CHAPITRE 1 : SYSTEMES DE NUMERATION ET CODAGE DE L'INFORMATION

1.1 Introduction

Les ordinateurs et les machines de calcules modernes ne font pas le calcul en système décimale (base 10). La base 10 est généralement utilisée par les humains, donc, il faut convertir les nombres d'une base à une autre.

1.2 Systèmes de numération

Plusieurs systèmes de numération existent :

- Système octal
- Système hexadécimal
- Système décimal
- Système binaire

Dans une base donnée b un nombre peut être exprimé sous la forme d'un polynôme. Supposons le nombre suivant X = ($a_n\, a_{n-1} \ldots a_0$) la forme pronominale de ce nombre est $X = \sum_{i=0}^{n} a_i\, b^i$. Où a_i sont les coefficients et b est la base.

avec:

a_0 : le chiffre ayant le poids le plus faible (LSB : Less Significant Bit)

a_n: le chiffre ayant le poids le plus fort (MSB: Most Significant Bit).

Exemples :

$(7134)_{10}$ sa forme pronominale est : $7 \times 10^3 + 1 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$

$(11010)_2$ sa forme pronominale est : $1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$(726)_8$ sa forme pronominale est : $7 \times 8^2 + 2 \times 8^1 + 6 \times 8^0$

$(B27)_{16} = B \times 16^2 + 2 \times 16^1 + 7 \times 16^0$

Les bases des systèmes de numération usuelles sont représentées par le tableau suivant:

Tableau 1.1: Systèmes de numération usuelles.

Système	*Décimale*	*Binaire*	*Octal*	*Hexadécimal*
Base	10	2	8	16
Chiffres	0,1,2,3,4,5,6,7,8,9	0,1	0,1,2,3,4,5,6,7	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

1.3 Conversion entre les systèmes de numération

1.3.1 Conversion d'un nombre de base (b) en un nombre décimal

La valeur décimale d'un nombre N dans une base b peut être obtenue comme suit :

$(N)_b = (a_n\, a_{n-1} \ldots a_0)_b = a_n \times b^n + a_{n-1} \times b^{n-1} + \ldots + a_0 \times b^0$.

a) Conversion binaire – décimal:

$(11100101)_2 = (\ldots)_{10}$

$1.2^7 + 1.2^6 + 1.2^5 + 0.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = 128+64+32+0+0+4+0+1 = (229)_{10}$

b) conversion octal – décimal:

$(417)_8 = (\ldots)_{10}$

$4.8^2 + 1.8^1 + 7.8^0 = 256+8+7 = (271)_{10}$

c) conversion hexadécimal – décimal:

$(C26)_{16} = (\ldots)_{10}$

$12.16^2 + 2.16 + 6 = 3072 + 32 + 6 = (3110)_{10}$

1.3.2 Conversion d'un nombre décimal en un nombre de base (b)

L'algorithme de conversion est le suivant :

1. Diviser (division entière) le nombre décimal par la base « b ».
2. Prendre le résultat de la division, et le diviser de nouveau par la base « b ».
3. Reprendre l'étape 2 jusqu'à obtenir un résultat nul.
4. Le nombre équivalant en base « b » s'obtient en prenant les restes de la division de la dernière à la première de gauche à droite.

a. Conversion en base 2

$(19)_{10} = (\ldots)_2$

19 :2 = 9 reste = 1

9 :2 = 4 reste = 1

4 :2 = 2 reste = 0

2 :2 = 1 reste = 0

1 :2 = 0 reste = 1

$(19)_{10} = (10011)_2$

b. Conversion en base 8

$(309)_{10} = (\ldots)_8$

309:8 = 38 reste =5

38:8 = 4 reste = 6

4 :8 = 0 reste = 4

$(309)_{10} = (465)_8$

c. Conversion en base 16

$(504)_{10} = (\ldots)_{16}$

504:16 = 31 reste = 8

31 :16 = 1 reste = 15 (F)

1 : 16=0 reste = 1 $(504)_{10} = (1F8)_{16}$

1.4 Conversion des nombres fractionnaires

Le nombre fractionnaire est un nombre entier suivi d'une fraction inférieur à 1.

Exemple : 9/4 ; 92.105 , 0.73.

1.4.1 Conversion d'un nombre fractionnaire de base (b) en un nombre décimal

La valeur décimale d'un nombre fractionnaire « *N* » exprimé dans une base « b » peut être obtenue par la forme polynomiale en prenant en considération les puissances négatives d'une manière décroissante.

Exemple: $(29,053)_{16} = 2 \times 16^{1}+9 \times 16^{0}+0 \times 16^{-1} +5 \times 16^{-2}+3 \times 16^{-3}$

$(29,053)_{16} = (41.02026)_{10}$

1.4.2 Conversion d'un nombre décimal en un nombre fractionnaire de base (b)

La conversion se fait comme suit :

1. Multiplier le nombre décimal fractionnaire par la base « b ».
2. Prendre la partie fractionnaire du résultat et la multiplier une autre fois par la base « b ».
3. Répéter l'étape 2 jusqu'à ce que la partie fractionnaire soit nulle, où atteindre le nombre max de chiffres après la virgule.
4. Prendre les parties entières des résultats de multiplications de la première à la dernière de gauche à droite.

Exemple : on veut calculer l'équivalent en binaire du nombre décimal 0.8125.

2	0.8125
1	.625
1	.25
0	.5
1	0

Alors, $(0,8125)_{10}= (0,1101)_{2}$

1.4.3 Conversion Binaire ↔ Octal

Le système octal contient sept chiffres de zéro à sept, ils nécessitent trois bits pour être codée en binaire 0→000, 1→001, 7→111. Ainsi, pour convertir un nombre octal en binaire, on remplace sur trois bits chaque chiffre par son équivalant binaire.

Inversement, pour convertir un nombre binaire en octal, on rassemble à trois, de droite à gauche pour la partie entière, on ajoute des zéros à *gauche* si c'est nécessaire (le groupe contient moins de 3), et de gauche à *droite* pour la partie fractionnaire (on ajoute des zéros à droite si le groupe contient moins de 3), ensuite on convertit chaque groupe par son équivalant octal.

Exemple :

$(73)_{8}=(111\ 011)_{2}$

$(001\ 110\ 101)_{2}=(165)_{8}$

1.4.4 Conversion Binaire ↔ hexadécimal

Le système hexadécimal contient 16 chiffres : de zéro à neuf et de A à F, ces chiffres nécessitent quatre bits pour être codée en binaire 0→0000, F→1111. Pour convertir un nombre hexadécimal en binaire, on remplace simplement sur quatre bits, chaque chiffre par son équivalant binaire.

D'une manière similaire, afin de convertir un nombre binaire en hexadécimal, on regroupe à 4 les bits de droite à gauche pour la partie entière (on ajoutant des zéros à *gauche* si le nombre contient moins de 4) et de gauche à droite pour la partie fractionnaire (on ajoutant des zéros à *droite* si le nombre contient moins de 4), on remplace par la suite chaque groupe par son équivalant hexadécimal.

Exemples :

$(3A6)_{16} = (0011\ 1010\ 0110)_2$

$(0001\ 1011\ 0111)_2 = (1B7)_{16}$

1.5 Operations arithmétique de base

1.5.1 Addition binaire

Opération	Résultat
0 + 0	0
0 + 1	1
1 + 0	1
1 + 1	0 et on retient 1
1 + 1+ 1	1 et on retient 1

1.5.2 Soustraction binaire

Opération	Résultat
0 - 0	0
0 - 1	1 et on retient 1
1 - 0	1
1 - 1	0
0 – 1- 1	0 et on retient 1
1 -1 -1	1 et on retient 1

1.5.3 Multiplication binaire

Opération	Résultat
0 * 0	0
0 * 1	0
1 * 0	0
1 * 1	1

1.5.3 Division binaire

Opération	Résultat
0 / 1	0
1 / 1	1

1.6 Codage de l'information

Les signaux numériques (sons, images, textes, vidéos ...) sont codés sous forme de "0" et de "1" compréhensibles par un processeur. Un"bit" (abréviation de "binary digit") peut prendre la valeur 0 ou 1.

Remarques

- Un nombre binaire qui se termine par 0 est *pair*, alors qu'un nombre binaire qui se termine par 1 est un nombre *impair*.
- Un ensemble de 4 bits s'appelle un quartet.
- Un ensemble de 8 bits est appelé un octet ou un byte.
- Un ensemble de 16 bits est appelé un mot ou un word.
- Un ensemble de 32 bits est appelé un mot long ou un double word.

1.7 Les codes binaires

1.7.1 Code binaire naturel

C'est un code simple qui représente des chiffres positifs. Le tableau suivant donne le code binaire naturel pour un exemple d'un mot de 4 bits.

Tableau 1.2: Code binaire naturel.

	Puissances successives			
$(\text{Chiffre})_{10}$	2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
⋮	⋮	⋮	⋮	⋮
15	1	1	1	1

1.7.2 Code BCD (Décimal Codé Binaire)

C'est un système de numération utilisé en informatique et en électronique pour coder des nombres en ressemblant à la représentation humaine ordinaire (base 10). Dans ce code, les nombres sont représentés par les chiffres décimaux qui les composent où chaque chiffre est codé sur quatre bits.

Exemple : $(271)_{10}$= 0010 0111 0001

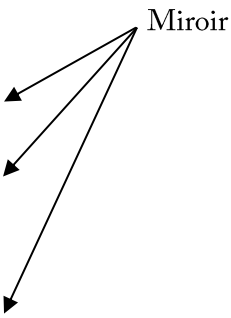
Remarque : Les codes (1010, 1011, 1100, 1101, 1110, 1111) ne sont pas inclus en BCD. Dans le système binaire ordinaire certaines variables peuvent changer d'état en même temps, (pour passer de 3 à 4, 3 variables changent d'état). Dans un système automatique, il est souvent irréalisable que deux variables changent d'état simultanément.

1.7.3 Code GRAY ou code binaire réfléchi

Le code de Gray, appelé aussi code binaire réfléchi, est un type de codage binaire qui se base sur le principe de ne pas modifier qu'un seul bit à la fois lorsque un nombre est incrémenté d'une unité. Il est nommé réfléchi puisque il faut recopier les valeurs comme si elles étaient réfléchies dans un miroir.

Tableau 1.3: Code binaire réfléchi (GRAY).

Décimal	Code binaire réfléchi			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1



1.7.3.1 Conversion binaire - Gray

1. le MSB gray garde la même valeur du MSB binaire.
2. Sommer le MSB binaire avec le bit de degré inferieur, (on ne considère pas la retenue).
3. Continue l'addition des bits adjacents jusqu'à atteindre le LSB.
4. Si le bit g_{n+1} et b_{n+1} ont la même valeur, alors g_n=0.
5. Si le bit g_{n+1} est différent de b_{n+1}, alors g_n=1.

Exemple :

$N=110110_{(2)naturel}$

Naturel= 1→ 1→0→ 1→1→0
↓ ↓ ↓ ↓ ↓ ↓
Gray= 1 0 1 1 0 1

1.7.3.2 Conversion gray - binaire

1. On garde le même MSB.
2. On compare le bit binaire naturel de rang n+1 au bit binaire réfléchi de rang n (on met 1 s'ils sont différents et 0 si sont égaux).

Exemple :

Gray = 1 1 0 1 0
Naturel= 1 0 0 1 1

1.7.5 Code ASCII (American Standard Code for Information Interchange)

C'est une norme informatique de codage de caractères apparue dans les années 1960. C'est la norme de codage de caractères la plus influente à ce jour. ASCII définit 128 codes à 7 bits, comprenant 95 caractères imprimables : les chiffres arabes de 0 à 9, les 26 lettres de l'alphabet latin en minuscules et en majuscules, et des symboles mathématiques et de ponctuation.

Tableau 1.4 : Les 32 premiers caractères de la table ASCII sont des codes de contrôle non imprimables et sont utilisés pour contrôler des périphériques tels que des imprimantes.

DEC	OCT	HEX	BIN	Symbol	Description
0	000	00	00000000	NUL	Null char
1	001	01	00000001	SOH	Start of Heading
2	002	02	00000010	STX	Start of Text
3	003	03	00000011	ETX	End of Text
4	004	04	00000100	EOT	End of Transmission
5	005	05	00000101	ENQ	Enquiry
6	006	06	00000110	ACK	Acknowledgment
7	007	07	00000111	BEL	Bell
8	010	08	00001000	BS	Back Space
9	011	09	00001001	HT	Horizontal Tab
10	012	0A	00001010	LF	Line Feed
11	013	0B	00001011	VT	Vertical Tab
12	014	0C	00001100	FF	Form Feed
13	015	0D	00001101	CR	Carriage Return
14	016	0^E	00001110	SO	Shift Out / X-On
15	017	0F	00001111	SI	Shift In / X-Off
16	020	10	00010000	DLE	Data Line Escape
1 7	021	11	00010001	DC1	Device Control 1 (oft. XON)
18	022	12	00010010	DC2	Device Control 2
19	023	13	00010011	DC3	Device Control 3 (oft. XOFF)
20	024	14	00010100	DC4	Device Control 4
21	025	15	00010101	NAK	Negative Acknowledgement
22	026	16	00010110	SYN	Synchronous Idle
23	027	17	00010111	ETB	End of Transmit Block
24	030	18	00011000	CAN	Cancel
25	031	19	00011001	EM	End of Medium
26	032	1A	00011010	SUB	Substitute
27	033	1B	00011011	ESC	Escape
28	034	1C	00011100	FS	File Separator
29	035	1D	00011101	GS	Group Separator
30	036	1^E	00011110	RS	Record Separator
31	037	1F	00011111	US	Unit Separator

Tableau 1.5 : Les codes 32-127 sont communs à toutes les différentes variantes de la table ASCII, ils sont appelés caractères imprimables (lettres, chiffres, signes de ponctuation et quelques symboles divers).

DEC	OCT	HEX	BIN	Symbol	Description
32	040	20	00100000		Space
33	041	21	00100001	!	Exclamation mark
34	042	22	00100010	"	Double quotes (or speech marks)
35	043	23	00100011	#	Number
36	044	24	00100100	$	Dollar
37	045	25	00100101	%	Per cent sign
38	046	26	00100110	&	Ampersand
39	047	27	00100111	'	Single quote
40	050	28	00101000	(	Open parenthesis (or open bracket)

41	051	29	00101001	)	Close parenthesis (or close bracket)
42	052	2A	00101010	*	Asterisk
43	053	2B	00101011	+	Plus
44	054	2C	00101100	,	Comma
45	055	2D	00101101	-	Hyphen
46	056	2^E	00101110	.	Period, dot or full stop
47	057	2F	00101111	/	Slash or divide
48	060	30	00110000	0	Zero
49	061	31	00110001	1	One
50	062	32	00110010	2	Two
51	063	33	00110011	3	Three
52	064	34	00110100	4	Four
53	065	35	00110101	5	Five
54	066	36	00110110	6	Six
55	067	37	00110111	7	Seven
56	070	38	00111000	8	Eight
57	071	39	00111001	9	Nine
58	072	3A	00111010	:	Colon
59	073	3B	00111011	;	Semicolon
60	074	3C	00111100	<	Less than (or open angled bracket)
61	075	3D	00111101	=	Equals
62	076	3^E	00111110	>	Greater than (or close angled bracket)
63	077	3F	00111111	?	Question mark
64	100	40	01000000	@	At symbol
65	101	41	01000001	A	Uppercase A
66	102	42	01000010	B	Uppercase B
67	103	43	01000011	C	Uppercase C
68	104	44	01000100	D	Uppercase D
69	105	45	01000101	E	Uppercase E
70	106	46	01000110	F	Uppercase F
71	107	47	01000111	G	Uppercase G
72	110	48	01001000	H	Uppercase H
73	111	49	01001001	I	Uppercase I
74	112	4A	01001010	J	Uppercase J
75	113	4B	01001011	K	Uppercase K
76	114	4C	01001100	L	Uppercase L
77	115	4D	01001101	M	Uppercase M
78	116	4^E	01001110	N	Uppercase N
79	117	4F	01001111	O	Uppercase O
80	120	50	01010000	P	Uppercase P
81	121	51	01010001	Q	Uppercase Q
82	122	52	01010010	R	Uppercase R
83	123	53	01010011	S	Uppercase S
84	124	54	01010100	T	Uppercase T
85	125	55	01010101	U	Uppercase U
86	126	56	01010110	V	Uppercase V
87	127	57	01010111	W	Uppercase W
88	130	58	01011000	X	Uppercase X
89	131	59	01011001	Y	Uppercase Y

90	132	5A	01011010	Z	Uppercase Z
91	133	5B	01011011	[	Opening bracket
92	134	5C	01011100	\	Backslash
93	135	5D	01011101	]	Closing bracket
94	136	5^E	01011110	^	Caret - circumflex
95	137	5F	01011111	_	Underscore
96	140	60	01100000	`	Grave accent
97	141	61	01100001	a	Lowercase a
98	142	62	01100010	b	Lowercase b
99	143	63	01100011	c	Lowercase c
100	144	64	01100100	d	Lowercase d
101	145	65	01100101	e	Lowercase e
102	146	66	01100110	f	Lowercase f
103	147	67	01100111	g	Lowercase g
104	150	68	01101000	h	Lowercase h
105	151	69	01101001	i	Lowercase i
106	152	6A	01101010	j	Lowercase j
107	153	6B	01101011	k	Lowercase k
108	154	6C	01101100	l	Lowercase l
109	155	6D	01101101	m	Lowercase m
110	156	6^E	01101110	n	Lowercase n
111	157	6F	01101111	o	Lowercase o
112	160	70	01110000	p	Lowercase p
113	161	71	01110001	q	Lowercase q
114	162	72	01110010	r	Lowercase r
115	163	73	01110011	s	Lowercase s
116	164	74	01110100	t	Lowercase t
117	165	75	01110101	u	Lowercase u
118	166	76	01110110	v	Lowercase v
119	167	77	01110111	w	Lowercase w
120	170	78	01111000	x	Lowercase x
121	171	79	01111001	y	Lowercase y
122	172	7A	01111010	z	Lowercase z
123	173	7B	01111011	{	Opening brace
124	174	7C	01111100	|	Vertical bar
125	175	7D	01111101	}	Closing brace
126	176	7^E	01111110	~	Equivalency sign - tilde
127	177	7F	01111111		Delete

1.8 Représentation des nombres signés

Plusieurs représentations des nombres entiers signés en binaire:

1. Signe-amplitude
2. Complément à 1
3. Complément à 2^n

1.8.1 Signe-amplitude

Le bit de poids le plus fort (MSB) indique le signe: 0 pour positif, 1 pour négatif, Les bits restants désignent la valeur absolue du nombre.

Avec *n* bits on peut représenter des entiers entre: $-(2^{n-1}-1)$ et $+(2^{n-1}-1)$

Exemple : n=4:

+5= | 0 | 1 | 0 | 1 |

-5= | 1 | 1 | 0 | 1 |

signe	amplitude

Exemples d'opérations arithmétiques (avec n=4):

résultat faux (0): dépassement de capacité

la soustraction doit être traitée comme une addition: 5-3= 5+(-3)

```
+ 5 :   0 1 0 1
+3 :    0 0 1 1
  8 :   1 0 0 0
```

Remarque : Le problème de cette représentation est l'existence de deux représentations distinctes pour le zéro (+0) et (-0).

1.8.2 Complément à 1

Le complément à 1 est obtenu en inversant (complémentant) bit par bit le nombre en question, de ce fait, la somme du nombre et de son complément sera égale à 2^n-1.

C1(A) = not (A).

Exemple:

Le complément à 1 de $(1001100)_2$ est $(00110011)_2$

```
  1 1 0 0 1 1 0 0
+ 0 0 1 1 0 0 1 1
  1 1 1 1 1 1 1 1        n=8 ⇒ 255= 2^8-1
```

Remarque : cette représentation n'est pas utilisée, puisque le même problème du zéro persiste.

1.8.3 Complément à 2

La représentation en complément à 2^n d'un nombre négatif –A, s'obtient en calculant le complément à 2^n de +A, soit : C2(A) = 2^n – A. On peut aussi calculer le complément à 2^n d'un nombre en inversant chacun de ses *n* bits, puis en ajoutant 1 au résultat.

Exemple:

```
+ 18 :        0 1 0 0 1 0
C1(18) :      1 0 1 1 0 1
+1 :                    1
C2(18) :      1 0 1 1 1 0
```

Maintenant, pour calculer la soustraction en complément à 2 de 25-18 en binaire, il est équivalent à calculer [25+C2(18)]

```
Retenue :   1 1 1     on écarte la dernière retenue
+ 25 :        0 1 1 0 0 1
+ C2(18) :    1 0 1 1 1 0
+ 7 :         0 0 0 1 1 1     ⇒ Résultat signé +7     (correct)
```