PRACTICAL FILE

COMPUTER SYSTEM ARCHITECTURE

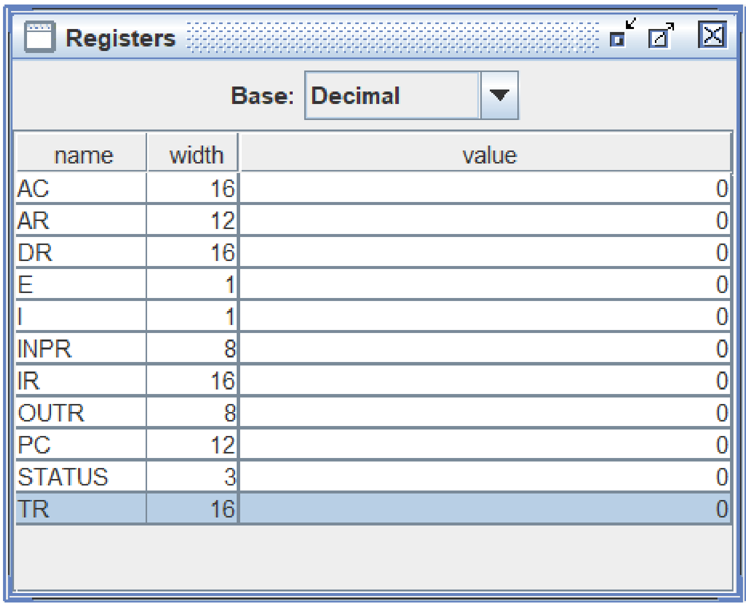
BSc(H) Computer Science

FIRST SEMESTER

SUBMITTED BY:	SUBMITTED TO:
ACHALA SINGH	PROF. JITENDRA
22/CS/01	SINGH

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

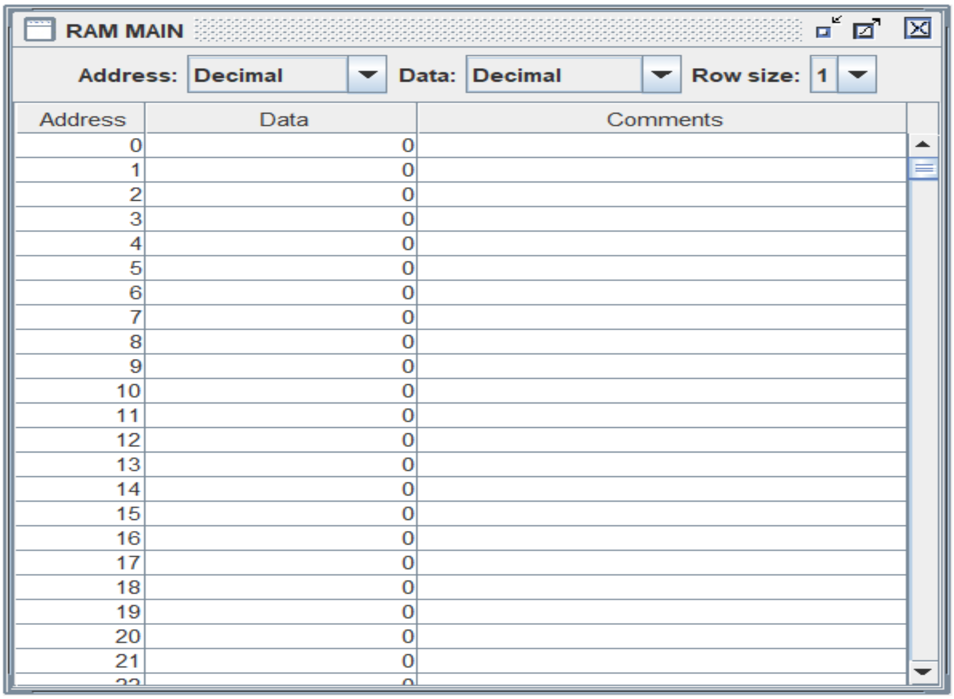
REGISTER:



Base: Decimal

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

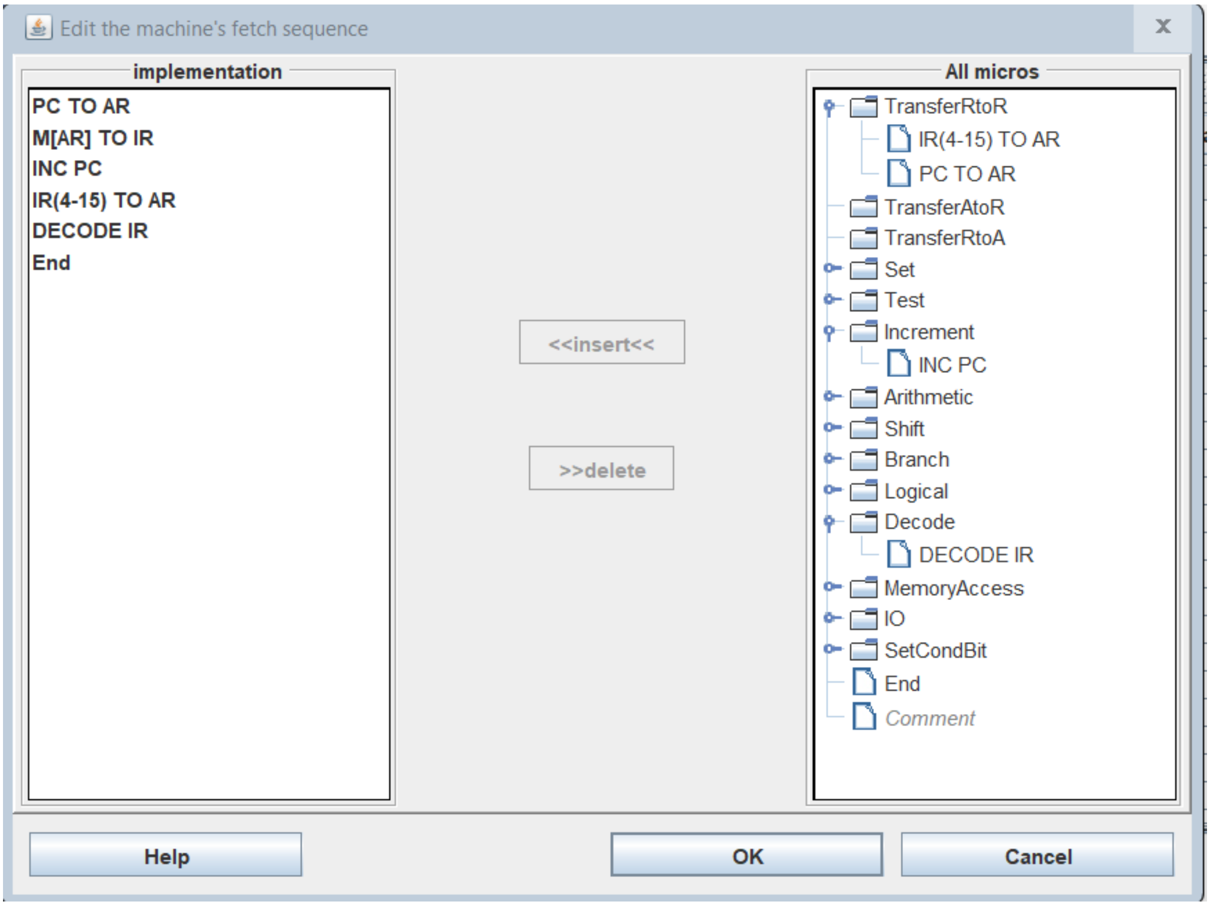
RAM:



Address: Decimal Data: Decimal Row size: 1

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	

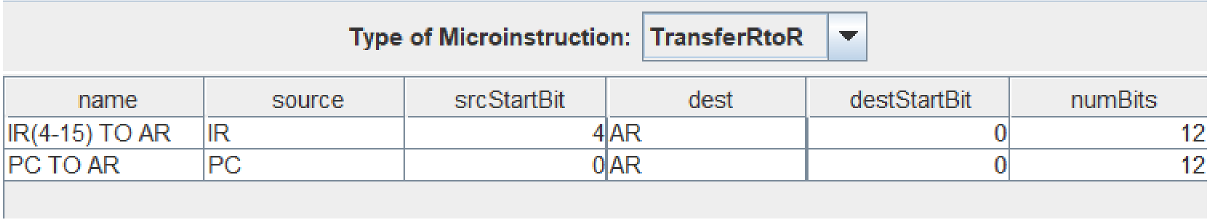
QUESTION 2:CREATE A FETCH ROUTINE OF THE INSTRUCTION CYCLE.



Edit the machine's fetch sequence

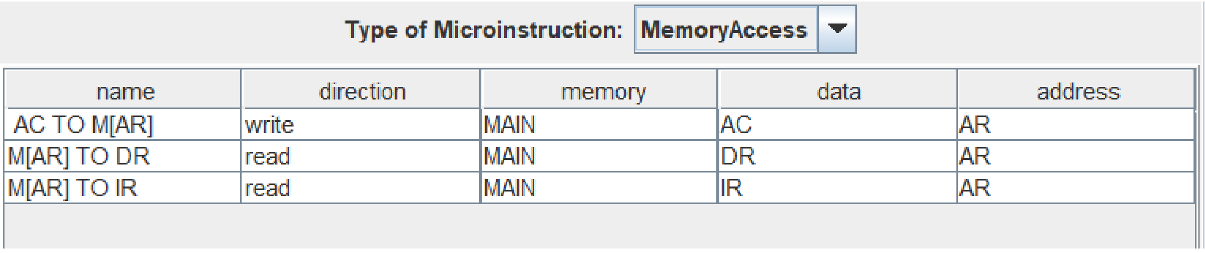
implementation

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



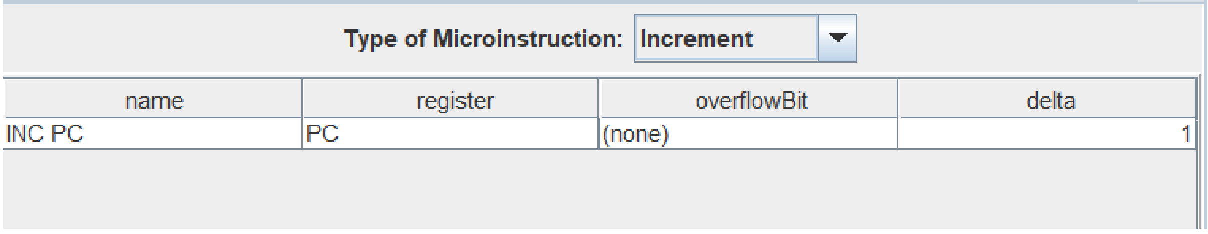
Type of Microinstruction: TransferRtoR

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



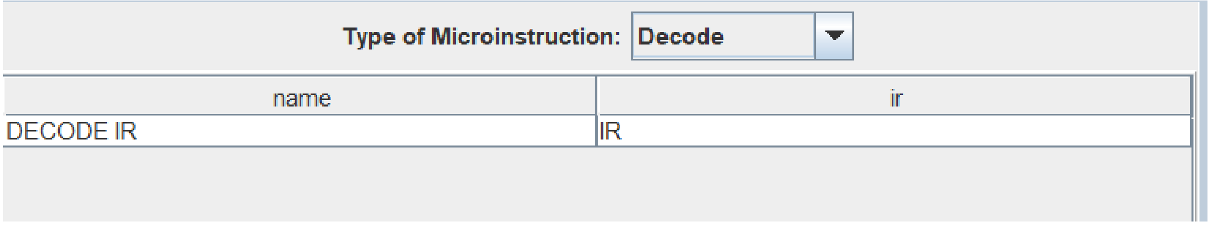
Type of Microinstruction: MemoryAccess

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)

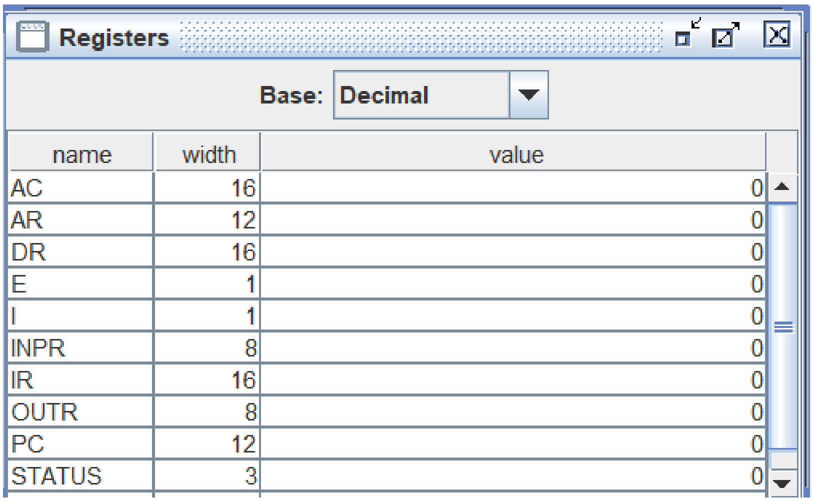
ADD NUM (DR <- M(AR) & AC <- AC + DR)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

NUM: .data 1 0

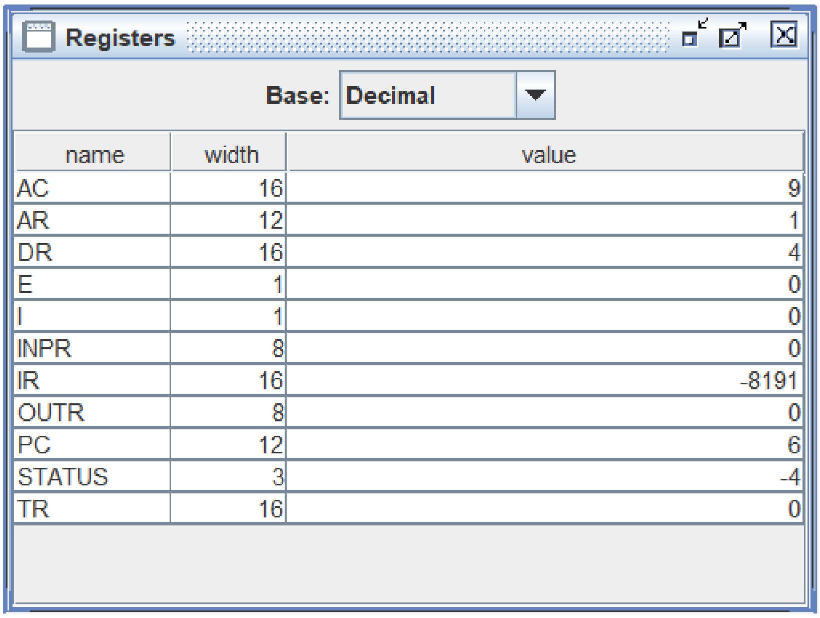
VALUE OF REGISTER BEFORE EXECUTION:



Base: Decimal

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

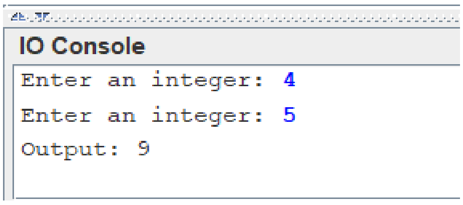
VALUE OF REGISTER AFTER EXECUTION:



Base: Decimal

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:



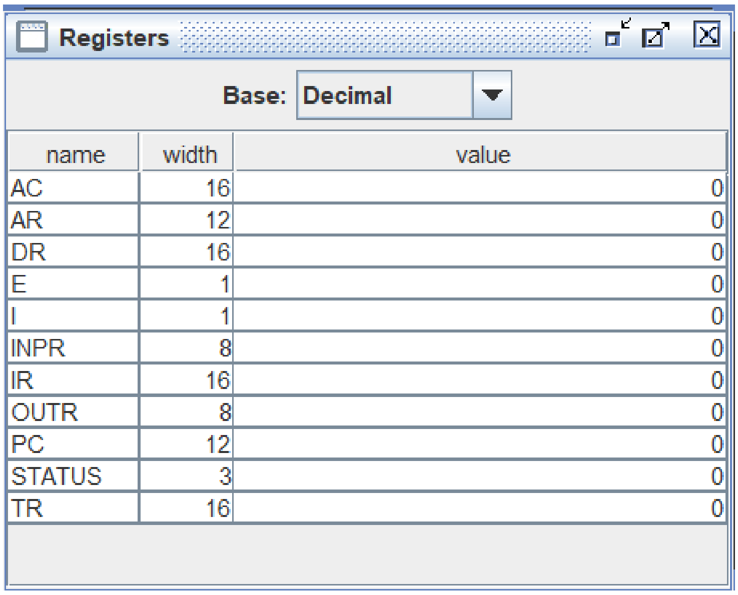
```
Enter an integer: 4
Enter an integer: 5
Output: 9
```

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
```

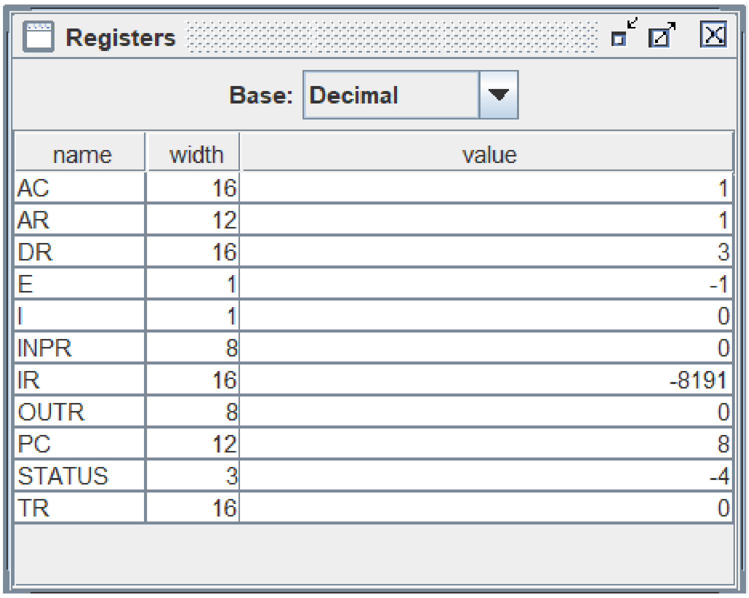
VALUE OF REGISTERS BEFORE EXECUTION:



Base: Decimal

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

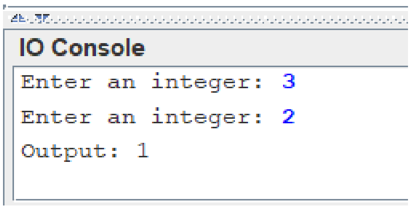
VALUE OF REGISTERS AFTER EXECUTION:



Base: Decimal

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:



IO Console

```
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)

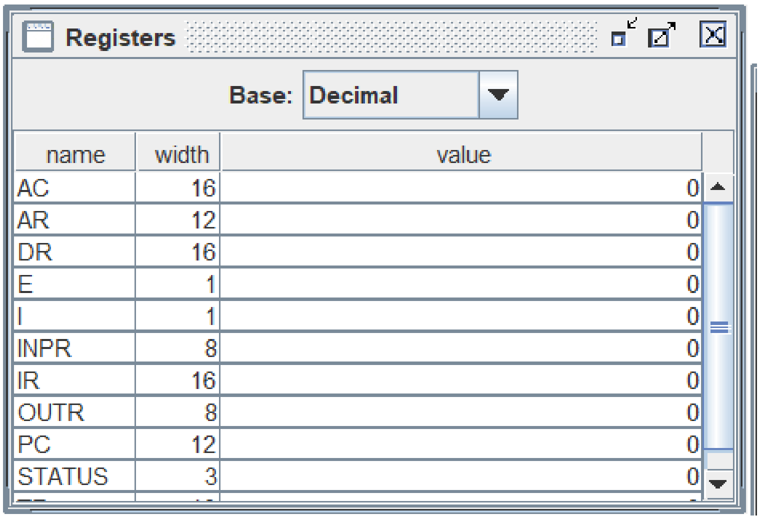
AND NUM (DR <- M(AR) & AC <- AC ^ DR)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

NUM: .data 1 0

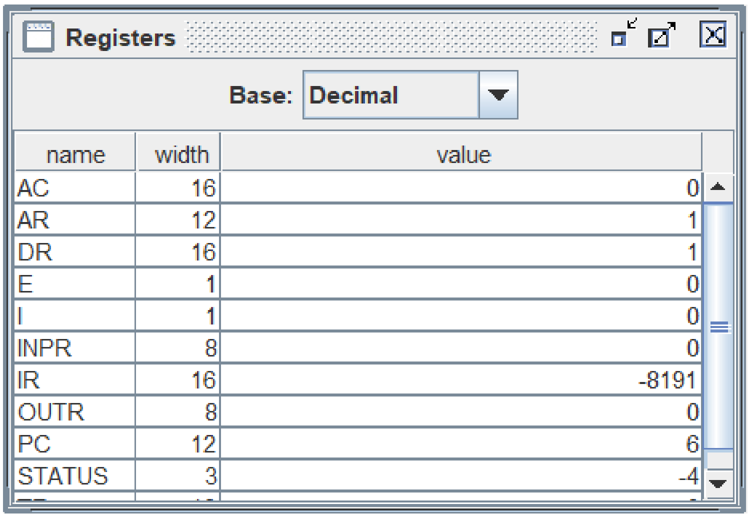
VALUE OF REGISTER BEFORE EXECUTION:



Base: Decimal

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

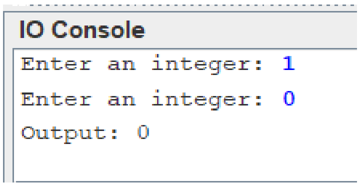
VALUE OF REGISTER AFTER EXECUTION:



Base: Decimal

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)

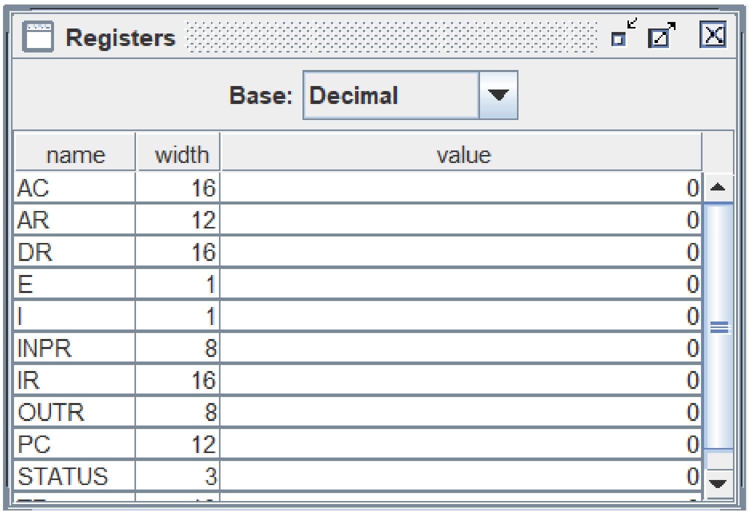
OR NUM (DR <- M(AR) & AC <- AC + DR)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

NUM: .data 1 0

VALUE OF REGISTER BEFORE EXECUTION:

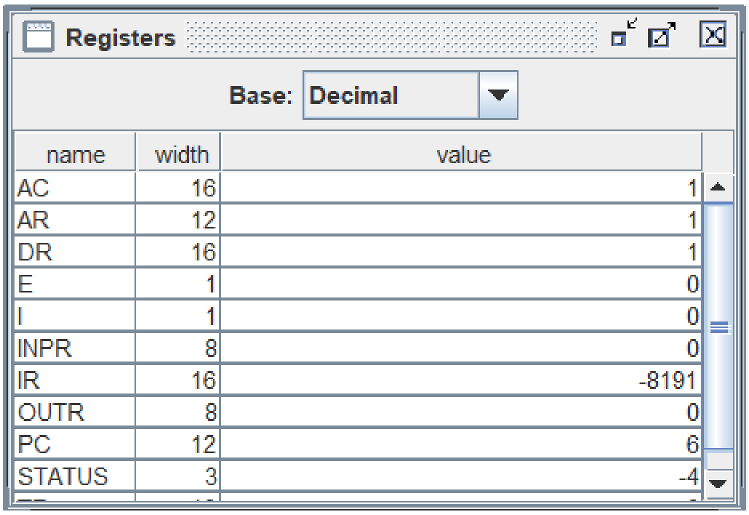


Registers

Base: Decimal

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:

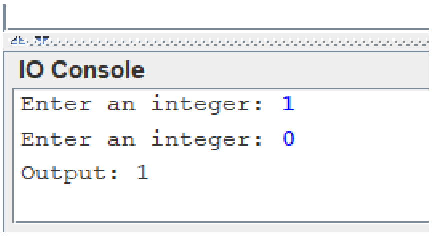


Registers

Base: Decimal

name	width	value
AC	16	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
```

3.NOT

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

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

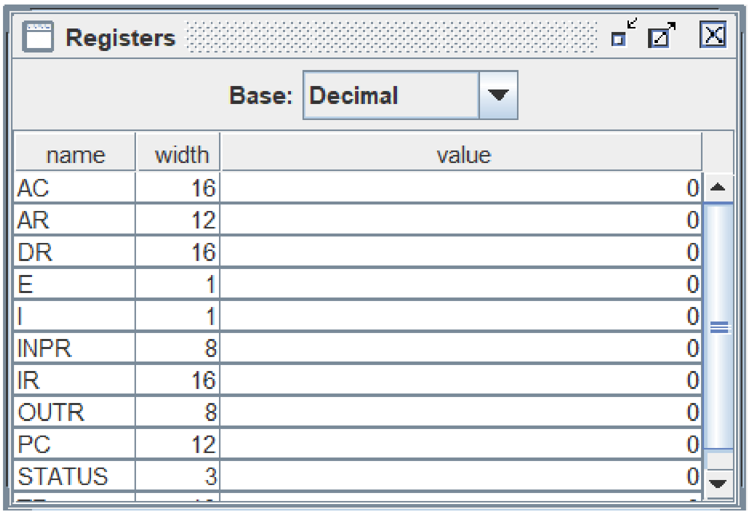
NOT NUM (AC <- AC')

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

NUM: .data 1 0

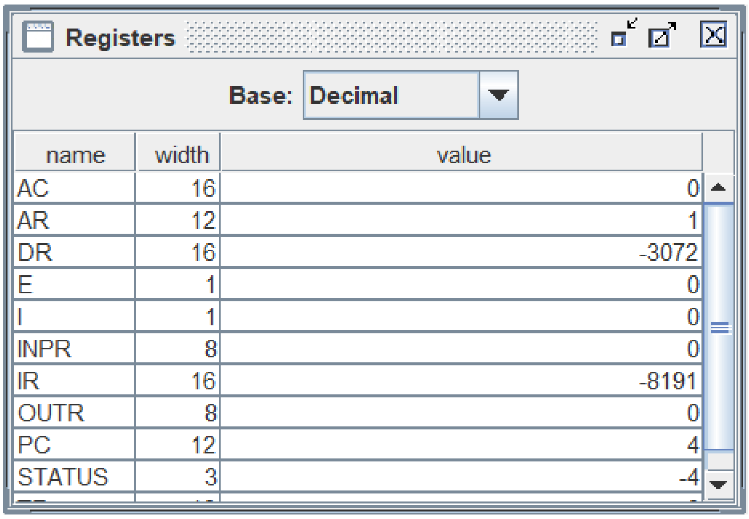
VALUE OF REGISTER BEFORE EXCUTION:



Base: Decimal

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:

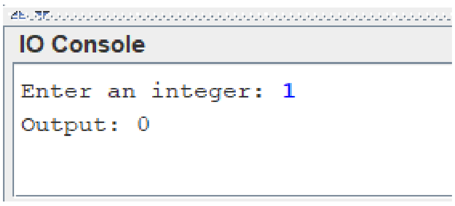


Registers

Base: Decimal

name	width	value
AC	16	0
AR	12	1
DR	16	-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:



IO Console

```
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)

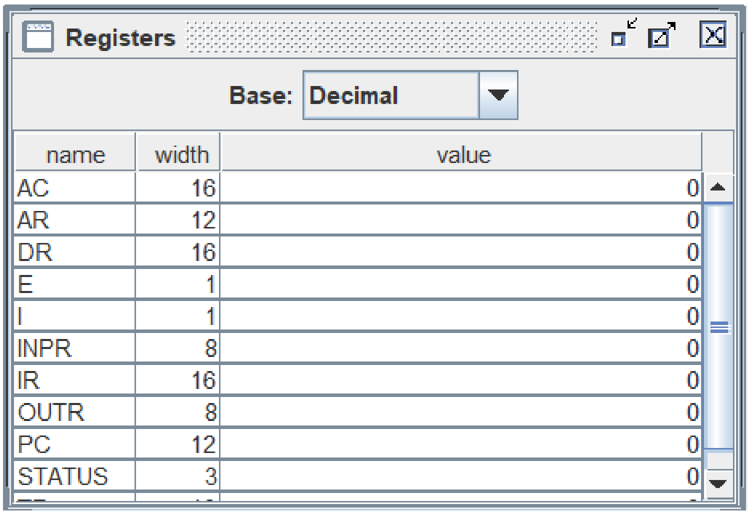
XOR NUM (DR <- M(AR) & AC <- AC xor DR)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

NUM: .data 1 0

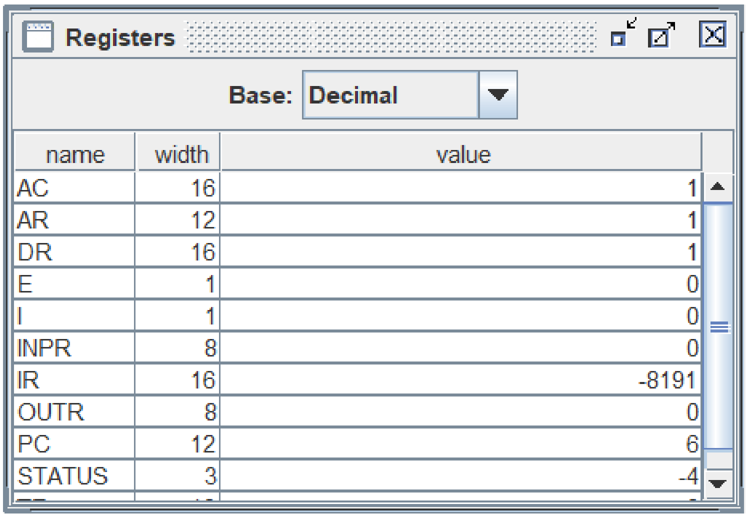
VALUE OF REGISTER BEFORE EXECUTION:



Base: Decimal

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

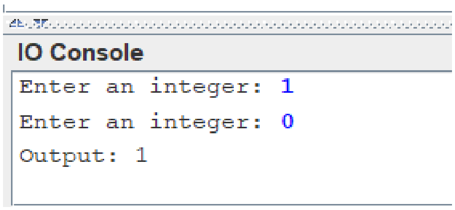
VALUE OF REGISTER AFTER EXECUTION:



Base: Decimal

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:



```
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)

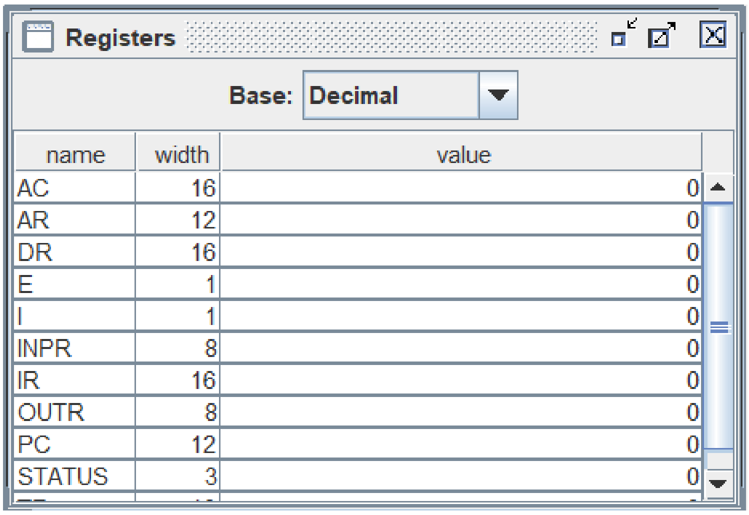
NOR NUM (DR <- M(AR) & AC <- AC nor DR)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

NUM: .data 1 0

VALUE OF REGISTER BEFORE EXECUTION:

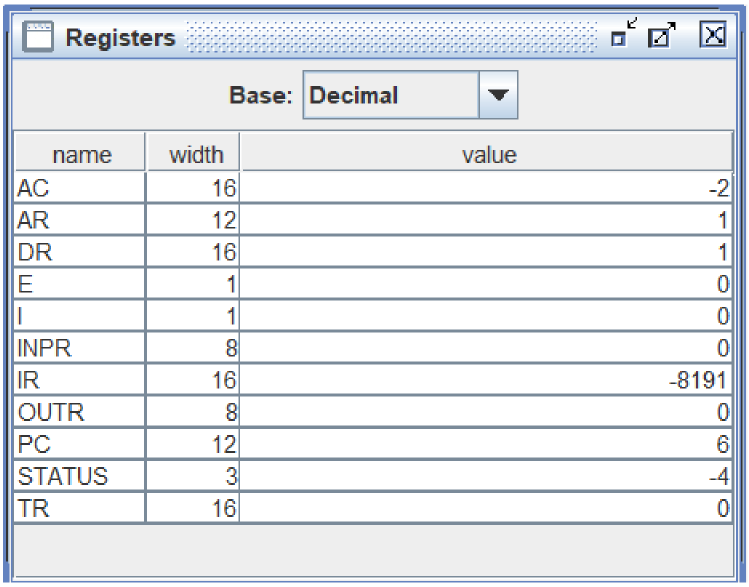


Registers

Base: Decimal

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:

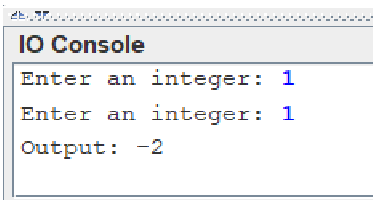


Registers

Base: Decimal

name	width	value
AC	16	-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:



IO Console

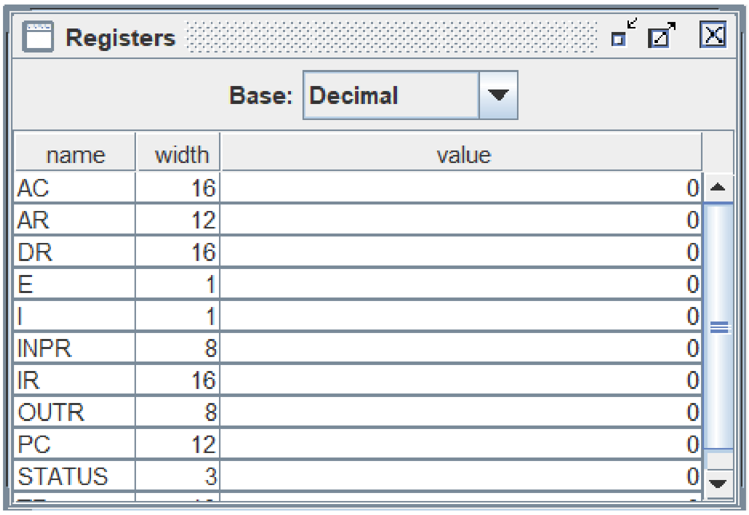
```
Enter an integer: 1
Enter an integer: 1
Output: -2
```

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:

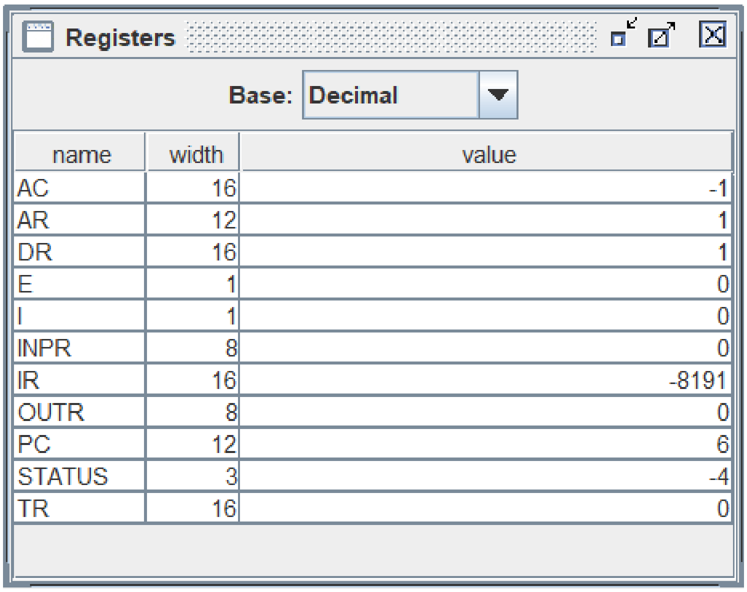


Registers

Base: Decimal

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:

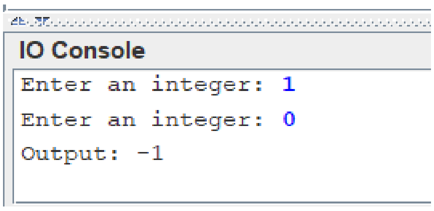


Registers

Base: Decimal

name	width	value
AC	16	-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:



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

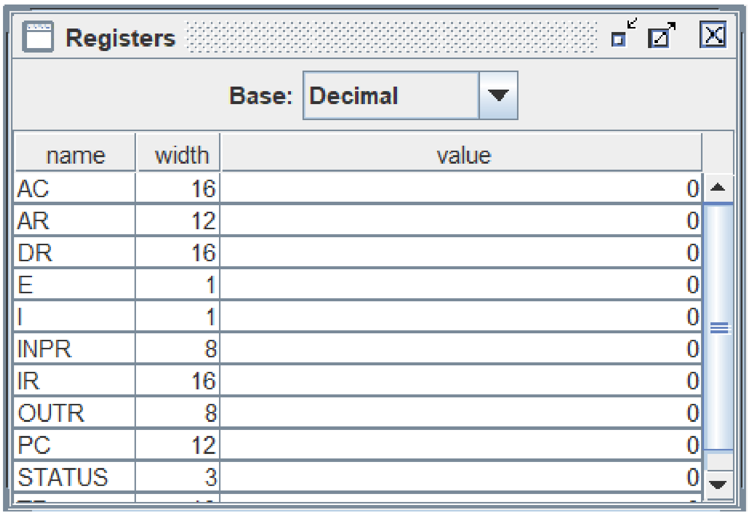
ANS:ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

1.ADD

```
 INP (TAKES INPUT FROM USER AND STORE IT IN AC)
 STA NUM ( M(AR) <- 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:

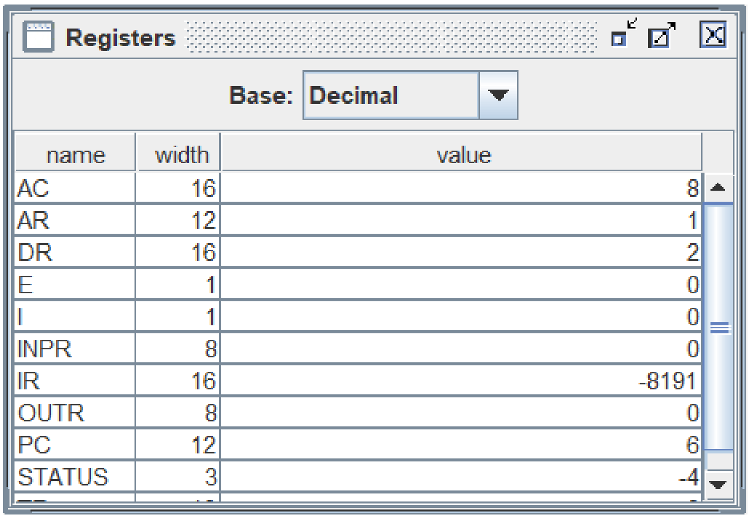


Registers

Base: Decimal

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:

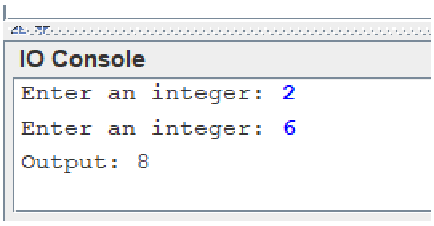


Registers

Base: Decimal

name	width	value
AC	16	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:



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) <- AC)

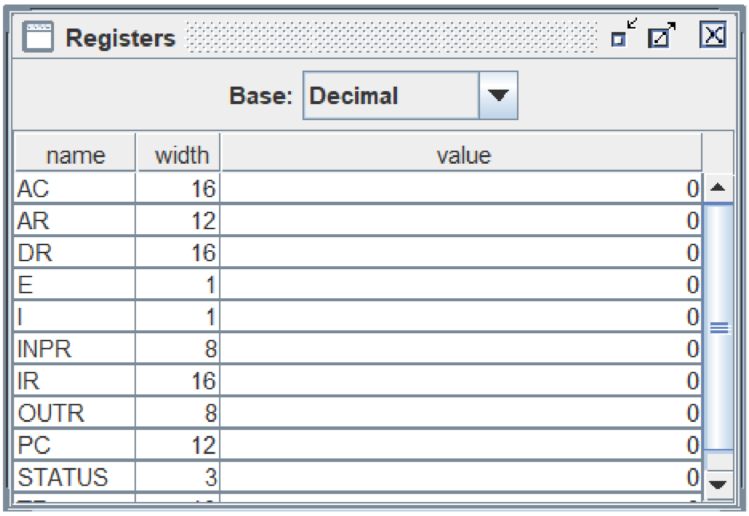
LDA NUM (DR <- M(AR) AND AC <- DR)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

NUM: .data 1 0

VALUE OF REGISTER BEFORE EXECUTION:

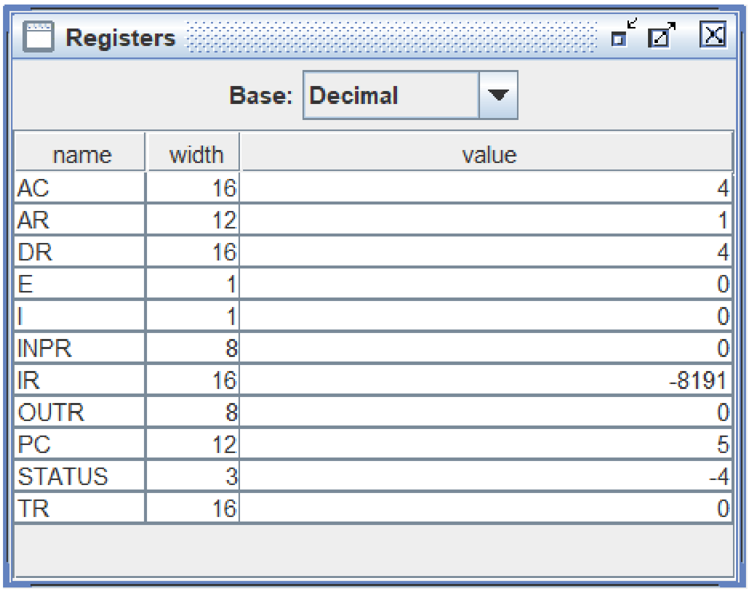


Registers

Base: Decimal

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:

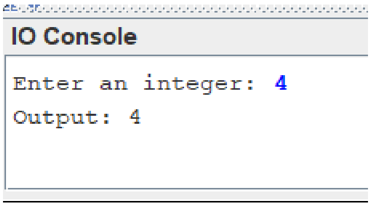


Registers

Base: Decimal

name	width	value
AC	16	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:



IO Console

```
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)

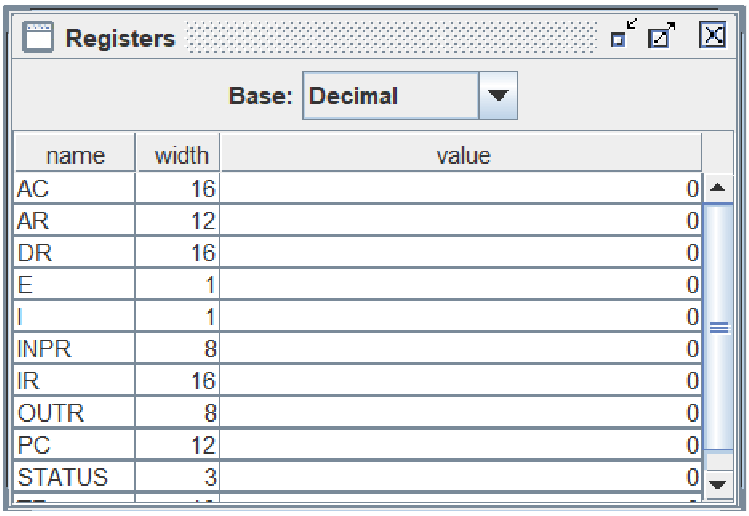
STA NUM (M(AR) <- AC)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

NUM: .data 1 0

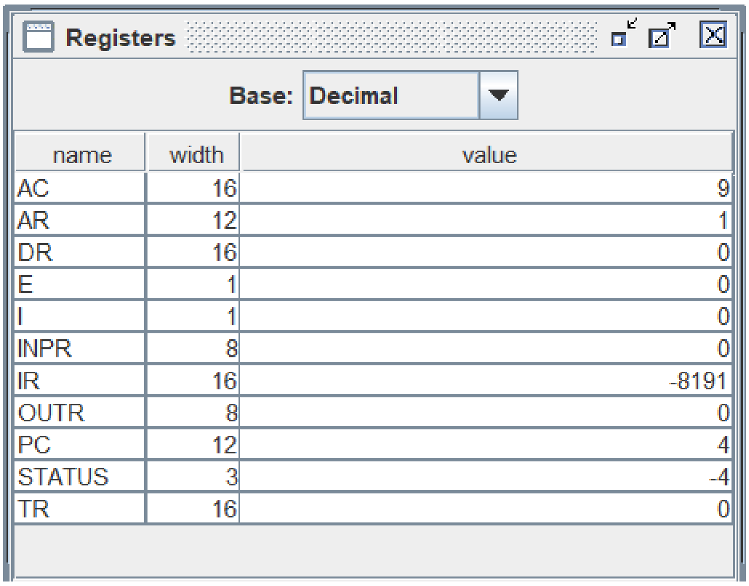
VALUE OF REGISTER BEFORE EXECUTION:



Base: Decimal

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

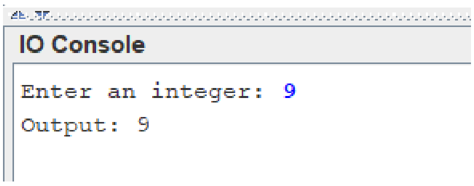
VALUE OF REGISTER AFTER EXECUTION:



Base: Decimal

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:



```
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)

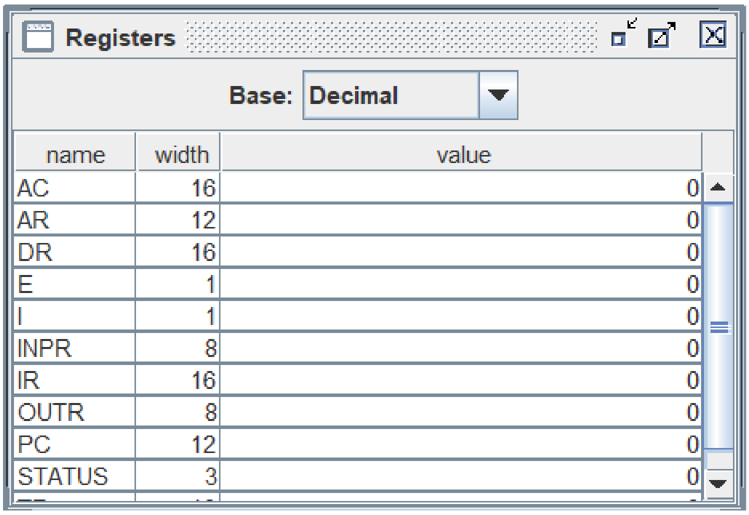
BUN K (PC <- AR AND K ACTS AS A FLAG)

INP

K: OUT

HLT (HALT–BIT = 1)

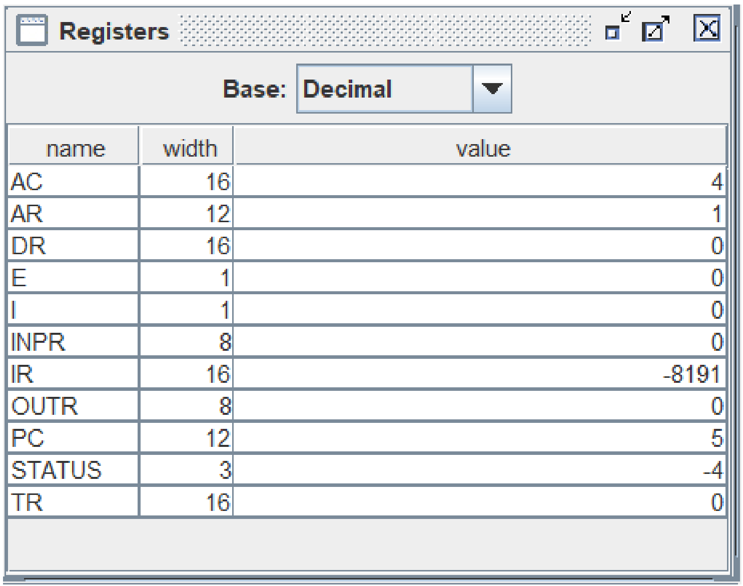
VALUE OF REGISTER BEFORE EXECUTION:



Base: Decimal

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:

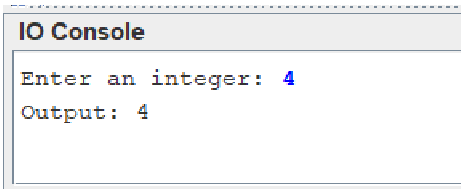


Registers

Base: Decimal

name	width	value
AC	16	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:



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

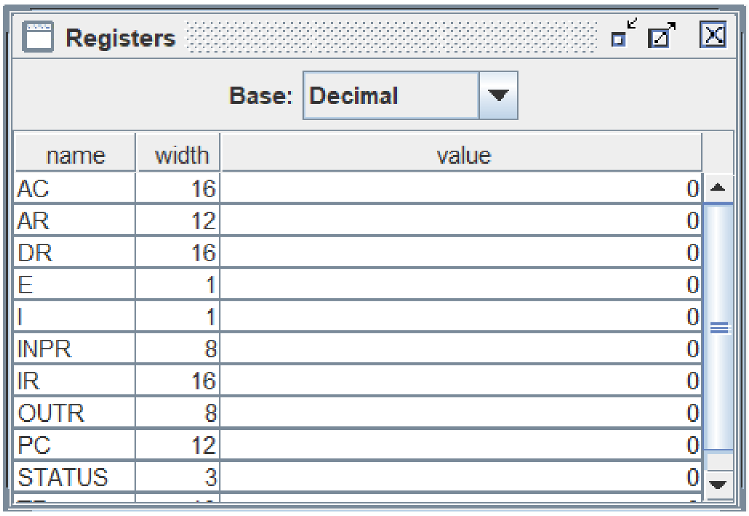
HLT (HALT–BIT = 1)

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.

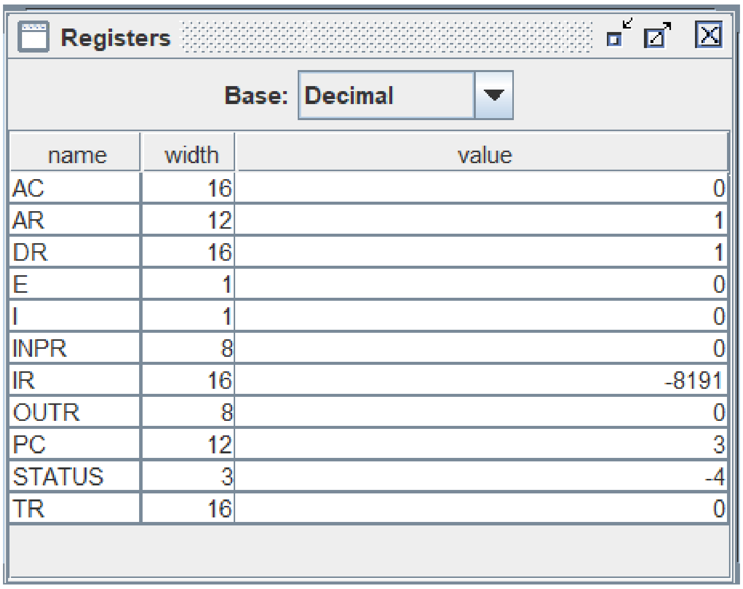
VALUE OF REGISTER BEFORE EXECUTION:



Base: Decimal

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

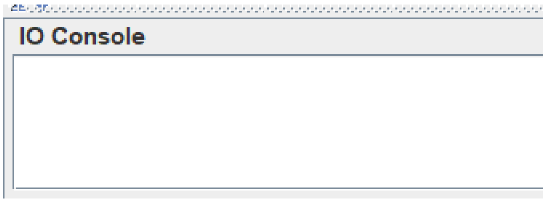
VALUE OF REGISTER AFTER EXECUTION:



Base: Decimal

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

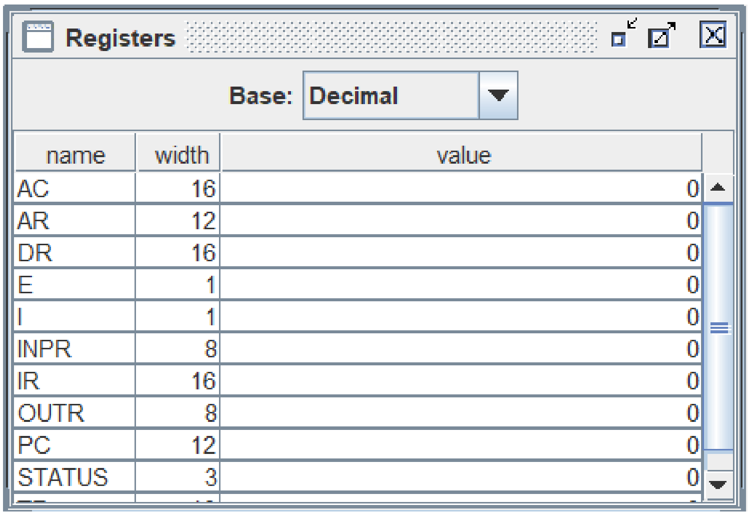
ANS:1. CLA : Clear Accumulator

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

CLA (AC <- 0)

HLT (HALT–BIT = 1)

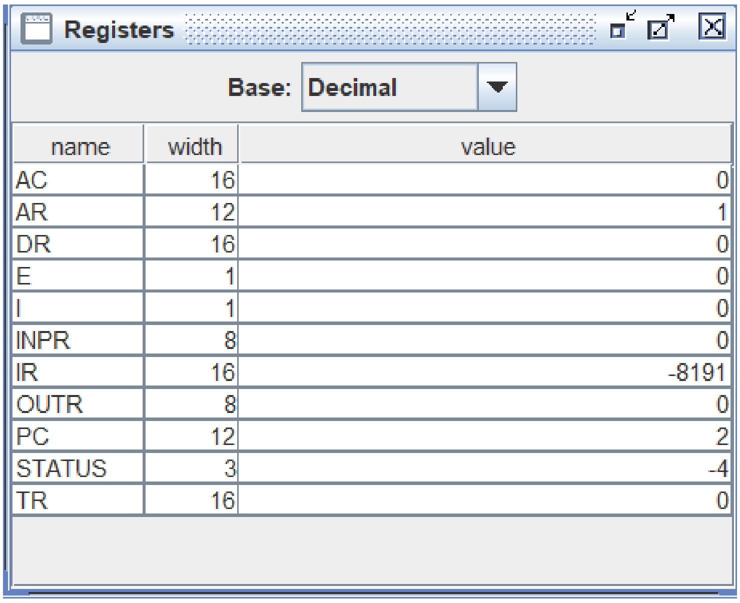
VALUE OF REGISTER BEFORE EXECUTION:



Base: Decimal

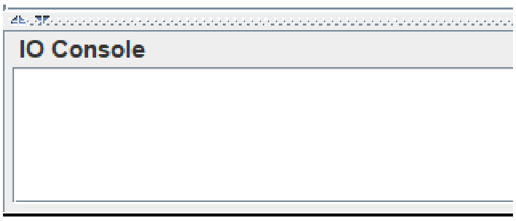
name	width	value
AC	16	0
AR	12	0
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	0
OUTR	8	0
PC	12	0
STATUS	3	0

VALUE OF REGISTER AFTER EXECUTION:



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	2
STATUS	3	-4
TR	16	0



NOTE : IO CONSOLE WILL STAY EMPTY.

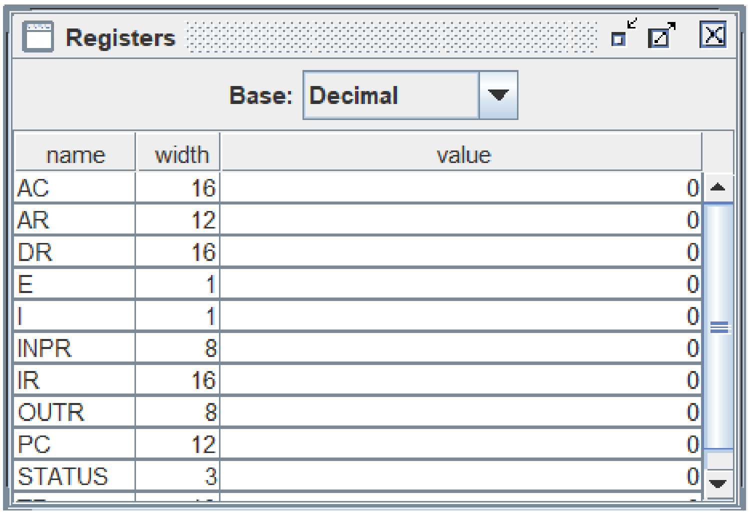
2. CMA : Complement Accumulator

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

CMA (AC <- AC’)

HLT (HALT–BIT = 1)

VALUE OF REGISTER BEFORE EXECUTION:

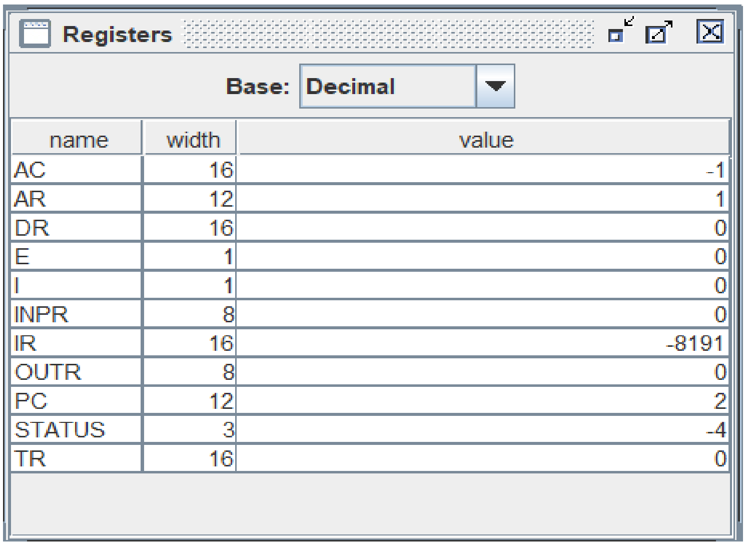


Registers

Base: Decimal

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:

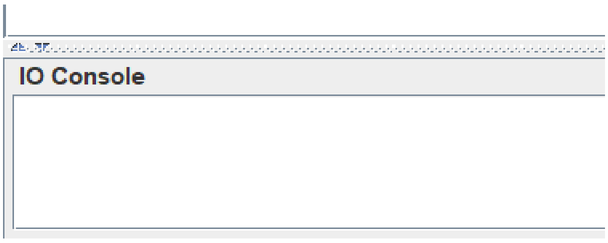


Registers

Base: Decimal

name	width	value
AC	16	-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:



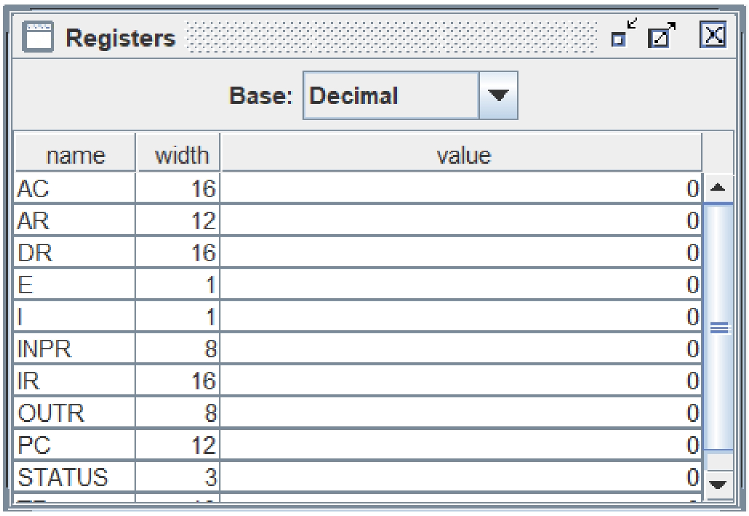
3. CME : Complement Extended Bit

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS :

CME (E <- E’)

HLT (HALT–BIT = 1)

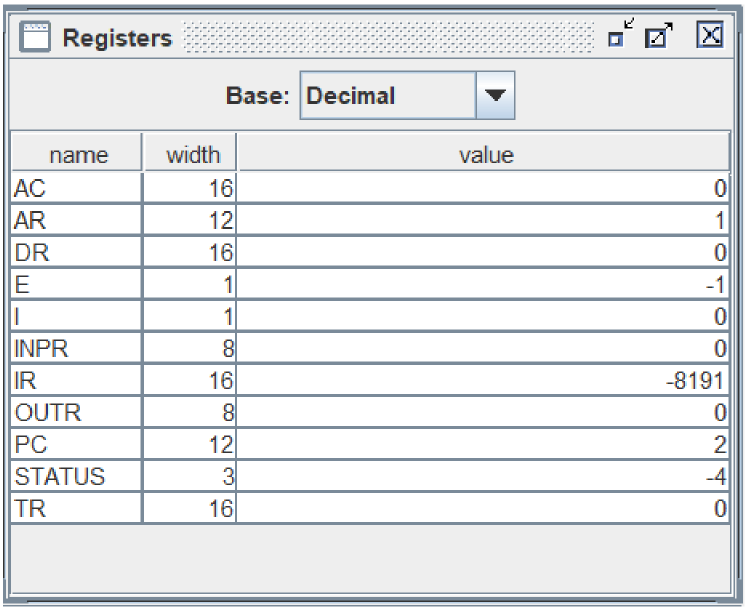
VALUE OF REGISTER BEFORE EXECUTION:



Base: Decimal

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:

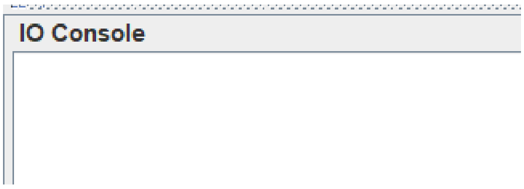


Registers

Base: Decimal

name	width	value
AC	16	0
AR	12	1
DR	16	0
E	1	-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:



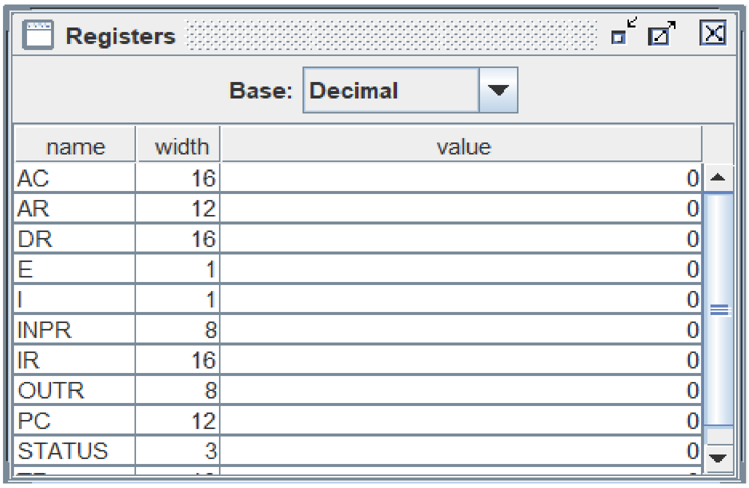
IO Console

4. HLT : HALT

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS

HLT (HALT–BIT = 1 AND END)

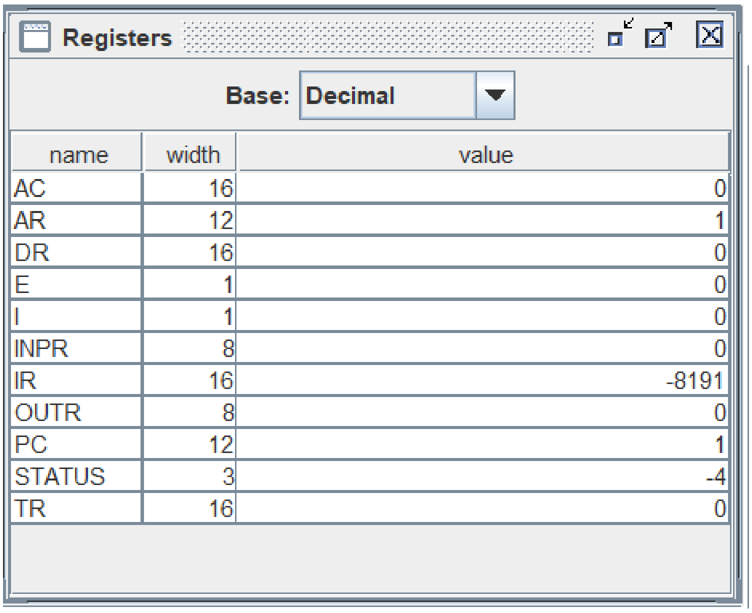
VALUE OF REGISTER BEFORE EXECUTION:



Base: Decimal

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:



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	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

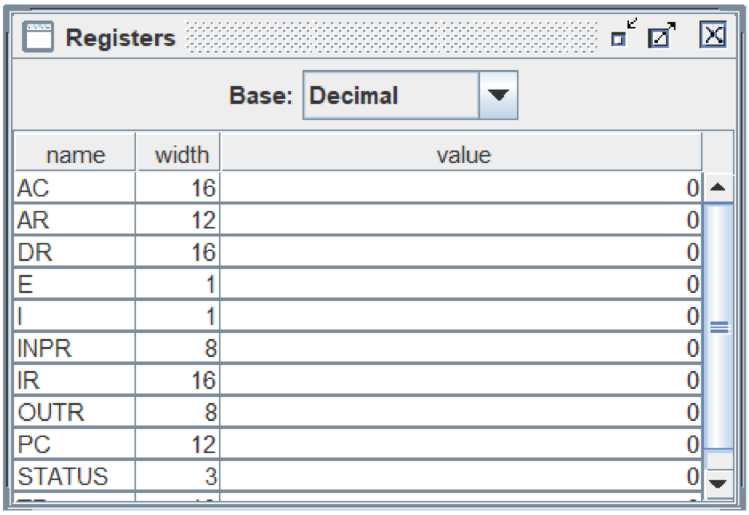
INP (TAKES INPUT FROM USER AND STORE IT IN AC)

INC (AC <- AC + 1)

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

VALUE OF REGISTER BEFORE EXECUTION:

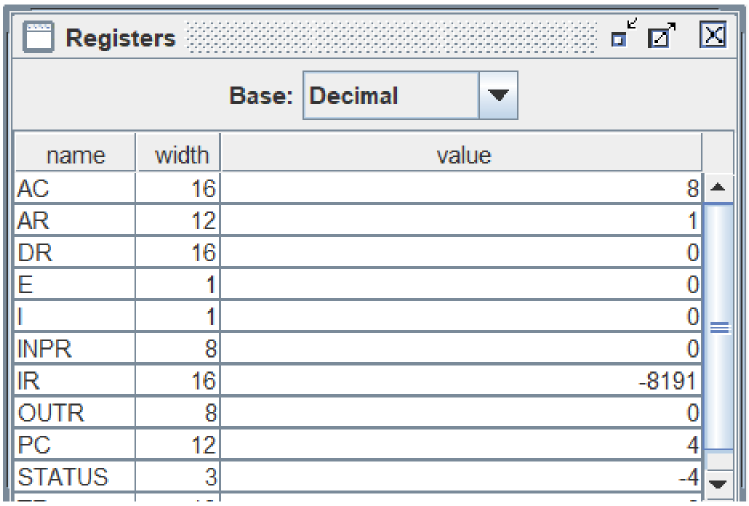


Registers

Base: Decimal

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:

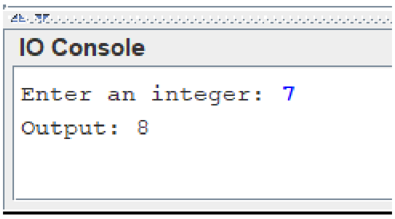


Registers

Base: Decimal

name	width	value
AC	16	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:



IO Console

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

2. SPA : Skip if Positive

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS

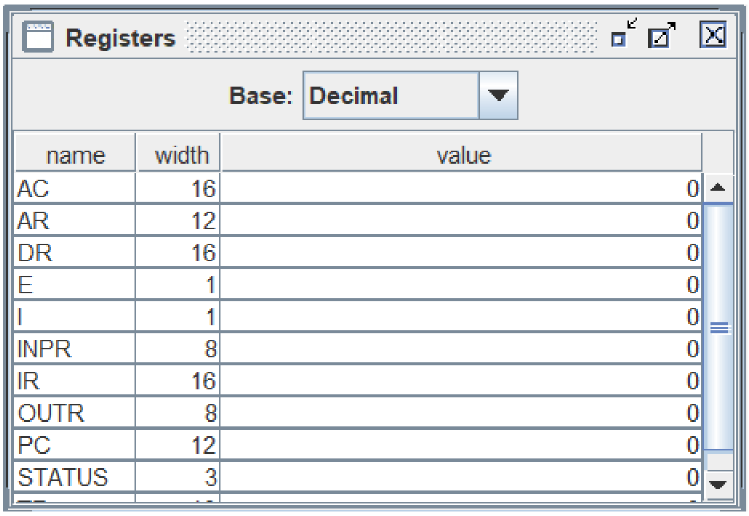
INP (TAKES INPUT FROM USER AND STORE IT IN AC)

SPA

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

VALUE OF REGISTER BEFORE EXECUTION:

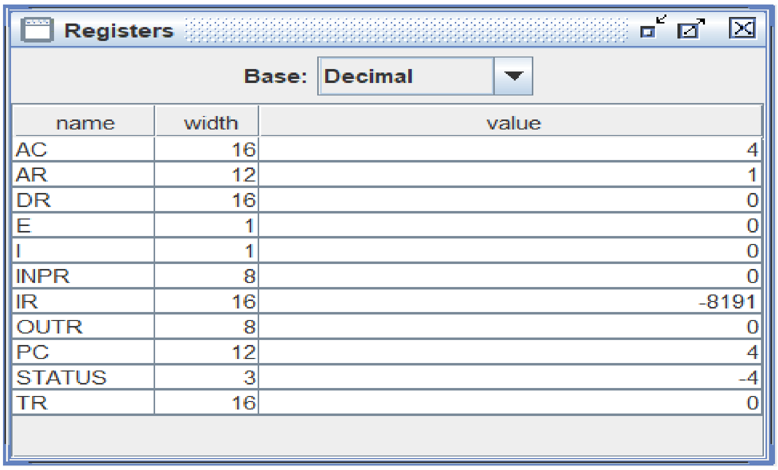


Registers

Base: Decimal

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:

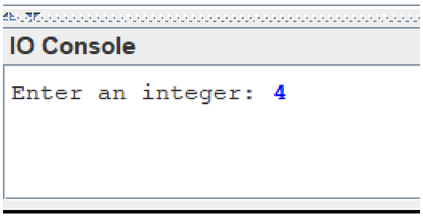


Registers

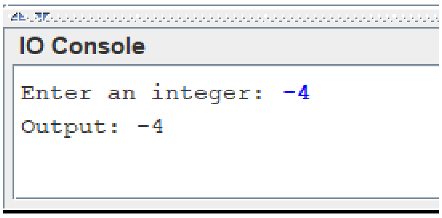
Base: Decimal

name	width	value
AC	16	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:



```
Enter an integer: 4
```



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

3. SNA : Skip if Negative

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS

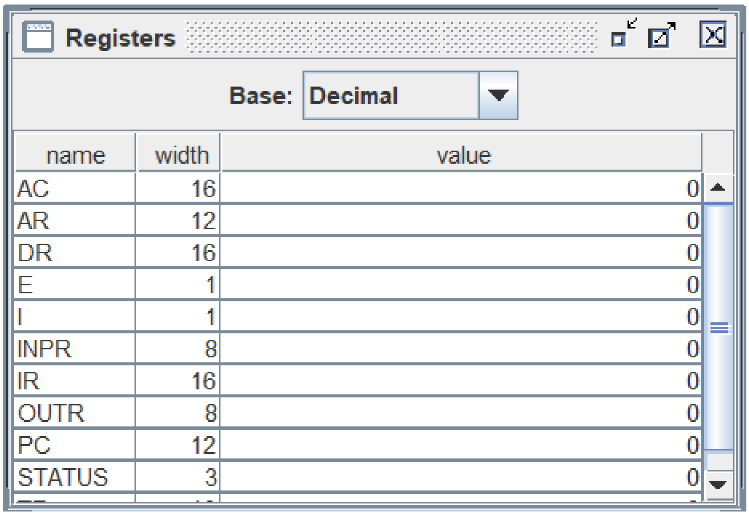
INP (TAKES INPUT FROM USER AND STORE IT IN AC)

SNA

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

VALUE OF REGISTER BEFORE EXECUTION:

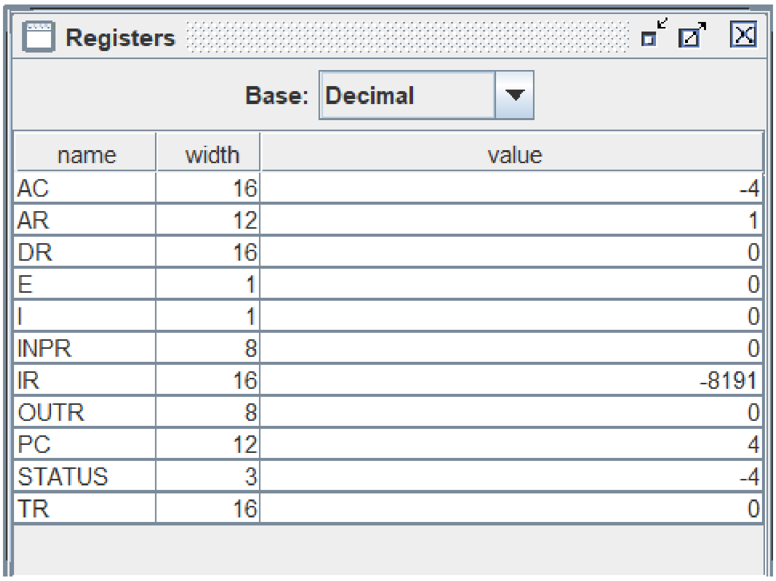


Registers

Base: Decimal

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:

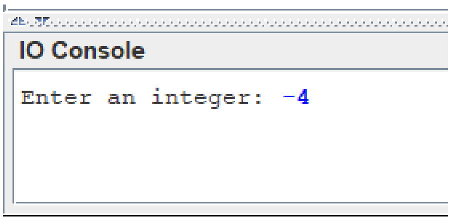


Registers

Base: Decimal

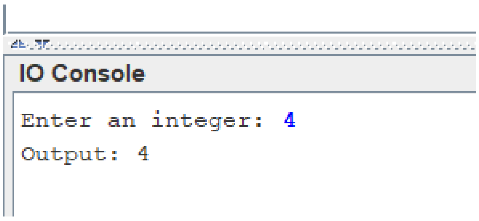
name	width	value
AC	16	-4
AR	12	1
DR	16	0
E	1	0
I	1	0
INPR	8	0
IR	16	-8191
OUTR	8	0
PC	12	4
STATUS	3	-4
TR	16	0

INPUT OUTPUT WINDOW:



IO Console

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



IO Console

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

4. SZE : Skip if Extended bit is 0

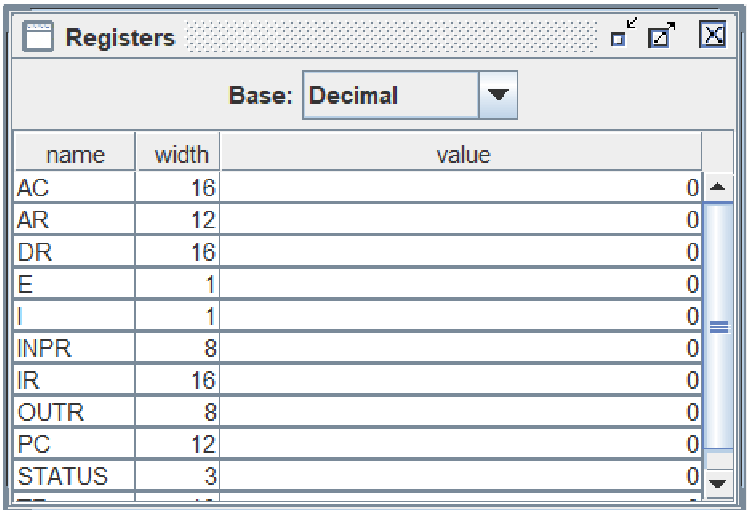
ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS

SZE

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

VALUE OF REGISTER BEFORE EXECUTION:

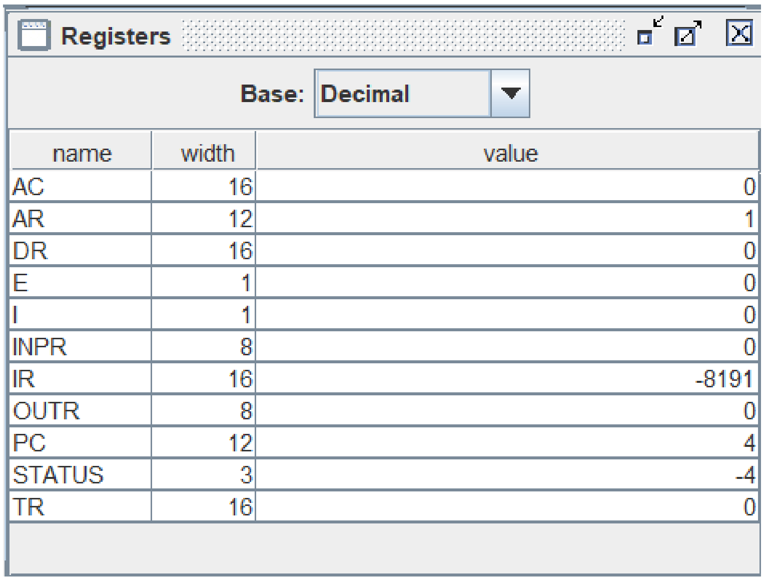


Registers

Base: Decimal

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

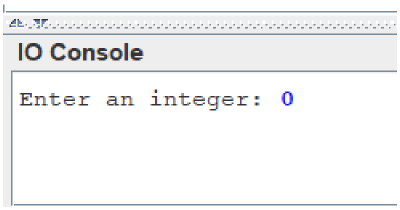
VALUE OF REGISTER AFTER EXECUTION:



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:



```
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

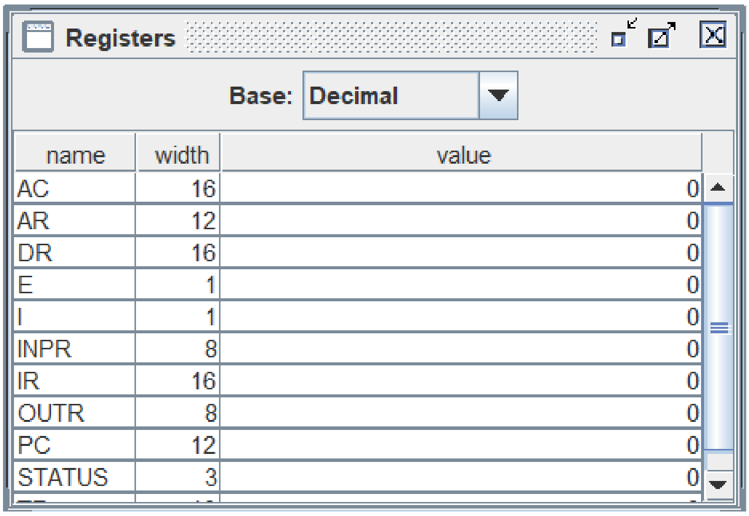
INP (TAKES INPUT FROM USER AND STORE IT IN AC)

CIR

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

VALUE OF REGISTER BEFORE EXECUTION:

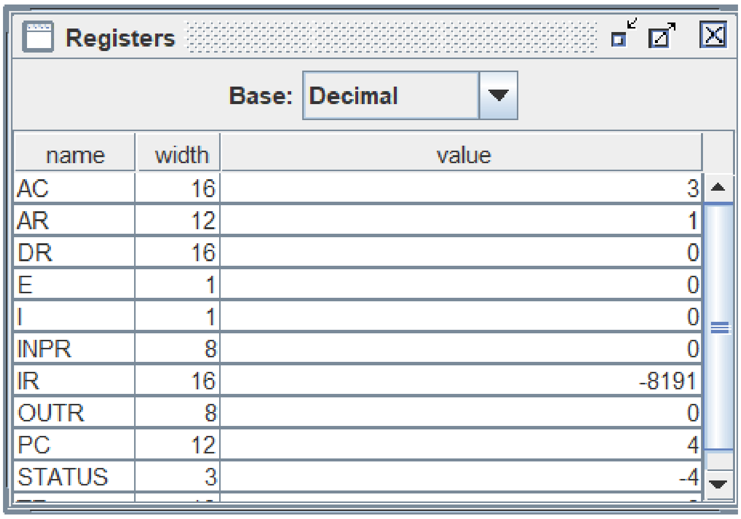


Registers

Base: Decimal

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

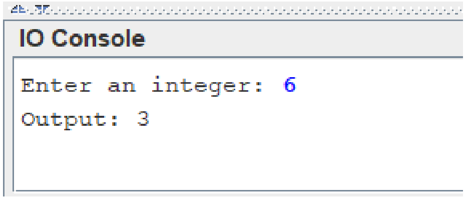
VALUE OF REGISTER AFTER EXECUTION:



Base: Decimal

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:



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

2. CIL : Circulate Left

ASSEMBLY LANGUAGE PROGRAM ALONG WITH MICROINSTRUCTIONS

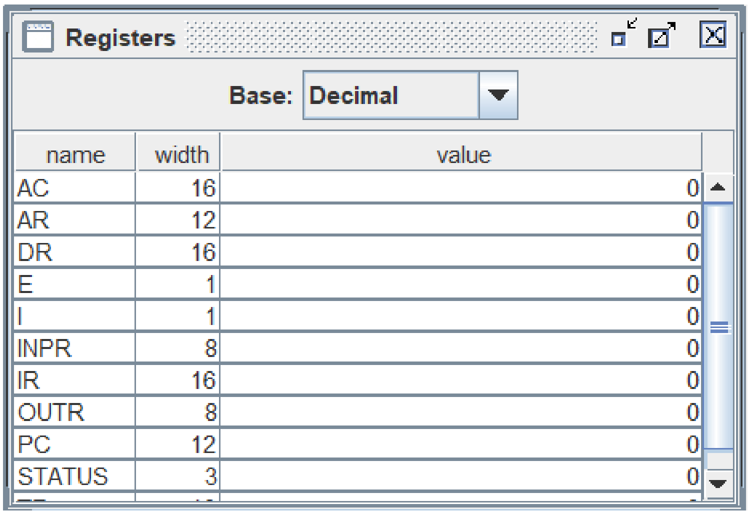
INP (TAKES INPUT FROM USER AND STORE IT IN AC)

CIL

OUT (TAKES OUTPUT FROM AC AND DISPLAY ON SCREEN)

HLT (HALT–BIT = 1)

VALUE OF REGISTER BEFORE EXECUTION:

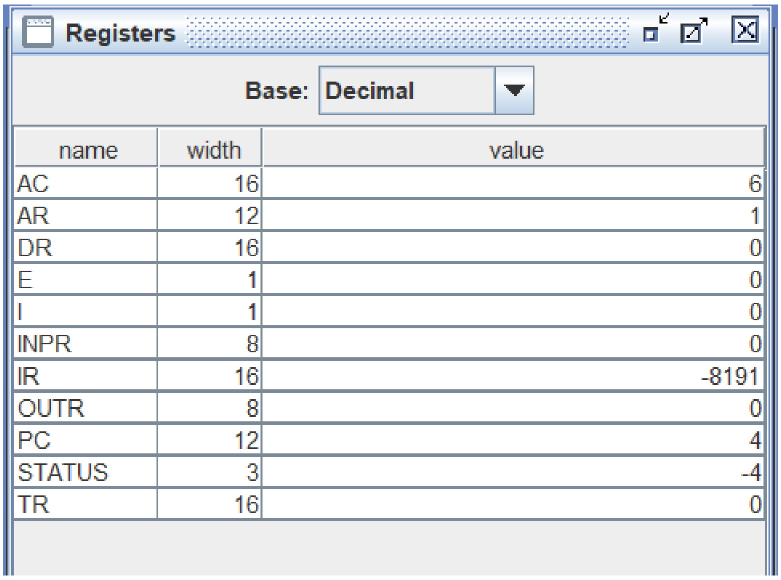


Registers

Base: Decimal

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:

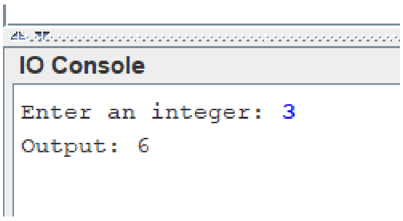


Registers

Base: Decimal

name	width	value
AC	16	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:



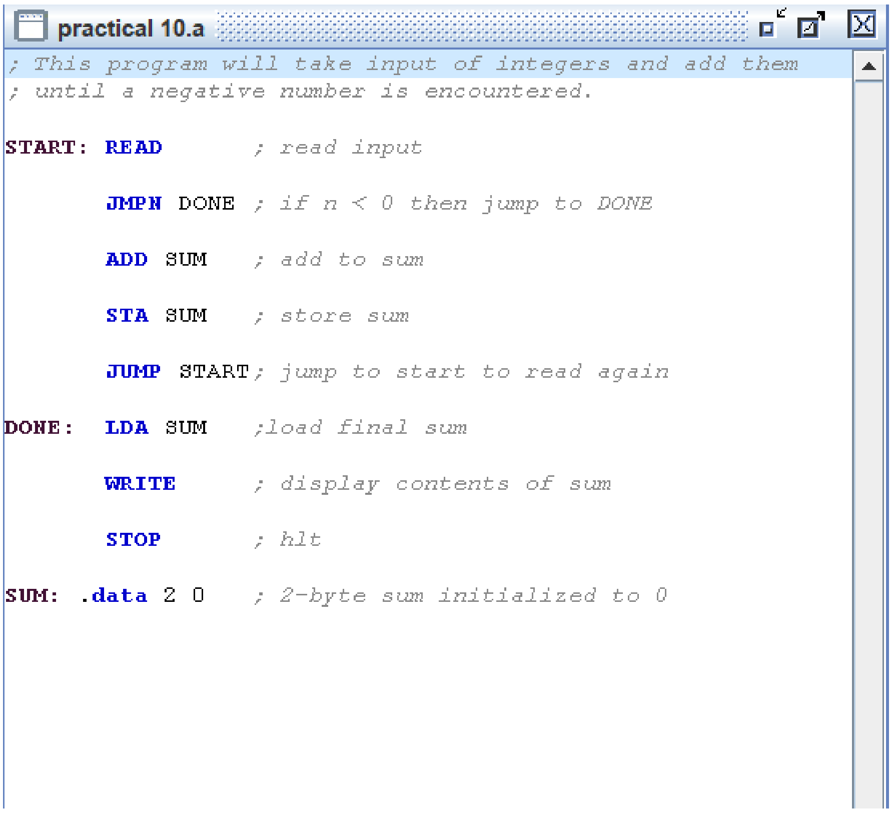
IO Console

```
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:



```
; 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 :

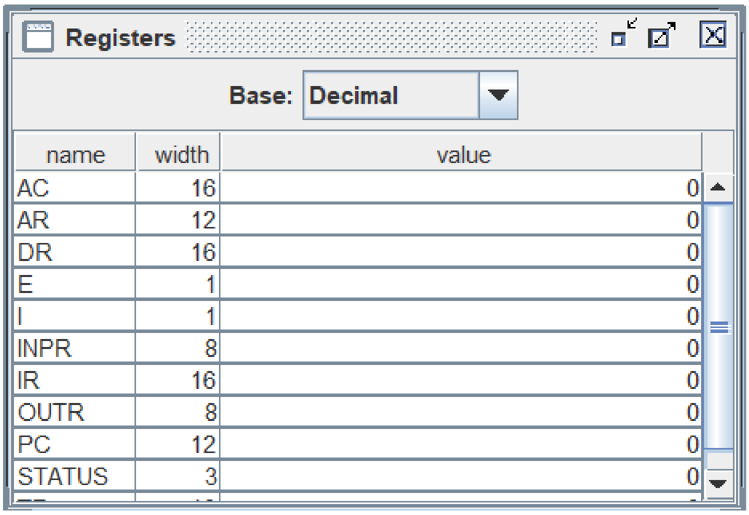
IF (AC > 0) SKIP 1

```
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

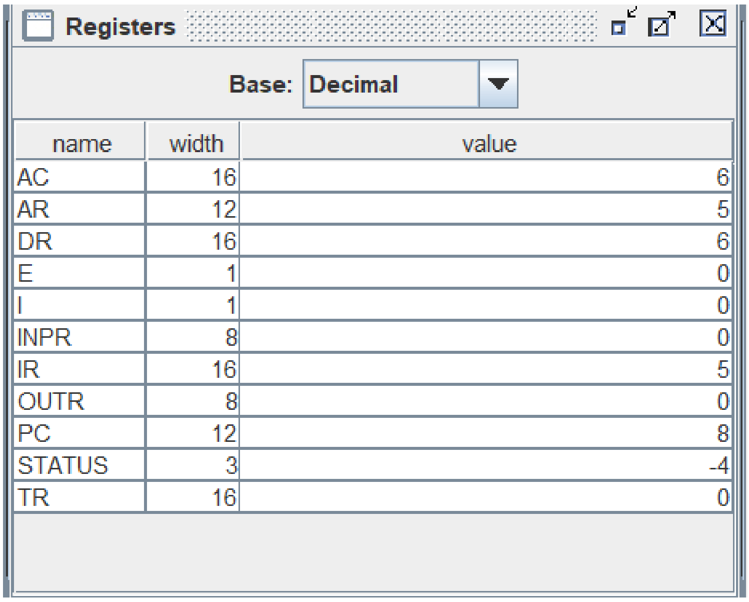
VALUE OF REGISTER BEFORE EXECUTION:



Base: Decimal

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

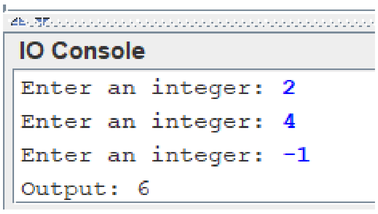
VALUE OF REGISTER AFTER EXECUTION:



Base: Decimal

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

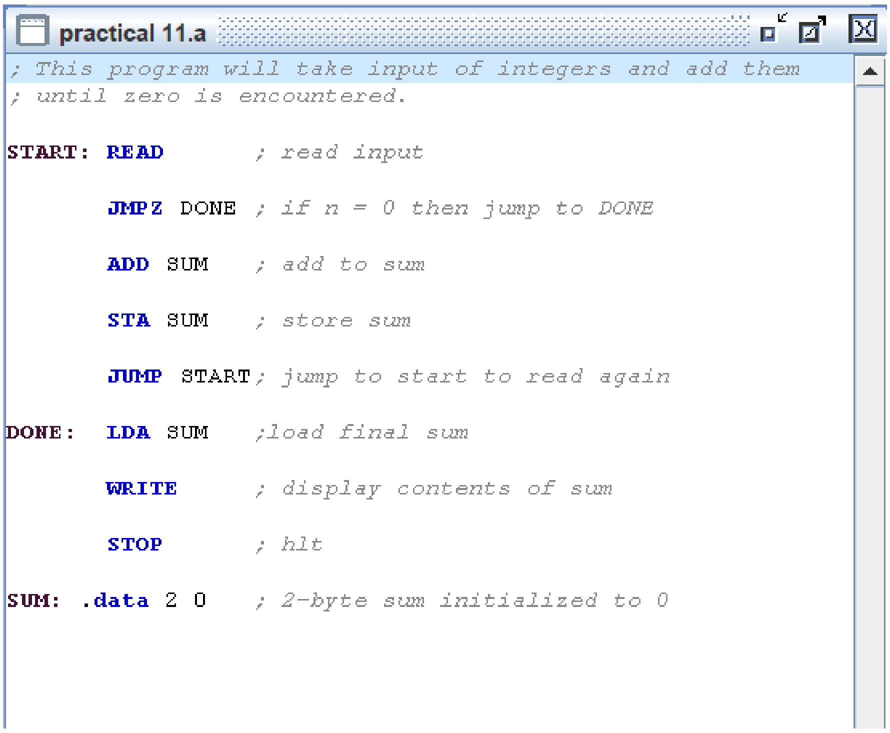
INPUT OUTPUT WINDOW:



```
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



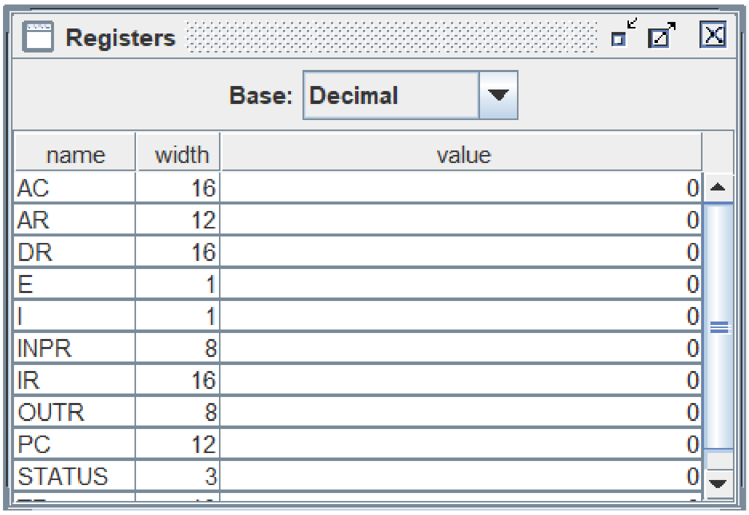
```
; This program will take input of integers and add them
; until zero is encountered.

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
```

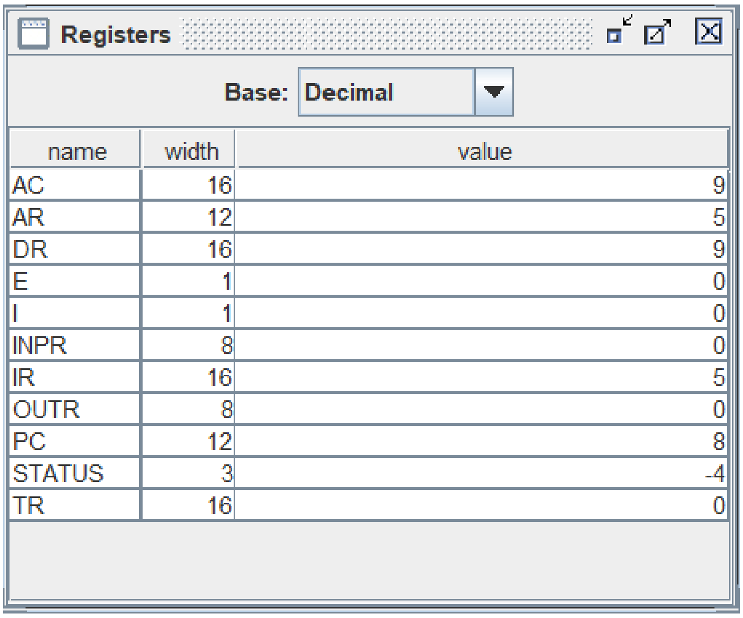
VALUE OF REGISTER BEFORE EXECUTION:



Base: Decimal

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

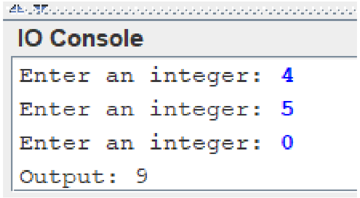
VALUE OF REGISTER AFTER EXECUTION:



Base: Decimal

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:



```
Enter an integer: 4
Enter an integer: 5
Enter an integer: 0
Output: 9
```