


CPU a singolo ciclo

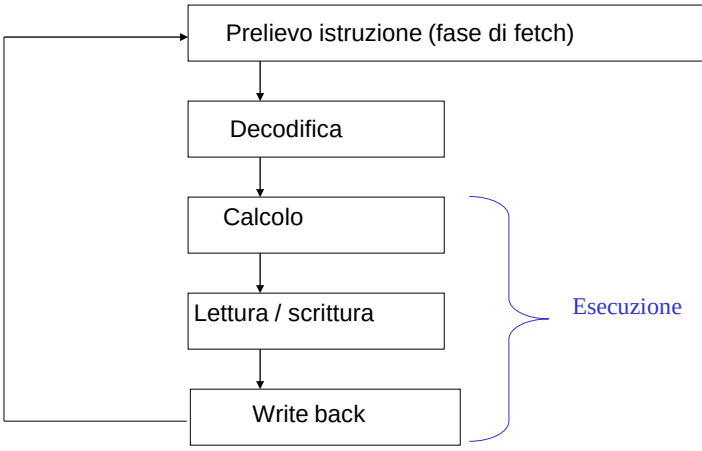
Prof. Alberto Borghese
Dipartimento di Informatica
alberto.borghese@unimi.it

Università degli Studi di Milano

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


Sommario

Ciclo di esecuzione di un'istruzione MIPS



Obbiettivo

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				

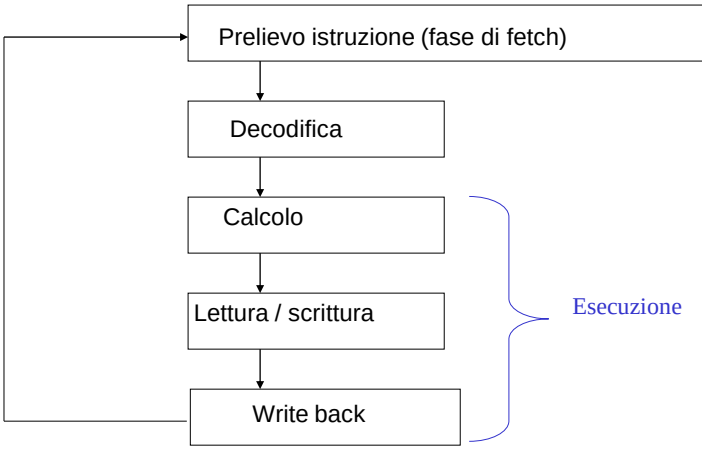
I componenti di un'architettura

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 istruzione;
- 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.

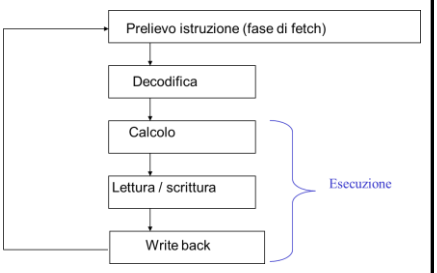
MEMORIA PRINCIPALE

Ciclo di esecuzione di un'istruzione MIPS

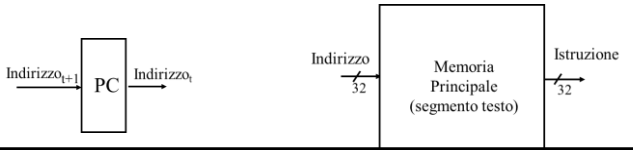


Lettura dell'istruzione (fetch)

- 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).

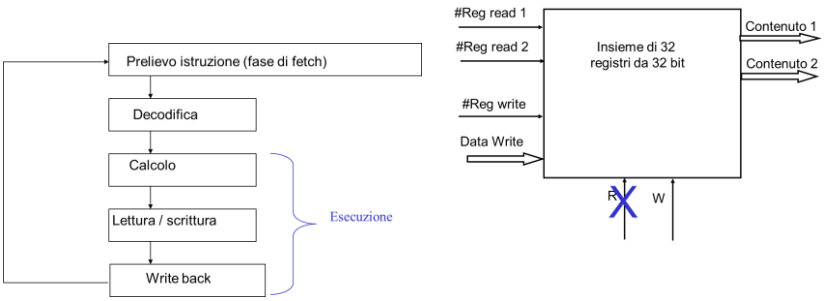


- Il segnale di controllo per la lettura (READ) viene inviato alla memoria.
- Trascorso il tempo necessario alla lettura dalla memoria, la parola indirizzata (in questo caso la prima istruzione del programma) si troverà nel registro IR.
- Il contenuto del PC viene incrementato in modo da puntare all'istruzione successiva.



Decodifica dell'istruzione

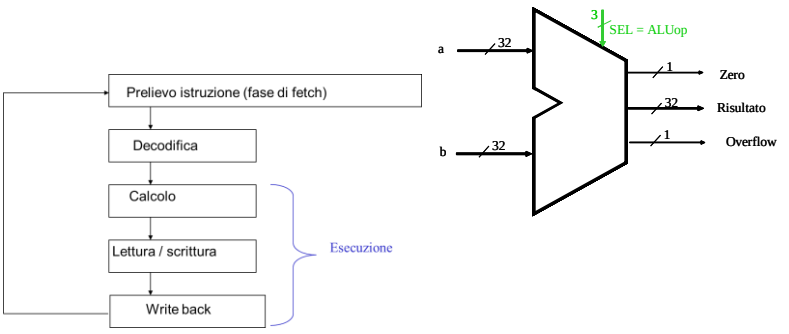
- L'istruzione contenuta nel registro IR viene decodificata per essere eseguita. Alla fase di decodifica corrisponde la predisposizione della CPU (apertura delle vie dicomunicazione appropriate) all'esecuzione dell'istruzione.
- In questa fase vengono anche recuperati gli operandi. Nelle architetture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

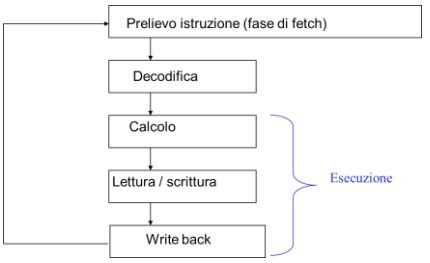


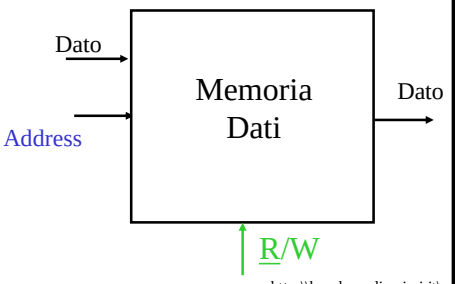
Lettura / Scrittura in memoria dati

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

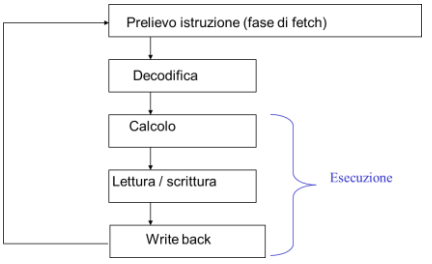
Nel caso particolare di Architetture LOAD/STORE, quali MIPS, le istruzioni di caricamente dalla memoria sono separate da quelle aritmetico/logiche. Se effettuo una Lettura / Scrittura, **non** eseguo operazioni aritmetico logiche sui dati.

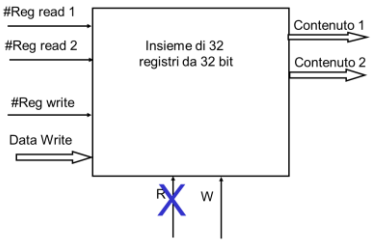




Scrittura in register file (write-back)

- 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

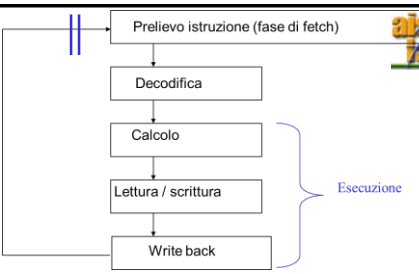
Introduzione alla CPU

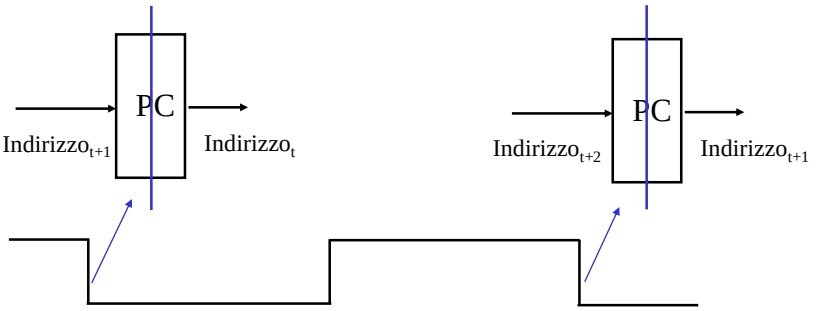
CPU per le istruzioni di tipo R

CPU per le istruzioni di tipo I

Temporizzazione

1 istruzione per ciclo di clock.
Temporizzazione del PC.

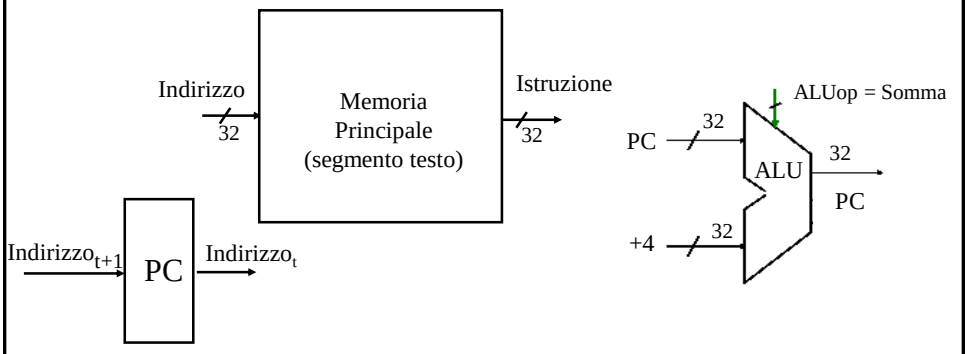




T > Tempo necessario per percorrere il cammino critico

Fase di fetch

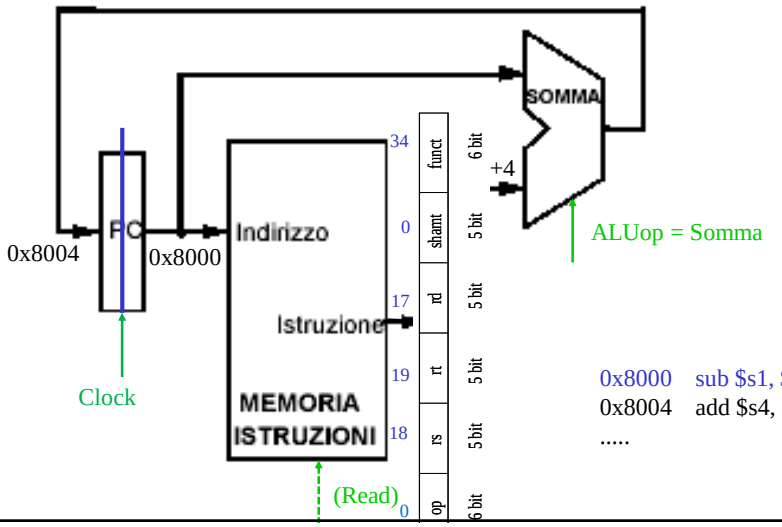
1) Memorizzare l'indirizzo dell'istruzione nel PC.
2) Leggere l'istruzione dalla memoria.
3) Aggiornare l'indirizzo in modo che in PC sia contenuto l'indirizzo dell'istruzione successiva.



Circuito della fase di fetch

R

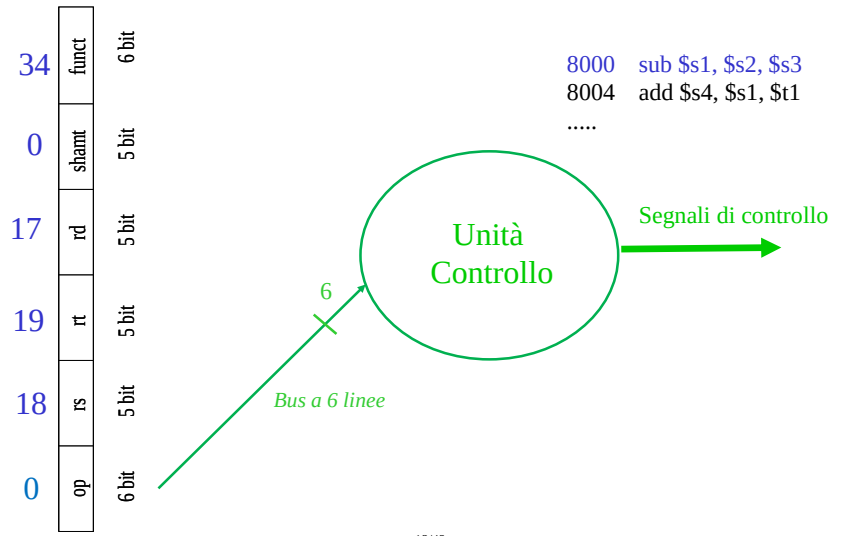
op = 0	rs = 18	rt = 19	rd = 17	Shamt=0	funct=34
6 bit	5 bit	5 bit	5 bit	5 bit	6 bit



0x8000 sub $s1, $s2, $s3
0x8004 add $s4, $s1, $t1
.....

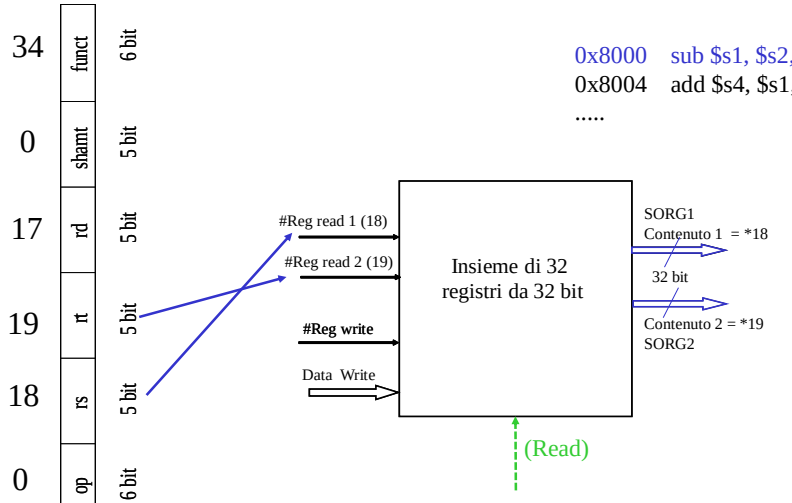
Fase di decodifica

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



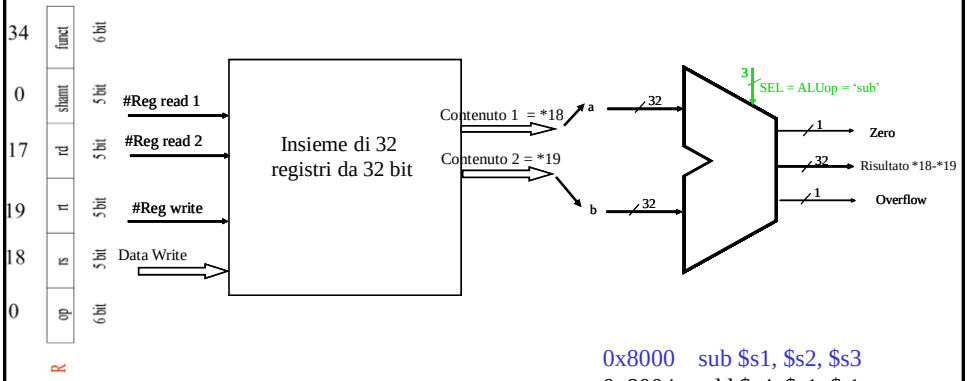
Lettura dei registri (istruzioni di tipo R)

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



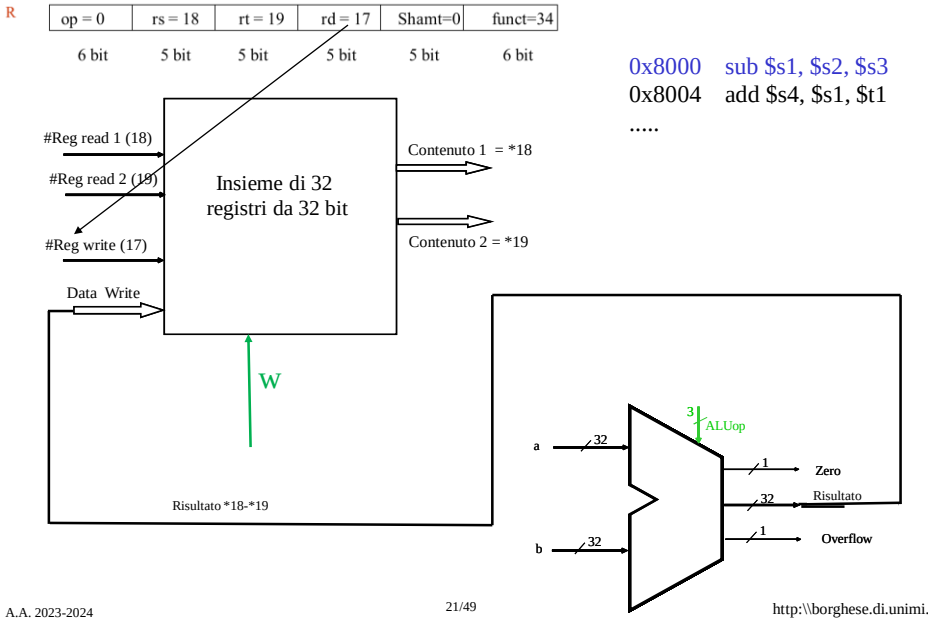
0x8000 sub $s1, $s2, $s3
0x8004 add $s4, $s1, $t1
.....

Fase di Calcolo (tipo R)



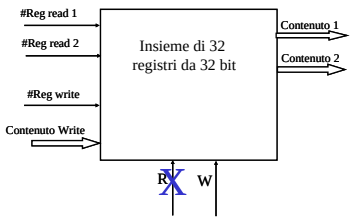
0x8000 sub $s1, $s2, $s3
0x8004 add $s4, $s1, $t1
.....

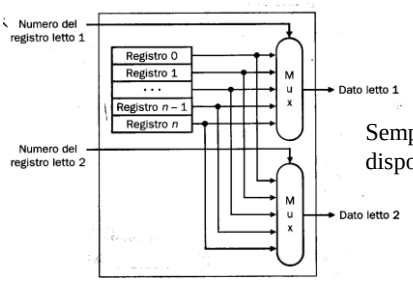
Fase di Write back (tipo R)



0x8000 sub $s1, $s2, $s3
0x8004 add $s4, $s1, $t1
.....

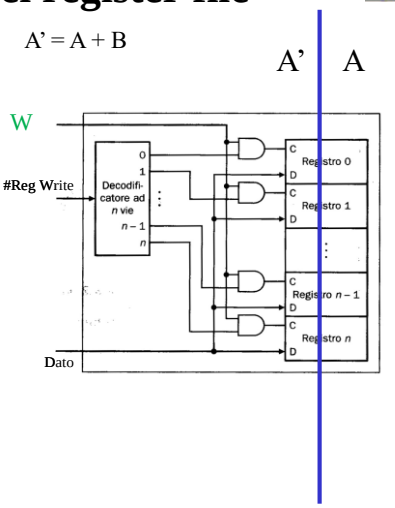
Porta di scrittura del register file





Sempre disponibili

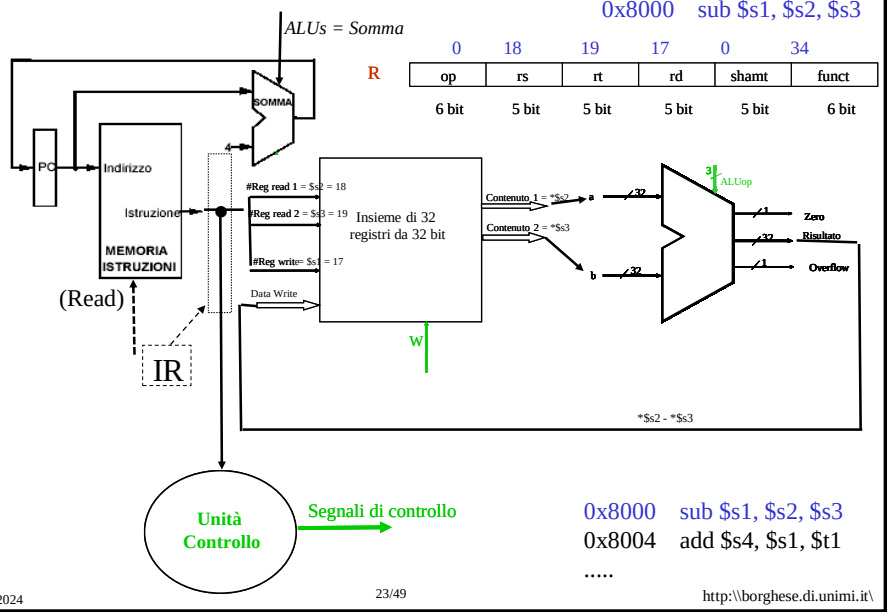
A' = A + B



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.

A

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



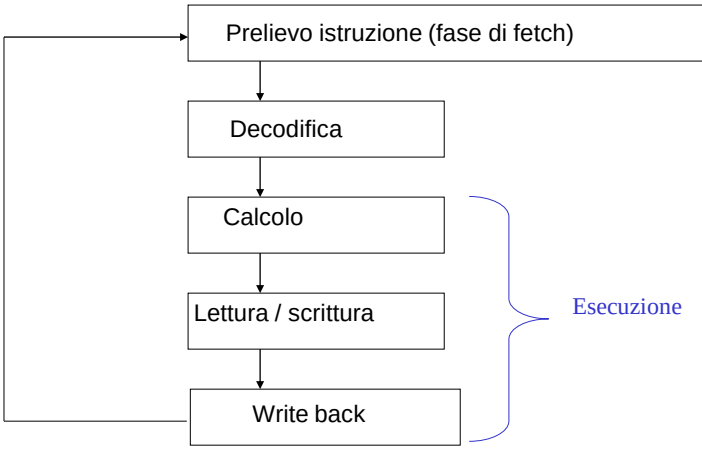
Sommario

Introduzione alla CPU

CPU per le istruzioni di tipo R

CPU per le istruzioni di tipo I

Ciclo di esecuzione di un'istruzione MIPS

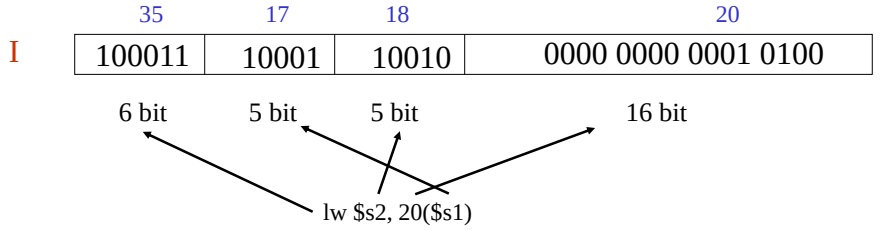


Codifica delle istruzioni

- 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

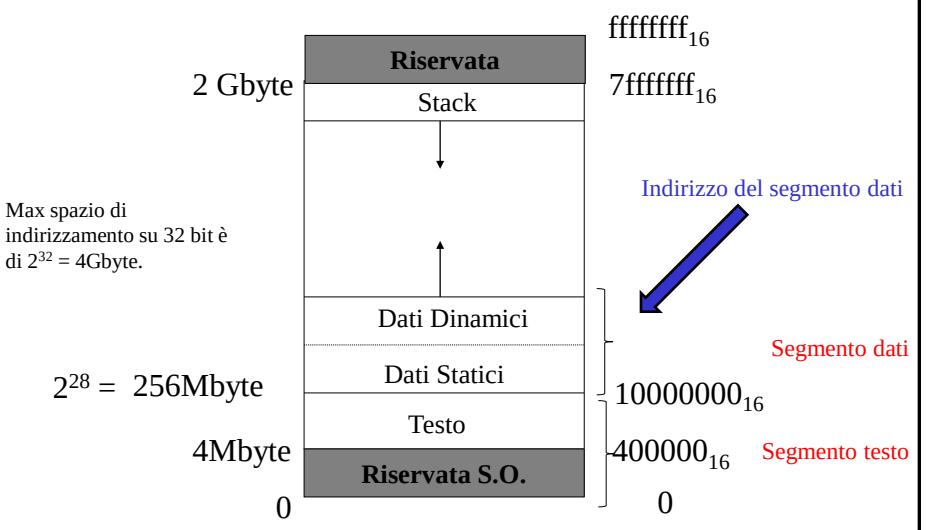


L'indirizzo della memoria (dati) sarà:

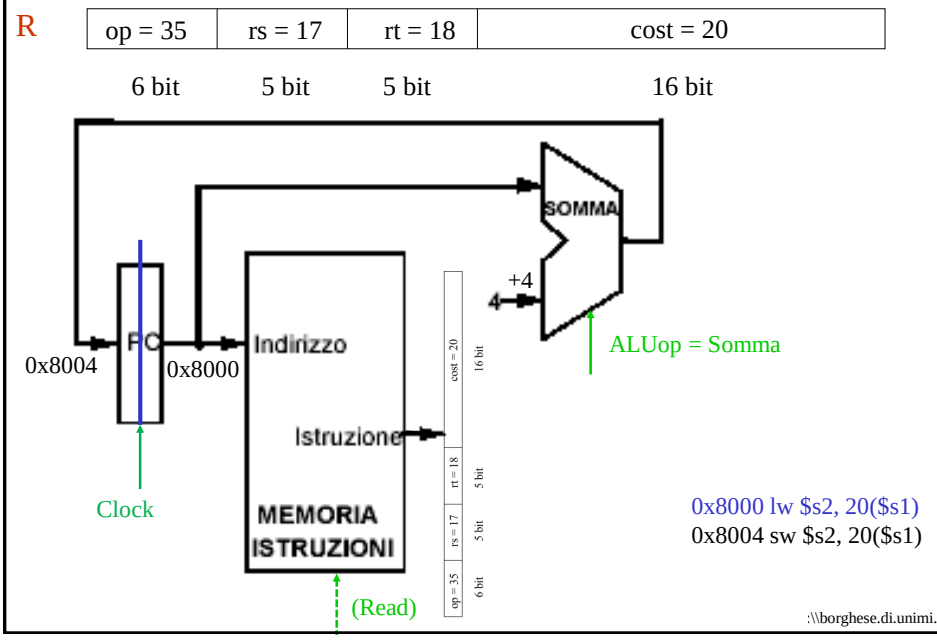
Base = *$s1 0000 1000 0011 0001 1011 1011 1011 1011 +
Offset = 20 0000 0000 0001 0100 +

Indirizzo finale 0000 1000 0011 0001 1011 1011 1100 1111
*$s1 + 20

Organizzazione logica della memoria



Circuito della fase di fetch

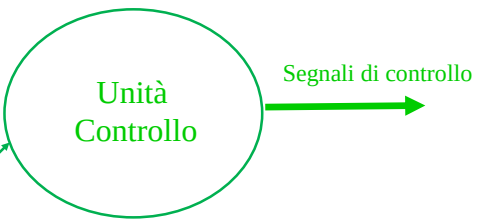


0x8000 lw $s2, 20($s1)
0x8004 sw $s2, 20($s1)

Fase di decodifica

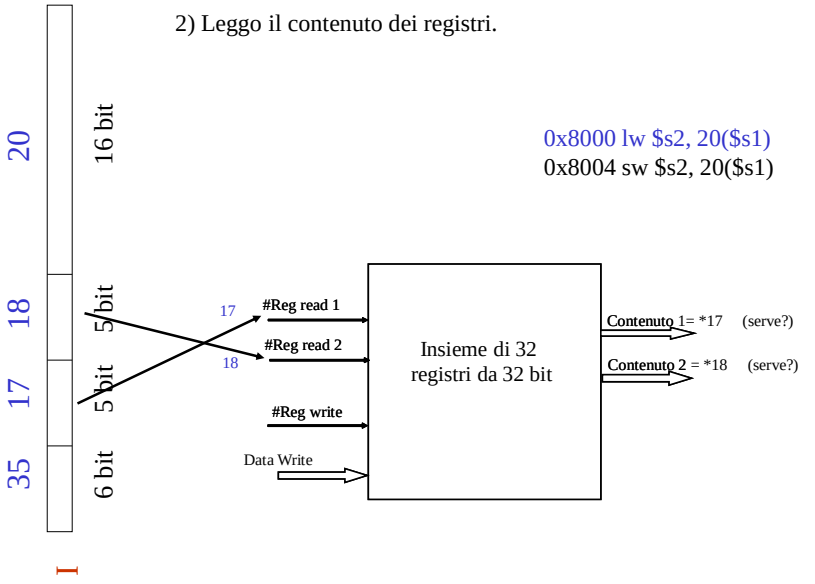
1) Leggo l'istruzione e genero i segnali di controllo opportuni.
Bus che connette i 6 bit del CodOp con la UC

0x8000 lw $s2, 20($s1)
0x8004 sw $s2, 20($s1)
.....



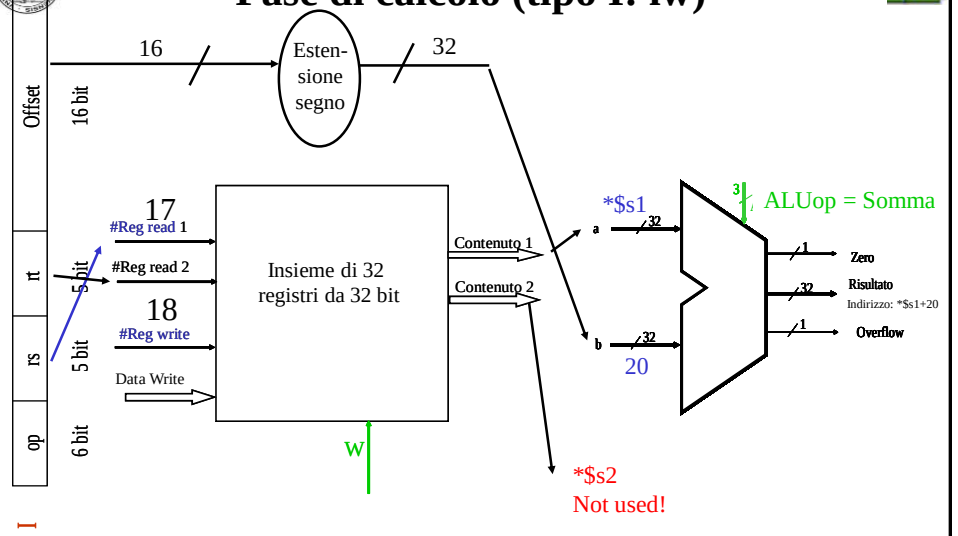
Lettura dei registri (istruzioni di tipo I)

2) Leggo il contenuto dei registri.



0x8000 lw $s2, 20($s1)
0x8004 sw $s2, 20($s1)

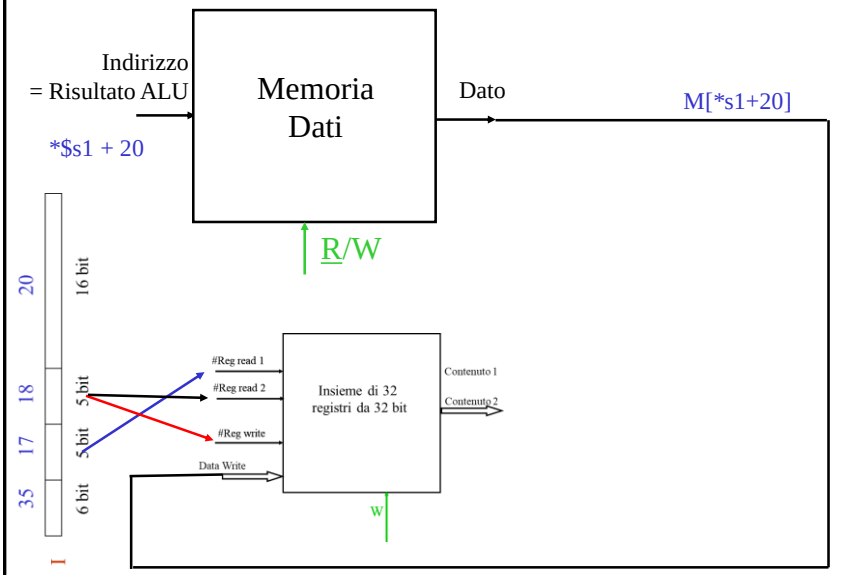
Fase di calcolo (tipo I: lw)



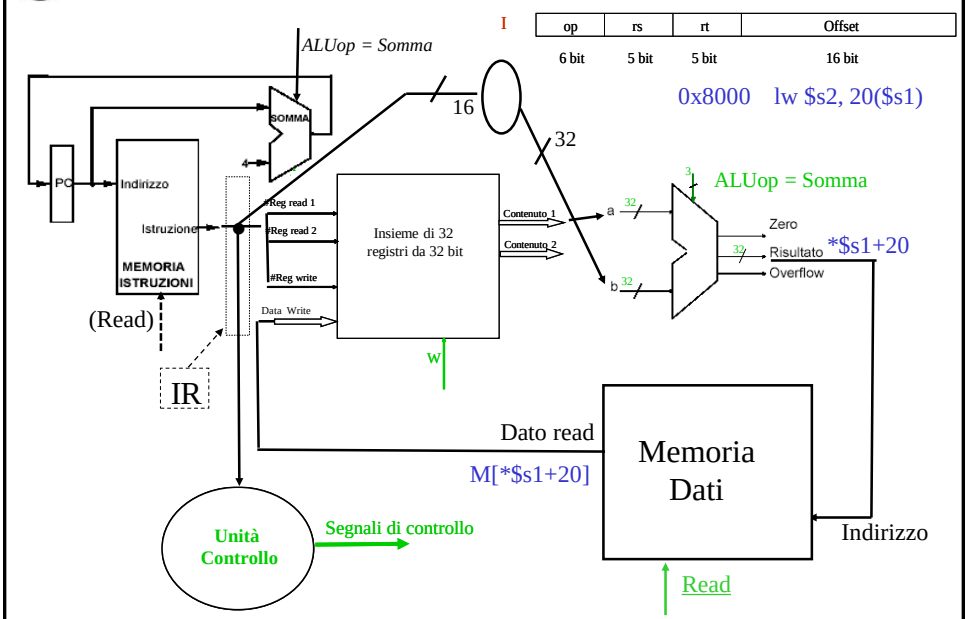
Il Risultato è un indirizzo della memoria: *s1 + 20

0x8000 lw $s2, 20($s1)

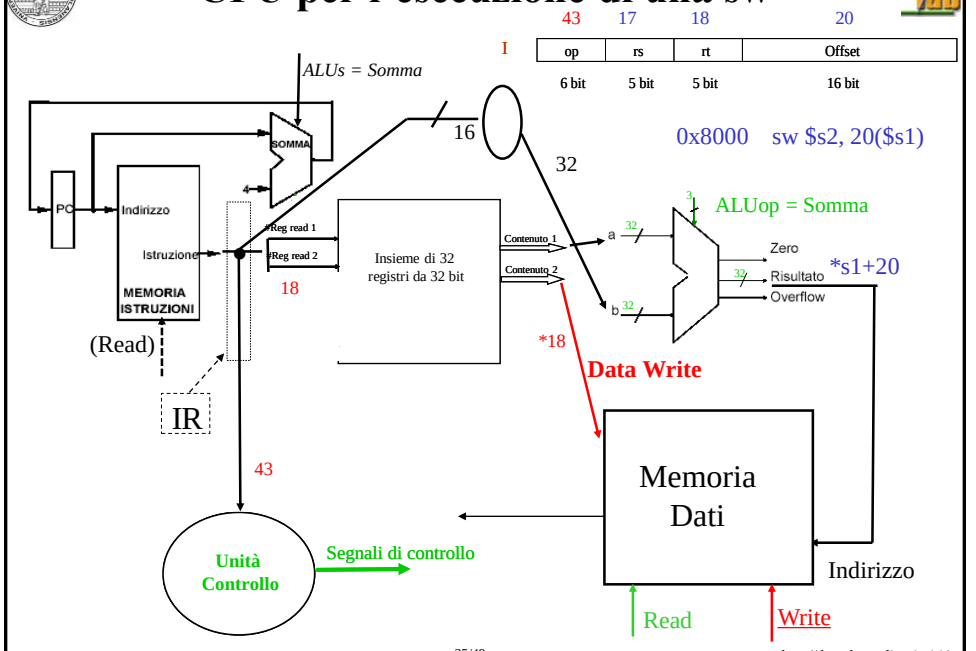
Lettura della memoria + write-back



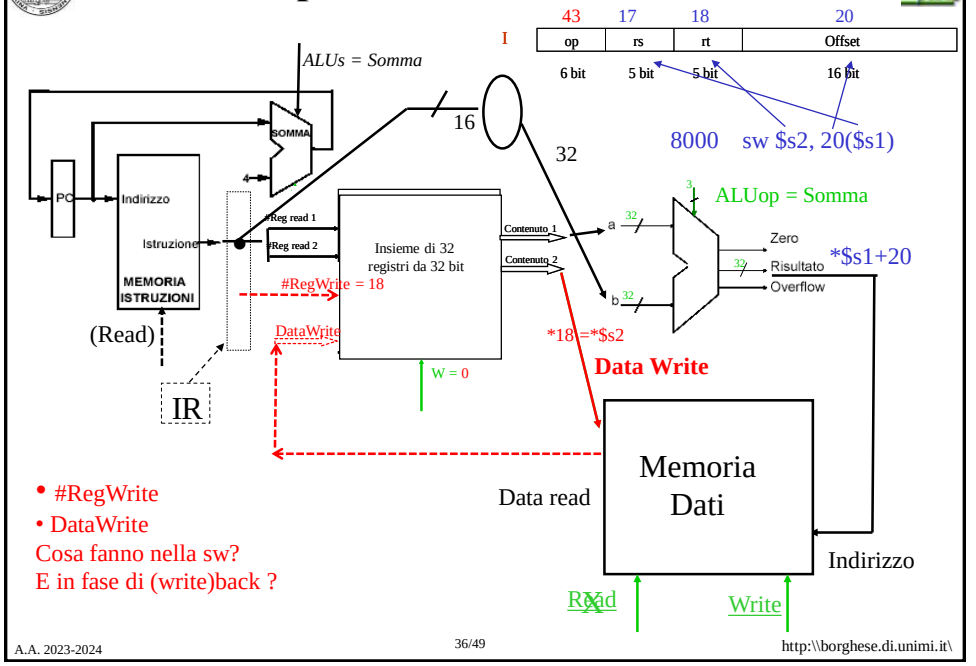
CPU per l'esecuzione di una lw



CPU per l'esecuzione di una sw



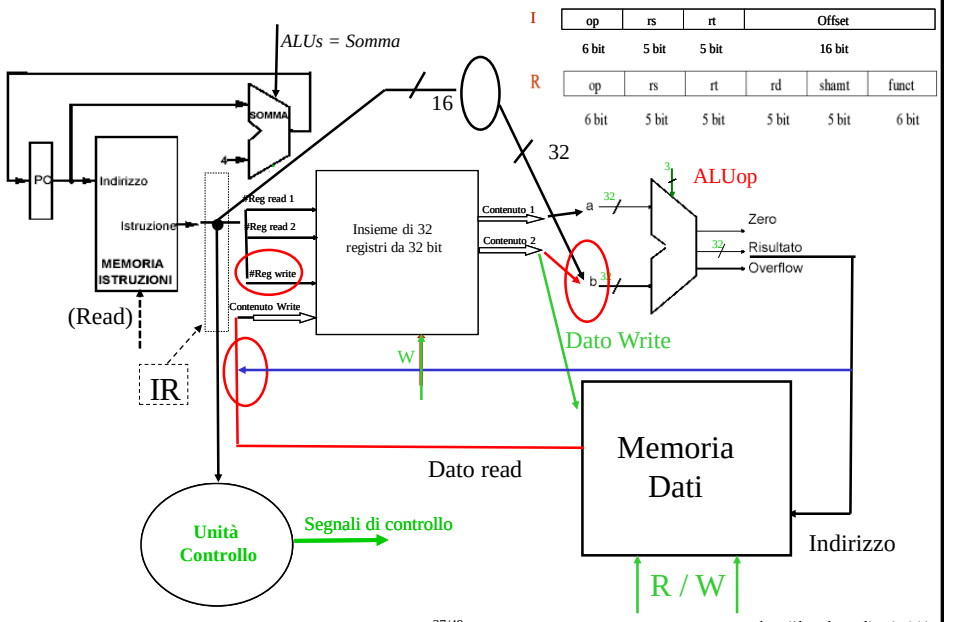
CPU per l'esecuzione di una sw



- #RegWrite
- DataWrite

Cosa fanno nella sw?
E in fase di (write)back ?

CPU per l'esecuzione di una R, lw, sw



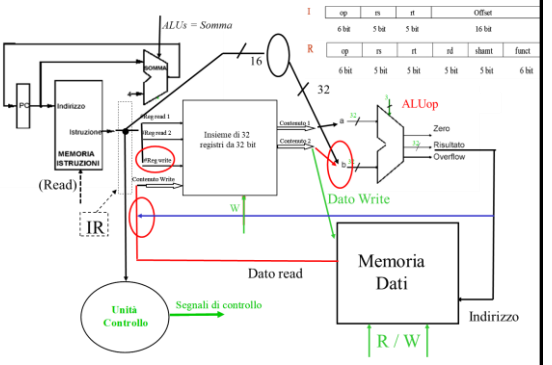
Commenti

Questa CPU può eseguire istruzioni di tipo R, lw, sw

Fase di fetch e decodifica è uguale per tutte e 3 i tipi di istruzioni

Le fase di Exe, Memoria e WriteBack sono diverse:

- Le istruzioni di tipo R hanno solo Exe e WriteBack.
- Le istruzioni di sw hanno Exe e Mem
- Le istruzioni di lw hanno Exe, Mem e Write Back.



- Nelle istruzioni di tipo R i due operandi vengono prelevati dal RF e il risultato scritto nel RF proviene dall'uscita della ALU
- Nelle istruzioni di sw e lw il secondo operando è la costante estesa di segno.
- Nelle istruzioni di sw rt contiene il dato da scrivere in memoria, nelle istruzioni di lw rt conterrà il dato letto dalla memoria.
- Nelle istruzioni di lw viene scritto nel RF il dato letto dalla memoria dati.

L'UC guida le scelte dei cammini

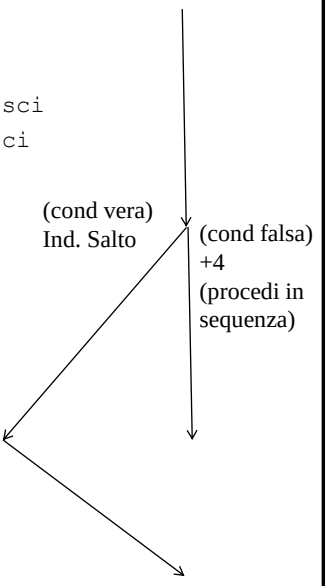
L'UC imposta i segnali di controllo per le unità funzionali.

Istruzioni di salto condizionato

- Salti condizionati relativi:
  - **`beq r1, r2, L1`** (*branch on equal*)
  - **`bne r1, r2, L1`** (*branch on not equal*)

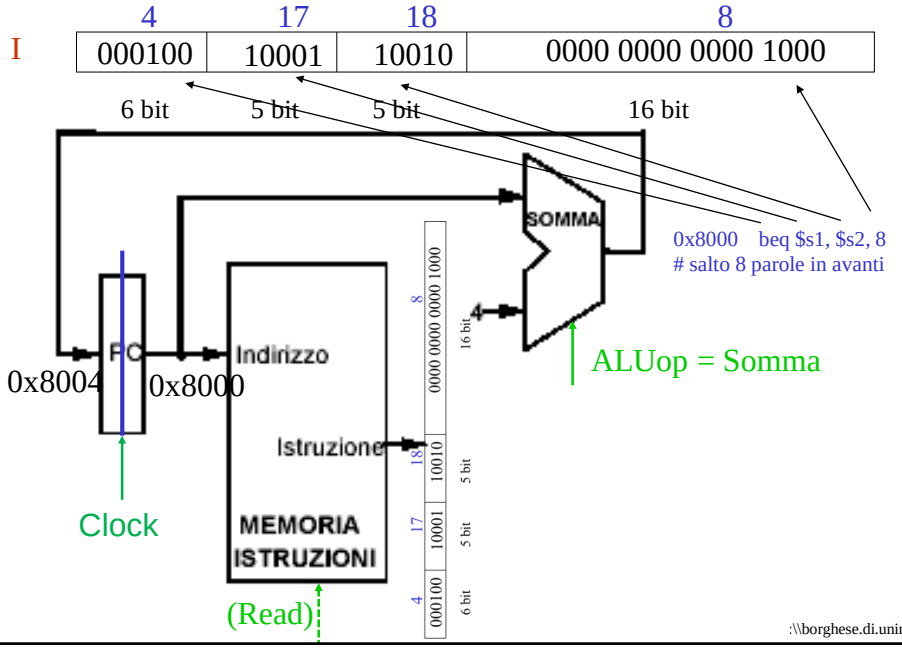
```
beq $s1, $s0, esci # if (s1 == s0) esci
bne $s1, $s0, esci # if (s0 ≠ s0) esci
```

- 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:
  - PC_{dest} = (PC + 4) + offset * 4



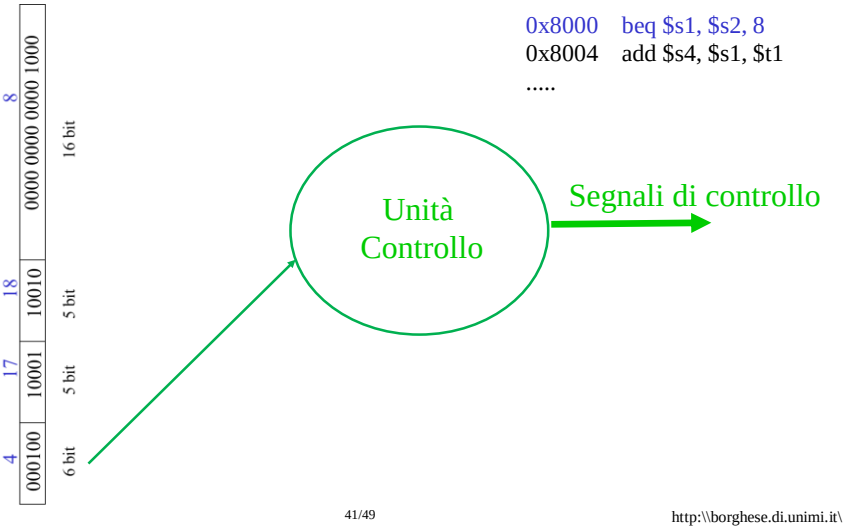
Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`beq $s1, $s2, L1`	000100	10001	10010	0000 0000 0001 1001

Circuito della fase di fetch



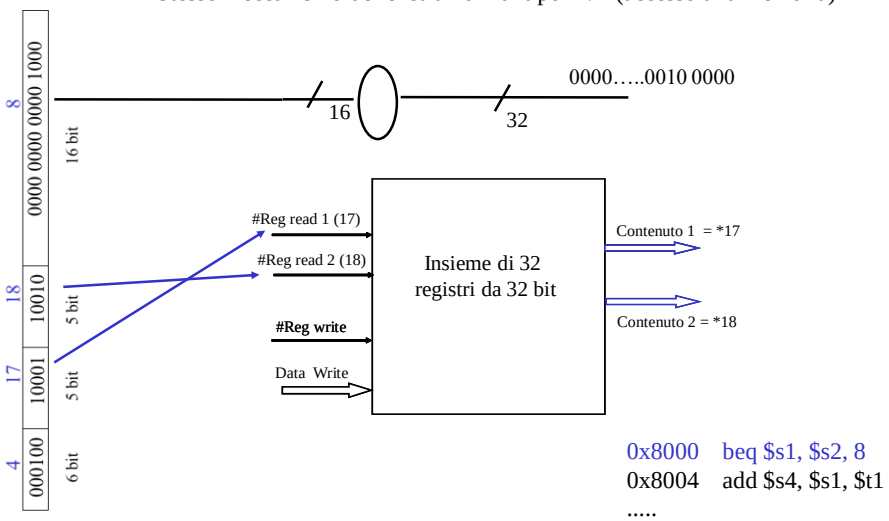
Fase di decodifica

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

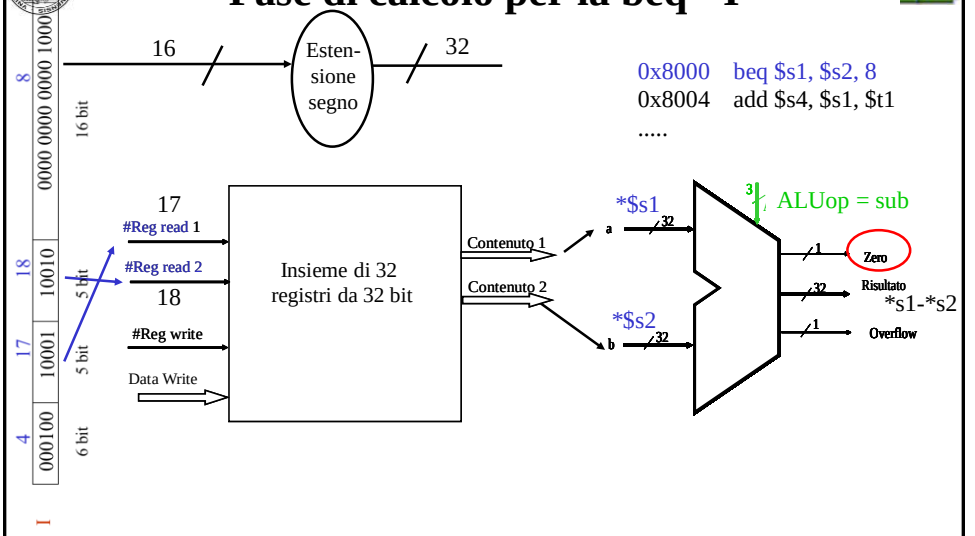


Lettura dei registri

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



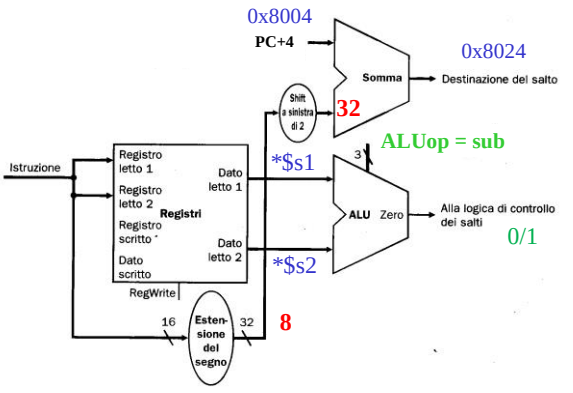
Fase di calcolo per la beq - I



Il risultato utile è il **flag di Zero**

Cosa manca?

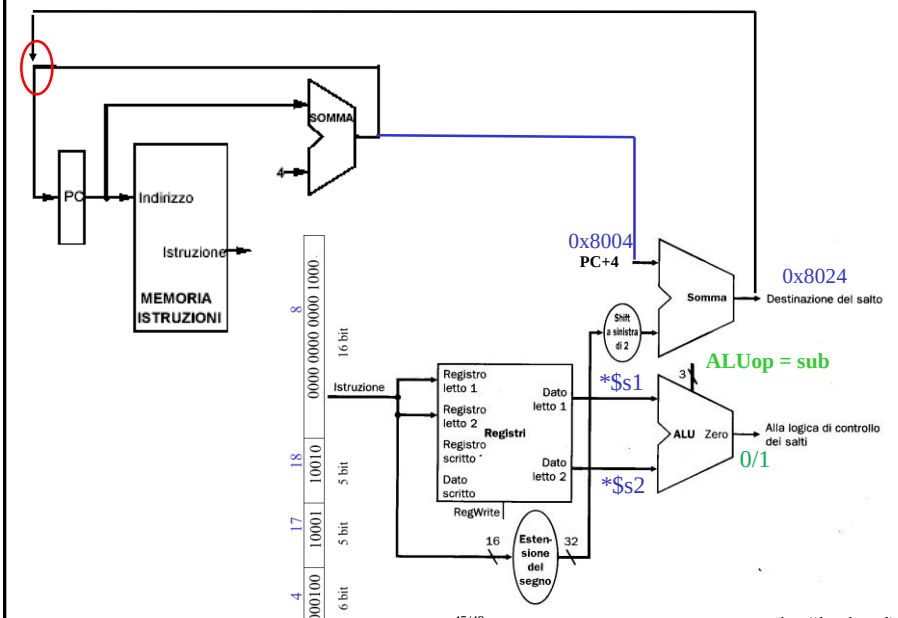
Fase di calcolo per la beq - II



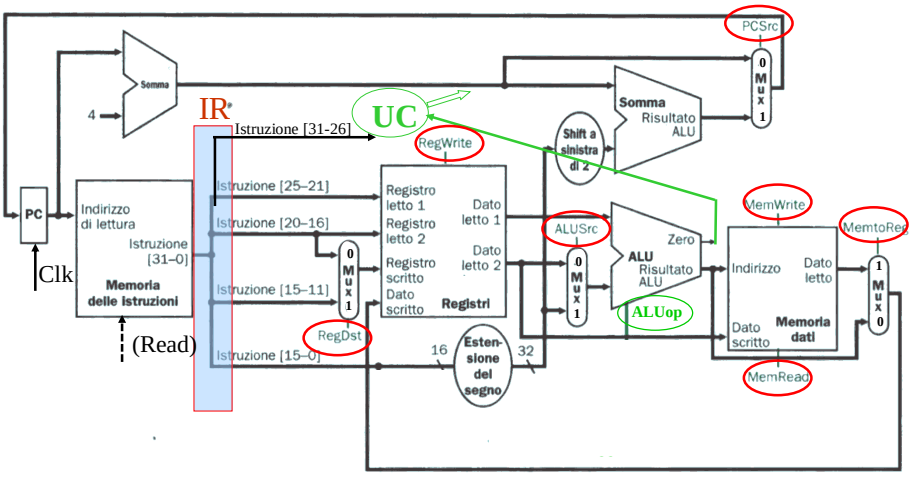
0x8000 beq $s1, $s2, 8
0x8004 add $s4, $s1, $t1
.....

Indirizzo di salto = (PC+4) + (**8***4)

Implementazione del salto



Schema generale CPU



{R, lw, sw, addi, beq}

Osservazioni

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

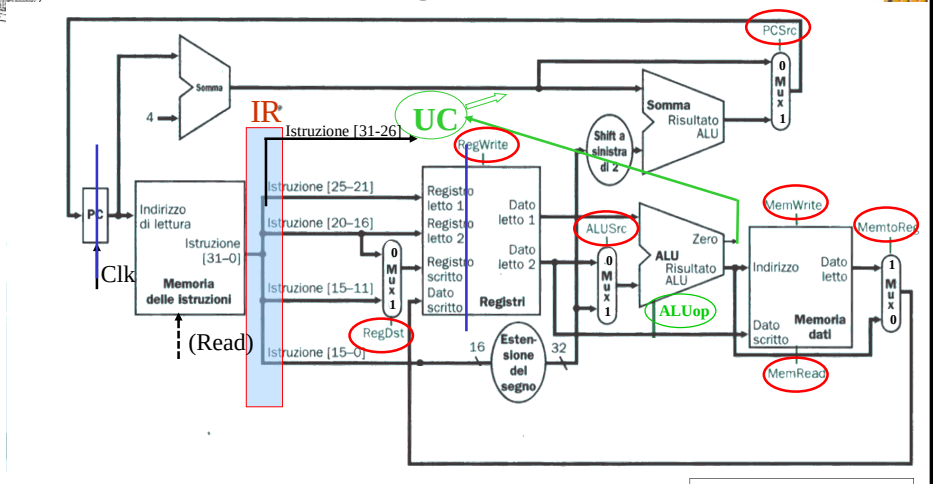
⇩

Ogni unità funzionale può essere utilizzata 1 sola volta.

⇩

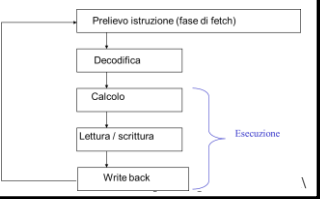
Duplicazione Memoria: Memoria dati e memoria istruzioni (split memory).
Triplicazione ALU: 2 sommatori + 1 ALU general purpose.

Schema generale CPU



E' un'architettura retroazionata sincronizzata dal clock

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



Sommario

Introduzione alla CPU

CPU per le istruzioni di tipo R

CPU per le istruzioni di tipo I