Architettura degli elaboratori

UniVR - Dipartimento di Informatica

Fabio Irimie

2° Semestre 2023/2024

Indice

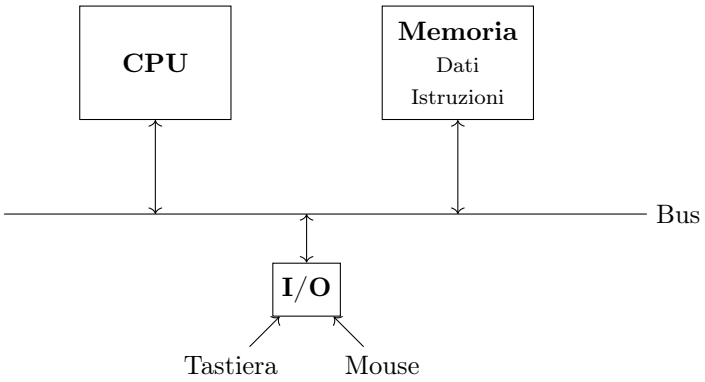
1 Architettura di Von Neumann

L'esigenza era quella di avere un'architettura che permettesse di eseguire programmi in modo automatico, senza dover cambiare il cablaggio del calcolatore, quindi il circuito deve essere abbastanza generale per poter eseguire programmi diversi.

1.1 Struttura

L'architettura di Von Neumann è composta da 5 parti principali:

- **Unità aritmetico-logica**: si occupa di eseguire le operazioni aritmetiche e logiche
- **Unità di controllo**: si occupa di controllare il flusso delle istruzioni
- **Memoria**: contiene i dati e le istruzioni
- **Input/Output**: permette di comunicare con l'esterno
- **Bus**: permette di trasferire i dati tra la memoria e l'unità aritmetico-logica (generalmente in oro)



1.2 Caratteristiche

Le istruzioni hanno bisogno di un'operazione che permetta di effettuare dei salti, in modo da poter implementare i cicli e le strutture di controllo. Inoltre le istruzioni devono essere eseugite in sequenza (un'istruzione alla volta).

1.3 CPU

Ogni processore ha un'set di istruzioni diverso in base all'architettura, questo set di istruzioni è chiamato **ISA** (Instruction Set Architecture). **Assembly** è un linguaggio di programmazione che permette di scrivere programmi in base all'ISA del processore e questo linguaggio viene tradotto in linguaggio binario attraverso un **assembler**. In questo corso viene usata l'architettura x86 (80x86).

1.3.1 Modello semplificato

Per rappresentare il funzionamento di un processore si può usare un modello semplificato rappresentato ad alto livello. Questo modello è composto da:

- **Central Processing Unit (CPU)**: esegue le istruzioni
- **Control Unit (CU)**: controlla il flusso delle istruzioni
- **Bus Dati (BD)**: trasferisce i dati alla CPU
- **Bus Istruzioni (BI)**: trasferisce le istruzioni alla CPU
- **Bus di Controllo (BC)**: trasferisce i segnali di controllo
- **Memory Address Register (MAR)**: contiene l'indirizzo di memoria da leggere
- **Memory Data Register (MDR)**: contiene i dati letti dalla memoria
- **Program Counter (PC)**: tiene conto dell'indirizzo dell'istruzione da eseguire
- **Instruction Register (IR)**: contiene l'istruzione corrente
- **Program Status Word (PSW)**: contiene i flag del processore (es. zero, carry, overflow)
- **Register File**: contiene i registri del processore (es. EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP)
- **Arithmetic Logic Unit (ALU)**: esegue le operazioni aritmetiche e logiche

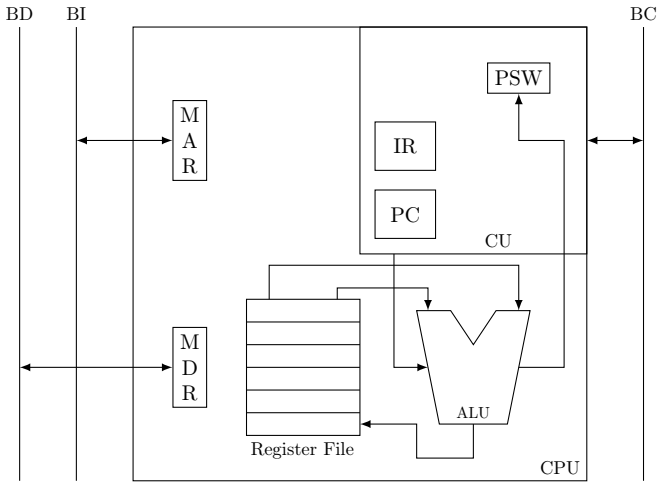


Figura 1: Struttura di un processore

Il flusso di esecuzione delle istruzioni è il seguente:

- **Fetch**: CU legge l'istruzione dalla memoria
- **Decode**: CU decodifica l'istruzione
- **Execute**: ALU esegue l'istruzione

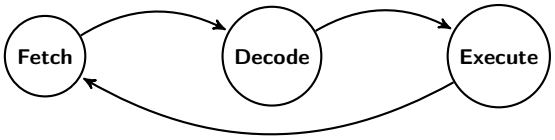
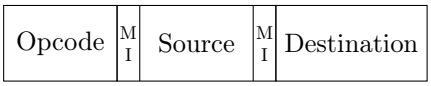


Figura 2: Flusso di esecuzione delle istruzioni

1.3.2 Istruzioni

Ogni istruzione è codificata nel seguente modo:



Le istruzioni più comuni sono:

- **MOV** (o MOVL): copia il contenuto di un registro in un altro

1.3.3 Metodi di indirizzamento

I metodi di indirizzamento (MI) sono diversi modi per accedere ai dati in memoria, i più comuni sono:

- **Registro**: Un'istruzione può accedere direttamente ai registri ad esempio: `MOVL %EAX, %EBX`
- **Immediato**: Un'istruzione può accedere direttamente ai dati ad esempio: `MOVL $10, %EBX`
- **Assoluto**: Un'istruzione può accedere direttamente ai dati ad esempio: `MOVL DATO, %EBX`
  dove `DATO` è un'etichetta che punta ad un'indirizzo di memoria
- **Indiretto Registro**: Un'istruzione può contenere un registro che punta ad un altro registro ad esempio: `MOVL (%EAX), %EBX`
- **Indiretto Registro con Spiazzamento**: Un'istruzione può mettere un offset rispetto al registro contenuto nell'istruzione ad esempio: `MOVL $8(%EAX), %EBX`