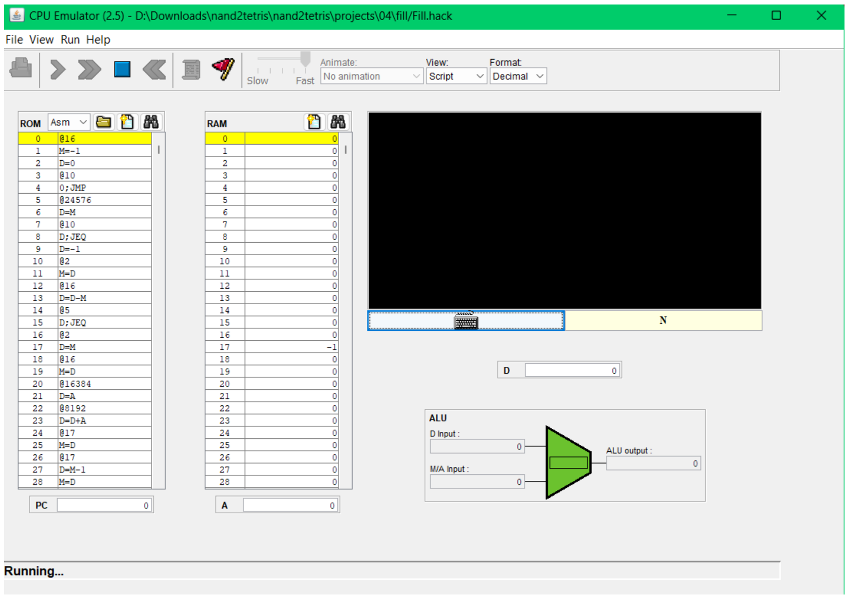
- @status: Initialize the status variable in RAM. M=-1 sets the status to 0xFFFF in binary, which corresponds to all 1 bits. This will represent the black color for the screen.
- D=0: Set the D register to 0, representing the argument for what to set the screen bits to.
- @SETSCREEN: Jump to the SETSCREEN label.
- (LOOP): This is a label marking the beginning of the infinite loop.
- @KBD: Load the value of the keyboard register (KBD) into the D register.
- D;JEQ: If the keyboard input is equal to 0 (no key pressed), jump to SETSCREEN with the argument 0 (white color). Otherwise, continue with the next instruction.
- D=-1: If a key is pressed, set the D register to -1 (0xFFFF in binary).
- (SETSCREEN): This is a label that sets the new status in the ARG register before jumping to SETLOOP.
- @ARG: Load the argument into the D register.
- M=D: Save the new status argument in the status variable.
- @status: Load the current status into the D register.
- D=D-M: Subtract the old status from the new status and store the result in the D register.
- @LOOP: If the result is zero (new status equals old status), jump to LOOP and do nothing.
- D;JEQ: Jump to LOOP if the new status equals the old status.
- The program proceeds to the next instructions if the new status is different from the old status:
- @ARG: Load the argument (new status) into the D register.
- @status: Store the new status in the status variable.
- @SCREEN: Load the address of the SCREEN variable into the D register.
- D=A: Set the D register to the SCREEN address.
- @8192: Load the value 8192 into the A register.
- D=D+A: Add the SCREEN address to 8192, storing the result in the D register. This gives the byte just past the last screen address.
- @i: Store the value in the D register in the i variable.
- (SETLOOP): This is a label marking the beginning of the loop that sets the screen.
- @i: Load the value of i into the D register.
- D=M-1: Decrement the value of i by 1 and store the result in the D register.
- M=D: Update the value of i.
- @LOOP: If i is less than 0, jump to LOOP and repeat the process.
- D;JLT: Jump to LOOP if i is less than 0.
- After the loop, the program updates the screen status for each pixel and then jumps back to SETLOOP, effectively creating a loop that continuously updates the screen based on keyboard input.
- This program uses a simple loop to repeatedly check the keyboard input and update the screen accordingly. When a key is pressed, it sets the entire screen to black, and when no key is pressed, it sets the entire screen to white.


CPU Emulator (2.5) - D:\Downloads\nand2tetris\nand2tetris\projects\04\fill/Fill.hack

Animate: No animation   View: Script   Format: Decimal

ROM	Asm	RAM	
0	@16	0	0
1	M=-1	1	0
2	D=0	2	0
3	@10	3	0
4	0;JMP	4	0
5	@24576	5	0
6	D=M	6	0
7	@10	7	0
8	D;JEQ	8	0
9	D=-1	9	0
10	@2	10	0
11	M=D	11	0
12	@16	12	0
13	D=D-M	13	0
14	@5	14	0
15	D;JEQ	15	0
16	@2	16	0
17	D=M	17	-1
18	@16	18	0
19	M=D	19	0
20	@16384	20	0
21	D=A	21	0
22	@8192	22	0
23	D=D+A	23	0
24	@17	24	0
25	M=D	25	0
26	@17	26	0
27	D=M-1	27	0
28	M=D	28	0

PC 0   A 0   D 0

ALU
D Input : 0
M/A Input : 0
ALU output : 0

Running...