Le but du TD est de comprendre comment on code un entier relatif dans un ordinateur, sachant qu'il ne peut y avoir que des nombres binaires sans signe. On appelle les entiers relatifs des *entiers signés*, et les entiers naturels sont appelés *entiers non signés*.

« Il est quelle heure ?
- Il est 50.
- Moins 10 ?
- Oui, c'est ça et le cours commence à moins 5.
- Tu veux dire 55 ?
- Oui. »

Le mécanisme pour représenter les entiers négatifs est exactement le même que pour les minutes, si on ne tient pas compte des heures. On peut indiquer soit un nombre négatif, soit son complément à 60. Par exemple, « moins une » c'est aussi 59, c'est à dire $60 + (-1) = 59$.

La différence avec le codage des entiers sur machine, c'est que la référence n'est pas 60 mais une puissance de 2.

1. Représentation graphique

La figure 1 montre une roue qui représente tous les entiers codés sur 4 bits. Elle permet de comprendre la convention de représentation des entiers relatifs.

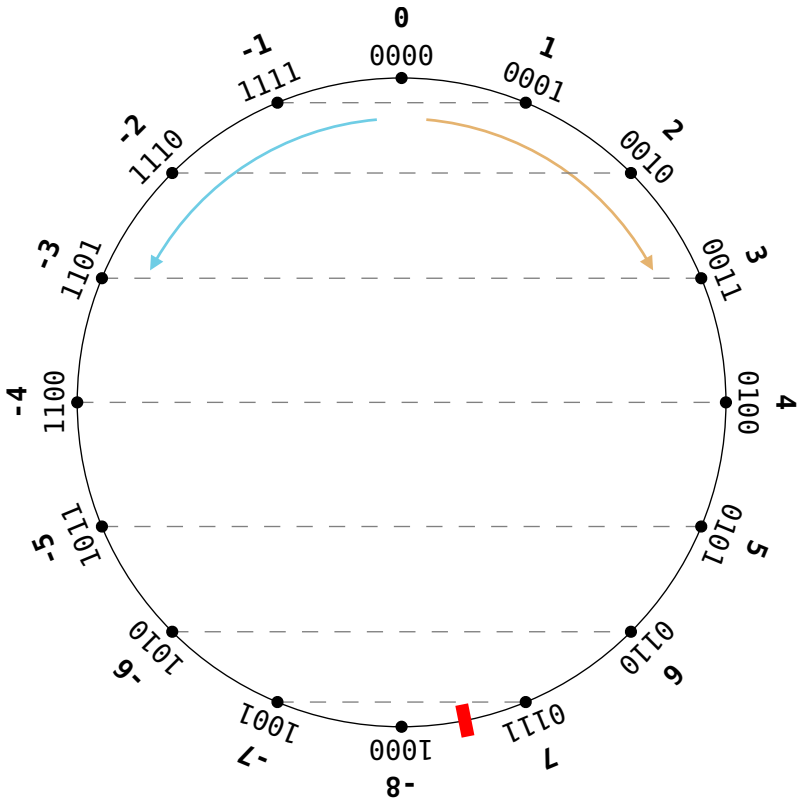

Figure 1: Roue des nombres relatifs

Par exemple, -2 est codé $(1110)_2$ sur 4 bits. Comparez avec le tableau des 16 premiers entiers écrits en base 2 sur 4 bits.

a. Sans aucun calcul, seulement en regardant la roue des nombres, écrivez les codages de -1, de -2 et -3 sur **8 bits**.
b. Des lignes horizontales ont été tracées entre les valeurs opposées, par exemple -3 et 3. Additionnez les deux nombres en base 2 qui sont de part et d'autre de ces lignes, par exemple $(1011)_2$ et $(0101)_2$. Quelle est la valeur obtenue à chaque fois ? À quoi ça correspondrait dans l'exemple initial des minutes, d'additionner par exemple $+5$ et l'autre dénomination de -5 (55) ?

Vous devez comprendre que 16, dans cette roue sur 4 bits, est assimilé à 0, de même que 60 à 0 pour les minutes. Quand on généralise à un codage sur n bits, la valeur 2^n est assimilée à 0, parce que 2^n s'écrit 1000...000 sur $n+1$ bits, et quand on tronque à n bits de poids faible, alors il ne reste que n zéros.

Le codage d'un entier relatif N sur n bits est $(2^n + N)$ écrit en base 2 et tronqué à n bits.

Vous constatez qu'il y a un conflit avec le code `1000` tout en bas. Il pourrait représenter à la fois $+8$ et -8. Le choix a été fait de considérer que c'est uniquement -8. Donc $+8$ ne peut pas être codé. Les nombres codables sur 4 bits vont de -8 à $+7$.

Le signe d'un entier codé avec cette convention se lit dans le bit de poids fort : $0xxx$ => positif ou nul, $\quad 1xxx$ => négatif.

Avec cette convention, 0 est considéré comme étant positif.

c. Quels sont les signes des nombres suivants, codés avec cette convention sur 8 bits (ne cherchez pas leur valeur) ?
   i. $(10101110)_2$
   ii. $(01101011)_2$
   iii. $(10111010)_2$

d. Pour les additions suivantes, effectuer le calcul en base 2 sur 4 bits, puis utilisez la roue pour convertir nombres et résultat en base 10 relatifs et vérifier l'addition.
   i. $(0101)_2 + (0001)_2$
   ii. $(1101)_2 + (1101)_2$
   iii. $(0110)_2 + (1110)_2$
   iv. $(1100)_2 + (1011)_2$

On constate avec ces exemples que la somme des codages est égale au codage de la somme, sauf en cas de dépassement.

Il y a dépassement quand on additionne deux nombres de même signe et que le résultat est d'un signe différent.

2. Convention $C2^n$

Les exercices précédents vous ont fait découvrir la convention $C2^n$ pour représenter des nombres relatifs sur n bits. On dit souvent « en complément à deux », mais le nom entier de cette convention est « codage en complément à 2^n », parce que le zéro est assimilé à 2^n et qu'un nombre N est codé par $(2^n + N)$ tronqué à n bits.

Voici le principe : on code N par $(2^n + N)$ et on garde les n bits de poids faible. Par exemple, $+3$ est codé sur 4 bits par $2^n + 3 = 2^4 + 3 = 16 + 3 = 19 = (10011)_2$ ce qui donne $(0011)_2$. Et -4 est

codé par $2^n + -4 = 16 - 4 = 12 = (1100)_2$.

Heureusement il y a des simplifications :

- Si N est positif ou nul, alors N est codé directement sur n bits (si c'est possible). Par exemple, 6 est codé $(0110)_2$ sur 4 bits et $(00000110)_2$ sur 8 bits, et 9 ne peut pas être codé sur 4 bits.
- Si N est négatif, alors N est codé par $CHS_n(-N)$.

CHS_n est une fonction qui **CH**ange le **S**igne d'un nombre codé sur n bits. Plus exactement, elle calcule le codage de l'opposé du nombre codé sur n bits qu'on lui fournit. C'est la fonction qui fait correspondre les deux nombres reliés par une ligne horizontale dans la roue des nombres. C'est l'équivalent de la touche `+/-` des calculatrices.

Par exemple,

- $CHS_4(+5)$ retourne le codage de -5 sur 4 bits.
- $CHS_8(-7)$ retourne le codage de $+7$ sur 8 bits.

La fonction $CHS_n(N)$ fournit le nombre qui est relié à N par une ligne horizontale dans la roue des nombres. Par exemple, $CHS_4(0101) = 1011$, c'est à dire $CHS(+5) = -5$. Elle marche dans l'autre sens aussi. Par exemple, $CHS_4(1101) = 0011$, c'est à dire $CHS(-3) = +3$. Donc évidemment, $CHS_n(CHS_n(N)) = N$ pour tout entier représentable. Sur 4 bits, on ne peut pas représenter $+8$, donc $CHS_4(-8)$ ne peut pas être calculé.

Plus haut, il est écrit que si N est négatif, alors il est codé par $CHS_n(-N)$, c'est à dire le changement de signe de l'opposé de N. Oui, car N étant négatif, l'opposé de N est $-N$ et il est positif. Par exemple, si $N = -3$, alors $-N = +3$. Donc, on applique CHS_n à $-N$ pour obtenir le codage de N. Ce codage sera bien celui d'un entier négatif.

Voici maintenant comment on calcule cette fonction CHS_n. Il existe trois méthodes.

2.1. Méthode 1

On calcule $CHS_n(N) = 2^n - N$ et on garde les n bits de poids faible du résultat.

- Exemple : calculer $CHS_4(+3)$, c'est à dire obtenir le codage du nombre -3, en changeant le signe de $+3$:
  - $CHS_4(+3) = 2^4 - 3 = 16 - 3 = 13$
  - Le codage en base 2 de 13 est $(1101)_2$.
  - On garde seulement $(1101)_2$ et c'est ça le codage de -3. Vérifier sur la roue des nombres.
- Exemple : calculer $CHS_4(-3)$, c'est à dire obtenir le codage du nombre $+3$, en changeant le signe de -3 :
  - $CHS_4(-3) = 2^4 - (-3) = 16 + 3 = 19$
  - Le codage en base 2 de 19 est $(10011)_2$.
  - On garde seulement $(0011)_2$ et c'est ça le codage de $+3$.

La méthode 1 vient du fait que $codage_n(N) = 2^n + N$, et donc $CHS_n(N) = codage_n(-N) = 2^n - N$.

2.2. Méthode 2

On part du codage de N sur n bits. On inverse tous les bits et on additionne 1.

- Exemple : calculer $CHS_4(+4)$ pour obtenir le codage de -4 :
  - Le codage de $+4$ est $(0100)_2$.
  - On inverse tous les bits : 0 devient 1 et 1 devient 0, ça donne : $(1011)_2$.
  - On ajoute 1 au nombre inversé : $(1011)_2 + 1 = (1100)_2$, c'est ça le codage de -4.
- Exemple : calculer $CHS_4(-4)$ pour obtenir le codage de $+4$:
  - Le codage de -4 est $(1100)_2$.
  - On inverse tous les bits : $(0011)_2$.
  - On ajoute 1 au nombre inversé : $(0011)_2 + 1 = (0100)_2$, c'est ça le codage de $+4$.
- Exemple : calculer $CHS_4(-5)$:
  - Le codage de -5 est $(1011)_2$ (vu sur la roue des nombres).
  - On inverse tous les bits : $(0100)_2$.
  - On ajoute 1 au nombre inversé : $(0100)_2 + 1 = (0101)_2$, c'est ça le codage de $+5$.

Pour comprendre cette méthode, examinez la roue des nombres, et cherchez où sont situés deux nombres dont tous les bits sont inversés. Par exemple, où est l'inversion de 0011, celle de 0100, et aussi les inversions de 1101 et 1111 ? Vous remarquerez qu'il y a un décalage systématique et que pour le rectifier, il faut incrémenter le nombre inversé. Incrémenter revient à tourner d'un cran dans le sens horaire, dans le sens de la flèche orange clair.

2.3. Méthode 3

On parcourt le codage de N de droite à gauche, on recopie tous les bits jusqu'au premier 1 inclus, ensuite on inverse tous les bits suivants.

Exemple : soit 1011010100 à traiter avec cette méthode

1. On commence tout à droite
2. On trouve 0, on le recopie
3. On trouve 0, on le recopie
4. On trouve 1, c'est le premier 1 rencontré, alors on le recopie mais maintenant on recopiera toute la suite en l'inversant.
5. On trouve 0, ça devient un 1
6. On trouve 1, ça devient un 0
7. etc.

Le résultat est 0100101111, obtenu sans aucun calcul.

Exemple : soit 011010111 à traiter avec cette méthode

1. On commence tout à droite
2. On trouve 1, c'est le premier 1 rencontré, alors on le recopie mais maintenant on recopie toute la suite en l'inversant.
3. On trouve 1, ça devient un 0
4. On trouve 1, ça devient un 0
5. On trouve 0, ça devient un 1
6. etc.

Le résultat est 100101001.

- Exemple : calculer $CHS_4(+6)$ pour obtenir le codage de -6 :
  - Le codage de $+6$ est $(0110)_2$.

  - On parcourt le codage de droite à gauche.
    1. On recopie tous les zéros (il y en a un seul)
    2. Quand on tombe sur le premier 1 à droite, on le recopie aussi
    3. mais ensuite on inverse tous les bits suivants ($01 \rightarrow 10$)
  - On obtient donc : $(1010)_2$, c'est ça le codage de -6.
- Exemple : calculer $CHS_4(-5)$ pour obtenir le codage de $+5$:
  - Le codage de -5 est $(1011)_2$.
  - On parcourt le codage de droite à gauche.
    1. On recopie tous les zéros : ici, il n'y en a pas
    2. On tombe directement sur le premier 1 à droite, on le recopie
    3. mais ensuite on inverse tous les bits suivants ($101 \rightarrow 010$)
  - On obtient donc : $(0101)_2$, c'est ça le codage de $+5$.

2.4. Choix de la méthode

La méthode 1 n'est pas du tout pratique car elle oblige à faire une soustraction compliquée.

La méthode 2 demande de faire une incrémentation (addition avec 1), cependant le report des retenues est souvent limité à quelques chiffres.

La méthode 3 ne demande aucun calcul, mais il ne faut pas se tromper de direction pour traiter les bits, savoir lesquels recopier et lesquels inverser.

Les trois méthodes donnent exactement le même résultat. Choisissez la méthode que vous préférez, en ayant conscience que ce n'est pas la seule.

2.5. Exercices

Codez les entiers suivants en convention $C2^8$ ($n = 8$). La méthode est toujours la même :

> - Si N est positif, alors son codage est lui-même écrit en base 2 sur n bits.
> - Si N est négatif, alors écrire $|N|$ (càd $-N$) en base 2 sur n bits (divisions successives, voir TD1), puis appliquer CHS_n sur cette écriture.
>
> Dans les deux cas, si le résultat n'est pas du même signe que N, alors le codage n'est pas possible.

i. -7
ii. -12
iii. $+17$
iv. -28
v. $+27$
vi. -140

Question optionnelle

vii. Quelle est la plage des entiers relatifs codables sur n bits : $min_n \ldots max_n$. Il est question de 2^{n-1}. Regardez ce qu'il en est sur la roue des nombres relatifs pour laquelle $n = 4$.

3. Décodage d'un entier en convention $C2^n$

Toute la partie précédente explique comment coder un entier relatif dans la convention $C2^n$. On s'intéresse maintenant à retrouver un entier relatif N qui a été codé dans cette convention. Il y a une méthode pour cela :

- Si le bit de poids fort est 0, l'entier est positif, et il est égal à son codage. Il suffit de l'écrire en base 10 et mettre un «+» devant.
- Si le bit de poids fort est 1, l'entier est négatif. On trouve sa valeur en base 10 ainsi :
  1. on calcule $CHS_n(N)$ avec la méthode que vous préférez,
  2. on écrit le résultat en base 10 (polynôme, voir TD1),
  3. on rajoute un «−» devant.

Quels sont les entiers en base 10 codés par ces nombres sur 8 bits ?

i. $(11101110)_2$
ii. $(01001000)_2$
iii. $(11001110)_2$
iv. $(00111000)_2$
v. $(11101011)_2$

Rappel: les puissances de 2 sont 128, 64, 32, 16, 8, 4, 2, 1.