Cours Électronique numérique

Chapitre 1 : Systèmes de Numération et Codes

Année universitaire 2023/2024

Table des matières

I. Introduction

Généralement, on utilise le système décimal pour représenter les nombres, mais il est possible d'utiliser d'autres systèmes de numération. Nous nous intéressons dans ce chapitre aux systèmes de numération fréquemment rencontrés en technologie numérique. Il s'agit des systèmes binaire, octal, décimal et hexadécimal.

Les codes numériques ne traitent que des nombres binaires composés de **0** et de **1.** Les codes alphanumériques permettent l'exécution de fonctions spécifiques afin d'améliorer les performances du matériel numérique. En effet, le clavier d'un ordinateur porte des touches sur lesquelles sont indiquées les lettres de l'alphabet, les chiffres, les signes de ponctuation (Information source). Le nombre de fils conducteurs reliant l'unité centrale au clavier est nettement inférieur au nombre de ces touches.

Une simple analyse montre que l'information source est codée avant d'être transférée à l'unité de traitement sous forme d'information image.

Bienvenue dans le monde merveilleux de la logique et de l'électronique numérique, dans lequel on a : 1+1=1

Mais aussi : 1+1=10

II. Définition :

Un système de numération permet de coder une information en lui associant un symbole ou une combinaison de symboles qui permettent de la faire communiquer.

La connexion entre le clavier, l'unité centrale et l'écran peut être matérialisée par le schéma Synoptique suivant :

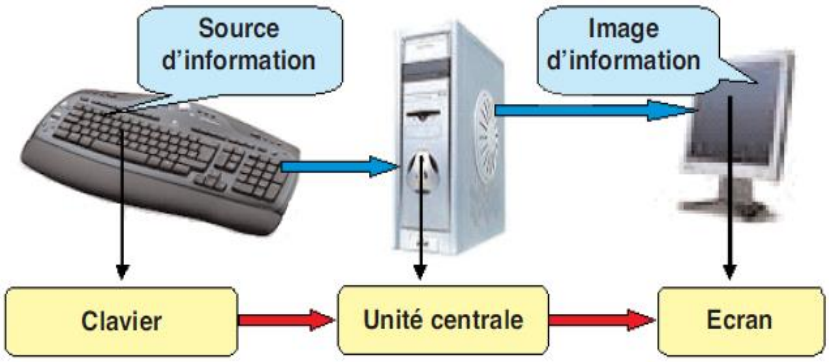


Figure 1. Schéma synoptique d'un système numérique

III. Les systèmes de numération

1. Principe

D'une façon générale, soit une base **B**, donc associée à B symboles : {S_0, S_1, …, S_{B-1}} ; un nombre **N**, a les caractéristiques suivantes :

- Il s'écrit $\mathbf{N = A_{n-1} \dots A_i \dots A_1A_0}$ avec $\mathbf{Ai \in \{S_0, S_1, \dots, S_{B-1}\}}$.
- Il a pour valeur $\mathbf{N = A_{n-1}.B^{n-1} + \dots + A_i.B^i + \dots + A_1.B^1 + A_0.B^0}$ (forme polynomiale).
- $\mathbf{A_i}$ est le chiffre (digit) de rang **i** et de poids $\mathbf{B^i}$.
- $\mathbf{A_{n-1}}$ est le chiffre **le plus significatif** (MSD (MSB): Most Significant Digit (Bit)).
- $\mathbf{A_0}$ est le chiffre **le moins significatif** (LSD (LSB): Less Significant Digit (Bit)).

On va étudier les **bases 2, 4, 8 et 16** pour leur intérêt dans les circuits logiques. La référence à la base **10** est d'un usage pratique, on étudiera alors la conversion de la base **2** vers la base **10** et vice versa.

2. Code Décimal Base (10)

Le système usuel de numération utilise le code décimal. Dans ce système, le nombre de symboles utilisés est **10** {**0, 1, 2, 3, 4, 5, 6, 7, 8, 9**}. Ces symboles s'appellent des chiffres. Le nombre (**10**) représente la base (**B**) du code décimal.

Ainsi le nombre décimal **2056** peut être représente sous la forme suivante : **2056 = 2000 + 0 + 50 + 6** ou :

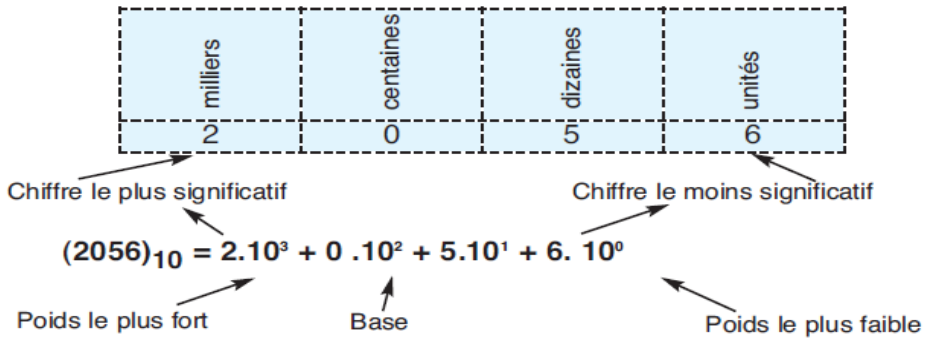


Figure 2. Représentation d'un nombre décimal

3. Code Binaire Base (2)

L'homme connaît la base 10; il fait alors ses calculs dans cette base. Alors, puisque les systèmes numériques ne reconnaissent que 2 états **0** ou **1**, ils seront très aptes de faire les calculs dans la base 2.

D'une façon générale un nombre binaire s'écrit comme suit :

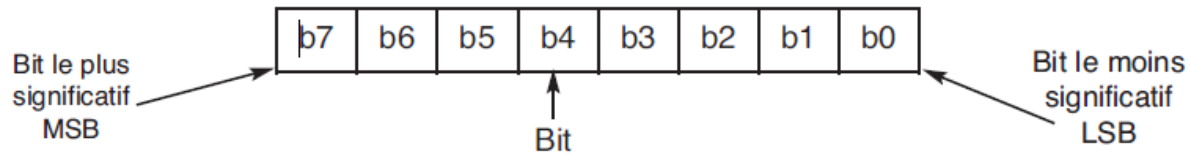


Figure 3. Représentation d'un mot binaire

- **<u>Mot binaire:</u>** Ce nombre $(b7\ b6\ b5\ b4\ b3\ b2\ b1\ b0)_2$ est généralement appelé mot binaire.

- **Poids ou pondération:** Coefficient attaché au rang d'un chiffre dans un système de numération.
- **LSB et MSB:** en numération binaire, on parle du **bit de plus faible poids (LSB)** qui est en position binaire la plus à droite dans un mot et du **bit de plus fort poids (MSB)** qui représente le bit situe le plus à gauche dans un mot.

Exemple : $(10010101)_2$; $(1000011)_2$

Selon la forme polynomiale, nous pouvons écrire : $(N)_2 = \sum_{i=0}^{n} a_i 2^i$

Où : $a_i \in \{0,1\}$

Exemple : $(1011)_2 = (1*2^3)+(0*2^2)+(1*2^1)+(1*2^0)$

4. Code octal (Base 8)

Ce système dispose de huit symboles : {**0,1,2,3,4,5,6,7**}.

De la même manière, un nombre octal s'écrit: $(N)_8 = (a_n.....a_0)_8$

Où : $a_i \in \{0,1,2,3,4,5,6,7\}$

Exemple : $(1457)_8$; $(402)_8$

Selon la forme polynomiale, nous pouvons écrire : $(N)_8 = \sum_{i=0}^{n} a_i 8^i$

5. Code Hexadécimal (Base 16)

Le code hexadécimal est le résultat d'une opération de conversion permettant de traduire un nombre quelconque, en un nombre ne comportant que les signes de **0** à **9, A, B, C, D, E** et **F.** Les chiffres **A**, **B**, **C**, **D**, **E** et **F** représentent respectivement **10**, **11**, **12**, **13**, **14** et **15**.

A chaque nombre codé en décimal correspond un seul nombre codé en hexadécimal. On donne ci-dessous des exemples :

en décimal	0	1	...	9	10	11	12	13	14	15	16
en hexadécimal	0	1	...	9	A	B	C	D	E	F	10

Un nombre hexadécimal s'écrit : $(N)_H = (a_n.....a_0)_{16}$

Où : $a_i \in \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

Exemple : $(E7D)_{16}$; $(701)_h$

Selon la forme polynomiale, nous pouvons écrire : $(N)_H = \sum_{i=0}^{n} a_i 16^i$

6. Code Tétral (Base 4)

Ce système appelé aussi base 4 comprend quatre chiffres possibles {**0, 1, 2, 3**}. Un nombre Tétral peut s'écrire sous la forme: $(N)_4 = (a_n \ldots\ldots\ldots a_0)_4$

Exemple : $(2301)_4$; $(100)_4$

Selon la forme polynomiale, nous pouvons écrire : $(N)_4 = \sum_{i=0}^{n} a_i 4^i$

IV. Conversion entre les systèmes de numération

Il s'agit de la conversion d'un nombre écrit dans une base $\mathbf{B_1}$ à son équivalent dans une autre base $\mathbf{B_2}$.

Le passage d'un code décimal vers un autre code de base $\mathbf{B_i}$ s'appelle un **codage**, ainsi l'opération inverse (passage d'un code $\mathbf{B_i}$ vers le code décimal) s'appelle un **décodage**. De plus, le passage d'un code $\mathbf{B_1}$ vers un autre code $\mathbf{B_2}$ autres que le décimal s'appelle un **transcodage**. Le schéma suivant illustre clairement ce principe de conversion entre les systèmes de numération.

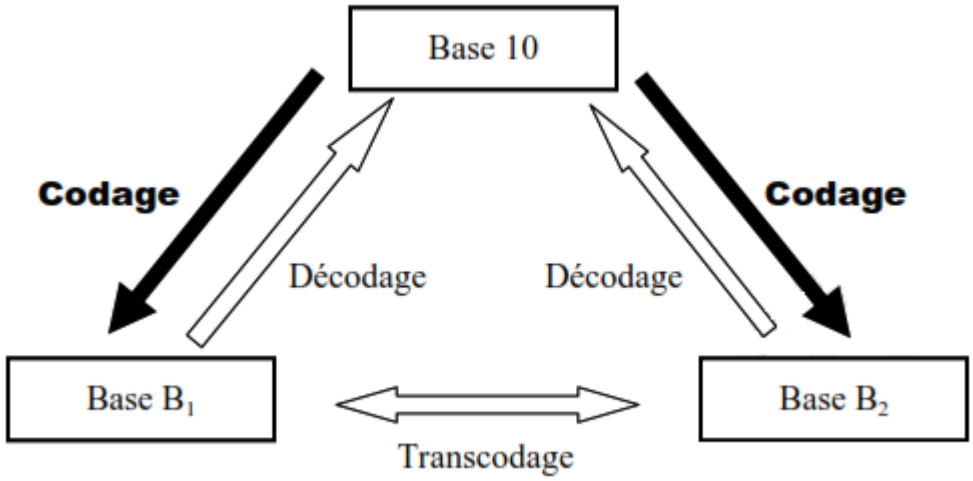


Figure 4. Codage, décodage et transcodage d'information

1. Conversion d'un nombre décimal entier en un nombre binaire (codage)

D'une façon générale, pour convertir un nombre décimal entier en un nombre de base **B** quelconque, il faut faire des divisions entières successives par la base **B** et conserver à chaque fois le reste de la division. On s'arrête lorsqu'on obtient un résultat inferieur à la base **B**. Le

nombre recherche **N** dans la base **B** s'écrit de la gauche vers la droite en commençant par le dernier résultat allant jusqu'au premier reste.

Alors, pour écrire $(88)_{10}$ en binaire, on utilise la méthode des divisions **successives par 2** jusqu'à un quotient **égal à 0**. Les restes successifs pris de bas en haut forment le nombre binaire recherché.

Exemple : Coder les nombres $(88)_{10}$ et $(45)_{10}$ en binaire.

Solution :

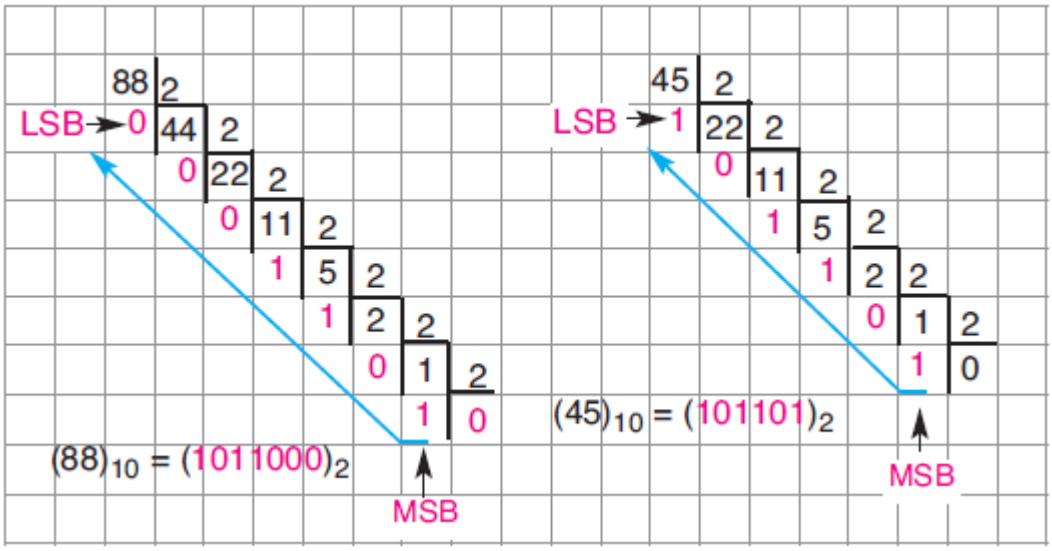


Figure 5. Exemple de codage d'information

Remarque :

- Le nombre binaire obtenu par divisions successives par **2** est appelé **nombre binaire pur ou nombre binaire naturel.**
- Ce système de numération binaire présente l'avantage d'être traité par des dispositifs numériques.

2. Conversion d'un nombre décimal à virgule en un nombre binaire

D'une façon générale, pour convertir un nombre décimal à virgule dans une base **B** quelconque, il faut :

- Convertir la partie entière en effectuant des divisions successives par **B** (comme nous l'avons vu précédemment).
- Convertir la partie fractionnaire en effectuent des multiplications successives par **B** et en conservant à chaque fois le chiffre devenant entier.

Exemple : Coder le nombre $(58{,}625)_{10}$ en base 2.

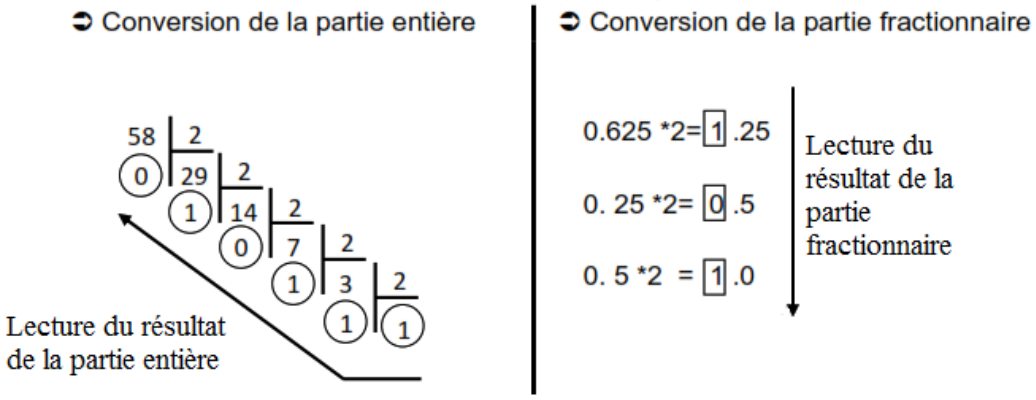


Figure 6. Codage d'un nombre décimal à virgule

Enfin, la valeur de conversion est : $(58.625)_{10}=(111010.101)_2$

Remarque :

Parfois en multipliant la partie fractionnaire par la base **B** on n'arrive pas à convertir toute la partie fractionnaire. Ceci est dû essentiellement au fait que le nombre à convertir n'a pas un équivalent exacte dans la base **B** et sa partie fractionnaire est **cyclique.**

Exemple : $(\mathbf{0.15})_{10}=(?)_2$

0.15 *2 = **0** .3

0.3 *2 = **0** .6

0.6 *2 = **1** .2

0.2 *2 = **0** .4

0.4*2 = **0** .8

0.8*2 = **1** .6

0.6 *2 = **1** .2

0.2 *2 = **0** .4

0.4*2 = **0** .8

0.8*2 = **1** .6

Alors, la valeur de conversion est : $(\mathbf{0.15})_{10} = (\mathbf{0.0010011001})_2$

On dit que le nombre $\mathbf{(0.15)_{10}}$ est cyclique dans la base 2 de période **1001.**

3. Conversion d'un nombre binaire en un nombre décimal (décodage)

L'opération de conversion d'un nombre binaire en décimal est appelée **décodage.** On exploite directement la forme polynomiale.

Exemple : Ecrire en décimal (ou décoder) les nombres $(101101)_2$ et $(11001100)_2$

Solution :

$$(101101)_2 = 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0$$
$$= 32 + 0 + 8 + 4 + 0 + 1$$
$$= (45)_{10}$$
$$(11001100)_2 = 1x2^7 + 1x2^6 + 0x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 0x2^0$$
$$= 128 + 64 + 0 + 0 + 8 + 4 + 0 + 0$$
$$= (204)_{10}$$

4. Conversion d'un nombre décimal en un nombre hexadécimal (codage)

L'opération de conversion d'un nombre décimal en hexadécimal est appelée **codage**.

Pour écrire $(88)_{10}$ en hexadécimal, on utilise la méthode des divisions successives par **16** jusqu'à un quotient égal à **0**. Les restes successifs, pris de bas en haut, forment le nombre code en Hexadécimal recherché.

Exemple : Coder les nombres $(88)_{10}$ et $(45)_{10}$ en hexadécimal.

Solution :

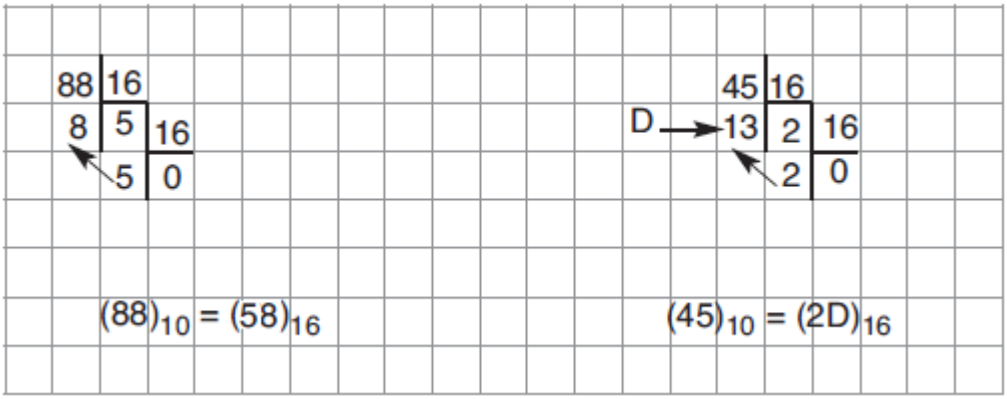


Figure 7. Codage d'un nombre décimal en hexadécimal

5. Conversion d'un nombre hexadécimal en un nombre décimal (décodage)

L'opération de conversion d'un nombre hexadécimal en décimal est appelée **décodage.** On exploite directement la forme polynomiale.

Exemple : Écrire en décimal (ou décoder) les nombres $(123)_{16}$ et $(A7E)_{16}$

Solution :

$$(123)_{16} = 1x16^2 + 2x16^1 + 3x16^0$$
$$= 256 + 32 + 3$$
$$= (291)_{10}$$

$$(A7E)_h = 10x16^2 + 7x16^1 + 14x16^0$$
$$= 2560 + 112 + 14$$
$$= (2686)_{10}$$

6. Conversion d'un nombre décimal en octal (codage)

Même principe qu'avant, sauf qu'au lieu de diviser par 2 ou par 16, on divise par **8**.

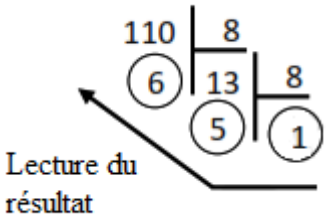


Figure 8. Codage d'un nombre décimal en Octal

Finalement, la valeur de conversion est : $(110)_{10}=(156)_8$

7. Conversion d'un nombre Octal en un nombre décimal (décodage)

On exploite directement la forme polynomiale, le décodage d'un nombre Octal est le suivant :

Exemple : Écrire en décimal (ou décoder) le nombre $(7452)_8$.

Solution :

$$(7452)_8 = 7x8^3 + 4x8^2 + 5x8^1 + 2x8^0$$
$$= 3584 + 256 + 40 + 2$$
$$= (3882)_{10}$$

8. Conversion d'un nombre décimal en Tétral (codage)

L'opération de codage ici se fait par division du nombre décimal par **4**. Par la suite un exemple de codage du nombre $(105)_{10}$ en Tétral.

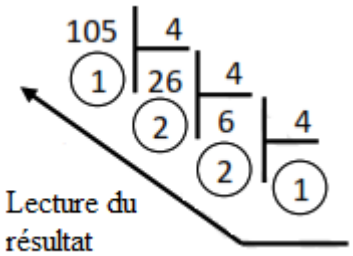


Figure 9. Codage d'un nombre décimal en Tétral

Le résultat de codage est : $(105)_{10}=(1221)_4$

9. Conversion d'un nombre Tétral en un nombre décimal (décodage)

On exploite aussi la forme polynomiale, le décodage d'un nombre Tétral est le suivant :

Exemple : Écrire en décimal (ou décoder) le nombre $(231102)_4$.

Solution :

$$(231102)_4 = 2x4^5 + 3x4^4 + 1x4^3 + 1x4^2 + 0x4^1 + 2x4^0$$

$$= 2048 + 768 + 64 + 16 + 2$$

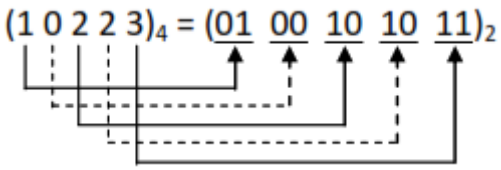
$$= (2898)_{10}$$

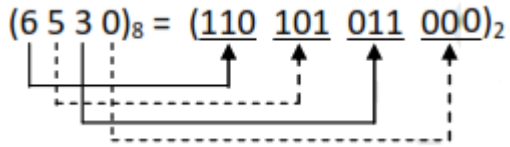
10. Autres conversions (transcodage)

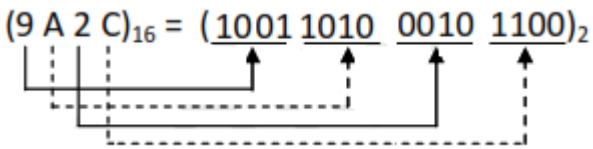
Pour faire La conversion d'un nombre d'une base quelconque vers une autre base $\mathbf{B_2}$ il faut passer par la base **10**. Mais si la base $\mathbf{B_1}$ et $\mathbf{B_2}$ s'écrivent respectivement sous la forme d'une puissance de **2** on peut passer par la base **2** (binaire) :

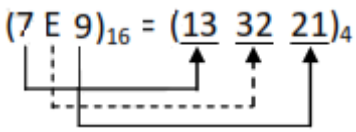
- Base Tétrale (base 4) : $4=2^2$ chaque chiffre Tétral se convertit tout seul sur **2 bits** en binaire.
- Base octale (base 8) : $8=2^3$ chaque chiffre octal se convertit tout seul sur **3 bits** en binaire.
- Base hexadécimale (base 16) : $16=2^4$ chaque chiffre hexadécimal se convertit tout seul sur **4 bits** en binaire.

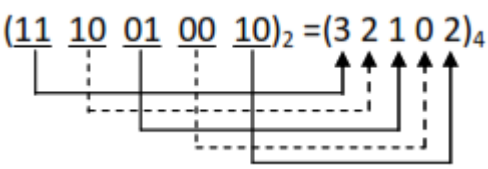
Exemple :

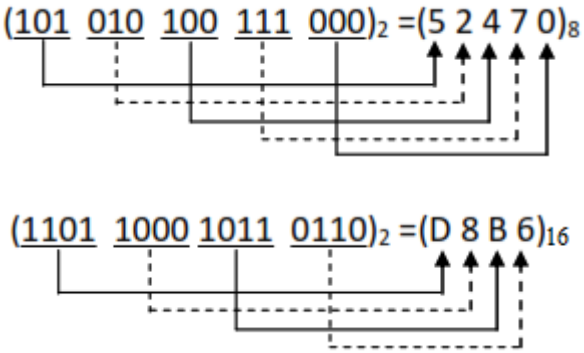












Le tableau ci-dessous représente un récapitulatif sur les systèmes de numération les plus utilisés.

Décimal	Binaire	Tétral	Octal	Hexadécimal
0	0000	0	0	0
1	0001	1	1	1
2	0010	2	2	2
3	0011	3	3	3
4	0100	10	4	4
5	0101	11	5	5
6	0110	12	6	6
7	0111	13	7	7
8	1000	20	10	8
9	1001	21	11	9
10	1010	22	12	A
11	1011	23	13	B
12	1100	30	14	C
13	1101	31	15	D
14	1110	32	16	E
15	1111	33	17	F

V. Notions d'arithmétique binaire

On procède de la même façon que celle utilisée dans la base décimale, on peut faire des calculs arithmétiques pour les nombres binaires.

1. L'opération d'addition

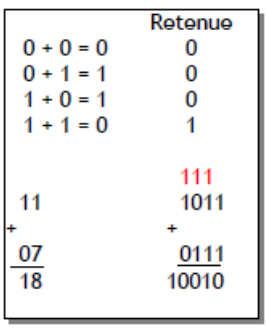


Figure 10. Règles de l'opération d'addition

Exemple :

$$\begin{array}{rr} & 1101110 \\ + & 100010 \\ \hline = & (10010000)_2 \end{array}$$

2. L'opération de soustraction

L'opération de la soustraction suit les règles mentionnées ci-dessous :

Retenue/emprunt	
0 - 0 = 0	0
0 - 1 = 1	1
1 - 0 = 1	0
1 - 1 = 0	0
27	11011
-	-
7	00111
	1
20	10100

Figure 11. Règles de l'opération de soustraction

Exemples :

$$\begin{array}{rr} & 1110110 \\ - & 110101 \\ \hline = & (1000001)_2 \end{array} \qquad \begin{array}{rr} & 1000001001 \\ - & 11110011 \\ \hline = & (100010110)_2 \end{array}$$

3. L'opération de multiplication

L'opération de multiplication se fait de la même manière avec les nombres décimaux.

$$\begin{array}{rr} & 1110110 \\ * & 11011 \\ \hline & 1110110 \\ & 1110110 \\ & 1110110 \\ & 1110110 \\ \hline = & (110001110010)_2 \end{array}$$

4. L'opération de division

La division des nombres binaires respecte la même philosophie d'une division décimale. Par la suite un exemple qui explique l'opération de la division en binaire.

$$\begin{array}{r|l} 101100 & 100 \\ \underline{-100} & \underline{} \\ 00110 & 1011 \\ \underline{-\ 100} & \\ 0100 & \\ \underline{-\ 100} & \\ 000 & \end{array}$$

Alors, la division des nombres (44/4=11).

VI. Codage de l'information binaire

Un système électronique traite les informations en binaire. Or, ces informations à traiter sont de différentes natures. Par exemple, en traitement de texte, on manipule des caractères ; pour qu'un ordinateur traite ces caractères, il faut associer alors à chaque caractère un nombre binaire. On étudie en particulier : Le code binaire pur, le code GRAY, le code BCD et le code ASCII.

1. Le code binaire pur

C'est une représentation numérique des nombres dans la base 2. Il est aussi appelé **code binaire naturel**. C'est le code binaire sans aucune codification, c'est à dire qui découle directement du principe général de la numération. C'est le code naturel utilisé dans les systèmes numériques (ordinateur, etc.).

Le tableau suivant donne le code binaire pur pour un exemple d'un mot de **4 bits** ($A_3A_2A_1A_0$).

Valeur décimale	Code binaire			
	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

Figure 12. Code binaire naturel

Remarque : Ce code présente l'inconvénient de changer plus qu'un seul bit quand on passe d'un nombre à un autre immédiatement supérieur.

2. Le code binaire réfléchi (code GRAY)

Dans les systèmes industriels où on a besoin de mesurer un déplacement linéaire ou angulaire, on utilise le **"code GRAY".** La raison de ce choix est que si le système qui mesure le déplacement (capteur) utilise le code binaire pur, le déplacement d'une position à une autre voisine génère des combinaisons intermédiaires fausses, car plusieurs bits varient en même temps.

Pour remédier à ce problème, il suffit de coder chaque position de façon que les valeurs de positions successives ne différent que d'un seul bit. C'est pour cela qu'on l'appelle **"code à distance unité".** On l'appelle aussi "**code binaire réfléchi**" parce que pour le construire, on procède par réflexion comme l'indique le tableau suivant avec 4 bits.

Valeur décimale	Code binaire				Code GRAY			
	A_3	A_2	A_1	A_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

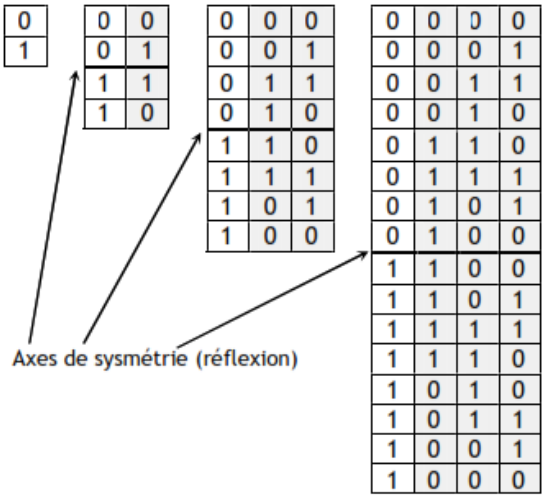


Figure 13. Codes binaire naturel & code GRAY

Remarque 1:

- On a 16 combinaisons différentes.
- Dans le passage d'une combinaison à une autre, il n'y a qu'un bit qui change.

Remarque 2:

Conversion du Binaire Naturel vers le Binaire Réfléchi :

Il s'agit de comparer les bits b_{n+1} et le bit b_n du binaire naturel, le résultat est b_r du binaire réfléchi qui vaut **0** si $b_{n+1}=b_n$ ou **1** sinon. Le premier bit à gauche reste inchangé.

$(6)_{10}=(?)_{BR}$	$(10)_{10}=(?)_{BR}$
$(6)_{BN} = 1 \leftrightarrow 1 \leftrightarrow 0$ $\downarrow \quad \downarrow \quad \downarrow$ $(6)_{BR} = 1 \quad 0 \quad 1$ $(6)_{10}=(110)_{BN}=(101)_{BR}$	$(10)_{BN} = 1 \leftrightarrow 0 \leftrightarrow 1 \leftrightarrow 0$ $\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$ $(10)_{BR} = 1 \quad 1 \quad 1 \quad 1$ $(10)_{10}=(1010)_{BN}=(1111)_{BR}$

Conversion du Binaire Réfléchi vers le Binaire Naturel:

Il s'agit de comparer le bit b_{n+1} du binaire naturel et le bit b_n du binaire réfléchi, le résultat est b_n du binaire naturel qui vaut **0** si $b_{n+1}=b_n$ ou **1** sinon. Le premier bit à gauche reste inchangé.

$(10)_{10}=(?)_{BN}$	$(13)_{10}=(?)_{BN}$
$(10)_{BR}=1\ 1\ 1\ 1$ $(10)_{BN}=1\ 0\ 1\ 0$ $(10)_{10}=(1111)_{BR}=(1010)_{BN}$	$(13)_{BR}=1\ 0\ 1\ 1$ $(13)_{BN}=1\ 1\ 0\ 1$ $(13)_{10}=(1011)_{BR}=(1101)_{BN}$

3. Le code décimal codé binaire (code DCB)

Le code BCD (Binary Coded Decimal) qui veut dire **Binaire Codé en Décimal** est la traduction en binaire des **9** premiers chiffres du système décimal.

Sa propriété est d'associer **4 bits** représentent chaque chiffre en binaire naturel. L'application la plus courante est celle de l'affichage numérique ou chaque chiffre est associé à un groupe de **4 bits** portant le code **BCD**.

Valeur décimale	Code BCD			
	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

Figure 14. Code BCD

Exemple :

$(571)_2$ = 1000111011 en binaire pur
= 0101 0111 0001 en BCD
5 7 1

$(9\ 4\ 2\ 7)_{10} = (1001\ 0100\ 0010\ 0111)_{BCD}$

4. Le code ASCII

Le code ASCII (American Standard Code for Information Interchange) est utilisé en informatique pour communiquer entre le clavier du micro-ordinateur et l'unité centrale. On distingue deux codes ASCII : le code **ASCII standard** et **le code ASCII étendu**.

Le clavier est équipé d'un circuit spécial qui contrôle ses circuits en permanence. A chaque touche correspond un mot binaire. Sous sa forme standard, il utilise **7 bits**. Ce qui permet de générer 2^7**=128** caractères. Ce code représente les lettres alphanumériques majuscules et minuscules, les chiffres décimaux, des signes de ponctuation et des caractères de commande. Le code ASCII entendu en possède **256** caractères.

Chaque code est défini par **3 bits** d'ordre supérieur $b_6b_5b_4$ et **4 bits** d'ordre inférieur $b_3b_2b_1b_0$. Ainsi le caractère "**A**" a pour code hexadécimal 41_H.

Exemple :

- **A** :**(65)** $_{ASCII}$ = **(01000001)**$_2$ = **(41)** $_H$
- **B** : **(66)** $_{ASCII}$ = **(01000010)**$_2$ = **(42)** $_H$
- **z** : **(122)** $_{ASCII}$ = **(01111010)**$_2$ = **(7A)** $_H$
- **[** : **(91)** $_{ASCII}$ = **(01011011)**$_2$ = **(5B)** $_H$

Binaire				b6	0	0	0	0	1	1	1	1
				b5	0	0	1	1	0	0	1	1
				b4	0	1	0	1	0	1	0	1
				Hexadécimal	0	1	2	3	4	5	6	7
b3	b2	b1	b0	Décimal	0	16	32	48	64	80	96	112
0	0	0	0	0	+0	NUL	TC7 (DEL)	SP	0	@	P	`	p
0	0	0	1	1	+1	TC1 (SOH)	DC1	!	1	A	Q	a	q
0	0	1	0	2	+2	TC2 (STX)	DC2	"	2	B	R	b	r
0	0	1	1	3	+3	TC3 (ETX)	DC3	#	3	C	S	c	s
0	1	0	0	4	+4	TC4 (EOT)	DC4	$	4	D	T	d	t
0	1	0	1	5	+5	TC5 (ENQ)	TC8 (NAK)	%	5	E	U	e	u
0	1	1	0	6	+6	TC6 (ACK)	TC9 (SYN)	&	6	F	V	f	v
0	1	1	1	7	+7	BEL	TC 10 (ETB)	'	7	G	W	g	w
1	0	0	0	8	+8	FE0 (BS)	CAN	(	8	H	X	h	x
1	0	0	1	9	+9	FE1 (HT)	EM	)	9	I	Y	i	y
1	0	1	0	A	+10	FE2 (LF)	SUB	*	:	J	Z	j	z
1	0	1	1	B	+11	FE3 (VT)	ESC	+	;	K	[	k	é
1	1	0	0	C	+12	FE4 (FF)	IS4 (FS)	,	<	L	\	l	ù
1	1	0	1	D	+13	FE5 (CR)	IS3 (GS)	-	=	M	]	m	è
1	1	1	0	E	+14	SO	IS2 (RS)	.	>	N	^	n	¯
1	1	1	1	F	+15	SI	IS1 (US)	/	?	O	_	o	DEL

Figure 15. Code ASCII standard

VII. Écriture des nombres signés

Les circuits numériques n'assimilent pas les signes **'+'** et **'-'** tels quels. D'où la nécessité d'adopter une certaine convention pour représenter les nombres binaires signés. Pour cela, on ajoute un bit supplémentaire appelé bit de signe et on attribue au nombre positif le bit de signe **'0'** et au nombre négatif le bit de signe **'1'**.

Les nombres signés sont représentés selon trois notations :

- Notation module plus signe.
- Notation complément à 1.
- Notation complément à 2.

Remarque : Les nombres positifs ont la même représentation dans les trois notations.

1. Notation module plus signe

Un nombre binaire signé comprend un bit de signe (**0** pour les nombres positifs et **1** pour les nombres négatifs) et **n** bits indiquant le module. Ainsi, si un nombre **N** est codé sur **n+1** bits, alors ce nombre est compris entre **$-(2^n-1)$** et **$+(2^n-1)$**.

Exemple : Avec **5** bits, **N** est compris entre **-15** et **15**.

	Signe	Module
+12	0	1100
-12	1	1100

2. Notation complément à 1

Pour représenter un nombre **N négatif** en notation complément à **1**, il suffit d'attribuer au bit de signe la valeur de **1** et transformer le module en son **complément à 1**.

Exemple :

+12	0	1100	
-12	1	0011	notation en complément à 1

3. Notation complément à 2

Pour représenter un nombre négatif en notation **complément à 2**, il suffit d'attribuer au **bit de signe** la valeur de **1** et de transformer le module en son **complément à 2**.

Exemple :

+12	0	1100	
-12	1	0100	notation en complément à 2

VIII. Bibliographie/Webographie

- « Système logique : logique combinatoire » , Institut Supérieur des Etudes Technologiques de Nabeul, Ben Amara Mahmoud & Gâaloul Kamel 2015/2016.
- Circuits Numériques Théorie et Applications, Ronald J.Tocci. Reynald Goulet inc. 1996.
- Manuel Sciences de l'ingénieur, EL MIMOUNI EL HASSAN et al. 2006.
- http://sebastien.bernard.free.fr/cours-tp-td-exo/TD-E-Logique-sequentielle-Fonction-Comptage.pdf

LISTE DES FIGURES