CC-TD-M - LIF6 Architecture matérielle et logicielle

Lundi 23 avril 2012 - durée 1h30 - aucun document autorisé

I Représentation à virgule flottante (5 pts)

On travaille sur une machine où les flottants binaires sont codés sur 12 bits. On considère le code :

$c =$

11	10	9	8	7	6	5	4	3	2	1	0
s	e_c				m_c						

Le nombre représenté est $f(c) = (-1)^s \times (1, m_c)_2 \times 2^{e(c)}$. Si $1 \leq N(e_c) \leq 14$, alors $e(c) = N(e_c) - 2^3$ (on ne se préoccupe pas du codage des sous-normaux ni des valeurs exceptionnelles ici).

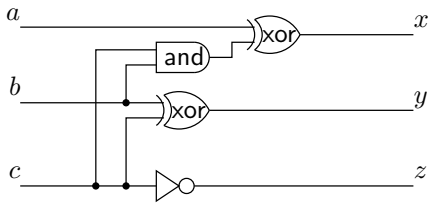
Q.I.1) - (1,5 pts) Quel nombre est représenté par le code octal $c = 7257_8$, si on le regarde comme le codage d'un flottant ? Donnez votre résultat sous la forme d'une représentation positionnelle à virgule en décimal.

Q.I.2) - (1,5 pts) Comment est représenté le nombre $(3, 2)_{10}$ en représentation positionnelle à virgule en binaire ? Détaillez un peu vos calculs.

Q.I.3) - (2 pts) Comment peut-on représenter le nombre $(3, 2)_{10}$ dans le format flottant binaire décrit ci-dessus ? Vous effectuerez un arrondi au plus proche, et exprimerez votre résultat sous la forme d'un code hexadécimal.

II Un circuit combinatoire (4 pts)

On considère le circuit combinatoire suivant, dont les entrées sont a, b et c, et les sorties x, y et z :



Q.II.1) - (1 pt) Donnez des expressions booléennes pour x, y et z en fonction de a, b et c.

Q.II.2) - (1 pt) Complétez la table de vérité suivante :

a	b	c	x	y	z
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Q.II.3) - (2 pts) Quelle est la fonction réalisée par le circuit considéré (exprimez $(xyz)_2$ en fonction de $(abc)_2$) ?

III Décompte des bits non-nuls (6 pts)

En C, le type `unsigned char` permet de coder des entiers naturels sur 8 bits. A titre d'exemple, on donne la fonction C suivante qui retourne le nombre de bits non-nuls de son argument (rappelons que `n/2` et `n%2` désignent respectivement le quotient et le reste dans la division euclidienne de l'entier `n` par 2) :

```
int nzbcount(unsigned char m) {
  n=m; i=0;
  while(n != 0) {
    if(n%2 == 1) i++;
    n=n/2;
  }
  return(i);
}
```

On va mettre au point une autre fonction pour compter le nombre de bits non-nuls d'un `unsigned char`. Si `a` et `b` sont deux entiers naturels, on rappelle que `a&b` calcule en C le « ET bit-à-bit » entre `a` et `b`.

Q.III.1) - (1 pt) Pour `a` $= (00000001)_2$, `b` $= (10111100)_2$, `c` $= (10000000)_2$, calculer `(a-1)&a`, `(b-1)&b` et `(c-1)&c`. Détaillez vos calculs (posez à chaque fois la soustraction en binaire, puis effectuez le « ET bit-à-bit »).

Q.III.2) - (2 pts) Si `n` est un `unsigned char`, que permet de faire l'opération `(n-1)&n` ?

Q.III.3) - (2 pts) Déduisez de la question précédente une nouvelle fonction C pour calculer le nombre de bits non nuls d'un `unsigned char`.

Q.III.4) - (1 pt) Avec la fonction initiale `nzbcount`, combien d'itérations de la boucle `while` sont nécessaires pour compter le nombre de bits non-nuls d'un `unsigned char` ? Même question pour votre fonction.

IV Mémoire cache (5 pts)

On considère le programme C suivant :

```
int main(void) {
  double A[8], B[8], C[8];
  int i;
  for(i=0; i<8; i++) C[i] = A[i] + B[i];
  ...
}
```

On suppose que l'on compile et que l'on exécute ce programme sur un ordinateur qui ne dispose que d'un seul niveau de cache de données direct : ce cache comporte 4 entrées de 32 octets (on ne se préoccupe pas du chargement des instructions, ni de la variable `i` dans cet exercice). On note les lignes de cache $\ell_0, \ell_1, \ell_2, \ldots$, et les entrées du cache e_0, e_1, e_2, e_3. Les flottants de type `double` sont d'une taille de 64 bits. Les tableaux sont stockés de manière contigüe en mémoire, dans l'ordre de leur déclaration.

Q.IV.1) - (1 pt) Combien de flottants de type `double` peut contenir une ligne de cache ?

Q.IV.2) - (1 pt) Combien de lignes de caches consécutives occupe chacun des tableaux `A`, `B` et `C` ?

Q.IV.3) - (1,5 pt) En supposant que `A[0]` est aligné sur le début de la ligne de cache ℓ_0, indiquez à quelles lignes de cache appartiennent les élément de `A`, `B` et `C`. Indiquez également dans quelles entrées du cache seront rangés ces élements lorsqu'ils seront accédés.

Q.IV.4) - (1,5 pts) Dans un tableau de la forme indiquée ci-dessous, indiquez quelle ligne de cache contient chaque entrée du cache à la fin de chaque itération de la boucle `for(i=0; i<8; i++) C[i] = A[i] + B[i]` ; (Vous placerez un « ? » quand on ne peut pas savoir). Indiquez également le nombre de *cache miss* qui ont eu lieu au cours de chaque itération.

`i`	0	1	2	3	4	5	6	7
e_0								
e_1								
e_2								
e_3								