

LifLF – Architecture matérielle

Sylvain Brandel

2021 – 2022

sylvain.brandel@univ-lyon1.fr

CM 3

CODAGE DES DONNÉES EN MACHINE

PARTIE 2 – CODAGE DES NOMBRES RATIONNELS

Rationnels

- Nombre de la forme $\frac{p}{q}$ avec $p \in \mathbb{Z}$ et $q \in \mathbb{N} - \{0\}$
- Format de longueur fixe là où écriture binaire potentiellement infinie
→ **Approximation**
- Tout $x \in \mathbb{Q}$ positif décomposé en
 - Partie entière $[x] \in \mathbb{N}$ telle que $[x] \leq x < [x] + 1$
 - Partie fractionnaire $\{x\} = x - [x]$ avec $0 \leq \{x\} < 1$

- Notation positionnelle pour l'écriture de $\{x\}$: s'il existe $q \in \mathbb{N}$ tq

$$\{x\} = (0, x_{-1} \dots x_{-q})_{\beta} = \sum_{i=0}^q x_{-i} \beta^{-i}$$

- Alors $(x_{p-1} \dots x_1 x_0, x_{-1} \dots x_{-q})_{\beta}$ écriture de x en base β
- Écriture en base β **pas forcément finie** mais **forcément périodique**
- Ex : $\beta = 10$, écriture de $13/7$ en **notation partie entière - fractionnaire**

$$13/7 = (1, \underline{857142})_{10}$$

Rationnels

Changement de base

- $0 \leq x < 1$ écrit en base 10
- Décimal vers binaire

$$x = (0, x_{-1} \dots x_{-q})_2$$

– Or $2 \times x = (x_{-1}, x_{-2} \dots x_{-q})_2$, donc $x_{-1} = \lfloor 2 \times x \rfloor$

- Multiplications successives par 2
→ extraction bits écriture binaire de x
- Ex : Convertir $1/10 = (0,1)_{10}$ en **écriture binaire**

$$(0,1)_{10} = (0, \underline{00011})_2$$

Rationnels

Changement de base

- $0 \leq x < 1$ écrit en base 2
- Binaire vers décimal
- Multiplications successives par $(10)_{10} = (1010)_2$
→ en calculant en binaire : chiffres décimaux de x
- Fastidieux à la main
- Si pas trop de bits, il suffit d'additionner les poids de ces bits :
 - $2^{-1} = 0,5$ $2^{-2} = 0,25$ $2^{-3} = 0,125$ $2^{-4} = 0,0625$
- Ex : Convertir $(0,1011)_2$ en **écriture décimale**

$$0,5 + 0,125 + 0,0625 = 0,6875$$

Rationnels

Représentation en machine ?

- On n'a pas réellement les rationnels
- Un **nombre flottant normalisé** x est
 - Soit zéro
 - Soit un rationnel de la forme $x = (-1)^s \times \underbrace{(1, b_1 \dots b_{p-1})_2}_{p \text{ bits de précision } (b_0 = 1)} \times 2^e$
 - $s \in \{0,1\}$: signe
 - $(1, b_1 \dots b_{p-1})_2$: mantisse fractionnaire
 - $e \in \mathbb{Z}$: exposant tel que $e_{\min} \leq e \leq e_{\max}$
- On parle de **nombre à virgule flottante**, ou **nombre flottant**, ou **flottant**
- Le **1**, en tête garantit l'unicité de la représentation
- Types `float` et `double` en C


Nombres à virgule flottante

- On ne les a pas tous !
- P. ex $(0,1)_{10}$ pas dans l'exemple suivant ...
- Ex : flottants ≥ 0
avec 3 bits de précision,
 $e_{\min} = -1, e_{\max} = 2$
 $-1 \leq e \leq 2$

e	Binaire	Décimal
-	0	0
e = -1	$(1,00)_2 \times 2^{-1}$ $(1,01)_2 \times 2^{-1}$ $(1,10)_2 \times 2^{-1}$ $(1,11)_2 \times 2^{-1}$	0,5 0,625 0,75 0,875
e = 0	$(1,00)_2 \times 2^0$ $(1,01)_2 \times 2^0$ $(1,10)_2 \times 2^0$ $(1,11)_2 \times 2^0$	1 1,25 1,5 1,75
e = 1	$(1,00)_2 \times 2^1$ $(1,01)_2 \times 2^1$ $(1,10)_2 \times 2^1$ $(1,11)_2 \times 2^1$	2 2,5 3 3,5
e = 2	$(1,00)_2 \times 2^2$ $(1,01)_2 \times 2^2$ $(1,10)_2 \times 2^2$ $(1,11)_2 \times 2^2$	4 5 6 7

Nombres à virgule flottante

- Ex : $(0,1)_{10}$ avec 8 bits de précision

- $(0,1)_{10} = (0,00011)_2$
 $= (0,00011001100110011)_2$
 $= (1,1001100110011)_2 \times 2^{-4}$


- On doit tronquer à 7 bits de mantisse

$$\text{Donc } (1,1001100)_2 \times 2^{-4} \leq (0,1)_{10} \leq (1,1001101)_2 \times 2^{-4}$$

- Il faut faire un choix → **arrondi**
 - Le milieu de cet intervalle est $(1,10011001)_2 \times 2^{-4}$
 - Ici on choisit d'arrondir au plus proche
 - $(0,1)_{10}$ est supérieur au milieu de l'intervalle, donc

L'approximation $(0,1)_{10} \approx (1,1001101)_2 \times 2^{-4}$

Nombres à virgule flottante – norme IEEE-754

- Codage en précision p :

$c =$	1 bit	k bits	p-1 bits
	Signe	Code de l'exposant : c_e	Code de la mantisse : c_m

- Pour un flottant **normal**, la valeur codée est

$$f(c) = (-1)^{s(c)} \times m(c) \times 2^{e(c)}$$

- $s(c)$: signe du flottant
- $m(c) = (1, c_m)_2$ (le **1**, est implicite)
- $e(c) = (c_e)_2 - (2^{k-1} - 1)$

Nombres à virgule flottante – norme IEEE-754

- Codage en précision p :

$c =$	1 bit	k bits	p-1 bits
	Signe	Code de l'exposant : c_e	Code de la mantisse : c_m

- $(c_e)_2 = 0$ et $(c_m)_2 = 0$: nombre représenté est **zéro** (2 codages)
- $(c_e)_2 = 2^k - 1$: valeur **exceptionnelle** (+Inf, -Inf, NaN)
- $1 \leq (c_e)_2 \leq 2^k - 2$: nombre représenté **normal**, alors

$$m(c) = (1, c_m)_2 \quad \text{et} \quad e(c) = (c_e)_2 - \underbrace{(2^{k-1} - 1)}_{\text{biais}}$$

Nombres à virgule flottante – norme IEEE-754

- Simple précision

31	30 ... 23	22 ... 0
Signe	c_e	c_m

- codé sur 32 bits
- type `float` en C)
- Précision $p = 24$ bits, $k = 8$, biais = 127, $e_{\min} = -126$, $e_{\max} = 127$
- $\llbracket w \rrbracket = (-1)^{b_{31}} \times (1, b_{22} \dots b_1 b_0)_2 \times 2^{(\sum_{i=23}^{30} b_i 2^{(i-23)-127})}$
- Valeurs représentables $[\pm 2^{-126}, (2 - 2^{-23}) \times 2^{127}]$

- Double précision

63	62 ... 52	51 ... 0
Signe	c_e	c_m

- codé sur 64 bits
- type `double` en C
- Précision $p = 53$ bits, $k = 11$, biais = 1023, $e_{\min} = -1022$, $e_{\max} = 1023$
- $\llbracket w \rrbracket = (-1)^{b_{63}} \times (1, b_{51} \dots b_1 b_0)_2 \times 2^{(\sum_{i=52}^{62} b_i 2^{(i-52)-1023})}$
- Valeurs représentables $[\pm 2^{-1022}, (2 - 2^{-52}) \times 2^{1023}]$

Nombres à virgule flottante – norme IEEE-754

- Rationnel ou réel : représentation **arrondie** en général
- 4 modes d'arrondis
 - Arrondi au plus proche : RN(c)
 - Si c équidistant de deux flottants consécutifs, on prend celui dont la mantisse termine par 0
 - Arrondi vers $-\infty$ (RD), $+\infty$ (RU), 0 (RZ)
- La norme impose **l'arrondi correct** pour $+$, $-$, \times , \div , $\sqrt{\quad}$:
 - Résultat calculé = **résultat exact arrondi** selon le mode d'arrondi courant
 - Mode d'arrondi par défaut : **arrondi au plus proche** en général

Les float NE SONT PAS les réels