Introduction à l'architecture des ordinateurs

Cours 3

Le codage à virgule flottante

Version du 11 septembre 2023

Plan du cours

Rappels sur les nombres décimaux

Représentations en machine des nombres fractionnaires

Le format IEEE 754

Réel, rationnels, décimaux, flottants

Si les machines manipulent assez facilement les entiers (naturels ou relatifs), il en est autrement des autres ensembles de nombre que l'on utilise couramment en mathématiques comme $\mathbb{R}$, $\mathbb{Q}$, ou encore $\mathbb{D}$.

Pour mémoire :

- $\mathbb{R}$ est l'ensemble des réels.
- $\mathbb{Q} = \{q = \frac{n}{d} \mid n \in \mathbb{Z} \land d \in \mathbb{N}^+\}$ est l'ensemble des rationnels.
- $\mathbb{D} = \{d = \frac{n}{10^k} \mid n \in \mathbb{Z} \land k \in \mathbb{N}\}$ est l'ensemble des décimaux.
- $\mathbb{D} \subsetneq \mathbb{Q} \subsetneq \mathbb{R}$.

Notation décimale positionnelle

$$\mathbb{D} = \left\{ d = \frac{n}{10^k} \mid n \in \mathbb{Z} \wedge k \in \mathbb{N} \right\}$$

Pour écrire les nombres décimaux, on peut étendre la notation positionnelle pour inclure la partie fractionnaire : on parle alors de notation décimale positionnelle.

$$\begin{aligned} s_n \dots s_0, s_{-1} \dots s_{-m} &= s_n \times 10^n + \dots + s_0 \times 10^0 \\ &\quad + s_{-1} \times 10^{-1} + \dots + s_{-m} \times 10^{-m} \\ &= \sum_{i=-m}^{i=n} s_i \times 10^i \end{aligned}$$

avec $\forall i, s_i \in [0..10 - 1]$.

Exemples

$\frac{123}{10^2} = 1,23$ $\quad$ $\frac{123}{10^4} = 0,0123$ $\quad$ $\frac{314159}{10^5} = 3,14159$

Nombres « décimaux binaires »

Sur le modèle de $\mathbb{D}$, on peut construire l'ensemble suivant :

$$\left\{ b = \frac{n}{2^k} \mid n \in \mathbb{Z} \wedge k \in \mathbb{N} \right\}$$

Et pour représenter les nombres de cet ensemble, on peut étendre la notation positionnelle en base 2 :

$$\begin{aligned} b_n \dots b_0, b_{-1} \dots b_{-m} &= b_n \times 2^n + \dots + b_0 \times 2^0 \\ &\quad + b_{-1} \times 2^{-1} + \dots + b_{-m} \times 2^{-m} \\ &= \sum_{i=-m}^{i=n} b_i \times 2^i \end{aligned}$$

avec $\forall i, b_i \in [0..2-1]$.

Notation des nombres « décimaux binaires »

Exemples

$\frac{111}{10^{11}} = 0,111$

soit en base 10 : $2^{-1} + 2^{-2} + 2^{-3} = 0,875$

$\frac{1010101}{10^{11}} = 1010,101$

soit en base 10 : $2^3 + 2^1 + 2^{-1} + 2^{-3} = 10,625$

$\frac{11001001001}{10^{1001}} = 11,001001001$

soit en base 10 : $2^1 + 2^0 + 2^{-3} + 2^{-6} + 2^{-9} = 3,142578125_{10}$

Plan du cours

Rappels sur les nombres décimaux

Représentations en machine des nombres fractionnaires

Le format IEEE 754

Comment représenter les nombres fractionnaires en machine ?

Pour les entiers, le choix est fait de se limiter aux nombres d'un intervalle dont la taille dépend de la « largeur des circuits » de l'UAL, typiquement 64 bits sur une machine moderne :

- Entiers naturels de $[0, 2^{64} - 1]$
- Entiers relatifs de $[-2^{63}, 2^{63} - 1]$

Pour les besoins plus spécifiques (p.ex. cryptographie), il existe des bibliothèques de « grands entiers » (*BigInt*) qui permettent d'atteindre une précision arbitraire, au prix de performances déterioriées (pourquoi ?)

Peut-on utiliser cette approche pour manipuler des nombres rationnels ?

Comment représenter un nombre décimal

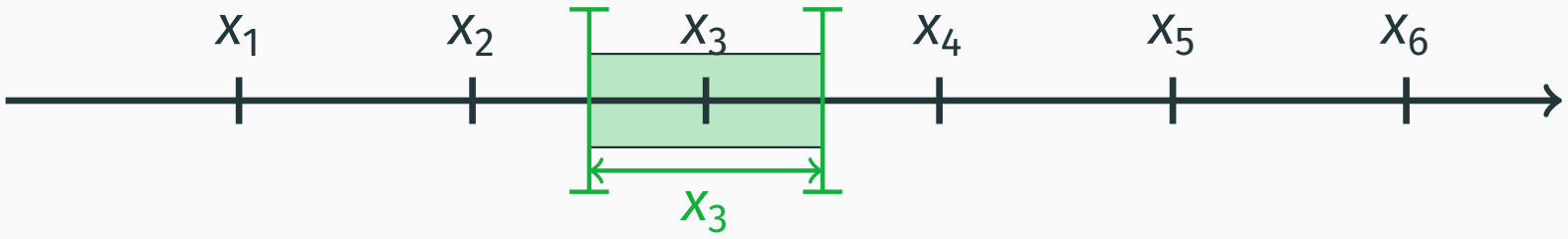
Les rationnels sont un ensemble « dense », et la représentation choisie ne comptera nécessairement qu'un nombre limité d'éléments $\{x_1, x_2, \dots, x_n\}$.

Comment représenter un nombre décimal

Les rationnels sont un ensemble « dense », et la représentation choisie ne comptera nécessairement qu'un nombre limité d'éléments $\{x_1, x_2, \dots, x_n\}$.

Comment représenter un nombre décimal

Les rationnels sont un ensemble « dense », et la représentation choisie ne comptera nécessairement qu'un nombre limité d'éléments $\{x_1, x_2, \dots, x_n\}$.



Chaque élément x_i doit donc être utilisé pour représenter l'ensemble des nombres de l'intervalle $[x_i - \delta_i, x_i + \delta_i]$.

Choisir une représentation pour les décimaux

Choisir une représentation, c'est choisir les x_i et les δ_i :

- Quelle taille (en bits) donner à la partie entière ? à la partie fractionnaire ?
- Comment concilier précision de la représentation et amplitude de la plage des nombres représentables ?
- La représentation garantit-elle l'unicité de la représentation des nombres ?
- La représentation permet-elle une réalisation matérielle efficace (temps, surface de silicium, énergie) des fonctions arithmétiques de base ?

Choisir une représentation pour les décimaux

Choisir une représentation, c'est choisir les x_i et les δ_i :

- Quelle taille (en bits) donner à la partie entière ? à la partie fractionnaire ?
- Comment concilier précision de la représentation et amplitude de la plage des nombres représentables ?
- La représentation garantit-elle l'unicité de la représentation des nombres ?
- La représentation permet-elle une réalisation matérielle efficace (temps, surface de silicium, énergie) des fonctions arithmétiques de base ?

Deux grandes approches sont possibles : la représentation à virgule fixe, et la représentation à virgule flottante.

Représentation à virgule fixe

Le nombre de bit de la partie entière et celui de la partie fractionnaire sont fixés une fois pour toute.

Avantages

- Utilisation des opérateurs d'arithmétique entière :
  - Pas d'unité de calcul dédiée.
  - Performances élevées.
- Assez simple d'estimer l'erreur d'un calcul complexe.
- Permet d'optimiser le format en fonction de la précision et de l'amplitude ciblée pour un cas particulier.

Inconvénients

- Solution spécialisée par cas d'usage → s'oppose à la logique « *general purpose computer* »

Exemple : le format $Q_{m.n}$

Le format $Q_{m.n}$

Codage qui utilise la représentation à virgule fixe : le nombre est codé sous la forme d'un entier en complément à 2 sur $m + n$ bits, qui est ensuite divisé par 2^n.

Exemple : le format $Q_{4.12}$

$\texttt{0xCAFE}_{Q_{4.12}} = \frac{-13\,570}{2^{12}} = -3,312\,988\,281\,25$

Le format $Q_{4.12}$ a les caractéristiques suivantes :

- Mot de 16 bits, dont 12 pour la partie fractionnaire
- Valeur minimale : $\frac{-2^{15}}{2^{12}} = -2^3 = -8$ (codage : `0x8000`)
- Valeur maximale : $\frac{2^{15}-1}{2^{12}} = 2^3 - \frac{1}{2^{12}} = 7,99975585938$ (codage : `0x7FFF`)
- Précision : $\frac{1}{2^{12}} = \frac{1}{4096} = 0,000244140625$

Représentation à virgule flottante

La positione de la virgule n'est pas fixe : reprend le principe de la notation scientifique, avec un nombre fixe de chiffre significatifs et une plage limitée d'exposants.

Avantages

Allie précision importante pour les petits nombres, et capacité à représenter l'ordre de grandeur des grands nombres → s'inscrit dans la logique « *general purpose computer* ».

Inconvénients

Les calculs sont plus complexes qu'en virgule fixe :

- Nécessite l'utilisation d'unité de calcul dédiée → les FPU (pour *Floating Point Unit*)
- Performance moins bonne qu'en virgule fixe.
- Difficile d'évaluer les erreurs de calculs induites par l'approximation.

Rappel : notation scientifique

La notation scientifique du décimal x est définie par le triplet (s, m, e) tel que

$$x = (-1)^s \times m \times 10^e$$

avec $s \in \{0, 1\}$, $m \in [1, 10[$ et $e \in \mathbb{Z}$.

Example

$-12.375 = (-1)^1 \times 1.2375 \times 10^1$

$123\,456\,789 = (-1)^0 \times 1.23456789 \times 10^8$

Notation scientifique en base 2

La notation scientifique est indépendante de la base, et peut donc en particulier être instanciée en base 2 :

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

avec $s \in \{0, 1\}$, $m \in [1, 2[$ et $e \in \mathbb{Z}$.

Example

$-12.375_{10} = -1100,011_2 = (-1)^1 \times 1,100011_2 \times 2^3$

$123\,456\,789 = 111010110111100110100010101_2 =$
$(-1)^0 \times 1,11010110111100110100010101_2 \times 2^{26}$

La suite, ce n'est que de la syntaxe !

Les représentations à virgule flottante codent le triplet (s,m,e) de la notation scientifique en base 2.

Plan du cours

Rappels sur les nombres décimaux

Représentations en machine des nombres fractionnaires

Le format IEEE 754

Les format IEEE 754 simple

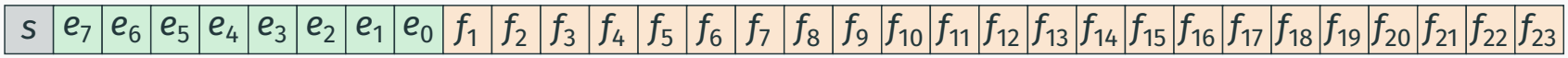
L'IEEE (*Institute of Electrical and Electronics Engineers*) a proposé un format standardisé pour la représentation binaire en virgule flottante, qu'on appelle communément IEEE 754.

Le format définit plusieurs précisions.

Les langages et processeurs modernes utilisent le format *simple precision* sur 32 bits et *double precision* sur 64 bits.

Ces formats correspondent respectivement aux types `float32` et `float64` de golang.

Le format IEEE 754 simple précision



$$x = (-1)^s \times \left(1 + \sum_{i=1}^{23} f_i \times 2^{-i}\right) \times 2^{\left(\sum_{i=0}^{7} e_i \times 2^i\right) - 127}$$

- La partie entière est toujours égale à 1 et n'est pas codée.
- Les 22 premiers bits après la virgule sont codés et le 23ème est un arrondi.
- L'exposant est codé avec un biais de 127 (plage de valeur : $[-126, 127]$ car les valeurs `0x00` et `0xFF` sont réservées)
- La précision (= poids le plus faible) varie avec l'exposant : de 2^{-149} à 2^{104}.

Exemple

On veut coder $1234,5$

1. $1234,5 = 10011010010,1_2$

Exemple

On veut coder $1234,5$

1. $1234,5 = 10011010010,1_2$
2. $10011010010,1_2 = (-1)^0 + 1,00110100101_2 \times 2^{10}$

Exemple

On veut coder $1234,5$

1. $1234,5 = 10011010010,1_2$
2. $10011010010,1_2 = (-1)^0 + 1,00110100101_2 \times 2^{10}$
3. $(-1)^0 + 1,00110100101_2 \times 2^{10} = (-1)^0 + 1 + 0,00110100101_2 \times 2^{137-127}$

Exemple

On veut coder $1234,5$

1. $1234,5 = 10011010010,1_2$
2. $10011010010,1_2 = (-1)^0 + 1,00110100101_2 \times 2^{10}$
3. $(-1)^0 + 1,00110100101_2 \times 2^{10} = (-1)^0 + 1 + 0,00110100101_2 \times 2^{137-127}$
4. $s = 0$, $e = 10001001$, $f = 00110100101000000000000$

Exemple

On veut coder $1234,5$

1. $1234,5 = 10011010010,1_2$
2. $10011010010,1_2 = (-1)^0 + 1,00110100101_2 \times 2^{10}$
3. $(-1)^0 + 1,00110100101_2 \times 2^{10} = (-1)^0 + 1 + 0,00110100101_2 \times 2^{137-127}$
4. $s = 0$, $e = 10001001$, $f = 00110100101000000000000$
5. $1234,5 = 01000100100110100101000000000000_{ieee754sp}$

Exemple

On veut coder $1234,5$

1. $1234,5 = 10011010010,1_2$
2. $10011010010,1_2 = (-1)^0 + 1,00110100101_2 \times 2^{10}$
3. $(-1)^0 + 1,00110100101_2 \times 2^{10} = (-1)^0 + 1 + 0,00110100101_2 \times 2^{137-127}$
4. $s = 0$, $e = 10001001$, $f = 00110100101000000000000$
5. $1234,5 = 01000100100110100101000000000000_{ieee754sp}$
6. $1234,5 = 0x449A5000$

Les valeurs dénormalisées

Pour augmenter la densité de valeurs autour de 0, le codage change quand $e = \texttt{0x00}$:

$$x = (-1)^s \times \left(0 + \sum_{i=1}^{23} f_i \times 2^{-i} \right) \times 2^{-126}$$

Ce codage permet de coder les valeurs $+0$ et -0.

Par ailleurs, $e = \texttt{0xFF}$ est réservé pour les valeurs spéciales :

- NaN pour *Not a Number*, quand $f \neq 0$
- $+\infty$, quand $s = 0$ et $f = 0$
- $-\infty$, quand $s = 1$ et $f = 0$

Le format double précision

Le format IEEE754 double précision utilise 64 bits :

- un bit de signe
- 11 bit d'exposant (biais : –1023)
- 52 bit de partie fractionnaire (dans l'écriture scientifique en base 2)

Il fonctionne exactement comme le format simple précision. En particuliers, les valeurs spéciales et les valeurs dénormalisées sont codées avec les mêmes convetions.

Arithmétique IEEE754 : principe

On veut effectuer une opération arithmétique binaire entre deux nombres IEEE754, x_1 et x_2. On suppose que $e_2 < e_1$. Pour effectuer l'opération, il faut :

1. « ré-écrire » x_2 sous la forme $(-1)^{s2} \times 2^{e_1-127} \times (f_2')$ → $f'2 = (1 + f_2) \div 2^{e_1-e_2}$
2. calculer l'opération : $f_s = op(f_1, f_2')$
3. re-normaliser le nombre $(-1)^{s_1 \times s_2} \times 2^{e_1-127} \times f_s$

Lors de l'étape 1, tous les bits significatifs de x_2 peuvent disparaître : on dit que x_2 est absorbé par x_1. De ce fait, l'arithmétique IEEE754 n'a pas toutes les propriétés usuelles !

Arithmétique IEEE754 : propriétés élémentaires

Les propriétés ci-dessous portent sur *FP* l'ensemble des valeurs codables avec le standard IEEE754.

Propriétés conservées :

- 0 élément neutre pour $+$, absorbant pour $\times$.
- 1 élément neutre pour $\times$.
- $\forall x,\ x - x = 0$
- $\times$ et $+$ sont commutatives : $a + b = b + a$ et $a \times b = b \times a$.

Propriétés non conservées

- Dans FP, $+$ et $\times$ ne sont pas associatives :
  $(a + b) + c \neq a + (b + c)$ et $(a \times b) \times c \neq a \times (b \times c)$
- Dans FP, $\times$ ne se distribue pas sur $+$: $a \times (b + c) \neq (a \times b) + (a \times c)$
- $\exists x,\ x \times \frac{1}{x} \neq 1$
- $\exists x, \exists y,\ x \neq 0 \wedge y \neq 0 \wedge x + y = x$
- ...