

Chapitre 1: Systèmes de numération et Codes

Plan du Chapitre

- Objectif du chapitre
- Systèmes de numération
- Changement de base
- Opérations arithmétiques binaires
- Opérations arithmétiques hexadécimales
- Les codes

Objectif du chapitre

- Ce chapitre traite en détail les différents systèmes de numération : **systèmes décimal, binaire, octal et hexadécimal** ainsi que les méthodes de **conversion** entre ces systèmes de numération.
- Nous étudierons également les **opérations arithmétiques** sur les nombres binaires signés, après avoir
- introduit la notion du **complément à 2** d'un nombre binaire. Nous terminons ce chapitre par l'étude de **plusieurs codes numériques** tels que les codes **B.C.D, Gray et A.S.C.I.I**

Systèmes de numération

De nombreux systèmes de numération sont utilisés en électronique numérique. Les plus utilisés sont les systèmes :

- Binaire (base 2).
- Octal (base 8).
- Décimal (base 10).
- Hexadécimal (base 16)

• Représentation polynomiale

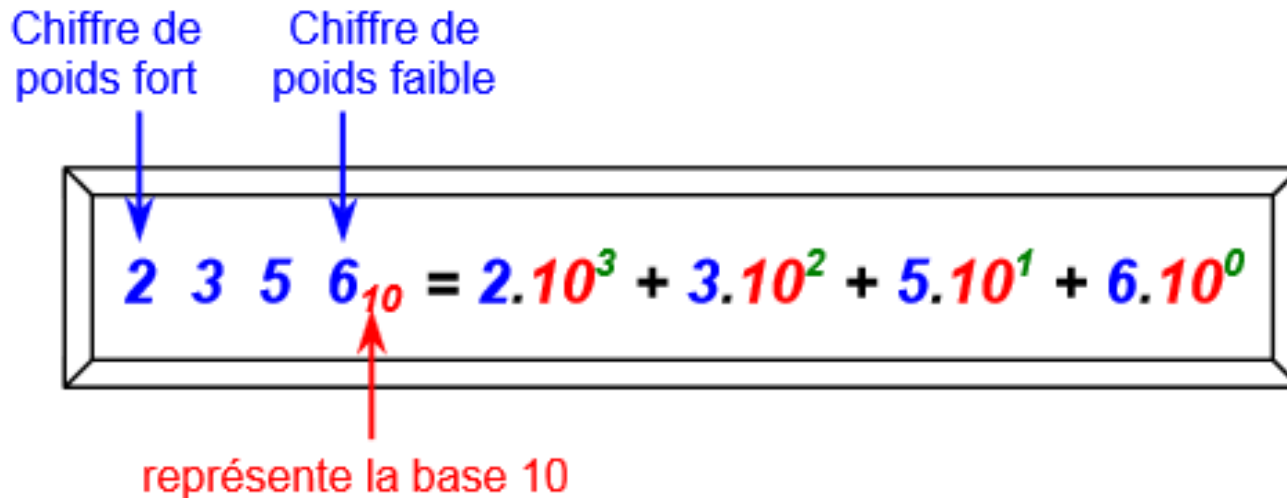
Tout nombre N peut se décomposer en fonction des puissances entières de la base de son système de numération. Cette décomposition s'appelle la forme polynomiale du nombre N et qui est donnée par :

$$N = a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \dots + a_1 \cdot b^1 + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + \dots + a_{-m} \cdot b^{-m}$$

- b : base du système de numération, il représente le nombre de chiffres différents qu'utilise ce système de numération.
- a_i : un chiffre (ou digit) parmi les chiffres de la base du système de numération.
- i : Rang du chiffre a_i .

- **Systeme decimal (base 10)**

Le système décimal comprend 10 chiffres qui sont $\{0,1,2,3,4,5,6,7,8,9\}$.



$4 \ 2 \ 9, \ 6 \ 5 \ 7_{10} = 4 \cdot 10^2 + 2 \cdot 10^1 + 9 \cdot 10^0 + 6 \cdot 10^{-1} + 5 \cdot 10^{-2} + 7 \cdot 10^{-3}$

- **Système binaire (base 2)**

Dans ce système de numération, il n'y a que deux chiffres possibles $\{0,1\}$ qui sont souvent appelés bits "binary digit".

Exemples :

Bit de poids fort
noté **M.S.B**

Bit de poids faible
noté **L.S.B**

$$10110_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 22_{10}$$

représente la base 2

$$10110,1011_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} = 22,6875_{10}$$

- **Système octal (base 8)**

Ce système octal ou à base 8, comprend 8 chiffres qui sont $\{0,1,2,3,4,5,6,7\}$. Les chiffres 8 et 9 n'existent pas dans cette base.

Exemples :

$$6057_8 = 6 \cdot 8^3 + 0 \cdot 8^2 + 5 \cdot 8^1 + 7 \cdot 8^0 = 3119_{10}$$

$$6057,14_8 = 6 \cdot 8^3 + 0 \cdot 8^2 + 5 \cdot 8^1 + 7 \cdot 8^0 + 1 \cdot 8^{-1} + 4 \cdot 8^{-2} = 3119_{10}$$

- **Systeme hexadecimal (base 16)**

Le système hexadécimal ou base 16 contient seize éléments qui sont $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$.

Exemples :

$$N_1 = 356_{16} = 3 \cdot 16^2 + 5 \cdot 16^1 + 6 \cdot 16^0 = 854_{10}$$

$$N_2 = 2AF_{16} = 2 \cdot 16^2 + 10 \cdot 16^1 + 15 \cdot 16^0 = 687_{10}$$

$$N_3 = 81,B_{16} = 8 \cdot 16^1 + 1 \cdot 16^0 + 11 \cdot 16^{-1} = 129,6875_{10}$$

- Résumé

CODES	BASE	SYMBOLES	UTILISATIONS
Décimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	Usage courant
Binaire	2	0, 1	C'est le code le plus utilisé en électronique numérique Ex : '0' = 0 volt '1' = 5 volts
Octal	8	0, 1, 2, 3, 4, 5, 6, 7	
Hexadécimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	Utilisé en informatique pour simplifier l'écriture des nombres binaires. Ex : $(110001110101)_2 = (C75)_{16}$

Changement de base

Il s'agit du processus de conversion d'un nombre écrit dans une base b_1 à une autre base b_2

- Conversion d'un nombre en base $b \rightarrow$ base 10

La valeur décimale d'un nombre N , écrit dans une base b , s'obtient par sa forme polynomiale

Exemples:

$$N_1 = 101101_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 45_{10}$$

$$N_2 = 6734_8 = 6 \cdot 8^3 + 7 \cdot 8^2 + 3 \cdot 8^1 + 4 \cdot 8^0 = 3548_{10}$$

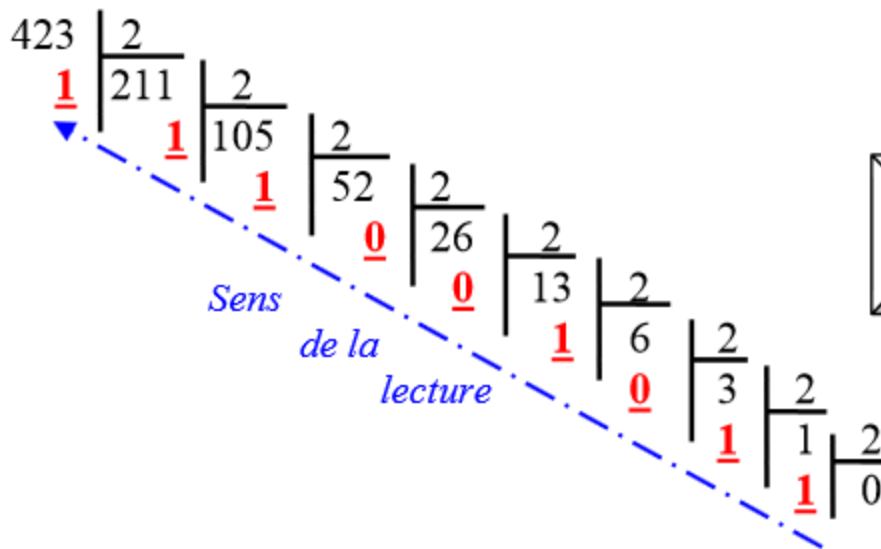
$$N_3 = A73_{16} = 10 \cdot 16^2 + 7 \cdot 16^1 + 3 \cdot 16^0 = 2675_{10}$$

- Conversion d'un nombre en base 10 $\rightarrow$ base b

Exemples

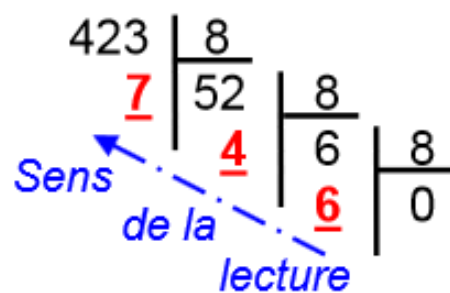
Convertir le nombre décimal 423_{10} en nombre binaire, octal et hexadécimal

a/ Conversion décimale binaire



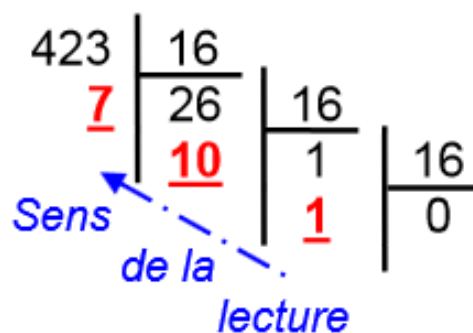
Résultat : $423_{10} = 110100111_2$

b/ Conversion décimale octale



Résultat : $423_{10} = 647_8$

c/ Conversion décimale hexadécimale



Résultat : $423_{10} = 1A7_{16}$

- Pour convertir un nombre décimal à virgule dans une base b quelconque, nous effectuons la division successive par b de la partie entière et on multiplie la partie fractionnaire du nombre à convertir par la base b et on note sa partie entière

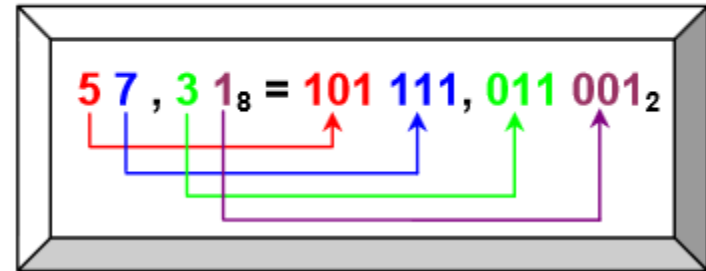
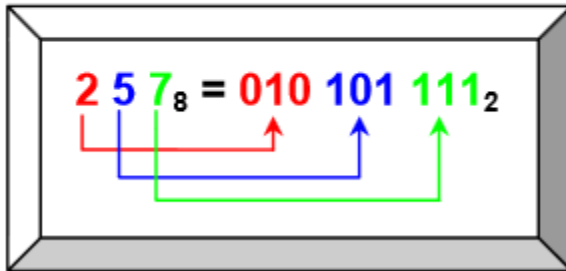
Exemple:

Conversion du nombre $58,32_{10}$ en base 2.	
Recherche de la partie entière par divisions successives par 2	Recherche de la partie fractionnaire par multiplications successives par 2
Sens de la lecture	Sens de la lecture Pour la partie décimale, on s'arrête quand le nombre de bits permet de respecter la précision souhaitée.
Résultat : $58,32_{10} = 111010,01010..._2$	

- Autres conversions : octal-binaire, hexadécimal-binaire, et vice versa

a- La conversion octal-binaire s'obtient en remplaçant chaque chiffre du nombre octal par son équivalent binaire écrit sur trois bits.

Exemples:

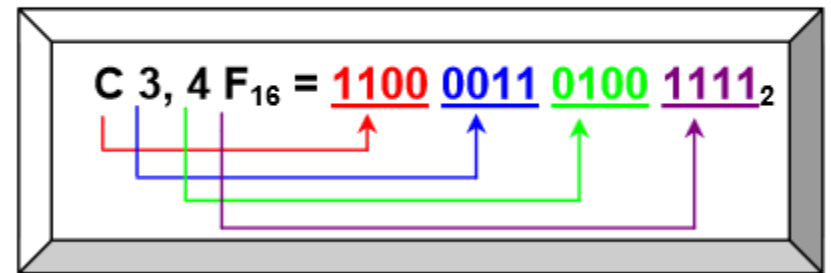
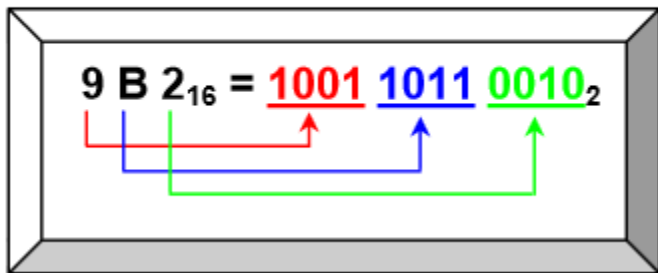


Inversement, pour convertir un nombre binaire en un nombre octal, il faut regrouper les bits du nombre binaire par trois en allant vers la gauche à partir de la virgule pour la partie entière, et vers la droite pour la partie fractionnaire, puis chaque groupe est remplacé par le chiffre octal correspondant.

- **b- La conversion hexadécimal-binaire**

s'obtient en remplaçant chaque chiffre du nombre hexadécimal par son équivalent binaire sur quatre bits.

Exemples:



Opérations arithmétiques binaires

Avant d'effectuer des opérations arithmétiques sur les nombres binaires, il faut définir au préalable, la représentation des nombres binaires signés. Le codage des nombres négatifs se fait le plus souvent en notation complément à 2

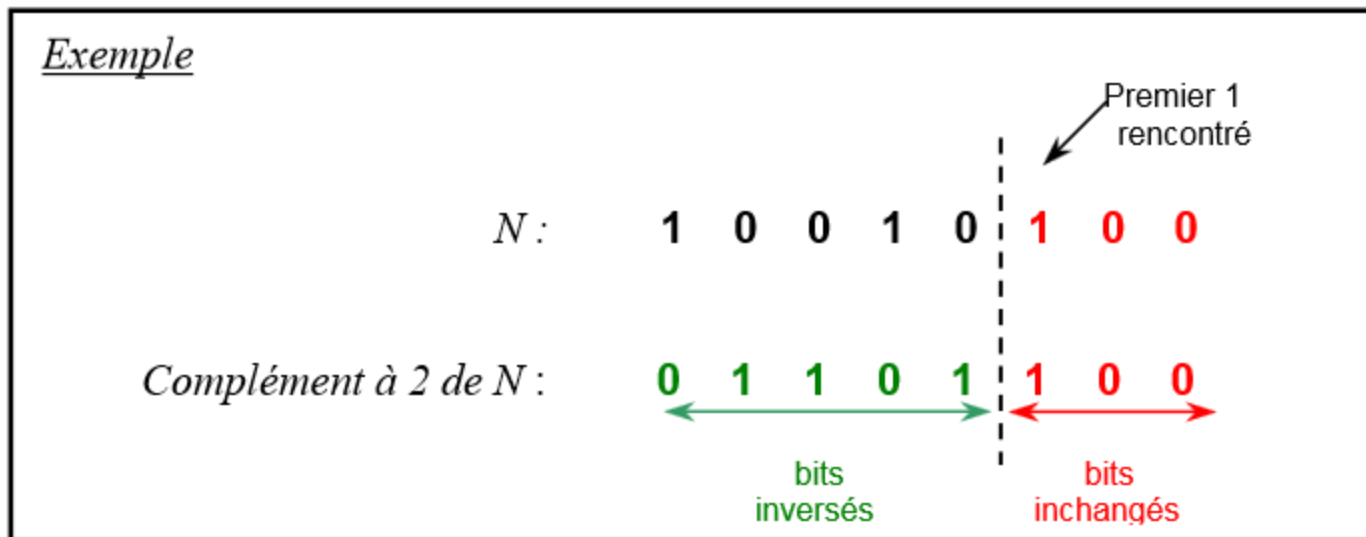
Représentation en complément à 2

Un nombre binaire signé est écrit en notation complément à 2 comme suit:

- Si le nombre est positif, il est écrit en nombre binaire pur avec un bit de signe 0, représenté par le bit de poids le plus fort.
- Si le nombre est négatif, il possède un bit de signe 1, mais il est écrit en notation en complément à 2. Le complément à 2 d'un nombre binaire est obtenu en changeant chaque 0 par 1 et chaque 1 par 0 (cette étape est appelée complément à 1) et en ajoutant 1 au bit de poids le plus faible.

Méthode pratique pour le calcul du complément à 2:

Cette méthode consiste à examiner chaque bit du nombre binaire, en commençant par celui de poids le plus faible (L.S.B). Tous les zéros à partir du L.S.B (s'il y en a) jusqu'au premier 1 rencontré seront conservés et tous les bits suivants seront inversés.



- Le tableau ci-dessous donne tous les nombres binaires écrits en notation en complément à 2 sur 4 bits :

Valeur décimale	Nombre binaire en complément à 2
$7=2^3 - 1$	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
$-8=-2^3$	1000

D'après le tableau ci-dessus, on remarque :

- En notation en complément à 2, les nombres positifs sont représentés avec un bit de signe 0 et les nombres négatifs par un bit de signe 1.
- En notation en complément à 2 et avec n bits, on représente les nombres signés compris dans l'intervalle $[-2^{n-1}, 2^{n-1}-1]$.

Addition binaire

La méthode d'addition pour des nombres binaires signés, consiste à écrire les nombres positifs en binaire avec un bit de signe 0, et à remplacer les nombres négatifs par leur complément à 2 avant l'addition. Si le résultat est positif, il est en notation binaire, s'il est négatif, il est en notation complément à 2.

En voici quelques exemples, où les nombres binaires sont écrits avec n=5 bits.

a/ $9_{10} + 4_{10} = ?$

	0	1	0	0	1	← 9_{10}
+	0	0	1	0	0	← 4_{10}
=	0	1	1	0	1	← 9_{10}

b/ $9_{10} - 4_{10} = ?$

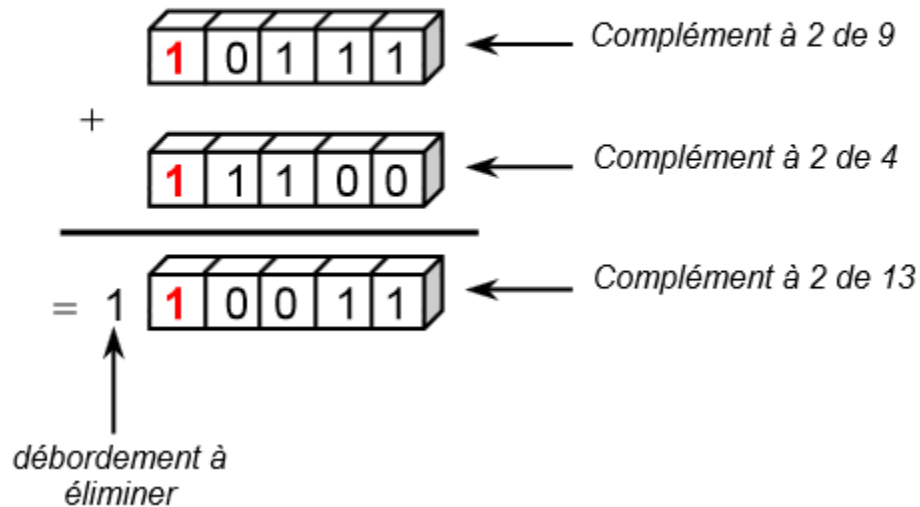
Comme -4_{10} est un nombre négatif, il faut le remplacer par son complément à 2.

	0	1	0	0	1	← 9_{10}
+	1	1	1	0	0	← Complément à 2 de 4
=	1	0	0	1	0	← 5_{10}

↑
débordement à éliminer

c/ $-9_{10} - 4_{10} = ?$

Comme -9_{10} et -4_{10} sont des nombres négatifs, il faut les remplacer par leurs compléments à 2 respectifs.



Le bit de signe est 1, donc le résultat de la somme est négatif, Il faut le complémenter à 2 pour trouver sa valeur absolue.

Remarque: Le débordement sur le $(n+1)$ ème bit du résultat de l'addition est toujours ignoré, car la taille des nombres binaires est limitée à n bits uniquement.

Opérations arithmétiques hexadécimales

Addition hexadécimale

La procédure proposée est la suivante :

- Additionner les deux chiffres hexadécimaux comme des chiffres décimaux, en remplaçant mentalement les lettres par leurs équivalents décimaux.
- Si la somme ≤ 15 , inscrire directement le chiffre hexadécimal.
- Si la somme est égale ou supérieur à 16, soustraire 16 et reporter 1 sur le rang à gauche.

Exemples:

$$\begin{array}{r} 5 \ 3 \\ + \ 8 \ 8 \\ \hline = \ D \ B \end{array}$$

$$\begin{array}{r} 5 \ 4 \\ + \ A \ 8 \\ \hline = \ F \ C \end{array}$$

$$\begin{array}{r} 3 \ A \ F \\ + \ 4 \ 3 \ C \\ \hline = \ 7 \ E \ B \end{array}$$

$$\begin{array}{r} C \ A \ F \ E \\ + \ C \ A \ F \ E \\ \hline = \ 1 \ 9 \ 5 \ F \ C \end{array}$$

Soustraction hexadécimale

On peut soustraire les nombres hexadécimaux en utilisant la même méthode que celle pour les nombres décimaux.

Exemples:

$$\begin{array}{r} C \ 5 \ B \\ - \ 5 \ 9 \ 2 \\ \hline = \ 6 \ C \ 9 \end{array}$$

$$\begin{array}{r} 1 \ 5 \ C \ E \\ - \ 7 \ D \ 5 \\ \hline = \ 0 \ D \ F \ 9 \end{array}$$

$$\begin{array}{r} B \ 4 \ C \ 9 \\ - \ 9 \ D \ 7 \ A \\ \hline = \ 1 \ 7 \ 4 \ F \end{array}$$

Les codes

L'action de faire correspondre à des nombres, des lettres ou des mots, un groupe spécial de symboles s'appelle codage. On distingue deux types de codes: les codes numériques (codes B.C.D, Gray...) et alphanumériques (code A.S.C.I.I, ...).

Code B.C.D (Binary Coded Decimal)

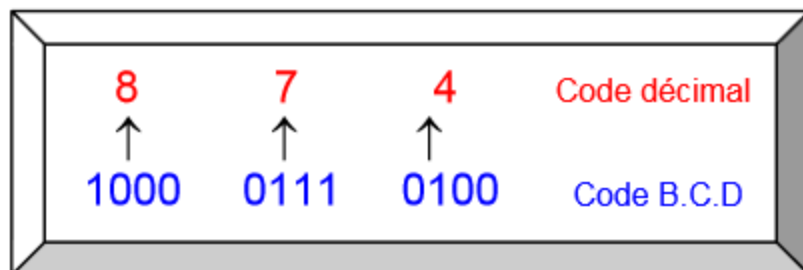
Le code B.C.D est un code pondéré qui représente chaque chiffre décimal par son équivalent binaire sur 4 bits, comme le montre le tableau suivant:

Code décimal	Code B.C.D
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

On note que les codes 1010, 1011, 1100, 1101, 1110, 1111 sont des combinaisons interdites, on les appelle des pseudo-tétrades.

Le codage en B.C.D d'un nombre décimal se fait en remplaçant chaque chiffre individuel de ce nombre par son équivalent binaire sur 4 bits.

Exemple:



On note bien la facilité relative avec laquelle on passe de ce code au nombre décimal correspondant et vice versa.

Il y a une méthode qui permet d'effectuer l'addition sur des nombres en B.C.D méthode.

Code Gray)

Le code Gray (ou code binaire réfléchi) est un code qui appartient à la catégorie des codes dit à distance minimale, du fait qu'une représentation codée en Gray ne diffère de celle qui la précède que d'un élément binaire et d'un seul, comme le montre le tableau 1.3.

Code décimal	Code binaire	Code Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Code décimal, binaire et Gray

Codage des nombres réels

- ✚ La représentation en virgule fixe : ces nombres sont représentés comme des entiers mais avec une virgule virtuelle gérée par le programmeur (exp : pour 8 bits on peut prendre 4 bits pour coder la partie entière et 4 bits pour la partie fractionnaire).

Exemple: $(3,25)_{10} = (11,01)_2$



Codage des nombres réels

- ✚ La représentation en virgule flottante : elle consiste à représenter les nombres sous la forme suivante :

$$N = M * B^E$$

Avec : **B** c'est la base (2, 8, 10, 16...)

M c'est la mantisse (un nombre purement fractionnaire n'ayant pas de chiffre à gauche de la virgule)

E c'est l'exposant (entier)

Exp : 12,45 $\rightarrow$ $0,1245 * 10^2$

0,000015 $\rightarrow$ $0,15 * 10^{-4}$

Codage des nombres réels

En notation scientifique standard, ces nombres sont représentés avec un seul chiffre avant la virgule.

Exemple: $12,45 \rightarrow 1,245 * 10^1$
 $0,000015 \rightarrow 1,5 * 10^{-5}$

De même en binaire la notation scientifique normalisé (en virgule flottante) sera :

$1, \text{xxxxxxxx} * 2^{\text{yyyy}}$

$1,101101 . 2^{0100}$

Codage des nombres réels

En 1985, le standard IEEE a défini trois formats de représentation des nombres flottants : la simple précision (sur 32 bits), la double précision (sur 64 bits) et la précision étendue (sur 80 bits) surtout utilisé dans les unités de calcul flottant.



Simple précision

Chaque format commence par un bit de signe (celui le plus à gauche), qui vaut 0 pour les nombres positifs et 1 pour les nombres négatifs. L'exposant est codé en excédent 127 pour la simple précision et en excédent 1023 pour la double précision). Enfin, la mantisse qui est codé sur 23 ou 52 bits.

simple précision sur 32 bits :

1 bit de signe de la mantisse

8 bits pour l'exposant

23 bits pour la mantisse

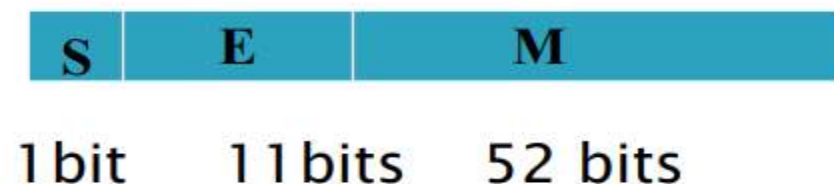


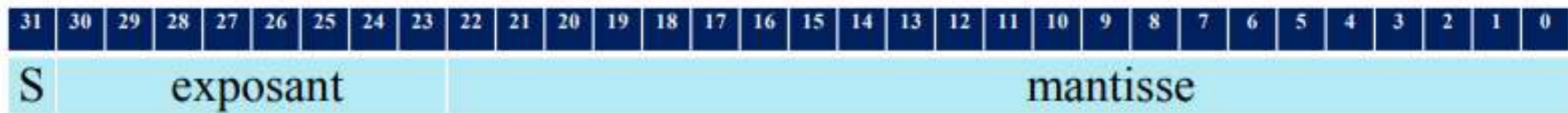
double précision sur 64 bits :

1 bit de signe de la mantisse

11 bits pour l'exposant

52 bits pour la mantisse





La valeur numérique d'un nombre N donné en simple précision:

$$N = (-1)^S \times (1 + \text{mantisse}) \times 2^{\text{exposant}}$$

$$= (-1)^S \times (1 + (m_1 \times 2^{-1}) + (m_2 \times 2^{-2}) + (m_3 \times 2^{-3}) + (m_4 \times 2^{-4}) + \dots) \times 2^{\text{exposant}}$$

Quel est le nombre décimal représenté par le mot suivant?

s	Exposant									Mantisse																					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...

$$N = (-1)^s \times (1 + \text{mantisse}) \times 2^{\text{exposant}}$$

$$(10000001)_2 = (129)_{10}$$

$$\rightarrow \text{exposant} = 129 - 127 = 2$$

$$\rightarrow N = (-1)^1 \times (1 + 0.25) \times 2^2$$

$$= -1.25 \times 4$$

$$= -5$$

Donner la représentation binaire IEEE 754 de
 -0.75 en simple précision.

$$(0.75)_{10} = (0.11)_2$$

$$= (1.1 \times 2^{-1})_2 \text{ En notation scientifique standard}$$

Exposant = -1

Et -1 en excédant 127 c'est **01111110**

Mantisse = $0.100000\dots$

$S = 1$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...

Caractéristiques des nombres flottants au standard IEEE

	Simple précision	Double précision
Bit de signe	1	1
Bit d'exposant	8	11
Bit de mantisse	23	52
Nombre total de bits	32	64
Codage de l'exposant	Excédant 127	Excédant 1023
Variation de l'exposant	-126 à +127	-1022 à +1023
Plus petit nombre normalisé	2^{-126}	2^{-1022}
Plus grand nombre normalisé	Environ 2^{+128}	Environ 2^{+1024}
Echelle des nombres décimaux	Environ 10^{-38} à 10^{+38}	Environ 10^{-308} à 10^{+308}
Plus petit nombre dénormalisé	Environ 10^{-45}	Environ 10^{-324}

Divers formats du standard IEEE

Normalisé	$\pm$	$0 < \text{Exp.} < \text{Max}$	Configuration quelconque de bits
Dénormalisé	$\pm$	0	Toute configuration sauf tous les bits à 0
Zéro	$\pm$	0	0
Infini	$\pm$	111...1	0
NaN (Not a Number)	$\pm$	111...1	Toute configuration sauf tous les bits à 0

Codage des caractères

De nombreux codes ont été utilisés pour représenter les caractères (codes BAUDOT sur 5 bits : insuffisants pour coder chiffres et lettres, EBCDIC sur 8 bits (IBM)). Un code normalisé, le code *ASCII* (*American Standard, Code for Information Interchange*) s'est peu à peu imposé en raison de son utilisation dans les micro-ordinateurs.

Le code ASCII représente les caractères sur 7 bits.

- ✗ Les 32 premiers (00–1F) représentent des caractères de contrôle utilisés soit dans des protocoles d'échange d'informations (ENQ, ACK, NAK, ...), soit pour le contrôle de certains terminaux (LF : Line Feed, CR : Carriage Return, BS : BackSpace,...).
- ✗ Les codes suivants (20–7F), sont utilisés pour coder les caractères alphabétiques (majuscules ou minuscules) et les caractères de ponctuation.
 $A = 41_{16} = 65_{10}$, $a = 61_{16} = 97_{10}$, or les caractères sont consécutifs dans le code •
- ✗ pour passer des majuscules aux minuscules on ajoute 32 (on force le 6^e bit à 1).

Table ASCII

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	\$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-`	63	3F	?	95	5F	_	127	7F	DEL