



# Architettura degli elaboratori CPU a ciclo singolo

Prof. Alberto Borghese  
Dipartimento di Informatica  
[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

Costruzione di una CPU per le istruzioni di tipo R  
Costruzione di una CPU per le istruzioni di tipo I



## Obiettivo



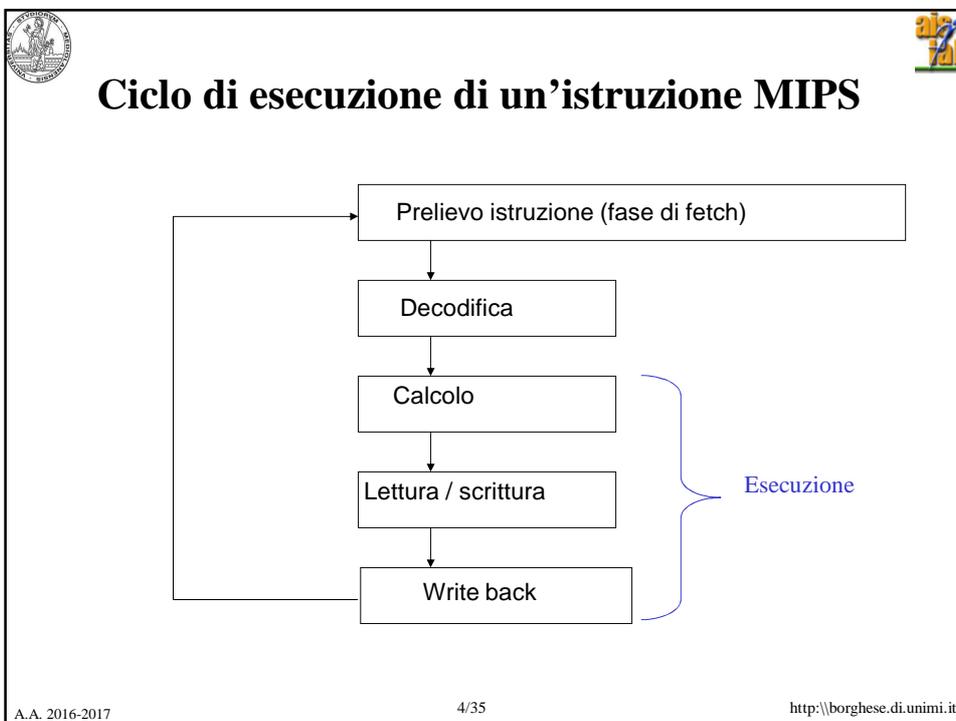
Costruzione di una CPU completa che sia in grado di eseguire:

- Istruzioni logico-matematiche (e.g. add, sub, and....). e.g. add \$t0, \$t1, \$t2).
- 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

A.A. 2016-2017

3/35

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





## 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**

A.A. 2016-2017

5/35

<http://borghese.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. 2016-2017

6/35

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



## Istruzioni



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

A.A. 2016-2017
7/35
<http://borghese.di.unimi.it/>



## 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.

A.A. 2016-2017
8/35
<http://borghese.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. 2016-2017 9/35 http://borghese.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. 2016-2017 10/35 http://borghese.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. 2016-2017

13/35

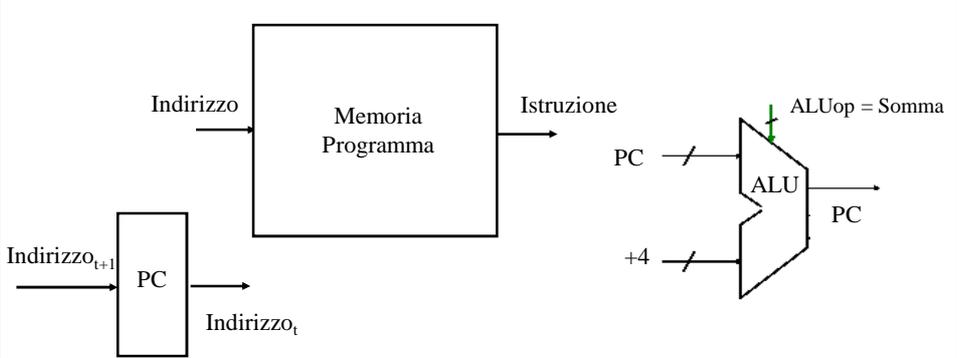
<http://borghese.di.unimi.it/>



## Fase di fetch



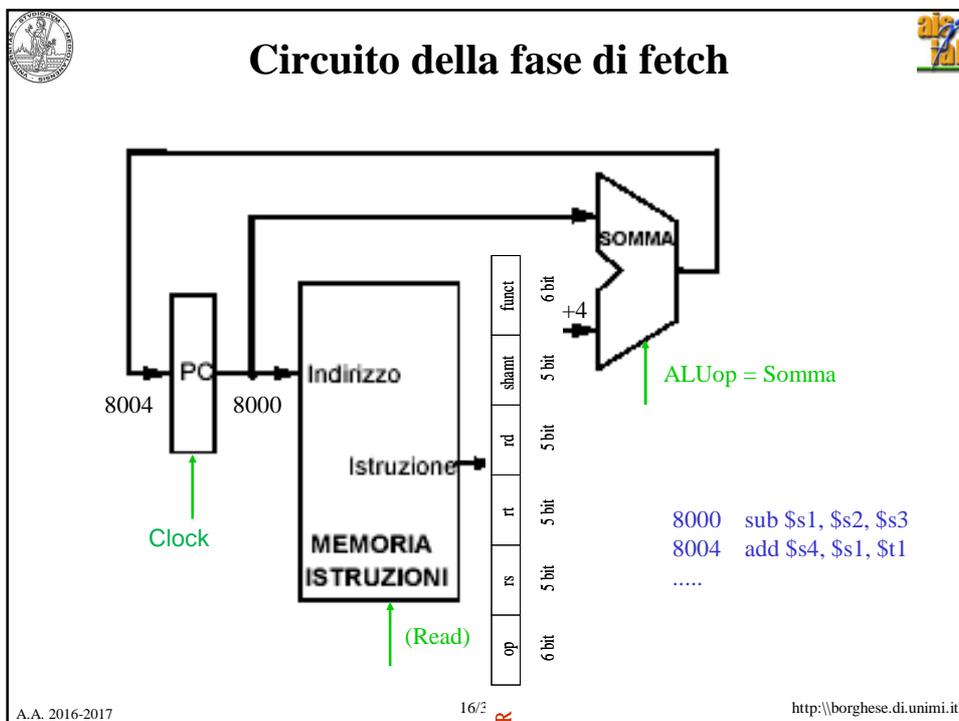
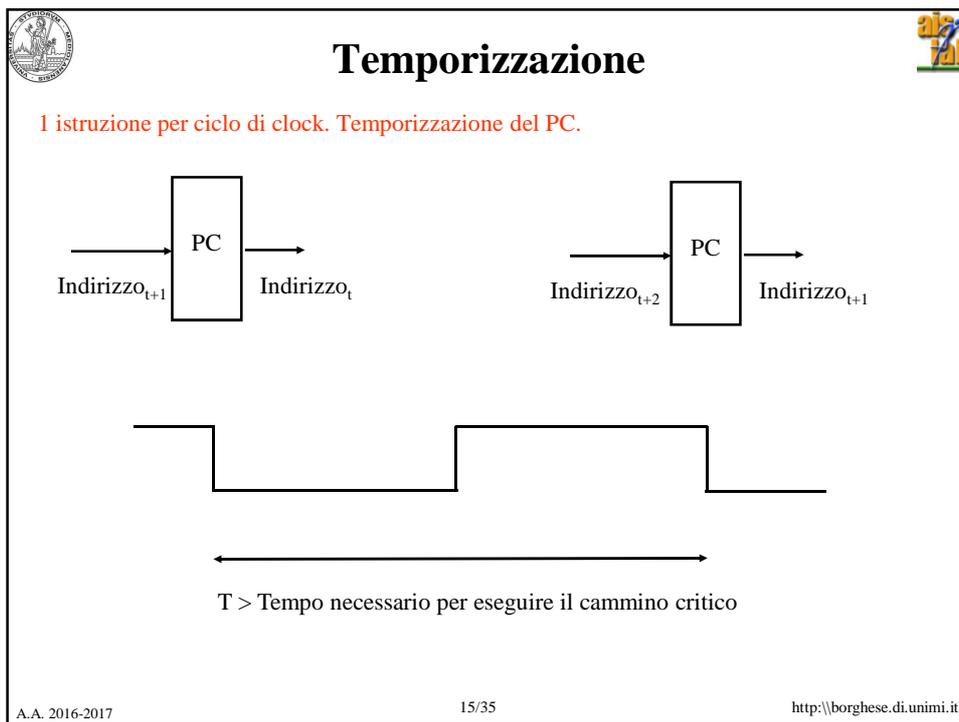
- 1) Memorizzare l'indirizzo dell'istruzione nel PC.
- 2) Leggere l'istruzione dalla memoria.
- 3) Aggiornare l'indirizzo in modo che in PC sia contenuto l'indirizzo dell'istruzione successiva.



A.A. 2016-2017

14/35

<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. 2016-2017

17/35

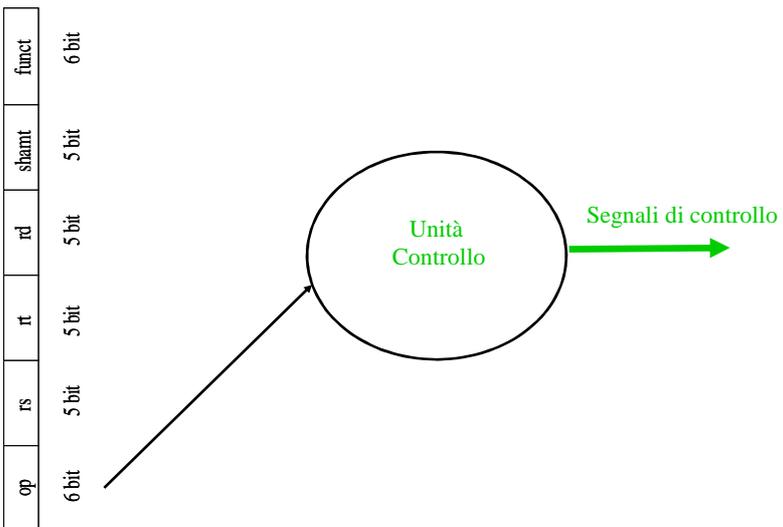
<http://borghese.di.unimi.it/>




## Fase di decodifica

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

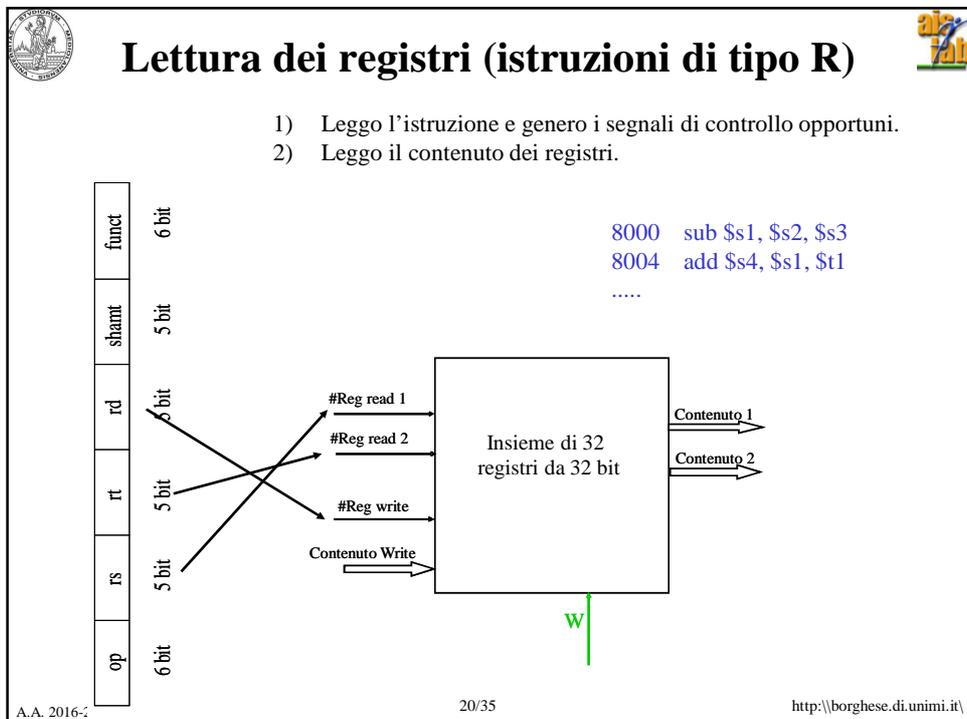
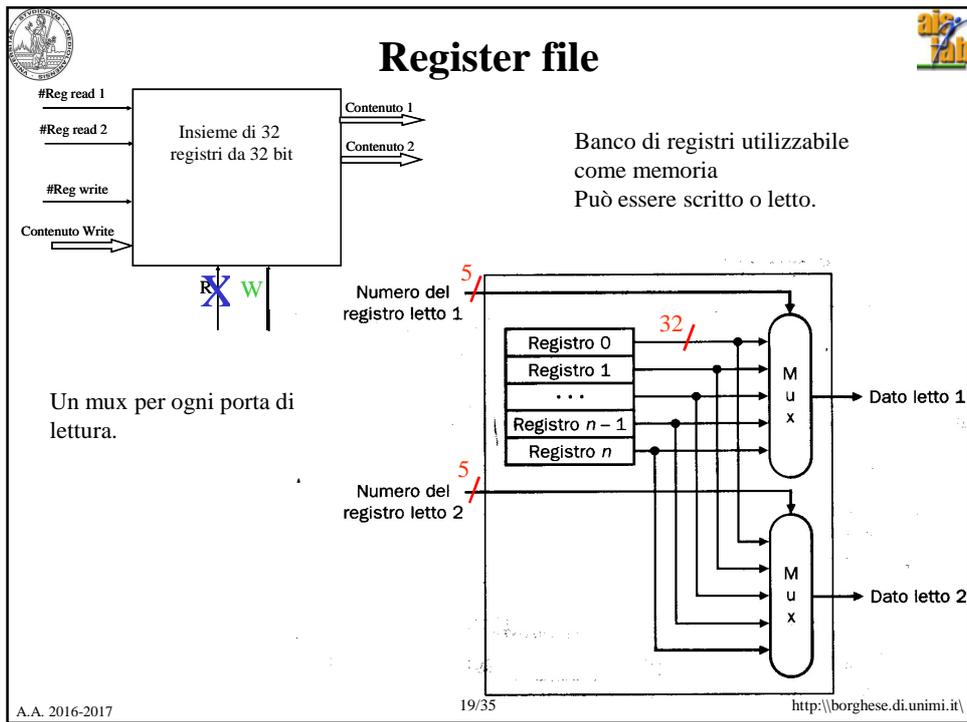
op	rs	rt	rd	shamt	funct
6 bit	5 bit	5 bit	5 bit	5 bit	6 bit

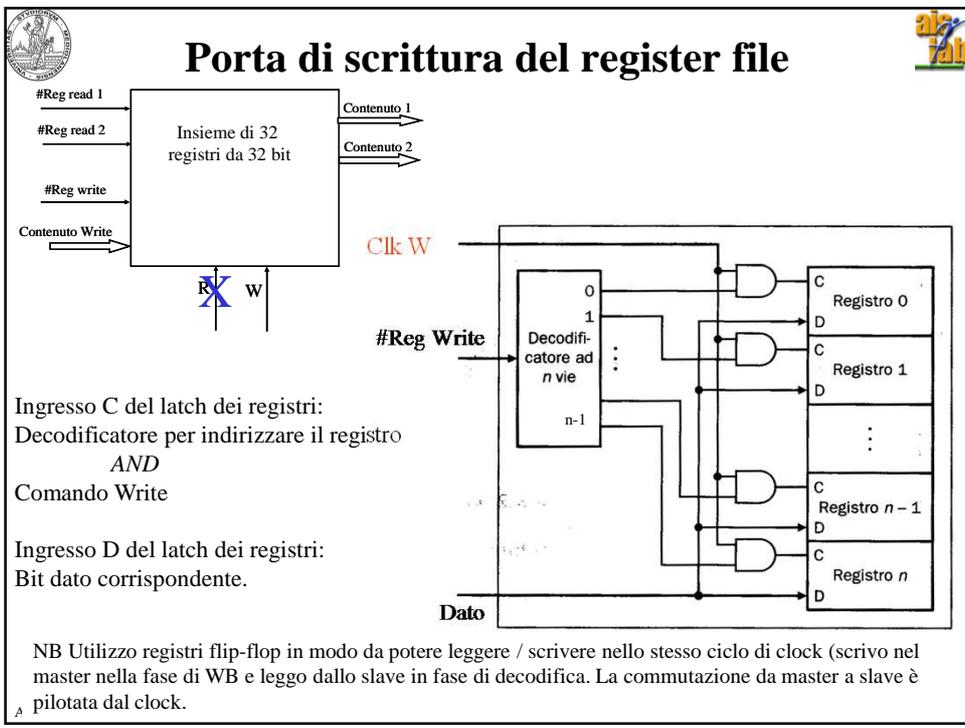
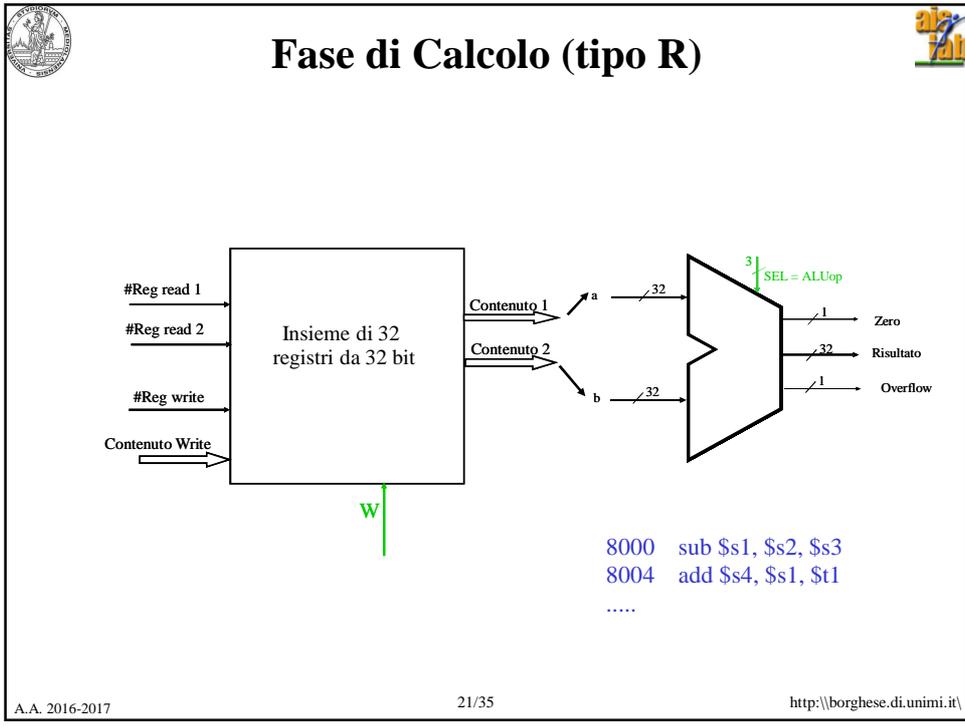


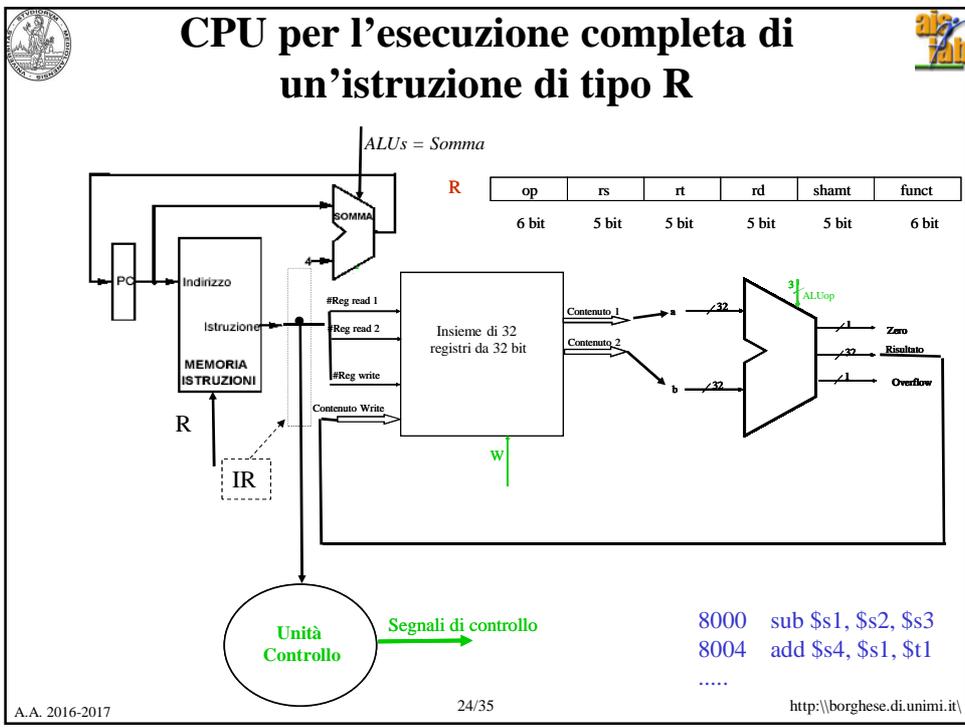
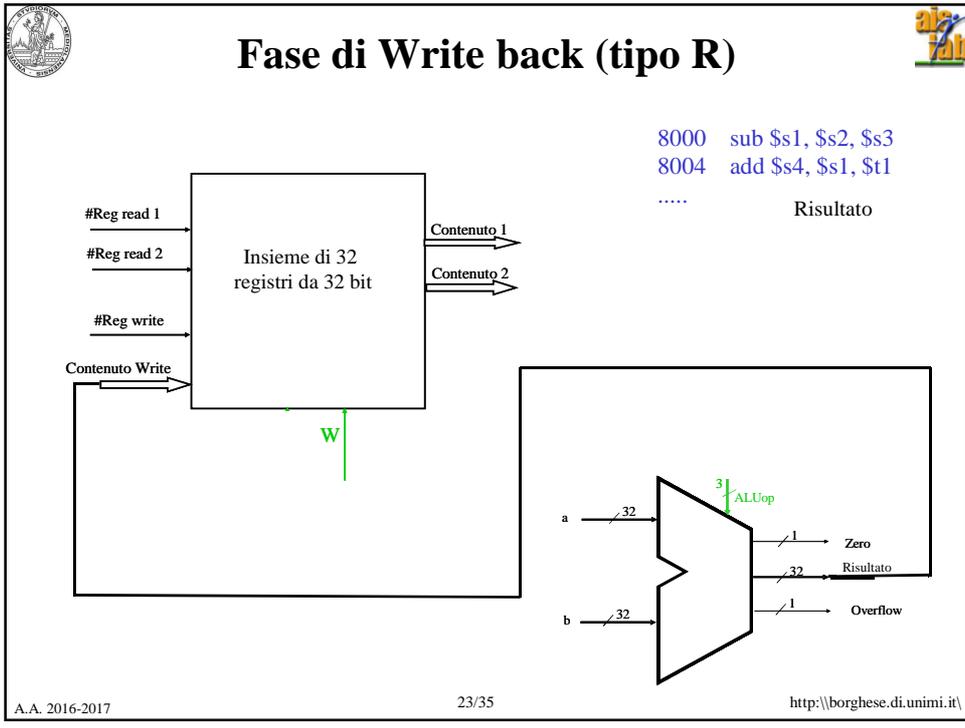
A.A. 2016-2017

18/35

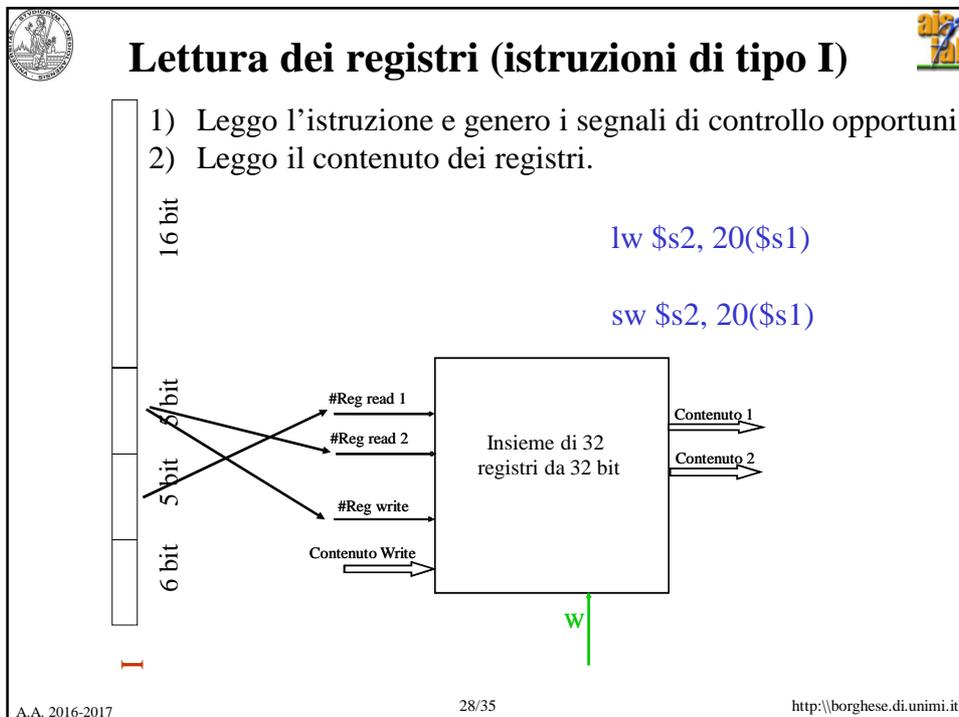
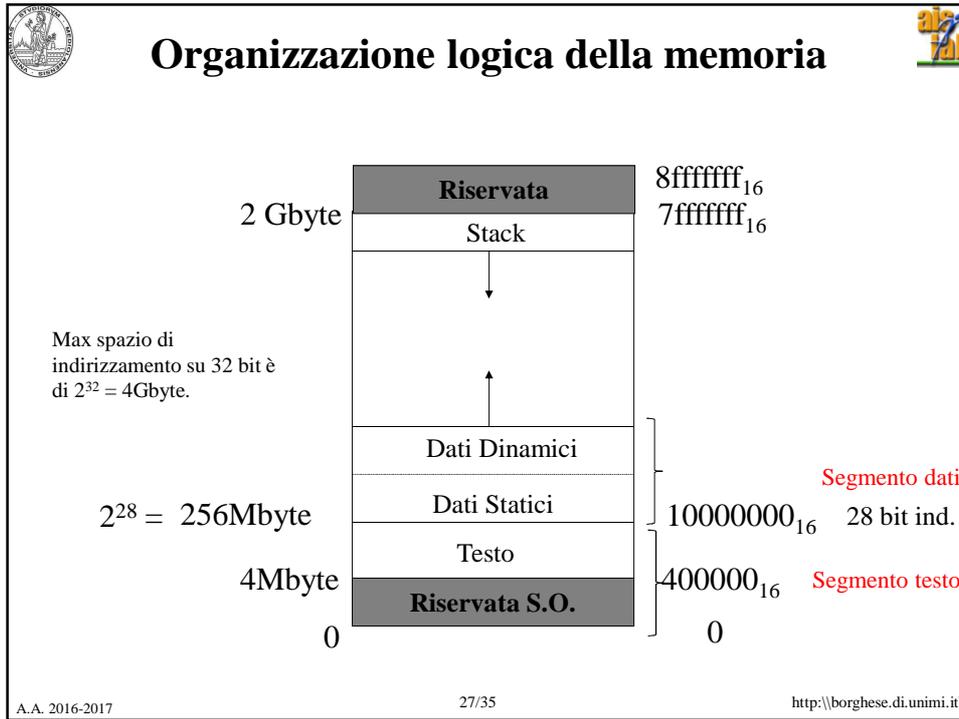
<http://borghese.di.unimi.it/>

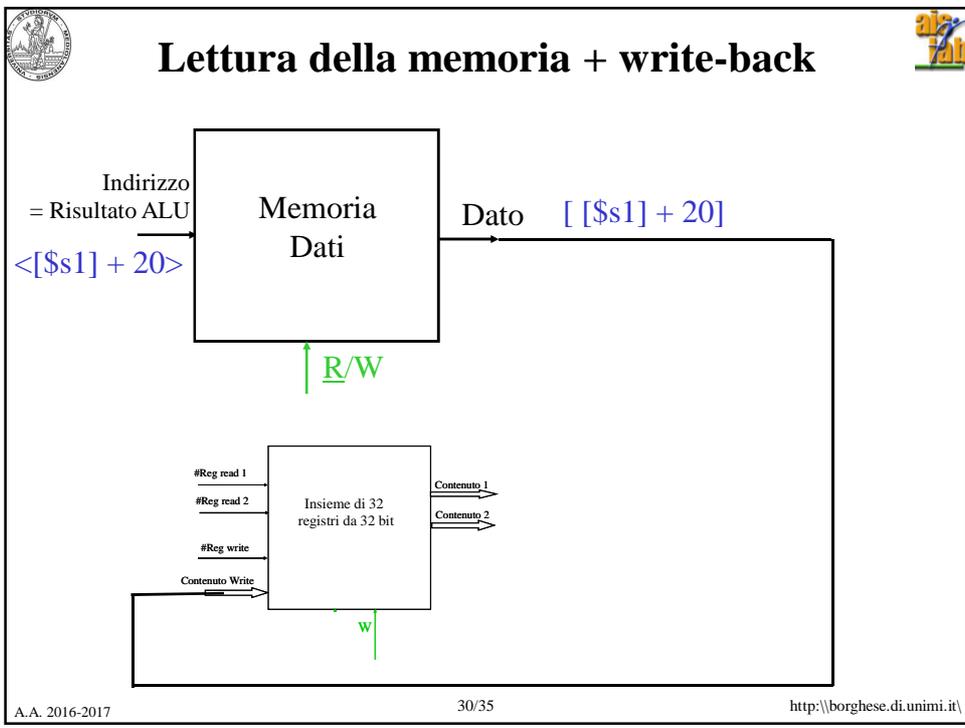
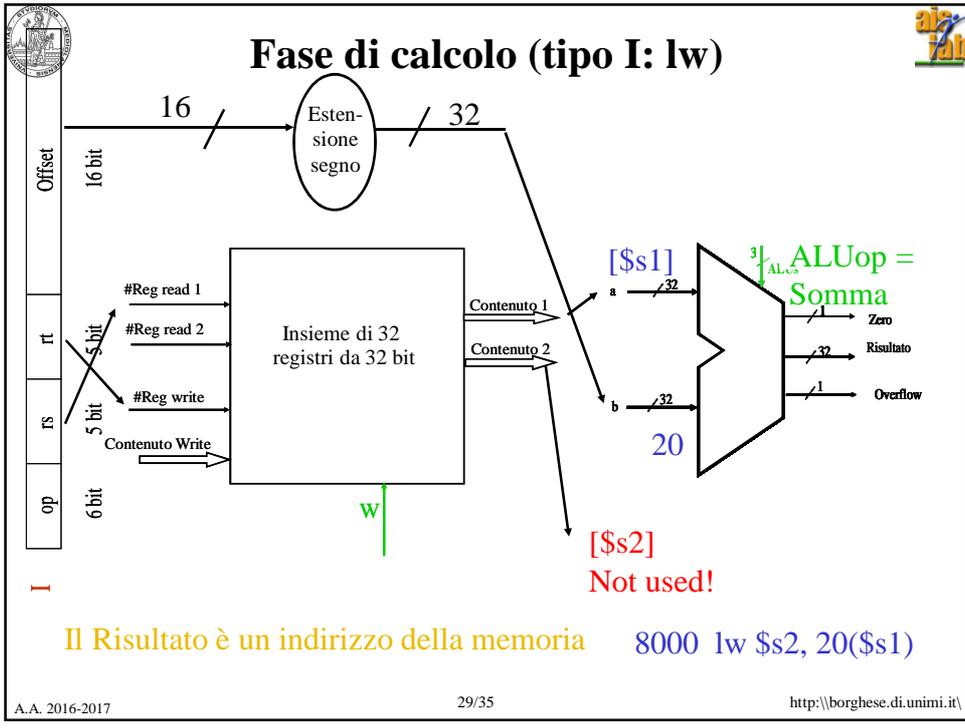


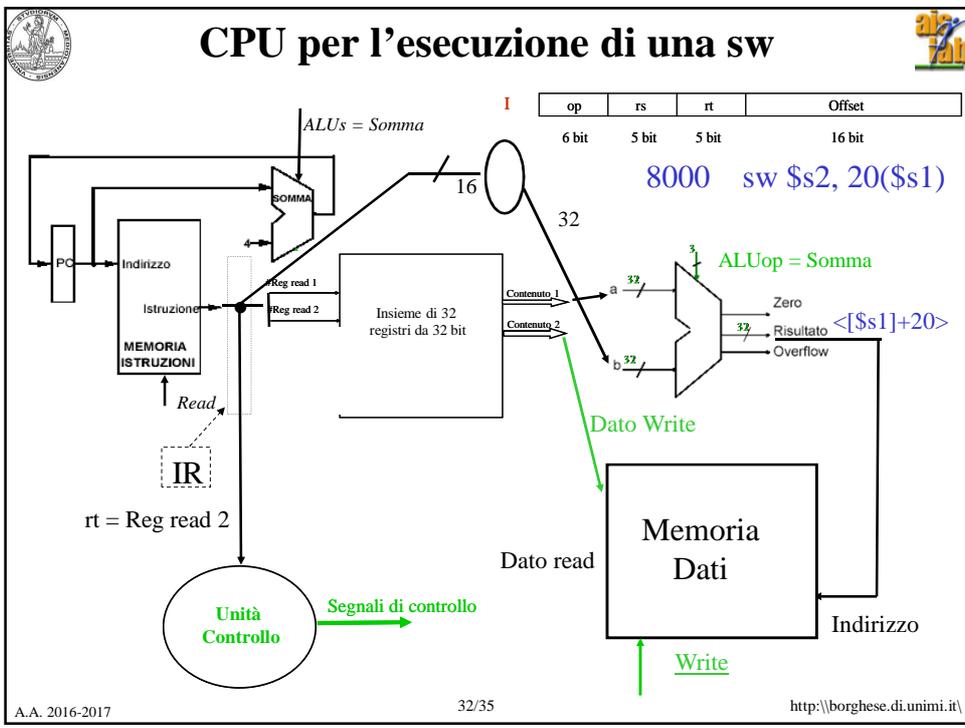
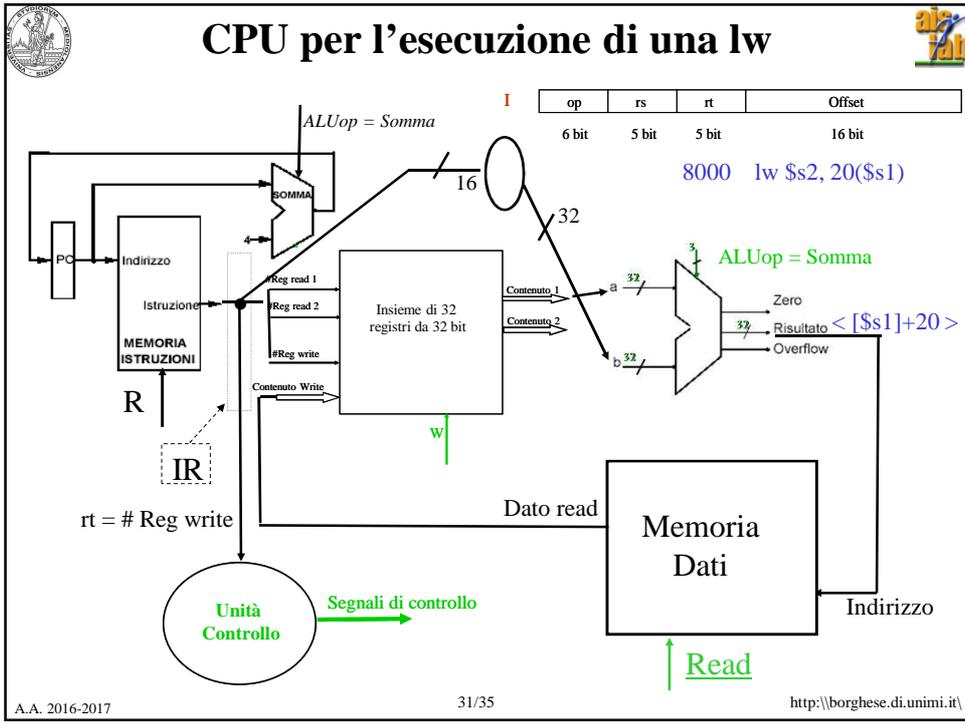


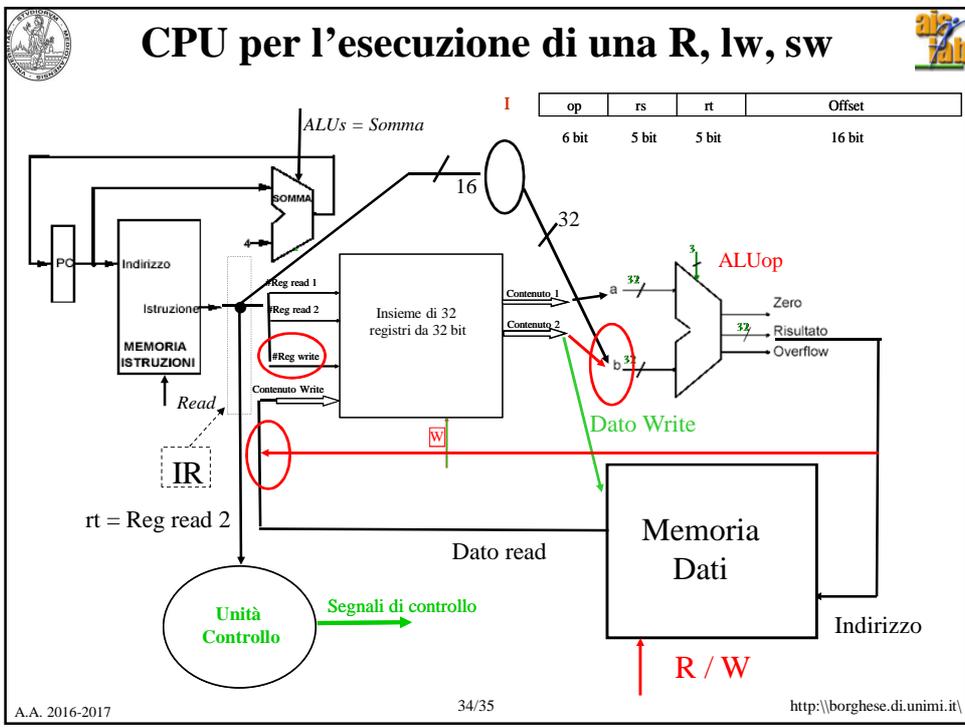
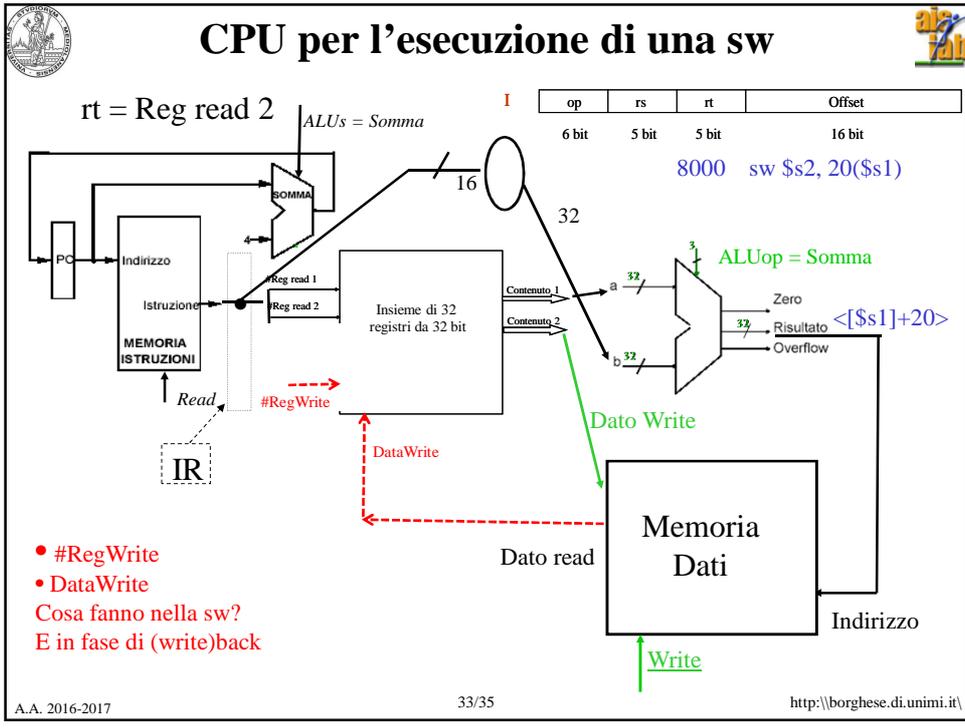














## Sommario



Costruzione di una CPU per le istruzioni di tipo R  
Costruzione di una CPU per le istruzioni di tipo I