The Chinese University of Hong Kong, Shenzhen

CSC3050

Computer Architecture

Project2

Author:
Shi Wenlan

Student Number:
ID

May 14, 2021

1 Introduction

I wrote a set of verilog files, these files take file *instructions.bin* as input, read the machine code instructions in the file, and then execute these MIPS codes in pipeline. When all the MIPS instructions are executed, this set of verilog files will output the main memory to the *main_memory.txt* file. This set of verilog files can solve most hazards (except hazards of jump and branch).

2 Big picture thoughts and ideas

In order to speed up the operation of the CPU, we adopted the idea of pipeline, dividing the execution of an instruction into five stages: IF, ID, EX, MEM, and WB. Although this cannot reduce the execution time of a single instruction, it can increase the number of simultaneous execution instructions, thereby achieving the purpose of speeding up.

The content of the tasks to be performed by each stage and the general circuit diagram of the overall CPU are shown in the following figure.

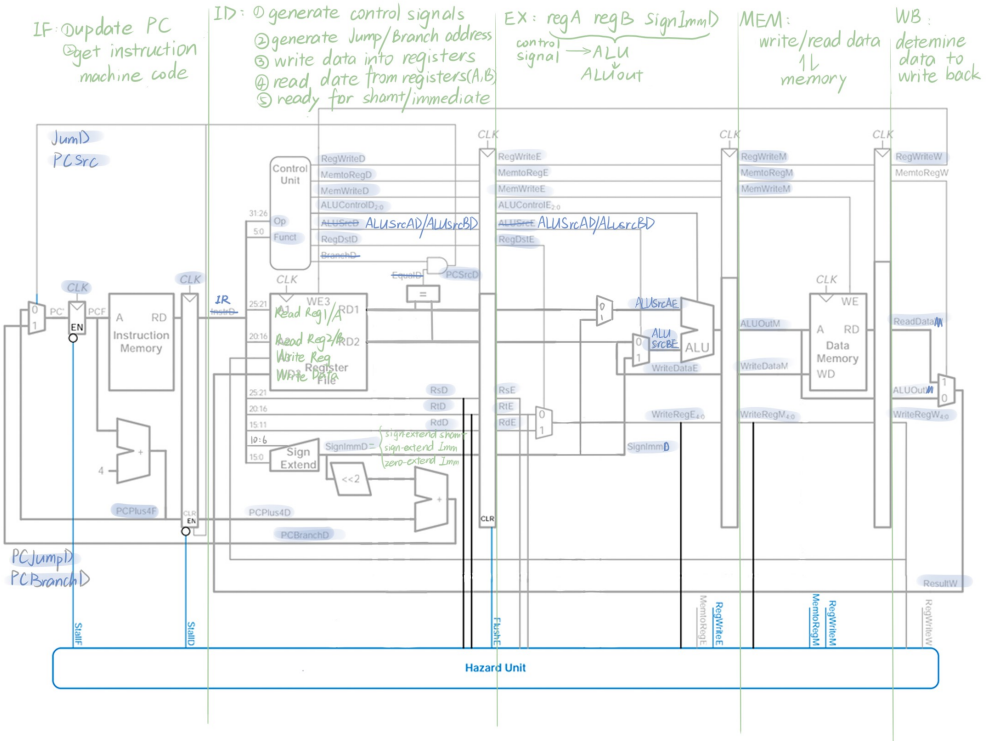

Figure 1: Big picture thoughts and ideas

3 A data flow chart

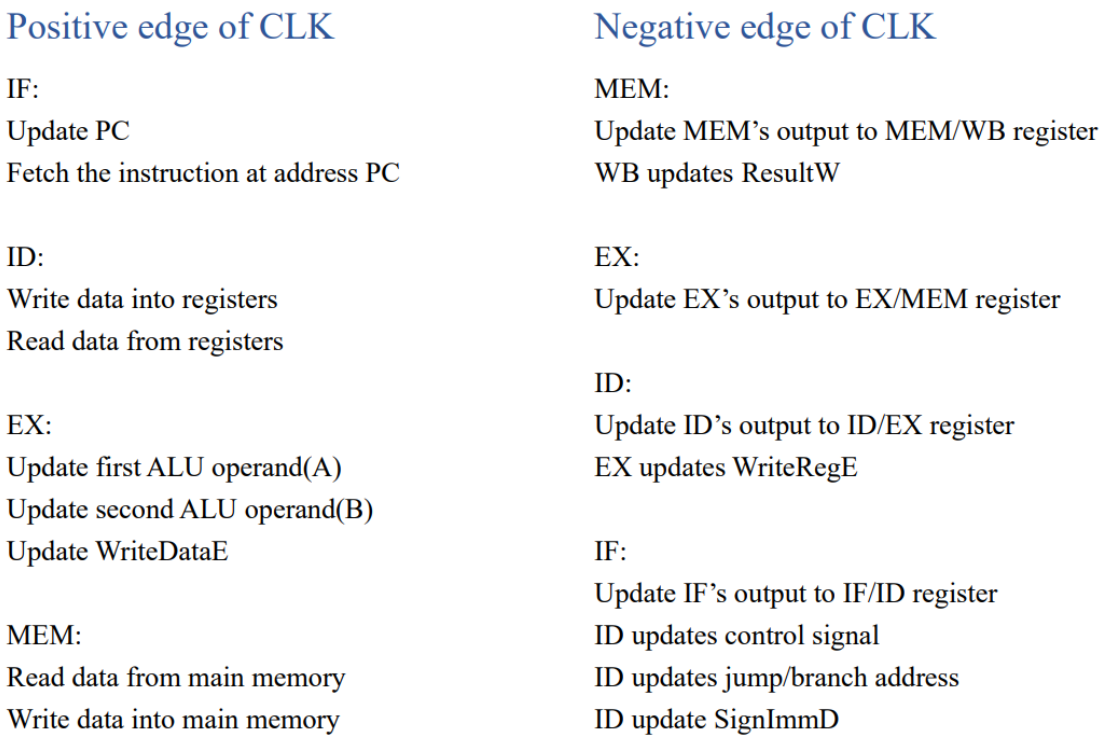

Figure 2: A data flow chart

4 High level implementation ideas

About relation between modules:

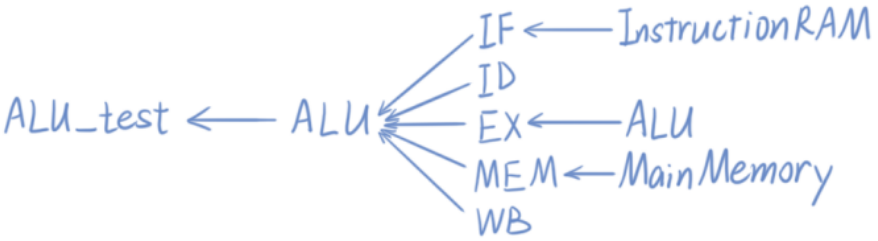

Figure 3: Relation between modules

About the design of major modules:

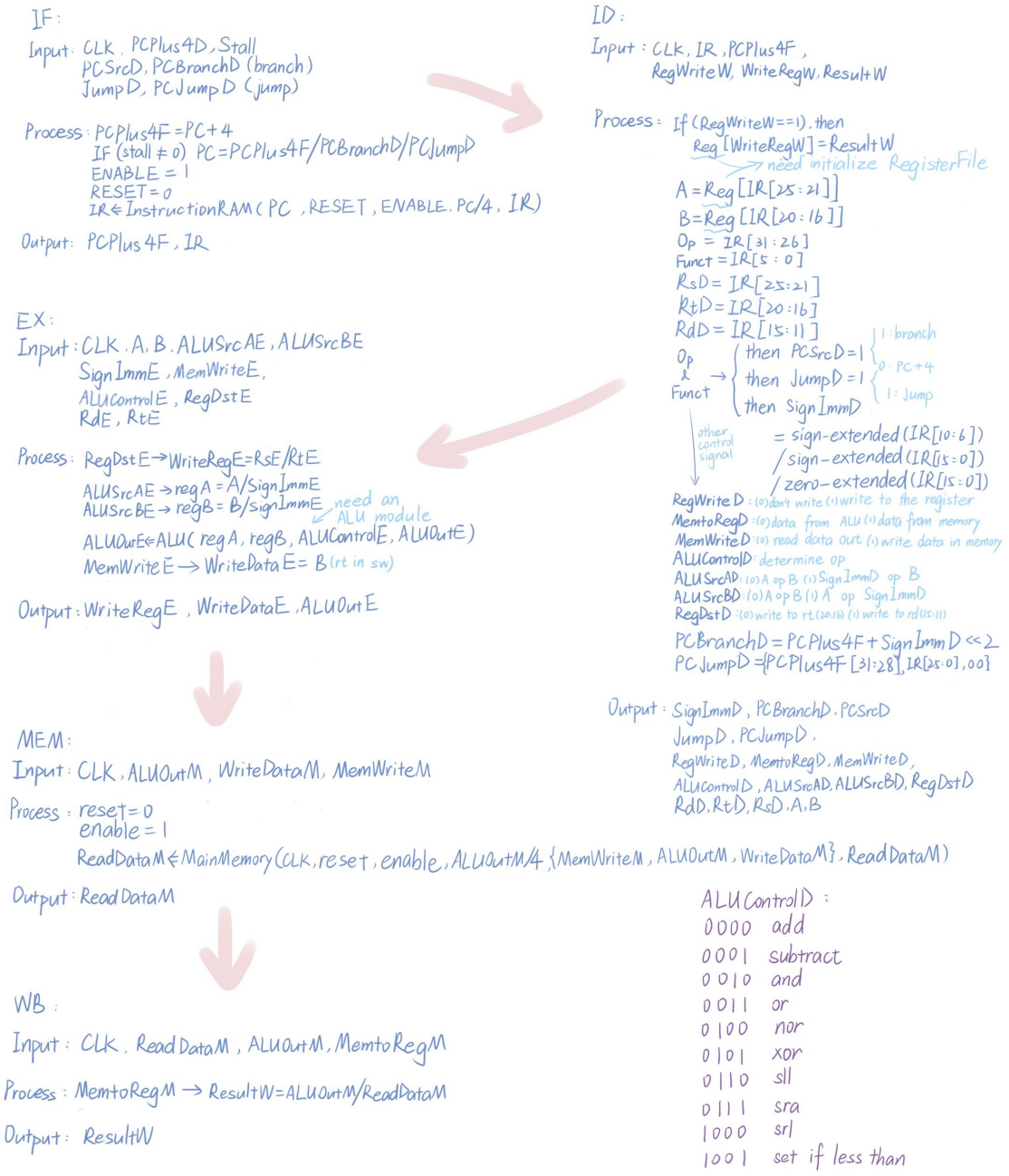

Figure 4: Design of major modules

5 Implementation details

About solving hazard: I only used stall to solve hazard, and did not use fowarding. In the ID module, RsD, RtD, and RdD are all wire types, and the input is directly related to the output. So, at the negative edge of CLK after the positive edge of CLK where IF gets the instruction, after the contents of registers between two stages are updated, RsD and RtD will be updated accordingly. The $WriteRegE$ EX module adopted the same operation, the goal is to get the variable value as soon as possible for the stall comparison. Then the CPU can judge whether there is a write back requirement in the EX, MEM, or WB stages. If there is a write back, then check whether the register address for written back conflicts with the RsD or RtD. If there is a conflict, the stall signal will be set to 1, preventing the PC in the IF from being updated when the next positive edge CLK occurs.
And in order to complete the update of the jump/branch signal before the next PC update, the trigger conditions for the decode and control signal generation codes in the ID module are set to the input change instead of a certain edge of the CLK. In such a design, the generation time of the stall is later than the control signal and address generation of the jump and branch, which makes the stall unable to handle the hazard of the jump and branch.

6 Operation manual

Enter terminal and enter commands as shown below:

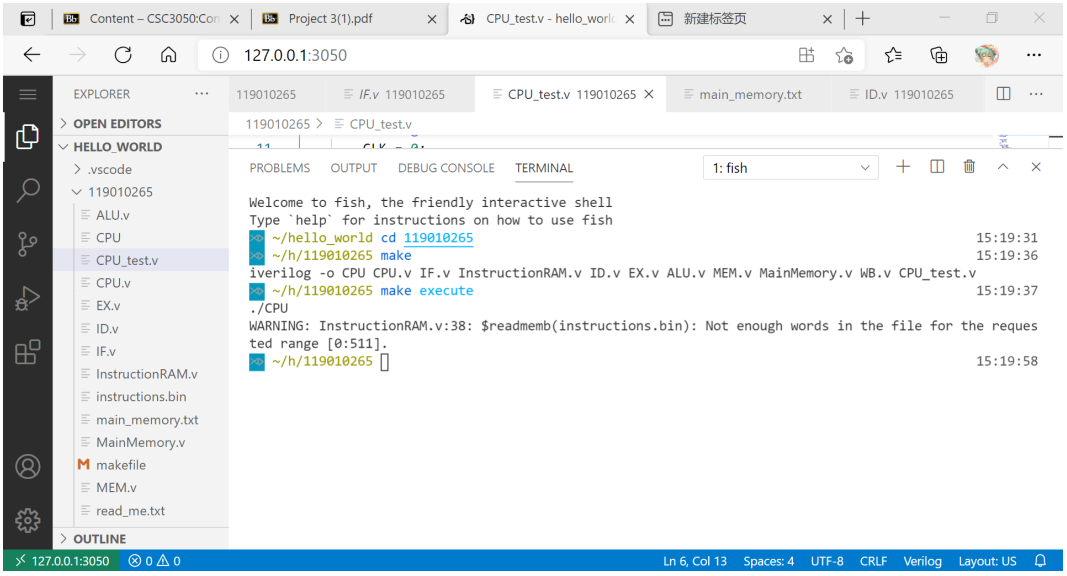

Figure 5: Operation manual 1

Open the file named $main_memory.txt$ to view the results:

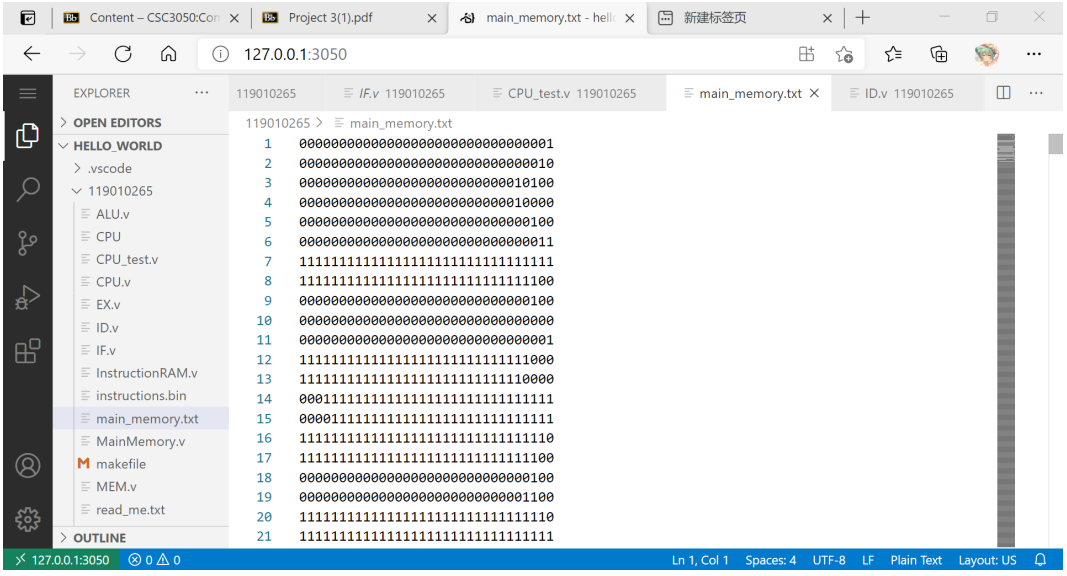

Figure 6: Operation manual 2

This code can pass the first 6 tests.

7 Conclusion

This project helped me understand how the CPU execute MIPS instructions in pipeline, and made me have an in-depth grasp of Verilog writing and debugging.