Chapitre 1

Numérations

0. Introduction *(Wikipedia)*

→ principe de représentation des nombres (écrit, oral, gestuelle, informatique)

→ s'appuie en général sur quelques symboles de valeurs qui forment une base et des règles de combinaisons

Il existe deux grands principes de numération :

1. systèmes additifs (votes, numération égyptienne ou romaine)
2. notation positionnelle (numération arabe ou codage informatique).

I. Le système décimal

→ nos nombres basés sur dix « signes » : $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

→ pourquoi ?

Exemple : Soit un nombre décimal $N = 2348$.
Ce nombre est la somme de 8 unités, 4 dizaines, 3 centaines et 2 milliers.
Nous pouvons écrire :

$$N = (2 \times 1000) + (3 \times 100) + (4 \times 10) + (8 \times 1)$$

$$2348 = \left(2 \times 10^{3}\right) + \left(3 \times 10^{2}\right) + \left(4 \times 10^{1}\right) + \left(8 \times 10^{0}\right)$$

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.

II. Le système binaire

→ très utile en automatisme, de l'électronique et de l'informatique

→ Tous les nombres s'écrivent avec seulement deux chiffres (0 et 1→ base 2)

De même que nous utilisons le système décimal parce que nous avons commencé à compter avec nos dix doigts, nous utilisons le binaire car les systèmes technologiques ont souvent deux états stables.

- Un interrupteur est ouvert ou fermé
- Une diode est allumée ou éteinte
- Une tension est présente ou absente
- Une surface est réfléchissante ou pas (CD)
- Un champ magnétique est orienté Nord-Sud ou Sud-Nord (disque dur)

A chaque état du système technologique, on associe un état logique binaire.
La présence d'une tension sera par exemple notée 1 et l'absence 0.
Le chiffre binaire qui peut prendre ces deux états est nommé "Bit" (Binary digIT)

II. Le système binaire

Avec un bit nous pouvons coder deux états

0
1

Avec deux bits nous pouvons coder quatre états

0	0
0	1
1	0
1	1

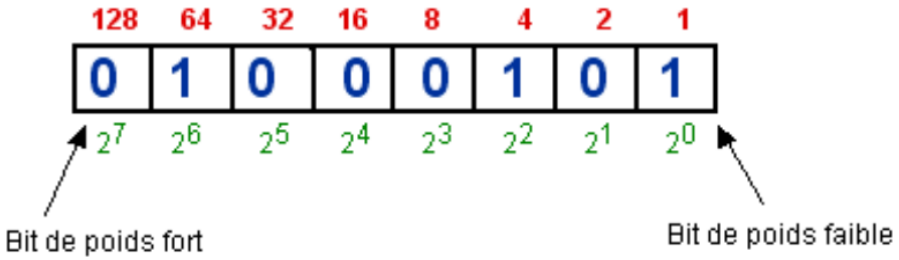
Avec trois bits nous pouvons coder huit états

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

A chaque nouveau bit, le nombre de combinaisons possibles est doublé.
Un groupe de bits est appelé un mot, un mot de huit bits est nommé un octet (byte).
Avec un octet, nous pouvons écrire $2^8 = 256$ nombres binaires de 0 à 255.
Les règles sont les mêmes que pour le décimal.

II. Le système binaire : correspondance binaire décimal

Exemple : Le nombre ou mot $0100\ 0101_{(2)}$ sera décrit comme suit :



→ Il suffit de faire la somme des poids de chaque bit à 1

$$01000101_{(2)} = (1 \times 2^6) + (1 \times 2^2) + (1 \times 2^0)$$
$$01000101_{(2)} = 64 + 4 + 1$$
$$01000101_{(2)} = 69_{(10)}$$

II. Le système binaire : correspondance décimal binaire

Exemple : Le nombre $172_{(10)}$ sera converti en binaire selon deux méthodes :

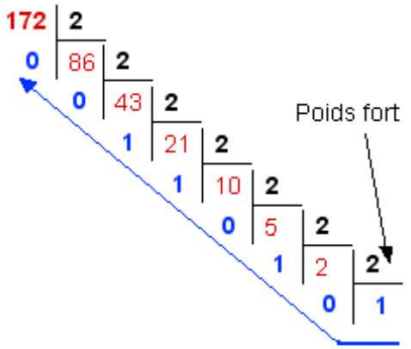
1. Méthode par soustractions :

172	44	12	4
- 128	- 32	- 8	- 4
44	12	4	0

172 = 128 + 32 + 8 + 4

$172_{(10)} = 10101100_{(2)}$

2. Méthode par divisions :



III. Le système hexadécimal

→ utilisé notamment en électronique numérique et en informatique car très commode

→ permet un compromis entre le code binaire des machines et une base de numération pratique à utiliser pour les ingénieurs.

→ chaque chiffre hexadécimal correspond exactement à quatre bit (conversions simples et écriture plus compacte).

Pour écrire les nombres en base 16 nous devons disposer de 16 chiffres, pour les dix premiers, nous utilisons les chiffres de la base 10, pour les suivant nous utiliserons des lettres de l'alphabet.

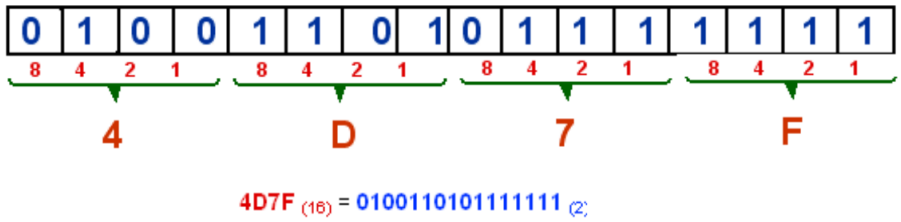
Déci	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexa	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Les règles sont ici aussi les mêmes que pour le décimal

III. Le système hexadécimal : conversions

Exemples :

- hexadécimal vers décimal :
  $A3F_{(16)} = (A \times 16^2) + (3 \times 16^1) + (F \times 16^0)$
  $A3F_{(16)} = 10 \times 256 + 3 \times 16 + 15 \times 1$
  $A3F_{(16)} = 2560 + 48 + 15 = 2623_{(10)}$
- binaire vers hexadécimal : très simple, c'est d'ailleurs la raison pour laquelle nous utilisons cette base.



- décimal vers hexadécimal (méthode par division)

IV. Opérations arithmétiques et logiques

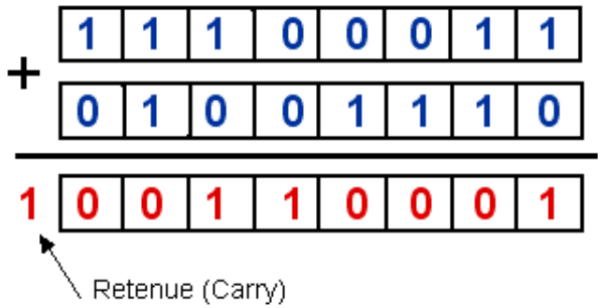
1. Addition en binaire

   L'addition est réalisée bit à bit.

   $1 + 0 = 1$

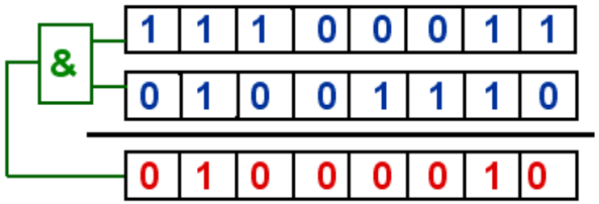
   $1 + 1 = 10$

   $1 + 1 + 1 = 11$



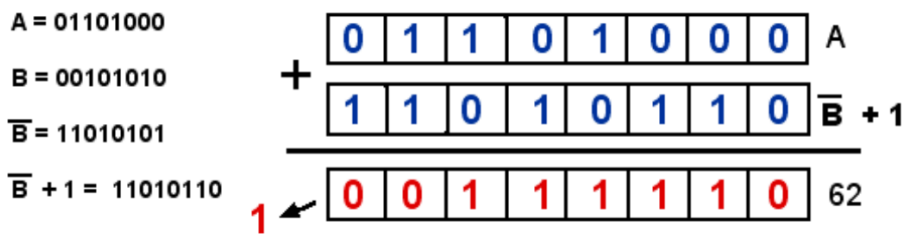
2. Produit logique en binaire

   La fonction ET est appliquée bit à bit



3. Les nombres signés
   En binaire, le négatif d'un nombre est son complément à 2, c'est à dire son complément + 1.
   Soient deux nombres $A = 104$ et $B = 42$.

$$A - B = A + (-B) = A + \overline{B} + 1$$



Le format est sur 8 bits, il ne faut ignorer le bit de dépassement à gauche.

Le premier bit est à 0 pour les nombres négatifs et à 1 pour les nombres positifs.

Conséquence :
Le plus grand nombre signé sur 8 bits est +127 (01111111)
Le plus petit nombre signé sur 8 bits est -128 (10000000)
-128 à +127 => 256 combinaisons

V. Application : codage UTF-8

→ UTF-8 (Universal Character Set Transformation Format 8 bits)

→ codage de caractères informatiques conçu pour coder l'ensemble des caractères du « répertoire universel de caractères codés »

→ totalement compatible avec le standard Unicode et ASCII.

- Le caractère A par exemple à pour code 65 soit 01000001 en binaire.
- Le caractère f : 102
- le point d'interrogation ? : 63
- Le chiffre 2 : 50

Exemples de codage UTF-8

Type	Caractère	Point de code (hexadécimal)	Valeur scalaire		Codage UTF-8	
			décimal	binaire	binaire	hexadécimal
Contrôle (C0)	[NUL]	U+0000	0	0	00000000	00
	[US]	U+001F	31	1·1111	00011111	1F
Texte (US-ASCII)	[SP]	U+0020	32	10·0000	00100000	20
	0	U+0030	48	11·0000	00110000	30
	9	U+0039	57	11·1001	00111001	39
	?	U+003F	63	11·1111	00111111	3F
	@	U+0040	64	100·0000	01000000	40
	A	U+0041	65	100·0001	01000001	41
	Z	U+005A	90	101·1010	01011010	5A
	a	U+0061	97	110·0001	01100001	61
	z	U+007A	122	111·1010	01111010	7A
	~	U+007E	126	111·1110	01111110	7E
Contrôle (C0 et C1)	[DEL]	U+007F	127	111·1111	01111111	7F
	[PAD]	U+0080	128	1000·0000	11000010 10000000	C2 80
	[APC]	U+009F	159	1001·1111	11000010 10011111	C2 9F
Texte (PMB)	[NBSP]	U+00A0	160	1010·0000	11000010 10100000	C2 A0
	¿	U+00BF	191	1011·1111	11000010 10111111	C2 BF
	À	U+00C0	192	1100·0000	11000011 10000000	C3 80
	é	U+00E9	233	1110·1001	11000011 10101001	C3 A9
	?	U+07FF	2047	111 1111·1111	11011111 10111111	DF BF
	?	U+0800	2048	1000 0000·0000	11100000 10100000 10000000	E0 A0 80
	€	U+20AC	8 364	10·0000 1010·1100	11100010 10000010 10101100	E2 82 AC
	?	U+D7FF	55 295	1101·0111 1111·1111	11101101 10011111 10111111	ED 9F BF

VI. Binaire réfléchi (Code de Gray)

→ Le code Gray ou "binaire réfléchi" permet de coder une valeur numérique en cours d'évolution en une suite de configurations binaires se différenciant l'une de l'autre par le changement d'état d'un seul bit à la fois.

→ Ce code permet par exemple d'éviter l'aléa suivant : En binaire pur, le passage de la valeur 0111 à 1000 peut engendrer des valeurs aléatoires comprises entre 0 et 1000, les bits ne changeant pas de valeur de manière parfaitement simultanée.

→ Le code binaire réfléchi est utilisé pour simplifier des équations dans les tableaux de Karnaugh en automatismes

Binaire pur				
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Binaire réfléchi			
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0