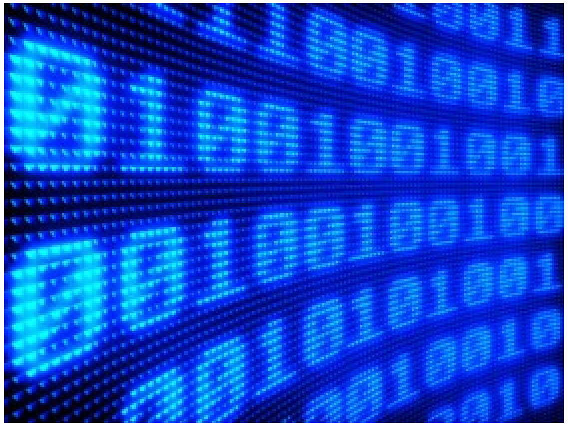
Qu'est-ce que le binaire ?



Les systèmes numériques réagissent à des signaux « tout ou rien ». On représente les deux états stables ainsi définis par les symboles « 0 » et « 1 » ou encore par « L » (*Low*) et « H » (*High*).

Le système de numération adapté à la représentation de tels signaux est donc la **base 2**, on parle aussi de **codage binaire**.

L'**unité de codage** de l'information est donc un élément ne pouvant prendre que les valeurs 0 ou 1 : le **bit**, contraction de ***Binary Digit***, c'est-à-dire chiffre binaire.

L'octet

Pour les échanges de données, les informations élémentaires (bits) sont manipulés par groupes qui forment ainsi des **mots binaires**.
L'unité de transfert utilisée pour les échanges de données est le mot de 8 bits appelé **octet** (byte).

Dans un mot binaire, le bit situé à gauche est le bit le plus significatif, **MSB** (*Most Significant Bit*), celui de droite est le bit le moins significatif, **LSB** (*Less Significant Bit*).

Numéro du bit →	7	6	5	4	3	2	1	0
Bit 0 ou 1 →								
	MSB							LSB

Les multiples de l'octet

La capacité en octets des différents constituants tels que circuits mémoires, disques durs, … est souvent très importante : il devient alors indispensable d'utiliser des **unités multiples de l'octet** :

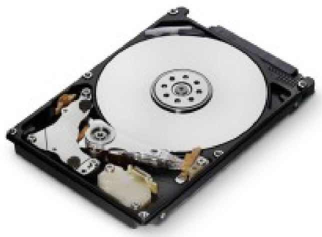
1 **kilo-octet** (ko)	=	10^3 octets	= 1000 octets	
1 **méga-octet** (Mo)	=	……… octets	= ……… Ko	= ……………………………… octets
1 **giga-octet** (Go)	=	……… octets	= ……… Mo	= ……………………………… octets
1 **téra-octet** (To)	=	……… octets	= ……… Go	= ……………………………… octets

La normalisation des préfixes binaires de 1998 par la Commission Electronique Internationale (CEI) spécifie les préfixes suivants pour représenter les puissances de 2 :

1 **kibi-octet** (Kio)	= 2^{10} octets	= 1 024 octets	
1 **mébi-octet** (Mio)	= 2^{20} octets	= 1 024 Kio	= 1 048 576 octets
1 **gibi-octet** (Gio)	= 2^{30} octets	= 1 024 Mio	= 1 073 741 824 octets
1 **tébi-octet** (Tio)	= 2^{40} octets	= 1 024 Gio	= 1 099 511 627 776 octets

Applications

Ex1 : la fiche technique d'un disque dur externe indique une capacité de 320 GB. Exprimer cette capacité en Mio et Gio.



Ex2 : votre fournisseur ADSL vous annonce un débit descendant de 8192 kibits/s. Vous faîtes une mesure de débit et vous trouvez une moyenne de 3280 kibits/s. Quel sera le temps réel minimal de téléchargement d'une application de taille égale à 25 Mo ?

Conversion décimal => binaire

Il faut décomposer le nombre en somme de puissance de 2 :

$$123_{(10)} = 64 + 32 + 16 + 8 + 2 + 1 = 2^6 + 2^5 + 2^4 + 2^3 + 2^1 + 2^0 = 1111011_{(2)}$$

Conversion binaire => décimal

Il suffit de réécrire la somme de puissance de 2

$$1010101_{(2)} = 2^6 + 2^4 + 2^2 + 2^0 = 85_{(10)}$$

Remarque : en informatique un nombre écrit en binaire est précédé du caractère *%* ou encore de *0b*.

Applications

Ex1 : convertir en binaire les nombres 29, 136 et 224.

Ex2 : convertir en décimal les nombres binaires $11111111_{(2)}$, $10000000_{(2)}$, $10101010_{(2)}$.

L'hexadécimal

Le binaire est fastidieux à manipuler... l'**hexadécimal** est du binaire « réarrangé » : chaque quartet binaire (4 bits) est remplacé par un symbole unique. On utilise la base 16, Il faut donc trouver 16 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Le tableau montre comment est représenté, en informatique, un nombre hexadécimal.

Préfixe	Exemple	Langages
0x	0xAE4F	C, C++, Java
$	$AE4F	Pascal
&h	&HAE4F	Basic
#	#AE4F	HTML

Codage des informations : le binaire

Applications

Ex1 : compter de 0 à 15 en hexadécimal

Ex2 : convertir en hexadécimal les nombres 29, 136 et 224.

décimal	binaire	hexadécimal
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Codage des informations : le binaire

Les nombres négatifs

Pour coder en binaire un nombre négatif on utilise le **MSB** comme **bit de signe** :

MSB = 1 : le nombre est négatif

MSB = 0 : les nombre est positif ou nul

Codage réel	Hexa.	Interprétation décimale signée	
% 1000.0000	($80)	- 128	(plus grand nombre négatif)
% 1000.0001	($81)	- 127	
% 1000.0010	($82)	- 126	
... ...		...	...
% 1111.1110	($FE)	- 2	
% 1111.1111	($FF)	- 1	
% 0000.0000	($00)	0	(valeur centrale)
% 0000.0001	($01)	1	
% 0000.0010	($02)	2	
... ...		...	...
% 0111.1110	($7E)	126	
% 0111.1111	($7F)	127 (plus grand nombre positif)	

Les nombres négatifs

Le passage d'une valeur positive à la valeur négative de même valeur absolue peut être obtenue par le **complément à 2** : inversion bit à bit puis ajout de 1 : **$- N = \overline{N} + 1$**

Ex : $126_{(10)} = 01111110_{(2)}$

$-126_{(10)} = 10000001 + 1 = 10000010_{(2)}$

Applications

Ex1 : Écrire $-123_{(10)}$ et $-67_{(10)}$ en binaire.

Ex2 : Trouver la représentation décimale des entiers relatifs dont la représentation binaire sur 8 bits est $01111111_{(2)}$ et $10000001_{(2)}$.