TP2 – Représentation des entiers naturels relatifs - correction

Nous avons déjà vu comment représenter les entiers positifs, nous allons maintenant nous intéresser aux entiers relatifs.
La première idée qui pourrait nous venir à l'esprit est, sur un nombre comportant n bits, d'utiliser 1 bit pour représenter le signe et n-1 bits pour représenter la valeur absolue du nombre à représenter. Le bit de signe étant le bit dit "de poids fort" (c'est à dire le bit le plus à gauche), ce bit de poids fort serait à 0 dans le cas d'un nombre positif et à 1 dans le cas d'un nombre négatif.
Un exemple : on représente l'entier 5 sur 8 bits par 00000101, -5 serait donc représenté par 10000101 .
Il existe un énorme inconvénient à cette méthode : l'existence de deux zéros, un zéro positif (00000000) et un zéro négatif (10000000) !
Il existe un autre inconvénient que vous allez vérifier maintenant.
Effectuez l'opération suivante : 5 + (-5) et vérifiez le résultat.

$$\mathbf{5 + (-5) = (00000101)_2 + (10000101)_2 = (10001010)_2 = (-10)_{10}}$$

Nous allons donc devoir utiliser une autre méthode : le complément à deux.

Le complément à deux

Avant de représenter un entier relatif, il est nécessaire de définir le nombre de bits qui seront utilisés pour cette représentation (souvent 8, 16 , 32 ou 64 bits)
Prenons tout de suite un exemple : déterminons la représentation de -12 sur 8 bits.

- Commençons par représenter 12 sur 8 bits (sachant que pour représenter 12 en binaire seuls 4 bits sont nécessaire, les 4 bits les plus à gauche seront à 0:
  $(12)_{10} = (00001100)_2$
- Inversons tous les bits (les bits à 1 passent à 0 et vice versa) : 11110011
- Ajoutons 1 au nombre obtenu à l'étape précédente : les retenues sont notées en rouge

```
    11
  11110011
+ 00000001
----------
  11110100
```

- La représentation de -12 sur 8 bits est donc : 11110100

Comment peut-on être sûr que 11110100 est bien la représentation de -12 ?
Nous pouvons affirmer sans trop de risque de nous tromper que 12 + (-12) = 0, vérifions que cela est vrai pour notre représentation sur 8 bits.

```
   11111
+ 00001100
  11110100
----------
  00000000
```

Dans l'opération ci-dessus, nous avons un 1 pour le 9e bit, mais comme notre représentation se limite à 8 bits, il nous reste bien 00000000.

Cette opération, qui permet d'écrire un nombre négatif, correspond au calcul de $2^n - |x|$,
où n est la longueur de la représentation et |x| la valeur absolue du nombre à coder.
Ainsi, −1 s'écrit comme $256-1 = 255 = 11111111_2$, pour les nombres sur 8 bits.

Ceci est à l'origine du nom de cette opération : " complément à 2 puissance n ", quasi-systématiquement tronqué en " complément à 2 ".
Exemple avec -12 : 256 – 12 = 244 → $(11110100)_2$ qui correspond aussi à -12 !

Exercices.
a) En utilisant le complément à 2, représentez -15 (représentation sur 8 bits).

$(15)_{10}$ = $(00001111)_2$; inversion : 11110000 ; ajout de 1 : 11110001

$(-15)_{10}$ = $(11110001)_2$

a bis) En utilisant le complément à 2, représentez -27 (représentation sur 8 bits).

$(27)_{10}$ → $(00011011)_2$; inversion → 11100100 ; ajout de 1 → 11100101

donc $(11100101)_2$ → $(-27)_{10}$

Remarque :
Il faut noter qu'il est facile de déterminer si une représentation correspond à un entier positif ou un entier négatif : si le bit de poids fort est à 1, nous avons affaire à un entier négatif, si le bit de poids fort est à 0, nous avons affaire à un entier positif.

b) Représentez sur 8 bits l'entier 4 puis représentez, toujours sur 8 bits, l'entier -5. Additionnez ces 2 nombres (en utilisant les représentations binaires bien évidemment), vérifiez que vous obtenez bien -1.

$(4)_{10}$ → $(00000100)_2$; $(-5)_{10}$ → 00000101 ; inversion → 11111010 ; ajout de 1 → $(11111011)_2$
4 +(- 5) = -1 ; 00000100 + 11111011 = 11111111 (normalement c'est -1)
vérification : 11111111 ; retrait de 1 → 11111110 ; inversion → $(00000001)_2$ = $(1)_{10}$

b bis) Représentez sur 8 bits l'entier 8 puis représentez, toujours sur 8 bits, l'entier -15. Additionnez ces 2 nombres (en utilisant les représentations binaires bien évidemment), vérifiez que vous obtenez bien -7.

13 + (-29) = -16 ; -29 → 00011101 puis inversion : 11100010 et ajout de 1 : 11100011
$(00001101)_2$ + $(11100011)_2$ = $(11110000)_2$ est-ce égal à -16 ?
Vérification : chemin contraire ;
enlever 1 : 11110000 – 1 = 11101111
inversion : 00010000
et $(00010000)_2$ → $(16)_{10}$

c) Quel est le plus petit entier négatif que l'on peut représenter sur 8 bits ?

$(10000000)_2$ = -128
Vérification :
(10000000 – 1 = 011111111 ; inversion : 10000000 –> 128
$(10000000)_2$ → $(-128)_{10}$

d) Quel est le plus grand entier positif que l'on peut représenter sur 8 bits ?

$(01111111)_2$ = $(+127)_{10}$

Remarque :
Plus généralement, nous pouvons dire que pour une représentation sur n bits, il sera possible de coder des valeurs comprises entre -2^{n-1} et $+2^{n-1} - 1$.

e) Quelles sont les bornes inférieure et supérieure d'un entier relatif codé sur 16 bits ?

-2^{16-1} = - 32768 **$2^{16-1} - 1$ = 32767**

Annexe.
Article wikipédia sur le complément à deux :
https://fr.wikipedia.org/wiki/Compl%C3%A9ment_%C3%A0_deux