Complément sur la norme IEE754	1 NSI

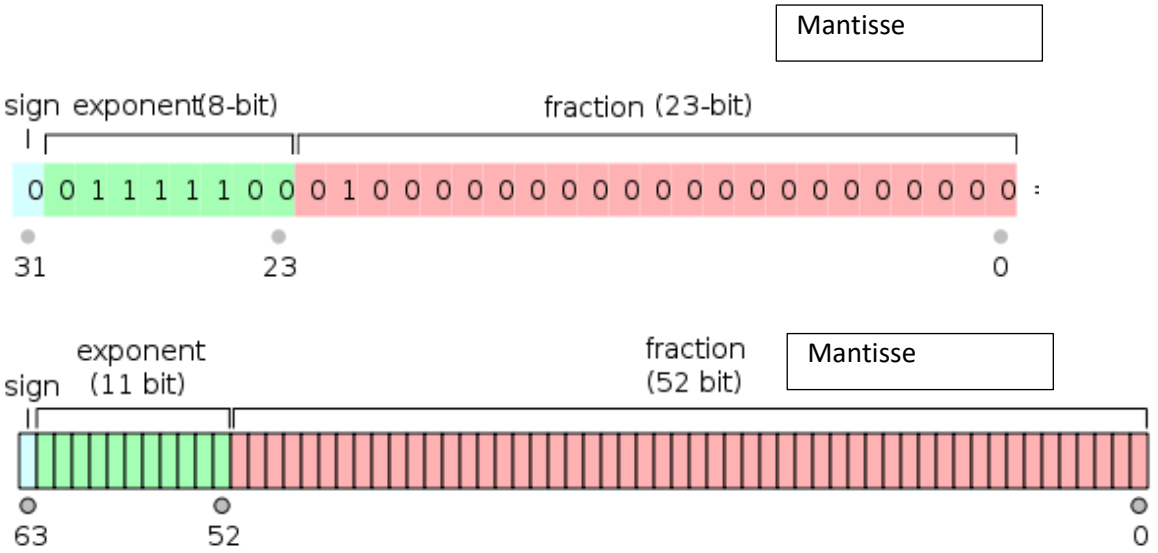
1. Représentation en machine

A la fin des années 70, chaque ordinateur possédait sa propre représentation pour les nombres à virgule flottante. En 1985 la norme IEEE 754 est apparue pour normaliser le codage des nombres.

Il existe deux formats associés à cette norme : le format dit "simple précision" et le format dit "double précision". Le format "simple précision" utilise 32 bits pour écrire un nombre flottant alors que le format "double précision" utilise 64 bits. Dans la suite nous travaillerons principalement sur le format 32 bits.

Que cela soit en simple précision ou en double précision, la norme IEEE754 utilise :

- 1 bit de signe (1 si le nombre est négatif et 0 si le nombre est positif)
- des bits consacrés à l'exposant (8 bits pour la simple précision et 11 bits pour la double précision)
- des bits consacrés à la mantisse (23 bits pour la simple précision et 52 bits pour la double précision)



Nous pouvons vérifier que l'on a bien 1 + 8 + 23 = 32 bits pour la simple précision et 1 + 11 + 52 = 64 bits pour la double précision.

Pour écrire un nombre flottant en respectant la norme IEEE754, il est nécessaire de commencer par écrire le nombre sous la forme 1,XXXXX x 2^e (avec e l'exposant), il faut obligatoirement qu'il y ait un seul chiffre à gauche de la virgule et il faut que ce chiffre soit un "1". Par exemple le nombre "1010,11001" devra être écrit "1,01011001x2^{11}". Autre exemple, "0,00001001" devra être écrit "1,001x2^{-101}".

La partie "XXXXXX" de "1,XXXXX.2^{e}" constitue la mantisse (dans notre exemple "1010,11001" la mantisse est "01011001"). Comme la mantisse comporte 23 bits en simple précision, il faudra compléter avec le nombre de zéro nécessaire afin d'atteindre les 23 bits (si nous avons "01011001", il faudra ajouter 23 - 8 = 15 zéros à droite, ce qui donnera en fin de compte "01011001000000000000000")

Notre première intuition serait de dire que la partie "exposant" correspond simplement au "e" de "1,XXXXX.2e" (dans notre exemple "1010,11001", nous aurions "11"). En fait, c'est un peu plus compliqué que cela. En effet, comment représenter les exposants négatifs ? Aucun bit pour le signe de l'exposant n'a été prévu dans la norme IEEE754, une autre solution a été choisie :

Pour le format simple précision, 8 bits sont consacrés à l'exposant, il est donc possible de représenter 256 valeurs, nous allons pouvoir représenter des exposants compris entre (-126)10 et (+127)10 (les valeurs -127 et +128 sont des valeurs réservées, nous n'aborderons pas ce sujet ici). Pour avoir des valeurs uniquement positives, il va falloir procéder à un décalage : ajouter systématiquement 127 à la valeur de l'exposant. Prenons tout de suite un exemple (dans la suite, afin de simplifier les choses nous commencerons par écrire les exposants en base 10 avant de les passer en base 2 une fois le décalage effectué) :

Repartons de "1010,11001" qui nous donne 1,01011001.2^{3}, effectuons le décalage en ajoutant 127 à 3 : "1,01011001.2^{130}", soit en passant l'exposant en base 2 : "1,01011001.2^{10000010}". Ce qui nous donne donc pour "1010,11001" une mantisse "01011001000000000000000" (en ajoutant les zéros nécessaires à droite pour avoir 23 bits) et un exposant "10000010" (même si ce n'est pas le cas ici, il peut être nécessaire d'ajouter des zéros pour arriver à 8 bits, ATTENTION, ces zéros devront être rajoutés à gauche).

À noter que pour le format double précision le décalage est de 1023 (il faut systématiquement ajouter 1023 à l'exposant afin d'obtenir uniquement des valeurs positives)

Nous sommes maintenant prêts à écrire notre premier nombre au format simple précision : soit le nombre "-10,125" en base 10 représentons-le au format simple précision :

nous avons (10)10 = (1010)2 et (0,125)10 = (0,001)2 soit (10,125)10 = (1010,001)2

Décalons la virgule : 1010,001 = 1,010001.2^{3}, soit avec le décalage de l'exposant 1,010001.2^{130}, en écrivant l'exposant en base 2, nous obtenons 1,010001.2^{10000010}

Nous avons donc : notre bit de signe = 1 (nombre négatif), nos 8 bits d'exposant = 10000010 et nos 23 bits de mantisse = 01000100000000000000000

Soit en "collant" tous les "morceaux" : 11000001001000100000000000000000

Cette écriture étant un peu pénible, il est possible d'écrire ce nombre en hexadécimal : C1220000

Ex

Déterminez la représentation au format simple précision de (-32,75)10 en binaire et en hexadécimal.

Site faisant la conversion
https://www.binaryconvert.com

2. Représentation des flottants dans un ordinateur : IEEE 754 -> decimal

Il est aussi possible de passer d'une représentation au format IEEE 754 à une représentation "classique" en base 10

Soit le nombre flottant au format simple précision :
01111101000000000000000000000000:

00111110100000000000000000000000,

- bit de signe à 0 : nombre positif
- l'exposant décalé : $(01111101)_2 = (125)_{10}$, soit une fois le décalage supprimé, 125 - 127 = -2.
- la mantisse : les 23 bits suivants sont uniquement des zéros, ce qui nous donne en fin de compte : $1{,}000.2^{-2}$. Ce qui donne, en base 10 également $(1{,}000.2^{-2})_{10}$ soit $(0{,}25)_{10}$.

Ex 5 Déterminez la représentation au format simple précision de (0,1)10 en binaire.

Nous avons ici un problème : comme déjà évoqué plus haut, nous nous retrouvons avec une "conversion" qui ne s'arrête jamais (le schéma "0011" se répète à "l'infini"), problème, en simple précision, la mantisse est limitée à 23 bits.

Vous devriez donc obtenir : 00111101110011001100110011001100

Ex 6 : Soit le nombre flottant au format simple précision : 00111101110011001100110011001100.

Trouvez la représentation en base 10 de ce nombre.

La réponse à la question posée ci-dessus est (0,099999994)10, or, en toute logique, nous devrions trouver (0,1)10. Cette "légère" erreur est logique quand on y réfléchit un peu. N'oubliez qu'à cause de la limitation de la mantisse à 23 bits, nous avons dû "tronquer" notre résultat (de toutes les façons, même avec une mantisse beaucoup plus grande, on aurait aussi eu le problème, car le schéma "0011" se répète à l'infini). Cette représentation avec un nombre limité de bits des nombres flottants est un problème classique en informatique. Cela peut entrainer des erreurs d'arrondi dans les calculs qui peuvent être très gênants si on n'y prend pas garde :

Sources Wikipédia / David Roche / Eduscol