CSA PRACTICAL FILE

RAMANUJAN COLLEGE

DSC 02: COMPUTER SYSTEM ARCHITECTURE

SEMESTER-1

(2025-26)

Submitted By:-

Name: ***Aditya Kumar Jha***

College Roll No: ***25570003***

University Roll No: ***25020570033***

Course: ***B.Sc. (Hons) Computer Science***

Submitted To:-

Dr. Kamlesh Kumar Raghuvanshi

Department of Computer Science

Acknowledgement

I would like to take this opportunity to acknowledge everyone who has helped us in every stage of this project.

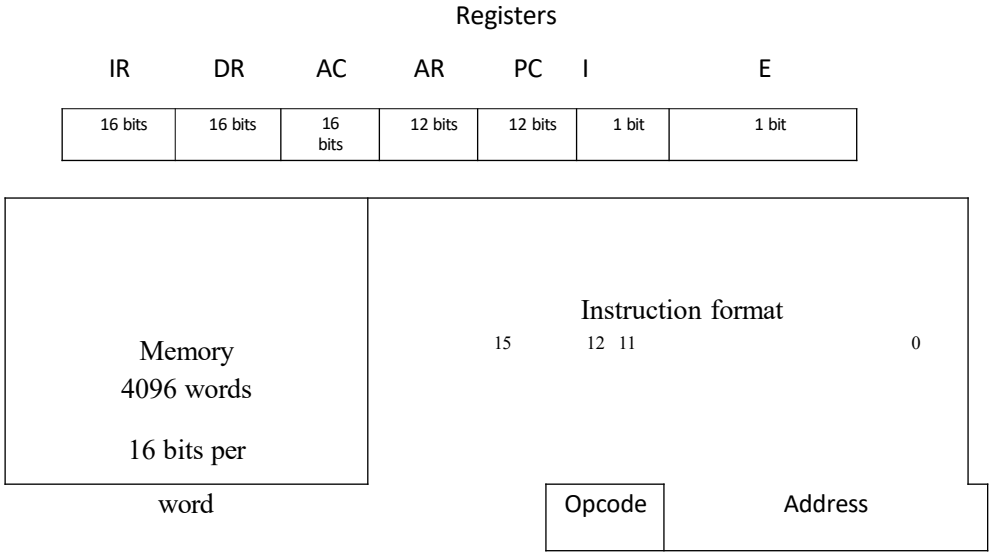
I am deeply indebted to my computer system architecture professor, **Dr Kamlesh Kumar Raghuvanshi** for his guidance and suggestions in completing this project. The completion of this project was possible under his guidance and support.

I am also very thankful to my parents and my friends who have boosted me up morally with their continuous support.

At last but not least, I am very thankful to God almighty for showering his blessings upon me.

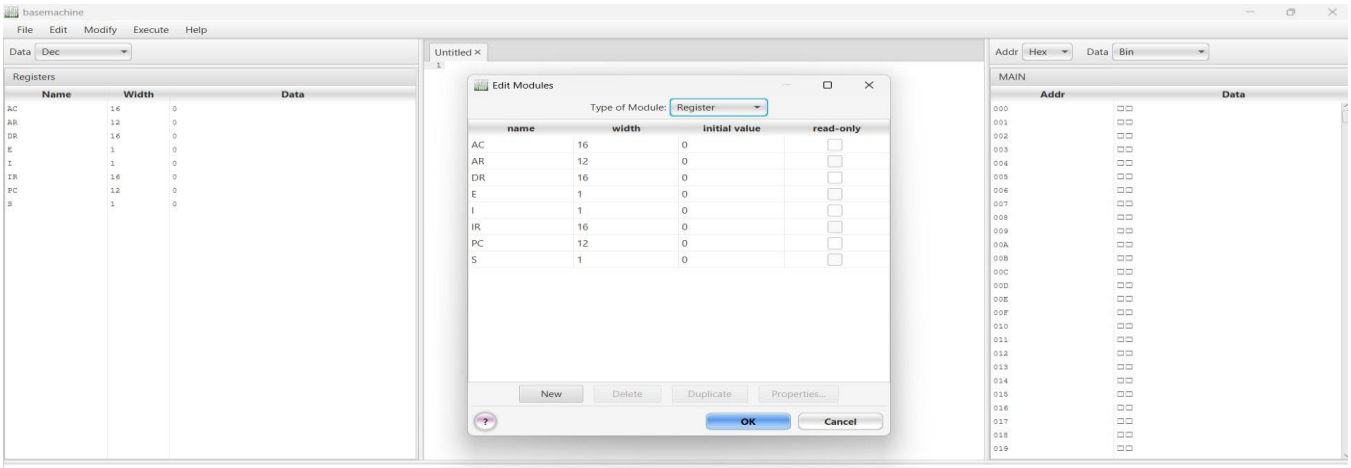
INDEX

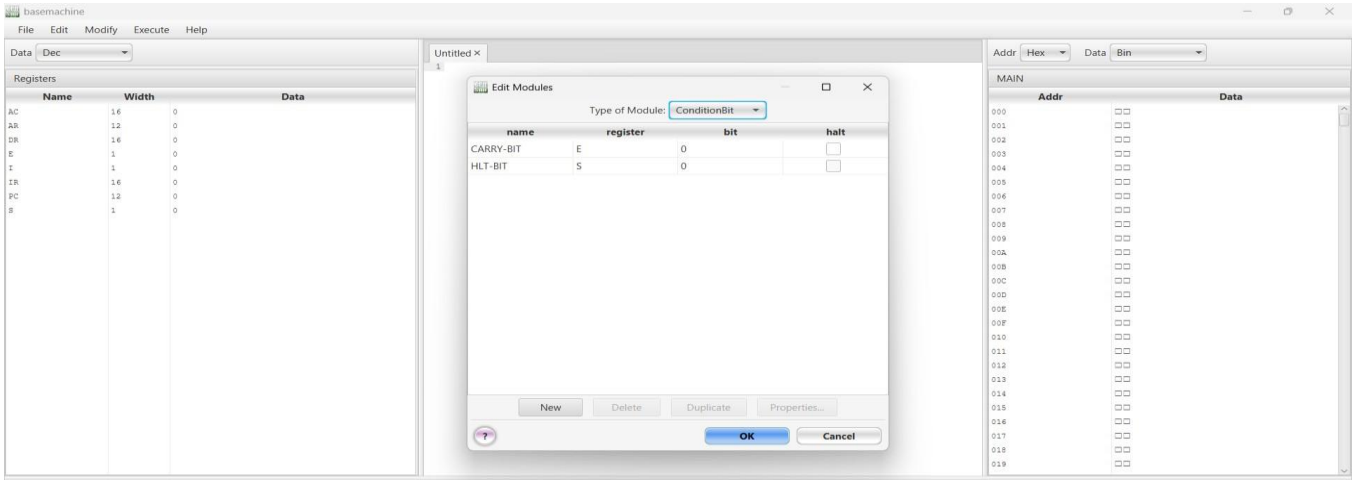
Q.1. Create a machine based on the following architecture:

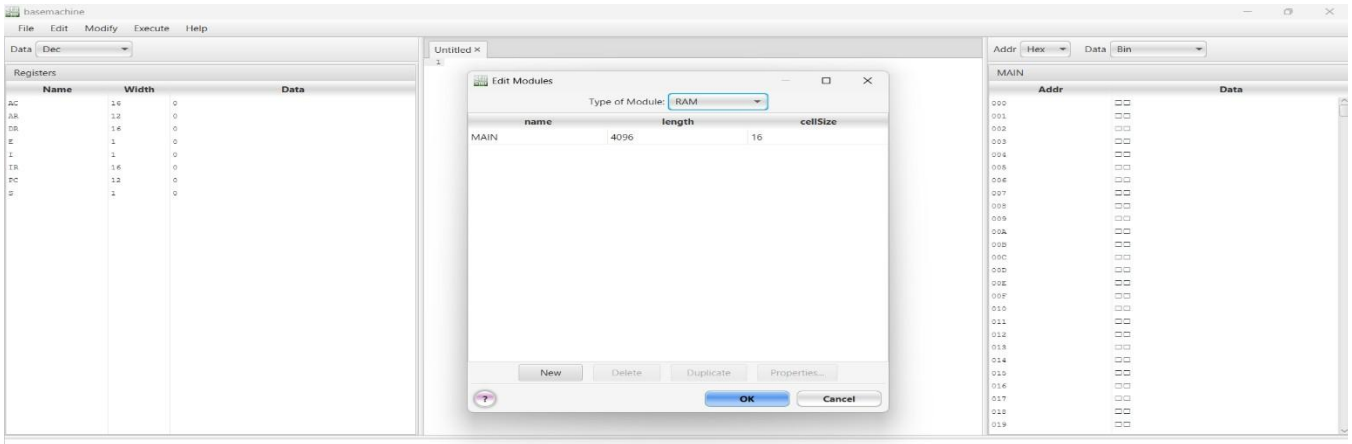


Basic Computer Instructions

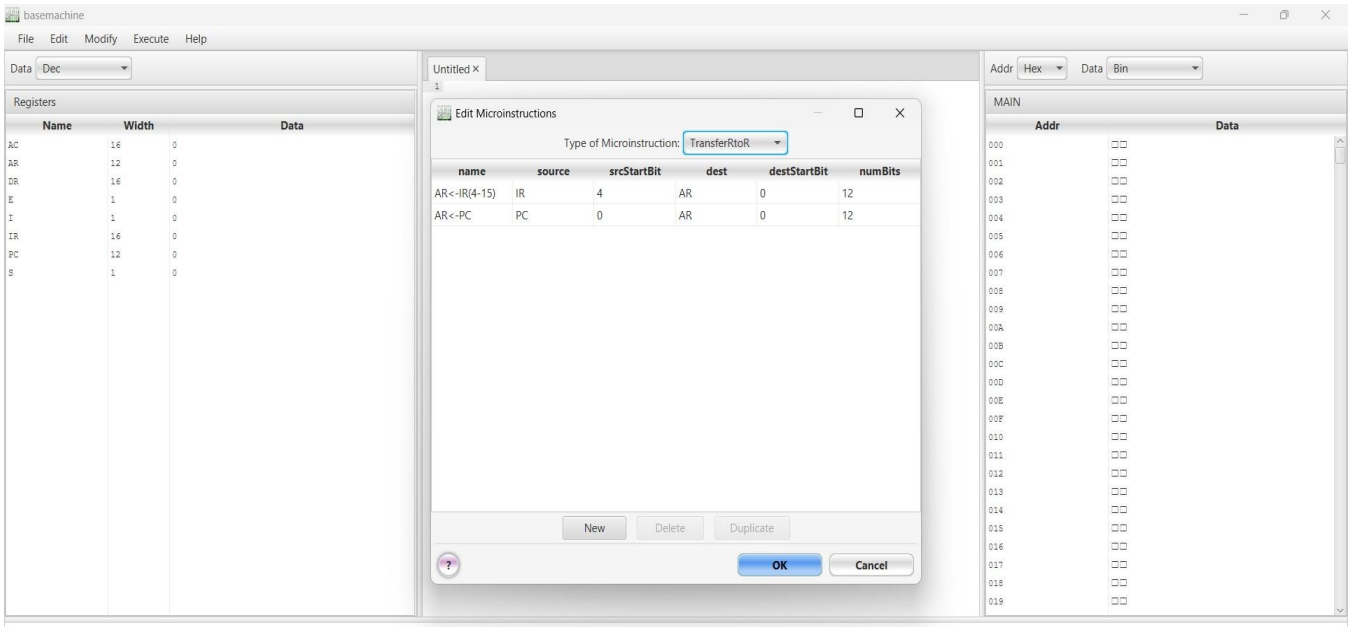
Memory Reference			Register Reference	
Symbol	Hex		Symbol	Hex
AND	0xxx	Direct Addressing	CLA	7800
ADD	1xxx		CLE	7400
LDA	2xxx		CMA	7200
STA	3xxx		CME	7100
BUN	4xxx		CIR	7080
BSA	5xxx		CIL	7040
ISZ	6xxx		INC	7020
AND_I	8xxx	Indirect Addressing	SPA	7010
ADD_I	9xxx		SNA	7008
LDA_I	Axxx		SZA	7004
STA_I	Bxxx		SZE	7002
BUN_I	Cxxx		HLT	7001
BSA_I	Dxxx		INP	F800
ISZ_I	Exxx		OUT	F400

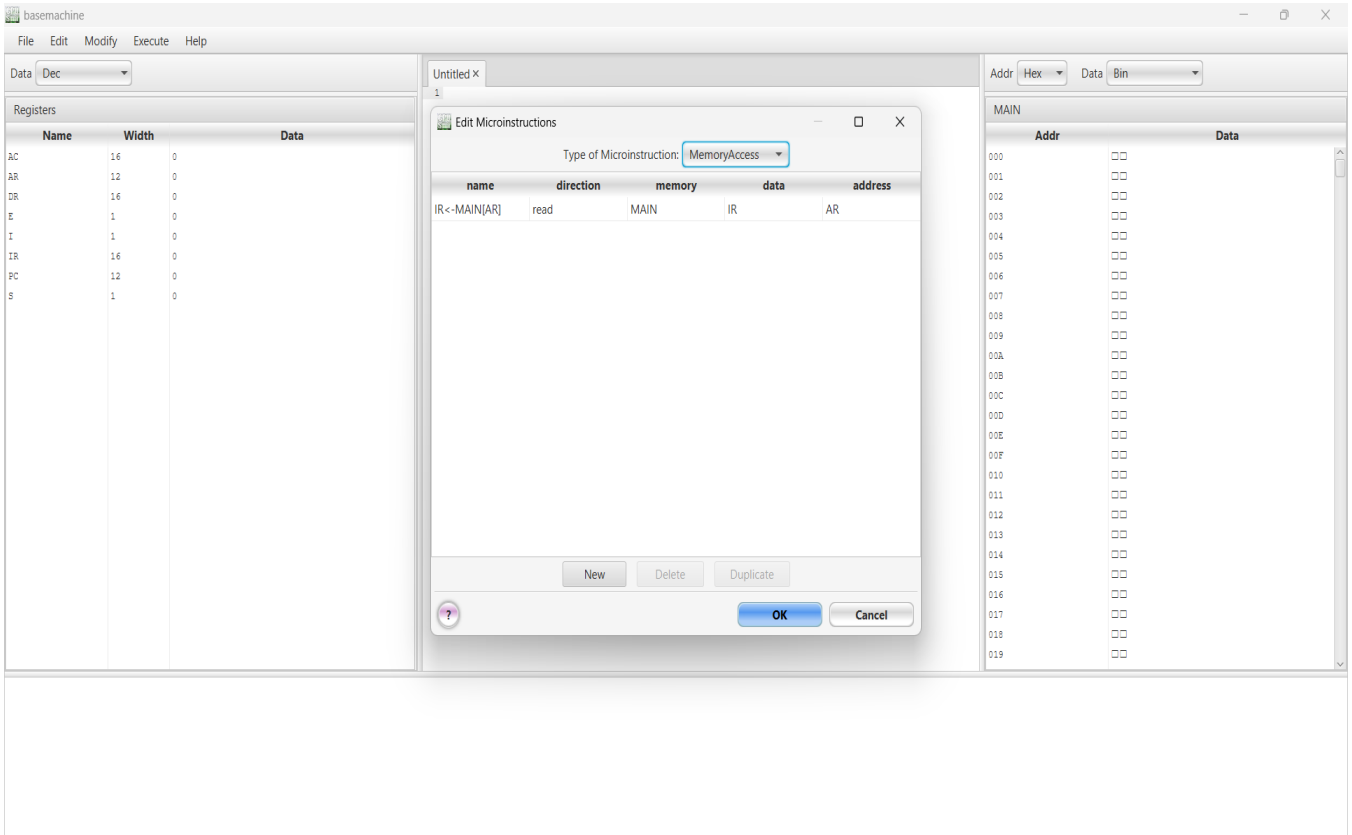

basemachine
File Edit Modify Execute Help
Data Dec
Registers
Name Width Data
AC 16 0
AR 12 0
DR 16 0
E 1 0
I 1 0
IR 16 0
PC 12 0
S 1 0
Untitled ×
Edit Modules
Type of Module: Register
name width initial value read-only
AC 16 0
AR 12 0
DR 16 0
E 1 0
I 1 0
IR 16 0
PC 12 0
S 1 0
New Delete Duplicate Properties...
OK Cancel
Addr Hex Data Bin
MAIN
Addr Data

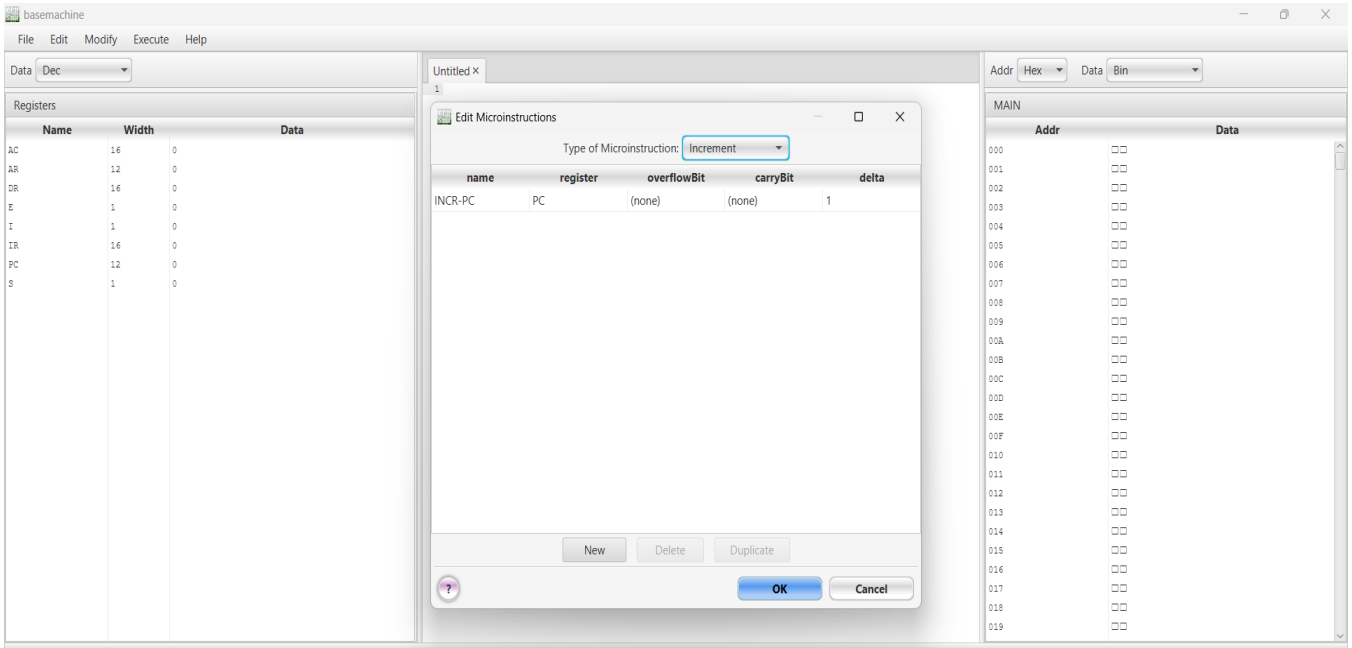

basemachine
File Edit Modify Execute Help
Data Dec
Registers
Name Width Data
Untitled ×
Edit Modules
Type of Module: ConditionBit
name register bit halt
CARRY-BIT E 0
HLT-BIT S 0
New Delete Duplicate Properties...
OK Cancel
Addr Hex Data Bin
MAIN
Addr Data

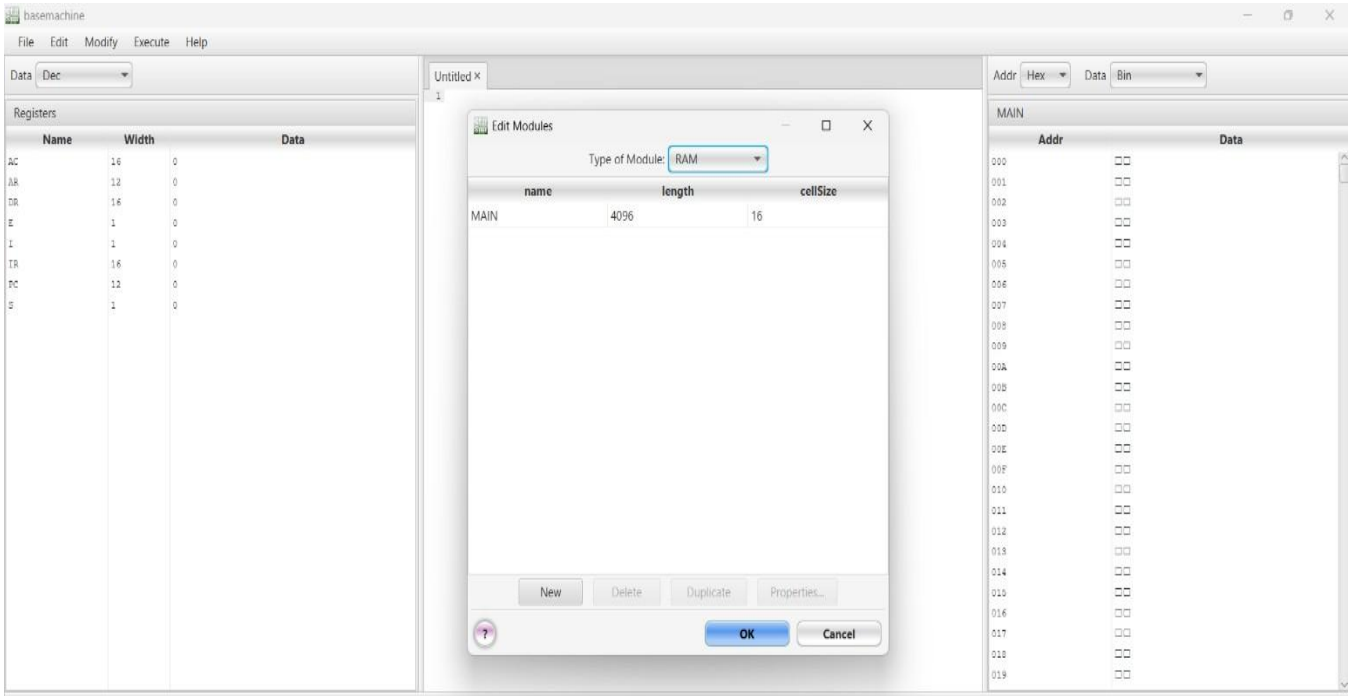

basemachine
File Edit Modify Execute Help
Data Dec
Registers
Name Width Data
Untitled ×
Edit Modules
Type of Module: RAM
name length cellSize
MAIN 4096 16
New Delete Duplicate Properties...
OK Cancel
Addr Hex Data Bin
MAIN
Addr Data

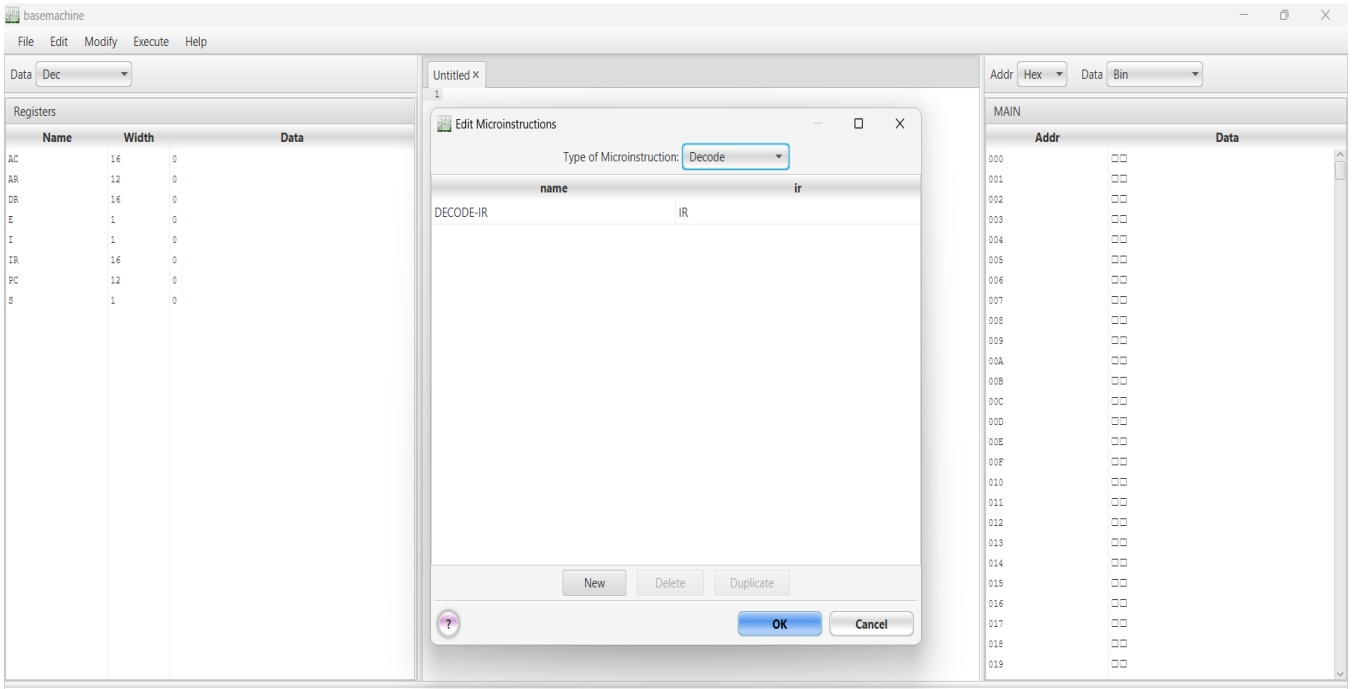
Q2. Creating a FETCH Routine for the Instruction Cycle.

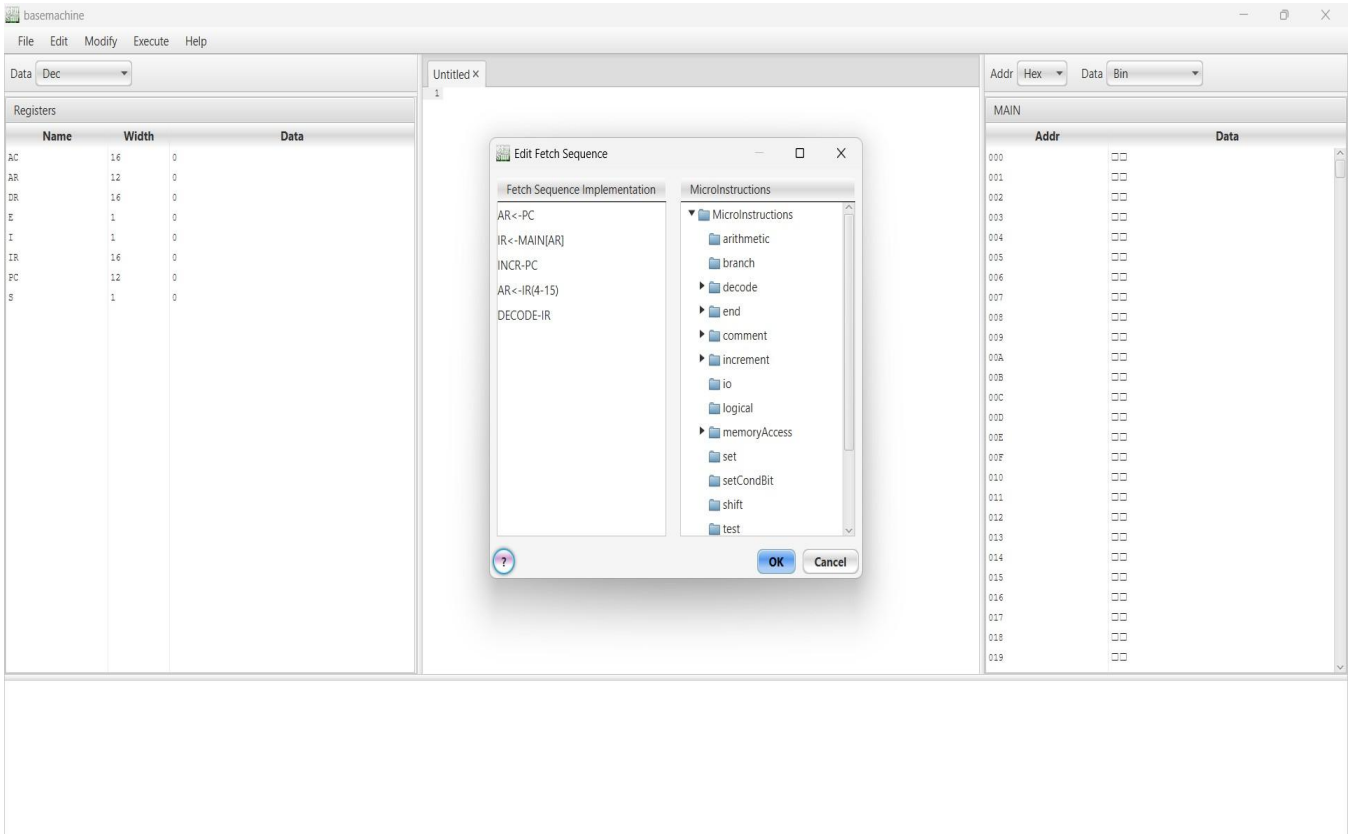




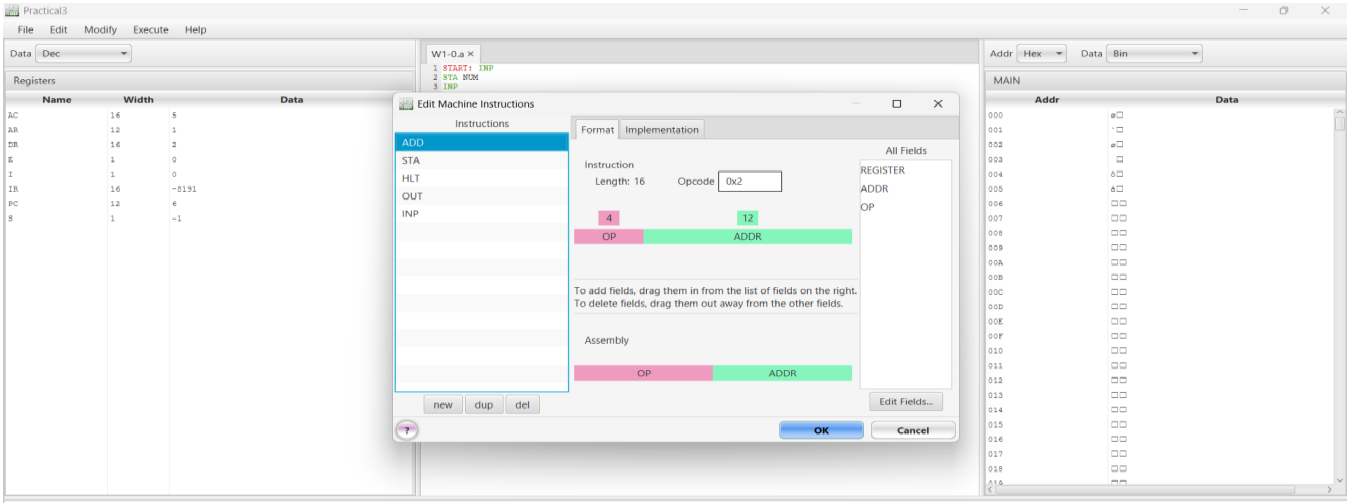

basemachine
File Edit Modify Execute Help
Data Dec
Registers
Name Width Data
AC 16 0
AR 12 0
DR 16 0
E 1 0
I 1 0
IR 16 0
PC 12 0
S 1 0
Untitled ×
1
Edit Microinstructions
Type of Microinstruction: Increment
name register overflowBit carryBit delta
INCR-PC PC (none) (none) 1
New Delete Duplicate
OK Cancel
Addr Hex Data Bin
MAIN
Addr Data

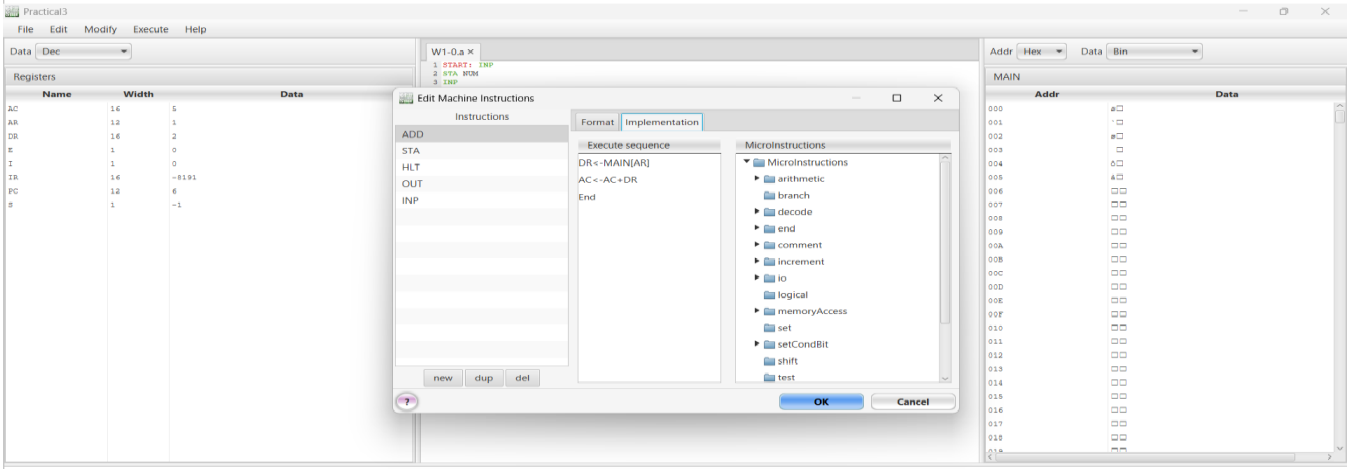

basemachine
File Edit Modify Execute Help
Data Dec
Registers
Name Width Data
Untitled ×
1
Edit Modules
Type of Module: RAM
name length cellSize
MAIN 4096 16
New Delete Duplicate Properties...
OK Cancel
Addr Hex Data Bin
MAIN
Addr Data

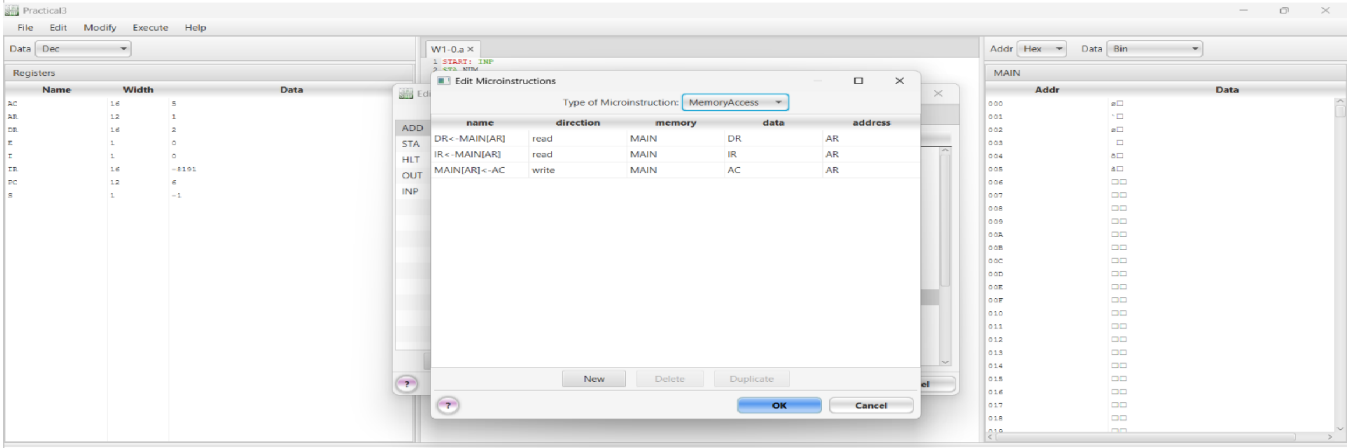

basemachine
File Edit Modify Execute Help
Data Dec
Registers
Name Width Data
AC 16 0
AR 12 0
DR 16 0
E 1 0
I 1 0
IR 16 0
PC 12 0
S 1 0
Untitled ×
1
Edit Microinstructions
Type of Microinstruction: Decode
name ir
DECODE-IR IR
New Delete Duplicate
OK Cancel
Addr Hex Data Bin
MAIN
Addr Data

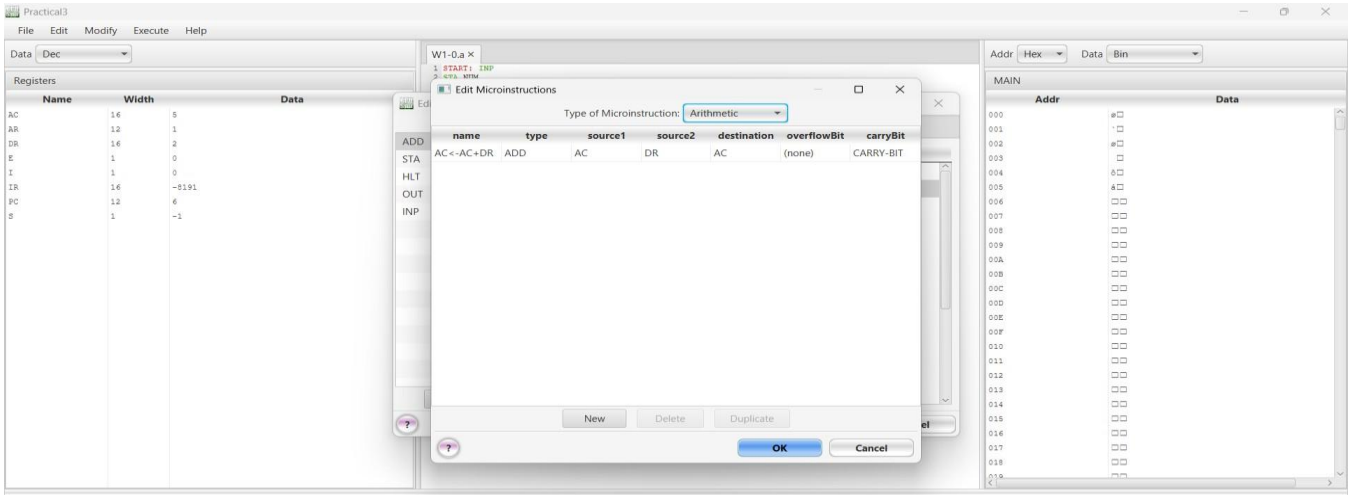

basemachine
File Edit Modify Execute Help
Data Dec
Registers
Name Width Data
AC 16 0
AR 12 0
DR 16 0
E 1 0
I 1 0
IR 16 0
PC 12 0
S 1 0
Untitled ×
1
Edit Fetch Sequence
Fetch Sequence Implementation
AR<-PC
IR<-MAIN[AR]
INCR-PC
AR<-IR(4-15)
DECODE-IR
MicroInstructions
MicroInstructions
arithmetic
branch
decode
end
comment
increment
io
logical
memoryAccess
set
setCondBit
shift
test
OK Cancel
Addr Hex Data Bin
MAIN
Addr Data

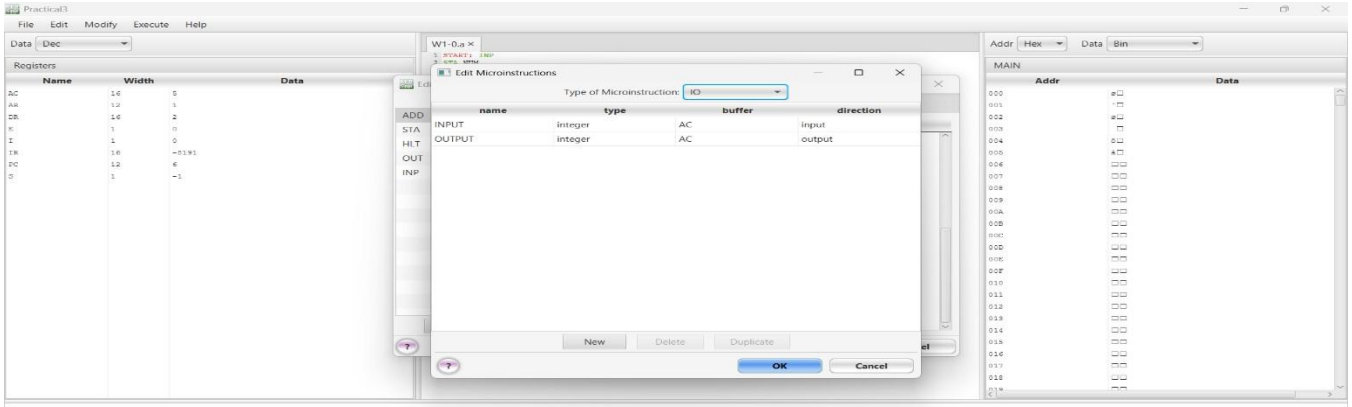
Q. 3. Write an Assembly Program to simulate ADD operation on two user entered numbers.

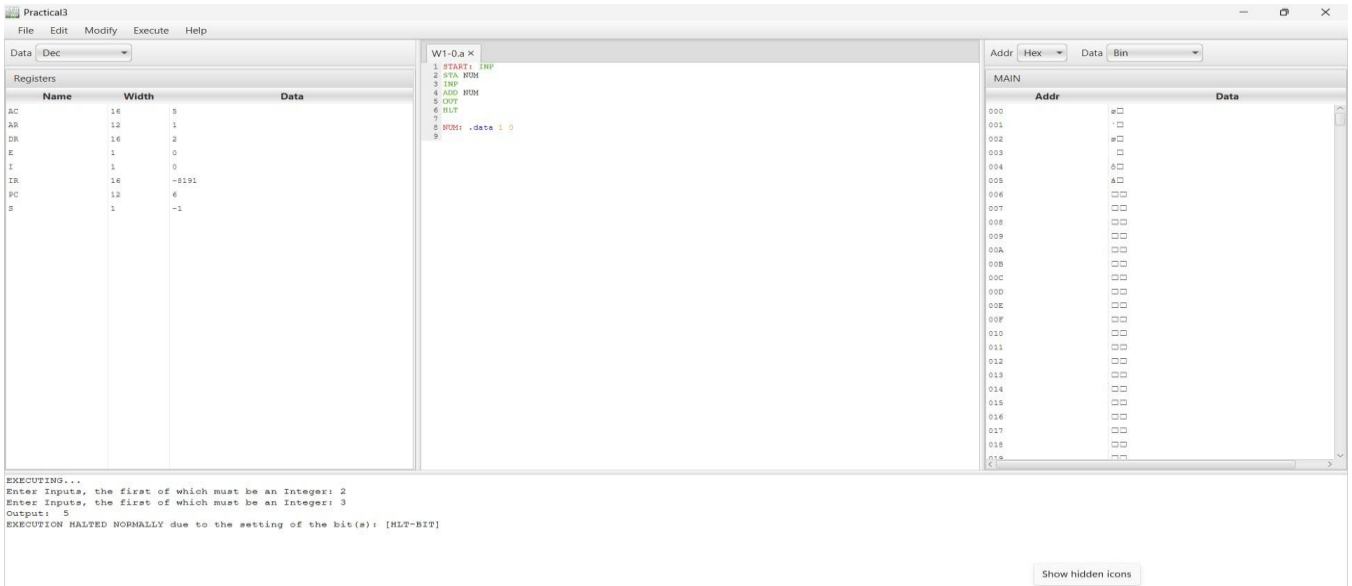




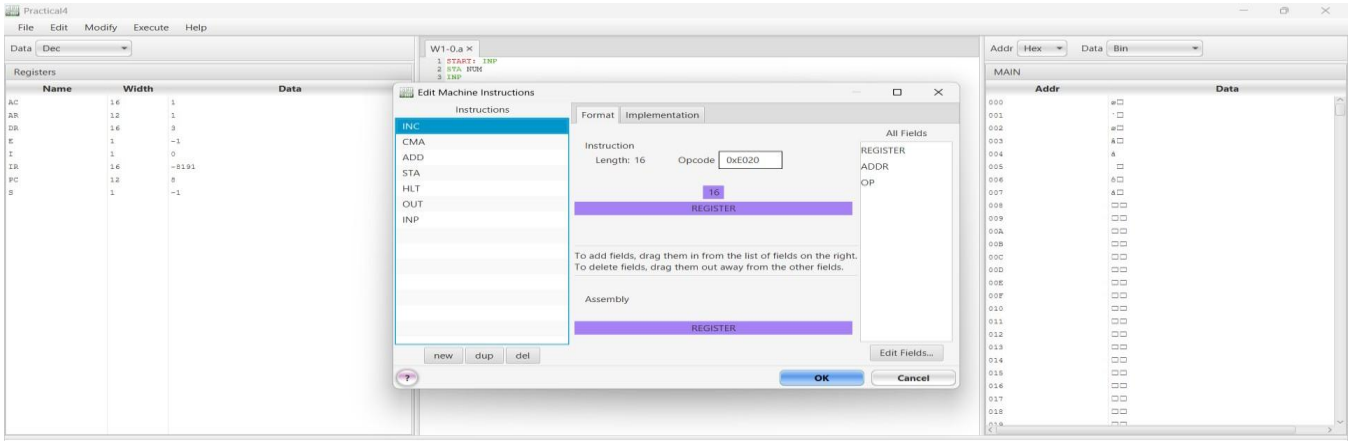


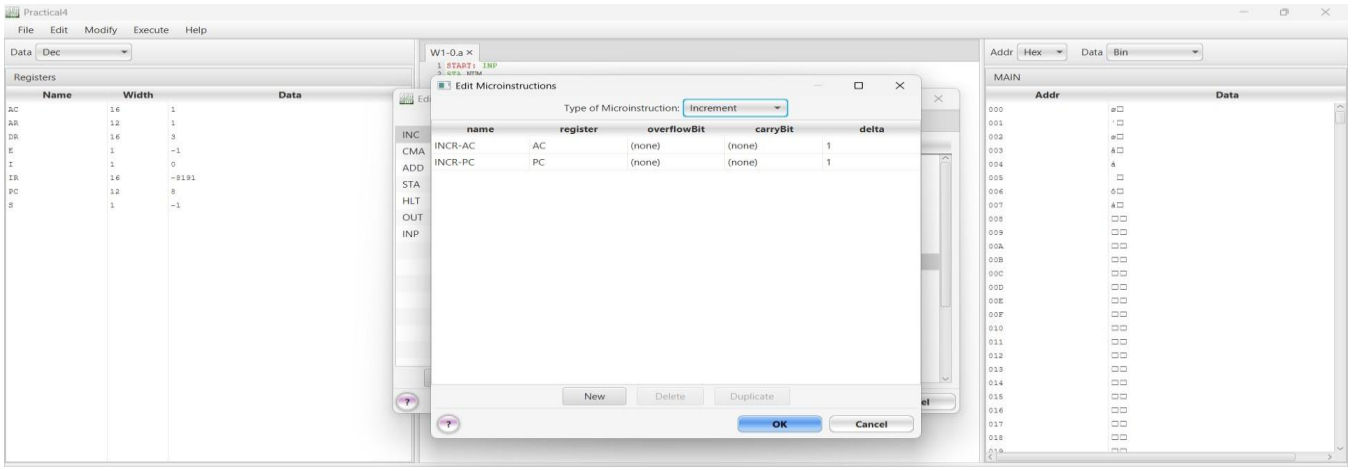

Practical3
File Edit Modify Execute Help
Edit Microinstructions
Type of Microinstruction: Arithmetic
name type source1 source2 destination overflowBit carryBit
AC<-AC+DR ADD AC DR AC (none) CARRY-BIT
New Delete Duplicate
OK Cancel

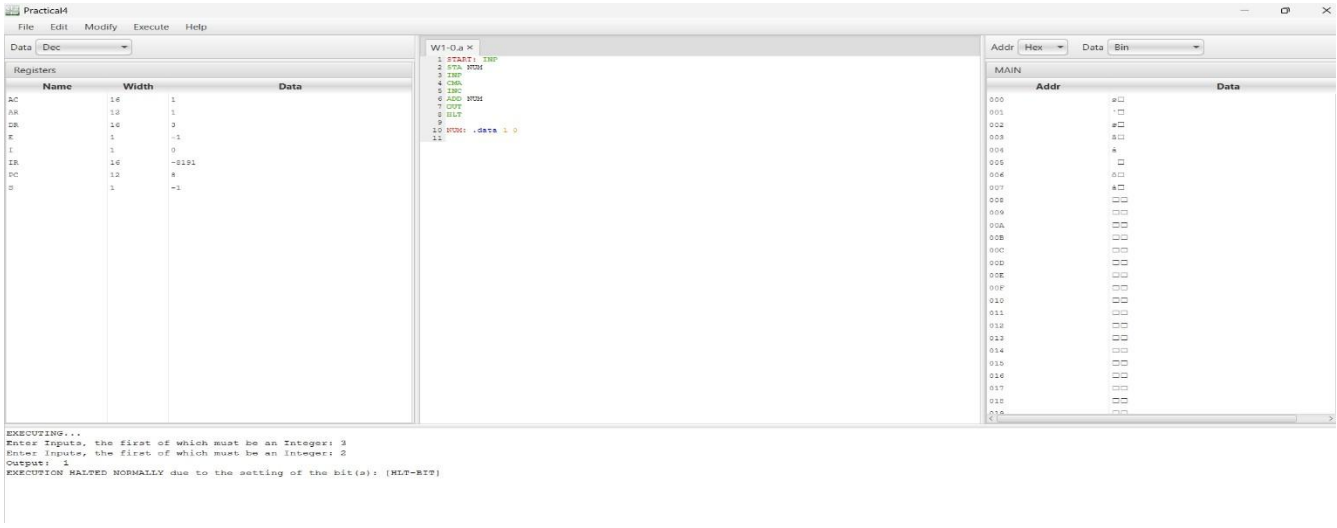

Practical3
File Edit Modify Execute Help
Edit Microinstructions
Type of Microinstruction: IO
name type buffer direction
INPUT integer AC input
OUTPUT integer AC output
New Delete Duplicate
OK Cancel


Practical3
File Edit Modify Execute Help
1 START: INP
2 STA NUM
3 INP
4 ADD NUM
5 OUT
6 HLT
7
8 NUM: .data 1 0
9
EXECUTING...
Enter Inputs, the first of which must be an Integer: 2
Enter Inputs, the first of which must be an Integer: 3
Output: 5
EXECUTION HALTED NORMALLY due to the setting of the bit(s): [HLT-BIT]
Show hidden icons

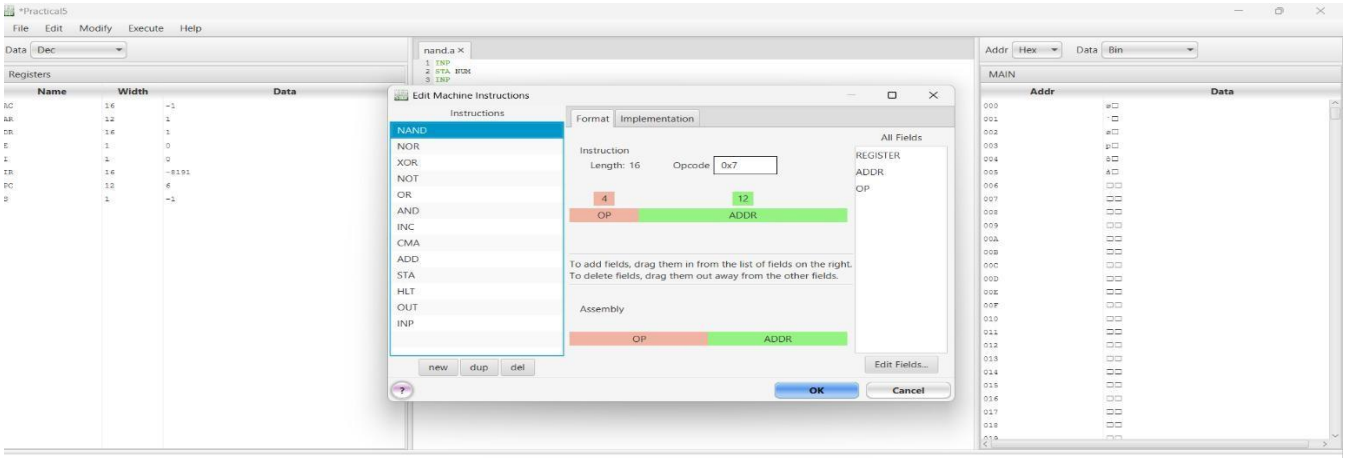
Q. 4. Write an Assembly Program to simulate SUBTRACT operation on two user entered numbers.

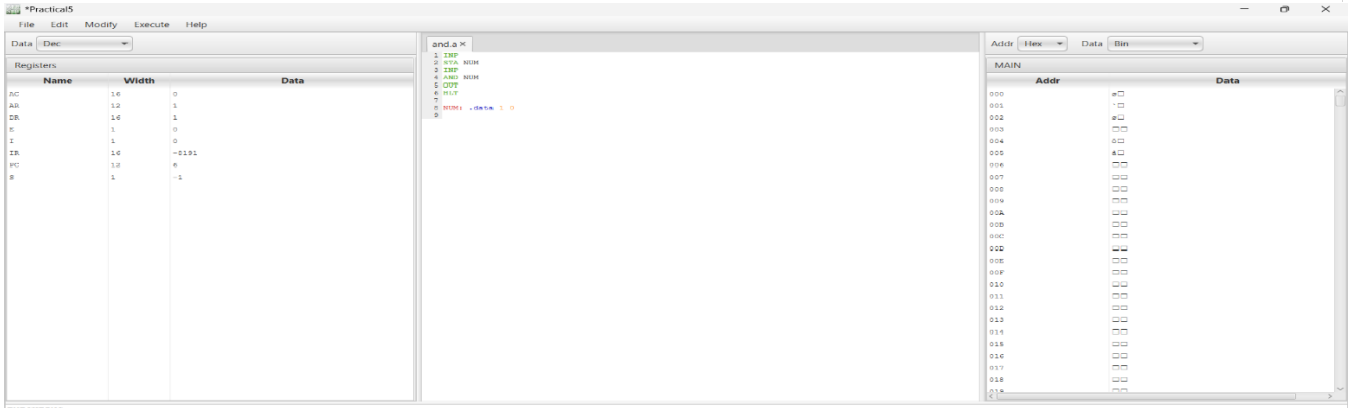


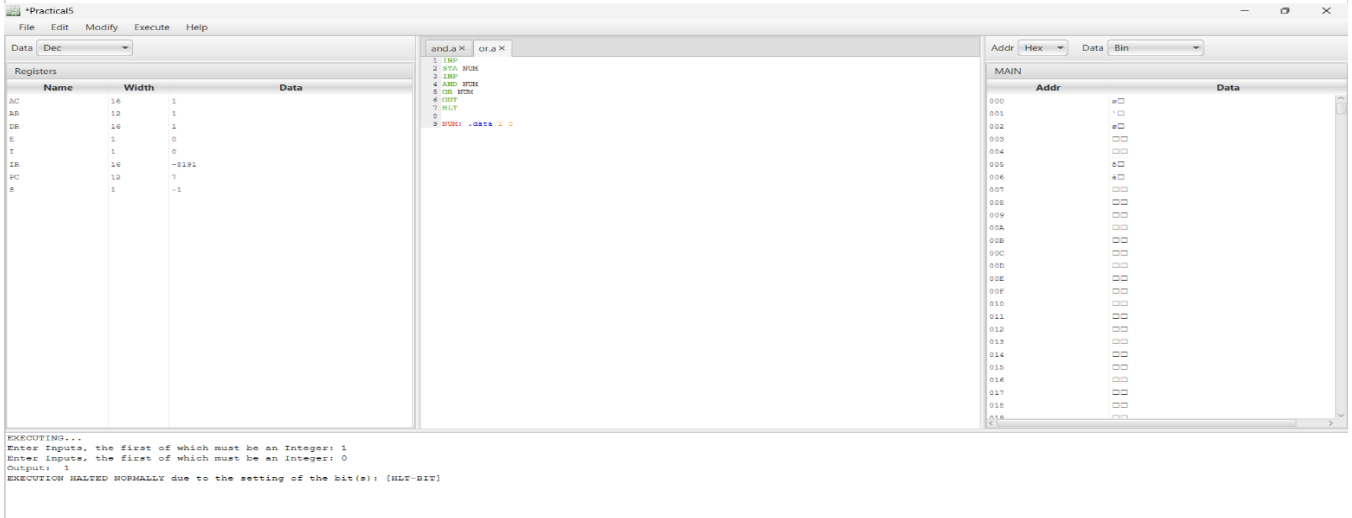


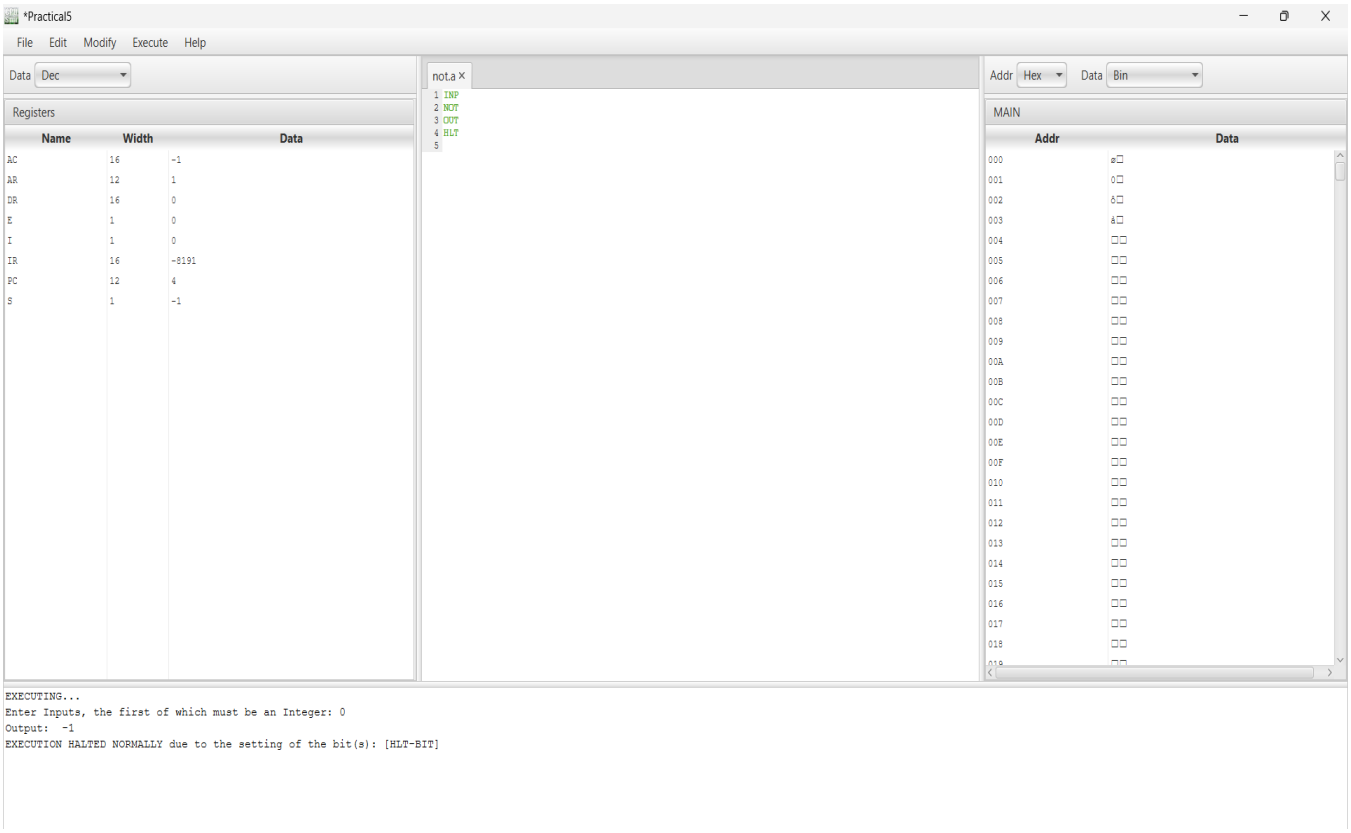


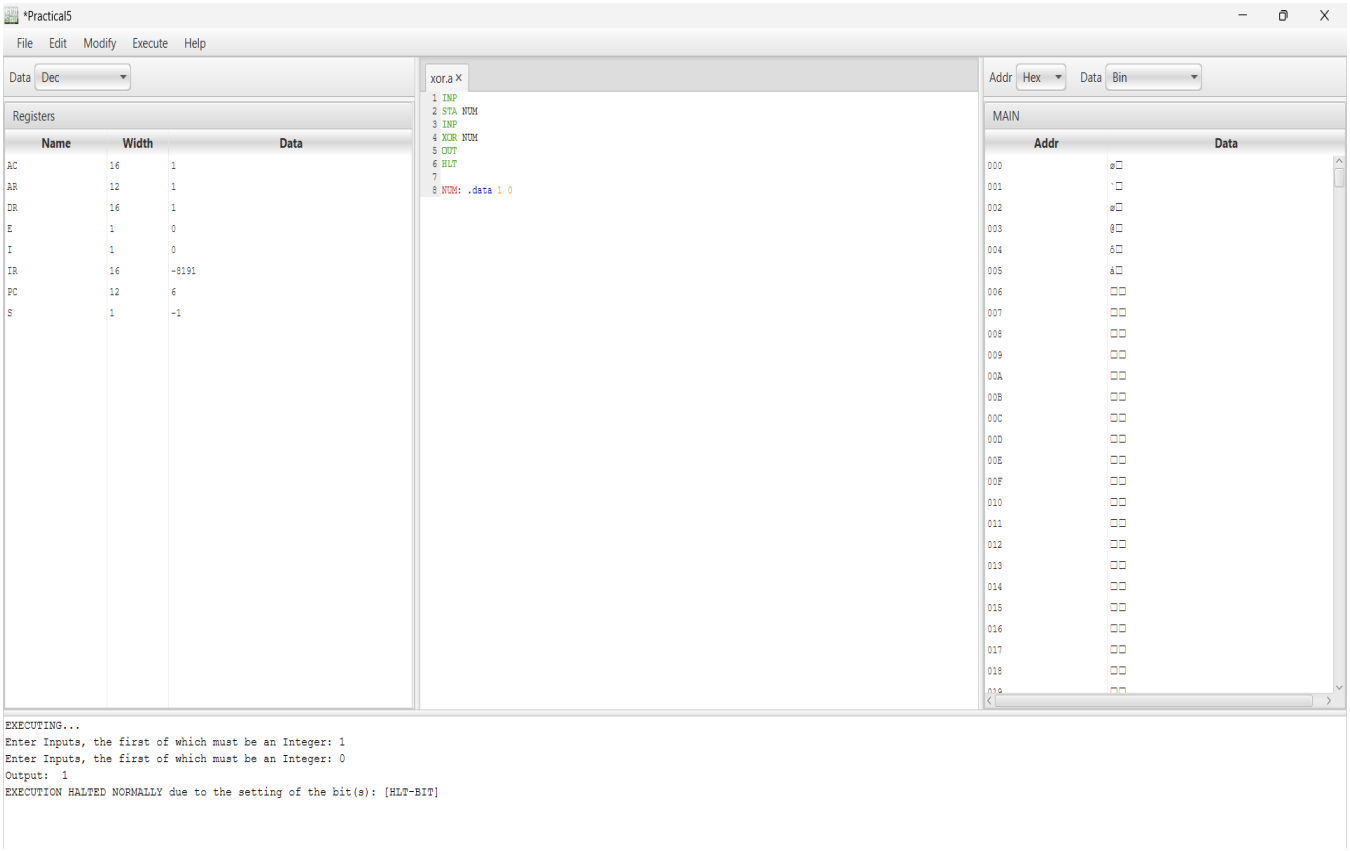
Q. 5. Write an Assembly Program to simulate following logical operations : AND, OR, NOR, NAND, XOR.

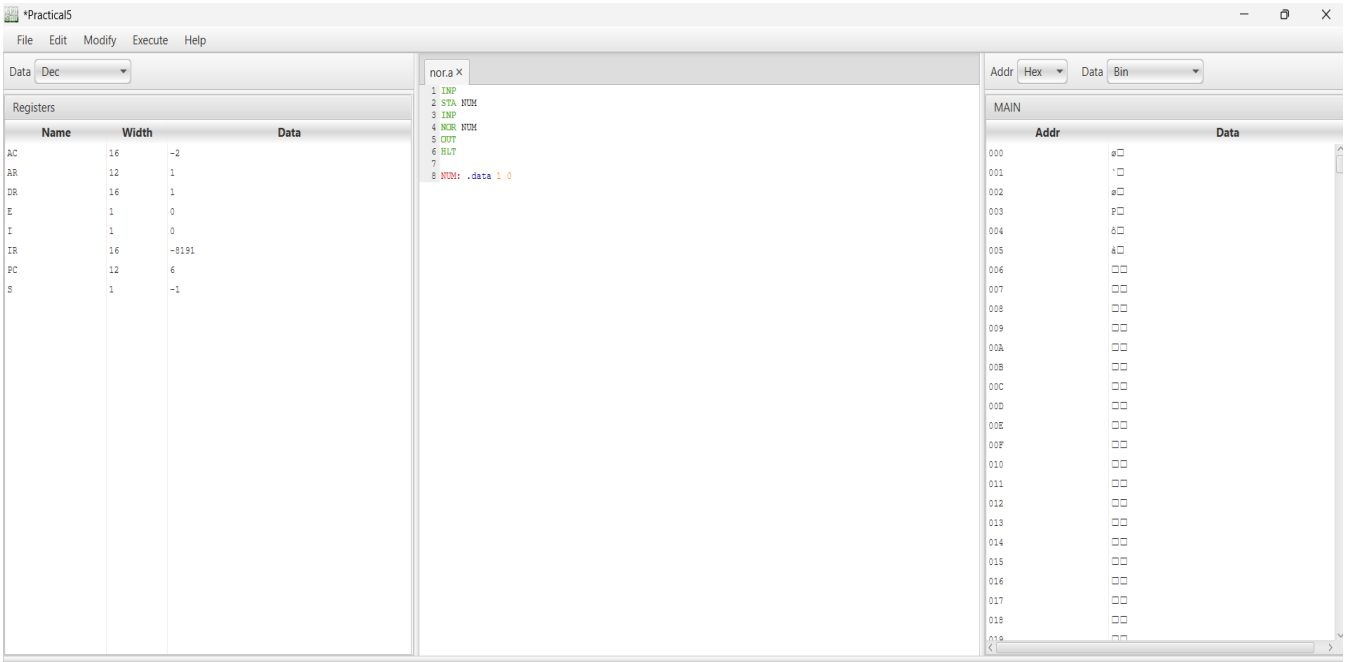


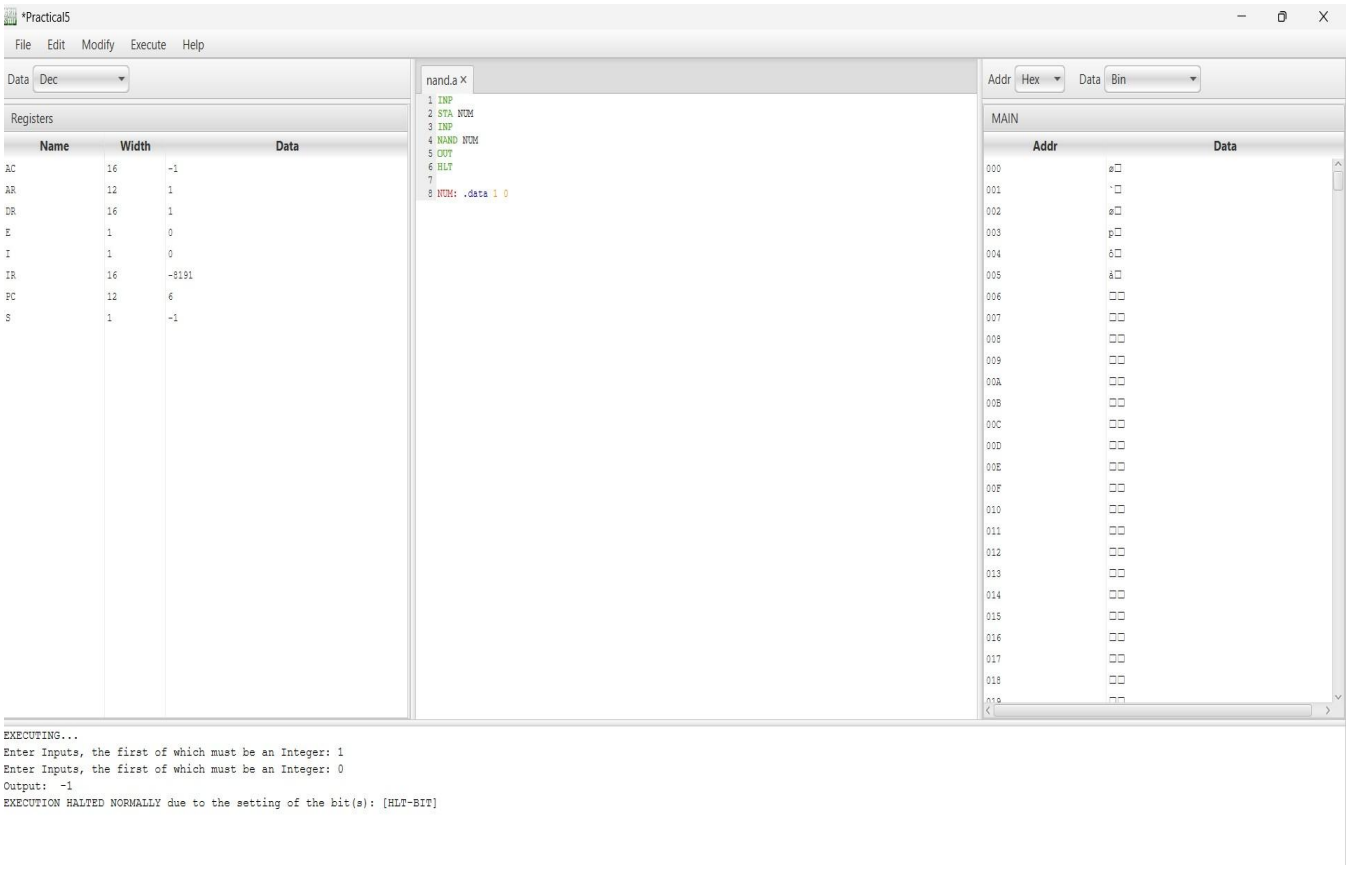




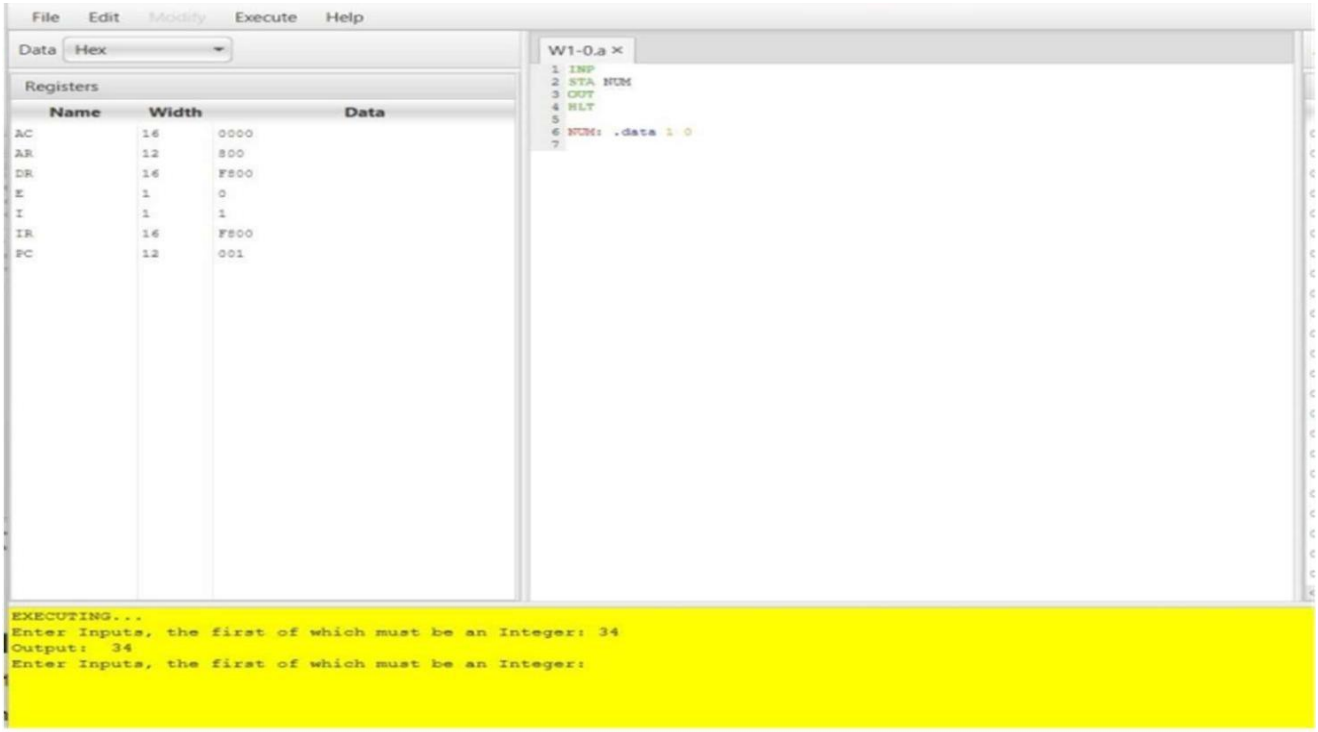

*Practical5
File Edit Modify Execute Help
Data Dec
Registers
Name Width Data
AC 16 -1
AR 12 1
DR 16 0
E 1 0
I 1 0
IR 16 -8191
PC 12 4
S 1 -1
not.a
1 INP
2 NOT
3 OUT
4 HLT
5
Addr Hex Data Bin
MAIN
Addr Data
EXECUTING...
Enter Inputs, the first of which must be an Integer: 0
Output: -1
EXECUTION HALTED NORMALLY due to the setting of the bit(s): [HLT-BIT]

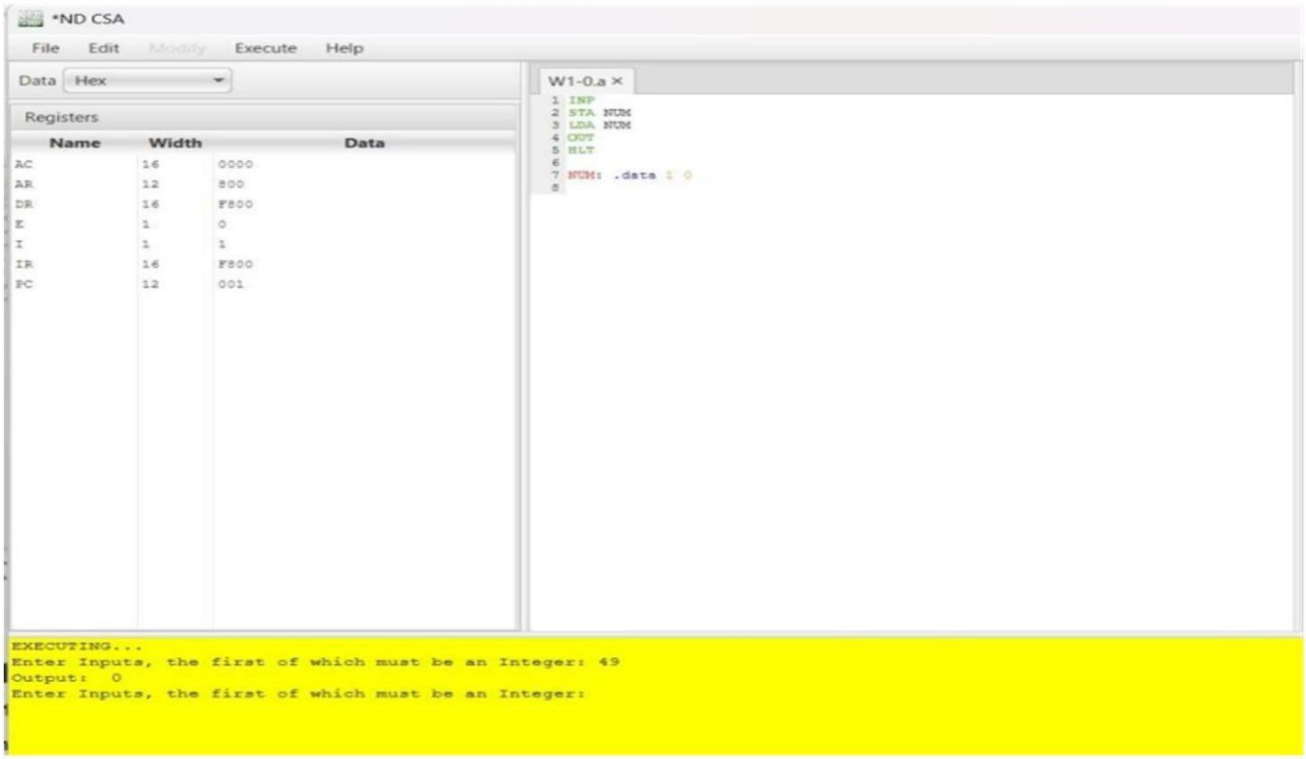

*Practical5
File Edit Modify Execute Help
Data Dec
Registers
Name Width Data
AC 16 1
AR 12 1
DR 16 1
E 1 0
I 1 0
IR 16 -8191
PC 12 6
S 1 -1
xor.a
1 INP
2 STA NUM
3 INP
4 XOR NUM
5 OUT
6 HLT
7
8 NUM: .data 1 0
Addr Hex Data Bin
MAIN
Addr Data
EXECUTING...
Enter Inputs, the first of which must be an Integer: 1
Enter Inputs, the first of which must be an Integer: 0
Output: 1
EXECUTION HALTED NORMALLY due to the setting of the bit(s): [HLT-BIT]

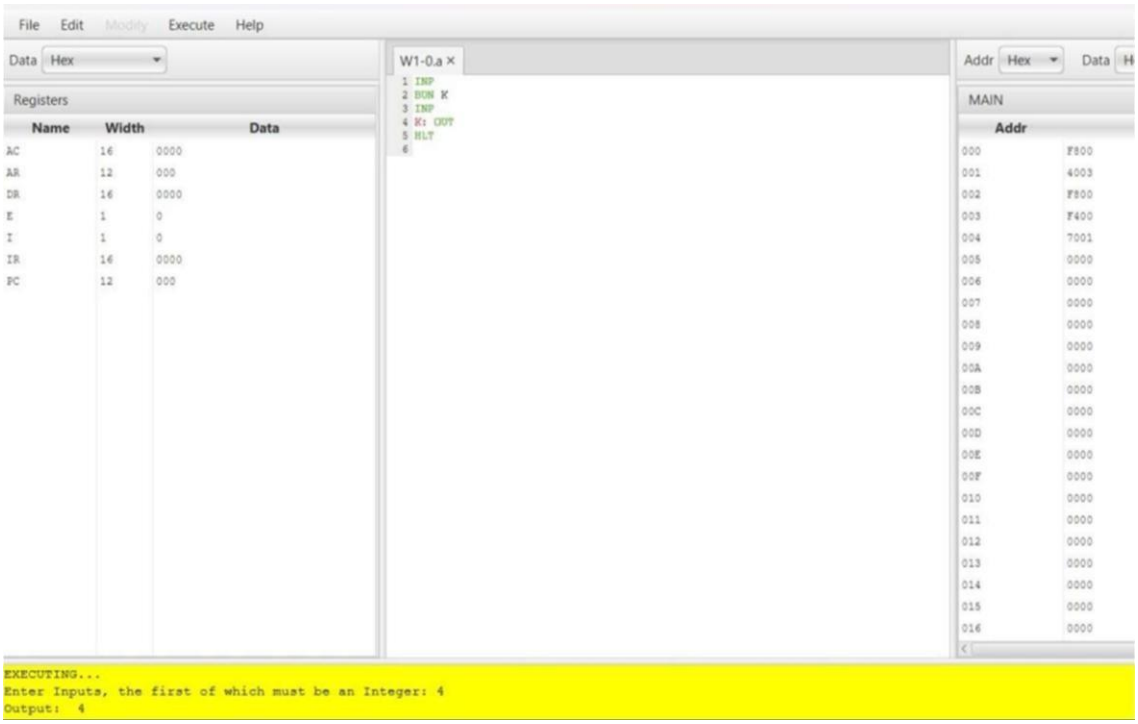

*Practical5
File Edit Modify Execute Help
Data Dec
Registers
Name Width Data
AC 16 -2
AR 12 1
DR 16 1
E 1 0
I 1 0
IR 16 -8191
PC 12 6
S 1 -1
nor.a
1 INP
2 STA NUM
3 INP
4 NOR NUM
5 OUT
6 HLT
7
8 NUM: .data 1 0
Addr Hex Data Bin
MAIN
Addr Data
EXECUTING...
Enter Inputs, the first of which must be an Integer: 1
Enter Inputs, the first of which must be an Integer: 1
Output: -2
EXECUTION HALTED NORMALLY due to the setting of the bit(s): [HLT-BIT]

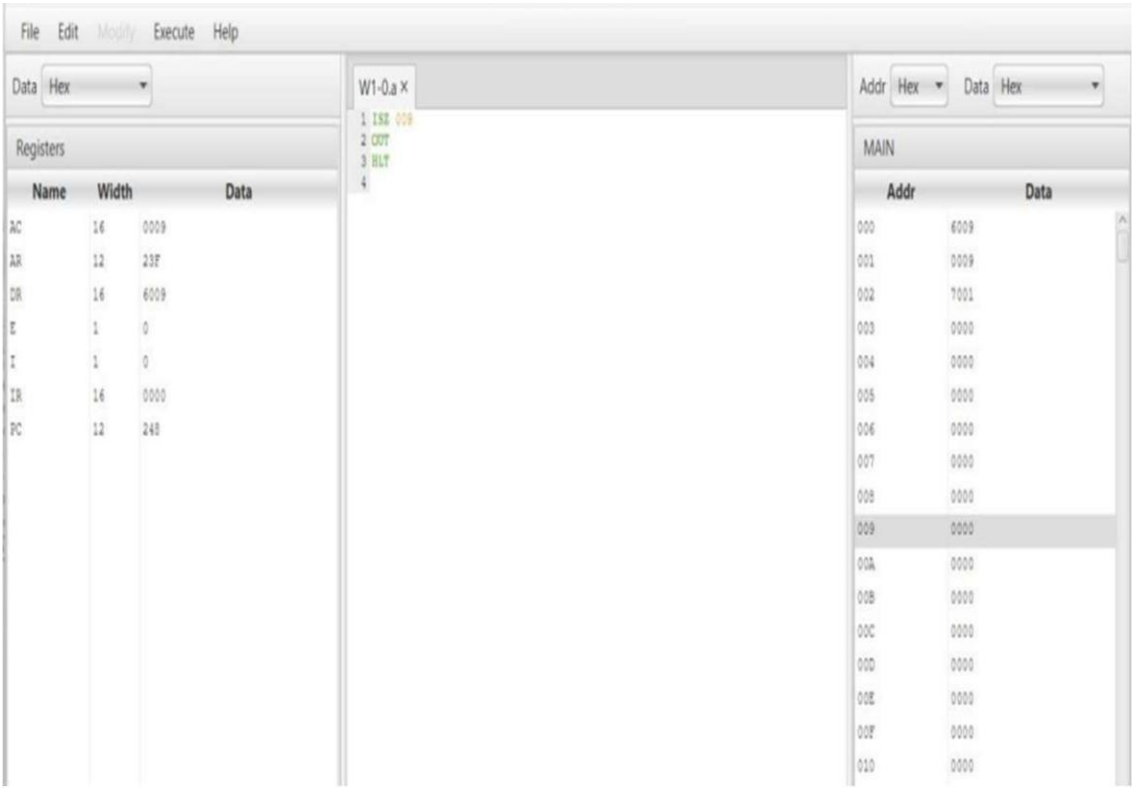

*Practical5
File Edit Modify Execute Help
Data Dec
Registers
Name Width Data
AC 16 -1
AR 12 1
DR 16 1
E 1 0
I 1 0
IR 16 -8191
PC 12 6
S 1 -1
nand.a
1 INP
2 STA NUM
3 INP
4 NAND NUM
5 OUT
6 HLT
7
8 NUM: .data 1 0
Addr Hex Data Bin
MAIN
Addr Data
EXECUTING...
Enter Inputs, the first of which must be an Integer: 1
Enter Inputs, the first of which must be an Integer: 0
Output: -1
EXECUTION HALTED NORMALLY due to the setting of the bit(s): [HLT-BIT]

Q6. Write an Assembly program to simulate following memory reference instructions : LDA, STA, BUN, ISZ.

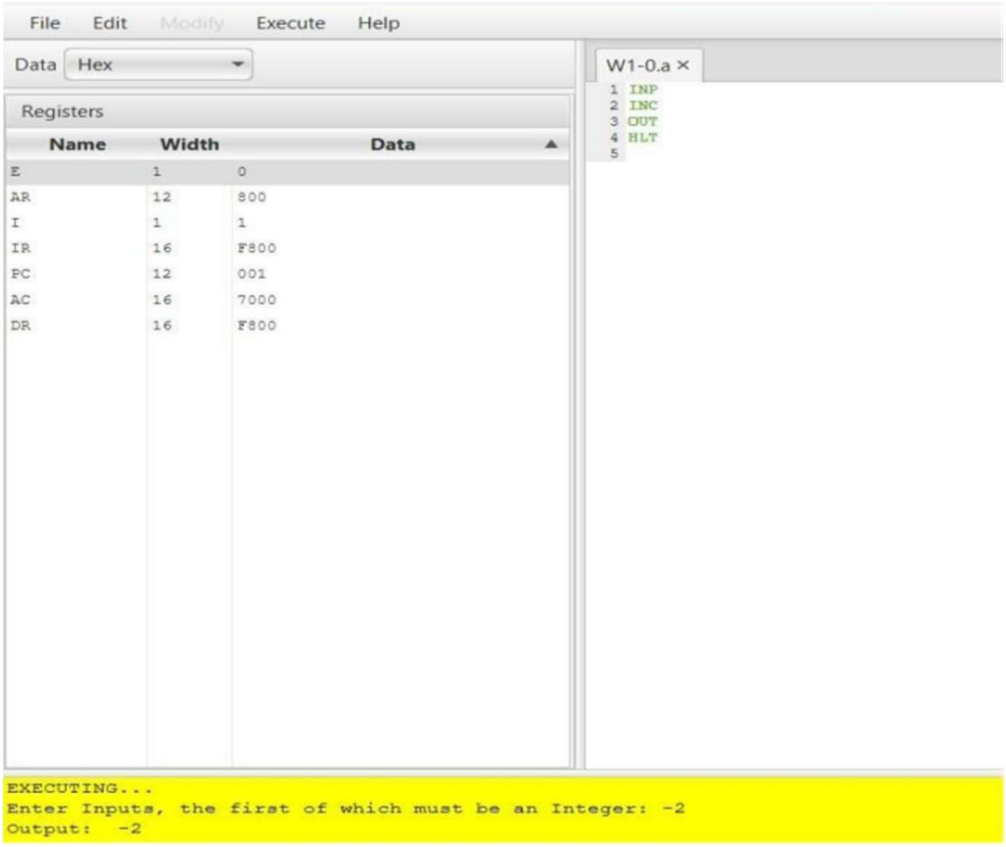


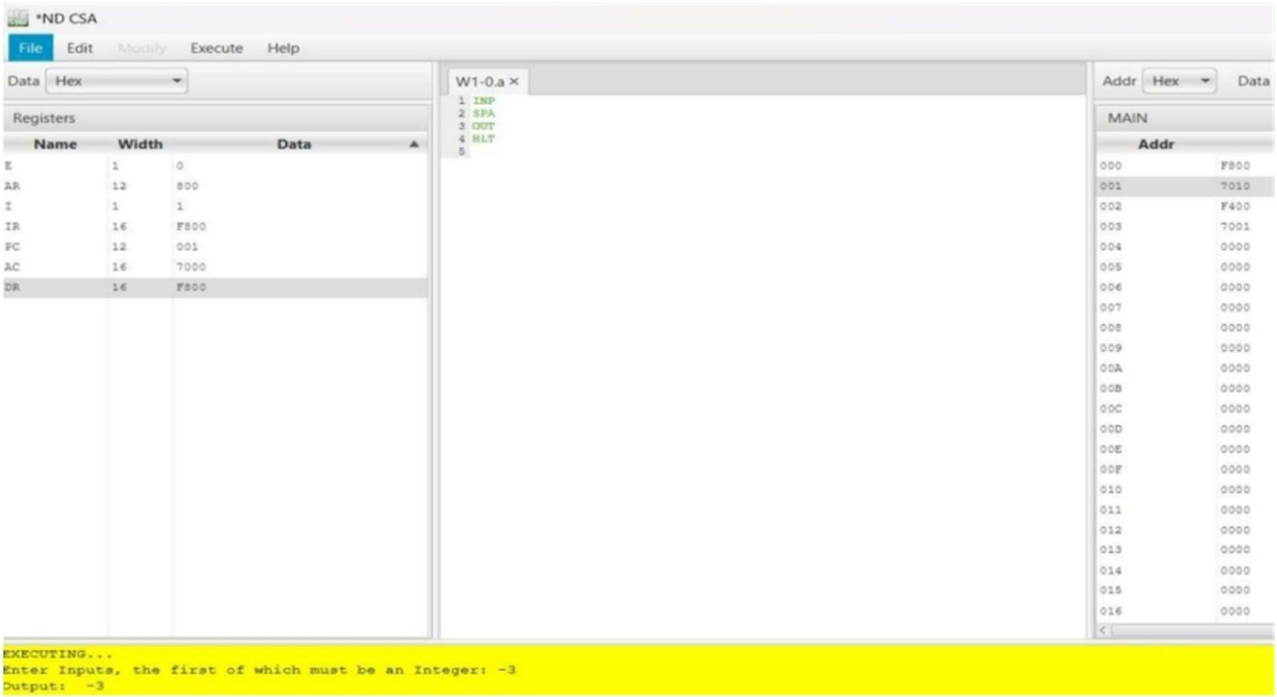


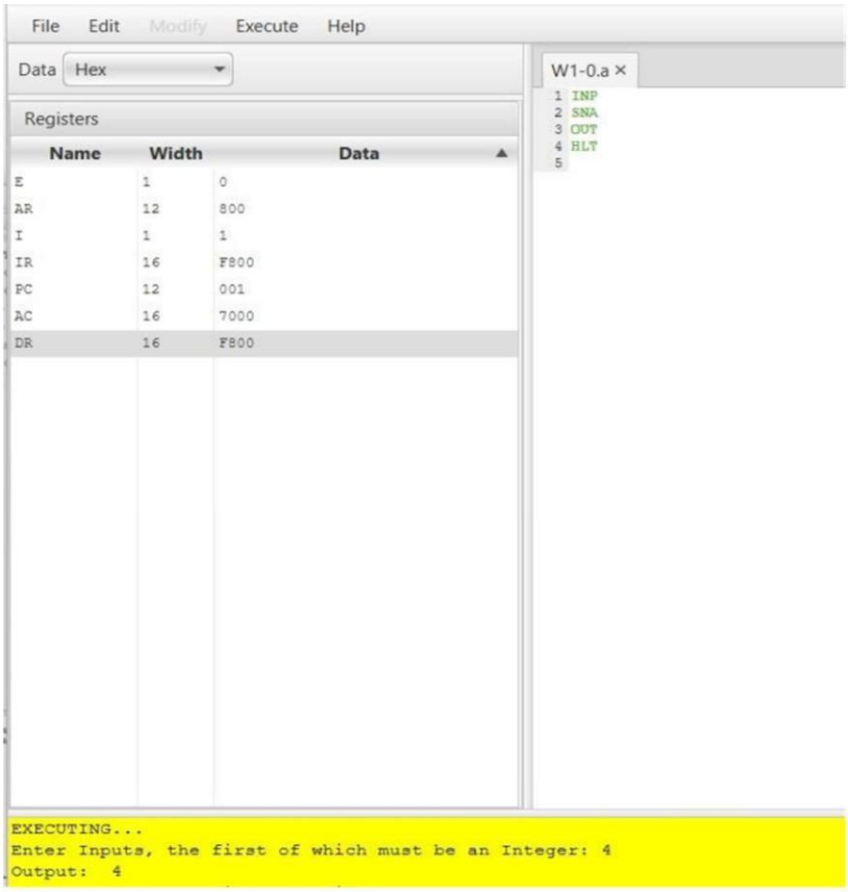

File Edit Modify Execute Help
Data Hex
Registers
Name Width Data
AC 16 0000
AR 12 000
DR 16 0000
E 1 0
I 1 0
IR 16 0000
PC 12 000
W1-0.a
1 INP
2 SUB K
3 INP
4 K: CUT
5 HLT
6
Addr Hex Data
MAIN
Addr
000 F800
001 4003
002 F800
003 F400
004 7001
005 0000
006 0000
007 0000
008 0000
009 0000
00A 0000
00B 0000
00C 0000
00D 0000
00E 0000
00F 0000
010 0000
011 0000
012 0000
013 0000
014 0000
015 0000
016 0000
EXECUTING...
Enter Inputs, the first of which must be an Integer: 4
Output: 4

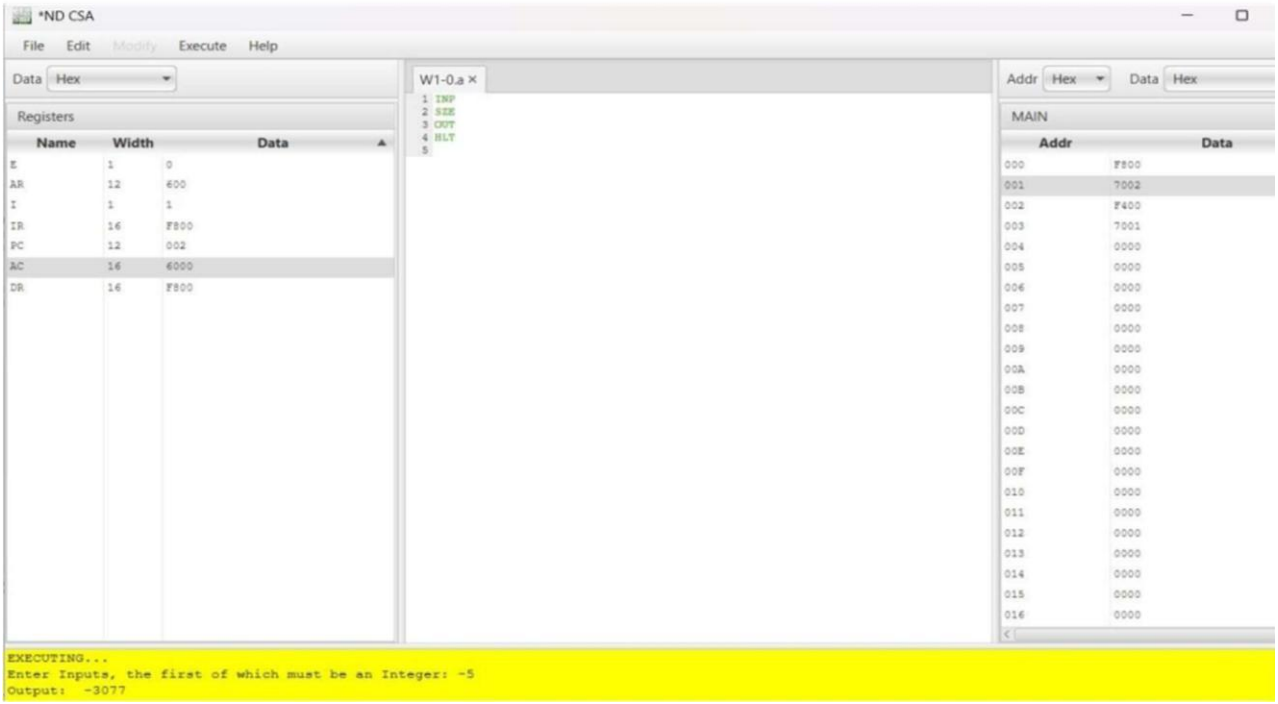

File Edit Modify Execute Help
Data Hex
Registers
Name Width Data
AC 16 0009
AR 12 23F
DR 16 6009
E 1 0
I 1 0
IR 16 0000
PC 12 248
W1-0.a
1 ISZ 009
2 CUT
3 HLT
4
Addr Hex Data Hex
MAIN
Addr Data
000 6009
001 0009
002 7001
003 0000
004 0000
005 0000
006 0000
007 0000
008 0000
009 0000
00A 0000
00B 0000
00C 0000
00D 0000
00E 0000
00F 0000
010 0000

Q7. Write an Assembly Program to simulate following register instructions : INC, SPA, SNA, SZE.

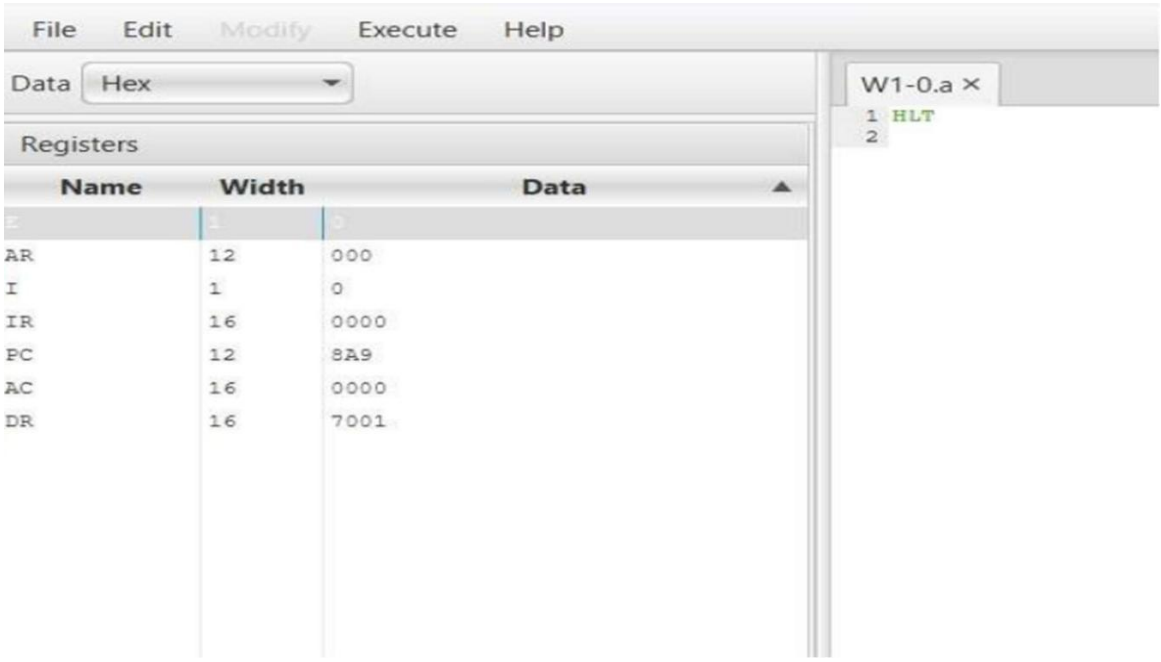


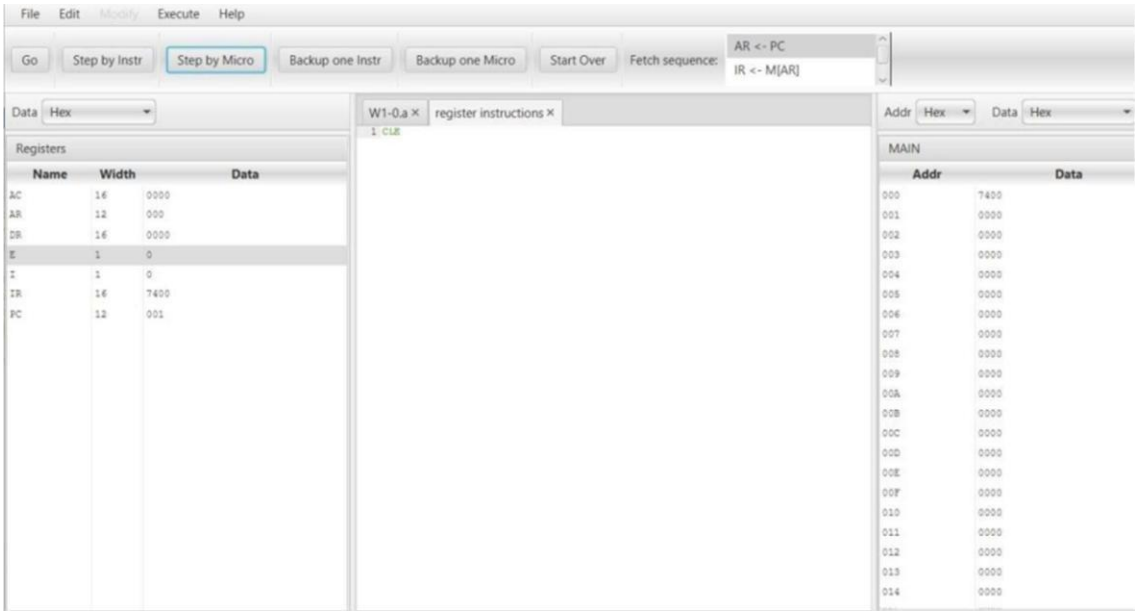


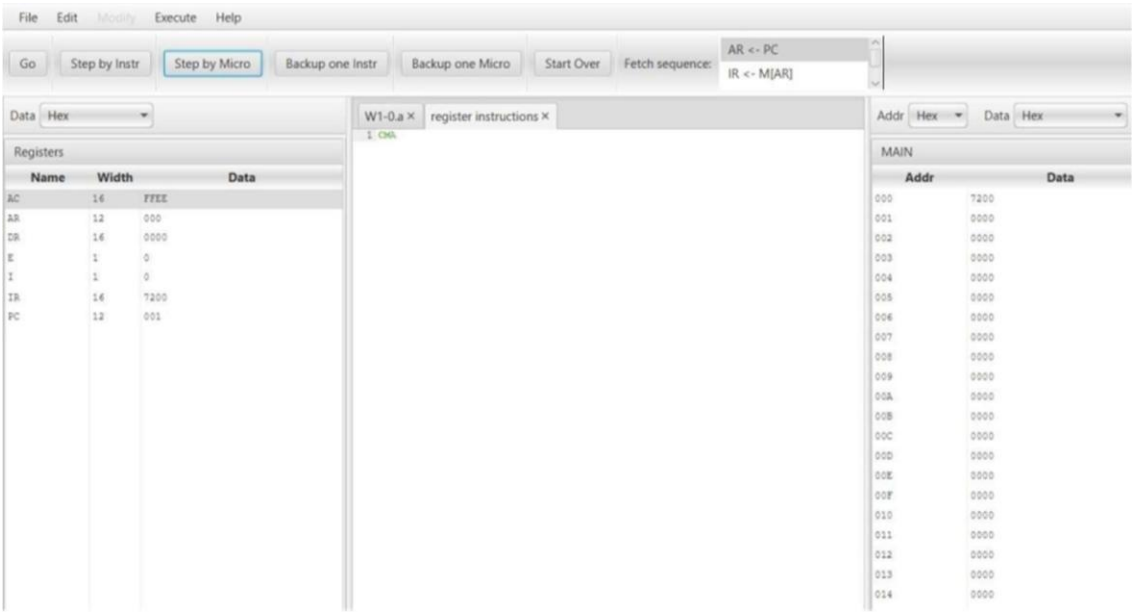

File Edit Modify Execute Help
Data Hex
Registers
Name Width Data
E 1 0
AR 12 800
I 1 1
IR 16 F800
PC 12 001
AC 16 7000
DR 16 F800
W1-0.a ×
1 INP
2 SNA
3 OUT
4 HLT
5
EXECUTING...
Enter Inputs, the first of which must be an Integer: 4
Output: 4


*ND CSA
File Edit Modify Execute Help
Data Hex
Registers
Name Width Data
E 1 0
AR 12 600
I 1 1
IR 16 F800
PC 12 002
AC 16 6000
DR 16 F800
W1-0.a ×
1 INP
2 SZE
3 OUT
4 HLT
5
Addr Hex Data Hex
MAIN
Addr Data
000 F800
001 7002
002 F400
003 7001
004 0000
005 0000
006 0000
007 0000
008 0000
009 0000
00A 0000
00B 0000
00C 0000
00D 0000
00E 0000
00F 0000
010 0000
011 0000
012 0000
013 0000
014 0000
015 0000
016 0000
EXECUTING...
Enter Inputs, the first of which must be an Integer: -5
Output: -3077

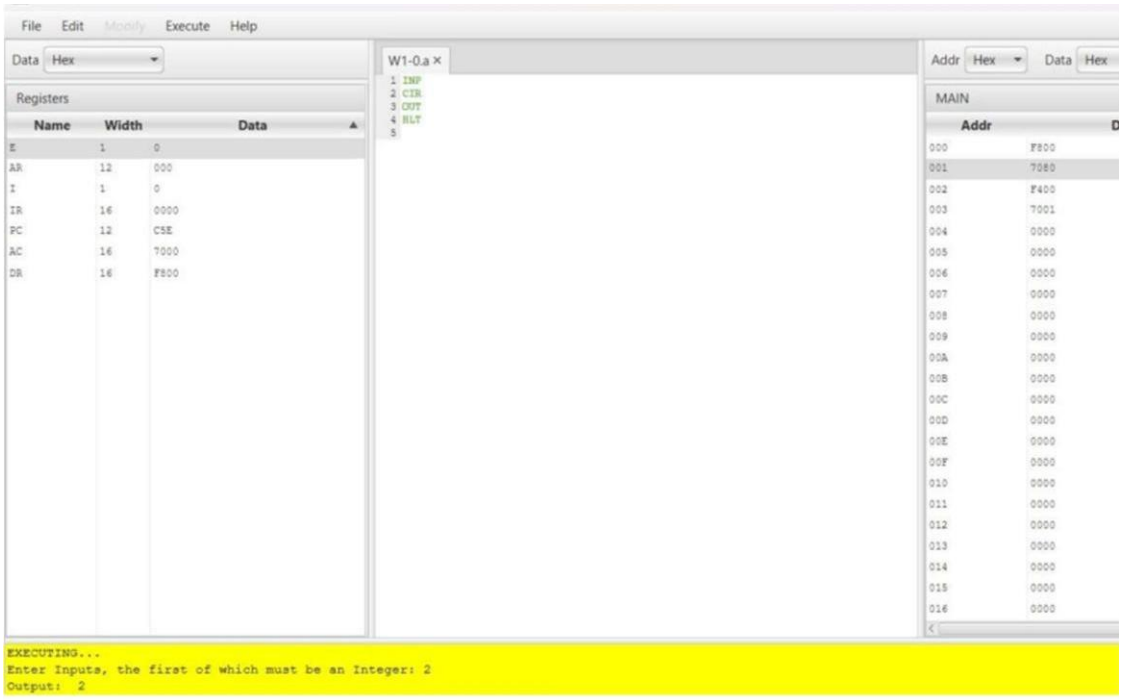
Q8. Write an Assembly Program to simulate following register instructions: CLA, CMA, CME, HLT.

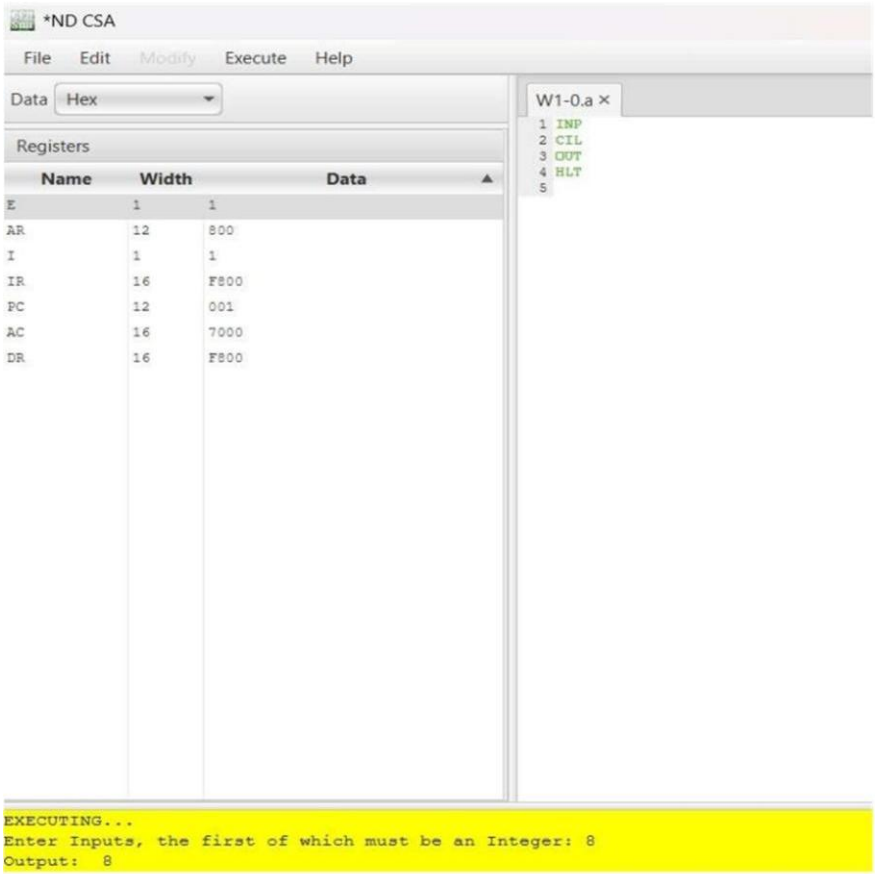



File Edit Modify Execute Help
Go
Step by Instr
Step by Micro
Backup one Instr
Backup one Micro
Start Over
Fetch sequence:
AR <- PC
IR <- M[AR]
Data Hex
W1-0.a ×
register instructions ×
1 CLE
Addr Hex
Data Hex
Registers
Name Width Data
AC 16 0000
AR 12 000
DR 16 0000
E 1 0
I 1 0
IR 16 7400
PC 12 001
MAIN
Addr Data
000 7400
001 0000
002 0000
003 0000
004 0000
005 0000
006 0000
007 0000
008 0000
009 0000
00A 0000
00B 0000
00C 0000
00D 0000
00E 0000
00F 0000
010 0000
011 0000
012 0000
013 0000
014 0000

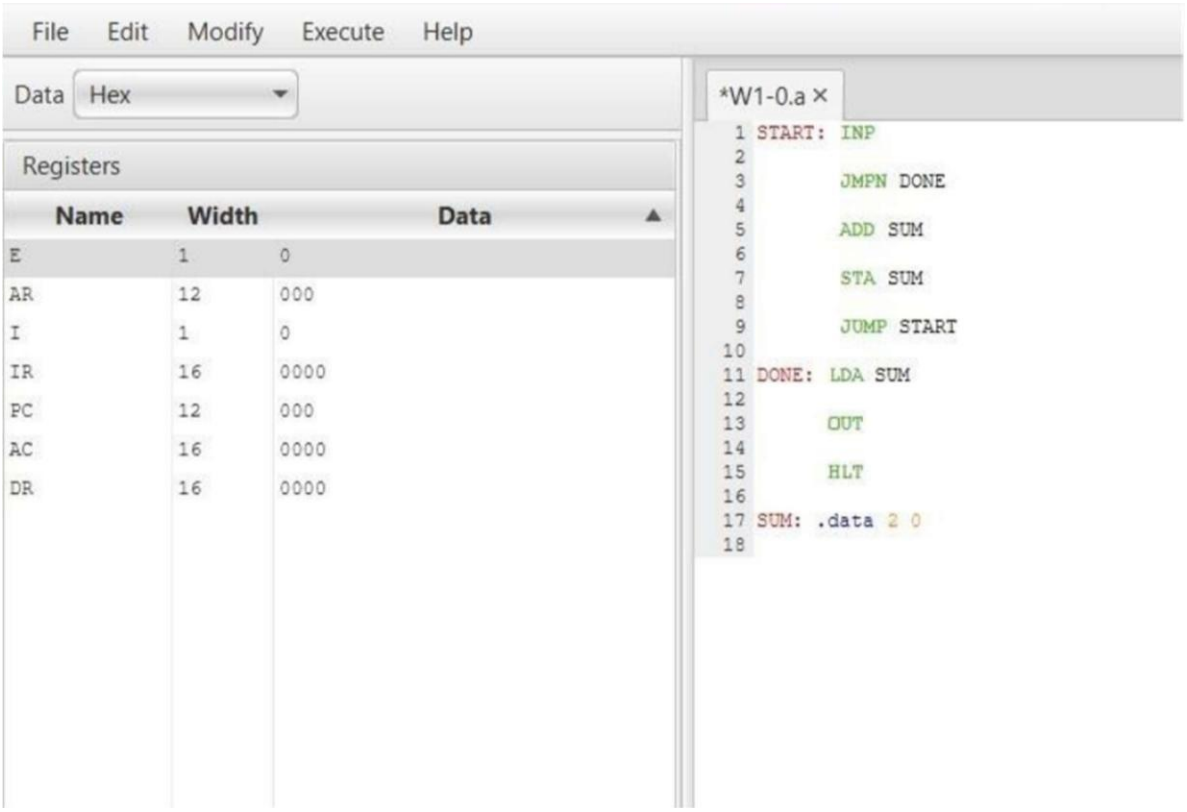

File Edit Modify Execute Help
Go
Step by Instr
Step by Micro
Backup one Instr
Backup one Micro
Start Over
Fetch sequence:
AR <- PC
IR <- M[AR]
Data Hex
W1-0.a ×
register instructions ×
1 CMA
Addr Hex
Data Hex
Registers
Name Width Data
AC 16 FFEE
AR 12 000
DR 16 0000
E 1 0
I 1 0
IR 16 7200
PC 12 001
MAIN
Addr Data
000 7200
001 0000
002 0000
003 0000
004 0000
005 0000
006 0000
007 0000
008 0000
009 0000
00A 0000
00B 0000
00C 0000
00D 0000
00E 0000
00F 0000
010 0000
011 0000
012 0000
013 0000
014 0000

Q. 9. Write an Assembly Program to simulate following register instructions : CIR, CIL.

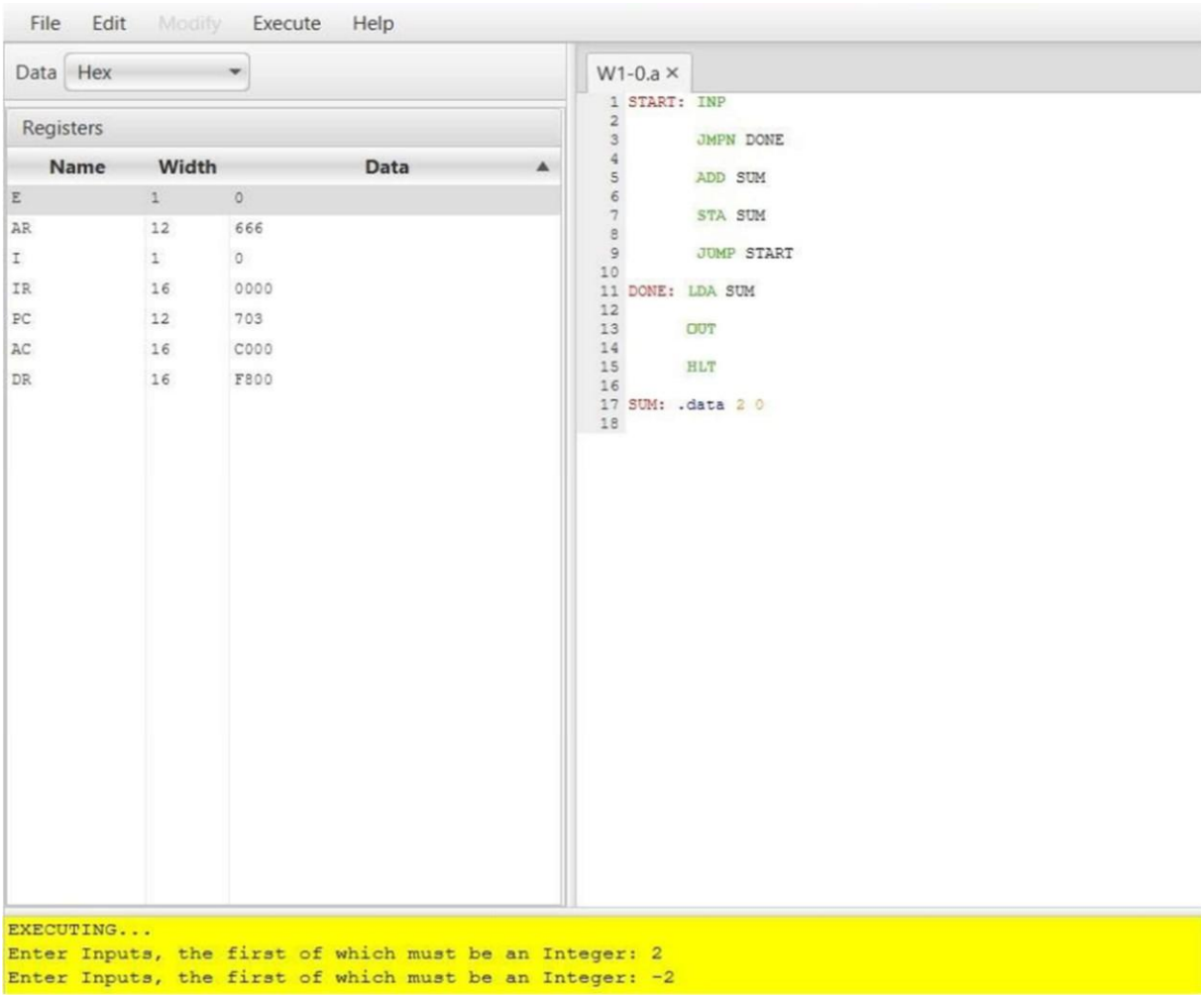




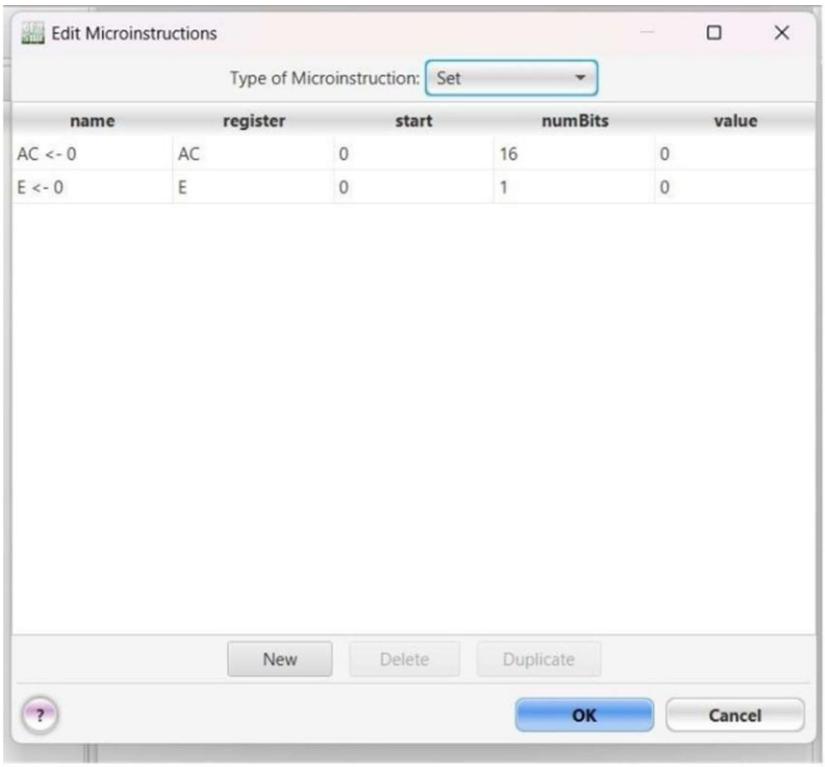
Q. 10. Write an Assembly Program that reads in integers and adds them together ; until a negative non zero number is read in.

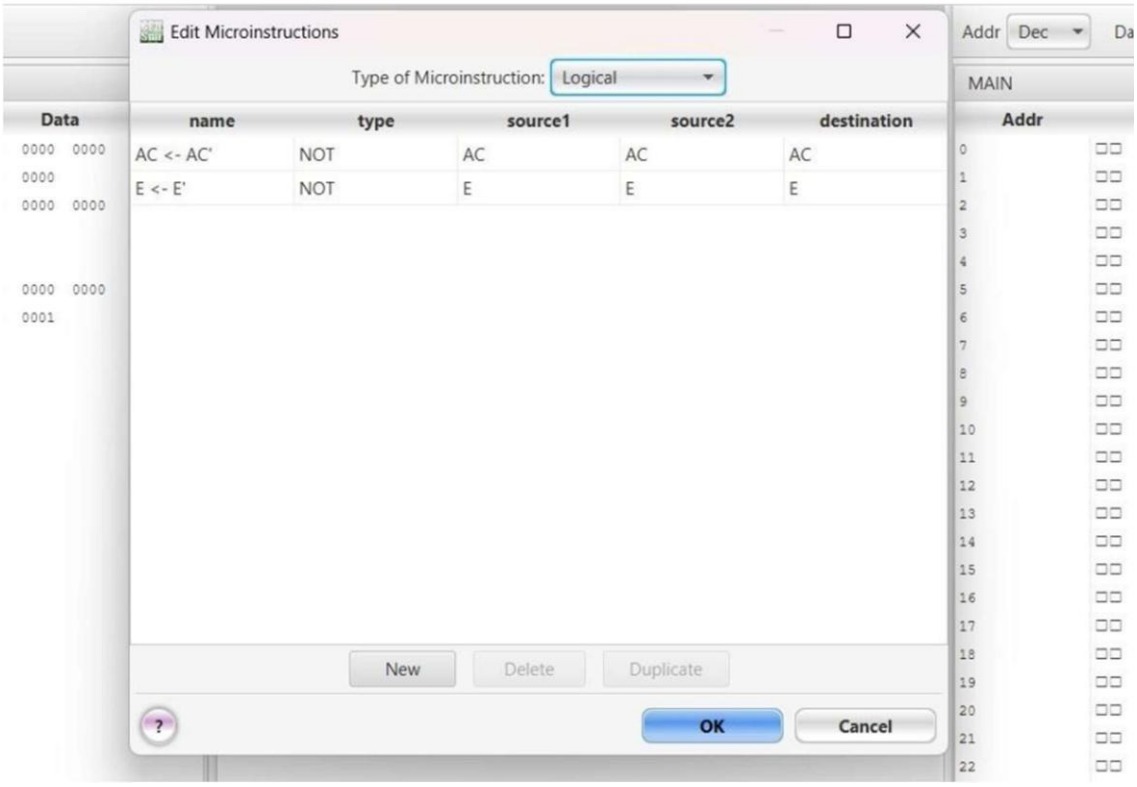


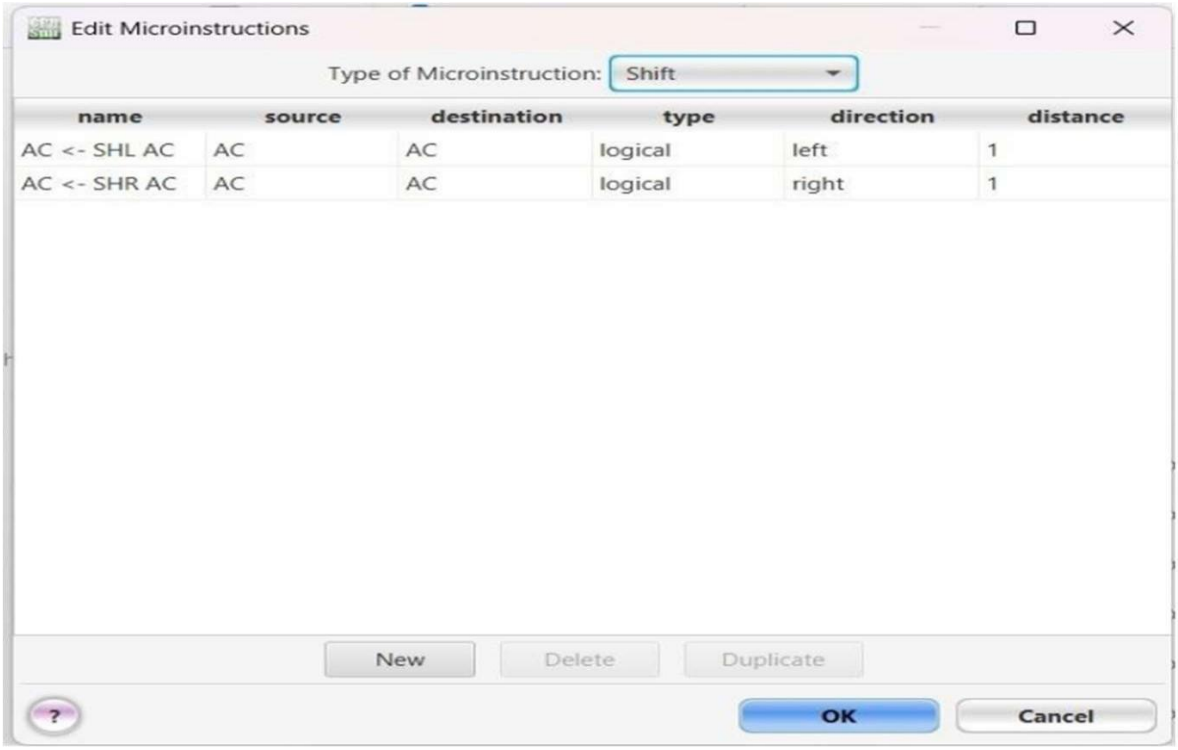
Q. 11. Write an Assembly Program that reads in integers and adds them together ; until zero is read in.

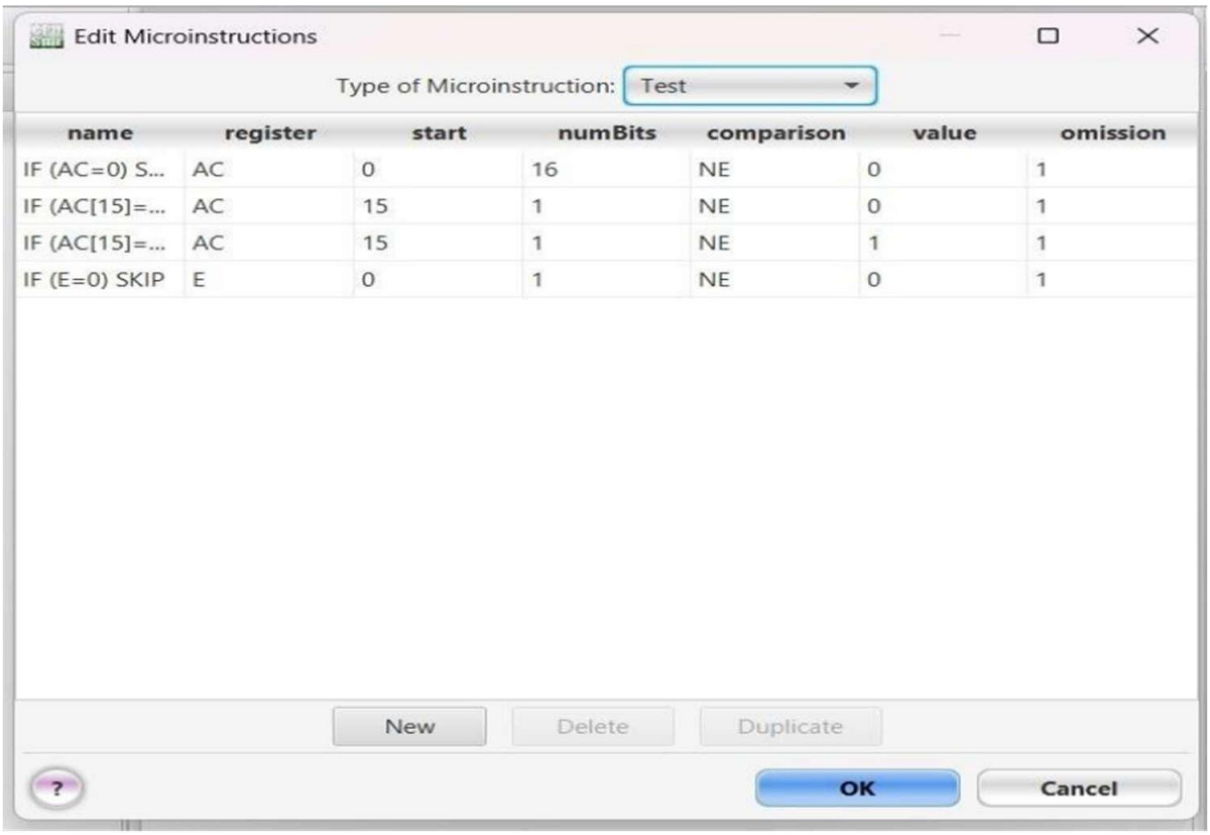


IMPORTANT MICRO INSTRUCTIONS :






Edit Microinstructions
Type of Microinstruction: Shift
name source destination type direction distance
AC <- SHL AC AC AC logical left 1
AC <- SHR AC AC AC logical right 1
New
Delete
Duplicate
OK
Cancel


Edit Microinstructions
Type of Microinstruction: Test
name register start numBits comparison value omission
IF (AC=0) S... AC 0 16 NE 0 1
IF (AC[15]=... AC 15 1 NE 0 1
IF (AC[15]=... AC 15 1 NE 1 1
IF (E=0) SKIP E 0 1 NE 0 1
New
Delete
Duplicate
OK
Cancel