Fiche de TP N° 3 Architecture des ordinateurs (AO)

Exercice 1

Revenir sur l'exercice 6 de la fiche de TP2 permettant d'écrire le code MIPS pour calculer les N premiers éléments d'une suite de Fibonacci comme indiqué dans la fenêtre ci-dessous.

1. Modifier le programme déjà fait, en remplaçant la partie du code permettant d'afficher un séparateur entre chaque paire de termes par une procédure « **AfficheSeparateur** » permettant d'afficher le séparateur « **-** »
2. Mettre à jour le programme en remplaçant la procédure « **AfficheSeparateur** » par une procédure « **AfficheChaine** » permettant d'afficher une chaine de caractères qui doit être passée en paramètre avant d'appeler cette procédure.
3. Ajouter, à la dernière version du programme, une procédure « **LireEntier** » permettant de lire un entier et faire appel à cette procédure aux endroits adéquats.
4. Ajouter, à la dernière version du programme, une procédure « **AfficheEntier** » permettant d'afficher un entier et faire appel à cette procédure aux endroits adéquats.

Exercice 2

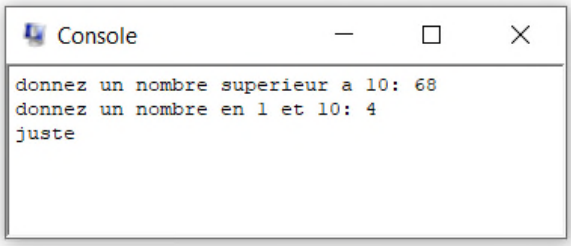
Écrire en assembleur MIPS le code permettant de :

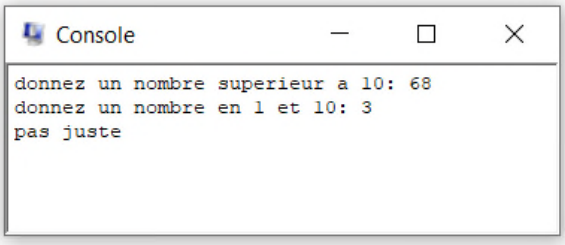
1. Définir la fonction **int impair(int X)** retournant **1** si **X** est **impair** et **0** sinon.
2. Afficher un message à l'utilisateur, lui demandant de donner un nombre entier **X**.
3. Appeler la fonction **impair(X)**
4. Afficher un **1** si **X** est **impair** et **0** si **X** est **pair** en appelant une fonction **print_int**.

Exercice 3

Pour deux nombres saisis au clavier : Le premier supérieur à 10 et le deuxième entre 1 et 10.

Ecrivez un code mips pour appeler la **fonction diviseur** qui vérifie si le nombre 2 est un diviseur du nombre 1 et afficher le message « juste » sinon « pas juste » comme indiqué dans la fenêtre ci-dessous.





Exercice 4

Que fait ce programme ? et que stockent les registres suivants : $t0, $t1, $t2 et $t3

```
.data
tab : .space 500 #allouer 500 bits
msg1 : .asciiz "Donnez la dimonsion de votre tableau : "
msg2 : .asciiz "Inserer S.V.P. une valeur : "
espace : .asciiz " "

.text
main:
la $t0 , tab
addi $t1 , $zero , 0

li $v0 , 4
la $a0 , msg1
syscall
li $v0 , 5
syscall
move $t2,$v0

remplire:
bge $t1 , $t2 , suite
li $v0 , 4
la $a0 , msg2
syscall
li $v0 , 5
syscall
move $t3,$v0
sw $t3 , ($t0)
addi $t0 , $t0 , 4
addi $t1 , $t1 , 1

b remplire

suite:
la $t0 , tab
addi $t1 , $zero , 0

afficher:
bge $t1 , $t2 , fin
lw $t3, ($t0)

li $v0, 1
move $a0, $t3
syscall

li $v0, 4
la $a0, espace
syscall

addi $t0 , $t0 , 4
addi $t1 , $t1 , 1
b afficher

fin:
li $v0,10
syscall
```

Exercice 5

Ajouter au programme précédent le calcul de la somme des éléments du tableau.

Exercice 6

Ecrire un programme qui recherche le plus petit élément dans tableau saisi par l'utilisateur.

Fiche de TP N° 3 Architecture des ordinateurs (AO)
(Solution)

Exercice 1

```
#Solution TP3/Exercice 1 /Question 1
.data
msg1: .asciiz "Donnez S.V.P. un nombre positif : "
msg2: .asciiz "La suite de Fibonacci est comme suit : "
separateur: .asciiz " - "
.text
AfficheSeparateur:
    li $v0 , 4
    la $a0 , separateur
    syscall     # Affichage du Separateur
    jr $ra      # Saut de retour vers l'instruction après jal

main:
li $v0 , 4
la $a0 , msg1
syscall         # Affichage du msg1
li $v0 , 5
syscall
move $s0 , $v0  # lecture de n dans s0, donc n est s0
li $v0 , 4
la $a0 , msg2
syscall         # Affichage du msg2
li $t0 , 1      # t0 est U0, il est initialisé à 1
li $t1 , 1      # t1 est U1, il est initialisé à 1
li $v0 , 1
move $a0 , $t0
syscall         # Affichage de t0
jal AfficheSeparateur
li $v0 , 1
move $a0 , $t1
syscall         # Affichage de t1
jal AfficheSeparateur
Boucle:
add $t2 , $t1 , $t0 # t2 ← t1 + t0 ;
li $v0 , 1
move $a0 , $t2
syscall         # Affichage de t2
jal AfficheSeparateur
move $t0,$t1    # t0 ← t1 ;
move $t1,$t2    # t1 ← t2 ;
sub $s0,$s0,1   # N-- ;
bgt $s0,$0,Boucle # Condition do-while, si c'est vérifié ça boucle au début du bloque Boucle
li $v0 , 10
syscall         # Arrêter le programmee
```

Code source pour les questions 2, 3 et 4

```
#Solution TP3/Exercice 1 /Questions 2, 3 et 4
.data
msg1: .asciiz "Donnez S.V.P. un nombre positif : "
msg2: .asciiz "La suite de Fibonacci est comme suit : "
separateur: .asciiz " - "
.text
AfficheChaine:      # Procédure d'affichage d'une chaine de caractères / code 4
    li $v0 , 4
    syscall         # Affichage de la chaine de caractères
    jr $ra          # Saut de retour vers l'instruction après jal
AfficheEntier:      # Procédure d'affichage d'un entier / code 1
    li $v0 , 1
    syscall         # Affichage de l'entier
    jr $ra          # Saut de retour vers l'instruction après jal
LireEntier:         # Procédure de lecture d'un entier / code 5
    li $v0 , 5
    syscall         # Lecture de l'entier
    jr $ra          # Saut de retour vers l'instruction après jal
main:
la $a0 , msg1       # passage de paramètre msg1 dans $a0
jal AfficheChaine   # Appel de procédure AfficheChaine pour affichage du msg1
jal LireEntier      # Appel de procédure LireEntier pour lire un entier
move $s0 , $v0      # Mettre N dans $s0, donc la valeur de N est déplacé dans $s0
la $a0 , msg2       # Passage de paramètre msg2 dans $a0
jal AfficheChaine   # Appel de procédure AfficheChaine pour affichage du msg2
li $t0 , 1          # t0 est U0, il est initialisé à 1
li $t1 , 1          # t1 est U1, il est initialisé à 1
move $a0 , $t0      # Passage de paramètre $t0 dans $a0
jal AfficheEntier   # Appel de procédure AfficheEntier pour affichage de t0
la $a0,separateur   # Passage de paramètre separateur dans $a0
jal AfficheChaine   # Appel de procédure AfficheChaine pour l'affichage d'un séparateur
move $a0 , $t1      # Passage de paramètre $t1 dans $a0
jal AfficheEntier   # Appel de procédure AfficheEntier pour affichage de t1
la $a0,separateur   # Passage de paramètre separateur dans $a0
jal AfficheChaine   # Appel de procédure AfficheChaine pour l'affichage d'un séparateur
Boucle:
add $t2 , $t1 , $t0 # t2 ← t1 + t0 ;
move $a0 , $t2
jal AfficheEntier   # Appel de procédure AfficheEntier pour affichage de t2
la $a0,separateur   # Passage de paramètre separateur dans $a0
jal AfficheChaine   # Appel de procédure AfficheChaine pour l'affichage d'un séparateur
move $t0 , $t1      # T0 ← T1 ;
move $t1 , $t2      # T1 ← T2 ;
sub $s0 , $s0 , 1   # N-- ;
bgt $s0 , $0 , Boucle   # condition do-while, si c'est vérifié ça boucle au début du bloque Boucle
li $v0 , 10
syscall                 # Arrêter le programmee
```

Exercice 2

```
.data
meg: .asciiz "donnez un nombre X svp: "
.text
impair:
move $t0,$a0
div $t1,$t0,2
mfhi $t2
bnez $t2, then
la $a0, 0
j end
then:
li $a0, 1
```

```
main:
li $v0, 4
la $a0, meg
syscall
li $v0, 5
syscall
move $a0, $v0
jal impair
jal print_int
li $v0, 10
syscall
```

```
end:
jr $ra
print_int:
li $v0, 1
syscall
jr $ra
```

Exercice 3

```
.data
meg1: .asciiz "donnez un nombre superieur a 10: "
meg2: .asciiz "donnez un nombre en 1 et 10: "
meg3: .asciiz "juste "
meg4: .asciiz "pas juste "
.text

diviseur:
move $t0,$a1
div $t1,$t0,$a2
mfhi $t2
bnez $t2, then
la $a0, meg3
j end
then:
la $a0, meg4
end:
jr $ra
```

```
main:
li $v0, 4
la $a0, meg1
syscall
li $v0, 5
syscall
move $a1, $v0
li $v0, 4
la $a0, meg2
syscall
li $v0, 5
syscall
move $a2, $v0
jal diviseur
li $v0, 4
syscall
li $v0, 10
syscall
```

Exercice 4

Le programme permet de lire un tableau saisi au clavier puis l'afficher après à l'écran

$t0 : stock l'adresse du tableau

$t1 : stock la variable compteur

$t2 : stock la constante dimension du tableau

$t3 : stock le contenu d'un élément du tableau

Exercice 5

```
.data
tab : .space 500 #allouer 500 bits
msg1 : .asciiz "Donnez la dimonsion de votre tableau : "
msg2 : .asciiz "Inserer S.V.P. une valeur : "
espace : .asciiz " "
som: .asciiz "\n la somme est: "

.text
main:
la $t0 , tab
addi $t1 , $zero , 0
```

```
suite:
la $t0 , tab
addi $t1 , $zero , 0

afficher:
bge $t1 , $t2 , fin
lw $t3, ($t0)

li $v0, 1
move $a0, $t3
add $t5, $t5, $t3
syscall
```

```
li $v0 , 4
la $a0 , msg1
syscall
li $v0 , 5
syscall
move $t2,$v0

remplire:
bge $t1 , $t2 , suite
li $v0 , 4
la $a0 , msg2
syscall
li $v0 , 5
syscall
move $t3,$v0

sw $t3 , ($t0)
addi $t0 , $t0 , 4
addi $t1 , $t1 , 1

b remplire
```

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

addi $t0 , $t0 , 4
addi $t1 , $t1 , 1
b afficher

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

li $v0, 1
move $a0, $t5
syscall

li $v0,10
syscall
```

Exercice 6

```
.data
tab : .space 500 #allouer 500 bits
msg1 : .asciiz "Donnez la dimonsion de votre tableau : "
msg2 : .asciiz "Inserer S.V.P. une valeur : "
min: .asciiz "min est: "
.text
main:
la $t0 , tab
addi $t1 , $zero , 0

li $v0 , 4
la $a0 , msg1
syscall
li $v0 , 5
syscall
move $t2,$v0

remplire:
bge $t1 , $t2 , suite
li $v0 , 4
la $a0 , msg2
syscall
li $v0 , 5
syscall
move $t3,$v0

sw $t3 , ($t0)
```

```
suite:
la $t0 , tab
li $t1,2
lw $t3, ($t0)

rechercher:
bge $t1 , $t2 , fin
lw $a3, ($t0)
ble $a3, $t3, lab
addi $t0 , $t0 , 4
addi $t1 , $t1 , 1
j rechercher

lab:
move $t3, $a3
addi $t0 , $t0 , 4
addi $t1 , $t1 , 1
j rechercher
syscall

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

move $a0,$t3
li $v0, 1
```

addi $t0 , $t0 , 4 addi $t1 , $t1 , 1 b remplire	syscall li $v0,10 syscall