

ESERCITAZIONE 8

Sommario

- Unità centrale di elaborazione
- Meccanismo di interruzione
- CPU-Memoria
- Architettura di una CPU
- Linguaggio macchina
- Modi di indirizzamento

1. Unità centrale di elaborazione

L'unità centrale di elaborazione (*CPU*) ha il compito di eseguire un programma (*sequenza di istruzioni*) contenuto in memoria.

1.1. Ciclo base di un'istruzione

L'esecuzione di un'istruzione è costituita da due operazioni elementari:

- fetch;
- esecuzione.

Durante la fase di fetch la CPU preleva dalla memoria il codice macchina dell'istruzione da eseguire.

Durante la fase di esecuzione la CPU decodifica l'istruzione, se necessario preleva dalla memoria i dati, esegue lo svolgimento dell'istruzione ed eventualmente scrive in memoria il risultato dell'operazione.

1.2. Meccanismo di interruzione

Il percorso normale delle operazioni di una CPU può essere interrotto da eventi esterni aventi priorità più elevata del programma in corso di esecuzione.

Nel caso di interruzione, la CPU termina l'istruzione corrente, salva lo stato del processo in corso di esecuzione e passa il controllo alla procedura di gestione dell'interruzione.

Al termine della procedura di gestione dell'interruzione, la CPU riprende l'esecuzione del processo interrotto.

1.2.1. Possibili eventi di interruzione

1. mancanza della tensione di alimentazione

2. errore h/w
3. errore s/w (divisione per 0)
4. operazioni di Input/Output.

2. CPU - Memoria

La CPU è un componente unico all'interno di un sistema di calcolo e dunque deve essere realizzata con la migliore tecnologia possibile (anche costosa)

La memoria principale (di grande capacità) deve essere realizzata con la tecnologia più conveniente economicamente.

Il tempo di ciclo della CPU è il tempo richiesto per eseguire la più veloce operazione della CPU e può variare dai 100 ai 300nanosecondi.

Il tempo di accesso alla memoria corrisponde al tempo minimo tra due accessi consecutivi in memoria e può variare da 200 a 1000nanosecondi.

Nel tempo migliorano le prestazioni della CPU. Il tasso di variazione della velocità delle CPU è migliore rispetto al tasso di variazione dei tempi di accesso della memoria. Un problema critico è relativo allo scambio di informazione tra CPU e memoria.

Per risolvere tale problema è possibile:

- utilizzare i registri interni della CPU
- tecniche che aumentano l'efficienza (pre-fetching);
- livelli gerarchici di memoria.

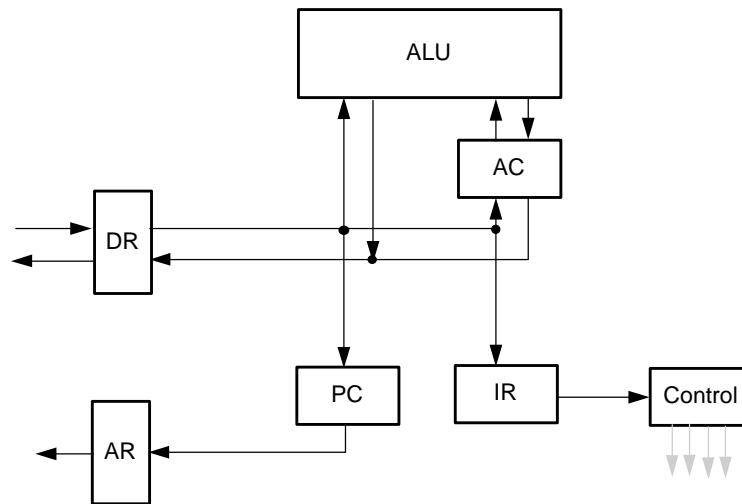
L'utilizzo dei registri interni permette di ridurre lo scambio di informazione con l'esterno: le operazioni tra i registri interni avvengono con tempi dell'ordine del tempo di ciclo della CPU.

Per minimizzare i tempi di attesa delle fasi di lettura delle istruzioni dalla memoria è possibile caricare preventivamente le istruzioni, sfruttando i periodi di inattività del canale di comunicazione tra la CPU e la memoria. All'interno della CPU viene memorizzata una coda di istruzioni contenente le istruzioni precaricate dalla memoria. Non è sempre detto che le istruzioni caricate preventivamente siano effettivamente eseguite.

Per migliorare i tempi di risposta degli accessi dalla memoria in molti sistemi viene inserita tra la memoria principale e la CPU un secondo tipo di memoria più veloce, detta *memoria cache*. La memoria cache costituisce un deposito temporaneo dei dati che la CPU utilizza più frequentemente. La memoria principale è generalmente di grande dimensione e normalmente non è molto veloce; la memoria cache è di piccola dimensione, ma è generalmente molto più veloce.

Le prestazioni della CPU sono calcolati attraverso indicatori statistici che indicano il numero di milioni di istruzioni al secondo (MIPS) ed il numero di milioni di istruzioni in virgola mobile al secondo (MFLOPS).

3. Architettura di una CPU



Registro DR (data register): memorizza i dati provenienti dalla memoria e diretti in memoria.

Registro AR (address register): memorizza gli indirizzi per accedere in memoria.

Registro PC (program counter): memorizza l'indirizzo della prossima istruzione da eseguire.

Registro IR (instruction register): memorizza il codice dell'istruzione da eseguire; è collegato all'unità di controllo. Sulla base del codice memorizzato in IR l'unità di controllo eseguirà le operazioni opportune per permettere la corretta esecuzione dell'istruzione.

Modulo ALU (unità aritmetico-logica): esegue le operazioni aritmetiche e logiche.

Registro ACC (accumulatore): immagazzina i dati in ingresso ed in uscita all'ALU.

3.1. Ciclo base di un'istruzione

Fase di Fetch

- $AR \leftarrow PC$
- $DR \leftarrow M(AR)$
- $IR \leftarrow DR(\text{opcode})$

Fase di esecuzione

- $AR \leftarrow DR(\text{address})$
- $DR \leftarrow M(AR)$
- $AC \leftarrow AC + DR$
- $PC \leftarrow INC(PC)$

3.2. Architettura interna ed esterna

Si distingue l'architettura interna e quella esterna.

L'architettura interna è costituita dalla struttura interna della CPU. Per la progettazione dell'architettura interna occorre valutare il miglior compromesso tra le prestazioni ed i costi avendo come vincolo la tecnologia.

L'architettura esterna è strettamente legata alla programmazione e fa riferimento a come il processore è visto da chi lo deve programmare: insieme delle istruzioni, registri, modi di indirizzamento, tipi di dato ammessi dalle istruzioni.

4. Linguaggio macchina

Il linguaggio macchina è costituito dai comandi eseguiti dalla CPU.

Si distinguono due possibili rappresentazioni:

- codice macchina: 1011010
- codice mnemonico: MOV A, B

Corrispondenza uno ad uno tra i due codici. Il programma assemblatore converte ogni codice mnemonico nel corrispondente codice macchina.

L'insieme delle istruzioni espresse come codice mnemonico rappresenta il linguaggio *assembler*.

4.1. Istruzioni macchina

Ogni istruzione deve definire:

- operazione da svolgere;
- operandi coinvolti;
- posizione dell'istruzione successiva.

Le istruzioni sono eseguite in sequenza, dunque l'informazione della prossima istruzione da eseguire è resa implicita: salvo eccezioni l'istruzione successiva è l'istruzione successiva in sequenza. Il Program Counter memorizza l'indirizzo della prossima istruzione da eseguire. Per alterare la sequenza devono essere introdotte istruzioni di salto (PC←X).

Il codice macchina è suddiviso in campi:

Codice Operativo	I operando	II operando
------------------	------------	-------------

Corrispondente codice mnemonico:

- ADD A, B

ADD: codice operativo

A e B: operando.

Il codice mnemonico rispecchia la struttura del codice macchina.

5. Modi di indirizzamento

Il modo di indirizzamento specifica come viene localizzata la posizione del dato. Ogni operando è associato ad un dato, di cui occorre conoscere la localizzazione attraverso il suo indirizzo. I modi di indirizzamento permettono di indicare la posizione del dato.

Si distinguono i seguenti modi di indirizzamento elementari:

- immediato;
- diretto;
- indiretto.

5.1. Modo di indirizzamento immediato

Il dato è contenuto nel codice macchina.

Esempio:

MOV A, #99

La precedente istruzione copia nel registro A il valore decimale 99.

MOV	A	99
-----	---	----

5.2. Modo di indirizzamento diretto

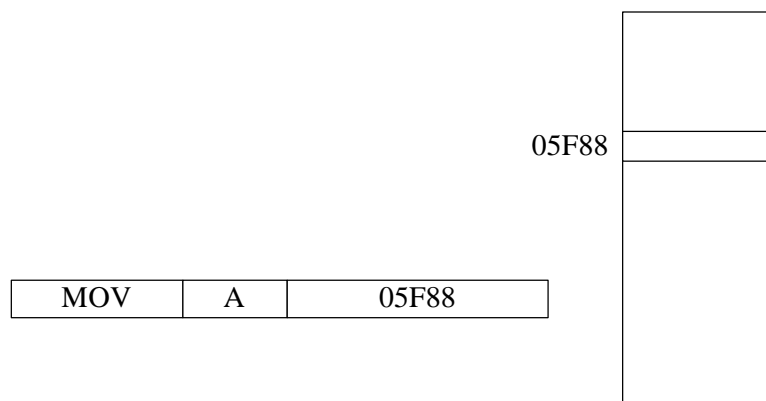
Il codice macchina contiene l'indirizzo del dato.

Esempio:

MOV A, X

X può essere:

1. l'indirizzo della cella di memoria che contiene il dato (occorre precisare nel codice macchina l'indirizzo);
2. il codice del registro interno (sono sufficienti pochi bit per codificare il registro).



5.3. Modo di indirizzamento indiretto

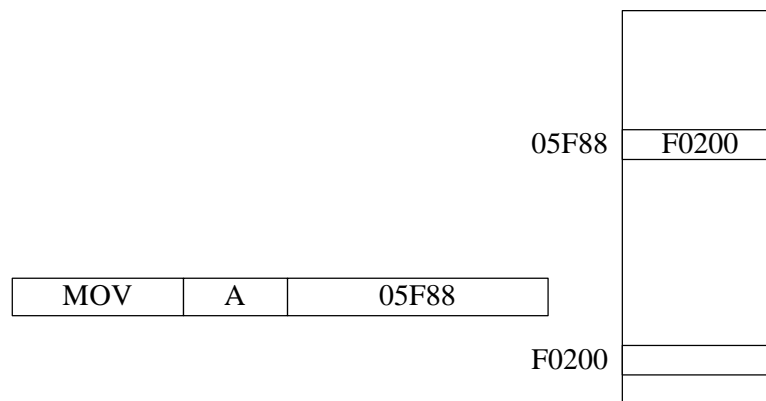
Il codice contiene l'indirizzo della locazione di memoria nella quale è contenuto l'indirizzo del dato.

Esempio:

MOV A, (X)

X può essere:

1. l'indirizzo della cella di memoria che contiene il dato;
2. il codice del registro interno.



5.4. Tipi di indirizzo

Il valore dell'indirizzo può essere espresso in modo:

- assoluto;
- relativo.

Con un indirizzo di tipo assoluto l'indirizzo completo compare nel campo operando. Lo svantaggio è legato alla lunghezza del campo indirizzo: genera un codice di dimensione elevata.

Nel caso di indirizzo di tipo relativo, nel campo operando compare solo lo spiazzamento relativo al contenuto in quell'istante nel registro PC. Lo spiazzamento può essere:

- contenuto in un byte (+127, -128)
- contenuto in una word (+32k, -32k).

Utilizzando un indirizzamento di tipo relativo non si può accedere a qualunque dato all'interno della memoria, ma permette di risparmiare nella lunghezza del codice.

[Torna al Sommario](#)