



Linguaggio macchina e register file

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

Università degli Studi di Milano

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

A.A. 2016-2017 1/36 http:\\borgese.di.unimi.it\



Il linguaggio macchina

- I diversi tipi di istruzioni: Istruzioni di tipo I
- I diversi tipi di istruzioni: Istruzioni di tipo J
- Register file

A.A. 2016-2017 2/36 http:\\borgese.di.unimi.it\




Formato istruzioni di tipo I

op	rs	rt	costante
6 bit	5 bit	5 bit	16 bit

- In questo caso, i campi hanno il seguente significato:
 - **op** identifica il tipo di istruzione;
 - **rs** indica il registro sorgente. Nel caso di una lw contiene il registro base;
 - **rt** indica il registro target. Nel caso di una lw, contiene il registro destinazione dell'istruzione di caricamento;
 - **costante**. Nel caso di una lw riporta lo spiazzamento (offset).

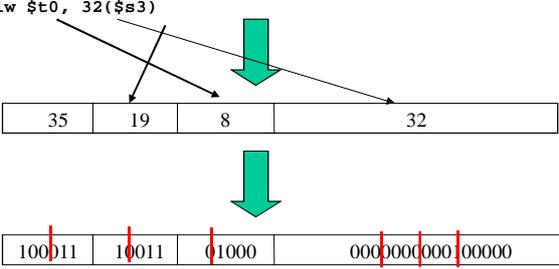
A.A. 2016-2017
3/36
<http://borghese.di.unimi.it/>




Istruzioni di tipo I: accesso a memoria

Con questo formato una istruzione **lw** (**sw**) può indirizzare byte nell'intervallo -2^{15} ($-32K$) + $+2^{15}-1$ ($32K-1$) rispetto all'indirizzo base: indirizzo = indirizzo_base + offset (= costante)

lw \$t0, 32(\$s3)



0x8E680020

A.A. 2016-2017
4/36
<http://borghese.di.unimi.it/>



Istruzioni di tipo I: accesso memoria



Nome campo	op	rs	rt	indirizzo			
Dimensione	6-bit	5-bit	5-bit	16-bit			
<code>lw \$t0, 32 (\$s3)</code>	100011	10011	01000	0000	0000	0010	0000

Nome campo	op	rs	rt	indirizzo			
Dimensione	6-bit	5-bit	5-bit	16-bit			
<code>sw \$t0, 32 (\$s3)</code>	101011	10011	01000	0000	0000	0010	0000

A.A. 2016-2017
5/36
<http://borghese.di.unimi.it/>



Versione I di istruzioni aritmetico-logiche



Nome campo	op	rs	rt	costante			
Dimensione	6-bit	5-bit	5-bit	16-bit			
<code>addi \$t0, \$s3, 64</code>	001000	10011	01000	0000	0000	0100	0000

Nome campo	op	rs	rt	costante			
Dimensione	6-bit	5-bit	5-bit	16-bit			
<code>addi \$s1, \$s2, 4</code>	001000	10010	10001	0000	0000	0000	0100

Nome campo	op	rs	rt	costante			
Dimensione	6-bit	5-bit	5-bit	16-bit			
<code>slti \$t0, \$s2, 8</code>	001010	10010	01000	0000	0000	0000	1000

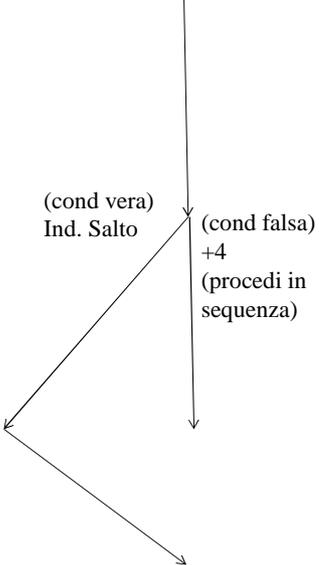
↙ # \$t0 = 1 if \$s2 < 8

A.A. 2016-2017
6/36
<http://borghese.di.unimi.it/>




Istruzioni di salto condizionato

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



A.A. 2016-2017

7/36

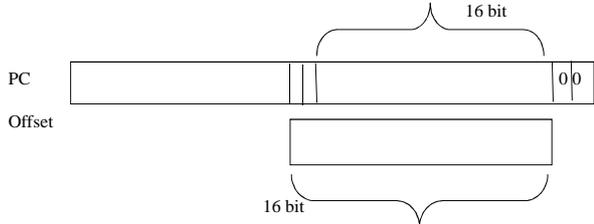
<http://borghese.di.unimi.it/>




Allargamento dello spazio di indirizzamento relativo

0000	0	0
0100	1	4
1000	2	8
1100	3	12

Calcolo lo spiazzamento in numero di parole invece che di Byte.
 Considero 64Mword (64M istruzioni) invece di 64Kbyte. Lo spazio indirizzabile all'interno del segmento di testo è di 64Kword * 4 = 256Kbyte.
 Moltiplicare per 4 vuol dire operare uno shift a sinistra di due posizioni dell'offset



La costante su 16 bit rappresenta l'offset in termini di numero di istruzioni

A.A. 2016-2017

8/36

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



Calcolo dell'indirizzo di salto



```

                                0x400          addi $t1, $zero, 10 # N=10
for (i=0; i++; i<10)           0x404          addi $t0,$zero,0    # i =0;
{   ....                       0x408   loop:    addi $t0, $t0,1    # i++
                                0x40C          ....
                                0x420          bne $t0, $t1, loop
}

```

Quanto vale il campo costante da inserire nella bne?

A.A. 2016-2017 9/36 http://borghese.di.unimi.it/



Calcolo dell'indirizzo di salto



```

0x400          addi $t1, $zero, 10 # N=10
0x404          addi $t0,$zero,0    # i =0;
0x408   loop:  addi $t0, $t0,1    # i++
0x40C          ....
0x420          bne $t0, $t1, loop

```

L'indirizzo di destinazione è 0x408 (indirizzo dell'etichetta loop)

Lo spiazzamento del salto in byte è pari a: $(0x408 - 0x424) = -28$ Byte
 Lo spiazzamento del salto in numero di istruzioni è pari a -7 istruzioni

Prova: $\text{Indirizzo di salto} = \text{Indirizzo PC} + 4 + \text{Offset} (\# \text{istruzioni}) * 4$
 $\text{Loop} = 0x408 = 0x424 + -7 * 4$

5	8	9	-7 = 1111 1111 1111 1001
---	---	---	--------------------------

A.A. 2016-2017 10/36 http://borghese.di.unimi.it/



Istruzioni di tipo I - Branch



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

L1 = PC+4 + 100 byte Codifica su 18 bit: (00) 000 0000 0001 1001(00) in binario.

Nome campo	op	rs	rt	indirizzo		
Dimensione	6-bit	5-bit	5-bit	16-bit		
beq \$s1, \$s2, L1	000100	10001	10010	1111	1111	1110 0111

L1 = PC+4 -100 byte Codifica su 18 bit: (11)111 1111 1110 0111(00) in binario.

A.A. 2016-2017
11/36
<http://borghese.di.unimi.it/>



Formato R ed operazioni logico-matematiche



Non tutte le operazioni logico-matematico, sono di tipo R.

Le operazioni logico-matematiche di tipo R hanno codice operativo 0.

Non tutte le operazioni con codice operativo 0 sono logico-matematiche (ad esempio ci sono le istruzioni di *jr, syscall...*).

Occorre distinguere il funzionamento dell'istruzione elementare dalla sua codifica.

- Codifiche simili (e.g. Tipo R) possono essere condivise da istruzioni di tipo diverso (e.g. aritmetico-logiche, salto).
- Codifiche diverse (e.g. Tipo I e Tipo R) possono essere condivise da istruzioni dello stesso tipo (e.g. add ed addi)

Non c'è corrispondenza 1 a 1, tra tipi strutturali e tipi funzionali.

A.A. 2016-2017
12/36
<http://borghese.di.unimi.it/>



Il linguaggio macchina



- I diversi tipi di istruzioni: Istruzioni di tipo I
- **I diversi tipi di istruzioni: Istruzioni di tipo J**
- Register file

A.A. 2016-2017

13/36

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



Formato istruzioni di tipo J



- E' il formato usato per le istruzioni di salto incondizionato (*jump*):

op	indirizzo
6 bit	26 bit

- In questo caso, i campi hanno il seguente significato:
 - **op** indica il tipo di operazione;
 - **indirizzo** (composto da **26-bit**) riporta una parte (26 bit su 32) dell'indirizzo **assoluto** di destinazione del salto.
- I 26-bit del campo **indirizzo** rappresentano un indirizzo di parola (**word address**) 2^{26} parole = 256 Mbyte (segmento testo).

PC

		00
4 bit (invariati)	26 bit	2 bit

A.A. 2016-2017

14/36

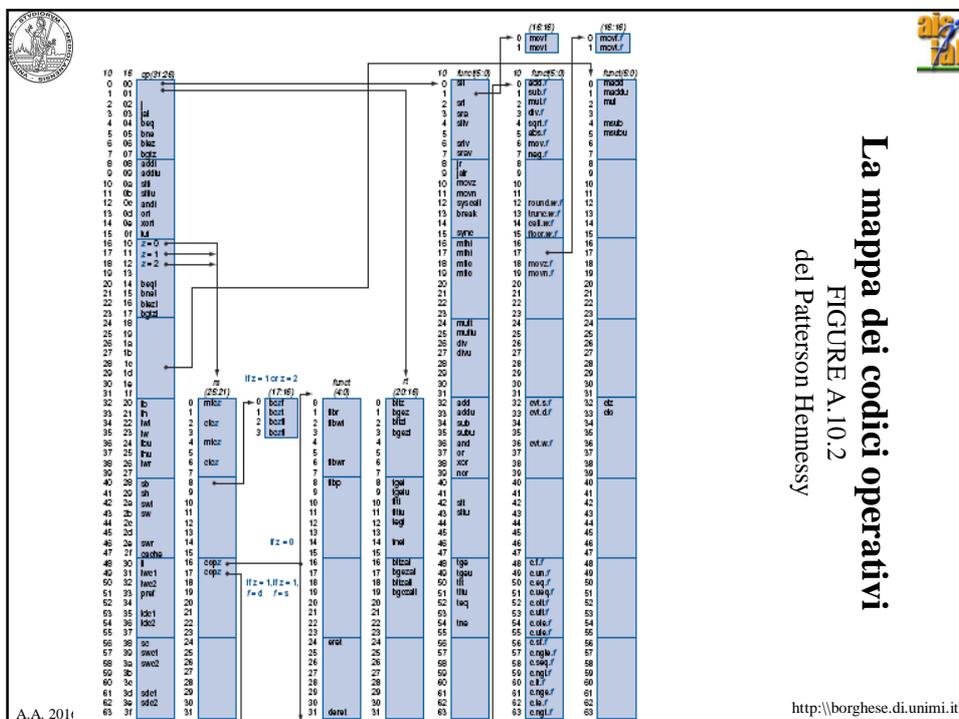
<http://borghese.di.unimi.it/>

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 / costante		
J	op					

A.A. 2016-2017
15/36
http://borghese.di.unimi.it/






Il linguaggio macchina

- I diversi tipi di istruzioni: Istruzioni di tipo I
- I diversi tipi di istruzioni: Istruzioni di tipo J
- **Register file**

A.A. 2016-2017

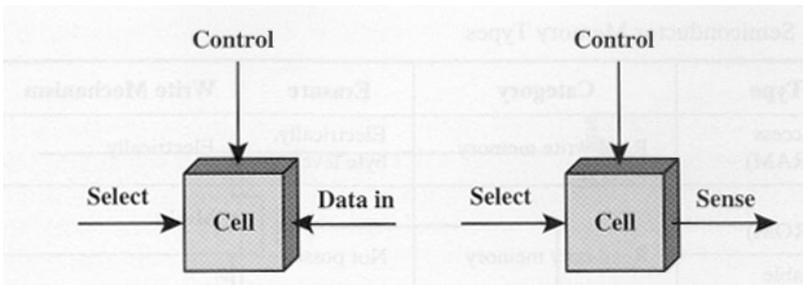
17/36

<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).

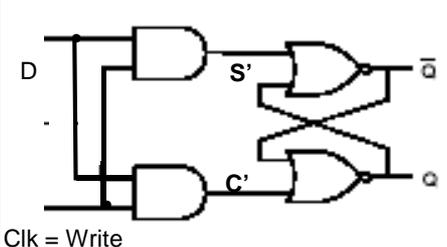
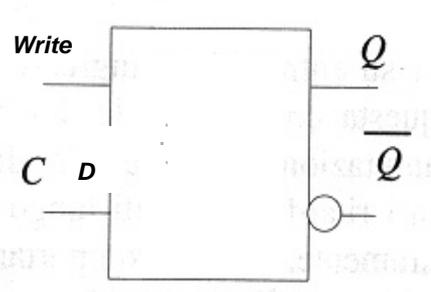
A.A. 2016-2017

18/36

<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. 2016-2017 19/36 <http://borghese.di.unimi.it/>

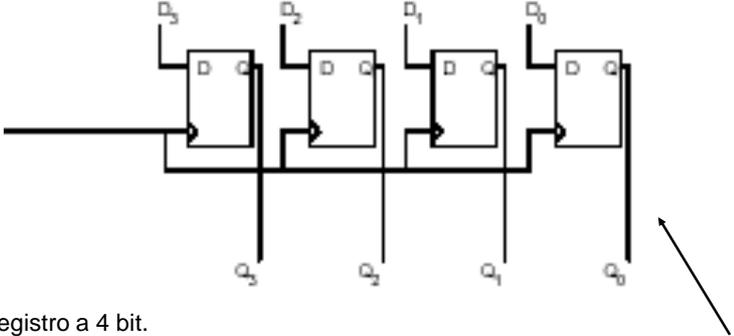
Registro



A.A. 2016-2017 20/36 <http://borghese.di.unimi.it/>




Registri



Un registro a 4 bit.
Memorizza 4 bit.

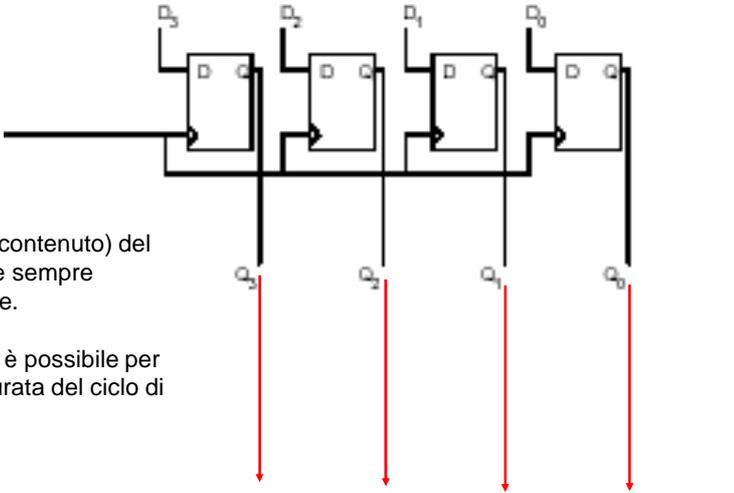
NB Non è un registro a scorrimento (shift register!)

Latch di tipo D

A.A. 2016-2017
21/36
<http://borghese.di.unimi.it/>




Lettura di un registro



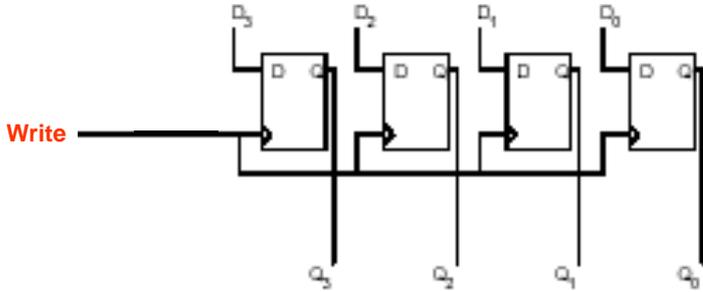
Lo stato (contenuto) del bistabile è sempre disponibile.

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

A.A. 2016-2017
22/36
<http://borghese.di.unimi.it/>

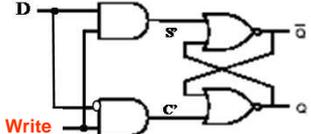
Scrittura di un registro

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



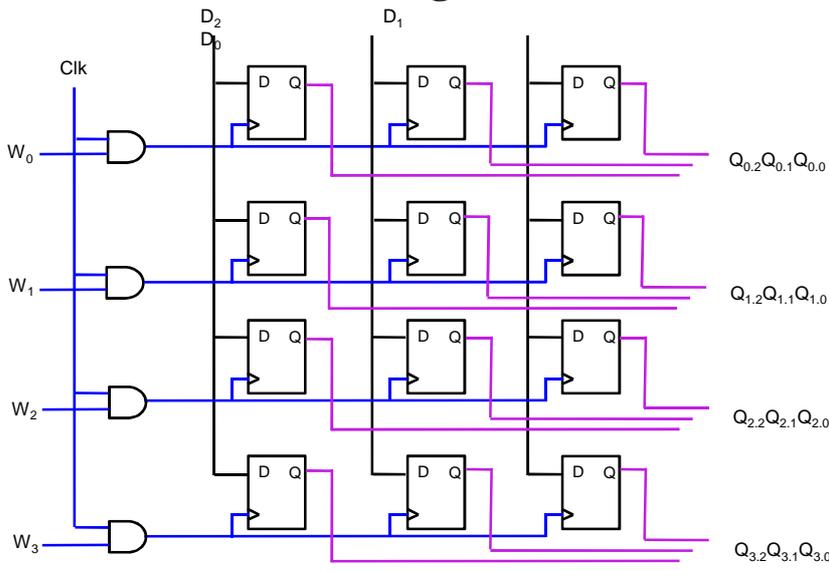
Cosa occorre modificare perchè il registro venga scritto quando serve?
 Introdurre una sorta di *"apertura del cancello (chiusura circuito)"*.
 Può essere sincronizzata o meno con il clock.

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

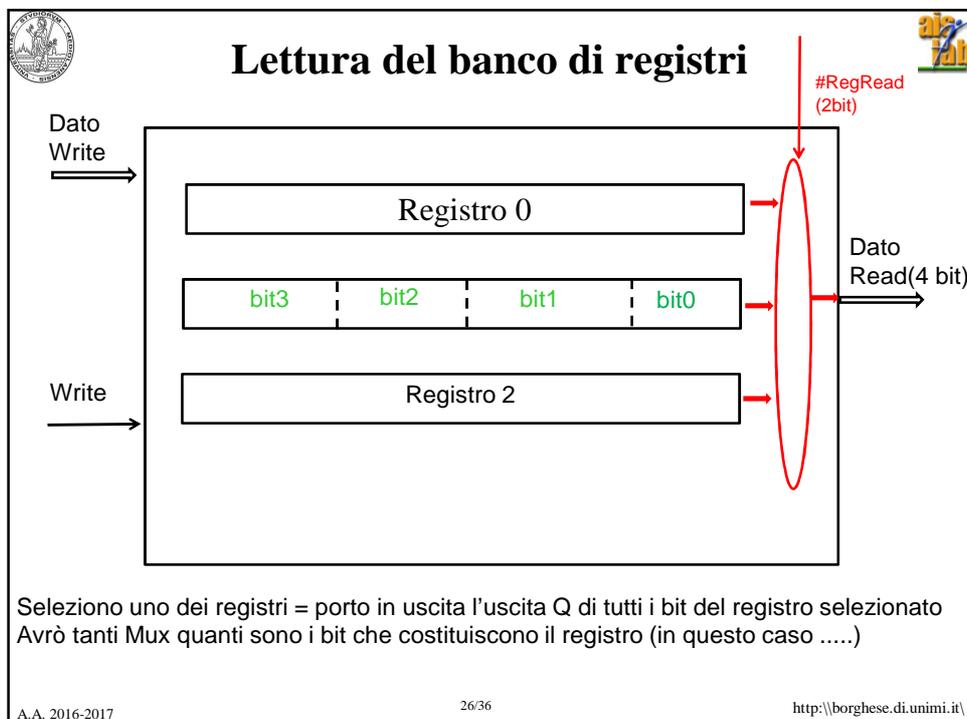
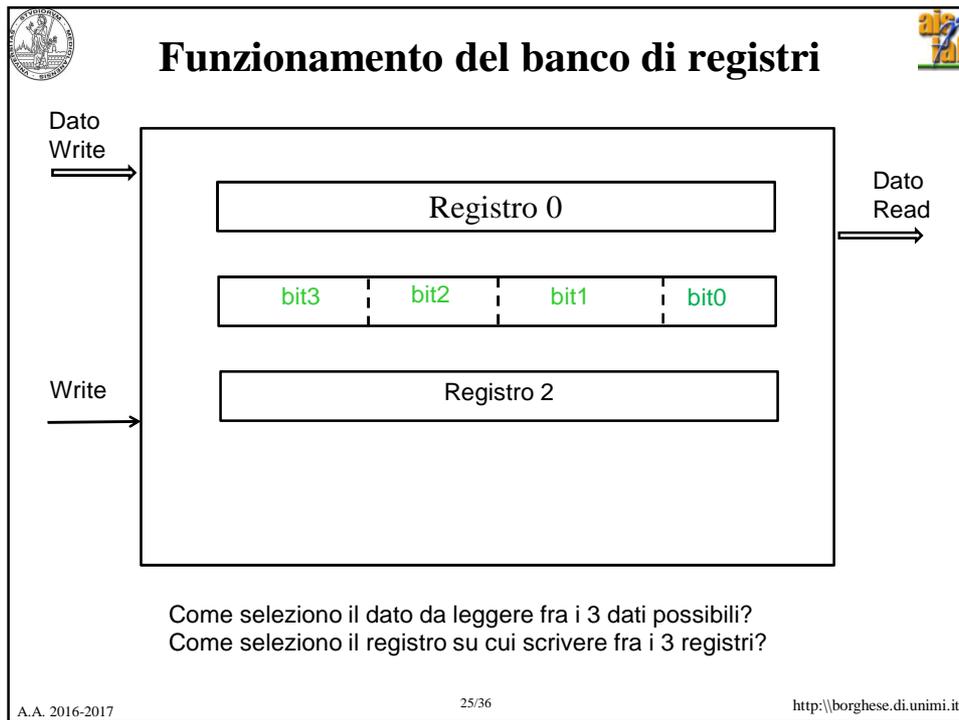


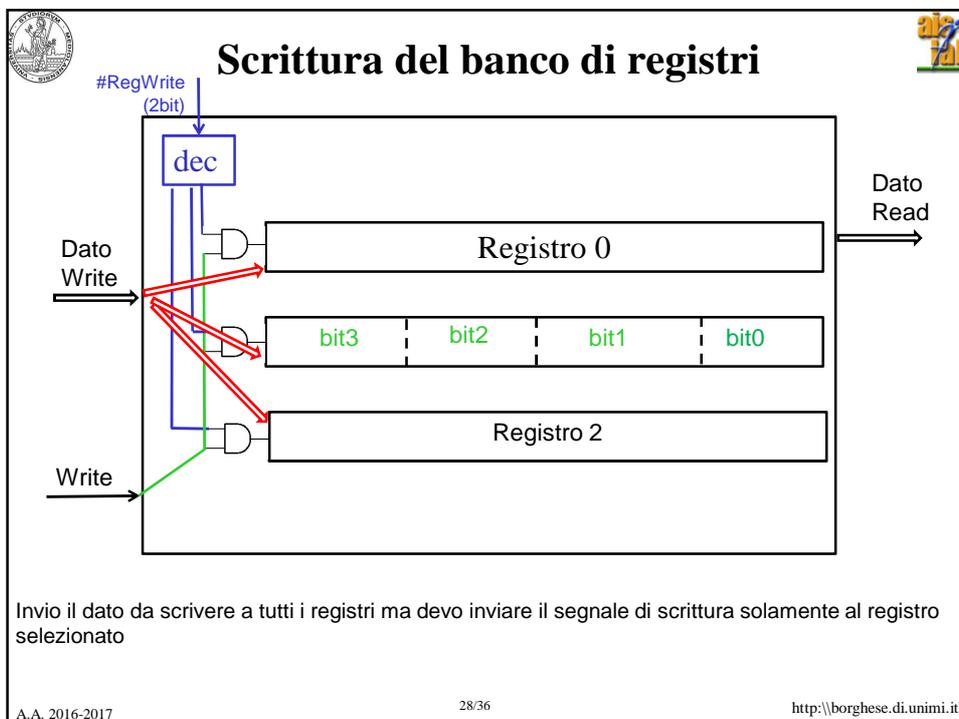
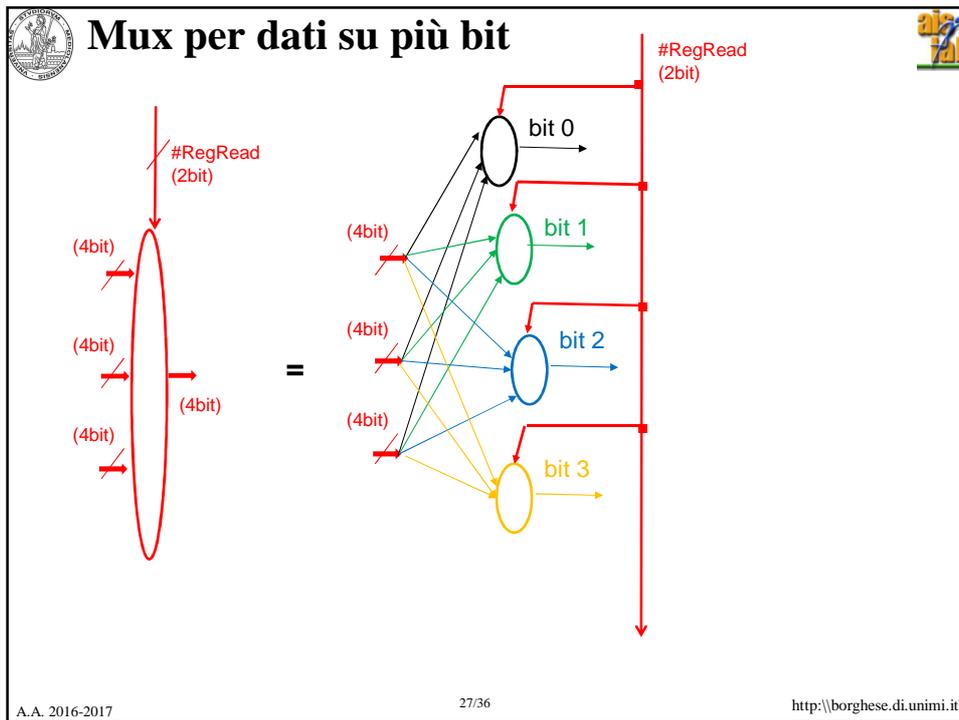
A.A. 2016-2017 23/36

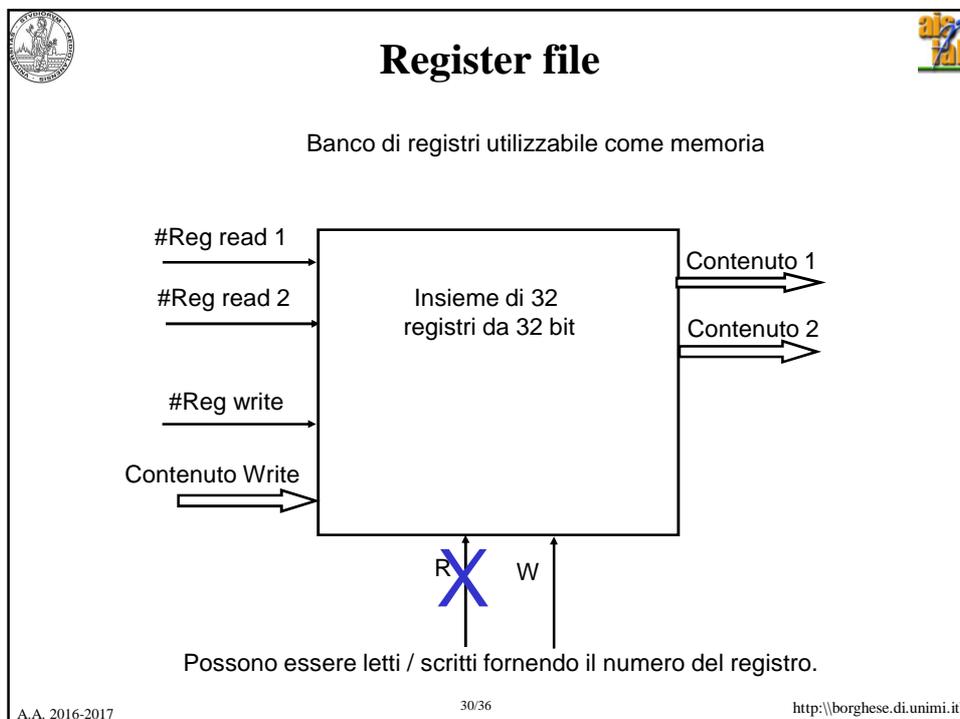
Un banco 4 registri x 3bit



A.A. 2016-2017 24/36 http://borghese.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. 2016-2017 31/36 <http://borghese.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. 2016-2017 32/36 <http://borghese.di.unimi.it/>

