Lezione 17

Linguaggio macchina II: utilizzo di costanti, metodi di indirizzamento

Proff. A. Borghese, F. Pedersini

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

Sommario

- Utilizzo di costanti
- Metodi di indirizzamento

Utilizzo di costanti

- Spesso le operazioni richiedono l'uso di costanti
  - Esempio: somma del valore decimale 4 al contenuto di un registro
- Possibili 3 opzioni:
  - le costanti risiedono in memoria e sono caricate con **`lw`**
  - utilizzo di registri speciali (es: $zero)
  - utilizzare istruzioni di tipo I, in cui un operando è una costante: utilizzo di modalità di indirizzamento immediato
- Indirizzamento immediato:
  - la costante è memorizzata nel campo di 16 bit dedicato al dato immediato

Esempi

```
addi $s0, $s0, 4        # $s0 ← $s0 + 4 (sign-extended)
slti $t0, $s2, 10       # $t0 = 1 if $s2 < 10
andi $s0, $s0, 6        # $s0 ← $s0 and 6
ori $s0, $s0, 10        # $s0 ← $s0 or 10
li  $s0, 20             # $s0 ← 20 (pseudo-instruction)
```

- Le istruzioni di tipo I consentono di rappresentare costanti esprimibili in 16 bit
- I valori immediati possono essere espressi in Assembly:
  - in rappresentazione decimale: 65535_{10}
  - in rappresentazione esadecimale: 0xFF FF
  - in rappresentazione binaria: $1111\ 1111\ 1111\ 1111_2$

Gestione di costanti su 32-bit

- Le istruzioni di tipo I consentono di rappresentare costanti (immediate) esprimibili in 16 bit (max. 65535 unsigned).
- Se 16 bit non sono sufficienti per rappresentare la costante, l'assemblatore (o il compilatore) deve fare due passi:
  1. si utilizza l'istruzione **`lui`** (*load upper immediate*) per caricare i 16 bit più significativi della costante nei 16-bit più significativi di un registro. I 16-bit meno significativi del registro sono posti a 0.
  2. una successiva istruzione, ad esempio **`ori`** o **`addi`**, specifica i rimanenti 16 bit meno significativi della costante.
- Il registro **`$at (=$1)`** è riservato all'assemblatore per creare costanti su 32-bit.

Istruzione: `lui` (tipo I)

- L'istruzione ***load upper immediate***: **`lui rt, immval`**
  - carica i 16-bit del campo immediato nei 16-bit più significativi del registro **`rt`**.
  - I rimanenti 16-bit meno significativi del registro **`rt`** sono posti a 0.

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`lui $at, 61`	001111	00000	00001	0000 0000 0011 1101

- Pseudo-istruzione:
- load immediate: **`li rdest, immval`**
  - carica il valore imm nel registro **`rdest`**.

Assegnamento di costanti a 32 bit

ESEMPIO: si vuole assegnare la costante 400.000_{10} a $s0:

```
li $s0, 400000          # carica 400000 in $s0
```

- **400.000 = 0x 003C 0900 = 61 * 2^{16} + 2304**
- In binario, su 32 bit: 0000 0000 0011 1101 0000 1001 0000 000

```
lui $s0, 61             # 61   = 0000 0000 0011 1101
```

valore di $s0: 0000 0000 0011 1101 0000 0000 0000 0000

```
addiu $s0, $s0, 2304    # 2304 = 0000 1001 0000 0000
```

valore di $s0: 0000 0000 0011 1101 0000 1001 0000 0000

($400{,}000_{10}$)

Alternativa esadecimale:

```
lui $s0, 0x3C           # 0x3C = 0000 0000 0011 1101
addiu $s0, $s0, 0x900   # 0x900 = 0000 1001 0000 0000
```


Caricamento costante 32 bit - Esempio 2

- Si consideri la costante: 118345
  - decimale: **118345_{10}**
  - esadecimale: **0x1CE49**
  - binario: **0000 0000 0000 0001** **1100 1110 0100 1001**
    16-bit più significativi 16-bit meno significativi

HI: 0000 0000 0000 0001 corrispondenti al valore 1_{10}

LO: 1100 1110 0100 1001 corrispondenti al valore 52809_{10}

$$118345 = 1 \times 2^{16} + 52809$$

- Si consideri la pseudo-istruzione: **`li $t1, 118345`**
  - L'assemblatore la sostituisce con le seguenti istruzioni-macchina:

```
lui $at, 1              # $at: 0000 0000 0000 0001 0000 0000 0000 0000
ori $t1, $at, 52809     # $t1: 0000 0000 0000 0001 1100 1110 0100 1001
```

Sommario

- Utilizzo di costanti
- Modalità di indirizzamento

Modalità di indirizzamento

- *Nelle istruzioni di un processore, per modalità di indirizzamento si indicano le diverse modalità attraverso le quali si fa riferimento agli 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
```

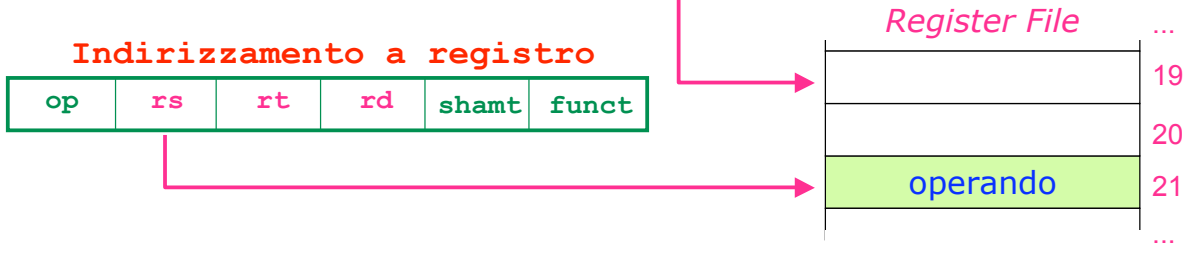
Modalità di indirizzamento

- 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

- Indirizzamento A REGISTRO: l'operando (l'indirizzo) è il contenuto di un registro della CPU, di cui l'istruzione contiene l'identificativo.
  - il nome (numero = indirizzo) 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



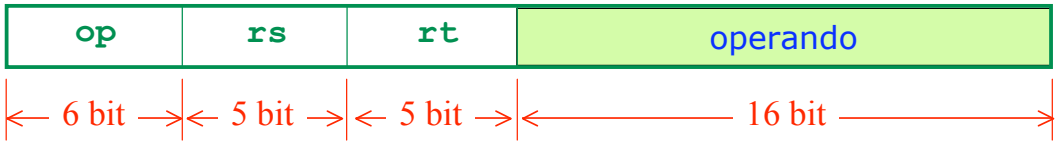
Indirizzamento immediato

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 indirizzamento immediato sono di **tipo I**
  - La costante è memorizzata nel campo operando (16-bit)

Indirizzamento immediato



Indirizzamento immediato

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

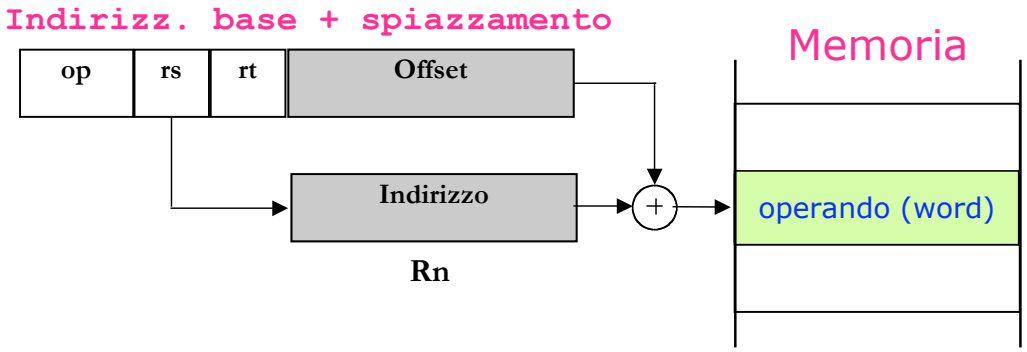
Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`addi $s1, $s1, 4`	001000	10001	10001	0000 0000 0000 0100

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

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

Indirizzamento con base e spiazzamento

- **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 costante (offset o spiazzamento) contenuto nell'istruzione
  - Le istruzioni con questo tipo di indirizzamento hanno **formato I**



Indirizzamento con base e spiazzamento

- Esempio:
  Istruzioni di load/store (formato I):

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`lw $t0, 32 ($s3)`	100011	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)`	101011	10011	01000	0000 0000 0010 0000

Indirizzamento con base e spiazzamento

- Esempi:
- istruzione di load

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

  - L'operando di trova in memoria all'indirizzo 32+[$s3]

- istruzione di store

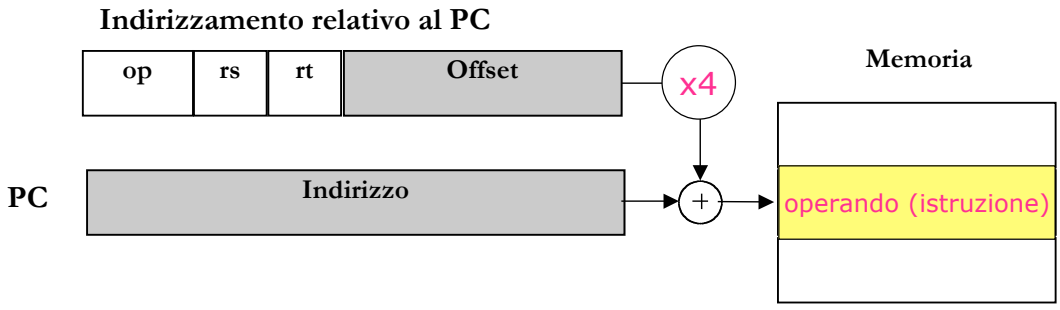
```
sw $t0, 32($s3)
```

  - L'operando viene copiato in memoria all'indirizzo 32+[$s3]
  - L'indirizzo e' espresso in byte

Indirizzamento relativo al PC

- **RELATIVO AL PROGRAM COUNTER:**
  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

  - Avendo a disposizione 16 bit di Offset ⇒ è possibile saltare in un range: $-2^{15} \div 2^{15}-1$ parole rispetto all'istruzione corrente



Indirizzamento relativo al PC

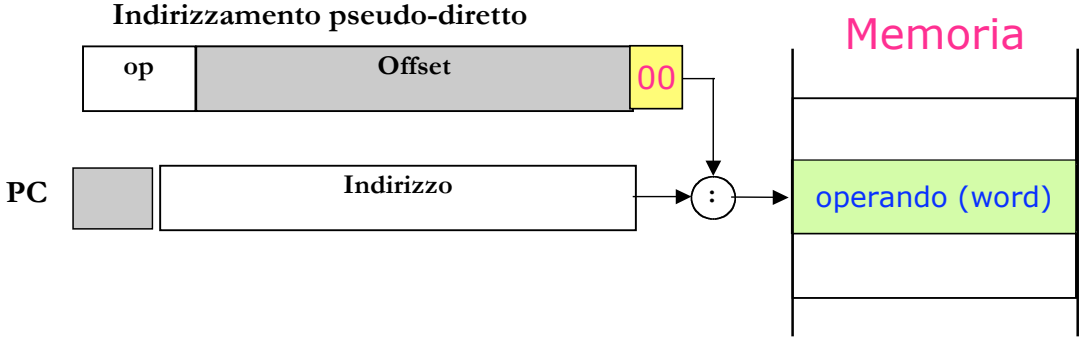
- Le istruzioni che usano questo tipo di indirizzamento hanno **formato I**
- **Esempio: BRANCH**

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

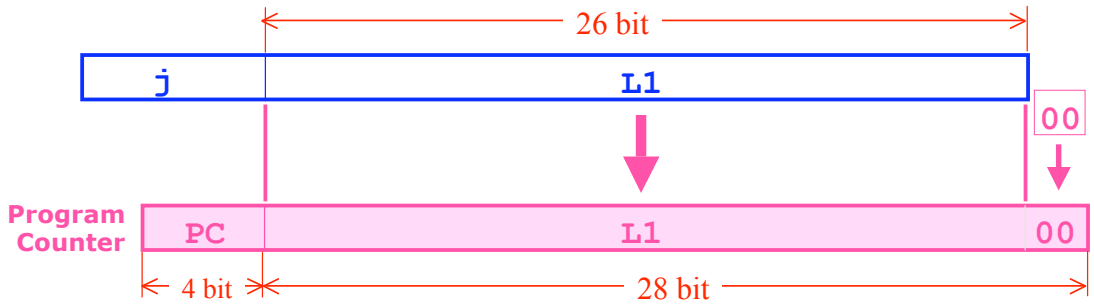
Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`bne $s1, $s2, 100`	000101	10001	10010	0000 0000 0001 1001

Indirizzamento pseudo-diretto

- **PSEUDO–DIRETTO:** Parte dell'operando (indirizzo) è presente come valore immediato nell'istruzione.
  La parte restante proviene da un registro (**Program Counter**)
- L'indirizzo di salto si calcola facendo:
  - Uno shift a sinistra di 2 bit dei 26 bit di offset contenuti nell'istru-zione (aggiungendo 00 nei bit meno significativi) – ottenendo 28 bit.
  - Concatenando ai 28 bit, i 4 bit più significativi del Program Counter.



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