9. Les modes d'adressage et la programmation des structures de contrôles

Cette section présente deux notions importantes pour l'élaboration des programmes assembleur, à savoir les techniques utilisées par le CPU pour l'accès aux opérandes (modes d'adressage. Voir section 9.1) et la programmation des structures de contrôles (section 9.2).

9.1. Les modes d'adressage du MIPS R3000

Un "mode d'adressage" est la technique utilisée par le CPU pour la localisation et l'accès aux opérandes. Dans ce qui suit, on présente les modes d'adressage les plus importants dans le cas du MIPS R3000:

- **Adressage immédiat :**

La valeur de l'opérande est précisée directement dans l'instruction. On appel ce type d'opérande "un opérande immédiat" (constante). Dans ce cas on n'a pas besoin d'un accès à la M.C.

Exemple :

```
li $t0, 120    #l'opérande 120 est l'opérande immédiat
```

- **Adressage direct :**

Dans ce cas, l'opérande est précisé au niveau de l'instruction à travers son nom, c'est à dire son adresse logique en M.C [1] .

Exemple:

```
var : .byte 120
.
.
.
lb $t0, var    #l'opérande source var  est associée à un adressage direct
```

- **Adressage par registre :**

Dans ce cas, l'opérande est précisé à travers un registre.

Exemple :

```
 li $t5, 120    #adressage immédiat)
 add $t0, $t5, 80     #l'opérande source 80 est associée à un adressage immédiat et
                      #  l'opérande source $t5 est associée à un adressage par registre.
```

[1] Il est à noter que lors de la traduction d'une instruction, le nom d'une variable (par exemple la variable var) sera remplacé par son adresse logique (offset ou déplacement). Voir la partie cours pour plus de détails sur les adresses logiques.

- **Adressage indirect :**

Dans ce mode d'adressage, le registre fournit par l'instruction ne représente pas l'endroit où se trouve l'opérande mais plutôt l'endroit qui contient l'adresse de l'opérande dans la M.C. Pour accéder à la donnée, le CPU fait un premier accès (au registre) pour avoir l'adresse de l'opérande puis un deuxième accès (à la M.C) pour avoir l'opérande. Ce mode d'adressage est symbolisé par des parenthèses "(" ")" qui entourent le nom d'un registre : **(registre)** . Ce mode d'adressage est généralement utilisé pour accéder aux éléments d'un tableau, comme le montre les deux exemples suivants.

Exemple 1 :

```
vect: .word 1, 2, 3
.
.
.
la $t0, vect        # mettre l'adresse de  vect dans $t0

lw $t1, ($t0)     # mettre le 1er élément de vect  c-a-d  1 dans $t1

add $t0, $t0, 4  #  passer à l'élément suivant.....
                  #  un élément de vect contient 4 octets et chaque octet est associé
                 #  à une adresse mémoire.

lw $t2, ($t0)     # mettre le 2eme élément de vect  c-a-d   2 dans $t2
```

Exemple 2 :

Soit un vecteur "vect" contenant trois éléments 1, 2 et 3 ayant la taille d'un mot (word). Ecrire un programme en assembleur qui permet de faire la somme des éléments de vect.

Solution :

```
.data
vect: .word 1, 2, 3

.text
.globl main
.ent main
main:

la $t0, vect        # mettre l'adresse de  vect dans $t0

lw $t1, ($t0)     # mettre le 1er élément de vect c-a-d  1 dans $t1

add $t0, $t0, 4  # passer à l'élément suivant.....
                  #  un élément de vect contient 4 octets et chaque octet est associé
                 #  à une adresse mémoire.

lw $t2, ($t0)     # mettre le 2eme élément de vect c-a-d  2 dans $t2
```

```
add $t0, $t0, 4  #  passer à l'élément suivant..... c-a-d 3eme élément

lw $t3, ($t0)     # mettre le 3eme élément de vect c-a-d  2 dans $t3


# ************additionner  $t1+$t2+$t3   **********************************
add $t4,$t1, $t2
add $t4,$t4, $t3

#************************ Afficher la somme
move $a0, $t4
li $v0, 1
syscall

li $v0, 10
syscall
.end main
```

9.2. Les instructions de contrôles

Les instructions de contrôles permettent de réaliser les structures de contrôles : les structures conditionnelles (IF-THEN, IF-THEN-ELSE), et les boucles.

Il existe deux types d'instructions de contrôles : Les instructions de branchement inconditionnel et de branchement conditionnel.

9.2.1. L'Instruction de branchement inconditionnel

L'instruction de branchement inconditionnel est l'instruction **j** ayant le format général suivant :

j <label> ou bien d'une manière équivalente **b <label>**

Cette instruction signifie un saut inconditionnel à l'instruction étiquetée par le label ou l'étiquette "label".

Exemple 1 :

```
.
.
.
j Sortir
.
.
.
Sortir :
li $v0, 10
syscall
.end main
```

9.2.2. Les instructions de branchement Conditionnel

Une instruction de branchement conditionnel permet de se brancher ou de se déplacer vers une étiquette ou label donné "label" dans le cas où une condition donnée (comparaison entre les deux opérandes sources : reg_{sr} et source) est vérifiée. Cette instruction est l'équivalent de la structure IF-THEN. Ce type d'instruction a le format général suivant :

b<relation_ abréviation> reg_{sr}, source, <label>

où :

- "source" est un registre ou une valeur immédiate entière.
- <relation_ abréviation> exprime le type de relations entre les opérandes reg_{sr} et source qui forment la condition.

Les instructions de branchement conditionnel les plus importantes sont données par le tableau suivant :

Branchement conditionnel	Signification
beq reg_{sr}, source, <label>	**IF reg_{sr} = source THEN se brancher vers <label>**
bne reg_{sr}, source, <label>	**IF reg_{sr} ≠ source THEN se brancher vers <label>**
blt reg_{sr}, source, <label>	**IF reg_{sr} < source THEN se brancher vers <label>**
ble reg_{sr}, source, <label>	**IF reg_{sr} ≤ source THEN se brancher vers <label>**
bgt reg_{sr}, source, <label>	**IF reg_{sr} > source THEN se brancher vers <label>**
bge reg_{sr}, source, <label>	**IF reg_{sr} ≥ source THEN se brancher vers <label>**
bltu reg_{sr}, source, <label>	**Version non signée de l'instruction blt**
bleu reg_{sr}, source, <label>	**Version non signée de l'instruction ble**
bgtu reg_{sr}, source, <label>	**Version non signée de l'instruction bgt**
bgeu reg_{sr}, source, <label>	**Version non signée de l'instruction bge**

Tableau 3.1. Quelques opérations de branchement conditionnel

Dans le cas où la condition compare l'opérande reg_{sr} avec 0 (source=0), l'opérande "source" peut être omis et le code opération étendu avec la lettre "z".