Université des sciences et de la technologie d'Oran – MB
Faculté des Mathématiques et de l'Informatique - Département d'informatique
2ème Année LMD - S3 - 2021/2022

Corrigé du TP N°3 – Architecture des ordinateurs

Exercice 1 :

1. Code source du programme contenant la procédure « **AfficheSeparateur** »

```
#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 :

```
#Solution TP3/Exercice 2 / Fonction addition
.data
X:  .word  0                # Déclaration de la variable X de type entier dans la mémoire
                            # la solution reste correcte si un registre était considéré comme x
.text
addition:                   # La fonction addition
    move  $t0 , $a0
    move  $t1 , $a1         # Récupération par la fonction des arguments
    add   $t2 , $t0 , $t1   # l'addition
    move  $v0 , $t2         # remplissage de la valeur de retour
    jr    $ra               # saut de retour vers l'instruction après jal
main:
    li    $a0 , 5
    li    $a1 , 15          # remplissage des 2 paramètres de la fonction avant l'appel
    jal   addition          # instruction d'appel de la fonction addition,
                            # $ra reçoit l'adresse de l'instruction suivante la $s0 , x
    la   $s0 , X            # retour de la fonction
    sw   $v0 , 0($s0)       # sauvegarde de la valeur retournée dans X, l'immédiate dans
                            # sw doit être à 0, puisque il n'y a pas de décalage
    li    $v0 , 10
    syscall                 # Fin du programme
```

Exercice 3 :

```
1   .data
2   Message: .asciiz "Donnez un nombre X svp: "
3   .text
4   impair:
5   sub $sp, $sp, 32
6   sw $ra, 20($sp)
7   sw $fp, 16($sp)
8   add $fp, $sp, 32
9   and $v0, $a0, 1
10  lw $ra, 20($sp)
11  lw $fp, 16($sp)
12  add $sp, $sp, 32
13  jr $ra
14  print_int:
15  li $v0,1
16  syscall
17  jr $ra
18  main:
19  li $v0,4
20  la $a0,Message
21  syscall
22  li $v0,5
23  syscall
24  move $a0,$v0
25  #move $a0,$t1
26  jal impair
27  move $a0, $v0
28  jal print_int
29  move $t0,$v0
30  # Fin du programme
31  li $v0, 10
32  syscall
```