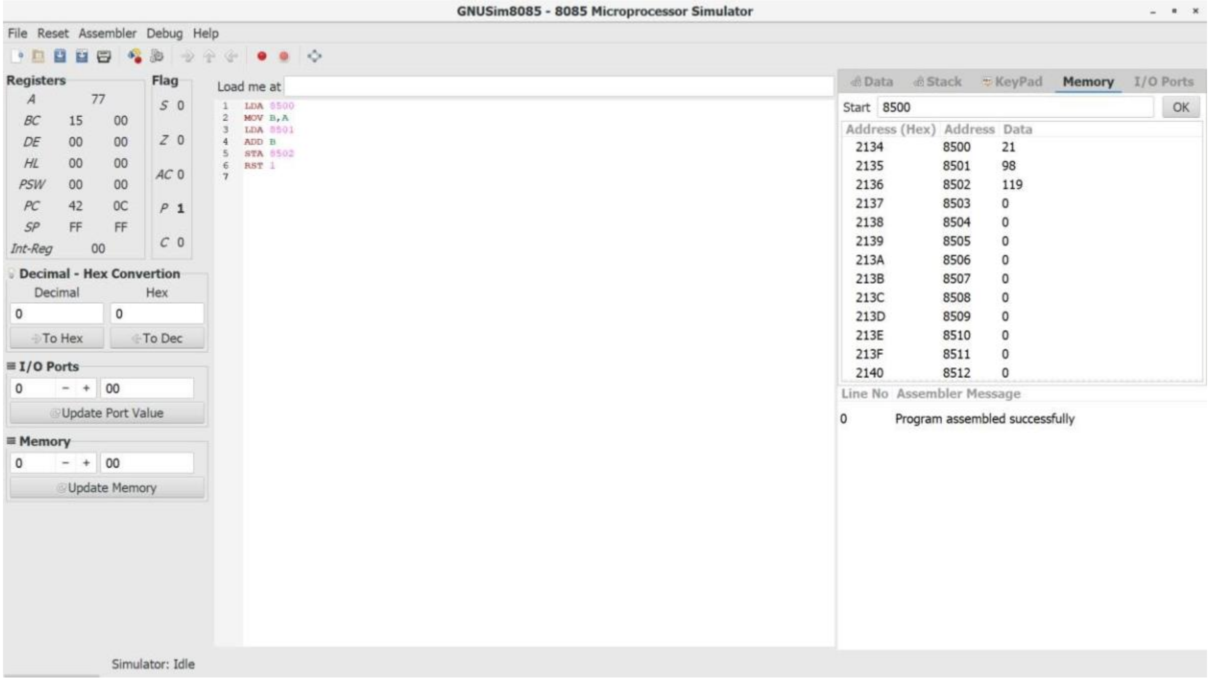
K.Rakesh
192011120

8-BIT ADDITION PROGRAM:

```
LDA 8500
MOV B,A
LDA 8501
ADD B
STA 8502
RST 1
```

OUTPUT:



Registers

A	77	
BC	15	00
DE	00	00
HL	00	00
PSW	00	00
PC	42	0C
SP	FF	FF
Int-Reg	00	

Flag

S	0
Z	0
AC	0
P	1
C	0

```
LDA 8500
MOV B,A
LDA 8501
ADD B
STA 8502
RST 1
```

Address (Hex)	Address	Data
2134	8500	21
2135	8501	98
2136	8502	119
2137	8503	0
2138	8504	0
2139	8505	0
213A	8506	0
213B	8507	0
213C	8508	0
213D	8509	0
213E	8510	0
213F	8511	0
2140	8512	0

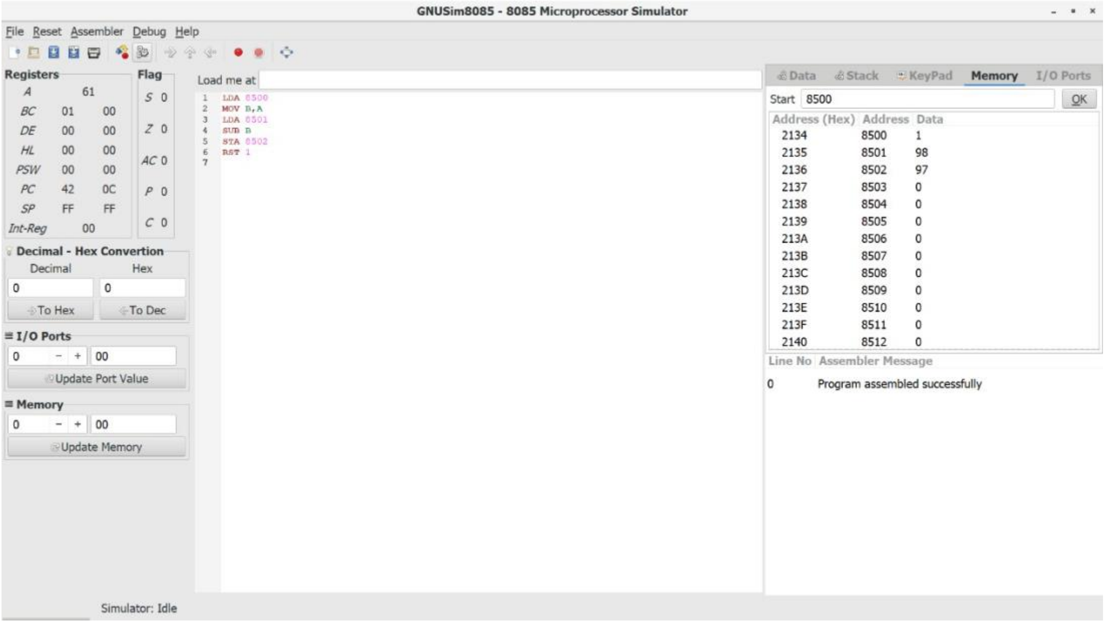
Line No	Assembler Message
0	Program assembled successfully

8-BIT SUBTRACTION

PROGRAM:

```
LDA 8500
MOV B,A
LDA 8501
SUB B
STA 8502
RST 1
```

OUTPUT:



Registers

A	61	
BC	01	00
DE	00	00
HL	00	00
PSW	00	00
PC	42	0C
SP	FF	FF
Int-Reg	00	

Flag

S	0
Z	0
AC	0
P	0
C	0

Load me at

```
LDA 8500
MOV B,A
LDA 8501
SUB B
STA 8502
RST 1
```

Start 8500

Address (Hex)	Address	Data
2134	8500	1
2135	8501	98
2136	8502	97
2137	8503	0
2138	8504	0
2139	8505	0
213A	8506	0
213B	8507	0
213C	8508	0
213D	8509	0
213E	8510	0
213F	8511	0
2140	8512	0

Line No	Assembler Message
0	Program assembled successfully

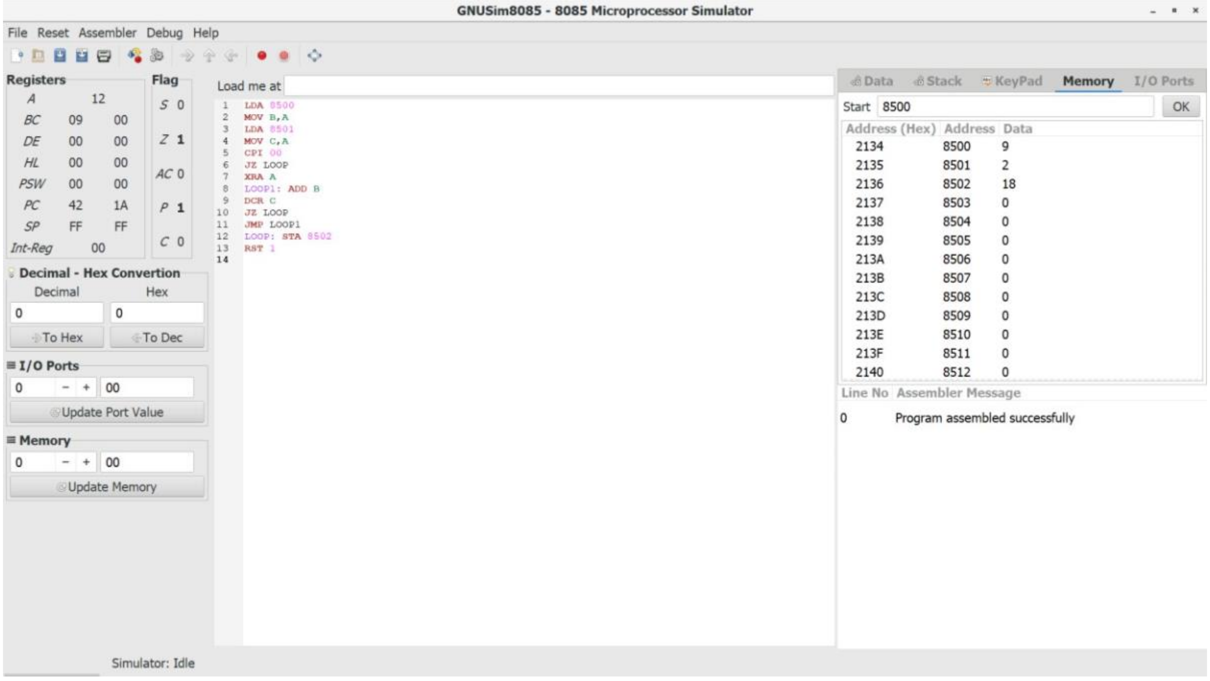
Simulator: Idle

8-BIT MULTIPLICATION

PROGRAM:

```
LDA 8500
MOV B,A
LDA 8501
MOV C,A
CPI 00
JZ LOOP
XRA A
LOOP1: ADD B
DCR C
JZ LOOP
JMP LOOP1
LOOP: STA 8502
RST 1
```

OUTPUT:



Registers		
A	12	
BC	09	00
DE	00	00
HL	00	00
PSW	00	00
PC	42	1A
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	1
AC	0
P	1
C	0

Decimal	Hex
0	0

I/O Ports: 0 − + 00

Memory: 0 − + 00

```
LDA 8500
MOV B,A
LDA 8501
MOV C,A
CPI 00
JZ LOOP
XRA A
LOOP1: ADD B
DCR C
JZ LOOP
JMP LOOP1
LOOP: STA 8502
RST 1
```

Start 8500

Address (Hex)	Address	Data
2134	8500	9
2135	8501	2
2136	8502	18
2137	8503	0
2138	8504	0
2139	8505	0
213A	8506	0
213B	8507	0
213C	8508	0
213D	8509	0
213E	8510	0
213F	8511	0
2140	8512	0

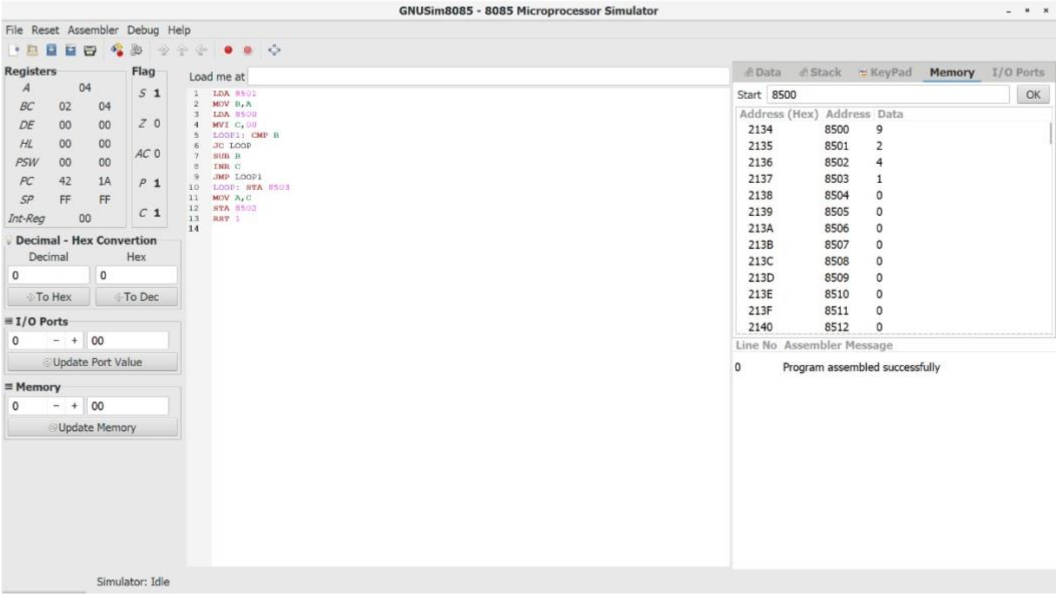
Line No	Assembler Message
0	Program assembled successfully

PROGRAM:

8-BIT DIVISION

```
LDA 8501
MOV B,A
LDA 8500
MVI C,00
LOOP1: CMP B
JC LOOP
SUB B
INR C
JMP LOOP1
LOOP: STA 8503
MOV A,C
STA 8502
RST 1
```

OUTPUT:



Registers		
A	04	
BC	02	04
DE	00	00
HL	00	00
PSW	00	00
PC	42	1A
SP	FF	FF
Int-Reg	00	

Flag	
S	1
Z	0
AC	0
P	1
C	1

Decimal: 0 Hex: 0

I/O Ports: 0 00

Memory: 0 00

```
LDA 8501
MOV B,A
LDA 8500
MVI C,00
LOOP1: CMP B
JC LOOP
SUB B
INR C
JMP LOOP1
LOOP: STA 8503
MOV A,C
STA 8502
RST 1
```

Start 8500

Address (Hex)	Address	Data
2134	8500	9
2135	8501	2
2136	8502	4
2137	8503	1
2138	8504	0
2139	8505	0
213A	8506	0
213B	8507	0
213C	8508	0
213D	8509	0
213E	8510	0
213F	8511	0
2140	8512	0

Line No	Assembler Message
0	Program assembled successfully

PROGRAM:

counter=1

a=int(input("ENTER NUMBER

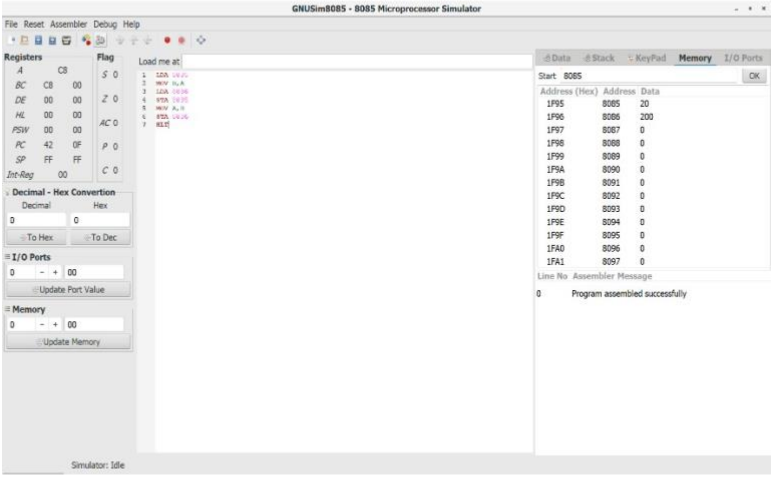
SWAPPING OF TWO NUMBERS

PROGRAM:

```
LDA 8085
MOV B,A
LDA 8086
STA 8085
MOV A,B
STA 8086
HLT
```

OUTPUT:

(BEFORE SWAP)



Registers		
A	C8	
BC	C8	00
DE	00	00
HL	00	00
PSW	00	00
PC	42	0F
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	0
AC	0
P	0
C	0

```
LDA 8085
MOV B,A
LDA 8086
STA 8085
MOV A,B
STA 8086
HLT
```

Decimal: 0 Hex: 0

I/O Ports: 0 - + 00

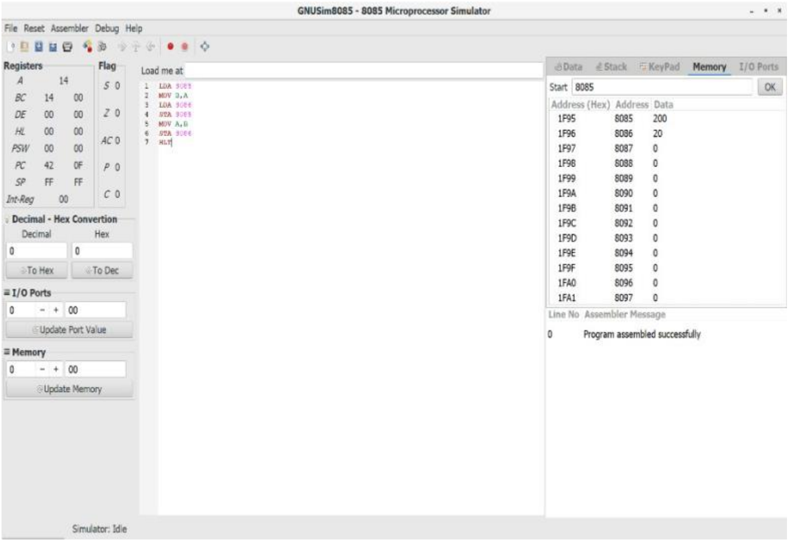
Memory: 0 - + 00

Start: 8085

Address (Hex)	Address	Data
1F95	8085	20
1F96	8086	200
1F97	8087	0
1F98	8088	0
1F99	8089	0
1F9A	8090	0
1F9B	8091	0
1F9C	8092	0
1F9D	8093	0
1F9E	8094	0
1F9F	8095	0
1FA0	8096	0
1FA1	8097	0

Line No	Assembler Message
0	Program assembled successfully

(AFTER SWAP)



Registers		
A	14	
BC	14	00
DE	00	00
HL	00	00
PSW	00	00
PC	42	0F
SP	FF	FF
Int-Reg	00	

Flag	
S	0
Z	0
AC	0
P	0
C	0

```
LDA 8085
MOV B,A
LDA 8086
STA 8085
MOV A,B
STA 8086
HLT
```

Address (Hex)	Address	Data
1F95	8085	200
1F96	8086	20
1F97	8087	0
1F98	8088	0
1F99	8089	0
1F9A	8090	0
1F9B	8091	0
1F9C	8092	0
1F9D	8093	0
1F9E	8094	0
1F9F	8095	0
1FA0	8096	0
1FA1	8097	0

Line No	Assembler Message
0	Program assembled successfully

TWO STAGE PIPELINING

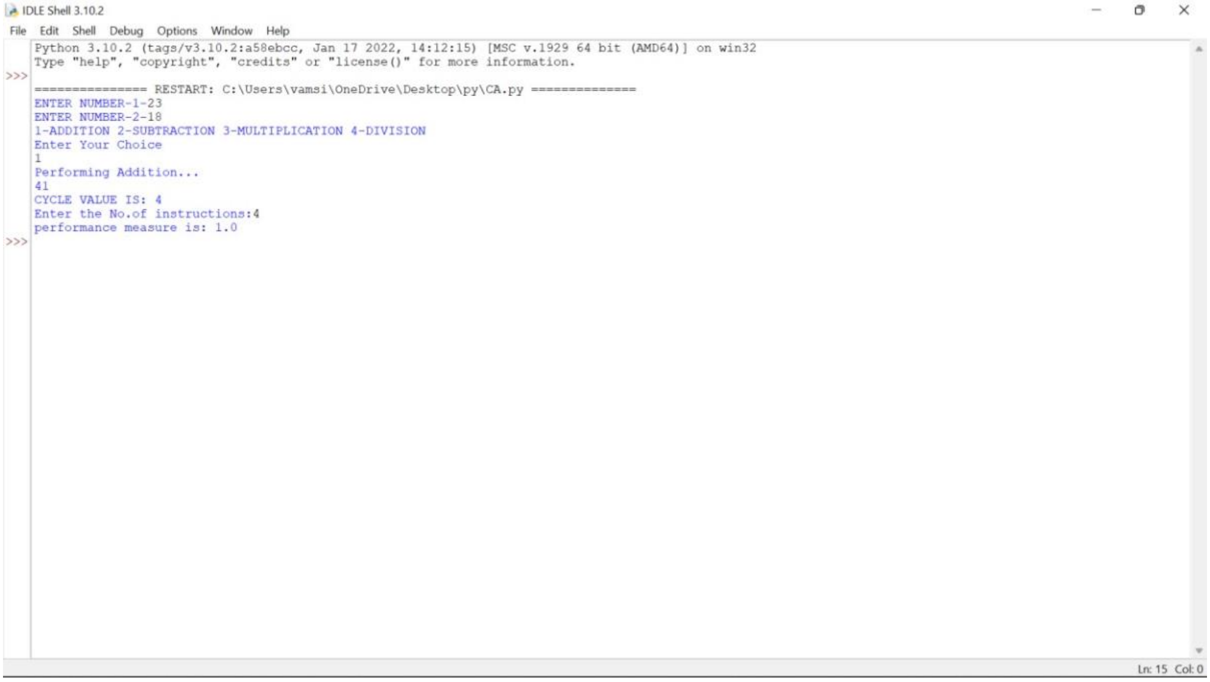
```
-1-"))
counter=counter+1 b=int(input("ENTER
NUMBER-2-")) counter=counter+1
print("1-ADDITION 2-SUBTRACTION 3-MULTIPLICATION 4-DIVISION")
print("Enter Your Choice") choice=int(input()) if choice==1:
print("Performing Addition...")     res=a+b    counter=counter+1 if choice==2:
print("Performing Subtraction...")     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)
```

PROGRAM:

counter=1

a=int(input("ENTER NUMBER

OUTPUT:



```
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=============== RESTART: C:\Users\vamsi\OneDrive\Desktop\py\CA.py ==============
ENTER NUMBER-1-23
ENTER NUMBER-2-18
1-ADDITION 2-SUBTRACTION 3-MULTIPLICATION 4-DIVISION
Enter Your Choice
1
Performing Addition...
41
CYCLE VALUE IS: 4
Enter the No.of instructions:4
performance measure is: 1.0
>>>
```

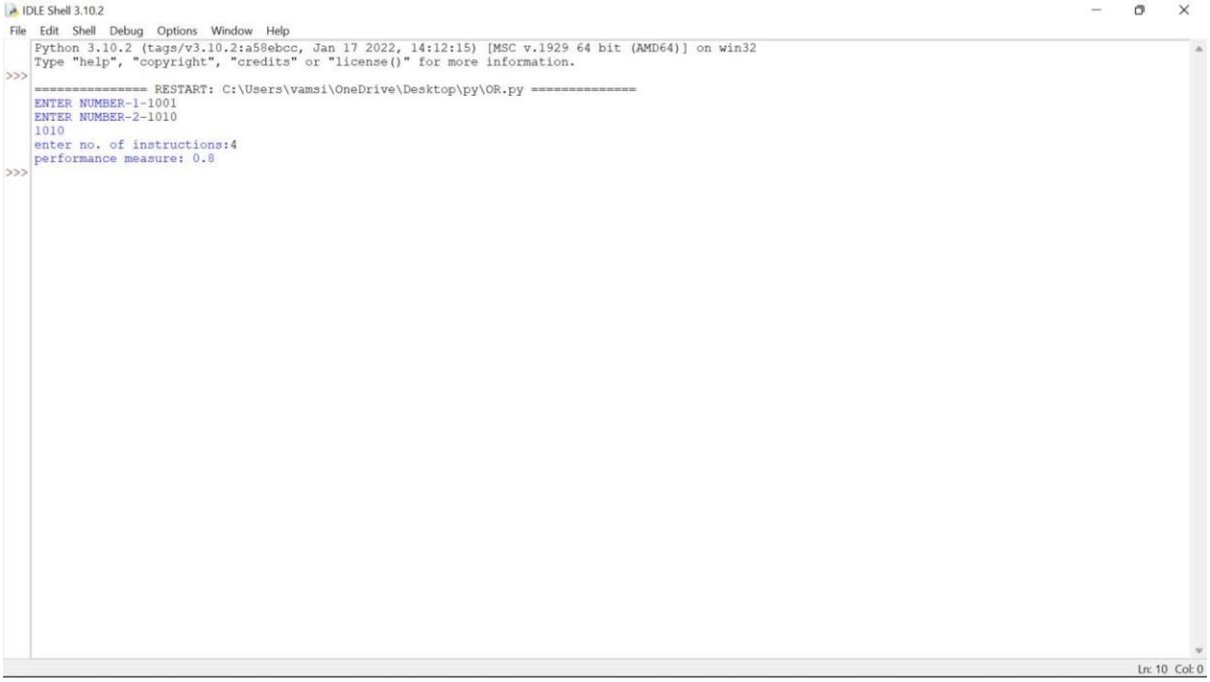
THREE STAGE PIPELINE (AND)

-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:



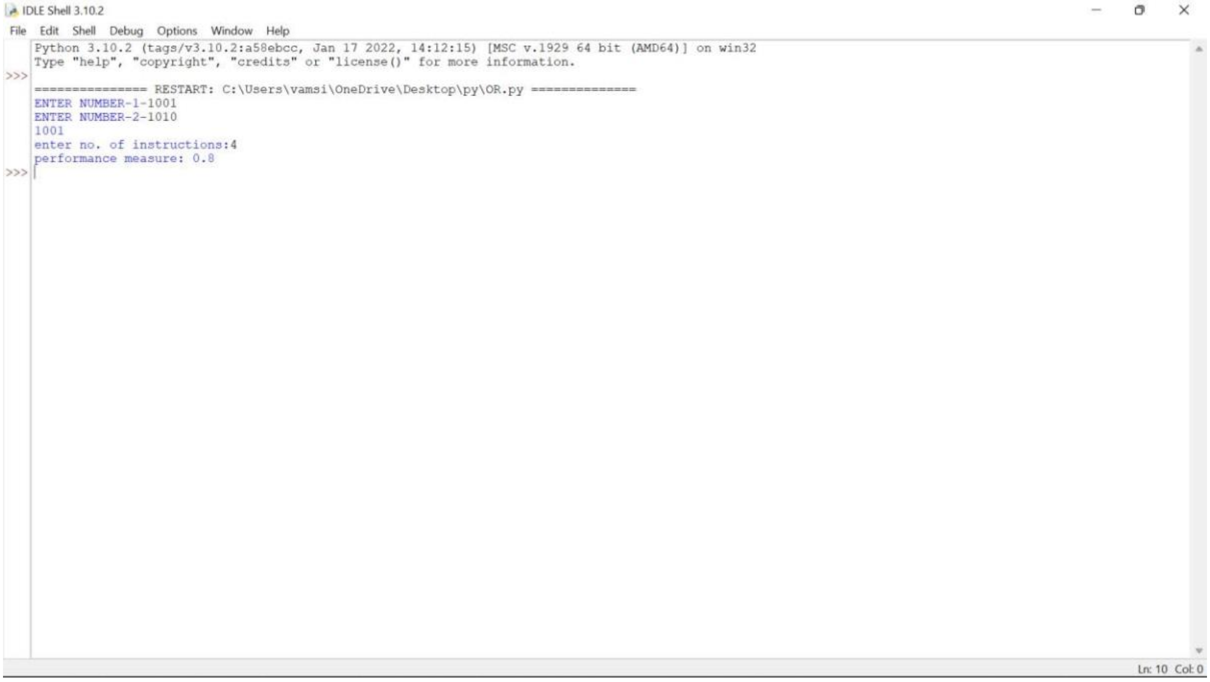
```
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: C:\Users\vamsi\OneDrive\Desktop\py\OR.py ==============
ENTER NUMBER-1-1001
ENTER NUMBER-2-1010
1010
enter no. of instructions:4
performance measure: 0.8
>>>
```

THREE STAGE PIPELINE (OR)

PROGRAM:

```
counter=1
a=int(input("ENTER NUMBER-1-"))
counter=counter+1
b=int(input("ENTER NUMBER-2-"))
counter=counter+1 res= a or 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:



```
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=============== RESTART: C:\Users\vamsi\OneDrive\Desktop\py\OR.py ==============
ENTER NUMBER-1-1001
ENTER NUMBER-2-1010
1001
enter no. of instructions:4
performance measure: 0.8
>>>
```

FOUR STAGE PIPELINE

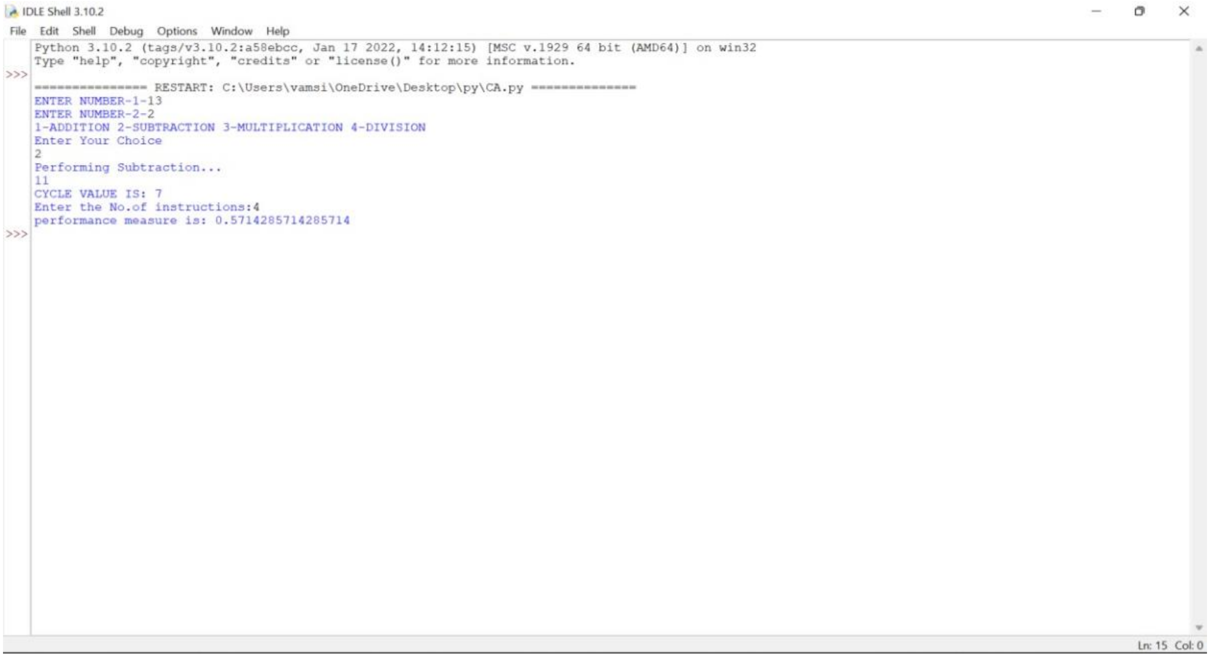
```
                                          -1-"))
counter=counter+1 b=int(input("ENTER
NUMBER-2-")) counter=counter+1
print("1-ADDITION 2-SUBTRACTION 3-MULTIPLICATION 4-DIVISION")
print("Enter Your Choice") choice=int(input()) if choice==1:
print("Performing Addition...")   res=a+b   counter=counter+1 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:

PROGRAM:

```
counter=1

a=int(input("ENTER NUMBER
```



```
    Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more information.
>>>
    =============== RESTART: C:\Users\vamsi\OneDrive\Desktop\py\CA.py ==============
    ENTER NUMBER-1-13
    ENTER NUMBER-2-2
    1-ADDITION 2-SUBTRACTION 3-MULTIPLICATION 4-DIVISION
    Enter Your Choice
    2
    Performing Subtraction...
    11
    CYCLE VALUE IS: 7
    Enter the No.of instructions:4
    performance measure is: 0.5714285714285714
>>>
```

PROGRAM:

16 BIT ADDITION(8086)

```
MOV SI,1200H

LODSW

MOV BX,AX

LODSW

ADD BX,AX

MOV DI,1300H

MOV [DI],BX

HLT
```

INPUT:

1200	13H	1201	13H
1202	14H	1203	14H

OUTPUT:

1300	27H
1301	27H

16 BIT SUBTRACTION

PROGRAM:

```
MOV SI,1200H

LODSW

MOV BX,AX

LODSW

SUB BX,AX

MOV DI,1300H

MOV [DI],BX

HLT
```

INPUT:

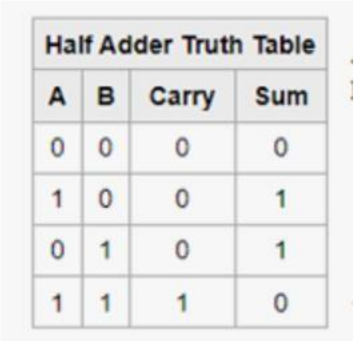
1200	08H	1201	08H
1202	04H	1203	04H

OUTPUT:

1300	04H
1301	04H

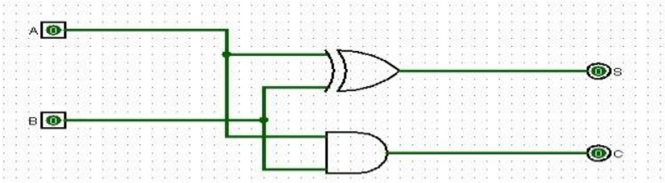
HALF ADDER

TRUTH TABLE:

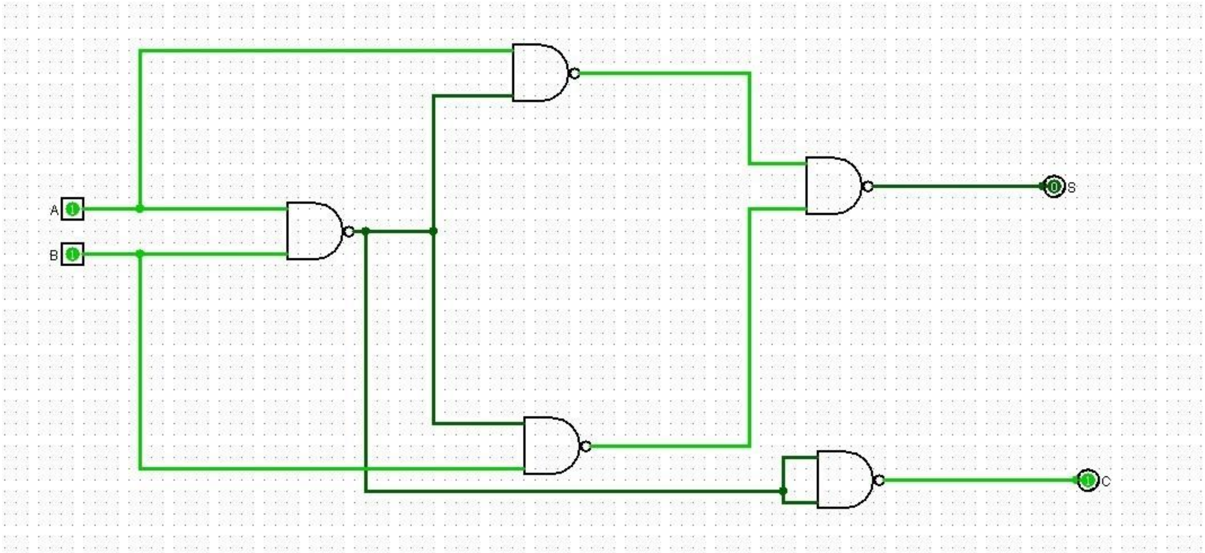


Half Adder Truth Table			
A	B	Carry	Sum
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

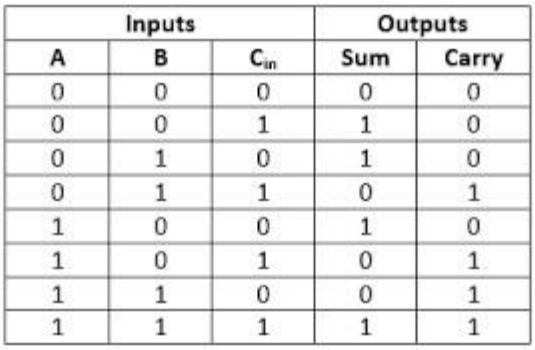
LOGISM CIRCUIT DIAGRAM:



LOGISM CIRCUIT DIAGRAM (USING NAND GATES)

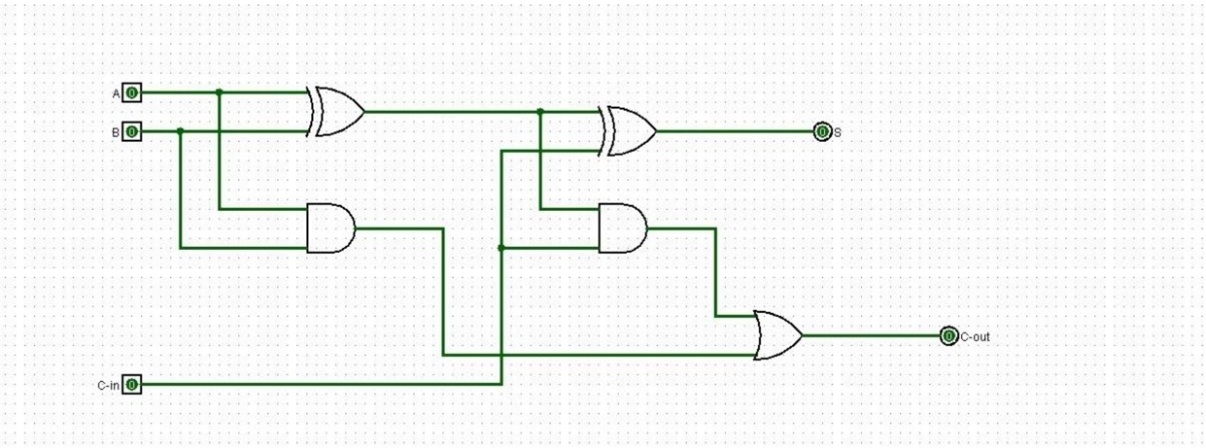


FULL ADDER TRUTH TABLE:



Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

LOGISM CIRCUIT DIAGRAM:



LOGISM CIRCUIT DIAGRAM (USING NAND GATE):