is si on initialise i à 10 et on obligés à utiliser bgee.

.text

```
li $to, 10 # i:=10;  $to=10
li $t1, 0  # S:=0;   $t1=0
Beq $to, 0, sorti # si i=0 sortir
sinon
addu $t1, $t1, $to # S:= S+i
subui $to, $to, 1  # i:= i-1
Bgez $to, label # si i>=0 retour à label
sinon :
li $v0, 10
syscall # sortir
```

our afficher le nombre on fait.

```
li $v0, 1
move $a0, $t1
syscall # afficher S

li $v0, 10
syscall # sortir
```

```
li $to, -10 # i:= -10;
li $to, 0   # S:=0;
Bgez $to, label si i>=0 sortir
# sinon
subi $t1, $t1, $to # 0 - (-10) ex
ADD $to, $to, 1 # -10 + 1 = -9 ex
J etiq1 # revenir au test
sorti
```


Exercice - $S = \sum_{i=1}^{9} i!$ (somme des facts de i)

$S = 1! + 2! + 3! + 4! + 5! + 6! + 7! + 8!$

Solution :

```
.data
mess1: .asciiz "La somme des fact est :"
mess2: .asciiz " le fact de "
mess3: .asciiz " est : "

.text
main:
    li $to, 1   # fact:=1
    li $t3, 0   # somme:=0
    li $t2, 1   # i:=1
    move $t2, $t2 # t1:=i
label1: beq $t2, 9, fin # si i=9 alors fin
                        sinon
label: Blez $t1, etiq1 # si t1=0 alors
                       afficher le fact (1)
    mul $to, $to, $t2 # fact:= fact x i
    sub $t1, $t1, 1   # i:= i-1
    J label           # revenir à la boucle

etiq1: li $v0, 4
    la $a0, mess2
    syscall # Afficher le fact de

    li $v0, 1
    move $a0, $t2
    syscall # Afficher i

    li $v0, 4
    la $a0, mess3
    syscall # Afficher: est :
```

```
li $v0, 1
move $a0, $t0      } # Afficher le fact ($t0).
syscall
```

```
fin: li $v0, 4
     la $a0, mes1  } # Afficher: "la somme des facts est :"
     syscall

     li $v0, 1
     move $a0, $t3 } # Afficher la somme ($t3)
     syscall

     li $v0, 10    } # Sortir.
     syscall
```

3) Les tableaux :

Pour déclarer un tableau, on met le nom du tableau suivi de deux points (:) suivi d'un point, le type du tableau et d'une liste de nombres ou char.

nom: .type liste nombres ou caractères

Exemple : T: .byte 0, -1, 5, 2

Exercice :

Ecrire le programme qui fait la somme de tout les éléments du tableau.

Le tableau T contient les éléments 0, -1, 5, 2.

Solution :

la manipulation d'un tableau se fait comme des variables.

n > 255 on utilise par le .byte { half = 2 octet
1 word = 4 octets (i := i + 4)

```
.data
T: .byte 0, -1, 5, 2   # 0 ≤ n ≤ 255
ch: .asciiz "La somme est"

.text
main:
    li $t0, 0    # $t0 = Somme = 0
    li $t1, 0    # $t1 = i = 0

boucle: beq $t1, 4, fin   # si i=4 aller
    lb $t2, T($t1)        # $t2 = elemen
    add $t0, $t0, $t2     # Somme := somm
    add $t1, $t1, 1       # i := i + 1
    j boucle

fin : li $v0, 4
    la $a0, ch    } # Afficher "La somm
    syscall         $a0 contient @ e

    li $v0, 1
    move $a0, $t0 } # Afficher la S
    syscall

    li $v0, 10    } # sortir
    syscall
```

Les avantages du MIPS R3000 est que l'instruction est une 4-instruction.

Comme pour l'instruction la $a0, ch.

on met l'@ du 1er char dans $a0 chaque fois en incrémente à 1, o de l - a - / - S - O ... etc jus fin.