**UNIVERSITE M'Hamed BOUGARA – BOUMERDES**
**FACULTE DES SCIENCES**
**DEPARTEMENT D'INFORMATIQUE**

**Module :** Architecture des Ordinateurs **Responsable :** Mokrani Hocine
**Filière :** Licence Informatique **Semestre :** 3

# Série N°3 : Assembleur MIPS 32 bits

**Exercice 1** *
Quel est le code assembleur MIPS pour le code C ci-dessous ?

**f=(g+h)-(i+j);**

Supposons que les variables **f, g, h, i et j** peuvent être assignées aux registres **$16 à $20** respectivement. En pratique, c'est au compilateur que revient cette tâche délicate d'assignement des variables).

**Exercice 2***

Quel est le code assembleur MIPS pour le code C ci-dessous ?

```
if (i==j)  f=g+h; else f=g-h;
```

Supposons que **f, g, h, i** et **j** correspondent aux registres **$16 à $20** respectivement.

**Exercice 3** *

Quel est le code assembleur MIPS pour le code C ci-dessous ?

```
T[i]=h+T[i] ;
```

Supposons que:

1. **T** est un tableau d'entiers.
2. La variable **h** est dansle registre **$18**.
3. La variable **i** est dans le registre **$19**, et que le tableau débute à l'adresse **Tstart**.

**Exercice 4** **

Traduire le code C suivante en langage assembleur MIPS.

```
While (stock[i]==k) {i=i+j;}
```

Supposons que **i, j et k** correspondent aux registres **$19 à $21**, le tableau stock débute à **Sstart** et le registre **$10** contient la valeur **4**.

**Exercice 5****

Traduire le code C suivante en langage assembleur MIPS.

```
For (i=0; i<1000; i++) {C[i] = A[i] + B[i];}
```

Supposons que les adresses des variables soient :

| | |
|---|---|
| **24000-27996** | **vecteur A** |
| **28000-31996** | **vecteur B** |
| **32000-35996** | **vecteur C** |
| **36000** | **constante 0** |
| **36004** | **constante 3996** |

**Exercice 6 (D-à-D)**

Donner le code pour la procédure suivante

```
change(int v[], int k){
     int temp;
     temp = v[k];
     v[k]=v[k+1];
     v[k+1]=temp;
     }
```

Les paramètres **v et k** sont alloués à **$4 et $5**, une variable **temp** en **$15** et **$2** contient la base du tableau

NB : les adresses de mots contigus différent de 4 et non de 1.

**Rappels**

Pour traduire du C en assembleur :

- Allouer des registres aux variables du programme.
- Produire du code pour le corps de la procédure.
- Préserver les registres à travers l'appel de la procédure

Convention MIPS : pour le passage de paramètres sont utilisés $4 à $7

**Solution Exo1 :**

```
add $8,$17,$18        #Registre $8 contient g+h
add $9,$19,$20        #Registre $9 contient i+j
sub $16,$8,$9         # f reçoit $8-$9, ou (g+h)-(i+j)
```

**Solution Exo2 :**

```
        bne $19,$20,Else        #aller en Else si i≠j
        add $16,$17,$18         #f=g+h (sauté si i≠j)
        j Exit                  #aller en Exit (jump)
Else:   sub $16,$17,$18         #f=g-h (sauté si i=j)
Exit:
```

**Solution Exo3 :**

```
L'expression d'affectation C devient
muli $19,$19,4  #i=i*4
lw $8,Tstart($19)       #reg temporaire $8 reçoit T[i]
add $8,$18,$8           #reg temporaire $8 reçoit h+T[i]
sw $8,Tstart($19)       #on recopie h+T[i]dans T[i]
```

**Solution Exo4 :**

```
loop :  mult $9,$19,$10         #reg temporaire $9=i*4
        lw $8,Sstart($9)        #reg temporaire $8=stock[i]
        bne $8,$21,Exit         #aller en Exit si stock[i]≠k
        add $19,$19,$20         #i=i+j
        j Loop                  #aller en Loop
Exit:
```

**Solution Exo5 :**

```
8000 main :  lw $1,36000($0)        #charger 0 dans r1
8004         lw $2,36004($0)        #charger 3996 dans r2
8008 For :   lw $3,24000($1)        #charger A[i] dans r3
8012         lw $4,28000($1)        #charger B[i] dans r4
8016         add $3,$3,$4           #ajouter [r4] à [r3]
8020         sw $3,32000($1)        #ranger [r3] dans C[i]
8024         beq $1,$2,Exit         #si [r1]=[r2] sauter en 8036
8028         addi $1,$1,4           #incrémenter r1
8032         j For                  # sauter en 8008
8036 Exit :
```

**Solution Exo6 :**

```
Le corps de la procédure modifie les registres $2, $15 et $16

addi $29,$29,-12      #on ajuste la tête de pile
sw $2,0($29)          #range $2 au sommet
sw $15,4($29)         #range $15 au sommet
sw $16,8($29)         #range $16 au sommet
muli $2,$5,4          #reg $2=k*4
add $2,$4,$2          #reg $2=v+(k*4)
                      #reg $2 a l'adresse de v[k]
lw $15,0($2)          #reg $15 (temp)=v[k]
lw $16,4($2)          #reg $16=v[k+1] ; fait référence à
                      #l'élément suivant de v
sw $16,0($2)          #v[k]=registre $16
sw $15,4($2)          #v[k+1]=registre $15 (temp)
lw $2,0($29)          #restitue $2 du sommet
lw $15,4($29)         #restitue $15 du sommet
lw $16,8($29)         #restitue $16 du sommet
addi $29,$29,12       # restitue la tête de pile
jr $31                # retour à la procédure appelante
```