Digital System Design

Homework 3
Hardware Implementation of Single Cycle MIPS

Speaker: Ya-Cheng Ko

Instructor: 吳安宇教授

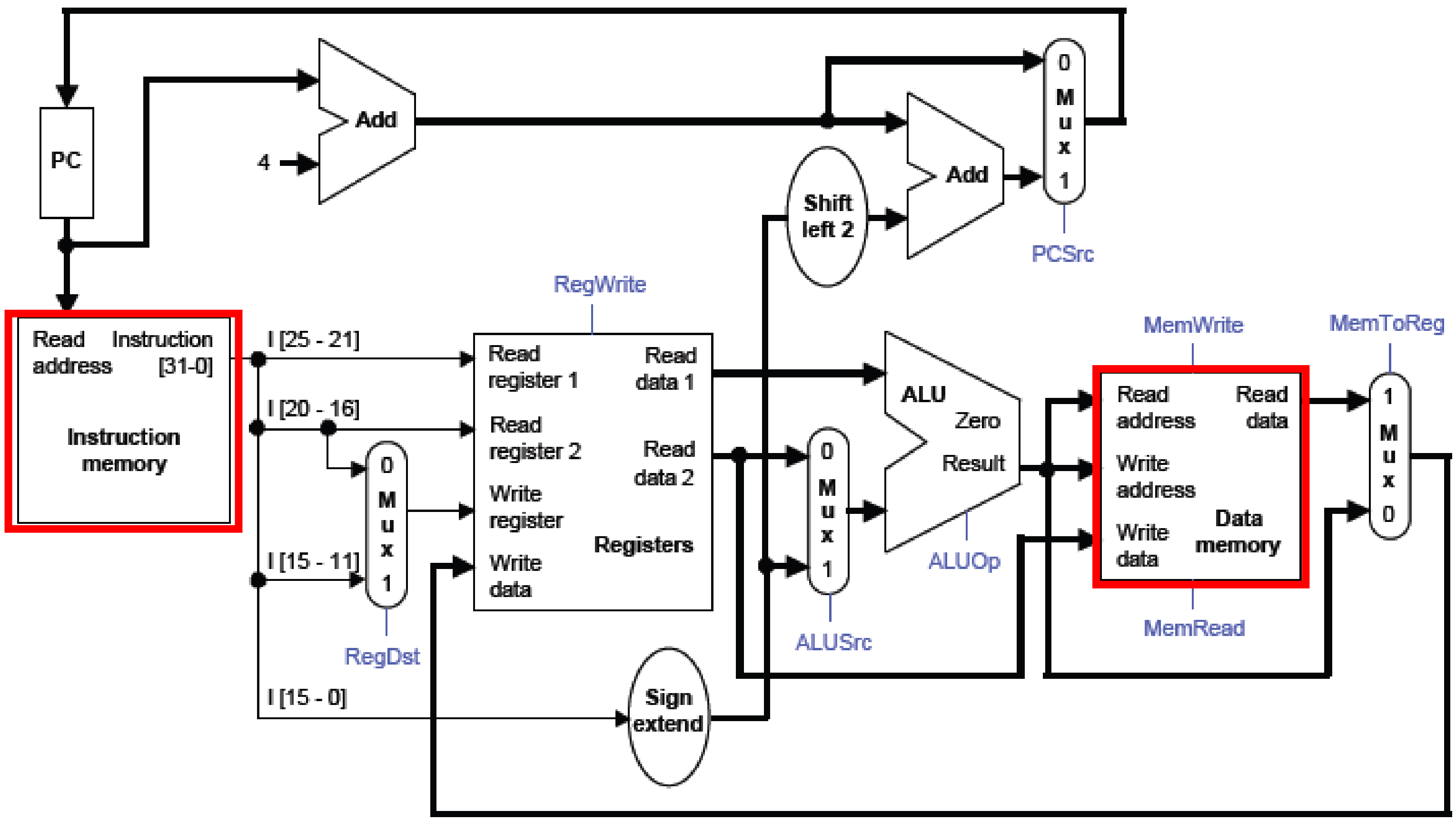
Date: 2020/04/16

Introduction to Single-cycle MIPS Processor

MIPS processor

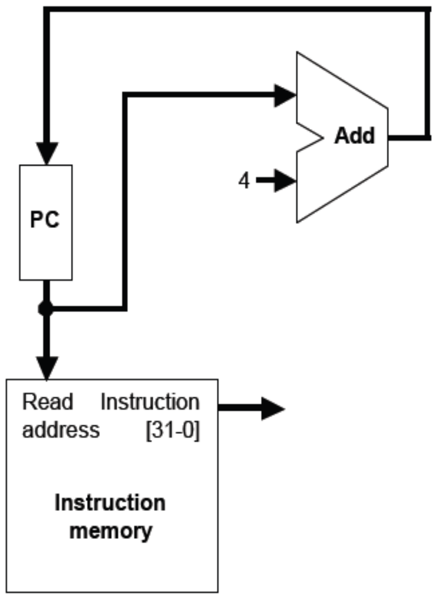
- Any instruction set can be implemented in many different ways.
  - — In a basic single-cycle implementation all operations take the same amount of time—a single cycle.
  - — A multicycle implementation allows faster operations to take less time than slower ones, so overall performance can be increased.
  - — Finally, pipelining lets a processor overlap the execution of several instructions, potentially leading to big performance gains.(Final project)

Datapath


PC
4
Add
Add
Shift left 2
0 Mux 1
PCSrc
RegWrite
Read address
Instruction [31-0]
Instruction memory
I [25 - 21]
I [20 - 16]
I [15 - 11]
I [15 - 0]
0 Mux 1
RegDst
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
Sign extend
0 Mux 1
ALUSrc
ALU
Zero
Result
ALUOp
MemWrite
Read address
Write address
Write data
Read data
Data memory
MemRead
MemToReg
1 Mux 0

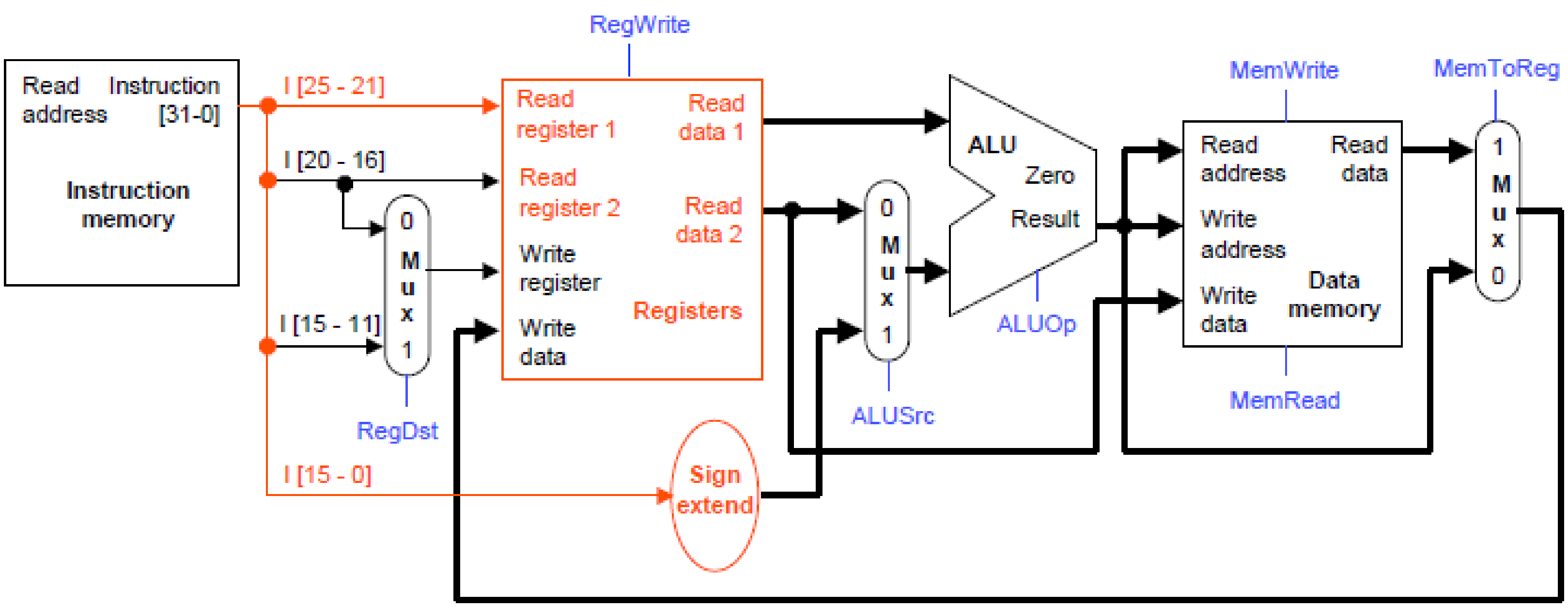
Instruction fetching

- The CPU is always in an infinite loop, fetching instructions from memory and executing them.
- The program counter or PC register holds the address of the current instruction.
- MIPS instructions are each four bytes long, so the PC should be incremented by four to read the next instruction in sequence.



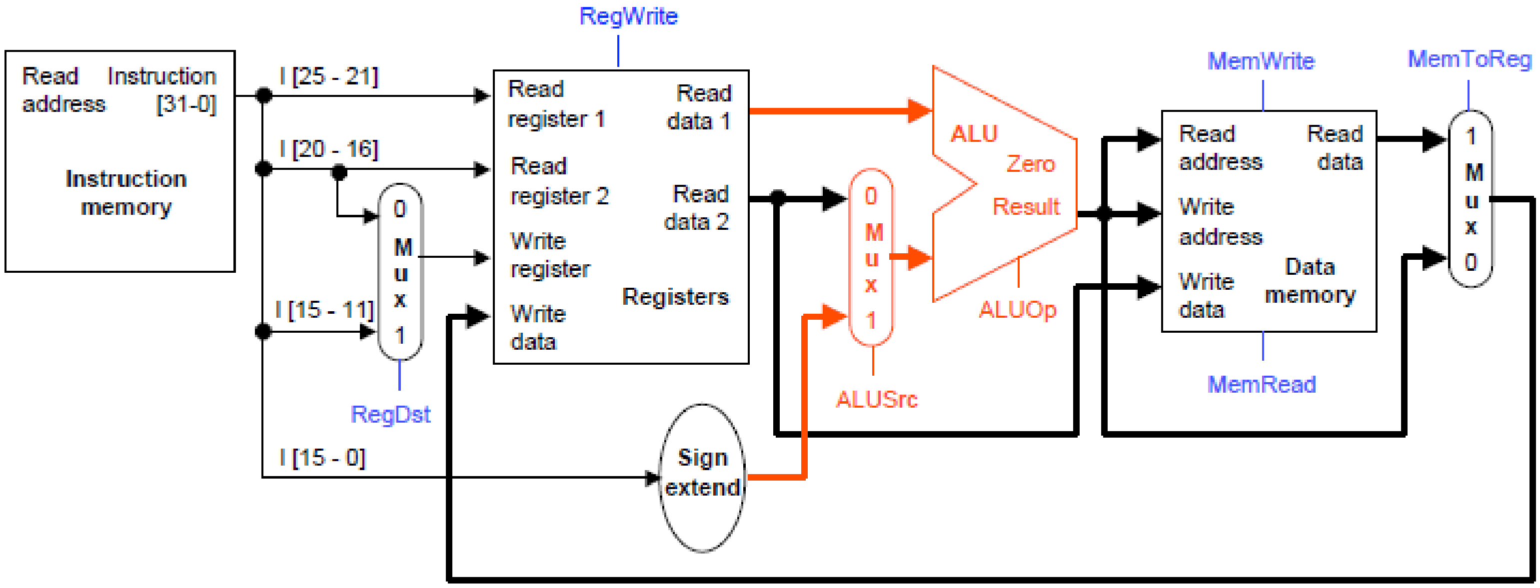
Instruction Decode

- The Instruction Decode (ID) step reads the source register from the register file.



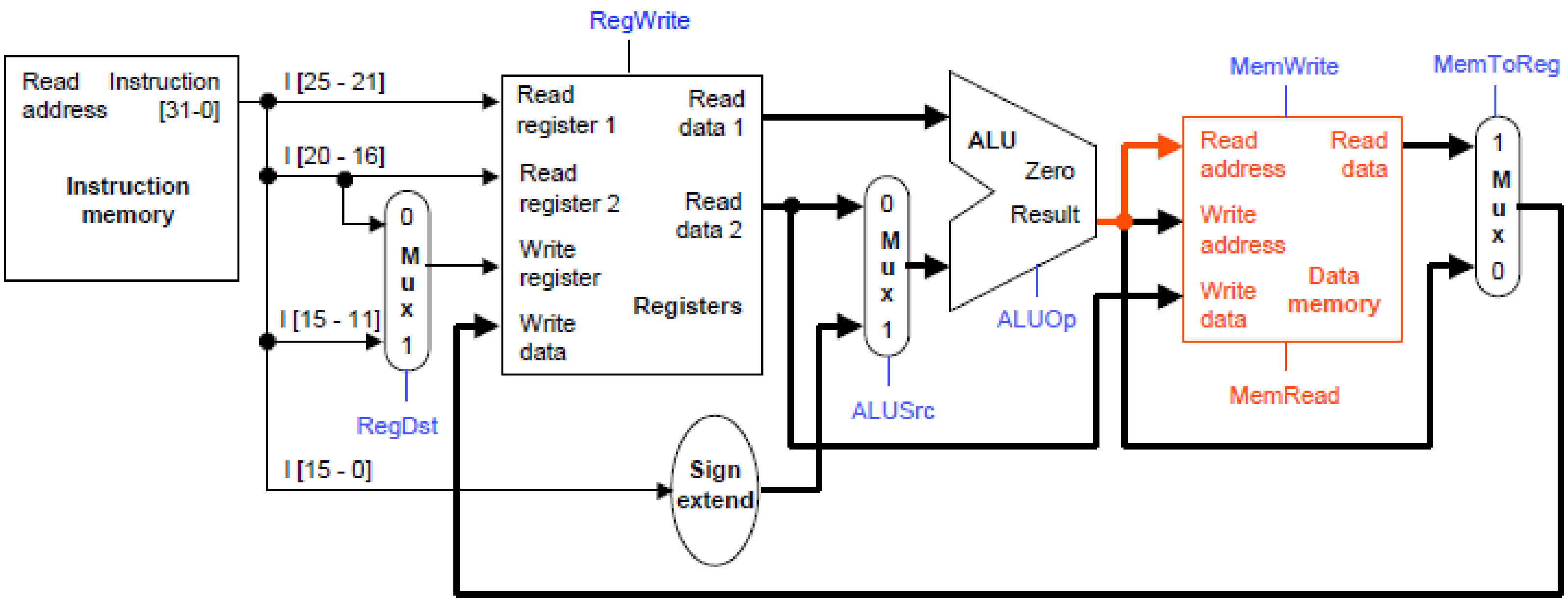
Execute

- The third step, Execute (EX), computes the effective memory address from the source register and the instruction's constant field.



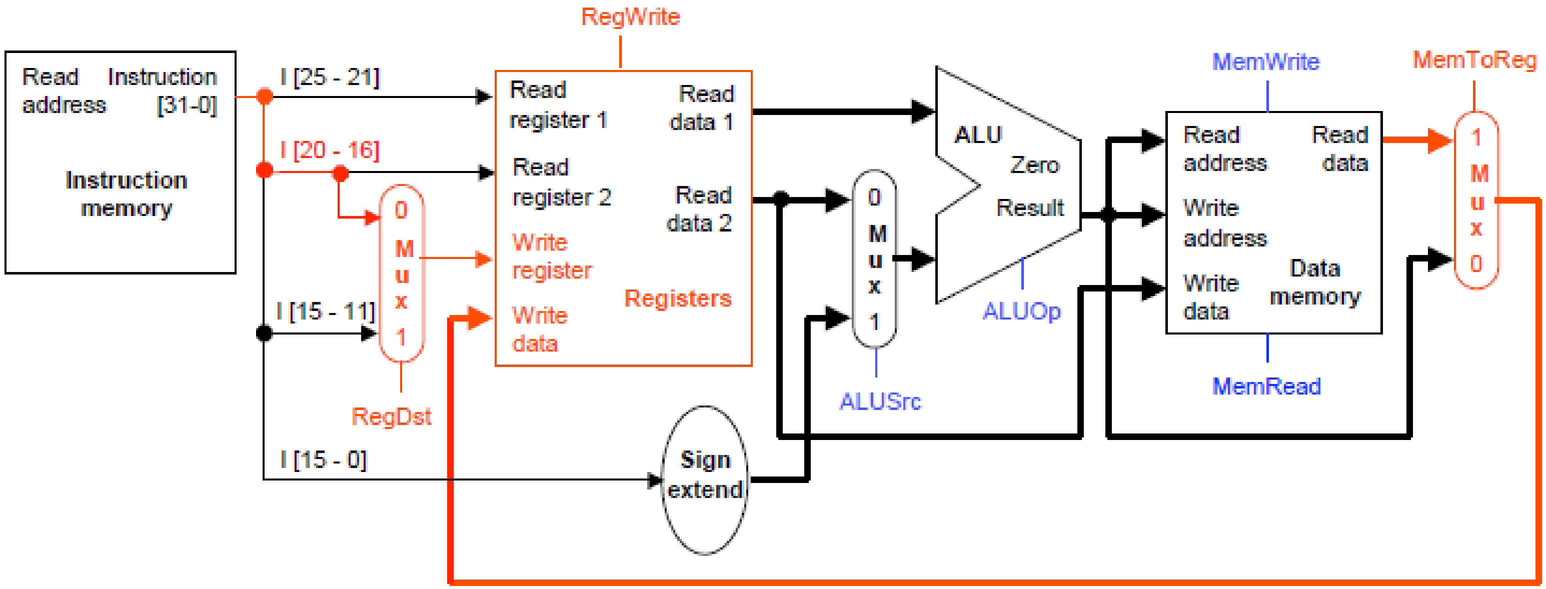
Memory

- The Memory (MEM) step involves reading the data memory, from the address computed by the ALU.



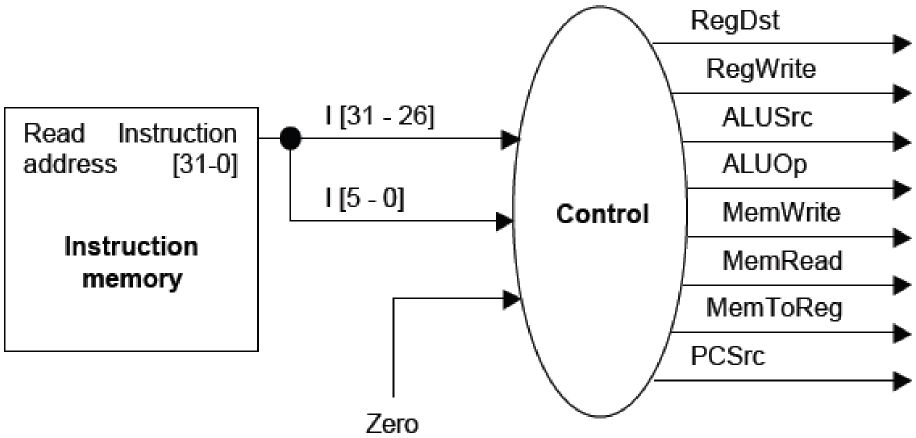
Write Back

- Finally, in the Writeback (WB) step, the memory value is stored into the destination register



Control Unit

- The control unit needs 13 bits of inputs.
  - — Six bits make up the instruction's opcode.
  - — Six bits come from the instruction's func field.
  - — It also needs the Zero output of the ALU.
- The control unit generates 10 bits of output, corresponding to the signals mentioned on the previous page.

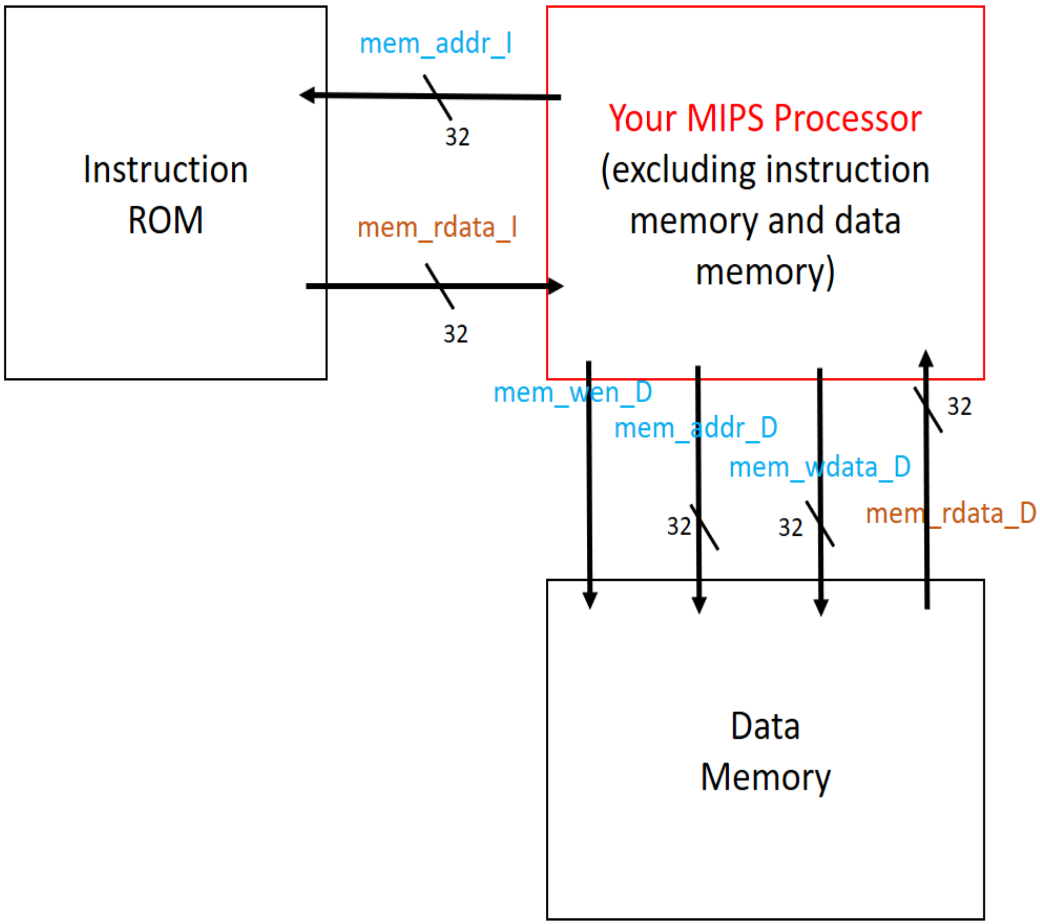


Homework 3
Single-cycle MIPS Processor

Problem Statement

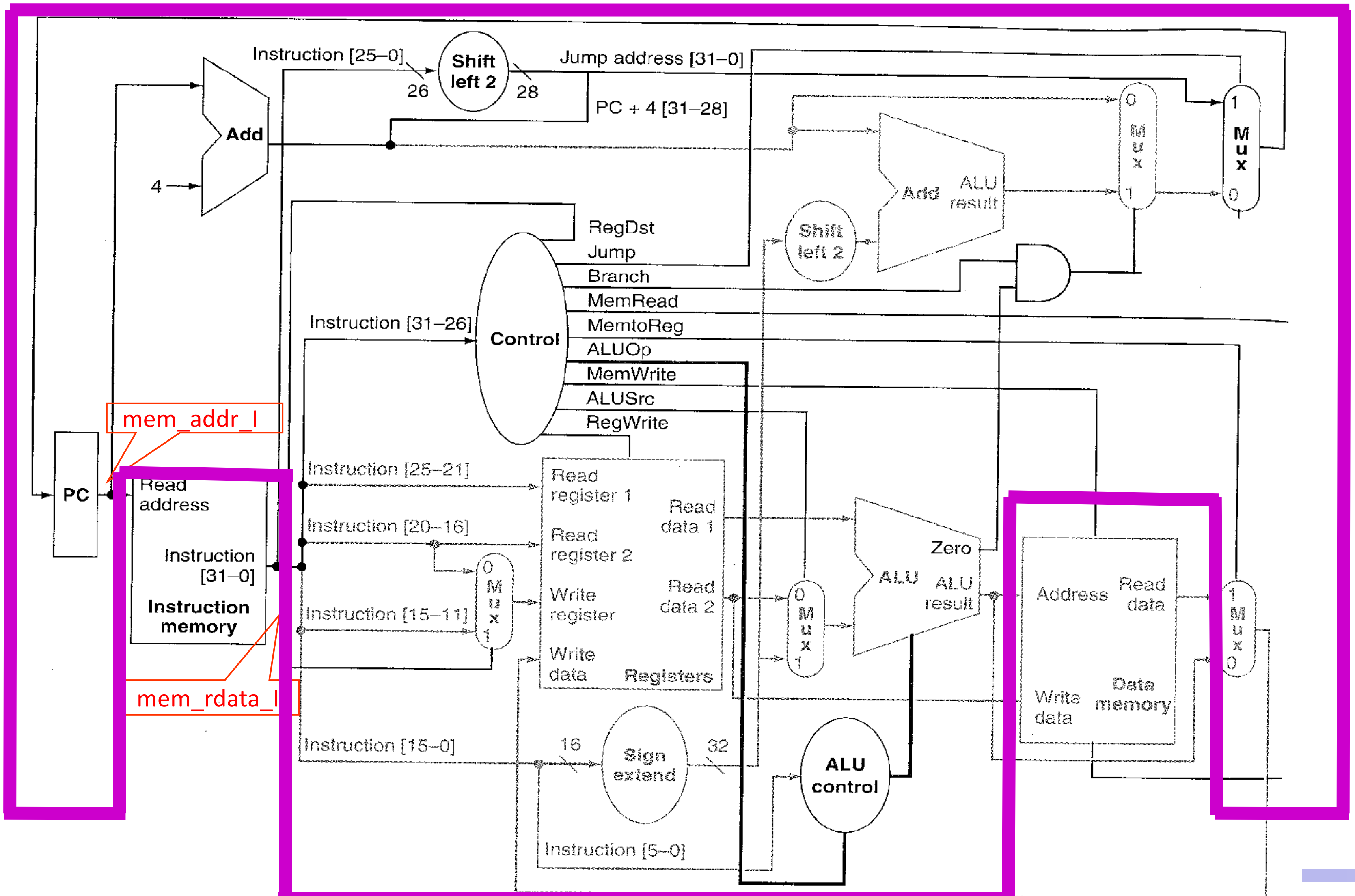
- Using Verilog, implement the single-cycle MIPS processor:
  - Supported instructions:
    - add, sub, and, or, slt
    - lw, sw
    - beq
    - j, jal, jr
- Testbench/Memory model provided

Block Diagram(1/2)



- Instruction ROM: contains the testing instructions
- Data Memory: contains the stored data
  - Used for testing your circuit
- mem_wen_D: mem_wen_D is high, writes data to D-mem when the next clk arrives; else reads data from memory to chip.

Block Diagram(2/2)



Testbench

- The testbench will
  - Initialize the instruction ROM and the data memory
  - Reset your circuit
  - Execute the instructions, and check the values stored in the *data memory* to see whether your circuit is correct
  - If your function is correct, you will see the following

```
------------------------------------------------------------

START!!! Simulation Start .....

------------------------------------------------------------

============================================================

Success!
The test result is .....PASS :)

============================================================
```

Clock/Reset/Register File

- Clock: positive edge triggered
- Reset: active low synchronous reset

- Register file
  - All registers are reset to 0 when reset occurs.
  - Register $0 must be always 0

Memory

- Instruction ROM and data memory are included in the testbench
- As for data memory
  - 32 words x 32 bits
  - The input signal mem_wen_D is high, writes data to D-mem when the next clk arrives; else reads data from memory to chip.

Memory Addressing

- In MIPS, the memory address is in byte address.
- In Instruction ROM and data memory, the memory address is in word address.
- Both the memory size of Instruction ROM and data memory in this work are 32x32, so their input address is 5-bit wide.
  - You are encouraged to observe the connection between each module in MIPS_tb.v .

Simulation & Synthesis

- **Check “MIPS/ verilog/ readme.txt”**
- 3 Major Things
  - RTL coding & simulation
  - Logic Synthesis
  - Gate-level simulation & debugging/refinement
- Files needed for simulation
  - RTL code: *CHIP.v*
  - Gate-level code: *CHIP_syn.v*
  - Timing info (SDF file): *CHIP_syn.sdf*
  - Design library (DDC file): *CHIP_syn.ddc*

※Notice

1. ~~Latches~~ are not allowed in gate level code after synthesis, use Flip-flop instead.

2. Negative Slack and Timing Violations are not allowed after synthesis.

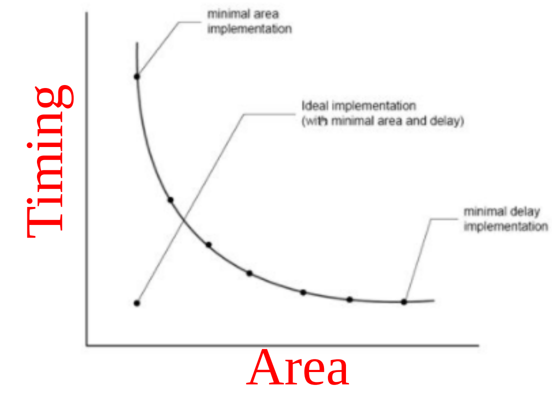
3. The tsmc13.v file is not allowed to be downloaded! Or you may offend the copyright protected by NTU & CIC!

Grading Policy

- RTL (40%): function correctness
- Synthesis (30%): correctness
- Report (10%)
- Area*Timing (20%)
- Bonus (10%): Complete Single Cycle RISCV(5%) + Compressed Instruction(5%)

- TA: 柯亞承
- b05902056@ntu.edu.tw

Report



1. **Simulated timing (ns)**
   - Gate-level simulation clock cycle

   (i.e. The cycle you passed testbench after synthesis)
2. **Area(um^2)**
   - report_area
3. **Cost(A*T)**
   - Area*Gate-level simulation clock cycle
4. **ScreenShot**
   - Inferred memory devices in process

     (※No latch should be inferred!)

```
Number of ports:                          172
Number of nets:                           367
Number of cells:                          130
Number of combinational cells:            125
Number of sequential cells:                 0
Number of macros:                           0
Number of buf/inv:                         39
Number of references:                      22

Combinational area:        43665.613947
Noncombinational area:     32960.112083
Net Interconnect area:       undefined  (No wire load specified)

Total cell area:           76625.726031
Total area:                  undefined
```

```
Inferred memory devices in process
        in routine CHIP line 149 in file
                '/home/raid7_2/userb05/b5902056/1082DSD_TA/HW3_TA/HW3/MIPS/verilog/CHIP.v'.
===============================================================================
|    Register Name     |    Type     | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|       PC_reg         | Flip-flop |  32   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================
```

Submission(1/2)

- For each topic, you need to submit 4 files + 1 report
  - RTL code: *CHIP.v*
  - Synthesis:

    CHIP_syn.v,

    CHIP_syn.sdf,

    CHIP_syn.ddc
  - Report: *report.pdf*
- Compress all the files into one **ZIP** file
  - File name: DSD_HW3_學號.zip
  - EX: DSD_HW3_b06901001.zip
- Upload the file to Ceiba
- Deadline: 2020/05/06 24:00 ※Late submission is not allowed

Submission(2/2)

- *DSD_HW3_學號/*
  - *MIPS/*
    - *CHIP.v*
    - *CHIP_syn.v*
    - *CHIP_syn.sdf*
    - *CHIP_syn.ddc*
  - *RISCV/ (Optional)*
    - *CHIP.v*
    - *CHIP_syn.v*
    - *CHIP_syn.sdf*
    - *CHIP_syn.ddc*
    - *CHIP_RV32IC.v*
    - *CHIP_RV32IC_syn.v*
    - *CHIP_RV32IC_syn.sdf*
    - *CHIP_RV32IC_syn.ddc*
  - *report.pdf*

Fill out the form

- Choose the topic you want to do, due 2020/04/20 24:00
  - https://forms.gle/qTdBs66pvL1LXvEw7
- After you submitted your work, please fill out the following form to illustrate what you have done, due 2020/05/06 24:00
  - https://forms.gle/rgTx3CfPQMjQVKvw8

※You must fill out the form to finish HW3