Relatório EP 1

EACH-USP ORGANIZAÇÃO DE COMPUTADORES DIGITAIS

AUTOR:

Gabriel faria de Oliveira 10403210

São Paulo

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 a quatro operações básicas se utilizando de String's e arranjo dos mesmas 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 mesmo 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.

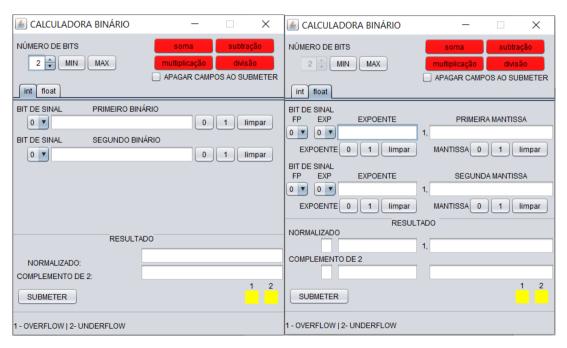
Essas operações também são usadas nas operações de FLOAT um vez que a classe desse tipo de operação tem uma relação de herança com a de operação de INT, que também tem uma relação de herança com a classe de operações básicas com binários. Ou seja, na soma de FLOAT's é usada a soma de INT's para manipular os FLOAT's, que já tiveram seus expoente igualadas 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 classe 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 calcula, 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 tem 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:



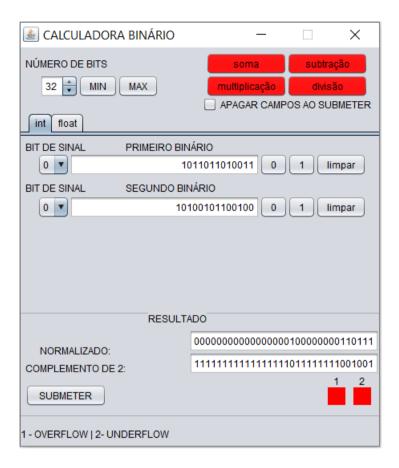
Para executar a calculadora é necessário abrir o arquivo "calculadora.jar", dando um duplo clique, e começar a usar ela seguindo os seguintes passos:

- Passo 1: Escolher qual operação deve ser feita clicando em uns dos quatro botões de operações possíveis. Caso nenhuma seja escolhida, uma mensagem de erro vai aparecer dizendo que nenhuma operação foi escolhida.
- Passo 2: Escolher o modo de operação (INT ou FLOAT) para preencher os binários, que podem ser alterados clicando nos botões (combobox's para alterar o bit de sinal, "0" para adicionar 0's, "1" para adicionar 1's e "limpar" para limpar o binário).
 - Modo 1 (INT): No primeiro modo há duas combobox's para selecionar o bit de sinal de cada binário, dois textfield's para preencher os binários (com um digito a menos devido ao bit de sinal) e a possibilidade de alterar a quantidade de bits que serão usados clicando no spinner acima das tabs de operação.
 - Modo 2 (FLOAT): No segundo modo há quatro combobox's para selecionar o bit de sinal de cada binário (duas para o binário em floating point e duas para os expoentes), quatro textfield's para preencher os binários (dois para as mantissas e duas para os expoentes).
- Passo 3: Clicar no botão 'submeter" para fazer a operação desejada com os binários preenchidos nos devidos campos.
- Passo 4: Observar a resposta que vai aparecer nos textfield's do modo de operação escolhido e voltar ao passo um para usar a calculadora novamente.
 - Modo 1: No primeiro modo vão ser preenchidos os dois textfield's abaixo da linha "RESULTADO", com o de cima contendo o binário e o de baixo contendo o seu complemento de dois.
 - Modo 2: No segundo modo vão ser preenchidos os seis textfield's abaixo da linha "RESULTADO", com os três sendo, respectivamente, o bit de sinal, o expoente e a mantissa.

Testes:

Os testes foram realizados usando o site do Clevert (disponível nas referências) fazendo os cálculos na calculadora do EP, convertendo as entradas e a saída para decimal e testando para ver se o resultado era o mesmo quando feita em decimal. Já para os erros de cálculo (como o overflow) foram realizados cálculos que claramente iriam dar erro (como multiplicar dois número grandes que iriam estourar os bits). Alguns exemplos de testes são os seguintes:

• Usando 32 bits para somar 5843 (000000000000000001011011010011) e 10596 (00000000000000000101001001001):



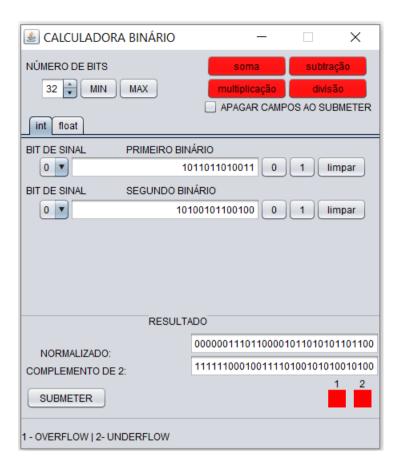
O resultado é 16439 (00000000000000010000000110111).

• Usando 32 bits para subtrair 5843 (000000000000000001011011010011) e 10596 (00000000000000000101001001001):



O resultado é -4753 (1000000000000000100101001001).

Usando 32 bits para multiplicar 5843 (0000000000000000001011011010011)
 e 10596 (000000000000000010100101100100):



O resultado é 61912428 (00000011101100001011010101101100).

Usando 32 bits para dividir 10596 (000000000000000010100101100100) e
 5843 (0000000000000000001011011010011):

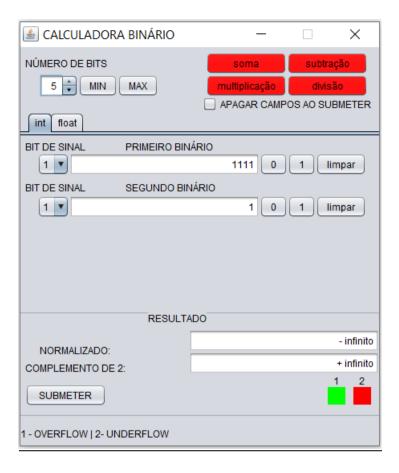


• Usando 5 bits para somar 15 (01111) com 1 (00001):



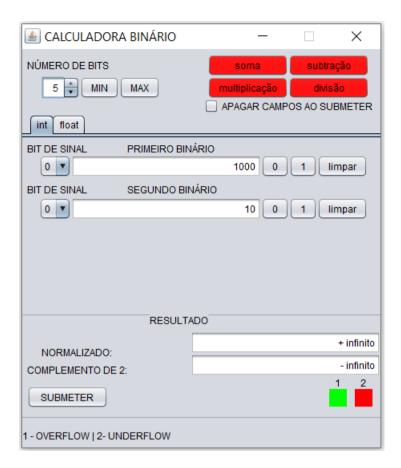
O resultado é +infinito (overflow).

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



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.