الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE DES FRERES MENTOURI CONSTANTINE 1
Institut Des Sciences Et Techniques Appliquées

جامعة الاخوة منتوري قسنطينة 1
معهد العلوم و التقنيات التطبيقية

Chapitre II
Systèmes de numération et codage

SYSTEMES DE NUMERATION ET CODAGE

I. Introduction

Le traitement de l'information (Image, mot, nombre,…etc.) est effectué par l'ordinateur ou calculateur électronique. Les circuits électroniques qui les constituent ne peuvent prendre que deux états représentés par **0** et **1**. Donc, le langage utilisé dans ces machines est appelé système de numération binaire **(suite de 0 et de 1)**.

II. Différents systèmes de numération

Il existe quatre systèmes de numération qui sont :

- Système décimal ;
- Système binaire ;
- Système octal ;
- Système hexadécimal.

II.1. Système décimal

C'est un système utilisé dans le monde extérieur, les chiffres utilisés sont des entiers composés par des valeurs de 0 à 9. La base de ce système est 10.

Exemple

1995 : ce nombre s'écrit dans le système **à** base 10 sous la forme :

$$1* 10^3 + 9 * 10^2 + 9 * 10^1 + 5 * 10°$$

II.2. Système binaire

C'est le système qui est utilisé par la machine (l'ordinateur), il est composé seulement par des suites binaires **(1 et 0)**. La base de ce système est 2.

Exemple

1001101 : la valeur décimale de cette suite est **:**

$$1* 2^6 + 0 * 2^5 + 0 * 2^4 + 1* 2^3 + 1* \mathbf{2^2} + 0 * \mathbf{2'} + 1*2° = 77$$

Chaque chiffre binaire **(0 ou 1)** est appelé bit qu'on note par b, et toutes combinaisons binaires composées de huit bits est appelée Octet.

Example: 11100100

$$B_0 = 0, b_1= 0, b_2=l, b_3= 0, b_4=0, b_5= 1, b_6=l, b_7= 1$$

Seize (16) bits est appelée « mot-mémoire » ou bien mot machine

Example: 1110001010100110

$$B_0 = 0\ b_1= 1\ b_2 = 1\ b_3 = 0\ b_4= 0\ b_5= 1 \ldots\ldots\ldots b_{14}=l\ b_{15}= l$$

II.3. Système octal

Le système décimal est celui qui est le plus utilisé par l'homme, et le système binaire est celui le plus utilisé dans les ordinateurs.

Le système binaire présente l'inconvénient d'être difficile à manipuler par l'homme car les nombres sont représentés par une suite de 0 et 1 qu'il est difficile de lire ou d'interpréter sans erreur. Il est donc plus commode d'écrire les nombres codés en binaire sous une forme plus compacte, dans un système dont la base est une puissance de 2, de façon à permettre une conversion facile avec le système binaire.

Nous définissons dans ce paragraphe le système octal, et plus loin le système hexadécimal.

Certains calculateurs numériques utilisent le système octal qui est un système de numération dont la base 8.

On dispose de huit symboles pour représenter un nombre dans le système octal. Ce sont les chiffres successifs allant de 0 à 7. Chaque chiffre d'un nombre octal a une pondération égale à la valeur du chiffre multipliée par la puissance de 8 correspondant au rang qu'il occupe dans le nombre.

Exemple : $N = 536_8$

- Chiffre 5 3 6
- Puissance 8^2 8^1 8^0
- Pondération 320 24 6

L'addition des pondérations donne l'équivalent décimal du nombre octal considéré.

320+24+6 = 350

Soit $536_8 \Rightarrow 350_{10}$

II.4. Système hexadécimal

Ce système est utilisé sur la plupart des nouveaux calculateurs numériques. Sa base est égale à 16. On dispose de 16 symboles pour représenter un nombre hexadécimal. Ces symboles sont les ***10 digits*** du système décimal auxquels on a ajouté les 6 premières lettres de l'alphabet. Les 16 symboles sont alors :

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

Chaque signe d'un nombre hexadécimal a une pondération qui s'obtient en multiplient la valeur numérique du symbole par la puissance de 16 correspondant au rang qu'il occupe dans le nombre.

Exemple : $N = 4CA2_{16}$

-	Signes	4	C	A	2
-	Puissance	16^3	16^2	16^1	16^0
-	Pondérations	16384	3072	160	2

L'addition des pondérations donne l'équivalent décimal du nombre hexadécimal considéré :

16384+3072+160+2 = 19618

Soit $4AC2_{16} \Rightarrow 19618_{10}$

L'intérêt de ces deux systèmes vient de ce que **8 et 16** ont des **puissances entières de** groupe de **3 et 4 bits** et par conséquent, on peut **effectuer les conversions** Octale/Décimale ainsi que Hexadécimale/Décimale. **(Voir Tableau) :**

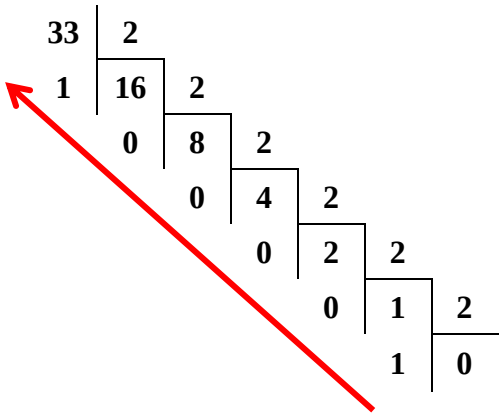
Nombre décimal	Binaire	octal	Hexadécimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	10001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14

III. Transformations (conversions)

III.1 Décimal / Binaire ,

Pour convertir un nombre décimal en un nombre binaire, diviser continûment le nombre décimal par la valeur 2, retenir la suite des restes de chaque division en commençant de bas en haut pour avoir le nombre équivalent.

Exemple :



$(33)_{10} = (100001)_2$

Dans le cas où le nombre donné dispose d'une partie fractionnaire, l'opération de conversion se fait de la manière suivante :

- Multiplier la partie, fractionnaire du nombre donné par 2 et retenir la partie entière du résultat trouvé.
- Continuer à multiplier successivement par la valeur 2 les parties fractionnaires obtenues pendant chaque opération tout en retenant leurs parties entières, jusqu'à ce qu'on obtienne une partie fractionnaire nulle ou déjà obtenue, sinon ce sera une suite binaire infinie et dans ce cas , on peut se limiter à quelques bits (problème d'arrondi).
- Le résultat de cette conversion est l'ensemble, des parties entières conservées à chaque étape de multiplication.

Exemple :

Transformer le numéro 23 en binaire ?
$(23)_{10} = (10111)_2$

Exemple :

Convertir le nombre (23.625) de la base 10 à la base 2.

La partie entière 23 est déjà déterminée, son résultat est : $(23)_{10} = (1011l)_2$.

Pour la partie fractionnaire 0.625, on procède comme suit :

$$0.625 * 2 = 1.250$$

$$0.250 * 2 = 0.500$$

$$0.500 * 2 = 1.00$$

On constate que la dernière partie fractionnaire est nulle, donc le résultat sera :

$$\mathbf{(0.625)_{10} = (0.101)_2}$$

Le résultat final : $(23.625)_{10} = (10111.101)_2$

Prenons le cas infini par exemple $(23.56)_{10}$

Pour $(23)_{10}$ c'est toujours (10111): mais pour la partie fractionnaire $(0.56)_{10}$, on aura :

$$0.56 *2=1.12$$

$0.12 * 2 = 0.24$

$0.24 * 2 = 0.48$

$0.48 * 2 = 0.96$

$0.96 * 2 = 1.92$

$0.92 * 2 = 1.84$ etc.,

On remarque, si on continue l'opération de multiplication de la partie fractionnaire par la valeur 2 , on ne tombe jamais sur le critère d'arrêt de cette opération , cela veut dire que la suite binaire est infinie et nous pouvons la tronquer en nous limitant à un certain nombre de bits .

D'où le résultat final de : $(23.56)_{10}$ est $(10111.100011\quad)_2$.

III.2. binaire/décimale

Pour convertir une suite binaire en système décimal, on procède de la manière suivante

Faisons l'addition de tous les termes de la forme (bit * 2), en commençant de droite à gauche pour les parties entières et de gauche à droite pour les parties fractionnaires.

« bit » : peut prendre les valeurs 0 ou 1. Le symbole " i " indique les puissances de 2 qui sont positives pour les parties entières et vont de 0 **à** (n-1) et négatives pour les parties fractionnaires et vont de la valeur (-1) jusqu'à (-m) avec n et m sont respectivement les longueurs de la suite binaire avant et après la virgule.

Exemple :

$(10110)_2 = 0*2° + 1 * 2^1 + 1 * 2^2 + 0 * 2^3 + 1 * 2^4 = (22)_{10}$

(10110 . 1011)2 = ?

- Pour la partie entière $(10110)_2 = (22)_{10}$
- Pour la partie fractionnaire :

$(0.1011)_2 = (1*2^1 + 0*2^2 + 1 * 2^3 + 1 * 2^4)$

Le résultat final :

$(10110.1011)_2 = (22.6875)_{10}$

III.3 Binaire /Octale - Binaire / Hexadécimale

La conversion d'une suite binaire en système octal et en système hexadécimal, se fait respectivement par le regroupement de trois et de quatre bits, en commençant de droite à gauche, chaque-bloc de bits sera convertible en système demandé

Exemple :

$(1010110111)_2 = (?)_8$ et $(?)_{16}$

En base 8 (regroupement par des blocs de trois bits):

$(111)_2 = 7$; $(110)_2 = 6$; $(010)_2=2$; $(001\ )_2 = 1$ D'où $(101011011l)_2 = (1267)_8$

- En base 16 (regroupement par des blocs de quatre bits), |

$(0111)_2 = 7$; $(1011)_2 = B$; $(0010)_2 = 2$ D'où (1010110111); $= (2B7)_{16}$

IV. Codage

Pour traiter les informations il existe deux codes numériques qui sont :

- Les codes numériques qui utilisent dans le codage des nombre.
- Les codes alphanumériques qui utilisent dans le traitement d'une information quelconque comme les lettres, les chiffres, les symboles, les images et les programmes…etc.

IV.1. Codes numériques

IV.1.1. Code binaire naturel

Le code binaire naturel est utilisé pour représenter un nombre dans la base de numération binaire.

- **Inconvénients du code binaire naturel:**
  - La longueur du nombre (nécessite une grande quantité de bits) ;
  - Les variables changent d'état entre deux combinaisons successives ;
  - Lors du codage ce type de système est non adapté à la correction automatique des erreurs.

IV.1.2. Code binaire réfléchi (Code GRAY)

Dans ce type de code un seul bit change de valeur entre deux codages successifs.

Il se compose de trois bits et quatre bits, comme l'indique sur les deux tableaux :

Code binaire Naturel…….	..Code binaire réfléchi
Sur 3 bits	
..... .000.....	..000
001.....	..001
010.....	..011
011.....	..010
100.....	..110
101.....	..111
110.....	..101
1 11.....	..100

Code binaire Naturel…….		..Code binaire réfléchi
Sur 4 bits		
0	0000.....	..0000
1	0001.....	0001
2	0010.....	..0011
3	0011.....	..0010
4	0100.....	..0110
5	0101.....	..0111
6	0110.....	..0101
7	0111.....	..0100
8	1000.....	..1100
9	1001.....	..1101
10	1010.....	..1111
11	**1011.....**	**..1110**
12	1100.....	..1010
13	1101.....	..1011
14	1110.....	..1001
15	1111.....	..1000

IV.1.3. Code DCB 8421 (Décimal Codé Binaire)

Dans ce code chaque chiffre décimal de 0 à 9 est représenté par 4 bits du binaire pur, avec dans l'ordre les poids 8, 4, 2, 1, c'est-à-dire $8=2^3$

Pour le bit N^0 3, $4=2^2$ pour le bit N^0 2, $2=2^1$ pour le bit $N^0$1, $1=2^0$.

Pour le bit N^0 0.

Exemple1 :

$1011_{DCB} = 1*8+0*4+1*2+1*1= 11_{10}$

$32_{10} = 00110010_{DCB}$

Exemple 2

$(1297)_{10} = (0001\ 0010\ 1001\ 0111)_{BCD}$

IV.1.4. Le code à excès de trois (ou code de STIBITZ)

Le code à excès de trois (Excess 3 noté XS 3 en abrégé) s'obtient en ajoutant 3 à chaque mot-code du code BCD. Tout comme le BCD, le code à excès de 3 est un code décimal, son tableau de conversion ne concerne donc que les chiffres de 0 à 9. Comme en BCD, pour coder un nombre en code à excès de trois, il faudra Concaténer une succession de tétrades, traduisant chacune un chiffre du nombre à coder.

Tableau de conversion
Décimal → Code à excès de 3

Décimal	Code à excès de 3
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

	1	9	8	2	en décimal
	↓	↓	↓	↓	
devient	0100	1100	1011	0101	en XS 3
	1000	0011	1010	0110	en XS 3
	↓	↓	↓	↓	
devient	5	0	7	3	en décimal

Propriété du code XS 3

Le code à excès de trois a été créé pour permettre la réalisation simple des opérations de soustraction. Le complément à 1 d'un mot-code représente le complément à 9 dans l'ensemble source : les codes possédant cette propriété sont appelés des codes auto-complémentaires.

Exemple avec le chiffre 7 :

En XS 3, le chiffre 7 se code 1010. En XS 3 le complément à 1 de 7 est donc 0101 (complément de chacun des bits). En décimal le complément à 9 de 7 est 2 (9-7=2). On remarque que 0101 est bien le mot-code de 2 en XS 3. Et cette propriété d'auto-complémentarité peut être vérifiée pour les 10 chiffres : **Le complément à 1 d'un mot-code en XS 3 correspond au complément à 9 du chiffre en décimal**.

Remarque : le code XS 3 est un code **auto-complémentaire**, mais il n'est pas pondéré.

IV.1.5. Le code Aïken

Le code Aïken regroupe les deux propriétés des codes BCD et XS 3 précédents : c'est un code décimal pondéré et auto-complémentaire. Les poids des éléments binaires sont **2 4 2 1.** La différence entre le code Aïken et le code BCD est le poids du premier bit à gauche : il valait 8 en BCD alors qu'il vaut 2 en code Aïken.

Tableau de conversion
Décimal → Code Aïken

Décimal	Code Aïken
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	1 0 1 1
6	1 1 0 0
7	1 1 0 1
8	1 1 1 0
9	1 1 1 1

Exemple de codage en code Aïken

	1	9	8	2	en décimal
	↓	↓	↓	↓	
devient	0001	1111	1110	0010	en Aïken
	1011	0100	0111	1100	en Aïken
	↓	↓	↓	↓	
devient	5	4	7	6	en décimal

:

Propriété d'auto-complémentarité du code Aïken :

Exemple avec le chiffre 8 :

- En décimal le complément à 9 de 8 est 1
- En Aïken le complément à 1 de 1110 (mot-code du 8) est 0001 (mot-code du 1) Et cette propriété d'auto-complémentarité peut être vérifiée pour les 10 chiffres : ***Le complément à 1 d'un mot-code en code Aïken correspond au complément à 9 du chiffre en décimal.***

IV.1.6. Les codes « 2 parmi 5 »

Avec les codes « 2 parmi 5 », les mots-code comprennent 5 bits dont 2 sont à 1 (et les 3 autres à 0). Il existe plusieurs codes « 2 parmi 5 », et les plus utilisés sont le code « 8 4 2 1 0 » et le code « 7 4 2 1 0 ». Ces deux codes sont pondérés, la liste des poids figurant dans la dénomination du code.

Tableau de conversion
Décimal → Code « 2 parmi 5 »

Décimal	Code « 2 parmi 5 » 8 4 2 1 0	Code « 2 parmi 5 » 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

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 codé 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.

Remarque : contrairement à d'autres codes plus perfectionnés (code de Hamming par exemple), les codes « 2parmi 5 » permettent de détecter une erreur, mais ne permettent pas de la corriger. De plus, si lors de la transmission, 2 bits de valeurs différentes changent simultanément d'état, aucune erreur ne pourra être détectée à l'arrivée.

Références et sites web

[1]***http://www***.gecif.net/articles/genie_electrique/cours/terminale/cours/les_differents_codes.pdf

[2] ***KHATORY*** : **Initiation informatique i (Système de numération) (1° GIM)** université sidi mohammed ben abdellah mr khatory Ecole Supérieure de Technologie de Fès Filière Génie Industriel et Maintenance.

[3] ***Nadia SOUAG***: logique combinatoire cours et exercices corrigés 'alger 2004

[4] ***L.Djeffal*** : application de l'algèbre de Boole aux circuits combinatoires et séquentiels '' cours et exercices avec solution''. Presses de l'université de batna 1996.

[5] ***Menacer Said ; Menacer Mohamed ; Menacer abderahmene*** **:** Électronique digitale Tome 1 analyse combinatoires et séquentielles aout 1990.