Project 4

1. Overview

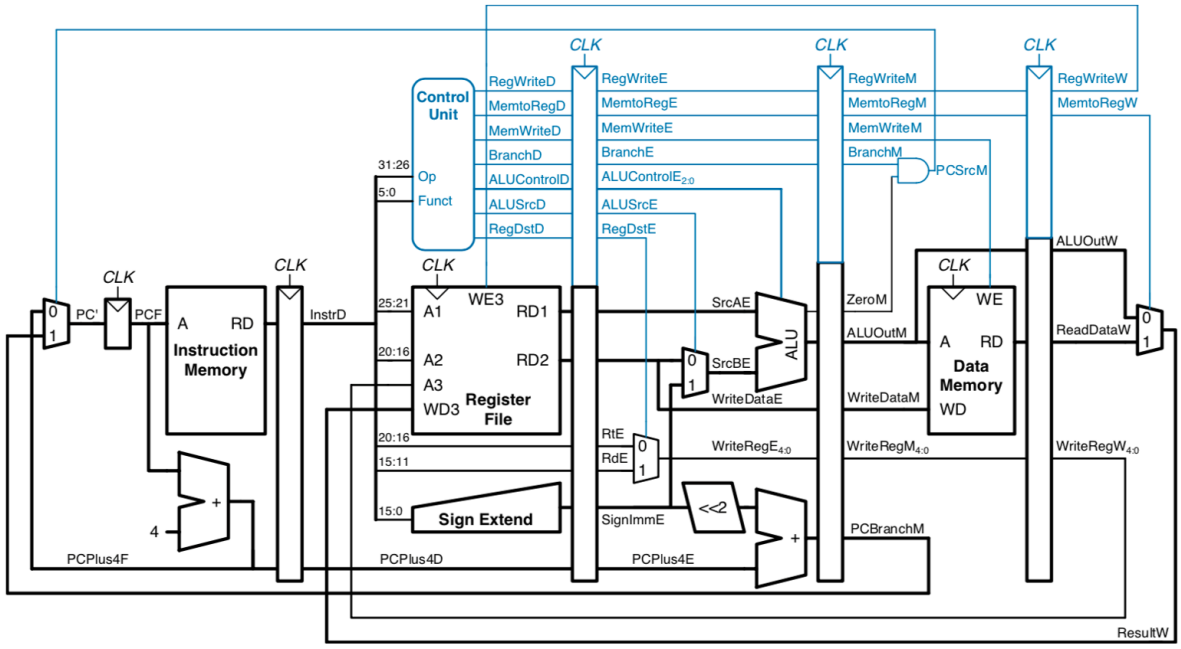
In project 4, you are required to implement a 5-stage pipelined CPU using Verilog language which can execute the MIPS instructions. We will provide you the instruction memory module and data memory module. You are going to implement the register file, ALU module, and the control unit. For testing, we will provide different workloads to test the correctness and performance of your CPU.

2. Requirements

2.1 5-stage Pipelined CPU

As we learnt in the lecture, the processor acquires 5 clock cycle to execute one instruction.

In the first clock cycle, the CPU read one instruction in the instruction memory with the new given pc address. In the second, the processor will divide the instruction to different parts and decode the MIPS instruction with the registers and control unit. The operation code and function code in MIPS instruction are sent to the control unit, which recognize the type of an instruction. In the third cycle 3, ALU will handle arithmetic, logical, shifting, and conditional branch instructions. In the fourth cycle, data will be fetched from or stored to the data memory by data transfer instructions. In the fifth cycle, data will be written back to registers if needed.



2.2 Supported MIPS Instructions

You are required to support the following MIPS instructions.

```
Data transfer instructions:
- lw, sw

Arithmetic instructions:
- add, addu, addi, addiu, sub, subu

Logical instructions
```

```
- and, andi, nor, or, ori, xor, xori

Shifting instructions
- sll, sllv, srl, srlv, sra, srav

Branch/Jump instructions:
- beq, bne, slt
- j, jr, jal
```

2.3 Usage of the Provided Module

We have provided the instruction memory and data memory to you. For the simplicity, the memory space of the instructions and data are separate. The usage is listed below. You can modify the module. However, you should preserve the **module name** ,**RAM**, and **DATA_RAM** variables.

Note: The address unit of both memory is in word. Hence, you should manually transfer address from byte space to word space.

```
module InstructionRAM // read-only
    ( // Inputs
      input  CLOCK // clock
    , input  RESET // reset
    , input  ENABLE
    , input [31:0] FETCH_ADDRESS

      // Outputs
    , output reg [31:0] DATA
    );
    //...
endmodule

module MainMemory //data memory
    ( // Inputs
      input  CLOCK // clock
    , input  RESET // reset
    , input  ENABLE
    , input [31:0] FETCH_ADDRESS
    , input [64:0] EDIT_SERIAL // {1'b[is_edit], 32'b[write_add],
32'b[write_data]}

      // Outputs
    , output reg [31:0] DATA
    );
    //...
endmodule
```

2.4 Testing Samples

You should evaluate the performance of your code and write your report based on these testing samples. The folder includes 8 testing cases and you can find the details in the "*About test.pdf*" file.

Regarding the code you submit. your CPU should display the whole Main Memory in the screen. The strategy to end your program is at the point where your CPU execute `32'hffffffff` instruction.

we will use your testbench module and an instruction file named "*instructions.bin*" to test your project. Please set the input file name as above in your project.

3. Grading

3.1 Requirements

1. Your project 4 should be written in Verilog only.
2. Your submission should contain at least two Verilog file:
    - CPU.v
    - test_CPU.v

   You can handle more than two Verilog files. **You will need to write your own makefile**.
3. You are encouraged to write your code in OO programming style.

4. Miscellaneous

4.1 Deadline

The deadline of this project is 05/01/2022 midnight.

4.2 Submission

You should put all of your source files in a folder and compress it in a **zip**. Name it with your **student ID**. Submit it through BB.

4.3 Grading

This project worth 20% of your total grade. There will be 20% extra credits in terms of the completeness of pipeline. The code you submit needs to have detailed comments, otherwise 10% of your grade will be deducted. The grading details of this project will be:

1. Correctness of 5-stage CPU functionalities. - 50%
2. Pipeline without countering any hazards. - 35%
3. Report - 15%
4. Pipeline with countering all hazards. - Max 20% (bonus)
    - The less clock cycle your CPU spends on the program, the more bonus credits you get.

NOTICE: If your code does not support makefile, you will lose 50% automatic.

4.4 Report

1. The report of this project should be **no longer than 5 pages.** Keep your words concise and clear.
2. In your report, you should include:
    1. Your big picture thoughts and ideas, showing us you really understand pipelined 5-stage CPU.
    2. A data flow chart explaining what you have extend.
    3. The high level implementation ideas. i.e. how you break down the problem into small problems, and the modules you implemented, etc.
    4. The implementation details. i.e. explain some special tricks used.

3. In your report, you should not:
    1. Include too many screenshots your code.
    2. Copy and paste others' report.

4.5 Honesty

We take your honesty seriously. **If you are caught copying others' code, you will get an automatic 0 in this project.** Please write your own code.