RAT Assignment #5
Control Unit / RAT MCU
It's Alive!

Learning Objectives:

- ✓ To understand the basic functionality and implementation (FSM) of a MCU control unit
- ✓ To understand the basics of fetch/execute instruction cycles
- ✓ To understand how to set control signals for a given instruction

General Notes:

The overall purpose of this experiment is to assemble the various RAT modules you worked with in the previous few experiments into a working computer. Though this computer will be able to execute an actual RAT assembly language program, it will only use a limited subset of the available RAT instructions. This computer will only be a subset of the final RAT computer, as you'll be leaving out several important modules in order to reduce the scope of this experiment; you'll be adding those modules in the next experiment.

This experiment has many important aspects, none of which is by any means trivial. While previous experiments involved assembling individual RAT modules, this experiment requires a true systems-level approach in order to successfully complete the assignment. Understanding the RAT at a systems level, including the interface to actual hardware, will provide you with a complete understanding of the internal workings of the RAT and computers in general.

The Control Unit

The control unit is a FSM that provides control signals to the various RAT modules. Recall that all of the RAT modules had control inputs; these inputs are the outputs of the control unit. The control unit provides the proper sequencing of modules operations in order to generate a working RAT computer.

1) Complete the control signal table (RAT_MCU_ControlSignal_Table.xls on Polylearn) for the following:
    a. Instructions: IN, MOV, EXOR, OUT, BRN
    b. States: FETCH, INIT, EXEC

    If any signals are "don't cares" for a given instruction/state, enter them as '0'. Note that you may need to change signal names to match those in the architecture diagram and control unit you use.

2) Transfer your signal values from the spreadsheet to your skeleton control unit. To avoid latches, you need to make sure every output signal from the control unit gets set for every case. The easiest way to assure this is to always initialize every signal to '0' and then change only those that matter based on the instruction (opcode).

3) Enter the appropriate values for EACH OUTPUT signal for the states Fetch and RST. As with (a), note that you only need to match the first five bits of the opcode values for the reg/imm type instructions instead of all seven bits.

The RAT MCU

Figure 1 shows the black box diagram for the RAT_MCU module (the module that connects all the components you've made so far). Use the provided RAT architecture diagram to design the VHDL architecture for this structural module. Note: the minimum modules you'll be port mapping for this lab include: the control unit, the program counter, the prog_rom (you'll generate this in part 3), the register file, and the ALU.

You will also need to implement the flag registers based on the simpler model from the architecture diagram. These can be made individually as 1 bit registers or D flip flops. These can be port mapped individually or wrapped in a FLAGS wrapper and port mapped into the RAT MCU.

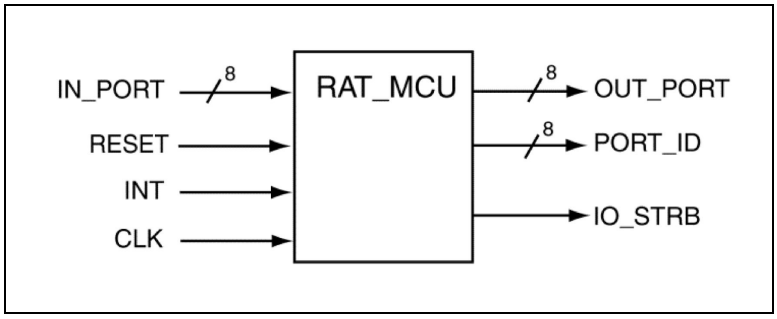

Figure 1: Black Box Diagram for RAT_MCU

The prog_rom

Recall the prog_rom.vhd is a model of a read only memory (ROM) that is automatically generated by the assembler upon successfully assembling a RAT assembly language program.

1) Simulate the following assembly program with the RAT simulator and add the resulting prog_rom.vhd file to your Xilinx project. Run through the simulator multiple times in order to obtain an understanding of the program's functionality.

```
;-------------------------------------------------------------------
;- Port and Memory Constants
;-------------------------------------------------------------------
.EQU SWITCH_PORT = 0x20 ; port for switch input
.EQU LED_PORT    = 0x40 ; port for LED output

.CSEG
.ORG 0x10

main:  IN    r10,SWITCH_PORT
       MOV   r11,0xFF
       EXOR  r10,r11
       OUT   r10,LED_PORT
       BRN   main
```

Assignment

The RAT report can be divided into 3 parts.

1. Verify the control unit works properly for the implemented instructions.
2. Verify the C and Z flag registers works properly
3. Verify the RAT_MCU executes the given prog_rom correctly. Also verify the RST input works as expected. The INT_IN can be kept at '0' because the hardware has not been made to support it yet. Show the pertinent signals internal to the RAT MCU for verification purposes.