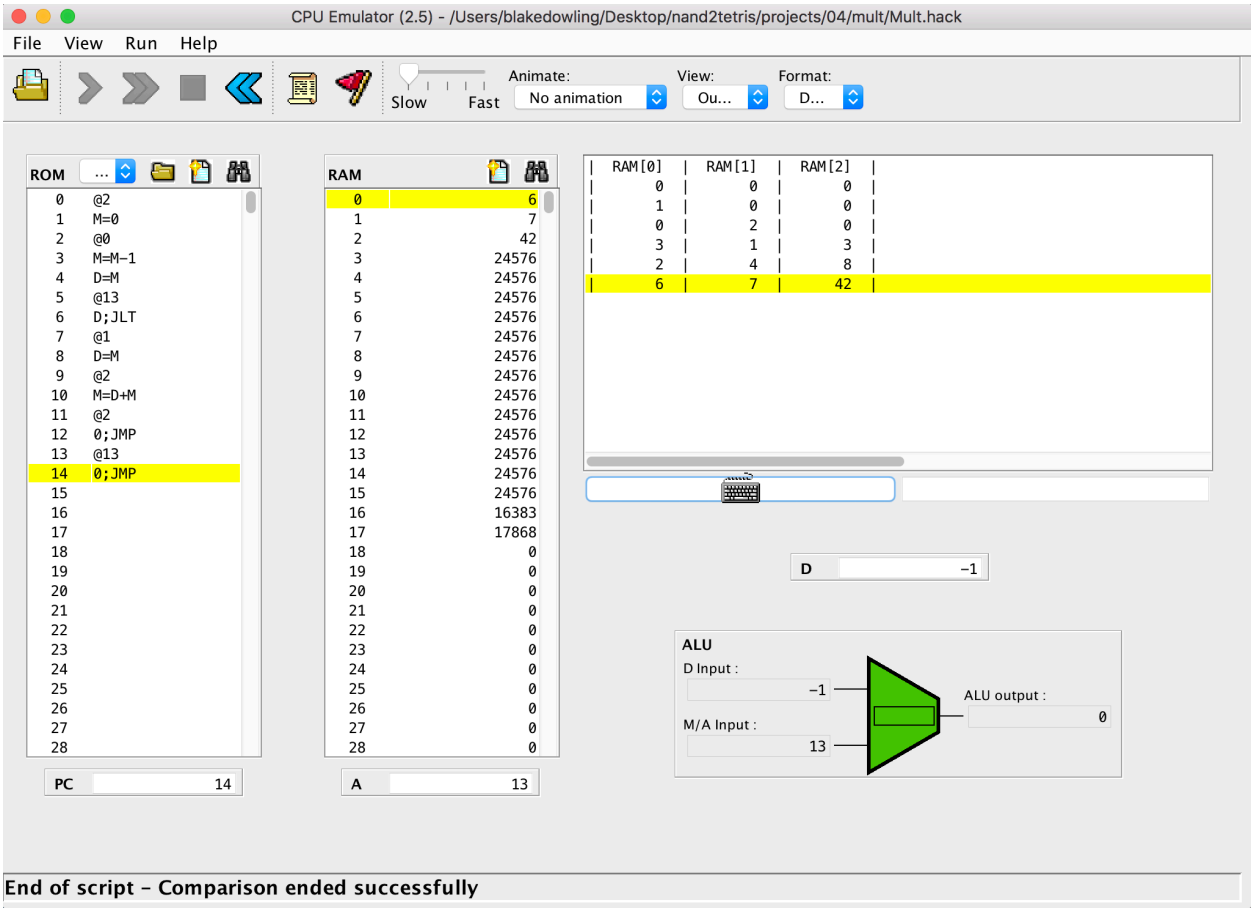
Blake Dowling
Professor Bhrigu Celly
14 October, 2020
CS220 Homework #05 - Machine Language

*Note: indentation of code was altered slightly when copying to this file.

Mult.asm Code

```
// 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/04/Mult.asm

// Multiplies R0 and R1 and stores the result in R2.
// (R0, R1, R2 refer to RAM[0], RAM[1], and RAM[2], respectively.)

    //initialize R2 (answer address) to 0
    @R2 //go to R2
    M=0 //set R2 to 0

    //One iteration of this loop adds R1 to R2 if R0 > 0.
(LOOP) //loops R0 + 1 times, adds R1 to R2, R0 times (R1*R0)
    //if incoming R0 = 0, jump to end
    @R0 //go to R0. *This is referred to as "incoming R0 value".
    M=M-1 //decrement R0 (if R0 initially  equals 0, D will be < 0, and program jumps to END
label)
    D=M //retrieve decremented R0 value
    @END //will jump to END label if incoming R0 value is 0 (if D < 0, because D = R0 - 1)
    D;JLT //jump to END label if D < 0, because D = (incoming R0 value) - 1

    //if incoming R0 value is > 0, continue
    @R1 //go to R1
    D=M //get R1 value
    @R2 //go to R2
    M=M+D //add R1 value to R2 and set to R2
    @LOOP //go to LOOP label unconditionally, we already know R0 >= 0
    0;JMP //jump unconditionally to LOOP label

(END) //jump here when incoming R0 value is 0
    @END //will jump to END label infinitely
    0;JMP //infinite loop to END label
```

Mult.tst Result

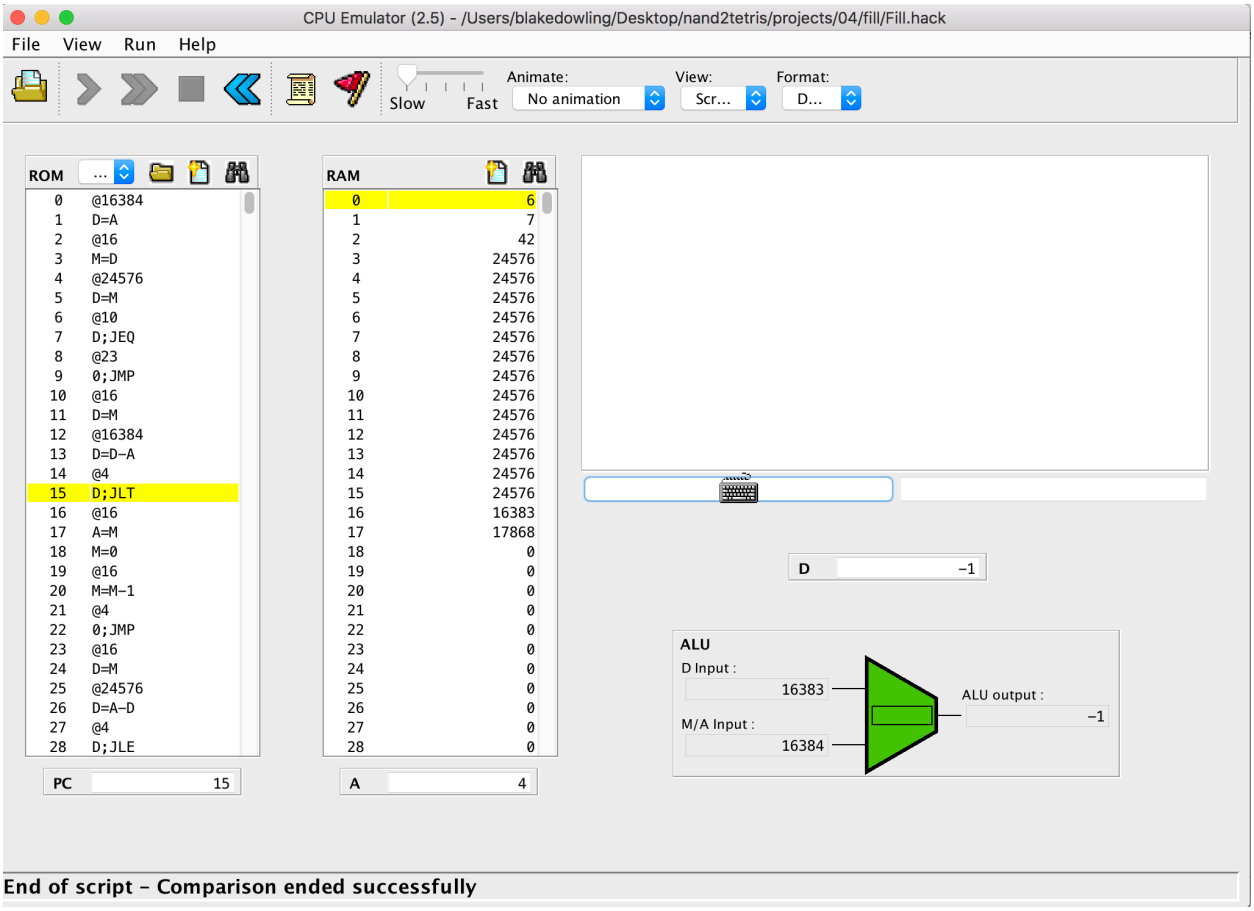


Fill.asm Code

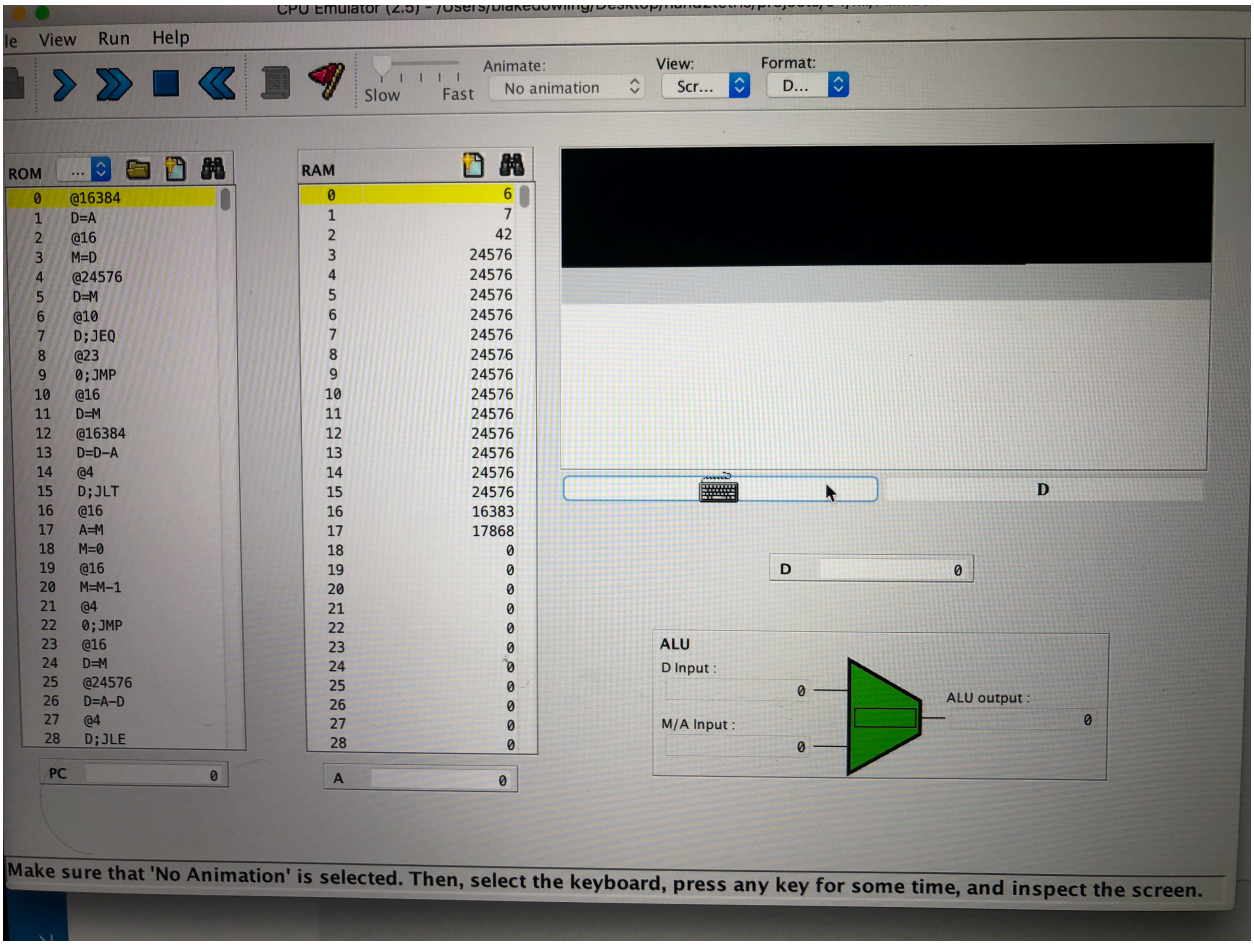
```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/04/Fill.asm

// Runs an infinite loop that listens to the keyboard input.
// When a key is pressed (any key), the program blackens the screen,
// i.e. writes "black" in every pixel;
// the screen should remain fully black as long as the key is pressed.
// When no key is pressed, the program clears the screen, i.e. writes
// "white" in every pixel;
// the screen should remain fully clear as long as no key is pressed.


    //pointer variable starts at top-left of screen
    @SCREEN //go to first screen address
    D=A //retrieve first screen address
    @pointer //go to screen pointer variable address
    M=D //initialize screen pointer value at first screen address, 16384

(MAIN) //takes value of KBD address and jumps to ERASE label if 0 or otherwise jumps to DRAW
label.
    @KBD //go to address of register holding keyboard input value
    D=M //retrieve keyboard input value
    @ERASE //jump to ERASE if keyboard input is equal to 0 (no keys are pressed)
    D;JEQ //jump to ERASE if keyboard input is equal to 0
    @DRAW //keyboard input value is not equal to zero, jump to DRAW (a key is being pressed)
    0;JMP //keyboard input value is not equal to zero, jump to DRAW


(ERASE) //If pointer is greater than or equal to minimum screen address, pixel at pointer is
turned white,
    //and pointer is decremented. Unconditionally jumps to MAIN label (infinite loop).
    @pointer //go to screen pointer address
    D=M //get value (screen address) of pointer
    @SCREEN //get first screen value to compare with pointer location
    D=D-A //difference between pointer and top-left of screen (first screen address)
    @MAIN //will jump to MAIN label if pointer value is less than first screen address
    D;JLT //in order to prevent decrementing pointer value below first screen address (infinite
loop)
    @pointer //go to pointer address
    A=M //go to address with pointer value
    M=0 //set 16-bit register at pointer address equal to 0 in order to make white.
    @pointer //go to pointer variable address
    M=M-1 //decrement value of pointer
    @MAIN //address MAIN label
```

```
    0;JMP //unconditionally jump to MAIN label

(DRAW) //If pointer is less than or equal to max screen address, pixel at pointer is turned
black,
    //and pointer is incremented. Unconditionally jumps to MAIN label (infinite loop).
    @pointer //go to pointer variable address
    D=M //retrieve pointer value
    @KBD //go to address after end of screen
    D=A-D //number of pixels between end of screen and pointer
    @MAIN //if pointer value is greater than screen address (equal to KBD address),
    D;JLE //jump to main, allowing infinite loop without going beyond screen addresses
    @pointer //go to pointer variable address
    A=M //go to address with pointer value
    M=-1 //set 16-bit register at pointer address equal to -1 in order to make black.
    @pointer //go to pointer variable address
    M=M+1 //increment value of pointer
    @MAIN //address MAIN label
    0;JMP //unconditionally jump to MAIN label
```

FillAutomatic.tst Result



Fill.tst With Key Pressed (Filling)



Fill.tst With Key Released (Erasing)