

ÉCRIRE UN ENTIER POSITIF EN BINAIRE

1) Compter en binaire

Historiquement, on doit la naissance des premiers ordinateurs à l'association de trois idées géniales :

- 1) Créer une machine à calculer qui soit électrique et non mécanique
- 2) Calculer en binaire : 0 = pas de courant, 1 = courant (cf travaux de Leibnitz vers 1700)
- 3) Utiliser l'algèbre de Boole (cf travaux de Georges Boole vers 1850)

Les ordinateurs actuels étant toujours basé sur ces 3 idées, il nous faut donc notamment apprendre à compter en binaire !

décimal (base 10)	binaire (base 2)	hexadécimal (base 16)
0	0	0
1	1	1
2	10	2
3	11	3
4	<input type="text"/>	4
5	<input type="text"/>	5
6	<input type="text"/>	6
7	<input type="text"/>	7
8	<input type="text"/>	8
9	<input type="text"/>	9
10	<input type="text"/>	A
11	<input type="text"/>	B
12	<input type="text"/>	C
13	<input type="text"/>	D
14	<input type="text"/>	E
15	<input type="text"/>	F
16	<input type="text"/>	<input type="text"/>
17	<input type="text"/>	<input type="text"/>

Remarque :

Les deux chiffres binaires 0 et 1 sont appelés en anglais « binary digit » qui a été contracté en « bit ».

II) Conversions décimal \Leftrightarrow binaire

1) Premières puissances de 2 :

2^n	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
valeur	1	2	4	8				
binaire	1	10	100					

2) Décimal vers binaire :

a) *Méthode 1 : En décomposant le nombre décimal en somme de puissances de 2 :*

$$\begin{aligned}
 \text{Ex : } 173 &= 128 + 32 + 8 + 4 + 1 \\
 &= 2^7 + 2^5 + 2^3 + 2^2 + 2^0 \\
 &= \boxed{}
 \end{aligned}$$

b) *Méthode 2 : En effectuant des divisions entières par 2 :*

$$\text{Ex : } 173 = 2 \times 86 + \mathbf{1}$$

$$86 = 2 \times 43 + \mathbf{0}$$

$$43 = 2 \times 21 + \mathbf{1}$$

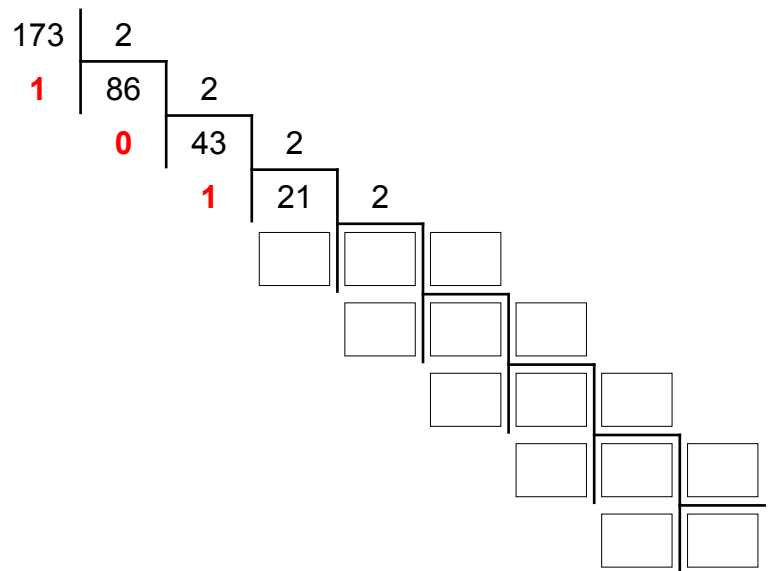
$$\boxed{} = 2 \times \boxed{} + \boxed{}$$

$$\boxed{} = 2 \times \boxed{} + \boxed{}$$

$$\boxed{} = 2 \times \boxed{} + \boxed{}$$

$$\boxed{} = 2 \times \boxed{} + \boxed{}$$

$$\boxed{} = 2 \times \boxed{} + \boxed{}$$



En lisant les restes de bas en haut, on retrouve le résultat : **10101101**₂

Explication :

Refaisons exactement les mêmes calculs que ci-dessus mais directement en base 2 :

$$10101101 = 10 \times 1010110 + \mathbf{1}$$

$$1010110 = 10 \times 101011 + \mathbf{0}$$

$$101011 = 10 \times 10101 + \mathbf{1}$$

$$10101 = 10 \times 1010 + \mathbf{1}$$

$$1010 = 10 \times 101 + \mathbf{0}$$

$$101 = 10 \times 10 + \mathbf{1}$$

$$10 = 10 \times 1 + \mathbf{0}$$

$$1 = 10 \times 0 + \mathbf{1}$$

3) Binaire vers décimal :

$$\begin{aligned}\text{Ex : } 1101001_2 &= 2^0 + \boxed{} \\ &= 1 + \boxed{} \\ &= \boxed{}\end{aligned}$$

III) Conversions binaire \Leftrightarrow hexadécimal

Puisque $1111_2 = F_{16}$, il suffit de regrouper les bits par groupes de 4 !

$$\text{Ex : } 0110\ 1001_2 = 69_{16}$$

$$7F_{16} = 0111\ 1111_2$$

IV) Conversions décimal \Leftrightarrow hexadécimal

On utilise le même principe que pour les conversions de décimal vers binaire sauf que l'on utilise les puissances de 16 et non celles de 2.

16^0	16^1	16^2	16^3
1	16	256	4096

$$\begin{aligned}\text{Ex : } 723 &= \boxed{} \times 16^2 + \boxed{} \times 16 + \boxed{} \\ &= \boxed{}\end{aligned}$$

$$\begin{array}{r|l} \text{Ex : } 723 & 16 \\ \hline \boxed{} & \boxed{} \boxed{} \\ \boxed{} & \boxed{} \boxed{} \boxed{} \\ \boxed{} & \boxed{} \boxed{} \end{array}$$

$$\begin{aligned}\text{Ex : } A1F_{16} &= \boxed{} \times 16^2 + \boxed{} \times 16 + \boxed{} \\ &= \boxed{}\end{aligned}$$

V) Opérations posées en binaire :

On peut faire des opérations « posées » en binaire en utilisant les méthodes apprises en primaire pour le système décimal :

Exemple d'addition :

$$\begin{array}{r} 1\ 1\ 0\ 1\ 1\ 0 \\ + \quad\quad 1\ 0\ 1\ 1 \\ \hline \square\square\square\square\square\square\square \end{array}$$

Exemple de multiplication

$$\begin{array}{r} 1\ 1\ 0\ 1\ 1\ 0 \\ \times \quad\quad 1\ 0\ 1\ 1 \\ \hline \square\square\square\square\square\square\square\square\square\square \\ \square\square\square\square\square\square\square\square\square\square \\ \square\square\square\square\square\square\square\square\square\square \\ \hline \square\square\square\square\square\square\square\square\square\square \end{array}$$

VI) Quels entiers positifs peut-on coder avec n bits ?

$1_2 = 1$ donc avec 1 bit, on peut coder les entiers positifs de 0 à

$11_2 = 3$ donc avec 2 bits, on peut coder les entiers positifs de 0 à

$111_2 = 7$ donc avec 3 bits, on peut coder les entiers positifs de 0 à

donc avec n bits, on peut coder les entiers positifs de 0 à

VII) Avec Python

>>> 0b11

>>> 0x1F

>>> bin(31)

>>> hex(31)