

PRACTICAL FILE

COMPUTER SYSTEM

ARCHITECTURE

BSc(H) Computer Science
FIRST SEMESTER

SUBMITTED BY:

ACHALA SINGH

22/CS/01

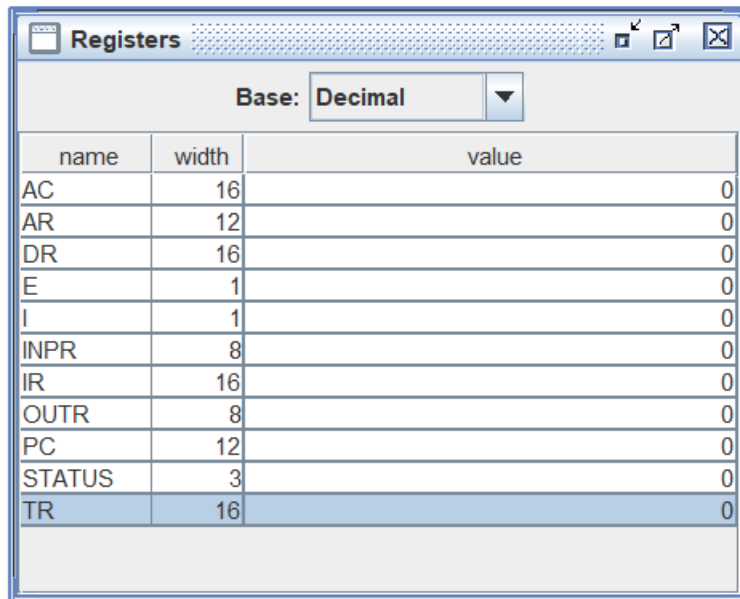
SUBMITTED TO:

PROF. JITENDRA

SINGH

QUESTION 1:CREATE A MACHINE DESIGNING THE REGISTER SET,MEMORY AND THE INSTRUCTION SET.

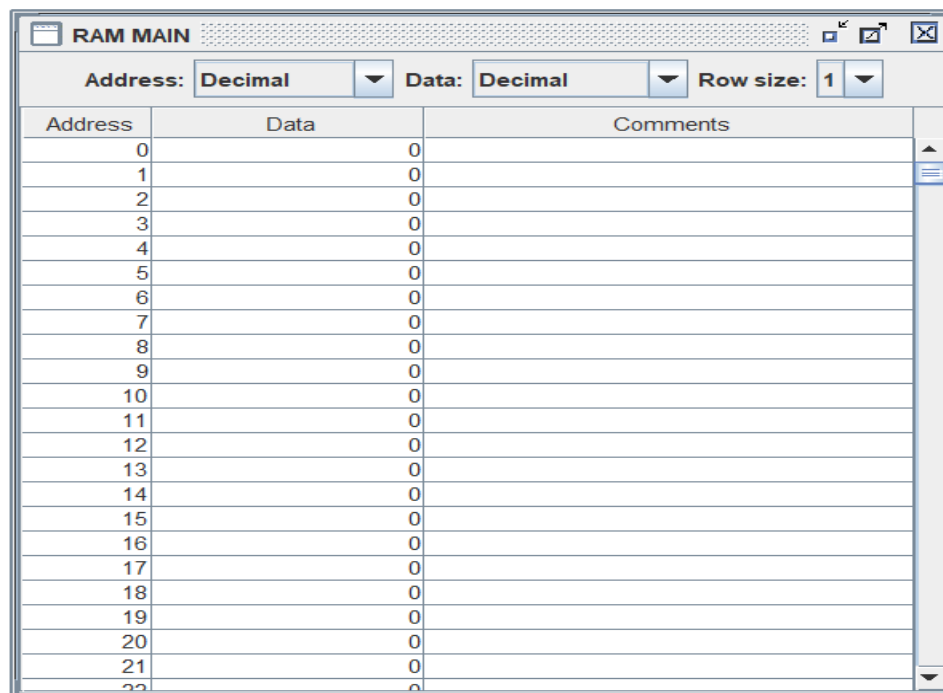
REGISTER:



The Registers window displays a table of registers. The 'Base' is set to 'Decimal'. The table has three columns: 'name', 'width', and 'value'. The registers listed are AC, AR, DR, E, I, INPR, IR, OUTR, PC, STATUS, and TR. The TR register is currently selected.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0
TR	16	0

RAM:



The RAM MAIN window displays a memory address table. The 'Address' and 'Data' fields are set to 'Decimal', and the 'Row size' is set to '1'. The table has three columns: 'Address', 'Data', and 'Comments'. The addresses range from 0 to 22, and the data values are all 0.

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

QUESTION 2:CREATE A FETCH ROUTINE OF THE INSTRUCTION CYCLE.

Edit the machine's fetch sequence

implementation

```

PC TO AR
M[AR] TO IR
INC PC
IR(4-15) TO AR
DECODE IR
End

```

<<insert<<

>>delete

All micros

- TransferRtoR
 - IR(4-15) TO AR
 - PC TO AR
- TransferAtoR
- TransferRtoA
- Set
- Test
- Increment
 - INC PC
- Arithmetic
- Shift
- Branch
- Logical
- Decode
 - DECODE IR
- MemoryAccess
- IO
- SetCondBit
- End
- Comment

Help

OK

Cancel

Type of Microinstruction: **TransferRtoR**

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

Type of Microinstruction: **MemoryAccess**

name	direction	memory	data	address
AC TO M[AR]	write	MAIN	AC	AR
M[AR] TO DR	read	MAIN	DR	AR
M[AR] TO IR	read	MAIN	IR	AR

Type of Microinstruction: **Increment**

name	register	overflowBit	delta
INC PC	PC	(none)	1

Type of Microinstruction: Decode	
name	ir
DECODE IR	IR

QUESTION 3:WRITE AN ASSEMBLY PROGRAM TO STIMULATE ADD OPERATION ON TWO USER-ENTERED NUMBERS.

ANS: ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

START: INP (TAKES INPUT FROM USER AND STORE IT IN AC)

STA NUM (M(AR) <- AC)

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

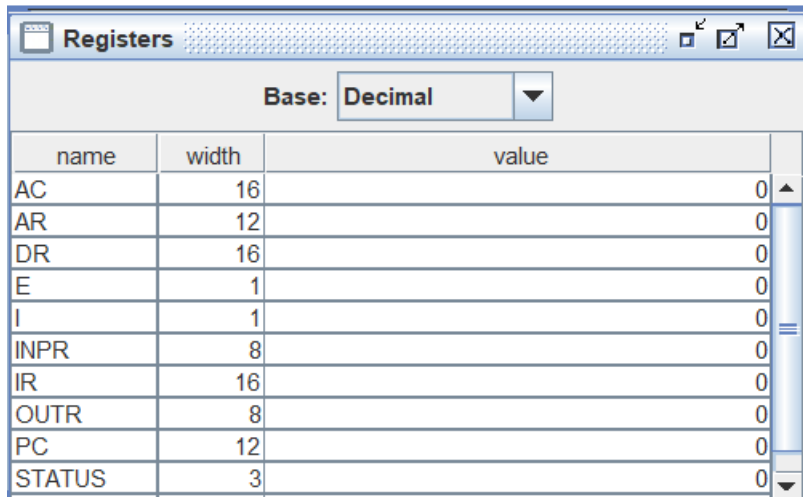
ADD NUM (DR <- M(AR) & AC <- AC + DR)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT-BIT = 1)

NUM: .data 1 0

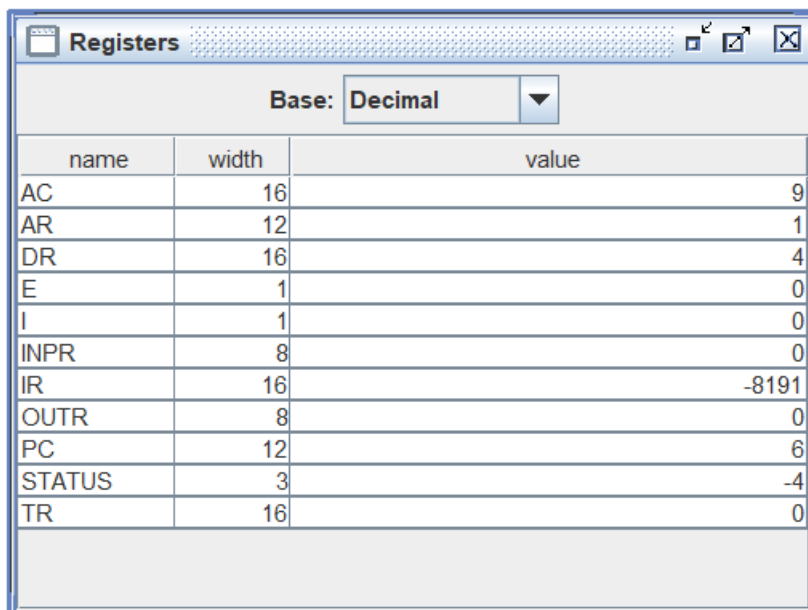
VALUE OF REGISTER BEFORE EXECUTION:



The Registers window shows the state of various registers before execution. The base is set to Decimal. All registers have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

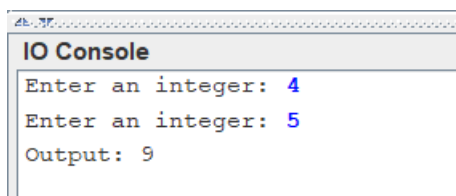
VALUE OF REGISTER AFTER EXECUTION:



The Registers window shows the state of various registers after execution. The base is set to Decimal. The values have changed significantly.

name	width	value
AC	16	9
AR	12	1
DR	16	4
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	6
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



The IO Console window shows the input and output of the program. It displays two prompts for integers and the resulting output.

```
IO Console
Enter an integer: 4
Enter an integer: 5
Output: 9
```

QUESTION 4:WRITE AN ASSEMBLY PROGRAM TO STIMULATE SUBTRACT OPERATION ON TWO USER-ENTERED NUMBERS.

ANS: ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

START: INP (TAKES INPUT FROM USER AND STORE IT IN AC)

STA NUM (M(AR) <- AC)

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

CMA (AC <- AC')

INC (AC <- AC+1)

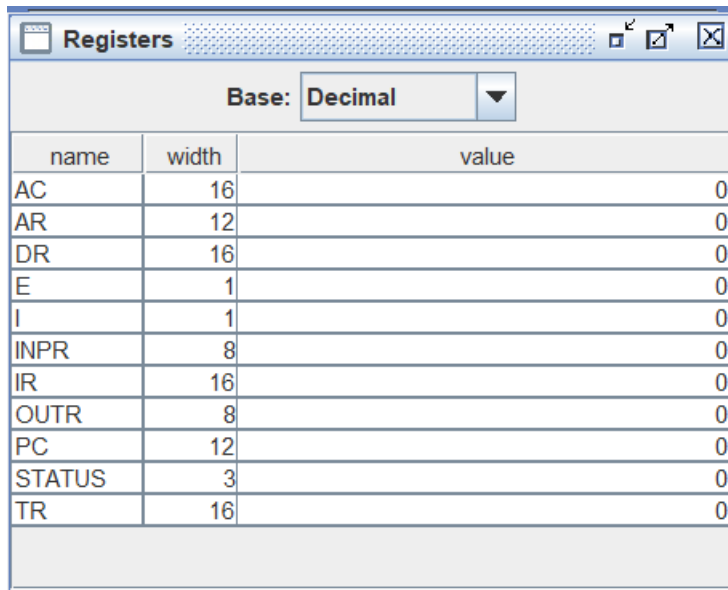
ADD NUM (DR <- M(AR) & AC <- AC + DR)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT-BIT = 1)

NUM: .data 1 0

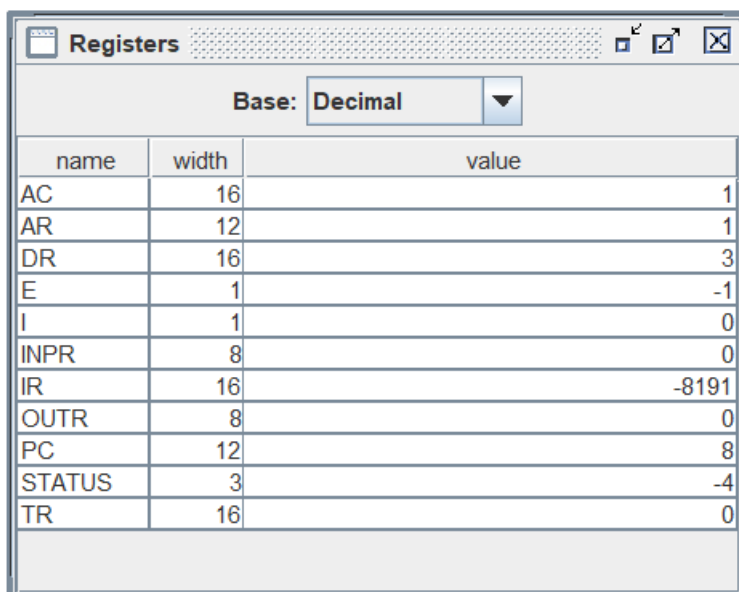
VALUE OF REGISTERS BEFORE EXECUTION:



The Registers window shows the initial state of the processor registers. The base is set to Decimal. All registers contain the value 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0
TR	16	0

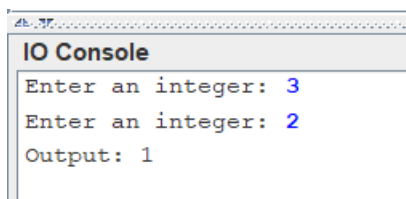
VALUE OF REGISTERS AFTER EXECUTION:



The Registers window shows the state of the processor registers after execution. The base is set to Decimal. The values have changed as follows:

name	width	value
AC	16	1
AR	12	1
DR	16	3
E	1	-1
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	8
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



The IO Console window displays the input and output of the program. It shows two prompts for integers and the resulting output.

```
Enter an integer: 3
Enter an integer: 2
Output: 1
```

QUESTION 5:WRITE AN ASSEMBLY PROGRAM TO SIMULATE THE FOLLOWING LOGICAL OPERATIONS ON TWO USER ENTERED NUMBERS.

- 1. AND**
- 2. OR**
- 3. NOT**
- 4. XOR**
- 5. NOR**
- 6. NAND**

ANS:1. AND

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

START: INP (TAKES INPUT FROM USER AND STORE IT IN AC)

STA NUM (M(AR) <- AC)

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

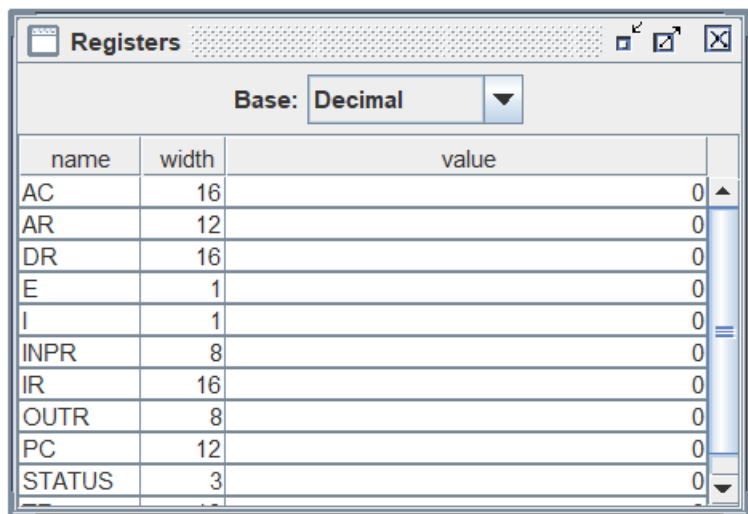
AND NUM (DR <- M(AR) & AC <- AC ^ DR)

**OUT (TAKES OUTPUT FROM AC AND DISPLAY
ON SCREEN)**

HLT (HALT-BIT = 1)

NUM: .data 1 0

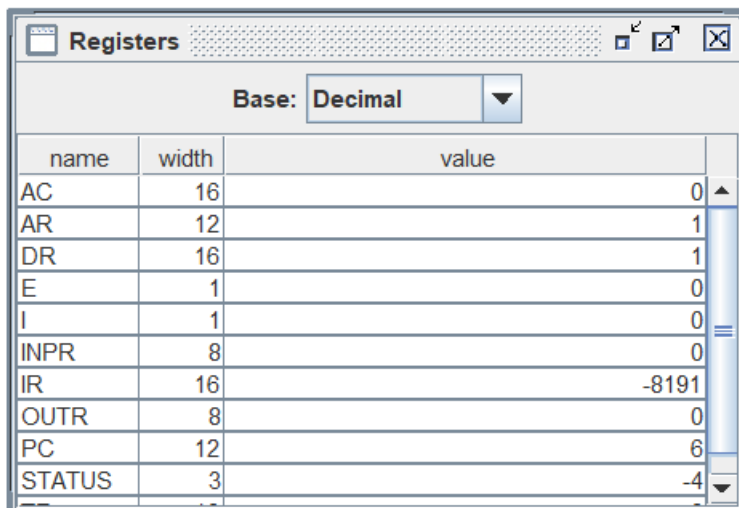
VALUE OF REGISTER BEFORE EXECUTION:



The image shows a 'Registers' window with a 'Base' dropdown set to 'Decimal'. It contains a table with 11 rows, each representing a register. All values are 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

VALUE OF REGISTER AFTER EXECUTION:



The image shows the same 'Registers' window after execution. The values have changed: AR is 1, DR is 1, IR is -8191, PC is 6, and STATUS is -4. All other registers remain at 0.

name	width	value
AC	16	0
AR	12	1
DR	16	1
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	6
STATUS	3	-4

INPUT OUTPUT WINDOW:

```
IO Console
Enter an integer: 1
Enter an integer: 0
Output: 0
```

2. OR

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

START: INP (TAKES INPUT FROM USER AND STORE IT IN AC)

STA NUM (M(AR) <- AC)

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

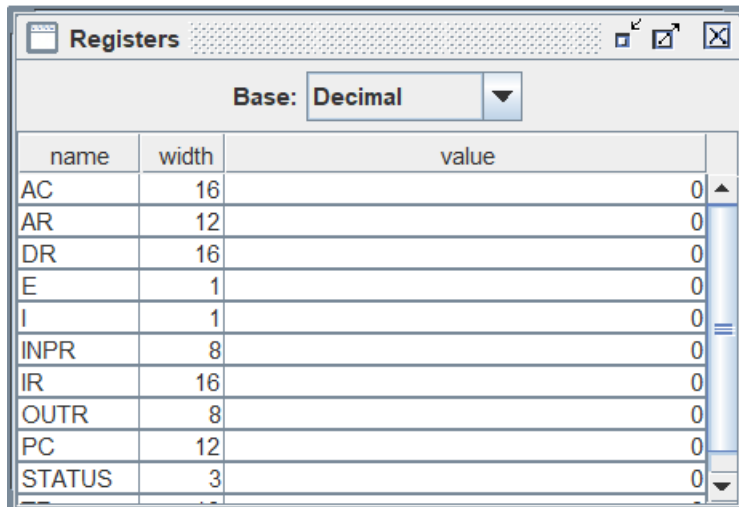
OR NUM (DR <- M(AR) & AC <- AC + DR)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT-BIT = 1)

NUM: .data 1 0

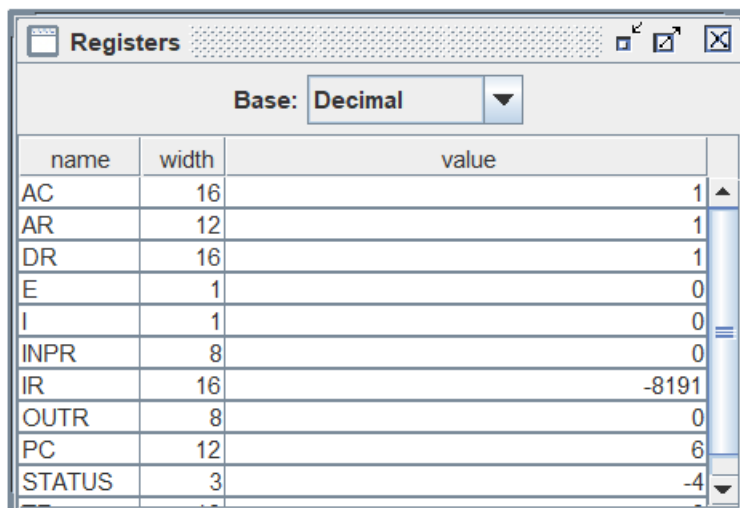
VALUE OF REGISTER BEFORE EXECUTION:



The Registers window shows the state of various registers before execution. The base is set to Decimal. All registers have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

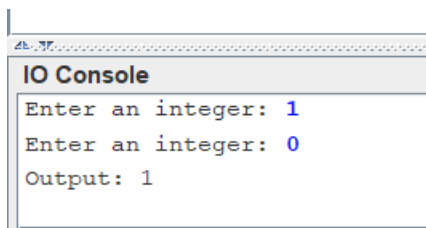
VALUE OF REGISTER AFTER EXECUTION:



The Registers window shows the state of various registers after execution. The base is set to Decimal. The values have changed: AC, AR, and DR are now 1; E, I, INPR, OUTR, and STATUS are 0; IR is -8191; and PC is 6.

name	width	value
AC	16	1
AR	12	1
DR	16	1
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	6
STATUS	3	-4

INPUT OUTPUT WINDOW:



The IO Console window shows the input and output of the program. It displays two prompts for integers and the resulting output.

```
IO Console
Enter an integer: 1
Enter an integer: 0
Output: 1
```

3.NOT

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

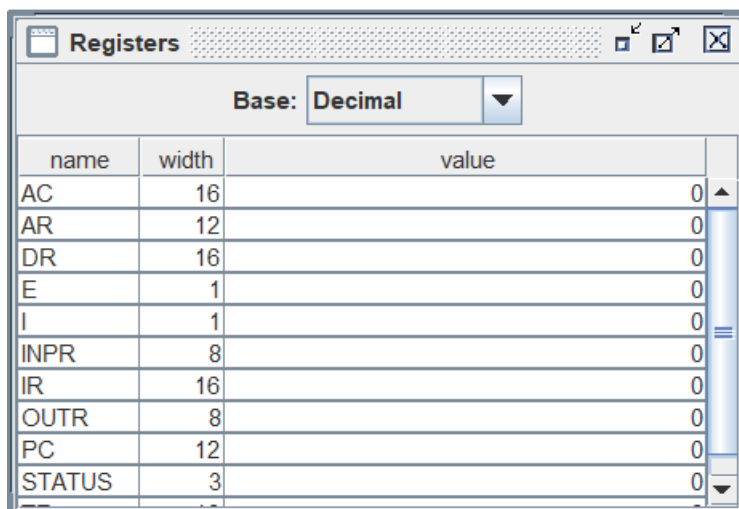
NOT NUM (AC <- AC')

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT-BIT = 1)

NUM: .data 1 0

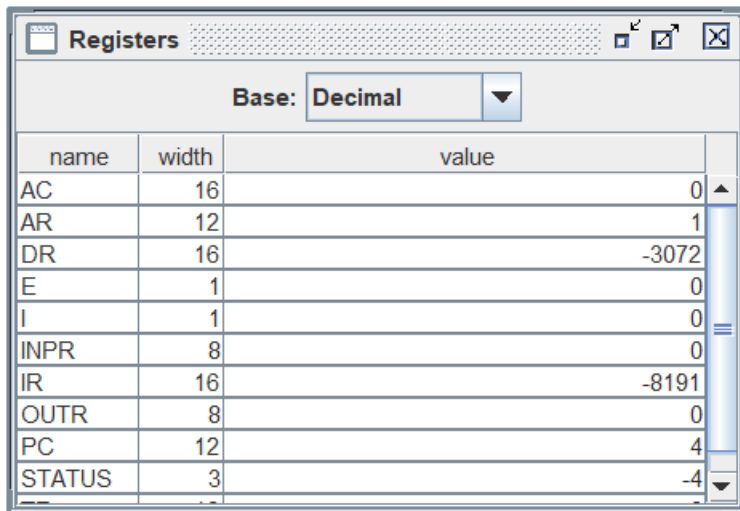
VALUE OF REGISTER BEFORE EXECUTION:



The image shows a 'Registers' window from a debugger. It contains a table with columns for 'name', 'width', and 'value'. The 'Base' is set to 'Decimal'. All registers listed have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

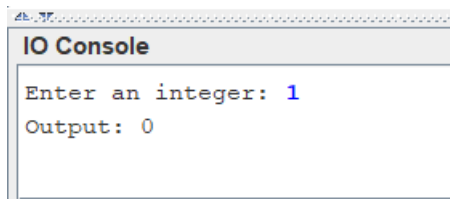
VALUE OF REGISTER AFTER EXECUTION:



The image shows a 'Registers' window with a table of register values. The base is set to 'Decimal'. The registers and their values are as follows:

name	width	value
AC	16	0
AR	12	1
DR	16	-3072
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	4
STATUS	3	-4

INPUT OUTPUT WINDOW:



The image shows an 'IO Console' window with the following text:

```
Enter an integer: 1
Output: 0
```

4. XOR

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

STA NUM (M(AR) <- AC)

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

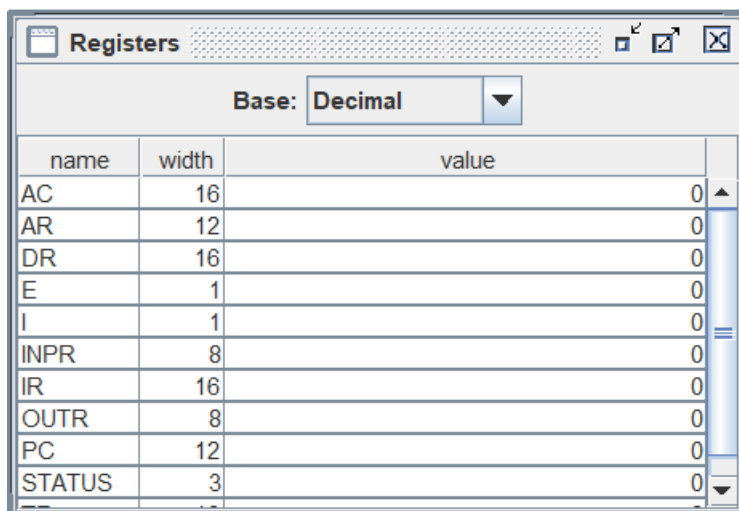
XOR NUM (DR <- M(AR) & AC <- AC xor DR)

**OUT (TAKES OUTPUT FROM AC AND DISPLAY
ON SCREEN)**

HLT (HALT-BIT = 1)

NUM: .data 1 0

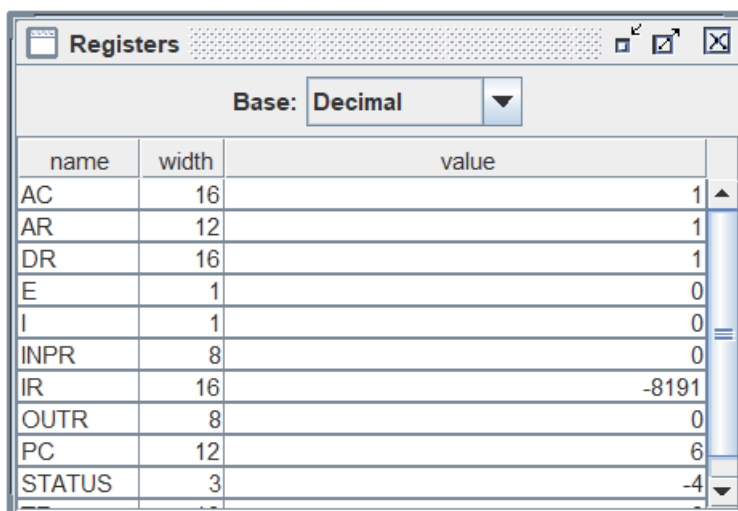
VALUE OF REGISTER BEFORE EXECUTION:



The image shows a 'Registers' window with a table of register values. The base is set to 'Decimal'. All registers have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

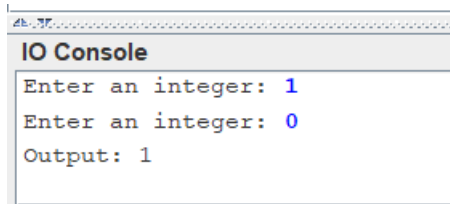
VALUE OF REGISTER AFTER EXECUTION:



The image shows the same 'Registers' window after execution. The values have changed: AC, AR, and DR are now 1; E, I, INPR, OUTR, and PC are 0; IR is -8191; and STATUS is -4.

name	width	value
AC	16	1
AR	12	1
DR	16	1
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	6
STATUS	3	-4

INPUT OUTPUT WINDOW:



```
IO Console
Enter an integer: 1
Enter an integer: 0
Output: 1
```

5. NOR

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

STA NUM (M(AR) <- AC)

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

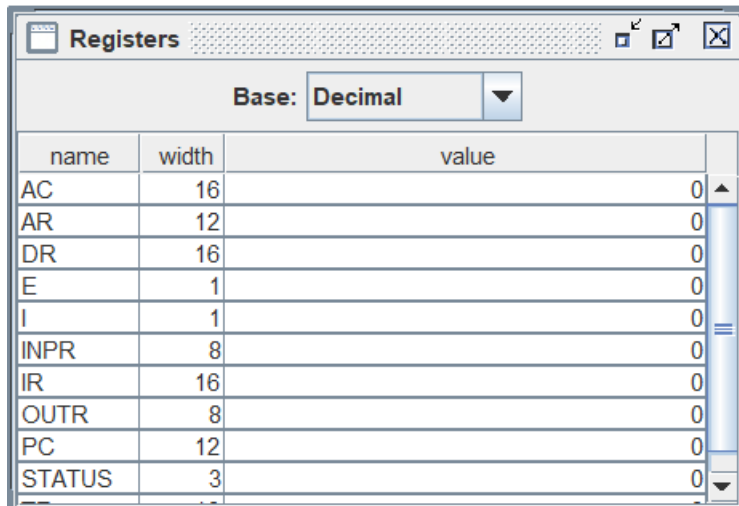
NOR NUM (DR <- M(AR) & AC <- AC nor DR)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT-BIT = 1)

NUM: .data 1 0

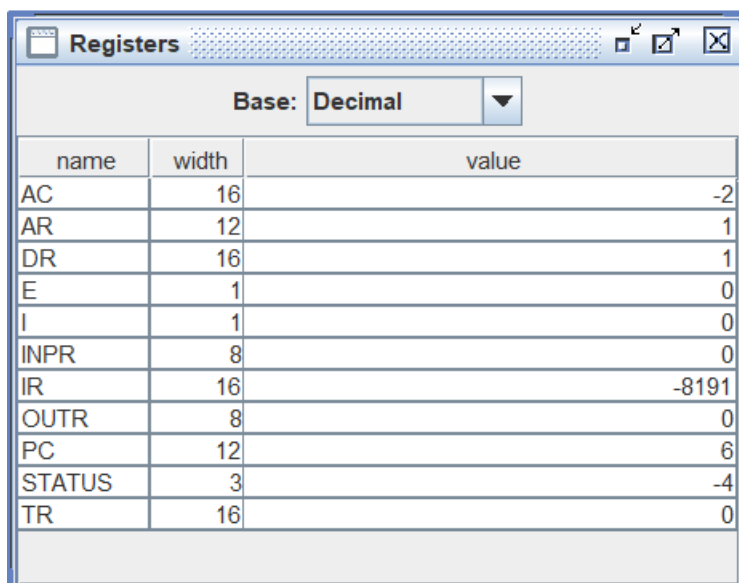
VALUE OF REGISTER BEFORE EXECUTION:



The Registers window shows the state of various registers before execution. The base is set to Decimal. All registers have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

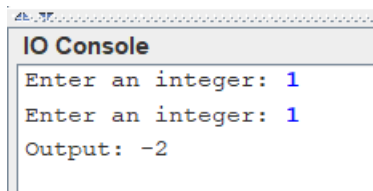
VALUE OF REGISTER AFTER EXECUTION:



The Registers window shows the state of various registers after execution. The base is set to Decimal. The values have changed: AC is -2, AR is 1, DR is 1, IR is -8191, PC is 6, and STATUS is -4. The E, I, INPR, OUTR, and TR registers remain at 0.

name	width	value
AC	16	-2
AR	12	1
DR	16	1
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	6
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



The IO Console window displays the input and output of the program. It shows two prompts for integers, both with the input '1', and the resulting output '-2'.

```
IO Console
Enter an integer: 1
Enter an integer: 1
Output: -2
```


6. NAND

**ASSEMBLY LANGUAGE PROGRAM ALONG WITH
MICROINSTRUCTIONS :**

**INP (TAKES INPUT FROM USER AND STORE IT IN
AC)**

STA NUM (M(AR) <- AC)

**INP (TAKES INPUT FROM USER AND STORE IT IN
AC)**

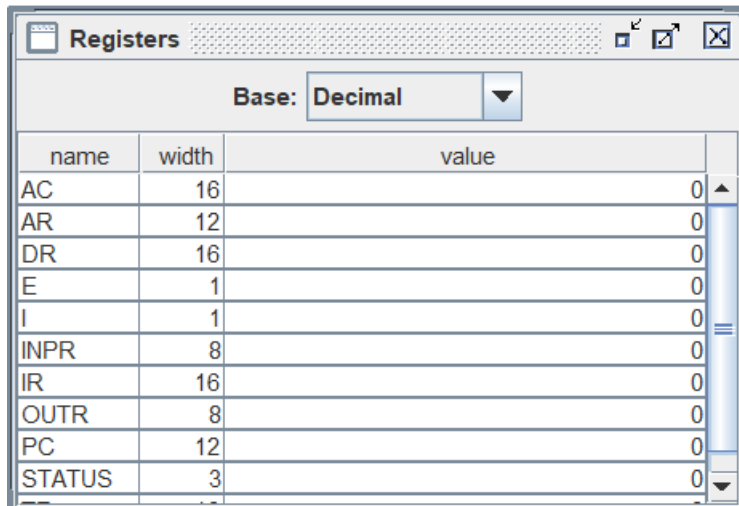
NAND NUM (DR <- M(AR) & AC <- AC nand DR)

**OUT (TAKES OUTPUT FROM AC AND DISPLAY
ON SCREEN)**

HLT (HALT-BIT = 1)

NUM: .data 1 0

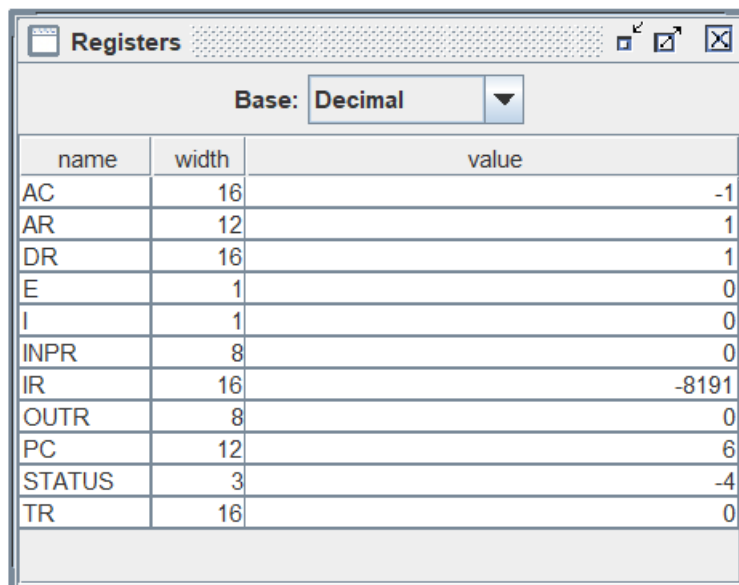
VALUE OF REGISTER BEFORE EXECUTION:



The Registers window shows the state of various registers before execution. The base is set to Decimal. All registers have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

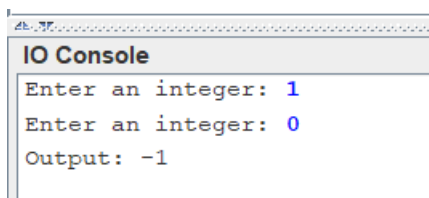
VALUE OF REGISTER AFTER EXECUTION:



The Registers window shows the state of various registers after execution. The base is set to Decimal. The values have changed: AC is -1, AR is 1, DR is 1, IR is -8191, PC is 6, and STATUS is -4. Registers E, I, INPR, OUTR, and TR remain at 0.

name	width	value
AC	16	-1
AR	12	1
DR	16	1
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	6
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



The IO Console window displays the input and output of the program. It shows two prompts for integers, with the first input being 1 and the second being 0. The output is -1.

```
IO Console
Enter an integer: 1
Enter an integer: 0
Output: -1
```

QUESTION 6:WRITE AN ASSEMBLY LANGUAGE PROGRAM FOR SIMULATING FOLLOWING MEMORY REFERENCE INSTRUCTIONS:

1.ADD

2.LDA

3.STA

4.BUN

5.ISZ

ANS:ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

1.ADD

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

STA NUM ($M(AR) \leftarrow AC$)

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

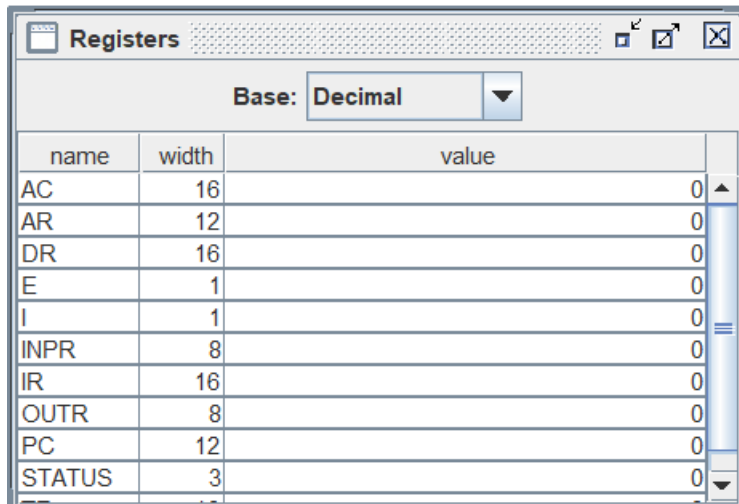
ADD NUM ($DR \leftarrow M(AR)$ & $AC \leftarrow AC + DR$)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT ($HALT-BIT = 1$)

NUM: .data 1 0

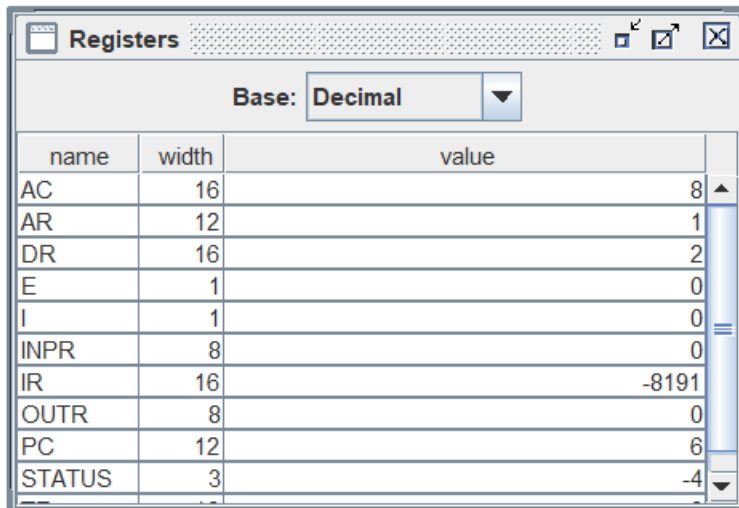
VALUE OF REGISTER BEFORE EXECUTION:



The Registers window shows the initial state of the processor registers. The base is set to Decimal. All registers (AC, AR, DR, E, I, INPR, IR, OUTR, PC, STATUS) have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

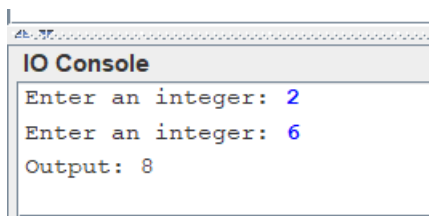
VALUE OF REGISTER AFTER EXECUTION:



The Registers window shows the state of the processor registers after execution. The base is set to Decimal. The values are: AC=8, AR=1, DR=2, E=0, I=0, INPR=0, IR=-8191, OUTR=0, PC=6, STATUS=-4.

name	width	value
AC	16	8
AR	12	1
DR	16	2
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	6
STATUS	3	-4

INPUT OUTPUT WINDOW:



The IO Console window displays the input and output of the program. It shows two prompts 'Enter an integer:' with inputs 2 and 6, and an output of 8.

```
IO Console
Enter an integer: 2
Enter an integer: 6
Output: 8
```

2. LDA : Load To AC

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

STA NUM ($M(AR) \leftarrow AC$)

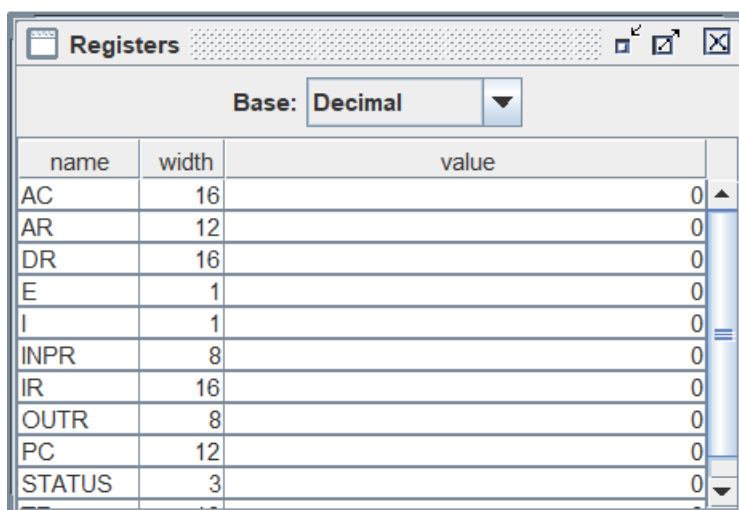
LDA NUM ($DR \leftarrow M(AR)$ AND $AC \leftarrow DR$)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT-BIT = 1)

NUM: .data 1 0

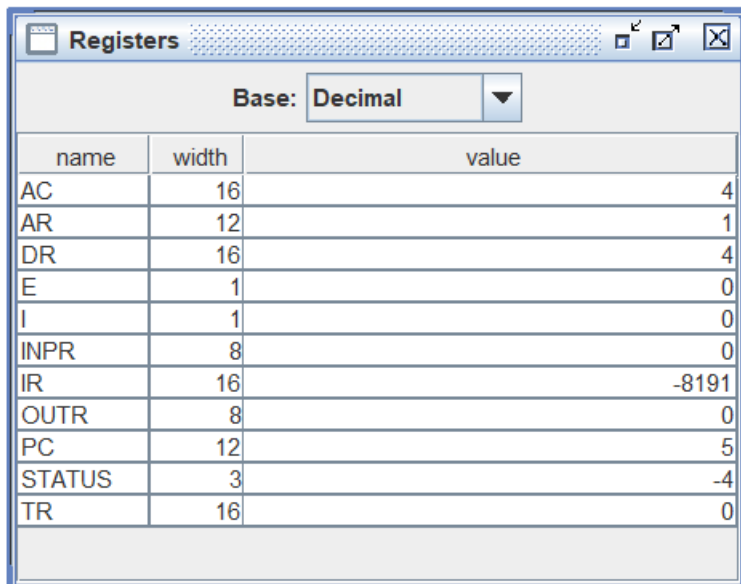
VALUE OF REGISTER BEFORE EXECUTION:



The image shows a 'Registers' window from a debugger. It contains a table with columns for 'name', 'width', and 'value'. The 'Base' is set to 'Decimal'. All registers listed have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

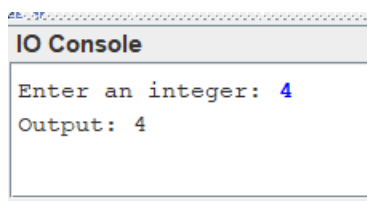
VALUE OF REGISTER AFTER EXECUTION:



The image shows a 'Registers' window with a 'Base' dropdown set to 'Decimal'. It contains a table with columns 'name', 'width', and 'value'.

name	width	value
AC	16	4
AR	12	1
DR	16	4
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	5
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



The image shows an 'IO Console' window with the following text:

```
Enter an integer: 4
Output: 4
```

3. STA : Store AC

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

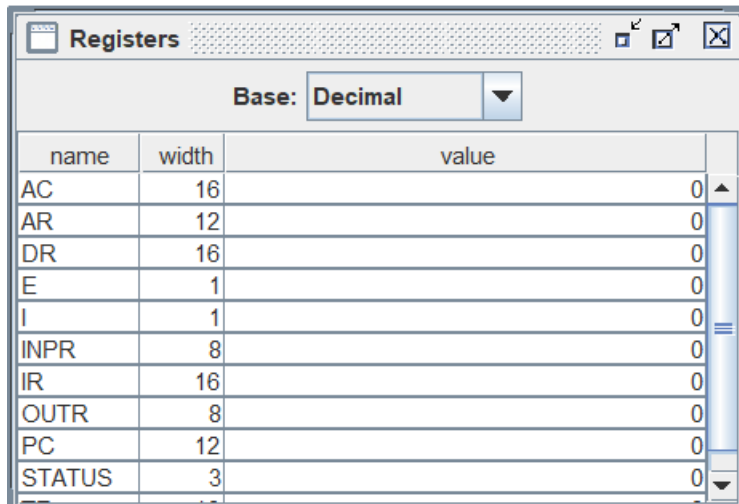
STA NUM (M(AR) <- AC)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT-BIT = 1)

NUM: .data 1 0

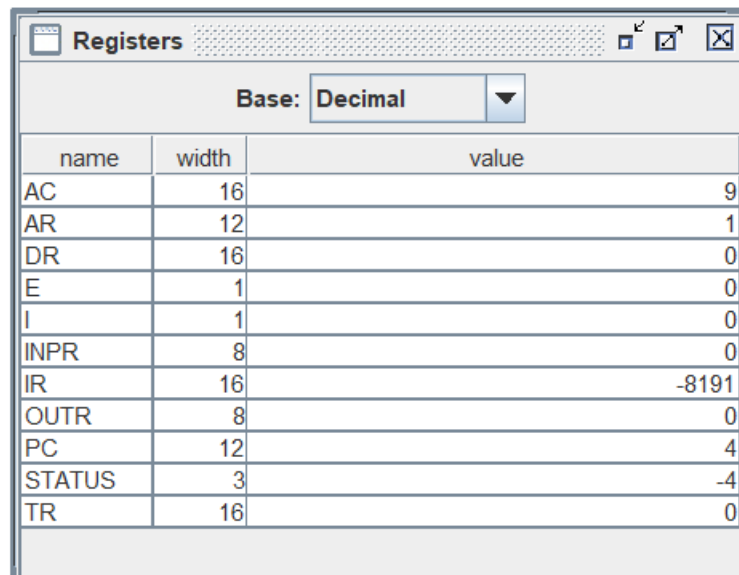
VALUE OF REGISTER BEFORE EXECUTION:



The Registers window shows the initial state of the processor registers. The base is set to Decimal. All registers have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

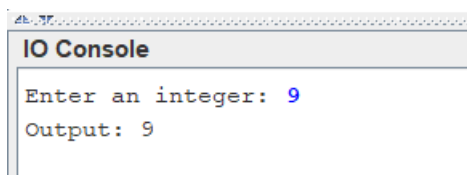
VALUE OF REGISTER AFTER EXECUTION:



The Registers window shows the state of the processor registers after execution. The base is set to Decimal. The values have changed: AC is 9, AR is 1, IR is -8191, PC is 4, and STATUS is -4. The TR register is also present with a value of 0.

name	width	value
AC	16	9
AR	12	1
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	4
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



The IO Console window displays the input and output of the program. It shows the prompt "Enter an integer:" followed by the user input "9", and the output "Output: 9".

```
IO Console
Enter an integer: 9
Output: 9
```

4. BUN : Branch Unconditionally

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

INP (TAKES INPUT FROM USER AND STORE IT IN AC)

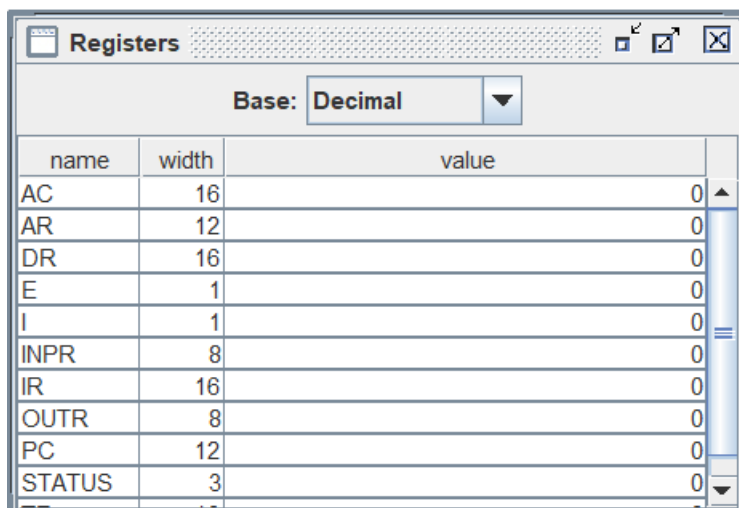
BUN K (PC \leftarrow AR AND K ACTS AS A FLAG)

INP

K: OUT

HLT (HALT-BIT = 1)

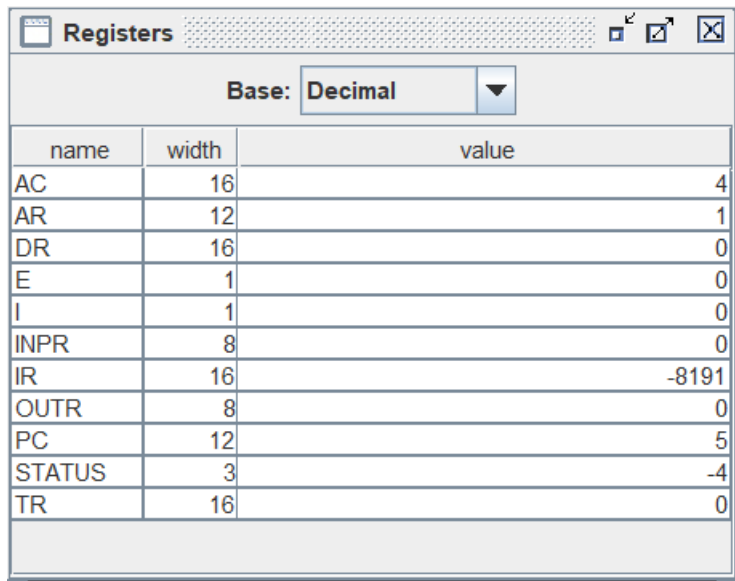
VALUE OF REGISTER BEFORE EXECUTION:



The image shows a 'Registers' window from a software application. It features a 'Base' dropdown menu set to 'Decimal'. Below this is a table with columns for 'name', 'width', and 'value'. The registers listed are AC (16 bits), AR (12 bits), DR (16 bits), E (1 bit), I (1 bit), INPR (8 bits), IR (16 bits), OUTR (8 bits), PC (12 bits), and STATUS (3 bits). All values are currently 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

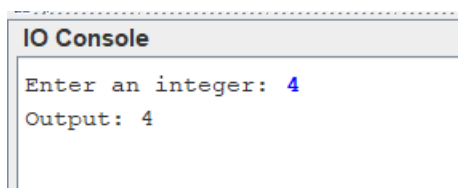
VALUE OF REGISTER AFTER EXECUTION:



The image shows a 'Registers' window with a 'Base' dropdown set to 'Decimal'. It contains a table with columns for 'name', 'width', and 'value'. The registers and their values are: AC (4), AR (1), DR (0), E (0), I (0), INPR (0), IR (-8191), OUTR (0), PC (5), STATUS (-4), and TR (0).

name	width	value
AC	16	4
AR	12	1
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	5
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



The image shows an 'IO Console' window. It displays the prompt 'Enter an integer:' followed by the user input '4'. Below this, it shows 'Output: 4'.

```
IO Console
Enter an integer: 4
Output: 4
```

NOTE: THE SECOND INPUT COMMAND GOT SKIPPED DUE TO BUN STATEMENT.

5. ISZ : Increment and Skip if Zero

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

ISZ 009

OUT

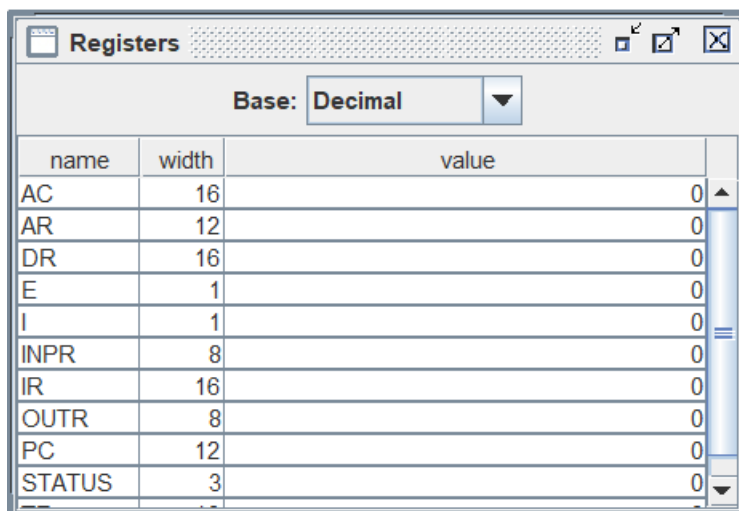
HLT (HALT-BIT = 1)

IMPLEMENTATION:

FIRST THREE MICROINSTRUCTIONS ARE FOR INCREMENT THE VALUE IN MAIN MEMORY

LAST THREE MICROINSTRUCTIONS ARE FOR CHECKING WHETHER IT IS ZERO OR NOT AND SKIPPING IF IT IS.

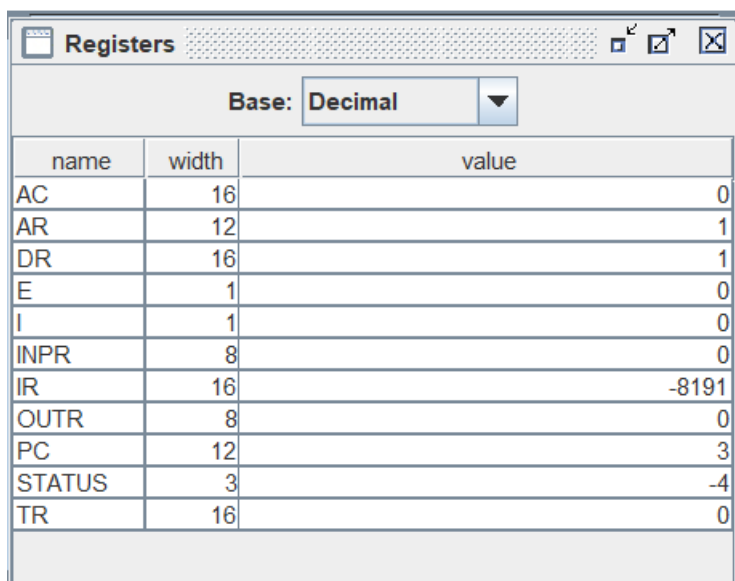
VALUE OF REGISTER BEFORE EXECUTION:



A screenshot of a 'Registers' window with a 'Base' dropdown set to 'Decimal'. It contains a table with 11 registers, all of which have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

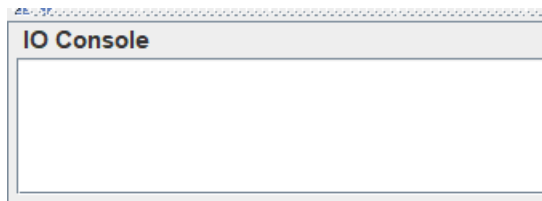
VALUE OF REGISTER AFTER EXECUTION:



A screenshot of the same 'Registers' window after execution. The values for several registers have changed: AR is 1, DR is 1, IR is -8191, PC is 3, and STATUS is -4. The other registers remain at 0.

name	width	value
AC	16	0
AR	12	1
DR	16	1
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	3
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



NOTE: OUTPUT COMMAND GOT SKIPPED AS AFTER INCREMENT THE VALUE AT ADDRESS 009 BECAME "0".

QUESTION 7:WRITE AN ASSEMBLY LANGUAGE PROGRAM TO SIMULATE THE MACHINE FOR FOLLOWING REGISTER REFERENCE INSTRUCTIONS AND DETERMINE THE CONTENTS OF AC,E,PC,AR AND IR REGISTERS IN DECIMAL AFTER THE EXECUTION.

1.CLA

2.CMA

3.CME

4.HLT

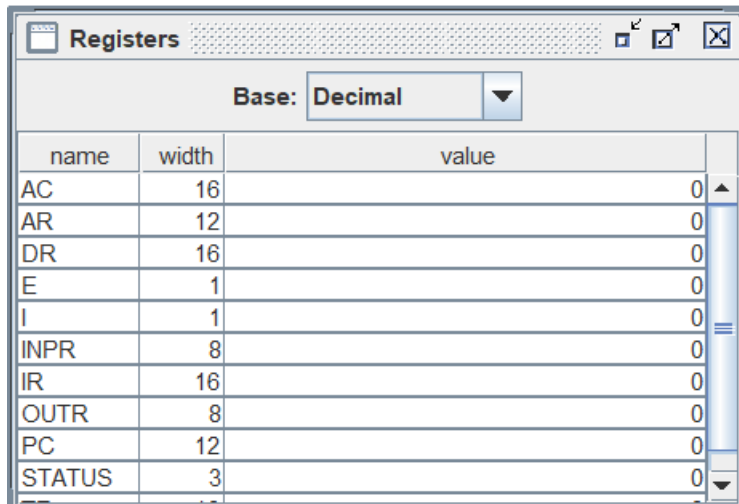
ANS:1. CLA : Clear Accumulator

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

CLA (AC <- 0)

HLT (HALT-BIT = 1)

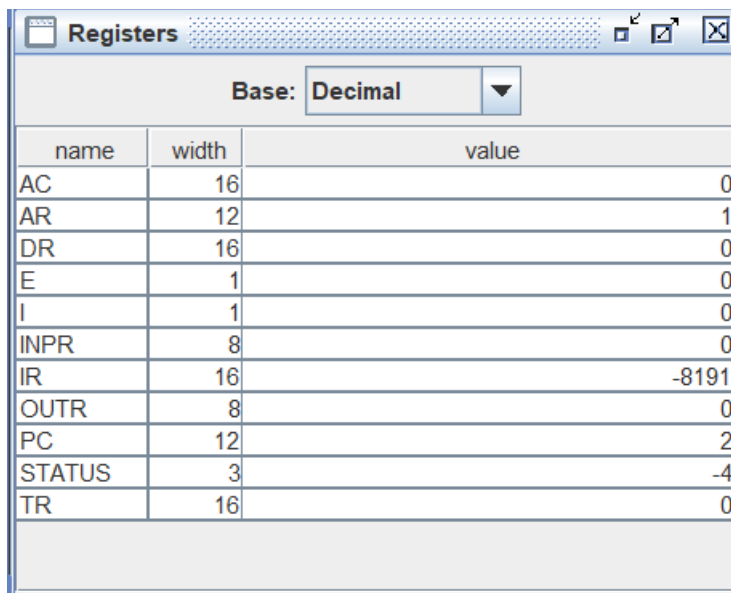
VALUE OF REGISTER BEFORE EXECUTION:



The 'Registers' window displays the state of various registers before execution. The base is set to 'Decimal'. All registers have a value of 0.

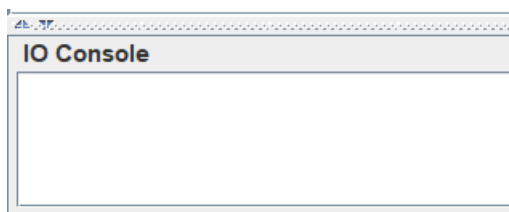
name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

VALUE OF REGISTER AFTER EXECUTION:



The 'Registers' window displays the state of various registers after execution. The base is set to 'Decimal'. The values have changed for several registers.

name	width	value
AC	16	0
AR	12	1
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	2
STATUS	3	-4
TR	16	0



The 'IO Console' window is currently empty.

IO Console

NOTE : IO CONSOLE WILL STAY EMPTY.

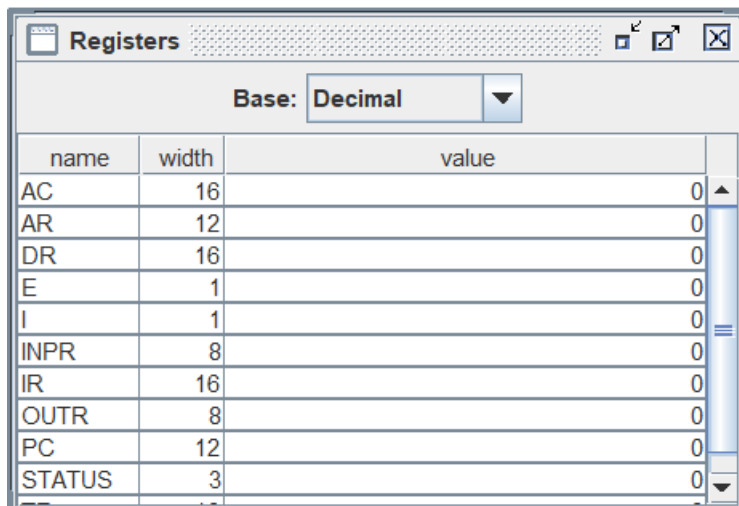
2. CMA : Complement Accumulator

**ASSEMBLY LANGUAGE PROGRAM ALONG WITH
MICROINSTRUCTIONS :**

CMA (AC <- AC')

HLT (HALT-BIT = 1)

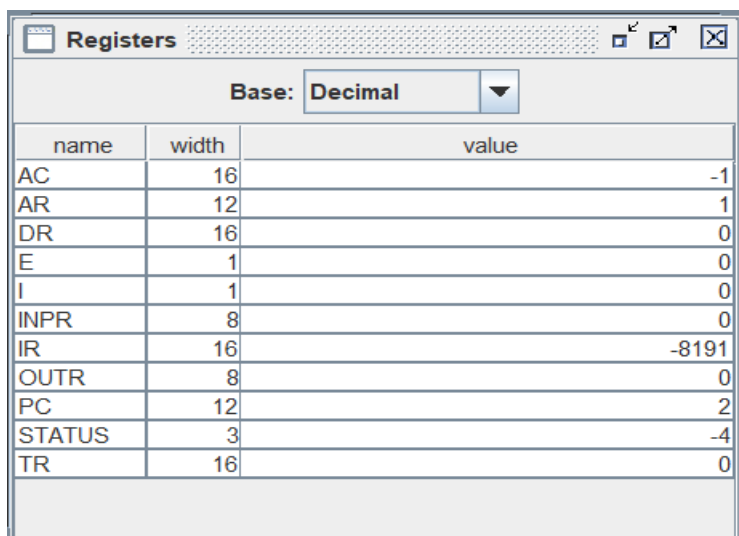
VALUE OF REGISTER BEFORE EXECUTION:



The image shows a 'Registers' window with a 'Base' dropdown set to 'Decimal'. It contains a table with 10 rows of registers and their values.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

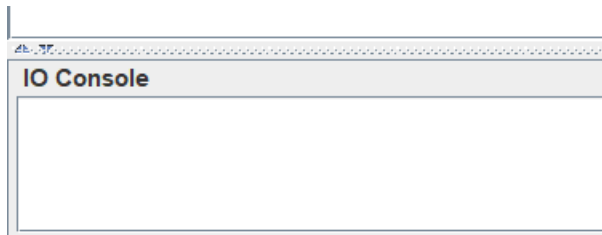
VALUE OF REGISTER AFTER EXECUTION:



The image shows the same 'Registers' window after execution. The values have changed: AC is -1, AR is 1, IR is -8191, and STATUS is -4. A new register 'TR' with width 16 and value 0 has been added at the bottom.

name	width	value
AC	16	-1
AR	12	1
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	2
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



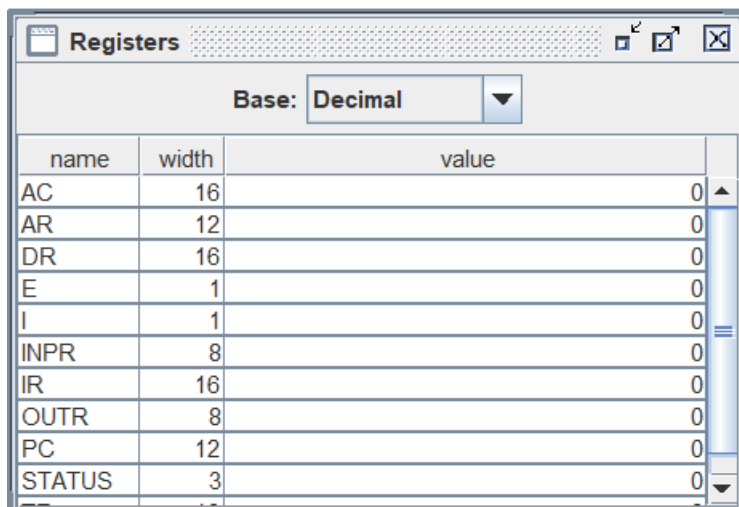
3. CME : Complement Extended Bit

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

CME (E <- E')

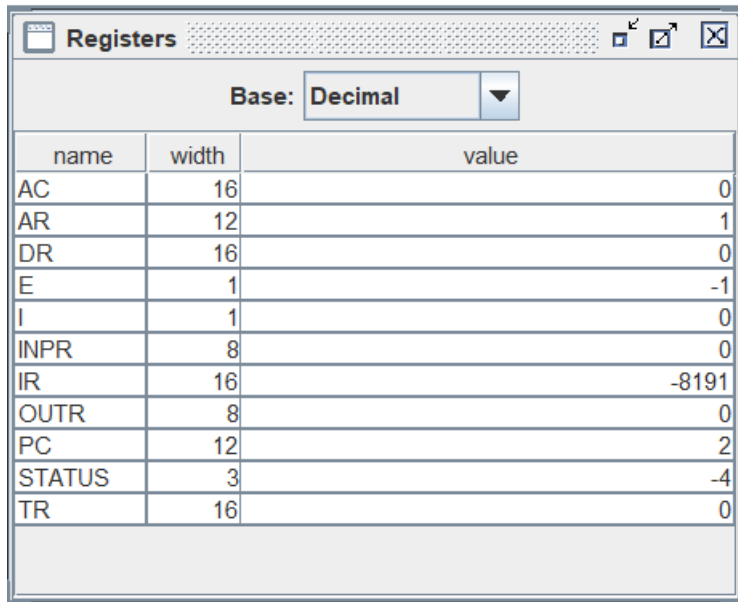
HLT (HALT-BIT = 1)

VALUE OF REGISTER BEFORE EXECUTION:

A screenshot of a software window titled "Registers". The window has a title bar with standard Windows icons and a menu bar. Below the menu bar is a "Base:" dropdown menu set to "Decimal". Below this is a table with three columns: "name", "width", and "value". The table lists several registers and their current values.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

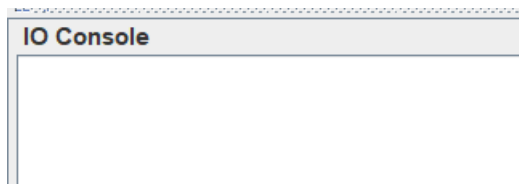
VALUE OF REGISTER AFTER EXECUTION:



The image shows a 'Registers' window from a debugger. It has a title bar with a 'Registers' label and standard window controls. Below the title bar is a 'Base:' dropdown menu set to 'Decimal'. The main area is a table with three columns: 'name', 'width', and 'value'. The table lists several registers with their current values.

name	width	value
AC	16	0
AR	12	1
DR	16	0
E	1	-1
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	2
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:

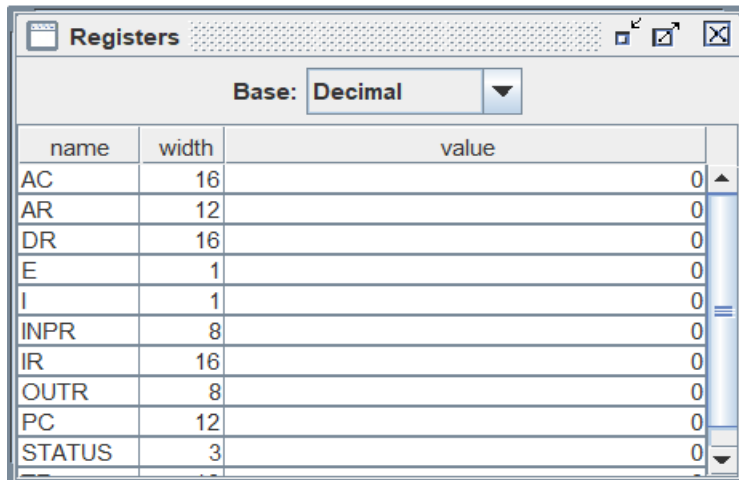


4. HLT : HALT

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS

HLT (HALT-BIT = 1 AND END)

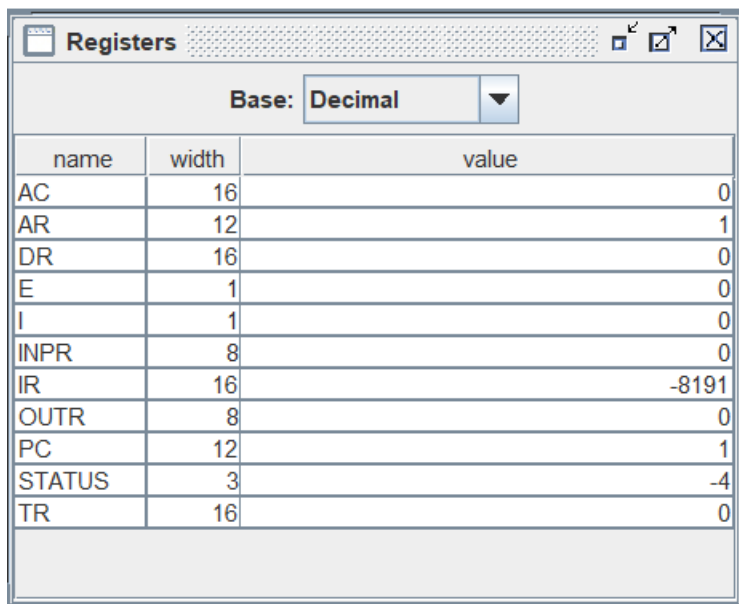
VALUE OF REGISTER BEFORE EXECUTION:



The image shows a 'Registers' window with a 'Base' dropdown set to 'Decimal'. It contains a table with 10 registers: AC, AR, DR, E, I, INPR, IR, OUTR, PC, and STATUS. All registers have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

VALUE OF REGISTER AFTER EXECUTION:



The image shows the same 'Registers' window after execution. The values have changed: AR is 1, IR is -8191, PC is 1, and STATUS is -4. A new register, TR, has appeared at the bottom with a value of 0.

name	width	value
AC	16	0
AR	12	1
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	1
STATUS	3	-4
TR	16	0

NOTE : IO CONSOLE WILL STAY EMPTY.

QUESTION 8: WRITE AN ASSEMBLY LANGUAGE PROGRAM TO SIMULATE THE MACHINE FOR FOLLOWING REGISTER REFERENCE INSTRUCTIONS AND DETERMINE THE CONTENTS

**OF AC,E,PC,AR AND IR REGISTERS IN DECIMAL
AFTER THE EXECUTION.**

1.INC

2.SPA

3.SNA

4.SZE

ANS: 1. INC : Increment AC

**ASSEMBLY LANGUAGE PROGRAM ALONG WITH
MICROINSTRUCTIONS**

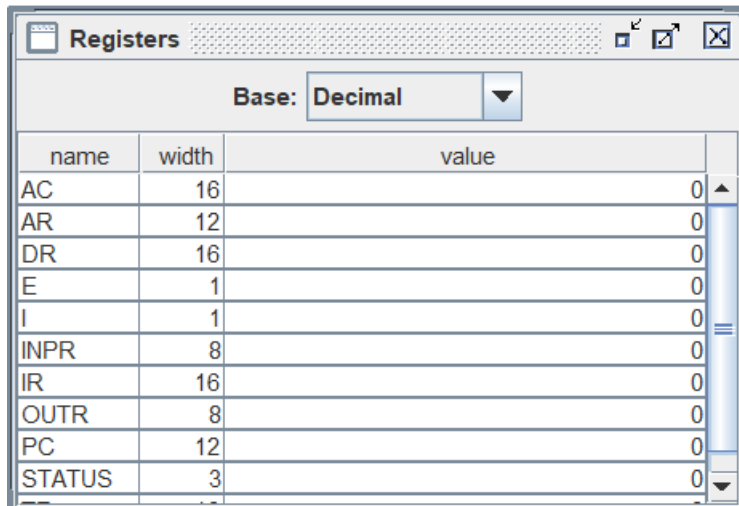
**INP (TAKES INPUT FROM USER AND STORE IT IN
AC)**

INC ($AC \leftarrow AC + 1$)

**OUT (TAKES OUTPUT FROM AC AND DISPLAY
ON SCREEN)**

HLT ($\text{HALT-BIT} = 1$)

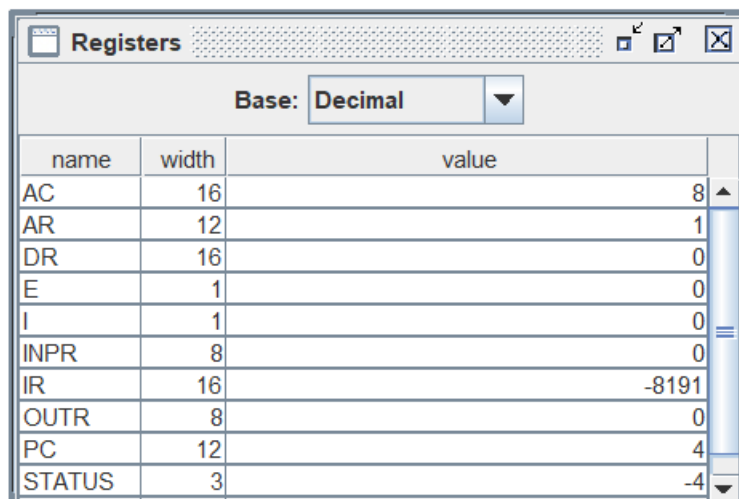
VALUE OF REGISTER BEFORE EXECUTION:



The Registers window shows the state of various registers before execution. The base is set to Decimal. All registers have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

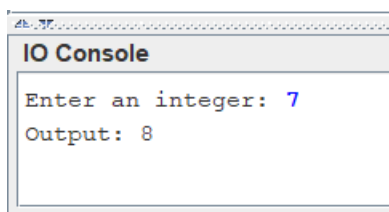
VALUE OF REGISTER AFTER EXECUTION:



The Registers window shows the state of various registers after execution. The base is set to Decimal. The values have changed: AC is 8, AR is 1, IR is -8191, PC is 4, and STATUS is -4. DR, E, I, INPR, and OUTR remain at 0.

name	width	value
AC	16	8
AR	12	1
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	4
STATUS	3	-4

INPUT OUTPUT WINDOW:



The IO Console window shows the input and output of the program. The input is "Enter an integer: 7" and the output is "Output: 8".

```
IO Console
Enter an integer: 7
Output: 8
```

2. SPA : Skip if Positive

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS

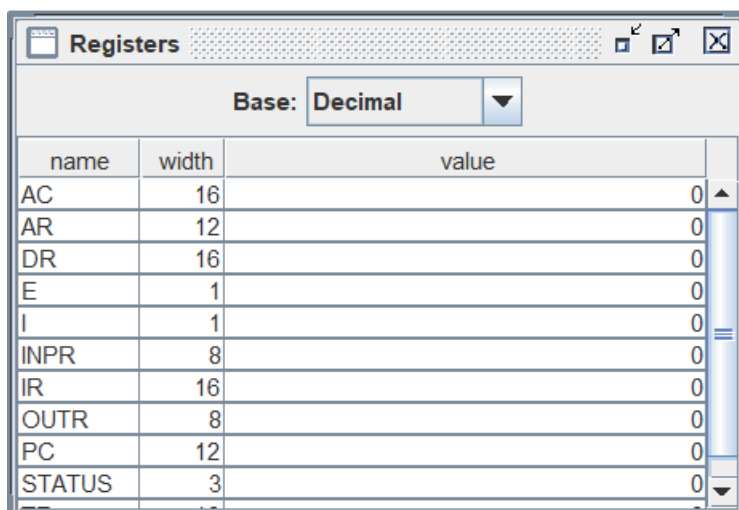
INP (TAKES INPUT FROM USER AND STORE IT IN AC)

SPA

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT-BIT = 1)

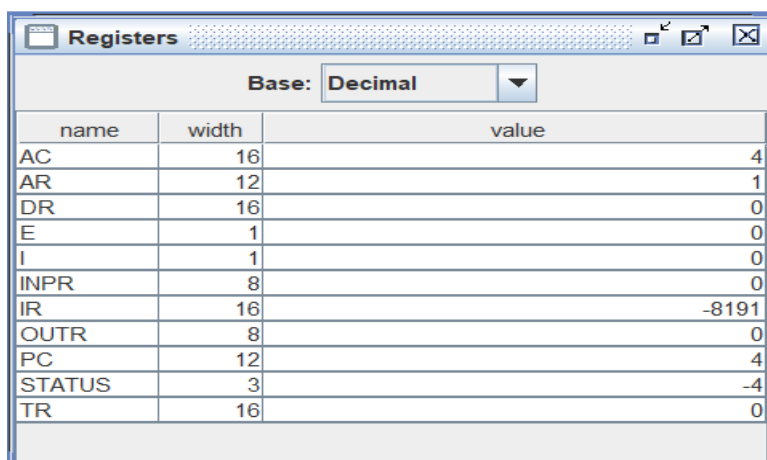
VALUE OF REGISTER BEFORE EXECUTION:



The image shows a 'Registers' window with a 'Base' dropdown set to 'Decimal'. It contains a table with 11 rows of registers and their values.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

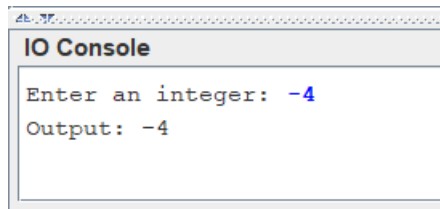
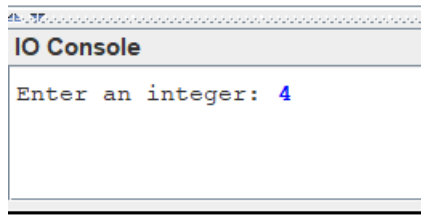
VALUE OF REGISTER AFTER EXECUTION:



The image shows the same 'Registers' window after execution. The values have changed for several registers.

name	width	value
AC	16	4
AR	12	1
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	4
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



3. SNA : Skip if Negative

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS

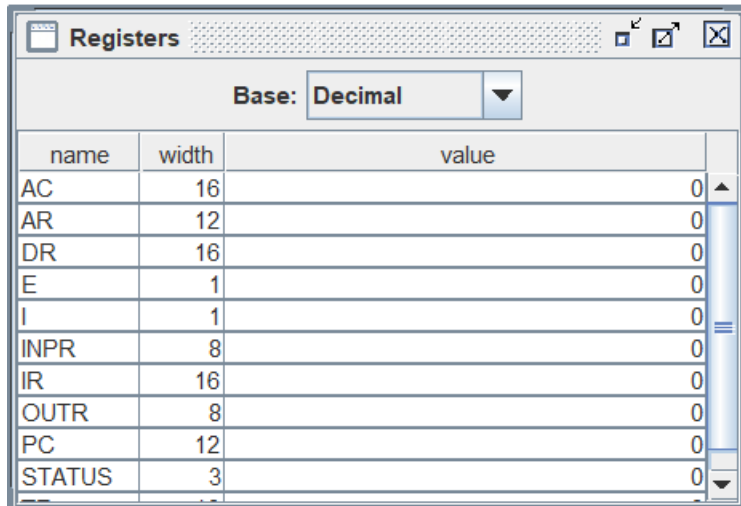
INP (TAKES INPUT FROM USER AND STORE IT IN AC)

SNA

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT-BIT = 1)

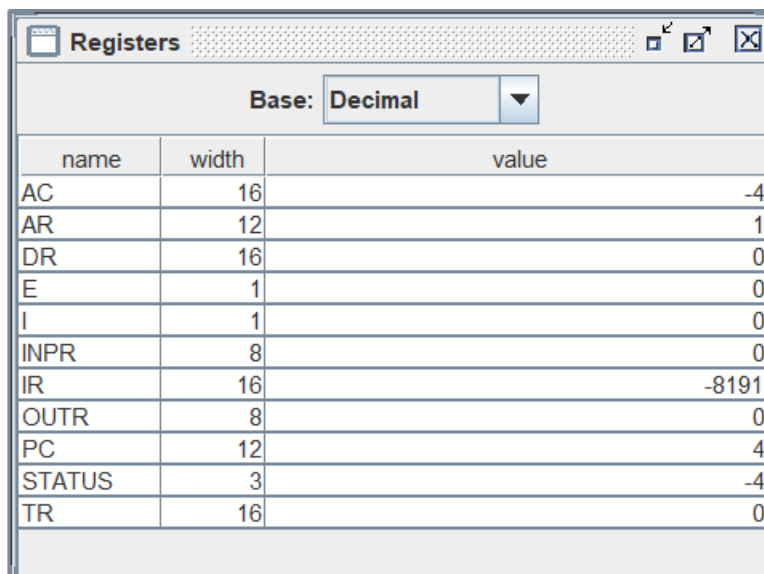
VALUE OF REGISTER BEFORE EXECUTION:



The Registers window shows the initial state of the processor registers. The base is set to Decimal. All registers have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

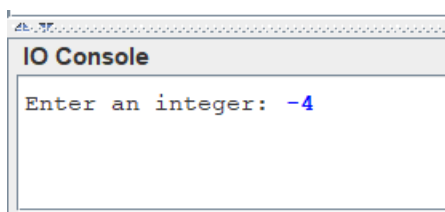
VALUE OF REGISTER AFTER EXECUTION:



The Registers window shows the state of the processor registers after execution. The base is set to Decimal. The values have changed: AC is -4, AR is 1, DR is 0, E is 0, I is 0, INPR is 0, IR is -8191, OUTR is 0, PC is 4, STATUS is -4, and TR is 0.

name	width	value
AC	16	-4
AR	12	1
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	4
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



The IO Console window displays the prompt "Enter an integer:" followed by the user input "-4".

```
Enter an integer: -4
```

```
IO Console
Enter an integer: 4
Output: 4
```

4. SZE : Skip if Extended bit is 0

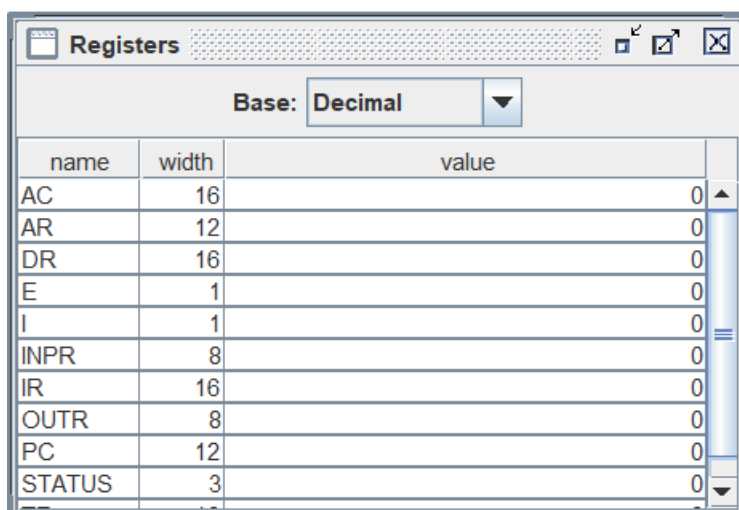
ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS

SZE

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT-BIT = 1)

VALUE OF REGISTER BEFORE EXECUTION:



The image shows a 'Registers' window from a simulator. It has a title bar with standard window controls. Below the title bar, there is a 'Base:' dropdown menu set to 'Decimal'. The main area is a table with three columns: 'name', 'width', and 'value'. The table lists several registers and their current values, all of which are 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

VALUE OF REGISTER AFTER EXECUTION:

Registers		
Base: Decimal		
name	width	value
AC	16	0
AR	12	1
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	4
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:

IO Console
Enter an integer: 0

QUESTION 10: WRITE AN ASSEMBLY LANGUAGE PROGRAM TO SIMULATE THE MACHINE FOR FOLLOWING REGISTER REFERENCE INSTRUCTIONS AND DETERMINE THE CONTENTS OF AC,E,PC,AR AND IR REGISTERS IN DECIMAL AFTER THE EXECUTION.

1.CIR

2.CIL

ANS: 1. CIR : Circulate Right

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS

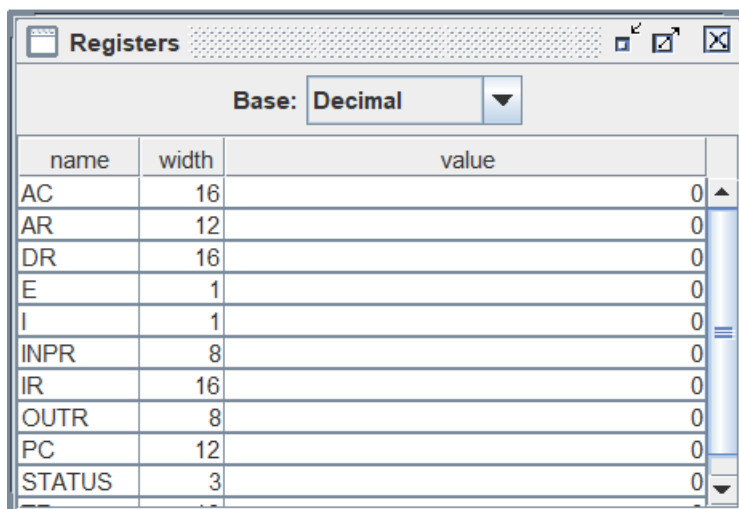
INP (TAKES INPUT FROM USER AND STORE IT IN AC)

CIR

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT-BIT = 1)

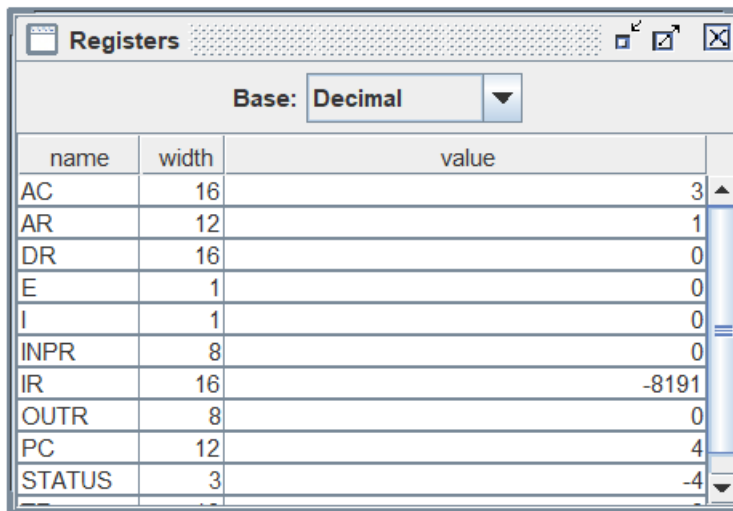
VALUE OF REGISTER BEFORE EXECUTION:



The image shows a 'Registers' window from a debugger or simulator. It displays a list of registers with their names, widths, and current values. The base is set to 'Decimal'. All registers show a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

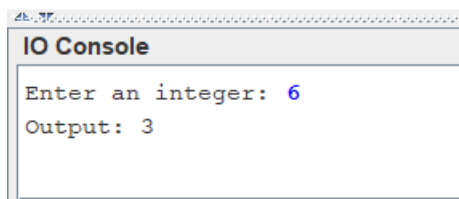
VALUE OF REGISTER AFTER EXECUTION:



The image shows a 'Registers' window from a debugger. It has a title bar with standard window controls. Below the title bar, there is a 'Base:' dropdown menu set to 'Decimal'. The main area is a table with three columns: 'name', 'width', and 'value'. The table lists several registers with their current values.

name	width	value
AC	16	3
AR	12	1
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	4
STATUS	3	-4

INPUT OUTPUT WINDOW:



2. CIL : Circulate Left

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS

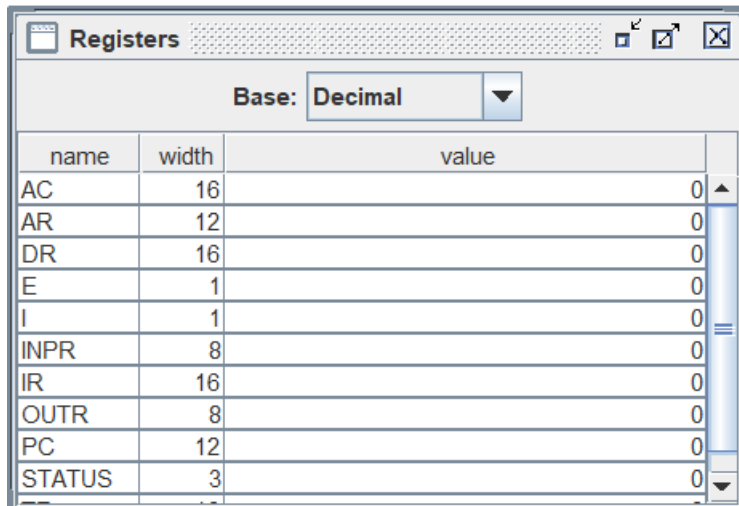
INP (TAKES INPUT FROM USER AND STORE IT IN AC)

CIL

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT-BIT = 1)

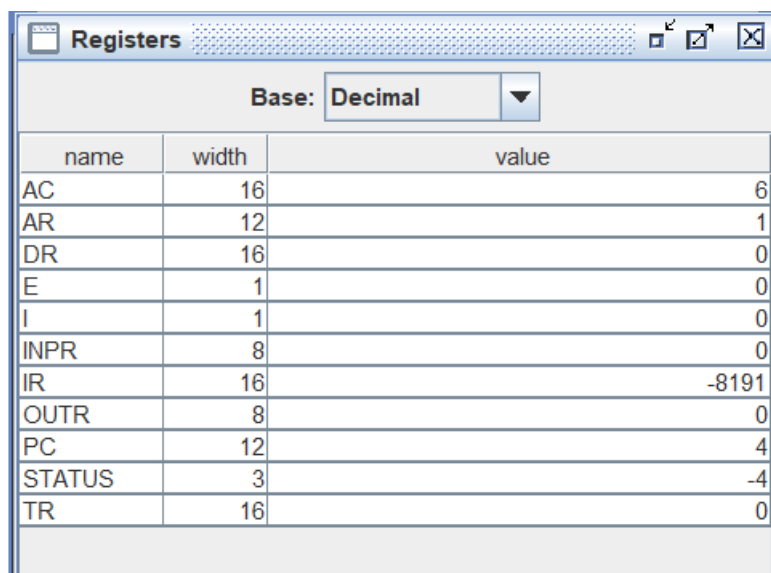
VALUE OF REGISTER BEFORE EXECUTION:



The Registers window shows the initial state of the processor registers. The base is set to Decimal. All registers have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

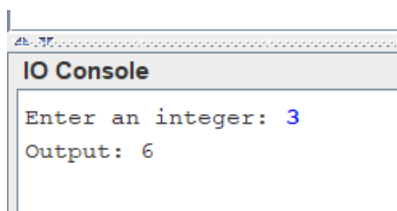
VALUE OF REGISTER AFTER EXECUTION:



The Registers window shows the state of the processor registers after execution. The base is set to Decimal. The values have changed: AC is 6, AR is 1, IR is -8191, PC is 4, and STATUS is -4. A new register TR is also visible with a value of 0.

name	width	value
AC	16	6
AR	12	1
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	4
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



The IO Console window shows the input and output of the program. The input is "Enter an integer: 3" and the output is "Output: 6".

```
Enter an integer: 3
Output: 6
```

QUESTION 10:WRITE AN ASSEMBLY PROGRAM THAT READS IN INTEGER AND ADDS THEM

TOGETHER UNTIL A NEGATIVE-ZERO NUMBER IS READ IN.THEN IT OUTPUTS THE SUM(NOT INCLUDING THE LAST NUMBER.)

ANS: ASSEMBLY LANGUAGE PROGRAM:



```
practical 10.a
; This program will take input of integers and add them
; until a negative number is encountered.

START: READ      ; read input

        JMPN DONE ; if n < 0 then jump to DONE

        ADD SUM   ; add to sum

        STA SUM   ; store sum

        JUMP START; jump to start to read again

DONE:  LDA SUM    ;load final sum

        WRITE     ; display contents of sum

        STOP     ; hlt

SUM:   .data 2 0   ; 2-byte sum initialized to 0
```

MICROINSTRUCTIONS:

READ :

INPUT

END

JMPN :

IF (AC > 0) SKIP 1

PC <- AR

END

ADD :

DR <- M(AR)

AC <- AC + DR

END

STORE :

M(AR) <- AC

END

JUMP :

PC <- AR

END

LDA :

DR <- M(AR)

AC <- DR

WRITE :

OUTPUT

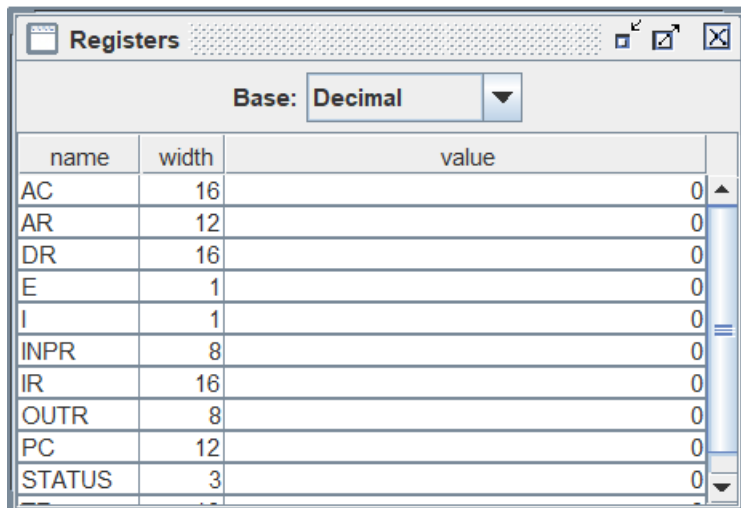
END

STOP :

HALT

END

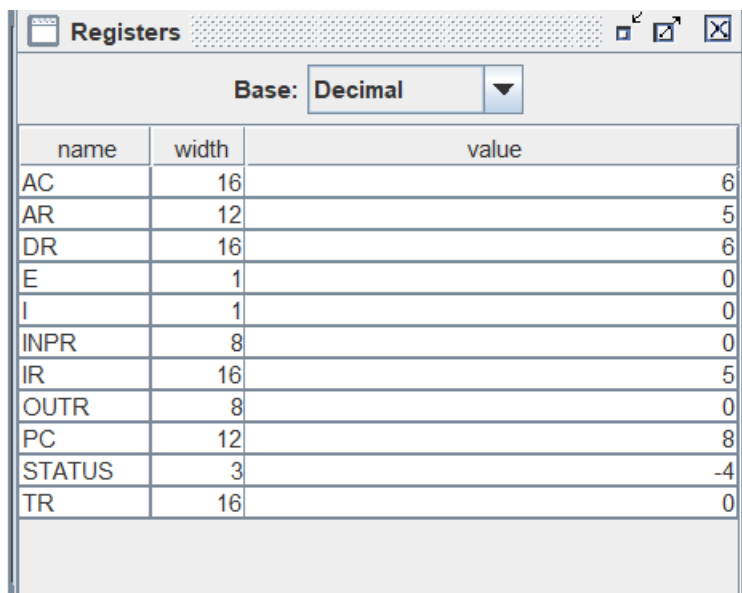
VALUE OF REGISTER BEFORE EXECUTION:



The image shows a 'Registers' window with a 'Base' dropdown set to 'Decimal'. It contains a table with 10 rows of registers. All values are 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

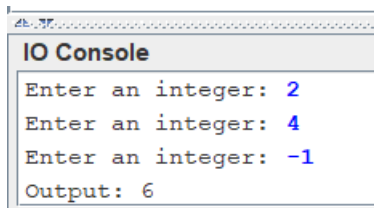
VALUE OF REGISTER AFTER EXECUTION:



The image shows the same 'Registers' window after execution. The values have changed for several registers.

name	width	value
AC	16	6
AR	12	5
DR	16	6
E	1	0
I	1	0
INPR	8	0
IR	16	5
OUTR	8	0
PC	12	8
STATUS	3	-4
TR	16	0

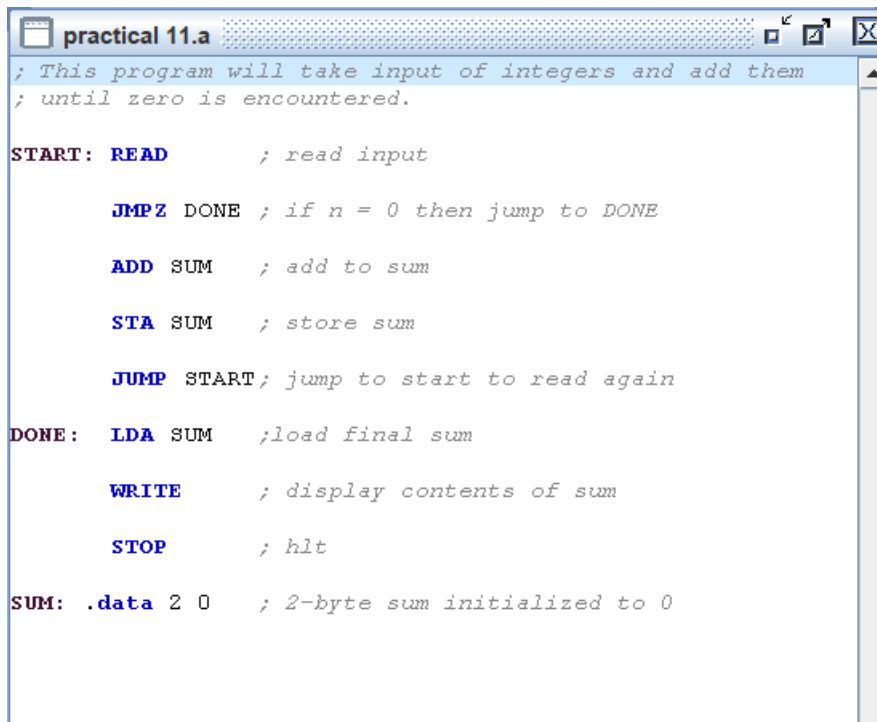
INPUT OUTPUT WINDOW:



```
IO Console
Enter an integer: 2
Enter an integer: 4
Enter an integer: -1
Output: 6
```

QUESTION 11:WRITE AN ASSEMBLY PROGRAM THAT READS IN INTEGER AND ADDS THEM TOGETHER UNTIL A ZERO IS ENCOUNTERED.THEN IT OUTPUTS THE SUM(NOT INCLUDING THE LAST NUMBER.)

ANS: ASSEMBLY LANGUAGE PROGRAM



```
practical 11.a
; This program will take input of integers and add them
; until zero is encountered.

START: READ      ; read input

        JMPZ DONE ; if n = 0 then jump to DONE

        ADD SUM   ; add to sum

        STA SUM   ; store sum

        JUMP START; jump to start to read again

DONE: LDA SUM     ;load final sum

        WRITE     ; display contents of sum

        STOP      ; hlt

SUM: .data 2 0    ; 2-byte sum initialized to 0
```

MICROINSTRUCTIONS

READ :

INPUT

END

JMPZ :

IF (AC != 0) SKIP 1

PC <- AR

END

ADD :

DR <- M(AR)

AC <- AC + DR

END

STORE :

M(AR) <- AC

END

JUMP :

PC <- AR

END

LDA :

DR <- M(AR)

AC <- DR

DIVIDE :

DR <- M(AR)

AC <- AC/DR

END

WRITE :

OUTPUT

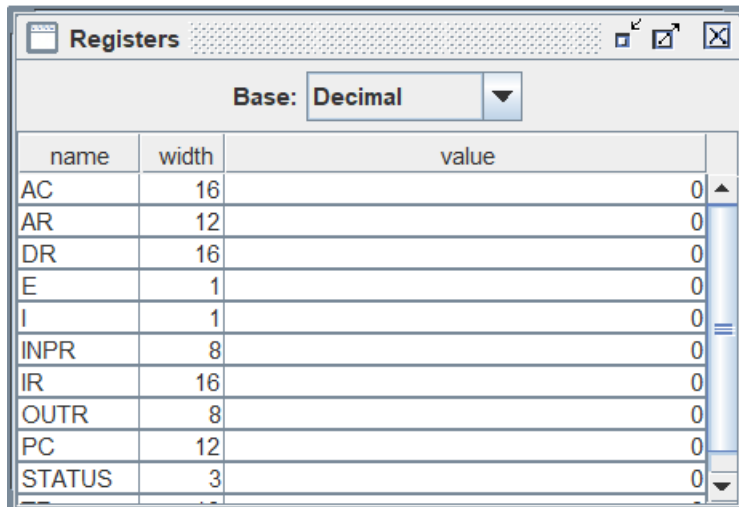
END

STOP :

HALT

END

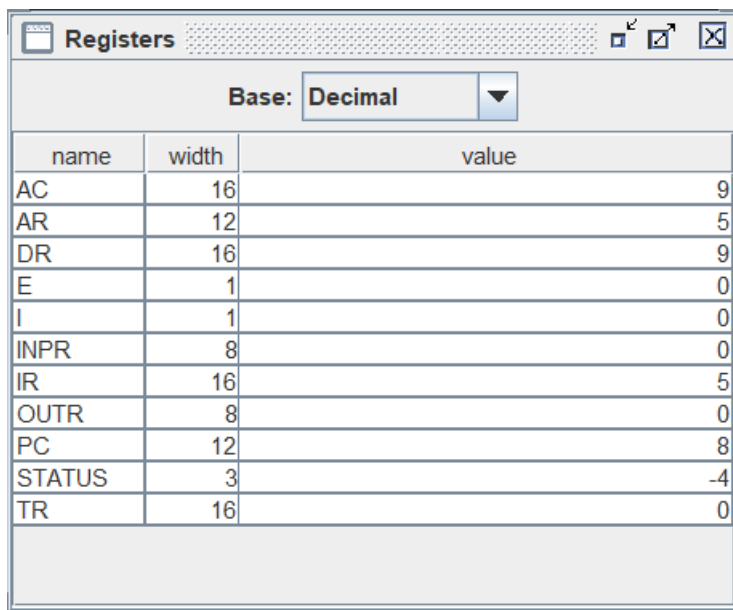
VALUE OF REGISTER BEFORE EXECUTION:



The Registers window shows the state of various registers before execution. The base is set to Decimal. All registers have a value of 0.

name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

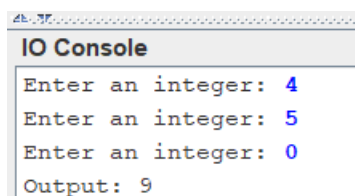
VALUE OF REGISTER AFTER EXECUTION:



The Registers window shows the state of various registers after execution. The base is set to Decimal. The values have changed for several registers.

name	width	value
AC	16	9
AR	12	5
DR	16	9
E	1	0
I	1	0
INPR	8	0
IR	16	5
OUTR	8	0
PC	12	8
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



The IO Console window shows the sequence of input and output operations. It displays three input prompts followed by an output line.

IO Console
Enter an integer: 4
Enter an integer: 5
Enter an integer: 0
Output: 9

