


Architettura degli elaboratori
CPU a ciclo singolo

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

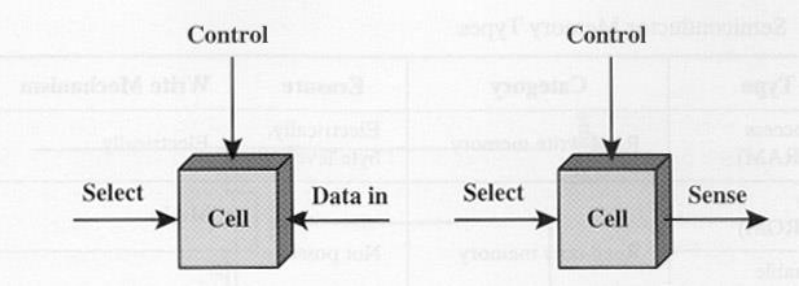
Università degli Studi di Milano

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


Sommario

Cella di memoria

La memoria è suddivisa in celle, ciascuna delle quali assume un valore binario stabile.
Si può scrivere il valore 0/1 in una cella.
Si può leggere il valore di ciascuna cella.

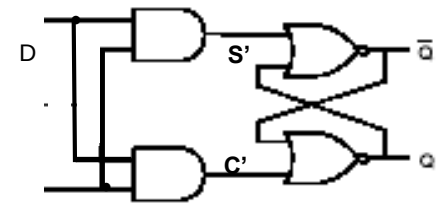


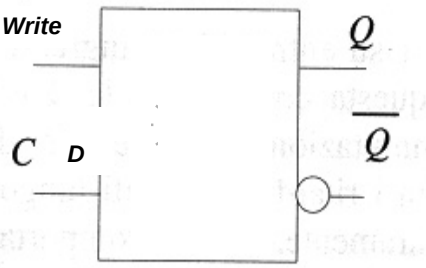
Control (lettura – scrittura)
Select (selezione)
Data in & Sense (Data in & Data out).

Latch sincrono come elemento di memoria

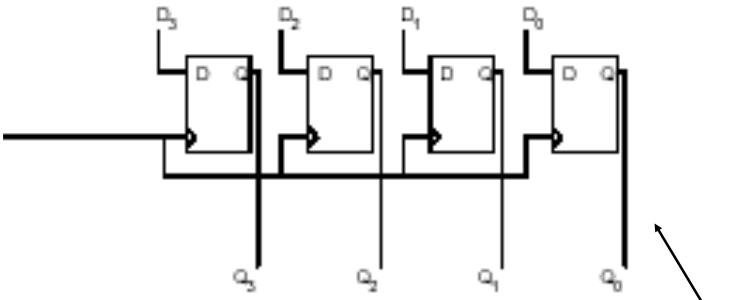
L'ingresso "clock" del bistabile viene utilizzato come segnale di write

E' trasparente quando Write = 1
Se Write = 1 $Q_{t+1} = D$
Se Write = 0 $Q_{t+1} = Q_t$





Registri

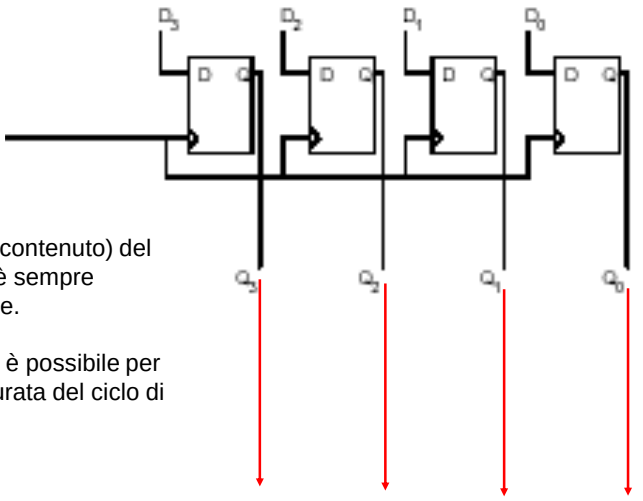


Un registro a 4 bit.
Memorizza 4 bit.

Latch di tipo D

NB Non è un registro a scorrimento (shift register!)

Lettura di un registro

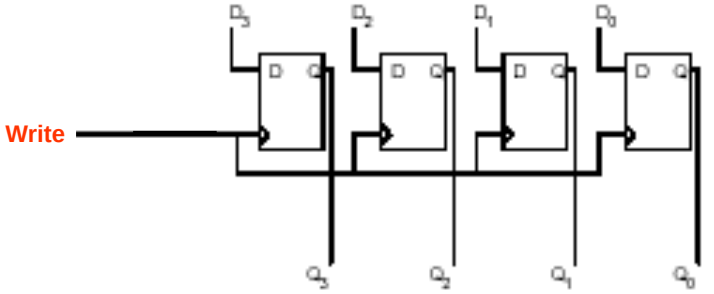


Lo stato (contenuto) del bistabile è sempre disponibile.

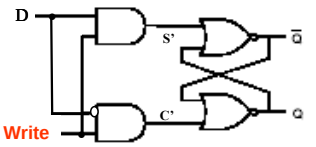
La lettura è possibile per tutta la durata del ciclo di clock.

Scrittura di un registro

Ad ogni colpo di clock lo stato del registro assume il valore dell'ingresso dati.



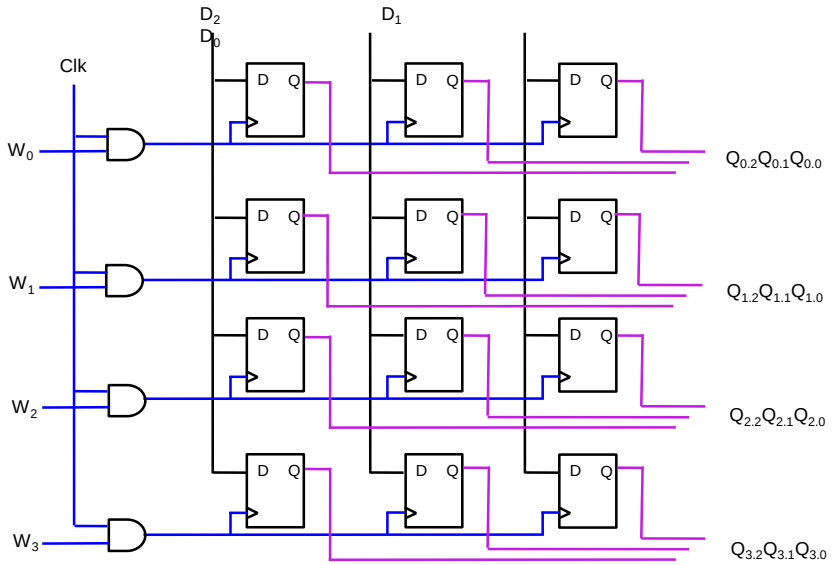
Cosa occorre modificare perchè il registro venga scritto quando serve?
Introdurre una sorta di *"apertura del cancello (chiusura circuito)"* .
Può essere sincronizzata o meno con il clock.



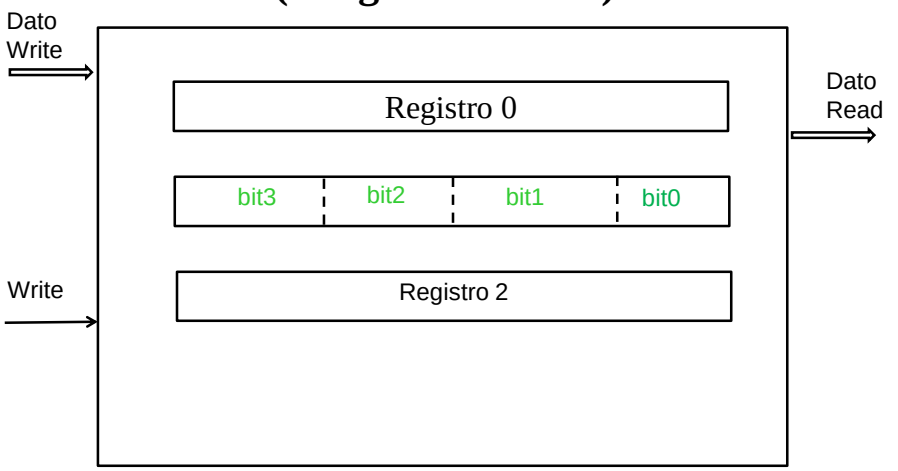
Il clock apre il passaggio al conteuto di D attraverso il latch. Quando il segnale di Write è a zero, lo stato non varia.

Registro

Un banco 4 registri x 3bit

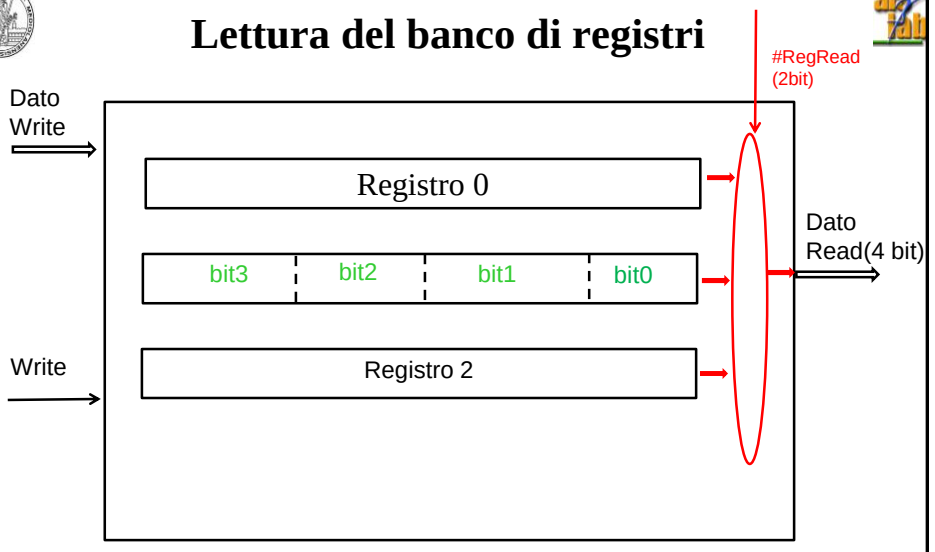


Funzionamento di un banco di registri (3 registri da 4 bit)



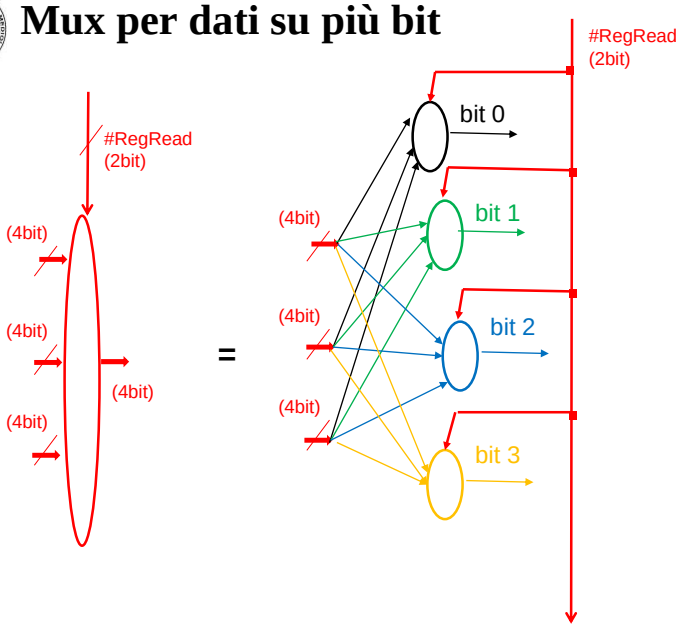
Come seleziono il dato da leggere fra i 3 dati possibili?
Come seleziono il registro su cui scrivere fra i 3 registri?

Lettura del banco di registri

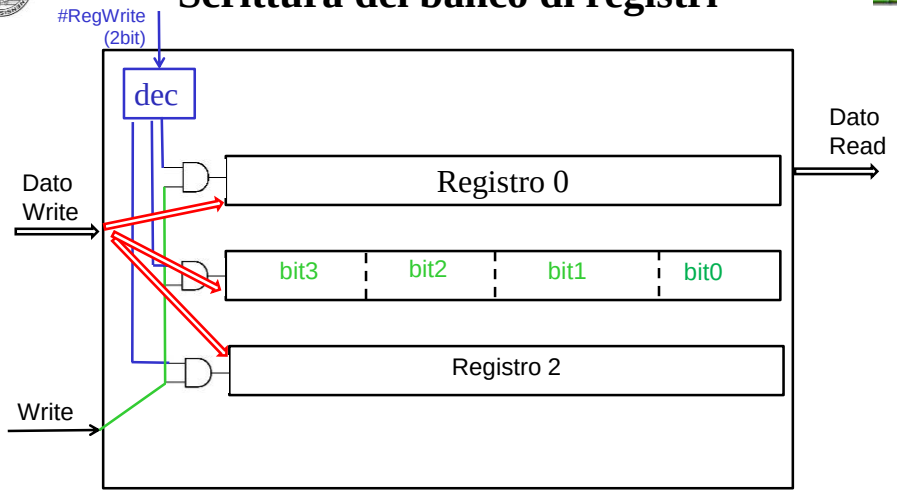


Seleziono uno dei registri = porto in uscita l'uscita Q di tutti i bit del registro selezionato
Avrò tanti Mux quanti sono i bit che costituiscono il registro (in questo caso .....)

Mux per dati su più bit



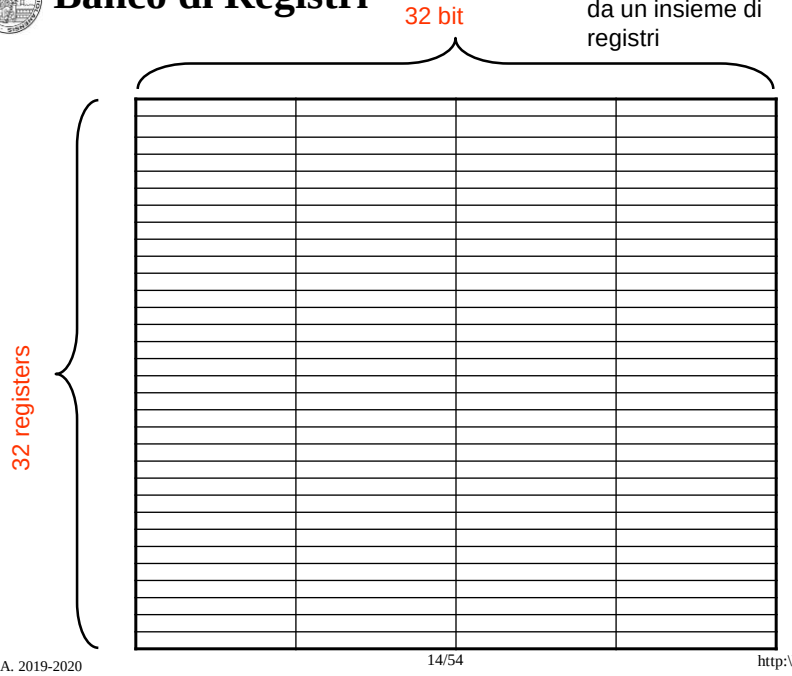
Scrittura del banco di registri



Invio il dato da scrivere a tutti i registri ma devo inviare il segnale di scrittura solamente al registro selezionato

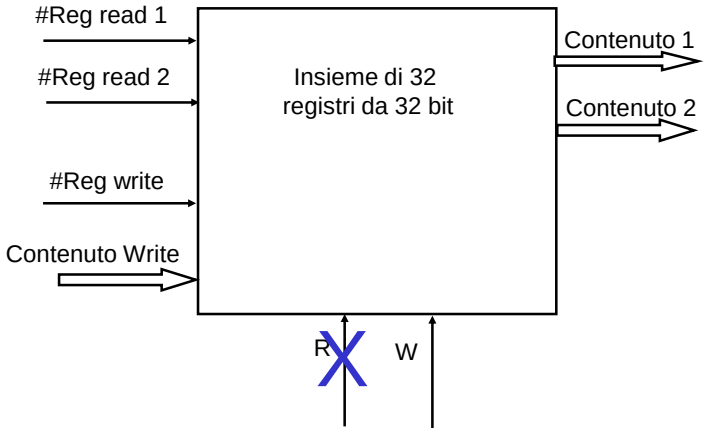
Banco di Registri

Struttura costituita da un insieme di registri



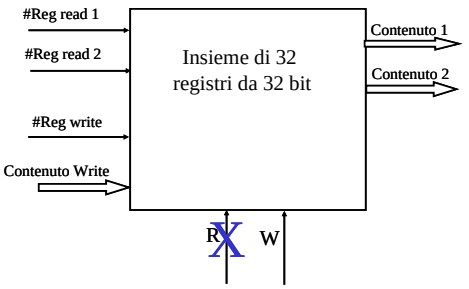
Register file

Banco di registri utilizzabile come memoria



Possono essere letti / scritti fornendo il numero del registro.

Gestione del register file

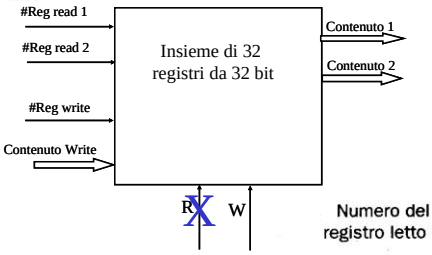


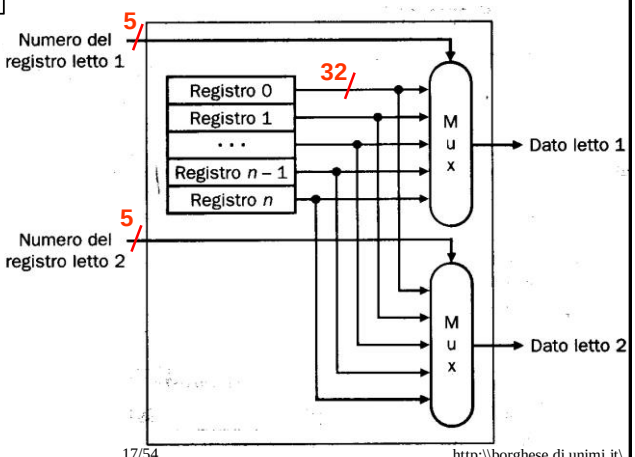
La lettura non modifica il contenuto di un registro (collego uscita Slave con il circuito combinatorio).

La scrittura invece richiede la modifica. Occorre il segnale W.

#bit_indirizzamento = $\log_2$ #bit

Porta di lettura del register file

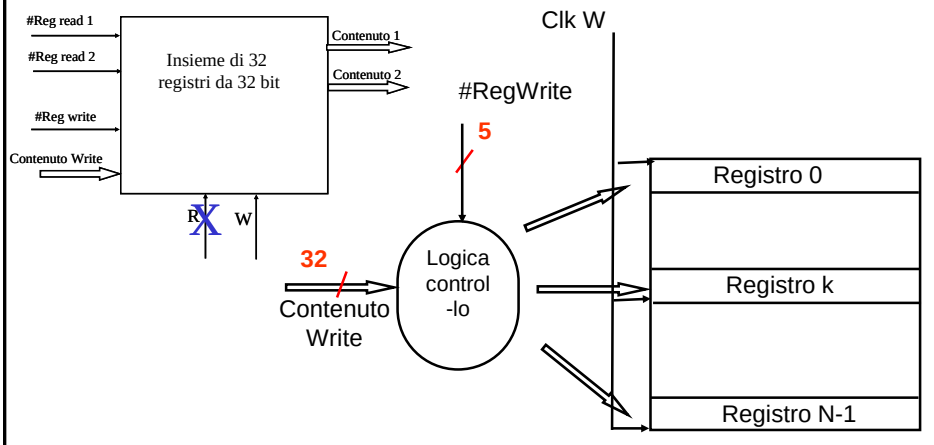




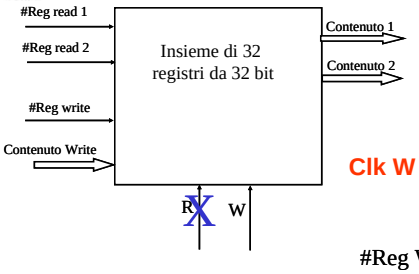
Un mux per ogni porta di lettura.

Ciascun Mux la la complessità di 32 mux, uno per ogni bit.

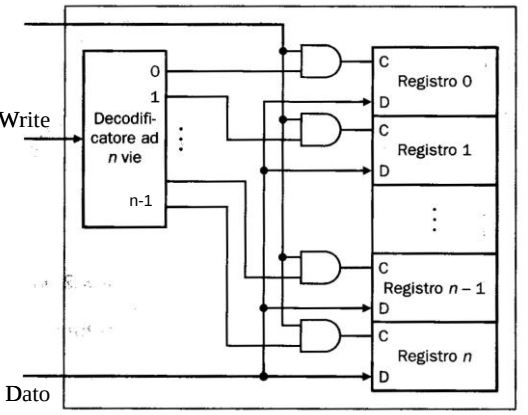
Possibile porta di scrittura del register file



Porta di scrittura del register file



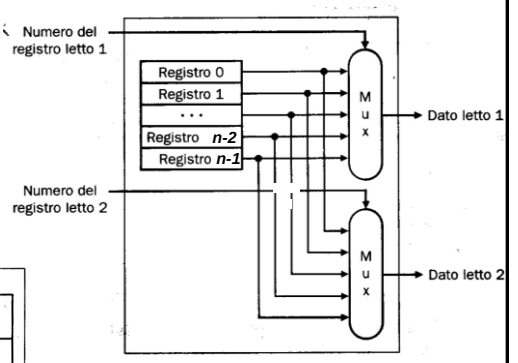
Ho ridotto drasticamente il numero di porte AND.

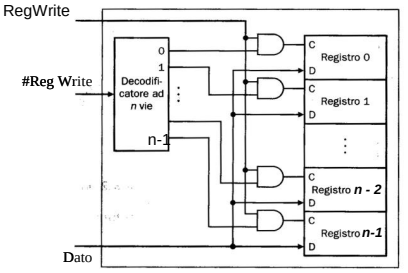


Ingresso C del latch dei registri:
Decodificatore per indirizzare il registro
AND
Comando W

Ingresso D del latch dei registri:
Bit dato corrispondente.

Register file





Sommario

Register File

Costruzione di una CPU per le istruzioni di tipo R

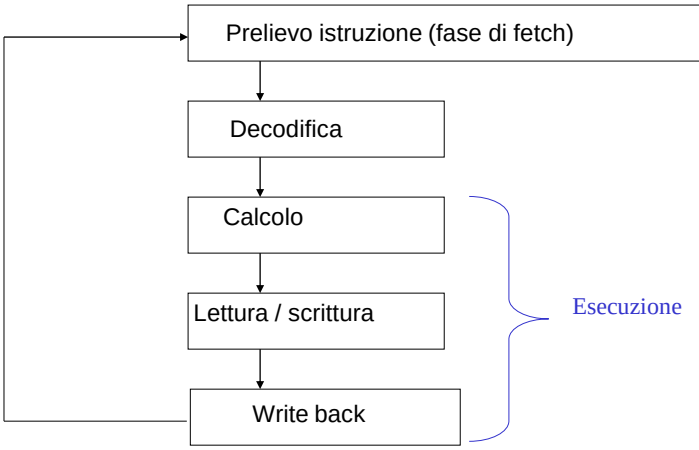
Costruzione di una CPU per le istruzioni di tipo I

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

Ciclo di esecuzione di un'istruzione MIPS



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 multi-ciclo.
- 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

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		
J	op	indirizzo				

Istruzioni

`add $s1, $s2, $s3`	000000	10010	10011	10001	00000	100000	
`beq $s1, $s2, -100`	000100	10001	10010	1111	1111	1110	0111
`lw $t0, 32 ($s3)`	100011	10011	01000	0000	0000	0010	0000
`sw $t0, 32 ($s3)`	101011	10011	01000	0000	0000	0010	0000
`addi $t0, $s3, 64`	001000	10011	01000	0000	0000	0100	0000
`j 0x80000`	000010	00 0000 0100 0000 0000 0000 0000					

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.
- Il segnale di controllo per la lettura (READ) viene inviato alla memoria.
- Trascorso il tempo necessario all'accesso in memoria, la parola indirizzata (in questo caso la prima istruzione del programma) viene letta dalla memoria e trasferita 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 a registri:
    - Se un operando risiede in memoria, deve essere prelevato caricando l'indirizzo dell'operando nel registro MAR della memoria e attivando un ciclo di READ della memoria.
    - L'operando letto dalla memoria viene posto nel registro della memoria MDR per essere trasferito alla ALU, che esegue l'operazione. Nelle architetture MIPS, l'operando viene trasferito nel Register file nella fase di Scrittura.
  - Architetture LOAD/STORE:
    - Le istruzioni di caricamente dalla memoria sono separate da quelle aritmetico/logiche.

Calcolo dell'istruzione (execution - 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

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 Letura / Scrittura, **non** eseguo operazioni aritmetico logiche sui dati.

Sistema di memoria "sganciato" dalla coppia register-file + CPU.

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?

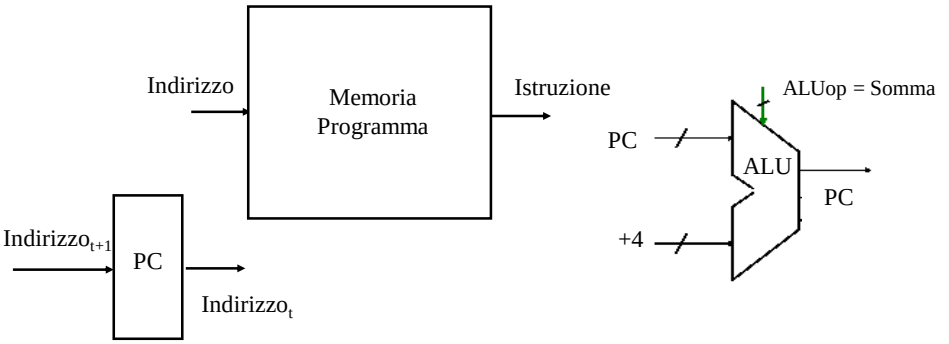
- Usa un registro, il Program Counter (PC) per ottenere l'indirizzo dell'istruzione.
- Preleva l'istruzione dalla memoria e la inserisce nell'IR.
- Capisce di che tipo di istruzione si tratta (decodifica).
  - usa l'istruzione stessa per decidere cosa fare esattamente.
- Legge il contenuto dei registri.

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.
- Scrittura del risultato nel register file.

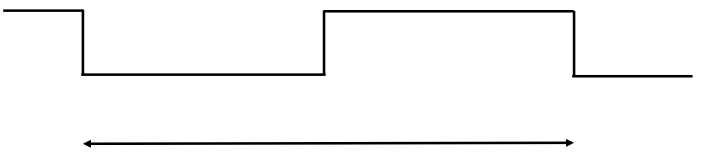
Fase di fetch

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.



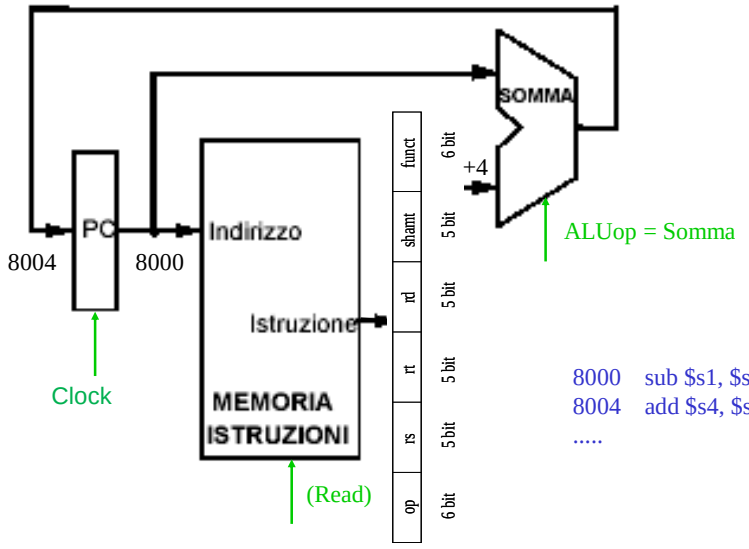
Temporizzazione

1 istruzione per ciclo di clock. Temporizzazione del PC.



T > Tempo necessario per eseguire il cammino critico

Circuito della fase di fetch



Istruzioni di tipo R

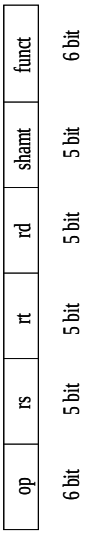
R

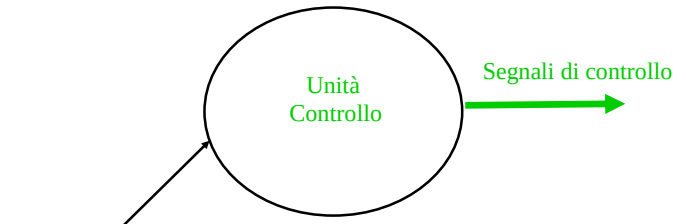
op = 0	rs = 18	rt = 19	rd = 17	Shamt=0	funct=34
6 bit	5 bit	5 bit	5 bit	5 bit	6 bit

sub $s1, $s2, $s3

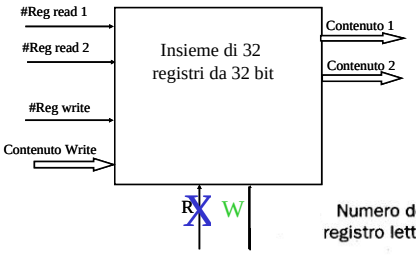
Fase di decodifica

1) Leggo l'istruzione e genero i segnali di controllo opportuni.
2) Leggo il contenuto dei registri.



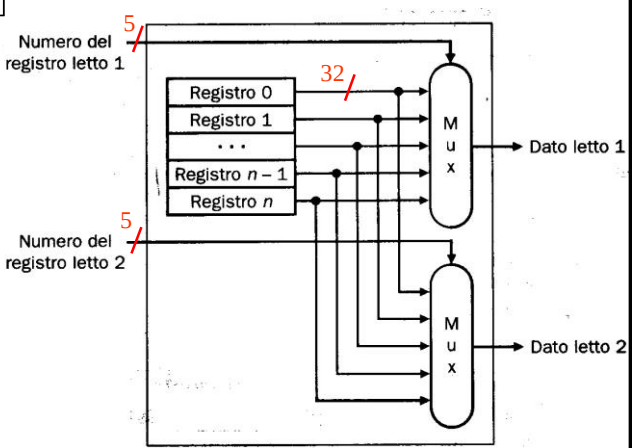


Register file



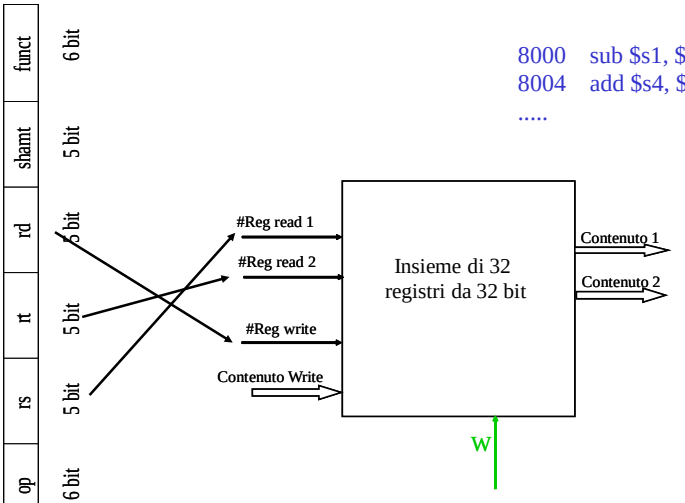
Banco di registri utilizzabile come memoria
Può essere scritto o letto.

Un mux per ogni porta di lettura.



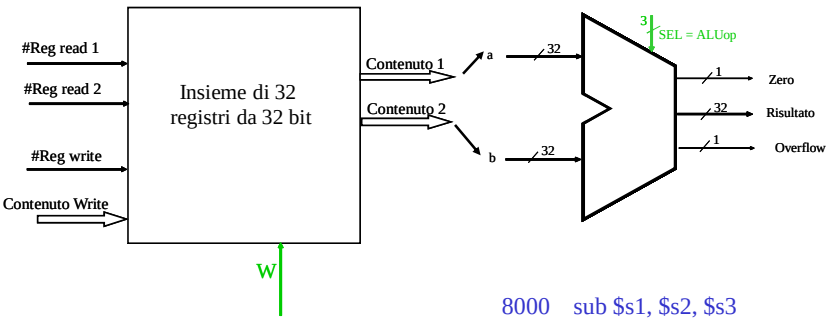
Lettura dei registri (istruzioni di tipo R)

1) Leggo l'istruzione e genero i segnali di controllo opportuni.
2) Leggo il contenuto dei registri.



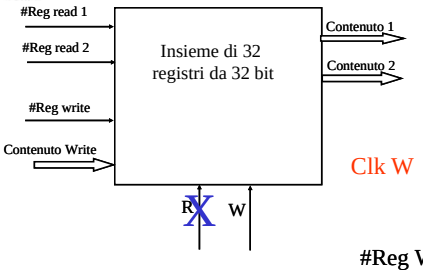
8000 sub $s1, $s2, $s3
8004 add $s4, $s1, $t1
.....

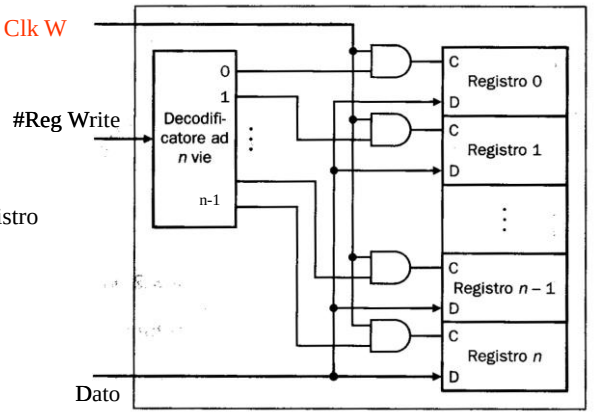
Fase di Calcolo (tipo R)



8000 sub $s1, $s2, $s3
8004 add $s4, $s1, $t1
.....

Porta di scrittura del register file





Ingresso C del latch dei registri:
Decodificatore per indirizzare il registro
AND
Comando Write

Ingresso D del latch dei registri:
Bit dato corrispondente.

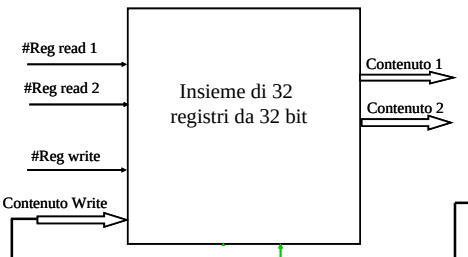
NB Utilizzo registri flip-flop in modo da potere leggere / scrivere nello stesso ciclo di clock (scrivo nel master nella fase di WB e leggo dallo slave in fase di decodifica. La commutazione da master a slave è pilotata dal clock.

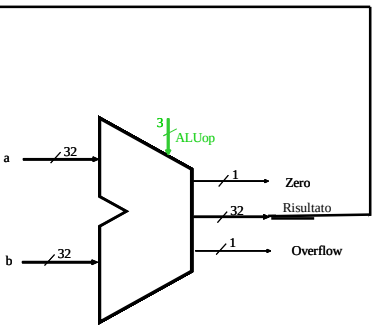
A

Fase di Write back (tipo R)

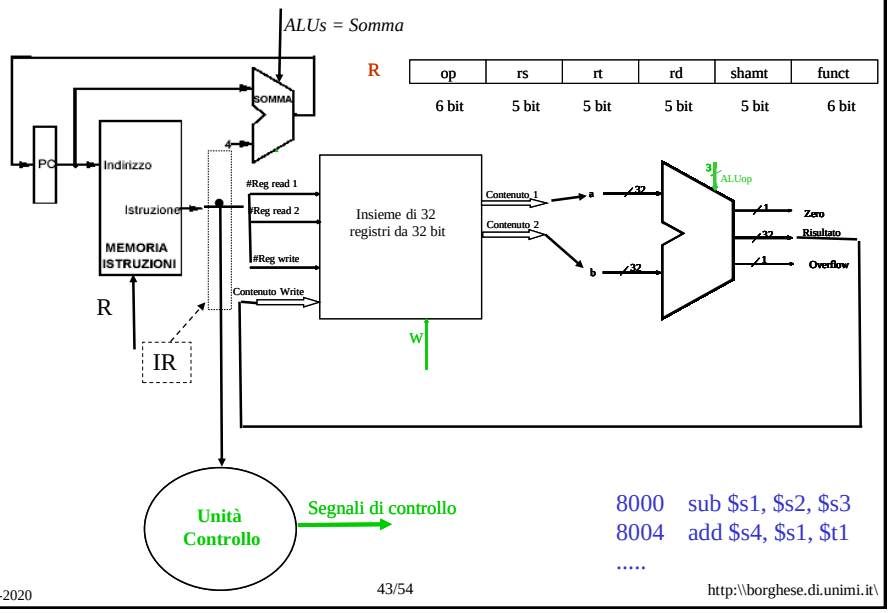
8000 sub $s1, $s2, $s3
8004 add $s4, $s1, $t1
.....

Risultato





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



R

op	rs	rt	rd	shamt	funct
6 bit	5 bit	5 bit	5 bit	5 bit	6 bit

8000 sub $s1, $s2, $s3
8004 add $s4, $s1, $t1
.....

Sommario

Register File

Costruzione di una CPU per le istruzioni di tipo R

Costruzione di una CPU per le istruzioni di tipo I

Istruzioni di tipo I: lw/sw

I

100011	10001	10010	0000 0000 0001 0100
6 bit	5 bit	5 bit	16 bit

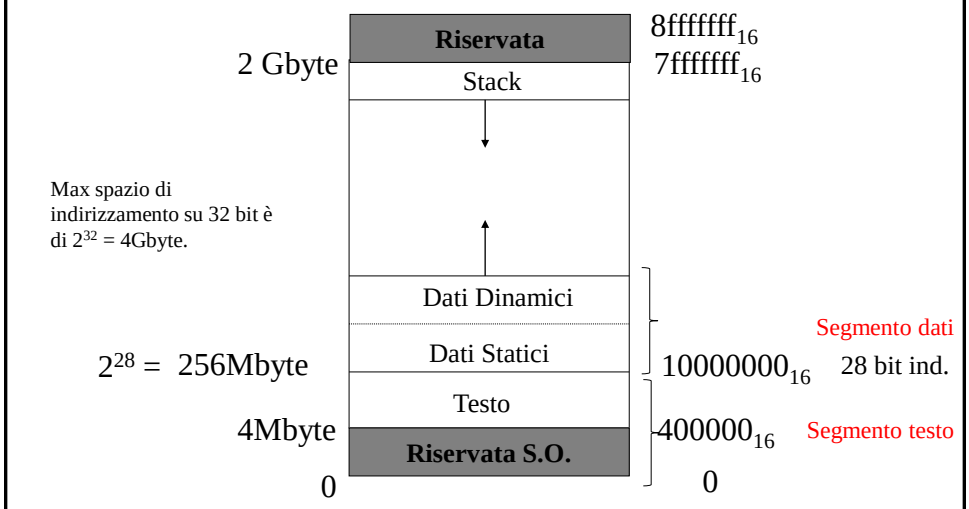
lw $s2, 20($s1)

L'indirizzo di memoria sarà:

Base [$s2] 0100 1000 0011 0001 1011 1011 1011 1011 +
Offset 0000 0000 0001 0100 +

Indirizzo dato 0100 1000 0011 0001 1011 1011 1100 1111

Organizzazione logica della memoria

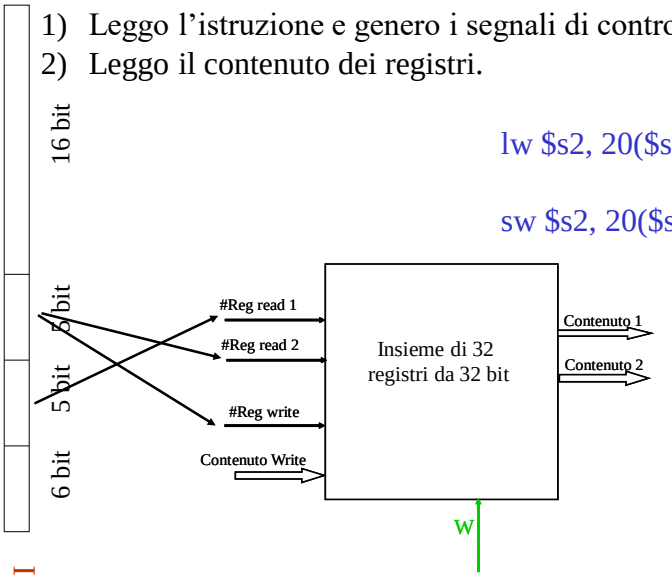


Lettura dei registri (istruzioni di tipo I)

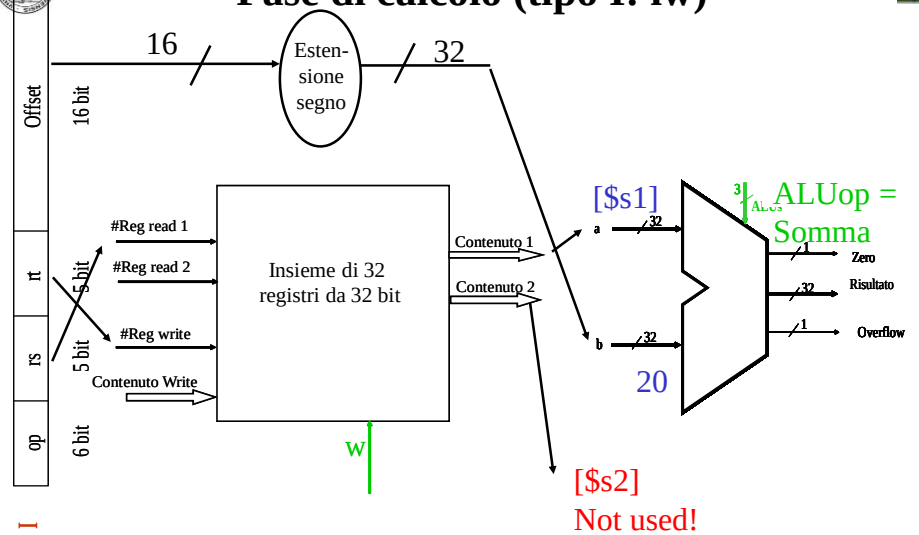
1) Leggo l'istruzione e genero i segnali di controllo opportuni.
2) Leggo il contenuto dei registri.

lw $s2, 20($s1)

sw $s2, 20($s1)



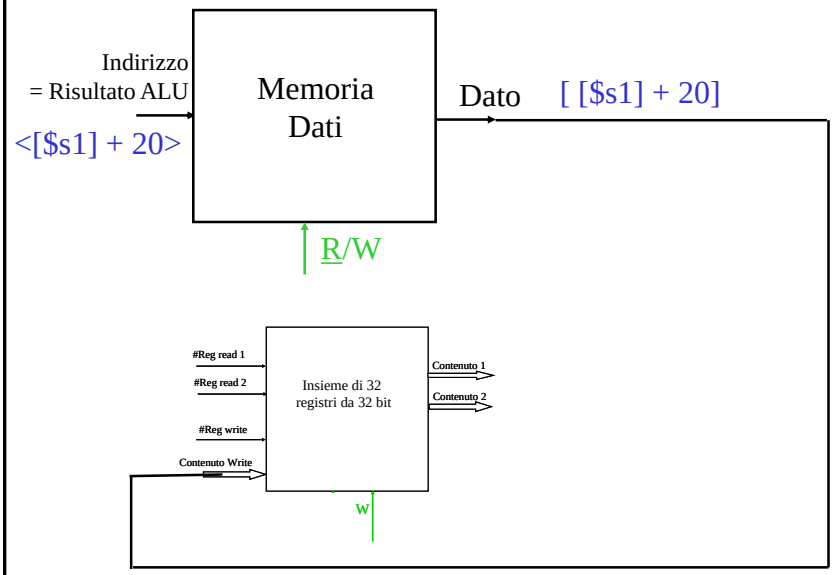
Fase di calcolo (tipo I: lw)



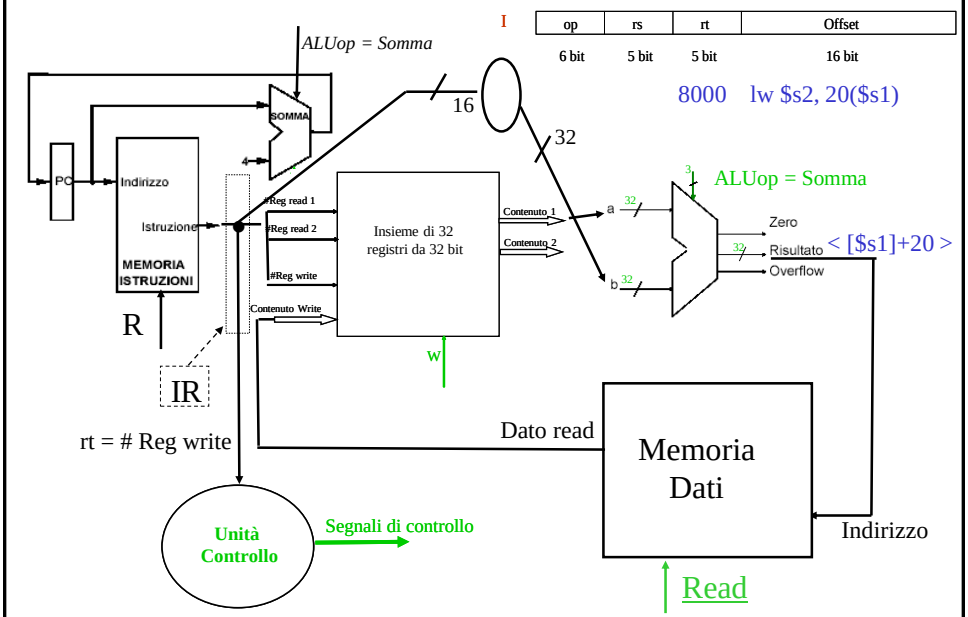
Il Risultato è un indirizzo della memoria

8000 lw $s2, 20($s1)

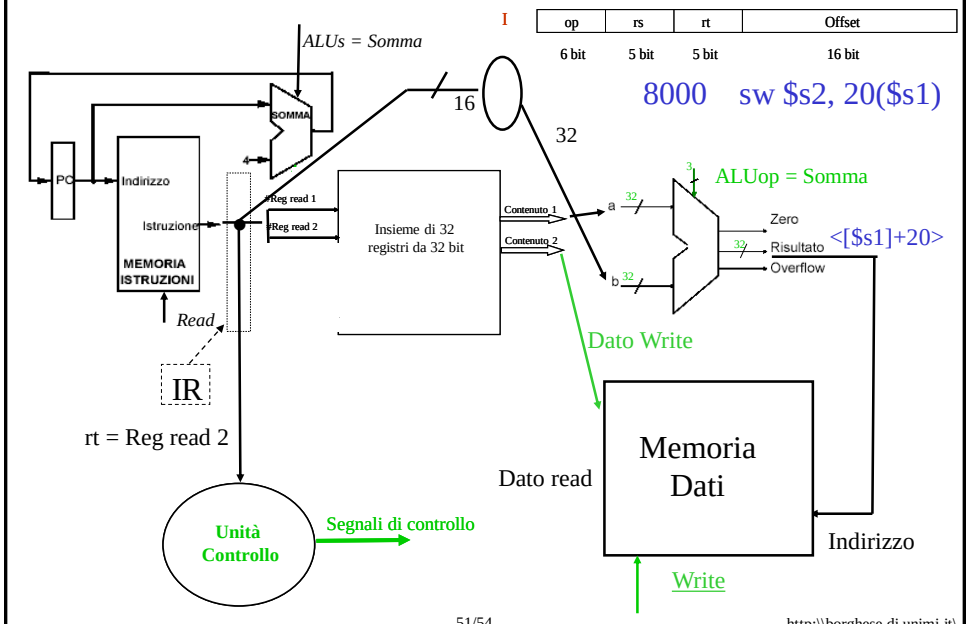
Lettura della memoria + write-back



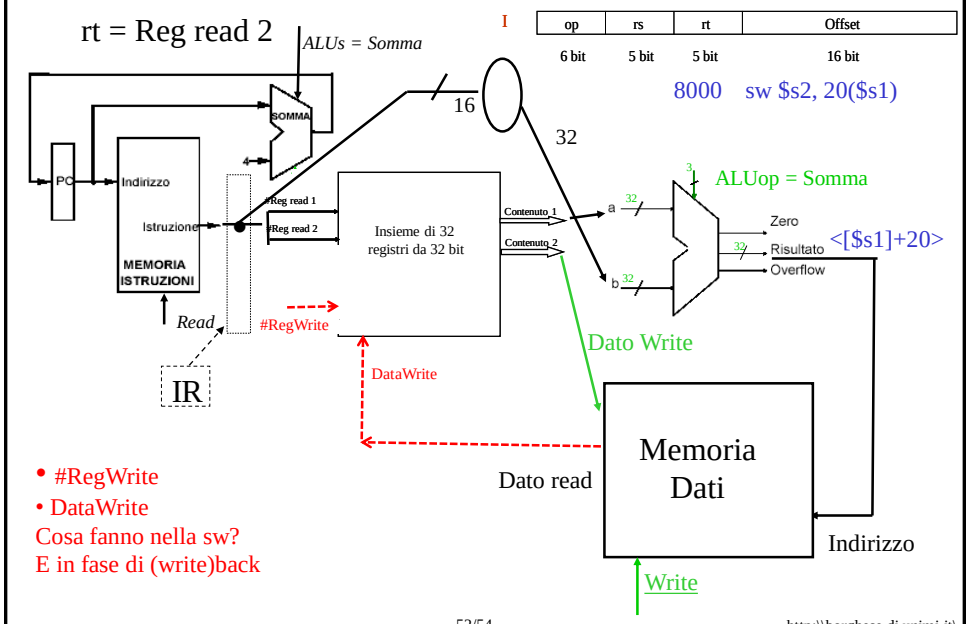
CPU per l'esecuzione di una lw



CPU per l'esecuzione di una sw



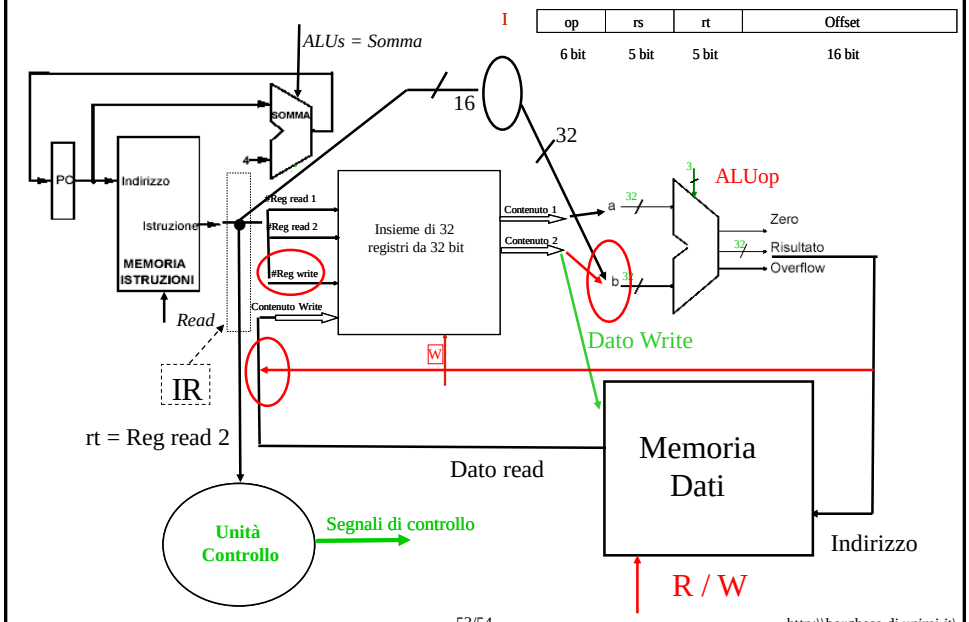
CPU per l'esecuzione di una sw



- #RegWrite
- DataWrite

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

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



Sommario

Register File

Costruzione di una CPU per le istruzioni di tipo R

Costruzione di una CPU per le istruzioni di tipo I