Lab 05:
HW 4

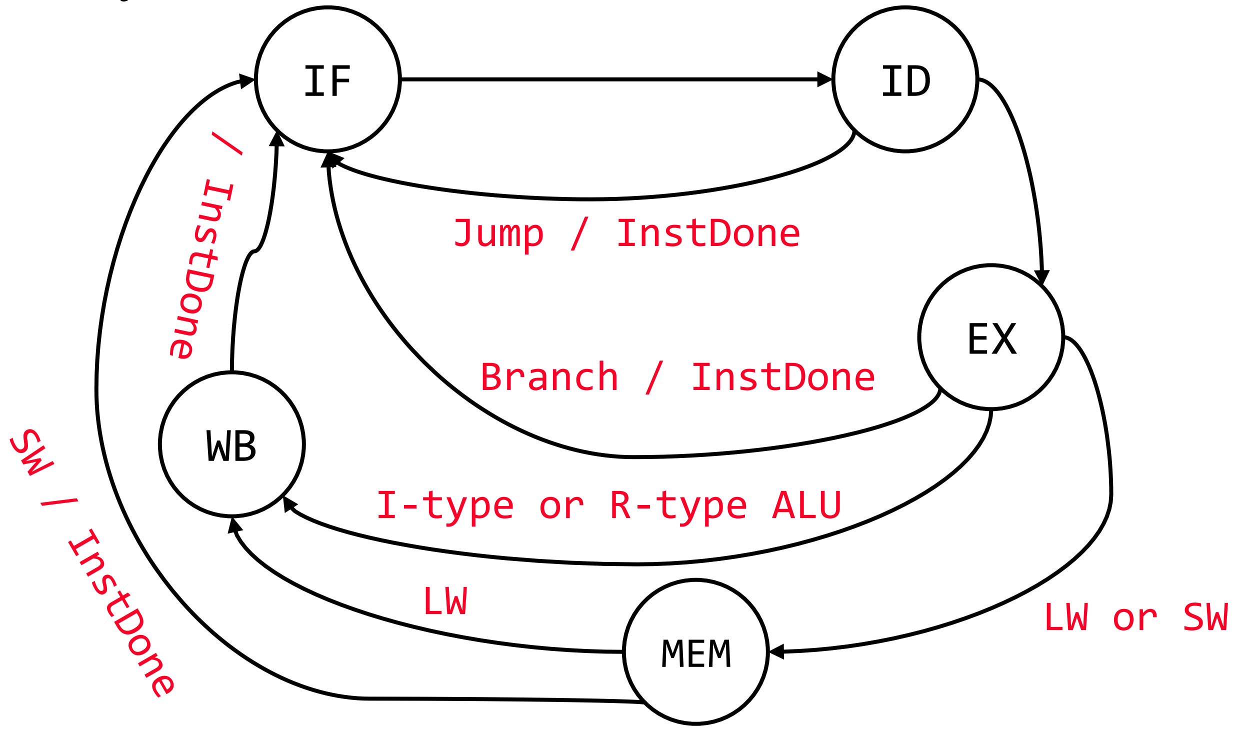
Hunjun Lee
hunjunlee@hanyang.ac.kr

Overview

- You will upgrade your single cycle CPU into a multicycle CPU
- What to do?
  - Define additional control signals for multi-cycle architecture
  - Build an FSM to control multi-cycle execution
  - Rearchitect the datapath and finalize your design

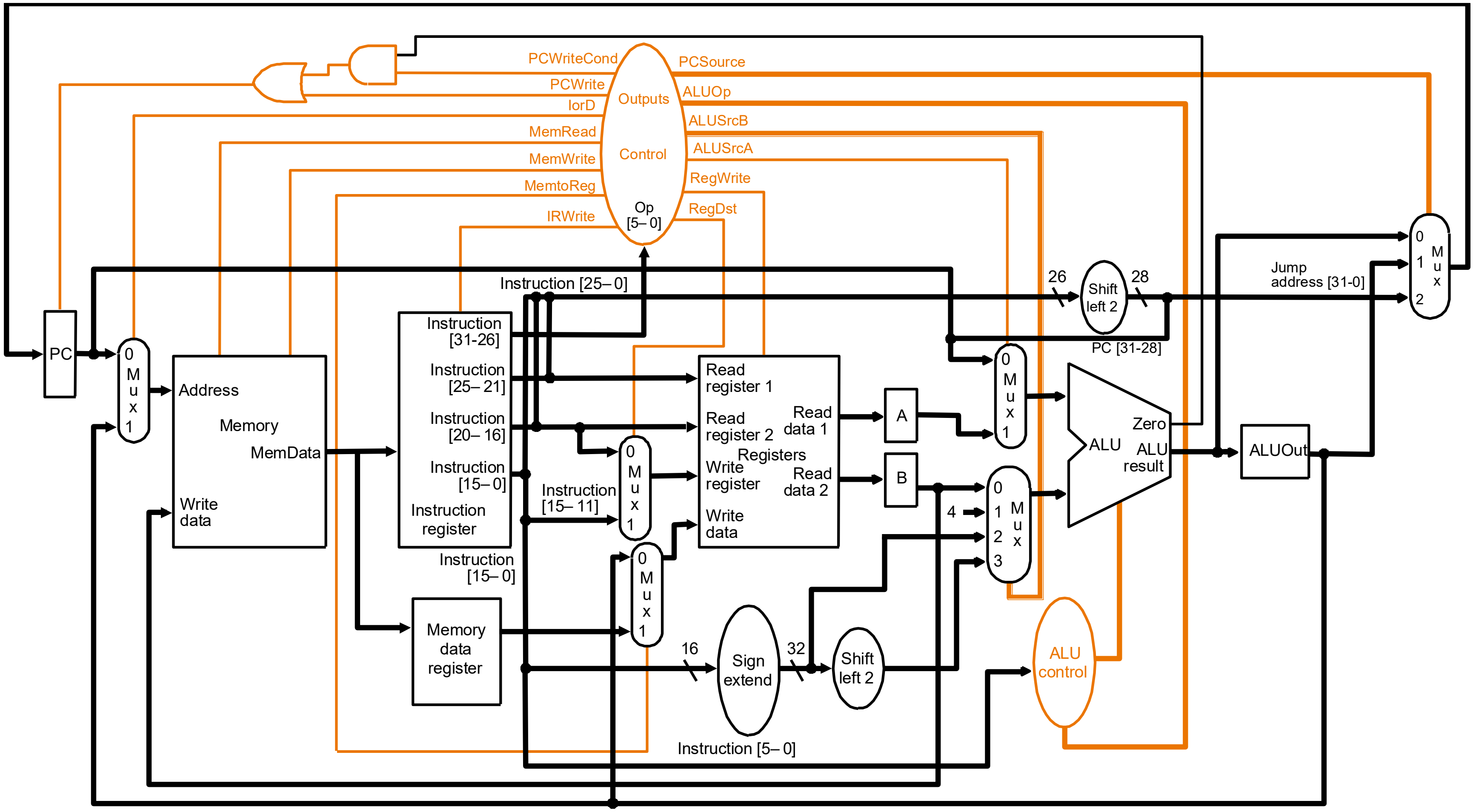
Multicycle Implementation

- You will be implementing a **up-to five-stage multicycle CPU**
- Each stage will take a single clock cycle
  - IF -> ID -> EX -> MEM -> WB
- Refer to the following FSM
  - Define your own FSM for JR and JAL, …



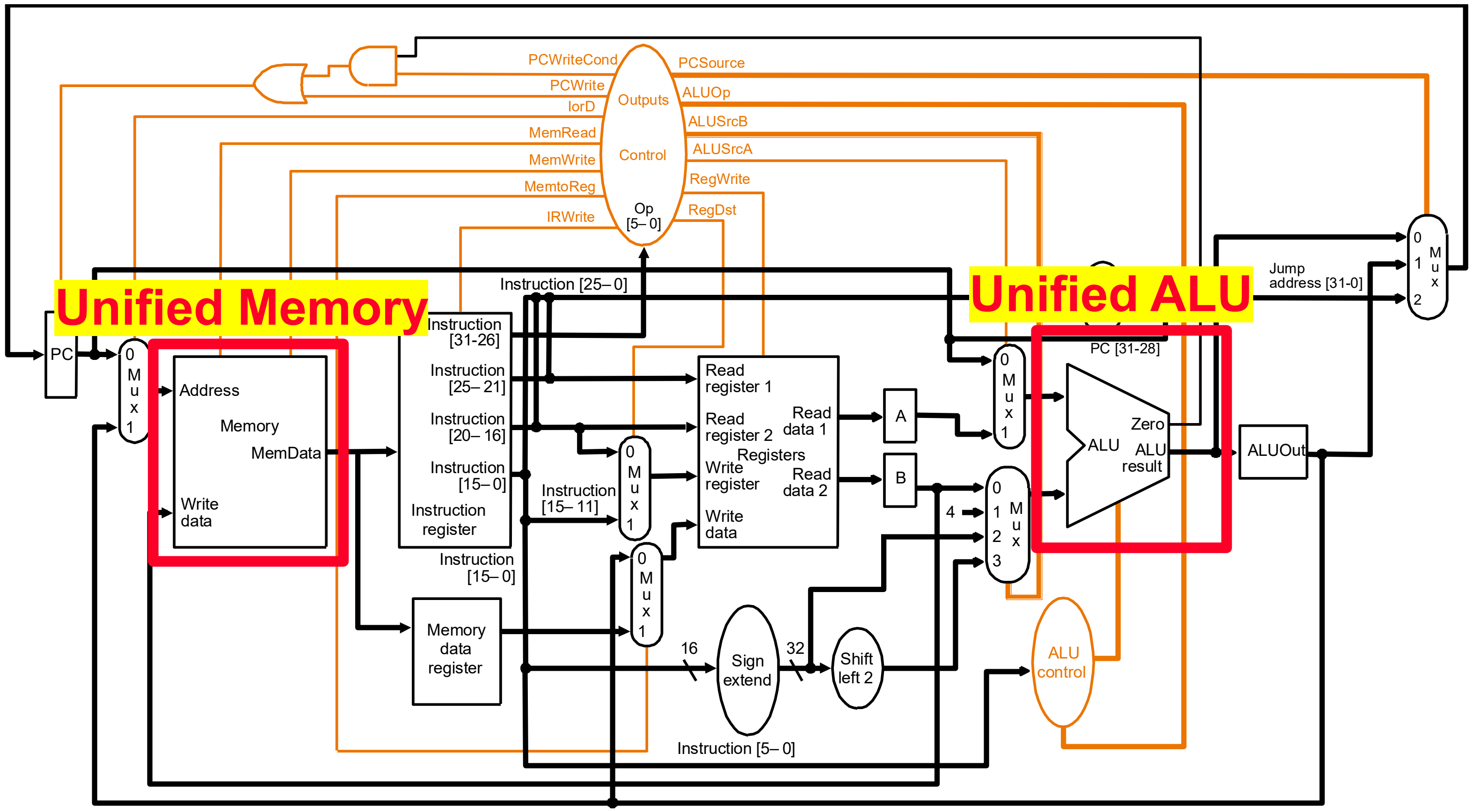
Target CPU Microarchitecture - 1

- Implement what you learned in the class, but again you need to **support jal and jr**



Target CPU Microarchitecture - 2

- Also, you should **share resources** among different computation stages (penalty if you do not or only partially share resources)

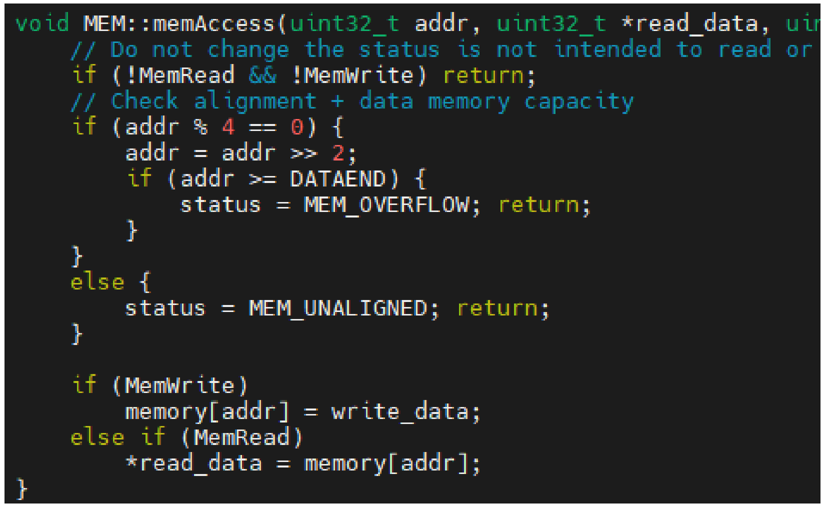


Assignment

- **Files:**
  - **global.h + GLOBAL.v** **(HW3)**
    - Include constant values (opcode, funct …)
    - You may change it to define states
  - **ALU.cpp/h + ALU.v** **(HW1)**
  - **RF.cpp/h + RF.v** **(HW2)**
  - **MEM.cpp/h + MEM.v** **(Provided)**
    - Modify the memory module (make it share a single port)
  - **CTRL.cpp/h + CTRL.v** **(Modify HW3)**
    - A controller to determine control signals
  - **CPU.cpp/h + CPU.v** **(Modify HW3)**
    - The top module to combine these modules
  - **main.cpp + CPU_tb.v** **(HW3)**
    - I gave you the test files for HW3 (Use it)

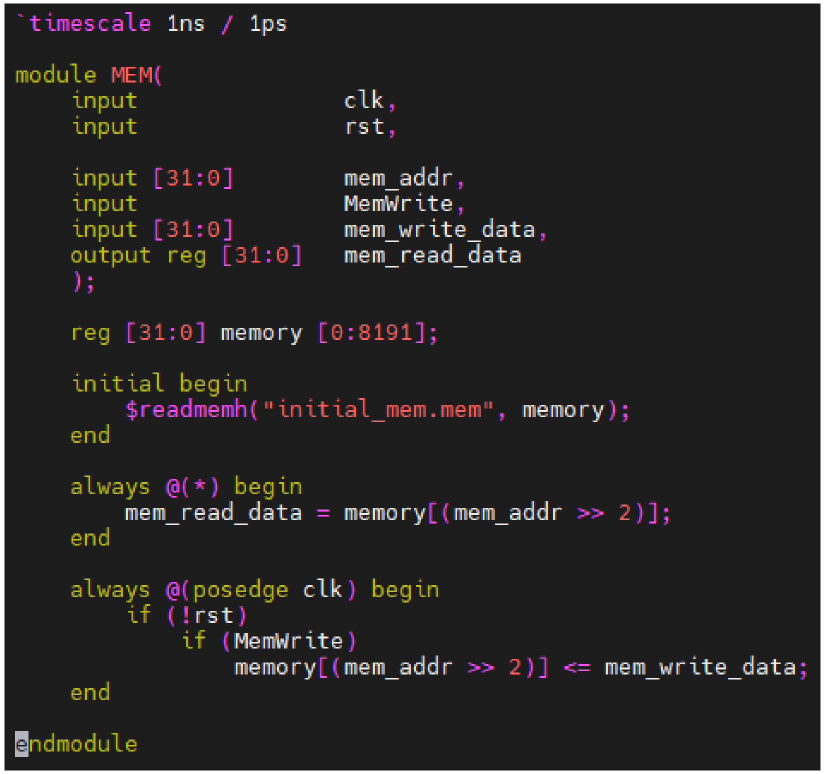
Unified Memory Module Implementation

- Now, your memory module should integrate
  - instruction addr w/ memory addr
  - read data w/ inst



```
void MEM::memAccess(uint32_t addr, uint32_t *read_data, uir
    // Do not change the status is not intended to read or
    if (!MemRead && !MemWrite) return;
    // Check alignment + data memory capacity
    if (addr % 4 == 0) {
        addr = addr >> 2;
        if (addr >= DATAEND) {
            status = MEM_OVERFLOW; return;
        }
    }
    else {
        status = MEM_UNALIGNED; return;
    }

    if (MemWrite)
        memory[addr] = write_data;
    else if (MemRead)
        *read_data = memory[addr];
}
```



```
`timescale 1ns / 1ps

module MEM(
    input                   clk,
    input                   rst,

    input [31:0]            mem_addr,
    input                   MemWrite,
    input [31:0]            mem_write_data,
    output reg [31:0]       mem_read_data
    );

    reg [31:0] memory [0:8191];

    initial begin
        $readmemh("initial_mem.mem", memory);
    end

    always @(*) begin
        mem_read_data = memory[(mem_addr >> 2)];
    end

    always @(posedge clk) begin
        if (!rst)
            if (MemWrite)
                memory[(mem_addr >> 2)] <= mem_write_data;
    end

endmodule
```

Submission

- Submission (Zip all the files)
  - For a two-people team: lab4_student_id1_student_id2.zip
  - For a one-person team: lab4_student_id.zip
    - You must follow the format (-10% for wrong file format)
    - Example:
      - lab4_2020102030.zip
      - lab4_2020102030_2022103040.zip
  - The zip file should contain:
    - every cpp/h/v files
    - lab4_report.pdf

Submission

- You need to write a 4+-page report
  - Explain the overall structure and how you implemented each program
  - What's your FSM to implement JAL and JR instruction?
  - Draw the hardware modules if needed …
  - How did you implement JAL / JR instruction and modified the uarchitecture?

- Due: Fri. May 9th
  - 1 week delay: -20%
  - 2 week delay: -50%
  - Further delay: 0 point

- You can submit HW3 by May 9th with only 20% deduction