

Aritmética computacional

5. Representação de ponto flutuante, operações com ponto flutuante

Infinito e NaNs

- Expoente = 111...1, Fração = 000...0
 - \pm Infinito
 - Pode ser usado em cálculos para evitar a checagem de *overflow*
- Expoente = 111...1, Fração \neq 000...0
 - Not-a-Number (NaN)
 - Indica uma operação ilegal ou um resultado não definido
 - e.g., 0.0 / 0.0

Representações IEEE 754

O padrão IEEE 754 especifica as seguintes representações especiais.

Precisão simples		Precisão dupla		Representação
Expoente	Fração	Expoente	Fração	
0	0	0	0	0
0	não zero	0	não zero	\pm número desnormalizado
1-254	qualquer coisa	1-2046	qualquer coisa	\pm número de ponto flutuante
255	0	2047	0	\pm infinito
255	não zero	2047	não zero	NaN (Not a Number)

Adição em ponto flutuante

- Exemplo:
 - $9.999 \times 10^1 + 1.610 \times 10^{-1}$
- 1. Alinhar o ponto decimal (igualar os expoentes)
 - Dar shifts à direita no número com menor expoente:
 $9.999 \times 10^1 + 0.016 \times 10^1$
- 2. Somar os significandos
 - $9.999 \times 10^1 + 0.016 \times 10^1 = 10.015 \times 10^1$
- 3. Normalizar o resultado
 - 1.0015×10^2
- 4. Arredondar e renormalizar, se necessário
 - 1.002×10^2

Adição em ponto flutuante

- Exemplo 2:
 - $1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2}$ (0.5 + -0.4375)
- 1. Alinhar o ponto decimal
 - Dar shifts à direita no número com menor expoente
 - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1}$
- 2. Somar os significandos
 - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1} = 0.001_2 \times 2^{-1}$
- 3. Normalizar o resultado
 - $1.000_2 \times 2^{-4}$, sem underflow/overflow
- 4. Arredondar e renormalizar, se necessário
 - $1.000_2 \times 2^{-4}$ (sem mudanças) = 0.0625

Multiplicação em ponto flutuante

- Exemplo
 - $1.110 \times 10^{10} \times 9.200 \times 10^{-5}$
- 1. Soma os expoentes
 - Novo expoente = $10 + -5 = 5$
- 2. Multiplica os significandos
 - $1.110 \times 9.200 = 10.212 \Rightarrow 10.212 \times 10^5$
- 3. Normaliza o resultado e checa por overflow/underflow
 - 1.0212×10^6
- 4. Arredonda e renormaliza, se necessário
 - 1.021×10^6
- 5. Determina o sinal do resultado a partir dos sinais dos operandos
 - $+1.021 \times 10^6$

Multiplicação em ponto flutuante

- Exemplo
 - $1.000_2 \times 2^{-1} \times -1.110_2 \times 2^{-2}$ (0.5×-0.4375)
- 1. Soma os expoentes
 - Sem o bias: $-1 + -2 = -3$
 - Com o bias: $(-1 + 127) + (-2 + 127) = -3 + 254 - 127 = -3 + 127$
- 2. Multiplica os significandos
 - $1.000_2 \times 1.110_2 = 1.110_2 \Rightarrow 1.110_2 \times 2^{-3}$
- 3. Normaliza o resultado e checa por over/underflow
 - $1.110_2 \times 2^{-3}$ sem overflow
- 4. Arredonda e renormaliza
 - $1.110_2 \times 2^{-3}$ sem arredondamento
- 5. Determina o sinal: $+ve \times -ve \Rightarrow -ve$
 - $-1.110_2 \times 2^{-3} = -0.21875$

Instruções em MIPS

- Há um hardware próprio para ponto flutuante: o coprocessador 1
 - Processador adjunto que expande a ISA
- Registradores separados
 - 32 registradores de 32 bits: \$f0, \$f1, ... \$f31
 - Precisão dupla aos pares: \$f0/\$f1, \$f2/\$f3, ...
- As instruções que vimos até agora não funcionam no coprocessador 1
 - O coprocessador 1 possui suas instruções próprias
- Instruções de acesso à memória
 - `load word` e `load double`
 - `lwc1`, `ldc1`, `swc1`, `sdc1`
 - e.g., `ldc1 $f8, 32($sp)`

Instruções em MIPS

- Aritmética de precisão simples
 - `add.s`, `sub.s`, `mul.s`, `div.s`
 - e.g., `add.s $f0, $f1, $f6`
- Aritmética de precisão dupla
 - `add.d`, `sub.d`, `mul.d`, `div.d`
 - e.g., `mul.d $f4, $f4, $f6`
- Comparação
 - `c.xx.s`, `c.xx.d` (`xx` is `eq`, `lt`, `le`, ...)
 - Recebe dois operações, e define o bit de condição
 - e.g. `c.lt.s $f3, $f4`
- Faz o desvio baseado no bit de condição
 - `bc1t`, `bc1f`
 - e.g., `bc1t TargetLabel`

Exemplo: °F to °C

- Código C:

```
float f2c (float fahr) {  
    return ((5.0/9.0)*(fahr - 32.0));  
}
```