CHAPITRE 1

Représentation des nombres

La nécessité de quantifier, notamment les échanges commerciaux, s'est faite dès la structuration de la vie sociale. Les tentatives de représentation symbolique de quantités furent nombreuses — bâtons, chiffres romains, etc. — avant que ne s'impose la numération arabe, universellement adoptée étant donné sa bonne capacité à traiter les calculs courants.

L'emploi quotidien de ce système nous fait oublier la structure et les règles qui régissent l'écriture des nombres, notamment la notion de base.

Après un bref rappel de ces règles, nous décrirons dans ce chapitre l'utilisation de bases autres que la base dix communément utilisée. Nous nous intéresserons notamment à la base deux qui représente le fondement de toute représentation numérique dans les systèmes électroniques.

1.1. PRINCIPE DE NUMÉRATION

La numération traditionnelle représente un nombre par la juxtaposition de **symboles,** appelés **chiffres,** pris parmi un ensemble. Par exemple, dans le système décimal, cet ensemble contient dix symboles différents.

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Le nombre de symboles définit la **base de numération.**

Lorsqu'un nombre est écrit, la position respective des chiffres détermine leurs **poids :** ... milliers, centaines, dizaines, unités, dizièmes, centièmes... Par exemple :

$$1\,515 = 1 \times \underbrace{1\,000}_{10^3} + 5 \times \underbrace{100}_{10^2} + 1 \times \underbrace{10}_{10^1} + 5 \times \underbrace{1}_{10^0}$$

$$3{,}14 = 3 \times \underbrace{1}_{10^0} + 1 \times \underbrace{\frac{1}{10}}_{10^{-1}} + 4 \times \underbrace{\frac{1}{100}}_{10^{-2}}$$

Ce type de représentation peut être généralisé en modifiant le nombre de symboles disponibles, c'est-à-dire en modifiant la base.

Soit une base b associée à b symboles $\{S_0, S_1, \ldots S_{b-1}\}$. Un nombre N s'écrit suivant la règle précédente :

$$N = (a_n\, a_{n-1} \ldots a_i \ldots a_0, a_{-1} \ldots a_{-m})_b \text{ avec } a_i \in \{S_0, \ldots, S_{b-1}\}$$

Ce nombre a pour valeur :

$$N = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_i b^i + \ldots a_0 b^0 + a_{-1} b^{-1} + \ldots + a_{-m} b^{-m}$$

ou

$$\sum_{i=-m}^{n} a_i b^i$$

Cette forme est appelée **forme polynomiale.**

Dans la représentation $(a_n \ldots a_i \ldots a_0, a_{-1} \ldots a_{-m})_b$ l'élément a_i est le chiffre (digit) de rang i. Son poids est b^i ; a_n est le chiffre le plus significatif et a_{-m} le moins significatif.

La virgule sépare le nombre en deux parties : $a_n \ldots a_0$, appelée la partie entière, et $a_{-1} \ldots a_{-m}$, la partie fractionnaire,

Quatre bases sont fréquemment rencontrées :

— **La base 2 ou système de numération binaire.** C'est la base la plus utilisée en électronique. On dispose de deux symboles {0,1} appelés **bits** (contraction de *binary digit*). L'élément a_n est alors le bit le plus signficatif (**MSB***) et a_{-m} le moins significatif (**LSB***).

Le nombre $N = (a_n \ldots a_0, a_{-1} \ldots a_{-m})_2$ s'écrit, d'après la décomposition polynomiale :

$$N = a_n 2^n + \ldots + a_0 + a_{-1} 2^{-1} + \ldots a_{-m} 2^{-m}$$

Exemple :

$$(1010,1)_2 = (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) + (1 \times 2^{-1}) = (10,5)_{10}$$

— **La base 8 ou numération octale.** Cette base est plutôt utilisée par les informaticiens. Les symboles sont {0, 1 ... 7}. Un nombre en octal se décompose de la façon suivante :

$$N = (a_n \ldots a_0, a_{-1} \ldots a_{-m})_8$$
$$= (a_n 8^n) + \ldots + (a_0 8^0) + (a_{-1} 8^{-1}) + \ldots + (a_{-m} 8^{-m})$$

— **La base 16 ou numération hexadécimale.** La base 16 est apparue avec la logique microprogrammée et les microprocesseurs. L'ensemble des symboles contient 16 éléments. Comme il n'est pas possible traditionnellement d'écrire, avec un seul caractère, un chiffre dont la valeur est supérieure à 9, l'ensemble comporte des lettres.

* MSB : *most significant bit ;* LSB : *less significant bit.*

Par convention, A est équivalent à 10, B à 11 et ainsi de suite. L'ensemble des symboles de la base 16 est donc :

$$\{0, 1, \ldots, 8, 9, A, B, C, D, E, F\}$$

La base 16 est une forme contractée de la base 2 (*cf.* § 1.2.3).

— **La base 10 ou système de numération décimale.** C'est la base utilisée dans la vie courante.

Le tableau suivant donne pour quelques nombres leurs correspondances dans les bases 2, 8 et 16.

$(N)_{10}$	$(N)_2$	$(N)_8$	$(N)_{16}$
0	0 0 0 0 0	0 0	0 0
1	0 0 0 0 1	0 1	0 1
2	0 0 0 1 0	0 2	0 2
3	0 0 0 1 1	0 3	0 3
4	0 0 1 0 0	0 4	0 4
5	0 0 1 0 1	0 5	0 5
6	0 0 1 1 0	0 6	0 6
7	0 0 1 1 1	0 7	0 7
8	0 1 0 0 0	1 0	0 8
9	0 1 0 0 1	1 1	0 9
1 0	0 1 0 1 0	1 2	0 *A*
1 1	0 1 0 1 1	1 3	0 *B*
1 2	0 1 1 0 0	1 4	0 *C*
1 3	0 1 1 0 1	1 5	0 *D*
1 4	0 1 1 1 0	1 6	0 *E*
1 5	0 1 1 1 1	1 7	0 *F*
1 6	1 0 0 0 0	2 0	1 0
1 7	1 0 0 0 1	2 1	1 1

On peut remarquer que, plus la base d'un système est faible, plus il faut de chiffres pour représenter le même nombre.

1.2. CONVERSION D'UNE BASE DANS UNE AUTRE

Position du problème : comment exprimer le même nombre dans des bases différentes ?

$$N = (a_n \ldots, \ldots a_{-m})_{b1} = (c_j \ldots, \ldots c_{-k})_{b2}$$

En pratique trois cas se présentent :

1.2.1. CONVERSION D'UN NOMBRE ÉCRIT EN BASE *b* EN UN NOMBRE ÉCRIT EN BASE 10

C'est l'exploitation directe de la forme polynomiale

$$N = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b + a_0 + a_{-1} b^{-1} + \ldots + a_{-m} b^{-m}$$

soit, par exemple, $(1\,0\,1\,1\,0\,,\,1\,1)_2$ à convertir en base 10 :

$$(1\,0\,1\,1\,0\,,\,1\,1)_2 = 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 1\times 2 + 0\times 1 + 1\times 2^{-1} + 1\times 2^{-2} = (22{,}75)_{10}$$

1.2.2. CONVERSION D'UN NOMBRE ÉCRIT EN BASE 10 EN UN NOMBRE ÉCRIT EN BASE *b*

Le problème consiste alors à trouver la suite de symboles caractérisant le nombre N écrit en base b. Cependant, le nombre N étant donné à l'origine en base 10, tous les calculs seront également effectués en base 10. Le nombre N s'écrira

$$N = (a_n \ldots a_0, a_{-1} \ldots a_{-m})_b = (a_n \ldots a_0)_b + (0, a_{-1} \ldots a_{-m})_b$$

$(a_n \ldots a_0) = PE(N)$ est la partie entière de N exprimée en base b .
$(0, a_{-1} \ldots a_{-m}) = PF(N)$ est la partie fractionnaire de N exprimée en base b.

La partie entière peut s'écrire $(a_n b^{n-1} + \ldots + a_1)\, b + a_0$. La division de $PE(N)$ par b donne $PE'(N) = (a_{n-1} b^{n+1} + \ldots + a_1)$, avec pour reste a_0

Il est alors facile de reproduire ce calcul avec $PE'(N)$, le reste est alors a_1, et ainsi de suite par divisions successives jusqu'à a_n.

La partie fractionnaire peut s'écrire $b^{-1}(a_{-1} + \ldots + a_{-m} b^{-m+1})$. La multiplication de $PF(N)$ par b donne

$$b\,.\,PF(N) = a_{-1} + (a_{-2} b^{-1} + \ldots + a_{-m} b^{-m+1}),$$

soit également $b\,.\,PF(N) = a_{-1} + PF'(N)$.

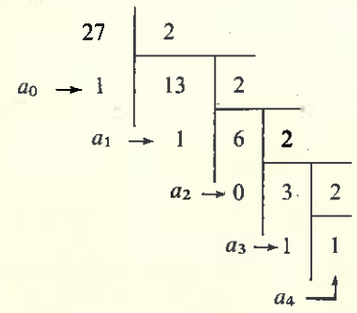
Par une suite de multiplications successives, on trouve facilement la suite $a_{-1} \ldots a_{-m}$.

Remarque : La représentation de b^{-1} en base 10 peut comporter une suite infinie de chiffres après la virgule, on limite $a_{-1} \ldots a_{-m}$ en respectant la précision avec laquelle on connaît N.

Si, par exemple, un nombre est exprimé avec trois chiffres après la virgule en base 10, il est connu avec une précision de 10^{-3}. Exprimé en base 2, la partie fractionnaire devra comporter au plus 10 chiffres ; en effet, $2^{-10} = 1/1024 \approx 10^{-3}$.

Exemple : soit $(27{,}3)_{10}$ à convertir en base 2 :

Partie entière :



$PE(N) = (1\,1\,0\,1\,1)_2$

Partie fractionnaire :

$$0{,}3 \times 2 = 0{,}6 \to 0{,}6 \times 2 = 1{,}2 \to 0{,}2 \times 2 = 0{,}4 \to 0{,}4 \times 2 = 0{,}8$$
$$\downarrow 0 \qquad \downarrow 1 \qquad \downarrow 0 \qquad \downarrow 0$$

$PF(N) = (0\,,\,0\,1\,0\,0)_2$,
d'où $(27{,}3)_{10} = (1\,1\,0\,1\,1\,,\,0\,1\,0\,0)_2$.

1.2.3. CONVERSION D'UN NOMBRE ÉCRIT EN BASE *b* EN UN NOMBRE ÉCRIT EN BASE b^k ET INVERSEMENT

A partir du développement polynomial de N dans la base b, il est possible de regrouper les termes k par k à partir de la virgule et d'effectuer la mise en facteur qui s'impose.

$$N = a_n b^n + \ldots + a_5 b^5 + a_4 b^4 + a_3 b^3 + a_2 b^2 + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + a_{-3} b^{-3} + \ldots + a_{-m} b^{-m}$$

Si par exemple $k = 3$, N s'écrit :

$$N = \ldots + (a_5 b^2 + a_4 b + a_3)\, b^3 + (a_2 b^2 + a_1 b + a_0)\, b^0 + (a_{-1} b^2 + a_{-2} b + a_{-3})\, b^{-3} + \ldots$$

Le contenu de chaque parenthèse est forcément inférieur à b^k. Ce nombre peut donc constituer un chiffre de la base b^k, et alors être représenté par le symbole correspondant dans cette base.

Exemple : 1) Passage de la base 2 à la base 8 = 2^3 :

$$N = (\underbrace{0\,1\,0}\ \underbrace{1\,1\,1}\ \underbrace{1\,1\,0}\ \underbrace{1\,0\,1}\ ,\ \underbrace{0\,1\,1}\ \underbrace{0\,1\,0})_2$$

$$N = (2\quad 7\quad 6\quad 5\ ,\ 3\quad 2)_8$$

2°) Ecriture du même nombre en base 16 = 2^4 :

$$N = (\underbrace{0\,1\,0\,1}\ \underbrace{1\,1\,1\,1}\ \underbrace{0\,1\,0\,1}\ ,\ \underbrace{0\,1\,1\,0}\ \underbrace{1\,0\,0\,0})_2$$

$$N = (5\quad F\quad 5\ ,\ 6\quad 8)_{16}$$

$$F \equiv (15)_{10}$$

Le même principe, mais utilisé en sens inverse, permet les conversions dans l'autre sens.

Les conversions de b^k à b^p sont obtenues en passant tout d'abord par la base b.

1.3. REPRÉSENTATION DES NOMBRES

1.3.1. REPRÉSENTATION BINAIRE DES NOMBRES SIGNÉS

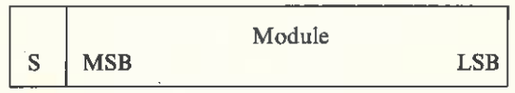
Dans les conversions vues précédemment, il n'est pas fait allusion au signe. Seule la valeur absolue des nombres peut être représentée par un ensemble d'éléments binaires. Plusieurs modes de représentations de nombres signés sont effectivement adoptés en fonction des calculs à effectuer et des caractéristiques technologiques des systèmes utilisés.

1.3.1.1. Représentation Module plus signe

C'est la solution la plus simple. Un élément binaire est ajouté au module pour la représentation du signe. Habituellement, il est utilisé la correspondance suivante

$$0 \Leftrightarrow +$$
$$1 \Leftrightarrow -$$

et le signe est placé à gauche du module :



Exemple $(-23)_{10}$:

$$\overset{1}{\Updownarrow}\quad \underbrace{0\,0\,1\,0\,1\,1\,1}_{23}$$

En conséquence, si le **format** (nombre de bits utilisés) d'un nombre N est de $n+1$ éléments binaires, alors ce nombre est compris entre $-(2^n-1)$ et $+(2^n-1)$.

Exemple :

$n = 4 \quad \Leftrightarrow \quad -7 \leqslant N \leqslant +7$

Nombres positifs		Nombres négatifs	
7	0 1 1 1	−7	1 1 1 1
6	0 1 1 0	−6	1 1 1 0
.	. . . . .	. . .	. . . . .
1	0 0 0 1	−1	1 0 0 1
0	0 0 0 0	0	1 0 0 0
	↑ Signe Module		↑ Signe Module

Remarque : Il existe deux représentations différentes du nombre 0.

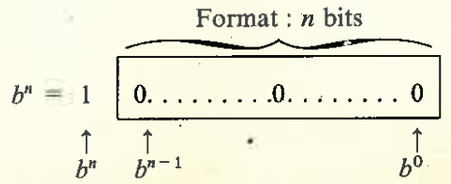
Ce mode de représentation présente l'avantage d'être simple, mais est peu adapté aux additions. Pour ce domaine d'application il est préférable d'utiliser le mode « complément vrai ».

1.3.1.2. Représentation des nombres en complément vrai

Par définition, le complément vrai $\tilde{N}$ du nombre N exprimé à l'aide de n chiffres dans la base b est :

$$\tilde{N} = b^n - N$$

b^n s'écrit avec un « 1 » suivi de n zéros.



Le « 1 » de poids b^n ne rentre pas dans le format défini. Par conséquent, le nombre b^n est interprété par le calculateur comme le nombre 0, d'où :

$$\tilde{N} = 0 - N = (-N)$$

Le complément vrai s'identifie donc à l'**opposé arithmétique du nombre *N*.**

— *Dans le cas du système binaire* il vient

$$(-N) = 2^n - N$$

ce qui s'écrit également :

$$(-N) = (2^n - 1) - N + 1$$

Le nombre $(2^n - 1)$ est le plus grand nombre que l'on puisse représenter avec le format de n bits.

La différence $(2^n - 1) - N$ se calcule alors facilement puisqu'il n'apparaît jamais de retenue.

Exemple :

$n = 4$ $\qquad N = 0011 \qquad 2^n - 1 = 1111$

$$\left.\begin{matrix} 2^n - 1 \\ -\ N \end{matrix}\right\} \Rightarrow \begin{array}{r} 1111 \\ -\ 0011 \\ \hline 1100 \end{array}$$

On remarque que chaque élément binaire de la différence est le complément à 1 de l'élément binaire correspondant de *N*.

Par définition, le **complément à 1 d'un élément binaire** A est un élément noté $\overline{A}$ tel que :

$$A + \overline{A} = 1 \text{ soit encore si } A = 0 \Leftrightarrow \overline{A} = 1$$
$$\text{si } A = 1 \Leftrightarrow \overline{A} = 0$$

En généralisant, le **complément à 1 d'un nombre binaire** N écrit à l'aide de n bits, $a_{n-1} \ldots a_i \ldots a_0$, est le nombre $\overline{N}$ qui s'écrit $\overline{a_{n-1}} \ldots \overline{a_i} \ldots \overline{a_0}$.

N et $\overline{N}$ sont liés par la relation : $N + \overline{N} = 2^n - 1$

$$\begin{array}{lll} N & \rightarrow & a_{n-1} \ldots a_i \ldots a_0 \\ \overline{N} & \rightarrow & +\overline{a_{n-1}} \ldots \overline{a_i} \ldots \overline{a_0} \\ \hline 2^n - 1 & & = 1 \ \ldots \ 1 \ \ldots \ 1 \end{array}$$

Le calcul de l'opposé de N devient alors :

$$(-N) = (2^n - 1) - N + 1 = \overline{N} + 1$$

$(-N)$ est appelé **complément à 2 de *N*.**

Exemple : Pour $n = 4$ on obtient :

$(N)_{10}$	=	N	⇔	Complément à 1 $\overline{N}$	⇔	Complément à 2 $(-N)$	=	opposé de N
7		0111		1000		1001		−7
6		0110		1001		1010		−6
5		0101		1010		1011		−5
4		0100		1011		1100		−4
3		0011		1100		1101		−3
2		0010		1101		1110		−2
1		0001		1110		1111		−1
0		0000		1111		0000		0

On remarque que :

— L'élément binaire situé à l'extrême gauche est encore représentatif du signe avec la convention

$$0 \Leftrightarrow +$$
$$1 \Leftrightarrow -$$

— Le nombre 0 n'a qu'une représentation.

— La combinaison 1000 n'est jamais rencontrée. Cette combinaison résulte de la suppression de la double représentation du zéro. Comme elle est représentative d'un nombre négatif, la valeur décimale correspondante est −8.

En effet l'addition de ce nombre avec 1 donne −7 :

$$\begin{array}{rr} 1000 & -\ 8 \\ +\ 0001 & +\ 1 \\ \hline 1001 & -\ 7 \end{array}$$

L'échelle représentative des nombres exprimés en complément à 2 (complément vrai base 2) sur n digits est :

$$-2^{n-1} \leqslant N \leqslant +2^{n-1} - 1$$

— Enfin, on vérifie que l'opposé d'un nombre négatif s'obtient avec la même règle, à l'exception du nombre -2^{n-1} qui n'a pas d'opposé.

$$-(-N) = \overline{(-N)} + 1$$

Ce mode de représentation des nombres signés est le plus utilisé, surtout en ce qui concerne l'arithmétique binaire.

— *Dans le cas général* (base b), l'opposé se calcule avec la relation

$$(-N) = (b^n - 1) - N + 1 = \overline{N} + 1$$

De la même façon que dans le système binaire, la quantité $(2^n - 1) - N$ est noté $\overline{N}$ et s'exprime par

$$\overline{N} = \overline{a_{n-i}} \ldots \overline{a_i} \ldots \overline{a_0}$$

$\overline{N}$ est le **complément restreint** et les éléments de $\overline{N}$ sont les compléments à $b-1$:

$$a_i + \overline{a_i} = b - 1$$

1.3.2. REPRÉSENTATION DES NOMBRES DANS UN CALCULATEUR

Les nombres dans un calculateur sont représentés par des éléments binaires suivant un format déterminé. On parle de nombres écrits en **virgule fixe** ou en **virgule flottante.**

Dans chaque représentation sont examinées les performances, c'est-à-dire :

— la **résolution :** différence entre deux nombres consécutifs,

— la **dynamique :** différence entre le nombre le plus petit et le plus grand.

1.3.2.1. La représentation en virgule fixe

La représentation d'un nombre dans un calculateur se fait sous forme d'un bloc de n éléments binaires *considéré comme un entier N*. Pour l'utilisateur, cet entier peut être représentatif d'un nombre fractionnaire s'il connaît la place de la virgule. Celle-ci doit donc avoir une place déterminée et fixe, d'où le nom de cette représentation. Par exemple, à la mesure 1,250 m on peut associer le nombre entier 125 si l'utilisateur traduit cette longueur en centimètres ou 1 250 si l'unité est le millimètre.

Pour un nombre entier représenté en virgule fixe, la résolution est de 1 et la dynamique est égale à 2^{n-1}, n étant le nombre d'éléments binaires. La représentation de cet entier est faite en binaire naturel ou suivant l'une des représentations des nombres signés.

Pour l'utilisateur ayant placé la virgule au rang k, la résolution devient 2^{-k} et la dynamique est alors réduite à 2^{n-k}.

$N =$ | 2^{n-1} ... 2^0 | , virgule au rang 0 — $0 \leqslant N \leqslant 2^n - 1$

MSB LSB

$N_k =$ | 2^{n-1-k} ... 2^0 2^{-1} 2^{-k} | , virgule au rang k — $0 \leqslant N_k \leqslant (2^n - 1)\,2^{-k}$

MSB LSB

Ce mode de représentation se prête particulièrement bien aux opérations d'addition puisque la virgule ne change pas de place au cours de l'opération :

```
  17,84
+ 16,18
--------
  34,02
```

Par contre dans le cas de la multiplication il y a un décalage de la virgule dans le résultat correspondant au nombre total de chiffre après la virgule dans le multiplicateur et le multiplicande.

```
   13,4      1 chiffre après la virgule
 × 10,1    + 1 chiffre après la virgule
--------
    134
   134. .
--------
  135,34     = 2 chiffres après la virgule
```

D'autre part le nombre de chiffres du résultat est inférieur ou égal à la somme des chiffres des deux opérandes, ce qui peut provoquer des débordements de capacités.

```
   77      2 chiffres
 × 34    × 2 chiffres
-------
   308
  231.
-------
  2618     = 4 chiffres
```

Lorsque la virgule est placée à l'extrême gauche du nombre, la position de la virgule reste fixe. Le nombre de chiffres à conserver reste le seul problème : la troncature du résultat provoque un arrondi qui ne modifie pas la précision relative du calcul.

,87
,74

348
609.

,6438 ⇒ 0,64 pour un format à 2 chiffres après la virgule

Afin d'éliminer les problèmes de gestion de la virgule, notamment dans les multiplications, une solution consiste à placer la virgule systématiquement à gauche du nombre et à utiliser un autre ensemble d'éléments binaires permettant de connaître la position exacte de la virgule ; c'est le mode de représentation en virgule flottante.

1.3.2.2. La représentation en virgule flottante

Dans ce mode de représentation, semblable à la notation scientifique, les nombres sont écrits sous la forme

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

M est appelé la **mantisse** et E l'**exposant,** b étant la base du système.

Exemple : En base 10 le nombre π peut s'écrire à l'aide de 8 chiffres pour la mantisse

	,31415926	$\times 10^1$
ou	,00314159	$\times 10^3$
ou	,00000314	$\times 10^6$
	mantisse	exposant

Dans l'exemple précédent, il est évident que la précision diminue au fur et à mesure que l'exposant augmente. La position qui donne le maximum de précision est celle où le chiffre le plus significatif (non nul) est placé à l'extrême gauche de la mantisse. Dans ce cas, la représentation est dite **normalisée**. En base 2, le principe reste le même.

Le mode de représentation en virgule flottante est celui qui autorise la plus grande dynamique. Pour un format de n élément binaires pour la mantisse et p éléments binaires pour l'exposant (signe non compris), le nombre le plus grand est obtenu lorsque la mantisse est maximale et l'exposant le plus élevé, soit :

$$N_{MAX} = \underbrace{(+, 11 \ldots 1)_2}_{\approx 1} \cdot 2^{(+111 \ldots 1)_2} \approx (2^{(2^p-1)})_{10}$$

et le nombre positif non nul le plus petit en représentation normalisée est :

$$N_{min} = \underbrace{(+, 100 \ldots 0)_2}_{2^{-1}} \cdot 2^{(-1111 \ldots 1)_2} = (2^{-2^p})_{10}$$

Les valeurs décimales correspondantes sont plus éloquentes que la représentation en puissance de 2. Par exemple, si $p = 7$, le nombre le plus grand est l'ordre de $1{,}7 \cdot 10^{38}$ alors que le plus petit non nul vaut 3×10^{-39} environ.

La résolution est fonction de l'exposant. Elle est égale à $2^{-n} \times 2^E$.

En virgule flottante, le nombre réel N est représenté par le nombre rationnel N_0 le plus proche. L'erreur d'arrondi est au maximum égal à la résolution précédemment calculée.

$$N_0 \leqslant N < N_0 + \Delta N \qquad \text{avec} \qquad \Delta N = 2^{-n} \times 2^E$$

La quantité $\frac{\Delta N}{N_0}$ fixe la précision relative de la représentation du nombre N si

$$N_0 = 0{,}100 \ldots 0 \times 2^E \qquad \text{alors } \frac{\Delta N}{N_0} = 2^{-n+1}$$

ou si

$$N_0 = 0{,}11 \ \ldots 1 \times 2^E \qquad \text{alors } \frac{\Delta N}{N_0} = 2^{-n}$$

Pour $n = 24$, la précision relative étant comprise entre $1{,}2 \cdot 10^{-7}$ et 6×10^{-8}, les nombres décimaux correspondants sont donc exprimés avec sept chiffres significatifs.

Afin de faciliter la comparaison de deux nombres exprimés en virgule flottante, les deux parties mantisse et exposant sont regroupées dans l'ordre signe, exposant, mantisse, les éléments ayant le plus fort poids étant placés à gauche du nombre.

S	Exp	Mantisse

Si l'exposant est exprimé en complément à 2 avec complémentation du bit de signe, il apparaît comme un nombre croissant de 0 à $2^{p+1} - 1$.

Complément à 2		Exposant	
+ 15	0 1 1 1 1	31	1 1 1 1 1
	0 1 1 1 0		1 1 1 1 0
1	0 0 0 0 1		1 0 0 0 1
0	0 0 0 0 0	16	1 0 0 0 0
− 1	1 1 1 1 1		0 1 1 1 1
− 2	1 1 1 1 0		0 1 1 1 0
− 16	1 0 0 0 0	0	0 0 0 0 0

La mantisse peut être exprimée en binaire (module) ou éventuellement en complément à 2.

1.3.2.3. La représentation des nombres sous forme de « chaîne »

Dans certaines machines, en particulier les calculatrices, on peut conserver la représentation habituelle des nombres, chaque chiffre étant traduit en binaire. L'ensemble forme une chaîne. Les signes et la virgule sont traduits par des caractères spéciaux.

Exemple : Le nombre π se représente par

Début	+	3	,	1	4	1	5	9	Fin

Il est possible d'utiliser également la notation scientifique, ce que donne :

Début	+	3	1	4	,	1	5	9	E	−	2	Fin

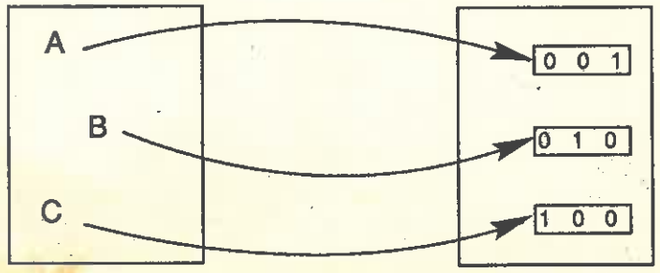
Des caractères de début et de fin sont indispensables lorsque les chaînes formées sont de longueur variable. Une combinaison de caractères spéciaux peut se substituer à un caractère FIN, par exemple deux espaces successifs.

Pour réaliser ce mode de représentation, il est indispensable d'écrire chaque chiffre et chaque caractère sous une forme particulière. Cette opération s'appelle le « codage ».

1.4. CODAGE

1.4.1. GÉNÉRALITÉS

Le codage est une opération qui établit une correspondance entre les éléments de deux ensembles



En termes plus mathématiques, le codage peut être défini comme une application bijective d'un ensemble source vers un autre ensemble appelé ensemble but.

L'application inverse est appelée le décodage.

L'ensemble source est en général un sous-ensemble des nombres rationnels et l'ensemble but contient des combinaisons ordonnées de 0 et de 1. A un nombre donné correspond une combinaison binaire appelée mot-code. L'ensemble des mots-code forme le **code**.

Le choix d'un code dépend évidemment de l'application envisagée : dans le cas d'une transmission de données, par exemple, il sera préférable de choisir des **codes détecteurs,** ou mieux **correcteurs d'erreurs,** le message risquant d'être dégradé au cours de la transmission.

D'autre part des tentatives de normalisation de codage des caractères alpha-numériques dans les calculateurs ont donné naissance à des codes très utilisés tel l'ASCII (American Standard Code for Information Interchange).

1.4.2. LES CODES DÉCIMAUX

Les codes décimaux sont utilisés pour la représentation des chiffres du système décimal. Ils contiennent par conséquent dix mots-code. Parmi les plus fréquemment rencontrés, citons les **codes BCD, Excess 3, Aïken**.

$(N)_{10}$	BCD	Excess 3	Aïken
0	0 0 0 0	0 0 1 1	0 0 0 0
1	0 0 0 1	0 1 0 0	0 0 0 1
2	0 0 1 0	0 1 0 1	0 0 1 0
3	0 0 1 1	0 1 1 0	0 0 1 1
4	0 1 0 0	0 1 1 1	0 1 0 0
5	0 1 0 1	1 0 0 0	1 0 1 1
6	0 1 1 0	1 0 0 1	1 1 0 0
7	0 1 1 1	1 0 1 0	1 1 0 1
8	1 0 0 0	1 0 1 1	1 1 1 0
9	1 0 0 1	1 1 0 0	1 1 1 1

Le code BCD (*Binary Coded Decimal*) contient des mots-code qui sont la traduction en binaire de chacun des chiffres du système décimal. Chaque élément binaire d'un mot-code a un poids comme en binaire. Pour retrouver l'original à partir du mot-code, on peut procéder comme cela est indiqué précédemment (*cf.* 1.2.1). Ce code est donc un code décimal pondéré.

Le code Excess 3 (XS 3) a été créé pour permettre la réalisation simple des opérations de soustraction. Le complément à 1 d'un mot représente le complément à 9 dans l'ensemble source.

Exemple :

$$\overline{8} = 1 \Rightarrow \overline{1\,0\,1\,1} = 0\,1\,0\,0$$

Les codes possédant cette propriété sont appelés des codes auto-complémentaires.

Le code Aïken regroupe les deux propriétés. C'est un code décimal pondéré auto-complémentaire. Les poids des éléments binaires sont 2 4 2 1.

Pour représenter un nombre, c'est-à-dire une succession de chiffres, il est simple d'utiliser une chaîne d'éléments binaires groupés par quatre. Chaque groupe de quatre éléments binaires (ou tétrade) est représentatif d'un chiffre.

Exemple : En utilisant le code BCD le nombre 1789 s'écrit

0001	0111	1000	1001
1	7	8	9

1.4.3. CODE A DISTANCE UNITÉ

Si l'on examine les informations à la sortie d'un dispositif de comptage d'objets par exemple, on va trouver la succession des nombres binaires au fur et à mesure que les objets sont détectés. L'instant d'observation par rapport à l'évolution de chacune des informations binaires peut introduire des erreurs importantes.

Entre 7 (0 1 1 1) et 8 (1 0 0 0) tous les éléments binaires changent, mais cette commutation n'est pas forcément instantanée ; on peut avoir par exemple la succession d'état

$$(0\,1\,1\,1) \rightarrow (0\,1\,1\,0) \rightarrow (0\,1\,0\,0) \rightarrow (0\,0\,0\,0) \rightarrow (1\,0\,0\,0)$$

Suivant l'instant d'observation, les résultats sont différents. Pour éviter ce phénomène il suffit de coder chaque nombre de façon que deux nombres successifs diffèrent d'un élément binaire et d'un seul. Le code répondant à cette caractéristique est un code à distance unité.

On appelle **distance** entre deux mots-code, le nombre d'éléments binaires qui diffèrent.

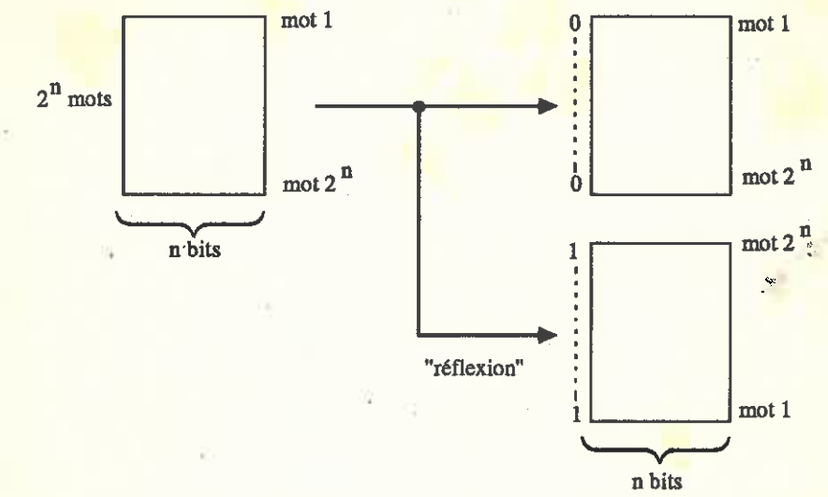
Le code à distance unité le plus fréquemment utilisé est le **code de Gray,** ou code réfléchi. Ce nom provient des propriétés d'optique utilisé pour sa construction. Le principe en est le suivant.

Si l'on dispose d'un ensemble de 2^n mots réalisant un code Gray, il est facile d'obtenir un ensemble de 2^{n+1} mots respectant le même code.

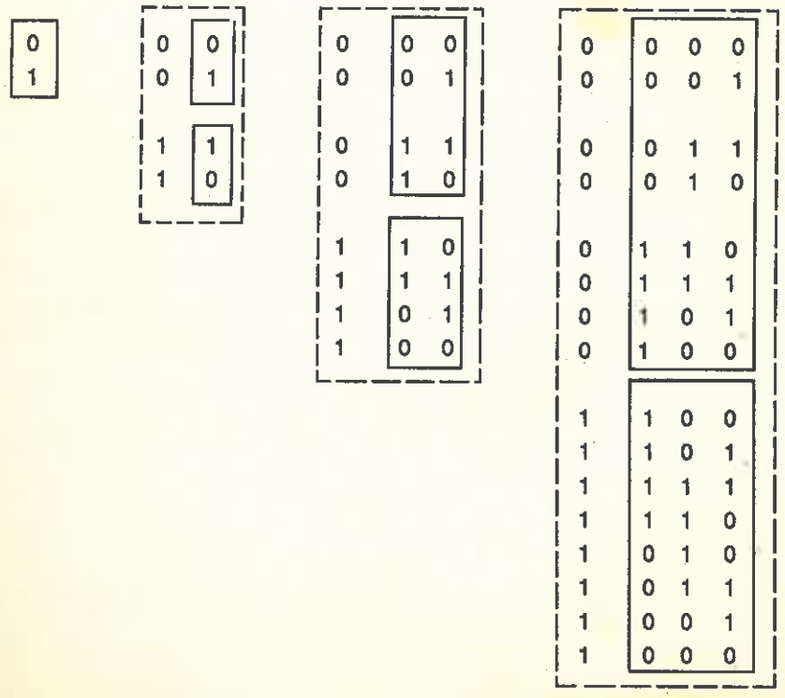
Ce nouvel ensemble est constitué :

— d'une part des mots-code au code initial, précédés de « 0 » ;

— d'autre part des mots-code du même code initial, présentés dans *l'ordre inverse*, précédés de « 1 ».

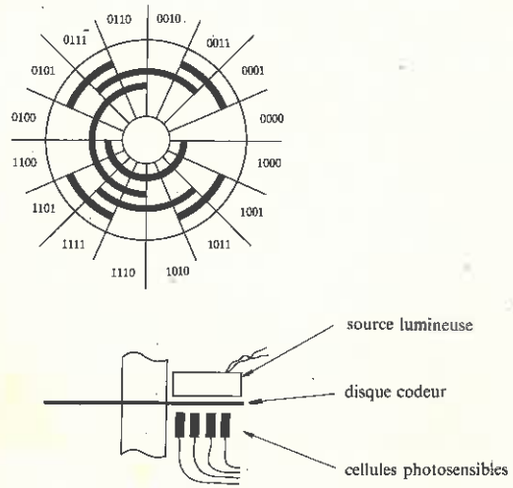


La construction est alors possible par étapes successives à partir de l'ensemble des deux mots 0 et 1.



On remarque qu'à la fin de chaque étape de la construction le premier et le dernier mot-code ont encore une distance unité.

Ces codes sont utilisés, par exemple, dans les asservissements, où la position angulaire d'un axe peut être codée par un dispositif optique composé d'un disque sur lequel on a gravé un motif tel que des capteurs photosensibles reproduisent directement la combinaison désirée :



EXERCICES

1. Convertir les nombres décimaux suivants en base 2 :
   - **a.** 12
   - **b.** 24
   - **c.** 192
   - **d.** 2 079
   - **e.** 15 492
   - **f.** 0,25
   - **g.** 0,375
   - **h.** 0,376
   - **i.** 17,150
   - **j.** 192,1875
2. Convertir les nombres décimaux de l'exercice 1 en hexadécimal.
3. Convertir les nombres binaires suivants en base 10 :
   - **a.** 1 0 1 1
   - **b.** 1 1 0 1 1
   - **c.** 1 0 1 , 1
   - **d.** 0 , 1 1 0 1
   - **e.** 0 , 0 0 1
   - **f.** 1 1 0 , 0 1
   - **g.** 1 0 1 1 0 1 1
   - **h.** 1 0 1 0 1 1 0 1 0 1 1
   - **i.** 1 0 0 1 0 0 1 1 0 1
   - **j.** 1 0 1 1 0 0 1 0 0 1 . 1 0 1 1 0
4. Convertir les nombres hexadécimaux suivants en base 10 :
   - **a.** FF
   - **b.** 1A
   - **c.** 789
   - **d.** 0,13
   - **e.** ABCD,EF
5. Ecrire les nombres décimaux suivants en binaire dans la représentation Module plus signe avec un format de huit éléments binaires :
   - **a.** +48
   - **b.** −48
   - **c.** +17
   - **d.** −24
   - **e.** −15
6. Ecrire les nombres décimaux de l'exercice 5 en binaire sous la représentation en complément à 2 avec un format de huit éléments binaires
7. Coder les nombres décimaux suivants en utilisant le code BCD :
   - **a.** 12
   - **b.** 192
   - **c.** 2079
   - **d.** 15463
   - **e.** 0,375
   - **f.** 17,250
8. Coder les nombres décimaux de l'exercice 7 (Excess 3).
9. Ecrire les nombres décimaux suivants dans le mode de représentation Module plus signe en utilisant le code BCD :
   - **a.** +048
   - **b.** −048
   - **c.** −157
   - **d.** +103
   - **e.** −124
10. Ecrire les nombres décimaux de l'exercice 9 dans le mode de représentation « complément vrai » en utilisant le code BCD et un format de 16 éléments binaires.
11. Même exercice que 9 mais en utilisant le code XS 3.
12. Même exercice que 10 mais en utilisant le code XS 3.
13. Donner la représentation des nombres décimaux suivants écrite en virgule flottante normalisée.
    - **a.** 157,37
    - **b.** 0,00015