Chapitre 4

Série d'exercices n°4 - Théorie chapitre II

4.1 Correctifs des exercices

4.1.1 Exercice supplémentaire 6

Énoncé : Donner une représentation des nombres suivants dans le format IEEE 754 simple précision. (S'il existe plusieurs représentations, ou aucune, le signaler.)

1. $\frac{23}{8}$
2. 2^{-133}
3. -10^{-50}
4. 0
5. NaN

Solution :

(1) On commence par exprimer le réel en nombre décimal : on a $\frac{23}{8} = 2,875$. Ce nombre ne semble ni être si grand qu'il faille l'exprimer comme un produit d'une puissance 2^{128} ou supérieure, ni si petit qu'il faille l'exprimer au moyen d'un 2^{-127}. On doit donc être dans un cas tout à fait standard d'une valeur "moyenne" encodable en simple précision avec une mantisse normalisée.

Afin de l'encoder au format IEEE 754 en simple précision, on construit successivement les champ du signe, de l'exposant et la mantisse comme suit :

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

— Pour encoder l'exposant, on cherche la puissance de 2 qui pourrait permettre d'obtenir une mantisse normalisée, c'est-à-dire la puissance n tel que $\frac{23}{8} = (2^n) \cdot B$ où B est un

nombre inclus dans l'intervalle $[1;2[$. Pour ce faire, on calcule $log_2(\frac{23}{8}) \approx 1.52$. On détermine donc que la bonne puissance se trouve parmi les candidats $n = \{1, 2\}$. On calcule les quotients de $\frac{23}{8}$ et de 2^1 et 2^2 et on trouve que $\frac{23}{8} = (2^1) \cdot 1,4375$. Cette expression répond à nos exigences et peut être encodée au moyen d'une mantisse normalisée. Afin d'encoder l'exposant $n = 1$, on doit représenter $1 + 127 = 128$. On a $128 = 2^7$, donc sur l'encodage 8 bit non signé de l'exposant, le bit 7 est à 1 tandis que les autres sont à 0. On a la représentation suivante :

$\boxed{10000000}$ *(exposant IEEE 754 simple précision)*

— Comme on travaille avec une mantisse normalisée, on doit encoder le nombre B calculé ci-dessus après lui avoir soustrait 1. On a que $B - 1 = 0,4375$ qu'on décompose en puissances strictement négatives de 2 tel que $B - 1 = 2^{-2} + 2^{-3} + 2^{-4}$; on remarque que ce nombre peut être parfaitement encodé sur le format simple précision, sans aucune erreur d'arrondi. On a la représentation suivante :

$\boxed{01110000000000000000000}$ *(mantisse IEEE 754 simple précision)*

On a donc au total la représentation suivante :

0	10000000	01110000000000000000000

(2) Ce nombre est immédiatement lisible comme une puissance de 2. Cette puissance de -133 est inférieure au minimum de -126 pour permettre une mantisse normalisée ; on aura donc nécessairement une mantisse dénormalisée.

Afin de l'encoder au format IEEE 754 en simple précision, on construit successivement les champ du signe, de l'exposant et la mantisse comme suit :

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

— Il est impossible d'encoder ce nombre A comme un produit $A = (2^n) \cdot B$ où n est dans l'intervalle $[-126; 127]$, dès lors on est nécessairement en mantisse dénormalisée. En toute généralité, on peut calculer $log_2(2^{-133}) = -133$ pour déterminer l'exposant nécessaire pour exprimer ce réel. Vu qu'on est dans un cas de mantisse dénormalisée, on est contraint d'utiliser la puissance la plus petite de 2^{-127} représentable dans le champ de l'exposant comme $-127 + 127 = 0$. On a la représentation suivante :

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

— On travaille avec une mantisse dénormalisée, et les bits du champ de la mantisse expriment donc des puissances de 2 non strictement négatives. Pour arriver à 2^{-133} on a besoin de la puissance de 2 égale à $-133 - (-127) = -6$. Dès lors, en se rappellant qu'en mantisse dénormalisée le premier bit à gauche exprime la puissance 2^0, on a besoin de fixer à 1 le 7ème bit en partant de la gauche pour exprimer que la mantisse est égale à 2^{-6}. On a la représentation suivante :

$$\boxed{00000010000000000000000} \textit{ (mantisse IEEE 754 simple précision)}$$

On a donc au total la représentation suivante :

0	00000000	00000010000000000000000

(3) À première vue, on peut s'attendre à ce que le nombre -10^{-50} pose des problèmes d'encodage ; on a que $2^{10} \approx 10^3$ et donc on s'attend à avoir besoin d'une puissance de 2 au moins égale à $-50 \cdot 3 = -150$, ce qui est strictement innaccessible en simple précision.

— Le bit de signe, à l'extrémité gauche du format, est fixé à 1 puisque le nombre est négatif.

— On peut calculer $log_2(10^{-50}) \approx -166$ pour déterminer l'exposant nécessaire pour exprimer ce réel. On a bien besoin d'une puissance de 2 inférieure à -150, et largement inférieure au maximum de -127 accessible en mantisse normalisée en simple précision. On fixe donc par défaut un exposant de -127, tel qu'on représente la valeur $-127 + 127 = 0$ qui correspond à remplir entièrement de 0 le champ de l'exposant, comme suit :

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

— On est dans le cas d'un nombre trop petit pour être représenté en mantisse normalisée. On ne peut donc que chercher à le représenter avec une mantisse dénormalisée, où on encode bien la quantité B du développement $A = B \cdot 2^n$. Puisqu'on ne peut fixer la puissance n qu'à -127 dû aux limitations du format simple précision, il nous reste à représenter $B = \frac{10^{-50}}{2^{-127}} \approx 1,7 \cdot 10^{-12}$. Si l'on cherche à encoder ce nombre en puissances de 2 inclues sur l'intervalle $\{0, -22\}$, on se retrouve face à une impasse : en effet, $log_2(1,7 \cdot 10^{-12}) \approx -39$, ce qui revient à dire qu'on à réussit à encoder les 127 premières puissances de 2 du total de 166 calculé plus haut, mais qu'il en reste 39 avant d'atteindre le nombre à représenter dans la mantisse. Cependant, la mantisse dénormalisée en simple précision représente au mieux la plus petite quantité de 2^{-22}.
On pourrait être tenté de fixer l'entiereté de la mantisse à 0 sauf pour le dernier bit indiquant la présence de la puissance 2^{-22} afin de représenter ce nombre comme la plus petite quantité représentable en simple précision, qui sera alors égale à $(-1) \cdot 2^{-127} \cdot 2^{-22} = -2^{-149} \approx -1,4 \cdot 10^{-45}$. Cependant, procéder de la sorte nous mène à une erreur d'approximation de $-10^{-50} - (-1,4 \cdot 10^{-45}) \approx -1,4 \cdot 10^{-45}$! Dans ce cas, on aurait la représentation suivante :

$$\boxed{00000000000000000000001} \textit{ (mantisse IEEE 754 simple précision)}$$

Afin de se limiter à la plus petite erreur d'approximation possible, on va préférer représenter ce nombre comme un "vrai" zéro du point de vue du format simple précision, en annulant tous les bits de la mantisse. De cette manière, on commet une erreur d'approximation de $-10^{-50} - 0 = -10^{-50}$, ce qui est une erreur bien plus petite que celle calculée au paragraphe précédent. On a donc la représentation suivante :

$$\boxed{00000000000000000000000} \textit{ (mantisse IEEE 754 simple précision)}$$

On a au total la représentation suivante, qui exprime en réalité une quantité de "zéro négatif", c'est-à-dire une quantité trop petite pour être encodée au format simple précision, mais dont on conserve le signe négatif à toutes fins utiles. (Par exemple, ce signe peut être celui du résultat d'un calcul de convergence vers zéro par les négatifs)

1	00000000	00000000000000000000000

(4) Pour représenter le zéro au format IEEE 754, il faut se rappeller qu'une mantisse normalisée encode un nombre A tel qu'on représente $A - 1$. Dès lors, annuler tous les bits de la mantisse ne suffit pas ; cela correspondrait au nombre 1 élevé à la puissance indiquée par le champ de l'exposant.

— Le bit de signe peut être fixé à 0 ou a 1 selon si l'on a un 0 "positif" ou "négatif". Ceci permet d'indiquer des quantités trop petites pour être représentées sur le format choisi tout en conservant leur signe qui peut être d'une importance capitale. On a ici aucune information sur un éventuel signe du "presque zéro", habituellement noté 0^- et 0^+. Les représentations positive et négative sont donc tout aussi valables.

— Afin d'encoder un zéro, on doit explicitement éviter une mantisse normalisée qui encode automatiquement la valeur de $+1$ dans le nombre représenté. On se place donc au plus petit exposant de -127 pour utiliser une mantisse dénormalisée et représenter un vrai zéro. On représente la valeur $-127 + 127 = 0$ qui correspond à remplir entièrement de 0 le champ de l'exposant, comme suit :

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

— Pour fixer le nombre représenté à zéro, on doit strictement annuler la mantisse dénormalisée. On a donc la représentation entièrement nulle suivante :

$$\boxed{00000000000000000000000}\ \textit{(mantisse IEEE 754 simple précision)}$$

Au total, on a donc deux représentations valables à défaut d'avoir plus d'information sur le signe du zéro :

0	00000000	00000000000000000000000
1	00000000	00000000000000000000000

(5) Afin d'encoder un *NaN*, on doit simplement suivre les consignes du standard IEEE 754 associés à ce cas de figure. On rappelle que le *NaN* est utilisé pour encoder des résultats qui ne sont pas formellement définis comme des nombres réels, i.e. des cas "pathologiques".

— Le bit de signe peut être fixé à 0 ou a 1 selon si l'on a un 0 "positif" ou "négatif", comme pour le "zéro". Associer un signe à un résultat qui n'est formellement pas défini comme un nombre est certainement discutable d'un point de vue mathématique, cependant il peut dans certains cas donner une information de signe réellement utile. Par exemple, il peut être intéressant de distinguer le signe du le résultat d'une division par zéro afin d'obtenir des informations sur les opérandes de cette division ; l'utilisation du signe d'un *NaN* ne fait pas partie du standard, mais peut être précisé par certaines implémentations. Vu qu'on a ici pas d'information explicite de signe, les deux représentations sont a priori valables.

— Afin d'encoder un *NaN*, le standard IEEE 754 nous indique de fixer l'exposant à sa valeur maximum, comme pour le cas d'un dépassement arithmétique rencontré ci-dessus. On a donc la représentation suivante :

$$\boxed{11111111} \textit{ (exposant IEEE 754 simple précision)}$$

— Afin de différencier un dépassement arithmétique d'un *NaN*, le standard IEEE 754 indique qu'au moins un bit de la mantisse doit être non-nul. D'après cette définition, on a donc $2^{23} - 1$ représentations différentes de la mantisse d'un *NaN* en simple précision, i.e. tous les nombres encodables sur 23 bits sauf le zéro. On a donc **n'importe quelle** représentation **SAUF** la suivante :

$$\boxed{00000000000000000000000} \textit{ (mantisse IEEE 754 simple précision)}$$

Au total, on a donc $2^{23} - 1$ représentations valables du *NaN* en simple précision, voir $2^{24} - 1$ si l'on prend en compte l'ambiguité sur le signe qui rend aussi valable une représentation positive que négative, à défaut de plus de précision.

Si l'on note "X" les bits dont la valeur n'a aucune importance, on a donc les représentations suivantes, où le 1 peut se déplacer n'importe où dans la mantisse :

X	11111111	1XXXXXXXXXXXXXXXXXXXXXX

4.1.2 Exercice supplémentaire 8

Énoncé : Un format imaginaire semblable à la norme IEEE 754 attribue 7 bits à l'exposant et 16 bits à la mantisse. Quel serait le plus petit nombre strictement positif représentable dans ce format ?

Solution :

On peut approcher ce problème selon le même schéma que les exercices sur la norme IEEE 754, en cherchant à construire la plus petite valeur possible avec des champs d'exposant et de mantisse de longueur différente. On procède comme suit :

— Puisqu'on cherche à représenter le plus petit nombre positif représentable, on fixe le bit de signe à 0. Ce champ n'est pas modifié par rapport au standard IEEE 754.

— Si ce format imaginaire suit les principes de la norme IEEE 754, on peut supposer qu'il encode les exposants comme une puissance de n de 2 à laquelle on doit ajouter $2^{k-1}-1$ où k est le nombre de bits de l'exposant. Pour une mantisse à 8 bits, cette règle nous indique bien qu'il faut encoder un exposant $e = n + 2^{8-1} - 1 = n + 127$. Pour une mantisse à 7 bits, cette règle nous indique qu'il faut encoder un exposant $e = n + 2^{7-1} - 1 = n + 63$. Dès lors, si l'on fixe tous les bits de l'exposant à 0 afin d'employer la plus petite puissance de 2 accessible sur ce format à 7 bits, on aura $e = 0 = n + 63 \Leftrightarrow n = -63$. Autrement dit, le nombre A encodé comme $A = B \cdot 2^n$ sera $A = B \cdot 2^{-63}$. On suppose également que dans ce cas, comme pour la norme IEEE 754, on emploie une mantisse dénormalisée.

— Si l'on se place dans le cadre d'une mantisse dénormalisée sur un format à 16 bits, chaque bit encode une puissance n de 2^n nulle ou négative tel que n = {0,-1,...,-15}. Dès lors, la plus petite quantité strictement positive, c'est-à-dire non-nulle, représentable sur ce format est le bit le plus à droite encodant la puissance 2^{-15} ; on fixe ce bit à 1, et tous les autres à 0, pour représenter le plus petit nombre accessible sur ce format.

Au total, on a assemble donc un nombre égal à $(+1) \cdot 2^{-63} \cdot 2^{-15} = 2^{-78}$. C'est la plus petite quantité positive et non-nulle représentable sur ce format fictif où l'exposant occupe 7 bits et la mantisse occupe 16 bits. Ce nombre vaut environ $3.3 \cdot 10^{-24}$.

4.1.3 Exercice supplémentaire 12

Énoncé :

1. Quelle est la représentation sur n bits (avec $n \geq 2$) du nombre -1 par les procédés
   (a) par complément à un ?
   (b) par complément à deux ?
   (c) en virgule fixe par complément à deux avec 1 bit après la virgule ?
2. Calculer la somme $-2+(-1)$ à l'aide de la représentation par complément à un des entiers sur 4 bits.

Solution :

(1) Pour représenter un nombre négatif en complément à un, en toute généralité, on peut encoder sa valeur absolue puis complémenter chacun de ses bits, y compris le bit le plus à gauche de format qui passera de 0 à 1 et sera directement à la bonne valeur pour un nombre négatif dans une représentation comprenant un bit de signe.

Sur n bits, on peut représenter la valeur absolue de |-1| = 1 comme suit :

$$\boxed{0...01} \textit{ (valeur absolue)}$$

Si l'on complémente chacun des bits de ce format, on a immédiatement la représentation de -1 en complément à un sur n bits en changeant l'unique 1 en 0 et tous les autres 0 en 1 :

$$\boxed{1...10} \textit{ (complément à un)}$$

Pour obtenir la représentation en complément à deux, on doit effectuer les étapes pré-citées, puis ajouter +1 au résultat. Effectuer cette opération sur la représentation en complément à un est immédiate, puisqu'elle revient à fixer le premier bit originellement à 0 à 1 :

$$\boxed{1...11} \textit{ (complément à deux)}$$

Pour obtenir la représentation de -1 en complément à deux en virgule fixe avec 1 bit après la virgule, on doit appliquer les étapes explicitées ci-dessus à partir de l'encodage de la valeur absolue en virgule fixe. Pour représenter |-1| = 1, on doit fixer le premier bit à gauche de la virgule à 1 et tous les autres à 0. Puisqu'on a 1 bit après la virgule, ceci correspond à fixer à 1 le second bit en partant de la droite. On a la représentation suivante :

$$\boxed{0...010} \textit{ (valeur absolue, virgule fixe à 1 bit après la virgule)}$$

On complémente chaque bit pour obtenir la représentation en complément à un :

$\boxed{1...101}$ *(complément à un, virgule fixe à 1 bit après la virgule)*

On ajoute +1 pour obtenir la représentation en complément à deux. Lorsqu'on effectue cette opération, on a un report du premier bit sur le second, tel qu'on obtient finalement :

$\boxed{1...110}$ *(complément à deux, virgule fixe à 1 bit après la virgule)*

(2) Pour calculer cette somme en complément à un, on doit d'abord obtenir la représentation en complément à un sur 4 bits des deux opérandes. En suivant les étapes présentées dans la première partie de cette question, on a successivement :

0010	valeur absolue de -2 = 2
1101	complément à un de -2

0001	valeur absolue de -1 = 1
1110	complément à un de -1

On peut alors effectuer l'addition comme selon les règles habituelles de calcul écrit :

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

Comme 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.

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

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 :	1100
on complémente chacun des 4 bits :	0011

La valeur absolue se lit donc comme $2^1 + 2^0 = 3$ et le nombre encodé est donc -3, ce qui correspond au résultat de l'addition de -2 et -1.

4.1.4 Exercice supplémentaire 13

Énoncé :

1. Quel est le nombre représenté par la constante `0x80000001` dans les représentations
   — par valeur signée ?
   — par complément à un ?
   — par complément à deux ?
   — IEEE754 simple précision ?
2. Quels sont les nombres dont les représentations sur n bits par complément à un et par complément à deux sont égales ?
3. En utilisant la notation hexadécimale, donnez une représentation d'une valeur indéterminée (*Not A Number*) dans le système IEEE754 en double précision.

Solution :

(1) On commence par traduire la valeur hexadécimale en valeur binaire. Pour ce faire, on transforme chaque chiffre individuel en base 16 en sa représentation en 4 bits en base 2. On a la conversion suivante :

8	0	0	0	0	0	0	1	encodage hexadécimal
1000	0000	0000	0000	0000	0000	0000	0001	encodage binaire

Qui revient à une représentation binaire qu'on peut noter plus simplement comme :

$$\boxed{10...01} \textit{ (il y a 30 zéros)}$$

Qu'on inteprète cette représentation selon un format par valeur signée, en complément à un, en complément à deux ou selon le standard IEEE 754, on doit interpréter le bit le plus à gauche comme un bit de signe. Dans tous ces cas, on a donc un nombre négatif dont le reste de la représentation consiste en une suite de 30 zéros et un unique 1.

En valeur signée, on encode la valeur absolue d'un nombre négatif comme une valeur positive. On lit donc immédiatement la valeur 2^0, c'est-à-dire 1, et le nombre encodé est donc -1.

En complément à un, on encode la valeur absolue d'un nombre négatif en complémentant chaque bit de sa représentation binaire. Pour lire un nombre négatif, on doit donc également complémenter chaque bit de sa représentation en complément à un. On a successivement :

10...01	représentation d'un nombre inconnu A en complément à un
01...10	représentation de |A|, la valeur absolue de A

Le premier bit de la représentation absolue ci-dessus correspond au bit de signe, et ne compte donc pas comme faisant partie de la valeur absolue de A. Pour le reste, on a 30 bits à 1 et un unique bit à 0 qui correspond à la puissance 2^0. On peut donc calculer cette valeur absolue

comme la capacité totale d'un format à 31 bits, $2^{31} - 1$, et y soustraire 2^0. La valeur absolue encodée est donc $2^{31} - 2$ et la valeur de A avec son signe est $2 - 2^{31}$. En base 10, ce nombre vaut -2.147.483.646.

Pour obtenir la valeur absolue d'un nombre encodé en complément a deux, on doit opérer comme pour le complément à un et ajouter +1 au résultat. On a donc successivement :

10...01	représentation d'un nombre inconnu A en complément à deux
01...10	représentation de |A-1|
01...11	représentation de |A|, la valeur absolue de A

Contrairement au complément à un, on a ici toutes les puissances de 2 du format à 31 bits représentant la valeur absolue, en omettant le bit de signe ne contribuant pas à cette valeur. On peut donc lire une valeur absolue correspondant à la capacité totale du support à 31 bits de $2^{31} - 1$, et donc une valeur du nombre négatif A de $1 - 2^{31}$ qui est égale en base 10 à -2.147.483.647.

Pour interpréter cette représentation selon les règles du standard IEEE 754 en simple précision, on doit l'analyser selon les champs du signe, de l'exposant et de la mantisse. On procède comme suit :

— Le premier bit est égal à 1, et on a donc un nombre négatif.

— Les 8 bits suivants sont interprétés comme le champ de l'exposant, et sont tous nuls. On a donc un nombre élevé à la puissance $n = 0 - 127 = -127$. Par ailleurs, on est dans le cadre d'une mantisse dénormalisée.

— Les 23 bits restants sont interprétés comme une mantisse dénormalisée, ce qui signifie que ces bits encodent les puissances n de $2^n 0$ tel que $n = \{0, ..., -22\}$. Puisque seul le bit le plus à droite est fixé à 1, on a ici pour unique contribution la valeur 2^{-22}.

Au total, on a assemble donc un nombre égal à $(-1) \cdot 2^{-127} \cdot 2^{-22} = -2^{-149}$. C'est la plus grande quantité négative (plus petite valeur absolue) et non-nulle représentable en simple précision.

(2) Tout d'abord, on sait que tous les nombres strictement positifs ont la même représentation en complément à un et complément à deux, puisqu'il s'agit de leur représentation normale comme en valeur non signée.

Le zéro est un cas particulier. On peut le représenter de la même manière dans les deux formats comme un encodage où tous les bits sont à 0, mais il dispose d'une représentation supplémentaire en complément à un, où il peut également être représenté par un encodage où tous les bits sont à 1, puisque la lecture de la valeur absolue d'un nombre en complément à un consiste à complémenter chaque bit d'une représentation dont le bit de signe est à 1. On peut donc dire qu'il existe une représentation égale dans les deux formats, mais aussi une représentation unique au complément à un.

La lecture des nombres strictement négatifs implique pour le complément à deux la même opération de complémentation des bits que pour le complément à un, mais également d'ajouter +1 à la valeur obtenue après complémentation de la représentation négative. Dès lors, toute représentation d'un nombre strictement négatif dans un format sera lu comme une valeur systématiquement décallée d'une unité dans l'autre format ; il n'existe donc aucun nombre strictement négatif qui a la même représentation dans les deux formats.

On peut également le prouver en employant les expressions sommatoires des valeurs encodées en complément à un et à deux ; on cherche une égalité entre les deux valeurs négatives lues selon les règles du compément à un et à deux, respectivement $v_{[c1]}$ et $v_{[c2]}$:

$$v_{[c1]} = -2^n + 1 + \sum_{i=0}^{n-1} 2^i\, b_i = v_{[c2]} = -2^n + \sum_{i=0}^{n-1} 2^i\, b_i.$$

Cette égalité se simplifie immédiatement en l'équation $1 = 0$, qui n'a aucune solution.

(3) On construit notre valeur en base 2 en suivant les indications du standard IEEE 754, puis on convertira en représentation hexadécimale la représentation complète du nombre en double précision. Il n'y a pas vraiment de sens à construire chaque champ en hexadécimal, puisque ces champs n'ont pas un nombre de bits correspondant systématiquement à des représentations convertissables en hexadécimal sans extension vers un nombre de bit égal à un multiple de 4. On procède comme suit :

— Le bit de signe d'un "NaN", c'est-à-dire une valeur qui n'est pas formellement définie comme un signe, n'est pas spécifié dans le standard IEEE 754. On peut formellement utiliser de manière équivalente l'un ou l'autre. En pratique, plusieurs implémentations pratiques du standard IEEE 754 ajoutent des conventions particulières pour l'interpretation de signes associés à des NaNs.

— Le standard IEEE 754 nous indique que pour représenter un NaN, on doit se placer à un exposant maximum correspondant à fixer à 1 tous les bits du champ associé. Cette modalité est partagée avec la représentation d'un dépassement arithmétique. En double précision, l'exposant est un format de 11 bits. Dès lors, on a la représentation suivante :

11111111111	*(exposant IEEE 754 double précision)*

— Le standard IEEE 754 nous indique que pour représenter un NaN, il est nécessaire et suffisant qu'au moins un des bits de la mantisse soit non-nul. Une mantisse entièrement nulle correspond au cas d'un dépassement arithmétique. En double précision, l'exposant est un format de 52 bits. On a donc **n'importe quelle** représentation **SAUF** la suivante où il y a 52 zéros :

0...0	*(mantisse IEEE 754 simple précision)*

Au total, on a donc $2^{52}-1$ représentations valables du *NaN* en simple précision, voir $2^{53}-1$ si l'on prend en compte l'ambiguité sur le signe qui rend aussi valable une représentation positive que négative, à défaut de plus de précision.

Si l'on note "X" les bits dont la valeur n'a aucune importance, on a donc les représentations suivantes, où le 1 peut se déplacer n'importe où dans la mantisse et où la mantisse contient au total 51 valeurs quelconques notées "X" :

X	11111111111	1X...X

La conversion en hexadécimal n'est pas immédiate, puisque l'unique 1 obligatoire de la mantisse en représentation binaire peut se déplacer à n'importe quel endroit dans celle-ci. Ceci implique que parmi toutes les valeurs hexadécimales incluses dans la mantisse, l'une d'elles sera restreinte aux valeurs = 1,...,F, c'est-à-dire que le zéro lui sera interdite. De plus, l'ambiguité sur le bit de signe fait que la première valeur hexadécimale pourra être 0xF = 0b1111 ou 0x7 = 0b0111. On peut exprimer la représentation hexadécimale avec le tableau de conversion suivant :

X111	1111	1111	1X...X	représentation binaire avec 51 X dans la mantisse
Y	F	F	ZX...X	représentation hexadécimale avec 12 X dans la mantisse

où dans la représentation hexadécimale :
- X désigne une valeur hexadécimale quelconque = {0,...,F}
- Y désigne une valeur = {7,F}
- Z désigne une valeur = {1,...,F} qui peut se déplacer n'importe où dans la mantisse

Il y a bien 12 X dans la représentation hexadécimale de la mantisse, puisqu'à chaque groupe de 4 X en binaire on associe 1 X en hexadécimal, sauf pour le groupe de 4 bits comprenant l'unique 1 obligatoire, et que $\frac{52-4}{4} = 12$