

# Esercitazioni - Assembler MIPS - Architettura dei calcolatori e sistemi operativi a.a. 2016/2017

Architettura dei calcolatori e sistemi operativi (Politecnico di Milano)



Scan to open on Studocu

### Assembler MIPS - Esercizi di base

# Negli esercizi seguenti:

- NO PSEUDO significa che è vietato utilizzare pseudo-istruzioni
- MAX N significa che è permesso utilizzare al massimo N istruzioni
- 1) Scrivere una sequenza di istruzioni MIPS che scrive il valore –1 nel registro s6 se il bit in settima posizione (da destra) del registro s0 vale 1 (altrimenti il registro s6 rimane inalterato). Si suppone che i primi 16 bit di s0 siano tutti a 0.

```
NO PSEUDO, MAX 3
```

### Soluzione

```
0x0040 // in binario 0000 0000 0100 0000
      $t1, $s0,
andi
      $t1, $zero, FALSO
beq
      $s6, $zero, -1
addi
```

### FALSO:

2) Scrivere una sequenza di istruzioni che scrive il valore -1 nel registro s6 se il bit in decima posizione (da sinistra) del registro s0 vale 1 (altrimenti il registro s6 rimane inalterato).

```
NO PSEUDO, MAX 5
```

# Soluzione

```
// in binario 0000 0000 0100 0000
lui
      $at, 0x0040
ori
      $t1, $at,
                   0x0000
and
      $t1, $s0,
                   $t1
beq
      $t1, $zero, FALSO
addi
      $s6, $zero, -1
```

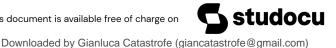
### FALSO:

3) Scrivere una sequenza di istruzioni che assegna i valori 1, 2 o 3 al registro s0 in base al risultato del confronto tra i registri t1 e t0, che contengono numeri relativi, secondo lo schema seguente:  $t1 < t0 \rightarrow 1$ ,  $t1 > t0 \rightarrow 2$  e  $t1 = t0 \rightarrow 3$ 

MAX 7

# Soluzione

```
bne
                $t1, $t0,
                             NE
          addi
                $s0, $zero, 3
                                    // eq
                FINE
          blt
                 $t1, $t0,
NE:
          addi
                $s0, $zero, 2
                                    // gt
                FINE
          j
                $s0, $zero, 1
                                    // lt
          addi
LT:
FINE:
```



4) Come l'esercizio precedente, ma i registri t1 e t0 contengono due indirizzi.

MAX 7

Soluzione

```
bne $t1, $t0, NE
                              // eq
         addi $s0, $zero, 3
              FINE
             $t1, $t0, LT
NE:
         bltu
              $s0, $zero, 2
         addi
                               // gt
              FINE
             $s0, $zero, 1
                               // lt
LT:
         addi
FINE:
```

5) Indicare il risultato dell'esecuzione dei programmi degli esercizi 3 e 4 nei casi seguenti.

contenuto dei registri	s0 in programma 3	s0 in programma 4
\$t0 = 0x 0000 0005	1	2
\$t1 = 0x FFFF FFF0	1	2
\$t0 = 0x FFFF FFF1	1	1
\$t1 = 0x FFFF FFF0	1	1

6) Come l'esercizio 4, ma NO PSEUDO.

Soluzione

```
$t1, $t0, NE
         bne
         addi $s0, $zero, 3
                                  // eq
               FINE
         j
NE:
         sltu $t3, $t1,
                          $t0
               $t3, $zero, LT
$s0, $zero, 2
         bne
         addi
                                // gt
               FINE
         j
         addi $s0, $zero, 1 // lt
LT:
FINE:
```

7) **Scrivere** una sequenza di istruzioni che assegna al registro *s0* il valore della dimensione dell'intervallo tra due etichette di programma L1 e L2; è noto che L2 > L1.

### Soluzione

```
.data
L1: ...
L2: ...
.text
la $t0, L1
la $t1, L2
subu $s0, $t1, $t0
```

8) Siano L1 e L2 due etichette di un programma. **Scrivere** una sequenza di istruzioni che assegna al registro *s0* il valore 1 se L1 > L2, il valore 0 in caso contrario.

### Soluzione

```
la $t0, L1
la $t1, L2
sltu $s0, $t1, $t0
```

9) Dato il programma seguente, **indicare** in esadecimale il valore dei registri *s1-s7* e *t2-t4* dopo l'esecuzione. Si ricorda che la rappresentazione interna è Little Endian.

```
.data
STRINGA:
         .ascii
                  "abcd"
          .word
BYTE:
                  0x 8081 8283
          .text
          la
               $t1, STRINGA
               $t2, BYTE
          la
               $s1, 0($t2)
          1b
               $s2, 0($t2)
          lh
               $s3, 0($t2)
          lw
          1bu $s4, 0($t2)
          1bu $s5, 1($t2)
          1bu
              $s6, 2($t2)
          1bu $s7, 3($t2)
               $t2, 0($t1)
          lb
               $t3, 1($t1)
          1b
          lh
               $t4, 0($t1)
```

### Soluzione

```
s1 = 0x \text{ FFFF FF83} s2 = 0x \text{ FFFF 8283} s3 = 0x 8081 8283 s4 = 0x 0000 0083 s5 = 0x 0000 0082 s6 = 0x 0000 0081 s7 = 0x 0000 0080 t2 = 0x 0000 0061 (a) t3 = 0x 0000 0062 (b) t4 = 0x 0000 6261 (ba)
```

