Nous avons vu comment représenter des nombres entiers naturels (en base 2,10 et 16). Nous allons maintenant apprendre à représenter les entiers qu'ils soient positifs ou négatifs : les entiers relatifs. L'ensemble des nombres binaires entiers relatifs se nomme **les nombres binaires signés**.

# Représentation des entiers relatifs (positifs ou négatifs)

## Approche "naive"

La première idée qui pourrait venir à l'esprit pour représenter un entier négatif serait d'utiliser un bit pour indiquer si on a affaire à un nombre positif ou négatif (0 sur le bit de poids fort pour les nombres positifs, et 1 pour les négatifs), puis avec le reste des bits on code la valeur absolue comme vue précédemment.

Cette méthode présenterait 2 problèmes :

* Deux représentations possibles pour 0 , exemple sur 8 bits : 00000000 ou 10000000
* Cette notation ne respecterait pas le principe de l'addition : réalisons l'opération 23+ (-23) sur un registre 8 bits :

23 = 16+4+2+1= et  −23=

Une image contenant texte

Description générée automatiquement

## Complément à deux : encodage vers binaire

Cette méthode permet de remédier aux problèmes évoqués ci-dessus.

On utilise toujours un bit de signe tout à gauche:



**Exemple : représentation en complément à 2 de -61 sur 8 bits**

* **Méthode 1 :**  
  a) On calcule −61+28=   
  b) On code 195 en binaire "classiquement" : 195 =
* **Méthode 2 :**  
  a) On code sur 8 bits la valeur absolue du nombre à coder :

|−61| =

b) On inverse chaque bit (complément à 1) :    
c) On ajoute 1 : −61=

* **Méthode 3 : rapide**

a) On calcule 60 = 32 +16 + 8 + 4 + 1 =

b) On garde tous les zéro de droite jusqu'au premier

c) Pour le reste on inverse tous les 1 et tous les 0

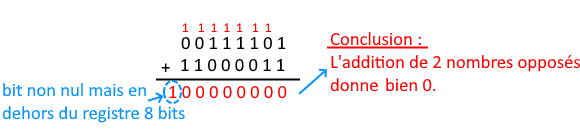
-61 =

Exemple : Pour 60 on a et pour -60 on a

**Point important :** on remarque que le résultat est bien un nombre négatif car son bit de poids fort (le premier) vaut 1.

**Remarques :**

* On peut vérifier que l'addition d'un nombre et de son opposé en complément à 2 donne bien 0 :



* On peut évidemment utiliser la notation hexadécimale pour les nombres binaires signés :

−61 = A3(16)

**Écrire les nombres suivants en binaire complément à deux :**

-87

-42

**Exercice :**

Implanter en Python dans la fichier ***conversion\_dec\_bin\_comp2*** la fonction donnant le complément à 2 d'un nombre, ceci sur n bits (vous pourrez réutiliser les fonctions vus dans le chapitre précédent. (Vous écrierez les préconditions)

## Complément à deux : décodage

Pour trouver à quel nombre décimal correspond un nombre binaire signé négatif, on applique les méthodes précédentes "à l'envers".

Exemple avec 10101010(2)

**Méthode 1 :**  
a) On traduit le nombre binaire en décimal comme s'il était non signé :

10101010(2) =

b) On lui soustrait 2n (n nombre de bits sur lequel est codé le nombre binaire signé :

170−28 =

**Méthode 2 :(moins simple)**  
a) On soustrait 1 : 10101010(2)−1 =

b) On inverse les bits (compléments à 1) :

c) On convertit en base 10 et on ajoute le signe - :

Déterminer la valeur décimal de ce code binaire en complément à deux

11011100

1101

10001000

**Exercice :**

Implanter la fonction de décodage d'un nombre binaire signé dans le fichier : ***conversion\_bin\_dec\_comp2***

## Valeur maximale

Pour un nombre binaire signé sur n bits. La valeur maximale est constitué de n-1 bits à 1, donc :

*Nmax* = 2n−1−1

La valeur minimale est le nombre constitué d'un 1 (bit de signe négatif) puis que des 0, soit :

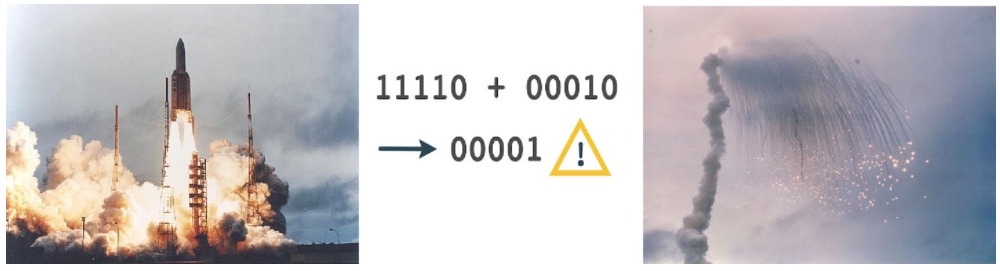
Nmin =−2n−1

Ainsi, un nombre binaire signé sur 8 bits sera compris entre −27 = −128−27 =−128 et 27−1 = 127.

**Anecdote :**

Le 4 juin 1996, la toute nouvelle Ariane est détruite lors de son vol inaugural, ceci à cause d'un bug informatique d'overflow ( dépassement de capacité).

Afin de gagner du temps (et donc de l'argent), le logiciel permettant notamment la mesure de l'accélération horizontale de la fusée avait été reprise de la fusée ariane IV. La valeur de l'accélération (binaire signé) y était stockée dans un registre 8 bits. Le problème est qu'Ariane V étant bien plus puissante que Ariane IV, la valeur d'accélération était plus importante et dépassait les capacités de stockage du registre 8 bits. Il en résulta une valeur stockée absurde par overflow qui dévia la fusée de son orientation normale et entrainera son autodestruction. Destruction des quatre satellites d'une valeur de 370 millions de dollars.



Dans le même style de bug, le boeing 787 dreamliner doit être rebooté complétement tous les 248 jours à cause d'un compteur 32 bits.