Digital System Design

HW3:
Hardware Implementation of Single Cycle RISC-V

Speaker: Kane

Instructor:吳安宇教授

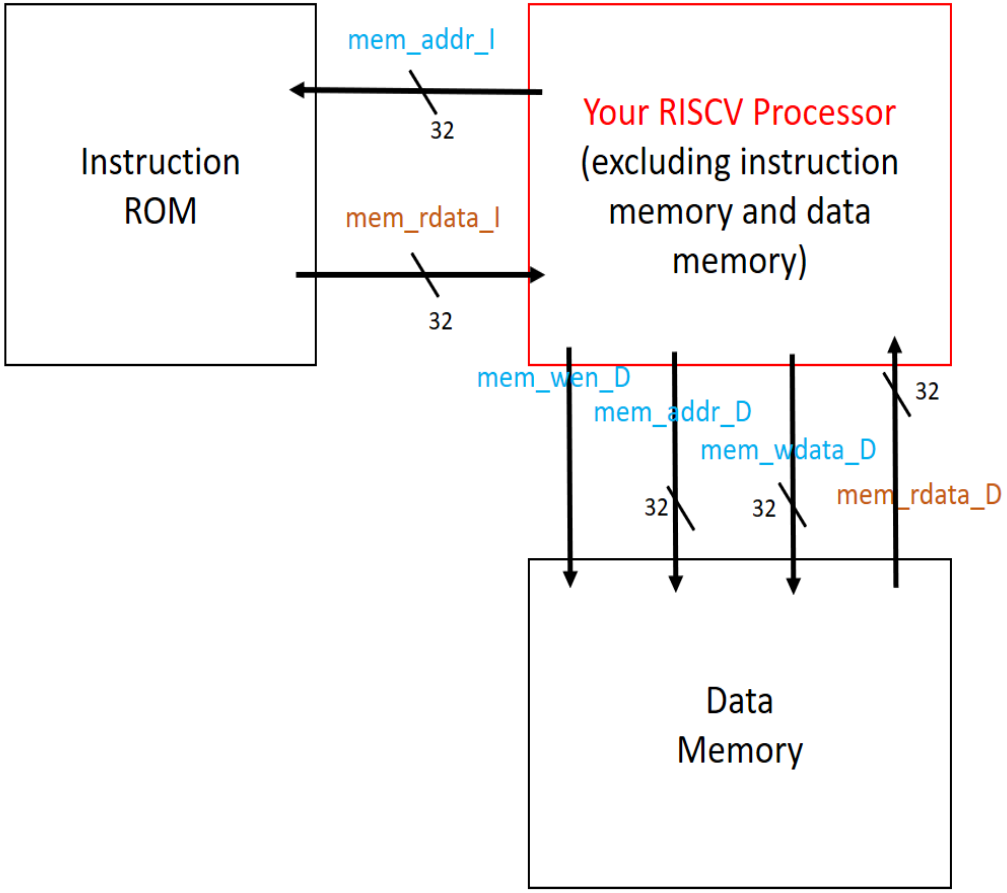
Date: 2021/04/08

Homework 3
Single-cycle RISCV Processor

Problem Statement

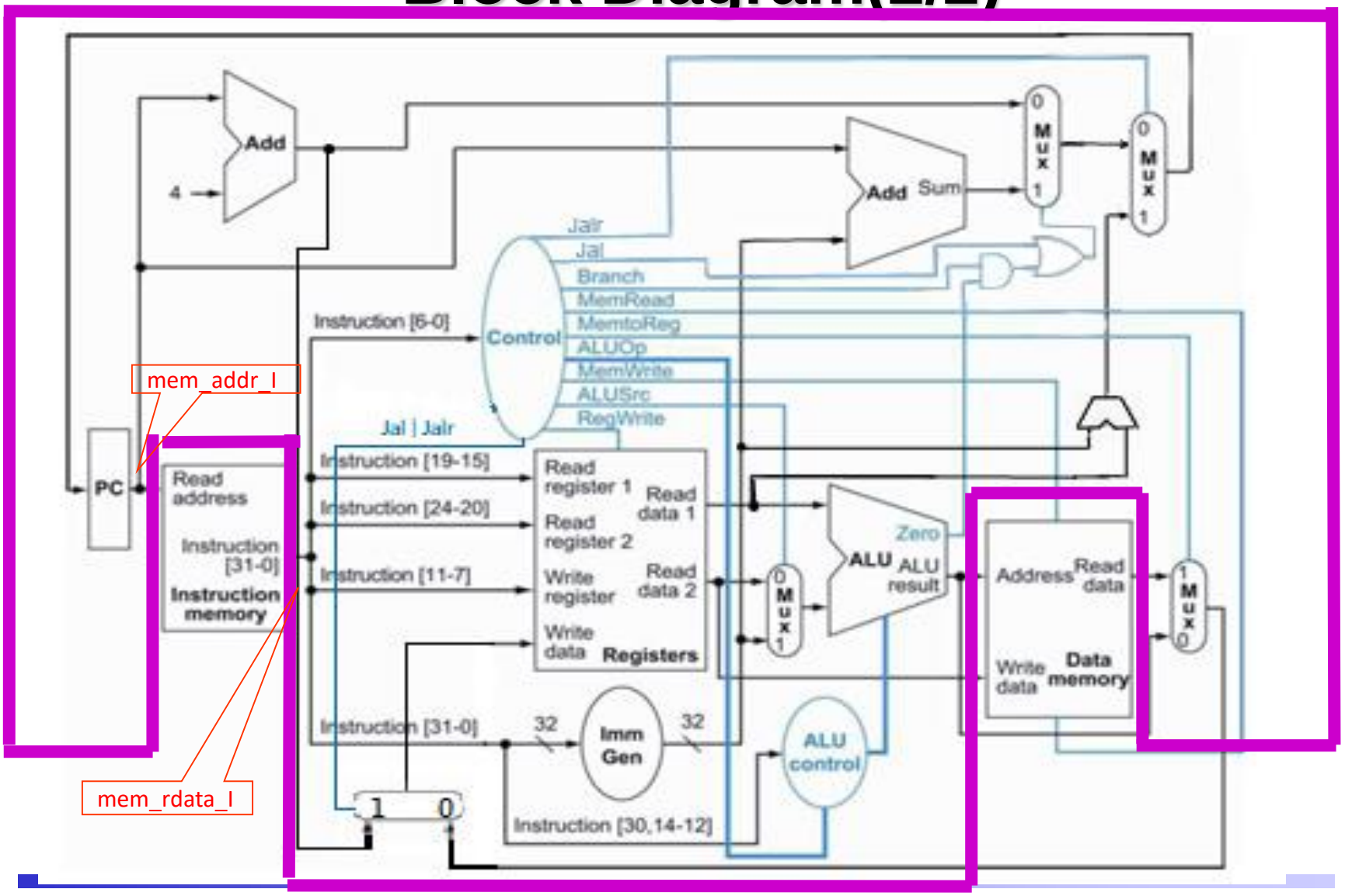
- Using Verilog, implement the single-cycle RISC-V processor:
  - Supported instructions:
    - add, sub, and, or, slt
    - lw, sw
    - beq
    - **jal, jalr**
- Most specifications are the same
- (Bonus) Compressed-Instruction(p.11)

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, writing data to D-mem when the next clk arrive; else reading 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 *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 x0 must be always 0

- There is no endianness issue!
  - If you store 32'h12345678 in x8,
    RF_8_w[31:0] = 32'h12345678

Memory Layout

- Instruction memory for RISC-V

```
03_24_00_00 // 000000000000_00000_010_01000_0000011
83_24_40_00 // 000000000100_00000_010_01001_0000011
33_04_84_00 // 0000000_01000_01000_000_01000_0110011
33_05_94_40 // 0100000_01001_01000_000_01010_0110011
```

- Data memory for RISC-V

```
0F_00_00_00 // 0x0000000F
14_00_00_00 // 0x00000014
00_00_00_00
00_00_00_00
```

- Conversion between big/little-endian
  - out[31:0] = {in[7:0], in[15:8], in[23:16], in[31:24]};

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, writing data to D-mem when the next clk arrive; else reading data from memory to chip.

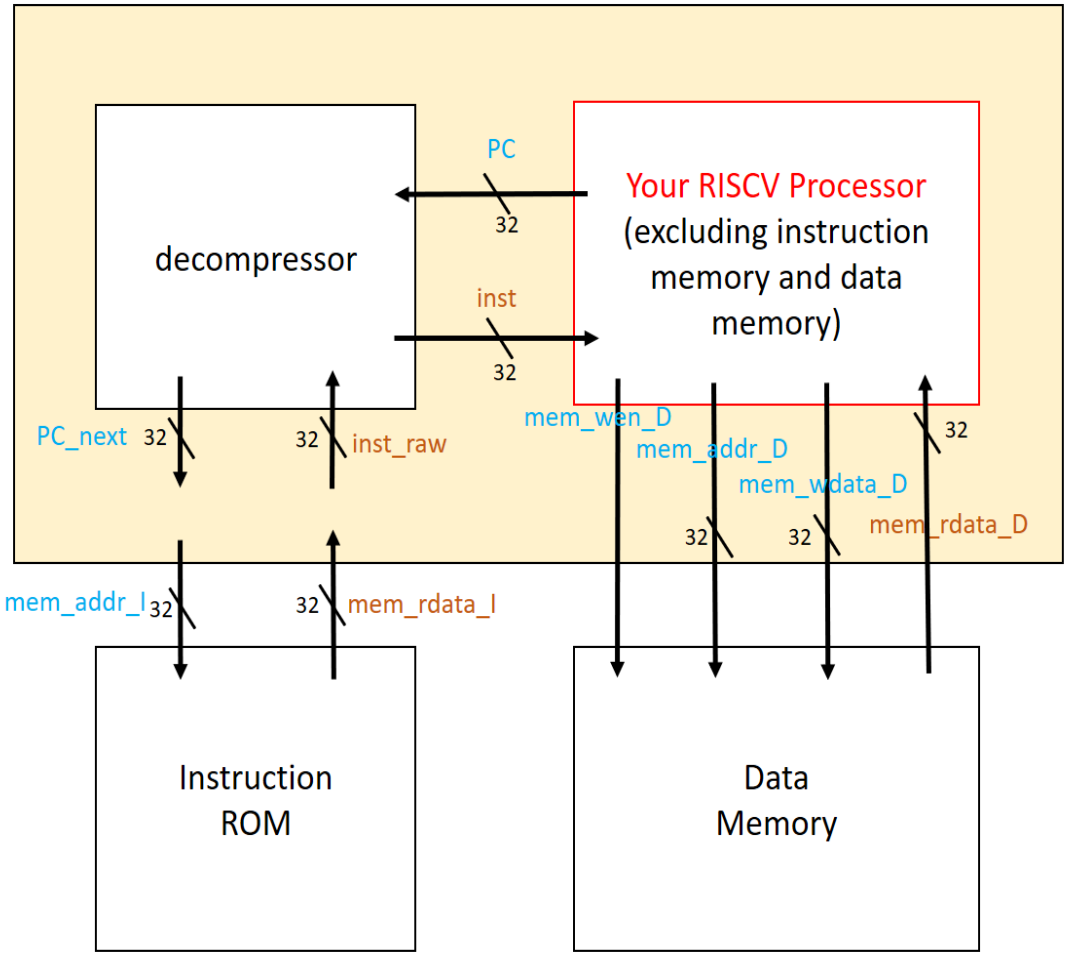
Memory Addressing

- In RISCV, the memory address is byte address.
- In Instruction ROM and data memory , the memory address is 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 RISCV_tb.v .

(Bonus) Compressed-Instruction

- Your RISCV also needs to support 16-bit compressed instructions.
- One of the suggestion is creating a decompressor module to distinguish whether the inst. is compressed or not. Recovering the 16-bit inst. to 32-bit inst. by 1-1 mapping, and then processing as usual.
  - Supported instructions:
    - C.add, C.sub, C.and, C.or
    - C.lw, C.sw
    - C.beqz
    - C.jal, C.jalr
  - Use *+define+RTL+RV32IC / +define+SYN+RV32IC* to test
  - More information: *Introduction_to_RISCV_C-extension&implementation.pptx*

Block Diagram



- Instruction ROM:
  contains the testing instructions
- Data Memory:
  contains the stored data
  - Used for testing your circuit
- decompressor:
  - Resolve the raw inst. from I-ROM and output the correct mem_rdata_I to CHIP
  - Maintain the PC and output the correct address to fetch inst. from I-ROM

Simulation & Synthesis

- **Check "RISCV/ 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.

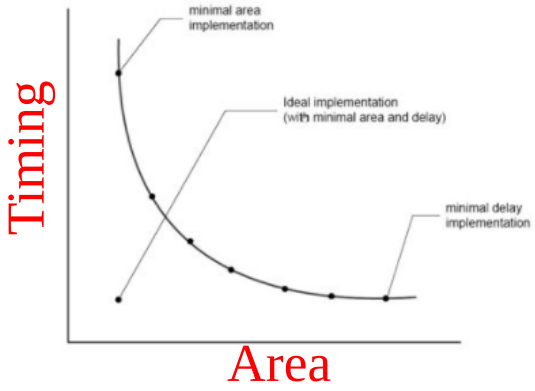
3. The tsmc13.v file is not allowed to be downloaded! Or you may offend the copyright protected by NTU & CIC!

Grading Policy

- RTL (40%): function correctness
- Synthesis (30%): correctness
- Report (10%)
- Area*Timing (20%)

- TA: 蔡文喬
- daniel@access.ee.ntu.edu.tw
- TA: 馬咏治
- kane@access.ee.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: 2021/04/29 24:00 ※Late submission is not allowed

Submission(2/2)

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

Appendix A

- Why Little endian?
  - Fetch with the same address if a given value is stored in different width
    - 32bit 0x0D0C0B0A
    - 64bit 0x000000000D0C0B0A
    - We can always fetch the lowest 32bit address
  - Mainstream
    - Intel x86