Brown University
School of Engineering
EN1640 Computer System Design
Professor Sherief Reda
Assignment 04 (100 points)
Due Date: Friday April 1 by 12:00 pm on Canvas.

Canvas submission: Submit one zip file with your name and lab number; for example, lab04_john_smith.zip. The zip file should have (1) the project archive files .qar (click on Project from the menu and the Project archive) for all problems and (2) the assignment report in pdf. Name your .qar files using the lab number and problem number as follows. For instance, the first question would have the submission lab04q01.qar. Please refer to the videos from 1630 to refresh on how to use the Quartus tool and Verilog.

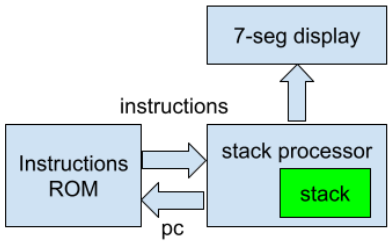
(1) [100 points] Your objective is to design a programmable stack-based calculator, which is also known as a postfix calculator, that can do 8-bit integer arithmetic. You have to include five instructions:

Instruction	**Meaning**
push *value*	push the immediate *value* into the top of the stack.
add	pops the two top elements in the stack and push the result of the addition to the stack.
sub	pops the two top elements in the stack and push the result of the subtraction to the stack.
mult	pops the two top elements in the stack and push the result of the multiplication to the stack.
halt	**Stops execution and displays the top of the stack on the 7 segment display in decimal.**

Here are two examples that show the operation of the calculator.

Example A: 8-5*3	**Example B: 5 + ((1 + 2) * 4) – 3**
push 8 push 5 push 3 mult sub halt	push 5 push 1 push 2 add push 4 mult add push 3 sub halt

A. Design an instruction set format of this machine. Include this design and encoding in your report.
B. Translate the two example assembly programs into machine code using the format you created in part (A). Include in your report the machine code of examples A and B.
C. Write the Verilog code for this machine. Feel free to leverage code you write in lab02 towards this lab. Assume the following:
    - The machine has a stack of exactly 10 8-bit entries (users should never create programs that require more than 10 items in the stack).
    - One LED on the board will turn on if there is an arithmetic overflow (only care about addition and subtraction overflow). Feel free to use the ALU code you developed in Assignment02.
    - When the processor boots, it starts executing instructions from address 0 until a halt is encountered. That is, your PC should be initialized to 0. The processor always displays the top of the stack on the 7 segment display in decimal. Feel free to use the code you developed in ENGN1630 to display numbers on the 7-seg decimals.
    - The processor should stop executing when a halt instruction is encountered. At that point the top of the stack should be the final result and it is displayed on the 7-segment display.
    - The machine code for programs is stored in a ROM, which is conceptually presented in the figure on the right. You will need to instantiate a ROM from the Tools->IP Catalog function in Quartus. Make sure the output of your ROM is not clocked. Initialize the ROM's content from a .mif file. The .mif file can be created and edited by choosing File -> New -> Memory Files -> Memory Initialization File. Interface the ROM to your CPU module using appropriate address, data and control buses. **For your Verilog testbench**, the "ROM" will be part of your testbench stored in a simple Verilog array and your testbench will supply the instructions over the instruction bus to the processor and will receive the output directed for the 7-segment displays, which you can then monitor. You can also monitor other internal signals in your processor for debugging. If you like to view waveforms in ModelSim, select View -> Wave from modelsim menu and then drag and drop your variables in the waveform window.



D. Please report:
    - your Verilog testbench and its simulation results using $display or $monitor outputs giving the results from the execution of the two examples;
    - the total ALM resources using by your circuit;
    - RTL circuit view;
E. **10 points of this question will be determined upon your final design size in terms of LEs. You should not use any RAM/ROM elements to substitute for logic functions implemented in ALMs. Optimize your design to achieve <= 67 ALMs. Document all the optimization you have done to achieve that value.**