Codage numérique : du nombre au pixel - Cours 6

Arithmétique entière et codage des entiers relatifs

L1 – Université de Lorraine
B. Girau et N. de Rugy Altherre

Transparents disponibles sur la plateforme de cours en ligne

Addition en base 2

On veut calculer $S = A + B$ avec $A = a_{n-1} \dots a_0$, $B = b_{n-1} \dots b_0$, $S = s_{n-1} \dots s_0$.
Méthode "à la main" : on pose l'addition comme en décimal :

- on commence par les poids faibles
- on calcule chiffre par chiffre
- on somme les chiffres des deux opérandes et la retenue issue du chiffre précédent

Exemple

$1234_{10} + 567_{10} = 1801_{10}$

			1	1	1	1		1	1		
	1	0	0	1	1	0	1	0	0	1	0
+		1	0	0	0	1	1	0	1	1	1
=	1	1	1	0	0	0	0	1	0	0	1

Addition en base 2

Propagation de la retenue c_i (poids faibles en tête) :

- $c_0 = 0$
- calcul récurrent : trouver c_{i+1} et s_i dans $\{0, 1\}$ tels que $a_i + b_i + c_i = 2c_{i+1} + s_i$
  - s_i =parité$(a_i + b_i + c_i)$
  - $c_{i+1} = (a_i + b_i + c_i \geqslant 2)$

Table

$a_i b_i c_i$	000	001	010	011	100	101	110	111
s_i	0	1	1	0	1	0	0	1
c_{i+1}	0	0	0	1	0	1	1	1

Taille

entier de taille n + entier de taille n = entier de taille $n+1$
$\Longrightarrow$ dépassements de capacité éventuels (overflow)

Détection dépassement

Retenue sortante $c_n = 1$

Algorithme (addition)

$c \leftarrow 0$

pour i de 0 à $n-1$

$s_i \leftarrow \mathit{parité}(a_i + b_i + c)$

$c \leftarrow (a_i + b_i + c \geqslant 2)$

fpour

Soustraction en base 2

On veut calculer $D = A - B$ avec $A = a_{n-1} \dots a_0$, $B = b_{n-1} \dots b_0$, $D = d_{n-1} \dots d_0$.
Méthode "à la main" : on pose la soustraction comme en décimal :

- on commence par les poids faibles
- on calcule chiffre par chiffre
- on soustrait la somme de la retenue et du chiffre du second opérande au chiffre du premier opérande (s'il est trop petit, on génère une nouvelle retenue)

Exemple

$1234_{10} - 567_{10} = 667_{10}$

	1	0	0	1	1	0	1	0	0	1	0
−		1	0	0	0	1	1	0	1	1	1
	1				1	1	1	1	1	1	
=		1	0	1	0	0	1	1	0	1	1

Soustraction en base 2

Propagation de la retenue c_i (poids faibles en tête) :

- $c_0 = 0$
- calcul récurrent : trouver c_{i+1} et d_i dans $\{0, 1\}$ tels que $2c_{i+1} + a_i - b_i - c_i = d_i$
  - d_i =parité$(a_i - b_i - c_i)$
  - $c_{i+1} = (a_i - b_i - c_i < 0)$

Table

$a_i b_i c_i$	000	001	010	011	100	101	110	111
d_i	0	1	1	0	1	0	0	1
c_{i+1}	0	1	1	1	0	0	0	1

Taille

entier de taille n − entier de taille n = entier de taille n
mais problème du signe

Détection signe négatif du résultat

Retenue sortante $c_n = 1$.
Le résultat ne peut pas alors être codé en binaire simple (voir plus loin pour le codage des entiers relatifs).

Algorithme (soustraction)

$c \leftarrow 0$

pour i de 0 à $n-1$

$d_i \leftarrow \mathit{parité}(a_i - b_i - c)$

$c \leftarrow (a_i - b_i - c < 0)$

finpour

Multiplication en base 2

- addition et décalage (cf base 10)
- additions au fur et à mesure en base 2
- optimisation : on ignore les multiplications par 0
- taille : produit de deux nombres de taille $n \longrightarrow$ taille $2n$

Exemple

$26_{10} \times 13_{10} = 338_{10}$

				1	1	0	1	0
x					1	1	0	1
				1	1	0	1	0
		1	1	0	1	0	.	.
	1	0	0	0	0	0	1	0
	1	1	0	1	0	.	.	.
1	0	1	0	1	0	0	1	0

Algorithme (multiplication)

p $\leftarrow 0$
m $\leftarrow a$
pour i de 0 à $n-1$
    si $b_i = 1$ *alors*
        $p \leftarrow p+m$
    finsi
    $m \leftarrow m<<1$
finpour

Division en base 2

- soustraction et décalage (cf base 10)
- quotient et reste
- taille : quotient de la taille du 1er opérande, reste de la taille du 2ème

Exemple

$107_{10}/6_{10} = 17_{10}$ et $107_{10}\%6_{10} = 5_{10}$

1	1	0	1	0	1	1	1	1	0		
1	1	0									
		0	1	0	1	1	1	0	0	0	1
				1	0	1					

Algorithme (division)

q $\leftarrow$ 0
r $\leftarrow$ 0
pour i de $n-1$ à 0

$r \leftarrow 2r+a_i$
$q \leftarrow 2q$
si $r \geqslant b$ *alors*

$q \leftarrow q+1$
$r \leftarrow r\text{-}b$

finsi

finpour

Gestion du signe

- entiers non signés : de 0 à $2^n - 1$
- entiers signés : de $-(2^{n-1} - 1)$ à $2^{n-1} - 1$
- complément à deux (Cà2) : de -2^{n-1} à $2^{n-1} - 1$

$$x = -x_{n-1}.2^{n-1} + x_0 + x_1.2 + x_2.2^2 + \ldots + x_{n-2}.2^{n-2}$$

Cà2 : code les nombres négatifs par $2^n - |x|$ en non signé

$\longrightarrow$ nom complet : complément à deux puissance n

Cà2 : opposé

$-x$ s'obtient :

- en prenant le complémentaire de x, puis en ajoutant 1
- ou en gardant les bits de poids faible de x jusqu'au premier '1', puis en complémentant tous les suivants

Attention : il faut toujours savoir la taille du codage, sinon le codage Cà2 n'a pas de sens, et les opérateurs doivent travailler sur des opérandes de même taille

Ne pas confondre

Le complément à deux est une forme de codage à ne pas confondre avec l'opération qui consiste à calculer le complémentaire (inversion bit à bit).

Néanmoins, on peut retenir

- si x est positif, le codage de x en Cà2 sur n bits est égal à son codage en binaire non signé sur n bits
- si x est négatif, le codage de x en Cà2 sur n bits peut être obtenu en codant $|x|$ en binaire non signé sur n bits, puis en calculant l'opposé de $|x|$ par une des deux méthodes fournies

Cà2 : addition

- on ignore la dernière retenue
- détection de dépassement : les deux dernières retenues c_{n-1} et c_n doivent être identiques

Cà2 : soustraction

- on ajoute l'opposé
- détection de dépassement : idem

Cà2 : changement de taille de codage

Pour passer de n_1 à $n_2 > n_1$ bits, il suffit de rajouter des bits en poids forts tous égaux à x_{n_1-1}.
Pour passer de n_1 à $n_2 < n_1$ bits, il suffit de supprimer les $n_1 - n_2$ bits de poids forts à condition qu'il soient tous égaux à x_{n_2-1}.