



# La struttura delle istruzioni elementari: il linguaggio Macchina

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

Università degli Studi di Milano



## Sommario

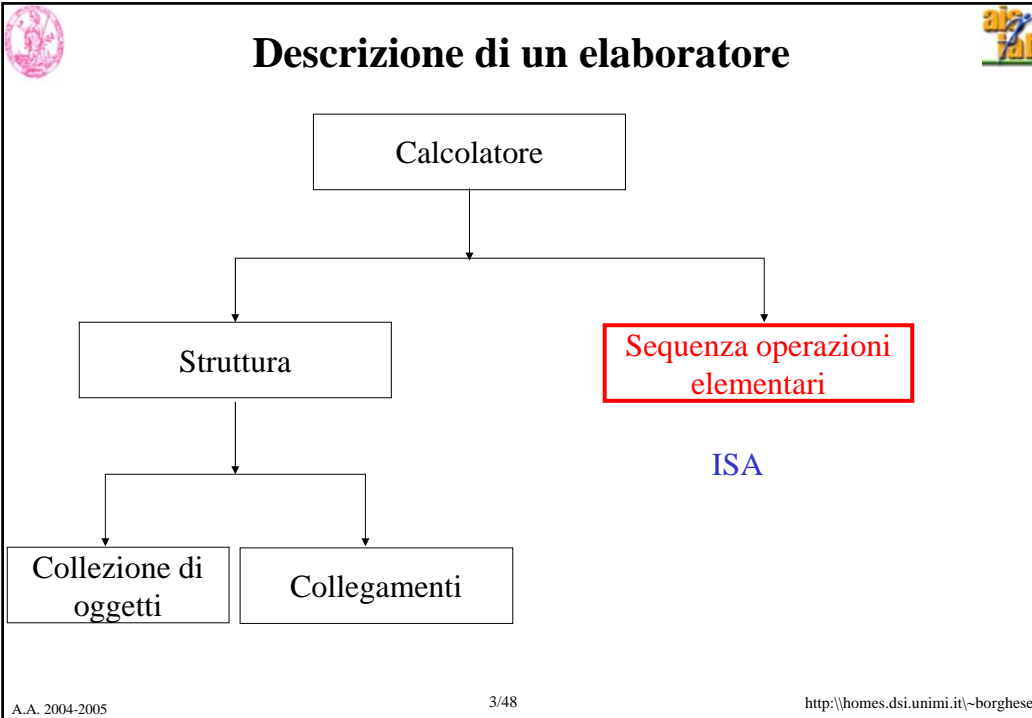
### Il linguaggio macchina: le istruzioni di tipo R

Le istruzioni di tipo I

Le istruzioni di tipo J

Modalità di indirizzamento

Trattamento delle costanti



## Linguaggio macchina

- Le istruzioni in linguaggio assembly devono essere tradotte in linguaggio macchina (cioè in sequenze di 0 e 1) per poter essere eseguite.
- Le istruzioni in linguaggio macchina sono lunghe **32 bit** (come i registri e le parole di memoria).
- Il parsing di un'istruzione in linguaggio macchina viene fatta dalla CPU che ricava le informazioni necessarie all'esecuzione dell'istruzione stessa nella fase di decodifica.
- Occorre definire l'architettura delle istruzioni:
  - Come vengono raggruppati i bit?*
  - Cosa viene rappresentato nel singolo bit?*
- **Sono definiti tre tipi di istruzione: tipo R (Register), tipo I (Immediate), tipoJ (Jump).**

A.A. 2004-2005 4/48 http://homes.dsi.unimi.it/~borghese



## Formato istruzioni di tipo R



**Campo di un'istruzione:** numero di bit consecutivi contenenti un'informazione per l'esecuzione.

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

- Ai vari campi sono stati assegnati dei nomi mnemonici:
  - **op**: (opcode) identifica il tipo di istruzione
  - **rs**: registro contenente il primo operando sorgente
  - **rt**: registro contenente il secondo operando sorgente (target)
  - **rd**: registro destinazione contenente il risultato
  - **shamt**: shift amount (scorrimento)
  - **funct**: indica la **variante** specifica dell'operazione



## MIPS: Software conventions for Registers



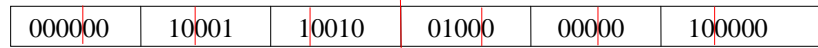
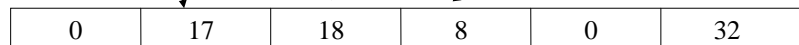
0 <b>zero</b> constant 0	16 <b>s0</b> callee saves
1 <b>at</b> reserved for assembler	... (caller can clobber)
2 <b>v0</b> expression evaluation &	23 <b>s7</b>
3 <b>v1</b> function results	24 <b>t8</b> temporary (cont'd)
4 <b>a0</b> arguments	25 <b>t9</b>
5 <b>a1</b>	26 <b>k0</b> reserved for OS kernel
6 <b>a2</b>	27 <b>k1</b>
7 <b>a3</b>	28 <b>gp</b> Pointer to global area
8 <b>t0</b> temporary: caller saves	29 <b>sp</b> Stack pointer
... (callee can clobber)	30 <b>fp</b> frame pointer ( <b>s8</b> )
15 <b>t7</b>	31 <b>ra</b> Return Address (HW)



## Istruzioni di tipo R: esempio



```
add $t0, $s1, $s2
```



0x02324020



## Istruzioni di tipo R



- Istruzioni aritmetico-logiche con il tipo di formato visto, vengono chiamate di **tipo R** (registro).
- Esempi:
  - somma, prodotto, divisione
  - shift (scorrimento)
  - AND, OR, NOT
- Le diverse istruzioni aritmetico-logiche di tipo R si distinguono tra loro in base al campo **funct.**



## 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>sub \$s1, \$s2, \$s3</b>	000000	10010	10011	10001	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, 7</b> <i>s1 = s2*2<sup>7</sup></i>	000000	X	10010	10001	00111 (7)	000000

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, 7</b> <i>s1 = s2*2<sup>-7</sup></i>	000000	X	10010	10001	00111 (7)	000010

A.A. 2004-2005

9/48

<http://homes.dsi.unimi.it/~borghese>



## Sommario



Il linguaggio macchina: le istruzioni di tipo R

Le istruzioni di tipo I

Le istruzioni di tipo J

Modalità di indirizzamento

Trattamento delle costanti

A.A. 2004-2005

10/48

<http://homes.dsi.unimi.it/~borghese>



## Formato istruzioni di tipo I

op	rs	rt	Indirizzo (offset, 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 spiazamento (offset).
- Con questo formato una istruzione **lw (sw)** può indirizzare byte nell'intervallo  $-2^{15} + 2^{15}-1$  rispetto all'indirizzo base:  $\text{indirizzo} = \text{indirizzo\_base} + \text{offset}$ .



## Istruzioni di tipo I: esempio

lw \$t0, 32(\$s3)



0x8E680020



## Istruzioni di tipo I: esempi



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

Nome campo	op	rs	rt	indirizzo			
Dimensione	6-bit	5-bit	5-bit	16-bit			
<code>addi \$t0, \$s3, 32</code>	001000	10011	01000	0000	0000	0010	0000



## Dal C al linguaggio macchina



`A[300] = h + A[300]`



```
lw $t0, 1200($t1)
add $t0, $s2, $t0
sw $t0, 1200($t1)
```

`$s2` → `h`

`$t1` → Indirizzo base di `A`

35	9	8	1200		
0	18	8	8	0	32
43	9	8	1200		



100011	01001	01000	0000010010110000			
000000	10010	01000	01000	00000	100000	
101011	01001	01000	0000010010110000			

```
0x8D2804B0
0x02484020
0xAD2804B0
```



## Versione I di istruzioni aritmetico-logiche



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

Nome campo	op	rs	rt	"indirizzo"
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



## Istruzioni di controllo di flusso (salto)



- Il PC viene incrementato di 4 (byte) durante l'esecuzione di un'istruzione.
- Salti condizionati relativi (beq, bne...) – **Formato I**:
  - Il flusso sequenziale di controllo cambia solo se la condizione è vera.
  - Il calcolo del valore dell'etichetta **L1** (indirizzo di destinazione del salto) è relativo al Program Counter (PC).
- Salti incondizionati assoluti (j, jal...) – **Formato J**:
  - Il salto viene sempre eseguito.
  - L'indirizzo di destinazione del salto è un indirizzo assoluto di memoria.





## 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.
  - Il calcolo del valore dell'etichetta **L1** (indirizzo di destinazione del salto) è relativo al Program Counter (PC).



## Esempio



### Assembly

```
Loop: add $t1, $s3, $s3
.....
      bne $t0, $s5, Exit
      add $s3, $s3, $s4
      beq $t0,$s5, Loop
Exit:
.....
```

### Assembly con le etichette risolte

```
80000: add $t1, $s3, $s3
.....
80016: bne $t0, $s5, 8
80020: add $s3, $s3, $s4
80024: beq $t0,$s5, -28
80028:
```

**Nota:** quando si esegue la **bne**, PC punta già all'istruzione successiva (e.g. **bne** -> PC = 80020)



## Analisi dell'offset



- Per il **principio di località** degli indirizzi di memoria è utile calcolare l'indirizzo di destinazione del salto come **offset** rispetto all'istruzione corrente.
- Nelle architetture MIPS tutte le istruzioni hanno 32 bit (RISC) e sono allineate al MSB. I due bit meno significativi dell'indirizzo delle istruzioni sono quindi sempre 00
- Per l'offset si hanno a disposizione solo 16-bit del campo **indirizzo**. Questo indirizzo è espresso relativamente al PC. (**PC-relative address**).
- Inoltre sfruttando al massimo i 16 bit  $\Rightarrow$  rappresentano un offset di **parola (PC-relative word address)**.
- Una istruzione di **salto** può indirizzare **parole** nell'intervallo  $-2^{15} + 2^{15}-1$  rispetto all'indirizzo base.
- L'indirizzamento relativo al Program Counter permette di fare dei salti condizionati ad aree di memoria il cui indirizzo non è esprimibile con 16-bit.



## Esempio



Esempio: `bne $s0, $s1, L1`

$$L1 = (\text{Ind\_salto} - \text{PC}) / 4$$

Offset  $\nearrow$

L'assemblatore sostituisce l'etichetta **L1** con l'indirizzo **di parola** relativo a PC: **(L1-PC)/4**

- Il PC contiene già l'indirizzo dell'istruzione successiva al salto
- La divisione per 4 serve per calcolare lo spaziamento in numero di parole.



## Esempio



Assembly	Assembly	Linguaggio Macchina
Loop: add \$t1, \$s3, \$s3	add \$t1, \$s3, \$s3	80000: 0 19 19 9 0 32
.....	.....	
bne \$t0, \$s5, Exit	bne \$t0, \$s5, 8	80016: 5 8 21 2
add \$s3, \$s3, \$s4	add \$s3, \$s3, \$s4	80020:
beq \$t0,\$s5, Loop	beq \$t0,\$s5, -28	80024: 4 8 21 -7
Exit:		80028: (Exit) ...
.....		

$$2 = (80028 - 80020) / 4$$

$$-7 = (80000 - 80028) / 4$$



Nota: quando si esegue la **bne**, PC punta già all'istruzione successiva (e.g. bne -> PC = 80020)



## Formato istruzioni di salto condizionato



op	rs	rt	Indirizzo
6 bit	5 bit	5 bit	16 bit

- Nel caso di salti condizionati, i campi hanno il seguente significato:
  - **op** identifica il tipo di istruzione;
  - **rs** indica il primo registro;
  - **rt** indica il secondo registro;
  - **indirizzo** riporta lo spiazzamento (offset). Il valore del campo indirizzo può essere negativo (salti all'indietro)



## Branch: esempi



Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
<code>beq \$s1, \$s2, 100</code>	000100	10001	10010	0000 0000 0001 1001

L1 = 100 in 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
<code>beq \$s1, \$s2, -100</code>	000100	10001	10010	1111 1111 1110 0111

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



## Sommario



Il linguaggio macchina: le istruzioni di tipo R

Le istruzioni di tipo I

**Le istruzioni di tipo J**

Modalità di indirizzamento

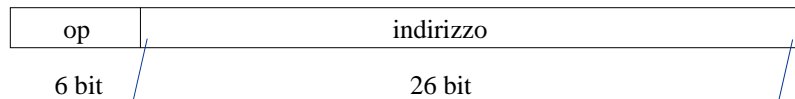
Trattamento delle costanti



## Formato istruzioni di tipo J

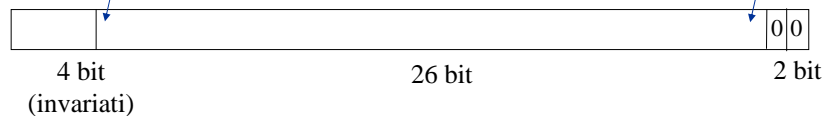


- E' il formato usato per le istruzioni di salto incondizionato (*jump*):



- 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**)

PC



A.A. 2004-2005

25/48

<http://homes.dsi.unimi.it/~borghese>



## Istruzioni di salto incondizionato



- L'assemblatore sostituisce l'etichetta **L1** con i 28 bit meno significativi traslati a destra di 2 (divisione per 4 per calcolare l'indirizzo di parola) per ottenere 26-bit
  - In pratica elimina i due 0 finali
  - Si amplia lo spazio di salto:  
si salta tra 0 e  $2^{28}$  Byte ( $2^{26}$  word) = 256 Mbyte.
- I 26-bit di indirizzo nelle jump rappresentano un indirizzo di parola (word address)  $\Rightarrow$  corrispondono ad un indirizzo di byte (byte address) composto da 28-bit.
- Poiché il registro PC è composto da 32-bit  $\Rightarrow$  l'istruzione jump rimpiazza solo i 28-bit meno significativi del PC, lasciando inalterati i rimanenti 4-bit più significativi.
- La memoria testo (S.O. + codice) è compresa tra 0 e 256Mbyte ( $2^{28}$ ), i rimanenti 4 bit saranno 0000.

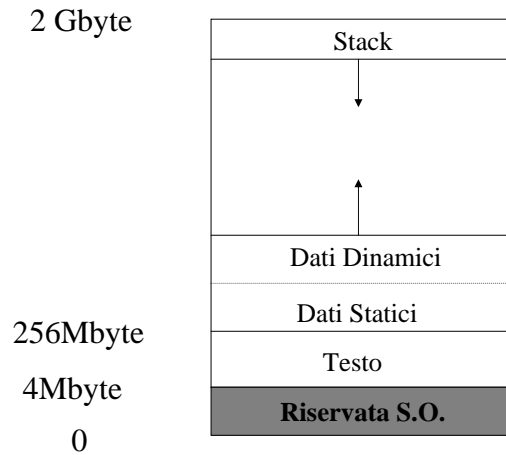
A.A. 2004-2005

26/48

<http://homes.dsi.unimi.it/~borghese>



## Organizzazione logica della memoria



A.A. 2004-2005

27/48

<http://homes.dsi.unimi.it/~borghese>



## Esempio



<b>Loop:</b>	<code>add \$t1, \$s3, \$s3</code>	<code>80000: 0 19 19 9 0 32</code>	<code>do { t1 = s3*2;</code>
	.....	.....	
	<code>bne \$t0, \$s5, Exit</code>	<code>80016: 5 8 21 2</code>	<code>if (t1 != s5) break;</code>
	<code>add \$s3, \$s3, \$s4</code>	<code>80020: 0 19 20 19 0 32</code>	<code>s3 +=s4;</code>
	<code>beq \$t0, \$s5, Loop</code>	<code>80024: 4 8 21 -7</code>	<code>} while (t0 == s5);</code>
<b>Exit:</b>		<code>80028: Exit ...</code>	



<b>Loop:</b>	<code>add \$t1, \$s3, \$s3</code>	<code>80000: 0 19 19 9 0 32</code>	<b>Loop:</b> <code>t1 = s3*2</code>
	.....	.....	
	<code>bne \$t0, \$s5, Exit</code>	<code>80016: 5 8 21 2</code>	<code>if (t1 != s5) break;</code>
	<code>add \$s3, \$s3, \$s4</code>	<code>80020: 0 19 20 19 0 32</code>	<code>s3 +=s4;</code>
	<b><code>j Loop (j 80000)</code></b>	<b><code>80024: 2 20000</code></b>	<b><code>goto Loop;</code></b>
<b>Exit:</b>		<code>80028: Exit</code>	

A

se



## Istruzioni di tipo J: esempio

Nome campo	op	indirizzo					
Dimensione	6-bit	26-bit					
j 32	000010	00	0000	0000	0000	0000	1000

32 = 100(00)

Nome campo	op	indirizzo						
Dimensione	6-bit	26-bit						
j 80000	000010	00	00000	0001	0100	1110	0010	0000

80000 = 000 0000 0001 0011 1000 1000 00(00)



## Salti incondizionati indiretti

**j<sub>r</sub> rs** (jump register con **formato R**)

- Salta all'indirizzo di memoria **assoluto** contenuto nel registro **rs** (spazio di  $2^{32}$  Word cioè  $2^{34}$  byte = 8 Gbyte > intero spazio di memoria)

0	rs	0	0	0	8
---	----	---	---	---	---



## 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. 2004-2005

31/40

<http://homes.dsi.unimi.it/~borghese>



## 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)

A.A. 2004-2005

32/48

<http://homes.dsi.unimi.it/~borghese>







## Modalità di indirizzamento



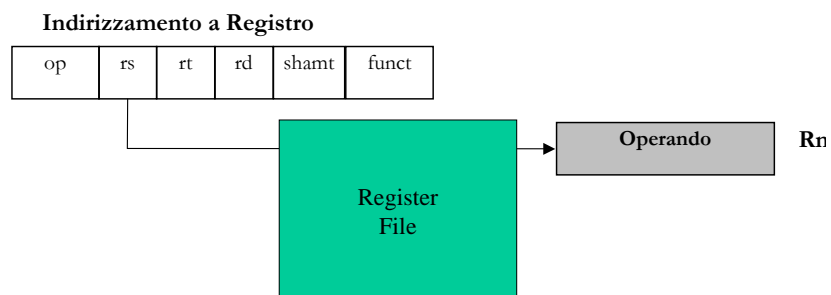
- Le modalità di indirizzamento indicano le diverse modalità attraverso le quali far riferimento ai dati ed alle istruzioni in memoria e nel register file.
- L'esempio più comune di modalità di indirizzamento è l'indirizzamento **a registro** nel quale gli operandi dell'istruzione sono contenuti nei registri:  
ad esempio `add $s0, $s1, $s2`.
- MIPS ha solo 5 modalità di indirizzamento:
  - A registro
  - Immediato
  - Con base o spiazzamento
  - Relativo al Program Counter
  - Pseudo-diretto
- Una singola istruzione può usare più di una modalità di indirizzamento.



## Indirizzamento a registro



- L'operando (l'indirizzo) è il contenuto di un registro della CPU: il nome (numero = indirizzo) del registro è specificato nell'istruzione.





## Esempio di indirizzamento a registro



- Le istruzioni che usano **solamente** questo tipo di indirizzamento hanno formato di tipo R.
- Esempio: istruzione aritmetico-logica:

Nome campo	op	rs	rt	rd	shamt	funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
add \$s1, \$s2, \$s3	000000	10010	10011	10001	00000	100000



## Indirizzamento immediato



- L'operando è una costante il cui valore è contenuto nell'istruzione.
- L'indirizzamento immediato si usa per specificare il valore di un operando sorgente, non ha senso usarlo come destinazione.



- Le istruzioni che usano questo tipo di indirizzamento hanno formato I
  - La costante è memorizzata nel campo a 16-bit



## Indirizzamento immediato



- Esempio: operazione aritmetico-logica con operando immediato (formato tipo I):

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

- Esempio: operazione di confronto con operando immediato (formato tipo I):

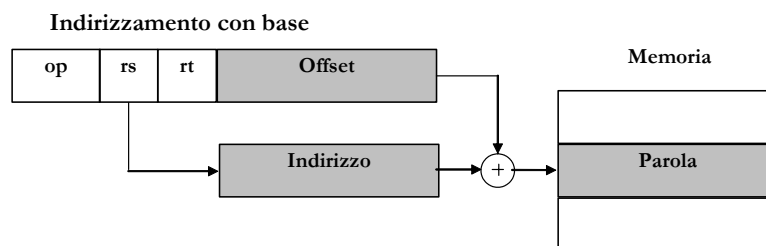
Nome campo	op	rs	rt	indirizzo			
Dimensione	6-bit	5-bit	5-bit	16-bit			
<code>slti \$t0, \$s2, 8</code>	001010	10010	01000	0000	0000	0000	1000



## Indirizzamento con base



- L'operando è in una locazione di memoria il cui indirizzo si ottiene sommando il contenuto di un registro base ad un valore costante (*offset o spiazzamento*) contenuto nell'istruzione.



- Le istruzioni che usano questo tipo di indirizzamento hanno formato di tipo I.

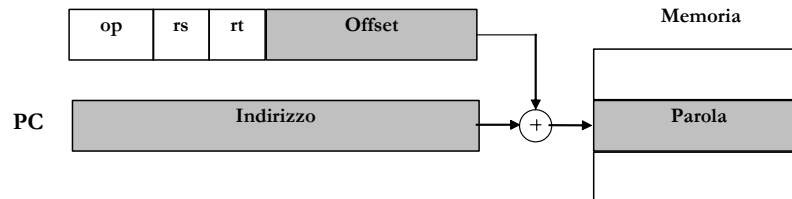




## Indirizzamento relativo al PC



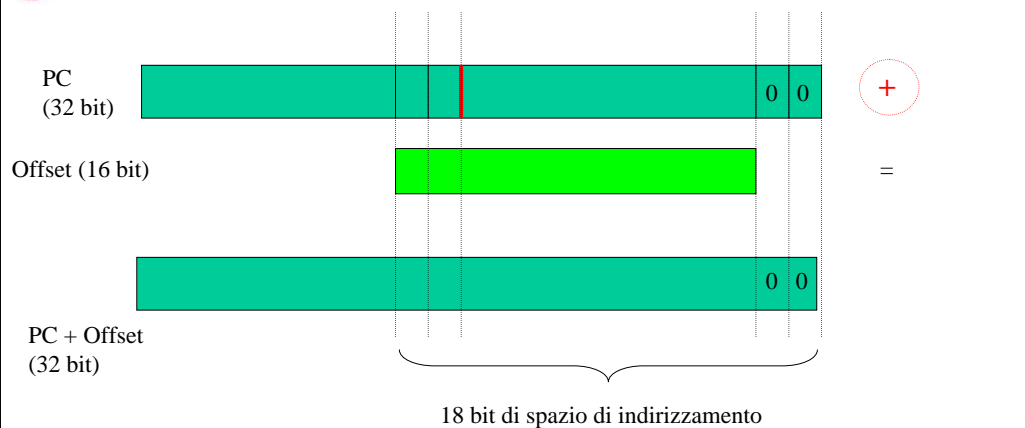
- L'**istruzione** è in una locazione di memoria il cui indirizzo si ottiene sommando il contenuto del *Program Counter* ad un valore costante (*offset o spiazzamento*) contenuto nell'istruzione:



- Le istruzioni che usano questo tipo di indirizzamento hanno formato di tipo I.



## Utilizzo dell'offset





## Esempio di indirizzamento relativo al PC



- Esempio: Operazione di salto condizionato (formato tipo I):
- Si usa l'indirizzamento relativo al PC nei salti condizionati in quanto la destinazione del salto in tali istruzioni è in genere prossima al punto di salto.
- Avendo a disposizione 16 bit di *Offset*  $\Rightarrow$  è possibile saltare in un'area tra  $-2^{15}$  e  $+2^{15}-1$  parole rispetto all'istruzione corrente.

Nome campo	op	rs	rt	indirizzo			
Dimensione	6-bit	5-bit	5-bit	16-bit			
<code>beq \$s1, \$s2, 100</code>	000100	10001	10010	0000	0000	0001	1001

Nome campo	op	rs	rt	indirizzo			
Dimensione	6-bit	5-bit	5-bit	16-bit			
<code>bne \$s1, \$s2, 100</code>	000101	10001	10010	0000	0000	0001	1001

A.A. 2004-2005

45/48

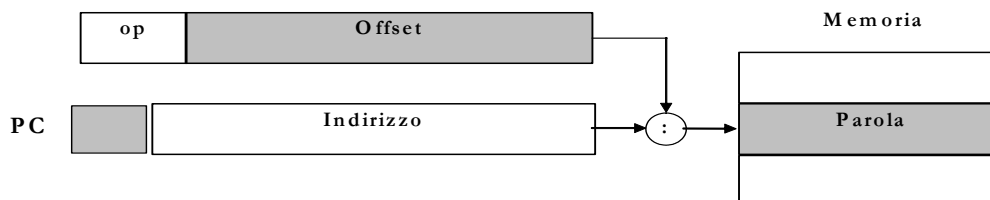
<http://homes.dsi.unimi.it/~borghese>



## Indirizzamento pseudo-diretto



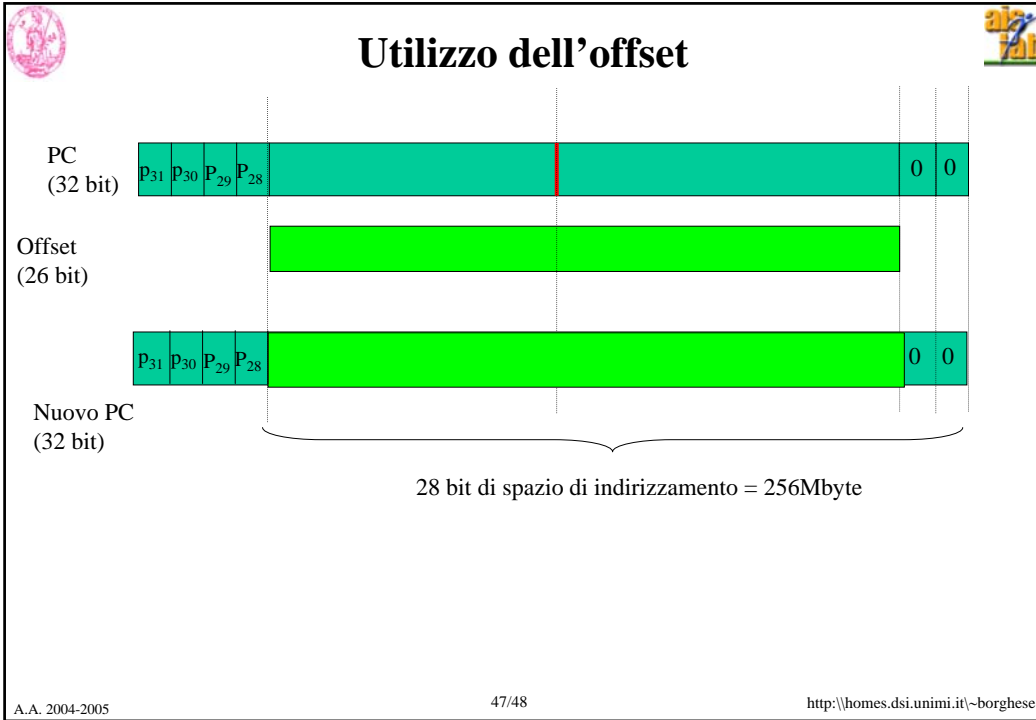
- Una parte dell'indirizzo è presente come valore costante (offset) nell'istruzione ma deve essere completato.
- Le istruzioni che usano questo tipo di indirizzamento hanno formato di tipo J.
- L'indirizzo di salto si calcola facendo uno shift a sinistra di 2 bit dei 26-bit di offset contenuti nell'istruzione (aggiungendo 00 nei bit meno significativi per passare da 26 a 28-bit) e concatenando i 28-bit con i 4-bit più significativi del Program Counter.



A.A. 2004-2005

46/48

<http://homes.dsi.unimi.it/~borghese>



## Esempio di indirizzamento pseudo-diretto

- Esempio: operazione di salto incondizionato (formato J)

Nome campo	op	indirizzo				
Dimensione	6-bit	26-bit				
j 32	000010	00	0000	0000	0000	0000 1000

A.A. 2004-2005 48/48 <http://homes.dsi.unimi.it/~borghese>





## Sommario



Il linguaggio macchina: le istruzioni di tipo R

Le istruzioni di tipo I

Le istruzioni di tipo J

Modalità di indirizzamento

Trattamento delle costanti