A. Codage des nombres réels :

Ce qu'il faut savoir : Le codage ***sur 64 bits*** (8 octets) en binaire des nombres réels pour les représenter en machine est défini par ***la norme IEEE 754***.

Pour représenter un nombre à virgule en binaire, on va utiliser une représentation similaire à la notation scientifique en base 10, mais ici en base 2 : il s'agit de ***la représentation en virgule flottante***.

Cette représentation est $Sm\times 2^n$ où ***S*** est ***le signe*** (+ ou -), ***m*** est ***la mantisse*** ($1\leqslant m<2$) et ***n*** ***l'exposant*** qui est un nombre entier relatif.

On codera en binaire ce nombre ainsi représenté sur 64 bits ainsi :

64è bit	63è bit	61è bit	…......	53è bit	52è bit	51è bit	50è bit	49è bit	…......	3è bit	2è bit	1er bit
signe	**exposant décalé**				**m – 1 = partie décimale de la matisse**							

1 bit
(0 code + et 1 code -)

11 bits *codent un nombre entier naturel*
d *=* ***exposant décalé*** *= n+1 023*
($0\leq d\leq 2\,047$).

Si d=0 ou 2 047, cela est réservé à des situations exceptionnelles telles que ***l'infini*** *ou* ***« not a number »****.*

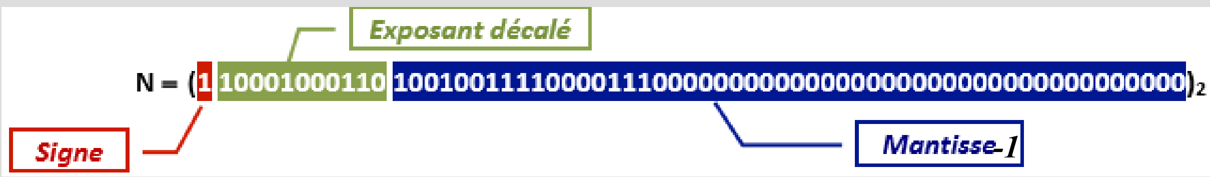
Si $1\leq d\leq 2\,046$, ***on soustrait 1 023*** *pour obtenir l'exposant n qui est un entier relatif.*

52 bits *codent m-1 avec $1\leq m<2$.*

La partie entière de m est toujours égale à 1, *il est inutile de coder ce chiffre et de perdre 1 bit.*

m-1, la partie décimale de m est une somme de puissances négatives de 2.

Exemple :



Signe = 1 : le nombre ***est négatif***.

Exposant décalé d = $(10001000110)_2=2^{10}+2^6+2^2+2^1=1024+64+4+2=$ 1 094.
Donc ***l'exposant n*** = 1 094-1 023=***71***.

Mantisse = $(1{,}10010011110000111000\ldots)_2=1+2^{-1}+2^{-4}+2^{-7}+2^{-8}+2^{-9}+2^{-10}+2^{-15}+2^{-16}+2^{-17}\approx 1{,}5772$.

La partie entière de la mantisse est toujours 1.

DONC ce code représente le réel : $\mathbf{-1{,}5772\times 2^{71}\approx -3{,}724\times 10^{21}}$ **.**

Cas particulier : On convient que le réel ***0 est codé par (0 00000000000 0000...00)$_2$***.

__*Activité 1 :*__

1. Trouve le nombre réel codé au format IEEE 754 par :

0 00100000011 1101001110010101100000000000000000000000000000000000.

Signe = 0 : le nombre est positif.

Exposant décalé d = $(00100000011)_2 = 2^8 + 2^1 + 2^0 = 259$ **.**

Donc l'exposant n = 259-1 023=-764.

Mantisse $= (1,110100111001010110000\ldots)_2$ **=** $1 + 2^{-1} + 2^{-2} + 2^{-4} + 2^{-7} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-14} + 2^{-16} + 2^{-17}$

$\approx$ ***1,826 499 939***

Donc ce code représente le réel 1,826 499 939×2^{-764} $\approx$ ***1,882 361*** 254×10^{-230}

2. Inversement, on veut maintenant coder le nombre réel -118,375 au format IEEE 754.

a. Divise 118,375 par 2 autant de fois qu'il le faut afin d'obtenir un résultat supérieur ou égal à 1 et strictement inférieur à 2. Déduis-en l'écriture de -118,375 sous la forme $Sm \times 2^n$ où S est le signe, m la matisse et n l'exposant.

$118,375 \div 2 = 59,1875$

$59,1875 \div 2 = 29,59375$

$29,59375 \div 2 = 14,796875$

$14,796875 \div 2 = 7,3984375$ ***donc*** $-118,375 = -1,849609375 \times 2^6$

$7,3984375 \div 2 = 3,69921875$

$3,69921875 \div 2 = 1,849609375$

b. A quel bit correspond le signe S ? ***- correspond au bit 1.***

c. Calcule l'exposant décalé. ***L'exposant décalé est*** $n + 1023 = 6 + 1023 = 1029$ **.**

d. A quel code binaire correspond cet exposant décalé ?

1 029 se code sur 11 bits par : (10000000101)$_2$***.***

e. Calcule la partie décimale de la mantisse. $1,849609375 - 1 = 0,849609375$ **.**

f. Multiplie la partie décimale de la mantisse par 2. Si le résultat est strictement inférieur à 1, le premier bit du code de la partie décimale de la mantisse est 0. En revanche, si le résultat est supérieur à 1, le premier bit est 1 et tu dois soustraire 1 au résultat.

Puis recommence ceci : multiplie le dernier résultat par 2. Si tu obtiens un résultat strictement inférieur à 1, ajoute le bit 0 à droite du code, sinon ajoute 1 à droite du code et soustrais 1 au résultat. Etc...

Tu devras t'arrêter lorsque tu obtiendras exactement 1. A ce moment, tu ajouteras le bit 1 à droite du code, puis tu compléteras par des 0 à droite du code pour qu'il y ait bien 52 bits.

Donne ainsi le code de la partie décimale de la mantisse sur 52 bits.

$0,849609375 \times 2 = 1,69921875$ ***donc le premier bit est 1.***

$0,69921875 \times 2 = 1,3984375$ ***donc le second bit est 1.***

$0,3984375 \times 2 = 0,796875$ ***donc le 3è bit est 0.***

$0,796875 \times 2 = 1,59375$ ***donc le 4è bit est 1.***

$0,59375 \times 2 = 1,1875$ ***donc le 5è bit est 1.***

$0,1875 \times 2 = 0,375$ ***donc le 6è bit est 0.***

$0,375 \times 2 = 0,75$ ***donc le 7è bit est 0.***

$0,75 \times 2 = 1,5$ ***donc le 8è bit est 1.***

$0,5 \times 1 = 1$ ***donc le 9è bit est 1, et c'est terminé, tous les bits suivants sont 0.***

Ainsi la partie décimale de la mantisse se code (110110011000...00)$_2$***.***

g. Déduis-en le code au format IEEE 754 de -118,375.

Donc le code de -118,375 au format IEEE 754 est (1 10000000101 110110011000...)₂.

3. En reproduisant la méthode de la question 2., détermine le code au format IEEE 754 de ces nombres réels.

a. 75,281 25. ***b.*** -0,687 5.

75,281 25= $\mathbf{1,1762695313\times 2^{6}}$ $\mathbf{-06875=-1,375\times 2^{-1}}$

(0 10000000101 00101101001000...)₂. ***(1 01111111110 011000...)₂***

<u>Activité 2 :</u>

1. a. Comment code-t-on en binaire le nombre entier -7 (sur 8 bits) ? Et le nombre réel -7,0 (sur 64 bits) ?

Le nombre entier relatif -7 se code sur 8 bits (avec la méthode du complément à 2) par l'entier naturel $\mathbf{-7+2^{8}=249}$ ***. On obtient donc*** $\mathbf{(11111001)_2}$ ***.***

$\mathbf{-7=-1,75\times 2^{2}}$ ***, ainsi le code au format IEEE 754 est : (1 10000000001 1100000000000000000000000000000...)₂.***

b. Pourquoi le même nombre n'est-il pas codé de la même façon ? ***Parce que les types sont différents.***

Quelle précaution faudra-t-il prendre lorsqu'on écrit un programme (notamment avec Python) ?

Lorsqu'on écrit un programme il est donc essentiel, lorsqu'on définit une variable, de savoir de quel type elle est (ici l'entier -7 est de type « integer » et le réel -7,0 est de type « float »).

2. a. Applique la méthode de l'activité 1 pour tenter de convertir le nombre réel 0,2 au format IEEE 754.

Que se passe-t-il ?

$\mathbf{0,2\times 2=0,4}$ ***;*** $\mathbf{0,4\times 2=0,8}$ ***;*** $\mathbf{0,8\times 2=1,6}$ ***. Donc*** $\mathbf{0,2=1,6\times 2^{-3}}$ ***.***

Le signe est + donc le premier bit est 0.

L'exposant décalé est -3+1 023=1 020 qui se code (01111111100)₂.

La partie décimale de la mantisse est 0,6.

$\mathbf{0,6\times 2=1,2}$ ***donc le premier bit est 1.***

$\mathbf{0,2\times 2=0,4}$ ***donc le 2è bit est 0.***

$\mathbf{0,4\times 2=0,8}$ ***donc le 3è bit est 0.***

$\mathbf{0,8\times 2=1,6}$ ***donc le 4è bit est 1.***

Comme 1,6-1=0,6 c'est cyclique, et les bits« 1001 »vont se répéter indéfiniment= (10011001...1001...)₂.

b. Quand les 52 bits de la mantisse ne suffisent pas, si le 53è bit est 1, on arrondit le bit de la mantisse qui est le plus à droite par excès. Mais si le 53è bit est 0, on arrondit le bit de la mantisse qui est le plus à droite par défaut. Quel sera finalement le code de 0,2 au format IEEE 754 ?

La mantisse se code $\mathbf{(1001100110011\ldots)_2}$ ***On a donc (1001 1001 1001...1001 1...)₂ , en arrondissant :***

(10011001100...1010)₂ .

53è bit

Ainsi le code au format IEEE 754 de 0,2 est (0 01111111100 10011001100...1010)₂.

c. Déduis-en le code au format IEEE 754 du réel 0,1. Puis applique la méthode de l'activité 1 pour trouver celui du réel 0,3.

On a vu que $\mathbf{0,2=(1+2^{-1}+2^{-4}+2^{-5}+2^{-8}+2^{-8}+\ldots)\times 2^{-3}}$ ***. En divisant par 2(ou en multipliant par*** $\mathbf{2^{-1}}$ ***) on obtient*** $\mathbf{0,1=(1+2^{-1}+2^{-4}+2^{-5}+2^{-8}+2^{-8}+\ldots)\times 2^{-4}}$ ***. Donc seul l'exposant change (c'est -4).***

Ainsi l'exposant décalé est -4+1 023 = 1 019 qui se code $\mathbf{(01111111011)_2}$ ***.***

Le code au format IEEE 754 de 0,1 est (0 01111111011 10011001100...1010)₂.

On a $0{,}3\times2=0{,}6$ ***;*** $0{,}6\times2=1{,}2$ ***donc*** $0{,}3=1{,}2\times2^{-2}$ ***.***
Le bit correspondant au signe + est 0.
L'exposant décalé est -2+1 023=1 021 qui se code $(01111111101)_2$ ***sur 11 bits.***
La partie décimale de la mantisse est 0,2.
$0{,}2\times2=0{,}4$ ***donc le premier bit est 0.***
$0{,}4\times2=0{,}8$ ***donc le 2è bit est 0.***
$0{,}8\times2=1{,}6$ ***donc le 3è bit est 1.***
$0{,}6\times2=1{,}2$ ***donc le 4è bit est 1.***
Comme 1,2-1=0,2 c'est cyclique, et les bits« 0011 »vont se répéter indéfiniment= (00110011...0011...)$_2$***.***
La mantisse se code (0011 0011 0011...0011 0)$_2$ ***, en arrondissant : (0011 0011 0011...0011)***$_2$***.***

53è bit

Ainsi le code au format IEEE 754 de 0,3 est (0 01111111101 001100110011...0011)$_2$***.***

d. Ouvre Python et tape dans la console : ***>>>(0.1+0.2)-0.3*** puis ***« Entrée »***.
Pourquoi le résultat est-il surprenant ? Comment l'expliquer ?

```
>>> (0.1+0.2)-0.3
5.551115123125783e-17
```

Ce résultat différent de 0 est surprenant car en théorie le résultat devrait être 0.
Ceci s'explique par les arrondis réalisés sur les mantisses au 52è chiffre.

<u>**Ce qu'il faut savoir :**</u>
• Il est impératif de bien distinguer le type <u>***« integer »***</u> du type <u>***« float »***</u> dans un programme. En particulier l'entier 7 et le flottant 7.0 ne se codent pas de la même manière en binaire dans une machine.

• En binaire, sur 8 octets (64 bits), avec <u>***la méthode du complément à 2***</u>, les codes représentent <u>***exactement***</u> les nombres entiers compris entre -9 223 372 036 854 775 808 et 9 223 372 036 854 775 807.
En revanche avec <u>***la norme IEEE 754***</u> (sur 64 bits aussi), les codes représentent <u>***parfois des approximations***</u> des nombres réels compris entre $-1{,}7976931348623157\times10^{308}$ et $1{,}7976931348623157\times10^{308}$.

• Sur machine les calculs avec les nombres entiers sont exacts, mais les calculs avec les nombres flottants peuvent présenter d'infimes erreurs d'arrondi.

B. <u>Exercices d'application :</u>

<u>**Exercice 1 :**</u> Détermine à quels nombres réels correspondent ces codes au format IEEE 754.

a. (1 01111111110 1000...0)$_2$.
signe = -
exposant décalé = 1 022
exposant = 1 022-1 023 = -1
mantisse $=1+2^{-1}=1{,}5$
le réel est $-1{,}5\times2^{-1}=-0{,}75$ ***.***

b. (0 10000001000 0001100101011100...0)$_2$.
signe = +
exposant décalé = 1 032
exposant = 1 032 – 1 023 = 9
mantisse $=1+2^{-4}+2^{-5}+2^{-8}+2^{-10}+2^{-12}+2^{-13}+2^{-14}$ ***=1,09906005859375***
le réel est $1{,}09906005859375\times2^{9}=562{,}71875$ ***.***

c. (0 00000000000 000...0)$_2$.
d'après le cours, le réel est 0.

Exercice 2 : Convertis les nombres suivants au format IEEE 754.
Pour chaque cas, précise si les codes représentent exactement ou approximativement les nombres réels.

a. 3,625. ***b.*** -4,5. ***c.*** -50. ***d.*** $-\frac{11}{15}=-0,7333$ ... ***e.*** 1.

$3,625\div 2^1=1,8125$ ***(1 10000000001 00100...00)₂.*** ***(1 01111111110 01110111...0111)₂.***

$3,625=+1,8125\times 2^1$ ***(1 10000000100 100100...00)₂.*** ***(0 01111111111 00...)₂.***

signe + : 0

expo décalé 1024 : 10...0

partie décimale mantisse 0,8125 :

$0,8125\times 2=1,625$ ***donc bit 1***

$0,625\times 2=1,25$ ***donc bit 1***

$0,25\times 2=0,5$ ***donc bit 0***

$0,5\times 2=1$ ***donc bit 1 et fin***

partie décimale de la mantisse : 110100...00

Donc 3,625 se code (0 10000000000 110100...00)₂.