Solution of Lab Series No. 4

Exercise No. 1:

```
.data
msg1: .asciiz " \n Entrez un entier SVP :"
msg2: .asciiz " \n Entrez un opérande : "
msg3: .asciiz " \n le résultat de l'opération est : "
msg4: .asciiz " Error, veuillez saisir à nouveau un opérande correct"
.text
# Initialisation
li $v0,4
la $a0, msg1
syscall
li $v0,5
syscall
move $t1, $v0
li $v0,4
la $a0, msg1
syscall
li $v0,5
syscall
move $t2, $v0
sais_car:
li $v0,4
la $a0, msg2
syscall
li $v0,12
syscall
move $t3, $v0
#test
beq $t3,'+',Sp1
beq $t3,'-',Sp2
beq $t3,'*',Sp3
beq $t3,'/',Sp4
#gestion erreur
li $v0,4
la $a0, msg4
syscall
j sais_car
#Appel de fonction
Sp1: jal addition
j affichage
Sp2: jal soustraction
j affichage
Sp3: jal multiplication
j affichage
Sp4: jal division
j affichage
```

```
#affichage du résultat
affichage:
li $v0,4
la $a0, msg3
syscall
move $a0,$a1
li $v0,1
syscall
# exit
li $v0,10
syscall
# partie des fonctions
addition:
add $a1,$t1,$t2
jr $ra
soustraction:
sub $a1,$t1,$t2
jr $ra
multiplication:
mul $a1,$t1,$t2
jr $ra
division:
div $a1,$t1,$t2
jr $ra
```

Exercise No. 2:

```
.data
msg1: .asciiz"Donner un nombre: "
newline: .asciiz"\n"
msg2: .asciiz"\nL'inverse est:"
.text
main:
li $t0,0
li $s0,10
lecture:
beq $t0,$s0,inverse
li $v0,4
la $a0,msg1
syscall
li $v0,5
syscall
subi $sp,$sp,4
sw $v0,0($sp)
addi $t0,$t0,1
j lecture
inverse:
li $v0,4
la $a0,msg2
syscall
affiche:
beqz $t0,exit
lw $t1,0($sp)
```

```
li $v0,4
la $a0,newline
syscall
li $v0,1
move $a0,$t1
syscall
addi $sp,$sp,4
subi $t0,$t0,1
j affiche
exit:
li $v0,10
syscall
```

Exercise No. 3:

```
.data
array:   .word 5, 2, 19, 11, 6  # Tableau d'exemple
size:    .word 5             # Taille du tableau
result:  .asciiz "Le maximum est : "

.text
.globl main

main:
   # Charger l'adresse du tableau et sa taille
   la $a0, array       # $a0 = adresse du tableau
   lw $a1, size        # $a1 = taille du tableau

   # Appeler le sous-programme find_max
   jal find_max


    move $t9, $v0       # $v0 contient le maximum

     # Afficher le résultat
   li $v0, 4
   la $a0, result
   syscall

   move $a0, $t9       # $v0 contient le maximum
   li $v0, 1
   syscall

   # Fin du programme
   li $v0, 10
   syscall

# Sous-programme find_max
find_max:
   # Sauvegarder $ra et $s0 sur la pile
   addiu $sp, $sp, -8
   sw $ra, 4($sp)
   sw $s0, 0($sp)

   # Initialiser le maximum à la première valeur
   lw $t0, 0($a0)      # $t0 = array[0]
   move $s0, $t0       # $s0 = max

   # Parcourir le tableau
   li $t1, 1           # $t1 = index
find_max_loop:
   bge $t1, $a1, find_max_end  # Si index >= taille, sortir de la boucle
   mul $t2, $t1, 4             # Calculer l'adresse de array[index]
   add $t2, $a0, $t2
   lw $t3, 0($t2)              # Charger array[index]
   bgt $t3, $s0, update_max    # Si array[index] > max, mettre à jour

   j next_iteration
```

```
update_max:
   move $s0, $t3            # max = array[index]

next_iteration:
   addiu $t1, $t1, 1        # index++
   j find_max_loop

find_max_end:
   move $v0, $s0            # Retourner max

   # Restaurer $ra et $s0
   lw $ra, 4($sp)
   lw $s0, 0($sp)
   addiu $sp, $sp, 8
   jr $ra
```

Exercise No. 4:

```
.data
prompt:   .asciiz "Entrez un entier positif : "
result:   .asciiz "La factorielle est : "

.text
.globl main

main:
   # Afficher le message d'invite
   li $v0, 4
   la $a0, prompt
   syscall

   # Lire l'entier depuis l'utilisateur
   li $v0, 5
   syscall
   move $a0, $v0   # Stocker l'entier dans $a0 (paramètre pour fact)

   # Appeler le sous-programme fact
   jal fact

   move $t9, $v0
    # Afficher le résultat
   li $v0, 4
   la $a0, result
   syscall

   move $a0, $t9   # Résultat dans $v0
   li $v0, 1
   syscall

   # Fin du programme
   li $v0, 10
   syscall

# Sous-programme fact (calcul récursif de la factorielle)
```

```
fact:
   # Sauvegarder $ra sur la pile
   addiu $sp, $sp, -8
   sw $ra, 4($sp)
   sw $a0, 0($sp)

   # Cas de base : si n == 0 ou n == 1, retourner 1
   li $t0, 1
   ble $a0, $t0, fact_base

   # Calcul récursif : n * fact(n-1)
   addiu $a0, $a0, -1   # $a0 = n - 1
   jal fact             # Appel récursif
   lw $a0, 0($sp)       # Restaurer n
   mul $v0, $a0, $v0    # $v0 = n * fact(n-1)

   j fact_end

fact_base:
   li $v0, 1            # Retourner 1

fact_end:
   # Restaurer $ra et $a0
   lw $ra, 4($sp)
   addiu $sp, $sp, 8
   jr $ra
```