Calcolatori Elettronici
Esercitazione 6

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

Politecnico di Torino
Dipartimento di Automatica e Informatica

Obiettivi

- Chiamata a procedura
- Passaggio di un parametro tramite registro
- Passaggio del valore di ritorno tramite registro

Esercizio 1

- Si scriva una procedura `stampaTriangolo` che mostra a video un triangolo rettangolo isoscele di lato 8, tramite una opportuna sequenza di asterischi.
- Si scriva una procedura `stampaQuadrato` che mostra a video un quadrato di lato 8, tramite una opportuna sequenza di asterischi.
- A destra è mostrato l'output ottenuto richiamando le due procedure dal main.

```
*
**
***
****
*****
******
*******
********
********
********
********
********
********
********
********
********
```

Soluzione

```
            .data
            .text
            .globl main
            .ent main
main:       jal stampaTriangolo
            jal stampaQuadrato
            li $v0, 10
            syscall
            .end main
```

Soluzione [cont.]

```
                .ent stampaTriangolo
stampaTriangolo: li $t0, 1        #numero iniziale asterischi
                li $v0, 11
cicloRigheTriangolo:
                li $a0, '*'
                li $t1, 0
cicloColonneTriangolo:
                syscall
                addi $t1, $t1, 1
                bne $t1, $t0, cicloColonneTriangolo
                li $a0, '\n'
                syscall
                addi $t0, $t0, 1
                bne $t0, 9, cicloRigheTriangolo
                jr $ra
                .end stampaTriangolo
```

Soluzione [cont.]

```
                .ent stampaQuadrato
stampaQuadrato: li $t0, 0       #indice riga
                li $v0, 11
cicloRigheQuadrato:
                li $a0, '*'
                li $t1, 0       #indice colonna
cicloColonneQuadrato:
                syscall
                addi $t1, $t1, 1
                bne $t1, 8, cicloColonneQuadrato
                li $a0, '\n'
                syscall
                addi $t0, $t0, 1
                bne $t0, 8, cicloRigheQuadrato
                jr $ra
                .end stampaQuadrato
```

Esercizio 2

- Si modifichino le due procedure implementate nell'esercizio precedente, in modo che ricevano come parametro la dimensione del lato del triangolo e del quadrato.
- Il parametro è passato attraverso il registro $a0.
- All'inizio del main, chiedere all'utente la dimensione del lato.

Soluzione

```
            .data
input:      .asciiz "Introduci un numero: "
            .text
            .globl main
            .ent main
main:       la $a0, input
            li $v0, 4
            syscall
            li $v0, 5
            syscall
            move $s0, $v0
            move $a0, $s0
            jal stampaTriangolo
            move $a0, $s0
            jal stampaQuadrato
            li $v0, 10
            syscall
            .end main
```

Soluzione [cont.]

```
                .ent stampaTriangolo
stampaTriangolo: add $t2, $a0, 1
                li $t0, 1       #numero iniziale asterischi
                li $v0, 11
cicloRigheTriangolo:
                li $a0, '*'
                li $t1, 0
cicloColonneTriangolo:
                syscall
                addi $t1, $t1, 1
                bne $t1, $t0, cicloColonneTriangolo
                li $a0, '\n'
                syscall
                addi $t0, $t0, 1
                bne $t0, $t2, cicloRigheTriangolo
                jr $ra
                .end stampaTriangolo
```

Soluzione [cont.]

```
                .ent stampaQuadrato
stampaQuadrato: move $t2, $a0
                li $t0, 0       #indice riga
                li $v0, 11
cicloRigheQuadrato:
                li $a0, '*'
                li $t1, 0       #indice colonna
cicloColonneQuadrato:
                syscall
                addi $t1, $t1, 1
                bne $t1, $t2, cicloColonneQuadrato
                li $a0, '\n'
                syscall
                addi $t0, $t0, 1
                bne $t0, $t2, cicloRigheQuadrato
                jr $ra
                .end stampaQuadrato
```

Esercizio 3

- Si scriva un programma per la conversione di una parola di caratteri minuscoli in caratteri maiuscoli, attraverso un'opportuna procedura.
- Si passi alla procedura il codice ASCII di un carattere alla volta come parametro *by value* utilizzando il registro $a0; il carattere convertito è restituito attraverso $v0.

Soluzione

```
            .data
stringa:    .asciiz "parola"
            .text
            .globl main
            .ent main
main:       li $s0, 0
ciclo:      lbu $a0, stringa($s0)
            beq $a0, 0, fine
            jal converti
            sb $v0, stringa($s0)
            addi $s0, $s0, 1
            b ciclo
            li $v0, 10
            syscall
            .end main
```

Soluzione [cont.]

```
            .ent converti
converti:   addi $a0, $a0, 'A'
            li $v0, 'a'
            sub $v0, $a0, $v0
            jr $ra
            .end converti
```

Esercizio 4

- Si scriva una procedura `massimo` in grado di calcolare il valore massimo di un vettore di interi *word*.
- La procedura riceve l'indirizzo del vettore in `$a0` e la sua lunghezza in `$a1`, e salva il risultato in `$v0`.
- Al termine della procedura, il *main* deve stampare a video il valore del massimo trovato.

Soluzione

```
DIM = 7
            .data
vettore:    .word 15, 870, 1200, -21, -1000, 15003, -1039581
            .text
            .globl main
            .ent main
main:       la $a0, vettore
            li $a1, DIM
            jal massimo
            move $a0, $v0
            li $v0, 1
            syscall
            li $v0, 10
            syscall
            .end main
```

Soluzione [cont.]

```
            .ent massimo
massimo:    move $t0, $a0
            move $t1, $a1     # per ipotesi $a1>0
            lw $v0, ($t0)

ciclo:      add $t0, $t0, 4
            sub $t1, $t1, 1
            beqz $t1, fine
            lw $t2, ($t0)
            blt $t2, $v0, next
            move $v0, $t2
next:       j ciclo

fine:       jr $ra
            .end massimo
```

Esercizio 5

- Nel calcolo combinatorio si definisce *combinazione semplice* (*senza ripetizioni*) una presentazione di elementi di un insieme nella quale non ha importanza l'ordine dei componenti e non si può ripetere lo stesso elemento più volte. Dati *n* elementi distinti e un numero intero positivo $k \leq n$, il numero di combinazioni semplici possibili $C(n, k)$ è dato dalla seguente formula:

$$C(n,k) = \binom{n}{k} = \frac{n \cdot (n-1) \cdot (n-2) \ldots \cdot (n-k+1)}{k!}$$

- Si scriva una procedura `combina` in grado di calcolare il numero di combinazioni semplici dati i parametri n e k ricevuti rispettivamente tramite $a0 e $a1. Il risultato dovrà essere restituito attraverso il registro $v0.
- Sia lecito supporre che durante le operazioni intermedie non si presenti *overflow*.
- Esempi:
  - `n = 6;  k = 3` `C(n, k) = 20`
  - `n = 12; k = 2` `C(n, k) = 66`

Soluzione

```
            .data
stringa:    .asciiz "introdurre "
            .text
            .globl main
            .ent main
main:       li $v0, 4     # print string
            la $a0, stringa
            syscall
            li $v0, 11    # print char
            li $a0, 'n'
            syscall
            li $a0, ':'
            syscall
            li $v0, 5     # read integer
            syscall
            move $t0, $v0
            li $v0, 4     # print string
            la $a0, stringa
            syscall
            li $v0, 11    # print char
            li $a0, 'k'
            syscall
            li $a0, ':'
            syscall
            li $v0, 5     # read integer
            syscall
            move $a0, $t0
            move $a1, $v0
            jal combina
            move $t0, $v0
            li $v0, 11    # print char
            li $a0, 'C'
            syscall
            li $a0, '='
            syscall
            move $a0, $t0
            li $v0, 1
            syscall
            li $v0, 10
            syscall
           .end main
```

Soluzione [cont.]

```
                .ent combina
combina:        subu $t1, $a0, $a1
                addu $t1, $t1, 1
                move $v0, $a0
ciclo1:         beq $a0, $t1, fine1
                subu $a0, $a0, 1
                mul $v0, $v0, $a0
                j ciclo1

fine1:          divu $v0, $v0, $a1
ciclo2:         bltu $a1, 2, fine2      # evito divisione per 1
                sub $a1, $a1, 1
                divu $v0, $v0, $a1
                j ciclo2

fine2:          jr $ra
                .end combina
```