Codage des entiers relatifs

1 Introduction

Les **entiers relatifs** sont les nombres entiers positifs et négatifs.

Pour les coder, une **première idée** serait d'utiliser **un bit pour le signe** (le bit de poids fort) et les autres bits pour le nombre. Le bit du signe sera 0 si le nombre est positif et 1 si le nombre est négatif.

Remarque : Le **bit de poids fort** est celui correspondant à la puissance de 2 la plus élevée, celui de **poids faible** est celui qui correspond à 2^0.

Exemple :
Coder le nombre -17 sur 1 octet (8 bits) :

$17 = 10001_2 = 0010001_2$ sur 7 bits (on complète avec des zéros pour avoir 8 bits.)
$-17 = 10010001_2$: le dernier bit est pour le signe ; ici 1 car le nombre est négatif.

Mais cet encodage **pose problèmes** :

- il y a deux zéros, un négatif et un positif.
  Par exemple : sur 4 bits, 1000 et 0000 codent -0 et $+0$;

- ensuite cela complique les opérations arithmétiques.
  Par exemple : additionner 3 et -2, codés sur 4 bits, ne donne pas 1 !

$$\begin{array}{ccccccc} & 0 & 0 & 1 & 1 & \rightarrow & 3 \\ + & 1 & 0 & 1 & 0 & \rightarrow & -2 \\ \hline & 1 & 1 & 0 & 1 & \rightarrow & -5 \end{array}$$

2 Complément à 2^n

On va utiliser la méthode par **complément à** 2^n, où n est le nombre de bits choisis pour coder les nombres relatifs (on dit aussi complément à 2 ou C2).

Pour l'entier x :

- si l'entier x est **positif**, on le représente par son code en binaire ;

- si l'entier x est **négatif**, on le représente par l'entier $x + 2^n$, codé en binaire.

En fonction du nombre n de bits utilisés pour le codage on peut représenter **les entiers relatifs de** -2^{n-1} **à** $2^{n-1} - 1$.

Remarques :
- le **bit de poids fort** représente toujours le **signe de l'entier** :
  0 s'il est positif ; 1 s'il est négatif ;

- avec cet encodage, on peut représenter sur 8 bits les entiers compris entre $-128 = -2^7$ et $127 = 2^7 - 1$; sur 16 bits les entiers compris entre -32768 et 32767.

Exemple : Codage des entiers relatifs sur 4 bits.
On peut coder les entiers de -8 à 7. On obtient le tableau suivant :

nombre entier x	codage en C2
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111

Remarque :

Avec cet encodage, on peut additionner deux entiers relatifs sans se soucier de leurs signes. On ne tiendra pas compte de la retenue finale.

Exemple : additionner 3 et -2 codés en complément à 2^4 :

$$\begin{array}{cccccccc} & 0 & 0 & 1 & 1 & \rightarrow & 3 \\ + & 1 & 1 & 1 & 0 & \rightarrow & -2 \\ \hline (1) & 0 & 0 & 0 & 1 & \rightarrow & 1 \end{array}$$

Exercice 1 : Coder les nombres -10, -42 et 97 en complément à 2^8, c'est à dire en C2 sur 8 bits.

(11110110 ; 11010110 ; 01100001)

Methodes :

• Convertir un entier relatif sous forme décimale en C2 :

- s'il est positif, on le code en binaire ;
- sinon : on code la valeur absolue du nombre en binaire (le nombre sans son signe) ;
  puis, on complète avec des zéros à gauche si nécessaire pour obtenir le nombre de bits souhaités ;
  on inverse les bits : les zéros deviennent des 1 et les 1 des zéros ;
  enfin on ajoute 1 (sans tenir compte de la retenue finale).

Exercice 2 : Coder les nombres -42, 27 et -128 en C2 sur 8 bits avec cette méthode.

(11010110 ; 00011011 ; 10000000)

• Convertir un nombre codé en C2 en nombre décimal :

- si le bit de poids fort est 0, on convertit le nombre de façon classique ;
- si le bit de poids fort est 1, on inverse les bits ; puis on ajoute 1, et on convertit le nombre obtenu en décimal ; enfin on ajoute le signe $-$ devant le nombre obtenu.

Exercice 3 : Convertir les nombres suivants codés en C2, en décimal : 10011010_2 et 01111110_2.

(-102 ; 126)