Ponto Fixo e Ponto Flutuante

Arquitetura de Computadores

Introdução (1/2)

- É trivial para um computador atual tratar e operar com números inteiros.
- Entretanto, em muitas aplicações do dia a dia é necessário realizar operações com quantidades fracionárias, ou representar valores muito grandes ou muito pequenos.
- Para estas situações deve-se representar números em ponto fixo ou em ponto flutuante.

Introdução (2/2)

- De fato, a maioria dos computadores atuais já possui internamente uma (ou mais) unidade de ponto flutuante, para operar com números em notação científica.

Representação em Ponto Fixo (1/5)

- Para representar frações em ponto fixo, deve-se reservar certo número de bits para armazenar a parte fracionária.
  - Naturalmente, a quantidade de bits utilizáveis para a parte inteira diminui de forma correspondente.
- Tomando como exemplo o número binário em complemento de 2 que corresponde ao número decimal 103.

0 1 1 0 0 1 1 1

Representação em Ponto Fixo (2/5)

- Para a mesma cadeia de bits, tem-se os seguintes números, conforme a posição da vírgula:

```
0 1 1 0 0 1 1, 1 = 51,5
0 1 1 0 0 1, 1 1 = 27,75
0 1 1 0 0, 1 1 1 = 12,875
0 1 1, 0 0 1 1 1 = 3,21875
0, 1 1 0 0 1 1 1 = 0,8046875
```

Representação em Ponto Fixo (3/5)

- Para uma determinada notação em ponto fixo, indica-se somente quantos bits são usados para a fração e quantos bits representam a parte inteira.
  - Todos os números manipulados seguem então a mesma notação.
- Dos *n* bits utilizados para representar os números, empregam-se *t* bits ($t \geq 0$) para a parte inteira e *f* bits ($f \geq 0$) para a parte fracionária, com $t + f = n$.

Representação em Ponto Fixo (4/5)

- A quantidade total de valores representáveis permanece a mesma (2^n), independentemente da posição da vírgula.
- A faixa de valores representáveis depende da posição da vírgula.
- Os números fracionários não são contínuos, mas sim, estão separados entre si por uma diferença igual a 2^{-f}.

Representação em Ponto Fixo (5/5)

- As operações de soma e subtração em ponto fixo são realizadas exatamente da mesma maneira que para números inteiros.
  - Naturalmente, podem ser somados (ou subtraídos) apenas números que possuam a mesma posição para a vírgula.
  - Entretanto, os números em ponto fixo com diferentes posições para a vírgula também podem ser operados, desde que um dos números seja convertido para a representação do outro.

Representação em Ponto Flutuante (1/5)

- A faixa de números que podem ser representados em ponto fixo é insuficiente para a maioria das aplicações científicas, onde existe a necessidade de representar números muito grandes e/ou números muito pequenos.
- Para contornar este problema, desenvolveu-se a notação científica, na qual um quintilhão é representado por $1{,}0 \times 10^{18}$, em vez de escrevê-lo por extenso.
  - 1 000 000 000 000 000 000.

Representação em Ponto Flutuante (2/5)

- A representação de números em ponto flutuante é basicamente a versão binária da notação científica.
- A cada número em ponto flutuante estão associados, na realidade, três outros números: a mantissa *m*, o expoente *e* e a base *b*.
- No caso dos computadores atuais, a base utilizada é a binária, ou seja, $b = 2$.

Representação em Ponto Flutuante (3/5)

- O número em ponto flutuante é então calculado por:

$$N = m \times b^e$$

- Como a base é uma constante para um determinado sistema, o número em ponto flutuante é então representado por um par (*m*, *e*), onde *m* é uma fração ou um inteiro, e *e* é o expoente (sempre inteiro).
  - Note que ambos, mantissa ou expoente, podem ser positivos ou negativos.

Representação em Ponto Flutuante (4/5)

- A precisão de um número em ponto flutuante é determinada principalmente pelo número de bits utilizados pela mantissa.
- A faixa de representação depende do número de bits do expoente.

Representação em Ponto Flutuante (5/5)

- Os números em ponto flutuante são inerentemente redundantes, no sentido de que um mesmo número pode ser representado de maneiras diferentes.
  - Um quintilhão é representado por $1,0 \times 10^{18}$, ou $0,1 \times 10^{19}$, ou $100,0 \times 10^{16}$.
- Assim, é desejável que exista uma forma normalizada de representar um número.
  - Para tanto utiliza-se somente mantissas normalizadas.

Normalização (1/2)

- Uma mantissa está normalizada quando é constituída somente de uma parte fracionária (não existe parte inteira) e quando o primeiro dígito à direita da vírgula é diferente de zero.
- Assim, a forma normalizada de representar um quintilhão é dada por $0{,}1 \times 10^{19}$.

Normalização (2/2)

- Na base binária, a normalização da mantissa exige que seus dois bits mais significativos sejam diferentes.
  - Para números em complemento de 2, isso implica que o dígito mais significativo da mantissa e o bit de sinal sejam diferentes.
    - Para números positivos a mantissa inicia por $(0,1)_2$.
    - Para número negativos, por $(1,0)_2$.
- Um número não normalizado é normalizado facilmente por meio de deslocamentos da mantissa para a direita ou esquerda e incrementos ou decrementos do expoente, respectivamente.

Formatos de Números em Ponto Flutuante

- Existem diversos formatos adotados para representar os números em ponto flutuante.
- Muitos deles são específicos para uma família de computadores ou para um fabricante.
- Existe, porém, formato recomendado pelo IEEE (*Institute of Electrical and Electronics Engineers*), o IEEE 754.
  - O bit de sinal é representado no bit mais significativo; os bits seguintes representam o expoente e os bits menos significativos são destinados à mantissa.

Formato IEEE para Ponto Flutuante (1/6)

- A IEEE define três formatos.

	Simples (32 bits)	Duplo (64 bits)	Quádruplo (128 bits)
Sinal (S)	1 bit	1 bit	1 bit
Expoente (E)	8 bits	11 bits	15 bits
Mantissa (M)	23 bits	52 bits	112 bits

- Nesta notação, cinco grupos de números podem ser representados:
  - Números normalizados, zero, números não normalizados, infinito e não números (NaN).

Formato IEEE para Ponto Flutuante (2/6)

- Os números normalizados utilizam um expoente que vai de 1 a 254 (ou 1 a 2046, ou 1 a 32766), o primeiro bit da mantissa é sempre zero e, por isso, não é representado.
- O valor do número é calculado por

$$N = (-1)^{S} \times 2^{E-excesso} \times M$$

- O excesso é definido em 127 para 32 bits, 1023 para 64 bits e 16383 para 128 bits.

Formato IEEE para Ponto Flutuante (3/6)

- O zero é representado por um número todo em zero ($E = M = 0$).
  - Note que o zero neste caso pode ter sinal.

```
S|    E    |                  M                    |Valor
0|0000 0000|000 0000 0000 0000 0000 0000 0000 0000| +0
1|0000 0000|000 0000 0000 0000 0000 0000 0000 0000| -0
```

Formato IEEE para Ponto Flutuante (4/6)

- Números não normalizados possuem o expoente em zero (*E = 0*) e uma fração não zero.
- Seu uso é restrito para a representação de números que não podem ser normalizados sem causar *underflow*.

Formato IEEE para Ponto Flutuante (5/6)

- O infinito é representado pelo maior valor do expoente (*E* = *255* ou *2047* ou *32767*) e por uma fração em zero (*M* = *0*).
- Note que o infinito pode ter sinal.

```
S|    E    |                  M                    |Valor
0|1111 1111|000 0000 0000 0000 0000 0000 0000| +Inf
1|1111 1111|000 0000 0000 0000 0000 0000 0000| -Inf
```

Formato IEEE para Ponto Flutuante (6/6)

- Não números (NaN) são representados pelo maior expoente e por uma fração diferente de zero.
- Seu uso previsto inclui a indicação de códigos de erro, situações imprevistas, etc.

```
S|    E    |                    M                    |Valor
0|1111 1111|010 0000 0000 0000 0000 0000|  NaN
1|1111 1111|010 0000 0000 0000 0000 0000|  NaN
```

Soma e Subtração (1/3)

- Números em ponto flutuante, para serem somados ou subtraídos, devem apresentar o mesmo expoente.
- Neste caso, a soma ou subtração é realizada sobre as mantissas.
- O número resultado é formado com a mantissa resultado e o expoente dos operandos.

Soma e Subtração (2/3)

- Como os expoentes devem ser iguais antes de realizar a operação propriamente dita, expoentes diferentes precisam ser igualados.
  - O menor dos expoentes deve ser igualado ao maior, e a mantissa correspondente a este expoente tem de ser convenientemente deslocada para a direita, de forma que o número representado pelo par (mantissa, expoente) não se altere.

Soma e Subtração (3/3)

- Como os números em ponto flutuante são armazenados na forma normalizada, após a operação a mantissa resultado deve ser normalizada por meio de deslocamentos para a esquerda (ou direita) e decrementos (ou incrementos) do expoente.

Exemplo

- Para entender na prática a conversão e soma de binários em ponto flutuante, vamos fazer um exemplo passo a passo.
- Utilizando o padrão IEEE 754, vamos efetuar a seguinte operação: *25,5 + 39,72*.

Exemplo

- 1º Passo: transformar *25,5* em algo parecido com $1,x \times 2^y$.
  - Isso é alcançado através de divisões ou multiplicações: divisões quando o número é maior que 1 e multiplicações quando menor.
  - O sinal de negativo do número, caso exista, deve ser ignorado.

```
25,5 / 2    = 12,75
12,75 / 2   = 6,375
6,375 / 2   = 3,1875
3,1875 / 2 = 1,59375
```

Exemplo

- Ao final desta etapa, obtemos *R1 = 1,59375* através de 4 divisões.
- Com isso, apenas encontramos outra forma de representar *25,5* que é:

$$1,59375 \times 2^4$$

Exemplo

- 2º Passo: calcular a mantissa baseado na parte fracionária de R1.
  - Esse processo se dá sempre através de multiplicações sucessivas, até que o resultado seja 0, ou até um máximo de 23 multiplicações.
  - Como *R1 = 1,59375*, logo, sua parte fracionária é *0,59375*.

```
0,59375 × 2  = 1,1875
0,1875 × 2   = 0,375
0,375 × 2    = 0,75
0,75 × 2     = 1,5
0,5 × 2      = 1,0
0,0 × 2      = 0,0
```

Exemplo

- Mantissa calculada = 100110.
- Como a quantidade de bits da mantissa ficou inferior à 23, completa-se o final com zeros até formar 23 dígitos.
- Mantissa final:

10011000000000000000000

Exemplo

- 3º Passo: coletar as informações necessárias para criar a representação binária.
  - Sinal: Positivo = 0.
    - Se fosse negativo seria 1.
  - Expoente: 4 (calculado no 1º passo).
    - Porém, devido ao padrão estabelecido pela IEEE para números de ponto flutuante, esse expoente deve ser acrescido de 127. Assim:

$$4 + 127 = 131 = 10000011$$

Exemplo

- 4º Passo: montar a representação binária.
  - Basta juntar o sinal, o expoente e a mantissa coletados nos passos anteriores, para montar o valor de 32 bits que representa o valor inicial.
  - Assim, a representação final é:

01000001100110000000000000000000

Exemplo

- 5º Passo: efetuar os mesmos passos anteriores para o número 39,72.
  - O número 39,72 pode ser representado em notação científica como:

$$1,24125 \times 2^5$$

  - A mantissa calculada é:

00111101111000010100100 0

  - Sinal: Positivo = 0.

Exemplo

- Expoente: 5

$$5 + 127 = 132 = 10000100$$

- A representação final é:

01000010001111011100001010001000

Exemplo

- 6º Passo: Igualar os expoentes
  - Percebe-se que o expoente dos números não é o mesmo.

$$25{,}5 = 1{,}59375 \times 2^{4}$$
$$39{,}72 = 1{,}24125 \times 2^{5}$$

  - O menor dos expoentes deve ser igualado ao maior.
    - Logo, *25,5* tem que ficar com expoente 5.

Exemplo

– A diferença entre os expoentes é 1.

$$5 - 4 = 1$$

– Descola-se a mantissa 1 posição para a direita e iguala-se os expoentes.

1001100000000000000000

11001100000000000000000

1,59375 x 2^4 = **0,1**59375 x 2^5

Exemplo

– Agora as representações dos dois números estão no mesmo expoente (5).

```
25,5  = 01000010011001100000000000000000
39,72 = 01000010000111101111000010100100
```

Exemplo

- 7º Passo: efetuar a soma.
  - Com os dois números no mesmo expoente, basta somar as mantissas.

```
  0,11001100000000000000000
+ 1,00111101110000101001000
--------------------------
 10,00001001110000101001000
```

  - Nota-se que o resultado não ficou normalizado.

Exemplo

- 8º Passo: normalizar o resultado.
  - Para normalizar o resultado, desloca-se a mantissa para a direita e elimina-se o bit excedente.

$$1,00000100111000010100100 0$$

  - Como a mantissa foi deslocada para a direita, incrementa-se o expoente.

$$10000100 + 1 = 10000101$$

Exemplo

- Como resultado final, temos a seguinte representação:

01000010000001001110000101001100

- Que equivale à notação científica:

$$1,0190625 \times 2^{6}$$

- Que denota o número decimal:

$$65,22 = 25,5 + 39,72$$

Exemplo

- 9º Passo: extrair os campos.

01000010000001001110000101001 00

  - Sinal:

0 = Positivo

  - Expoente:

10000101

  - Mantissa:

00000100111000010100100

Exemplo

- 10º Passo: calcular o expoente real.

$$(10000101)_2 = (133)_{10}$$

  - Devido ao padrão estabelecido pela IEEE para números de ponto flutuante, o expoente foi acrescido de 127 durante a conversão original.
  - Por isso, deve-se remover 127 do expoente encontrado para obter o expoente real.

$$133 - 127 = \mathbf{6}$$

Exemplo

- 11º Passo: calcular o valor decimal da mantissa.
  - Basta multiplicar cada bit da mantissa pela potência de 2 correspondente àquela posição, e somar todos os resultados.
  - Esse método é análogo a alguns métodos de conversão de números binários (inteiros) para decimal, com uma diferença: aqui todas as potências são negativas, pois está se calculando uma fração e não um número inteiro.
    - O primeiro bit deve ser multiplicado por 2^{-1}, o segundo por 2^{-2}, o terceiro por 2^{-3}, e assim por diante.

Exemplo

00000100111000010100100

$0 \times 2^{-1} = 0$
$0 \times 2^{-2} = 0$
$0 \times 2^{-3} = 0$
$0 \times 2^{-4} = 0$
$0 \times 2^{-5} = 0$
$1 \times 2^{-6} = 0,015625$
$0 \times 2^{-7} = 0$
$0 \times 2^{-8} = 0$
$1 \times 2^{-9} = 0,001953125$
$1 \times 2^{-10} = 0,0009765625$
$1 \times 2^{-11} = 0,00048828125$
$0 \times 2^{-12} = 0$
$0 \times 2^{-13} = 0$
$0 \times 2^{-14} = 0$
$0 \times 2^{-15} = 0$
$1 \times 2^{-16} = 0,000015258789625$
$0 \times 2^{-17} = 0$
$1 \times 2^{-18} = 0,000003814697265625$
$0 \times 2^{-19} = 0$
$0 \times 2^{-20} = 0$
$1 \times 2^{-21} = 0,000000476837158203125$
$0 \times 2^{-22} = 0$
$0 \times 2^{-23} = 0$

Exemplo

– Somam-se todos os resultados diferentes de zero.

```
  0,01562500000000000000
  0,00195312500000000000
  0,00097656250000000000
+ 0,00048828125000000000
  0,00001525878906250000
  0,00000381469726562500
  0,00000047683715820312 5
-------------------------
  0,01906251907348632812 5
```

Exemplo

- 12º Passo: montar a representação decimal.
  - Basta juntar o sinal, o expoente e a mantissa, para montar o valor decimal na forma $1,x \times 2^{y}$.
  - Assim, a representação final é:

$$+1,0190625190734863281 25 \times 2^{6}$$

  - Ainda, pode-se calcular a potência e a multiplicação.

$$65,22000122070312 5$$