LA NUMERATION POSITIONNELLE

ET LE CHANGEMENT DE BASE

Économe en symboles et efficace dans les opérations, le système positionnel s'est finalement imposé à toutes les civilisations modernes. Mais quelle base employer ? Bien que nous utilisions la base 10, en bijection avec nos dix doigts, d'autres bases sont légitimes. Comment passe-t-on d'une base à l'autre ? Pour y répondre, comprenons le fonctionnement du système de numération positionnel et ses propriétés, partons à l'autre bout du monde pour découvrir des bases "exotiques" et examinons le rôle que jouent toutes ces bases dans l'informatique.

Toutes les bases sont dans la nature

Compter implique d'associer une séquence de noms (de symboles) ou (de cailloux, d'entailles) à une quantité. On ne peut pas retenir une infinité de symboles ni prévoir une infinité de cailloux. Il a donc fallu regrouper les quantités par paquets. La taille d'un paquet constitue la base.

L'invention du système de numération positionnel n'a rien changé à cela : la base dans un système de numération positionnel correspond au nombre de symboles ou chiffres utilisés pour coder le nombre dans ce système.

Bien que le système décimal soit le plus répandu dans l'histoire et adopté universellement de nos jours, l'homme emploie encore ou a employé d'autres systèmes de numération positionnel.

Nous avons déjà rencontré les systèmes vigésimaux en base 20 des Mayas et sexagésimaux en base 60 des Babyloniens.

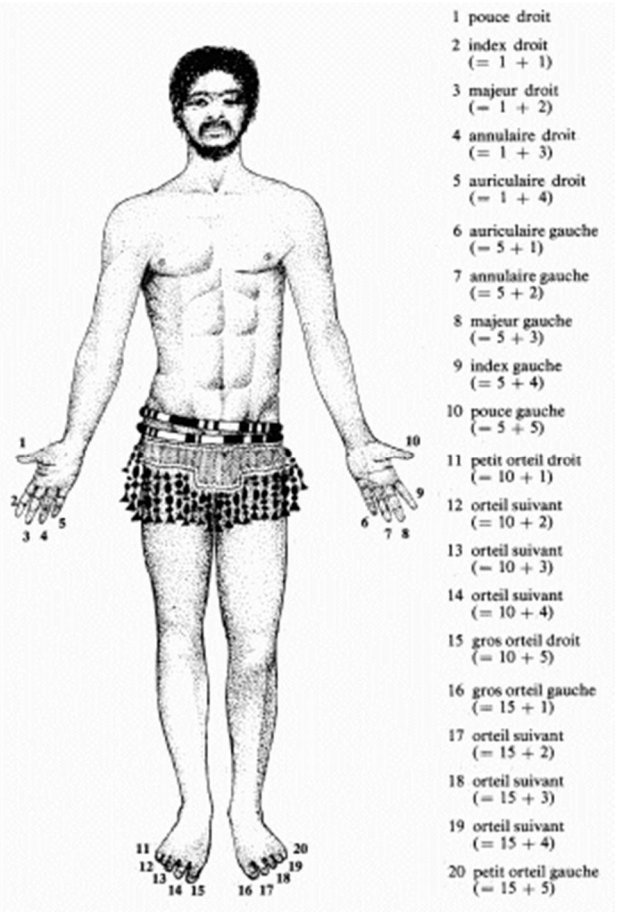


Figure II.1 : *La base vingt fut employée par certaines cultures car l'homme possède vingt doigts et orteils sur lesquels il peut compter.*

D’autres peuples ont choisi de regrouper les objets par 5, sans doute pour des raisons anthropomorphiques : une seule main pour compter.
Le système duodécimal en base 12 s’est également bien répandu et laisse des traces aujourd’hui dans notre vocabulaire (une *douzaine* d’œufs !). Les Anglais ont d’ailleurs établi leur système métrique sur cette base : un pied, valant 12 pouces !

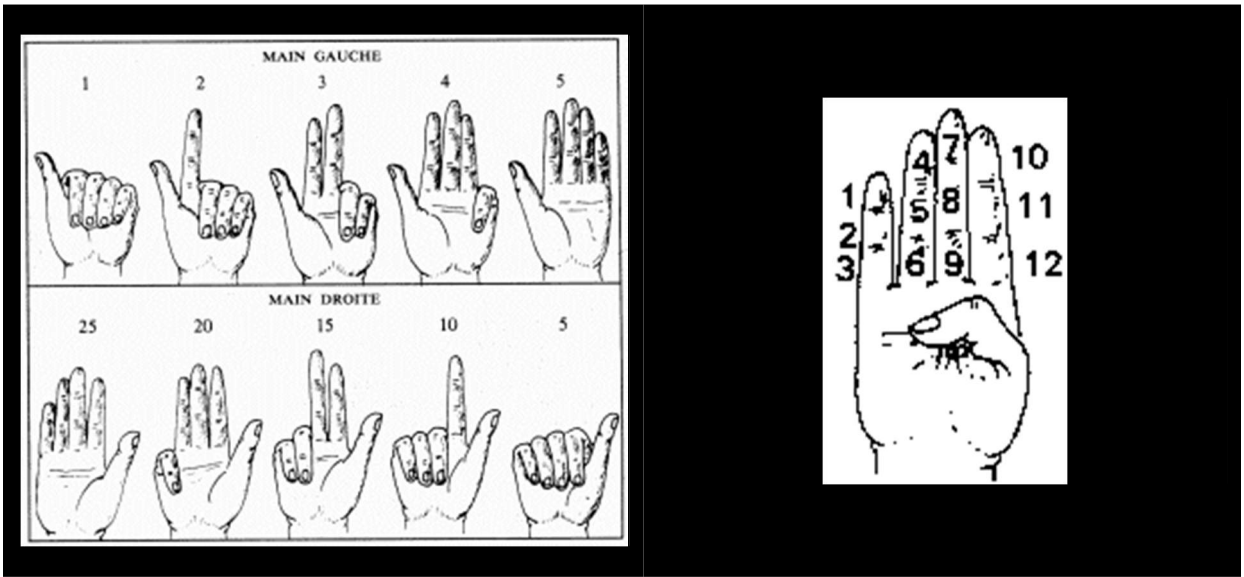


Figure II.2 et II.3 : *Voici deux techniques digitales pour compter en base 5 (à gauche) et en base 12 (à droite) où, en s'opposant aux autres doigts, le pouce permet de compter 12 phalanges.*

Aujourd’hui, l’homme utilise encore d’autres bases. Ainsi, la base 2, représentative de la logique et de l’algèbre de Boole, est utilisée par l’ordinateur. Les systèmes octal (base 8) et hexadécimal (base 16), issus du binaire, permettent de représenter des valeurs binaires de façon compacte avec une conversion facile comme nous le montrerons en page 71.

I. DEFINITION

De façon générale, un nombre N exprimé dans un système positionnel de base *b* est codé à l'aide de *b* symboles différents nommés chiffres.

Le codage d'un nombre naturel en base *b* repose sur le théorème suivant.

Considérons $b \in \mathbb{N}_{0,1}$, une base de numération.

Pour tout naturel *N* non nul, **il existe une seule et unique** décomposition de *N* sous la forme

$$N = a_n b^n + a_{n-1} b^{n-1} + ... + a_1 b^1 + a_0 \quad (1)$$

Où $0 \leq a_i < b \quad \forall i \in \{0,...,n\}$

Où $n \in \mathbb{N}$, est tel que $a_n \neq 0$.

On a donc effectué la décomposition de l'entier *N* en base *b* et on note $N = \overline{a_n a_{n-1} ... a_1 a_0}^{\,b}$.

L'expression (1) porte le nom de *représentation polynomiale.*

Si un système de numération est à base *b*, alors il y aura un nombre *b* de chiffres compris entre 0 et *b*-1.

DEMONSTRATION DE L'EXISTENCE

Cette démonstration repose sur le théorème de la division euclidienne que nous vous rappelons ici et constitue un algorithme d'obtention de l'écriture d'un nombre en base *b*.

Soit $b \in \mathbb{Z}_0$, $n \in \mathbb{Z}$; $\exists!(q,r) \in \mathbb{Z}^2$ tel que $n = q.b + r$ où $0 \leq r < |b|$

Pour démontrer notre théorème, nous appliquons plusieurs fois l'algorithme de division d'Euclide.

Si $N < b$, on peut choisir directement $a_0=N$.

Si $N \geq b$, divisons N par b.

$\exists!(q_0,r_0)\in \mathbb{Z}^2$ tel que $N = q_0.b + r_0$ où $0 \leq r_0 < |b|$

Si $q_0<b$, alors on pose $a_1 = q_0$ et $a_0 = r_0$.

Nous avons dans ce cas $N = \overline{a_1 a_0}^{b}$.

Si $q_0 \geq b$, on effectue la division euclidienne de q_0 par b.

$\exists!(q_1,r_1)\in \mathbb{Z}^2$ tel que $q_0 = q_1.b + r_1$ où $0 \leq r_1 < |b|$

Si $q_1<b$, alors on a $N = (q_1.b + r_1).b + r_0 = q_1.b^2 + r_1.b + r_0$

On pose alors $a_2 = q_1$, $a_1 = r_1$ et $a_0 = r_0$.

Nous avons dans ce cas $N = \overline{a_2 a_1 a_0}^{b}$.

Si $q_1 \geq b$, on recommence …

On obtient alors une suite $(q_0,q_1,q_2,...)$ décroissante et finie car $b>1$.

Il existe donc un entier p tel que $q_p<b$ et $q_{p-1} \geq b$.

On écrit la suite des divisions obtenues :

$$N = b.q_0 + r_0$$
$$q_0 = b.q_1 + r_1$$
$$\ldots$$
$$q_{p-1} = b.q_p + r_p$$

On multiplie alors chaque égalité par b^{i+1} $\forall i \in \{0,...,p-1\}$

$$N = b.q_0 + r_0$$
$$b.q_0 = b^2.q_1 + b.r_1$$
$$\ldots$$
$$b^p.q_{p-1} = b^{p+1}.q_p + b^p.r_p$$

On effectue alors la somme des égalités membre à membre :

$$N + b.q_0 + ... + b^p.q_{p-1} = b.q_0 + ... + b^p.q_{p-1} + b^{p+1}.q_p + r_0 + br_1 + ... + b^p.r_p$$

Des termes se simplifient. Finalement, nous obtenons :

$$N = b^{p+1}.q_p + b^p.r_p + ... + br_1 + r_0$$

Où $0 \le r_i < |b| \quad \forall i \in \{0,...,p\}$

Et $0 < q_p < |b|$.

Nous pouvons alors poser $a_n = q_p$, $a_i = r_i \quad \forall i \in \{0,...,p\}$.

Nous avons dans ce cas $N = \overline{a_n ... a_1 a_0}^{b}$.

DEMONSTRATION DE L'UNICITE

L'unicité provient simplement de l'unicité des couples (q_i, r_i) dans la division euclidienne.

Illustrations…

En base décimale

$N = \overline{6734}^{10} = 6.10^3 + 7.10^2 + 3.10^1 + 4.10^0$

Ce même nombre N…

En base octale

$N = \overline{15116}^{8} = 1.8^4 + 5.8^3 + 1.8^2 + 1.8^1 + 6.8^0$

En base binaire

$N = \overline{1101001001110}^{2}$

$= 1.2^{12} + 1.2^{11} + 0.2^{10} + 1.2^9 + 0.2^8 + 0.2^7 + 1.2^6 + 0.2^5 + 0.2^4 + 1.2^3 + 1.2^2 + 1.2^1 + 0.2^0$

On voit que, plus la base est petite, plus le codage des nombres est fastidieux.

Propriétés

De la démonstration précédente, nous pouvons tirer les propriétés suivantes :

Si $b \in \mathbb{N}_{0,1}$ est une base de numération, alors

- $\overline{0}^{10} = \overline{0}^{b}$
- $\overline{1}^{10} = \overline{1}^{b}$
- $\overline{b}^{10} = \overline{10}^{b}$
- $\forall n \in \mathbb{N},\ \overline{b^n}^{10} = \overline{1\underbrace{0...0}_{n \text{ zéros}}}^{b}$

Codage d'un nombre non naturel

Il est finalement assez simple d'étendre la décomposition en base b aux entiers négatifs. Il suffit pour cela d'inventer un symbole, notre signe «-», et de construire $\mathbb{Z}^-$ en miroir au départ de $\mathbb{N}$.

La situation se corse par contre quand il s'agit de coder un nombre non entier q.

On pourrait très bien contourner le problème en ne considérant que les nombres rationnels représentés en écriture fractionnaire.

Soit $q \in \mathbb{Q}$. Alors $\exists\ \alpha, \beta \in \mathbb{Z} : q = \frac{\alpha}{\beta}$. On pourrait représenter q en base b, en décomposant α et β dans la base b voulue.

Donc, si $\alpha = \overline{\alpha_n ... \alpha_1 \alpha_0}^{b}$ et $\beta = \overline{\beta_n ... \beta_1 \beta_0}^{b}$, alors $q = \frac{\overline{\alpha_n ... \alpha_1 \alpha_0}^{b}}{\overline{\beta_n ... \beta_1 \beta_0}^{b}}$.

Cependant, si on souhaite écrire le nombre q sous forme développée en puissances de la base, ou si q est irrationnel, cela nécessite un nouvel algorithme de codage.

Si $q \notin \mathbb{Z}$, $\exists\, n \in \mathbb{Z}$: n<q<n+1 où n est la partie entière par défaut de q. On écrira $n \underset{noté}{=} [q]$.

On peut écrire : $q=[q]+\{q\}$ où $q-[q] \underset{noté}{=} \{q\}$

On peut coder un nombre en base b en traitant séparément la partie entière $[q]$ et la partie décimale $\{q\}$.

On sait déjà représenter l'entier $[q]$ en suivant la procédure de la page 44 : $[q]=\overline{a_n \ldots a_1 a_0}^{b}$.

Il reste à savoir comment représenter l'élément $\{q\} \in]0,1[$.

$\forall \{q\} \in]0,1[$, $\exists$ $(a_{-1},a_{-2},\ldots,a_{-n},\ldots)$ une suite – éventuellement infinie – de naturels vérifiant :

1. $a_{-i} < b, \forall i \in \mathbb{N}$
2. $\{q\} = \sum_{i=1}^{+\infty} a_{-i} b^{-i}$

Théorème que nous avons admis

Nous avons alors :

(1) $$q = [q]+\{q\} = \sum_{i=0}^{n} a_i b^i + \sum_{i=1}^{+\infty} a_{-i} b^{-i} \underset{noté}{=} \overline{a_n \ldots a_1 a_0 , a_{-1} a_{-2} \ldots a_{-n} \ldots}^{b}$$

Le type de représentation dans la décomposition d'un nombre en base *b* diffère selon la nature de celui-ci. En effet, la suite de chiffres du développement (1) peut être limitée, illimitée mais présenter une périodicité ou encore être infinie.

1. $\frac{1}{8}^{10} = \overline{0,125}^{10}$ est un rationnel présentant un développement décimal limité.
2. $\frac{1}{6}^{10} = \overline{0,1666\ldots6\ldots}^{10}$ est un rationnel présentant un développement décimal illimité mais périodique.
3. $\sqrt{2}^{10} = \overline{1,41421356\ldots}^{10}$ est un réel qui présente un développement décimal illimité non périodique.

Quelle que soit la base $b \in \mathbb{N}_{0,1}$ choisie, un nombre rationnel $q \in \mathbb{Q}$ possède toujours un développement limité (sa partie décimale se termine par une suite de 0) ou illimité mais périodique à partir d'un certain rang. Un nombre irrationnel est toujours représenté par un développement illimité non périodique.

En pratique, nous ne manipulons jamais de réels lors de calculs, nous ne manipulons que des approximations rationnelles de ces réels. Une machine (ordinateur ou calculatrice) approchera toujours un réel par un rationnel. Par conséquent, nous ne devons jamais perdre de vue que les opérations arithmétiques sur les réels ne doivent pas être considérées comme exactes.

Si nous n'imposons pas de condition supplémentaire, le nouvel algorithme de codage montre que ce système de numération positionnel est redondant : certains rationnels peuvent être codés de façons différentes.

Illustrations

$\overline{1}^{10} = \overline{0,9999...}^{10}$.

En effet, $\overline{0,9999...}^{10} = \sum_{i=1}^{+\infty} 9.10^{-i}$. Il s'agit de la somme d'une série convergente dont le terme général est 9.10^{-i}. Or, nous avons $\sum_{i=1}^{+\infty} 9.10^{-i} = \lim_{n\to+\infty} \sum_{i=1}^{n} 9.10^{-i} = \lim_{n\to+\infty} \frac{9}{10}.\frac{1-10^{-n}}{1-10^{-1}} = 1$.

$\overline{0,22}^{4} = \overline{0,213333...}^{4}$ pour la même raison.

Le développement en base b illimité d'un nombre réel, et à fortiori d'un nombre rationnel, est unique si on s'interdit de finir par une séquence périodique composée de 'b-1'.

II. Conversion de nombres en différentes bases.

Tout naturel excepté 0 et 1 peut fournir une base. De plus, l'écriture d'un nombre en base *b* s'écrit à l'aide de *b* chiffres allant de 0 à *b*-1. A partir de la base 11, nous devons utiliser des lettres afin d'éviter toute ambiguïté entre, par exemple, le nombre 13 et les chiffres 1 et 3.

Voici ce que l'on obtient :

- A = $\overline{10}^{10}$
- B = $\overline{11}^{10}$
- C = $\overline{12}^{10}$
- D = $\overline{13}^{10}$
- ...

Il existe différentes notations pour désigner la base d'un nombre.

- $\overline{1011101}^{2}$
- $(1011101)_2$

Le « 2 » noté sous forme d'indice ou d'exposant indique ici qu'on compte dans la base 2.

Il n'y a que dans le système en base 10 que l'on peut utiliser les termes centaines, milliers, etc. Dans les autres bases, on doit lire le nombre chiffre par chiffre. Le nombre $\overline{101101}^{2}$ se dit dès lors "un, zéro, un, un, zéro, un".

Les bases les plus couramment utilisées sont:

- la base 2 (système binaire): en électronique numérique et informatique,
- la base 3 (système ternaire) : en électronique et dans le développement des logiciels.
- la base 8 (système octal): en informatique
- la base 9 (système nonaire) : est au système ternaire ce qu'est le système octal au binaire, une version compacte.
- la base 10 (système décimal): notre base actuelle
- la base 12 (système duodécimal)
- la base 16 (système hexadécimal): en informatique
- la base 20 (système vigésimal)

Figure II. 4 : *Al-Khawarizmi (783-850), mathématicien perse, dont le nom est à l'origine des mots « algorithme » et « algèbre ».*

Conversion d'un nombre de la base *b* vers la base 10

En ce qui concerne la conversion de la base *b* à la base 10 des nombres entiers positifs, il faut écrire le nombre dans sa représentation polynomiale. En d'autres termes, il faut représenter le nombre comme la somme de puissances successives de la base. On obtient donc:

$$\overline{a_n a_{n-1} \ldots a_2 a_1 a_0}^{b} = \overline{a_n b^n + a_{n-1} b^{n-1} + \ldots + a_2 b^2 + a_1 b^1 + a_0 b^0}^{10} = \overline{\sum_{i=0}^{n} a_i b^i}^{10}$$

Illustrations...

$\overline{101101}^{2} = 1.2^5 + 0.2^4 + 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 = \overline{45}^{10}$

$\overline{743}^{8} = 7.8^2 + 4.8^1 + 3.8^0 = \overline{483}^{10}$

$\overline{3980}^{8}$ est une représentation impossible car 9 et 8 ne sont pas des chiffres de la base 8 (8>7 et 9>7).

La conversion des nombres entiers négatifs fonctionne comme la conversion des nombres entiers positifs, il nous faut juste rajouter un signe devant pour montrer que l'on travaille avec des nombres négatifs. On obtient donc:

$$\overline{-a_n a_{n-1} \ldots a_2 a_1 a_0}^{b} = \overline{-(a_n b^n + a_{n-1} b^{n-1} + \ldots + a_2 b^2 + a_1 b^1 + a_0 b^0)}^{10} = \overline{-\sum_{i=0}^{n} a_i b^i}^{10}$$

Illustration...

$\overline{-101101}^{2} = -(1.2^5 + 0.2^4 + 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0) = \overline{-45}^{10}$

Pour convertir des nombres non entiers $q \notin \mathbb{Z}$, il faut aussi représenter le nombre sous sa forme polynomiale, en séparant le codage de la partie entière $[q]$ et de la partie restante $\{q\}$ (cfr. page 47).

Cette représentation deviendra donc :

$$N = \overline{a_n a_{n-1} \ldots a_2 a_1 a_0 , a_{-1} a_{-2} \ldots a_{-m+1} a_{-m}}^{b}$$

$$N = \overline{a_n b^n + a_{n-1} b^{n-1} + \ldots + a_2 b^2 + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + \ldots + a_{-m} b^{-m}}^{10}$$

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

Illustration...

$$\overline{26,54}^{8} = 2.8^1 + 6.8^0 + 5.8^{-1} + 4.8^{-2} = \frac{\overline{363}}{16}^{10} = \overline{22,6875}^{10}$$

Les **nombres non entiers négatifs** ont le même système de conversion que les nombres décimaux positifs. En effet, comme pour les nombres entiers, il faut ajouter un signe afin de savoir que l'on travaille avec des nombres négatifs. La conversion ressemblera donc à :

$$N = -\overline{a_n a_{n-1} \ldots a_2 a_1 a_0 , a_{-1} a_{-2} \ldots a_{-n+1} a_{-n}}^{b}$$

$$N = -\overline{a_n b^n + a_{n-1} b^{n-1} + \ldots + a_2 b^2 + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + \ldots + a_{-m} b^{-m}}^{10}$$

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

Illustration...

$$-\overline{3,741}^{8} = -3.8^0 - 7.8^{-1} - 4.8^{-2} - 1.8^{-3} = -\frac{\overline{2017}}{512}^{10} = -\overline{3,939453125...}^{10}$$

On a l'impression que ce nombre perd sa nature rationnelle selon qu'il est codé en base 8 ou en base 10. Il n'en est rien. Un nombre au développement illimité périodique le conserve, seule la période change.

Par contre, les **nombres irrationnels** sont impossibles à convertir. En effet, les nombres irrationnels sont des nombres dont la représentation est infinie et non périodique. De plus, il est impossible de les écrire sous la forme d'une fraction $\frac{a}{b}$ telle que *a* et *b* soient des nombres entiers relatifs et *b* différent de 0. Il est donc impossible de les convertir. Toutefois, on peut convertir des nombres proches de leur valeur en arrondissant les nombres irrationnels voulus. C'est ce que font les processeurs d'ordinateurs lorsqu'ils convertissent une quantité exprimée dans le système décimal en une expression binaire par exemple.

Conversion d'un nombre de la base 10 vers la base *b*

En ce qui concerne la conversion de la base 10 à la base *b* des **nombres entiers positifs**, il existe plusieurs méthodes : une méthode algébrique et une méthode utilisant des tableaux, plus ergonomique.

POUR LA METHODE ALGEBRIQUE

Convertissons le nombre N= $\overline{a_n a_{n-1} ... a_2 a_1 a_0}^{10}$.

Soit à obtenir, N sous la forme $\overline{a'_{n'} ... a_1 'a'_0}^{b}$.

Pour convertir ce nombre, nous procédons à l'algorithme que décrit la démonstration en page 44.

Par une succession de divisions par *b*, nous cherchons à obtenir les suites de couples (q_i, r_i) jusqu'à l'obtention d'un q_p tel que $0 < q_p < |b|$.

$$N = b.q_0 + r_0$$
$$q_0 = b.q_1 + r_1$$
$$...$$
$$q_{p-1} = b\ .q_p\ + r_p$$

En substituant les quotients successifs par leur expression fournie par la division euclidienne, on peut obtenir la représentation polynomiale du nombre.

$$N = b^{p+1}.q_p + b^p.r_p\ + ... + br_1 + r_0$$

Nous avons $a'_{n'} = q_p$, $a'_i = r_i\ \ \forall i \in \{0,...,p\}$.

Et nous obtenons $N = \overline{a'_{n'} ... a_1 'a'_0}^{b}$

Illustration…

$$\begin{aligned}\overline{3980}^{10} &= 497.8+4\\ &=(62.8+1).8+4\\ &=((7.8+6).8+1).8+4\\ &=7.8^3+6.8^2+1.8^1+4.8^0\\ &=\overline{7614}^{8}\end{aligned}$$

POUR LA METHODE PAR TABLEAUX

Il existe plusieurs présentations utilisant des tableaux pour faciliter la conversion d'un nombre.

Voici deux exemples de présentation avec leur méthode :

Soit N= $\overline{a_n a_{n-1}...a_2 a_1 a_0}^{10}$.

Nombre à convertir	Base désirée (b)
Quotient de la division de N par b (q_0)	Reste de la division du nombre N par b (r_0)
Quotient de la division de q_0 par b (q_1)	Reste de la division de q_0 par b (r_1)

Pour obtenir le nombre dans la base désirée, il nous faut continuer ainsi de suite jusqu'à l'obtention d'un quotient nul.

Pour coder le nombre N, il faut lire les restes successifs de bas en haut.

Illustration… $\overline{3980}^{10} = \overline{?}^{8}$

Suite des quotients q_i

Suite des restes r_i

3980	8
497	4
62	1
7	6
0	7

Finalement, $\overline{3980}^{10} = \overline{7614}^{8}$

Voici une autre façon de présenter les choses :

Nombre à convertir N	Base désirée *b*	
	Quotient de la division de N par *b* (q_0)	Base désirée *b*
	Reste de la division du nombre N par *b* (r_0)	Quotient de la division de q_0 par *b* (q_1)
		Reste de la division de q_0 par *b* (r_1)

Comme pour la première représentation, pour obtenir le nombre dans la base désirée, il nous faut continuer ainsi de suite jusqu'à ce qu'on obtienne un quotient nul. Pour connaître le nombre recherché, il faut lire les restes successifs de bas en haut.

Illustration... $\overline{3980}^{10} = \overline{?}^{8}$

3980	8			
	497	8		
	4	62	8	
		1	7	8
			6	0
				7

Suite des quotients q_i

Suite des restes r_i

Finalement, $\overline{3980}^{10} = \overline{7614}^{8}$

La conversion des **nombres entiers négatifs** fonctionne de la même façon que celle des nombres entiers positifs. Comme vu précédemment, il nous faut juste rajouter le signe"-" pour montrer que l'on parle d'un nombre négatif.

Avec les **nombres décimaux**, pour passer de la base 10 à une base *b*, il faut séparer le nombre en deux parties. La partie entière et la partie décimale.

Le nombre N= $\overline{a_n a_{n-1}...a_2 a_1 a_0 , a_{-1} a_{-2}...a_{-m+1} a_{-m}}^{10}$ deviendra alors

N= $\overline{a_n a_{n-1}...a_2 a_1 a_0}^{10} + \overline{a_{-1} a_{-2}...a_{-m+1} a_{-m}}^{10} == \sum_{i=0}^{n} a_i 10^i + \sum_{i=-m}^{-1} a_i 10^i$

Pour exprimer dans une autre base un nombre décimal, il faut donc effectuer deux démarches.

Premièrement, il faut convertir la partie entière selon la méthode vue précédemment (« *Passage de la base 10 à une base b d'un nombre entier positif* »).

Comme pour la partie entière, il existe plusieurs méthodes pour convertir la partie décimale d'un nombre : on peut convertir la partie décimale algébriquement ou au moyen de tableaux de conversion.

POUR LA METHODE ALGEBRIQUE

Convertissons la partie décimale $\overline{a_{-1} a_{-2}...a_{-m+1} a_{-m}}^{10}$ d'un nombre N.

Pour convertir la partie décimale du nombre, nous devons la multiplier par $\frac{b}{b}$, b étant la base finale désirée.

Le numérateur de la fraction obtenue peut s'exprimer comme la somme de sa partie entière (E_1) et de sa partie décimale (D_1).

On recommence la multiplication de D_1 par $\frac{b}{b}$ et sa décomposition en partie entière et décimale. On obtient une suite (E_i, D_i) qui s'arrête à l'obtention d'une partie décimale nulle.

On obtient une représentation polynomiale de la partie décimale, avec des puissances de b négatives.

$$\overline{a_{-1}...a_{-m+1} a_{-m}}^{10} = \frac{b}{b} \times \overline{(a_{-1}...a_{-m+1} a_{-m})}^{b}$$

$$= \frac{1}{b} \times \overline{(b \times a_{-1}...a_{-m+1} a_{-m})}^{b}$$

$$= \frac{1}{b} \times \overline{(E_1 + D_1)}^{b}$$

$$= \overline{\frac{E_1}{b} + \frac{D_1}{b}}^{b}$$

$$= \overline{\frac{E_1}{b} + (\frac{b}{b} \times \frac{D_1}{b})}^{b}$$

$$= \overline{\frac{E_1}{b} + \frac{E_2}{b^2} + \frac{D_2}{b^2}}^{b}$$

...

Pour obtenir la conversion du nombre recherché, il faut additionner la conversion de la partie entière (faite avec la méthode « *Passage de la base 10 à une base b d'un nombre entier positif* ») et la conversion de la partie décimale.

Illustrations...

$$\overline{0,8125}^{10} = \overline{?}^{8}$$

$$\begin{aligned}
\overline{0,8125}^{10} &= \frac{8}{8} \times 0,8125 \\
&= \frac{6,5}{8} \\
&= \frac{6}{8} + \frac{0,5}{8} \\
&= \frac{6}{8} + \frac{8}{8} \times \frac{0,5}{8} \\
&= \frac{6}{8} + \frac{4}{8^2} \\
&= 6.8^{-1} + 4.8^{-2} \\
&= \overline{0,64}^{8}
\end{aligned}$$

$$\overline{94,8125}^{10} = \overline{94 + 0,8125}^{10}$$

$$\begin{aligned}
\overline{94,8125}^{10} &= \left(11.8 + 6\right) + \left(\frac{6}{8} + \frac{0,5}{8}\right) \\
&= \left(\left(1.8 + 3\right).8 + 6\right) + \left(\frac{6}{8} + \frac{4}{8^2}\right) \\
&= 1.8^2 + 3.8^1 + 6.8^0 + 6.8^{-1} + 4.8^{-2} \\
&= \overline{136,64}^{8}
\end{aligned}$$

POUR LA METHODE PAR TABLEAUX

Pour la conversion des nombres décimaux par le biais de tableaux, on a relevé une seule représentation :

Partie décimale du nombre à convertir (= Déc.)	Base désirée (b)
Partie décimale de Déc. multiplié par b (D_1)	Partie entière de Déc. multiplié par b (E_1)
Partie décimale de D_1 multiplié par b (D_2)	Partie entière de D_1 multiplié par b (E_2)

Pour obtenir la conversion du nombre, il nous faut continuer ainsi de suite jusqu'à l'obtention d'un nombre entier sans partie décimale. Pour connaître la partie décimale recherchée, il faut lire les parties entières successives de haut en bas. Comme cité plus tôt dans ce document, pour clore le processus, il faut juxtaposer les codages obtenus par la conversion des parties entières et décimales du nombre N.

Illustration... $\overline{94,8125}^{10} = \overline{94+0,8125}^{10} = \overline{?}^{8}$

94	8
11	6
1	3
0	1

8	0,8125
0.5	6
0	4

Parties décimales D_i

Parties entières E_i

Finalement, $\overline{94,8125}^{10} = \overline{136,64}^{8}$

Le passage d'une base à l'autre peut réserver parfois des surprises. C'est ce que montre l'exemple suivant.

$\overline{3,948}^{10} = \overline{3+0,948}^{10} = ?^{8}$

3	8
0	3

0,948	8
0,584	7
0,672	4
0,376	5
0,008	3
0,064	0
0,512	0
0,096	4
…	…

$\overline{3,948}^{10} = \overline{3,7453004\ldots}^{8}$

On a l'impression que certains nombres rationnels, c'est-à-dire, des nombres dont l'écriture est limitée ou illimitée périodique dans une base b se comportent comme des irrationnels dans une autre base : l'écriture de la partie inférieure à 1 se fait de façon apparemment illimitée.

En réalité, l'écriture de tout nombre rationnel, si elle n'est pas limitée, est illimitée périodique, et ce, quelle que soit la base $b \in \mathbb{N}_{0,1}$. Le caractère périodique ne disparaît donc pas, seule la période peut effectivement changer.

De la même façon, tout nombre irrationnel sera toujours représenté avec une suite illimitée non périodique de chiffres.

Voici divers exemples:

$\overline{3,948}^{10} = \overline{3,74530040611156457}^{8}$

Nous observons que ce que nous prenions pour écriture illimitée non périodique est en réalité limitée.

$\overline{4,11\ldots11}^{10} = \overline{4,0707\ldots0707}^{8} = \overline{100,000111000111\ldots000111}^{2}$

La période vaut 1 en écriture décimale, 2 en numération octale et 6 en numération binaire.

$\overline{0,6}^{10} = \overline{0,12101210\ldots1210}^{3}$ et $\overline{0,111\ldots1}^{10} = \frac{\overline{1}^{10}}{9} = \overline{0,01}^{3}$.

Ces derniers exemples montrent qu'un rationnel de représentation limitée dans une base peut s'écrire avec une suite illimitée mais périodique dans une autre base.

Conversion d'une base b_1 vers une autre base b_2

Quand b_1 **n'est pas une puissance de** b_2, il n'existe aucune méthode pratique pour passer de la base b_1 à la base b_2. On doit obligatoirement passer par la base 10 pour convertir un nombre.

Par contre, il est possible de simplifier le changement de base quand b_1 **est une puissance de** b_2.

Observons avec l'exemple qui suit. Convertissons un nombre du système octal au système binaire.

Dans ce cas, nous avons $b_1 = 8$ $b_2 = 2$.

Et nous observons que $b_1 = b_2^3$.

Baptisons k la puissance à laquelle est élevée b_2.

Ici, k=3.

Voici une méthode rapide pour passer de la base b_1 à la base b_2.

Illustration... $\overline{1345}^{8} = \overline{?}^{2}$

1. Ecrire chaque chiffre du nombre en base b_1 dans une colonne.

2. Exprimer chacun des chiffres de la base b_1 dans la base b_2.

Ici, k=3, il y a donc automatiquement 3 chiffres par colonne. On complète par des « 0 » si nécessaire pour obtenir des assemblages en triplet.

3. Ecrire le nombre final en n'oubliant pas les zéros.

1	3	4	5
$1*2^0$	$1*2^1+1*2^0$	$1*2^2$	$1*2^2+1*2^0$
001	011	100	101

$$\overline{1345}^{8} = \overline{1011100101}^{2}$$

Remarquons qu'il y aura 4 chiffres par colonnes, pour le passage de la base hexadécimale (b_1=16) à la base binaire, car comme $b_1 = b_2^4$, k=4.

Illustration... $\overline{2E5}^{16} = \overline{?}^{2}$

1. Ecrire chaque chiffre du nombre en base b_1 dans une colonne.

2. Exprimer chacun des chiffres de la base b_1 dans la base b_2.

Ici, k=4, il y a donc automatiquement 4 chiffres par colonne.

3. Ecrire le nombre final en n'oubliant pas les zéros.

2	E ($=\overline{14}^{10}$)	5
$1*2^1$	$1*2^3+1*2^2+1*2^1$	$1*2^2+1*2^0$
0010	1110	0101

$$\overline{2E5}^{16} = \overline{1011100101}^{2}$$

A l'inverse, pour passer **d'une base b_1 à b_2** quand **b_2 est une puissance de b_1**, il faut fractionner le nombre de départ.

Convertissons un nombre en allant du système binaire vers le système octal.

Dans ce cas, nous avons $b_1 = 2$ $b_2 = 8$.

Et nous observons que $b_2 = b_1^3$.

Baptisons k la puissance à laquelle est élevée b_1. Ici, k=3.

Illustration... $\overline{1011100101}^{2} = \overline{?}^{8}$

1. On fractionne le nombre N en parties de k chiffres, en commençant par la droite.

Ici, k=3, on effectue des séparations tous les 3 chiffres.

2. Ensuite, il faut convertir les nombres obtenus dans la base b_2.

3. Enfin, on juxtapose les chiffres obtenus à l'étape 2 et on obtient le nombre final.

$$\overline{1|011|100|101}^{2}$$

001	011	100	101
$1*2^0$	$1*2^1+1*2^0$	$1*2^2$	$1*2^2+1*2^0$
1	3	4	5

$$\overline{1011100101}^{2} = \overline{1345}^{8}$$

Remarquons que la séparation s'effectuera tous les 4 chiffres pour le passage de la base binaire à la base hexadécimale (b_2=16), car, comme $b_2 = b_1^k$, k=4.

Illustration... $\overline{1011100101}^{2} = \overline{?}^{16}$

1. On fractionne le nombre N en parties de k chiffres, en commençant par la droite.

$$\overline{10|1110|0101}^{2}$$

Ici, k=4, on effectue des séparations tous les 4 chiffres.

10	1110	0101
$1*2^1$	$1*2^3+1*2^2+1*2^1$	$1*2^2+1*2^0$
2	14 = E	5

2. Ensuite, il faut convertir les nombres obtenus dans la base b_2.

3. Enfin, on juxtapose les chiffres obtenus à l'étape 2 et on obtient le nombre final.

$$\overline{1011100101}^{2} = \overline{2E5}^{16}$$

Jusqu'ici on a parlé de systèmes de numération assez conventionnels. Il en existe d'autres, plus exotiques et amusants et nous vous en présentons quelques-uns dans cette partie.

Les systèmes d'Avizienis

Les chiffres utilisés dans les systèmes de numération positionnels décrits dans les parties précédentes appartenaient à l'ensemble des naturels. Qu'arrive-t-il si on autorise des entiers négatifs ? Quel avantage y trouve-t-on ?

Voici un système de numération inventé pour faciliter l'addition et autorisant les chiffres négatifs.

Soit b>1. Un système de numération d'Avizienis de base b est la donnée d'un ensemble de chiffres C tel que :

1. $C=\{-a,-a+1,...,0,...,a-1,a\}$
2. $a \in \mathbb{N}, a \leq b-1$
3. $b+1 \leq 2a$

L'ensemble C regroupe l'ensemble des chiffres nécessaires pour écrire les nombres en base b. Evidemment, à cause des conditions infligées sur C, les systèmes d'Avizienis ne peuvent être utilisés que sur des bases strictement plus grandes que 2.

Pour chaque base valide, deux possibilités d'ensembles existent.

$$C_b^1 = \{-(b-1),-(b-2),...,0,...,b-2,b-1\}$$

Ou

$$C_b^2 = \left\{-E\left(\frac{b+1}{2}\right),-E\left(\frac{b}{2}\right),...,0,...,E\left(\frac{b}{2}\right),E\left(\frac{b+1}{2}\right)\right\}$$

Où $E(x)$ représente la fonction partie entière.

La coexistence de deux ensembles de chiffres et le fait que l'ensemble C_b^1 contienne plus de b éléments fait du système d'Avizienis un système de numération redondant : on peut coder un nombre de plusieurs façons dans celui-ci.

En général, les chiffres négatifs sont représentés surmontés d'une barre.

Illustration… Considérons le système d'Avizienis de base 10.

$C_{10}^{1} = \{\bar{9},\bar{8},\bar{7},\bar{6},\bar{5},\bar{4},\bar{3},\bar{2},\bar{1},0,1,2,3,4,5,6,7,8,9\}$.

Dans ce système à base 10, le nombre 6 admet logiquement 6 comme écriture mais également $1\bar{4}$ (10-4=6) ou encore $1\overline{94}$ (100-94=6), $1\overline{994}$, etc.

Nous pouvons utiliser $C_{10}^{2} = \{\bar{5},\bar{4},\bar{3},\bar{2},\bar{1},0,1,2,3,4,5\}$. Dans ce cas, le nombre de possibilités d'écriture est réduit.

Voyons maintenant un système d'Avizienis de base 8.
$C_{8}^{2} = \{\bar{4},\bar{3},\bar{2},\bar{1},0,1,2,3,4\}$

Dans ce système à base 8, le nombre 6 n'admet plus 6 comme écriture mais $1\bar{2}$ (8-2=6).

L'avantage de ce système est qu'il évite les retenues dans les additions.

Plutôt que ce calcul :

	+1		+1	
		7	1	4
+		5	4	9
	1	2	6	3

On effectue ce calcul :

	1	$\bar{3}$	1	4
+		5	5	$\bar{1}$
	1	2	6	3

Par ailleurs, le passage d'un nombre N à son opposé –N se fait facilement : il suffit de changer le signe des chiffres du nombre N, ce qui implique qu'il faut additionner pour soustraire.

$$-\bar{1}\bar{3}\bar{4}2 = 1\bar{3}\bar{4}\bar{2}$$

L'avantage cité ci-dessus fait des systèmes d'Avizienis des systèmes très utilisés dans l'informatique et plus spécialement dans le codage, afin de réaliser des économies substantielles sur les temps de calcul.

Base Fibonacci dans la numération de Zeckendorf

Tout le monde connaît la suite des nombres de Fibonacci $(F_n)_{n\in\mathbb{N}}$, définie par la formule de récurrence suivante :

$$\begin{cases} F_0 = 1 \\ F_1 = 2 \\ F_{n+2} = F_{n+1} + F_n \quad \forall n \in \mathbb{N} \end{cases}$$

La suite se développe comme suit :

k	0	1	2	3	4	5	6	7	8	9	...
F_k	1	2	3	5	8	13	21	34	55	89	

La base Fibonacci utilise les valeurs de cette célèbre suite.

On doit cette numération, certes peu pratique, à un de nos compatriotes, Edouard Zeckendorf (1901 -1983), qui énonça ce théorème

« Tout nombre N appartenant aux naturels strictement positifs peut s'écrire sous une et une seule forme d'une somme de nombres non consécutifs de la suite de Fibonacci.»

Voyons par un exemple comment ce système de numération fonctionne.

Illustration... Soit à décomposer N=30.

L'observation de la suite permet d'écrire

$$30 = 21+8+1$$

Ou encore $30 = 1.F_6 + 0.F_5 + 1.F_4 + 0.F_3 + 0.F_2 + 0.F_1 + 1.F_0$

C'est-à-dire 30 = 1010001_Z

On peut donc assimiler cette écriture à une numération positionnelle ne possédant que deux symboles (0 et 1).

En généralisant, nous avons :

$$\forall n \in \mathbb{N}_0 : n = \sum_{i=0}^{k} \varepsilon_i(n).F_i$$

où $\varepsilon_i(n) \in \{0,1\}$ et $\varepsilon_k(n) = 1$,

avec $\varepsilon_i(n).\varepsilon_{i+1}(n) = 0$ *(2 termes consécutifs ne sont pas permis).*

On pourrait pousser l'investigation plus loin en considérant une suite de « Tribonacci » $(T_n)_{n\in\mathbb{N}}$ et son système de numération de la façon suivante :

$$\begin{cases} T_0 = 1 \\ T_1 = 2 \\ T_3 = 3 \\ T_{n+3} = T_{n+2} + T_{n+1} + T_n \quad \forall n \in \mathbb{N} \end{cases}$$

Ce qui donnerait ces premières valeurs :

k	0	1	2	3	4	5	6	7	8	…
T_k	1	2	3	6	11	20	37	68	125	

Illustration… Soit à décomposer N=100.

$100 = 68+20+11+1$

Ou encore $100 = 1.T_7+0.T_6+0.T_5+1.T_4+0.T_3+0.T_2+0.T_1+1.T_0$

C'est-à-dire 100 = 10010001_T

Cette numération positionnelle n'employant que deux chiffres (0 et 1) n'est pas redondante pour autant qu'on interdise une séquence de trois chiffres « 1 » consécutifs dans l'écriture du nombre. Ce principe peut être généralisé à des systèmes de numération de k-bonacci.

Si la numération de Zeckendorf est peu commode à employer dans les opérations usuelles comme l'addition et la multiplication, elle peut se révéler utile dans certains problèmes dont la solution réside dans la décomposition en termes de Fibonacci. Ainsi, une application ludique du théorème de Zeckendorf est le jeu de « Fibonacci-Nim ».

Dans ce jeu, comme dans le jeu de Nim à un tas, des bâtons sont alignés et les joueurs en prélèvent tour à tour un certain nombre en respectant les règles suivantes :

- Si un joueur prend k bâtons, le joueur suivant doit prendre entre 1 et 2k pions.
- Le joueur qui commence doit obligatoirement laisser un pion au premier coup.
- Le joueur qui prend le dernier pion gagne la partie.

La stratégie gagnante repose sur la décomposition du nombre N de bâtons restant sur le plateau en base Fibonacci. Il s'agit en effet de retirer à chaque fois un nombre de bâtons égal au plus petit terme de la décomposition_Z de N et continuer de la sorte en plaçant à chaque coup son adversaire dans une position perdante.

Base en or

Nous avons vu avec le système d'Avizienis que l'on pouvait coder un nombre en base *b* en utilisant un ensemble de chiffres négatifs.

Tout aussi surprenant, il existe des systèmes de numération positionnels employant une base non entière, voire même irrationnelle.

Par exemple, aussi célèbre que la suite de Fibonacci à laquelle il est associé, le nombre d'or φ peut être utilisé comme base de numération. On parle alors dans ce cas de base d'or, ou base phinaire.

Pour rappel, le nombre d'or est l'unique solution positive de l'équation « $x^2=x+1$ » et vaut $\frac{1+\sqrt{5}}{2}$, soit en l'arrondissant au millième $\varphi \sim 1{,}618$ (exprimé dans le système décimal).

Soit $N \in \mathbb{N}_0$. Pour effectuer le développement du nombre N dans la base d'or, il faut utiliser un algorithme glouton, c'est-à-dire comparer N à des puissances décroissantes de φ.

$N>0 \Rightarrow \exists\, n \in \mathbb{N} : \quad \varphi^n \leq N < \varphi^{n+1}$.

Comme $1 < \varphi < 2$, cela signifie automatiquement que

$N = 1.\varphi^n + r_n$.

On recommence avec r_n de la même façon.

Il est possible de montrer ce fait remarquable : malgré l'usage d'une base irrationnelle, tous les naturels possèdent une représentation unique (pour autant que l'on évite deux « 1 » consécutifs dans l'écriture du nombre) dans la base phinaire en développement fini, et seuls les chiffres 0 et 1 sont utilisés.

De la même façon, les nombres rationnels possèdent des représentations présentant une périodicité.

Illustrations... Codons 1, 2, 3, 4 et 5 en base phinaire

Il est très facile de coder 1 : $\overline{1}^{10} = 1_\varphi$ car $\varphi^0 = 1$.

Nous observons ensuite que, par définition, (1) $\varphi^2 = \varphi + 1$

En la divisant par φ, nous obtenons cette relation : $\varphi = 1 + \varphi^{-1}$

Et en divisant (1) par φ^2, nous obtenons celle-ci: $1 = \varphi^{-1} + \varphi^{-2}$

On voit ici que, sans la condition de l'évitement des séquences de type « 11 » dans le codage, la base phinaire constituerait un système redondant, puisqu'on pourrait écrire également : $\overline{1}^{10} = 0,11_\varphi$

On peut également en tirer le codage de 2 : $\overline{2}^{10} = \varphi^1 + \varphi^{-2} = 10,01_\varphi$

Ajoutons 1, nous obtenons : $\overline{3}^{10} = \varphi^1 + \varphi^0 + \varphi^{-2} = 11,01_\varphi$

Ajoutons de nouveau $1 = \varphi^2 - \varphi$: $\overline{4}^{10} = \varphi^2 + \varphi^0 + \varphi^{-2} = 101,01_\varphi$

En effectuant $\overline{5}^{10} = \overline{3}^{10} + \overline{2}^{10}$, nous obtenons :

$$\overline{5}^{10} = \underbrace{\varphi^1 + \underbrace{(\varphi^1 + \varphi^0)}_{\varphi^2}}_{\varphi^3} + \varphi^{-2} + \varphi^{-2} = \varphi^3 + \varphi^{-1} - \varphi^{-3} + \varphi^{-3} + \varphi^{-4} = \varphi^3 + \varphi^{-1} + \varphi^{-4} = 1000,1001_\varphi$$

Un algorithme de décomposition permet de développer des valeurs décimales.

Illustration... Codons $\frac{1}{2}$ en base phinaire.

$$\varphi^{-2} \le \frac{1}{2} < \varphi^{-1} \Rightarrow \frac{1}{2} = 1.\varphi^{-2} + r_{-2} \text{ avec } 0 < r_{-2} = \frac{1}{2} - \varphi^{-2} < \varphi^{-2}$$

$$r_{-2} \approx 0,118 \Rightarrow \varphi^{-5} < r_{-2} < \varphi^{-4} \Rightarrow r_{-2} = 1.\varphi^{-5} + r_{-5} \text{ avec } 0 < r_{-5} = r_{-2} - \varphi^{-5} < \varphi^{-5}$$

$$r_{-5} \approx 0,028 \Rightarrow \varphi^{-8} < r_{-5} < \varphi^{-7} \Rightarrow r_{-5} = 1.\varphi^{-8} + r_{-8} \text{ avec } 0 < r_{-8} = r_{-5} - \varphi^{-8} < \varphi^{-8}$$

$$r_{-8} \approx 0,007 \Rightarrow \varphi^{-11} < r_{-8} < \varphi^{-10} \Rightarrow r_{-8} = 1.\varphi^{-11} + r_{-11} \text{ avec } 0 < r_{-11} = r_{-8} - \varphi^{-11} < \varphi^{-11}$$

Finalement, $\frac{1}{2} = 1.\varphi^{-2} + 1.\varphi^{-5} + 1.\varphi^{-8} + 1.\varphi^{-11} + \ldots = \sum_{i=0}^{n} \varphi^{-(2+3i)}$

Ou encore : $\frac{1}{2} = 0,0100\overline{1001}_\varphi$

Il ne faut pas croire que l'emploi de la base d'or en numération est le fait de mathématiciens excentriques. Elle peut se révéler très utile, notamment dans la mise en place de codes correcteurs d'erreurs dont le rôle est de protéger l'information d'erreurs de transmission ou de stockage, inévitablement rencontrées dans les communications à distance.

Très souvent, le message transmis est numérisé dans un alphabet binaire et ramené à une séquence de bits 0 et 1. Lors de la transmission, d'inévitables parasites détériorent le message et une partie de l'information, des symboles 0 et 1 en quelque sorte sont perdus.

Pour améliorer la fiabilité de la transmission des données, une des méthodes de codage les plus simples est alors de répéter chaque bit : la séquence de bits sera donc répliquée une ou plusieurs fois. C'est cette redondance qui permet la détection et la correction d'erreurs. Lors de la réception du message, le décodeur peut ainsi comparer chaque couple de bits reçus. S'ils sont différents, alors il y a détection d'erreur.

Le caractère redondant de la numération phinaire qui utilise l'alphabet binaire peut être un avantage. Il suffit de coder la même valeur en utilisant plusieurs représentations et ensuite de les comparer à la réception du message.

Les systèmes reposant sur l'emploi d'une base irrationnelle furent introduits en 1957 par le mathématicien George Bergman qui les baptisa « *Tau systems* ».

Juste pour rire… la numération Shadok

La numération Shadok est une invention rigolote tirée du dessin animé "Les Shadoks" créé par Jacques Rouxel.

Celui-ci l'introduit comme suit :

Figure II. 5 :
Une devise Shadok

"Les Shadoks ressemblaient à des oiseaux : ils avaient un bec et des pattes mais leurs ailes étaient ridiculement ridicules ! Eduquer les Shadoks n'était pas chose facile. Leurs cerveaux, en effet, avaient une capacité tout à fait limitée. Ils ne comportaient en tout que quatre cases. Et encore, ce n'était pas toujours vrai parce que bien souvent il y en avait de bouchées. Pour remplir les cases déjà, ce n'était pas facile et cela prenait un certain temps. C'est alors que commençait la difficulté parce que, quand les cases étaient pleines, il n'y avait plus de place et le Shadok, on ne pouvait plus rien lui apprendre. Si on essayait quand même, alors obligatoirement il y avait une case qui se vidait pour faire de la place. De sorte que, quand un Shadok avec une tête pleine voulait apprendre quelque chose, il fallait qu'il en oublie une autre. Comme ils n'avaient que quatre cases, évidemment les Shadoks ne connaissaient pas plus de quatre sons : GA BU ZO MEU."

Vous l'aurez compris, s'ils veulent compter, les Shadoks ne peuvent utiliser que la base 4 ! Avec les conventions suivantes :

0=GA	1= BU	2=ZO	3=MEU

Pour le fun, imaginez ce que peut endurer un Shadok vous souhaitant une bonne année 2014.

Sachant que $\overline{2014}^{10} = \overline{133132}^{4}$, voici le décompte:

BU-MEU-MEU-BU-MEU-ZO !

IV. Les bases dans l'informatique et dans l'électronique

Depuis leur apparition, les systèmes informatiques et électroniques ont fonctionné à l'aide du binaire. Cette base est, en effet, l'unique utilisable pour le traitement des informations par les processeurs. Ces processeurs sont composés de transistors, qui ne peuvent gérer que les deux états du courant : soit le courant est présent, soit il ne l'est pas (on et off). Lorsque le courant passe dans le transistor, l'ordinateur reçoit un message « 1 ». Lorsqu'il ne passe pas, le message envoyé est « 0 ». De cette manière, l'ordinateur peut traiter de grandes quantités d'informations.

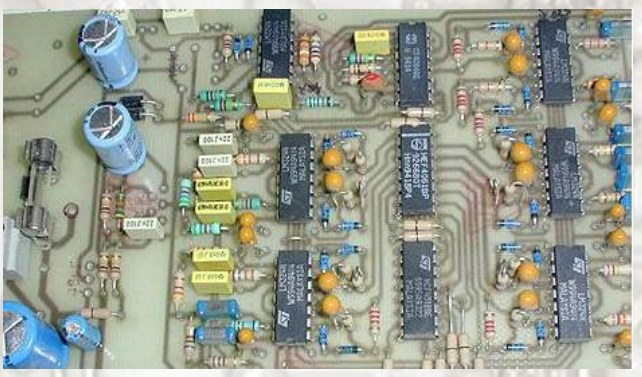


Figure II. 6 : *Les processeurs sont des systèmes automatiques de traitement des informations. Ils manipulent ces informations sous forme de données binaires (groupe de bits).*

Le système de numération binaire est donc parfaitement approprié pour tous les circuits qui, pour des raisons technologiques (passage de courant ou non, sens de polarisation, magnétisation, perforation ou non, vrai ou faux) ne présentent que deux états possibles. Les chiffres 0 et 1 de ce système portent le nom de bits (**BI**nary digi**T**). Un groupe de 8 bits se nomme un octet ou byte et un groupe de 4 bits se nomme un quartet.

Le binaire est aussi très utilisé dans la programmation des systèmes électroniques de la vie courante comme dans les télécommandes, les machines à laver ou encore les serrures à code pour n'en citer que quelques-uns.

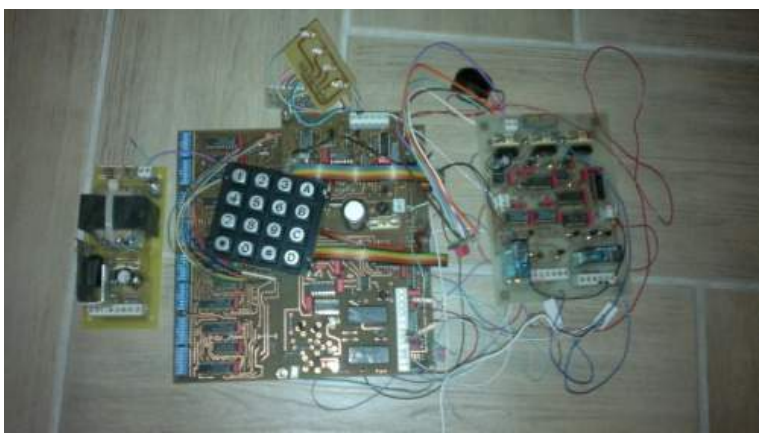


Figure II. 7 : *Les serrures à code sont des circuits électroniques fonctionnant dans le système binaire.*

Les trois principaux systèmes

Vous connaissez déjà le système binaire. Deux autres bases sont d'usage en informatique.

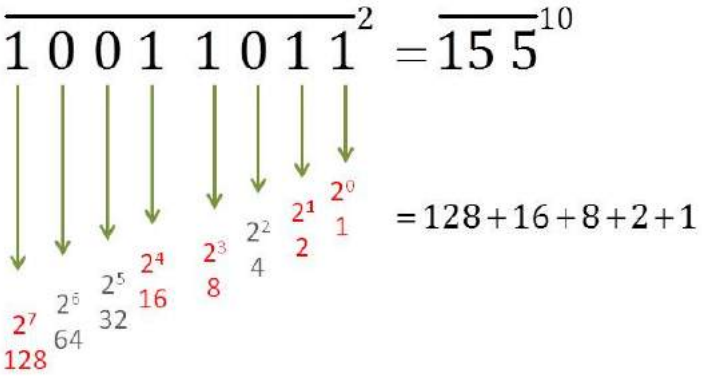
Le système octal n'est plus guère utilisé dans l'informatique et l'électronique. Cependant, on l'utilise quelquefois à la place du système hexadécimal pour ces deux avantages : il peut grouper des séries de trois bits, les rendant plus lisibles, et ne nécessite pas l'emploi de lettre pour les chiffres supérieurs à neuf.

Le système hexadécimal est fort utilisé par les informaticiens et les électroniciens. Cette base est en effet très commode pour les concepteurs de machines fonctionnant grâce au binaire. Elle permet de simplifier la lecture en regroupant les bits par paquet de quatre. De plus, il concorde avec l'octet correspondant à la plus petite unité de stockage adressable. Cet octet équivaut à deux valeurs hexadécimales.

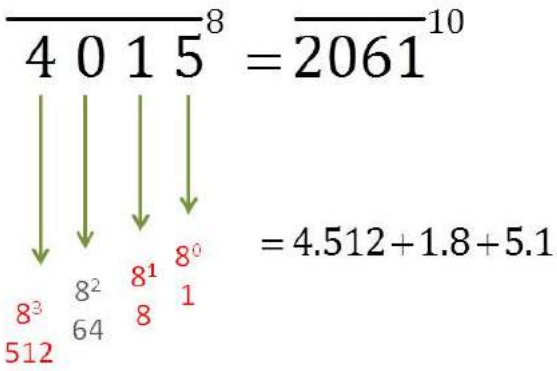
Voici une table présentant les 3 systèmes les plus courants en informatique et électronique ainsi que les chiffres qu'ils emploient pour coder les nombres.

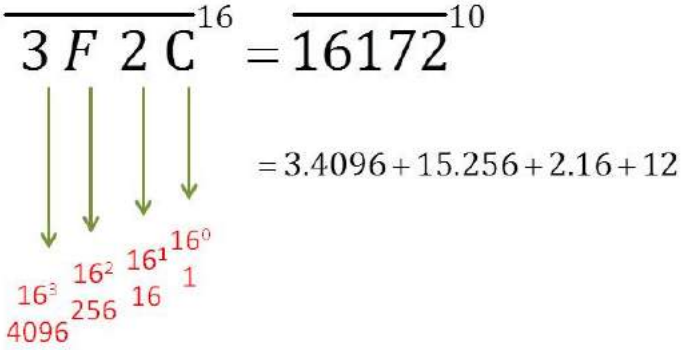
Système	Décimal	Binaire	Octal	Hexadécimal
Base	10	2	8	16
Chiffres	*0,1,2,3,4, 5,6,7,8,9*	*0,1*	*0,1,2,3, 4,5,6,7*	*0,1,2,3,4,5,6,7, 8,9,A,B,C,D,E,F*
	0	0000	0	0
	1	0001	1	1
	2	0010	2	2
	3	0011	3	3
	4	0100	4	4
	5	0101	5	5
	6	0110	6	6
	7	0111	7	7
	8	1000	10	8
	9	1001	11	9
	10	1010	12	A
	11	1011	13	B
	12	1100	14	C
	13	1101	15	D
	14	1110	16	E
	15	1111	17	F

Illustrations…



Anecdote : En 1968, un chanteur humoriste, Bobby Lapointe, inventa un système hexadécimal, le **système bibi-binaire** qui était à la fois drôle et cohérent.





Changement de base

Il est inutile de revenir sur la conversion du système décimal vers ces différentes bases et réciproquement. Toutefois, une observation dans le passage de la base binaire, vers la base octale ou hexadécimale est intéressante.

Voyons cela à l'aide d'un exemple.

Il est très simple de convertir un nombre du binaire à l'octal car 8 vaut 2 élevé à la puissance 3.

Nous avons déjà vu à la page 59 comment procéder.

Comme pour l'octal, il est très simple de passer du binaire à l'hexadécimal et vice versa. C'est la raison pour laquelle ce système est fort utilisé en électronique. La base 16 est une forme compacte de la base 2. En effet, la conversion de chaque quartet dans le système hexadécimal offre une représentation contractée, ce qui présente un gros avantage dans les systèmes à grande capacité de mémoire.

Pour simplifier le passage d'un chiffre du binaire en hexadécimal (pour un humain car un ordinateur ne sépare pas les chiffres), nous allons regrouper par quartet.

Illustrations…

$$\overline{1\,0\,0\,1\;\;1\,0\,1\,1}^{2} = \overline{9B}^{16}$$

9 B

Ou encore $\overline{000111010100}^{2} = \overline{0001\;1101\;0100}^{2} = \overline{1\;D\;4}^{16}$

Codage en BCD

En informatique, il arrive qu'on utilise une version différente du code binaire : le code binaire codé décimal ou BCD. Dans ce système, chaque chiffre décimal est codé individuellement en son équivalent en code binaire naturel sur 4 bits.

Pour passer du code BCD en code décimal, il suffit de regrouper les bits par quartets à partir de la droite et convertir ces paquets de 4 bits en un chiffre décimal.

Illustration…

$$\overline{2\;\;4\;\;6}^{10} = \overline{0010\;0100\;0110}^{BCD}$$

0010 0100 0110

Les nombres binaires négatifs

Vous pensez sûrement que pour transformer un nombre binaire positif en négatif, il faut simplement ajouter un signe '–' devant le nombre. Vous n'avez pas totalement tort. Oui, si vous écrivez un nombre en binaire à la main un simple signe '–' suffit. Mais pour un ordinateur, c'est plus compliqué car il ne sait que gérer les « 0 » et les « 1 » donc malheureusement un vulgaire signe « - » ne fonctionnera pas dans ce cas.

Les informaticiens ont trouvé une solution : le premier bit (le premier chiffre) va nous indiquer le signe du nombre. Si le premier chiffre est un « 0 », le nombre est positif et si le premier chiffre est un « 1 », le nombre est négatif.

Pour les nombres commençant par « 0 », on les lit naturellement.

Pour les nombres commençant par « 1 », on prend l'opposé du bit de poids fort (à l'extrémité gauche) et on ajoute la somme des bites restant sur la droite.

Nombres de 8 bits		**Valeur dans le système décimal**
Lu en hexadécimal	**Lu en binaire**	
7F	0111 1111	127
7E	0111 1110	126
…	…	…
10	0001 0000	16
0F	0000 1111	15
0E	0000 1110	14
0D	0000 1101	13
0C	0000 1100	12
0B	0000 1011	11
0A	0000 1010	10
09	0000 1001	9
08	0000 1000	8
…	…	…
03	0000 0011	3
02	0000 0010	2
01	0000 0001	1
00	0000 0000	0
FF	1111 1111	-1
FD	1111 1110	-2
GC	1111 1101	-3
FE	1111 1100	-4
…	…	…
83	1000 0011	-125
82	1000 0010	-126
81	1000 0001	-127
80	1000 0000	-128

Illustration… $\overline{1001\ 1010}^{2} = -128+16+8+2^{10} = -\overline{102}^{10}$

La soustraction dans le système binaire

Dans le chapitre suivant, nous vous montrerons comment effectuer les opérations usuelles dans un système de base *b*, notamment dans le système binaire. Le principe est le même que celui des opérations écrites que vous avez apprises à l'école primaire.

Il existe une méthode de soustraction que les machines emploient et qui consiste à effectuer une addition du premier nombre au complément à 2 du nombre à soustraire. Pour trouver le complément à 2 d'un nombre en binaire, il faut d'abord trouver son complément à 1 en remplaçant simplement les « 0 » par des « 1 » et les « 1 » par des « 0 ». Ensuite, il suffit d'ajouter 1 au nombre trouvé.
Cette méthode ne peut être utilisée que dans des cas où la taille des nombres est limitée.

Illustration…

Soit le nombre N $\overline{1101\ 1110}^{2}$

Voilà son complément à 1 $\overline{0010\ 0000}^{2}$

Et voici son complément à 2 $\overline{0010\ 0001}^{2}$

Si le nombre à soustraire possède un nombre de bits inférieur à celui du nombre auquel on soustrait, on ajoute des « 0 » à la gauche du nombre que l'on soustrait de sorte à égaler le nombre de chiffres composant les termes de la soustraction.
De plus, si le résultat obtenu possède plus de chiffres que le nombre auquel on soustrait, on ne prend pas en compte les chiffres en surplus depuis la gauche.

Illustration…

$$\overline{0100\ 0100}^{2} - \overline{10\ 1010}^{2} = ?$$

$$\overline{0100\ 0100}^{2} - \overline{0010\ 1010}^{2} = ?$$

$$\overline{1101\ 0101}^{2}$$

$$+1$$

$$\overline{1101\ 0110}^{2} \quad \textit{le complément à 2}$$

$$\overline{0100\ 0100}^{2} + \overline{1101\ 0110}^{2} = \overline{1\,0001\ 1010}^{2}$$

$$\overline{0100\ 0100}^{2} + \overline{1101\ 0110}^{2} = \overline{0001\ 1010}^{2}$$

Et la mémoire ?

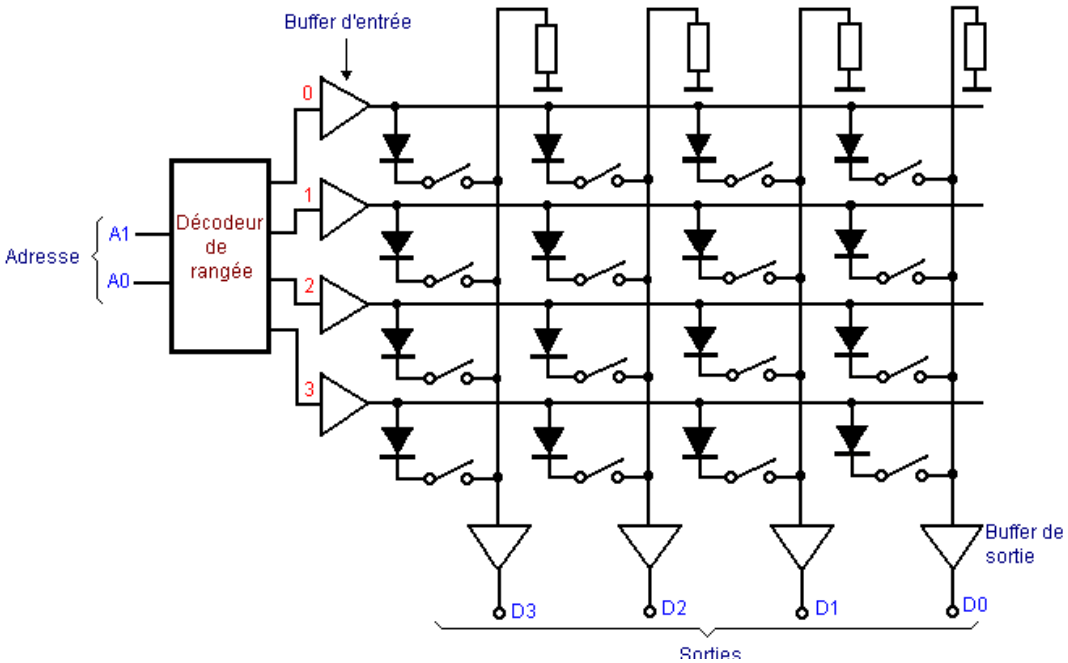


Figure II.8 : *Structure interne d'une mémoire morte à 16 bits disposés en quatre quartets.*

Le schéma ci-dessus montre la structure matricielle interne d'une mémoire à diodes (ROM pour Read Only Memory ou mémoire à lecture seule) de 16 bits (4x4). Les interrupteurs représentés correspondent à la présence d'une diode en fonctionnement s'ils sont fermés et de l'absence de diode s'ils sont ouverts.

Exemple : supposons, pour la première ligne que les interrupteurs de la 1ère et de la 3ème colonne soient fermés. Le principe est d'appliquer un niveau logique « 1 » (message « 1 ») successivement à chaque ligne (ligne 0, puis ligne 1….). Dès lors, dans l'exemple choisi, si nous appliquons un tel niveau logique (1) à la ligne 0, nous allons récupérer un niveau 1 aux sorties D1 et D3, et un niveau logique pour les sorties D0 et D2.

En pratique, aujourd'hui, on utilise dans les ordinateurs des mémoires de plusieurs millions voire milliards de bits. Si on remplace les diodes par des condensateurs (et qu'on adapte la logique de contrôle), nous obtenons alors une mémoire RAM (Random Access Memory pour mémoire à accès aléatoire).

Petite info supplémentaire : en informatique, dans un souci de simplicité, le K devant l'octet (Ko) ne signifie pas 1000 x 1 octet mais 1024 x 1 octet car 1024 est égal à 2^{10} et que 1024 est très proche de 1000 (on emploie un K majuscule à la place d'un k minuscule pour faire une différence entre 1000 et 1024). Le Mo est lui aussi composé de 1024 x 1024 x 1 octet. Cette particularité ne s'applique que dans ces 2 cas.