Aritmética computacional

1. Introdução. Overflow.

Aritmética computacional

- Representação numérica no computador
  - Representação matemática
  - Como lidar com infinitude e continuidade?
  - Problemas resultantes: *underflow* e *overflow*
- Operações com inteiros
  - Adição e subtração
  - Multiplicação e divisão
  - *Overflow*
- Números reais de ponto flutuante
  - Representação e operaçoes

Características do sistema numérico

- Possui uma base
  - Quantidade de símbolos distintos
  - Binário, octal e hexadecimal
- Representação posicional
  - Valor do dígito varia de acordo com a posição
  - Um número

$$v = d_{n-1}d_{n-2}\ldots d_2d_1d_0$$

vale, numa base b,

$$v = \sum_{i=n-1}^{0} d_i \times b^i$$

Adição de inteiros

- Exemplo: 7 + 6

	(0)	(0)	(1)	(1)	(0)	(Carries)
. . .	0	0	0	1	1	1
. . .	0	0	0	1	1	0
. . . (0)	0 (0)	0 (0)	1 (1)	1 (1)	0 (0)	1

- Há *overflow* se houver um carry excedente
  - Somar um positivo com negativo, sem *overflow*
  - Somar dois operandos positivos
    - *Overflow* se o sinal do resultado for 1
  - Somar dois operandos negativos
    - *Overflow* se o sinal do resultado for 0

Subtração de inteiros

- Somar o negativo do segundo operando
- Exemplo: 7 – 6 = 7 + (–6)

+7:	0000 0000 … 0000 0111
–6:	1111 1111 … 1111 1010
+1:	0000 0000 … 0000 0001

- *Overflow* se o resultado for maior que o limite
  - Subtrair dois negativos ou positivos, sem *overflow*
  - Subtrair negativo de um positivo
    - Mesmo que somar dois positivos
    - *Overflow* se o sinal do resultado for 1
  - Subtrair positivo de um negativo
    - Mesmo que somar dois negativos
    - Overflow se o sinal do resultado for 0

Representando sinais - Magnitude

- O MSB é o bit de sinal
  - 0 positivo
  - 1 negativo
- Exemplo: numa arquitetura de 8 bits
  - $+19 = 0001\,0011$
  - $-19 = 1001\,0011$
- Problema:
  - $+0 = 0000\,0000$
  - $-0 = 1000\,0000$

Representando sinais - Excesso

- Considerando um sistema de 4 bits
  - O menor negativo é -8
  - Representar tudo em positivo, considerando um excesso de 8
  - Problema: não favorece operações aritméticas

Binário	Decimal	Excesso	Decimal por Excesso
0000	0	0 – 8	-8
0001	1	1 – 8	-7
0010	2	2 – 8	-6
0011	3	3 – 8	-5
0100	4	4 – 8	-4
0101	5	5 – 8	-3
0110	6	6 – 8	-2
0111	7	7 – 8	-1
1000	8	8 – 8	0
1001	9	9 – 8	1
1010	10	10 – 8	2
1011	11	11 – 8	3
1100	12	12 – 8	4
1101	13	13 – 8	5
1110	14	14 – 8	6
1111	15	15 – 8	7

Representando sinais – Complemento a 1

- O MSB é o bit de sinal
  - 0 positivo
  - 1 negativo
- Negativo: nega-se todos os bits
  - $+19 = 0001\,0011$
  - $-19 = 1110\,1100$
- Problema:
  - $+0 = 0000\,0000$
  - $-0 = 1111\,1111$

Representando sinais – Complemento a 2

- O MSB é o bit de sinal
  - 0 positivo
  - 1 negativo
- Negativo: nega-se todos os bits e soma-se 1
  - $+19 = 0001\,0011$
  - $-19 = 1110\,1101$
- Representação única do zero
  - $+0 = \quad 0000\,0000$
  - $-0 = 1\,0000\,0000$ (1 é *overflow*)
- Complemento a 2 de x numa arquitetura de n bits vale $2^n - x$.

MORGAN KAUFMANN

Lidando com *overflow*

- Números com sinal
  1. Calcula a soma (usando addu)
  2. Se os operandos tiverem sinais diferentes, não há *overflow*
  3. Se os operandos tiverem mesmo sinal e o resultado tiver sinal diferente, há *overflow*
- Números sem sinal
  1. Calcula a soma (usando addu)
  2. Se o resultado for $\geq 2^{32} - 1$, *overflow*
     - *Como verificar essa condição?*

Lidando com *overflow*

- Números com sinal

```
addu $t0, $t1, $t2

# Verifica se os sinais de $t1 e $t2 são diferentes
xor $t3, $t1, $t2

# Se o bit mais significativo de $t3 = 1, sinais diferentes
slt $t3, $t3, $zero
bne $t3, $zero, sem_overflow

# Se o sinal da soma for igual ao dos operandos, sem overflow
xor $t3, $t0, $t1
slt $t3, $t3, $zero
bne $t3, $zero, overflow
```

Lidando com *overflow*

- Números sem sinal

```
addu $t0, $t1, $t2

# Negativa $t1 (primeiro passo para complemento a dois)
nor $t3, $t1, $zero

# Com isso, $t3 = 2^{32} - $t1 - 1
# Para verificar overflow, verifica-se se $t3 < $t2
# 2^{32} - $t1 - 1 < $t2 => 2^{32} - 1 < $t1 + $t2 => overflow
sltu $t3, $t3, $t2
Bne $t3, $zero, overflow
```