La numération informatique des nombres :

1. Les entiers naturels :

Vous connaissez déjà la représentation des nombres par leur équivalent binaire. Il existe une seconde numération encore très utilisée dans les systèmes électronique : le BCD (Décimal Codé Binaire). Chaque chiffre d'un nombre est représenté, dans l'ordre, par une code binaire.

Il existe des milliards de codes différents, mais le plus connus reste le Excess-3 (XS-3) : Chaque chiffre est encodé séparément par sa valeur binaire augmentée de 3.
Par exemple pour le nombre 123 :
On code séparément : 1 , 2 , 3 augmenté de 3 : => 1+3 , 2+3 , 3 + 3 => 4 , 5 , 6
Soit en binaire : 0100 0101 0110

Opérations arithmétiques :

Complémentation : Prenons l'exemple du chiffre 4 codé binaire :

4 en XS-3 : 0111

non(4) : 1000 soit 5 et magiquement : 9 - 4 = 5 !

Pour effectuer, 10 - N il suffit donc de faire non(N) + 1, ce qui est très rapide à calculer.

Additions :

```
          A:   10    0100 0011
          B: + 12  + 0100 0101
          C:   22    1000 1000
Décimale:      22       8    8
```

Nous avons donc un décalage de 6 : nous trouvons 88 au lieu de 22. Il faut donc *corriger* le nombre.

Pour corriger, il faut enlever 3 à chaque « chiffre » (avant conversion) s'il est >= 1010 (10) ou ajouter 3 s'il est plus petit.

```
            28      0000 0101 1011
          + 93      0000 1100 0110
           121      0001 0010 0001
Correction:          +3   +3   +3
                    0100 0101 0100
Conversion:            1    2    1
```

(0001, 0010 et 0001 < 1010)

```
              3  0110
            + 4  0111
              7  1101
correction:      -3
conversion:       7
```

(1101 >= 1010)

Soustrations :

```
                1
          7     1010
        - 4     0111
          3     0011
Correction:     +3
conversion:      3
```

La méthode fonctionne toujours avec des soustractions (pour les soustractions en binaire voir la partie 2).

2. Les entiers relatifs :

Les ordinateurs n'utilisent pas de « bit de signe » pour représenter les nombres négatifs. On pourrait penser qu'il réservent le premier bit du nombre pour codifier le signe, mais cela rendrait les calcules beaucoup plus complexes car il faudrait 3 opérateurs d'additions pour positif+positif, positif+négatif et négatif+négatif !
On utilise donc la méthode de la *complémentation à deux* qui permet de simplifier les calcules ! Notez que les processeurs n'ont pas d'autre moyen de soustraction que l'addition de l'opposé (du complément à deux).

Complément à deux :

Ajout de 1 au complément à un (inversion) du nombre binaire.
Soit : -N = non(N) + 1

Ex : 21 => 0001 0101
-21 = non(0001 0101) + 1 => 1110 1010 + 1 = 1110 1011

3. Les nombres décimaux :

a. Virgule fixe :

Principe : nous attribuons M bits pour la partie entière et N bits pour la partie décimale. Il s'agit simplement d'un ajout de bits à droite.

B^4	B^3	B^2	B^1	B^0	B^{-1}	B^{-2}	B^{-3}	B^{-4}
A	B	C	D	E	F	G	H	I

La conversion est la même qu'en binaire « classique ». $A.B^4 + B.B^3 + \ldots + I.B^{-4}$. Sauf que les poids négatifs sont inférieurs à 1.

Il existe une méthode simple pour convertir un nombre décimal dans une base B. Il faut multiplier la partie décimale par la base B, et à chaque fois récupérer la partie entière.

Exemple :

0,25 en hexadécimal. :

0,251*16 = **4**,016
0,016*16 = **0**,256
0,256*16 = **4**,096
0,096*16 = **1**,506
0,506*16 = **8**,096

On recopier dans l'ordre : 0x 0,**40<u>418</u>**. J'ai arrêté la conversion, car on repère un motif, la conversion ne s'arrêtera donc jamais.

b. Virgule flottante :

Nous allons voir la norme IEEE 754 qui est la plus répandue.

<u>Nombres normalisés :</u>

Conversion d'un nombre décimal en binaire IEEE 754 :

Norme :	**Bit de signe s**	**Exposant e**	**Pseudo-mantisse m**
Simple : 32 bits	1	8	23
Double : 64 bits	1	11	52
Quadruple : 128 bits	1	15	112

Il faut mettre le nombre décimal en binaire :

$345,55_{/10}$ avec la méthode vu en partie *3.a* :

$345_{/10}$ = 0b 101011001

0,625 * 2 = **1**,25

1,250 * 2 = **0**,5

0,500 * 2 = **1**

$345,55_{/10}$ = 0b 101011001, 101

Il faut l'écrire en *notation scientifique* :

0b 1,01011001101*2^8.

Traitement de l'exposant :

Il faut ajouter 2^{e-1}-1 à l'exposant où e représente le nombre de bits de l'exposant. (8 bits en 32 bits, 11 en 64 bits).

Ici il faut donc ajouter 2^{8-1}-1 à notre exposant 8 : 8 + 2^{8-1}-1 = 8 + 127 = 135

Encodage :

Bit de signe :	**Exposant**	**Partie pseudo-mantisse (partie fractionnaire de la forme scientifique sans le 1)**
<u>0</u>	8+127=135 1000 0111	**0101100110**1000000000000
1 : négatif 0 : positif	Exposant augmenté de 2^{e-1}-1 sur e bits.	Mantisse sur m bits.

Ce qui donne <u>0</u>100 0011 1**010 1100 1101** 0000 0000 0000