REPRESENTATION DES NOMBRES NEGATIFS EN BINAIRE

Préambule :

Sur une machine, on ne peut stocker que des « 0 » et des « 1 », il n'est donc **pas possible de stocker un signe « - «** comme on utilise en maths. Il fallait donc trouver une « astuce »

L'**astuce** se base sur une particularité du stockage des infos numériques : **les nombres sont stockés sur un format précis** (8 bits, 16 bits, ….) et ne sont pas extensibles !

Ainsi, sur 8 bits, 11111111 + 1 = 00000000 ; le « 1 » qui devrait se trouver en 9éme bit ne peut pas apparaitre : il y a **débordement** !

La où ca devient drôle : si je suis ce même raisonnement : si je retranche « 1 » à « 00000000 », j'obtiens … 11111111 !! (exactement comme si je faisais « -1 » à un compteur kilométrique de voiture à 0 , j'obtiendrais « 999999km » à la place de « -1 km »)

Preuve que cela « fonctionne » : 11111111 + 1 = 00000000 tout comme -1 + 1 = 00000000
je peux donc dire que 11111111 = -1 !

On dira alors que le nombre est **signé**

Définition : Si on admet que le nombre peut représenter des valeurs négatives, on parle de nombres "signés".

Dans cette représentation des nombres négatifs, le MSB renseigne directement sur le signe du nombre :

1 = signe moins

0 = signe plus

Exemples :

11111111 en nombre non signé = 255
11111111 en nombre signé = -1
11111110 en nombre signé = -2
11111110 en nombre non signé = 254
11111101 en nombre signé = -3
11111101 en nombre non signé = 253

00000010 en nombre non signé = 2
00000010 en nombre signé = 2 aussi
10000000 en nombre signé = -128

Il y a donc moyen de représenter :

- 128 codes avec le bit de signe à 1
ce sont 128 nombres négatifs (de –1 à – 128)

- 128 codes avec le bit de signe à 0
le nombre 0 et 127 nombres positifs (de 0 à +127)

Détermination du code du nombre négatif en binaire :

On utilise ce qu'on appelle le « **complément à deux** » , qui se détermine à partir du « complément à 1 »

Complément à 1 : trop simple : il suffit d'inverser la valeur de chaque bits

Le complément à 1 de 00000000 est 11111111 ; le complément à 1 de 11110000 est 00001111 ; le complément à 1 de 10101100 est 01010011

Complément à 2 : presque aussi simple :

(complément à 2) = (complément à 1) + 1

Exemple :

Représenter – 7 en binaire signé sur 8 bits :

7 = 00000111

Complément à 1 : 11111000

Complément à 2 : 11111001 ; -7 s'écrit 11111001

Vérification : vérifions que – 7 + 7 = 0 :

```
  11111001
+ 00000111
----------
```

Exercice : représenter -14 en binaire signé sur 8 bits

14 = 0b00001110

-14 = 0b11110010 = 0xF2

Exercice inverse :

Donner la valeur de 0xF5 sachant que ce code est en binaire signé sur 8 bits :

Extension de la taille d'un nombre signé

Pour étendre la taille d'un nombre non signé, on ajoute des 0 à sa gauche.
Pour étendre la taille d'un nombre signé, on ajoute sur la gauche des bits identiques au bit de signe.
Exemples :
-4 code sur un octet = 0xFCsur deux octets ce code devient 0xFFFC
0xFC= 0b1111 1100≡ 0xFFFC= 0b 1111 1111 1111 1100
de même :
+ 4 en un octet = 0x04 sur deux octets = $0x0004_{(16)}$
0x04 = 0b 0000 0100 = 0x0004 = 0b 0000 0000 0000 $0100_{(2)}$