Homework 5: Machine Language

Objective:

Build the two Assembly Language programs described in Chapter 4, which will test your understanding of Assembly programs for our HACK architecture. Highly recommend you go through the simulator tutorial for assembly programs before attempting the homework.

Grading method:

If the code passes *all* the tests described in the supplied test script, it receives 60% of the grade. 30% goes to it being well built (the shortest number of assembly code instructions), with the remaining 10% going towards documentation provided for each chip. Generally speaking, we prefer implementations that *use as few A- and C- instructions as possible,* documentation/commenting do not count towards that line count. Documentation will be in form of commenting AT LEAST EVERY line of assembly code, describing what the instruction is doing overall or to the registers. This may be over-commenting in high-level languages like Java, but is absolutely necessary here to understand the machine code.

Notes on "Fill":

In order to get all 55 points for the implementation, the program should provide a smooth user experience. In particular, the program must sample the keyboard every once in a while, and respond to "press" and "no-press" state changes effectively while minimizing annoying visual effects.

For example, suppose that a key was pressed, and the program starts coloring the screen in black. Then the key is released. In that case, good programs should stop coloring the screen in black, and proceed to color in white only the part of the screen that was already blackened.

Annoying visual imperfections may result in reducing up to 15 points from the grade.

What do you turn in?

Create one Word document (or PDF) with the following in order:

1. The **Mult.asm** source code (make sure to comment every line)
2. A screen shot containing the output from running the **Mult.tst** file agains your source code.
3. The **Fill.asm** source code (make sure to comment every line)
4. A screen shot containing the output from running the **Fill.tst** file against your source code.

Program	***Working?***	***Well built?***	***Documentation?***
Mult	20	/ 15	/ 5
Fill	40	/ 15	/ 5
Subtotal	60	/ 30	/ 10

See http://nand2tetris.org/04.php *for test scripts/output files (if you need clarification email your instructor. You will be graded based on this documents requirements).*