CODER UN ENTIER POSITIF EN BINAIRE

I) 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		4
5		5
6		6
7		7
8		8
9		9
10		A
11		B
12		C
13		D
14		E
15		F
16		
17		

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

II) Conversions décimal ⇔ 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 :

Ex : 173 = 128 + 32 + 8 + 4 + 1

= $2^7 + 2^5 + 2^3 + 2^2 + 2^0$

= $\square$

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

Ex : 173 = 2 × 86 + **1**

86 = 2 × 43 + **0**

43 = 2 × 21 + **1**

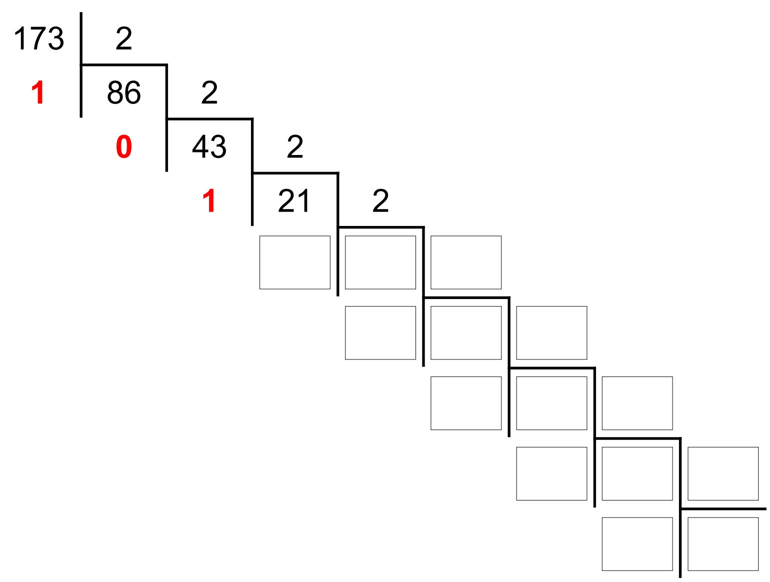
$\square = 2 \times \square + \square$

$\square = 2 \times \square + \square$

$\square = 2 \times \square + \square$

$\square = 2 \times \square + \square$

$\square = 2 \times \square + \square$



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 × 1010110 + **1**
1010110 = 10 × 101011 + **0**
101011 = 10 × 10101 + **1**
10101 = 10 × 1010 + **1**
1010 = 10 × 101 + **0**
101 = 10 × 10 + **1**
10 = 10 × 1 + **0**
1 = 10 × 0 + **1**

3) Binaire vers décimal :

Ex : $1101001_2 = 2^0 + \boxed{\quad\quad\quad\quad\quad\quad}$

$= 1 + \boxed{\quad\quad\quad\quad\quad\quad}$

$= \boxed{\quad\quad\quad\quad\quad\quad\quad}$

III) Conversions binaire ⇔ hexadécimal

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

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

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

IV) Conversions décimal ⇔ 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

Ex : $723 = \boxed{\quad} \times 16^2 + \boxed{\quad} \times 16 + \boxed{\quad}$

$= \boxed{\quad\quad\quad\quad\quad\quad\quad}$

Ex : 723 | 16

Ex : $A1F_{16} = \boxed{\quad} \times 16^2 + \boxed{\quad} \times 16 + \boxed{\quad}$

$= \boxed{\quad\quad\quad\quad\quad\quad\quad}$

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 :

	1	1	0	1	1	0
+			1	0	1	1

Exemple de multiplication

	1	1	0	1	1	0
×			1	0	1	1

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

$111_2 =$ 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)
'      '
```