UNIVIRSITE MOHAMMED V FACULTE DES SCIENCES DEPARTEMENT D'INFORMATIQUE	**Travaux dirigés** **Architecture des Ordinateurs** **SMI4**	**S.ZITI** **2019/2020**

TD N° 2 : Correction

Exercice 4 :

1. Donner les adresses physiques des mémoires telles que les adresses logiques sont comme suit :

   3500 : AB00 ➔ 3500*10 + AB00 (H) = 35000 + AB00 (H) = 3FB00*H*

   1CD0 : 1111 ➔ 1CD0*10 + 1111 (H) = 1CD00 + 1111 (H) = 1CE11*H*

   0022 : FFFF ➔ 0022*10 + FFFF (H) = 00220 + FFFF (H) = 1021F*H*

2. Proposer une (des) adresse(s) segment : offset pour les mémoires d'adresse physique :

   10000 ➔ 1000 : 0000 ; 0FFF :0010 ; 0EEF:1110 ; 0ABC : 5440

   FFFFF ➔ FFFF : 000F ; FFFE : 001F ; F111 : EEEF ; FABC : 543F

   00000 ➔ 0000 : 0000

Exercice 5 :

Soit un processeur disposant de 3 registres **8 bits**, un bus de données **8 bits**, un bus d'adresse de **4 bits**. Le jeu d'instruction du processeur est composé de 4 Instructions : **Mov** (Affectation), **Add** (Addition), **OR** (Ou logique), **AND** (ET Logique). Ces Instructions utilisent deux opérandes **8 bits** (Reg. ou mémoire), exemple :

Ins1 : MOV Reg1, [adr] *; Lis l'emplacement mémoire adr et stocke le résulat dans Reg1*

Ins2 : ADD [adr], Reg2 *; Ajoute le contenu de l'adresse à Reg2 et stocke résultat dans adr.*

Ins3 : Or Reg1, Reg2 *; Ou logique entre Reg1 et Reg2, résultat dans Reg1*

On suppose que le premier opérande est l'opérande destination. On suppose aussi qu'on ne peut avoir deux opérandes mémoire dans une même instruction.

1. Combien de bits sont nécessaires pour coder les opérations ?

➔ Nous avons un bit permet de coder deux valeur (le 0 et le 1), deux bits permettent de coder quatre valeurs (le 0, le 1, le 2 et le 3). Comme, il existe quatre opérations, donc deux bits sont nécessaires pour coder les opérations.

2. Combien de bits sont nécessaires pour coder les registres ?

➔ **Nous avons un bit permet de coder deux valeur (le 0 et le 1), deux bits permettent de coder quatre valeurs (le 0, le 1, le 2 et le 3). Comme, il existe trois registres, donc deux bits sont nécessaires pour coder les registres (il resterait une valeur libre : le 3).**

3. Les combinaisons possibles des opérandes sont appelées mode d'adressage, combien de modes d'adressage peut-t-on avoir, donnez leurs codes machine.

➔**Les modes d'adressage existants sont Reg/[adr], [adr]/Reg et Reg/Reg ([adr]/[adr] est éliminé par les spécifications car le processeur ne peut envoyer un signal de lecture R/W=1 et un signal d'écriture R/W=0 simultanément)**

4. Si les adresses sont codées par leur valeur et que le code chaque instruction est composée de 4 champs : **opcode modeAdressage. registre mémoire**, donnez le code machine de chaque instruction (**avec adr = 3**).

Opcode(MOV) = 00 **Mode@(Reg/[adr]) = 00** **Reg1 : 00**
Opcode(ADD) = 01 **Mode@([adr]/Reg) = 01** **Reg2 : 01**
Opcode(OR) = 10 **Mode@(Reg/Reg) = 10** **Reg3 : 10**
Opcode(AND) = 11 **[adr] : 11 (specification)**

Instruction	opcode	modeAdressage	Registre	Mémoire (ou reg)
Ins1	00	00	00	11
Ins2	01	01	01	11
Inst3	10	10	00	01

5. Si la première instruction est stockée à l'adresse 0, donnez le contenu des 3 premières adresses.

ADRESSE	CONTENU
0H	00000011
1H	01010111
2H	10100001

6. En reprenant les mêmes instructions de l'exemple précédents, et si on fait les hypothèses simplificatrices suivantes :

- Le microprocesseur met le même temps t_i pour décoder et exécuter chaque instruction ;
- Un accès à la mémoire nécessite t_m;
- Les codes des instructions sont au départ en mémoire.

a. Donnez le temps d'exécution de chaque instruction

⇨ **Ins1 : le processeur accède à la mémoire pour rechercher le code d'instruction en utilisant le temps t_m. Ensuite, il décode ce code avec t_i et revient vers la mémoire pour lire le deuxième opérande avec t_m. Enfin, il exécute l'instruction avec t_i ➔ $T1= 2* (ti + t_m)$**

⇨ **Ins2 : le processeur accède à la mémoire pour rechercher le code d'instruction en utilisant le temps t_m. Ensuite, il décode ce code avec t_i et revient vers la mémoire pour lire le premier opérande avec t_m. Enfin, il l'exécute l'addition avec t_i et revient vers la mémoire pour écrire le résultat dans le premier opérande avec t_m ➔ $T2= 2* ti + 3* t_m$**

⇨ **Ins1 : le processeur accède à la mémoire pour rechercher le code d'instruction en utilisant le temps t_m. Ensuite, il décode ce code avec t_i l'exécute l'instruction avec t_i sans aucun autre accès à la mémoire. ➔ $T3= 2* ti + t_m$**

b. Calculer le temps d'exécution de ce programme en fonction de t_i et t_m. On donne : Fréquence processeur = 100 Mhz, temps d'accès mémoire = 100 ns.

$$T=T1+T2+T3= 6* t_i + 6* t_m = 6 * 10 * 10^{-9} + 6 * 100 * 10^{-9} \ (s)$$

$$= 60 + 600 \ (ns) = 660 \ ns$$

$$t_m = 100 \ ns = 100 * 10^{-9} \ s$$

$$t_i = 1/(100 \ Mhz)=1/ (100*10^6 \ hz)= 10* 10^{-9} \ s$$