Fonctionnement des Ordinateurs

TP4 - Nombres flottants

B. QUOITIN
Faculté des Sciences
Université de Mons

Résumé

L'objectif de cette séance d'exercices est de renforcer votre compréhension de la représentation des nombres flottants et du standard IEEE 754.

Table des matières

1 Motivation

1.1 Erreur suite à une addition

Ecrivez un programme en Python 3 qui stocke dans les variables x et y, les nombres $0,1$ et $0,2$ respectivement. Affichez les valeurs de ces variables à la console. Effectuez ensuite l'addition $x + y$ et affichez la somme à la console. Que constatez-vous ? Comment expliquez-vous ce phénomène ?

Q1) Programme en Python

```
>>> x = 0.1
>>> y = 0.2
>>> x
0.1
>>> y
0.2
>>> x+y
0.30000000000000004
```

```
>>> print("%.17f" % (x))
???
>>> print("%.17f" % (y))
???
```

1.2 Swamping lors de l'addition

Ecrivez un programme en langage C ou Java qui effectue l'addition des nombres $x = 2$ et $y = 33554432$ (2^{25}) placés dans des variables de type `float` (IEEE 754, en simple précision). Que constatez-vous ?

Q2) Programme en C

```c
#include <stdio.h>
int main() {
  float x= 2.0;
  float y= 33554432.0;
  printf("x=%f\n", x);
  printf("y=%f\n", y);
  printf("x+y=%f\n", (x+y));
  return 0;
}
```

1.3 Somme de fractions

Ecrivez un programme en C ou Java qui ajoute 700 fois la fraction $\frac{1}{7}$ à une variable x valant initialement 0. Utilisez d'abord le type `float` et ensuite le type `double`. Que constatez-vous ?

Q3) Programme en C

```c
#include <stdio.h>

int main()
{
  float sum= 0.0;
  int i;
  for (i= 0; i < 700; i++)
```

```
    sum+= (1.0/7);
  printf("somme␣=␣%.10f\n", sum);
  printf("resultat␣attendu␣=␣%.10f\n", 700*(1.0/7));
  return 0;
}
```

Pouvez-vous écrire un programme donnant un comportement similaire en Python ?

2 Virgule flottante

2.1 Représentation IEEE754

Le standard IEEE 754 définit des représentations de nombres flottants. Cette représentation est composée d'un *bit de signe*, d'un *exposant* représenté sur E bits et d'une *mantisse* représentée sur M bits. Donnez les paramètres des représentations en simple et double précisions. La variable B est le biais.

Paramètres IEEE 754

Précision	Taille (bits)	E	M	B
simple	32	8	23	127
double	64	11	52	1023

Donnez les formules permettant d'interpréter un mot binaire de 32 bits (simple précision) selon ce format. Attention, par convention certaines valeurs spéciales sont encodées et il est important de distinguer les représentations normalisée et dénormalisée.

Q4) Interprétation nombre IEEE 754 simple précision

Représentation normalisée ($0 < e < 2^E - 1$) **:**

$$x = (-1)^s \left(1 + \frac{m}{2^M}\right) 2^{e-B}$$

Représentation dénormalisée ($e = 0$) **:**

$$x = (-1)^s \left(0 + \frac{m}{2^M}\right) 2^{1-B}$$

Infini ($e = 2^E - 1$ **et** $m = 0$)
Not-a-Number* ou *NaN ($e = 2^E - 1$ **et** $m \neq 0$)

Donnez les nombres représentés en IEEE 754 simple précision avec les mots binaires suivants.

Q5) `0 00000000 00000000000000000000000`

$m = 0$ et $e = 0$. Représentation dénormalisée. Il s'agit d'une représentation de $\left(0 + \frac{0}{2^M}\right) 2^{1-B} = 0$.

Q6) `0 10000000 00000000000000000000000`

$m = 0$ et $e = 128$. Représentation normalisée. $x = \left(1 + \frac{0}{2^M}\right) 2^{128-127} = 2$.

Q7) `0 10000000 11000000000000000000000`

$m = 2^{22} + 2^{21}$ et $e = 128$. $x = \left(1 + \frac{2^{22}+2^{21}}{2^{23}}\right) 2^{128-127} = 1.75 \times 2 = 3,5$.

Q8) `0 11111111 01111111111111111111111`

$m = 2^{22} - 1$ et $e = 255$. *NaN*.

Q9) `1 00000000 10000000000000000000000`

$m = 2^{22}$ et $e = 0$ et $s = 1$. Représentation dénormalisée : $s = -\left(0 + \frac{2^{22}}{2^{23}}\right) 2^{1-127} = -2^{-127}$

2.2 Limites de la représentation IEEE754 (format réduit)

Donnez les plus grand et plus petit nombres positifs non nuls normalisés représentables avec les tailles de mantisse M, d'exposant E et les biais B suivants.

Q10) $M = 4, E = 3, B = 4$

$m_{\text{min}} = 0$
$e_{\text{min}} = 1$
$x_{\text{min}} = \left(1 + \frac{0}{2^M}\right) 2^{e_{\text{min}} - B} = \frac{1}{8}$ *(0,125)*
Ce nombre serait représenté par `0 001 0000`.

$m_{\text{max}} = 2^M - 1 = 15$
$e_{\text{max}} = 2^E - 2 = 6$ *(rappel : $2^E - 1$ signale NaN ou ∞)*
$x_{\text{max}} = \left(1 + \frac{m_{\text{max}}}{2^M}\right) 2^{e_{\text{max}} - B} = \frac{31}{4}$ *(7,75)*
Ce nombre serait représenté par `0 110 1111`.

Q11) $M = 8, E = 4, B = 8$

$x_{\text{min}} = \left(1 + \frac{0}{256}\right) 2^{1-B} = \frac{1}{128}$ *(0,0078125)*
Ce nombre serait représenté par `0 0001 00000000`.

$x_{\text{max}} = \left(1 + \frac{255}{256}\right) 2^{14-B} = \frac{511}{4}$ *(127,75)*
Ce nombre serait représenté par `0 1110 11111111`.

Question identique pour les nombres dénormalisés.

Q12) $M = 4, E = 3, B = 4$

$x_{\text{min}} = \left(0 + \frac{1}{2^M}\right) 2^{1-B} = \frac{1}{16} \times 2^{-3} = \frac{1}{128}$ *(0,0078125)*

$x_{\text{max}} = \left(0 + \frac{2^M - 1}{2^M}\right) 2^{1-B} = \frac{15}{16} \times 2^{-3} = \frac{15}{128}$ *(0,1171875)*

Q13) $M = 8, E = 4, B = 8$

$x_{\text{min}} = \left(0 + \frac{1}{2^M}\right) 2^{1-B} = \frac{1}{256} \times 2^{-7} = \frac{1}{32768}$ *(0,000030517578125)*

$x_{\text{max}} = \left(0 + \frac{2^M - 1}{2^M}\right) 2^{1-B} = \frac{255}{256} \times 2^{-7} = \frac{255}{32768}$ *(0,007781982421875)*

2.3 Normalisation d'un nombre

Ecrivez un algorithme qui prend en argument un nombre réel x non nul et qui donne comme résultat la valeur x_{norm} normalisée de x, c'est-à-dire telle que $x_{\text{norm}} \in [1, 2[$. Arrangez-vous également pour que votre algorithme fournisse l'exposant e de 2 tel que $x = x_{\text{norm}} \times 2^e$, avec $e \in \mathbb{Z}$.

Q14) Programme en C

```c
#include <stdio.h>

double normalize(double x, int * pn) {
  int n = 0;
  while (1) {
    if (x < 1) {
      x *= 2;
      n--;
```

```
    }
    else if (x >= 2) {
      x /= 2;
      n++;
    }
    else
      break;
  }
  if (pn != NULL)
    *pn = n;
  return x;
}

int main() {
  double x, xp;
  int n;

  x = 7.33;
  printf("x = %.10f\n", x);
  xp = normalize(x, &n);
  printf("x' = %.10f\n", xp);
  printf("n = %d\n", n);

  x = 0.017;
  printf("x = %.10f\n", x);
  xp = normalize(x, &n);
  printf("x' = %.10f\n", xp);
  printf("n = %d\n", n);
}
```

2.4 Conversion vers IEEE754 (format réduit)

Soit une taille de mantisse $M = 4$, une taille d'exposant $E = 3$ et un biais $B = 4$. Donnez la représentation des nombres suivants.
Pour déterminer la représentation en virgule flottante d'un nombre x, l'approche suivante est typiquement utilisée :

1. **Signe :**
   Si $x < 0$, le bit de signe s vaut 1, sinon il vaut 0. Par la suite, on considère la valeur absolue de x.
2. **Normalisation :**
   x est normalisé à l'aide de n divisions par 2 successives afin d'obtenir $x' \times 2^n = x$ où $1 \leq x' < 2$. Par exemple, $x = 6,2$ est divisé $n = 2$ fois par 2 pour obtenir le nombre normalisé $x' = 1,55$ tel que $x' \times 2^2 = x$.
3. **Déduction de l'exposant :**
   On pose $2^n = 2^{e-B}$, ce qui permet de déduire $e = n + B$.
   On vérifie que le nombre est bien représentable. Si $0 < e < 2^E - 1$, alors la représentation normalisée doit être utilisée. Si $e \leq 0$, alors la représentation dénormalisée doit être utilisée. Si $e \geq 2^E - 1$, alors le nombre ne peut être représenté.
4. **Déduction de la mantisse en représentation normalisée :**
   On pose $x' = 1 + \frac{m}{2^M}$, ce qui permet de déduire $m = (x' - 1) \times 2^M$.
5. **Déduction de la mantisse en représentation dénormalisée :**
   Ici, l'exposant est fixe $e = 1$. On pose $x = \frac{m}{2^M} \times 2^{1-B}$, ce qui permet de déduire $m = x \times 2^{M-(1-B)}$.

Q15) $1,5$

$x = 1,5$ est déjà normalisé.
On déduit $e = n + B = 4$ (100). La représentation normalisée peut être utilisée car $0 < e < 7$.
On déduit $m = (x - 1) \times 2^4 = 8$ (1000).
La représentation de x est par conséquent `0 100 1000`.

Q16) $0,625$

$x = 0,625$ n'est pas normalisé. On trouve $x' = 1,25$ en multipliant x par 2 une seule fois ($n = -1$).
On déduit $e = n + B = 3$ (`101`). La représentation normalisée peut être utilisée car $0 < e < 7$.
On déduit $m = (x' - 1) \times 2^4 = 4$ (`0100`).
La représentation de x est par conséquent `0 011 0100`.

Q17) $3,2$

$x' = 1,6$ et $n = 1$.
$e = n + B = 5$ (`101`).
$m = (x' - 1) \times 2^4 = 9,6$. Ici, la mantisse n'est pas représentable exactement, il faut arrondir. La plus proche valeur entière de m est 10 (`1010`).
La représentation de $\hat{x}$, l'approximation de x, est par conséquent `0 101 1010`. La valeur de $\hat{x}$ est 3,25.

Q18) $64,17$

$x' = 1,00265625$ et $n = 6$.
$e = n+B = 10$. Ce nombre n'est pas représentable dans le système proposé car il nécessite un exposant trop grand ($e \geq 2^E - 1$).

Q19) $0,03125$

$x' = 1$ et $n = -5$
$e = n + B = -1$. La représentation dénormalisée doit être utilisée car l'exposant est petit ($e < 1$).
$e = 1$ (mais `000` signale représentation dénormalisée)
$m = x \times 2^{M-(1-B)} = x \times 2^7 = 4$ (`0100`)
La représentation de x est par conséquent `0 000 0100`.

Q20) $-0,015625$

$s = 1$, $x = 1$ et $n = -6$
$e = n + B = 11$. La représentation dénormalisée doit être utilisée car l'exposant est petit ($e < 1$).
$e = 1$ (mais `000` signale représentation dénormalisée)
$m = x \times 2^{M-(1-B)} = x \times 2^7 = 2$ (`0010`)
La représentation de x est par conséquent `1 000 0010`.

2.5 Biais adéquat

Soit une taille d'exposant E, donner la valeur du biais permettant d'équilibrer au mieux le nombre d'exposants positifs et négatifs.

Q21) $E = 3$

Les valeur d'exposant e représentables sur $E = 3$ bits sont $\{0, 1, 2, 3, 4, 5, 6, 7\}$. L'objectif du biais est d'obtenir sur cette base autant d'exposants positifs que négatifs en leur soustrayant une valeur B constante.

Les deux valeurs au centre de l'intervalle d'exposants sont $2^{E-1} - 1 = 3$ et $2^{E-1} = 4$. Par convention, dans IEEE 754, le biais choisi est $B = 2^{E-1} - 1 = 3$.

L'intervalle d'exposants résultant est donc $\{-3, -2, -1, 0, 1, 2, 3, 4\}$.

Q22) $E = 8$

$B = 2^{E-1} - 1 = 127$

2.6 Erreur absolue, erreur relative et epsilon machine

Soit une taille de mantisse $M = 4$, une taille d'exposant $E = 3$ et un biais $B = 4$. Fournissez la formule donnant l'epsilon machine, ϵ_M, la borne supérieure sur l'erreur relative. Donnez sa valeur pour $M = 4$.

Q23) epsilon machine (ϵ_M)

$$\epsilon_M = 2^{-(M+1)}$$

Pour $M = 4$, sa valeur est $\epsilon_M = 2^{-5} = 0,03125$.

Calculez l'erreur d'approximation causée lors de la représentation des nombres suivants. Fournissez l'erreur absolue (Δ_x) et l'erreur relative (ϵ_x). Comparez cette dernière à l'epsilon machine. Note : les nombres ci-dessous sont identiques à ceux de la Section 2.4.

Q24) $1,5$

Ce nombre est représentable exactement. Par conséquent, $\Delta_x = 0$ et $\epsilon_x = 0$.

Q25) $0,625$

Idem.

Q26) $3,2$

Le nombre $x = 3,2$ est approximé par $\hat{x} = 3,25$. L'erreur absolue vaut $\Delta_x = |x - \hat{x}| = 0,05$ et l'erreur relative est $\epsilon_x = \frac{\Delta_x}{|x|} = 0,015625$, ce qui est bien inférieur à ϵ_M.

Q27) $64,17$

Non représentable.

Q28) $0,03125$

Représentable exactement.

Q29) $0,015625$

Représentable exactement.

2.7 Arrondis

En utilisant la méthode *round-to-nearest-even* (*arrondi au plus proche pair*), arrondissez les nombres dont la représentation binaire est donnée ci-dessous, de façon à ne garder que la partie entière.

Q30) `101.000000`

La partie fractionnelle est nulle. Aucun arrondi n'est nécessaire. Le résultat est `101`.

Q31) `1.010101`

Ce nombre est compris entre `1` et `10`. Cependant, il est plus proche de `1`.

Q32) `11.110000`

Ce nombre est compris entre `11` et 100. Cependant, il est plus proche de `100`.

Q33) `0001.100001`

Ce nombre est compris entre `1` et `10`. Cependant, il est plus proche de `10`.

Q34) `0.1`

Ce nombre est pile-poil entre `0` et `1`. On arrondi vers le plus proche pair `0`.

Q35) `0.01111111111111111111111111111111111111`

Ce nombre est compris entre `0` et `1`. Cependant, il est plus proche de `0`.

Q36) `11111.100000`

Ce nombre est pile-poil entre `11111` et `100000`. On arrondi vers le plus proche pair `100000`.

Q37) `100.100000`

Ce nombre est pile poil entre `100` et `101`. On arrondi vers le plus proche pair `100`.

2.8 Addition

Effectuez l'addition des nombres x et y dont les représentations sont données ci-dessous. Indiquez les différentes étapes de l'opération. Le format utilisé est similaire à IEEE 754 avec les tailles de mantisse $M = 6$, d'exposant $E = 4$ et un biais $B = 8$. On ne considère pas le bit de signe. Afin de vérifier vos additions, donnez également les valeurs de x, de y et du résultat $\widehat{x+y}$ de l'addition en décimal. Attention, $\widehat{x+y}$ peut-être différent de $x+y$ en raison d'arrondis ou de dépassements de capacité.

Attention, ne pas oublier le bit caché en représentation normalisée. Dans les solutions ci-dessous, il sera indiqué entre parenthèses avant la mantisse. Par exemple, pour une mantisse valant 0, encodée sur 6 bits, on affichera `(1)000000`.

Q38) $x =$`1000 000000`; $y =$`1000 000000`

Alignement	x `1000`	`(1)000000`	
	y `1000`	`(1)000000`	
Somme		`(10)000000`	
Normalisation	`1001`	`(1)0000000`	*(décalage vers la droite, incrément de l'exposant)*
Arrondi	`1001`	`(1)000000`	

$x = 1$ et $y = 1$. Le résultat correspond à la représentation de $\left(1 + \frac{0}{2^M}\right) 2^{9-B} = 2$.

Q39) $x =$`1000 000000`; $y =$`0110 000000`

Alignement	x `1000`	`(1)000000`	
	y `1000`	`(0)01000000`	*(y doit être décalé de 2 positions vers la droite)*
Somme		`(1)01000000`	
Normalisation	`1000`	`(1)01000000`	*(rien à faire)*
Arrondi	`1000`	`(1)010000`	*(rien à faire)*

$x = 1$ et $y = 0,25$. Le résultat correspond à $\left(1 + \frac{16}{64}\right) 2^{8-8} = 1,25$.

Q40) $x =$`1000 100000`; $y =$`1000 110000`

Alignement	x	1000	(1)100000
	y	1000	(1)110000
Somme			(11)010000
Normalisation		1001	(1)1010000 *(décalage vers la droite, incrément de l'exposant)*
Arrondi		1001	(1)101000

$x = 1,5$ et $y = 1,75$. Le résultat correspond à $\left(1 + \frac{40}{64}\right) 2^{9-8} = 3,25$.

Q41) $x =$`1000 000000`; $y =$`0001 000000`

Alignement	x	1000	(1)000000
	y	1000	(0)0000001000000
Somme			(1)0000001000000
Normalisation		1000	(1)0000001000000
Arrondi		1000	(1)000000

$x = 1$ et $y = 0,0078125$ (2^{-7}). Le résultat $\widehat{x+y}$ correspond à $x = 1$ en raison de l'arrondi. Il s'agit d'un cas de *swamping*. Les deux nombres à additioner sont d'ordres de grandeur tellement différents que le plus petit des deux n'a aucun impact sur la somme.
On peut calculer l'erreur absolue $\Delta_{x+y} = \left|(x+y) - (\widehat{x+y})\right| = y = 0,0078125$ et l'erreur relative $\epsilon_{x+y} = \frac{\Delta_{x+y}}{|x+y|} \approx 0,0078$.

Q42) $x =$`1110 000000`; $y =$`1110 000000`

Alignement	x	1110	(1)000000
	y	1110	(1)000000
Somme			(10)000000
Normalisation		1111	(1)0000000
Arrondi		1111	(1)000000

$x = 64$ et $y = 64$. Le nombre représenté correspond à $+\infty$ car la somme $x + y$ n'est pas représentable dans ce système.

Q43) $x =$`0000 100000`; $y =$`0000 110000`

Attention, ici, les deux représentations sont dénormalisées (signalé par l'encodage d'un exposant nul).

Alignement	x	0000	(0)100000
	y	0000	(0)110000
Somme		0001	(1)010000 *(attention : passage en normalisé)*
Normalisation		0001	(1)010000
Arrondi		0001	(1)010000

$x = 0.5 * 2^{-7}$ et $y = 0.75 * 2^{-7}$. Le nombre représenté correspond à $\left(1 + \frac{16}{64}\right) 2^{1-8} = 1,25 \times 2^{-7}$. Attention, il y a une petite subtilité dans cet exemple car bien que les arguments soient en représentation dénormalisée, le résultat est normalisé.

Q44) $x =$`1010 110000`; $y =$`1000 000100`

Alignement	x	1010	(1)110000
	y	1010	(0)01000100
Somme			(10)00000100
Normalisation		1011	(1)000000100
Arrondi		1011	(1)000000

$x = 7$ et $y = 1,0625$. Le nombre représenté correspond à $\left(1 + \frac{0}{64}\right) 2^{11-8} = 8$. L'erreur absolue vaut $\Delta_{x+y} = 0,0625$ tandis que l'erreur relative vaut $\epsilon_{x+y} \approx 0,0078$.

Q45) $x =$`1010 110000`; $y =$`1000 000110`

Alignement	x	1010	(1)110000
	y	1010	(0)01000110
Somme			(10)00000110
Normalisation		1011	(1)000000110
Arrondi		1011	(1)000001

$x = 7$ et $y = 1,09375$. Le nombre représenté correspond à $\left(1 + \frac{1}{64}\right) 2^{11-8} = 8,125$. L'erreur absolue vaut $\Delta_{x+y} = 0,03125$ tandis que l'erreur relative vaur $\epsilon_{x+y} \approx 0,0039$.

2.9 Soustraction

En utilisant le même système qu'à la Section 2.8, effectuez la soustraction des nombres dont les représentations sont données ci-dessous.

Q46) $x =$`1011 101010`; $y =$`1011 101010`

Alignement	x	1011	(1)101010
	y	1011	(1)101010
Soustraction		0000	(0)000000 *(passage en dénormalisé)*

$x = y = \left(1 + \frac{42}{64}\right) 2^{11-8} = 13,25$. $x - y = 0$, ce qui est bien le nombre représenté par `0 0000 000000`.

Q47) $x =$`1001 010000`; $y =$`1000 100000`

Alignement	x	1001	(1)010000
	y	1001	(0)1100000
Soustraction			(0)1000000
Normalisation		1000	(1)000000
Arrondi		1000	(1)000000

$x = 2,5$ et $y = 1,5$. $x - y = 1$, ce qui est bien le nombre représenté : $\left(1 + \frac{0}{64}\right) 2^{8-8} = 1$.