# T.D. 2 – Corrigé Systèmes de numération flottante

# **Exercice 1**

Donnez la représentation flottante, simple précision, des nombres suivants :

- 1. 128
  - S = 0
  - $|128| = 128 = 1000\ 0000_2$
  - $128 = (1,0)_2.2^7$

$$M = 00...0_2$$
 et e = 7

• 
$$E = e + biais = 7 + 127 = 6 + 128$$

$$E = 1000 \ 0110_2$$

- 2. -32,75
  - S=1

• 
$$0,75 \times 2 = 1,5$$

$$0.5 \times 2 = 1$$

$$|-32,75| = 32,75 = 10\ 0000,11_2$$

•  $32,75 = (1,0000011)_2.2^5$ 

$$M = 00000110...0_2$$
 et  $e = 5$ 

• 
$$E = e + biais = 5 + 127 = 4 + 128$$

$$E = 1000 \ 0100_2$$

- 3. 18,125
  - S = 0
  - $0.125 \times 2 = 0.25$

$$0.25 \times 2 = 0.5$$

$$0.5 \times 2 = 1$$

$$|18,125| = 18,125 = 10010,001_2$$

•  $18,125 = (1,0010001)_2.2^4$ 

$$M = 00100010...0_2$$
 et  $e = 4$ 

• E = e + biais = 4 + 127 = 3 + 128

 $E = 1000 \ 0011_2$ 

- 4. 0,0625
  - S = 0

• 
$$0.0625 \times 2 = 0.125$$

$$0,125 \times 2 = 0,25$$

$$0.25 \times 2 = 0.5$$

$$0.5 \times 2 = 1$$

$$|0,0625| = 0,0625 = 0,0001_2$$

•  $0.0625 = (1.0)_2.2^{-4}$ 

$$M = 00...0_2$$
 et e = -4

• 
$$E = e + biais = -4 + 127$$

$$E = 0111 \ 1011_2$$

## **Exercice 2**

Donnez la représentation flottante, double précision, des nombres suivants :

- 1. 1
  - S = 0
  - $|1| = 1 = 1_2$
  - $1 = (1,0)_2.2^0$

$$M = 00...0_2$$
 et  $e = 0$ 

• 
$$E = e + biais = 0 + 1023$$

$$E = 011 \ 1111 \ 1111_2$$

- $1 \rightarrow 0 \ 011111111111 \ 00.....0$
- 2. -64
  - S = 1
  - $|-64| = 64 = 100\ 0000_2$
  - $64 = (1,0)_2.2^6$

$$M = 00...0_2$$
 et e = 6

• 
$$E = e + biais = 6 + 1023 = 5 + 1024$$

$$E = 100\ 0000\ 0101_2$$

•  $-64 \rightarrow 1\ 10000000101\ 00.....0$ 

#### 3. 12,06640625

```
• S = 0
```

• 0, 
$$06640625 \times 2 = 0,1328125$$
  
 $0,1328125 \times 2 = 0,265625$   
 $0,265625 \times 2 = 0,53125$   
 $0,53125 \times 2 = 1,0625$   
 $0,0625 \times 2 = 0,125$   
 $0,125 \times 2 = 0,25$   
 $0,25 \times 2 = 0,5$   
 $0,5 \times 2 = 1$ 

 $|12,06640625| = 12,06640625 = 1100,00010001_2$ 

•  $12,06640625 = (1,10000010001)_2.2^3$ 

$$M = 100000100010...0_2$$
 et  $e = 3$ 

• E = e + biais = 3 + 1023 = 2 + 1024

 $E = 100\ 0000\ 0010_2$ 

- $12,06640625 \rightarrow 0\ 10000000010\ 100000100010.....0$
- 4. 0,2734375

• 
$$S = 0$$

• 
$$0,2734375 \times 2 = 0,546875$$
  
 $0,546875 \times 2 = 1,09375$   
 $0,09375 \times 2 = 0,1875$   
 $0,1875 \times 2 = 0,375$   
 $0,375 \times 2 = 0,75$   
 $0,75 \times 2 = 1,5$   
 $0,5 \times 2 = 1$   
 $|0,2734375| = 0,2734375 = 0,0100011_2$ 

0.2724275 — (1.00011) 2-2

•  $0.2734375 = (1.00011)_2.2^{-2}$ **M** = **000110...0**<sub>2</sub> et e = -2

• 
$$E = e + biais = -2 + 1023$$

 $E = 011 \ 1111 \ 1101_2$ 

•  $0,2734375 \rightarrow 0.01111111101.000110.....0$ 

T.D. 2 – Corrigé 3/11

Donnez la représentation décimale des nombres codés en simple précision suivants :

- - $S = 1 \rightarrow n\acute{e}gatif$
  - $e = E biais = 0111 \ 1010_2 127$

$$e = 122 - 127$$

$$e = -5$$

- $\mathbf{m} = (1, \mathbf{M})_2 = (1, 1)_2$
- $-m.2^e = -(1,1)_2.2^{-5}$
- =  $-(11)_2.2^{-6}$

$$= -3.2^{-6} = -0.046875$$

- $2. \quad 0101 \quad 0101 \quad 0110 \quad 0000 \quad 0000 \quad 0000 \quad 0000 \quad 0000_2$ 
  - $S = 0 \rightarrow positif$
  - $e = E biais = 1010 \ 1010_2 127$

$$e = 170 - 127$$

$$e = 43$$

• 
$$\mathbf{m} = (1, M)_2 = (1, 11)_2$$

• 
$$+\text{m.}2^{\text{e}} = +(1,11)_2.2^{43}$$

$$\bullet = +(111)_2.2^{41}$$

$$=+7.2^{41}\approx+1.5393.10^{13}$$

- - $S = 1 \rightarrow négatif$
  - $e = E biais = 1000 \ 0011_2 127$

$$e = 131 - 127$$

$$e = 4$$

• 
$$\mathbf{m} = (1, M)_2 = (1, 111)_2$$

• 
$$-m.2^e = -(1,111)_2.2^4$$

• = 
$$-(11110)_2.2^0$$

$$= -30$$

- - $S = 1 \rightarrow négatif$
  - $E = 1111 \ 1111_2 \rightarrow \infty$
  - **-**∞

T.D. 2 – Corrigé

- 5. 0000 0000 0100 0000 0000 0000 0000  $0000_2$ 
  - $S = 0 \rightarrow positif$
  - $E = 0 \rightarrow$  représentation dénormalisée
  - $m = (0,M)_2 = (0,1)_2$
  - $+\text{m.2}^{1-\text{biais}} = +(0,1)_2.2^{-126}$
  - =  $+(1)_2.2^{-127}$ =  $+2^{-127} \approx +5.877.10^{-39}$

Donnez la représentation décimale des nombres codés en double précision suivants :

- 1. 403D 4800 0000 0000<sub>16</sub>
  - = 0100 0000 0011 1101 0100 1000 0000.....0
  - $S = 0 \rightarrow positif$
  - $e = E biais = 100\ 0000\ 0011_2 1023 = 1027 1023$ 
    - e = 4
  - $\mathbf{m} = (1, M)_2 = (1, 1101010101)_2$
  - $+m.2^e = +(1,110101001)_2.2^4$
  - =  $+(11101,01001)_2 = 29 + 2^{-2} + 2^{-5} = 29 + 0,25 + 0,03125$ = +29,28125

2. C040 0000 0000 0000<sub>16</sub>

- = 1100 0000 0100 0000.....0
- $S = 1 \rightarrow négatif$
- $e = E biais = 100\ 0000\ 0100_2 1023 = 1028 1023$

$$e = 5$$

- $\mathbf{m} = (1, \mathbf{M})_2 = (1, \mathbf{0})_2$
- $-m.2^e = -(1,0)_2.2^5$
- $= -2^5 = -32$
- 3. BFC0 0000 0000 0000<sub>16</sub>
  - = 1011 1111 1100 0000.....0
  - $S = 1 \rightarrow n\acute{e}gatif$
  - $e = E biais = 011 \ 1111 \ 1100_2 1023 = 1020 1023$

$$e = -3$$

- $\mathbf{m} = (1, \mathbf{M})_2 = (1, \mathbf{0})_2$
- $-m.2^e = -(1.0)_2.2^{-3}$
- $-2^{-3} = -0.125$

T.D. 2 – Corrigé

- $4. \quad 8000 \quad 0000 \quad 0000 \quad 0000_{16}$ 
  - = 1000 0000 0000 0000.....0
  - E = 0 et  $M = 0 \rightarrow 0$
- 5. FFF0 0001 0000 0000<sub>16</sub>
  - = 1111 1111 1111 0000 0000 0000 0001 0000.....0
  - $E = 111 \ 1111 \ 1111_2 \ \text{et } M \neq 0 \rightarrow \mathbf{NaN}$

Pour chaque question, vous traiterez le cas des codages **simples et doubles précisions** du format à mantisse normalisée.

1. Déterminez, en valeur absolue, le plus petit et le plus grand nombre flottant.

## · Simple précision

# • Minimum

$$\begin{split} & Min_{simple} = m_{min}.2^{emin} \\ & m_{min} = (1,0)_2 = 1 \\ & e_{min} = E_{min} - biais \qquad avec \ E_{min} = 1 \\ & e_{min} = 1 - 127 = -126 \\ & \textbf{Min}_{simple} = 2^{-126} \approx 1,1755.10^{-38} \end{split}$$

#### Maximum

$$\begin{split} &Max_{simple} = m_{max}.2^{emax} \\ &m_{max} = (1, M_{max})_2 = 1 + (0, M_{max})_2 = 1 + M_{max}.2^{-23} \ avec \ M_{max} = 2^{23} - 1 \\ &m_{max} = 1 + (2^{23} - 1).2^{-23} = 1 + 1 - 2^{-23} = 2 - 2^{-23} = 2.(1 - 2^{-24}) \\ &e_{max} = E_{max} - biais \qquad avec \ E_{max} = (2^8 - 1) - 1 = 254 \\ &e_{max} = 254 - 127 = 127 \\ &Max_{simple} = 2.(1 - 2^{-24}).2^{127} \\ &Max_{simple} = (1 - 2^{-24}).2^{128} \approx 3,4028.10^{38} \end{split}$$

### · Double précision

#### Minimum

$$\begin{split} & Min_{double} = m_{min}.2^{emin} \\ & m_{min} = (1,0)_2 = 1 \\ & e_{min} = E_{min} - biais \qquad avec \ E_{min} = 1 \\ & e_{min} = 1 - 1023 = -1022 \\ & \textbf{Min}_{\textbf{double}} = \textbf{2}^{-1022} \approx \textbf{2,2251.10}^{-308} \end{split}$$

T.D. 2 – Corrigé 6/11

#### Maximum

$$\begin{split} &Max_{double} = m_{max}.2^{emax} \\ &m_{max} = (1, M_{max})_2 = 1 + (0, M_{max})_2 = 1 + M_{max}.2^{-52} \ avec \ M_{max} = 2^{52} - 1 \\ &m_{max} = 1 + (2^{52} - 1).2^{-52} = 1 + 1 - 2^{-52} = 2 - 2^{-52} = 2.(1 - 2^{-53}) \\ &e_{max} = E_{max} - biais \qquad avec \ E_{max} = (2^{11} - 1) - 1 = 2046 \\ &e_{max} = 2046 - 1023 = 1023 \\ &Max_{double} = 2.(1 - 2^{-53}).2^{1024} \approx 1,7977.10^{308} \end{split}$$

2. Quel est le plus petit nombre strictement positif qui, ajouté à 1, donne un résultat différent de 1?

# · Simple précision

On pose :  $1 = m.2^e$  avec  $m = (1,0)_2$  et e = 0. Le codage de la mantisse M contient donc 23 zéros.

Observons maintenant l'addition ci-dessous :

Le codage de la mantisse du résultat doit contenir autre chose que 23 zéros si l'on souhaite obtenir une différence avec le codage de la mantisse du nombre 1. Le plus petit nombre possible pour obtenir cette différence est donc 2<sup>-23</sup>.

## double précision

Avec un raisonnement identique à celui du codage en simple précision, on obtient 2<sup>-52</sup>.

T.D. 2 – Corrigé 7/11

Soit le programme suivant écrit en langage C :

```
#include <stdio.h>

void main()
{
    float f1, f2, f3, r;

    f1 = 1E25;
    f2 = 16;

    f3 = f1 + f2;
    r = f3 - f1;

    printf("r = %f\n", r);
}
```

**Indication**:  $10^{25} \approx 2^{83}$ 

- 1. Donnez la valeur de **r** affichée à la fin de l'exécution du programme en expliquant votre raisonnement
  - On pose :  $\mathbf{f_1} = (1, \mathbf{M_1})_2 \cdot 2^{e_1}$ Sachant que  $\mathbf{f_1} \approx 2^{83}$ , on en déduit que  $\mathbf{f_1} \approx (1, 0)_2 \cdot 2^{83}$  avec  $\mathbf{e_1} = 83$  et  $\mathbf{M_1} = 0 \dots 0$
  - On pose :  $f_3 = (1, M_3)_2.2^{e3}$
  - Observons maintenant l'addition ci-dessous :

On constate que la plus petite valeur de  $\mathbf{f_2}$  pouvant modifier la valeur de  $\mathbf{M_3}$  est la valeur  $2^{60}$ . Or dans le programme,  $\mathbf{f_2}$  possède la valeur 16: aucun changement n'apparaît donc sur le résultat de l'addition.

```
L'instruction \mathbf{f}_3 = \mathbf{f}_1 + \mathbf{f}_2 est équivalente à \mathbf{f}_3 = \mathbf{f}_1.
La soustraction \mathbf{r} = \mathbf{f}_3 - \mathbf{f}_1 devient alors \mathbf{r} = \mathbf{f}_3 - \mathbf{f}_3 = \mathbf{0}.
```

La valeur affichée à la fin de l'exécution du programme est donc 0.

T.D. 2 – Corrigé 8/11

- 2. Dans le programme  $f_1=10^{25}$ . Supposons maintenant que  $f_1=10^n$  avec **n** entier positif. Jusqu'à quelle valeur de **n**, un résultat correct apparaîtra-t-il sur **r**?
  - Pour que l'addition  $\mathbf{f_3} = \mathbf{f_1} + \mathbf{f_2}$  soit valide, il faut que la valeur 16, contenue dans  $\mathbf{f_2}$ , puisse modifier le codage de la mantisse du résultat  $\mathbf{M_3}$ . La valeur du poids le plus faible de  $\mathbf{M_1}$  doit donc être au maximum de  $2^4$ , car à partir de  $2^5$ , la valeur de  $\mathbf{f_2}$  sera trop petite pour être prise en compte.

- La variable  $\mathbf{f_1}$  doit donc être strictement inférieure à  $2^{28}$  afin que l'addition avec la variable  $\mathbf{f_2}$  puisse entraı̂ner un changement dans le codage de la mantisse  $\mathbf{M_3}$ .
- Ce qui donne:

$$\begin{aligned} &f_1 < 2^{28} \\ &10^n < 2^{28} \\ &n < Log(2^{28}) \\ &n < 8,42 \end{aligned}$$

$$n_{max} = 8$$

3. Même question si les variables  $f_1$ ,  $f_2$ ,  $f_3$  et r sont déclarées en double précision.

Avec un raisonnement identique à celui du codage en simple précision, on obtient :

$$\begin{split} &f_1 < 2^{5+52} \\ &10^n < 2^{57} \\ &n < Log(2^{57}) \\ &n < 17,15 \end{split}$$

$$n_{\text{max}} = 17$$

Il ne faut jamais sous-estimer les risques d'erreur liés à la manipulation de variables entières ou flottantes. Des débordements ou des problèmes de précision peuvent survenir à tout moment et déjouer la vigilance de n'importe quel développeur, même expérimenté.

T.D. 2 – Corrigé 9/11

## À propos de la fusée Ariane

Voici un exemple célèbre d'erreur de programmation minime aux conséquences énormes. Lors de son tout premier vol le 4 juin 1996, la fusée Ariane 5 a explosé quarante secondes seulement après son décollage de la base de Kourou en Guyane. La perte financière fut estimée à environ 500 millions de dollars. Le CNES (Centre National d'Études Spatiales) et l'ESA (European Space Agency) ont immédiatement lancé une enquête. Un comité d'experts internationaux fut réuni et un rapport sans équivoque fut livré un mois plus tard : l'explosion était due à une erreur de logiciel.

En cause, les SRI ou Systèmes de Référence Inertiels. Cette partie du logiciel, qui provenait du lanceur Ariane 4, n'avait pas été adaptée à la plus grande vitesse horizontale d'Ariane 5. Du coup, lors d'une conversion d'un nombre flottant sur 64 bits, contenant la vitesse horizontale, en un entier sur 16 bits, l'opération a provoqué un débordement et une exception a été générée. Malheureusement, aucune routine de traitement de cette exception n'ayant été prévue, c'est le traitement d'exception par défaut qui fut exécuté et le programme tout entier termina son exécution.

Depuis, les concepteurs du logiciel d'Ariane ont mis en place un plan de programmation défensive qui est reconnu comme une référence en la matière.

Jean-Christophe Arnulfo, Métier Développeur, Paris, Dunod, 2003

### Exercice 7

Sachant que votre compilateur C utilise la norme IEEE 754 pour la gestion des flottants, donnez une fonction en langage C, **de quelques lignes seulement**, permettant de visualiser sous forme hexadécimale la représentation IEEE 754 d'un nombre flottant, simple précision, passé en paramètre.

Le principe de base consiste à trouver l'emplacement où le nombre flottant est stocké en mémoire. Il faut ensuite accéder à cette valeur, en la considérant comme un nombre entier, et l'afficher au format hexadécimal.

Ceci peut être fait soit à l'aide de pointeurs, soit à l'aide du mot clé union :

#### • Solution à l'aide de pointeurs :

```
void Convert(float fv)
{
    // Déclare un pointeur sur un entier 32 bits.
    long* pl;

    // Convertit le pointeur flottant vers un pointeur entier.
    // Le pointeur entier pointe la même adresse mémoire que le pointeur flottant.
    pl = (long*)(&fv);

    // Affiche le résultat au format hexadécimal sur 8 chiffres.
    printf("Code hexadecimal IEEE 754 de %f : %08x\n", fv, *pl);
}
```

T.D. 2 – Corrigé

# · Solution à l'aide du mot clé union :

```
void Convert(float fv)
{
    // Déclare un flottant et un entier dans le même espace mémoire.
    union
    {
        float f;
        long l;
    };

    // Copie la valeur à convertir dans le flottant de l'union.
    f = fv;

    // Affiche le résultat au format hexadécimal sur 8 chiffres.
    printf("Code hexadecimal IEEE 754 de %f : %08x\n", f, l);
}
```

T.D. 2 – Corrigé 11/11