LCC 2019/20

Sistemas de Computação

Práticas Laboratoriais

Aula 3: Representação de números reais
Fevereiro 2020

António Esteves
António Pina
Bruno Medeiros

Números fracionários

- Qual o número decimal correspondente a $1011,101_2$?
  - Parte inteira $\Rightarrow$ $1011 \rightarrow 11_{10}$
  - Parte fracionária $\Rightarrow$ $101 \rightarrow 5 / 2^3 = 0,625$
  - Valor = $11,625_{10} = 11625 \times 10^{-3}$

- Ou seja, se pudermos ajustar a posição da vírgula podemos tratar os número reais da mesma forma que tratamos os números inteiros

Números fracionários

- Mais exemplos:
  - 5 ¾
    - $101{,}11_{2}$
  - 2 5/8
    - $10{,}101_{2}$
  - 63/64
    - $0{,}111111_{2}$

Limitações da representação fracionária em binário

- Só conseguimos representar exatamente valores que podem ser escritos na forma **$x / 2^y$**

- Exemplos de valores que não podem ser representados exatamente:

  Valor Representação

  -------- ----------------------------------

  - 1/3 $0{,}01010101[01]_2$
  - 1/5 $0{,}001100110011[0011]_2$
  - 1/10 $0{,}0001100110011[0011]_2$

Representação em vírgula fixa

- Nesta representação fixamos a vírgula (que separa a parte inteira da fracionária) numa posição pré-definida
- Por exemplo, usando vírgula fixa com 8 bits:
  - **EX1:** a vírgula binária fica entre os bits 2 e 3
    - b7 b6 b5 b4 b3 , b2 b1 b0
  - **EX2:** a vírgula binária fica entre os bits 4 e 5
    - b7 b6 b5 , b4 b3 b2 b1 b0
- No **EX1** podemos representar números até **31,875**
- No **EX2** podemos representar números até **7,96875**
- Há um compromisso entre a **precisão** e a **gama** de valores que podemos representar [Min, Max]

Representação em vírgula fixa

- Que representação escolhemos ? Ou seja, onde colocamos a vírgula?
  - Irá depender da precisão e da gama que queremos
- Na representação em vírgula fixa existe um compromisso entre a gama de valores que se vai poder representar e a precisão dos valores
  - Uma gama maior implica menos precisão e vice-versa

Vírgula fixa em computador

- **Exemplo:** pretendemos representar números reais negativos e positivos com 8 bits em vírgula fixa
  - Usamos:
    - 1 bit para o sinal (vermelho)
    - 3 bits para a parte inteira (azul)
    - 4 bits para a parte fracionária (verde)
  - Representação de $11{,}11_2$
    - 00111100

Exemplo (continuação)

- 0 000 0001 = 1/16
- 0 000 0010 = 2/16
- 0 000 0011 = 3/16
- ...
- 0 000 1111 = 15/16 (número mais próximo de 1)
- 0 001 0000 = 1,0
- ....
- 0 111 1111 = 7,9375 (⇔ 7 15/16)

Exemplo (continuação)

- No exemplo anterior, a distância entre dois valores consecutivos é sempre a mesma → 1/16 = 0,0625
- Veremos mais à frente que no caso da **vírgula flutuante** a distância não é sempre a mesma

Vírgula fixa

- **Vantagens**:
  - É fácil de utilizar
  - O *hardware* que faz a aritmética de inteiros também faz a aritmética de reais
- **Desvantagens**:
  - Não é fácil decidir onde colocar a vírgula
  - Por vezes queremos mais precisão, outras vezes queremos uma gama maior

Vírgula fixa vs. Vírgula flutuante (FP)

- **Vírgula fixa**
  - Esta representação não é muito utilizada
- A norma para a representação binária de números reais é a **Vírgula flutuante** (*Floating Point* ou FP)
  - Permite flutuar/variar a vírgula binária da seguinte forma:
    - Quando queremos representar números muito pequenos pedimos bits "emprestados" à parte inteira
    - Quando queremos representar números muito grandes pedimos bits "emprestados" à parte fracionária

Notação científica

- No dia a dia deparamo-nos com situações em que podemos ter que lidar com números muito grandes e/ou muito pequenos
  - Diâmetro da terra:
    - 12 800 000 metros
  - Massa de uma molécula de água:
    - 0,00 000 000 000 000 000 000 00299 gramas
  - Como se pode verificar, não é prático trabalhar com números com grandes quantidades de algarismos
  - Mas, podemos escrever estes números usando potências de 10

Notação científica

- Diâmetro da terra: $1,28 \times 10^{7}$ m
- Molécula de água: $2,99 \times 10^{-23}$ g
- Facilita as operações aritméticas. Exemplo:

  300 000 000

  x 0,000 000 15 **?**
- Em notação científica é bem mais fácil responder:

  3×10^{8}

  x $1,5 \times 10^{-7}$

  $= (3 * 1,5) * 10^{(8-7)} = 4,5 \times 10^{1}$

Notação científica

- Em notação científica um número é representado na forma:
  - **A** x $10^{\mathbf{b}}$
    - Em que **b** é a **ordem de grandeza** (um número inteiro)
    - **A** é a **mantissa** (um número real)
    - Se o número é negativo, coloca-se o **sinal** '-' à esquerda de A
  - Este formato permite que o mesmo número possa ser representado de várias formas diferentes
  - Por exemplo, 350 pode ser representado como:
    - 3,5 x 10^2
    - 35 x 10^1
    - 350 x 10^0
    - ...

Notação científica normalizada

- Para facilitar a sua representação, impõe-se um formato específico **normalizado**:
  - **$1 \leq |A| < 10$**
  - Ou seja, existe sempre um único dígito, e diferente de zero, antes da vírgula
- No caso de 350, em notação científica normalizada fica:
  - $3{,}5 \times 10^2$ $(1 \leq |3| < 10)$
- Esta forma facilita comparações entre números, uma vez que o expoente **b** é a sua ordem de grandeza

Notação científica normalizada

- É mais fácil comparar números na **forma normalizada**:

  $3{,}5 \times 10^2$ com $4{,}5 \times 10^2$

  do que na **forma não normalizada**:

  35×10^1 com $4{,}5 \times 10^2$

Problema da notação científica normalizada

- Consideremos o caso da representação com 7 algarismos: 5 para a mantissa e 2 para o expoente

- Como representamos os números no intervalo $[0{,}0000\times10^{-99} : 1{,}0000\times10^{-99}[$?

  - Na forma normalizada não conseguimos representar estes números → desperdiça-se este intervalo

- Solução: utilizar a **notação desnormalizada**

$$\mathbf{0{,}nnnnn \times 10^{expoente}}$$

Da notação científica para vírgula flutuante

- A notação científica do tipo
  - **Valor = $(-1)^{s}$ x Mantissa x Radix^{Exp}**
  - é a que permite representar melhor os números reais em vírgula flutuante
- **Radix** é a base
  - **10** em decimal e **2** (ou uma **potência de 2**) em binário; mas vamos apenas tratar o caso **2,** que é a norma atual
- **S** é o sinal
  - S = 0 para número positivos → $(-1)^{0} = 1$
  - S = 1 para número negativos → $(-1)^{1} = -1$

Da notação científica para vírgula flutuante

- A **Mantissa** será
  - **Y,xxxx** (com $1 \leq Y < radix$, no caso normalizado)
  - **0,xxxx** (no caso desnormalizado)
- **Exp** é o expoente

Vírgula flutuante

- Vai permitir representar tanto valores muito grandes como muito pequenos ao mesmo tempo, ao contrário da vírgula fixa
- O truque é representar o número real na sua forma científica
- Exemplo:
  - $25,7 \times 10^{4} = 257000$
  - $25,7 \times 10^{-1} = 2,57$
  - $25,7 \times 10^{-2} = 0,257$
  - Como podemos ver o **expoente** é responsável pela **flutuação da vírgula** decimal
  - No caso dos binários vai acontecer o mesmo em relação à vírgula binária

Vírgula flutuante

- Para podermos representar um número real em computador, com o mesmo estilo da representação científica
  - Temos que fixar a vírgula binária
  - Flutuamos a vírgula mexendo apenas no expoente
  - Usamos a forma normalizada, que no caso binário é:

    $1,yyyy \times 2^{exp}$

Vírgula flutuante

- O formato binário usado para representar valores em FP usa 3 campos:
  - **Sinal S –** Fica mais à esquerda, o que permite usar o mesmo hardware (que trabalha com inteiros) para testar o sinal de um valor em FP
  - **Expoente E –** Fica a seguir ao sinal
    - Permite fazer comparações quanto à grandeza entre valores absolutos em FP, sem a necessidade de separar os 3 campos. Basta comparar os valores como se de valores meramente binários se tratasse
  - **Parte fracionária F –** É o campo mais à direita

Bit escondido

- Sabemos que um valor normalizado tem sempre o dígito mais à esquerda diferente de zero
  - Se for no sistema decimal podemos ter 9 valores diferentes
  - Mas em binário só temos um valor possível = 1
  - Logo podemos omitir este bit durante a representação interna de um FP em binário
  - Ganha-se um bit extra para melhorar a precisão → este bit será usado na parte fracionária

A norma IEEE 754 para vírgula flutuante

- Esta norma foi estabelecida em 1985 e define a representação e aritmética de números em vírgula flutuante

- Antes desta norma havia formatos diferentes, os quais era difícil combinar

- A norma é suportada pela grande maioria dos CPUs atuais

- Permite portabilidade dos dados entre diferentes sistemas computacionais

Aspectos relevantes da norma IEEE 754

- **Representação do sinal**
  - O bit mais à esquerda representa o sinal: 0=positivo e 1=negativo
- **Parte fracionária**
  - Representa-se o módulo da parte fracionária
  - Utiliza-se o **bit escondido** na representação de valores normalizados

Aspectos relevantes da norma IEEE 754

- **Representação do expoente**
  - Para facilitar o processo de comparação de números reais, sem obrigar a separar os campos da notação, o expoente é codificado da seguinte forma:
    - Os expoentes menores (os negativos) terão uma representação binária menor do que os expoentes maiores (os positivos)
  - Sinal e magnitude ?
  - Complemento para 1 ?
  - Complemento para 2 ?
  - Nenhum destas 3 representações possui a propriedade anterior porque usam um 1 no bit mais à esquerda (sinal) nos números negativos, o que torna a sua representação binária maior do que a dos números positivos

Representação do expoente

- **Utilização da notação em Excesso**
  - Vai permitir que o zero seja representado por um valor no meio da tabela e não por 00..00
  - Usa-se o excesso de $\mathbf{2^{n-1} - 1}$
- Formato global da norma IEEE 754:

| S | Expoente | Mantissa | **ou** **| S | E | F |**

Norma IEEE 754

- **O valor decimal de um número binário, representado em v. f. normalizada, é dado por**
  - $V = (-1)^{s} \times (1,F) \times 2^{(E - Excesso)}$
  - Em que S, F e E representam respetivamente o valor binário do sinal, da mantissa e do expoente da norma
- **Representação de valores desnormalizados**
  - A norma reserva as combinações com **E**=00..00 e **F**≠00..00 para número desnormalizados
  - O valor decimal representado é dado por:

    $V = (-1)^{S} \times (0,F) \times 2^{(1-Excesso)}$

  - Porquê 1-Excesso? Veremos mais à frente a razão!

Norma IEEE 754 (casos especiais)

- **Representação de zero**
  - Expoente = 00..00 e parte fracionária = 00..00
- **Representação de infinito**
  - Expoente = 11..11 e parte fracionária = 00..00
    - Com sinal = 0 → +∞
    - Com sinal = 1 → -∞
- **Representação de NaN** (***N****ot* ***A*** ***N****umber*)
  - Expoente = 11..11 e parte fracionária diferente de 00..00
  - Exemplo: raiz quadrada de -1

Norma IEEE 754

- **32 bits** (vírgula flutuante de **precisão simples**)
  - 1 bit para sinal
  - 8 bits para expoente
  - 23 bits para a parte fracionária
  - Expoente em excesso de $2^{8-1}-1 = 127$
- **64 bits** (vírgula flutuante de **precisão dupla**)
  - 1 bit para sinal
  - 11 bits para expoente
  - 52 bits para a parte fracionária
  - Expoente em excesso de $2^{11-1}-1 = 1023$

Exemplo prático

- Vamos usar um exemplo apenas com 8 bits, para se perceber melhor, mas seguindo as mesmas regras que a norma IEEE 574:
  - **1** bit para o sinal
  - **3** bits para o expoente
  - **4** bits para a mantissa

Passar de decimal para representação binária FP

- Número: 0,5625

1) Converter para binário

   0,1001

2) Normalizar o binário

   $1{,}001 \times 2^{-1}$

3) Identificar o sinal do valor

   Positivo → S = 0

- 0 _ _ _ _ _ _ _

Passar de decimal para representação binária FP

4) Representar a mantissa sem bit escondido

001 (em 3 bits) passa a 0010 (em 4 bits)

0 _ _ _ 0010

5) Calcular o Expoente (E)

Calcular o valor do excesso: $2^{n-1} - 1 = 2^{3-1} - 1 = 3$

- Expoente = -1 + excesso = -1 + 3 = 2
- Converter Expoente para binário $2_{10} = 010_2$ (3 bits)

- 0.5625 em formato binário vírgula flutuante fica:
  - 0 010 0010

Passar de FP para decimal

- Utilizamos a fórmula
  - $V = (-1)^{S} \times 1{,}F \times 2^{(E - excesso)}$
- Calcular o valor decimal de 0 010 0010
- S = 0 (bit mais à esquerda)
- F = 0010
- $E = 010_2 = 2_{10}$
  - Temos que retirar ao expoente o excesso (3)
  - Uma vez que o expoente é guardado em excesso de 3, temos que subtrair este excesso ao valor guardado:
  - Expoente = 2 – 3 = -1

Passar de FP para decimal

- Utilizamos a fórmula

  $V = (-1)^{S} \times 1,F \times 2^{(E - excesso)}$

- Substituímos os valores de S, F e E nesta fórmula

$V = (-1)^{0} \times 1,001 \times 2^{(2 - 3)}$

$= 1 \times 1,001 \times 2^{-1}$

$= 1 \times 1,001 \times 0,5$

$=$ 0,5005 → Porque é que não deu 0,5625?

$= 1 \times 1,001 \times 2^{-1} =$

$0,1001_{2}$ ou $1,001_{2} \times 0,5_{10}$ ou $1,125_{10} \times 0,5_{10} = 0,5625$

Casos especiais

- 1 111 0000 ?
  - E = 111 e F = 0000 e S=1 → **-∞**
- 0 111 0000 ?
  - E = 111 e F = 0000 e S=0 → **+∞**
- Não vale a pena tentar converter os padrões de bits em cima porque não faz sentido. Estes padrões estão reservados pela norma IEEE 754 para representar infinito

Casos especiais

- 0 111 0010 ?
  - E (expoente) = 111 e F ≠ 0000 → **NaN** (por ex. 0/0)
- Significa que o maior valor que podemos representar neste formato é 0 110 1111 ?
  - $(-1)^0 \times (1{,}1111) \times 2^{(6-3)} = 1{,}9375 \times 8 = 15{,}5$
  - E (expoente) não pode ser 111 senão o valor seria NaN
- 0 000 0000
  - E = 000 e F = 0000 a norma diz que o valor é **zero**

Números desnormalizados

- O maior número positivo desnormalizado ?

  0 000 1111

- O menor número positivo desnormalizado?

  0 000 0001

- Porque se usa Expoente=1–excesso em vez de Expoente=0–excesso?
  - Desta forma o salto entre o maior desnormalizado e o menor normalizado positivo é o mesmo que entre dois valores desnormalizados consecutivos

Distância entre valores FP consecutivos

- Números desnormalizados:

0 000 0001 = 1/64

0 000 0010 = 2/64

. . .

0 000 1111 = 15/64

A distância entre dois números consecutivos desnormalizados é sempre a mesma → neste caso 1/64

Distância entre valores FP consecutivos

- Números normalizados:

0 001 0000 = 0,25

0 001 0001 = ? (0,25+1/64)

0 001 0010 = ? (0,25+2/64)

. . .

0 001 1111 = ? (0,25+15/64)

0 010 0000 = 0,5

0 010 0001 = ? (0,5+1/32)

0 010 0010 = ? (0,5+2/32)

. . .

A partir de E=2, a distância entre 2 valores consecutivos duplica. Então, sempre que o expoente aumenta a distante entre 2 valores aumenta.

Distância entre valores FP consecutivos

- A ideia é ter maior precisão nos valores próximo de zero do que nos valores mais afastados de zero.