

# 1 Code binaire

## 1.1 Introduction

## 1.2 Base

Actuellement, notre système numérique s'est généralisé sur une *base 10*, ou encore *système décimal*. Celui-ci nous est naturellement venu que nous comptons avec nos dix doigts. Cela signifie que l'on peut compter jusqu'au *chiffre 9* avant de devoir former un *nombre* pour passer au rang supérieur (dizaine, centaines...). On utilise d'autres systèmes numériques ou dérivés pour certains usages, comme les minutes où les heures sur une base 60 ou les années sur une base 12.

### Définition 1.1: Base

Base dont le nombre indique le nombre de symboles distincts dans système de numération. Dans une base X, chaque symbole doit être strictement inférieur à X et chaque nombre peut être écrit sa forme polynomiale.

### Exemple: Bases

base 10 / système décimal : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9

base 2 / système binaire : 0 et 1

base 16 / système hexadécimal : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 et 15

En base 10, nous pouvons dès lors décomposer un nombre sous sa forme *polynomiale* :

$$7962,56 = 7 \cdot 10^3 + 9 \cdot 10^2 + 6 \cdot 10^1 + 2 \cdot 10^0 + 5 \cdot 10^{-1} + 6 \cdot 10^{-2}$$

Jusqu'à la fin du Moyen Âge, le système numérique utilisé était en base 20, la base 10 importée avec les nombres arabes n'étant pas encore utilisée. On trouve des traces de cette base dans la langue française avec par exemple le nombre quatre-vingt, qui est resté malgré le changement de base.

## 1.3 Système binaire

Les premières traces écrites du système binaire remontent à 1703, dans **Leibnitz 1703**. L'auteur fait remonter à la Chine ancienne les origines du système binaire avec les *hexagrammes* présentés dans le *Yi Jing*, un traité de divination chinois datant du 1<sup>er</sup> millénaire av. J.-C.. Ces hexagrammes correspondaient aux nombres binaires de 0 à 111 111. Il a placé cette numération au centre de sa théologie, avec l'idée symbolique de la *creatio ex nihilo* chrétienne (création à partir de rien). Pour autant, il n'a pas trouvé d'utilité à ce système numérique.

En 1807, George Boole publie un article, *L'analyse mathématique de la logique*, décrivant un système algébrique de la logique, prénommé maintenant *algèbre de Boole*. Il s'agit du point de départ du code informatique et des portes logiques, dont l'usage fut théorisé par Claude Shannon en 1937.

Pour toute donnée traitée par un système informatique, la base utilisée est la *base 2*, ou encore le *système binaire*. Cela est dû au fait que les systèmes informatiques sont constitués de composants



électronique soumis à une tension ou non, ils ne peuvent que présenter deux états. Le système binaire permet de traduire numériquement ces deux états, avec le chiffre 0 qui traduit une absence de tension et le chiffre 1 qui traduit une présence de tension. Ce système est à la base des *opérations logiques de base*, que l'on retrouve dans la pratique en automatisme.

# 1.4 Écriture binaire

Pour écrire en base 2, il faut donc passer de rang une fois le chiffre 1 atteint, pour former un nombre. Les chiffres en système binaire sont appelés des *bit* (en minuscule) pour *binary digit*. Comme dans un nombre décimal, le bit le plus à gauche est le appelé le bit du poids fort, celui le plus à droite le bit du poids faible.

Valeur en décimal	Équivalent en binaire	Valeur en décimal	Équivalent en binaire
0	0	10	1010
1	1	11	1011
2	10	12	1100
3	11	13	1101
4	100	14	1110
5	101	15	1111
6	110	16	10000
7	111	17	10001
8	1000	18	10010
9	1001	19	10011

TAB. 1.1 – Nombre en binaire avec équivalent décimal

On peut également écrire :

$$(19)_{10} = (10011)_2$$

# 1.5 Conversion binaire/décimal

## 1.5.1 Méthode 1 : puissance de deux

Pour convertir un nombre décimal en binaire, il faut décomposer le nombre décimal en somme de puissance de 2.

...	2 <sup>12</sup>	2 <sup>11</sup>	2 <sup>10</sup>	2 <sup>9</sup>	2 <sup>8</sup>	2 <sup>7</sup>	2 <sup>6</sup>	2 <sup>5</sup>	2 <sup>4</sup>	2 <sup>3</sup>	2 <sup>2</sup>	2 <sup>1</sup>	2 <sup>0</sup>
...	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

Si l'on prend par exemple le nombre 26, il faut le décomposer avec des nombres qui sont des puissances de deux en partant en débutant toujours avec le plus élevé :

### Exemple 1.1: Décomposition d'un réel en puissance de deux

$$26 = 16 + 8 + 2$$

$$26 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$$



Ce qui donne :

$$(26)_{10} = (11010)_2$$

Cette méthode est simple à comprendre mais peut se révéler fastidieuse pour la conversion de grand nombre.

### 1.5.2 Méthode 2 : division euclidienne

Dans cette méthode de conversion, il convient d'effectuer des divisions euclidiennes successives du nombre décimal par 2, le reste de chaque division formera le nombre binaire.

#### Exemple 1.2: Conversion d'un réel entier d'une base 10 à une base 2

$$\begin{array}{r|l}
 568_{10} : & \\
 2 \overline{) 568} & 0 \\
 2 \overline{) 284} & 0 \\
 2 \overline{) 142} & 0 \\
 2 \overline{) 71} & 1 \\
 2 \overline{) 35} & 1 \\
 2 \overline{) 17} & 1 \\
 2 \overline{) 8} & 0 \\
 2 \overline{) 4} & 0 \\
 2 \overline{) 2} & 0 \\
 2 \overline{) 1} & 1
 \end{array}
 \left. \vphantom{\begin{array}{r|l} 568_{10} : \\ 2 \overline{) 568} \\ 2 \overline{) 284} \\ 2 \overline{) 142} \\ 2 \overline{) 71} \\ 2 \overline{) 35} \\ 2 \overline{) 17} \\ 2 \overline{) 8} \\ 2 \overline{) 4} \\ 2 \overline{) 2} \\ 2 \overline{) 1} \end{array}} \right\} = 1000111000_2$$

$$\begin{array}{r|l}
 34_{10} : & \\
 2 \overline{) 34} & 0 \\
 2 \overline{) 17} & 1 \\
 2 \overline{) 8} & 0 \\
 2 \overline{) 4} & 0 \\
 2 \overline{) 2} & 0 \\
 2 \overline{) 1} & 1
 \end{array}
 \left. \vphantom{\begin{array}{r|l} 34_{10} : \\ 2 \overline{) 34} \\ 2 \overline{) 17} \\ 2 \overline{) 8} \\ 2 \overline{) 4} \\ 2 \overline{) 2} \\ 2 \overline{) 1} \end{array}} \right\} = 100010_2$$

Dans le cas d'un nombre réel avec partie fractionnelle :

- la partie entière est transformée en effectuant des divisions successives ;
- la partie fractionnelle est transformée en effectuant des multiplications successives. Le chiffre avant la virgule de cette multiplication donnera le bit et il faut multiplier le résultat en ne conservant que la partie fractionnelle, la partie entière est remise à zéro.

La conversion d'un nombre réel avec partie fractionnelle s'effectue avec une marge de précision, plus la partie fractionnelle sera multipliée par deux, plus précise sera la conversion.

#### Exemple 1.3: Conversion d'un réel fractionnel d'une base 10 à une base 2

$$(0,7)_{10} = (?)_2 \rightarrow (0,7)_{10} = (0,10110)_2$$

$$0,7 \cdot 2 = 1,4$$

$$0,4 \cdot 2 = 0,8$$

$$0,8 \cdot 2 = 1,6$$

$$0,6 \cdot 2 = 1,2$$

$$0,2 \cdot 2 = 0,4$$

## 1.6 Représentation de l'information

### 1.6.1 Généralités

L'information en langage binaire est structurée en plusieurs catégories selon l'usage et leur format différent pour permettre leur bonne lecture par la machine.



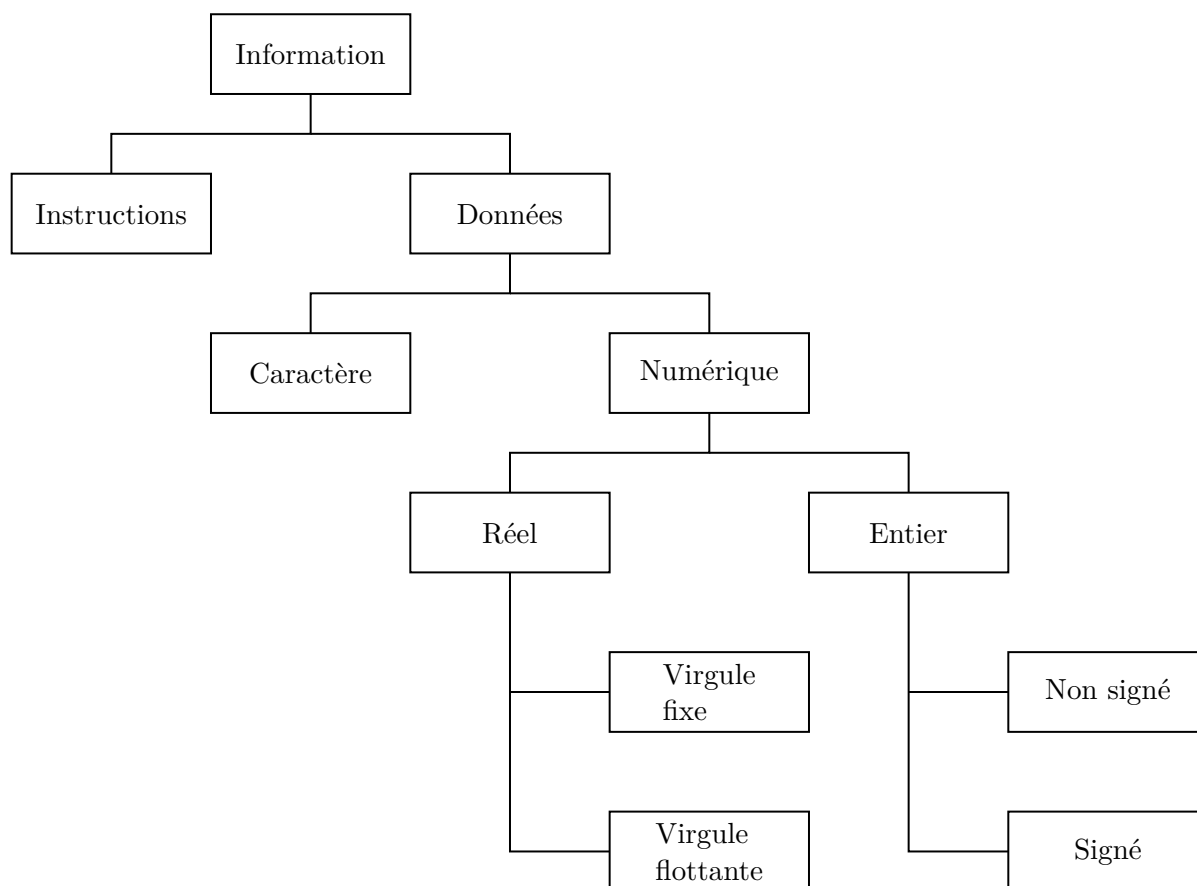


FIG. 1.1 – Organigramme de l'information numérique

### 1.6.2 Nombre entier

Il existe deux types d'entiers :

- les entiers *non signés* strictement positifs ;
- les entiers *signés* positifs ou négatifs.

Il existe trois méthodes pour indiquer à une machine que l'entier est positif ou négatif en binaire :

- signe/ valeur absolue ;
- complément à 1 / restreint ;
- complément à 2 / à vrai.

### 1.6.3 Nombre réel

Il existe deux types de réels :

- réels à virgules fixes ;
- réels à virgules flottantes.

Les réels à virgules flottantes sont caractérisés par une écriture mathématique incluant une *mantisse*  $M$ .

$$N = \pm M \cdot b^e \quad 13,11 = 0,1311 \cdot 10^2$$

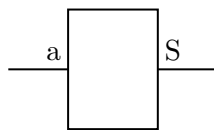
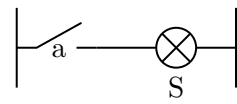
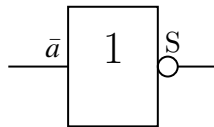
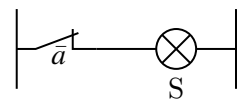
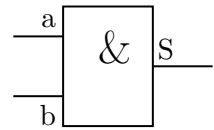
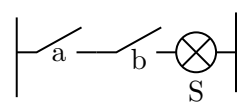
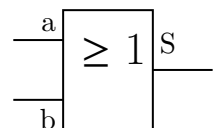
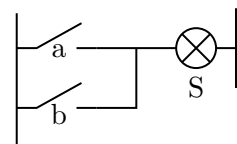
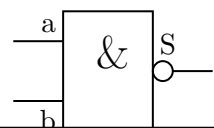
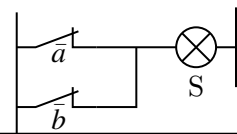
En virgule flottante, on attribue un bit pour le signe (0=+ et 1=-),  $p$  bit pour l'expression de l'exposant (complément à deux ou exposant biaisé) et 8 bit pour l'expression du mantisse.



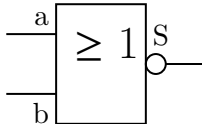
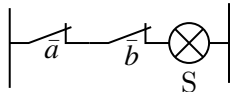
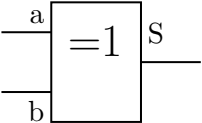
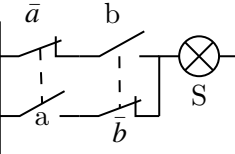
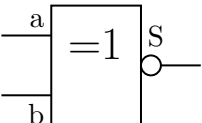
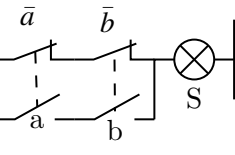
### 1.6.4 Portes logiques

Les portes logiques permettent de créer des circuits logiques à partir de fonctions complexes. Elles sont souvent utilisées en programmation pour caractériser les fonctions booléennes.



Équation logique	Table de vérité	Identité remarquable	Propriété	Symbole	Schéma															
Fonction OUI																				
$S = a$	<table><tr><td>a</td><td>S</td></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	a	S	0	0	1	1													
a	S																			
0	0																			
1	1																			
Fonction NON																				
$S = \bar{a}$	<table><tr><td>a</td><td>S</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	a	S	0	1	1	0													
a	S																			
0	1																			
1	0																			
Fonction ET																				
$S = a \cdot b$	<table><tr><td>a</td><td>b</td><td>S</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	a	b	S	0	0	0	0	1	0	1	0	0	1	1	1	<ul style="list-style-type: none"><li>- commutative ;</li><li>- associative ;</li><li>- distributive.</li></ul>	<p><b>élément neutre :</b> <math>a \cdot 1 = a</math></p> <p><b>élément absorbant :</b> <math>a \cdot 0 = 0</math></p> <p><b>idempotence :</b> <math>a \cdot a = a</math></p> <p><b>complément :</b> <math>a \cdot \bar{a} = 0</math></p>		
a	b	S																		
0	0	0																		
0	1	0																		
1	0	0																		
1	1	1																		
Fonction OU																				
$S = a + b$	<table><tr><td>a</td><td>b</td><td>S</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	a	b	S	0	0	0	0	1	1	1	0	1	1	1	1	<ul style="list-style-type: none"><li>- commutative ;</li><li>- associative ;</li><li>- distributive.</li></ul>	<p><b>élément neutre :</b> <math>a + 1 = a</math></p> <p><b>élément absorbant :</b> <math>a + 1 = 1</math></p> <p><b>idempotence :</b> <math>a + a = a</math></p> <p><b>complément :</b> <math>a + \bar{a} = 1</math></p>		
a	b	S																		
0	0	0																		
0	1	1																		
1	0	1																		
1	1	1																		
Fonction NAND (NON-ET)																				
$S = \overline{a \cdot b}$	<table><tr><td>a</td><td>b</td><td>S</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table>	a	b	S	0	0	1	<ul style="list-style-type: none"><li>- commutative.</li></ul>	<ul style="list-style-type: none"><li>- <math>\overline{a \cdot 1} = \bar{a}</math></li><li>- <math>\overline{a \cdot 0} = 1</math></li><li>- <math>\overline{a \cdot a} = \bar{a}</math></li><li>- <math>\overline{a \cdot \bar{a}} = 0</math></li></ul>											
a	b	S																		
0	0	1																		

Page précédente

Équation logique	Table de vérité	Identité remarquable	Propriété	Symbole	Schéma												
	<table><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	0	1	1	1	0	1	1	1	0							
0	1	1															
1	0	1															
1	1	0															
Fonction NOR (NON-OU)																	
$S = \overline{a/b}$	<table><tr><td>a</td><td>b</td><td>S</td></tr></table>	a	b	S	- commutative.	<ul style="list-style-type: none"><li>- <math>\overline{a+1} = 0</math></li><li>- <math>\overline{a+0} = \bar{a}</math></li><li>- <math>\overline{a+a} = \bar{a}</math></li><li>- <math>\overline{a+\bar{a}} = 0</math></li></ul>											
a	b	S															
$S = \bar{a} \cdot \bar{b}$	<table><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	0	0	1	0	1	0	1	0	0	1	1	0				
0	0	1															
0	1	0															
1	0	0															
1	1	0															
Fonction XOR																	
$S = a \oplus b$	<table><tr><td>a</td><td>b</td><td>S</td></tr></table>	a	b	S	<ul style="list-style-type: none"><li>- commutative ;</li><li>- associative</li></ul>	<ul style="list-style-type: none"><li>- <math>a \oplus 1 = \bar{a}</math></li><li>- <math>a \oplus 0 = a</math></li><li>- <math>a \oplus a = 0</math></li><li>- <math>a \oplus \bar{a} = 1</math></li></ul>											
a	b	S															
$S = \bar{a} \cdot b + a \cdot \bar{b}$	<table><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	0	0	1	0	1	1	1	0	1	1	1	0				
0	0	1															
0	1	1															
1	0	1															
1	1	0															
Fonction XNOR																	
$S = a \odot b$	<table><tr><td>a</td><td>b</td><td>S</td></tr></table>	a	b	S	- commutative.	<ul style="list-style-type: none"><li>- <math>a \odot 1 = a</math></li><li>- <math>a \odot 0 = \bar{a}</math></li><li>- <math>a \odot a = 1</math></li><li>- <math>a \odot \bar{a} = 0</math></li></ul>											
a	b	S															
$S = \overline{a \oplus b}$	<table><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	0	0	1	0	1	0	1	0	0	1	1	1				
0	0	1															
0	1	0															
1	0	0															
1	1	1															

Page suivante

*Page précédente*

Équation logique	Table de vérité	Identité remarquable	Propriété	Symbole	Schéma
------------------	-----------------	----------------------	-----------	---------	--------

TAB. 1.2 – Portes logiques





## 1.6.5 Nombres entiers

