5. Codage des nombres entiers relatifs

Principes de fonctionnement des ordinateurs

Jonas Lätt

Centre Universitaire d'Informatique

UNIVERSITÉ DE GENÈVE
CENTRE UNIVERSITAIRE D'INFORMATIQUE

Trouvé une erreur sur un transparent? Envoyez-moi un message

- *sur Twitter @teachjl ou*
- *par e-mail jonas.latt@unige.ch*

Contenu du cours

Partie I: Introduction

1. Introduction
2. Histoire de l'informatique

Partie II: Codage de l'information

3. Information digitale et codage de l'information
4. Codage des nombres entiers naturels
5. **Codage des nombres entiers relatifs**
6. Codage des nombres réels
7. Codage de contenu média

Partie III: Circuits logiques

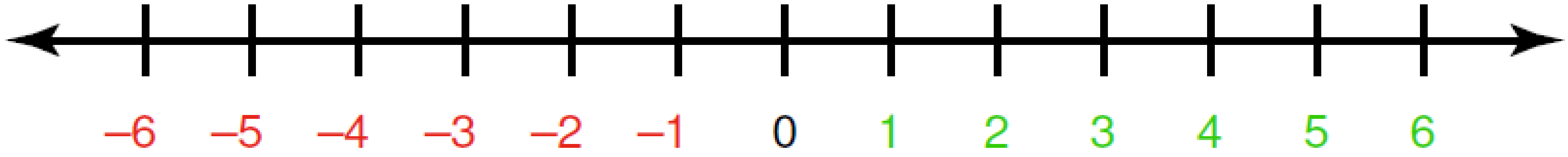
8. Portes logiques
9. Circuits logiques combinatoires et algèbre de Boole
10. Réalisation d'un circuit combinatoire
11. Circuits combinatoires importants
12. Principes de logique séquentielle
13. Réalisation de la bascule DFF

Partie IV: Architecture des ordinateurs

14. Architecture de von Neumann
15. Réalisation des composants
16. Code machine et langage assembleur
17. Réalisation d'un processeur
18. Performance et micro-architecture
19. Du processeur au système

PARTIE I:
PRINCIPES DU CODAGE DES NOMBRES RELATIFS

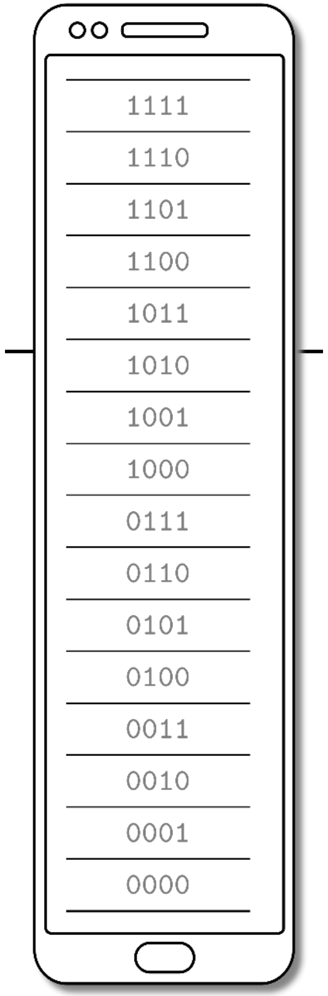
Les nombres entiers relatifs



$$\mathbb{Z} = \{\ldots -2, -1, 0, 1, 2, \ldots\}$$

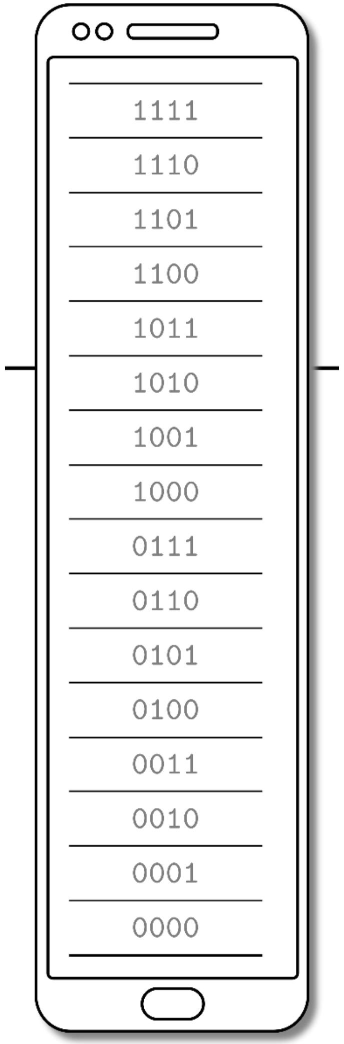
Première idée: Codage en signe-norme

$\texttt{code}_{\mathbb{N}(4)}$
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0



Première idée: Codage en signe-norme

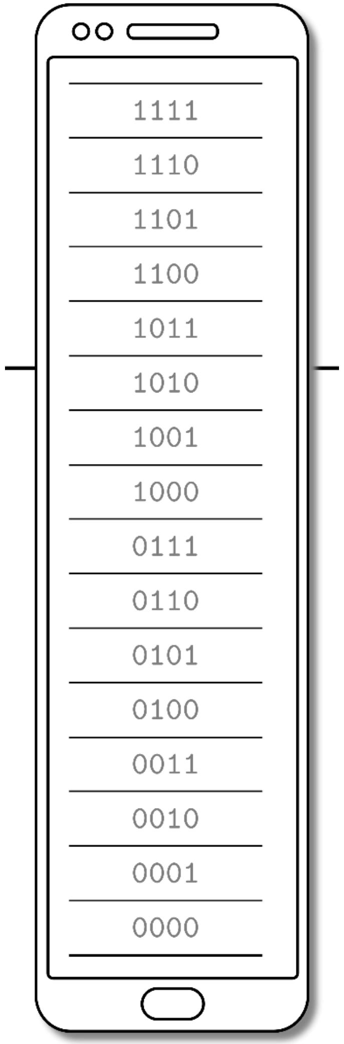
$\texttt{code}_{\mathbb{N}(4)}$	Signe-Norme
15	-7
14	-6
13	-5
12	-4
11	-3
10	-2
9	-1
8	-0
7	7
6	6
5	5
4	4
3	3
2	2
1	1
0	0



Problèmes …

Meilleure idée: Codage par complément à 2

$\texttt{code}_{\mathbb{N}(4)}$	Signe-Norme
15	-7
14	-6
13	-5
12	-4
11	-3
10	-2
9	-1
8	-0
7	7
6	6
5	5
4	4
3	3
2	2
1	1
0	0

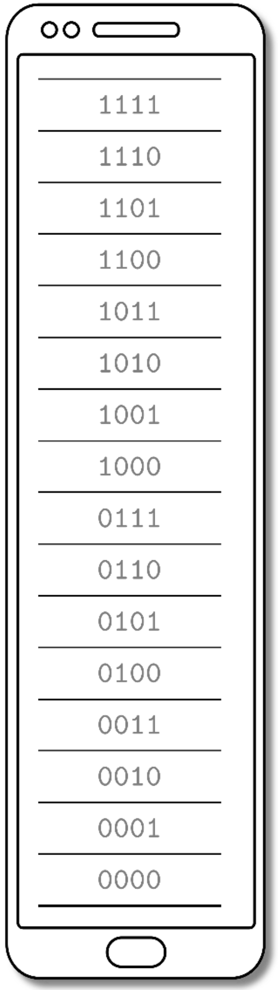


Jonas Lätt

Meilleure idée: Codage par complément à 2

Nombres négatifs
$-2^{k-1} \leq x < 0$

$\texttt{code}_{\mathbb{N}(4)}$	Signe-Norme	$\texttt{code}_{\mathbb{Z}(4)}$
15	-7	-1
14	-6	-2
13	-5	-3
12	-4	-4
11	-3	-5
10	-2	-6
9	-1	-7
8	-0	-8
7	7	7
6	6	6
5	5	5
4	4	4
3	3	3
2	2	2
1	1	1
0	0	0



Nombres positifs
$0 \leq x < 2^{k-1}$

Le sous-ensemble des nombres entiers relatifs $\mathbb{Z}_{(k)}$

Rappel sur $\mathbb{N}_{(k)}$: $\mathbb{N}_{(k)} = \{x \in \mathbb{N} |\ 0 \leq x < 2^k\}$

$\mathbb{Z}_{(k)}$: «On décale $\mathbb{N}_{(k)}$ à gauche» $\mathbb{Z}_{(k)} = \{x \in \mathbb{Z} |\ -2^{k-1} \leq x < 2^{k-1}\}$.

Exemple: $\mathbb{Z}_{(8)} = \{-128\ \ldots 127\}$.

A l'aide de la règle du complément à 2, on peut représenter tous les nombres dans l'intervalle $\mathbb{Z}_{(k)}$ par un *mot* à *k* bits.

PARTIE II:
COMPLÉMENT À 2 – RÈGLES DE CODAGE ET DÉCODAGE

Codage par complément à 2

Nombres négatifs
$-2^{k-1} \leq x < 0$

$\texttt{code}_{\mathbb{N}(4)}$	Signe-Norme	$\texttt{code}_{\mathbb{Z}(4)}$	
15	-7	-1	1111
14	-6	-2	1110
13	-5	-3	1101
12	-4	-4	1100
11	-3	-5	1011
10	-2	-6	1010
9	-1	-7	1001
8	-0	-8	1000
7	7	7	0111
6	6	6	0110
5	5	5	0101
4	4	4	0100
3	3	3	0011
2	2	2	0010
1	1	1	0001
0	0	0	0000

Nombres positifs
$0 \leq x < 2^{k-1}$

Codage par complément à 2:

$$\texttt{code}_{\mathbb{Z}(\mathrm{k})}$$

$$\texttt{code}_{\mathbb{Z}(\mathrm{k})} : x \longmapsto \begin{cases} \texttt{code}_{\mathbb{N}(\mathrm{k})}(x) & \text{si } 0 \leq x < 2^{k-1} \\ \texttt{code}_{\mathbb{N}(\mathrm{k})}(x+2^k) & \text{si } -2^{k-1} \leq x < 0 \end{cases}$$

Codage et Décodage: Exemples

Plaçons-nous dans l'espace $\mathbb{Z}_{(8)}$, dans lequel les nombres de -128 … 127 sont codés sur des mots d'un octet (8 bits).

$$\texttt{code}_{\mathbb{Z}_{(8)}}(80) =$$

$$\texttt{code}_{\mathbb{Z}_{(8)}}(-80) =$$

$$\texttt{decode}_{\mathbb{Z}_{(8)}}(1010\ 0101) =$$

Décodage en $\mathbb{Z}_{(k)}$

Interprétation: du décodage en $\mathbb{Z}_{(k)}$: le bit de poids élevé possède un poids négatif:

Décodage en $\mathbb{N}_{(k)}$: $x = a_0 2^0 + a_1 2^1 + \cdots + a_{N-2} 2^{N-2} + a_{N-1} 2^{N-1}$

Décodage en $\mathbb{Z}_{(k)}$: $x = a_0 2^0 + a_1 2^1 + \cdots + a_{N-2} 2^{N-2} - a_{N-1} 2^{N-1}$

Il s'agit d'une manière intéressante d'interpréter, et donner une signification, à la règle de codage par complément à 2.

PARTIE III:
ADDITION DE NOMBRES RELATIFS

Addition en $\mathbb{Z}_{(k)}$

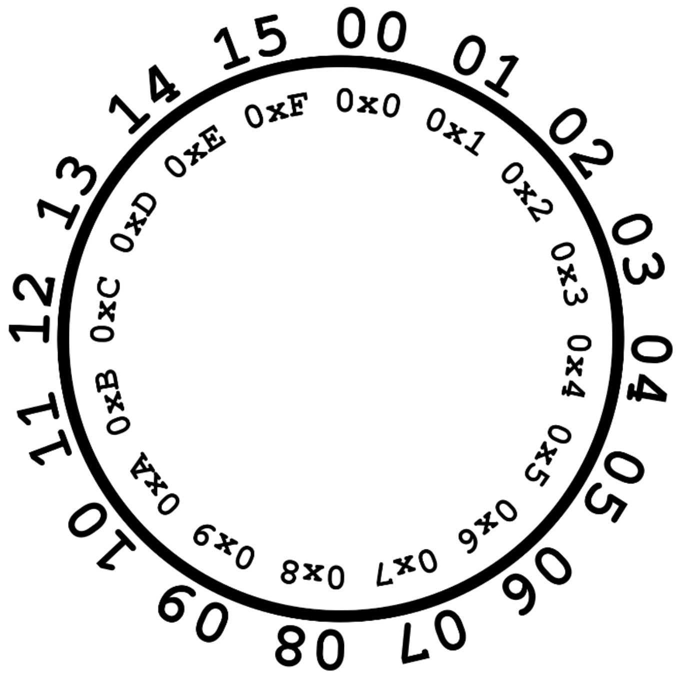
Idée

- On n'informe pas le processeur de l'ordinateur si les données représentées sont en $\mathbb{N}_{(k)}$ ou en $\mathbb{Z}_{(k)}$.
- Le même circuit additionneur s'utilise dans les deux cas.
- Tour de magie: ça marche en $\mathbb{N}_{(k)}$ et en $\mathbb{Z}_{(k)}$!

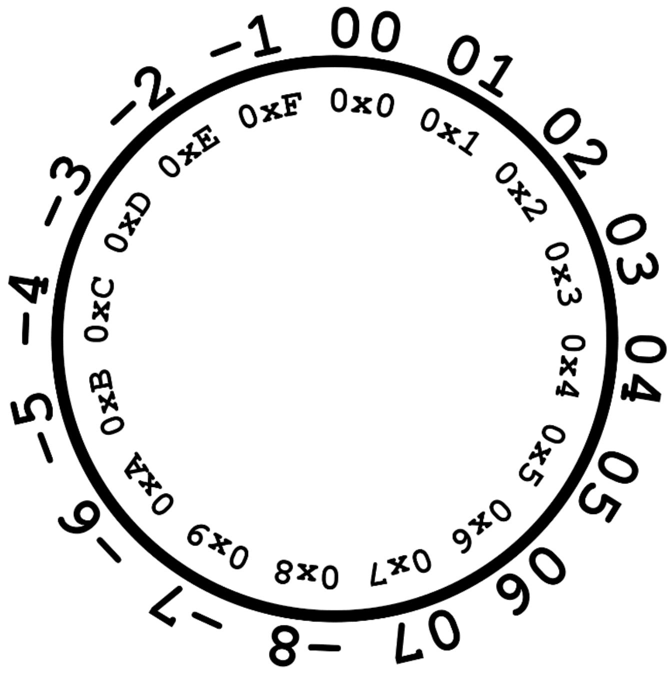
Equivalence de l'addition / soustraction entre $\mathbb{N}_{(4)}$ et $\mathbb{Z}_{(4)}$

Entiers naturels $\mathbb{N}_{(4)}$

Entiers relatifs $\mathbb{Z}_{(4)}$ (complément à 2)



En représentation interne, une fonction logique lit une séquence de bits et produit une séquence de bits.



11 + 3 = 14

0xB -> 0xE

-5 + 3 = -2

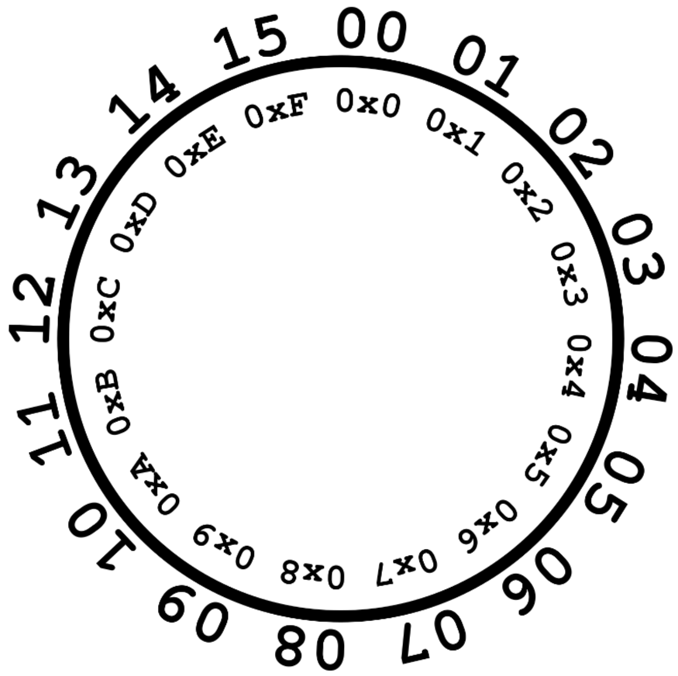
L'opération effectuée peut s'interpréter comme une addition en $\mathbb{N}_{(4)}$…

… ou alors, comme une addition en $\mathbb{Z}_{(4)}$.

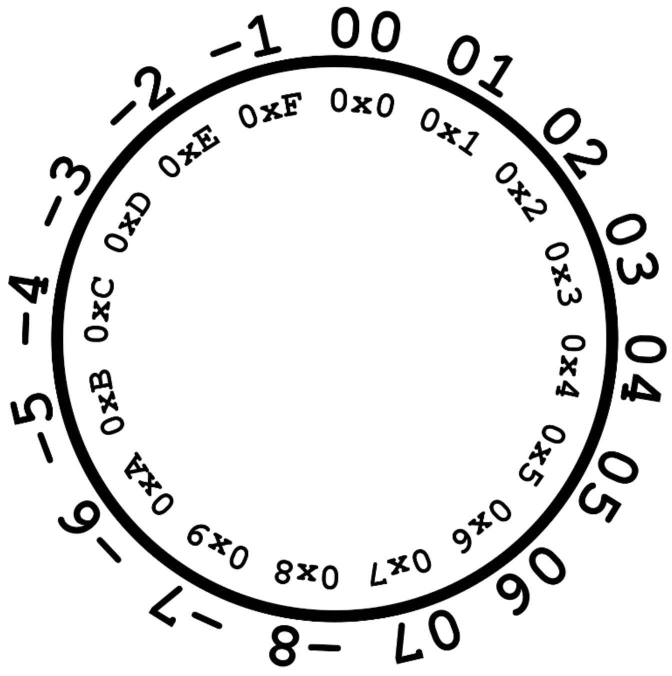
Jonas Lätt

Equivalence de l'addition / soustraction entre $\mathbb{N}_{(4)}$ et $\mathbb{Z}_{(4)}$

Entiers naturels $\mathbb{N}_{(4)}$



Entiers relatifs $\mathbb{Z}_{(4)}$ (complément à 2)



6 + 3 = 9

$\boxed{\text{0x6 -> 0x9}}$

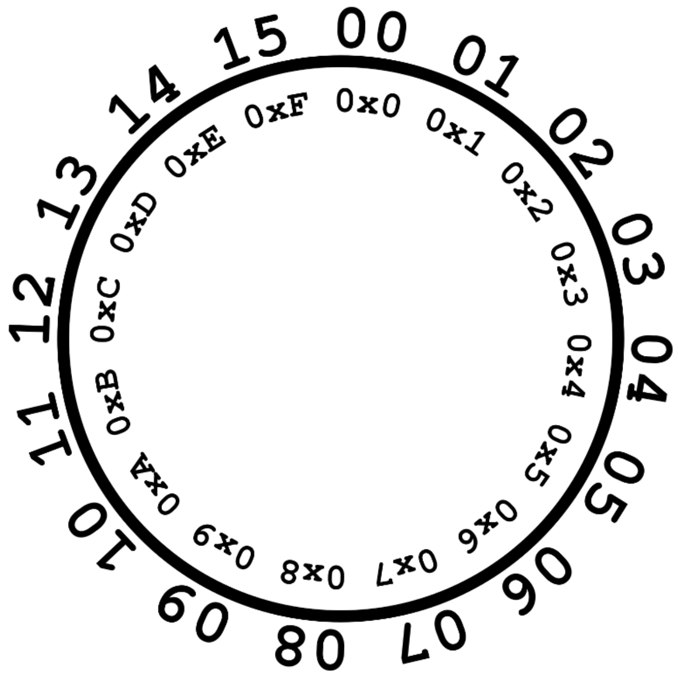
6 + 3 = 9 (Overflow) -> -7

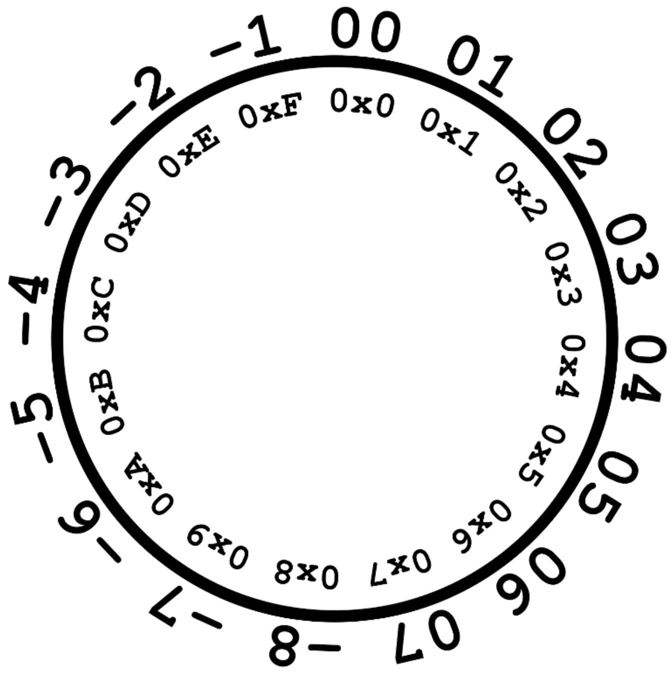
Jonas Lätt

Equivalence de l’addition / soustraction entre $\mathbb{N}_{(4)}$ et $\mathbb{Z}_{(4)}$

Entiers naturels $\mathbb{N}_{(4)}$

Entiers relatifs $\mathbb{Z}_{(4)}$ (complément à 2)





14 + 3 = 17 (overflow) -> 1

$\boxed{\text{0xB -> 0xE}}$

-5 + 3 = -2

Jonas Lätt

Règles de débordement en $\mathbb{Z}_{(k)}$

$$\text{Addition en } \mathbb{Z}_{(\mathrm{k})} : \quad x, y \quad \longmapsto \begin{cases} x+y+2^k & \text{si } x+y < -2^{k-1} \\ x+y-2^k & \text{si } x+y \geq 2^{k-1} \\ x+y & \text{sinon} \end{cases}$$

Avantages du complément à 2

1. Le circuit `add` peut effectuer des additions pour des valeurs en $\mathbb{N}_{(k)}$ tout comme en $\mathbb{Z}_{(k)}$: le circuit n'a même pas besoin de savoir lequel des deux codages est utilisé.
2. A voir dans la prochaine partie: le circuit `add` permet d'effectuer non seulement des additions, mais aussi des soustractions, en $\mathbb{N}_{(k)}$ tout comme en $\mathbb{Z}_{(k)}$.

PARTIE IV:
SOUSTRACTION DE NOMBRES RELATIFS

Rappel: algorithme de soustraction de nombres

- Technique classique: l'emprunt de retenues.
- Exemple: La soustraction z = x - y = 4321 – 1234.

	4	3	2	1
-	1	2	3	4

Cet algorithme peut être réalisé dans un ordinateur, mais mène à un circuit complexe. On cherche une solution techniquement plus simple pour l'ordinateur.

Soustraction de nombres entiers

Le circuit `add` peut être réutilisé pour la soustraction de nombres (on peut utiliser `add` pour réaliser le circuit de soustraction `sub`)!

- Si les arguments de `sub` sont des représentations de nombres relatifs: Pour tout argument `Y` qui représente un nombre y, on peut trouver une séquence binaire $\mathtt{Y}_0$ qui représente la valeur opposée $-y$.
- La soustraction $z = x - y$ peut s'écrire $z = x + (-y)$: Il est possible d'effectuer la soustraction à l'aide du circuit additionneur `add`.
- Si les arguments `sub` sont des représentations de nombres entiers naturels, il suffit de les *interpréter* comme des entiers relatifs, et l'argument ci-dessus reste valable. Comme nous l'avons vu au chapitre précédent, le circuit `add` ne fait pas la difference entre entiers naturels et relatifs.

Comment calculer la valeur opposée de y?

$\mathtt{Y}$ est la representation de y.

Comment trouver $\mathtt{Y}_0$, qui est la représentation de $-y$, en manipulant les bits de $\mathtt{Y}$? Tout ça doit se faire au niveau d'un circuit, donc en représentation interne.

Codage en complément à 2

Nombres négatifs
$-2^{k-1} \leq x < 0$

$\texttt{code}_{\mathbb{N}(4)}$	Signe-Norme	$\texttt{code}_{\mathbb{Z}(4)}$	
15	-7	-1	1111
14	-6	-2	1110
13	-5	-3	1101
12	-4	-4	1100
11	-3	-5	1011
10	-2	-6	1010
9	-1	-7	1001
8	-0	-8	1000
7	7	7	0111
6	6	6	0110
5	5	5	0101
4	4	4	0100
3	3	3	0011
2	2	2	0010
1	1	1	0001
0	0	0	0000

Nombres positifs
$0 \leq x < 2^{k-1}$

"Passage d'une valeur x vers son opposé":

- Calculer $16 - x$
- Calculer $2^k - x$

Comment calculer la valeur opposée de y?

$\mathtt{Y}$ est la representation de y.

Comment trouver $\mathtt{Y}_{\mathtt{0}}$, qui est la représentation de $-y$, en manipulant les bits de $\mathtt{Y}$?

En représentation externe $\mathbb{N}_{(k)}$, c'est facile:

- On cherche la valeur $\mathtt{decode}_{\mathbb{N}_k}(\mathtt{Y}_{\mathtt{0}}) = 2^k - y$

Mais en représentation interne ??

Mais en représentation interne? Astuce:

$$\boxed{\texttt{decode}_{\mathbb{N}_k}(\texttt{Y}_{\texttt{O}}) = 2^k - y = 1 + (2^k - 1) - y}$$

Astuce: représentation de -y

Entrée: Codage de y, Y

Sortie: Codage de $-y$, Y_0

$11111111_{(2)}$

On aligne k fois le “1”.

Astuce: $2^k - y = 1 + (2^k - 1) - y$

$$
\begin{array}{cccccccccc}
 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
- & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
\hline
 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{array}
$$

Il n’y a pas de retenue!

Il suffit d’inverser tous les bits de x.

Calcul du complément à 2

Ce procédé en deux étapes s’appelle le *calcul du complément à 2*:

1. Inversion de tous les bits
2. Addition de la valeur 1 au résultat

Complément à 2: représentation de -x

Règle:

- On inverse tous les bits de x.
- On ajoute 1 au résultat.

Exemple. Entrée: La valeur 3 codée en $\mathbb{Z}_{(4)}$.

0011 — Le codage de la valeur 3

1100 — Inversion des bits

1101 — +1

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

Le circuit C2: Calcul du complément à 2

Le circuit $C2_{(k)}$

- Inverse tous les bits
- Rajoute la valeur 1 au résultat

Cela permet de soustraire deux nombres, $z = x - y$, à l'aide du circuit `add`:

$$\mathtt{Z} = \mathtt{add}_{(k)}(\mathtt{X}, \mathtt{C2}(\mathtt{Y}))$$

Terminologie: Complément à 2

Le terme “complément à 2” s’utilise de différentes manières:

- Un nombre relatif est codé selon la **règle du complément à 2** lorsqu’on applique le codage $\texttt{code}_{\mathbb{Z}(\mathrm{k})}$
- On **calcule le complément à 2** d’un séquence de bits en appliquant l’opérateur $C2_{(k)}$: inversion des bits, calcul +1.
- La **méthode du complément** est une méthode qui permet de soustraire deux nombres en n’effectuant que des additions.