CS1428 Lab 08h

# Name: Section:

In this lab you will continue to developing a VERY simple programming language for a VERY simple processor. You will need to download and rename your CPP file from last lab. If you need starting code, download

1. Now that every instruction is in the array, each instruction is associated with an index. With the for-loop keeping track of what index we are currently processing, we can have a NEW instruction that *modifies* the index variable to move it back or forward. We will call this instruction **jump** because it jumps around the array.

The form of the jump instruction is: **8 offset 0 0** where offset is how far ahead to jump or how far back to jump (if negative). For example if we are currently at address 5 and we are jumping to offset -2, our new current address is 3. **Watch out for the for-loop, it will automatically increment the for-loop index variable after the switch statement runs!** Write a case statement to handle jump.

1. To make full use of our jumping facilities we need a *conditional jump* instruction. This is just like jump, but the jump only happens if two things are equal. The form of the conditional jump instruction is: **9 offset a b** where again offset is where to jump and a/b are the **indices** of the memory cells being compared. If the two memory cells hold the same value, then the jump is performed, otherwise nothing happens. Write the case statement to implement conditional jump.
2. Using the segments you have written above and what you did in last lab, combine them into a single program (**lab07h.cpp**) that:
   1. Reads the program all at once into memory
   2. Goes through instructions array one at a time
   3. Properly handles the new programs with jumps and conditional jumps
   4. Properly handles the programs from last time
   5. **IMPLEMENTS EACH INSTRUCTION AS ITS OWN FUNCTION** 
      1. You get to determine the proto-types, parameters and return values, but every instruction must be implemented as its **own function**.
      2. The loading of the program into the program array must be implemented as its own function with the following prototype:  
         **int loadProgram(string filename, int programArray[512][4]);**  
         Return value is number of instructions loaded, 0 if failure.

**(\*\*Make sure to include the standard header and to name the file correctly\*\*)**