Architecture des Ordinateurs, corrigé TP 2

Recommandations

— Etudier préalablement, et utiliser le MémoMIPS proposé en ligne.
— Commenter soigneusement *chaque* ligne écrite en assembleur...

Exercice 1. A partir des possibilités d'appels système sur les entrées–sorties, écrire un programme qui lit répétitivement un entier au clavier et l'ajoute aux entiers précédemment lus. Le programme s' arrête quand l'entier lu est 0, et affiche alors le message `"Somme = "`, suivi du résultat.

Correction.

```
          .data
str1:     .asciiz "Entrez une suite d'entiers, terminez par 0 : \n"
str2:     .asciiz "Somme = "

          .text
main:     ori $v0, $zero, 4        # v0 <- 4
          la $a0, str1             # a0 <- str1
          syscall                  # "Entrez une suite d'entiers, terminez par 0 : \n"
          or $t0, $zero, $zero     # t0 <- 0
lecture:  ori $v0, $zero, 5        # v0 <- 5
          syscall                  # lecture
          add $t0, $v0, $t0        # t0 <- v0 + t0
          bne $v0, $zero, lecture  # if (vo != 0) goto lecture
          ori $v0, $zero, 4        # v0 <- 4
          la $a0, str2             # a0 <- str2
          syscall                  # "Somme = "
          or $a0, $zero, $t0       # a0 <- t0
          ori $v0, $zero, 1        # v0 <- 1
          syscall                  # affichage de la somme
          ori $v0, $zero, 10       # v0 <- 10
          syscall                  # return 0
```

Exercice 2. Le programme C++ suivant permet d'afficher le contenu d'un tableau `T` :

```
#include <iostream>
#include <iomanip>

using namespace std;

int main (void)
{
  const unsigned N=5;
  unsigned T[]={1, 2, 3, 4, 5};

  cout << "T :\n";
  for (unsigned i = 0 ; (i < N) ; ++i)
    cout << setw(3) << T[i];

  return 0;
} // main()
```

En respectant la structure et la sémantique de ce programme, le traduire en assembleur MIPS en optimisant au mieux les échanges entre mémoire et registres, et en commentant chaque instruction en assembleur.

Correction.

```
                .data
N:              .word 5                 # unsigned N
T:              .word 1, 2, 3, 4, 5     # unsigned T[]
str:            .asciiz "T : \n"

                .text
main:           ori $v0, $zero, 4       # affichage d'une chaine
                la $a0, str             # $a0 <- @chaine
                syscall                 # "T : \n"

                la $t0, N               # $t0 <- @N
                lw $s0, 0($t0)          # $s0 <- N

                la $s1, T               # T:$s1 <- @T

                or $t0, $zero, $zero    # i:$t0 <- 0
for:            beq $t0, $s0, ExitFor   # if (i==N) goto exitfor
                sll $t1, $t0, 2         # $t1 <- i:$t0 * 4
                add $t1, $s1, $t1       # $t1 <- @T + deplacement physique
                lw $a0, 0($t1)          # $a0 <- T[i]
                ori $v0, $zero, 1       # affichage d'un entier
                syscall                 # affiche le contenu de $a0
                addi $t0, $t0, 1        # ++i
                j for                   # retour au debut de boucle

exitfor:        or $v0, $zero, 10       # return 0
                syscall
```

Exercice 3. On considère les registres `$a0`, `$a1` ; `$a2` associés respectivement à l'adresse de début d'un tableau `save`, au paramètre `k`, et au paramètre `n` passés à la fonction `Recherche` du programme C/C++ suivant :

```
#include <iostream>

using namespace std;

int Recherche (int save[], int k, int rang)
{   unsigned i;

    for(i=0; i <= rang && save[i] != k; ++i) {};
    return i;
} // Recherche()

int main (void)
{
    int rangmax=9;
    int save [] = {0, 0, 0, 0, 0, 0, 0, 1, 2, 3};
    cout << Recherche (save, 1, rangmax);
    return 0;
} // main()
```

En associant par ailleurs le registre `$s0` à la variable locale `i` de la fonction, traduire ce programme aussi fidèlement que possible en assembleur MIPS (en commentant chaque instruction).

Correction.

```
            .data
            .word
save:       0x000000000,              # int save[];
            0x000000000,
            0x000000000,
            0x000000000,
            0x000000000,
            0x000000000,
            0x000000000,
            0x000000001,
            0x000000002,
            0x000000003               # save[9]

            .text
main:       la $a0, save              # $a0 <- @save
            ori $a1, $zero, 1         # $a1 = 1
            ori $a2, $zero, 9         # $a2 = 9;
            jal Recherche             # appel de Recherche()

            or $a0, $zero, $v0        # $a0 <- entier i a afficher
            ori $v0, $zero, 1         # affichage
            syscall

            ori $v0, $zero, 10        # $v0 <- 10
            syscall                   # return 0

Recherche:  or $s0, $zero, $zero      # $s0:i <- 0
for:        slt $t0, $a2, $s0         # (n<i)?$t0=1:$t0=0;
            bne $t0, $zero, exitfor   # if (i>n) goto exitfor
            sll $t0, $s0, 2           # $t0 <- 4 * i
            add $t0, $t0, $a0         # $t0 <- @save[i]
            lw $t0, 0($t0)            # $t0 <- save[i]
            beq $t0, $a1, exitfor     # If (save[i]=k) goto exitfor
            addi $s0, $s0, 1          # ++i;
            j for
exitfor:    or $v0, $zero, $s0        # return i;
            jr $ra
```