Project 4

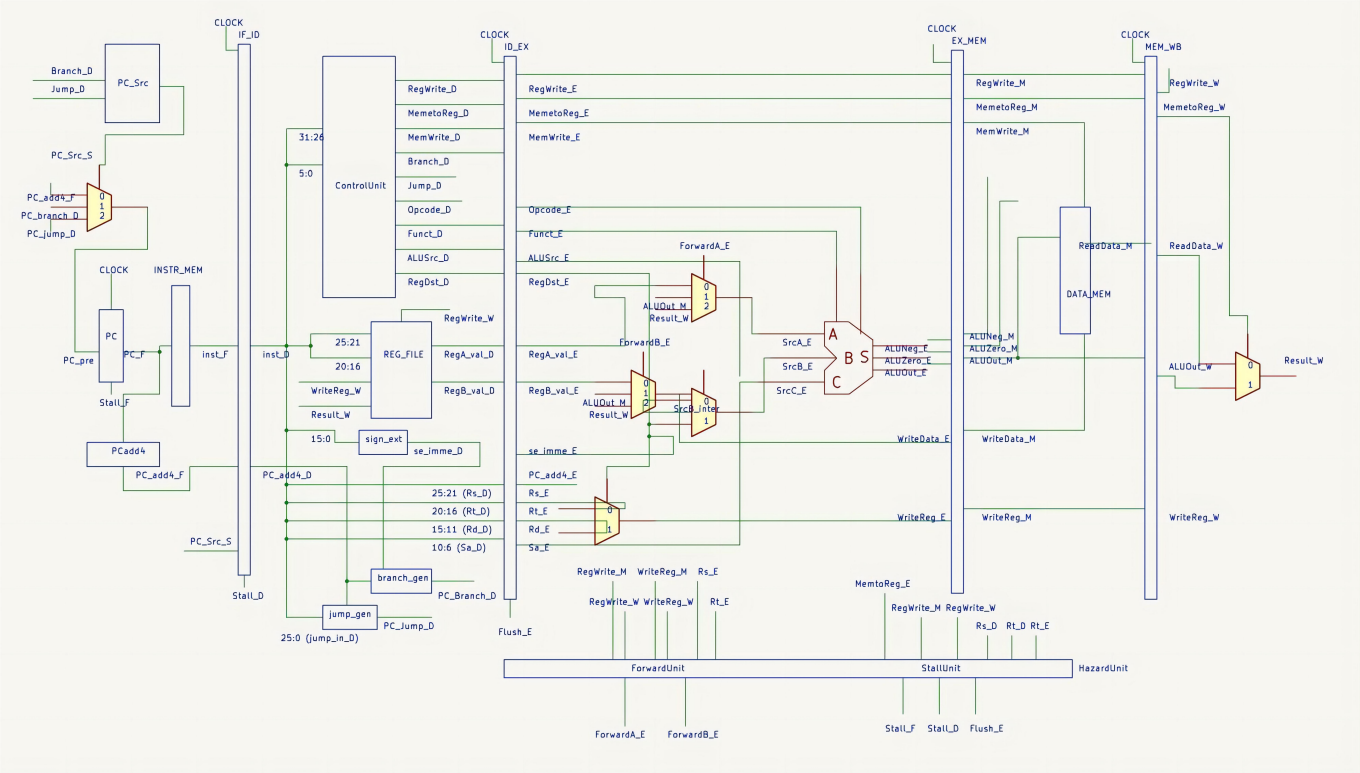
1. Overview

In project 4, you are required to implement a 5-stage pipelined CPU using Verilog language, which can execute the MIPS instructions. To reduce the difficulty of this project, we provide you some poorly written templates as references (Thanks to USTF Ziyu Xie for providing us with the templates). For testing, we provide 8 test cases to test the correctness and performance of your CPU. What you need to do is to implement a 5-stage CPU that can pass the aforementioned test cases. **We remark here that the provided template is only for reference. You are free to modify it as you wish or even create your own from scratch.**

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. The following figure is an example for the structure of the 5-stage CPU.



2.2 Supported MIPS Instructions

You are required to support the following MIPS instruction in your CPU:

> - lw, sw
> - add, addu, addi, addiu, sub, subu
> - and, andi, nor, or, ori, xor, xori
> - sll, sllv, srl, srlv, sra, srav
> - beq, bne, slt
> - j, jr, jal

3. What You Can Do In This Project

Basically, We provide a poorly writtrn templates as reference, which divide the five stages into 6 separate ".v" files, i.e., `IF.v`, `ID.v`, `EX.v`, `MEM.v`, `WB.v`, and a additional `Hazard.v` for solving different hazards. Moveover, we integrate all modules through `cpu.v`. Each individual ".v" files we provided is incomplete, with some code missing that can be filled in by you. For instance, in `IF.v`, you can fill in the missing pars to implement some multiplexer, as can be seen from the following figure.

```
module MUX2_BIT32 (
    input [31:0] A0,
    input [31:0] A1,
    input S,
    output [31:0] Y
);
    assign Y = Y_out(A0, A1, S);
    function [31:0] Y_out;
        /* Write your code here */
    endfunction
endmodule
```

All the places where you can fill in missing code will be indicated by `/* Write your code here */`. When all the missing parts are correctly filled in with code, you will have a fully functional 5-stage pipeline CPU.

However, the template provided by USTF was intended to reduce complexity and serve as a reference rather than a fixed standard form of this project. It itself may not be perfect, and certain parts of the code may contain specific ideas from the provider that might not be easy to comprehend. If you wish to customize certain areas, feel free to modify it yourself; it's not mandatory to follow what's already written in the template (in fact, we encourage you to write the code on your own as much as possible).

4. Testing Samples

You should evaluate the performance of your code. We provide 8 testing cases and you can find the details in the "About test.pdf" file. The strategy to end your program is at the point where your CPU execute 32'hffffffff

instruction. We will provide your the testbench module, an instruction file named `instructions.bin`, and a `data.bin` to test your project. What you can do with the given files is to put the instructions into `instructions.bin`, run the code through `make test` in terminal, and then see the output result in `data.bin`.

5. Grading

5.1 Submission Requirements

- Your project 4 should be written in Verilog only.
- Your submission should contain the following file:
  - The complete implementation of the CPU via Verilog (which can consist of multiple separate .v files)
  - Report.pdf
- If necessary, you can also rewrite the makefile.
- You should place all of your source files in a folder and compress it into a .zip file. Name it with your student ID and submit it through BB.
- **Submission Deadline: 23:59, December 14, 2023**
- **If you submit after the deadline, you will lose 10 points for each day of delay, and if you submit more than 3 days late, you will receive a score of 0.**

5.2 Report

- The report of this project should be `no longer than 5 pages`. Keep your words concise and clear.
- In your report, you should include:
  - Your big picture thoughts and ideas, showing us you really understand pipelined 5-stage CPU.
  - The implementation details. i.e., explain some special tricks used.
  - Which of the provided test cases can your code pass.
- In your report, you should not:
  - Include too many screenshots of your code.
  - Copy and paste others' reports.

5.3 Grading Details

- This project worth 15% of your total grade. The grading details of this project will be:
  - Correctness of CPU functionalities `80%`. (Your grade will be determined by the eight test cases provided. For each test case that your code successfully passes, you will receive 10% of the total score.)
  - Report `20%`

5.4 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.