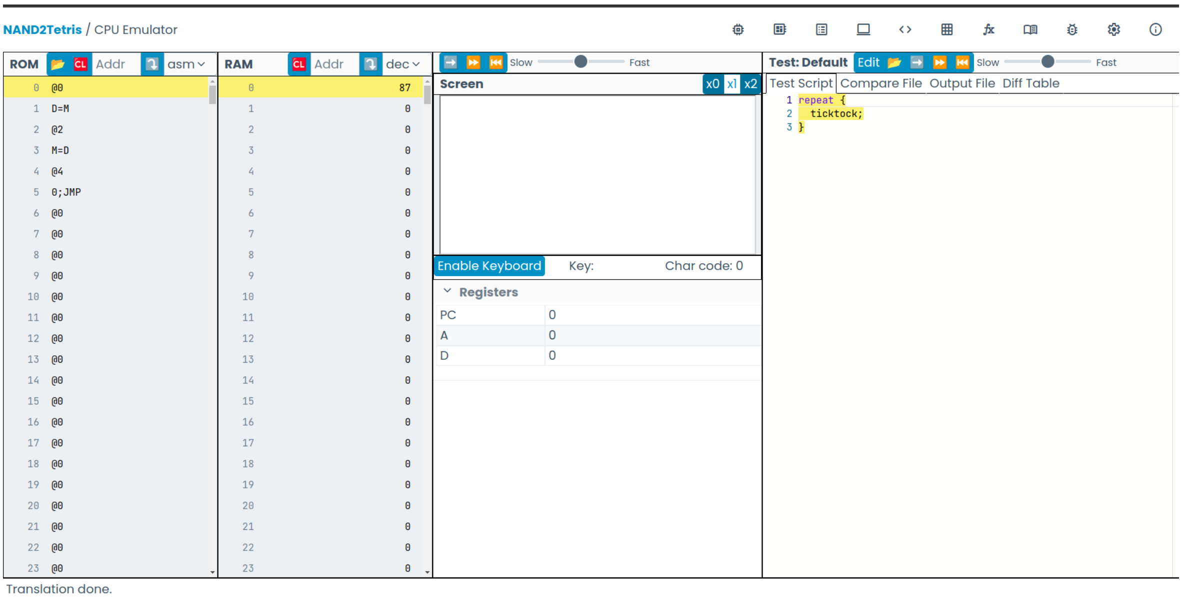
Question Number 0

Write a Hack assembly program that **copies** the value from memory location 0 into memory location 2.

Solution.

- **Before Runnig program**



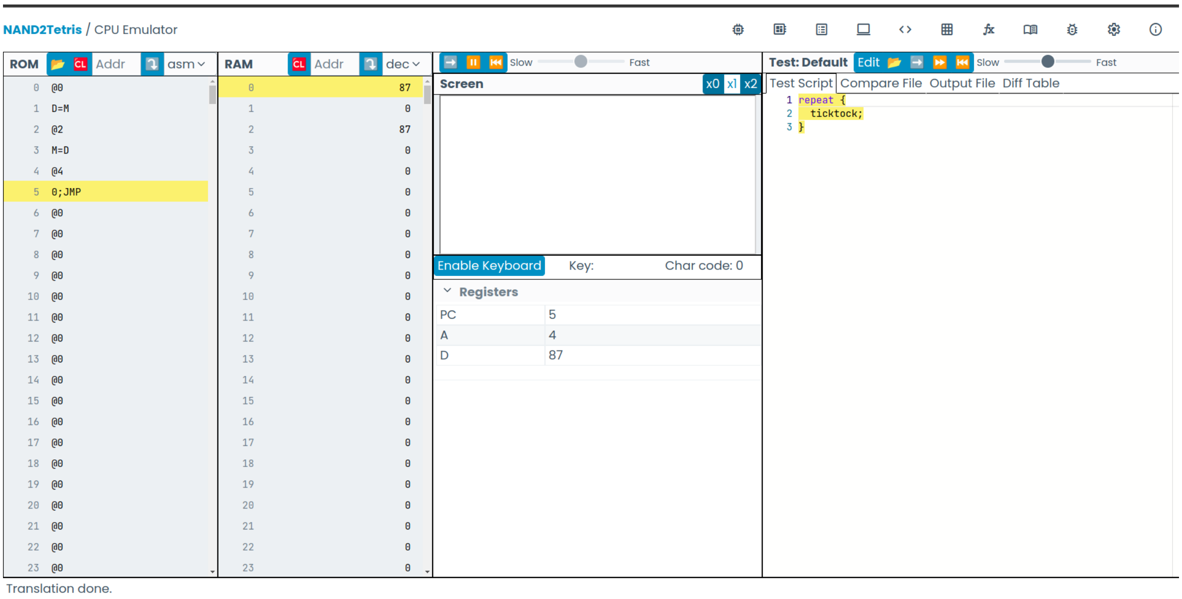
```
ROM
 0  @0
 1  D=M
 2  @2
 3  M=D
 4  @4
 5  0;JMP
 6  @0
 7  @0
 8  @0
 9  @0
10  @0
11  @0
12  @0
13  @0
14  @0
15  @0
16  @0
17  @0
18  @0
19  @0
20  @0
21  @0
22  @0
23  @0
```

```
RAM
 0   87
 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
23    0
```

```
PC  0
A   0
D   0
```

```
repeat {
  ticktock;
}
```

- **After Runnig program**



```
ROM
 0  @0
 1  D=M
 2  @2
 3  M=D
 4  @4
 5  0;JMP
 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
23  @0
```

```
RAM
 0   87
 1    0
 2   87
 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
23    0
```

```
PC  5
A   4
D   87
```

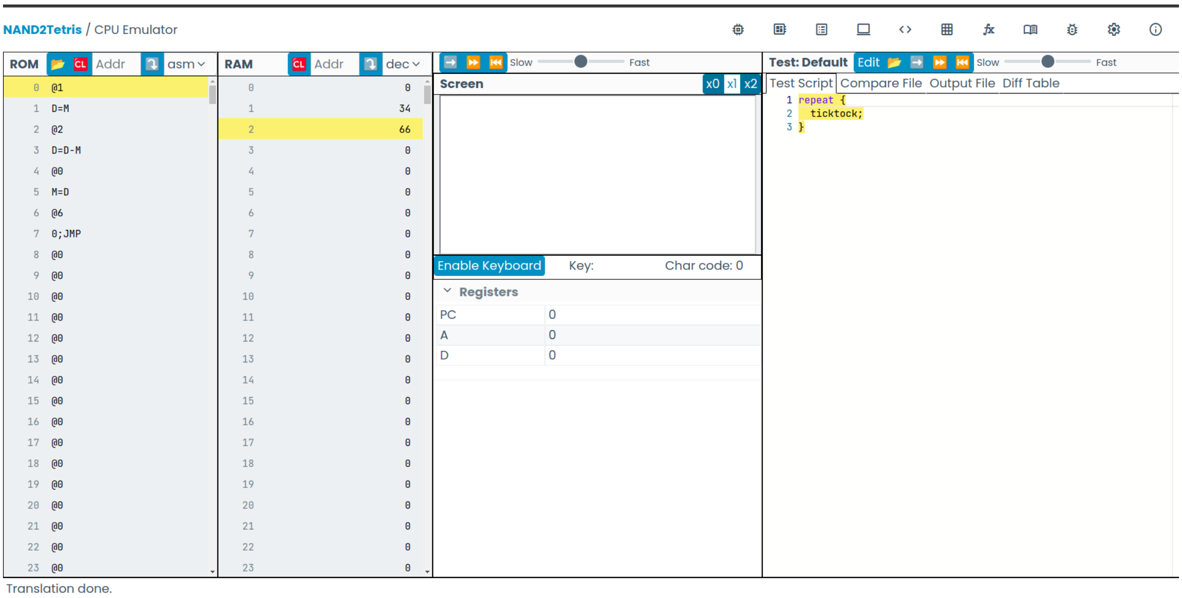
```
repeat {
  ticktock;
}
```

Question Number 1

Write a Hack assembly program that **subtracts** the values stored in memory locations 1 and 2, and stores the result in memory location 0.

Solution.

- **Before Runnig program**



ROM:

```
0  @1
1  D=M
2  @2
3  D=D-M
4  @0
5  M=D
6  @6
7  0;JMP
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
23 @0
```

RAM:

```
0   0
1   34
2   66
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
23  0
```

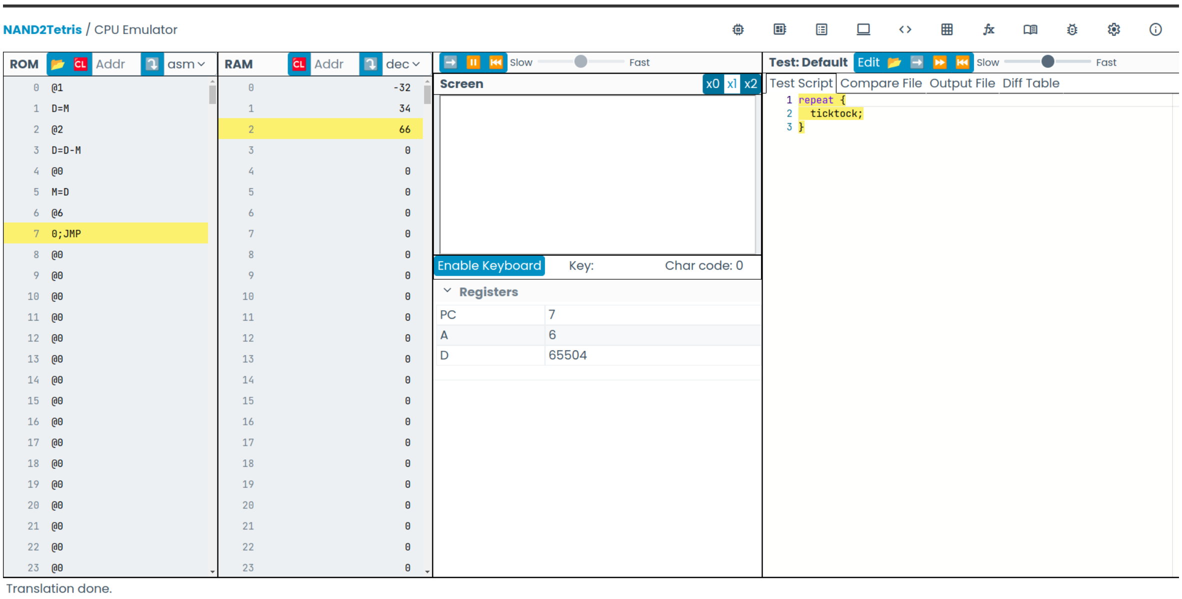
Registers

PC	0
A	0
D	0

```
repeat {
  ticktock;
}
```

Translation done.

- **After Runnig program**



ROM:

```
0  @1
1  D=M
2  @2
3  D=D-M
4  @0
5  M=D
6  @6
7  0;JMP
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
23 @0
```

RAM:

```
0   -32
1   34
2   66
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
23  0
```

Registers

PC	7
A	6
D	65504

```
repeat {
  ticktock;
}
```

Translation done.

Note

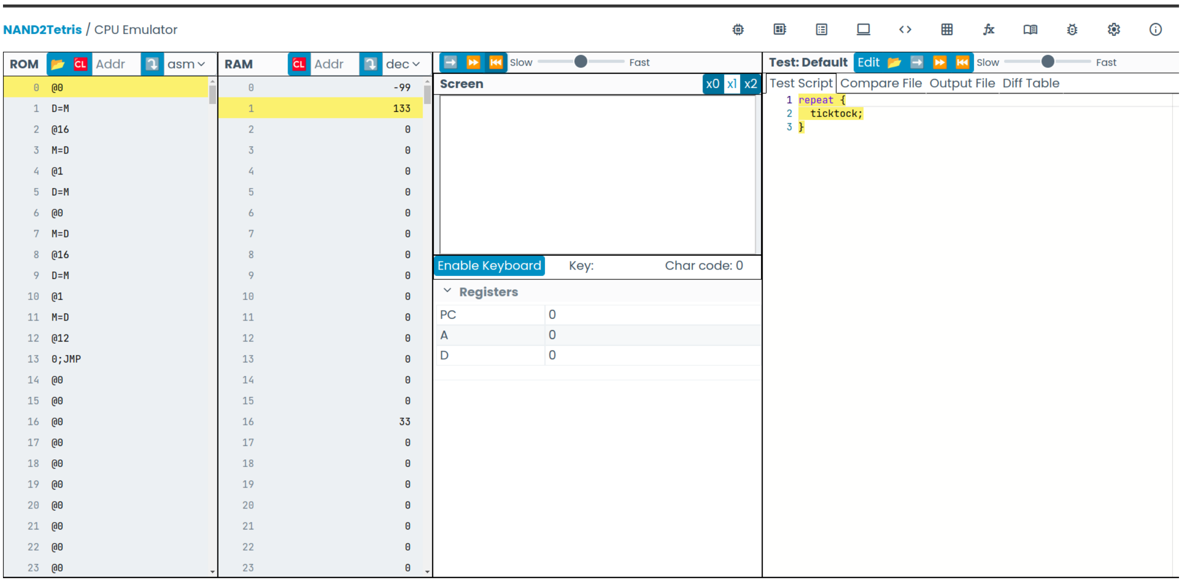
Here I am subtracting $Ram[0] = Ram[1] - Ram[2]$

Question Number 2

Write a Hack assembly program that **swaps** the values stored in memory locations 0 and 1.

Solution.

- **Before Runnig program**



ROM:

```
 0  @0
 1  D=M
 2  @16
 3  M=D
 4  @1
 5  D=M
 6  @0
 7  M=D
 8  @16
 9  D=M
10  @1
11  M=D
12  @12
13  0;JMP
14  @0
15  @0
16  @0
17  @0
18  @0
19  @0
20  @0
21  @0
22  @0
23  @0
```

RAM:

```
 0   -99
 1   133
 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    33
17     0
18     0
19     0
20     0
21     0
22     0
23     0
```

Registers

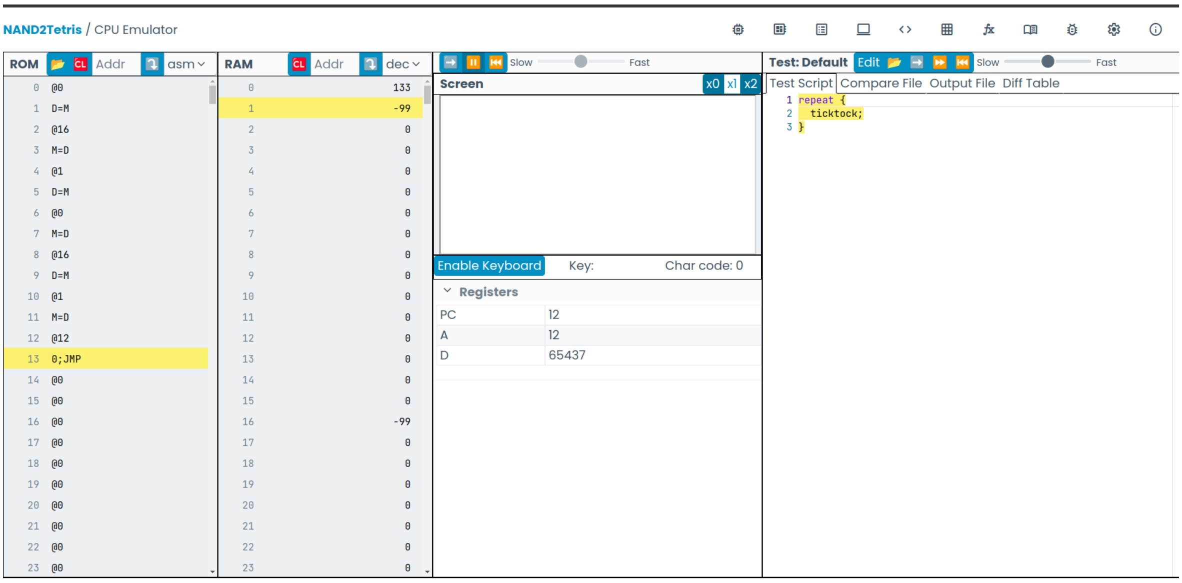
PC	0
A	0
D	0

Test Script:

```
repeat {
  ticktock;
}
```

Translation done.

- **After Runnig program**



ROM:

```
 0  @0
 1  D=M
 2  @16
 3  M=D
 4  @1
 5  D=M
 6  @0
 7  M=D
 8  @16
 9  D=M
10  @1
11  M=D
12  @12
13  0;JMP
14  @0
15  @0
16  @0
17  @0
18  @0
19  @0
20  @0
21  @0
22  @0
23  @0
```

RAM:

```
 0   133
 1   -99
 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   -99
17     0
18     0
19     0
20     0
21     0
22     0
23     0
```

Registers

PC	12
A	12
D	65437

Test Script:

```
repeat {
  ticktock;
}
```

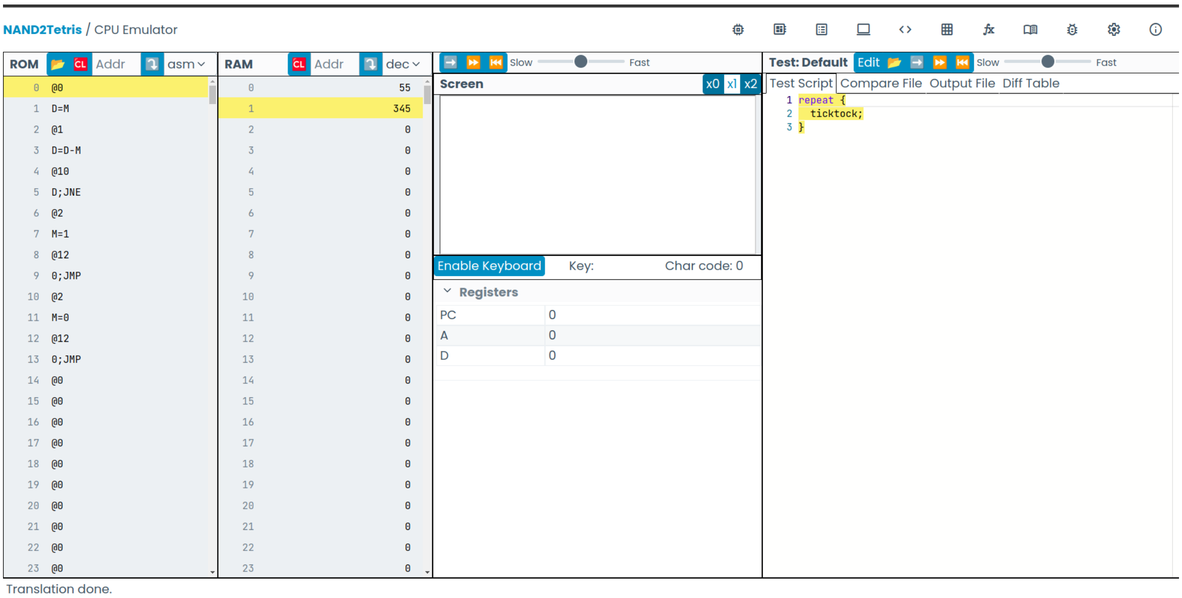
Translation done.

Question Number 3

Write a Hack assembly program that **checks** if the value stored in memory location 0 equals that stored in memory location 1. If they are equal, store 1 in memory location 2; otherwise, store 0.

Solution.

- For **Not Equal** Numbers
  - **Before Runnig program**



```
ROM
0   @0
1   D=M
2   @1
3   D=D-M
4   @10
5   D;JNE
6   @2
7   M=1
8   @12
9   0;JMP
10  @2
11  M=0
12  @12
13  0;JMP
14  @0
15  @0
16  @0
17  @0
18  @0
19  @0
20  @0
21  @0
22  @0
23  @0
```

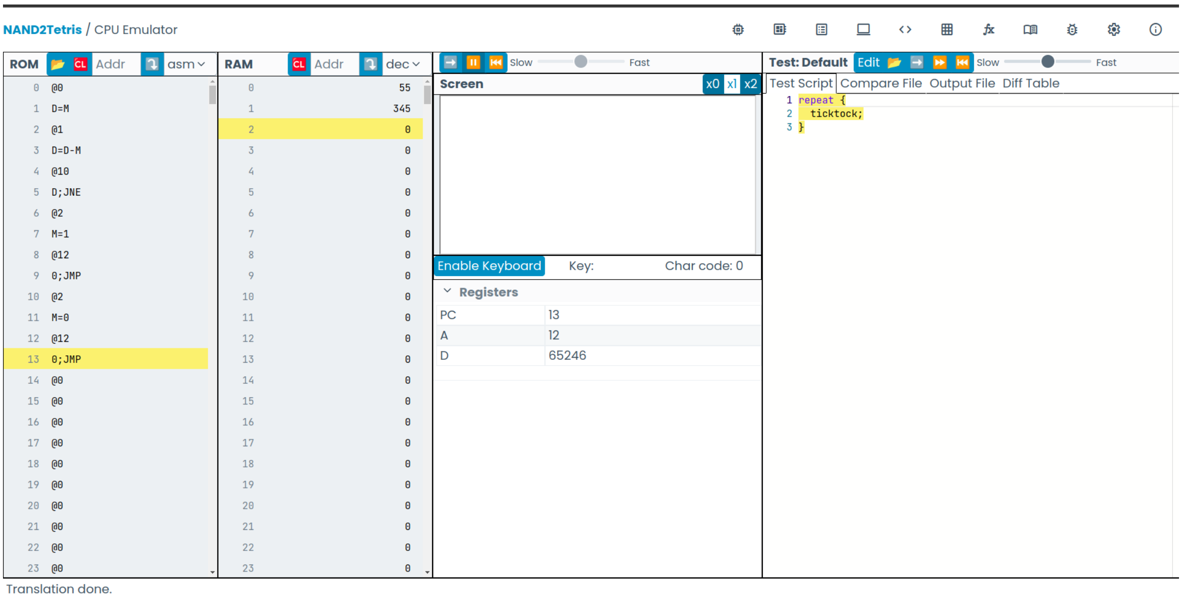
```
RAM
0    55
1    345
2    0
3    0
4    0
5    0
6    0
7    0
8    0
9    0
10   0
11   0
12   0
13   0
14   0
15   0
16   0
17   0
18   0
19   0
20   0
21   0
22   0
23   0
```

```
Registers
PC  0
A   0
D   0
```

```
repeat {
  ticktock;
}
```

Translation done.

  - **After Runnig program**



```
ROM
0   @0
1   D=M
2   @1
3   D=D-M
4   @10
5   D;JNE
6   @2
7   M=1
8   @12
9   0;JMP
10  @2
11  M=0
12  @12
13  0;JMP
14  @0
15  @0
16  @0
17  @0
18  @0
19  @0
20  @0
21  @0
22  @0
23  @0
```

```
RAM
0    55
1    345
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
23   0
```

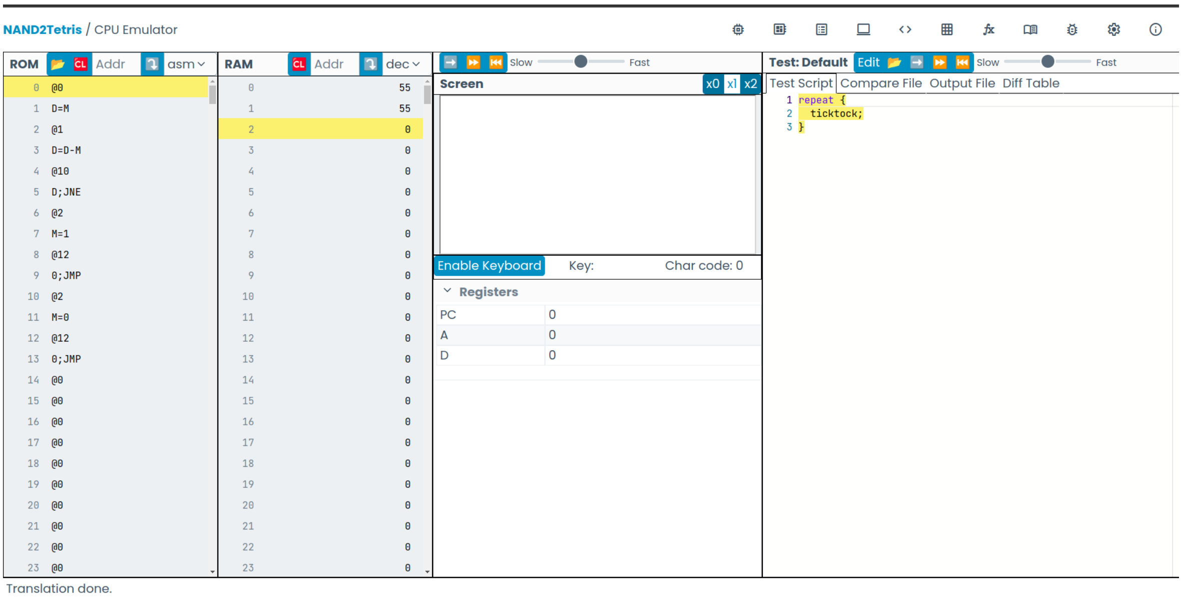
```
Registers
PC  13
A   12
D   65246
```

```
repeat {
  ticktock;
}
```

Translation done.

- For **Equal** Numbers

  - **Before Runnig program**



```
ROM
 0  @0
 1  D=M
 2  @1
 3  D=D-M
 4  @10
 5  D;JNE
 6  @2
 7  M=1
 8  @12
 9  0;JMP
10  @2
11  M=0
12  @12
13  0;JMP
14  @0
15  @0
16  @0
17  @0
18  @0
19  @0
20  @0
21  @0
22  @0
23  @0
```

```
RAM
 0   55
 1   55
 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
23    0
```

Key: Char code: 0

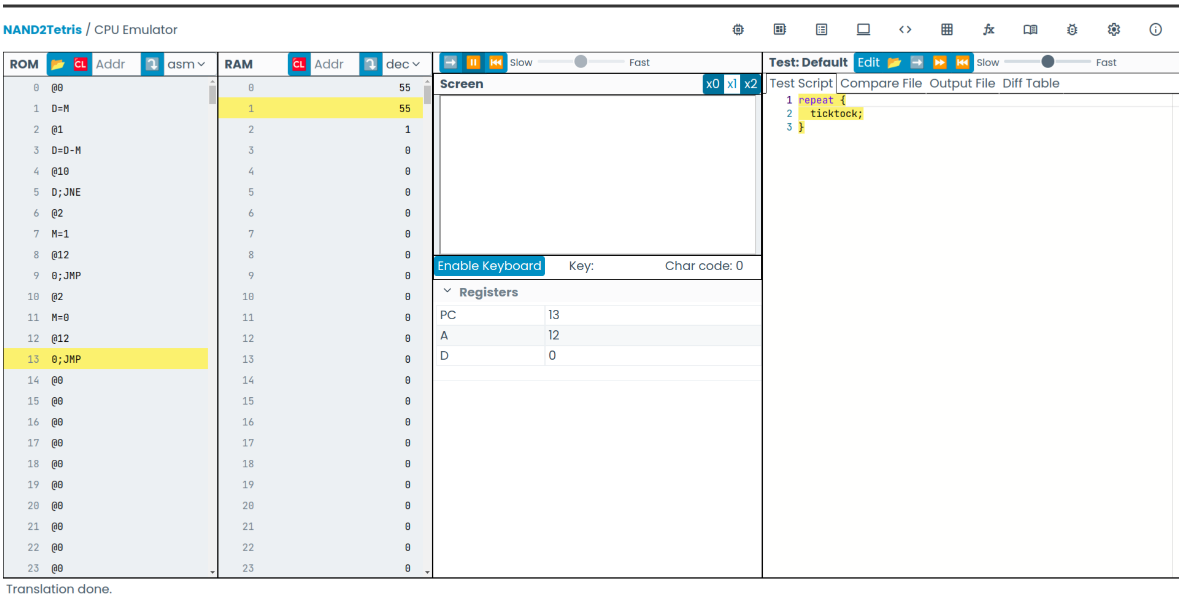
Registers

PC	0
A	0
D	0

```
repeat {
  ticktock;
}
```

Translation done.

  - **After Runnig program**



```
ROM
 0  @0
 1  D=M
 2  @1
 3  D=D-M
 4  @10
 5  D;JNE
 6  @2
 7  M=1
 8  @12
 9  0;JMP
10  @2
11  M=0
12  @12
13  0;JMP
14  @0
15  @0
16  @0
17  @0
18  @0
19  @0
20  @0
21  @0
22  @0
23  @0
```

```
RAM
 0   55
 1   55
 2    1
 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
23    0
```

Key: Char code: 0

Registers

PC	13
A	12
D	0

```
repeat {
  ticktock;
}
```

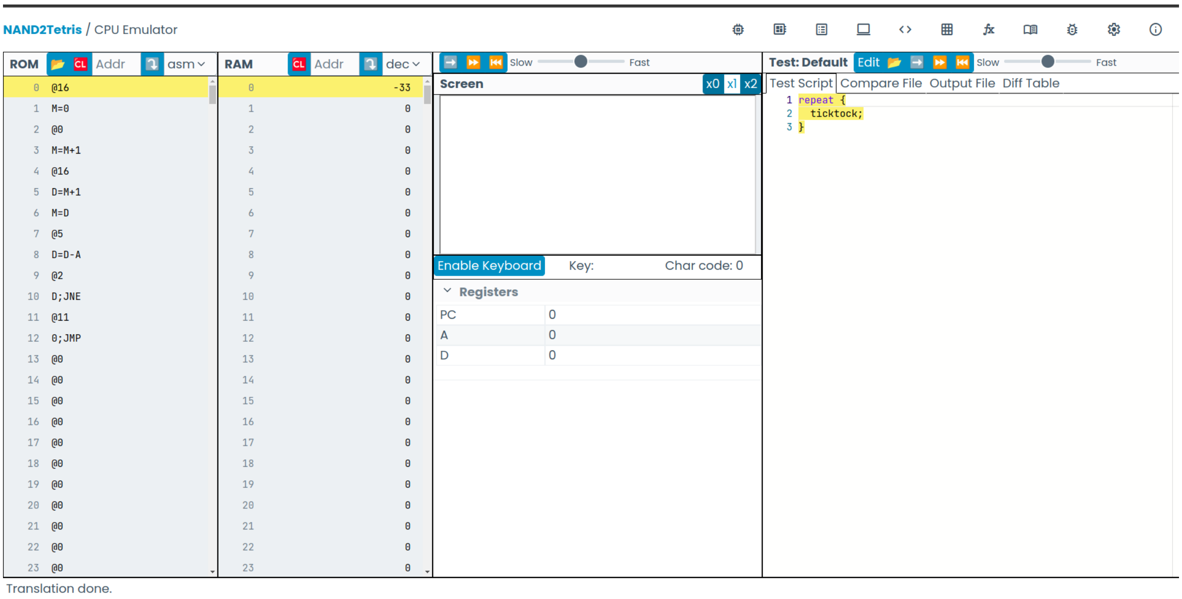
Translation done.

Question Number 4

Write a Hack assembly program that implements a simple **loop** to increment the value in memory location 0 by 1 a total of 5 times, storing the result in memory location 1.

Solution.

- **Before Runnig program**



ROM:

```
 0  @16
 1  M=0
 2  @0
 3  M=M+1
 4  @16
 5  D=M+1
 6  M=D
 7  @5
 8  D=D-A
 9  @2
10  D;JNE
11  @11
12  0;JMP
13  @0
14  @0
15  @0
16  @0
17  @0
18  @0
19  @0
20  @0
21  @0
22  @0
23  @0
```

RAM:

```
 0  -33
 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
23    0
```

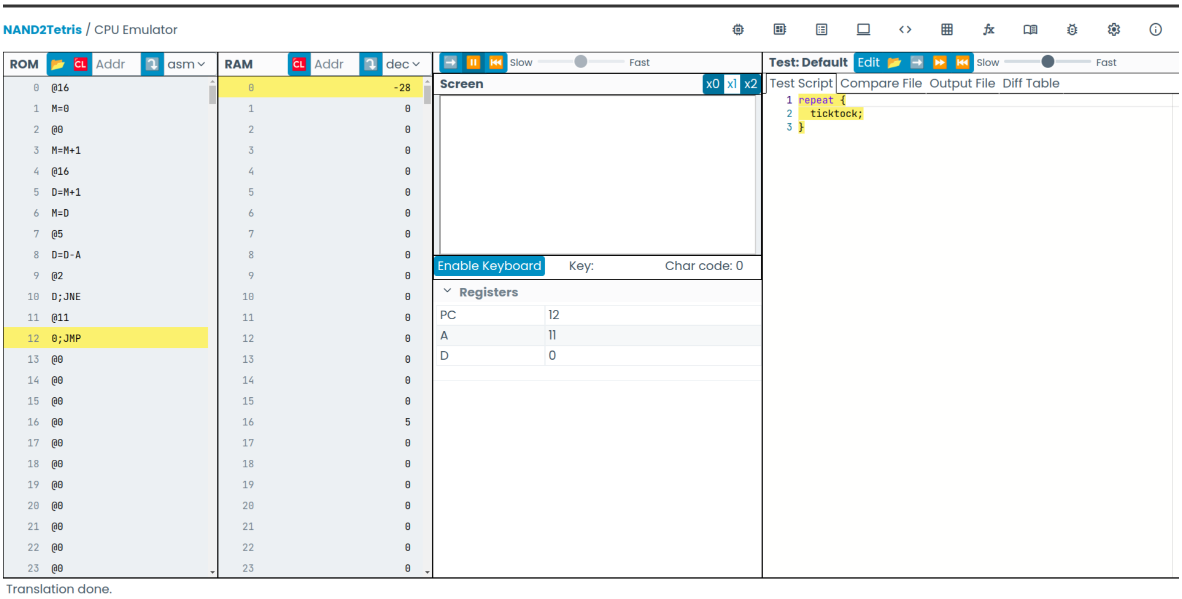
Registers

PC	0
A	0
D	0

```
repeat {
  ticktock;
}
```

Translation done.

- **After Runnig program**



ROM:

```
 0  @16
 1  M=0
 2  @0
 3  M=M+1
 4  @16
 5  D=M+1
 6  M=D
 7  @5
 8  D=D-A
 9  @2
10  D;JNE
11  @11
12  0;JMP
13  @0
14  @0
15  @0
16  @0
17  @0
18  @0
19  @0
20  @0
21  @0
22  @0
23  @0
```

RAM:

```
 0  -28
 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    5
17    0
18    0
19    0
20    0
21    0
22    0
23    0
```

Registers

PC	12
A	11
D	0

```
repeat {
  ticktock;
}
```

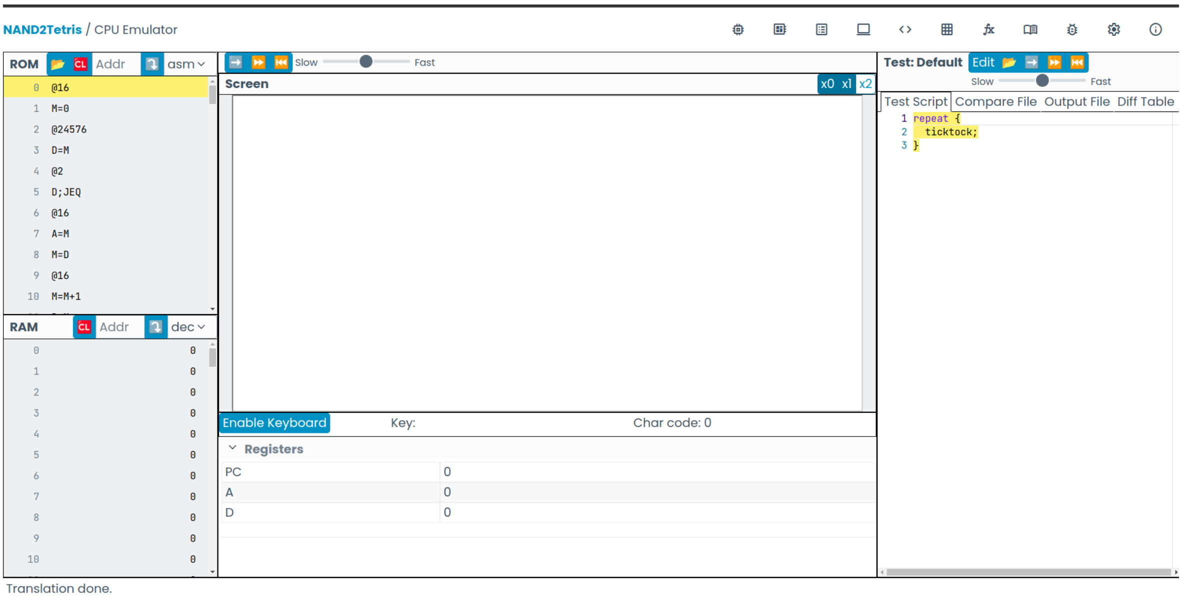
Translation done.

Question Number 5

Write a Hack assembly program that **reads from the keyboard** and stores the code of the first key at RAM[0] and code of the second key at RAM[1] and then adds the codes and stores at RAM[2]. after that it **blackens** the first 16 pixels of row 6 of the screen.

Solution.

- **Before Runnig program**



NAND2Tetris / CPU Emulator

ROM

```
0   @16
1   M=0
2   @24576
3   D=M
4   @2
5   D;JEQ
6   @16
7   A=M
8   M=D
9   @16
10  M=M+1
```

RAM

0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

Key: Char code: 0

Registers

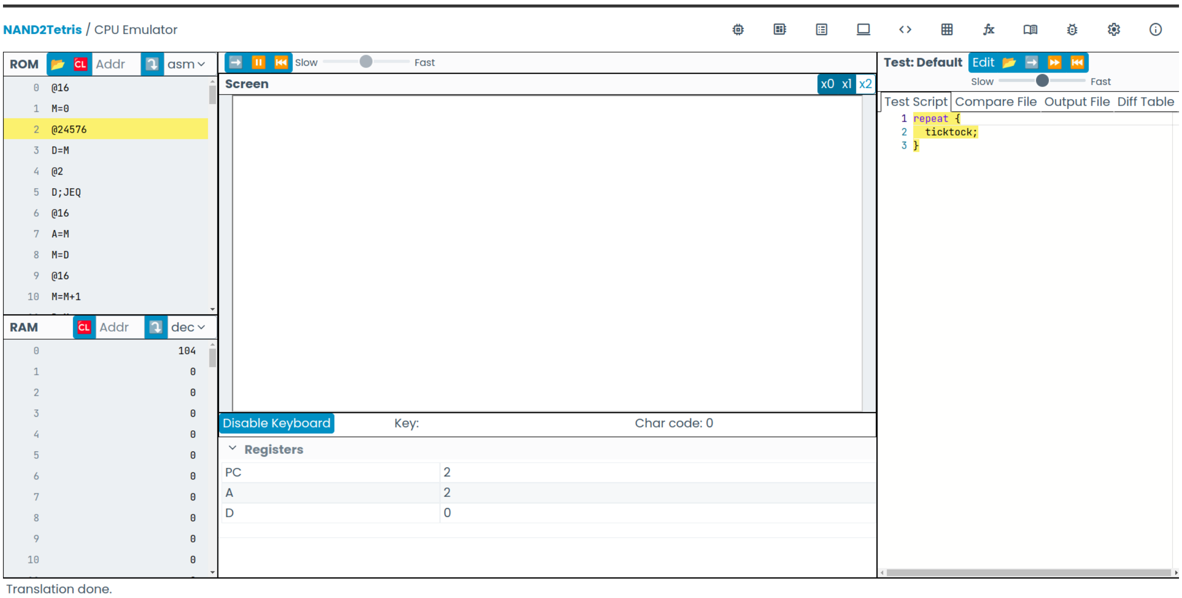
PC	0
A	0
D	0

Test: Default

```
repeat {
  ticktock;
}
```

Translation done.

- **After 1 Key Press**



NAND2Tetris / CPU Emulator

ROM

```
0   @16
1   M=0
2   @24576
3   D=M
4   @2
5   D;JEQ
6   @16
7   A=M
8   M=D
9   @16
10  M=M+1
```

RAM

0	104
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

Key: Char code: 0

Registers

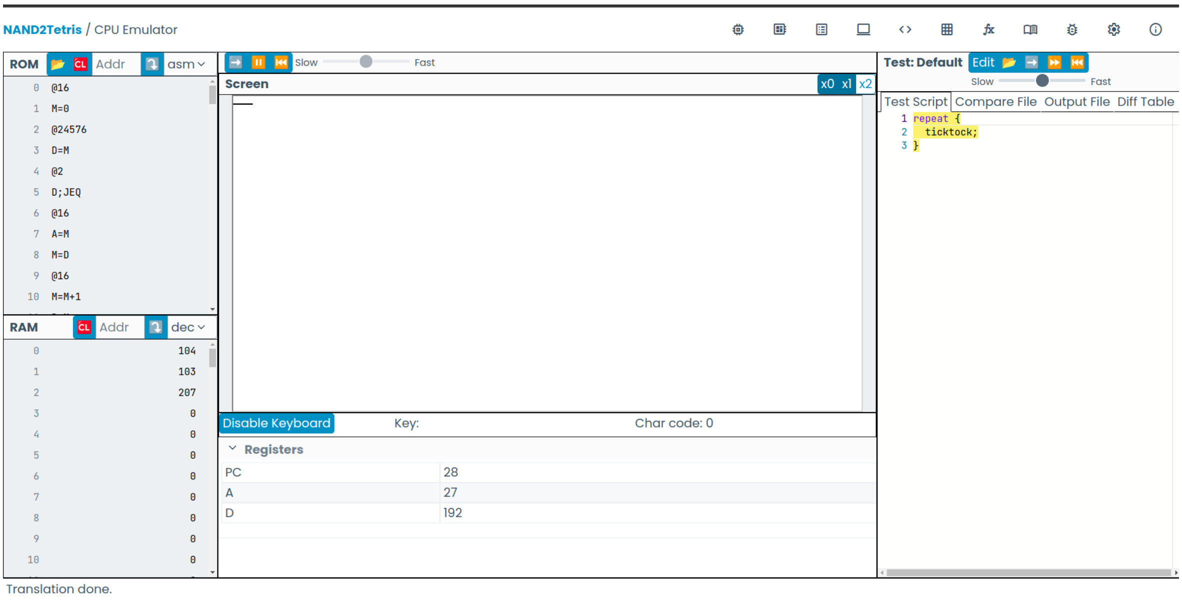
PC	2
A	2
D	0

Test: Default

```
repeat {
  ticktock;
}
```

Translation done.

- **After 2 Key Press**



```
0   @16
1   M=0
2   @24576
3   D=M
4   @2
5   D;JEQ
6   @16
7   A=M
8   M=D
9   @16
10  M=M+1
```

0	104
1	103
2	207
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

Key:

Char code: 0

Registers

PC	28
A	27
D	192

```
repeat {
  ticktock;
}
```

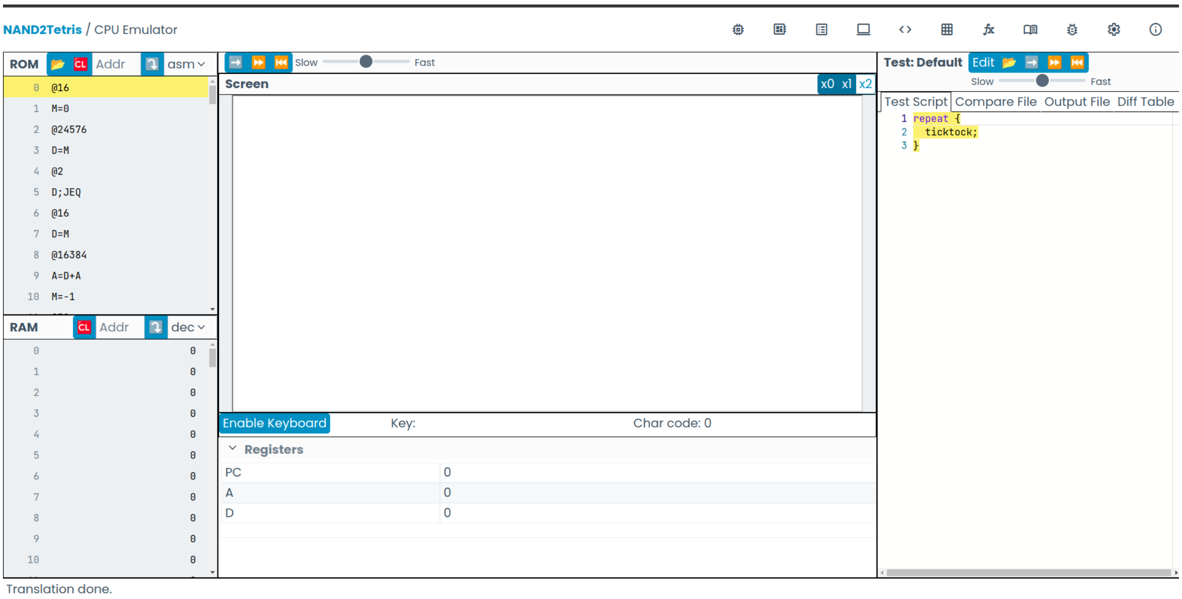
Translation done.

Question Number 6

Write a Hack assembly program that continuously **checks for keyboard input**. Whenever any key is pressed, the program should **black** the first 16 pixels of the top row(top left corner) of the screen. The program should keep running, waiting for additional key presses, and each key press should result in a black line being drawn on the screen.

Solution.

- **Before Runnig program**



```
0  @16
1  M=0
2  @24576
3  D=M
4  @2
5  D;JEQ
6  @16
7  D=M
8  @16384
9  A=D+A
10 M=-1
```

```
RAM
0   0
1   0
2   0
3   0
4   0
5   0
6   0
7   0
8   0
9   0
10  0
```

Key: Char code: 0

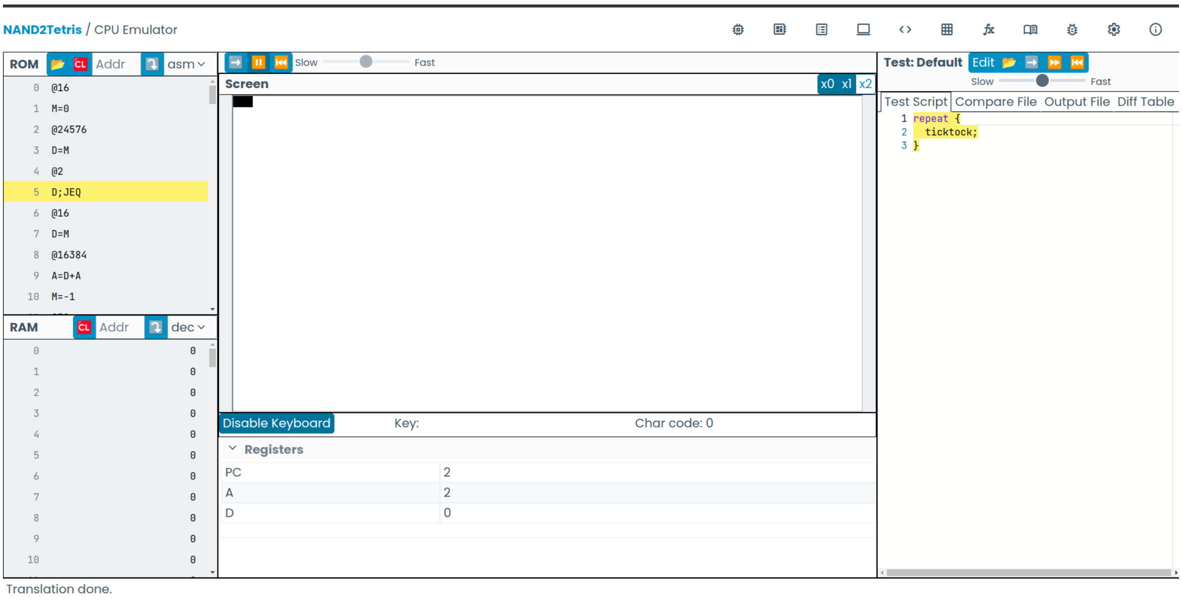
Registers

```
PC  0
A   0
D   0
```

```
repeat {
  ticktock;
}
```

Translation done.

- **After 10 Key Press**



```
0  @16
1  M=0
2  @24576
3  D=M
4  @2
5  D;JEQ
6  @16
7  D=M
8  @16384
9  A=D+A
10 M=-1
```

```
RAM
0   0
1   0
2   0
3   0
4   0
5   0
6   0
7   0
8   0
9   0
10  0
```

Key: Char code: 0

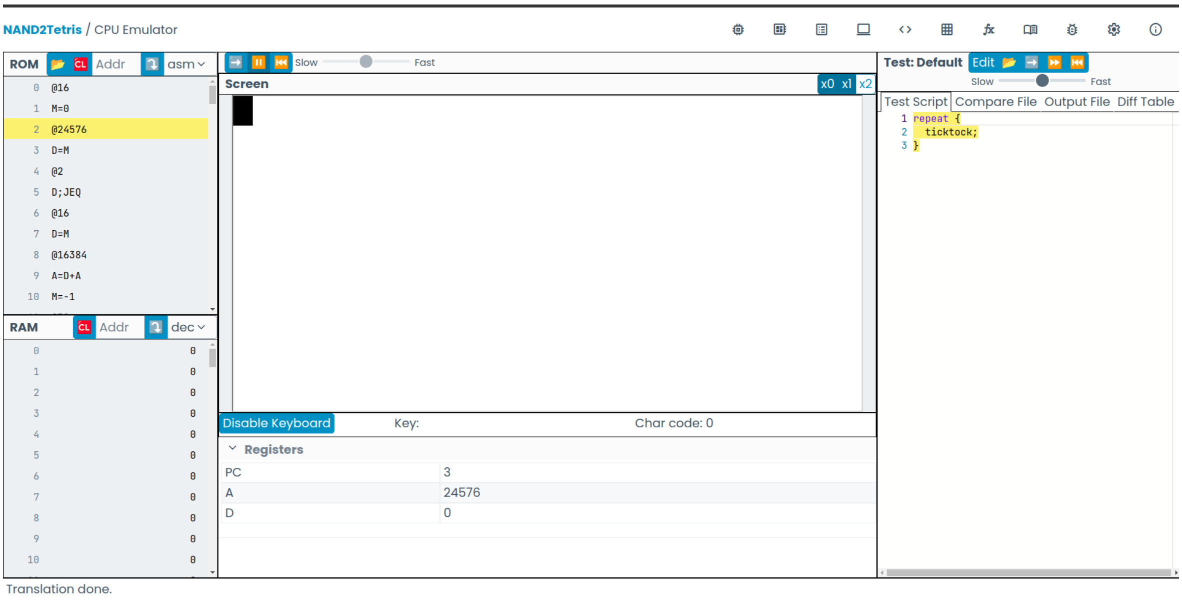
Registers

```
PC  2
A   2
D   0
```

```
repeat {
  ticktock;
}
```

Translation done.

- **After 25 Key Press**



ROM

```
0   @16
1   M=0
2   @24576
3   D=M
4   @2
5   D;JEQ
6   @16
7   D=M
8   @16384
9   A=D+A
10  M=-1
```

RAM

Addr	Value
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

Disable Keyboard Key: Char code: 0

Registers

Register	Value
PC	3
A	24576
D	0

Test Script

```
repeat {
  ticktock;
}
```

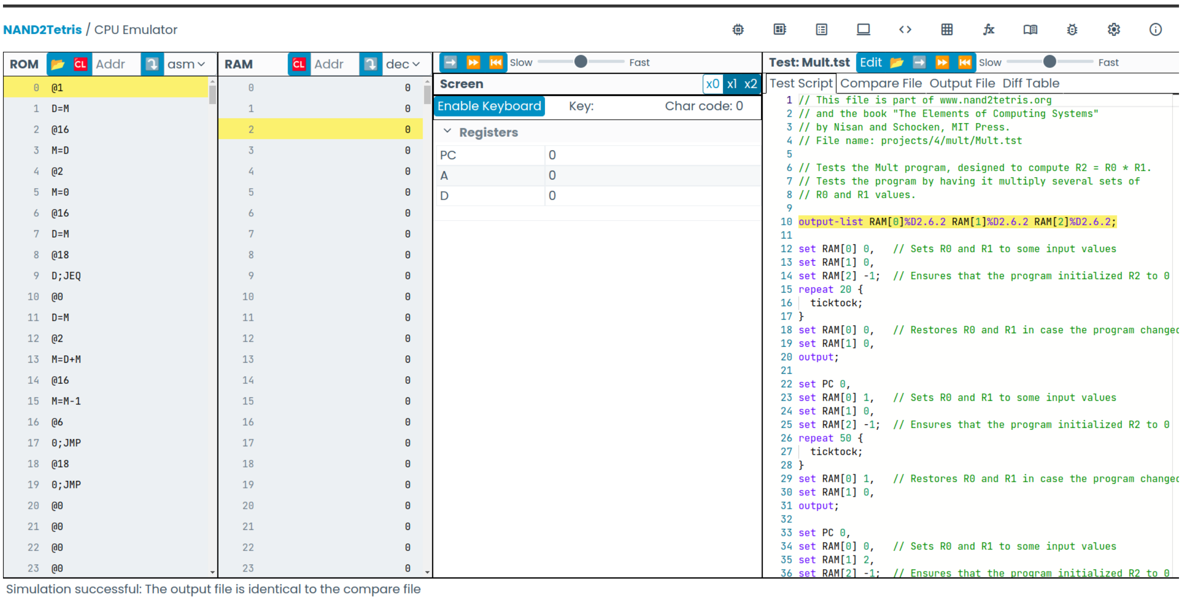
Translation done.

Question Number 7

Mult

Solution.

- **Before Runnig program**



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

```
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  0
17  0
18  0
19  0
20  0
21  0
22  0
23  0
```

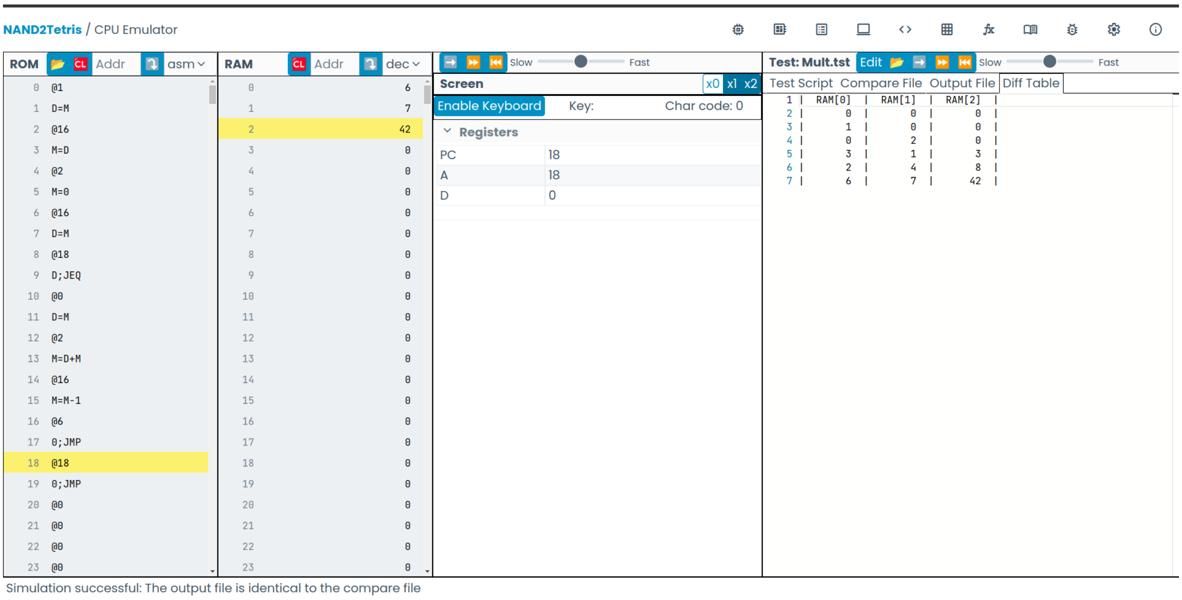
Registers

PC	0
A	0
D	0

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/4/mult/Mult.tst

// Tests the Mult program, designed to compute R2 = R0 * R1.
// Tests the program by having it multiply several sets of
// R0 and R1 values.

output-list RAM[0]%D2.6.2 RAM[1]%D2.6.2 RAM[2]%D2.6.2;

set RAM[0] 0,   // Sets R0 and R1 to some input values
set RAM[1] 0,
set RAM[2] -1;  // Ensures that the program initialized R2 to 0
repeat 20 {
  ticktock;
}
set RAM[0] 0,   // Restores R0 and R1 in case the program changec
set RAM[1] 0,
output;

set PC 0,
set RAM[0] 1,   // Sets R0 and R1 to some input values
set RAM[1] 0,
set RAM[2] -1;  // Ensures that the program initialized R2 to 0
repeat 50 {
  ticktock;
}
set RAM[0] 1,   // Restores R0 and R1 in case the program changec
set RAM[1] 0,
output;

set PC 0,
set RAM[0] 0,   // Sets R0 and R1 to some input values
set RAM[1] 2,
set RAM[2] -1;  // Ensures that the program initialized R2 to 0
```

Simulation successful: The output file is identical to the compare file

- **After running program using mult.tst file**



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

```
RAM
0   6
1   7
2   42
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
23  0
```

Registers

PC	18
A	18
D	0

```
|  RAM[0]  |  RAM[1]  |  RAM[2]  |
|       0  |       0  |       0  |
|       1  |       0  |       0  |
|       0  |       2  |       0  |
|       3  |       1  |       3  |
|       2  |       4  |       8  |
|       6  |       7  |      42  |
```

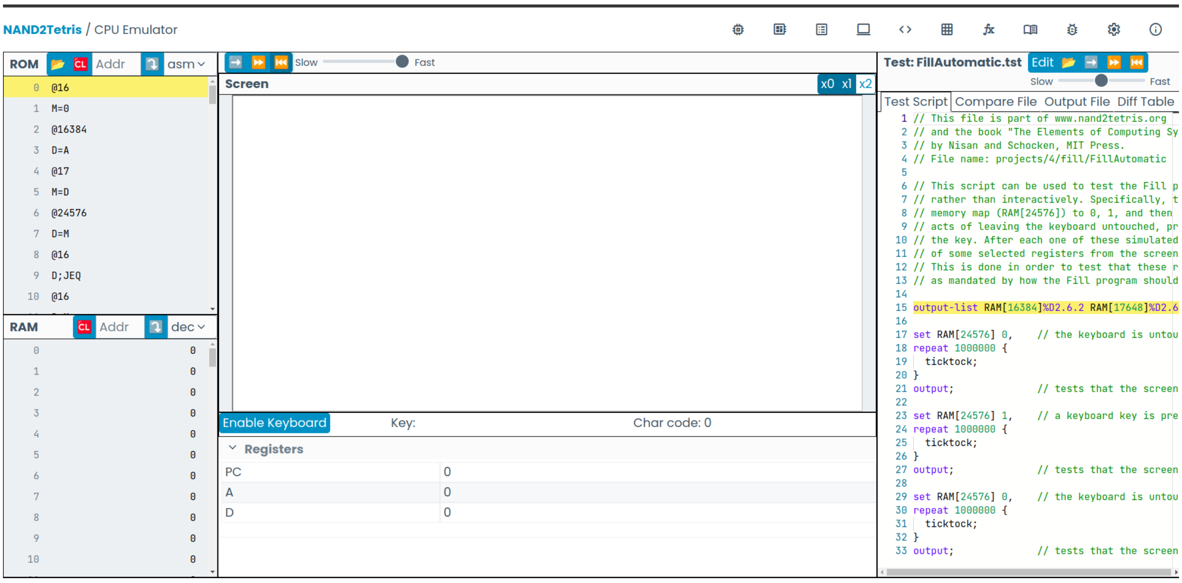
Simulation successful: The output file is identical to the compare file

Question Number 8

Fill

Solution.

- **Before Runnig program**



ROM:

```
 0  @16
 1  M=0
 2  @16384
 3  D=A
 4  @17
 5  M=D
 6  @24576
 7  D=M
 8  @16
 9  D;JEQ
10  @16
```

RAM:

```
 0    0
 1    0
 2    0
 3    0
 4    0
 5    0
 6    0
 7    0
 8    0
 9    0
10    0
```

Registers:

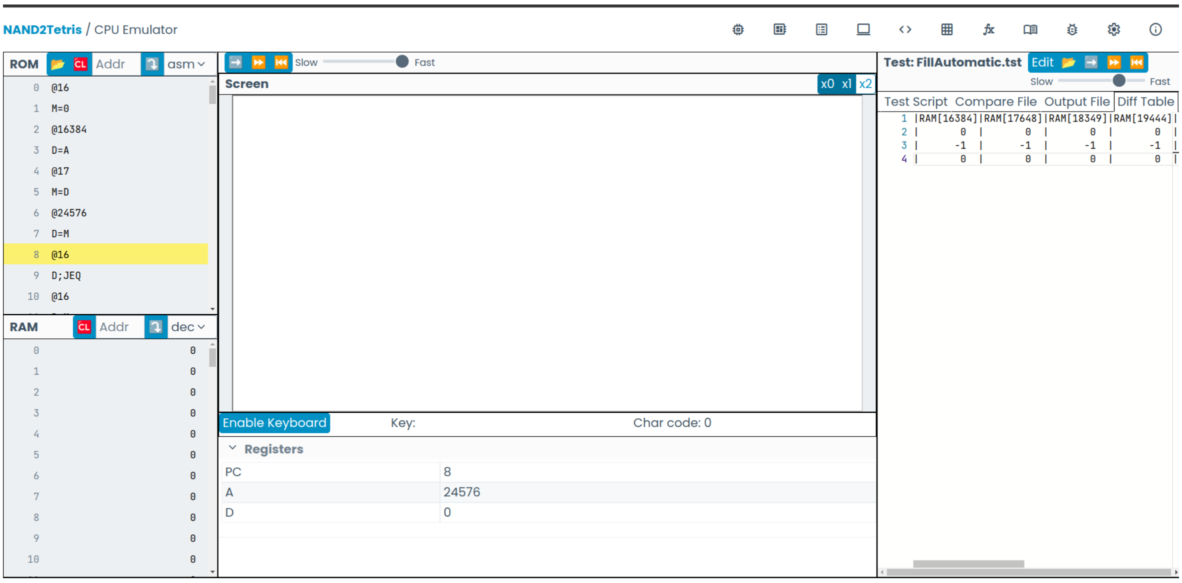
```
PC   0
A    0
D    0
```

Test Script:

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Sy
// by Nisan and Schocken, MIT Press.
// File name: projects/4/fill/FillAutomatic

// This script can be used to test the Fill p
// rather than interactively. Specifically, t
// memory map (RAM[24576]) to 0, 1, and then
// acts of leaving the keyboard untouched, pr
// the key. After each one of these simulated
// of some selected registers from the screen
// This is done in order to test that these r
// as mandated by how the Fill program should

output-list RAM[16384]%D2.6.2 RAM[17648]%D2.6

set RAM[24576] 0,    // the keyboard is untou
repeat 1000000 {
  ticktock;
}
output;              // tests that the screen

set RAM[24576] 1,    // a keyboard key is pre
repeat 1000000 {
  ticktock;
}
output;              // tests that the screen

set RAM[24576] 0,    // the keyboard is untou
repeat 1000000 {
  ticktock;
}
output;              // tests that the screen
```

Translation done.

- **After running program using FillAutomatic.tst file**



ROM:

```
 0  @16
 1  M=0
 2  @16384
 3  D=A
 4  @17
 5  M=D
 6  @24576
 7  D=M
 8  @16
 9  D;JEQ
10  @16
```

RAM:

```
 0    0
 1    0
 2    0
 3    0
 4    0
 5    0
 6    0
 7    0
 8    0
 9    0
10    0
```

Registers:

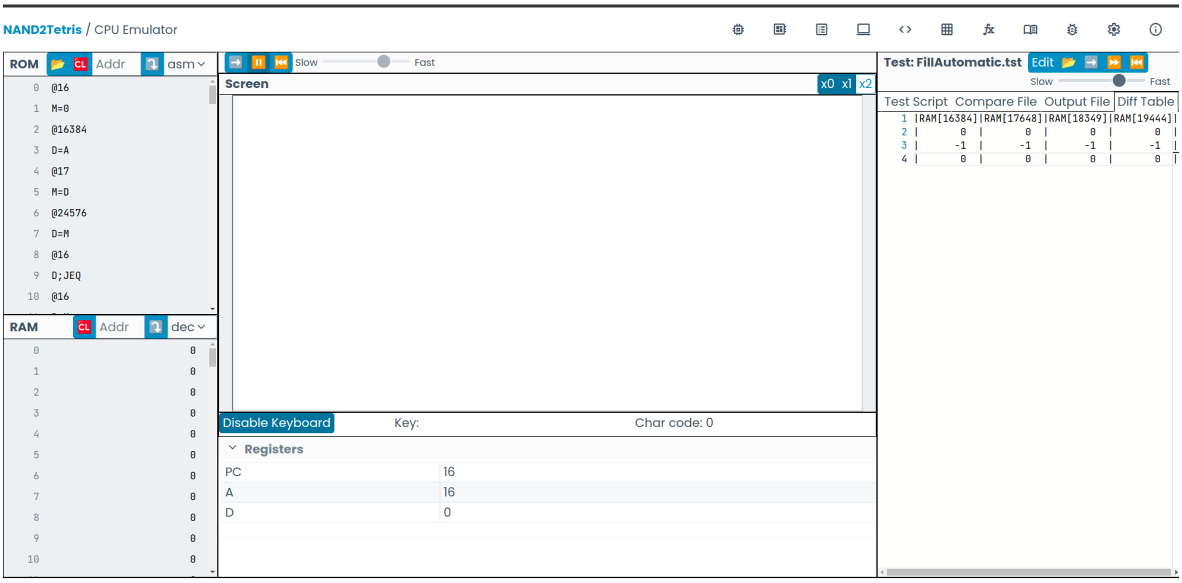
```
PC   8
A    24576
D    0
```

Diff Table:

```
|RAM[16384]|RAM[17648]|RAM[18349]|RAM[19444]|
|     0    |     0    |     0    |     0    |
|    -1    |    -1    |    -1    |    -1    |
|     0    |     0    |     0    |     0    |
```

Simulation successful: The output file is identical to the compare file

- **When no key pressed**



```
0  @16
1  M=0
2  @16384
3  D=A
4  @17
5  M=D
6  @24576
7  D=M
8  @16
9  D;JEQ
10 @16
```

```
0   0
1   0
2   0
3   0
4   0
5   0
6   0
7   0
8   0
9   0
10  0
```

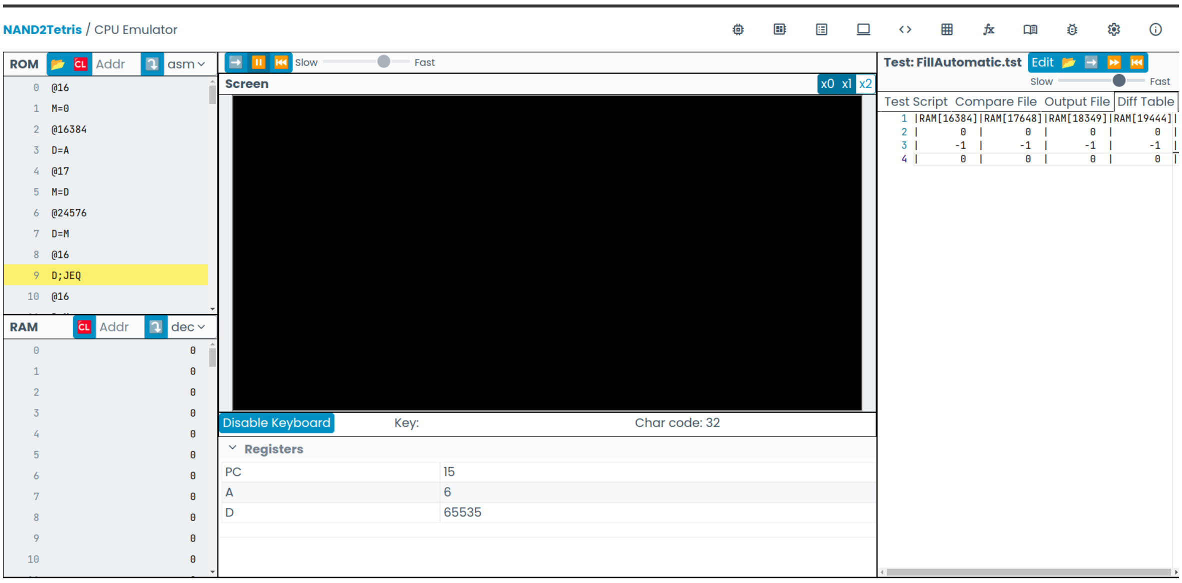
Key: Char code: 0

Registers	
PC	16
A	16
D	0

```
|RAM[16384]|RAM[17648]|RAM[18349]|RAM[19444]|
|        0 |        0 |        0 |        0 |
|       -1 |       -1 |       -1 |       -1 |
|        0 |        0 |        0 |        0 |
```

Simulation successful: The output file is identical to the compare file

- **When key get pressed**



```
0  @16
1  M=0
2  @16384
3  D=A
4  @17
5  M=D
6  @24576
7  D=M
8  @16
9  D;JEQ
10 @16
```

```
0   0
1   0
2   0
3   0
4   0
5   0
6   0
7   0
8   0
9   0
10  0
```

Key: Char code: 32

Registers	
PC	15
A	6
D	65535

```
|RAM[16384]|RAM[17648]|RAM[18349]|RAM[19444]|
|        0 |        0 |        0 |        0 |
|       -1 |       -1 |       -1 |       -1 |
|        0 |        0 |        0 |        0 |
```

Simulation successful: The output file is identical to the compare file