Pratik Gangwani
ECE 3056
Lab 1 Extra Credit Report

Results:

Val 0x4000	0xFF	0x00	0x000	0xFF	0xFF	0x01	0x12	0x64	0xAA	0x96
Val 0x4001	**0xFF**	**0x00**	**0xFF**	**0x00**	**0x01**	**0xFF**	**0x34**	**0x64**	**0x55**	**0x69**
Result at 0x4002	0xFE01	0x0000	0x0000	0x0000	0x00FF	0x00FF	0x03A8	0x2710	0x3872	0x3D86
Num Inst	781	13	16	13	781	19	70	316	526	466

The avg. # of instructions to perform MUL is: 3001/10 = 300.1 or ~ 301

To start the assignment I looked at the byteadd.asm provided to us to get back into writing assembly code. Reading through the code I realized that much of it was useful to our own assignment for writing a MUL function. As such, much of the beginning and end parts are the same as byteadd.asm; mainly the storing of the answer at the end, and retrieving the inputs at the beginning.

I chose to go with the repeated addition method to write the 'mul' function. It is inefficient compared to shifting the bits, but was much easier to implement and debug. The first time I wrote the function I used a counter and flipped it's bits and added 1 to get the value negative one, while repeatedly adding the first input to it self. I ended up having 393,234 instructions for 0x01 * 0xFF! I then rewrote the code using the second input minus a constant of negative one to control my loop and added in branches to check for whether the inputs themselves were zero. This cut my instructions in half and brought 0x01 * 0xFF to 19 instructions.