T.D. 1 – Corrigé
Systèmes de numération entière

Exercice 1

Représentez le nombre 248_{10} dans les bases 2, 3, 8, 9 et 16.
(Utilisez la technique des divisions successives pour les bases 2, 3 et 16.)

- **Base 2**

```
248 / 2 = 124 reste 0
124 / 2 =  62 reste 0
 62 / 2 =  31 reste 0
 31 / 2 =  15 reste 1
 15 / 2 =   7 reste 1
  7 / 2 =   3 reste 1
  3 / 2 =   1 reste 1
  1 / 2 =   0 reste 1
```
⇨ $\mathbf{248_{10} = 11111000_2}$

- **Base 3**

```
248 / 3 = 82 reste 2
 82 / 3 = 27 reste 1
 27 / 3 =  9 reste 0
  9 / 3 =  3 reste 0
  3 / 3 =  1 reste 0
  1 / 3 =  0 reste 1
```
⇨ $\mathbf{248_{10} = 100012_3}$

- **Base 8**

On peut s'aider de la représentation binaire en regroupant les chiffres par paquets de trois ($2^3 = 8$).

$248_{10} = 11\ 111\ 000_2$ ⇨ $\mathbf{248_{10} = 370_8}$

- **Base 9**

On peut s'aider de la représentation en base 3 en regroupant les chiffres par paquets de deux ($3^2 = 9$).

$248_{10} = 10\ 00\ 12_3$ ⇨ $\mathbf{248_{10} = 305_9}$

- **Base 16**

```
248 / 16 = 15 reste 8
 15 / 16 =  0 reste 15
```
⇨ $\mathbf{248_{10} = F8_{16}}$

Exercice 2

Représentez les nombres 1312_5, 1312_8, $2FA8_{16}$ en base 10.

- $\mathbf{1312_5} = 1.5^3 + 3.5^2 + 1.5^1 + 2.5^0 = \mathbf{207_{10}}$
- $\mathbf{1312_8} = 1.8^3 + 3.8^2 + 1.8^1 + 2.8^0 = \mathbf{714_{10}}$
- $\mathbf{2FA8_{16}} = 2.16^3 + 15.16^2 + 10.16^1 + 8.16^0 = \mathbf{12200_{10}}$

Exercice 3

Représentez les nombres 28_{10}, 129_{10}, 147_{10}, 255_{10} sous leur forme binaire par une autre méthode que les divisions successives.

On écrit la valeur des différents poids binaires puis, en commençant par le poids le plus fort, on positionne les bits à 0 ou à 1 en fonction de la somme de leur poids.

	128	64	32	16	8	4	2	1
$\mathbf{28_{10}} \rightarrow$	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**
$\mathbf{129_{10}} \rightarrow$	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
$\mathbf{147_{10}} \rightarrow$	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**1**
$\mathbf{255_{10}} \rightarrow$	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Exercice 4

1. Les nombres 11000010_2, 10010100_2, 11101111_2, 10000011_2, 10101000_2 sont-ils pairs ou impairs ?

   Les nombres pairs se terminent par au moins un zéro :
   $\mathbf{11000010_2}$, $\mathbf{10010100_2}$, $\mathbf{10101000_2}$

2. Lesquels sont divisibles par 4, 8 ou 16 ?

   - Les nombres divisibles par 4 se terminent par au moins deux zéros :
     $\mathbf{10010100_2}$, $\mathbf{10101000_2}$

   - Les nombres divisibles par 8 se terminent par au moins trois zéros :
     $\mathbf{10101000_2}$

   - Les nombres divisibles par 16 se terminent par au moins quatre zéros :
     Aucun nombre.

3. Donnez le quotient et le reste d'une division entière par 2, 4 et 8 de ces nombres.

	11000010		10010100		11101111		10000011		10101000	
	quotient	**reste**	**quotient**	**reste**	**quotient**	**reste**	**quotient**	**reste**	**quotient**	**reste**
/2	1100001	0	1001010	0	1110111	1	1000001	1	1010100	0
/4	110000	10	100101	00	111011	11	100000	11	101010	00
/8	11000	010	10010	100	11101	111	10000	011	10101	000

4. En généralisant, que suffit-il de faire pour obtenir le quotient et le reste d'une division entière d'un nombre binaire par 2^n ?

- Pour le quotient : il faut réaliser un **décalage de n bits vers la droite** du nombre.
- Pour le reste : il faut réaliser un **ET logique de 2^n-1** avec le nombre.

Les décalages et autres opérations logiques sont nettement plus rapides à réaliser pour un microprocesseur que l'opération de division.

Exercice 5

1. Si l'on désire multiplier un nombre binaire quelconque par 2 ou une puissance de 2, quelle autre opération peut-on réaliser pour éviter la multiplication ?

Un décalage logique d'un seul bit vers la gauche est équivalent à une multiplication par 2. Ainsi, un décalage logique de **n** bits vers la gauche est équivalent à une multiplication par 2^n.

2. Multipliez le nombre binaire 10001001_2 par 3 et par 10 en utilisant la technique traditionnelle de la multiplication.

- **Multiplication par 3**

$$
\begin{array}{r}
10001001_2 \\
\times \underline{\quad\quad 11_2} \\
10001001_2 \\
+\underline{100010010_2} \\
110011011_2
\end{array}
$$

- **Multiplication par 10**

$$
\begin{array}{r}
10001001_2 \\
\times \underline{\quad\quad 1010_2} \\
100010010_2 \\
+\underline{10001001000_2} \\
10101011010_2
\end{array}
$$

3. Si l'on désire multiplier un nombre binaire quelconque par 3 ou par 10, quelle méthode peut-on utiliser pour éviter la multiplication ?

   - **`3n = 2n + n`**
     Sous cette forme, il apparaît une multiplication par 2 (équivalente à un décalage d'un bit vers la gauche) et une addition.

   - **`10n = 8n + 2n`**
     Sous cette forme, il apparaît une multiplication par 8 (équivalente à un décalage de 3 bits vers la gauche), une multiplication par 2 (équivalente à un décalage d'un bit vers la gauche), et une addition.

   Si le multiplicateur est connu, on peut le décomposer de sorte à n'avoir comme opérations que des décalages et des additions. Ces dernières sont nettement plus rapides à réaliser pour un microprocesseur que la multiplication.

Exercice 6

Donnez les valeurs décimales, minimales et maximales, que peuvent prendre des nombres signés et non signés codés sur 4, 8, 16, 32 et n bits.

Bits	Non Signés	Signés
4	0 → 15	-8 → 7
8	0 → 255	-128 → 127
16	0 → 65535	-32768 → 32767
32	0 → $2^{32} - 1$	-2^{31} → $2^{31} - 1$
n	0 → $2^{n} - 1$	-2^{n-1} → $2^{n-1} - 1$

Exercice 7

1. Combien faut-il de bits, au minimum, pour coder les nombres non signés 48965_{10} et 9965245_{10} ?

   - **<u>48965</u>**
     À partir du tableau de l'exercice 6, on en déduit que la plus grande valeur d'un nombre non signé codé sur **n** bits (2^n - 1) doit être supérieure ou égale à 48965.

     $2^n - 1 \geqslant 48965$

     $2^n \geqslant 48966$

     $\ln(2^n) \geqslant \ln(48966)$

     $n \cdot \ln(2) \geqslant \ln(48966)$

     $n \geqslant \dfrac{\ln(48966)}{\ln(2)}$

     $n \geqslant 15{,}58$ ⇨ $\mathbf{n_{min} = 16}$

- **<u>9965245</u>**
  En utilisant le même raisonnement que précédemment, on obtient :
  $2^n - 1 \geqslant 9965245$
  $2^n \geqslant 9965246$
  $\ln(2^n) \geqslant \ln(9965246)$
  $n \cdot \ln(2) \geqslant \ln(9965246)$
  $n \geqslant \frac{\ln(9965246)}{\ln(2)}$
  $n \geqslant 23{,}25$ ⇨ $\mathbf{n_{min} = 24}$

2. Combien faut-il de bits, au minimum, pour coder les nombres signés -5_{10} et 28_{10} ?

- **<u>-5</u>**
  À partir du tableau de l'exercice 6, on en déduit que la plus petite valeur d'un nombre signé codé sur **n** bits (-2^{n-1}) doit être inférieure ou égale à -5.
  $-2^{n-1} \leqslant -5$
  $2^{n-1} \geqslant 5$
  $\ln(2^{n-1}) \geqslant \ln(5)$
  $(n-1) \cdot \ln(2) \geqslant \ln(5)$
  $n-1 \geqslant \frac{\ln(5)}{\ln(2)}$
  $n \geqslant \frac{\ln(5)}{\ln(2)} + 1$
  $n \geqslant 3{,}33$ ⇨ $\mathbf{n_{min} = 4}$

- **<u>+28</u>**
  À partir du tableau de l'exercice 6, on en déduit que la plus grande valeur d'un nombre signé codé sur **n** bits ($2^{n-1} - 1$) doit être supérieure ou égale à 28.
  $2^{n-1} - 1 \geqslant 28$
  $2^{n-1} \geqslant 29$
  $\ln(2^{n-1}) \geqslant \ln(29)$
  $(n-1) \cdot \ln(2) \geqslant \ln(29)$
  $n-1 \geqslant \frac{\ln(29)}{\ln(2)}$
  $n \geqslant \frac{\ln(29)}{\ln(2)} + 1$
  $n \geqslant 5{,}86$ ⇨ $\mathbf{n_{min} = 6}$

Exercice 8

1. Représentez sous forme décimale le nombre 11111111_2 codé sur 8 bits signés.

   Le bit de poids fort vaut 1 : le nombre est négatif.
   On effectue son complément à 2 :
   $(11111111_2)_{C2} = 00000000_2 + 1_2 = 1_2$
   La représentation décimale est donc de -1_{10}.

2. Représentez sous forme décimale le nombre 11111111_2 codé sur 16 bits signés.

   Le bit de poids fort vaut 0 ($\underline{\mathbf{0}}000000011111111_2$) : le nombre est positif.
   On effectue une simple conversion binaire-décimal :
   $11111111_2 = 128_{10}+64_{10}+32_{10}+16_{10}+8_{10}+4_{10}+2_{10}+1_{10} = 255_{10}$
   La représentation décimale est donc de $+255_{10}$.

3. Représentez les opposés binaires et hexadécimaux, sur 8 bits signés, du nombre 80_{10}.

   On convertit sa valeur absolue en binaire : $80_{10} = 01010000_2$
   On effectue son complément à 2 :
   $(01010000_2)_{C2} = 10101111_2 + 1_2 = 10110000_2$
   Ce qui donne : **10110000_2 en binaire.**
   $B0_{16}$ en hexadécimale.

4. Représentez les opposés binaires et hexadécimaux, sur 16 bits signés, du nombre 80_{10}.

   Une simple extension de signe suffit pour passer de 8 bits à 16 bits signés.
   Ce qui donne : **$\underline{11111111}10110000_2$ en binaire.**
   $\underline{FF}B0_{16}$ en hexadécimale.