CODAGE DES INFORMATIONS

Représentation des nombres entiers et à virgule

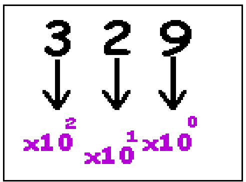
Vus de l'extérieur, les ordinateurs et les programmes que nous utilisons tous les jours permettent de mémoriser, de transmettre et de transformer des nombres, des textes, des images, des sons, etc.

Un ordinateur ne manipule que des 0 et des 1. Les nombres, les textes, les images, les sons, etc. mémorisés, transmis, transformés par les ordinateurs doivent être représentés comme des suites de 0 et de 1. Une telle valeur, 0 ou 1, s'appelle un **booléen**, un **chiffre binaire** ou encore un **bit** (*binary digit).* Une suite de bits, par exemple 0000110100 , est appelé un mot.

I) Les systèmes de numération : représentation des entiers naturels

A) Numération décimale

La numération décimale utilise 10 chiffres : 0 1 2 3 4 5 6 7 8 9



L'écriture du nombre 329 se traduit par $329 = 3\text{x}100 + 2\text{x}10 + 9\text{x}1 = 3\text{x}10^2 + 2\text{x}10^1 + 9\text{x}10^0$.
329 est la représentation en base 10, encore appelée écriture décimale.

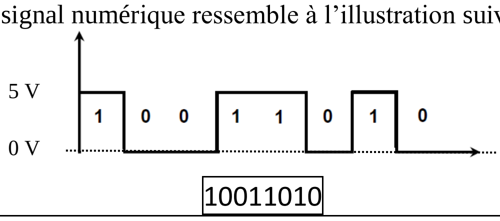
1°) Écrire une égalité semblable pour les nombres 2 134 et 805.

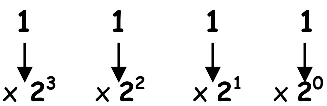
B) Numération binaire

L'informatique utilise des courants électriques, des aimantations, des rayons de lumière...
Chacun de ces phénomènes met en jeu deux états possibles. :

- tension nulle ou tension non nulle (5V par ex),
- aimantation dans un sens ou dans l'autre sens,
- lumière ou pas de lumière.

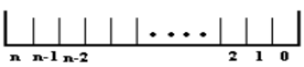
Il suffit de deux chiffres pour traduire ces états : c'est la numération binaire qui utilise les chiffres 0 et 1.

Un rayon de lumière peut parfaitement traduire ces deux valeurs: • 1 = lumière • 0 = pas de lumière 	Du courant peut être envoyé sur le fil électrique pour transmettre un '1' binaire, et l'absence de courant équivaut à un '0' binaire. Un signal numérique ressemble à l'illustration suivante. 

Le nombre binaire $(1111)_2$ se traduit par: 	$1111 = 1 \text{ x } \mathbf{2^3} + 1 \text{ x } \mathbf{2^2} + 1 \text{ x } \mathbf{2^1} + 1 \text{ x } \mathbf{2^0} = 8 + 4 + 2 + 1 = 15$ 1111 est donc la représentation binaire(en base 2) de 15

Chaque chiffre binaire (0 ou 1) se nomme **BIT** (de **BI**nary digi**T).**

Voici le schéma d'une mémoire à n+1 bits (au minimum 8 bits dans un micro-ordinateur) :



Les cases du schéma représentent les bits, le chiffre marqué en dessous d'une case, indique la puissance de 2 à laquelle est associé ce bit (on dit aussi le *rang* du bit).

Le bit de rang 0 est appelé le bit de **poids faible**.
Le bit de rang n est appelé le bit de **poids fort**.

2°) Traduire en écriture décimale les nombres binaires: 1 0 1 0 ; 1 1 0 0 1.

Un nombre binaire de huit chiffres est un **octet**. (octo = huit)

3°) Quel est le plus grand nombre que l'on peut représenter avec un octet ?

4°) Justifier qu'avec un **mot** de *n* bits on peut représenter les nombres de 0 à $2^n - 1$.

5°) Pour multiplier par dix un entier naturel exprimé en base dix, il suffit d'ajouter un 0 à sa droite, par exemple, 12 × 10 = 120.
Quelle est l'opération équivalente pour les entiers naturels exprimés en base deux ? Exprimer en base deux les nombres 3, 6 et 12 pour illustrer cette remarque

Les multiples et sous-multiples des unités employées en informatique

	Valeur exacte	en pratique dans le système international SI	Exemples d'utilisation
1 kilo-octet(Ko)	2^{10} = 1024 octets	10^3 octets	un modem téléphonique débite à 56,6 Kbps, un téléphone GSM à 9.6 Kbps une page de texte correspond environ 2 Ko.
1 Méga-octet(Mo)	2^{10} Ko = 1 048 576 octets	10^6 octets	un CDROM contient 640 Mo
1 Giga-octet (Go)	2^{10} Mo =1 073 741 824 octets	10^9 octets	1 DVD a une capacité de stockage de 4,7 Go 10 mètres de livres dans une bibliothèque tiendraient sur environ 1 Go un étage rempli complètement de journaux équivaut à 100 Go, c'est à dire au disque dur d'un PC
1 Téra-octet (To)	2^{10} Go =1 099 511 627 776 octets	10^{12} octets	une grande bibliothèque publique tiendrait sur 1 To environ l'ordinateur Blue Gene/L d'IBM calcule à la vitesse de 36Tflops (36 Téra opérations par seconde) YouTube annonce plus de mille milliards de vidéos vues

C) Numération hexadécimale

La numération hexadécimale utilise 16 chiffres: 0 1 2 3 4 5 6 7 8 9 A B C D E F

A est donc le chiffre "dix", B le chiffre "onze" , etc …..

Par exemple, le nombre hexadécimal 23A vaut : $2.16^2 + 3.16^1 + 10.16^0 = 512 + 48 + 10 = 570$

7°) Traduire en nombre décimal le nombre hexadécimal : B 8 C

Cette numération est utilisée pour les adresses des mémoires. Exemple d'adresse : B8AC 000F

Elle est aussi utilisée pour coder les couleurs:

000000	0000FF	00FF00	FF0000	FFFF00	FFFFFF
Noir	*Bleu*	*Vert*	*Rouge*	*Jaune*	*Blanc*

8°) A quel nombre décimal correspond le nombre hexadécimal F F ?

9°) Quelle remarque faites-vous par rapport à l'octet?

La taille rapidement encombrante de l'écriture binaire a conduit les informaticiens à faire un usage intensif du système hexadécimal (base 16).

II) Changement de base

A) Passage de la base *b* à la base 10.

$\overline{a_k a_{k-1} a_{k-2} \dots a_2 a_1 a_0}^{(b)}$ écrit en base *b* désigne l'entier $a_k \times b^k + a_{k-1} \times b^{k-1} + a_{k-2} \times b^{k-2} + \dots + a_2 \times b^2 + a_1 \times b^1 + a_0 \times b^0$

Le chiffre le plus à droite a_0 représente le chiffre des unités, le précédent a_1 les paquets de b^1, le précédents les paquets de b^2 etc….

10°) Donner l'écriture décimale de $\overline{76152}^{(8)}$, de $\overline{BA7645}^{12}$.

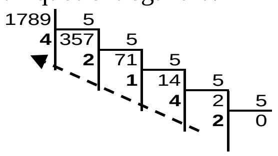
B) Passage de la base 10 à la base *b*.

Approche

$$3294 = 329 \times 10 + 4 = \left((32 \times 10) + 9\right) \times 10 + 4 = \left(\left(\left(3 \times 10 + 2\right) \times 10\right) + 9\right) \times 10 + 4 = 3 \times 10^3 + 2 \times 10^2 + 9 \times 10^1 + 4 \times 10^0$$

L'écriture en base 10 consiste donc effectuer des regroupements successifs en paquets de 10 (10 symboles pour l'écriture décimale). Cela revient donc à faire des divisions successives par 10 tant que possibles.

Pour écrire les entiers naturels en base *b*, on a besoin de *b* chiffres. Quand on a *n* objets, on les groupe par paquets de *b*, qu'on regroupe à leur tour en paquets de *b* paquets, etc.
Autrement dit, on fait une succession de divisions par *b*, jusqu'à obtenir un quotient égal à 0.



⇒ arrêt

1789 s'écrit donc $\overline{24124}^{(5)}$

11°) Trouver la représentation en base seize puis en base 2 des nombres 6725 et 18 379 (écrits en base 10).

C) Passerelle entre la base 2 et la base 16.

- **Passage de la base 2 à la base 16**

On décompose ce nombre par tranches de 4 bits à partir du bit de poids faible ($2^4 = 16$).

- On complète la dernière tranche (celle des bits de poids forts)par des 0 s'il y a lieu.
- On convertit chaque tranche en son symbole de la **base 16.**
- On réécrit à sa place le nouveau symbole par changements successifs de chaque groupe de 4 bits,

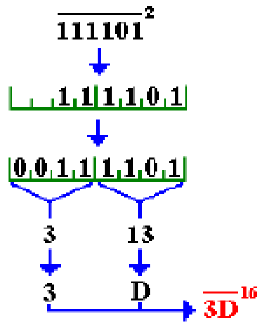
Exemple :

Soit le nombre **111101**$_2$

à convertir en héxadécimal.

Résultat obtenu :

111101$_2$ **= 3D**$_{16}$



- **Passage de la base 16 à la base 2**

Cette conversion est l'opération inverse de la précédente.

- On convertit chaque symbole hexadécimal du nombre en son écriture binaire (nécessitant au plus 4 bits).
- Pour chaque tranche de 4 bits, on complète les bits de poids fort par des 0 s'il y a lieu.
- On regroupe toutes les tranches de 4 bits à partir du bit de poids faible, sous forme d'un seul nombre binaire.

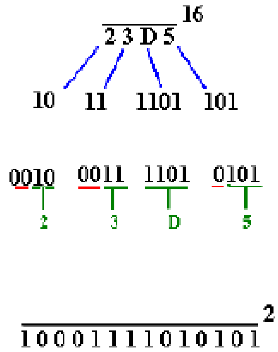
Exemple :

Soit le nombre **23D5**$_{16}$

à convertir en binaire.

Résultat obtenu :

23D5$_{16}$ **= 10001111010101**$_2$



12°) convertir $\overline{1100011010}^{2}$ en base 16 et $\overline{7CF21}^{16}$ en base 2.

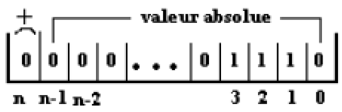
Vous réaliserez des mini-projets ayant pour objectif de programmer des convertisseurs binaire en décimal et vice et versa à la suite des TP de programmation.

III) <u>Représentation des entiers relatifs (encore nommés entiers signés)</u>

Pour représenter les entiers relatifs en notation binaire, on doit étendre la représentation des positifs aux nombres négatifs.

<u>1ère solution avec les premières machines</u> :
une solution est de réserver le bit de poids fort pour le signe de l'entier à représenter (1 pour – et 0 pour +) et d'utiliser les autres pour représenter sa valeur absolue

Exemple du codage en binaire signé des nombres **+14** et **-14** :

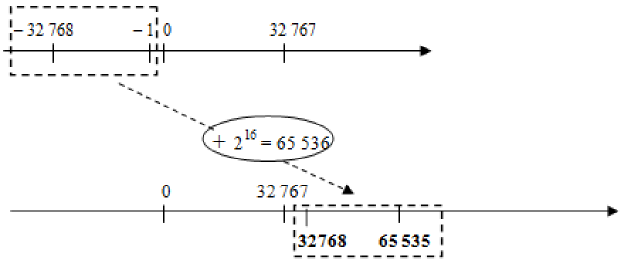
	
+14 est représenté par 0000...01110	*-14 est représenté par 1000...01110*

13°) Ainsi, avec des mots de 16 bits, en utilisant 1 bit pour le signe et 15 bits pour la valeur absolue, quels sont les entiers relatifs que l'on peut représenter ?

14°) Donner un inconvénient de cette méthode

Ce codage n'est pas utilisé en pratique. En effet le traitement spécifique du signe coûte cher en circuits électroniques et en temps de calcul. C'est une version améliorée qui est utilisée dans la plupart des calculateurs : elle se nomme le complément à deux.

<u>2ème solution</u> : codage sur 16 bits
Si on utilise des mots de 16 bits, cette fois-ci les nombres 1000 0000 0000 0000 et 0000 0000 0000 0000 représente 2 nombres distincts donc on peut représenter les entiers relatifs compris entre $-2^{15} = -32\,768$ et $2^{15} - 1 = 32\,767$. L'idée est de transporter les entiers strictement négatifs dans le monde des entiers naturels et de laisser intact les entiers positifs (voir ci-dessous) :



Avec cette méthode, tous les entiers relatifs sont donc représentés par un entier naturel

15°) comment est alors représenté l'entier négatif – 1 ?

Cette manière de représenter les entiers relatifs s'appelle la notation en ***complément à deux***.

16°) Quels entiers relatifs peut-on représenter avec des mots de 8 bits ? Combien sont-ils ?
Même question avec des mots de 32 bits et 64 bits.

Trouver la représentation binaire sur *n* bits d'un entier relatif donné en décimal
On a vu que :
- Si l'entier relatif x est positif ou nul, on le représente comme l'entier naturel x.
- S'il est strictement négatif, on le représente comme l'entier naturel $x + 2^n$.

(avec corrigé)
Trouver la représentation binaire sur <u>huit</u> bits des entiers relatifs 0 et -128.
L'entier relatif 0 est représenté comme l'entier naturel 0 : 0000 0000.
L'entier relatif -128 est représenté comme l'entier naturel $-128 + 2^8 = 128 + 256 = 128$ *: 1000 0000.*

17°) Trouver la représentation binaire sur huit bits des entiers relatif 127 et -127.

Trouver la représentation décimale d'un entier relatif donné en binaire sur n bits
Si cet entier relatif est donné par le mot *m*, on commence par calculer l'entier naturel *p* représenté par ce mot.

- Si *p* est strictement inférieur à 2^{n-1}, c'est l'entier relatif représenté
- s'il est supérieur ou égal à 2^{n-1}, l'entier relatif représenté est $p - 2^n$.

(avec corrigé)
Trouver la représentation décimale des entiers relatifs dont la représentation binaire sur huit bits est 0000 0000 et 1000 0000.
Le mot 0000 0000 représente l'entier naturel 0 et donc l'entier relatif 0.
Le mot 1000 0000 représente l'entier naturel $128 \geq 2^7$ *et donc l'entier relatif* $128 - 2^8 = 128 - 256 = -128$.

18°) Trouver la représentation décimale des entiers relatifs dont la représentation binaire sur huit bits est 0111 1111 et 1000 0001.

IV) <u>La représentation des nombres à virgule</u>

En notation décimale, on utilise la notation scientifique pour représenter un nombre décimal : $a \times 10^p$ avec $1 \leq a < 10$ et $p \in \mathbb{Z}$.
Par exemple : $52980480 = 5,298048 \times 10^7$; $0,00000004 = 4 \times 10^{-8}$
De manière analogue, nous allons utiliser cette représentation en base 2 :
$s\ m\ 2^n$ où *s* est le signe du nombre : + ou –, *n* son exposant : $n \in \mathbb{Z}$ et *m* sa mantisse : $1 \leq m < 2$

Quand on utilise 64 bits pour représenter un nombre à virgule, on utilise 1 bit pour le signe, 11 bits pour l'exposant et 52 bits pour la mantisse. Le signe + est représenté par 0 et le signe - par 1.
19°) Est-il nécessaire d'utiliser les 52 bits pour coder toute la mantisse ? Justifier

<u>Un exemple</u> :
Quel est le nombre à virgule en décimal représenté par le mot ci-dessous ?

$$\underset{\text{signe}}{1}\ \underbrace{10001000110}_{\text{exposant}}\underbrace{1001001111000011100000000000000000000000000000000000}_{\text{mantisse}}$$

Le signe du nombre est donc –.
Le nombre 100 0100 0110 est égal à 1 094 et l'exposant du nombre est n = 1094 - 1023 = 71 (car $1094 \geq 2^{11-1}$, *voir* **18°)** *)*

Sa mantisse est :
$m = \boxed{1}$. 1001 0011 1100 0011 1000 0000 0000 0000 0000 0000 0000 0000 0000
le chiffre devant la virgule

Comment lire convertir en décimal les chiffres après la virgule ?
Prenons exemple sur la notation décimale : 1,023507 s'écrit $1 + \frac{2}{10^2} + \frac{3}{10^3} + \frac{5}{10^4} + \frac{7}{10^6}$
symbole
base
rang

Sur ce modèle cela donne :
$m = \boxed{1}$. 1001 0011 1100 0011 1000 0000 0000 0000 0000 0000 0000 0000 0000

$$= 1 + \frac{1}{2} + \frac{1}{2^4} + \frac{1}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{1}{2^{10}} + \frac{1}{2^{15}} + \frac{1}{2^{16}} + \frac{1}{2^{17}}$$

$$= \frac{206727}{131072}$$

Le nombre représenté est donc $-\frac{206727}{131072} \times 2^{71} = -3,724\ldots\ldots \times 10^{21}$

20°) Trouver le nombre à virgule représenté par le mot :
0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.

21°) Comment est représenté le nombre entier 7 ? Et le nombre à virgule 7.0 ?

22°) Quel est le plus grand nombre à virgule que l'on peut représenter en binaire ?
quel est le plus petit nombre à virgule, donc négatif, que l'on peut représenter en binaire ?
quel est le plus petit nombre à virgule strictement positif que l'on peut représenter en binaire ?

Pour les plus rapides, voici [un jeu *Binary-game*]() pour manipuler des octets façon Tetris.