TD 5 : Architecture des ordinateurs
Solutionnaire

Exercice 1

Traduire le fragment de code C suivant en langage d'assemblage MIPS (la variable « quantite » et « prix » correspondent respectivement aux registres $s0 et $s1).

```
switch (quantite) {
        case 20: prix  = 2; break;
        case 50: prix  =3; break;
        case 100: prix = 5; break;
        default: prix = 0;
}
```

Solution :

```
# $s0 = quantite, $s1 = prix
case20:
        addi $t0, $0, 20        # $t0 = 20
        bne $s0, $t0, case50    # quantite ≠ 20, branchement vers case50
        addi $s1, $0, 2         # sinon, prix = 2
        j suite                 # break  et quitter case
case50:
        addi $t0, $0, 50         # $t0 = 50
        bne $s0, $t0, case100    # quantite ≠ 50, branchement vers case100
        addi $s1, $0, 3         # sinon, prix = 3
        j suite                 # break  et quitter case
case100:
        addi $t0, $0, 100       # $t0 = 100
        bne $s0, $t0, default   # quantite ≠ 100, branchement vers default
        addi $s1, $0, 5         # sinon, prix = 5
        j suite                 # break  et quitter case
default:
        add $s1, $0, $0         # prix = 0
suite:
```

Exercice 2

Soit la fonction d'un programme en C suivante :

```
int leaf_example (int g, int h, int i, int j) {

int f;

f = (g + h) – (i + j);

return f; }
```

Quel est le code MIPS correspondant à ce programme ? Sachant que les variables g, h, i, et j correspondent respectivement aux registres $a0, $a1, $a2, et $a3 et f correspond au registre $s0. Le programme compilé débute par l'étiquette de la procédure : *leaf_example.*

Solution :

```
leaf_example :
        addi $sp,$sp, -12
        sw $ra, 12($sp)
        sw $t1, 8($sp)
```

TD 5 : Architecture des ordinateurs
Solutionnaire

```
sw $t0, 4($sp)
sw $s0, 0($sp)
add $t0, $a0, $a1
add $t1, $a2,$a3
sub $s0, $t0, $t1

move $v0, $s0       # retour de la valeur de f , f est copiée dans le registre de retour $v0

# Avant le retour à l'appelant on effectue une restauration des anciennes valeurs pour l'appelant
lw $s0, 0($sp)
lw $t0, 4($sp)
lw $t1, 8($sp)
lw $ra, 12($sp)
addi $sp, $sp, 12   # ajustement de la pile
jr $ra
```

Exercice 3

Donner le code MIPS de la fonction « *strcpy* » utilisée en langage C afin de copier la chaine de caractère « *y* » *dans « x »* sachant que la fin d'une chaine de caractères est caractérisée par l'octet « \0 ». On suppose que les index des tableaux *« x »* et *« y »* se trouvent dans les registres $a0 et $a1 et la variable *i* dans *$s0.*

```
void strcpy(char x[ ], char y[ ])
{
        int i ;
        i = 0 ;
        while ( (x[ i ] = y[ i ]) != '\0')  /* copier et test de la fin de la chaine*/
        i += 1 ;

}
```

Solution :

```
strcpy:
        addi $sp,$sp, -4        # adjuster la pile pour un seul argument
        sw   $s0, 0($sp)        # rangement de $s0
        add $s0, $zero, $zero # i = i + 0
L1:
        add $t1, $s0, $a1       # rangement de l'adresse de y[i] dans $t1
        lbu  $t2, 0($t1)        # $t2 = y[i]
        add $t3, $s0, $a0       # $t3 = x[i]
        sb    $t2, 0($t3)       # x[i] = y[i]
        beq  $t2, $zero, L2     # if y[i] == 0, branchement vers L2
        addi $s0, $s0, 1        # i = i + 1
        j      L1               # saut vers L1
L2 :
        lw     $s0, 0($sp)      # y[i] ==0, la fin de la chaine, chargement de $s0
        addi  $sp,$sp, 4        # ajustement de la pile à l'état initial
        jr      $ra             # retour
```