ENSAH

DMI

Département de Mathématiques et Informatique
Mathematics and Computer Science Department

Représentation binaire de l'information

Génie informatique
Semestre 1
Année universitaire 2019/2020

Généralités

- L'ordinateur d'aujourd'hui est un ensemble de circuits électroniques qui manipulent des données sous forme binaire.
- Toutes ces informations sont représentées par l'ordinateur sous forme binaire. Ces informations sont des suites de 0 et de 1.
- Le traitement de ces informations n'est autre que le traitement des suites binaires correspondantes.
- Dans ce cas, l'unité d'information est le chiffre binaire (0 ou 1) usuellement appelé bit (binary digit, chiffre binaire).
- Toute information utile à nos yeux que l'on désire traiter automatiquement, doit être codée.

Généralités

- Le codage consiste donc à faire une correspondance entre la représentation externe (usuelle) de l'information et la représentation à l'intérieur de la machine qui n'est autre qu'une suite binaire.

- À titre d'exemple, le nombre 13, le caractère T, un son, une Image, une séquence vidéo, représentent des informations pour l'être humain.

- Pour être traitées de façon automatique, ces informations sont transformées pour devenir des données (**int, char, float**, ….) et par la suite devenir des séquences binaires sans aucune signification.

- Dans les années 1930, Claude SHANON a démontré que toutes les opérations logiques (aussi les opérations arithmétiques) peuvent être réalisées en n'utilisant que des interrupteurs à deux états. Après, sont apparus les transistors qui sont à l'origine de la micro-informatique

Changement de base

- Nous avons pris l'habitude de manipuler des nombres décimaux composés de dix symboles distincts, qui sont les chiffres (0,1,2,3,4,5,6,7,8,9).

- Dans une autre base b, on utilise b symboles, (b chiffres).

- Soit le nombre *x* représenté par la suite des chiffres a_i , $x = a_n a_{n-1} \ldots . a_2 a_1 a_0$
- a_0 :représente le chiffre des unités.

  - ✓ En décimal, *b = 10*, a_i ∈ *{0; 1; 2; 3; 4; 5; 6; 7; 8; 9};*
  - ✓ En binaire, *b = 2*, a_i ∈ *{0; 1}* : *2* chiffres binaires, ou bits ;
  - ✓ En hexadécimal, *b = 16*, a_i ∈ *{0; 1; 2; 3; 4; 5; 6; 7; 8; 9; A;B;C;D;E; F}*

Changement de base

-Représentation des nombres entiers-

- *En base 10, la représentation du nombre 45678 est :*

$$45678 = 4 . 10^4 + 5 . 10^3 + 6 . 10^2 + 7 . 10^1 + 8 . 10^0$$

- *Dans le cas général, le nombre représenté par la suite* $(a_n a_{n-1} \dots a_2 a_1 a_0)_b$ *dans une base b est donné par :*

$$(a_n a_{n-1} \dots a_2 a_1 a_0) = \sum_{i=0}^{n} a_i b^i$$

- *Le résultat est donné en décimal.*
- a_0 *est le chiffre de poids faible, et* a_n *le chiffre de poids fort.*
- *Exemple: Soit le nombre binaire* $(1001011)_2$
- $= 1 . 2^6 + 0 . 2^5 + 0 . 2^4 + 1 . 2^3 + 0 . 2^2 + 1 . 2^1 + 1 . 2^0 = 75.$
- *L'écriture* $( )_b$ *signifie que le nombre est écrit dans la base b.*

Changement de base

-Représentation des nombres fractionnaires

- *Les nombres fractionnaires sont les nombres qui comportent des chiffres après la virgule. Dans le système décimal, le nombre 436,765 est écrit comme suit :*

$$436{,}765 = 4.\mathbf{10^{2}} + 3.\mathbf{10^{1}} + 6.\mathbf{10^{0}} + 7.\mathbf{10^{-1}} + 6.\ \mathbf{10^{-2}} + 5.\mathbf{10^{-3}}$$

- *Remarque:*

En décimale, on ne peut représenter avec exactitude que les nombres fractionnaires qui s'écrivent sous la forme $\frac{x}{10^{m}}$

- *En général, dans une base b, l'écriture est comme suit :*
- $\boldsymbol{a_n a_{n-1} \dots a_2 a_1 a_0, a_{-1} a_{-2} \dots \dots a_{-p} = a_n\, b^{n} + a_{n-1} b^{n-1} + \dots + a_0 b^{0} + a_{-1} b^{-1} + \dots + a_{-p}\, b^{-p}}$.
- *Là aussi, pour une base b, les seuls nombres qui peuvent être représenté avec exactitude sont ceux qui s'écrivent sous la forme* $\frac{x}{10^{n}}$

Changement de base

- Passage d'une base b vers la base 10

- *Il suffit d'écrire le nombre comme somme de produits vue précédemment.*
- $(11011)_2 = 1.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 = 27$
- $(1367)_8 = 1.8^3 + 3.8^2 + 6.8^1 + 7.8^0 = 759.$
- $(12A5D)_{16} = 1.16^4 + 2.16^3 + 10.16^2 + 5.16^1 + 13.16^0 = 76381.$

Changement de base

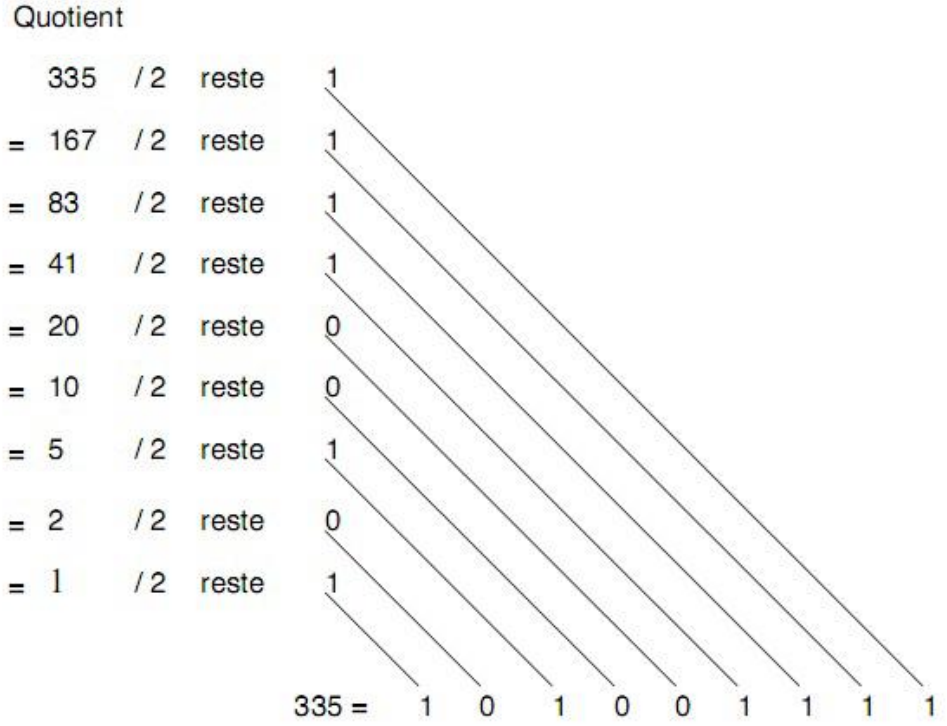
- Passage d'une base10 vers la base b

❖ **Nombres entiers**

❑ On procède par divisions successives. On divise le nombre par la base, puis le quotient obtenu par la base, et ainsi de suite jusqu'a obtention d'un quotient nul.

❑ La suite des restes obtenus correspond aux chiffres dans la base visée, $a_n \ldots a_1 \, a_0$

❑ Soit à convertir le nombre 335 en base 2,



Changement de base

- Passage d'une base10 vers la base b

❖ ***Code BCD des entiers***

❑ Une représentation binaire des entiers particulière est très utilisée. Cette représentation consiste à représenter en binaire chacun des chiffre indépendamment des autres.

❑ Comme exemple, prenons la valeur décimale suivante 7859

❑ On code en binaire chaque chiffre appart :

7---------- > 0111
8---------- > 1000
5---------- > 0101
9---------- > 1001

Alors la valeur codée en BCD est : $(\mathbf{01111000\ 0101\ 1001})_{BCD}$

Changement de base

- Passage d'une base10 vers la base b

❖ ***Nombres fractionnaires***

❑ Soit à convertir en binaire le nombre fractionnaire suivant $(99{,}125)_{10}$

❑ Pour la partie entière $(99)_{10} = (1100110)_2$

❑ Pour la partie fractionnaire:

0,125	X	2	=	0,25	a_{-1}= 0
0,25	X	2	=	0,5	a_{-2}= 0
0,5	X	2	=	1	a_{-3}= 1

❑ Nous avons réalisé 3 opérations de multiplication.

❑ $(\mathbf{1100110,001})_2 = 1.2^6 + 1.2^5 + 0.2^4 + 0.2^3 + 1.2^2 + 1.2^1 + 0.2^0 + 0.2^{-1} + 0.2^{-2} + 1.2^{-3}$

❑ ===== > $(99{,}125)_{10} = (\mathbf{1100110,001})_2$

Changement de base

- Passage d'une base10 vers la base b

❖ ***Nombres fractionnaires***

❑ Soit à convertir en binaire le nombre fractionnaire suivant $(14,3)_{10}$

❑ Pour la partie entière $(14)_{10} = (1110)_2$

❑ Pour la partie fractionnaire:

0,3	X	2	=	0,6	a_{-1}= 0
0,9	X	2	=	1,2	a_{-2}= 1
0,2	X	2	=	0,4	a_{-3}= 0
0,4	X	2	=	0,8	a_{-4}= 0
0,8	X	2	=	1,6	a_{-5}= 1

❑ Nous avons réalisé 5 opérations de multiplication.

❑ ===== > $(14,3)_{10}$=$(\mathbf{1110,01001})_2$ avec une précision de 5 bits après la virgule.

Changement de base

- Passage d'une base10 vers la base b

❖ ***Cas des bases 2, 8, 16***

- Ces bases correspondent à des puissances de 2 (2^1; 2^3 et 2^4), d'où des passages de l'une à l'autre très simples. Les bases 8 et 16 sont pour cela très utilisées en informatique, elles permettent de représenter rapidement et de manière compacte des configurations binaires.

- La base 8 est appelée notation octale, et la base 16 notation hexadécimale. Chaque chiffre en base 16 (2^4) représente un paquet de 4 bits consécutifs.

- Pour convertir un nombre de la base décimale vers la base octale ou hexadécimale, il suffit de le convertir en base binaire puis faire effectuer un regroupement par 3 ou 4 bits,

Changement de base

- Passage d'une base10 vers la base b

❖ ***Cas des bases 2, 8, 16***

- Par exemple :
- $(10011010)_2 = (1001 \quad 1010)_2 = (9A)_{16}$
- De même, chaque chiffre octal représente 3 bits.
- $(10011010)_2 = (010 \quad 011 \quad 010)_2 = (232)_8$
- On manipule souvent des nombres formés de 8 bits, nommés octets, qui sont donc notés sur 2 chiffres hexadécimaux.
- Les opérations arithmétiques s'effectuent en base quelconque b avec les mêmes méthodes qu'en base 10. Une retenue ou un report apparait lorsque l'on atteint ou dépasse la valeur b de la base.

Changement de base
-Opérations arithmétiques

- Les opérations arithmétiques s'effectuent en base quelconque b avec les mêmes méthodes qu'en base 10. Une retenue ou un report apparait lorsque l'on atteint ou dépasse la valeur b de la base.
- **Exemples:**
- **Addition octale:**

$(23)_8+(32)_8=(55)_8$	$(243)_8+(742)_8=(1185)_8$
$(453)_8+(567)_8=(1242)_8$	$(7457)_8+(7647)_8=(??)_8$
$(8573)_8+(1124)_8=(??)_8$	$(472)_8+(32)_8=(??)_8$

Changement de base
-Opérations arithmétiques

- **Exemples:**
- **Multiplication binaire:**

```
  11
x 10
----
 00
+11
----
 110
```

```
  1010
x   11
------
  1010
+1010
------
 11110
```

```
  1010
x 11,01
-------
  10 10
+ 000 0
+ 1010
+ 1010
---------
100000 10
```

- **A faire:**

$(110011)_2 * (101)_2 = (??)_2$	$(110)_2 * (110)_2 = (??)_{16}$
$(11)_2 * (11,011)_2 = (??)_2$	$(1101)_2 * (1111)_2 = (??)_8$
$(E,A)_{16} * (11,1)_2 = (??)_8$	$(15)_8 * (1011,01)_2 = (??)_{16}$

Changement de base
-Codification des nombres entiers

- Il est nécessaire de coder des nombres dans le but de stocker et manipuler ces derniers par un ordinateur. Le problème majeur est la limitation de la taille du codage.

- En effet, un nombre peut prendre des valeurs arbitraires qui soient très grandes, et l'ordinateur doit effectuer le codage de ces nombres sur un nombre de bits fixé à l'avance.

- En général, les entiers naturels(positifs ou nuls) sont codés sur un nombre d'octets fixé. Les codages les plus utilisés sont sur 1, 2, 4 octets . Le codage sur 8 octet est rarement utilisé

Changement de base
-Codification des nombres entiers

❖ **Entiers naturels:**

❑ De façon générale, un codage sur n bits permet de coder les nombres naturels compris entre 0 est 2^n- 1 . Pour le cas d'un octet (8 bits), il permet de coder les nombre compris entre 0 est 255=2^8– 1 (codage de 256 valeurs, 2^n). La valeur maximale vaut 2^n-1

Changement de base
-Codification des nombres entiers

❖ **Entiers relatifs:**

❑ Pour le cas des entiers relatifs, il faut penser à coder le signe. Pour réaliser ce codage, on utilise le codage en complément à deux. Le complément à deux est le complément à 1 + 1.

❑ En adoptant une représentation sur n bits fixés, un nombre entier positif ou nul est représenté en binaire en prenant soin de garder toujours le $n^{ème}$ bits (bit du poids fort) à zéro; seulement n-1 bits sont utilisés.

❑ Dans ce cas, la valeur maximale à coder est $2^{(n-1)} - 1$.

❑ **Exemple**: sur 8 bits, la valeur positive maximale est 127

Changement de base
-Codification des nombres entiers

❖ **Entiers relatifs:**

❑ Si x est une valeur négative, le code décimal correspondant est $2^n - |x|$.

❑ La représentation binaire de $-x$ (x >0) est obtenue par complément à 2 de la représentation binaire de x. Le complément à 2 d'une représentation binaire n'est autre que le complément à 1 de cette représentation plus 1.

❑ **Exemple**: soit à coder la valeur (– 21) représenté sur 8 bits.

- On commence par trouver la représentation de 21 sur 8 bits:
- $(21)_{10}$=$(00010101)_2$.
- On détermine le complément à 1 de (00010101) ===== > (11101010)
- puis on ajoute 1 ==== > (11101011)
- Donc le code binaire de (– 21) est $(11101011)_2 = (233)_{10} = (EB)_{16} = (353)_8$

Changement de base
-Codification des nombres entiers

❖ **Entiers relatifs:**

❑ Pour des nombre relatif représentés sur n bits :

- Pour un nombre négatif, le bit du poids fort de la représentation est toujours égal à 1.
- La plus grande valeur positive à coder est $2^{n-1} - 1$.
- La plus petite valeur négative à coder est -2^{n-1}.

Changement de base
-Codification des caractères

- Les caractères sont des données non numériques qui ne peuvent pas être additionnés ou multipliés. Cependant, il est parfois utile de les comparer pour pouvoir les trier dans l'ordre alphabétique**.**

- Les caractères, aussi appelés symboles alphanumériques, incluent les lettres minuscules et majuscules, les chiffres et les symboles de ponctuation (‘ ; , : & ~ # etc).

- Dans ce cas, un texte sera représenté comme une suite de caractères.

Changement de base
-Codification des caractères

- Le codage des caractères est fait par une table de correspondance indiquant la configuration binaire représentant chaque caractère.

- Les deux codes les plus connus sont:

- Le code **EBCDIC** (***Extended Binary Coded Decimal Interchage Code***) en voie de disparition. C'est un mode de codage de caractères sur 8 bits créé par IBM lorsque les carte perforées étaient à la mode. Il existe au moins six versions différentes.

- Le code **ASCII** (***American Standard Code for Information interchange***) qui représente chaque caractère sur 7 bits (de nos jours sur 8 bits: code ASCII étendu pour le codage des caractères supplémentaires) .

Changement de base
-Codification des caractères

- Il faut noté que, à l'origine, le code ASCII développé pour l'informatique en anglais, ne prenait pas en considération les caractères accentués (é, è , ê , ï …). Les caractères arabe, chinois et bien d'autres ne sont pas pris en considération par ce code.

- Pour ces langes, d'autres types de codage existent utilisant 16 bits.

Changement de base
-Codification des caractères

- Les code compris entre 0 et 31 ne représentent pas de caractères, il ne sont pas affichables. Ces codes portent le nom de : caractères de contrôles.

- Les lettres se suivent dans l'ordre alphabétique (codes 65 à 90 pour les majuscules, 97 à 122 pour les minuscules), ce qui simplifie les comparaisons.

- On passe des majuscules au minuscules en modifiant le $6^{ième}$ bit, ce qui revient à ajouter 32 (ou retrancher 32) au code ASCII décimal.

- Les chiffres sont rangés dans l'ordre croissant (codes 48 à 57), et les 4 bits de poids faibles définissent la valeur en binaire du chiffre.

Changement de base

-Codification des caractères

Caractères de contrôle du code ASCII

ASCII	Caract.	Signification	ASCII	Caract.	Signification
00	NUL	*null*, nul	16	DLE	*data link escape*, échap liaisondonnées
01	SOH	*start of heading*, début d'en-tête	17	DC1	*device control 1*, commande unité 1
02	STX	*start of text*, début de texte	18	DC2	*device control 2*, commande unité 2
03	ETX	*end of text*, fin de texte	19	DC3	*device control 3*, commande unité 3
04	EOT	*end of transmission*, fin de transmission	20	DC4	*device control 4*, commande unité 4
05	ENQ	*enquiry*, interrogation	21	NAK	*negative acknowledge*, acc. récep. nég.
06	ACK	*acknowledge*, accusé de réception	22	SYN	*synchronous idle*, inactif synchronisé
07	BEL	*bell*, sonnerie	23	ETB	*end of transmission block*, fin tran. bloc
08	BS	*backspace*, espacement arrière	24	CAN	*cancel*, annuler
09	HT	*horizontal tabulation*, tabulation horiz.	25	EM	*end of medium*, fin du support
10	LF	*line feed*, saut de ligne	26	SUB	*substitute*, substitut
11	VT	*vertical tabulation*, tabulation verticale	27	ESC	*escape*, échappement
12	FF	*form feed*, saut de page	28	FS	*file separator*, séparateur de fichiers
13	CR	*carriage return*, retour chariot	29	GS	*group separator*, séparateur de groupes
14	SO	*shift out*, hors code	30	RS	*record separator*, sép. d'enregistr.
15	SI	*shift in*, en code	31	US	*unit separator*, séparateur d'unités

Changement de base

-Codification des caractères

ASCII	Caractère.
32	SP (*space*, espace)
33	!
34	"
35	#
36	$
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?

ASCII	Caractère
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[
92	\
93	]
94	^
95	_

ASCII	Caractère
96	`
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~
127	DEL (*delete*, sup.)

Changement de base

- Représentation des nombres réels norme (IEEE)

- Soit à codifier le nombre 5,25 qui s'écrit en base 2 $(101{,}01)_2$.

- On va **normaliser la représentation** en base 2 de telle sorte qu'elle s'écrive sous la forme : $\mathbf{1}, \ldots * \mathbf{2}^{\boldsymbol{n}} = \boldsymbol{s}\mathbf{1}, \boldsymbol{M} * \mathbf{2}^{\boldsymbol{e}}$
- Dans l'exemple 101,01 = **1, 0101 * $\mathbf{2}^{\mathbf{2}}$**

- La représentation **IEEE 754** code séparément le **signe** (s) du nombre (dans notre cas +), **l'exposant** (e) n (dans notre cas 2), et la **mantisse** (M) (la suite de bits après la virgule), le tout sur 32 bits (simple précision).

Changement de base
- Représentation des nombres réels norme (IEEE)

- Soit à codifier le nombre 128 qui s'écrit en base 2 $(1000\ 0000)_2$.

- On va **normaliser la représentation** en base 2 de telle sorte qu'elle s'écrive sous la forme : $\mathbf{1,\ldots * 2^n = s1,M * 2^e}$

- **Le signe S=0**
- $\mathbf{|128| = 128} = 1000\ 0000_2 = (1{,}0)_2 * 2^7$
- $\boldsymbol{e = 7\ et\ M} = 00000\ldots0_2$
- $\boldsymbol{E = e + 127 = 7 + 127 = 128 + 4 + 2} = 1000\ 0110_2$
- **Alors 128** $\rightarrow$ 0 10000110 00000000000000000000000

0 10000110 00000000000000000000000

Signe: **S** **(1 bit)**

Exposant : **E** **(8bits)**

Mantisse **M (23 bits)**

La représentation hexadécimale de ce code sur 32 bits est : 43000000_{16}

Changement de base

- Représentation des nombres réels norme (IEEE)

❑ Soit à codifier le nombre -32,75 qui s'écrit en base 2 $(10\ 0000{,}11)_2$.

- **Le signe S=1**
- $|\mathbf{-32,75}| = 10\ 0000{,}11_2 = (1{,}0000011)_2 * \ 2^5$
- $\boldsymbol{e = 5\ et\ M} = 0000011 \ldots 0_2$
- $\boldsymbol{E = e + 127 = 5 + 127 = 128 + 4} = 1000\ 0100_2$
- **Alors** $\mathbf{-23,75}$ $\rightarrow 1\ 10000100\ 00000110000000000000000$

1 10000100 00000110000000000000000

Signe: **S** **(1 bit)**

Exposant : **E** **(8bits)**

Mantisse **M (23 bits)**

La représentation hexadécimale de ce code sur 32 bits est : $C2030000_{16}$

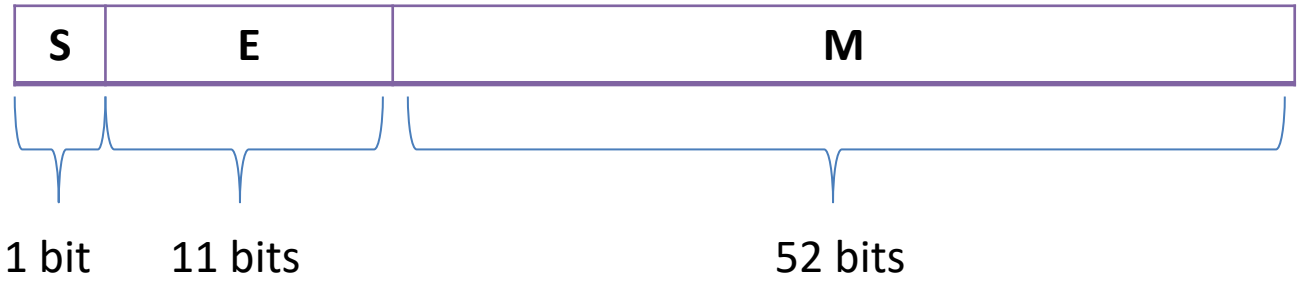
Changement de base
- Représentation des nombres réels norme (IEEE)

- L'exposant biaisé (00000000) indique que le nombre est dénormalisé lorsque la mantisse est différente de zéro.
- Lorsque l'exposant biaisé vaut (00000000) et les bit de la mantisse sont tous à 0, cela signifie que la valeur codée est zéro
- Lorsque tous les bits de la mantisse sont à 0 est l'exposant biaisé vaut (11111111), cela signifie que le nombre représenté est infini.
- Lorsque au moins un bits de la mantisse est différent de zéro et que l'exposant biaisé vaut(11111111), c'est la configuration NaN (Not a Number) qui est utilisée pour signaler des erreurs de calculs comme 0/0 ou √-1

Changement de base
- Représentation des nombres réels norme (IEEE)

- Il existe aussi un deuxième codage pour les nombres réels sur 64 bits c'est le standard IEEE 754 double précision :

- Pour ce type de codage sur 64 bits, la répartition est comme suit : Le signe est codé sur un bits celui du poids fort. L'exposant est codé sur 11 bits. La mantisse est codée sur les 52 bits de faible poids.



Changement de base

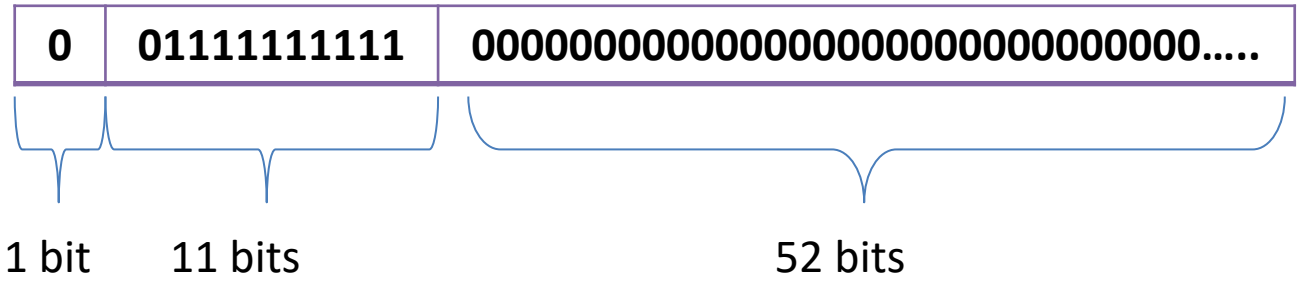
- Représentation des nombres réels norme (IEEE)

- Le signe est représenté sur le bit de poids fort **S**, + est représenté par 0 et - par 1.
- L'exposant n est codé sur 11 bits comme suit :on code en binaire la valeur **E**=n+1023.
- La mantisse **M** est codée sur les 52bits de poids faibles.

Changement de base
- Représentation des nombres réels norme (IEEE)

- Soit à codifier le nombre 1 en représentation flottante, en double précision

- **Le signe S=0**
- $|\mathbf{1}| = 1_2 = (1)_2 * \ 2^0$
- $\boldsymbol{e = 0 \ et \ M} = 00000 \ldots 0_2$
- $\boldsymbol{E = 0 + 1023 = 011\ 1111\ 1111_2}$
- **Alors** $\mathbf{1} \ \rightarrow 0\ 011\ 1111\ 1111\ 0 \ldots .0$



La représentation hexadécimale de ce code sur 32 bits est : $3FF0000000000000_{16}$