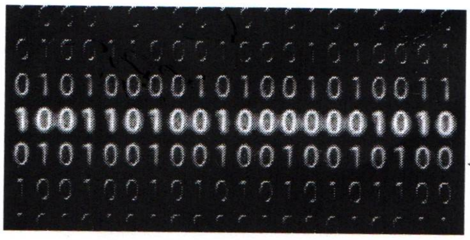
Chapitre 7 - Codage nombres, caractères, images, sons, vidéos

On voit dans ce chapitre comment sont numérisés :

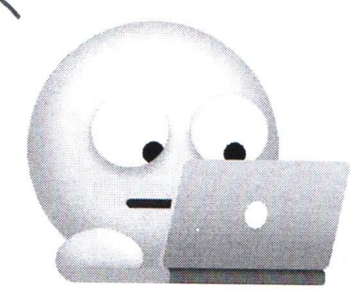


- Les nombres entiers relatifs de l'ensemble $\mathbb{Z}$,
- Les nombres réels de l'ensemble $\mathbb{R}$,
- Les caractères : lettres des différentes langues, symboles, …
- Les images, sons et vidéos.

1
2,58
3,14159
2023

Aa Bb Cc Dd Ee Ff
Gg Hh Ii Jj Kk Ll
Mm Nn Oo Pp Qq
Rr Ss Tt Uu Vv Ww
Xx Yy Zz

Аа Бб Вв Гг Дд Ее
Жж Зз Ии Йй Кк
Лл Мм Нн Оо Пп
Рр Сс Тт Уу Фф Хх
Цц Чч Шш Щщ Ъъ
Ьь Юю Яя



1- Comment on numerise des nombres entiers relatifs ?

a. Methode du bit signe :

La première idée qui vient à l'esprit est de réserver le bit le plus à gauche pour identifier le signe du nombre (bit à 0 pour + et à 1 pour −) . Par exemple, pour un codage des nombres 57 et −57 , sur 1 octet, on aurait : format imposé

Codage de $(+57)_{10}$:	Codage de $(-57)_{10}$:
0 011 1001 (↑ +)	1 011 1001 (↑ −)

Si le codage est fait sur 1 octet, comme 1 bit sur les 8 est réservé au signe, seul 7 bits sont exploitables pour numériser la valeur absolue du nombre. Ainsi il ne sera que possible de coder des nombres entre −127 et +127.

L'avantage de cette méthode est qu'il est facile de faire la correspondance avec la base 10. Par contre, l'inconvénient apparait lorsque l'on additionne un nombre avec son opposé. Par exemple, l'opération (+57) + (−57) réalisée en binaire donne :

```
        1 11    1
+57   0011 1001
 +
-57   1011 1001
---------------
     1111 0010  -> -114
```

Le résultat de cette opération ne donne donc pas le nombre 0. Comme les nombres sont souvent utilisés pour réaliser des opérations mathématiques, cet inconvénient devient majeur. Cette méthode, dite du « *bit signé* », n'est ainsi pas utilisée.

b. METHODE DU COMPLEMENT A 2 : On impose le format 8, 16, 32, 64 bits

On présente cette technique dite du « *complément à 2* », pour un codage sur 1 octet, des nombres *57* et *-57*. Comme pour la méthode précédente, sur 1 octet, il ne sera possible de coder uniquement des nombres compris entre -127 et $+127$.

Codage de $N = (+57)_{10}$:	Codage de $(-57)_{10}$:
Entier relatif positif idem nombre entier avec le format imposé. 0011 1001	- Complément à 1 de $(+57)_{10}$ noté $\overline{\overline{N}}$: NON 0011 1001 -> 1100 0110 = N - Complément à 1 de $(+57)_{10}$ noté $\overline{\overline{N}} + 1$: N+1 = C2 : 1100 0110 + 1 = 1100 0111 - -> 1100 0111 - Le codage de $(-57)_{10}$ est : $(1100\ 0111)_{C2} = (-57)_{10}$

Avec cette méthode, l'opération $(+57) + (-57)$ réalisée en binaire donne :

```
            1 1 11  11 1
+57         0 0 11  10 0 1
 +
-57         1100    0111
-------------------------
        -> 1|0000   0000
```

dépasse le format n'est pas pris en compte.

Si on veut faire la correspondance entre la notation binaire d'un nombre entier relatif et sa notation en base 10, on procède de la manière suivante :

Codage du nombre $(0011\ \ 1001)_2$ sur 1 octet :	Codage du nombre $(1100\ \ 0111)_2$ sur 1 octet :
0011 1001 + donc $(57)_{10}$	1100 0111 - $-128 + 64 + 4 + 2 + 1 = (-57)$

2- COMMENT ON NUMERISE DES NOMBRES REELS ?

a. CAS DES NOMBRES DECIMAUX :

En notation décimal, les chiffres à gauche de la virgule représentent les unités, les dizaines, centaines etc ... et ceux à droite de la virgule, les dixièmes, les centièmes etc.. Par exemple, le nombre décimal 526,73 peut se décomposer ainsi :

$526,73 = 5\times10^2 + 2\times10^1 + 6\times10^0 + 7\times10^{-1} + 3\times10^{-2}$

De la même manière en binaire, les chiffres à droite de la virgule représentent les demis, les quarts, les huitièmes, les seizièmes etc ... Par exemple, le nombre décimal 11,375 peut se décomposer ainsi :

2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
8	4	2	1	0,5	0,25	0,125
1	0	1	1	, 0	1	1

On convertit facilement un nombre décimal de la base 2 vers la base 10. Par exemple, la conversion en base 10 du nombre $(111{,}101)_2$ donne :

4	2	1	0,5	0,250	0,125
1	1	1	, 1	0	1
	7		, 6	25	

Par contre, la conversion de la base 10 à la base 2, ne peut pas toujours être réalisée de manière exacte. Par exemple, pour la conversion sur 1 octet du nombre décimal $(0{,}3)_{10}$, on aurait :

2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
1	0,5	0,25	0,125	0,0625	0,03125	0,015625	0,0078125
0 ,	0	1	0	0	1	1	0

Méthode de conversion d'un nombre en base 10 décimal en base 2

ex $(5,375)_{10}$

- partie entière $5 \rightarrow 101$
- partie décimale

$0,375 \times 2 = \boxed{0},750$

$0,750 \times 2 = \boxed{1},500$

$0,5 \times 2 = \boxed{1}.$

on garde la partie décimale

la partie décimale de 1 est nulle donc on s'arrête ici

$\Rightarrow (5,375)_{10} = (101,011)_2$

ex $0,3$ $(0,3)_{10} =$

$0,3 \times 2 = \boxed{0},6$

$0,6 \times 2 = \boxed{1},2$

$0,2 \times 2 = \boxed{0},4$

$0,4 \times 2 = \boxed{0},8$

$0,8 \times 2 = \boxed{1}6$

$0,6 \times 2 = \boxed{1}2$

$0,0\,1001\,1001\ldots$

Les nombres tels que 0,1 et 0,3 ne sont pas représentables exactement en binaire. Ils ont une représentation tronquée en machine, ce qui introduit des erreurs dans les opérations :

```
>>> 0.2 +0.1
0.30000000000000004
```

b. NOTATION SCIENTIFIQUE

Pour représenter de manière générale les nombres réels on utilise donc une valeur approchée. Cette valeur approchée s'inspire de la notation scientifique. Par exemple la notation scientifique décimale des nombres 53,47 et $-0{,}0382$ est :

$53{,}47 \rightarrow +5{,}347 \times 10^{1}$

$-0{,}0382 \rightarrow -3{,}82 \times 10^{-2}$

En notation scientifique décimale, un nombre est représenté sous la forme : s ***m*** x 10^n où :

- s : est le signe du nombre + ou -
- m est un nombre réel tel que $1 \leq m < 10$ il s'appelle la **mantisse**
- n est un entier relatif il s'appelle **l'exposant**

En notation scientifique binaire un nombre est représenté également sous la forme : s ***m*** x 2^n où :

- s : est le signe du nombre + ou -
- m est un nombre réel tel que $1 \leq m < 2$ il s'appelle la **mantisse**
- n est un entier relatif il s'appelle **l'exposant**

Par exemple, la notation scientifique binaire des nombres suivants est :

- $(+57)_{10} = (0011\ 1001)_2$ $\quad 1{,}11001 \times 2^5$
- $(+0{,}3)_{10} = (0{,}010011001)_2$ $\quad 1{,}0011001 \times 2^{-2}$

Cette écriture est appelée virgule flottante car la virgule peut flotter de gauche à droite. Un réel est donc représenté en machine par un nombre décimal d'un type particulier : les nombres flottants (floats).

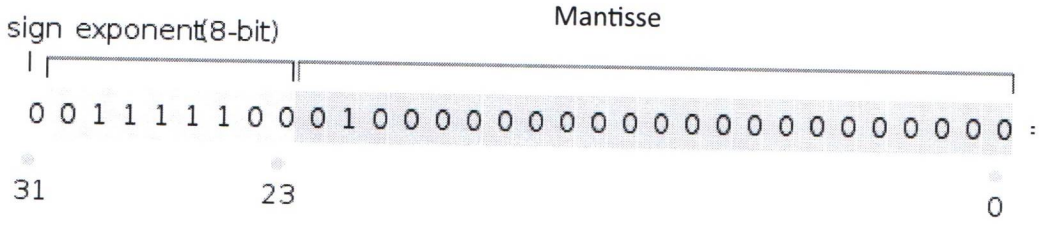
c. REPRESENTATION EN MACHINE :

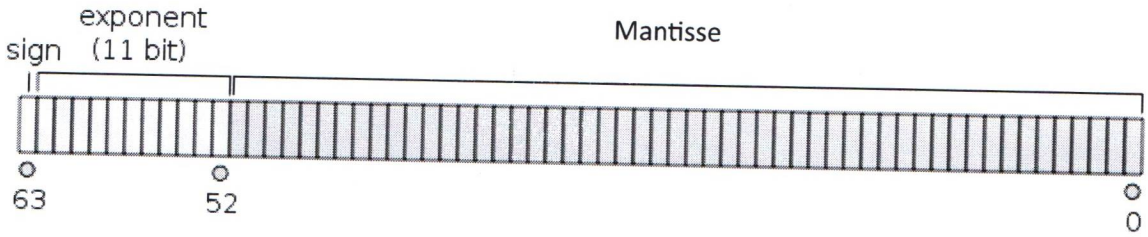
A la fin des années 70, chaque ordinateur possédait sa propre représentation pour les nombres à virgule flottante. En 1985 la norme IEEE 754 est apparue pour normaliser le codage des nombres.

Il existe deux formats associés à cette norme : le format dit "simple précision" et le format dit "double précision". Le format "simple précision" utilise 32 bits pour écrire un nombre flottant alors que le format "double précision" utilise 64 bits. Dans la suite nous travaillerons principalement sur le format 32 bits.

Que cela soit en simple précision ou en double précision, la norme IEEE754 utilise :

- 1 bit de signe (1 si le nombre est négatif et 0 si le nombre est positif)
- des bits consacrés à l'exposant (8 bits pour la simple précision et 11 bits pour la double précision)
- des bits consacrés à la mantisse (23 bits pour la simple précision et 52 bits pour la double précision)





Nous pouvons vérifier que l'on a bien 1 + 8 + 23 = 32 bits pour la simple précision et 1 + 11 + 52 = 64 bits pour la double précision.

⇨ Pour comprendre comment cela fonctionne, on commente la numérisation du nombre $(+57)_{10} = (0011\ 1001)_2$:

5 18

11001 000000000000000000

$\leftarrow 1,11001 \times 2^5$

0x42640000 = 01000010 01100100 00000000 00000000

0 5

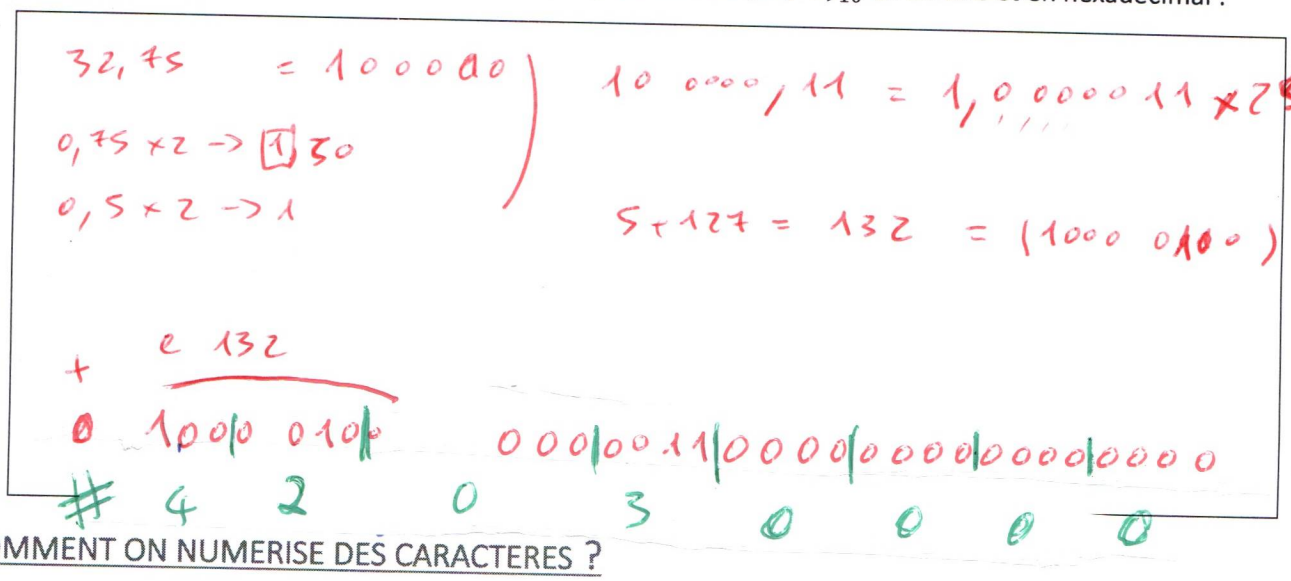
$5 + 127 \rightarrow 132 = 128 + 4$

⇨ On commente de même la conversion du nombre $(+0,3)_{10} = (0,010011001)_2$:

0x3E99999A = 00111110 10011001 10011001 10011010

d. EXERCICE :

Déterminer la représentation au format simple précision de $(32{,}75)_{10}$ en binaire et en hexadécimal :



3- COMMENT ON NUMERISE DES CARACTERES ?

a. CODE ASCII :

L'***American Standard Code for Information Interchange*** (Code américain normalisé pour l'échange d'information), plus connu sous l'acronyme **ASCII** ([aski:]), est une norme informatique de codage de caractères apparue dans les années 1960. C'est la norme de codage de caractères la plus influente à ce jour. Le tableau ci-dessous permet de retrouver le codage en hexadécimal de caractères sur 1 octet. L'unité du nombre hexa est donné par la colonne, la ligne donne le second chiffre :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STH	ETH	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	CD2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8	€	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	Ž	□
9	□	'	'	"	"	•	–	—	˜	™	š	›	œ	□	ž	Ÿ
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Par exemple :

- Le caractère 'a' est codé de la manière suivante : $(61)_{16} = (0110\ 0001)_2$
- Le caractère 'A' est codé de la manière suivante : $(41)_{16} = (0100\ 0001)_2$