Computer Organization
Lab 3: Single Cycle CPU - Complete Edition

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 ModelSim or Xilinx as your HDL simulator.
b. Please **attach student IDs as comments** at the top of each file.
c. Please zip the Verilog files and the report and **name it as "ID.zip"**.
d. Program Counter, Instruction Memory, Register File and Testbench are supplied.
e. REGISTER_BANK [29] represents the stack pointer register value (initially 128). Other registers are initialized to 0.
f. You may add more control signals to the decoder:
    i.   Branch_o
    ii.   Jump_o
    iii.   MemRead_o
    iv.   MemWrite_o
    v.   MemtoReg_o
g. A brief review of different types of MIPS instructions:
    i.   R-type

Op[31:26]	Rs[25:21]	Rt[20:16]	Rd[15:11]	Shamt[10:6]	Func[5:0]

    ii.   I-type

Op[31:26]	Rs[25:21]	Rt[20:16]	Immediate[15:0]

    iii.   Jump

Op[31:26]	Address[25:0]

h. Basic Instructions: the following instructions have to be executed correctly in your CPU design (50%).

(For the ones who can't read testcase txt files or dump result txt file, please change the relative path in testbench and 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.   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]
```
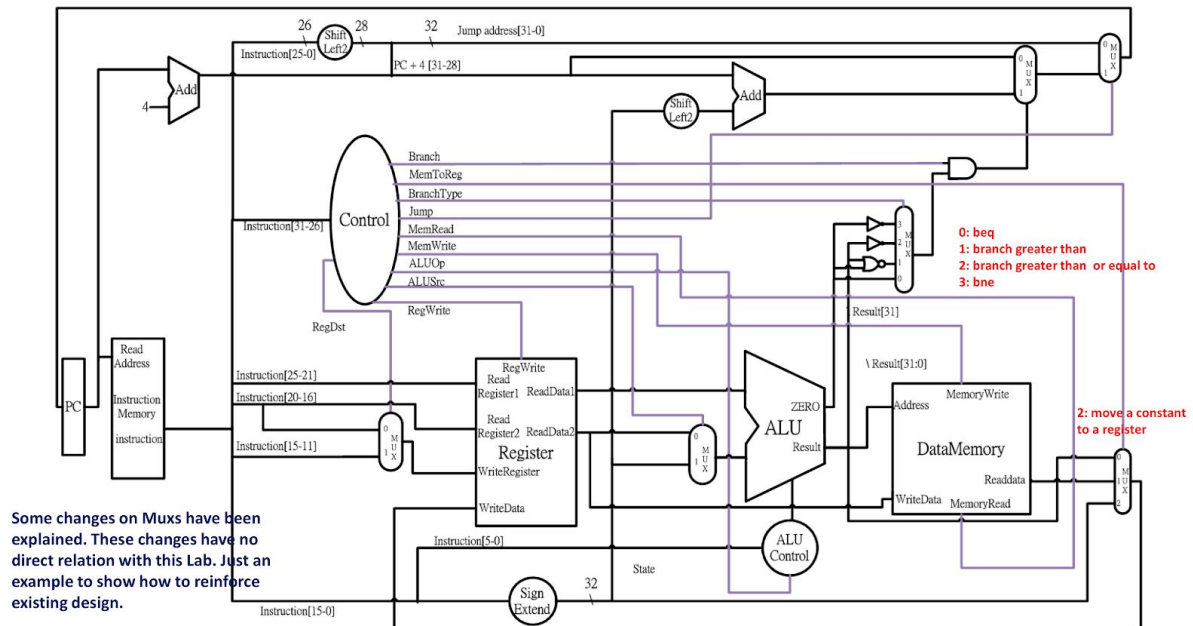
6'b000000	Rs[25:21]	0	0	0	6'b001000

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**.

Some changes on Muxs have been explained. These changes have no direct relation with this Lab. Just an example to show how to reinforce existing design.

0: beq
1: branch greater than
2: branch greater than or equal to
3: bne

2: move a constant to a register

4. Report

a. Architecture Diagram ⇐ 像上面那張圖
b. Hardware Module Analysis ⇐ 分別解釋重要的module怎麼實作
c. Result ⇐ 波形、輸出訊息 …
d. Summary ⇐ 實作心得

5. Grade

a. **Total:** 100 points (plagiarism will get 0 point)
b. **Report:** 20 points (please use **pdf format**)