



# CPU a singolo ciclo

Prof. Alberto Borghese  
Dipartimento di Informatica  
[alberto.borghese@unimi.it](mailto:alberto.borghese@unimi.it)

Università degli Studi di Milano

Riferimento sul Patterson: capitolo 4.2 , 4.4, D1, D2.



## Sommario

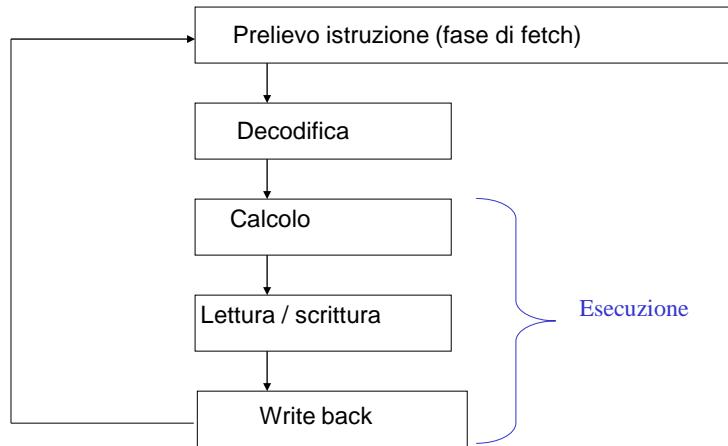
Introduzione alla CPU

CPU per le istruzioni di tipo R

CPU per le istruzioni di tipo I



# Ciclo di esecuzione di un'istruzione MIPS



# Obiettivo



Costruzione di una CPU completa che sia in grado di eseguire:

- Istruzioni logico-matematiche di tipo R (e.g. add, sub, and....). e.g. add \$t0, \$t1, \$t2.
- Istruzioni logico-matematiche di tipo I (e.g. addi, ori...) e.g. addi \$t0, \$t1, 24.
- Accesso alla memoria in lettura (lw) o scrittura (sw). e.g. lw \$t0, 24(\$t1)
- Istruzioni di salto condizionato (branch). e.g. beq \$t0, \$t1, etichetta
- Istruzioni di salto incondizionato (jump). e.g. j etichetta



## Codifica delle istruzioni



- Tutte le istruzioni MIPS hanno la stessa dimensione (**32 bit**) – **Architettura RISC**.
- I 32 bit hanno un significato diverso a seconda del formato (o tipo) di istruzione
  - il tipo di istruzione è riconosciuto in base al valore di alcuni bit (**6 bit**) più significativi (**codice operativo - OPCODE**)
- Le istruzioni MIPS sono di **3** tipi (formati):
  - **Tipo R (register)** – **Lavorano prevalentemente su 3 registri.**
    - Istruzioni aritmetico-logiche.
  - **Tipo I (immediate)** – **Lavorano su 2 registri. L'istruzione è suddivisa in un gruppo di 16 bit contenenti informazioni + 16 bit riservati ad una costante.**
    - Istruzioni di accesso alla memoria o operazioni con una costante.
  - **Tipo J (jump)** – **Lavora senza registri: codice operativo + indirizzo di salto.**
    - Istruzioni di salto incondizionato.

	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
R	op	rs	rt	rd	shamt	funct
I	op	rs	rt			Indirizzo / costante
J	op				Indirizzo / costante	



## I componenti di un'architettura



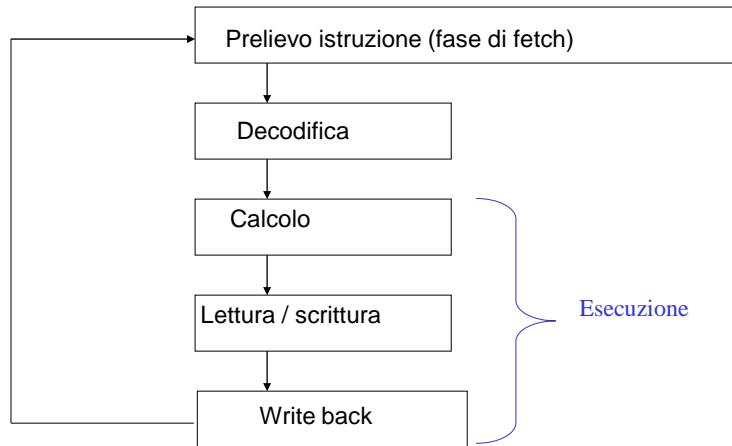
### CPU

- Banco di registri (*Register File*) ad accesso rapido, in cui memorizzare i dati di utilizzo più frequente. Il tempo di accesso ai registri è circa 10 volte più veloce del tempo di accesso alla memoria principale.
- Registro *Program counter (PC)*. Contiene l'indirizzo dell'istruzione corrente da aggiornare durante l'evoluzione del programma, in modo da prelevare dalla memoria la corretta sequenza di istruzioni;
- Registro *Instruction Register (IR)*. Contiene l'istruzione in corso di esecuzione. Questo registro verrà utilizzato più avanti nelle architetture dotate di pipe-line.
- Unità per l'esecuzione delle operazioni aritmetico-logiche (*Arithmetic Logic Unit - ALU*). I dati forniti all'*ALU* possono provenire da registri oppure direttamente dalla memoria, a seconda delle modalità di indirizzamento previste;
- Unità aggiuntive per elaborazioni particolari come unità aritmetiche per dati in virgola mobile (*Floating Point Unit - FPU*), sommatori ausiliari, ecc.;
- *Unità di controllo*. Controlla il flusso e determina le operazioni di ciascun blocco.

### MEMORIA PRINCIPALE



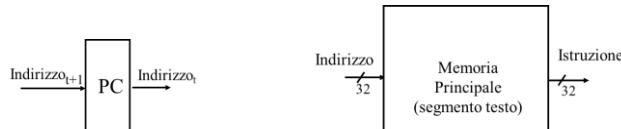
# Ciclo di esecuzione di un'istruzione MIPS



## Lettura dell'istruzione (fetch)



- Istruzioni e dati risiedono nella memoria principale, dove sono stati caricati attraverso un'unità di ingresso.
- L'esecuzione di un programma inizia quando il registro PC punta alla (contiene l'indirizzo della) prima istruzione del programma in memoria (segmento testo).
- Il segnale di controllo per la lettura (READ) viene inviato alla memoria.
- Trascorso il tempo necessario alla lettura dalla memoria, la parola indirizzata (in questo caso la prima istruzione del programma) si troverà nel registro IR.
- Il contenuto del PC viene incrementato in modo da puntare all'istruzione successiva.



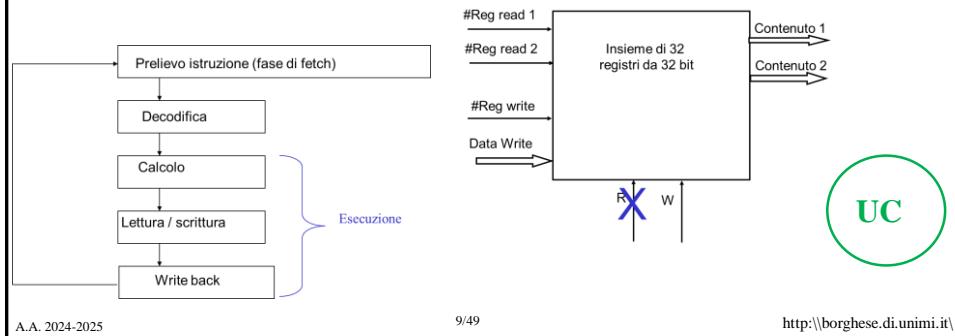


# Decodifica dell'istruzione



- L'istruzione contenuta nel registro IR viene decodificata per essere eseguita. Alla fase di decodifica corrisponde la predisposizione della CPU (apertura delle vie di comunicazione appropriate) all'esecuzione dell'istruzione.
- In questa fase vengono anche recuperati gli operandi. Nelle architetture MIPS gli operandi possono essere solamente nel Register File oppure letti dalla memoria.

Architetture LOAD/STORE: Le istruzioni di caricamento dalla memoria sono separate da quelle aritmetico/logiche.

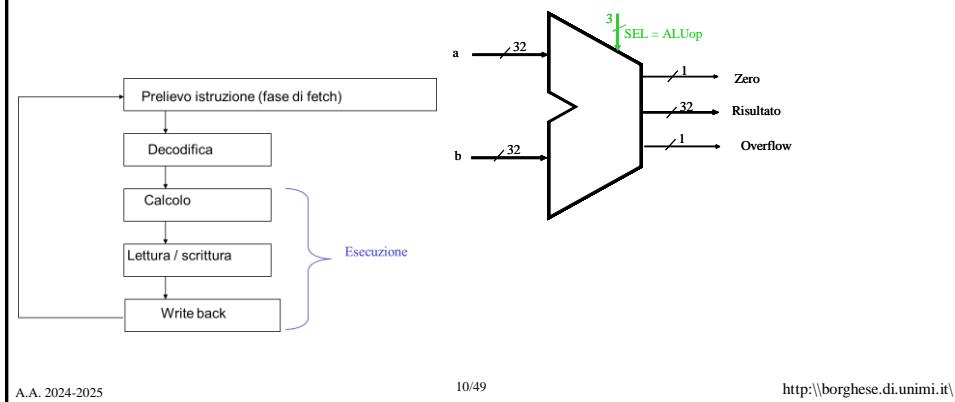


# Calcolo dell'istruzione (execute - calcolo)



Viene selezionata all'interno della ALU l'operazione prevista dall'istruzione e determinata in fase di decodifica dell'istruzione.

Tra le operazioni previste, c'è anche la formazione dell'indirizzo di memoria da cui leggere o su cui scrivere un dato.





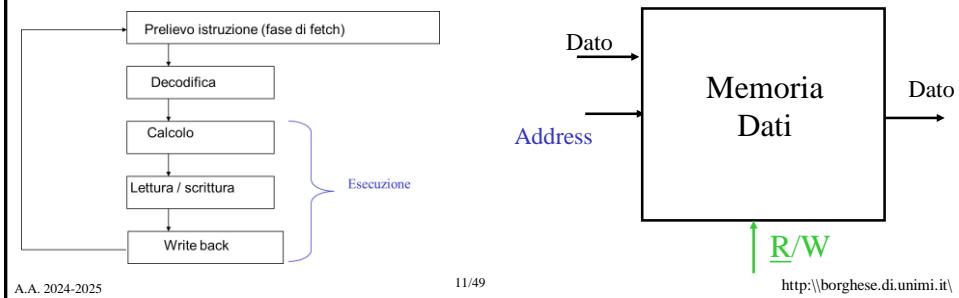
### **Lettura / Scrittura in memoria dati**



In questa fase il dato presente in un registro, viene scritto in memoria oppure viene letto dalla memoria un dato e trasferito ad un registro.

Questa fase non è richiesta da tutte le istruzioni

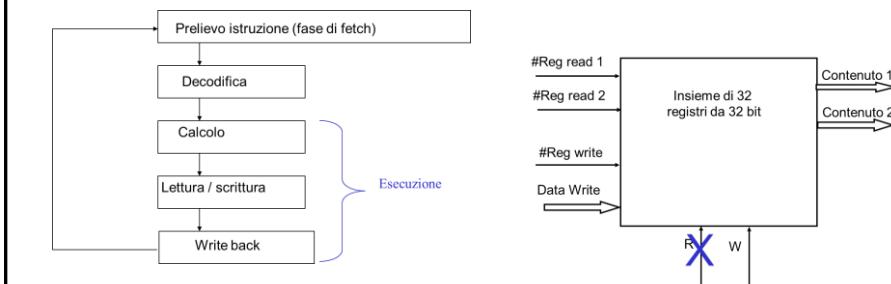
Nel caso particolare di Architetture LOAD/STORE, quali MIPS, le istruzioni di caricamento dalla memoria sono separate da quelle aritmetico/logiche. Se effettuo una Lettura / Scrittura, **non** eseguo operazioni aritmetico logiche sui dati.



## Scrittura in register file (write-back)



- Il risultato dell'operazione può essere memorizzato nei registri ad uso generale oppure in memoria.
  - Non appena è terminato il ciclo di esecuzione dell'istruzione corrente (termina la fase di Write Back), si preleva l'istruzione successiva dalla memoria.





# Come funziona una CPU?

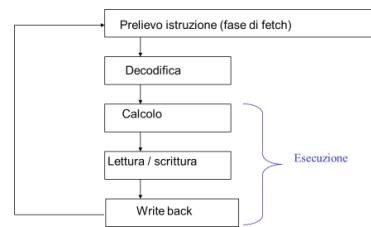


- Usa un registro, il Program Counter (PC) per ottenere l'indirizzo dell'istruzione.
- Preleva l'istruzione dalla memoria e la inserisce nell'IR.

- L'UC capisce di che tipo di istruzione si tratta (decodifica).
  - usa l'istruzione stessa per decidere cosa fare esattamente.
- Legge il contenuto dei registri.

Da qui le istruzioni si differenziano.

- Calcolo: utilizzo dell'ALU dopo aver letto i registri:
  - per calcolare l'indirizzo in memoria.
  - per eseguire un'operazione logico-aritmetica.
  - per effettuare test (uguaglianza, disuguaglianza, <...).



- Accesso alla memoria (se richiesto).
- Scrittura del risultato nel register file (se richiesto).



# Sommario



Introduzione alla CPU

CPU per le istruzioni di tipo R

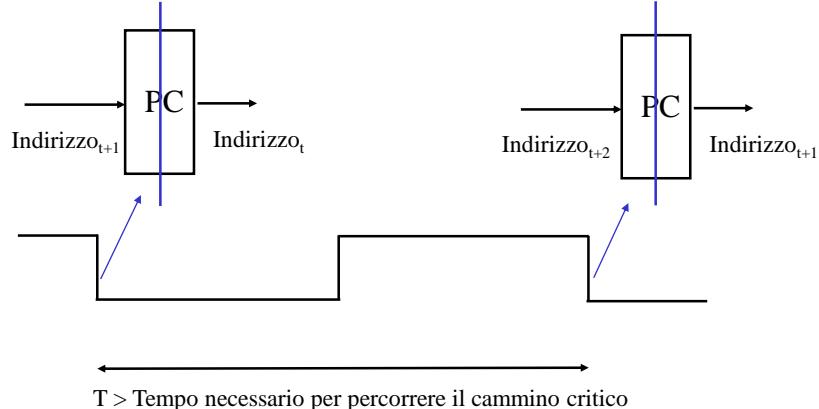
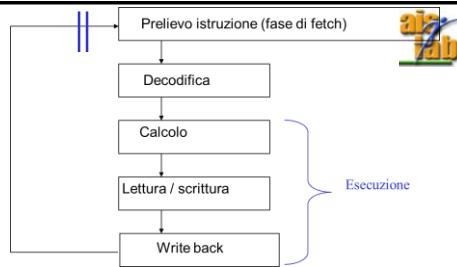
CPU per le istruzioni di tipo I



# Temporizzazione

1 istruzione per ciclo di clock.

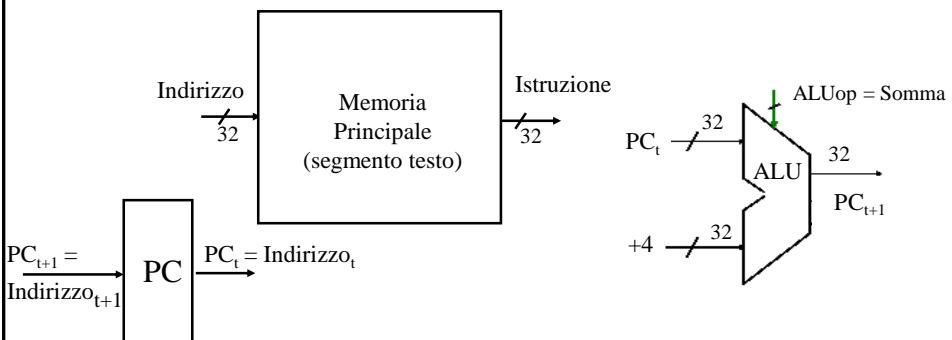
Temporizzazione del PC.



# Fase di fetch



- 1) Memorizzare l'indirizzo dell'istruzione nel PC.
- 2) Leggere l'istruzione dalla memoria.
- 3) Aggiornare l'indirizzo in modo che in PC sia contenuto l'indirizzo dell'istruzione successiva.



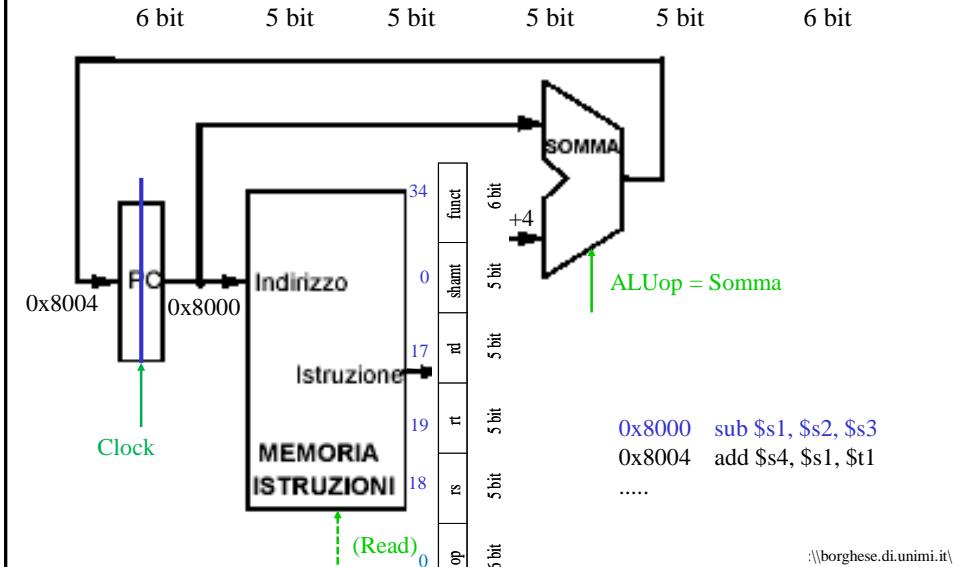


## Circuito della fase di fetch



R

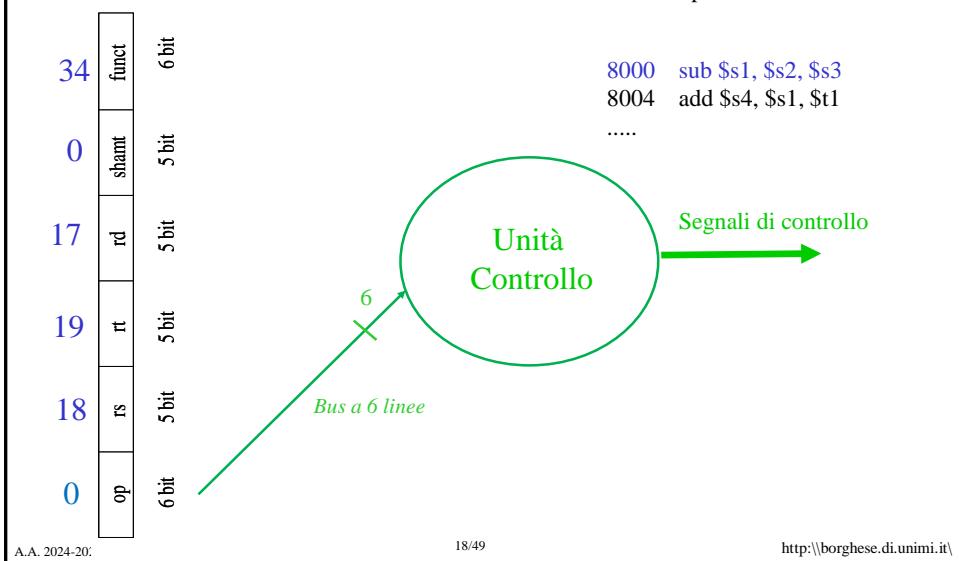
op = 0	rs = 18	rt = 19	rd = 17	Shamt=0	funct=34
--------	---------	---------	---------	---------	----------



## Fase di decodifica



- Leggo l'istruzione e genero i segnali di controllo opportuni.  
Bus che connette i 6 bit del CodOp con la UC

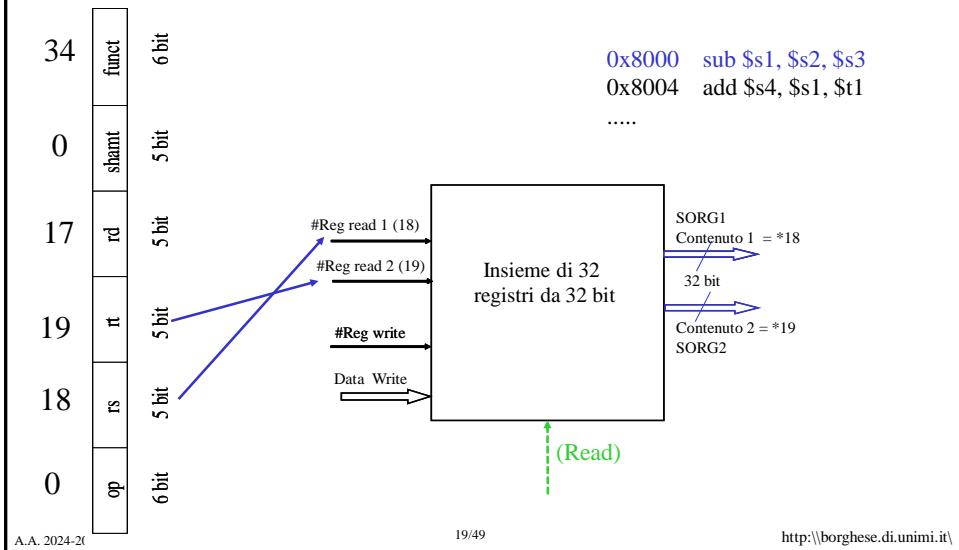




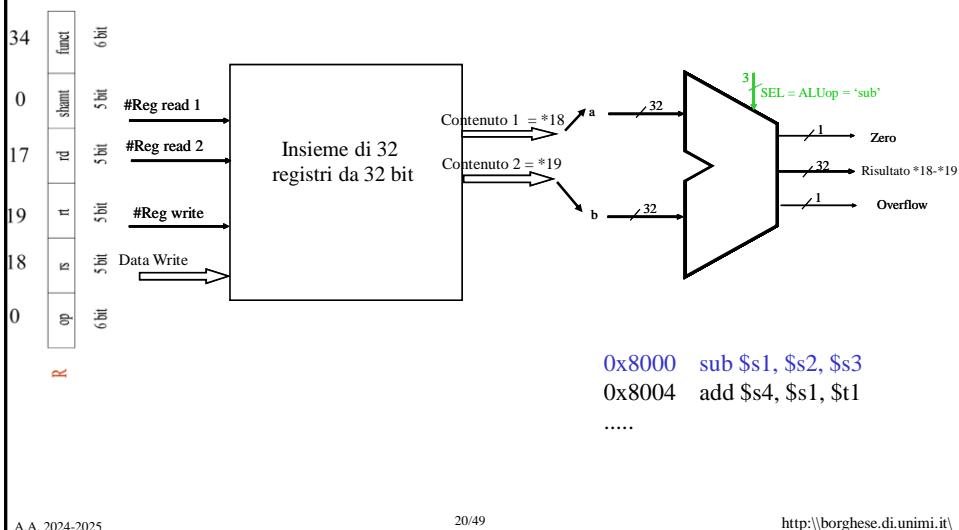
## Lettura dei registri (istruzioni di tipo R)



2) Leggo il contenuto dei registri. 2 bus che collegato i campi rs e rt con gli input corrispondenti del register file

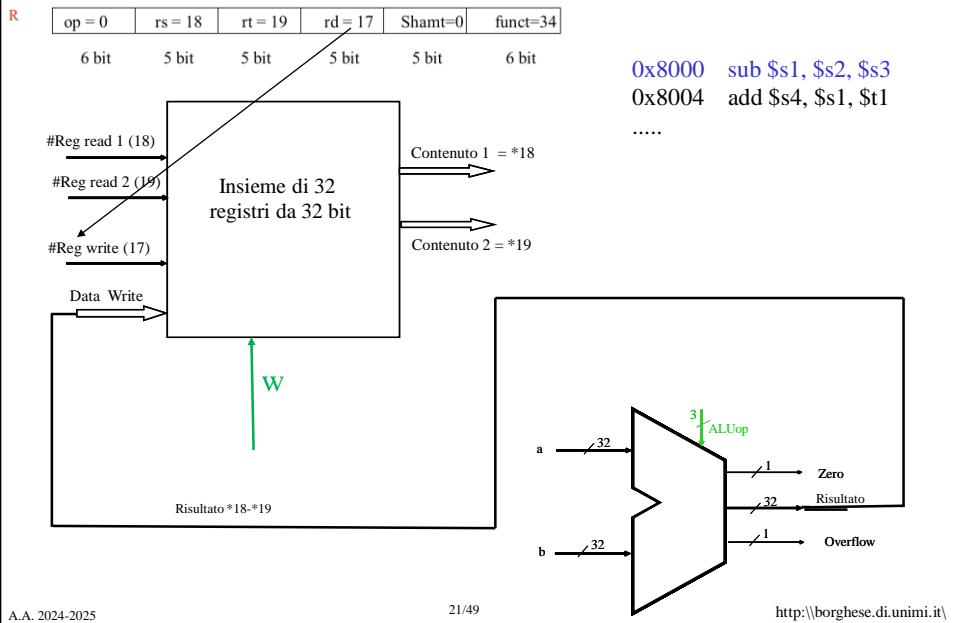


## Fase di Calcolo (tipo R)

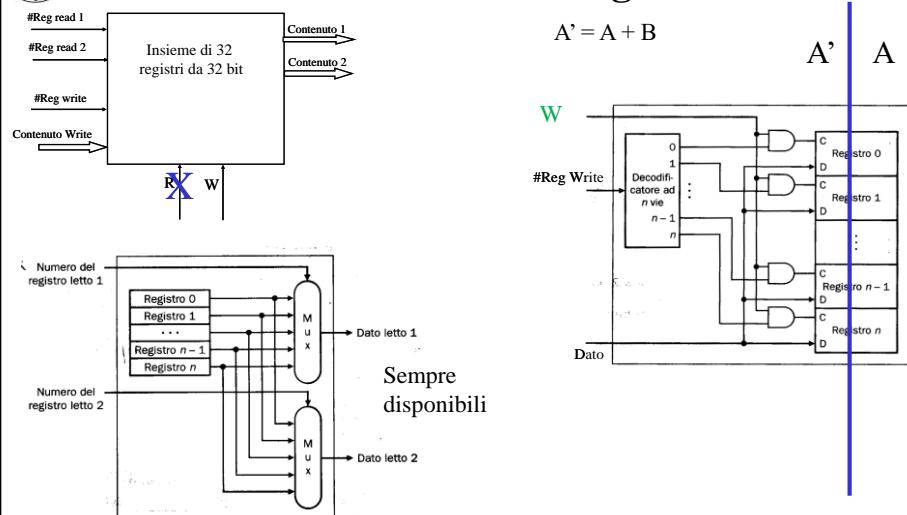




## Fase di Write back (tipo R)



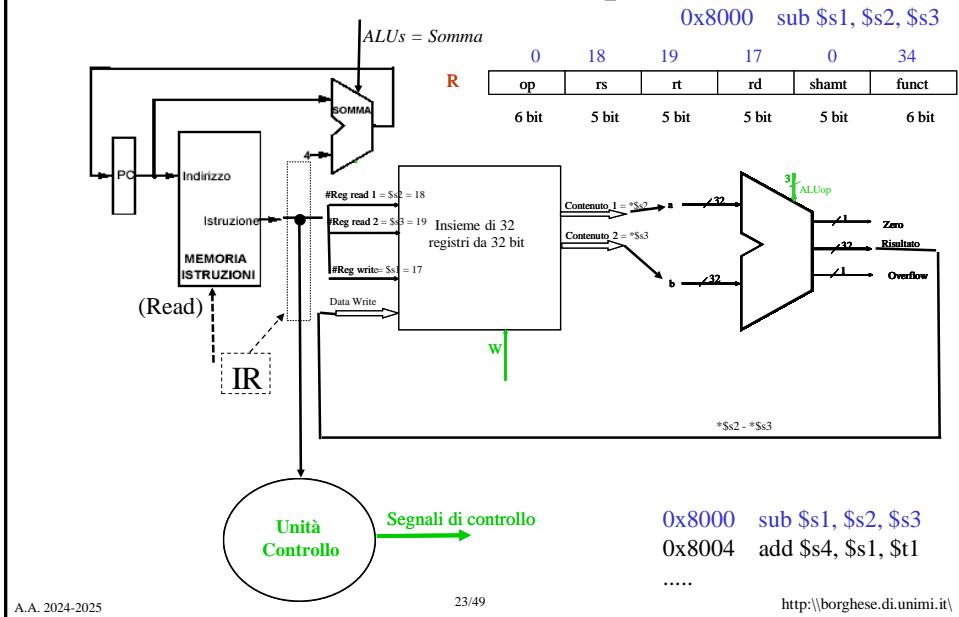
## Porta di scrittura del register file



NB Utilizzo registri flip-flop in modo da potere leggere / scrivere nello stesso ciclo di clock (scrivo nel master nella fase di WB e leggo dallo slave in fase di decodifica. La commutazione da master a slave è pilotata dal clock.



# CPU per l'esecuzione completa di un'istruzione di tipo R



# Sommario



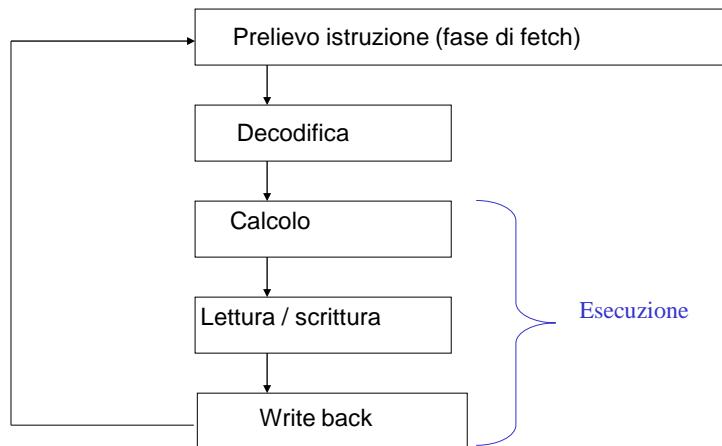
Introduzione alla CPU

CPU per le istruzioni di tipo R

CPU per le istruzioni di tipo I



# Ciclo di esecuzione di un'istruzione MIPS



# Codifica delle istruzioni



- Tutte le istruzioni MIPS hanno la **stessa dimensione (32 bit)** – **Architettura RISC**.
- I 32 bit hanno un significato diverso a seconda del formato (o tipo) di istruzione
  - il tipo di istruzione è riconosciuto in base al valore di alcuni bit (**6 bit**) più significativi (**codice operativo - OPCODE**)
- Le istruzioni MIPS sono di 3 tipi (formati):
  - Tipo R (register)** – Lavorano su 3 registri.
    - Istruzioni aritmetico-logiche.
  - Tipo I (immediate)** – Lavorano su 2 registri. L'istruzione è suddivisa in un **gruppo di 16 bit contenenti informazioni + 16 bit riservati ad una costante**.
    - Istruzioni di accesso alla memoria o operazioni contenenti delle costanti.
  - Tipo J (jump)** – Lavora senza registri: codice operativo + indirizzo di salto.
    - Istruzioni di salto incondizionato.

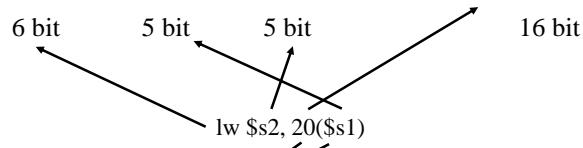
	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
R	op	rs	rt	rd	shamt	funct
I	op	rs	rt	Indirizzo / costante		
J	op	indirizzo				



## Istruzioni di tipo I: lw/sw



I	35	17	18	20
	100011	10001	10010	0000 0000 0001 0100



L'indirizzo della memoria (dati) sarà:

Base = \*\$s1  
Offset = 20

0000 1000 0011 0001 1011 1011 1011 1011 +  
0000 0000 0001 0100 +

Indirizzo finale  
\*\$s1 + 20

0000 1000 0011 0001 1011 1011 1100 1111



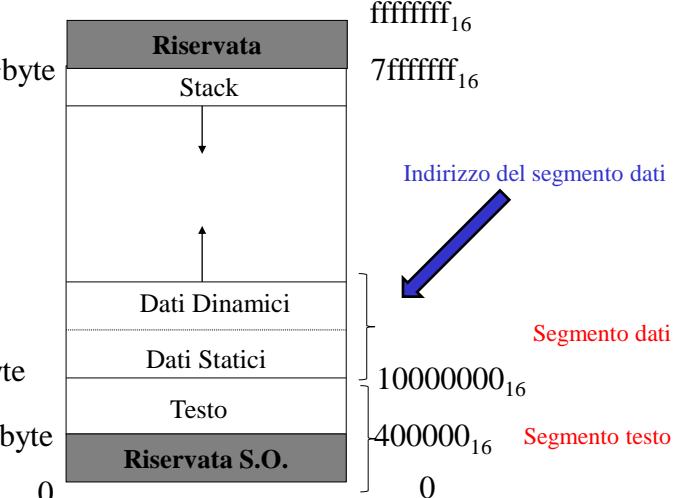
## Organizzazione logica della memoria



Max spazio di indirizzamento su 32 bit è di  $2^{32} = 4\text{Gbyte}$ .

$2^{28} = 256\text{Mbyte}$

4Mbyte  
0



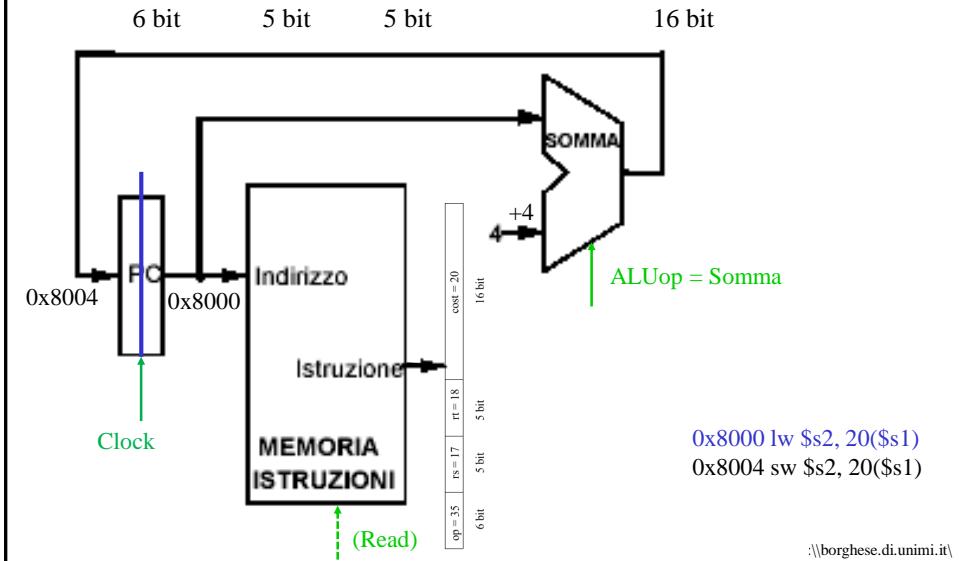


## Circuito della fase di fetch



I

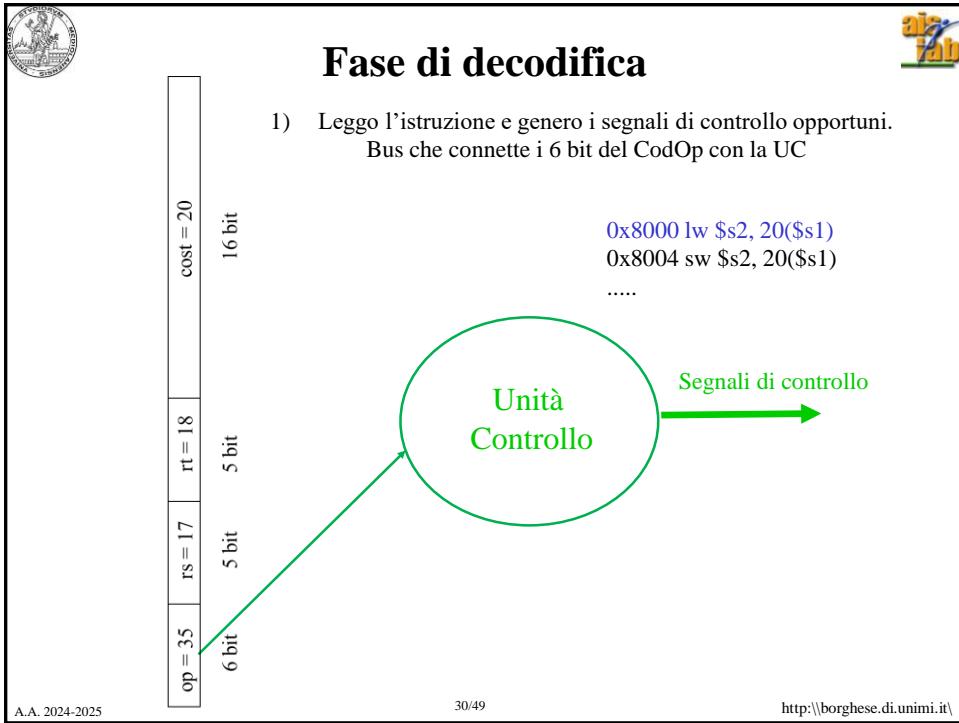
op = 35	rs = 17	rt = 18	cost = 20
---------	---------	---------	-----------



## Fase di decodifica



- 1) Leggo l'istruzione e genero i segnali di controllo opportuni.  
Bus che connette i 6 bit del CodOp con la UC

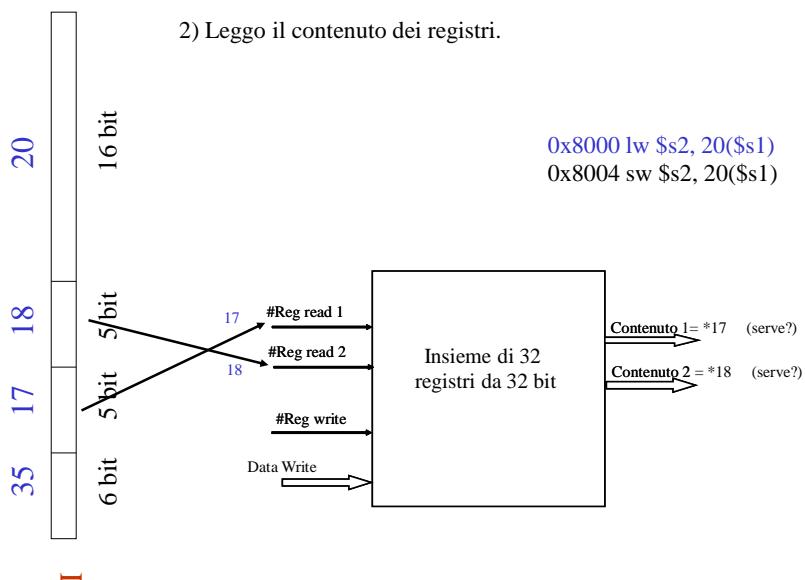




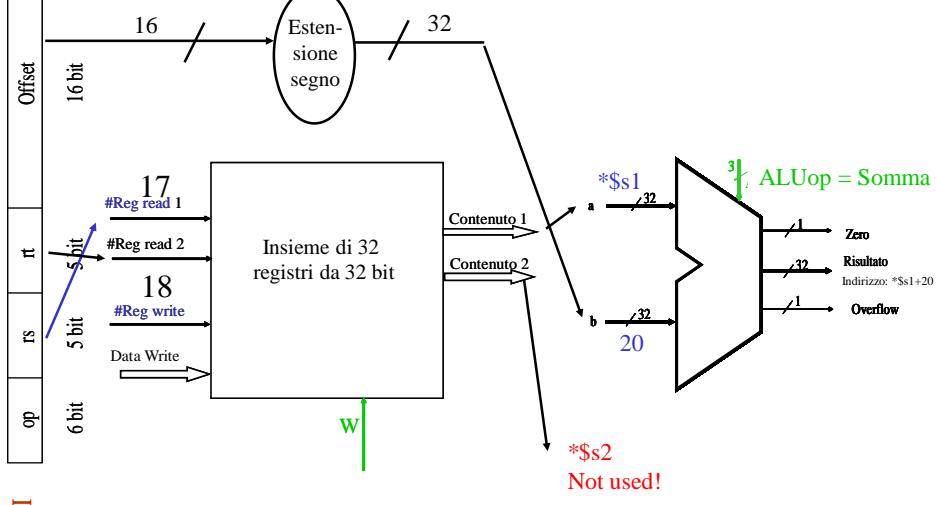
## Lettura dei registri (istruzioni di tipo I)



2) Leggo il contenuto dei registri.



## Fase di calcolo (tipo I: lw)



Il Risultato è un indirizzo della memoria:  $*s1 + 20$

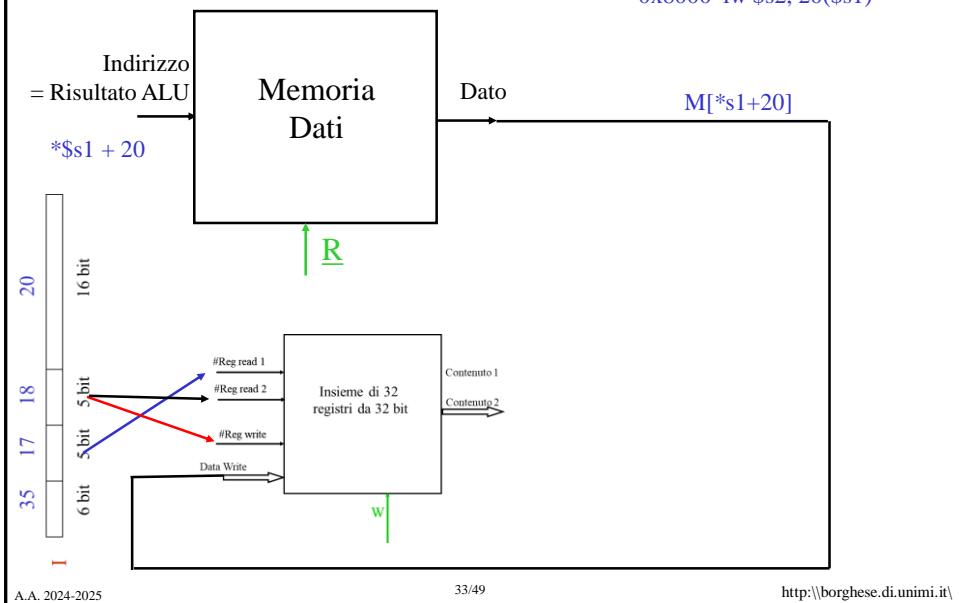
0x8000 lw \$s2, 20(\$s1)



## Lettura della memoria + write-back



0x8000 lw \$s2, 20(\$s1)

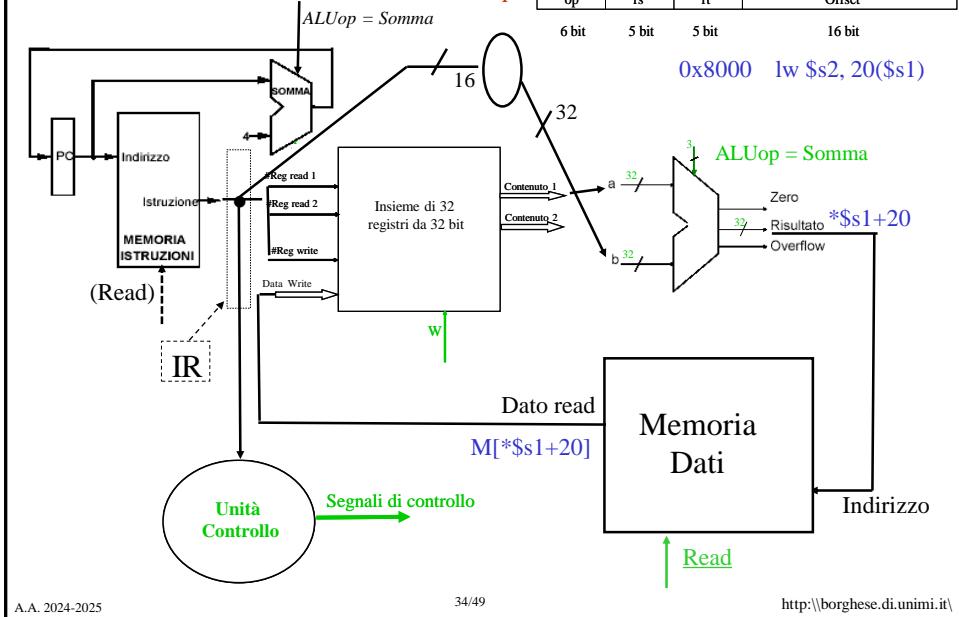


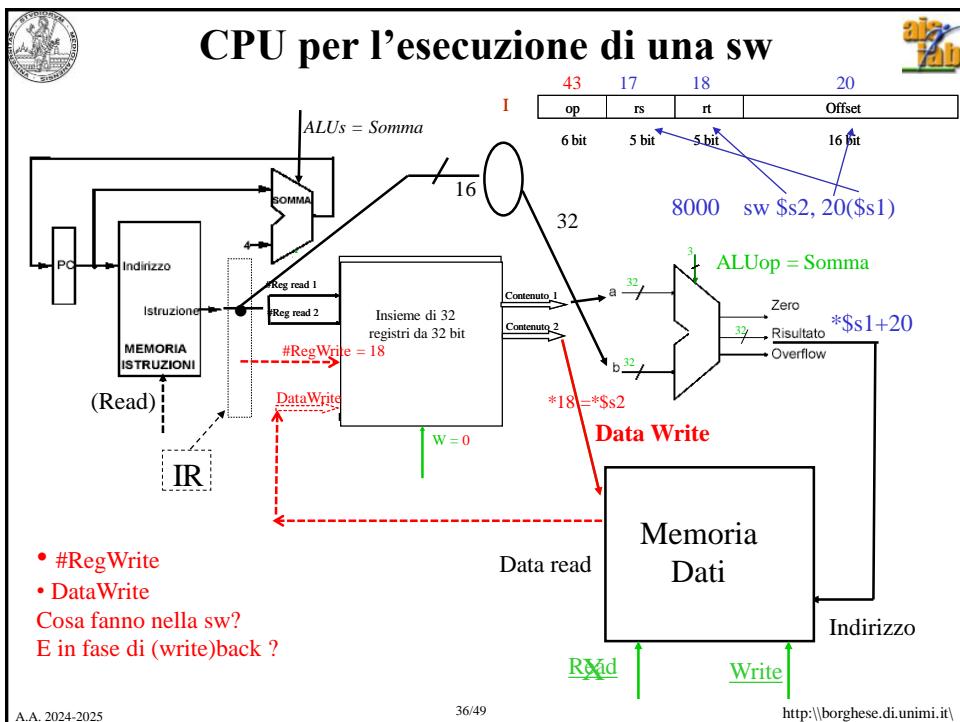
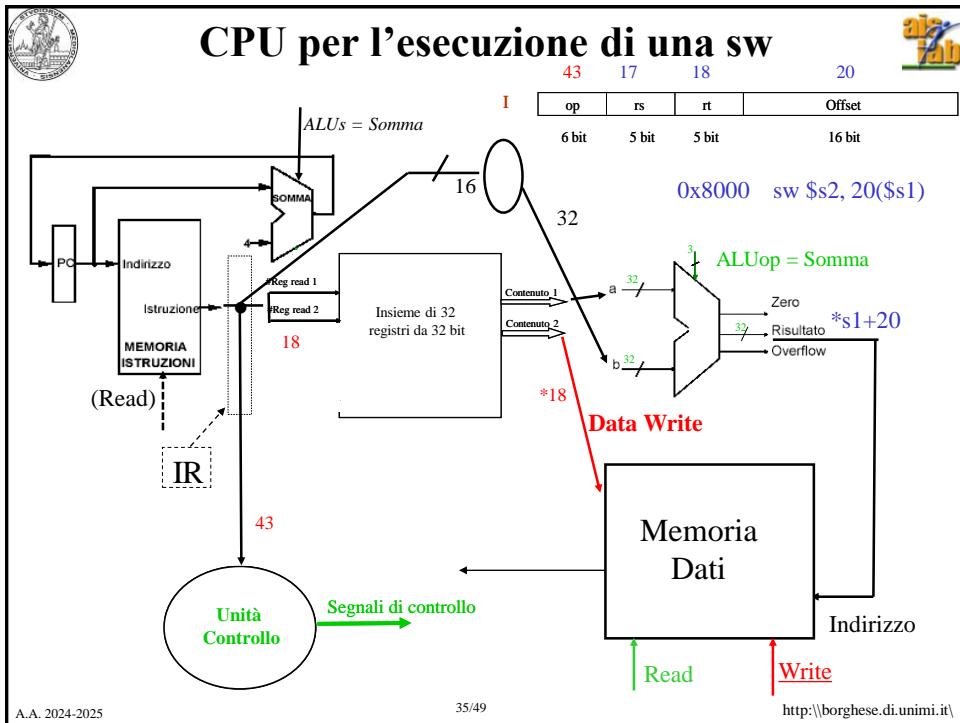
## CPU per l'esecuzione di una lw

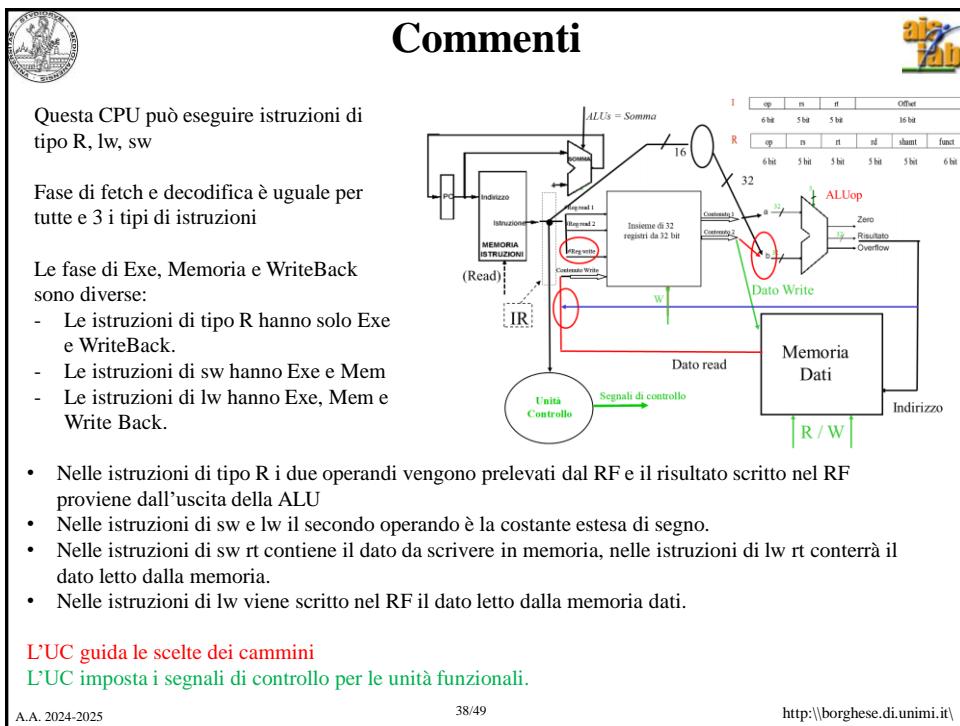
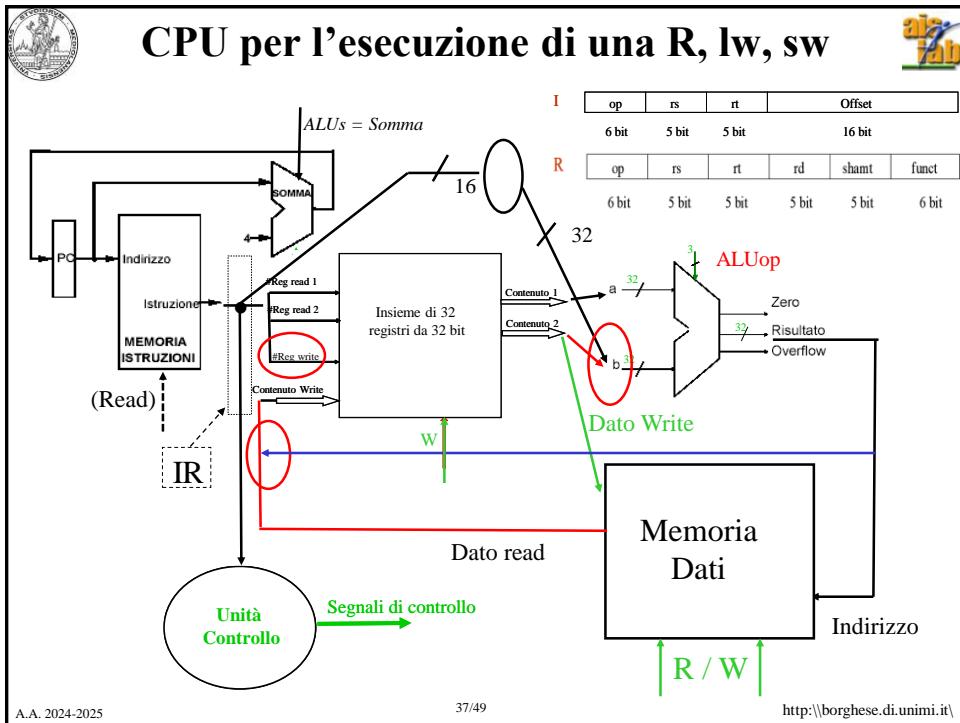


I	op	rs	rt	Offset
	6 bit	5 bit	5 bit	16 bit

0x8000 lw \$s2, 20(\$s1)









## Istruzioni di salto condizionato

- Salti condizionati relativi:

- **beq r1, r2, L1** (*branch on equal*)
  - **bne r1, r2, L1** (*branch on not equal*)

```
beq $s1, $s0, esci # if (s1 == s0) esci  
bne $s1, $s0, esci # if (s0 != s0) esci
```

- Salti condizionati relativi:

- Il flusso sequenziale di controllo cambia solo se la condizione è vera (`beq`)
  - Il calcolo del valore dell’etichetta **L1** (**indirizzo di destinazione del salto**) avviene a partire dal **Program Counter** (PC).
  - Indirizzamento del tipo Base (PC) + Spiazzamento.

- Indirizzo destinazione del salto:

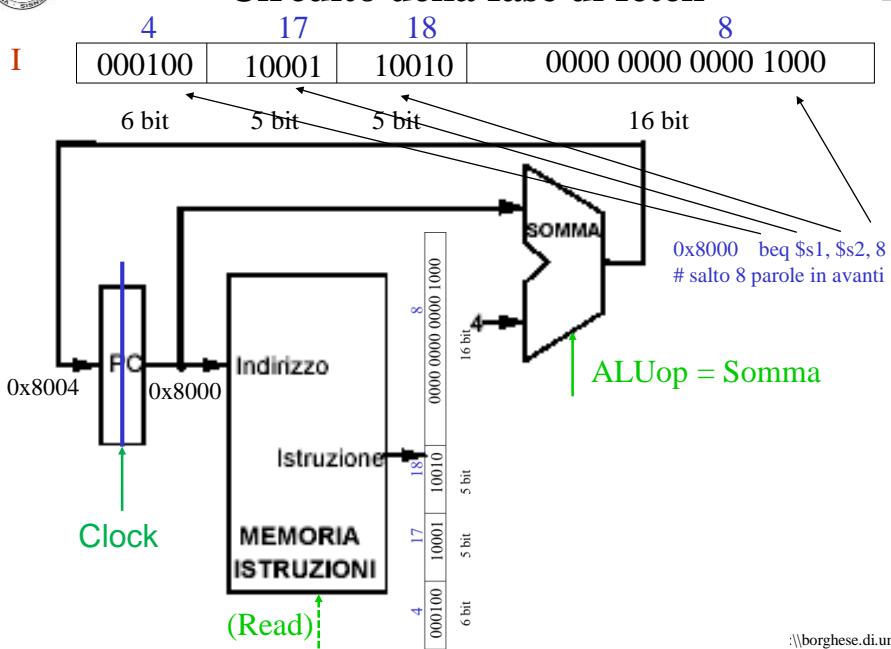
- $$= \text{PC}_{\text{curr}} \equiv (\text{PC} + 4) + \text{offset} * 4$$

Nome campo	op	rs	rt	indirizzo				
Dimensione	6-bit	5-bit	5-bit	16-bit				
beq \$s1, \$s2, L1	000100	10001	10010	0000	0000	0001	1001	

A.A. 2024-2025

39/49

<http://borghese.di.unimi.it>

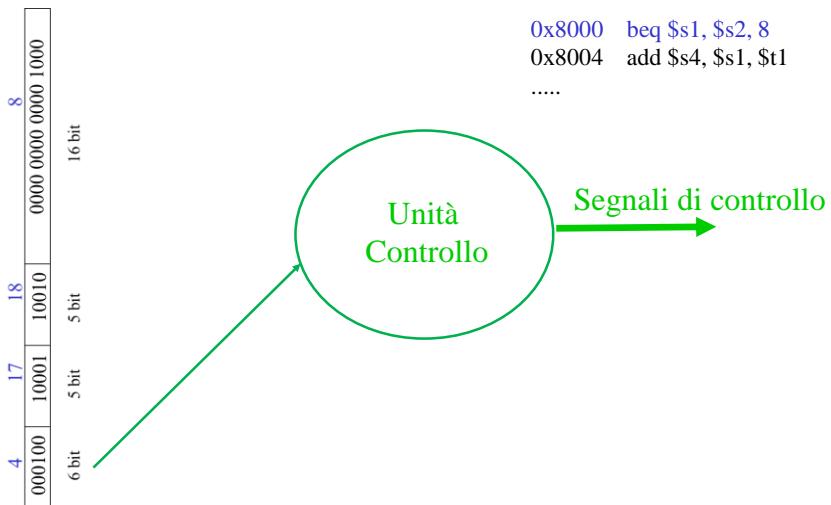




## Fase di decodifica



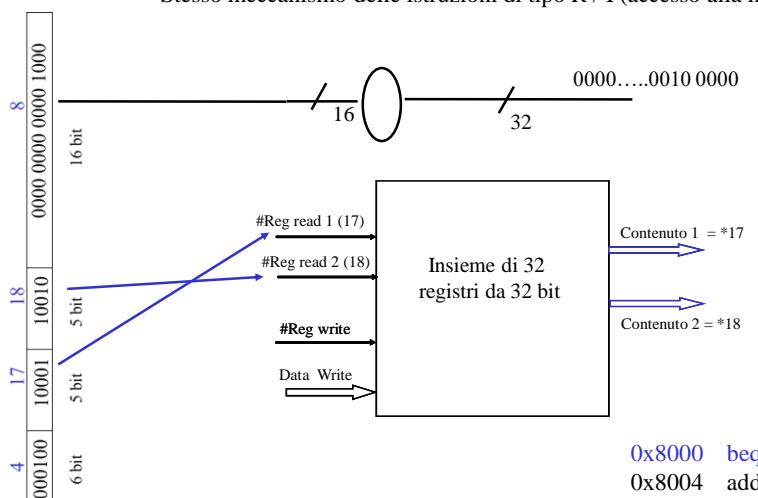
- 1) Leggo l'istruzione e genero i segnali di controllo opportuni.  
Bus che connette i 6 bit del CodOp con la UC

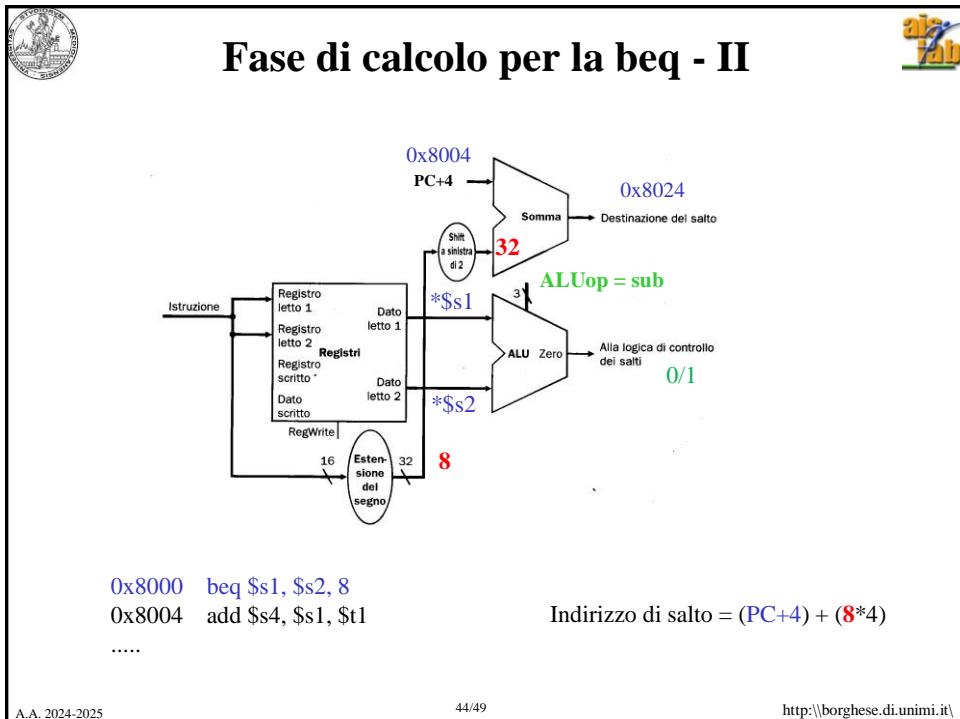
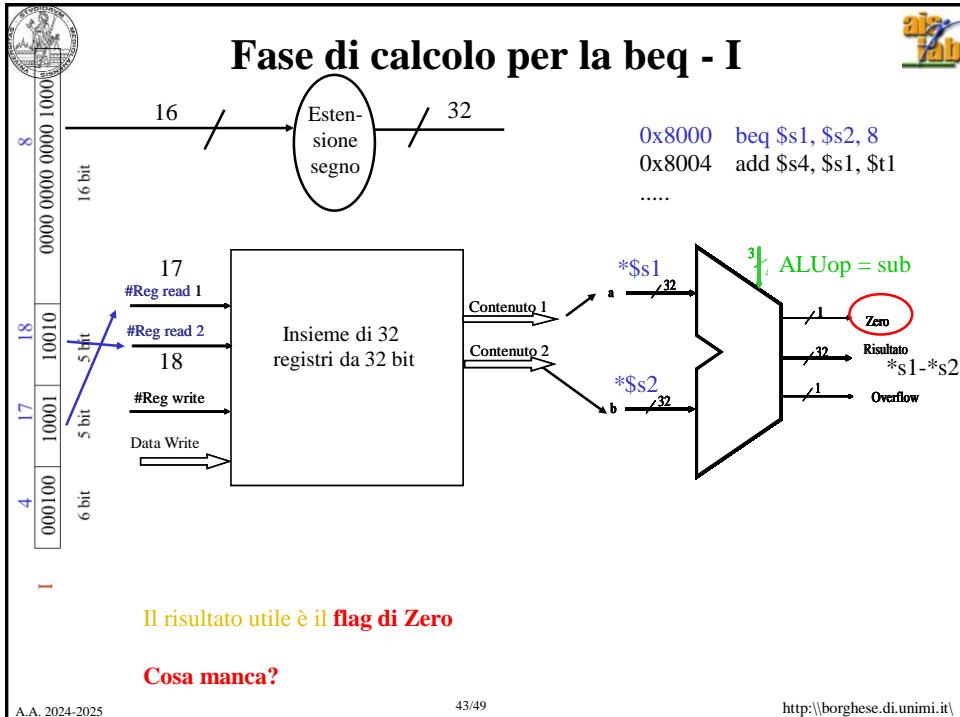


## Lettura dei registri



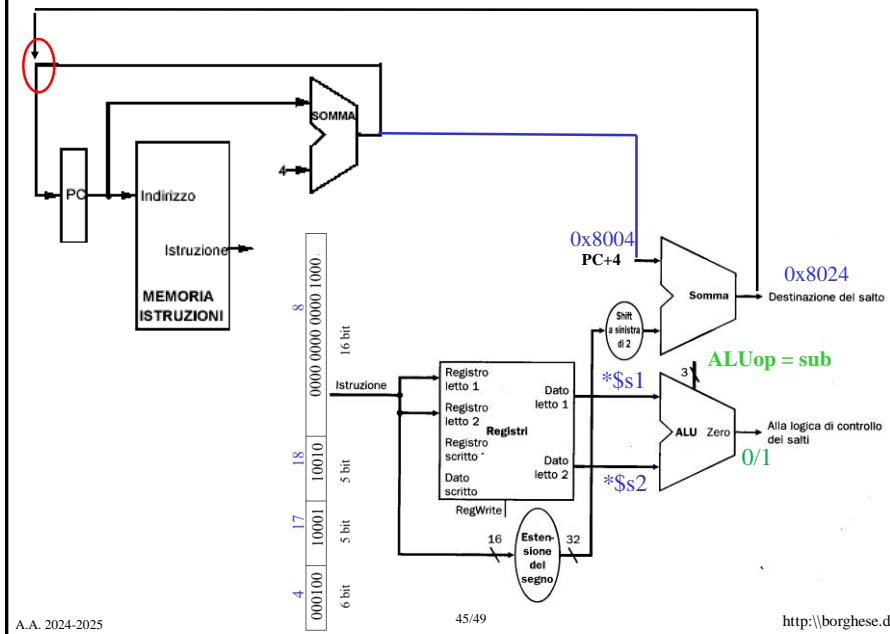
- 2) Leggo il contenuto dei registri. 2 bus che collegano i campi rs e rt con gli input corrispondenti del register file.  
Stesso meccanismo delle istruzioni di tipo R / I (accesso alla memoria)







## Implementazione del salto



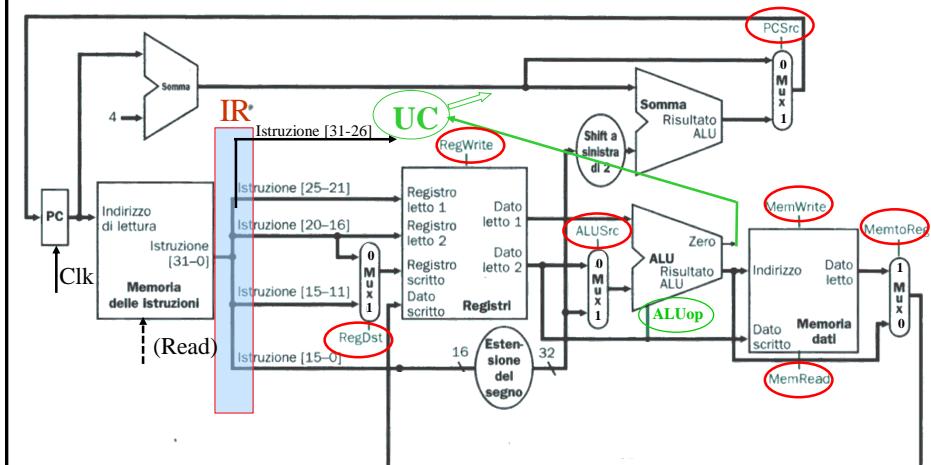
A.A. 2024-2025

<http://borghese.di.unimi.it/>

45/49



## Schema generale CPU



{R, lw, sw, addi, beq}

A.A. 2024-2025

46/49

<http://borghese.di.unimi.it/>



# Osservazioni



Il ciclo di esecuzione di un'istruzione si compie in un **unico** ciclo di clock.

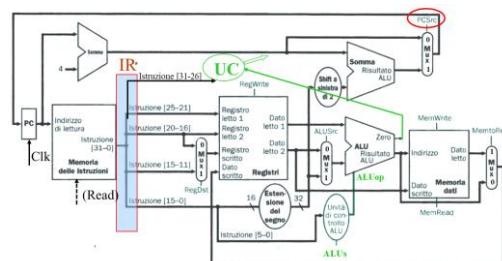


Ogni unità funzionale può essere utilizzata 1 sola volta.



Duplicazione Memoria: Memoria dati e memoria istruzioni (split memory).

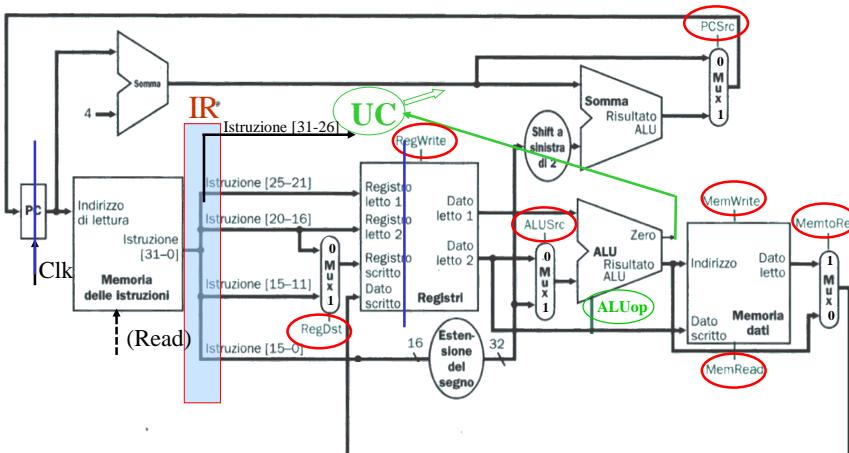
Tripli cazione ALU: 2 sommatori + 1 ALU general purpose.



A.A. 2024-2025



# Schema generale CPU



E' un'architettura retroazionata sincronizzata dal clock

Sincronizzazione del controllo (sul PC)  
Sincronizzazione sui dati (sul RF)

A.A. 2024-2025

48/49





# Sommario



Introduzione alla CPU

CPU per le istruzioni di tipo R

CPU per le istruzioni di tipo I