Computer Organization
Lab 4: Pipelined CPU

Due: 2022/6/6

1. Goal

Based on your Lab 3 CPU design, implement a pipelined CPU.

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/
├── report_310551072.pdf
├── Adder.v
├── ALU_Ctrl.v
├── ALU.v
├── Decoder.v
├── MUX_2to1
├── Shift_Left_Two_32. v
├── Sign_Extend.v
├── Pipe_CPU_1.v
└── (Any .v file that you need)
```

d. Please **do not add unnecessary or given files and folders** (like .DS_Store, __MACOSX)

e. Instruction Memory, Data Memory, Pipe Reg, Reg File, Testbench are supplied. (don't need to modify these files)

f. In the top module, please change N to the value which is total lengths of input signal (include data and control) of pipeline register.

Pipe_Reg #(.size(N)) ID_EX

(google → verilog+parameter / parameterized modules / 參數式模組)

g. Your CPU needs to support the following instructions: (80%)

i. ADD
ii. ADDI
iii. SUB
iv. AND
v. OR
vi. SLT
vii. SLTI
viii. LW
ix. SW
x. BEQ

xi. MULT

mult rd, rs, rt; // rd=rs*rt

0	rs	rt	rd	0	24

h. Testbench ("CO_P4_test_1.txt"):
Use this testbench to test the basic instructions:

```
begin:
addi    $1,$0,3;        //  a = 3
addi    $2,$0,4;        //  b = 4
addi    $3,$0,1;        //  c = 1
sw      $1,4($0);       //  A[1] = 3
add     $4,$1,$1;       //  $4 = 2*a
or      $6,$1,$2;       //  e = a |     b
and     $7,$1,$3;       //  f = a &     c
sub     $5,$4,$2;       //  d = 2*a     - b
slt     $8,$1,$2;       //  g = a <     b
beq     $1,$2,begin
lw      $10,4($0);      //  i = A[1]
```

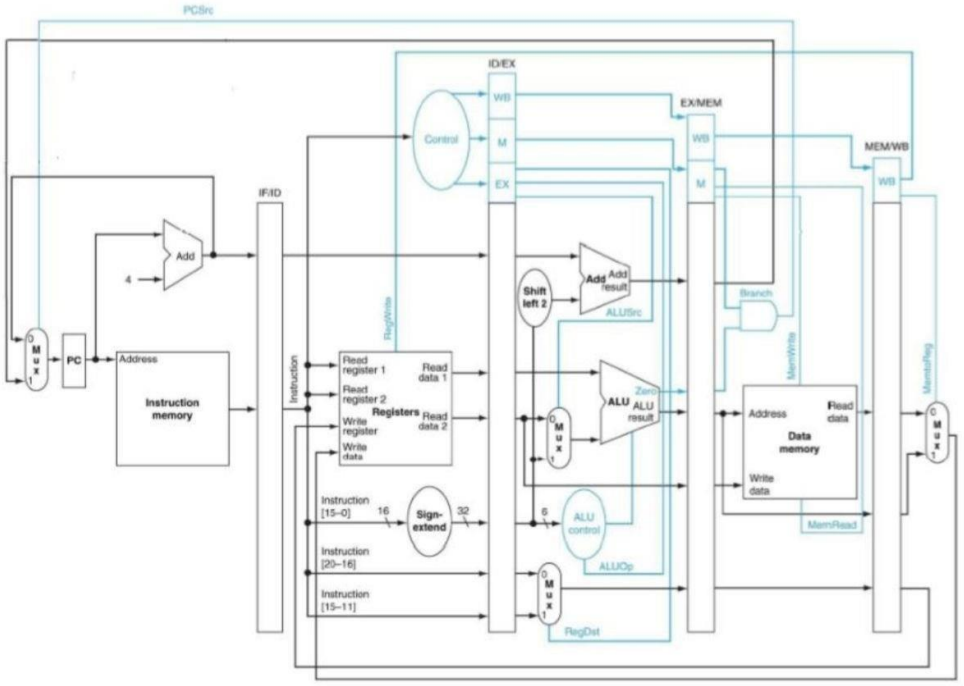
i. Bonus: Answer the question below and write it on your report. (20%)

Consider "CO_P4_test_2.txt", try to solve the data hazard of I1/I2, I5/I6, and I8/I9 data dependency. Just modify the machine code of the testbench and test it on your pipeline CPU. (Write down the machine code and show the execution result in your report.)

```
I1:     addi    $1,$0,16
I2:     addi    $2,$1,4
I3:     addi    $3,$0,8
I4:     sw      $1,4($0)
I5:     lw      $4,4($0)
I6:     sub     $5,$4,$3
I7:     add     $6,$3,$1
I8:     addi    $7,$1,10
I9:     and     $8,$7,$3
I10:    addi    $9,$0,100
```

Hint: You may (1) insert NOP or (2) reorder instructions.

3. Architecture Diagram



You can modify this design arbitrarily. For example, the logic of the **Mux3** is a little different from our expectation. If the ***MemtoReg*** control signal is 0, it selects the output from the memory. I think that it is much more intuitive if it selects the output from the memory if ***MemtoReg*** is 1. As a result, you can add an inverter, revise the logic in the decoder, or swap the multiplexer inputs whenever you need. By the way, you can write down the modifications of your design in your report.

4. Report

a. Your Architecture

b. Hardware Module Analysis

c. Problems You Met and Solutions

d. Result

e. Summary

5. Grade

a. **Total:** 120 points (plagiarism will get 0 point),

- Report: 20 points (please use **pdf format**)
- Hardware design: 80 points
- Bonus: 20 points

b. **Late submission:** Score * 0.8 before 6/13. After 6/13, 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.