Nome: Pedro Augusto Santos Giorgi n°USP:10828972

Nome: Otávio Hideki

Relatório-Operações com binários

**Informações Gerais**

**O que faz:** Esse programa calcula soma, subtração, multiplicação e divisão de números binários, tanto na forma de inteiros como na forma de floats.

**O que foi usado:** Para a criação desse programa, foram utilizadas as IDEs “Eclipse” e “Intellij Ultimate”. Também foi usada a biblioteca padrão do java, como java.util.Scanner e java.util.Arrays

**Classes:**

*Operacoes*

**Descrição**: classe mãe das classes Inteiros e Floats. Contém alguns métodos essenciais aos dois tipos de representação binária.É nela que o método main() está contido

**Métodos:**

*public static void imprimeResultado (String res):* imprime o resultado da operação

*public static void main(String[] args):* método main que executa o programa

----------------------------------------------------------------------------------------------------------

*Inteiros*

**Descrição:** classe filha de Operações. Contém todos os métodos essenciais para as operações com números binários inteiros

**Métodos:**

*char[] conta(char[] binario1,char[] binario2):* método genérico de conta, que serve de base para as classes filhas de Inteiros a sobrescreverem

*private static Inteiros verificaOperacaoInt(String operacao):* verifica qual operação(adição,subtração,divisão ou multiplicação) foi chamada pelo usuário

*public static void script(Scanner entrada):* contém o que aparece para o usuário quando ele seleciona fazer operações com inteiros

----------------------------------------------------------------------------------------------------------

*Floats*

**Descrição:** classe filha de Operações. Contém todos os métodos essenciais para as operações com números binários em formato de float

**Métodos:**

*public String conta(String binario1,String binario2,String expoente1,String expoente2):* método genérico de conta, que serve de base para as classes filhas de Floats a sobrescreverem

*static String somarCentoEVinteOito(String expoente):* soma 128(10000000) ao binário passado como parâmetro

*private static Floats verificaOperacaoFloats(String operacao):* verifica qual operação(adição,subtração,divisão ou multiplicação) foi chamada pelo usuário

*public static void script(Scanner entrada):* contém o que aparece para o usuário quando ele seleciona fazer operações com floats

----------------------------------------------------------------------------------------------------------

IntAdicao

**Descrição:** classe filha de Inteiros. Faz a adição de binários inteiros

**Métodos:**

*static char[] combinar(char[] c1,char[] c2):* combina dois arrays de caracteres

*static char adicaoSimples(char b1,char b2,char ci):* soma três bits

*static char carryOut(char b1,char b2,char ci):* determina o carryOut de três bits

*char[] conta(char[] binario1,char[] binario2):* sobrescreve o método conta() da classe Inteiros. Faz a adição de inteiros.

----------------------------------------------------------------------------------------------------------

IntSubtracao

**Descrição:** classe filha de Inteiros. Faz a subtração de binários inteiros

**Métodos:**

*static char[] complementoDeDois(char[] subtracao):* aplica complemento de dois em um binário passado como parâmetro

*char[] conta(char[] binario1,char[] binario2):* sobrescreve o método conta() da classe Inteiros. Faz a subtração de inteiros.

----------------------------------------------------------------------------------------------------------

IntMultiplicacao

**Descrição:** classe filha de Inteiros. Faz a multiplicação de binários inteiros

**Métodos:**

*public void rightShift(char[] A, int tamanho):* desloca os bits para a direita

*private void display(char[] P, char ch):* mostra os passos do algoritmo na interface

*char[] conta(char[] binario1, char[] binario2):* sobrescreve o método conta() da classe Inteiros. Faz a multiplicação de inteiros.

----------------------------------------------------------------------------------------------------------

IntDivisao

**Descrição:** classe filha de Inteiros. Faz a divisão de binários inteiros

**Métodos:**

*char[] conta(char[] binario1, char[] binario2):* sobrescreve o método conta() da classe Inteiros. Faz a divisão de inteiros.

----------------------------------------------------------------------------------------------------------

FloatAdicao

**Descrição:** classe filha de Floats. Faz a adição de binários em forma de float

**Métodos:**

*public static String rightShift(String binario):* desloca a vírgula para a esquerda

*static String igualarExpoente(String expoente1, String expoente2):* iguala dois expoentes

*public String conta(String binario1,String expoente1,String binario2,String expoente2):* sobrescreve o método conta() da classe Floats. Faz a adição de floats.

----------------------------------------------------------------------------------------------------------

FloatSubtracao

**Descrição:** classe filha de Floats. Faz a subtração de binários em forma de float

**Métodos:**

*public String conta(String binario1,String expoente1,String binario2,String expoente2):* sobrescreve o método conta() da classe Floats. Faz a subtração de floats.

----------------------------------------------------------------------------------------------------------

FloatMultiplicacao

**Descrição:** classe filha de Floats. Faz a multiplicação de binários em forma de float

**Métodos:**

----------------------------------------------------------------------------------------------------------

FloatDivisao

**Descrição:** classe filha de Floats. Faz a divisão de binários em forma de float

**Métodos:**

**Execução**

**Como executar:** para execução pelo cmd, deve ser compilada e executada a classe Operacoes(todas as classes devem estar na mesma pasta): vá ao diretório onde estão contidas as classes do programa e digite “javac Operacoes.java”. Após isso, digite “java Operacoes”.

Após a execução, é só seguir as instruções impressas.

**Alguns detalhes da implementação:**

- Para as operações com float, utilizamos excesso de 128 nos expoentes. Além disso a mantissa impressa é tudo após “1,” e não “0,”

- Na adição e subtração de inteiros, é perguntado a posição do bit de sinal. Isso significa apenas que a resposta não poderá ter mais bits(contando o de sinal) do que o número passado, e a resposta terá o número de bits passado

- A divisão de inteiros devolve o resultado e o resto: ela não dá um resultado “quebrado”

-Quando o resultado de inteiros é negativo, o programa imprime a versão com complemento de 2 e sem complemento de 2.

- Se o binário for negativo, coloque o bit de sinal “1” e passe o binário na forma positiva, pois o programa aplica o complemento de dois automaticamente

- Nas operações com float, o binário passado sempre precisará ter uma vírgula, mesmo que o número após a mesma seja 0. Nesse caso, a escrita seria: xxxx,0 e não xxxx apenas.

-O expoente não tem vírgula

- Alguns erros não têm uma mensagem associada, como por exemplo passar qualquer outro caractere que não seja “0”,”1”, e no caso de floats, “,” para o programa. Se o usuário digitar por exemplo “012\*ni”, o programa só irá parar

**Erros:**

-Overflow: quando o resultado da soma ou subtração da um número com mais bits do que o número passado

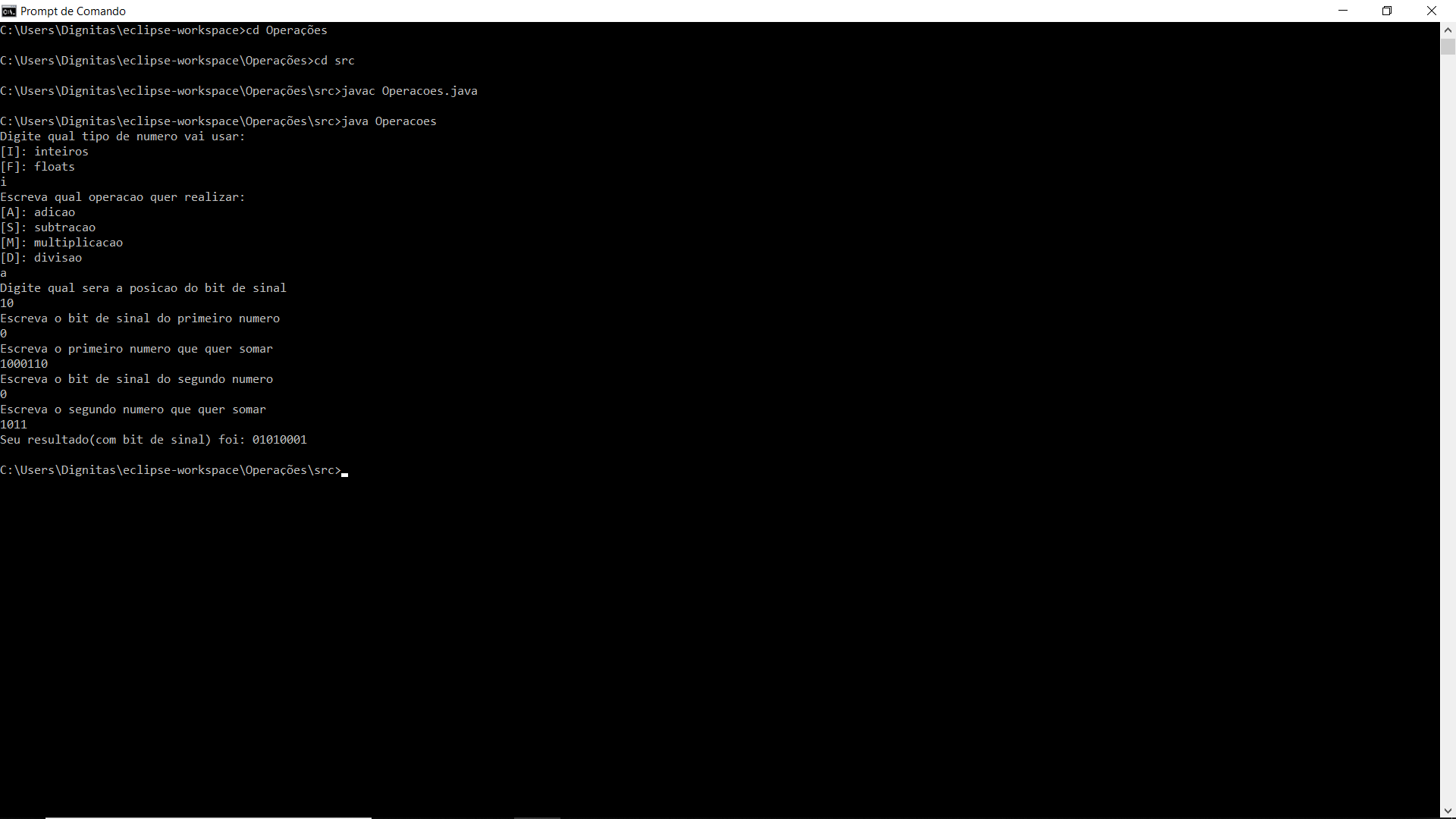
-Overflow de Expoente: quando o expoente passado ou o expoente do resultado é maior que 127(considerando que não se somou 128)

-Underflow de Expoente: quando o expoente passado ou o expoente do resultado é menor que -128(considerando que não se somou 128)

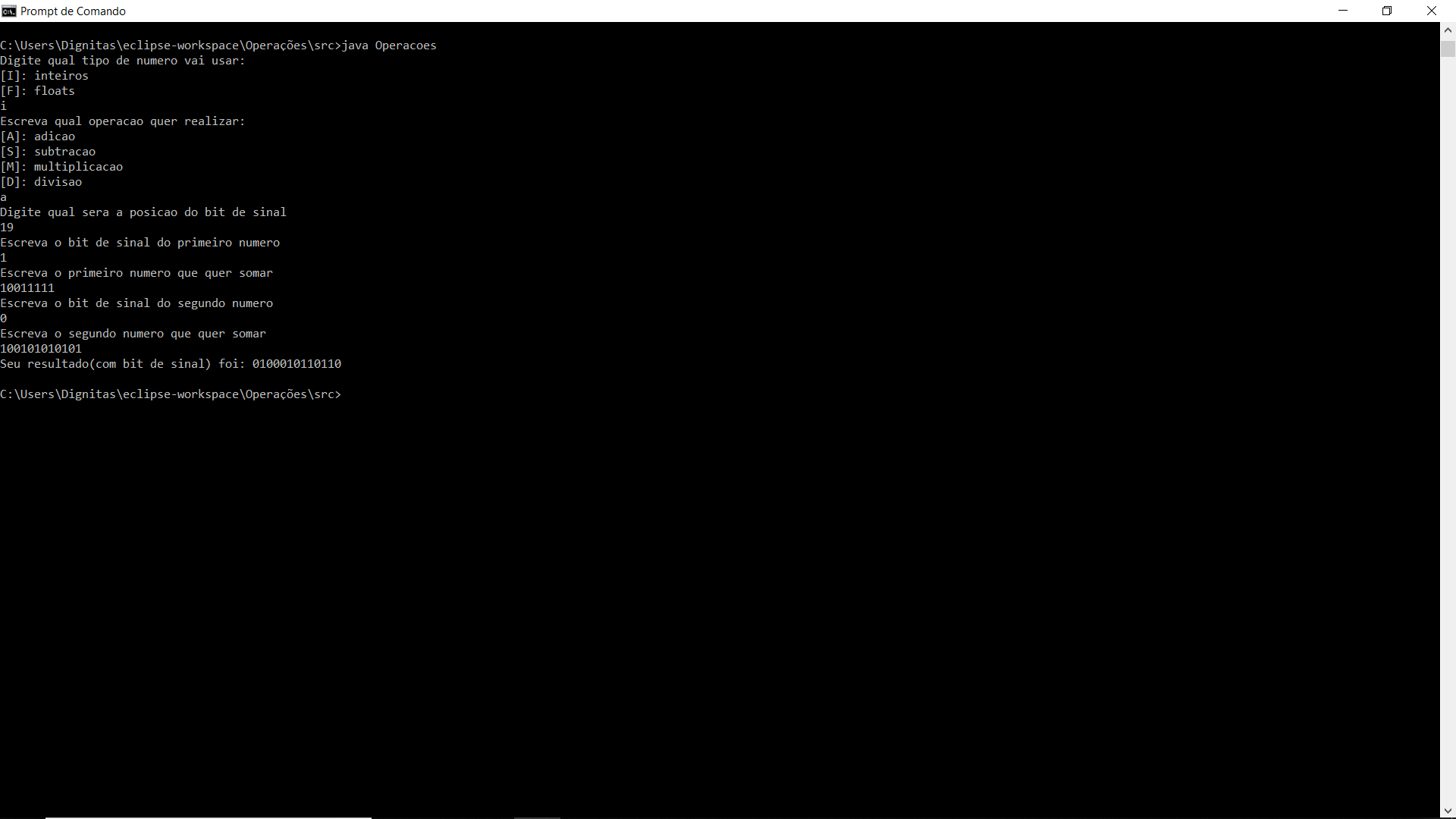
**Testes:**

**IntAdicao**

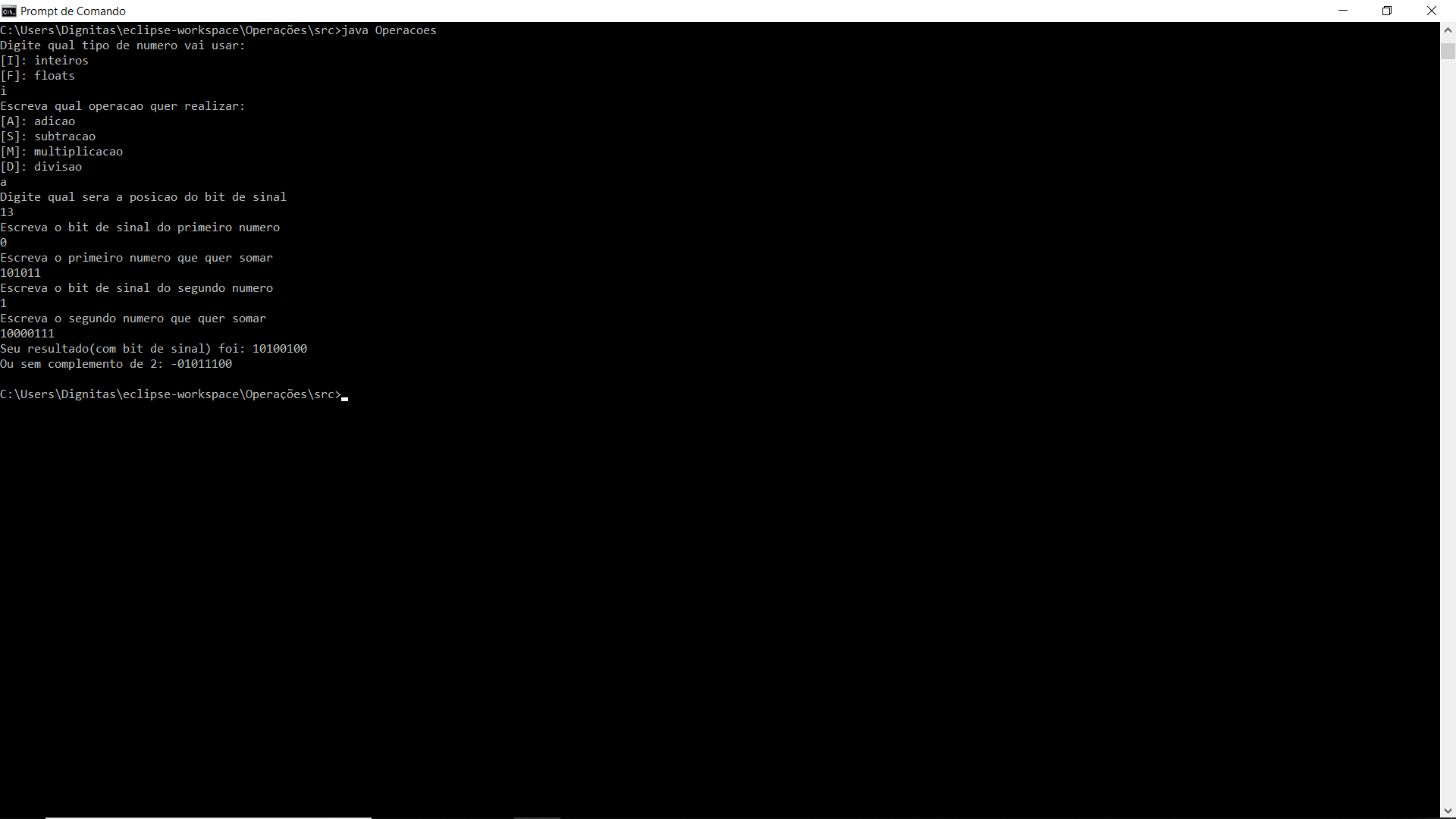
Com os dois números com bit de sinal 0:



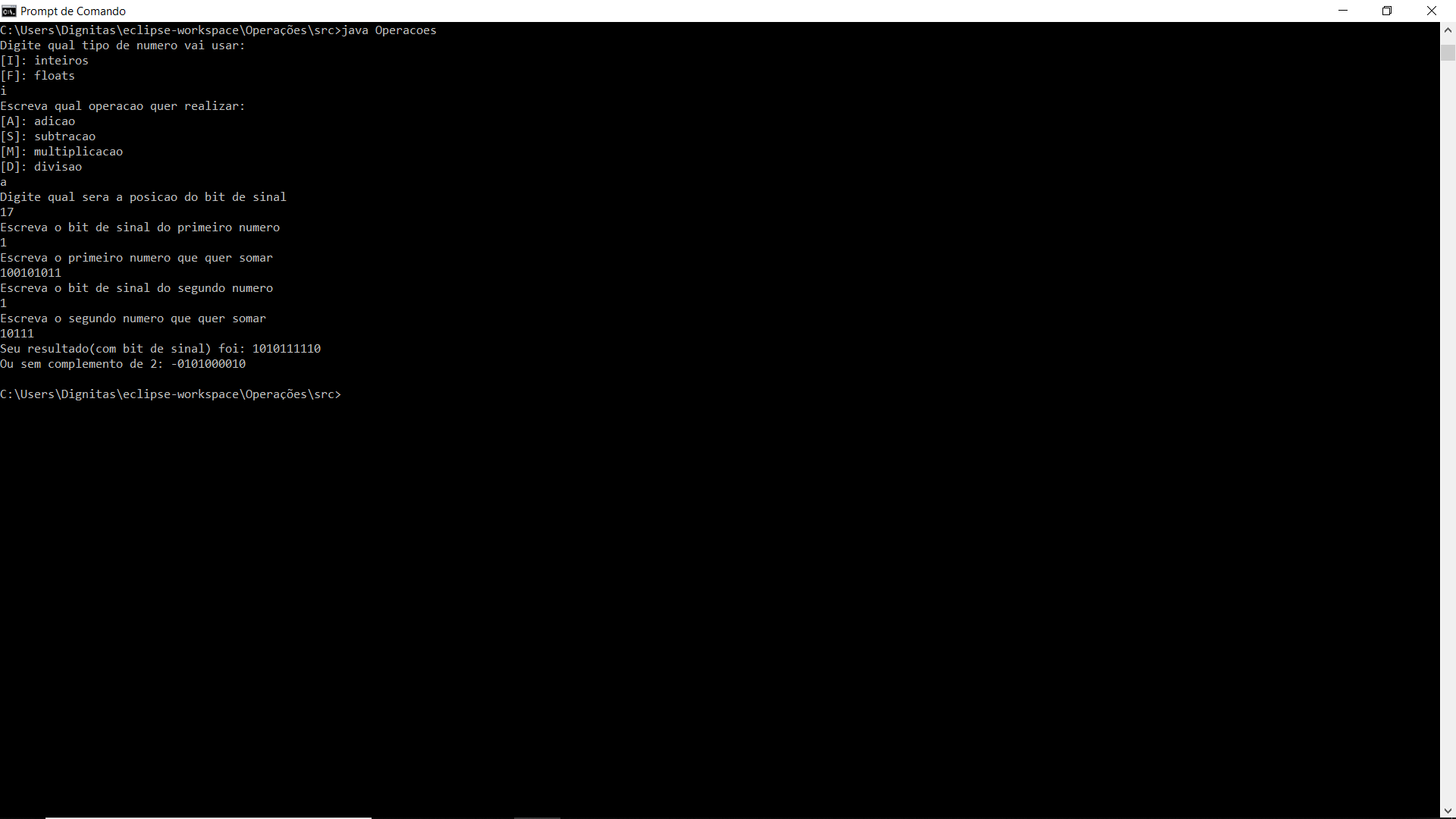
Com o segundo número com bit de sinal 1:



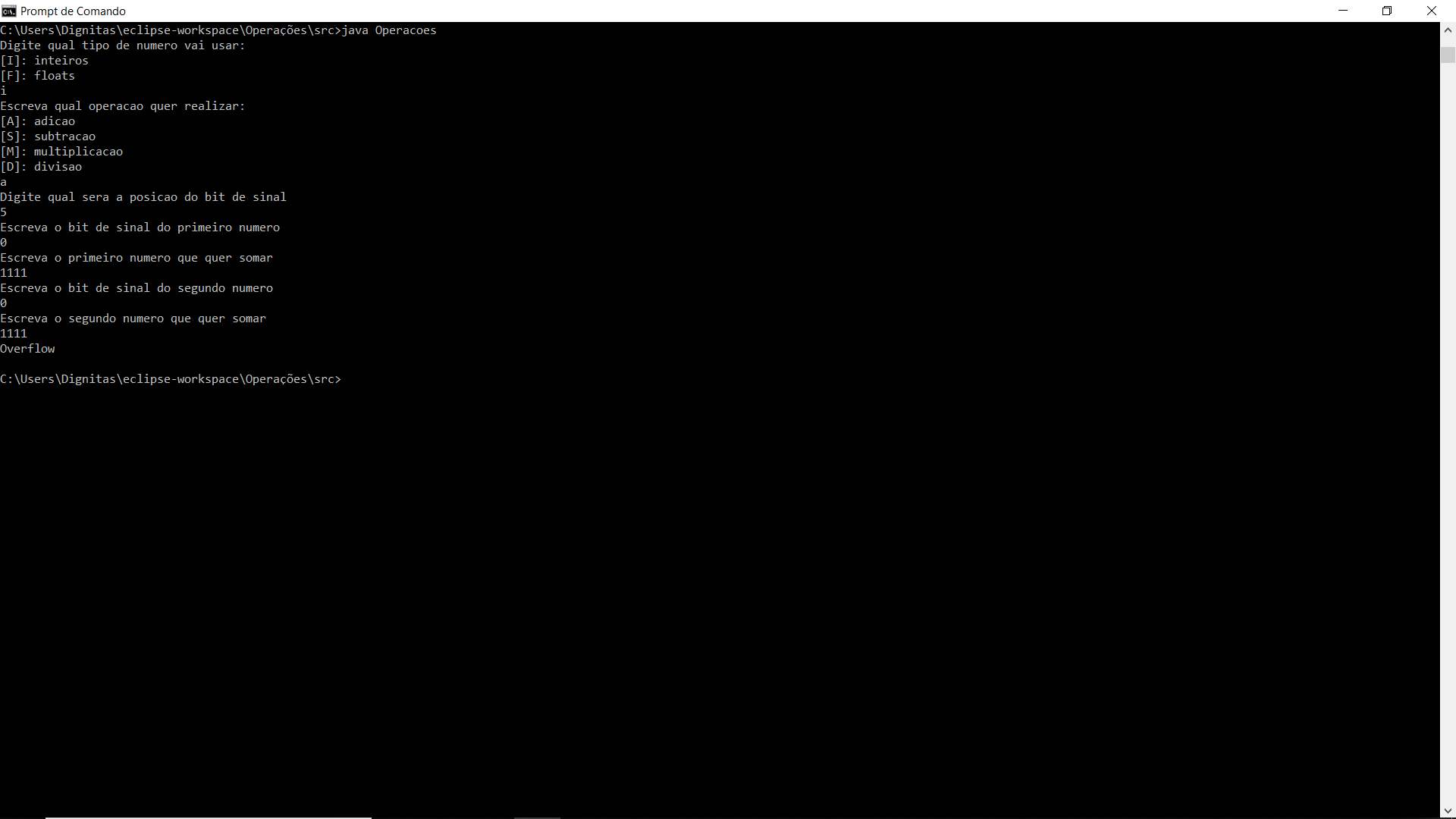
Com o primeiro número com bit de sinal 1:



Com os dois números com bit de sinal 1:

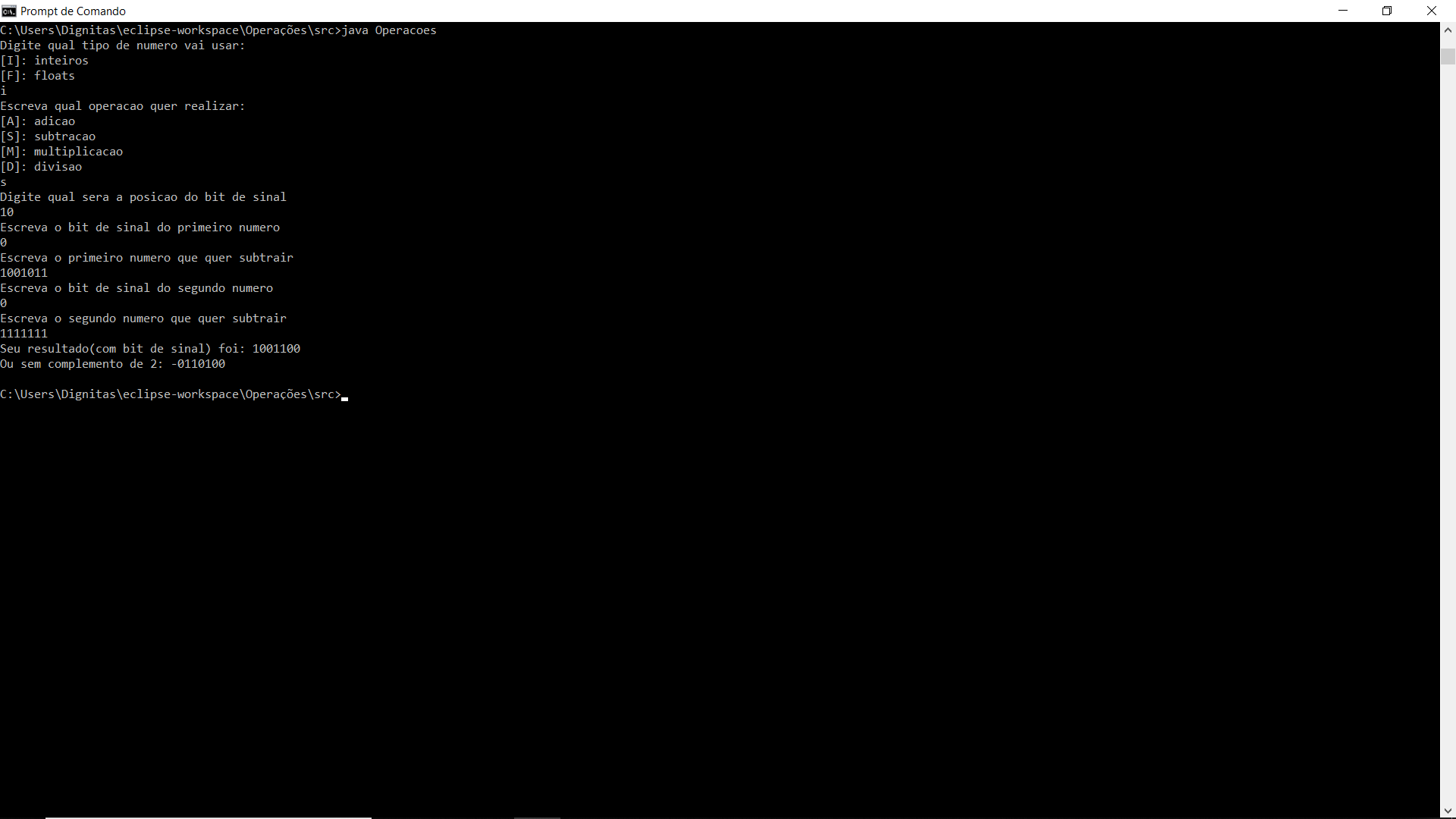


Overflow:

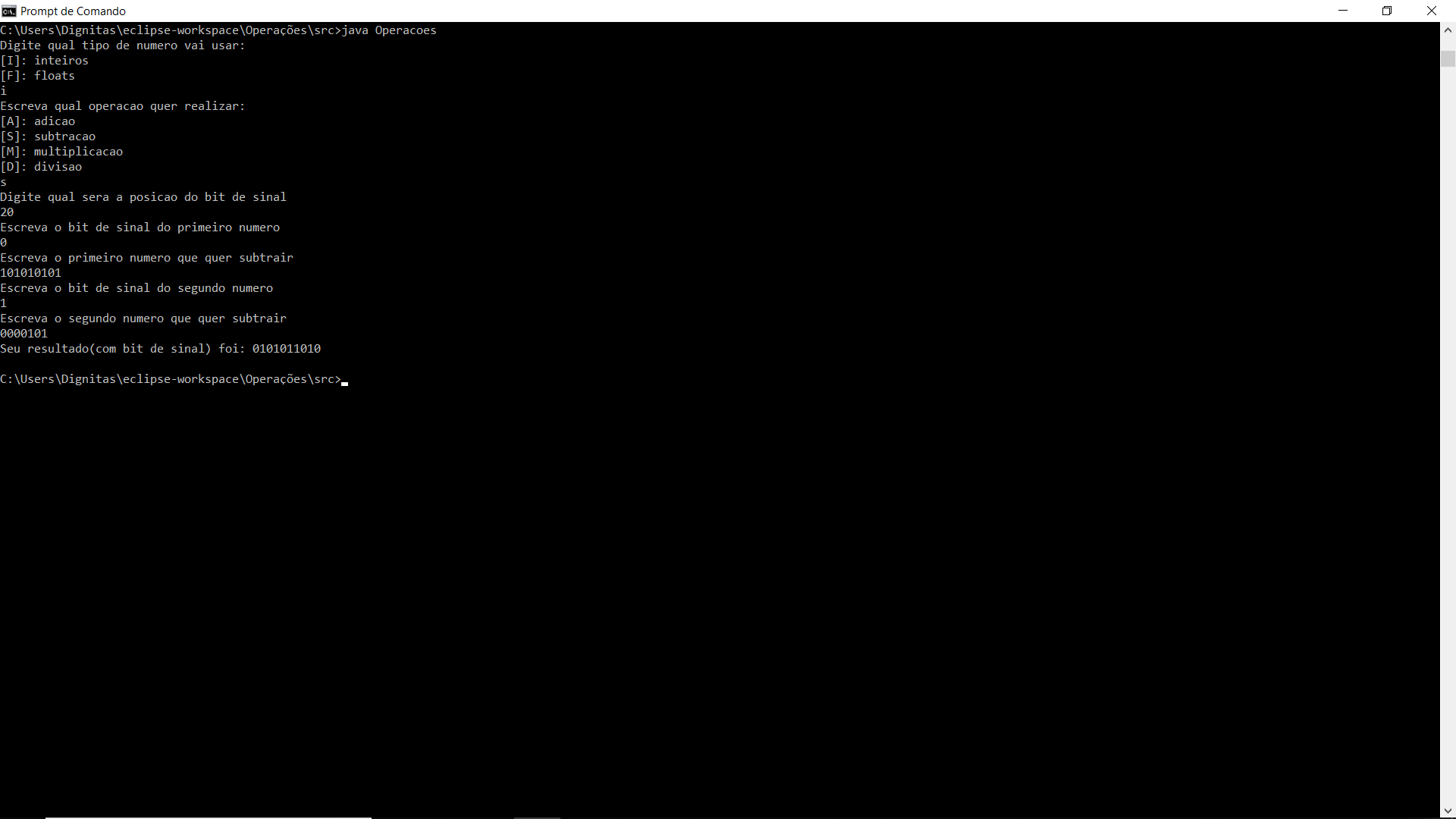


**IntSubtracao**

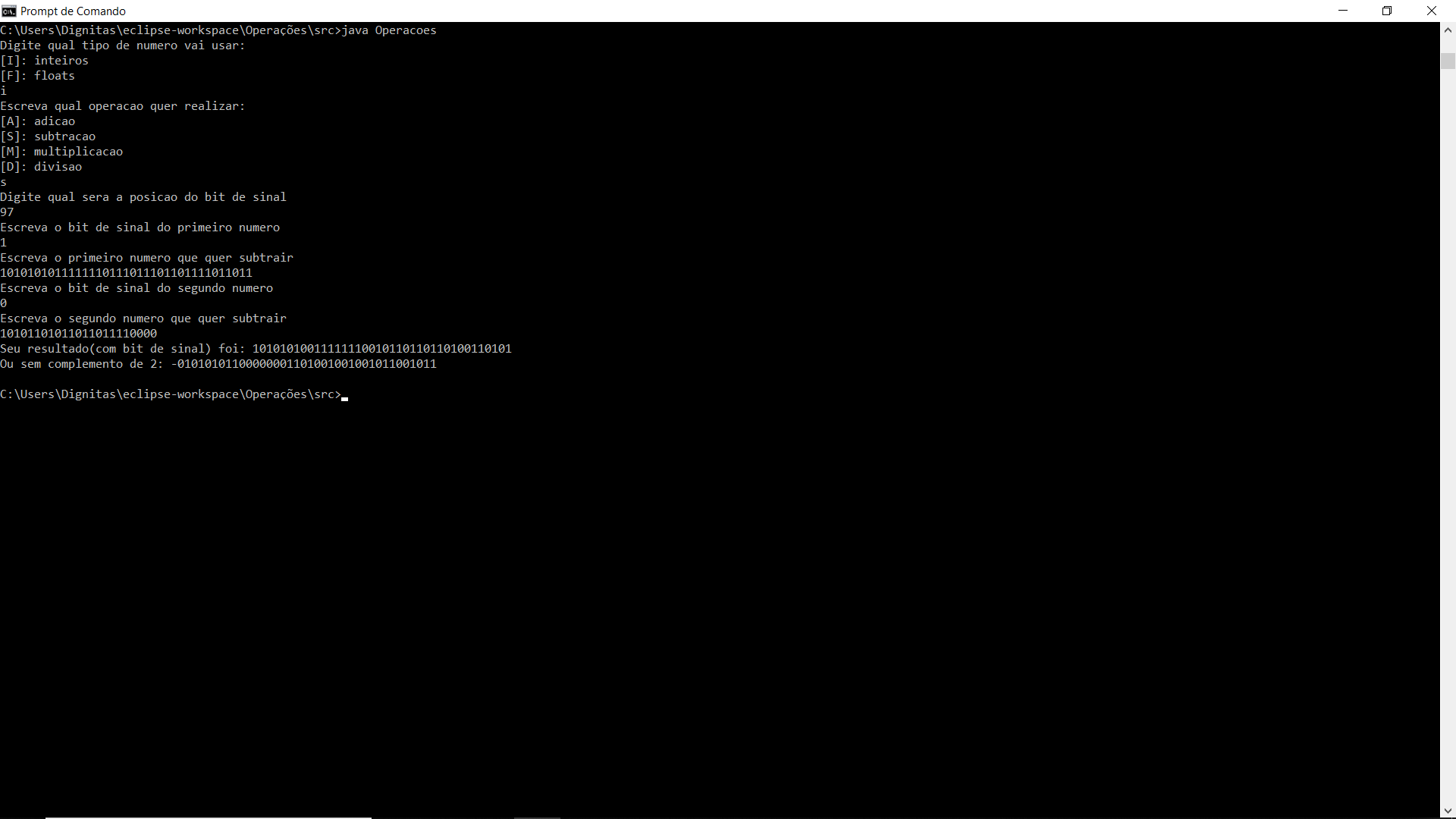
Com os dois números com bit de sinal 0:



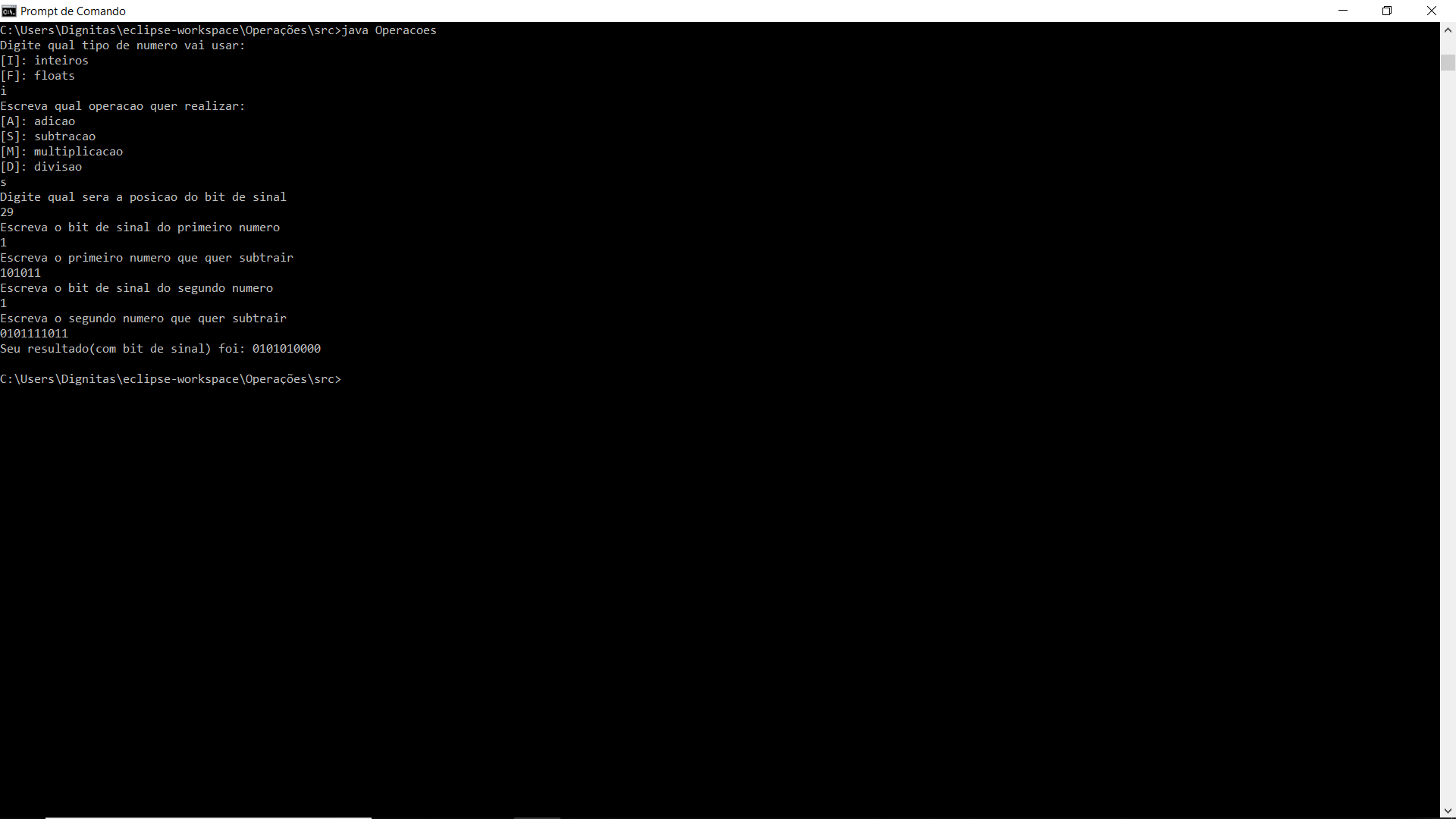
Com o segundo número com bit de sinal 1:



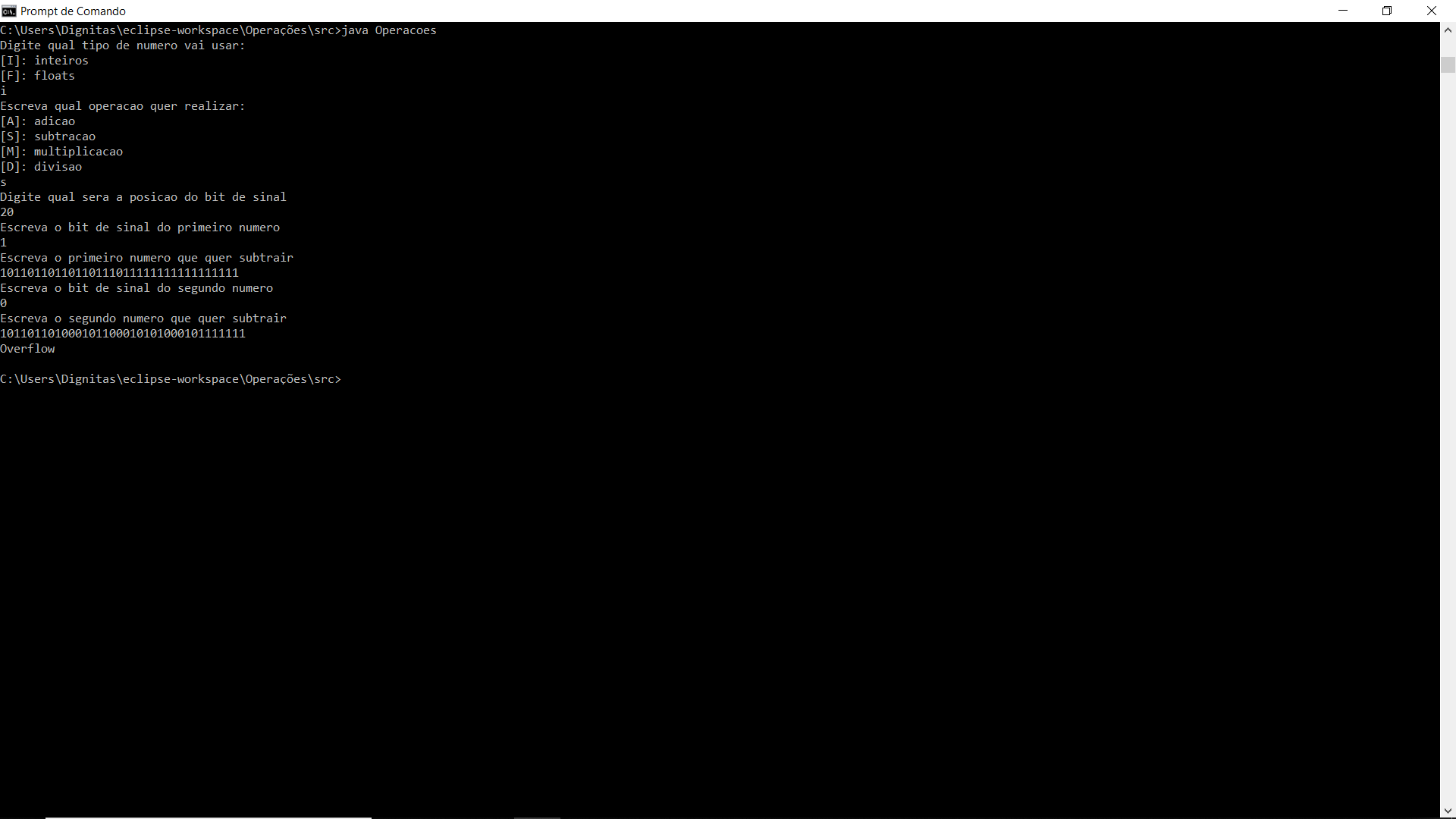
Com o primeiro número com bit de sinal 1:



Com os dois números com bit de sinal 1:

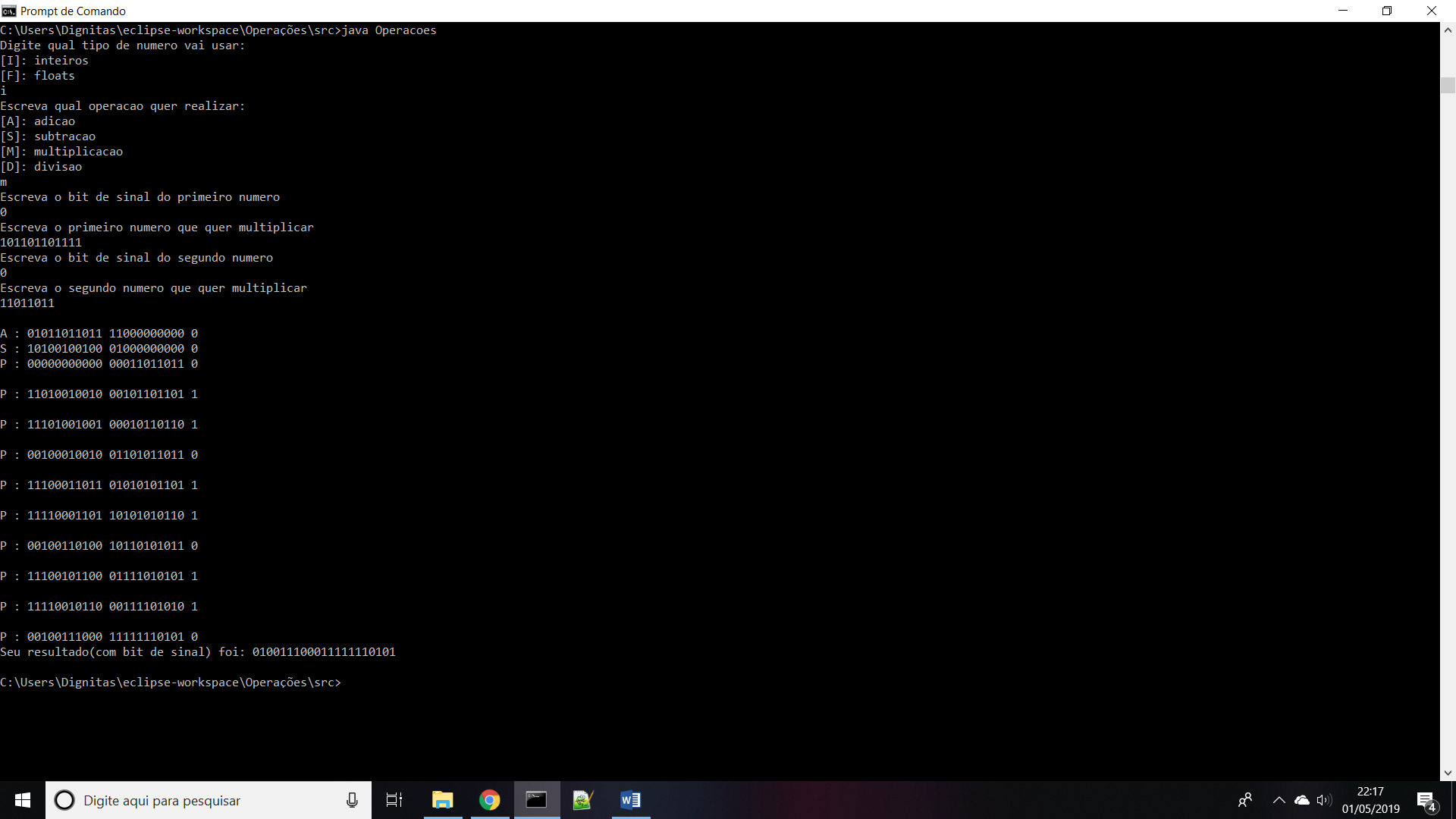


Overflow:

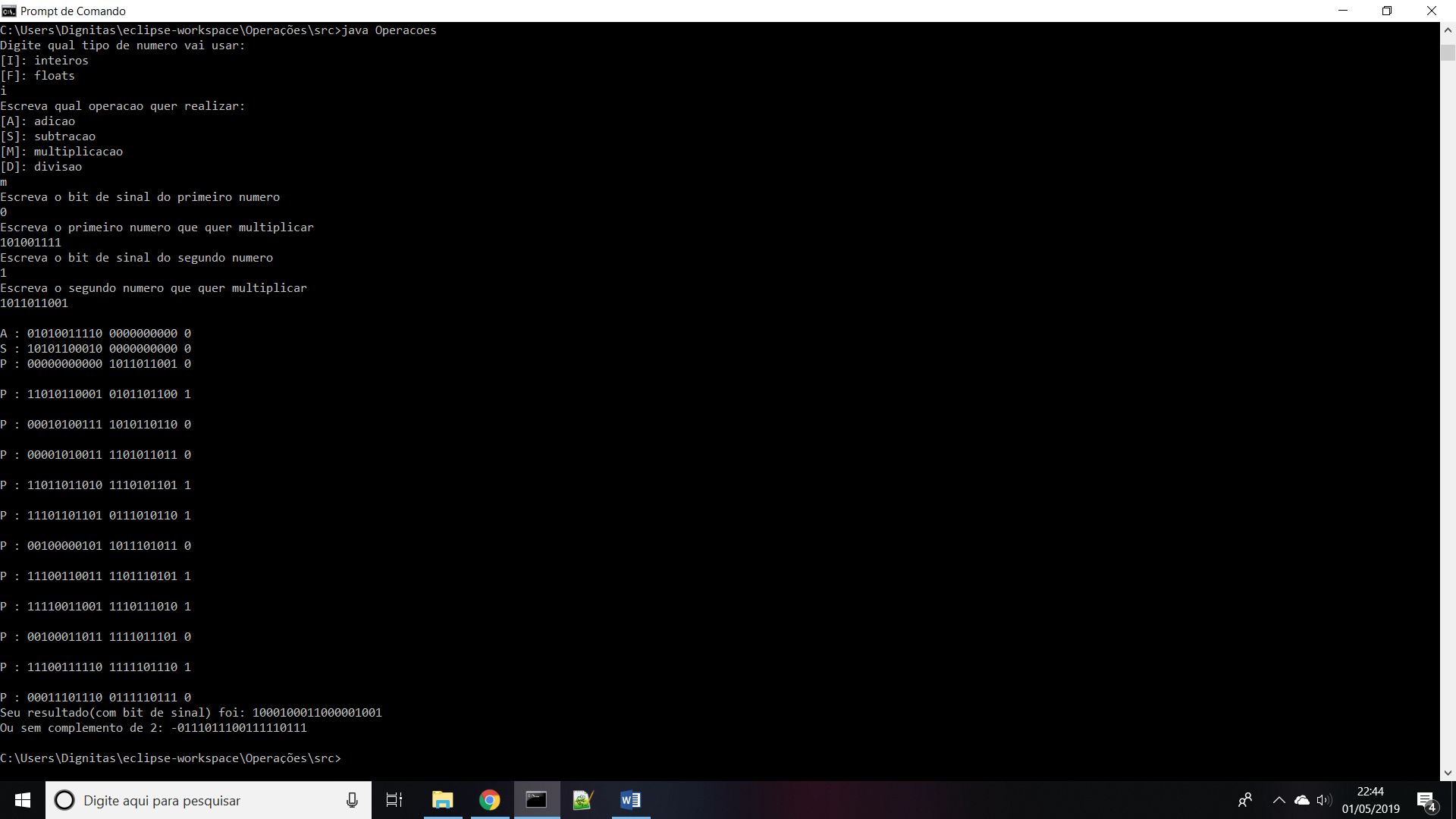


**IntMultiplicacao**

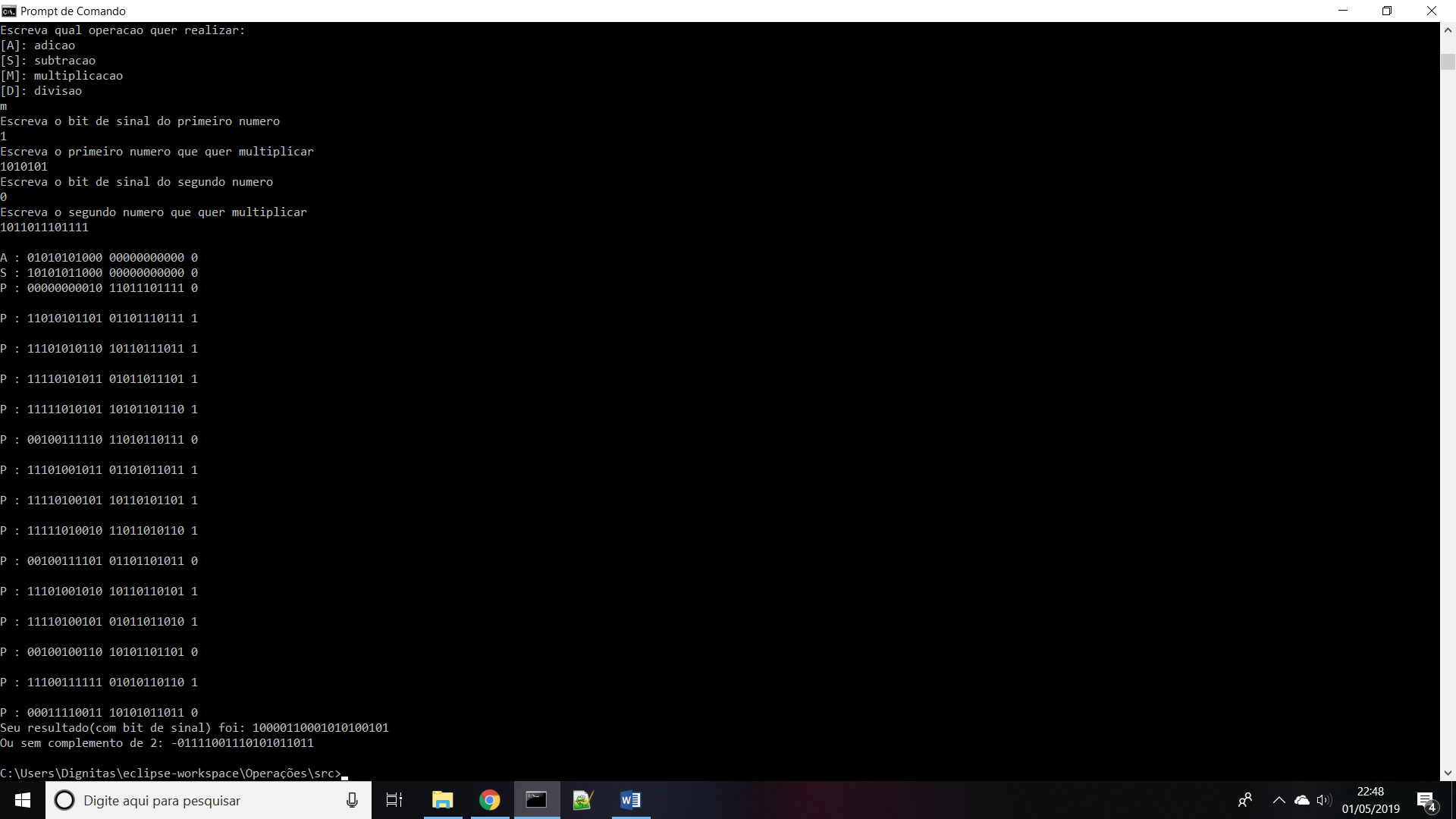
Com os dois números com bit de sinal 0:



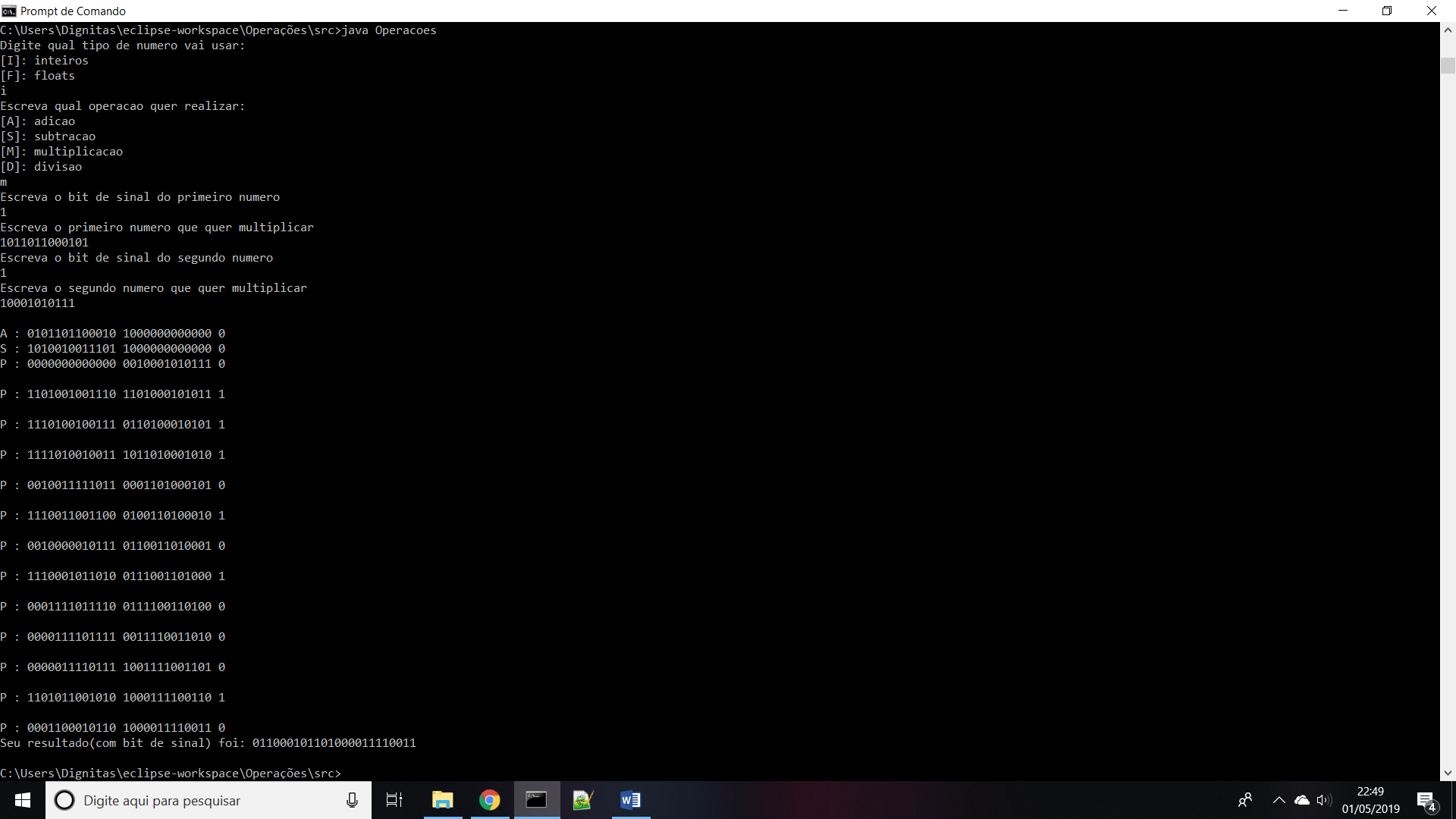
Com o segundo número com bit de sinal 1:



Com o primeiro número com bit de sinal 1:

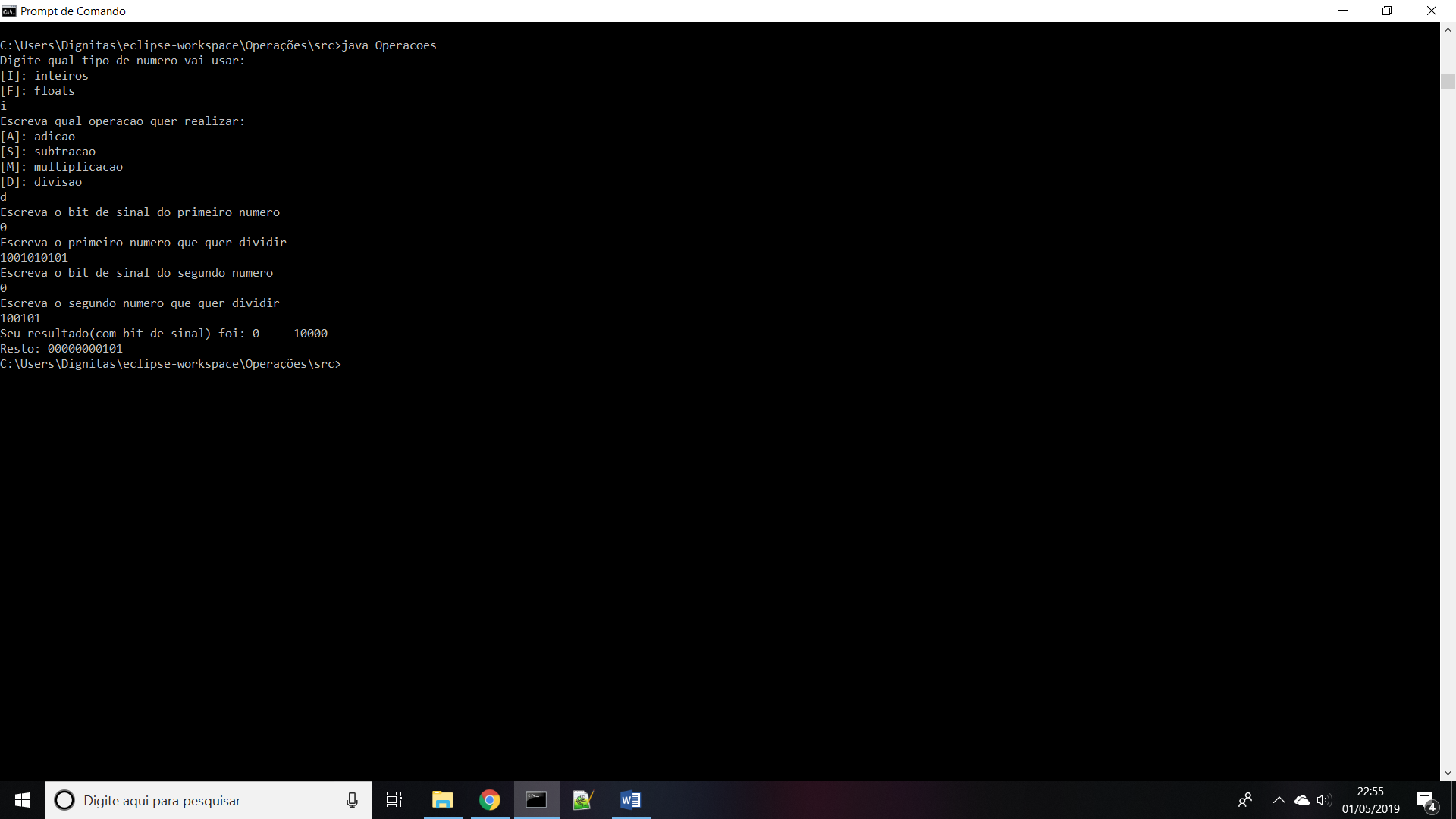


Com os dois números com bit de sinal 1:

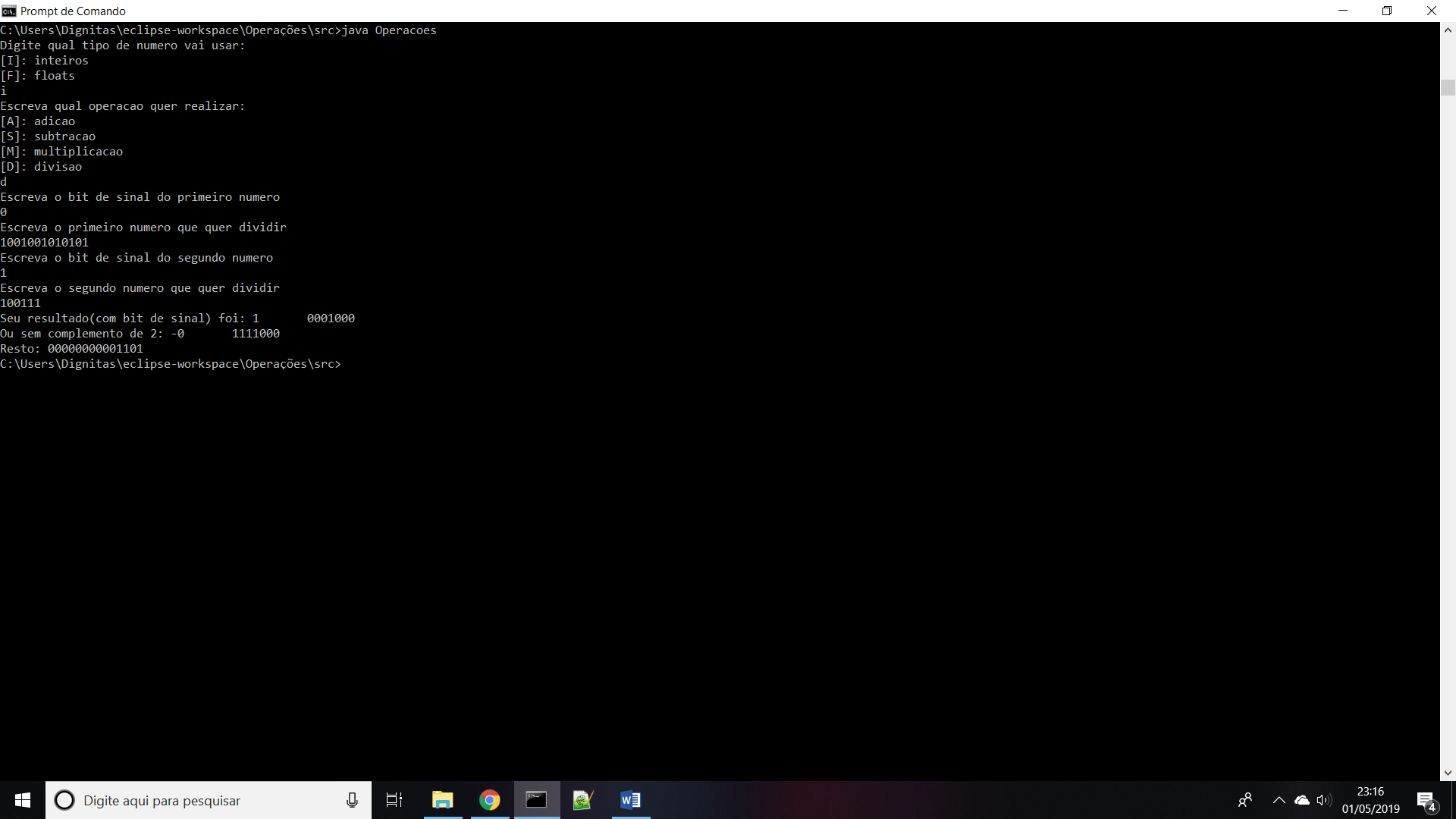


**IntDivisao**

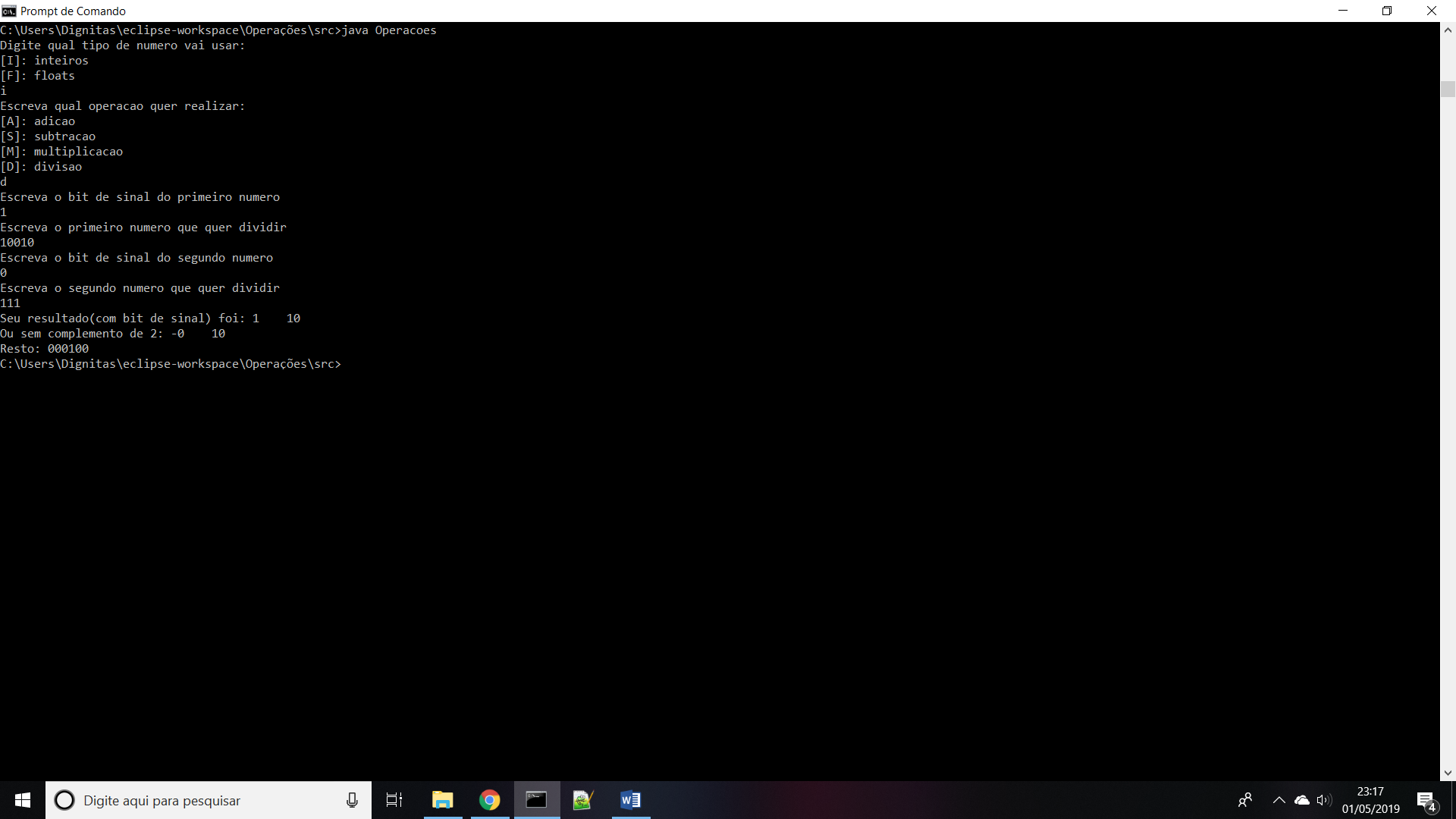
Com os dois números com bit de sinal 0:



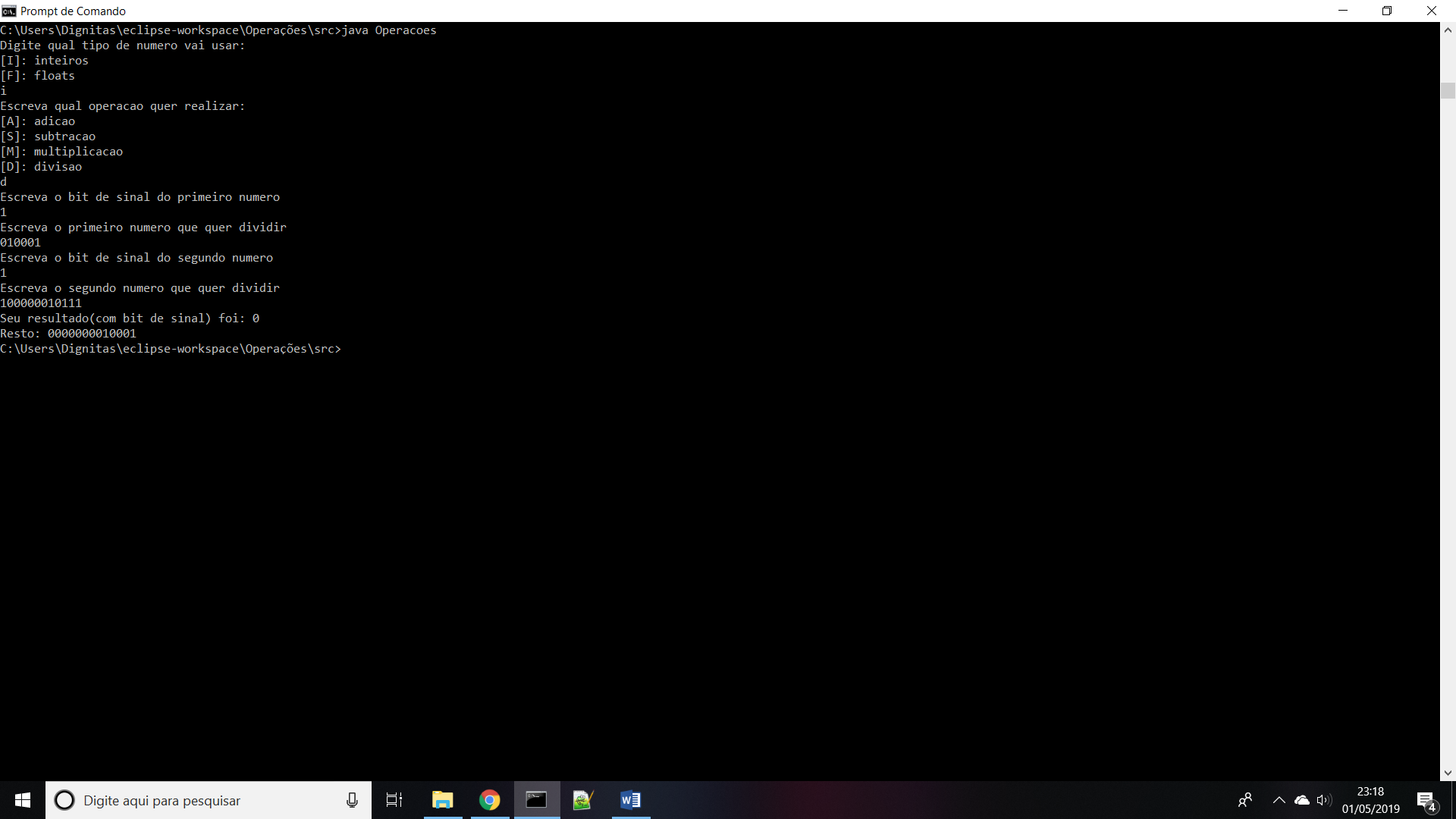
Com o segundo número com bit de sinal 1:



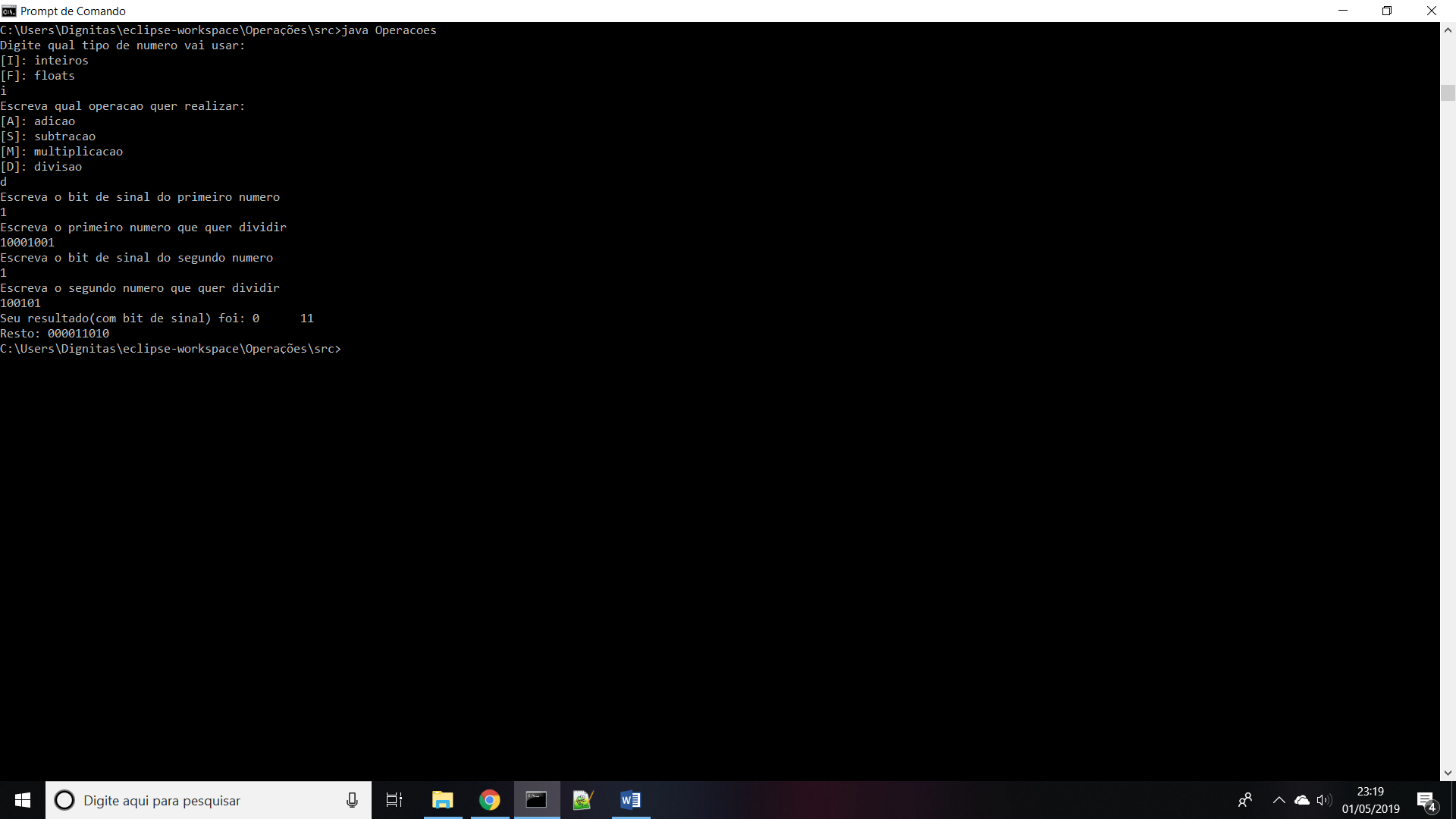
Com o primeiro número com bit de sinal 1:



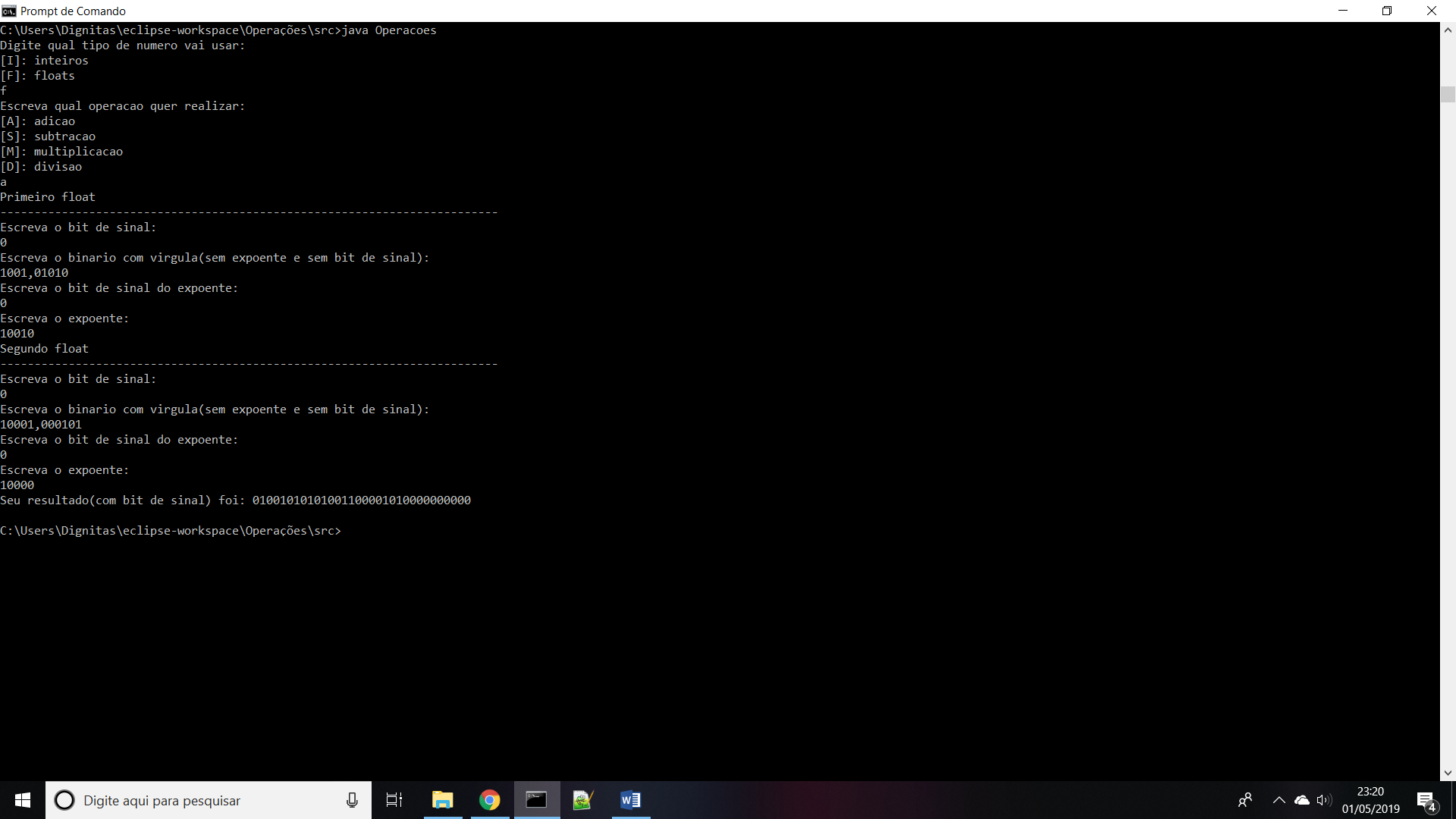
Quando o resultado dá 0:



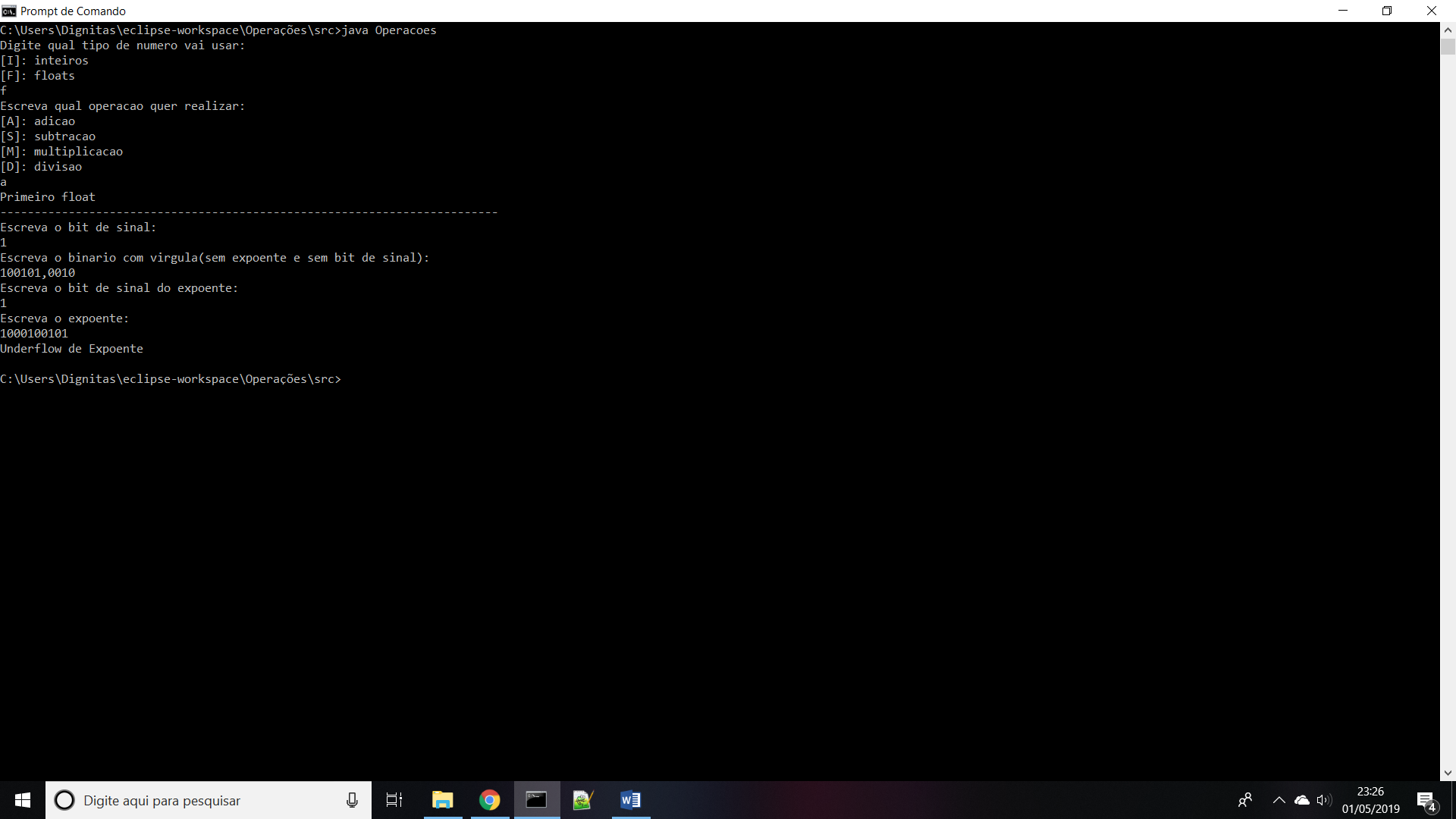
Com os dois números com bit de sinal 1:



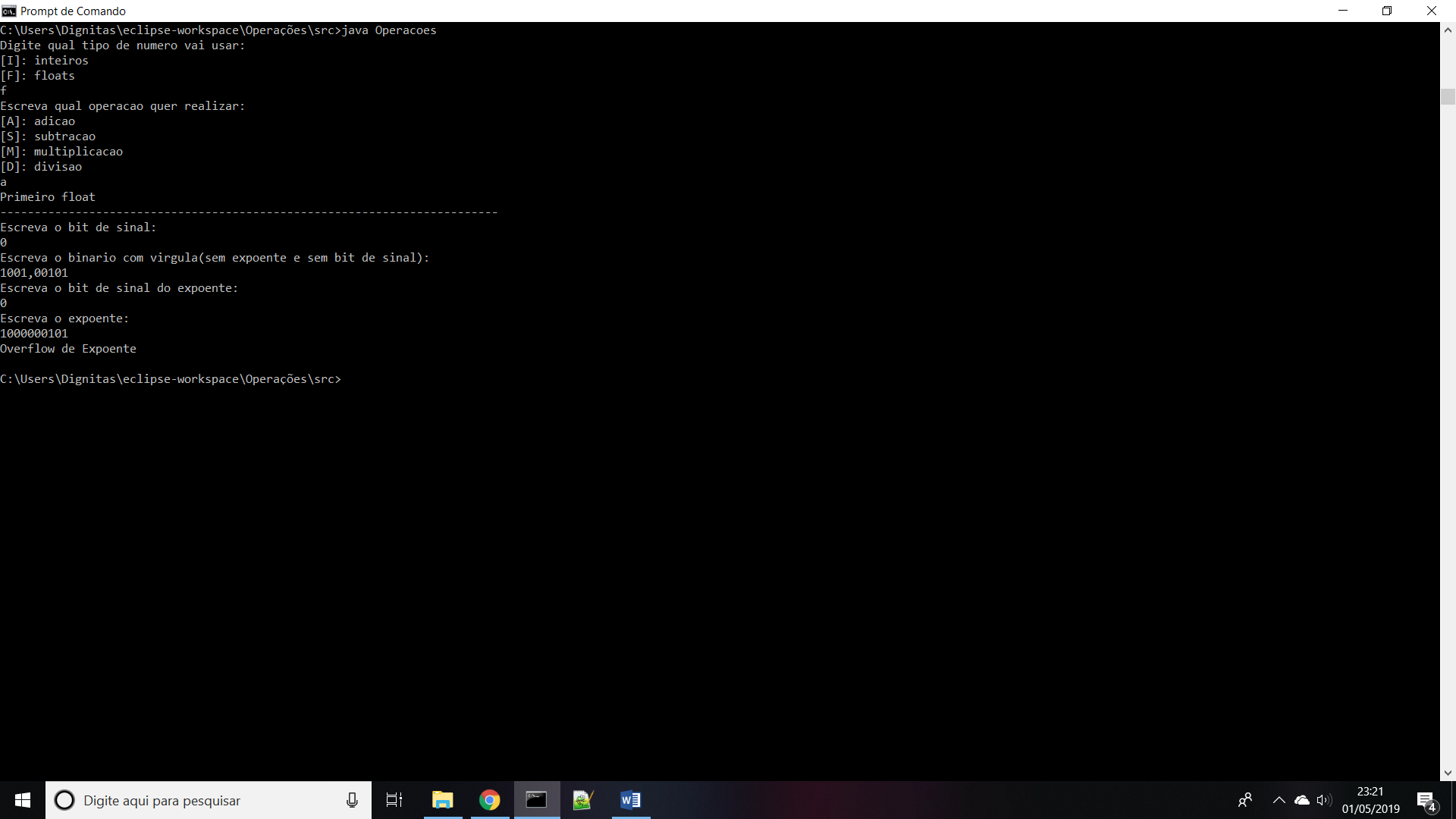
**FloatAdicao**



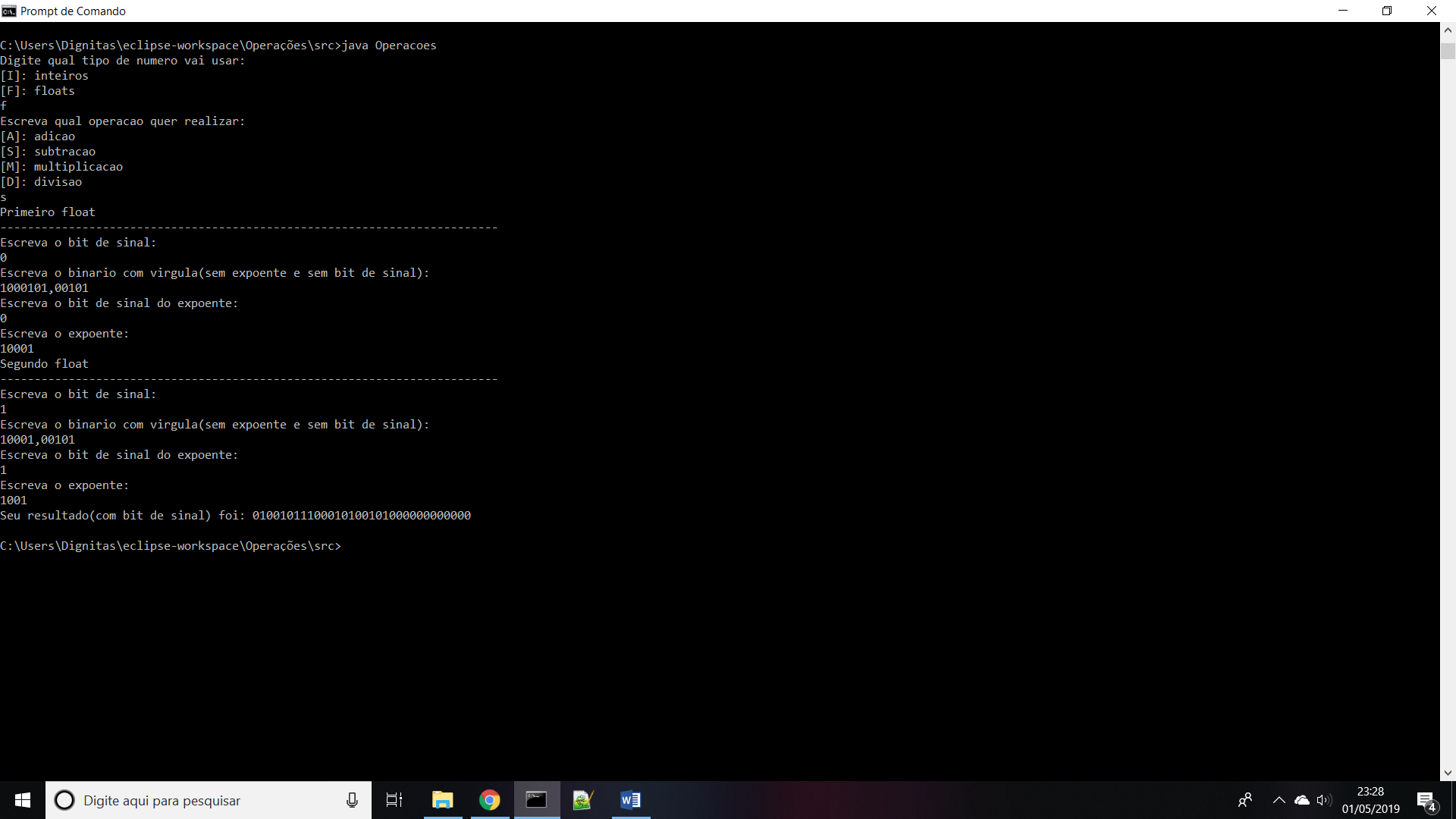
Underflow de expoente:



Overflow de expoente:



FloatSubtracao



FloatMultiplicacao

Com os dois números com bit de sinal 0:

Com o segundo número com bit de sinal 1:

Com o primeiro número com bit de sinal 1:

Com os dois números com bit de sinal 1:

Underflow de expoente:

Overflow de expoente:

FloatDivisao

Com os dois números com bit de sinal 0:

Com o segundo número com bit de sinal 1:

Com o primeiro número com bit de sinal 1:

Com os dois números com bit de sinal 1:

Underflow de expoente:

Overflow de expoente: