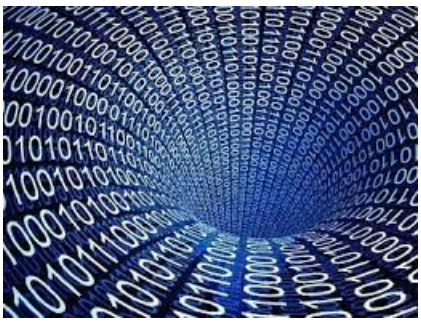
Numération et codage

Table des matières

1. Introduction....2
2. La représentation des entiers naturels....2
3. Codage....2
2.1. Base 2 ou binaire....2
2.1. Conversion Décimale → Binaire....2
2.1.1. La division successive par 2....3
2.1.2. Par soustractions successives....3
2.2. Base 16 ou hexadécimale....3
2.3. Conversion décimale → hexadécimale....4
2.3.1. La division successive par 16....4
2.3.2. Par soustractions successives....5
2.4. DCB....5
3. Unité de transfert....5
4. Un peu d'histoire sur les encodages....6
4.1. Le code US-ASCII....6
4.2. Les encodages ISO-latin....7

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 ?

Quelles sont les techniques utilisées pour représenter numériquement les grandeurs qui nous entourent ?



1. Introduction

La mémoire des ordinateurs est constituée d'une multitude de petits circuits électroniques qui ne peuvent être, chacun, que dans deux états conventionnellement appelés 0 et 1, mais on aurait pu tout aussi bien les appeler faux et vrai. Une telle valeur, 0 ou 1, s'appelle un booléen, un chiffre binaire ou encore un bit (binary digit). Un tel circuit à deux états s'appelle un circuit mémoire un bit et se décrit donc par le symbole 0 ou par le symbole 1.

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). L'état d'un circuit, composé de plusieurs de ces circuits mémoire un bit, se décrit par une suite finie de 0 et de 1, que l'on appelle un mot.

2. La représentation des entiers naturels

On écrit un entier naturel en notant, de droite à gauche, le nombre d'objets isolés, le nombre de paquets de dix, le nombre de paquets de cent, le nombre de paquets de mille, etc. Chacun de ces nombres étant compris entre zéro et neuf, seuls dix chiffres sont nécessaires : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9. Par exemple, l'écriture 2359 exprime un entier naturel formé de 9 unités, 5 dizaines, 3 centaines et 2 milliers.

Soit $2359 = 2.10^3 + 3.10^2 + 5.10^1 + 9.10^0$

Le choix de faire des paquets de dix est peut-être dû au fait que l'on a dix doigts, mais on aurait pu tout aussi bien décider de faire des paquets de deux, de huit, de seize, etc. On écrirait alors les nombres entiers naturels en notation à position en base deux (binaire), huit (octal) ou seize (hexadécimal). La notation décimale à position s'appelle donc aussi la notation à position en base dix.

3. Codage

2.1. Base 2 ou binaire

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

Des indices ou un préfixe peuvent être utilisés pour les nombres binaires : $01101110_{(2)}$ ou %01101110.

Le mot de 8 bits %01101110 = $0.2^7 + 1.2^6 + 1.2^5 + 0.2^4 + 1.2^3 + 1.2^2 + 1.2^1 + 0.2^0$ en décimale.

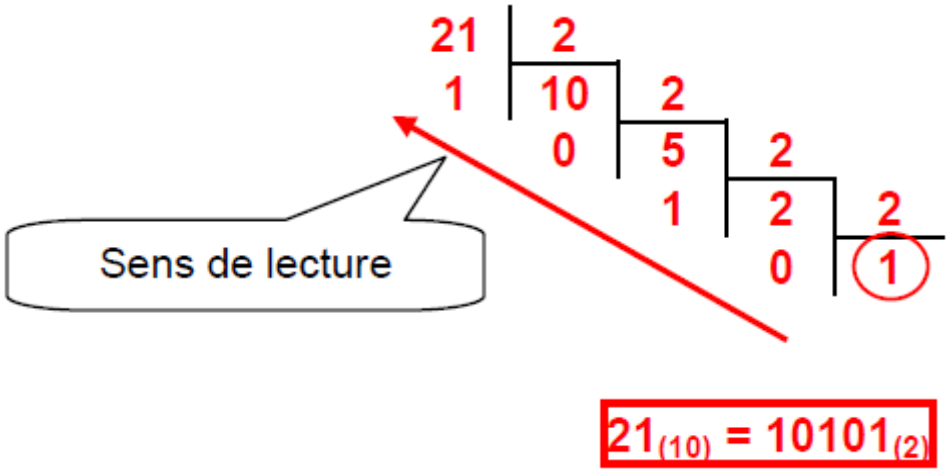
Puissances	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valeurs en décimale	128	64	32	16	8	4	2	1
exemple	0	1	1	0	1	1	1	0
Valeur en décimale correspondante	64 + 32 + 8 + 4 + 2 = 110							

2.1. Conversion Décimale → Binaire

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

Exemple : codage du nombre $21_{(10)}$ en binaire.

2.1.1. La division successive par 2



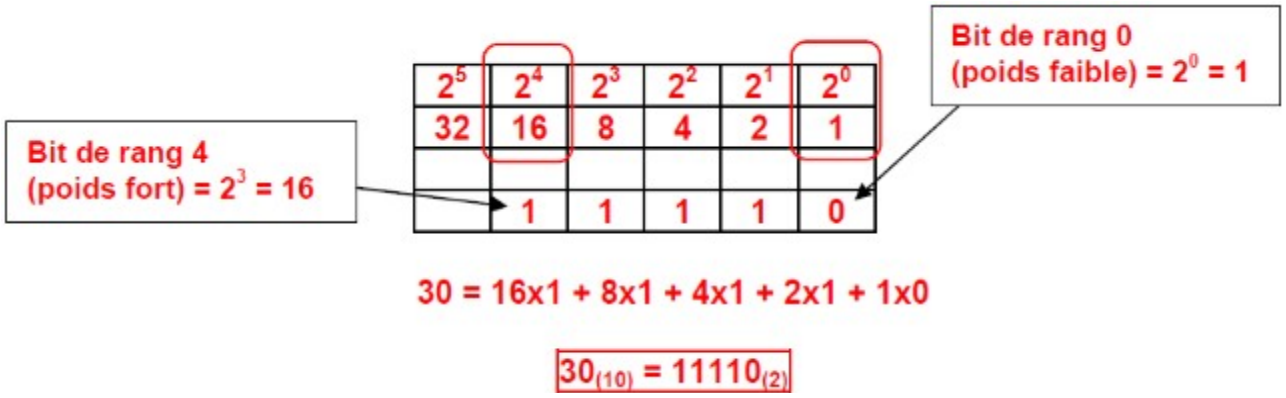
2.1.2. Par soustractions successives

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

Par exemple 30 à convertir en binaire :

30 – 16 = 14 puis 14 – 8 = 6 puis 6 – 4 = 2 puis 2 – 2 = 0 reste 0

30 = 16 + 8 + 4 + 2 + 0 = %11110



2.2. 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 à l'aide d'une base 16 qui permet de représenter 2 mots de 8 bits (2 octets).

Comme en base 10, un chiffre du mot en base 16 correspond à une puissance de 16.

Des indices ou un préfixe peuvent être utilisés pour les nombres hexadécimaux : $21_{(16)}$, 0x21, #21 ou $21.

Remarque :

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

Le nombre 0x21 = $2.16^1 + 1.16^0$ en décimale.

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.

À chaque quartet du mot binaire correspond un symbole en hexadécimal

Exemple : codage du nombre 0x6B en décimale.

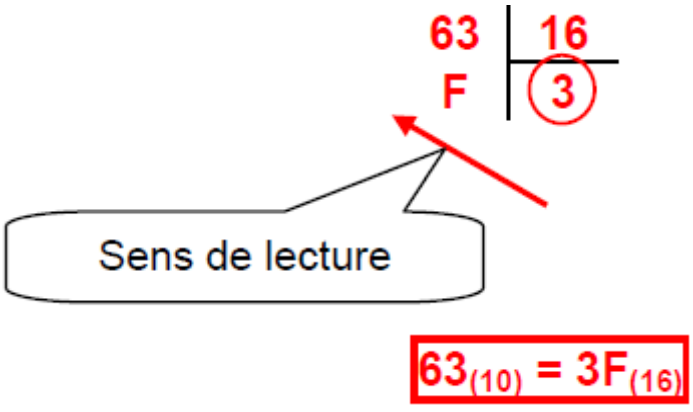
Puissances	16^4	16^3	16^2	16^1	16^0
Valeurs en décimale	65536	4096	256	16	1
exemple	0	0	0	6	B
Valeur en décimale correspondante	$6 \times 16^1 + 11 \times 16^0 = 107$				

2.3. Conversion décimale → hexadécimale

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

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

2.3.1. La division successive par 16

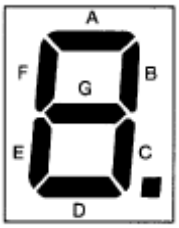


2.3.2. Par soustractions successives

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

$$80 = 5 \times 16 + 0 \times 1$$
$$80_{(10)} = 50_{(16)}$$

2.4. 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 chaque chiffre décimal 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

3. 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).

Exemple : 1010 1110

MSB							**LSB**
1	**0**	**1**	**0**	**1**	**1**	**1**	**0**
octet							

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).

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)…

4. Un peu d'histoire sur les encodages

4.1. 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.

Le jeu de caractères codés ASCII[1] (**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, …

1 inventé par Robert William Bemer en 1960

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]

Table ASCII sur 7 bits

4.2. 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.