# TD 3 – Architecture logicielle : instructions mémoire

On considère le jeu d'instructions MIPS32.

## 1. Instructions mémoire

Initialement, le registre R0 contient 0, le registre R1 contient 0x95842103, le registre R2 contient 0x00001000 ; le contenu de la mémoire est 0 dans toutes les adresses, sauf celles décrites table 1.

| Adresse | Contenu |
|---|---|
| 0x00001000 | 0x12 |
| 0x00001001 | 0x34 |
| 0x00001002 | 0x56 |
| 0x00001003 | 0x78 |
| 0x00001004 | 0x9A |
| 0x00001005 | 0xBC |
| 0x00001006 | 0xDE |
| 0x00001007 | 0xF0 |

*Table 1 – Contenu de la mémoire*

En repartant à chaque fois de l'état initial, donner l'état des registres et des cases mémoire modifiés après l'exécution des instructions :

a) LW R1, 0(R2)
b) LW R1, 4(R2)
c) LW R1, 3(R2)
d) LB R1, 1(R2)
e) LBU R1, 2(R2)
f) LB R1, 5(R2)
g) LBU R1, 6(R2)
h) LH R1, 6(R2)
i) LHU R1, 6(R2)
j) SB R1, 0(R2)
k) SH R1, 0(R2)
l) SW R1, 0(R2)

## 2. Accès mémoire

### Exercice 1

On considère le plan mémoire de la table 2. Toutes les variables des programmes sont des *int.*

| Adresses | Contenu |
|---|---|
| 0x00001000 - 0x00001003 | x |
| 0x00001004 - 0x00001007 | t |
| 0x00001008 - 0x0000100B | z |
| 0x0000100C - 0x0000100F | M[0] |
| 0x00001010 - 0x00001013 | M[1] |
| 0x00001014 - 0x00001017 | M[2] |

*Table 2 – Plan mémoire*

a. Ecrire une séquence d'instructions qui correspond à l'affectation `x = x+2*z`

b. Ecrire une séquence d'instructions qui correspond au fragment de programme suivant,

```
z = M[0]+ x ;
t = M[2] + x ;
x = M[1] + z + t ;
```

**Exercice 2**

a. Donner la suite d'instructions pour exécuter le code C suivant, où les variables sont en mémoire à partir de l'adresse 0x10000010.

```
char D,E,F ;
D=E+F ;
```

b. Donner la suite d'instructions pour exécuter le code C suivant, où les variables sont en mémoire à partir de l'adresse 0x10000020.

unsigned `char H,J,K ;`

```
H=J+K ;
```

## 3. Alignement

La déclaration en C de la fig. 1 définit une occupation mémoire ; les valeurs sont notées en hexadécimal en commentaire. Le placement est aligné et en big endian. La structure est implantée à partir de l'adresse 0x00002000.

Donner le contenu des octets mémoire concernés.

```
struct     {
       int         a; /*x'11121314'*/
       double      b; /*x'2122232425262728'*/
       char*       c; /*x'31323334'*/
       char        d[7]; /*'A', 'B', 'C', 'D', 'E', 'F', 'G'*/
       int         f; /*x'61626364'*/
       short       e; /*x'5152'*/
} data;
```

Figure 1 – Déclaration de variables en LHN