Numération

Partie 2 : nombres entiers signés

Objectifs :

Additionner et multiplier des nombres représentés en binaire			
Comprendre ce qu'est un dépassement de capacité (overflow)			
Utiliser le complément à 2 pour représenter un nombre négatif en binaire			
Écrire en base 10 un nombre entier relatif écrit en binaire			

I Addition et multiplication en binaire

Poser les opérations en binaire ce fait comme en base 10. Il faut faire surtout attention aux retenues.

1) Additions en binaire

La table d'addition en binaire est :

+	0	1
0	0	1
1	1	10

Exemples

- 7 + 5 ce qui donne en binaire sur 4 bits $0111_2 + 0101_2$

	0	1	1	1
+	0	1	0	1
	1	1	0	0

  Or $1100_2 = 12$. Nous retrouvons bien 7+5 = 12.
- 12 + 5 ce qui donne en binaire sur 4 bits $1100_2 + 0101_2$

	1	1	0	0
+	0	1	0	1
1	0	0	0	1

  Or $1\,0001_2 = 17$. Nous retrouvons bien 12+5 = 17, mais nous avons eu besoin de 5 bits si nous disposons de 4 bits cela ne suffit pas, il y a un *dépassement de capacité (overflow)*. Le risque est que la machine ne garde que les 4 premiers bits et dans ce cas 12+5 est égale à 1 ce qui est bien sûr faux.

2) Multiplications en binaire

La table de multiplication en binaire est :

x	0	1
0	0	0
1	0	1

Exemple

7×3 ce qui donne en binaire sur 4 bits $0111_2\times0011_2$

	0	1	1	1
x	0	0	1	1
	0	1	1	1
0	1	1	1	
1	0	1	0	1

Or 1100_2 = 12. Nous retrouvons bien 7+5 = 12.

- 12 + 5 ce qui donne en binaire sur 4 bits 1100_2 +0101_2

Or 1 0101_2 = 21. Nous retrouvons bien $7\times3=21$, mais nous sommes encore dans un cas de dépassement de capacité (overflow)

I Une fausse bonne idée

Nous avons vu comment représenter les entiers naturels en langage binaire. Une première idée serait de coder le signe de l'entier sur le bit de poids fort qui est celui le plus à gauche puis la valeur absolue de l'entier sur les bits restants.

Exemple : Codons -3 sur 4 bits avec la convention :

- Si l'entier est positif ou nul, le bit de signe est 0.
- Si l'entier est négatif le bit de signe est 1.

On obtient $\underbrace{1}_{bit\ de\ signe}\ \underbrace{011}_{3}$

Ce codage n'est pas satisfaisant :

- Zéro admet deux codages : 0000 et 1000.
- En électronique, un *circuit additionneur* permet d'additionner deux entiers naturels écrits sur un nombre donné de bits On voudrait que le codage d'un nombre signé cela soit compatible avec l'addition sur les entiers non signés.
  . Or par exemple :

$0101\rightarrow 5$
$1011\rightarrow -3$
$0000\rightarrow 0$

Ce qui ne convient pas.

II Codage d'un entier signé – complément à 2

1) Analogie avec la base 10

En base 10, nous connaissons les compléments aux puissances de 10.

Le complément à 10 de 7 est 3 car 7 + 3 = 10. Le complément à 1000 de 966 est 34 car

966 + 34 = 1000.

Décidons que l'on écrive les nombres avec trois chiffres quitte à compléter par des 0 à gauche et en négligeant la retenue qui dépasse la taille fixée. On a alors, par exemple :

- Le complément à 1000 de 049 est 951 car 049 + 951 = 000.
- Le complément à 1000 de 001 est 999 car 001 + 999 = 000

De ces deux exemples, on peut avoir alors l'idée de coder -49 par $951=1000-49$ et -1 par $999=1000-1$

Dans ces conditions cela permet de coder les nombres de -500 à 499. Il n'y a qu'un seul 0 et cela reste compatible avec l'addition des nombres entiers positives sur trois chiffres.

En binaire, on remarque que lorsque on ajoute à un nombre le nombre obtenu en échangeant les valeurs des bits on obtient une suite de 1. Il suffit alors d'ajouter 1 pour obtenir une suite de 0 toujours en négligeant la retenue.

Par exemple sur trois bits :
$011+100=111$ et $111+1=000$

2) Le méthode du complément à 2

On décide du nombre de bits pour représenter le nombre. Le bit de poids forts indique le signe.

Propriété

Sur n bits, on peut représenter les entiers signés de $-2^{(n-1)}$ à $2^{(n-1)}-1$.

Par exemple sur 8 bits, on peut représenter les nombres compris entre $-2^{(8-1)}=-128$ et $2^{(8-1)}-1=127$

La méthode

- Pour coder un nombre positif, on met le bit de poids fort à 0 et l'on détermine l'écriture binaire du nombre.
- Pour coder un nombre négatif :
  1. On écrit en binaire le nombre sans son signe ;
  2. On remplace les 1 par des 0 et les 0 par des 1 ;
  3. On ajoute 1 au résultat.

Exemples :

Codage de 89 sur 8 bits : $\underbrace{0}_{signe}\ \underbrace{1011001}_{89}$.

Codage de -45 sur 8 bits.

1. On code 45 sur 8 bits en $45 = 0010\,1101_2$;
2. On remplace les 0 par des 1 et les 1 par des 0 : $1101\,0010_2$
3. On ajoute 1 : $1101\,0010_2 + 1 = 1101\,0011_2$

-45 est représenté par $1101\,0011_2$ en binaire sur 8 bits.
Vérifions la cohérence du résultat. On devrait avoir $45 + (-45) = 0$

$$\begin{array}{r} 0010\,1101 \\ +1101\,0011 \\ \hline 0000\,0000 \end{array}$$

Il reste une retenue (sur le neuvième bit), mais l'entier étant codé sur 8 bits, on obtient bien 0.

Remarques :

- $1101\ 0011_2$ = 211 donc -45 est représenté par le nombre 211.
  On peut constater que $2^8 - 45 = 256 - 45 = 211$. Sur 8 bits, les nombres en négatifs sont représentés par les nombres supérieurs ou égaux à 128.
- Le nombre de bits doit être défini au départ. Un nombre négatif n'aura pas le même codage sur 8 bits et sur 16 bits. -37 s'écrit 1101 1011 sur 8 bits mais 1111 1111 1101 1011 sur 16 bits. On constate qu'il suffit d'ajouter suffisamment de 1 à gauche du nombre.

Méthode 2
Sur n bits, pour obtenir l'écriture du nombre négatif $-2^{n-1} \leq x \leq -1$ écrit sur n bits, il suffit d'écrire en binaire le nombre
$2^n - |x|$.
Par exemple, pour écrire -45 sur 8 bits, on calcule $2^8 - |-45| = 256 - 45 = 211$ ce qui s'écrit en binaire $1101\ 0011_2$. On retrouve le résultat de l'exemple précédent.

Méthode 3
On peut utiliser la méthode des poids. Sur n le bit de poids fort aura la valeur $-2^{(n-1)}$.
Par exemple :

Valeurs	$-2^7 = -128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
Bits	1	1	0	1	0	0	1	1

$1101\,0011_2 = 1 \times (-128) + 1 \times 64 + 1 \times 16 + 1 \times 2 + 1 \times 1 = -45$

Méthode 4-méthode rapide

Pour utiliser la méthode du complément à 2 :

On convertit en binaire le nombre sans son signe. Puis on parcourt les bits un par un en partant de la droite, on change les bits qu'à partir du premier bit 1 rencontré, sans toucher celui-ci.

Exemples :

- 16 = 0001 0000 donc $-16 = 1111\,0000$
- 19 = 0001 0011 donc $-19 = 1110\,1101$

3) Convertir en base 10 un entier signé en binaire

Méthode 1 :

- Si le bit de poids fort est 0, le nombre est positif. On convertit le nombre en base 10 comme auparavant.
  Exemple : convertir le nombre signé sur un octet par $0001\ 1011_2$
  Le bit de poids fort (le plus à gauche) est 0 : le nombre est positif
  $0001\ 1011_2 = 27$. Ce nombre code l'entier 27

- Si le bit de poids fort est 1, le nombre est négatif.
  On détermine son complément à 2
  On ajoute 1
  On convertit le nombre en base 10
  On ajoute le signe moins devant ce nombre.
  Exemple : convertir le nombre signé sur un octet par : $1111\ 0011_2$
  Le bit de poids fort est 1, le nombre est négatif.
  Son complément à 2 est $0000\ 1100_2$
  $0000\ 1100+1 = 0000\ 1101_2$
  $0000\ 1101_2 = 13$
  On ajoute le signe devant. $1111\ 0011_2 = -13$

Méthode 2

On peut utiliser les poids :

Par exemple :

Valeurs	$-2^7=-128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
Bits	1	1	1	1	0	0	1	1

$$11110011_2 = 1\times(-128)+1\times 64+1\times 32+1\times 16+1\times 2+1\times 1=-13$$

Méthode rapide :

Pour convertir un nombre entier relatif en base 10 dans le cas où le bit de poids fort est 1.

On parcourt les bits un par un en partant de la droite, on change les bits qu'à partir du premier bit 1 rencontré, sans modifier celui-ci. On obtient l'opposé du nombre que l'on souhaite convertir. Enfin on convertit le nombre obtenu en base 10 et on lui ajoute un signe moins devant.

En reprenant l'exemple précédent :

$1111\ 0011_2$ a pour opposé $0000\ 1101_2$

Or $0000\ 1101_2 = 13$ donc $1111\ 0011_2 = -13$