TD5

Exercice1 :

Ecrire un programme en assembleur qui permet de lire au clavier votre nom et prénom et de lire aussi un nombre d'itérations au clavier, ensuite il affiche à l'écran le message Bonjour nom prénom en nombre d'itérations lu au clavier. L'exécution du programme doit ressembler à ça :

```
Entrez votre nom et prenom SVP:henni karim
Donnez le nombre d'itérations SVP:10
Bonjour henni karim
Bonjour henni karim
Bonjour henni karim
Bonjour henni karim
Bonjour henni karim
Bonjour henni karim
Bonjour henni karim
Bonjour henni karim
Bonjour henni karim
Bonjour henni karim
```

Exercice 2 :

Ecrire un programme en assembleur qui lit 2 entiers a et b au clavier, et calcule x^y et il l'affiche à l'écran, l'exécution du programme doit ressembler à ceci :

```
donnez la valeur de x:2
donnez la valeur de y:10
x a la puissance de y est:1024
```

Exercice 3 :

Ecrire un programme en assembleur qui lit deux valeurs entières *a* et *b* entrées au clavier, il calcule la valeur absolue de leur soustraction, il multiplie la valeur de a par 2 au nombre de fois que cette valeur absolue, l'exécution doit ressembler à ça :

```
donnez la valeur de a:10
donnez la valeur de b:5
le résultat est:320
```

```
donnez la valeur de a:5
donnez la valeur de b:9
le résultat est:80
```

Exercice 4 :

Ecrire un programme en assembleur Mips qui permet de lire un entier au clavier et l'ajouter aux entiers précédemment lus, le programme s'arrête quand l'entier lu est "0" et affiche le message "la somme est :" , suivi du résultat.

```
5
6
9
4
14
12
0
la somme est:50
```

TP5

Refaites tous les exercices précédents sur machine et vérifiez leurs exécution.

Correction TD5

Correction exo 1

```
.data
msg1: .asciiz "Entrez votre nom et prenom SVP:"
msg2: .asciiz "Donnez le nombre d'itérations SVP:"
msg3: .asciiz "Bonjour "
entree: .space 30

.text
li $v0, 4
la $a0, msg1
syscall

li $v0, 8
la $a0, entree
li $a1, 30
syscall

li $v0, 4
la $a0, msg2
syscall

li $v0, 5
syscall
move $t1, $v0

addi $t2, $zero, 1
boucle:
bgt $t2, $t1, fin

li $v0, 4
la $a0, msg3
syscall

li $v0, 4
la $a0, entree
syscall

addi $t2, $t2, 1
b boucle

fin:
li $v0,10
syscall
```

Correction exo 2

```
.data
entree1 : .asciiz "donnez la valeur de x:"
entree2 : .asciiz "donnez la valeur de y:"
sortie : .asciiz "x a la puissance de y est:"
.text
```

```
addi $t3, $zero, 0
addi $t4, $zero, 1

li $v0, 4
la $a0, entree1
syscall

li $v0, 5
syscall
move $t1, $v0

li $v0, 4
la $a0, entree2
syscall

li $v0, 5
syscall
move $t2, $v0

calcul:
beq $t3, $t2, fin
mul $t4, $t4, $t1
addi $t3, $t3, 1
b calcul

fin:
li $v0, 4
la $a0, sortie
syscall

li $v0, 1
move $a0, $t4
syscall
```

Correction exo 3

```
.data
 entree1: .asciiz "donnez la valeur de a:"
 entree2 :.asciiz "donnez la valeur de b:"
 sortie: .ascii "le résultat est:"
 zero: .byte 0
 .text
 lb $t4, zero

 li $v0, 4
 la $a0, entree1
 syscall

 li $v0, 5
 syscall
 move $t1, $v0
```

```
li $v0, 4
la $a0, entree2
syscall

li $v0, 5
syscall
move $t2, $v0

bgt $t1, $t2, calcul1
sub $t3, $t2, $t1
b calcul2

calcul1:
sub $t3, $t1, $t2
calcul2:
beq $t4, $t3, fin
mul $t1, $t1, 2
addi $t4, $t4, 1
b calcul2

fin:
li $v0, 4
la $a0, sortie
syscall

li $v0, 1
move $a0, $t1
syscall
```

Correction exo 4

```
.data
sortie:.asciiz "la somme est:"
.text
addi $t2, $zero, 0
boucle:
li $v0, 5
syscall
move $t1, $v0
add $t2, $t2, $t1
beqz $t1, fin
b boucle
fin :
li $v0, 4
la $a0, sortie
syscall
li $v0, 1
move $a0,$t2
syscall
```