



**POLITECNICO**  
MILANO 1863

# **Architettura dei calcolatori e sistemi operativi**

## **Architettura MIPS e set istruzioni** **Capitolo 2 P&H**

# Instruction Set Architecture – ISA

Linguaggio assembler e linguaggio macchina

ISA processore MIPS

- Modello di memoria
- Registri
- Istruzioni macchina e tipi di formati
- Modalità di indirizzamento



# Instruction Set Architecture - ISA

E' la descrizione del calcolatore riferita al suo linguaggio macchina, cioè all'insieme delle istruzioni (*instruction set*) che possono essere interpretate direttamente dall'architettura del processore

programma in linguaggio macchina = rappresentazione del programma per essere eseguito su un processore

Ogni architettura di processore ha il suo linguaggio macchina

- Architettura definita dall'insieme delle istruzioni

## **ISA (Instruction Set Architecture)**

- Due processori con lo stesso linguaggio macchina hanno la stessa architettura anche se le implementazioni hardware possono essere diverse



# Instruction Set Architecture – ISA (cont.)

E' necessario definire

- elementi a disposizione delle istruzioni macchina: modello della memoria e registri
- insieme delle istruzioni macchina
  - formato e dimensione
- riferimenti agli operandi
  - in memoria e/o nei registri del processore
  - tipi di indirizzamento
- tipi di dati e dimensioni
- modalità operative



# Linguaggio assembler e linguaggio macchina

Il linguaggio assembler è il **linguaggio simbolico** che consente di programmare un calcolatore utilizzando le istruzioni del linguaggio macchina

- le istruzioni del linguaggio assembler sono in corrispondenza (quasi) uno a uno con quelle linguaggio macchina
- la notazione simbolica consente di rappresentare istruzioni, registri, dati e riferimenti alla memoria

La questione delle *pseudo-istruzioni*

Useremo il linguaggio simbolico mettendo in evidenza le differenze tra linguaggio macchina e linguaggio assembler



# Linguaggio assembler

Per poter essere eseguito un programma scritto in assembler deve essere **tradotto in linguaggio macchina** in modo da tradurre i codici mnemonici delle istruzioni in codici operativi, sostituire tutti i riferimenti simbolici degli indirizzi con la loro forma binaria e riservare spazio di memoria per le variabili

❑ l'operazione di traduzione viene eseguita dall'**ASSEMBLATORE**

❑ se nel codice sorgente sono presenti

- riferimenti simbolici definiti in moduli (file) esterni a quello assemblato
- riferimenti simbolici che dipendono dalla rilocazione del modulo

è necessario anche il **LINKER**



# Linguaggio assembler: traduzione

**Programma  
in linguaggio  
assembler (MIPS)**

```
add $2, $4, $2  
add $3, $3, $2  
lw $15, 4($2)  
.....
```

**Assemblatore  
e Linker**

**Programma  
in linguaggio  
macchina**

```
00000011100010101010 ...  
00000011010100011101 ...  
01001000001000000011 ...  
001000100010000 ..... 
```



# Linguaggio C

Programma in  
linguaggio ad alto  
livello (C)

```
a = a + c  
b = b + a
```

**Compilatore**

Programma in  
linguaggio assembler  
(MIPS)

assembler come linguaggio  
target della fase di compilazione

```
add $2, $4, $2  
add $3, $3, $2  
lw . . .
```

**Assemblatore  
e Linker**

Programma  
in linguaggio  
macchina

```
011100010101010 ..  
000110101000111 ..  
000010000010000  
001000100010000
```





# Istruzioni e variabili in linguaggio macchina

**istruzioni** - costituite da

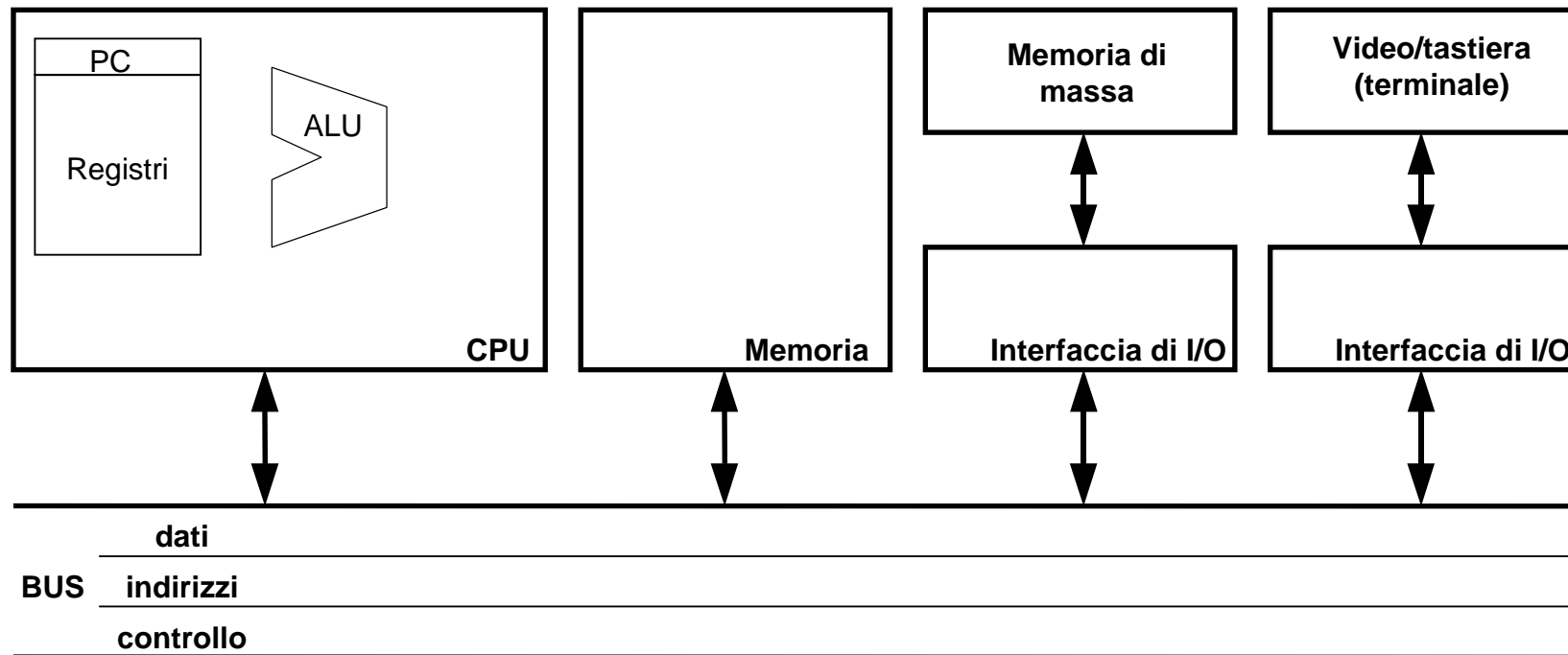
- *codice operativo* (opcode) che identifica in modo univoco l'istruzione e definisce l'operazione associata
- *riferimento/i all'operando/i* su cui agisce l'istruzione. Il riferimento può essere
  - *implicito* nel codice operativo
  - direttamente il *valore numerico* dell'operando
  - un *registro* del processore
  - in modo diretto o indiretto una *locazione di memoria*

**variabili** - accessibili dal processore

- *riferimento* rappresentato da un indirizzo di memoria (o di registro)
- *valore* contenuto nella parola di memoria associata all'indirizzo e rappresentato tramite codifica binaria opportuna



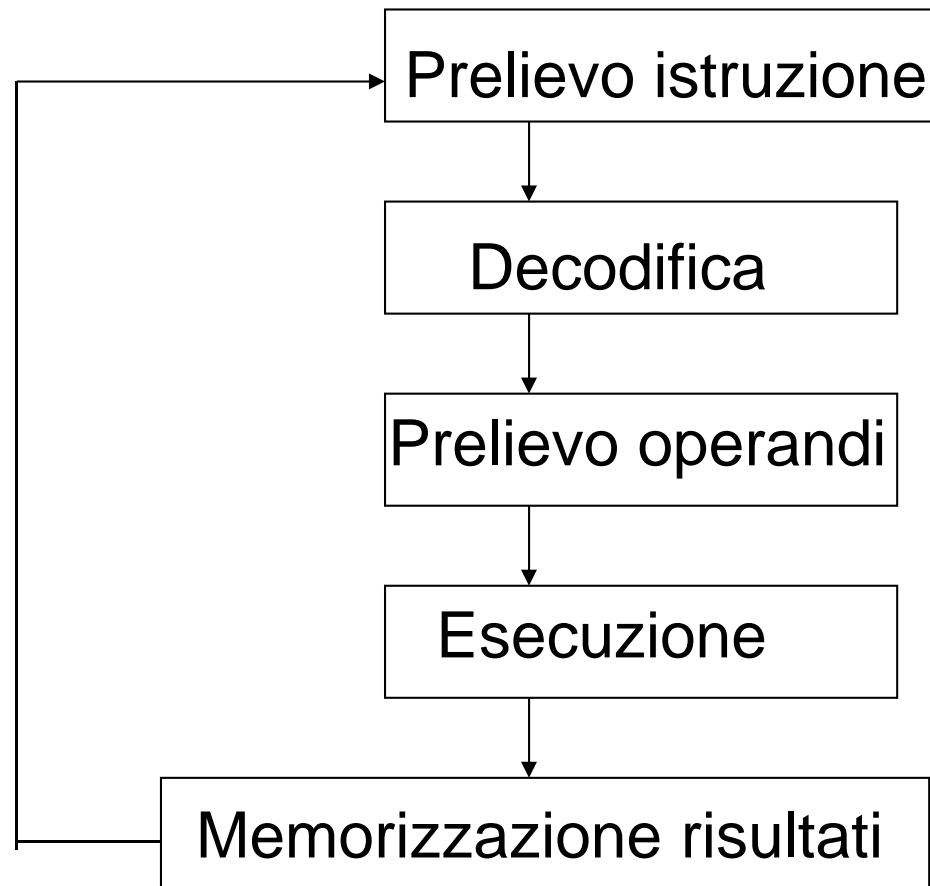
# Architettura di riferimento dei calcolatori



Architettura di Von Neumann



## Esecuzione di una istruzione in linguaggio macchina



# Architettura del processore MIPS

Linguaggio assembler dell'architettura MIPS

Architettura MIPS appartiene alla famiglia delle architetture **RISC (Reduced Instruction Set Computer)** sviluppate dal 1980 in poi

- Esempi: Sun Sparc, HP PA-RISC, IBM Power PC, DEC Alpha, ARM

Principali obiettivi delle architetture RISC:

- Semplificare la progettazione dell'hardware e del compilatore
- Massimizzare le prestazioni
- Minimizzare i costi



# Processore di tipo RISC

## CPU RISC (Reduced Instruction Set Computer)

ispirata al principio di eseguire soltanto istruzioni semplici: le operazioni complesse vengono scomposte in una serie di istruzioni più semplici da eseguire in un ciclo base ridotto, con l'obiettivo di migliorare le prestazioni ottenibili dalle *CPU CISC*

- *CPU* caratterizzata da istruzioni molto semplificate
- *CPU* relativamente semplice: si riducono i tempi di esecuzione delle singole istruzioni, che sono però meno potenti delle istruzioni *CISC*
  - ➔ massimizzazione della frequenza di completamento dell'esecuzione delle istruzioni
- Dimensione **fissa** delle istruzioni: più semplice la gestione della fase di prelievo (fetch) e della decodifica delle istruzioni da eseguire
- Gli operandi dell'*ALU* possono provenire dai registri ma *non* dalla memoria. Per il trasferimento dei dati da memoria ai registri e viceversa si utilizzano delle apposite operazioni di caricamento (*load*) e di memorizzazione (*store*):  
*architetture load/store*



# Memoria – struttura e indirizzamento

## Architettura MIPS

- parole da 32 bit (4 byte), memoria indirizzabile a byte
- spazio di indirizzamento a byte 4 Gbyte (32 bit di indirizzo)
- spazio di indirizzamento a parole 1 Gparola (30 bit di indirizzo)
  - **vincolo di allineamento**: gli indirizzi di parola sono multipli di 4
- Enumerazione dei byte nella parola (indirizzamento per byte)
  - **big-endian**: l'indirizzo del byte più significativo specifica l'indirizzo di parola (byte numerati da sinistra a destra, ad es CPU Sparc, Motorola, **MIPS** ..), quindi il byte meno significativo si trova all'indirizzo più alto
  - **little-endian**: l'indirizzo del byte meno significativo specifica l'indirizzo di parola (byte numerati da destra a sinistra, ad es. Intel), quindi il byte meno significativo si trova all'indirizzo più basso
  - La modalità di enumerazione dei byte nella parola è significativo nel caso di accesso alla parola byte a byte



# Memoria: indirizzamento a byte e allineamento

	Indirizzo di byte			
Parola 0	0	1	2	3
Parola 4	4	5	6	7
Parola 8	8	9	10	11
	MS byte			LS byte
...				
Parola $2^k-4$	$2^k-4$	$2^k-3$	$2^k-2$	$2^k-1$

**big-endian** alla parola viene assegnato lo stesso indirizzo del suo byte più significativo

	Indirizzo di byte			
Parola 0	3	2	1	0
Parola 4	7	6	5	4
Parola 8	11	10	9	8
	MS byte			LS byte
...				
Parola $2^k-4$	$2^k-1$	$2^k-2$	$2^k-3$	$2^k-4$

**little-endian** alla parola viene assegnato lo stesso indirizzo del suo byte meno significativo



# Registri

Sono elementi di memoria interni alla CPU: a livello ISA interessano tutti e soli i **registri referenziabili** nelle istruzioni macchina

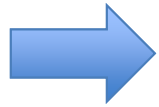
## Architettura MIPS

- 32 registri da 32 bit, accessibili anche a byte, organizzati in un banco di registri (*Register File*)
  - referenziabili con nome simbolico (preceduto da \$) nelle istruzioni assembler e associati univocamente a un «numero» tra 0 e 31
  - identificabili (indirizzabili) con 5 bit in linguaggio macchina
  - possono avere la loro specificità in termini di utilizzo, ma sono trattati in modo omogeneo
  
- 3 registri da 32 bit non referenziabili in ISA e di uso specifico





## Registri referenziabili



Nome	Numero	Utilizzo
\$0	0	Costante 0
\$at	1	Riservato all'assemblatore
\$v0-\$v1	2-3	Valori restituiti da una funzione e risultati calcolo espressioni
\$a0-\$a3	4-7	Argomenti
\$t0-\$t7	8-15	Variabili temporanee
\$s0-\$s7	16-23	Variabili da preservare
\$t8-\$t9	24-25	Altre variabili temporanee
\$k0-\$k1	26-27	Riservati al kernel del sistema operativo
\$gp	28	Global Pointer (puntatore area dati globali/statici)
\$sp	29	Stack Pointer (puntatore allo stack – prima piena)
\$fp	30	Frame Pointer (puntatore frame funzione)
\$ra	31	Return Address (utilizzato nelle chiamate a funzione)

## Non referenziabili

Registri non referenziabili	
pc	Program Counter
hi	Registro per moltiplicazioni e divisioni
lo	Registro per moltiplicazioni e divisioni



# Insieme delle istruzioni

Classi di istruzioni tipiche in linguaggio macchina:

- aritmetico-logiche
- trasferimento da e in memoria (e tra registri)
- modifica del flusso di esecuzione:
  - salto condizionato, incondizionato
  - salto a sottoprogramma, ritorno da sottoprogramma
- trasferimento dati da e in periferica (istruzioni di I/O)
- istruzioni speciali (controllo)



# Formato delle istruzioni MIPS

Tutte le **istruzioni macchina** MIPS hanno la **stessa** dimensione (**32 bit**) e quindi occupano una sola parola di memoria

I 32 bit hanno un significato diverso a seconda del formato (o tipo) di istruzione

- il tipo di istruzione è riconosciuto in base al **codice operativo** dell'istruzione (**OPCODE**): **6 bit** più significativi dell'istruzione MIPS
  - OPCODE definisce non solo «che cosa fa» l'istruzione ma anche come «utilizzare» i bit rimanenti

Le istruzioni MIPS sono di **3** formati (tipi):

## **Tipo R (register)**

- Istruzioni aritmetico-logiche

## **Tipo I (immediate)**

- Istruzioni di accesso alla memoria o contenenti valori costanti

## **Tipo J (jump)**

- Istruzioni di salto (incondizionato)



# Modalità di indirizzamento in MIPS

Le modalità di indirizzamento previste in linguaggio macchina MIPS sono:

- Immediato
- A registro
- Con base e spiazzamento
- Relativo al Program Counter
- Pseudo-diretto (rispetto al Program Counter)

La **modalità di indirizzamento** è associata in modo univoco al **formato dell'istruzione** (cioè a come vengono interpretati i bit nell'istruzione in linguaggio macchina)

- **Tipo R (register):** a registro
- **Tipo I (immediate):** immediato, con base e spiazzamento, relativo al Program Counter
- **Tipo J (jump):** psuedo-diretto

Una singola istruzione «logica» può usare più di una modalità di indirizzamento, ad es. `add` e `addi` (che sono di due formati diversi, R e I rispettivamente e hanno anche due codici operativi diversi)



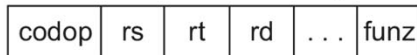
# Modalità di indirizzamento in MIPS (2)

## 1. Indirizzamento immediato



Operando  
nell'istruzione

## 2. Indirizzamento tramite registro

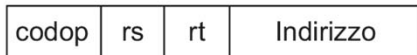


Registri

Registro

Operando in  
registro

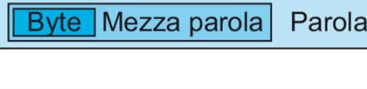
## 3. Indirizzamento tramite base



Memoria

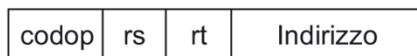
Registro

+



Operando in  
memoria

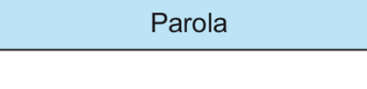
## 4. Indirizzamento relativo al PC



Memoria

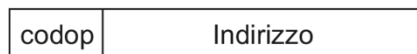
PC

+



Operando in  
memoria

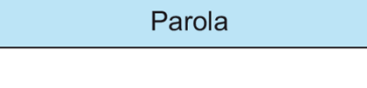
## 5. Indirizzamento pseudodiretto



Memoria

PC

:



Operando in  
memoria

