

2.3

Représentation d'un réel en base 2

NSI 1ÈRE - JB DUTHOIT

2.3.1 Virgule fixe

Le codage des nombres à virgules fixe nous permettra ensuite d'étudier le codage avec virgule flottante.

Dans le système décimal, écrire 56.375 signifie :

$$56.375 = 5 \times 10^1 + 6 \times 10^0 + 3 \times 10^{-1} + 7 \times 10^{-2} + 5 \times 10^{-3}$$

La nouveauté est ici la présence des puissances de 10 négatives pour les chiffres après la virgule. Il en est de même en binaire, avec des puissances de 2 :

Exemple

On désire coder en virgule fixe le nombre réel 56.375.

1. Commencer par coder la partie entière : 56

2^6	2^5	2^4	2^3	2^2	2^1	2^0
= 64	= 32	= 16	= 8	= 4	= 2	= 1
0	1	1	1	0	0	0

56 se décompose de façon unique $56_{10} = 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$

Donc $56_{10} = 111000_2$

2. En ce qui concerne la partie fractionnaire : 0.375

Le fonctionnement est identique, mais avec des exposants de 2 négatifs :

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
= 0.5	= 0.25	= 0.125	= 0.0625	= 0.03125	= 0.015625
0	1	1			

0.375 se décompose de façon unique $0.375 = 0 \times 0.5 + 1 \times 0.25 + 1 \times 0.125$

Donc $0.375 = 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$

Donc $0.375_{10} = 0.011_2$

☞ **Attention** : Seconde méthode pour la partie fractionnaire :

- $0.375 \times 2 = 0.75 = 0$ (partie entière) + 0.75(partie fractionnaire)
- 0.75 × 2 = 1.5 = 1(partie entière) + 0.5
- 0.5 × 2 = 1 = 1(partie entière) + 0(partie fractionnaire)

Finalement, on retrouve bien $0.375_{10} = 0.011_2$

3. On peut ensuite conclure :

$$56.375_{10} = 111000.011_2$$



Savoir-Faire 2.10

coder en virgule fixe les nombres suivants :

- 123.6875_{10}
- 14.5_{10}
- 435_{10}
- 171.78515625_{10}



Savoir-Faire 2.11

Décoder maintenant les nombres suivants (virgule fixe) en nombres décimaux :

- 11.101_2
- 100111.101_2
- 1001.111_2

2.3.2 Virgule flottante

Si on reprend le dernier exemple $171,78515625_{10}$ est codé en 10101011.11001001_2 , c'est à dire en la succession des bits suivants :

1	0	1	0	1	0	1	1	1	1	0	0	1	0	0	1
Partie entière								Partie fractionnaire							

Le trait rouge représente la virgule. La position de cette virgule est à entrer dans le préambule. Ici , on a un codage en virgule fixe "8 × 8"

- **Avantages :** L'addition de deux nombres en virgules flottante est très facile et rapide en nombre de calculs : cela se passe comme pour l'addition de deux entiers binaires, en ajoutant juste au "bon endroit" la virgule.

- **Inconvénients :**

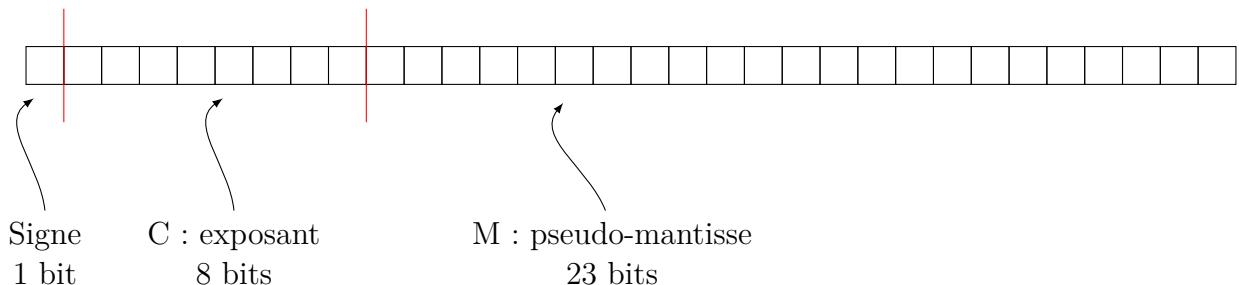
- Le nombre de bits nécessaires pour coder le nombre n'est pas fixe, et dépend du nombre à coder.

- Il faut connaître à l'avance (lors de l'écriture du programme) l'ordre de grandeur des nombres à manipuler afin de positionner au mieux la virgule; une fois la position de la virgule choisie, on ne peut plus changer. Ceci est un inconvénient majeur : ce type de connaissance a priori existe pour certaines applications, mais dans le cas général ce n'est pas le cas.

☞ Pour pallier à ce manque de flexibilité, le concept de virgule flottante est nécessaire !

Il existe différentes façons de coder un nombre en virgule flottante.

L'une d'elles propose ce format (format IEEE754 simple précision) :



Principe :

- Le signe est codé 1 lorsque le nombre est négatif, 0 sinon.
- Il faut ensuite mettre le nombre sous la forme $\pm 1.M \times 2^C$:
 - On commence par coder le nombre souhaité en utilisant la méthode de la virgule fixe
 - On représente ensuite ce nombre de la forme $1.M \times 2^C$
 - * C correspond à l'exposant codé en excédent à 127 (sur 8 bits).
 - * M correspond à la pseudo-mantisse (pseudo, car le '1.' n'est pas codé, c'est toujours 1 et on fait donc l'économie d'un bit) (sur 23 bits)

Exemple

On veut ici coder en virgule flottante simple précision le nombre $171,78515625_{10}$.

- Le premier bit est 0 (signe positif)
- On a vu que $171,78515625_{10} = 10101011.11001001_2$.
Exprimons ce nombre sous la forme $1.M \times 2^C$:
 $10101011.11001001 = 1.01010111001001 \times 2^7$ (décalage de 7 chiffres vers la gauche)
- On a donc $C = 7 + 127$ (excédent) = 134 donc $C = 10000110_2$.
- On trouve ensuite M , en enlevant "1." à 1.01010111001001 : $M = 01010111001001$.
On complète ensuite avec des zéros pour arriver à 23 bits : $M = 01010111001001000000000$
- On a donc : $171,78515625_{10} = 0100001100101011100100100000000_2$

 **Savoir-Faire 2.12**

SAVOIR CODER UN NOMBRE DÉCIMAL EN UTILISANT LA VIRGULE FLOTTANTE SIMPLE PRÉCISION.

Coder en virgule flottante simple précision les nombres suivants :

- 123.6875_{10}
- 14.5_{10}
- 435_{10}
- $171,78515625_{10}$
- 0.25
- 0.1 (plus difficile)
- $\frac{1}{3}$ (plus difficile)

 **Savoir-Faire 2.13**

SAVOIR DÉCODER EN NOMBRE DÉCIMAL UN NOMBRE ÉCRIT EN VIRGULE FLOTTANTE SIMPLE PRÉCISION.

Décoder en virgule flottante simple précision les nombres suivants : Décode en nombre décimal les nombres binaires suivants, exprimés avec une virgule flottante simple précision :

- 01000000001011100001010001111010₂
- 01000011000100100001000000000000₂
- 11000101101100011011001100000000₂
- 001110001111000010111001011110₂