Numération et codage

Table des matières

Aujourd'hui nos ordinateurs, téléphones et autres appareils savent manipuler aussi bien des nombres et du texte que des images, de la vidéo ou de la musique… Mais comment représenter, au sein d'un système numérique, cette diversité des objets du monde réel ou virtuel ?

1. DÉFINITION

1.1. UNITÉ DE CODAGE

Les composants constituant un système informatique réagissent, de manière interne, 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 adaptée à la représentation de tels signaux est **la base 2**, on parle alors de **codage binaire.**

L'unité de codage de l'information est un élément ne pouvant prendre que les valeurs 0 ou 1 ; le **bit**.

1.2. UNITÉ DE TRANSFERT

Pour les échanges de données, les informations élémentaires (bits) sont manipulées par groupes qui forment ainsi des mots binaires. La taille de ces mots est le plus souvent un multiple de 8 = 2^3.

L'unité de transfert utilisée pour les échanges de données est le mot de 8 bits appelé **octet** (byte en anglais).

Exemples : (2 octets)

1111 0011

1010 1111

Remarque :

Pour faciliter les manipulations, un octet peut être divisé en deux mots de 4 bits que l'on appelle des **quartets** : celui situé à gauche est le quartet de poids fort, **MSQ** (**M**ost **S**ignificant **Q**uartet), et celui situé à droite, le quartet de poids faible, **LSQ** (**L**ess **S**ignificant **Q**uartet).

Exemple :

MSQ							**LSQ**
1	**0**	**0**	**1**	**1**	**1**	**1**	**0**
quartet de poids fort				**quartet de poids faible**			
octet							

1.3. MOTS BINAIRES

Dans un mot binaire, le bit situé le plus à gauche est le bit le plus significatif, **MSB** (**M**ost **S**ignificant **B**it), celui situé le plus à droite est le bit le moins significatif, **LSB** (**L**ess **S**ignificant **B**it).

Exemple :

MSB															**LSB**
1	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
octet de poids fort								**octet de poids faible**							
mot (16 bits)															

Dans les notations de quantité binaires « kilo », « méga », … sont utilisés pour exprimer des

multiples en puissances de 2, mais cet usage est contraire aux normes SI (Système International).

ko	(kB)	=	kilo-octet	(kiloByte)	=	10^3	octets	=	1000	octets
Mo	(MB)	=	Méga-octet	(MegaByte)	=	10^6	octets	=	1000	ko
Go	(GB)	=	Giga-octet	(GigaByte)	=	10^9	octets	=	1000	Mo
To	(TB)	=	Téra-octet	(TeraByte)	=	10^{12}	octets	=	1000	Go
kio	(kiB)	=	kibi-octet	(kibiByte)	=	2^{10}	octets	=	1024	octets
Mio	(MiB)	=	Mébi-octet	(MebiByte)	=	2^{20}	octets	=	1024	ko
Gio	(GiB)	=	Gibi-octet	(GibiByte)	=	2^{30}	octets	=	1024	Mo
Tio	(TiB)	=	Tébi-octet	(TebiByte)	=	2^{40}	octets	=	1024	Go

(k, M, G, T, … = multiple du système international , b=bit, B=Byte, bi=binary)

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

En dehors de l'unité de transfert (octet), des regroupements plus importants sont couramment utilisés : **le mot de 16 bits** = 2 octets (word), **le mot de 32 bits** = 4 octets (double word), et **le mot de 64 bits** = 8 octets (quad word)…

1.4. EXERCICES D'APPLICATIONS

1. La fiche technique d'un disque dur indique une capacité de 320 GB.

   Exprimer cette capacité en Mio.

2. Votre FAI[1] vous annonce un débit descendant de 8 192 kibits/s.

   Vous faites une mesure de débit réel et vous trouvez une moyenne de 3 280 kibits/s.

   Quelle sera le temps théorique minimal de téléchargement d'une application de taille égale à 25 Mo ?

2. CODAGE

2.1. Base 10 ou décimale

Les nombres que nous utilisons habituellement sont ceux de la base 10 (système décimal).

Nous disposons de dix chiffres différents de 0 à 9 pour écrire tous les nombres.

D'une manière générale, toute base N est composée de N chiffre de 0 à N-1.

Soit un nombre décimal N = 2348. Ce nombre est la somme de 8 unités, 4 dizaines, 3 centaines et 2 milliers.

Nous pouvons écrire N = (2 x 1000) + (3 x 100) + (4 x 10) + (8 x 1)

$$2348 = (2 \times \mathbf{10^3}) + (3 \times \mathbf{10^2}) + (4 \times \mathbf{10^1}) + (8 \times \mathbf{10^0})$$

10 représente la base et les puissances de 0 à 3 le rang de chaque chiffre.

Quelque soit la base, le chiffre de droite est celui des unités.

Celui de gauche est celui qui a le poids le plus élevé.

1 Fournisseur d'Accès Internet

2.2. Base 2 ou binaire

Elle possède 2 symboles (0 et 1) appelés également **digits ou plus communément bits**[2].

Comme en base 10, à chaque bit du mot en base 2 correspond une puissance de 2

Puissances	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valeurs en décimal	128	64	32	16	8	4	2	1
exemple	0	1	1	0	1	1	1	0
Valeur en décimale correspondante	64 + 32 + 8 + 4 + 2 = 110							

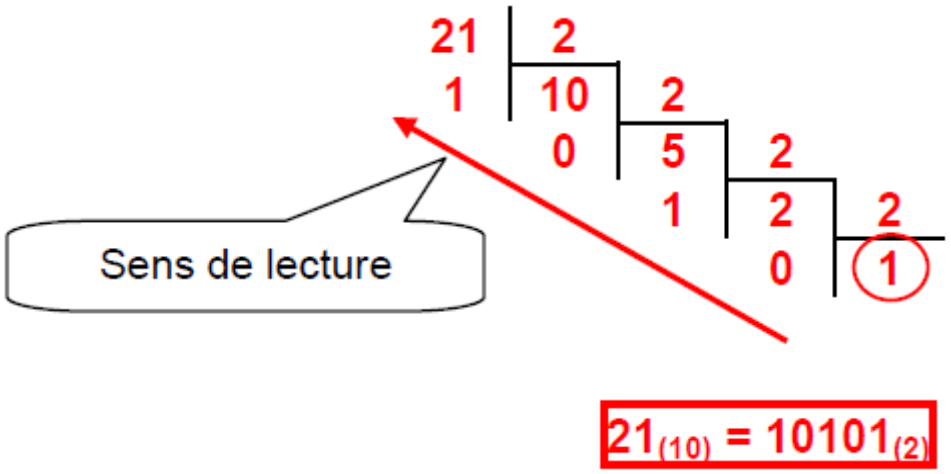
2.3. DÉCIMALE → BINAIRE

Des indices ou un préfixe peuvent être utilisés pour les nombres binaires : $110011_{(2)}$, $1101_{(BIN)}$, %111000.

Pour coder un nombre décimal en binaire, on peut utiliser plusieurs méthodes.

Exemple : codage des nombres $21_{(10)}$ et $30_{(10)}$ en binaire.

2.3.1. la division successive par 2



2.3.2. par soustractions successives

Par soustractions successives : on soustrait successivement la plus grande puissance de 2 et inférieure au nombre décimal à convertir.

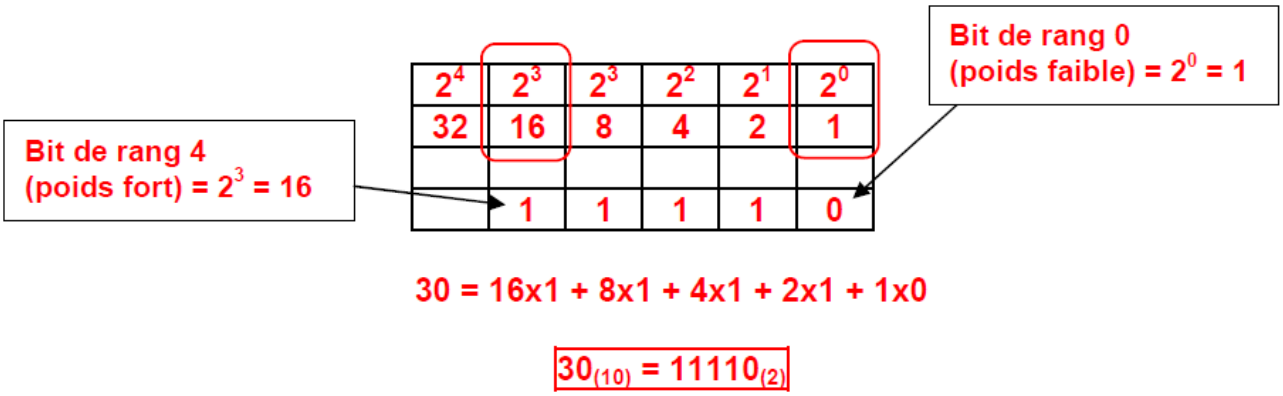
Par exemple 211 à convertir en binaire :

211 – 128 = 83 puis 83 – 64 = 19 puis 19 – 16 = 3 puis 3 – 2 = 1 reste 1

211 est donc composé de 128 + 64 +1 6 + 2 + 1 = %11010011

2 abréviation de **bi**nary digi**t**

2.3.3. le tableau



$30 = 16x1 + 8x1 + 4x1 + 2x1 + 1x0$

$30_{(10)} = 11110_{(2)}$

2.4. Base 16 ou HEXADÉCIMALE

Le binaire, s'il est très représentatif du codage interne des machines, reste très délicat et fastidieux à manipuler. Les programmeurs ont très vite ressenti la nécessité d'utiliser une représentation plus rapide des nombres binaires.

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

Dans le système hexadécimale les dix premiers symboles correspondent à ceux utilisés dans le système décimal : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9, et les six derniers correspondent aux premières lettres de l'alphabet latin : A, B, C, D, E et F, lesquelles valent respectivement 10, 11, 12, 13, 14 et 15 en base 10.

Puissances	16^4	16^3	16^2	16^1	16^0
Valeurs en décimal	65536	4096	256	16	1
exemple	0	0	0	6	B
Valeur en décimale correspondante	6 x 16 + 11 x 1 = 107				

2.5. DÉCIMALE → HEXADÉCIMALE

Des indices ou un préfixe peuvent être utilisés pour les nombres hexadécimaux : $15_{(16)}$, $23_{(HEX)}$, 0x55F, $AF4, &h38, #44B.

Remarque :

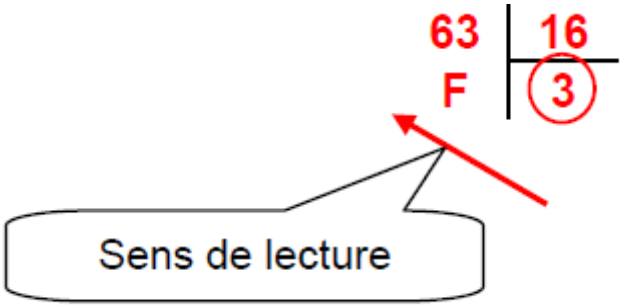
- Le préfixe 0x est utilisé dans le langage C, C++ et JAVA
- $ est utilisé dans le langage Pascal
- &h dans le langage Basic
- le # dans le HTML.

Une autre écriture courante est l'ajout du suffixe « h » à la fin du nombre (F15Ah par exemple).

Pour coder un nombre décimal en hexadécimale, on peut utiliser plusieurs méthodes.

Exemple : codage des nombres $63_{(10)}$ et $80_{(10)}$ en hexadécimale.

2.5.1. la division successive par 16



$63_{(10)} = 3F_{(16)}$

2.5.2. le tableau

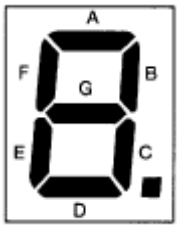
16^2	16^1	16^0
256	16	1
	5	0

$80 = 5 \times 16 + 0 \times 1$

$80_{(10)} = 50_{(16)}$

2.6. DÉCIMALE → BCD (OU DCB)

DCB signifie Décimal Codé en Binaire. Ce code est utilisé principalement pour les afficheurs 7 segments.



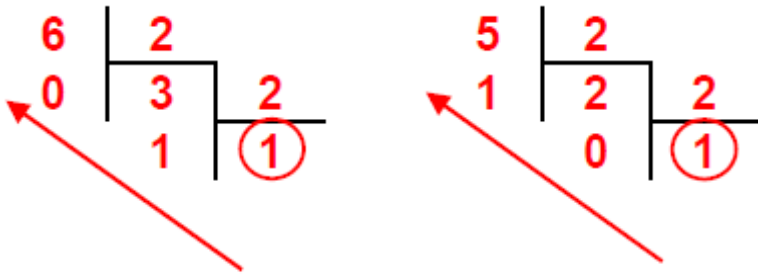
Il faut ici coder les chiffres décimaux individuellement afin d'obtenir pour chaques chiffres décimaux son équivalent codé en binaire sur 4 bits (quartet).

Décimal	Binaire
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Pour coder un nombre décimal en BCD, on peut utiliser plusieurs méthodes.

Exemple : codage des nombres $65_{(10)}$ et $78_{(10)}$ en BCD.

2.6.1. la division successive par 2



$$65_{(10)} = 01100101_{(BCD)}$$

2.6.2. le tableau

2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0
8	4	2	1	8	4	2	1
0	1	1	1	1	0	0	0

$$7 = 0x8 + 1x4 + 1x2 + 1x1$$
$$8 = 1x8 + 0x4 + 0x2 + 0x1$$

$$78_{(10)} = 01111000_{(BCD)}$$

3. Un peu d'histoire sur les encodages

3.1. Le code US-ASCII

Une grosse part des informations manipulées par les systèmes numériques concerne le langage parlé ou écrit matérialisé sous formes de textes, eux-mêmes constitués de caractères typographiques. Comment coder universellement ces caractères et permettre ainsi l'échange d'informations entre machines et/ou utilisateurs, quelle que soit la langue utilisée ?

Remarque :

Le morse inventé en 1844 est le premier codage à permettre une communication orientée caractère à longue distance. Ce code est composé de points et de tirets (une sorte de codage binaire).

SOS : ●●● ▬▬ ▬▬ ▬▬ ●●●

Le jeu de caractères codés ASCII[3] (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) ou code américain normalisé pour l'échange d'informations, est la norme de codage de caractères en informatique la plus connue, la plus ancienne et la plus largement compatible.

Le code ASCII est un code sur 7 bits (valeurs 0 à 127), il permet de définir :

- des caractères imprimables universels : lettres minuscules et majuscules, chiffres, symboles,…
- des codes de contrôle non imprimables : indicateur de saut de ligne, de fin de texte, codes de contrôle de périphériques, …

3 inventé par Bob BERNER en 1961

Binaire						b6	0	0	0	0	1	1	1	1
						b5	0	0	1	1	0	0	1	1
						b4	0	1	0	1	0	1	0	1
				Hexadécimal			0	1	2	3	4	5	6	7
b3	b2	b1	b0		Décimal		0	16	32	48	64	80	96	112
0	0	0	0	0	+0		NUL	TC7 (DEL)	SP	0	@	P	`	p
0	0	0	1	1	+1		TC1 (SOH)	DC1	!	1	A	Q	a	q
0	0	1	0	2	+2		TC2 (STX)	DC2	"	2	B	R	b	r
0	0	1	1	3	+3		TC3 (ETX)	DC3	#	3	C	S	c	s
0	1	0	0	4	+4		TC4 (EOT)	DC4	$	4	D	T	d	t
0	1	0	1	5	+5		TC5 (ENQ)	TC8 (NAK)	%	5	E	U	e	u
0	1	1	0	6	+6		TC6 (ACK)	TC9 (SYN)	&	6	F	V	f	v
0	1	1	1	7	+7		BEL	TC 10 (ETB)	'	7	G	W	g	w
1	0	0	0	8	+8		FE0 (BS)	CAN	(	8	H	X	h	x
1	0	0	1	9	+9		FE1 (HT)	EM	)	9	I	Y	i	y
1	0	1	0	A	+10		FE2 (LF)	SUB	*	:	J	Z	j	z
1	0	1	1	B	+11		FE3 (VT)	ESC	+	;	K	[	k	é
1	1	0	0	C	+12		FE4 (FF)	IS4 (FS)	,	<	L	\	l	ù
1	1	0	1	D	+13		FE5 (CR)	IS3 (GS)	-	=	M	]	m	è
1	1	1	0	E	+14		SO	IS2 (RS)	.	>	N	^	n	¨
1	1	1	1	F	+15		SI	IS1 (US)	/	?	O	_	o	DEL

Table ASCII sur 7 bits

EXERCICES D'APPLICATIONS

1. *Décrypter la chaîne ASCII ci-dessous représentée sous la forme d'une suite d'octets :*
   0011 0001 0101 0011 0100 1001
2. *Retrouver la chaîne ASCII sous la forme décimale du texte page suivante.*

3.2. Les encodages ISO-latin

Dans les années 1990, pour satisfaire les besoins des pays européens, ont été définis plusieurs encodages alternatifs, connus sous le nom de ISO-latin, ou encore ISO-8859. Idéalement, on aurait pu et certainement dû définir un seul encodage pour représenter tous les nouveaux caractères; mais entre toutes les langues européennes, le nombre de caractères à ajouter était

substantiel, et cet encodage unifié aurait largement dépassé 256 caractères différents, il n'aurait donc pas été possible de tout faire tenir sur un octet.

Mais on a préféré préserver la "bonne propriété" du modèle un caractère = un octet, ceci afin de préserver le code existant qui aurait sinon dû être retouché ou récrit.

Dès lors il n'y avait pas d'autre choix que de définir plusieurs encodages distincts; par exemple pour le français on a utilisé à l'époque ISO-latin-1; pour le russe ISO-latin-5.

4. UNICODE

Pour coder de manière universelle l'ensemble des symboles utilisés quelque soit la langue (anglais, français, grec, chinois,…) il faut attribuer à tout caractère ou symbole de n'importe quel système d'écriture de langue un nom et un identifiant numérique, et ce de manière unifiée, quelle que soit la plate-forme informatique ou le logiciel.

C'est ce qui a été fait par le standard Unicode qui définit 3 nouveaux encodages:

- UTF-8 : un encodage à taille variable, à base d'octets, qui maximise la compatibilité avec ASCII,
- UTF-16 : un encodage à taille variable, à base de mots de 16 bits
- UTF-32 : un encodage à taille fixe, à base de mots de 32 bits

Ces 3 standards couvrent le même jeu de caractères (113 021 tout de même dans la dernière version). Parmi ceux-ci le plus utilisé est certainement utf-8. Un texte ne contenant que des caractères du code US-ASCII initial peut être lu avec l'encodage UTF-8.

Exemples :

Chaque symbole d'écriture est représenté par un nom et une valeur hexadécimale préfixée par « **U+** ».

A « lettre majuscule latine A » **U+0041**

é « lettre minuscule latine e accent aigü » **U+00E9**

€ « symbole euro » **U+20AC**

Pour stocker sur un support informatique un texte constitué de caractères Unicode, il faut encore choisir un procédé transformant chaque définition Unicode en une suite d'octets et réciproquement… C'est le **processus d'encodage.**

Actuellement un des systèmes d'encodage couramment utilisés (Unix, Internet, …) est **UTF-8** (**U**nicode **T**ransformation **F**ormat).

5. DES IMAGES

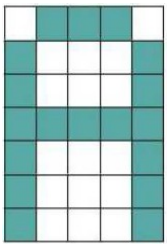
Après le texte, l'image est le support le plus utilisé pour communiquer.

Dans le domaine du numérique, les images sont constituées d'une **matrice L x H** (Largeur x Hauteur) de points élémentaires que l'on nomme généralement **des pixels**[4].

Chaque pixel a une couleur codée sur un nombre plus ou moins grand de bits.

Le périphérique de sortie (écran, imprimante, …) se doit de restituer les pixels de manière ordonnée en fonction de leur position respective (x, y) et de leur couleur.

Lettre pixelisée



Exemple :

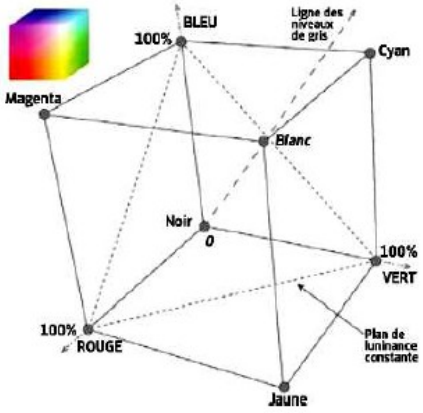
Le caractère ASCII $41 peut être simplement représenté sur une matrice 5 x 7 en « allumant » les pixels adéquats. La « couleur » peut-être ici matérialisée par un unique bit (pixel allumé ou éteint).

Le codage d'une image peut donc dans un premier temps se résumer en la succession des codages des pixels suivant un ordre bien défini (balayage lignes-colonnes en partant du coin supérieur gauche).

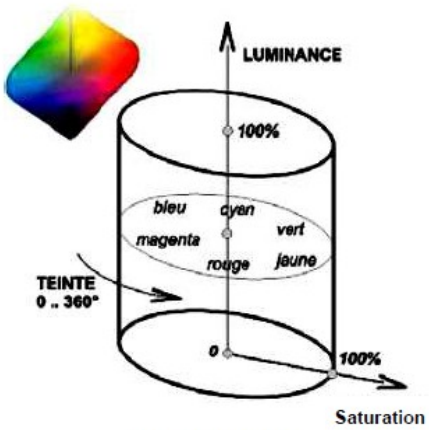
6. DE LA COULEUR

Le mode de représentation **RVB** (**R**ouge, **V**ert et **B**leu, ou en anglais RGB) correspond à celui fourni par la plupart des caméras couleur, il est naturellement utilisé pour la reproduction de couleurs sur écran (base noire). C'est le mode de composition des couleurs basé sur le principe des couleurs **additives** : le rouge, le vert et le bleu sont les trois primaires utilisés dans la constitution de couleurs à partir de sources lumineuses.

Synthèse additive



Modèle RVB



Modèle TSL

4 abréviation de Picture Element

Le modèle **TSL** (**T**einte-**S**aturation-**L**uminance) est un autre modèle plus proche de la perception humaine des couleurs. Ses coordonnées se calculent à partir des proportions RVB. La composition de la teinte et de la saturation est appelée **chrominance.**

Exemple : (codage RVB)[5]

Pixel blanc → codage RVB = $FFFFFF

7. DE LA PROFONDEUR D'IMAGE

Les images codées sur 24 bits sont dites en vraies couleurs (true color) ; une composante Alpha permettant d'inclure une information de transparence peut être ajoutée à ce type de codage, chaque pixel est alors codé sur 32 bits.

Le terme de profondeur d'image est utilisé pour spécifier le nombre de bits alloué au codage de chaque pixel ; les valeurs courantes de profondeur sont 1 (image binaire), 8 (256 couleurs ou niveaux de gris) et 32 (vraies couleurs avec canal alpha).

EXERCICES D'APPLICATIONS

1. *Calculer le poids (en octets) d'une image non compressée, de définition 640 x 480 et de profondeur 24 bits (RVB).*

   Calculer le gain d'espace réalisé en convertissant l'image en 256 niveaux de gris.

2. Un ordinateur affiche sans problème des images de définition 1024 x 768 et de profondeur 32 bits (RVBA).

   Que peut-on en déduire quant à la taille mémoire de la carte vidéo ?

5 http://primatice.net/logiciels/chromoweb/aide/codage.htm