Représentation des données – Arithmétique flottante

Tanmoy MONDAL

tanmoy.mondal@lirmm.fr

Diapos de D. Delahaye et Chouki TIBERMACINE

Système de numération octal

- Utilise huit chiffres, 0,1,2,3,4,5,6,7.
- Aussi appelé système de base 8

Step	Octal Number	Decimal Number
Step 1	12570_8	$((1 \times 8^4) + (2 \times 8^3) + (5 \times 8^2) + (7 \times 8^1) + (0 \times 8^0))_{10}$
Step 2	12570_8	$(4096 + 1024 + 320 + 56 + 0)_{10}$
Step 3	12570_8	5496_{10}

Système de numération Hexadécimal

- Utilise 10 chiffres et 6 lettres, 0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F.
- Les lettres représentent des nombres à partir de 10. A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.
- Aussi appelé système de base 16.

Step	Hexadecimal Number	Decimal Number
Step 1	$19FDE_{16}$	$((1 \times 16^4) + (9 \times 16^3) + (F \times 16^2) + (D \times 16^1) + (E \times 16^0))_{10}$
Step 2	$19FDE_{16}$	$((1 \times 16^4) + (9 \times 16^3) + (15 \times 16^2) + (13 \times 16^1) + (14 \times 16^0))_{10}$
Step 3	$19FDE_{16}$	$(65536 + 36864 + 3840 + 208 + 14)_{10}$
Step 4	$19FDE_{16}$	106462_{10}

Conversions de base

- Suivre le tableau
- voir : https://www.tutorialspoint.com/digital_circuits/digital_circuits_base_conversions.htm

Représentation des nombres réels

- Un nombre réel dans le système décimal peut s'écrire :

$$n = d_m d_{m-1} \ldots d_1 d_0 . d_{-1} d_{-2} \ldots d_{-p}$$

- La valeur du nombre :

$$n = \sum_{i=-p}^{m} d_i \text{ x } 10^i$$

$5 * 10^1$	50
$6 * 10^0$	6
$4 * 10^{-1}$	4/10
$8 * 10^{-2}$	8/100
$2 * 10^{-3}$	2/1000

- Exemple :

$$23.375 = 2x10^1 + 3x10^0 + 3x10^{-1} + 7x10^{-2} + 5x10^{-3} = 23375/1000$$

Nombres réels en binaire

- Un nombre réel dans le système binaire peut être écrit :

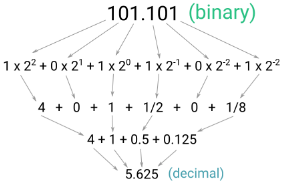
$$n = b_m b_{m-1} \ldots b_1 b_0 . b_{-1} b_{-2} \ldots b_{-p}$$

- La valeur du nombre :

$$n = \sum_{i=-p}^{m} b_i \, x \, 2^i$$

$1 * 2^2$	4
$0 * 2^1$	0
$1 * 2^0$	1
$1 * 2^{-1}$	1/2
$0 * 2^{-2}$	0
$1 * 2^{-3}$	1/8

- Exemple :



Du décimal au binaire

- La multiplication est utilisée
- S'il s'agit d'un nombre entier ($> = 1.0$), le bit est 1
- $0.375_{10} \longrightarrow ?_2$
  $0.375 \text{ x } 2 = 0.75 = \boxed{0} + 0.75$
  $0.75 \text{ x } 2 = 1.5 = \boxed{1} + 0.5$
  $0.5 \text{ x } 2 = 1.0 = \boxed{1} + 0.0$
- $0.375_{10} \longrightarrow 0.\boxed{011}_2$

- La partie décimale est ensuite utilisée pour le calcul suivant
- Une fois que le résultat atteint 1.0, la conversion est terminée

Du décimal au binaire

- Il y a beaucoup de nombres qui n'aboutissent pas à un résultat de 1.0
- Une fois que le résultat atteint 1.0, la conversion est terminée
- Puisqu'il y a *t* bits possibles pour la mantisse
- La conversion se termine dès que *t* bits sont atteints
- $0.4_{10} \longrightarrow ?_2$

  $0.4 \times 2 = 0.8 = \boxed{0} + 0.8$

  $0.8 \times 2 = 1.6 = \boxed{1} + 0.6$

  $0.6 \times 2 = 1.2 = \boxed{1} + 0.2$

  $0.2 \times 2 = 0.4 = \boxed{0} + 0.4$

  $\Rightarrow 0.4 \times 2 = 0.8 = \boxed{0} + 0.8$
- $0.4_{10} \longrightarrow 0.\boxed{0110}\boxed{[0110]}_2$

Du binaire au décimal

Quelle est la valeur en décimal des nombres binaires suivants ?

- $0.1_2 = ?_{10}$
- $0.01_2 = ?_{10}$
- $0.11_2 = ?_{10}$
- $0.1001_2 = ?_{10}$

Du binaire au décimal

Quelle est la valeur en décimal des nombres binaires suivants ?

- $0.1_2 = 0.5_{10}$
- $0.01_2 = 0.25_{10}$
- $0.11_2 = 0.5_{10} + 0.25_{10} = 0.75_{10}$
- $0.1001_2 = 0.5_{10} + 0.0625_{10} = 0.5625_{10}$

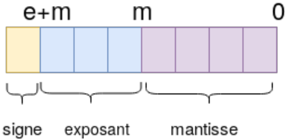
Nombres réels en notation scientifique

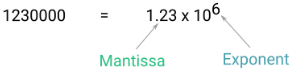
- Notation : $\pm m \; x \; 10^n$ ou $\pm m$ E^e
  où : $\pm$ est le **signe**, m est la **mantisse** et e est l'**exposant**
- 123 400 000 000 000 s'écrit $1.234 \; x \; 10^{14}$ ou 1.234E14
- 0.000 000 000 000 123 s'écrit $1.23 \; x \; 10^{-13}$ ou 1.23E-13

Nombres réels binaires en virgule flottante (équiv. notation scientifique)

- Dans le cas général, on écrit : $\pm m \ x \ B^e$
  où B est la base
- Dans le système binaire : $\pm m \ x \ 2^e$
  où *m* (mantisse) est exprimée sous la forme d'un nombre binaire
- En variant l'exposant, on fait "flotter" la virgule
- Avantage : Pour un même nombre de bits donné, on peut représenter un intervalle de nombres plus important que les représentations des entiers ou à virgule fixe

Codage binaire des nombres réels





- Le codage sur un nombre n (=e+m+1) de bits, fixe, implique un nombre fini de valeurs
- Ceci implique des calculs arrondis (perte de précision) et des erreurs d'arrondi
- Un même nombre peut être représenté de différentes façons : $0.110 \times 2^5 = 110 \times 2^2 = 0.0110 \times 2^6$

Codage binaire des nombres réels -suite-

1. Convertir séparément les entiers et les décimales
2. Ajoutez $\times 2^0$ à la fin du nombre binaire (qui ne change pas sa valeur)
3. Pour éviter des représentations différentes d'un même nombre, la mantisse est normalisée
   - Couramment, un nombre (différent de zéro) avec une mantisse normalisée a la forme suivante : $\pm 1.bbb\ldots x2^e$
   - Le chiffre 1 à gauche du point décimal est retiré de la représentation pour gagner un bit (il devient implicite)
4. Avec la notation normalisée (mantisse : 1.bbb x 2^e), omettez 1 tout à gauche et remplissez avec des zéros à droite

Codage binaire des nombres réels -suite-

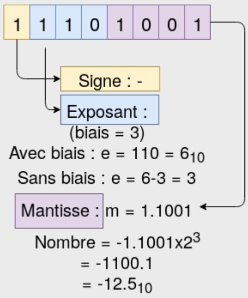
1. Représenter l'exposant avec un décalage ("Biais")
2. Biais = $2^{|e|-1} - 1$ (|e| : taille de l'exposant)
3. Pour un exposant sur |e| bits :
   i Au lieu de représenter les nombres de 0 à $2^{|e|} - 1$ (Ex : |e|=8, [0,255])
   ii On représentera les nombres [$-2^{|e|-1}$-1 , $2^{|e|-1}$]
   iii (pour |e|=8, biais = 2^{8-1}-1 = 127 → [-127,128])
4. Définissez le bit de signe, 1 pour négatif, 0 pour positif, en fonction du signe du nombre initial

Représentation des nombres flottants

Exercice

Quelle est la valeur du nombre représenté en virgule flottante de la façon suivante, avec |e|=3 |m|=4 ? 1 110 1001

Solution



Normalisation du codage des flottants

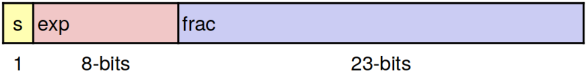
Norme IEEE 754 (standard)

- Format simple précision : 32 bits
  - Bit du signe (1 bit) ;
  - Exposant (8 bits) ;
  - Mantisse (23 bits).
- Format double précision : 64 bits
  - Bit du signe (1 bit) ;
  - Exposant (11 bits) ;
  - Mantisse (52 bits).
- Autres formats :
  - Simple précision étendue ($\geq$ 43 bits, obsolète) ;
  - Double précision étendue ($\geq$ 79 bits, long double de C).

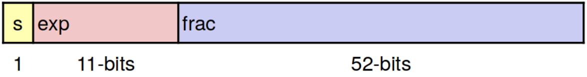
Normalisation du codage des flottants

Precisions

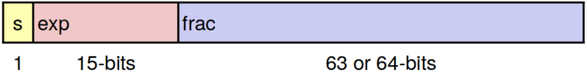
- Single precision: 32 bits



- Double precision: 64 bits



- Extended precision: 80 bits (Intel only)



D'une représentation à l'autre

Exemples en simple précision

- Valeur décimale de :

  0 10000010 11000000000000000000000

  - 0 = positif $\Rightarrow$ s=+1 ;
  - e_d = 10000010_2 = 130_{10}, e = 130_{10} - 127_{10} = 3_{10} ;
  - m = 1.11_2 = 1.75_{10} ;
  - n = $+1 \times 1.75 \times 2^3 = 14$;
  - ou bien n = $+1 \times 1.11_2 \times 2^3 = 1110_2 = 14$.

D'une représentation à l'autre

Exemples en simple précision

- Valeur binaire de : -118.625_{10}
  - bit de signe = 1 (négatif) ;
  - 118_{10} = 1110110_2 ;
  - $0.625 \times 2 = 1.25 = \underline{1} + 0.25$;
  - $0.25 \times 2 = 0.5 = \underline{0} + 0.5$;
  - $0.5 \times 2 = 1.0 = \underline{1} + 0$;
  - 0.625 = 101_2 ;
  - 118.625_{10} = 1110110.101_2 = 1.110110101×2^6 ;
  - e_d = 6 + 127 = 133 = 10000101_2 ;
  - représentation : 1 10000101 11011010100000000000000.

Problèmes de représentation de certains nombres

Comment représenter les exposants négatifs ?

- Encodage : on ajoute le biais
  Ex sur 3 bits (biais = 3) : exposant = $2_{10} \longrightarrow 2+3 = 5 = 101_2$
- décodage : on retire le biais
  Ex sur 3 bits : $010_2 \longrightarrow 2-3 = -1$

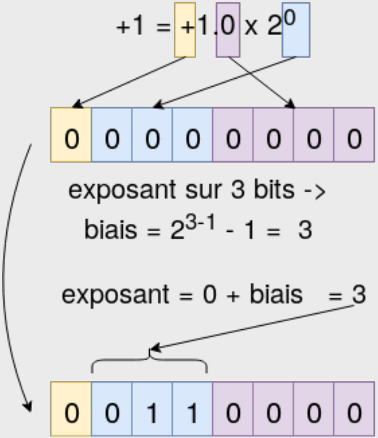
Problèmes de représentation de certains nombres

Exercice

Écrire le nombre +1 avec un exposant de 3 bits et une mantisse de 4 bits

Problèmes de représentation de certains nombres

Solution



D'une représentation à l'autre

Exercice en simple précision

- Donner la valeur décimale de :
  - 1 10000010 11110110000000000000000.
- Donner la représentation de :
  - 3.1416015625_{10}.
- Que se passe-t-il si l'on souhaite représenter 3.14_{10} ?
- Comment obtient-on le plus petit nombre normalisé positif ?
- Comment obtient-on le plus petit nombre dénormalisé positif ?

Correction

- Donner la valeur décimale de :
  - 1 10000010 11110110000000000000000 ;
  - Solution : -15.6875.
- Donner la représentation de :
  - 3.1416015625_{10} ;
  - Solution : 0 10000000 10010010001000000000000.
- Pour 3.14_{10}, il n'est pas exactement représentable. Il est donc approximé (on s'arrête à la fin de la mantisse, troncature).
- Plus petit nombre normalisé positif :
  - Plus petit e_d non nul (00000001) et mantisse nulle ;
  - Résultat : 2^{-126}.
- Plus petit nombre dénormalisé positif :
  - Exposant décalé : 0 (par définition) ;
  - Mantisse : 00000000000000000000001 ;
  - Résultat : $2^{-23} \times 2^{-126} = 2^{-149}$.

Calculs sur les flottants

Addition

- Addition de :
  - X = 0 10000001 11000000000000000000000 ;
  - Y = 0 01111111 00000000000000000000000 ;
  - X = 1.11×2^2, Y = 1.0×2^0 ;
  - On aligne les exposants (sur le plus grand) ;
  - Y = 0.01×2^2 ;
  - On additionne les mantisses :
    1.11000000000000000000000 +
    0.01000000000000000000000 =
    10.00000000000000000000000
  - Résultat = $10.0 \times 2^2 = 1.0 \times 2^3$;
  - Codage : 0 10000010 00000000000000000000000.

Calculs sur les flottants

Exercice

- Additionner les deux flottants suivants :
  - X = 0 10000001 11000000000000000000000 ;
  - Y = 0 01111110 11000000000000000000011.
- Additionner les deux flottants suivants :
  - X = 0 11111110 11000000000000000000000 ;
  - Y = 0 01111111 11000000000000000000000.

Calculs sur les flottants

Solution : Additionner les deux flottants

- X = 0 10000001 11000000000000000000000 ;
- Valeur de X :
  - $e_d = 10000001_2 = 129_{10}$, e = 129-127 = 2
  - Valeur de X = 1.11 x 2^2
- Y = 0 01111110 11000000000000000000011.
- Valeur de Y :
  - $e_d = 01111110_2 = 126_{10}$, e = 126-127 = -1
  - Valeur de Y = 1.11000000000000000000011 x 2^{-1}
- Aligner les exposants :
  Valeur de Y = 0.00111 x 2^2 **(perte de précision)**
- Somme = 1.11111 x 2^2
- Codage : 0 10000001 11111000...0

Calculs sur les flottants

Solution : Additionner les deux flottants

- X = 0 11111110 11000000000000000000000
- Valeur de X :
  - $e_d = 11111110_2 = 254_{10}$, e = 254-127 = 127
  - Valeur de X = 1.11×2^{127} (Très grand nombre)
- Y = 0 01111111 11000000000000000000000
- Valeur de Y :
  - $e_d = 01111111_2 = 127_{10}$, e = 127-127 = 0
  - Valeur de Y = 1.11×2^{0}
- Aligner les exposants :
  Valeur de Y = $0.00....0 \times 2^{127}$ **(perte de précision)**
- Somme = 1.11×2^{127} = Valeur de Y (absorption)
- Codage = X

Calculs sur les flottants

Exercice

- Multiplier les deux flottants suivants :
  - X = 0 10000001 01000000000000000000000 ;
  - Y = 0 01111111 11000000000000000000000.

Calculs sur les flottants

Correction

- Multiplier les deux flottants suivants :
  - X = 0 10000001 01000000000000000000000 ;
  - Y = 0 01111111 11000000000000000000000 ;
  - $X = 1.01 \times 2^2$, $Y = 1.11 \times 2^0$;
  - Addition des exposants : $2 + 0 = 2$;
  - Multiplication des mantisses : $1.01 \times 1.11 = 10.0011$;
  - Résultat = 1.00011×2^3 (exposant incrémenté) ;
  - Codage : 0 10000010 00011000000000000000000.

Diapos et références

Diapos constuites sur la base du cours de :

David Delahaye, professeur à la FDS (mon prédécesseur)
Chouki TIBERMACINE, MCF à PolyTech-Montpellier (mon prédécesseur)

Références bibliographiques

- Paolo Zanella, Yves Ligier et Emmanuel Lazard. Architecture et technologie des ordinateurs - 6e éd. - Cours et exercices corrigés. Septembre 2018
- Utilisation des nombres à virgule flottante (risques) : `https://www.ekito.fr/people/les-nombres-virgule-flottante/`