Il set di istruzioni di MIPS32
Modalità di indirizzamento

Proff. A. Borghese, F. Pedersini

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

Instruction Set Architecture (ISA) di MIPS

Formato delle istruzioni macchina:

FORMATO: definizione di come un'istruzione macchina viene codificata e organizzata nella parola binaria che la rappresenta

Formato delle istruzioni MIPS32

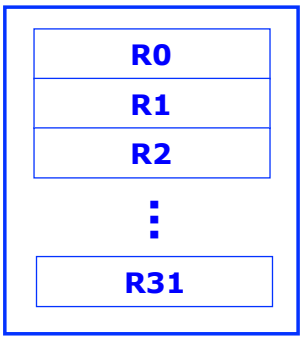
- Tutte le istruzioni MIPS hanno la stessa dimensione: **32 bit**
- La categoria di istruzione è riconosciuto in base al valore dei **6 bit** più significativi: (codice operativo - "**OPCODE**")
- Le istruzioni MIPS sono di **3 tipi** (formati)
  - **Tipo R (register)** Istruzioni che operano su registri
  - **Tipo I (immediate)** Istruzioni in cui un operando è "immediato" cioè è contenuto nell'istruzione stessa
  - **Tipo J (jump)** Istruzioni di salto

Uso dei registri: convenzioni

Il **Register File** di **MIPS32**

Nome *e* ***utilizzo convenzionale*** *dei registri*
(general purpose registers) del Register File di MIPS32

Register File



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

Istruzioni di tipo R

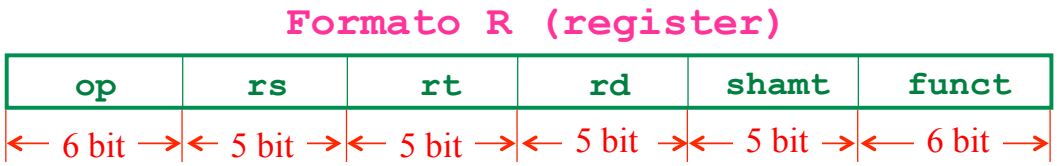
Istruzioni MIPS "Tipo R":
istruzioni che operano **su registri**

- Categorie:
  - Generalmente aritmetico-logiche, ma non solo.

    `add, sub, mult, div, and, or, ... , slt`
  - Gestione registri speciali: `mflo, mfhi`
  - Salto: jump register: `jr`

- **Struttura comune** a tutte le tipo R,
  si differenziano in base al campo `funct (6 bit)`

Formato istruzioni: **formato/tipo R**



op: (**opcode**) identifica il tipo di istruzione

rs: registro contenente il primo operando sorgente

rt: registro contenente il secondo operando sorgente

rd: registro destinazione contenente il risultato

shamt: shift amount (scorrimento)

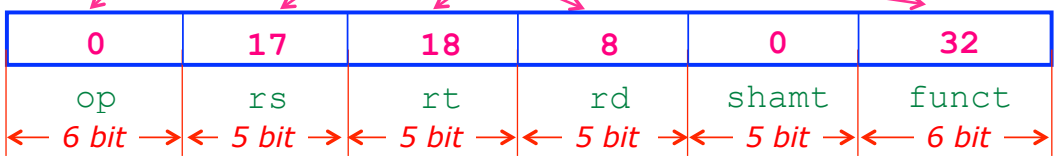
funct: indica la variante specifica dell'operazione

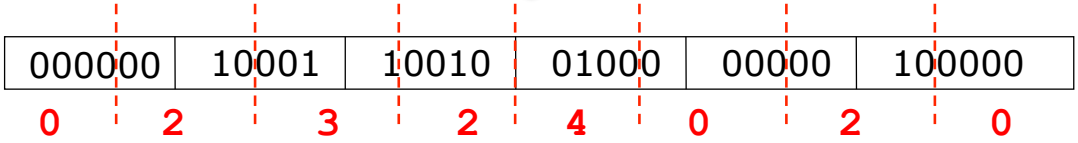
Istruzioni di tipo R: esempio

Esempio: `add $t0, $s1, $s2`

```
add: 0 (funct=32)
$t0 → $8
$s1 → $17
$s2 → $18
```

`add $8, $17, $18`





0x02324020

Istruzioni di tipo R: esempi

- Le istruzioni di **tipo R** hanno **opcode=0**
  - Non tutte le operazioni logico-aritmetiche sono di tipo R
  - Non tutte le operazioni con codice operativo 0 sono logico-aritmetiche

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

Nome campo	op	rs	rt	rd	shamt	funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
`sub $s1, $s2, $s3`	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
`and $s1, $s2, $s3`	000000	10010	10011	10001	00000	100100

`0x02538824` ←

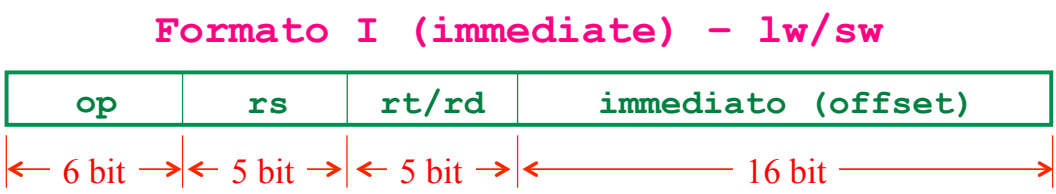
Istruzioni di tipo I

Formato I

Type I: istruzioni che contengono **un immediato**

- Categorie:
  - Trasferimento CPU / memoria: `lw, sw`
  - Aritmetico/logiche con operando immediato: `addi, subi, slti`
  - Salto condizionato: `beq, bne`
- A seconda della categoria, il campo **immediato** (di 16 bit) contiene informazioni di natura diversa.

Formato istruzioni di **tipo I**



Categoria: load/store

- `op` identifica il tipo di istruzione;
- `rs` indica il primo registro (sorgente – **registro BASE**);
- `rt/rd` indica il secondo registro, sorgente (`sw`) o destinazione (`lw`);
- `Indirizzo` riporta lo spiazzamento (offset)

- **16 bit** di spiazzamento
  - Intervallo: $-2^{15} \div +2^{15}-1$ (±32 kB) rispetto all'indirizzo base.

Istruzioni di tipo I: esempio

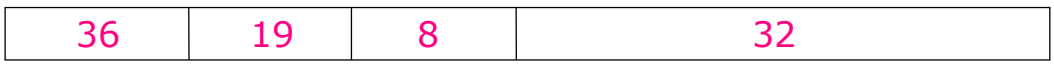
Istruzione **`lw`**: linguaggio macchina

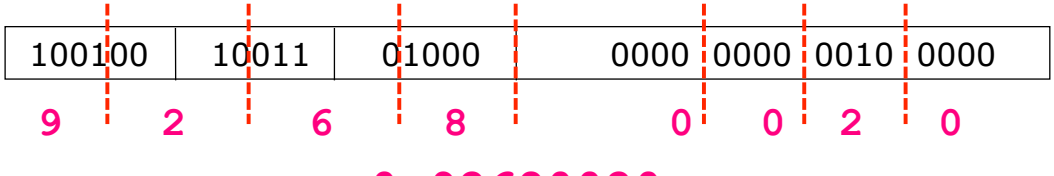
Esempio:

```
lw $t0, 32($s3)
```

```
lw $8, 32($19)
```

```
lw: 36
$s3 → $19
$t0 → $8
```





0x92680020

Istruzioni load/store di tipo I

- *Istruzioni:*
  - *Categoria:* ***load/store***
  - *Formato:* ***tipo I:***

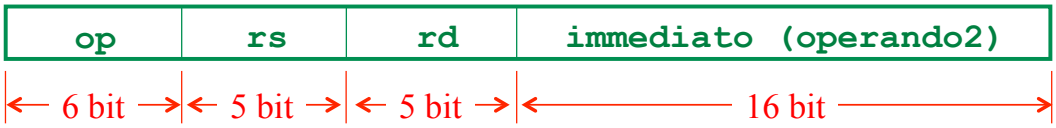
Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`lw $t0, 32 ($s3)`	100100	10011	01000	0000 0000 0010 0000

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`sw $t0, 32 ($s3)`	101100	10011	01000	0000 0000 0010 0000

Formato I – istr. aritmetico-logiche

Categoria: Istruzione aritmetico-logica con operando immediato
Formato: `tipo I`

Formato I (immediate) – aritm.logiche



- `op` identifica il tipo di istruzione;
- `rs` indica il registro sorgente (primo operando);
- `rd` indica il registro destinazione;
- `Valore` riporta il secondo operando (costante)

Operando immediato: 16 bit

- **con segno**: intervallo $-2^{15} \div +2^{15}-1$ ($\pm$ 32 k)
- **senza segno**: intervallo: $0 \div +2^{16}-1$ ($0 \div 64$ k)

Istruzioni aritmetico-logiche di **tipo I**

Esempio: **`addi` (add immediate)** in linguaggio macchina:

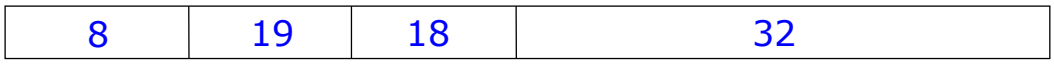
```
addi $s2, $s3, 32
```

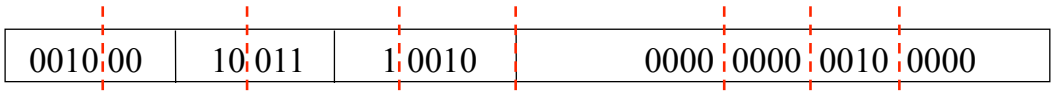
↓

```
addi $18, $19, 32
```

addi: 8

$s2 → $18
$s3 → $19





2 2 7 2 0 0 2 0

0x22720020

Istruzioni logico-aritmetiche di tipo I

Esempi di istruzioni aritmetico/logiche con operando immediato (istruzioni: tipo I):

- **`addi`** (*add immediate*)
- **`slti`** (*set less than immediate*)

Nome campo	op	rs	rt	"Indirizzo"
Dimensione	6-bit	5-bit	5-bit	16-bit
`addi $s1, $s1, 4`	001000	10001	10001	0000 0000 0000 0100

Nome campo	op	rs	rt	"indirizzo"
Dimensione	6-bit	5-bit	5-bit	16-bit
`slti $t0, $s2, 8`	001010	10010	01000	0000 0000 0000 1000

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

Istruzioni di salto

La categoria delle istruzioni di salto comprende:

- Salto condizionato: `beq, bne` (branch on equal / on not equal)
- Salto incondizionato: `j,jr` (jump)
- Salto a procedura: `jal` (jump and link)

SALTO: aggiornamento del Program Counter

Nel Program Counter viene inserito l'indirizzo dell'istruzione a cui saltare.

Attenzione!

*il **Program Counter** viene **incrementato di 4** (4 byte = 1 word)*
***subito dopo il fetch** di un'istruzione (prima del termine)*

➔ *il **PC punta già all'istruzione successiva** a quella in esecuzione*

Salti condizionati (branch) in MIPS:

indicazione di salto: **– relativa al PC,**
– in n. di istruzioni (word)

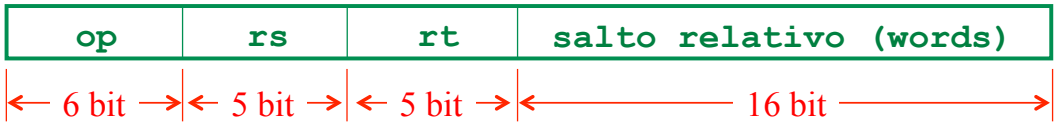
```
beq r1, r2, L1      (branch on equal)
bne r1, r2, L1      (branch on not equal)
```

- **condizionati:**
  - Il flusso sequenziale di controllo cambia solo se la condizione è vera.
- **salto relativo al PC:**
  - Il calcolo dell'indirizzo di destinazione del salto è basato sul valore attuale del Program Counter, ed è espresso, in linguaggio macchina, in **numero di words (istruzioni) di cui spostarsi**

Formato I – Salto condizionato (branch)

Categoria: salto condizionato (**branch**) – relativo al PC
Formato: **tipo I**

Formato I (immediate) – branch



- **op** identifica il tipo di istruzione (branch);
- **rs** indica il primo registro da confrontare;
- **rt** indica il secondo registro da confrontare;
- **Salto_relativo** riporta **l'ampiezza di salto:**
  – a partire dal contenuto attuale del Program Counter (PC + 4)
  – espressa **in parole (words)**

16 bit del campo: **Salto relativo**

Mi sposto di **Salto_relativo** *parole rispetto alla posizione attuale indicata dal PC*

Gittata di salto: $-2^{15} \div +2^{15}-1$ **parole** → $-2^{17} \div +2^{17}-1$ **Bytes (±128 kB)**

Istruzioni di salto condizionato (tipo I)

Offset nel salto condizionato MIPS32

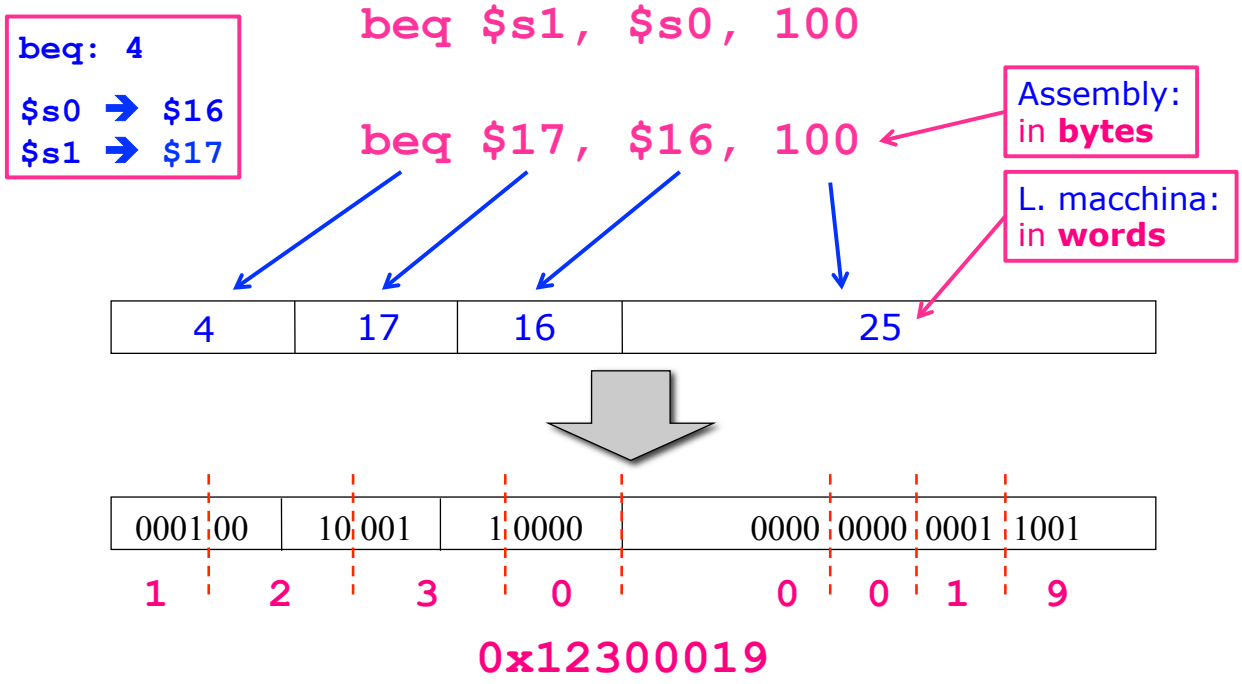
- **Offset** relativo al PC rappresentato in complemento a due per permettere salti in avanti e all'indietro.
  L'offset varia tra: -2^{15} e $+2^{15}-1$ **parole**
  Campo indirizzo negativo → salti all'indietro

- **Indirizzo di byte**:
  corrisponde all' **indirizzo di parola moltiplicato per 4**
  I due bit meno significativi sono sempre: '00'
  Ha senso per la **dimensione fissa (32 bit) delle istruzioni macchina**.

- Si può saltare ad una ***parola*** nell'intervallo:
  $-2^{17} \div +2^{17}-1$ bytes
  Range: 2^{18} **= 256 K (± 128 K) bytes**

Istruzioni di salto condizionato (branch) – **tipo I**

Esempio: istruzione `beq` (branch on equal)



Istruzioni di branch: esempio

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

L1: +25 word = +100 byte → Codifica: 0000 0000 0001 1001 (00)

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`beq $s1, $s2, -100`	000100	10001	10010	1111 1111 1110 0111

L1: −25 word = −100 byte → Codifica: 1111 1111 1110 0111 (00)

Esempio

ASSEMBLY

```
Loop:  add $t1, $s3, $s3
        .....
       bne $t0, $s5, Exit
      (bne $t0, $s5, 8)
       add $s3, $s3, $s4
       beq $t0,$s5, Loop
Exit:
        ............
```

LINGUAGGIO MACCHINA

```
80000: 0 19 19 9 0 32
.....
80016: 5  8 21  2
80028: (Exit) ...
```

n. di **byte** di cui mi sposto

n. di **parole** di cui mi sposto

2 = (80028 – 80020) / 4

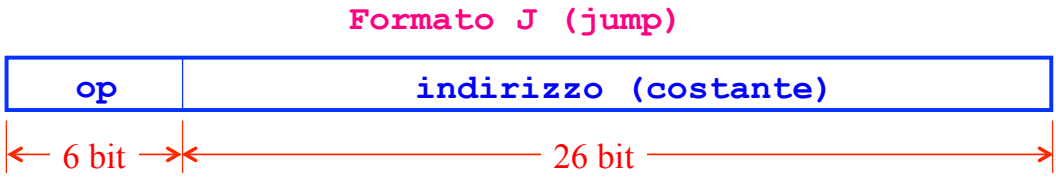
Quando si esegue la **bne**,
PC punta già all'istruzione successiva (80020)

Esempio

```
Loop: add $t1, $s3, $s3        80000: 0 19 19 9 0 32
        ............           .........
      bne $t0, $s5, Exit       80016: 5 8 21  2
      add $s3, $s3, $s4        80020: 0 19 20 19 0 32
      beq $t0,$s5, Loop        80024: 4 8 21 -7
Exit:                          80028: (Exit) ...
        ............
```

-7 = (80000 – 80028)/4

Istruzioni di tipo J



- **Tipo J**: usato per le istruzioni di **salto incondizionato** (`jump`)
  - **op (6 bit)**: indica il tipo di operazione
  - **indirizzo (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 assoluto, di parola** (word address)
  - Poiché si punta sempre e comunque ad un'istruzione di programma

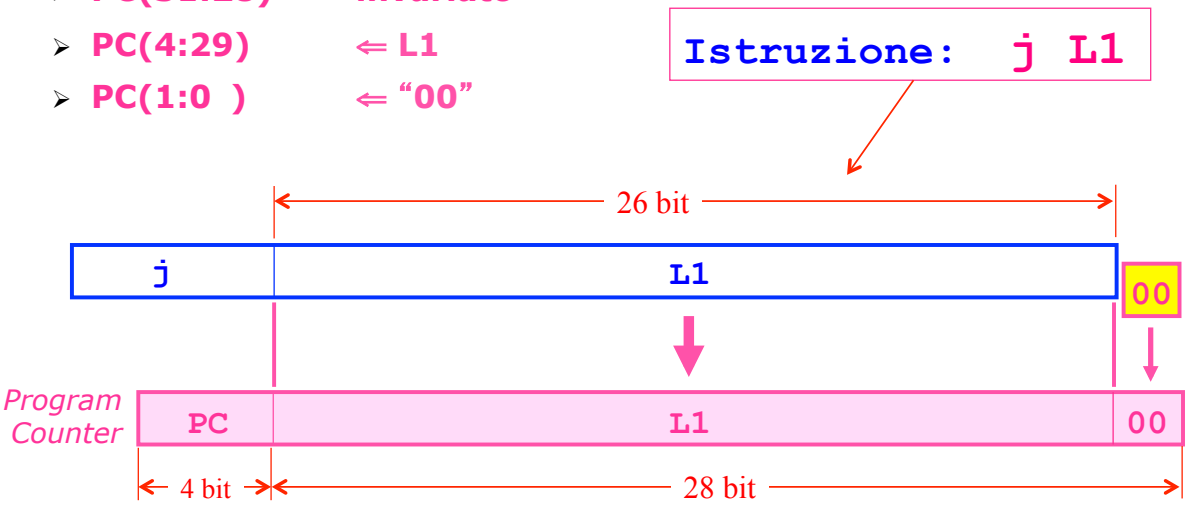
Jump: modifica del *Program Counter*

- Il campo **L1** di 26 bit rappresenta un **indirizzo di parola**

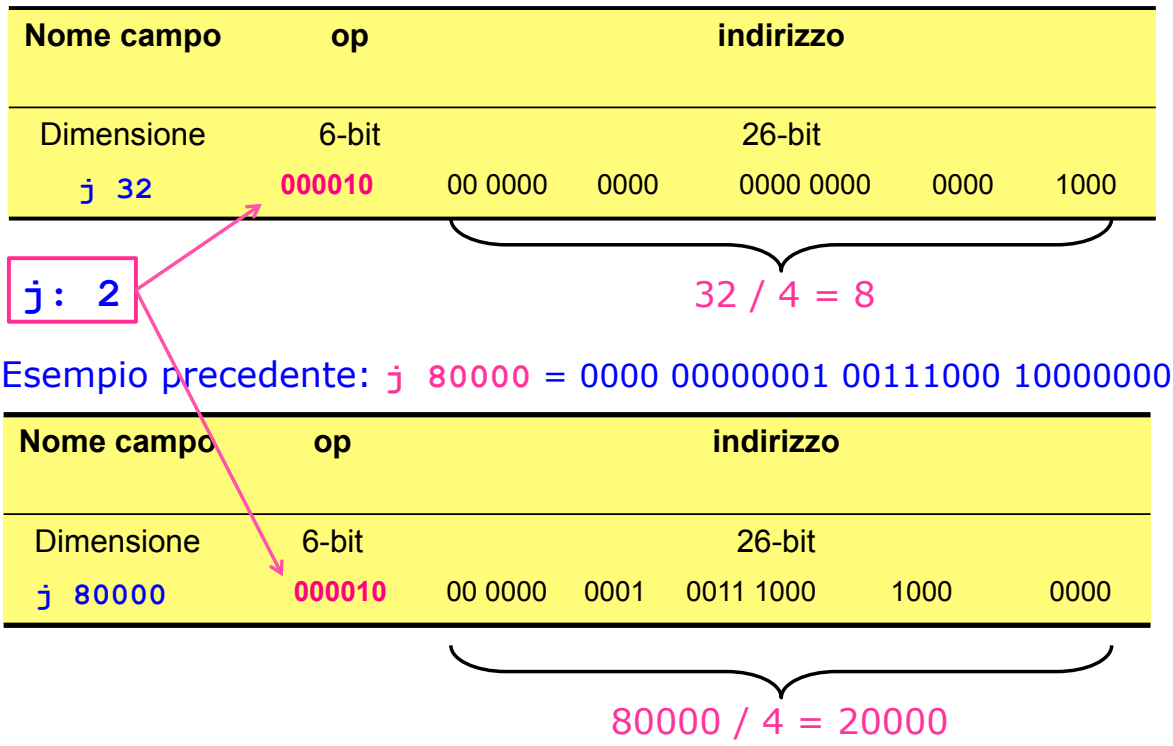
  [indirizzo di byte] = [indirizzo di parola] x 4

  In base 2: *[indirizzo di byte] = [indirizzo di parola | **00**]*

  - **PC(31:28)** **invariato**
  - **PC(4:29)** ⇐ **L1**
  - **PC(1:0)** ⇐ **"00"**



Istruzioni di tipo J: esempi



Istruzioni di salto incondizionato (tipo J)

- L' Assembler sostituisce l'etichetta `L1` con i **28 bit** meno significativi dell'indirizzo di memoria (byte) traslati a destra di 2 posizioni, ottenendo **26 bit**
  - I **26 bit** di indirizzo rappresentano un indirizzo di parola (word address)
  - indirizzo di byte (byte address) composto da **28 bit** (word address | 00)
  - Si amplia lo spazio di salto: tra 0 e 2^{28} Byte (2^{26} word) = 256 Mbyte
- Registro PC è composto da **32 bit** (spazio: 4 GB) ⇒
  - `jump` rimpiazza solo i 28 bit meno significativi del PC
  - lascia inalterati i 4 bit più significativi
- Per saltare su 32 bit devo utilizzare: `jr $reg`

Esempio : jump (**formato J**)

Esempio:
sostituiamo j a beq nel codice dell'esempio precedente

```
beq $t0,$s5, Loop   ➔   j Loop
```

Assembly

```
Loop:  add $t1, $s3, $s3
        ..........
       bne $t0, $s5, Exit
       add $s3, $s3, $s4
       j Loop (j 80000)
Exit:
```

indirizzo al byte

Linguaggio macchina

```
0x0080000: 0 19 19 9 0 32
.........
0x0080016: 5 8 21  2
0x0080020: 0 19 20 19 0 32
0x0080024: 2 20000
0x0080028: Exit
```

indirizzo in word

Salti incondizionati: istruzione `jump register`

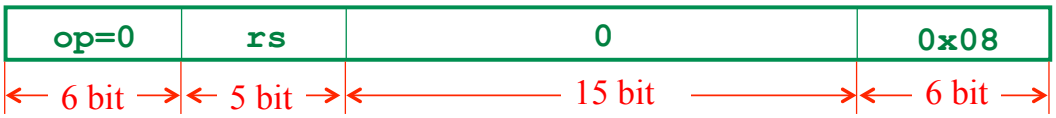
Per saltare ad indirizzi più lontati di 2^{28} **celle** si usa l'istruzione:

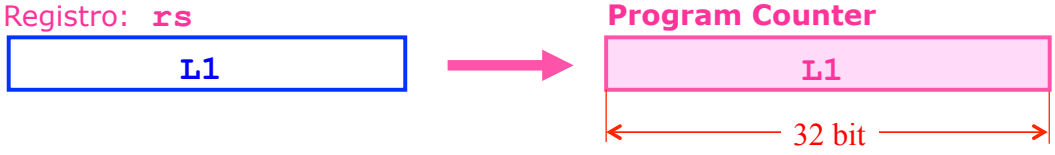
```
jr rs        (jump register)
```

Salta all'indirizzo di memoria **assoluto** contenuto nel registro `rs`

- In questo caso `L1` (contenuto in `rs`) è un BYTE Address.

Istruzione: `jr $ra` (formato R)





Istruzioni di salto: sintesi

- Salti condizionati, relativi: Formato **I**
  - Si salta solo se la condizione è vera.
  - L'indirizzo di destinazione del salto è relativo al Program Counter (PC).
  - `beq r1, r2, L1` (*branch on equal*)
  - `bne r1, r2, L1` (*branch on not equal*)

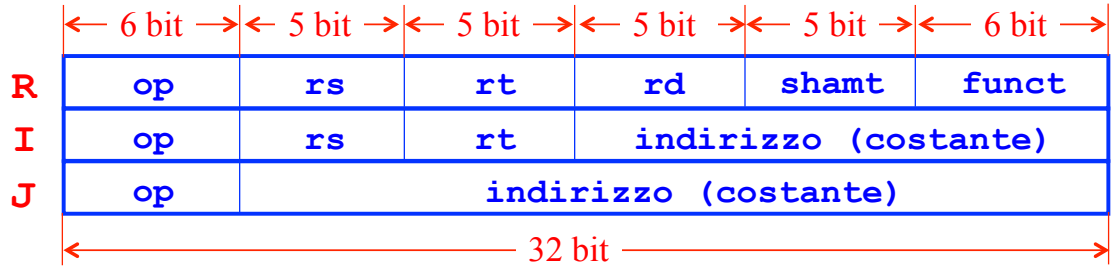
- Salti incondizionati, assoluti: Formato **J**
  - Il salto viene sempre eseguito
  - L'indirizzo di destinazione del salto è un indirizzo assoluto di memoria
  - `j L1` *(jump)*
  - `jal L1` *(jump and link)*

- Salto incondizionato a registro: formato **R**
  - L'indirizzo è contenuto in un registro
  - `jr rs` *(jump register)*

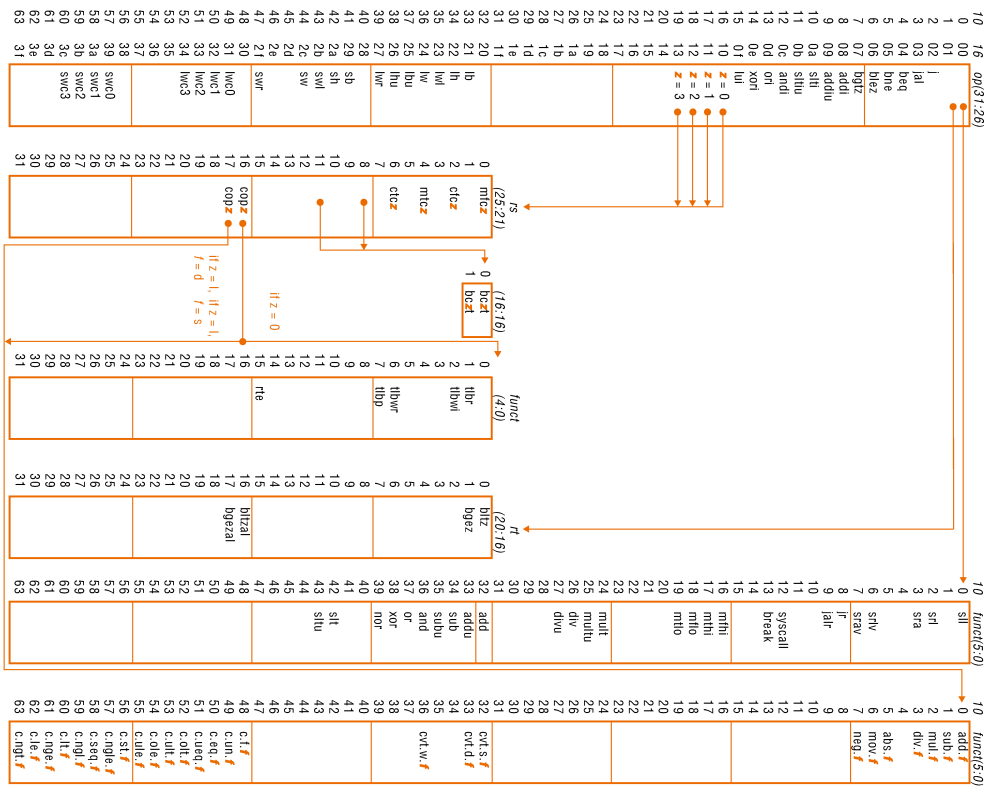
Istruzione	Tipo
`beq`	I
`bne`	I
`j`	J
`jal`	J
`jr`	R

Riepilogo: Formato istruzioni

- **3 diversi formati: R, I, J**
- Formato riconosciuto dalla CPU tramite il valore del primo campo:
  ***codice operativo* (*opcode*) – (6 bit più significativi)**
  - che indica alla macchina come trattare i rimanenti bit dell'istruzione.



Riepilogo: **MIPS Instruction Set**



Modalità di indirizzamento

Modalità di indirizzamento

Nel set istruzioni di un processore, per modalità di indirizzamento si indicano le diverse modalità attraverso le quali, a partire ***dalle indicazioni nell'istruzione, si recuperano gli operandi***

Esempi:

- indirizzamento **a registro**
  - gli operandi dell'istruzione sono contenuti nei registri:

    `add $s0, $s1, $s2`

- indirizzamento **immediato**
  - un operando è costante e contenuto nell'istruzione stessa

    `addi $s0, $s1, 1000`

MIPS ha **5 modalità** di indirizzamento:

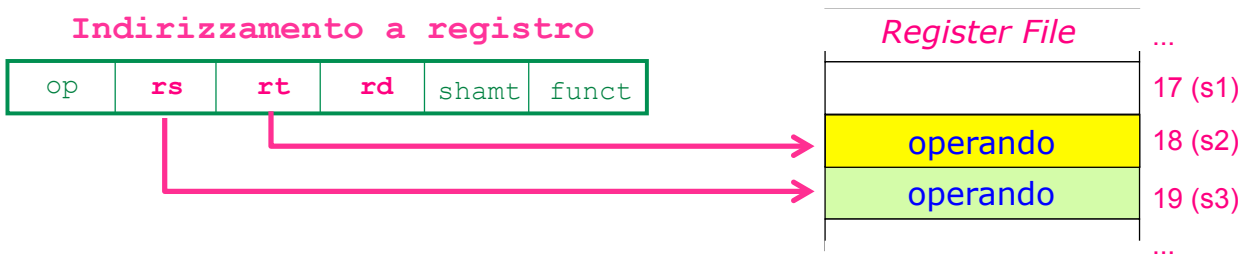
1. **a registro**
2. **immediato**
3. **con base e spiazzamento**
4. **relativo al Program Counter**
5. **pseudo-diretto**

- 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, di cui l'istruzione contiene l'identificativo.
  - il numero identificativo (0÷31) del registro è specificato nell'istruzione

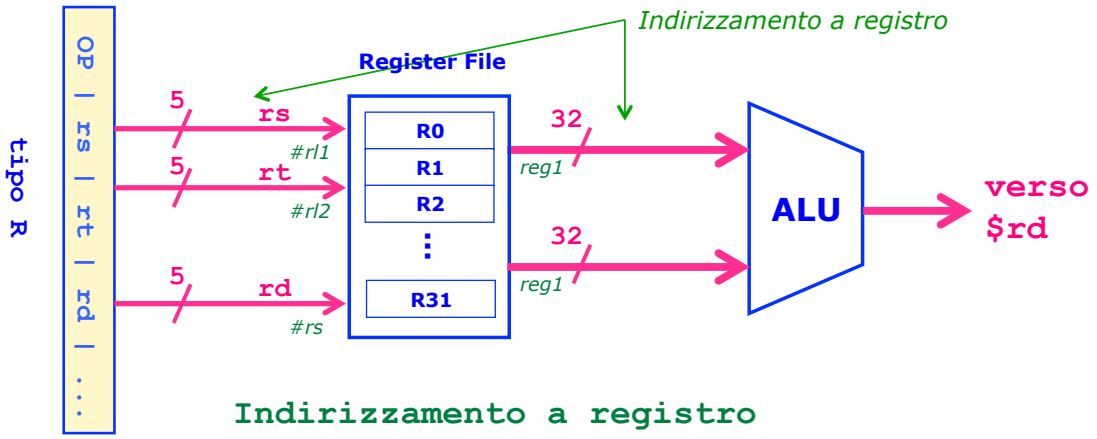
Nome campo	op	rs	rt	rd	shamt	funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
`add $s1,$s2,$s3`	000000	**10010**	**10011**	**10001**	00000	100000



Indirizzamento a registro

Esempio: operazione aritmetico-logica tra registri (tipo **R**)

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 a registro

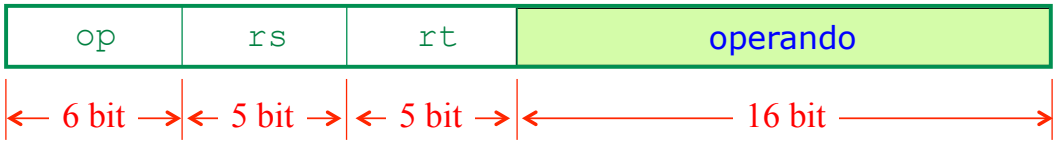
Indirizzamento immediato

Indirizzamento IMMEDIATO:

- L'operando è un valore immediato contenuto **nell'istruzione**.
- Le istruzioni che usano indirizzamento immediato sono di **tipo I**
  - La costante è memorizzata nel campo operando (16-bit)

Nome campo	op	rs	rt	indirizzo
Dimensione	6 bit	5 bit	5 bit	16 bit
`addi $s2, $s1, 4`	001000	10001	10001	0000 0000 0000 0100

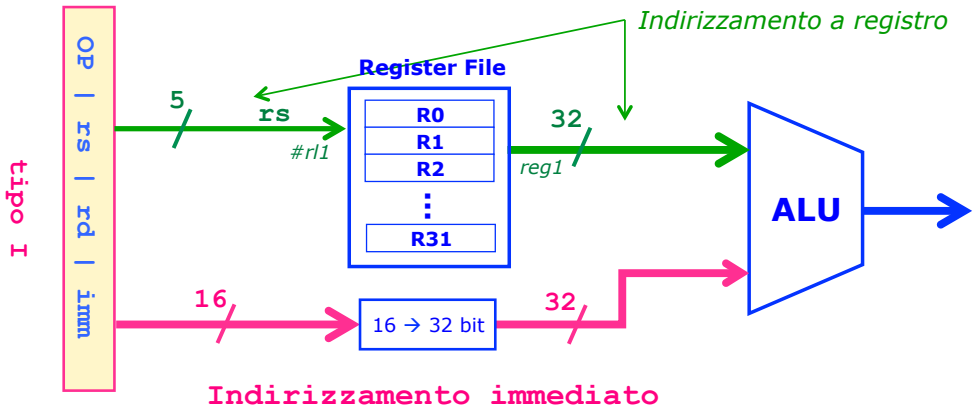
Indirizzamento immediato



Indirizzamento immediato

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

Nome campo	op	rs	rt	indirizzo
Dimensione	6 bit	5 bit	5 bit	16 bit
`addi $s2, $s1, 4`	001000	10001	10001	0000 0000 0000 0100

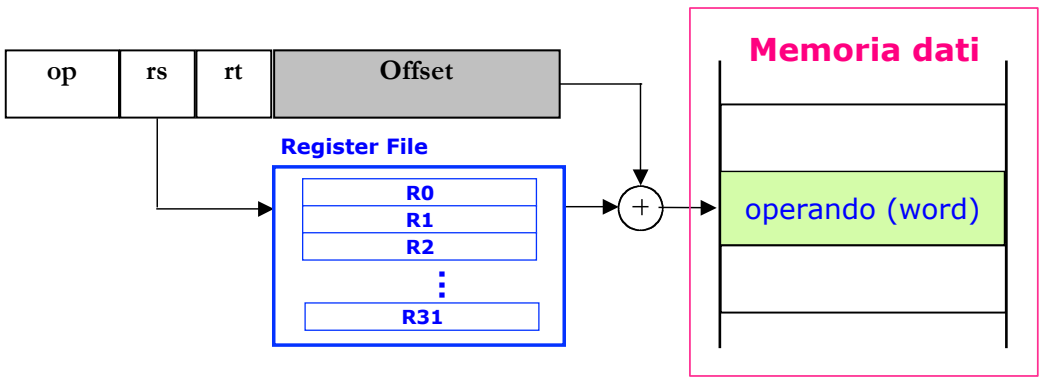


Indirizzamento con base e spiazzamento

Indirizzamento a BASE e SPIAZZAMENTO:

- L'operando è in una locazione di memoria il cui indirizzo si ottiene sommando il contenuto di un registro base ad un valore immediato (offset o spiazzamento) contenuto nell'istruzione
  - Le istruzioni con questo tipo di indirizzamento hanno **formato I**

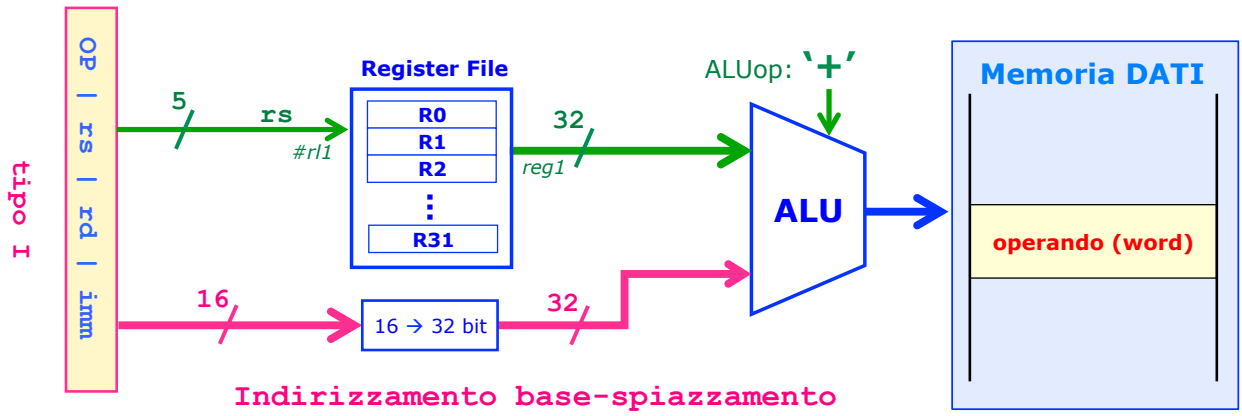
Indirizz. base + spiazzamento



Indirizzamento base-spiazzamento

Esempio: operazioni **`lw/sw:`**

Nome campo	op	rs	rt	indirizzo
Dimensione	6 bit	5 bit	5 bit	16 bit
`lw $5, 4($10)`	100100	01010	00101	0000 0000 0000 0100

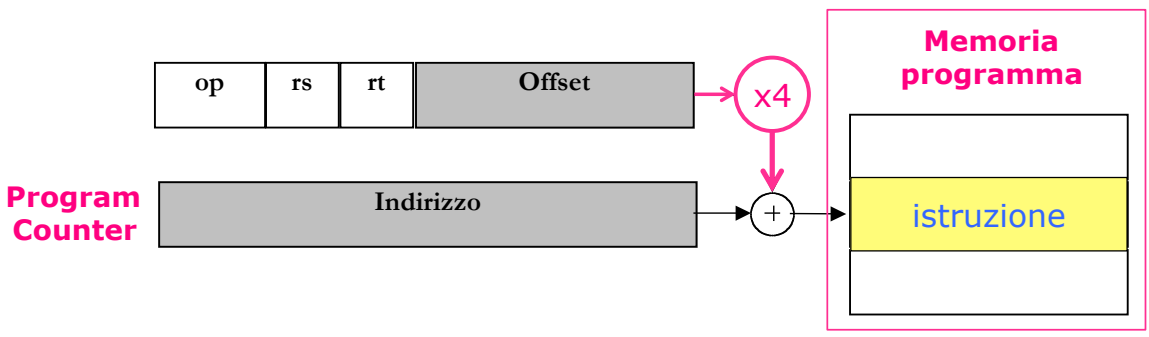


Indirizzamento relativo al PC

Indirizzamento RELATIVO AL PROGRAM COUNTER:

- l'operando è un indirizzo che si ottiene sommando il contenuto del Program Counter ad un valore immediato contenuto nell'istruzione, moltiplicato per 4
  - Avendo a disposizione 16 bit di immediato, è possibile saltare in un range: $-2^{15} \div 2^{15}-1$ **words** (→ $-2^{17} \div 2^{17}-4$ bytes) rispetto all'indirizzo corrente presente nel PC.

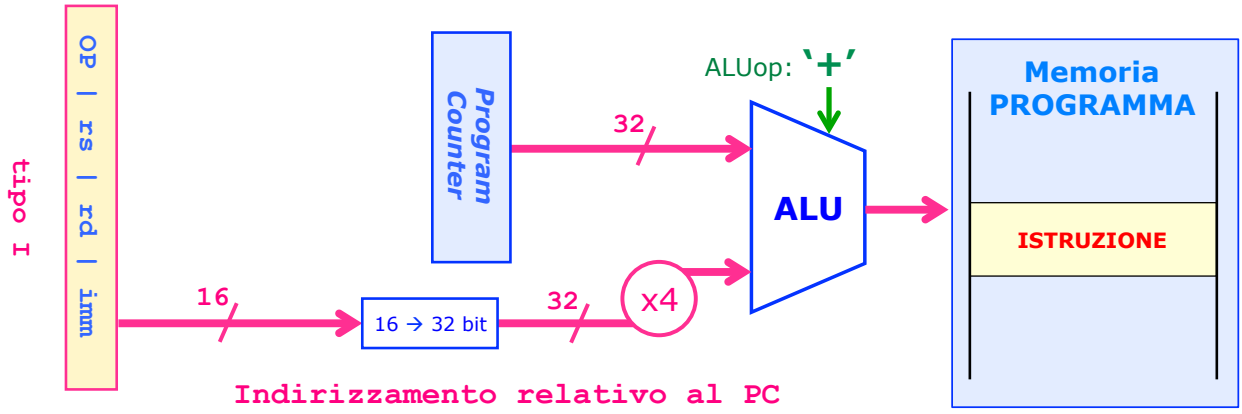
`Indirizzamento relativo al PC`



Indirizzamento base-spiazzamento

Esempio: operazioni di BRANCH – **beq/bne:**

Nome campo	op	rs	rt	indirizzo
Dimensione	6 bit	5 bit	5 bit	16 bit
`lw $5, 4($10)`	100100	01010	00101	0000 0000 0000 0100

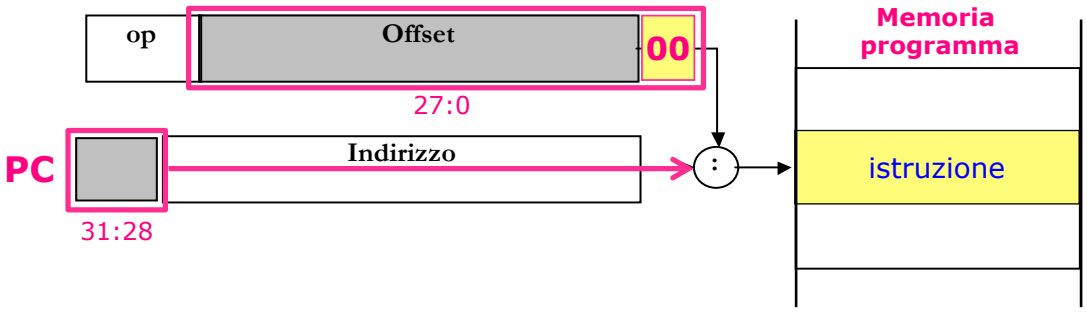


Indirizzamento pseudo-diretto

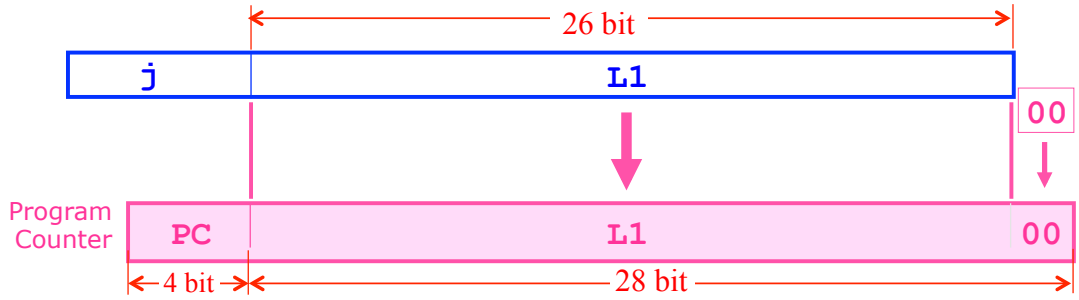
Indirizzamento PSEUDO–DIRETTO:

- Parte dell'operando (indirizzo) è un valore immediato contenuto nell'istruzione. La parte restante proviene dal Program Counter.
- L'indirizzo di salto si calcola:
  - Aggiungendo ai 26 bit di immediato altri due bit ("**00**") a destra (i 2 bit meno significativi) – ottenendo 28 bit.
  - Concatenando, a sinistra dei 28 bit, i 4 bit più significativi del PC.

Indirizzamento pseudo-diretto



Indirizzamento pseudo-diretto



- Le istruzioni che usano questo tipo di indirizzamento hanno **formato J**
  - Esempio: operazione di salto incondizionato (**jump**)

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