

REPRÉSENTATION DES NOMBRES ENTIERS

Dans un ordinateur, toute l'information est contenue dans des éléments de mémoire (les *bits*, pour *Binary digIT*) qui n'ont que deux états possibles : 0 et 1. Il faut donc représenter les nombres entiers uniquement avec deux symboles.

I. Écriture binaire des entiers naturels

1) Définition

Théorème : *Tout entier strictement positif n s'écrit de manière unique sous la forme*

$$n = \sum_{i=0}^k d_i 2^i$$

avec pour tout $i \in \llbracket 0, k \rrbracket$, $d_i \in \{0; 1\}$ et $d_k \neq 0$.

Les d_i sont les *chiffres binaires* de n , et en concaténant les chiffres $d_k d_{k-1} \dots d_1 d_0$, on obtient l'*écriture binaire* de n . Pour distinguer l'écriture binaire de l'écriture usuelle (décimale) des nombres, on note $\underline{d_k d_{k-1} \dots d_1 d_0}_2$.

Exemple

- $13 = 8 + 4 + 1 = 2^3 + 2^2 + 1 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = \underline{1101}_2$
- $\underline{11011100}_2 = 2^7 + 2^6 + 2^4 + 2^3 + 2^2 = 128 + 64 + 16 + 8 + 4 = 220$
- Le plus grand entier qui s'écrit avec n chiffres binaires est $2^n - 1$.
- L'écriture binaire de $2n$ s'obtient à partir de celle de n en rajoutant un zéro à droite. Inversement, celle du quotient de la division euclidienne de n par 2 s'obtient en supprimant son dernier chiffre d_0 .
- Un nombre est pair si et seulement si son chiffre de droite est 0, autrement dit le chiffre de droite est le reste de la division euclidienne du nombre par 2.

2) Algorithme d'écriture d'un entier en binaire

Il est fondé sur le principe suivant :

1. Le dernier chiffre d_0 est 0 si n est pair, 1 si n impair.
Autrement dit d_0 est le reste de la division euclidienne de n par 2.
2. Les autres chiffres, $d_1, d_2 \dots$ sont les chiffres binaires du quotient de n par 2.

Algorithme 1 : Écriture en binaire

Données : Un entier naturel $n > 0$

Résultat : les chiffres $d_0, d_1, d_2 \dots$ de l'écriture binaire de n

```

1  $k = 0$  ;                               /* Indice des chiffres  $d_0, d_1, \dots$  */
2 tant que  $n \neq 0$  faire
3    $d_k \leftarrow$  reste de la division euclidienne de  $n$  par 2;
4    $n \leftarrow$  quotient de  $n$  par 2;
5    $k \leftarrow k + 1$ 
6 fin
```

Faisons tourner l'algorithme avec $n = 13$. Les valeurs des différentes variables sont prises à la fin des lignes 1 et 5.

ligne	n	k	d_0	d_1	d_2	d_3
1	13	0				
5 (1 ^{er} passage)	6	1	1			
5 (2 ^e passage)	3	2	1	0		
5 (3 ^e passage)	1	3	1	0	1	
5 (4 ^e passage)	0	4	1	0	1	1

On obtient bien : $\underline{13}_{10} = \underline{1101}_2$

II. Les entiers des microprocesseurs

Les « entiers » utilisés par les microprocesseurs pour effectuer toutes les opérations arithmétiques (et les adressages) sont stockés dans ce que l'on appelle des *registres*, qui ont un nombre fixé de bits.

Lorsqu'on parle de processeurs « 8 bits », ou « 32 bits », ou « 64 bits », on se réfère à cette taille des registres.

Cette limitation de la taille des registres implique que le nombre d'entiers représentables par le processeur est fini : ce nombre est, au maximum, 2^n pour un processeur n -bits.

Pour les besoins de programmation, on a défini essentiellement deux conventions pour le codage des entiers : le codage en binaire pour les entiers entiers non signés et le codage en *complément à deux* pour les entiers signés.

1) Les entiers non signés

L'entier n , s'il n'est pas trop grand, est représenté par son écriture binaire. Sur un ordinateur 32 bits, on peut ainsi représenter tous les entiers de 0 à $2^{32} - 1 = 4\,294\,967\,295 = \underline{111\dots111}_2$.

Exemple

Sur un ordinateur 32 bits :

- 0 est représenté par $\underline{000\dots000}_2$ (32 chiffres 0)
- $2^{31} - 1$ est représenté par $\underline{011\dots111}_2$ (31 chiffres 1)
- 2^{31} est représenté par $\underline{100\dots000}_2$ (31 chiffres 0)
- $2^{32} - 1$ est représenté par $\underline{111\dots111}_2$ (32 chiffres 1)

2) Les entiers signés : la représentation en complément à 2

Définition : Pour un nombre de bits b , la représentation en « complément à deux » d'un entier n satisfaisant $-2^{b-1} \leq n \leq 2^{b-1} - 1$ est :

- si $n \geq 0$: sa représentation est simplement son écriture binaire
- si $n < 0$: le premier bit (poids fort) est 1, les $b - 1$ autres représentent $2^{b-1} + n$.

Exemple

sur $b = 4$ bits, les nombres représentés sont les entiers de -8 à 7 , et leurs codages sont :

nombre	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
codage	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

Remarques :

- Le premier bit donne le signe : s'il vaut 1, l'entier représenté est négatif ; s'il vaut 0, l'entier représenté est positif.
- Si n est négatif, le nombre binaire formé par les chiffres du codage de n (le premier bit, qui est 1 et qui vaut 2^{b-1} plus les $b - 1$ autres bits, qui codent $2^{b-1} + n$) est $2^{b-1} + 2^{b-1} + n$, c'est-à-dire $2^b + n$. Par exemple, sur $b = 4$ bits, -5 est codé par 1011, qui en tant que nombre binaire, vaut $\underline{1011}_2 = 11 = 2^4 - 5$.
- Si on appelle $c(n)$ le nombre binaire qui code n , on a $c(n) \equiv n \pmod{2^b}$.
- L'addition de deux nombres signés utilise exactement le même algorithme que l'addition de deux nombres non signés, par exemple (sur 4 bits), l'opération :

$$\begin{array}{rcccc}
 & & 1 & & 1 \\
 & & 1 & 0 & 1 & 1 \\
 + & 0 & 0 & 1 & 1 & \\
 \hline
 & 1 & 1 & 1 & 0 &
 \end{array}$$

s'interprète, en non signé, par $11 + 3 = 14$, et en signé par : $-5 + 3 = -2$

3) Débordements (« overflow »)

Lorsqu'une opération sur deux nombres entiers donne un nombre qui n'est plus représentable, le résultat obtenu n'est pas exact, ce phénomène est appelé *débordement* ou *dépassement de capacité*.

Exemple

- sur 4 bits en non signé : l'opération $13 + 4$ donne lieu à un débordement.
- En signé, l'opération $-3 + 4$ (noter que pour cette opération, le processeur effectue exactement le même calcul que pour $13 + 4$ en non signé) ne donne pas lieu à un débordement.

Le processeur avertit qu'il y a eu débordement (en signé ou en non signé) à l'aide de *drapeaux*, qui sont des bits d'un registre particulier.