
Représentation des nombres

BARBIER J.M. - LGT Dumezil - NSI

29 septembre 2025

1 Représentation des nombres entiers

1.1 Nombres positifs

1.1.1 Principe

Représentation d'un nombre entier positif ou nul : binaire de base

1.2 Nombres négatifs

1.2.1 Problème

comment représenter le signe –

- avec uniquement des bits
- dans un nombre de bits fixé

1.2.2 Bit de signe

1 bit indique le signe, le reste = le nombre en valeur absolue

exemples sur 8 bits :

- 0 000 0101 = +5
- 1 000 0101 = –5

inconvénients :

- comment représenter zéro ? 1 000 0000 et 0 000 0000
- l'addition marche plus : $-5 + 3$

$$1\ 000\ 0101 + 0\ 000\ 0011 = 1\ 000\ 1000 = -8$$

1.2.3 Complément à deux

- change rien pour les nombres positifs
- on obtient les nombres négatifs en
 - inversant tous les bits de la valeur absolue (= complément à 1)
 - on ajoute 1 au résultat (sans s'occuper des dépassements)

1.2.4 Complément à deux : exemples

- pour coder -5 sur 8 bits

1. 0000 0101
2. 1111 1010
3. 1111 1011

— coder -3 sur 16 bits

1. 0000 0000 0000 0011
2. 1111 1111 1111 1100
3. 1111 1111 1111 1101

vérification dans RISC simulator 1111 1111 1111 1101

— pour coder -0

1. 0000 0000
2. 1111 1111
3. 0000 0000

avantage : l'addition fonctionne correctement

$$-5 + 8 = 11111011 + 00001000 = 00000011 = 3$$

1.2.5 Complément à deux : exos conversion



Écrire les nombres suivants en complément à deux sur 8 bits

- -14
- -55
- -20



Convertir les nombres suivants (écrits en complément à deux) en décimal

- 1101 1011
- 1011 0110
- 0001 0011

1.2.6 Complément à deux : exos opérations



Faire les opérations suivantes en décimal puis en binaire complément à deux; vérifier le résultat

- $-8 + -5$
- $-12 + 15$
- $-8 + 9$
- $9 + -12$

2 Représentation des nombres décimaux

2.1 Représentation en virgule fixe

2.1.1 Principe

lorsque le nombre de chiffres en base 10 après la virgule est fixé, on peut simplement décaler tous les nombres d'un facteur donné

exemple opérations financières en 64 bits :

- on ne se préoccupe pas des millièmes de centimes
- on a rarement à traiter des opérations dépassant plusieurs milliers de milliards d'euros ($2^{64} = 18446744073709551615$)

=> on code les montants en centimes, ou en décicentimes, ou en centicentimes

2.1.2 Exercices



Sur 32 bits, en complément à deux, peut-on représenter en virgule fixe les montants d'une grosse société, sachant que

- on veut une précision au millième de centimes
- on veut pouvoir manipuler des montants jusqu'à 10 milliards d'euros

2.2 Conversion décimal à virgule - binaire à virgule

2.2.1 Principe

Conversion en binaire à virgule :

- on fait des divisions successives pour la partie entière
- on fait des multiplications successives par 2 pour la partie à virgule

2.2.2 Exemple : conversion de 28,8625 en binaire

– Conversion de 28 : 11100_2 – Conversion de 0,8625 :

- $0,8625 \times 2 = 1,725 = 1 + 0,725$
- $0,725 \times 2 = 1,45 = 1 + 0,45$
- $0,45 \times 2 = 0,9 = 0 + 0,9$
- $0,9 \times 2 = 1,8 = 1 + 0,8$
- $0,8 \times 2 = 1,6 = 1 + 0,6$
- $0,6 \times 2 = 1,2 = 1 + 0,2$
- $0,2 \times 2 = 0,4 = 0 + 0,4$
- $0,4 \times 2 = 0,8 = 0 + 0,8$

=> $11100,11011100..._2$

2.2.3 Exercices



Représenter en binaire à virgule les nombres suivants :

- 0,1
- 0,25
- 1/3



Convertir, puis calculer en binaire : $0,1 + 0,2$

2.3 Représentation en virgule flottante

2.3.1 Principe

pour les calculs scientifiques (la mole :) on peut avoir besoin de nombres très grands ou très petits => notation scientifique

pour représenter un nombre en notation scientifique dans un nombre fixé de bits, on utilise une représentation normalisée (IEEE 754)

en gros

- 1er bit = le signe
- e bits suivants = l'exposant
- le reste des bits = la mantisse

l'exposant peut être positif ou négatif; on pourrait le représenter en complément à deux, mais c pas facile pour les comparaisons => on décale de $2^{e-1} - 1$ sa valeur

2.3.2 Flottant 32 bits (float32)

- 1 bit de signe (1 pour négatif)
- 8 bits pour l'exposant
- 23 bits pour la mantisse (la partie après la virgule)

=> on décale l'exposant de $2^{8-1} - 1 = 127$

- exposant 3 = exposant 130
- exposant 0 = exposant 127
- exposant 10 = exposant 137

Valeur du nombre

$$v = s \times 2^{e-127} \times (m + 1)$$

- $s = 1$ ou -1 selon le bit de signe
- e est l'exposant
- m est la mantisse

2.3.3 Exemple : $-118,625$ en float32

- le signe (1er bit) : nombre négatif, le premier bit est 1
- écriture du nombre (sans le signe) en binaire : $111\,0110,101$
- décalage de la virgule vers la gauche, de façon à ne laisser qu'un 1 sur sa gauche : $1110110,101_2 = 1,110110101_2 \times 2^6$. C'est un nombre flottant normalisé : la mantisse est la partie à droite de la virgule, remplie de 0 vers la droite pour obtenir 23 bits. Cela donne $110\,1101\,0100\,0000\,0000\,0000$ (on omet le 1 avant la virgule, qui est implicite).
- l'exposant est égal à 6, il faut le décaler et le convertir. Pour le format 32-bit IEEE 754, le décalage est $28 - 1 - 1 = 127$. Donc $6 + 127 = 133$ et $133_{10} = 1000\,0101_2$.

Résultat :

$$-118,625_{10} = 1100\,0010\,1110\,1101\,0100\,0000\,0000\,0000_2 = C2ED4000_{16}$$

2.3.4 Exercices



Écrire $452,778$ en **float32**.



Combien vaut $0\,10000110\,101111010000000000000000$ en décimal ?

Aide : vérification et entraînement sur

<https://www.h-schmidt.net/FloatConverter/IEEE754.html>

2.3.5 Conséquences

Les valeurs stockées en virgule flottante ne sont **pas exactes**, idem pour les calculs

- on ne teste pas l'égalité de deux flottants
- on fait attention aux erreurs qui s'accumulent dans les calculs