

NSI 1ère - Données

Complément à deux

qkzk

Le complément à deux : comment coder les entiers négatifs dans une machine ?

Les entiers relatifs

Rappels :

- **Entiers naturels** : entiers positifs ou nuls (0, 1, 2 etc.)
- **Entiers relatifs** : entiers de n'importe quel signe (... , -2, -1, 0, 1, ...)

Les entiers relatifs

Le problème du signe :

Un signe n'est pas un nombre. . .

On ne peut pas l'encoder directement en binaire.

Le principe est d'attribuer au *bit de poids fort* (premier bit) le signe du nombre.

- Si le *bit de poids fort* est **0**, le nombre est **positif**,
- Si le *bit de poids fort* est **1**, le nombre est **négatif**.

Nombres encodés sur un octet

Contrainte immédiate :

Il faut que la machine sache quelle est la taille du nombre !

Sinon :

- Comment déterminer "le bit de poids fort" ?
- Comment savoir où s'arrête le nombre ?

Durant tout le chapitre, on encodera **nos nombres entiers sur 8 bits**.

Approche naïve : binaire signé

Essayons avec cette simple règle :

Pour encoder un entier sur 8 bits,

- On détermine la représentation binaire de sa valeur absolue
- Ensuite on remplit de 0 à gauche.

Signe

- Si le nombre est positif, on garde le bit de poids fort à 0,
- **Sinon, on met le bit de poids fort à 1.**

Approche naïve : binaire signé

27

27 = 0b11011

On complète sur 8 bits :

27 = 0b 0001 1011

27 > 0 on garde le premier bit à 0

Approche naïve : binaire signé

-9

La valeur absolue de -9 est 9.

9 = 0b1001

On complète sur 8 bits :

9 = 0b 0000 1001

-9 < 0 on remplace le premier bit par -1 :

-9 = 0b 1000 1001

Jusqu'ici tout va bien...

Et soudain, c'est le drame...

Essayons d'ajouter ces exemples :

Vérifions que $27 + (-9) = 18$

```
  0b 0001 1011
+ 0b 1000 1001
-----
= 0b 1010 0100
```

... mais 0b 1010 0100 = -36 en binaire signé...

Échec total ! Le binaire signé ne permet pas de réaliser les additions habituelles

Exercice 1

On suppose toujours nos entiers encodés sur un octet.

1. Donner la représentation binaire naïve de 12, de -100 et de -88.
2. Réaliser l'addition binaire bit à bit $12 + (-100)$.
3. Comparer avec le résultat obtenu.

La méthode naïve ne permet pas de faire de calculs !

Avec la méthode naïve, on ne peut plus réaliser d'opération naturelle sur les entiers. On a maintenant deux objectifs :

1. Représenter les entiers relatifs,
2. Conserver le même algorithme pour l'addition

Le complément à deux

Complètement à deux : entiers positifs

Pour les entiers positifs

1. coder l'entier en binaire comme d'habitude,
2. compléter l'octet avec des 0 devant.

Complément à deux : entiers négatifs

Pour les entiers négatifs

1. Coder la valeur absolue du nombre en base 2,
2. compléter l'octet avec des 0 devant,
3. échanger tous les bits ($1 \leftrightarrow 0$),
4. ajouter 1.

Signe du complément à deux

- Si le bit de poids fort est 0 : le nombre est positif
- Si le bit de poids fort est 1 : le nombre est négatif

Exemple 1 : 27

27

1. coder l'entier en binaire comme d'habitude,
27 = 0b11011
2. compléter l'octet avec des 0 devant.
27 = 0b 0001 1011

Le complément à 2 sur un octet de 27 est 0b 0001 1011

Exemple 2 : -9

-9

1. coder la valeur absolue du nombre :
9 = 0b1001
2. compléter l'octet :
0b 0000 1001
3. échanger tous les bits :
0b 1111 0110
4. ajouter 1 :
0b 1111 0111

Le complément à 2 sur un octet de -9 est 0b 1111 0111

Complément à 2, méthode rapide

Méthode rapide

La méthode précédente se code facilement, elle est plus pénible à la main.

La méthode rapide : complément à 2 sur un octet de n :

Si l'entier est négatif :

1. donner la représentation binaire de $|n|$
2. Compléter à gauche jusqu'à avoir la taille voulue
3. de droite à gauche, conserver tous les bits jusqu'au premier 1 inclus
4. changer tous les bits à gauche

Méthode rapide : exemple

$n = -108$

Valeur absolue :

$$|n| = 108 = 64 + 32 + 8 + 4$$

Représentation binaire :

$$|n| = 108 = 0b110 1100$$

Compléter :

$$|n| = 108 = 0b0110 1100$$

Conserver les 0 à droite jusqu'au premier 1 inclus, changer tous les bits à gauche.

$$n = -108 = 0b1001 0100$$

Exercice 2

Donner les compléments de à 2 sur un octet de 12, -100 et de -88.

Vérifions : $27 + (-9)$

Vérifions : $27 + (-9) = 18$

```
  0001 1011
+ 1111 0111
-----
= 0001 0010
```

On vérifie immédiatement que $18 = 0b10010$

Remarque la dernière retenue (tout à gauche) disparaît.

Exercice 3

1. Réaliser l'addition binaire des compléments à 2 des nombres 12 et -100.
2. Vérifier qu'on retrouve bien le résultat précédent pour -88.

Complément à deux vers décimal

Si l'entier est positif (son premier bit est 0)

On fait comme d'habitude.

Exemple : $0b\ 0001\ 1011$

$$1 \times 1 + 1 \times 2 + 1 \times 8 + 1 \times 16 = 27$$

Du complément à deux vers le décimal

Si l'entier est négatif (son premier bit est 1)

1. Échanger tous les bits $0 \leftrightarrow 1$,
2. Ajouter 1,
3. Convertir en décimal normalement,
4. Changer le signe.

Exemple : $0b\ 1111\ 0111$

Exemple : $0b\ 1111\ 0111$

1. On échange tous les bits,
 $0b\ 0000\ 1000$
2. On ajoute 1,
 $0b\ 0000\ 1001$
3. On converti en binaire comme d'habitude,
 $0b\ 1001 = 1 * 1 + 1 * 8 = 9$
4. On change le signe.
 $0b\ 1111\ 0111 = -9$

Du complément à 2 vers le décimal, méthode rapide.

1. Conserver tous les bits à 0 droite jusqu'au premier 1 inclus,
2. Échanger tous les bits à gauche de ce 1,
3. Convertir en décimal normalement,
4. Changer le signe,

Du complément à 2 vers le décimal, méthode rapide : exemple

On part de $n = 0b\ 1010\ 1000$, en complément à 2 sur 8 bits,

1. Conserver tous les bits à 0 droite jusqu'au premier 1 inclus,
2. Échanger tous les bits à gauche de ce 1,
 $0b\ 0101\ 1000$

3. Convertir en décimal normalement,
 $0b\ 0101\ 1000 = 8 + 16 + 64 = 88$
4. Changer le signe :
 en complément à 2 sur un octet $n = 0b\ 1010\ 1000 = -88$

Exercice 4

Donner les notations décimales des compléments à deux sur un octet suivants :

1. $0b1111\ 1111$
2. $0b1000\ 0000$
3. $0b0111\ 1111$
4. $0b1010\ 0011$

Table de valeurs

bit
de
signe

0	1	1	1	1	1	1	1	=	127
0							...	=	...
0	0	0	0	0	0	1	0	=	2
0	0	0	0	0	0	0	1	=	1
0	0	0	0	0	0	0	0	=	0
1	1	1	1	1	1	1	1	=	-1
1	1	1	1	1	1	1	0	=	-2
1							...	=	...
1	0	0	0	0	0	0	1	=	-127
1	0	0	0	0	0	0	0	=	-128

Combien d'entiers relatifs sur un octet ?

Règle

Sur 8 bits en complément à deux, on peut encoder de -128 à 127 ,

Combien d'entiers relatifs sur avec n bits ?

Règle

Sur un octet on peut encoder de -2^{n-1} à $2^{n-1} - 1$.

Complément à 2 : résumé

- On dispose d'une méthode permettant d'ajouter des entiers (et donc de faire les opérations habituelles...) qui fonctionne aussi avec les entiers *négatifs*.
- Cette méthode permet d'encoder les entiers de -2^{n-1} à $2^{n-1} - 1$
- La méthode rapide pour encoder un entier en complément à 2 sur un octet :
 - s'il est positif, on l'écrit en binaire et on complète l'octet avec des 0 à gauche,
 - s'il est négatif, on écrit sa valeur absolue en binaire qu'on complète à gauche,
 - on conserve tous les 0 à droite jusqu'au premier 1 inclus,
 - on inverse tous les bits à gauche de ce premier 1.

Python et le complément à 2.

Les opérations précédentes imposent de choisir une taille maximale pour les entiers : **codés sur un octet**

Dans Python les entiers ont une **taille arbitraire**, il ne peut afficher nativement le complément à deux.

```
>>> bin(12)
'0b1100'
>>> bin(-12)
'-0b1100'
```

Pour obtenir le complément à 2, il faut le programmer.