Chapitre 1

Représentation numérique de l'information

Objectifs :
— Comprendre les différentes manières d'encoder des nombres ;
— Distinguer les avantages et les inconvénients de ces encodages.

1.1 Introduction

Lorsque nous utilisons un logiciel, chaque action que nous effectuons avec la souris ou le clavier par exemple est traduite en langage informatique puis exécutée par l'ordinateur.

"Transmettre des informations sous forme numérique suppose entre autres d'optimiser la taille des messages transmis pour éviter de surcharger les canaux de transmission, d'être capable de rectifier des erreurs apparues en cours de transmission, de crypter les contenus et d'authentifier les émissaires et les destinataires..." (Dunod, 2006)

Les images, le son, le texte et les vidéos sont des informations qu'un ordinateur peut traiter. Un ordinateur est composé de circuits électroniques qui fonctionnent à l'électricité. L'information est représentée physiquement par un signal électrique ou magnétique qui, au-delà d'un certain seuil, correspond à la valeur 1 si non par 0. Par conséquent, l'information est toujours sous forme d'un ensemble de nombres écrit en base 2 par exemple 011001. En 1939, Claude Shannon a été le premier à faire un parallèle entre l'algèbre booléenne (une algèbre binaire, n'acceptant que vrai et faux comme valeurs, et trois fonctions logiques "Et (*), Ou(+), Non(-)") et le fonctionnement des circuits électriques. Le vrai/faux se transforme en 1 : le courant passe, 0 : il ne passe pas. C'est Shannon qui popularise le terme de bit :

Définition 1 :

Le terme **bit** (b minuscule dans les notations) signifie *binary digit* , c'est-à-dire 0 ou 1 en numérotation binaire. Il s'agit de la plus petite unité d'information manipulable par une machine numérique.

L'**octet** (en anglais **byte** ou B majuscule dans les notations) est une unité d'information composée de 8 bits. Il permet par exemple de stocker un caractère comme une lettre ou un chiffre.

D'autres termes sont aussi utilisés pour définir certaines longueurs de nombre :
— Une unité d'information composée de 16 bits est généralement appelée **mot** (en anglais **word**).

— Une unité d'information de 32 bits de longueur est appelée **mot double** (en anglais **double word**, d'où l'appellation dword).

Jusqu'en 1998 ; 1024 octets valaient 1 kilooctet. Et depuis les nouvelles unités standardisées par l'organisme international IEC sont les suivantes :

— Un kilooctet (ko) = 10^3 octets
— Un mégaoctet (Mo) = 10^6 octets
— Un gigaoctet (Go) = 10^9 octets
— Un téraoctet (To) = 10^{12} octets
— Un pétaoctet (Po) = 10^{15} octets
— Un exaoctet (Eo) = 10^{18} octets
— Un zettaoctet (Zo) = 10^{21} octets
— Un yottaoctet (Yo) = 10^{24} octets

Remarque 1 :

Un mégaoctet devrait en principe valoir 1000 x 1000 octets, c'est-à-dire 1'000'000 d'octets, mais il vaut 1024 x 1024 octets en informatique, c'est-à-dire 1'048'576 octets... ce qui correspond à une différence de 4.63 % !

1.2 Les bases décimales, binaires et hexadécimales

1.2.1 La base décimale

Le système décimal (base 10) est celui que nous utilisons dans la vie quotidienne parce que nous avons commencé à compter avec nos doigts. Il comporte 10 symboles de 0 à 9. C'est un système positionnel, c'est-à-dire que l'endroit où se trouve le symbole définit sa valeur.

Par exemple : $(9'875)_{(10)} = 9 \cdot 10^3 + 8 \cdot 10^2 + 7 \cdot 10^1 + 5 \cdot 10^0$

10 représente la base et les puissances de 0 à 3 le rang de chaque chiffre. Quelle que soit la base, le chiffre de droite est celui des unités. Celui de gauche est celui qui a le poids le plus élevé.

1.2.2 Binaire

Dans les domaines de l'automatisme, de l'électronique et de l'informatique, nous utilisons la base 2. Tous les nombres s'écrivent avec deux chiffres uniquement, 0 et 1. Car l'algèbre booléenne est à la base de l'électronique numérique. Par exemple, un interrupteur est ouvert ou fermé, une diode électroluminescente (ou LED en anglais) est allumée ou éteinte, une tension est présente ou absente, un champ magnétique est orienté Nord-Sud ou Sud-Nord.

Le chiffre binaire, qui peut prendre ces deux états, est nommé **Bit** (Binary digit). Les règles sont les mêmes que pour le décimal.

Par exemple, $1101_{(2)} = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13_{(10)}$

Conversion binaire décimale

Une autre façon d'obtenir le nombre en base 10, connaissant son écriture en base 2, est d'établir la valeur de chaque chiffre du nombre : celui le plus à droite représente 1, le deuxième représente 2, le troisième représente $2^2 = 4$, le quatrième représente $2^3 = 8$... Par exemple si nous voulons connaitre la valeur décimale du nombre 10 110, nous établissons d'abord une grille avec les différentes puissances de 2 :

64	32	16	8	4	2	1

Ensuite, nous plaçons le nombre en binaire sous cette grille en mettant bien le chiffre des unités sous le carré le plus à droite et nous barrons les cases dont le chiffre associé est 0 :

64	32	16	8	4	2	1
		1	0	1	1	0

Il ne reste plus qu'à additionner les nombres qui restent : $16 + 4 + 2 = 22$

1 Écrire en base 10 les nombres suivants :

1. $101_{(2)}$
2. $10000_{(2)}$
3. $1111_{(2)}$
4. $10101_{(2)}$
5. $10_{(2)}$
6. $111000_{(2)}$

Conversion décimale binaire

Pour trouver l'écriture binaire d'un nombre écrit en base 10, il existe principalement deux méthodes.

La première méthode consiste à décomposer le nombre en somme de puissance de 2 et à utiliser la grille précédente.

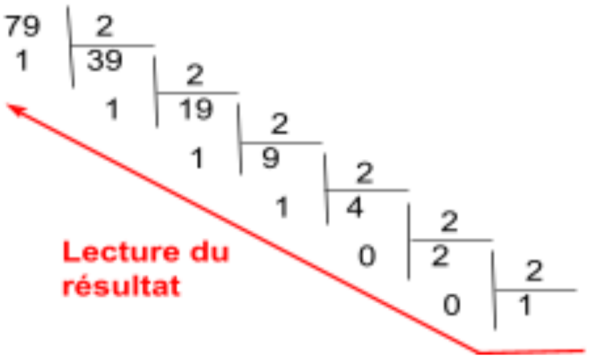
Plus concrètement, si nous voulons écrite 99 en binaire. La plus grande puissance de 2 que nous pouvons prendre dans 99 est 64 (128, la suivante, est trop grande). Il reste ensuite : 99-64=35. Dans 35 on peut mettre 32, et il restera 35-33=3. Dans 3 on ne peut ni mettre 16, ni mettre 8, ni mettre 4. On peut mettre 2. Il reste 1. Dans 1 on peut mettre 1.

Cela signifie que 99=64+32+2+1. Sur notre grille cela donne :

64	32	16	8	4	2	1
1	1	0	0	0	1	1

Donc $99_{(10)} = 1100011_{(2)}$.

La deuxième méthode consiste effectuer des divisions successives par 2. Le nombre en binaire se lira à l'aide des restes des différentes divisions :



Ainsi, $79_{(10)} = 1001111_{(2)}$

2 Les nombres suivants sont écrits en base 10. Donner leur écriture en base 2 :

1. 75
2. 12
3. 27
4. 153
5. 100
6. 200
7. 1000
8. 2000

1.2.3 *La base hexadécimale

C'est le code utilisé dans la programmation de certains automates et microprocesseurs. Il est composé de : 10 chiffres de 0 à 9, et 6 lettres de A à F. Les adresses MAC (adresses uniques associées à chaque carte réseau dans le monde) sont aussi écrites en base hexadécimale.

La manipulation des nombres écrits en binaire est difficile pour l'être humain et la conversion en décimal n'est pas simple. C'est pourquoi nous utilisons de préférence le système hexadécimal (base 16).

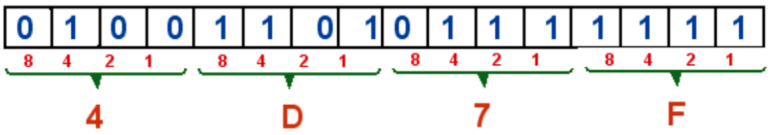
Le tableau ci-dessous montre la représentation des nombres de 0 à 15 dans les bases 10, 2 et 16.

Décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Binaire	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Hexadécimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Par exemple, $B4F_{(16)} = B \cdot 16^2 + 4 \cdot 16^1 + F \cdot 16^1 = 11 \cdot 16^2 + 4 \cdot 16^1 + 15 \cdot 16^0 = 2895_{(16)}$

Conversion en binaire

Pour convertir un nombre binaire en hexadécimal il suffit de remarquer que chaque groupe de 4 bits représente un chiffre en hexadécimal :



3 Écrire les nombres suivants en base hexadécimale :

1. $10010010110_{(2)}$
2. $111110_{(2)}$
3. $1000110101110101_{(2)}$
4. $11110000000011_{(2)}$

Conversion en décimal

La méthode par division (par 16) s'applique comme en binaire (par 2).

4 Écrire les nombres suivants en hexadécimal :

1. 92
2. 256
3. 500
4. 1023

1.2.4 Opération sur les nombres en binaires

Tout comme pour l'addition en colonne, pour additionner deux nombres écrits en binaires il faut les additionner bit à bit.

5 Quel est le résultat des calculs binaires suivants :

a) $0+0=$

b) $1+0=$

c) $0+1=$

d) $1+1=$

e) $1+1+1=$

À l'aide de l'exercice suivant effectué l'addition suivante :

$$1100101001 + 101110101$$

Pour cela, compléter l'addition en colonne suivante :

$$\begin{array}{cccccccccc} & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ + & & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ \hline \end{array}$$

6 Effectuer les additions binaires suivantes :

1. $1001 + 101$
2. $10011 + 110011$
3. $110001 + 11010$
4. $110101 + 101110$

1.3 Représentation des nombres entiers

Nous arrivons à écrire tous les nombres entiers naturels en binaire. Il faut maintenant utiliser un certain nombre de règles pour pouvoir représenter n'importe quel nombre avec un ordinateur : nombres entiers, nombres relatifs, nombres rationnels...

1.3.1 Représentation des nombres entiers naturels

Rappel : Les nombres entiers naturels sont l'ensemble des nombres entiers et positifs. Il est désigné par la lettre $\mathbf{N} = \{0, 1, 2, \ldots\}$.

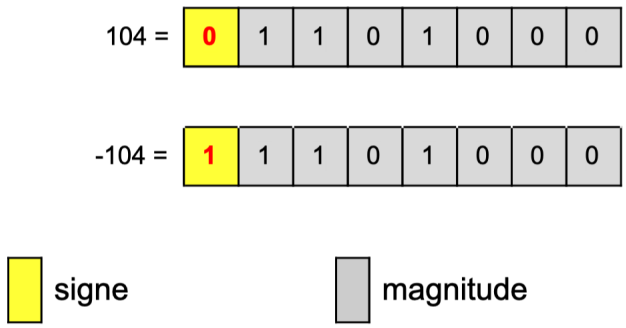
Nous pouvons par exemple décider que les nombres seront codés sur un octet (8 bits). Le plus petit nombre sera 00000000 = 0 et le plus grand sera 11111111=255.

Problème : En faisant ainsi, nous ne pouvons pas représenter des nombres négatifs.

1.3.2 Représentation des nombres entiers relatifs signés

Rappel : Les nombres entiers sont l'ensemble des nombres entiers négatifs et positifs. Il est désigné par la lettre $\mathbf{Z} = \{\ldots, -2, -1.0, 1, 2, \ldots\}$.

La première façon pour représenter les nombres négatifs est de décider que le bit le plus gauche représente le signe du nombre.



Sur 8 bits plus grand nombre sera 01111111=127 et le plus petit sera 111111111=-127.

Il est possible d'additionner deux nombres pour autant qu'ils soient tous les deux positifs et que le résultat ne dépasse pas le nombre plus grand nombre pouvant être écrit :

104	**0**	1	1	0	1	0	0	0
+ 12	**0**	0	0	0	1	1	0	0
116	**0**	1	1	1	0	1	0	0

104	**0**	1	1	0	1	0	0	0
+ 30	**0**	0	0	1	1	1	1	0
Résultat faux	**1**	1	1	1	0	1	1	0

Le résultat de 104 + 30=134 amène un dépassement de capacité. Le nombre entier signé maximum sur 8 bits est 127.

Problème : Bien que pratique, cette façon de coder les nombres négatifs présente deux problèmes :

1. zéro est représenté de deux façons différentes : 1000000 et 00000000.
2. si on additionne deux nombres opposés, nous n'obtenons pas 0.

1.3.3 * Représentation des nombres en complément à deux

Cette représentation résout les problèmes de la représentation signée. Dans cette représentation, pour obtenir l'opposé d'un nombre positif écrit en binaire, nous allons effectuer les deux étapes suivantes :

1. On inverse tous les bits du nombre positif (on change les 0 en 1 et inversement). Cela s'appelle le **complément à 1**.
2. On ajoute 1.

Par exemple, nous aimerions écrire le nombre -80.
Premièrement, 80 s'écrit en binaire : **01010000**.

On prend le complément à 1 de ce nombre, ce qui donne : **10101111**

Puis on ajoute 1 :

$$\begin{array}{cccccccc} & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ + & & & & & & & & 1 \\ \hline & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}$$

L'avantage de cette méthode est que la somme deux nombres opposés donnera bien 0.

7 En supposant que les nombres soient écrits sur 8 bits, vérifier que 80 + (-80) = 0.

1.3.4 Résumé

n=4	non signé	signe-magnitude	complément à 2
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	0	-8
1001	9	-1	-7
1010	10	-2	-6
1011	11	-3	-5
1100	12	-4	-4
1101	13	-5	-3
1110	14	-6	-2
1111	15	-7	-1

8 Complétez le tableau ci-dessous :

Nombres	Bases			
	2	10	16	ascii
11001110101_2				
23670_{10}				
$FA2C5_{16}$				

9 Utiliser un tableur pour faire une conversion d'une base à l'autre.

10 Donner la représentation signée sur un octet des nombres suivants :

1. -100
2. 38
3. -200
4. -83

11 *

a) Donner la représentation en complément à deux des nombres suivants :

(a) -95
(b) -64

b) Vérifier que 95+(-95) donne bien 0 sur un octet.

c) Vérifier que 64+(-6) donne bien 0111010 sur un octet.

12

Nous donnons trois nombres binaires :

1) 00101010
2) 1011
3) 10111111

1. Que valent ces nombres en représentation non signée ?
2. * Que valent ces nombres en représentation signée ?
3. * Que valent ces nombres en complément à 2 ?