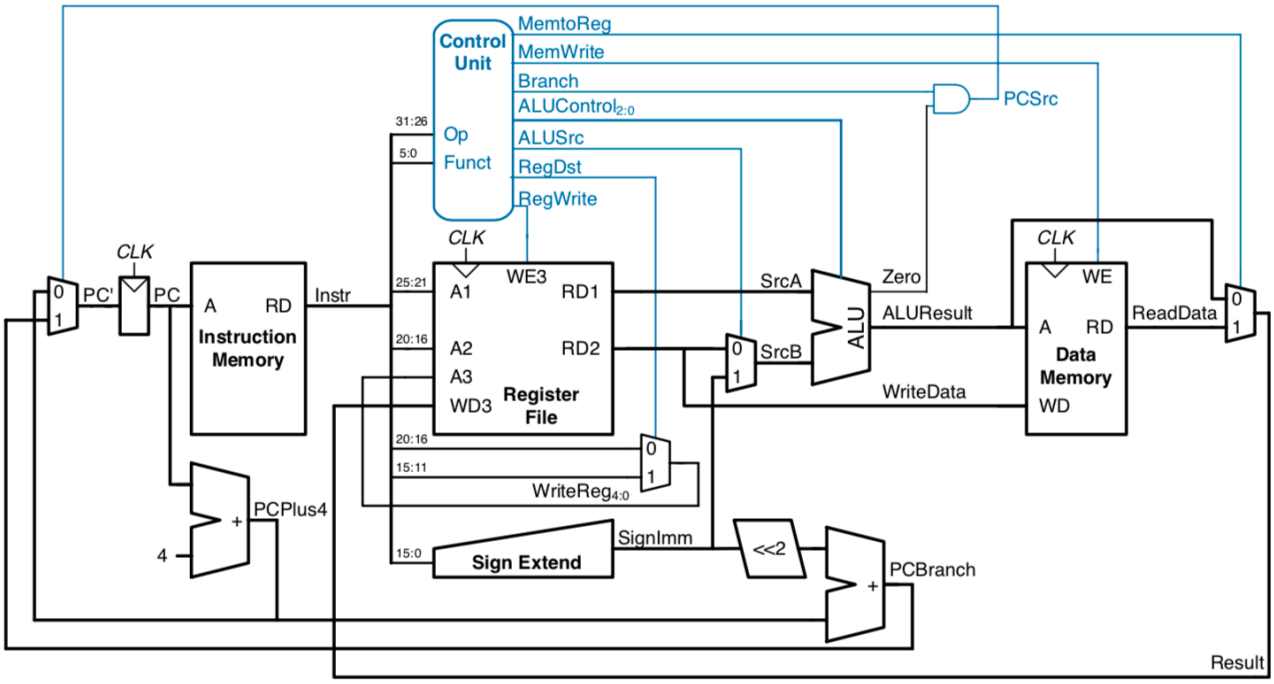
Programme 4 Instruction

Overview

You have learnt MIPS instructions and finished a ALU module using verilog in the 3 projects before. In this programme, you will deal with a single-cycle processor which can execute MIPS instructions with verilog.
Compared with programme 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 datapath in a verilog file and test the MIPS instructions in the test file, then display the result like programme 3.

Block Diagram



The CPU must support:
1) Data transfer instructions:
- lw, sw
2) Arithmetic instructions:
- add, sub, addu, subu
- addi, addiu
3) Logical instructions:
- and, or, nor, xor
- andi, ori
4) Branch/Jump instructions:
- beq, bne, slt
- j, jr, jal

We can see that, in one clock cycle, the cpu read one instruction in the instruction memory 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. For data transfer instructions, you can read and write data between the data memory and registers. 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.

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

And your project report.
I will give you example verilog code in my slides and you should implement the whole cpu file and test file by yourself and analyze the final result in your report.

Grading

Support the Data transfer instructions - - - - - - - - 10%

Support the Arithmetic/Logic instructions - - - - - - - - 50%
- R-type unsigned arithmetic operations - 10%
- R-type signed arithmetic operations - 10%
- I-type arithmetic operations - 10%
- R-type logical arithmetic operations - 10%
- I-type logical arithmetic operations - 10%

Support the Branch/Jump instructions - - - - - - - - 30%

Project report - - - - - - - - 10%

*You will get a 10% bonus if you implement a pipeline cpu.