REPRÉSENTATION DES NOMBRES ENTIERS RELATIFS EN INFORMATIQUE

1ère (*fausse bonne*) idée :

Problème n°1 : que représentent avec cette convention les nombres binaires suivants ?

- 00000000_2 = **+ 0_{10}**
- 10000000_2 = **– 0_{10}** → **2 représentations différentes pour la même valeur (0), on n'aime pas ça en informatique !!!**

Problème n°2 : dans les systèmes informatiques, pour des questions de coût et de simplicité, c'est le même circuit électronique (appelé « additionneur ») qui réalise aussi bien les additions que les soustractions ; en effet, soustraire deux nombres équivaut à *additionner l'un à l'opposé de l'autre* :

$$a - b = a + (- b)$$

Avec la convention suivante sur le bit de signe, que donnerait alors l'opération suivante ?

```
                 1 1
    0 0 0 0 0 1 0 1₂    ( = 5₁₀ )
+   1 0 0 0 0 0 1 1₂    ( = – 3₁₀ )
-----------------------
    1 0 0 0 1 0 0 0₂    ( = 2₁₀ ? )
```

→ **non, cela donne – 8_{10} … Problème !!!**

2. La solution : le complément à deux

1. En complément à deux, quel est maintenant la notation de « – 0 » ? Que constate-t-on alors ?

+ 0_{10} = 00000000_2
→ représentation de « – 0 » : 11111111_2
+ 1
=(1)00000000_2 ***(on laisse de côté la dernière retenue)***
→ **identique, donc une seule représentation pour le nombre 0.**

2. Qu'obtient-on si l'on fait l'addition 5 + (– 5) ?

$5_{10} = 00000101_2$
→ **–** $5_{10} = 11111011_2$

donc : 00000101_2
+ 11111011_2
= 00000000_2 → **le résultat est bien égal à 0, ça marche !!**

3. Calculer de même la valeur de la différence 5 – 3 :

$3_{10} = 00000011_2$ → **–** $3_{10} = 11111101_2$

donc : $5_{10} + (- 3_{10}) = 00000101_2$
+ 11111101_2
= **(1)00000010_2** ***(on laisse de côté la dernière retenue)***
=+ 2_{10} **OK !**

III. Applications

1. Donner les représentations en binaire sur 8 bits des nombres suivants en complément à 2 :

- 58_{10} = $0011\ 1010_2$
- $– 1_{10}$ = $1111\ 1111_2$
- $– 127_{10}$ = $1000\ 0001_2$

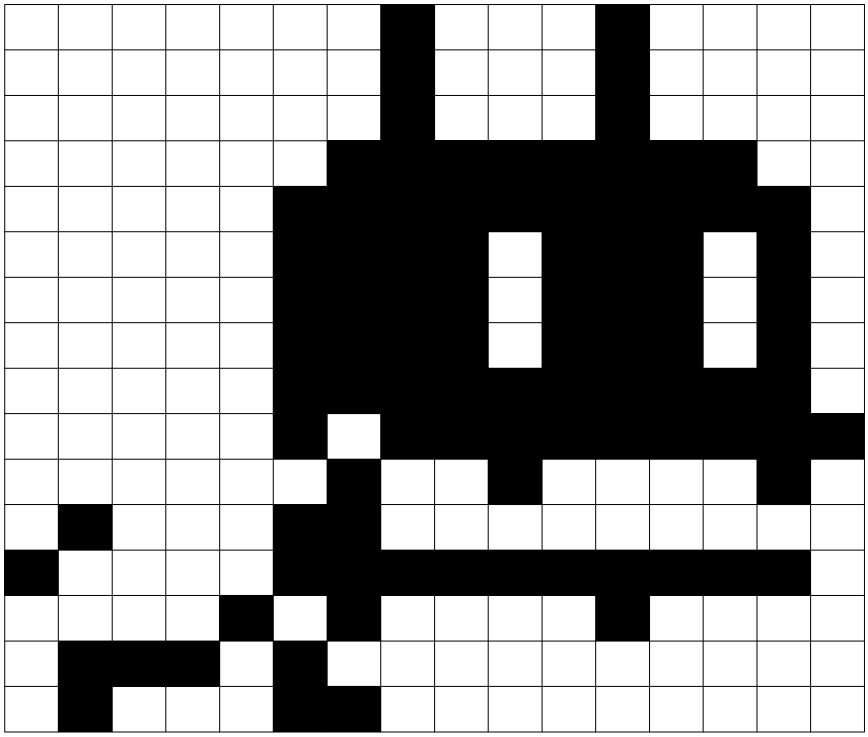
2. Indiquer :
   - la valeur entière la plus grande que l'on peut coder sur 8 bits en utilisant le complément à 2
   - la valeur entière la plus petite

valeur la plus grande	01111111_2 **= +127_{10}**
valeur la plus petite	10000000_2 **= – 128_{10}**

→ **en complément à deux, on ne peut donc coder que des nombres de valeur absolue deux fois plus petite qu'avec des entiers non signés (*0 à 255 par exemple sur 8 bits*).**

3. Et pour terminer sur la représentation des entiers naturels et relatifs...

L'objectif de cet exercice est de dessiner une image matricielle dans le quadrillage 16x16 ci-dessous grâce aux réponses aux questions de conversions entre les bases numériques.
Chaque case de l'image correspond à un bit. Une ligne de l'image fait 16 cases, soit 16 bits, soit 2 octets.
   - Lorsque le nombre est négatif, on considérera le binaire complément à 2.
   - Lorsque le bit est à 1, la case est noire, lorsque le bit est à 0, la case est blanche.



L1-L2-L3 = 272_{10} = 1 0001 0000_2
L4 = $3FC_{16}$ = 11 1111 1100_2
L5-L9 = $3D8_{16}$ + 426_{16} = $7FE_{16}$ = 11111111110_2
L6-L7-L8 = 1978_{10} = 11110111010_2
L10 = 2^{15} – $7A01_{16}$ = 8000_{16} – $7A01_{16}$ = $5FF_{16}$ = 10111111111_2
L11 = 100 1000010_2
L12-L16 = 100_{16} x 46_{16} = 4600_{16} = 100 0110 0000 0000_2
L13 = 1000 0111 1111 1110_2
L14 = 1010 0001 0000_2
L15 = 111 0010 0000 0000_2