Chapitre 3

Série d'exercices n°3 - Théorie chapitre II

3.1 Rappels mathématiques

— Dans le milieu des sciences et de l'ingénieurie, on exprime en général les nombres d'ordres de grandeur très variables que l'on est amené à rencontrer comme le produit d'une valeur en général inférieure à 10, et d'une puissance de 10. Par exemple, on exprime la permittivité électrique du vide ϵ_0 comme environ $8,854 \cdot 10^{-12} Fm^{-1}$. On peut aussi noter 8,854 E-12, comme rencontré parfois dans des logiciels de calcul numérique. On fait de même dans le cadre de la norme IEEE 754, mais en base 2.

— Pour trouver comment exprimer en puissances de 2 des nombres couvrant de grands ordres de grandeur en puissance de 10, on peut utiliser la fonction logarithme, puisque $log_2(A) = log_2(B \cdot 2^n) = log_2(B) + log_2(2^n) = log_2(B) + n$. Par exemple, $log_2(8,854 \cdot 10^{-12}) \approx -36.71$. On essaie donc $\frac{8,854 \cdot 10^{-12}}{2^{-37}}$ et on trouve ≈ 1.21 : on a réussi à déterminer que $\epsilon_0 \approx 1,21 \cdot 2^{-37}$, qu'on peut alors encoder selon la norme IEE754 avec une puissance de -37 et une mantisse normalisée de 1,21 augmentée de toutes les décimales qu'on a négligé ici mais qu'on peut noter dans le format concerné.

3.2 Mode d'emploi

— **Appliquer les mêmes étapes qu'à la session précédente :**
On doit connaître le format (combien de bits ?) et la convention d'encodage choisie. On doit avoir une idée de la faisabilité de l'opération (le résultat d'une multiplication de 2 nombres sur n bits aura-t-il du sens sur moins que $2 \cdot n$ bits ?), et d'à quoi ressemblera le résultat de l'opération s'il y a un dépassement (pour la multiplication ci-dessus, à quoi ressemblera le résultat du produit modulo 2^n ?). Enfin, on doit pouvoir encoder, faire les opérations arithmétiques de base, puis décoder et critiquer le résultat.

— **Aligner les virgules et abandonner les bits de poids faible :**

Pour que les opérations des formats à virgules aient du sens, il faut aligner les nombres selon leur virgule. Cet alignement, dans la plupart des cas, implique la perte d'une partie des nombres encodés. Afin de perdre la partie la moins importante du résultat de l'opération, on aligne toujours de manière à conserver les bits de poids fort, au détriment des bits de poids faible. Ceci est valable en virgule fixe comme selon le standard IEEE 754, où il faudra donc diviser la mantisse de l'opérande le plus petit par l'écart entre l'exposant des deux opérandes.

— **Gérer les mantisses normalisées ou dénormalisées :**
Une des différences importantes du format en virgule flottante selon la norme IEEE 754 est que les bits de la mantisse n'ont pas le même sens selon le nombre encodé. Par exemple, en simple précision, pour les nombres sur l'intervalle $[2^{-126}; 2^{127}]$ la mantisse est dite "normalisée" et indique la présence de puissances de 2 strictement négatives telles que $m = 1 + \sum_{i=0}^{n-1}(b_i * 2^{-i-1})$. Pour les nombres plus petits que 2^{-126}, l'exposant est fixé à une puissance de 2^{-127} et la mantisse est dite "dénormalisée" et indique la présence de puissances de 2 négatives ou nulle telles que $m = \sum_{i=0}^{n-1}(b_i * 2^{-i})$. Pour les nombres plus grands que 2^{127}, la mantisse peut prendre n'importe quelle valeur strictement non-nulle, ou bien être strictement nulle ; dans le premier cas, le nombre sera toujours lu comme "NaN", i.e. "Not A Number". Dans le second, elle sera lue comme un dépassement arithmétique.

3.3 Correctifs des exercices

3.3.1 Exercice supplémentaire 3 (seconde partie)

Énoncé : Quel est le nombre représenté par la suite de bits

$$\boxed{10000000000000000000000000000001}$$

(il y a 30 chiffres 0) dans les représentations :

(e) non signée en virgule fixe, avec autant de bits avant et après la virgule ?

(f) IEEE 754 ?

Solution :

(e) La lecture d'un nombre réel encodé en représentation à virgule fixe est identique à la lecture d'un nombre entier encodé en représentation sans virgule, à ceci près que les puissances de 2 situées après la virgule sont négatives. On peut aussi considérer que le nombre encodé en virgule fixe est égal à l'entier encodé sur le même format, divisé par 2^k, où k est le nombre de bits après la virgule. Puisqu'on nous indique qu'il y a "autant de bits avant et après la virgule", et qu'on compte au total 32 bits, la virgule doit se situer entre les bits 16 et 15 tel que k = 16.

Si l'on considère le nombre comme un entier encodé en représentation non-signée, on lit immédiatement que :

— Le bit de poids fort indique que le nombre comporte la puissance 2^{n-1} où n est le nombre de bits total du format, ici égal à 32. Ce n'est pas un bit de signe, puisqu'on est en représentation non-signée. Le bit de poids fort indique donc la présence de la puissance 2^{31}.

— Le bit du poids faible indique que le nombre comporte la puissance 2^0.

On a donc un entier = $2^{31} + 2^0$ = 2.147.483.648 + 1 = 2.147.483.649. Si l'on divise cet entier par $2^k = 2^{16}$ avec une calculatrice, on peut obtenir comme résultat le réel 32.768,000.02.

Si l'on considère maintenant le nombre comme un réel, tel que les bits après la virgule correspondent à des puissances de 2 négatives, on lit que :

— Le bit de poids fort indique que le nombre comporte la puissance 2^{n-k-1} où n est le nombre de bits total du format et k le nombre de bits après la virgule. On a $n - k = 32 - 16 = 16$. Le bit de poids fort indique donc la présence de la puissance 2^{15}.

— Le bit du poids faible indique que le nombre comporte la puissance 2^{-k}. Puisque on a $k = 16$, on a donc la présence de la puissance 2^{-16}.

On a donc un réel = $2^{15} + 2^{-16} = 32.768 + 1,52... \cdot 10^{-5}$, où on a tronqué le second terme à partir de la 3ème décimale. Si l'on effectue cette somme sur une calculatrice, on peut retrouver le

nombre 32.768,000.02 calculé précédemment. On peut remarquer que ce nombre est lui-même une représentation tronquée du résultat de la somme où $1,52... \cdot 10^{-5}$ a été arrondi en $2 \cdot 10^{-5}$. Ceci est dû aux limitations de l'affichage de la calculatrice. Dans ce cas, il est donc plus exact de conserver la notation $2^{15} + 2^{-16}$ pour exprimer la valeur du nombre encodé.

(f) La lecture d'un nombre encodé selon le standard IEEE 754 implique d'interpreter différemment les différentes sections de la représentation binaire. Puisque nous ne disposons que de 32 bits, ce nombre doit être encodé en simple précision. Dès lors, il faut analyser séparément les trois champs qui suivent :

— Le premier bit de la représentation en partant de la gauche est un bit de signe. Puisqu'il est à 1, il indique que le nombre représenté est négatif.

— Les 8 bits qui suivent indiquent la valeur de l'exposant. Ils sont tous nuls, ce qui indique un exposant de $0 - (2^{n-1} - 1)$ où n est le nombre de bits de l'exposant, donc $-(2^7 - 1) = -127$. Dans ce cas, il faut également se rappeller que la mantisse est dite "dénormalisée", ce qui conditionne l'interpretation des bits restants.

— Les 23 bits restants indiquent la valeur de la mantisse. Puisque le champ indiquant la valeur de l'exposant est nul, nous sommes dans le cas spécifique d'une mantisse dénormalisée. Ceci implique qu'il faut lire la mantisse comme un entier en représentation non-signée qui doit être divisé par 2^{k-1}, où k est le nombre de bits de la mantisse. On peut également l'expliquer comme un réel à virgule fixe, tel que la virgule est placée juste après le premier bit, et que les bits à droite de la virgule indiquent des puissances négatives de 2. Puisque seul le bit de poids faible est non-nul, on a donc une mantisse de $\frac{1}{2^{22}} = 2^{-22}$.

Au total, on a donc un réel dont la valeur se calcule comme $(-1) \cdot 2^{-127} \cdot 2^{-22}$ qui vaut -2^{-149}. On remarque que c'est le plus grand nombre négatif représentable en simple précision. De manière équivalente, c'est le nombre négatif non-nul de plus petite valeur absolue encodable en simple précision. En base 10, ce nombre vaut approximativement $-1.4 \cdot 10^{-45}$.

3.3.2 Exercice supplémentaire 7

Énoncé : Le nombre suivant est représenté en notation binaire non signée :

$$\boxed{10000000000000000000000000000001}$$

(Il y a 30 bits égaux à 0 entre les deux égaux à 1.)

1. Encoder ce nombre dans le format IEEE 754 simple précision, le plus précisément possible.

2. Après cette conversion, quelle est la valeur exacte du nombre réellement représenté ?

3. Si l'on convertissait à nouveau ce nombre vers la notation binaire non signée, quelle serait la différence entre le résultat obtenu et le nombre original ?

Solution :

(1) Comme vu dans l'exercice supplémentaire 4, on peut immédiatement lire ce nombre comme un entier encodé en représentation non-signée sur 32 bits égal à $2^{31} + 2^0 = 2.147.483.648 + 1 = 2.147.483.649$. On peut ramener ce nombre à un produit d'un nombre inférieur à 10 et d'une puissance de 10 tel que $2.147.483.649 = 2,147.483.649 \cdot 10^9$. De la même manière, on peut le ramener en un produit d'un nombre inférieur à 2 et d'une puissance de 2. Pour trouver la valeur de l'exposant, on peut calculer $log_2(2.147.483.649) \approx 31$ (ce résultat est trivial puisqu'on savait que le nombre s'exprime comme une somme comportant comme plus haute puissance de 2 le terme 2^{31}). On a bien que $\frac{2^{31}+2^0}{2^{31}} = 1 + 2^{-31}$ est compris entre 1 et 2, ce qui correspond au cas d'une mantisse qui peut être normalisée.

Sachant cela, on peut l'encoder au format IEEE 754 en simple précision comme suit :

— Le bit de signe, à l'extrémité gauche du format, est fixé à 0 puisque le nombre est positif (ce qui est obligatoire vu qu'on l'a originellement interpreté selon une convention non-signée).

— Pour encoder l'exposant, on représente au format non-signé sur 8 bits la valeur réelle de la puissance de 2 à laquelle on ajoute 127. On doit donc représenter ici 31 + 127 = 158. Pour ce faire, on opère comme dans la série d'exercices sur les encodages non-signés. On a $158 = 2^7 + 2^4 + 2^3 + 2^2 + 2^1$, donc les bits 7,4,3,2 et 1 du format sont à 1 tandis que les autres sont à 0. Vu que les bits de l'exposant ne sont pas soit tous nuls soit tous à 1, on a une mantisse normalisée. On a la représentation suivante :

$$\boxed{10011110}\ \textit{(exposant IEEE 754 simple précision)}$$

— Les 23 bits restants indiquent la valeur de la mantisse. Dans le cas d'une mantisse normalisée, on peut encoder ce champ en lui soustrayant 1, puis en mulipliant le résultat par 2^k, où k est le nombre de bits de la mantisse, ici égal à 23, et en le considérant comme

nu entier non-signé, ou en lui soustrayant 1 puis en le considérant comme un réel à virgule fixe, tel que la virgule est placée juste avant le premier bit, et que les bits à droite de la virgule (tous les bits, donc) indiquent des puissances négatives de 2. Vu comme un entier, on effectue $(1 + 2^{-31} - 1) \cdot 2^{23}$ et on espère obtenir un entier représentable sur 23 bits. Malheureusement, $2^{-31+23} = 2^{-8}$ n'est lui-même pas un entier et n'est donc strictement pas représentable selon cette convention. Vu comme un réel à virgule fixe, on a que $(1 + 2^{-31} - 1) = 2^{-31}$ n'est strictement pas représentable sur les 23 bits de la mantisse normalisée, puisqu'on a accès au plus bas à la puissance de 2^{-23}. Dès lors, on est contraint d'abandonner cette partie du nombre dans sa représentation en simple précision. On a la représentation suivante :

00000000000000000000000	*(mantisse IEEE 754 simple précision)*

On a donc au total la représentation suivante :

0	10011110	00000000000000000000000

(2) On peut relire ce nombre encodé au format IEEE 754 en appliquant les étapes d'encodage à l'envers, comme suit :

— Le premier bit en partant de la gauche, le bit de signe, est à 0. On a donc un nombre positif.

— Les 8 bits qui suivent indiquent un exposant encodé en représentation non-signée sur 8 bits, tel que la valeur réellement encodée est égale à la valeur représentée - 127. On lit une valeur représentée de $2^7 + 2^4 + 2^3 + 2^2 + 2^1 = 158$ donc l'exposant vaut $158 - 127 = 31$. Le nombre est élevé à la puissance 2^31. Vu que les bits de l'exposant ne sont pas soit tous nuls soit tous à 1, on a une mantisse normalisée.

— Les 23 bits restants indiquent la valeur de la mantisse. Dans le cas d'une mantisse normalisée, on peut lire ce champ comme un entier en représentation non-signée auquel on doit ajouter 2^k et qui doit être divisé par 2^k, où k est le nombre de bits de la mantisse, ici égal à 23. On peut également l'expliquer comme un réel à virgule fixe auquel on doit ajouter 1, tel que la virgule est placée juste avant le premier bit, et que les bits à droite de la virgule (tous les bits, donc) indiquent des puissances négatives de 2. Vu comme un entier, on lit 0, auquel on doit ajouter 2^{23} puis qu'on doit diviser par 2^{23}. On a donc 1. Vu comme un réel à virgule fixe, on fait directement la somme des puissances négatives à laquelle on ajoute 1, tel qu'on a 1.

On assemble les trois champs et on a une valeur totale de $(+1) \cdot 2^{31} \cdot 1$, qui en base 10 vaut 2.147.483.648.

(3) Pour convertir ce nombre au format non-binaire, on doit encoder 2^{31} sur 32 bits, ce qui donne la représentation suivante :

$$\boxed{10000000000000000000000000000000}$$

En toute logique vu les explications de l'étape d'encodage au format IEEE 754 où on a été contraint d'abandonner le terme égal à 2^0, on a bien une différence de 1 entre la représentation du nombre avant et après nos manipulations.

3.3.3 Exercice supplémentaire 14

Énoncé :

1. Représenter le nombre -16 sur 8 bits par valeur signée et par complément à deux.
2. Représenter le nombre $-1{,}25$ en virgule fixe par complément à deux, avec 4 chiffres avant la virgule et 4 chiffres après la virgule.
3. Représenter le nombre 2^{-128} dans le format IEEE754 en simple précision, en exprimant le résultat en hexadécimal.
4. Exprimer la valeur du nombre entier dont la représentation par complément à deux forme la suite de bits $b_3b_2b_1b_0$.
5. Effectuer l'addition $(-2) + (-5)$ en représentant les nombres par complément à un sur 4 bits.

Solution :

(1) Comme vu lors de la série d'exercices précédente, on encode un nombre en représentation signée sur 8 bits en encodant sa représentation non-signée sur 7 bits, c'est-à-dire la représentation de sa valeur absolue, et en y affixant un bit de signe égal à 1 si le nombre est négatif ou 0 s'il est positif.

On décompose le nombre en puissances de 2 tel que $16 = 2^4$. Le bit 4 de la représentation non-signée est donc fixé à 1, et tous les autres à 0. On a la représentation non-signée de la valeur absolue de -16 sur 8 bits :

$$\boxed{00010000}\ \textit{(ns)}$$

On fixe le bit de signe égal à 1 puisqu'on encode un nombre négatif, et on a la représentation suivante :

$$\boxed{10010000}\ \textit{(s)}$$

À partir de la représentation non-signée de la valeur absolue d'un nombre négatif, pour en obtenir la représentation en complément à deux, on doit complémenter chacun de ses bits (y compris le bit de signe de la représentation en valeur absolue qui était à 0, ou on le fixe simplement directement à 1), puis ajouter +1 au résultat. On a donc successivement :

représentation non-signée de la valeur absolue :	00010000
on complémente chacun des 8 bits :	11101111
on ajoute +1 :	11110000

Et on obtient que la représentation de -16 en complément à 2 est la suivante :

$$\boxed{11110000}\ \textit{(c2)}$$

(2) On détermine tout d'abord que s'il y a 4 chiffres avant et après la virgule, c'est qu'on est sur un support de 8 bits où la virgule se situe entre les bits 4 et 3. Comme vu lors des exercices précédents, on décompose tout d'abord la valeur absolue du nombre en puissances de 2, dont certaines peuvent être négatives. On a $1.25 = 2^0 + 2^{-2}$, donc les bits 4 et 2 de la représentation non-signée en virgule fixe sont fixé à 1, et tous les autres à 0. On a la représentation non-signée de la valeur absolue de -1.25 sur 8 bits suivante :

$$\boxed{00010100}\ \textit{(ns)}$$

À partir de la représentation non-signée de la valeur absolue du nombre négatif, on complémente chaque bit et on ajoute +1 tel qu'on a successivement :

représentation non-signée de la valeur absolue :	00010100
on complémente chacun des 8 bits :	11101011
on ajoute +1 :	11101100

Et on obtient que la représentation de -1.25 en complément à 2 est la suivante :

$$\boxed{11101100}\ \textit{(c2)}$$

(3) Comme vu dans les exercices précedents, on cherche d'abord à normaliser le nombre à encoder au format IEEE 754. En simple précision, le plus petit exposant auquel nous avons accès est -127. Dans ce cas, la mantisse est nécessairement dénormalisée.

Sachant cela, on peut encoder 2^{-128} au format IEEE 754 en simple précision comme suit :

— Le bit de signe, à l'extrémité gauche du format, est fixé à 0 puisque le nombre est positif.

— L'exposant est par défaut fixé à la valeur de -127, ce qui correspond à fixer tous les bits du champs à 0, tel que la valeur de l'exposant soit bien de $0 - 127 = -127$. Dans ce cas, on a une mantise dénormalisée. On a la représentation suivante :

$$\boxed{00000000}\ \textit{(exposant IEEE 754 simple précision)}$$

— Dans le cas d'une mantisse dénormalisée, chaque bit de la mantisse indique la présence d'une puissance de 2 négative ou nulle. On a que $\frac{2^{-128}}{2^{-127}} = 2^{-1}$, donc il faut fixer à 1 non pas le premier bit, mais le second bit en partant de la gauche de la mantisse. On a la représentation suivante :

$$\boxed{01000000000000000000000}\ \textit{(mantisse IEEE 754 simple précision)}$$

On a donc au total la représentation suivante :

0	00000000	01000000000000000000000

Pour passer d'une représentation binaire à une représentation hexadécimale, il faut grouper chaque bloc successif de 4 bits et le convertir en sa représentation hexadécimale, de 0 à F. On a donc :

0000	0000	0010	0000	0000	0000	0000	0000
0	0	2	0	0	0	0	0

Et le nombre s'exprime en représentation IEEE 754 hexadécimale comme 0x00200000.

(4) Il faut tout d'abord différencier la situation selon le signe du nombre encodé :

— Si b3 = 0, le nombre encodé est positif. Dès lors, les bits restant de sa représentation indiquent sa valeur absolue et peuvent être lus immédiatement tels que le nombre entier sera égal à $b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$. Par exemple, 0b 0101 sera lu comme $1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 2^2 + 2^0 = 5$.

— Si b3 = 1, le nombre encodé est négatif. Dès lors, les bits restants doivent être complémentés, puis il faut leur ajouter 1 afin d'obtenir la représentation de la valeur absolue de ce nombre négatif. Dans ce cas, le nombre entier sera donc égal à $(-1) \cdot ((1 - b_2) \cdot 2^2 + (1 - b_1) \cdot 2^1 + (1 - b_0) \cdot 2^0) + 1)$. Par exemple, 0b 1101 sera lu comme $(-1) \cdot ((1-1) \cdot 2^2 + (1-0) \cdot 2^1 + (1-1) \cdot 2^0 + 1) = (-1) \cdot (2^1 + 1) = -3$. On peut aussi utiliser la définition du format en complément à 2 et exprimer que le nombre lu vaut $-2^3 + (b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0)$. On a bien que 0b 1101 est lu comme $-2^3 + 5 = -8 + 5 = -3$.

(5) Comme vu lors de la série d'exercices précédente, on encode un nombre en complément à un en encodant la représentation non-signée de sa valeur absolue, puis en complémentant chaque bit de celle-ci.

On a donc pour -2 :

représentation non-signée de la valeur absolue :	0010
on complémente chacun des 4 bits :	1101

Et pour -5 :

représentation non-signée de la valeur absolue :	0101
on complémente chacun des 4 bits :	1010

On effectue l'addition :

$$\begin{array}{cccccc} \boxed{1} & & & & & \\ & 1 & 1 & 0 & 1 \\ + & 1 & 0 & 1 & 0 \\ \hline & 0 & 1 & 1 & 1 \end{array}$$

Puisque nous sommes en complément à un, il faut prendre garde au report sur du bit de signe : puisqu'il y a un report au bit n, il faut ajouter +1 au résultat final.

$$\begin{array}{rcccc} & \boxed{1} & \boxed{1} & \boxed{1} & \\ & 0 & 1 & 1 & 1 \\ + & 0 & 0 & 0 & 1 \\ \hline & 1 & 0 & 0 & 0 \end{array}$$

On obtient comme résultat un nombre négatif, dont il faut complémenter les bits pour obtenir la valeur absolue. On a :

représentation en complément à un :	1000
on complémente chacun des 4 bits :	0111

La valeur absolue se lit donc comme $2^2 + 2^1 + 2^0 = 7$ et le nombre encodé est donc -7, ce qui correspond au résultat de l'addition de -5 et -2.