Architecture des ordinateurs

Solutions à l' examen mi-terme

COMPSC/itec 3610

1 Géneral

1. Je disais une fois dans notre cours d'architecture des ordinateurs :

   > En Français, on exagère la prononciation des "e" à la fin du vers dans les chansons **ou** dans la poésie[1].

   Est-ce que je suis correct quand j'utilise le "ou" (logique) ? Si oui, est-ce que le "ou" dans la langue Française envisage la même notion que celle exprimée par un ordinateur ?
   Si non, auriez-vous la bonté de me corriger ?

   Correction : *En Français, on exagère la prononciation des "e" à la fin du vers dans les chansons* **et** *dans la poésie.*

2. Indiquer la valeur codée par la suite 1101100101110101 qui représente un entier signé en complément à 2 sur 16 bits.

   Correction : *C'est un nombre négatif. Complément à 2 : 0010011010001011 donc* -9867.

   Même question avec la suite 1001000011101101.

   Correction : *C'est un nombre négatif. Complément à 2 : 0110111100010011 donc* -28435.

3. Représentation binaire des entiers négatifs
   (a) Coder sur 4 bits les entiers 7, 2, 0, -2, -7 et -8 avec les représentations suivantes :
   - signe et valeur absolue ;

     Correction : *0111, 0010, 0000 ou 1000, 1010, 1111, n/a*

   - complément à 1 ;

     Correction : *0111, 0010, 0000 ou 1111, 1101, 1000, n/a*

   - complément à 2.

     Correction : *0111, 0010, 0000, 1110, 1001, 1000*

   (b) Coder les entiers 61 et -61 sur un octet en utilisant la représentation par le signe et la valeur absolue. Montrer que l'addition binaire de ces entiers ainsi codés produit un résultat incorrect. Montrer qu'en revanche le résultat est correct si ces entiers sont codés en utilisant la représentation par le complément à 2.

   Correction :

   Signe et valeur absolue :

	00111101	(61)
+	10111101	(−61)
	11111010	(−122)

   Complément à deux :

	00111101	(61)
+	11000011	(−61)
	00000000	(0)

[1] Au clair de la lune
Mon ami Pierrot..

4. Effectuer en binaire (8 bits) les opérations $1-2$, $51+127$, $-3-127$, $-127+127$, $-63-63$. Préciser, pour chaque opération, la retenue et le débordement.

Correction : *On code les nombres négatifs en complément à 2.*
Débordement :
- *L'addition de deux nombres de signes différents ne produit jamais de débordement (la valeur absolue du résultat est toujours inférieure au maximum des valeurs absolues des deux opérandes).*
- *L'addition de deux nombres de même signe produit un débordement si le signe du résultat est différent du signe des deux opérandes.*

	00000001	(1)		00110011	(51)		11111101	(−3)
+	11111110	(−2)	+	01111111	(127)	+	10000001	(−127)
	11111111	(−1)		10110010	(−78)		01111110	(126)
retenue : 0, débordement : 0			*retenue : 0, débordement : 1*			*retenue : 1, débordement : 1*		

	10000001	(−127)		11000001	(−63)
+	01111111	(127)	+	11000001	(−63)
	00000000	(0)		10000010	(−126)
retenue : 1, débordement : 0			*retenue : 1, débordement : 0*		

5. Représentation des réels
   (a) En virgule fixe, décoder le nombre binaire 11.011 puis coder en binaire le réel 11.625.

   Correction :

$$11.011_2 = \left[1\times 2^1 + 1\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2} + 1\times 2^{-3}\right]_{10} = \left[2+1+0.25+0.125\right]_{10} = 3.375_{10}$$
$$11.625_{10} = \left[8+2+1+0.5+0.125\right]_{10} = \left[2^3+2^1+2^0+2^{-1}+2^{-3}\right]_{10} = 1011.101_2$$

   (b) En virgule flottante normalisée, coder en binaire au format simple précision le réel 12.575

   Correction :

$$12.575_{10} = 1100.100\underline{1001}\ldots_2 = \left[0.11001001\underline{1001}\ldots\times 10^{100}\right]_2$$
$$0|10000011|11001001001100110011001|$$

   puis effectuer le codage inverse.

   Correction : *bit de signe = 0 $\longrightarrow$ nombre positif.*
   exposant biaisé $= 10000011_2 = 131_{10} \longrightarrow$ *exposant :* $10000011 - 01111111 = 100_2 = 4_{10}$
   la mantisse est normalisée : 0.11001001001100110011001

$$\begin{aligned}\left[0.11001001001100110011001\times 10^{100}\right]_2 &= 1100.10010011001100110011001_2\\ &= \left[2^3+2^2+2^{-1}+2^{-4}+2^{-7}+2^{-8}+2^{-11}\right.\\ &\quad \left.+2^{-12}+2^{-15}+2^{-16}+2^{-19}\right]_1 0\\ &= 12.574998855590820312 5_{10}\end{aligned}$$

6. Opérations en virgule flottante.
   Soit $a = \left[0.10010\times 10^{101}\right]_2$ et $b = \left[0.11010\times 10^1\right]_2$. Calculer $a+b$ et $a\times b$.

   Correction : *Avant de faire l'addition, il faut que les deux exposants soient égaux ($a = 1001\times 10^1$, $b = 0.1101\times 10^1$). Pour faire la multiplication, on multiplie les mantisses puis on additionne les exposants.*

Dans les deux cas, le résultat doit ensuite être normalisé.

$$\begin{array}{rr} & 1001.0000 \times 10^{1} \\ + & 0.1101 \times 10^{1} \\ \hline & 1001.1101 \times 10^{1} \\ = & 0.10011101 \times 10^{101} \end{array}$$

$$\begin{array}{rr} & 0.1001 \times 10^{101} \\ \times & 0.1101 \times 10^{1} \\ \hline & 1001 \\ & 1001 \\ + & 1001 \\ \hline & 0.01110101 \times 10^{110} \\ = & 0.1110101 \times 10^{101} \end{array}$$

7. Codage et transmission de caractères

   Lors de la transmission d'informations entre un ordinateur et ses périphériques (clavier, écran, imprimante, ...), ou entre ordinateurs via un réseau, il est nécessaire de s'assurer que les informations ont été transmises convenablement. Différents moyens de détection (et éventuellement de correction) des erreurs de transmissions peuvent être mis en oeuvre, la plus simple étant le *contrôle de parité*.

   Le principe du contrôle de parité est le suivant : au moment de la transmission l'émetteur ajoute un bit supplémentaire aux n bits que comporte le code d'origine, ce bit supplémentaire (le *bit de parité*) étant positionné de tel sorte que le nombre total de bits à 1 soit paire. Au moment où le recepteur reçoit une donnée, il vérifie sa parité : si le nombre total de bits reçu est impair, une erreur de transmission s'est produite. Dans ce cas le récepteur demande alors la retransmission de la donnée erronée.

   Ce type de contrôle est utilisé lors de la transmission de caractères. En effet, le code ASCII utilisé pour coder les caractères nécessite seulement 7 bits. Pour effectuer un contrôle de parité, on modifie ce codage en ajoutant un 8éme bit (bit de poids fort) de tel sorte que la parité de l'octet soit pair (le nombre de bits valant 1 soit pair). On obtient ainsi un codage sur un octet.

   Exemple :

Caractère	ASCII (héxadécimal)	ASCII (binaire)	OCTET TRANSMIS (bits de parité en gras)
a	61	110 0001	**1**110 0001
c	63	110 0011	**0**110 0011
m	6d	110 1101	**1**110 1101

   Complétez le tableau pour les caractères suivants :

Caractère	ASCII (héxadécimal)	ASCII (binaire)	octet transmis
B			
L			

   Un perfectionnement de cette méthode de détection consiste à transmettre des blocs de 7 octets consécutifs (eventuellement complétés par des octets valant 0), auquels on ajoute un 8éme octet formé des 8 bits rétablissant la parité longitudinale.

   Par exemple pour transmettre la chaîne de caractères `'ac'`, on transmet :

```
1110 0001                <— octet représentant le 'a'
0110 0011                <— octet représentant le 'c'
0000 0000                <— octets nuls de complément
0000 0000
0000 0000
0000 0000
0000 0000
1000 0010   <— octet regroupant les bits de parité longitudinale
```

   En supposant que l'appareil récepteur reçoit le bloc suivant :

1110 0010
0110 0101
1110 0111
0100 1101
1110 1110
0000 0000
0000 0000
1110 0101

Ce bloc présente-t'il une erreur de transmission ?

Correction : *Le contrôle de parité permet de détecter que la dernière ligne est incorrecte.*

2 Arithmétique des ordinateurs

Dans toute cette section les nombres seront **codés sur 8 bits**.

2.1 Codage des entiers

1. Codez en binaire pur les nombres décimaux : 15, 122 et 223.

   Correction : *(a)* $15 = 1 + 2 + 4 + 8$. *D'où* $15_{10} = 00001111_2$.
   (b) $122 = 64 + 32 + 16 + 8 + 2$. *D'où* $122_{10} = 01111010_2$.
   (c) $223 = 128 + 64 + 16 + 8 + 4 + 2 + 1$. *D'où* $223_{10} = 11011111_2$.

2. Codez, en représentation avec signe et valeur absolue, les nombres décimaux -122, -15, 15 et 122.

   Correction : *(a)* $-122_{10} = 11111010_2$.
   (b) $-15_{10} = 10001111_2$.
   (c) $15_{10} = 00001111_2$.
   (d) $122_{10} = 01111010_2$.

3. Codez, dans un système avec complément à deux, les nombres décimaux -122, -15, 15 et 122.

   Correction : *(a)* -122_{10}. *Dans un tel système, les nombres négatifs sont représentés par le complément à deux de leur valeur absolue. Codage en binaire pur de* 122 *:* 01111010*; codage en complément à un :* $C_1(122) = 10000101$*; codage en complément à deux :* $C_2(122) = 10000110$. *D'où :* $-122_{10} = 10000110_2$.
   (b) -15_{10}. *Codage en binaire pur de* 15 *:* 00001111*; codage en complément à un :* $C_1(15) = 11110000$*; codage en complément à deux :* $C_2(15) = 11110001$. *D'où :* $-15_{10} = 11110001_2$.
   (c) 15. *Dans un tel système, les nombres positifs sont représentés par leur codage binaire pur :* $15_{10} = 00001111_2$.
   (d) 122. $122_{10} = 01111010_2$.

4. Réalisez, dans un système en complément à deux sur huit bits, le calcul : $59 - 107$. Vous vérifierez le résultat obtenu.

   Correction : – *Dans un tel système on n'effectue pas une soustraction mais l'addition de* 59 *et de* -107.
   – *Le nombre positif* 59 *est représenté par son codage binaire pur sur huit bits :* $59 = 00111011_2 = 32 + 16 + 8 + 2 + 1$.
   – *Le nombre* -107 *étant négatif, il est représenté par le codage en complément à deux de sa valeur absolue.* $107 = 64 + 32 + 8 + 2 + 1 = 01101011_2$. *D'où,* $C_1(107) = 10010100$ *et* $C_2(107) = 10010101$.

– *Addition des deux nombres :*

$$\begin{array}{rr} & 00111011 \\ + & 10010101 \\ \hline & 11010000 \end{array}$$

– *Le nombre* 11010000 *est le codage d'un nombre négatif puisque son bit de poids fort est 1 : c'est la représentation en complément à deux de la valeur absolue du nombre. Or* $C_1(11010000_2) = 00101111$ *et* $C_2(11010000_2) = 00110000$ *et* $00110000_2 = 32 + 16 = 48$. *Le résultat du calcul est donc :* -48.

2.2 Addition d'entiers

1. Calculez l'addition des deux nombres décimaux -122 et 15 dans un système avec complément à deux.

   Correction : *Dans un tel système, on effectue l'addition des représentations des deux nombres : le codage binaire pur du nombre positif et le codage en représentation à deux du nombre négatif.*

$$\begin{array}{rr} & 10000110 \\ + & 00001111 \\ \hline & 10010101 \end{array}$$

2. Convertissez le résultat en base 10.

   Correction : *Le bit de poids fort du nombre binaire* 10010101 *étant « 1 », le nombre est négatif, et ce code est celui du complément à deux de la valeur absolue du nombre. Pour trouver la valeur absolue, on prend le complément à deux de* 10010101 *(le complément à deux est une fonction involutive : le complément à deux du complément à deux est l'identité).*
   $C_1(10010101) = 01101010$ *et* $C_2(10010101) = 01101011$. *Or* $01101011_2 = 0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 64 + 32 + 8 + 2 + 1 = 107$. *Donc :* $10010101_2 = -107_{10}$.

3 Algèbre de Boole et table de Karnaugh

1. Proposez une expression booléenne ayant pour table de vérité la table ci-dessous :

A	B	C	D	$f(A,B,C,D)$
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

   Correction : *La solution classique sous forme de somme de produits :*

$$\begin{aligned} f(A,B,C,D) = \quad & \overline{A}.\overline{B}.\overline{C}.\overline{D} + \overline{A}.\overline{B}.\overline{C}.D + \overline{A}.\overline{B}.C.\overline{D} + \overline{A}.\overline{B}.C.D + \overline{A}.B.\overline{C}.D \\ & + \overline{A}.B.C.D + A.\overline{B}.\overline{C}.\overline{D} + A.\overline{B}.\overline{C}.D + A.\overline{B}.C.\overline{D} + A.\overline{B}.C.D + A.B.C.D \end{aligned}$$

CD \ AB	$A\,B$	$\overline{A}\,B$	$\overline{A}\,\overline{B}$	$A\,\overline{B}$
$C\,D$	1	1	1	1
$\overline{C}\,D$	0	1	1	1
$\overline{C}\,\overline{D}$	0	0	1	1
$C\,\overline{D}$	0	0	1	1

FIG. 1 – Table de Karnaugh correspondant à la table de vérité et à l'expression précédentes.

2. Simplifiez l'expression booléenne de la question précédente au moyen d'une table de Karnaugh.

   Correction : *La figure 1 présente la table de Karnaugh désirée.*
   On en déduit l'expression booléenne simplifiée :

$$f(A,B,C,D) = \overline{B} + C.D + \overline{A}.D.$$

4 Circuits logiques et table de Karnaugh

1. Réalisez le circuit logique correspondant à l'expression booléenne simplifiée trouvée à la question précédente.

   Correction : *A venir.*

2. Réaliser un circuit logique qui implémente la fonction F.

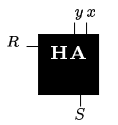
$$F = (A + B + C) \cdot (A + \overline{B} + \overline{C}) \cdot (\overline{A} + \overline{B} + \overline{C})$$

   Correction : *S.V.P. voir correction des TP 2.*

5 Additionneur

1. Rappeler les principes d'un demi-additionneur puis d'un additionneur complet. Déduire de ces principes un circuit logique qui implémente le complément à 2 sur n bits.

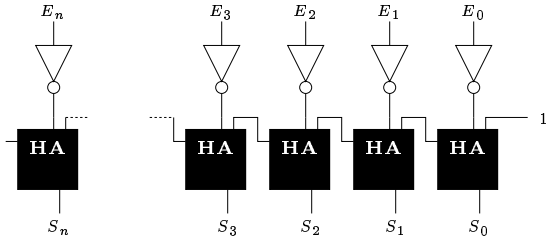
   Correction : *Le demi-additionneur possède deux entrées (x et y) et deux sorties (R et S). S correspond au bit de rang zéro du résultat de l'addition binaire de x et y, R au bit de rang 1 (retenue).*



x	y	R	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = x \oplus y$$
$$R = x \cdot y$$

Un additionneur complet s'obtient en enchaînant des demi-additionneurs de manière à propager correctement la retenue.
On obtient selon le même principe le circuit effectuant un complément à deux :



Circuit logique

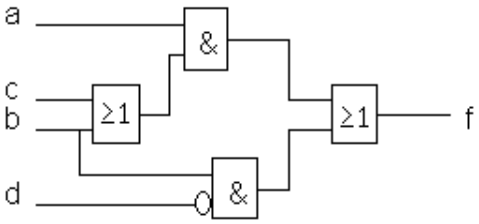
1) $f(a,b,c,d) = \bar{c}\bar{d}b + cab + \bar{c}dab + cd\bar{a}b + ca\bar{b}$

- tableau de Karnaugh :

cd \ ab	00	01	11	10
00	0	1	1	0
01	0	0	1	0
11	0	0	1	1
10	0	1	1	1

donc $f = ab + ac + b\bar{d} = a.(b + c) + b\bar{d}$

- schéma à l'aide de portes ET, OU et inverseurs



- on peut regrouper les 0 dans le tableau de Karnaugh ou utiliser le théorème de Morgan :

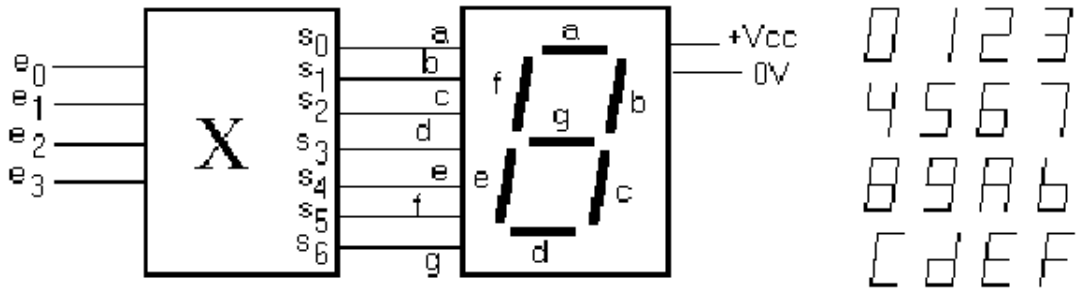
$\bar{f} = \bar{a}d + \bar{b}\bar{c} + \bar{a}\bar{b}$

2) $f(a,b,c,d) = da + c(d\bar{b} + \bar{a}b) + \bar{d}(c\bar{b} + ab) + \bar{c}\bar{d}a$

$\bar{f} = \bar{a}.\bar{c}$ donc f=a+c

A venir.

afficheur 7 segments



Trouver le schéma du composant X. Ses 4 entrées correspondent à la représentation binaire d'un chiffre entre 0 et 15.

Il faut fournir en sortie les 7 signaux nécessaires à l'affichage du chiffre hexadécimal correspondant.

On suppose qu'il faut un 0 pour allumer un segment, et un 1 pour l'éteindre.

table de vérité

regroupons dans un table l'état désiré pour les sorties dans chaque cas. L'ordre n'a pas d'importance.

décimal		hexa		binaire					a	b	c	d	e	f	g
0		0		0	0	0	0		0	0	0	0	0	0	1
1		1		0	0	0	1		1	0	0	1	1	1	1
2		2		0	0	1	0		0	0	1	0	0	1	0
3		3		0	0	1	1		0	0	0	0	1	1	0
4		4		0	1	0	0		1	0	0	1	1	0	0
5		5		0	1	0	1		0	1	0	0	1	0	0
6		6		0	1	1	0		0	1	0	0	0	0	0
7		7		0	1	1	1		0	0	0	1	1	1	1
8		8		1	0	0	0		0	0	0	0	0	0	0
9		9		1	0	0	1		0	0	0	0	1	0	0
10		A		1	0	1	0		0	0	0	1	0	0	0
11		B		1	0	1	1		1	1	0	0	0	0	0
12		C		1	1	0	0		0	1	1	0	0	0	1
13		D		1	1	0	1		1	0	0	0	0	1	0
14		E		1	1	1	0		0	1	1	0	0	0	0
15		F		1	1	1	1		0	1	1	1	0	0	0

recherche des équations

On peut maintenant analyser chaque sortie indépendamment, pour déterminer les équations. Nous allons utiliser des tableaux de Karnaugh

e1e0

e3e2 \ ***a***	00	01	11	10
00	0	1	0	0
01	1	0	0	0
11	0	1	0	0
10	0	0	1	0

$$a = \overline{e_1}e_0\,(\overline{e_3}\overline{e_2} + e_3e_2) + \overline{e_3}e_2\overline{e_1}\overline{e_0} + e_3\overline{e_2}e_1e_0$$

e1e0

e3e2 \ ***b***	00	01	11	10
00	0	0	0	0
01	0	1	0	1
11	1	0	1	1
10	0	0	1	0

$$b = e_2\overline{e_0}\,(e_1+e_3) + e_3e_1e_0 + \overline{e_3}e_2\overline{e_1}e_0$$

e1e0

e3e2

c	00	01	11	10
00	0	0	0	1
01	0	0	0	0
11	1	0	1	1
10	0	0	0	0

$$c = e_3e_2\,(e_1+\overline{e_0}) + \overline{e_3}\,\overline{e_2}e_1\overline{e_0}$$

e1e0

e3e2

d	00	01	11	10
00	0	1	0	0
01	1	0	1	0
11	0	0	1	0
10	0	0	0	1

$$d = e_2e_1e_0 + \overline{e_3}\,\overline{e_2}\,\overline{e_1}e_0 + \overline{e_3}e_2\overline{e_1}\,\overline{e_0} + e_3\overline{e_2}e_1\overline{e_0}$$

e1e0

e3e2

e	00	01	11	10
00	0	1	1	0
01	1	1	1	0
11	0	0	0	0
10	0	1	0	0

$$e = \overline{e_3}\,(e_0 + e_2\overline{e_1}) + \overline{e_2}\,\overline{e_1}e_0$$

e1e0

e3e2

f	00	01	11	10
00	0	1	1	1
01	0	0	1	0
11	0	1	0	0
10	0	0	0	0

$$f = \overline{e_3}\,\overline{e_2}(e_1+e_0) + \overline{e_3}e_1e_0 + e_3e_2\overline{e_1}e_0$$

e1e0

e3e2

g	00	01	11	10
00	1	1	0	0
01	0	0	1	0
11	1	0	0	0
10	0	0	0	0

$$g = \overline{e_3}\,(\overline{e_2}\,\overline{e_1} + e_2e_1e_0) + e_3e_2\overline{e_1}\,\overline{e_0}$$