V. Encodage des nombres réels

V. 1. Écriture scientifique approximatived'un nombre

Définition 5 (Écriture scientifique d'un nombre réel en base b)

Soit $M \in \mathbb{N}$, $E \in \mathbb{N}$ et $b \in \mathbb{N}$ avec $b \geq 2$.
L'écriture scientifique à M symboles significatifs d'un nombre x en base b est l'écriture **approximative** de ce nombre comme le produit d'un nombre à virgule n'ayant qu'un seul symbole différent de 0 devant la virgule par une puissance de b :

$$x \approx \pm\, m_0.m_1m_2m_3 \ldots m_M \times b^{e_{E-1}e_{E-2}\ldots e_0}$$

où $m_0 \in S - s_0$ et

$$m_0.m_1m_2m_3 \ldots m_M = m_0 + m_1 \times b^{-1} + m_2 \times b^{-2} + \ldots + m_M \times b^{-M}$$

Propriété 6

L'écriture scientifique à M symboles significatifs d'un nombre dans une base b donnée est unique.

Exemples 11

1. L'écriture scientifique de 145.78 à 4 symboles significatifs en base 10 est 1.4578×10^2
2. L'écriture scientifique de 145.78 à 2 symboles significatifs en base 10 est 1.45×10^2
3. L'écriture scientifique de $\frac{1}{3}$ à 3 chiffres significatifs en base 10 est 3.333×10^{-1}

V. 2. Représentation en machine : la norme IEEE 754

V. 2. a. Principe

Pour encoder une information numérique, nous ne disposons que de deux symboles : 0 et 1. On utilise donc toujours la base 2 qui possède exactement deux symboles.
Comme le symbole m_0 devant la virgule doit être différent de 0 par définition de l'écriture scientifique, en base 2, c'est forcément 1 ! On sait que ce sera 1, donc il n'est même pas nécessaire de stocker m_0.

Définition 6 (Écriture scientifique d'un nombre réel en base 2)

L'écriture scientifique à M symboles significatifs d'un nombre x en base 2 est :

$$x = \pm\, 1.m_1m_2m_3 \ldots m_M \times b^{e_{E-1}e_{E-2}\ldots e_0}$$

où les m_i et les e_i sont des bits donc des symboles 0 ou 1.

Remarque. Pour la plupart des nombres réels, cette écriture est approximative.
L'écriture scientifique en base 2 est aussi parfois appelée écriture en **virgule flottante**.

V. 2. b. Description de la norme

La représentation des nombres flottants et les opérations arithmétiques qui les accompagnent ont été définies dans la norme internationale IEEE 754. C'est la norme la plus couramment utilisée dans les ordinateurs de nos jours. Selon cette norme, les nombres réels sont encodés de la manière suivante :

— 1 bit de signe

— suivi de E bits $e_{E-1} \dots e0$ pour encoder l'exposant biaisé

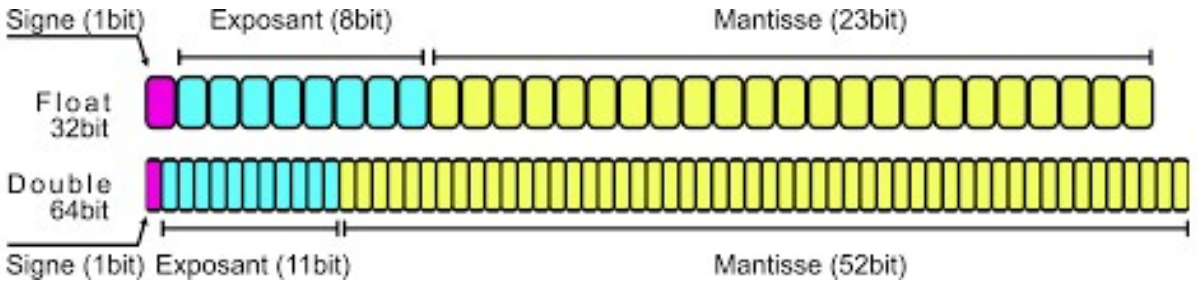
— suivis de M bits $m_1 m_2 \dots m_M$ de mantisse

Il existe deux modalités d'encodage des nombres réels :

Simple précision : sur $N = 32$ bits avec 1 bit de signe + $E = 8$ bits d'exposant biaisé + $M = 23$ bits de mantisse. Le type correspondant est noté `float` en Python, C et OCaml ;

Double précision : sur $N = 64$ bits avec 1 bit de signe + $E = 11$ bits d'exposant biaisé + $M = 52$ bits de mantisse. Le type correspondant est le type noté `float` en langage OCaml et Python et `double` en langage C.

Il existe également un type `long double` en langage C qui correspond à un encodage sur 128 bits d'un nombre réel.



V. 2. c. Mantisse

Définition 7 (Mantisse)

La série de 0 et de 1 après la virgules $m_1 m_2 \dots m_M$ est appelée **mantisse**.
Sa valeur est :

$$\text{mantisse} = 1 + m_1 \times 2^{-1} + m_2 \times 2^{-2} + \dots + m_M \times 2^{-M}$$

Exemple 12

La mantisse de $\frac{1}{3}$ à 12 chiffres significatifs en base 2 se trouve de la manière suivante.

$$\begin{aligned}\tfrac{1}{3} &= \tfrac{4}{3} \times \tfrac{1}{4} = \tfrac{4}{3} \times 2^{-2} \\ &\approx (1 + 2^{-2} + 2^{-4} + 2^{-6} + 2^{-8} + 2^{-10} + 2^{-12}) \times 2^{-2} \\ &\approx (1 + 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + 0 \times 2^{-5} + 1 \times 2^{-6} + 0 \times 2^{-7} + 1 \times 2^{-8} \\ &\quad + 0 \times 2^{-9} + 1 \times 2^{-10} + 0 \times 2^{-11} + 1 \times 2^{-12}) \times 2^{-2}\end{aligned}$$

Donc la mantisse à 12 chiffres significatifs en base 2 de ce nombre est :

$$m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8 m_9 m_{10} m_{11} m_{12} = 010101010101$$

et il resterait normalement à écrire l'exposant -2 en base 2 mais nous verrons juste après qu'en pratique, on encode un exposant *biaisé*.

V. 2. d. Exposant et exposant biaisé

L'exposant peut être positif ou négatif. Cependant, la représentation habituelle des nombres signés (complément à 2) rendrait la comparaison entre les nombres flottants un peu plus coûteuse. Pour régler ce problème, dans la norme IEEE754, l'exposant est *biaisé* (décalé), afin de le stocker sous forme d'un nombre non signé.
Ce biais est de $2^{E-1} - 1$ (E représente le nombre de bits utilisés pour l'exposant) il s'agit donc d'une valeur constante une fois que le nombre de bits d'exposant E est fixé.

Définition 8 (Exposant biaisé)

$$\text{exposant\_biaisé} = \text{exposant} + 2^{E-1} - 1$$

C'est l'exposant biaisé qui est encodé sur la série de bits d'exposant : $e_{E-1} e_{E-2} \dots e_0$.
Sa valeur est :

$$\text{exposant\_biaisé} = e_{E-1} \times 2^{E-1} + \dots + e_2 \times 2^2 + e_1 \times 2 + e_0$$

L'exposant biaisé doit être positif ou nul. De plus, étant codé sur E bits, sa valeur maximale est $2^E - 1$. Ainsi

$$\begin{aligned}& 0 \leq \text{exposant\_biaisé} \leq 2^E - 1 \\ \Leftrightarrow\ & 0 \leq \text{exposant} + 2^{E-1} - 1 \leq 2^E - 1 \\ \Leftrightarrow\ & -2^{E-1} + 1 \leq \text{exposant} \leq 2^E \cancel{-1} - 2^{E-1} \cancel{+1} \\ \Leftrightarrow\ & -2^{E-1} + 1 \leq \text{exposant} \leq 2^{E-1} \times (2 - 1) \\ \Leftrightarrow\ & -2^{E-1} + 1 \leq \text{exposant} \leq 2^{E-1}\end{aligned}$$

Les deux valeurs extrêmes d'exposant sont utilisées pour des cas particuliers (valeurs infinies, division par 0, nombres extrêmement petits...), cf Section V. 5..

En dehors de ces cas particuliers, on parle de nombres normalisés.

Propriété 7

Pour les nombres normalisés :

$$\text{L'exposant biaisé appartient à l'intervalle } [\![1;\ 2^E - 2]\!]$$

$$\Leftrightarrow \quad \text{L'exposant appartient à l'intervalle } \in [\![-2^{E-1} + 2;\ 2^{E-1} - 1]\!]$$

Propriété 8 (Simple précision)

En simple précision, la série de 32 bits :

$$(s\ e_7 e_6 \dots e_1 e_0\ m_1 m_2 \dots m_{23})_2$$

correspond à la valeur :

$$(-1)^s \times \left(1 + m_1 \times 2^{-1} + m_2 \times 2^{-2} + \dots + m_{23} \times 2^{-23}\right) \times 2^{\left(e_7 \times 2^7 + \dots + e_2 \times 2^2 + e_1 \times 2 + e_0\right) - 127}$$

Exemple 13

Le mot de 32 bits suivant :

$$1\ 10000110\ 10101101100000000000000$$

correspond à :

signe : $(-1)^1 = -1$

exposant : $(2^7 + 2^2 + 2^1) - 127 = (128 + 4 + 2) - 127 = 7$

mantisse : $1 + 2^{-1} + 2^{-3} + 2^{-5} + 2^{-6} + 2^{-8} + 2^{-9} = 1.677734375$

Soit, au final, le nombre décimal -1.677734375×2^7.

Propriété 9 (Double précision)

En double précision, la série de 64 bits :

$$(s\ e_{11} e_{l10} \dots e_1 e_0\ m_1 m_2 \dots m_{52})_2$$

correspond à la valeur :

$$(-1)^s \times \left(1 + m_1 \times 2^{-1} + m_2 \times 2^{-2} + \dots + m_{52} \times 2^{-52}\right) \times 2^{\left(e_{11} \times 2^{11} + \dots + e_2 \times 2^2 + e_1 \times 2 + e_0\right) - 1023}$$

V. 3. De la valeur décimale à l'encodage binaire

1. On commence pour chercher l'écriture scientifique du nombre en base 2 à M chiffres significatifs, M étant donné par le contexte
2. On calcule l'encodage binaire de l'exposant biaisé
   - On part de l'exposant (positif ou négatif) de l'écriture scientifique
   - On lui ajoute $2^{E-1}-1$
   - On encode la valeur obtenue, qui doit être un entier positif, en binaire sur E bits
3. On encode directement la mantisse (bits après la virgule)

Exemple 14

Terminons l'exemple 12 pour décrire complètement l'encodage binaire de $\frac{1}{3}$ sur $N = 32bits$.

Signe Le signe est positif donc le bit de poids fort est à 0

Exposant biaisé L'exposant vaut $(-2)_{10}$. Si l'on effectue un encodage de l'exposant sur $E = 8$ bits, l'exposant biaisé vaut :

$$-2+\underbrace{2^7-1}_{\text{biais}} = 128-3 = (10000000)_2-(00000011)_2 = (10000000)_2+(11111101)_2 = (01111101)_2$$

Mantisse La mantisse vaut :

$$1+2^{-2}+2^{-4}+2^{-6}+2^{-8}+2^{-10}+2^{-12}+\dots$$

Son encodage sur $M = 23$ bits commence donc par $(010101010101.....)_2$.

L'encodage binaire sera donc, sur $N = 32$ bits :

$$(\underbrace{0}_{\text{bit de signe}}\ \underbrace{01111101}_{\text{exposant biaisé}}\ \underbrace{010101010101.....}_{\text{mantisse}})_2$$

En hexadécimal, cette série de 32 bits s'écrit :

$$(\underbrace{0011}_{3}\ \underbrace{1110}_{14\to E}\ \underbrace{1010}_{10\to A}\ \underbrace{1010}_{A}\ \underbrace{1010}_{A}\ 1...\ ....\ ....\ )_2 = (3EAAA...)_{16}$$

V. 4. Quelques valeurs limites

V. 4. a. Plus grand nombre normalisé représentable.

Avec cet encodage, le plus grand nombre normalisé représentable est :

$$x_{max} \approx 2 \times 2^{2^{E-1}-1} \approx 2^{2^{E-1}}$$

Pour trouver l'écriture scientifique en base 10 de ce nombre, on cherche $n \in \mathbb{N}$ et $\varepsilon \in [0;1[$ tels que :

$$10^{n+\varepsilon} = 2^{2^{E-1}}$$

Ce qui donne :

$$\begin{aligned} & 10^{n+\varepsilon} = 2^{2^{E-1}} \\ \Leftrightarrow \quad & (n+\varepsilon)\ln(10) = 2^{E-1}\ln(2) \\ \Leftrightarrow \quad & n+\varepsilon = 2^{E-1}\frac{\ln(2)}{\ln(10)} \end{aligned}$$

donc n est la partie entière de $2^{E-1}\dfrac{\ln(2)}{\ln(10)}$ et ε sa partie fractionnaire.

V. 4. b. Plus petit nombre normalisé représentable.

Avec cet encodage, le plus petit nombre normalisé représentable est :

$$x_{min} = 1 \times 2^{-2^{E-1}+2} = 2^{-2^{E-1}+2}$$

Pour trouver l'écriture scientifique en base 10 de ce nombre, on cherche $n \in \mathbb{N}$ et $\varepsilon \in [0;1[$ tels que :

$$10^{n+\varepsilon} = 2^{-2^{E-1}+2}$$

et par un calcul similaire au précédent, on trouve que n est la partie entière de $2^{-2^{E-1}+2}\dfrac{\ln(2)}{\ln(10)}$ et ε sa partie fractionnaire.

V. 4. c. Précision absolue.

La variation minimale sur la mantisse est de :

$$\delta M = 2^{-M}$$

Précision absolue sur les grands nombres. La variation entre deux nombres successifs normalisés d'exposant maximal est donc de :

$$\delta x_{abs.big} = \delta M \times 2^{2^{E-1}-1} = 2^{-M} \times 2^{2^{E-1}-1} = 2^{2^{E-1}-1-M}$$

Précision absolue sur les petits nombres. La variation entre deux nombres successifs normalisés d'exposant minimal est donc de :

$$\delta x_{abs.small} = \delta M \times 2^{-2^{E-1}+2} = 2^{-M} \times 2^{-2^{E-1}+2} = 2^{-2^{E-1}+2-M}$$

V. 4. d. Précision relative normalisée.

La précision relative normalisée mesure le saut relatif entre deux nombres adjacents.

$$\delta x_{rel} = \frac{|x' - x|}{|x|} = 2^{-M}$$

La précision relative normalisée indique le nombre de chiffres en base 10 qui sont correctement représentés par l'écriture flottante de ce nombre.

V. 4. e. Simple précision

La valeur réelle maximale représentable en simple précision est donc

$$x_{max} \approx 3.4 \times 10^{38}$$

La valeur réelle minimale représentable en simple précision avec un nombre normalisé est :

$$x_{min} \approx 1.17 \times 10^{-38}$$

La précision absolue sur les grands nombres est :

$$\delta x_{abs.big} = 2^{2^{8-1}-1-23} = 2^{104} \approx 2.03 \times 10^{31}$$

Et sur les petits nombres, elle est de :

$$\delta x_{abs.small} = 2^{-2^{8-1}+2-23} = 2^{-149} \approx 1.40 \times 10^{-45}$$

La précision relative est quant à elle de :

$$\delta x_{rel} = 2^{-23} = 1.2 \times 10^{-7}$$

Les 7 premiers chiffres sont représentés correctement en simple précision (6 chiffres après la virgule).

V. 4. f. Double précision

La valeur réelle maximale représentable en double précision est :

$$x_{max} \approx 10^{0.25} \times 10^{308} \approx 1.797 \times 10^{308}$$

La valeur réelle minimale représentable en double précision avec un nombre normalisé est :

$$x_{min} \approx 2.22 \times 10^{-308}$$

La précision absolue sur les grands nombres est :

$$\delta x_{abs.big} = 2^{2^{11-1}-1-52} = 2^{1024-53} = 2^{971} \approx 1.99 \times 10^{292}$$

Et sur les petits nombres, elle est de :

$$\delta x_{abs.small} = 2^{-2^{11-1}+2-52} = 2^{-1074} \approx 4.94 \times 10^{-324}$$

La précision relative est quant à elle de :

$$\delta x_{rel} = 2^{-52} = 2.22 \times 10^{-16}$$

Les 16 premiers chiffres sont représentés correctement en double précision (15 chiffres après la virgule).
En résumé, pour les encodages simple et double précision représentés sur la Figure **??**, on a :

	Min	Max	Précision relative
`float`	`1.17e-38`	`3.40e+38`	$\approx 10^{-7}$
`double`	`2.22e-308`	`1.80e308`	$\approx 10^{-16}$

V. 5. Cas particuliers

V. 5. a. Valeurs spéciales

Infinis et `NaNs`. Si tous les bits de l'exposant biaisé sont à 1 (exposant maximal), il y 2 cas :

Représentation des infinis $+\infty$ et $-\infty$. Si tous les bits de la mantisse sont nuls, x est considéré comme étant infini. Informatiquement, ce nombre est noté `inf`.[9]

Représentation des `NaN` *Not a Number*. Si certains bits de la mantisse ne sont pas nuls, x est une valeur qui n'est pas un nombre. Plusieurs types de `Nan` peuvent être encodés selon les bits mis à 1 dans la mantisse.

Encodage du 0. Le zéro réel est un cas particulier car il n'a pas à proprement parler d'écriture scientifique. Par convention, le nombre 0 est encodé de la façon suivante :

— tous les bits de l'exposant biaisé sont à 0

— tous les bits de mantisse sont à 0

Il y a néanmoins deux représentations de 0 dans ce système : `+0` si le bit de signe est à 0 et `-0` si le bit de signe est à 1

En résumé :

Signe	Exposant biaisé	Mantisse	Nombre représenté
+1	0	0	zéro positif `+0`
-1	0	0	zéro négatif `-0`
$\pm$ 1	0	$\neq 0$	nombre dénormalisé
+1	maximal	0	infini positif $+\infty$ `+inf`
-1	maximal	0	infini négatif $-\infty$ `-inf`
$\pm$ 1	Max	$\neq 0$	valeur de type `NaN`

V. 5. b. Nombres dénormalisés.

Comme nous l'avons vu ci-dessus, si l'exposant biaisé d'un nombre flottant (sur $N = 32$ bits, donc $E = 8$ bits d'exposant) est compris entre 1 et $2^8 - 2 = 254$, alors la valeur représentée est :

$$(-1)^s \times \left(1 + m_1 \times 2^{-1} + m_2 \times 2^{-2} + \ldots + m_{23} \times 2^{-23}\right) \times 2^{\left(e_7 \times 2^7 + \ldots + e_2 \times 2^2 + e_1 \times 2 + e_0\right) - 127}$$

Les nombres représentés ainsi sont dit **normalisés**. Le plus petit nombre représentable est alors approximativement $1 \times 2^{1-127} = 2^{-126}$ (soit environ 10^{-38})
Cependant, puisque la mantisse est de la forme $1.m_0m_1 \ldots m_23$, il n'y a pas de nombre représentable dans l'intervalle $[0,\ 2^{-126}[$ alors qu'il y en a 2^23 dans l'intervalle $[1 \times 2^{-126},\ 2 \times 2^{-126}] = [2^{-126},\ 2^{-125}]$.
Afin de rééquilibrer la représentation des nombres autour de 0, la norme IEEE 754 permet d'encoder des nombres de la forme :

$$(-1)^s \times 0.m_1m_2 \ldots m_M \times 2^{-126}$$

avec une mantisse commençant implicitement par 0 (alors que pour les nombres normalisés, elle commence implicitement à 1).

9. Il y a néanmoins deux représentations de l'infini dans ce système : `+inf` si le bit de signe est à 0 et `-inf` si le bit de signe est à 1

Ces nombres flottants sont appelés nombres **dénormalisés** et correspondent à un **exposant biaisé nul** mais une **mantisse non nulle**. De cette manière, la plus petite valeur représentable avec des nombres dénormalisés est :

$$2^{-23} \times 2^{-126} = 2^{-149} \approx 1.4 \times 10^{-45}$$

Avantage n°1 : On peut donc représenter des nombres plus petits.

Avantage n°2 : Cela assure également une continuité avec les nombres normalisés, puisque :

- le plus petit nombre normalisé vaut $(1.000\ldots00)_2 \times 2^{-126}$ (car $(1.000\ldots00)_2 \times 2^{-127}$ est réservé pour l'encodage du 0)
- le plus grand nombre dé normalisé vaut $(0.111\ldots11)_2 \times 2^{-126} = 1.111111 \times 2^{-127}$

Soit un « saut » de $(0.000\ldots01)_2 \times 2^{-126} = 2^{-23} \times 2^{-126} = 2^{-149} \approx 1.4 \times 10^{-45}$.

Nous avons donc une continuité entre les nombres dénormalisés et les nombres normalisés, comme indiqué ci-dessous pour $N = 32$ bits [10] :

Type	Exposant	Mantisse	Valeur approchée	Écart / préc
Zéro	0000 0000	000 0000 0000 0000 0000 0000	0,0	
Plus petit **nombre dénormalisé**	0000 0000	000 0000 0000 0000 0000 0001	$1{,}4 \times 10^{-45}$	$1{,}4 \times 10^{-45}$
Nombre dénormalisé suivant	0000 0000	000 0000 0000 0000 0000 0010	$2{,}8 \times 10^{-45}$	$1{,}4 \times 10^{-45}$
Nombre dénormalisé suivant	0000 0000	000 0000 0000 0000 0000 0011	$4{,}2 \times 10^{-45}$	$1{,}4 \times 10^{-45}$
...	...	...	...	...
Plus grand nombre dénormalisé	0000 0000	111 1111 1111 1111 1111 1111	$1{,}175\,494\,21 \times 10^{-38}$	$1{,}4 \times 10^{-45}$
Plus petit **nombre normalisé**	0000 0001	000 0000 0000 0000 0000 0000	$1{,}175\,494\,35 \times 10^{-38}$	$1{,}4 \times 10^{-45}$
Nombre normalisé suivant	0000 0001	000 0000 0000 0000 0000 0001	$1{,}175\,494\,49 \times 10^{-38}$	$1{,}4 \times 10^{-45}$

V. 6. Problèmes liés à la représentation des nombres réels

V. 6. a. Arrondis

Il est très important que la représentation des nombres réels par des flottants est approximative. Par exemple, le nombre 1.6 ne peut être représenté exactement avec cet encodage. Il faut arrondir cette valeur pour trouver le *meilleur* nombre flottant pour le représenter. Mais encore faut-il s'accorder sur ce que signifie *meilleur*. La norme IEEE754 adopte par défaut l'arrondi au plus près : c'est le flottant dont la valeur est la plus proche de la valeur exacte qui est choisi. En cas d'églité, on privilégie la représentation ayant une mantisse se terminant par 0.

Exemple 15

Pour 1.6, le nombre le flottant simple précision le plus proche est :

0 01111111 10011001100110011001101

qui correspond au nombre décimal 1.60000002384185791015625

10. Source : Wikipedia

En simple précision, on ne peut représenter proprement que 7 chiffres significatifs décimaux

Exemples 16

1. $1000333.444555 \to 1000333.xxxxxx$
2. $0.000000000000000012 \to 0.000000000000000012$
3. $0.00000000000012345678765 \to 0.000000000000012345 67xxxx$

V. 6. b. Opérations dangereuses

Phénomène d'absorption. Ce phénomène se produit lorsque l'on additionne deux nombres qui ont des ordres de grandeur très différents.

Exemple 17

Essayons d'effectuer l'opération ci-dessous en simple précision $N = 32$ bits.

$$2^{20} + 2^{-13} = (1 + 2^{-33}) \times 2^{20}$$

Dans la représentation de la somme, le terme de mantisse 2^{-33} est perdu car la mantisse n'a que 32 bits.

Phénomène de perte de chiffres significatifs. On soustrait deux nombres très proches : on passe sous la précision absolue et le calcul devient faux. Cela est particulièrement vrai lorsque l'on soustrait deux grands nombres car l'écartement entre deux grands nombres est plus grand qu'entre deux petits.

Exemple 18

Essayons d'effectuer l'opération ci-dessous en simple précision $N = 32$ bits.

$$1 - 0.99999999 = (1 + 2^{-33}) \times 2^{20}$$

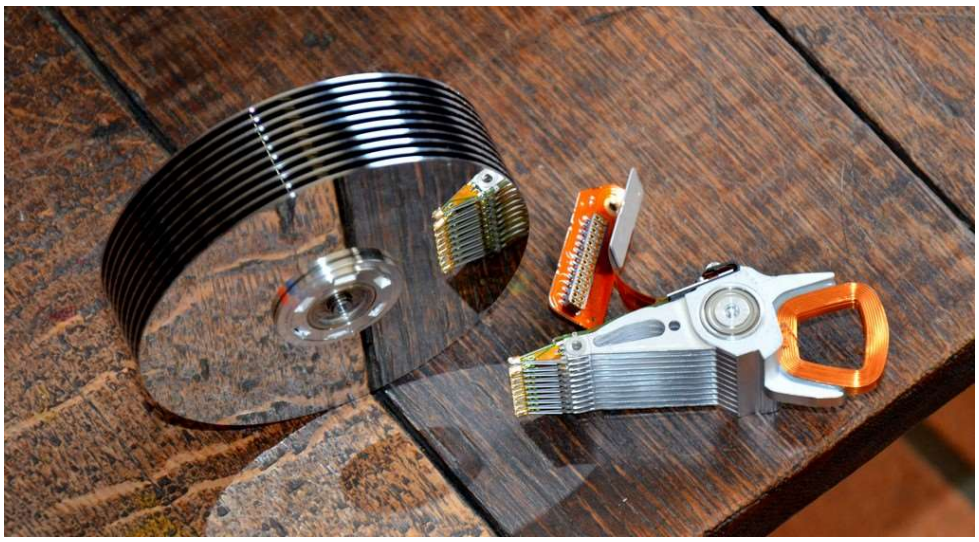
Dans la représentation de la somme, le terme de mantisse 2^{-33} est perdu car la mantisse n'a que 32 bits.

Problèmes sur les tests d'égalité. Il est très très imprudent d'écrire des programmes dans lesquels on utilise des tests d'égalité sur des flottants. Si l'on compare $0.1 + 0.2$ et 0.3 avec un test d'égalité, la machine nous répondra que ces deux nombres ne sont pas égaux.

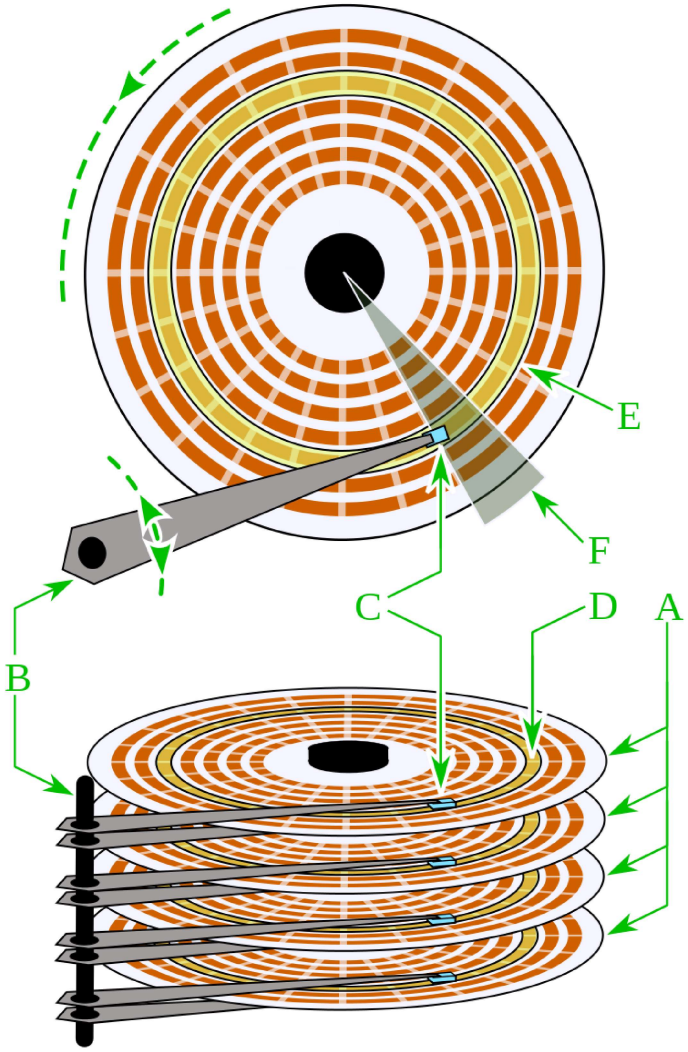
Exemple 19

Tout le logiciel de la fusée Ariane 5 est programmé en Ada, langage très sûr. Et pourtant ... l'échec du premier tir d'Ariane 5 a été causé par une erreur du système informatique de guidage. Cette erreur est survenue lors d'une conversion de type qui a causé un dépassement de capacité d'une variable. Parmi les recommandations émises suite à

cet accident on notera : *Vérifier la plage des valeurs prises dans les logiciels par l'une quelconque des variables internes ou de communication.*

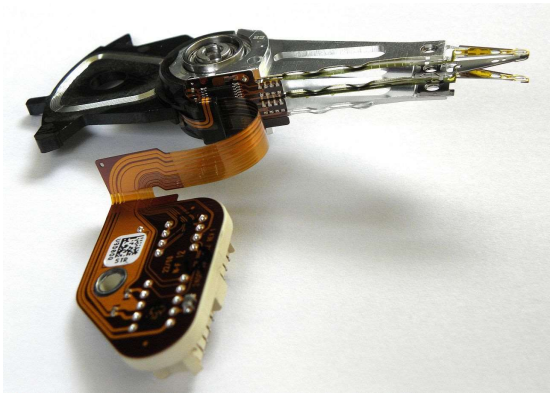


(a) Plateaux et têtes de lecture/écriture en peigne.

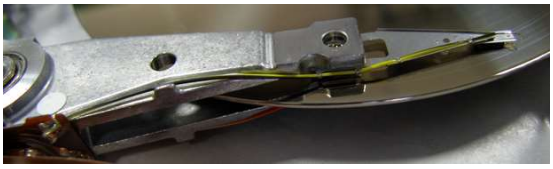


(b) Plateaux (A), Têtes (B), Zone mémoire visée (C), repérage CHS associé : Cylinder (D) , Head (E), Sector (F)

FIGURE 3 – Disque dur magnétique à plusieurs plateaux et ses têtes de lecture. Repérage des données par CHS (Cylinder-Head-Sector)



(a) Tête seule



(b) Tête positionnée sur disque

FIGURE 4 – Tête de lecture/écriture d'un disque dur magnétique

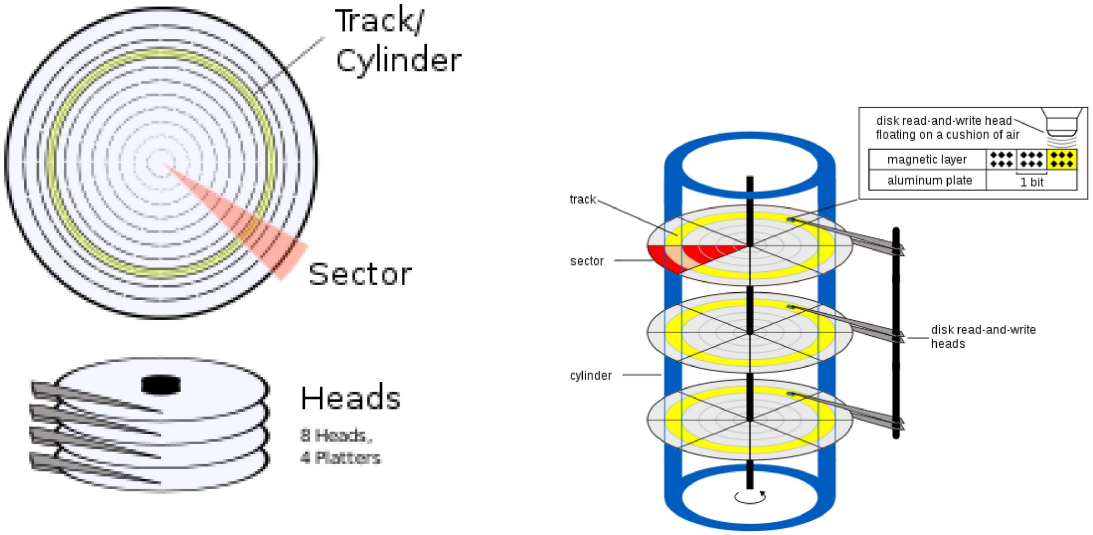
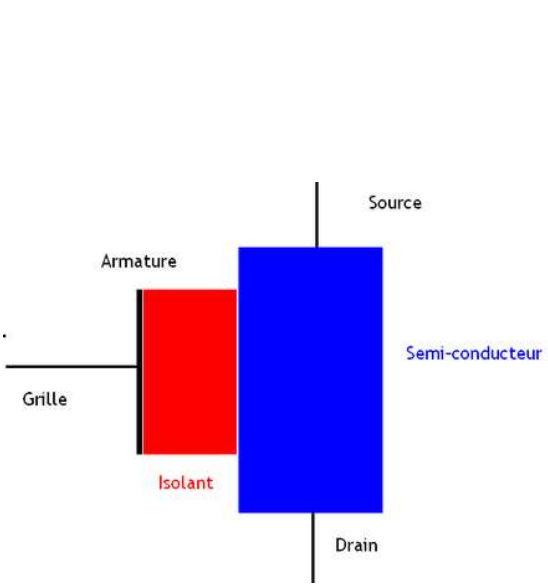
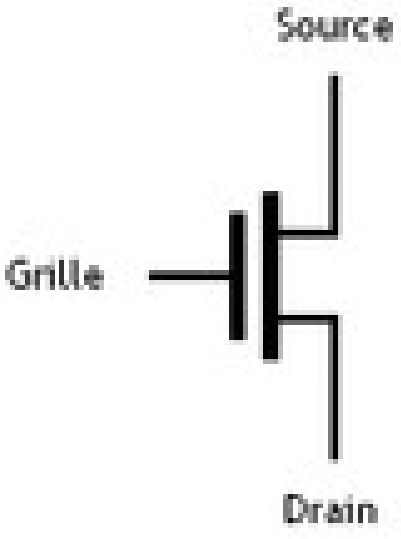


FIGURE 5 – Repérage CHS. Source : Wikipédia `https://en.wikipedia.org/wiki/Cylinder-head-sector`

FIGURE 6 – Deux exemples de supports optiques : le CD-ROM et le DVD Blu-Ray



(a) Physique d'un transistor CMOS



(b) Schéma logique d'un transistor CMOS

FIGURE 7 – Le transistor CMOS : un interrupteur pilotable

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

FIGURE 8 – Convention adoptées pour le jeu de caractères ASCII. Source : `https://www.fil.univ-lille1.fr/~wegrzyno/portail/Info/Doc/HTML/seq7_codage_caracteres.html`

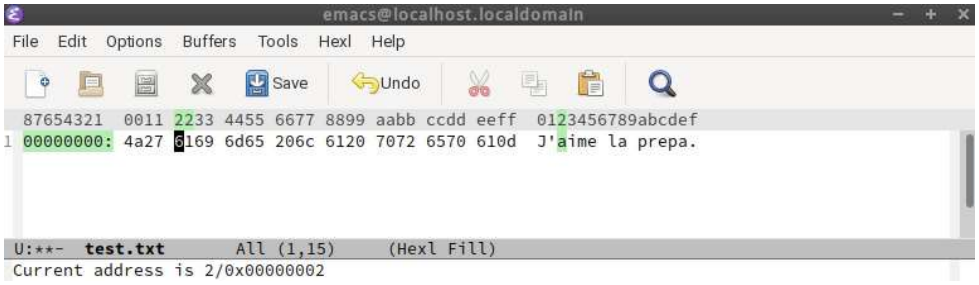


FIGURE 9 – Encodage hexadécimal d'un texte obtenu grâce au mode `hexl-mode` de l'éditeur de texte `emacs`