

---

ANNO ACCADEMICO 2024/2025

---

# Architettura degli Elaboratori II

---

Teoria

Altair's Notes



**UNIVERSITÀ**  
**DI TORINO**



---

DIPARTIMENTO DI INFORMATICA

---



<b>CAPITOLO 1</b>	<b>CONCETTI DI BASE</b>	<b>PAGINA 5</b>
1.1	Introduzione	5
	Tassonomia delle architetture — 6	
1.2	Alcuni concetti fondamentali	6
1.3	Una semplice macchina RISC	7
	MIPS - Versione monociclo — 8 • Banco dei registri — 10 • Una semplice Control Unit — 11	
<b>CAPITOLO 2</b>	<b>INSTRUCTION LEVEL PARALLELISM (ILP)</b>	<b>PAGINA 13</b>
<b>CAPITOLO 3</b>	<b>CACHING</b>	<b>PAGINA 15</b>
<b>CAPITOLO 4</b>	<b>ARCHITETTURE PARALLELE</b>	<b>PAGINA 17</b>
<b>CAPITOLO 5</b>	<b>QUANTUM COMPUTING</b>	<b>PAGINA 19</b>
<b>CAPITOLO 6</b>	<b>GPU</b>	<b>PAGINA 21</b>



# Premessa

## Licenza

Questi appunti sono rilasciati sotto licenza Creative Commons Attribuzione 4.0 Internazionale (per maggiori informazioni consultare il link: <https://creativecommons.org/version4/>).



## Formato utilizzato

Box di "Concetto sbagliato":

### Concetto sbagliato 0.1: Testo del concetto sbagliato

Testo contenente il concetto giusto.

Box di "Corollario":

### Corollario 0.0.1 Nome del corollario

Testo del corollario. Per corollario si intende una definizione minore, legata a un'altra definizione.

Box di "Definizione":

### Definizione 0.0.1: Nome delle definizioni

Testo della definizione.

Box di "Domanda":

### Domanda 0.1

Testo della domanda. Le domande sono spesso utilizzate per far riflettere sulle definizioni o sui concetti.

Box di "Esempio":

### Esempio 0.0.1 (Nome dell'esempio)

Testo dell'esempio. Gli esempi sono tratti dalle slides del corso.

**Box di "Note":**

**Note:-**

Testo della nota. Le note sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive.

**Box di "Osservazioni":**

**Osservazioni 0.0.1**

Testo delle osservazioni. Le osservazioni sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive. A differenza delle note le osservazioni sono più specifiche.



# Concetti di Base



### 1.1.1 Tassonomia delle architetture

Il contenuto del corso può essere descritto dalla "Tassonomia di Flynn".

#### Definizione 1.1.1: Tassonomia di Flynn

La Tassonomia di Flynn organizza i vari tipi di processori in base a determinate caratteristiche che verranno approfondite in questo corso.

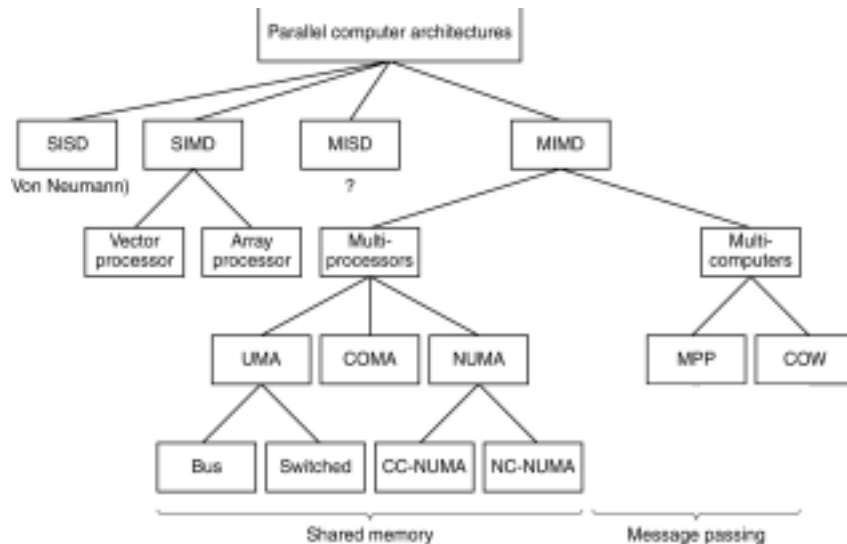


Figure 1.2: La Tassonomia di Flynn

## 1.2 Alcuni concetti fondamentali

#### Definizione 1.2.1: Microarchitettura

L'architettura interna di un processore: com'è fatto a partire dal suo datapath.

#### Corollario 1.2.1 Datapath

Il percorso che compiono le istruzioni all'interno del processore per venire eseguite.

#### Note:-

Diversi tipi di istruzioni percorrono diverse parti del datapath per venire eseguite.

#### Definizione 1.2.2: ISA

L'Instruction Set Architecture (ISA) è l'insieme di istruzioni macchina di un processore.

#### Note:-

Due processori possono avere lo stesso ISA, ma microarchitetture diverse (e.g. AMD e Intel).

## 1.3 Una semplice macchina RISC

### Domanda 1.1

Qual è la differenza tra un processore a 32 bit e un processore a 64 bit?

**Risposta:** il processore a 64 bit manipola in maniera naturale informazione scritta con 64 bit e il processore a 32 bit manipola in maniera naturale informazione scritta con 32 bit.

### Caratteristiche fondamentali dell'architettura RISC:

- ⇒ le istruzioni hanno tutte la stessa lunghezza (o a 32 bit o a 64 bit);
- ⇒ le istruzioni sono semplici;
- ⇒ la Control Unit è semplice (poche porte logiche, quindi frequenze di clock più elevate).

#### Note:-

Ciò che verrà descritto in questa sezione è una versione semplificata di MIPS, la prima macchina RISC. Si considerano 32 registri a 32 bit e si ignorano le operazioni floating point.

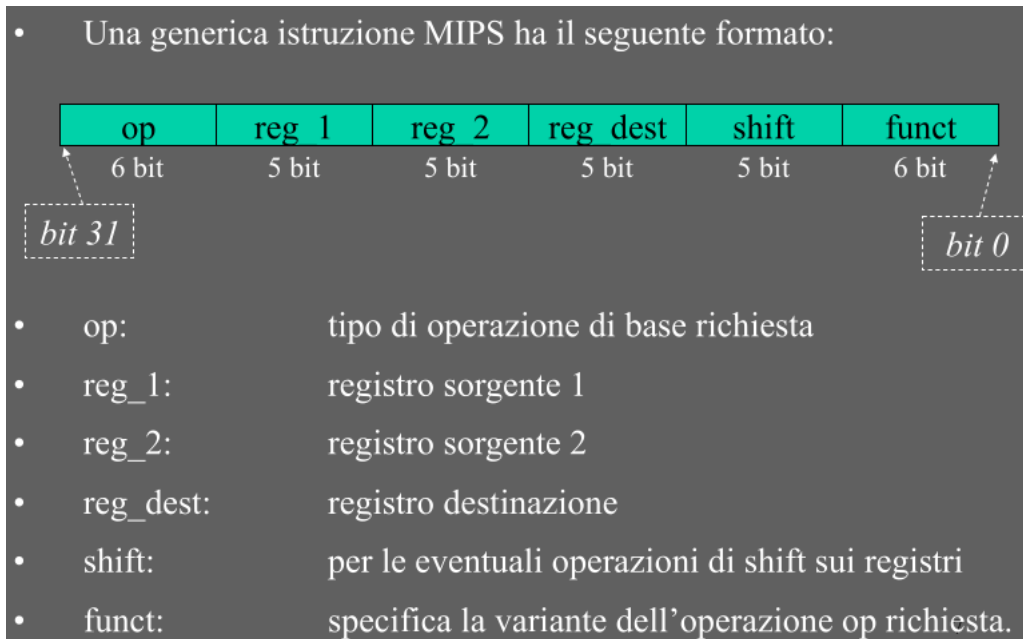


Figure 1.3: Istruzione MIPS

### Definizione 1.3.1: Istruzioni di tipo-R

Le istruzioni di tipo-R usano due registri e restituiscono il risultato a un terzo registro. La convenzione prevede che il campo OP sia 0. L'operazione specifica si trova nel campo funct.

#### Note:-

Solitamente si usa la lettera D quando si parla di dati interi (DADD, DSUB, etc.), F per i floating point.

### Definizione 1.3.2: Istruzioni di tipo-I

Le istruzioni di tipo-I usano un valore immediato. La convenzione prevede che il campo op sia 8.

### Definizione 1.3.3: LOAD e STORE

La LOAD carica in un registro un valore che si trova in memoria (op = 35). La STORE salva in memoria il valore di un registro (op = 43).

### Definizione 1.3.4: Salti condizionati (BRANCH)

Salta solo se si verifica una determinata condizione (op = 5).

### Definizione 1.3.5: Salti incondizionati (JUMP)

Salta sempre (op = 4).

## 1.3.1 MIPS - Versione monociclo

Generalmente i primi due passi di ogni istruzione sono:

1. usa il Program Counter (PC) per prelevare dalla "memoria di istruzioni"<sup>1</sup> la prossima istruzione da eseguire;
2. Decodifica l'istruzione e contemporaneamente legge i registri.

### Note:-

I passi successivi dipendono dal tipo di istruzione (tutte usano la ALU).

⇒ LOAD e STORE accedono alla memoria dati e nel caso di LOAD viene aggiornato un registro;

⇒ le istruzioni logico-aritmetiche aggiornano un registro;

⇒ le istruzioni di salto possono alterare il valore di PC.

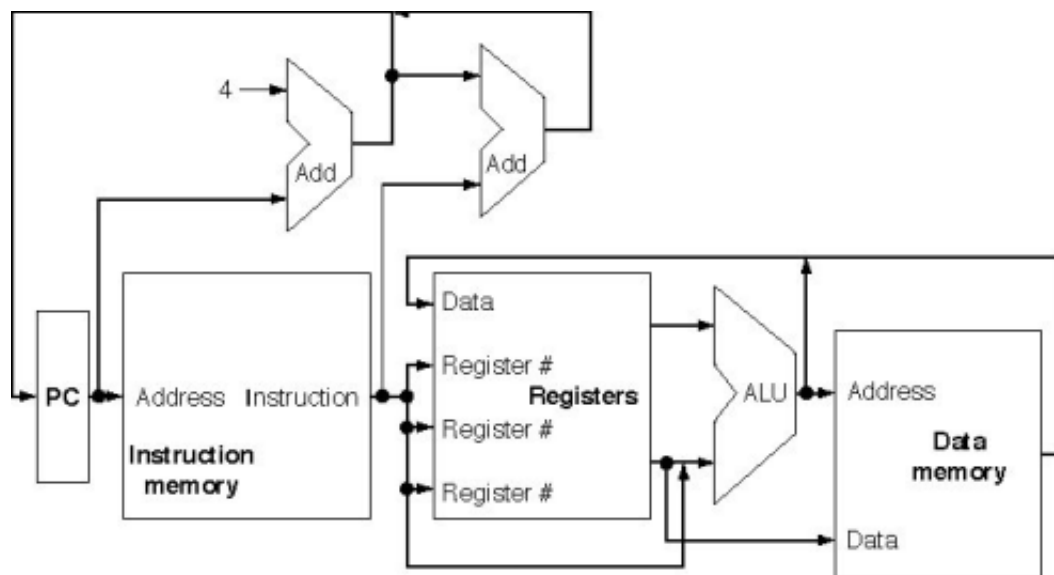


Figure 1.4: Schema ad alto livello del datapath MIPS

### Note:-

Il fluire delle informazioni nel datapath deve essere controllato da una "Control Unit".

<sup>1</sup>Cache di primo livello.

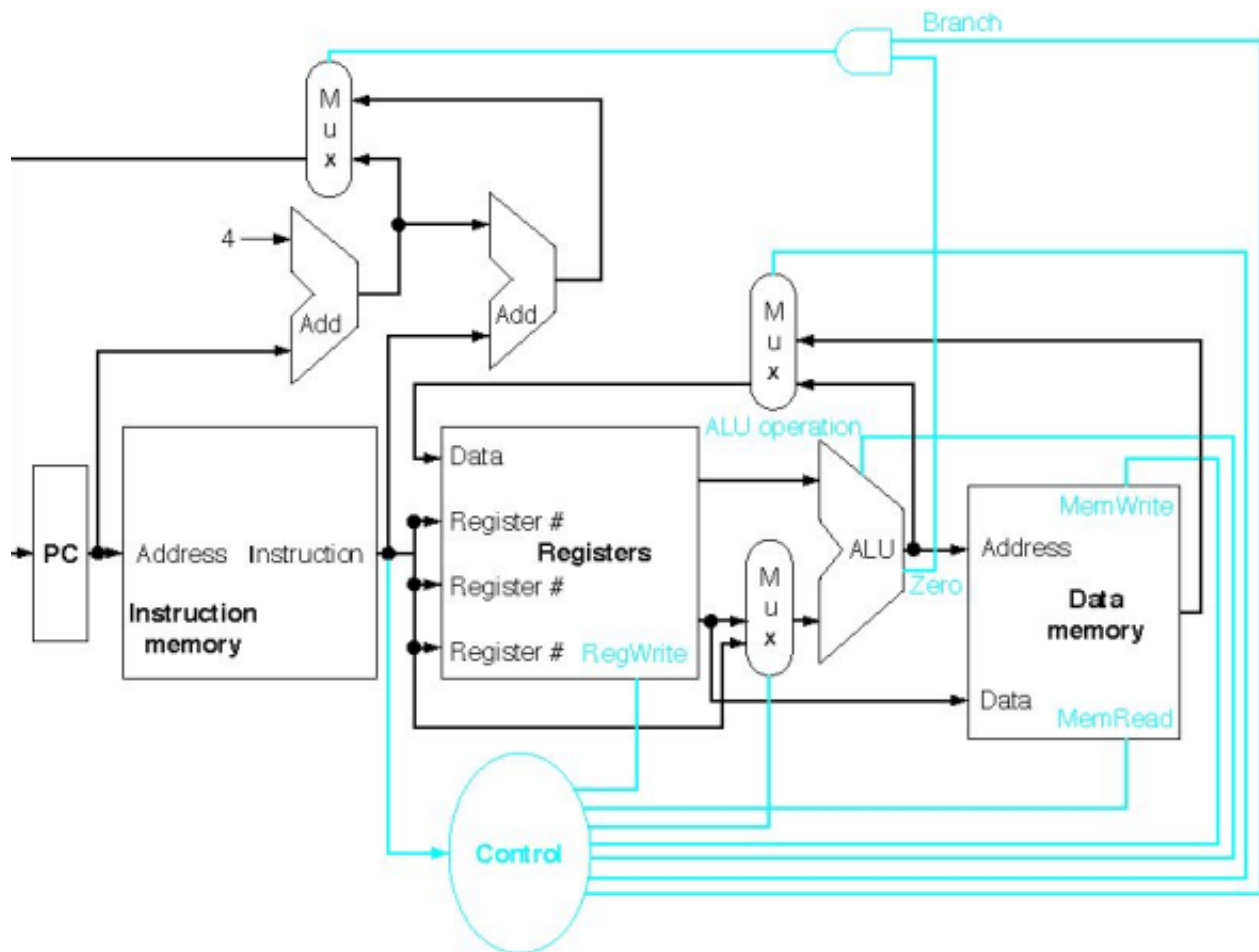


Figure 1.5: Versione modificata del MIPS con una Control Unit

**Note:-**

L'esecuzione di ciascuna istruzione può avvenire in un unico ciclo di clock, purché sia stato adeguatamente dimensionato.

Per capire come funziona il datapath di una macchina monociclo si osserva che esso è composto da due tipi di elementi logici:

- ⇒ gli elementi di *stato*;
- ⇒ gli elementi di tipo *combinatorio*.

**Definizione 1.3.6: Elementi di stato**

Gli elementi di stato sono quelli in grado di memorizzare uno *stato* (e.g. flip flop, registri e memorie). Un elemento di stato possiede almeno 2 ingressi e un'uscita. Gli ingressi richiedono:

- ⇒ il valore da scrivere nell'elemento;
- ⇒ il clock per determinare quando scriverlo.

Il dato presente in uscita è sempre quello scritto in un ciclo di clock precedente.

**Note:-**

Solitamente esiste un terzo ingresso "di controllo" che stabilisce se l'elemento di stato può effettivamente memorizzare l'input.

**Definizione 1.3.7: Elementi combinatori**

Gli elementi combinatori sono quelli in cui le uscite dipendono solamente dai valori d'ingresso in un dato istante (e.g. ALU e Multiplexer).

### 1.3.2 Banco dei registri

**Definizione 1.3.8: Banco dei registri**

Nelle immagini precedenti i registri della CPU sono rappresentati da un'unità funzionale detta *register file* (o banco dei registri). Essa è un'unità di memoria molto piccola e veloce.

**Note:-**

Si può accedere a ognuno dei 32 registri (da 0 a 31) specificando il suo indirizzo. Ogni registro può essere letto o scritto.

**Operazione di scrittura:** quando il segnale di controllo (RegWrite) è a 1, il valore proveniente dalla ALU o dalla Data Memory e presente in input in *DST data* viene memorizzato nel registro di destinazione specificato da *DST addr*.

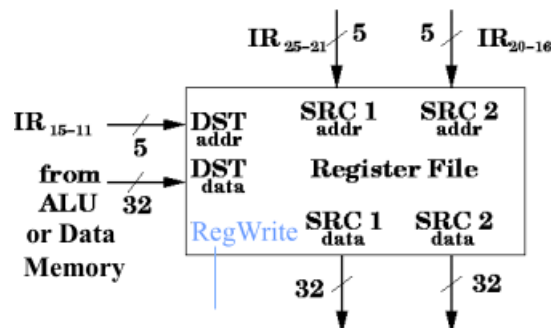


Figure 1.6: Operazione di scrittura

**Operazione di lettura:** le letture sono immediate. In qualunque momento alle uscite *SRC1 data* e *SRC2 data* è presente il contenuto dei registri i cui numeri sono specificati da *SRC1 addr* e *SRC2 addr*.

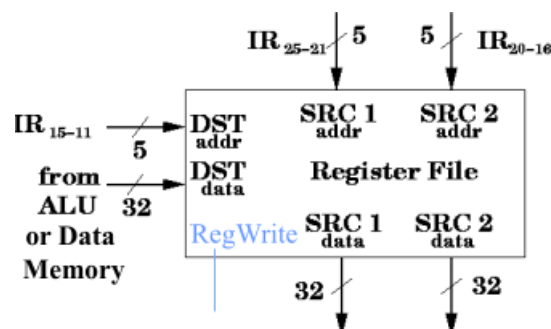


Figure 1.7: Operazione di lettura

### 1.3.3 Una semplice Control Unit

**Definizione 1.3.9: Control Unit**

La Control Unit riceve in input i 6 bit del campo op dell'istruzione e deve generare in output i segnali per comandare:

- ⇒ la scrittura dei registri;
- ⇒ la lettura/scrittura della memoria dati (MemRead/MemWrite);
- ⇒ i Multiplexer che selezionano gli input da usare;
- ⇒ la ALU (ALUOp) che deve eseguire ciascuna operazione aritmetico-logica appropriata per la specifica istruzione in esecuzione.

**Corollario 1.3.1 Segnale ALUOp**

Il segnale ALUOp dipende:

- ⇒ dal tipo di istruzione in esecuzione, specificato nel campo op;
- ⇒ dalla specifica operazione da eseguire, determinata dal campo funct.



# 2

## Instruction Level Parallelism (ILP)





# 3

## Caching



# 4

## Architetture Parallele



# 5

## Quantum Computing



# 6

GPU



