Chapitre 3: REPRESENTATION DES INFORMATIONS DE BASE

1. Systèmes de numération

1.1. Le système décimal: système de base 10.

- *Les nombres entiers:*

N_1 : $842 = 8x10^2 + 4x10^1 + 2x10^0$

N_2 : $22856 = 2x10^4 + 2x10^3 + 8x10^2 + 5x10^1 + 6x10^0$

$N_{(10)} = a_nx10^n + a_{n-1}x10^{n-1} + \ldots\ldots + a_2x10^2 + a_1x10^1 + a_0$ (forme explicite)

$= a_na_{n-1}\ldots\ldots a_2a_1a_0$ (forme implicite) $\}$ $(0 \le a_i \le 9)$

Expression d'un nombre N dans une base B :

$N_{(b)} = a_nxB^n + a_{n-1}xB^{n-1} + \ldots\ldots + a_2xB^2 + a_1xB^1 + a_0$ ou $N_{(b)} = a_na_{n-1}\ldots\ldots a_2a_1a_0$ / $(0 \le a_i \le B-1)$

- *Les nombres fractionnaires:*

$687.48_{(10)} = 6x10^2 + 8x10^1 + 7x10^0 + 4x10^{-1} + 8x10^{-2}$

$0.485_{(10)} = 4x10^{-1} + 8x10^{-2} + 5x10^{-3}$

$N_{(b)} = a_nxB^n + a_{n-1}xB^{n-1} + \ldots\ldots + a_1xB^1 + a_0 + a_{-1}x10^{-1} + a_{-2}x10^{-2} + a_{-3}x10^{-3} + \ldots\ldots$ / $(0 \le a_i \le B-1)$

1.2. Le système octal: système de base 8.

- *Les nombres entiers:*

$653_{(8)} = 6x8^2 + 5x8^1 + 3x8^0 = 6x64 + 5x8 + 3 = 427_{(10)}$

$N_{(8)} = a_nx8^n + a_{n-1}x8^{n-1} + \ldots\ldots + a_2x8^2 + a_1x8^1 + a_0$ (forme explicite)

$= a_na_{n-1}\ldots\ldots a_2a_1a_0\ _{(8)}$ (forme implicite) $\}$ $(0 \le a_i \le 7)$

- *Les nombres fractionnaires:*

$653.12_{(8)} = 6x8^2 + 5x8^1 + 3x8^0 + 1x8^{-1} + 2x8^{-2}$

$= 6x64 + 5x8 + 3 + 1/8 + 2/64 = 427.15_{(10)}$

$N_{(8)} = a_nx8^n + a_{n-1}x8^{n-1} + \ldots\ldots + a_1x8^1 + a_0 + a_{-1}x8^{-1} + a_{-2}x8^{-2} + a_{-3}x8^{-3} + \ldots\ldots$ avec $(0 \le a_i \le 7)$

1.3. Le système hexadécimal: système de base 16**.**

- *Les nombres entiers:*

$N_{(16)} = a_n x16^n + a_{n-1} x16^{n-1} + \ldots\ldots + a_2 x16^2 + a_1 x16^1 + a_0$ avec $a_i \in \{0,1,\ldots,9,A,B,C,D,E,F\}$

$= a_n a_{n-1} \ldots\ldots a_2 a_1 a_0\ (16)$

$890_{(16)} = 8x16^2 + 9x16^1 + 0x16^0 = 8x256 + 9x16 + 0 = 2192_{(10)}$

$1FA_{(16)} = 1x16^2 + 15x16^1 + 10x16^0 = 1x256 + 15x16 + 10 = 506_{(10)}$

- *Les nombres fractionnaires:*

$N_{(16)} = a_n x16^n + a_{n-1} x16^{n-1} + \ldots\ldots + a_1 x16^1 + a_0 + a_{-1} x16^{-1} + a_{-2} x16^{-2} + a_{-3} x16^{-3} + \ldots\ldots$

$1.1F_{(16)} = 1x16^0 + 1x16^{-1} + 15x16^{-2} = 1.12_{(10)}$

1.4. Le système binaire: système de base 2.

- *Les nombres entiers:*

$N_{(2)} = a_n x2^n + a_{n-1} x2^{n-1} + \ldots\ldots + a_2 x2^2 + a_1 x2^1 + a_0$ avec $a_i \in \{0, 1\}$

$= a_n a_{n-1} \ldots\ldots a_2 a_1 a_0\ (2)$

$99_{(10)} = 64 + 32 + 2 + 1 = 1x2^6 + 1x2^5 + 1x2^1 + 1x2^0 = 1100011_{(2)}$

- *Les nombres fractionnaires:*

$N_{(2)} = a_n x2^n + a_{n-1} x2^{n-1} + \ldots\ldots + a_1 x2^1 + a_0 + a_{-1} x2^{-1} + a_{-2} x2^{-2} + a_{-3} x2^{-3} + \ldots\ldots$

$27.25_{(10)} = 16 + 8 + 2 + 1 + 1/4 = 1x2^4 + 1x2^3 + 1x2^1 + 1x2^0 + 1x2^{-2} = 11011.01_{(2)}$

2. Le transcodage: On appelle transcodage le passage d'un système de base B_1 vers un système de base B_2.

Définition 1 : On donne le nom de **CODAGE** à l'opération qui consiste à représenter dans un certain code (C), les nombres donnés dans un système de numération décimal.

Définition 2 : On donne le nom de **DECODAGE** à l'opération qui consiste à représenter dans le système de numération décimal les nombres donnés dans un certain code (C).

Définition 3 : Soient deux codes Cl et C2, on appelle **TRANSCODAGE** de Cl à C2, l'opération qui consiste à déterminer la représentation dans le code (C2) de tout nombre dont on connaît la représentation dans le code Cl.

2.1. Passage du système décimal au système binaire

• *Les nombres entiers:*

$23_{(10)} = 11x2+1$
$= (5x2 + 1)x2 + 1$
$=5x2^2+1x2+1$
$=(2x2+1)x2^2+1x2+1$
$=1x2^4+1x2^2+1x2+1$
$=1x2^4+0x2^3+1x2^2+1x2^1+1x2^0$
$= 10111_{(2)}$
$23_{(10)} = 10111_{(2)}$

23	2
1	11
	2
1	5
	2
1	2
	2
0	1
	2
1	0

$23_{(10)}$	2^4	2^3	2^2	2^1	2^0
23	1				
7			1		
3				1	
1					1
0	**1**	**0**	**1**	**1**	**1**

• *Les nombres fractionnaires:*

$0.43_{(10)} = a_{-1}.2^{-1} + a_{-2}.2^{-2} + a_{-3}.2^{-3} + a_{-4}.2^{-4} + \ldots$

$0.43x2 = a_{-1} + a_{-2}.2^{-1} + a_{-3}.2^{-2} + a_{-4}.2^{-3} + \ldots$ $\quad 0.43x2 = 0.86 < 1 \Rightarrow a_{-1} = 0$

$0.86x2 = a_{-2} + a_{-3}.2^{-1} + a_{-4}.2^{-2} + a_{-5}.2^{-3} \ldots$ $\quad 0.86x2 = 1.72 > 1 \Rightarrow a_{-2} = 1$

$0.72x2 = a_{-3} + a_{-4}.2^{-1} + a_{-5}.2^{-2} \ldots$ $\quad 0.72x2 = 1.44 > 1 \Rightarrow a_{-3} = 1$

$0.44x2 = a_{-4} + a_{-5}.2^{-1} + a_{-6}.2^{-2} \ldots$ $\quad 0.44x2 = 0.88 < 0 \Rightarrow a_{-4} = 0$

$0.88x2 = a_{-5} + a_{-6}.2^{-1} + a_{-7}.2^{-2} \ldots$ $\quad 0.88x2 = 1.76 > 1 \Rightarrow a_{-5} = 1$

$0.76x2 = a_{-6} + a_{-7}.2^{-1} \ldots$ $\quad 0.76x2 = 1.52 > 1 \Rightarrow a_{-6} = 1$

$\Rightarrow 0.43_{(10)} = 0.011011_{(2)}$

2.2. Passage du système décimal a un système de base B

• *Les nombres entiers:*

1) $\mathbf{N_{(10)}} = K_1 . B + a_0$ si $K_1 < B$ arrêter
2) $\mathbf{K_1} = K_2 . B + a_1$ si $K_2 < B$ arrêter
3) $\mathbf{K_2} = K_3 . B + a_2$ si $K_3 < B$ arrêter sinon continuer le processus.

• *Les nombres fractionnaires:*

1) $N_{(10)} < 1 \Rightarrow P_1 = N . B \quad \Rightarrow a_{-1}$ =partie entière de P_1
2) $P_2 = (P_1 - a_{-1}) . B \quad \Rightarrow a_{-2}$ =partie entière de P_2
3) $P_3 = (P_2 - a_{-2}) . B \quad \Rightarrow a_{-3}$=partie entière de P_3

Le processus jusqu'à atteindre la précision souhaitée.

2.3. Passage du système décimal au système octal

• *Les nombres entiers:*

$458_{(10)} = 57x8 + 2 = (7x8 + 1) x8 + 2 = 7x8^2 + 1x8 + 2 = 712_{(8)}$

• *Les nombres fractionnaires:*

$0.49_{(10)} = 0.3727_{(8)}$

$0.49x8 = 3.98 \quad \Rightarrow a_{-1} = 3$

$0.98x8 = 7.36 \quad \Rightarrow a_{-2} = 7$

$0.36x8 = 2.88 \quad \Rightarrow a_{-3} = 2$

$0.88x8 = 7.04 \quad \Rightarrow a_{-4} = 7$

2.4. Passage du système décimal au système hexadécimal

• *Les nombres entiers:*

$1254_{(10)} = 78x16 + 6 = (4x16 + 14) x16 +62 = 4x16^2 + 14x16 + 2 = 4E6_{(16)}$

• *Les nombres fractionnaires:*

$0.51_{(10)} = 0.828F5_{(16)}$

$0.51x16 = 8.16 \quad \Rightarrow a_{-1} = 8$

$0.16x16 = 2.56 \quad \Rightarrow a_{-2} = 2$

$0.56x16 = 8.96 \quad \Rightarrow a_{-3} = 8$

$0.96 \times 16 = 15.36 \quad \Rightarrow a_{-4} = F$

$0.36 \times 16 = 5.76 \quad \Rightarrow a_{-5} = 5$

Remarque: Pour passer du système octal ou hexadécimal au système binaire, il suffit de développer chaque coefficient du nombre sur trois ou quatre positions. Pour l'opération inverse, il suffit de regrouper par trois ou quatre chiffres binaires.

3. Représentation des informations

Les informations traitées par l'ordinateur de différents types, mais elles sont toujours représentées sous forme binaire. Le codage d'une information consiste a établir une correspondance entre la représentation externe de l'information et sa représentation interne qui est une suite de bits.

3.1. Représentation des nombres entiers

3.1.1. Représentation en binaire pur (Le codage binaire naturel) : On appelle ainsi la représentation des nombres dans le système de numération binaire, pondérée de telle façon que le bit a_j soit affecté du poids 2^j; c'est la représentation de la valeur d'un nombre dans le système binaire sur n bits.

Exemple 01: Avec 4 bits on représente les nombres de 0 à $2^4-1 \quad \Rightarrow 0 \ldots 15$

10^1	10^0	2^3	2^2	2^1	2^0
0	0	0	0	0	0
0	1	0	0	0	1
0	2	0	0	1	0
0	3	0	0	1	1
0	4	0	1	0	0
0	5	0	1	0	1
0	6	0	1	1	0
0	7	0	1	1	1
0	8	1	0	0	0
0	9	1	0	0	1
1	0	1	0	1	0
1	1	1	0	1	1
1	2	1	1	0	0
1	3	1	1	0	1
1	4	1	1	1	0
1	5	1	1	1	1

Exemple 02:

Avec 8 bits on représente les nombres de 0 à 2^8-1 $\Rightarrow$ 0 . . . 255.
Avec 16 bits on représente les nombres de 0 à 2^{16}-1 $\Rightarrow$ 0 . . . 65535.
Avec n bits on représente les nombres de 0 à 2^n-1.

3.1.2. Le codage binaire réfléchi(Code Gray): La règle de construction est la suivante: On écrit les 2n combinaisons qui représentent les nombres entiers de 0 à 2^{n-1}. Pour obtenir les combinaisons suivantes, de 2^n à 2^{n+1}-1, on réécrit les mêmes 2^n combinaisons dans l'ordre inverse, en ajoutant un 1 à la gauche de chacune d'elles.

Exemples :

* n = 1

0 → 0
1 → 1

11 → 2
10 → 3

* n = 2

00 → 0
01 → 1
11 → 2
10 → 3

110 → 4
111 → 5
101 → 6
100 → 7

D'où pour n = 4 nous obtenons le tableau suivant :

10^1	10^0				
0	0	0	0	0	0
0	1	0	0	0	1
0	2	0	0	1	1
0	3	0	0	1	0
0	4	0	1	1	0
0	5	0	1	1	1
0	6	0	1	0	1
0	7	0	1	0	0
0	8	1	1	0	0
0	9	1	1	0	1
1	0	1	1	1	1
1	1	1	1	1	0
1	2	1	0	1	0
1	3	1	0	1	1
1	4	1	0	0	1
1	5	1	0	0	0

Propriétés:

P1 : Le code binaire réfléchi n'est pas pondéré.

P2 : La règle de construction montre que le code est continu et cyclique :

-**Continu** : Il s'agit d'un code dont toutes les combinaisons consécutives sont deux à deux adjacentes (un seul bit change de valeur pour passer d'une combinaison à l'autre).

-**Cyclique** : Il s'agit d'un code dont la dernière combinaison est adjacente à la première.

3.1.3. Le système décimal codé binaire (DCB): Nous représentons chaque chiffre décimal au moyen d'une combinaison de chiffres binaires. Pour disposer de dix combinaisons différentes il faut au moins 4 bits. On affecte au codage de chaque chiffre décimal un codage binaire suivant la pondération 8, 4, 2, 1. Autrement dit chaque chiffre décimal est codé par son équivalent en binaire naturel.

Exemple :

N_d	=	1	9	9	3
N_{dcbn}	=	0001	1001	1001	0011

3.2. Représentation des nombres entiers signés

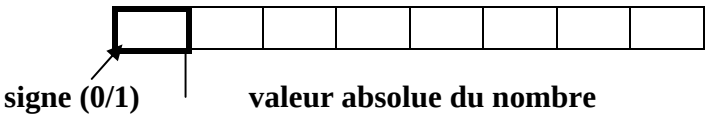
3.2.1. Représentation en binaire signé: On considère un nombre relatif nb de valeur absolue N = |nb| codé au moyen de p bits : N = |n| = a_{p-1} a_{p-2} …a_1 a_0. Pour la représentation de n on se contente de rajouter un bit supplémentaire a_p pour représenter le signe. Par convention, on représente le signe '+' par la valeur $a_p = 0$ et le signe '-' par la valeur $a_p = 1$.

Exemple 01: Dans ces conditions nous obtenons pour p = 2 :

+3	011
+2	010
+1	001
+0	000
-0	100
-1	101
-2	110
-3	111

Cette représentation est intéressante dans le cas où l'on veut réaliser des opérations de multiplication et de division. Par contre elle présente des inconvénients pour des opérations addition et de soustraction du fait qu'il existe une ambiguïté au niveau d u codage du zéro (+0 et -0).

Exemple 02: Avec 8 bits



On peut représenter les nombres de $-(2^7-1)$ à (2^7-1).

Avec n bits on peut représenter les nombres de $-(2^{n-1}-1)$ à $(2^{n-1}-1)$.

3.2.2. Représentation en complément a 1: les nombres positifs sont représentés comme dans la représentation binaire signé tandis que les négatifs sont obtenus en complémentant leurs valeurs absolus a 1.

3.2.3. Représentation en complément a 2: les nombres positifs sont représentés comme dans le représentation binaire signé tandis que les négatifs sont représentés en complément a 1 puis on rajoute 1. Ce codage permet de représenter les nombres de -2^{n-1} jusqu'à $2^{n-1} - 1$ sur n bits.

Remarque :

- On considère un nombre N = |n| codé sur p bits:

  Si $n \geq O$ alors $n = N$ sinon Si $n < 0$ alors $n = 2^{p+1} - N$

Conséquences:

1- N est codé sur p bits d'où $N \leq 2^p - 1$

or dans le cas où $n<O$ nous avons $N = 2^{p+1} - n$ d'où : $2^{p+1} - n \leq 2^p - 1$

$$2^{p+1} - 2^p + 1 \leq n$$
$$2^p (2-1) + 1 \leq n$$
$$2^p + 1 \leq n$$

Ainsi n négatif est codé sur p+1 bits et le bit de poids fort vaut 1 alors $n = 1a_{p-1}\, a_{p-2} \ldots a_1\, a_0$

Afin d'uniformiser l'écriture nous coderons également n positif sur p+1 bits, où le bit de poids fort prendra la valeur 0 :

$n>0 \longrightarrow n = 0a_{p-1}\, a_{p-2} \ldots a_1\, a_0$

$n<0 \longrightarrow n = 1\, a_{p-1}\, a_{p-2} \ldots a_1\, a_0$

2- Dans le cas où N = 0 nous pouvons écrire :

- $N = n^+ = 00 \ldots 0$
- $N = n^- = 2^{p+1} - 0 = 2^{p+1}$

Or la valeur 2^{p+1} s'exprime sur p+2 bits et par conséquent nous ne sommes plus cohérents avec la capacité que nous nous étions fixés pour exprimer les nombres. La seule représentation de la valeur zéro est 00 ... 0. Avec ce code nous n'avons plus d'ambiguïté $\Rightarrow$ Le 0 est unique.

- La somme de deux nombres est toujours vraie.
- La valeur décimale du nombre en complément a 2 sur n bits ($a_{n-1}, a_{n-2}, \ldots, a_2, a_1, a_0$) est :

  $N = -a_{n-1}.2^{n-1} + a_{n-2}.2^{n-2} + \ldots + a_2.2^2 + a_1.2^1 + a_0$

Exemple01:

p = 4

$n = +6 \Rightarrow N = |n| = 6 \longrightarrow n = 0\ 0110$

$n = -6 \Rightarrow N = |n| = 6 \longrightarrow n = 2^5 - 6 = 26_{(10)} = 1\ 1010$

$n = +8 \Rightarrow N = |n| = 8 \longrightarrow n = 0\ 1000$

$n = -8 \Rightarrow N = |n| = 8 \longrightarrow n = 2^5 - 8 = 24_{(10)} = 1\ 1000$

Tableau représentant l'ensemble des valeurs codées sur p = 3 bits

0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	O
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

Remarque: On obtient les mêmes résultats en appliquent la règle de définition qui pour exprimer le complément à 2 d'un nombre consiste à complémenter tous les bits puis de rajouter 1.

Exemple02: Pour le codage du nombre n=-6 on aura:

$n = -6 \Rightarrow N = |n| = 6 \longrightarrow n = +6 = 0\ 0110$

$n = 1\ 1001 + 1 = 1\ 1010$

3.3. Représentation des nombres réels

3.3.1.Représentation en virgule fixe: Il s'agit de la représentation par la suite de symboles suivante :

$Nb = a_n a_{n-1} \ldots\ldots a_2 a_1 a_0, a_{-1} a_{-2} a_{-3} \ldots\ldots a_{-k}$

$Nb = \Sigma\ a_i \cdot 2^i$ où a_i est la représentation de ai dans la base 2.

Exemple:

$17{,}125_{(10)} = 10001{,}001_{(2)}$

$= 10001001 \times 2^{-3}$

1	0	0	1	0	0	0	1

La position de la virgule est fixée arbitrairement a la troisième case du registre a 8 bits, la position n'est pas visible. Sachant que la case la plus a droite correspond au poids 0 ce qui est faux. Ceci conduit à supposer que le contenu du registre est multiplié par 2^{-3}. 2^{-3} représente la position de la virgule fixe, il est stocké dans un autre registre.

Exemple:

$22{,}57_{(10)} = 10110{,}1001_{(2)} = 101101001 \times 2^{-4}$

Le premier registre représente la valeur :

0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	1

Le deuxième registre représente la position de la virgule :

0	0	0	0	0	1	0	0

3.3.2. Représentation en virgule flottante: Nous avons l'habitude de représenter les nombres flottants suivant la "notation scientifique" ou encore la "représentation semi-logarithmique". Cette représentation est nécessaire pour les nombres non entiers ou bien très grands ou encore très petits. Ils sont alors exprimés sous la forme : $N = M.B^e$, où M s'appelle la mantisse, B représente la base et e l'exposant.

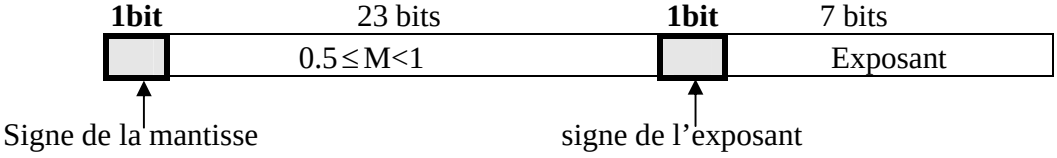
L inconvénient de cette représentation est que l'écriture d'un nombre n'est pas unique. En effet $300\ 000 = 3.\ 10^5 = 300.10^3 = \ldots$, il est donc nécessaire de normaliser l'écriture. Nous dirons que l'écriture est normalisée si : $M < 1$. Par conséquent M sera de la forme 0, ... où le premier chiffre après la virgule est forcément différent de 0. Nous écrirons alors : $0{,}3.\ 10^6$.

La transposition de cette écriture au sein d'une machine numérique revient à mettre les nombres sous la forme : $N = M.2^x$, où M s'appelle la mantisse et x représente l'exposant.

Remarque: si $0.5 \leq M < 1$ alors la mantisse est dite normalisée.

a) Exposant et mantisse en binaire signé: Cette représentation revient à affecter un bit de signe et représenter la mantisse par sa valeur absolue, la même représentation peut être adopter pour l'exposant.

32 bits $\Rightarrow$ 24 bits (pour la mantisse) et 8 bits (pour l'exposant)



Exemple:

$N = -42{,}75_{(10)} = -101010{,}11_{(2)} = -0.10101011 \times 2^6$

Signe de le mantisse = - $\Rightarrow$ 1

Signe l'exposant = + $\Rightarrow$ 0

Mantisse = 10101011000000000000000

Exposant = 0000110

$\Rightarrow$ la représentation de N est: **1**10101011000000000000000**0**0000110 = $32540000006_{(8)}$

b) Exposant biaisé : Parmi les différentes conventions de représentation des nombres flottants, celle qui est la plus souvent utilisée consiste à représenter la mantisse sous la forme de sa valeur absolue plus un bit pour le signe. La représentation générale est la suivante :

Signe	Caractéristique	Mantisse
1 bit	p bits	m bits

Le signe : il est codé sur un bit et en générale il prend la valeur 0 pour indiquer un nombre positif et la valeur i pour indiquer un nombre négatif.

La caractéristique : elle a pour expression: $c = e + \Delta$, avec:

- e représentant la valeur de **l'exposant**.
- Δ représentant le biais. On dit que l'exposant est biaisé ou encore qu'il est exprimé en code excédent A. En général $\Delta = 2^{p-1}$. Cette représentation nous permet d'économiser un bit de signe pour les valeurs négatives de l'exposant.

La mantisse : le nombre de bits utilisés pour son codage donne la précision avec laquelle le nombre est enregistré et définit par conséquent le nombre de chiffres significatifs.

Exemple:

Pour 4 bits $\Rightarrow$ le biais = $2^{4-1} = 2^3 = 8$
Pour un exposant sur n bits $\Rightarrow$ le biais = 2^{n-1}

Valeur du nombre	valeur biaisé	Valeur en Complément à 2
-8	0000	1000
-7	0001	1001
-6	0010	1010
-5	0011	1011
-4	0100	1100
-3	0101	1101
-2	0110	1110
-1	0111	1111
0	1000	0000
+1	1001	0001
+2	1010	0010
+3	1011	0011
+4	1100	0100
+5	1101	0101
+6	1110	0110
+7	1111	0111

- La caractéristique $c = e + \Delta$,
  Dans notre cas $\Delta = 8 \Rightarrow$ pour e= +7 donc c= 7+8 = 15

Remarque: La représentation de la caractéristique est la même que celle en complément a 2 sauf que le signe est inversé.

La représentation d'un nombre réel en virgule flottante avec un exposant biaisé est réalisée sur 32 bits avec:

1 bit	8 bits	23 bits
S	Caractéristique	$0.5 \leq M < 1$

Exemple:

$-0{,}25_{(10)} = -\ 0{,}01_{(2)} = -\ 0.1 \times 2^{-1}$

Signe de le mantisse = - $\Rightarrow$ 1

Caractéristique c= e + Δ = -1 + 2^7 = -1 +128 = 127 = 01111111

Mantisse = 10000000000000000000000

$\Rightarrow$ la représentation de N est: 10111111110000000000000000000000 = $27760000000_{(8)}$

Remarque :

1- La connaissance du nombre de bits de l'exposant et du biais nous permet de définir les bornes inférieures et supérieures de la valeur absolue des nombres flottants :

- La borne inférieure : M = 1/2 et c = 0 d'où N= $1/2.2^{-\Delta}$ soit : N = $2^{-(\Delta+1)}$
- La borne supérieure : M $\cong$ 1 et c = 2^p-1 d'où e = 2^p-1- Δ soit : N $\cong$ $2^{(c-\Delta)}$

2- L'addition de deux nombres exige une dé normalisation du plus petit opérande pour l'amener à ce qu'il ait le même exposant que l'autre opérande. Une fois l'opération réalisée il faut "re-normaliser" le résultat.

c) Format IEEE :

c.1) Simple précision : le nombre réel est représenté sur 32 bits comme suit :

1 bit	8 bits	23 bits
S	Caractéristique	$0.5 \leq M < 1$

Sachant que la mantisse est normalisé ($0.5 \leq M < 1$); le premier bit est toujours a 1 alors il n'est pas nécessaire de le représenter afin de gagner en précision.

c.1) Double précision : le nombre réel est représenté sur 64 bits comme suit :

1 bit	12 bits	51 bits
S	Caractéristique	$0.5 \leq M < 1$

3.4. Représentation des caractères

La plupart des codes utilisés aujourd'hui pour représenter les caractères alpha-numériques sont exprimés au moyen de huit bits. Les principales contraintes que ces codes doivent intégrer sont :

- Une représentation des chiffres pas trop délicate à manipuler. Elle doit être en fait une simple extension des chiffres décimaux.
- Une représentation des lettres qui facilite leur manipulation. Par exemple des codes de valeurs numériques croissantes pour exprimer les lettres suivant leur ordre alphabétique.
- Une certaine redondance pour faciliter la détection, voire la correction d'éventuelles erreurs de transmission.

3.4.1. Exemples de codages

Le code **BAUDOT** : Il est utilisé en télégraphie depuis 1874. Le codage s'effectue sur 5 bits.

Le code **ELA** : Electronic Industry Association. Il s'agit d'un codage sur 7 bits plus un bit de contrôle de parité impaire. Il est utilisé depuis 1961 pour certaines machines à commande numérique.

Le code **BCDIC** : Binary Coded Decimal Interchange Code, Il est utilisé initialement sur les premiers IBM. Il se compose de 47 symboles: 26 lettres, 10 chiffres, et des caractères graphiques. Les caractères sont codés sur 6 bits et ne comportent pas de minuscules ni de caractères accentués. L'évolution de ce code est 1'**EBCDTC**.

Le code **ERCDIC** : Extended BCDJC. Cette fois les caractères sont codés sur 8 bits. Le codage des caractères est le suivant :

- les lettres minuscules de a à z sont codées par les valeurs de 8lh à A9h.
- les lettres majuscules de A à Z sont codées par les valeurs de Clh à E9h.
- les chiffres de O à 9 sont codés par les valeurs de F0h à F9h.

Le code **ASCII** : American Standard Code for Information Interchange. Aujourd'hui tout le monde, ou presque, s'est rallié à cette représentation dont le codage sur 8 bits est le suivant :

- les caractères de 1 à 37 sont des caractères spéciaux de commande d'appareil ou nécessaire à l'établissement du processus de transmission.
- les lettres minuscules de a à z sont codées par les valeurs de 61h à 7Ah.
- les lettres majuscules de A à Z sont codées parles valeurs de 41h à 5Ah.
- les chiffres de O à 9 sont codés par les valeurs de 30h à 39h.