Representação em Ponto Flutuante

O sistema de representação binário é um sistema posicional. Isso quer dizer que o valor de um dígito nesse sistema depende do seu valor absoluto e de sua posição dentro do número. A mesma interpretação posicional pode ser atribuída a suas quantidades fracionárias, à direita do ponto binário. $d_0d_1...d_n\ .d_{n+1}d_{n+2}....=d_02^n+...+d_n2^0+d_{n+1}\ 2^{-1}d_{n+2}\ 2^{-2}$

Exemplo:

valor decimal de 110.01101_2

$110.01101 = 1.2^2 + 1.2^1 + 0.2^0 + 0.2^{-1} + 1.2^{-2} + 1.2^{-3} + 0.2^{-4}+1.2^{-5}$

$= 6{,}40625_{10}$

Qual a representação binária de 0.375_{10}?
Qual é 0.984_{10} na base 16?

Notação Científica

Um número R, em notação científica, pode ser representado através do seguinte formato:

R = $\pm$ M + B $\pm$ E

onde M é a mantissa;

E é o expoente; e

B é a base do expoente

Esse número pode ser escrito como

R = ($\pm$ M, B, $\pm$ E)

A base B refere-se à representação no número R e não do sistema numérico (que nos computadores tradicionais é sempre binário).

Ex.: $1228,8 = 1,2288 \times 10^3$

n = 1,2288

E = 3

B = 10

Nos sistemas computacionais, a base B é definida por hardware e não pode ser alterada pelo usuário. Assim ela pode ser omitida da representação formal: R = ($\pm$ M, $\pm$ E).

Normalização

Há várias formas de representação para os mesmos números = $0,12288.10^4 = 0,012288.10^5 = 0,0012288.10^6$ = ...

Para não haver confusão, adota-se um padrão de representação para os números em ponto flutuante, chamado normalização. Essa forma, que pretende maximizar a precisão da representação, deixa o ponto binário imediatamente à esquerda do primeiro dígito significativo, evitando assim armazenar zeros não significativos desnecessários. Para isso, a mantissa deve estar de acordo com a seguinte relação:

$$\frac{1}{B} \leq |M| < 1$$

Ex.: normalização de 103,5 . 10 (B = 10)

$103.5_{10} = 0,1035 . 10^3$ (0,1035,3)

normalização de 0.000001101 (B = 2)

$0.000001101 = 1102 \times 2^5$ (.1101,-5)

normalize 10011.110×2^{10} (B = 2)

normalize $.000011_2 \times 8^2$ (B = 8)

Normalizar:

a) $000011_2 * 8^2$ (B = 8)
   $.011 * 8^1$

b) $000101_2 * 16^5$ (B = 16)
   já está normalizado

c) representa 7.5_{10} como um número real, utilizando
   B = 2 e B = 16 e mantissa com 4 dígitos

$7.5_{10} = 111,1 \quad = 0.1111 * 2^3$
$= 0.0111_2 * 16^1$

Observações:

- Quanto maior for a mantissa, maior poderá ser a precisão dos números representados, pois poderão haver mais dígitos significativos.
- Quanto maior o expoente, maior poderá ser o escopo, ou a diferença entre o menor o maior número possível.
- A tentativa de representar um número grande demais (isto é, não suportado pelo campo de expoente) acarreta um erro de overflow. Por outro lado, a tentativa de representar, um número pequeno demais (isto é, muito próximo de zero), causa um erro de underflow.
- Algumas modificações podem ser introduzidas no projeto para aumentar a eficiência, velocidade ou precisão de um computador. Por exemplo, se um computador utilizar B = 2, a mantissa de um número representado vai ser sempre maior ou igual a ½ (0.1_2). Isso significa que o bit de mais alta ordem da representação vai ser igual a 1. Dessa forma, não há necessidade de armazenar esse bit explicitamente. Assim, é possível obter (n + 1) bits significativos com a alocação de (n) bits para a mantissa. Que problema de representação isso pode acarretar?
- Nem todas as representações de números (decimais, por exemplo) podem ser convertidas para os seus binários equivalentes. O problema é similar em outras representações:

1/3= 0.3333 ....$_{10}$
1/5 = $0,2_{10}$ = $0.001100110011_{...2}$

Não importa qual o tamanho da mantissa (a precisão da máquina) será sempre introduzido um erro de arredondamento. Esse tipo de erro é inerente à representação finita dos números.

- Para operar com números reais, de magnitudes diferentes, é preciso primeiro escalar seus expoentes, isto é, torna-los idênticos. Depois disso, pode-se operar normalmente , renormalizando o resultado.

Exemplos: (B = 10)

a) $11 + 0,25 = 0,11 * 10^2 + 0,25 * 10^2 = 11 * 10^2 + 0,0025 * 10^2 = 0,1125 * 10^2$

b) $12/0,25 = 0,12 * 10^2 / 0,0025 * 10^2 = 0,12/0,0025 = 0,48 * 10^2$

c) $12/12 = 0,12 * 10^2 / 0,12 * 10^2 = 0,12 / 0,12 = 0,1 * 10^1$

- O processo de operar com números de magnitudes diferentes pode também introduzir erros, muitas vezes denominados erros de programação.

Ex.: se tivéssemos apenas quatro bits para a mantissa e quatro para o expoente:
11 + 0,25 = (.1011,0100) + (.1000,-0001) =
= (.1011,0100) + (.0000,0100)=
= (.1011,0100) = 11

Exercícios:

1) Converta os seguintes números binários para decimais:

a) 0.110010 — 0.78125
b) 0.1110001 — 0.8828125
c) 0.101 — 0,625

2) Converta os seguintes decimais para binários:

a) 0.001 — 0.0000000001
b) 0,4 — 0.011001100...
c) 0.153827 — 0.001001110110

3) Normalize
   a) 0.000001 (B = 2)
   b) 110.01 (B = 2)
   c) 0.0001101 (B = 4)
   d) 0.0101 (B = 4)
   e) 101.101 (B = 8)
   f) 1.00001 (B = 8)

   a) 0.1×2^{-5}