CSE 372
Lab Assignment #5

Objective:

In this lab, you will design a single cycle microprocessor loosely based on the design given in the course text.

Instructions:

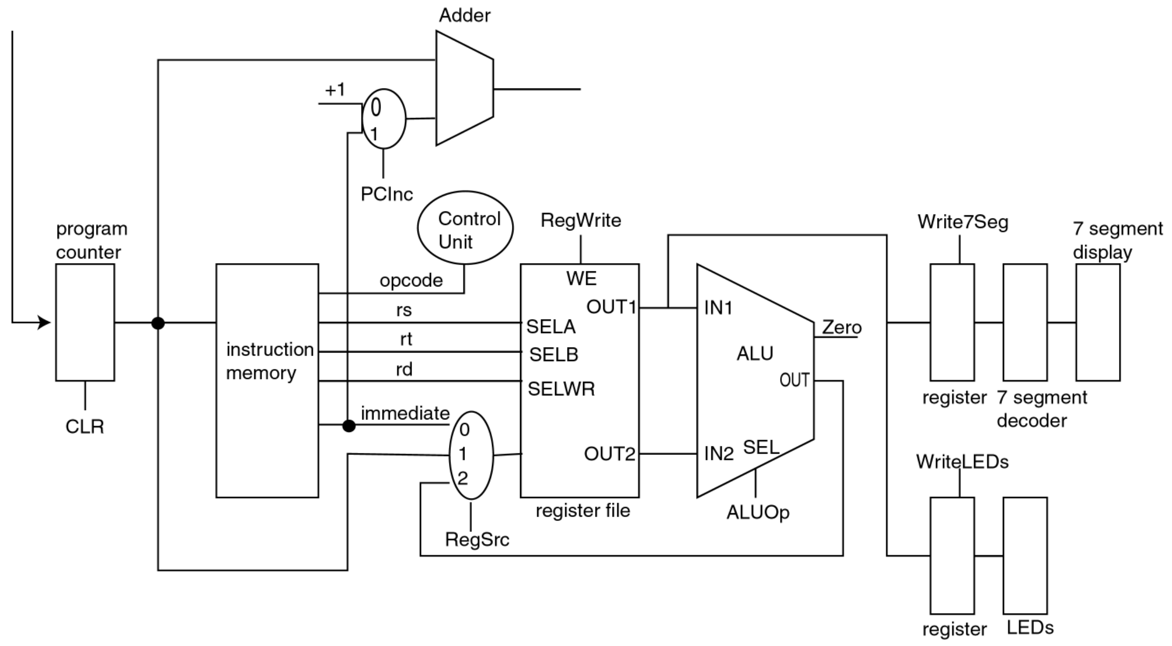
You will be designing a microprocessor that executes 15 different instructions. Each instruction is 21 bits long and consists of the following five fields: a 4-bit opcode, two 3 bit fields rs and rt that denote operand registers, one 3-bit field rd that denotes the destination register and an 8 bit immediate field. Note that although all of the instructions share the same layout most instructions do not use all of the available fields.

20 17	16 14	13 11	10 8	7 0
Opcode 4	rs 3	rt 3	rd 3	Immediate 8

The instructions that the processor will execute are given below:

1. **add rs rt rd**: (opcode = 0) Add the contents of the two registers denoted by the rs and rt fields and store the result in the register indicated in the rd field
2. **sub rs rt rd**: (opcode = 1) Subtract the contents of the register denoted by the rt field from the register denoted by the rs field and store the result in the register indicated in the rd field
3. **and rs rt rd**: (opcode = 2) And the contents of the two registers denoted by the rs and rt fields and store the result in the register indicated in the rd field
4. **or rs rt rd**: (opcode = 3) Or the contents of the two registers denoted by the rs and rt fields and store the result in the register indicated in the rd field
5. **xor rs rt rd**: (opcode = 4) Xor the contents of the two registers denoted by the rs and rt fields and store the result in the register indicated in the rd field
6. **not rs rd**: (opcode = 5) Invert the contents of the register denoted by the rs field and store the result in the register indicated in the rd field
7. **shl rs rd**: (opcode=6) Shift the contents of the register denoted by the rs field left one position (filling the vacated slot on the right with a zero) and store the result in the register indicated in the rd field
8. **shr rs rd**: (opcode=7) Shift the contents of the register denoted by the rs field right one position (filling the vacated slot on the left with a zero) and store the result in the register indicated in the rd field
9. **ld rd i**: (opcode=8) Load the register denoted by the rd field with the contents of the immediate field of the instruction
10. **spc rd**: (opcode=9) Store the current contents of the 8 bit program counter in the register indicated in the rd field
11. **beq rs rt i**: (opcode=10) If the contents of the two registers denoted by the rs and rt fields are equal then increment the program counter with the value in the immediate field
12. **ji i**: (opcode=11) Reset the program counter to the value found in the immediate field
13. **jr rs**: (opcode=12) Reset the program counter to the value stored in the register denoted by the rs field
14. **w7seg rs**: (opcode = 13) Write the contents of the register denoted by the rs field into another 8-bit register connected to the seven segment displays
15. **wleds rs**: (opcode = 14) Write the contents of the register denoted by the rs field into another 8-bit register connected to the 8 LEDs

The figure given below shows a partial datapath for the processor you will implement.



Question 1: For each of the 15 instructions indicate how the **next** value of the program counter (PC) should be determined.

Question 2: The conditional branch instruction requires us to compare the contents of two registers using the ALU. What ALU operation could we use to implement this comparison? Are there any other options?

Question 3: The PCInc signal controls whether the current PC value is added to the constant 1 or the immediate field of the current instruction. How should this signal be set during a **beq** instruction?

Based on your answers to these questions complete the datapath for the processor adding in any required datapath elements and control signals. Turn in a diagram of the completed datapath.

In order to complete the design of this processor we must provide an implementation of the control unit. The control unit is a combinational circuit which chooses the settings of all of the control lines in the circuit based on the opcode of the current instruction.

(N.B. The control signals labeled Write7Seg and WriteLEDs control whether the respective registers will get updated on the next rising edge)

Question 4: Provide a table which indicates what the settings of **all** of the control signals in your circuit should be for each of the 15 instructions.

Design an ABEL macro that implements the functionality of this control block and add it to your schematic.

In addition to the control signals which are internally generated by the control unit, two external signals must be provided: clock and reset. The clock signal must be fed to all registers as in every sequential circuit. The reset signal is used to ensure that on power up, the processor starts the execution of the program from some predefined address, which will be zero on this processor.

Question 5: Which register must be initialized by the reset signal ? Is it necessary to initialize other registers, and if not what would be the benefit of doing it, and what would be the drawback ?

Use the lowest frequency of the on-chip oscillator as the clock source. Use one of the DIP switches as the source for the RESET input. Hold the device in reset (keep the reset switch in HI position) while downloading. Move the reset switch to LO position to start executing your program.

Consider the following Assembly program:

```
Address      Code              Instruction
0000         100000            ld R0,0
0001         100101            ld R1,1
0002         100201            ld R2,1
0003         1C8000            wleds R2
0004         100307            ld R3,7
0005         0C8200      loop1 shl R2, R2
0006         1C8000            wleds R2
0007         000800            add R0, R1, R0
0008         141802            beq R0, R3 loop2
0009         160005            ji loop1
000A         0E8200      loop2 shr R2, R2
000B         1C8000            wleds R2
000C         020800            sub R0, R1, R0
000D         1408F8            beq R0, R1, loop1
000E         16000A            ji loop2
```

Question 6: What does this program do?

To implement the instruction memory, use the ABEL code provided below. Create a macro and paste it into the schematic of your processor. Based on the address provided on inputs `a[7..0]` this component will place the appropriate content of the memory location on the output `d[20..0]`. In other words, the ABEL code provided below will emulate the memory that stores your program.

```
module simmem
Title 'simmem'

Declarations

a7..a0 PIN;
a = [a7..a0];
d20..d0 PIN istype 'com';
d = [d20..d0];

Equations

Truth_table (a -> d)
      0  -> ^H100000;
      1  -> ^H100101;
      2  -> ^H100201;
      3  -> ^H1C8000;
      4  -> ^H100307;
      5  -> ^H0C8200;
      6  -> ^H1C8000;
```

```
        7  -> ^H000800;
        8  -> ^H141802;
        9  -> ^H160005;
        10 -> ^H0E8200;
        11 -> ^H1C8000;
        12 -> ^H020800;
        13 -> ^H1408F8;
        14 -> ^H16000A;

end simmem
```

Add this macro to your circuit, then simulate, implement and demonstrate the resulting circuit.

Make sure that your writeup includes answers to all of the questions on this laboratory along with any schematics and ABEL code you used in your implementation. You must turn in a timing diagram showing how your processor performs on the test code and demonstrate a working version of your system to a TA.