EXERCICES DIRIGES
Représentation des informations
CORRECTION

Exercice 1. Circuit logique

1. $(423)_{10} = 2^8+2^7+2^5+2^2+2^1+2^0$=**$(110100111)_2$**
$(75)_{10} = 8^2+8^1+3*8^0$=**$(113)_8$**
$(11010001)_2 = 2^7 + 2^6 + 2^4 + 2^0$ = **$(209)_{10}$**
$(A92)_{16} = 10 * 16^2 + 9 * 16^1 + 2 * 16^0$ = **$(2706)_{10}$**
$(A9F)_{16}$ = **$(1010\ 1001\ 1111)_2$**
$(11010001)_2$ = **$(321)_8 = (D1)_{16}$**

2.
- bit de signe à 0 => **0**0011101 est positif => la suite binaire derrière le bit de signe code la valeur absolue du nombre : $(0011101)_2 = 2^4+2^3+2^2+2^0$=**$(29)_{10}$**
$(00011101)_2$ =**$(+29)_{10}$**

- bit de signe à 1=> **1**100 0001 1000 0110 est négatif
on calcule son complément à 2

	1100 0001 1000 0110
complément à 1	0011 1110 0111 1001
complément à 2	0011 1110 0111 1010

soit 256 * (32+16+8+4+2)+ (64+32+16+8+2)=256 *(62)+122=15994
donc le nombre est : **$(-15994)_{10}$**

3.
$(- 444)_{10}$: nombre négatif
$(444)_{10}= 2^8+2^7 +2^5+2^4+2^3+2^2$ donc $(+ 444)_{10}=(0000\ 0001\ 1011\ 1100)_2$
valeur signée : $(- 444)_{10}=(1000\ 0001\ 1011\ 1100)_2$
complément à 2 : $(- 444)_{10}$=cplt2$(0000\ 0001\ 1011\ 1100)_2$ = **$(1111\ 1110\ 0100\ 0100)_2$**

$(1975)_{10}$: nombre positif => même représentation en valeur signée et en complément à 2
$(1975)_{10}= 2^{10}+2^9+2^8+2^7 +2^5+2^4+2^2 +2^1+2^0=(11110110111)_2$
$(1975)_{10}= (0000\ 0111\ 1011\ 0111)_2$

4.$(125,50)_{10}$: $(125)_{10}$=$2^6+2^5+2^4+2^3+2^2+2^0$=$(1111101)_2$ et $(0,5)_{10}$=$(0,1)_2$ car 0,5*2=1=1+**0**
$(125,50)_{10}$ =$(1111101,1)_2$= $1,1111011 * 2^6$

exposant biaisé = 6 + 127 = 133 = 10000101
signe positif

0 10000101 11110110000000000000000

soit **$(42FB0000)_{16}$**

$(-1514,125)_{10}$: $(1514)_{10}=2^{10}+2^8+2^7+2^6+2^5+2^3+2^1= (10111101010)_2$
$(0,125)_{10}$=$(0,001)_2$ car *0,125**2= 0,25=**0**+*0,25* -> *0,25**2=0,5=**0**+*0,5* -> *0,5**2=1=**1**+***0***
$(-1514,125)_{10}$= $(10111101010,001)_2$=$1,0111101010001 * 2^{10}$

exposant biaisé = 10 + 127 = 137 = 10001001
signe négatif

1 10001001 01111010100010000000000

soit **$(C4BD4400)_{16}$**

5. $(C75A21C0)_{16}$

1 10001110 10110100010000111000000

Bit de signe à 1 : signe négatif
exposant : **c'**=$(10001110)_2=2^7+2^3+2^2+2^1$=142 => **c**=142 - 127 = **15**
mantisse : 1,10110100010000111
soit le nombre représenté est : $- 1,10110100010000111 * 2^{15} = -(2^{15} + 2^{14} + 2^{12} + 2^{11} + 2^9 + 2^5+ 2^0 + 2^{-1} + 2^{-2})$ = **-55841,75**

6. Codez l'information – 78 selon les formats suivants :
$(78)_{10}$= 64 + 8 + 4 + 2 = $2^6 + 2^3 +2^2 +2^1$ = **$(1001110)_2$**

- valeur signée sur 8 bits : **1**100 1110
- complément à 2 sur 8 bits : cplt2$(0100\ 1110)_2$ = 10110010
- IEEE 754 simple précision :

$(78)_{10}= (1001110)_2= (1,001110)_2*2^{\mathbf{6}}$
signe =1 (négatif) **c'**=6+127=133= 128+4+1=$(10000101)_2$

=> (1 10000101 00111000000000000000000)$_2$ soit **$(C29C0000)_{16}$**

Exercice 2. Carry et Overflow

a)

127 + 2 : 01111111 + 00000010 = 10000001 : overflow
127 - 2 : 01111111 + 11111110 = **1** 01111101 : pas d overflow mais carry
-127 - 2 : 10000001 + 11111110 = **1** 01111111 : carry et overflow
-127 + 2 : 10000001 + 00000010 = 10000011 : pas d'overflow

b)

A1 $a_{n-1}\ a_{n-2}$ .....$a_1\ a_0$
\+ +
B1 $b_{n-1}\ b_{n-2}$ .....$b_1\ b_0$

C1 $c_{n-1}\ c_{n-2}$ .....$c_1\ c_0$
$r_{n-1}\ r_{n-2}$ .. ..$r_1\ r_0$ retenue propagée

Il se produit un overflow seulement lorsqu'on additionne deux nombres de même signe, soit
- deux nombres positifs, alors on a $a_{n-1} = b_{n-1} = 0$
- deux nombres négatifs, alors on a $a_{n-1} = b_{n-1} = 1$

Etudions ces deux cas :

Cas 1 : deux nombres positifs, alors on a $a_{n-1} = b_{n-1} = 0$
Il se produit un overflow si le résultat de l'addition est négatif, c'est-à-dire si on a $c_{n-1} = 1$
On a : $a_{n-1} = b_{n-1} = 0 \Rightarrow a_{n-1} + b_{n-1} = 0 \Rightarrow r_{n-1} = 0$
d'où c_{n-1} ne peut être égal à 1 que si $r_{n-2} = 1$
on peut noter que $\mathbf{r_{n-2} \neq r_{n-1}}$

Cas 2 : deux nombres négatifs, alors on a $a_{n-1} = b_{n-1} = 1$
Il se produit un overflow si le résultat de l'addition est positif, c'est-à-dire si on a $c_{n-1} = 0$
On a : $a_{n-1} = b_{n-1} = 1 \Rightarrow r_{n-1} = 1$
d'où c_{n-1} ne peut être égal à 0 que si $r_{n-2} = 0$
on peut noter que $\mathbf{r_{n-2} \neq r_{n-1}}$

A présent, observons quelle est la relation existante entre r_{n-2} et r_{n-1} si les deux nombres à additionner sont de signes différents, c'est-à-dire si on a $a_{n-1} \neq b_{n-1}$
Si $a_{n-1} \neq b_{n-1}$, alors $a_{n-1} + b_{n-1} = 1$.
Si $r_{n-2} = 1$, $c_{n-1} = 0$ et $r_{n-1} = 1$
Si $r_{n-2} = 0$, $c_{n-1} = 1$ et $r_{n-1} = 0$
on peut noter que $\mathbf{r_{n-2} = r_{n-1}}$

Conclusion : l'Unité Arithmétique et Logique peut détecter un overflow en effectuant un test de comparaison entre r_{n-2} et r_{n-1}. Il y a overflow si $\mathbf{r_{n-2} \neq r_{n-1}}$

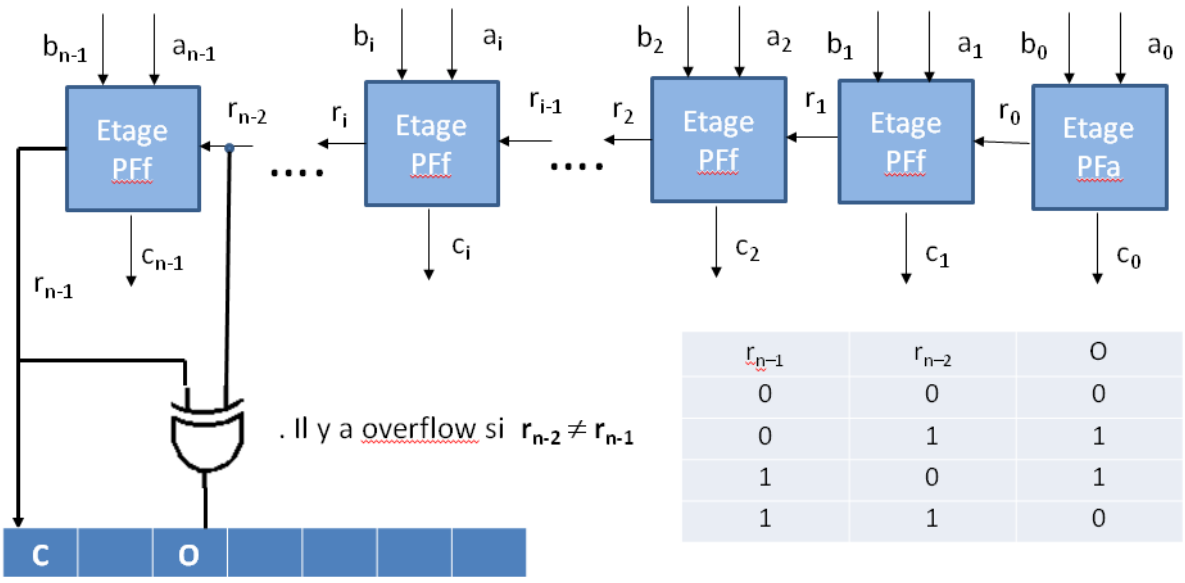
Exercice 3. Circuit logique

Il y a overflow si $\mathbf{r_{n-2} \neq r_{n-1}}$.
La table de vérité correspondante est :

r_{n-2}	r_{n-1}	overflow
0	0	0
0	1	1
1	0	1
1	1	0

Ce qui correspond à un circuit XOR.

Le schéma ci-dessous montre cette détection sur le circuit additionneur. Ce circuit additionneur est composé de n étages qui permettent d'additionner les n bits des nombres A et B (cf cours).



Indicateurs de l'UAL

r_{n-1}	r_{n-2}	O
0	0	0
0	1	1
1	0	1
1	1	0

Table de vérité pour l'indicateur d'overflow O
Ce qui correspond à une porte XOR.
Donc $O = r_{n-2} \oplus r_{n-1}$.