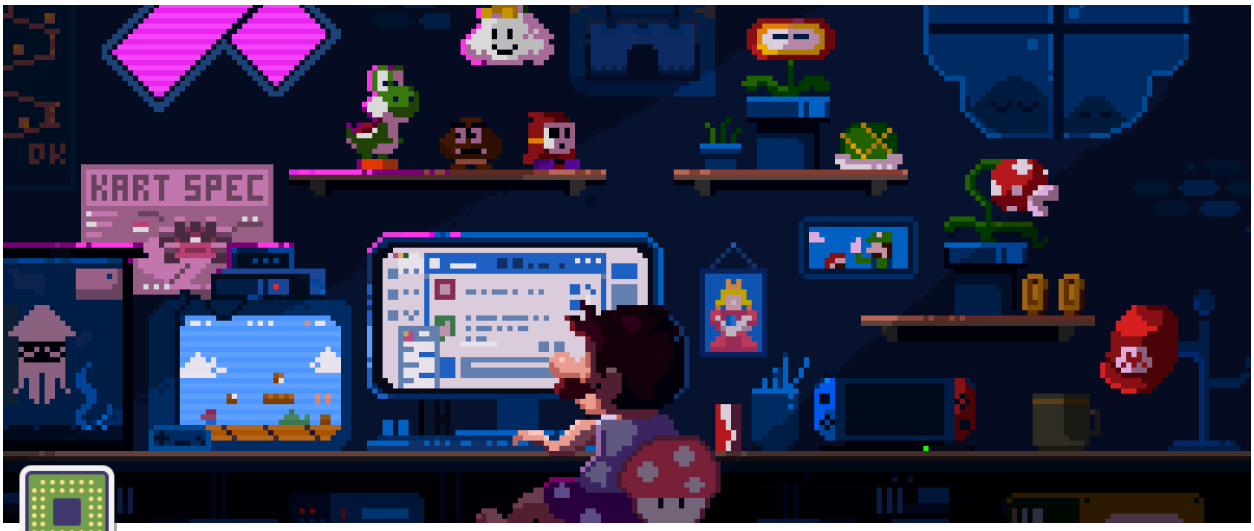


Introduzione all’architettura MIPS

▼ ***INDICE***

1 Introduzione all’architettura MIPS

1.1 Istruzioni, Assemblare e Compilare

Per poter comunicare con un sistema elettronico dobbiamo semplicemente inviare dei segnali elettrici.

Esistono solo 2 tipi di segnali elettrici:

- 1 che rappresenta un segnale attivo;
- 0 che rappresenta un segnale spento.

Quest’ultimo prendono il nome di sistema numerico binario, ed ogni cifra binaria($1\ o\ 0$), prende il nome di bit.

I primi programmatori comunicavano con i calcolatori utilizzando i numeri binari per dare istruzioni da svolgere.

Questo metodo come si può ben capire non era molto efficiente visto che è molto complesso e anche un semplice errore, come un bit sbagliato, può cambiare totalmente l'output.

Per questo utilizzarono i calcolatori stessi per programmare altri calcolatori, questo portò alla creazione dei programmi in grado di tradurre delle notazioni simboliche più semplici da utilizzare in vere e proprie istruzioni.

Questi programmi presero il nome di assemblatori.

📎 ***ESEMPIO DI COMANDO:***

Un esempio può essere la seguente istruzione:

```
add A, B
```

che viene tradotta dall'assemblatore come:

$$1000110010100000$$

In questo caso stiamo comunicando al calcolatore di sommare il numero A il numero B. Questo linguaggio simbolico viene detto linguaggio Assembler(o Assembly).

1.2 Architettura di Von Neumann, CPU e Memorie

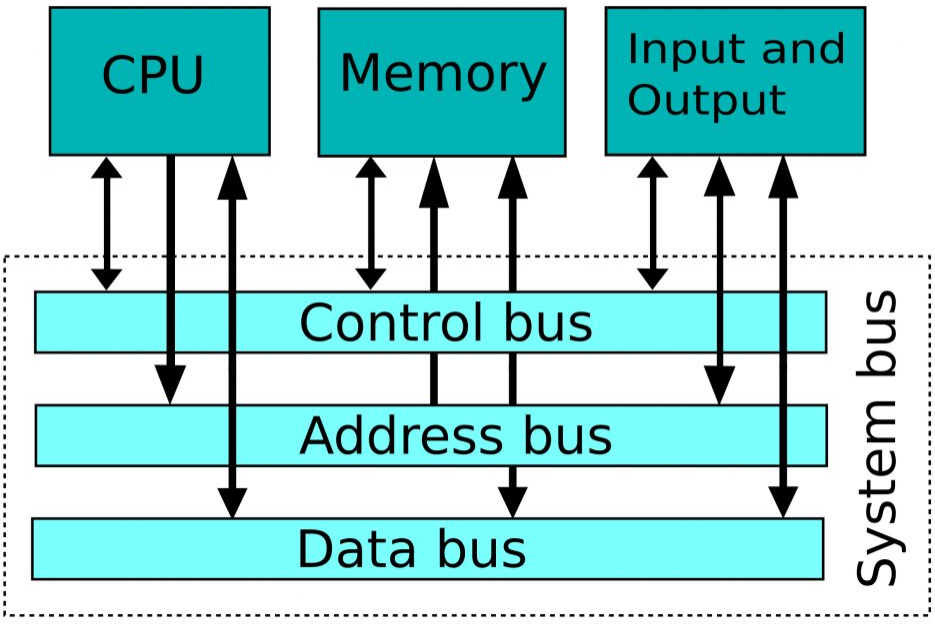
Un esempio classico di architettura generica di un computer è l'architettura di Von Neumann.

Tale modello prevedeva che il calcolatore dovesse essere costituito da 4 elementi fondamentali:

- CPU: ossia l'unita centrale di un elaboratore, anche detto processore. Si occupa di eseguire una per volta le istruzioni. Quest'ultima è a sua volta costituita da 3 elementi:

    - Control Unit(CU): che va a coordinare e svolgere le operazioni;
    - Arithmetic Logic Unit(ALU): svolge le operazioni aritmetiche e logiche;
    - Registri: ossia piccole memorie utilizzate per immagazzinare dati temporanei.
- Memoria: permette di memorizzare le istruzioni e i dati;
- Periferiche di input/output: che permettono al pc di comunicare con l'esterno;
- Bus di sistema: ossia un canale unico di comunicazione fra tutti i componenti.

  È suddiviso in 3 sotto-canali:
    - Control-bus: dove vengono comunicati i segnali di controllo, permettendo così ai componenti di coordinarsi;
    - Address-bus: dove vengono comunicati gli indirizzi delle istruzioni da eseguire;
    - Data-bus: dove vengono scambiati i dati all'interno del sistema,



I primi modelli di computer potevano eseguire un solo processo alla volta, mentre quelli moderni riescono a gestire più processi in contemporanea.

Per poter far questo usano un sistema di scheduling, ossia una volta eseguita l'istruzione di un processo, quest'ultimo viene momentaneamente sospeso,

permettendo così l'esecuzione dell'istruzione di un secondo processo attivo.

Per eseguire ogni istruzione, la CPU compie un ciclo perenne composto da 3 fasi:

- Fetch: la lettura della prossima istruzione;
- Decode: la decodifica dell'operazione da compiere;
- Execute: l'esecuzione dell'istruzione.

Come ultima cosa da sapere sul modello di Von Neumann è che prima di essere eseguiti, i programmi devono essere spostati nella memoria per poi poter essere eseguiti.

Nel momento in cui il programma entra nella memoria prende il nome di processo(ossia programma di esecuzione).

Ogni processo ha un effettivo ciclo di vita, poiché durante la loro esecuzione essi si evolvono raggiungendo vari stati.

1.3 L'architettura MIPS 2000

Attualmente possiamo individuare 2 tipologie di architetture di calcolatori:

- Architettura RISC: ossia l'acronimo di Reduced Istruction, le istruzioni sono di dimensione fissa(quindi non è necessario decodificarle prima del fetch).

  Non è necessario accedere alla memoria visto che gli operandi vengono effettuati dall'ALU e solo tra i registri. Ha molti registri interni e ha modi di indirizzamento semplici poiché ogni istruzione ha una dimensione fissa(dunque non si creano conflitti).

- Architettura CISC: ossia l'acronimo di Complex Istruction Set Computer, qui le istruzioni hanno dimensione variabile(dunque a differenza del RISC, è necessario decodificarle prima del fetch.

  Gli operandi vengono effettuati in memoria e necessitano molti accessi alla memoria. Ha pochi registri interni, per questo la memoria viene utilizzata anche per conservare i dati temporanei e ha modi di indirizzamento più complessi e con parziali conflitti tra le istruzioni più complesse.

Informalmente possiamo dire che l'Architettura CISC è usata per singoli scopi, essendo più complesse, mentre l'Architettura BISC, essendo più semplici è adatto a scopi più

generici.

A causa delle sue caratteristiche l'Architettura MIPS 2000, sta all'interno delle Architetture RISC.

Ed è composta da:

- Ogni word ha una dimensione fissa di 32 bit;
- Lo spazio di indirizzamento è di 2^{30} word di 32 bit, per un totale di 4 GB;
- Una memoria indicizzata al byte, dunque, dato un indirizzo di memoria t che corrisponde all'inizio di una word, per leggere la word successiva è necessario utilizzare l'indirizzo $t + 4$, poiché 4 byte corrispondono a 32 bit;
- Gli interi vengono salvati utilizzando la notazione del complemento a 2 su 32 bit;
- Dotata di 3 microprocessori:
  - La CPU principale(dotata di ALU), di 32 registri;
  - Il processore 0, non è dotato di 32 registri e non ha accesso alla memoria, ed è solo addetto alla gestione di "Trap";
  - Il processore 1, addetto ai calcoli in virgola mobile e dotato di 32 registri da 32 bit.
- I 32 registri, indicizzati da 0 a 31, della CPU principale:
  - Registro $0($0$)$: contenente un valore costante pari a 0 ed immutabile;
  - Registro $AT($1$)$: usato dalle pseudoistruzioni e dall'assemblatore;
  - Registri v0$ e v1$($2,3)$: utilizzati per gli output delle funzioni utilizzate dal programma;
  - Registri dall' a0$ a a3$($4,…,7)$: utilizzati per gli input delle funzioni;
  - Registri dal t0$ a t7$($8,…,15)$: utilizzati per i valori ricorrenti;
  - Registri k0$ e k1$($26,27)$: utilizzati dal Kernel del Sistema Operativo;
  - Registro $GP($28$)$: utilizzato per la gestione della memoria dinamica;
  - Registro $SP($29$)$: utilizzato per la gestione dello Stack delle funzioni;
  - Registro $FP($30$)$: utilizzato dalle funzioni di sistema;

- Registro $RA($31): utilizzato come puntatore di ritorno dalle funzioni.