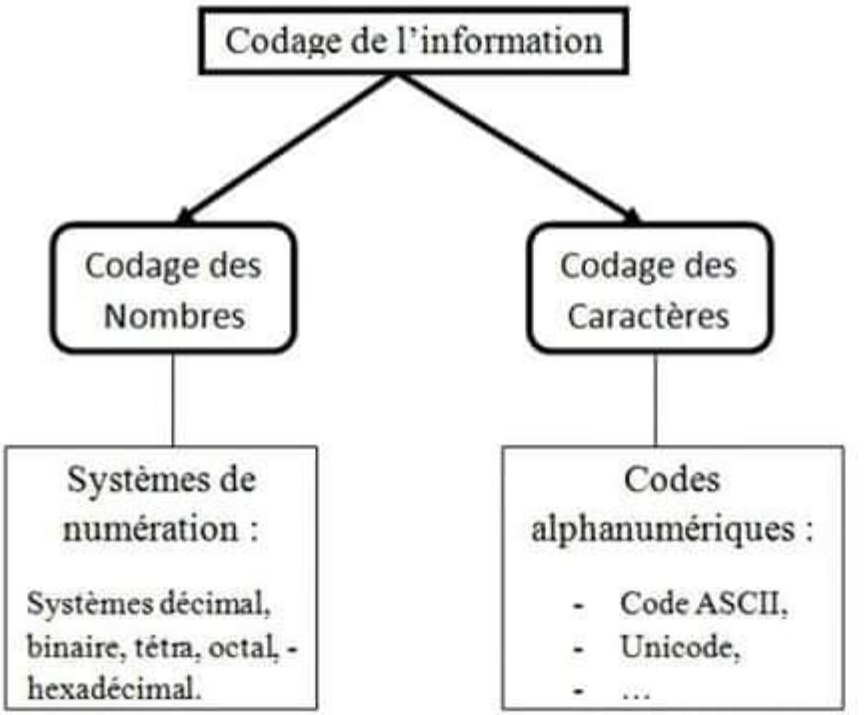
Chapitre II

Codage de l'information

Systèmes de codage de l'information :



1- Codage de l'information

L'ensemble des outils informatiques sont basés sur les mêmes principes de calcul humain. Les calculs habituels sont effectués dans le système de numération décimal, par contre le calculateur électronique ne peut pas utiliser ce système car le circuit électronique ne permet pas de distinguer 10 états. Le système de numération binaire ne comportera que 2 états 0 et 1. Il doit donc tout "traduire" pour que la machine puisse exécuter les instructions relatives aux informations qu'on peut lui donner.

Le codage étant une opération purement humaine, il faut produire des algorithmes qui permettront à la machine de traduire les informations que nous voulons lui voir traiter, en établissant une bijection entre une **information** et une **suite de " 0 " et de " 1 "** qui sont représentables en machine.

1-1- Codage des nombres entiers

Le codage des entiers (ou bien la numération) permet de représenter un nombre par la juxtaposition ordonnée de symboles pris parmi un ensemble.

Exemple :

Le système de numération, usuel dans la vie quotidienne, dispose de dix symboles (chiffres) qui sont : {0, 1, 2, 3, 4, 5, 6, 7, 8,9}.

Il s'agit en fait d'un cas particulier de numération en base 10 (le système décimal).

Lorsque nous écrivons 5876 en base 10, la position des chiffres 5, 8, 7, 6 indique la puissance de 10 à laquelle ils sont associés :

5 est associé à 10^3

8 est associé à 10^2

7 est associé à 10^1

6 est associé à 10^0

Donc, il est possible de représenter tous les nombres dans un système à base b (b entier, b>1).

1-2- Codage des caractères

Les caractères également doivent être représentés en binaire de manière unique. La convention adoptée est d'associer à chaque caractère un nombre décimal et de convertir ce nombre en binaire.

Voyons quelles sont les nécessités minimales pour l'écriture de documents alphanumériques simples dans la civilisation occidentale. Nous avons besoin de :

- Un alphabet de lettres minuscules ={a,b,c,....,z}. soient 26 caractères.
- Un alphabet de lettres majuscules ={A,B,C,....,Z}. soient 26 caractères.
- Des chiffres {0,1,....,9}. soient 10 caractères.
- Des symboles syntaxiques {? , ; (" ... au minimum 10 caractères.

TOTAL = 72 caractères.

Si l'on avait choisi un code à 6 bits le nombre de caractères codifiables aurait été de $2^6 = 64$ nombres donc insuffisant pour nos besoins. Il faut au minimum 1 bit de plus qui permet de définir ainsi $2^7 = 128$ nombres binaires différents, autorisant le codage de 128 caractères.

De nombreux autres codages existent adaptés à diverses solutions de stockage de l'information. Parmi les codages les plus connus et utilisés :

- ✓ **Code ASCII :** Le code ASCII (ASCII = American Standard Code for Information Interchange) est initialement un code à 7 bits, qui permet le codage de 128 caractères. Il englobe des lettres, des chiffes, des signes de ponctuations et un certain nombre de signaux de commande. Toutes ces correspondances sont fixées par l'American National Standards Institutes. Initialement le code ASCII est un code à 7 bits (128 caractères) ; il a été étendu à un code 8 bits ($2^8 = 256$ caractères) permettant le codage des caractères nationaux (en France les caractères accentués comme : ù,à,è,é,â,...etc) et les caractères semi-graphiques. Les pages HTML qui sont diffusées sur le réseau Internet sont en **Code ASCII (8 bits)**. Ce code est normalisé ISO (International Standard Organisation). C'est une extension du code ASCII étendu. Il permet le codage de caractères sur 8 bits, soit 256 caractères possibles.
- ✓ **UNICODE (16 bits) :** Ce code permet de représenter des caractères appartenant à plusieurs langues (arabes, hébreu, japonais, coréen,...) : 65536 caractères.
- ✓ **Code EBCDIC (IBM) :** Le code EBCDIC (Extended Binary Decimal Interchange Code) est un code à 8 éléments binaires utiles, soit 256 combinaisons possibles.

2- Les systèmes de numération

Dans un système de numération de base B tout nombre N peut se décomposer en fonction des puissances entières de la base. Cette décomposition s'appelle la forme polynomiale du nombre N et qui est donnée par :

$$N = \sum_{k=0}^{n-1} a_k B^k = (a_{n-1}a_{n-2}a_{n-3} \dots a_2 a_1 a_2)_B$$

Avec :

B : Base ou nombre de chiffres utilisé dans ce système.

a_k : Chiffre de rang k.

B^k : Pondération associée à a_k.

a. Système décimale : (Base10)

Ce système de numération travaille en base 10, il utilise dix symboles (en l'occurrence des chiffres) qui sont : {0, 1, 2, 3, 4, 5, 6, 7, 8,9}.

Exemple :

$$7239 = (7.10^3 + 2.10^2 + 3.10^1 + 9.10^0)_{10}$$

b. Système binaire : (Base2)

La numération binaire (ou base 2) utilise deux symboles : 0 et 1.

Exemple :

$$4 = 1.2^2 + 0.2^1 + 0.2^0 = (100)_2$$

$$(11110010)_2 = 1.2^7 + 1.2^6 + 1.2^5 + 1.2^4 + 0.2^3 + 0.2^2 + 1.2^1 + 0.2^0$$
$$= (242)_{10}$$

Un code à n bit en base 2 distingue 2^n états ou combinaisons.

d. Système octal : (Base8)

Ce système de numération est très peu utilisé de nos jours. Anciennement, il servait au codage des nombres dans les ordinateurs de première génération. Il utilise 8 symboles : 0, 1, 2, 3, 4, 5, 6, 7.

Exemple :

$$(572)_8 = (5.8^2 + 7.8^1 + 2.8^0)_{10} = (378)10$$

e. Système hexadécimal : (Base16)

Ce système de numération est très utilisé dans les systèmes ordinateurs et micro ordinateurs ainsi que dans le domaine des transmissions de données. Il comporte 16 symboles les chiffres de 0 à 9 et les lettres $\{A, B, C, D, E, F\}$.

Exemple :

$$(D62C)_{16} = (13.16^3 + 6.16^2 + 2.16^1 + 12.16^0)_{10} = (54828)_{10}$$

$$(F7)_{16} = F.16^1 + 7.16^0 = (247)_{10}$$

Table de correspondance entre nombre décimal, binaire, octal et hexadécimal :

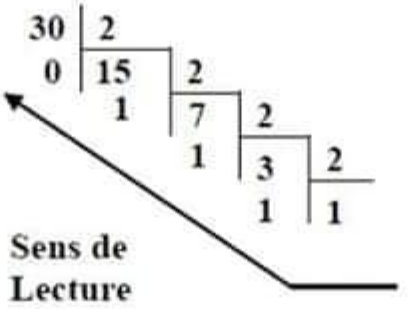
Décimal	Binaire	Tétral	Octal	Hexadécimal
0	0	0	0	0
1	1	1	1	1
2	10	2	2	2
3	11	3	3	3
4	100	10	4	4
5	101	11	5	5
6	110	12	6	6
7	111	13	7	7
8	1000	20	10	8
9	1001	21	11	9
10	1010	22	12	A
11	1011	23	13	B
12	1100	30	14	C
13	1101	31	15	D
14	1110	32	16	E
15	1111	33	17	F

2- Conversions

a. Conversion du système Décimal vers une base quelconque

Pour convertir un nombre de la base **10** vers une base **B** quelconques, il faut faire des divisions successives par B et retenir à chaque fois le reste jusqu'à l'obtention à un quotient inférieur à la base B, dans ce cas le nombre s'écrit de la gauche vers la droite en commençant par le dernier quotient allant jusqu'au premier reste.

Exemple :



Donc : $(30)_{10} = (11110)_2$.

b. Conversion du système Binaire vers l'hexadécimal

Pour convertir du binaire vers l'hexadécimal, on divise le nombre binaire en tranches de 4, en partant de la droite pour la partie entière et en partant de la gauche pour la partie fractionnaire. Chacun des paquets est ensuite converti en hexadécimal.

Exemple :

$(110101110001)_2 = (1101\ 0111\ 0001)_2 = (D71)_{16}$.

c. Conversion du système Hexadécimal vers le Binaire

C'est le processus directement inverse, on écrit chaque quartet sur 4 bits en complétant éventuellement avec des zéros.

Exemple :

$(FA3)_{16} = (1111\ 1010\ 0111)_2$.

d. Conversion du système Binaire vers l'Octal et inversement

On reprend les mêmes principes de la conversion Binaire-Hexadécimal et Hexadécimal-Binaire mais cette fois ci en groupant les données en tranches de 3.

Exemple :

$(101010)_2 = [101]_2\,[010]_2 = (52)_8$.

Remarque : Pour la conversion Octal-Hexadécimal et Hexadécimal-Octal, la plus simple méthode et de passer par le système Binaire,

Exemple :

$(34.61)_8 = (011100{,}110001)_2 = (1C.C4)_{16}$.

3- Représentation des nombres fractionnaire

a. Conversion de la base 10 vers une Base quelconque

Principe de conversion

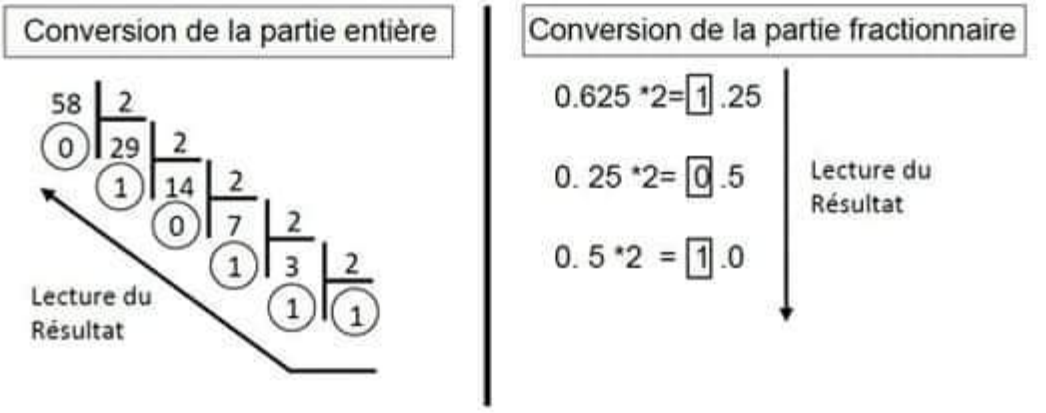
❖ **Partie entière :**

- Divisions entières successives par la base (condition d'arrêt : quotient nul).
- Lecture du reste.

❖ **Partie fractionnaire :**

- Multiplications successives par la base (condition d'arrêt : partie fractionnaire nulle).
- Lecture de la partie entière.

Exemple :



Donc : $(58.625)_{10} = (111010.101)_2$.

4- Représentation des nombres signés

La plupart des dispositifs numériques traitent également les nombres négatifs. Le signe (+) ou (-) est identifié par un bit, dit le bit de signe et le nombre ainsi formé est dit signé . On peut identifier trois principales façons de représenter les nombres négatifs:

1. Représentation en valeur absolue et signe (VAS).
2. Représentation par le complément restreint appelé complément à 1.
3. Représentation par le complément vrai appelé complément à 2.

4-1- La représentation en valeur absolue et signe

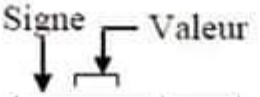
Il s'agit ici d'utiliser un bit pour représenter le signe de la valeur à représenter. Selon que le nombre est positif ou négatif, le bit d'extrême gauche prendra par convention la valeur 0 ou la valeur 1 (0 : positif, 1 : négatif). Par exemple, sur 4 bits, 1 bit sera réservé au signe et trois bits seront utilisés pour représenter les nombres en valeur absolue:

Sur n bits:

- ✓ Signe : bit de poids fort (0 : positif, 1 : négatif).
- ✓ Valeur absolue : $n-1$ bits.
- ✓ Intervalle de valeurs représentées : $[-2^{n-1}+1, 2^{n-1}-1]$.

Exemple :

Sur 3 bits, l'intervalle de valeurs représentées : $[-3, +3]$.



0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	-0
1 0 1	-1
1 1 0	-2
1 1 1	-3

Inconvénients: Cette méthode impose que le signe soit traité indépendamment de la valeur. Il faut donc des circuits différents pour l'addition et la soustraction. De plus, on obtient deux représentations différentes pour 0, soit +0 et -0.

4-2- La notation en complément à 1

On pourrait définir le complément à 1 comme ce qu'il faut ajouter à une valeur pour obtenir la valeur maximale représentable sur le nombre de bits disponibles. On appel complément à un d'un nombre N un autre nombre N' tel que :

$N + N' = 2n - 1$. n : est le nombre de bits de la représentation du nombre N.

Exemple :

Soit $N = 1010$ sur 4 bits donc son complément à un est :

$N' = (2^n - 1) - N.$

$N' = (16 - 1) - (1010)_2 = (15) - (1010)_2 = (1111)_2 - (1010)_2 = 0101.$

On constate que le complément à 1 d'un nombre binaire se trouve simplement en remplaçant les 0 par des 1 et les 1 par des 0.

Notons que l'utilisation du complément à 1 pour représenter les nombres négatifs nous donne encore une double représentation pour le 0.

Exemple :

Valeur	Complément à 1
000	111
001	110
010	101
011	100

Exemple :

On va déterminer la valeur décimale représentée par la valeur 101011 en complément à 1 sur 6 bits :

Le bit poids fort indique qu'il s'agit d'un nombre négatif. Le complément à 1 de la valeur (101011)

- $CA1\ (101011) = -(010100)_2 = -(24)_{10}.$

4-3- La notation en complément à 2

La représentation en complément à deux (complément à vrai) est la représentation la plus utilisée pour la représentation des nombres négatifs dans la machine.

Le complément à 2 d'une valeur binaire est ce qu'il faut ajouter à cette valeur pour qu'elle atteigne une unité de plus que la valeur maximale qu'on peut représenter sur n bits. C'est donc le (complément à 1) + 1.

Cette technique élimine le problème de la double représentation du 0 (+0 et -0) comme c'est le cas dans la représentation "signe et valeur absolue" ou celle du complément à 1.Cela s'explique parce que le complément à 2 permet d'éliminer la double représentation de 0 tout en gardant la facilité de reconnaître le signe par le bit d'extrême gauche. Notons que le complément à 2 du complément à 2 d'un nombre redonne le nombre.

Ainsi, sur 4 bits, avec le signe représenté sur le bit le plus significatif et 3 bits qui permettent de représenter les valeurs, on peut représenter les entiers de -8 à 7, soit un entier négatif de plus qu'un complément à 1.

Sur n bits :

- Complément à 1 : $+1.\ |x| + (-|x|) = 2^n$.
- Intervalle de valeurs représentées : $[-2^{n-1}, 2^{n-1} - 1]$.

Exemple :

Valeur	Complément à 2
001	111
010	110
011	101

5- Opérations arithmétiques

a. L'addition

Il suffit de savoir que :

0+0=0
0+1=1
1+0=1
1+1=10

Et d'effectuer éventuellement une retenue comme dans le cas d'une addition décimal

Exemples :

```
  1 0 1 1 0 1
+   1 0 0 1 0
-----------------
  1 1 1 1 1 1
```

```
    1 1 1 1 1
-------------
    1 1 0 1 1
+     1 1 0 1
-------------
  1 0 1 0 0 0
```

Remarque : L'addition s'effectue de la même manière dans les autres bases.

Exercice :

Effectuer les opérations suivantes:

$(37)_8 + (65)_8 + (116)_8 = (242)_8$.

$(D5E)_{16} + (2F36)_{16} = (3C94)_{16}$.

b. La soustraction

On peut opérer comme dans la soustraction décimale. Voilà ci dessous la table de soustraction binaire :

0-0=0
0-1=1 avec un retenue de 1
1-0=1
1-1=0

Exemple :

	1		**1**	**1**			
	1	**0**	**1**	**0**	**0**	**1**	**1**
-		**1**	**0**	**1**	**1**	**0**	**1**
	0	**1**	**0**	**0**	**1**	**1**	**0**

Remarque : La soustraction s'effectue de la même manière dans les autres bases.

Exercice :

Effectuer les opérations suivantes:

$(137)_8 - (63)_8 = (54)_8$.

$(F23)_{16} - (2A6)_{16} = (C7D)_{16}$

$(FD28)_{16} - (E5E)_{16} - (2F36)_{16} = (FD28)_{16} - [(E5E)_{16} + (2F36)_{16}] = (FD28)_{16} - (3D94)_{16} = (BF94)_{16}$.

c. La multiplication

La multiplication en binaire est très simple, voilà la table de multiplication :

$0\times0=0$
$0\times1=0$
$1\times0=0$
$1\times1=1$

Remarque : On doit bien tenir compte des décalages.

Exemple :

					1	0	1	1
					1	1	0	1
×					1	0	1	1
			1	0	1	1		
+		1	0	1	1			
	1	0	0	0	1	1	1	1

Exercice :

Effectuer les opérations suivantes:

$(237)_8 * (63)_8 = (17655)_8$.

			2	3	7
×				6	3
+			7	3	5
	1	6	7	2	
=	1	7	6	5	5

$(F3)_{16} * (206)_{16} = (1EBB2)_{16}$.

d. La division

La division entre deux nombres binaires est identique à la division euclidienne.

Exemple :

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

Il suffit en fait de soustraire 101 lorsqu'on le peut, et d'abaisser le chiffre suivant :

$11101 = 101 \times 101 + 100.$