

Calcolatori Elettronici

Esercitazione 1

M. Sonza Reorda – M. Monetti

M. Rebaudengo – R. Ferrero

L. Sterpone – M. Grosso

Politecnico di Torino

Dipartimento di Automatica e Informatica

Obiettivi

- Assegnazione di valori a registri e in memoria
- Operazioni aritmetiche: ADD e SUB
 - con segno/senza segno
 - tra due registri o tra registro e immediato
- Istruzioni di input/output
 - lettura di un intero inserito da tastiera
 - stampa a video di interi e stringhe

Esercizio 1

- Siano date le seguenti variabili di tipo byte già inizializzate in memoria:
 - `n1: .byte 10`
 - `n2: .byte 0x10`
 - `n3: .byte '1'`
- Sia inoltre definita la variabile di tipo byte, non inizializzata, `res`
- Si calcoli la seguente espressione e si verifichi il risultato: $res = n1 - n2 + n3$

Soluzione

```
        .data
n1:      .byte 10
n2:      .byte 0x10
n3:      .byte '1'
res:     .space 1
        .text
        .globl main
        .ent main
main:    lb $t0, n1
        lb $t1, n2
        sub $t0, $t0, $t1
        lb $t1, n3
        add $t0, $t0, $t1
        sb $t0, res
        li $v0, 10
        syscall
        .end main
```

Esercizio 2

- Siano date cinque variabili di tipo byte:

`var1 = 'm', var2 = 'i', var3 = 'p', var4 = 's', var5 = 0`

- Si scriva un programma che converta in maiuscolo le prime 4 variabili.
- Successivamente, stampare una stringa utilizzando la system call 4 e copiando in `$a0` l'indirizzo di `var1`.
- Quali sono i caratteri stampati a video? A cosa serve `var5`?

Soluzione

```
.data
var1:  .byte 'm'
var2:  .byte 'i'
var3:  .byte 'p'
var4:  .byte 's'
var5:  .byte 0

.text
.globl main
.ent main
main:  li $t0, 'A'
      li $t1, 'a'
      sub $t0, $t0, $t1
      lb $t1, var1      # conversione prima variabile
      add $t1, $t1, $t0
      sb $t1, var1
```

Soluzione

```
lb $t1, var2      # conversione seconda variabile
add $t1, $t1, $t0
sb $t1, var2
lb $t1, var3      # conversione terza variabile
add $t1, $t1, $t0
sb $t1, var3
lb $t1, var4      # conversione quarta variabile
add $t1, $t1, $t0
sb $t1, var4
la $a0, var1      # stampa
li $v0, 4
syscall
li $v0, 10
syscall
.end main
```

Esercizio 3

- Siano date le seguenti variabili di tipo byte inizializzate in memoria:
 - `op1: .byte 150`
 - `op2: .byte 100`
- Si stampi a video la somma delle due variabili, utilizzando la system call 1, e si verifichi che il risultato sia corretto.

Soluzione

```
.data
op1:  .byte 150
op2:  .byte 100

.text
.globl main
.ent main
main:  lbu $t0, op1
      lb $t1, op2      # equivalente a lbu $t1, op2
      add $a0, $t0, $t1
      li $v0, 1
      syscall
      li $v0, 10
      syscall
      .end main
```

Esercizio 4

- Sia data la seguente variabili di tipo word inizializzata in memoria:
`var: .word 0x3FFFFFFF0`
- Si memorizzi nel registro `$t1` il doppio del valore di `var` e poi lo si stampi a video.
- Aggiungere a `$t1` il valore immediato 40 (usando un altro registro come destinazione per non modificare `$t1`). Cosa accade? E' possibile stampare un risultato numerico?

Esercizio 4 (cont.)

- Ripetere l'operazione precedente, ma questa volta porre 40 nel registro $\$t2$ e poi sommare $\$t1$ e $\$t2$. E' possibile stampare a video un risultato numerico?

Soluzione

```
var:      .data
          .word 0x3FFFFFF0
          .text
          .globl main
          .ent main
main:     lw $t0, var
          add $t1, $t0, $t0    # prima somma
          move $a0, $t1
          li $v0, 1
          syscall
          addiu $a0, $t1, 40   # seconda somma
          li $v0, 1
          syscall
          #addi $a0, $t1, 40   #scatena l'eccezione di overflow
          #li $v0, 1
          #syscall
```

Soluzione

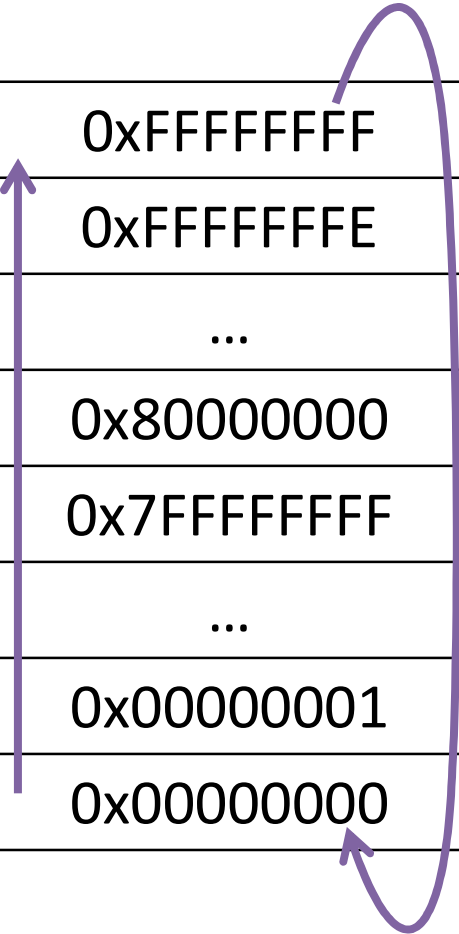
```
li $t2, 40          # terza somma
addu $a0, $t1, $t2
li $v0, 1
syscall
```

```
#add $a0, $t1, $t2  # scatena l'eccezione di overflow
#li $v0, 1
#syscall
```

```
li $v0, 10
syscall
.end main
```

Rappresentazione dei numeri

Complemento a 2		Binario puro
-1	0xFFFFFFFF	4.294.967.296
-2	0xFFFFFFFEE	4.294.967.295
	...	
-2.147.483.648	0x80000000	2.147.483.648
2.147.483.647	0x7FFFFFFF	2.147.483.647
	...	
1	0x00000001	1
0	0x00000000	0



Verifica dell'overflow in Ca2

- Sommando 1 a 0x7FFFFFFF, il risultato è corretto solo se considerato in binario puro.
- `ADDU` e `ADDIU` suppongono che i numeri siano senza segno ed effettuano la somma
- `ADD` e `ADDI` suppongono che i numeri siano in complemento a 2 e scatenano un'eccezione.
- Per verificare l'overflow in `Ca2`, si usano `ADD` e `ADDI`, oppure si confronta il segno del risultato con quello degli operandi: se la somma di due operandi con lo stesso segno produce un risultato di segno opposto, c'è overflow.

Verifica dell'overflow in binario puro

- Sommando 1 a 0xFFFFFFFF, il risultato è corretto solo considerato in complemento a 2.
- `ADD`, `ADDI`, `ADDU` e `ADDIU` si comportano allo stesso modo: effettuano la somma e non scatenano nessuna eccezione (suppongono che i numeri siano in complemento a 2).
- Per verificare l'overflow in binario puro, si deve controllare che il risultato sia maggiore degli operandi.

Esercizio 5

- Utilizzando la system call 5, leggere un intero introdotto tramite tastiera e salvarlo in $\$t1$.
- Leggere un altro intero e salvarlo in $\$t2$.
- Senza utilizzare altri registri, scambiare il valore di $\$t1$ e $\$t2$.
- Suggerimento: utilizzare istruzioni di somma e sottrazione.

Soluzione

```
        .data
message: .asciiz "Inserisci un numero: "

        .text
        .globl main
        .ent main
main:    la $a0, message           #lettura primo numero
        li $v0, 4
        syscall
        li $v0, 5
        syscall
        move $t1, $v0
```

Soluzione

```
la $a0, message           #lettura secondo numero
li $v0, 4
syscall
li $v0, 5
syscall
move $t2, $v0

#scambia $t1 e $t2 senza usare altri registri
add $t1, $t1, $t2
sub $t2, $t1, $t2
sub $t1, $t1, $t2

li $v0, 10
syscall
.end main
```