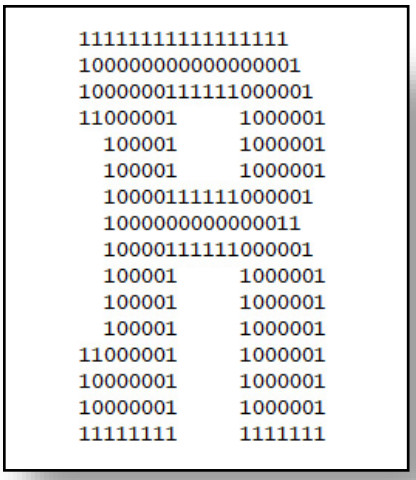


REPRESENTATION DES DONNEES :
Types et valeurs de base (4)

- *Représentation approximative d'un réel*

"En mathématiques, un nombre réel est un nombre qui peut être représenté par une partie entière et une liste finie ou infinie de décimales."

Numérique et Sciences Informatiques
1ère

Support de cours :
Jean-Christophe BONNEFOY

Objectifs :

- Calculer sur des exemples la représentation de nombres réels
- Notion de nombres flottants

1. Qu'est qu'un réel ?

Un nombre réel exprimé dans la base 10 est constitué de deux parties, séparées par une virgule :

- une partie entière : elle est située avant la virgule. Il s'agit de la décomposition d'un nombre en puissance de 10 positives (par exemple : $1 \times 10^1 + 1 \times 10^0 = 11$),
- une partie décimale (ou partie fractionnaire) : elle est située après la virgule. Il s'agit de la décomposition d'un nombre en puissance de 10 négatives (par exemple : $8 \times 10^{-1} + 1 \times 10^{-2} + 1 \times 10^{-3} + 5 \times 10^{-4} = 0{,}8125$).

2. Représentation à virgule fixe d'un réel en base 2

Prenons l'exemple de la conversion du nombre réel $(11{,}8125)_{10}$

Méthode :

On décompose

- En partie réelle (à gauche de la virgule). Il s'agit d'un entier donc on sait déjà faire la conversion en binaire en faisant des divisions successives par 2. Nous avons une décomposition en valeurs positives de puissances de 2
- En partie décimale ou fractionnaire (à droite de la virgule). Nous allons faire une décomposition en valeurs négatives de puissances de 2
- On rassemble les 2 conversion

Dans notre exemple 11,8125, la partie entière vaut 11

$(11)_{10} = (1011)_2$ en effet $11 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$

et la partie décimale vaut 0.8125. Alors pour convertir, on réalise des multiplications successives par 2

0,8125*2 = 1,625

0,625*2 = 1,25

0,25*2 = 0,5

0,5*2 = 1

On remarque qu'ici, la conversion est exacte. (il n'y a plus de virgule). Ce n'est pas toujours le cas !

On a donc $(0{,}8125)_{10} = (1101)_2$ en effet $0{,}8125 = 1 * 2^{-1} + 1 * 2^{-2} + 0 * 2^{-3} + 1 * 2^{-4} = 0{,}5 + 0{,}25 + 0{,}125 + 0{,}0625$

On a donc $(11{,}8125)_{10} = (1011{,}1101)_2$

Le problème est que le caractère « , » n'est pas autorisé dans le langage binaire. Seuls 0 et 1 sont autorisés. Alors l'idée est de prévoir un nombre défini de bits pour la partie entière et un nombre défini de bits pour la partie décimale

Inconvénients :

- L'espace réservé à la partie fractionnaire limite le nombre de bits réservés à la partie entière. Ce qui est gênant pour représenter de très grands nombres, pour lesquels la partie fractionnaire est généralement peu signifiante.
- Inversement, la précision sur de très petits nombres est limitée par le manque d'espace dans la partie fractionnaire alors que pour ces nombres, la partie entière ne contient que des zéros.

Conclusion : L'espace est mal utilisé dans les deux cas.

3. Représentation à virgule flottante

3.1 Rappels sur la notation scientifique en base 10

Le nombre réel $-0{,}6234$ s'écrit en notation scientifique : $-6{,}2334 * 10^{-3}$. De façon générale, tout nombre réel x peut s'écrire avec sa notation scientifique :

$$x = s * m * 10^e$$

Où :

- s représente le **signe** : il ne peut donc prendre que 2 valeurs : -1 ou 1
- m représente la **mantisse**, autrement dit le nombre de chiffres significatifs, c'est à dire un nombre réel compris dans l'intervalle [1,10[
- e représente **l'exposant**, c'est un entier relatif

Exemple :

Nombre réel	s	m	e
$-0{,}00712345$	-1	7,12345	-3
54321,123456	1	5,4321123456	4

On a décalé la virgule, cette écriture est aussi appelée **virgule flottante**. Il nous reste plus qu'à l'adapter en binaire.

3.2 La norme IEEE-754

La norme IEEE 754 est une norme sur larithmétique à virgule flottante. Elle est la norme la plus employée actuellement pour le calcul des nombres à virgule flottante dans le domaine informatique.

Dans cette norme, tout nombre réel x peut s'écrire sous la forme :

$$x = s * m * 2^e$$

Où :

- s représente le **signe** : il ne peut donc prendre que 2 valeurs : +1 (positif) ou -1 (négatif)
  - **En binaire : 1 seul bit : 0 (positif) ou 1 (négatif)**
- m représente la **mantisse**, c'est à dire un nombre réel (positif) compris dans l'intervalle [1,2[. La partie entière de ce nombre sera toujours égale à 1. On posera donc $\boldsymbol{m = 1 + p_f}$ **en notant** $\boldsymbol{p_f}$ **la partie**

fractionnaire de la mantisse. La partie entière 1 ne sera pas codée en binaire mais il ne faudra pas l'oublier dans les conversions. On parle de bit implicite.

- **En binaire : on ne représente que la partie fractionnaire p_f (sur 23, 52 ou 112 bits) selon le format**

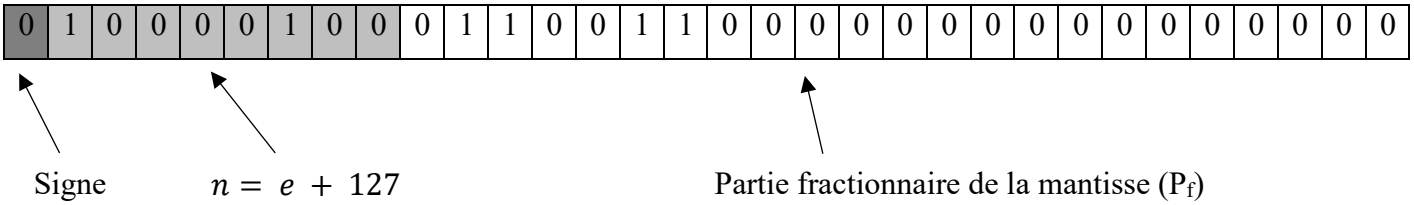
- e représente **l'exposant**, toujours un entier relatif.
  - **En binaire : on représente le code n tel que $e = n - décalage$.** En effet, avec k bits, on souhaite représenter les nombres entiers relatifs e compris dans l'intervalle $[-2^{k-1} ; 2^{k-1} - 1]$. On a donc n compris dans l'intervalle $[0 ; 2^k - 1]$ avec un décalage correspondant à 2^{k-1}

Exemple :

Sur 8 bits, $e \in [-127 ; 128]$ *et* $n \in [0; 255]$

Il existe plusieurs formats :

Format	s	$Décalage$	$n = e + Décalage$	$p_f = m - 1$
Simple précision : 32 bits	Sur 1 bit	127	Sur 8 bits	Sur 23 bits
Double précision : 64 bits	Sur 1 bit	1023	Sur 11 bits	Sur 52 bits
Quadruple précision : 128 bits	Sur 1 bit	16383	Sur 15 bits	Sur 112 bits



Exemple de représentation en IEE-754 en simple précision

Convertir (coder) un réel en binaire selon la norme IEEE-754 :

Méthode :

- On détermine le signe, on obtient donc la valeur binaire de s (0 si positif et 1 si négatif)
- On écrit notre nombre réel par la méthode de la virgule fixe, et on déplace la virgule pour obtenir une écriture de la forme $(1,xxxxxxxxx)_2$, ce déplacement de la virgule donne la valeur de l'exposant e, il est exprimé en base 10 et un décalage de la virgule à gauche donne un exposant e positif tandis qu'un décalage de la virgule à droite donne un exposant e négatif.

On applique ensuite la formule $n = e + décalage$ qu'il nous reste à convertir en binaire sur le nombre de bits correspondant au format.

- On garde ensuite la partie fractionnaire de la mantisse P_f que l'on converti en binaire sur le nombre de bits correspondant au format

Exemple :

Nous voulons représenter le réel -14,2357 en binaire norme IEEE-754 32 bits

- Le nombre est négatif, donc on représente s par un 1
- Ensuite on écrit $(14{,}2357)_{10}$ en base 2 par la méthode de la virgule fixe
  - Partie entière : $(14)_{10} = (1110)_2$
  - On code la partie décimale : $(0{,}2357)_{10}$ sur 23 bits au maximum

0.2357*2 = 0.4714

0.4714*2 = 0.9428

0.9428*2 = 1.8856

0.8856*2 = 1.7712

0.7712*2 = 1.5424

0.5424*2 = 1.0848

0.0848*2 = 0.1696

0.1696*2 = 0.3392

0.3392*2 = 0.6784

0.6794*2 = 1.3568

0.3568*2 = 0.7136 donc $(0{,}2357)_{10} = (0011110\ 00101011\ 01101010)_2$ sur 23 bits

0.7136*2 = 1.4272 Attention il s'agit ici d'une approximation !

0.4272*2 = 0.8544

0.8544*2 = 1.7088

0.7088*2 = 1.4176

0.4176*2 = 0.8352

0.8352*2 = 1.6704

0.6704*2 = 1.3408

0.3408*2 = 0.6816

0.6816*2 = 1.3632

0.3632*2 = 0.7264

0.7264*2 = 1.4528

0.4528*2 = 0.9056

Alors on virgule fixe, on a $(14{,}2357)_{10} \approx (1110{,}0011110\ 00101011\ 01101010)_2$

On décale la virgule à gauche de 3 rangs, et à chaque fois que l'on décale d'un rang à gauche en binaire, on multiplie par 2, donc ici on a multiplié par 2*2*2 = 2^3 donc **e = $(3)_{10}$**

$$\boldsymbol{n = e + décalage = 3 + 127 = 130}$$

donc n en binaire = $(130)_{10} = (1000\ 0010)_2$

On a ainsi $(14{,}2357)_{10} = (1110{,}0011110\ 00101011\ 01101010)_2 = (1{,}11\ 00011110\ 00101011\ 01101010)_2 * 2^3$

Or la partie fractionnaire de la mantisse ne doit avoir que 23 bits, donc on ne garde que 23 bits

$\boldsymbol{p_f} \approx (1100\ 0111\ 1000\ 1010\ 1101\ 101)_2$, ce qui donne pour ce nombre la représentation suivante

1	1	0	0	0	0	0	1	0	1	1	0	0	0	1	1	1	1	0	0	0	1	0	1	0	1	1	0	1	1	0	1

Convertir un nombre codé selon la norme IEEE-754 en décimal :

Méthode :

- On détermine le signe à partir du premier bit (0 si positif et 1 si négatif)
- On convertit n en décimal et on lui retranche le décalage, cela nous donne l'exposant e
- On convertit la partie fractionnaire de la mantisse P_f en décimal et on ajoute 1 pour obtenir la mantisse m
- On fait le bilan avec la formule $x = s * m * 2^e$

Exemple :

Prenons la représentation suivante, et donner le nombre réel correspondant en décimal

0	1	0	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$\boldsymbol{s = 0}$**, donc le nombre est positif**

$n = (1000\ 0100)_2 = (132)_{10}$

Or $\boldsymbol{e = n - 127 = 132 - 127 = 5}$

La partie fractionnaire de la mantisse est $(011\ 0011\ 0000\ 0000\ 0000\ 0000)_2$

$$= 2^{-2} + 2^{-3} + 2^{-6} + 2^{-7} = (0{,}3984375)_{10}$$

Donc **m** = 1 + 0.3984375 ≈ **1.3984375**

Donc le nombre réel codé dans l'exemple ci-dessus est : + 1,3984375 $*2^5 = 44{,}75$

Remarque :

- On considère par convention qu'un nombre vaut zéro si tous les bits de son exposant décalé n et de sa partie fractionnaire P_f sont nuls
- La valeur extrême (tous les bits à 1) de l'exposant décalé n avec tous les bits de la partie fractionnaire P_f à 0 correspond aux infinis (positif ou négatif suivant le signe)