

UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES

GABRIEL ESTAVARINGO FERREIRA
YAN HIDEKI KAWAKAMI

RELATÓRIO DE ENTREGA DO EP

SÃO PAULO
2018

GABRIEL ESTAVARINGO FERREIRA
YAN HIDEKI KAWAKAMI

RELATÓRIO DE ENTREGA DE EP
CALCULADORA DE BINÁRIOS

Relatório de entrega do EP, contendo detalhes de utilização, como foi feito, testes realizados e comentários adicionais.

professor: João Bernardes

SÃO PAULO

Sumário

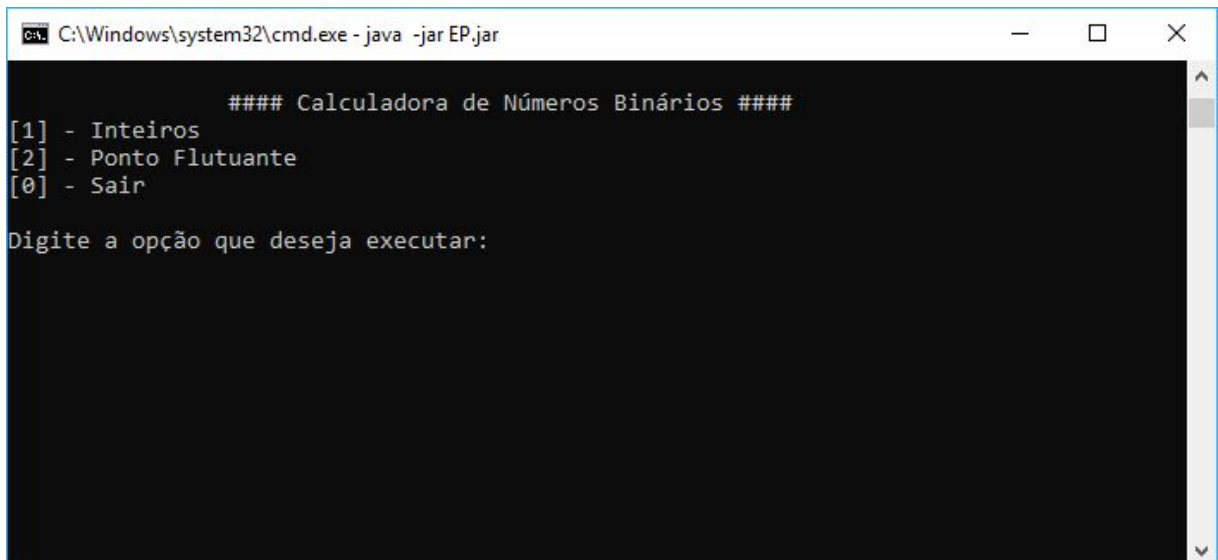
Como Utilizar	3
Código e E.D.	5
Estrutura de Dados	5
Códigos	5
Testes	9
Referências	18

Como Utilizar

Para iniciar o programa, execute o arquivo “EP.bat” ou rode o arquivo EP.jar através do prompt de comando, executando a seguinte linha: `java -jar EP_OCD.jar`. (SÓ WINDOWS)

Ao iniciar o programa, o menu principal será exibido, dando a opção do usuário decidir o tipo de cálculo que irá executar.

As opções são: cálculo com inteiro ou cálculo com ponto flutuante.



```
C:\Windows\system32\cmd.exe - java -jar EP.jar

#### Calculadora de Números Binários ####
[1] - Inteiros
[2] - Ponto Flutuante
[0] - Sair
Digite a opção que deseja executar:
```

Para selecionar a opção desejada, digite o número correspondente e pressione enter.

Após selecionar a opção, será solicitado que você informe o primeiro número da operação, qual operação você deseja efetuar e o segundo número.

Para inteiros, primeiramente você deve inserir a quantidade de bits que deseja. Essa quantidade será utilizada para armazenar os números em binário e efetuar as operações.

Caso a quantidade de bits for insuficiente para armazenar os números, o programa irá retornar um erro. Os números devem ser inseridos na base 10.

Para ponto flutuante, os números devem ser inseridos no formato: sinal + mantissa normalizada + expoente sem excesso (ex: 0 1,001010000000000000000000 00000011)

Caso o expoente inserido for negativo, o mesmo deverá estar em complemento de 2.

```
C:\Windows\system32\cmd.exe - java -jar EP.jar

Exemplo de entrada: sinal + mantissa + expoente sem excesso (Ex: 0 1,001010000000000000000000 00000010)
Digite o primeiro número em ponto flutuante padrão IEE754:
0 1,001010000000000000000000 00000010

[1] - Soma
[2] - Subtração
[3] - Divisão
[4] - Multiplicação

Digite a operação:
1

Digite o segundo número em ponto flutuante padrão IEE754:
0 1,011000000000000000000000 00000001
Operação: Soma
Número 1:
    Decimal: 4.625 Binário: 01000000100101000000000000000000
Número 2:
    Decimal: 5.5   Binário: 01000000101100000000000000000000
Resultado:
    Decimal: 10.125   Binário: 01000001001000100000000000000000

Entre com qualquer valor para retornar ao menu
```

O resultado é exibido em decimal e binário, conforme demonstra a imagem acima.

Para ponto flutuante, o número em formato binário é exibido exatamente da maneira como ele é armazenado na memória e, para facilitar a interpretação, é exibido o seu equivalente em base 10

Após finalizar a operação, digite qualquer valor e pressione enter para retornar ao menu.

Código e E.D.

Estrutura de Dados

Para a realização das operações com números binários foi utilizado a classe “ArrayList” do java. Um ArrayList é basicamente um vetor comum, sendo que o diferencial (para essa aplicação) é que o próprio java já faz o gerenciamento do tamanho do vetor conforme você insere ou remove valores. Sendo assim, as operações que incluem números com quantidades de bits diferentes é simplificada e é eliminada a necessidade da declaração prévia da quantidade de bits que será utilizada. Porém, como requisito do professor, ao realizar a conta com números inteiros, é solicitado ao usuário para que informe a quantidade de bits que os números devem possuir.

Todos os números binários devem possuir sinal e os números negativos devem ser armazenados em complemento de 2.

Utilizei um vetor comum(int[]) apenas para armazenar o número em ponto flutuante no padrão IEEE 754, sendo que para realizar as operações, separei-o em dois ArrayList's, um contendo a mantíssa e o outro contendo o expoente.

Códigos

Para as operações com inteiros, os números são recebidos em base decimal, convertidos para binário, efetuadas as operações e o resultado é então convertido para base decimal. Para as operações com ponto flutuante, o número é recebido em binário no padrão IEEE 754, feitas as operações, os operadores e o resultado são convertidos para base decimal. Para realizar todas as operações, foram criadas os seguintes métodos:

ArrayList<Integer> converter(int dec, int qtdBits)

Recebe um número inteiro em base decimal e retorna o seu valor em base binária, armazenado em um ArrayList, com a quantidade de bits definida pelo parâmetro qtdBits. Se a quantidade de bits for igual a 0, devolve o número com a quantidade mínima necessária

int converter(ArrayList<Integer> bin)

Recebe um número binário armazenado em um arraylist e devolve o seu valor na base decimal em uma variável do tipo inteiro.

ArrayList<Integer> somar(ArrayList<Integer> bin1, ArrayList<Integer> bin2)

Recebe dois números binários armazenados em ArrayList's e devolve a soma dos dois em um outro ArrayList. Os dois números precisam ter a mesma quantidade de bits

ArrayList<Integer> complemento2(ArrayList<Integer> bin)

Recebe um número binário em um ArrayList e devolve o seu complemento de 2 armazenado em outro ArrayList.

ArrayList<Integer> subtrair(ArrayList<Integer> bin1, ArrayList<Integer> bin2)

Recebe dois números binários armazenados em ArrayList's e devolve o resultado da subtração dos dois em um outro ArrayList. A subtração é efetuada utilizando complemento de 2.

ArrayList<Integer> multiplicarBooth(ArrayList<Integer> bin1, ArrayList<Integer> bin2)

Recebe dois números binários armazenados em ArrayList's e devolve o resultado da multiplicação dos dois em um outro ArrayList. A Multiplicação é feita através do método de Booth.

ArrayList<Integer> dividir(ArrayList<Integer> bin1, ArrayList<Integer> bin2, boolean mantissa)

Recebe dois números binários armazenados em ArrayList's e devolve o resultado da divisão dos dois em um outro ArrayList. A divisão é obtida efetuando diversas subtrações consecutivas e retornando a quantidade de subtrações efetuadas. Além disso, esse método recebe um boolean como parâmetro, indicando se é uma divisão de mantissa ou não. Quando uma divisão de mantissa é efetuado, o resto da divisão é considerado e é realizadas diversas divisões consecutivas até se obter a quantidade de bits suficientes para guardar na mantissa do resultado.

float converter(int[] pontoFlutuante)

Recebe um número em ponto flutuante armazenado em um vetor de inteiros e devolve o seu correspondente em base decimal, armazenado em uma variável do tipo float.

int[] somarSubtrair(int[] pontoFlutuante, int[] pontoFlutuante2, int op)

Recebe dois números em ponto flutuante armazenados em vetores de inteiros e devolve a soma ou subtração dos dois em um outro vetor de inteiros. Se op = 0, retorna a soma. Se op = 1, retorna a subtração. Esse método faz a checagem de overflow e underflow do expoente. O overflow acontece quando o expoente é maior que 128 e o underflow quando o expoente é menor que -127.

int compara(ArrayList<Integer> bin1, ArrayList<Integer> bin2)

Compara dois números binários armazenados em ArrayList's. Utilizado para comparar os expoentes dos números em ponto flutuante. Retorna 0 se forem iguais, -1 se o primeiro for menor e 1 se o primeiro for maior.

void rightShift(ArrayList<Integer> mantissa)

Recebe um número binário armazenado em um ArrayList e faz o right shift. Utilizado para deslocar as casas da mantissa.

void leftShift(ArrayList<Integer> mantissa)

Recebe um número binário armazenado em um ArrayList e faz o left shift. Utilizado para deslocar as casas da mantissa.

int[] multiplicar(int[] pontoFlutuante, int[] pontoFlutuante2)

Recebe dois números em ponto flutuante armazenados em vetores de inteiros e devolve a multiplicação dos dois em um outro vetor de inteiros. Esse método faz a checagem de overflow do expoente. O overflow acontece quando o expoente é maior que 128.

int[] dividir(int[] pontoFlutuante, int[] pontoFlutuante2)

Recebe dois números em ponto flutuante armazenados em vetores de inteiros e devolve a divisão dos dois em um outro vetor de inteiros. Esse método faz a checagem de underflow do expoente. O underflow acontece quando o expoente é menor que -127.

Para mais detalhes de implementação, consulte o código fonte, pois o mesmo está todo comentado, com detalhes de cada passo que é executado em cada operação.

Testes

Inteiros

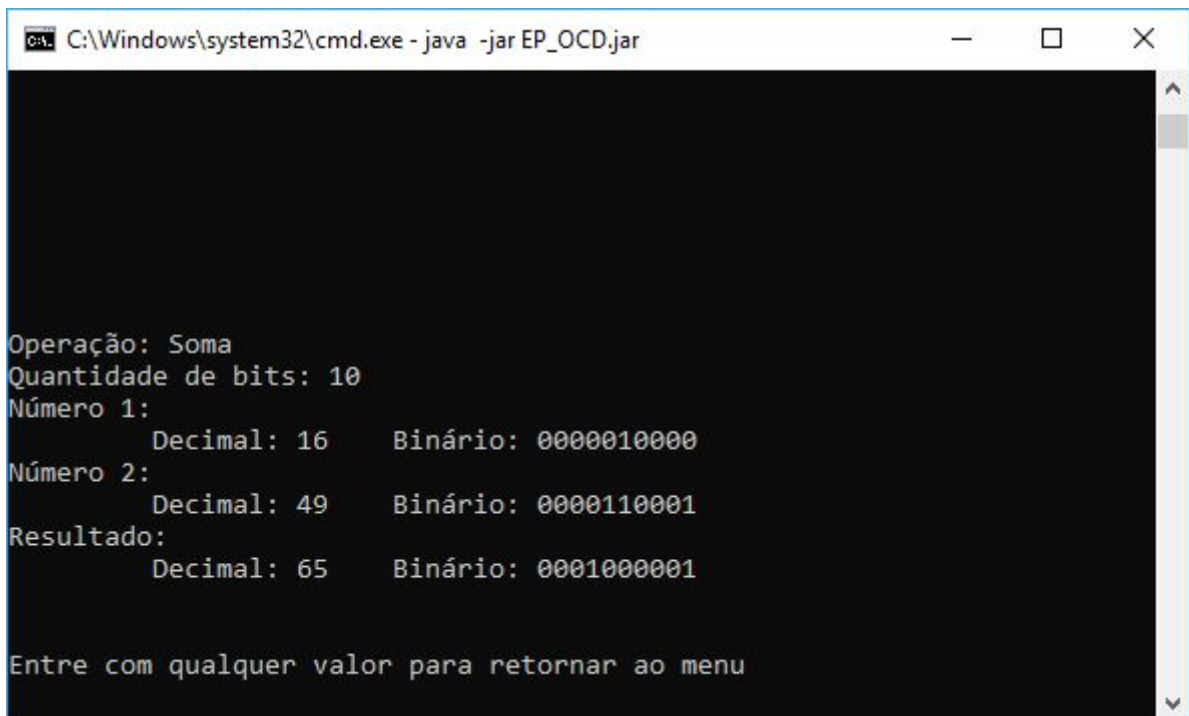
Soma

16 + 49

Resultado Esperado: 65

Resultado Obtido: 65

bits:10



```
C:\Windows\system32\cmd.exe - java -jar EP_OCD.jar

Operação: Soma
Quantidade de bits: 10
Número 1:
    Decimal: 16    Binário: 0000010000
Número 2:
    Decimal: 49    Binário: 0000110001
Resultado:
    Decimal: 65    Binário: 0001000001

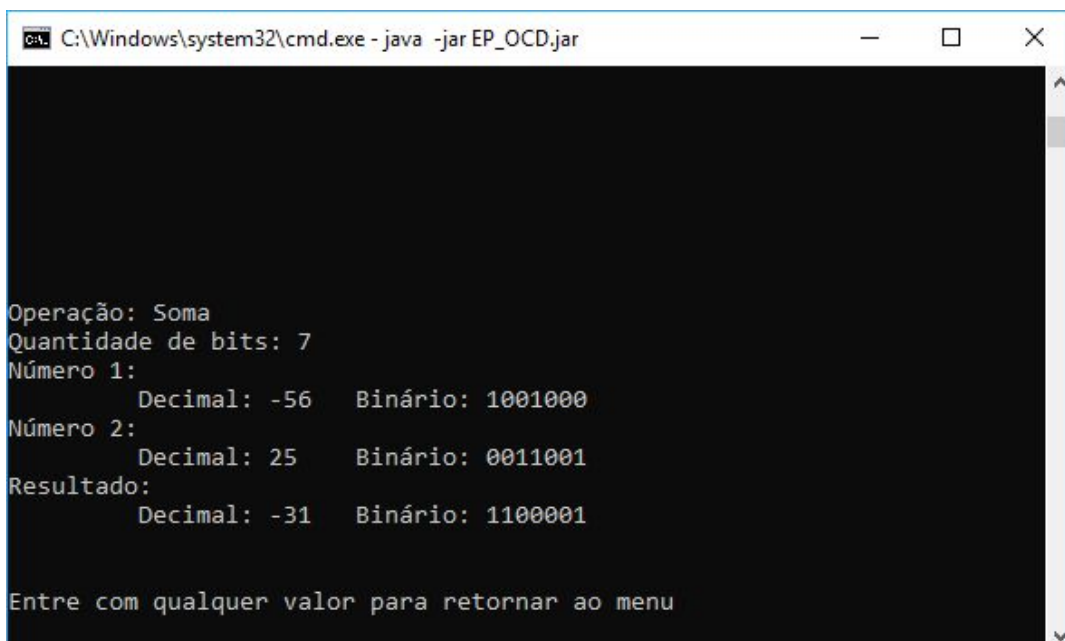
Entre com qualquer valor para retornar ao menu
```

-56 + 25

Resultado Esperado:-31

Resultado Obtido: -31

bits:7



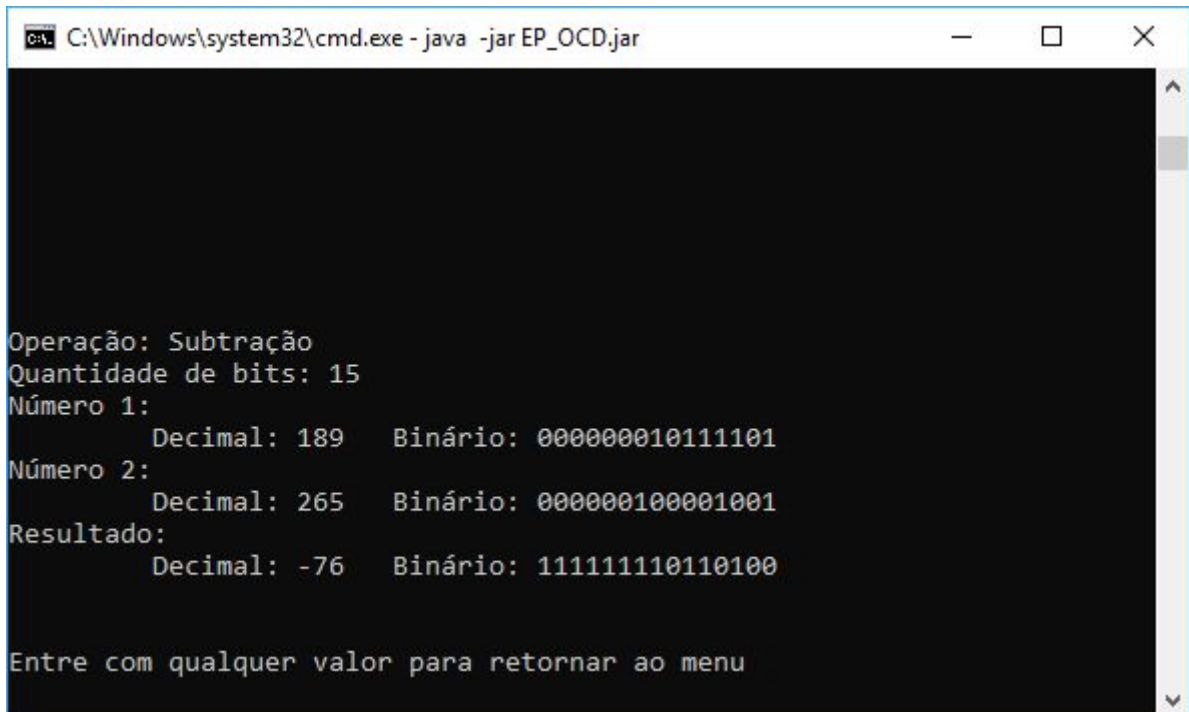
```
C:\Windows\system32\cmd.exe - java -jar EP_OCD.jar

Operação: Soma
Quantidade de bits: 7
Número 1:
    Decimal: -56   Binário: 1001000
Número 2:
    Decimal: 25    Binário: 0011001
Resultado:
    Decimal: -31   Binário: 1100001

Entre com qualquer valor para retornar ao menu
```

Subtração

189 - 265 Resultado Esperado:-76 Resultado Obtido: -76 bits:15

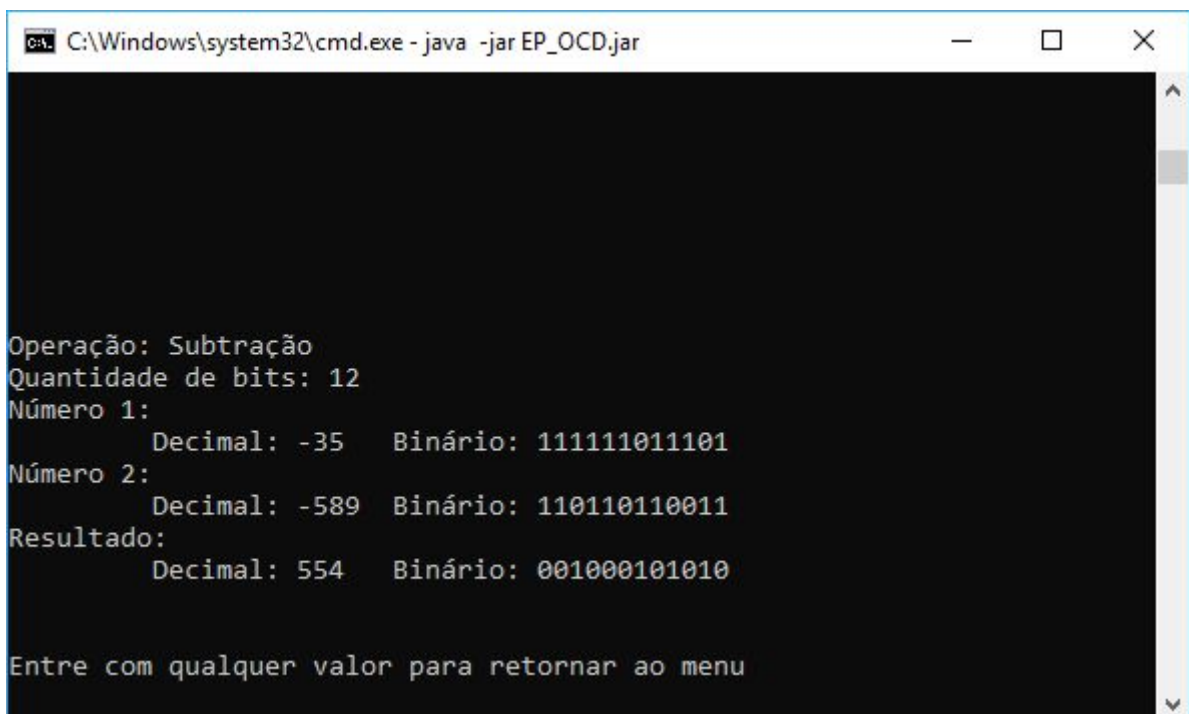


```
C:\Windows\system32\cmd.exe - java -jar EP_OCD.jar

Operação: Subtração
Quantidade de bits: 15
Número 1:
    Decimal: 189    Binário: 000000010111101
Número 2:
    Decimal: 265    Binário: 000000100001001
Resultado:
    Decimal: -76    Binário: 11111110110100

Entre com qualquer valor para retornar ao menu
```

-35 - (-589) Resultado Esperado:554 Resultado Obtido:554 bits:12



```
C:\Windows\system32\cmd.exe - java -jar EP_OCD.jar

Operação: Subtração
Quantidade de bits: 12
Número 1:
    Decimal: -35    Binário: 111111011101
Número 2:
    Decimal: -589   Binário: 110110110011
Resultado:
    Decimal: 554    Binário: 001000101010

Entre com qualquer valor para retornar ao menu
```

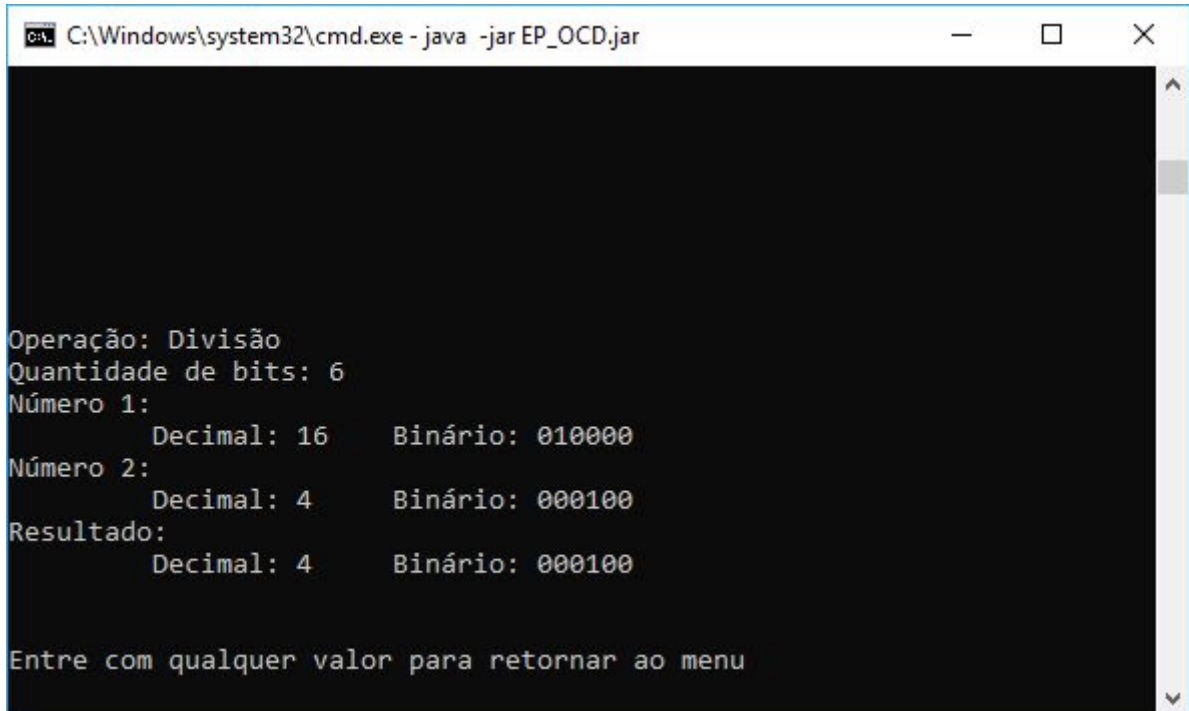
Divisão

16/4

Resultado Esperado: 4

Resultado Obtido: 4

bits:6



```
C:\Windows\system32\cmd.exe - java -jar EP_OCD.jar

Operação: Divisão
Quantidade de bits: 6
Número 1:
    Decimal: 16    Binário: 010000
Número 2:
    Decimal: 4     Binário: 000100
Resultado:
    Decimal: 4     Binário: 000100

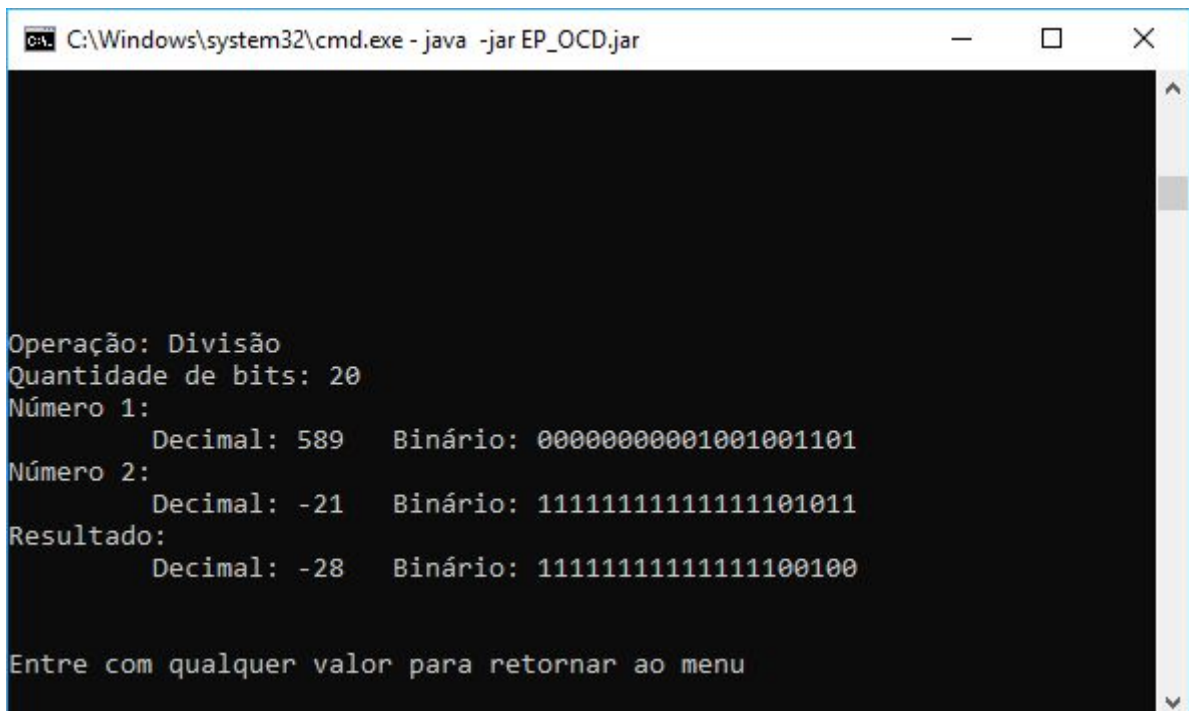
Entre com qualquer valor para retornar ao menu
```

589 / (-21)

Resultado Esperado: -28

Resultado Obtido: -28

bits:20



```
C:\Windows\system32\cmd.exe - java -jar EP_OCD.jar

Operação: Divisão
Quantidade de bits: 20
Número 1:
    Decimal: 589   Binário: 00000000001001001101
Número 2:
    Decimal: -21   Binário: 1111111111111101011
Resultado:
    Decimal: -28   Binário: 1111111111111100100

Entre com qualquer valor para retornar ao menu
```

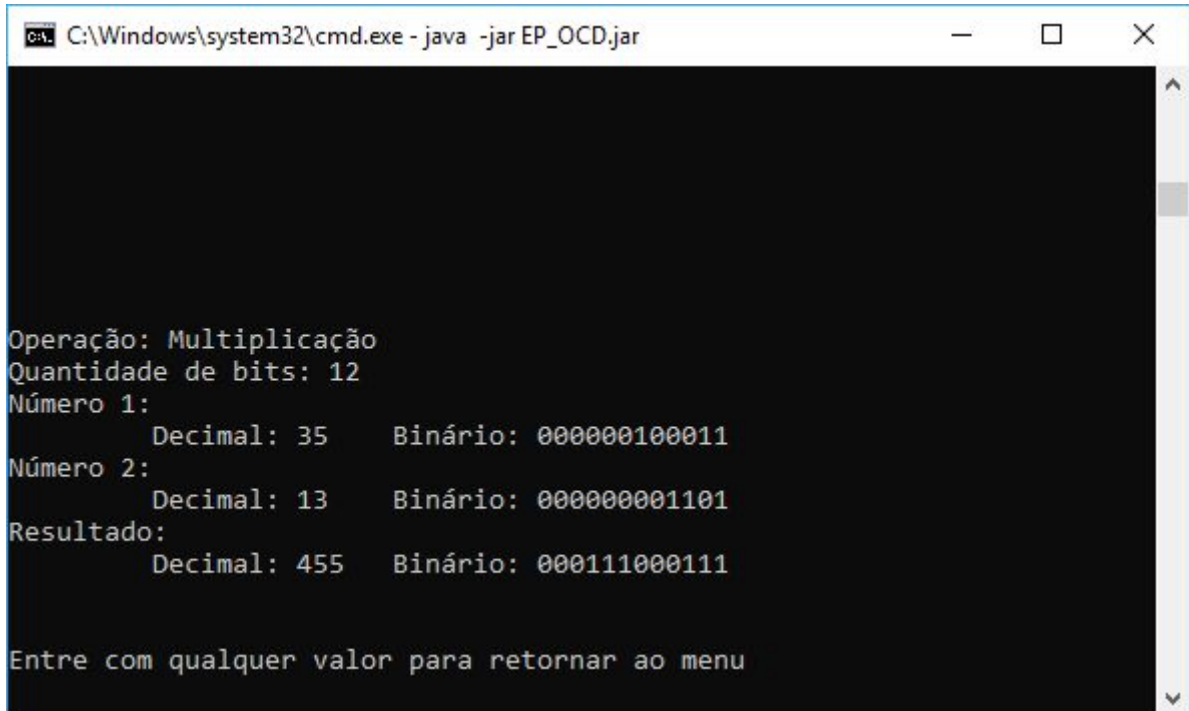
Multiplicação

35 * 13

Resultado Esperado: 455

Resultado Obtido: 455

bits:12



```
C:\Windows\system32\cmd.exe - java -jar EP_OCD.jar

Operação: Multiplicação
Quantidade de bits: 12
Número 1:
    Decimal: 35    Binário: 000000100011
Número 2:
    Decimal: 13    Binário: 000000001101
Resultado:
    Decimal: 455   Binário: 000111000111

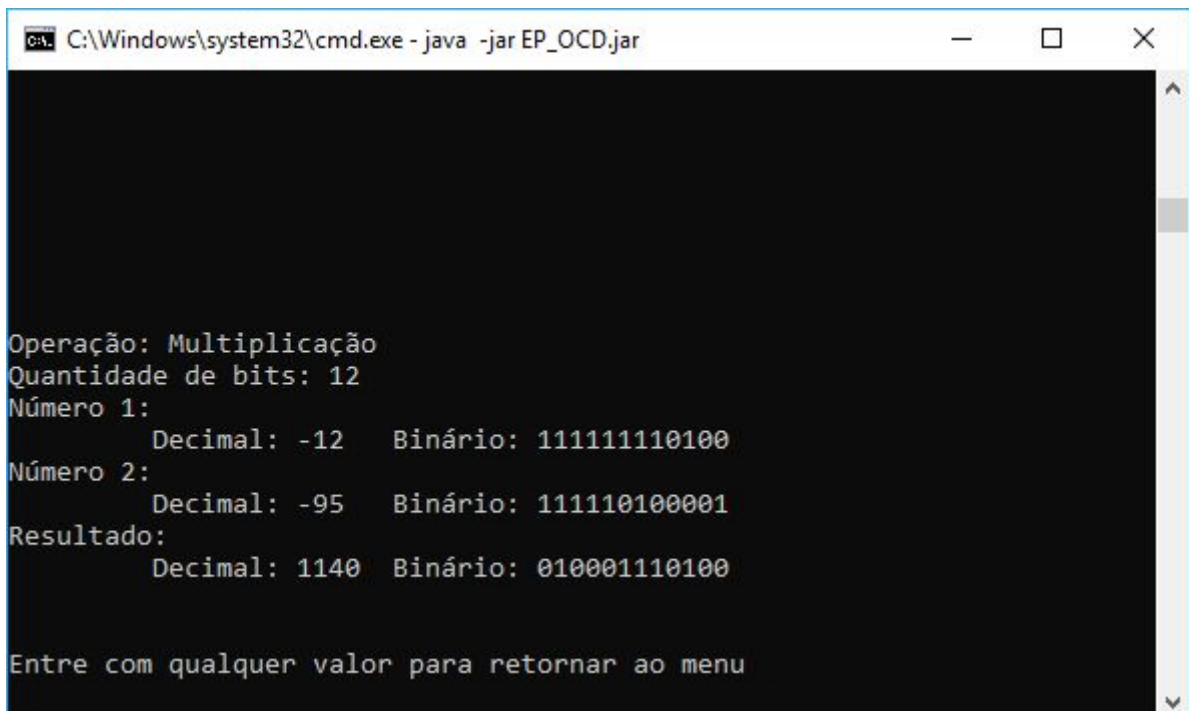
Entre com qualquer valor para retornar ao menu
```

(-12) * (-95)

Resultado Esperado: 1140

Resultado Obtido: 1140

bits:12



```
C:\Windows\system32\cmd.exe - java -jar EP_OCD.jar

Operação: Multiplicação
Quantidade de bits: 12
Número 1:
    Decimal: -12   Binário: 111111110100
Número 2:
    Decimal: -95   Binário: 111110100001
Resultado:
    Decimal: 1140  Binário: 010001110100

Entre com qualquer valor para retornar ao menu
```

Ponto Flutuante

Soma

4,625 + 2,75

Resultado Esperado: 7,375 Resultado Obtido: 7,375

```
C:\Windows\system32\cmd.exe - java -jar EP_OCD.jar
[4] - Multiplicação
Digite a operação:
1
Digite o segundo número em ponto flutuante padrão IEE754:
0 1,011000000000000000000000 00000001

Operação: Soma
Número 1:
    Decimal: 4.625 Binário: 01000000100101000000000000000000
Número 2:
    Decimal: 2.75  Binário: 01000000011000000000000000000000
Resultado:
    Decimal: 7.375 Binário: 01000000111011000000000000000000

Entre com qualquer valor para retornar ao menu
```

10,5 + (-8,75)

Resultado Esperado: 1,75 Resultado Obtido: 1,75

```
C:\Windows\system32\cmd.exe - java -jar EP_OCD.jar
[4] - Multiplicação
Digite a operação:
1
Digite o segundo número em ponto flutuante padrão IEE754:
1 1,000110000000000000000000 00000011

Operação: Soma
Número 1:
    Decimal: 10.5  Binário: 01000001001010000000000000000000
Número 2:
    Decimal: -8.75 Binário: 11000001000011000000000000000000
Resultado:
    Decimal: 1.75  Binário: 00111111111000000000000000000000

Entre com qualquer valor para retornar ao menu
```


Subtração

23,625 - 3,5

Resultado Esperado: 20,125 Resultado Obtido: 20,125

```
C:\Windows\system32\cmd.exe - java -jar EP_OCD.jar
[4] - Multiplicação
Digite a operação:
2
Digite o segundo número em ponto flutuante padrão IEE754:
0 1,11000000000000000000000000000000 00000001

Operação: Subtração
Número 1:
    Decimal: 23.625      Binário: 01000001101111010000000000000000
Número 2:
    Decimal: 3.5        Binário: 01000000011000000000000000000000
Resultado:
    Decimal: 20.125     Binário: 01000001101000010000000000000000

Entre com qualquer valor para retornar ao menu
```

1.548.288 - (-48.408) Resultado Esperado:1.596.696 Resultado Obtido: 1.596.696

```
C:\Windows\system32\cmd.exe - java -jar "EP_OCD.jar"
[4] - Multiplicação
Digite a operação:
2
Digite o segundo número em ponto flutuante padrão IEE754:
1 1,011110100011000000000000 00001111

Operação: Subtração
Número 1:
    Decimal: 1548288.0   Binário: 01001001101111010000000000000000
Número 2:
    Decimal: -48408.0    Binário: 11000111001111010001100000000000
Resultado:
    Decimal: 1596696.0   Binário: 01001001110000101110100011000000

Entre com qualquer valor para retornar ao menu
```

Divisão

24,75 / 4,5

Resultado Esperado: 5,5

Resultado Obtido: 5,5

```
C:\Windows\system32\cmd.exe - java -jar "EP_OCD.jar"

[4] - Multiplicação

Digite a operação:
3

Digite o segundo número em ponto flutuante padrão IEE754:
0 1,00100000000000000000000000000000 00000010

Operação: Divisão
Número 1:
    Decimal: 24.75 Binário: 01000001110001100000000000000000
Número 2:
    Decimal: 4.5   Binário: 01000001001000000000000000000000
Resultado:
    Decimal: 5.5   Binário: 01000001011000000000000000000000

Entre com qualquer valor para retornar ao menu
```

6,626x10⁻³⁴ / 10,5

Resultado Esperado: 6,31047619x10⁻³⁵

Resultado Obtido: 6,310531 x 10⁻³⁵

```
C:\Windows\system32\cmd.exe - java -jar "EP_OCD.jar"

[4] - Multiplicação

Digite a operação:
3

Digite o segundo número em ponto flutuante padrão IEE754:
0 1,01010000000000000000000000000000 00000011

Operação: Divisão
Número 1:
    Decimal: 6.626068E-34 Binário: 00001000010111000011000001011010
Número 2:
    Decimal: 10.5   Binário: 01000001001010000000000000000000
Resultado:
    Decimal: 6.310531E-35 Binário: 00000110101001111100001101000000

Entre com qualquer valor para retornar ao menu
```


6,626068x10⁻³⁴ / 1,8377686x10¹⁹

Resultado Esperado: Underflow no Expoente

Resultado Obtido: Underflow no Expoente

```
C:\Windows\system32\cmd.exe - java -jar "EP_OCD.jar"

3

Digite o segundo número em ponto flutuante padrão IEE754:
0 1,111111100001010101000 00111111

Underflow no expoente!!!

Operação: Divisão
Número 1:
    Decimal: 6.626068E-34  Binário: 00001000010111000011000001011010
Número 2:
    Decimal: 1.8377686E19  Binário: 01011111011111110000101010101000
Resultado:
    Decimal: 0.0    Binário: 00000000000000000000000000000000

Entre com qualquer valor para retornar ao menu
```

7,25 / 4,5

Resultado Esperado: 1,611111...

Resultado Obtido: 1,6111107

```
C:\Windows\system32\cmd.exe - java -jar EP_OCD.jar

[4] - Multiplicação

Digite a operação:
3

Digite o segundo número em ponto flutuante padrão IEE754:
0 1,001000000000000000000000 00000010

Operação: Divisão
Número 1:
    Decimal: 7.25  Binário: 01000000111010000000000000000000
Número 2:
    Decimal: 4.5   Binário: 01000000100100000000000000000000
Resultado:
    Decimal: 1.6111107  Binário: 00111111110011100011100011100000

Entre com qualquer valor para retornar ao menu
```

Multiplicação

2,75 * (-23,625)

Resultado Esperado: -64,96875

Resultado Obtido: -64,96875

```
C:\Windows\system32\cmd.exe - java -jar "EP_OCD.jar"

[4] - Multiplicação

Digite a operação:
4

Digite o segundo número em ponto flutuante padrão IEE754:
1 1,011110100000000000000000 00000100

Operação: Multiplicação
Número 1:
    Decimal: 2.75  Binário: 01000000001100000000000000000000
Número 2:
    Decimal: -23.625  Binário: 11000001101111010000000000000000
Resultado:
    Decimal: -64.96875  Binário: 11000010100000011111000000000000

Entre com qualquer valor para retornar ao menu
```

6,022x10²³ * 1,2046952x10²⁴

Resultado Esperado: Overflow no Expoente

Resultado Obtido: Overflow no Expoente

```
C:\Windows\system32\cmd.exe - java -jar "EP_OCD.jar"

4

Digite o segundo número em ponto flutuante padrão IEE754:
0 1,11111110001101010101000 01001111

Overflow no expoente

Operação: Multiplicação
Número 1:
    Decimal: 6.022E23  Binário: 01100110111111110000101010101000
Número 2:
    Decimal: 1.2046952E24  Binário: 01100110111111110001101010101000
Resultado:
    Decimal: 0.0  Binário: 00000000000000000000000000000000

Entre com qualquer valor para retornar ao menu
```

Referências

Aritmética com Ponto Flutuante:

https://sites.google.com/a/sga.pucminas.br/puc2010-2_a417889/intro-ciencia-computacao/operacoes-ponto-flutuante-ieee754

https://pt.wikipedia.org/wiki/IEEE_754

<https://www.h-schmidt.net/FloatConverter/IEEE754.html>

Algoritmo Booth:

https://pt.wikipedia.org/wiki/Algoritmo_de_multiplicação_de_Booth