Feuille de travaux dirigés n° 5
Assembleur MIPS

Pour tous les exercices, on commencera par écrire l'algorithme demandé en C/C++ et l'on traduira ensuite ce code en suivant les schémas vus en cours.

Exercice 5.1

Traduire en assembleur MIPS les affectations C suivantes entre entiers signés :

1. `$v0` = `$t0` + `$t1` − `$t2`
2. `$v0` = `$t0` − `$t1` + `$t2`
3. `$v0` = `$t0` − (`$t1` + `$t2`)
4. `$v0` = 4 + `$t0`
5. `$v0` = 4 × `$t0`
6. `$v0` = 3 × `$t0`
7. `$v0` = (`$t0` + `$t1`)/8 + `$t2` % 2

Exercice 5.2

Écrire un programme MIPS plaçant dans le registre `$v0` le n-ième bit (à droite) du registre `$a0`, l'entier n étant une valeur contenue dans le registre `$a1`.

Exercice 5.3

Traduire en assembleur les structures de contrôle ci-dessous :

si *condition* alors
  action$_1$
sinon
  action$_2$
finsi
	choix *variable* selon
	  val$_1$: *action*$_1$
	  ...
	  val$_n$: *action*$_n$
	  défaut: *action*$_{défaut}$
	finchoix
		tantque *condition* faire
		  action
		ftq

répéter
  action
jusqu'à *condition*
	pour *variable* = *1* jusqu'à *100* faire
	  action
	finpour	

avec :
– *condition* : `$t0 == $t1`
– *variable* : `$t0`

Exercice 5.4

Écrire un programme en assembleur déclarant un tableau initialisé de 10 entiers sur 32 bits et en recherchant le plus petit et le plus grand élément.

Exercice 5.5

Écrire un programme en assembleur testant si une chaîne de caractères est un palindrome (i.e. telle que la chaîne inversée est identique à la chaîne elle-même). On considèrera des chaînes sans espace ni lettre accentuée.

Exercice 5.6

Écrire en assembleur MIPS la fonction `int impair(int x)` retournant `1` si `x` est impair et `0` sinon.

Exercice 5.7

Écrire une suite d'instructions MIPS mettant à l'adresse `Ad` le déterminant $ad - bc$ d'une matrice $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ dont les éléments (entiers sur 32 bits) sont rangés par ligne dans la mémoire à partir d'une adresse `Am`.

Exercice 5.8

Écrire une suite d'instructions MIPS calculant la somme des éléments d'un tableau d'entiers rangé à l'adresse `tab` de la mémoire ; le premier élément indique la taille du tableau.

Exercice 5.9

Traduire en assembleur MIPS le programme C suivant

```
int multrusse (int a, int b)
{
  int p = 0;
  while (b > 0) {
    if (b%2 == 1) {
       p += a;
    }
    a = 2*a;
    b = b/2;
  }
  return p;
}
```

Exercice 5.10

Écrire une fonction permettant de calculer le produit scalaire de deux vecteurs. Les arguments sont les adresses des deux vecteurs et leur taille. On supposera que le résultat est codable sur 32 bits.

Exercice 5.11

Traduire en assembleur MIPS le programme C suivant :

```
int combi (int n, int p)
{
  if (p==0 || p==n || n==0) {
    return 1;
  } else {
    return combi(n-1,p) + combi (n-1, p-1);
  }
}

int main(void)
{
  cout << combi(2,3) << endl;
  cout << combi(5,8) << endl;
}
```