V. La numérisation

1. Principe des systèmes de numération

Quel que soit le système de numération (base B), tout nombre X se représente sous la forme :

$$X = (a_n , a_{n-1}, \dots , a_0)_{Base}$$

Et peut se calculer dans la base décimale comme : $X = \sum_{i=0}^{n} a_i B^i$ où :

- $\mathbf{B^n, B^{n-1},..., B^0}$ sont les poids.
  - $\boldsymbol{B = 2}$ **: Binaire (base 2)**
  - $\boldsymbol{B = 8}$: **Octave (base 8)**
  - $\boldsymbol{B = 10}$: **Décimale (base 10)**
  - $\boldsymbol{B = 16}$: **hexadécimale (base 16)**
- $\boldsymbol{a_n, a_{n-1},..., a_0}$ sont les coefficients ou les chiffres.
- ***n , n-1, ..., 0*** sont les rangs

Ce principe d'écriture des nombres est appelé "numération positionnelle", dans lequel c'est la position du symbole graphique qui donne sa valeur. Les systèmes de numération ainsi constitués sont dits pondérés.

En base B, avec n rangs, on a B^n combinaisons. On peut donc compter de 0 à B^{n-1}

2. Numération décimale

Il s'agit d'une base (10) contenant 10 symboles : 0, 1, …, 9

Exemple : $\mathbf{7239 = 7.10^3 + 2.10^2 + 3.10^1 + 9.10^0}$ ← Rang

Chiffre ↑ Poids ↑

3. Numération binaire

Il s'agit d'une base (2) contenant 2 symboles : 0 et 1

Exemple : $\mathbf{1011 = 7.2^3 + 2.2^2 + 3.2^1 + 9.2^0}$

Avec

1	0	1	1	0	0	1

MSB ↑ **LSB** ↑

MSB : bit le plus significatif (Most Significant Bit)

LSB : bit le moins significatif **(**Low Significant Bit)

4. Numération hexadécimale

Ce système de numération est très utilisé dans les systèmes pour contracter la lecture et l'écriture des données binaires.

Cette base (16) contient 16 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Exemple : $\boldsymbol{A34F = A.16^3 + 2.16^2 + 3.16^1 + 9.16^0}$

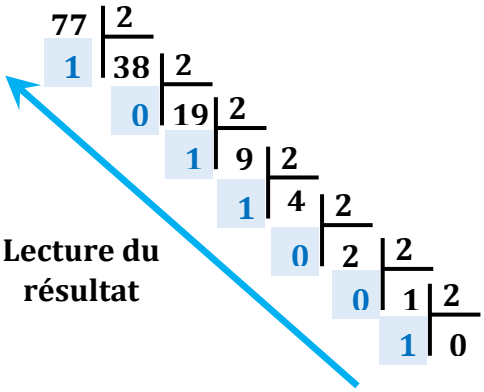
5. Changement de base

- **Décimale/Binaire**

La conversion de la partie entière d'un nombre décimal en nombre binaire consiste en des divisions successives par 2 jusqu'à ce qu'un 0 soit obtenu comme quotient avec un reste de 1. Les restes correspondent aux bits consécutifs dans l'ordre inverse.

Exemple :

$(77)_{10} = (1001101)_2$

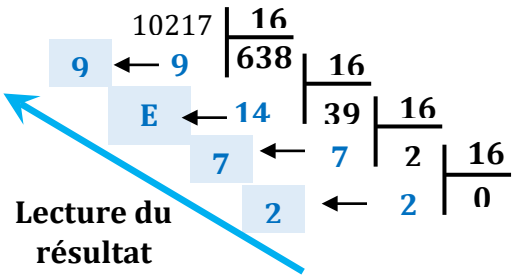


- **Décimale/hexadécimale**

La conversion de la partie entière d'un nombre décimal en nombre hexadécimale consiste en des divisions successives par 16 jusqu'à ce qu'un 0 soit obtenu comme quotient avec un reste de 1. Les restes correspondent aux digits consécutifs dans l'ordre inverse.

Exemple :

$(10217)_{10} = (27E9)_{16}$



- **Binaire ⇔ hexadécimale**

On regroupe les bits 4 par 4 (en quartets). Chaque quartet peut alors être converti directement en hexadécimal (max. : 1111 = $ F).

Ex : 1111 1011 = $FB

Chaque symbole (chiffre ou lettre) en hexadécimal est converti dans le quartet binaire correspondant.

Exemples : $\$F0A8 = (1111\ 0000\ 1010\ 1000)_2$

VI. Codage des systèmes numériques

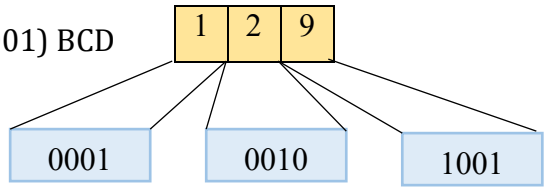
1. Décimal codé en binaire (BCD)

Ce codage permet une lecture décimale directe à partir d'un nombre binaire, il est surtout utilisé pour l'affichage.

Décimal	0	1	2	3	4	5	6	7	8	9
Binaire	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Exemple :

129 = (0001 0010 1001) BCD



2. Code Gray ou binaire réfléchi

Le code Gray ou "binaire réfléchi" permet de coder une valeur numérique en cours d'évolution en une suite de configurations binaires se différenciant l'une de l'autre par le changement d'état d'un et d'un seul bit.

Décimal	0	1	2	3	4	5	6	7	8	9
Binaire	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
GRAY	**0000**	**0001**	**0011**	**0010**	**0110**	**0111**	**0101**	**0100**	**1100**	**1101**

Parmi les applications du code de gray on cite :

- Les capteurs angulaires ou de positionnement.
- Le tableau de Karnaugh (conception des circuits logiques).

VII. Représentation des nombres dans les systèmes numériques

1. Représentation des nombres entiers relatifs - nombre signé

La notation est utilisée sur des écritures de nombres de longueur donnée (nombres écrits couramment sur 8, 16, 32 ou 64 bits). Dans une telle écriture, on utilise le MSB (bit le plus à gauche) du nombre pour contenir la représentation de son signe (positif ou négatif, le zéro étant considéré comme positif).

Pour le MSB 0 : indique une **valeur positive**

1 : indique une **valeur négative**

Il faut donc connaître dès le départ le format de codage (Registre n bits), et si les nombres sont signés ou non.

Exemple :

0	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**+ 6**
1	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**- 6**

2. Représentation des nombres réels

2.1. Représentation en virgule fixe

On rappelle qu'un nombre réel se compose de deux parties distinctes :

- une partie entière, située à gauche de la virgule ;
- une partie décimale, située après celle-ci.

Ce codage peut s'écrire sous la forme d'un doublet :

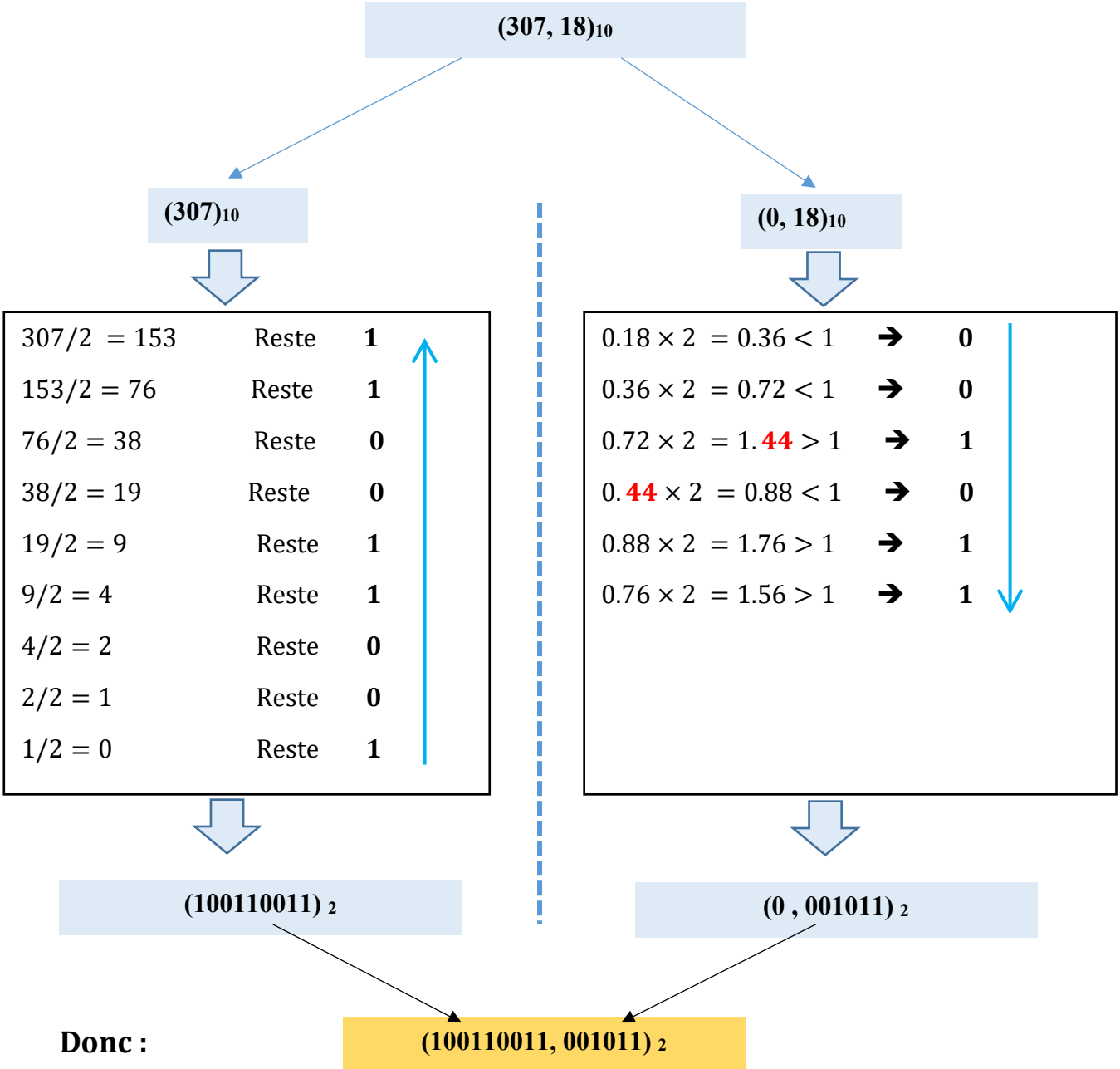
[Partie entière en binaire, partie décimale en binaire]

Remarque :

Du fait que le processeur ne sait pas représenter le symbole "virgule", il faut, pour que les réels soient correctement interprétés, que l'emplacement de la virgule soit défini une fois pour toute et ne varie plus, quels que soient les nombres à représenter.

- **Méthode de conversion**

La conversion en binaire de $(307,18)_{10}$



Donc :

Conclusion : Cette méthode possède l'inconvénient : le nombre de chiffre après la virgule n'est pas limité

2.2. Représentation en virgule flottante

Pour pallier les inconvénients de la représentation en virgule fixe, on utilise une autre représentation similaire à la notation scientifique des calculatrices, sauf qu'elle est en base 2 et non en base 10.

La virgule flottante est une méthode d'écriture de nombres réels fréquemment utilisée dans les ordinateurs. Elle consiste à représenter un nombre par **un signe s** (égal à -1 ou 1), une **mantisse m** (aussi appelée significande) et **un exposant e** (entier relatif, généralement borné).

Un nombre est représenté en virgule flottante s'écrit :

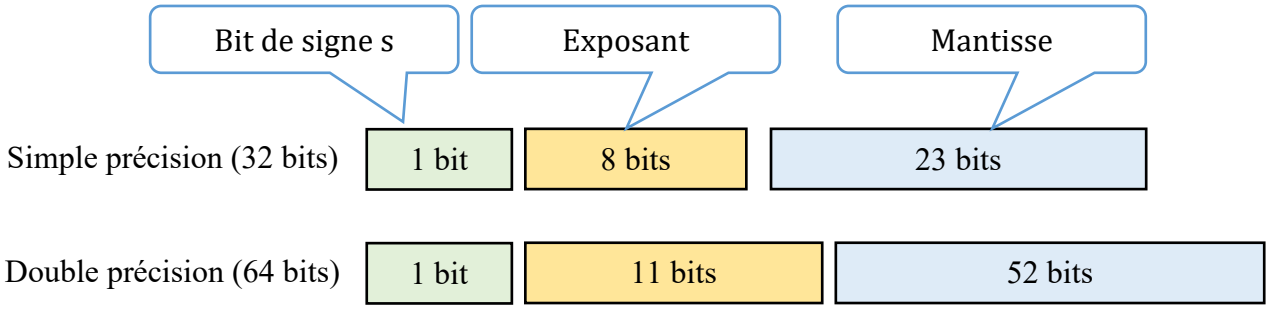
$$X = s.m.2^e$$

s : bit de signe (**s=0** : nombre positif) et (**s=1** : nombre positif).

m : mantisse

e : exposant

La norme **IEEE 754** distingue la représentation binaire simple précision sur **32 bits**, et la double précision sur **64 bits**.



- La mantisse est écrite sous la forme : 1, M où M est la mantisse suivant la norme **IEEE 754**.
- Un nombre flottant simple précision est stocké dans un mot de 32 bits : 1 bit de signe, 8 bits pour l'exposant et 23 pour la mantisse.

L'exposant est décalé de $\mathbf{2^{8-1} - 1 = 127}$

Exemple :

Traduire en binaire format floutant simple précision (32 bits) le nombre $X = -6{,}625$?

Le format attendu (simple précision (32 bits)) : $(-1)^{signe}.Mantisse.2^{(E+127)}$.

- **Etape 1 : le signe**

  Le nombre X est négative :

$$\boldsymbol{signe = 1}$$

- **Etape 2 : le mantisse M**
  - Convertir la valeur absolue en binaire $|X| = 6,625$: $(6,625)_{10} = (110,1010)_2$
  - Former la mantisse : $1,mantisse$

    On a : $(110,1010)_2 = 1,101010 . 2^2 = 1,mantisse . 2^E$

    La mantisse est codée sur 23 bits, alors :

$$Mantisse = 10101000000000000000000$$

- **Etape 3 : l'exposant**

  La valeur à stocker dans la partie exposant est exprimer par : $n = 127 + E$

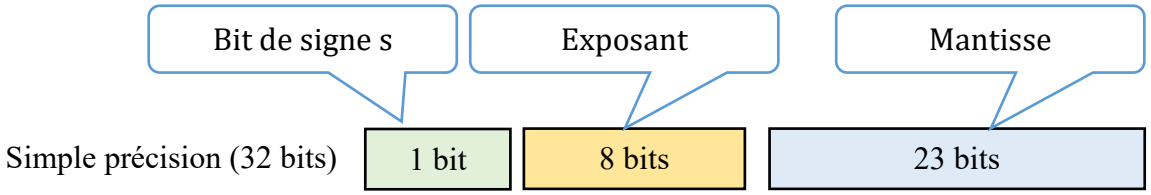
  On a $(110,1010)_2 = 1,101010 . 2^2 = 1,mantisse . 2^E$ ➔ $E = 2$

  D'où : l'exposant : $n = 127 + E = 127 + 2 = 129$

  Cette valeur doit être convertie en binaire (codée sur 8 bits voir en haut) :

$$n = 10000001$$

Donc la valeur convertie est :



1	1	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C				**0**				**D**				**4**				**0**				**0**				**0**				**0**			

Soit en hexadécimal : X= (C0D40000) h