MIPS & SPIM

Modulo del Corso di Architettura degli Elaboratori

Nicola Paoletti

Università di Camerino
Scuola di Scienze e Tecnologie

17 Aprile 2013
AA 2012/2013

Riepilogo

Tipi di istruzioni

- **Aritmetico-logiche:** effettuano operazioni aritmetico-logiche tra registri del processore (add, sub, and, or, . . .)
- **Trasferimento dati:** si occupano di trasferire dati tra la memoria e i registri del processore (lw, sw, . . .)
- **Controllo e salto:** questo tipo di istruzioni sono utilizzate per confrontare valori e per effettuare salti a particolari istruzioni (beq, bne, . . .)

Riepilogo

1. MIPS Instruction Set
   - Istruzioni aritmetiche
   - Istruzioni di trasferimento

Istruzioni aritmetiche

Esempio

```
add a,b,c # a=b+c
sub a,b,c # a=b-c
```

Le istruzioni aritmetiche sono ternarie (per mantenere l'hw semplice)

$$\text{OPERATORE op1 op2 op3.}$$
$$\Downarrow$$
$$\text{op1} = \text{op2 OPERATORE op3}$$

Nota: il carattere # è utilizzato per i commenti.

Istruzioni aritmetiche - Esercizio

Scrivere la seguente istruzione in linguaggio assembly MIPS

```
a = (b+c) - (d-f) +g
```

Istruzioni aritmetiche - Esercizio

Scrivere la seguente istruzione in linguaggio assembly MIPS

```
a = (b+c) - (d-f) +g
```

Risultato

```
add x,b,c
sub y,d,f
sub x,x,y
add a,x,g
```

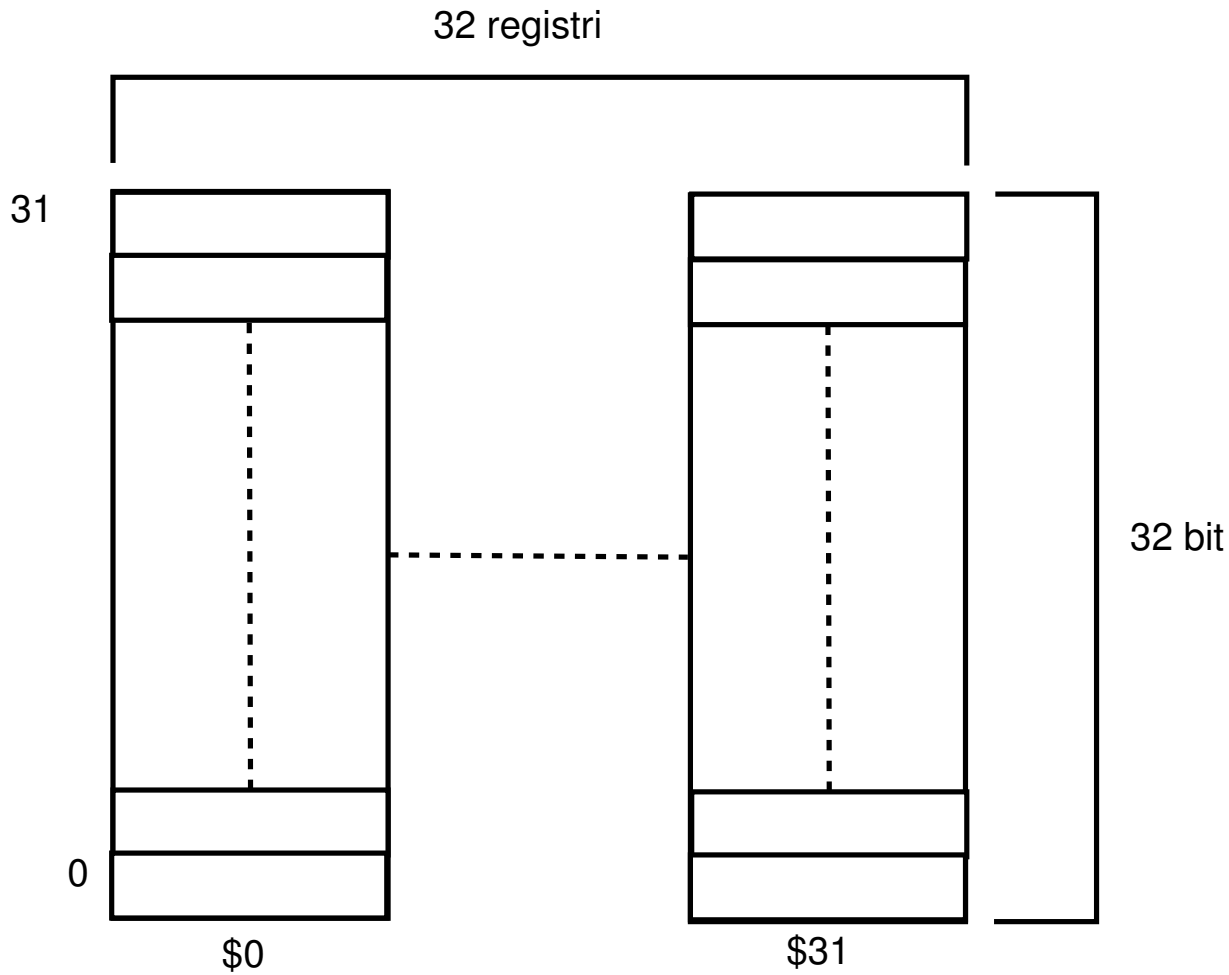
Registri (1/3)

Nelle istruzioni aritmetiche:
Operandi $\sim$ Registri

Non confondere variabili e registri:

- Variabili sono **infinite**
- Registri sono **finiti**

Registri (2/3)



Registri (3/3)

- **Program Counter (PC):**
  - Contiene l'indirizzo della prossima istruzione da eseguire
  - Incrementato ad ogni esecuzione di istruzione
  - Controlla il flusso del programma; viene manipolato nelle istruzioni di salto condizionato/incondizionato
- **32 registri general purpose:**
  - Numerati da 0 a 31 ($0-$31)
  - Hanno anche nomi simbolici (convenzionali)
  - Il registro $0 contiene sempre il valore **0** (forzato via hw)
  - Hanno lunghezza **32 bit** (1 word)
  - Supportano solo valori interi; valori decimali gestiti attraverso istruzioni e registri floating-point ($f0-$f31).

Registri - Convenzioni (1/2)

- $at ($1), $k0 ($26), $k1 ($27) sono riservati all'assembler e dal sistema operativo
- $v0 ($2) $v1 ($3) sono usati per restituire i valori delle funzioni
- $a0 - $a3 ($4 - $7) sono usati per passare alle procedure i primi 4 parametri (eventuali altri parametri vanno in memoria)
- $t0 - $t9 ($8 - $15, $24, $25) servono a memorizzare valori temporanei
- $s0 - $s7 ($16 - $23) sono registri temporanei, salvati in memoria per essere disponibili alle procedure chiamate

Registri - Convenzioni (2/2)

- $ra ($31) contiene l'indirizzo di ritorno da una procedura
- $fp ($30), frame pointer, punta alla prima parola del frame di memoria
- $sp ($29), stack pointer, punta all'ultima parola dello stack di memoria
- $gp $28), global pointer, punta alla metà del segmento di memoria riservato ai dati statici

Riepilogo

Istruzioni di trasferimento (1/5)

Si occupano di trasferire dati dalla memoria fisica ai registri (**load**), e viceversa (**store**).

Formato:

OPERATORE $registro indirizzo_memoria

Istruzioni di trasferimento (2/5)

Example

```
lw $t0, num1
# carica la parola (load word) puntata da num1 in $t0
sw $t0, num1
# salva la parola (store word) in $t0 all'indirizzo
num1
li $v0, 4
# carica la costante 4 in $v0 (load immediate)
la $a0, num1
# carica l'indirizzo (load address) num1 in $a0
```

Istruzioni di trasferimento (3/5)

Esercizio

Procedura MIPS per memorizzare una word da num1 a num2, lasciando una copia in $t0

Istruzioni di trasferimento (3/5)

Esercizio

Procedura MIPS per memorizzare una word da num1 a num2, lasciando una copia in $t0

Risultato

```
lw $t0, num1
sw $t0, num2
```

Istruzioni di trasferimento (4/5)

- La memoria fisica è indirizzata in byte
- Può essere vista come un vettore
- Le istruzioni aritmetiche richiedono che gli operandi siano memorizzati nei registri

Example

```
lw $s1, C($s2)
# carica in $s1 i 32 bit (parola) all'indirizzo di
memoria $s2 + C ($s1 := mem[$s2 + C])
sw $s1, C'($s2)
# salva la parola in $s1 all'indirizzo $s2 + C'
(mem[$s2 + C'] := $s1
```

Ci sono anche istruzioni per caricare/salvare double word, half word e singoli byte.

Istruzioni di trasferimento (5/5)

Example

```
lw $s1, C($s2)
# carica in $s1 i 32 bit (parola) all'indirizzo di
memoria $s2 + C ($s1 := mem[$s2 + C])
sw $s1, C'($s2)
# salva la parola in $s1 all'indirizzo $s2 + C'
(mem[$s2 + C'] := $s1
```

In particolare, C e C' sono gli indirizzi di richiesti a partire da un offset $ s2, che in genere si usa per puntare all'inizio o alla fine della memoria visibile ($ s2 = $ fp, $ sp)

lw vs la

Notare la differenza tra lw e la

Se Num1 sta alla locazione 0x10001000 e contiene 0xfffffffe (-2)

```
la $a0, Num1 # carica 0x10001000, mentre
lw $a0, Num1 # carica 0xfffffffe
```

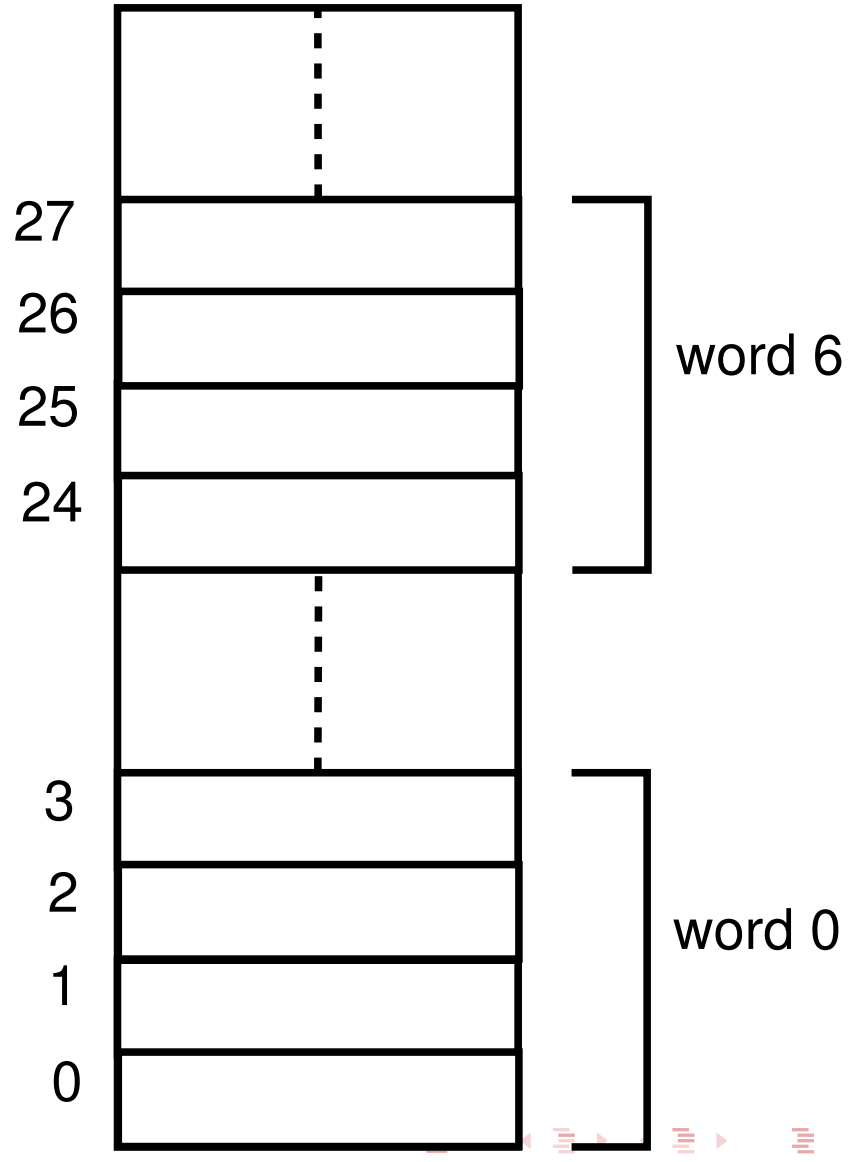
`la` si riferisce allindirizzo di `Num1`, mentre `lw` si riferisce al valore di `Num1`

Esempio

Supponiamo di voler caricare in $s1 la sesta parola nella memoria fisica (A[6])...

Esempio

Supponiamo di voler caricare in $s1 la sesta parola nella memoria fisica (A[6])...

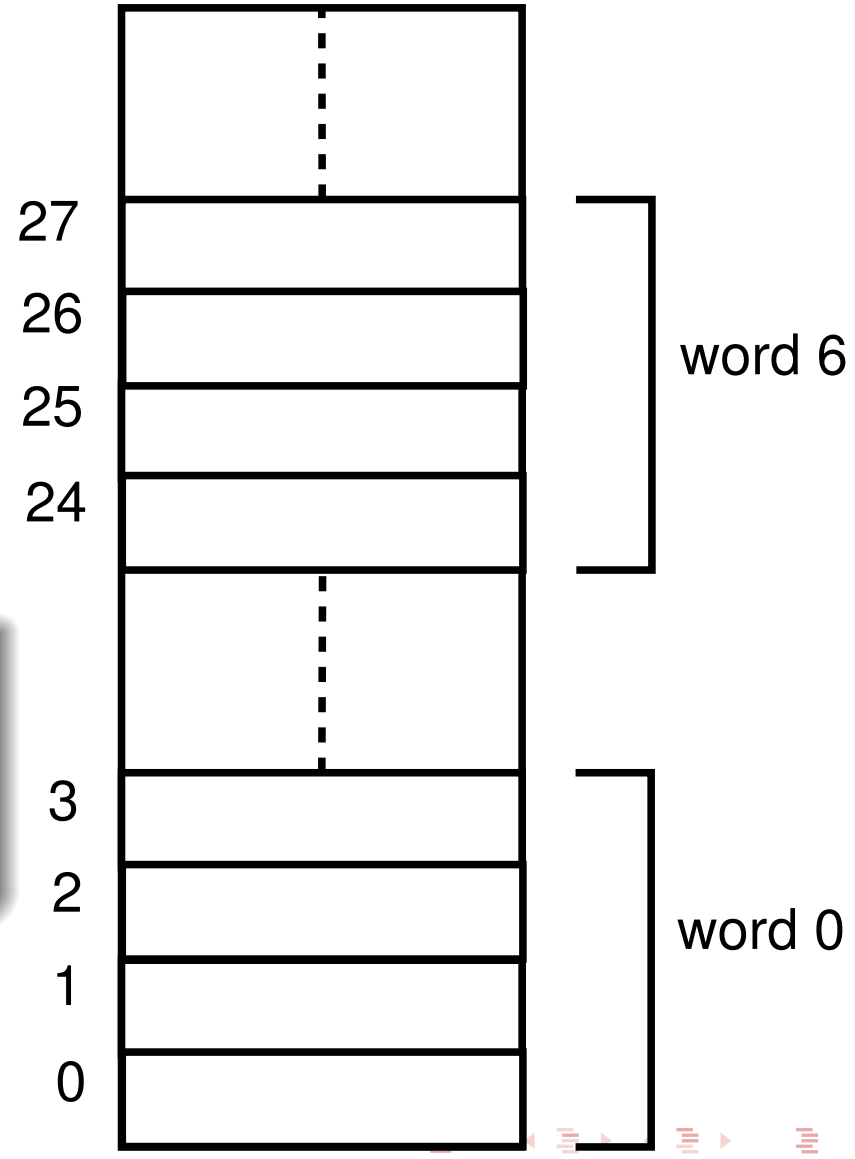


Esempio

Supponiamo di voler caricare in $s1 la sesta parola nella memoria fisica (A[6])...

Soluzione

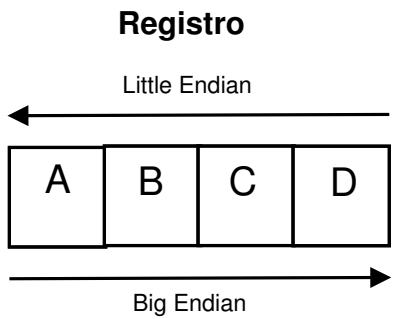
```
lw $s1, 24($s2) # $s2 punta
all'indirizzo del primo byte
in memoria - offset
```

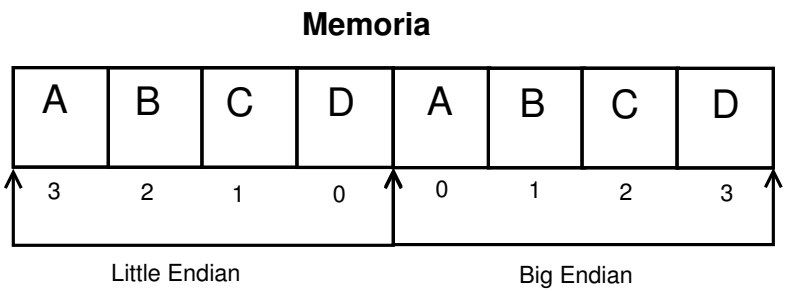


Little endian & Big Endian

Come sono memorizzati i valori nella memoria fisica? In che ordine?

- **Little Endian:** a partire dai byte meno significativi
- **Big Endian:** a partire dai byte più significativi





- Intel x86 utilizza Little Endian
- Power Architecture, ARM e SPARC utillizzano Big Endian
- **I processori MIPS sono in grado di operare con entrambi**

Riepilogo

1. MIPS Instruction Set
   - Istruzioni aritmetiche
   - Istruzioni di trasferimento

Esercizio1

Scrivere la procedura MIPS per calcolare:

$$\frac{(x + 5 - y) \cdot 35}{3}$$

x e y vanno caricate dalla memoria; moltiplicazione: **mul** a,b,c; divisione: **div** a,b,c.

Esercizio2

Scrivere in assembly la seguente procedura:

$$A[4] = b + A[i]$$

offset di A $\rightarrow$ \$s1; b $\rightarrow$ \$s2; i $\rightarrow$ \$s3