Avec l'addition et la multiplication, la soustraction et la division non-euclidienne sont deux autres opérations élémentaires.
La soustraction est équivalente à l'addition mais avec l'utilisation de nombre négatifs. Cela nécessite donc l'expansion de la représentation des nombres aux entiers relatifs.
La division non-euclidienne, quant à elle, requière de pouvoir représenter les nombres flottants (les nombre à virgule).
Comment représenter des nombres négatifs ou flottant en binaire, en considérant que le nombre de bits est limité à 8, 16, 32 ou 64 bits ?

# Partie I : Entiers relatifs

Dans cette partie, nous nous concentrerons sur un codage sur 8 bits.

## A. Méthode naïve

Pour coder des entiers relatifs sur 8 bits, on peut appliquer la méthode suivante, dite méthode naïve (à cause de sa simplicité).

La méthode naïve est la suivante :

On utilise le bit le plus à gauche, appelé bit de poids fort, pour déterminer le signe du nombre entier : 0 pour des nombres positifs et 1 pour des nombres négatifs.

Le codage d'un entier relatif $n$ **sur 8 bits** se fait par la méthode naïve comme suit :

- Si $0 \leq n \leq 127$, alors $n$ est représenté par 0 (pour « positif ») puis son codage en base 2.
- Si $-127 \leq n \leq 0$, alors $n$ est représenté par 1 (pour « négatif ») puis le codage de son opposé $-n$.
- Si $n > 127$ ou si $n < -127$, alors on ne peut pas représenter ainsi $n$ en binaire.

**I.A.1.** D'où proviennent les limites « 127 » et « -127 » ?

**I.A.2.** En suivant cette méthode, donner les représentations binaires des nombres décimaux suivants.

**a.** 80 **b.** -1 **c.** -127 **d.** -32 **e.** -60 **f.** -200

**I.A.3.** En suivant cette méthode, donner les représentations décimales des nombres binaires sur 8 bits suivants.

**a.** $\overline{01101101}^2$ **b.** $\overline{11101101}^2$ **c.** $\overline{01110010}^2$ **d.** $\overline{00110010}^2$ **e.** $\overline{11100101}^2$ **f.** $\overline{10100111}^2$

**I.A.4.** Compléter les définitions de la méthode naïve sur 16 bits puis sur 32 bits présentées ci-dessous en utilisant des puissances de 2.

Le codage d'un entier relatif $n$ **sur 16 bits** se fait par la méthode naïve comme suit :

- Si $0 \leq n \leq$ ..... , alors $n$ est représenté par 0 (pour « positif ») puis son codage en base 2.
- Si ..... $\leq n \leq 0$, alors $n$ est représenté par 1 (pour « négatif ») puis le codage de son opposé $-n$.
- Si $n >$ ..... ou si $n <$ ..... , alors on ne peut pas représenter ainsi $n$ en binaire.

On peut alors représenter ..... entiers relatifs en binaire grâce à cette méthode.

Le codage d'un entier relatif $n$ **sur 32 bits** se fait par la méthode naïve comme suit :

- Si $0 \leq n \leq$ ..... , alors $n$ est représenté par 0 (pour « positif ») puis son codage en base 2.
- Si ..... $\leq n \leq 0$, alors $n$ est représenté par 1 (pour « négatif ») puis le codage de son opposé $-n$.
- Si $n >$ ..... ou si $n <$ ..... , alors on ne peut pas représenter ainsi $n$ en binaire.

On peut alors représenter ..... entiers relatifs en binaire grâce à cette méthode.

Cette méthode de représentation a deux défauts :

**I.A.5.** Pourquoi le nombre zéro pose-t-il problème ?

**I.A.6.** Combien de nombres entiers relatifs peuvent être représentés en binaire sur 8 bits par cette méthode ? Comparer avec le nombre d'entier naturels pouvant être représentés sur 8 bits.

**I.A.7.** Effectuer l'addition **sur 8 bits** des représentations binaires de 80 et de -60. Que remarque-t-on ?

### B. Méthode du complément à 2

La méthode du complément à 2 est la suivante :

Le codage d'un entier relatif *n* **sur 8 bits** se fait par la méthode du complément à 2 comme suit :

- Si $0 \leq n \leq 127$, alors *n* est représenté par son codage en base 2.
- Si $-128 \leq n < 0$, alors *n* est représenté par la représentation binaire du nombre $n + 256$.
- Si $n > 127$ ou si $n < -128$, alors on ne peut pas représenter ainsi *n* en binaire.

**I-B.1.** En suivant cette méthode, donner les représentations binaires des nombres décimaux suivants.

**a.** 80 **b.** -1 **c.** -127 **d.** -32 **e.** -60 **f.** -200

**I-B.2.** En suivant cette méthode, donner les représentations décimales des nombres binaires sur 8 bits suivants.

**a.** $\overline{01101101}^2$ **b.** $\overline{11101101}^2$ **c.** $\overline{01110010}^2$ **d.** $\overline{00110010}^2$ **e.** $\overline{11100101}^2$ **f.** $\overline{10100111}^2$

Les problèmes posés par la méthode naïve sont alors résolus.

**I-B.3.** Combien y-a-t-il de représentations du nombre zéro ?

**I-B.4.** Combien de nombres entiers relatifs peuvent être représentés en binaire sur 8 bits par cette méthode ? Comparer avec le nombre d'entier naturels pouvant être représentés sur 8 bits.

**I-B.5.** Faire l'addition des représentations binaires des nombres 80 et -60 **sur 8 bits**. Que remarque-t-on par rapport à la méthode naïve ?

**I-B.6.** Effectuer l'addition **sur 8 bits** des représentations binaires de 114 et -89, puis de 27 et -82. Vérifier le résultat en décimal.

***Note** : Une soustraction binaire correspond donc à l'addition d'entier relatifs !*

**I-B.7.** Quels sont les nombres entiers encodables par la méthode des compléments à 2 sur 16 bits ? Sur 32 bits ? Sur *N* bits, où *N* est un entier naturel quelconque ?

## Partie II : Nombres flottants

En notation décimale, les chiffres à gauche de la virgule représentent des unités, des dizaines, des centaines, etc. et ceux à droite de la virgule, des dixièmes, des centièmes, des millièmes, etc…

*Exemple : Le nombre décimal 41,863*

$26\ 541{,}863 = 4\times10^{1} + 1\times10^{0} + 8\times10^{-1} + 6\times10^{-2} + 3\times10^{-3}$

Mathématiquement, on peut considérer une technique similaire en binaire : les chiffres à droite de la virgule représentent des demis ($2^{-1}$), des quarts ($2^{-2}$), des huitièmes ($2^{-3}$), des seizièmes ($2^{-4}$), etc…

*Exemple : Le nombre décimal 1,25*

$1{,}25 = 1 + \frac{1}{4} = 1\times2^{0} + 0\times2^{-1} + 1\times2^{-2} = \overline{1{,}01}^2$

**II.1.** En suivant cette méthode, donner les représentations décimales des nombres binaires suivants.

**a.** $\overline{0{,}1}^2$ **b.** $\overline{11{,}101}^2$ **c.** $\overline{0{,}10101}^2$

**II.2.** En suivant cette méthode, donner les représentations binaires des nombres décimaux suivants.

**a.** 1,125 **b.** 2,75 **c.** 5,3125

En informatique, cette méthode n'est pas applicable : on ne peut pas représenter la virgule ! Il faut une représentation n'utilisant que des 0 et des 1.
On utilise alors la représentation similaire à la notation scientifique, avec un signe, un exposant de puissance de 2 et une mantisse.

*Rappel : Notation scientifique en binaire*

*Pour écrire le nombre décimal 12345 :*

- *Il est positif donc son signe est « + ».*
- *De plus, comme $2^{13} \leq 12345 \leq 2^{14}$, l'exposant sera 13. C'est l'ordre de grandeur du nombre.*
- *Enfin, le nombre multiplicateur m est tel que $m \times 2^{13} = 12345$.*

*Ainsi, $m = \frac{12345}{2^{13}} = 1{,}506958008$.*

*En définitive, $12345 = 1{,}506958008 \times 2^{13}$*

Pour représenter les nombres le plus précisément possible, il faut utiliser plusieurs octets (un seul ne suffit pas). Nous nous baserons ici sur une représentation sur 32 bits (4 octets).

Un nombre est donc représenté sous la forme : $s.m.2^n$ où :

- s représente le signe du nombre (+ ou -)
- n est son exposant, un entier relatif
- m est sa mantisse, un nombre à virgule compris entre 1 inclus et 2 exclu.

Lorsqu'on utilise 32 bits pour représenter un nombre à virgule, on utilise 1 bit pour le signe, 8 bits pour l'exposant et 23 bits pour la mantisse.

- Le signe + est représenté par 0 et le signe - par 1.
- L'exposant n est un entier relatif compris entre -128 et 127 ; on le représente par la méthode du complément à 2.
- La mantisse m est un nombre binaire à virgule compris entre 1 inclus et 2 exclu, comprenant 23 symboles après la virgule.

  Comme cette mantisse est comprise entre 1 et 2, elle a toujours un seul chiffre avant la virgule et ce chiffre est toujours un 1 ; il est donc inutile de le représenter et on utilise les 23 bits pour représenter les 23 symboles après la virgule.

*Exemple : Le nombre flottant 11010011010010011110000111000000*

*On a :* $\underbrace{1}_{s}\ \overbrace{10100110}^{n \text{ ou } n+256}\ \underbrace{10010011110000111000000}_{\text{partie flottante de m}}$

- *Le premier bit est 1. Ainsi, ce nombre est négatif.*
- *Les 8 bits suivants sont 10100110. Ceci est la représentation binaire du nombre 166 = n+256.*

*L'exposant est donc n = 166 -256 = -90.*

- *La mantisse est le nombre binaire à virgule :*

$$m = 1,\underbrace{10010011110000111000000}_{\text{23 derniers bits}}$$

*On a donc* $m = 1 + \frac{1}{2} + \frac{1}{2^4} + \frac{1}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{1}{2^{10}} + \frac{1}{2^{15}} + \frac{1}{2^{16}} + \frac{1}{2^{17}} = 1{,}57720184$

*Le nombre représenté est donc* $-1{,}57720184 \times 2^{-90} = -1{,}2741 \times 10^{-27}$

**II.3.** En suivant cette méthode, trouver les nombres à virgule représenté par les mots :

**a.** 0 00100011 11010011100101011000000 **b.** 1 11001100 01010011000101000000000

**II.4.** Comment est représenté le nombre à virgule $2^{-128}$ ?

**II.5.** Représenter 9,5 sur 32 bits.

**Ce qu'il faut retenir**

### ❖ Entiers relatifs par la méthode du complément à 2

Le codage d'un entier relatif $n$ sur $N$ bits, où $N$ est un entier naturel quelconque, se fait par la méthode du complément à 2 comme suit :

- Si $0 \leq n \leq 2^{N-1} - 1$, alors $n$ est représenté par son codage en base 2.
- Si $-2^N \leq n < 0$, alors $n$ est représenté par la représentation binaire du nombre $n + 2^N$.
- Si $n > 2^{N-1} - 1$ ou si $n < -2^N$, alors on ne peut pas représenter ainsi $n$ en binaire.

Cette méthode permet de coder les entiers relatifs compris entre $-2^{N-1}$ et $2^{N-1} - 1$, soit un total de $2^N$ nombres (zéro inclus), ainsi que de procéder à des soustractions (en additionnant un entier positif et un entier négatif).

### ❖ Nombre flottant en binaire (mathématiquement)

Mathématiquement, en binaire, les chiffres à droite de la virgule représentent des demis ($2^{-1}$), des quarts ($2^{-2}$), des huitièmes ($2^{-3}$), des seizièmes ($2^{-4}$), etc…

**Savoir faire**

### ❖ Conversion décimale-binaire et binaire décimale des entiers relatifs

### ❖ Conversion d'un mot informatique de 32 bits en nombre flottant

Un nombre est donc représenté sous la forme : $s.m.2^n$ où :

- s représente le signe du nombre (+ ou -)
- n est son exposant, un entier relatif
- m est sa mantisse, un nombre à virgule compris entre 1 inclus et 2 exclu.

Lorsqu'on utilise 32 bits pour représenter un nombre à virgule, on utilise 1 bit pour le signe, 8 bits pour l'exposant et 23 bits pour la mantisse.

- Le signe + est représenté par 0 et le signe - par 1.
- L'exposant n est un entier relatif compris entre -128 et 127 ; on le représente par la méthode du complément à 2.
- La mantisse m est un nombre binaire à virgule compris entre 1 inclus et 2 exclu, comprenant 23 symboles après la virgule.

Comme cette mantisse est comprise entre 1 et 2, elle a toujours un seul chiffre avant la virgule et ce chiffre est toujours un 1 ; il est donc inutile de le représenter et on utilise les 23 bits pour représenter les 23 symboles après la virgule.