Problem Statement

1. Use Logisim to implement the instruction set and circuit below including a pipeline for Fetching, Decoding, and Executing your instructions; and including the design of a Finite State Machine in your control unit logic to drive your pipeline.

2. Design a piece of software to embed using your instruction set

3. Use an 8051 microcontroller simulator to perform the exact same task as your machine's embedded code; or some alternative assembly language execution (ARM, Motorola, etc.) Microcontroller or Microprocessor.

4. Compare the performance of your machine to that of at least one existing assembly language

New Element: Stack

The stack fetches the operands based on the clock, which then uses a decoder to properly activate the next correct instruction set, which then executes. The stack was designed with a 5-bit decoder and a series of OR gates with a splitter and a control buffer.

Comparison with the 8051 Microcontroller

We used an 8051 simulator in order to compare our Logisim circuit to machine code that performs the same exact task. When doing the programming in machine code there were no counters available, so in order to simulate a counter we had to increment and decrement two registers after each instruction was completed. This adds quite a bit of lines to the code, which may result in the 8051's simulation to be slower than it potentially could be. Also many of the functions would not let registers R0-R7 directly interact with each other so we had to move the value of the registers needed to ones that could interact directly, perform the operation and then store the result in a separate register. Doing all of these move functions also increases the time it will take to run through the simulation. Another small difference between the machine code and the Logisim simulation is the number of registers in the register bank, as there are 8 in the Logisim simulation while there are only 6 in the 8051 simulator (due to limitations of the program). This however should not have a significant impact on how fast the entire set of instructions will take to complete.

8051 Code for an ADD operation:

```
0000 | INC IE;
0002 | DEC IP;
0004 | MOV A, R0;
0005 | INC IE;
0007 | DEC IP;
0009 | MOV B, IE;
000C | INC IE;
000E | DEC IP;
0010 | ADD A, B;
0012 | INC IE;
0014 | DEC IP;
0016 | MOV R0, A;
```

Acknowledgments

A Special Thanks to:
Dr. Joseph Wunderlich