Chapitre 1 - Le codage numérique

Objectifs :

- ▷ Connaître le système binaire, le bit et l'octet
- ▷ Comprendre le codage des informations en informatique
- ▷ Savoir convertir dans les bases binaire, décimale ou hexadécimale
- ▷ Connaître le principe des couleurs à l'écran

1 Le binaire

L'informatique utilise des courants électriques, des aimantations, des rayons lumineux,..., *etc.* Chacun de ces phénomènes met en jeu 2 états possibles. D'un point de vue logique, on considère que ces 2 états correspondent à 2 valeurs ou 2 « **bits** » : 0 et 1.

C'est le « **système binaire** » utilisé pour coder tout type d'informations (textes, sons, images, vidéos...) : on parle alors de **numérisation**. Pour la machine, l'unité de stockage la plus petite n'est pas le bit mais l'octet.

A retenir !

L'octet (en anglais **Byte**) est une unité d'information composée de 8 bits. L'octet est utilisé pour mesurer la capacité de stockage en mémoire ou sur un disque dur.

1 Byte = 1 octet = 8 bits

Ainsi, on a les multiples de l'octet tels que :

- ▷ 1 Kilooctet (Ko) = 10^3 octets = 1 000 octets
- ▷ 1 Mégaoctet (Mo) = 10^6 octets = 1 000 ko = 1 000 000 octets
- ▷ 1 Gigaoctet (Go) = 10^9 octets = 1 000 Mo = 1 000 000 000 octets
- ▷ 1 Téraoctet (To) = 10^{12} octets = 1 000 Go = 1 000 000 000 000 octets
- ▷ 1 Pétaoctet (Po) = 10^{15} octets = 1 000 To = 1 000 000 000 000 000 octets
- ▷ 1 Exaoctet (Eo) = 10^{18} octets = 1 000 Po = 1 000 000 000 000 000 000 octets
- ▷ 1 Zéttaoctet (Zo) = 10^{21} octets = 1 000 Eo = 1 000 000 000 000 000 000 000 octets
- ▷ 1 Yottaoctet (Yo) = 10^{24} octets = 1 000 Zo = 1 000 000 000 000 000 000 000 000 octets

Ce qui correspond au tableau de correspondance suivant :

Nom	Symbole	Valeur
Kilooctet	Ko	10^3
Mégaoctet	Mo	10^6
Gigaoctet	Go	10^9
Téraoctet	To	10^{12}
Pétaoctet	Po	10^{15}
Exaoctet	Eo	10^{18}
Zettaoctet	Zo	10^{21}
Yottaoctet	Yo	10^{24}

Il existe déjà à l'heure actuelle des disques durs de plusieurs téraoctets.

2 Les supports de stockage

Les support de stockage ont évolué dans le temps pour répondre à la quantité croissante de données.

L'augmentation de la capacité tout en diminuant la dimension des supports est due à la miniaturisation des composants électroniques et l'augmentation de leur capacité de traitement qui a été exponentielle.

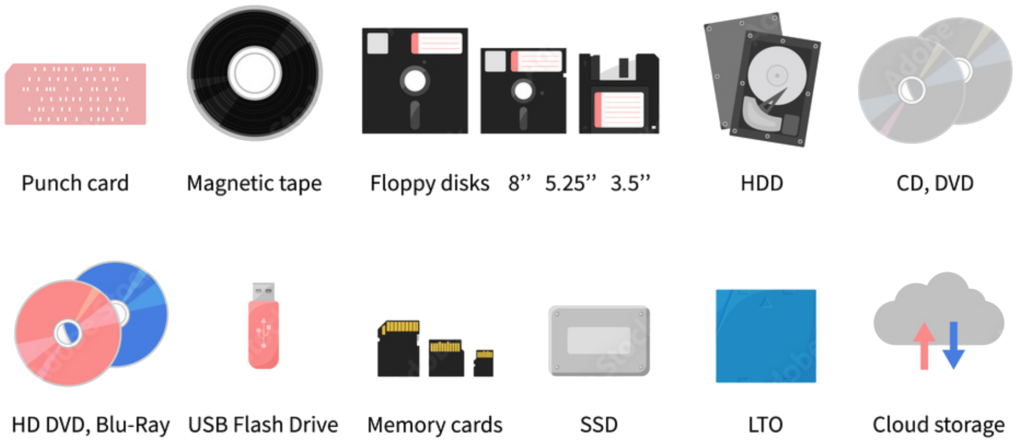


FIGURE 1 – Evolution des supports de stockage

Pour aller plus loin :

https://www.ina.fr/contenus-editoriaux/articles-editoriaux/la-saga-des-supports-de-stockage-de-donnees/

3 Conversion binaire - décimal

3.1 Conversion avec un tableau

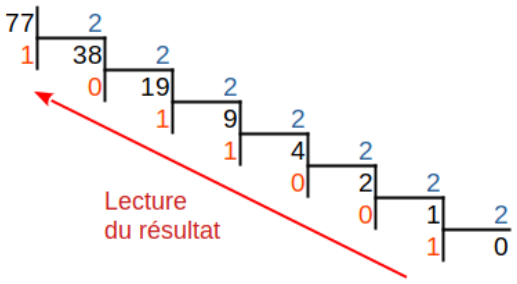
Pour convertir un nombre binaire en décimal, il faut additionner les puissances de 2 successives comme dans le tableau ci-dessous :

Puissances de 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valeur en décimal	128	64	32	16	8	4	2	1
Valeur binaire	0	0	0	1	1	0	1	0

Ainsi, on a : $(00011010)_2 = (26)_{10}$

3.2 Conversion par divisions successives

Pour convertir un nombre décimal en nombre binaire, on peut aussi utiliser la **méthode divisions entières successives par 2** jusqu'à ce que le quotient devienne nul. Le résultat sera la juxtaposition des restes dans le sens inverse.



Exemple : $(77)_{10} = (1001101)_2$

4 La base hexadécimale

Les nombres en binaires peuvent être très longs. On utilise donc souvent la base 16 pour les manipuler plus facilement.

4.1 Base 16

Il existes 16 *"chiffres"* hexadécimaux qui sont : `0 1 2 4 5 6 7 8 9 A B C D E F`

Voici leur correspondance avec la base 10 :

Décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadécimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Par exemple :

- A est un chiffre en hexadécimal. Il vaut 10 en décimal.
- F est un chiffre en hexadécimal. Il vaut 15 en décimal.

4.2 Notation

La notation dépendra du domaine d'utilisation.

Maths	Python	CSS
$(A3BC)_{16}$	`0xA3BC`	`#A3BC`

4.3 Hexadécimal - Binaire

Comme $16 = 2^4$, convertir un binaire en hexadécimal est relativement facile.

Chaque paquet de 4 bits donne un chiffre hexadécimal :

1010	0011	1011	1100
A	3	B	C

4.4 Hexadécimal - Décimal

Pour convertir $(4D5)_{16}$ de l'hexadécimal vers le décimal, on commence par le dernier chiffre :

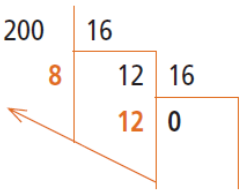
- 5×16^0 et on recule :
- $D \times 16^1 = 13 \times 16$ (D correspond au nombre 13)
- $4 \times 16^2 = 4 \times 256$

On obtient donc : $(4D5)_{16} = 4 \times 16^2 + 13 \times 16^1 + 5 \times 16^0 = (1237)_{10}$

4.5 Décimal - Hexadécimal

Comme pour la conversion décimal - binaire, on peut utiliser la méthode des divisions entières successives par 16 jusqu'à trouver 0. Les restes donnent les chiffres dans l'ordre inverse.

Par exemple :



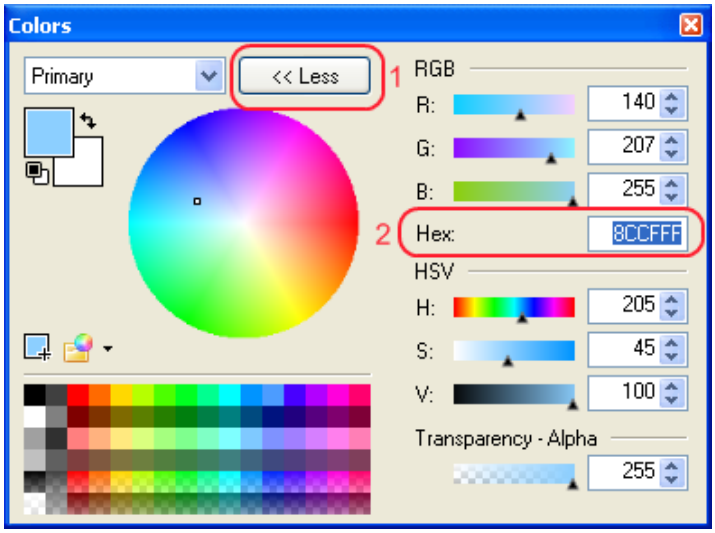
Ainsi, $(200)_{10} = 12 \times 16^1 + 8 \times 16^0 = C \times 16^1 + +8 \times 16^0 = (C8)_{16}$

Car C est un chiffre en hexadécimal. Il vaut 12 en décimal.

5 Couleurs

En informatique, c'est le système RGB (*Red, Green, Blue*) qui est utilisé dans la reproduction des couleurs sur les écrans d'ordinateur, les téléviseurs, les écrans de téléphone portable et autres dispositifs électroniques.

Il s'agit d'un modèle de synthèse additive qui permet de créer un large spectre de couleurs en combinant différentes intensités de lumière rouge, verte et bleue.



En synthèse additive on utilise 256 niveaux de couleur pour les composantes *Rouge*, *Vert* et *Bleu*. Chaque niveau de couleur est codé sur un octet codé en hexadécimal.

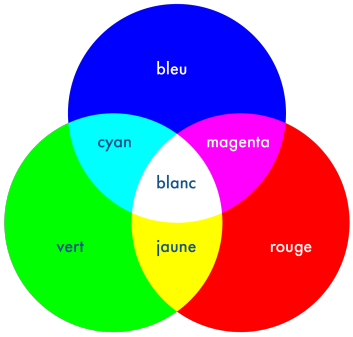
Par exemple ; la couleur #`FF0080` correspond à :

- `FF` : rouge à fond
- `00` : pas de vert
- `80` : bleu à moitié

On obtient un joli rose, noté parfois : `rgb(255, 0, 128)`

Quelques exemples que vous devez être capables de reconnaître :

- #FFFFFF correspond à blanc
- #000000 correspond à noir
- #FF0000 correspond à rouge
- #FFFF00 correspond à jaune
- #00FF00 correspond à vert
- #00FFFF correspond à cyan
- #0000FF correspond à bleu
- #FF00FF correspond à magenta



On rappelle que $(FF)_{16} = (255)_{10}$.

Dans ce modèle (1 octet par niveau de rouge / vert / bleu), on peut représenter : $256^3 = 16777216$ de couleurs !

6 Exercices

Exercice 1 : Convertir en binaire et en décimal les nombres hexadécimaux suivants :

1. `0x10`
2. `0xA0`
3. `0xFF`
4. `0xA0FA`

Exercice 2 : Convertir en hexadécimal les nombres suivants :

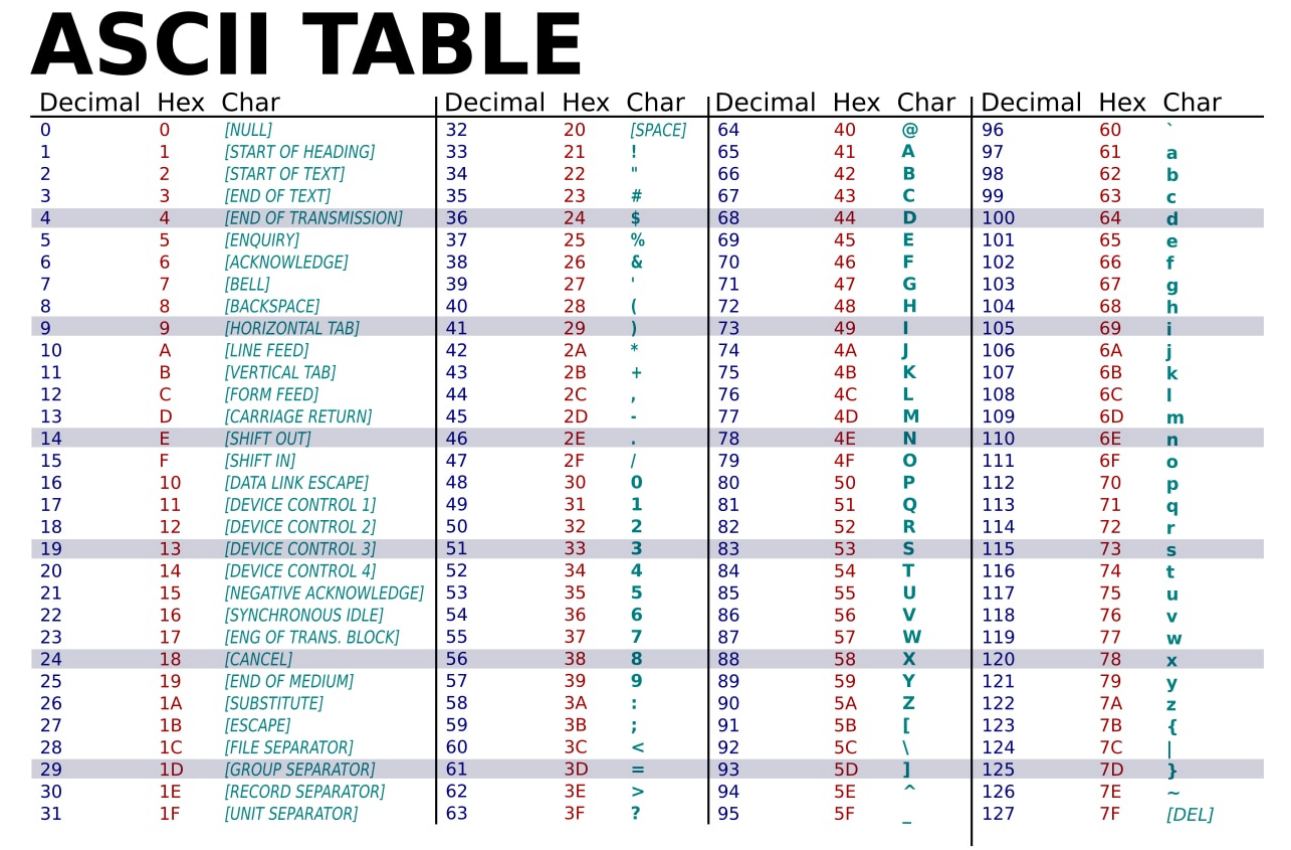
1. `10000000`
2. `11011100`
3. `11101001`
4. `10011010`

Exercice 3 : Convertir en hexadécimal les nombres suivants :

1. 80
2. 160
3. 128
4. 233

Exercice 4 : Grâce à la table ASCII ci-dessous, décoder le message suivant où chaque nombre représente un octet codé en hexadécimal.

Message : `4A 65 20 73 61 69 73 20 64 65 63 6F 64 65 72 20 61 76 65 63 20 75 6E 65 20 74 61 62 6C 65 20 41 53 43 49 49 2E`



ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]