Complément Serie TP 02 avec Solution Module AO

Univ Guelma

Promo 2017-2018

Table des matières

1 TRI trois entiers par ordre croissant

Exercice n°1 :

Ecrire un programme en assembleur Mips, intitulé **TRI**, qui fait lire trois entiers A, B et C, les permute de façon à les classer par ordre croissant puis affiche le résultat.

1.1 Solution Exo 01

```
.data
val_A:.asciiz "Entrer val A: "
val_B:.asciiz "Entrer val B: "
val_C:.asciiz "Entrer val C: "
res:.asciiz "TRI de A,B,C: "
space: .asciiz " "
.text

main:

li $v0,4
la $a0,val_A   #Affichage Entrer val A
syscall

la $v0,5
syscall
move $t1,$v0   #Lire A & stocki A sur registre $t1

li $v0,4
la $a0,val_B   #Affichage Entrer val B
syscall

la $v0,5
syscall
move $t2,$v0   #Lire B & stocki B sur registre $t2

li $v0,4
la $a0,val_C   #Affichage Entrer val C
syscall

la $v0,5
syscall
move $t3,$v0  #Lire C & stocki C sur registre $t3

##################
bgt $t1,$t3,lab1
move  $t4,$t3
move  $t3,$t2
move $t2 ,$t4
```

```
40  lab1:
41  bgt $t1,$t2,lab2  ##lab1 & les tests de TRI
42  move $t4,$t2
43  move $t2,$t1
44  move $t1,$t4
45  lab2:
46  bgt $t2,$t3,lab3
47  move $t4,$t3
48  move $t3,$t2
49  move $t2,$t4
50  ##################
51  lab3:
52  li $v0,4
53  la $a0,res  #Affichage TRI de A,B,C
54  syscall
55
56  ##############
57  move $a0,$t3
58  li $v0,1
59  syscall
60          #Affichage le minimum & faire un espace
61  li $v0,4
62  la $a0,space
63  syscall
64  ##############
65
66  ##############
67  move $a0,$t2
68  li $v0,1
69  syscall
70          #Affichage le milieu & faire un espace
71  li $v0,4
72  la $a0,space
73  syscall
74  ##############
75
76  move $a0,$t1
77  li $v0,1  #Affichage le maximum
78  syscall
```

2 Le factoriel d'un entier naturel N donné

Exercice n°2 :

Ecrire un programme en assembleur Mips, faisant calculer et afficher le **factoriel** d'un entier naturel N donné. Sachant que (pour N>0) : N ! = N x (N-1) x (N-2) x ..... 3 x 2 x 1.

2.1 Solution Exo 02

```
1  .data
2  val_n:.asciiz "Entrer valeur de N: "
3  fact:.asciiz "N!= "
4  .text
5
6  main:
7  li $v0,4
8  la $a0,val_n #Affichage "Entrer valeur de N"
9  syscall
10
11 li $v0,5
12 syscall
13 move $t1,$v0  #Lire N & stocki N dans le registe $t1
14
15 li $t2,1     # initialiser $t2=1 (co)
16 li $t3,1     # initialiser $t3=1  (F)
17
18 ##################
19 while:
20 bgt $t2,$t1,labe
21 mul $t3,$t3,$t1   #while (co<=N) {F=F*N; N=N-1;}
22 subi   $t1,$t1,1
23 j while
24 ##################
25 labe:
26 li $v0,4
27 la $a0,fact  #Affiche "N!="
28 syscall
29
30 move $a0,$t3
31 li $v0,1
32 syscall     #Affichage le resultat
```

3 Calculer la multiplication par par additions successives

Exercice n°3 :

Ecrire un programme en assembleur Mips, qui saisit deux entiers X et Y, et fait calculer l'expression S=X*Y par **additions successives** (X*Y=X+X+X+...).

3.1 Solution Exo 03

```
1  .data
2  val_X:.asciiz "X="
3  val_Y:.asciiz "Y="
4  res:.asciiz  "X*Y="
5  .text
6
7  main:
8  li $v0,4
9  la $a0,val_X    #Affichage "X="
10 syscall
11
12 la $v0,5
13 syscall
14 move $t1,$v0    #Lire X & stocki X dans $t1
15
16 li $v0,4
17 la $a0,val_Y   #Affichage "Y="
18 syscall
19
20 la $v0,5
21 syscall
22 move $t2,$v0   #Lire Y & stocki Y dans $t2
23
24 li $t3,1       #compteur (i) $t3=1
25
26 while:
27 bgt $t3,$t2,lab  #stocki "res" dans $t4
28 add $t4,$t4,$t1  #while(i<=Y) {res=res+X; i=i+1;}
29 addi $t3,$t3,1
30 j while
31
32 lab :
33 li $v0,4
34 la $a0,res #Affichage "X*Y="
35 syscall
36
37 move $a0,$t4
38 li $v0,1
39 syscall       #Affichage le resultat "res"
```

4 Calculer le PGCD de deux nombres entiers

Exercice n°4 :

L'algorithme d'Euclide permet de calculer le pgcd de deux nombres entiers, c'est à dire le plus grand entier positif divisant ces deux nombres, par des divisions successives.
Ecrire un programme MIPS qui permet de calculer la PGCD de deux nombre entier A et B, selon l'algorithme d'Euclide.

Principe : si a = b, le pgcd est a.
Sinon on calcule le pgcd du couple formé par la différence entre a et b, et le plus petit des deux.
Donc

$$pgcd(a, b) = pgcd(a - b, b) \text{ si } a > b$$
$$pgcd(a, b) = pgcd(a, b - a) \text{ si } b > a$$
$$pgcd(a, b) = a \text{ si } a = b$$

4.1 Solution Exo 04

```
.data
val_A:.asciiz "Entrer A: "
val_B:.asciiz "Entrer B: "
pgcd:.asciiz "PGCD entre A & B= "
.text

main:
li $v0,4
la $a0,val_A
syscall
li $v0,5
syscall
move $t1,$v0  #Affichage "Entrer A" & lire A & stocki A dans $t1

li $v0,4
la $a0,val_B
syscall
li $v0,5
syscall
move $t2,$v0  #Affichage "Entrer B" & lire B & stocki B dans $t2

bne $t1,0,while
bne  $t2,0,while

li $a0,0
li $v0,1
syscall
j lab7

while: #Boucle de traitement
bgt $t1,$t2,lab1
bgt $t2,$t1,lab2
move $a0,$t1
j lab
lab1:
sub $t1,$t1,$t2
j while
lab2:
  sub $t2,$t2,$t1
j while

lab: #Affichage de PGCD
li $v0,4
la $a0,pgcd
syscall
move $a0,$t1
li $v0,1
syscall
lab7:
```

5 La nature d'un triangle

Exercice n°5 :

Ecrire un programme qui permet de saisir les coordonnées des trois sommets A, B et C d'un triangle puis détermine et affiche la nature du triangle (isocèle, équilatéral, quelconque).

5.1 Solution Exo 05

```
.data
x:.asciiz "Entrer x: "
y:.asciiz "Entrer y: "
isocéle:.asciiz "Triangle est isocéle "
équilatéral:.asciiz "Triangle est équilatéral"
quelconque:.asciiz "Triangle est quelconque"
.text

main:
li $v0,4
la $a0,x
syscall

la $v0,5
syscall
move $t1,$v0  # Entrer X de A & stocki X dans $t1

li $v0,4
la $a0,y
syscall

la $v0,5
syscall
move $t2,$v0 # Entrer Y de A & stocki Y dans $t2

li $v0,4
la $a0,x
syscall

la $v0,5
syscall
move $t3,$v0 # Entrer X de B & stocki X dans $t3

 li $v0,4
 la $a0,y
 syscall

 la $v0,5
 syscall
 move $t4,$v0 # Entrer Y de B & stocki Y dans $t4

 li $v0,4
 la $a0,x
 syscall

 la $v0,5
 syscall
 move $t5,$v0 # Entrer X de C & stocki X dans $t5

 li $v0,4
```

```
la $a0,y
syscall

la $v0,5
syscall
move $t6,$v0 # Entrer Y de C & stocki Y dans $t6

sub $s1,$t1,$t3
sub $s2,$t2,$t4
mul $s1,$s1,$s1
mul $s2,$s2,$s2     # $t7=ba
add $t7,$s1,$s2


sub $s3,$t1,$t5
sub $s4,$t2,$t6
mul $s3,$s3,$s3
mul $s4,$s4,$s4
add $t8,$s3,$s4       # $t8=ca



sub $s6,$t3,$t5
sub $s7,$t4,$t6
mul $s6,$s6,$s6     # $t9=cb
mul $s7,$s7,$s7
add $t9,$s6,$s7




beq $t7,$t8, lab
  beq $t7,$t9,lab1
     beq $t8,$t9,lab1
        li $v0,4
        la $a0,quelconque
        syscall
     j ard

lab:
   beq $t8,$t9,lab2
    li $v0,4
    la $a0,équilatéral
    syscall
lab1:
    li $v0,4
    la $a0,équilatéral
    syscall
j ard
```

```
lab2:
    li $v0,4
    la $a0,isocéle
    syscall
ard:
```