IUT de Lens – BUT Info S1
R1.03 – Introduction à l'architecture des ordinateurs

TD n°2

codage des entiers relatifs

CALCULS SUR n BITS

Vous avez entendu parler de processeurs 64 bits ou 32 bits. Un processeur 64 bits effectue tous ses calculs sur 64 bits. Les générations précédentes de processeurs calculaient sur 32 bits, 16 bits ou 8 bits.
Quand on effectue les calculs sur n bits, si le résultat d'un calcul requiert plus de n bits, les bits de poids 2^i avec $i \geq n$ sont perdus. Concrètement, si on ne prend pas de précaution particulière, on calcule modulo 2^n. Exemple sur 8 bits : 250+10=0b1111'1010 + 0b0000'1010=0b0000'0100 (le 9ème bit est perdu) =4=260 mod 256.

CODAGE SIGNE ET MODULE

Si on travaille sur n bits, on peut utiliser le bit de plus fort poids (2^{n-1}) pour coder le signe (0 pour +, 1 pour -) et les $n-1$ bits restants pour coder la valeur absolue du nombre. Ce codage correspond exactement à notre notation usuelle des nombres (+12, -8). Exemples sur 8 bits : +12 est codé par 0b0'000'1100, -8 est codé par 0b1'000'1000.
Avantages/inconvénients

- (+) codage très simple
- (-) deux codages pour zéro
- (-) addition/soustraction plus compliquées
- (-) les négatifs sont codés après les positifs

Exercice 1 : Donnez le codage signe et module sur 8 bits des nombres ci-dessous.

1. +36
2. -100
3. +45
4. -21
5. -0

Exercice 2 : Donnez l'intervalle de nombres que l'on peut coder en signe et module sur n bits.

Exercice 3 : En utilisant le codage signe et module sur 8 bits, donnez le résultat des opérations ci-dessous (telles qu'elles sont réalisées par l'ordinateur)

1. +10 + (+5)
2. +5 + (-10)
3. +120 + (+10)
4. -20 + (+30)

CODAGE EN COMPLÉMENT À 2

Le complément à 1 d'un nombre x s'obtient en codant x en binaire et en inversant chaque bit (0 devient 1 et 1 devient 0). On le notera $compl_1(x)$. On pourra noter que sur n bits, $x + compl_1(x) = 2^n - 1$ (que des 1 en binaire).
Le complément à 2 d'un nombre x s'obtient en ajoutant 1 au complément à 1. On le notera $compl_2(x)$. Donc,$compl_2(x) = compl_1(x) + 1$. Le calcul $compl_2(x)$ permet d'obtenir le codage de $-x$ (l'opposé de x). En calculant sur n bits, on peut vérifier que $compl_2(compl_2(x)) = x$ (donc $-(-x) = x$) et $x + compl_2(x) = 0$ (donc $x + (-x) = 0$).
Il résulte de la définition précédente qu'on peut obtenir le complément à 2 d'un nombre en repérant le 1 le plus à droite et en inversant tous les bits qui se trouvent à sa gauche. Cas particulier : $compl_2(0) = 0$.
Un nombre positif se code en binaire comme les entiers naturels. Pour un nombre négatif, il faut coder sa valeur absolue et calculer le complément à 2 pour obtenir l'opposé. Quand on a un nombre négatif, pour le décoder, il faut calculer son complément à 2 pour obtenir la valeur absolue du nombre.
Un nombre positif a toujours son bit de poids fort à 0, un nombre négatif a toujours son bit de poids fort à 1.
Exemples sur 8 bits : +12 est codé par 0b0000'1100, -8 est codé par 0b1111'1000 (on code 8, ce qui donne 0b0000'1000, on inverse chacun des bits 0b1111'0111, et on ajoute 1 pour obtenir 0b1111'1000
Avantages/inconvénients

- (+) codage simple
- (+) un seul codage pour zéro
- (+) addition/soustraction très simple (addition binaire usuelle)
- (-) les négatifs sont codés après les positifs
- (-) il y a un nombre positif de moins que chez les négatifs

Exercice 4 : Donnez le codage en complément à 2 sur 8 bits des nombres ci-dessous.

1. +36
2. -100
3. +45
4. -21
5. -0
6. -1
7. -128
8. +127
9. -127
10. -40
11. +40

Exercice 5 : On donne ci-dessous la représentation mémoire d'entiers codés en complément à 2 (sur 8 ou 16 bits selon les cas). Donnez la valeur en base 10 de chacun de ces entiers.

1. 0xC8
2. 0x57
3. 0x9A
4. 0x0400
5. 0xFFFF
6. 0xFFC8
7. 0x8000
8. 0xFC00

Exercice 6 : Donnez l'intervalle de valeurs que l'on peut coder en complément à 2 sur n bits.

Exercice 7 : En utilisant le codage en complément à 2 sur 8 bits, donnez le résultat des opérations ci-dessous (telles qu'elles sont réalisées par l'ordinateur)

1. +10 + (+5)
2. +5 + (-10)
3. +120 + (+10)
4. -20 + (+30)
5. -128 + (+127)

Exercice 8 : Retrouvez le codage en complément à 2 des nombres +40, -40, 0, +127, -127) que vous avez calculé à l'exercice 4 . Pour chaque nombre x, calculez le complément à 2 du codage de x et vérifiez que vous obtenez bien le codage de $-x$.

Exercice 9 : Montrez que dans tous les cas, en calculant sur n bits, $compl_2(compl_2(x)) = x$ et $x + compl_2(x) = 0$.

CODAGE AVEC BIAIS (OFFSET)

On choisit de coder 0 par une valeur o (offset) et on code un nombre x par $o+x$. Si on travaille sur n bits, choisir $o = 2^{n-1}$ facilite les calculs car $(o + x) + (o + y) = x + y + 2o = x + y + 2^n = x + y$ (modulo 2^n).
Exemples sur 8 bits avec $o = 128$: +12 est codé 128+12 soit 0b1000'1100, -8 est codé par 128-8 soit 0b0111'1000.
Avantages/inconvénients

- (+) codage simple
- (+) un seul codage pour zéro
- (+) addition/soustraction très simple
- (+) les négatifs sont codés avant les positifs
- (-) il y a un nombre positif de moins que chez les négatifs
- (-) le zéro ne correspond plus à un nombre avec tous les bits à 0.
- (-) les entiers positifs ne sont plus codés comme les entiers naturels

Exercice 10 : Donnez le codage sur 8 bits avec un biais égal à 128 des nombres ci-dessous.

1. +36
2. -100
3. +45
4. -21

Exercice 11 : En utilisant le codage sur 8 bits avec un biais égal à 128, donnez le résultat des opérations ci-dessous (telles qu'elles sont réalisées par l'ordinateur)

1. +10 + (+5)
2. +5 + (-10)