Chapitre 1

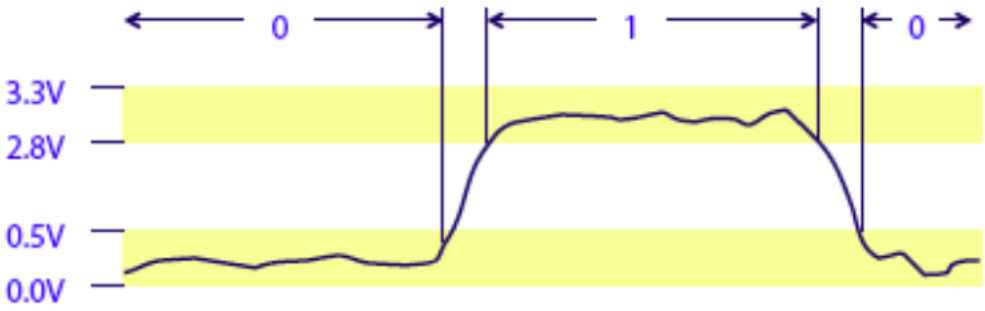
Représentation numérique de l'information

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 $2^3 = 8$, etc. comme dans la grille ci-dessous :

64	32	16	8	4	2	1

Par exemple si nous voulons connaitre la valeur décimale du nombre 10110, nous établissons d'abord une grille avec les différentes puissances de 2 ; 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. $111000_{(2)}$

Vous pouvez utiliser le tableau ci-dessous :

		64	32	16	8	4	2	1

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.

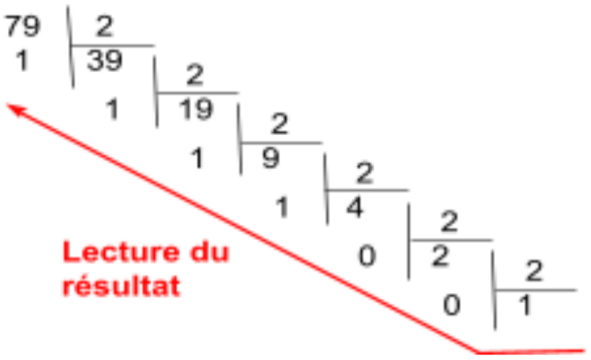
Plus concrètement, si nous voulons écrite 99 en binaire. La plus grande puissance de 2 que nous pouvons prendre dans 99 est 64 (128, la suivante, est trop grande). Il reste ensuite :

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 :
3-**2** = 1 Il reste 1. Dans 1 on peut mettre 1 :
1-**1** = 0
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).

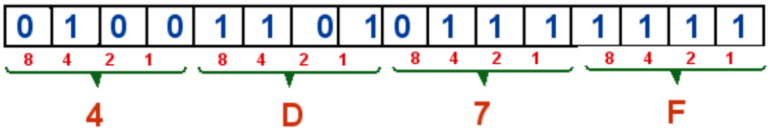
Le tableau ci-dessous montre la représentation des nombres de 0 à 15 dans les bases 10, 2 et 16.

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

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

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 :

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

6 Effectuer les additions binaires suivantes :

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

7

1. On voudrait encoder des emojis en binaire ; en d'autres mots, on voudrait faire correspondre chaque emoji à un code binaire différent.
   (a) Combien peut-on encoder de symboles (ici des emojis) différents sur 3 bits ?
   (b) Sur 5 bits ?
   (c) Sur 11 bits ?
2. Quel est le plus grand nombre qu'on peut écrire sur 5 bits ?
3. On se pose maintenant la question inverse :
   (a) Combien me faut-il de bits pour encoder 8 symboles différents ?
   (b) 9 symboles ?
   (c) 69 symboles ?

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 $\mathbf{N} = \{0, 1, 2, \ldots\}$.

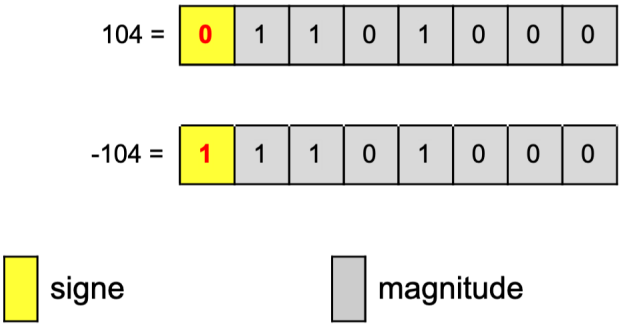
Nous pouvons par exemple décider que les nombres seront codés sur un octet (8 bits). Le plus petit nombre sera 00000000 = 0 et le plus grand sera 11111111=255.

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 :

$$\begin{array}{r} 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1 \\ +\qquad\qquad\qquad 1 \\ \hline 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0 \end{array}$$

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

8 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

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

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

10 *** Complément à deux**

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.

11 Nous donnons trois nombres binaires :

a) 00101010

b) 1011

c) 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 ?

1.4 Les codes de caractères

1.4.1 La table ASCII

Decimal - Binary - Octal - Hex – ASCII Conversion Chart

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	Space	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	Backspace	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

Le binaire permet de coder les nombres que les systèmes informatiques peuvent manipuler. Cependant, l'ordinateur doit aussi utiliser des caractères alphanumériques pour mémoriser et transmettre des textes par exemple de l'ordinateur vers l'imprimante, d'un automate programmable vers un terminal, d'un clavier vers un processeur, etc.

Le code **ASCII** (American Standard Code for Information Interchange) représente les caractères sur 7 bits (c'est-à-dire 128 caractères possibles, de 0 à 127). Les codes 0 à 31 ne sont pas des caractères. On les appelle caractères de contrôle car ils permettent de faire des actions telles que : retour à la ligne (CR). Bip sonore (BEL). Les codes 65 à 90 représentent les majuscules. Les codes 97 à 122 représentent les minuscules. Le caractère A par exemple à pour code 65 soit 01000001 en binaire.

Par exemple :

a) Le caractère f : 102

b) le point d'interrogation ? : 63

c) Le chiffre 2 : 50

Le code ASCII a été mis au point pour la langue anglaise, il ne contient donc pas de caractères accentués, ni de caractères spécifiques à une langue. Le code ASCII a donc été étendu à 8 bits pour pouvoir coder plus de caractères (on parle d'ailleurs de code ASCII étendu...)

1.4.2 Unicode

Le code ASCII est utilisable pour l'anglais mais limité pour les autres langues. Il n'y a que 95 caractères imprimables.

Le code Unicode est un système de codage des caractères sur 16 bits mis au point en 1991. Le système Unicode permet de représenter n'importe quel caractère par un code sur 16 bits, indépendamment de tout système d'exploitation ou langage de programmation (environ 64 000 caractères).

Il regroupe ainsi la quasi-totalité des alphabets existants (arabe, arménien, cyrillique, grec, hébreu, latin, ...) et est compatible avec le code ASCII.

L'ensemble des codes Unicode est disponible sur le site **http ://www.unicode.org**.

1.4.3 Exercices

12 Traduire en code ASCII binaire les textes :

1. Sismondi
2. Cc cv ?

13 Traduire en lettres les mots suivants qui sont codés en ASCII (binaire) :

1. 01100011 01101111 01110101 01100011 01101111 01110101
2. 01001000 01100101 01101100 01101100 01101111 00100000 01110111 01101111 01110010 01101100 01100100 00100000 00100001

14 * En s'aidant de la table ASCII, classer par ordre croissant les caractères suivants (ne pas tenir compte des virgules) : ESC, A, NUL, Delete, m, 8, <, a, ?

15 Complétez le tableau ci-dessous :

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