Pourquoi diviser par 2 donne la représentation binaire ?

Introduction

Pour convertir un nombre entier en base 10 (décimale) vers la base 2 (binaire), on utilise souvent la méthode suivante :

> **On divise le nombre par 2 de manière répétée et on garde les restes. Les bits obtenus sont les chiffres de la représentation binaire, en les lisant de bas en haut.**

Mais pourquoi cette méthode fonctionne-t-elle ? C'est ce que nous allons expliquer.

Exemple

Convertissons le nombre décimal 13 en binaire :

Division	Quotient	Reste
$13 \div 2$	6	1
$6 \div 2$	3	0
$3 \div 2$	1	1
$1 \div 2$	0	1

On lit les restes de bas en haut : 1 1 0 1.
Donc : $13_{10} = 1101_2$.

Explication intuitive

Le système binaire utilise uniquement les chiffres 0 et 1 pour représenter les nombres à l'aide des puissances de 2 :

$$n = a_k \cdot 2^k + a_{k-1} \cdot 2^{k-1} + \cdots + a_1 \cdot 2^1 + a_0 \cdot 2^0$$

où chaque $a_i \in \{0, 1\}$.
Chaque division par 2 nous dit :
— si le nombre est pair : le bit de droite (bit de poids faible) est 0 ;
— si le nombre est impair : ce bit est 1.
En divisant à chaque fois le quotient précédent, on « remonte » dans les puissances de 2, et on obtient tous les bits.

Justification formelle

Soit $n \in \mathbb{N}$ un entier naturel. Par le **théorème de division euclidienne**, on peut écrire :

$$n = 2q_0 + r_0 \quad \text{avec } r_0 \in \{0, 1\}$$

C'est exactement ce qu'on fait lors de la première division par 2 :
— q_0 est le quotient
— r_0 est le reste — c'est le bit le plus à droite.
On applique maintenant le même procédé à q_0 :

$$q_0 = 2q_1 + r_1 \Rightarrow n = 2(2q_1 + r_1) + r_0 = 4q_1 + 2r_1 + r_0$$

On continue ainsi :

$$q_1 = 2q_2 + r_2 \Rightarrow n = 8q_2 + 4r_2 + 2r_1 + r_0$$

En généralisant, on obtient :

$$n = r_0 \cdot 2^0 + r_1 \cdot 2^1 + r_2 \cdot 2^2 + \cdots + r_k \cdot 2^k$$

avec $r_i \in \{0, 1\}$ et $q_k = 0$ à la dernière étape.
Ce sont exactement les coefficients de la représentation binaire de n. En lisant les r_i dans l'ordre inverse, on reconstitue la représentation binaire.

Conclusion

La méthode de division par 2 et prise des restes fonctionne car elle extrait successivement les bits de la représentation binaire, en partant du bit de poids faible jusqu'au bit de poids fort. Elle repose sur le principe même de la numération en base 2, et sur la division euclidienne répétée.

Autre méthode : décomposition en puissances de 2

Une autre manière de convertir un nombre décimal en binaire consiste à :

1. Trouver la **plus grande puissance de 2** qui tient dans le nombre ;
2. Soustraire cette puissance, et répéter sur le reste ;
3. Mettre un **1** aux positions correspondantes, et des **0** ailleurs.

Exemple : Convertir 13 en binaire.
— La plus grande puissance de $2 \leq 13$ est $8 = 2^3 \Rightarrow$ on met un 1 à la position 2^3
— Reste : $13 - 8 = 5$
— La plus grande puissance de $2 \leq 5$ est $4 = 2^2 \Rightarrow$ on met un 1 à la position 2^2
— Reste : $5 - 4 = 1$
— La plus grande puissance de $2 \leq 1$ est $1 = 2^0 \Rightarrow$ on met un 1 à la position 2^0
On n'a jamais utilisé $2^1 = 2$, donc on met un 0 à cette position.
Donc :

$$13 = 8 + 4 + 1 = 2^3 + 2^2 + 2^0 \quad \Rightarrow \quad 1101_2$$

Remarque : Cette méthode est équivalente à la précédente, mais elle construit la représentation en partant des **bits les plus à gauche** (poids fort), au lieu de commencer par les bits de poids faible.