ESIEE – Unité d'informatique IN101

REPRÉSENTATION DES NOMBRES
et ERREURS NUMÉRIQUES

Référence :

« Analyse numérique pour l'ingénieur », André Fortin, Editions de l'Ecole Polytechnique de Montréal

1- Rappel

Pour transformer un entier naturel N exprimé en base 10 dans sa représentation en base 2, il faut déterminer les chiffres binaires b_i tels que :

$$(N)_{10} = (b_{n-1}b_{n-2}\, b_{n-3} \ldots b_2b_1b_0)_2$$

c'est-à-dire

$$N = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + b_{n-3} \times 2^{n-3} + \ldots + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0$$

On obtient successivement les valeurs des bits b_0 , b_1 , b_2 , … , b_{n-2} , b_{n-1} par la méthode suivante :

```
Tant que N =/= 0 faire
        bit <- N modulo 2
        N <- N / 2      // division entière
FinTantQue
```

Exemple :

Avec N = (100)$_{10}$ on a :

100 / 2	=	*50*	*reste 0*	->	$b_0 = 0$
50 / 2	=	*25*	*reste 0*	->	$b_1 = 0$
25 / 2	=	*12*	*reste 1*	->	$b_2 = 1$
12 / 2	=	*6*	*reste 0*	->	$b_3 = 0$
6 / 2	=	*3*	*reste 0*	->	$b_4 = 0$
3 / 2	=	*1*	*reste 1*	->	$b_5 = 1$
1 / 2	=	*0*	*reste 1*	->	$b_6 = 1$

L'entier décimal 100 s'écrit 1100100 en base 2, ce qu'on notera :

(100)$_{10}$ = (1100100)$_2$

2- Nombre entiers

2.1- Représentation des entiers relatifs

La représentation en complément à 2 est la plus fréquemment utilisée. Si on dispose d'un mot de n bits ($b_{n-1}b_{n-2}\, b_{n-3} \ldots b_2b_1b_0$) pour exprimer un entier relatif N, on pose :

$$N = -b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + b_{n-3} \times 2^{n-3} + \ldots + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0$$

A noter le signe négatif devant le terme b_{n-1}. Il s'en suit :

$N >= 0 \iff b_{n-1} = 0$

$N < 0 \iff b_{n-1} = 1$

Les entiers positifs sont donc représentés par 0 suivi de leur expression en binaire naturel sur n-1 bits. Pour obtenir

la représentation d'un nombre négatif, il suffit de lui ajouter 2^{n-1} et de transformer le résultat en binaire sur n-1 bits, puis de lui adjoindre le bit b_{n-1} de valeur 1.

Sur n bits on peut représenter tout entier du segment $[-2^{n-1}, +2^{n-1}-1]$.

Exemples :

La représentation sur 4 bits de 0101 vaut :

$$-0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

soit 5 en décimal.

La représentation sur 4 bits de 1101 vaut :

$$-1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

soit $-8 + 5 = -3$.

La représentation binaire de – 6 sera 1 suivi de la représentation sur 3 bits de :

$$-6 + 2^3 = 2$$

qui est 010. On aura donc $(-6)_{10} = (1010)_2$ dans la représentation en complément à 2

2.2- Erreur de débordement arithmétique d'entier

Un débordement arithmétique d'entier survient quand un entier théorique à représenter tombe hors de la plage des valeurs représentables en machine (c'est-à-dire hors du segment $[-2^{n-1}, +2^{n-1}-1]$ de Z si on a à faire à une représentation binaire en complément à 2).

Exemples :

Supposons n = 5.

La représentation sur 5 bits de +15 vaut 01111. Ajouter 1 au nombre binaire 01111 donne 10000. Or 10000 est la représentation sur 5 bits de –16. Ainsi, par débordement arithmétique d'entier, +15 + 1 = –16 !

La représentation sur 5 bits de -16 vaut 10000. Retrancher 1 au nombre binaire 10000 donne 01111. Or 01111 est la représentation sur 5 bits de +15. Ainsi, par débordement arithmétique d'entier, –16 – 1 = +15 !

Une somme d'entiers positifs peut ainsi, suite à un débordement arithmétique, fournir une valeur négative ; de même, une somme d'entiers négatifs peut, par débordement, donner une valeur positive !

Pour des raisons d'efficacité, afin de ne pas pénaliser les temps de réponses de tout calcul, les systèmes ne surveillent pas en général les débordements arithmétiques d'entiers. Si donc un débordement se produit, il ne provoquera pas d'erreur fatale (erreur provoquant l'arrêt du programme), mais … le résultat obtenu sera faux.

Le débordement arithmétique d'entiers vérifie les propriétés suivantes :

Quel que soit k appartenant à $[1, 2^n]$ inclus dans N, on a :

$(-2^{n-1} - k)_{\text{théorique}} = (2^{n-1} - k)_{\text{machine}}$ par débordement négatif

$(2^{n-1} - 1 + k)_{\text{théorique}} = (-2^{n-1} - 1 + k)_{\text{machine}}$ par débordement positif

Si le nombre de bits normalement attribué à un entier ne suffit pas pour une application donnée, il est parfois possible d'utiliser un type d'entier plus étendu. Ainsi en Java, il existe 4 types d'entiers (de 1 à 4 octets).

La tentation existe aussi parfois de quitter les entiers pour passer en réels en espérant étendre la plage de valeurs possibles. Ce choix ne serait pas judicieux pour deux raisons (voir section suivante) :

- le nombre de chiffres significatifs de la mantisse d'un nombre réel est du même ordre de grandeur que celui d'un entier
- les calculs en réels sont susceptibles de générer en plus des erreurs d'arrondis