Document élève

Cours :

Codage de l'information

Dans ce cours, vous découvrirez les différents **systèmes de numération**, dont le système binaire et ses variantes.

La puissance des systèmes numériques...

Les systèmes numériques, au sens large, ont envahi nos existences. Ils sont partout. Ordinateurs de salon ou portables, tablettes ou encore smartphones sont équipés de **processeurs** puissants, capables de traiter, en temps réel, une très grande quantité d'informations : images fixes, vidéos, musique...

Enfin, des multitudes de **capteurs connectés** de toutes sortes, souvent équipés de petits microcontrôleurs bien moins puissants, réalisent les contrôles d'accès grâce aux technologies RFID. Cela concerne par exemple la surveillance de la pollution (en tant que « nez électroniques ») ou de la météo.

Du côté des systèmes puissants, **la 3D est entrée massivement dans nos vies**. Les maquettes des êtres extraordinaires de nos films préférés ne sont plus faites de pâte à modeler, mais sont virtuelles, numériques, traitées par les processeurs des ordinateurs.

Pourtant, l'ordinateur ne connaît et ne manipule **que des nombres**. Des nombres, et encore des nombres… En fait, il ne manipule que des 0 et des 1. C'est ainsi qu'on parle d'*électronique binaire*. Mais il les manipule vite, et en quantité astronomique.

Vocabulaire

Quelle que soit la nature de l'information traitée par un ordinateur (image, son, texte, vidéo), elle l'est toujours sous la forme d'un ensemble de nombres écrits en base 2, par exemple 01001011.Le terme **bit** signifie « binary digit », c'est-à-dire 0 ou 1 en numérotation binaire.
Il s'agit de la plus petite unité d'information manipulable par une machine numérique. Il est possible de représenter physiquement cette information binaire par un signal électrique ou magnétique, qui, au-delà d'un certain seuil, correspond à la valeur 1.

L'octet est une unité d'information composée de 8 bits. Il permet par exemple de stocker un caractère comme une lettre ou un chiffre. Une unité d'information composée de 16 bits est généralement appelée mot (en anglais word). Une unité d'information de 32 bits de longueur est appelée mot double.

Beaucoup d'informaticiens ont appris que 1 kilooctet valait 1024 octets. Or, depuis décembre1998, l'organisme international IEC a statué sur la question.

Voici les unités standardisées :

• Un kilooctet (ko) = 10^3 octets	• Un pétaoctet (Po)=10^{15} octets
• Un mégaoctet (Mo)=10^6 octets	• Un exaoctet (Eo)=10^{18} octets
• Un gigaoctet (Go)=10^9 octets	• Un zettaoctet (Zo)=10^{21} octets
• Un téraoctet (To)= 10^{12} octets	• Un yottaoctet (Yo)=10^{24} octets

Les bases décimale, binaire et hexadécimale

Nous utilisons le système décimal (base 10) dans nos activités quotidiennes. Ce système est basé sur dix symboles, de 0 à 9, avec une unité supérieure (dizaine, centaine, etc.) à chaque fois que dix unités sont comptabilisées.

C'est un système positionnel, c'est-à-dire que l'endroit où se trouve le symbole définit sa valeur. Ainsi, le 2 de 523 n'a pas la même valeur que le 2 de 132.

En fait, 523 est l'abréviation de 5 x 102 + 2 x 101 + 3 x 100.

On peut selon ce principe imaginer une infinité de systèmes numériques fondés sur des bases différentes. En informatique, outre la base 10, on utilise très fréquemment le système binaire (base 2) puisque l'algèbre booléenne est à la base de l'électronique numérique. Deux symboles suffisent : 0 et 1.

On utilise aussi très souvent le système hexadécimal (base 16) du fait de sa simplicité d'utilisation et de représentation pour les mots machines. Il faut alors six symboles supplémentaires : A (qui représente le 10), B (11), C (12), D (13),E (14) et F (15).Le tableau ci-dessous montre la représentation des nombres de 0 à 15 dans les bases 10, 2 et 16.

Décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Binaire	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Hexadécimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

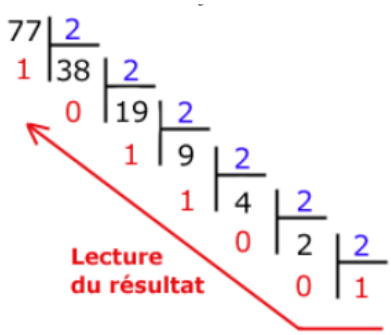
Conversion décimal - binaire

Convertissons 01001101 en décimal à l'aide du schéma ci-dessous :

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	1	1	0	1

Le nombre en base 10 est $2^6 + 2^3 + 2^2 + 2^0 = 64 + 8 + 4 + 1 = 77$.

Allons maintenant dans l'autre sens et écrivons 77 en base2. Il s'agit de faire une suite de divisions euclidiennes par 2. Le résultat sera la juxtaposition des restes.



Le schéma ci-contre explique la méthode :

77 s'écrit donc en base 2 : 1001101.

Si on l'écrit sur un octet, cela donne: 01001101.

Conversion hexadécimal - binaire

Pour convertir un nombre binaire en hexadécimal, il suffit de faire des groupes de quatre bits (en commençant depuis la droite).

Par exemple, convertissons 1001101 :

Binaire	0100	1101
Pseudo-décimal	4	13
Hexadécimal	4	D

1001101 s'écrit donc en base 16: 4D.

Pour convertir d'hexadécimal en binaire, il suffit de lire ce tableau de bas en haut.

Exercice 1 :

Donnez la méthode pour passer de la base décimale à la base hexadécimale (dans les deux sens).

Exercice 2 :

Complétez le tableau ci-dessous. L'indice indique la base dans laquelle le nombre est écrit.

	Bases		
	2	10	16
1001010110_2			
2002_{10}			
$A1C4_{16}$			

Représentation des nombres entiers

Représentation d'un entier naturel

Un entier naturel est un nombre entier positif ou nul.

Pour coder des nombres entiers naturels compris entre 0 et 255, il nous suffira de 8 bits (un octet) car $2^8 = 256$.

D'une manière générale un codage sur **n bits** pourra permettre de représenter des nombres entiers naturels comprisentre 0 et $2^n - 1$.

Exemples : $9 = 00001001_2$, $128 = 10000000_2$, etc.

Représentation d'un entier relatif

Un entier relatif est un entier **pouvant être négatif**. Il faut donc coder le nombre de telle façon que l'on puisse savoir s'il s'agit d'un nombre positif ou d'un nombre négatif, et il faut de plus que les règles d'addition soient conservées.

L'astuce consiste à utiliser un codage que l'on appelle **complément à deux**.

Cette représentation permet d'effectuer les opérations arithmétiques usuelles naturellement.

- **Un entier relatif positif** ou nul sera représenté en binaire (base 2) comme un entier naturel, à la seule différence que le bit de poids fort (le bit situé à l'extrême gauche) représente le signe. Il faut donc s'assurer pour un entier positif ou nul qu'il est à zéro (0 correspond à unsigne positif, 1 à un signe négatif). Ainsi, si on code un entier naturel sur 4 bits, le nombrele plus grand sera 0111 (c'est-à-dire 7 en base décimale).
    - Sur 8 bits (1 octet), l'intervalle de codage est [-128, 127] ;
    - Sur 16 bits (2 octets), l'intervalle de codage est [-32768, 32767] ;
    - Sur 32 bits (4 octets), l'intervalle de codage est [-2147483648, 2147483647].

    D'une manière générale le plus grand entier relatif positif codé sur n bits sera $2^{n-1}-1$.
- **Un entier relatif négatif** sera représenté grâce au codage en complément à deux.

Principe du complément à deux :

1. Écrire la valeur absolue du nombre en base 2. Le bit de poids fort doit être égal à 0.
2. Inverser les bits : les 0 deviennent des 1 et vice versa. On fait ce qu'on appelle le complément à un.
3. On additionne 1 au résultat (les dépassements sont ignorés).

Cette opération correspond au calcul de 2^n-|x|, où n est la longueur de la représentation et |x| lavaleur absolue du nombre à coder.

Ainsi -1 s'écrit comme 256 - 1 = 255 = 11111111_2, pour les nombres sur 8 bits.

Exemple :

On désire coder la valeur -19 sur 8 bits.

Il suffit :

1. d'écrire 19 en binaire : 00010011 ;
2. d'écrire son complément à 1 : 11101100 ;
3. et d'ajouter 1 : 11101101

La représentation binaire de -19 sur 8 bits est donc 11101101.

On remarquera qu'en additionnant un nombre et son complément à deux on obtient 0. En effet, 00010011 + 1101101 = 00000000 (avec une retenue de 1 qui est éliminée).

Astuce :

Pour transformer de tête un nombre binaire en son complément à deux, on parcourt le nombre de droite à gauche en laissant inchangés les bits jusqu'au premier 1 (compris), puis on inverse tous les bits suivants.

Prenons comme exemple le nombre 20 : 00010100.

1. On garde la partie à droite telle quelle: 00010**100** ;
2. On inverse la partie de gauche après le premier un : **11101**100 ;
3. Et voici -20: 11101100.

Le 4 juin 1996, une fusée Ariane 5 a explosé 40 secondes après l'allumage. La commission d'enquête a rendu son rapport : Il s'agissait d'une erreur de programmation dans le système inertiel de référence. À un moment donné, un nombre codé en virgule flottante sur 64 bits (qui représentait la vitesse horizontale de la fusée par rapport à la plate-forme de tir) était converti en un entier sur 16 bits. Malheureusement, le nombre en question était plus grand que 32768 (le plus grand entier quel'on peut coder sur 16 bits) et la conversion a été incorrecte.

Exercice 3 :

1. *Codez les entiers relatifs suivants sur 8 bits (16 si nécessaire) : 456, -1, -56, -5642.*
2. *Que valent en base dix les trois entiers relatifs suivants :*
   - *01101100*
   - *11101101*
   - *1010101010101010*

Exercice 4 :

Certains logiciels utilisent la représentation POSIX du temps, dans laquelle le temps est représenté comme un nombre de secondes écoulées depuis le 1er janvier 1970 à minuit (0 heure).

Sur les ordinateurs 32 bits, la plupart des systèmes d'exploitation représentent ce nombre comme un nombre entier signé de 32 bits.

1. *Quel est le nombre de secondes maximum que l'on peut représenter ?*
2. *À partir de l'application de conversion de « timestamp » en « date », indiquer à quelle date cela correspond-il (jour, mois, année, heures, minutes, secondes).*
3. *Que se passera-t-il une seconde plus tard ? Quel sera le nombre de secondes affiché (en base 10) ?*
4. *À quelle date cela correspond-il?*

Représentation des nombres réels

En base 10, l'expression 652,375 est une manière abrégée d'écrire :

$6 \cdot 10^2 + 5 \cdot 10^1 + 2 \cdot 10^0 + 3 \cdot 10^{-1} + 7 \cdot 10^{-2} + 5 \cdot 10^{-3}$.

Il en va de même pour la base 2. Ainsi, l'expression 110,101 signifie :$1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$

Conversion de binaire en décimal

On peut ainsi facilement convertir un nombre réel de la base 2 vers la base 10.

Par exemple : $110{,}1012 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 4 + 2 + 0{,}5 + 0{,}125 = 6{,}62510$.

Exercice 5 :

- *Transformez $0{,}0101010101_2$ en base 10.*
- *Transformez $11100{,}10001_2$ en base 10.*

Conversion de décimal en binaire

Le passage de base 10 en base 2 est plus subtil.

Par exemple : convertissons 1234,347 en base 2.

- La partie entière se transforme comme vu précédemment : $1234_{10} = 10011010010_2$
- On transforme la partie décimale selon le schéma suivant :

$0{,}347 \cdot 2 = \mathbf{0}{,}694$	$0{,}347 = 0{,}\mathbf{0}...$
$0{,}694 \cdot 2 = \mathbf{1}{,}388$	$0{,}347 = 0{,}0\mathbf{1}...$
$0{,}388 \cdot 2 = \mathbf{0}{,}766$	$0{,}347 = 0{,}01\mathbf{0}...$
$0{,}766 \cdot 2 = \mathbf{1}{,}552$	$0{,}347 = 0{,}0101...$
$0{,}552 \cdot 2 = \mathbf{1}{,}104$	$0.347 = 0{,}01011...$
$0{,}104 \cdot 2 = \mathbf{0}{,}208$	$0{,}347 = 0{,}010110...$
$0{,}208 \cdot 2 = \mathbf{0}{,}416$	$0{,}347 = 0{,}0101100...$
$0{,}416 \cdot 2 = \mathbf{0}{,}832$	$0{,}347 = 0{,}01011000...$
$0{,}832 \cdot 2 = \mathbf{1}{,}664$	$0{,}347 = 0{,}010110001...$

On continue ainsi jusqu'à la précision désirée…

Attention ! Un nombre à développement décimal fini en base 10 ne l'est pas forcément en base 2.

Cela peut engendrer de mauvaises surprises.

Prenons par exemple le programme ci-contre en Python :

```
i=0.0
while i<1:
    print(i)
    i+=0.1
```

Résultat attendu	Résultat réel
0.0	0.0
0.1	0.1
0.2	0.2
0.3	0.30000000000000004
0.4	0.4
0.5	0.5
0.6	0.6
0.7	0.7
0.8	0.7999999999999999
0.9	0.8999999999999999
	0.9999999999999990

Le problème vient du fait que l'on n'arrive pas à représenter exactement 0.3 en binaire.

Le 25 février 1991, pendant la Guerre du Golfe, une batterie américaine de missiles Patriot, à Dharan (Arabie Saoudite), a échoué dans l'interception d'un missile Scud irakien. Le Scud a frappé un baraquement de l'armée américaine et a tué 28 soldats. La commission d'enquête a conclu à un calcul incorrect du temps de parcours, dû à un problème d'arrondi. Les nombres étaient représentés en virgule fixe sur 24 bits. Le temps était compté par l'horloge interne du système en dixième de seconde. Malheureusement, 1/10 n'a pas d'écriture finie dans le système binaire : 1/10= 0,1 (dans le système décimal) = 0,00011001100110011001100110011... (dans le système binaire). L'ordinateur de bord arrondissait 1/10 à 24 chiffres, d'où une petite erreur dans le décompte dutemps pour chaque 1/10 de seconde. Au moment de l'attaque, la batterie de missile Patriot étaitallumée depuis environ 100 heures, ce qui a entraîné une accumulation des erreurs d'arrondi de0,34 s. Pendant ce temps, un missile Scud parcourt environ 500 m, ce qui explique que le Patriotsoit passé à côté de sa cible.

Exercice 6 :

Transformez en base 2 :

- $0{,}5625_{10}$
- $0{,}15_{10}$
- $12{,}9_{10}$