Homework 1
Out: September 8, 2015, Tuesday -- DUE: September 24, 2015, Thursday, 11:59pm
EC327 Introduction to Software Engineering – Fall 2015

Total: 100 points

This homework is based on the simplified assembly we discussed in class. A reference document for the simplified assembly is posted on Blackboard. **Please note: The programs are loaded such that the first command is in addresses 31 (decimal).** Your answers should take that into account, especially when converting loops from assembly to machine code.

For each of the following questions, provide in a .txt or .pdf document:

- **Assembly code.**
- **Machine code (in binary and hexadecimal format; don't forget addresses).**
- **Short comment explanation (used if your code does not work).**
- **The one-line string of code to enter into the simulator. (KEY! We start with this for grading)**

Here is an example of a submission:

```
Address          Machine code                      Assembly        Comments
31          1001 000000 000000 (Hex: 9000)         mov R0, 0       ;R0 holds 0
32          1001 000001 001010 (Hex: 904A)         mov R1, 10      ;R1 hold 10

Simulator Hex
9000904A
```

Try to use as few instructions as possible. Short descriptions/comments of the code will be examined in the event that your code does not function properly. **Be as clear and concise as possible in your descriptions**.

Resources:
Simulator w/Assembler webpage: http://cjwoodall.com/misc/sx86-emulator/
For more help see: "Simulator Overview" on Blackboard under HW1.

Suggestion: For easy conversion into hex or binary, google '<number> in binary' or '<number> in hex'. Like '63 in hex'. Google will give you the answer like: '63 = 0b111111', or '63 = 0x3F'. You should make sure you can do this by hand as well because frankly this is a basic computer engineering skill.

Submission

1. Completed assignments **must** be submitted via Blackboard. (set up by 9/11)
2. Write your answers clearly in your favorite text editor, and submit a **.txt or .pdf** file for each question (Q1.txt, Q2.txt, Q3.txt, Q4.txt) ZIPPED into a file named
    a. **<BU Username>_HW1.zip** For example: **dougd_HW1.zip** -> Has Q1.txt Q2.txt Q3.txt Q4.txt

Late Lab Submissions:

3. For such submissions, you should use the following format for the zip file:
   Example: **dougd_HW1_Late.zip**

Failure to follow any of these guidelines **WILL** result in loss of points on the assignment.

Q1. [25 points]
Put numbers 11 to 20 in memory cells 10 to 19, respectively (i.e., cell 10 contains 11, cell 11 has 12, etc.). Increment the value of memory cell 19 twice.

Q2. [25 points]
Write code that computes ((a + (b – c)) * d), where a, b, c, d are the values in memory locations (decimal) 1, 2, 3, and 4 respectively. The result should be in memory location 30 (decimal). *Note b ≥ c can be assumed. Also take into account d may equal 0.

Note: assume 4 numbers are stored in memory locations 1-4. Use first 12 lines of the code to store your numbers (these will be replaced with our test examples). For example, use the following code to store number 3 at memory location 1:

```
31      1001_0000_0000_0001 = 0x9001   mov  R0, 1     ;; R0 stores 1
32      1001_0000_0100_0011 = 0x9043   mov  R1, 3     ;; R1 stores 3
33      1011_0000_0000_0001 = 0xB001   mov [R0], R1   ;; Store number 3 in location 1
```

which is: 90019043B001

To store the number 4 at memory location 2, use: 90029044B001, etc.

So your code should start at memory location decimal 43 (that is 31+12 instructions to store 4 numbers). Please provide only your calculation code (i.e. not the 12 instructions that store values) that starts from line 43. We will be adding our own initial 12 store instructions.

Q3. [25 points]
Consider all the numbers are unsigned integers. Write code that calculates: n^m, where n is the value in memory location 1 and m is the value in memory location 2. You should save the result in memory location 30). Use the first 6 lines of code to store these two values (as explained in Q2). So your code should start at location 37.

Q4. [25 points]
Put 55 in memory cell 10 and 24 in memory cell 9. Write code such that starting memory cell 8 down to 1, each memory cell has the *sum* of the values in the previous two memory cells (adding the two values together) as shown in the following table. Use the first 6 lines of code to store 55 and 24 (as explained in Q2). So your code should start at location 37.

Mem. Cell	Value
1	?
. .	. .
8	79
9	**24**
10	**55**

**Note that your code should compute the sum at each step (i.e., you should not hard-code numbers).*