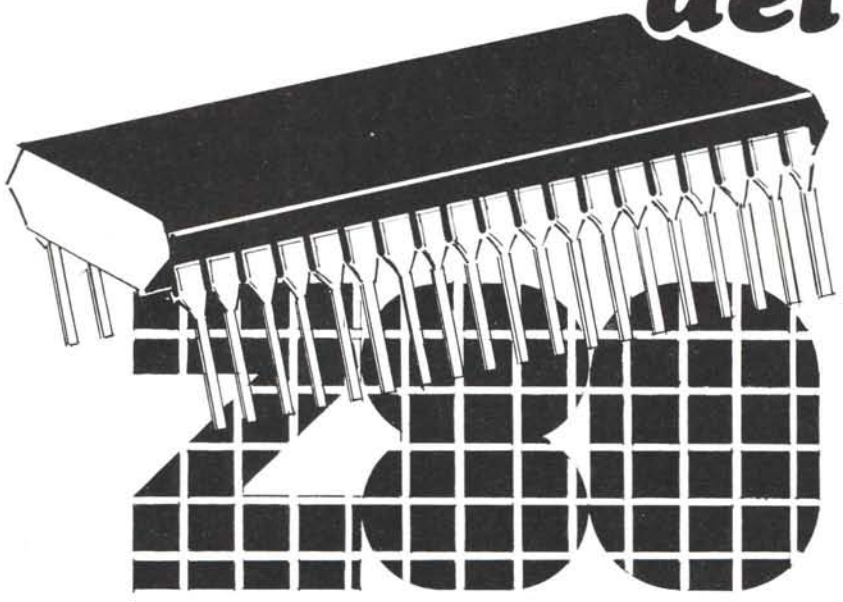
L'ASSEMBLER dello Z80



di Pierluigi Panunzi

prima parte

Iniziamo in questo numero una nuova serie di articoli riguardanti i microprocessori ed in particolare il popolarissimo Z80.

Cercheremo di spiegare nel miglior modo possibile concetti in alcuni casi anche piuttosto complessi, rivolgendoci principalmente a quei lettori (e sono tanti...) che si stanno avvicinando al mondo dei computer e vogliono saperne qualcosa in più.

Non di rado infatti si acquista o si riceve in regalo un personal computer e lo si (sotto-) utilizza quasi esclusivamente come videogame, senza creare nulla di "proprio".

Si può diventare super-esperti nella distruzione di alieni, senza però nemmeno preoccuparsi di conoscere quello che c'è effettivamente "dentro" al nostro personal.

Parecchi, per fortuna, dopo un po' desiderano saperne qualcosa in più: armati di cacciavite aprono il proprio computer-scatola nera, trovando una miriade di componenti elettronici, per lo più circuiti integrati di grosse dimensioni, a 40 se non a 64 piedini.

Per conoscere questi complessi circuiti occorre un po' di buona volontà di apprendimento, qualche cognizione di algebra booleana, un minimo di conoscenze di elettronica e soprattutto questa serie di articoli...

Il microprocessore

Innanzitutto diamo una generica spiegazione del termine: per microprocessore si intende un circuito comprendente un complesso insieme di elementi e funzioni logiche completamente programmabili, al centro delle quali c'è un'unità di calcolo e di supervisione. Gli enormi progressi della tecnologia hanno permesso in questi ultimi anni di raggruppare tutte queste funzioni in un solo circuito; in più la programmazione comporta così enormi potenzialità, che solo dopo un attento studio si possono apprezzare e sfruttare in pieno.

In sostanza il microprocessore (e d'ora in poi ci riferiremo allo Z80) è costituito da un insieme di registri ad 8 o 16 bit, un'unità logico-aritmetica (ALU), un circuito di decodifica e temporizzazione e dei buffer di interfaccia con il mondo esterno (fig. 1).

In parole (molto) povere l'unità logico-aritmetica è quella che si occupa di eseguire un certo tipo di operazioni (somme, sottrazioni, confronti, and, or, shift, operazioni di input/output, ecc.) "su" operandi che possono essere contenuti nei registri interni oppure nella memoria esterna: il tutto al rigoroso scandire di un clock e con la contemporanea ricezione ed emissione di opportuni segnali da e verso il mondo esterno.

Anche se a prima vista può sembrare strano, lo Z80 da solo, anche se correttamente alimentato, non fa niente: necessita di parecchi circuiti ad esso collegati.

Innanzitutto ha bisogno di un circuito di temporizzazione (clock); poi una certa quantità di memoria con relativi circuiti di selezione ed interfacciamento, nonché di un certo numero di porte di interfaccia vera e propria "verso di noi", attraverso una tastiera, un display, un tubo a raggi catodici con relativo controllore, ecc. (fig. 2).

Ecco che senza accorgercene abbiamo descritto la struttura interna di un personal computer...

Abbiamo parlato di memoria: lo Z80 necessita di un certo numero di celle (a 8 bit) per immagazzinare i dati elaborati (fino a 64 kbyte di RAM), nonché di un certo numero di celle contenenti il programma da eseguire (alcune decine di kbyte di ROM o EPROM).

Ed è proprio in questi kbyte di EPROM che risiede quello che si è soliti chiamare il software di base o meglio il firmware dell'apparecchio.

In sostanza questi termini si riferiscono all'insieme di routine, subroutine, tabelle varie, messaggi, che permettono il funzionamento di tutto il sistema: tali routine per l'appunto sono tutte in linguaggio macchina.

Ecco che per capirne il significato ed il tipo di operazioni che compiono, bisogna

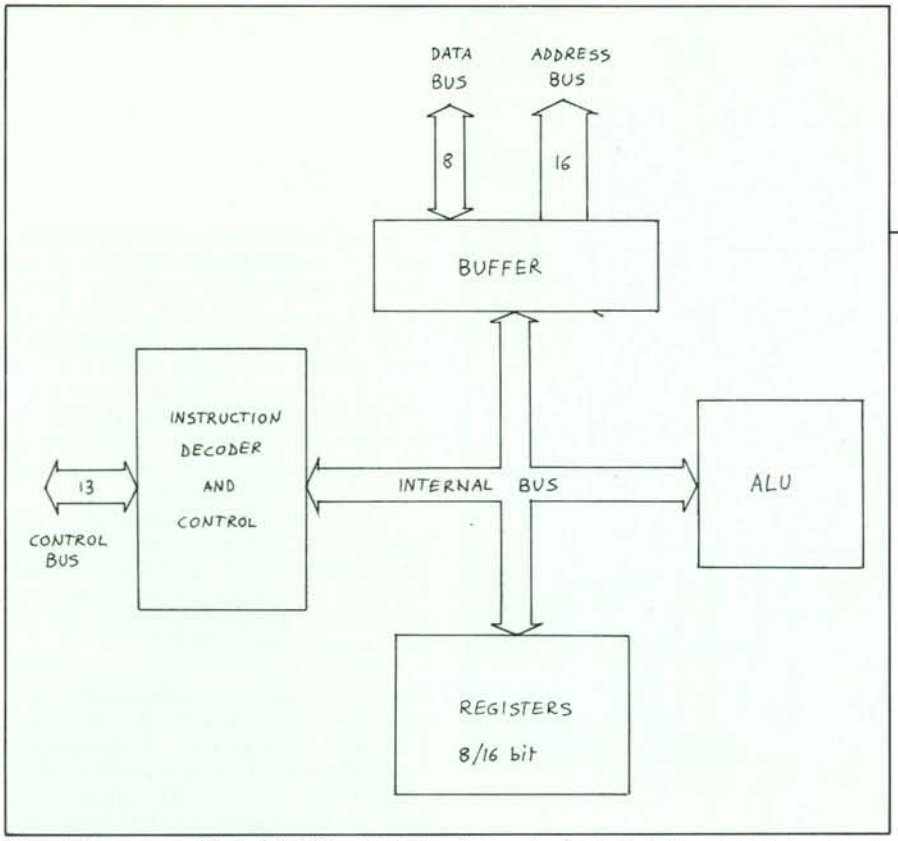


Figura 1 - Struttura semplificata dello Z80, costituito da: un insieme di registri ad 8 o 16 bit, un'unità logico-aritmetica (ALU), un circuito di decodifica istruzioni e di controllo, buffer di interfaccia con componenti esterni.

imparare a lavorare in un linguaggio nuovo, ben diverso ad esempio dal Basic: il linguaggio macchina. Bisogna cioè imparare ad armeggiare con stringhe di 8 bit W (0 e 1) oppure con valori esadecimali, sfruttando le caratteristiche di un linguaggio simbolico, l'assembler. Ancora meglio si tratta di lavorare con un linguaggio a "basso livello", strettamente vicino alla costituzione fisica della macchina (l'hardware), ottenendo programmi strettamente dedicati e velocissimi.

Rassicuriamo subito i lettori: l'assembler si impara più facilmente di quanto si possa credere e le soddisfazioni che se ne ricavano ripagano ampiamente i necessari sforzi preliminari.

Struttura dello Z80

Abbiamo detto che lo Z80 possiede al suo interno alcuni registri: facendo riferimento alla figura 3, vediamo che esistono due "set" di registri "gemelli" (chiamati

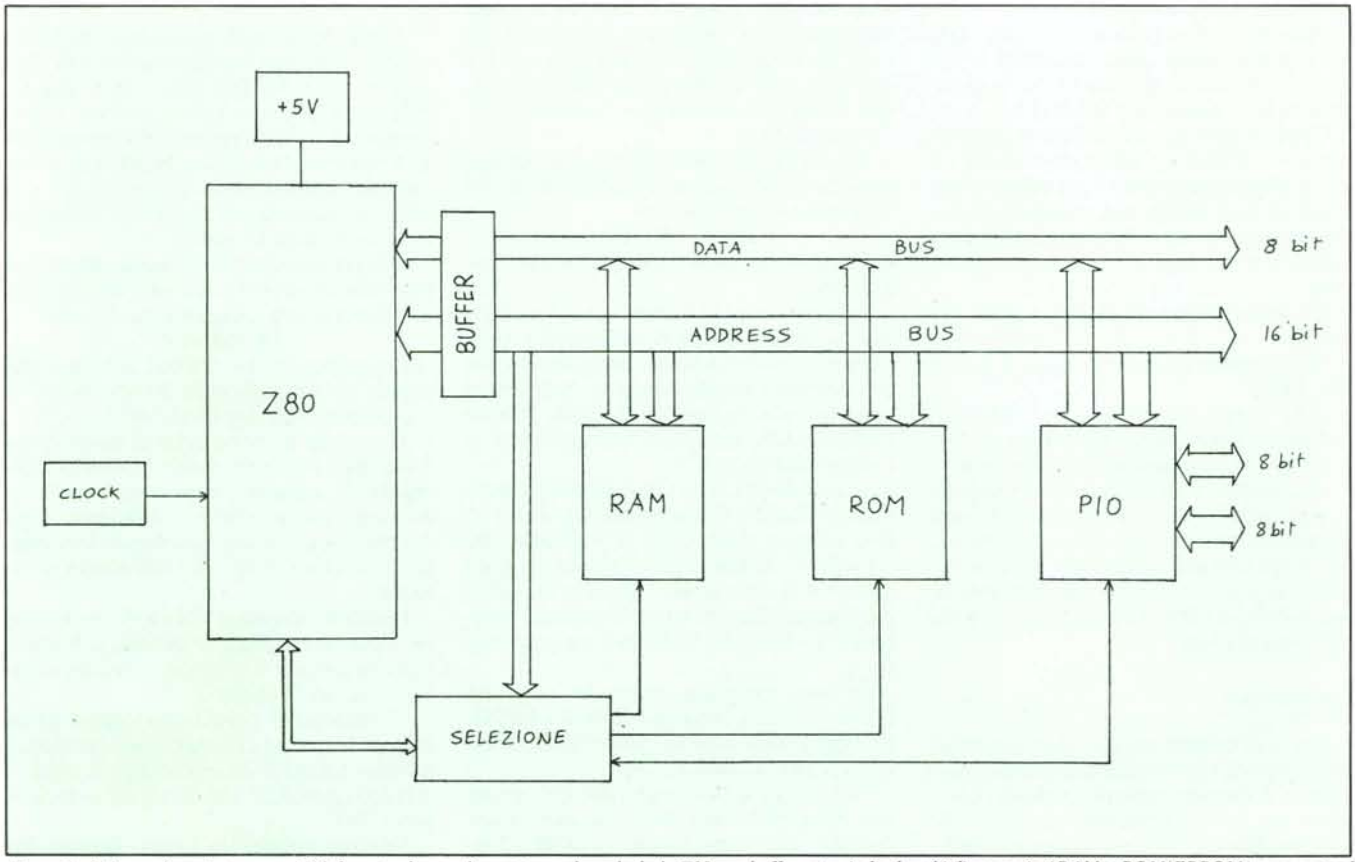


Figura 2 - Schema di un sistema con Z80 formato da: un alimentatore ed un clock, lo Z80 con buffer esterni, due banchi di memoria (RAM e ROM/EPROM), una porta di Input/Output (PIO), circuiti di selezione e decodifica indirizzi.

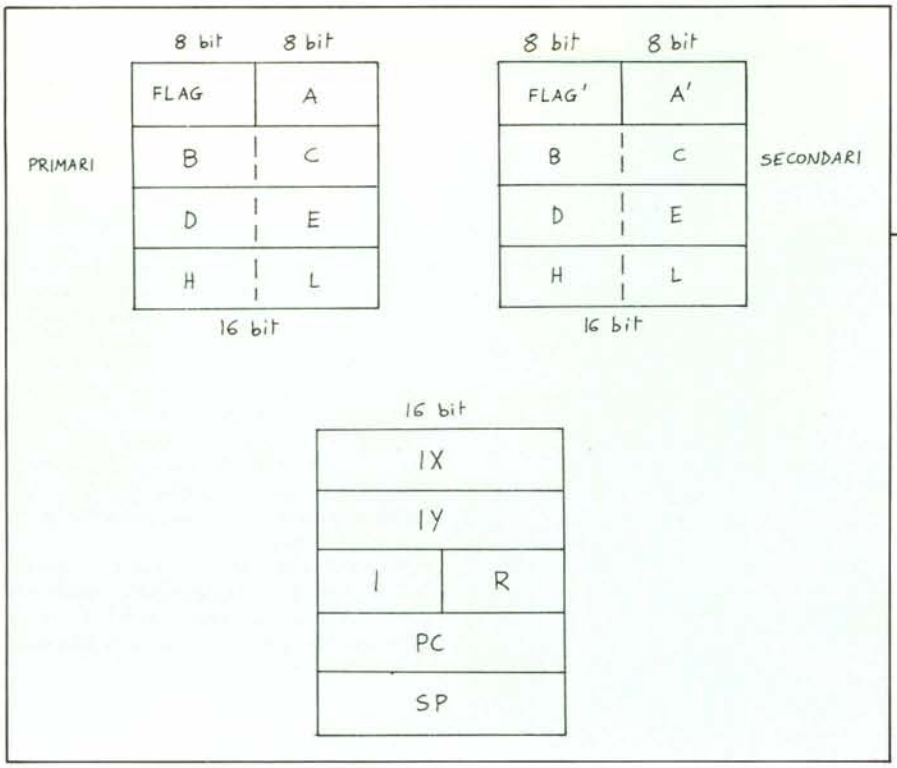


Figura 3 - I registri dello Z80 si dividono in tre gruppi: principali, secondari, di utilità generale.

principali e secondari) più qualche altro.

In particolare abbiamo questi due set formati ognuno da:

— un accumulatore (A) ad 8 bit

— tre coppie di registri a 16 bit (BC, DE, HL) ognuna delle quali scindibile a sua volta in coppie di registri ad 8 bit, arrivando così a 6 registri (B,C,D,E,H,L).

Come si può notare in figura, praticamente "abbinato" all'accumulatore c'è uno speciale registro ad 8 bit contenente i "flag" o segnalatori, che indicano l'avverarsi di certe condizioni durante il funzionamento e dei quali ci occuperemo in seguito.

Per quanto riguarda gli altri registri abbiamo:

— una coppia di registri indice a 16 bit (IX, IY)

— una coppia di registri ad 8 bit (R,I) dedicati a particolarissime funzioni e dei quali non ci occuperemo molto presto

— un registro a 16 bit chiamato Program Counter (PC), che contiene memorizzato istante per istante l'indirizzo di memoria dell'istruzione che lo Z80 sta eseguendo

— un registro a 16 bit rappresentante lo Stack Pointer (SP) che conosceremo nelle prossime puntate.

Un esempio

Prima di entrare nel vivo dell'assembler cerchiamo di capire qual è la filosofia imposta dal microprocessore: volendo compiere una certa operazione vediamo che cosa succede all'interno e all'esterno dello Z80. Come primo esempio prendiamone uno facile facile, un'operazione (banalissima) di caricamento del contenuto di una certa cella di memoria nell'accumulatore.

Innanzitutto questa cella di memoria avrà un certo indirizzo, che per esempio poniamo pari a 100; il suo contenuto (8 bit) avrà un valore qualsiasi compreso tra 0 e 255: è proprio questo valore che desideriamo "estrarre" dalla cella e depositare nell'accumulatore.

Per far eseguire allo Z80 tale operazione, esso dovrà trovare ad un certo punto del programma l'istruzione:

```
LD      A,(100)
```

della quale andiamo subito a vedere il significato.

Innanzitutto "LD" indica che ci si riferisce ad un'operazione di caricamento (LoaD), del quale dobbiamo precisare la cella di partenza e quella di arrivo. Nel nostro caso la cella "sorgente" è quella avente indirizzo 100, mentre la "destinazione" è l'accumulatore ("A").

L'istruzione LD A, (100) significa perciò "carica (LoaD) l'accumulatore (A) con (,) il contenuto della cella di indirizzo 100 ("(100)")". Abbiamo sottolineato che a noi interessa il "contenuto" della cella 100 e ciò è stato indicato "simbolicamente" mettendo il valore dell'indirizzo tra parentesi tonde.

Se non avessimo messo le parentesi ("LD A, 100") avremmo indicato tutt'altra operazione: quella di immettere nell'accumulatore il "valore" 100.

Nel primo caso si parla di operazione con riferimento alla memoria, mentre nel secondo caso si tratta di un cosiddetto "caricamento immediato".

Già da queste due semplici istruzioni vediamo quella che è la struttura di una generica istruzione in assembler. Essa è formata da una prima parte chiamata "Op Code" (Codice Operativo) che rappresenta un comando: nel nostro caso è LD cioè "carica". La seconda parte dell'istruzione prende il nome di "Operand" (Operando) ed indica appunto qual è o quali sono gli operandi sui quali o per mezzo dei quali viene eseguita l'operazione.

Vediamo perciò ora altri esempi di operazioni simili a queste già viste, che prendono il nome di "istruzioni di trasferimento" e che coinvolgono come sorgente e/o destinazione uno dei registri ad 8 bit primari e cioè A,B,C,D,E,H,L.

Possiamo inventarci così una serie di 49 istruzioni differenti, tante quante sono le possibili combinazioni di tali registri presi a coppia: abbiamo così ad esempio LD C,D come pure LD H,B ed anche LD B,A.

Tra l'altro sono previste istruzioni del tipo LD A,A, oppure LD E,E apparentemente inutili (ma nemmeno tanto "apparentemente"....), ma evidentemente esistenti solo per il gioco delle varie possibilità.

Andando avanti possiamo applicare quanto visto per l'accumulatore agli altri registri ad 8 bit (gli ormai noti B,C,D, E,H,L): si possono infatti caricare valori immediati in tali registri (ad esempio LD E, 5 oppure LD B,0), ma NON si possono caricare direttamente i contenuti di locazioni di memoria (ad esempio NON esiste l'istruzione LD D, (408)).

Viceversa, tornando all'accumulatore, è possibile inviarne il contenuto ad una cella di memoria con un'istruzione del tipo:

```
LD (5558),A
```

che significa, come oramai abbiamo già capito, "carica nella cella di indirizzo 5558 il contenuto dell'accumulatore".

C'è subito da notare che in questo caso l'uso delle parentesi tonde è dettato dalla regola di indicare i contenuti di celle di memoria tra parentesi: a differenza però del primo caso visto, eliminando le parentesi tonde non otteniamo una nuova istruzione.

Infatti un'ipotetica LD 44, A non avrebbe alcun senso logico: ancora nessuno è riuscito a porre il contenuto dell'accumulatore in un "valore"...

Concludiamo perciò con questi primi esempi la puntata iniziale: nella prossima daremo un'occhiata ai vari tipi di indirizzamento possibili con il set di istruzioni dello Z80.

Daremo presto, tra l'altro, un risvolto pratico al discorso, riferendoci ad una macchina basata su Z80. MC

COMPUTER SYSTEMS

COMPORRE
SCOMPORRE
CREARE
GIOCARE

Wild By - Rome

RIVENDITORE AUTORIZZATO

apple computer®

ROMA - VIA G. LANZA 101 - 103 - 105 - Tel. 738224-738854 M VIA VITTORIO EMANUELE (linea A) V. CAVOUR (linea B)

OSTIA LIDO - VIA ARISTIDE CARABELLI 108 - 110 - 112 - Tel. 5697686 **SABATO APERTO**

CORSI DI BASIC APPLESOFT