



# Architettura degli elaboratori Linguaggio Macchina e Register File

Prof. Alberto Borghese  
Dipartimento di Informatica  
[Alberto.borghese@unimi.it](mailto:Alberto.borghese@unimi.it)

Università degli Studi di Milano

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

A.A. 2024-2025

1/49

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



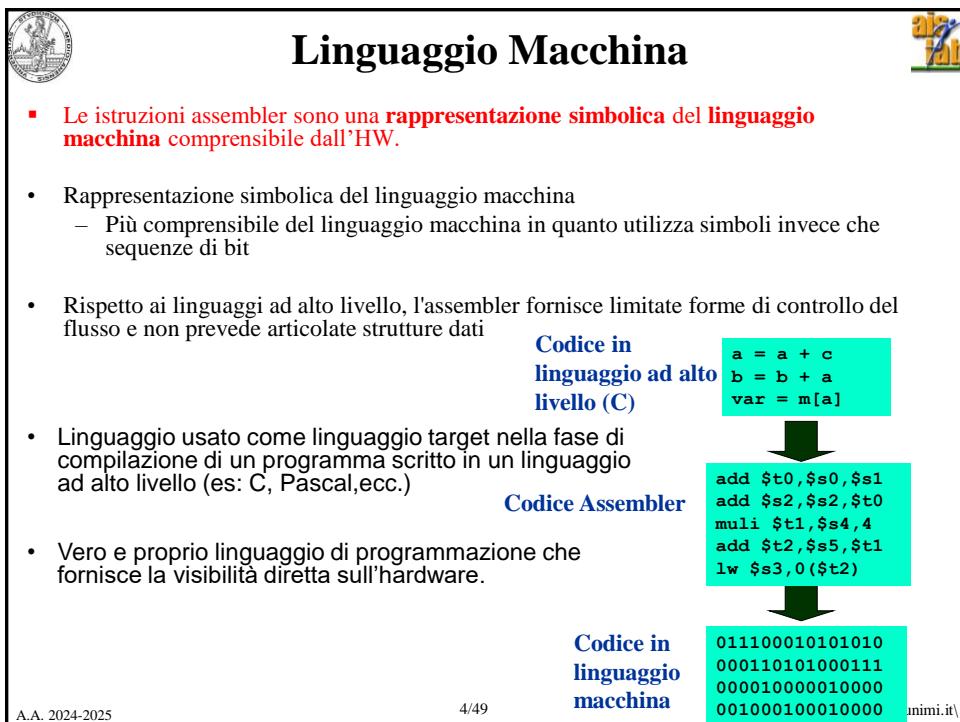
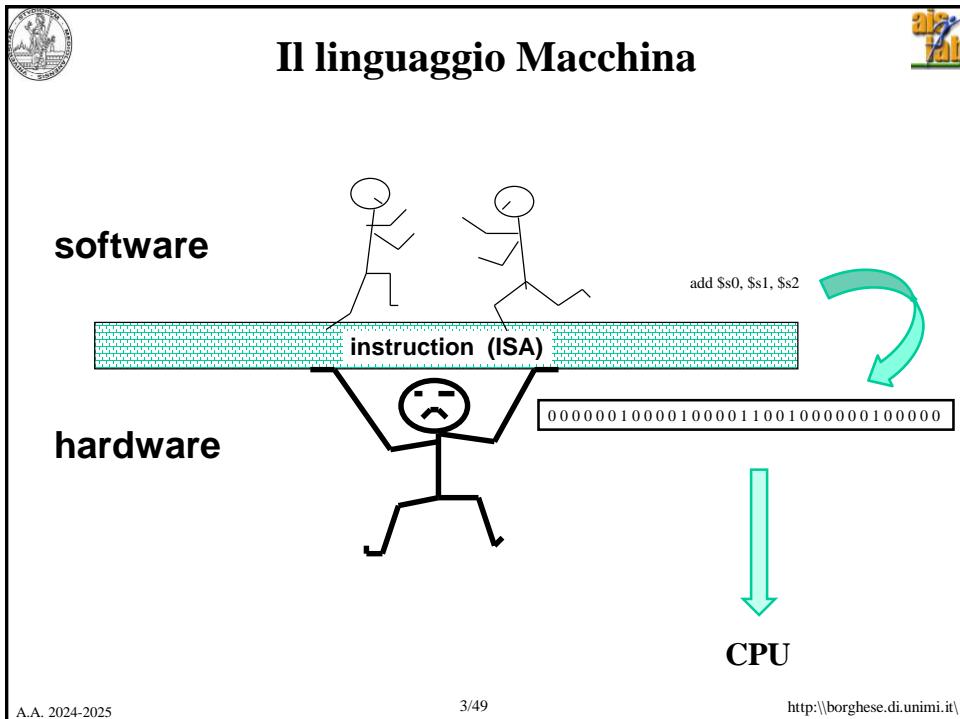
## Sommario

- Linguaggio macchina
- Formato R
- Formato I
- Formato J
- Il register file

A.A. 2024-2025

2/49

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





# Definizione di un'ISA (Instruction Set Architecture)

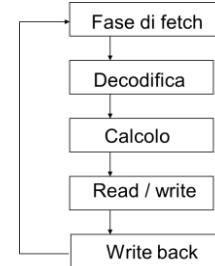


ISA: definisce le istruzioni messe a disposizione dalla macchina (in linguaggio macchina).

IS = Instruction Set = Insieme delle istruzioni. Non solo elenco ma anche:

*Definizione del funzionamento: ISA è interfaccia verso i linguaggi ad alto livello.*

- Tipologia di istruzioni.
- Meccanismo di funzionamento delle istruzioni.



*Definizione del formato: codifica delle istruzioni (ISA è interfaccia verso l'HW).*

- Formato delle istruzioni.
- Suddivisione in gruppi omogenei dei bit che costituiscono l'istruzione.
- Formato dei dati.

Le istruzioni devono contenere tutte le informazioni necessarie ad eseguire il ciclo di esecuzione dell'istruzione: registri, comandi, ....

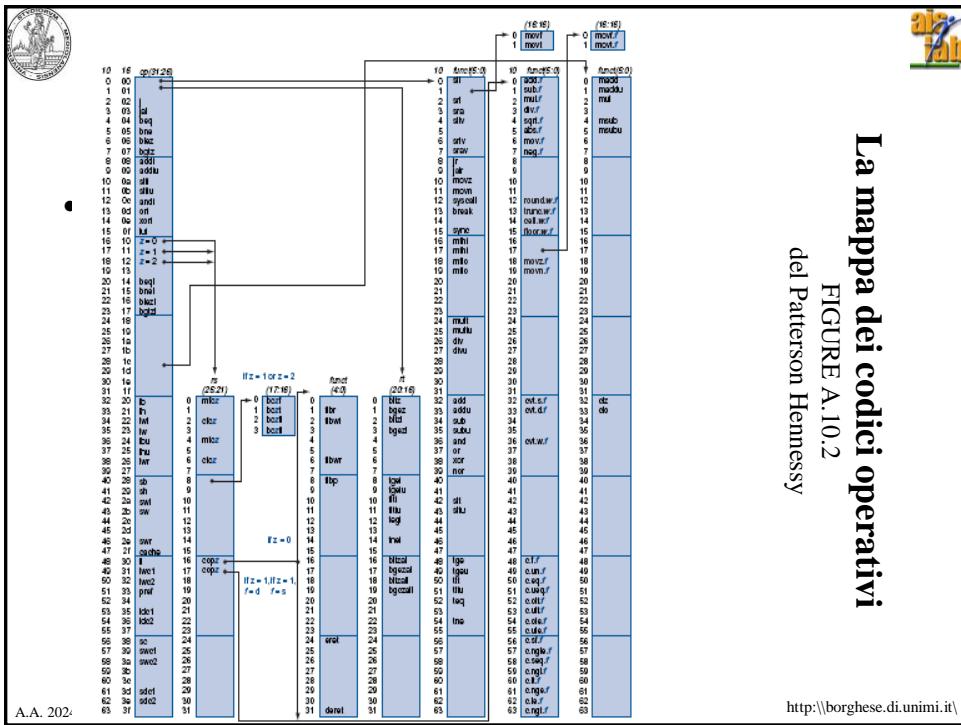


# Codifica delle istruzioni in linguaggio macchina



- Tutte le istruzioni MIPS hanno la **stessa dimensione (32 bit = 1 parola)** – 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 prevalentemente 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 con una costante.
  - **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			Indirizzo / costante		



## La mappa dei codici operativi

FIGURE A.10.2  
del Patterson Hennessy



## Sommario

- Linguaggio macchina
- Formato R
- Formato I
- Formato J
- Il register file

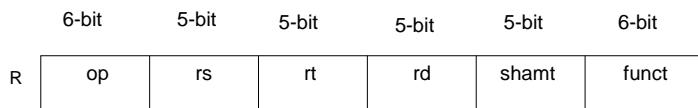


## Formato delle istruzioni di tipo R



Contiene:

- Un codice operativo su 6 bit
- Un registro source, rs, su 5 bit (32 registri)
- Un registro target, rt, su 5 bit (32 registri)
- Un registro destinazione, rd, su 5 bit (32 registri)
- Un numero di posizioni di shift (shift amount, shamt), su 5 bit
- Un codice di funzione (cf. selettore ALU), su 6 bit



## Istruzioni di tipo R: esempio

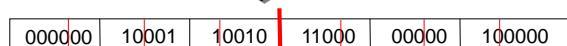


**add \$t8, \$s1, \$s2 # \$t8 = \$s1+\$s2**

Nome	Numero	Utilizzo
\$zero	0	costante zero
\$at	1	riservato per l'assemblatore
\$v0-\$v1	2-3	valori di ritorno di una procedura
\$a0-\$a3	4-7	argomenti di una procedura
\$t0-\$t7	8-15	registri temporanei (non salvati)
\$s0-\$s7	16-23	registri salvati
\$t8-\$t9	24-25	registri temporanei (non salvati)
\$k0-\$k1	26-27	gestione delle eccezioni
\$gp	28	puntatore alla global area (dati)
\$sp	29	stack pointer
\$s8	30	registro salvato (fp)
\$ra	31	indirizzo di ritorno



6-bit      5-bit      5-bit      5-bit      5-bit      6-bit



0x0232A020



## Istruzioni di tipo R: esempi

Nome campo	Op	Rs	rt	rd	Shamt	Funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
<b>add \$t8, \$s1, \$s2</b>	000000	10001	10010	11000	00000	100000
Nome campo	op	rs	rt	rd	shamt	funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
<b>sub \$t8, \$s1, \$s2</b>	000000	10001	10010	11000	00000	100010
Nome campo	op	rs	rt	rd	shamt	funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
<b>and \$s1, \$s2, \$s3</b>	000000	10010	10011	10001	00000	100100
Nome campo	op	rs	rt	rd	shamt	funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
<b>sll \$s1, \$s2, 3</b>	000000	X	10010	10001	00011	000000
$\$s1 = \$s2 * 2^3$ Se \$s2 contiene 20 (0000....0010100) => \$s1 conterrà = $20 * 2^3 = 160$ (0000....0010100000)						
Nome campo	op	rs	rt	rd	shamt	funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
<b>srl \$s1, \$s2, 3</b>	000000	X	10010	10001	00011	000010
$\$s1 = \$s2 / 2^3$						

A.A. 2024-2025

11/49

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

## Sommario

- Linguaggio macchina
- Formato R
- Formato I
- Formato J
- Il register file

A.A. 2024-2025

12/49

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



## Formato istruzioni di tipo I



I	op	rs	rt	costante / indirizzo
	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 / indirizzo**. Nel caso di una lw riporta lo spiazzamento (offset) nel segmento dati, nel caso di una beq riporta lo spiazzamento nel segmento testo.

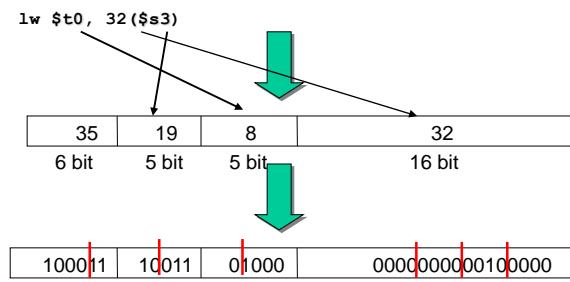


## Istruzioni di tipo I: accesso a memoria



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

$$\text{indirizzo} = \text{indirizzo\_base} + \text{offset} (= \text{costante})$$



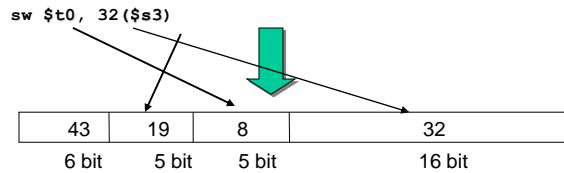
La costante può essere positiva o negativa



## Istruzioni di tipo I: accesso memoria

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

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



Differenza di 1 bit (distanza di Manhattan pari a 1) ->  
cambia la direzione del trasferimento con la memoria.



## Istruzioni di tipo I: aritmetico-logiche

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

Nome campo	op	rs	rt	costante
Dimensione	6-bit	5-bit	5-bit	16-bit
<b>andi \$t0, \$s3, 64</b>	001100	10011	01000	0000 0000 0100 0000

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

# \$t0 = 1 if \$s2 < 8

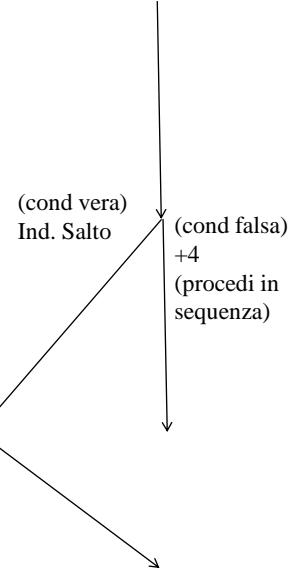
NB ruolo del registro target: nelle istruzioni di addi e di lw rappresenta **dove nel register file vado a scrivere**. Nelle istruzioni di tipo R e di sw contiene **uno dei dati sorgente**.



## Istruzioni di tipo I: salto condizionato



- Salti condizionati relativi:
  - breq 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 (**breq**)
  - 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.



I	op	rs	rt	indirizzo
	6 bit	5 bit	5 bit	16 bit

A.A. 2024-2025

17/49

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

## Allargamento dello spazio di indirizzamento relativo



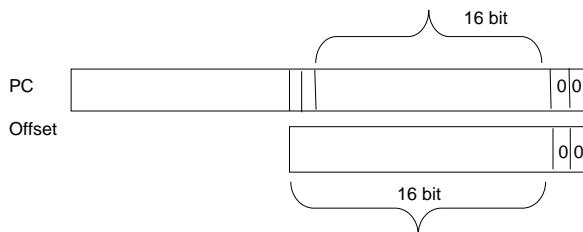
0000	0	0
0100	1	4
1000	2	8
1100	3	12

Gli offset in Byte avranno sempre **00 come ultimi 2 bit** perché gli indirizzi sono multipli di 4.

**Calcolo lo spiazzamento in numero di parole invece che di Byte.**

Considero 64 Kword (64K istruzioni) invece di 64Kbyte. Lo spazio indirizzabile all'interno del segmento di testo è di  $64\text{Kword} * 4 = 256\text{Kbyte}$ .

Moltiplicare per 4 vuol dire operare uno shift a sinistra di due posizioni dell'offset



**La costante su 16 bit, contenuta nel codice, rappresenta l'offset in termini di numero di istruzioni**

A.A. 2024-2025

18/49

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



## Calcolo dell'indirizzo di salto



```
t1 = 10; t0 = 0;           0x0000      addi $t1,$zero,10 # t1=10
                           0x0004      addi $t0,$zero,0  # t0 =0;

Repeat {          t0++;    0x0008  loop:   addi $t0,$t0,1  # t0++
                           0x000C      ....
...
} until (t0 == t1)        0x0020      bne $t0,$t1,loop
```

Quanto vale il campo costante da inserire nella stringa dell'istruzione bne?

A.A. 2024-2025

19/49

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

## Calcolo dell'indirizzo di salto



Ind	Ind	Istruzione	# istruzione
Dec	Exadec		
000	0x0000	addi \$t1,\$zero,10 # N=10	0
004	0x0004	addi \$t0,\$zero,0 # i=0;	1
008	0x0008	loop: addi \$t0,\$t0,1 # i++	2
012	0x000C	....	3
032	0x0020	bne \$t0,\$t1,loop	9
036	0x0024		10

L'indirizzo di destinazione è 0x8 (indirizzo dell'etichetta loop)

Lo spiazzamento del salto in byte è pari a:  $(0x8 - 0x24) = (8-36) = -28$  Byte  
 Lo spiazzamento del salto in numero di istruzioni è pari a -7 istruzioni

$$\text{Prova: Indirizzo di salto} = \text{Indirizzo PC+4} + \text{Offset (#istruzioni)} * 4 \\ \text{loop} \Rightarrow 8 = 36 (24_{16}) + -7 * 4$$

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

A.A. 2024-2025

20/49

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



## Istruzioni di tipo I – Branch (esempi)



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

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

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

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

Salto di 24 istruzioni in avanti o di 26 istruzioni indietro



## Osservazioni



Nome campo	op	rs	rt	Indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
<b>beq \$s1, \$s2, Label</b>	000100	10001	10010	0000 0000 0001 1000

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
<b>lw \$s2, 24(\$s1)</b>	100011	10001	10010	0000 0000 0001 1000

Nome campo	op	rs	rt	costante
Dimensione	6-bit	5-bit	5-bit	16-bit
<b>addi \$s2,\$s1,24</b>	001000	10001	10010	0000 0000 0001 1000

Istruzioni molto diverse possono distare pochi bit una dall'altra.



## Formato R e 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.



## Sommario



- Linguaggio macchina
- Formato R
- Formato I
- **Formato J**
- Il register file

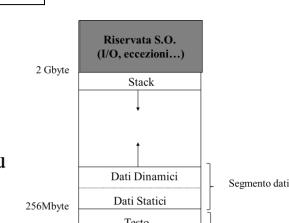
 **Formato istruzioni di tipo J** 

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

j	indirizzo
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).



The diagram illustrates the memory segmentation structure. It shows a vertical stack of memory segments. From bottom to top, the segments are labeled: Riservata S.O. (I/O, eccezioni...), Stack, Dati Dinamici, Dati Statici, Testo, and Riservata S.O. The total memory size is indicated as 2 Gbyte at the top. Arrows point from the labels "Segmento dati" and "Segmento testo" to their respective segments in the stack. Address markers 0, 4Mbyte, 256Mbyte, and 2 Gbyte are placed along the left side of the stack.

PC
4 bit  
(invariati)
26 bit
2 bit

$0 \leq \text{indirizzo} < 2^{28} = 256 \text{ Mbyte}$  (**segmento testo!**)

A.A. 2024-2025      25/49      <http://borgheze.di.unimi.it/>

The diagram illustrates the logical organization of memory based on a 26-bit address. The memory space is divided into several segments:

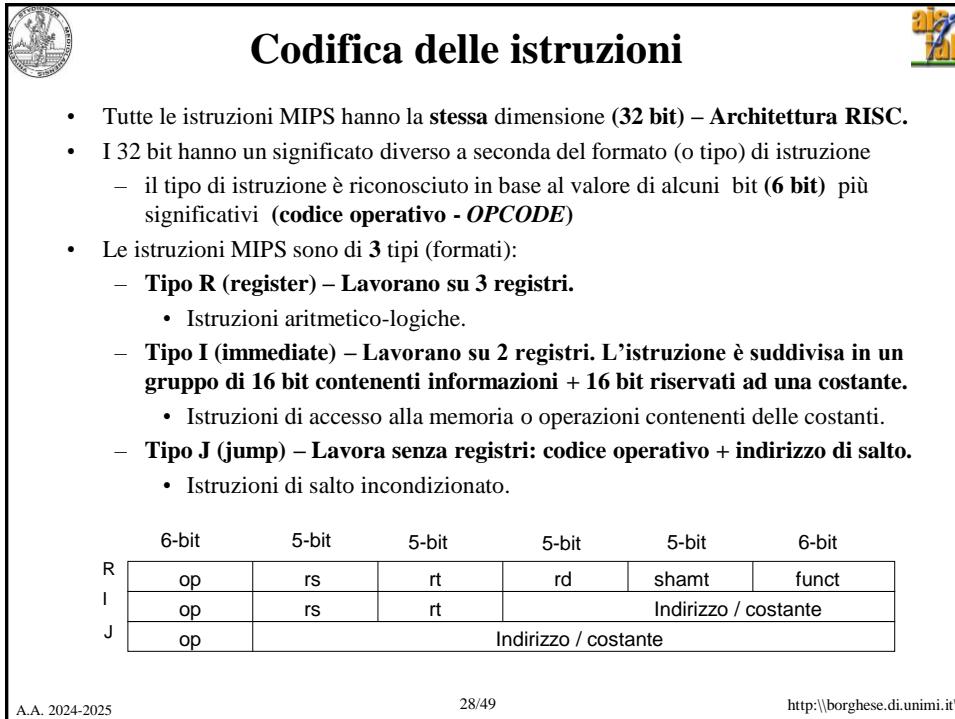
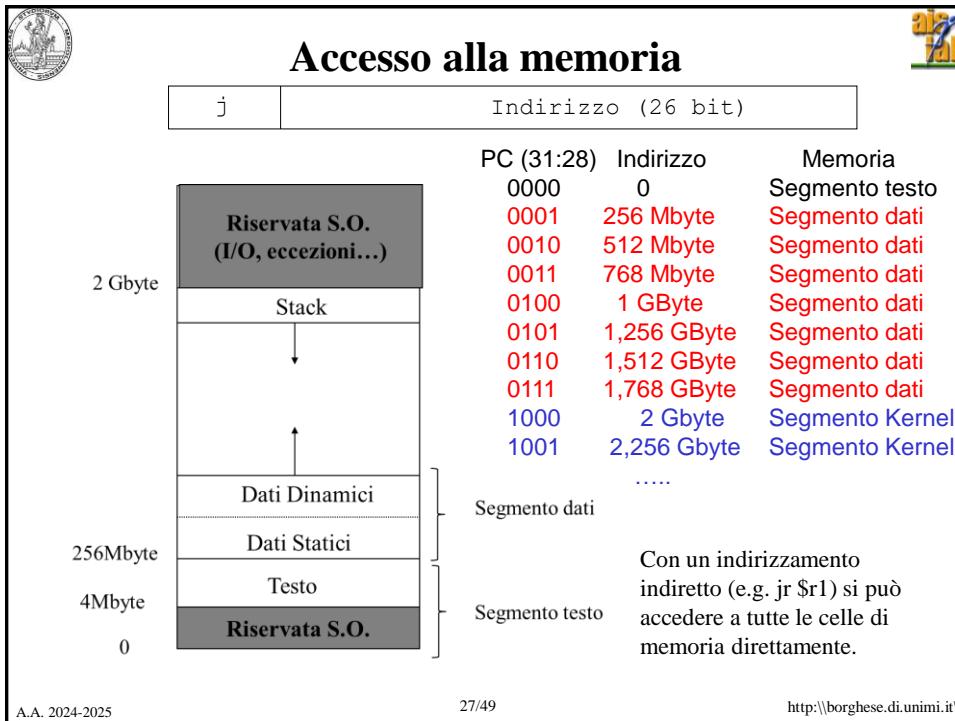
- Indirizzo (26 bit)**: The address is shown in a box at the top left.
- Riservata**: A reserved area at the top of the stack.
- Stack**: The stack grows downwards from the reserved area.
- Dati Dinamici**: Dynamic data segment.
- Dati Statici**: Static data segment.
- Testo**: Text segment.
- Riservata S.O.**: Reserved area for the operating system at the bottom.

On the left, it is noted that the maximum addressable space is 2<sup>32</sup> = 4Gbyte. On the right, the memory is divided into two main segments:

- Segmento dati**: Contains the dynamic, static, and text segments, spanning from address 10000000<sub>16</sub> down to 400000<sub>16</sub>.
- Segmento testo**: Contains the reserved S.O. area from address 400000<sub>16</sub> down to 0.

Below the stack, there is a note: "Max spazio di indirizzamento su 32 bit è di  $2^{32} = 4\text{Gbyte}$ ".

On the far left, it is indicated that the 28-bit index register has a value of  $2^{28} = 256\text{Mbyte}$ , with the circled  $2^{28}$  highlighted.





## Formati e tipi di istruzioni



Tipi di istruzioni di un'ISA

- Istruzioni aritmetiche
- Istruzioni di accesso a memoria
- Istruzioni di controllo di flusso
- (Istruzioni di I/O)

Formati

- R - Registro
- I – Immediato
- J - Jump (salto)

Non c'è corrispondenza 1 a 1 tra tipi di istruzioni e formati



## Sommario



- Linguaggio macchina
- Formato R
- Formato I
- Formato J
- Il register file



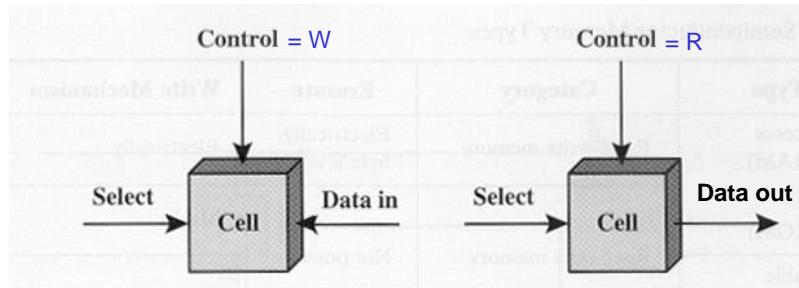
## Cella di memoria



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 oppure Data out (sense)

Base



## Latch sincrono come elemento di memoria

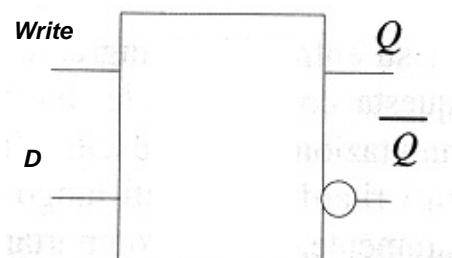
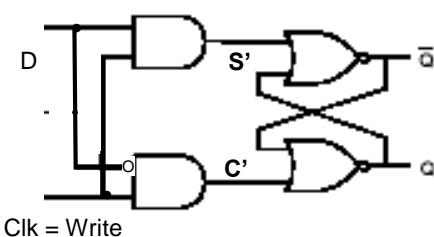


L'ingresso "clock" del bistabile viene utilizzato come segnale di write

E' trasparente quando Write = 1

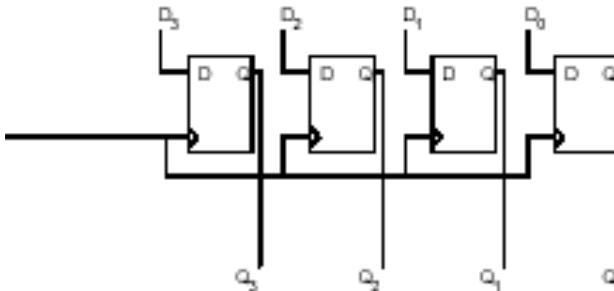
Se Write = 1  $Q_{t+1} = D$

Se Write = 0  $Q_{t+1} = Q_t$



## Registri



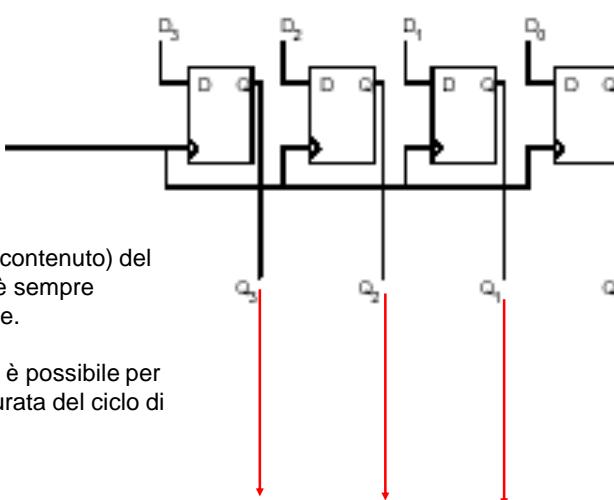
Un registro a 4 bit.  
Memorizza 4 bit.  
Tutti gli elementi di memoria ricevono lo stesso segnale di write

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

A.A. 2024-2025      33/49      <http://borghese.di.unimi.it/>

## Lettura di un registro



Lo stato (contenuto) del bistabile è sempre disponibile.

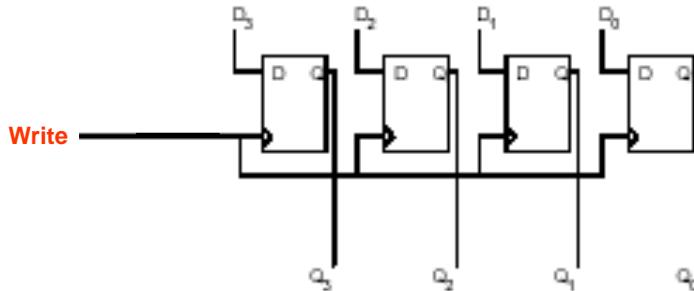
La lettura è possibile per tutta la durata del ciclo di clock.

A.A. 2024-2025      34/49      <http://borghese.di.unimi.it/>



## Scrittura di un registro

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



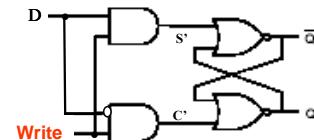
Cosa occorre modificare perché il registro venga scritto quando serve?

Introdurre una sorta di “*apertura del cancello (chiusura circuito)*”.

Può essere sincronizzata o meno con il clock.

3

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



A.A. 2024-2025

35/49



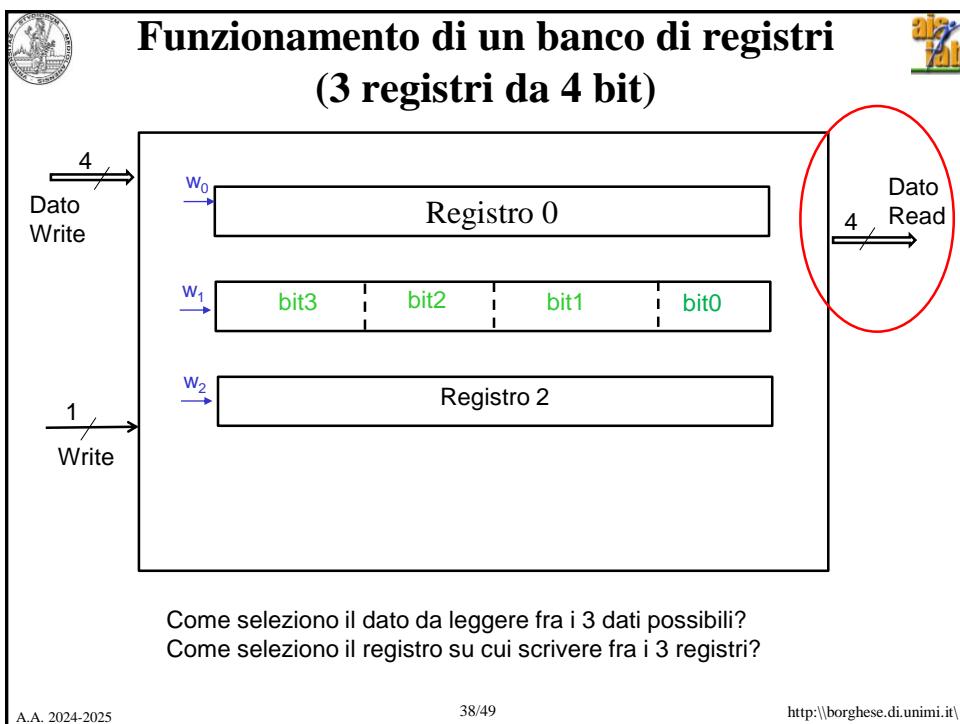
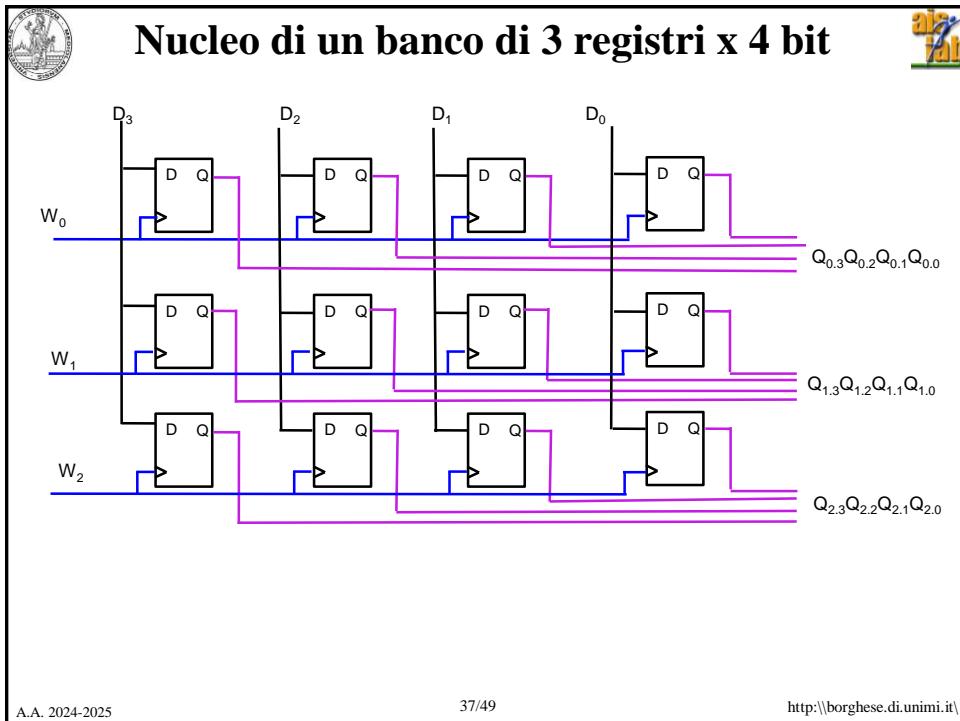
## Registro

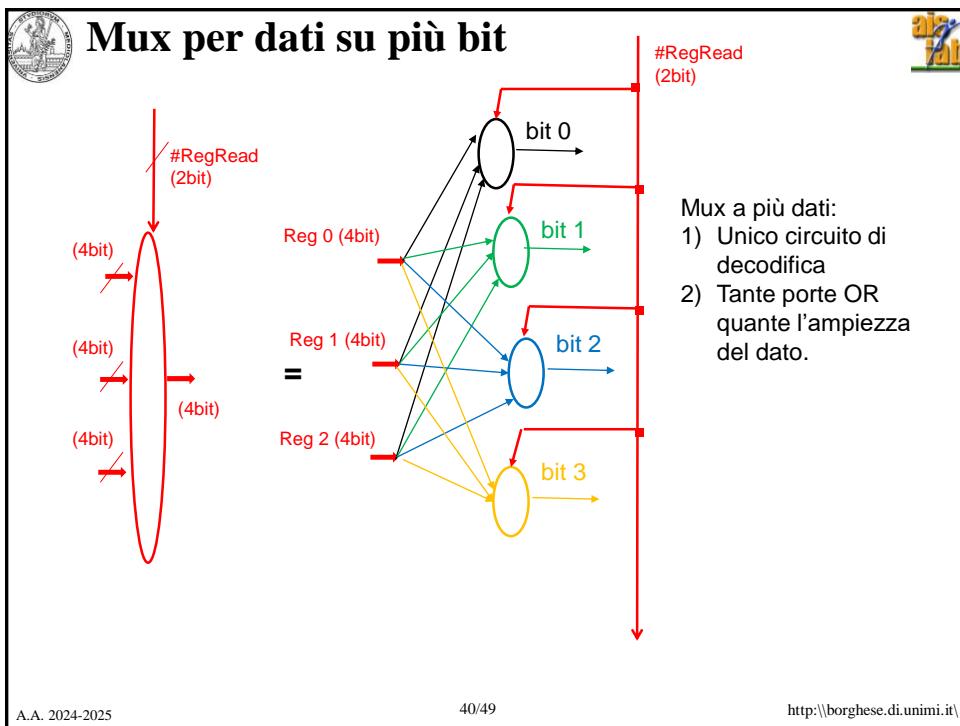
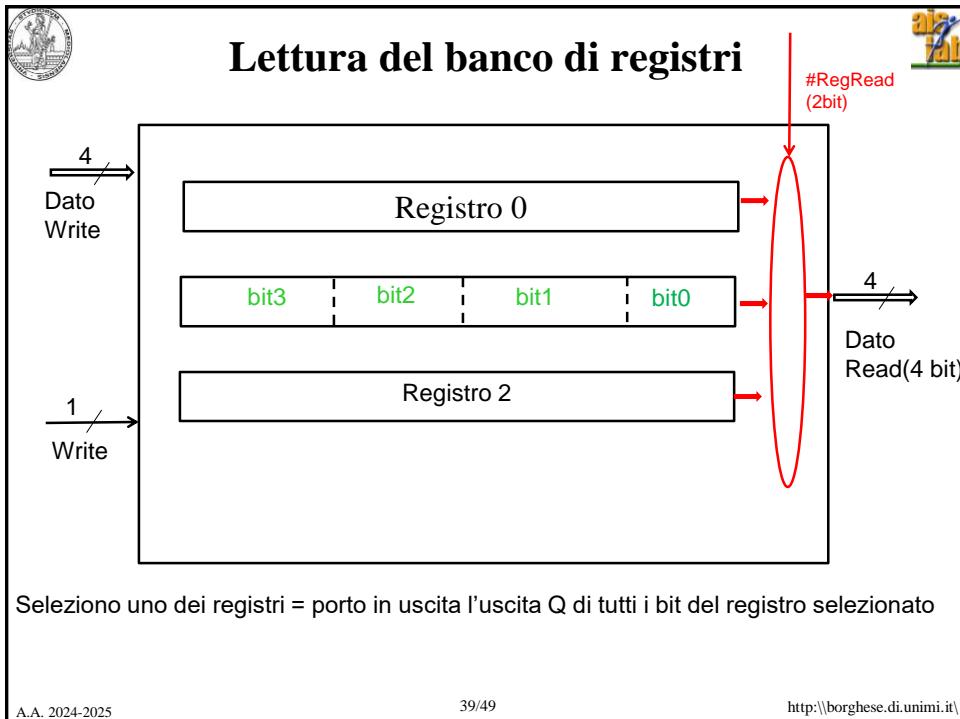


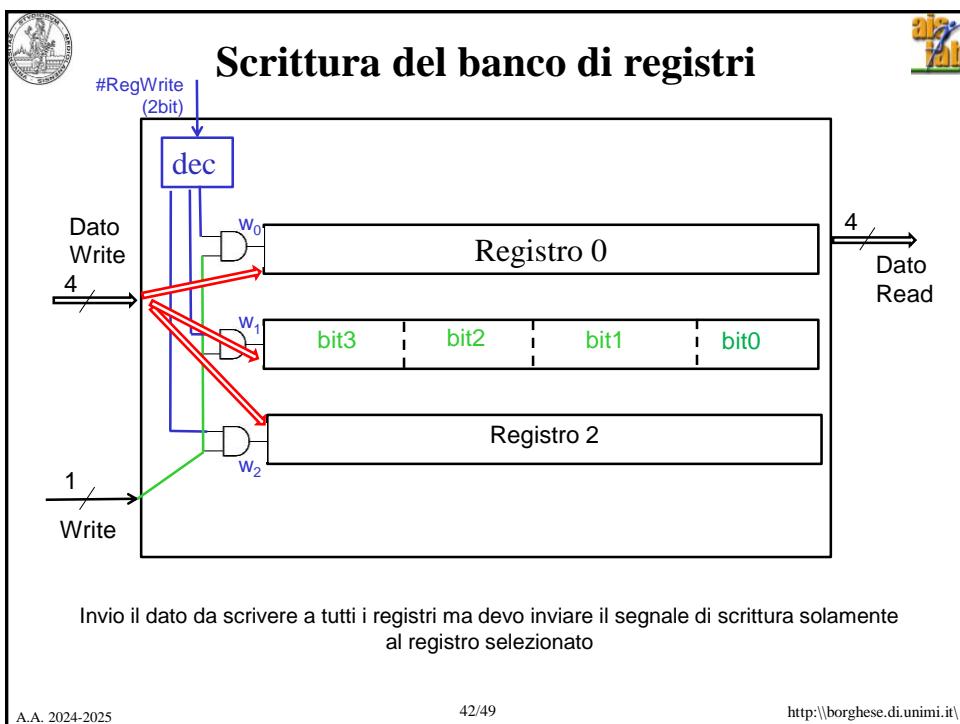
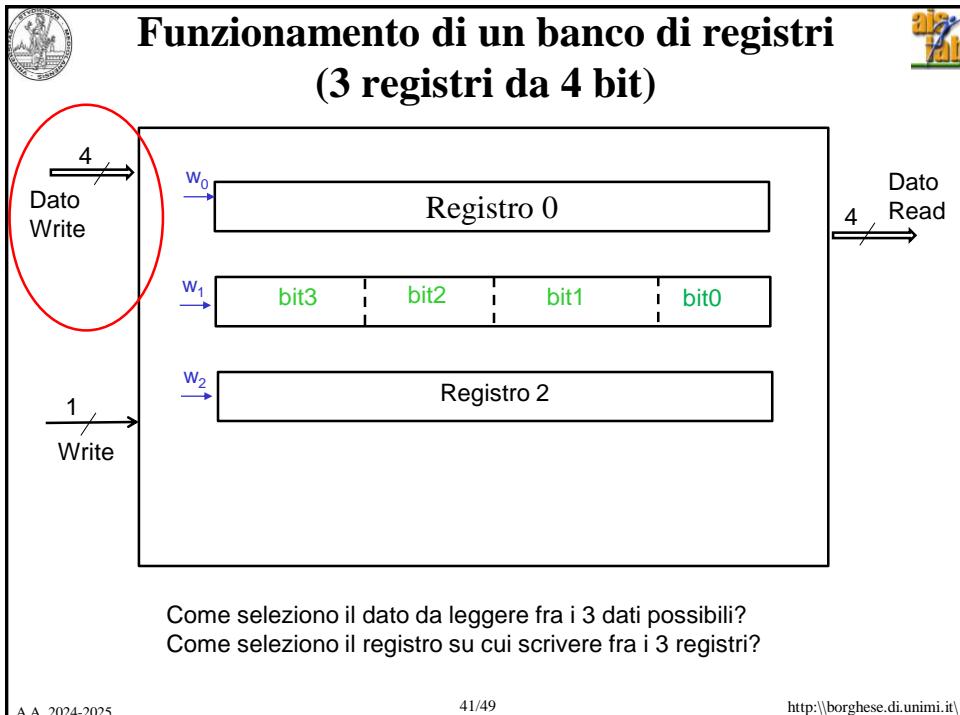
A.A. 2024-2025

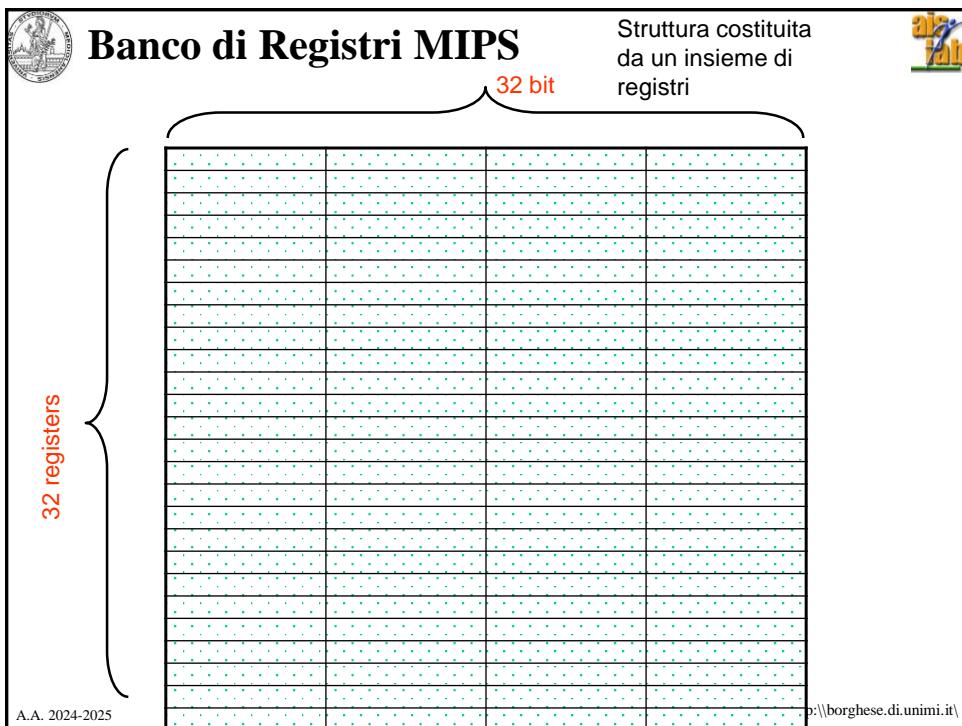
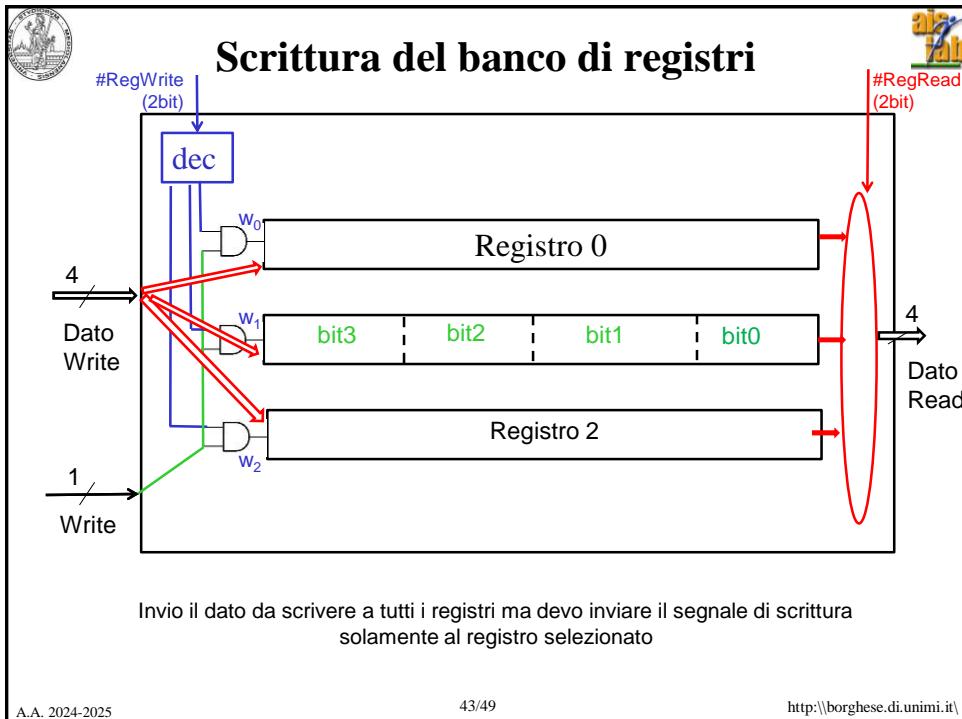
36/49

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







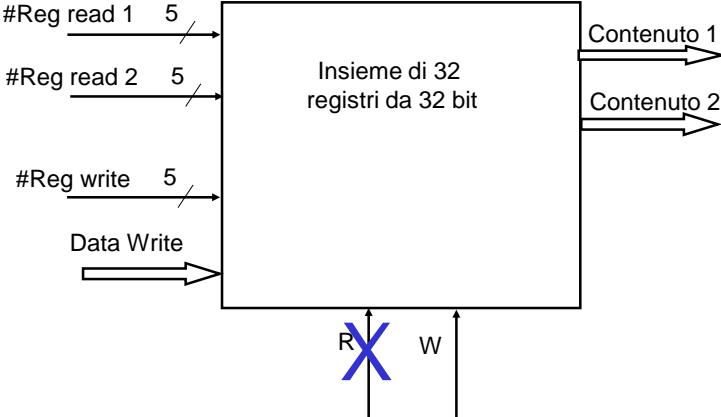


## Register file

Banco di registri utilizzabile come memoria:

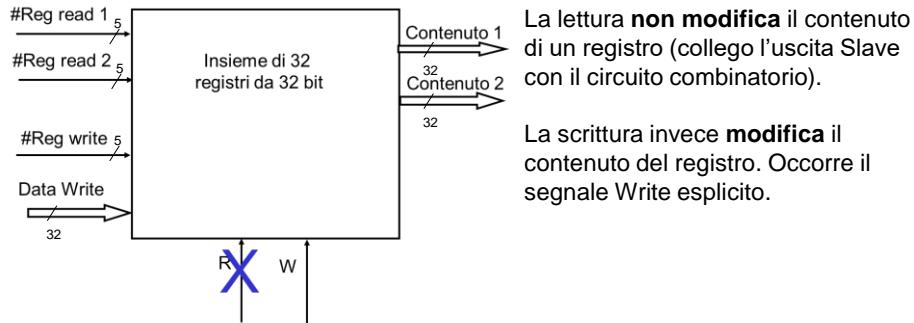
- 2 porte di lettura
- 1 porta di scrittura



Possono essere letti fornendo il numero del registro in lettura (uscita)  
Possono essere scritti fornendo il numero di registro in scrittura (ingresso) + segnale di write [animi.mi.it/](http://animi.mi.it/)

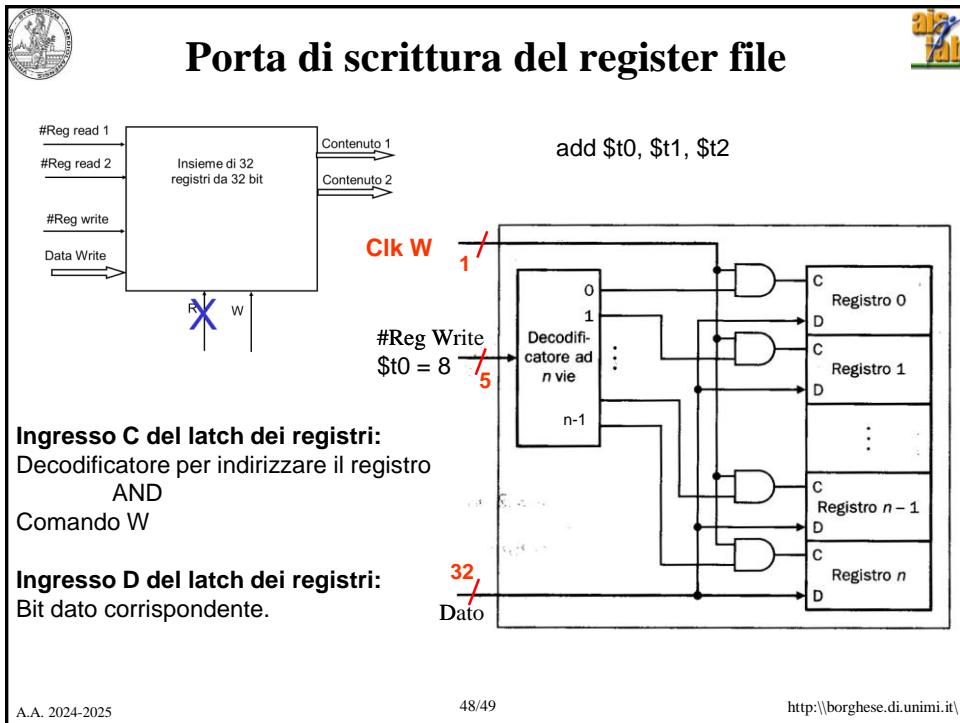
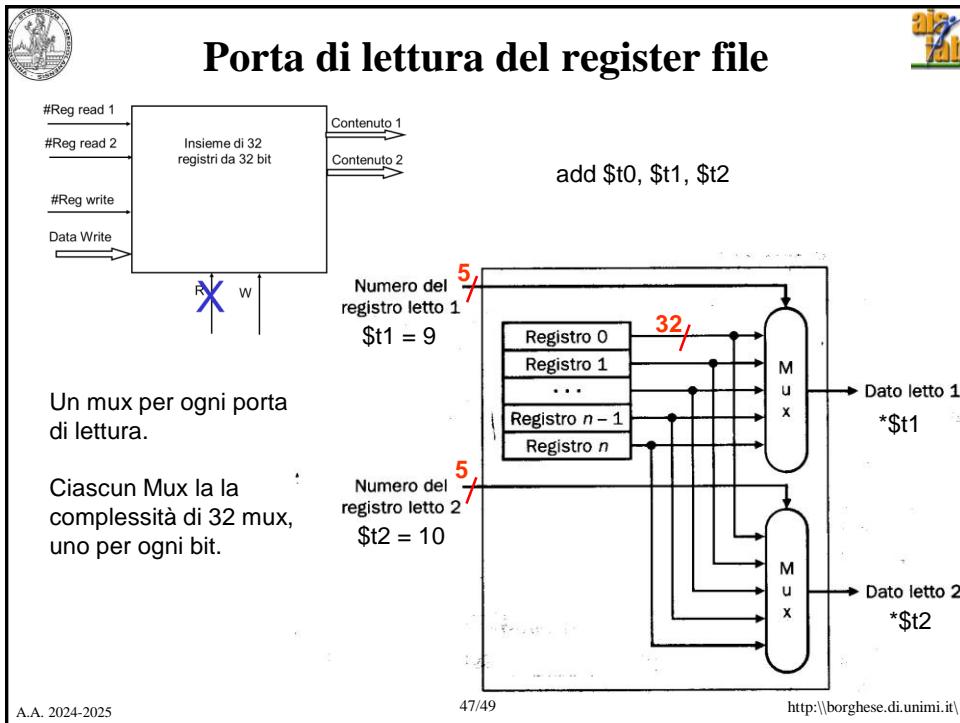
 

## Gestione del register file



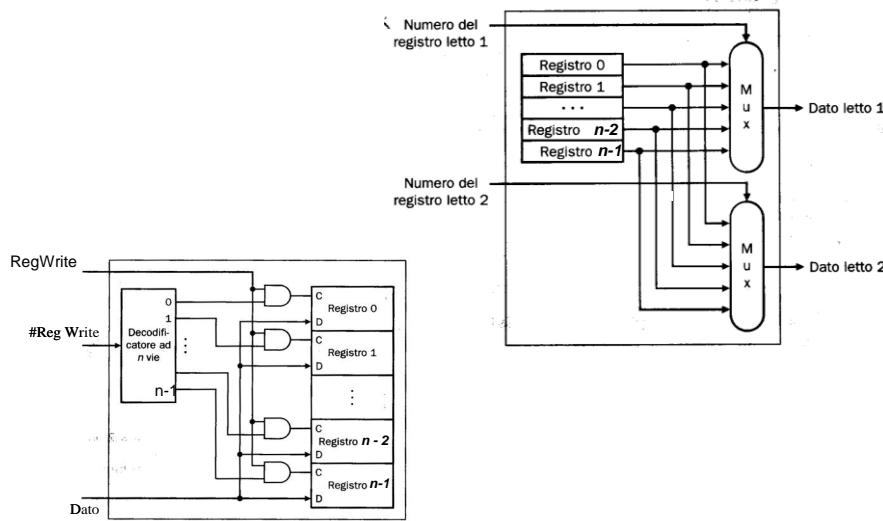
#bit<sub>indirizzamento</sub> = log<sub>2</sub> #bit

A.A. 2024-2025      46/49      <http://borgheze.di.unimi.it/>





## Register file del MIPS



A.A. 2024-2025

49/49

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

## Sommario



- Linguaggio macchina
- Formato R
- Formato I
- Formato J
- Il register file

A.A. 2024-2025

50/49

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