EOC ASSIGNMENT-2

Team Members: -

BATCH-A	TEAM-7
GAJULA SRI VATSANKA	CB.EN.U4AIE.21010
GUNNAM HIMAMSH	CB.EN.U4AIE.21014
M.PRASANNA TEJA	CB.EN.U4AIE.21035
VIKHYAT BANSAL	CB.EN.U4AIE.21076

QUESTION-1

Understand the basic concept of hack machine language in working with registers and memory for ‘SUB’ operation. Write and execute the hack assembly language program.

GAJULA SRI VATSANKA [CB.EN.U4AIE.21010]

HACK MACHINE LANGUAGE

Hack is a 16-bit assembly language assembler for the Hack Assembly Language. This was done as part of the book and Elements of Computing Systems, which is commonly known as nand2tetris, which teaches how to build a whole 16-bit computer from the ground up.

REGISTER

- Represents a set of memory banks used by the processor itself.
- The Read data and Write data registers temporarily store information that is loaded into or retrieved from the computer's memory.

MEMORY

- Data Memory represents the allotted storage space of the computer, which the processor must frequently access to store and retrieve information when an instruction calls for such action.

CODE EXPLANATION

```
@0
D=M
```

- // **0** is used to denote RAM[0] which takes input from user (when the code is executed) and store it in D-Register.
  D denotes value in RAM[0].

```
@1
D=D-M
```

- // 1 is used to denote RAM[1] which takes input from user.
  M denotes value of RAM[1].

@2
M=D

- // 2 is used to denote RAM[2] which shows the output 9when code is executed)
  M is stored as D- register

@6
0;jmp

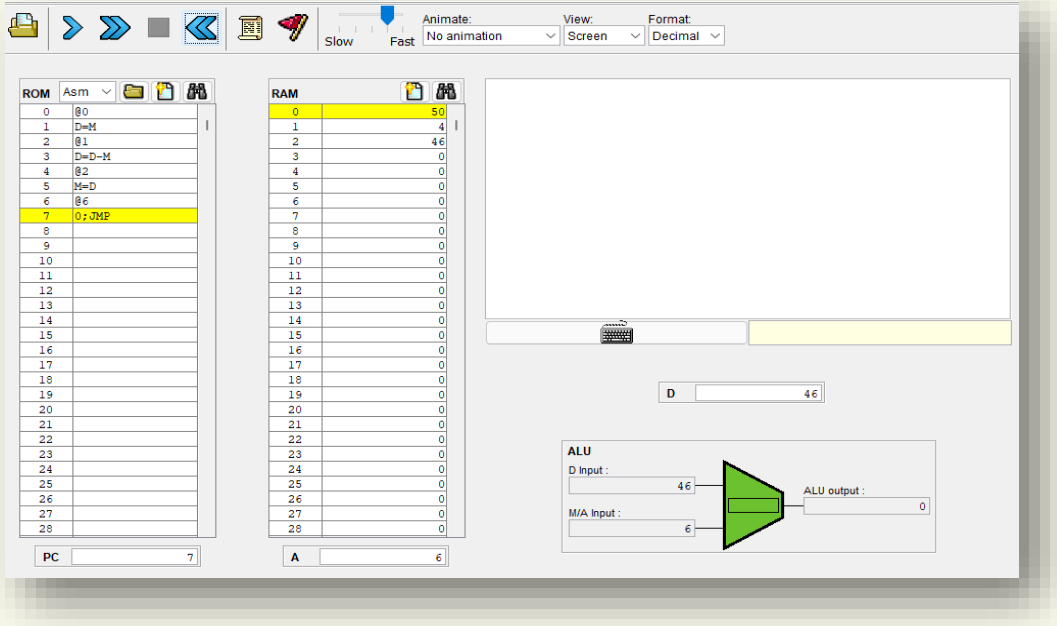
- // jmp indicates [unconditional jump to discontinue loop]

HDL CODE FOR "SUB"

```
//program sub.asm
//computes:RAM[2]=RAM[0]-RAM[1]
//useage:put values in RAM[0] and RAM[1]

@0
D=M  //D=RAM[0]

@1
D=D-M //D=Ram(0) - Ram(1)

@2
M=D  //RAM[2]=D

@6
0;JMP //unconditional jump to discontinue loop
```

OUTPUT

I have taken values of
RAM[0] = 50
RAM[1] = 4 ;
In RAM[2] we will get the difference between them as output.
[50-4 =46]



QUESTION-2

Write and execute the hack assembly language program for multiplication of two numbers.

GUNNAM HIMAMSH [CB.EN.U4AIE.21014]

HDL code for Multiplication of 2 numbers

```
// Put your code here.
@1
        D=M //copy R1 into D
        @times
        M=D // and now M[times] = R1

        // @sum will be the cumulative sum of the R0's; at end
        // of program, it's copied into R2 -> may as well always
        // live in R2
        @2
        M=0 //initialize sum to 0
(LOOP)
        // within the loop, if times = 0, need to break out
        @times
        D=M //d = times
        @END
        D; JEQ // if times = 0, break

        // if we're still looping, need to decrement times
        // and increment sum
        @1
        D=D-A //d = times - 1
        @times
        M=D // and now, times = times-1

        @2
        D=M
        @0
        D=D+M // d = R0 + SUM
        @2
        M=D //and, now sum = sum+R0

        @LOOP
        0;JMP

(END)   // and, the infinite loop that ends a hack program
        @END
        0; JMP
```

Explanation of Assembly language

- First go to address 1 by typing as @1 it represents A register. After that copy that into D register.
- next we have @16 now copy that to M register we know that M=RAM[A].
- Next, we have A=2 we wrote it as @2 we are summing up to zero that is M=0.

So here it is like RAM[2]=0

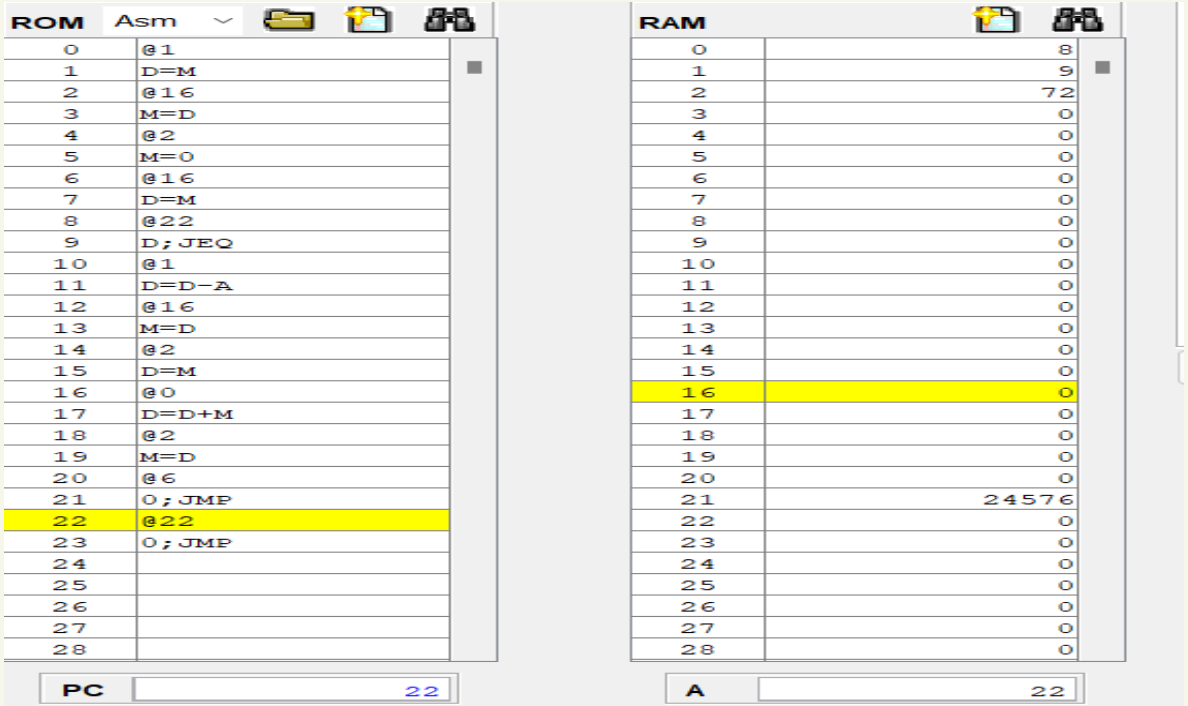
- Next start a loop by taking @16 as A value and next copy that value to D like D=M. After that end that and if it is equal to zero then break that it can be written as 0;JEQ.
- If we are still looping that we need to decrement @16 and increment sum. It is written as

D=D-A(we are decrementing it) after that copy that to M.

Explanation of Assembly language

- After that give A value as 2 and copy that to D by giving D=M. Next give A value as 0 and write as D=D+M this implies d=R0+sum , after that taking value as 2 we are doing sum=sum+R0.
- Now going back to loop by writing @loop it will go to loop we have written previously so it will become @6 and give an unconditional jump as 0;jmp.
- And next end the loop it is because to end the infinite hack program.
- So by writing this code and implementing in cpu emulator we obtain the following pic as given below

Result



ROM	Asm
0	@1
1	D=M
2	@16
3	M=D
4	@2
5	M=0
6	@16
7	D=M
8	@22
9	D;JEQ
10	@1
11	D=D-A
12	@16
13	M=D
14	@2
15	D=M
16	@0
17	D=D+M
18	@2
19	M=D
20	@6
21	0;JMP
22	@22
23	0;JMP
24	
25	
26	
27	
28	

PC 22

RAM	
0	8
1	9
2	72
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	24576
22	0
23	0
24	0
25	0
26	0
27	0
28	0

A 22

- So by giving 2 numbers as 8 and 9 the result obtained is 72 by multiplying them it can seen in that pic

QUESTION-3

Write an assembly language program to execute the following.
RAM[5]= x, where x is some value.
Apply branching if
RAM[5]>0, RAM[6]=1 else RAM[6]=0

VIKHYAT BANSAL [CB.EN.U4AIE.21076]

Hack Assembly Language

CODE_ASSIGNMENT_2_Q3 - Notepad

File Edit View

```
@R5
D = M          // D = RAM[5]

@8
D;JGT          // If R5>0 goto 8

@R6
M=0            // RAM[6]=0
@10
0;JMP          // goto end

@R6
M=1            // R6=1

@10
0;JMP
```

Working of Hack Assembly Language

@R5
D = M
Our hack assembly language starts as shown above which takes input given to RAM[5] and store that input in D-Register.

@8
D;JGT
Now, a conditional jump statement is used which tells us to jump to 8th line of code if RAM[5] > 0 .

@R6
M = 0
@10
0;JMP

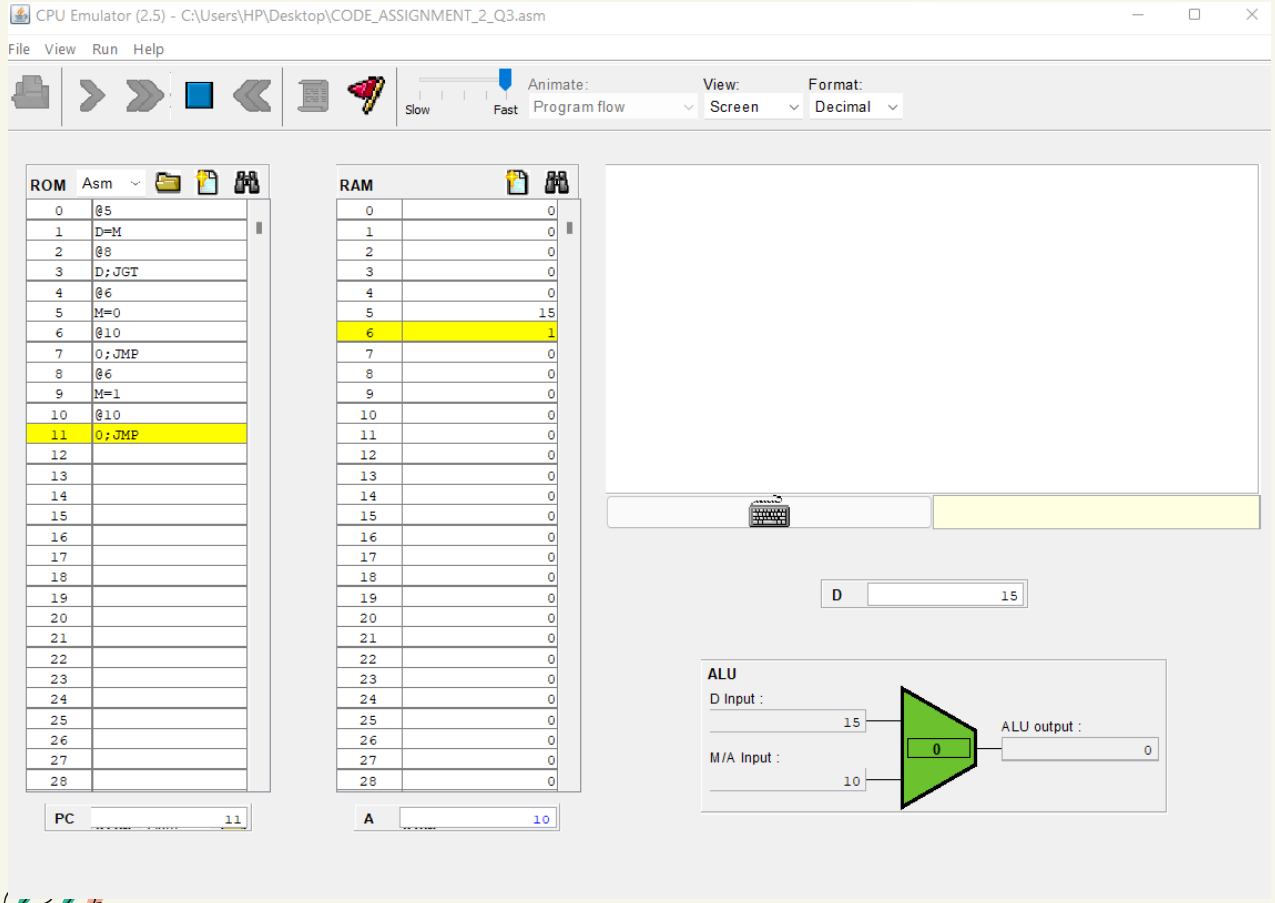
Code will come to an else condition where RAM[6] = 0 if any condition other than RAM[5]>0 follows and then the code will jump to end.

@R6
M=1

This is our 8th line of code where RAM[6] = 1. If condition RAM[5] > 0 is true.

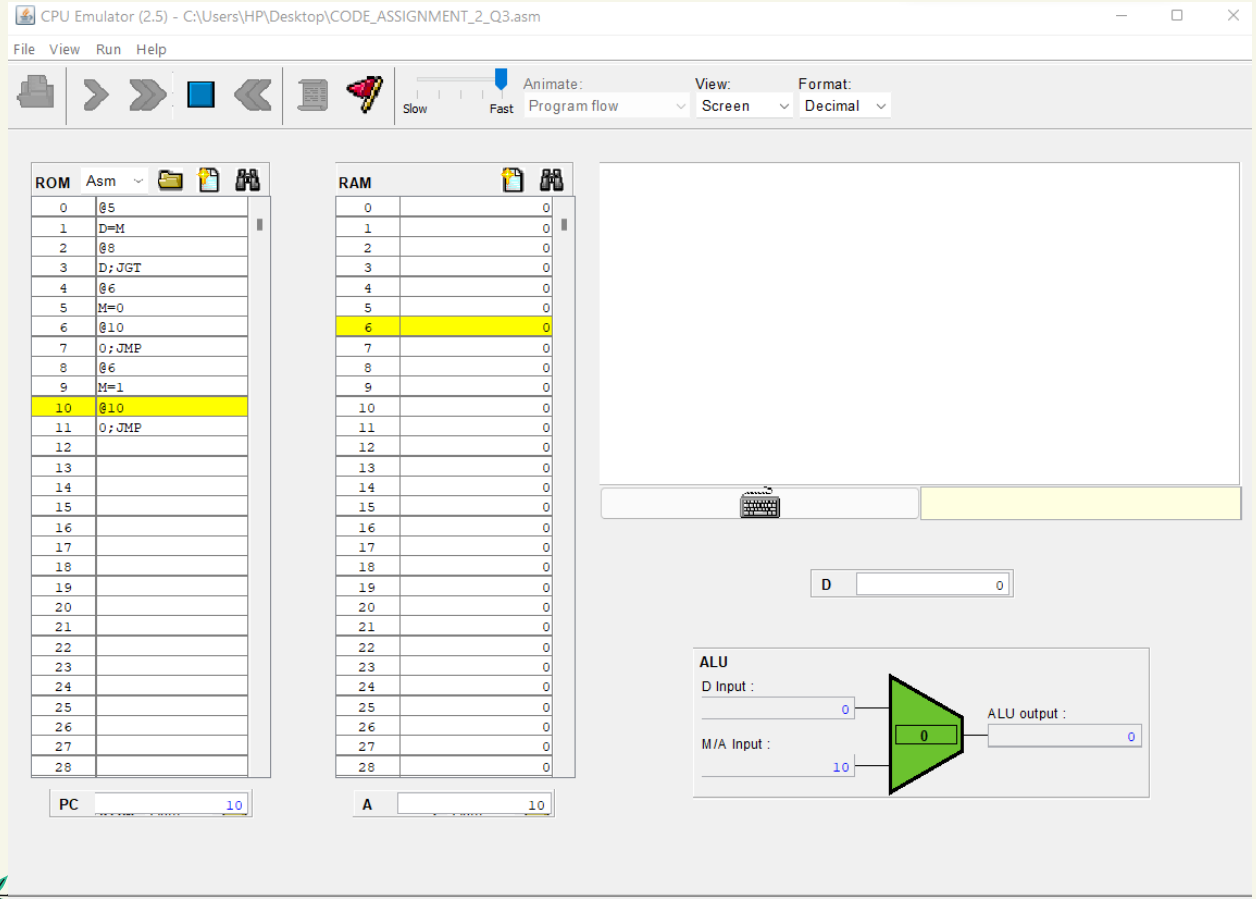
@10
0;JMP
Finally the code comes to an end.

Output - I



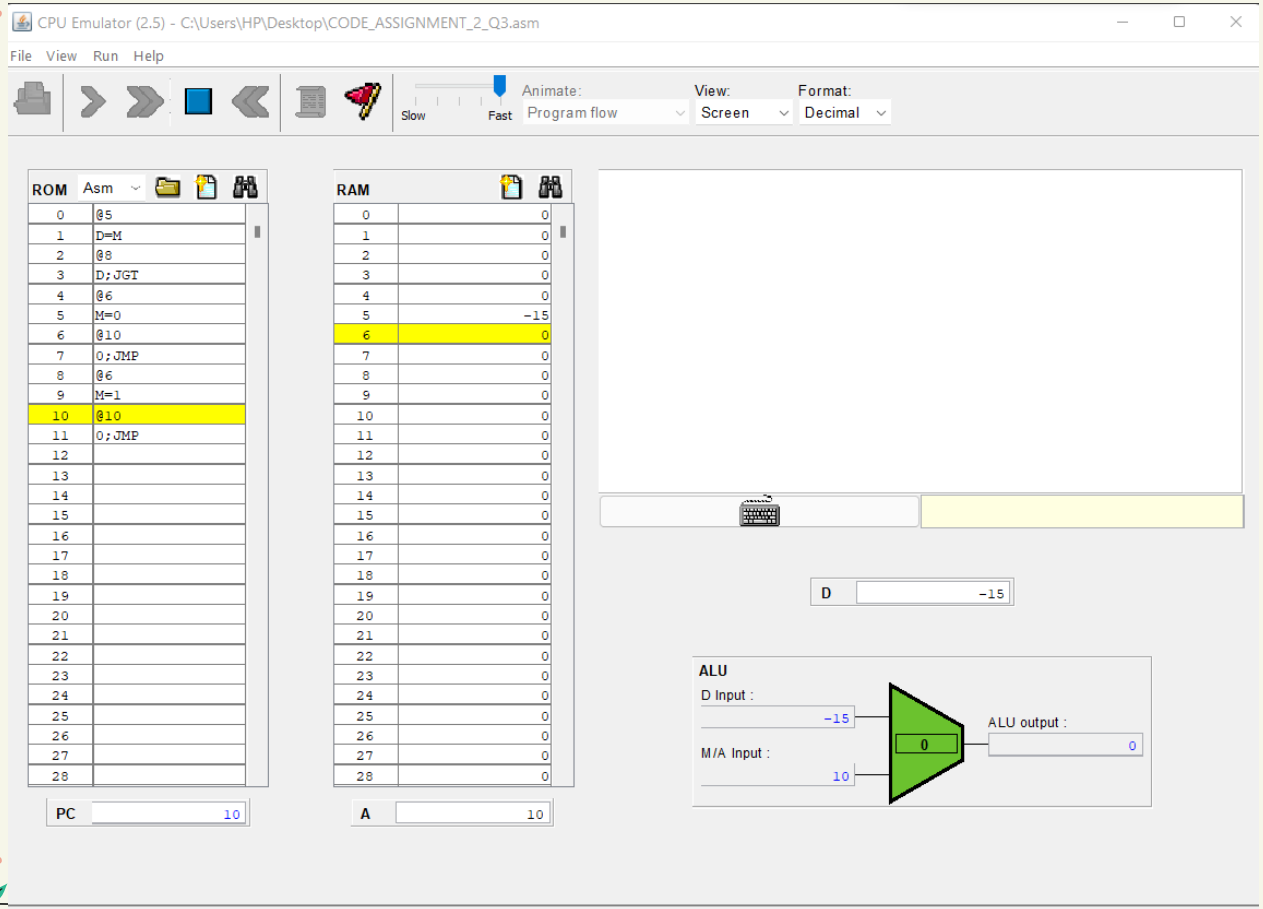
Input: RAM[5] = 15
Output: RAM[6] = 1

Output - II



Input: RAM[5] = 0
Output: RAM[6] = 0

Output - III



Input: RAM[5] = -15
Output: RAM[6] = 0

QUESTION-4

Write a hack assembly language program to swap two values using a temporary variable.

VIKHYAT BANSAL [CB.EN.U4AIE.21076]

HACK Assembly Language

CODE_ASSIGNMENT_2_Q4 - Notepad

File Edit View

```
// Swap.asm
// flips the values of
// RAM[0] and RAM[1]
// logic
// temp = R1
// R1 = R0
// R0 = temp

 @R1
 D=M
 @temp
 M=D        // temp = R1

 @R0
 D=M
 @R1
 M=D        // R1=R0

 @temp
 D=M
 @R0
 M=D       // R0 = temp

 @temp
 M=0

(END)
 @END
 0;JMP
```

Working of HACK Assembly Language

@R1
D = M
@temp (It is a temporary variable whose value may vary according to code and input.)
M = D

Symbol R1 is used to denote RAM[1] which takes input from user (when the code is executed) and store it in D-Register.
A temp variable is used to store the input from RAM[1] to location RAM[16].

@R0
D = M
@R1
M = D

Here symbol R0 is used to denote RAM[0] which takes second input from user (when the code is executed) and store it in D-Register and then symbol R1 is called to take the value in RAM[0] and store it in RAM[1] which creates RAM[0] = RAM[1].

Working of HACK Assembly Language

@temp
D = M
@R0
M = D

Now, the value stored in RAM[16] earlier from RAM[1] is now stored in RAM[0] as a value.

@temp
M = 0

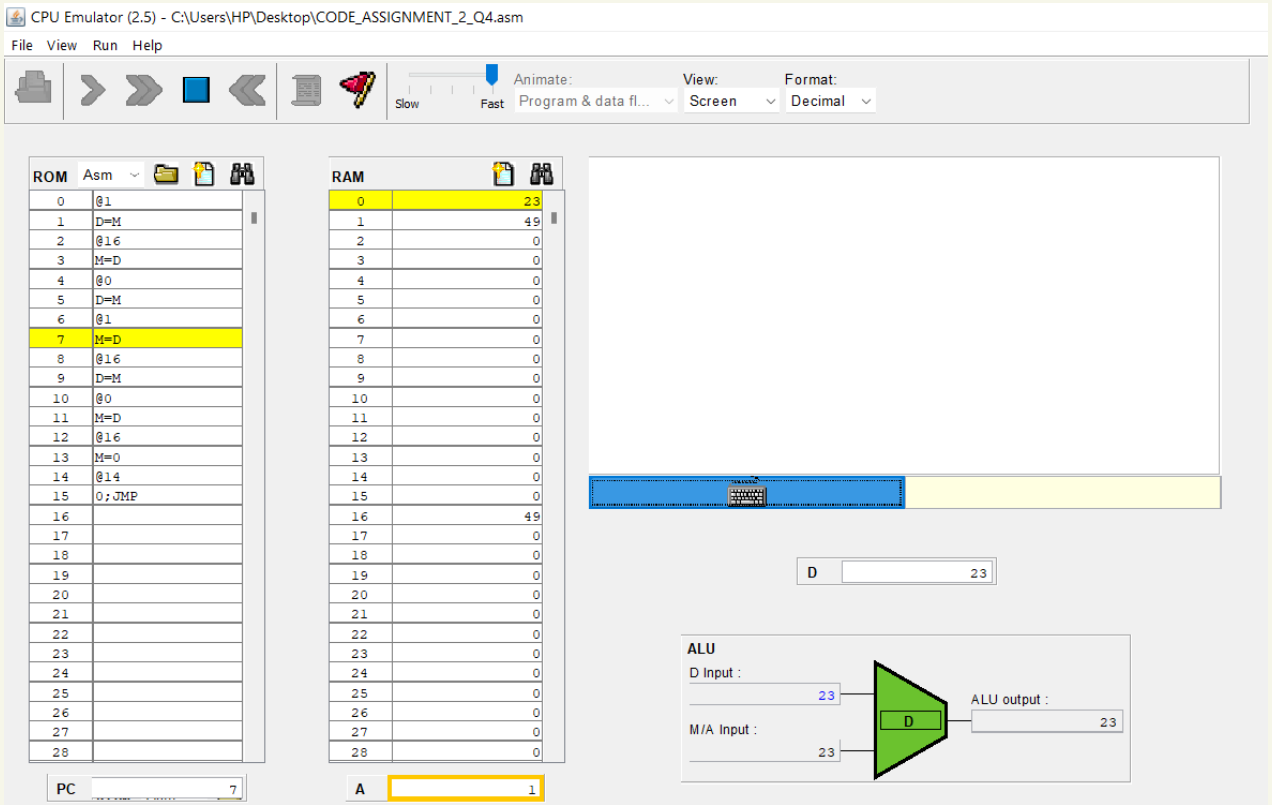
Temp variable is changed to zero i.e., RAM[16] = 0. **It is not necessary to change temp variable to 0.**

(END)
@end
0;JMP

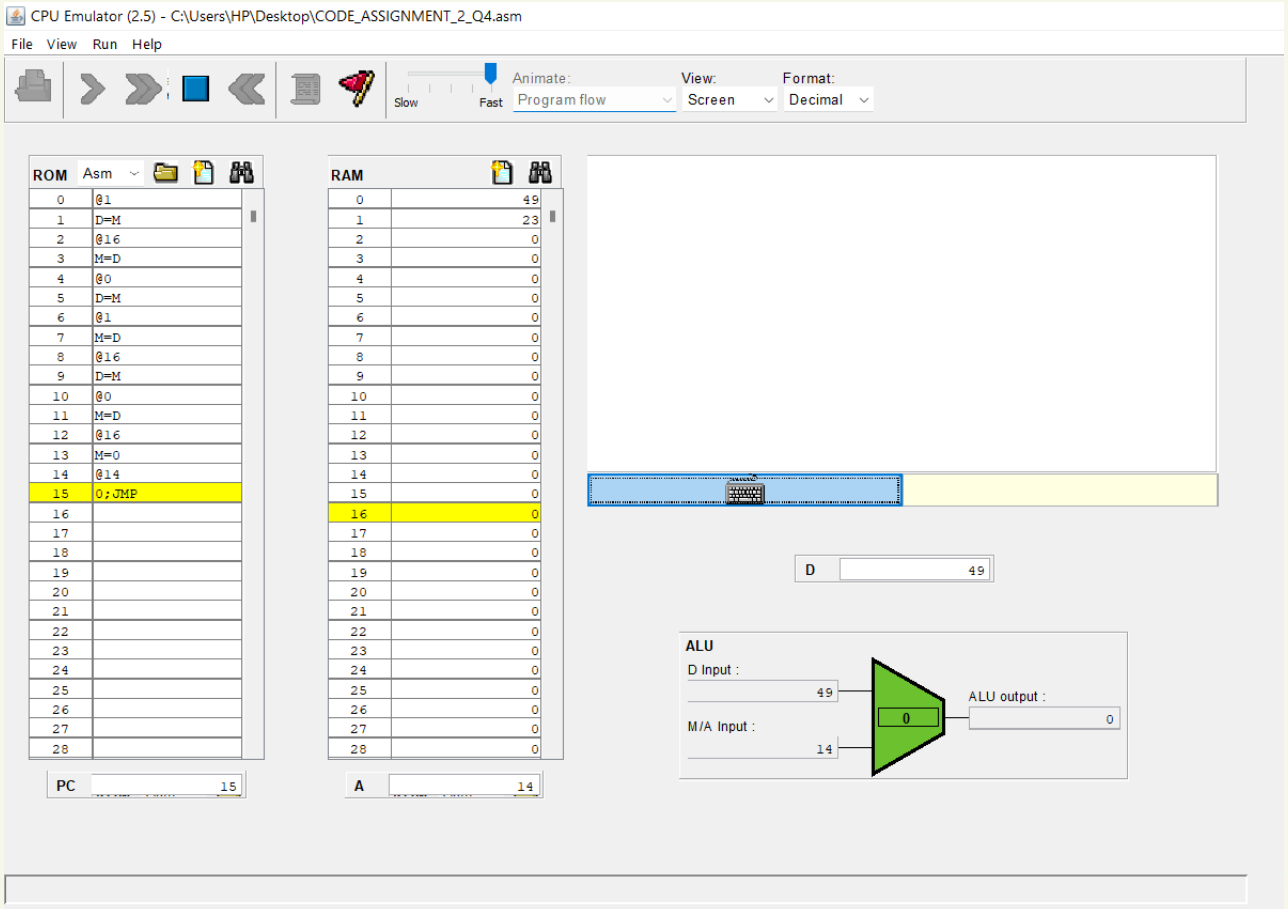
Program ends here with an unconditional jump and hence, our values are swapped.

OUTPUT (INPROGRESS)

RAM[0] = 23
RAM[1] = 49



OUTPUT (COMPLETE)



QUESTION-5

Write and execute a hack assembly language program to implement the sum of 50 numbers starting from 10. (Eg: 10+11+12+..).

M.PRASANNA TEJA [CB.EN.U4AIE.21035]

Hack assembly language

```
// Put your code here.

@50
D=A
@n
M=D // n = 50

@10
D=A
@i
M=D // i = 10

@sum
M=0 // sum = 0

(LOOP)
    @i
    D=M
    @n
    D=D-M

    @END
    D;JGT // if i > n goto STOP

    @i
    D=M
    @sum
    M=M+D       // sum = sum + i
    @i
    M=M+1 // i = i + 1


    @LOOP
    0;JMP


(END)
   @END
   0;JMP
```

ANALYSIS

QUESTION ANALYSIS:

It is basically a straightforward question which asks to execute the hack assembly language of sum from 10 to 50

ABSTRACT:

We have to make some variable symbols to define and to execute and have to use loop for the addition so that we can compute 10-50.

SYMBOLS USED:

I used 2 different symbols, but I can use all the 3 symbols, but I didn't use the pre-defined symbol because it is not mandatory.

I used 2 label symbols and 3 variable symbols in total

LABEL SYMBOLS : (LOOP) AND (END)

VARIABLE SYMBOLS : @n , @i , @sum

EXPLAINATION

The above written code implements the sum of 50 numbers starting with 10 (10+11+……+50)

- I will try to decode in and explain the code and for that I divided the code into 6 parts

```
@50
D = A
@n
M = D // n = 50
```

- First, we will initialize a variable n with 50. So, @50 is used which is A register and we have to the input in D register. @n is a first variable symbol is used it helps to restore the D register in M register.

```
@10
D = A
@i
M = D // i = 10
```

- Next, we will initialize a variable i with 10. It was written in the similar way like in the previous part @i is another variable I took to store the value of 10 in the M register.

```
@sum
M=0 // sum = 0
```

- Next, we will initialize the variable sum with 0 to store the final value of addition of given numbers.

```
(LOOP)
@i
D = M
@n
D = D – M

@END
D ; JGT // if i > n goto stop

@i
 D=M
 @sum
 M=M+D     // sum = sum + i
  @i
  M=M+1 // i = i + 1
```

LOOP EXPLAINATION:

1. Now, we will start a loop and write condition as if i is greater than n then goto end or execute next instruction so that I value will not exceed 50.

2. Next. we will add the value in sum with value of i and store it in sum which we created before.

3. The value of i is incremented by 1 in each iteration.

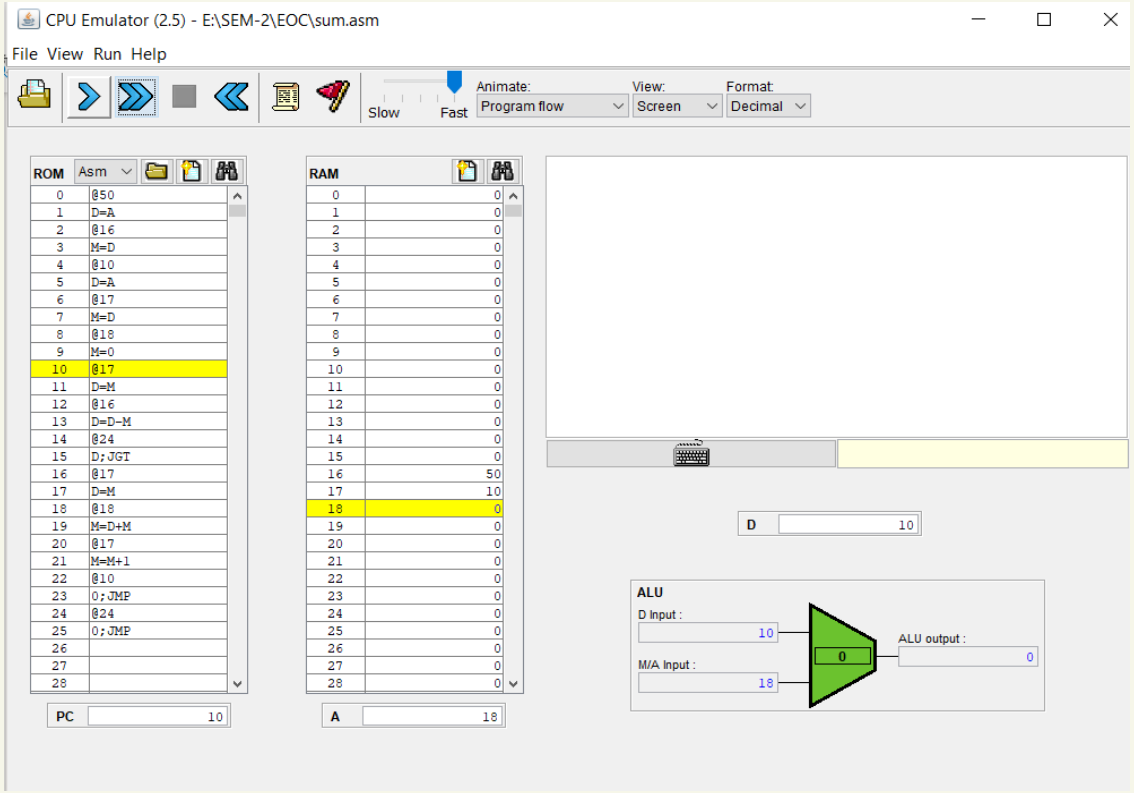
```
@LOOP
 0;JMP
```

- This helps us to make loop as many time as we need as we have to add 10 to 50, we have written this code it will jump again to the start and goes on executing until i becomes 50 here 0 indicates no condition required

```
(END)
  @END
  0;JMP
```

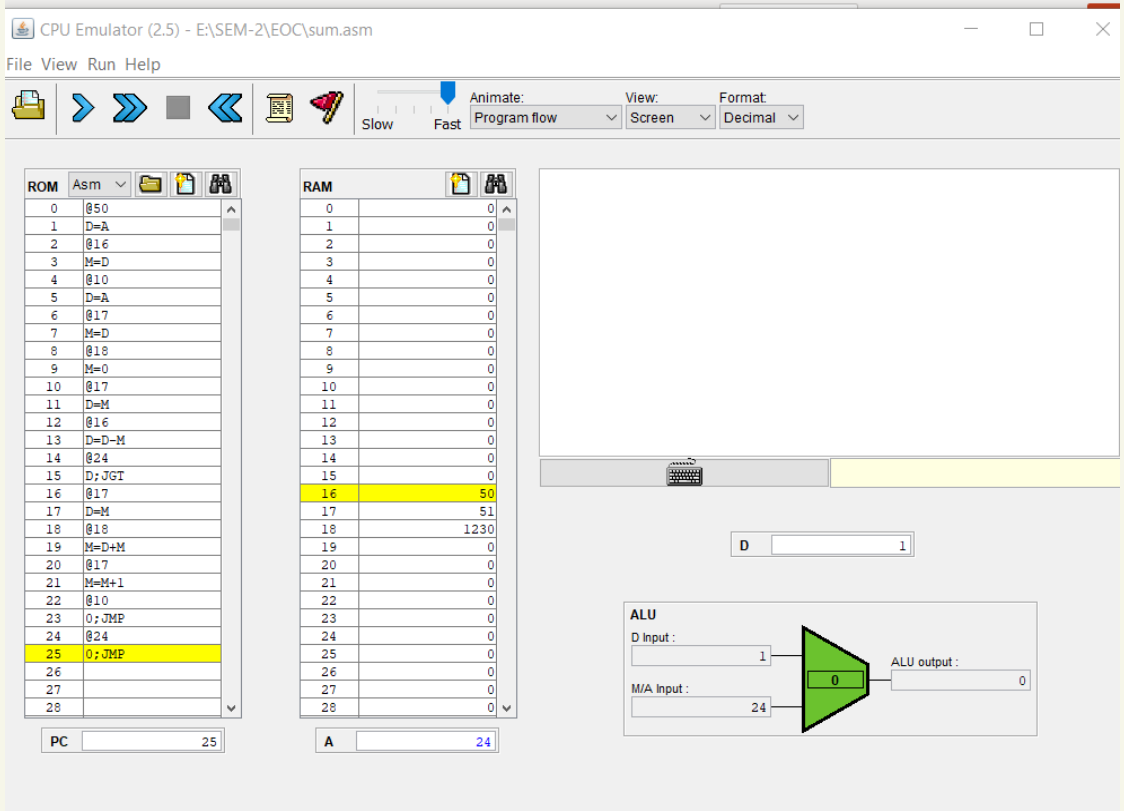
- Program ends here with an unconditional jump and hence, our values are swapped.

OUTPUT BEFORE START OF LOAP



We can see at RAM 16 the value is 50 as we given at the start and at RAM 17 it is 10 as we given the value I=10 at the start of loop because we need sum from 10 and at RAM 18 as it is 0 as we initially took sum as 0.

FINAL OUTPUT



Clearly at RAM 18 we can see the result we want when we add 10 to 50 and at RAM 17, we can see the I value and because I value is above 50 loop ends and RAM 16 is same as 50

THANK YOU