CC-TD-M – vendredi 22/04/2001, Thémis 69 – LIF6

Mathilde Noual – Nicolas Louvet

Aucun document autorisé. Les calculatrices, ordinateurs, téléphones et autres sont interdits.
Durée 1h30. Le barème est susceptible de modifications.

I Représentation à virgule flottante (6 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 exeptionnelles ici).

Q.I.1) - (2 pts) Quel nombre x_1 est représenté par le code hexadécimal $c = 637_H$, 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 pt) Quel est le nombre flottant x_2 immédiatement suppérieur à x_1 ?
– Donnez votre résulat sous la forme $x_2 = (-1)^s \times (1, m)_2 \times 2^e$,
– puis donnez le codage de x_2 en hexadécimal.
Expliquez succinctement votre raisonnement.

Q.I.3) - (1 pt) Comment est représenté le nombre $(13/3)_{10}$ en représentation positionnelle à virgule en binaire ? Détaillez un peu vos calculs.

Q.I.4) - (2 pts) Comment peut-on représenter le nombre $(13/3)_{10}$ 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 binaire.

II Encodeur octal (5 pts)

Le but est de concevoir un encodeur octal : il s'agit d'un circuit à 8 entrées $e_0, \ldots, e_7$ et 3 sorties s_0, s_1, s_2. Si l'entrée e_i est à 1 et que toutes les autres sont à 0, on veut que les sorties soient telles que $(s_2 s_1 s_0)_2 = i$.

Q.II.1) - (2 pts) Complétez un tableau de la forme suivante, en supposant qu'au plus l'une des entrées e_i peut être activée à la fois (dans les colonnes $e_0, \ldots, e_7$, n'indiquez que les 1, pas les 0).

e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	s_2	s_1	s_0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Q.II.2) - (2 pts) Exprimez les sorties s_2, s_1 et s_0 chacune sous la forme d'une somme de quatre littéraux.

Q.II.3) - (1 pt) Répresentez le circuit logique demandé (vous pouvez utilisez des portes AND à 4 entrées).

III Comparateur d'entiers (5 pts)

Le but de cette partie est de proposer un circuit pour comparer entre eux deux entiers codés en binaire sur trois bits, $(a_2 a_1 a_0)_2$ et $(b_2 b_1 b_0)_2$. La sortie y du circuit devra prendre pour valeur 1 si $(a_2 a_1 a_0)_2 = (b_2 b_1 b_0)_2$, 0 sinon. On souhaite un circuit simplifié, n'utilisant que des portes logiques NOT, AND, OR, XOR.

Q.III.1) - (1 pt) On note x_i le résultat de la comparaison des bits a_i et b_i. Donnez la table de vérité de x_i en fonction de a_i et b_i.

Q.III.2) - (1 pt) De la table de vérité, déduisez une expression simple de x_i (deux portes sont suffisantes).

Q.III.3) - (2 pts) Donnez l'expression de y en fonction des x_i, puis en fonctions des a_i et des b_i.

Q.III.4) - (1 pt) Représentez le circuit du comparateur 3 bits demandé.

IV Décompte des bits non-nuls (4 pts)

En C, le type `unsigned char` permet de coder des entiers naturels sur 8 bits. On rappelle que `n/2` et `n%2` désignent respectivement le quotient et le reste dans la division euclidienne de l'entier naturel `n` par 2. On considère le programme suivant qui permet d'afficher le nombre de bits non-nuls de l'entier entré (en notation décimal) par l'utilisateur :

```
int main(void) {
  unsigned char m, n;
  unsigned int i;

  printf("Entrez un unsigned char m      : ");
  scanf("%hhd", &m);
  n=m; i=0;
  while(n != 0) {
    if(n%2 == 1) i++;
    n=n/2;
  }
  printf("Nombre de bits non-nuls dans m : %d\n", i);

  return(0);
}
```

Voici un exemple d'utilisation du programme :

```
nlouvet@mini:~$ ./test
Entrez un unsigned char m      : 10
Nombre de bits non-nuls dans m : 2
```

En effet $(10)_{10} = (00001010)_2$, donc l'écriture de 10 en base 2 comporte bien 2 bits non-nuls.

Q.IV.1) - (1 pt) Supposons que l'utilisateur entre l'entier $\texttt{m} = (138)_{10} = (10001010)_2$, reportez dans un tableau de la forme suivante les valeurs de `n` (en écriture binaire) et `i` au cours des itérations de la boucle `while`. Précisez également à quelle étape la boucle prend fin. Au total, combien d'itérations ont été effectuées ?

	`n`	`i`
avant la première itération	$(10001010)_2$	0
à la fin de la 1ère itération		
à la fin de la 2ème itération		
à la fin de la 3ème itération		
⋮	⋮	⋮

Q.IV.2) - (2 pts) Justifiez le fait que le programme permet bien d'afficher le nombre de bits non-nuls de l'entier `m` entré par l'utilisateur.

Q.IV.3) - (1 pt) Avec le programme précédent, combien d'itérations de la boucle `while` sont nécessaires pour compter le nombre de bits non-nuls d'un `unsigned char` ?