

Ponto Flutuante e IEEE 754

Yuri Kaszubowski Lopes
Éverlin Fighera Costa Marques

UDESC

YKL e EFCM (UDESC)

Ponto Flutuante

1 / 19

Anotações

Números reais e o Hardware

- Converter $0,1_{10}$ para base 2, qual o problema?
 - ▶ parte fracionária é uma repetição periódica
- Observe o programa abaixo:

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main() {
5     float x = 0.0;
6
7     for (int i = 0; i < 10; i++) {
8         x += 0.1;
9     }
10
11     printf("%.20f\n", x);
12
13
14     return 0;
15 }
```

- Saída:
1.00000011920928955078

YKL e EFCM (UDESC)

Ponto Flutuante

2 / 19

Anotações

Números reais e o Hardware

- É impossível representar qualquer número real na máquina
 - ▶ Temos uma infinidade de números reais, e o hardware é finito
 - * Sempre há um número maior
 - * qualquer dois números reais há infinitos números reais
 - ▶ Armazenamos assim aproximações

YKL e EFCM (UDESC)

Ponto Flutuante

3 / 19

Anotações

Ponto Flutuante

- Uma forma de armazenar essas aproximações é através de pontos flutuantes
- Conceito implementado em grande parte dos processadores comerciais
- Similar a **notação científica normalizada**
 - ▶ O número tem **um e somente um dígito antes da casa decimal**, e **não possui zeros antes da casa decimal** (a IEEE 754 possui uma faixa de valores não normalizados também)
 - ★ $8.0_{10} \times 10^{-9}$ é normalizado
 - ★ $0.8_{10} \times 10^{-8}$ **não** é normalizado
 - ★ $80.0_{10} \times 10^{-10}$ **não** é normalizado
- Podemos fazer o mesmo com números binários
 - ▶ $1.0_2 \times 2^{-1}$ está normalizado
- Vamos chamar o ponto decimal de “ponto binário” para a base 2
- Exemplo: normalizar $1111, 11_2 = 1111, 11_2 \times 2^0$
 - ▶ $111, 111_2 \times 2^1$
 - ▶ $11, 1111_2 \times 2^2$
 - ▶ $1, 11111_2 \times 2^3 \leftarrow$ Normalizado
 - ▶ Vamos exibir a base e a potência na base 10 na disciplina para simplificar a visualização

Anotações

Exercício Resolvido

- 1 Normalize os seguintes valores binários
 - 1 11, 11₂
 - ★ = $1, 111_2 \times 2^1$
 - 2 111₂
 - ★ = $1, 11_2 \times 2^2$
 - 3 0,000001₂
 - ★ = $1, 0_2 \times 2^{-6}$

Anotações

Ponto Flutuante

- Para armazenar um valor binário normalizado arbitrário na memória, como $1, 11111_2 \times 2^3$, quais campos são importantes?
 - ▶ **Não** precisamos armazenar o 1 antes do ponto binário, nem a base.
 - ▶ Devemos armazenar:
 - ★ Sinal (0 positivo, 1 negativo)
 - ★ Expoente
 - ★ mantissa (parte à direita do ponto binário)
- Os valores então sempre tem o formato:
 - ▶ $\pm 1, mmmmmmm \times 2^{eeeeee}$
 - ★ $mmmmmm$ é a **mantissa** (ou fração)
 - ★ $eeeeee$ é o **expoente**
 - ★ Armazenamos apenas a mantissa, o expoente, e o sinal

Anotações

Padrão IEEE 754

- Utilizado na maioria dos processadores comerciais
 - x86-64, smartphones
 - Define tamanhos para os campos de expoente e mantissa
 - ★ Precisão simples (float)
 - ★ Precisão dupla (double)

Anotações

Padrão IEEE 754

- Precisão simples
- Declarado como float em C



- Precisão dupla
- Declarado como double em C



- Quais as vantagens e desvantagens do double em relação ao float?
 - (+) Consegue armazenar uma extensão maior de valores
 - (+) Maior precisão devida a mantissa ser muito maior
 - (-) Ocupa mais memória
 - (-) Pode precisar de mais ciclos do processador para efetuar cálculos

Anotações

Overflow e Underflow

- **Overflow:** o expoente é muito grande para caber na memória
- **Underflow:** o expoente é muito pequeno para caber na memória

Anotações

Valores Especiais

- Como podemos representar o número 0? Qual a dificuldade?
 - ▶ Concordamos que o 1 antes do ponto binário é implícito, e não é representado pelo hardware
 - ▶ $\pm 1, mmmmm \times 2^{e}$
 - ▶ Mas o número 0 em especial não tem 1 antes do ponto binário!
 - ▶ Essas e outras exceções são tratadas com valores especiais.

| Single precision | | Double precision | | Object represented |
|------------------|----------|------------------|----------|-----------------------------|
| Exponent | Fraction | Exponent | Fraction | |
| 0 | 0 | 0 | 0 | 0 |
| 0 | Nonzero | 0 | Nonzero | \pm denormalized number |
| 1–254 | Anything | 1–2046 | Anything | \pm floating-point number |
| 255 | 0 | 2047 | 0 | \pm infinity |
| 255 | Nonzero | 2047 | Nonzero | NaN (Not a Number) |

Valores especiais definidos no IEEE 754. Figura de Patterson e Hennessy (2017)

Anotações

Sinal do Expoente

- E o sinal do expoente? Como representar?
- Exemplo: $-1,11111_2 \times 2^{-2}$

$$-1,11111_2 \times 2^{-2}$$



$$-1,11111_2 \times 2^{-2}$$



$$-1,11111_2 \times 2^{-2}$$



- Poderíamos utilizar complemento de 2
- Ou então sinal e magnitude, como na mantissa
- ... mas não!
- O IEEE 754 especifica que o expoente utiliza uma notação com bias
 - ▶ Biased Exponent

Anotações

IEEE 754: notação com bias

- O bias é o valor intermediário entre todos os possíveis de serem representados no expoente
 - ▶ 127_{10} ($0111\ 1111_2$) para precisão simples
 - ▶ 1023_{10} ($011\ 1111\ 1111_2$) para precisão dupla
- O expoente é somado ao bias
 - ▶ Expoente -1_{10} na notação com bias se torna:
 - * $-1_{10} + 127_{10} = 126_{10} = 01111110_2$ (precisão simples)
 - * $-1_{10} + 1023_{10} = 1022_{10} = 01111111110_2$ (precisão dupla)
 - ▶ Expoente 1_{10} na notação com bias se torna:
 - * $1_{10} + 127_{10} = 128_{10} = 1000\ 0000_2$ (precisão simples)
 - * $1_{10} + 1023_{10} = 1024_{10} = 100\ 0000\ 0000_2$ (precisão dupla)
 - ▶ A notação com bias foi feita para tornar ordenações via hardware mais rápidas e simples

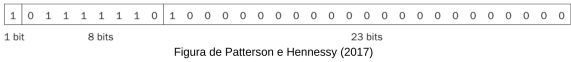
Anotações

IEEE 754

- Um ponto flutuante é então representado por:
 $(-1)^{\text{signal}} \times (1.0 + \text{mantissa}) \times 2^{(\text{expoente} - \text{bias})}$

Somente esses valores são armazenados

- Exemplo: Representar $-0,75_{10}$ em precisão simples
 - $-0,75_{10}$ pelo método das multiplicações nos dá $-0,11_2$
 - Normalizando, $-0,11_2 = -1,1_2 \times 2^{-1}$
 - A mantissa é então 1 (somente a parte fracionária)
 - O expoente é $-1_{10} + 127_{10} = 126_{10} = 0111\ 1110_2$
 - O bit de sinal é 1 (negativo)
 - Assim:

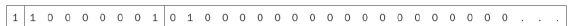


- Como ficaria em precisão dupla?
 - mantissa com 52 bits (neste caso só zeros a direita)
 - recalcular expoente: $-1_{10} + 1023_{10} = 1022_{10} = 011\ 1111\ 1110_2$

Anotações

Exemplo

- Converta o valor representado em precisão simples para decimal



- O expoente é $1000\ 0001_2 = 129_{10}$
 - $129_{10} - 127_{10} = 2_{10}$
 - A mantissa é $1,01_2 = 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 1_{10} + 0,25_{10} = 1,25_{10}$
 - O bit de sinal é 1 (negativo)
 - Logo temos: $-1,25 \times 2^2 = -5_{10}$

Anotações

Arredondamento

- A parte fracionária pode não caber na mantissa
 - O processador utilizará técnicas de arredondamento
 - Perda de precisão
- O IEEE 754 define 4 formas de arredondamento, que podem ser setadas no processador:
 - Arredondar para cima (teto)
 - Arredondar para baixo (piso)
 - Truncar (ignorar as demais casas)
 - Nearest even (par mais próximo) (mais utilizado)
- Requerem que o processador mantenha alguns bits extras para gerência
- Detalhes sobre o arredondamento e suas implementações, são encontrados em Patterson e Hennessy (2017) e Hennessy e Patterson (2014).

Anotações

IEEE 754

- O padrão para precisão simples e dupla é embutido no hardware
- Independe de linguagem
- Hardwares simples podem não implementar o padrão
 - ▶ e.g., microcontroladores
 - ▶ Nesse caso, o padrão é implementado via software

Anotações

Exercícios extras

- 1 Qual o maior e o menor valor que podem ser representados em ponto flutuante de precisão dupla e simples (desconsiderando infinito)? Quais são seus equivalentes em decimal?
- 2 Exiba os seguintes valores em ponto flutuante. Quando necessário trunque (ignore os bits que não couberem) os valores. Faça o desenho da memória como nos exemplos e coloque os endereços dos bits (para deixar claro a ordem dos bits).
 - ▶ $-16,015625_{10}$
 - * Em precisão simples
 - * Em precisão dupla
 - ▶ $-0,1_{10}$
 - * Em precisão simples
 - * Em precisão dupla
 - ▶ $0,1_{10}$
 - * Em precisão simples
 - * Em precisão dupla
 - ▶ $0,125_{10}$
 - * Em precisão simples
 - * Em precisão dupla
 - * Em meia precisão: 10 bits para mantissa, 5 para expoente e 1 para sinal

Anotações

Referências

- TOCCI, R.J.; WIDMER, N.S. **Sistemas digitais: princípios e aplicações**. 11a ed, Prentice-Hall, 2011.
- RUGGIERO, M.; LOPES, V. da R. **Cálculo numérico: aspectos teóricos e computacionais**. Makron Books do Brasil, 1996.
- NULL, L.; LOBUR, J. **Princípios Básicos de Arquitetura e Organização de Computadores**. 2014. Bookman, 2009. ISBN 9788577807666.

Anotações
