Représentation numérique de l'information

Séquence 4 : Nombres réels

Xavier OUVRARD
Lycée International Ferney-Voltaire
IREM de Lyon

Cours ISN 2012-13

Codage des nombres à virgule

- Un nombre décimal est composé d'une partie entière et d'une partie fractionnaire après la virgule.
- En base B, ce nombre X s'écrit :

$$(X)_B = a_n a_{n-1} ... a_0 , b_1 ... b_m$$

Il se convertit en décimal en :

$$(X)_{10} = a_n B^n + ... + a_0 B^0 , b_1 B^{-1} + ... + b_m B^{-m}.$$

Codage des nombres à virgule (2)

Exemples :

- $128,75 = 1 \times 10^2 + 2 \times 10^1 + 8 \times 10^0 + 7 \times 10^{-1} + 5 \times 10^{-2}$.
- $(101,01)_2 = 1 \times 2^2 + 1 \times 2^0 + 1 \times 2^{-2}$
  $= 1 \times 4 + 1 + 0,25$
  $= 5,25$
- $(AE,1F)_{16} = 10 \times 16^1 + 14 \times 16^0 + 1 \times 16^{-1} + 15 \times 16^{-2}$
  $= 160 + 14 + 0,0625 + 0,5859375$
  $= 174,12109375$

Conversion des nombres à virgule en base B

- Cela n'est faisable le plus souvent que de manière approchée, il faudra donc donner la précision voulue.
- Pour la partie entière, on fait comme pour les entiers
- Pour la partie décimale :
  - On multiplie la partie entière par B
  - On note la partie entière obtenue
  - On recommence avec la partie décimale restante
  - On s'arrête quand la partie décimale est nulle ou quand la précision souhaitée est atteinte

  La partie décimale est la concaténation des parties entières obtenues dans l'ordre de leur calcul.

Conversion des nombres à virgule en base B

- Exemple : conversion de 28,8625 en binaire
  - Conversion de 28 : $(11100)_2$.
  - Conversion de 0,8625 :
    - 0,8625 x 2 = 1,725 = 1 + 0,725
    - 0,725 x 2 = 1,45 = 1 + 0,45
    - 0,45 x 2 = 0,9 = 0 + 0,9
    - 0,9 x 2 = 1,8 = 1 + 0,8
    - 0,8 x 2 = 1,6 = 1 + 0,6
    - 0,6 x 2 = 1,2 = 1 + 0,2
    - 0,2 x 2 = 0,4 = 0 + 0,4
    - 0,4 x 2 = 0,8 = 0 + 0,8 …

28,8625 peut être représenté par $(11100,11011100\ldots)_2$

Conversion des nombres à virgule en base B

- La représentation précédente ne permet pas de représenter des nombres très petitement proche de zéro ou très grands.

Représentation en virgule fixe

- Principe : on représente un réel par un nombre entier
  - Utilisation d'un facteur d'échelle F implicite
  - Représenter un nombre entier en base b : on sait faire
- Soit x un réel représenté par X, nombre entier en base b sur n chiffres

$$X = (a_{n-1}...a_0)_B$$

$$x = X.b^{-F} = (a_{n-1}...a_F, a_{F-1}...a_0)_b$$

F chiffres

Représentation en virgule fixe

- Encore utilisé pour des raisons de rapidité, les opérations en virgule fixe étant des opérations entières
- Facteur de mise à l'échelle est implicite :
  - Suppose connaissance de l'ordre de grandeur des données par le développeur
  - Engendre des difficultés de développement
- Les nombres représentés ne doivent pas être d'ordres de grandeurs très différents

Codage des décimaux en virgule flottante

- Un nombre décimal X est représenté en base b par :

$$(-1)^s \; M \times b^E$$

  - s : signe du nombre
  - M : mantisse, écrite en virgule fixe en base b, sur p chiffres, de type $x_0 x_1 ... x_i , x_{i+1} .. x_{p-1}$ où x_i, pour i entre 0 et p-1, est entre 0 et b-1
  - E : exposant

Le nombre flottant X est alors dit de précision p.

Codage des décimaux en virgule flottante

- b^E correspond au facteur de mise à l'échelle

  => il est explicite

- La représentation n'est pas unique.

  Par exemple, avec b=10, et en précision 4, le nombre 2,617 peut se présenter de différentes manières :

  - 2617×10^{-3}
  - $261,7 \times 10^{-2}$
  - $26,17 \times 10^{-1}$
  - $2,617 \times 10^{0}$
  - $,2617 \times 10^{1}$

=> Nécessité de normaliser l'écriture pour qu'elle devienne unique
=> Rôle de la norme IEEE754, publiée en 1985 et révisée en 2008

Codage en virgule flottante

- La norme IEEE 754-2008 définit 3 formats de base :
  - Simple précision (float en java)
  - Double précision (double en java)
  - Quadruple précision

Format	Taille	Précision	E_{min}	E_{max}	Valeur max
Simple	32	23 bits + 1	-126	+127	$3.403...10^{38}$
Double	64	52 bits + 1	-1022	+1023	$1.798...10^{308}$
Quadruple	128	112 bits + 1	-16382	16383	$1.190...10^{4932}$

- Taille mantisse équivaut à la précision de p bits
- Taille exposant : w bits
- $E_{max} = 2^{w-1} - 1$ et $E_{min} = 1 - E_{max}$

Codage en virgule flottante

- Problème : même avec position de la virgule fixée dans la mantisse d'un nombre flottant, un nombre peut avoir plusieurs représentations :

$$2{,}190 \times 10^{1} \quad \text{et} \quad 0{,}219 \times 10^{2}.$$

- La mantisse s'écrit : $M = m_0 m_1 ... m_{p-1}$ pour une précision de p bits.
- Pour pallier au problème de la non unicité, on normalise la mantisse, i.e. $m_0 \neq 0$.
  - Représentation unique (pour les valeurs non nulles)
  - Représentation qui minimise l'exposant
  - En base 2, $m_0 = 1$ => m_0 n'est pas stocké en mémoire =>bit implicite=> utilisé pour le signe

Nombres flottants normalisés en mémoire

- On code sur N bits, avec une précision p

- Se souvenir que la partie entière de la mantisse en base 2 correspond au bit implicite, qui vaut 1
- L'exposant peut être négatif => comparaison est alors difficile

  => l'exposant est encodé en utilisant une représentation biaisée $E + E_{max}$.

- Pas de valeur codable entre 0 et 2^{Emin}.

Exemple

- Trouver le nombre à virgule correspondant à :

  1100010001101001001111000011100000000000000000000000000000000000

- => codage sur 64 bits
  - Signe 1 bit : 1, le nombre est donc négatif
  - Exposant codé sur 11 bits : 10001000110,

    $E_{codé}$ = 1094, donc $E_{réel} = E_{codé} – E_{max}$ = 1094-1023 = 71
  - Mantisse codée sur 53-1 = 52 bits (et en ajoutant le bit implicite)

  M = 1,1001001111000011100000000000000000000000000000000000

  = $1 + 1/2 + 1/2^4+1/2^7+1/2^8+1/2^9+1/2^{10}+1/2^{15}+1/2^{16}+1/2^{17}$

  = 206 727/131072

  Le nombre représenté est $-206727/131072 \times 2^{71} = -3{,}724... \times 10^{21}$

Sources

- Représentation des nombres, arithmétique flottante, norme IEEE 754 , Guillaume Revy, Université de Perpignan
- ISN en Terminale S, Gilles Dowek
- Codage des nombres, Eric Cariou Université de Pau et des Pays de l'Adour
- Arithmétique flottante Vincent Lefèvre, Paul Zimmermann, INRIA
- http://blog.netinfluence.com/2009/09/24/comprendre-les-nom

Exemple de la simple précision

- En simple précision, on code sur 32 bits

- Si $0 \leq E_{codé} < 255$ et $M_{codé} \neq 0$, cela représente le nombre

$$(-1)^s \times 1,M_{codé} \times 2^{Ecodé-127}$$

- Valeurs particulières :
  - + infini : s=0 $E_{codé}$=255 $M_{codé}$=0
    0 11111111 00000000000000000000000
  - - infini : s=1 $E_{codé}$=255 $M_{codé}$=0
    1 11111111 00000000000000000000000
  - NaN, Not a number : $E_{codé}$ = 255 $M_{codé} \neq 0$
  - Zéro : s=$\pm$1 $E_{codé}$=0 $M_{codé}$=0