

1. Write an assembly language program for adding two 8-bit data A7 A6 A5 A4 A3 A2 A1 A0 and B7 B6 B5 B4 B3 B2 B1 B0 using 8085 processor.

PROGRAM:

LDA 2000

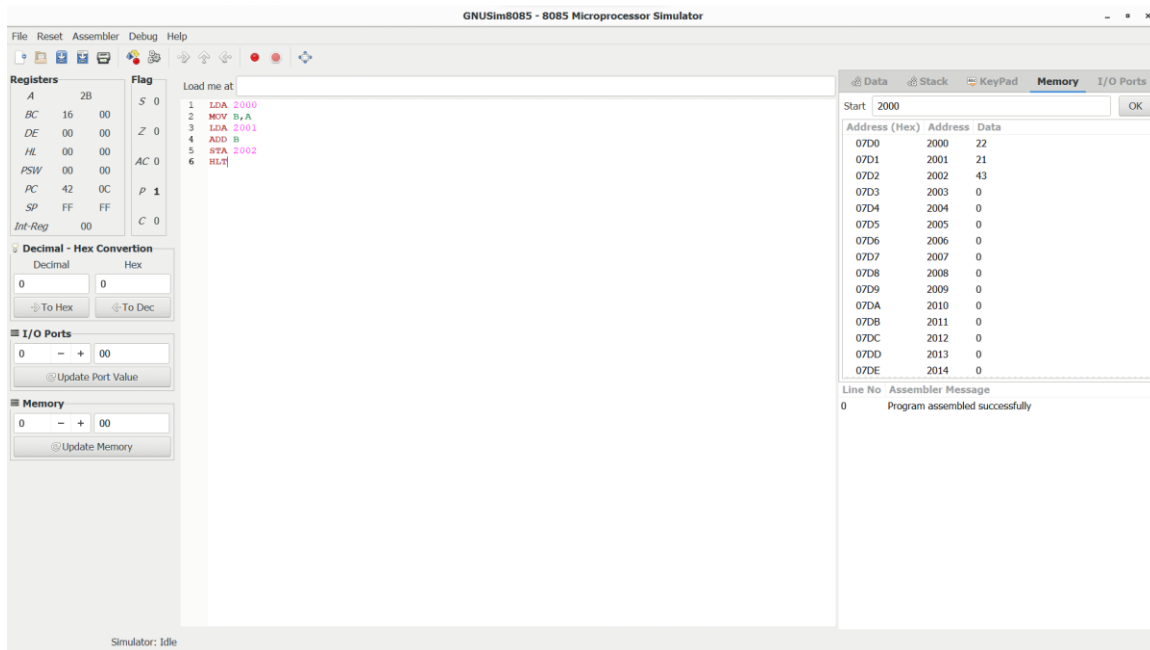
MOV B,A

LDA 2001

ADD B

STA 2002

HLT



2. Write an assembly language program for subtraction of two 8-bit data A7 A6 A5 A4 A3 A2 A1 A0 and B7 B6 B5 B4 B3 B2 B1 B0 using 8085 processor.

PROGRAM:

LDA 2000

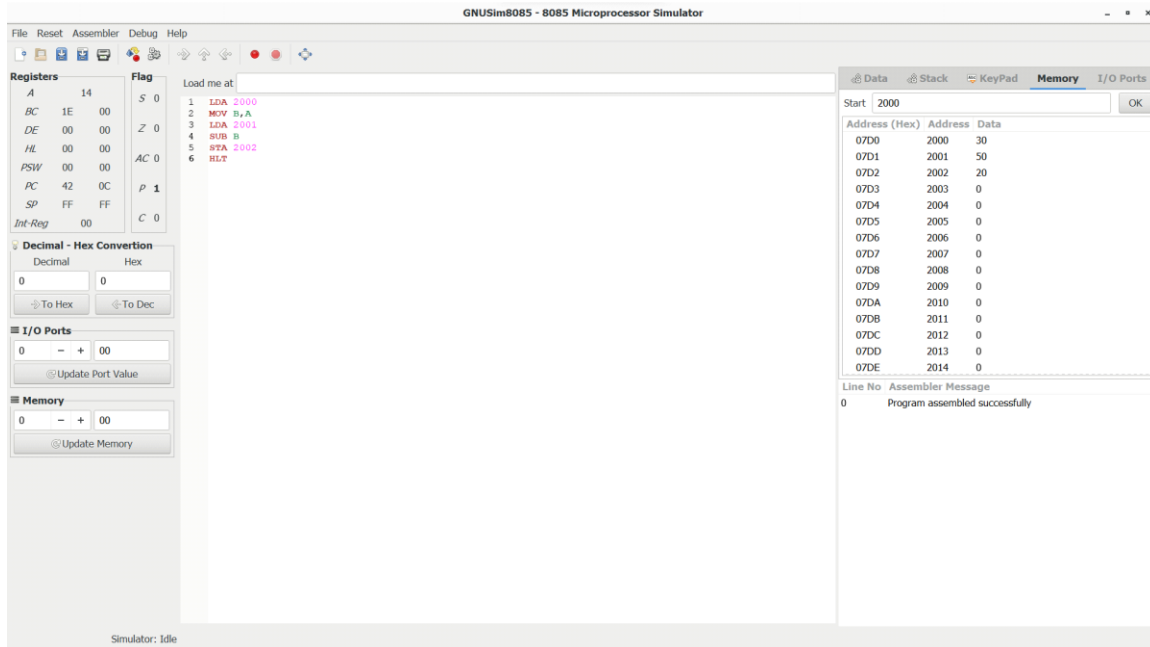
MOV B,A

LDA 2001

SUB B

STA 2002

HLT



3. Write an assembly language program for adding two 16-bit data using 8086 processor.

PROGRAM:

LDA 2050

MOV B,A

LDA 2052

ADD B

STA 2060

LDA 2051

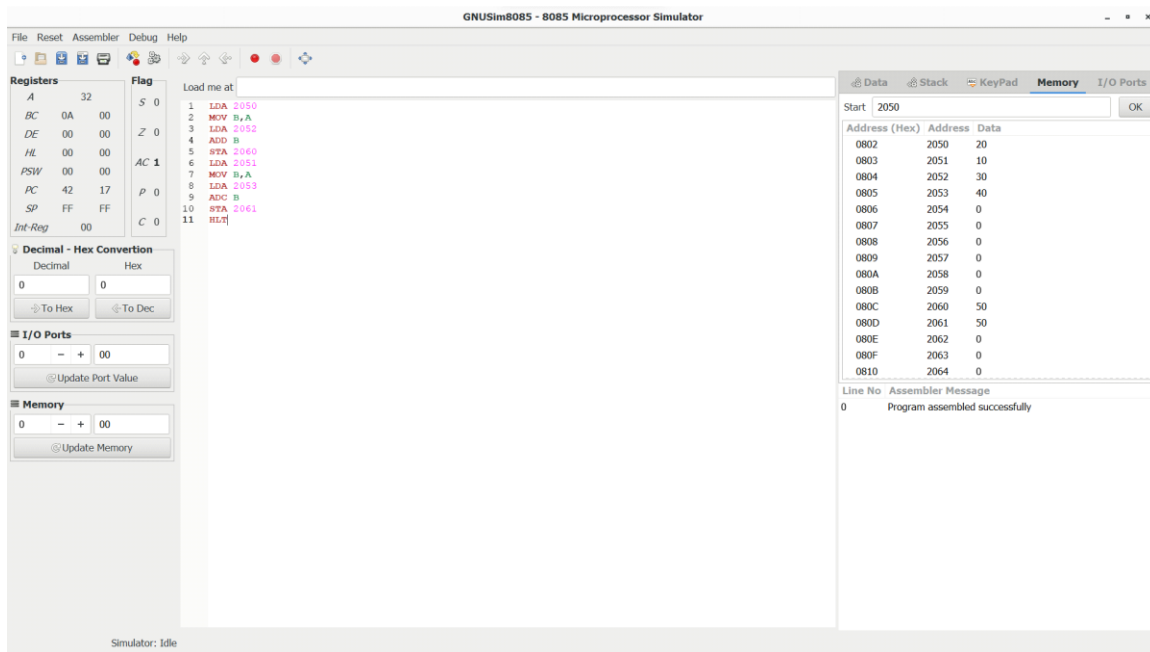
MOV B,A

LDA 2053

ADC B

STA 2061

HLT



4. Write an assembly language program for subtracting two 16-bit data using 8086 processor.

PROGRAM:

LDA 2050

MOV B,A

LDA 2052

SUB B

STA 2060

LDA 2051

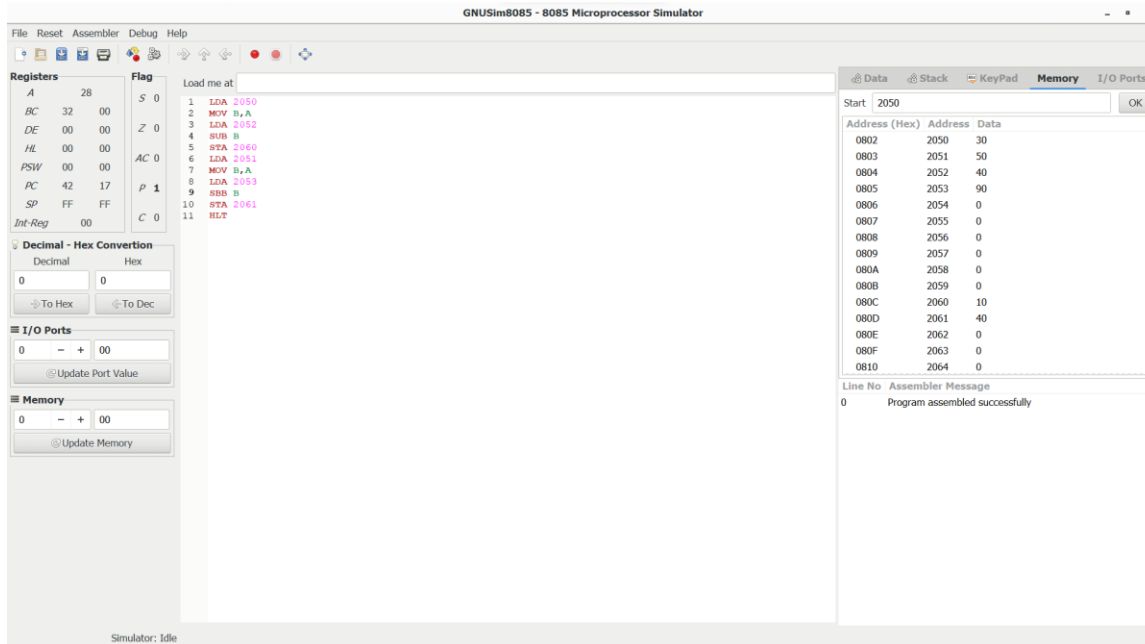
MOV B,A

LDA 2053

SBB B

STA 2061

HLT



5. Write an assembly language program for multiplication of two 8-bit data A7 A6 A5

A4 A3 A2 A1 A0 and B7 B6 B5 B4 B3 B2 B1 B0 using 8085 processor

PROGRAM:

LHLD 2050

XCHG

MOV C,D

MVI D 00

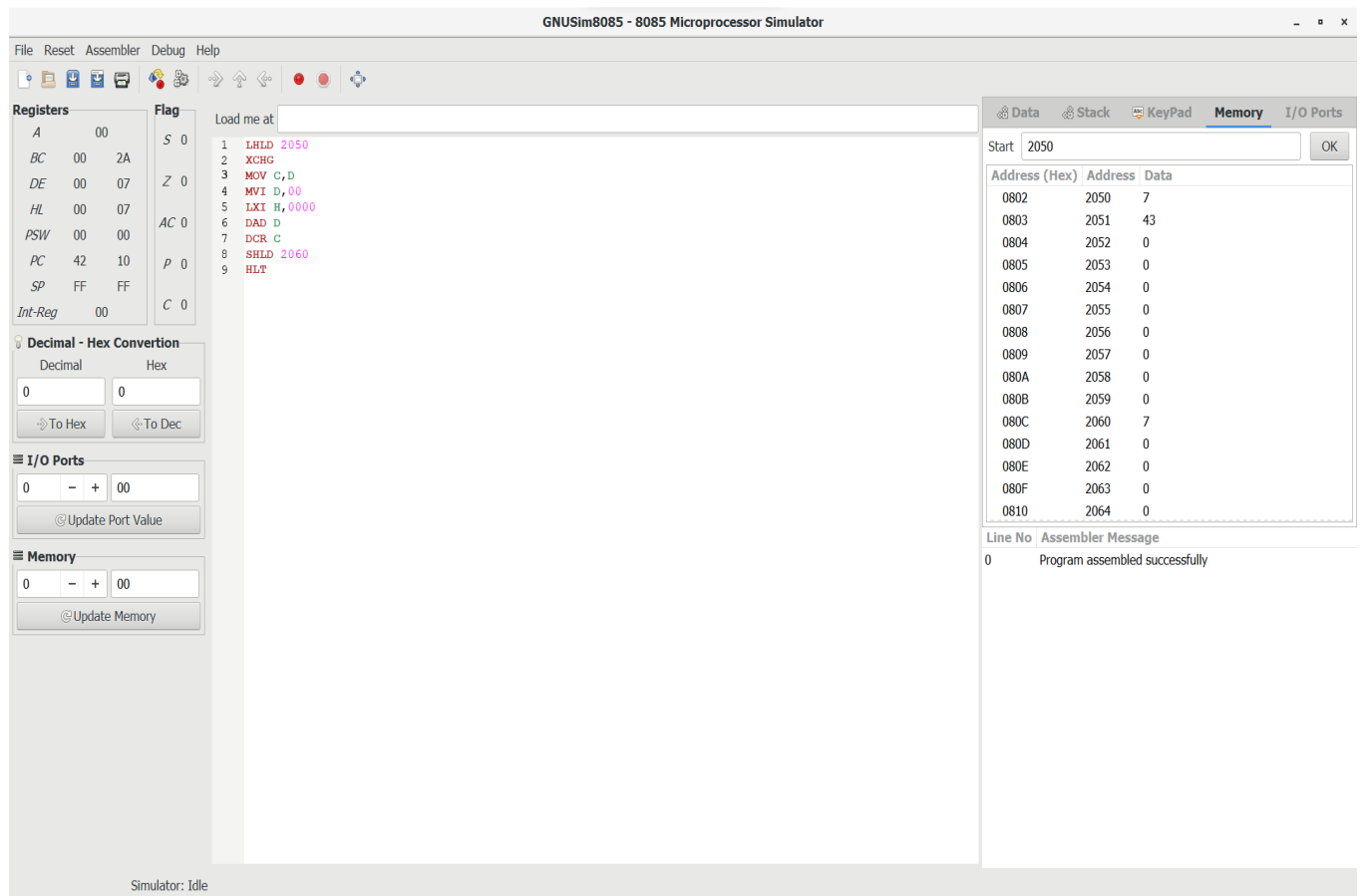
LXI H 0000

DAD D

DCR C

SHLD 2060

HLT



6. Write an assembly language program for division of two 8-bit data A7 A6 A5 A4 A3 A2 A1 A0 and B7 B6 B5 B4 B3 B2 B1 B0 using 8085 processor.

PROGRAM:

LXI H,2050

MOV B,M

MVI C,00

INX H

MOV A,M

CMP B

SUB B

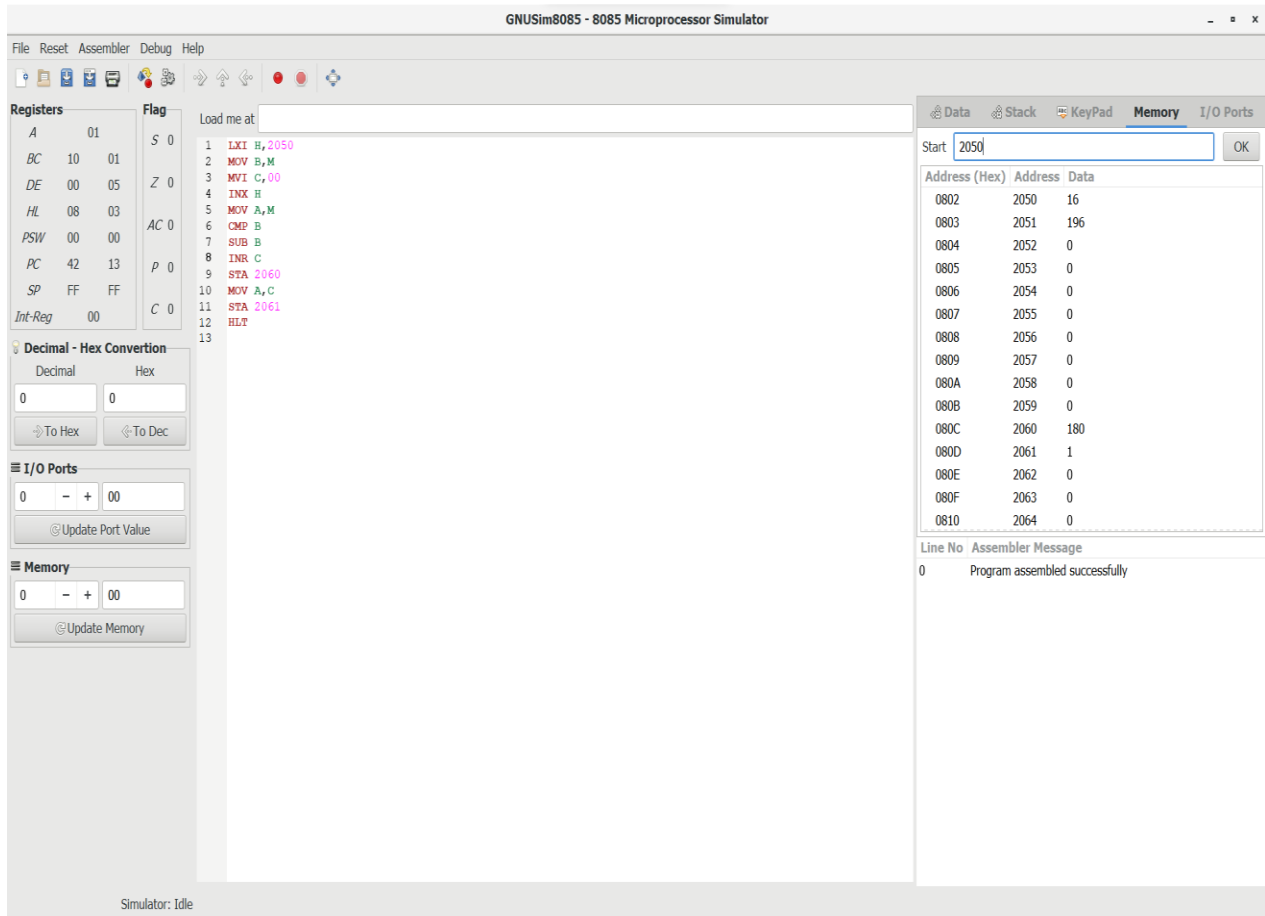
INR C

STA 2060

MOV A,C

STA 2061

HLT



7. Write an assembly language program for multiplying two 16-bit data using 8086 processor.

PROGRAM:

LHLD 2050

SPHL

LHLD 2052

XCHG

LXI H,0000H

LXI H,0000H

DAD SP

INX B

DCX D

MOV A,E

ORA D

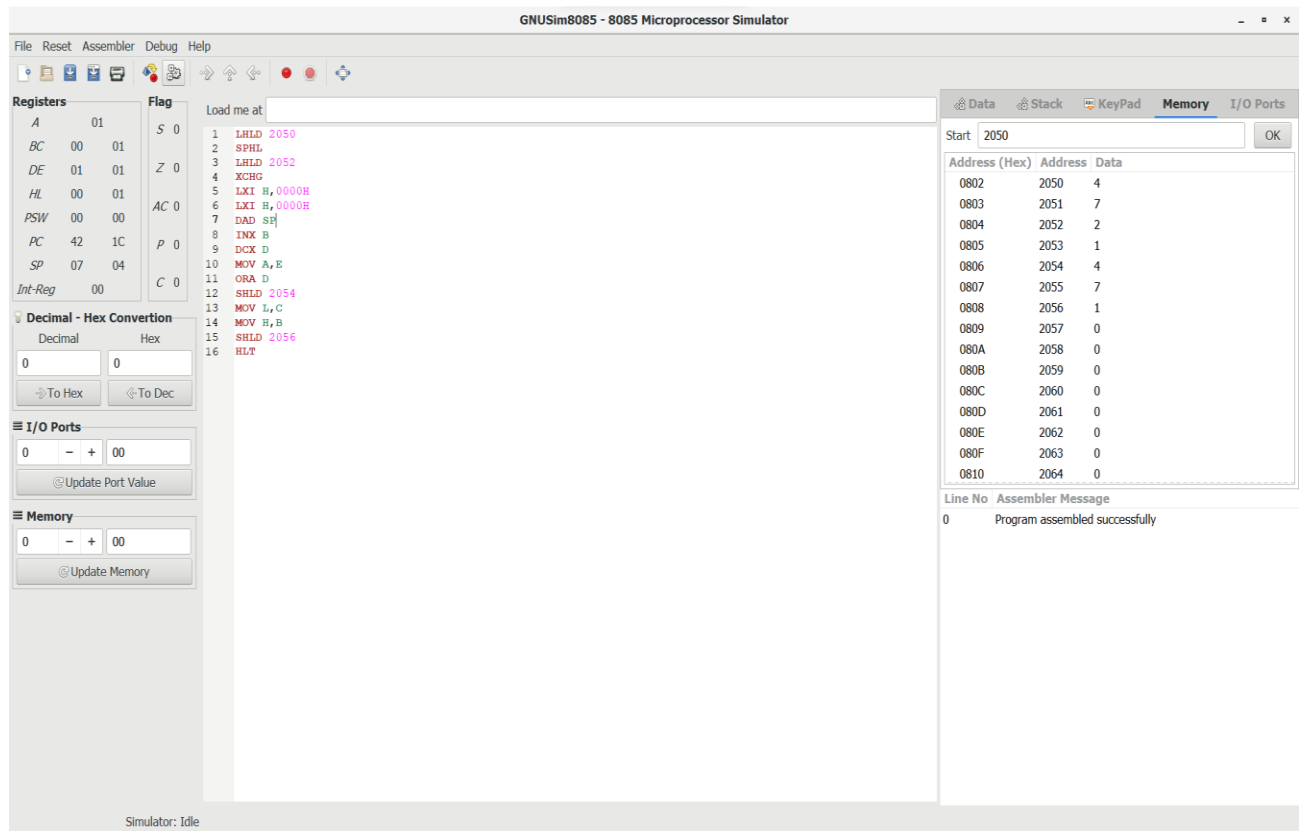
SHLD 2054

MOV L,C

MOV H,B

SHLD 2056

HLT



8. Write an assembly language program for dividing two 16-bit data using 8086 processor.

PROGRAM:

LXI B,0000H

LHLD 2052H

XCHG

LHLD 2050

MOV A,L

SUB E

MOV L,A

MOV A,H

SBB D

MOV H,A

INX B

DAD D

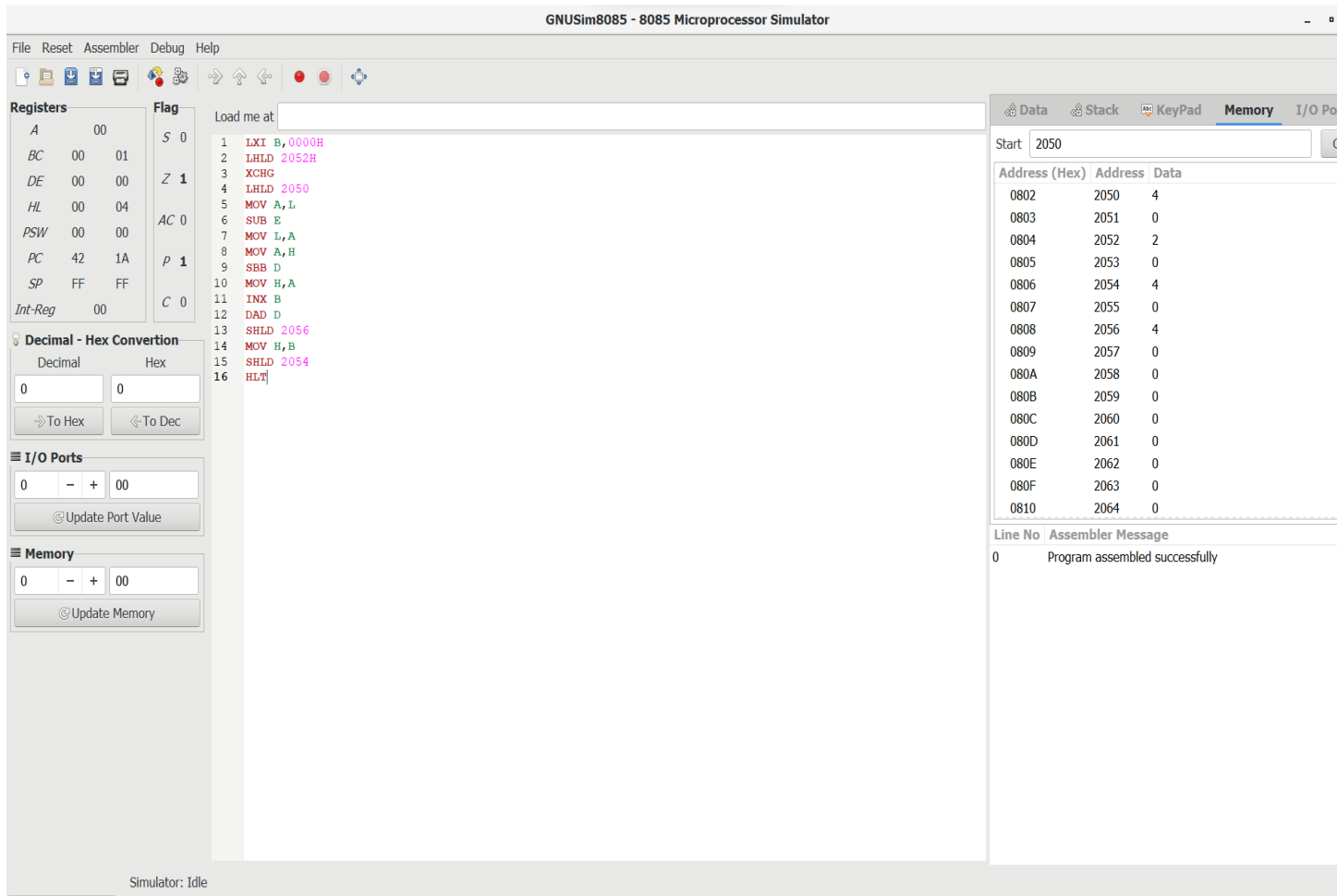
SHLD 2056

MOV L,C

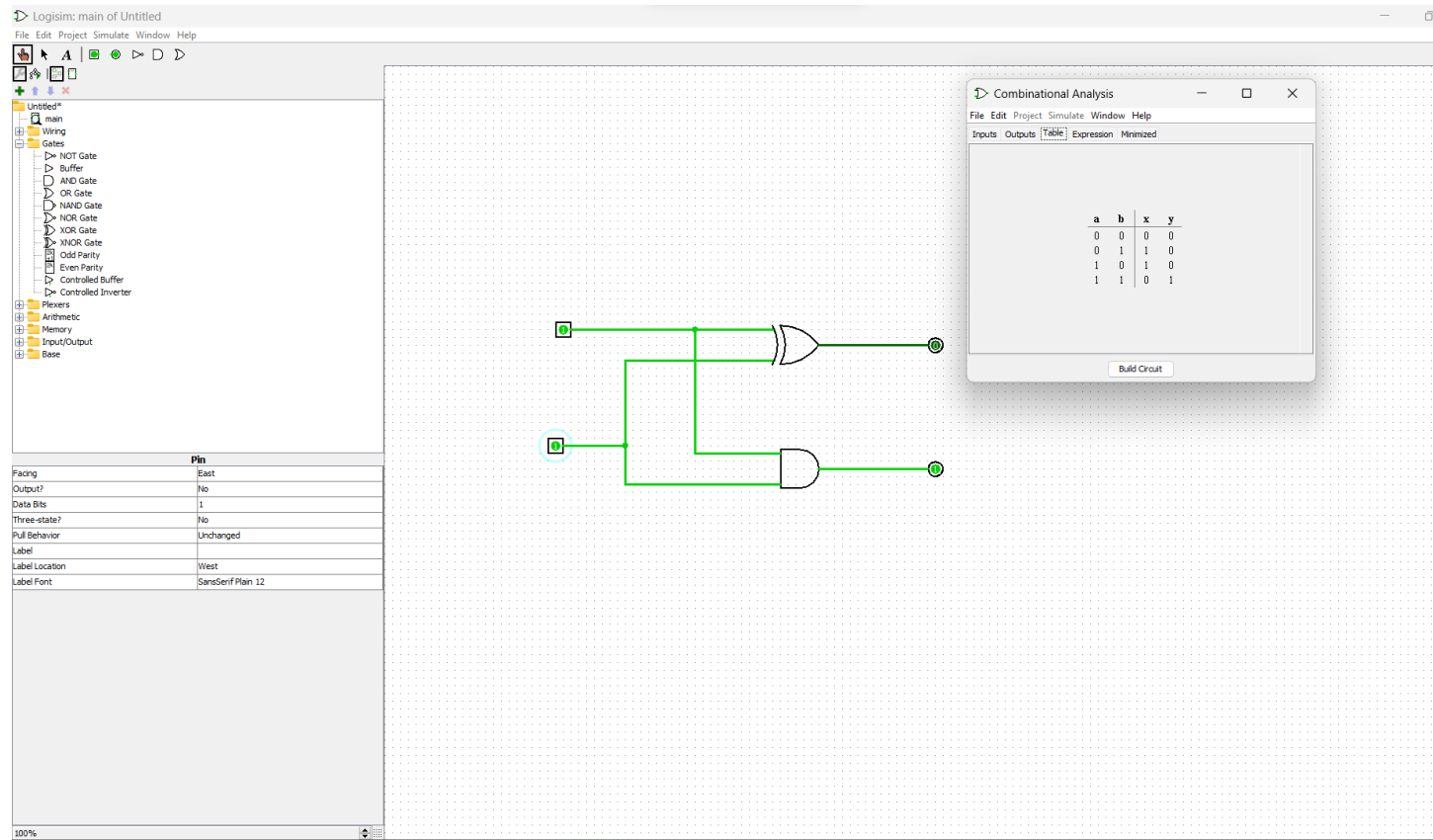
MOV H,B

SHLD 2054

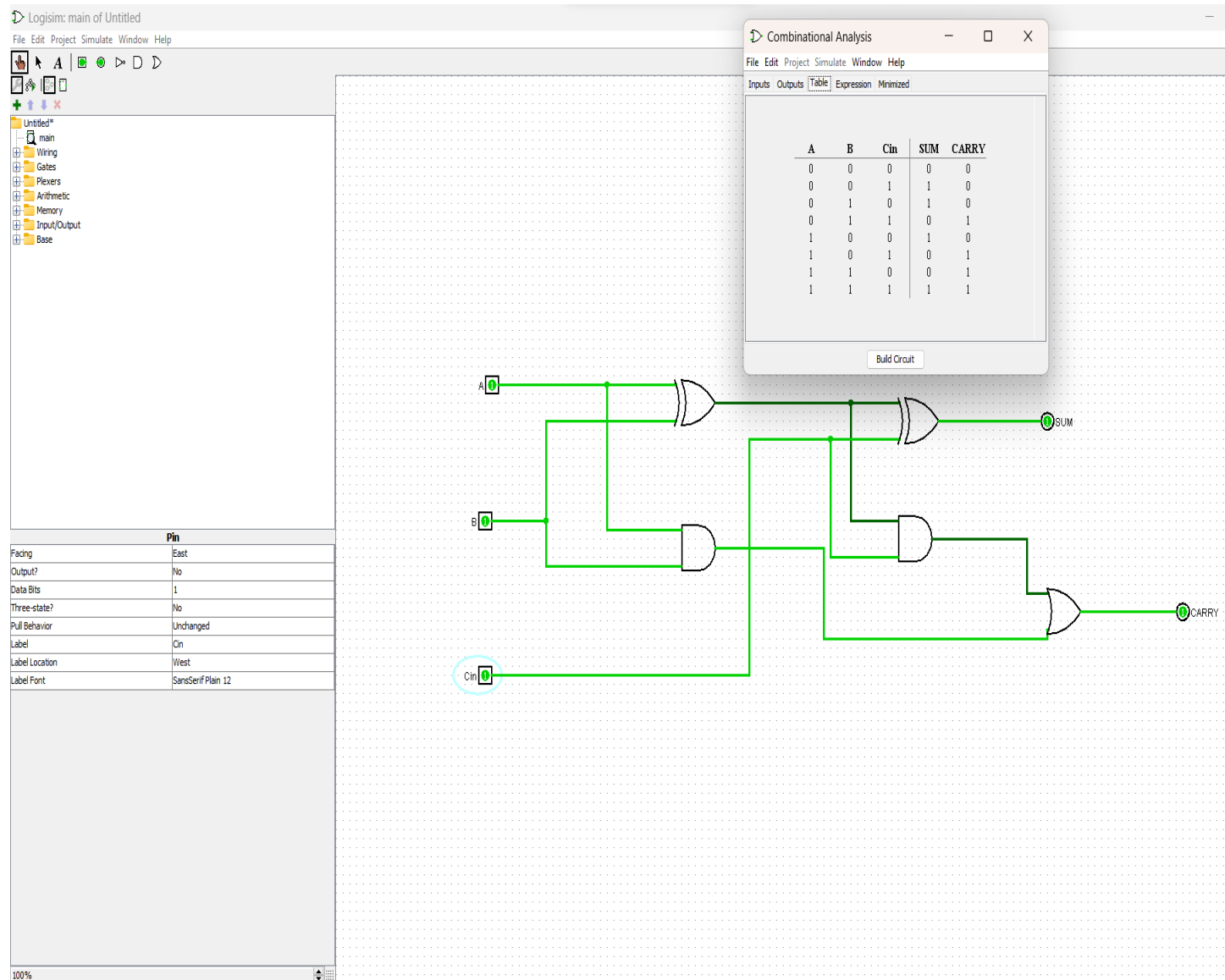
HLT



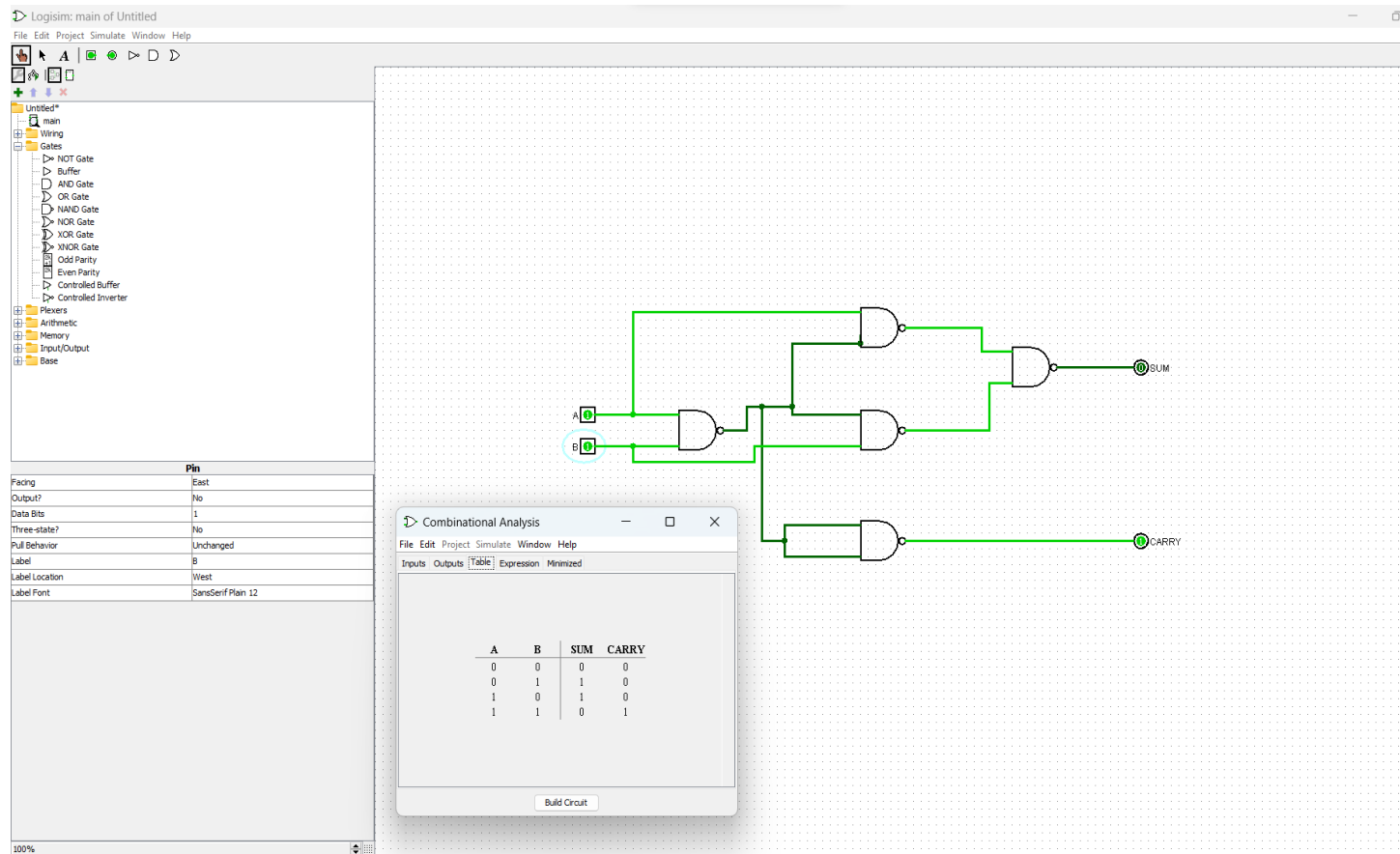
9. Design and implement 2-bit half adder using logisim simulator.



10. Design and implement 3-bit full adder using logisim simulator.



11. Design and implement 2-bit half adder with NAND using logisim simulator.



25. Write an assembly language program to swap two 8-bit data using 8085 processor

PROGRAM:

LDA 0000

MOV B,A

LDA 0001

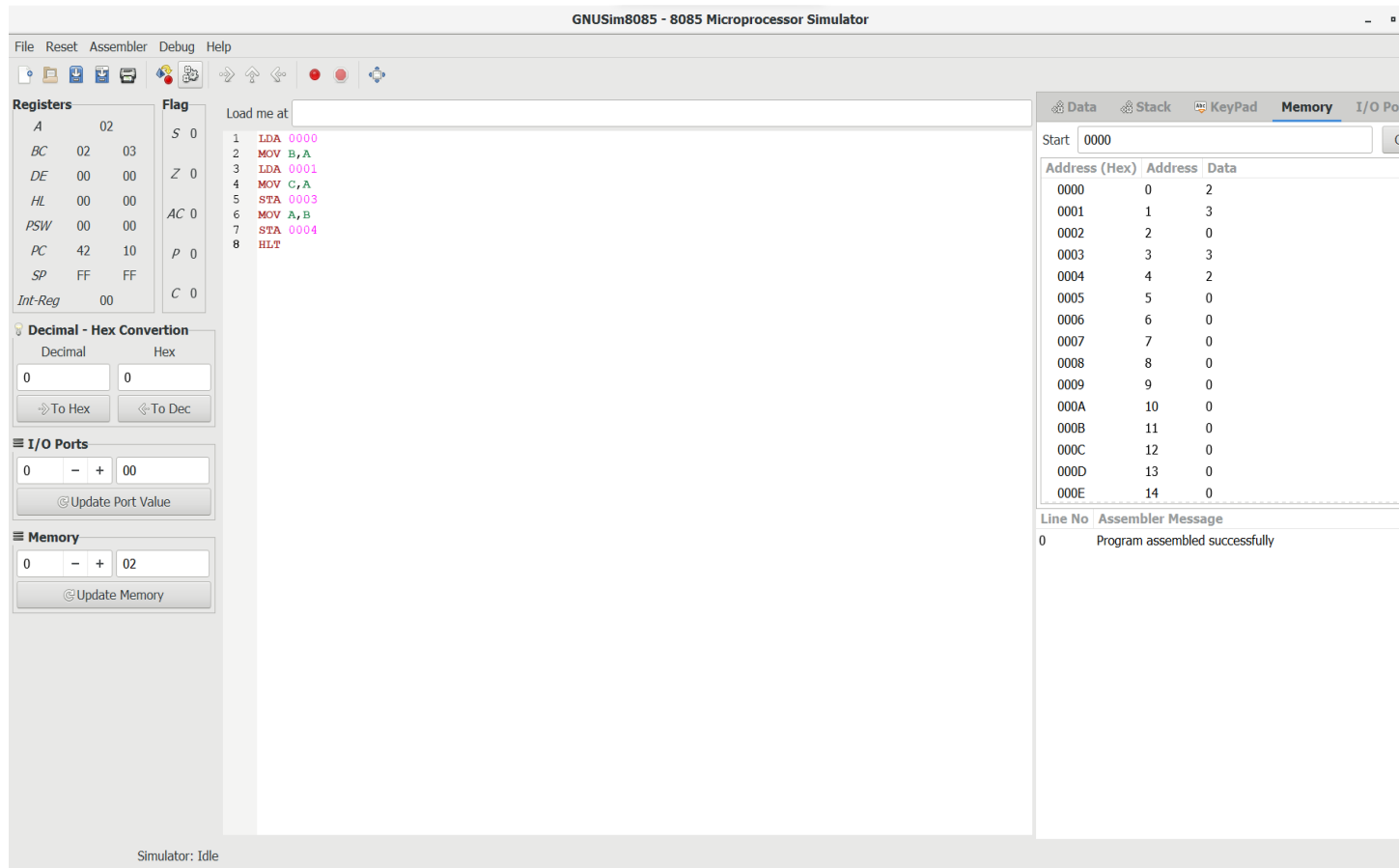
MOV C,A

STA 0003

MOV A,B

STA 0004

HLT



20. Write an assembly language program to find 1's and 2's complement of 8 bit number

PROGRM:

LDA 3000

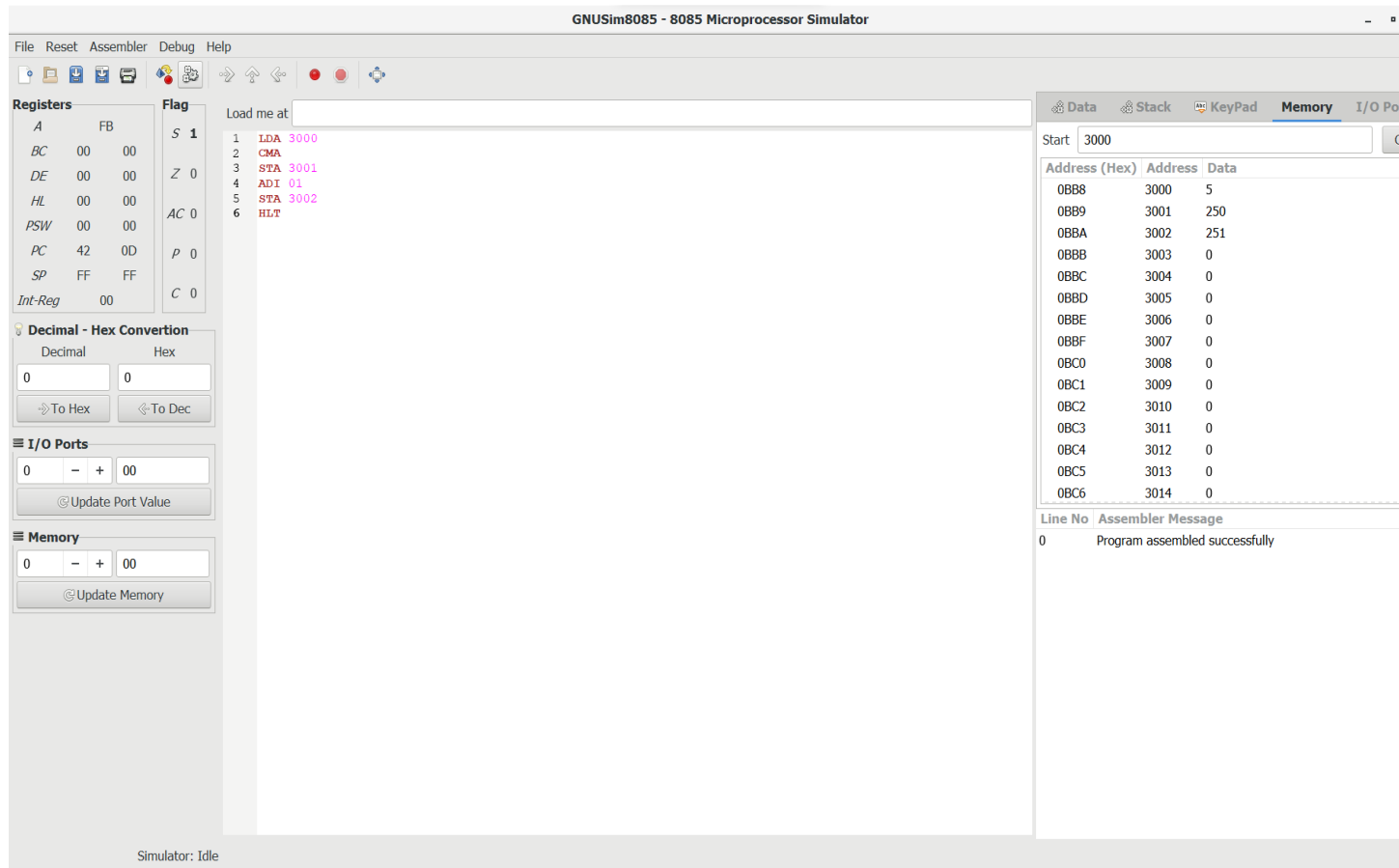
CMA

STA 3001

ADI 01

STA 3002

HLT



16. Design of 4 stage pipeline for multiplication and division of two numbers using any high level language.

PROGRAM:

```
a=int(input("Enter number1:"))
```

```
b=int(input("Enter number2:"))
```

```
c=8
```

```
mul=a*b
```

```
div=a/b
```

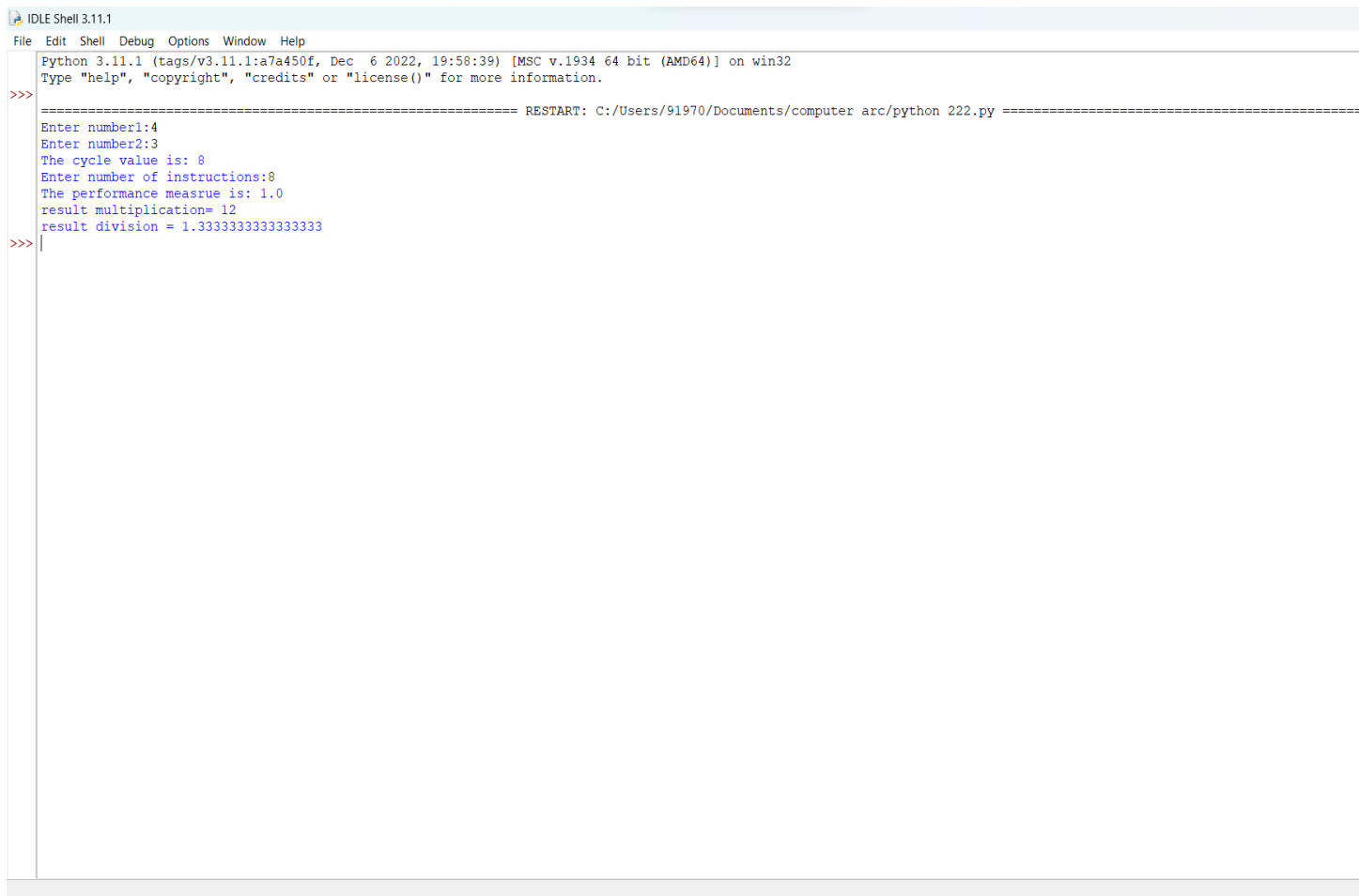
```
print("The cycle value is:",c)
```

```
ins=int(input("Enter number of instructions:"))
```

```
print("The performance measrue is:",ins/c)
```

```
print("result multiplication=",mul)
```

```
print("result division =",div)
```

A screenshot of the IDLE Shell 3.11.1 window. The title bar reads "IDLE Shell 3.11.1". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The status bar at the bottom indicates "Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32". The main text area shows the following text: "Type 'help', 'copyright', 'credits' or 'license()' for more information." followed by a separator line "===== RESTART: C:/Users/91970/Documents/computer arc/python 222.py =====". Below this, the program's output is displayed: "Enter number1:4", "Enter number2:3", "The cycle value is: 8", "Enter number of instructions:8", "The performance measrue is: 1.0", "result multiplication= 12", and "result division = 1.3333333333333333". The prompt ">>>" is visible at the end of the last line of output, with a cursor character "|" on the line below it.

```
IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/91970/Documents/computer arc/python 222.py =====
Enter number1:4
Enter number2:3
The cycle value is: 8
Enter number of instructions:8
The performance measrue is: 1.0
result multiplication= 12
result division = 1.3333333333333333
>>>|
```

15. Design of 3 stage pipeline for AND, OR, NAND of two numbers using any high level language.

PROGRAM:

```
a=int(input("Enter number 1:"))
```

```
b=int(input("Enter number 2:"))
```

```
c=4
```

```
rand=a&b
```

```
ror=a|b
```

```
rnand=~rand
```

```
print("The cycle value is",c)
```

```

ins=int(input("Enter no of instructions:"))

print("The Performance Measure:",ins/c)

print("result and=",rand)

print("result or=",ror)

print("result nand=",rnand)

```

The screenshot shows a Python IDE window titled 'python 222.py - C:/Users/91970/Documents/computer arc/python 222.py (3.11.1)'. The main editor contains the following code:

```

a=int(input("Enter number 1:"))
b=int(input("Enter number 2:"))
c=4
rand=a&b
ror=a|b
rnand=~rand
print("The cycle value")
ins=int(input("Enter no of instructions:"))
print("The Performance Measure:",ins/c)
print("result and=",rand)
print("result or=",ror)
print("result nand=",rnand)

```

An 'IDLE Shell 3.11.1' window is open in the foreground, displaying the execution output:

```

Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>
===== RESTART: C:/Users/91970/Documents/computer arc/python 222.py =====
Enter number 1:4
Enter number 2:3
The cycle value is 4
Enter no of instructions:2
The Performance Measure: 0.5
result and= 0
result or= 7
result nand= -1
>>>

```

The status bar at the bottom right of the shell window indicates 'Ln:13 Col:0'.

12. Write an assembly language program to find factorial of n in the given number.

PROGRAM:

LXI H,8000

MOV B,M

MVI D,01

FACT: CALL MUL

DCR B

JNZ FACT

INX H

MOV M,D

HLT

MUL: MOV E, B

XRA A

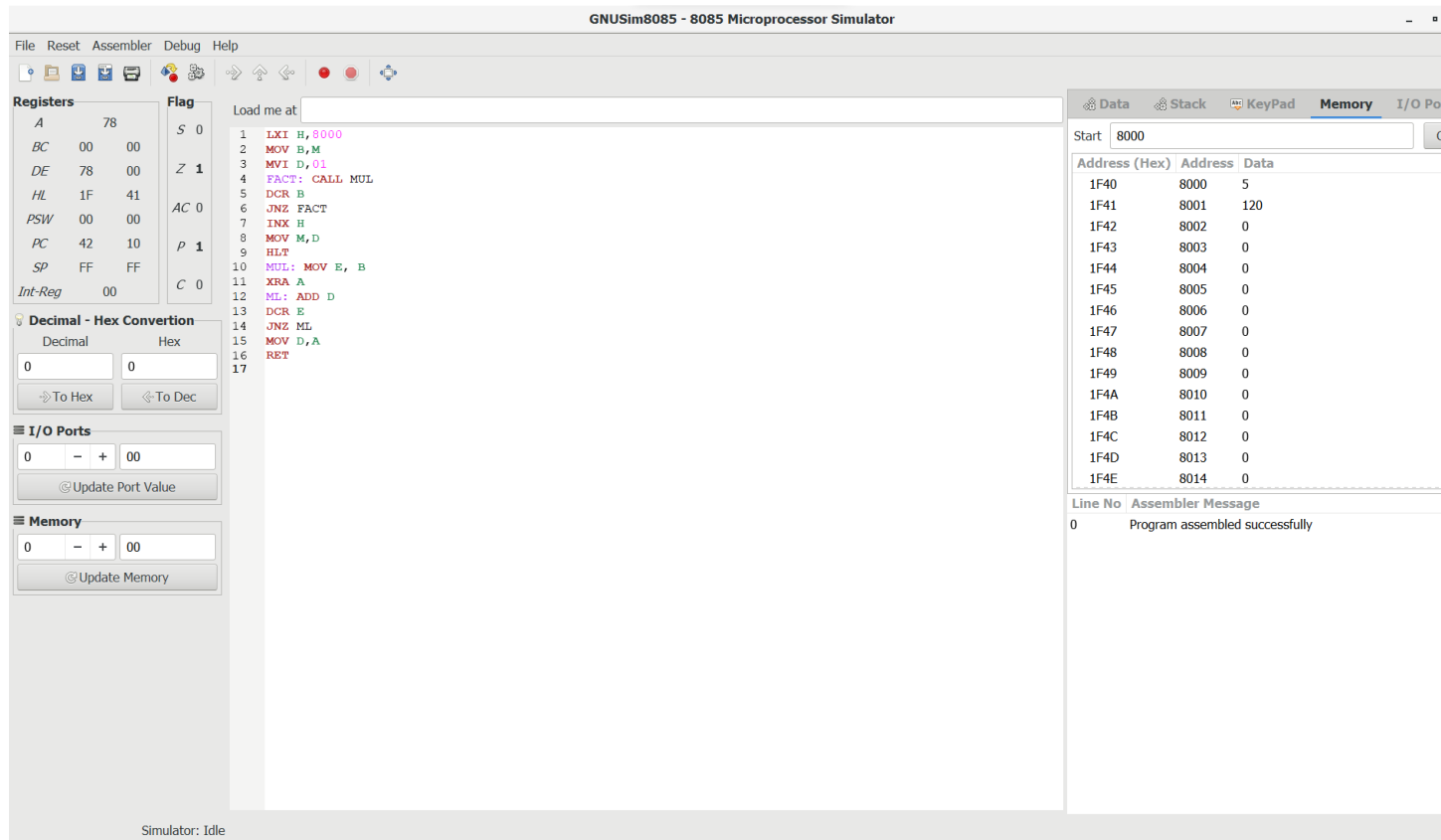
ML: ADD D

DCR E

JNZ ML

MOV D,A

RET



13. Write an assembly language program to find the largest number in an array.

PROGRAM:

LXI H,8000

MOV C,M

INX H

MOV B,M

DCR C

LOOP: INX H

MOV A,M

CMP B

JC SKIP

MOV B,A

SKIP: DCR C

JNZ LOOP

LXI H,8010

MOV M,B

HLT

GNUSim8085 - 8085 Microprocessor Simulator

File Reset Assembler Debug Help

Registers

A	00	Flag	S	0
BC	04 00		Z	1
DE	00 00		AC	0
HL	1F 4A		P	1
PSW	00 00		C	1
PC	42 17			
SP	FF FF			
Int-Reg	00			

Load me at

1 LXI H,8000
2 MOV C,M
3 INX H
4 MOV B,M
5 DCR C
6 LOOP: INX H
7 MOV A,M
8 CMP B
9 JC SKIP
10 MOV B,A
11 SKIP: DCR C
12 JNZ LOOP
13 LXI H,8010
14 MOV M,B
15 HLT
16

Decimal - Hex Conversion

Decimal	Hex
0	0
To Hex	To Dec

I/O Ports

0	-	+	00
Update Port Value			

Memory

0	-	+	00
Update Memory			

Data Stack KeyPad Memory I/O Ports

Start 8000

Address (Hex)	Address	Data
1F40	8000	4
1F41	8001	4
1F42	8002	3
1F43	8003	0
1F44	8004	0
1F45	8005	0
1F46	8006	0
1F47	8007	0
1F48	8008	0
1F49	8009	0
1F4A	8010	4
1F4B	8011	0
1F4C	8012	0
1F4D	8013	0
1F4E	8014	0

Line No Assembler Message

0 Program assembled successfully

Simulator: Idle

14. Design of 2 stage pipeline for addition and subtraction of two numbers using any high level language.

PROGRAM:

```
a=int(input("Enter number1:"))
b=int(input("Enter number2:"))

c=2

add=a+b

sub=a-b

print("The cycle value is:",c)

ins=int(input("Enter number of instructions:"))

print("The performance measrue is:",ins/c)

print("result addition=",add)

print("result subtraction =",sub)
```

```
IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/91970/14.py =====
Enter number1:5
Enter number2:3
The cycle value is: 2
Enter number of instructions:7
The performance measrue is: 3.5
result addition= 8
result subtraction = 2
>>>
```

24. Write a program to find the CPU performance of a processor using any high level language.

PROGRAM:

```

p=int(input("Enter no.of performance:"))

ct=[]

for i in range(0,p):

    cp=float(input("Enter the clock cycle per instruction:"))

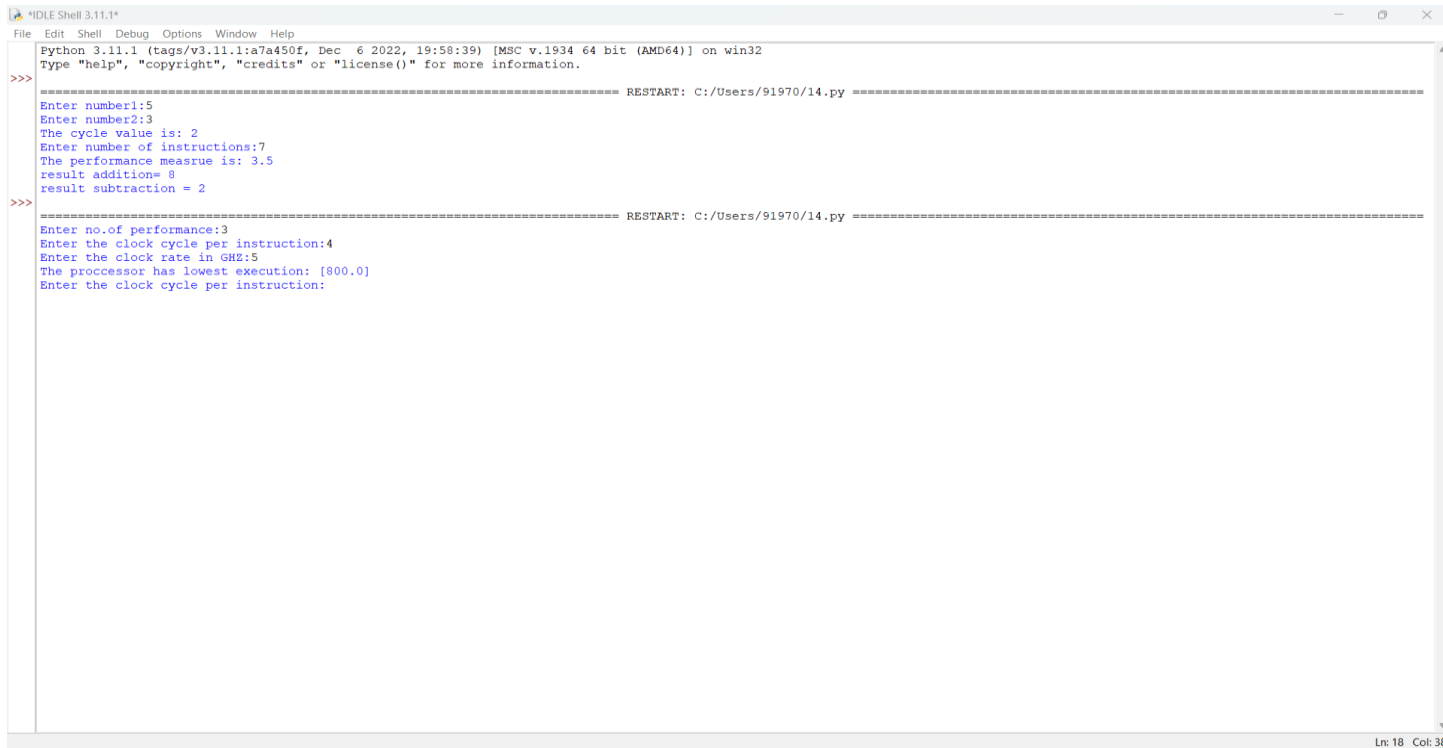
    cr=float(input("Enter the clock rate in GHZ:"))

    a=1000*cp/cr

    ct.append(a)

print("The proccessor has lowest execution:",ct)

```



```

Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/91970/14.py =====
Enter number1:5
Enter number2:3
The cycle value is: 2
Enter number of instructions:7
The performance measrue is: 3.5
result addition= 8
result subtraction = 2
>>>
===== RESTART: C:/Users/91970/14.py =====
Enter no.of performance:3
Enter the clock cycle per instruction:4
Enter the clock rate in GHZ:5
The proccessor has lowest execution: [800.0]
Enter the clock cycle per instruction:

```

19. Write a program to find the Hit ratio for the given number of Hits and Misses in Cache memory using any high level language.

PROGRAM:

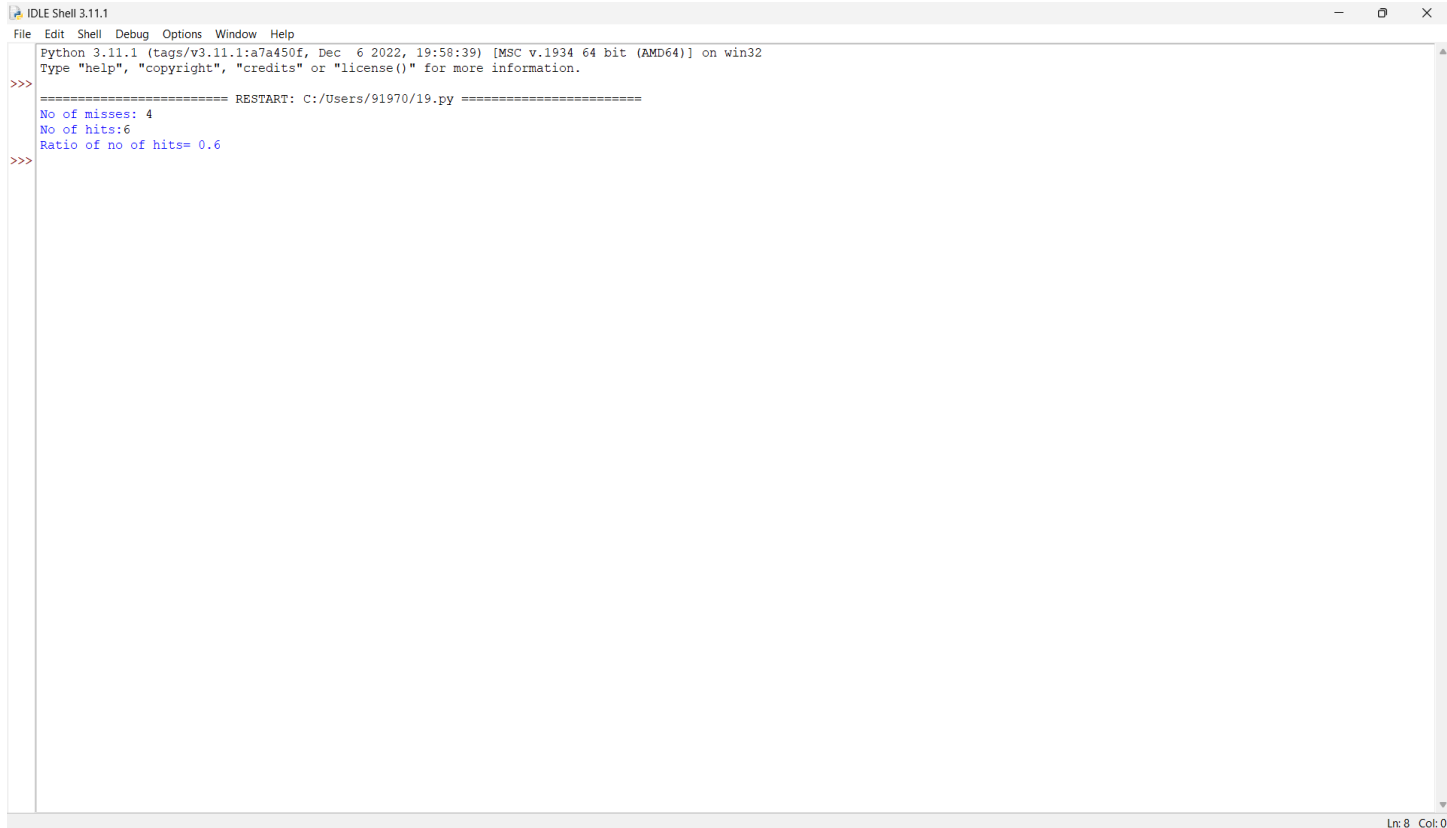
```

m=float(input("No of misses: "))

h=float(input("No of hits:"))

print("Ratio of no of hits=",h/(h+m))

```

A screenshot of an IDLE Shell window titled 'IDLE Shell 3.11.1'. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following output: Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32, Type "help", "copyright", "credits" or "license()" for more information. This is followed by a separator line '===== RESTART: C:/Users/91970/19.py ====='. The output then displays 'No of misses: 4', 'No of hits:6', and 'Ratio of no of hits= 0.6'. The prompt '>>>' is visible at the end of the last line. The status bar at the bottom right indicates 'Ln: 8 Col: 0'.

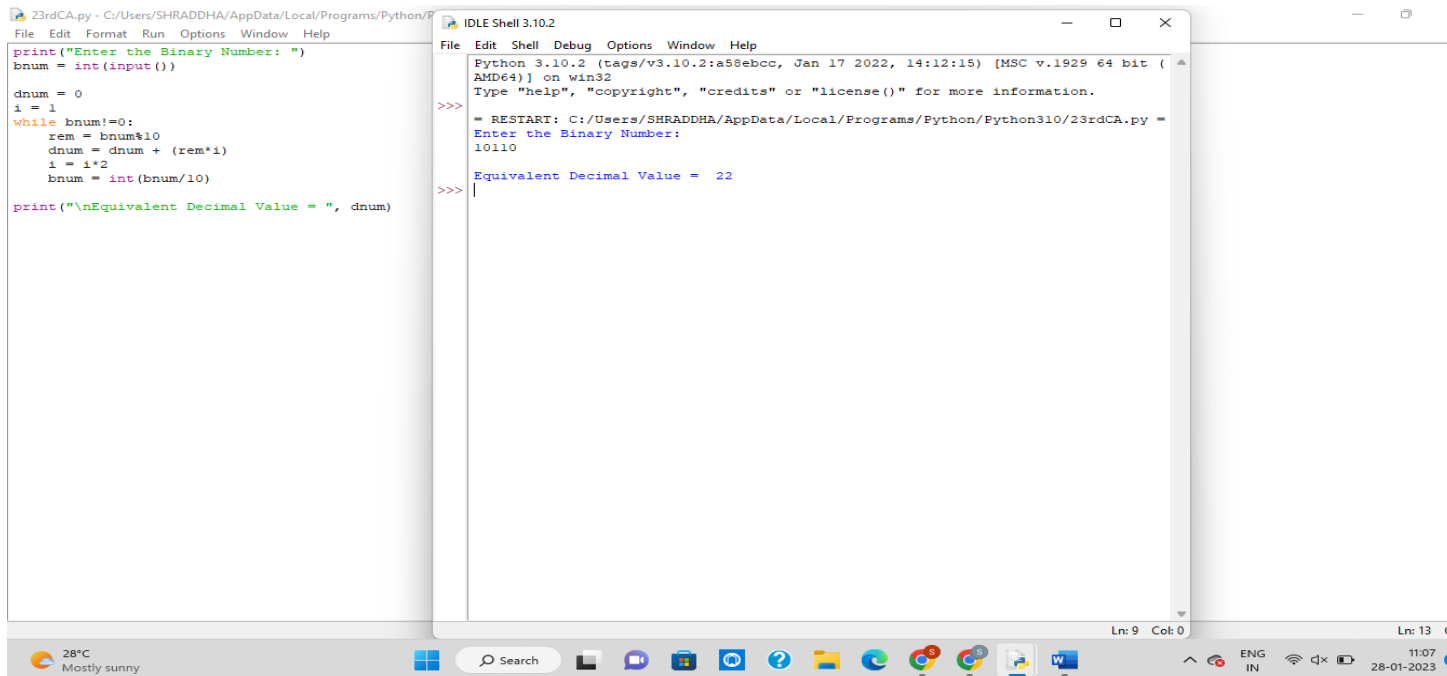
23. Write a program to convert Binary number to Decimal number using any high level language.

PROGRAM:

```
print("Enter the Binary Number: ")
bnum = int(input())

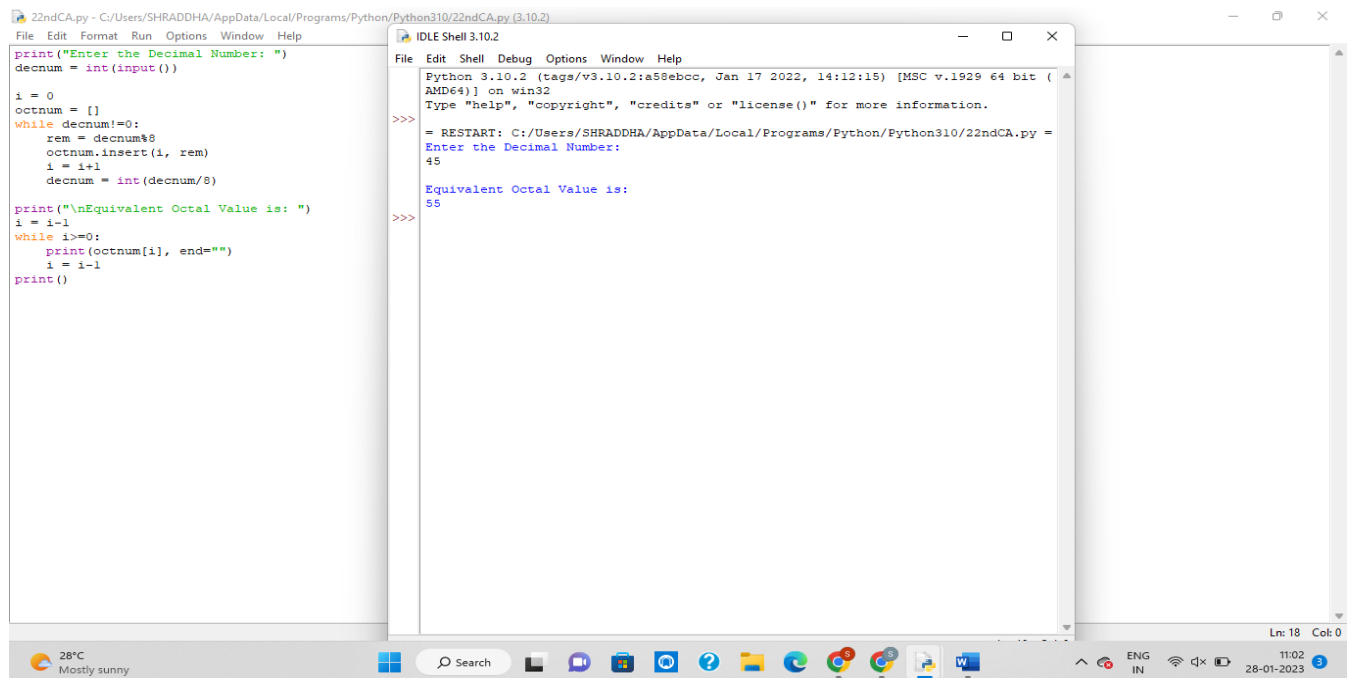
dnum = 0
i = 1
while bnum!=0:
    rem = bnum%10
    dnum = dnum + (rem*i)
    i = i*2
    bnum = int(bnum/10)

print("\nEquivalent Decimal Value = ", dnum)
```



22. Write a program to convert Decimal number to an Octal number using any high level language.
PROGRAM:

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

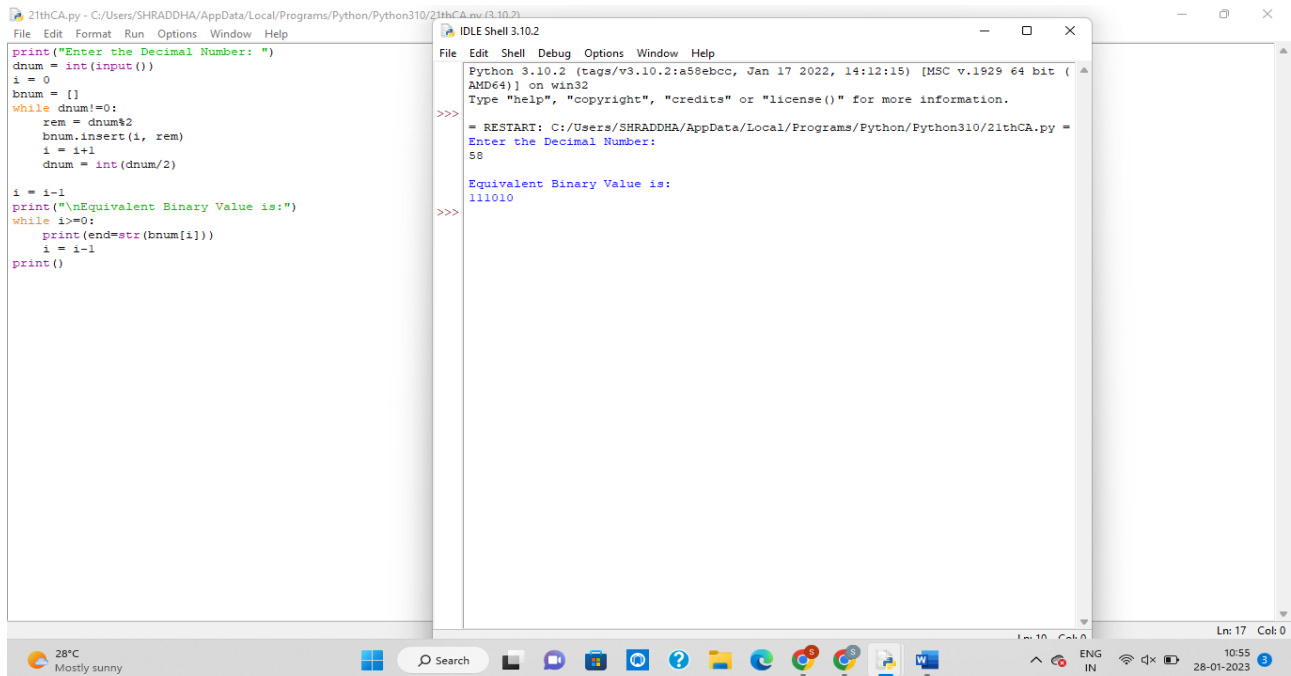


21. Write a program to convert Decimal number to Binary number using any high level language.

PROGRAM:

```
print("Enter the Decimal Number: ")
dnum = int(input())
i = 0
bnum = []
while dnum!=0:
    rem = dnum%2
    bnum.insert(i, rem)
    i = i+1
    dnum = int(dnum/2)

i = i-1
print("\nEquivalent Binary Value is:")
while i>=0:
    print(end=str(bnum[i]))
    i = i-1
print()
```

18. Write a program to perform Restoring Division of two numbers using any high level language.

PROGRAM:

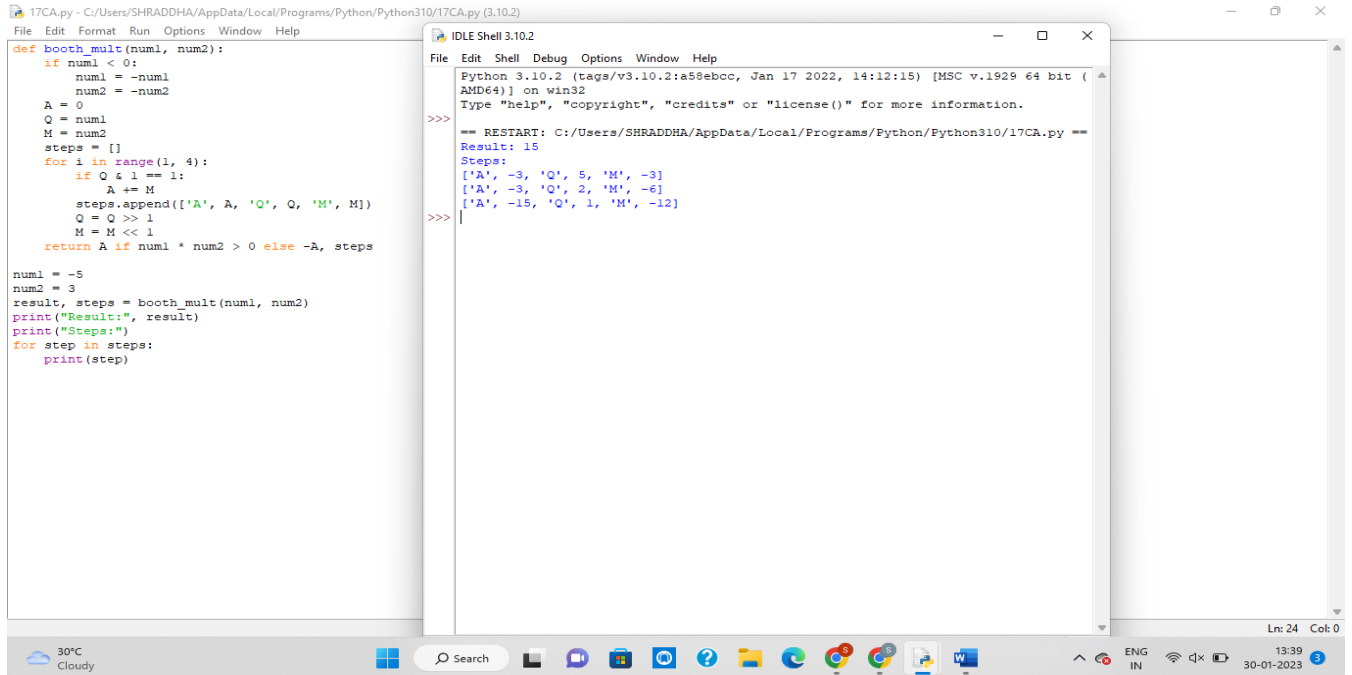
```
def booth_mult(num1, num2):
    if num1 < 0:
        num1 = -num1
        num2 = -num2
    A = 0
    Q = num1
    M = num2
    steps = []
    for i in range(1, 4):
        if Q & 1 == 1:
            A += M
        steps.append(['A', A, 'Q', Q, 'M', M])
        Q = Q >> 1
        M = M << 1
    return A if num1 * num2 > 0 else -A, steps

num1 = -5
num2 = 3
result, steps = booth_mult(num1, num2)
```

```

print("Result:", result)
print("Steps:")
for step in steps:
    print(step)

```



17. Write a program to perform Booth's multiplication of two signed numbers using any high level language

PROGRAM:

```
def restoring_division(dividend, divisor):
```

```

    quotient = 0
    remainder = dividend
    while remainder >= divisor:
        quotient += 1
        remainder -= divisor
    return quotient, remainder

```

```
dividend =int(input("enter number:"))
```

```
divisor =int(input("enter number:"))
```

```
quotient, remainder = restoring_division(dividend, divisor)
print("Quotient: ", quotient)
print("Remainder: ", remainder)
```