Représentation des entiers relatifs

Christophe Viroulaud

Première - NSI

DonRep 03

Un système *64 bits* peut représenter 2^{64} entiers.

```
>>> import sys
>>> sys.maxsize
9223372036854775807
```

Code 1 – Cette valeur correspond à $2^{63} - 1$.

Observation

Un des bits ne semble pas utilisé.

Maîtriser la représentation des entiers négatifs en mémoire.

Sommaire

1. Addition de deux nombres binaires

2. Une représentation naïve des entiers négatifs

3. Le complément à 2 puissance n

Addition de deux nombres binaires

À retenir

Une addition en base 2 applique les mêmes principes qu'en base 10 :

- $0 + 0 = 0$
- $1 + 0 = 1$
- $1 + 1 = 0$ et une retenue de 1
- $1 + 1 + 1 = 1$ et une retenue de 1

Activité 1 :

1. Convertir 25 et 12 en base 2, sur 1 octet.
2. Effectuer l'addition binaire de ces nombres.
3. Convertir le résultat en base 10. Le résultat est-il correct ?

Correction

$$
\begin{array}{ccccccccc}
 & & & 1 & 1 & & & & \\
 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
+ & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
\hline
 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1
\end{array}
$$

Vérification :

$$0\times2^7+0\times2^6+1\times2^5+0\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0 = 37$$

Sommaire

Bit de poids fort

Le bit le plus à gauche de la représentation n'est pour l'instant pas utilisé. C'est le **bit de poids fort**.

À retenir

Pour représenter un nombre entier, il faut connaître la taille du mot mémoire.

Une première idée serait d'utiliser ce bit comme marqueur de signe :

- 0 pour un entier positif,
- 1 pour un entier négatif.

Ainsi l'entier -25 serait encodé dans un mot mémoire de 1 octet :

$$-25_{10} = 10011001_2$$

À retenir

Cette représentation n'a pas été retenue car elle présente plusieurs défauts.

Sommaire

Le zéro

Dans un système *8 bits* le zéro est représenté par 00000000_2. Cependant 10000000_2 se traduit par -0. Il y a donc deux représentations pour zéro.

Erreur d'addition

$$-25 + 12$$

$$\begin{array}{ccccccccc}
 & & & 1 & 1 & & & & \\
 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
+ & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
\hline
 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1
\end{array}$$

Avec cette représentation :

$$2^5 + 2^2 + 2^0 = 37$$

$$10100101_2 = -37_{10}$$

$$-25 + 12 \neq -37$$

À retenir

Utiliser le bit de poids fort pour stocker le signe du nombre est une représentation erronée.

Sommaire

Définition

À retenir

La représentation naïve (utiliser le bit de gauche pour enregistrer le signe) présentée précédemment contient de nombreux défauts.
La représentation des entiers relatifs en mémoire utilise le principe **du complément à 2 puissance n**.

À retenir

Pour représenter un nombre en complément à 2, il faut connaître la taille (le nombre de bits utilisés) de la représentation.

Le complément à 2 puissance *n* est une représentation qui ne change rien pour les entiers positifs. Ainsi sur 8 bits :

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

À retenir

La valeur $2^n - |x|$ représente l'entier négatif x. Ainsi sur 8 bits, -1 est représenté en mémoire par

$$2^8 - 1 = 256 - 1 = 255_{10} = 11111111_2$$

1	1	1	1	1	1	1	1	$=$	-1	$2^8 - \|-1\| = 255$
1	1	1	1	1	1	1	0	$=$	-2	$2^8 - \|-2\| = 254$
1	...							$=$	...	
1	0	0	0	0	0	0	1	$=$	-127	$2^8 - \|-127\| = 129$
1	0	0	0	0	0	0	0	$=$	-128	$2^8 - \|-128\| = 128$
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	

Tableau 1 – Représentation des entiers en mémoire

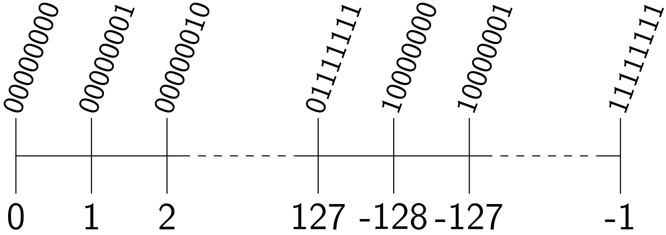
Activité 2 :

1. Quel est le plus grand entier positif représentable avec 1 octet ?
2. Quel est le plus petit entier négatif représentable avec 1 octet ?
3. Mêmes questions sur 10 bits ?

Correction

Sur 8 bits (1 octet), on peut représenter 256 valeurs :

- entier le plus grand : $2^{8-1} - 1 = 2^7 - 1 = 127$
- entier le plus petit : $-2^{8-1} = -2^7 = -128$



Remarque

Le zéro utilise une position dans les entiers positifs.

Sur 10 bits, on peut représenter 1024 valeurs :

- entier le plus grand : $2^{10-1} - 1 = 2^9 - 1 = 511$
- entier le plus petit : $-2^{10-1} = -2^9 = -512$

Sommaire

Calculer le complément à 2

Pour coder (-20) :

- Prendre le nombre positif 20 : 00010100
- Inverser les bits : 11101011
- Ajouter 1 : 11101100
- -20 : 11101100

Seconde méthode

Garder tous les chiffres depuis la droite jusqu'au premier 1 (compris) puis inverser tous les suivants.

- Prendre le nombre positif 20 : 00010100
- Garder la partie à droite telle quelle : 00010$\underline{100}$
- Inverser la partie de gauche après le premier un : $\underline{11101}$100
- -20 : 11101100

> **Activité 3 :** Calculer le complément à 2 (sur 1 octet) de -25.

Correction

- $25_{10} = 00011001_2$
- $0001100\underline{1}$
- $-25_{10} = 11100111$

Sommaire

Intérêt de la méthode

Il n'y a qu'un seul zéro.

Addition

$$-25 + 12$$

$$\begin{array}{ccccccccc}
 & & & & 1 & 1 & & & \\
 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
+ & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
\hline
 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1
\end{array}$$

Avec cette représentation :

$$2^7 + 2^6 + 2^5 + 2^4 + 2^1 + 2^0 = 243$$

Sur 8 bits, l'entier positif le plus grand est 127. Le résultat obtenu est donc un nombre négatif.

$$243 - 2^8 = 243 - 256 = -13$$

Activité 4 : Effectuer l'addition $-1 + 1$ en binaire représentée sur 8 bits.

Correction

$$-1+1$$

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

À retenir

La représentation utilise 8 bits. Les dépassements sont ignorés et on retrouve bien :

$$-1+1=0$$