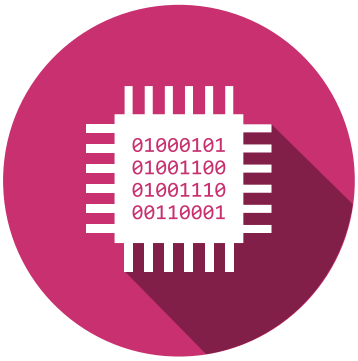


Représentations Numériques et Codes (NUM)

Cours Conception Numérique

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Orientation : Systèmes industriels (SYND)
Cours : Conception Numérique (Cnum)
Auteur : Christophe Bianchi, François Corthay, Pierre Pompili, Silvan Zahno
Date : 12 septembre 2023
Version : v2.1

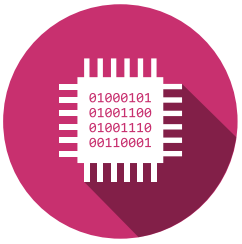
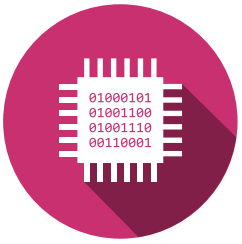


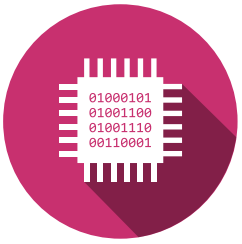
Table des matières



1 Introduction

Dans les sciences, dans les affaires et dans la plupart des domaines, nous sommes amenés à utiliser des grandeurs dont la valeur peut-être exprimée d'une manière analogique ou numérique. Les grandeurs **analogiques** sont caractérisées par le fait qu'elles varient graduellement à l'intérieur d'une gamme continue de valeurs. Les grandeurs **numériques** évoluent par contre de manière discontinue à l'intérieur d'un intervalle donné et prennent des valeurs discrètes représentées au moyen de symboles appelés chiffres.

Ce module décrit les représentations numériques les plus utilisées aujourd'hui et que l'on retrouve au sein d'une vaste gamme de produits électroniques tels que les jeux vidéo, les appareils de laboratoires, les calculatrices et bien sûr les ordinateurs.



2 Systèmes de Numération

2.1 Représentation générale des nombres entiers

Un nombre entier positif A, appelé aussi nombre logique (unsigned number) peut être représenté dans un système de numération de base p par une suite de chiffres a_i compris entre 0 et $p-1$ tel que :

$$A = \sum_{i=0}^{n-1} a_i * p^i \qquad 0 \leq a_i \leq p-1 \tag{1}$$

Par convention, pour une base donnée, le nombre A est représenté en numération de position, par :

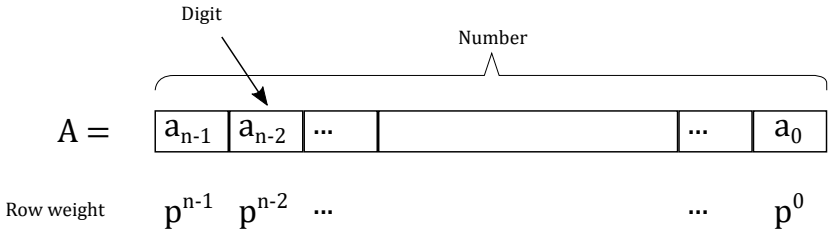


FIGURE 1 – Binary Representation

où a_0 est le chiffre de poids le plus faible **L**owest **S**ignificant **D**igit (LSD) et où a_{n-1} est le chiffre de poids le plus fort **H**ighest **S**ignificant **D**igit (HSD). Ainsi la valeur d'un chiffre dans un nombre dépend de sa position (rang) dans le nombre. Tout nombre peut donc être représenté par la somme des produits de chaque chiffre par le poids de son rang dans le nombre. Avec une telle représentation, A peut donc prendre toutes les valeurs entières possibles telles que :

$$0 \leq A \leq p^n \tag{2}$$

De nombreux systèmes de numération sont utilisés en technologie numérique. Les plus courants sont : le système décimal, le système binaire et le système hexadécimal.

2.2 Système décimal

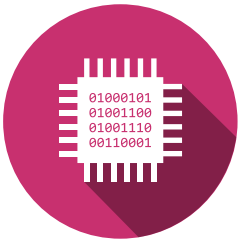
Ce système s'est imposé tout naturellement à l'homme puisque ce dernier possède dix doigts. Dans ce système la base p est égale à 10 et nous pouvons représenter un nombre à l'aide de 10 symboles ou chiffres qui sont 0,1,2,3,4,5,6,7,8 et 9. Ainsi le nombre décimal représenté par 245_{10} a comme valeur :

$$245_{10} = 2 * 10^2 + 4 * 10^1 + 5 * 10^0 \tag{3}$$

L'indice 10 indique la base qui a été choisie pour représenter le nombre, ici la base dix. Nous pouvons également indiquer la base dix par l'indice d.

2.3 Système binaire

Le système décimal étant malheureusement difficile à adapter aux mécanismes numériques, nous introduisons ici un système de base 2, plus simple, composé que de deux symboles '0' et '1' et que



nous appelons système binaire. Ce système binaire est directement assimilable au fonctionnement d'un transistor dont un des états représente le niveau de tension logique '0' (interrupteur fermé) et l'autre état le niveau de tension logique '1' (interrupteur ouvert).

Dans un tel système le nombre binaire représenté par 11110101_2 a comme valeur :

$$11110101_2 = 1 * 2^7 + 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 \quad (4)$$

Les symboles '0' et '1' sont appelés bit, abbréviation de **bi**nary digi**t**. Le bit de poids fort situé à gauche du nombre est appelé **M**ost **S**ignifiant **B**it (MSB), alors que le bit de poids faible situé à droite du nombre est appelé **L**east **S**ignifiant **B**it (LSB). Un groupe de huit bits forme un byte.

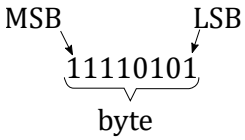


FIGURE 2 – Représentation binaire d'un byte

L'indice 2 indique la base qui a été choisie pour représenter le nombre, ici la base deux. Nous pouvons également indiquer la base deux par l'indice *b*.

2.4 Système hexadécimal

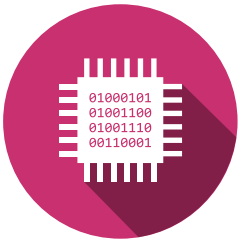
Le système binaire présente l'inconvénient d'être difficile à manipuler par l'homme, car les nombres sont représentés dans ce système par de longues suites de '1' et de '0' qu'il est difficile de lire ou d'interpréter sans erreurs. Afin d'écrire le nombre codé en binaire sous une forme plus compacte, nous choisissons de le représenter dans la base 16. Le fait de choisir cette base qui est issue d'une puissance de deux nous permet de réaliser des conversions faciles avec le système binaire.

Le système hexadécimal se compose de 16 symboles notés 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E et F. Le nombre hexadécimal représenté par $F5_{16}$ a donc comme valeur :

$$F5_{16} = F * 16^1 + 5 * 16^0 = 15 * 16^1 + 5 * 16^0 \quad (5)$$

L'indice 16 indique la base qui a été choisie pour représenter le nombre, ici la base seize. Nous pouvons également indiquer la base seize par l'indice *h*.

La Table 1 donne pour les nombres décimaux de 0 à 15 ainsi que leur équivalent hexadécimal et binaire.



Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

TABLE 1 – Représentation des nombres entiers compris entre 0 et 15 à l'aide du système décimal, hexadécimal et binaire

3 Conversion d'un systeme de numération à un autre

3.1 Conversion binaire – décimal

De part la définition de la représentation générale d'un nombre, tout nombre binaire peut être transformé en son équivalant décimal simplement en additionnant les poids des diverses positions où se trouve une valeur 1. Reprenant l'équation 6 nous obtenons :

$$\begin{aligned} 11110101_2 &= 2^7 + 2^6 + 2^5 + 2^4 + 2^2 + 2^0 \\ &128 + 64 + 32 + 16 + 4 + 1 = 245_{10} \end{aligned} \tag{6}$$

3.2 Conversion décimal – binaire

Il existe deux méthodes de convertir un nombre décimal en son équivalant binaire. Une première façon qui s'applique facilement à des petits nombres consiste à rechercher les puissances de 2 à prendre en compte pour que leur somme fournisse le nombre décimal donné. Voici un exemple :

$$77_{10} = \ldots + \cancel{128} + \underline{64} + \cancel{32} + \cancel{16} + \underline{8} + \underline{4} + \cancel{2} + \underline{1} = 01001101_2 \tag{7}$$

La deuxième méthode, qui convient mieux aux grands nombres, consiste à répéter la division par deux du nombre décimal donné jusqu'à ce que le quotient soit 0. Le nombre binaire se compose alors des restes successifs de chacune des divisions sachant que le premier reste trouvé correspond au bit de poids faible et que le dernier reste correspond au bit de poids fort. La Figure 3 illustre une conversion d'un nombre décimal en un nombre binaire.

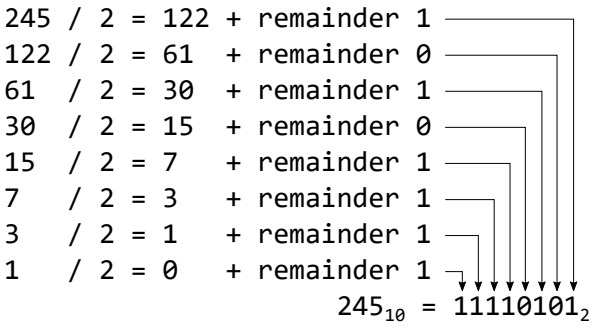


FIGURE 3 – Conversion décimal – binaire

3.3 Conversion hexadécimal – binaire

Le nombre hexadécimal se veut une façon abrégée de représenter les nombres binaires. Ainsi, la conversion d'un nombre hexadécimal en un nombre binaire consiste simplement à remplacer chaque chiffre hexadécimal dans son équivalent binaire sur 4 bits. Par exemple, la conversion de $F5_{16}$ en binaire est donnée par 11110101_2 :

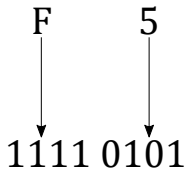


FIGURE 4 – Conversion hexadécimal - binaire

3.4 Conversion binaire – hexadécimal

La conversion d'un nombre binaire en un nombre hexadécimal est l'inverse de la méthode présentée sous le paragraphe 3.3. Il suffit donc de regrouper les bits par quatre en partant du bit de poids faible, puis de convertir ces groupes de quatre en leur équivalant hexadécimal. Si le nombre de bit du nombre binaire n'est pas un multiple de quatre alors il suffit de compléter le nombre par des zéros à gauche du bit de poids fort. A titre d'exemple la conversion du nombre 11110101_2 donne en hexadécimal $F5_{16}$:

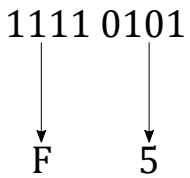
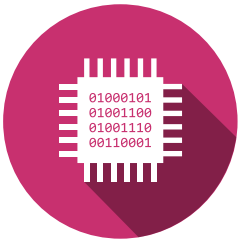


FIGURE 5 – Conversion binaire - hexadécimal

3.5 Conversion hexadécimal - décimal

De part la définition de la représentation générale d'un nombre, tout nombre hexadécimal peut être transformé en son équivalant décimal simplement en additionnant les produits formés de chaque chiffre hexadécimal et de leur poids positionnel correspondant. Reprenant l'équation 5,nous obtenons :



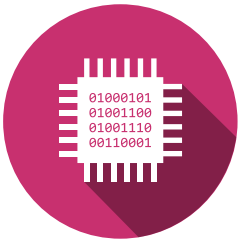
$$F5_{16} = 15 * 16^1 + 5 * 16^0$$
$$15 * 16 + 5 * 1 = 245_{10} \quad (8)$$

3.6 Conversion décimal - hexadécimal

De la même façon que celle présentée dans le paragraphe 3.2, la méthode de conversion d'un nombre décimal à un nombre hexadécimal consiste à répéter la division par 16 du nombre décimal donné jusqu'à ce que le quotient soit 0. Le nombre hexadécimal se compose alors des restes successifs de chacune des divisions sachant que le premier reste trouvé correspond au digit de poids faible et que le dernier reste correspond au digit de poids fort.

```
245 / 16 = 15 + remainder 5
15  / 16 = 0  + remainder 15
                  245₁₀ = F5₁₆
```

FIGURE 6 – Conversion décimal - hexadécimal



4 Opérations sur les nombres logiques

4.1 Addition de nombres logiques

Soit A et B deux nombres binaires entiers positifs de n bits :

$$A = a_{n-1}a_{n-2}\dots a_0 = \sum_{i=0}^{n-1} a_i 2^i \qquad 0 \leq a_i \leq 1 \tag{9}$$

$$B = b_{n-1}b_{n-2}\dots b_0 = \sum_{i=0}^{n-1} b_i 2^i \qquad 0 \leq b_i \leq 1 \tag{10}$$

la somme de ces deux nombres est égale à :

$$S = \sum_{i=0}^{n-1} (a_i + b_i)2^i = c_n * 2^n + \sum_{i=0}^{n-1} c_i 2^i \qquad 0 \leq c_i \leq 1 \tag{11}$$

Remarquons que la somme S de deux nombres de n bits est un nombre entier de $n+1$ bits au maximum ; c'est le cas si a_{n-1} et b_{n-1} sont égaux à '1'.

L'addition binaire s'effectue ainsi de la même manière qu'une addition décimale. La Figure 7 représente une addition binaire sur 4 bits.

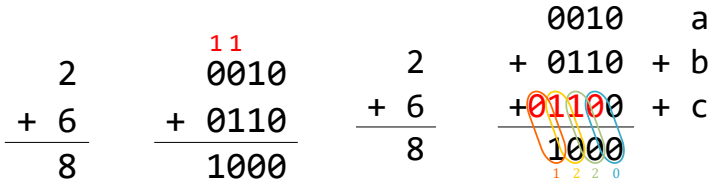


FIGURE 7 – Addition binaire de deux nombres logiques

4.2 Soustraction de nombres logiques

La soustraction binaire s'effectue de la même manière qu'une soustraction décimale. La Figure 8 représentent une soustraction binaire sur 4 bits.

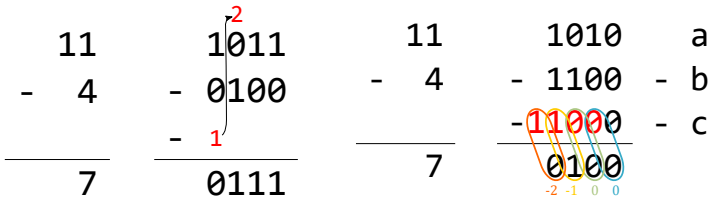
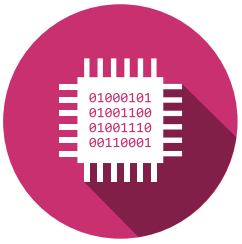


FIGURE 8 – Soustraction binaire de deux nombres logiques

4.3 Principe de multiplication

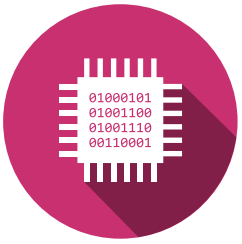
L'opération de multiplication, de par sa complexité, est toujours décomposée en opérations plus simples. A l'école primaire on effectue la multiplication en décalant le multiplicande multiplié par le chiffre correspondant du multiplicateur. La Figure 9 illustre une multiplication selon cette méthode. On remarquera que deux mots de n bits sont utilisés pour produire un résultat codé sur $2n$ bits.




```
  11          1011
x 13        x 1101
143       00000000
        +     1011
          00001011
        +    0000
          00001011
        +   1011
          00110111
        +  1011
          10001111
```

FIGURE 9 – Multiplication binaire

Ainsi, la multiplication de nombres peut être réalisée par une suite de décalages à gauche et d'additions.



5 Codes

Le codage consiste à faire correspondre à des nombres, des lettres ou des mots un groupe spécial de symboles que l'on appelle code.

5.1 Code Décimal Codé Binaire

La plupart du temps, les problèmes soumis à l'ordinateur sont formulés dans le système décimal, et l'utilisateur attend de l'ordinateur une réponse exprimée dans ce même système décimal. Or, en pratique l'ordinateur traite des informations binaires ce qui implique, lors des opérations d'entrée ou de sortie de données, la mise en œuvre d'opérations de conversion. Lorsque le traitement porte sur des nombres fractionnaires, les limitations de précision sur les calculs exécutés en ordinateur conduisent à effectuer des arrondis ; ceux-ci diffèrent selon qu'ils sont effectués en binaire ou en décimal.

Ainsi, en informatique de gestion par exemple, il est nécessaire que l'ordinateur donne un résultat avec une erreur d'arrondi calculée en décimal. Pour ce faire, on représente chaque chiffre d'un nombre décimal par son équivalent binaire, on obtient ainsi le code décimal codé binaire **B**inary **C**oded **d**igit (BCD). Comme le plus élevé des chiffres décimaux est 9, il faut donc 4 bits pour coder chaque chiffre décimal de 0 à 9. Ainsi, le nombre décimal 245 est codé en DCB par $0010'0100'0101_{BCD}$:

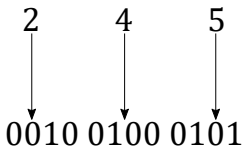
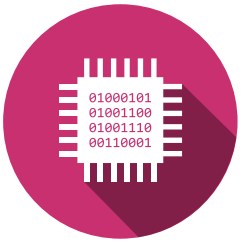


FIGURE 10 – Conversion décimal - BCD

Avec le système BCD, les seize combinaisons du code binaire à quatre bits qui représentent les différents chiffres ne sont pas toutes utilisées, puisque chaque chiffre à coder est compris entre 0 et 9. De ce fait, la représentation d'un nombre en BCD exige plus de bits que la représentation du même nombre avec un code binaire pur. Le code BCD présente en outre l'inconvénient de nécessiter des circuits arithmétiques plus complexes puisque les opérations au niveau des bits sont réalisées différemment de celles qui sont effectuées au niveau des chiffres. Pour ces raisons, le code BCD n'est employé que pour les applications où il est absolument nécessaire d'effectuer les calculs dans le système décimal.

5.2 Code Gray

En informatique industrielle, on a souvent besoin de capteurs qui mesurent les déplacements linéaires ou angulaires d'une pièce. On réalise ces capteurs sous forme de codeurs absolus capables de délivrer la position de l'objet sous forme d'un mot binaire directement utilisable par l'ordinateur. La Figure 11 montre une réalisation possible d'un codeur linéaire optique.



Decimal	Binary	Gray Code
$d_1 d_0$	$b_3 b_2 b_1 b_0$	$g_3 g_2 g_1 g_0$
0	0000	0000
1	0001	000$\underline{1}$
2	0010	0011
3	0011	00$\underline{10}$
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0$\underline{100}$
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

TABLE 2 – Construction du code de Gray binaire réfléchi

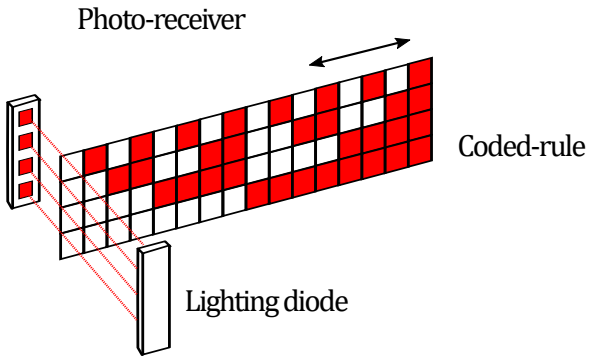


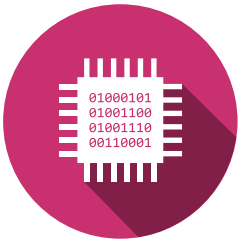
FIGURE 11 – Codeur linéaire optique

Un code de Gray est un codeur linéaire particulier conçu de telle sorte qu'un seul chiffre du nombre change lorsqu'on passe d'un mot au suivant. Ceci permet de garantir que la valeur lue au moment de la transition ne peut être que celle du mot précédant ou du mot suivant, à l'exclusion de toute valeur intermédiaire aberrante. Le code de Gray le plus utilisé est le code binaire réfléchi.

5.2.1 Conversion code binaire pur – code de Gray binaire réfléchi

Le code de Gray binaire réfléchi est construit par symétries successives.

Pour le construire rapidement, on débute par la colonne la plus à droite en posant les lignes 0 et 1. A l'endroit indiqué dans la table 2, nous plaçons un miroir qui nous permet de construire les lignes 2 et 3. En plaçant un nouveau miroir, nous construisons les lignes 4 à 7, et ainsi de suite jusqu'à l'obtention du code complet.



5.2.2 Conversion code Gray – code binaire pur

Le code de Gray est un code non pondéré, c'est-à-dire que les positions binaires des groupes codés ne sont affectées d'aucun poids. C'est pourquoi ce code ne convient pas du tout aux calculs arithmétiques, mais se trouve surtout dans des applications d'entrée et de sortie de systèmes numériques. Les mots codés en code Gray doivent donc être convertis en binaire pur par l'ordinateur avant d'être exploités. Le problème de conversion revient à déterminer les bits bi en binaire pur, en fonction des bits gi du code de Gray. Nous avons vu au paragraphe précédant que le code Gray se construit par symétries successives à partir des bits de poids faibles, selon la méthode de la table 2. On voit immédiatement que :

$$b_{n-1} = g_{n-1} \tag{12}$$

Les bits successifs b_{n-i} du nombre codé en binaire pur se calculent par récurrence tenant compte de l'équation suivante :

$$b_{n-i} = b_{n-i+1} \oplus g_{n-i} \tag{13}$$

où la fonction XOR peut être réalisée par une addition modulo 2. Sur 4 bits, la conversion du code de Gray binaire réfléchi au code binaire s'écrit donc à l'aide des 4 équations suivantes :

$$\begin{aligned} b_3 &= g_3 \\ b_2 &= b_3 \oplus g_2 = g_3 \oplus g_2 \\ b_1 &= b_2 \oplus g_1 = g_3 \oplus g_2 \oplus g_1 \\ b_0 &= b_1 \oplus g_0 = g_3 \oplus g_2 \oplus g_1 \oplus g_0 \end{aligned} \tag{14}$$

Ainsi le code de Gray binaire réfléchi 1010_{gray} correspond au code binaire pur 1100_2.

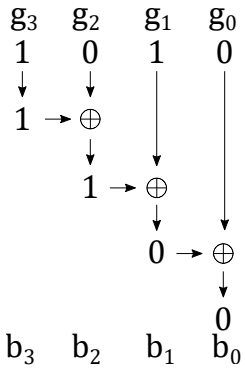
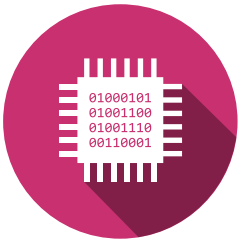


FIGURE 12 – Conversion code de Gray binaire réfléchi – code binaire

5.3 Code ASCII

En informatique, les lettres d'un fichier texte sont spécifiées par un code binaire. Le code **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange (ASCII) a été défini sur 7 bits, comme on le voit dans la table 3 et **??**, avec les 3 bits de poids fort pour sélectionner les colonnes et les 4 bits de poids faible pour sélectionner les lignes.

Ainsi, le caractère 'a'a un code ASCII de $110'0001_b$ ou 61_h. Les codes ASCII des deux premières colonnes spécifient des caractères spéciaux, comme par exemple **C**arriage **R**eturn (CR) pour revenir

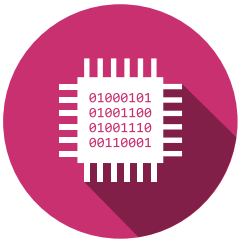


					000	001	010	011	100	101	110	111
					0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	‘	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	”	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	’	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	|
1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	_	o	DEL

TABLE 3 – Définition des codes 7bit-ASCII

en début de ligne et **L**ine **F**eed (LF) pour se déplacer à la ligne suivante.

Cette représentation ne compte aucun caractère accentué. Elle a été dans un premier temps augmentée sur 8 bits pour inclure des caractères supplémentaires ainsi que des caractères graphiques comme des bordures de tables ou des flèches. Aujourd'hui, les caractères unicode permettent de coder les caractères de toutes les langues connues ainsi que d'autres symboles graphiques, comme les emoji 😊.



6 Représentation des nombres signés

Les nombres qui sont susceptibles d'être positifs ou négatifs sont appelés nombres arithmétiques ou parfois nombres signés, par opposition aux nombres logiques qui sont considérés comme des nombres toujours positifs. Pour faciliter la manipulation des nombres signés et simplifier les opérations mathématiques plusieurs types de représentation existent :

- représentation par signe-amplitude
- représentation biaisée
- complément à 1
- complément à 2

6.1 Représentation par signe – amplitude

La méthode la plus naturelle de représentation des nombres négatifs consiste à faire précéder le nombre par un bit dont la valeur est par exemple '0' lorsque le signe est positif et '1' lorsque le signe est négatif. Ainsi, dans un format à 8 bits, le nombre décimal -125 s'écrit en binaire comme indiqué en 13. Cette représentation de type signe-amplitude présente l'avantage de permettre une détermination aisée du signe du nombre par simple consultation du bit le plus significatif.

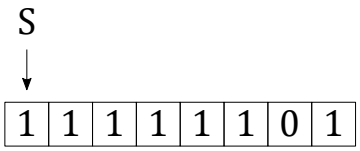


FIGURE 13 – Représentation par signe-amplitude

On voit immédiatement qu'un format à n bits permet de représenter en binaire tous les nombres entiers A tels que :

$$-(2^{n-1}-1) \leq A \leq 2^{n-1}-1 \tag{15}$$

avec une double représentation $+0$ et -0 pour la valeur zéro.

6.2 Représentation biaisée

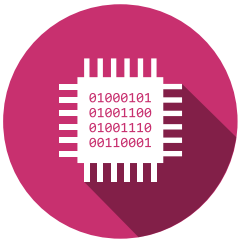
Afin d'éviter cette double représentation du zéro, nous pouvons introduire la représentation biaisée qui consiste à coder un nombre entier positif ou négatif A sous la forme d'un nombre N tel que :

$$N = A + R \tag{16}$$

où R est un biais positif choisi de telle sorte que N soit toujours positif pour tout le domaine de variation possible de A.

En pratique, on désire que le domaine de variation possible des nombres négatifs soit approximativement le même que celui des nombres positifs ; pour ce faire la valeur du biais R est fonction du nombre de bits n utilisés :

$$R = 2^{n-1}-1 \tag{17}$$



Value	Shift decimal	Decimal	Shift binary
Min	0	−127	0000 0000
	1	−126	0000 0001
...			
	126	−1	0111 1110
Zero	127	0	0111 1111
	128	1	1000 0000
...			
	254	127	1111 1110
Max	255	128	1111 1111

TABLE 4 – Représentation biaisée sur huit bits

Ainsi, une notation en binaire à 8 bits, biaisée avec $R = 127$, couvre le domaine des entiers compris entre -127 et $+128$ comme l'illustre la Table 4.

La notation biaisée conduit à une représentation unique du zéro, qui correspond ici à $N = R$. La représentation biaisée n'est en fait utilisée que pour les exposants des nombres en virgules flottantes.

6.3 Complément à 1

L'addition de deux nombres positifs ne posant aucun problème ; nous pouvons imaginer appliquer à nouveau la représentation biaisée, mais cette fois appliquée uniquement aux nombres négatifs. Avec cette méthode, les nombres positifs restent inchangés, et tout nombre négatif $-B$ (avec $B \leq 0$) est remplacé par son complément $R - B$, où R est un biais choisi de telle sorte que $R - B$ soit toujours positif quelle que soit la valeur de B. Pour un nombre binaire de n bits, le biais utilisé pour généré le complément à 1 est tel que :

$$R = 2^n - 1 \tag{18}$$

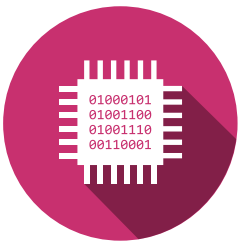
On voit immédiatement qu'un format à n bits permet de représenter en complément à 1 tous les nombres entiers A tels que :

$$-(2^{n-1} - 1) \leq A \leq 2^{n-1} - 1 \tag{19}$$

Le bit le plus significatif du mot est '0' pour les nombres positifs et '1' pour les nombres négatifs, ce qui permet une détection facile du signe. La détection de la valeur zéro est plus délicate, car la représentation est double, avec une valeur dite zéro positif lorsque tous les bits du mot sont à '0', et zéro négatif lorsque tous les bits du mot sont à '1'.

6.3.1 Changement de signe par transformation en complément à 1

En pratique, il s'avère que le complément à 1 d'un nombre négatif se calcule très simplement par inversion logique de tous les bits du nombre positif de même valeur d'amplitude. Ainsi, 125_{10} s'exprime en complément à 1 par :



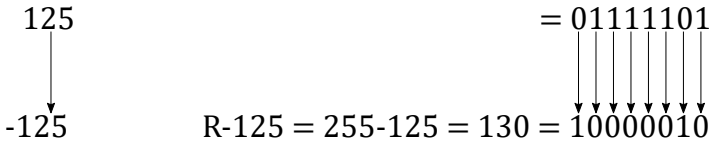


FIGURE 14 – Changement de signe complément à 1

Il s'ensuit que la complémentation à 1 d'un nombre signé transforme un nombre positif en un nombre négatif et vice versa.

6.4 Complément à 2

La méthode du complément à 2 est basée sur le même principe général que celui que nous avons décrit ci dessus, mais en choisissant ici un biais R tel que :

$$R = 2^n \tag{20}$$

On peut démontrer que dans la pratique le complément à 2 d'un nombre se calcule en inversant tous les bits du mot, ce qui donne le complément à 1, et en ajoutant $+1$ au résultat. Ainsi, 125_{10} s'exprime en complément à 2 par :

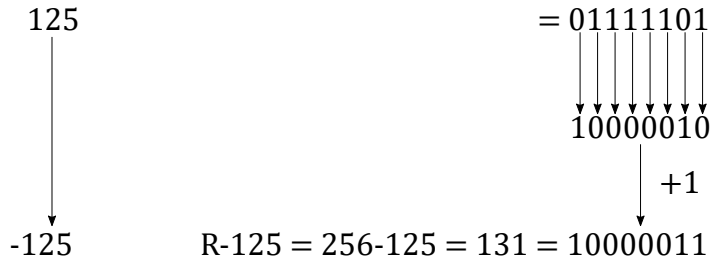


FIGURE 15 – Changement de signe complèment à 2 (1)

On voit immédiatement qu'un format à n bits permet de représenter en complément à 2 tous les nombres entiers A tels que :

$$-2^{n-1} \leq A \leq 2^{n-1} - 1 \tag{21}$$

Le bit le plus significatif du mot est '0' pour les nombres positifs et '1' pour les nombres négatifs, ce qui permet une détection facile du signe. La représentation unique du zéro, qui correspond au cas où tous les bits du mot sont à '0', facilite sa détection.

6.4.1 Changement de signe par transformation en complément à 2

En pratique, une méthode simple permet d'obtenir le complément à 2 d'un nombre binaire. On examine les bits à partir de la droite (poids faible) et on parcourt le nombre de droite à gauche en appliquant la règle suivante : tous les bits rencontrés jusqu'au premier '1' y compris sont conservés, tous les suivants sont inversés.

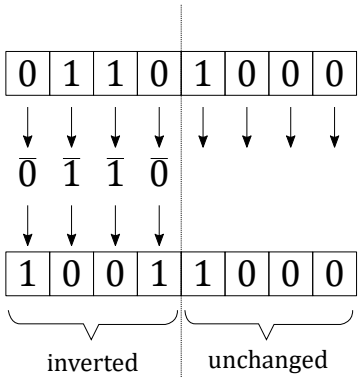
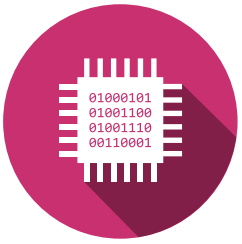


FIGURE 16 – Changement de signe complèment à 2 (2)

Il s'ensuit que la complémentation à 2 d'un nombre signé transforme un nombre positif en un nombre négatif et vice versa. Le complément à 2 tel que décrit ci-dessus peut être facilement manipulé par une personne. En fait, les systèmes numériques implémentent ce complément à 2 au moyen de portes logiques et en utilisant le complément à 1 $+1$.



Références

[1] Suhail Almani. *Electronic Logic Systems*. second edition. New-Jersey : Prentice-Hall, 1989.
[2] K Beuth. *Digitaltechnik*. 11. Auflage. Vogel Buchverlag, 2001. isbn : 3-8023-1755-6.
[3] Clive Maxfield. *Bebop to the Boolean Boogie*. Elsevier, 2009. isbn : 978-1-85617-507-4. doi : 10.1016/B978-1-85617-507-4.X0001-0. url : https://linkinghub.elsevier.com/retrieve/pii/B9781856175074X00010 (visité le 27/05/2021).
[4] Henry Nussbaumer. *Informatique Industrielle 1 : Représentation et Traitement de l'information*. Presses polytechniques romandess, 1986.
[5] Ronald J. Tocci. *Circuits numériques : théorie et applications*. 2ème édition. Éditions R. Goulet, 1987. 549 p. isbn : 978-2-89249-204-0. Google Books : 0F8_ygAACAAJ.
[6] John F. Wakerly. *Digital Design : Principles and Practices*. 3rd ed. Upper Saddle River, N.J : Prentice Hall, 2000. 949 p. isbn : 978-0-13-769191-3.

Acronymes

ASCII **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange. 1, 12, 13

BCD **B**inary **C**oded **d**igit. 10
bit **bi**nary digi**t**. 4

CR **C**arriage **R**eturn. 12

HSD **H**ighest **S**ignificant **D**igit. 3

LF **L**ine **F**eed. 13
LSB **L**east **S**ignifiant **B**it. 4
LSD **L**owest **S**ignificant **D**igit. 3

MSB **M**ost **S**ignifiant **B**it. 4

Glossaire

unsigned number unsigned number can only represent non-negative numbers (zero or positive numbers). 3