#3 : Representação de Números em Vírgula Flutuante

Computer Architecture 2021/2022

Ricardo Rocha

Computer Science Department, Faculty of Sciences, University of Porto

Números Fracionários

Os números fracionários são compostos por uma **parte inteira** (antes do ponto decimal) e por uma **parte fracionária** (após o ponto decimal) que pode ser igualmente **representada em potências da base**.

$$123.75 = 1\text{x}10^{2} + 2\text{x}10^{1} + 3\text{x}10^{0} + 7\text{x}10^{-1} + 5\text{x}10^{-2}$$

Em decimal, a sequência de algarismos de um número fracionário tem o valor da respetiva potência de 10, como se segue:

$$\ldots\ 10^{4}\ 10^{3}\ 10^{2}\ 10^{1}\ 10^{0}\ .\ 10^{-1}\ 10^{-2}\ 10^{-3}\ 10^{-4}\ \ldots$$

Em binário, a sequência tem o valor da respetiva potência de 2:

$$\ldots\ 2^{4}\ 2^{3}\ 2^{2}\ 2^{1}\ 2^{0}\ .\ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ \ldots$$

Conversão de Base

Partindo duma representação decimal, a conversão de base da parte fracionária é conseguida pela **realização de multiplicações sucessivas pela nova base até se atingir uma parte fracionária igual a zero ou ser identificada uma repetição** (dízima infinita). A representação do valor na nova base é a concatenação das partes inteiras obtidas no processo.

$6.625_{10} = 6_{10} + 0.625_{10}$

$6_{10} = 110_2$

$0.625_{10} = 0.101_2$

$0.625_{10} \times 2 = 1.25_{10}$

$0.25_{10} \times 2 = 0.5_{10}$

$0.5_{10} \times 2 = 1.0_{10}$

$6.625_{10} = 110_2 + 0.101_2 = 110.101_2$

Conversão de Base

$0.35_{10} = 0.01(0110)_2$

Repetição

$0.35_{10} \times 2 = 0.7_{10}$

$0.7_{10} \times 2 = 1.4_{10}$

$\boxed{0.4_{10} \times 2 = 0.8_{10}}$

$0.8_{10} \times 2 = 1.6_{10}$

$0.6_{10} \times 2 = 1.2_{10}$

$0.2_{10} \times 2 = 0.4_{10}$

$\boxed{0.4_{10} \times 2 = 0.8_{10}}$

Aritmética (Adição)

Independentemente da base usada, o **algoritmo usado para operações aritméticas é o mesmo**.

$100.1_2 + 110.11_2 = 1011.01_2$

$$
\begin{array}{r}
\color{red}{1\,0\,0\,1\ \ 0\ \ } \\
1\,0\,0\,.\,1\,0 \\
+\ 1\,1\,0\,.\,1\,1 \\
\hline
1\,0\,1\,1\,.\,0\,1
\end{array}
$$

Números em Vírgula Flutuante

$$(-1)^s \times 1.m_2 \times 2^e$$

Um número diz-se **representado em vírgula flutuante** quando está em notação científica e normalizado (sem zeros antes do ponto decimal).

$$110.11_2 = 1.1011_2 \times 2^2$$

Uma representação específica de N bits deve definir o tamanho da **mantissa (m)** e o tamanho do **expoente (e)**:

- Ao tamanho da mantissa corresponde a **precisão da representação**
- Ao tamanho do expoente corresponde o **alcance dos números representados**

Formato IEEE 754

Precisão simples (32 bits)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>s</td><td colspan="8">exponent</td><td colspan="23">fraction</td></tr>
<tr><td>1 bit</td><td colspan="8">8 bits</td><td colspan="23">23 bits</td></tr>
</table>

Precisão dupla (64 bits)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>s</td><td colspan="11">exponent</td><td colspan="20">fraction</td></tr>
<tr><td>1 bit</td><td colspan="11">11 bits</td><td colspan="20">20 bits</td></tr>
<tr><td colspan="32">fraction (continued)</td></tr>
<tr><td colspan="32">32 bits</td></tr>
</table>

Bit de sinal: ‘0’ número positivo, ‘1’ número negativo.

Bits de expoente: em excesso de 127 para precisão simples e em excesso de 1023 para precisão dupla.

Bits de mantissa: sem o bit da parte inteira pois é sempre 1.

Formato IEEE 754

$$(-1)^s \times (1+m) \times 2^{e_{10}-bias}$$

$-0.75 = -0.11_2$

$= -1.1_2 \times 2^{-1}$ (normalizado)

$= (-1)^1 \times (1.1000\ 0000\ 0000\ 0000\ 0000\ 000_2) \times 2^{126-127}$ (precisão simples)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 bit   8 bits   23 bits

Formato IEEE 754

Sinal (s)	Expoente (e)	Mantissa (m)	Valor
0/1	0 ... 0	0 ... 0	zero
0/1	0 ... 0	$a_1 \ldots a_{23}$	número não normalizado ($\exists i : a_i \neq 0$)
0/1	0 ... 01 - 1 ... 10	$a_1 \ldots a_{23}$	número normalizado
0/1	1 ... 1	0 ... 0	infinito
0/1	1 ... 1	$a_1 \ldots a_{23}$	NaN (Not a Number) ($\exists i : a_i \neq 0$)

Aritmética em IEEE754 (Adição)

Algoritmo de adição em vírgula flutuante para números em formato IEEE754:

1. Igualar os expoentes deslocando a vírgula do menor valor menor
2. Fazer a adição
3. Normalizar o resultado
4. Verificar se temos overflow/underflow (expoente não representável)
5. Arredondar ao número de bits
6. Caso não esteja normalizado voltar ao passo 3

Aritmética em IEEE754 (Adição)

$\mathbf{3.25_{10} + 0.75_{10}}$

$\mathbf{3.25_{10} = 11.01_2 = 1.101_2 \times 2^1}$

$\mathbf{0.75_{10} = 0.11_2 = 1.1_2 \times 2^{-1}}$

Passo 1: $\mathbf{1.101_2 \times 2^1 + 0.011_2 \times 2^1}$

Passo 2: $\mathbf{10.000_2 \times 2^1}$

Passo 3: $\mathbf{1.0000_2 \times 2^2}$ **(fim)**

$\mathbf{11.01_2 + 0.11_2 = 1.0_2 \times 2^2}$

Regras de Arredondamento

Round to the nearest: arredonda para o mais próximo em função do valor dos 3 bits extra usados para esse propósito:

- **guard/round bits**: 2 primeiros bits à direita da posição a arredondar
- **sticky bit**: marca a existência de outros bits com o valor 1 à direita do round bit
- ***m0XX*** $\longrightarrow$ ***m*** (arredonda para baixo)
- ***m11X*** $\longrightarrow$ ***m + 0… 01*** (arredonda para cima)
- ***m101*** $\longrightarrow$ ***m + 0… 01*** (arredonda para cima)
- ***m100*** $\longrightarrow$ ***m + 0… 01 / m*** (arredonda para cima se o bit menos significativo de m for 1, caso contrário arredonda para baixo)