

Architettura degli Elaboratori

Tutorato

a cura di Manuela Flores

02/12/2022 ore 9 - 11:30



Barbara Masucci

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

Su cosa ci esercitiamo oggi?

(PRIMA PARTE)

- Prime istruzioni MIPS
 - Istruzioni aritmetiche/logiche
 - Istruzioni di trasferimento tra registri e memoria



Esercizio 1

➤ Scrivere il codice assembler MIPS corrispondente alla seguente istruzione in C:

$$a = b + (c / 2);$$

supponendo che valgano i seguenti assegnamenti:

- \$s0 dovrà contenere a
- \$s1 contiene b
- \$s2 contiene c

Si ricorda che sia b che c possono essere **negativi**



Lezione 12 pag. 26

Shift aritmetico

- Corrisponde ad inserire un certo numero n di bit, **tutti uguali al bit di segno**, a partire dalla posizione più significativa di una word, **traslandola a dx** di n posti
- Ad esempio, se il contenuto del registro $\$s0$ è


```
11111111111111111111111111111100
```

l'istruzione **sra \$t2, \$s0, 1** memorizza in $\$t2$

```
11111111111111111111111111111110
```

che corrisponde ad effettuare

```
11111111111111111111111111111100
```



Esercizio 1: Soluzione

➤ Il codice assembler MIPS per l'operazione di assegnamento $a = b + (c / 2)$; è

`sra $s0, $s2, 1` #il registro \$s0 contiene (c/2)

`add $s0, $s1, $s0` #il registro \$s0 contiene b+(c/2)

Si ricorda che valgono i seguenti assegnamenti:

\$s0 dovrà contenere a

\$s1 contiene b

\$s2 contiene c



Esercizio 2

➤ Scrivere il codice assembler MIPS corrispondente alla seguente istruzione in C:

`a = (b || c) && !d;`

supponendo che valgano i seguenti assegnamenti:

- \$s0 dovrà contenere a
- \$s1 contiene b
- \$s2 contiene c
- \$s3 contiene d



Esercizio 2: Soluzione

➤ Il codice assembler MIPS per l'operazione di assegnamento $a = (b \parallel c) \&\& !d$; è

```
or $t0, $s1, $s2 #il registro $t0 contiene (b||c)
nor $t1, $s3, $zero #il registro $t1 contiene (!d)
and $s0, $t0, $t1 #il registro $s0 contiene (b||c)&&!d
```

Si ricorda che valgono i seguenti assegnamenti:

\$s0 dovrà contenere a

\$s1 contiene b

\$s2 contiene c

\$s3 contiene d



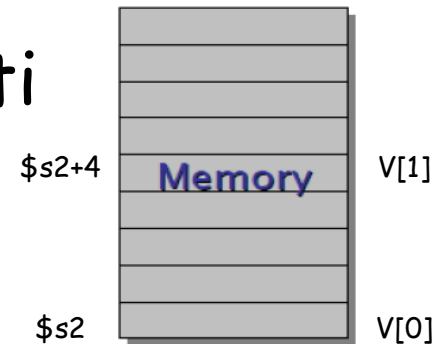
Esercizio 3

➤ Scrivere il codice assembler MIPS corrispondente alla seguente istruzione in C:

$$V[4] = 8 * a - (b + V[3]);$$

Supponendo che valgano i seguenti assegnamenti:

- \$s0 contiene a
- \$s1 contiene b
- \$s2 contiene indirizzo base vettore V



Esercizio 3: Soluzione

Il codice assembler MIPS per l'operazione di assegnamento $V[4] = 8 * a - (b + V[3]);$ è

```
lw $t0, 12($s2) #carica in $t0 la word che si trova in memoria
                  #all'indirizzo dato dal contenuto di $s2 + 12, cioè V[3]
add $t0, $s1, $t0 #il registro $t0 contiene b+V[3]
sll $s0, $s0, 3  #il registro $s0 contiene 8*a
sub $s0, $s0, $t0 #il registro $s0 contiene 8 * a - (b + V[3])
sw $s0, 16($s2)  #registra la word contenuta in $s0 nella locazione di
                  #memoria il cui indirizzo è dato dal contenuto di
                  # $s2 a cui va sommato 16, cioè V[4]
```



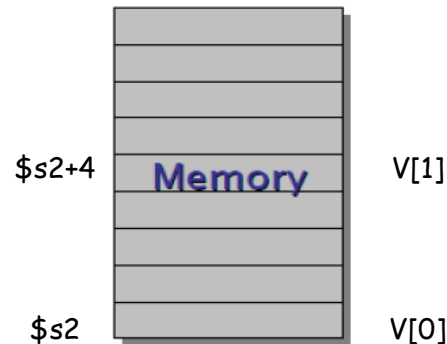
Esercizio 4

Supponendo che valgano i seguenti assegnamenti:

- \$s0 contiene i, \$s1 contiene j
- \$s2 contiene indirizzo base vettore V
- \$s3 contiene indirizzo base vettore W

scrivere il codice assembler MIPS corrispondente alla seguente istruzione C

$W[8] = V[i] + V[j-1];$



Esercizio 4: Soluzione

Supponendo che valgano i seguenti assegnamenti:

- \$s0 contiene i, \$s1 contiene j
- \$s2 contiene indirizzo base vettore V
- \$s3 contiene indirizzo base vettore W

scrivere il codice assembler MIPS corrispondente alla seguente istruzione C: $W[8] = V[i] + V[j-1]$;

sll \$t0, \$s0, 2

add \$t0, \$s2, \$t0

lw \$t1, 0(\$t0)

addi \$t2, \$s1, -1

sll \$t2, \$t2, 2

add \$t3, \$s2, \$t2

lw \$t4, 0(\$t3)

add \$t5, \$t1, \$t4

sw \$t5, 32(\$s3)

#il registro \$t0 contiene 4*i

#il registro \$t0 contiene \$s2+4*i

#carico nel registro \$t1 il contenuto di V[i]

#il registro \$t2 contiene j-1

#il registro \$t2 contiene 4*(j-1)

#il registro \$t3 contiene \$s2+4*(j-1)

#carico nel registro \$t4 il contenuto di V[j-1]

#il registro \$t5 contiene V[i]+V[j-1]

#salvo il contenuto di \$t5 nella locazione W[8]



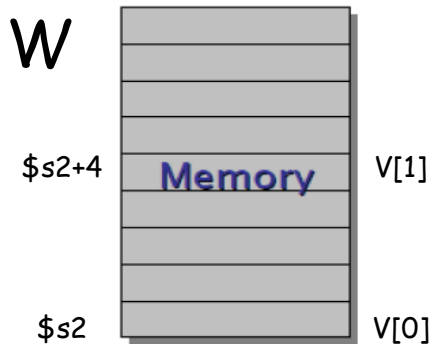
Esercizio 5

➤ Scrivere il codice assembler MIPS
corrispondente alla seguente istruzione in C

$W[i] = V[i+j];$

supponendo che valgano i seguenti assegnamenti:

- \$s0 contiene i
- \$s1 contiene j
- \$s2 contiene indirizzo base vettore V
- \$s3 contiene indirizzo base vettore W



Esercizio 5: Soluzione

Tenendo presente che

- \$s0 contiene i, \$s1 contiene j,
- \$s2 contiene indirizzo base V
- \$s3 contiene indirizzo base W
- Istruzioni assembler MIPS per $W[i] = V[i+j]$:
 - Innanzitutto notiamo che la word $V[i+j]$ si trova alla locazione di memoria con indirizzo $\$s2 + 4 \cdot (i+j)$
 - Dunque, memorizziamo in un registro temporaneo il valore $i+j$
 - `add $t0, $s0 $s1`
 - Poi effettuiamo uno shift logico a sinistra di 2 posizioni del registro \$t0, per ottenere il valore $4 \cdot (i+j)$
 - `sll $t1, $t0, 2`
 - Dunque, memorizziamo in un registro temporaneo il valore $\$s2 + 4 \cdot (i+j)$
 - `add $t2, $s2, $t1`



Esercizio 5: Soluzione

Tenendo presente che

- \$s0 contiene i, \$s1 contiene j,
- \$s2 contiene indirizzo base V
- \$s3 contiene indirizzo base W

➤ Istruzioni assembler MIPS per $W[i] = V[i+j]$:

- Poi carichiamo in un registro temporaneo la word che si trova in memoria alla locazione data dal contenuto di \$t2, cioè $V[i+j]$
 - `lw $t3, 0($t2)`
- Poi effettuiamo uno shift logico a sinistra di 2 posizioni del registro \$s0, per ottenere il valore $4*(i)$
 - `sll $t4, $s0, 2`
- Dunque, memorizziamo in un registro temporaneo il valore $\$s3 + 4*(i)$
 - `add $t5, $s3, $t4`
- Infine, registriamo la word contenuta in \$t3, cioè $V[i+j]$ nella locazione di memoria $W[i]$, il cui indirizzo è contenuto in \$t5
 - `sw $t3, 0($t5)`



Esercizio 5: Soluzione

Mettendo tutto insieme, il codice assembler MIPS per l'operazione di assegnamento $W[i] = V[i+j]$; è

```
add $t0, $s0, $s1 # $t0 contiene i+j
sll $t1, $t0, 2   # $t1 contiene 4*(i+j)
add $t2, $s2, $t1 # $t2 contiene l'indirizzo base di V sommato
                  # a 4*(i+j)
lw  $t3, 0($t2)   # carica in $t3 la word che si trova in memoria
                  # all'indirizzo dato dal contenuto di $t2, cioè V[i+j]
sll $t4, $s0, 2   # $t4 contiene 4*(i)
add $t5, $s3, $t4 # $t5 contiene l'indirizzo base di W sommato
                  # a 4*(i)
sw  $t3, 0($t5)   # registra la word contenuta in $t3 nella locazione di
                  # memoria il cui indirizzo è dato dal contenuto di
                  # $s3 a cui è stato sommato 4*(i), cioè W[i]
```



Esercizio 6

Supponendo che il contenuto dei registri \$s0, \$s1 siano rispettivamente 5 e 1024 e il contenuto della memoria all'indirizzo 1028 sia 10, quali saranno rispettivamente i contenuti di \$s0, \$s1 e della memoria all'indirizzo 1028, dopo l'esecuzione della seguente sequenza di istruzioni?

```
sw $s0,4($s1)  
lw $s0,4($s1)
```

- A. 5, 1024, 5
- B. 5, 1024, 10
- C. I dati sono insufficienti per poter rispondere
- D. Nessuna delle risposte precedenti



Esercizio 6: Soluzione

Supponendo che il contenuto dei registri \$s0, \$s1 siano rispettivamente 5 e 1024 e il contenuto della memoria all'indirizzo 1028 sia 10, quali saranno rispettivamente i contenuti di \$s0, \$s1 e della memoria all'indirizzo 1028, dopo l'esecuzione della seguente sequenza di istruzioni?

```
sw $s0,4($s1)  
lw $s0,4($s1)
```

A. 5, 1024, 5

B. 5, 1024, 10

C. I dati sono insufficienti per poter rispondere

D. Nessuna delle risposte precedenti



Esercizio 7

Supponendo che valgano i seguenti assegnamenti:

- \$s0 contiene i, \$s1 contiene j
- \$s2 contiene indirizzo base vettore V
- \$s3 contiene indirizzo base vettore W

scrivere il codice C corrispondente al seguente frammento di codice assembler MIPS

```
sll $t0, $s0, 2  
add $t1, $s2, $t0  
lw $t2, 0($t1)  
addi $t3, $t1, -8  
lw $t4, 0($t3)  
add $t5, $t2, $t4  
sll $t6, $s1, 2  
add $t7, $s3, $t6  
addi $t8, $t7, 4  
sw $t5, 0($t8)
```



Esercizio 7: Soluzione

Supponendo che valgano i seguenti assegnamenti:

- `$s0` contiene `i`, `$s1` contiene `j`
- `$s2` contiene indirizzo base vettore `V`
- `$s3` contiene indirizzo base vettore `W`

scrivere il codice `C` corrispondente al seguente frammento di codice assembler MIPS

`sll $t0, $s0, 2`

`#$t0 contiene $i \cdot 4$`

`add $t1, $s2, $t0`

`#$t1 contiene indirizzo base $V + i \cdot 4$`

`lw $t2, 0($t1)`

`#carica in $t2 la word $V[i]$`

`addi $t3, $t1, -8`

`#$t3 contiene indirizzo base $V + i \cdot 4 - 8$`

`lw $t4, 0($t3)`

`#$carica in $t4 la word $V[i-2]$`

`add $t5, $t2, $t4`

`#$t5 contiene $V[i] + V[i-2]$`

`sll $t6, $s1, 2`

`#$t6 contiene $j \cdot 4$`

`add $t7, $s3, $t6`

`#$t7 contiene indirizzo base $W + j \cdot 4$`

`addi $t8, $t7, 4`

`#$t8 contiene indirizzo base $W + j \cdot 4 + 4$`

`sw $t5, 0($t8)`

`#registra $V[i] + V[i-2]$ in $W[j+1]$`



Esercizio 7: Soluzione

`sll $t0, $s0, 2` $\# \$t0$ contiene $i*4$
`add $t1, $s2, $t0` $\# \$t1$ contiene indirizzo base $V + i*4$
`lw $t2, 0($t1)` $\#$ carica in $\$t2$ la word $V[i]$
`addi $t3, $t1, -8` $\# \$t3$ contiene indirizzo base $V + i*4 - 8$
`lw $t4, 0($t3)` $\#$ carica in $\$t4$ la word $V[i-2]$
`add $t5, $t2, $t4` $\# \$t5$ contiene $V[i] + V[i-2]$
`sll $t6, $s1, 2` $\# \$t6$ contiene $j*4$
`add $t7, $s3, $t6` $\# \$t7$ contiene indirizzo base $W + j*4$
`addi $t8, $t7, 4` $\# \$t8$ contiene indirizzo base $W + j*4 + 4$
`sw $t5, 0($t8)` $\#$ registra $V[i] + V[i-2]$ in $W[j+1]$

Il codice C corrispondente è

$W[j+1] = V[i] + V[i-2];$




Su cosa ci esercitiamo oggi?

(SECONDA PARTE)

- Ancora istruzioni MIPS
 - Istruzioni per prendere decisioni
 - Cicli



Esercizio 1

- 
- Scrivere il codice assembler MIPS corrispondente alla seguente istruzione in C:

`if (a!=b) d = c+2; else d = c*4;`

Si suppone che:

- \$s0 contiene a
- \$s1 contiene b
- \$s2 contiene c
- \$s3 contiene d



Lez. 13 pag. 3

Prendere decisioni

- Talvolta si ha la necessità di alterare il flusso di un programma **al verificarsi di certe condizioni**
- Il MIPS supporta questa necessità fornendo **istruzioni di salto condizionato**
 - **Branch if Equal: `beq reg1, reg2, L1`**
 - Salta all'istruzione con etichetta L1 se il contenuto del registro reg1 è uguale al contenuto del registro reg2
 - **Branch if Not Equal: `bne reg1, reg2, L1`**
 - Salta all'istruzione con etichetta L1 se il contenuto del registro reg1 è diverso dal contenuto del registro reg2
- Inoltre, il MIPS offre una istruzione di **salto incondizionato**
 - **Jump: `j L1`**
 - Salta all'istruzione con etichetta L1



Esercizio 1: Soluzione

- Il codice assembler MIPS dell'istruzione
`if (a!=b) d = c+2; else d = c*4; è`



\$s0 contiene a, \$s1 contiene b
\$s2 contiene c, \$s3 contiene d

Esercizio 1: Soluzione

- Il codice assembler MIPS dell'istruzione
`if (a!=b) d = c+2; else d = c*4; è`

```
beq $s0, $s1, ELSE #salta a ELSE se a = b  
addi $s3, $s2, 2 #calcola d=c+2 (se a ≠ b)  
j ESCI #vai a ESCI
```

```
ELSE: sll $s3, $s2, 2 #calcola d=c*4 (se a = b)
```

```
ESCI: ... #segna la fine della decisione
```



\$s0 contiene a, \$s1 contiene b
\$s2 contiene c, \$s3 contiene d

Esercizio 2

➤ Scrivere il codice assembler MIPS corrispondente alla seguente istruzione in C:

`if (a>b) d = c+2; else d = c*3;`

Si suppone che:

- \$s0 contiene a
- \$s1 contiene b
- \$s2 contiene c
- \$s3 contiene d



Lez. 13 pag. 19

Altri confronti

- Il MIPS offre anche il confronto "un operando è **minore** di un altro o di una costante?"
 - Il confronto avviene mediante istruzioni che settano a 1 il valore di **registri di flag**
 - **Set if Less Than**
 - `slt $t0, $s3, $s4` #setta \$t0=1 se il contenuto di \$s3
#è minore del contenuto di \$s4
 - `slti $t0, $s3, 10` #setta \$t0=1 se il contenuto di \$s3
#è minore di 10
 - I salti su condizioni di tipo minore uguale o maggiore uguale si ottengono
 - Combinando **slt** con **beq** o **bne** e
 - Usando la costante 0 (disponibile nel registro **\$zero**) per fare i test



Esercizio 2: Soluzione

- Il codice assembler MIPS dell'istruzione
`if (a>b) d = c+2; else d = c*3; è`



\$s0 contiene a, \$s1 contiene b
\$s2 contiene c, \$s3 contiene d

Esercizio 2: Soluzione

- Il codice assembler MIPS dell'istruzione
if ($a > b$) $d = c + 2$; else $d = c * 3$; è

```
slt $t0, $s1, $s0 #setta $t0=1 se  $b < a$  (cioè se  $a > b$ )  
beq $t0, $zero, ELSE #salta a ELSE se  $b \geq a$  ( $a \leq b$ )  
addi $s3, $s2, 2 #calcola  $d = c + 2$  (se  $a > b$ )  
j ESCI #vai a ESCI
```

```
ELSE: sll $s3, $s2, 2 #calcola  $d = c * 4$  (se  $a \leq b$ )  
sub $s3, $s3, $s2 #calcola  $d = d - c$ , cioè  $d = c * 3$ 
```

```
ESCI: ... #segna la fine della decisione
```



$\$s0$ contiene a , $\$s1$ contiene b
 $\$s2$ contiene c , $\$s3$ contiene d

Esercizio 3

➤ Scrivere il codice assembler MIPS corrispondente alla seguente istruzione in C:

`if (a<=b) d = c*3; else d = c*4;`

Si suppone che:

- \$s0 contiene a
- \$s1 contiene b
- \$s2 contiene c
- \$s3 contiene d



N.B. La soluzione più efficiente ha solo 4 istruzioni + l'etichetta di uscita

Esercizio 3

- Il codice assembler MIPS dell'istruzione
`if (a<=b) d = c*3; else d = c*4; è`



\$s0 contiene a, \$s1 contiene b
\$s2 contiene c, \$s3 contiene d

Esercizio 3: Soluzione A

- Il codice assembler MIPS dell'istruzione
`if (a<=b) d = c*3; else d = c*4; è`



\$s0 contiene a, \$s1 contiene b
\$s2 contiene c, \$s3 contiene d

Esercizio 3: Soluzione A

- Il codice assembler MIPS dell'istruzione
if ($a \leq b$) $d = c * 3$; else $d = c * 4$; è

```
slt $t0, $s1, $s0 #setta $t0=1 se  $b < a$  (cioè se  $a > b$ )  
bne $t0, $zero, ELSE #salta a ELSE se  $\$t0=1$  ( $a > b$ )  
sll $s3, $s2, 2 #calcola  $d=c*4$  (se  $a \leq b$ )  
sub $s3, $s3, $s2 #calcola  $d=d-c$ , cioè  $d=c*3$   
j ESCI #vai a ESCI
```

ELSE: sll \$s3, \$s2, 2 #calcola $d=c*4$ (se $a > b$)

ESCI: ... #segna la fine della decisione



$\$s0$ contiene a , $\$s1$ contiene b
 $\$s2$ contiene c , $\$s3$ contiene d

Esercizio 3: Soluzione B

- Il codice assembler MIPS dell'istruzione
`if (a ≤ b) d = c*3; else d = c*4; è`

```
slt $t0, $s1, $s0 #setta $t0=1 se b < a (cioè se a > b)
beq $t0, $zero, THEN #salta a THEN se $t0=0 (a ≤ b)
sll $s3, $s2, 2 #calcola d=c*4 (se a > b)
j ESCI #vai a ESCI
```

```
THEN: sll $s3, $s2, 2 #calcola d=c*4 (se a ≤ b)
      sub $s3, $s3, $s2 #calcola d=d-c, cioè d=c*3
```

```
ESCI: ... #segna la fine della decisione
```



\$s0 contiene a, \$s1 contiene b
\$s2 contiene c, \$s3 contiene d

Esercizio 3: Soluzione C

- Il codice assembler MIPS dell'istruzione
`if (a<=b) d = c*3; else d = c*4; è`

```
sll $s3, $s2, 2 #calcola d=c*4 (in qualunque caso)
slt $t0, $s1, $s0 #setta $t0=1 se b < a (cioè se a > b)
beq $t0, $zero, THEN #salta a THEN se $t0=0 (a ≤ b)
j ESCI #vai a ESCI
```

THEN: `sub $s3, $s3, $s2 #calcola d=d-c, cioè d=c*3`

ESCI: `... #segna la fine della decisione`



\$s0 contiene a, \$s1 contiene b
\$s2 contiene c, \$s3 contiene d

Esercizio 3: Soluzione D

- Il codice assembler MIPS dell'istruzione
if ($a \leq b$) $d = c * 3$; else $d = c * 4$; è

```
sll $s3, $s2, 2 #calcola  $d = c * 4$  (in qualunque caso)
slt $t0, $s1, $s0 #setta  $\$t0 = 1$  se  $b < a$  (cioè se  $a > b$ )
bne $t0, $zero, ESCI #salta a ESCI se  $\$t0 = 1$  ( $a > b$ )
sub $s3, $s3, $s2 #calcola  $d = d - c$ , cioè  $d = c * 3$ 
ESCI: ... #segna la fine della decisione
```



$\$s0$ contiene a , $\$s1$ contiene b
 $\$s2$ contiene c , $\$s3$ contiene d

Esercizio 4

➤ Scrivere il codice assembler MIPS corrispondente alla seguente istruzione in C:

```
i=1;  
j=V[7];  
while (i != j) i=7*i;
```

Si suppone che:

- \$s0 contiene i
- \$s1 contiene j
- \$s2 contiene indirizzo base vettore V



Esercizio 4: Soluzione

- Il codice assembler MIPS è

| | |
|------------------------------------|---|
| <code>i=1;</code> | <code>\$s0</code> contiene <code>i</code> |
| <code>j=V[7];</code> | <code>\$s1</code> contiene <code>j</code> |
| <code>while (i != j) i=7*i;</code> | <code>\$s2</code> contiene indirizzo base di <code>V</code> |



Esercizio 4: Soluzione

➤ Il codice assembler MIPS è

```
addi $s0, $zero, 1 #assegna ad i il valore 1
lw $s1, 28($s2) #carica in $s1 il contenuto di V[7]
CICLO: beq $s0, $s1, ESCI #salta ad ESCI se i==j
sll $t0, $s0, 3 #calcola $t0=i*8
sub $s0, $t0, $s0 #calcola i=$t0-i, cioè i=7*i
j CICLO #vai a CICLO
ESCI: ... #segna la fine del ciclo
```

i=1;

j=V[7];

while (i != j) i=7*i;

\$s0 contiene i

\$s1 contiene j

\$s2 contiene indirizzo base di V



Esercizio 5

➤ Scrivere il codice assembler MIPS corrispondente alla seguente istruzione in C:

```
for (i=0; i<50; i++) {  
    somma += V[i];  
}
```

Si suppone che:

- \$s0 contiene i
- \$s1 contiene somma
- \$s2 contiene indirizzo base vettore V



Esercizio 5: Soluzione

- Il codice assembler MIPS è

```
for (i=0; i<50; i++){  
    somma += V[i];  
}
```

\$s0 contiene i
\$s1 contiene somma
\$s2 contiene indirizzo base di V



Esercizio 5: Soluzione A

➤ Il codice assembler MIPS può essere:

```
    addi $s0, $zero, 0 #assegna ad i il valore 0
CICLO: slti $t0, $s0, 50 #setta $t0=1 se $s0 < 50 (se i<50)
    beq $t0, $zero, ESCI #se $t0≠1 ($t0=0, cioè i≥50) allora salta a ESCI
    sll $t1, $s0, 2 #il registro $t1 contiene 4*i
    add $t2, $s2, $t1 #indirizzo di V+4*i
    lw $t3, 0($t2) #carica in $t3 la word V[i]
    add $s1, $s1, $t3 #somma=somma+V[i]
    addi $s0, $s0, 1 #i=i+1
    j CICLO
```

```
ESCI: for (i=0; i<50; i++){
        somma += V[i];
    }
```

\$s0 contiene i
\$s1 contiene somma
\$s2 contiene indirizzo base di V



Esercizio 5: Soluzione B

- Oppure, il codice assembler MIPS può avere due istruzioni in meno:

```
addi $s0, $zero, 0 #assegna ad i il valore 0
CICLO: lw $t0, 0($s2) #carica in $t0 il contenuto di V[i]
add $s1, $s1, $t0 #somma=somma+V[i]
addi $s2, $s2, 4 #indirizzo di V+4 (word successiva)
addi $s0, $s0, 1 #i=i+1
slti $t1, $s0, 50 #setta $t1=1 se $s0 < 50 (se i<50)
bne $t1, $zero, CICLO #se $t1≠0 (cioè se i<50)
                        #salta a CICLO

for (i=0; i<50; i++){
    somma += V[i];
}
```

\$s0 contiene i

\$s1 contiene somma

\$s2 contiene indirizzo base di V



Riepilogo

➤ Istruzioni MIPS

- Istruzioni aritmetiche/logiche
- Istruzioni di trasferimento tra registri e memoria
- Istruzioni per prendere decisioni
- Cicli

➤ Prossima lezione:

- 9 dicembre 2022 ore 9 - 11:30, stesso Team

