CPU function and performance simulator

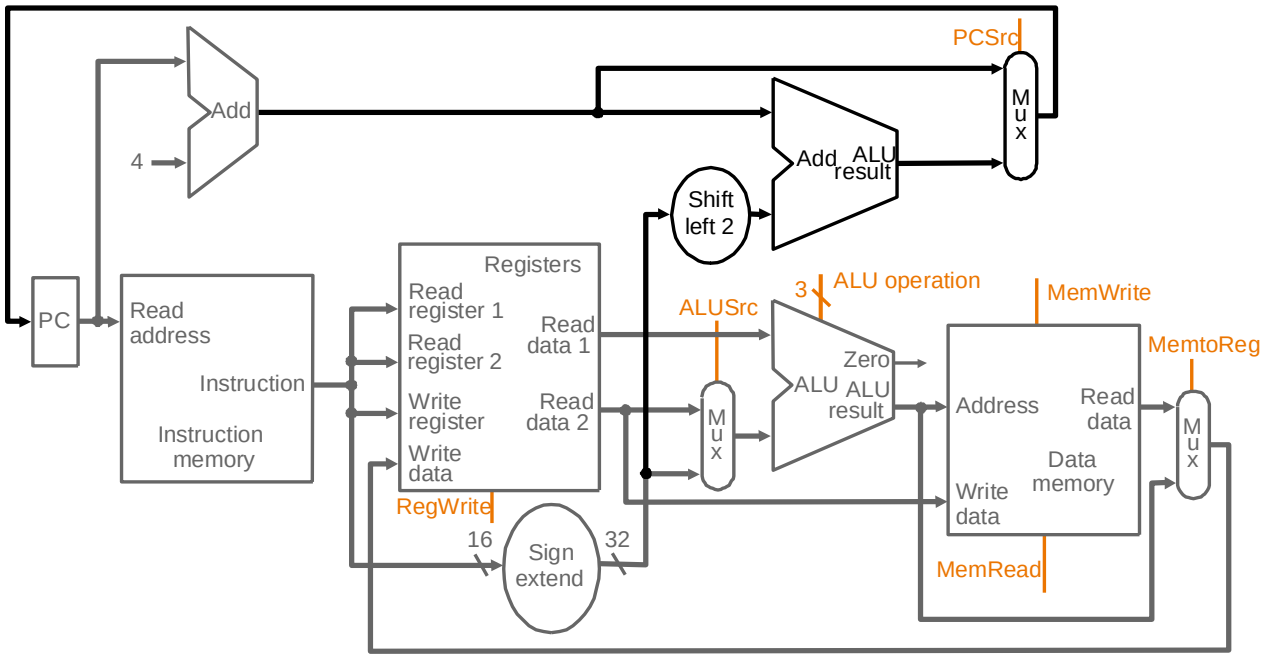
Performance simulation

Today's Goal

- **Single cycle performance simulation**
- **Multi-cycle performance simulation**

Single-Cycle Datapath

- **Attempt to execute all instructions in one clock-cycle.**
- **No datapath resource can be used more than once per instruction.**
- **Some of the functional units need to be duplicated for using more than once.**

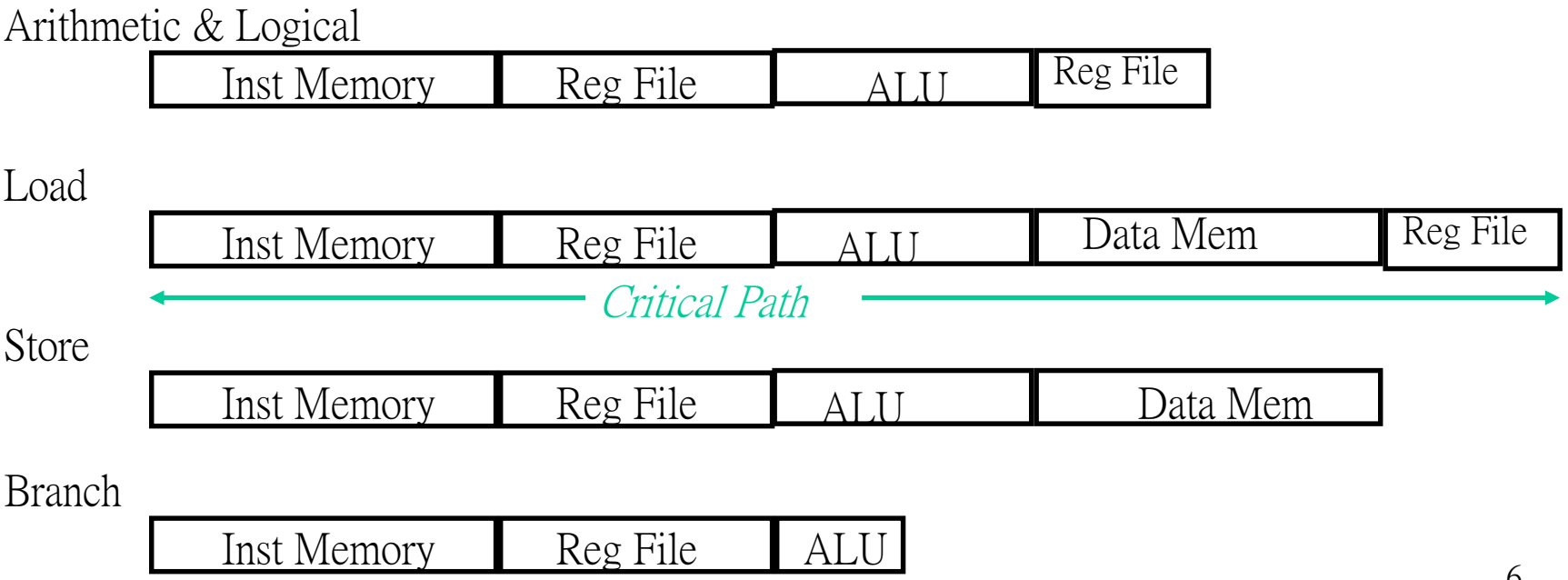


Multiple Cycle Datapath

- **Break the instruction into smaller steps**
- **Execute each step (instead of the entire instruction) in 1 clock cycle**
  - Cycle time: time it takes to execute the longest step
  - Try to make all the steps have similar length
- **The advantage of the multiple cycle processor:**
  - Cycle time is much shorter
  - Different instructions take different number of cycles to complete
  - Allows a functional unit to be used more than once per instruction

Multi : Five Execution Steps

- **Instruction Fetch**
- **Instruction Decode and Register Fetch**
- **Execution, Memory Address Computation, or Branch Completion**
- **Memory Access or R-type instruction completion**
- **Write-back step**



HOW TO

- **Set CPU cycle time in function init_CPU_cycle_time at machine.c**
  - **Multi Cycle time: time it takes to execute the longest step**
    - Step2&4 : MEMORY_ACC_TIME
    - Step3 : ALU_TIME
    - Step1&5 : REGISTER_ACC_TIME
- **Implement performance simulation in function inst_func_simulation at inst_process.c**

作業說明

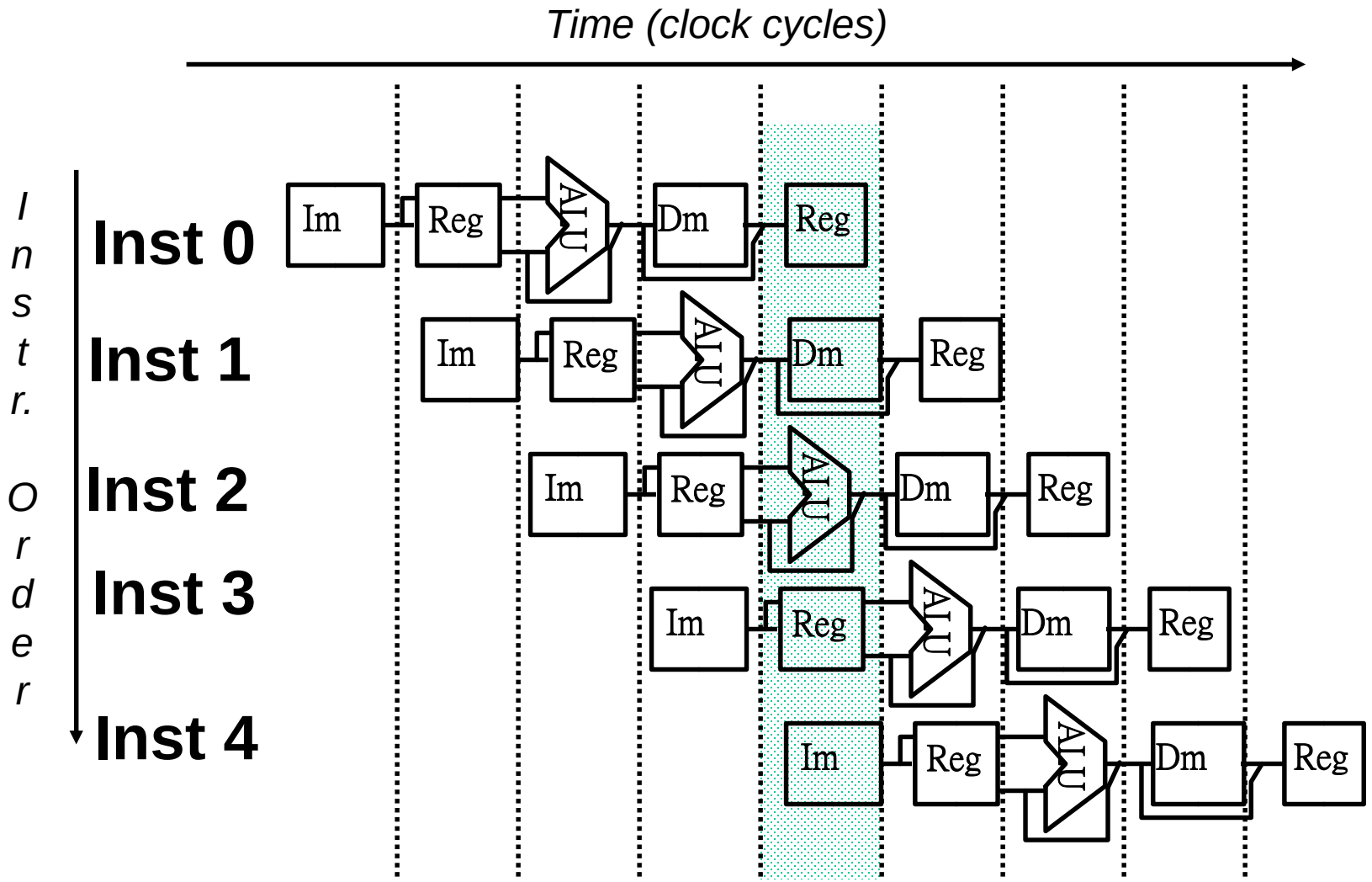
- **須完成**
  - cycle-by-cycle performance simulations
    - Single cycle datapath
    - Multi cycle datapath
- **步驟**
  - 將範例程式缺少的部分填寫完成
  - 執行程式並用simulator run範例assembly code
  - 觀看輸出結果

Pipeline simulation

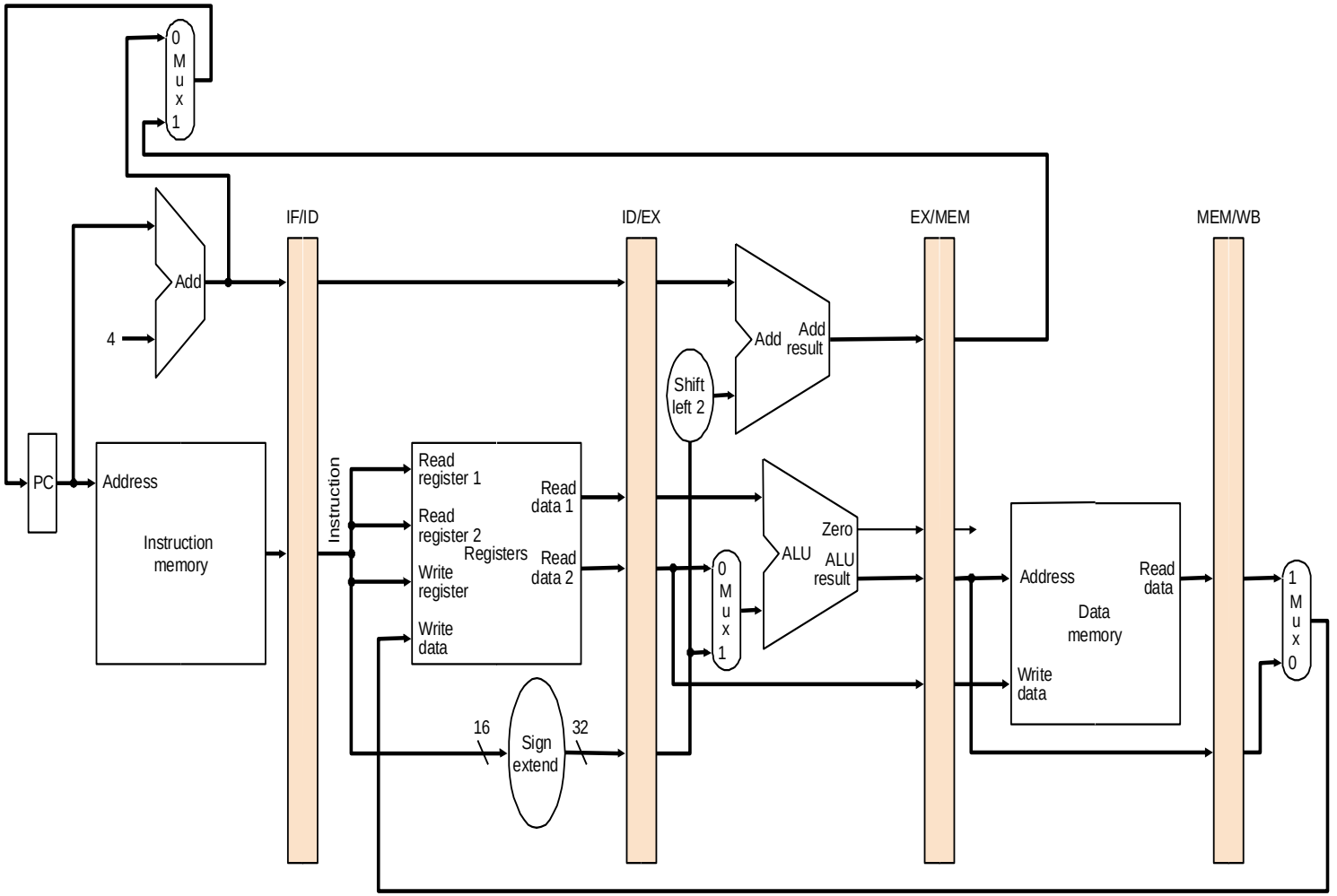
Goal

- **Pipeline functional simulation**
  - Pipeline 5 stage function
  - Hazard detection
  - Solving hazard
- **Pipeline performance simulation**

Pipeline overview



Pipeline datapath



Pipeline hazard

- **Pipeline Hazards**
  - structural hazards: attempt to use the same resource two different ways at the same time
    - E.g., combined washer/dryer would be a structural hazard or folder busy doing something else (watching TV)
  - data hazards: attempt to use item before it is ready
    - E.g., one sock of pair in dryer and one in washer; can't fold until get sock from washer through dryer
    - instruction depends on result of prior instruction still in the pipeline
  - control hazards: attempt to make a decision before condition is evaluated
    - E.g., washing football uniforms and need to get proper detergent level ; need to see after dryer before next load in
    - branch instructions
- **Can always resolve hazards by waiting**
  - pipeline control must detect the hazard
  - take action (or delay action) to resolve hazards

Homework

- **Implement 5 stage into 5 function in c code**
  - Inst_fetch()
  - Inst_decode()
  - Inst_excute()
  - Mem_writeback()
  - Reg_update()
- **Implement hazard detection**
  - If_hazard()
- **Solving hazard in each stage**
- **End of simulation**
  - Pipeline is null and “Exit:” is last instruction
- **Performance simulation**
  - Every stage completes the work in 1 cycle

HOWTO-pipeline process

- **Implement in pipeline.c**
- **Walks pipeline from mem_writeback to inst_fetch**
  - backward pipeline traversal eliminates relaxation problems
    - Easier to express pipeline stall

```
for(;;)
{
        Inst_fetch()
        Inst_decode()
        Inst_excute()
        Mem_writeback()
        Reg_update()
}
```

Inst_fetch()

- **Check if IF/ID register is null then**
- **Update PC**
- **Update IF/ID register**
  - Register rs、rt、rd
  - Immediate
  - Instruction type

Inst_decode()

- **Check if ID/EX register is null then**
- **Decode the instruction and fetch register**
- **Branch instruction should be done in this stage**
  - bne、beq、j

Inst_execute()

- **Check if EX/MEM register is null then**
- **Execute the instruction**
  - The operation you have written in single cycle datapath

Mem_writeback()

- **Write content to memory**
- **Read content from memory**

Reg_update()

- **Update the register content**
- **End the instruction**

Hazard detection

- **Assume structural hazard not exist**
- **Data hazard (2 cases)**
  - Data you need is in EX/MEM
  - Data you need is in MEM/WB
- **Control hazard**
  - Branch being executed in decode stage
  - Branch prediction always predict not taken
  - You have to detect flush the pipeline or not

Data hazard detection

- **Hazard conditions:**
  - 1a: EX/MEM. RegisterRd = ID/EX.RegisterRs
  - 1b: EX/MEM. RegisterRd = ID/EX.RegisterRt
  - 2a: MEM//WB.RegisterRd = ID/EX.RegisterRs
  - 2b: MEM/WB.RegisterRd = ID/EX.RegisterRt

Solving hazards

- **Forwarding data in case 1**
- **Forwarding data in case 2**
- **Flushing the pipeline for branch**
- **Stall the pipeline**

加分題配分

- 基本**pipeline** 功能**(2 points)**
  - Function of 5 stage
  - No hazard detection
  - You have to print out the intermediate states of all pipeline stages for every cycle
- **Hazard detection and resolving**
  - Data Hazard
    - Correct detection of case 1 (1 point)
    - Correct detection of case 2 (1 point)
    - Forwarding for case 1 when the producer is not “lw” (1 point)
    - Forwarding for case 2 (1 point)
    - Stall the pipeline for load-use case (1 point)
  - Control Hazard
    - Detection (1 point)
    - Predict not taken (1 point)
    - Flush the pipeline when mis-prediction (1 point)