# Computer Organization
# Lab 4: Pipelined CPU

**Due: 2023/5/21**

## 1. Goal

Based on your Lab3 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 ID as comment** at the top of each file.

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, student id = 110550000):

```
110550000/
├── Adder.v
├── ALU.v
├── ALU_Ctrl.v
├── Decoder.v
├── MUX_2to1.v
├── Shift_Left_Two_32.v
├── Sign_Extend.v
├── Pipelined_CPU.v
├── Lab4_110550000.pdf
└── (Any .v file that you need)
```

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

e. Instruction_Memory.v, Data_Memory.v, ProgramCounter.v, Pipe_Reg.v, Reg_File.v, and testbench.v are supplied. (Don’t modify these files.)

f. In the top module, for *MUX_2to1* modules and *Pipe_Reg* modules, please change ‘N’ to the total length of input / output signals (including data and control signals) of the module.

ex.

```
Pipe_Reg #(.size(N)) ID_EX
```

You can search “Verilog + parameter”, “parameterized modules”, or “參數式模組” for more information.

g. **Your CPU should be able to support the following instructions: (85%)**

1. **ADD**
2. **ADDI**
3. **SUB**
4. **AND**
5. **OR**
6. **SLT**
7. **SLTI**
8. **LW**
9. **SW**
10. **BEQ**
11. **XOR**
12. **MULT**

HINT: xor rd, rs, rt; // rd = rs ^ rt

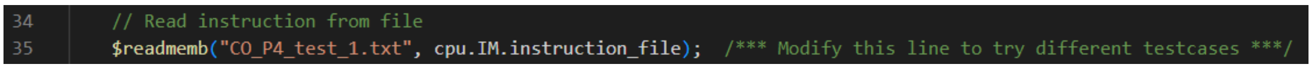
| | | | | | |
|---|---|---|---|---|---|
| 0 | rs | rt | rd | 0 | 38 |

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

| | | | | | |
|---|---|---|---|---|---|
| 0 | rs | rt | rd | 0 | 24 |

## 3. Testbench

In this lab, 3 test cases are provided. You can modify line 35 in testbench.v to try different test cases.



```
34      // Read instruction from file
35      $readmemb("CO_P4_test_1.txt", cpu.IM.instruction_file);  /*** Modify this line to try different testcases ***/
```

Among the test cases, CO_P4_test_3_hazard.txt is a bonus. Please solve the problem according to the following description.

a. CO_P4_test_1.txt:

Use this testbench to test the basic instructions:

```
begin:
I1:   addi    $1, $0, 3        // r1 = 3
I2:   addi    $2, $0, 4        // r2 = 4
I3:   addi    $3, $0, 1        // r3 = 1
I4:   sw      $1, 4($0)        // m[1] = r1
I5:   add     $4, $1, $1       // r4 = r1 + r1
I6:   or      $6, $1, $2       // r6 = r1 | r2
I7:   and     $7, $1, $3       // r7 = r1 & r3
I8:   sub     $5, $4, $2       // r5 = r4 – r2
I9:   slt     $8, $1, $2       // r8 = (r1 < r2)
I10:  beq     $2, $1, -10      // if (r2 == r1), go back to begin
I11:  lw      $10, 4($0)       // r10 = m[1]
```

b. CO_P4_test_2.txt:

Use this testbench to test the new instructions in this lab (xor, mult):

```
begin:
I1:   addi    $11, $0, 2        // r11 = 7
I2:   addi    $2, $0, 4         // r2 = 4
I3:   addi    $3, $0, 0         // r3 = 5
I4:   sw      $11, 4($0)        // m[1] = r11
I5:   mult    $4, $11, $11      // r4 = r11 * r11
I6:   xor     $6, $11, $2       // r6 = r11 ^ r2
I7:   and     $7, $11, $3       // r7 = r11 & r3
I8:   slti    $8, $11, 10       // if (r11 < 10), r8 = 1
I9:   lw      $10, 4($0)        // r10 = m[1]
```

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

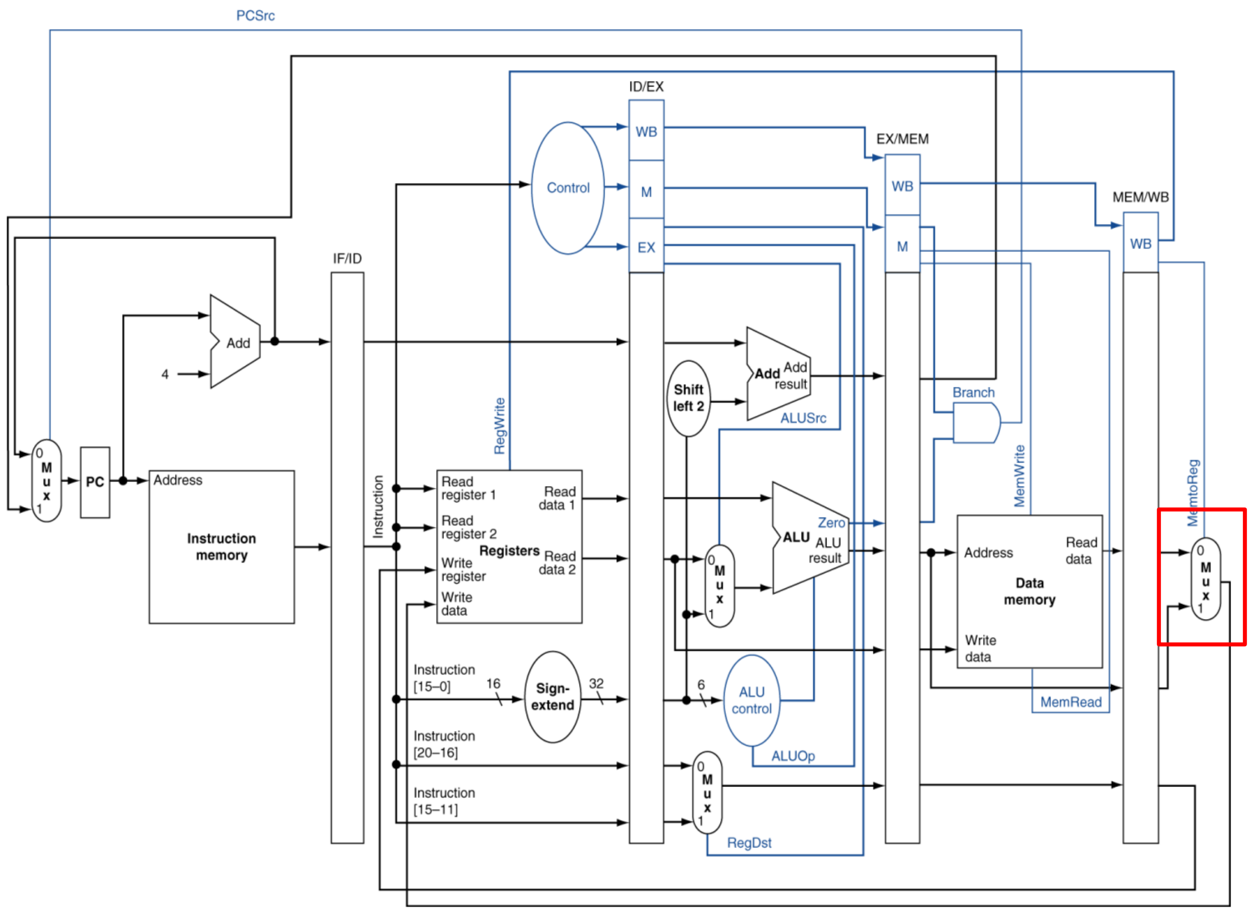
Consider "CO_P4_test_3_hazard.txt", try to solve the data hazards between I1/I2, I5/I6, and I8/I9. Just modify the machine code and test it on your pipeline CPU.

**Write down how you solve the data hazards and show the machine code and the execution result in your report to get bonus points. (You can write this part in "Simulation result" section.)**

```
begin:
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 **insert NOP** or **reorder instructions** to solve the data hazards.

## 4. Architecture Diagram



Note that the selection logic of the multiplexer in the red box is different from that in the architecture diagram in Lab3. This diagram is just for your reference, you can modify this design according to your needs. By the way, you can write down the modifications you've done in your report.

## 5. Report

a. Architecture diagrams

b. Hardware module analysis

c. Simulation results

d. Problems you met and solutions

e. Summary

## 6. Grade

a. **Total:** 100 points and 20 bonus points (plagiarism will get 0 point),

- Report: 15 points (please use **pdf format**)
- Hardware design: 85 points
- Bonus: 20 points

b. **Late submission:** Score * 0.8 before 5/28. After 5/28, you will get 0.

c. **Wrong format:** 10 points punishments

## 7. Q&A

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