CSE 141L Lab #3: 8-bit CPU
due Friday, November 12

In this assignment, you will finally design a single-cycle implementation of a processor to execute your 8-bit ISA. At a minimum, your design will have a program counter (PC), a PC incrementer, an ALU, memory, and probably some internal storage. The ALU and internal storage should not be significantly changed from the last lab. The CPU design will execute the programs you wrote (hopefully correctly) for Lab 1.

Some things to keep in mind for this lab:

- Use hierarchical design (subcircuits) to make your design easier to understand and think about. The highest-level schematic should mostly be functional elements (register file, alu, memory, etc.) and wires/buses.
- Isolate control. Follow the text's lead by generating all control signals in one place from the opcode. This should make the design easier conceptually, at least.
- You will create a ROM to hold instruction memory and a RAM to hold data memory. Both will be initialized – the ROM(s) to hold your programs, the RAM to hold your input data. Instructions to create memory parts in Xilinx will be on the web page.
- You can actually have three ROMS for your three programs, and swap them in and out of your design as needed. Otherwise, you could have a single ROM with three different initial values for the PC.
- I want you to have an init signal that sets the PC to some predetermined value, possibly initializes some memory locations, etc. I also want a done signal that goes high at the halt instruction.
- I want you to have a cycle counter that we will use to reliably determine dynamic instruction counts. It should initialize to 0 at the init signal, and count up until the done signal goes high.
- Keep in mind that you may have to debug your programs! Think about how to make your life easier before it happens.
- In the questions for the lab report, I am no longer looking for you to convince me that your design is wonderful (unless it is), but rather I am looking at how effectively you critique your own design.
- Remember that it is your responsibility to convince us that your CPU works, not our responsibility to figure that out ourselves. Providing sufficient and clear results and information is crucial.

**What you turn in:**

- A review of your ISA.
- Schematics of all circuits (including those presented in lab 2), hierarchically organized. The highest-level design needs to have all of the signals necessary to demonstrate correct program execution via the timing diagram.
- A timing diagram for each program demonstrating correct operation and other important data. It should at a minimum show results being generated (e.g., the accumulating sum of ones, etc.), the cycle counter, the PC. It need not show the whole execution of the program (particularly if it takes over 100 cycles or so!), but certainly the beginning and end and some execution of the main loop. It, once again, should be heavily annotated so we can figure out what is going on.
• The assembly and machine code for your three programs.
• Answers to the following questions:
  1. Have you made any changes to your ISA? What were they? Why did you make them?
  2. What are your dynamic instruction counts for program 1? program 2? program 3?
  3. What could you have done differently to better optimize for dynamic instruction count?
  4. How successful were you at optimizing for ease of design? Give examples.
  5. What could you have done differently to better optimize for ease of design?
  6. How easy/difficult would it be to extend your design to a multicycle implementation? A pipelined implementation? Give examples.
  7. What might you have done differently if a priority was ease of programming? Give examples.
  8. What instruction takes longest on your machine (and thus would set the cycle time)? (Use rough estimates, e.g. assume each device introduces a constant delay).
  9. What might you have done differently if a priority was short cycle time? (again, use rough estimates). Give examples.

The inputs
You can use the same input for all three programs. In fact, you can use the modified output after program 1 as input to program 2, etc., as long as the following still hold:

Memory location values:
Address (dec)  value (hex)
0        A8
1        D7
2        00
3        00
32       39
33       3A
34       0F
35       12
36       17
37       32
38       12
39       2C
40-47    same as 32-39
48-55    same as 32-39
56-63    same as 32-39
64-95    the address (i.e., 64, 65, 66… decimal, or 40, 41, 42… hex)