Représentation de données

Représentation approximative des nombres réels : notion de nombre flottant

M. Tellene

Représentation de données

Avant de commencer, il faut distinguer deux choses : les nombres décimaux et les nombres flottants

Nombres décimaux : pour les quantités fixes comme l'argent, où on veut un nombre spécifique de décimales

Nombres flottants : destinés à stocker des nombres de précision en virgule flottante

Représentation de données

Il n'est pas bien compliqué de représenter un nombre décimal en binaire

Exemple de $(123,25)_{10}$

La décomposition de ce nombre est :

$1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}$

Représentation de données

Il n'est pas bien compliqué de représenter un nombre décimal en binaire

Exemple de $(123,25)_{10}$

La décomposition de ce nombre est :
$1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}$

Exemple de $(101,11)_2$

Représentation de données

Il n'est pas bien compliqué de représenter un nombre décimal en binaire

Exemple de $(123,25)_{10}$

La décomposition de ce nombre est :

$1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}$

Exemple de $(101,11)_2$

La décomposition de ce nombre est :

$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$

Ici, je peux calculer la représentation décimale de $(101,11)_2$

Représentation de données

Il n'est pas bien compliqué de représenter un nombre décimal en binaire

Exemple de $(123,25)_{10}$

La décomposition de ce nombre est :

$1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}$

Exemple de $(101,11)_2$

La décomposition de ce nombre est :

$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$

Ici, je peux calculer la représentation décimale de $(101,11)_2$

$(101,11)_2 = (5,75)_{10}$

Représentation de données

Puissance négative de 2 :

2^{-1}	2^{-2}	2^{-3}	2^{-4}
0.5	0.25	0.125	0.0625

2^{-5}	2^{-6}	2^{-7}	2^{-8}
0.03125	0.015625	0.0078125	0.00390625

Représentation de données

Mais comment faire l'inverse ?

Reprenons $(123,25)_{10}$

Représentation de données

Mais comment faire l'inverse ?

Reprenons $(123,25)_{10}$

On commence par dissocier la partie entière de la partie décimale, ainsi nous devons représenter :

- $(123)_{10}$
- $(0,25)_{10}$

Représentation de données

Mais comment faire l'inverse ?

Reprenons $(123,25)_{10}$

On commence par dissocier la partie entière de la partie décimale, ainsi nous devons représenter :

- $(123)_{10}$
- $(0,25)_{10}$

Nous savons représenter $(123)_{10} \rightarrow (01111011)_2$

Représentation de données

Représentation de $(0,25)_{10}$: méthode des multiplications successives

$$\begin{array}{rrrr} & 0, & 2 & 5 \\ \times & & & 2 \\ \hline \color{red}{0+} & & 0, & 5 \end{array} \qquad\qquad \begin{array}{rrr} & 0, & 5 \\ \times & & 2 \\ \hline \color{red}{1+} & & 0 \end{array}$$

Représentation de données

Représentation de $(0,25)_{10}$: méthode des multiplications successives

$$\begin{array}{rrrr} & 0, & 2 & 5 \\ \times & & & 2 \\ \hline \color{red}{0+} & & 0, & 5 \end{array} \qquad \begin{array}{rrr} & 0, & 5 \\ \times & & 2 \\ \hline \color{red}{1+} & & 0 \end{array}$$

$(0,25)_{10} = (0,01)_2$

Représentation de données

Représentation de $(0,25)_{10}$: méthode des multiplications successives

$$\begin{array}{cccc} & 0, & 2 & 5 \\ \times & & & 2 \\ \hline \color{red}{0+} & & 0, & 5 \end{array} \qquad\qquad \begin{array}{ccc} & 0, & 5 \\ \times & & 2 \\ \hline \color{red}{1+} & & 0 \end{array}$$

$(0,25)_{10} = (0,01)_2$

On a :

- $(123)_{10} = (1111011)_2$
- $(0,25)_{10} = (0,01)_2$

$(123,25)_{10} = (1111011,01)_2$

Représentation de données

Les nombres réels non décimaux ont une écriture décimale avec une infinité de chiffres après la virgule
Par exemple : $1/3$, π, $\sqrt{2}$

Ces nombres réels auront nécessairement aussi une écriture en base deux avec une infinité de décimales.

Représentation de données

Exemple avec $(0,1)_{10}$:

- $0,1 \times 2 = 0,2 = 0 + 0,2$
- $0,2 \times 2 = 0,4 = 0 + 0,4$
- $0,4 \times 2 = 0,8 = 0 + 0,8$
- $0,8 \times 2 = 1,6 = 1 + 0,6$
- $0,6 \times 2 = 1,2 = 1 + 0,2$
- $0,2 \times 2 = 0,4 = 0 + 0,4$
- $0,4 \times 2 = 0,8 = 0 + 0,8$
- $0,8 \times 2 = 1,6 = 1 + 0,6$
- $0,6 \times 2 = 1,2 = 1 + 0,2$

A ce stade, on voit que l'on retombe sur $0,2$: on va donc nécessairement réécrire les mêmes lignes que précédemment. Et on retombera sur $0,6$ puis à nouveau sur $0,2$

Représentation de données

Ainsi $(0,1)_{10} = (0,00011001100110011...)_2$ où la séquence 0011 se répète indéfiniment

Afin de montrer que la séquence 0011 se répète, il est possible d'écrire $(0,1)_{10}$ de la manière suivante :

$(0,1)_{10} = (0,0\underline{0011})_2$

Cette notation indique que la séquence 0011 est répétée indéfiniment

Représentation de données

Ainsi $(0,1)_{10} = (0,00011001100110011...)_2$ où la séquence 0011 se répète indéfiniment

Afin de montrer que la séquence 0011 se répète, il est possible d'écrire $(0,1)_{10}$ de la manière suivante :

$(0,1)_{10} = (0,0\underline{0011})_2$

Cette notation indique que la séquence 0011 est répétée indéfiniment

C'est notamment à cause de ça que l'on a : $0.1 + 0.2 \neq 0.3$

Représentation des données

L’encodage des nombres flottants est inspiré de l’écriture scientifique des nombres décimaux qui se compose :

Représentation des données

L'encodage des nombres flottants est inspiré de l'écriture scientifique des nombres décimaux qui se compose :

- d'un signe (+ ou -)
- d'un nombre décimal m, appelé **mantisse**, compris dans $[1; 10[$
- d'un entier relatif n, appelé **exposant**

$\rightarrow$ 2156 s'écrit $+2,156 \times 10^3$

Représentation de données

D'une manière générale, l'écriture scientifique d'un nombre décimal est de la forme :

$$\pm m \times 10^n$$

avec m la mantisse et n l'exposant

On note en toute rigueur que le nombre 0 ne peut pas être représenté avec cette écriture

Représentation de données

La représentation des nombres flottants et les opérations arithmétiques qui les accompagnent ont été définies dans la norme internationale **IEEE 754**

C'est la norme la plus couramment utilisée dans les ordinateurs

Représentation de données

La représentation des nombres flottants et les opérations arithmétiques qui les accompagnent ont été définies dans la norme internationale **IEEE 754**

C'est la norme la plus couramment utilisée dans les ordinateurs

Il existe 2 types de format de données suivant la précision souhaitée :

- format de données 32 bits appelé *simple precision* ou *binary32*
- format de données 64 bits appelé *double precision* ou *binary64*

Représentation de données

La représentation d'un nombre flottant est similaire à l'écriture scientifique à l'écriture scientifique d'un nombre décimal, à savoir une décomposition en trois parties :

Représentation de données

La représentation d'un nombre flottant est similaire à l'écriture scientifique à l'écriture scientifique d'un nombre décimal, à savoir une décomposition en trois parties :

- un signe s
- une mantisse m
- un exposant n

Représentation de données

Exemple avec le format 32 bits :

- le bit de poids fort est utilisé pour représenter le signe *s* (0 pour le $+$)
- les 8 bits suivants sont réservés pour stocker la valeur de l'exposant *n*
- les 23 derniers bits servent à décrire la mantisse

1	8	23
signe	exposant	mantisse

Représentation de données

L'exposant n est un entier sur 8 bits qui a une valeur entre 0 et 255

Pour le format 32 bits, l'exposant est décalé avec $d = 127$

Ceci permet de représenter des exposants $[-127; 128[$

Les valeurs 0 et 255 étant réservées pour représenter des particuliers[1], les exposants sont donc ceux de l'intervalle $[-126; 127[$

1. que l'on verra plus tard

Représentation de données

Mais comment représenter un nombre flottant en binaire ?

1. Indiquer le signe du nombre (mettre un « 0 » ou un « 1 » dans la partie signe)
2. Écrire le nombre (sans le signe) en binaire
3. Décaler la virgule vers la gauche, de façon à ne laisser qu'un 1 sur sa gauche
4. Écrire la partie à droite de la virgule [2], complétée de 0 vers la droite pour obtenir 23 bits
5. Relevé l'exposant calculé à l'étape 3 et le convertir en binaire en tenant compte du biais

2. cf. c'est la mantisse

Représentation de données

En utilisant la norme IEEE 754, quelle est la représentation $(-118,625)_{10}$?

Représentation de données

En utilisant la norme IEEE 754, quelle est la représentation $(-118,625)_{10}$?

1. Le nombre est négatif, on met donc le bit du signe à 1

Représentation de données

En utilisant la norme IEEE 754, quelle est la représentation $(-118,625)_{10}$?

1. Le nombre est négatif, on met donc le bit du signe à 1
2. $(118,625)_{10} = (1110110,101)_2$

Représentation de données

En utilisant la norme IEEE 754, quelle est la représentation $(-118,625)_{10}$?

1. Le nombre est négatif, on met donc le bit du signe à 1
2. $(118,625)_{10} = (1110110,101)_2$
3. $(1110110,101)_2 \rightarrow (1,110110101)_2 \times 2^6$

Représentation de données

En utilisant la norme IEEE 754, quelle est la représentation $(-118,625)_{10}$?

1. Le nombre est négatif, on met donc le bit du signe à 1
2. $(118,625)_{10} = (1110110,101)_2$
3. $(1110110,101)_2 \rightarrow (1,110110101)_2 \times 2^6$
4. $(1,110110101)_2 \times 2^6$, la mantisse est $(110110101)_2$, cela donne donc
   $(110110101)_2 \rightarrow (110\ 1101\ 0100\ 0000\ 0000\ 0000)_2$

Représentation de données

En utilisant la norme IEEE 754, quelle est la représentation $(-118,625)_{10}$?

1. Le nombre est négatif, on met donc le bit du signe à 1
2. $(118,625)_{10} = (1110110,101)_2$
3. $(1110110,101)_2 \rightarrow (1,110110101)_2 \times 2^6$
4. $(1,110110101)_2 \times 2^6$, la mantisse est $(110110101)_2$, cela donne donc
   $(110110101)_2 \rightarrow (110\,1101\,0100\,0000\,0000\,0000)_2$
5. L'exposant calculé était 6, on a donc $(6)_{10} \rightarrow (110)_2$. Pour le format 32 bits, on rappelle que le biais est de $d = 127$, donc $(6)_{10} + (127)_{10} = (10000101)_2$

Représentation de données

En utilisant la norme IEEE 754, quelle est la représentation $(-118,625)_{10}$?

1. Le nombre est négatif, on met donc le bit du signe à 1
2. $(118,625)_{10} = (1110110,101)_2$
3. $(1110110,101)_2 \rightarrow (1,110110101)_2 \times 2^6$
4. $(1,110110101)_2 \times 2^6$, la mantisse est $(110110101)_2$, cela donne donc
   $(110110101)_2 \rightarrow (110\,1101\,0100\,0000\,0000\,0000)_2$
5. L'exposant calculé était 6, on a donc $(6)_{10} \rightarrow (110)_2$. Pour le format 32 bits, on rappelle que le biais est de $d = 127$, donc $(6)_{10} + (127)_{10} = (10000101)_2$

On a donc
$(-118,625)_{10} = (1100\,0010\,1110\,1101\,0100\,0000\,0000\,0000)_2$

Représentation de données

D'une manière générale, un nombre flottant a la forme suivante :

$$(-1)^s m \times 2^{(n-d)}$$

Représentation de données

D'une manière générale, un nombre flottant a la forme suivante :

$$(-1)^s m \times 2^{(n-d)}$$

Les différences entre la norme IEEE 754 et l'écriture scientifique sont :

Représentation de données

D'une manière générale, un nombre flottant a la forme suivante :

$$(-1)^s m \times 2^{(n-d)}$$

Les différences entre la norme IEEE 754 et l'écriture scientifique sont :

- la base choisie est la base 2
- la mantisse est dans $[1, 2[$
- l'exposant n est décalé d'une valeur d qui dépend du format choisi

Représentation de données

Afin de représenter des exposants positifs et négatifs, la norme IEEE 754 **n'utilise pas** l'encodage par complément à 2 des entiers relatifs

Une autre technique est utilisée : stocker l'exposant de manière décalée sous la forme d'un nombre non signé [3]

3. un nombre entier naturel

Représentation de données

La mantisse m étant toujours comprises dans l'intervalle $[1; 2[$, elle représente un nombre de la forme $1, x...x$, c'est à dire un nombre commençant **nécessairement** par 1

Représentation de données

La mantisse m étant toujours comprises dans l'intervalle $[1; 2[$, elle représente un nombre de la forme $1, x...x$, c'est à dire un nombre commençant **nécessairement** par 1

Ainsi, pour gagner 1 bit de précision, les 23 bits dédiées à la mantisse sont uniquement utilisés pour représenter les chiffres après la virgule, qu'on appelle **fraction**

Représentation de données

La mantisse m étant toujours comprises dans l'intervalle $[1; 2[$, elle représente un nombre de la forme $1, x ... x$, c'est à dire un nombre commençant **nécessairement** par 1

Ainsi, pour gagner 1 bit de précision, les 23 bits dédiées à la mantisse sont uniquement utilisés pour représenter les chiffres après la virgule, qu'on appelle **fraction**

Ainsi, si les 23 bits dédiées à la mantisse sont $b_1 b_2 ... b_{23}$ alors la mantisse représente le nombre :

$$1 + b_1 \times 2^{-1} + b_2 \times 2^{-2} + ... + b_{23} \times 2^{-23}$$

Représentation de données

Par exemple, le mot de 32 bits suivant :

$$(11000011010101101100000000000000)_2$$

Représentation de données

Par exemple, le mot de 32 bits suivant :

$$(11000011010101101100000000000000)_2$$

$$\underbrace{1}_{\textit{signe}}\ \underbrace{10000110}_{\textit{exposant}}\ \underbrace{10101101100000000000000}_{\textit{fraction}}$$

Représentation de données

Par exemple, le mot de 32 bits suivant :

$$(11000011010101101100000000000000)_2$$

$$\underbrace{1}_{\textit{signe}}\ \underbrace{10000110}_{\textit{exposant}}\ \underbrace{10101101100000000000000}_{\textit{fraction}}$$

$$\begin{aligned}\text{signe} &= (-1)^s \\ &= (-1)^1 \\ &= -1\end{aligned}$$

Représentation de données

Par exemple, le mot de 32 bits suivant :

$$(11000011010101101100000000000000)_2$$

$$\underbrace{1}_{\textit{signe}}\ \underbrace{10000110}_{\textit{exposant}}\ \underbrace{10101101100000000000000}_{\textit{fraction}}$$

$$\begin{aligned} \text{signe} &= (-1)^s \\ &= (-1)^1 \\ &= -1 \end{aligned}$$

$$\begin{aligned} \text{exposant} &= n - d \\ &= (2^7 + 2^2 + 2^1) - 127 \\ &= (128 + 4 + 2) - 127 \\ &= 134 - 127 = 7 \end{aligned}$$

Représentation de données

Par exemple, le mot de 32 bits suivant :

$$(11000011010101101100000000000000)_2$$

$$\underbrace{1}_{signe}\ \underbrace{10000110}_{exposant}\ \underbrace{10101101100000000000000}_{fraction}$$

$$\begin{aligned} \text{signe} &= (-1)^s \\ &= (-1)^1 \\ &= -1 \end{aligned}$$

$$\begin{aligned} \text{exposant} &= n - d \\ &= (2^7 + 2^2 + 2^1) - 127 \\ &= (128 + 4 + 2) - 127 \\ &= 134 - 127 = 7 \end{aligned}$$

$$\begin{aligned} \text{mantisse} &= 1 + b_1 \times 2^{-1} + \ldots + b_{23} \times 2^{-23} \\ &= 1 + 2^{-1} + 2^{-3} + 2^{-5} + 2^{-6} + 2^{-8} + 2^{-9} \\ &= (1,677734375)_{10} \end{aligned}$$

Représentation de données

Par exemple, le mot de 32 bits suivant :

$$(11000011010101101100000000000000)_2$$

$$\underbrace{1}_{signe}\ \underbrace{10000110}_{exposant}\ \underbrace{10101101100000000000000}_{fraction}$$

$$\begin{aligned} \text{signe} &= (-1)^s \\ &= (-1)^1 \\ &= -1 \end{aligned}$$

$$\begin{aligned} \text{exposant} &= n - d \\ &= (2^7 + 2^2 + 2^1) - 127 \\ &= (128 + 4 + 2) - 127 \\ &= 134 - 127 = 7 \end{aligned}$$

$$\begin{aligned} \text{mantisse} &= 1 + b_1 \times 2^{-1} + ... + b_{23} \times 2^{-23} \\ &= 1 + 2^{-1} + 2^{-3} + 2^{-5} + 2^{-6} + 2^{-8} + 2^{-9} \\ &= (1,677734375)_{10} \end{aligned}$$

$$\rightarrow (-1,677734375)_{10} \times 2^7 = (-214,75)_{10}$$

Représentation de données

Il a été vu précédemment qu'il existait 2 formats : 32 et 64 bits

La différence en ces derniers est la valeur d du décalage pour l'exposant et le nombre de bits alloués pour la fraction f de la mantisse m et l'exposant n

	exposant e	fraction f	valeur
32 bits	8 bits	23 bits	$(-1)^s \times 1,f \times 2^{e-127}$
64 bits	11 bits	52 bits	$(-1)^s \times 1,f \times 2^{e-1023}$

Représentation de données

Dans l'état actuel de nos connaissances, le format des nombres flottants ne permet pas de représenter le nombre « 0 »

Représentation de données

Dans l'état actuel de nos connaissances, le format des nombres flottants ne permet pas de représenter le nombre « 0 »

Rappel : Puisque un nombre flottant sur 32 bits correspond à la formule $(-1)^s \times 1,f \times 2^{e-127}$, la forme $1,f$ de la mantisse interdit la représentation du 0

Représentation de données

Dans l'état actuel de nos connaissances, le format des nombres flottants ne permet pas de représenter le nombre « 0 »

Rappel : Puisque un nombre flottant sur 32 bits correspond à la formule $(-1)^s \times 1,f \times 2^{e-127}$, la forme $1,f$ de la mantisse interdit la représentation du 0

Pour remédier à ce problème, la norme IEEE 754 utilise les valeurs de l'exposant jusqu'à présent inutilisées, 0 et 255, pour représenter le nombre 0 (mais aussi d'autres valeurs spéciales)

Représentation de données

signe	exposant	fraction	valeur spéciale
0	0	0	$+0$
1	0	0	-0
0	255	0	$+\infty$
1	255	0	$-\infty$
0	255	$\neq 0$	NaN

NaN : Not a Number, représente les résultats d'opérations invalides ($0/0$, $\sqrt{-1}$, $0 \times +\infty$)

Représentation de données

Comme nous l'avons vu, si l'exposant d'un nombre flottant (sur 32 bits) est compris entre 1 et 254, alors la valeur représentée par l'encodage est $(-1)^s \times 1, f \times 2^{e-127}$

Les nombres représentés ainsi sont les nombres flottants normalisés

Représentation de données

Comme nous l'avons vu, si l'exposant d'un nombre flottant (sur 32 bits) est compris entre 1 et 254, alors la valeur représentée par l'encodage est $(-1)^s \times 1, f \times 2^{e-127}$

Les nombres représentés ainsi sont les nombres flottants normalisés

Il existe également des nombres appelés nombres dénormalisés de la forme :

Représentation de données

Comme nous l'avons vu, si l'exposant d'un nombre flottant (sur 32 bits) est compris entre 1 et 254, alors la valeur représentée par l'encodage est $(-1)^s \times 1, f \times 2^{e-127}$

Les nombres représentés ainsi sont les nombres flottants normalisés

Il existe également des nombres appelés nombres dénormalisés de la forme :

$$(-1)^s \times 0, f \times 2^{-126}$$

Pour avoir ces nombres, il faut :

- un exposant à 0
- une mantisse différente de 0