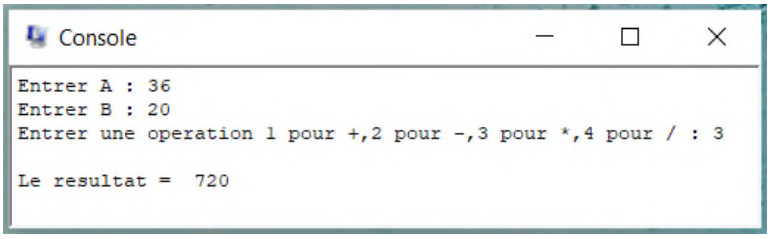
Fiche de TP N° 2 Architecture des ordinateurs (AO)

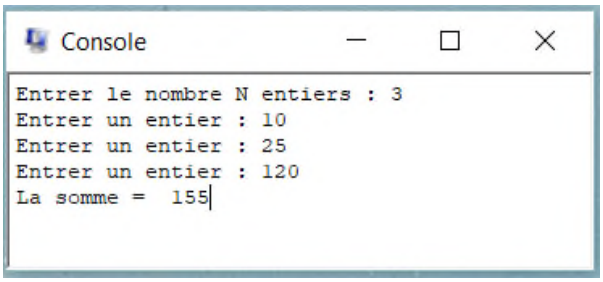
Exercice 1

Ecrire un programme qui permet, à partir de deux valeurs numériques, d'effectuer une addition (+), une soustraction (-), une multiplication (*) ou une division (/) selon le choix de l'utilisateur comme indiqué dans la fenêtre ci-dessous.



Exercice 2

Ecrire un programme MIPS qui calcule la somme de N nombres entiers quelconques comme indiqué dans la fenêtre ci-dessous.



Exercice 3

Ecrire un programme MIPS qui lit trois entiers A, B et C puis affiche la valeur Minimale et la valeur Maximale en sortie.

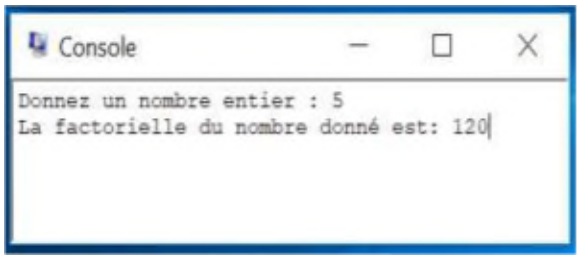
Exercice 4

Ecrire un programme MIPS qui permet de calculer le PGCD de deux nombres A et B, selon l'algorithme d'Euclide :

Principe : si a= b, PGCD (a, b) = a
Sinon si a>b, PGCD (a, b) = PGCD (a-b, b)
Sinon si b>a, PGCD (a, b) = PGCD (a, b-a)

Exercice 5

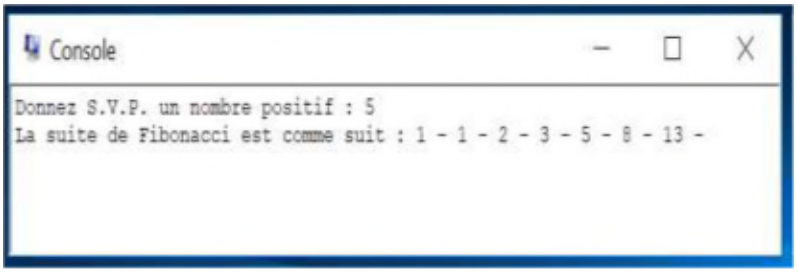
Pour un nombre **n** saisi au clavier, donner le code MIPS permettant de calculer la factorielle de **n** (**n !**) comme indiqué dans la fenêtre ci-dessous.



Exercice 6

Une suite de Fibonacci est une suite définie par ses deux premiers termes $U_0=1$ et $U_1=1$ et son terme général $U_N = U_{N-1}+U_{N-2}$

Ecrire le code MIPS permettant de calculer les **N** premiers éléments d'une suite de Fibonacci tel que **N** est lu au clavier comme indiqué dans la fenêtre ci-dessous.



Fiche de TP N° 2 Architecture des ordinateurs (AO)
(Solution)

Exercice 1

```
.data
msgA: .asciiz "Entrer A : "
msgB: .asciiz "Entrer B : "
msgOP: .asciiz "Entrer une operation 1 pour
+,2 pour -,3 pour *,4 pour / : "
res: .asciiz "\nLe resultat =  "
.text
main:
li $v0,4
la $a0, msgA   #enter A
syscall
li $v0,5
syscall
move $t1,$v0

li $v0,4
la $a0, msgB   #enter B
syscall
li $v0,5
syscall
move $t2,$v0

li $v0,4
la $a0, msgOP#enter operation
syscall
li $v0,5
syscall
move $t4, $v0
```

```
beq $t4,1,labe1
beq $t4,2,labe2
beq $t4,3,labe3
beq $t4,4,labe4

labe1:
add $t3,$t1,$t2
j labe10

labe2:
sub $t3,$t1,$t2
j labe10

labe3:
mul $t3,$t1,$t2
j labe10

labe4:
div $t3,$t1,$t2

labe10:
li $v0, 4
la $a0, res
syscall
move $a0, $t3
li $v0, 1
syscall
li $v0,10
syscall
```

Exercice 2

```
data
val: .asciiz "Entrer le nombre N entiers : "
vall: .asciiz "Entrer un entier : "
res: .asciiz "La somme =  "
.text
main:
li $v0, 4
la $a0, val       #enter N
syscall
li $v0, 5
syscall
move $t1, $v0
li $t2,1# i=1
li $t3,0          # s=0
li $t4,1# 1
while:
bgt $t2,$t1, done
```

```
li $v0, 4
la $a0, vall      # enter un entier
syscall
li $v0, 5
syscall
move $t5, $v0
add $t3,$t3,$t5 #som=som+entier
add $t2,$t2,$t4          #i=i+1
j while
done:
li $v0, 4
la $a0, res
syscall
move $a0, $t3
li $v0, 1
syscall
li $v0,10
syscall
```

Exercice 3

```
data
str: .asciiz "Entrer A : "
str1: .asciiz "Entrer B : "
str2: .asciiz "Entrer C : "
str3: .asciiz "Le Max est : "
str4: .asciiz "\nLe Min est : "
.text
main:
li $v0, 4
la $a0, str
syscall
li $v0, 5
syscall
move $t1, $v0

li $v0, 4
la $a0, str1
syscall
li $v0, 5
syscall
move $t2, $v0

li $v0, 4
la $a0, str2
syscall
li $v0, 5
syscall
move $t3, $v0

bgt $t1,$t2 labe1        #if a>b
move $t4,$t2  # else max=b
j labe2
labe1:
move $t4,$t1  # else max=a
```

```
labe2:
bgt $t3,$t4 labe3
move $t4,$t4
j lab
labe3:
move $t4,$t3
lab:
li $v0, 4
la $a0, str3
syscall
move $a0, $t4
li $v0, 1
syscall

blt $t1,$t2 labe4
move $t5,$t2
j labe5
labe4:
move $t5,$t1
labe5:
blt $t3,$t5 labe6
move $t5,$t5
j labe
labe6:
move $t5, $t3
labe:
li $v0, 4
la $a0, str4
syscall
move $a0, $t5
li $v0, 1
syscall

li $v0,10
syscall
```

Exercice 4

```
.data
strA: .asciiz "Entrer A : "
strB: .asciiz "Entrer B : "
strPGCD: .asciiz "PGCD entre A & B : "

.text
main:
li $v0, 4
la $a0, strA
syscall
li $v0, 5
syscall
move $t1, $v0
```

```

while:
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:
```

li $v0, 4 la $a0, strB syscall li $v0, 5 syscall move $t2, $v0 bne $t1,0,while bne $t2,0,while li $a0,0 li $v0,1 syscall j end	li $v0, 4 la $a0, strPGCD syscall move $a0, $t1 li $v0, 1 syscall end: li $v0,10

Exercice 5

```
1   #Solution TP2/Exercice 4
2   .data
3   Message1: .asciiz "Donnez un nombre entier : "
4   Message2: .asciiz "La factorielle du nombre donné est: "
5   .text
6   main:
7   li $v0,4
8   la $a0,Message1
9   syscall
10  li $v0,5
11  syscall
12  li $t1,1
13  boucle:
14  blt $v0,1, sortie
15  mul $t1,$v0,$t1
16  sub $v0,$v0,1
17  j boucle
18  sortie:
19  li $v0,4
20  la $a0,Message2
21  syscall
22  li $v0,1
23  move $a0,$t1
24  syscall
25  li $v0,10
26  syscall
```

Exercice 6

```
#Solution TP2/Exercice 6
.data
msg1: .asciiz "Donnez S.V.P. un nombre positif : "
msg2: .asciiz "La suite de Fibonacci est comme suit : "
separateur: .asciiz " - "
.text
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 T0, il est initialisé à 1
li $t1 , 1 # t1 est T1, il est initialisé à 1
li $v0 , 1
move $a0 , $t0
syscall # affichage de T0
li $v0 , 4
la $a0 , separateur
syscall # affichage du separateur
li $v0 , 1
move $a0 , $t1
syscall # affichage de T1
 li $v0 , 4
 la $a0 , separateur
 syscall # affichage du separateur
 loop: add $t2 , $t1 , $t0 # T2 ← T1 + T0 ;
 li $v0 , 1
 move $a0 , $t2
 syscall # affichage de T2
 li $v0 , 4
 la $a0 , separateur
 syscall # affichage du separateur
 move $t0 , $t1 # T0 ← T1 ;
 move $t1 , $t2 # T1 ← T2 ;
 sub $s0 , $s0 , 1 # n-- ;
 bgt $s0 , $0 , loop # condition do-while, si c'est true ça boucle au début du bloque
 li $v0 , 10
 syscall # arrêter le programmee
```