Computer Organization
Lab 3: Single Cycle CPU - Complete Edition

Due: 2022/5/23

1. Goal

Based on Lab 2, you need to add a memory unit and implement a complete single cycle CPU that can run R-type, I-type and jump instructions.

2. Homework Requirement

a. Please use Vivado as your HDL simulator(if the execution result in your environment is different from ours, you need to bring your laptop to the lab and demo it to us).
b. Please **attach student IDs as comments** at the top of each file.
   The file type of your report should be **PDF**.
c. Please add the files listed below into one directory named "your_student_id", and zip it as " your_student_id.zip".
   The file structure in this lab should be (for example, id=310551072):

```
310551072/
├── Adder.v
├── ALU_Ctrl.v
├── ALU.v
├── Decoder.v
├── MUX_2to1
├── report_310551072.pdf
├── Shift_Left_Two_32.v
├── Sign_Extend.v
└── Simple_Single_CPU.v
```

d. Please **do not add unnecessary or given files and folders** (like .DS_Store, __MACOSX).
e. Program Counter, Instruction Memory, Data Memory, Register File and Testbench are supplied. (don't need to modify these files).
f. REGISTER_BANK [29] represents the stack pointer register value (initially 128). Other registers are initialized to 0.
g. You may add more control signals to the decoder:
   i. Branch_o
   ii. Jump_o
   iii. MemRead_o
   iv. MemWrite_o
   v. MemtoReg_o

h. **Basic Instructions: the following instructions have to be executed correctly in your CPU design, and we may use some hidden cases to further evaluate your design (40%).**

(For those who can't read testcase txt files after adding the testcase into simulation sources, please change the relative path in instr_memory file to absolute path.)

Your CPU design has to support the instruction set from Lab 2 + the following instructions:

i. lw (load word)

Reg[Rt] ← Mem[Rs+Imm]

6'b100011	Rs[25:21]	Rt[20:16]	Imm[15:0]

ii. sw (store word)

Mem[Rs+Imm] ← Reg[Rt]

6'b101011	Rs[25:21]	Rt[20:16]	Imm[15:0]

iii. jump

PC ← { PC[31:28], address<<2 }

6'b000010	Address[25:0]

i. **Advanced Instructions: the following instructions have to be executed correctly in your CPU design, and we may use some hidden cases to further evaluate your design (40%).**

i. jal (jump and link)

In MIPS, the 31th register saves return address for function calls.

Reg[31] ← PC+4

PC ← { PC[31:28], address<<2 }

6'b000011	Address[25:0]

ii. jr (jump register)

In MIPS, you can use

jr r31

to jump to the return address linked from **jal** instruction.

PC ← Reg[Rs]

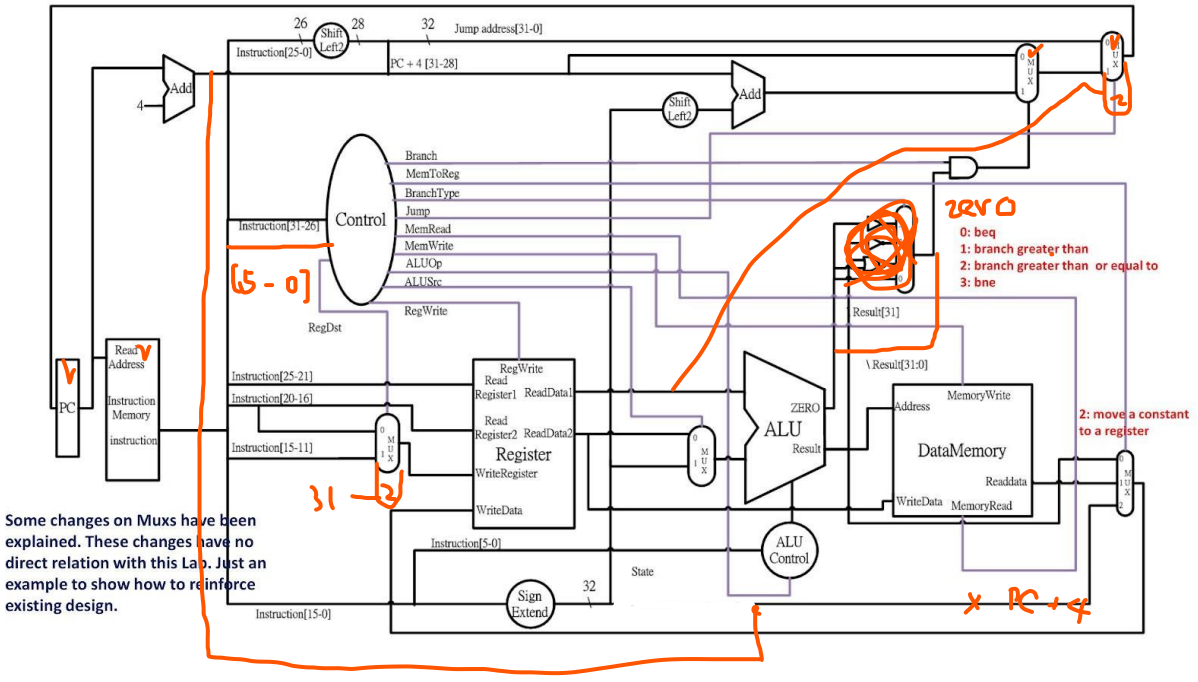
6'b000000	Rs[25:21]	0	0	0	6'b001000

j. Test program

The second testbench CO_P3_test_data2.txt is a Fibonacci function. **r2** represent the final answer. Please refer to **test2.txt**.

3. Architecture Diagram

This is a reference design. **You need to modify this design to meet the requirements**.



4. Report

a. Architecture Diagram
b. Hardware Module Analysis
c. Finished part
d. Problems you met and solutions
e. Summary

5. Grade

a. **Total:** 100 points (plagiarism will get 0 point)
   - **Report:** 20 points (please use **pdf format**)
   - **Hardware design**: 80 points
b. **Late submission:** Score * 0.8 before 5/30. After 5/30, you will get 0.
c. **Wrong format:** 10 points punishment

6. Q&A

If you have any question, it is recommended to ask in the Facebook discussion forum.