EOC ASSIGNMENT-3

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

Write a hack assembly language program to create an array of size 10 with values 20

Pseudo Code

```
arr = 100
n = 10
i = 0

LOOP:
   if i = n  goto END      // when i reaches final iteration that is 10 it will stop the loop
   RAM[arr+i] = 20         // each RAM address as an input is given as 20 as mentioned in Q
   i = i+1                 // loop is created, iteration will increase by one from 0 to 9  and at iteration 10, loop will end
   goto LOOP

END:
   Program Ends
```

Hack Assembly Language

```
// for (i=0; i<n; i++) {
//   arr[i] = 20
// }
// arr=100 and n=10

                    //array size = n = 10
@10
D=A
@n                  //variable name provided
M=D

@i                  //i = 0 as iteration is starting from 0
M=1                 //Variable name provided

@100                //arr = 100
D=A
@arr                //Variable name provided
M=D

(loop)
                    //if (i==n) goto END
@i
D=M
@n
D=D-M
@end
D;JGT
                    //RAM[arr+i] = 20
@arr
D=M
@i
D=D+M
@location
M=D
```

Hack Assembly Language(Contd.)

```
@arr
D=M
@i
D=D+M
@location
M=D


@20                 //M = 20
D=A
@location
A=M
M=D


@i                  //i++ Value of iteration increases by 1
M=M+1

@loop               //Referring to label LOOP
0;JMP               // goto LOOP

(end)               //Declaring label END
@end                // End of the Program
0;JMP               // Infinite loop
```

Working of language in CPU Emulator

We will allot three variables each with different functions,

First variable is array size(n) = 10,

Second variable is iteration (i) = 1 (it is used to keep a count that how many times a loop is repeated)

Third variable is array location. It tells us the RAM[location] where the input value given to an array will be stored.

After allotting variables a label of LOOP is declared, where a condition is being created, here it is checked that whether iteration is equal to size of array (i==n) ,if yes then LOOP will end, else LOOP will keep going.

Working of language in CPU Emulator

Now comes the part that tells the CPU Emulator to enter which values at what location.

@arr

D=M means that D=RAM[@arr]

(in register D value of @arr will be saved)

@i

D=D+M (for first time it will be D = 100 + 1)

(location gets changed every time loop repeats itself)

@location

M=D

It helps in determining the location where the value will be kept

Working of language in CPU Emulator

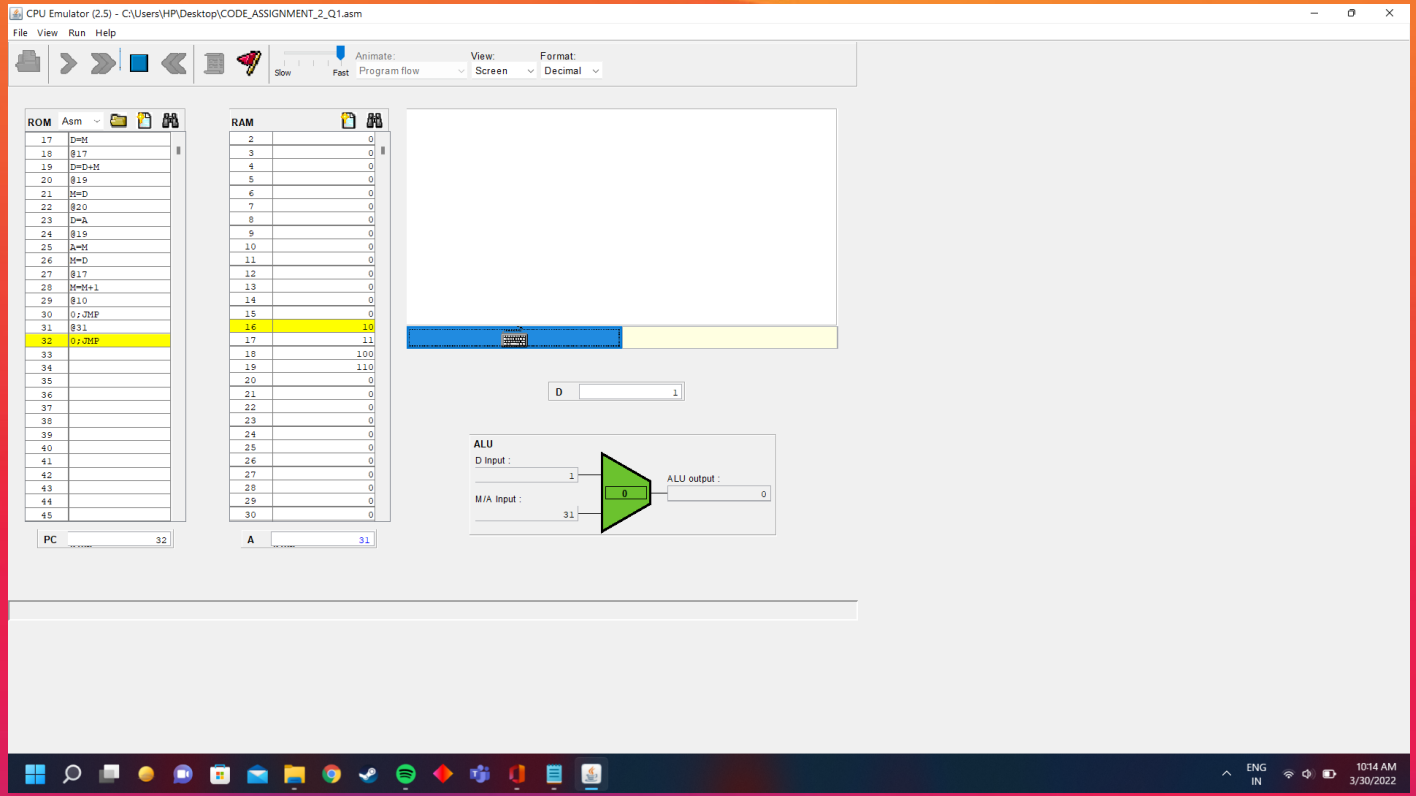
@20
D=A (D register is 20)
@location
A=M
M=D
With the help of lines written above the value is stored at the desired location.

@i
M=M+1
Here, the iteration increases by 1 until the LOOP ends

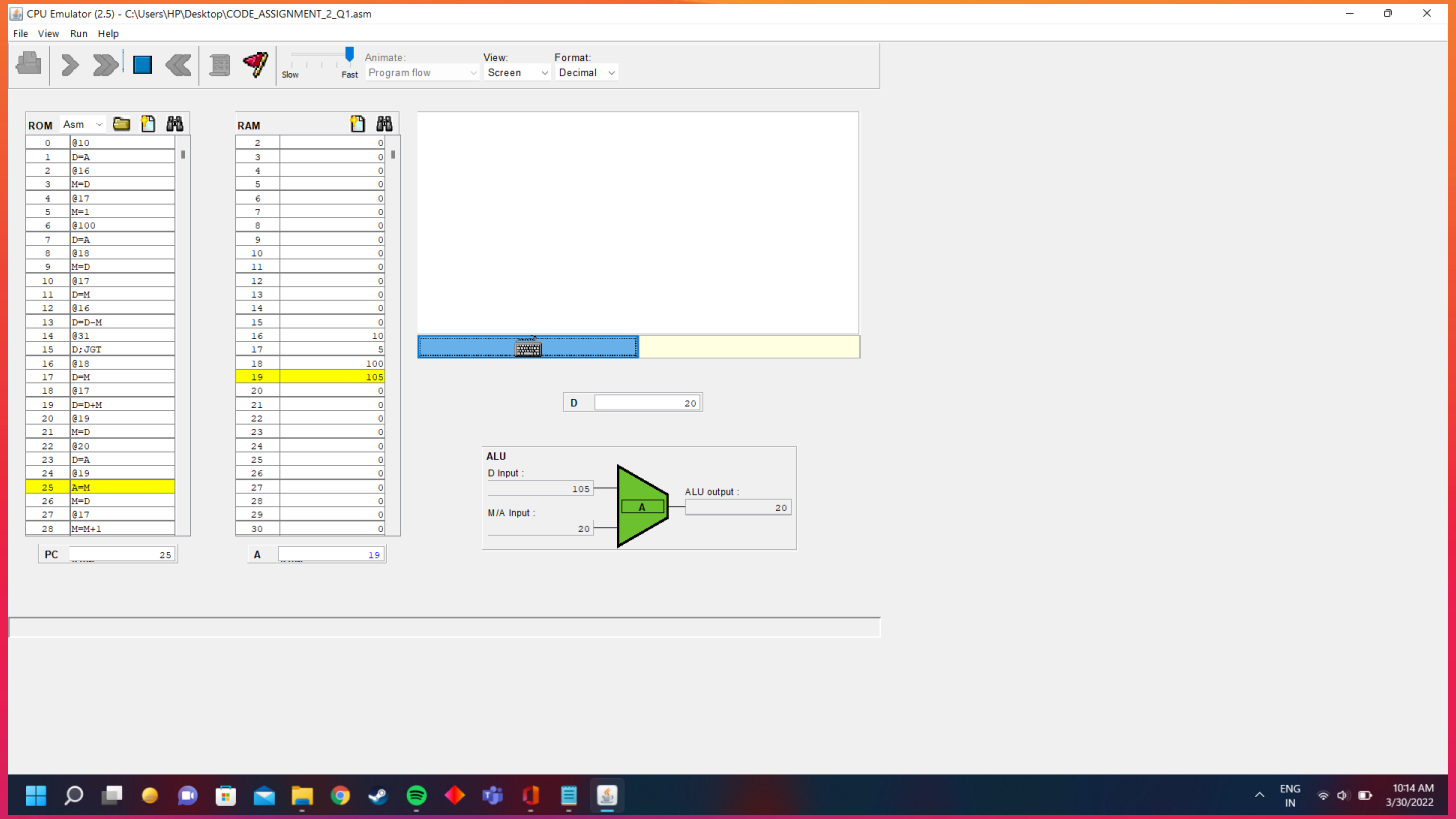
@loop
0;JMP
LOOP will go to the declaration point to check whether i== n or not, if not, then it will execute the lines again.

(end)
@end
0;JMP
Program ends here

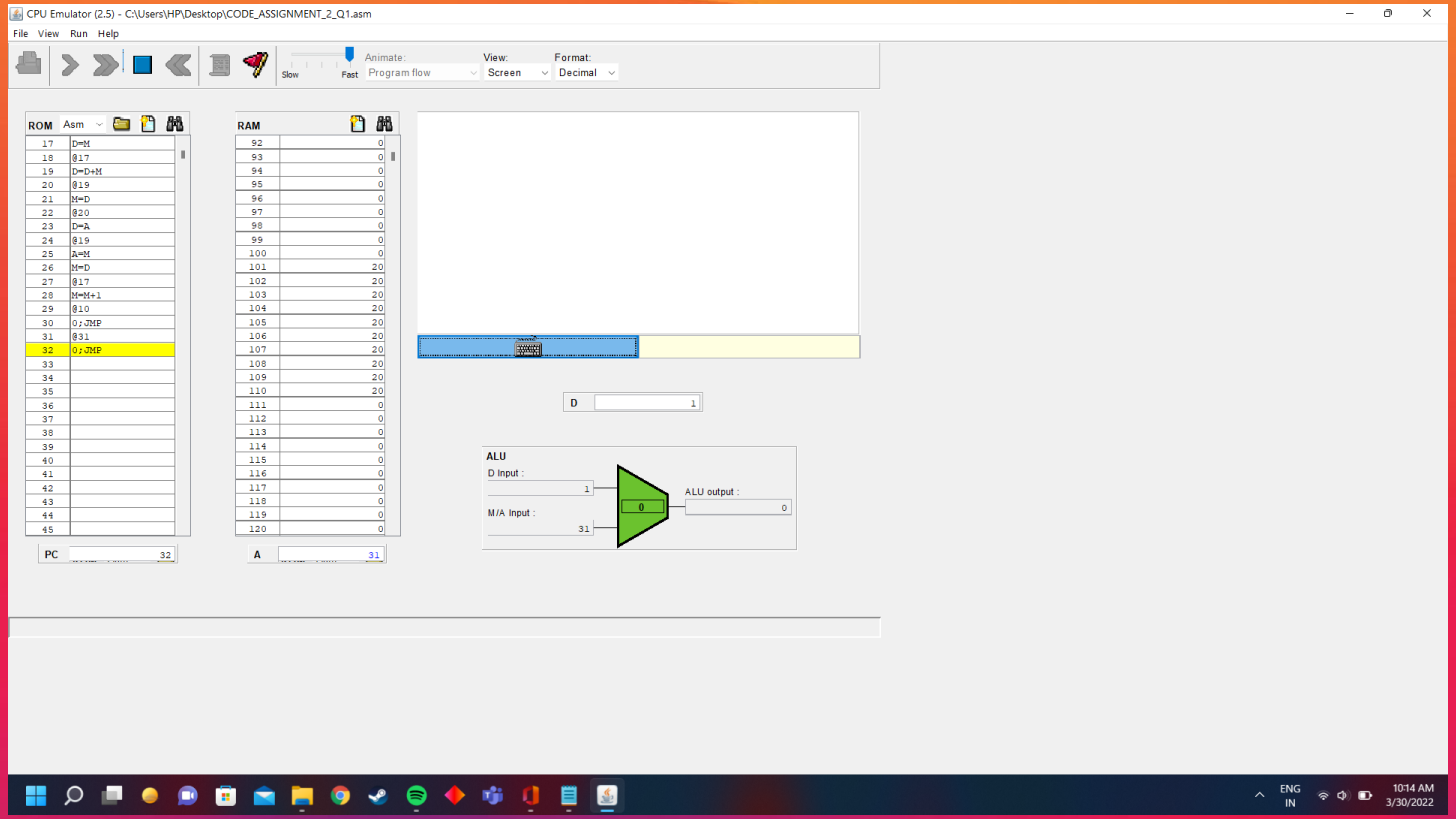
OUTPUT


CPU Emulator (2.5) - C:\Users\HP\Desktop\CODE_ASSIGNMENT_2_Q1.asm
File View Run Help
Slow
Fast
Animate:
Program flow
View:
Screen
Format:
Decimal
ROM
Asm
17 D=M
18 @17
19 D=D+M
20 @19
21 M=D
22 @20
23 D=A
24 @19
25 A=M
26 M=D
27 @17
28 M=M+1
29 @10
30 0;JMP
31 @31
32 0;JMP
33
34
35
36
37
38
39
40
41
42
43
44
45
PC 32
RAM
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 10
17 11
18 100
19 110
20 0
21 0
22 0
23 0
24 0
25 0
26 0
27 0
28 0
29 0
30 0
A 31
D 1
ALU
D Input : 1
M/A Input : 31
0
ALU output : 0
ENG
IN
10:14 AM
3/30/2022

OUTPUT


CPU Emulator (2.5) - C:\Users\HP\Desktop\CODE_ASSIGNMENT_2_Q1.asm
File View Run Help
Animate:
Slow
Fast
Program flow
View:
Screen
Format:
Decimal
ROM
Asm
0 @10
1 D=A
2 @16
3 M=D
4 @17
5 M=1
6 @100
7 D=A
8 @18
9 M=D
10 @17
11 D=M
12 @16
13 D=D-M
14 @31
15 D;JGT
16 @18
17 D=M
18 @17
19 D=D+M
20 @19
21 M=D
22 @20
23 D=A
24 @19
25 A=M
26 M=D
27 @17
28 M=M+1
PC 25
RAM
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 10
17 5
18 100
19 105
20 0
21 0
22 0
23 0
24 0
25 0
26 0
27 0
28 0
29 0
30 0
A 19
D 20
ALU
D Input :
105
M/A Input :
20
A
ALU output :
20
ENG
IN
10:14 AM
3/30/2022

OUTPUT


CPU Emulator (2.5) - C:\Users\HP\Desktop\CODE_ASSIGNMENT_2_Q1.asm
File View Run Help
Animate:
Slow
Fast
Program flow
View:
Screen
Format:
Decimal
ROM
Asm
17 D=M
18 @17
19 D=D+M
20 @19
21 M=D
22 @20
23 D=A
24 @19
25 A=M
26 M=D
27 @17
28 M=M+1
29 @10
30 0;JMP
31 @31
32 0;JMP
33
34
35
36
37
38
39
40
41
42
43
44
45
PC 32
RAM
92 0
93 0
94 0
95 0
96 0
97 0
98 0
99 0
100 0
101 20
102 20
103 20
104 20
105 20
106 20
107 20
108 20
109 20
110 20
111 0
112 0
113 0
114 0
115 0
116 0
117 0
118 0
119 0
120 0
A 31
D 1
ALU
D Input :
1
M/A Input :
31
0
ALU output :
0
ENG
IN
10:14 AM
3/30/2022

QUESTION-2

Create an array of size 10 with entries 1,2,3,.....10 and find the sum and average of array e.

HACK CODE

```
// We are writing a code to print array of length 10 starting
//from R0

@count  //count variable is used for counting and stopping the loop
@size  // size of the array

(LOOP)
@count
M=M+1  // Count starts from 0 and increases till 10
D=M
A=M  //Address will be same as the count value
M=D  // The register should store the same value
@sum  // Sum function
D=D+M
M=D  // Sum will get updated every

@count
D=M
@size
D=M-D  // Halting condition D
@LOOP
D;JGT  // Compares to the previous D
```

CONTINUTION

```
@sum
D=M  // D= Final sum

// Average means sum/no of elements, for doing division we did re substracting process.

(SUB)
@10
D=D-A // We have D value from sum register and this will subtract that value until the loop ends.
@END
D;JLT // If the subtracted value is less than 0, then loop will end.
@avg
M=D // This will store the average value in avg register.
@SUB
D;JGT

@END
(END) // End loop.
0;JMP
```

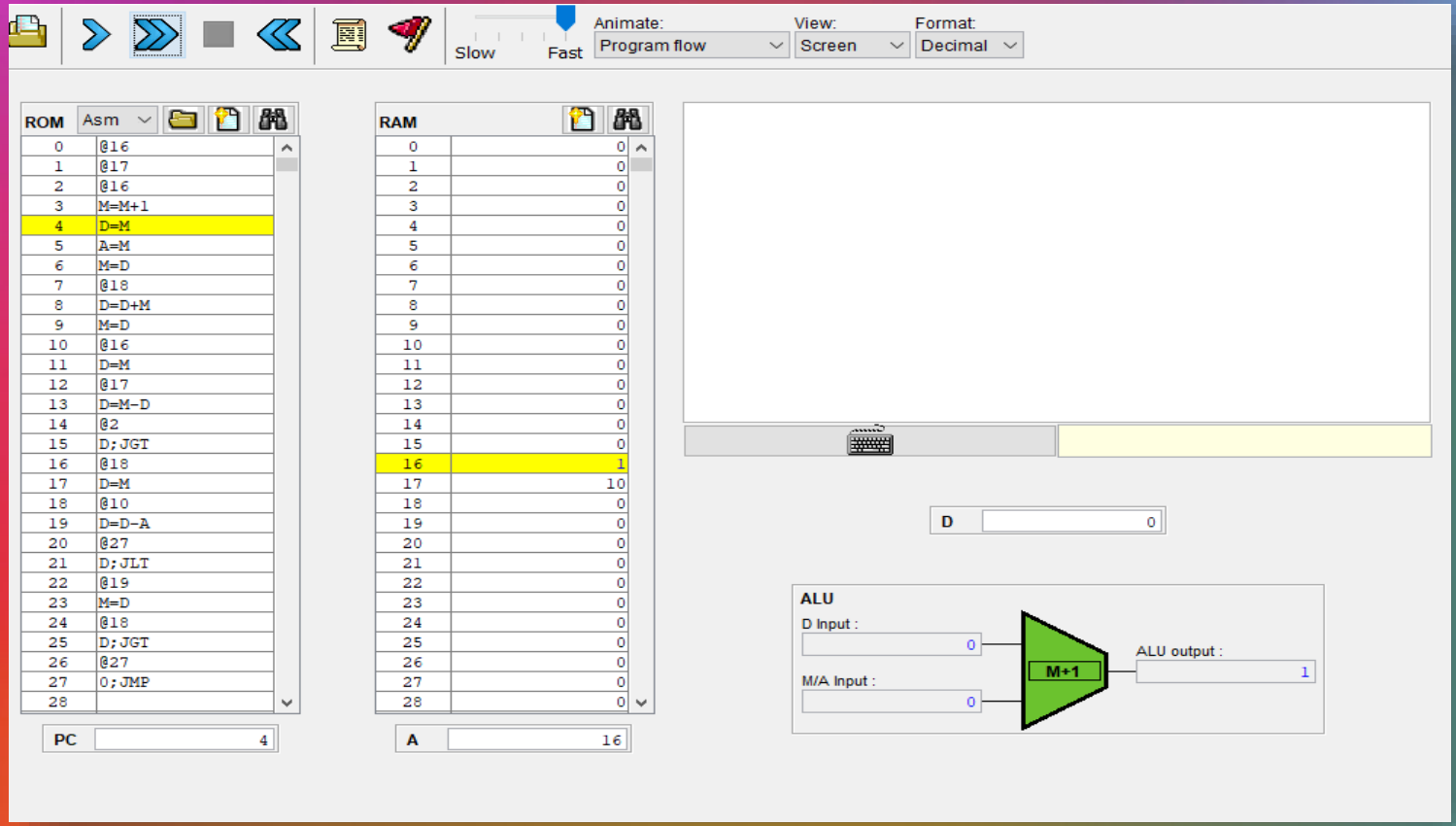
EXPLANATION

- The above code creates an array of 1 to 10 from RAM[0] and give the summation and average of all array elements.
- First, we will take a variable count which is used for counting and stopping the loop.
- Next, we declare another variable size to mention the size of array we required.
- Now , we start a loop and increment the count value by 1 and store this value in the memory location of value in count.
- This loop will continue to execute until the difference between the value in the variables count and size is greater than zero. And we will get the values 1 to 10 in the RAM location 1 to 10.

CONTINUTION

- ***After that we will find the average of these array elements, to calculate average the formula is sum of total number of elements divided by total number of elements.***
- ***As we do not have any operation to divide two number, we can do it by subtraction.***
- ***In this we subtract 10 with the value in the variable sum and continue this process until we get the last positive number and this the average value of array elements.***
- ***We will store this value in a variable avg and the program by writing an infinite loop.***

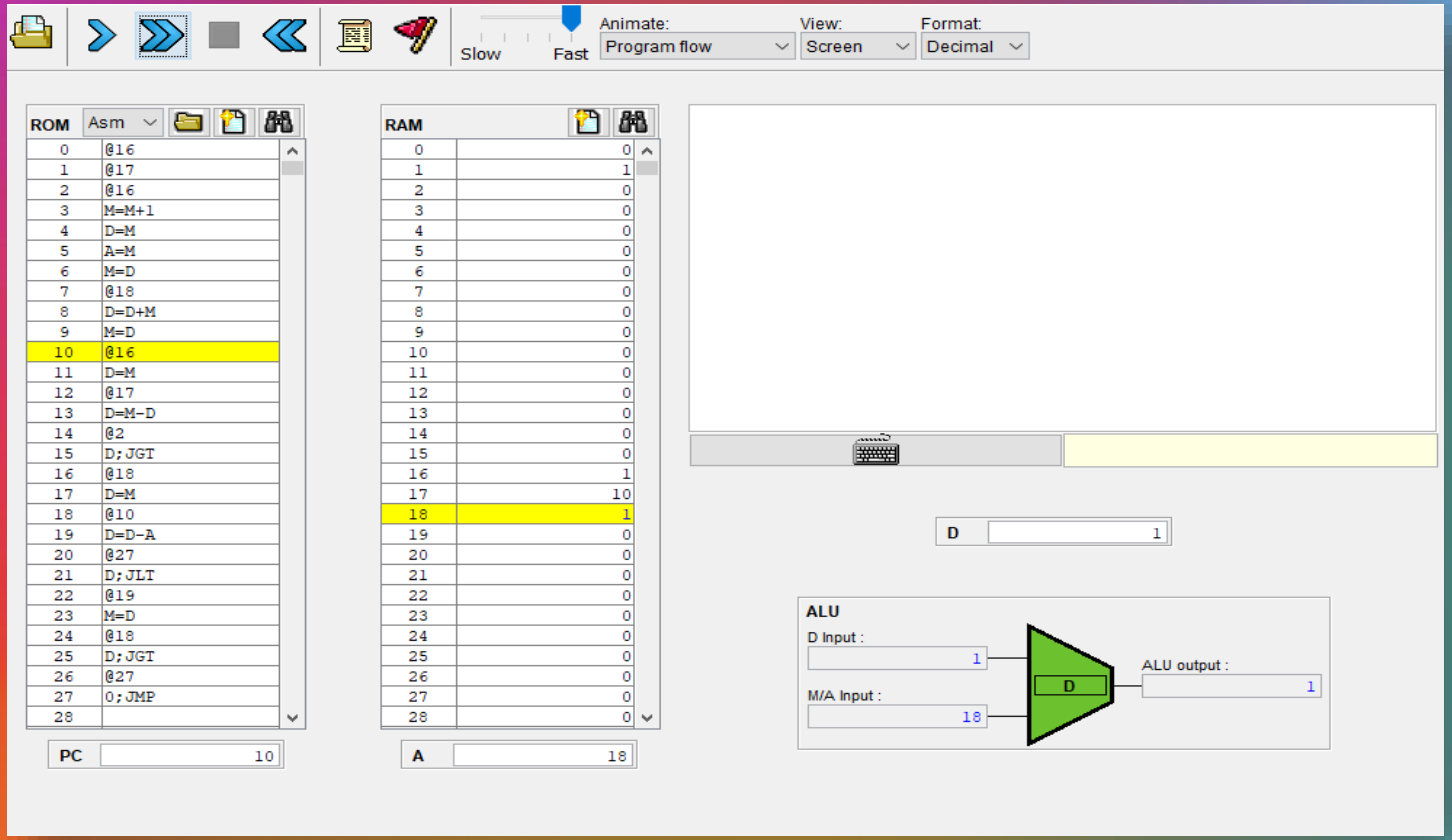
FINAL OUTPUT



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

Before start of loop

FINAL OUTPUT

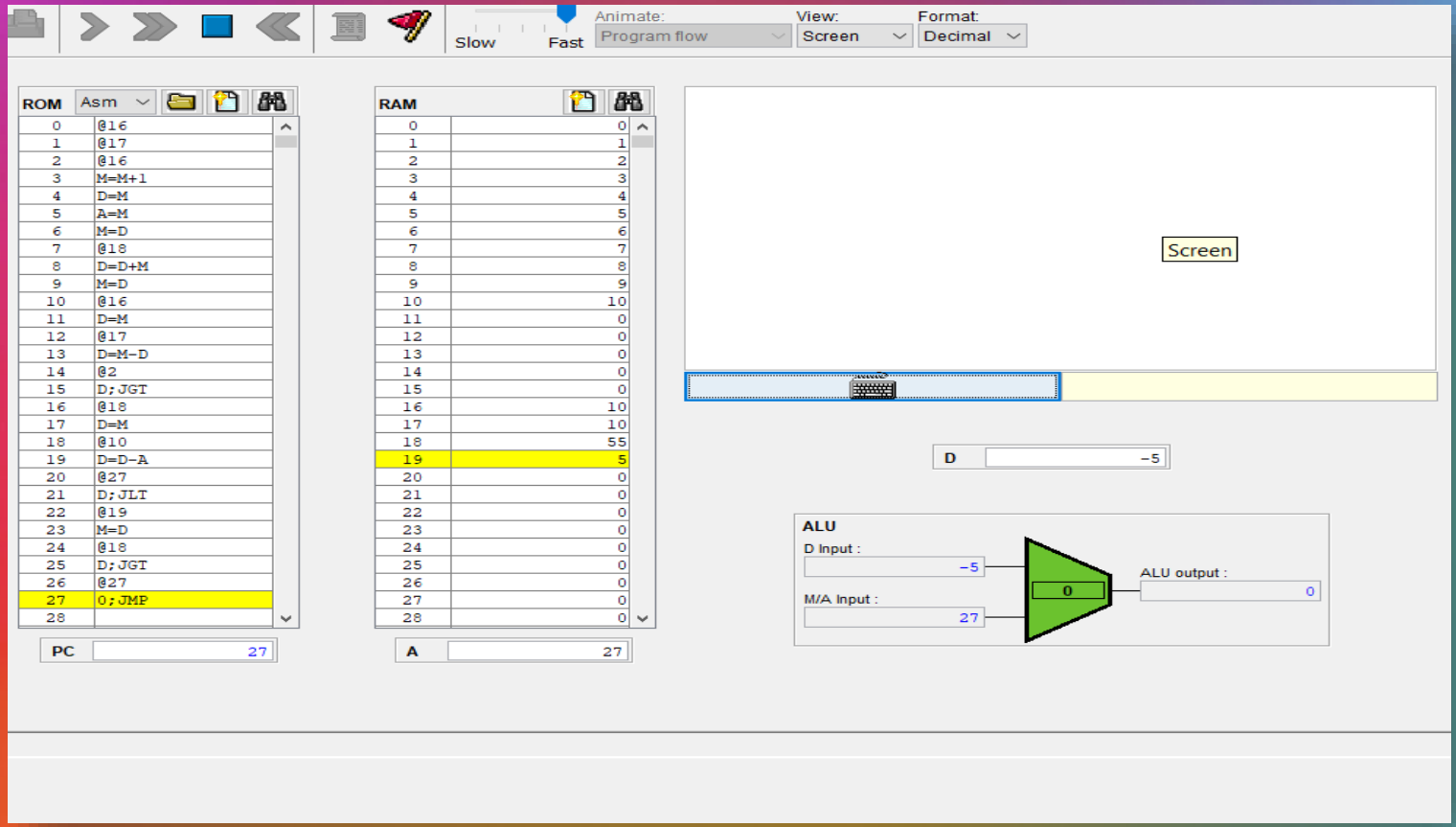


ROM	
0	@16
1	@17
2	@16
3	M=M+1
4	D=M
5	A=M
6	M=D
7	@18
8	D=D+M
9	M=D
10	@16
11	D=M
12	@17
13	D=M-D
14	@2
15	D;JGT
16	@18
17	D=M
18	@10
19	D=D-A
20	@27
21	D;JLT
22	@19
23	M=D
24	@18
25	D;JGT
26	@27
27	0;JMP
28	

RAM	
0	0
1	1
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	1
17	10
18	1
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0

At the start of the sum

FINAL OUTPUT



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

	RAM
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	0
12	0
13	0
14	0
15	0
16	10
17	10
18	55
19	5
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0

END OF THE STIMULATION

QUESTION-3

Write a hack assembly language program to draw a rectangle at the upper right corner, 16 pixel wide and RAM[0] pixels.

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

Question Analysis:

- We need to write a Hack assembly language program to draw a rectangle at the upper right corner, 16 pixel wide and RAM[0] pixels.

ABSTRACT:

- We have to make some variable symbols to define and to execute and have to use loop to make a rectangle at upper right corner , 16 pixel wide and RAM[0] pixel.

SYMBOLS USED

- Loop
- End
- @address , @i,@END

CODE EXPLANATION

```
@R0
D=M
@i
M=0
```

Using D-register and representing that n = RAM[0]

```
@i
M=0
```

Mentioning that the value I is zero

```
@screen
D=A
@address
M=D
```

Mentioning that address is 16384 which is base address of the hack screen.

```
        Loop
        @i
        D=M
        @n
        D=D-M
        @end
        D;JGT
```

JGT is used when if out>0 jump
If I>n then goto end

```
@address
A=M
M=-1
```

We are using a pointer to write memory. I.e RAM[address] = -1 which is 16 pixel

```
        @i
        M=M+1
        @32
        D=A
        @address
        M=D+M
        @loop
        0;JMP
```

@32 represents the address 32 and D+M represents address + 32.
JMP is used to goto loop.

```
@END
0;JMP
```

@end represents the end of program
JMp indicates infinite loop

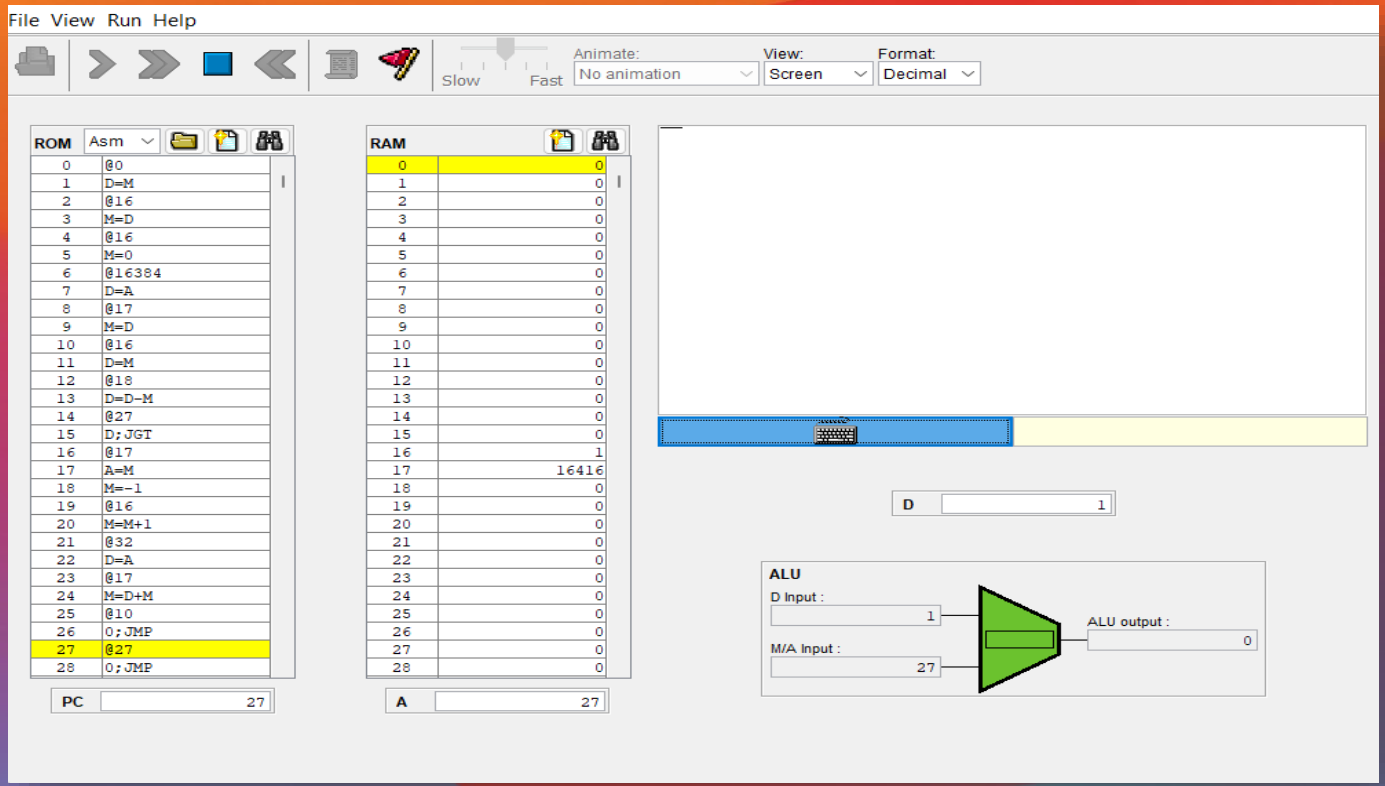
Now we are going to execute the code in the tool CPUemulator to get the output

HDL CODE

```
@R0
D = M
@i
M = D                  // n = RAM[0]

@i
M = 0                  // i = 0

@SCREEN
D = A
@address
M = D                  // address = 16384 (base address of the Hack screen)

(LOOP)
   @i
   D = M
   @n
   D = D - M
   @END
   D;JGT               // if i > n goto END
```

```
@address
A = M           // writing to memory using a pointer
M = -1          // RAM[address] = -1 (16 pixels)

@i
M = M + 1       // i = i + 1
@32
D = A
@address
M = D + M       // address = address + 32
@LOOP
0;JMP           // goto LOOP

(END)
   @END         // program's end
   0;JMP        // infinite loop
```

OUTPUT


File View Run Help
Slow
Fast
Animate:
No animation
View:
Screen
Format:
Decimal
ROM
Asm
0 @0
1 D=M
2 @16
3 M=D
4 @16
5 M=0
6 @16384
7 D=A
8 @17
9 M=D
10 @16
11 D=M
12 @18
13 D=D-M
14 @27
15 D;JGT
16 @17
17 A=M
18 M=-1
19 @16
20 M=M+1
21 @32
22 D=A
23 @17
24 M=D+M
25 @10
26 0;JMP
27 @27
28 0;JMP
PC 27
RAM
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 1
17 16416
18 0
19 0
20 0
21 0
22 0
23 0
24 0
25 0
26 0
27 0
28 0
A 27
D 1
ALU
D Input : 1
M/A Input : 27
ALU output : 0

QUESTION-4

Write a hack assembly language program to blacken the entire pixels on the screen on a ‘keypress’ by the User.

GUNNAM HIMAMSH [CB.EN.U4AIE.21010]

- Basically we have 2 types of Instructions A and C .
- A instruction is in the format of @ and C instruction is like M+D, D-M,D=A,A=D+1 and all are examples of C instruction type.
- So now the required HDL code for the given Question is as below.

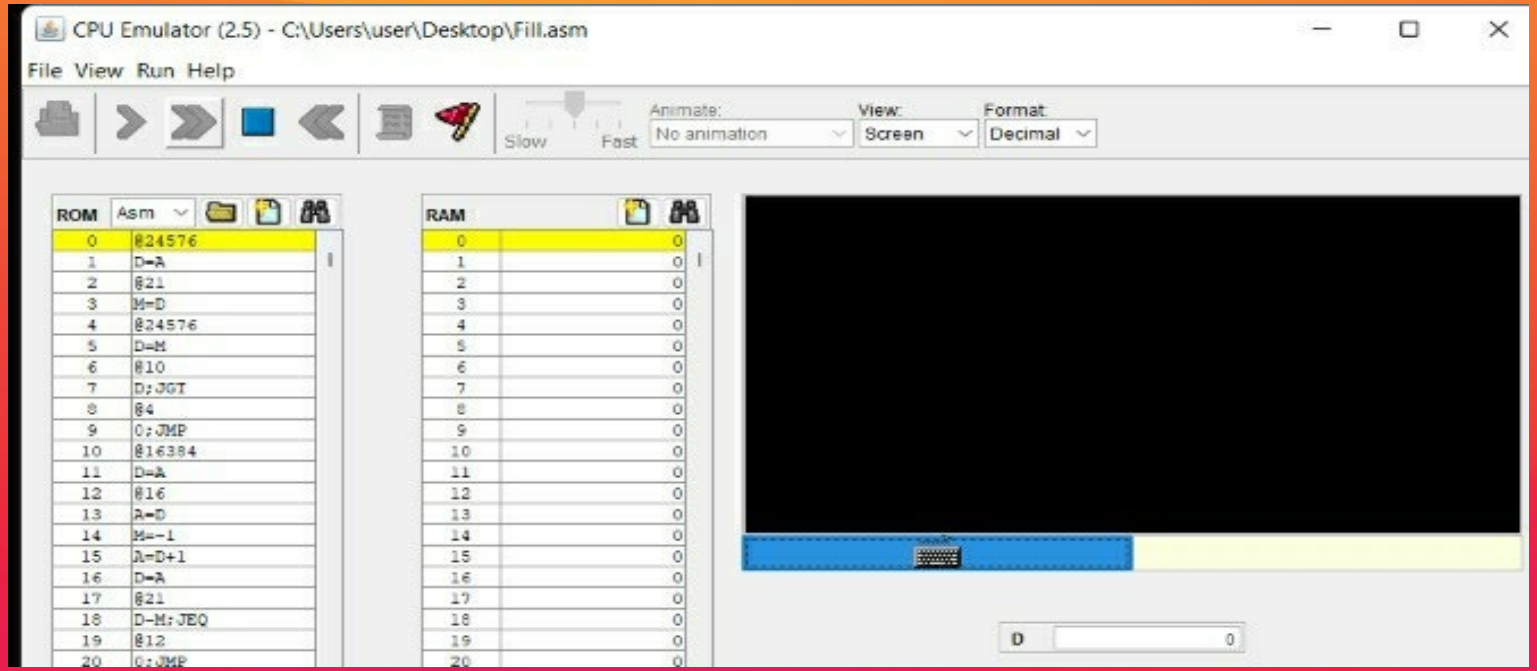
CODE

```
// Put your code here.
@24576 // Used to get 24576
D=A // Pass 24576 to D Register

@21 // Just a variable
M=D // Put the value of D into RAM[21]

(LOOP) // Start Loop here

@KBD // Base address of Keyboard Memory Map
D=M // Get the scan code and pass it to S


@10 // Variable
D;JGT // If D (scan code) is greater than 0 go to line 10

@LOOP // Loop from here
0;JMP // unconditional loop

@SCREEN // Get base address of Screen Memory Map. This is the 10th line
D=A // Store that in D Register
```

```
(MOMO) // Start Loop here
@fill // A variable -- also a pointer
A=D // Pass the value in D to A
M=-1 // Set value of M to -1 to turn on the pixels
A=D+1 // Increment A
D=A // Pass the incremented value to D

@21 // We declared this variable before. Its used to get 24576
D-M;JEQ  // Here if D-24576 is equal to 0 then jump to line 21

@MOMO // Go to (MOMO)
0;JMP // This is an unconditional loop.

@21 // Here is line 21
0;JMP // Infinite loop
```

- So, in the earlier code, we first have to gp to 24576 value, which can be achieved by writing @24576 and is an A type command. Then we must save it in D register.
- After that, choose a variable and store it into RAM, as we know M=RAM[A].
- After that start a loop we can do that by giving loop as (loop), next goto keyboard and get the scan code pass it to S.
- After that select a variable as 10 and write D;jgt as per the syntax of C instruction it indicates that if the value is greater than zero goto variable 10. After that start a loop from there and an unconditional jump.
- Next get the base address of screen memory map and store it in D registor.

- Start a loop, then address a variable, then transmit the value in D to A, then set M=-1 to turn on the pixels, then increment A, and finally return the increased value to D.
- We declared variable as 21 previously it is used to get 24576 so with M=24576 if D-24576 is equal to zero then jump to 21st line.
- Go to loop and then unconditional loop as 0; jmp.
- So finally 21st line followed by an infinite loop.
- Now we will run the above code in cpu emulator and see what we get,



This is the output obtained. We can see that entire screen is blackened.

QUESTION-5

Write a code to create a rectangle of length 50 pixel and width 16pixel at the bottom right corner.

Click to add text

CODE

```
//program to create a rectangle 16 pixel wide and of 50 pixel long on bottom right corne

@22975
D=A

@addr
M=D  //addr=22975

@0
D=M

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

@i
M=0

(loop)
@i
D=M

@n
D=D-M

@end
D;JGT  //if i>n go to end

@addr
A=M
M=-1   //RAM[addr]=11111
```

Continution

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

@32
D=A

@addr
M=D+M   //addr=addr+32

@loop
0;JMP   //go to loop

(end)
@end
0;JMP
```

Explanation

First goto address 22975 and store that using D register.

Now write M=D in that M=RAM[A] A is 22975.

Now write A value as zero and write M=D (n=ram(0))

After that @I and M=0 and next start a loop from there here D=M.

After that write @n and D=D-M and end the loop like if it is greater than zero goto end.

Next @addr and A=M next is M=-1 it is nothing but RAM[addr]=-1

Next goto @16 and M=M+1 (i=i+1)

After that give the A value as 32 and store it in D register

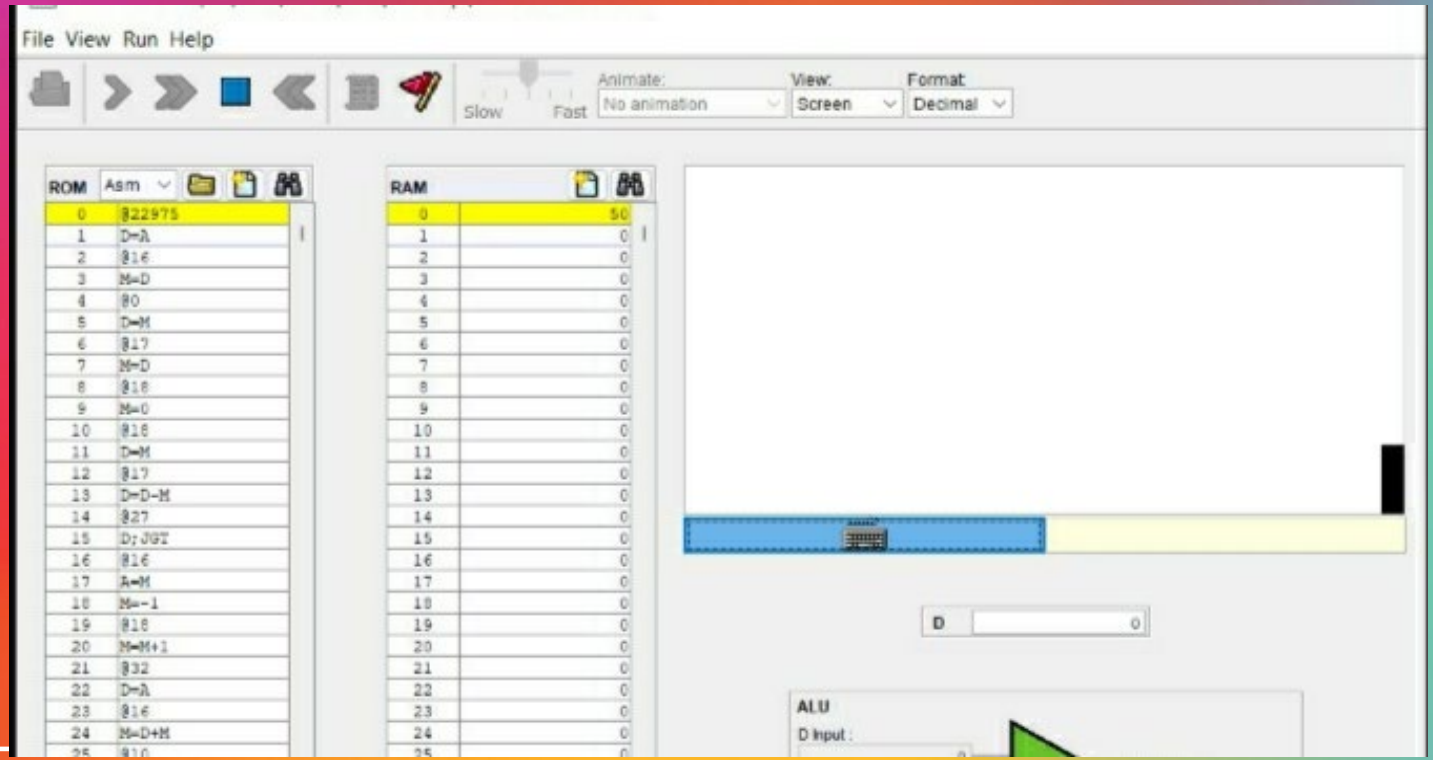
And then give A value as addr and then write M=D+M

Its simply addr=addr+32 and afterwards goto loop and write as 0;jmp.

After that goto @27 its nothing but end and write 0;jmp

Implementing this HDL code in CPU emulator we get as below

Output


File View Run Help
Animate:
No animation
View:
Screen
Format:
Decimal
Slow
Fast
ROM
Asm
RAM
D
0
ALU
D Input :

Contributions:

TEAM-7	Questions
GAJULA SRI VATSANKA	Q3
GUNNAM HIMAMSH	Q 4,5
M.PRASANNA TEJA	Q2
VIKHYAT BANSAL	Q1