Système de numération (entiers signés, Représentation des nombres réels, codage BCD, codage Gray)

1/Entiers signés

Ce sont des nombres possédant un signe + ou -, Il existe 3 méthodes pour les représenter :

.a/ Signe et valeur absolue (SVA)

Si un nombre est représenté sur n bits, alors la valeur absolue du nombre est codée sur (n-1) bits et le signe est codé sur le bit du poids fort, par convention 0 représente un nombre positif et 1 un nombre négatif tels que :

- L'intervalle des nombres représentables sur n bits en SVA est : $[-(2^{n-1}-1), +(2^{n-1}-1)]$
- Le nombre de représentations possibles sur n bits est 2^n.

Cette représentation parait simple mais elle souffre de problème de complication des opérations arithmétiques.

b /Complément à 1 (C1)

Dans cette représentation : le premier bit est réservé pour le signe et si le nombre est positif alors il garde son format, sinon (il est négatif) alors chaque bit est inversé (0 devient 1 et 1devient 0).

Tels que :

- L'intervalle des nombres représentables sur n bits en C1 est : $[-(2^{n-1}-1), +(2^{n-1}-1)]$
- Le nombre de représentations possibles sur n bits est 2^n.

Exemple : $(-5)_{10}$→ $(1101)_2$ en SVA→$(1010)_2$ en C1

Addition (soustraction) en C1

Elle se base sur le principe suivant :

Si aucune retenue n'est générée par le bit de signe, le résultat est correct, et il est représenté en C1

Sinon, elle sera enlevée et additionnée au résultat de l'opération, celui-ci est représenté en C1

exemple1 : $+13-4=+13+(-4)=(01101)_{SVA}+(10100)_{SVA}=(01101)_{C1}+(11011)_{C1}=(101000)_{C1}$
on remarque une retenue générée par le bit de signe donc elle sera ajoutée
:$(01000+1)_{C1}=(01001)_{C1}$

.c/ Complément à 2 (C2)

Dans cette représentation : le premier bit est réservé pour le signe, et si le nombre est positif alors il garde son format, sinon (il est négatif) il est transformé en C1 puis ajouté a 1.

Le nombre de représentations possibles sur n bits est 2^n. Ainsi que l'intervalle des nombres représentables sur n bits en C2 est : $[-(2^{n-1}), +(2^{n-1}-1)]$

Exemple : $(-5)_{10}$→ $(1101)_2$ en SVA→$(1010)_2$ en C1→$(1011)2$ en C2

Addition en complément à 2

Elle se base sur le principe suivant :

- S'il y a une retenue générée par le bit de signe, elle est ignorée et le résultat est en C2
- Sinon le résultat est correct et il est représenté en C2.

Exemple1 : $+13-4=+13+(-4)=(01101)_{SVA}+(10100)_{SVA}=(01101)_{C2}+(11100)_{C2}=(101001)_{C2}$

on remarque une générée par le bit de signe don il sera négligée : $(101001)_{C2}=(01001)_{C2}$
C'est un nombre positif donc sa forme en C2=sa forme naturelle=$(+1001)_2=(+9)_{10}$
Exemple 2 : $-13+4=(11101)_{SVA}+(00100)_{SVA}=(10011)_{C2}+(00100)_{C2}=(10111)_{C2}$
On remarque qu'il n'y a pas de retenue générée par le bit de singe et le résultat est un nombre négatif en C2 pour le transformer en forme naturelle il faut remplacer le 1 (bit de signe) par le signe (-) puis soustraire un 1 (devient en C1) puis inverser les bits,
$(10111)_{C2}=(10110)_{C1}=(-1001)_2=(-9)_{10}$

2/Représentation des nombres réels

Les formats de représentations des nombres réels sont :

a/Format virgule fixe

Utilisé par les premières machines, possède une partie 'entière' et une partie 'décimale' séparées par une virgule. La position de la virgule est fixe d'où le nom.
Exemple : 54,25(10) ; 10,001(2) ; A1,F0B(16)

b/ Format virgule flottante

Utilisé actuellement sur machine, défini par : $\pm m \cdot b^{\wedge}e$

- un signe + ou –
- une mantisse m (en virgule fixe)
- un exposant e (un entier relative)
- une base b (2,8,10,16,…)

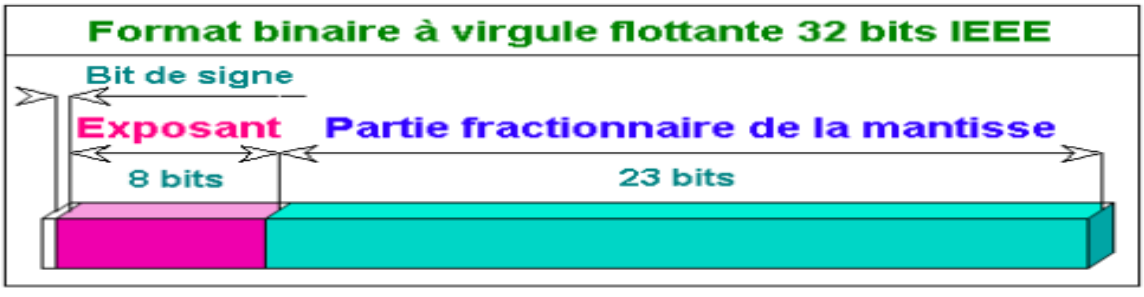
Exemple : $(3,25)_{10}=(32,5x10^{-1})_{10}=(325x10^{-2})_{10}=(0,325x10^{1})_{10}$

0,25x2=**0**,5
0,5x2=**1**,0
Donc $(3,25)_{10}=(11,01)_2$ (en virgule fixe)
$=(1,101x2^1)_2=(0,1101x2^2)_2=(110,1x2^{-1})_2$ (en virgule flottante)
Plusieurs manières de représenter un nombre réel en virgule flottante (différentes valeurs de m et e)
Donc il faut une **normalisation**

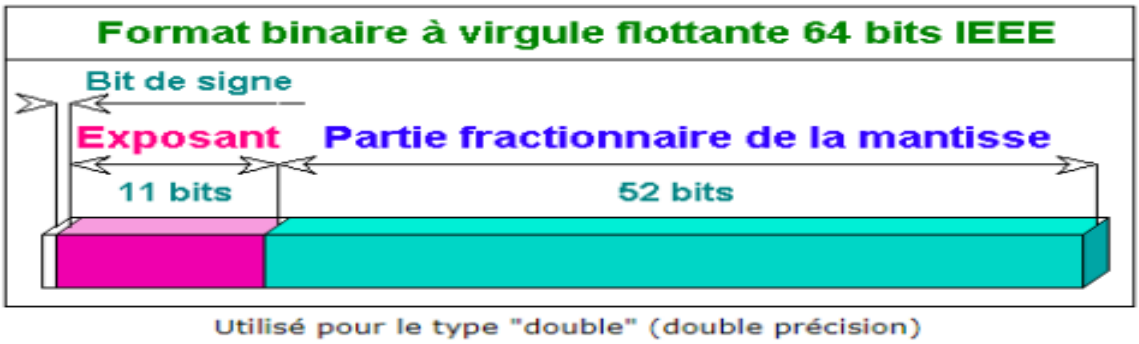
c/ Codage en Virgule Flottante Normalisé

IEEE 754 décrit de nombreux formats pour représenter un nombre à virgule flottante, les plus connus sont :

- **Simple Précision Single Precision (32 bits)**
- **Double Précision Double Precision (64 bits)**



Utilisé pour le type "float" (simple précision)



Simple Précision :

•**Bit de signe(**The sign bit) : peut avoir soit la valeur 1 si le nombre est négatif , ou bien 0 si le nombre est positif.

•**L'exposant (**The exponent) codé sur 8 bits et contient la valeur de la puissance qui élève le nombre 2.

•**La mantisse (**mantissa) codé sur 23 bits et contient la partie fractionnaire(fraction part) du nombre écrit avec la notation scientifique.

Exemple :

- 0,75 en simple précision

  0,75 x 2 = **1**, 5 ; 0, 5 x 2 = **1**, 0

- 0,75 → **0,11**

- soit **1, 1 x 2^{-1}** en **forme normalisée**

  => $(-1)^S$ x (1+. 1000 0000 0000 0000 0000 0000) x 2^{-1+127}

  En simple précision :

  1 |01111110 |10000000000000000000000

	Simple précision sur 32 Bits	**Double précision Sur 64 bits**
Bit de signe	1	1
Exposant	8	11
Mantisse	23	52
Codage de l'exposant	Décalage 127	Décalage 1023

3/Le code BCD (Binary Coded Decimal)

Certaines applications, notamment en gestion, exigent des calculs décimaux exacts, sans arrondi, ce qui implique de travailler avec des nombres décimaux. En effet, avec un nombre

fixé de bits, il est impossible de convertir de manière exacte des nombres binaires en nombres décimaux et réciproquement.
D'autre part de nombreuses applications visualisent ses données sous la forme d'un affichage décimal, exemple : L'affichage dans un ascenseur du N° d'étage ; L'affichage de la vitesse d'un véhicule, On utilise alors la représentation BCD, dans laquelle chaque chiffre décimal (0 - 9) est codé avec 4 chiffres binaires,

Chiffre décimal	Code BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Addition en BCD :

L'opération de l'addition se fait en deux étapes :
- Calculer individuellement la somme de chaque paire de groupe de quatre bits et sans porter la retenue au prochains groupe de bits.
- Corriger (de droit à gauche) chaque groupe de quatre bits dépassant le 1001 par ajout de 110, si le groupe dégage une retenue elle sera envoyée au prochain groupe.

Exemple : 39+58

```
      0011  1001
+     0101  1000
=  ---------------
      1000  10001
+        1  0110
=  ---------------
      1001  0111
```

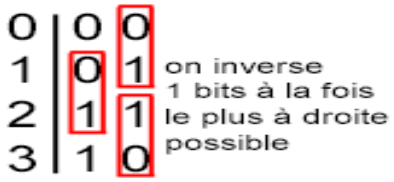
4/Codage Binaire réfléchi (code de GRAY)

Le codage binaire réfléchi, également connu sous le nom de code de Gray.

Un système de codage binaire dans lequel deux chiffres consécutifs ne diffèrent que d'un seul bit.

Construction du code Gray

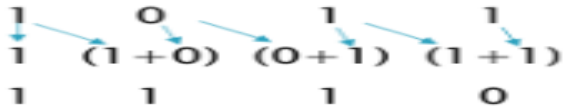
Ce processus de création d'un code de Gray en utilisant une méthode de réflexion par miroir(using the mirror reflection method) est une façon de générer le code de Gray.



Construction du code Gray : nous avons inversé 1 bits à la fois.

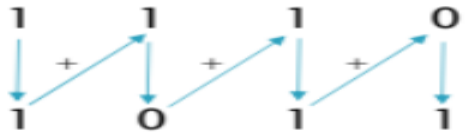
Conversion binaire pur vers binaire réfléchi :

(1011)2 = (?)GR = (1110)GR



Conversion binaire réfléchi vers binaire pur :

(1110)GR = (?)2 = (1011)2



5/Codage des caractères : Table ASCII

Le code ASCII Autrement dit : **American Standard Code for Information Interchange**.

C'est un codage sur 7 bits ce qui donne 128 codes différents :

• 0 à 31 : caractères de contrôle (retour à la ligne, Bip sonore, etc.…)

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]*