



# Architettura degli elaboratori CPU a ciclo singolo

Prof. Alberto Borghese  
Dipartimento di Scienze dell'Informazione  
[borgese@di.unimi.it](mailto:borgese@di.unimi.it)

Università degli Studi di Milano

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



## Sommario

**Il register file e le sue porte di lettura e scrittura**

Costruzione di una CPU per le istruzioni di tipo R



## 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*), sommatore ausiliari, ecc.;
- Unità di controllo**. Controlla il flusso e determina le operazioni di ciascun blocco.

**MEMORIA PRINCIPALE**

A.A. 2014-2015

3/45

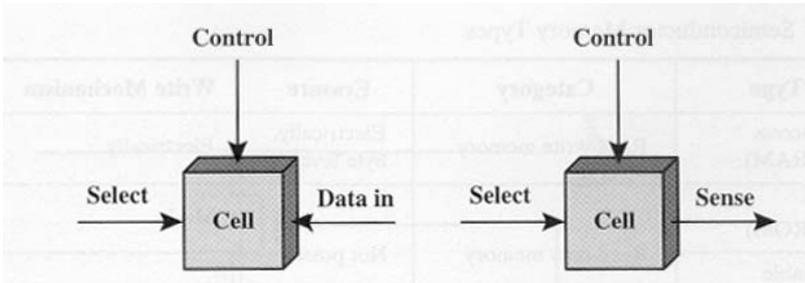
<http://borghese.di.unimi.it/>



## 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).

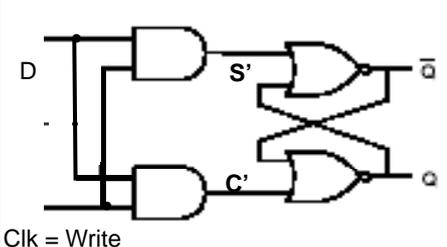
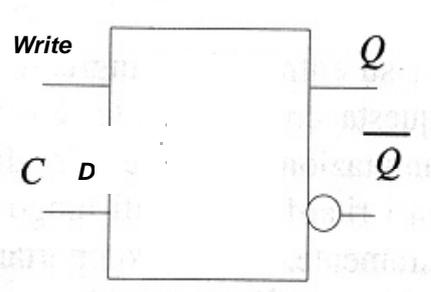
A.A. 2014-2015

4/45

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

**Latch sincrono come elemento di memoria**

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

A.A. 2014-2015 5/45 <http://borgese.di.unimi.it/>

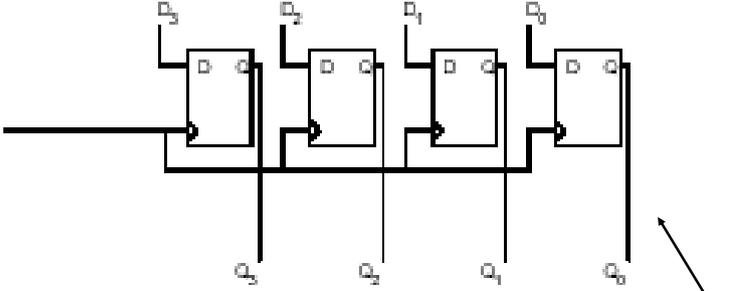
**Registro**



A.A. 2014-2015 6/45 <http://borgese.di.unimi.it/>




## Registri



Un registro a 4 bit.  
Memorizza 4 bit.

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

Latch di tipo D

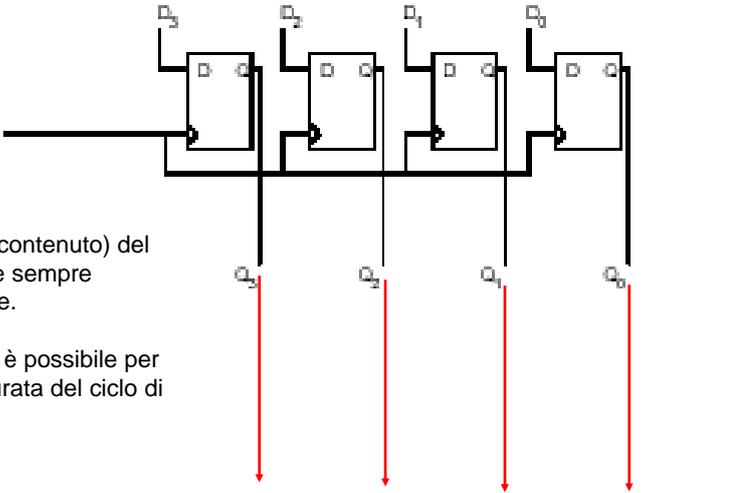
A.A. 2014-2015

7/45

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




## Lettura di un registro



Lo stato (contenuto) del bistabile è sempre disponibile.

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

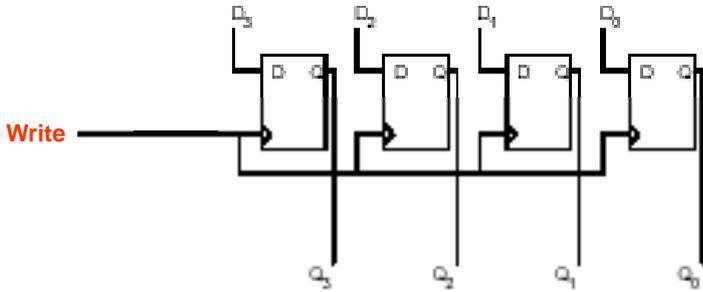
A.A. 2014-2015

8/45

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

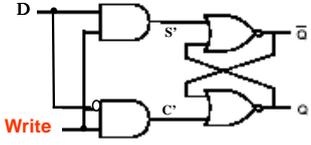
### Scrittura di un registro

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



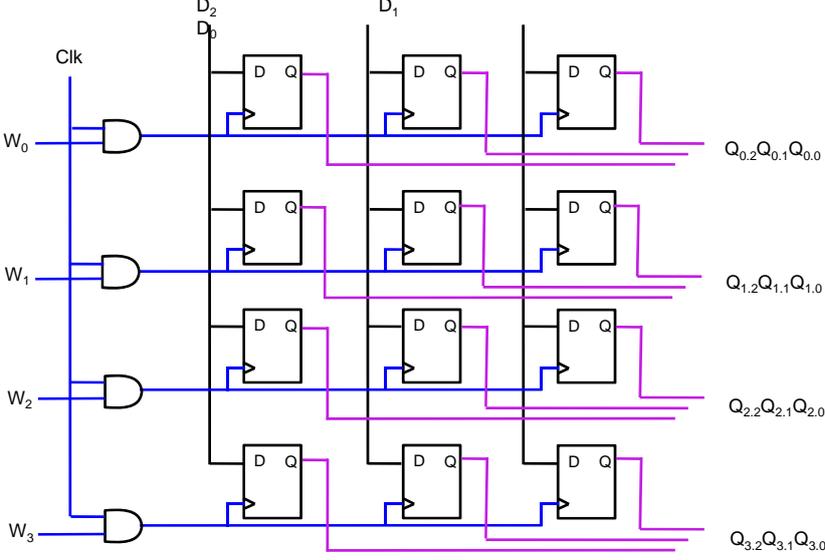
Cosa occorre modificare perchè il registro venga scritto quando serve?  
 Introdurre una sorta di *"apertura del cancello (chiusura circuito)"*.  
 Può essere sincronizzata o meno con il clock.

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

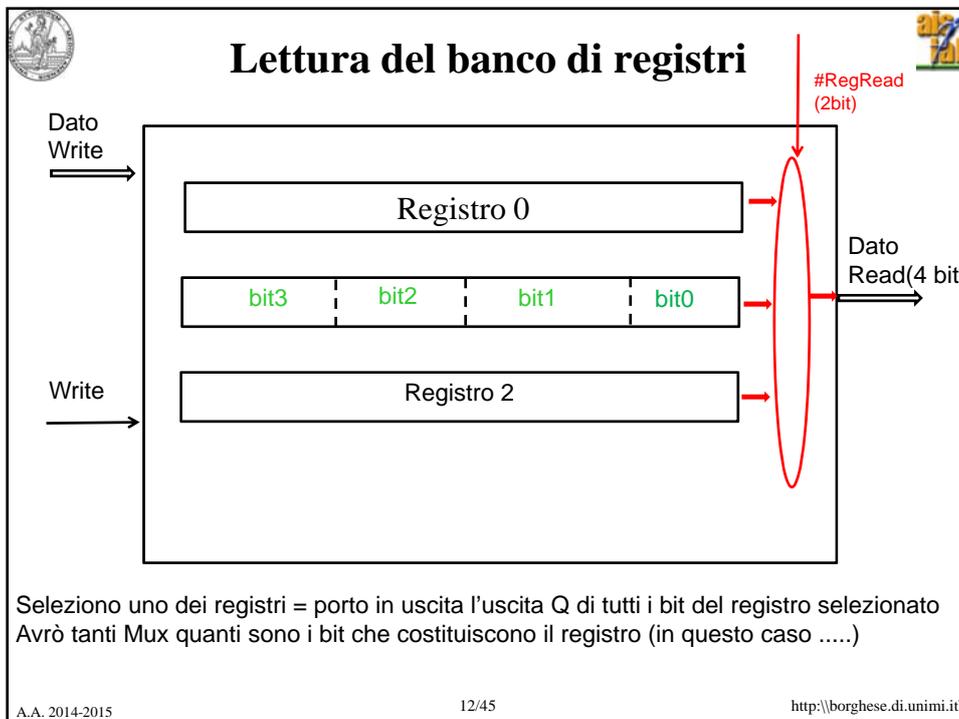
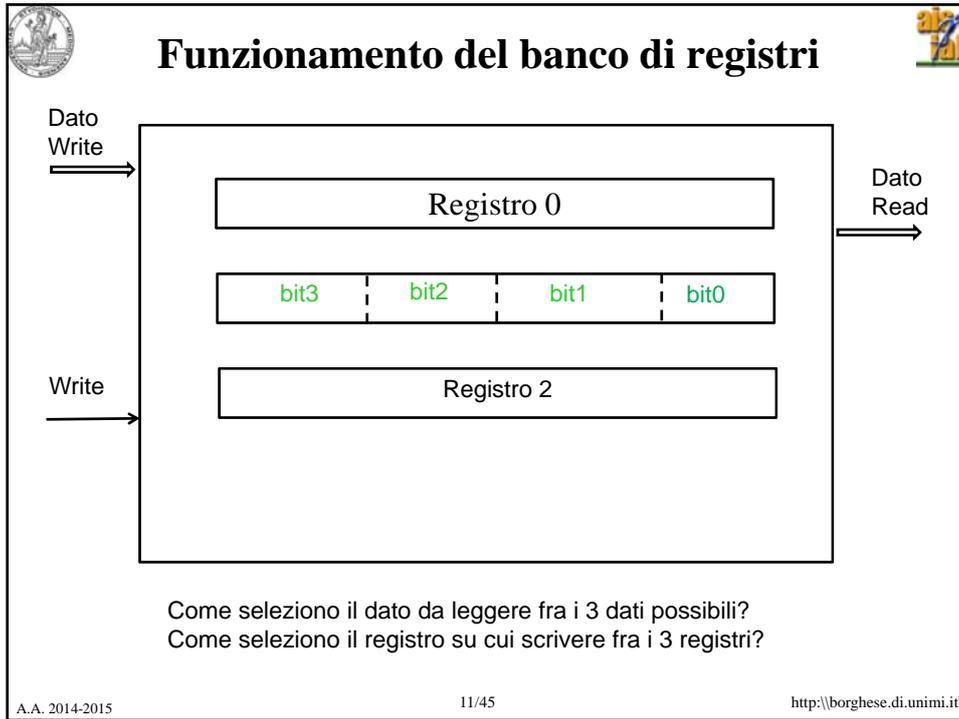


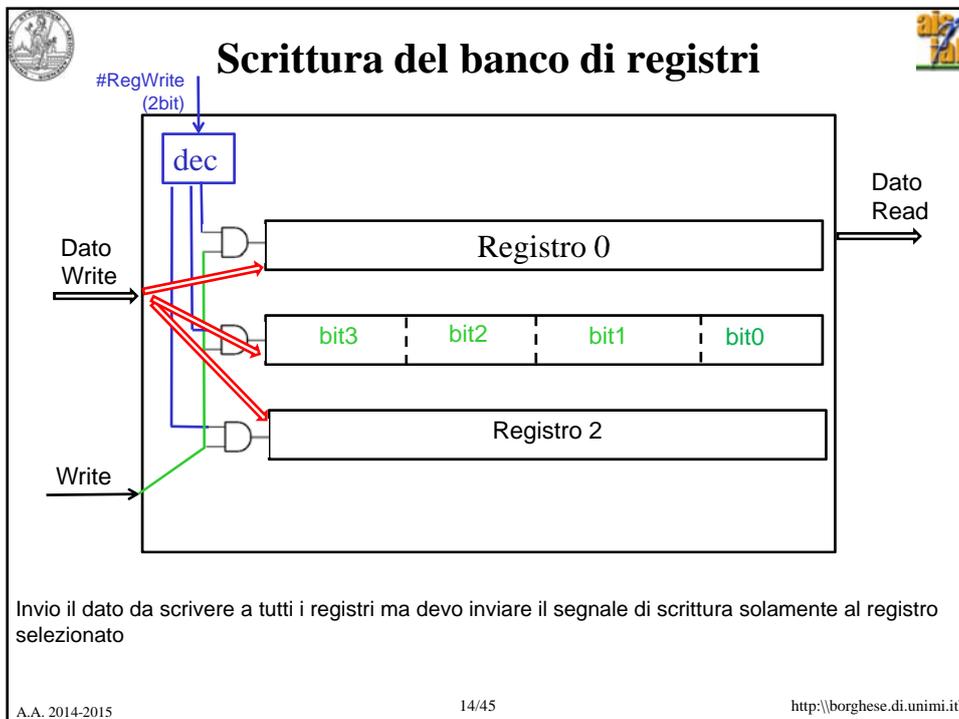
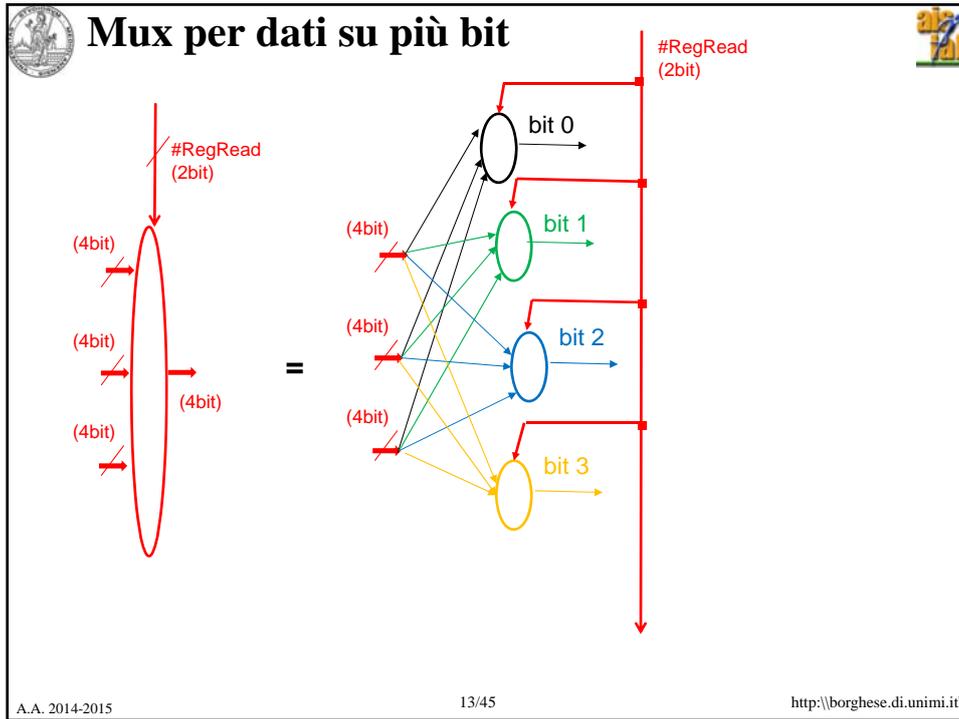
A.A. 2014-2015 9/45

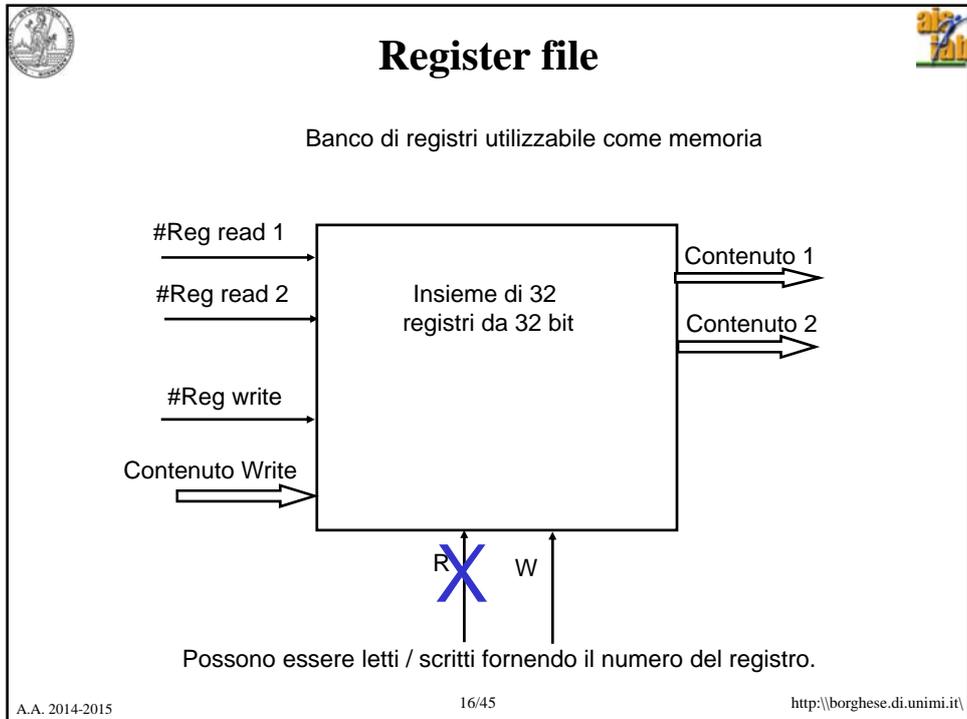
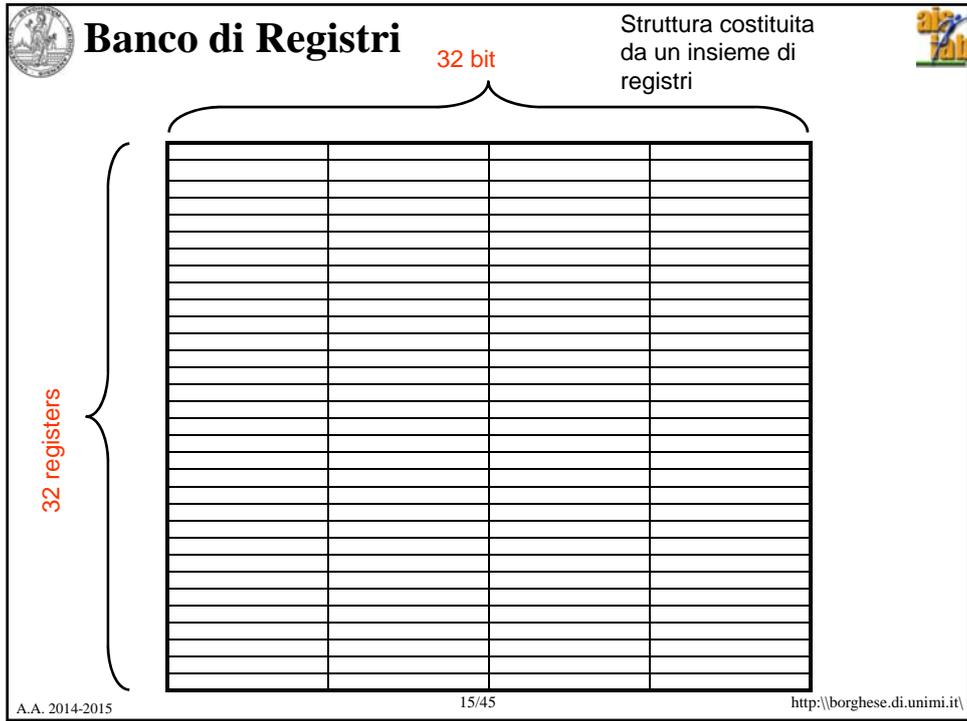
### Un banco 4 registri x 3bit



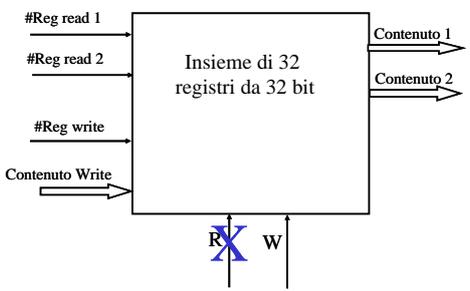
A.A. 2014-2015 10/45 <http://borgese.di.unimi.it/>







## Gestione del register file



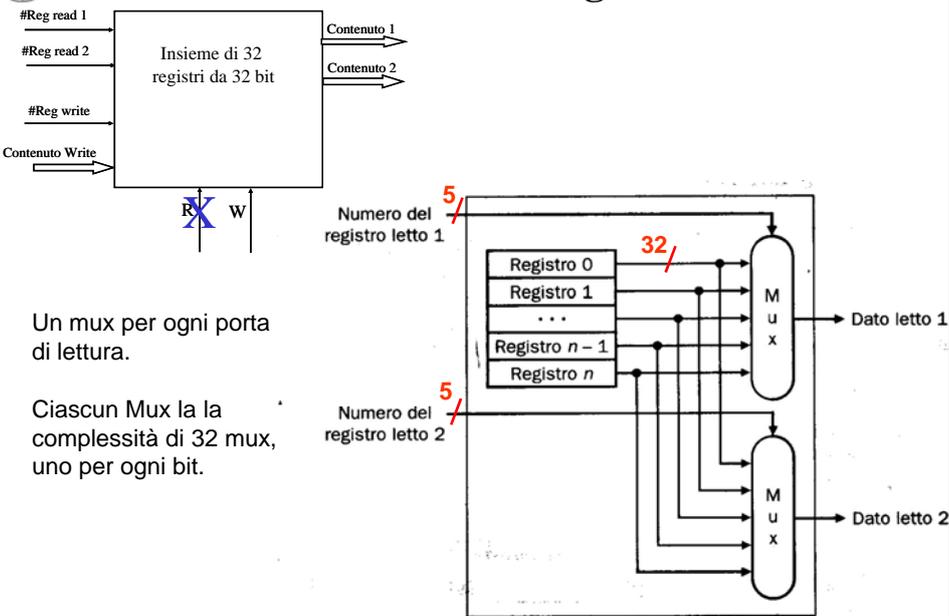
La lettura non modifica il contenuto di un registro (collego uscita Slave con il circuito combinatorio).

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

$\#bit\_indirizzamento = \log_2 \#bit$

A.A. 2014-2015 17/45 <http://borgese.di.unimi.it/>

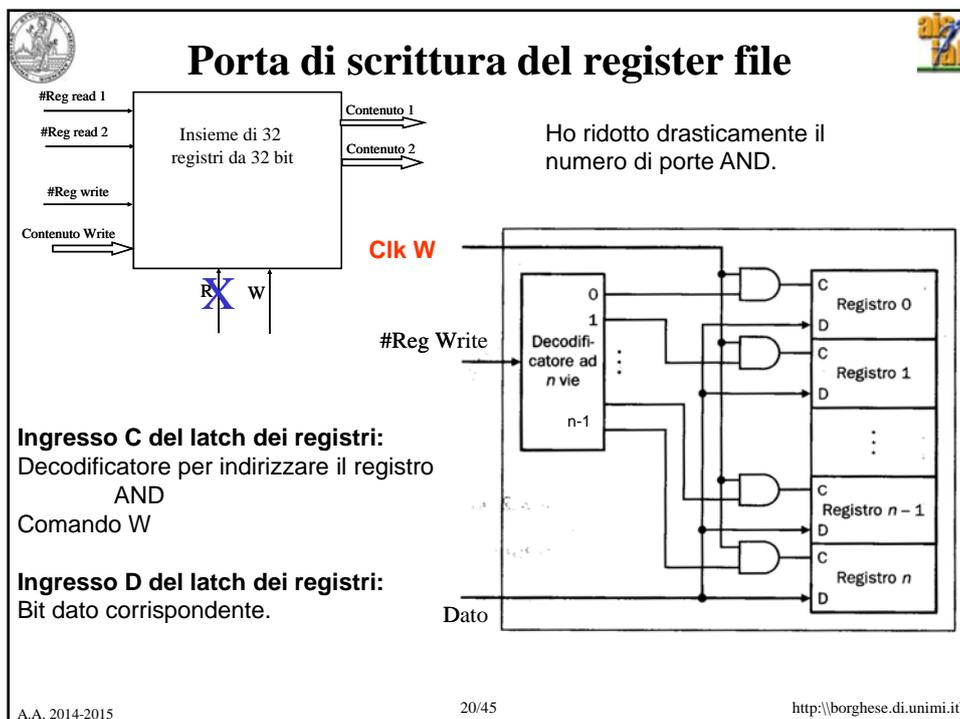
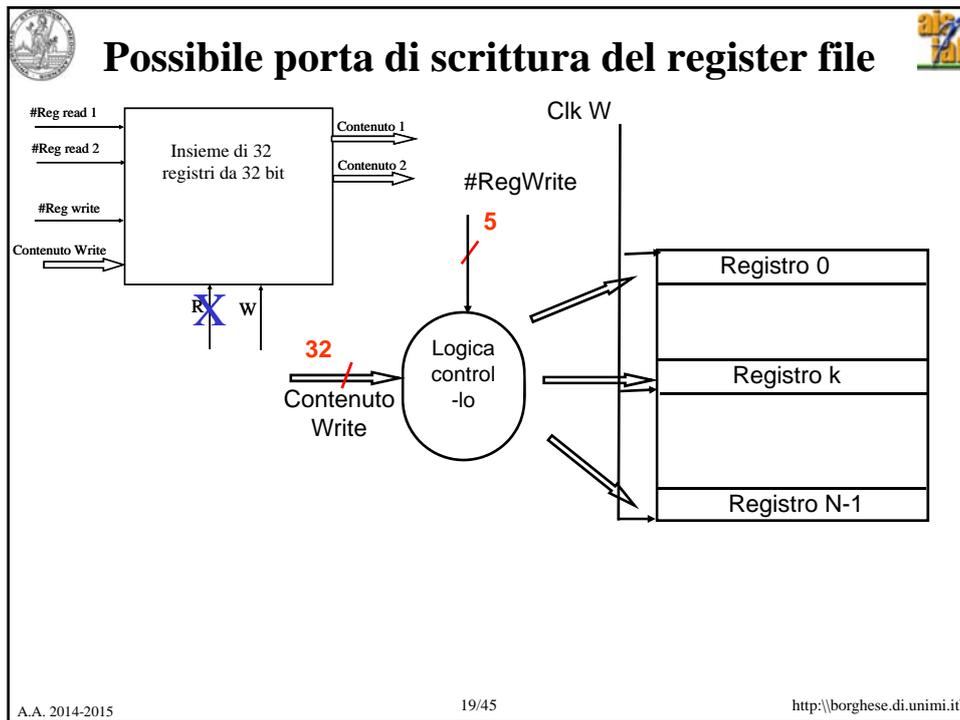
## Porta di lettura del register file

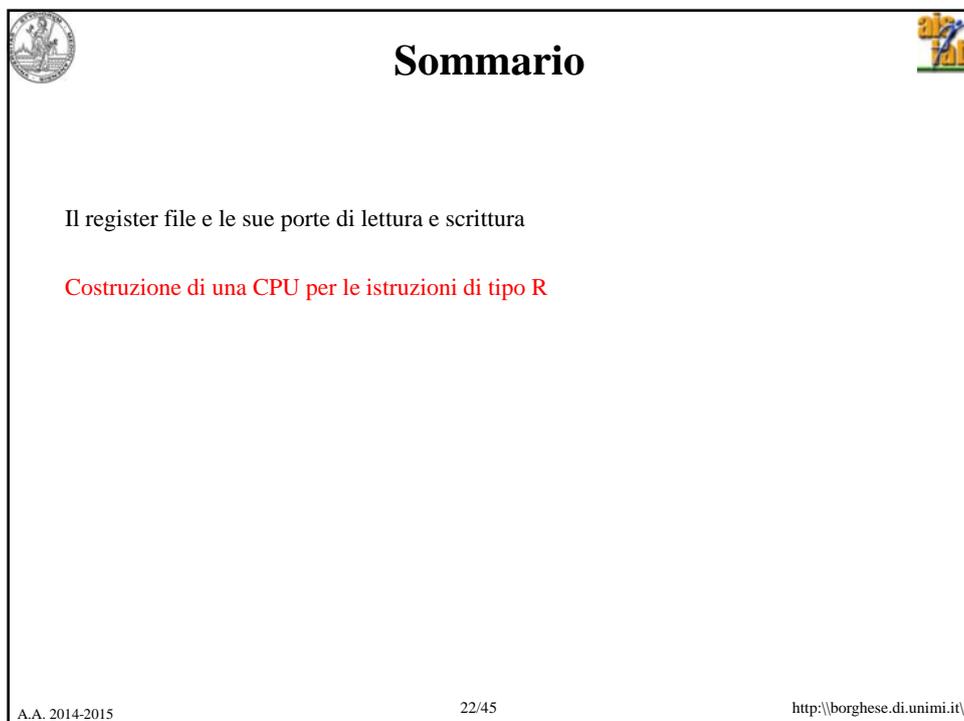
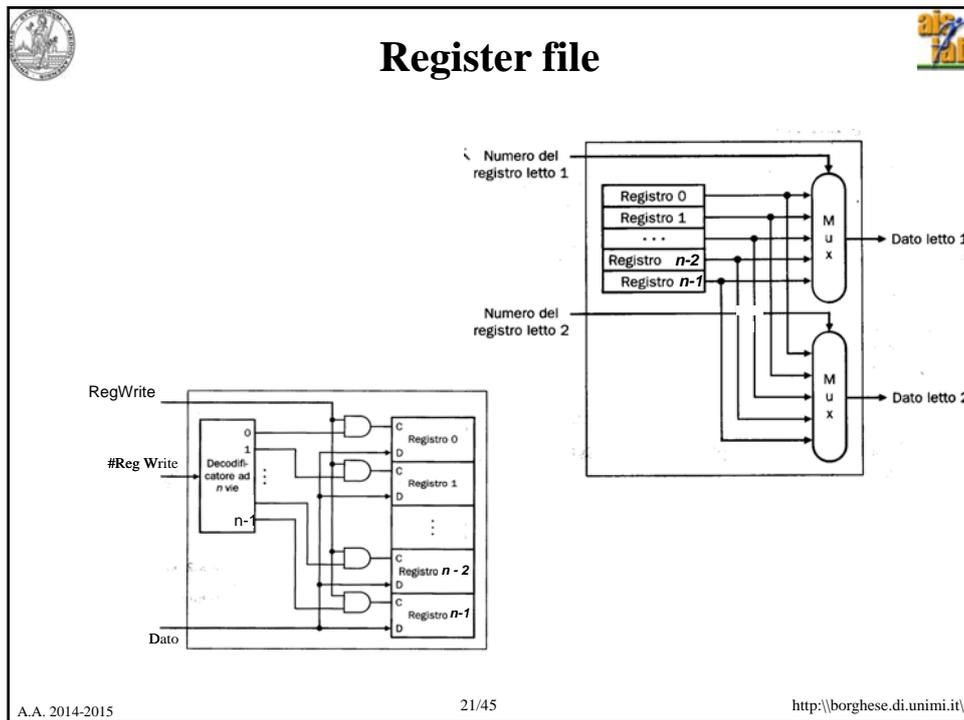


Un mux per ogni porta di lettura.

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

A.A. 2014-2015 18/45 <http://borgese.di.unimi.it/>







## Introduzione alla CPU

- Istruzioni di salto dell'ISA MIPS
- I diversi tipi di istruzioni: Istruzioni di tipo R
- I diversi tipi di istruzioni: Istruzioni di tipo I
- I diversi tipi di istruzioni: Istruzioni di tipo J

A.A. 2014-2015 23/45 <http://borgese.di.unimi.it/>



## Obiettivo

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).

A.A. 2014-2015 24/45 <http://borgese.di.unimi.it/>



## 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				

A.A. 2014-2015
25/45
<http://borgese.di.unimi.it/>

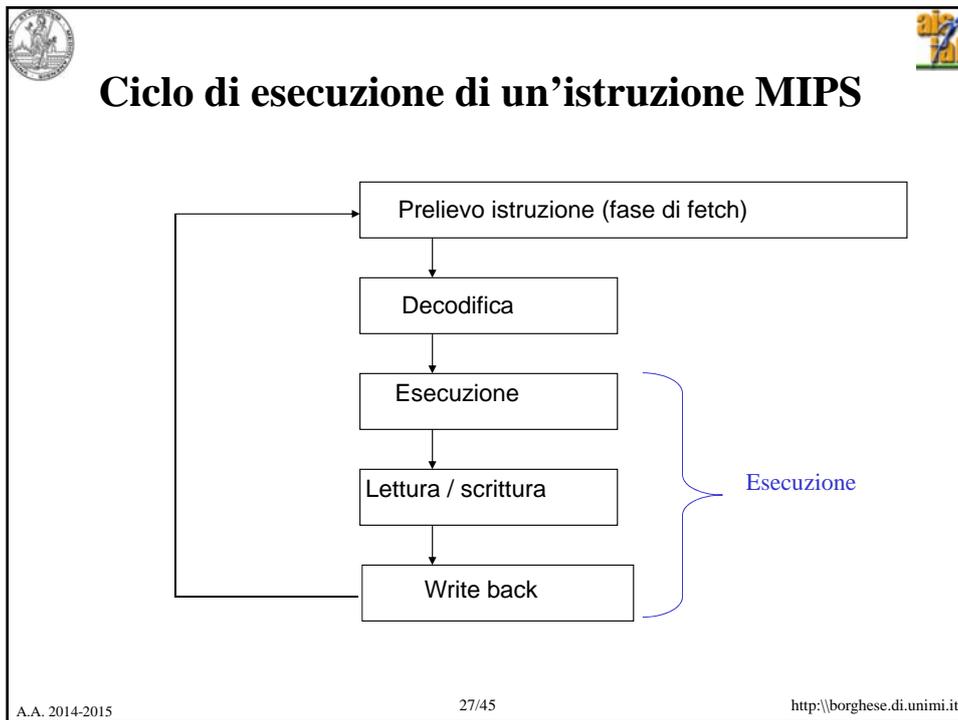


## Istruzioni



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

A.A. 2014-2015
26/45
<http://borgese.di.unimi.it/>



## Letture 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.

A.A. 2014-2015 28/45 http:\\borgnese.di.unimi.it\




## 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 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 caricamento dalla memoria sono separate da quelle aritmetico/logiche.

A.A. 2014-2015 29/45 <http://borgese.di.unimi.it/>




## 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.

A.A. 2014-2015 30/45 <http://borgese.di.unimi.it/>



## 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 caricamento dalla memoria sono separate da quelle aritmetico/logiche. Se effettui una Lettura / Scrittura, **non** esegui 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.*

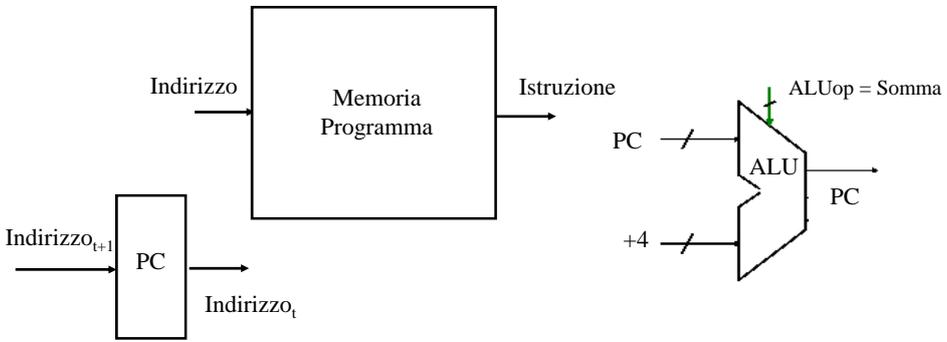
- Calcolo: utilizzo dell'ALU dopo aver letto i registri:
  - per calcolare l'indirizzo in memoria.
  - per eseguire un'operazione logico-aritmetica.
  - per effettuare test (uguaglianza, disuguaglianza, <...).
- Accesso alla memoria.
- Scrittura del risultato nel register file.

A.A. 2014-2015
33/45
<http://borghese.di.unimi.it/>

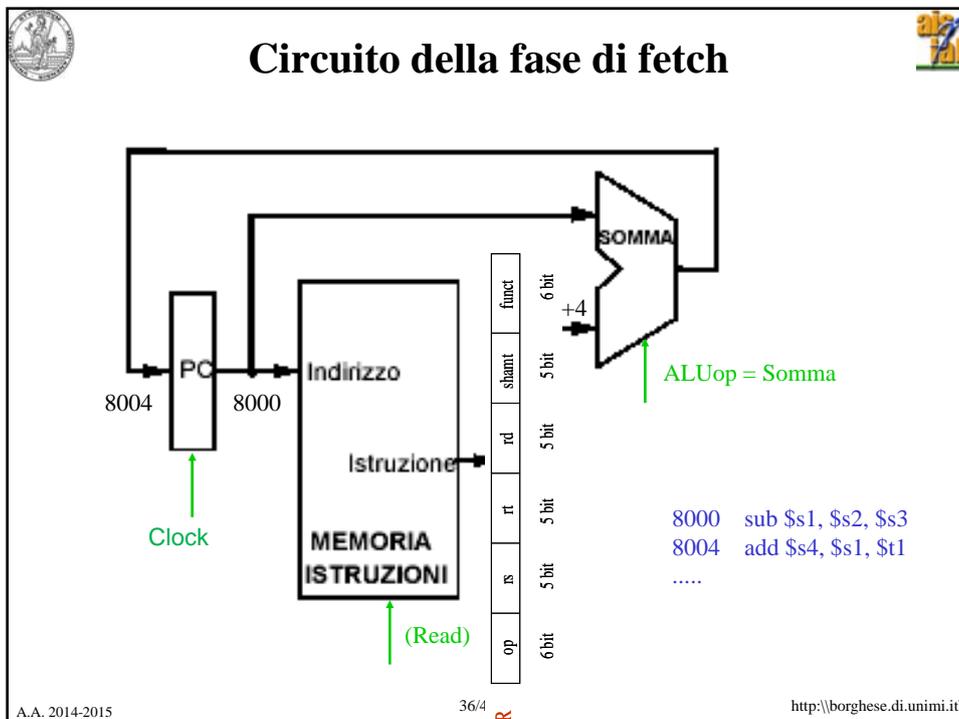
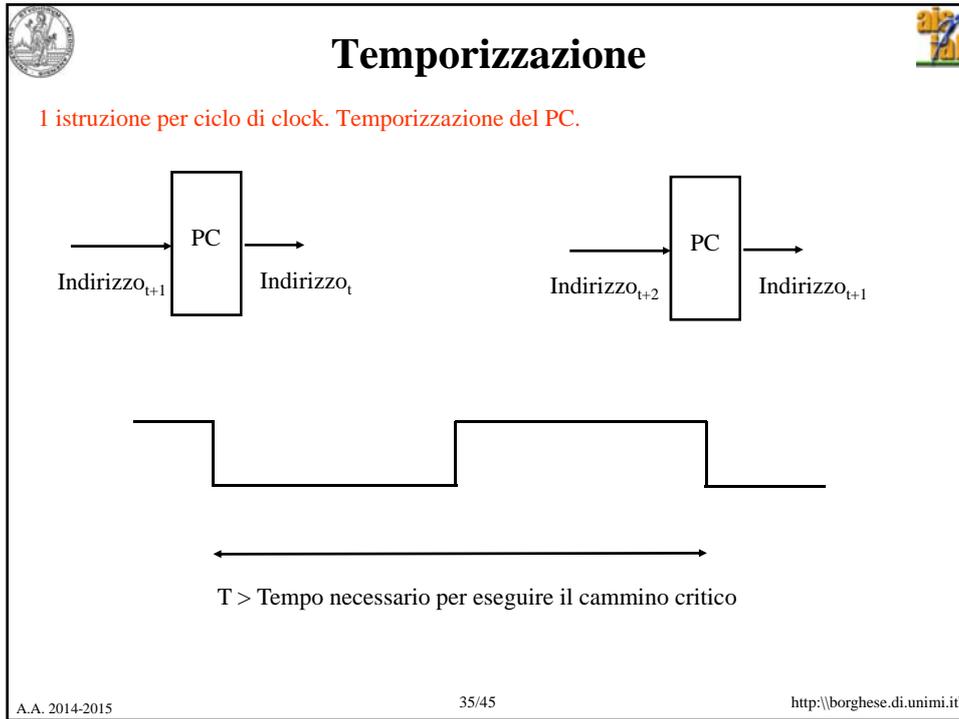



## Fase di fetch

- 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.



A.A. 2014-2015
34/45
<http://borghese.di.unimi.it/>






## 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

A.A. 2014-2015

37/45

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




## Fase di decodifica

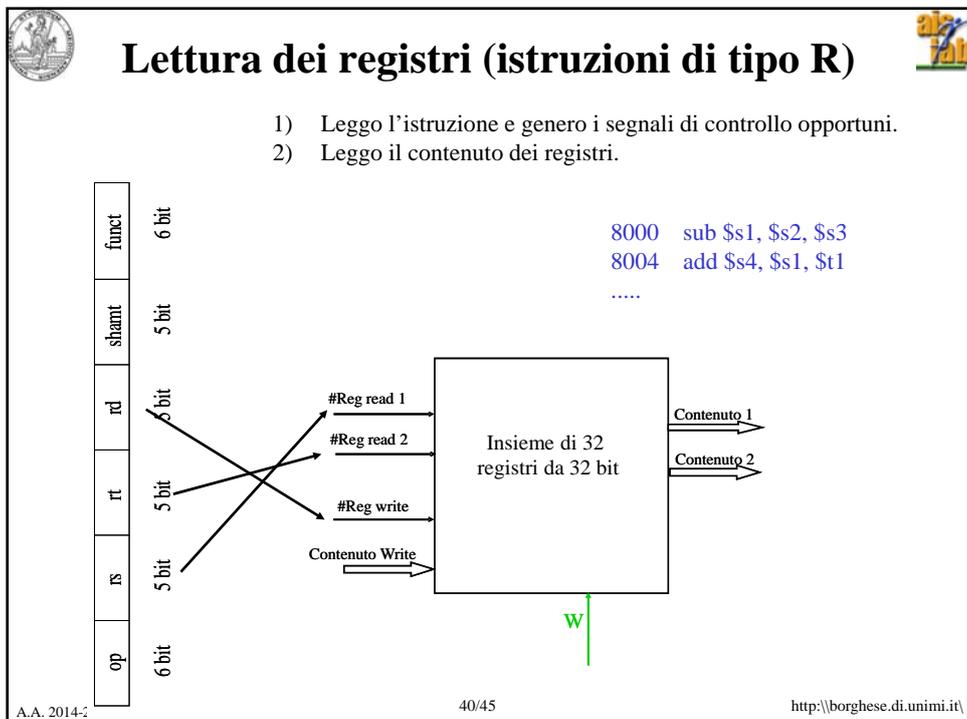
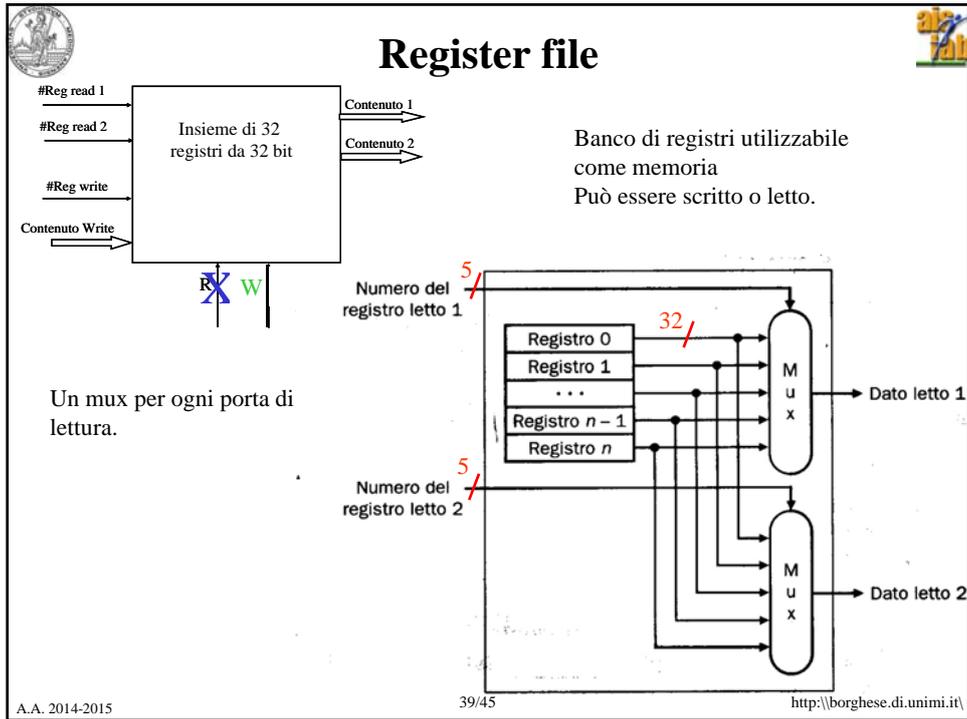
- 1) Leggo l'istruzione e genero i segnali di controllo opportuni.
- 2) Leggo il contenuto dei registri.

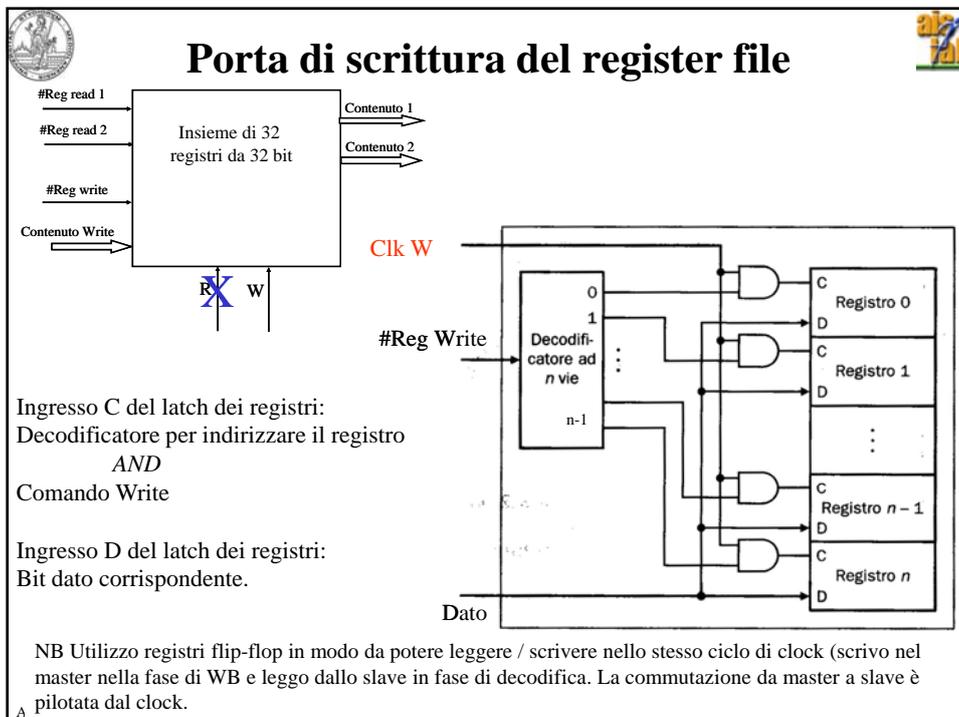
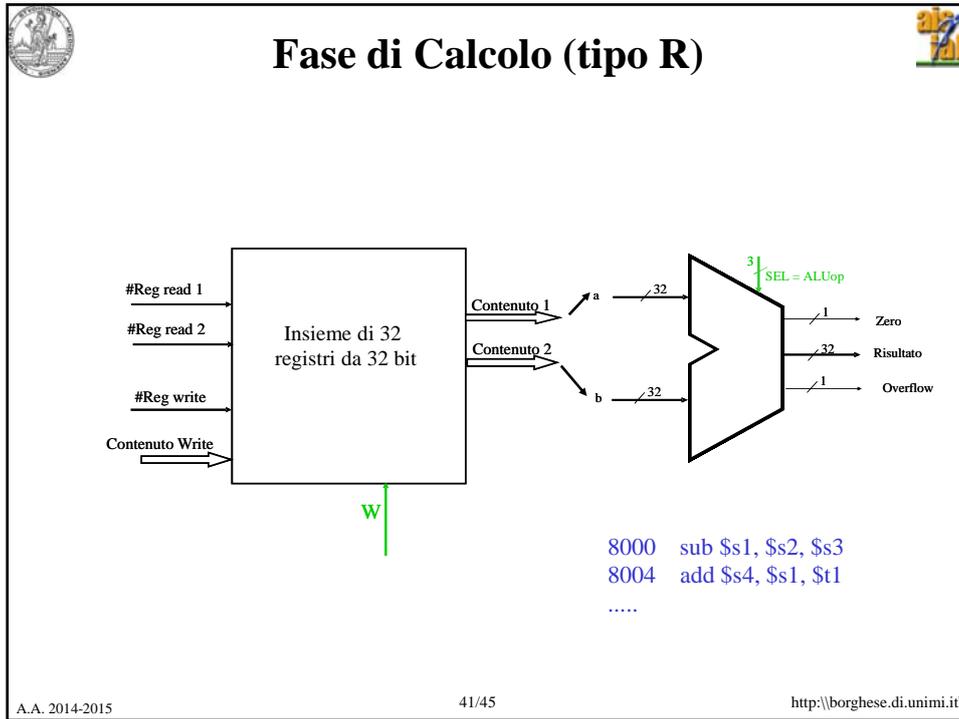
The diagram illustrates the decoding phase. On the left, a vertical stack of boxes represents the instruction format fields: op (6 bit), rs (5 bit), rt (5 bit), rd (5 bit), shamt (5 bit), and funct (6 bit). An arrow points from this stack to a central circle labeled 'Unità Controllo'. From the circle, a green arrow labeled 'Segnali di controllo' points to the right, indicating the output of the decoding process.

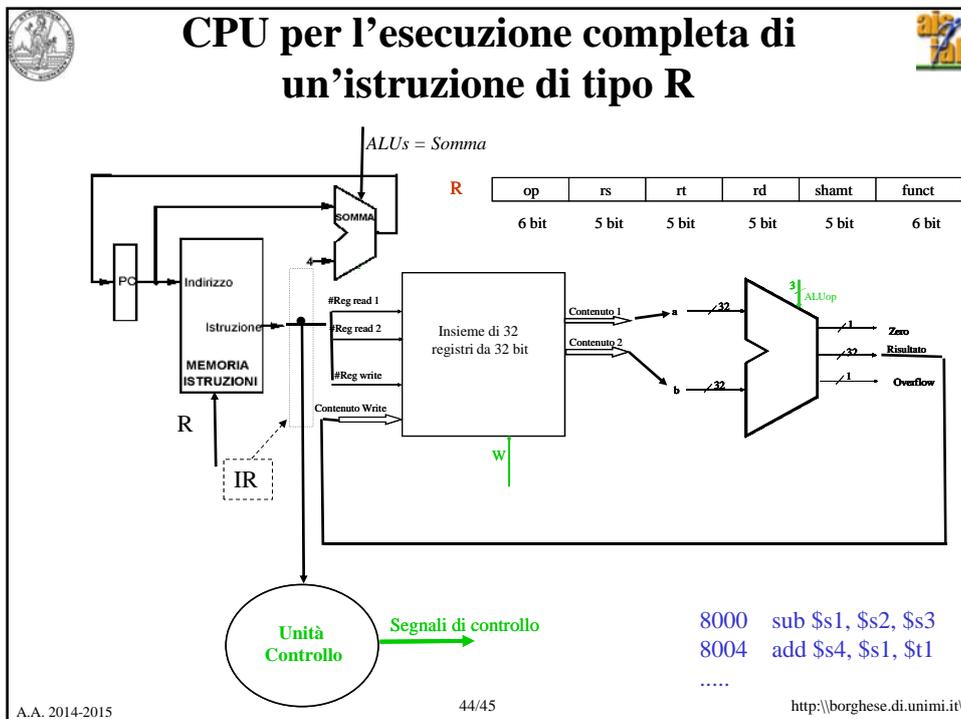
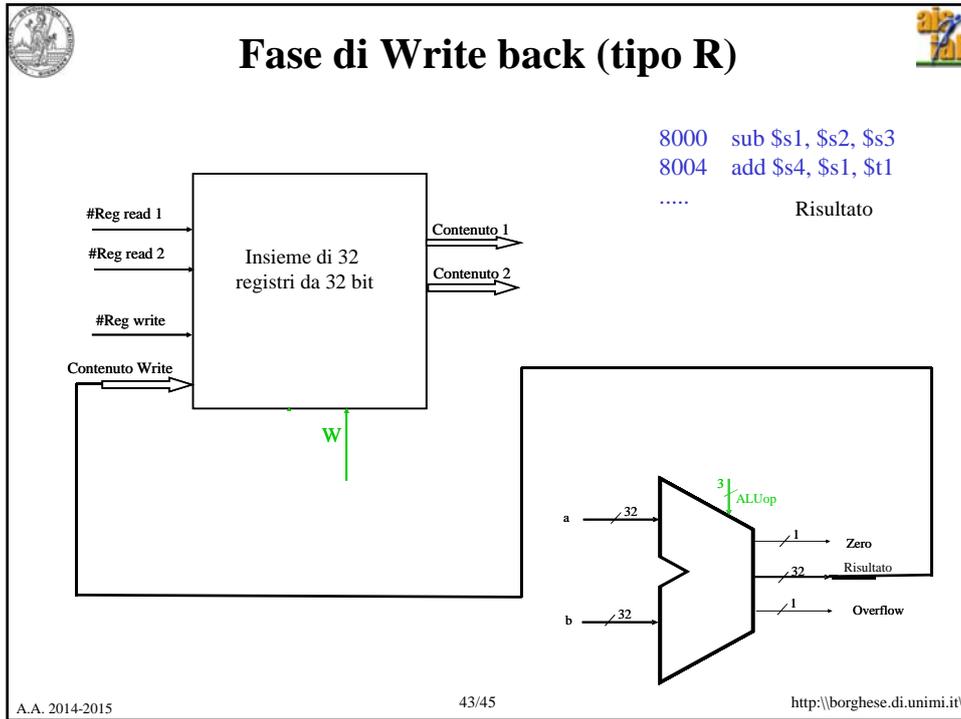
A.A. 2014-2015

38/45

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









## Sommario



Il register file e le sue porte di lettura e scrittura

Costruzione di una CPU per le istruzioni di tipo R