

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.

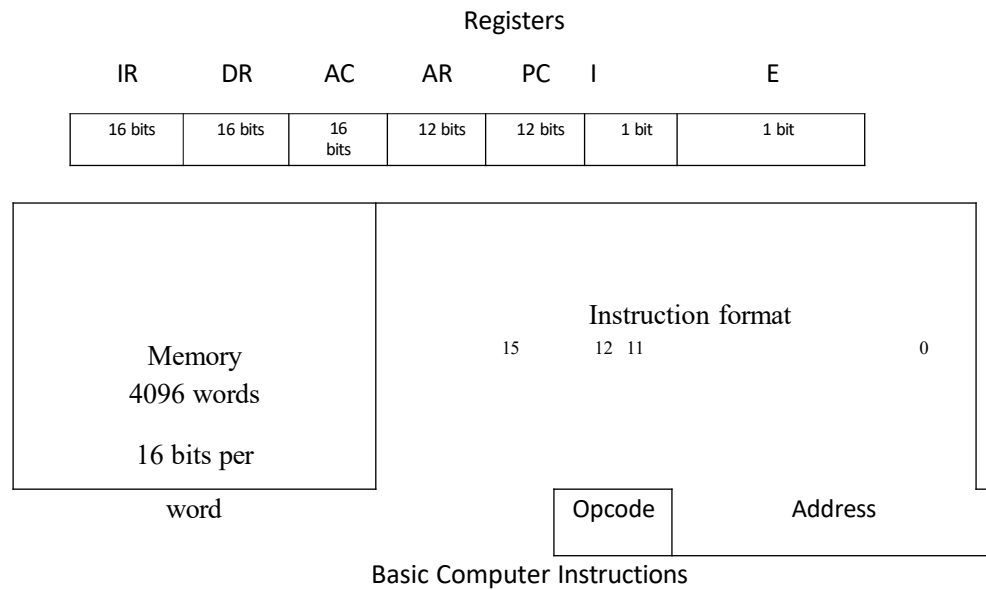
I am also very thankful to my parents and my friends who have boosted me up morally with their continuous support.

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

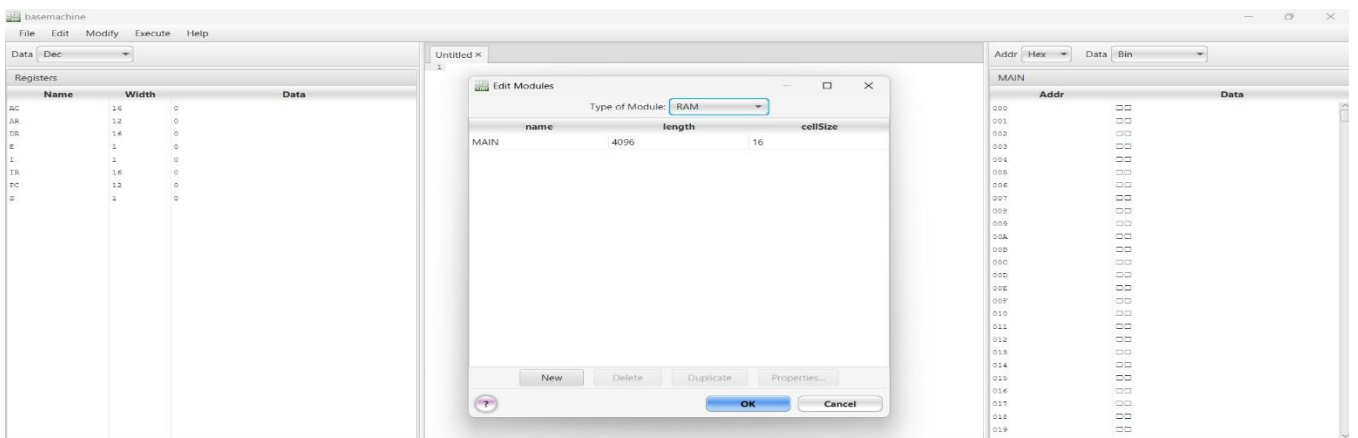
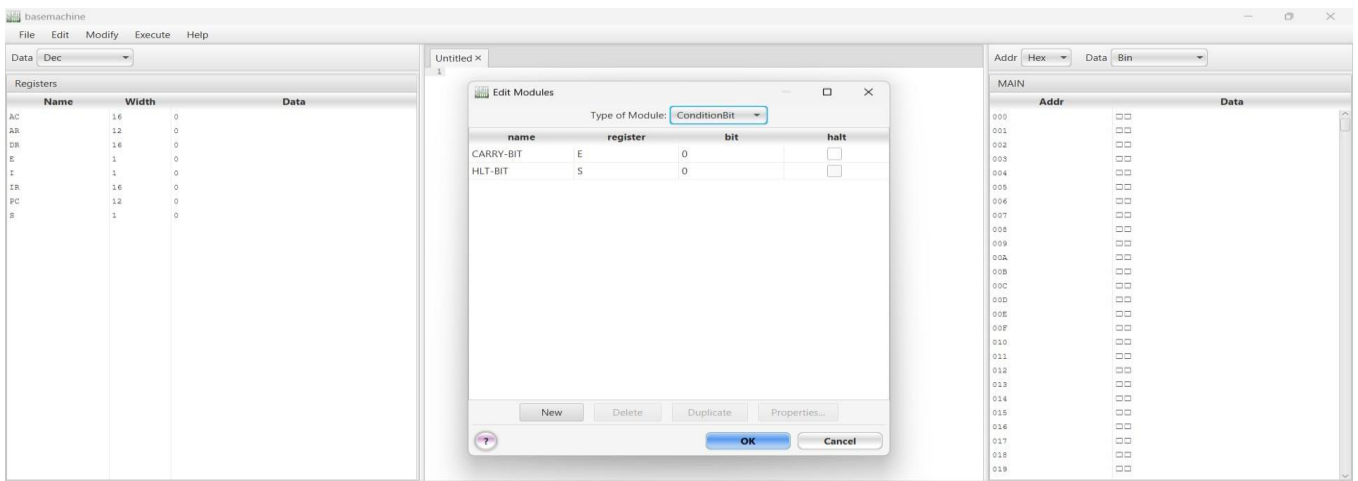
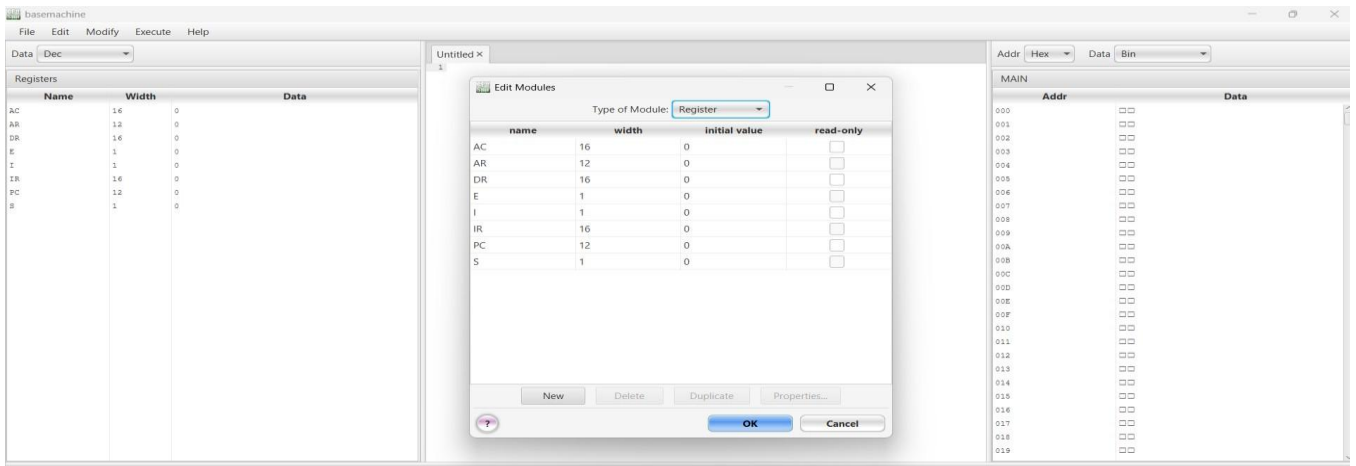
INDEX

S. No.	Topics
1.	Create a machine based on the given architecture (Registers, Memory, Instruction Set).
2.	Create a Fetch routine of the instruction cycle.
3.	Write an assembly program to simulate ADD operation on two user-entered numbers.
4.	Write an assembly program to simulate SUBTRACT operation on two user-entered numbers.
5.	Write an assembly program to simulate the following logical operations on two user-entered numbers: AND, OR, NOT, XOR, NOR, NAND.
6.	Write assembly programs to simulate the following memory-reference instructions: ADD, LDA, STA, BUN, ISZ.
7.	Write programs to simulate register reference instructions and determine contents of AC, E, PC, AR, IR in decimal after execution: CLA, CMA, CME, HLT.
8.	Write programs for register reference instructions and determine contents of AC, E, PC, AR, IR in decimal after execution: INC, SPA, SNA, SZE.
9.	Write programs for CIR and CIL and determine contents of AC, E, PC, AR, IR in decimal after execution.
10.	Write a program that reads integers and adds them until a negative non-zero number is entered; then output the sum (excluding last number).
11.	Write a program that reads integers and adds them until zero is read; then output the sum.

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



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



Q2. Creating a FETCH Routine for the Instruction Cycle.

The screenshot shows the 'basemachine' application window. On the left, the 'Registers' table lists AC (16 bits), AR (12 bits), DR (16 bits), E (1 bit), I (1 bit), IR (16 bits), PC (12 bits), and S (1 bit). In the center, the 'Edit Microinstructions' dialog is open, showing a table of microinstructions:

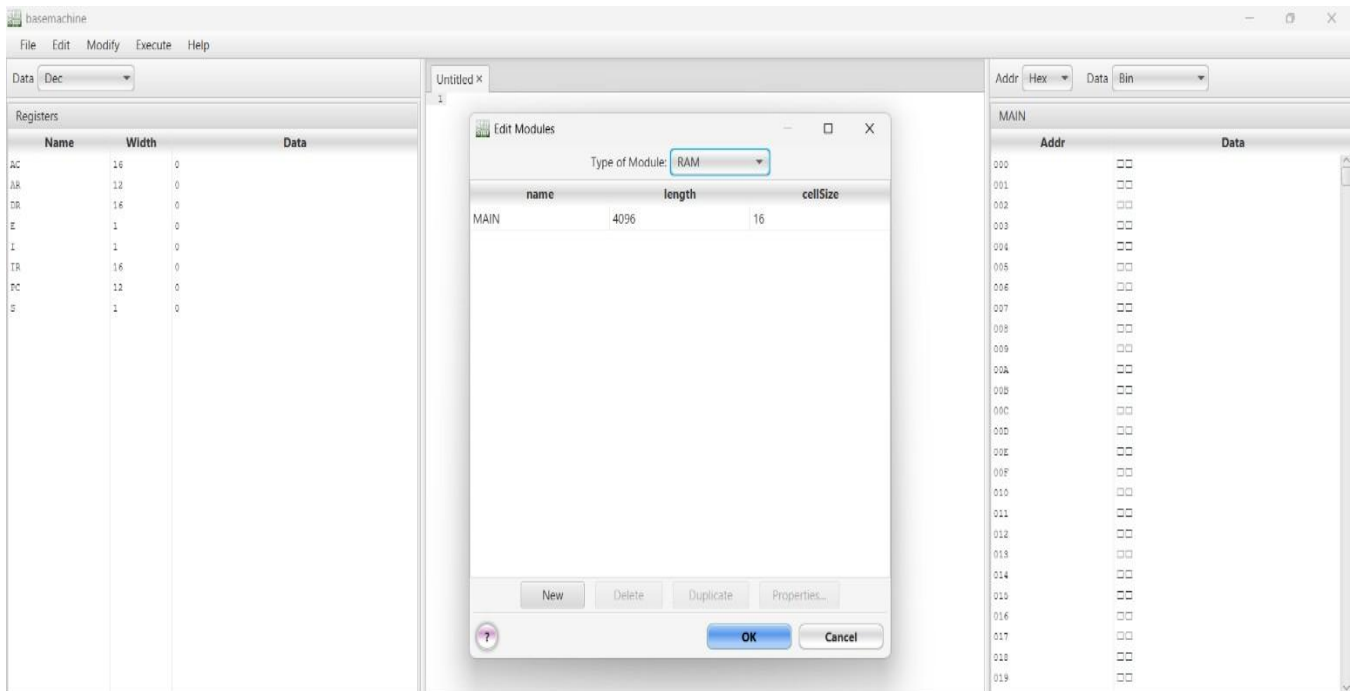
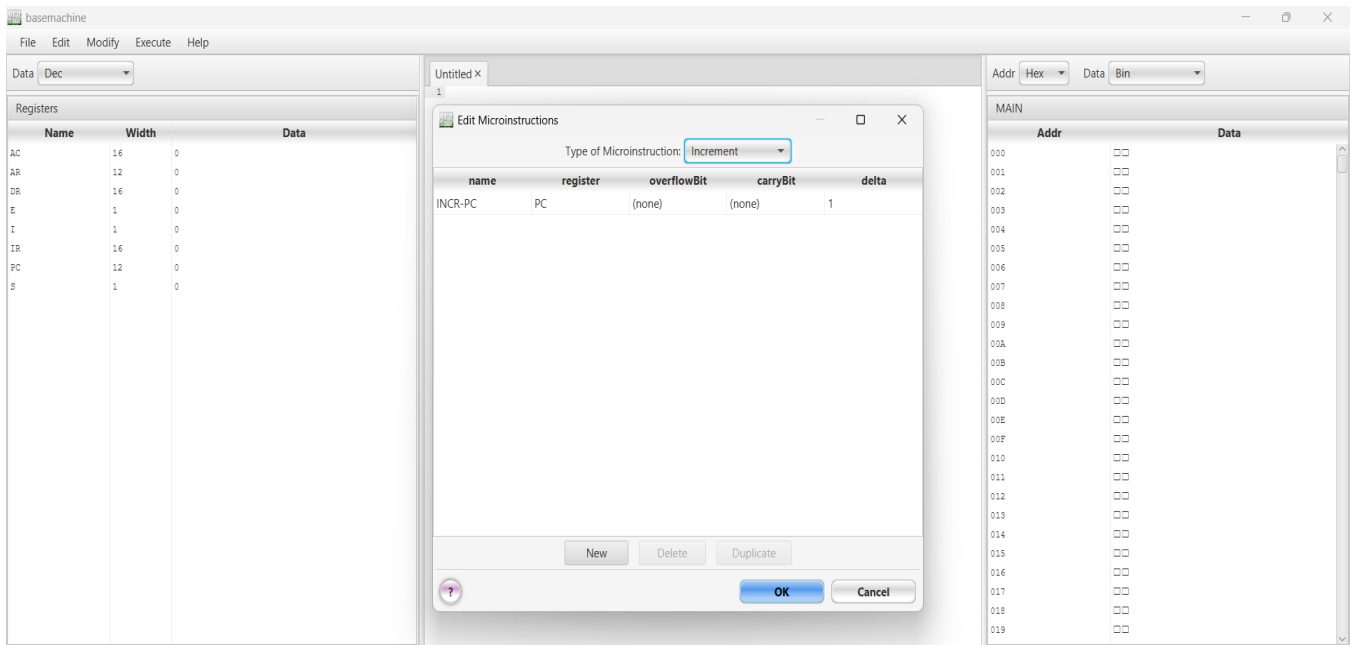
name	source	srcStartBit	dest	destStartBit	numBits
AR ← IR(4-15)	IR	4	AR	0	12
AR ← PC	PC	0	AR	0	12

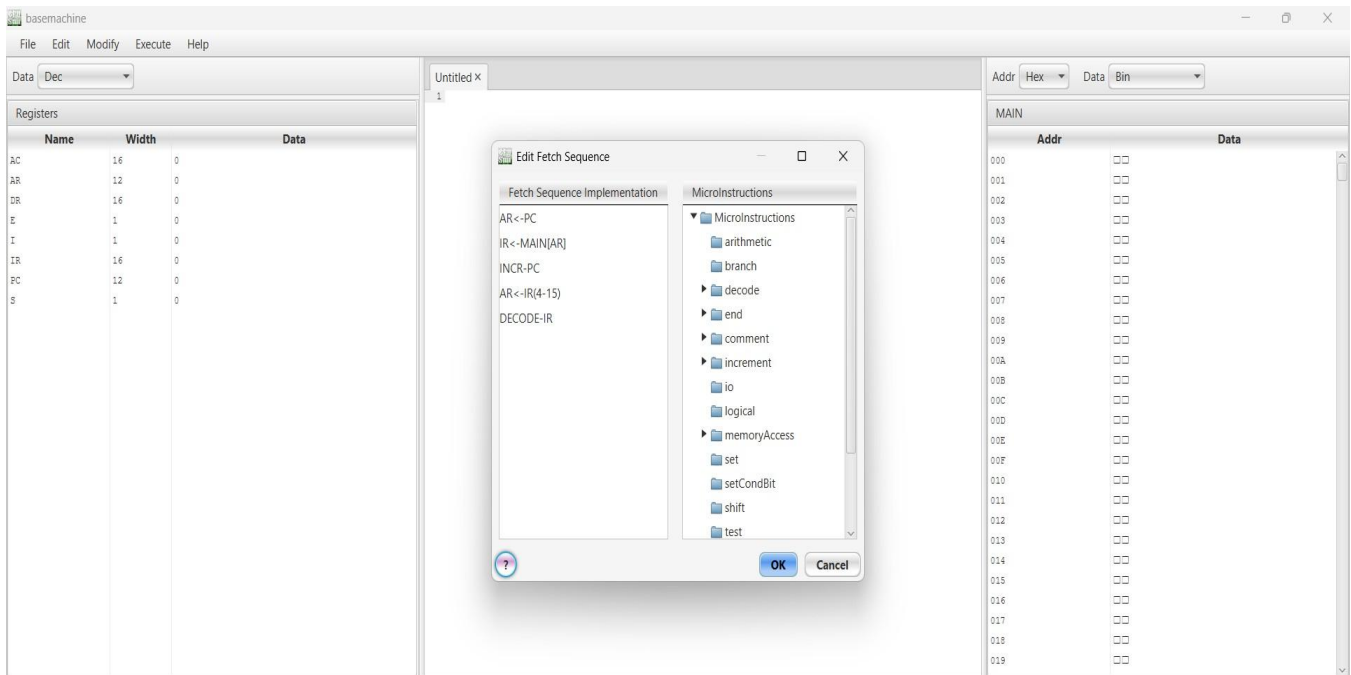
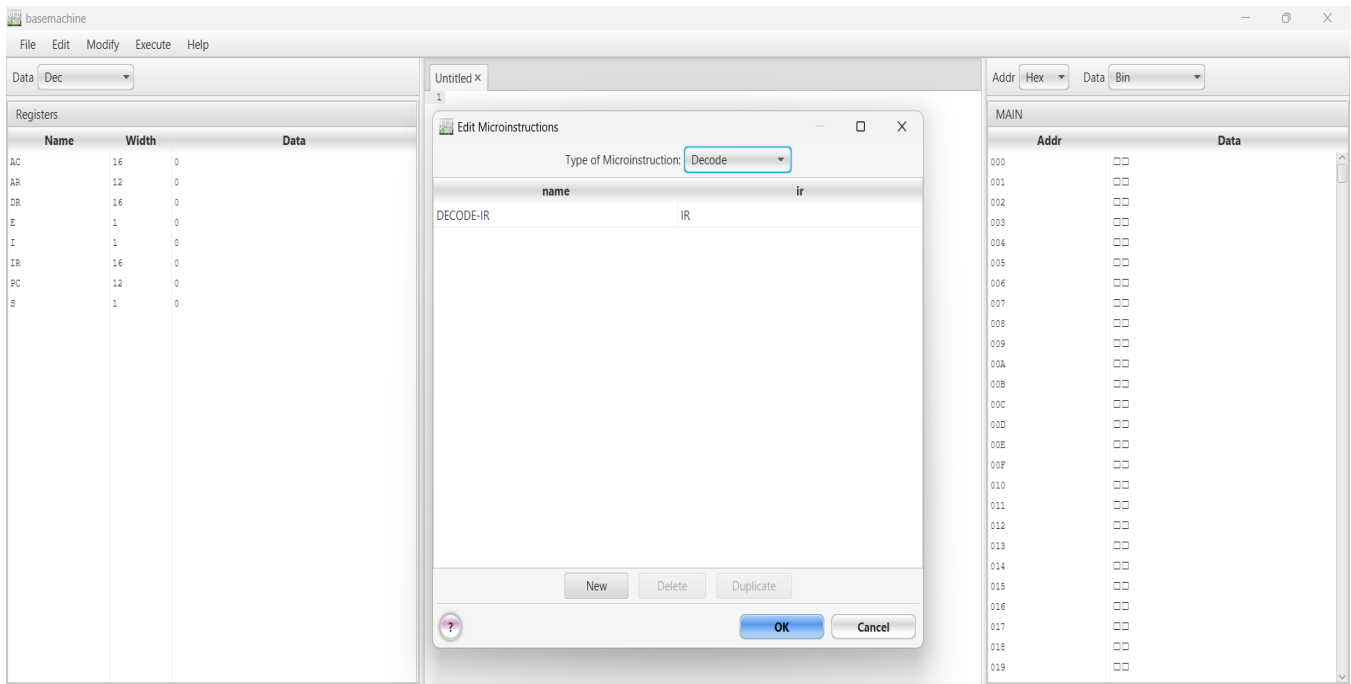
The 'Type of Microinstruction' dropdown is set to 'TransferRtoR'. On the right, the 'MAIN' memory window shows addresses 000 to 019, each with a data field represented by two hex boxes.

The screenshot shows the 'basemachine' application window. On the left, the 'Registers' table is the same as in the previous image. In the center, the 'Edit Microinstructions' dialog is open, showing a table of microinstructions:

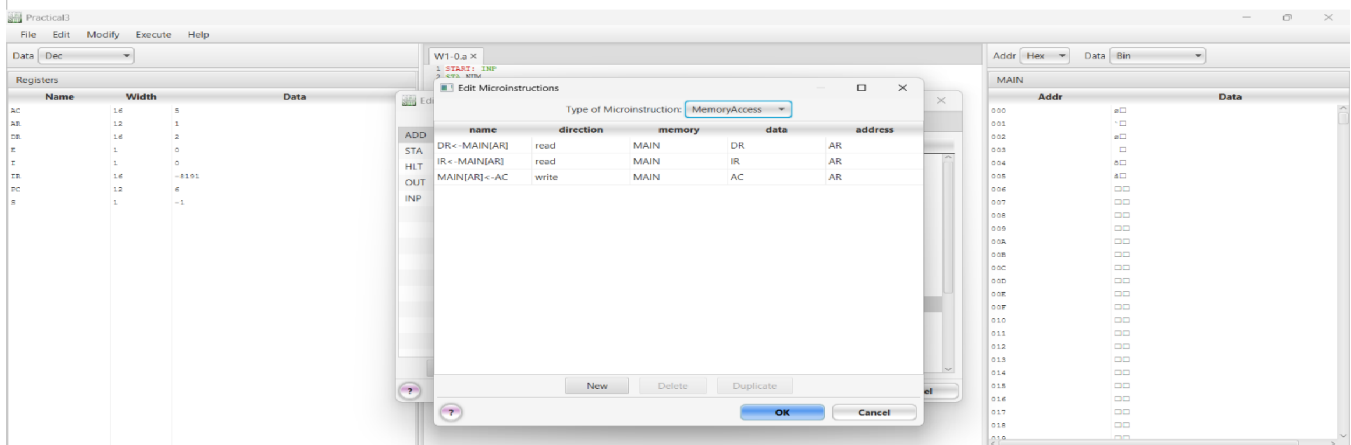
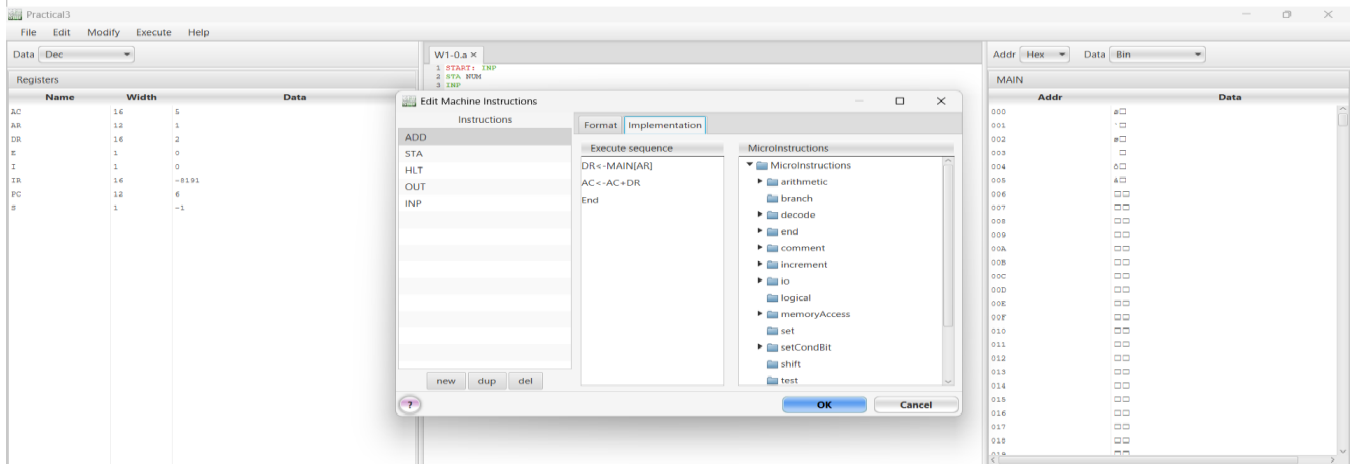
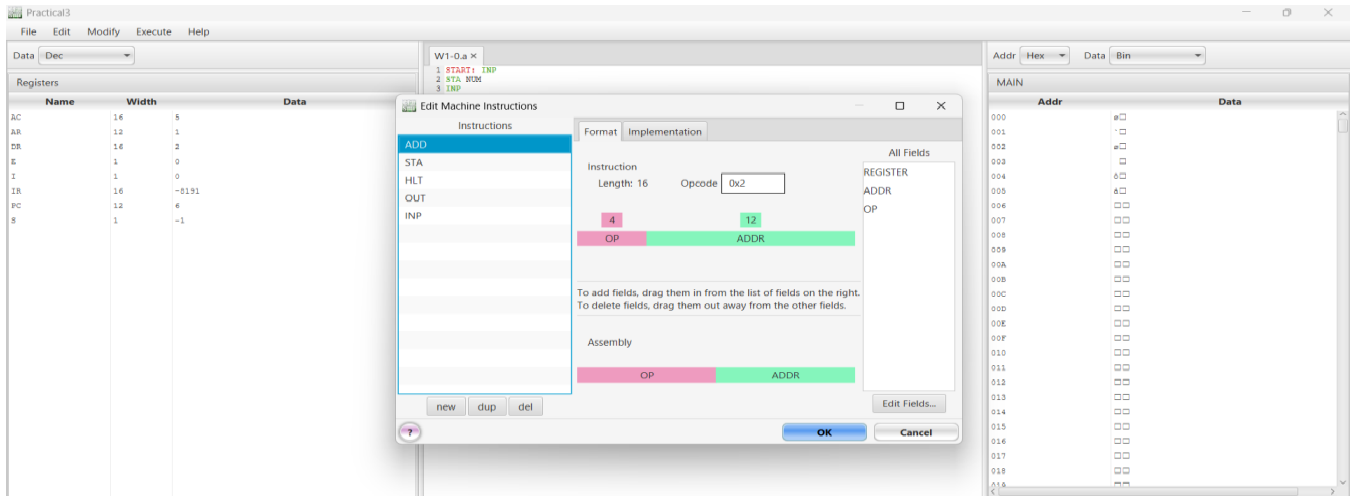
name	direction	memory	data	address
IR ← MAIN[AR]	read	MAIN	IR	AR

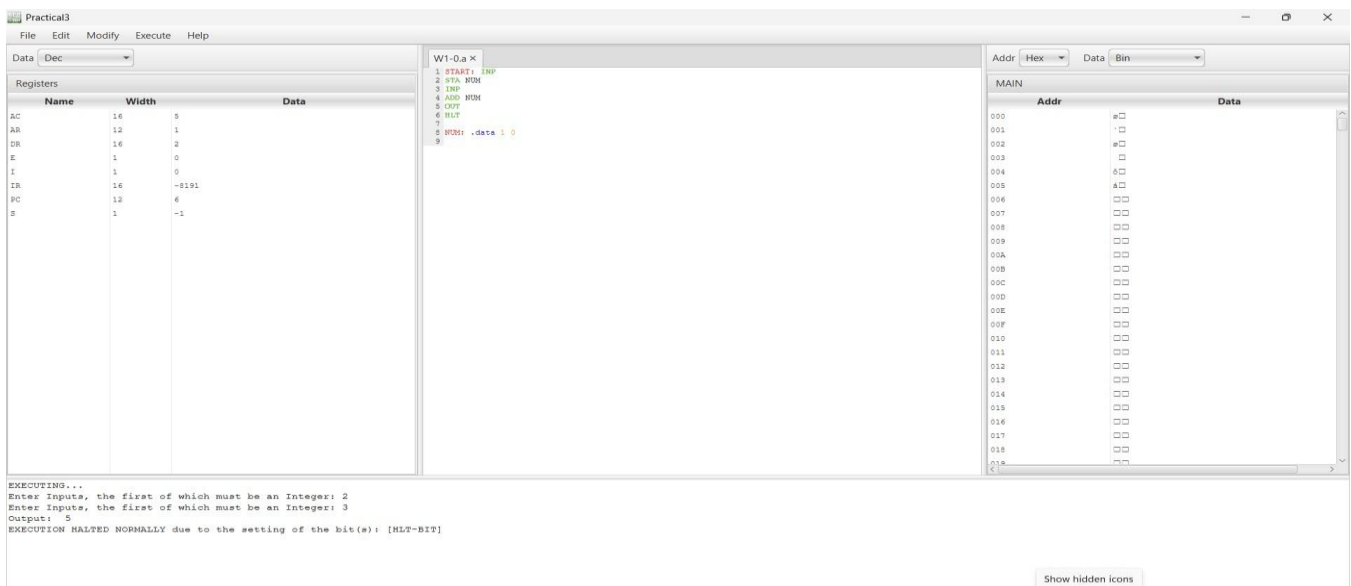
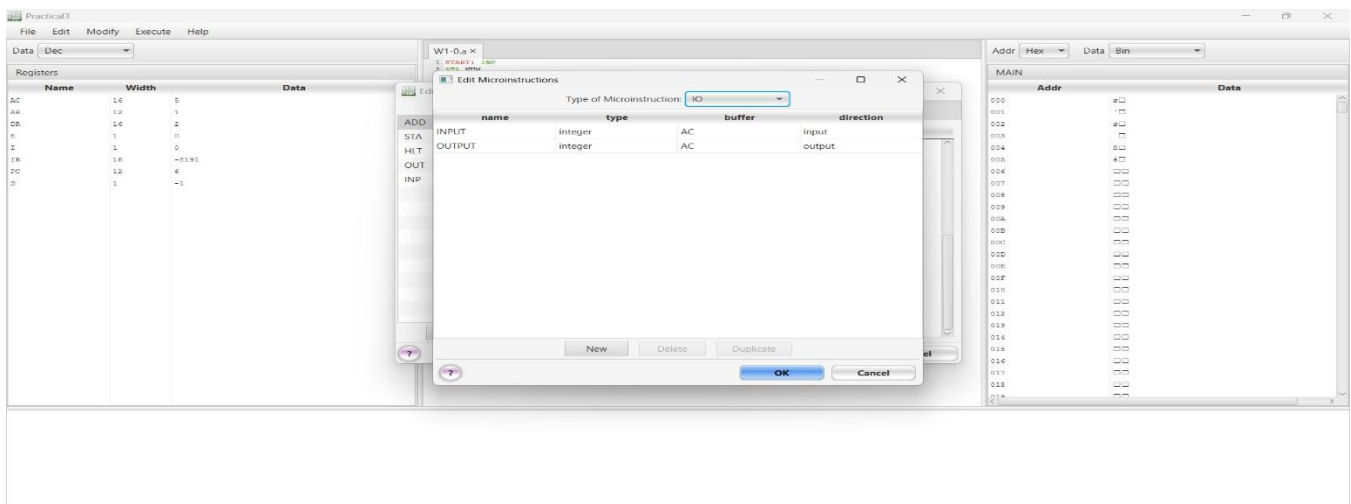
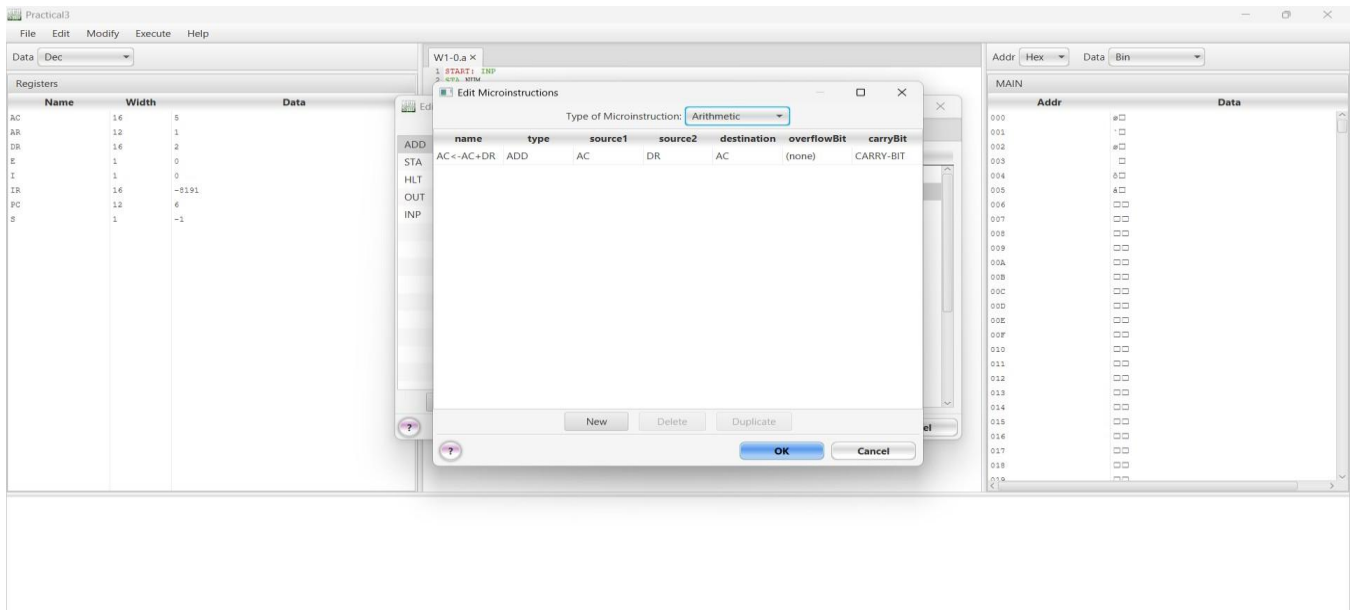
The 'Type of Microinstruction' dropdown is set to 'MemoryAccess'. On the right, the 'MAIN' memory window shows addresses 000 to 019, each with a data field represented by two hex boxes.



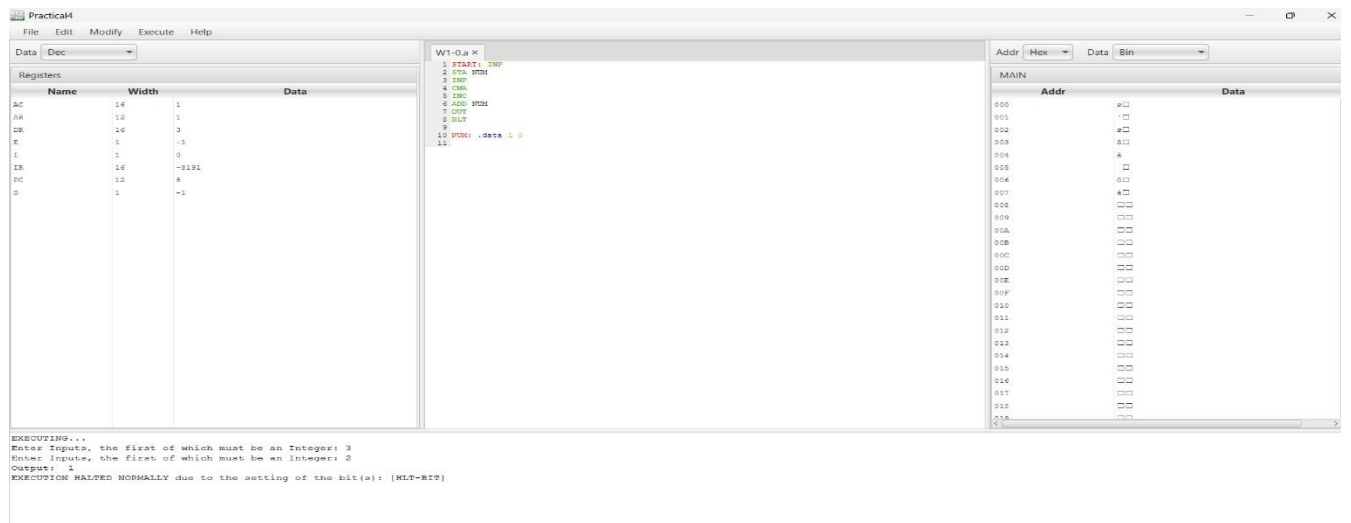
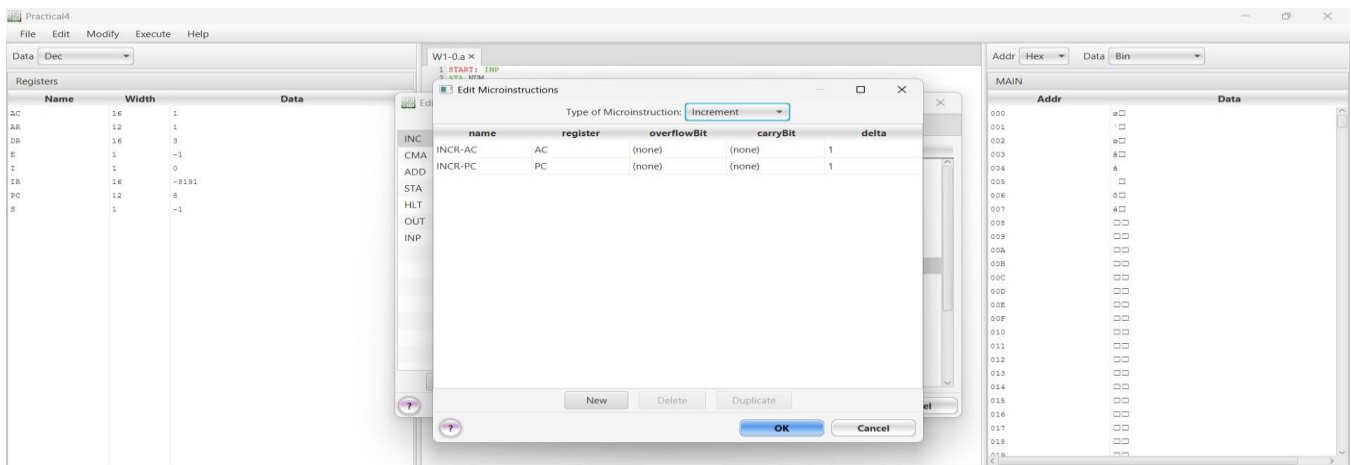
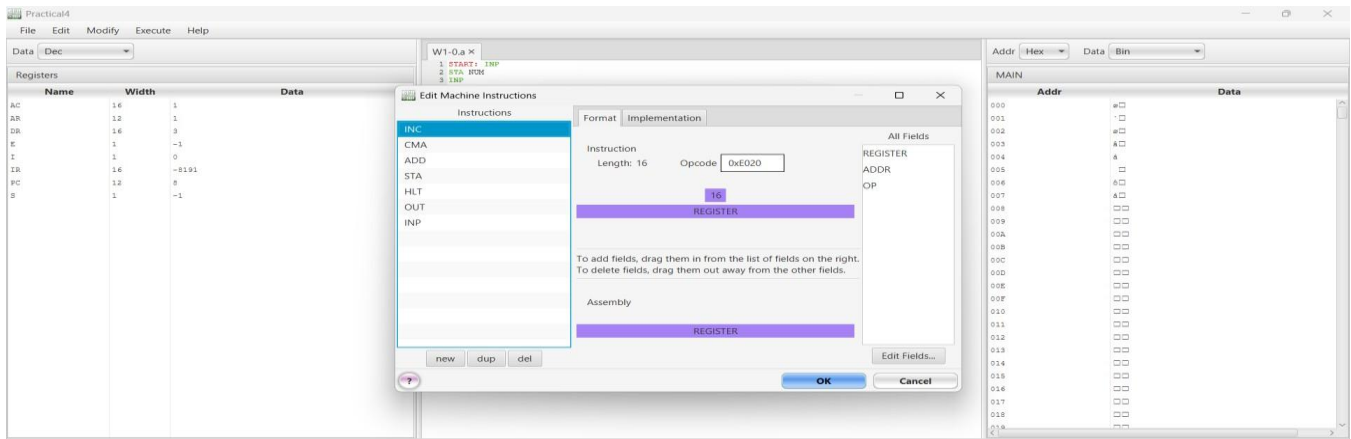


Q. 3. Write an Assembly Program to simulate ADD operation on two user entered numbers.

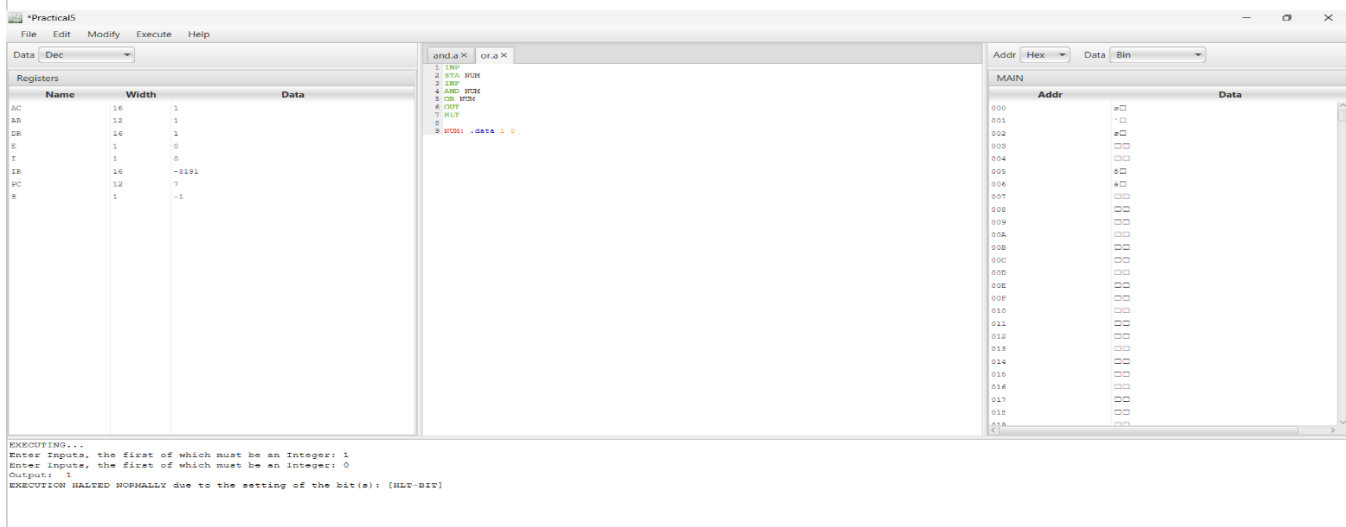
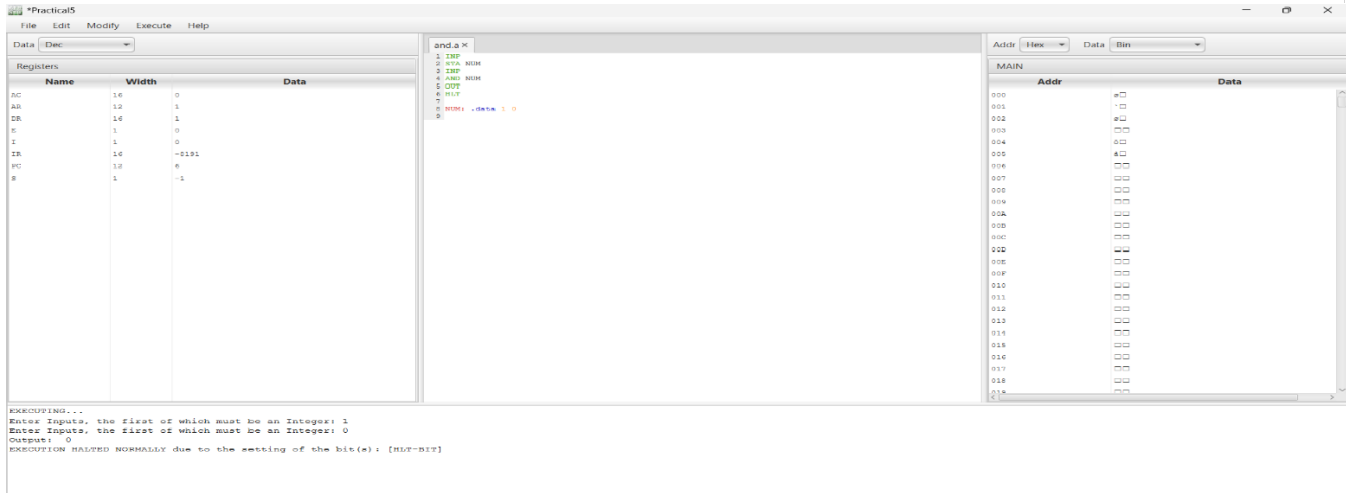
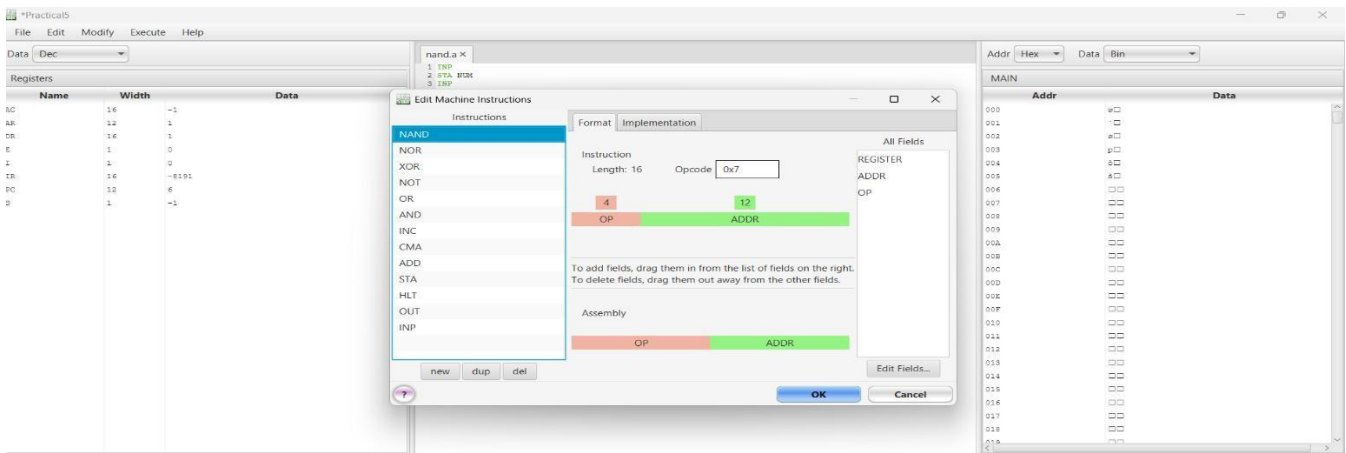




Q. 4. Write an Assembly Program to simulate SUBTRACT operation on two user entered numbers.



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

nota X

```

1 IMP
2 NOT
3 OUT
4 HLT
5

```

Addr: Hex Data: Bin

MAIN

Addr	Data
000	
001	
002	
003	
004	
005	
006	
007	
008	
009	
00A	
00B	
00C	
00D	
00E	
00F	
010	
011	
012	
013	
014	
015	
016	
017	
018	
019	

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

xora X

```

1 IMP
2 STA NUM
3 IMP
4 XOR NUM
5 OUT
6 HLT
7
8 NUM: .data 1 0

```

Addr: Hex Data: Bin

MAIN

Addr	Data
000	
001	
002	
003	
004	
005	
006	
007	
008	
009	
00A	
00B	
00C	
00D	
00E	
00F	
010	
011	
012	
013	
014	
015	
016	
017	
018	
019	

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

nora X

```

1 INP
2 STA MVM
3 INP
4 MVM MVM
5 OUT
6 HLT
7
8 MVM: .data 1 0

```

Addr Hex Data Bin

MAIN

Addr	Data
000	
001	
002	
003	
004	
005	
006	
007	
008	
009	
00A	
00B	
00C	
00D	
00E	
00F	
010	
011	
012	
013	
014	
015	
016	
017	
018	
019	

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]

```
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	-0191
PC	12	6
S	1	-1

nand.a X

```

1 INP
2 STA NUM
3 INP
4 NAND NUM
5 OUT
6 HLT
7
8 NUM: .data 1 0

```

MAIN

Addr	Hex	Data
000		
001		
002		
003		
004		
005		
006		
007		
008		
009		
00A		
00B		
00C		
00D		
00E		
00F		
010		
011		
012		
013		
014		
015		
016		
017		
018		
019		

EXECUTING...

Enter Inputs, the first of which must be an Integer: 1

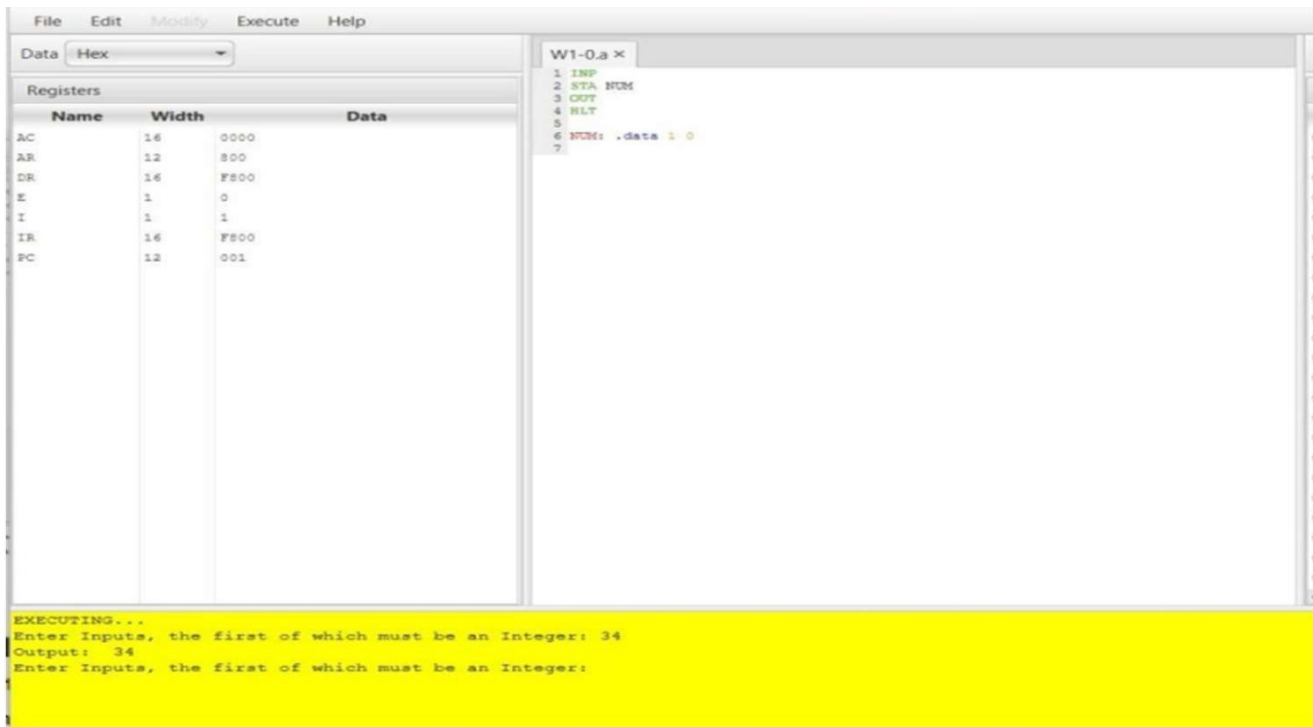
Enter Inputs, the first of which must be an Integer: 0

Output: -1

EXECUTION HALTED NORMALLY due to the setting of the bit(s): [HLT-BIT]

```
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.



The screenshot shows an assembly simulator window with a menu bar (File, Edit, Modify, Execute, Help) and a 'Data' dropdown set to 'Hex'. The 'Registers' table is as follows:

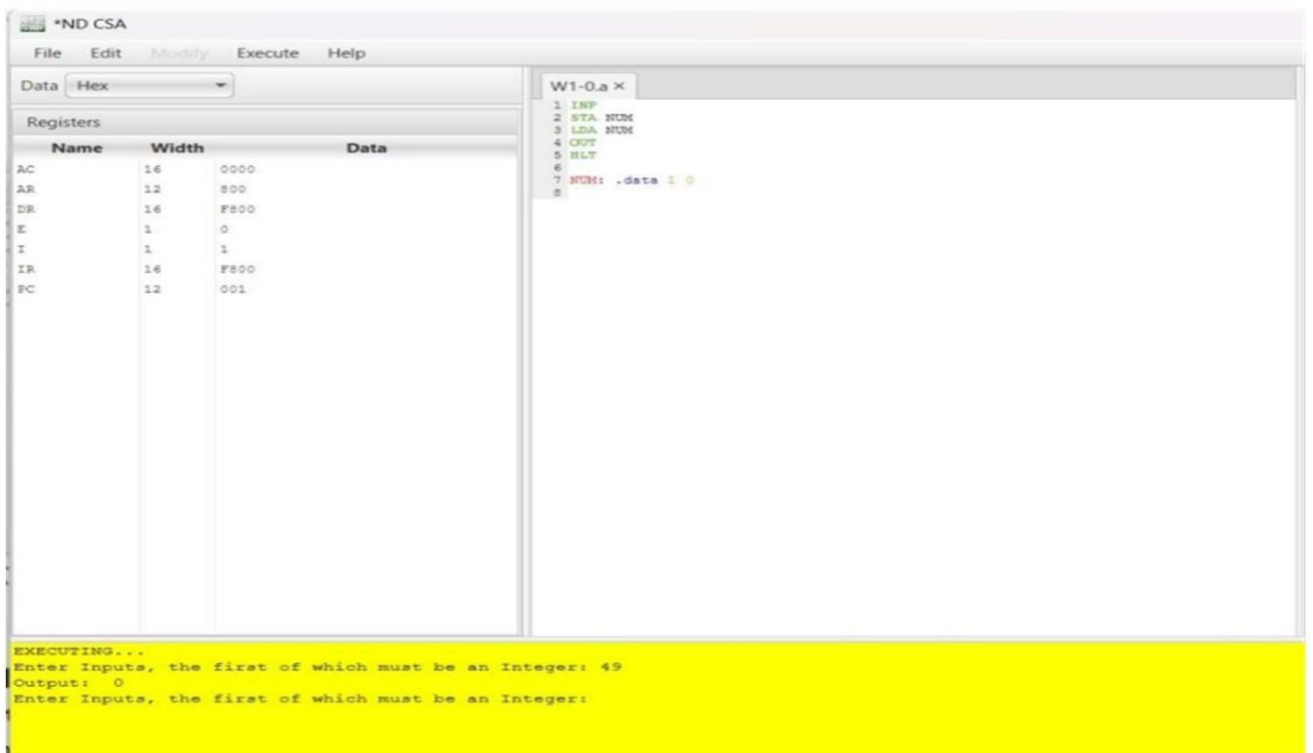
Name	Width	Data
AC	16	0000
AR	12	800
DR	16	F800
E	1	0
I	1	1
IR	16	F800
PC	12	001

The assembly code window (W1-0.a x) contains the following instructions:

```
1 INP
2 STA NUM
3 OUT
4 HLT
5
6 NUM: .data 1 0
7
```

The execution log at the bottom shows:

```
EXECUTING...
Enter Input, the first of which must be an Integer: 34
Output: 34
Enter Input, the first of which must be an Integer:
```



The screenshot shows an assembly simulator window with a menu bar (File, Edit, Modify, Execute, Help) and a 'Data' dropdown set to 'Hex'. The 'Registers' table is as follows:

Name	Width	Data
AC	16	0000
AR	12	800
DR	16	F800
E	1	0
I	1	1
IR	16	F800
PC	12	001

The assembly code window (W1-0.a x) contains the following instructions:

```
1 INP
2 STA NUM
3 LDA NUM
4 OUT
5 HLT
6
7 NUM: .data 1 0
8
```

The execution log at the bottom shows:

```
EXECUTING...
Enter Input, the first of which must be an Integer: 49
Output: 0
Enter Input, the first of which must be an Integer:
```

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 x

```

1 INP
2 DOW K
3 INP
4 W1 OUT
5 HLT
6

```

Addr Hex Data H

MAIN

Addr	Data
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 Input, 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	240

W1-0.a x

```

1 ISZ 009
2 OUT
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 x

```

1 INP
2 INC
3 OUT
4 HLT
5

```

EXECUTING...

Enter Inputs, the first of which must be an Integer: -2

Output: -2

*ND CSA

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 x

```

1 INP
2 SPA
3 OUT
4 HLT
5

```

Addr	Hex	Data
MAIN		
000		F800
001		7010
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: -3

Output: -3

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 x

```

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	400
I	1	1
IR	16	F800
PC	12	002
AC	16	6000
DR	16	F800

W1-0.a x

```

1 INP
2 SNA
3 OUT
4 HLT
5

```

MAIN

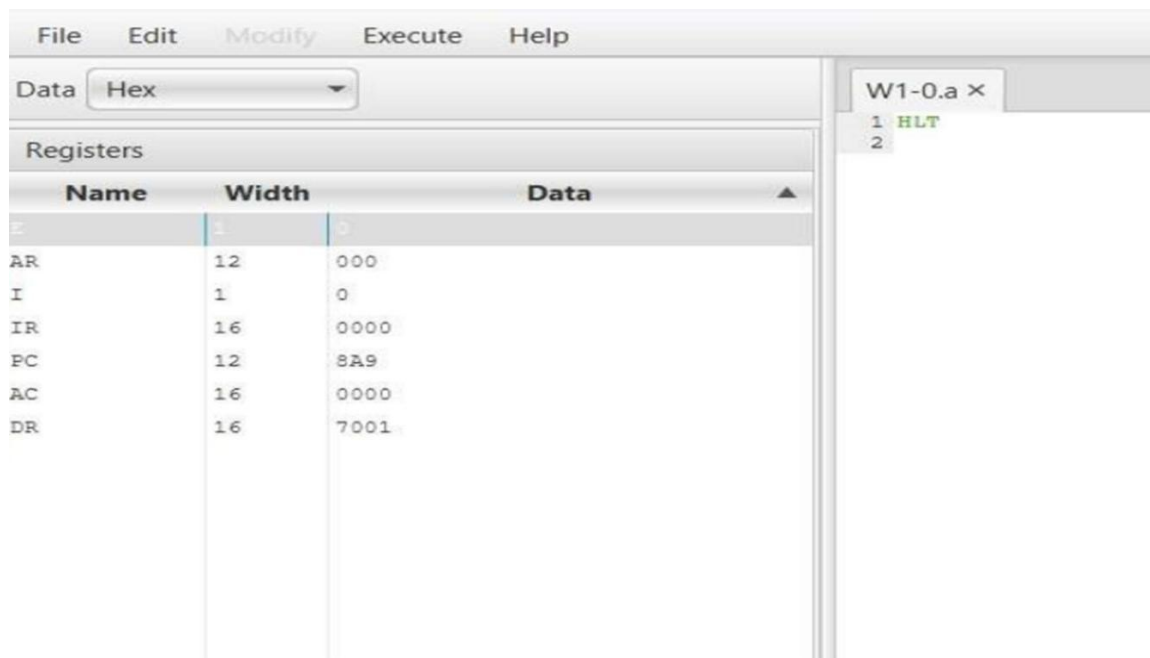
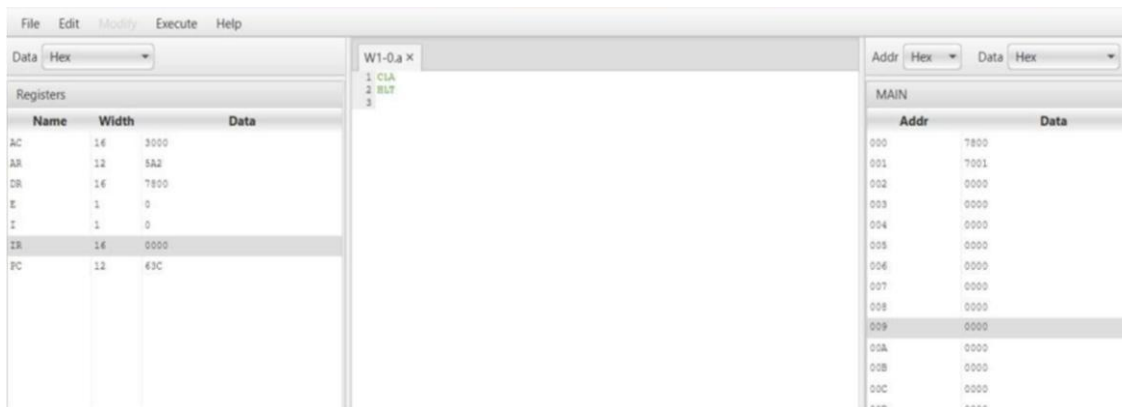
Addr	Data
000	F800
001	7002
002	F400
003	7001
004	0000
005	0000
006	0000
007	0000
008	0000
009	0000
00A	0000
00B	0000
00C	0000
00D	0000
00E	0000
00F	0000
010	0000
011	0000
012	0000
013	0000
014	0000
015	0000
016	0000

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

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

W1-0.a x register instructions x

1. CLR

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

Registers

Name	Width	Data
AC	16	FFFF
AR	12	000
DR	16	0000
E	1	0
I	1	0
IR	16	7200
PC	12	001

W1-0.a x register instructions x

1. ORA

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.

The screenshot shows an assembly simulator window with a menu bar (File, Edit, Modify, Execute, Help) and a toolbar. The 'Data' dropdown is set to 'Hex'. The 'Registers' panel on the left lists the following registers and their values:

Name	Width	Data
E	1	0
AR	12	000
I	1	0
IR	16	0000
PC	12	CSE
AC	16	7000
DR	16	F800

The assembly code window in the center shows the following instructions:

```
1 INP
2 CIR
3 OUT
4 HLT
5
```

On the right, the 'MAIN' memory segment is displayed with addresses from 000 to 016, all containing the value 0000.

The status bar at the bottom indicates 'EXECUTING...' and prompts the user to 'Enter Inputs, the first of which must be an Integer: 2'. The output shown is 'Output: 2'.

The screenshot shows the same assembly simulator window after executing the instructions. The 'Registers' panel now shows the following values:

Name	Width	Data
E	1	1
AR	12	800
I	1	1
IR	16	F800
PC	12	001
AC	16	7000
DR	16	F800

The assembly code window remains the same as in the previous screenshot.

The status bar at the bottom indicates 'EXECUTING...' and prompts the user to 'Enter Inputs, the first of which must be an Integer: 8'. The output shown is 'Output: 8'.

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

The screenshot shows an assembly editor with a menu bar (File, Edit, Modify, Execute, Help) and a toolbar. The 'Data' dropdown is set to 'Hex'. The 'Registers' panel on the left lists registers E, AR, I, IR, PC, AC, and DR with their widths and current values. The main editor window displays the following assembly code:

```
*W1-0.a x
1  START: INP
2
3      JMPN DONE
4
5      ADD SUM
6
7      STA SUM
8
9      JUMP START
10
11 DONE: LDA SUM
12
13      OUT
14
15      HLT
16
17 SUM: .data 2 0
18
```

Name	Width	Data
E	1	0
AR	12	000
I	1	0
IR	16	0000
PC	12	000
AC	16	0000
DR	16	0000

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

The screenshot shows an assembly program execution environment. The top menu bar includes File, Edit, Modify, Execute, and Help. Below the menu is a 'Data' section with a 'Hex' dropdown. The 'Registers' section displays a table of registers and their values.

Name	Width	Data
E	1	0
AR	12	666
I	1	0
IR	16	0000
PC	12	703
AC	16	C000
DR	16	F800

The 'W1-0.a x' window shows the following assembly code:

```
1 START: INP
2
3     JMPN DONE
4
5     ADD SUM
6
7     STA SUM
8
9     JUMP START
10
11 DONE: LDA SUM
12
13     OUT
14
15     HLT
16
17 SUM: .data 2 0
18
```

The bottom status bar displays the following text:

```
EXECUTING...
Enter Inputs, the first of which must be an Integer: 2
Enter Inputs, the first of which must be an Integer: -2
```

IMPORTANT MICRO INSTRUCTIONS :

Edit Microinstructions

Type of Microinstruction: Set

name	register	start	numBits	value
AC <- 0	AC	0	16	0
E <- 0	E	0	1	0

New Delete Duplicate

OK Cancel

Edit Microinstructions

Type of Microinstruction: Logical

name	type	source1	source2	destination
AC <- AC'	NOT	AC	AC	AC
E <- E'	NOT	E	E	E

New Delete Duplicate

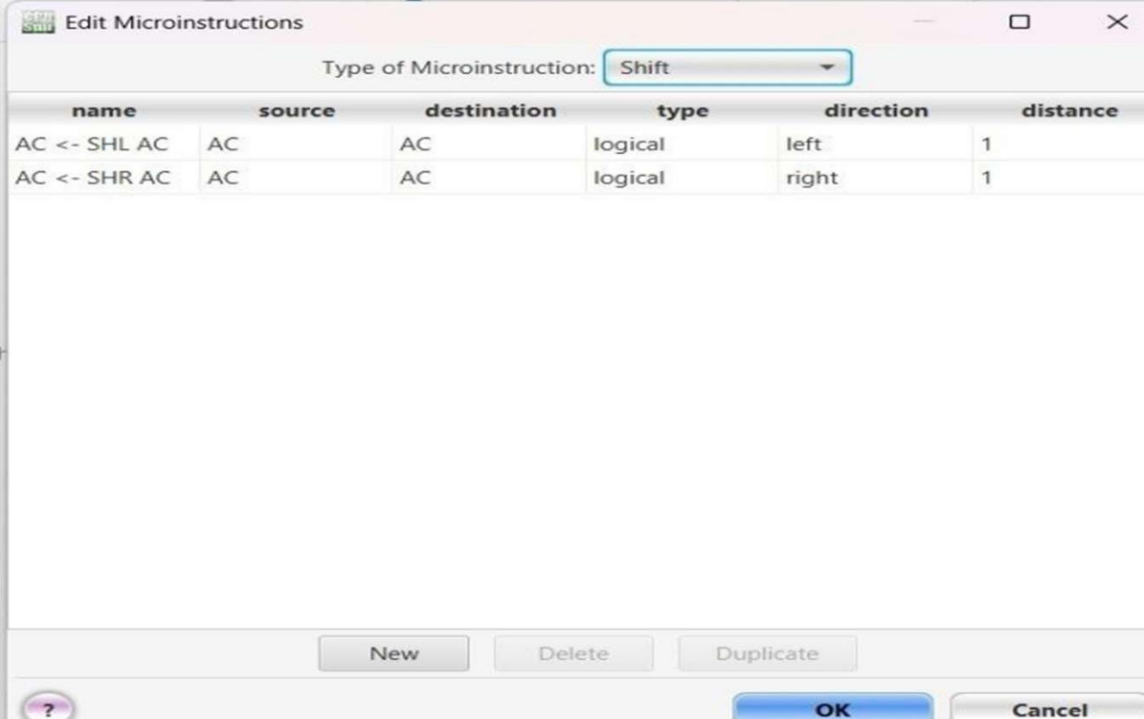
OK Cancel

Addr Dec Da

MAIN

Addr

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22



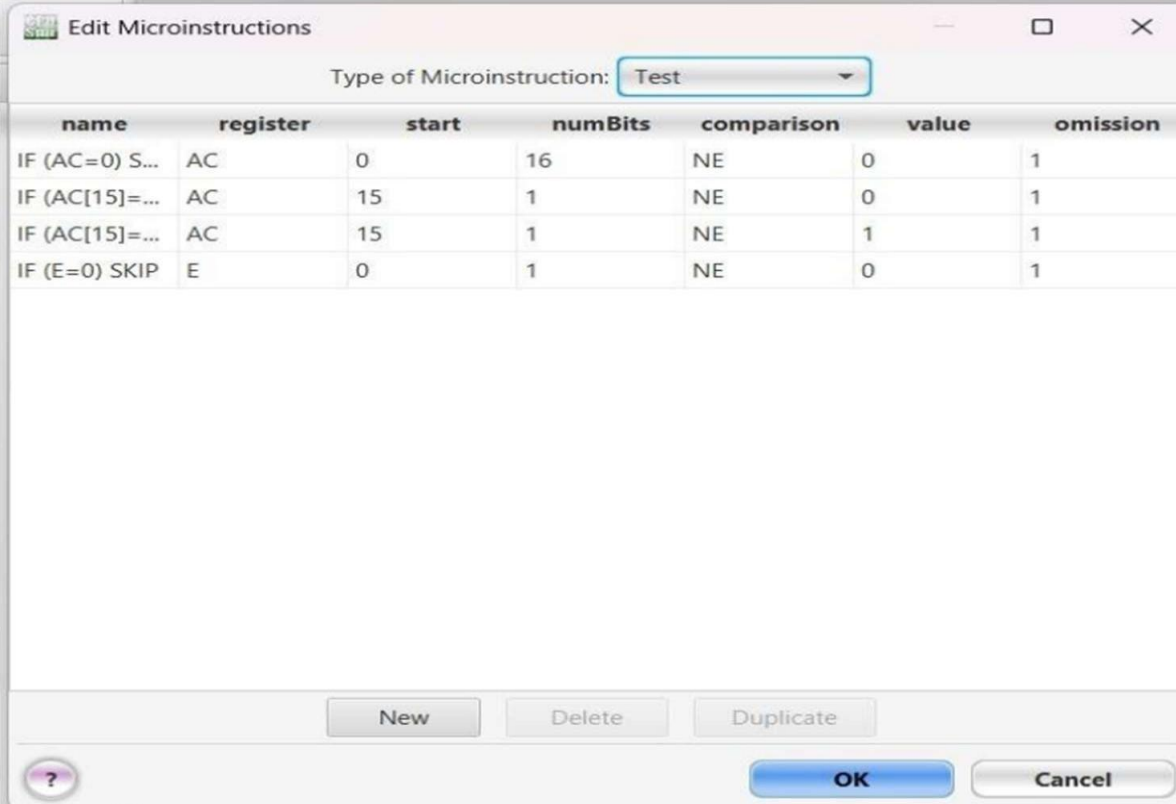
Edit Microinstructions

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