04 - Représentation des nombres

Les nombres en base 10 seront représentés sans indice, c'est-à-dire que 32_{10} sera représenté par 32.

I Entiers

Exercice 1 Quel est l'entier maximal non signé que l'on peut écrire sur 8 bits ? Sur 32 bits ? Sur 64 bits ?

Exercice 2 Convertir *à la main* en binaire les nombres suivants : 9 - 33 - 253 - 1026 - 2051 - 56 - 325 - 945

Exercice 3 Donner la valeur en base 10 *à la main* des nombres en binaire suivants :101_2 - 1111_2 - 10000_2 - 1010_2 - 10101011_2 - 100000001_2 - 11111111_2

Exercice 4 Que vaut 2^{10} en décimal ? En déduire rapidement l'ordre de grandeur des nombres suivants : 2^{30} - 2^{31} - 2^{64}

Exercice 5 Quel est l'entier maximal signé que l'on peut écrire sur 8 bits ? Sur 32 bits ? Sur 64 bits ?

Exercice 6 Que vaut, sur 8 bits signés, 127+1 ? Sur 8 bits non signés ?

Exercice 7 Donner la valeur des octets suivants, dans le cas de nombres signés.

- | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
- | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
- | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
- | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
- | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
- | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Que deviennent ces résultats quand les nombres sont non signés ?

Exercice 8 Convertir sur un octet les nombres -16, -43, -64, -127. Aura-t-on la même représentation sur 32bits ?

Exercice 9 On appelle base *hexadécimale* la base 16, où les symboles sont 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

9.1 Ecrire 15 en base hexadécimale

9.2 Quel est le nombre maximal que l'on peut représenter avec 2 symboles hexadécimaux ?

9.3 En déduire que la valeur d'un octet peut être représentée par un hexadécimal à 2 symboles.

9.4 Représenter les octets non signés suivants en hexadécimal :

- | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
- | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
- | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
- | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
- | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

9.5 Donner les octets correspondants à $1F_{16}$ - 48_{16}, DD_{16}, $0E_{16}$

II Flottants

Rappel des bases de la norme IEEE 754, simple précision : on code sur 32 bits avec 1 bit de signe s, 8 bits d'exposants exp et 23 bits de mantisse ma. Le décalage d de l'exposant est de $2^{8-1} - 1 = 127$.

L'exposant représenté par un nombre non signé, entre 0 et $2^8 - 1$*, puis l'exposant est calculé selon* $e = exp - d$*. La représentation de la mantisse ma est calculée par* $ma = \sum_{i=1}^{23} b_i * 2^{-i}$ *où i est la place du bit (*$i = 1$ *pour le bit de poids fort de ma.*

On distingue trois types de représentations :

- *Les nombres normalisés, où l'exposant est différent de 0 et de 255, dont la valeur vaut* $v = (-1)^s.m.2^e$ *avec* $m = ma + 1$
- *Les nombres où l'exposant vaut 0, dont la valeur vaut* $v = (-1)^s.ma.2^{-126}$ *(ce sont les plus petits nombres en valeur absolue)*
- *Les nombres où l'exposant vaut 255, dont la valeur vaut* ∞ *si la mantisse vaut 0, NaN sinon.*

Exercice 10 Calculer les mantisses ma, correspondant à (les nombres non donnés sont des zéros) $100..._b$, $010..._b$,$0010..._b$,$0001...._b$. Comment en déduire m ?

Exercice 11 Calculer les exposants, correspondant à $\boxed{1|1|1|1|1|1|1|0}$, $\boxed{0|0|0|0|0|0|0|1}$, $\boxed{0|1|1|1|1|1|1|1}$.

Exercice 12 Donner, pour une valeur donnée de e, l'écart entre deux nombres successifs. Quelle est la plus grande valeur de cet écart ?

Exercice 13 Donner la valeur :

- du plus petit nombre non nul dénormalisé (en valeur absolue)
- du plus grand nombre différent de ∞ ou NaN dénormalisé
- du plus petit nombre normalisé (en valeur absolue)
- du plus grand nombre normalisé

Exercice 14 Représentation de la partie décimale d'un nombre Si on prend un nombre réel, et qu'on désigne par x sa partie décimale ($x \in [0, 1[$), on peut écrire que :

$$x = \sum_{i=1}^{+\infty} \frac{b_i}{2^i}$$

où les b_i valent soit 0, soit 1 (il s'agit de la décomposition en base 2 de la partie décimale). Pour calculer successivement les b_i, il suffit d'appliquer l'algorithme suivant :

1. on multiplie x par 2
2. si $x \geq 1$, on ajoute un 1 à la représentation et on soustrait 1 à x
3. sinon on ajoute un 0 à la représentation.
4. et on recommence...

L'algorithme s'arrête dès que x est nul... ou jamais.

Par exemple, pour $x = 0,25$, on a $x * 2 = 0.5$ donc on écrit `0`. 0.5*2=1, on ajoute 1 : `01`, et x devient nul, on s'arrête (on trouve bien que $0,25 = \frac{0}{2} + \frac{1}{2^2} = 0,01_2$)

14.1 Calculer la représentation de 0,25 - 0,6875. Faire de même avec 0,3 et constater que dans ce cas, l'algorithme ne s'arrêtera pas.

14.2 Coder en python une fonction `calcul_mantisse(x:float)->list` qui retourne sous forme d'une liste (de 23 termes) les 23 premiers termes du développement décimal en binaire de x. Par exemple `calcul_mantisse(0.3)` va retourner :

```
[0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0]
```

14.3 Coder alors une fonction `verif_mantisse(l:list)->float` qui calcule le nombre $x = \sum_{i=1}^{23} \frac{l_i}{2^i}$, où les l_i sont les éléments de la liste retournée par `calcul_mantisse`.

Exercice 15 Donner la représentation en norme IEEE 754 simple précision de : 0 - 1 - 21 - 1563 - 2,2