Chapitre 6 :
Représentation de nombres
Nombres négatifs et flottants

I. Nombres négatifs

1) Le binaire signé : introduction aux nombres négatifs

Première idée :

On ne change rien à la manière de représenter un entier et on utilise le bit de poids fort (à gauche) comme un **bit de signe** : 0 indique un nombre positif, 1 indique un nombre négatif.

Exemple :
0101 est 5 en base 2.
1101 serait -5 en base 2.

Application :
Sur 8 bits, dans le cadre du binaire signé :
- quel nombre en base 10 est égal à 0001 1111 ? 1001 0010 ?
- pourriez-vous écrire +0 ? et -0 ?!
- comment s'écrirait 9 ? -9 ? Faire l'addition de ces deux nombres : que peut-on conclure ?

Conclusion : les règles d'addition ne fonctionnent plus avec cette règle. Bien que facile à utiliser pour représenter des nombres négatifs, le binaire signé ne permet pas de faire des opérations sur les nombres binaires. Et là… c'est le drame :(

2) Nombres négatifs avec le complément à 2

DM Complément à 2

Définition : le complément à 1 d'un nombre binaire est le nombre dont tous les 0 et les 1 ont été inversés.

Exemple :
Le complément à 1 de 0110 est 1001.

Principe de codage des entiers **relatifs :**

❖ Codage des entiers positifs :

Le codage des entiers positifs est analogue au codage en binaire.

❖ Codage des entiers négatifs :

Pour coder des entiers strictement négatifs, on procède ainsi :

a. On code la valeur absolue (le nombre positif) du nombre en binaire sur le nombre de bits indiqué.

b. On trouve le complément à 1 du nombre précédent : on inverse tous les 1 avec des 0

c. On ajoute 1 au nombre binaire obtenu

Exemple :

❖ Le nombre 77 se code $100\ 1101_2$. Sur 8 bits, le codage est $0100\ 1101_2$.

❖ Pour trouver le nombre -77, on fait le complément à 1 de 77 : $1011\ 0010_2$.

Puis on ajoute 1: $1011\ 0011_2$

Application :

a. Écrire -12 sur 8 bits avec la méthode du complément à 2. *Réponse : 11110100*

b. Écrire -64 sur 8 bits avec la méthode du complément à 2. *Réponse : 11000000*

c. Quelles sont les bornes inférieure et supérieure d'un entier relatif codé sur 16 bits ? (voir exercice 10)

Principe de décodage des entiers binaires relatifs

❖ Décodage des entiers positifs (0……) :

Le décodage des entiers binaires positifs est analogue au celui des entiers positifs.

❖ Décodage des entiers négatifs (1……) :

Pour coder des entiers strictement négatifs, on utilise la méthode inverse :

a. On enlève 1 au codage binaire de l'entier relatif.

b. On trouve le complément à 1 du nombre précédent : on inverse tous les 1 avec des 0

c. On lit la valeur absolue du nombre binaire.

Exemple : 1000 1001.

❖ On enlève 1 à ce nombre : 1000 1000

❖ On fait le complément à 1 : 0111 0111

❖ Ce nombre est la valeur absolue de notre nombre négatif initial :

$0111\ 0111 = 2^6 + 2^5 + 2^4 + 2^2 + 2^1 + 2^0 = 119$

❖ On en conclut : $1000\ 1001_2 = -119_{10}$

Application :

a. Quel est le nombre relatif égal à 11101011 sur 8 bits ? *Réponse : -21*

b. Quel est le nombre relatif égal à 10001010 sur 8 bits ? *Réponse : -118*

c. Quel est le nombre relatif égal à 10011000 sur 8 bits ? *Réponse : -104*

d. Quel est le nombre relatif égal à 10000001 sur 8 bits ? *Réponse : -127*

II. Nombres flottants

Problème : comment représenter π ou $\frac{1}{3}$ qui ont une virgule et un nombre infini de décimales avec des 0 et des 1 ?

1) Définition

Définition :

Un nombre flottant est un nombre réel codé sur un nombre **fini de bits**.

Rem : La précision d'un nombre flottant est donc limitée.

2) Nombre à virgule en base 2 non signé

Comment écrire des nombres décimaux en base 2 ?

On va faire une décomposition en puissances de 2, mais avec des **exposants négatifs** !

Exemple :

$$110{,}1011_2 = 1\times 2^2 + 1\times 2^1 + 0\times 2^0 + 1\times 2^{-1} + 0\times 2^{-2} + 1\times 2^{-3} + 1\times 2^{-4}$$
$$= 6{,}6875_{10}$$

Du coup, on se retrouve à nouveau avec des 0 et des 1 derrière la virgule.

Méthode : binaire→base 10

On utilise une décomposition en puissance de 2 avec des puissances de 2 négatives pour les bits après la virgule.

Application :

Trouvez la représentation décimale de $(100{,}001)_2$ et $(0{,}00101)_2$

Méthode : base 10→binaire

La partie entière est codée de la même manière que précédemment.

La partie fractionnaire est codée de la façon suivante :

- On multiplie par 2 la partie fractionnaire.
  - ➡ si elle est supérieure à 1, on retranche 1 au résultat et on enregistre le bit 1.
  - ➡ si elle est inférieure à 1, on enregistre le bit 0.

Exemple : Convertir 0,1875 en base 2

0,	$0{,}1875$	
0	$0{,}1875 \times 2 = 0{,}375$	
0	$0{,}375 \times 2 = 0{,}75$	
1	$0{,}75 \times 2 = 1{,}5$	1,5-1 = 0,5
1	$0{,}5 \times 2 = 1$	1-1 = 0
0	$0 \times 2 = 0$	Fin

Application :

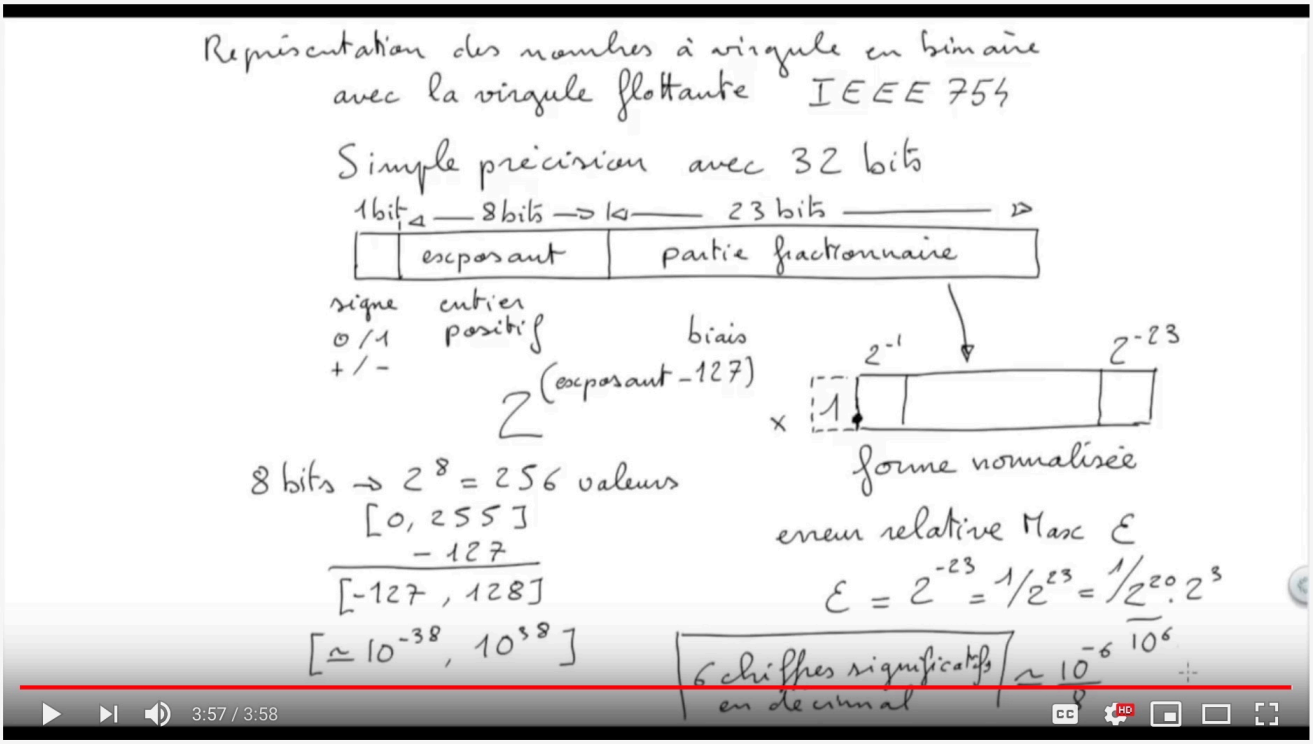
Trouvez la représentation binaire de $(4{,}65625)_{10}$ et $(0{,}1)_{10}$.

3) Représentation des nombres flottants par un ordinateur

Électriquement, comment se traduit la virgule ?! C'est exactement le même problème que pour le signe "moins".

Comme avec les signes, on va utiliser un grand nombre de bits (i.e. 32 ou 64) pour stocker les bits après la virgule !

Activité IEEE 754


Représentation des nombres à virgule en binaire
avec la virgule flottante IEEE 754
Simple précision avec 32 bits
1 bit
8 bits
23 bits
exposant
partie fractionnaire
signe
0 / 1
+ / -
entier positif
biais
2^(exposant - 127)
2^-1
2^-23
1
x
forme normalisée
8 bits → 2^8 = 256 valeurs
[0, 255]
- 127
[-127, 128]
[≃ 10^-38, 10^38]
erreur relative Max ε
ε = 2^-23 = 1/2^23 = 1/2^20 . 2^3
10^6
6 chiffres significatifs en décimal
≃ 10^-6
3:57 / 3:58