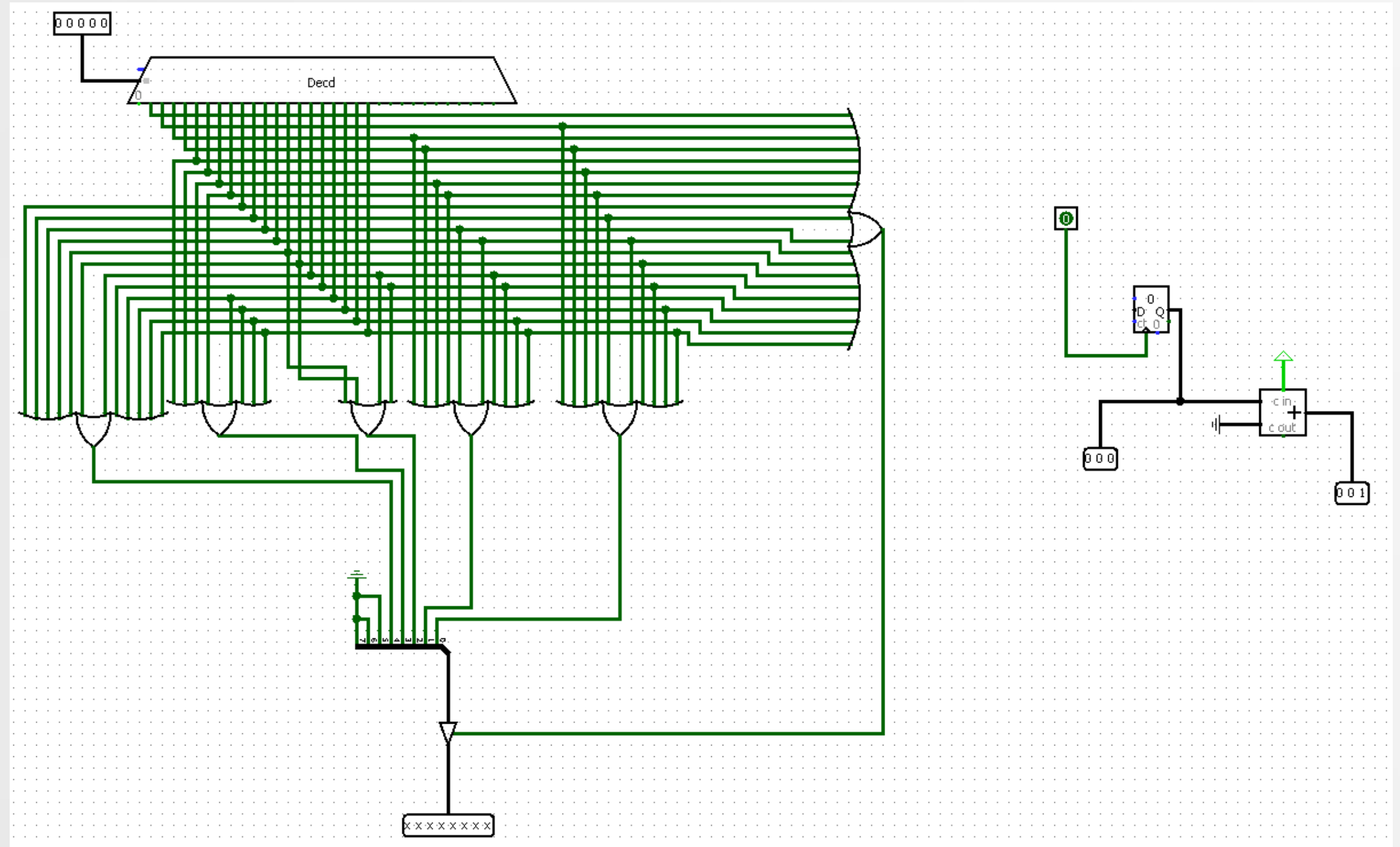
Computer Design Competition

Anthony Bird, David Cain, Ryan O'Connor
Dr. Wunderlich

Problem Statement

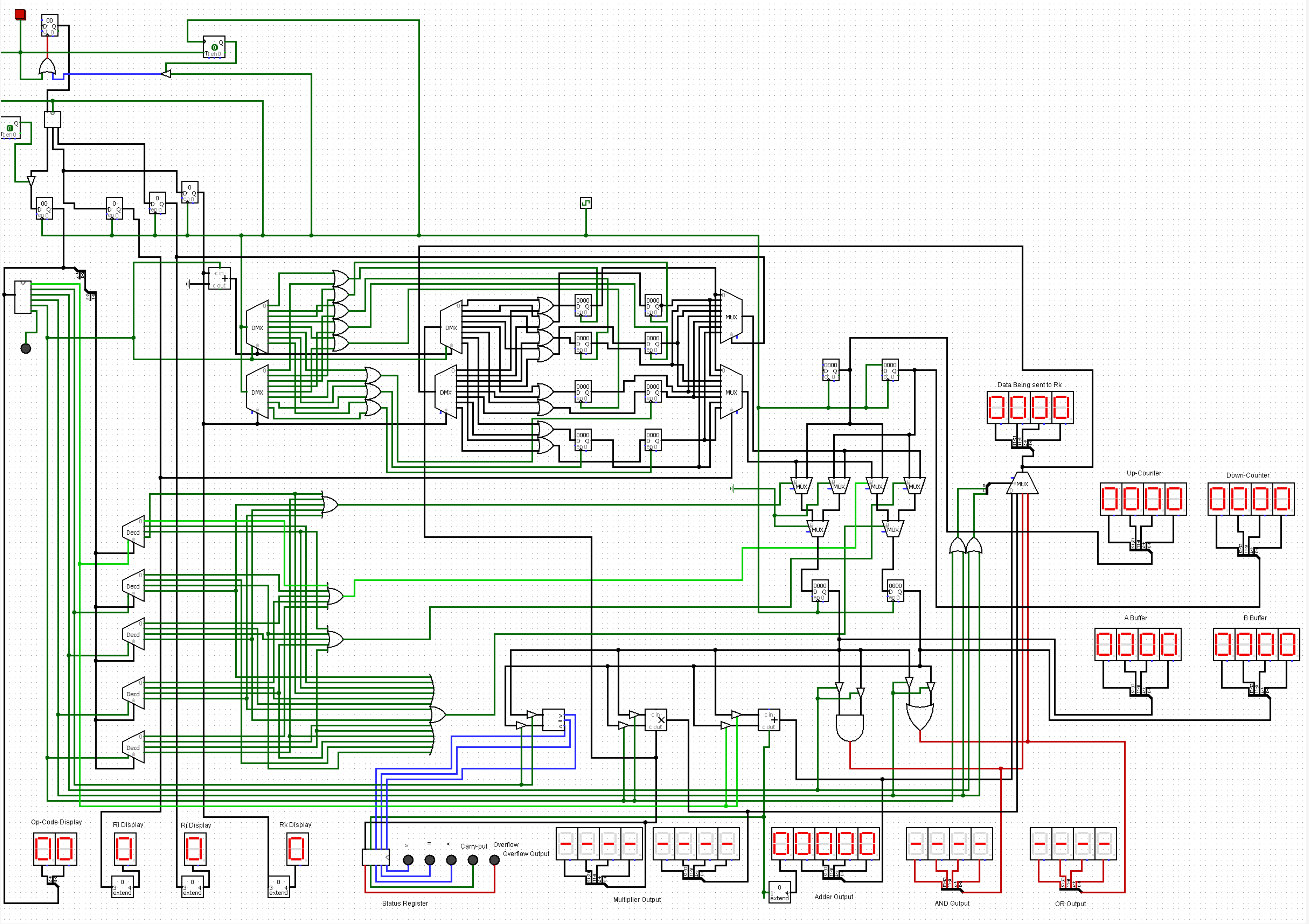
1. Use Logisim to Implement the instruction set and circuit below including a pipeline for Fetching, Decoding, and Executing your instructions; and including the design of a Finite State Machine in your control unit logic to drive your pipeline.
2. Design a piece of software to embed using your instruction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

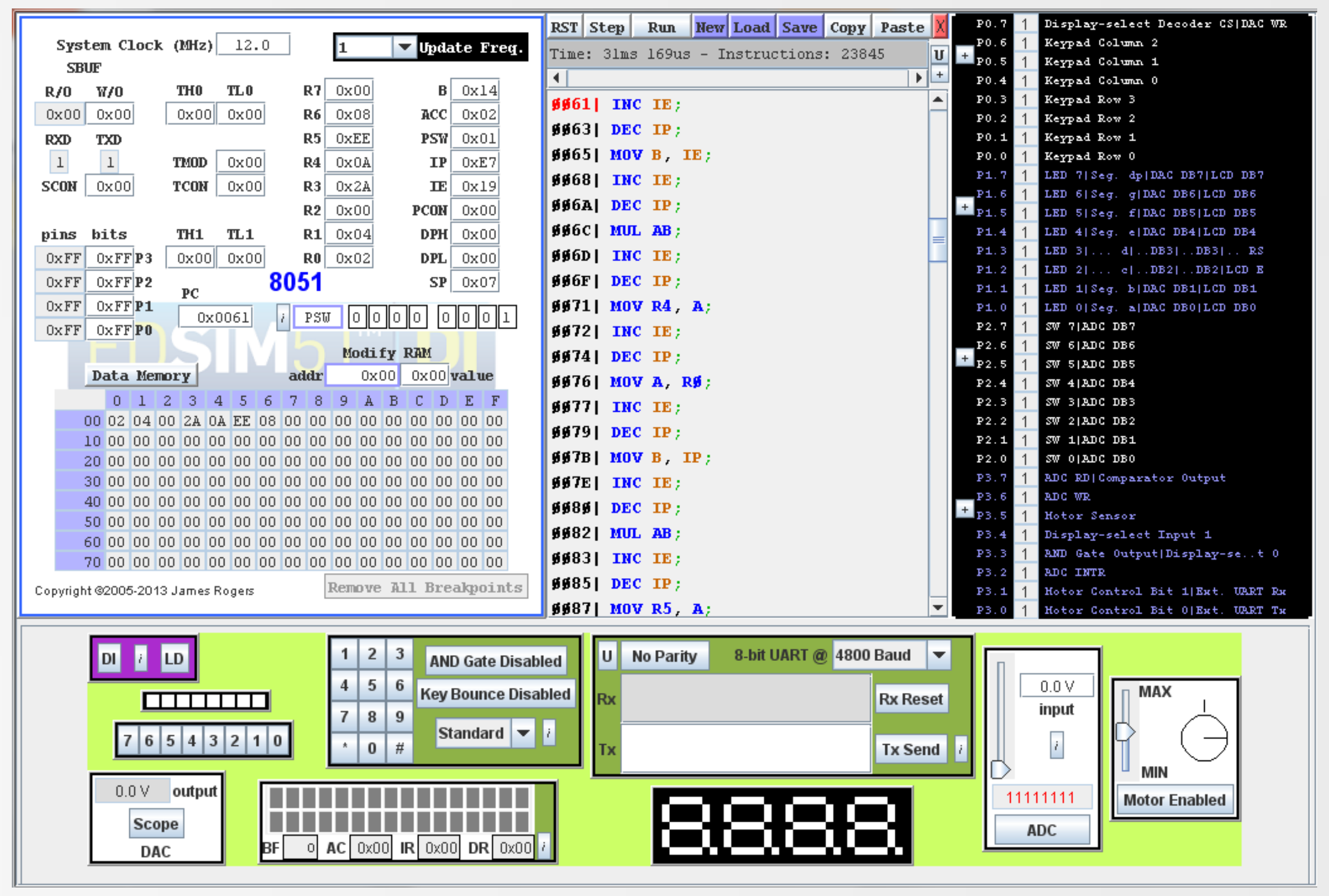
Circuit Overview



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.

Simulation



```
Time: 31ms 169us - Instructions: 23845
0061| INC IE;
0063| DEC IP;
0065| MOV B, IE;
0068| INC IE;
006A| DEC IP;
006C| MUL AB;
006D| INC IE;
006F| DEC IP;
0071| MOV R4, A;
0072| INC IE;
0074| DEC IP;
0076| MOV A, R0;
0077| INC IE;
0079| DEC IP;
007B| MOV B, IP;
007E| INC IE;
0080| DEC IP;
0082| MUL AB;
0083| INC IE;
0085| DEC IP;
0087| MOV R5, A;
```

```
R7 0x00   B    0x14
R6 0x08   ACC  0x02
R5 0xEE   PSW  0x01
R4 0x0A   IP   0xE7
R3 0x2A   IE   0x19
R2 0x00   PCON 0x00
R1 0x04   DPH  0x00
R0 0x02   DPL  0x00
          SP   0x07
PC 0x0061
```

```
P0.7 1 Display-select Decoder CS|DAC WR
P0.6 1 Keypad Column 2
P0.5 1 Keypad Column 1
P0.4 1 Keypad Column 0
P0.3 1 Keypad Row 3
P0.2 1 Keypad Row 2
P0.1 1 Keypad Row 1
P0.0 1 Keypad Row 0
P1.7 1 LED 7|Seg. dp|DAC DB7|LCD DB7
P1.6 1 LED 6|Seg. g|DAC DB6|LCD DB6
P1.5 1 LED 5|Seg. f|DAC DB5|LCD DB5
P1.4 1 LED 4|Seg. e|DAC DB4|LCD DB4
P1.3 1 LED 3|... d|..DB3|..DB3|.. RS
P1.2 1 LED 2|... c|..DB2|..DB2|LCD E
P1.1 1 LED 1|Seg. b|DAC DB1|LCD DB1
P1.0 1 LED 0|Seg. a|DAC DB0|LCD DB0
P2.7 1 SW 7|ADC DB7
P2.6 1 SW 6|ADC DB6
P2.5 1 SW 5|ADC DB5
P2.4 1 SW 4|ADC DB4
P2.3 1 SW 3|ADC DB3
P2.2 1 SW 2|ADC DB2
P2.1 1 SW 1|ADC DB1
P2.0 1 SW 0|ADC DB0
P3.7 1 ADC RD|Comparator Output
P3.6 1 ADC WR
P3.5 1 Motor Sensor
P3.4 1 Display-select Input 1
P3.3 1 AND Gate Output|Display-se..t 0
P3.2 1 ADC INTR
P3.1 1 Motor Control Bit 1|Ext. UART Rx
P3.0 1 Motor Control Bit 0|Ext. UART Tx
```

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