

Relatório EP 1

EACH-USP

ORGANIZAÇÃO DE COMPUTADORES DIGITAIS

AUTOR:

Gabriel faria de Oliveira 10403210

São Paulo

2020

Objetivo: Implementar uma calculadora que consiga fazer as quatro operações básicas com binários na forma de inteiro e floating point.

Implementação:

A calculadora foi implementada usando a IDE NetBeans para codificar (com um conjunto de 3 classes que contêm as funções) e montar a interface (a partir de um JFrame e componentes dentro dele). Ela faz as quatro operações básicas se utilizando de String's e arranjo dos mesmos para passar parâmetros e pegar o resultado, ou seja, para trabalhar com INT é passado um arranjo de tamanho igual a 2 [sendo, respectivamente, o bit de sinal (1 bit) e o INT (no máximo 31 bits)] ou 4 [sendo, respectivamente, o bit de sinal do FLOAT (1 bit), o do expoente (1 bit), o expoente (7 bits) e a mantissa (23 bits)]. E ela recebe as entradas com os binários sendo inseridos a partir de botões que colocam 0's e 1's nos campos imediatamente acima deles e as saídas são retiradas dos mesmos arranjos paramétricos e exibidas nos campos de resultado de cada modo de operação. Assim ela consegue passar parâmetros, fazer operações com eles e exibir os resultados para o usuário através dos campos de resultado.

Mais especificamente, temos que a calculadora faz suas operações de soma, subtração, multiplicação e divisão se utilizando, respectivamente, dos seguintes algoritmos: soma de binários, soma de binários com o sinal do segundo binário trocado, algoritmo de booth para multiplicar binários e algoritmo de booth para dividir binários. A soma de binários é feita comparando os char's contidos nas String's e, assim, fazendo a soma e a multiplicação é feita montando a grade de booth e somando A ou B e dando right shift "número de bits" vezes até chegar no resultado. Já a divisão é feita formando o AQ (arranjo de char contendo o acumulador e o dividendo) dando left shift subtraindo, divisor do A e restaurando ou não A "número de bits vezes" até chegar no resultado.

Essas operações também são usadas nas operações de FLOAT um vez que a classe desse tipo de operação tem uma relação de herança com a de operação de INT, que também tem uma relação de herança com a classe de operações básicas com binários. Ou seja, na soma de FLOAT's é usada a soma de INT's para manipular os FLOAT's, que já tiveram seus expoentes igualados se necessário. Já na multiplicação de FLOAT's são usadas a soma, para somar os expoentes, e a multiplicação, para multiplicar as mantissas, de INT's para manipular os FLOAT's, enquanto na divisão os expoentes são subtraídos e as mantissas são divididas. Todos os resultados dessas manipulações de FLOAT's são normalizados antes de serem retornados.

O modo das classes retornarem os resultados para a interface é, como já dito antes, através dos mesmos arranjos de String's de entrada. Ou seja, elas vão preencher o primeiro binário de entrada com o resultado normalizado e o segundo com ele em complemento de dois. Mas, em caso de erro de cálculo, elas vão preencher o segundo binário de entrada com o erro e o primeiro com o binário resultante, pois a classe de operações com INT deve passar o binário para a de FLOAT mesmo com erro de operação (por exemplo um overflow quando dois INT's são multiplicados).

Como compilar:

Para compilar o EP há duas possibilidades que são as seguintes: abrir o projeto usando NetBeans e compilar através dele, uma vez que a calculadora é um JFrame criado na IDE, ou somente usar as classes de operações binárias. Para abrir o EP no NetBeans é necessário ter a IDE instalada, ir em "File", clicar em "Open Project", ir até a pasta em que foi descompactado o zip e abrir a pasta calculadora, que é o projeto. Mas, caso queira executar a classe de operações binários isoladamente, é necessário ter as classes "operações.java", "operações_INT.java" e "operações_FLOAT.java" na pasta seguindo alguns passos. Eles são os seguintes: criar um objeto INT ou FLOAT, criar um arranjo de String's contendo, para o INT, nas primeiras colunas "1" ou "0" e nas segundas colunas os binários (que tem que ser do mesmo tamanho) ou, para o FLOAT, nas primeiras e segundas colunas "1" ou "0", nas terceiras os expoentes (tendo que ter 7 bits) e nas quartas as mantissas (tendo que ter 23 bits). Por fim, é necessário checar se deu algum erro de operação, que estarão na segunda linha (no INT estará na coluna segunda e no FLOAT estará na quarta) e podem ser os seguintes: "ERRO TODOS + (1)" (overflow positivo), "ERRO TODOS - (1)" (overflow negativo), "ERRO TODOS + (2)" (underflow positivo), "ERRO TODOS - (2)" (underflow negativo) e "ERRO DIVISAO ZERO" (divisão por zero).

Como executar:

The image displays two identical screenshots of a Java application window titled "CALCULADORA BINÁRIO". The window is divided into several sections for binary arithmetic operations.

Top Section: Includes a "NÚMERO DE BITS" field set to 2, with "MIN" and "MAX" buttons. To the right are four red buttons: "soma", "subtração", "multiplicação", and "divisão". A checkbox labeled "APAGAR CAMPOS AO SUBMETER" is located below these buttons. At the bottom of this section are two tabs: "int" (selected) and "float".

Input Section: Contains two rows of input fields. Each row has a "BIT DE SINAL" dropdown menu (set to 0), a text field for the binary number, and three buttons: "0", "1", and "limpar". The first row is labeled "PRIMEIRO BINÁRIO" and the second "SEGUNDO BINÁRIO".

Output Section: Labeled "RESULTADO", it contains two rows of output fields. Each row has a "NORMALIZADO" checkbox and a text field. Below these are two rows of "COMPLEMENTO DE 2:" checkboxes and text fields. At the bottom of the output section are two yellow buttons labeled "1" and "2".

Bottom Section: A legend indicating "1 - OVERFLOW | 2 - UNDERFLOW".

Para executar a calculadora é necessário abrir o arquivo "calculadora.jar", dando um duplo clique, e começar a usar ela seguindo os seguintes passos:

- ### Testes:

CALCULADORA BINÁRIO

NÚMERO DE BITS
 MIN MAX

soma subtração
multiplicação divisão

☐ APAGAR CAMPOS AO SUBMETER

int float

BIT DE SINAL PRIMEIRO BINÁRIO
 0 1 limpar

BIT DE SINAL SEGUNDO BINÁRIO
 0 1 limpar

RESULTADO

NORMALIZADO:

COMPLEMENTO DE 2:

SUBMITER

1 2

1- OVERFLOW | 2- UNDERFLOW

O resultado é 16439 (000000000000000000100000000110111).

- [illegible]

CALCULADORA BINÁRIO

— □ ×

NÚMERO DE BITS

32

▲ ▼

MIN

MAX

soma

subtração

multiplicação

divisão

☐ APAGAR CAMPOS AO SUBMETER

int

float

BIT DE SINAL

PRIMEIRO BINÁRIO

0

▼

1011011010011

0

1

limpar

BIT DE SINAL

SEGUNDO BINÁRIO

0

▼

10100101100100

0

1

limpar

RESULTADO

NORMALIZADO:

100000000000000000001001010010001

COMPLEMENTO DE 2:

111111111111111111111011010110111

1

2

1

2

SUBMETTER

1 - OVERFLOW | 2 - UNDERFLOW

O resultado é -4753 (100000000000000000001001010010001).

- [illegible]


CALCULADORA BINÁRIO
—
□
×

NÚMERO DE BITS

☐ APAGAR CAMPOS AO SUBMETER

BIT DE SINAL PRIMEIRO BINÁRIO

BIT DE SINAL SEGUNDO BINÁRIO

RESULTADO

NORMALIZADO:

COMPLEMENTO DE 2:

1

2

1 - OVERFLOW | 2 - UNDERFLOW

O resultado é 61912428 (00000011101100001011010101101100).

- Usando 32 bits para dividir 10596 (0000000000000000010100101100100) e 5843 (0000000000000000001011011010011):


CALCULADORA BINÁRIO
—
□
×

NÚMERO DE BITS

32

↑
↓

MIN
MAX

soma

subtração

multiplicação

divisão

☐ APAGAR CAMPOS AO SUBMETER

int

float

BIT DE SINAL
PRIMEIRO BINÁRIO

0
▼

10100101100100

0

1

limpar

BIT DE SINAL
SEGUNDO BINÁRIO

0
▼

1011011010011

0

1

limpar

RESULTADO

NORMALIZADO:

00000000000000000000000000000001

COMPLEMENTO DE 2:

11111111111111111111111111111111

SUBMETER

1

2

1 - OVERFLOW | 2 - UNDERFLOW

O resultado é 1 (00000000000000000000000000000001).

- Usando 5 bits para somar 15 (01111) com 1 (00001):

CALCULADORA BINÁRIO

NÚMERO DE BITS

5

▲

▼

MIN

MAX

soma

subtração

multiplicação

divisão

☐ APAGAR CAMPOS AO SUBMETER

int

float

BIT DE SINAL

0

▼

PRIMEIRO BINÁRIO

1111

0

1

limpar

BIT DE SINAL

0

▼

SEGUNDO BINÁRIO

1

0

1

limpar

RESULTADO

NORMALIZADO:

+ infinito

COMPLEMENTO DE 2:

- infinito

SUBMETER

1

2

1 - OVERFLOW | 2 - UNDERFLOW

NÚMERO DE BITS

5

MIN

MAX

soma

subtração

multiplicação

divisão

☐ APAGAR CAMPOS AO SUBMETER

int

float

BIT DE SINAL

PRIMEIRO BINÁRIO

0

▼ 1111

0

1

limpar

BIT DE SINAL

SEGUNDO BINÁRIO

0

▼ 1

0

1

limpar

RESULTADO

NORMALIZADO:

+ infinito

COMPLEMENTO DE 2:

- infinito

SUBMETER

1

2

1 - OVERFLOW | 2- UNDERFLOW

- Usando 5 bits para somar -15 (11111) com -1 (10001):

CALCULADORA BINÁRIO

NÚMERO DE BITS

5

▲

▼

MIN

MAX

soma

subtração

multiplicação

divisão

☐ APAGAR CAMPOS AO SUBMETER

int

float

BIT DE SINAL

PRIMEIRO BINÁRIO

1

▼

1111

0

1

limpar

BIT DE SINAL

SEGUNDO BINÁRIO

1

▼

1

0

1

limpar

RESULTADO

NORMALIZADO:

- infinito

COMPLEMENTO DE 2:

+ infinito

SUBMETER

1

2

1 - OVERFLOW | 2 - UNDERFLOW

O resultado é -infinito (overflow).

- Usando 5 bits para multiplicar 8 (01000) com 2 (00010):

O resultado é + infinito (overflow).

Esses teste são referentes as operações de INT e foram feitos com muitos outro número que não davam overflow e outro de davam. O mesmo vale para as operação de FLOAT, ou seja, teste foram realizados visando números que estouravam o capacidade de 8 bits do expoente, o que quer dizer o seguinte: a soma/subtração deles estourou os bits ou, devido a uma mantissa grande, não foi possível dar mais dar right shift ou, devido a uma mantissa pequena, não foi possível mais dar left shift. Algumas tentativas de conseguir atingir o underflow na calculadora foram realizados, mas nenhuma conseguiu.

Referências:

Converter número Decimal para Binário, Octal e Hexadecimal. Clevert. Disponível em: <<https://clevert.com.br/t/pt-br/base-convert>>. Acesso em: 29 de abr. de 2020.

Booth's Algorithm for Signed Multiplication. Youtube, 2018. Disponível em: <<https://www.youtube.com/watch?v=QFXaddi-Ag8>>. Acesso em: 16 de abr. de 2020.

Floating Point Arithmetic on Addition and Subtraction. Youtube, 2018. Disponível em: <<https://www.youtube.com/watch?v=w7NQtb1FTDU>>. Acesso em: 18 de abr. de 2020.

Floating Point Number Representation. Youtube, 2018. Disponível em: <
<https://www.youtube.com/watch?v=XOMTNy2qiZ0>>. Acesso em: 15 de abr. de 2020.

Floating Point Arithmetic on Multiplication. Youtube, 2018. Disponível em: <
<https://www.youtube.com/watch?v=0HiGruw9VcQ>>. Acesso em: 20 de abr. de 2020.

Floating Point Arithmetic on Division. Youtube, 2018. Disponível em: <
<https://www.youtube.com/watch?v=B3Sggj1HmR4>>. Acesso em: 20 de abr. de 2020.

Restoring Division Algorithm for Unsigned Integer. Youtube, 2018. Disponível em: <
<https://www.youtube.com/watch?v=PzV6gYpVLuc>>. Acesso em: 17 de abr. de 2020.

Booth's multiplication algorithm. Wikipedia, 2018. Disponível em: <
https://en.wikipedia.org/wiki/Booth%27s_multiplication_algorithm>. Acesso em: 15
de abr. de 2020.