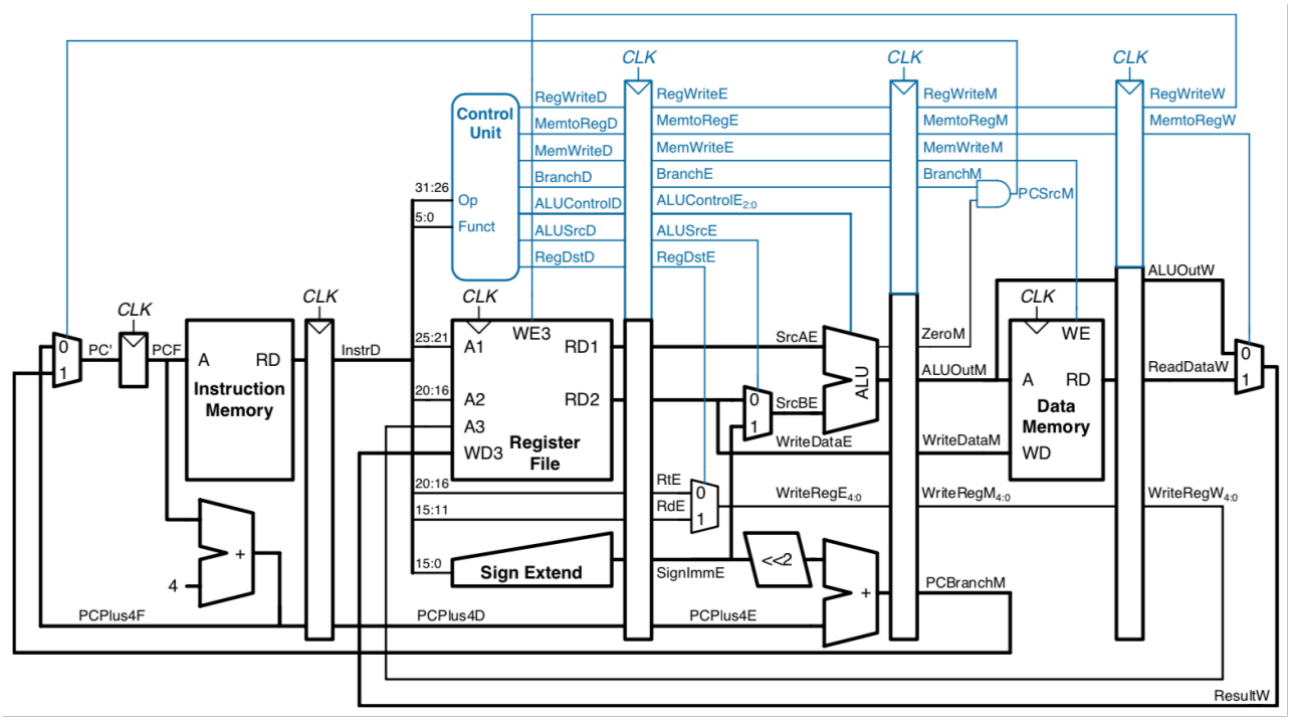
Program 4 Instruction

Overview

You have learnt MIPS instructions and finished a ALU module using verilog in the 3 projects before. In this project, you will deal with a pipeline processor which can execute MIPS instructions with verilog. Compared with Program 3, you should design a whole CPU module, including the clock, instruction memory, registers, ALU, data memory and control unit. You should define the MIPS instruction and build the data path in a verilog file and test the MIPS instructions in the test file, then display the result like Program 3.

Block Diagram



We can see that, for each instruction, this processor need five clock cycle to execute it. In the first clock cycle, the cpu read one instruction in the instruction memory with the new given pc address. In the second and third clock cycle, it is what we have done in the third project: 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 decide how to execute in the next part(Cycle 3). For data transfer instructions, you can read and write data between the data memory and registers(Cycle 4 & 5). For arithmetic or logic instructions, you can use the ALU module to get the answer, and write it to the registers. For conditional branch instructions, you can do comparison in the ALU module and branch to the address in the MIPS instruction. For jump instruction, you can directly jump to your target address(which was shown in my slides).

For this project, you should be able to finish an entire cpu by yourself. If you cannot finish a pipeline processor, Please at least handle a runnable single-cycle processor which can run these instruction:

The CPU must support:
1) Data transfer instructions:
- lw, sw
2) Arithmetic instructions:
- add, sub, addu, sub
- addi, addiu
3) Logical instructions:
- and, or, nor, xor
- andi, ori
4) Shifting instructions:
- sll, srl, sra
- sllv, srlv, srav
5) Branch/Jump instructions:
- beq, bne, slt (with comparison so put it here)
- j, jr, jal

You should handle two verilog files:
- CPU.v
- test_CPU.v

And your project report.
As the Project 3, you should also implement the whole cpu file and test file by yourself and analyze the final result in your report.

Grading

Support the Arithmetic/Logic instructions - - - - - - - - 45%
- ALU(Project 3) - 30%
- A clear data flow with sequential logic -15%

Support the Data transfer instructions - - - - - - - - 10%
- Data memory
- General registers

Support the Branch/Jump instructions - - - - - - - - 20%
- ALU part and data flow
- Changing of pc

Finish a pipeline processor - - - - - - - 15%

Project report - - - - - - - 10%

*You will get a 10% bonus if you solve the hazards in the pipeline processor.