MICROPROCESSORE ZILOG Z80

CARATTERISTICHE DELLO Z80

Lo Z80 è un microprocessore progettato e ideato dall'italiano Federico Faggin nel 1976.

Fu largamente utilizzato in numerosi sistemi come PC (Personal Computer) e console di gioco.

Questo microprocessore è rimasto in produzione fino al 15 giugno 2024, dopo aver avuto una durata di produzione di circa 50 anni.

Lo Z80 è stato creato con l'intenzione di essere retrocompatibile con l'Intel 8080.

Lo Z80 era un microprocessore progettato in Logica Cablata ed inserito in un chip da 40 pin (8 pin del Data Bus, 16 pin del Address Bus e 16 pin del Control Bus).

L'ALU al suo interno, così come i suoi Registri Interni, potevano essere ad 8 o a 16 bit (più rari).

LINGUAGGIO ASSEMBLY

Il Linguaggio Assembly è un linguaggio di programmazione a basso livello e mnemonico. Si tratta infatti di un Linguaggio Macchina che usa dei simboli alfanumerici per la rappresentazione dei codici delle istruzioni, rendendolo più leggibile rispetto ai codifica binaria che usano solitamente i Linguaggi Macchina.

Il Linguaggio Assembly ha inoltre una corrispondenza 1:1 al set del Linguaggio Macchina. Ciò vuol dire che ad ogni istruzione del Linguaggio Macchina corrisponde un'istruzione del Linguaggio Assembly.

Le istruzioni del microprocessore Z80 possono avere lunghezze diverse, tant'è che alcune sono brevi ed occupano solamente 1 byte, mentre altre possono essere più lunghe arrivando ad occupare fino a 4 byte.

Più lunghe sono le istruzioni, più informazioni contiene, ma potrebbe richiedere quindi più tempo per essere elaborata.

L'esecuzione di un'istruzione avviene in tre fasi:

- Fetch, che legge l'istruzione che andrà svolta;
- Decode, che decodifica ed interpreta questa istruzione;
- Execute, che esegue l'istruzione vera e propria.

Il tempo necessario per completare un'istruzione si misura in cicli di clock.

Alcune istruzioni richiedono un durata complessiva del ciclo Fetch-Decode-Execute di soli 4 cicli di clock per essere svolte, mentre le istruzioni più complesse possono arrivare anche ad una durata di 20 cicli di clock.

Ovviamente, meno cicli di clock impiegano e più saranno veloci le istruzioni ad essere eseguite, e viceversa.

Assembly è un linguaggio non case sensitive, ovvero che non fa distinzioni tra lettere minuscole e maiuscole.

Inoltre, è bene evidenziare il fatto che con "Assembly" ci riferiamo al linguaggio di programmazione a basso livello, mentre con "Assembler" ci riferiamo al compilatore che produce il Linguaggio Assembly.

REGISTRI INTERNI AD 8 BIT

A	F
B	C
D	E
H	L

Il Registro Interno A è un Accumulatore molto usato dalla ALU. Questo Registro Interno è quasi sempre utilizzato e coinvolto in tutte le operazioni che vengono svolte dalla ALU.

Il Registro Interno F è un registro dei Flag, che sono degli indicatori booleani che contengono le informazioni sul risultato delle operazioni eseguite dall'ALU.

Gli altri Registri Interni rimanenti possono formare delle coppie specifiche se messi insieme, così costruite:

- Coppia dei Registri Interni BC;
- Coppia dei Registri Interni DE;
- Coppia dei Registri Interni HL.

Presi singolarmente, questi Registri Interni sono dei normalissimi registri ad 8 bit.

I Registri Interni A e F vengono considerati come registri specializzati, mentre i Registri Interni B, C, D, E, H, e L vengono considerati dei registri generici.

Dobbiamo ricordarci anche che la Parte High (la colonna dove è presente il registro H) viene considerata la Parte MSb (Most Significant bit), mentre la Parte Low (la colonna in cui è presente il registro L) viene considerata la Parte LSb (Least Significant bit).

REGISTRI INTERNI A 16 BIT

PC
SP

I Registri Interni PC (Program Counter) e SP (Stack Pointer) sono due registri specializzati che indicano la posizione in memoria primaria della prossima istruzione che il Linguaggio Macchina deve eseguire.

IX
IY

I Registri Interni IX e IY sono anch'essi dei registri a 16 bit, ma che però sono generici (e non specializzati come i due sopra). Questi hanno il compito principale di contenere gli indirizzi di memoria primaria delle istruzioni che dovranno essere eseguite.

Esistono anche due particolari meccanismi, gli Interrupt ed i Refresh.

Gli Interrupt gestiscono gli eventi importanti e temporanei (come un tasto della tastiera che viene premuto), facendo interrompere l'istruzione che lo Z80 stava eseguendo (mettendola in secondo piano) ed iniziando a gestire la nuova istruzione. Una volta eseguita completamente la nuova istruzione, lo Z80 ritorna ad eseguire la prima istruzione che aveva interrotto.

I Refresh servono invece a mantenere i dati salvati nella memoria per evitare che vengano cancellati e persi. Questi dati vengono però cancellati se non viene aggiornata la memoria in cui sono contenuti.

FLAG EI, DI, IFF1 E IFF2

Un flag è un indicatore di stato booleano che specifica una condizione specifica all'interno di un sistema informatico.

Il microprocessore Z80 dispone di una raccolta di flag molto utili e particolari per il corretto svolgimento delle normali azioni che deve eseguire il microprocessore. Questi sono i flag EI, DI, IFF1 e IFF2.

Il Flag EI (Enable Interrupt), che ha il compito di abilitare gli Interrupt (se disabilitati).

Il Flag DI (Disable Interrupt), che ha il compito di disabilitare gli Interrupt, ed ignora quindi tutti gli Interrupt fino a quando non viene eseguito un comando del Flag EI.

Il Flag IFF1, così come il Flag IFF2, sono due particolari flag che indicano lo stato degli Interrupt (ci dicono quindi se questi sono abilitati o disabilitati).

TABELLA DELLE ISTRUZIONI DI LOAD

La prima colonna definisce la sintassi dell'istruzione in Linguaggio Assembly.

La seconda colonna definisce l'azione che svolge tale istruzione.

La terza colonna definisce i vari tipi di flag presenti nello Z80:

- Il Flag S (Sign Flag) ci indica il segno del risultato di un'operazione.
- Il Flag Z (Zero Flag) ci indica se il risultato di un'operazione è nullo o no.
- Il Flag H (Half Carry Flag) ci indica se è avvenuto un riporto dal quarto al quinto bit durante un'operazione di somma;
- Il Flag P/V (Parity/Overflow Flag) ci indica se il risultato di un'operazione supera il valore massimo rappresentabile (Overflow Flag) o se il numero di bit 1 nel risultato è pari (Parity Flag);
- Il Flag N (Add/Subtract) ci indica il tipo di operazione che è appena stata eseguita (si setta ad 1 se è una sottrazione, mentre si setta a 0 se è un'addizione);
- Il Flag C (Carry Flag) ci indica se c'è stata la presenza di un prestito o di un riporto nel bit più significativo durante un'operazione.

Guardando la tabella possiamo notare la presenza di due flag riservati, ovvero che non vengono utilizzati).

La quarta colonna definisce la rappresentazione binaria del Linguaggio Macchina.

La quinta colonna definisce invece la rappresentazione della codifica con simboli alfanumerici del Linguaggio Assembly.

La sesta colonna definisce il numero di byte che compiono tale istruzione.

La settima colonna definisce il numero di cicli macchina (rappresentano il tempo necessario per completare una singola operazione interna dello Z80) richiesti per poter eseguire l'istruzione.

Infine, l'ottava colonna definisce il numero di stati di clock (battiti di clock) richiesti per eseguire l'istruzione.

ISTRUZIONI DI LOAD

Le Istruzioni di Load (LD) sono delle istruzioni di caricamento. Svolgono quindi lo spostamento dei dati ed il caricamento dei valori della memoria in un determinato registro.

METODI DI INDIRIZZAMENTO

Un metodo di indirizzamento è una tecnica utilizzata dallo Z80 per individuare l'indirizzo della sorgente o della destinazione dei dati durante l'esecuzione di un'istruzione.

Esistono diversi metodi di indirizzamento, alcuni dei quali sono:

- **Indirizzamento Diretto a Registro**: copia il valore da un registro r' ad un altro registro r.
  - `LD r, r'`

- **Indirizzamento Indiretto a Registro**: il registro HL contiene un indirizzo di memoria, da cui viene letto il valore. Il simbolo "(…)" sta ad indicare che stiamo accedendo all'indirizzo di memoria.
  - `LD r, (HL)`
  - `LD (HL), r`
  - `LD (HL), n`
  - `LD A, (BC)`
  - `LD A, (DE)`
  - `LD (BC), A`
  - `LD (DE), A`

- **Indirizzamento Diretto a Memoria**: l'indirizzo di memoria del valore nn è direttamente specificato nell'istruzione, caricando poi tale valore nel registro A.
  - `LD A, (nn)`
  - `LD (nn), A`

- **Indirizzamento Indiretto a Memoria**: questo metodo utilizza un registro (tipicamente HL, IX o IY) che contiene l'indirizzo di memoria dal quale leggere o nel quale scrivere un valore a 8 o 16 bit.

- **Indirizzamento Immediato**: carica un valore immediato n in un registro r.
  - `LD r, n`
  - `LD (HL), n`
  - `LD (IX + d), n`
  - `LD (IY + d), n`

- **Indirizzamento con Offset**: usa il registro IX con un offset d per accedere alla memoria.
  - `LD r, (IX + d)`
  - `LD r, (IY + d)`
  - `LD (IX + d), r`
  - `LD (IY + d), r`
  - `LD (IX + d), n`
  - `LD (IY + d), n`

- **Indirizzamento Implicito**: in registro coinvolto è sottinteso nell'istruzione, ed è usato per le operazioni con i registri I e R.
  - `LD A, I`
  - `LD A, R`
  - `LD I, A`
  - `LD R, A`

- **Indirizzamento Relativo**: questo metodo viene utilizzato nelle istruzioni di salto, dove l'indirizzo di destinazione è calcolato come un offset rispetto al PC. L'offset è un valore a 8 bit con segno, quindi permette di saltare avanti o indietro fino a 127 byte.

Queste istruzioni di LD si riferiscono alla parte ***8-BIT LOAD GROUP***, ovvero le istruzioni il caricamento e lo spostamento di dati a 8 bit tra registri e memoria nello Z80.

16-BIT LOAD GROUP

Sono le istruzioni che gestiscono il caricamento e lo spostamento di dati a 16 bit tra registri e memoria nello Z80.

- LD dd, nn: carica un valore immediato a 16 bit (nn) nel registro a 16 bit dd (dove dd può essere BC, DE o HL). Utilizza in metodo di Indirizzamento Immediato.
- LD IX, nn: carica un valore immediato a 16 bit (nn) nel registro IX, utilizzando un metodo di Indirizzamento Immediato.
- LD IY, nn: carica un valore immediato a 16 bit (nn) nel registro IY, utilizzando un metodo di Indirizzamento Immediato.
- LD HL, (nn): carica nel registro HL il valore a 16 bit presente all'indirizzo di memoria specificato da (nn), utilizzando un metodo di Indirizzamento Indiretto a Memoria.
- LD dd, (nn): carica nel registro a 16 bit dd il valore presente all'indirizzo di memoria specificato da (nn), utilizzando un metodo di Indirizzamento Indiretto a Memoria.
- LD IX, (nn): carica nel registro IX il valore a 16 bit presente all'indirizzo di memoria specificato da (nn), utilizzando un metodo di Indirizzamento Indiretto a Memoria.
- LD IY, (nn): carica nel registro IY il valore a 16 bit presente all'indirizzo di memoria specificato da (nn), utilizzando un metodo di Indirizzamento Indiretto a Memoria.
- LD (nn), HL: memorizza il contenuto del registro HL all'indirizzo di memoria specificato da (nn), utilizzando un metodo di Indirizzamento Indiretto a Memoria.
- LD (nn), dd: memorizza il contenuto del registro a 16 bit dd all'indirizzo di memoria specificato da (nn), utilizzando un metodo di Indirizzamento Indiretto a Memoria.
- LD (nn), IX: memorizza il contenuto del registro IX all'indirizzo di memoria specificato da (nn), utilizzando un metodo di Indirizzamento Indiretto a Memoria.
- LD (nn), IY: memorizza il contenuto del registro IY all'indirizzo di memoria specificato da (nn), utilizzando un metodo di Indirizzamento Indiretto a Memoria.

GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS

Questi sono delle istruzioni base che gestiscono operazioni aritmetiche semplici e il controllo dello Z80.

Analizzandoli singolarmente possiamo trovare diversi tipi di istruzioni:

- CPL (**Complement Accumulator**): fa il NOT logico dei bit, facendo quindi il Complemento a 1 dei bit (inverte gli 0 con 1 e viceversa);
- NEG (**Negate Accumulator**): fa la negazione del numero, facendo quindi il Complemento a 2 dei bit;
- CCF (**Complement Carry Flag**): Inverte il Flag C (se era settato a 0 diventa 1, e viceversa);
- SCF (**Set Carry Flag**): setta il Flag C a 1;
- NOP (**No Operation**): non esegue nessuna operazione, ma semplicemente avanza al successivo ciclo di istruzione;
- HALT: Arresta temporaneamente il microprocessore;