IEEE 754



Cet article est une ébauche concernant l'informatique.

Vous pouvez partager vos connaissances en l'améliorant (**comment ?**) selon les recommandations des projets correspondants.

L'**IEEE 754** est un standard pour la représentation des nombres à virgule flottante en binaire. Il est le plus employé actuellement pour le calcul des nombres à virgule flottante dans le domaine informatique, avec les CPU et les FPU. Le standard définit les formats de représentation des nombres à virgule flottante (signe, mantisse, exposant, nombres dénormalisés) et valeurs spéciales (infinis et NaN), en même temps qu'un ensemble d'opérations sur les nombres flottants. Il décrit aussi quatre modes d'arrondi et cinq exceptions (comprenant les conditions dans lesquelles une exception se produit, et ce qui se passe dans ce cas).

Les quatre modes d'arrondi :

- Vers moins l'infini
- Vers plus l'infini
- Vers zéro
- Au plus proche

La version 1985 de la norme IEEE 754 définit 4 formats pour représenter des nombres à virgule flottante :

- simple précision (32 bits : 1 bit de signe, 8 bits d'exposant (-126 à 127), 23 bits de mantisse, avec bit 1 implicite),
- simple précision étendue (≥ 43 bits, obsolète, implémenté en pratique par la double précision),
- double précision (64 bits : 1 bit de signe, 11 bits d'exposant (-1022 à 1023), 52 bits de mantisse, avec bit 1 implicite),
- double précision étendue (≥ 79 bits, souvent implémenté avec 80 bits : 1 bit de signe, 15 bits d'exposant (-16382 à 16383), 64 bits de mantisse, sans bit 1 implicite).

Par exemple, dans le langage C, le compilateur gcc pour les architectures compatible Intel 32 bits utilise le format simple précision pour les variables de type `float`, double précision pour les variables de type `double`, et la double précision ou la double précision étendue (suivant le système d'exploitation) pour les variables de type `long double`. Cependant, si l'extension SSE2 n'est pas utilisée, tous les calculs sont arrondis à la même précision, suivant la configuration de la précision dynamique du processeur (en général, double précision ou double précision étendue, suivant le système d'exploitation, les options de compilation, et les changements effectués par les programmes).

Le titre complet du standard était *IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985)* (standard IEEE pour l'arithmétique binaire en virgule flottante), et est aussi connu sous le nom *IEC 60559:1989, Binary floating-point arithmetic for microprocessor systems*[1], ce qui en fait aussi une norme (américaine), approuvée depuis comme référence normative dans plusieurs normes internationales de l'ISO. Cependant ce standard a été étendu par une révision majeure en 2008 à d'autres types de nombres (soit plus précis, soit plus compacts et moins précis)[2] ; cette révision inclut également des modes d'arrondis supplémentaires et des exigences de conformité plus poussées concernant la précision attendue des opérations et calculs de fonctions transcendantales de base.

Format d'un nombre en virgule flottante

Conventions utilisées dans l'article

Dans un mot de longueur W, les bits sont indexés de 0 à W-1, inclus. Le bit 0 est placé à droite, et il représente le bit de poids faible (c'est-à-dire celui qui va provoquer la plus petite variation s'il est modifié).

Format général

Un nombre flottant est formé de trois éléments : la mantisse, l'exposant et le signe. Le bit de poids fort est le bit de signe. Cela signifie que si ce bit est à 1, le nombre est négatif, et s'il est à 0, le nombre est positif. Les *e* bits suivants représentent l'exposant décalé, et les *m* bits suivants (*m* bits de poids faible) représentent la mantisse.



Signe	Exposant décalé	Mantisse
(1 bit)	(*e* bits)	(*m* bits)

Décalage de l'exposant

L'exposant peut être positif ou négatif. Cependant, la représentation habituelle des nombres signés (complément à 2) rendrait la comparaison entre les nombres flottants un peu plus difficile. Pour régler ce problème, l'exposant est décalé, afin de le stocker sous forme d'un nombre non signé. Ce décalage est de 2^{e-1} - 1 (*e* représente le nombre de bits de l'exposant) ; il s'agit donc d'une valeur constante une fois que le nombre de bits *e* est fixé.

L'interprétation d'un nombre (autre qu'infini) est donc :

$$valeur = signe \times mantisse \times 2^{(exposant - décalage)}$$

avec

- $signe = \pm 1$
- $decalage = 2^{e-1}-1$

Exceptions

Le bit de poids fort de la mantisse est déterminé par la valeur de l'exposant décalé. Si l'exposant décalé est différent de 0 et de $2^e - 1$, le bit de poids fort de la mantisse est 1, et le nombre est dit "normalisé". Si l'exposant décalé est nul, le bit de poids fort de la mantisse est nul, et le nombre est 'dé-normalisé'.

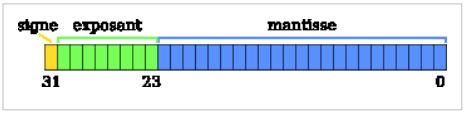
Il y a trois cas particuliers :

- si l'exposant décalé et la mantisse sont tous deux nuls, le nombre est ±0 (selon le bit de signe)
- si l'exposant décalé est égal à $2^e - 1$, et si la mantisse est nulle, le nombre est ±infini (selon le bit de signe)
- si l'exposant décalé est égal à $2^e - 1$, mais que la mantisse n'est pas nulle, le nombre est NaN (not a number : pas un nombre).

Nous pouvons le résumer ainsi :

Type	Exposant décalé	Mantisse
Zéros	0	0
Nombres dénormalisés	0	différente de 0
Nombres normalisés	1 à $2^e - 2$	quelconque
Infinis	$2^e - 1$	0
NaNs	$2^e - 1$	différente de 0

Format simple précision (32 bits)



Un nombre flottant simple précision est stocké dans un mot de 32 bits : 1 bit de signe, 8 bits pour l'exposant et 23 pour la mantisse.

L'exposant est donc décalé de $2^{8-1} - 1 = 127$ dans ce cas. L'exposant d'un nombre normalisé va donc de -126 à +127. L'exposant -127 (qui est décalé vers la valeur 0) est réservé pour zéro et les nombres dénormalisés, tandis que l'exposant 128 (décalé vers 255) est réservé pour coder les infinis et les NaN (voir le tableau précédent).

Un nombre flottant normalisé a une valeur *v* donnée par la formule suivante :

$$v = s \times 2^e \times m.$$

- $s = \pm 1$ représente le signe (selon le bit de signe) ;
- e est l'exposant avant son décalage de 127 ;
- $m = 1+mantisse$ représente la partie significative (en binaire), d'où $1 \leq m < 2$ (*mantisse* étant la partie décimale de la partie significative, comprise entre 0 et 1)

Par exemple pour 0b 0 01111100 01000000000000000000000 : le signe est nul, l'exposant est 124 - 127 = -3, et la partie significative est 0b 1,01 soit 1,25 en décimal ($1\times2^0 + 0\times2^{-1} + 1\times2^{-2}$) ; le nombre représenté est donc $+1{,}25\times2^{-3}$ soit +0,15625.

Les nombres dénormalisés suivent le même principe, sauf que $e = -126$ et $m = 0+mantisse$ (attention : pour le calcul, on veillera à prendre e = -126 et non -127, ceci afin de garantir la continuité de cette représentation avec la représentation normalisée, puisque $m = 0+mantisse$ et non plus $m = 1+mantisse$).

Remarques :

- Il y a deux zéros : +0 et −0 (zéro positif et zéro négatif), selon la valeur du bit de signe.
- Il y a deux infinis : +∞ et −∞, selon la valeur du bit de signe.
- Les zéros et les nombres dénormalisés ont un exposant décalé de -127+127=0 ; tous les bits du champ "exposant" sont donc à 0.
- Les NaNs et les infinis ont un exposant décalé de 128+127=255 ; tous les bits du champ "exposant" sont donc à 1.
- Les NaNs peuvent avoir un signe et une partie significative mais ceux-ci n'ont aucun sens en tant que valeur réelle (sauf pour la signalisation, qui peut activer une exception, et la correction d'erreurs).
- Le nombre dénormalisé non nul le plus proche de zéro est $\pm 2^{-149} \approx \pm 1{,}4012985\times10^{-45}$.
- Le nombre normalisé non nul le plus proche de zéro est $\pm 2^{-126} \approx \pm 1{,}175494351\times10^{-38}$.
- Le nombre normalisé dont la valeur absolue est la plus grande est $\pm (2-2^{-23}) \times2^{127} \approx \pm 3{,}4028235\times10^{38}$.

Voici un tableau résumant la partie précédente, avec des exemples de nombres 32 bits simple précision.

Type	Exposant	Mantisse	Valeur approchée	Écart / préc
Zéro	0000 0000	000 0000 0000 0000 0000 0000	0,0	
Plus petit **nombre dénormalisé**	0000 0000	000 0000 0000 0000 0000 0001	$1{,}4\times10^{-45}$	$1{,}4\times10^{-45}$
Nombre dénormalisé suivant	0000 0000	000 0000 0000 0000 0000 0010	$2{,}8\times10^{-45}$	$1{,}4\times10^{-45}$
Nombre dénormalisé suivant	0000 0000	000 0000 0000 0000 0000 0011	$4{,}2\times10^{-45}$	$1{,}4\times10^{-45}$
Autre nombre dénormalisé	0000 0000	100 0000 0000 0000 0000 0000	$5{,}9\times10^{-39}$	
Plus grand nombre dénormalisé	0000 0000	111 1111 1111 1111 1111 1111	1.17549421×10^{-38}	
Plus petit **nombre normalisé**	0000 0001	000 0000 0000 0000 0000 0000	1.17549435×10^{-38}	$1{,}4\times10^{-45}$
Nombre normalisé suivant	0000 0001	000 0000 0000 0000 0000 0001	1.17549449×10^{-38}	$1{,}4\times10^{-45}$
Presque le double	0000 0001	111 1111 1111 1111 1111 1111	$2{,}35098856\times10^{-38}$	$1{,}4\times10^{-45}$
Nombre normalisé suivant	0000 0010	000 0000 0000 0000 0000 0000	$2{,}35098870\times10^{-38}$	$1{,}4\times10^{-45}$
Nombre normalisé suivant	0000 0010	000 0000 0000 0000 0000 0001	2.35098898×10^{-38}	$2{,}8\times10^{-45}$
Presque 1	0111 1110	111 1111 1111 1111 1111 1111	0,99999994	$0{,}6\times10^{-7}$
1	0111 1111	000 0000 0000 0000 0000 0000	1,00000000	
Nombre suivant 1	0111 1111	000 0000 0000 0000 0000 0001	1,00000012	$1{,}2\times10^{-7}$
Presque le plus grand nombre	1111 1110	111 1111 1111 1111 1111 1110	$3{,}40282326\times10^{38}$	
Plus grand nombre normalisé	1111 1110	111 1111 1111 1111 1111 1111	$3{,}40282346\times10^{38}$	2×10^{31}
Infini	1111 1111	000 0000 0000 0000 0000 0000	Infini	
Première valeur (dénormalisée) de ***NaN*** avertisseur	1111 1111	000 0000 0000 0000 0000 0001	*NaN*	
NaN normalisé (avertisseur)	1111 1111	010 0000 0000 0000 0000 0000	*NaN*	
Dernière valeur (dénormalisée) de ***NaN*** avertisseur	1111 1111	011 1111 1111 1111 1111 1111	*NaN*	
Première valeur (dénormalisée) de ***NaN*** silencieux	1111 1111	100 0000 0000 0000 0000 0000	*NaN*	
Dernière valeur (dénormalisée) de ***NaN*** silencieux	1111 1111	111 1111 1111 1111 1111 1111	*NaN*	

Notes :

- La valeur de la mantisse de NaN présentée ci-dessus est un exemple de NaN, mais n'est pas la seule valeur possible pour coder une valeur NaN. Les valeurs de mantisse codant pour NaN sont toutes les valeurs possibles, sauf zero (qui code pour l'infini)[3].
- De plus, il est à noter la différence, sur la plupart des machines, entre une mantisse à 1xx xxxx xxxx xxxx xxxx xxxx, utilisée pour les NaN silencieux, et une mantisse à 0xx xxxx xxxx xxxx xxxx xxxx (où les x ne sont pas tous nuls), indiquant un NaN avec avertissement[4]. Sur certaines machines cependant, comme celles à base de PA-RISC, le choix inverse (moins pratique) a été fait[5].
- Lors des certaines opérations dont une valeur NaN est utilisée en argument, et dans lesquelles le bit d'avertissement ne déclenche pas d'exception ou dont l'exception déclenchée a traité l'avertissement en choisissant de l'ignorer, ce bit d'avertissement sera automatiquement inversé pour ne plus déclencher d'exception, et les autres bits de mantisse ne seront pas pris en compte pour différencier les différentes valeurs NaN dans les opérations sur ces valeurs : le résultat de ces opérations, y compris une simple copie d'un registre à un autre pourra donner une valeur NaN unique normalisée (où le bit de signe et le bit d'avertissement seront annulés, un seul bit sera positionné dans la mantisse pour conserver la sémantique NaN, et les autres bits de mantisse seront eux aussi

annulés).

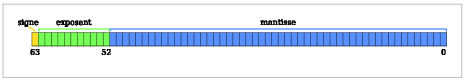
Un exemple plus complexe

Codons le nombre décimal −118,625 en utilisant le mécanisme IEEE 754.

1. Premièrement, nous avons besoin du signe, de l'exposant et de la partie fractionnaire. C'est un nombre négatif, le signe est donc "1".
2. Puis nous écrivons le nombre (sans le signe) en binaire. Nous obtenons 1110110,101 (avec divisions par deux successives pour la partie décimale)
3. Ensuite, nous décalons la virgule vers la gauche, de façon à ne laisser qu'un 1 sur sa gauche : 1110110,101 (bin) = 1,110110101 (bin) × 2^6. C'est un nombre flottant normalisé : la mantisse est la partie à droite de la virgule, remplie de 0 vers la droite pour obtenir 23 bits. Cela donne 110 1101 0100 0000 0000 0000 (on omet le 1 avant la virgule, qui est implicite).
4. L'exposant est égal à 6, et nous devons le convertir en binaire et le décaler. Pour le format 32-bit IEEE 754, le décalage est 2^{8-1}-1 = 127. Donc 6 + 127 = 133 (dec) = 1000 0101 (bin).

On a donc −118,625 (dec) = 1100 0010 1110 1101 0100 0000 0000 0000 (bin) = C2ED4000 (hexa)

Format double précision (64 bits)



Le format double précision est identique au simple précision, mis à part le fait que les champs sont plus grands. En effet, il possède 52 bits de mantisse au lieu de seulement 23, et 11 bits d'exposant au lieu de seulement 8.

La mantisse est très élargie, alors que l'exposant est peu élargi. Ceci est dû au fait que, selon les créateurs du standard, la précision est plus importante que l'amplitude.

Les NaN et les infinis sont représentés en mettant tous les bits de l'exposant à 1 (2047), mais distingués en mettant tous les 52 bits de la mantisse à 0 pour les infinis et au moins un de ces 52 bits à 1 pour les NaN.

Pour les nombres normalisés, le décalage de l'exposant est +1023. Pour les nombres dénormalisés, l'exposant est −1022 (l'exposant minimum pour un nombre normalisé). Ce n'est pas −1023 car les nombres normalisés ont un 1 avant la virgule, et les nombres dénormalisés n'en ont pas. Comme précédemment, zéro et l'infini sont signés.

Remarques :

- Le plus petit nombre positif différent de zéro, et le plus grand nombre négatif différent de zéro (représentés par une valeur dénormalisée avec tous les bits du champ Exposant à 0 et la valeur binaire 1 dans le champ Fraction) sont :

  $\pm 2^{-1074} \approx \pm 4{,}9406564584124654 \times 10^{-324}$
- Le plus petit nombre positif normalisé différent de zéro, et le plus grand nombre négatif normalisé différent de zéro (représentés par la valeur binaire 1 dans le champ Exp, et 0 dans le champ Fraction sont :

  $\pm 2^{-1022} \approx \pm 2{,}2250738585072020 \times 10^{-308}$
- Le plus grand nombre positif fini, et le plus petit nombre négatif fini (représenté par la valeur 2046 dans le champ Exp et tous les bits à 1 dans le champ Fraction) sont :

  $\pm(2^{1024} - 2^{971}) \approx \pm 1{,}7976931348623157 \times 10^{308}$

Comparer des nombres flottants

Il est généralement préférable de comparer des nombres flottants en utilisant les instructions de calcul flottant. Cependant, cette représentation rend les comparaisons de certains sous-ensembles possible octet par octet, s'ils ont le même ordre d'octets et le même signe, et que les NaNs sont exclus.

Par exemple, pour deux nombres flottants positifs a et b, la comparaison entre a et b (>, <, ou ==) donne les mêmes résultats que la comparaison de deux nombres signés (ou non signés) avec les mêmes bits que a et b. En d'autres mots, deux nombres flottants positifs (qui ne sont pas des NaN) peuvent être comparés avec une comparaison binaire signée (ou non signée). À cause du problème d'ordre des octets, cette comparaison ne peut pas être utilisée dans du code portable.

Arrondir les nombres flottants

Le standard IEEE spécifie quatre modes d'arrondi.

- **Au plus près** : si le nombre est entre deux, il est arrondi à la valeur la plus proche avec un bit de poids faible à 0 (mode d'arrondi par défaut) ;
- **Vers zéro** ;
- **Vers plus l'infini** ;
- **Vers moins l'infini**.

Révision du standard

En juin 2008, une révision majeure des normes IEEE 754 et IEEE 854 a été approuvée par l'IEEE. Voir : IEEE 754-2008 (en)[6],[7].

Cette révision apporte de nouveaux formats en base 2 et en base 10, et spécifie la représentation des formats en base 10 (en plus de la base 2).

Elle normalise également une relation d'ordre total pour chacun des types de données numériques normalisés, en complétant les relations d'ordre habituelles qui ne sont que partielles ; en effet la relation d'ordre normale n'est totale qu'à condition de supprimer de l'ensemble de valeurs, la valeur zéro négative (normalement comparée comme égale à la valeur zéro positive) et toutes les valeurs NaN (qui ne sont ni égales, ni supérieures, ni inférieures à aucune autre, pas même elles-mêmes).

En revanche, cette révision laisse la flexibilité de représentation et de distinction éventuelle des valeurs NaN (la position et la valeur du ou des bits d'avertissement dans le champ mantisse ne sont pas normalisées, et l'utilisation des autres bits du champ mantisse ou de signe d'une valeur NaN pour codifier une erreur reste dépendante de l'architecture ou des applications).

Bibliographie

- IEEE, Standard IEEE-754
- David Goldberg, *What Every Computer Scientist Should Know about Floating-Point Arithmetic*

Notes et références

[1] (http://www.opengroup.org/onlinepubs/009695399/frontmatter/refdocs.html)

[2] *IEEE Standard for Floating-Point Arithmetic (ANSI/IEEE Std 754-2008)* (http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4610933), ISBN 978-0-7381-5753-5

[3] http://en.wikipedia.org/wiki/IEEE_754-1985#NaN

[4] http://en.wikipedia.org/wiki/NaN#Floating_point

[5] http://grouper.ieee.org/groups/1788/email/msg03272.html

[6] Revising ANSI/IEEE Std 754-1985 ; 754R Draft Approved as IEEE Standard (http://www.validlab.com/754R/), sur www.validlab.com

[7] Revising ANSI/IEEE Std 754-1985 ; 754R Draft Approved as IEEE Standard (http://754r.ucbtest.org/), sur 754r.ucbtest.org

Liens externes

- Deterministic cross-platform floating point arithmetics (http://www.christian-seiler.de/projekte/fpmath/) : Nombreuses informations sur les différentes implémentations d'IEEE 754 sur diverses plateformes
- (en) Convertisseur binaire (http://www.binaryconvert.com/convert_float.html) : Convertisseur binaire interactif à précisions simple et double selon la norme IEEE 754
- *Voir aussi la liste d'articles (en anglais) sur la page du groupe chargé de la révision du standard* (http://grouper.ieee.org/groups/754/).

Sources et contributeurs de l'article

IEEE 754 *Source*: http://fr.wikipedia.org/w/index.php?oldid=96325132 *Contributeurs*: !Silent, (:Julien:), 16@r, Anierin, Bech, Camico, Cdang, Comg, Creak, David.Monniaux, DocteurCosmos, DuGNu, Epommate, Erasmus.new, Fluti, Francois Trazzi, GMjeanmatt, GabHor, Gnurou, Haypo, Hiredhel, La Corona, Litlok, MickaëlG, Mmenal, Mro, Neitsa, Ninety, Oiffrig, Olchat, Ork, Oxo, Phe, Président, Psaxl, Quarante-quatre, Soaocohoa, Speculos, Ubuntero89, Verdy p, Vincent Lefèvre, Zil, 74 modifications anonymes

Source des images, licences et contributeurs

Image: Crystal mycomputer.png *Source*: http://fr.wikipedia.org/w/index.php?title=Fichier:Crystal_mycomputer.png *Licence*: inconnu *Contributeurs*: Dake, Rocket000

Fichier:IEEE754_Format_General.png *Source*: http://fr.wikipedia.org/w/index.php?title=Fichier:IEEE754_Format_General.png *Licence*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributeurs*: GMjeanmatt

Fichier:IEEE754_simple_precision.png *Source*: http://fr.wikipedia.org/w/index.php?title=Fichier:IEEE754_simple_precision.png *Licence*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributeurs*: GMjeanmatt

Fichier:IEEE754 double precision.png *Source*: http://fr.wikipedia.org/w/index.php?title=Fichier:IEEE754_double_precision.png *Licence*: Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributeurs*: GMjeanmatt

Licence

Creative Commons Attribution-Share Alike 3.0 Unported
//creativecommons.org/licenses/by-sa/3.0/