Séquence : *Représentation binaire approximative d'un réels : les flottants.*	**Numérique et Sciences Informatiques**	***Thème :*** *Représentation des données (types de base)*

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** ».*

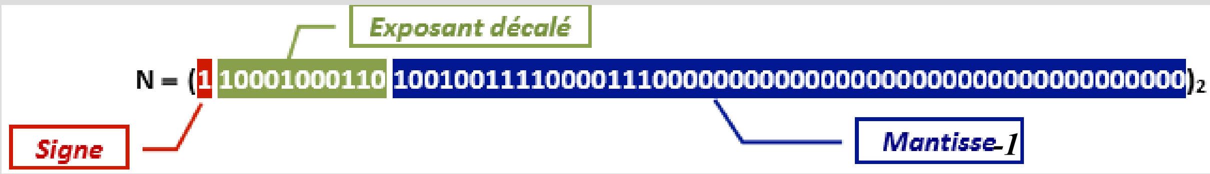
*Si $1\leq d\leq 2\ 046$, **on soustrait 1 023** pour obtenir l'exposant n qui est un entier relatif.*

***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.

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.

b. A quel bit correspond le signe S ?

c. Calcule l'exposant décalé

d. A quel code binaire correspond cet exposant décalé ?

e. Calcule la partie décimale de la mantisse.

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.

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

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.

Activité 2 :

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

b. Pourquoi le même nombre n'est-il pas codé de la même façon ?
Quelle précaution faudra-t-il prendre lorsqu'on écrit un programme (notamment avec Python) ?

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 ?

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 ?

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.

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 ?

Ce qu'il faut savoir :

• Il est impératif de bien distinguer le type ***« integer »*** du type ***« float »*** 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.

• 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. Exercices d'application :

Exercice 1 : Détermine à quels nombres réels correspondent ces codes au format IEEE 754.

a. $(1\ 01111111110\ 1000...0)_2$. ***b.*** $(0\ 10000001000\ 0001100101011100...0)_2$. ***c.*** $(0\ 00000000000\ 000...0)_2$.

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.