Calcolatori Elettronici
Esercitazione 2

M. Sonza Reorda – M. Monetti
M. Rebaudengo – R. Ferrero
L. Sterpone – M. Grosso

Politecnico di Torino
Dipartimento di Automatica e Informatica

Obiettivi

- Istruzioni di salto
- Istruzioni logiche

Esercizio 1

- Si scriva un programma che richieda all'utente un intero positivo e quindi dica se il valore introdotto è pari oppure dispari
- Per la determinazione del pari/dispari si utilizzi un'operazione di AND logico con il valore 1

Soluzione

```
          .data
paris:    .asciiz "pari"
disparis: .asciiz "dispari"

          .text
          .globl main
          .ent main
main:
          li $v0, 5
          syscall
          andi $t0, $v0, 1
          la $a0, paris
          beq $t0, $zero, salto
          la $a0, disparis
salto:    li $v0, 4
          syscall
          li $v0, 10
          syscall
.end main
```

Esercizio 2

- Si scriva un programma che
  - Acquisisca due interi positivi
  - Verifichi che gli interi acquisiti siano rappresentabili su byte, e in questo caso esegua la seguente operazione logica *bitwise* e scriva sulla console il risultato ottenuto (intero):

    `C = NOT(A AND (NOT(B))) OR (A XOR B)`
  - Altrimenti, dia un messaggio di errore.

Soluzione

```
          .data
err_mess: .asciiz "Introdurre valori compresi tra -127 e 128"

          .text
          .globl main
          .ent main
main:
          li $v0, 5
          syscall
          li $t1, 0xFFFFFF00
          and $t0, $v0, $t1     # A in $t0
          bne $t0, 0, errore

          li $v0, 5
          syscall
          and $t1, $v0, $t1     # B in $t1
          bne $t1, 0, errore
```

Soluzione [cont.]

```
            not $t3, $t1          # not B
            and $t3, $t0, $t3     # A and (not B)
            not $t3, $t3          # not (A and (not B))
            xor $t0, $t0, $t1     # A xor B
            or $t0, $t0, $t3      # not (A and (not B)) or (A xor B)

            li $t1, 0x000000FF
            and $t0, $t0, $t1
            move $a0, $t0
            li $v0, 1
            syscall

            j fine

errore:     la, $a0, err_mess
            li $v0, 4
            syscall

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

Esercizio 3

- Date tre variabili *word* inizializzate in memoria, si scriva un programma che le stampi a video in ordine crescente
  - È possibile usare l'algoritmo descritto con il seguente pseudocodice:

```
if (a > b)
    swap(a, b);
if (a > c)
    swap(a, c);
if (b > c)
    swap(b, c);
```

Soluzione

```
 .data
v0:       .word 1249
v1:       .word 2198
v2:       .word -968

          .text
          .globl main
          .ent main
main:     lw $t0, v0
          lw $t1, v1
          lw $t2, v2

          blt $t0, $t1, salto1
          move $t3, $t0
          move $t0, $t1
          move $t1, $t3
salto1:   blt $t0, $t2, salto2
          move $t3, $t0
          move $t0, $t2
          move $t2, $t3
salto2:   blt $t1, $t2, salto3
          move $t3, $t1
          move $t1, $t2
          move $t2, $t3
salto3:   move $a0, $t0
          li $v0, 1
          syscall
          li $a0, '\n'
          li $v0, 11
          syscall
          move $a0, $t1
          li $v0, 1
          syscall
          li $a0, '\n'
          li $v0, 11
          syscall
          move $a0, $t2
          li $v0, 1
          syscall
          li $a0, '\n'
          li $v0, 11
          syscall

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

Esercizio 4

- Si scriva un programma che conti il numero di bit a 1 nella rappresentazione binaria di una variabile di tipo *halfword*.

Soluzione

```
          .data
num:      .half 1979

          .text
          .globl main
          .ent main
main:     and $t3, $0, $0    # azzeramento risultato
          and $t4, $0, $0    # azzeramento indice
          lh $t0, num
          li $t1, 1
ciclo:    and $t2, $t0, $t1
          beq $t2, 0, next
          addi $t3, $t3, 1
next:     sll $t1, $t1, 1
          addi $t4, $t4, 1
          bne $t4, 16, ciclo

          move $a0, $t3      # stampa risultato
          li $v0, 1
          syscall
          li $v0, 10
          syscall
          .end main
```