Conversions des Flottants
IEEE 754

Ce document a pour objectif de vous familiariser avec les conversions entre plusieurs bases pour les flottants en respectant la norme IEEE 754.

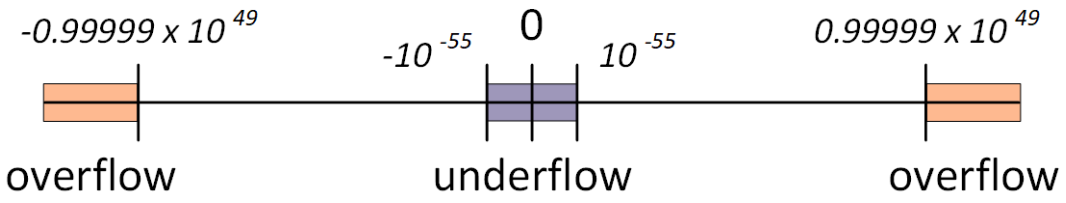
La plupart des conversions que nous effectuerons seront entre les bases 2, 10, et 16 pour les flottants IEEE 754.
Pour rappel, un nombre en indice peut indiquer la base (10_2 indique du binaire, 10_{10} indique du décimal, ...) tout comme un symbole en préfixe (% indique du binaire, et $ indique de l'hexadécimal). Sans symbole particulier, on considère qu'il s'agit de la base 10 usuelle.

1 Représentation des flottants

Les flottants concernent les nombres à virgules. Représenter et manipuler des nombres à virgules n'est pas aisé dans le sens où plusieurs questions se posent : combien de nombres après la virgule faut-il gérer ? Quel est le plus petit pas/décalage supporté par un ordinateur ? ... Ces questions touchent au principal problème rencontré par l'informatique dans le traitement des données scientifiques : la « précision » .

La précision décrit combien de bits seront utilisés pour représenter les nombres flottants. Dans la norme IEEE 754, il existe plusieurs formats pour représenter les flottants, dont la *simple précision* (sur 32 bits), la *double précision* (sur 64 bits), et la *double précision étendue* (sur 80 bits).

Parmi les contraintes que vous aviez déjà vus, un nombre entier codé sur 8 bits ne pourra pas aller au delà de la valeur 256 : si on a %1111 1111 et qu'on lui ajoute 1, alors de la valeur 256, on repassera à 0. Ce phénomène s'appelle un *integer overflow* (dépassement d'entier), qu'il ne faut pas confondre avec les autres types d'*overflows* (stack overflow, buffer overflow). Dans le cas des flottants, ce type de dépassement est également possible, mais, il existe également le cas inverse : l'*underflow* où l'on cherche à représenter une valeur trop petite pour la précision choisie.



(extrait de « *Englander : The Architecture of Computer Hardware and Systems Software* »)

Ainsi, non seulement il existe une limite haute/basse sur la partie entière représentable d'un nombre, mais également sur la quantité de chiffres après la virgule. Les nombres flottants sont donc parfois des approximations lorsqu'ils sont manipulés. En développement, on ne teste *jamais* l'égalité entre deux variables représentées par des nombres flottants, mais uniquement un écart entre elles (si cet écart est suffisamment négligeable, alors elles peuvent être considérées comme égales).

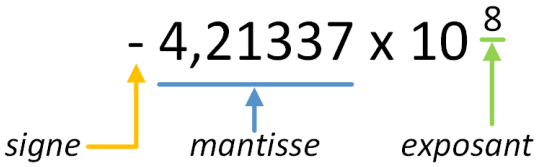
Lorsque l'on travaille sur les nombres à virgules, vous connaissez déjà une forme de notation dite scientifique. On y met un unique entier entre 1 et 9 (inclus), et on l'accompagne d'une virgule suivi d'un nombre, puis on le multiplie par une puissance de 10.

$$-2541,3945 = -2,5413945 \times 10^3$$

Cette notation est parfois simplifiée en utilisant un e pour *exposant*.

$$0,0001337 = 1,337 \times 10^{-4} = 1,337\text{e} - 4$$

Ce format $\pm a \times 10^n$ est également utilisé pour représenter les nombres dits flottants (car la virgule « flotte » selon la puissance de 10 utilisée). Cette notation se compose de trois éléments :



— *signe* : positif ou négatif (ici « - »)
— *exposant* : la puissance de 10 (ici 8)
— *mantisse* (ou significande) : le nombre décimal (ici $4,21337$)

Dans la notation IEEE 754, on utilise ces mêmes concepts, mais appliqués à la base 2 et avec une taille précise pour représenter chacun d'entre eux. Ainsi, la mantisse a une valeur minimale et une valeur maximale, tout comme l'exposant. De plus, le vocabulaire varie légèrement du fait que la norme impose quelques contraintes. On parlera donc dans le vocabulaire formel de :

— *signe* : positif ou négatif
— *exposant biaisé* : la puissance de 2, à laquelle il faut ajouter une valeur (le biais)
— *mantisse* : la partie décimale après la virgule (en ignorant le 1 de la partie entière)

représentation des flottants IEEE 754 simple précision (32 bits)

représentation des flottants IEEE 754 double précision (64 bits)

De plus, étant donné que les représentations numériques ont des limites, on distingue plusieurs cas dans la taille des flottants : la *simple précision* (ou *single precision* en anglais) sur 32 bits, la *double précision* sur 64 bits, et d'autres formats que nous n'étudierons pas. La précision fait varier la taille générale des flottants, ce qui implique que les flottants double précision couvreront plus de grandes valeurs, mais également plus de petites valeurs proches de 0, que les flottants simple précision.

— *signe* : 1 bit
— *exposant* : 8 bits (simple précision), 11 bits (double précision), 15 bits (quadruple precision)
— *mantisse* : 23 bits (simple précision), 52 bits (double précision), 112 bits (quadruple precision)

Enfin, pour convertir les décimaux en flottants IEEE 754, il existe une convention pour les *nombres normalisés* (valeur absolue supérieure à 1), et les *nombre dénormalisés* (valeur absolue inférieure à 1).

2 Valeurs réservées

La norme prévoit également une réservation de certaines valeurs clés. Vous devez les connaître pour pouvoir les reconnaître.

Type	Valeur héxadécimale	Signe	Exposant	Mantisse
+ Zéro	$ 0000 0000	0	%0000 0000	% 000 0000 0000 0000 0000 0000
- Zéro	$ 8000 0000	1	%0000 0000	% 000 0000 0000 0000 0000 0000
+ ∞	$ 7F80 0000	0	%1111 1111	% 000 0000 0000 0000 0000 0000
- ∞	$ FF80 0000	1	%1111 1111	% 000 0000 0000 0000 0000 0000
NaN (borne B)	$ _F81 0000	X	%1111 1111	% 000 0000 0000 0000 0000 0001
NaN (borne H)	$ _FFF 0000	X	%1111 1111	% 111 1111 1111 1111 1111 1111

3 Nombres normalisés

Les nombres normalisés dans le format IEEE 754 concernent les nombres dont la valeur absolue est supérieure à 1 (mais restent inférieurs à la borne maximale gérée par la précision choisie). L'objectif est de représenter les nombres Voici les étapes pour convertir les nombres normalisés :

1. Récupérer le signe du nombre (positif = 0, négatif = 1)
2. Séparer la partie entière de la partie décimale
3. Convertir la partie entière en binaire (sans gérer le signe)
4. Convertir la partie décimale en binaire (sans gérer le signe)
5. Fusion des parties entière et décimale en binaire tout en gardant la virgule
6. Réécriture en notation scientifique base 2
7. Calculer l'exposant pour le format IEEE 754 en l'ajoutant au biais de la précision choisie
8. Convertir l'exposant biaisé en binaire
9. Reporter la mantisse selon la précision choisie

Nous traiterons ici comme exemple la valeur $-42,1337$.

3.1 Récupérer le signe du nombre

On place dans le bit de poids fort servant à coder le signe un 0 si le nombre est positif, ou un 1 si le nombre est négatif. Dans l'ensemble des étapes suivantes, on peut ignorer le signe du nombre.

On retient donc 1 que l'on place au bit 31 en simple précision, et au bit 63 en double précision.
Pour les étapes suivantes, on travaillera donc sur $42,15625$.

3.2 Séparer la partie entière de la partie décimale

On sépare la partie entière de la partie décimale afin de les convertir en binaire chacune de leur façon.

$42,15625$ devient donc 42 et $0,15625$.

3.3 Convertir la partie entière en binaire

On convertit la partie entière en binaire avec l'algorithme classique de division par 2.

Dans les traitements suivants, le 1 de tête sera considéré comme implicite, donc il sera omis lors de l'écriture finale de la mantisse. Pour le traitement suivant concernant la conversion binaire de la partie décimale, il est parfois utile de connaître le nombre de bits utilisés pour représenter cette partie (moins le 1 de tête). Néanmoins, pour connaître le nombre de décalages nécessaires, il est nécessaire de conserver le nombre binaire tel quel.

$42_{10} = 101010_2$

3.4 Convertir la partie décimale en binaire

On convertit la partie décimale en binaire. Pour cela, on effectue des multiplications successives par 2 en faisant l'extraction de la partie entière. En effet, les bits représentent toujours des puissances de 2, mais des puissances négatives (donc 2^{-1}, 2^{-2}, 2^{-3}, ... soit $0,5$, $0,25$, $0,125$, ...).

La condition d'arrêt est soit d'atteindre $1,0$, soit que la taille de la partie entière convertie en binaire + le nombre de multiplication tienne sur la taille de la mantisse exploitée. Dans notre cas, 42 est converti en 101010_2, donc en omettant le 1 de tête, 01010_2, celui-ci prendra 5 bits dans la mantisse. En simple précision, la mantisse étant de 23 bits, on pourra donc placer 18 bits $(23 - 5)$ de nombres après la virgule. À la fin de la dernière multiplication applicable (la 18^e), on arrondit le résultat à 0 ou 1.

multiplication		résultat	entier
$0,15625 \times 2$	=	$0,31250$	0
$0,3125 \times 2$	=	$0,6250$	0
$0,625 \times 2$	=	$1,250$	1
$0,25 \times 2$	=	$0,50$	0
$0,5 \times 2$	=	$1,0$	1

On lit cette fois les multiplications dans l'ordre où elles ont été effectuées.

$0,15625_{10} = 0,00101_2$

3.5 Fusion des parties entière et décimale en binaire tout en gardant la virgule

On fusionne les parties entières et décimales converties en binaire, tout en conservant le placement de la virgule.

$42_{10} = 101010_2$

$0,15625_{10} = 0,00101_2$

$42,15625_{10} = 101010,00101_2$

3.6 Réécriture en notation scientifique base 2

On déplace la virgule pour atteindre le 1 le plus à gauche de la partie entière, et on garde ce décalage comme exposant.

$101010,00101_2 = 1,0101000101 \times 2^5$

3.7 Calculer l'exposant en ajoutant le biais

On ajoute le biais (lié à la précision choisie) à l'exposant précédemment obtenu. En simple précision, le biais est de 127. En double précision il est de 1023. En quadruple précision, il est de 16383.

$1,0101000101 \times 2^5$

Simple précision : $5 + 127 = 132$

Double précision : $5 + 1023 = 1028$

3.8 Convertir l'exposant biaisé en binaire

On convertit en binaire l'exposant, et on le reporte dans le champ prévu.

Simple précision (8 bits) : $132_{10} =$ 1000 0100_2

Double précision (11 bits) : $1028_{10} =$ 100 0000 0100_2

3.9 Reporter la mantisse selon la précision choisie

On reporte la mantisse dans le champ prévu en omettant le 1 en tête : en effet, celui-ci est implicite par l'écriture scientifique en base 2.
De plus, on ajoute autant de 0 que nécessaire à droite du nombre.

$1,0101000101_2 \times 2^5$

0101000101_2

Simple précision (23 bits) :
$01010001010000000000000_2$

Double précision (52 bits) :
$0101000101000000000000000000000000000000000000000000_2$

3.10 Résultat

Enfin, on peut fusionner l'ensemble des champs pour représenter un nombre normalisé au format IEEE 754 : signe (1), exposant (10000100_2 ou 10000000100_2), mantisse

Simple précision (32 bits) :
$11000010001010001010000000000000_2$

Double précision (64 bits) :
$1100000001000101000101000000000000000000000000000000000000000000_2$

Simple précision (32 bits) : $-42.15625_{10} = C228A000_{16}$

Double précision (64 bits) : $-42.15625_{10} = C045140000000000_{16}$

4 Nombres dénormalisés

Les nombres dénormalisés (*subnormal numbers*, *denormalized numbers*, ou *denormals* en anglais) concernent les nombres dont la valeur absolue est strictement inférieure strictement à 1 (c'est-à-dire les nombres tels que : $-1 < n < 1$). Comme ces nombres ont un 0 comme chiffre entier, on ne peut pas les représenter avec une mantisse dont un 1 est implicite.

Dans le format binaire, on reconnait un nombre comme dénormalisé si son exposant est nul (il ne contient que des 0).

Nous ne verrons pas en détails comment obtenir ces nombres, mais vous devrez savoir les repérer.

Ce document et ses illustrations ont été réalisés par Fabrice BOISSIER en novembre 2022