1NSI

Codage des entiers positifs

Dans un ordinateur, toutes les informations (les données) sont représentées à l'aide de deux chiffres 0 et 1, appelés **chiffres binaires** (ou **bits**).

Dans la mémoire d'un ordinateur, ces chiffres sont regroupés en **octets** (c'est à dire par "paquets" de 8 ou bytes), puis organisés en **mots machine** de 2, 4 ou 8 octets (soit 16, 32 ou 64 bits).

Ce regroupement des bits en octets ou mots machine permet de représenter (et manipuler) d'autres données que des 0 et des 1, comme par exemple des nombres entiers, des nombres réels, des caractères...

On a inventé des encodages pour représenter ces informations, qui dépend de leur nature.

1 Repères historiques

1679 : *Leibniz* découvre et met au point **une arithmétique binaire** (Calculer avec des 0 et des 1). Il invente aussi en 1694 une machine à calculer capable de traiter les multiplications et divisions.

1728 : *Falcon* construit le premier métier à tisser utilisant les **cartes perforées** (présence ou absence d'un trou) pour fonctionner.

1854 : *Boole* publie un ouvrage dans lequel il démontre que tout processus logique peut être décomposé en une suite d'opérations logiques (ET, OU, NON) appliquées sur deux états (ZERO-UN, OUI-NON, VRAI-FAUX, OUVERT-FERME...).

1884 : *Herman Hollerith* crée une tabulatrice à cartes perforées (inspirée des métiers à tisser de Jacquard) pour réaliser le recensement Américain de 1890. Il s'agit de la **première machine à traiter l'information**.

1924 : La firme créée par Herman Hollerith en 1896, Tabulating Machine Corporation, est renommée en International Business Machine ou **IBM** (Une des premières entreprises à concevoir et à commercialiser des matériels informatiques).

1938 : Thèse de *Shannon* qui le premier fait le parallèle entre les circuits électriques et l'algèbre Booléenne. Il définit le chiffre binaire : **bit** (BInary digiT).

2 Encodage des entiers naturels en base 10

Rappeler les valeurs des puissance suivantes :

$10^0 =$ ..... ; $10^1 =$ ..... ; $10^2 =$ ..... ; $10^3 =$ .....

Pour écrire un nombre entier en base 10, sous ***forme décimale****, on dispose de* 10 *symboles :*

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

En base 10 on compte en faisant des paquets de 10, des paquets de 100, des paquets de 1000 etc. On obtient ainsi l'écriture d'un nombre entier positif en base 10.

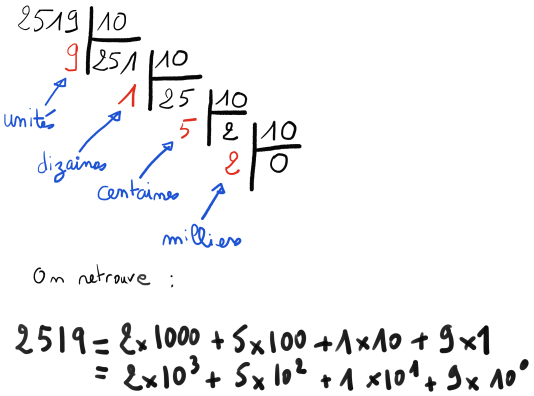
Exemple : On décompose le nombre 2 519 en une somme :

$$2\,519 = 2\,000 + 500 + 10 + 9$$

On en déduit une écriture du nombre 2 519 en fonction des puissances de 10 :

$2\,519 =$ .....

Remarque : On peut utiliser la division Euclidienne pour retrouver l'écriture en base 10 :



3 Encodage des entiers naturels en base 2

Pour écrire un entier naturel en base 2, en ***binaire****, on dispose de deux symboles :* 0 *et* 1.

De la même manière qu'en base 10 où l'on exprimait l'entier en une somme de puissance de 10, ici on va l'exprimer en une somme de puissances de 2.

Exemple :

$11 = 8 + 2 + 1 = 2^3 + 2^1 + 2^0 = 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^0$

On complète le tableau ci-dessous, en ajoutant des zéros si nécessaire :

Rang k	10	9	8	7	6	5	4	3	2	1	0
2^k	2048	1024	256	128	64	32	16	8	4	2	1
nombre de paquets	0	0	0	0	0	0	0	1	0	1	1

Ainsi en base 2, 11 s'écrit 1011.

Remarques :

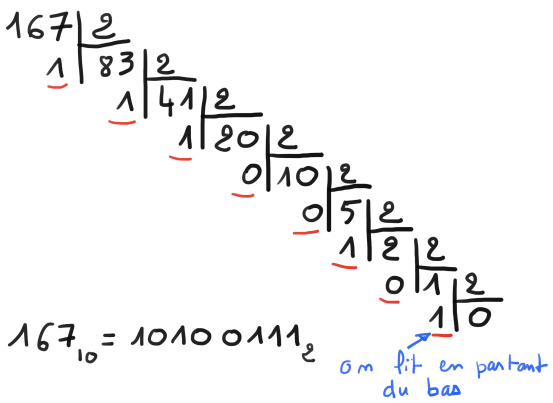
• Pour ne pas confondre le nombre "mille onze" et 1011 l'écriture binaire du nombre 11, on écrit la base en indice. Ainsi $11_{10} = 1011_2$.

• Ce nombre exprimé en base 2 sur 1 octet (8 bits) sera $11_{10} = 0000\ 1011_2$. *(On ajoute des zéros pour compléter l'écriture.)*

• Le **bit de poids faible** est le bit correspondant à la plus petite puissance (le plus à droite) ;
Le **bit de poids fort** est le bit correspondant à la plus grande puissance (le plus à gauche).

• La division Euclidienne permet de rapidement trouver l'écriture binaire d'un entier donné sous forme décimale *(jusqu'à obtenir un quotient nul).*

Exemple : Écrire 167 en base 2.



Exercice 1 : (Indiquer les calculs intermédiaires.)

1. Écrire les nombres suivants en base 2 (sous forme binaire) :

   14_{10} ; 57_{10} ; 411_{10} et 1023_{10}

2. Écrire les nombres suivants en base 10 (sous forme décimale).

   101_2 ; $11\ 001_2$; $110\ 1101_2$ et $1010\ 1010_2$

Préfixes Binaires :

En informatique on mesure, par exemple, la capacité mémoire d'un disque dur avec une unité de mesure exprimée comme un multiple d'octets.

Ces multiples sont souvent des puissances de 10 et on utilise des préfixes pour les nommer.

Préfixes décimaux :

Nom	Symbole	Valeur
kilooctet	Ko	10^3 octets
megaoctet	Mo	10^6 octets
gigaoctet	Go	10^9 octets
teraoctet	To	10^{12} octets

Mais on peut aussi utiliser des puissances de 2.

Préfixes binaires :

Nom	Symbole	Valeur
kibioctet	Kio	2^{10} octets
mebioctet	Mio	2^{20} octets
gibioctet	Gio	2^{30} octets
tebioctet	Tio	2^{40} octets

Valeurs maximales :

Suivant le nombre de bits utilisés pour coder un entier naturel, on est limité sur les valeurs possibles de cet entier.

Par exemple, si on code les nombres sur **un octet**, soit 8 bits, on peut coder $2^8 = 256$ entiers, de 0 à 255.

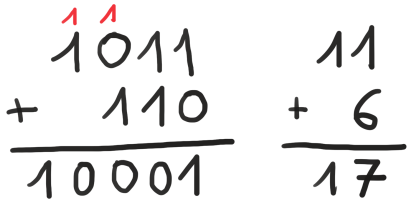
Si on code sur 2 octets, soit 16 bits, on peut coder $2^{16} = 65\ 536$ entiers, de 0 à 65 535.

Addition en binaire :

Règles d'addition :

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 1 = 10$$

Exemple : on additionne 1011_2 et 110_2 :



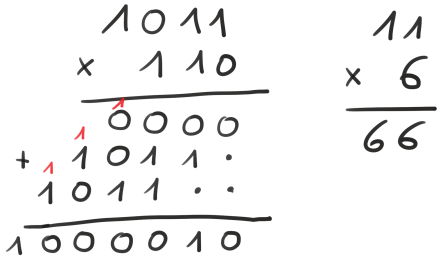
On a bien $1\ 0001_2 = 17_{10}$

Multiplication en binaire :

Règles de multiplication :

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 1 = 1$$

Exemple : on multiplie 1011_2 et 110_2 :



On a bien $100\ 0010_2 = 66_{10}$

Exercice 2 :

1. Effectuer les additions binaires suivantes :
$$A = 1001_2 + 11_2 \text{ et } B = 10111_2 + 1110_2$$
2. Effectuer les multiplications binaires suivantes :
$$C = 101_2 \times 10_2 \text{ et } D = 1101_2 \times 110_2$$

4 Encodage des entiers naturels en base 16

Pour écrire un entier naturel en base 16, en **hexadécimale**, on dispose de 16 symboles :
les chiffres de 0 à 9 auxquels on ajoute A, B, C, D, E et F.

Les symboles supplémentaires (par rapport à la base 10) correspondent aux valeurs suivantes :

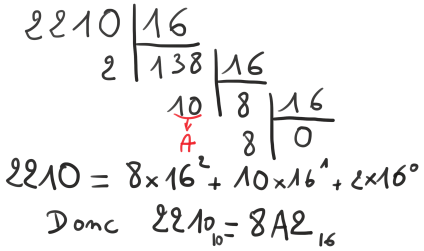
A	B	C	D	E	F
10	11	12	13	14	15

Les nombres codés en hexadécimale permettent de représenter de manière plus simple des nombres binaires.

On exprime l'entier en une somme de puissances de 16.

Exemples :

- $2019 = 7 \times 16^2 + 14 \times 16^1 + 3 \times 16^0$ donc $2019_{10} = 7E3_{16}$
- Écrire 2210_{10} en base 16 :



Exercice 3 :

1. Écrire les nombres suivants en base 16 (sous forme hexadécimale) :
$100\ 1010_2$; $1\ 0001\ 0001_2$ et $1010\ 0100\ 1111\ 0010_2$
(Pour cela on regroupe les chiffres binaires par 4 !)

2. Écrire les nombres suivants en base 2 (sous forme binaire).
   $BEEF_{16}$ et $C4D2_{16}$
3. Écrire les nombres suivants en base 16 (sous forme hexadécimale) :
   13_{10} ; 77_{10} et 10829_{10}
   (Il est plus facile de passer par la base 2 que de le faire directement.)
4. Écrire les nombres suivants en base 10 (sous forme décimale).
   $A5F3_{16}$ et 105_{16}

5 Encodage des entiers naturels en base b

Définition :

Une **base** d'un système de numération positionnel est un entier naturel b supérieur ou égal à 2.

Dire qu'un nombre s'écrit $a_n a_{n-1} a_{n-2} ... a_1 a_0$ **en base** b, signifie qu'il est égal à :

$$a_n \times b^n + a_{n-1} \times b^{n-1} + a_{n-2} \times b^{n-2} + ... + a_1 \times b^1 + a_0 \times b^0$$

où les entiers naturels a_i sont strictement inférieurs à b.

Exemple :

On écrit 47 en base 5 : $47_{10} = 1 \times 5^2 + 4 \times 5^1 + 2 \times 5^0 = 142_5$

6 Avec Python

Par défaut, en Python, les nombres entiers saisis ou affichés sont **en base** 10.

Pour signifier que le nombre est en base 2, on utilise la notation **0b....** où il suffit de remplacer les **....** par des 0 et des 1.

Inversement, on peut convertir un entier n en base 2 à l'aide de la fonction **bin()**.

(Cette fonction renvoie une chaîne de caractères.)

```
>>> 0b1110110001
945

>>> bin(513)
'0b1000000001'
```

Pour utiliser des nombres hexadécimaux on note **0x....** où **....** désigne n'importe quelle séquence de chiffres en base 16.

La fonction **hex()** permet de convertir un entier en base 16.

(Cette fonction renvoie une chaîne de caractères.)

```
>>> 0xAE5
2789

>>> hex(524)
'0x20c'
```

Exercice 4 :

Écrire une fonction **dectobin** qui convertit en binaire un entier naturel n donné en base 10.

Écrire une fonction **bintodec** qui prend pour argument un nombre en binaire et renvoie son écriture décimale.

Exercice 5 :

Écrire un programme qui lorsque l'on donne deux nombres entiers naturels écrits en base dix , les convertit en base deux, et affiche leur somme en binaire.

Correction :

exercice 1 :

1) $14 = 1110$; $57 = 11\ 1001$; $411 = 1\ 1001\ 1011$; $1023 = 11\ 1111\ 1111$

2) $101 = 5$; $11001 = 25$; $110\ 1101 = 109$; $1010\ 1010 = 170$

exercice 2 :

1) $A = 1\ 100$; $B = 100101$

2) $C = 1010$: $D = 100\ 1110$

exercice 3 :

1) $4A$; 111 ; $A4F2$

2) $1011\ 1110\ 1110\ 1111$; $1100\ 0100\ 1101\ 0010$

3) D ; $4D$; $2A4D$

4) 42483 ; 261