esercitazione 3 - a

stesura e commento by R. Galletti - © www.riccardogalletti.com/appunti_gratis
- testo esercitazione prelevato dal sito del prof. Tortorella http://webuser.unicas.it/tortorella/CalcEl1/Labs/lab3.htm
Scrivere un programma in Assembly MIPS che calcoli il Massimo Comune Denominatore tra due numeri interi letti da input e salvati in due spazi opportunamente allocati nel segmento .data. Nello stesso modo si salvi il valore calcolato prima di stamparlo in output.

Si consideri a titolo di riferimento il seguente codice C, che implementa l'algoritmo di Euclide:

```
  unsigned int a,b,mcd;
void main()

{  unsigned int x,y,r;
   cout << "Primo valore: ";
  cin >> a;
  cout << "Secondo valore: ";
  cin >> b;
   if (a>=b)
  { x = a;
    y = b;  }

  else
  {  x = b;
    y = a;  }

  r = x%y;
  while (r!=0)
  {   x = y;
     y = r;
     r = x%y;  }

  mcd = y;
  cout << "Il MCD tra " << a << " e " << b;
  cout << "e' " << mcd << "\n";
}
```

SVOLGIMENTO

```
    .data
a: .space 4 #riserva in memoria lo spazio per 1 word (senza inizializzarla, cioè darci un valore)
bb: .space 4  #non si può usare il nome b per un etichetta (si confonderebbe con branch)
mcd: .space 4

 val1: .asciiz "Primo valore: "
 val2: .asciiz "Secondo valore: "
str_fin1: .asciiz "Il MCD tra "
str_fin2: .asciiz " e "
str_fin3: .asciiz " é:\n"
str_cop: .asciiz "\n\nCreated by R. Galletti"

    .text
    .globl main

  main:
addiu $t0, $0, 0 # x
addiu $t1, $0, 0 # y
addiu $t2, $0, 0 # r

    li $v0, 4
la $a0, val1
   syscall #stampa la stringa val1


li $v0, 5
syscall # legge a e lo metto in $v0
sw $v0, a  # salvo a

    li $v0, 4
la $a0, val2
   syscall #stampa la stringa val2

li $v0, 5
syscall # legge b e lo metto in $v0
sw $v0, bb    #salvo b

lw $s0, a #carico il valore di a
lw $s1, bb #carico il valore di b
```

```
slt $s7, $s0,  $s1 #se b>a --> $s7=1
bne $s7, $0, else  #vai ad else se b>a
lw $t0, a #x=a
lw $t1, bb #y=b
j annanz # vai annanz, senza andare ad else

else:
lw $t0, bb #x=b
lw $t1, a #y=a

annanz:
div $t0, $t1
mfhi $t2 # il resto di x/y lo mette in $t2

while: beq $t2, $0, annanz2 # se r=0,  vai annanz2
move $t0, $t1 #x=y
move $t1, $t2  #y=r
div $t0, $t1
mfhi $t2
j while

annanz2:
sw $t1, mcd

 li $v0, 4
la $a0, str_fin1
   syscall #stampa la stringa str_fin1

li $v0, 1
lw $a0, a
 syscall #visualizza a

 li $v0, 4
la $a0, str_fin2
   syscall #stampa la stringa str_fin2

li $v0, 1
lw $a0, bb
 syscall #visualizza b

 li $v0, 4
la $a0, str_fin3
   syscall #stampa la stringa str_fin3

li $v0, 1
lw $a0, mcd
 syscall #visualizza il MCD

li $v0, 4
la $a0, str_cop
 syscall

li $v0,10 # serve per uscire
      syscall
```

esercitazione 3 - b

stesura e commento by R. Galletti - © galloimmenso.com
- testo esercitazione prelevato dal sito del prof. Tortorella http://webuser.unicas.it/tortorella/CalcEl1/Labs/lab3.htm

Scrivere un programma in Assembly MIPS che calcoli la somma degli elementi di un array di interi letti da input e salvati in uno spazio opportunamente allocato nel segmento .data. Nello stesso modo si salvi il valore calcolato prima di stamparlo in output.

Si consideri a titolo di riferimento il seguente codice C:

```
unsigned int vet[128];
unsigned int riemp;
unsigned int somma;


void main()

{  unsigned int x,i,n,s;

   cout << "Numero elementi: ";
  cin >> n;
  riemp=n;

  for(i=0;i<n;i++)

  {  cout << "Elemento " << i <<": ";

    cin >> x;

    vet[i]=x;  }

  s=0;

  for(i=0;i<n;i++)

  {   x=vet[i];

    s=s+x;  }

  somma = s;

  cout << "La somma degli " << n << " elementi e' " << somma << "\n";}
```

SVOLGIMENTO

```
   .data
vet: .space 512 #alloca 512 byte (128 word) consecutive
riemp: .space 4
somma: .space 4

 str_n: .asciiz "Numero di elementi: \n"
 str_elem: .asciiz "Elemento "
 str_accapo: .asciiz "\n"
str_fin1: .asciiz "La somma dei "
str_fin2: .asciiz " elementi è\n"
str_cop: .asciiz " \n\n Created by R. Galletti"

    .text
    .globl main

  main:
addiu $t0, $0, 0 # x
addiu $t1, $0, 0 # i
addiu $t2, $0, 0 # n
addiu $t3, $0, 0 # s

    li $v0, 4
la $a0, str_n
   syscall #stampa la stringa str_n

li $v0, 5
syscall # legge n e lo metto in $v0
move $t2, $v0  # sposto il valore letto nel registro n
sw $t2, riemp # riemp=n

for:
 ble $t2, $t1, annanz # vai annanz se n<= i

   li $v0, 4
```

```
la $a0, str_elem
   syscall #stampa la stringa str_elem

li $v0, 1
move $a0, $t1
 syscall #visualizza il numero i-esimo

   li $v0, 4
la $a0, str_accapo
   syscall #va a capo

li $v0, 5
syscall # legge l'elemento i-esimo e lo metto in $v0
move $t0, $v0  # sposto il valore letto nel registro x

move $s0, $t1
mul $s0, $s0, 4 #calcola l'indirizzo (cioè 4i, ma senza modificare i)
sw $t0, vet ($s0) #vet(i)=x

addiu $t1, $t1, 1 # i++
j for

annanz: move $t1, $0 # riazzeramento di i

for2:
 ble $t2, $t1, annanz2 # vai annanz2 se n<= i

move $s0, $t1
mul $s0, $s0, 4 #calcola l'indirizzo (cioè 4i, ma senza modificare i)
lw $t0, vet ($s0) # x=vet(i)

add $t3, $t3, $t0 #s=s+x

addiu $t1, $t1, 1 # i++
j for2

annanz2:
sw $t3, somma  # somma=s

 li $v0, 4
la $a0, str_fin1
   syscall #stampa la stringa str_fin1

li $v0, 1
lw $a0, riemp
 syscall #visualizza riemp

 li $v0, 4
la $a0, str_fin2
   syscall #stampa la stringa str_fin2

li $v0, 1
lw $a0, somma
 syscall #visualizza somma


li $v0, 4
la $a0, str_cop
 syscall

li $v0,10 # serve per uscire
      syscall
```