineremo il discorso sulle istruzioni di caricamento (le ormai ben note "LD"), introducendo tre nuovi utilissimi registri, dei quali abbiamo parlato nella prima puntata nel corso della descrizione dello Z80.

I primi due registri di cui ci occuperemo ora sono IX ed IY: si tratta di due registri a 16 bit praticamente "paralleli" nel senso che si può parlare indifferentemente dell'uno ed implicitamente fare riferimento all'altro.

Come coppia di registri, IX e IY vengono detti "registri indice" e consentono un nuovo tipo di indirizzamento della memoria, che prende appunto il nome di "indirizzamento indicizzato".

Genericamente si potrebbe dire che le funzioni svolte dai due registri in questione sono un'estensione di quelle della famosa coppia di HL: l'altra volta abbiamo visto il significato dell'indirizzamento indiretto e l'uso di HL come "supporto" per l'indirizzo di memoria della cella il cui dato ci interessa.

Ora la stessa cosa si può fare con IX ed IY, con una sostanziale differenza: salvo rarissime eccezioni, ma che non rientrano nel discorso delle istruzioni "LD", i registri IX ed IY servono ancora da supporto all'indirizzo di una certa cella, ma a questo indirizzo contenuto in IX e IY viene aggiunto uno "spostamento" (displacement) in più o in meno per ottenere l'indirizzo effettivo della cella desiderata.

Già ne avevamo fatto un accenno proprio alla fine della scorsa puntata, ma ora vediamo bene cosa significa un'istruzione del tipo:

```
LD A, (IX+5)
```

Analogamente a tutti i casi visti precedentemente noi vogliamo caricare (LD) l'accumulatore (A) con un certo valore: il fatto che c'è qualcosa di scritto tra parentesi vuol dire innanzitutto che non si tratta del caricamento di un valore immediato. Si tratta allora di caricamento che interessa una cella di memoria: dato che però non troviamo un indirizzo effettivo, vuol dire che l'indirizzamento non è diretto, bensì indiretto.

Nel nostro caso l'indirizzo della cella desiderata è posto in IX, ma ancora non basta!

Il "+5" indica infatti che a tale indirizzo di memoria posto in IX dobbiamo aggiungere 5, ottenendo per forza di cose un nuovo valore che indirizza una nuova cella.

Ma vediamo di seguire ancora meglio il ragionamento con uno schema, quello posto in figura 1.

Supponiamo perciò di avere in memoria una certa tabella formata da un certo numero di dati: caricando in IX (o IY) l'indirizzo della prima cella costituente la tabella, potremo far riferimento alle successive celle per mezzo del meccanismo del "displacement". Sapendo che la cella iniziale è ottenibile con uno spostamento nullo e cioè indirizzabile da (IX + 0), la quinta cella sarà indirizzata spostandoci di quattro posizioni e cioè con (IX + 4) ed in generale l'n-esima cella avrà un displacement di n-1: il valore massimo per n è fissato a 128, con displacement massimo pari a 127. Lo stesso discorso si può fare se ci si vuole spostare "all'indietro" nella memoria: la quinta cella precedente sarà indirizzata da (IX-5) e così via fino ad un valore massimo (negativo!) di 128.

La logica complementata

La ragione di questi valori (+127 e -128) risiede nel fatto che per il displacement è previsto un byte, che a priori consentirebbe di spostarci fino a 255 posizioni "in avanti" rispetto a quella originaria.

Però, per poter gestire valori "negativi" utilissimi ed indispensabili per quanto vedremo nelle prossime puntate, bisogna considerare il contenuto di un byte non come valore esadecimale "puro", ma, come dicono gli inglesi, "signed" e cioè dotato di segno, in logica complementata.

Si considera infatti il bit più significativo del byte come indice del "segno": se è 0 il byte è da considerarsi positivo, viceversa se è 1 il byte è considerato negativo ed il suo valore "assoluto" è ricavabile "complementando a 2" il byte stesso.

Ma vediamo subito di cosa si tratta.

Supponiamo che il byte abbia il valore 23H, espresso in binario come 00100011: il bit più significativo è 0 e perciò il valore è da considerarsi positivo e pari a 2 * 16 + 3 = 35 (decimale).

Se invece il byte ha valore FFH si ragiona in questo modo: FFH si scrive in binario 11111111 e perciò il bit più significativo è 1, indicante un numero negativo.

Ma quanto vale tale numero? Presto detto, si "complementa a 2" tale byte: si "complementano i bit" (e cioè si trasformano tutti gli zeri in uno e gli uno in zeri) e

		displacement	indirizzamento
IX punta a	cella n.1	0	(IX+0)
	cella n.2	1	(IX+1)
	cella n.3	2	(IX+2)
	cella n.4	3	(IX+3)
	cella n.5	4	(IX+4)
	cella n.6	5	(IX+5)
	cella n.7	6	(IX+6)
	cella n.8	7	(IX+7)

Figura 1 - Il registro IX punta alla prima cella (displacement = 0); la seconda cella ha un displacement pari ad 1 e così via.

si aggiunge 1 al risultato. Il valore ottenuto è il "valore assoluto" del numero negativo in questione.

Nel nostro caso, complementando FFH (11111111) otteniamo 0 (00000000) a cui aggiungiamo 1 per ottenere 1: perciò FFH rappresenta "-1" in "logica complemento a 2".

Altro esempio: B2H (10110010), complementato diventa 01001101 (4DH), ed incrementato di 1 dà 4EH che vale 4 * 16 + 4 = 78. Perciò B2H rappresenta - 78.

Vediamo dunque quali sono i valori massimo (positivo) e minimo (negativo) rappresentabili.

Per quanto riguarda i valori positivi il maggior numero è quello che si ottiene con uno zero (segno "+") seguito da 7 "uni" e cioè 01111111 (7FH) a sua volta pari a 127, mentre il minimo è quello formato da tutti zeri e cioè 0.

Per quanto riguarda invece i valori negativi, i casi estremi sono rispettivamente 11111111 (FFH) che abbiamo già visto rappresentare "-1" e 10000000 (80 H) che vale (facciamo un piccolo calcolo!): complementando 80H otteniamo 7F (01111111), aggiungendo 1 otteniamo ancora una volta 80H (magia!) che, tradotto in decimale vale 128. Come dire che 80H rappresenta "-128".

Ecco dunque spiegato il motivo di tali valori per il displacement.

Ritorno alle istruzioni

Torniamo dunque al nostro IX (e al "fratello" IY): secondo quanto visto finora e ricordando le istruzioni che utilizzano la coppia HL abbiamo tutta una nuova serie di istruzioni ognuna contenente un displacement.

Abbiamo perciò che:

a LD r,(HL) corrispondono LD r,(IX + d) e LD r,(IY + d)

mentre

a LD (HL),r corrispondono LD (IX + d),r e LD (IY + d),r

mentre

a LD (HL),n corrispondono LD (IX + d),n e LD (IY + d),n

dove con "r" abbiamo indicato un generico registro (A, B, C, D, E, H, L), con "d" il displacement e con "n" un valore immediato.

Analogamente abbiamo le istruzioni di incremento (INC HL che diventa INC IX e INC IY) e in decremento (DEC HL che diventa DEC IX e DEC IY), che rispettivamente incrementano e decrementano di un'unità il contenuto del registro a 16 bit indicato (la "coppia" HL, oppure IX, oppure ancora IY), come pure istruzioni di caricamento immediato (accanto a LD HL, abbiamo LD IX, nnnn e LD IY, nnnn), dove con "nnnn" abbiamo indicato un valore a 16 bit.

Aggiungiamo ora un ulteriore tipo di indirizzamento che coinvolge dati a 16 bit e perciò 2 celle di memoria e non più una soltanto: possiamo caricare una coppia di registri qualsiasi (BC, DE, HL e naturalmente IX ed IY) con il contenuto di due celle di memoria di indirizzo dato, consecutive. Facendo riferimento alla figura 2 vediamo il significato dell'istruzione

LD HL,(3000)

Con tale istruzione andiamo a considerare le due celle di memoria poste agli indirizzi 3000 e 3001: la cella "3000" contiene il byte meno significativo e la "3001" quello più significativo di un valore a 16 bit.

Tale valore viene posto nella coppia HL. Se come da figura in 3000 e 3001 c'è rispettivamente 20 e 4E, il valore a 16 bit sarà 4E20 e sarà posto come tale in HL, e cioè ponendo "4E" in H e "20" in L.

Analogamente si ha per le coppie BC e DE: per IX e IY invece c'è una piccola differenza.

Infatti l'istruzione LD IX (3000) carica in IX (in toto) il valore 4E20, senza potersi poi isolare (almeno ufficialmente!) la parte più significativa del registro IX da quella meno significativa, indicate nella letteratura con IX_H e IX_L rispettivamente.

Come è abbastanza lecito aspettarsi, esistono le analoghe istruzioni che caricano una coppia di celle consecutive di memoria con il contenuto di una coppia di registri (BC, DE, HL) o dei registri indice (IX, IY).

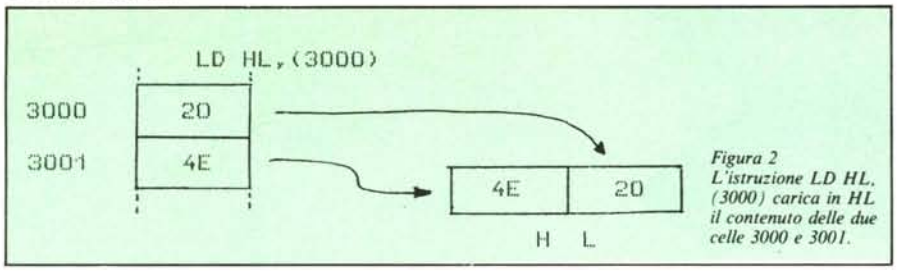

Figura 2
L'istruzione LD HL, (3000) carica in HL il contenuto delle due celle 3000 e 3001.

Ecco che perciò esistono le:

LD (nnnn),dd LD (nnnn),IX LD (nnnn),IY

dove con "nnnn" abbiamo indicato un indirizzo (a 16 bit!) e con "dd" una delle coppie BC, DE, HL.

Se ad esempio D contiene B4H ed E contiene EDH, l'istruzione

LD (804A),DE

pone nelle due celle 804A e 804B rispettivamente i valori ED e B4.

Il terzo registro a 16 bit: SP

Non ce ne vogliano i lettori che sanno già programmare: quella che adottiamo ora è una strategia completamente diversa da quella comune.

Introduciamo perciò il discorso su di un nuovo registro a 16 bit (SP) lo "Stack Pointer", prendendolo come un "puro" registro a 16 bit, completamente accantonando (per questa puntata!) quelle che sono le sue funzioni principali che lo pongono tra i registri più usati (tanto in maniera diretta, quanto indirettamente quasi "senza saperlo").

Evidentemente torneremo sul registro SP quando ne avremo la necessità cioè quando parleremo di "Stack", "Subroutine" e "Salvataggi di registri". Per ora, dicevamo, consideriamo SP come un registro a 16 bit.

Dal punto di vista delle istruzioni viste finora, abbiamo le seguenti:

1) INC SP e DEC SP

che incrementano e decrementano di 1 il contenuto di SP

2) LD (nnnn), SP

che carica le celle nnnn+1 e nnnn con il contenuto di SP

3) LD SP, (nnnn)

carica SP con il contenuto di nnnn+1, nnnn (attenzione ai byte più significativo e meno significativo)

4) LD SP, nnnn

carica SP con il valore nnnn

5) LD SP, HL
LD SP, IX
LD SP, IY

caricano SP con il contenuto di HL, IX, IY rispettivamente

Terminiamo le "LD"

Concludiamo il discorso sulle istruzioni di caricamento occupandoci per un istante dei registri R ed I già citati nella puntata introduttiva. Detto che R si riferisce al complesso meccanismo del "refresh" delle memorie dinamiche e che invece I si riferisce alla gestione degli "interrupt vettorizzati", (dei quali due argomenti parleremo molto più in seguito), abbiamo le ultime 4 istruzioni di caricamento:

LD A,R LD R,A LD A,I LD I,A

come si vede, dato che R e I sono registri particolarissimi, le istruzioni relative sono ben poche e apparentemente oscure.

Con questo concludiamo la puntata e le istruzioni di caricamento.

Prossimamente parleremo delle operazioni algebriche che lo Z80 può compiere; poi vedremo in seguito le istruzioni di trasferimento, quelle di Input/Output, quelle di trasferimento e confronto di blocchi e così via. MC