Département d'informatique **Corrigé type** **Licence 2eme Année**
UFA Sétif **Examen ADO** **Année 2012/2013**

Exercice 1 (4pts)

1)L'intervalle de représentation des nombres en complément à deux sur 3 bits est [-4, +3] mais seules les valeurs (-1, 0, 1, 2) sont autorisées en entrée car, au-delà, on ne peut pas représenter le résultat sur 3 bits signés(en complément à2), comme c'est le cas, par exemple, pour : $-2 \times -4 = 8$ et $-2 \times 3 = -6$, $-2 \times -2 = 4$. **(0.5pts)**

2) On peut donc écrire une table de vérité tronquée en ne mettant pas les valeurs interdites : (les entrées et les sorties sont en **C2) 0.5pts** par ligne juste = **2 pts**

A	$a_2a_1a_0$	$b_2b_1b_0$	B
-1	111	010	2
0	000	000	0
1	001	110	-2
2	010	100	-4

3)On a $b_0 = 0$ **0.5pts**

$b_1 = a_2a_1a_0 + a_2'a_1'a0 = a_0(a_1 \text{ ă } a_2)'$ **0.5pts**

$b_2 = a_2' (a_0 \text{ ă } a_1)$ **0.5pts**

Exercice 2 (7pts)

1)Taille(MC) =128K mots $=2^7 \times 2^{10} \times 2^2$ octets (le mot = 32 bits = 4 octets $=2^2$ octets) d'ou Taille(MC) = 2^{19} octets = **2^{19} unités adressables** alors taille(RA) = **19 bits**. (**1 pt)**

2) l'entier positif le plus grand représentable en complément à deux dans cette mémoire dépend de la taille du mot et non pas de l'adresse et puisque la taille du mot est 32 bits:
Alors c'est : **2^{31} -1** . (**0.5pts)**

3)pour qu'un nombre soit l'adresse d'un mot dans cette mémoire , il ne doit pas dépasser la taille du registre d'adresse et doit être positif alors : 698 , 3ABC sont acceptées et AD1FFE , -5 ne sont pas acceptées. (**1 pt)**
R**emarque : pas la peine de faire des calculs ,il suffit de compter le nombre de bits)**

4) Tout d'abord on représente 0.5072_8 en format IEEE à simple précision en effet :
$0.5072_8 = 0.101000111010_2 = 1.01000111010 \times 2^{-1}$
En calculant l'exposant biaisé et en respectant le format IEEE, on trouve :
0 01111110 01000111010000000000000
On aura sous forme hexadécimal: **3F23A000** **(1.5 pts** pour tout le calcul)

Les adresses avec les contenus sont: (**1 pt)**
M[0x1FE] = 0x3F, M[0x1FF] = 0x23, M[0x200] = 0xA0, M[0x201] = 0x00

5)Le nombre de lignes = 4K octets/16 octets = $2^{12} / 2^4 = 2^8$ **0.5 pts**
Alors **8 bits** sont nécessaires pour représenter les numéros de lignes **0.5 pts**
et **4 bits** pour représenter les numéros d'octets(16 octets = **2^4 octets**) **0.5 pts**
la zone nécessite **19 – (8 + 4) = 7 bits**. **0.5 pts**

Exercice 3 (7pts)

1°) 4pts = 1+1+1+1

a- sll $t2, $s2, 6

b- La fonction **xor** appliquée avec une suite de 1 complémente le contenu d'un registre Cependant on ne peut pas utiliser une seule instruction pour complémenter le contenu d'un registre 32 bits, il faut mettre le masque (**0×FFFFFFFF)** dans un registre (**$s1)**

Et on aura : **xor $s3, $s2, $s1**

Si on utilise **xori** l'immédiat sera sur 16 bit et pour les autres bits il y aura une extension automatique de zéros alors l'instruction **xori $s3, $s2, 0×FFFF** est correcte mais ne complémente que les 16 bits de poids faibles et les autres restent inchangés par contre **xori $s3, $s2, 0×FFFFFFFF** est incorrecte syntaxiquement.

c- slt $s3, $s4, $zéro ou slti $s3, $s4, 0

d- lw $t0 , 32($s3) (le premier élément TAB[0] pointe vers $s3 alors il faut ajouter 32 pour atteindre TAB[8] et pas 28

2°) 3 pts=1+1+1

En passant de l'hexadécimal au binaire ; on trouve que les deux instructions sont de type R Et donnent :

xor $t0 , $s0 , $s1

Slt $t1 ,$t0, $zéro

Les deux instructions déjà vues en TD leur fonction est : tester $s0 et $s1 s'ils ont des signes différents mettre $t1 à 1

Cours (2pts) : voir cours : comparaison CISC et RISC