Binaire et représentation des entiers

1. L'information binaire

1.1. Le bit

L'histoire de l'informatique est intrinsèquement dépendante de l'histoire de la maîtrise de l'électricité. Fondamentalement, tout ordinateur est construit à partir de circuits électroniques qui:

- soit ne laissent pas passer le courant électrique (*Off*);
- soit laissent passer le courant électrique (*In*).

Exemple

Un circuit contenant un unique interrupteur admet donc **deux états**, il est **binaire** dans ce sens.

Définition : bit

Un bit est l'unité élémentaire d'information pouvant prendre deux valeurs distinctes, notées 0 et 1 (binaire). Le mot « **bit** » vient de l'anglais « **Bi**nary Digi**t** », soit littéralement *chiffre binaire*.

La notation internationale pour le bit est b. On parlera alors de kb, Mb, Gb.

1.2. Compter en binaire

Pour compter en binaire, on applique le même algorithme qu'en décimal : on ajoute 1 jusqu'à épuiser les chiffres disponibles, et quand on a terminé, on ajoute un chiffre supplémentaire au nombre. En, binaire, cela donne :

Nombre en binaire	Nombre en décimal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
...	...

1.3. Grouper les bits

Une information binaire est donc une suite de 0 et de 1. Cette information peut-être de différente nature, tout dépend de la **norme d'encodage** utilisée. En soit, la même suite binaire peut signifier des choses totalement différentes comme :

- un nombre entier ;
- un nombre flottant ;
- un caractère ;
- une note de musique ;
- ...

Exemple

L'écriture binaire 101010 signifie :

- 42 dans les nombres entiers non signés (c'est-à-dire positifs) ;

```
>>> int('101010',2)
42
```

- Le caractère * en UTF-8 :

```
>>> chr(int('101010',2))
'*'
```

Bits et quantités d'informations

Avec un unique bit, on ne peux stocker que deux informations (deux nombres, deux caractères, deux notes de musiques,...). La quantité d'information différentes pouvant être représentées dépend donc du nombre de bits utilisés :

Avec 2 bits :

Done

Avec 3 bits :

Done

Avec 4 bits :

Done

Avec un système à n bits, on peut représenter 2^n informations différentes.

1.4. Octets

Octets

Pour quantifier les informations binaires, on utilise souvent le mot **octet** (abusivement appelé aussi *byte* dans le monde anglo-saxon).

Un octet est un groupement de 8 bits. Il permet de représenter $2^8 = 256$ informations différentes.

La notation internationale pour l'octet est o. On parlera alors de ko ($1\ ko = 1000o$), Mo ($1\ Mo = 10^3\ ko = 10^6\ o$), Go ($1\ Go = 10^3\ Mo = 10^9\ o$), etc, mais aussi de ko/s, Mo/s, etc...

Remarque

Les préfixes *kilo*, *Mega*, *Giga* ..., sont bien ceux du système international, c'est-à-dire ceux pour 10^3, 10^6, 10^9... On verra dans la partie suivante qu'ils sont parfois confondus avec les préfixes binaires (*kibi*, *Mibi*, *Gibi*...), pour lesquels le facteur de changement d'unité n'est pas 10^3 mais $2^10 = 1024$. Ainsi $1Mibi = 1024kibi$.

2. Ecritures en d'autres bases

2.1. Les entiers en base décimale

Rappels : Base décimale

Un nombre entier écrit dans une base décimale (base 10) vérifie les conditions suivantes :

- il est écrit avec les dix chiffres arabes : $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$;
- chaque chiffre possède un poids, représentant une puissance de 10, le poids augmentant de la droite vers la gauche en partant d'un exposant 0.

Exercice

Enoncé

1. $14\ 763 = 1 \times 10^{\cdots} + 4 \times 10^{\cdots} + 7 \times 10^{\cdots} + 6 \times 10^{\cdots} + 3 \times 10^{\cdots}$
2. $100\ 042 = 1 \times 10^{\cdots} + 4 \times 10^{\cdots} + 2 \times 10^{\cdots}$

Solution

1. $14\ 763 = 1 \times 10^4 + 4 \times 10^3 + 7 \times 10^2 + 6 \times 10^1 + 3 \times 10^0$
2. $100\ 042 = 1 \times 10^{\cdots} + 4 \times 10^{\cdots} + 2 \times 10^{\cdots}$

2.2. La base 2 (système binaire)

Système binaire

Un nombre entier écrit dans le système binaire vérifie les conditions suivantes :

- il est écrit avec les deux chiffres : 0 et 1.
- chaque chiffre possède un poids représentant une puissance de 2, le poids augmentant de la droite vers la gauche en partant d'un exposant 0.

Remarques et notations

L'écriture 101 possède aussi bien un sens en binaire (un-zéro-un) qu'en décimal (cent-un). Pour lever l'ambiguïté , on écrira :

- $(101)_{10}$ ou simplement 101 pour le nombre en base 10 ;
- $(101)_2$ pour le nombre en base 2.

Exercice

Enoncé

Considérons le nombre $(101010)_2$. Convertissez ce nombre binaire en décimal :

Solution

$$\begin{array}{rclllllll} (101010)_2 & = & 1\times 2^5 & +0\times 2^4 & +1\times 2^3 & +0\times 2^2 & +1\times 2^1 & +0\times 2^0 \\ & = & 1\times 32 & +0\times 16 & +1\times 8 & +0\times 4 & +1\times 2 & +0\times 1 \\ & = & 42 \end{array}$$

Exercice : Conversions de la base 2 vers la base 10

Enoncé

Ecrire les nombres suivants en base 10 :

1. $(101)_2$
2. $(11111111)_2$
3. $(10010011)_2$

Solution

A venir !

Exercice : Un peu de Python

Enoncé

Compléter la fonction suivante, **sans utiliser** `int(x,2)`, afin qu'elle renvoie en base 10 le nombre x passé en argument en base 2, sous la forme d'une chaine de caractères. *Vous pouvez cependant utiliser la fonction built-in* `int` *afin de convertir une chaine de caratère en un entier.*

```
def bin2dec(x) :
    """fonction convertissant le nombre (x)_2 en base 10
>>> bin2dec('0')
0
>>> bin2dec('1')
1
>>> bin2dec('11')
3
>>> bin2dec('1000')
8
>>> bin2dec('11111111')
255
    """
```

2.3. Conversions de la base 10 vers la base 2

Méthode : Algorithme de conversion du nombre $(n)_{10}$ en base 2

En langage naturel

```
fonction dec2bin(n) :
    base2 <- chaine de caractère vide
    Tant que n!=0 :
        base2 <- caractere(n%2)+base2
        n <- n//2
    Renvoyer base2
```

Sous forme de diagramme

```
flowchart TB
A(Nombre n positif en décimal) --> B(base2 chaine de caractère vide)
B --> C{n != 0}
C --> |Non| F(base2 est la représentation de n en binaire)
subgraph Z["Boucle principale"]
    direction LR
    C -->|oui| D[Ajouter n%2 au début de base2];
    D --> E[n prend la valeur n//2]
    E --> C
end
```

Exemple : Conversion de $(135)_{10}$ en base 2

A venir !

Conversions manuelles

Enoncé

Convertir les nombres suivants en base 2 :

1. $(26)_{10}$
2. $(104)_{10}$
3. $(256)_{10}$
4. $(42)_{10}$

Solution

A venir !

Exercice : De l'algorithme au programme

Enoncé

Compléter la fonction suivante en Python, qui renvoie la chaine de caractères correspondant à l'écriture de n en base 2.

```
def dec2bin(n) :
"""fonction convertissant le nombre n en base 2,
et renvoyant la chaine de caractères correspondante
>>> dec2bin(0)
'0'
>>> dec2bin(1)
'1'
>>> dec2bin(3)
'11'
>>> dec2bin(8)
'1000'
>>> dec2bin(255)
'11111111'
"""
```

Solution

A venir !

Nombre de bits nécessaires

Soit $(x)_{10}$ un nombre écrit en base 10.

Si n est l'entier tel que $2^{n-1} \leqslant x < 2^n$, alors le nombre x nécessitera n bits pour être représenté en binaire.

Exercice : Nombre de bits nécessaires

Enoncé

Déterminer pour chacun des nombres ci-dessous le nombre de bits minimal qui seront nécessaire dans l'écriture binaire de ce nombre :

1. 12
2. 123
3. 999
4. 1927

Solution

A venir !

2.4. Une autre base utile : l'hexadécimal

Outre que la lecture des nombres en écriture binaire par un humain est très compliquée, il faut remarquer que, de par la construction de ces nombres, la quantité de symboles utilisés en base 2 est largement supérieur à celui utilisé en base 10 - $3,2$ **fois plus grand** en moyenne sur les $100\ 000$ premiers entiers.

Il peut donc être utile de trouver un compromis entre la base 2, utile pour l'ordinateur, et la base 10, plus compréhensible par un être humain.

Ce compromis peut-être trouvé avec le système **hexadécimal**, c'est-à-dire un système de **base 16**.

Clé WIFI

Typiquement, une clé de sécurité WIFI est un nombre binaire à 128 bits (ou 256 bits). Lorsque vous vous connectez à un nouveau réseau WIFI, vous devez taper la clé sur votre ordinateur/smartphone (on élimine ici la possibilité d'un FlashCode), mais celle ci est présentée souvent sous la forme `38326f53685c2c256164712838`, déjà particulièrement pénible à taper, et les erreurs de recopiages sont faciles à faire et difficiles à retrouver.

Ce nombre, sous sa forme binaire est celui-ci :

`11100000110010011011110101001101101000010111000010110000100101011000010110010001110001001010000011100 0`

De quoi faire encore plus d'erreurs en le recopiant...

Base hexadécimale

Un nombre entier écrit dans une base hexadécimale (base 16) vérifie les conditions suivantes :

- il est écrit avec les seize chiffres hexadécimaux : $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$;
- chaque chiffre possède un poids représentant une puissance de 16, le poids augmentant de la droite vers la gauche en partant d'un exposant 0.

Exemple

Le nombre hexadécimal $(5B6)_{16}$ correspond donc en décimal à :

A compléter

$$\begin{array}{rcccc} (5B6)_{16} & = & \dots \times 16^{\dots} & + \dots \times 16^{\dots} & + \dots \times 16^{\dots} \\ & = & \dots \times \dots & + \dots \times \dots & + \dots \times \dots \\ & = & & & \end{array}$$

Solution

$$\begin{array}{rcccc} (5B6)_{16} & = & 5 \times 16^2 & +11 \times 16^1 & +6 \times 16^0 \\ & = & 5 \times 256 & +11 \times 16 & +6 \times 1 \\ & = & 1\ 462 & & \end{array}$$

Exercice

Enoncé

Convertir de l'hexadécimal vers le décimal :

- $(FF)_{16}$
- $(6E)_{16}$
- $(245A)_{16}$

Méthode : Convertir vers l'hexadécimal depuis le décimal

Pour convertir vers l'hexadécimal depuis le décimal, on utilise la même méthode qu'en binaire mais en divisant par 16.

Exemple

Enoncé

Convertir le nombre 244 en hexadécimal.

Solution

A venir !

Remarques

- A un chiffre dans la base 16, correspond exactement 4 chiffres dans la base 2.
- Cela signifie que pour écrire un octet en hexadécimal, **deux chiffres hexadécimaux suffisent**. C'est pour cette raison que les octets écrits en base deux sont **groupés par 4 chiffres** :

$$(189)_{10} = (BD)_{16} = (1011\ 1101)_2$$

- Ce système de notation est très pratique pour noter les codes des couleurs (par exemple en RGB : `#7455BA` signifie que l'octet représentant le canal rouge a pour valeur `74`, celui du canal vert `55`, et celui du canal bleu `BA`), pour les clés de chiffrement (code Wifi par exemple), ...

Roue numérique

3. Opérations élémentaires sur les nombres binaires

3.1. Sommes de nombres binaires

Méthode : Additionner deux nombres entiers en base 2

La technique d'addition de deux nombres binaire est la même que pour des nombres en écriture décimale :

En décimal

$$\begin{array}{cccc} & 1 & 4 & 9 \\ + & & 7 & 8 \\ \hline \end{array}$$

En binaire

$$\begin{array}{ccccccccc} & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ + & & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ \hline \end{array}$$

3.2. Produits de nombres binaires

Méthode : Multiplier deux nombres entiers en base 2

La technique de multiplication de deux nombres binaire est la même que pour des nombres en écriture décimale - mais la retenue peut se propager parfois plus loin que le rang immédiatement supérieur :

En décimal

$$\begin{array}{rrr} & 2 & 7 \\ \times & 1 & 3 \\ \hline & & \\ + & & \\ \hline \end{array}$$

En binaire

$$\begin{array}{rrrrrrrr} & & & 1 & 1 & 0 & 1 & 1 \\ \times & & & & 1 & 1 & 0 & 1 \\ \hline & & & & & & & \\ + & & & & & & & \\ + & & & & & & & \\ + & & & & & & & \\ \hline \end{array}$$