

Chapitre II : LOGIQUE BINAIRE

II.1. NIVEAUX D'ABSTRACTION DES DONNEES

Pour qu'une information numérique soit traitée par un circuit, elle doit être mise sous forme adaptée à celui-ci. Pour cela Il faut choisir un système de numération de base.

La matière première sur laquelle travaille un système numérique est la donnée (bits). Pour représenter les données dans la machine, il y a 3 niveaux d'abstraction :

- 1^{er} niveau d'abstraction → Lettres et chiffres
 - **Un Chiffre** : Symbole représentant une valeur numérique. Les chiffres sont des éléments singuliers dont leur combinaison peut représenter des nombres.
 - **Une lettre** : Caractère abstrait auquel on a donné une signification spéciale. Les lettres sont des éléments singuliers dont leur combinaison peut représenter des mots.

- 2^{ème} niveau d'abstraction → Mots et Nombres
 - **Les nombres** : sont des suites des chiffres auxquelles on a donné une signification particulière.
 - **Les mots** : sont des suites des lettres auxquelles on a donné une signification particulière.

- 3^{ème} niveau d'abstraction → Variables, Fonctions, Domaines.
 - **Variable** : grandeur pouvant prendre des états associés à un ou plusieurs caractères d'un événement tel que : « vrai » ou « faux »; fermé et ouvert; allumé et atteint.
 - **Fonction** : Ensemble des variables pouvant prendre n valeurs données reliées par des opérateurs logiques.

II.2. RAPPELS SUR LES SYSTEMES DE NUMEROTATION

2.2.1. Définition et représentation des nombres

La **numération** désigne les techniques de représentation des nombres. Aussi, elle concerne les mots, les signes, les caractères ou les symboles qui ont permis aux différents peuples d'énoncer, de mimer ou d'écrire et de compter.

Msc. Ir. KAPULUZA MUMBBA Dubois	Circuits Logiques Numériques Notes de Cours	VPL_2023-2024
---------------------------------	--	---------------

Les représentations écrites au moyen de signes constituent des systèmes de numération. Ces derniers sont nés, en même temps que l'écriture, de la nécessité d'organiser les récoltes, le commerce et la datation.

Un système de numération est constitué par un ensemble de :

- Caractères ou symboles appelés « chiffres » ;
- Règles concernant les nombres formés à l'aide de ces chiffres ;
- Opérations à effectuer sur ces nombres.

Un système de numération de base quelconque « b » est un système numérique utilisant « b » caractères (chiffres) allant de 0 à b-1.

Tout nombre N peut se décomposer en fonction des puissances entières de la base de son système de numération. Cette décomposition s'appelle la forme polynomiale du nombre N et qui est donnée par :

$$N = a_n B^n + a_{n-1} B^{n-1} + a_{n-2} B^{n-2} + \dots + a_2 B^2 + a_1 B^1 + a_0 B^0$$

- B : Base du système de numération, elle représente le nombre des différents chiffres qu'utilise ce système de numération.
- a_i : un chiffre (ou digit) parmi les chiffres de la base du système de numération.
- i : rang du chiffre a_i .

Exemple : $2083,45_{(10)} = 2 \times 10^3 + 0 \times 10^2 + 8 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$

De nombreux systèmes de numération sont utilisés en technologie numérique. Les plus utilisés sont les systèmes : Décimal (base 10), Binaire (base 2), Tétral (base 4), Octal (base 8) et Hexadécimal (base 16).

Décimal	Binaire	Tétral	Octal	Hexadécimal
0	0	0	0	0
1	1	1	1	1
2	10	2	2	2
3	11	3	3	3
4	100	10	4	4
5	101	11	5	5
6	110	12	6	6
7	111	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

2.2.2. Le système unaire

Le système unaire (système monadique), est le système de numération le plus élémentaire pour représenter les entiers naturels. Il est représenté par un **seul symbole** dont les différentes combinaisons donnent le quantitatif numérique. Ainsi pour représenter un nombre N, il suffit de répéter N fois un symbole pré-choisi.*

Exemple : Convertir le nombre « 17 » en système unaire ?

N.B. Le système des chiffres romains utilise une sous base quinaire, (**V, L, D**) superposée sur une base décimale.

SYS. DECIMAL	1	2	3	4	5	9	10	28	44	50	100	500	1000
SYS. ROMAIN													

2.2.3. Les systèmes évolués

De nombreuses bases de numération ont été employées par des peuples et à des époques variées :

a) Système binaire (B=2)

C'est un système de numération utilisant la base 2 et représente un ensemble de 2 valeurs antinomiques, **S₂=(0,1)** comme :

- vrai/faux,
- blanc/noir,
- ouvert/ fermé,
- marche/arrêt (*on* et *off*).

Ce système convient notamment pour représenter le fonctionnement de l'électronique numérique utilisée dans les ordinateurs, d'où son usage en informatique.

On peut également représenter un signal binaire en plusieurs codes sources :

- Signal binaire simple courant NRZ (= Signal binaire pur ou naturel) ;
- Signal binaire simple courant RZ
- Signal binaire double courant NRZ
- Signal binaire bipolaire RZ, etc.

b) Système trinaire (base 3)

Le **système trinaire** ou **ternaire** est le système de numération de la base 3. **TRITS** (Trinary digIT) = chiffres ternaires.

- Système trinaire ordinaire : 3 symboles → **S₃=(0, 1, 2)**.
- Système trinaire balancé : 3 symboles → **S₃=(-1, 0, 1)**.

Il est adapté pour représenter les variables booléennes dont les valeurs sont à 3 états logiques : « **vrai** », « **faux** » et « **indéterminé** ».

Dans un tel système, les nombres positifs et négatifs bénéficient de la même représentation.

c) Système quinaire (B=5)

Le système **quinaire** est le système de numération de base cinq. $S_5 = (0, 1, 2, 3, 4)$ pour représenter n'importe quel nombre entier.

Il fut une base très commode parmi les premières cultures, car les humains possèdent cinq doigts sur chaque main.

$$S_5 = (0, 1, 2, 3, 4).$$

d) Système octal (B=8)

Un système octal est un système de numération de base 8, et utilise les chiffres de 0 à 7. Il fut inventé par le roi Charles XII de Suède.

$$S_8 = (0, 1, 2, 3, 4, 5, 6, 7).$$

e) Système décimal (B=10)

Le système décimal a été utilisé par de nombreuses civilisations, comme les Chinois dès les premiers temps, et, probablement, les Proto-indo-européens. Aujourd'hui, il est de loin le plus répandu. Ce système utilise les caractères décimaux suivants :

$$S_{10} = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).$$

f) Système hexadécimal (B=16)

C'est un système de numération utilisant la base 16. Il utilise les 10 premiers chiffres arabes puis les 6 premières lettres de l'alphabet latin :

$$S_{16} = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).$$

L'usage de ces chiffres fut imposé mondialement par l'entreprise américain IBM qui commença à l'utiliser depuis 1963. Il est actuellement le standard reconnu.

Le système hexadécimal utilise jusqu'à quatre fois moins de chiffres que le système binaire pour représenter le même nombre.

g) Système sexagésimal (B=60)

Était utilisé pour la numération babylonienne, ainsi que par les Indiens et les Arabes en trigonométrie. Il sert actuellement dans la mesure du temps et des angles et pour préciser des coordonnées géographiques.

L'unité standard du sexagésimal est le degré (360 degrés), puis la minute (60 minutes = 1 degré) puis la seconde (60 secondes = 1 minute). L'usage moderne du sexagésimal est assez proche de celui de la mesure du temps, dans lequel il y a 24 heures dans une journée, 60 minutes dans une heure et 60 secondes dans une minute. La mesure moderne du temps correspond de façon arrondie à la durée de la rotation de la terre (jours) et de sa révolution (année).

Msc. Ir. KAPULULA MUMBA Dubois	Circuits Logiques Numériques Notes de Cours	UPL_2023-2024
--------------------------------	--	---------------

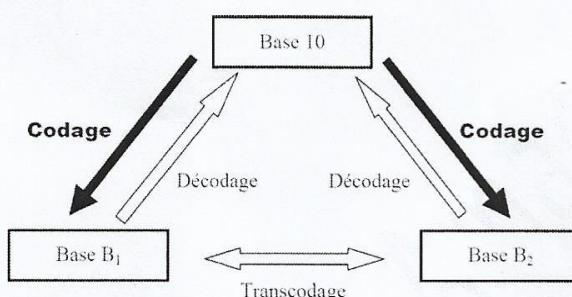
II.3. CONVERSION OU CHANGEMENT DES BASES DE NUMÉRATION

2.3.1. Codage des nombres

a) Définition

Dans les systèmes de numération, on appelle codage, le passage de la représentation d'un nombre de la base 10 (système décimal) vers une autre base (à l'occurrence la base 2).

En informatique, les nombres décimaux sont introduits dans les machines informatiques à l'aide du clavier, mais leur traitement se fait par le processeur (calculateur numérique) après leur conversion en système binaire et puis stockées dans les mémoires (bascules, registres, mémoires).



b) Procédé de conversion d'un entier positif décimal

« Si N est un entier positif dans la base 10, pour avoir sa représentation en base b, il suffit de diviser N par b, puis le quotient par b ainsi de suite jusqu'à ce que le quotient devienne inférieur à la base. Le résultat est le rangement de tous les restes notés en sens inverse (du dernier au premier reste) ».

Nombre	Base	Reste	Sens
202	2	0	
101	2	1	
50	2	0	
25	2	1	
12	2	0	
6	2	0	
3	2	1	
1	2	1	
0			↑

Exemple :

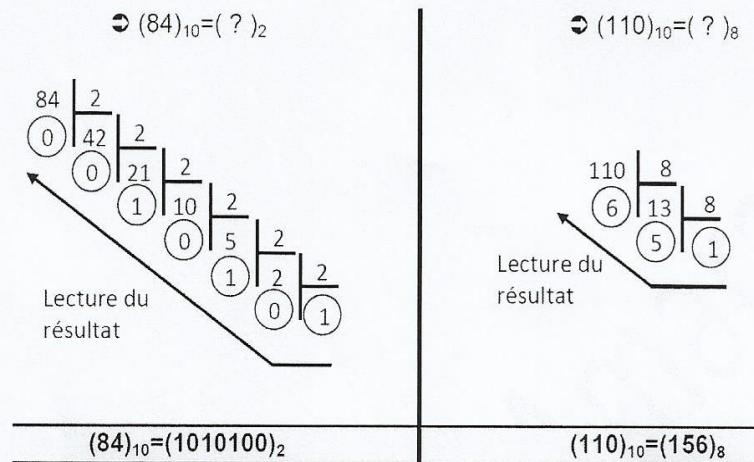
Soit à convertir :

- a) $202_{(10)}$ en base 2 = ?
- b) $54_{(10)}$ en base 8 = ?

$$\rightarrow 202_{(10)} = 11001010_{(2)}$$

$$\text{Convertir : } 84_{(10)} = ? \quad (2)$$

$$110_{(10)} = ? \quad (8)$$



- Soit à convertir : $54_{(10)} = ??? \quad (8)$

Nombre	Base	Reste	Sens
54	8	6	
6	8	6	↑

$$\rightarrow 54_{(10)} = \mathbf{66}_{(8)}$$

c) Conversion d'un nombre fractionnaire décimal

Pour effectuer le codage des nombres fractionnaires, on peut utiliser la méthode suivante : Conversion d'un nombre fractionnaire

- Méthode 1 : on retranche successivement du nombre les plus hautes puissances négatives de 2, jusqu'à l'épuisement du reste ou de la précision désirée.

Exemple : $0,875_{(10)} = ?_{(2)}$.

$$\begin{aligned} 0,875 - 1 \times 2^{-1} &= 0,875 - 0,500 = 0,375 \\ 0,375 - 1 \times 2^{-2} &= 0,375 - 0,250 = 0,125 \\ 0,125 - 1 \times 2^{-3} &= 0,125 - 0,125 = 0 \end{aligned}$$

$$\rightarrow 0,875_{(10)} = 0,111_{(2)}.$$

- Méthode 2 : La partie fractionnaire sera multipliée en base 10 par la base b, on obtiendra l'expression N1. On multipliera la partie fractionnaire de N1 par b et on obtiendra l'expression N2 et ainsi de suite. On tire la partie fractionnaire en base b en prenant de haut en bas les entiers de $(N_i)_{10}$ pour la représenter de gauche vers la droite.

Exemple 1 : 54 , 24 ₍₁₀₎ = ? ₍₂₎.

Partie entière partie fractionnaire

D'abord, on convertie la partie entière en suivra la procédure habituelle :

$$54_{(10)} = 110110_{(2)}$$

Enfin, on convertie la partie fractionnaire, suivant :

$$0,24_{(10)} = ?_{(2)}$$

$$\begin{aligned} 0,24 \times 2 &= 0,48 \\ 0,48 \times 2 &= 0,96 \\ 0,96 \times 2 &= 1,92 \\ 0,92 \times 2 &= 1,84 \\ 0,84 \times 2 &= 1,68 \\ 0,68 \times 2 &= 1,36 \\ 0,36 \times 2 &= 0,72 \\ 0,72 \times 2 &= 1,44 \dots \end{aligned} \quad \left. \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array} \right\} \quad 0,24_{(10)} = 00111101_{(2)}$$

$$\Rightarrow 54,24_{(10)} = 110110,110110_{(2)}$$

Exemple 2 : 38,5625 ₍₁₀₎ = **100110,1001** ₍₂₎

N.B.

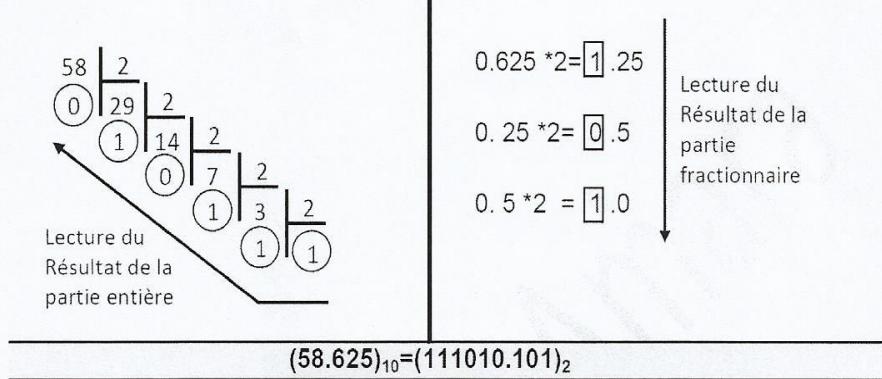
Règle pour arrondissement

Ri	ri+1	Ri
0	0	0
0	1	1
1	0	1
1	1	0

Conversion du nombre (58,625) en base 2

⇒ Conversion de la partie entière

⇒ Conversion de la partie fractionnaire



2.3.2. Décodage des nombres

Dans le système de numération, on appelle décodage, le passage de la représentation d'un nombre de la base quelconque « b » à la représentation de ce même nombre en base 10 (système décimal).

D'une façon générale, un nombre « N » donné dans une base quelconque « b » peut être représenté en base décimale comme suit :

$$N_{(b)} = a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + a_{n-2} \cdot b^{n-2} + \dots + a_2 \cdot b^2 + a_1 \cdot b + a_0 + a_{-1} \cdot b^{-1} + a_{-2} \cdot b^{-2} + \dots + a_{-n} \cdot b^{-n}$$

Avec :
 a : digit
 b : base
 n : position ou rang du digit
 b^n : poids de la colonne

Exemple 1 : $11001010_{(2)} = ?_{(10)}$.

Nombre binaire	1	1	0	0	1	0	1	0
Poids de la colonne	$1 \cdot 2^7$	$1 \cdot 2^6$	$0 \cdot 2^5$	$0 \cdot 2^4$	$1 \cdot 2^3$	$0 \cdot 2^2$	$1 \cdot 2^1$	$0 \cdot 2^0$
Valeur décimale	128 +	64 +	0 +	0 +	8 +	0 +	2 +	0

$$\rightarrow \mathbf{11001010_{(2)}} = \mathbf{202_{(10)}}.$$

Exemple 2 : $603,41_{(8)} = ?_{(10)}$.

Nombre binaire	6	0	3	4	1
Poids de la colonne	$6 \cdot 8^2$	$0 \cdot 8^1$	$3 \cdot 8^0$	$4 \cdot 8^{-1}$	$1 \cdot 8^{-2}$
Valeur décimale	384 +	0 +	3	0,5	0,015625

$$\rightarrow \mathbf{603,41_{(8)}} = \mathbf{387,515625_{(10)}}.$$

Exemple 3 : $110111,1011_{(2)} = 55,6875_{(10)}$

2.3.3. Transcodage

On appelle transcodage, le passage d'une base quelconque b_1 à une autre b_2 (avec b_1 et b_2 différents de la base 10).

Deux solutions sont possibles :

- Décodage suivi de codage ;
- Transcodage direct.

❖ Décodage suivi codage

Dans un premier temps, la valeur en b_1 est décodée. Dans un second temps, la valeur décimale précédemment obtenue est codée en base b_2 .

Exemple : convertir $23_{(8)} = ?_{(2)}$.

- On procède par le décodage $23_{(8)} = ?_{(10)}$
 $2 \cdot 8^{-1} + 3 \cdot 8^0 = 16 + 3 = 19_{(10)} \rightarrow 23_{(8)} = 19_{(10)}$
- Puis par le codage de $19_{(10)} = ?_{(2)}$

<u>NOMBR</u>	<u>BASE</u>	<u>RESTE</u>	
E			
19	2	1	
9	2	1	
4	2	0	
2	2	0	
1	2	1	
0			

$$19_{10} = 10011_{(2)}$$

❖ Transcodage direct

- Pour faire La conversion d'un nombre d'une base quelconque B1 vers une autre base B2, il faut passer par la base 10 (c'est le décodage suivi de codage). Mais si la base B1 et B2 s'écrivent respectivement sous la forme d'une puissance de 2 on peut passer par la base 2 (binaire).
- Pour convertir une valeur binaire en tétral, octal ou hexadécimal il suffit de regrouper les chiffres binaires par tranche de 2 ($4=2^3$), 3 bits ($8 = 2^3$) ou 4 bits ($16 = 2^4$) de droite à gauche (à partir du BLMS pour la partie entière et à partir du BLPS pour la partie fractionnaire) en complétant éventuellement par des zéros non significatifs à gauche pour avoir deux, trois ou quatre bits. Remplacer chaque groupe de 2, 3 ou 4 bits par son équivalent en tétral, en octal ou hexadécimal.

Binaire	11	10	01	00
Tétral	3	2	1	0

Binaire	111	110	101	100	011	010	001	000
Octal	7	6	5	4	3	2	1	0

BINAIRE	1111	1110	1101	1100	1011	1010	1001	1000
HEXA	F	E	D	C	B	A	9	8
BINAIRE	0111	0110	0101	0100	0011	0010	0001	0000
HEXA	7	6	5	4	3	2	1	0

Exemples :

a) $1110100111_{(2)} = ?_{(8)}$
 $001 -110 -100 -111_{(2)} = 1647_{(8)}$

b) $0,1011011_{(2)} = ?_{(8)}$
 $0,101 - 101 - 100_{(2)} = 0,554_{(8)}$

Exemples :

a) $1110100111_{(2)} = ?_{(16)}$
 $0011 - 1010 - 0111_{(2)} = 3A7_{(16)}$

b) $11011,01101_{(2)} = ?_{(16)}$
 $0001 - 1011, 0110 - 1000_{(2)} = 1B, 68_{(16)}$

- Pour convertir une valeur hexadécimale ou une valeur octale en valeur binaire, il suffit de regrouper les bits en tranche de 4 bits ou de 3 bits (à partir du BLMS pour la partie entière et à partir du BLPS pour la partie fractionnaire) et de coder chaque tranche en base 16 ou en base 8 b suivant les tableau de correspondance ci haut.

Exemples :

a) $27,612_{(8)} = ?_{(2)}$
 $27,612_{(8)} = 010\ 111, 110\ 001\ 010_{(2)} = 10111, 110001010_{(2)}$

b) $7A,54_{(16)} = ?_{(2)}$
 $7A,54_{(16)} = 0111\ 1010, 0101\ 0100_{(2)} = 1111010, 010101_{(2)}$

⊕ $(1\ 0\ 2\ 2\ 3)_4 = (01\ 00\ 10\ 10\ 11)_2$

⊕ $(6\ 5\ 3\ 0)_8 = (110\ 101\ 011\ 000)_2$

⊕ $(9\ A\ 2\ C)_{16} = (1001\ 1010\ 0010\ 1100)_2$

⊕ $(7\ E\ 9)_{16} = (13\ 32\ 21)_4$

II.4. REPRESENTATION DES INFORMATIONS EN MACHINE

On utilise plusieurs codes pour représenter les caractères en binaire :

2.4.1. Représentation des entiers naturels

❖ Code Naturel et Code réfléchi

Un code dans lequel deux combinaisons (mots binaires) consécutifs sont adjacentes est dit continu. Si de plus la dernière combinaison binaire est adjacente à la première, il est dit continu cyclique.

Les codes continus les plus répandus, sont les codes réfléchis. Dans un code naturel ou pur de base b , les chiffres d'un même rang de la suite des entiers se suivent indéfiniment dans l'ordre naturel, chacun d'eux étant répété un nombre de fois égal au poids du rang considéré.

Dans un code réfléchi, les suites se présentent alternativement dans l'ordre naturel et dans l'autre inverse. De manière générale seuls les codes réfléchis de base paire sont cycliques.

Pour un codage à longueur fini « n » avec « b » symboles, on aura « b^n » mots différents.

Exemple : code binaire où $b=2$ (base) ; avec $n=4$ (taille du mot) ; on aura : 16 mots binaires (16 combinaisons binaires possibles)

CODE DECIMAL NATUREL	CODE BINAIRE NATUREL BINAIRE PUR	CODE BINAIRE REFLECHI CODE GRAY
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

Le code binaire de GRAY est caractérisé par les propriétés ci-après :

- lorsque l'on passe d'un nombre au suivant un seul élément binaire change d'état ;
- Ce code a la même densité que le code binaire naturel, même élément binaire.
- Ce code est cyclique, il se ferme sur lui-même.

Il est fréquemment utilisé dans les organes de télémesure.

❖ Codes Progressifs

Les codes progressifs sont des codes continus cycliques dans lesquels toutes les combinaisons possibles ne sont pas utilisées ; deux mots représentants deux combinaisons consécutives ne diffèrent que par l'état d'un élément. Ils sont très utilisés en circuits séquentiels.

NOMBRE DECIMAL	CODE 1	CODE2
0	000 000	000 000
1	000 001	000 001
2	000 011	000 011
3	000 111	000 010
4	001 111	000 110
5	011 111	000 100
6	111 111	001 100
7	111 110	001 000
8	111 100	011 000
9	111 000	010 000
10	110 000	110 000
11	100 000	100 000

❖ Codes BCD (Binary Coded Decimal)

Ce sont des codes utilisés pour la représentation binaire des chiffres décimaux. Le principe de BCD (ou DCB = Décimal Codé Binaire) consiste simplement à représenter séparément chaque chiffre décimal d'un nombre par un entier sur quatre bits en binaire pur. Il faut donc, 10 combinaisons binaires différentes devant comporter au moins 4 chiffres binaires (Bits).

Certains systèmes utilisent des entrées-sorties sous ce format. C'est le cas des fours à micro-ondes, des jeux électroniques, des fréquencemètres à microprocesseurs.

→ BCD pondérés

En affectant à chacune des positions binaires un certain poids, on obtient un code pondéré. Si l'on ne tient compte que des poids positifs, on peut avoir 17 codes pondérés différents.

4311	5221	5321	4421
5211	6221	4221	5421
5311	3321	6321	6421
6311	4321	7321	7421
			8421

Exemple : Code pondéré 5211

Décimal	5	2	1	1	OBSERVATION
0	0	0	0	0	
1	0	0	0	1	0 0 1 0
2	0	0	1	1	0 0 1 1
3					0 1 1 0

4	0	1	0	1	
5	0	1	1	1	
6	1	0	0	1	1 0 1 0
7	1	1	0	0	1 0 1 1
8	1	1	0	0	1 1 0 1
9	1	1	1	0	
	1	1	1	1	

Inconvénients

Certains chiffres ont plusieurs combinaisons (exemple le chiffre 1, 2, 3, etc.).

Remarque

- On voit que le code pondéré 6211 ne constitue pas un code BCD, car le chiffre décimal 5 n'est pas représenté en binaire ;
- Le code pondéré 5211, présente beaucoup d'inconvénients, car certains chiffres ont plusieurs représentations en binaire (exemple le chiffre 1, 2, 3, etc.).
- Le code pondéré 8421 appelé aussi Code naturel dans le BCD et est le seul parmi les 17 à garantir une représentation unique des chiffres décimaux.

	8	4	2	1
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 : $25_{(10)} = 0010\ 0101_{(BCD-8421)}$
 $908_{(10)} = 1001\ 0000\ 1000_{(BCD-8421)}$

→ Codes BCD symétriques

Le **code AIKEN** est un code pondéré dit auto complémentaire, car le complément à 9 du nombre décimal considéré peut être obtenu par simple permutation des digits 0 et 1 dans la représentation codée de ce nombre décimal en code AIKE N.

Les chiffres 6 et 3 étant complémentaires au chiffre 9, 6 se code donc comme le complément restreint ou complément à 1 de 3. Ce code est utilisé dans les calculateurs où il sert pour effectuer des opérations de soustraction par addition de la forme complémentaire.

Le **code Excédent 3** est un code non pondéré. Un code pondéré (exemple le code BCD-8421) peut être transformé en un code non pondéré auto complémentaire par l'adjonction d'une quantité constante (3 pour le code BCD) afin de faciliter certaines opérations de calcul notamment la soustraction. Le « zéro » sera codé par 0011, le « un » par 0100, etc.

CODE BINAIRE NATUREL	CODE AIKEN	CODE EXCEDENT 3
0 → 0000	0 → 0000	
1 → 0001	1 → 0001	
2 → 0010	2 → 0010	{ N'existent pas
3 → 0011	3 → 0011	
4 → 0100	4 → 0100	
5 → 0101		0 → 0011
6 → 0110		1 → 0100
7 → 0111		2 → 0101
8 → 1000		3 → 0110
9 → 1001		4 → 0111
10 → 1010		5 → 1000
11 → 1011	5 → 1011	6 → 1001
12 → 1100	6 → 1100	7 → 1010
13 → 1101	7 → 1101	8 → 1011
14 → 1110	8 → 1110	9 → 1100
15 → 1111	9 → 1111	} n'existent pas

Exemples

25 : 0010 1011 (code AIKEN)
 25 : 0101 1000 (code excédent 3)

→ Codes BCD détecteurs d'erreurs

La fiabilité recherchée dans les systèmes de transmission et de traitement des informations codées, notamment en présence des bruits, conduit à joindre à l'information nécessaire une information surabondante (redondante) permettant de détecter une ou plusieurs erreurs éventuellement.

Le contrôle de parité est une méthode d'emploi général très fréquemment utilisé :

CODE DECIMAL	CODE EXCEDENT 3 + BIT DE PARITE	CODE BIQUINAIRE 2 parmi 7
0	0011 0	10 00001
1	0100 1	10 00010
2	0101 0	10 00100
3	0110 0	10 01000
4	0111 1	10 10000
5	1000 1	10 00001
6	1001 0	10 00010
7	1010 0	10 00100
8	1011 1	10 01000
9	1100 0	10 10000

N.B.

- Parité paire : nombre de « 1 » est paire.
- P parmi n : code non pondéré qui présente la particularité d'avoir n éléments binaires dont p sont nécessairement dans l'état « 1 ». Ce code « p parmi n » permet la vérification de l'information (éléments binaires obligatoirement présents), détecteur d'erreurs. Permet également le décodage par porte à p entrées au lieu de n.

→ Codes Correcteurs d'erreurs (cas du Code Hamming)

Code Hamming est un exemple d'un code permettant de détecter et de corriger une erreur unique.

Soit la représentation codée d'un nombre 0 à 15 dans le tableau ci-dessous, parmi les 7 positions binaires. Les positions **3, 5, 6** et **7** sont les positions binaires normales (exposées souvent aux erreurs), avec comme poids respectivement : **8 – 4 – 2 – 1**.

Les 3 groupes de position :

1, 3, 5, 7
2, 3, 6, 7
4, 5, 6, 7

Constituent des groupes de contrôle, ils comportent normalement un nombre paire de « 1 », de sorte que la somme modulo 2 de leurs chiffres successifs est nulle.

Cette somme peut cesser d'être nulle en cas d'erreur et le nombre binaire formé par les sommes modulo 2 des chiffres des 3 groupes de contrôle indique le rang du chiffre erroné, qu'il suffit donc de changer.

Code HAMMING							
N	Position						
	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1
2	0	1	0	1	0	1	0
3	1	0	0	0	0	1	1
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	1
6	1	1	0	0	1	1	0
7	0	0	0	1	1	1	1
8	1	1	1	0	0	0	0
9	0	0	1	1	0	0	1
10	1	0	1	1	0	1	0
11	0	1	1	0	0	1	1
12	0	1	1	1	1	0	0
13	1	0	1	0	1	0	1
14	0	0	1	0	1	1	0
15	1	1	1	1	1	1	1

Exemple : Le chiffre $7_{(10)}$ $\rightarrow 0001111$ (code Hamming)

Rang 1 2 3 4 5 6 7

1, 3, 5, 7
2, 3, 6, 7
4, 5, 6, 7

$$\begin{aligned} \rightarrow 0 + 0 + 1 + 1 = 2_{(10)} &= 10_{(2)} \# 0 \\ \rightarrow 0 + 0 + 1 + 1 = 2_{(10)} &= 10_{(2)} \# 0 \\ \rightarrow 1 + 1 + 1 + 1 = 4_{(10)} &= 100_{(2)} \# 0 \end{aligned}$$

On prend le reste de la somme en binaire

En cas d'erreur, par exemple au lieu de la séquence : 0001111,

on a : 000**1**1111 (1^{er} cas)
ou : 00001**0**111 (2^{ème} cas)

1^{er} cas :

1, 3, 5, 7
2, 3, 6, 7
4, 5, 6, 7

$$\begin{aligned} \rightarrow 0 + 1 + 1 + 1 = 3_{(10)} &= 11_{(2)} \# 1 \\ \rightarrow 0 + 1 + 1 + 1 = 3_{(10)} &= 11_{(2)} \# 1 \\ \rightarrow 1 + 1 + 1 + 1 = 4_{(10)} &= 100_{(2)} \# 0 \end{aligned}$$

011

Erreur sur le 3^{ème} bit

2^{ème} cas :

Code du nombre 7 est : 0001111

Avec erreur, on : 0001101

Considérons les 3 groupes de contrôle et effectuons la somme modulo 2 de leurs chiffres pour retrouver le rang du chiffre faux :

Groupe de contrôle	sommes modulo 2
1357	$0 \oplus 0 \oplus 1 \oplus 1 = 0$
2367	$0 \oplus 0 \oplus 0 \oplus 0 = 1$
4567	$1 \oplus 1 \oplus 0 \oplus 1 = 1$

sens de lecture, 10

110 ce nombre binaire constitué par ces 3 sommes équivaut à $6_{(10)}$ et correspond au rang du chiffre faux.

2.4.2. REPRESENTATION DES ENTIERS SIGNES

❖ Types

Avec n bits, il est possible de coder les 2^n entiers non signés ($\in \mathbb{N}$) de 0 à $2^n - 1$. On parle de codage en « binaire pur » ou en « **binaire naturel** ».

Pour représenter un sous ensemble fini des entiers relatifs ($\in \mathbb{Z}$), il faut :

- Le codage en valeur absolue plus signe ;
- Le codage en complément en un ou en complément restreint ;
- Le codage en complément en deux ou en complément vrai.

❖ Codage en valeur absolue plus signe

Ce codage consiste à représenter la valeur absolue du nombre à coder sur les $(n-1)$ bits de droite d'un mot et à coder son signe sur le bit le plus à gauche avec (BLPS), par convention :

- Pour une valeur positive : 0 ;
- Pour une valeur négative : 1.

L'avantage de cette méthode est sa simplicité de codage. Toutefois, in 'est plus guère utilisée pour les entiers car elle complique quelque peu les opérations arithmétiques. Pour $n = 3$, on aura 8 possibilités de codage, soit 2^3 .

Code	Valeur
000	+0
001	+1
010	+2
011	+3
101	-1
110	-2
111	-3

❖ Codage en complément restreint

Pour trouver le complément à un d'un nombre binaire, on complémente tous ses bits, ce qui revient à changer en « 1 » tous les bits de valeur « 0 » et en « 0 » tous les bits de valeur « 1 ».

Exemple : complément à 1 de 01010100 est : **10101011**

Ici, les entiers positifs sont représentés en binaire pur sur les $(n-1)$ bits de droite avec « 0 » dans le $n^{\text{ième}}$ bits. Les entiers négatifs N sont représentés par $(2^n - 1) - N$. Il en résulte que la somme de deux valeurs opposées sera à $2^n - 1$ qui se code par n bits tout égaux à 1.

Dans la pratique, l'algorithme de codage est le suivant :

- coder la valeur absolue du nombre sur les $(n-1)$ bits de droite avec zéro pour le bit le plus à gauche (cas d'un nombre positif).

- Si le nombre est négatif, alors inverser les n bits précédemment obtenus, c'est-à-dire remplacer tous les « 0 » par des « 1 » et tous les « 1 » par des « 0 ».

Pour n=3

Code	Valeur
000	+0
001	+1
010	+2
011	+3
100	-3
101	-2
110	-1
111	-0

Règle de décodage : il faut calculer la valeur du nombre ainsi représenté en binaire pur avant de lui affecter son signe.

❖ Codage en complément Vrai

Le complément à deux d'un nombre entier est son complément à un et ajouter de « 1 ». Le bit de gauche est un bit de signe ; en revanche il n'y a plus qu'une seule représentation du zéro.

Code	Valeur
000	+0
001	+1
010	+2
011	+3
100	-4
101	-3
110	-2
111	-1

Exemple 1 : Trouver le complément à 2 de 10010110

Solution

$$\begin{array}{l} \text{Complément à 1 : } 01101001 \\ \text{Plus 1 : } +\underline{00000001} \\ \hline \textbf{01101010} \end{array}$$

Exemple 2 : coder en complément à deux et calculer la valeur $(15)_{10} - (10)_{10}$

$$\begin{array}{ll} \text{Solution} & (15)_{10} = (1111)_2 \\ & (10)_{10} = (1010)_2 \end{array}$$

Il faut au moins 5 bits pour coder les données puisque le bit de gauche est un bit de signe :
 $(15)_{10} = (01111)_2$
 $(-10)_{10} = (11010)_2 = (10101)_2$ en complément restreint

D'où :

$$\begin{array}{l} 01111 \text{ codage de 15} \\ + \underline{10110} \text{ complément à deux de } +10 \end{array}$$

00101 résultat correct sur 5 bits.

NOMBRE BINAIRE 8 BITS	NOMBRE DECIMAL EQUIVALENT AU BINAIRE NON SIGNE	NOMBRE DECIMAL EQUIVALENT AU BINAIRE SIGNE (COMPLEMENT A 2)
0000 0000	0	0
0000 0001	1	+1
0000 0010	2	+2
0000 0011	3	+3
«	«	«
«	«	«
«	«	«
0111 1110	126	+126
0111 1111	127	+127
1000 0000	128	-128
1000 0001	129	-127
«	«	«
«	«	«
«	«	«
1111 1110	252	-4
1111 1101	253	-3
1111 1110	254	-2
1111 1111	255	-1

REMARQUES

Les 8 bits permettent de représenter 256 nombres binaires signés sont 127 nombres positifs, le zéro et 128 nombres négatifs. Avec 16 bits, on représentera 65536 nombres binaires dont 32767 positifs, le zéro et 32768 nombres négatifs.

2.4.3. REPRESENTATION DES NOMBRES REELS

❖ Types

Le problème qui se pose est celui de la représentation d'une valeur comprenant une partie entière et une partie décimale (fractionnaire). Deux solutions sont envisageables :

- le codage en virgule fixe ;
- le codage en virgule flottante.

❖ Codage en virgule fixe

La représentation binaire en virgule fixe (fixed point) traite des nombres avec une position fixe de la virgule.

Exemple : Avec un codage à 8 bits et en complément à 2, on a

$$0,296875_{(10)} \rightarrow 0,1001100_{(2)}$$

$$-0,296875_{(10)} \rightarrow 10110100_{(2)}$$

❖ Codage en virgule flottante

La représentation binaire en virgule flottante (floating point) est un mode de représentation de données fractionnaires dans lequel la virgule occupe une position variable quelconque et où le nombre N est représenté par deux parties : sa mantisse et sa caractéristique (base + exposant).

$$N = M \cdot b^E$$

M : mantisse (précision)

b : base de l'exposant (en général 2 et parfois 16)

E : exposant par lequel il convient de multiplier M.

Exemples :

$$123000 \rightarrow 123 \times 10^3$$

$$-0,0034567 \rightarrow -34567 \times 10^{-7}$$

$$12_{(10)} \rightarrow 1100_{(2)} \rightarrow 0,11 \times 2^{100}$$

avec **M** = 11, **b** = 2 et **E** = 100

2.4.4. REPRESENTATION DES CARACTERES (=Codage source)

- CODE BAUDOT (Alphabet international N°2)

Le code Baudot, encore appelé code télégraphique à 5 unités (moments), fut universellement utilisé pour le réseau télégraphique commuté (télex). Avec un code à 5 bits, le nombre de combinaisons possibles est limité à $2^5=32$; ce qui est déjà insuffisant pour l'ensemble des lettres et chiffres ($26+10=36$) et des commandes (fin de ligne, ...).

Pour pallier à cela, on a réservé deux caractères particuliers « inversion lettres » et « inversion chiffre » pour désigner à tout moment dans quel sous -jeu de 30 caractères devront être interprétés les groupes de 5 bits successifs et on obtient par cet artifice $2 \times 30=60$ combinaisons possibles.

- CODE ASCII

Le Code ASCII (*American Standard Code for Information Interchange*) est le système de codage universel. Il représente en binaire les données du type caractères (alphanumériques et signes) et des mots de commande utilisés par les systèmes périphériques des micro-ordinateurs. Le code ASCII de base représentait les caractères sur 7 bits (c'est-à-dire **2=128 caractères représentables**, de 0 à 127). **Le huitième bit étant réservé à la parité.**

Certains constructeurs, dont IBM suivi par tous les fabricants, ont enrichi cette table en utilisant le 8ème caractère (ASCII étendu), ce qui double le nombre de caractères représentables ($2 \times 128 = 256$ caractères possibles).

<i>Msc. Ir. KAPULULA MUMBA Dubois</i>	Circuits Logiques Numériques Notes de Cours	UPL_2023-2024
---------------------------------------	--	---------------

Table des caractères ASCII

code	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
10	LF	VT	NP	CR	SO	SI	DLE	DC1	DC2	DC3
20	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
30	RS	US	SP	!	"	#	\$	%	&	'
40	()	*	+	,	-		f	0	1
50	2	3	4	5	6	7	8	9	:	:
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[\]	^	_	.	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	())	~	DEL		

- Code EBCDIC

Un autre code utilisé est le code EBCDIC (Extended Binary Decimal Interchange Code). C'est un code à 8 bits, sans élément binaire de parité. Il constitue l'alphabet de base adopté par IBM et d'autres constructeurs pour leurs calculateurs (BULL, ...).

II.5. ALGEBRE BINAIRE

2.5.1. Addition binaire

❖ Règles d'addition

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$1 + 1 = 0$ avec retenue positive 1 (report ou carry).

❖ Exemple

<i>Termes</i>	<i>Addition décimale</i>	<i>Addition binaire</i>
Retenue	:	0011 1100
Terme 1	:	1000 0110
+ Terme 2	:	+ 0101 1111
Somme	: 229	1110 0101

2.5.2. Soustraction

❖ Règles d'addition

$$0 - 0 = 0$$

$0 - 1 = 1$ avec retenue négative 1 (emprunt)

$$1 - 0 = 1$$

$$1 - 1 = 0$$

❖ Exemples

<i>Termes</i>	<i>Soustraction décimale</i>	<i>Soustraction binaire classique</i>	<i>Soustraction binaire signée</i>
Retenue	:	0000 0100	
Terme 1	:	0001 0101	
- Terme 2	:	- 0001 0011	+ 1110 1101
Déférence	: 2	0000 0010	0000 0010

Lorsque le premier terme est inférieur au second, il est impossible de faire l'opération de soustraction binaire classique, on recourt à la méthode de complément à 2 qui est valable dans les cas.

<i>Termes</i>	<i>Soustraction décimale</i>	<i>Soustraction binaire non signé</i>	<i>soustraction binaire signé</i>
Terme 1	:	0001 0011	0001 0011
- Terme 2	:	- 0001 0101	+ 1110 1011 (*)
Déférence	: - 2	1111 1110	1111 1110

(*) Complément à 2 de 21

Remarque

Les logiques des binaires non signés et du complément à 2 sont identiques.

Msc. Ir. KAPULULA MUMBA Dubois	Circuits Logiques Numériques	UPL_2023-2024
	Notes de Cours	

TRAVAUX DIRIGÉS

1. Quelle est la valeur entière décimale (maximale) pouvant être représentée à l'aide d'un demi quartet ?
2. Citer un phénomène et un système matériel dont le fonctionnement repose sur trois états logique de base.
3. Combien de bits sont-ils nécessaires pour coder l'alphabet français ?
4. Disposant de 102 mots, combien de bits sont-ils nécessaires pour les coder ? Quelle est la redondance de ce code ?
5. Est-ce que le système de numération romain constitue –t-il un système unaire ou évolué ?
6. Quel est le nombre maximal d'éléments dans un système de numération évolué ?
7. Par quoi est caractérisé un système de numération ?
8. Quels sont les nombres représentables par un mot à virgule flottante de 32 bits
9. Pour le concept byte, combien des symboles peuvent être représentés par un mot de 6 bits ?
10. Pourquoi « 486 » ne peut-il pas être un nombre octal ?
11. Quel est le nombre d'éléments (chiffres) que peut avoir un système évolué ?
12. A partir du tableau de code ASCII, exprimer en décimal, en hexadécimal, puis en binaire :
 - a) les caractères : «W» et «9»
 - b) les caractères : «w» et «CR».

Solution : format 7 6 5 4 3 2 1

- a) *pour « W » on a : 1010111 (en binaire) = 57 (en hexa) = 87 (en décimal).
*pour le caractère « 9 », on a : 00111001 (en binaire) = 39 (en hexa) = 57 (en décimal)
- b) *pour « w » on a : 0111 0111 (en binaire) = 77 (Hexa) = 119 (en décimal)
*pour « CR » on a : 0000 1101 = D = 13.

13. Exprimer le caractère « w » en code ASCII en binaire et en hexadécimal si le bit de parité est ajouté ?

Solution

En binaire, on a : 1101 0111 et en hexadécimal, on a D7H.

