

Écriture d'un entier positif:

1. Le système décimal:

Pour que vous compreniez le fonctionnement du binaire, et des systèmes de comptage en général (plus communément appelés bases), je vais commencer par faire une petite réintroduction à la base 10 que vous connaissez tous.

En effet, tout le monde sait compter en base 10 (décimal). Mais comment ça marche ? Comment est construit notre système ? Pour répondre à cette question à l'apparence simple, oubliez tout et reprenons depuis le début : comment avez-vous appris à compter à l'école ?

Vous penserez peut-être que la base 10 vient du fait qu'on a 10 doigts, mais en tout cas deux choses sont sûres :

- Il y a 10 chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- Avec ces derniers, on peut compter jusqu'à 9.

Et si l'on veut aller au-delà de 9, il faut changer de rang.

Cela signifie que si le rang des unités est plein, il faut passer à celui des dizaines, puis des centaines, milliers et j'en passe.

Par exemple : à 19, le rang des unités est "saturé" (plein), car il contient le chiffre 9, et il n'y a pas (dans la base 10) de valeur plus élevée. Il faut donc incrémenter le rang périphérique puis réinitialiser l'état de celui des unités. Ce qui signifie : j'ai 19, je ne peux pas mettre plus de 9 à droite, donc j'ajoute 1 à celui de gauche et je mets à zéro celui de droite.



Comme je disais tout à l'heure, le nombre entier va être composé de rangs (unités, dizaines, centaines, etc). Chaque rang vaut le rang précédent multiplié par l'indice de la base. Une centaine vaut dix dizaines, et une dizaine vaut 10 unités. Par exemple, dans l'image ci-dessus, on peut voir le nombre 185_{10} (ici, le 10 signifie qu'il s'agit d'un nombre, en base 10). Dans ce nombre, on peut voir trois rangs : centaines, dizaines et unités. Pour n'importe quelle base, la valeur d'un rang est égale à b_n , où b est l'indice de la base (ici, 10) et n la position du rang. Ici, les unités ont la position 0, les dizaines la position 1 et les centaines la position 2. Nous pouvons donc écrire :

$$185 = 1 \cdot 10^2 + 8 \cdot 10^1 + 5 \cdot 10^0$$

Ce que je viens de faire, c'est décomposer 185 en puissance de 10 (unités, dizaines, centaines, etc).

Un nombre est égal à la somme des valeurs de ses rangs, et on peut décomposer n'importe quel nombre en puissance de sa base.

2. Le binaire:

Le binaire, c'est le système de comptage des ordinateurs. Pourquoi le binaire et pas le décimal comme les humains ? Et bien c'est très simple : un ordinateur est composé de circuits électroniques, et donc de composants électriques. Le plus simple pour compter est donc d'utiliser un système en base 2 (le binaire) car on peut représenter ses deux valeurs possibles (0 et 1) par un signal électrique : 1, y'a du courant, 0, y'en a pas (c'est la version simple).

Je vous ai parlé ci-dessus de rangs. En binaire, c'est pareil à la différence qu'on utilise le terme bit, qui est la contraction de "binary digit", littéralement "chiffre binaire". Par exemple, le nombre 10011 occupe 5 bits. Là où tout se complique, c'est que comme je l'ai expliqué, chaque rang en binaire ne peut avoir que deux valeurs (binaire = base 2) différentes : 0 ou 1. Pour la base 10, chaque rang représente une puissance de 10, pour la base 2, chaque rang occupe une puissance de 2. Voici comment compter en binaire jusqu'à 10 :

| Nombre en décimal | Nombre en binaire | Le pourquoi du comment |
|-------------------|-------------------|---|
| 0 | 0 | Pour l'instant, ça va. |
| 1 | 1 | Là encore, c'est simple. |
| 2 | 10 | Le premier rang ayant été rempli, on passe au suivant ! |
| 3 | 11 | On re-remplit le rang 1. |
| 4 | 100 | Le rang 2 est plein, le rang 1 aussi, qu'à cela ne tienne, on passe au suivant. |
| 5 | 101 | On continue en suivant la même méthode. |
| 6 | 110 | |
| 7 | 111 | |
| 8 | 1000 | On commence le rang 4. |
| 9 | 1001 | On continue comme tout à l'heure. |
| 10 | 1010 | |
| ... | | |

Retenez juste ceci : entamer le rang suivant quand l'actuel est plein.

Exercice 1:

- Combien de valeurs peut-on coder avec 1 bit ?
- Combien de valeurs peut-on coder avec 2 bit ?
- Combien de valeurs peut-on coder avec 3 bit ?
- Combien de valeurs peut-on coder avec n bit ?

Comme on a pu le voir, compter jusqu'à 10 ou 20 reste aisé, mais imaginons un instant que je vous demandasse d'écrire 185 en binaire ? Vous allez faire chaque rang, un par un ?

3. Conversion décimale binaire:

Pour l'instant, on n'a compté que jusqu'à 10. Mais pour les plus grands nombres, la méthode précédente peut se révéler fastidieuse. Il existe bien sûr plusieurs méthodes de conversion, mais nous allons étudier la plus simple et la plus rapide. Il s'agit de la méthode euclidienne. Cette méthode, en plus d'être facile à utiliser en programmation (c'est un algorithme) est une des

meilleures lorsqu'il s'agit de traiter les grands nombres.

Voici la méthode :

- On prend le nombre en base 10 (forme normale).
- On le divise par 2 et on note le reste de la division (soit 1 soit 0).
- On refait la même chose avec le quotient précédent, et on met de nouveau le reste de côté.
- On réitère la division, jusqu'à ce que le quotient soit 0.
- Le nombre en binaire apparaît alors : il suffit de prendre tous les restes de bas en haut.

Et l'exemple (le '+' doit être lu 'il reste'):

$$\begin{aligned}185 \div 2 &= 92 + 1 \\92 \div 2 &= 46 + 0 \\46 \div 2 &= 23 + 0 \\23 \div 2 &= 11 + 1 \\11 \div 2 &= 5 + 1 \\5 \div 2 &= 2 + 1 \\2 \div 2 &= 1 + 0 \\1 \div 2 &= 0 + 1\end{aligned}$$

Attention, il faut bien lire de bas en haut !

185 en base 10 vaut donc 10111001 en binaire.

Exercice 2: Convertir 42_{10} en base 2.

4. Conversion binaire décimale:

Je vous rassure tout de suite : c'est plus simple dans ce sens-là que dans l'autre.

Prenons un nombre au hasard, tel que 11010011. Il s'étale sur 8 rangs, et comme dit précédemment, chaque rang correspond à une puissance de deux. Le premier rang (en partant de la droite) est le rang 0, le second est le 1, etc.

Pour convertir le tout en décimale, on procède de la manière suivante : on multiplie par 20 la valeur du rang 0, par 2^1 la valeur du rang 1, par 2^2 la valeur du rang 2, [...], par 2^{10} la valeur du rang 10, etc.

Après ça, il ne reste plus qu'à remplacer les puissances de 2 par leurs valeurs et de calculer la somme :

$$11010011_2 = 1.2^7 + 1.2^6 + 0.2^5 + 1.2^4 + 0.2^3 + 0.2^2 + 1.2^1 + 1.2^0 = 211_{10}$$

Exercice 3: convertir 10011001_2 en base 10

5. L'hexadécimal:

Le binaire, c'est bien pratique : on peut coder des nombres uniquement avec des 0 et des 1. C'est

bien pour les signaux électriques et tout le bazar, mais dans la vie de tous les jours c'est pas bien facile d'utilisation. On utilise couramment la base 10. Le problème c'est qu'en informatique, tout est basé sur le binaire, et étant une base d'indice 2, c'est plus aisé d'encoder les informations sur un nombre multiple de 2. On utilise donc souvent la base 16, appelé système hexadécimal (hexa = 6, déci = 10, $16 = 6 + 10$) car 16 est un multiple de 2, et qu'il permet de représenter 8 bits avec seulement 2 chiffres. Ça paraît simple, mais il y a un autre problème : en base 10, on utilise 10 chiffres. En base 2 (binaire) on utilise seulement 2 chiffres : 0 et 1. Mais du coup, en base 16, il faut 16 chiffres. 0 1 2 3 4 5 6 7 8 9.. quoi après ? On prend des lettres de l'alphabet.

Ce qui donne :

0 1 2 3 4 5 6 7 8 9 A B C D E F On peut établir une liste de correspondances entre la base 10 et la base 16 (voire même la base 2) :

| Binaire (base 2) | Décimal (base 10) | Hexadécimal (base 16) |
|------------------|-------------------|-----------------------|
| 0 | 0 | 0 |
| 1 | 1 | 1 |
| 10 | 2 | 2 |
| 11 | 3 | 3 |
| 100 | 4 | 4 |
| 101 | 5 | 5 |
| 110 | 6 | 6 |
| 111 | 7 | 7 |
| 1000 | 8 | 8 |
| 1001 | 9 | 9 |
| 1010 | 10 | A |
| 1011 | 11 | B |
| 1100 | 12 | C |
| 1101 | 13 | D |
| 1110 | 14 | E |
| 1111 | 15 | F |

Comme vous pouvez le voir, le plus grand chiffre en hexadécimal est F, et il correspond à 15 en décimal et 1111 en binaire : F est donc encodé sur 4 bits ($F_{16} = 1111_2$, 4 chiffres binaires = 4 bits).

Pour convertir un nombre binaire en base 16, on regroupe les bits 4 à 4, chaque groupe donnant un chiffre hexadécimal. À l'inverse, passer d'un nombre hexadécimal à sa représentation binaire se fait en remplaçant chaque chiffre pour son équivalent sur 4 bits. Ainsi, $11011001_2 = D9_{16}$, tandis que $7F_{16} = 01111111_2$. Attention, si le nombre binaire de départ n'a pas un nombre de bits multiple de 4, il faut ajouter des zéros en tête (ce qui ne change pas sa valeur) afin de pouvoir les regrouper 4 par 4.

Exercice 4:

- Convertir 101111011001_2 en base 16
- Convertir 111011_2 en base 16
- Convertir $1A3E_{16}$ en base 2

6. Convertir un nombre décimal en hexadécimal:

Pour convertir un nombre décimal en hexadécimal, la méthode est similaire au binaire, sauf que cette fois on divise par 16.

$$\begin{aligned} 185 \div 16 &= 11 + 9 \quad 9 \\ 11 \div 16 &= 0 + 11 \quad B \end{aligned}$$

Attention, il faut bien lire de bas en haut !

185 en base 10 vaut donc B9 en hexadécimal.

Exercice 5: Convertir 1387_{10} en base 16

7. Convertir un nombre décimal en hexadécimal:

Le principe est le même que pour la conversion "binaire en décimal" sauf qu'au lieu d'utiliser des 2^n on utilise des 16^n .

exemple :

$$12B7_{16} = 1 \cdot 16^3 + 2 \cdot 16^2 + 11 \cdot 16^1 + 7 \cdot 16^0 = 1 \cdot 4096 + 2 \cdot 256 + 11 \cdot 16 + 7 = 4096 + 512 + 176 + 7 = 4791$$

Vous avez remarqué que quand on trouve un B dans le nombre écrit en hexadécimal, on le remplace par un 11 dans le calcul. C'est exactement la même chose quand on trouve :

- un A, on le remplace par un 10,
- un C, on le remplace par un 12,
- un D, on le remplace par un 13,
- un E, on le remplace par un 14,
- un F, on le remplace par un 15,

Exercice 6: Convertir $1B87_{16}$ en base 10

8. Généralisation pour une base b:

Nous avons vu ci-dessus comment convertir un nombre en base 10 en un nombre en base 16. Pour convertir un nombre n (en base 10) en sa représentation en base b , il faut suivre l'algorithme suivant :

- On appelle D la liste des chiffres (le résultat).
- Tant que $n > 0$
 - Faire la division euclidienne de n par b ,
 - Ajouter le reste à D ,
 - Mettre le quotient dans n
- Inverser l'ordre de D .

Soit x la représentation du nombre en base 10, r la représentation du nombre en base b , r étant écrit en utilisant n chiffres : $c_0, c_1 \dots c_{n-1}$, nous avons :

$$\begin{aligned} r &= c_{n-1}c_{n-2} \dots c_2c_1c_0 \\ x &= \sum_{i=0}^{n-1} c_i \cdot b^i \end{aligned}$$