REPRÉSENTATIONS DES ENTIERS RELATIFS – Z

La représentation des nombres relatifs n'a pas été développée pour nous faciliter la vie, mais celle du microprocesseur.

Rappel							R (1)		
	0		0		1			1	
+	0	+	1	+	0	+		1	Décimal
	0		1		1		1	0	$2_{(10)}$

	R									
	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Décimal
	1	1	1	1	1	1	1	1		
		0	0	0	0	1	0	1	1	$11_{(10)}$
+										
		1	1	1	1	0	1	0	1	$-11_{(10)}$
		0	0	0	0	0	0	0	0	

Pour représenter les nombres signés (relatifs) on va définir la taille de la donnée (8, 16, 32 bits …) et affecter au most significant bit (msb : bit le plus significatif) le bit de signe :

0 : nombre positif

1 : peut représenter un nombre négatif

Comment trouver la valeur d'un nombre pouvant être négatif ?

On ne peut pas connaître sa valeur de manière directe. Il faut utiliser une des méthodes suivantes :

1ère méthode : pour avoir la valeur négative du mot binaire recherché : la méthode du complément à 2.

•On prend le mot binaire et on inverse tous les bits (les bits à 1 passent à 0 et vice versa).
Valeur initiale : 1111 0101
Valeur complémentée (inversée) : 0000 1010
•On ajoute à ce mot inversé la valeur 1.
0000 1010 + 0000 0001 = 0000 1011
•La valeur trouvée est la représentation positive du nombre que l'on cherche.
0000 1011 correspond à 11 donc 1111 0011 correspond à -11.

2ème méthode : pour avoir la valeur négative du mot binaire recherché

Valeur initiale : 1111 0101
Prendre le bit de signe comme -1 x 2^7, ou 0 x 2^7, donc soit -128 ou 0 : ici -128

Additionner les autres poids $2^6+2^5+2^4+2^2+2^0$= 64 + 32 + 16 +4 +1 = 117

Additionner les 2 valeurs : -128 + 117 = -11

3ème méthode : pour avoir la représentation de la valeur négative à partir du positif

Convertir la valeur en décimal et soustraire 2^n avec n : nombre de bits

0000 1011 = 11_{10} sur 8 bits : 11 – 2^8 = 11 – 256 = -245 → 1111 0101 (128 + 64 + 32 + 16 + 4 + 1)

4ème méthode : pour avoir la représentation de la valeur négative à partir du positif : complément à 2

(11_{10}) = 0000 1011

inversion de tous les bits : 1111 0100

ajout de 1 : 1111 0101

Réalisons sans aucun contexte, l'opération suivante : l’addition de 2 nombres binaires.

1000 0100 + 0011 1010 = 1011 1110

Imaginons maintenant 2 contextes différents :

Le premier est le suivant :

Vous collectionnez des objets.

Le premier nombre représente le nombre d’objets que vous possédiez vendredi soir.

Le second nombre représente le nombre d'objets que vous avez trouvées dans un vide-grenier ce week-end.

La somme correspond donc au nombre d'objets que vous possédez lundi.

Vérifions : 132 + 58 = 190 → résultat correct.

Le second est le suivant :

Le premier nombre représente le solde de votre compte bancaire vendredi soir.

Le second nombre représente une opération réalisée ce week-end (débit ou crédit).

La somme correspond au nouveau solde de votre compte lundi.

Vérifions :

1000 0100 commence par un 1. Il peut être négatif (découvert). On ne peut pas connaître sa valeur de manière directe. Il faut utiliser la méthode du complément à 2.

1000 0100 devient 0111 10 11 auquel on ajoute 1 = 0111 1100 soit 124.

En conséquence, 1000 0100 représente -124.

0011 1010 aucun problème, il est positif et vaut 58

1011 1110 devient 0100 0001 auquel on ajoute 1 = 0100 0010 soit 66

En conséquence, 1011 1110 représente -66

– 124 + 58 = -66→ résultat correct.

Le microprocesseur n’a pas demandé le contexte pour faire l’opération. Il n’a pas fait une opération pour un contexte et une seconde pour un autre contexte. Il indiquera cependant dans un bit d’un registre interne particulier (registre d’état) que le nombre peut être interprété comme négatif, au programmeur d’en tenir compte ou pas lorsqu’il fait un test dans une condition « tant que A>0 » par exemple (Tout ceci est transparent en langage de haut niveau : python …)

Revenons sur l'exemple introductif : 0000 1011 + 1111 0101 = (1) 0000 0000

Si le contexte fait que les nombres représentés peuvent être positifs ou négatifs, cette somme est bien égale à 11 + (-11) = 0.

Si par contre, le contexte fait qu'il est impossible d'avoir des nombres négatifs, le résultat est correct également en tenant compte de la retenue 2^8 = 256 = 11 + 245

Il est trop fort ce microprocesseur ! Ou alors c'est nous ?

Exercices

•En utilisant le complément à 2, représentez -15 (représentation sur 8 bits)

•Représentez sur 8 bits l'entier 4 puis représentez, toujours sur 8 bits, l'entier -5. Additionnez ces 2 nombres (en utilisant les représentations binaires bien évidemment), vérifiez que vous obtenez bien -1.

•Quels sont les plus petit et plus grand entiers négatif et positif que l'on peut représenter sur 8 bits ?

Plus généralement, nous pouvons dire que pour une représentation sur n bits, il sera possible de coder des valeurs comprises entre -2^{n-1} et $+2^{n-1} - 1$

•Quelles sont les bornes inférieure et supérieure d'un entier relatif codé sur 16 bits, 32 bits, 64 bits ?

•Comment faire pour connaître la représentation de l'entier négatif représenté par le nombre positif 138 ?

1 -

15 = 0000 1111
0000 1111 → 1111 0000
1111 0000+0001=1111 0001
-15 = 1111 0001

2 -

4 = 0000 0100
5 = 0000 0101
0000 0101 →1111 1010
1111 1010+0001=1111 1011
-5=1111 1011

0100+1111 1011=1111 1111
1111 1111→0000 0000+0001=1
donc 1111 1111=-1

3 -

1111 1111= 255
Et 1111 1111 = -1

4 -

1 et 1 ?

5 -

$138-2^8$ = -118 → 0111 0110