REPRESENTATION D'UN NOMBRE REEL NOTATION A VIRGULE FLOTTANTE

1 – REPRESENTATION A VIRGULE FIXE

La première représentation est la notation à virgule fixe. La position de la virgule est décidée arbitrairement et elle est fixe.

1.1 – Conversion Binaire Décimal

Prenons par exemple le nombre binaire suivant : **$(1011,1101)_2$** constitué de 4 bits pour la partie « Entiers » et de 4 bits pour la partie « Fractions ».

1	**0**	**1**	**1**	**,**	**1**	**0**	**0**	**1**
I3	I2	I1	I0		F1	F2	F3	F4
Entiers					Fractions			

<u>Conversion de la partie « Entiers »</u> :

Pour la connaître la valeur décimale de la partie « Entiers », il faut appliquer la méthode vue pour la conversion binaire décimale d'un entier : $(1011)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 11$

<u>Conversion de la partie « Fractions »</u> :

Pour la connaître la valeur décimale de la partie « Fractions », il faut appliquer la même méthode mais en utilisant l'inverse des puissances de 2 :

$$(1001)_2 = 1 \times \frac{1}{2^1} + 0 \times \frac{1}{2^2} + 0 \times \frac{1}{2^3} + 1 \times \frac{1}{2^4} = 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4} = 0{,}5 + 0{,}0625 = 0{,}5625$$

Le nombre binaire **$(1011,1101)_2$** est égal à **11,5625**.

1.2 – Conversion Décimal Binaire

Prenons par exemple le nombre décimal suivant : **6,625**.

<u>Conversion de la partie « Entiers »</u> :

Poids		$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
Nombre binaire		**0**	**1**	**1**	**0**
Reste	6	6	2	0	0

$6 = (0110)_2$

Conversion de la partie « Fractions » :

Poids		$2^{-1} = 0,5$	$2^{-2} = 0,25$	$2^{-3} = 0,125$	$2^{-4} = 0,0625$
Nombre binaire		1	0	1	0
Reste	0,625	0,125	0,125	0	0

$0,625 = (1010)_2$

Le nombre **6,625** est égal au nombre binaire $\mathbf{(0110,1010)_2}$.

Prenons un autre exemple, le nombre décimal suivant : **14,27**.

Conversion de la partie « Entiers » :

Poids		$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
Nombre binaire		1	1	1	0
Reste	14	6	2	0	0

$14 = (1110)_2$

Conversion de la partie « Fractions » :

Poids		$2^{-1} = 0,5$	$2^{-2} = 0,25$	$2^{-3} = 0,125$	$2^{-4} = 0,0625$
Nombre binaire		1	0	0	0
Reste	0,27	0,27	0,02	0,02	0,02

$0,27 \Rightarrow (1000)_2$

Le nombre binaire $\mathbf{(1110,1000)_2}$.est une **approximation** du nombre décimal 14,27. Il est possible de s'approcher de 14,27 (sans jamais l'atteindre) en ajoutant un nombre un nombre plus important de bits après la virgule. La représentation la plus proche est **14,26999999999999995736743585439**.

Le nombre de bits après la virgule n'étant pas infini, il n'est possible de **représenter qu'un nombre fini de réels de manière exacte**, et pour tous les autres la représentation binaire sera une **valeur approchée**.

2 – REPRESENTATION A VIRGULE FLOTTANTE OU NORME IEEE-754

La représentation à virgule fixe présente des inconvénients :

- L'espace réservé à la partie « fractions » 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.

En informatique, pour représenter les nombres à virgule, on utilise une représentation similaire à la «notation scientifique» des calculatrices, sauf qu'elle est en base deux et non en base dix. Il s'agit de la représentation **en virgule flottante**, d'où le type **float**. Un tel nombre est représenté sous la forme suivante :

$Sm \times 2^n$	**S** : **Signe** du nombre **m** : **Mantisse** du nombre comprise ($1 \leq m < 2$) **n** : **Exposant** du nombre (entier relatif)

Les nombres à virgule flottante doivent respecter une forme normalisée, afin que la représentation ne varie pas d'un matériel à l'autre. La norme standard est la norme **IEEE 754**. Elle définit **4 formats** pour représenter des nombres à virgule flottante : **simple précision** (**32 bits**), **simple précision étendue** (**43 bits min**), la **double précision** (**64 bits**) et la **double précision étendue** (**80 bits**). Le format le plus courant est le format double précision :

b_{63}	b_{62}	b_{61}	...	b_{53}	b_{52}	b_{51}	b_{50}	b_{49}	...	b_2	b_1	b_0
Signe	**Exposant décalé**					**Mantisse**						

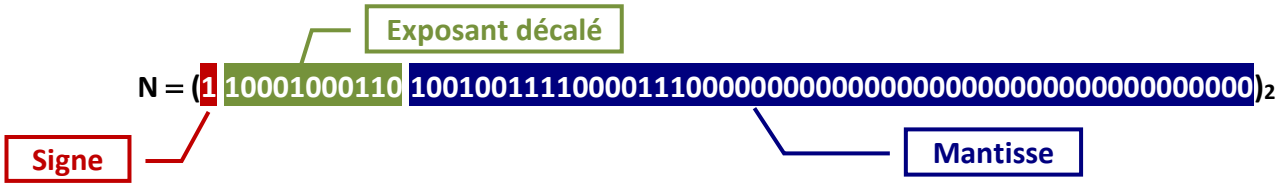
Elément	Nb de bits	Description
Signe	**1**	**0** : Nombre positif ; **1** : Nombre négatif
Exposant décalé	**11**	L'exposant n est un entier relatif compris entre **–1 022 et 1 023**. Or il doit être représenté comme un entier naturel. Pour cela on utilise un **décalage de 1023**. L'**exposant décalé** possède donc une valeur **comprise entre 1 et 2046**. Les deux entiers **0** et **2 047** sont réservés pour certaines situations : **NaN** = Not a Number ; +∞, –∞.
Mantisse	**52**	Il s'agit d'un **nombre binaire à virgule compris entre 1 inclus et 2 exclu** comprenant **52 chiffres après la virgule**. Il y a toujours un seul chiffre avant la virgule et ce chiffre est toujours un 1 ; il est donc inutile de le représenter. Les **52 bits** pour **représenter les 52 chiffres après la virgule**.

2.1 – Conversion Binaire Décimal

Soit le nombre flottant double précision :

$(C4693C3800000000)_{16} = (1100010001101001001111000011100000000000000000000000000000000000\ 0000)_2$

Il peut être décomposé selon la norme IEEE-754 :



1. Déterminer le signe

S = 1 : le nombre est **négatif**.

2. Déterminer l'exposant

Exposant décalé = $(10001000110)_2$ = **1094**, donc l'exposant **n** = 1094 – 1023 = **71**

3. Déterminer la mantisse

La valeur de matisse est égale à 1 virgule suivie des 52 bits qui expriment la mantisse :

1	**,**	b_{51}	b_{50}	b_{49}	...	b_2	b_1	b_0

Mantisse = $(\mathbf{1},1001001111000011100000000000000000000000000000000000)_2$

= $2^0+2^{-1}+2^{-4}+2^{-7}+2^{-8}+2^{-9}+2^{-10}+2^{-15}+2^{-16}+2^{-17} = (2^{17}+2^{16}+2^{13}+2^{10}+2^9+2^8+2^7+2^2+2^1+1)/2^{17}$

= 206 727/131 072 = **1,57720184326171875**

4. Calculer le nombre décimal

N = Sm×2^n =–1,57720.2^{71} = – 3,724062560669682106368 × 10^{21}

$\boxed{\mathbf{(C4693C3800000000)_{16} = -\ 3{,}724062560669682106368 \times 10^{21}}}$

2.2 – Conversion Décimal Binaire

Soit le nombre réel : **17,765625**.

1. Déterminer le signe

Le nombre est **positif** : **S = 0**.

2. Déterminer le nombre selon la notation de la virgule fixe

17,765625 = $(10001{,}110001)_2$

3. Décaler la virgule

$(10001{,}110001)_2 \Rightarrow (1{,}0001110001)_2$

4. Déterminer la mantisse m

La mantisse m est constituée des bits de la partie « fractions » du nombre précédent auxquels il faut ajouter des 0 pour compléter les 52 bits de la matisse.

m = (0001 1100 0100 0000 000 0000 0000 0000 0000 0000 0000 0000 0000 0000)$_2$

5. Déterminer l'exposant décalé

A chaque décalage de la virgule d'une position vers la gauche l'exposant augmente de 1 et à décalage de la virgule d'une position vers la droite l'exposant diminue de 1 :

(10001,110001)$_2$ = (1,0001110001 × 2^4)$_2$

L'exposant **n = 4.** L'exposant décalé a valeur 4 + 1023 = **1027 = (010 0000 0011)$_2$**

6. Déterminer le nombre binaire selon la norme IEEE-754

Exposant décalé

N = (0 10000000011 0001110001000000000000000000000000000000000000000000000)$_2$

Signe

Mantisse

N = (0100 0000 0011 0001 1100 0100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000)$_2$

= **(4031C40000000000)$_{16}$**

17,765625 = (4031C40000000000)$_{16}$

Remarque : Dans le cas des nombres flottant, le nombre de bits de matisse n'étant pas infini, il n'est possible de **représenter qu'un nombre fini de réels de manière exacte**, et pour tous les autres la représentation binaire sera une **valeur approchée**.

Cette approximation peut entrainer des résultats surprenants :

```
ex1 = (1.1 + 2.2) - 3.3
```

```
print(ex1)
```

```
4.440892098500626e-16
```

```
ex2 = (3.1 * 2.3) * 1.1
ex3 = 3.1 * (2.3 * 1.1)
```

```
print(ex2, ex3)
```

```
7.843000000000001 7.843
```

```
0.1 + 0.2 == 0.3
```

```
False
```

Les flottants sont **à utiliser avec précautions** et uniquement s'il est nécessaire de représenter des nombres réels. Dans ce cas il ne faut **jamais utiliser l'opération d'égalité (==) pour comparer deux réels**.