


Architettura degli elaboratori
CPU a ciclo singolo

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
borghese@di.unimi.it

Università degli Studi di Milano

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


Sommario

Costruzione di una CPU per le istruzioni di tipo R

Costruzione di una CPU per le istruzioni di tipo I (memoria).

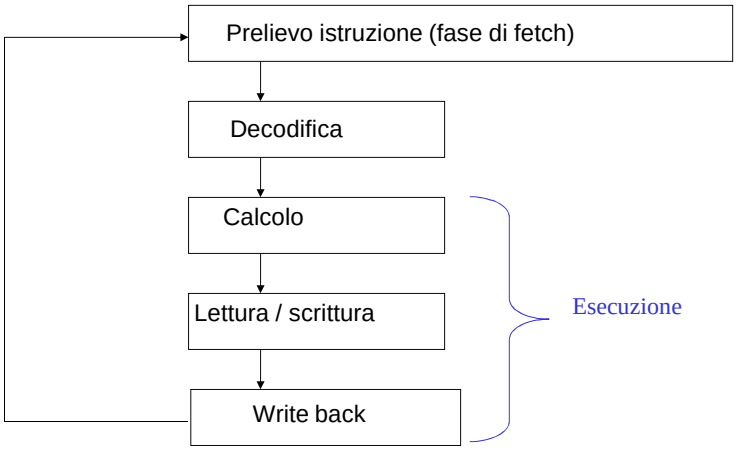
Costruzione di una CPU per le istruzioni di tipo I (branch).

Obbiettivo

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

- Accesso alla memoria in lettura (lw) o scrittura (sw).
- Istruzioni logico-matematiche (e.g. add, sub, and....).
- Istruzioni di salto condizionato (branch) o incondizionato (jump).

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		
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					

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

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.

Come funziona una CPU?

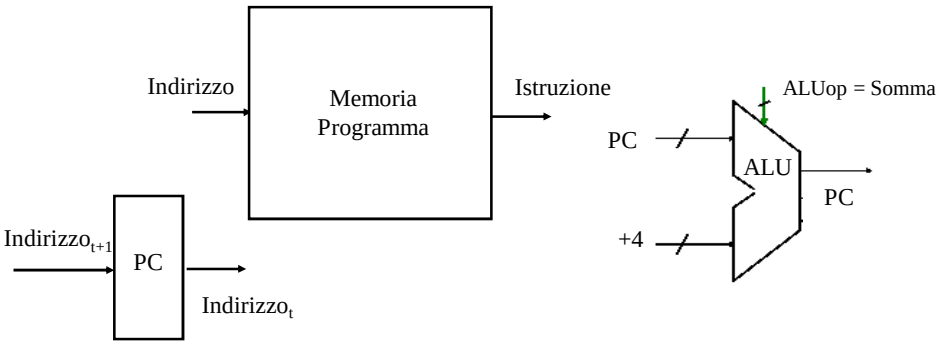
- Usa un registro, il Program Counter (PC) per ottenere l'indirizzo dell'istruzione.
- Preleva l'istruzione dalla memoria e la inserisce nell'IR.
- Capisce di che tipo di istruzione si tratta (decodifica).
  - usa l'istruzione stessa per decidere cosa fare esattamente.
- Legge il contenuto dei registri.

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.

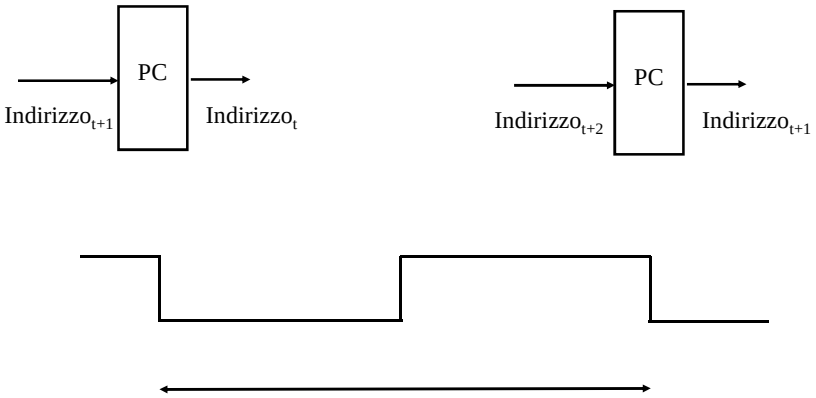
Fase di fetch

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.



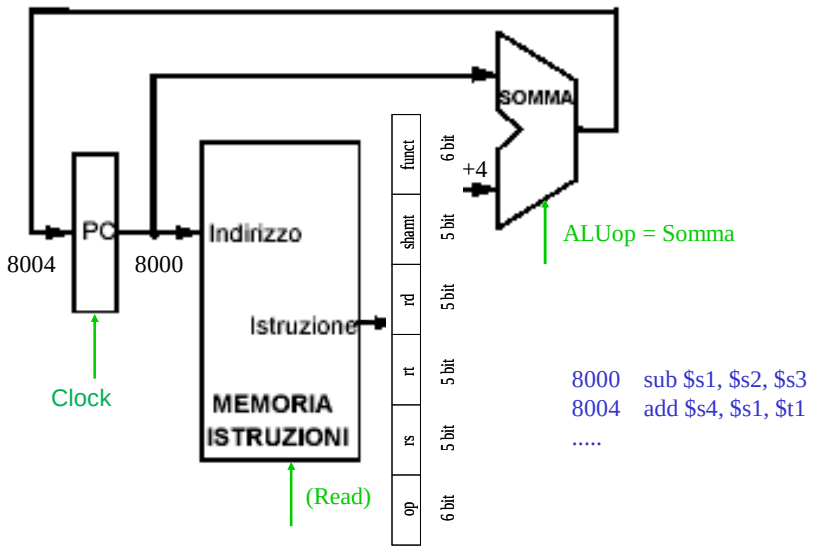
Temporizzazione

1 istruzione per ciclo di clock. Temporizzazione del PC.



T > Tempo necessario per eseguire il cammino critico

Circuito della fase di fetch



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

Istruzioni di tipo R

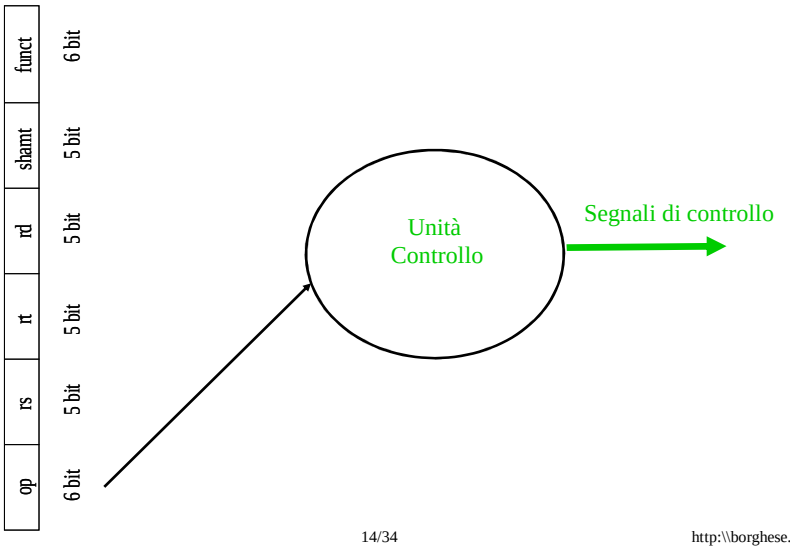
R

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

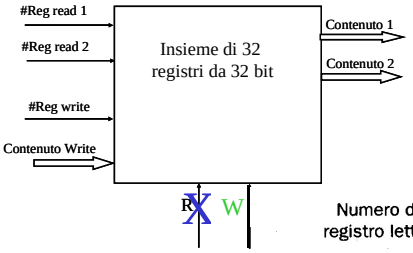
sub $s1, $s2, $s3

Fase di decodifica

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

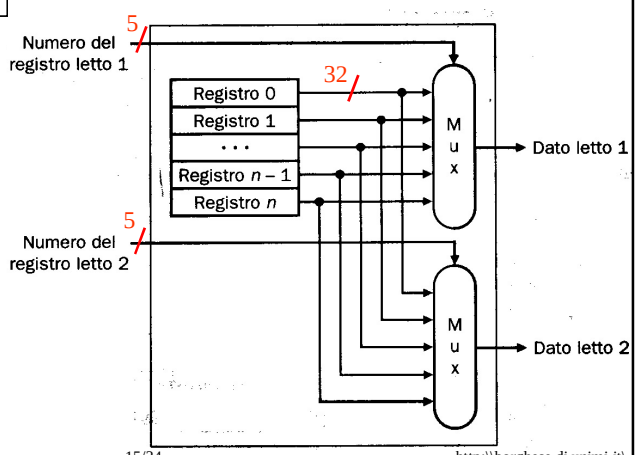


Register file



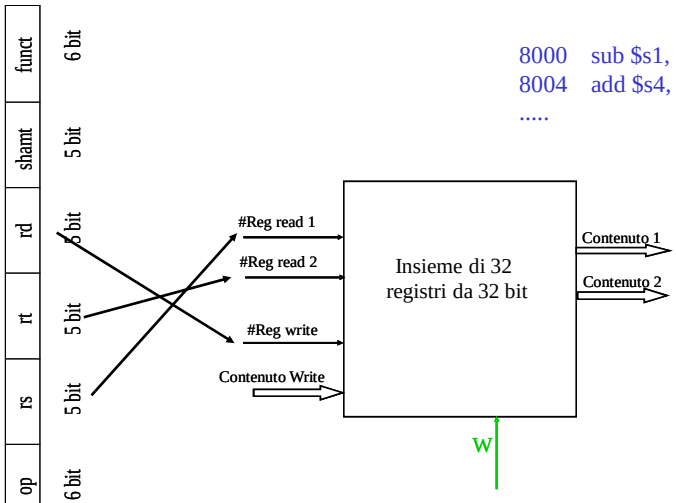
Banco di registri utilizzabile come memoria
Può essere scritto o letto.

Un mux per ogni porta di lettura.



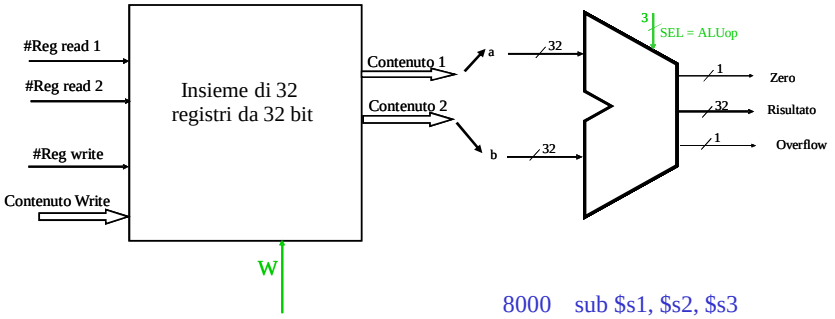
Lettura dei registri (istruzioni di tipo R)

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



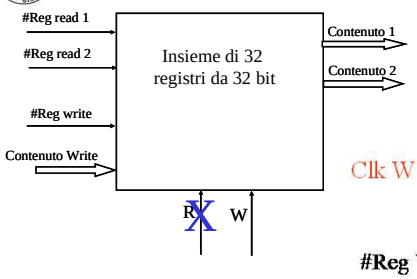

```
8000    sub $s1, $s2, $s3
8004    add $s4, $s1, $t1
.....
```

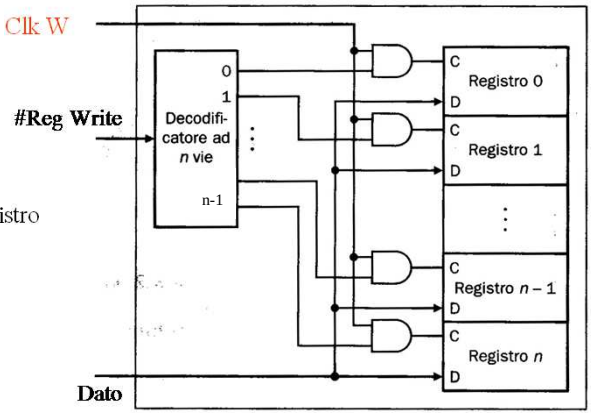
Fase di Calcolo (tipo R)




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


Porta di scrittura del register file





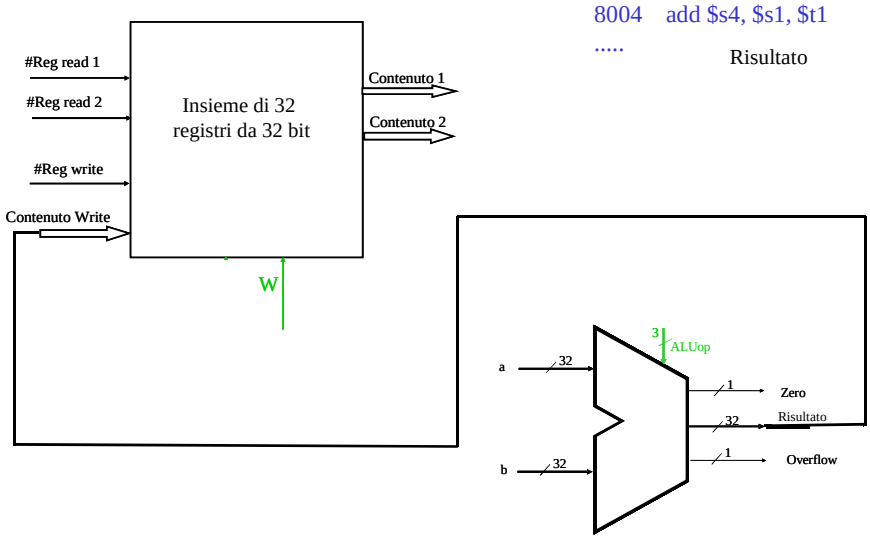
Ingresso C del latch dei registri:
Decodificatore per indirizzare il registro
AND
Comando Write

Ingresso D del latch dei registri:
Bit dato corrispondente.

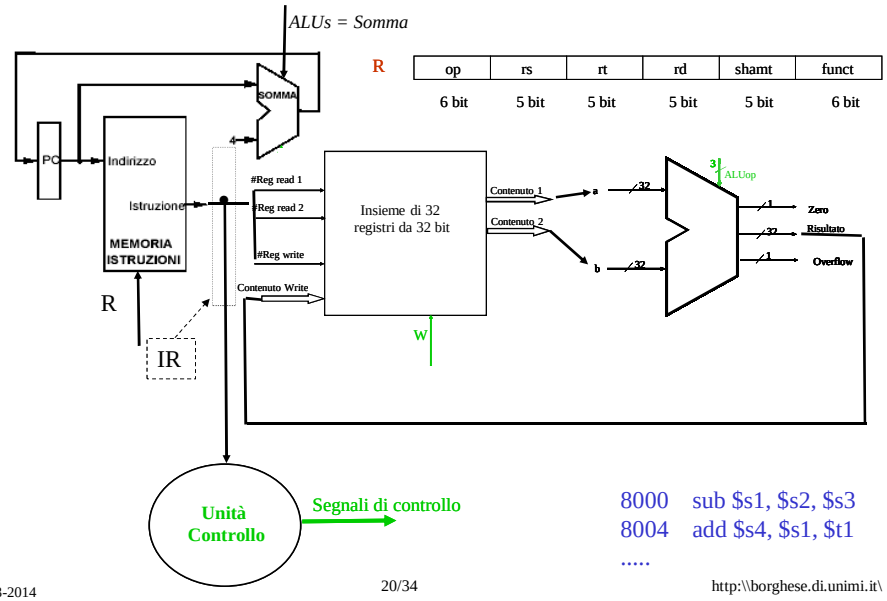
NB Utilizzo registri flip-flop in modo da potere leggere / scrivere nello stesso ciclo di clock (scrivo nel master nella fase di WB e leggo dallo slave in fase di decodifica. La commutazione da master a slave è pilotata dal clock.

Fase di Write back (tipo R)

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



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



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

Sommario

Costruzione di una CPU per le istruzioni di tipo R

Costruzione di una CPU per le istruzioni di tipo I (memoria).

Costruzione di una CPU per le istruzioni di tipo I (branch).

Istruzioni di tipo I: lw/sw

I

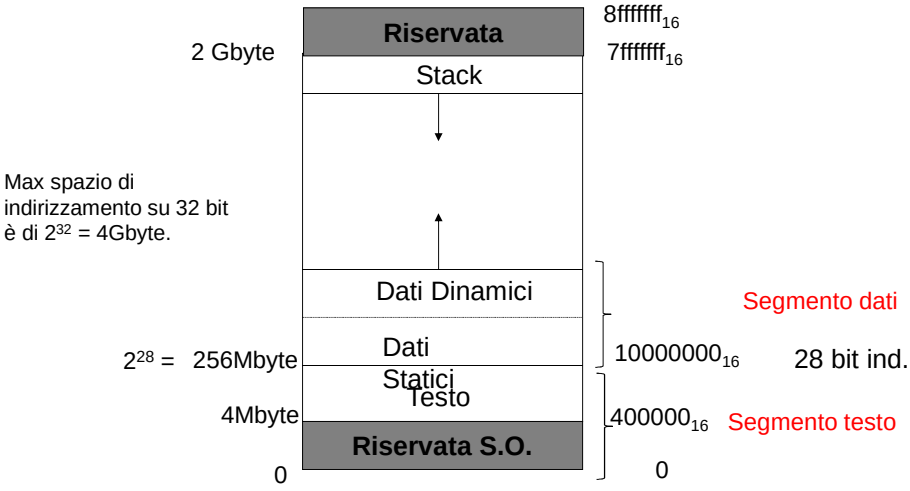
100011	10001	10010	0000 0000 0001 0100
6 bit	5 bit	5 bit	16 bit

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

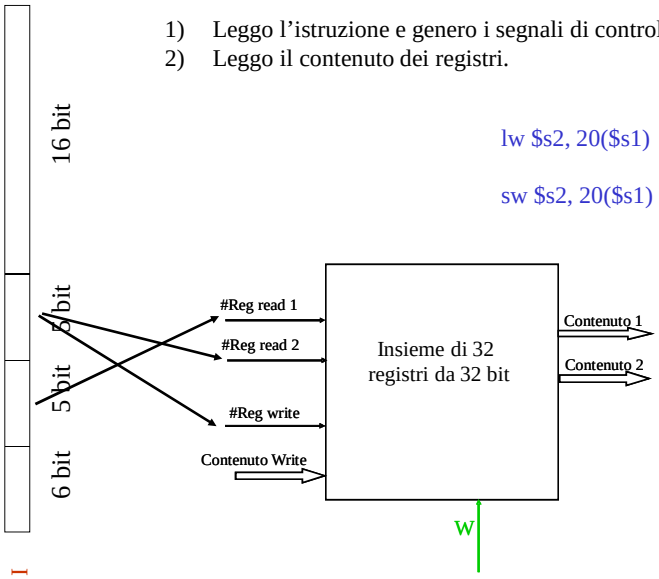


Lettura dei registri (istruzioni di tipo I)

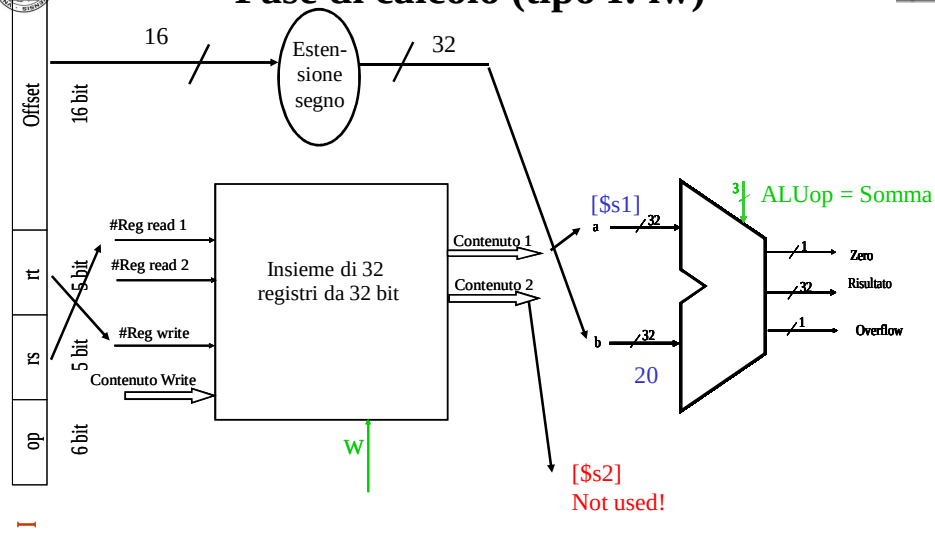
1) Leggo l'istruzione e genero i segnali di controllo opportuni.
2) Leggo il contenuto dei registri.

lw $s2, 20($s1)

sw $s2, 20($s1)



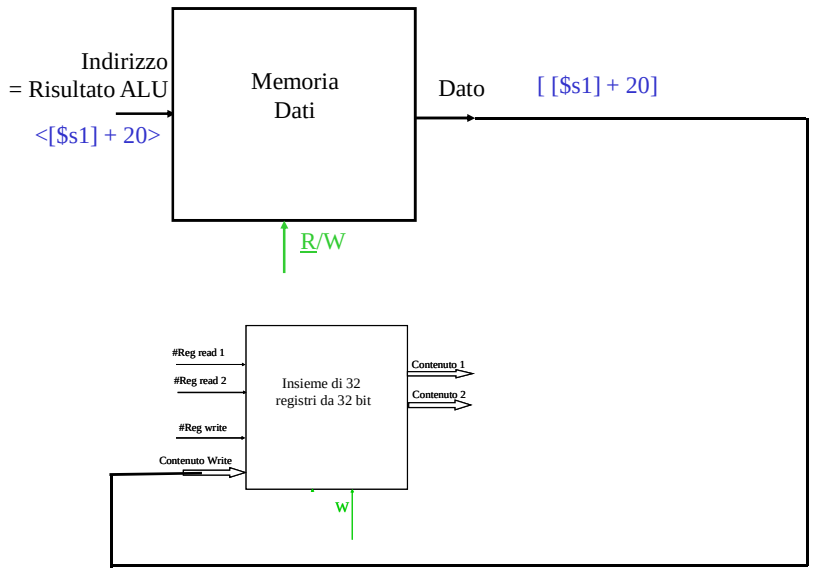
Fase di calcolo (tipo I: lw)



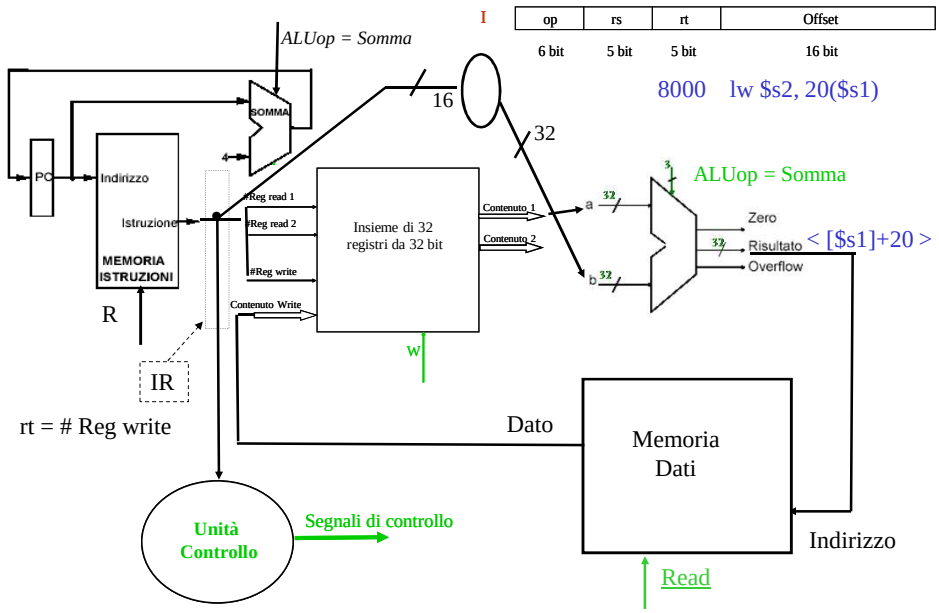
Il Risultato è un indirizzo della memoria

8000 lw $s2, 20($s1)

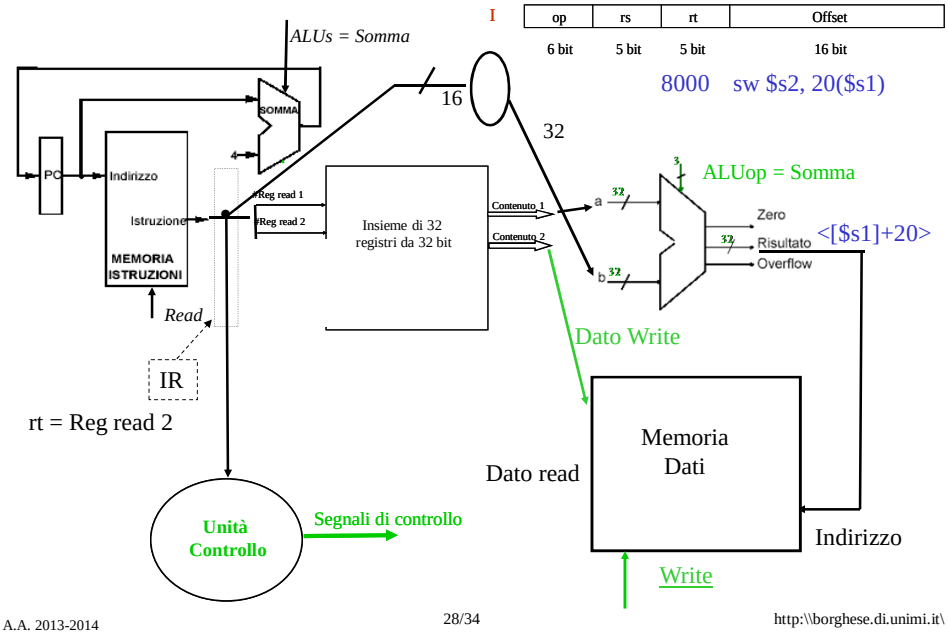
Lettura della memoria + write-back



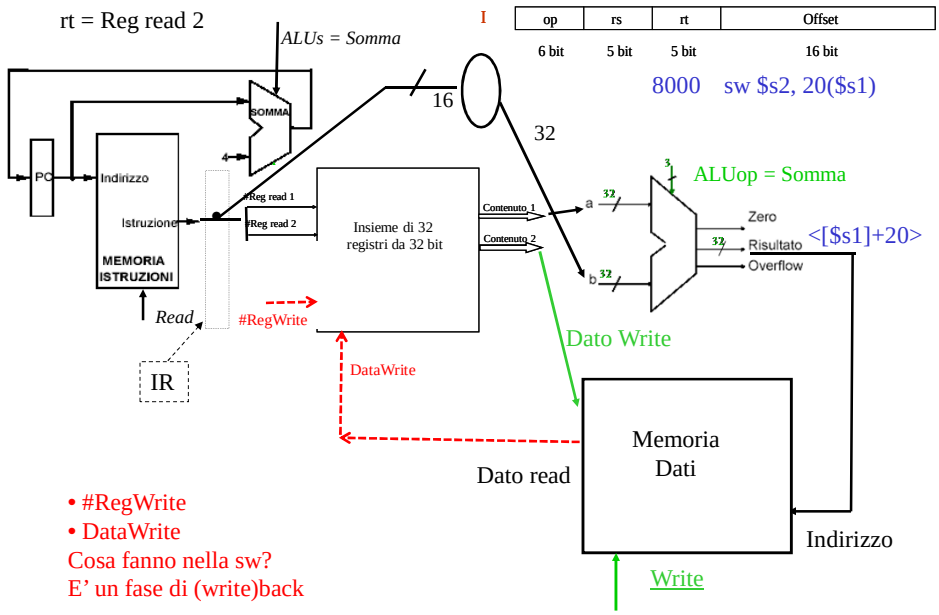
CPU per l'esecuzione di una lw



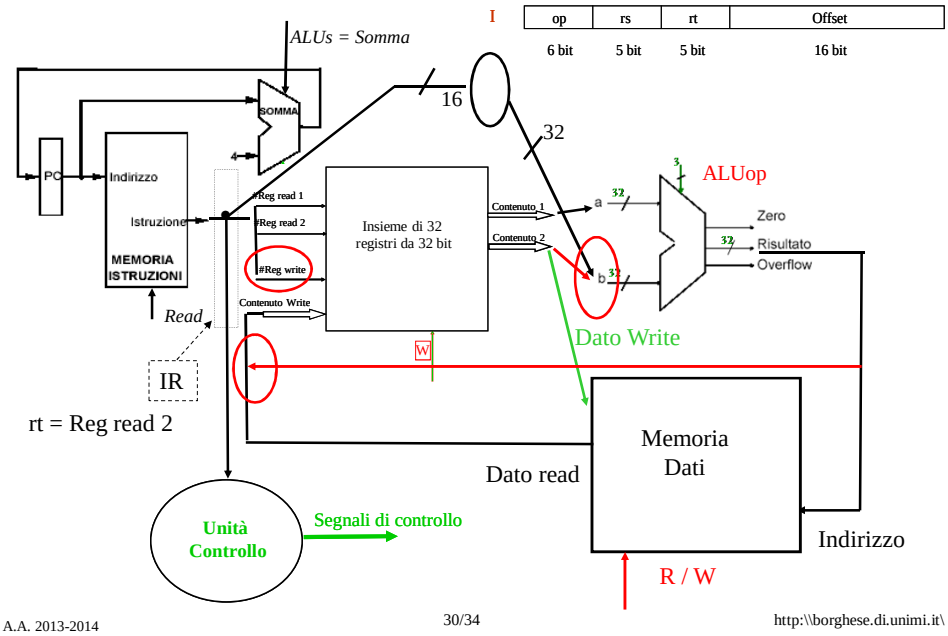
CPU per l'esecuzione di una sw



CPU per l'esecuzione di una sw



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



Sommario

Costruzione di una CPU per le istruzioni di tipo R

Costruzione di una CPU per le istruzioni di tipo I (memoria).

Costruzione di una CPU per le istruzioni di tipo I (branch).

Tipi di istruzioni dell'ISA

Consideriamo istruzioni di tipo R, di tipo lw/sw, salto condizionato:

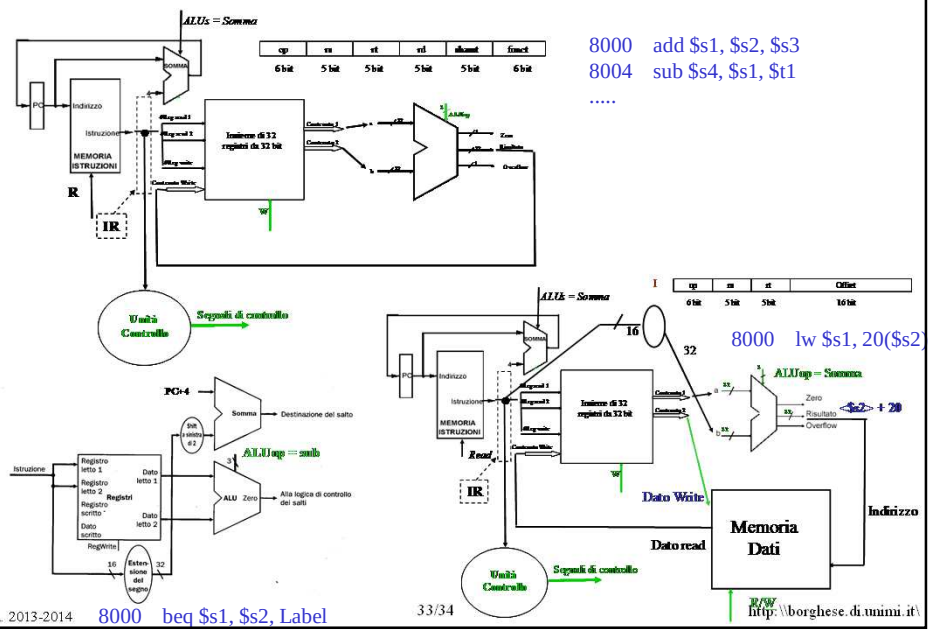
	31-26	25-21	20-16	15-11	10-6	5-0
R	op	rs	rt	rd	shamt	funct

	31-26	25-21	20-16	15-0
lw/sw	35/43	rs	rt	offset

	31-26	25-21	20-16	15-0
beq	4	rs	rt	indirizzo

Architettura RISC:
Campo Op sempre contenuto nei primi 6 bit, inseguito Op[5-0].
Registri da leggere, rs ed rt, in posizione 25-21 e 20-16. Vengono sempre letti dal Register File.
Registro base per lettura / scrittura, rs, sempre in posizione 25-21.
Offset sempre in posizione 15-0.
Registro destinazione, rd (15-11) per le istruzioni di tipo R, rt (20-16) per le istruzioni lw.

CPU per l'esecuzione di istruzioni di tipo R/lw/beq



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

8000 lw $s1, 20($s2)

8000 beq $s1, $s2, Label

Sommario

Costruzione di una CPU per le istruzioni di tipo R

Costruzione di una CPU per le istruzioni di tipo I (memoria).

Costruzione di una CPU per le istruzioni di tipo I (salti).