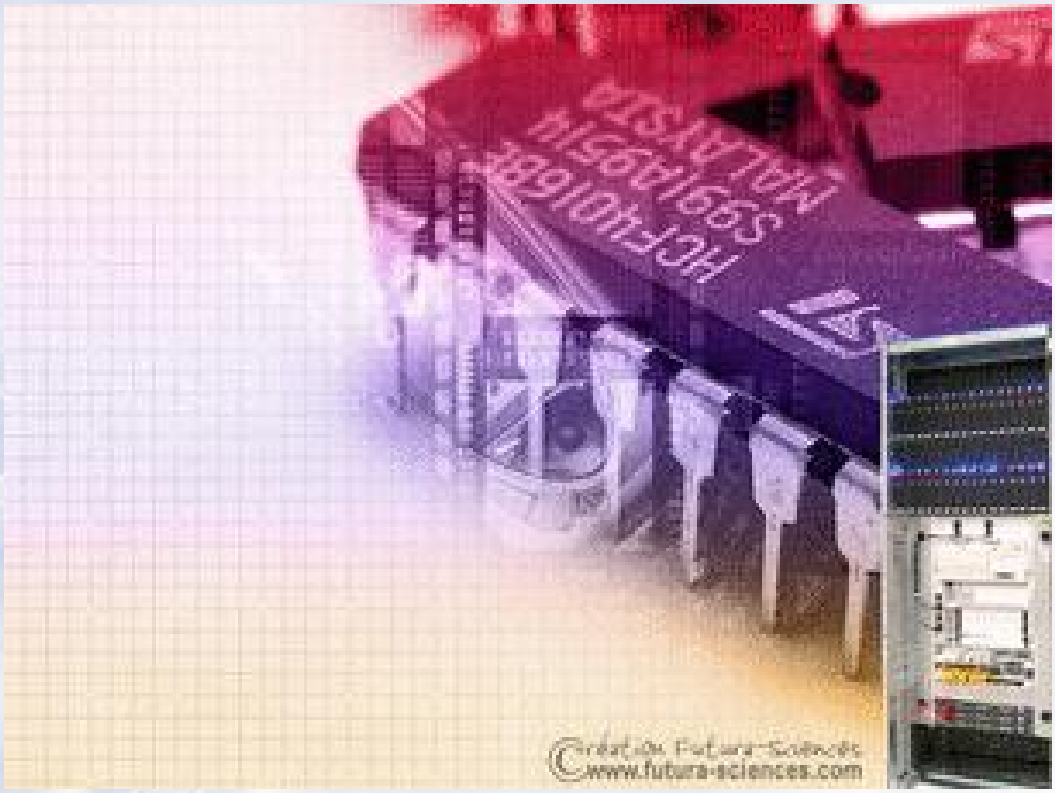
BASES DE CONNAISSANCE POUR L'INFORMATIQUE

Par JJ Pellé



SOMMAIRE

BASES DE CONNAISSANCE POUR L'INFORMATIQUE

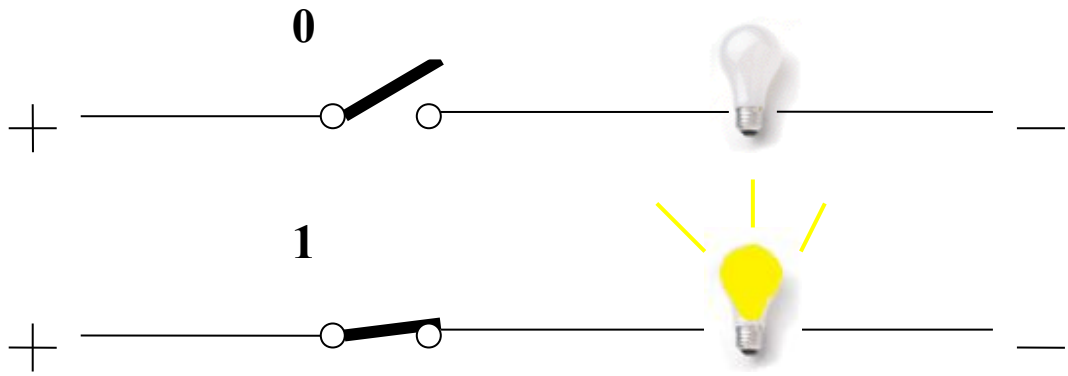
1 - Le système binaire

Pourquoi le système binaire a-t-il été choisi ?

Tout simplement parce que la fabrication d'un 0 ou d'un 1 est facilement réalisable en électronique et très économique au niveau composants.

2 - Bit

Le terme **bit** (*b* avec une minuscule dans les notations) signifie « **binary digit** » (élément binaire en français), 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 ;



Avec un bit il est ainsi possible d'obtenir deux états : soit 1, soit 0. Grâce à 2 bits, il est possible d'obtenir quatre états différents (2*2) :

Valeur binaire sur 2 bits		Valeur décimale
0	0	0
0	1	1
1	0	2
1	1	3

Avec 3 bits, il est possible d'obtenir huit états différents (2*2*2) :

Valeur binaire sur 3 bits	Valeur décimale
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Pour un groupe de n bits, il est possible de représenter 2^n valeurs.

Poids des bits

Dans un nombre binaire, la valeur d'un bit, appelée **poids**, dépend de la position du bit en partant de la droite. A la manière des dizaines, des centaines et des milliers pour un nombre décimal, le poids d'un bit croît d'une puissance de deux en allant de la droite vers la gauche comme le montre le tableau suivant :

Nombre binaire	1	1	1	1	1	1	1	1
Poids	$2^7 = 128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$

Conversions

Pour convertir un mot binaire en nombre décimal, il suffit de multiplier la valeur de chaque bit par son poids, puis d'additionner chaque résultat. Ainsi, le mot binaire 0101 vaut en décimal :

$$2^3x0 + 2^2x1 + 2^1x0 + 2^0x1$$
$$= 8x0 + 4x1 + 2x0 + 1x1$$
$$= 5$$

3 - L'Octet

L'**octet** (en anglais *byte* ou *B* avec une majuscule dans les notations) est une unité d'information composée de 8 bits. Il permet par exemple de stocker un caractère, tel qu'une lettre ou un chiffre.

Ce regroupement de nombres par série de 8 permet une lisibilité plus grande, au même titre que l'on apprécie, en base décimale, de regrouper les nombres par trois pour pouvoir distinguer les milliers. Le nombre « 1 256 245 » est par exemple plus lisible que « 1256245 ».

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** (en anglais *double word*, d'où l'appellation *dword*).

Pour un octet, le plus petit nombre est 0 (représenté par huit zéros 00000000), et le plus grand est 255 (représenté par huit chiffres « un » 11111111), ce qui représente 256 possibilités de valeurs différentes.

2^7 =128	2^6 =64	2^5 =32	2^4 =16	2^3 =8	2^2 =4	2^1 =2	2^0 =1
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1

KiloOctets, MégaOctets

Longtemps l'informatique s'est singularisée par l'utilisation de différentes valeurs pour les unités du système international. Ainsi beaucoup d'informaticiens ont appris que 1 kilooctet valait 1024 octets. Or, depuis décembre 1998, l'organisme international *IEC* a statué sur la question (http://physics.nist.gov/cuu/Units/binary.html). Voici donc les unités standardisées :

Contexte historique*

Les informaticiens ont défini que 1 Ko = 2^{10} octets soit 1024 octets, mais la micro informatique ayant explosée, il a fallu s'adresser à des gens ordinaires et en comparaison au kg qui vaut 1000, il a fallu s'adapter pour faciliter la compréhension.

- Un kilooctet (ko ou kB) = 1000 octets
- Un Mégaoctet (Mo ou MB) = 1000 ko = 1 000 000 octets
- Un Gigaoctet (Go ou GB) = 1000 Mo = 1 000 000 000 octets
- Un Téraoctet (To) = 1000 Go = 1 000 000 000 000 octets

Attention ! De nombreux logiciels (parfois même certains systèmes d'exploitation)

utilisent toujours la notation antérieure à 1998 pour laquelle :

- Un kilooctet (ko) = 2^{10} octets = 1024 octets
- Un Mégaoctet (Mo) = 2^{20} octets = 1024 ko = 1 048 576 octets
- Un Gigaoctet (Go) = 2^{30} octets = 1024 Mo = 1 073 741 824 octets

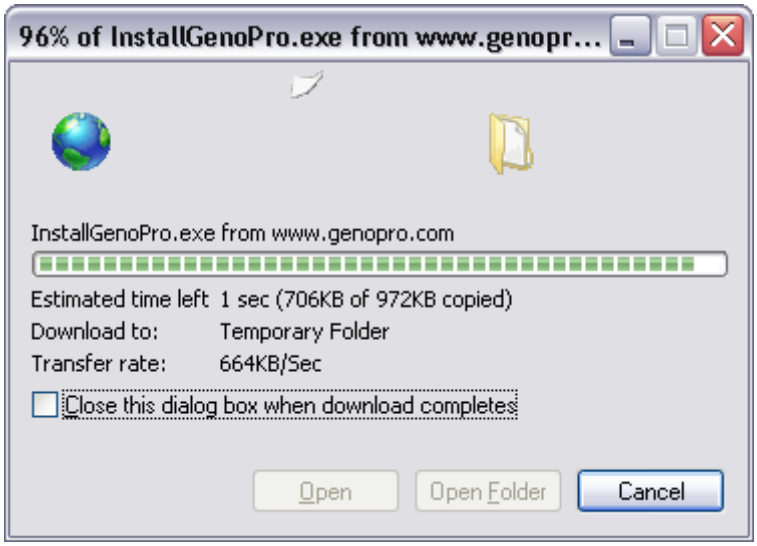
- Un Téraoctet (To) = 2^{40} octets = 1024 Go = 1 099 511 627 776 octets

Il est également utile de noter que la communauté internationale dans son ensemble utilise préférentiellement le nom de « byte » plutôt que le terme « octet » purement francophone. Cela donne les notations suivantes pour kilobyte, mégabyte, gigabyte et terabyte :

kB, MB, GB, TB

Notez l'utilisation d'un *B* majuscule pour différencier *Byte* et *bit*.

Voici une capture d'écran du logiciel *Internet Explorer*, navigateur internet, montrant l'utilisation de cette notation :



4 - Opérations en binaire

Les opérations arithmétiques simples telles que l'addition, la soustraction et la multiplication sont faciles à effectuer en binaire.

Addition binaire

L'addition en binaire se fait avec les mêmes règles qu'en décimale :
On commence à additionner les bits de poids faible (les bits de droite) puis on a des retenues lorsque la somme de deux bits de même poids dépasse la valeur de l'unité la plus grande (dans le cas du binaire : 1), cette retenue est reportée sur le bit de poids plus fort suivant...

Par exemple :

```
  0 1 1 0 1
+ 0 1 1 1 0
- - - - - -
  1 1 0 1 1
```

Multiplication binaire

La table de multiplication en binaire est très simple :

- 0x0=0
- 0x1=0
- 1x0=0
- 1x1=1

La multiplication se fait en formant un produit partiel pour chaque digit du multiplicateur (seuls les bits non nuls donneront un résultat non nul). Lorsque le bit du multiplicateur est nul, le produit partiel est nul, lorsqu'il vaut un, le produit partiel est constitué du multiplicande décalé du nombre de positions égal au poids du bit du multiplicateur.

Par exemple :

```
    0 1 0 1 multiplicande
x   0 0 1 0 multiplicateur
- - - - - -
    0 0 0 0
  0 1 0 1
0 0 0 0
- - - - - -
  0 1 0 1 0
```

5 - Le codage des informations

Le morse a été le premier codage à permettre une communication longue distance. C'est *Samuel F.B.Morse* qui l'a mis au point en 1844. Ce code est composé de points et de tirets (un codage binaire en quelque sorte...). Il permit d'effectuer des communications beaucoup plus rapides que ne le permettait le système de courrier de l'époque aux Etats-Unis : le Pony Express. L'interpréteur était l'homme à l'époque, il fallait donc une bonne connaissance du code...

De nombreux codes furent inventés dont le code d'Émile Baudot (portant d'ailleurs le nom de *code Baudot*, les anglais l'appelaient en revanche *Murray Code*).

Le 10 mars 1876, le Dr Graham Bell met au point le téléphone, une invention révolutionnaire qui permet de faire circuler de l'information vocale dans des lignes métalliques. Pour l'anecdote, la Chambre des représentants a décidé que l'invention du téléphone revenait à Antonio Meucci. Ce dernier avait en effet déposé une demande de brevet en 1871, mais n'avait pas pu financer celle-ci au-delà de 1874.

Ces lignes permirent l'essor des téléscripteurs, des machines permettant de coder et décoder des caractères grâce au code Baudot (les caractères étaient alors codés sur 5 bits, il y avait donc 32 caractères uniquement...).

Dans les années 60, le code ASCII (American Standard Code for Information Interchange) est adopté comme standard. Il permet le codage de caractères sur 8 bits, soit 256 caractères possibles.

Qu'est-ce que le code ASCII ?

La mémoire de l'ordinateur conserve toutes les données sous forme numérique. Il n'existe pas de méthode pour stocker directement les caractères. Chaque caractère possède donc son équivalent en code numérique : c'est le **code ASCII** (*American Standard Code for Information Interchange* - traduisez « Code Americain Standard pour l'Echange d'Informations »). Le code ASCII de base représentait les caractères sur 7 bits (c'est-à-dire 128 caractères possibles, de 0 à 127).

- Les codes 0 à 31 ne sont pas des caractères. On les appelle *caractères de contrôle* car ils permettent de faire des actions telles que :
  - retour à la ligne (CR)
  - Bip sonore (BEL)
- Les codes 65 à 90 représentent les majuscules
- Les codes 97 à 122 représentent les minuscules
  (Il suffit de modifier le 6ème bit pour passer de majuscules à minuscules, c'est-à-dire ajouter 32 au code ASCII en base décimale.)

Table des caractères ASCII

caractère	code ASCII	code hexadécimal
NUL (*Null*)	0	00
SOH (*Start of heading*)	1	01
STX (*Start of text*)	2	02
ETX (*End of text*)	3	03
EOT (*End of transmission*)	4	04
ENQ (*Enquiry*)	5	05
ACK (*Acknowledge*)	6	06
BEL (*Bell*)	7	07
BS (*Backspace*)	8	08

TAB (Tabulation horizontale)	9	09
LF (*Line Feed*, saut de ligne)	10	0A
VT (*Vertical tabulation*, tabulation verticale)	11	0B
FF (*Form feed*)	12	0C
CR (*Carriage return*, retour à la ligne)	13	0D
SO (*Shift out*)	14	0E
SI (*Shift in*)	15	0F
DLE (*Data link escape*)	16	10
DC1 (*Device control 1*)	17	11
DC2 (*Device control 2*)	18	12
DC3 (*Device control 3*)	19	13
DC4 (*Device control 4*)	20	14
NAK (*Negative acknowledgement*)	21	15
SYN (*Synchronous idle*)	22	16
ETB (*End of transmission block*, fin de bloc de transmission)	23	17
CAN (*Cancel*, annulation)	24	18
EM (*End of medium*, fin du médium)	25	19
SUB (*Substitute*, substitut)	26	1A
ESC (*Escape*, caractère d'échappement)	27	1B
FS (*File separator*, séparateur de fichier)	28	1C
GS (*Group separator*, séparateur de groupe)	29	1D
RS (*Record separator*, séparateur d'enregistrement)	30	1E
US (*Unit separator*, séparateur d'enregistrement)	31	1F
SP (*Space*, espace)	32	20
!	33	21
"	34	22
#	35	23
$	36	24
%	37	25
&	38	26
'	39	27
(	40	28
)	41	29
*	42	2A
+	43	2B
,	44	2C
-	45	2D
.	46	2E
/	47	2F
0	48	30
1	49	31
2	50	32
3	51	33
4	52	34
5	53	35

6	54	36
7	55	37
8	56	38
9	57	39
:	58	3A
;	59	3B
<	60	3C
=	61	3D
>	62	3E
?	63	3F
@	64	40
A	65	41
B	66	42
C	67	43
D	68	44
E	69	45
F	70	46
G	71	47
H	72	48
I	73	49
J	74	4A
K	75	4B
L	76	4C
M	77	4D
N	78	4E
O	79	4F
P	80	50
Q	81	51
R	82	52
S	83	53
T	84	54
U	85	55
V	86	56
W	87	57
X	88	58
Y	89	59
Z	90	5A
[	91	5B
\	92	5C
]	93	5D
^	94	5E
_	95	5F
`	96	60
a	97	61
b	98	62

c	99	63
d	100	64
e	101	65
f	102	66
g	103	67
h	104	68
i	105	69
j	106	6A
k	107	6B
l	108	6C
m	109	6D
n	110	6E
o	111	6F
p	112	70
q	113	71
r	114	72
s	115	73
t	116	74
u	117	75
v	118	76
w	119	77
x	120	78
y	121	79
z	122	7A
{	123	7B
|	124	7C
}	125	7D
~	126	7E
Touche de suppression	127	7F

Table des caractères ASCII Etendue

Le code ASCII a été mis au point pour la langue anglaise, il ne contient donc pas de caractères accentués, ni de caractères spécifiques à une langue. Pour coder ce type de caractère il faut recourir à un autre code. Le code ASCII a donc été étendu à 8 bits (un octet) pour pouvoir coder plus de caractères (on parle d'ailleurs de code ASCII étendu...).
Ce code attribue les valeurs 0 à 255 (donc codées sur 8 bits, soit 1 octet) aux lettres majuscules et minuscules, aux chiffres, aux marques de ponctuation et aux autres symboles (caractères accentués dans le cas du code *iso-latin1*).

```
Le code ASCII étendu n'est pas unique et dépend fortement de
la plateforme utilisée.
```

Les deux jeux de caractères ASCII étendus les plus couramment utilisés sont :

- Le code ASCII étendu OEM, c'est-à-dire celui qui équipait les premières machines de type IBM PC

- Le code ASCII étendu ANSI, utilisé par les systèmes d'exploitation récents

Le code EBCDIC

Le code *EBCDIC* (*Extended Binary-Coded Decimal Interchange Code*), développé par IBM, permet de coder des caractères sur 8 bits. Bien que largement répandu sur les machines IBM, il n'a pas eu le succès qu'a connu le code ASCII.

Unicode

Le code *Unicode* est un système de codage des caractères sur 16 bits mis au point en 1991. Le système Unicode permet de représenter n'importe quel caractère par un code sur 16 bits, indépendamment de tout système d'exploitation ou langage de programmation.

Il regroupe ainsi la quasi-totalité des alphabets existants (arabe, arménien, cyrillique, grec, hébreu, latin, ...) et est compatible avec le code ASCII.

L'ensemble des codes Unicode est disponible sur le site http://www.unicode.org.

6 - Comprendre les unités de stockage

La taille des mémoires est définie en Mo, Go ou To pour les derniers disques.

6 - 1Le Ko ou kilo-octet (1000 octets)

Le Ko est utilisé pour définir la taille de textes ou d'images de faible définition.

Que représente un Ko ? 1000 caractères alphabétiques par exemple (A,B,a,b…)

Un fichier de texte pèse quelques Ko
Une image ou photo de faible définition pèse 15 (GIF animé) à 100 Ko

6 – 2 Le Mo ou Méga-octet (1 million d'octets)

On trouve le Mo pour le format des photos, des fichiers vidéo, diaporamas ainsi que pour certains supports mémoire (mémoire vive, clefs USB anciennes).

Que représente un Mo ? 1 million de caractères alphabétiques par exemple, une photo de grande résolution, un diaporama (PPS)

6 – 3 Le Go ou Giga-octet (1 milliard d'octets)

On trouve le Go pour le format des fichiers vidéo et des supports mémoire (mémoire vive, disques durs, clefs USB….)

Que représente un Go ? 1 milliard de caractères alphabétiques par exemple, 1000 photos de grande résolution, un film Divx.

Les mémoires vives aujourd'hui font 3,4, jusqu'à 8 Go, les disques durs les plus courants ont une capacité de 250, 320, 500 Go

6 – 4 Le To ou Téraoctets (1000 milliards d'octets)

On trouve le To pour le format des derniers disques durs. La miniaturisation et l'évolution technologique ont permis de faire grandement progresser les capacités de stockage des disques durs.

Que représente un To ? 1000 milliards de caractères alphabétiques par exemple, 1000 000 de photos de grande résolution, 1000 films Divx.

Les nouveaux disques durs ont une capacité de 1voir 1,5 To