1. **BINARY TO DECIMAL CONVERTION**

**AIM:**

To write a Python program to convert binary to decimal numbers

**PROGRAM:**

b\_num = list(input("Input a binary number: "))

value = 0

for i in range(len(b\_num)):

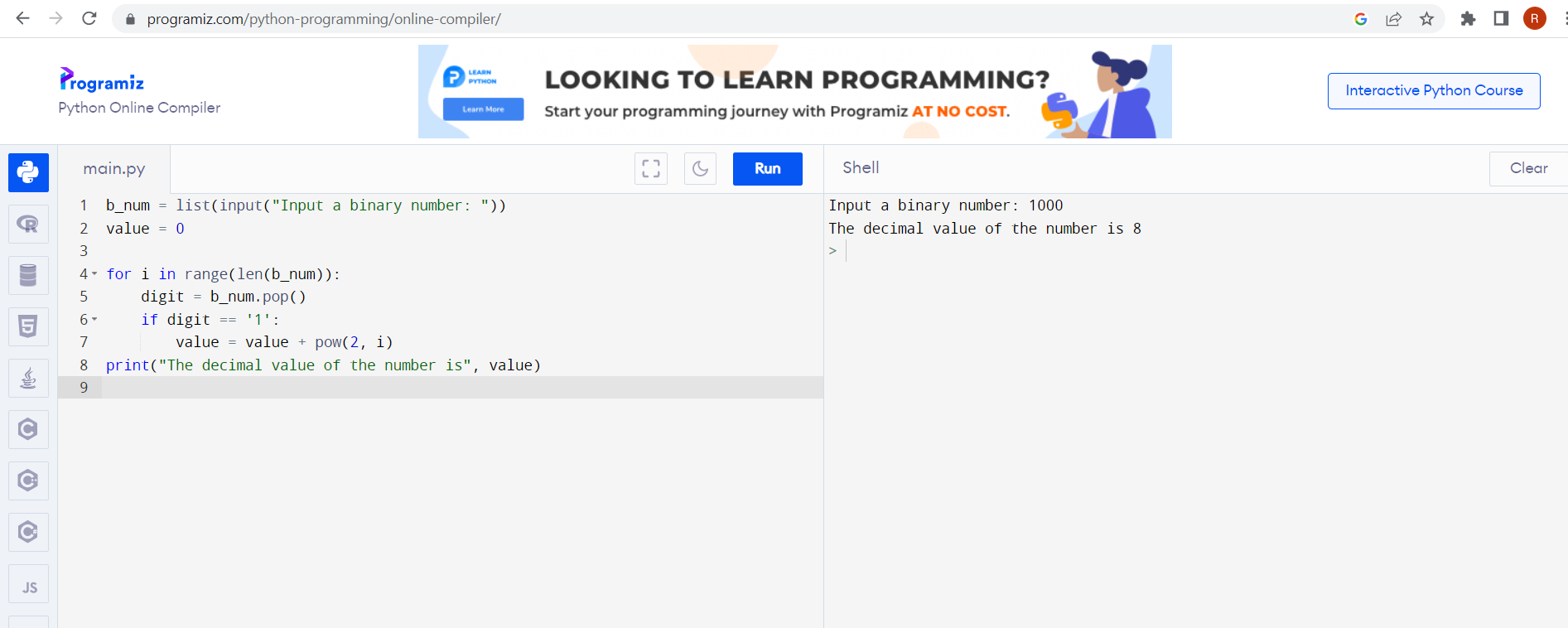
digit = b\_num.pop()

if digit == '1':

value = value + pow(2, i)

print ("The decimal value of the number is", value)

**OUTPUT:**

****

**RESULT:**

The given program to convert binary to decimal numbers is completed successfully

1. **BINARY TO OCTAL CONVERTION**

**AIM:**

To write a Python program to convert binary to octal numbers

**PROGRAM:**

b\_num = list(input("Input a binary number: "))

value = 0

for i in range(len(b\_num)):

digit = b\_num.pop()

if digit == '1':

value = value + pow(2, i)

print("The decimal value of the number is", value)

x= value

def dec2oct(x):

if x < 8:

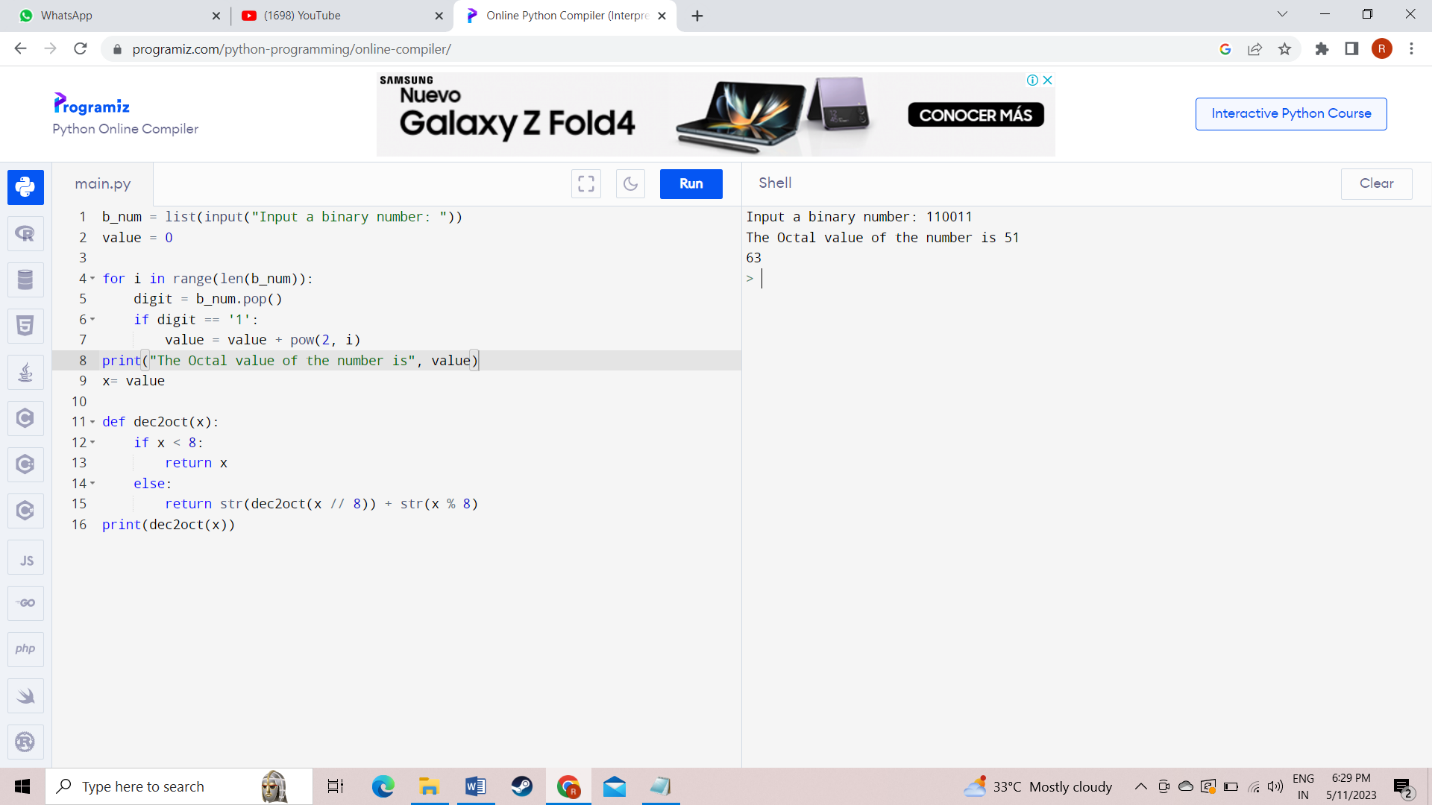
return x

else:

return str(dec2oct(x // 8)) + str(x % 8)

print(dec2oct(x))

**OUTPUT:**

****

**RESULT:**

The given program to convert binary to octal numbers is successfully completed

**3. BINARY TO HEXADECIMAL CONVERTION**

**AIM:**

To write a python program to convert binary to hexadecimal numbers

**PROGRAM:**

b\_num = list(input("Input a binary number: "))

value = 0

for i in range(len(b\_num)):

digit = b\_num.pop()

if digit == '1':

value = value + pow(2, i)

print("The decimal value of the number is", value)

dechimal\_nums = [value]

def dechimal\_to\_Hex(dechimal\_nums):

digits = "0123456789ABCDEF"

x = (dechimal\_nums % 16)

rest\_part = dechimal\_nums // 16

if (rest\_part == 0):

return digits[x]

return dechimal\_to\_Hex(rest\_part) + digits[x]

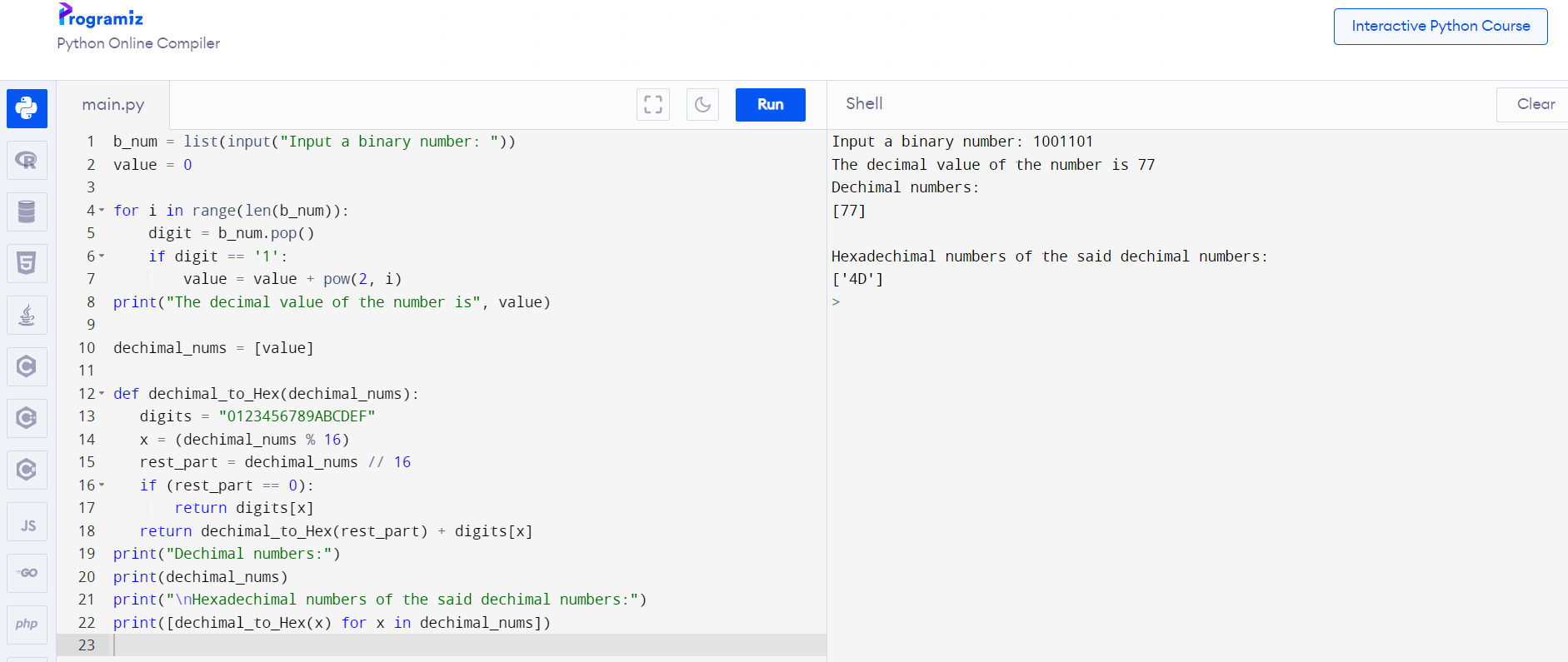
print("Decimal numbers:")

print(dechimal\_nums)

print("\nHexadecimal numbers of the said decimal numbers:")

print([dechimal\_to\_Hex(x) for x in dechimal\_nums])

**OUTPUT:**

****

**RESULT:**

The given program to convert binary to Hexadecimal numbers is successfully completed

1. **HEXADECIMAL TO BINARY CONVERTION**

**AIM:**

To write a python program to convert hexadecimal to binary numbers.

**PROGRAM:**

hexdecnum= input ("Enter the Hexadecimal Number: ")

binnum = ""

hexlen = len(hexdecnum)

i = 0

while i<hexlen:

if hexdecnum[i] == '0':

binnum = binnum + "0000"

elif hexdecnum[i] == '1':

binnum = binnum + "0001"

elif hexdecnum[i] == '2':

binnum = binnum + "0010"

elif hexdecnum[i] == '3':

binnum = binnum + "0011"

elif hexdecnum[i] == '4':

binnum = binnum + "0100"

elif hexdecnum[i] == '5':

binnum = binnum + "0101"

elif hexdecnum[i] == '6':

binnum = binnum + "0110"

elif hexdecnum[i] == '7':

binnum = binnum + "0111"

elif hexdecnum[i] == '8':

binnum = binnum + "1000"

elif hexdecnum[i] == '9':

binnum = binnum + "1001"

elif hexdecnum[i] == 'a' or hexdecnum[i] == 'A':

binnum = binnum + "1010"

elif hexdecnum[i] == 'b' or hexdecnum[i] == 'B':

binnum = binnum + "1011"

elif hexdecnum[i] == 'c' or hexdecnum[i] == 'C':

binnum = binnum + "1100"

elif hexdecnum[i] == 'd' or hexdecnum[i] == 'D':

binnum = binnum + "1101"

elif hexdecnum[i] == 'e' or hexdecnum[i] == 'E':

binnum = binnum + "1110"

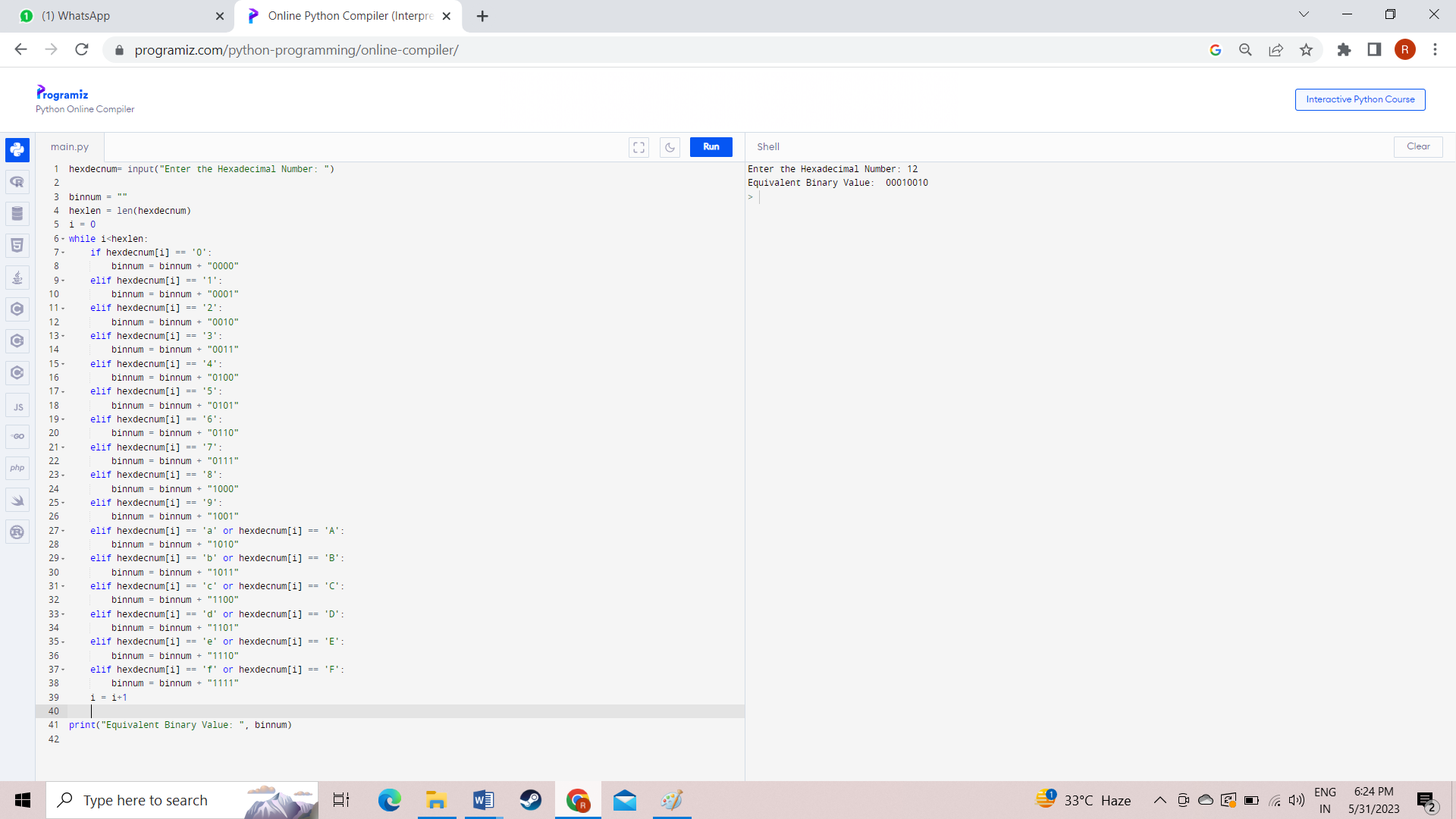
elif hexdecnum[i] == 'f' or hexdecnum[i] == 'F':

binnum = binnum + "1111"

i = i+1

print ("Equivalent Binary Value: ", binnum)

**OUTPUT:**

****

**RESULT:**

The given program to convert Hexadecimal to binary numbers is successfully completed

1. **HEXADECIMAL TO OCTAL CONVERTION**

**AIM:**

To write a python program to convert hexadecimal to octal numbers

**PROGRAM:**

hexdecnum= input("Enter the Hexadecimal Number: ")

chk = 0

decnum = 0

hexdecnumlen = len(hexdecnum)

hexdecnumlen = hexdecnumlen-1

i = 0

while hexdecnumlen>=0:

rem = hexdecnum[hexdecnumlen]

if rem>='0' and rem<='9':

rem = int(rem)

elif rem>='A' and rem<='F':

rem = ord(rem)

rem = rem-55

elif rem>='a' and rem<='f':

rem = ord(rem)

rem = rem-87

else:

chk = 1

break

decnum = decnum + (rem \* (16 \*\* i))

hexdecnumlen = hexdecnumlen-1

i = i+1

if chk==0:

i = 0

octnum = []

while decnum! =0:

rem = decnum%8

octnum.insert(i, rem)

i = i+1

decnum = int(decnum/8)

print("\nEquivalent Octal Value is: ")

i = i-1

while i>=0:

print(octnum[i], end="")

i = i-1

print()

else:

print("\nInvalid Input!")

**OUTPUT**



**RESULT:**

The given program to convert Hexadecimal to Octal numbers is successfully completed

1. **HEXADECIMAL TO DECIMAL CONVERTION**

**AIM:**

To write a python program to convert hexadecimal to decimal numbers.

**PROGRAM:**

hexnum= input ("Enter the Hexadecimal Number: ")

chk = 0

decnum = 0

hexnumlen = len(hexnum)

hexnumlen = hexnumlen-1

i = 0

while hexnumlen>=0:

rem = hexnum[hexnumlen]

if rem>='0' and rem<='9':

rem = int(rem)

elif rem>='A' and rem<='F':

rem = ord(rem)

rem = rem-55

elif rem>='a' and rem<='f':

rem = ord(rem)

rem = rem-87

else:

chk = 1

break

decnum = decnum + (rem \* (16 \*\* i))

hexnumlen = hexnumlen-1

i = i+1

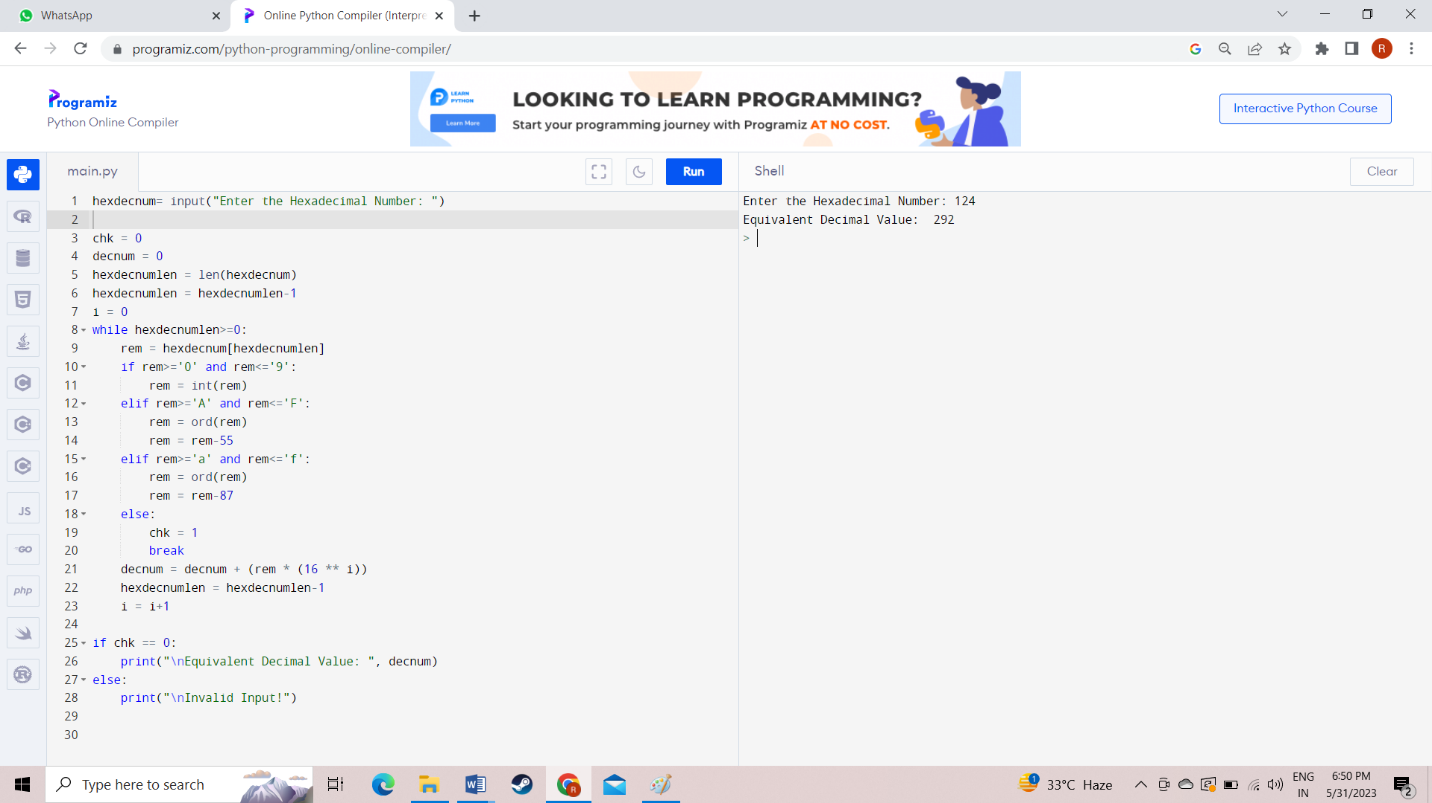
if chk == 0:

print ("\nEquivalent Decimal Value: ", decnum)

else:

print ("\nInvalid Input!")

**OUTPUT:**

****

**RESULT:**

The given program to convert Hexadecimal numbers to decimal is successfully completed

1. **DECIMAL TO BINARY CONVERTION**

**AIM:**

To write a python program to convert decimal to binary numbers

**PROGRAM:**

def convertToBinary(n):

if n > 1:

convertToBinary(n//2)

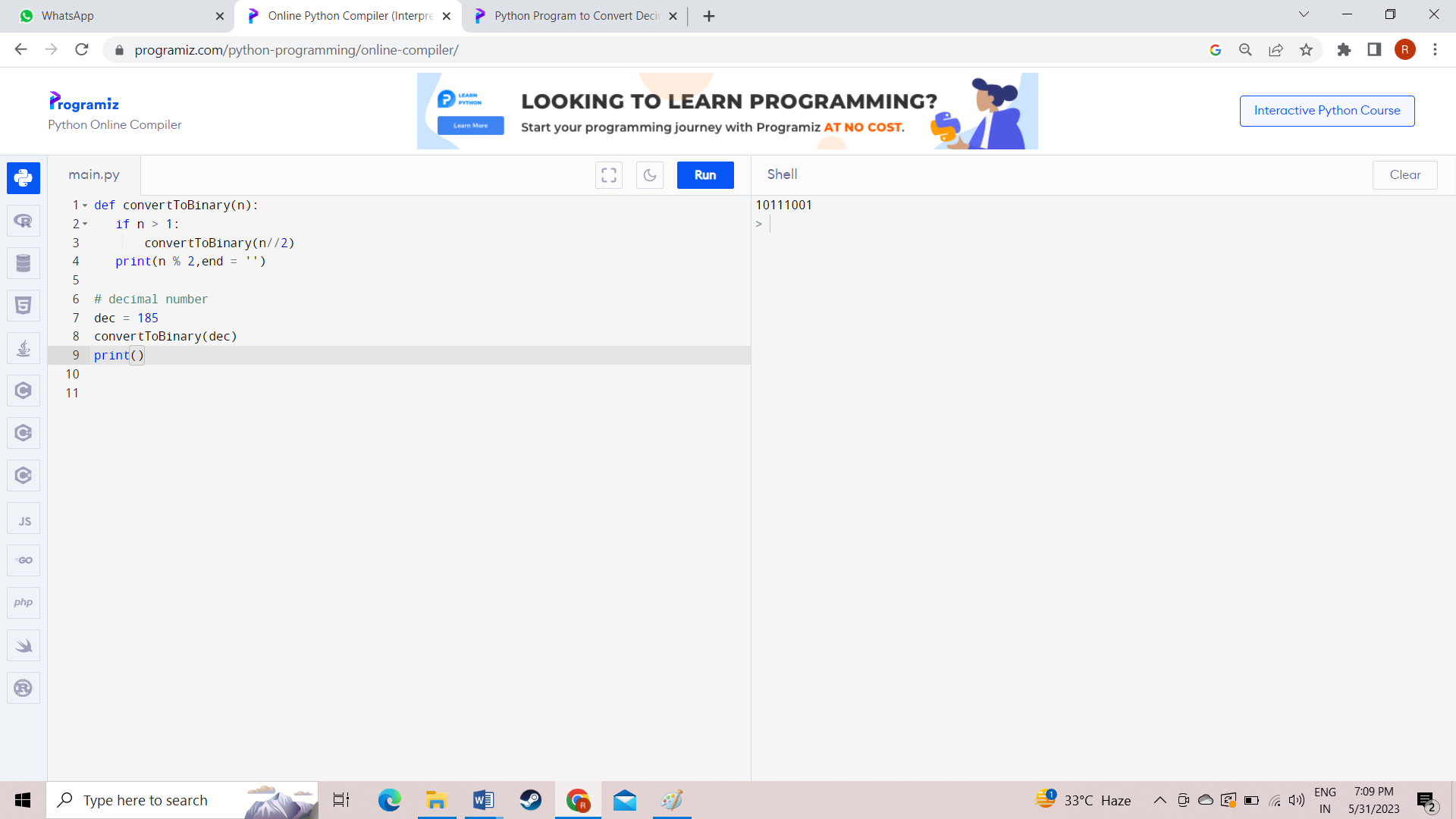
print (n % 2,end = '')

dec = 185

convertToBinary(dec)

print ()

**OUTPUT:**

****

**RESULT:**

The given program to convert decimal to binary numbers is successfully completed

1. **DECIMAL TO OCTAL CONVERTION**

**AIM:**

To write a python program to convert decimal to octal numbers

**PROGRAM:**

decnum = int(input("Enter the Decimal Number: "))

i = 0

octnum = [ ]

while decnum!=0:

rem = decnum%8

octnum.insert(i, rem)

i = i+1

decnum = int(decnum/8)

print("\nEquivalent Octal Value is: ")

i = i-1

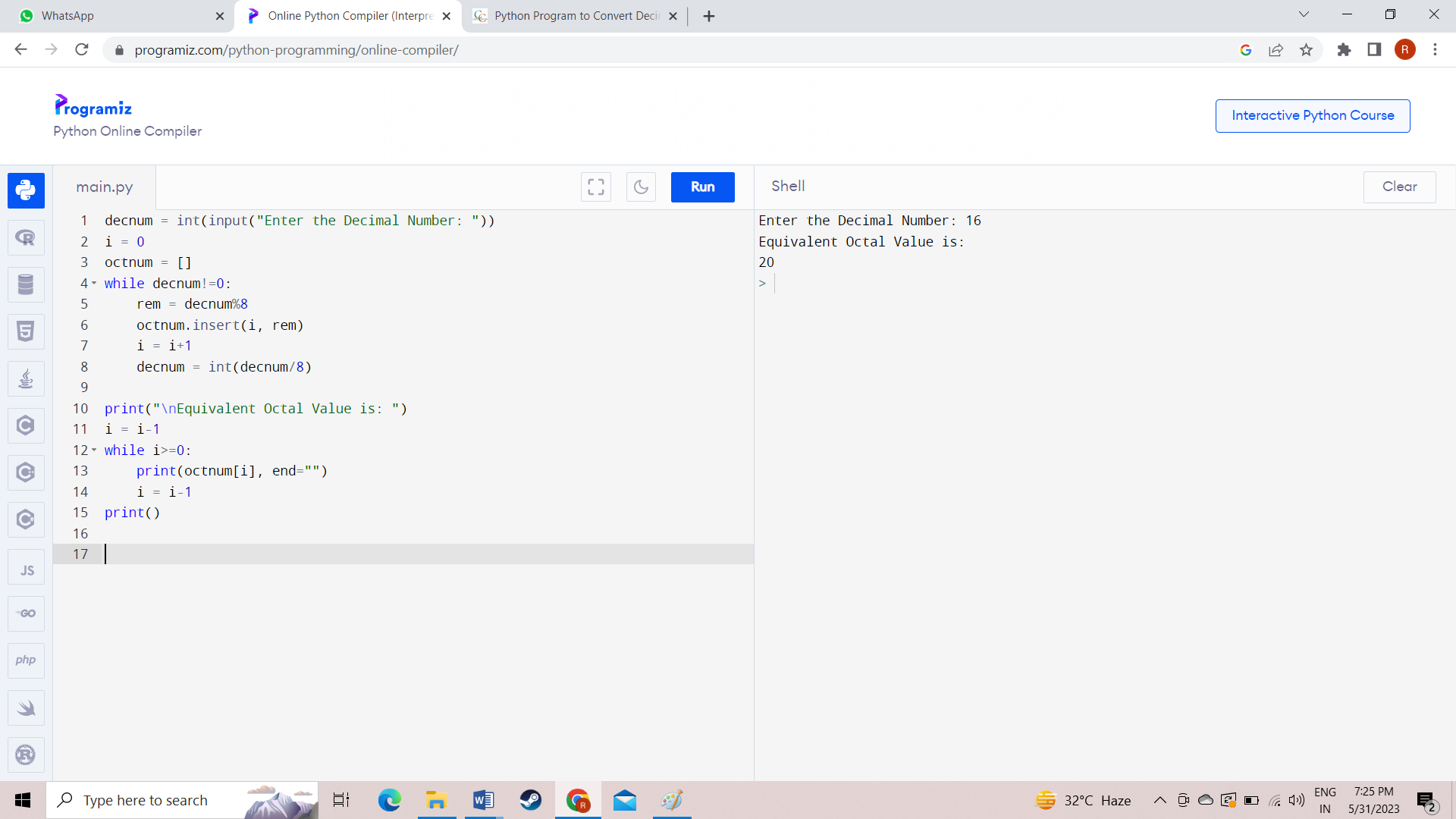
while i>=0:

print(octnum[i], end="")

i = i-1

print()

**OUTPUT:**

****

**RESULT:**

The given program to convert decimal to octal numbers is successfully completed

1. **DECIMAL TO HEXADECIMAL CONVERTION**

**AIM:**

To write a python program to convert decimal to hexadecimal numbers

**PROGRAM:**

value = 10

dechimal\_nums = [value]

def dechimal\_to\_Hex(dechimal\_nums):

digits = "0123456789ABCDEF"

x = (dechimal\_nums % 16)

rest\_part = dechimal\_nums // 16

if (rest\_part == 0):

return digits[x]

return dechimal\_to\_Hex(rest\_part) + digits[x]

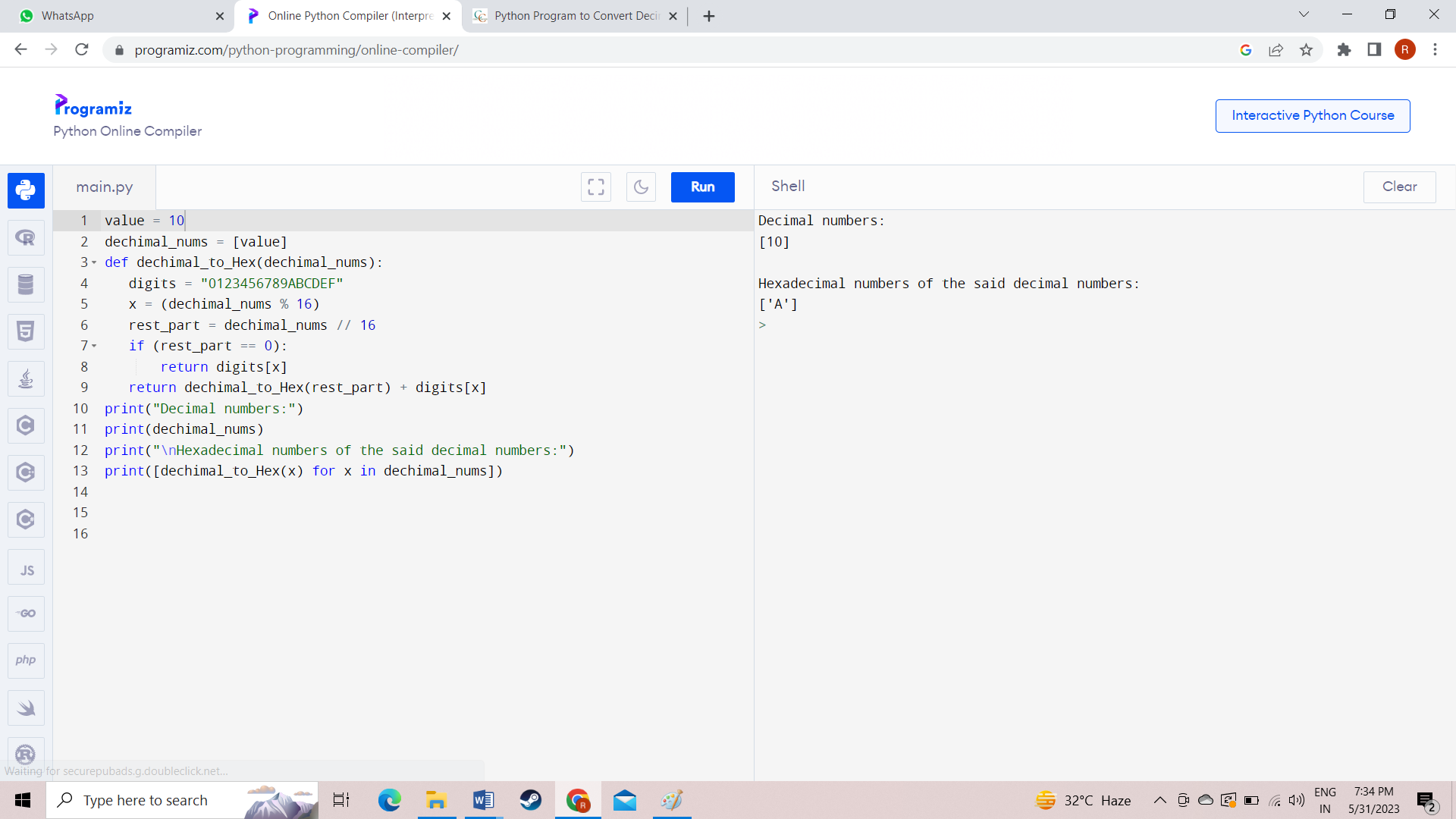
print("Decimal numbers:")

print(dechimal\_nums)

print("\nHexadecimal numbers of the said decimal numbers:")

print([dechimal\_to\_Hex(x) for x in dechimal\_nums])

**OUTPUT:**

****

**RELUST:**

The given program to convert decimal to hexadecimal numbers is successfully completed

1. **OCTAL TO BINARY CONVERTION**

**AIM:**

To write a python program to convert octal to binary numbers.

**PROGRAM:**

octnum = int(input("Enter the Octal Number: "))

rev = 0

chk = 0

while octnum!=0:

rem = octnum%10

if rem>7:

chk = 1

break

rev = rem + (rev\*10)

octnum = int(octnum/10)

if chk == 0:

octnum = rev

binnum = ""

while octnum!=0:

rem = octnum%10

if rem==0:

binnum = binnum + "000"

elif rem==1:

binnum = binnum + "001"

elif rem==2:

binnum = binnum + "010"

elif rem==3:

binnum = binnum + "011"

elif rem==4:

binnum = binnum + "100"

elif rem==5:

binnum = binnum + "101"

elif rem==6:

binnum = binnum + "110"

elif rem==7:

binnum = binnum + "111"

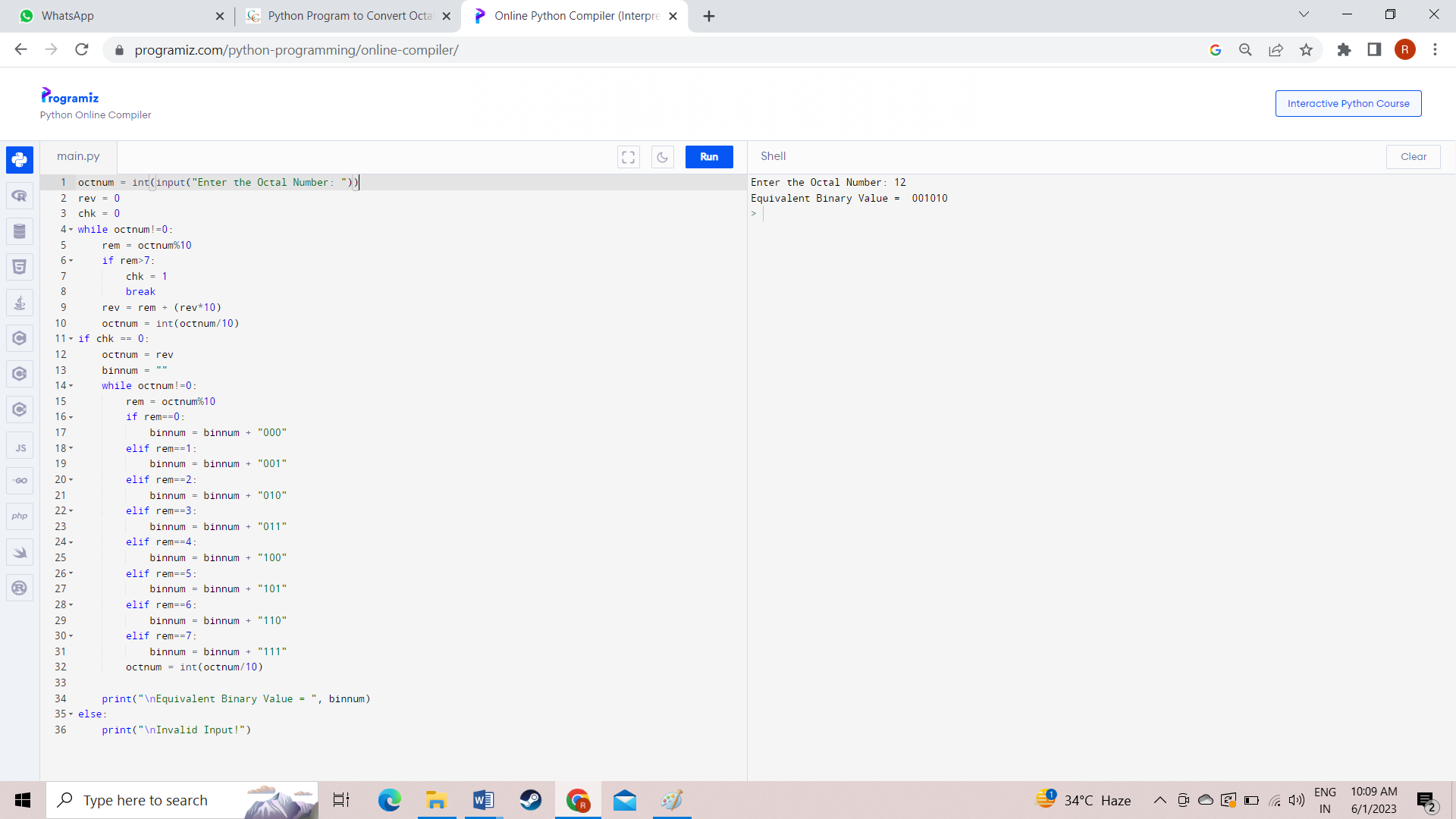
octnum = int(octnum/10)

print("\nEquivalent Binary Value = ", binnum)

else:

print("\nInvalid Input!")

**OUTPUT:**

****

**RESULT:**

The given program to convert octal to binary numbers is successfully completed

1. **OCTAL TO DECIMAL CONVERTION**

**AIM:**

To write a python program to convert octal to decimal numbers.

**PROGRAM:**

octnum = int(input("Enter the Octal Number: "))

chk = 0

i = 0

decnum = 0

while octnum!=0:

rem = octnum%10

if rem>7:

chk = 1

break

decnum = decnum + (rem \* (8 \*\* i))

i = i+1

octnum = int(octnum/10)

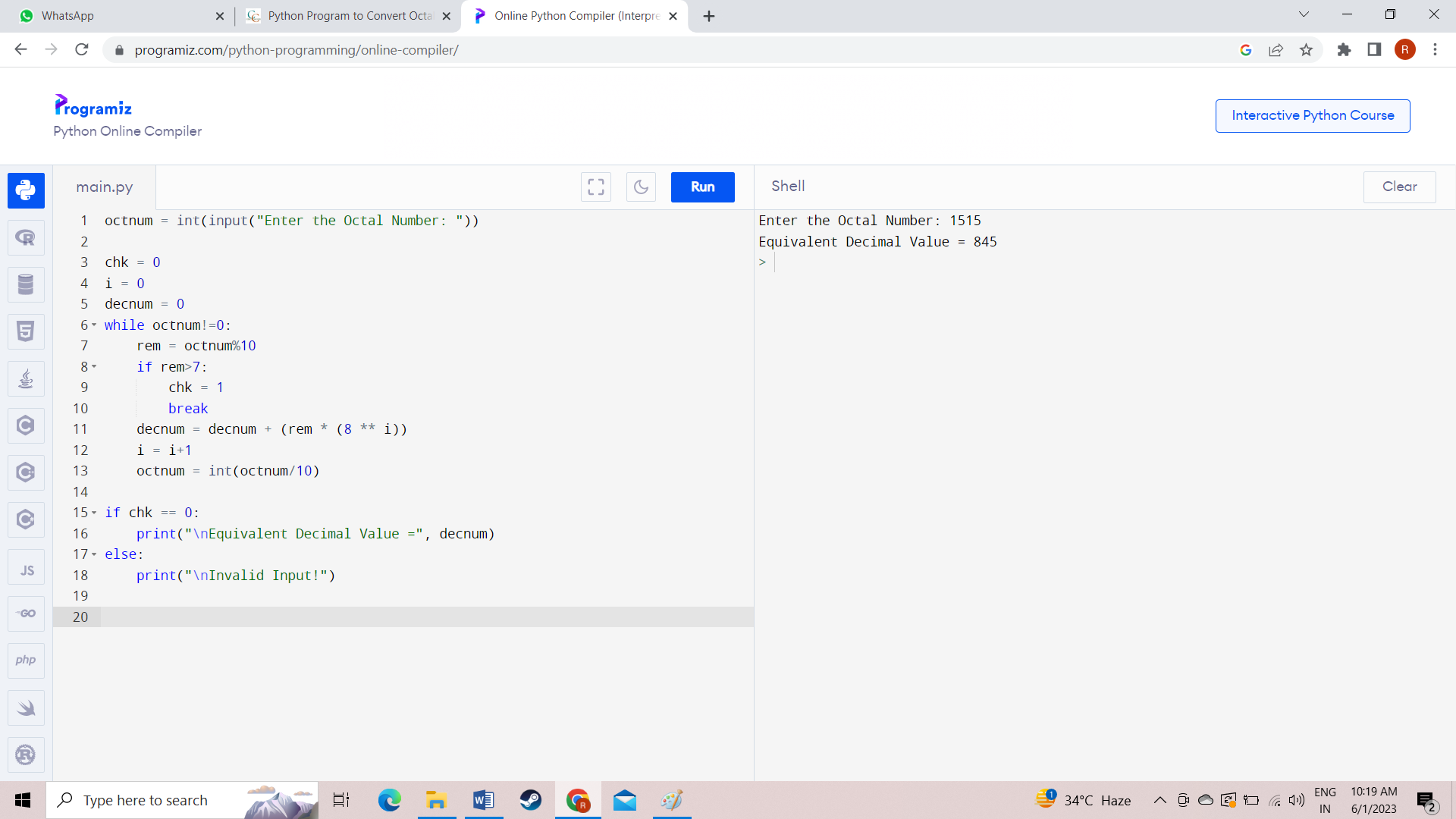
if chk == 0:

print("\nEquivalent Decimal Value =", decnum)

else:

print("\nInvalid Input!")

**OUTPUT:**

****

**RESULT:**

The given program to convert octal to decimal numbers is successfully completed

1. **OCTAL TO HEXADECIMAL CONVERTION**

**AIM:**

To write a python program to convert octal to hexadecimal numbers

**PROGRAM:**

octnum = int(input("Enter the Octal Number: "))

chk = 0

i = 0

decnum = 0

while octnum!=0:

rem = octnum%10

if rem>7:

chk = 1

break

decnum = decnum + (rem \* (8 \*\* i))

i = i+1

octnum = int(octnum/10)

if chk == 0:

print("\nEquivalent Decimal Value =", decnum)

else:

print("\nInvalid Input!")

value = decnum

dechimal\_nums = [value]

def dechimal\_to\_Hex(dechimal\_nums):

digits = "0123456789ABCDEF"

x = (dechimal\_nums % 16)

rest\_part = dechimal\_nums // 16

if (rest\_part == 0):

return digits[x]

return dechimal\_to\_Hex(rest\_part) + digits[x]

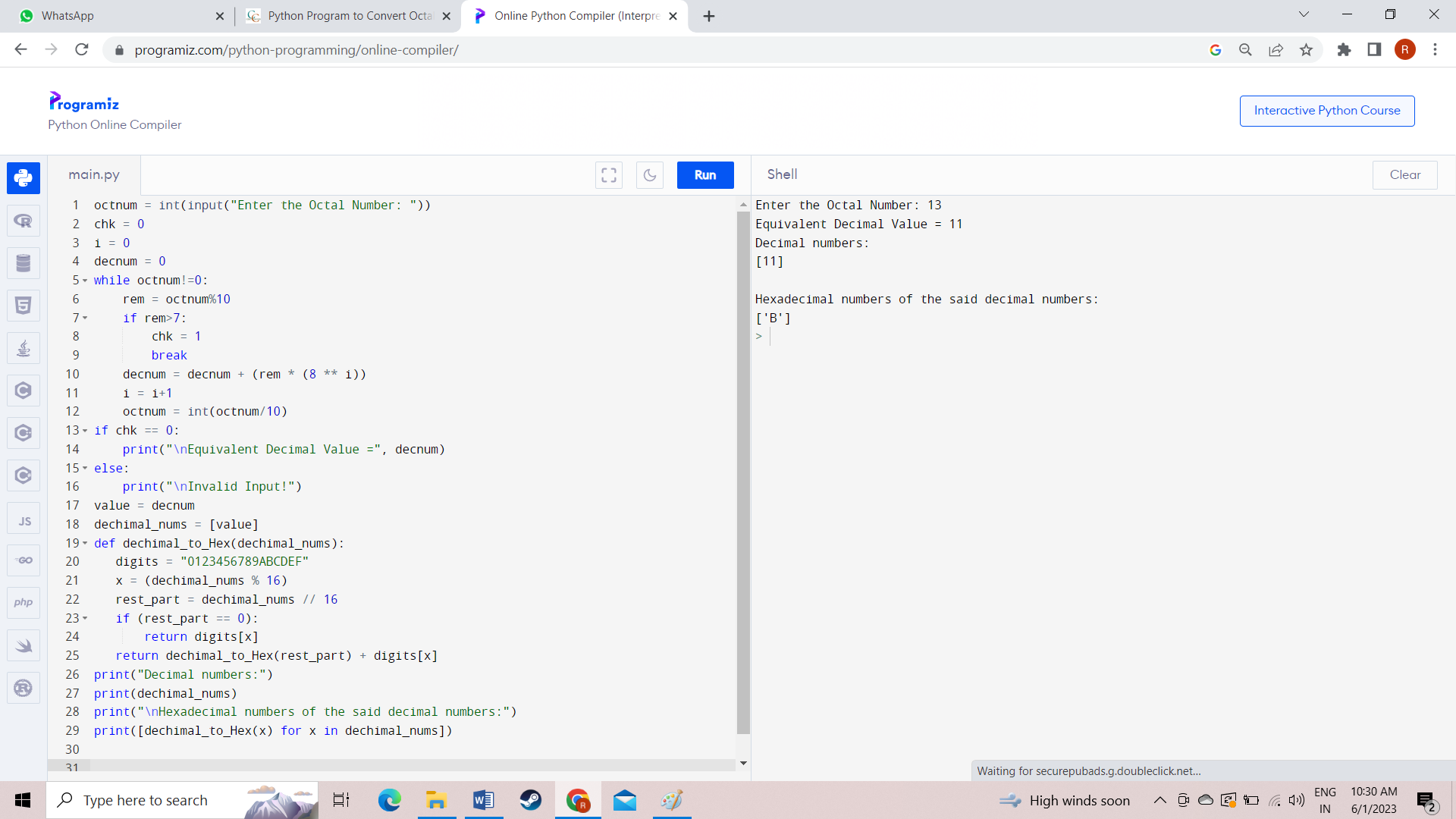
print("Decimal numbers:")

print(dechimal\_nums)

print("\nHexadecimal numbers of the said decimal numbers:")

print([dechimal\_to\_Hex(x) for x in dechimal\_nums])

**OUTPUT:**



**RESULT:**

The given program to convert octal to hexadecimal numbers is successfully completed.

1. **8** - **BIT ADDITION**

**AIM:**

To write an assembly level program to add 8-bit numbers.

**PROGRAM:**

start: nop

LDA 8000

MOV H,A

LDA 8001

ADD H

MOV L,A

MVI A,00H

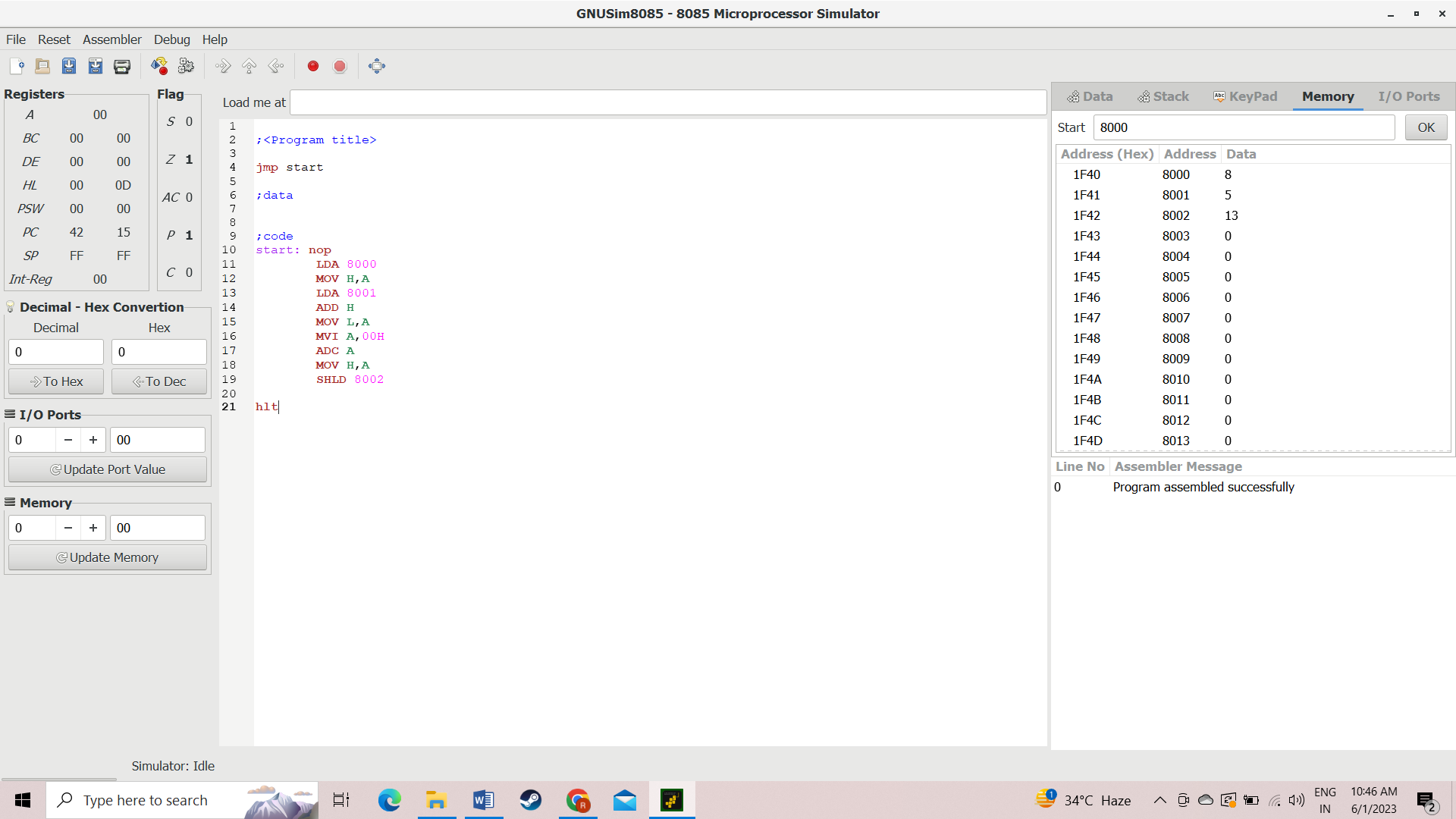
ADC A

MOV H,A

SHLD 8002

hlt

**OUTPUT:**



**RESULT:**

The given program to add two 8-bit numbers using assembly level program is completed successfully.

1. **8** - **BIT SUBTRACTION**

**AIM:**

To write an assembly level program to subtract two 8-bit numbers.

**PROGRAM:**

start: nop

LDA 8001

MOV H,A

LDA 8000

SUB H

MOV L,A

MVI A,00H

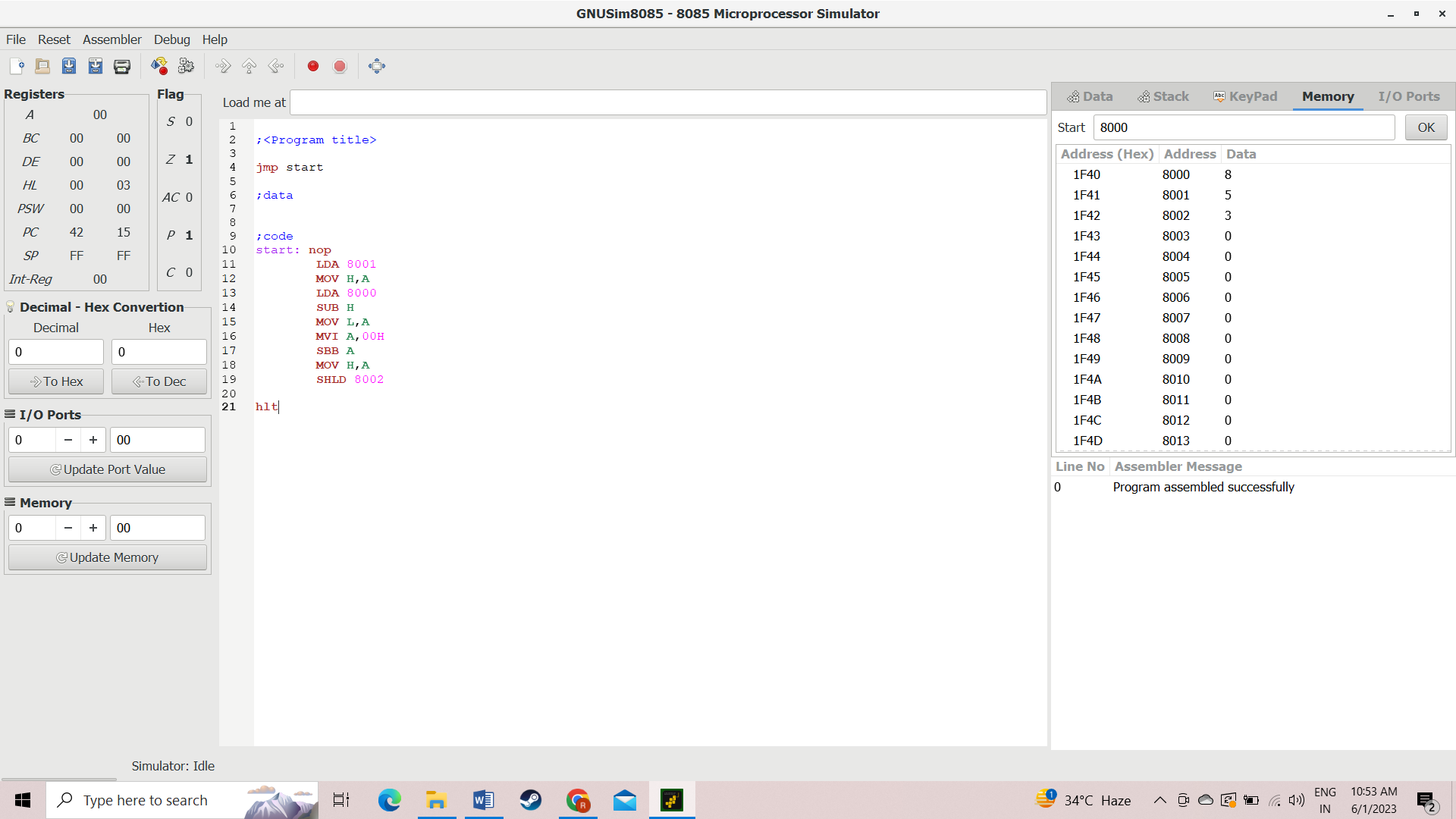
SBB A

MOV H,A

SHLD 8002

Hlt

**OUTPUT:**



**RESULT:**

The given program to add two 8-bit numbers using assembly level program is completed successfully.

1. **8** - **BIT MULTIPICATION**

**AIM:**

To write an assembly level program to multiply 8-bit numbers.

**PROGRAM:**

start: nop

LDA 8000

MOV B,A

LDA 8001

MOV C,A

MVI A,00H

LOOP: ADD B

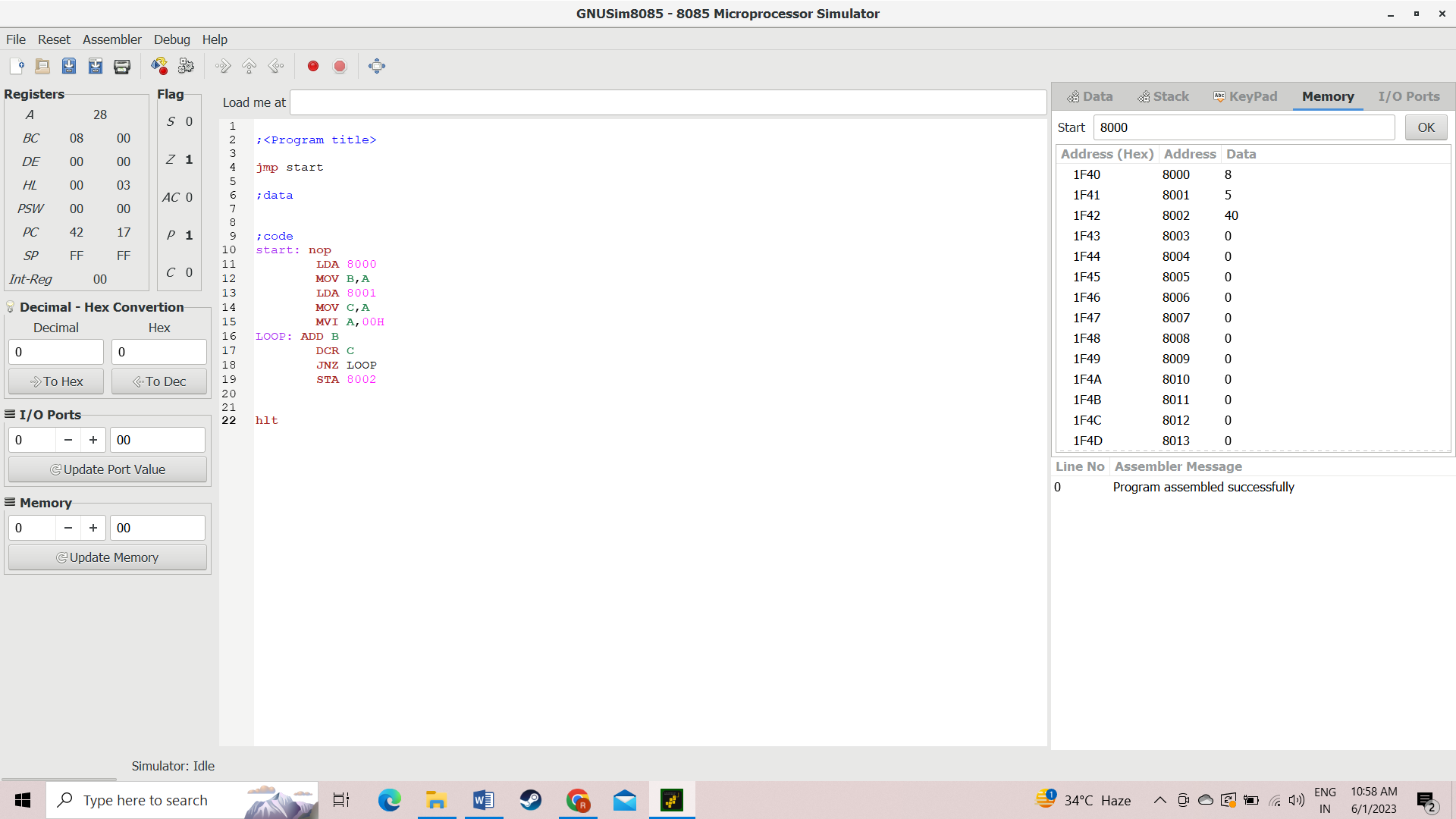
DCR C

JNZ LOOP

STA 8002

Hlt

**OUTPUT:**



**RESUT:**

The given program to multiply two 8-bit numbers using assembly level program is completed successfully.

1. **8** - **BIT DIVISION**

**AIM:**

To write an assembly-level program to divide 8-bit numbers.

**PROGRAM:**

start: nop

LDA 8000

MOV B,A

LDA 8001

MVI C,00H

LOOP2: CMP B

JC LOOP1

SUB B

INR C

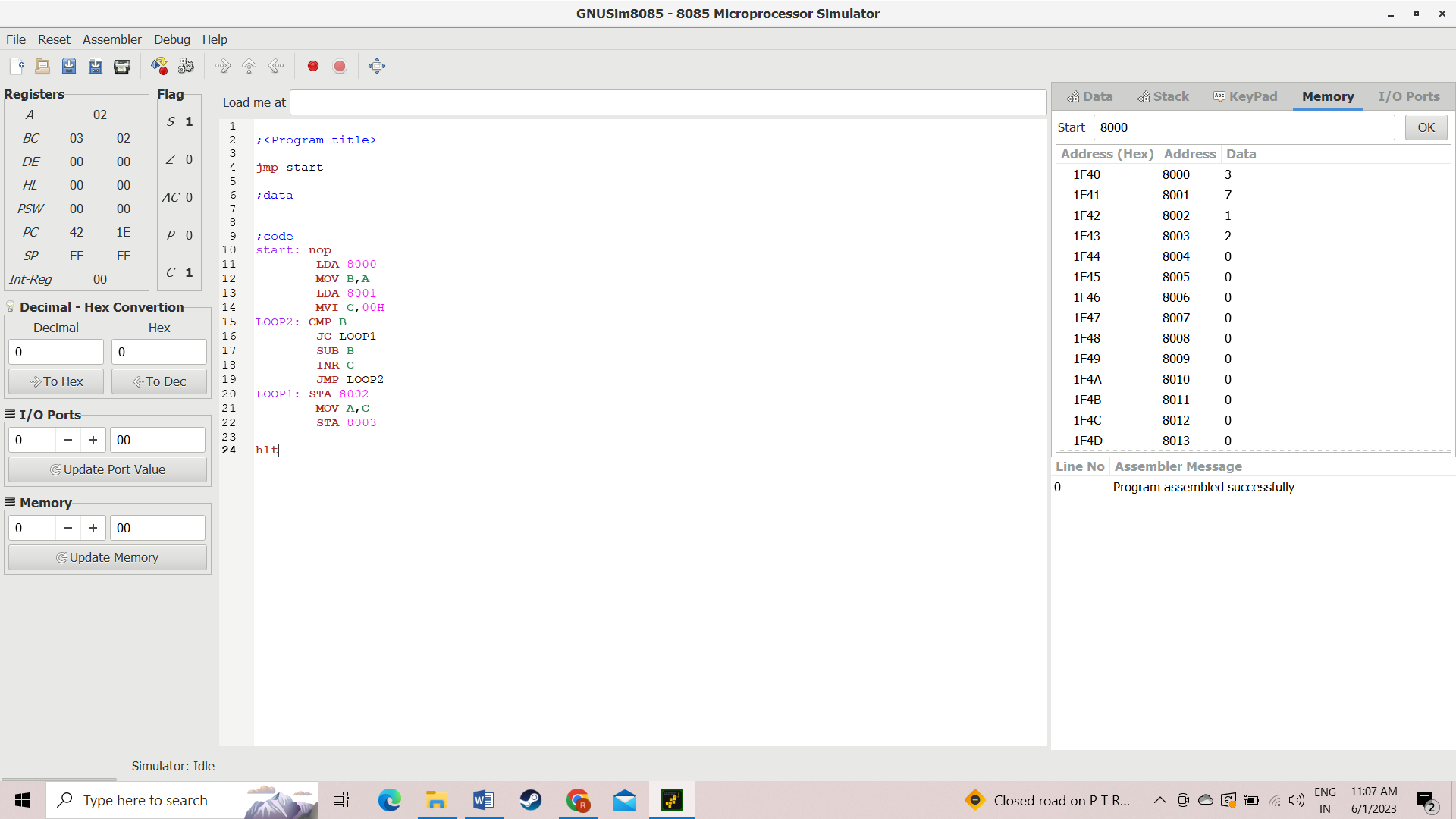
JMP LOOP2

LOOP1: STA 8002

MOV A,C

STA 8003

**OUTPUT:**



**RESULT:**

The given program to divide two 8-bit numbers using an assembly-level program is completed successfully.

1. **16** - **BIT ADDITION**

**AIM:**

To write an assembly-level program to add 16-bit numbers.

**PROGRAM:**

start: nop

LHLD 3001H

XCHG

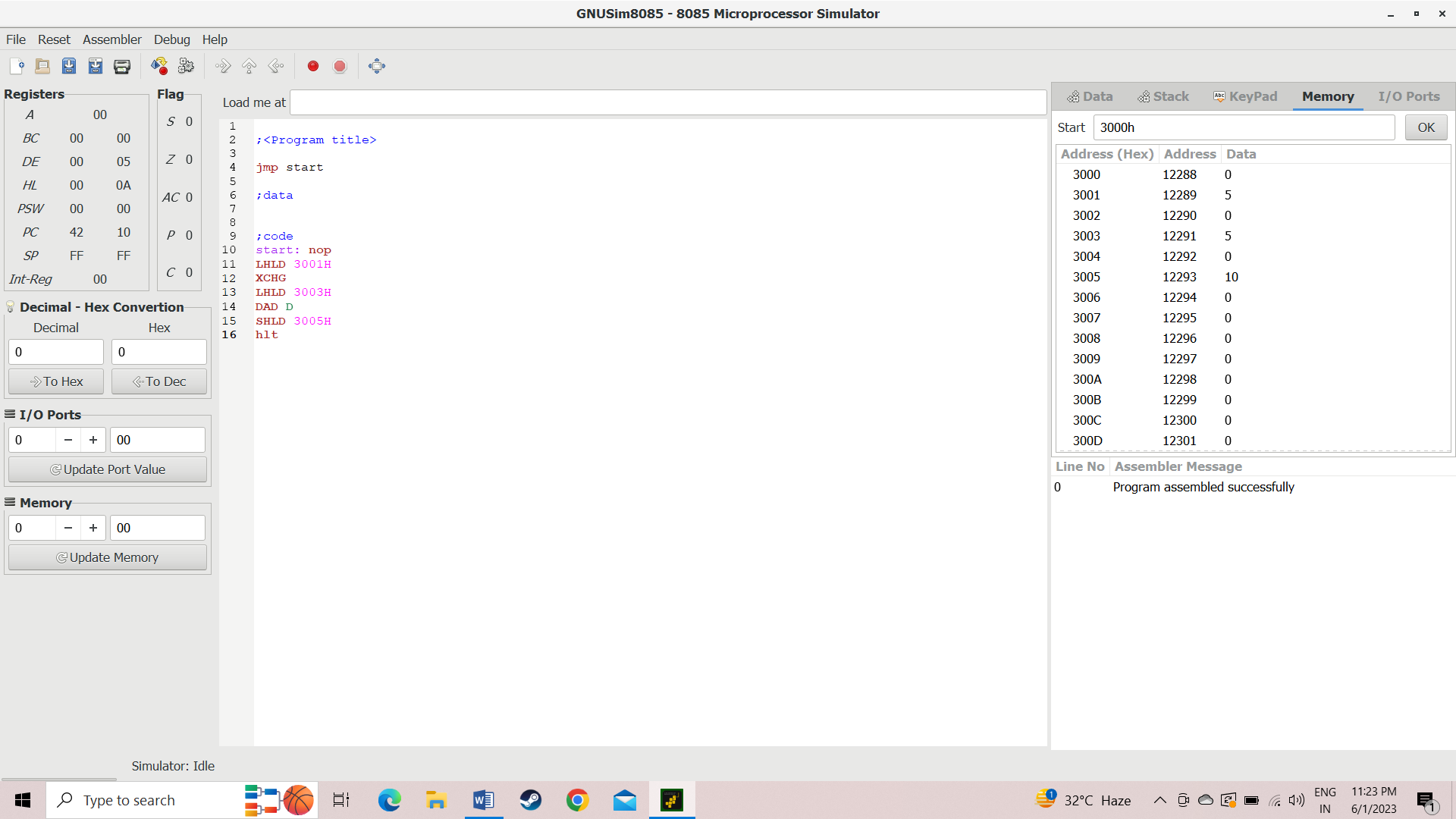
LHLD 3003H

DAD D

SHLD 3005H

Hlt

**OUTPUT:**



**RESULT:**

The given program to add two 16-bit numbers using an assembly-level program is completed successfully.

1. **16** - **BIT ADDITION USING TWO 8-BIT NUMBERS**

**AIM:**

To write an assembly-level program to add 16-bit numbers using two 8-bit numbers.

**PROGRAM:**

start: nop

LHLD 3001

XCHG

LHLD 3003

MOV A,E

ADD L

MOV L,A

MOV A,D

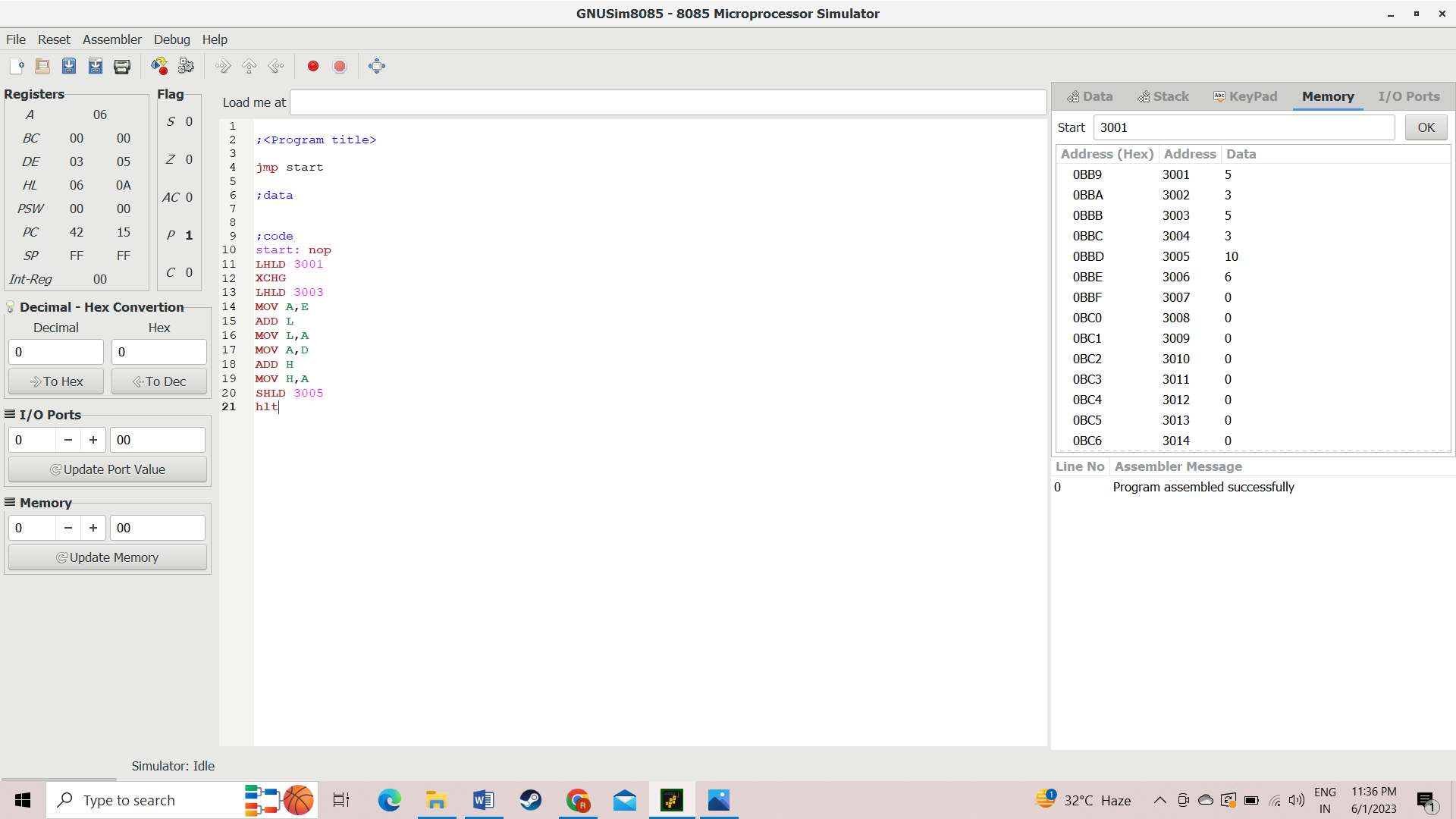
ADD H

MOV H,A

SHLD 3005

Hlt

**OUTPUT:**



**RESULT:**

The given program to add 16-bit numbers by two 8-bit numbers using assembly level program is completed successfully.

1. **16** - **BIT SUBTRACTION**

**AIM:**

To write an assembly level program to subtract 16-bit numbers.

**PROGRAM:**

start: nop

LHLD 3001

XCHG

LHLD 3003

MOV A,E

SUB L

MOV L,A

MOV A,D

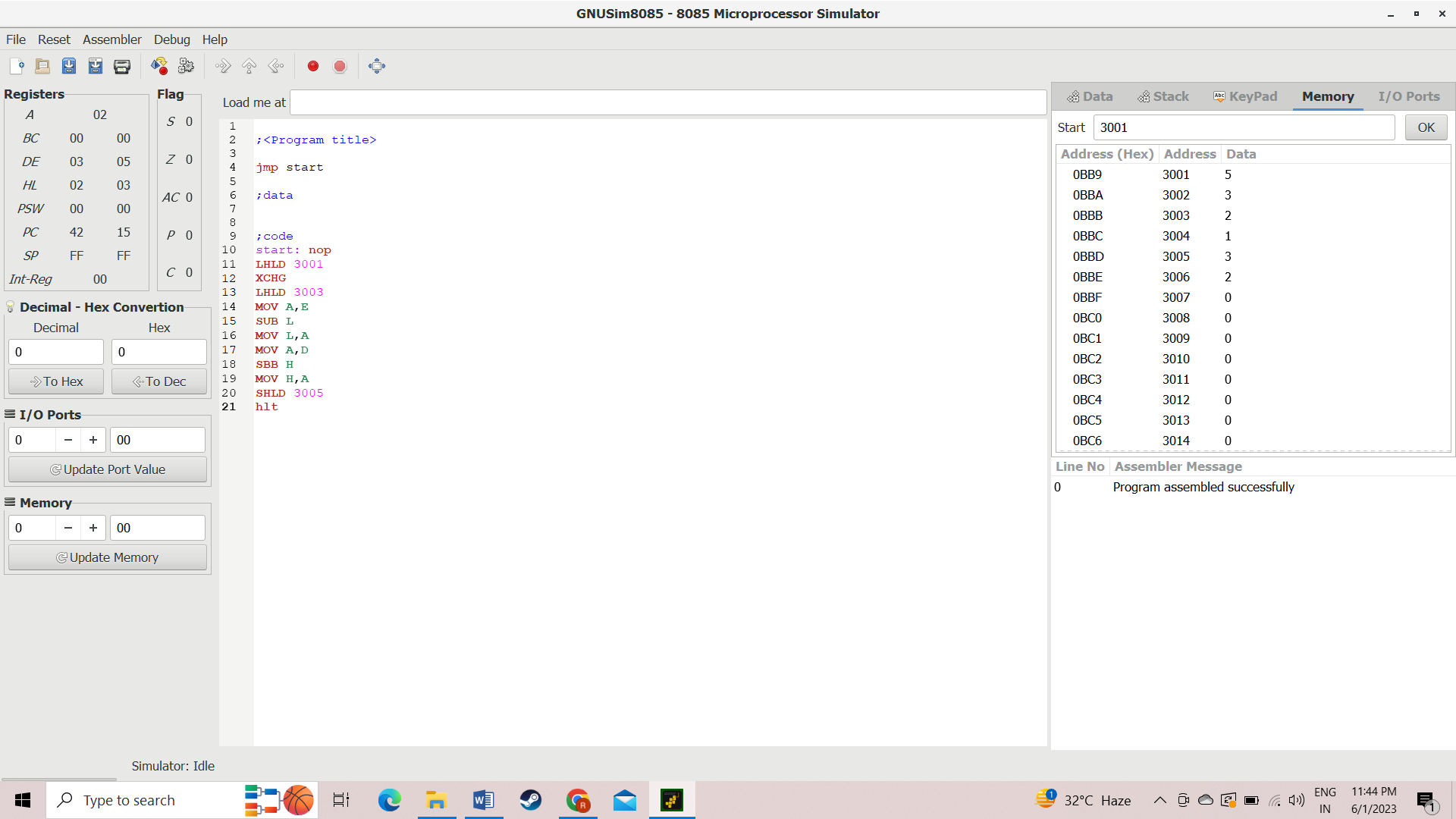
SBB H

MOV H,A

SHLD 3005

Hlt

**OUTPUT:**



**RESULT:**

The given program to subtract two 16-bit numbers using assembly level program is completed successfully.

1. **ASCENDING ORDER**

**AIM:**

To write an assembly level program to make ascending order.

**PROGRAM:**

start: nop

LXI H,8000

LOOP: MOV A,M

INX H

CMP M

JC SKIP

MOV B,M

MOV M,A

DCX H

MOV M,B

INX H

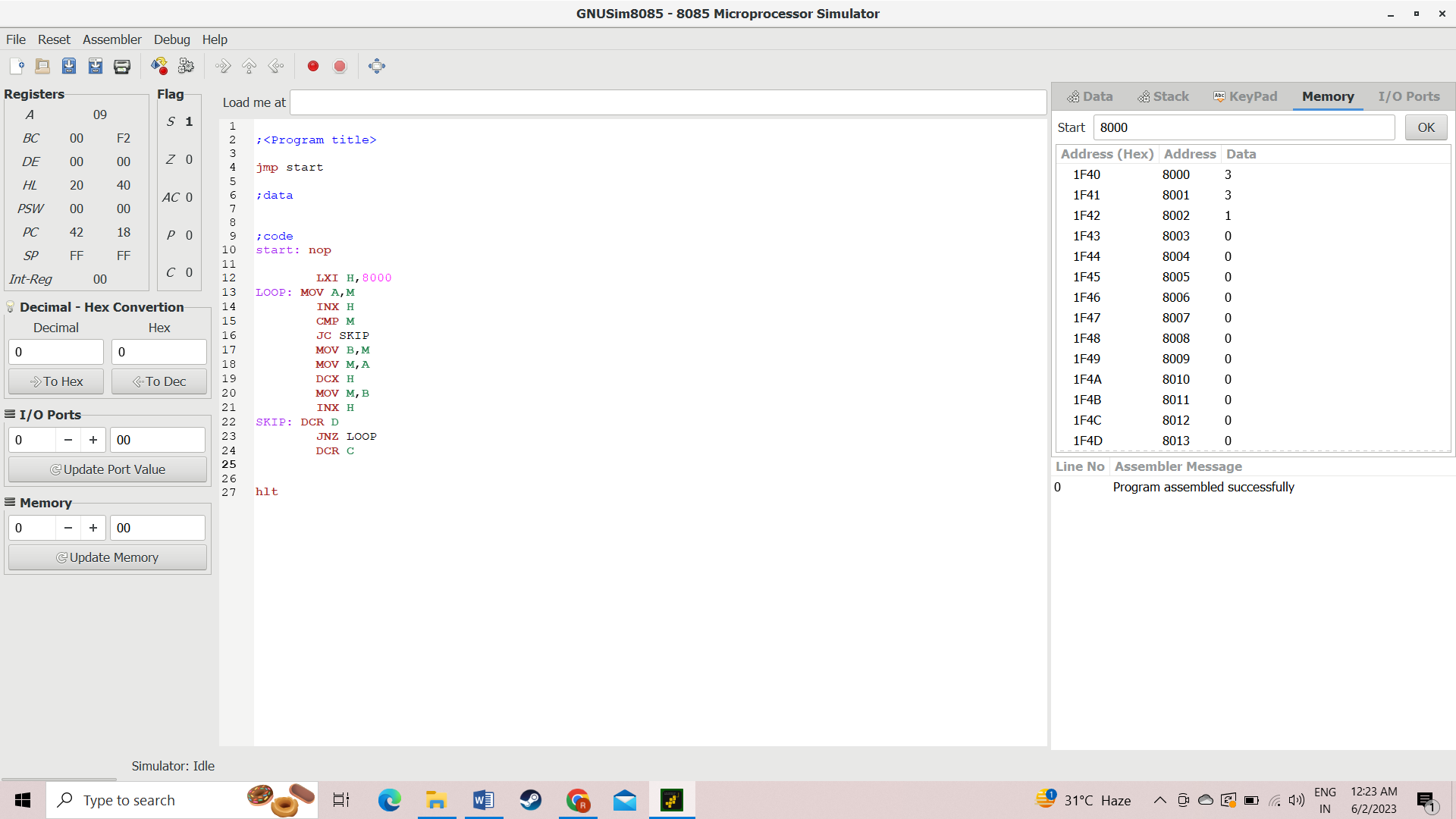
SKIP: DCR D

JNZ LOOP

DCR C

Hlt

**OUTPUT:**



**RESULT:**

The given program to make Ascending order using assembly level program is completed successfully.

1. **DESCENDING ORDER**

**AIM:**

To write an assembly level program to make descending order.

**PROGRAM:**

start: nop

LXI H,8000

LOOP: MOV A,M

INX H

CMP M

JNC SKIP

MOV B,M

MOV M,A

DCX H

MOV M,B

INX H

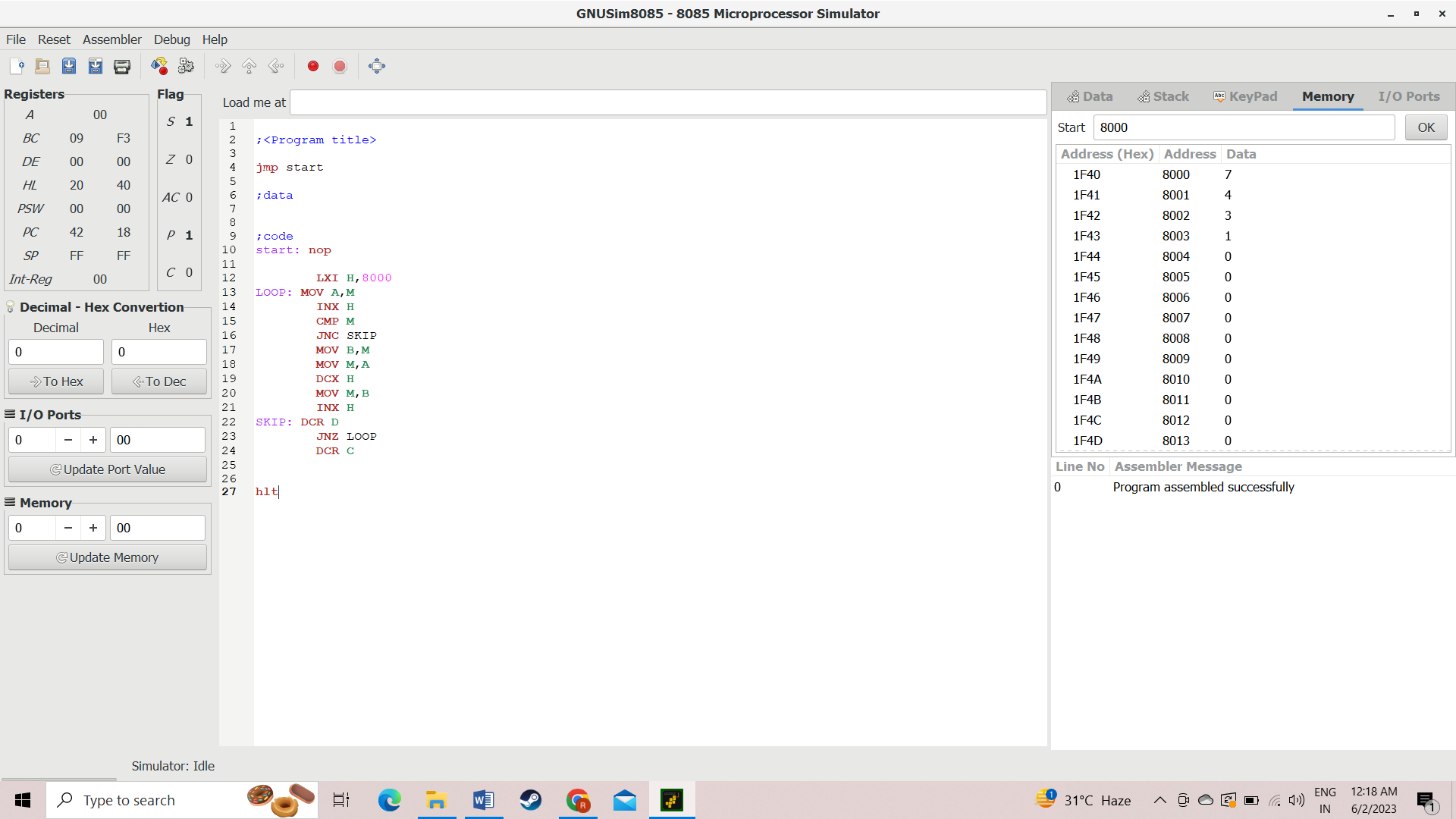
SKIP: DCR D

JNZ LOOP

DCR C

Hlt

**OUTPUT:**



**RESULT:**

The given program to make descending order using an assembly-level program is completed successfully.

1. **FIBONACCI SERIES**

**AIM:**

To write an assembly-level program to do the Fibonacci series.

**PROGRAM:**

start: nop

LXI H,8000

MOV C,M

INX H

LOOP: MOV A,M

INX H

MOV B,M

ADD B

INX H

MOV M,A

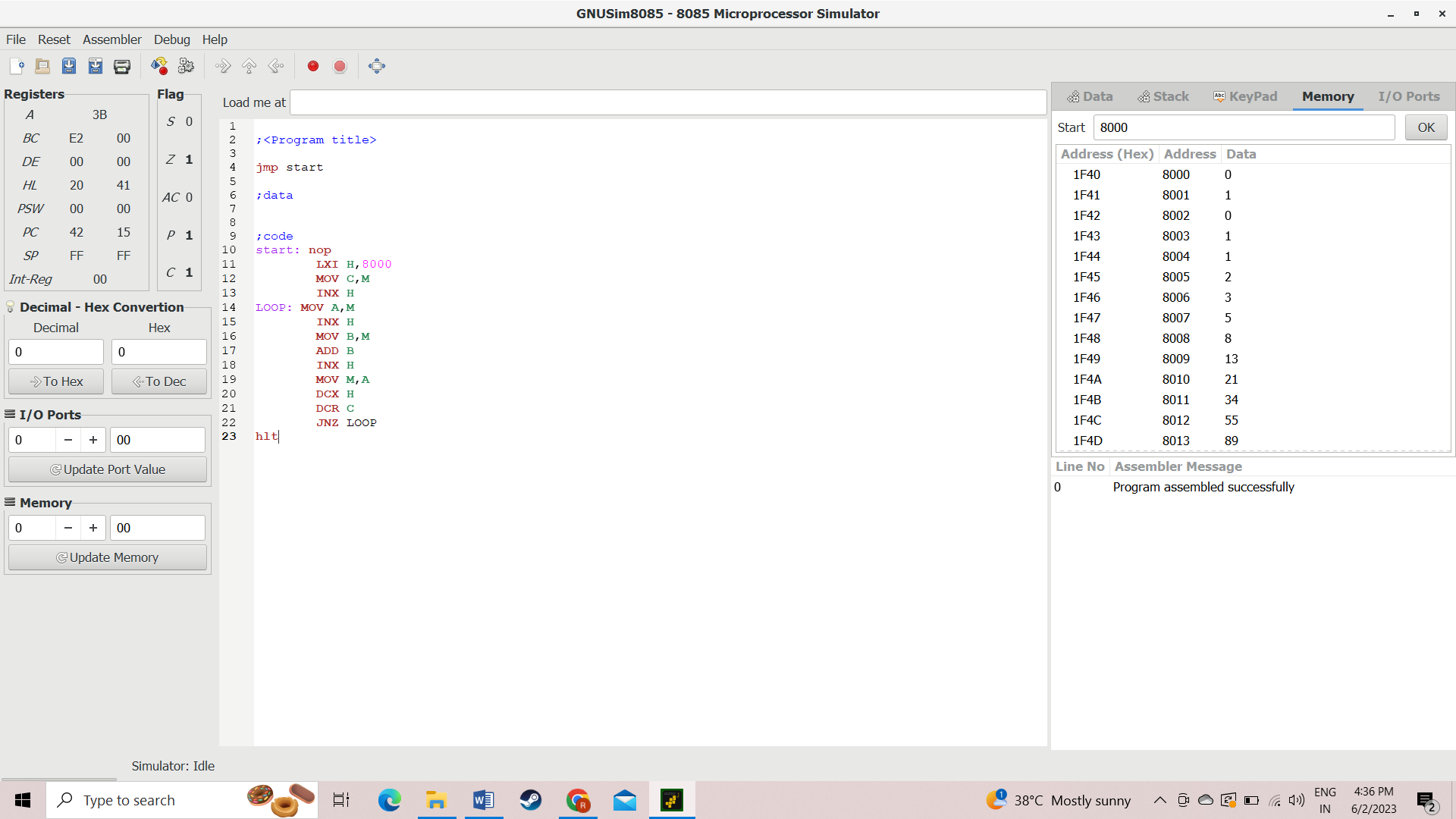
DCX H

DCR C

JNZ LOOP

hlt

**OUTPUT:**



**RESULT:**

The given program to do Fibonacci series using assembly level program is completed successfully.

1. **FIND THE LARGEST NUMBER IN ARRAY**

**AIM:**

To write an assembly level program to find the largest number.

**PROGRAM:**

            start: nop

LXI H,8000

MOV C,M

LXI H,8001

MOV B,M

DCR C

LOOP: INX H

MOV A,M

CMP B

JC LOOP2

MOV B,A

LOOP2: DCR C

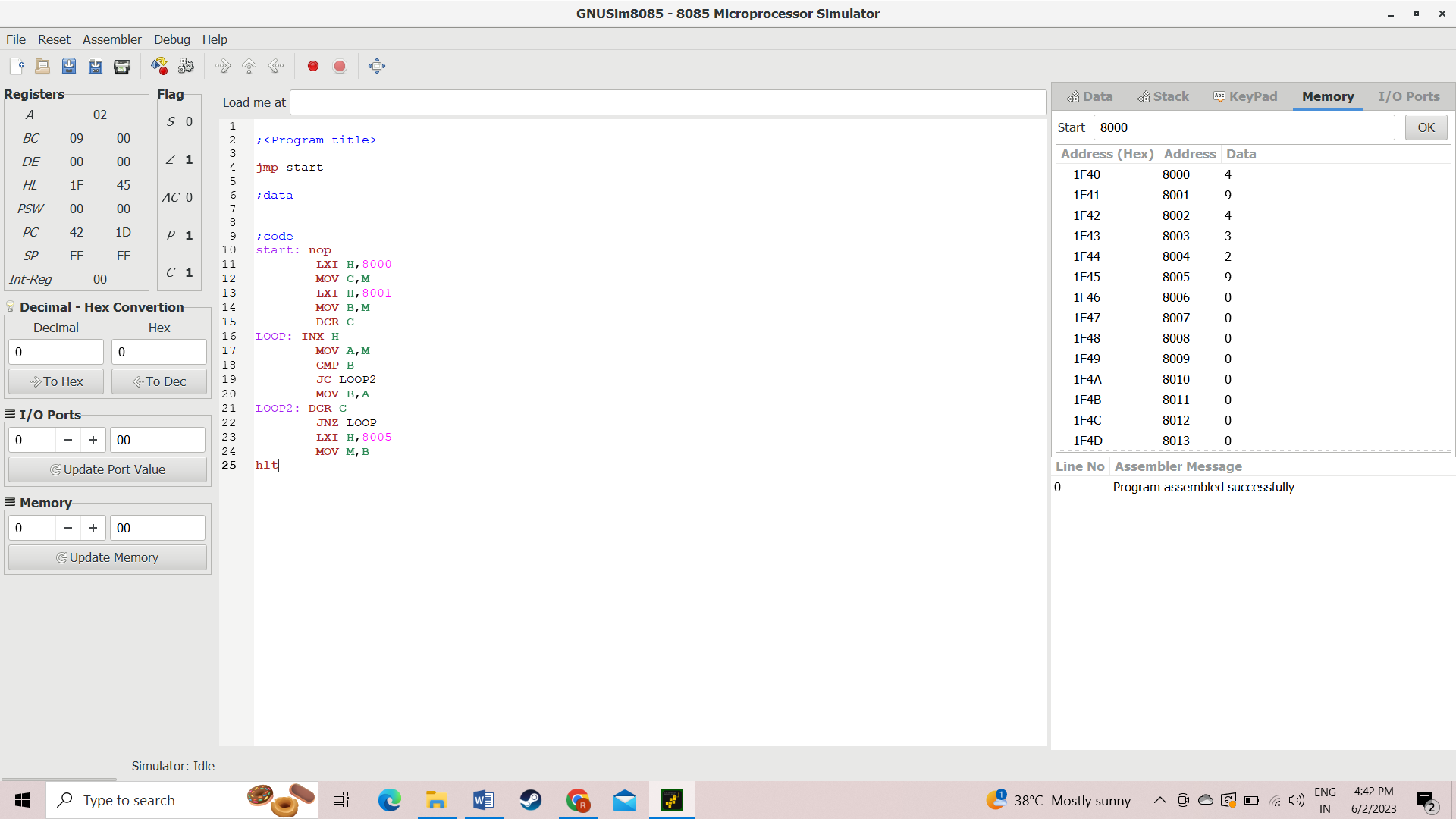
JNZ LOOP

LXI H,8005

MOV M,B

hlt

**OUTPUT:**



**RESULT:**

The given program to find the largest number using an assembly level program is completed successfully.

1. **FIND THE SMALLEST NUMBER**

**AIM:**

To write an assembly-level program to find the largest number.

**PROGRAM:**

            start: nop

LXI H,8000

MOV C,M

LXI H,8001

MOV B,M

DCR C

LOOP: INX H

MOV A,M

CMP B

JNC LOOP2

MOV B,A

LOOP2: DCR C

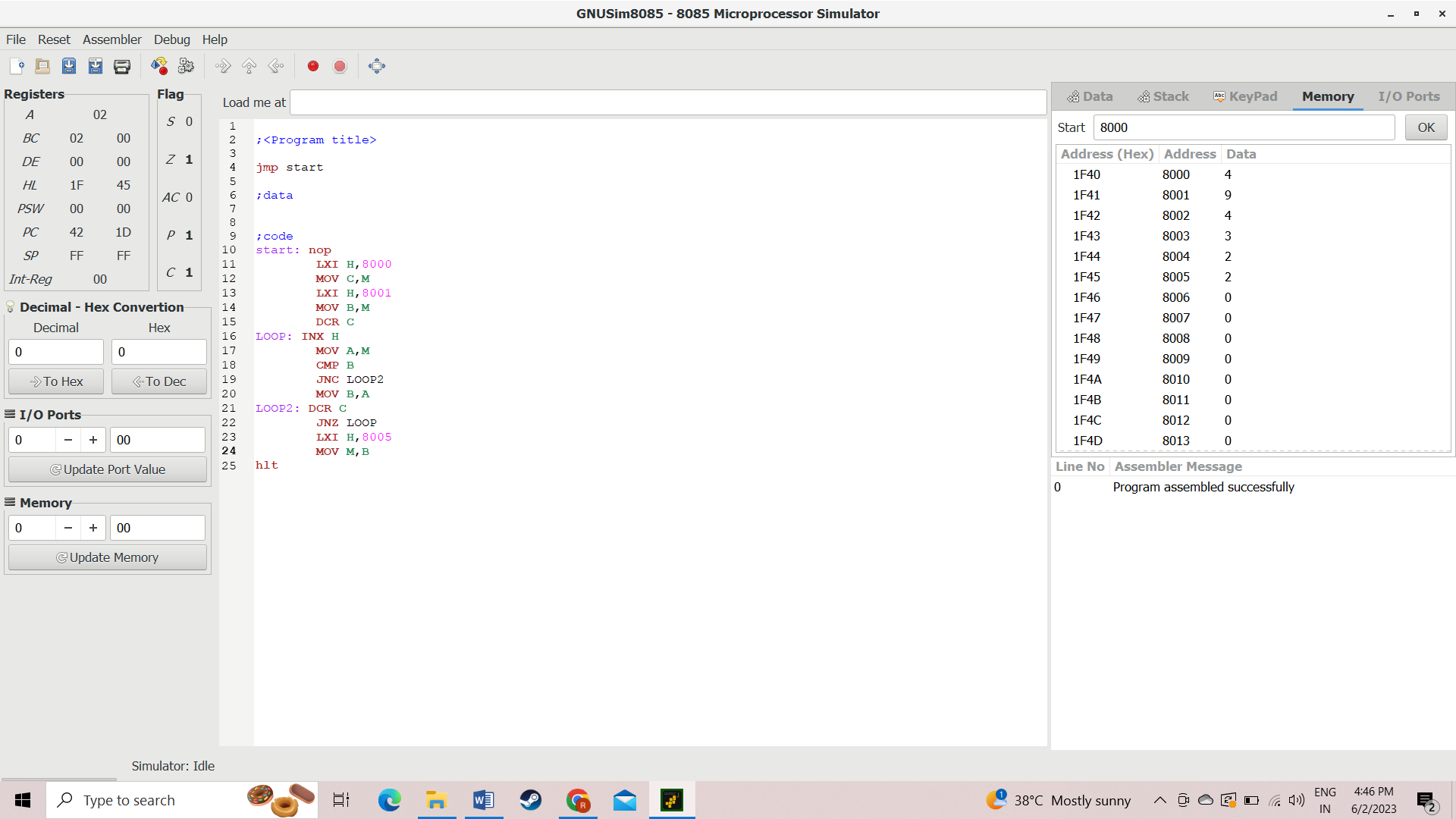
JNZ LOOP

LXI H,8005

MOV M,B

hlt

**OUTPUT:**



**RESULT:**

The given program to find the largest number using assembly level program is completed successfully.

1. **FIND THE SQUARE OF NUMBERS**

**AIM:**

To write an assembly level program to find the square of number.

**PROGRAM:**

start: nop

LDA 8000

MOV C, A

MOV B,A

MVI A,0

LOOP: ADD B

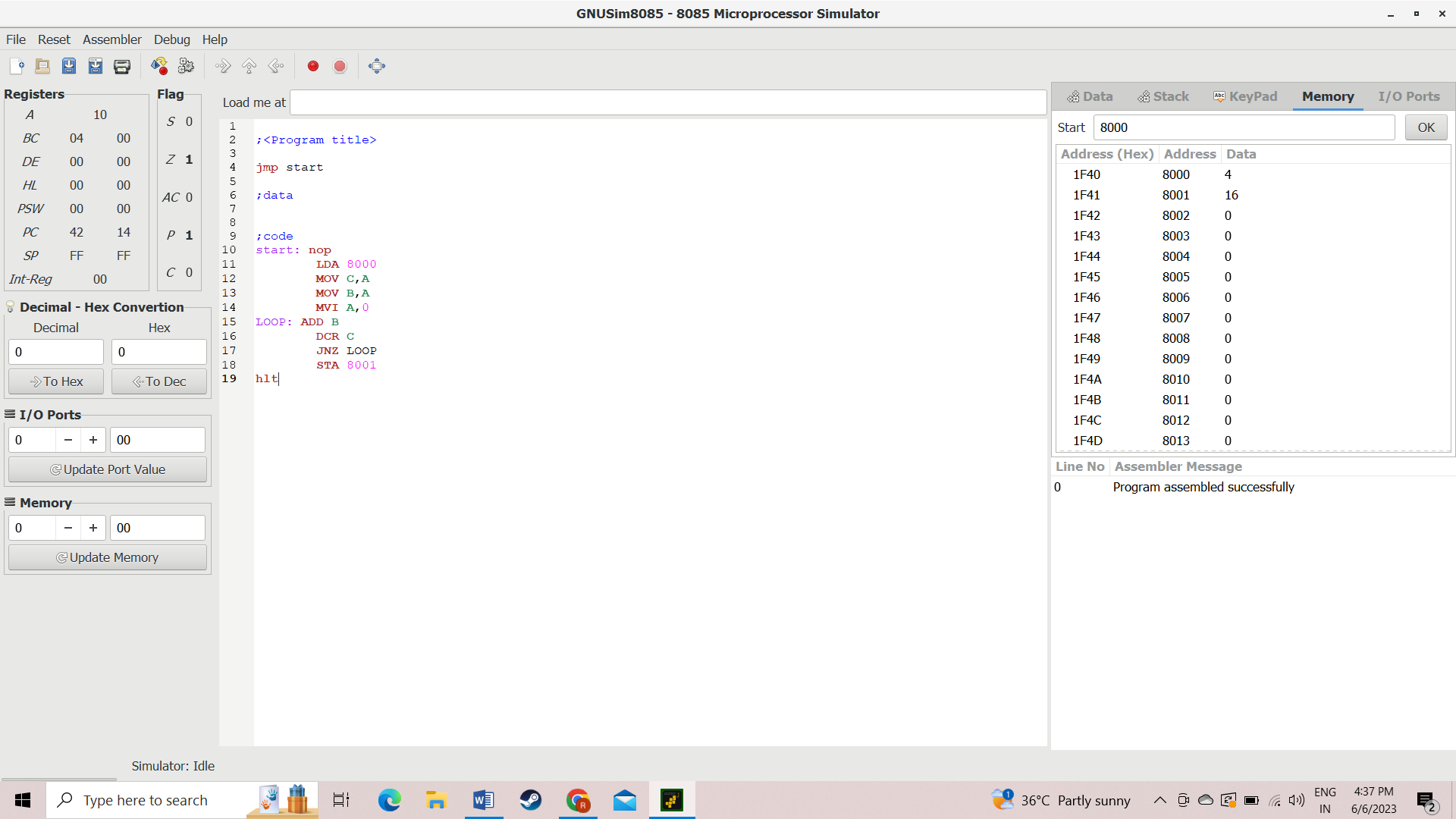
DCR C

JNZ LOOP

STA 8001

Hlt

**OUTPUT:**



**RESULT:**

The given program to find the square of number using assembly level program is completed successfully.

1. **FIND THE SUM OF NUMBERS**

**AIM:**

To write an assembly level program to find the sum of number.

**PROGRAM:**

start: nop

LDA 8000

MOV B,A

LXI H,8001

LOOP: MOV A,M

INX H

MOV D,M

ADD D

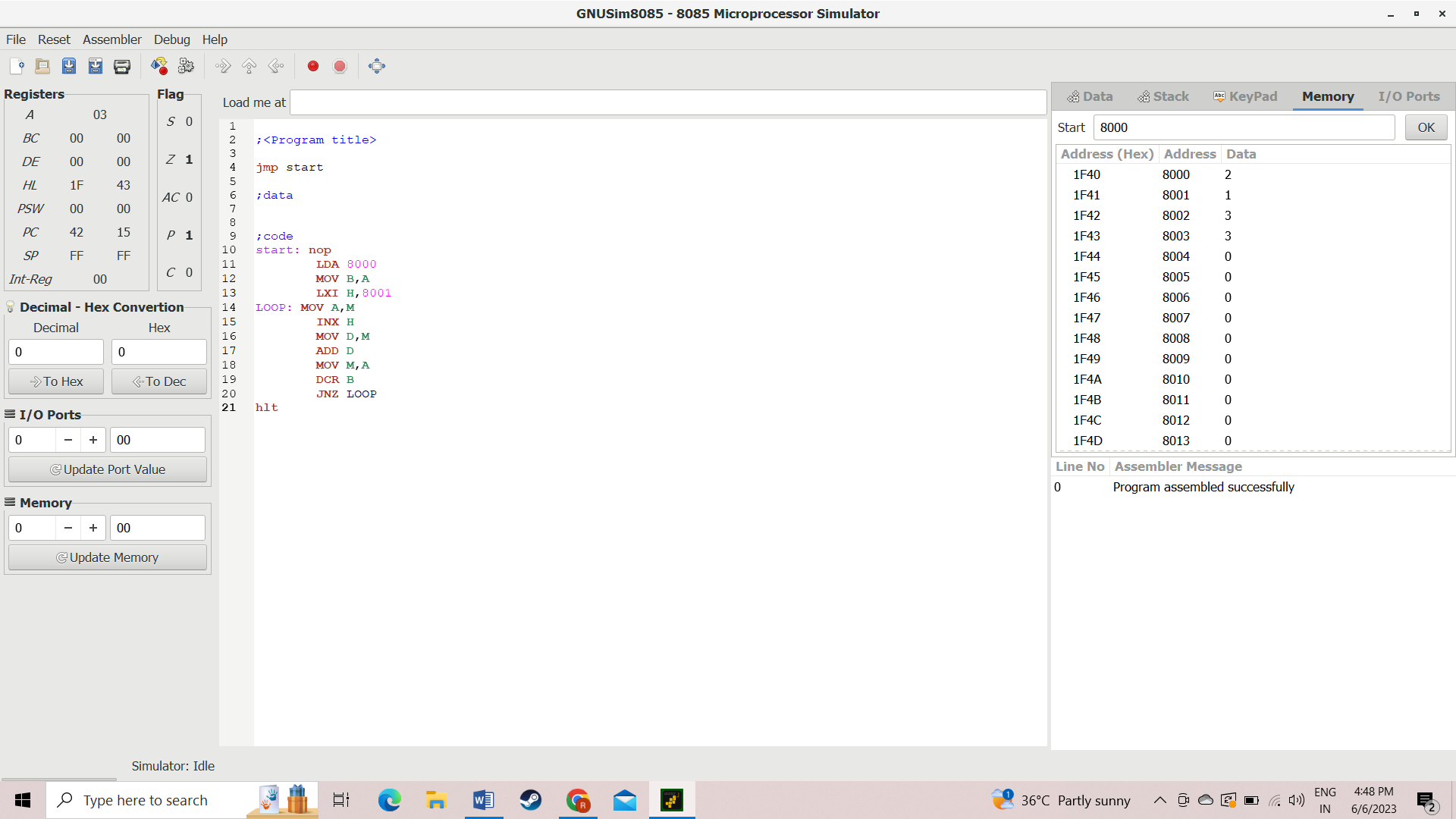
MOV M,A

DCR B

JNZ LOOP

Hlt

**OUTPUT:**



**RESULT:**

The given program to find the sum of number using assembly level program is completed successfully.

1. **TWO STAGE PIPELINING**

**AIM:**

To write a python program to achieve two stage pipelining.

**PROGRAM:**

counter=1

a=int(input("ENTER NUMBER-1-"))

counter=counter+1

b=int(input("ENTER NUMBER-2-"))

counter=counter+1

print("1-ADDITION 2-SUBTRACTION 3-MULTIPLICATION 4-DIVISION")

print("Enter Your Choice")

choice=int(input())

if choice==1:

print("Performing Addition...")

res=a+b

counter=counter+1

if choice==2:

print ("Performing Subtraction...")

res=a-b

counter=counter+1

if choice==3:

print ("Performing Multiplication")

res=a\*b

counter=counter+1

if choice==4:

if b==0:

print ("Denominator can't be Zero")

print ("Performing Division")

res=a/b

counter=counter+1

if choice>=5:

print ("Enter Correct Input")

print(res)

counter=counter+1

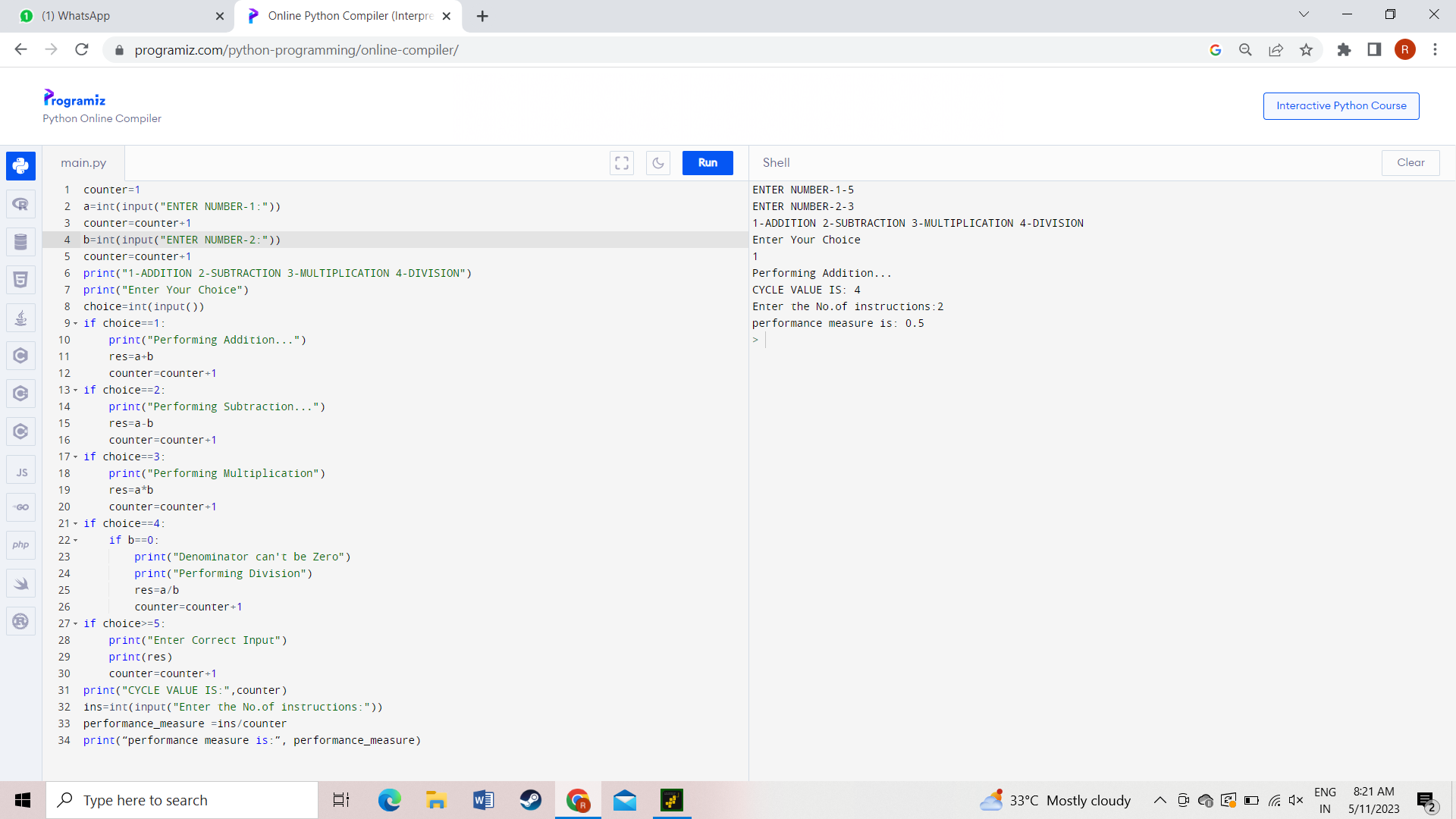
print ("CYCLE VALUE IS:",counter)

ins=int (input ("Enter the No.of instructions:"))

performance\_measure =ins/counter

print (“performance measure is:” performance\_measure)

**OUTPUT:**



**RESULT:**

The given program to achieve two stage pipelining using python program is completed successfully.

1. **THREE STAGE PIPELINING**

**AIM:**

To write a Python program to achieve three-stage pipelining.

**PROGRAM:**

counter=1

a=int(input("ENTER NUMBER-1-"))

counter=counter+1

b=int(input("ENTER NUMBER-2-"))

counter=counter+1

res= a and b

counter=counter+1

print(res)

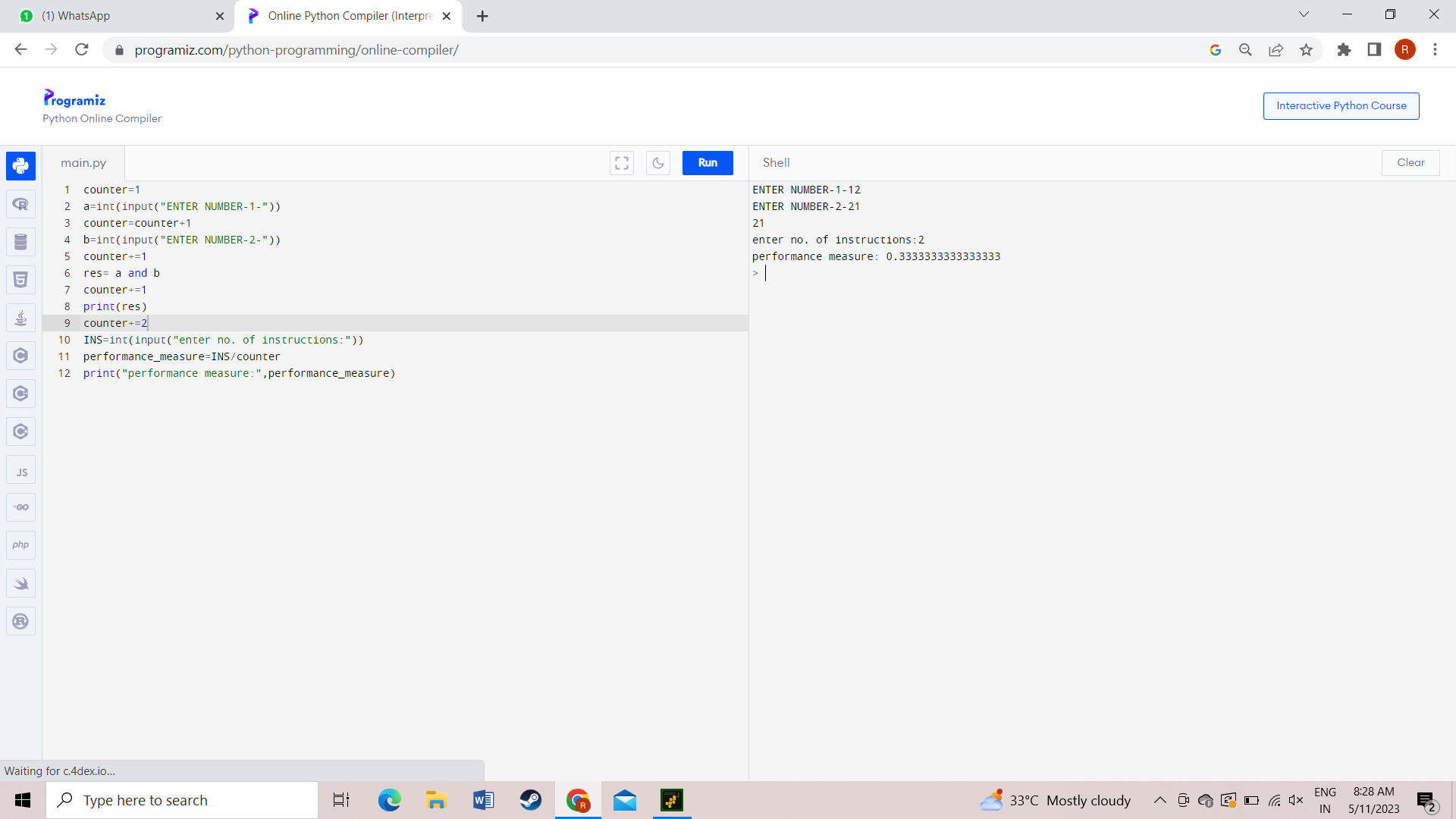
counter=counter+2

INS=int(input("enter no. of instructions:"))

performance\_measure=INS/counter

print("performance measure:",performance\_measure)

**OUTPUT:**



**RESULT:**

The given program to achieve three stage pipelining using python program is completed successfully.

1. **FOUR STAGE PIPELINING**

**AIM:**

To write a python program to achieve four stage pipelining.

**PROGRAM:**

counter=1

a=int(input("ENTER NUMBER-1-"))

counter=counter+1

b=int(input("ENTER NUMBER-2-"))

counter=counter+1

print("1-ADDITION 2-SUBTRACTION 3-MULTIPLICATION 4-DIVISION")

print("Enter Your Choice")

choice=int(input())

if choice==1:

print("Performing Addition...")

res=a+b

counter=counter+1

if choice==2:

print("Performing Subtraction...")

res=a-b

counter=counter+1

if choice==3:

print("Performing Multiplication")

res=a\*b

counter=counter+1

if choice==4:

if b==0:

print("Denominator can't be Zero")

print("Performing Division")

res=a/b

counter=counter+1

if choice>=5:

print("Enter Correct Input")

print(res)

counter=counter+3

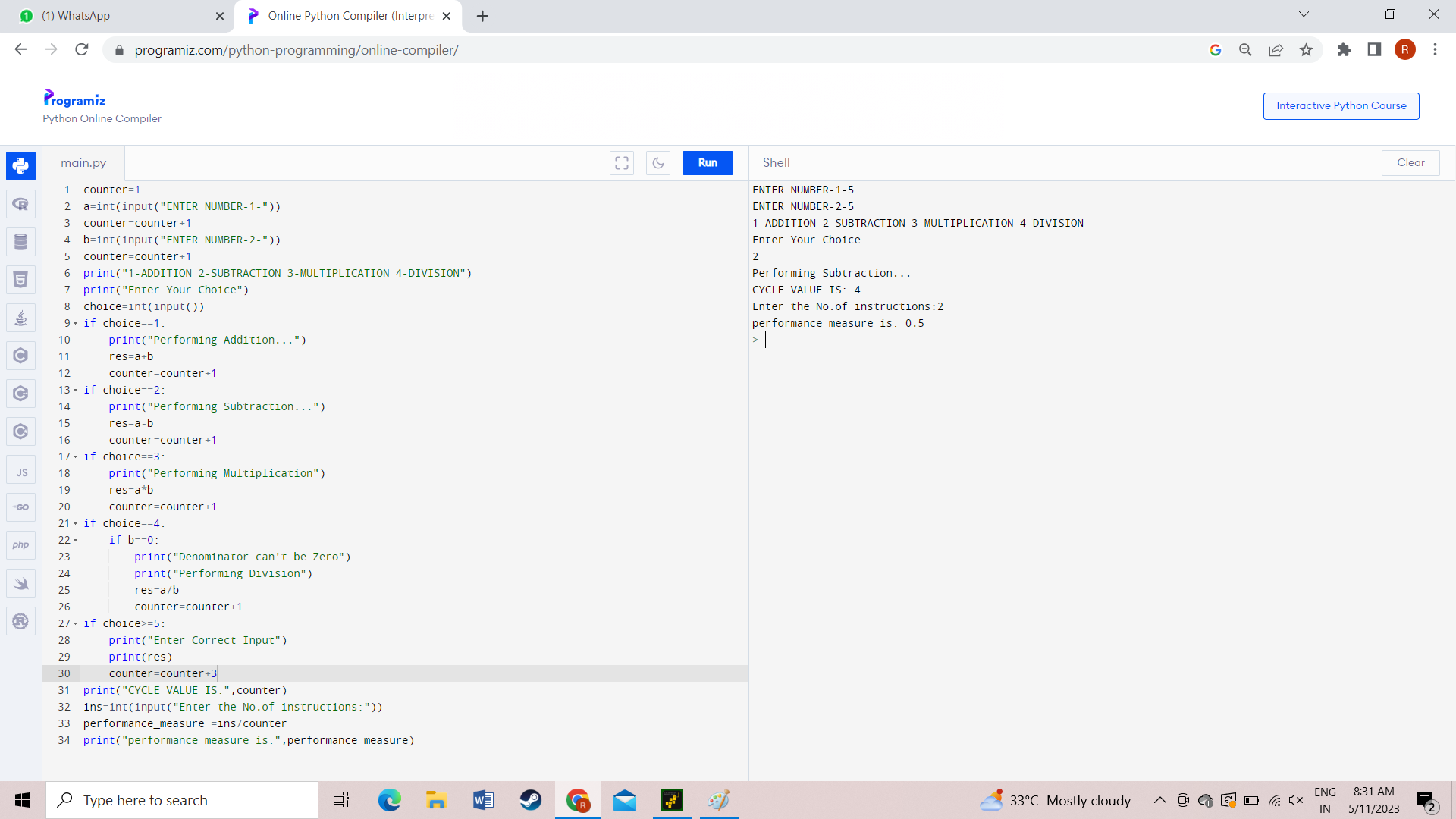
print("CYCLE VALUE IS:",counter)

ins=int(input("Enter the No.of instructions:"))

performance\_measure =ins/counter

print("performance measure is:",performance\_measure)

**OUTPUT:**



**RESULT:**

The given program to achieve four stage pipelining using python program is completed successfully.

1. **CPU PERFORMANCE PROGRAM**

**AIM:**

To write a python program to achieve CPU performance.

**PROGRAM:**

cpu = [0] \* 5

print("Enter the number of processors:")

p = int(input())

p1 = p

for i in range(p):

print("Enter the Cycles per Instruction of processor:")

cpi = float(input())

print("Enter the clock rate in GHz:")

cr = float(input())

ct = (1000 \* cpi) / cr

print("The CPU time is:", ct)

cpu[i] = ct

min\_time = cpu[0]

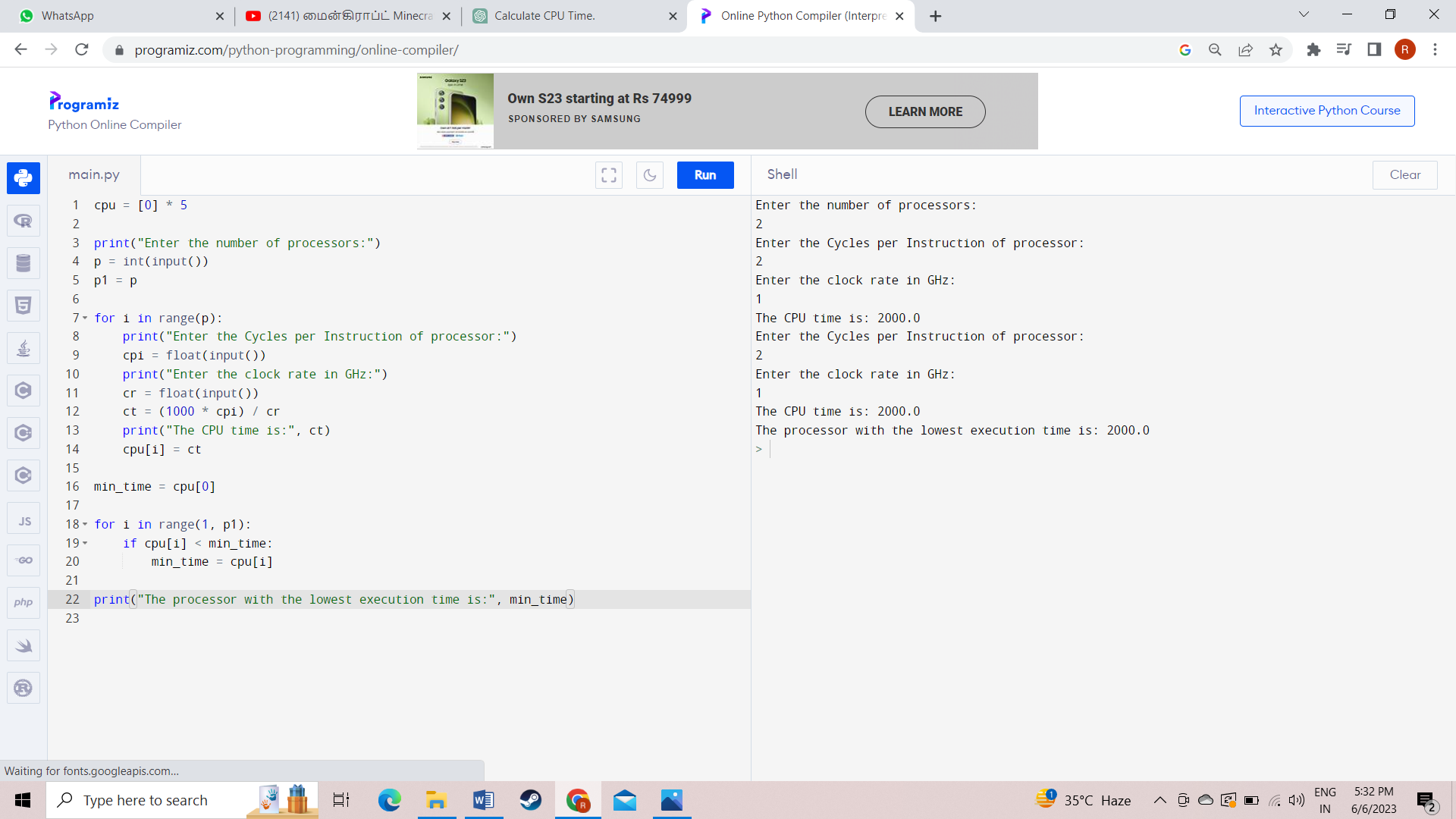
for i in range(1, p1):

if cpu[i] < min\_time:

min\_time = cpu[i]

print("The processor with the lowest execution time is:", min\_time)

**OUTPUT:**



**RESULT:**

The given program to achieve CPU performance using python program is completed successfully.

1. **LARGEST NUMBER**

AIM:

To write an assembly language program to find largest number between two numbers.

PROGRAM:

LDA 4000

MOV B,A

LDA 4001

CMP B

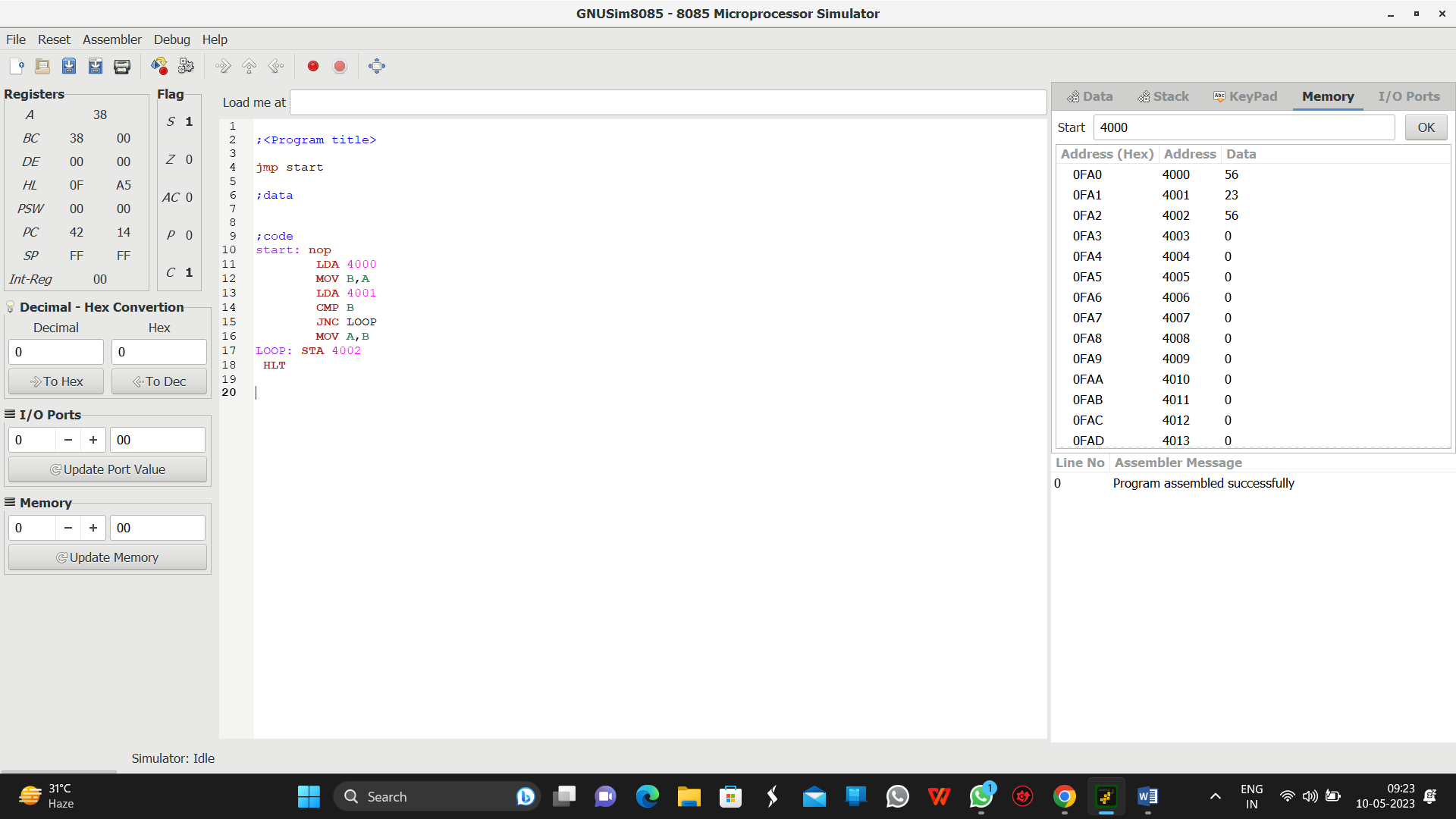
JNC LOOP

MOV A,B

LOOP: STA 4002

HLT

OUTPUT:



RESULT:

Thus the above progam is executed successfully.

1. **SMALLEST NUMBER**

**AIM:** To write an assembly language program to find smallest number between two numbers.

**PROGRAM:**

LDA 4000

MOV B,A

LDA 4001

CMP B

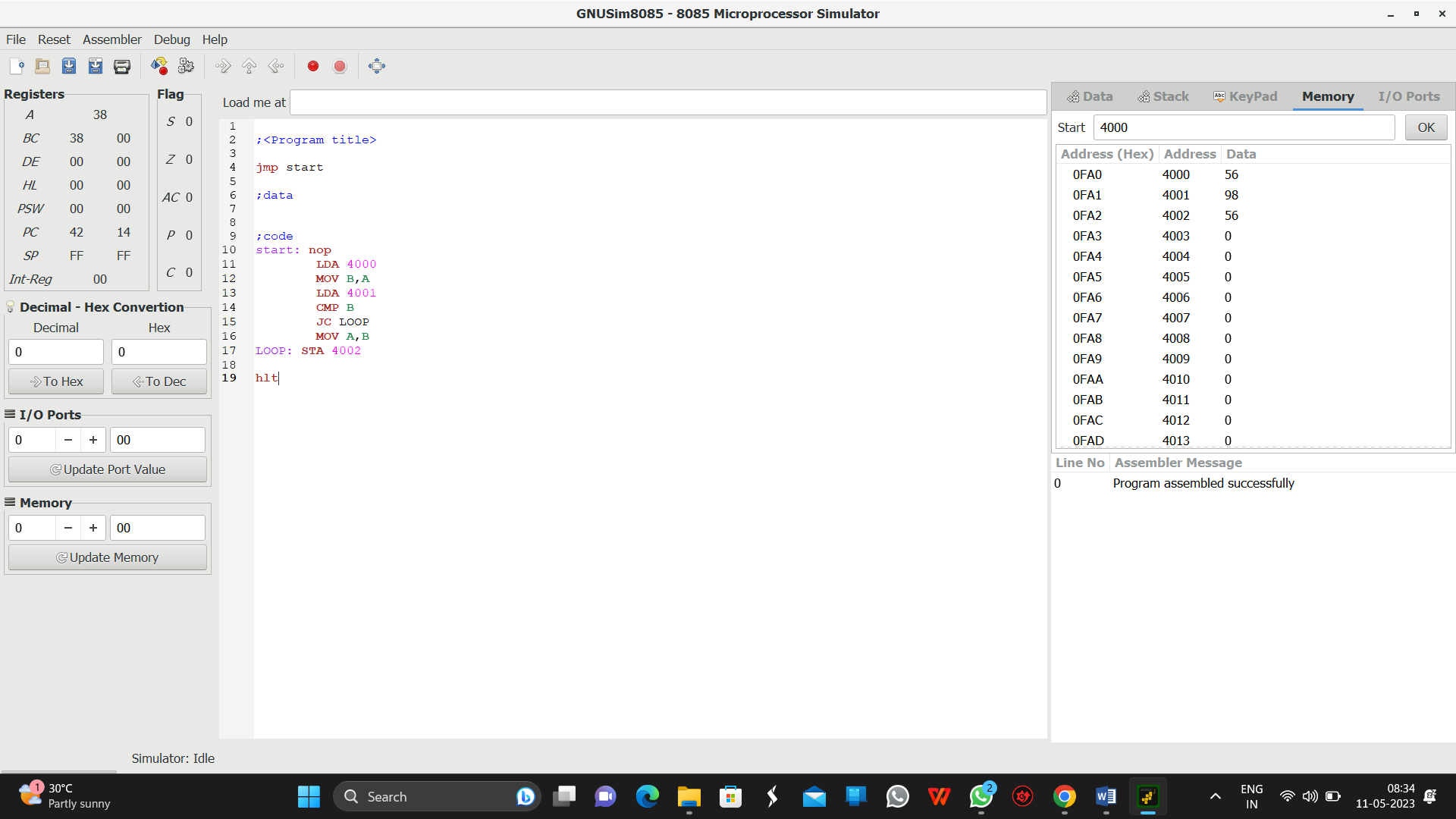
JC LOOP

MOV A,B

LOOP: STA 4002

Hlt

OUTPUT:



RESULT:

Thus the above progam is executed successfully.

1. **SWAP TWO NUMBERS**

AIM:

Write an assembly language program to swap two 8-bit datausing 8085 processor

PROGRAM:

start: nop

LDA 4000

MOV B,A

LDA 4001

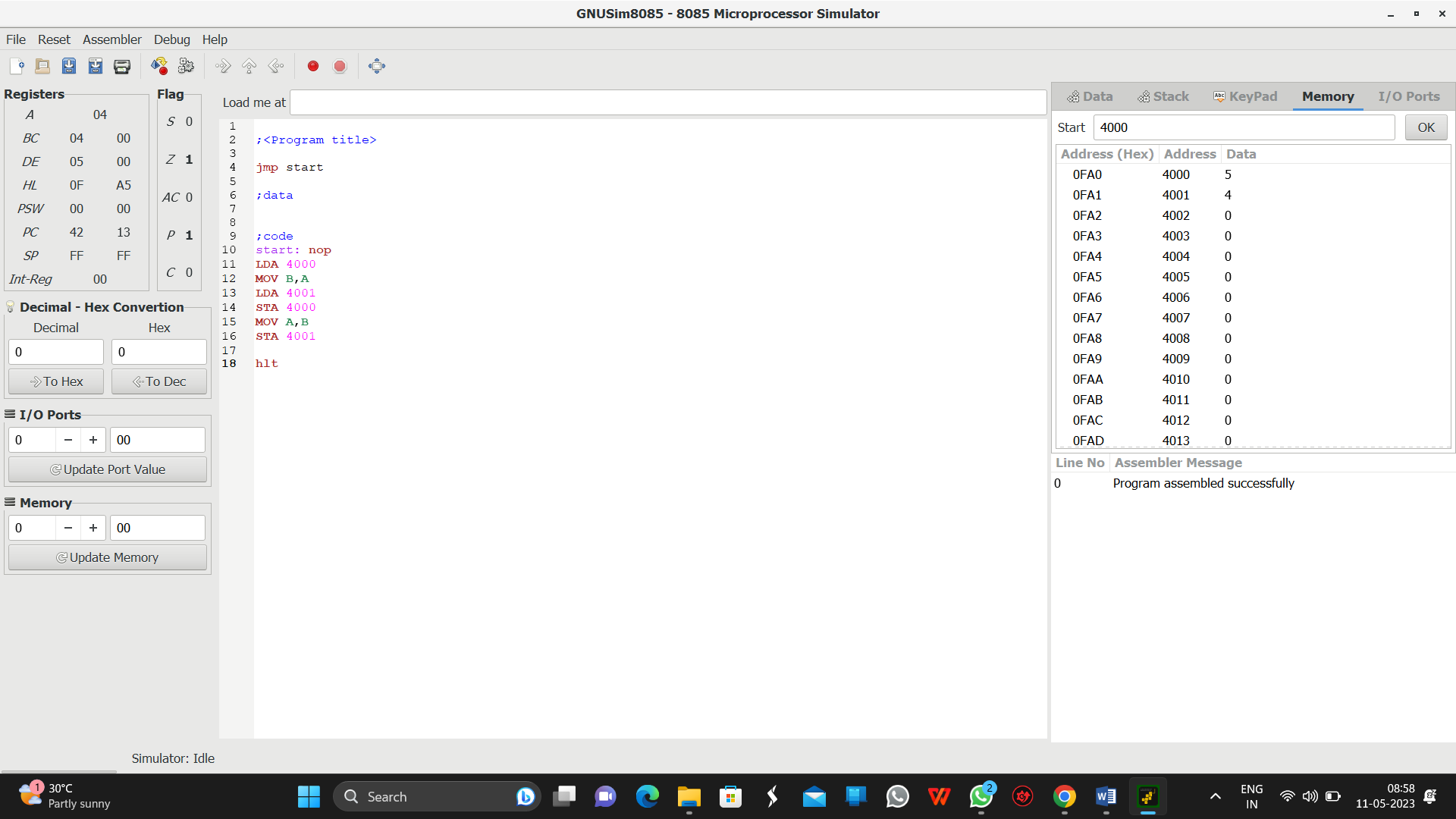
STA 4000

MOV A,B

STA 4001

Hlt

OUTPUT:



RESULT:

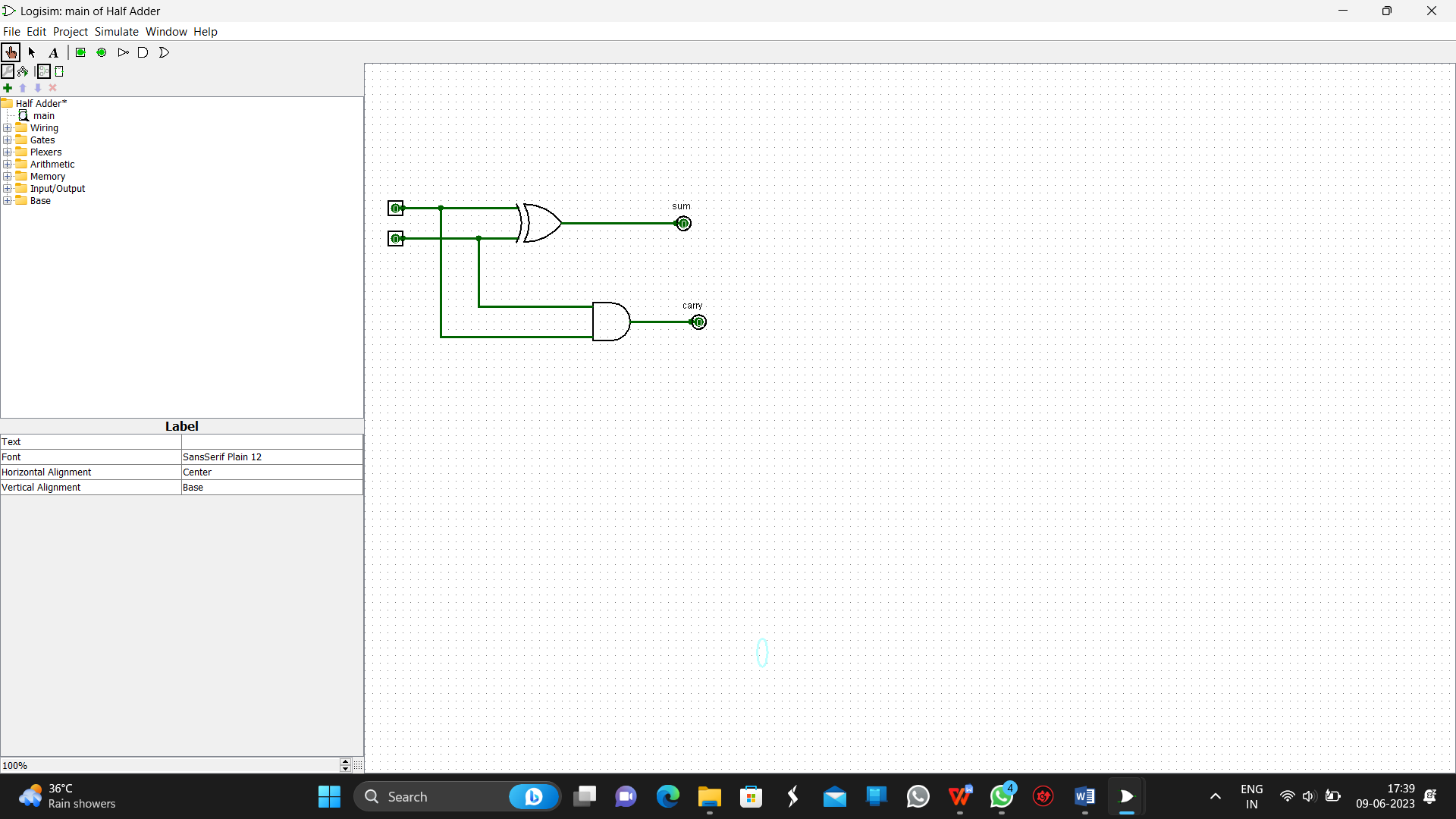
Swapping of a two numbers is successfully executed.

1. **HALF ADDER**

**AIM:**

To draw a half-adder circuit diagram using a logicism simulator.

DIAGRAM:



RESULT:

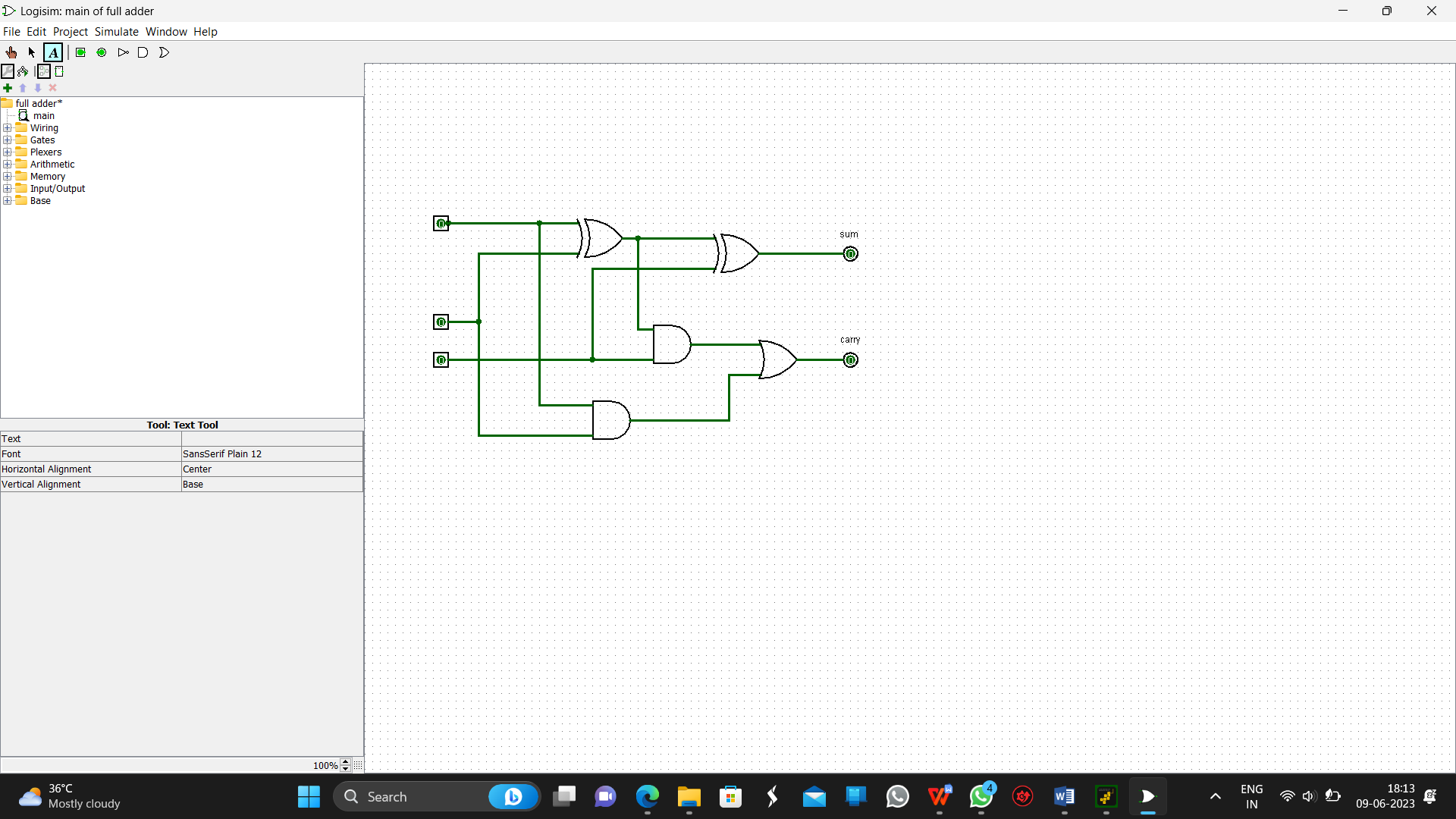
Thus the above diagram is successfully drawn using a logicism simulator.

1. **FULL ADDER**

AIM:

To draw a full adder diagram using a logicism simulator.

Diagram:



RESULT:

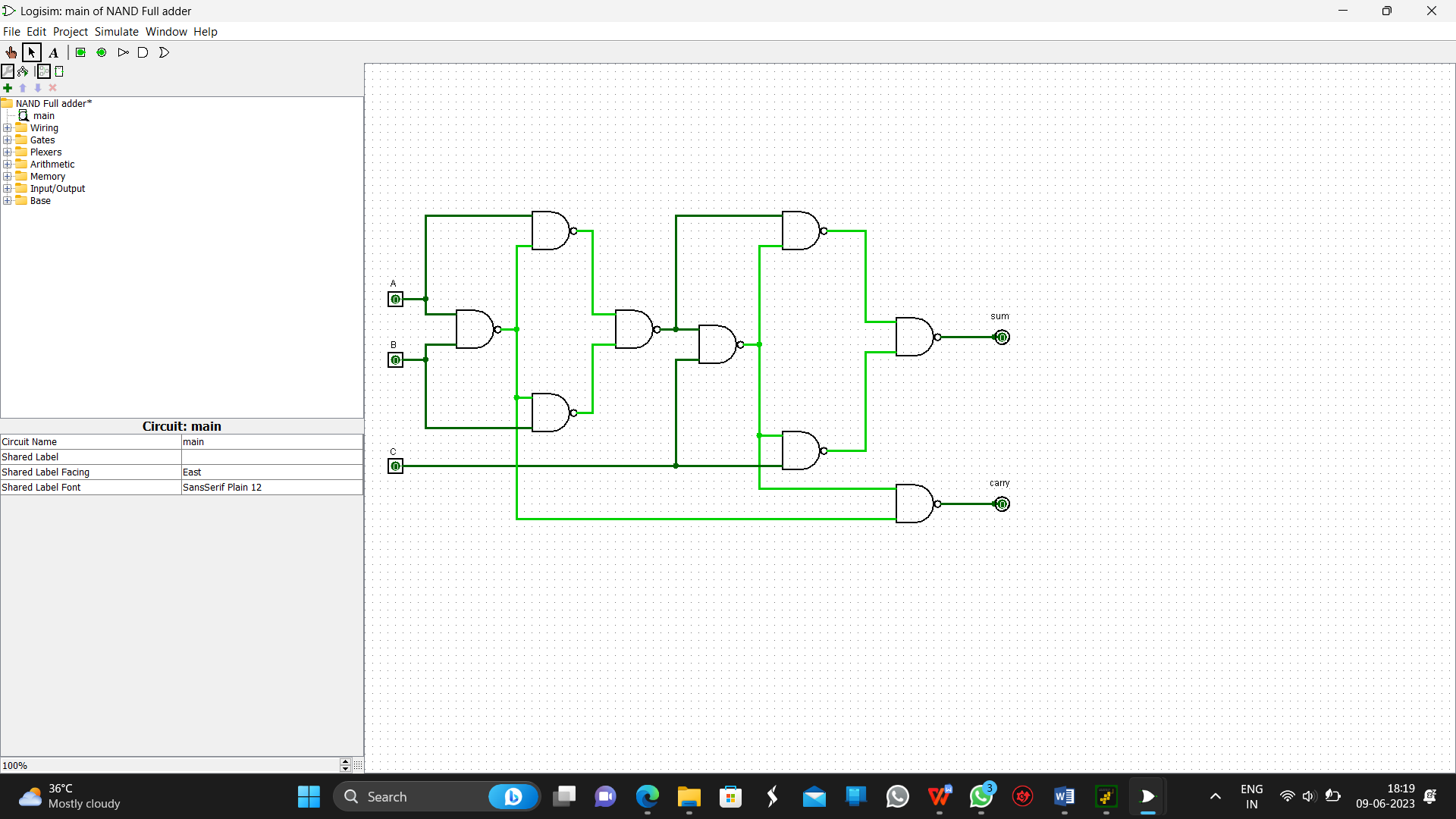
Thus the above diagram is drawn successfully.

1. **FULL ADDER USING NAND GATE**

**AIM:**

To draw the full adder circuit diagram using the NAND gate.

Diagram:



RESULT:

Thus the above diagram is drawn successfully using a logicism simulator.

1. **ADDITION OF 8-BIT WITHOUT CARRY**

**AIM:**

To write an assembly language program for adding two 8-bit data without carry.

PROGRAM:

start: nop

lda 3000

mov b,a

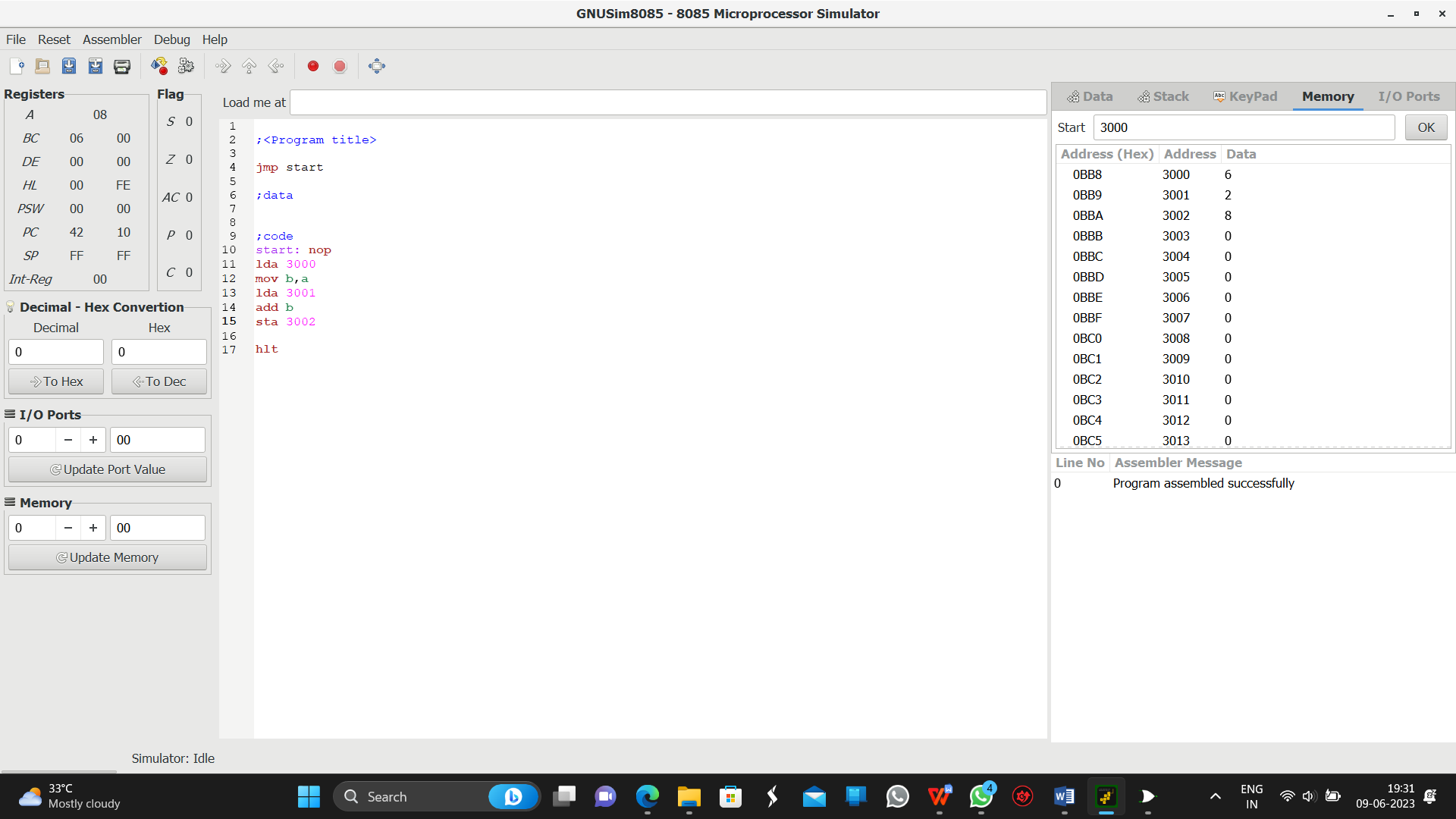
lda 3001

add b

sta 3002

hlt

OUTPUT:



RESULT:

Thus the above program is executed successfully.

1. **SUBTRACTION OF 8-BIT WITHOUT CARRY**

**AIM:**

To write an assembly language program for subtracting two 8-bit data without carry.

PROGRAM:

start: nop

LDA 3000

MOV B,A

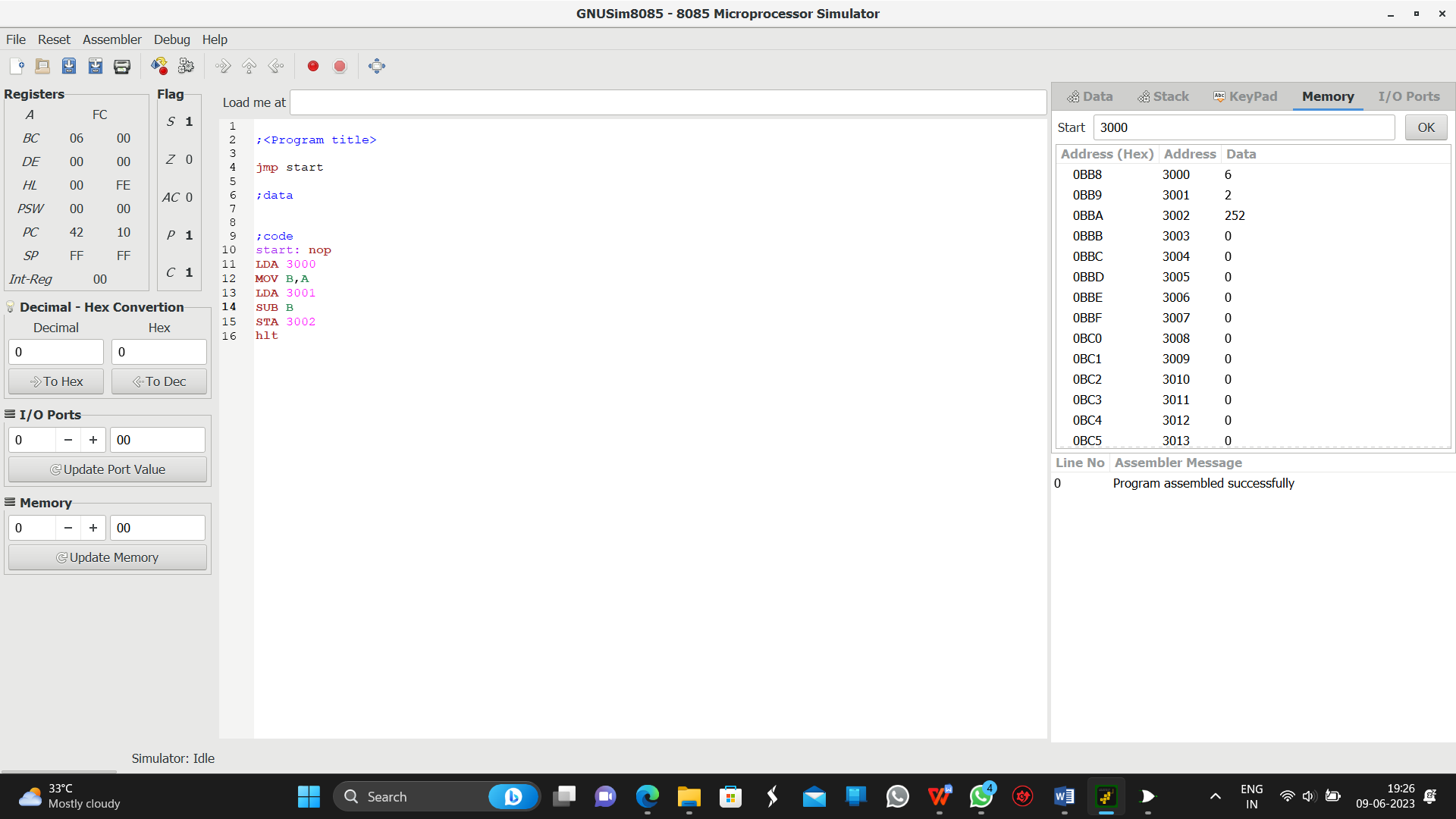
LDA 3001

SUB B

STA 3002

Hlt

OUTPUT:



RESULT:

Thus the above program is executed successfully.

1. **PYTHON PROGRAM FOR ASCENDING ORDER**

**AIM:**

To Python program to arrange the set of numbers in Ascending order.

PROGRAM:

arr = [5, 2, 8, 7, 1];

temp = 0;

print("Elements of original array: ");

for i in range(0, len(arr)):

print(arr[i]),

for i in range(0, len(arr)):

for j in range(i+1, len(arr)):

if(arr[i] > arr[j]):

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

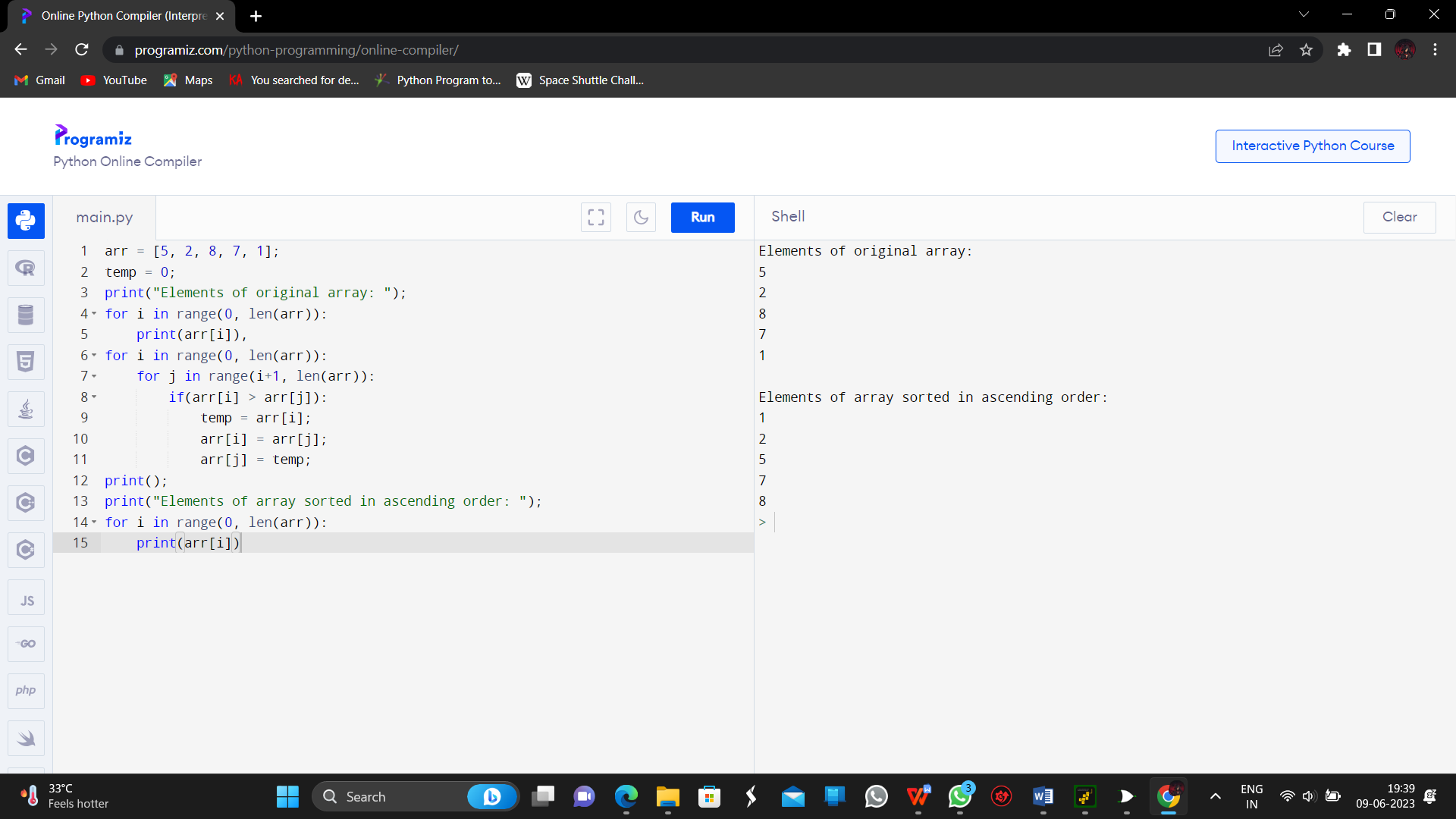
print();

print("Elements of array sorted in ascending order: ");

for i in range(0, len(arr)):

print(arr[i])

OUTPUT:



RESULT:

Thus the above program is executed successfully.

1. **PYTHON PROGRAM FOR DESCENDING ORDER**

**AIM:**

To Python program to arrange the set of numbers in Descending order.

PROGRAM:

arr = [5, 2, 8, 7, 1];

temp = 0;

print("Elements of original array: ");

for i in range(0, len(arr)):

print(arr[i]),

for i in range(0, len(arr)):

for j in range(i+1, len(arr)):

if(arr[i] < arr[j]):

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

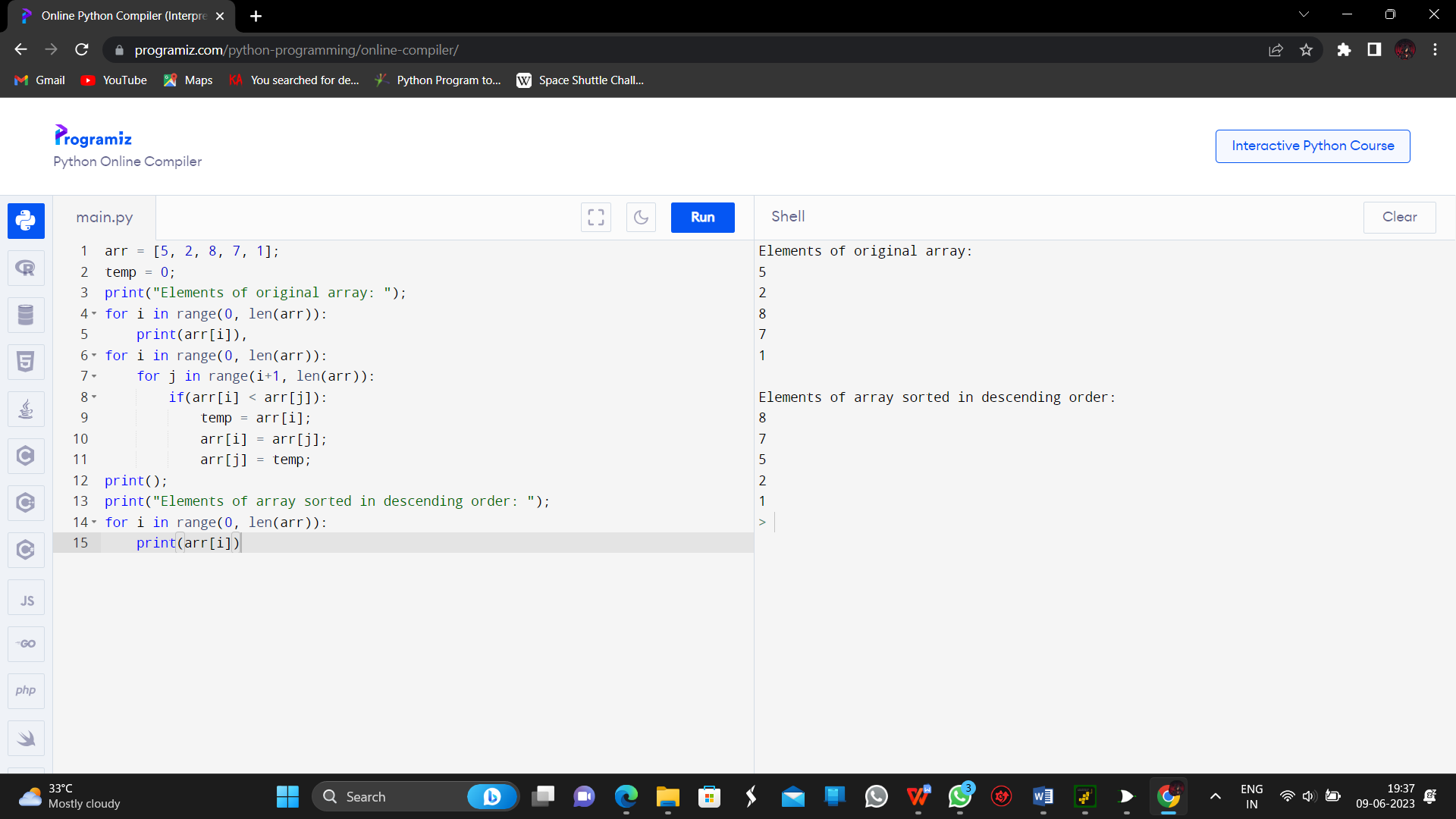
print();

print("Elements of array sorted in descending order: ");

for i in range(0, len(arr)):

print(arr[i])

OUTPUT:



RESULT:

Thus the above program is executed successfully.