Chapitre 1

Systèmes de numération et codage

Objectifs :

- ✓ **Connaitre les différents systèmes de numération.**
- ✓ **Apprendre les règles de passage d'un système de numération à un autre.**
- ✓ **Etudier les codes numériques usuels.**

1. Les systèmes de numération :

Le traitement d'une information numérique, par un circuit électronique, nécessite qu'elle soit mise sous forme adaptée à celui-ci. C'est-à-dire qu'elle soit représentée dans un système de numération (ou base) adéquat.

De nombreux systèmes de numération sont utilisés en technologie numérique. Les plus courants sont les systèmes : décimal (base 10), binaire (base 2). Les systèmes : hexadécimal (base 16) et octal (base 8) sont également utilisés dans les domaines de l'électronique et l'informatique.

1.1 Représentation polynomiale :

La décomposition d'un nombre N en fonction des puissances de son système de numération (sa base B) s'appelle sa forme polynomiale :

$$N = a_n B^n + a_{n-1} B^{n-1} + \ldots + a_1 B^1 + a_0 B^0$$

Avec :

- a_i : sont appelés coefficients : $0 <= a_n < B$
- i : est appelé rang de a_i
- B^i : est appelé le poids de a_i

1.2 Le système décimal (base 10) :

Il comprend 10 chiffres ou symboles qui sont 0,1,2,3,4,5,6,7,8 et 9.

Exemples :

- $(6327)_{10} = 6 \times 10^3 + 3 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$
- $(245,16)_{10} = 2 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 + 1 \times 10^{-1} + 6 \times 10^{-2}$

1.3 Le système binaire (base 2) :

Les nombres exprimés dans cette base sont appelés nombres binaires, ils ne composent que des deux symboles ou chiffres 0 et 1.

Exemples :

- $(11101)_2 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
- $(1011,101)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 + 2^{-2} + 1 \times {}^{-3}$

Dans le système binaire, l'expression chiffre binaire est abrégée en bit. Le bit de poids le plus fort d'un nombre est celui le plus à gauche, tandis que le bit de poids le plus faible est celui le plus à droite.

1.4 Le système octal (base 8) :

Le système hexadécimal ou base 8 contient huit éléments ou symboles qui sont : 0, 1, 2, 3, 4, 5, 6, 7.

Exemples :

- $(760)_8 = 7 \times 8^2 + 6 \times 8^1 + 0 \times 8^0$
- $(425,31)_8 = 4 \times 8^2 + 4 \times 8^1 + 4 \times 8^0 + 3 \times 8^{-1} + 1 \times 8^{-2}$

1.5 Le système hexadécimal (base 16) :

Le système hexadécimal ou base 16 contient seize éléments ou symboles qui sont : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Les lettres A, B, C, D, E et F représentent respectivement les valeurs 10, 11, 12, 13, 14 et 15.

Exemples :

- $(428)_{16} = 4 \times 16^2 + 2 \times 16^1 + 8 \times 16^0$
- $(1A5E)_{16} = 1 \times 16^3 + 10 \times 16^2 + 5 \times 16^1 + 14 \times 16^0$
- $(B7,6D)_{16} = 11 \times 16^1 + 7 \times 16^0 + 6 \times 16^{-1} + 13 \times 16^{-2}$

2. Changement de base :

Il s'agit de convertir un nombre écrit dans une base B_1 à son équivalent dans une autre base B_2.

2.1 Conversion d'un nombre N de base B en un nombre décimal :

Tout nombre écrit dans une base B peut être transformé en son équivalent décimal en utilisant sa forme polynomiale.

Exemples :

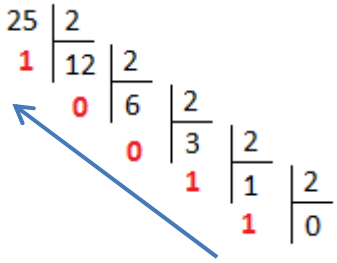
- Binaire - Décimal : $(1\ 1\ 1\ 0\ 1\ 1)_2 = 1x2^5 + 1x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = (59)_{10}$
- Octal - Décimal : $(1503)_8 = 1 \times 8^3 + 5 \times 8^2 + 0 \times 8^1 + 3 \times 8^0 = (835)_{10}$
- Hexadécimal - Décimal : $(2BF)_{16} = 2 \times 16^2 + 11 \times 16^1 + 15 \times 16^0 = (703)_{10}$

2.2 Conversion d'un nombre décimal N en un nombre de base B :

Cette conversion consiste à faire des divisions entières successives par la base Bet conserver à chaque fois le reste de la division jusqu'à ce que le quotient soit 0. Le résultat recherché s'obtient en écrivant le premier reste à la position du bit de poids le plus faible (premier chiffre à gauche) et le dernier reste à la position du bit de poids le plus fort (premier chiffre à droite).

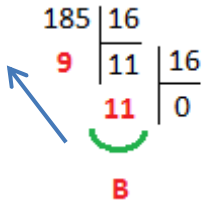
Exemples :

$(25)_{10} = (?)_2$



$(25)_{10} = (11001)_2$

$(185)_{10} = (?)_{16}$



$(185)_{10} = (B9)_{16}$

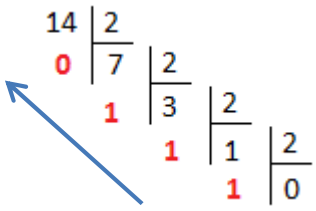
Remarque : Si le nombre décimal à convertir comporte une partie fractionnaire (à virgule), on procède sa conversion vers une autre base B en deux phases :

- conversion de la partie entière en effectuant des divisions successives par la base B comme précédemment.
- Conversion de la partie fractionnaire en effectuant des multiplications successives par B et en conservant à chaque fois le chiffre qui devient entier.

Exemple :

- **$(14,875)_{10} = (?)_2$**

Conversion de la partie entière



$(14)_{10} = (1110)_2$

Conversion de la partie fractionnaire

0,875 * 2 = **1**,75

0,75 * 2 = **1**,5

0,5 * 2 = **1**,0

$(0,875)_{10} = (0,111)_2$

$(14,875)_{10} = (1110,111)_2$

2.3 Conversion d'un nombre d'une base B_1 vers une autre base B_2:

Dans ce cas il faut passer de la base B_1 vers la base décimale (10) puis de la base 10 vers la base B_2. Si les bases B1 et B2 sont des puissances de 2, il est possible de passer par la base binaire (2) de la façon suivante : si $B_1=2^n$il faut convertir chaque chiffre de la base B_1 en binaire sur n bits :

- Base tétrale (4) : $4 =2^2$ chaque chiffre tétral sera convertit en binaire sur 2 bits.
- Base octale (8) : $8 =2^3$ chaque chiffre octal sera convertit en binaire sur 3 bits.
- Base hexadécimale (16) : $16 =2^4$ chaque chiffre hexadécimal sera convertit sur 4 bits.

Exemples :

- $(120)_3 = (?)_2$
  - ✓ Base 3 → Base 10 : $(120)_3 = 1 \times 3^2 + 2 \times 3^1 + 0 \times 3^0 = (15)_{10}$
  - ✓ Base 10 → Base 2 : $(15)_{10} = (1111)_2$

15 | 2		
1 | 7	2	
	1 | 3	2
		1 | 1 | 2
		1 | 0

donc : $(120)_3 = (1111)_2$

- $(213)_4 = (?)_8$
  - ✓ Base 4 → Base 2 : $(213)_4 = (\underline{100111})_2 = (100111)_2$
  - ✓ Base 2 → Base 8 : $(\underline{100111})_2 = (47)_8$

donc : $(213)_4 = (47)_8$

- $(B67)_{16} = (?)_4$
  - ✓ Base 16 → Base 2 : $(B67)_{16} = (\underline{101101100111})_2 = (101101100111)_2$
  - ✓ Base 2 → Base 4 : $(\underline{101101100111})_2 = (2\ 3\ 1\ 2\ 1\ 3)_4$

donc : $(B67)_{16} = (231213)_4$

- $(1110100110)_2 = (?)_{16}$
  - ✓ Base 2 → Base 16 : $(1110100110)_2 = (\underline{001110100110})_2 = (3A6)_{16}$

donc : $(1110100110)_2 = (3A6)_{16}$

3. Les codes :

Un code est une correspondance arbitraire entre un ensemble de *symboles* et un ensemble d'*objets*.

Les objets sont les lettres de l'alphabet, les chiffres, les signes de ponctuation, etc.Les symboles sont des combinaisons de « 0 » et de « 1 » représentés physiquement par les deux états stables des circuits utilisés dans les systèmes de traitement de l'information: les codes sont donc binaires.*Le code adapte le langage humain au langage de la machine et inversement.*

3.1 Les codes binaires :

Ils codent tous les nombres entiers.

3.1.1 Le code binaire naturel :

C'est une représentation numérique des nombres entiers dans la base 2. Le tableau ci-dessous représente le code binaire naturel relatif aux entiers de 0 à 15.

Décimal	Code binaire naturel			
	a_3	a_2	a_1	a_0
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

3.1.2 Le code binaire réfléchi (code GRAY) :

La propriété principale de ce code est que le passage d'un mot-code (ligne) au suivant entrainera toujours le changement d'un seul bit.

Décimal	Code binaire réfléchi			
	a_3	a_2	a_1	a_0
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
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

3.2 Les codes décimaux :

Les codes décimaux sont utilisés pour la représentation des chiffres 0 à 9 du système décimal. Ils contiennent par conséquent dix mots-code.

3.2.1 Le code DCB (Décimal codé Binaire) :

Le DCB (BCD en anglais) est le code le plus utilisé en électronique. Il contient des mots-code qui sont la traduction en binaire naturel (sur 4 bits) des dix chiffres du système décimal.

Décimal	Code DCB			
	a_3	a_2	a_1	a_0
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

Exemple de codage en DCB

2016 en décimal

devient : ↓ ↓ ↓ ↓

0010 0000 0001 0110 en DCB

0101 1001 0111 1000 en DCB

devient : ↓ ↓ ↓ ↓

5 9 7 8 en décimal

3.2.2 Le code à excès de 3 (ou code de STIBITZ) :

Le code à excès de 3 (noté XS 3) s'obtient en ajoutant 3 à chaque mot-code du code DCB. Il a été créé pour permettre la réalisation simple des opérations de soustraction.

Décimal	Code à excès de 3			
	a_3	a_2	a_1	a_0
0	0	0	1	1
1	0	1	0	0
2	0	1	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

Exemple de codage en XS 3

2 0 1 6 en décimal

devient :

0101001101001001 en XS 3

0101 1001 0111 1000 en XS 3

devient :

2 6 4 5 en décimal

3.2.3 Les codes «2 parmi 5 » :

Les codes 2 parmi 5 font partie des codes spécialement conçus pour la transmission de l'information et pour la détection des erreurs. En effet, si on reçoit un nombre code en 2 parmi 5, pour détecter une éventuelle erreur dans ce nombre il suffit de compter le nombre de 1 logiques présents dans chacun des groupes de 5 bits. Si un groupe ne présente pas deux 1 logiques, on peut en déduire avec certitude qu'il est erroné.

Décimal	Codes2 parmi 5									
	a_4	a_3	a_2	a_1	a_0	a_4	a_3	a_2	a_1	a_0
	8	4	2	1	0	7	4	2	1	0
0	1	0	1	0	0	1	1	0	0	0
1	0	0	0	1	1	0	0	0	1	1
2	0	0	1	0	1	0	0	1	0	1
3	0	0	1	1	0	0	0	1	1	0
4	0	1	0	0	1	0	1	0	0	1
5	0	1	0	1	0	0	1	0	1	0
6	0	1	1	0	0	0	1	1	0	0
7	1	1	0	0	0	1	0	0	0	1
8	1	0	0	0	1	1	0	0	1	0
9	1	0	0	1	0	1	0	1	0	0

3.3 Les codes alphanumériques :

L'ordinateur doit reconnaître des codes qui correspondent à des nombres, des lettres, des signes de ponctuation et des caractères spéciaux. Les codes de ce genre sont dit alphanumériques. Un ensemble de caractères complet doit renfermer les 26 lettres minuscules, les 26 lettres majuscules et les dix chiffres les 7 signes de ponctuation et les caractères spéciaux.

Le code **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) est le code alphanumérique le plus répandu.

C'est un code de 7 bits ce qui permet de générer (coder) 2^7=128 caractères.

Exemple :

Le message codé en ASCII = $(1000001\ 1001001\ 1000100\ 1000101)_2$

↓↓↓↓

En hexadécimal =$(\ 41\ 49\ 44\ 45)_H$

↓↓↓↓

Résultat = AIDE

4. L'arithmétique binaire :

4.1. Les quatre opérations de base :

Les mêmes règles de calcul s'appliquent dans tous les systèmes de numération :

Addition			Soustraction			Multiplication		Division		
Opération	Résultat	Retenue	Opération	Résultat	Retenue	Opération	Résultat	Opération	Résultat	Reste
0+0	0	0	0-0	0	0	0*0	0	0:0	imp	
0+1	1	0	0-1	1	1	0*1	0	0:1	0	1
1+0	1	0	1-0	1	0	1*0	0	1:0	imp	
1+1	0	1	1-1	0	0	1*1	1	1:1	1	0

4.1.1 Les nombres négatifs :

- *Représentation avec la valeur absolue et le signe :*

C 'est le bit de poids le plus fort qui représente le signe du mot traité : 0 : +, 1 : -

Exemple :

+35 = 0 100011

- 35 = 1 100011

- *Représentation par le complément restreint : (complément à 1)*

On obtient le complément restreint noté (CR(x)=C1(X)) d'un nombre X en remplaçant ses 0 par des 1 et ses 1 par des 0.

-X = C1(X)

Exemple :

+1 = 00000001

-1 = 11111110

+0=00000000

-0 = 11111111

- *Représentation par le complément vrai : (complément à 2)*

On obtient le complément à 2 en ajoutant 1 au complément à 1. Elle simplifie l'addition et la soustraction, en outre on a une seule représentation du 0.

-X = C2(X)= CR(X) + 1

Exemples :

+2 = 00000010

+1 = 00000001

+0 = 00000000

-0 = 00000000

-1 = 11111111

-2 = 11111110

4.1.2 Addition de deux nombres positifs :

Exemples :

A/ 7 + 5 = 12 00111 + 00101 = 01100

B/ En ajoutant deux nombres >0 (bit de signe = 0) on peut obtenir un résultat dont le bit de signe est 1, bien que le résultat est positif.

49 + 33 = 82 00110001 + 00100001 = 01010010

On remarque que le 8ème bit est à 1 : donc il y a un dépassement de capacité.

4.1.3 Addition des entiers signés :

Elle peut se faire soit en appliquant les règles énoncés au-dessus lorsque A >= B.

Si A < B, on utilise la méthode du complément à 2 :

Calculer la différence A – B revient à calculer la somme A + C2(B).

Sachant que :

- C2(B) = C1(B) +1
- C1(B) se calcule en remplaçant tous les 1 par 0 et inversement.

Exemple 1 :

A – B = 7 - 1 4 = -7

$(14)_{10}= (1110)_2$

C1(1110) = 0001

C2(1110) = C1(1110) + 1 = 0010

A – B revient à A + C2(B) = 0111 + 0010 = 1001

On remarque que la dernière retenue de cette somme est égale à zéro, la différence est alors négative et elle est égale au complément à 2 du résultat obtenue.

A – B = C2(1001) = 0110 + 1 = 0111 = $(-7)_{10}$

Exemple 2 :

A – B = 13 - 9 = 4

$(9)_{10}= (1001)_2$

C1(1001) = 0110

C2(1001) = C1(1001) + 1 = 0111

A – B revient à A + C2(B) = 1101 + 0111 = 10100

On remarque que la dernière retenue de cette somme est égale à un, la différence est alors positive et elle est égale au résultat obtenue en éliminant le dernier un à gauche.

A – B = 0100 = $(4)_{10}$