

**UNIVERSIDADE FEDERAL DA PARAÍBA**

**CENTRO DE INFORMÁTICA**

**ENGENHARIA DE COMPUTAÇÃO**

**ARQUITETURA DE COMPUTADORES**

**Relatório – Somador de Ponto Flutuante**

Nathália de Vasconselos -

Thiago Gonzaga Gomes - 11504760

Orientador: Prof. Dr. Hamilton Soares da Silva

João Pessoa – 13 de abril de 2017

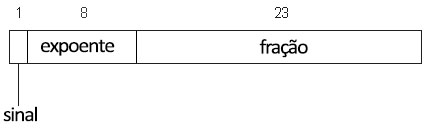
**Introdução e Fundamentação**

Neste relatório, será apresentada uma análise do funcionamento de um somador de ponto flutuante (FPU). *A utilização da notação de ponto flutuante é muito grande em computadores, tanto para cálculos matemáticos com precisão, renderização de imagens digitais (criação de imagens pelo computador) entre outros. Os processadores atuais se dedicam muito em aperfeiçoar a técnica de seus chips para a aritmética de ponto flutuante, devido à demanda de aplicativos gráficos e jogos 3D que se utilizam muito deste recurso.*

Ponto flutuante binário é uma forma de representação de um número real, englobando também números irracionais, cujas frações decimais são infinitas. Para tal, existem três precisões: a precisão simples (32 bits), a precisão dupla (64 bits) e o menos utilizado, a precisão dupla estendida (80 bits).

Para a **precisão simples** (32 bits): 1 bit de sinal, 8 bits de expoente e 23 bits para a parte fracionária do significando.  
Para a **precisão dupla** (64 bits): 1 bit de sinal, 11 bits de expoente e 52 para a parte fracionária do significando.  
Para a **precisão dupla estendida** (80 bits): 1 bit de sinal, 16 bits de expoente e 63 bits para a parte fracionária do significando.

No formato que será focado neste relatório, o formato de **precisão simples**, há o intervalo normalizado aproximado: 2–126 a 2127. Também chamado de short real.

  
*fração = mantissa*

**Componentes de um real de precisão simples**

**Sinal**: 1 = negativo, 0 = positivo.

**Significado**:  
-Dígitos decimais à esquerda e à direita do ponto decimal.  
-Notação posicional ponderada.

*Exemplo:*   
123.154 = (1 x 102) + (2 x 101) + (3 x 100) + (1 x 10–1) + (5 x 10–2) + (4 x 10–3)

**Expoente**:  
-Inteiros sem sinal.  
-*Bias* inteiro (127 para precisão simples)

**Frações decimais x ponto flutuante binário**

|  |  |
| --- | --- |
| **Ponto Flutuante Binário** | **Fração na Base 10** |
| 11.11 | 3 3/4 |
| 101.0011 | 5 3/16 |
| 1101.100101 | 13 37/64 |
| 0.00101 | 5/32 |
| 1.011 | 1 3/8 |
| 0.00000000000000000000001 | 1/8388608 |

|  |  |  |
| --- | --- | --- |
| **Binário** | **Fracção na Base 10** | **Valor Decimal** |
| .1 | 1/2 | .5 |
| .01 | 1/4 | .25 |
| .001 | 1/8 | .125 |
| .0001 | 1/16 | .0625 |
| .00001 | 1/32 | .03125 |

**Expoente:** para produzir o expoente polarizado (biased), soma-se 127.

|  |  |  |
| --- | --- | --- |
| **Expoente (E)** | ***Biased* (E + 127)** | **Binário** |
| +5 | 132 | 10000100 |
| 0 | 127 | 01111111 |
| -10 | 117 | 01110101 |
| +127 | 254 | 11111110 |
| -126 | 1 | 00000001 |
| -1 | 126 | 01111110 |

No **ponto flutuante binário normalizado**, a mantissa é normalizada quando **1** aparece à esquerda do ponto binário.

No **ponto flutuante binário não-normalizado**, o ponto binário pode variar até que o expoente seja **0**.

Ex:

|  |  |
| --- | --- |
| **Não-normalizado** | **Normalizado** |
| 1110.1 | 1.1101 x 23 |
| .000101 | 1.01 x 2-4 |
| 1010001. | 1.01001 x 26 |

Tendo em vista os conceitos anteriores, têm-se como ordem de exibição o **sinal,** seguido do **expoente** e da **fração**.

Ex:

|  |  |  |
| --- | --- | --- |
| **Valor Binário** | **Expoente *Biased* (+127)** | **Sinal, Expoente, Fração** |
| -1.11 | 127 | 1 01111111 11000000000000000000000 |
| +1101.101 | 130 | 0 10000010 10110100000000000000000 |
| -.00101 | 124 | 1 01111100 01000000000000000000000 |
| +100111.0 | 132 | 0 10000100 00111000000000000000000 |
| +.0000001101011 | 120 | 0 01111000 10101100000000000000000 |

**Convertendo frações para reais binários**(Expressar como uma soma de frações tendo denominadores que são potências de 2).

Ex:

|  |  |  |
| --- | --- | --- |
| **Fração Decimal** | **Fatorado como...** | **Binário real** |
| 1/2 | 1/2 | .1 |
| 1/4 | 1/4 | .01 |
| 3/4 | 1/2 + 1/4 | .11 |
| 1/8 | 1/8 | .001 |
| 7/8 | 1/2 + 1/4 + 1/8 | .111 |
| 3/8 | 1/4 + 1/8 | .011 |
| 1/16 | 1/16 | .0001 |
| 3/16 | 1/8 + 1/16 | .0011 |
| 5/16 | 1/4 + 1/16 | .0101 |

**Como converter um floating de precisão simples para decimal:**

1- Se o MSB (bit mais significante) é 1, o número é negativo. Caso contrário, positivo.

2- Os seguintes 8 bits representam o expoente. Subtrai-se o binário 01111111 (decimal 127), produzindo o expoente não-polarizado. Converte-se então o expoente não-polarizado (*biased*) para decimal.

3- Os 23 bits seguintes representam o significando. Notar um “1.”, seguido dos bits da mantissa. Zeros à direita podem ser ignorados.

4- Em seguida, escala-se a mantissa produzida no passo 3, deslocando o ponto binário o número de vezes igual ao valor do expoente. Desloca-se à direita se o expoente é positivo, ou à esquerda se o expoente for negativo.

5- Converte-se a representação real binária produzida no passo 4 para representação decimal, e finalmente inclui-se o sinal.

**Ex:**  
*Converter 0* ***10000010 01011000000000000000000*** *para decimal*

1- O número é positivo.

2- O expoente não-polarizado (*biased*) é o binário 00000011, ou decimal 3.

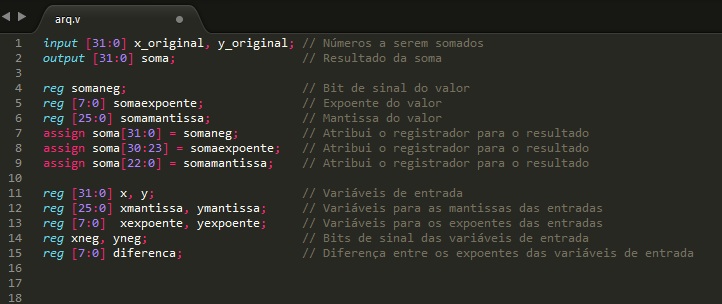
3- O significando é dado por 1. + 01011 = 1.01011.

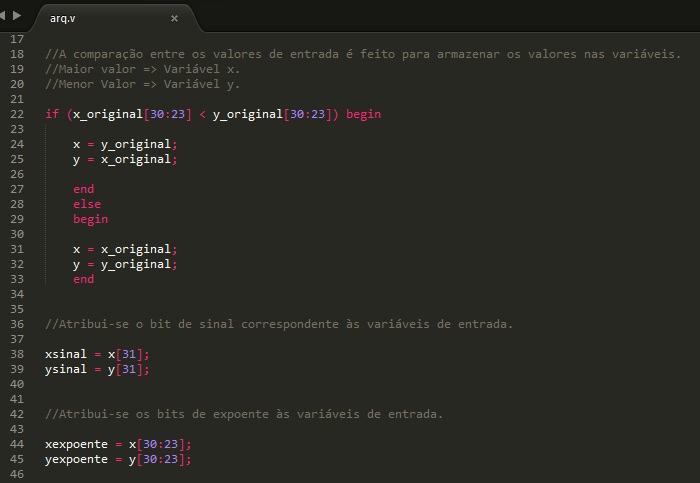
4- O número real binário é +1010.11.

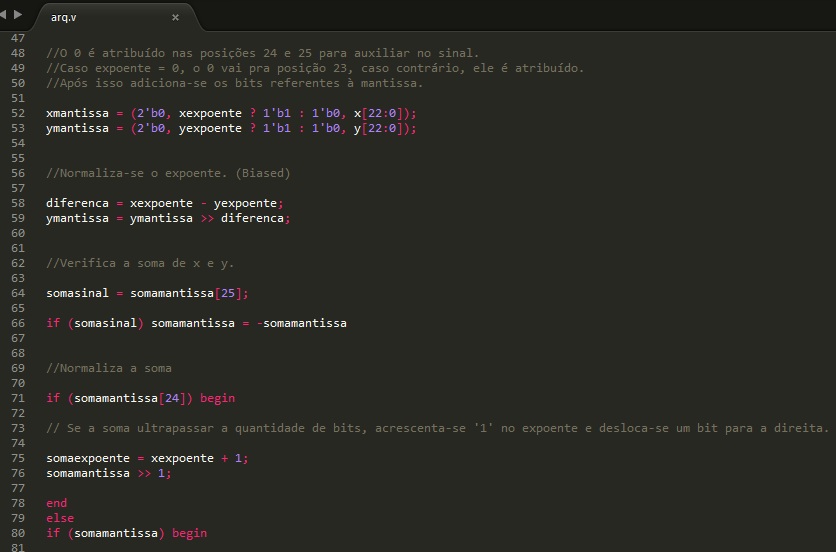
5- O valor decimal é **+10 3/4** ou **+10.75**.

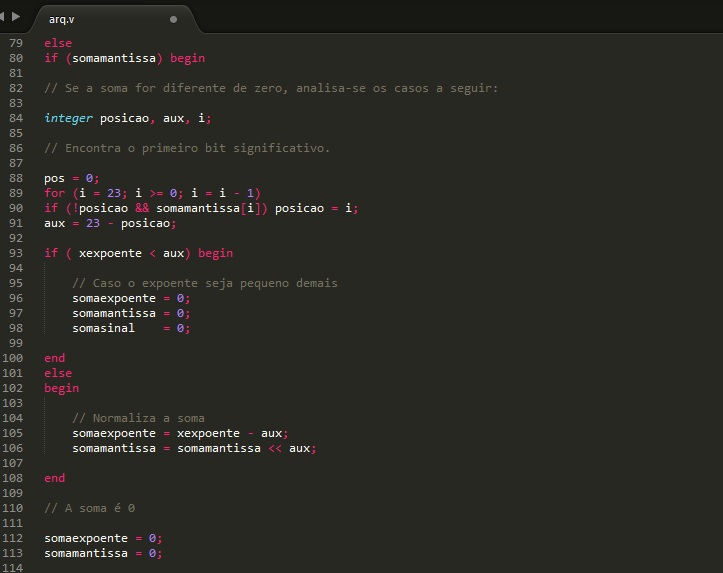
**Código**

O código do somador foi adaptado de outros códigos (vide Referências). Como o objetivo do relatório é mostrar o funcionamento do *FPU Adder* de 32 bits (precisão simples), modificações e alterações foram realizadas a fim de perfeito funcionamento.









**Referências**

BARRABÉS, Arturo. **Design Of Single Precision Float Adder (32-Bit Numbers) According To Ieee 754 Standard Using Vhdl.** Disponível em: <https://upcommons.upc.edu/bitstream/handle/2099.1/15467/32BitFloatingPointAdder.pdf>**.** Acesso em: 10/04/17.

KUMAR, Raj. **Efficient Floating Point 32-bit Single Precision Multipliers Design Using VHDL.** Disponível em: <http://www.ece.rochester.edu/~parihar/pres/Report_FP-Multipliers.pdf>**.** Acesso em: 10/04/17.

**Adding Floating Point Numbers.** Disponível em: https://www.cs.umd.edu/class/sum2003/cmsc311/Notes/BinMath/addFloat.html. Acesso em: 10/04/17.

IRVINE, Kip. **Assembly Language For Intel-Based Computers.** Disponível em: <http://profmsaeed.org/wp-content/uploads/2013/09/IntelAssemblyLanguage.pdf>. Acesso em: 10/04/17.