Chapitre 2
Systèmes de numération flottante

Version du 29/08/2024

Table des matières

I. Définitions....2
1. Signes, mantisses et exposants....2
2. Représentation binaire et normalisation de la mantisse....3
II. Norme IEEE pour l'arithmétique en virgule flottante....4
1. Introduction....4
2. Représentation de nombres flottants normalisés....4
3. Représentation du nombre 0....6
4. Représentation de l'infini....6
5. Représentation de NaN....6
6. Représentation de nombres flottants dénormalisés....7
7. La norme en bref....7
III. Conversions....8
1. Convertir une représentation décimale en une représentation IEEE normalisée....8
2. Convertir une représentation IEEE normalisée en une représentation décimale....8
3. Autres conversions....9

I. Définitions

1. Signes, mantisses et exposants

Les nombres peuvent prendre des représentations très variées. La notation scientifique (représentation couramment utilisée) en est une parmi tant d'autres.

Par exemple, le nombre 987,65 peut être représenté de différentes manières uniquement à l'aide de la notation scientifique :

987,65
$0{,}0000098765 \times 10^{8}$
$0{,}0098765 \times 10^{5}$
$9{,}8765 \times 10^{2}$
$98765{,}0 \times 10^{-2}$
$9876500000{,}0 \times 10^{-7}$

Il existe seulement deux différences entre toutes ces formes :

- **La valeur de l'exposant de la puissance de dix.**
- **La position de la virgule** (c'est la raison pour laquelle nous disons que la virgule est flottante).

Le même raisonnement s'applique aux nombres négatifs :

− 987,65
$-\,0{,}0000098765 \times 10^{8}$
$-\,0{,}0098765 \times 10^{5}$
$-\,9{,}8765 \times 10^{2}$
$-\,98765{,}0 \times 10^{-2}$
$-\,9876500000{,}0 \times 10^{-7}$

Nous pouvons aussi exprimer tout nombre binaire dans une notation scientifique. La puissance de dix doit être remplacée par une puissance de deux. Prenez par exemple les nombres binaires suivants (positifs ou négatifs) :

± 110,01
$\pm\,0{,}0000011001 \times 2^{8}$
$\pm\,0{,}0011001 \times 2^{5}$
$\pm\,1{,}1001 \times 2^{2}$
$\pm\,11001{,}0 \times 2^{-2}$
$\pm\,1100100000{,}0 \times 2^{-7}$

Nous pouvons conclure que tout nombre binaire peut être exprimé sous la forme suivante :

$$(-1)^s \times m \times 2^e$$

- *s* représente le signe du nombre. Il peut prendre la valeur 0 ou 1. À l'évidence, quand $s = 0$, le nombre est positif, et quand $s = 1$, le nombre est négatif.
- *m* est appelé *la mantisse* (ou *significande*) et est un nombre entier ou fractionnaire.
- *e* est appelé *l'exposant* et est un nombre entier.

Prenons par exemple le nombre suivant : $-0{,}0011001_2 \times 2^5$

- $s = 1$
- $m = 0{,}0011001_2$
- $e = 5$

2. Représentation binaire et normalisation de la mantisse

Dans sa forme binaire, une mantisse normalisée commence toujours par un *1* suivi d'une virgule. Par exemple, dans le tableau ci-dessous une seule mantisse est normalisée :

110,01	
$0{,}0000011001 \times 2^8$	
$0{,}0011001 \times 2^5$	
$\mathbf{1{,}1001} \times 2^2$	← Cette mantisse est normalisée, car elle commence par « **1,** »
$11001{,}0 \times 2^{-2}$	
$1100100000{,}0 \times 2^{-7}$	

Par conséquent, il n'y a qu'une seule et unique façon de représenter un nombre binaire en notation scientifique à l'aide d'une mantisse normalisée.

Il est important de remarquer que **le nombre 0 ne peut pas être exprimé à l'aide d'une mantisse normalisée**. Ceci est simplement dû au fait que cette dernière est supérieure ou égale à 1 et qu'une puissance de deux est supérieure à 0 (leur produit n'est donc jamais nul).

II. Norme IEEE pour l'arithmétique en virgule flottante

1. Introduction

L'institut des ingénieurs électriciens et électroniciens (*the Institute of Electrical and Electronics Engineers* ou IEEE) a défini une norme pour la représentation des nombres à virgule flottante. Il s'agit de la **norme IEEE 754**. Elle fut créée en 1985. À cette époque, une multitude de formats différents étaient utilisés pour encoder les nombres à virgule flottante, mais aujourd'hui, la norme IEEE 754 prédomine dans toute l'industrie informatique. On la retrouve dans les architectures logicielles et matérielles.

La norme IEEE définit trois champs binaires et une valeur spéciale appelée *biais*.

S	**E**	**M**

- *S* est le champ qui contient le codage du **signe**.
- *E* est le champ qui contient le codage de l'**exposant**.
- *M* est le champ qui contient le codage de la **mantisse**.

Le **biais** est une valeur spéciale qui sert à encoder l'exposant.

La taille des champs et la valeur du biais dépendent de la précision définie par la norme. En fait, la norme définit plusieurs niveaux de précision. Le tableau ci-dessous donne quelques exemples de la taille des champs et de la valeur du biais en fonction du niveau de précision.

	S (bits)	**E (bits)**	**M (bits)**	**Total (bits)**	**biais**
Demi-précision	1	5	10	16	15
Simple précision	1	8	23	32	127
Double précision	1	11	52	64	1 023
Quadruple précision	1	15	112	128	16 383

Les deux niveaux de précision les plus couramment utilisés sont la **simple** et la **double précision**.

2. Représentation de nombres flottants normalisés

Pour une raison qui sera expliquée ultérieurement, les valeurs minimale (tous les bits à 0) et maximale (tous les bits à 1) de *E* sont réservées pour le codage de valeurs spéciales. Par conséquent, les raisonnements décrits dans cette partie ne s'appliquent pas à ces deux valeurs extrêmes de *E*.

S	**E**	**M**
Φ	≠ 00..0 ≠ 11..1	ΦΦ.....Φ

Φ peut prendre les valeurs 0 ou 1

Nous avons vu que tout nombre binaire pouvait s'exprimer sous la forme suivante :

$$(-1)^s \times m \times 2^e$$

La norme IEEE utilise cette forme pour encoder le signe (*s*), l'exposant (*e*) et la mantisse (*m*). Les valeurs codées sont alors stockées respectivement dans les champs *S*, *E* et *M*.

Les relations entre les valeurs *s*, *e*, et *m* et leurs champs respectifs *S*, *E*, et *M* sont :

- $s = S$
- $e = E$ – biais
- $m = (1,M)_2$

À partir de ces expressions, un nombre normalisé peut s'exprimer sous la forme suivante :

$$(-1)^S \times (1,M)_2 \times 2^{E - biais}$$

- **Le champ *S* contient le codage du signe *s***. C'est-à-dire, si $S = 0$, le nombre est positif. Et si $S = 1$, le nombre est négatif.

- **Le champ *E* contient le codage de l'exposant *e***. Ce champ doit pouvoir représenter les valeurs positives et négatives de l'exposant. Pour cela, l'exposant est biaisé ; c'est-à-dire qu'un biais lui est ajouté. L'exposant biaisé est alors stocké dans le champ ($E = e + biais$). Par exemple, en simple précision, la valeur sur 8 bits de l'exposant biaisé (*E*) est comprise entre 1 et 254 (les valeurs 0 et 255 sont réservées pour des codages spéciaux) ; ce qui veut dire que la valeur de l'exposant (*e*) peut être comprise entre −126 et +127.

- **Le champ *M* contient la partie fractionnaire de la mantisse normalisée *m***. Par conséquent, pour encoder un nombre dans ce format, **la mantisse doit avant tout être normalisée**. Grâce à cette normalisation, le 1 à gauche de la virgule n'a pas besoin d'être stocké dans ce champ.

Comme cela a été dit précédemment, le nombre 0 ne peut pas être encodé à l'aide d'une mantisse normalisée. Par conséquent, cette valeur spéciale nécessite d'être encodée d'une autre façon. En fait, le nombre 0 n'est pas la seule valeur spéciale à encoder ; c'est la raison pour laquelle les valeurs minimale et maximale du champ *E* sont réservées.

3. Représentation du nombre 0

La représentation du nombre 0 se caractérise par des valeurs nulles (que des 0) pour les champs *E* et *M*.

S	E	M
Φ	00.....0	00.....0

Φ peut prendre les valeurs 0 ou 1

Remarquez que comme le champ *S* peut prendre les valeurs 0 ou 1, les valeurs −0 et +0 peuvent être considérées comme deux valeurs distinctes lorsque cela est nécessaire.

4. Représentation de l'infini

Les représentations de +∞ et de −∞ sont caractérisées par une valeur maximale pour *E* (que des 1) et une valeur nulle *M* (que des 0).

S	E	M
Φ	11.....1	00.....0

Φ peut prendre les valeurs 0 ou 1

5. Représentation de NaN

Un *NaN* ou *Not a Number* représente une valeur indéterminée ou invalide (par exemple, le résultat d'une division par 0), mais aussi toute entité qui n'est pas un nombre.

Un *NaN* se caractérise par une valeur maximale pour *E* (que des 1) et une valeur non nulle pour *M*.

S	E	M
Φ	11.....1	≠ 00.....0

Φ peut prendre les valeurs 0 ou 1

6. Représentation de nombres flottants dénormalisés

La norme IEEE permet également le codage de nombres à mantisses dénormalisées. Ces nombres se caractérisent par une valeur nulle pour *E* (que des 0) et une valeur non nulle pour *M*.

S	E	M
Φ	00...0	≠ 00.....0

Φ peut prendre les valeurs 0 ou 1

Les relations entre les valeurs *s*, *e*, et *m* et leurs champs respectifs *S*, *E*, et *M* sont :

- $s = S$
- $e = 1 - biais$
- $m = (0,M)_2$

Notez que l'exposant (*e*) est constant. À partir de ces expressions, un nombre dénormalisé peut s'exprimer sous la forme suivante :

$$(-1)^S \times (0,M)_2 \times 2^{1-biais}$$

En valeur absolue, le plus grand nombre dénormalisé est plus petit que le plus petit nombre normalisé. En d'autres mots, ils comblent l'espace entre la valeur 0 et le plus petit nombre normalisé.

7. La norme en bref

E	M	Correspondance	Commentaire
00.....0	00.....0	± 0	Zéro
00.....0	≠ 00.....0	$(-1)^S \times (0,M)_2 \times 2^{1-biais}$	Nombres dénormalisés
≠ 00....0 ≠ 11....1	ΦΦ.....Φ	$(-1)^S \times (1,M)_2 \times 2^{E-biais}$	Nombres normalisés
11.....1	00.....0	± ∞	Infinis
11.....1	≠ 00.....0	*NaN*	*Not a Number*

Φ peut prendre les valeurs 0 ou 1

III. Conversions

1. Convertir une représentation décimale en une représentation IEEE normalisée

Appliquez les cinq étapes suivantes pour réaliser ce type de conversion :
1. Déterminer la valeur de *S* en fonction du signe du nombre.
2. Convertir la valeur absolue du nombre dans sa représentation binaire (éventuellement fractionnaire).
3. Normaliser la représentation binaire et déterminer facilement les valeurs de *M* et *e*.
4. Déterminer la valeur de *E* en fonction de *e* et du biais : $E = e + biais$.
5. Écrire le résultat final dans sa forme binaire.

Exemple :
Convertir le nombre −145,625 dans sa représentation binaire IEEE simple précision.
1. **S = 1**
2. $|-145{,}625| = 145{,}625 = 10010001{,}101_2$
   0,625 × 2 = **1**,25
   0,25 × 2 = **0**,5
   0,5 × 2 = **1**
3. $145{,}625 = (1{,}0010001101)_2 \times 2^7$
   M = 00100011010…0$_2$ et e = 7
4. E = e + biais = 7 + 127 = 6 + 128
   E = 1000 0110$_2$
5. **−145,625 → 1 10000110 00100011010000000000000**

2. Convertir une représentation IEEE normalisée en une représentation décimale

Appliquez les cinq étapes suivantes pour réaliser ce type de conversion :
1. Déterminer le signe du nombre en fonction de la valeur de *S*.
2. Déterminer *e* en fonction de *E* et du biais : $e = E - biais$.
3. Déterminer *m* dans sa représentation binaire : $m = (1,M)_2$.
4. Écrire le résultat sous la forme suivante : $\pm m \times 2^e$.
5. Simplifier si nécessaire en déplaçant ou supprimant la virgule et convertir en représentation décimale.

Exemple :
Convertir la représentation IEEE double précision suivante dans sa représentation décimale.
$2401\ 8000\ 0000\ 0000_{16} = 0010\ 0100\ 0000\ 0001\ 1000\ 0000\ldots\ldots0_2$
1. S = 0 → **positif**
2. e = E – biais = $010\ 0100\ 0000_2 - 1023 = 576 - 1023$ **= –447**
3. **m** = $(1,M)_2$ = **$(1,00011)_2$**
4. $+m \times 2^e = (1,00011)_2 \times 2^{-447}$
5. $= (100011)_2 \times 2^{-452}$
   = **35×2^{-452}**

3. Autres conversions

Convertir les zéros, les infinies et les NaN est très simple ; il n'y a aucune difficulté particulière.

Pour convertir les nombres dénormalisés, il est possible d'utiliser les mêmes méthodes que pour les nombres normalisés. Les principales différences sont que $e = 1 - biais$ et $m = (0,M)_2$. En fait, convertir un nombre dénormalisé est même plus simple, car leurs exposants sont constants et leurs mantisses n'ont pas besoin d'être normalisées.