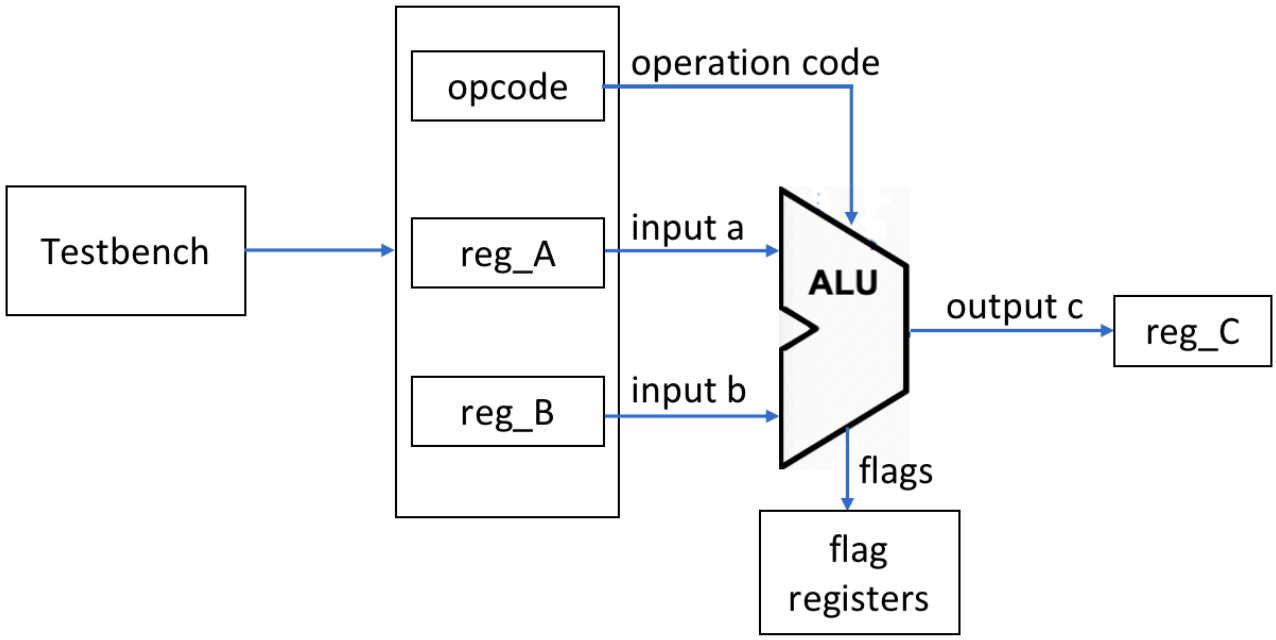
Program 3 Instruction

Overview

In the class, the teacher introduced the Arithmetic and Logic Unit(ALU) to us. This module is to do math co-processing. In this program, we will use verilog to complete it.
I will give you examples of verilog module file and testing file. You should add other operations into the code file and analyze the result.

Block Diagram



In the diagram, we also have test bench. In the CPU, we should get information from the instruction and find the data flow of each instruction. However, in the ALU module, the input is sometimes not only rely on the instructions. Therefore, the test bench should also contain the value of relevant registers.

The ALU must support:
- add, addi, addu, addiu
- sub, subu
- mult, multu, div, divu
- sqrt
- and, andi, nor, or, ori, xor, xori
- beq, bne, slt, slti, sltiu, sltu
- lw, sw
- sll, srl, sla, sra

I will give you the example of sll.

From test bench to the register, with different kind of instructions, we should send different thing to the ALU module. You should at least show me the value of these registers in the diagram. You can also display other important value in your own data flow and explain it in your report. I will check the value with my test cases.

The flag registers includes zero flag, negative flag and overflow flag. The negative flag will appear when you do subtraction with unsigned values or other arithmetic with signed values. In the additional part, there may be overflow and the overflow detection will better improve our ALU module. The zero flag can easily decide the changing of pc in the instructions like bne, beq.

You should handle at least two verilog file:
- ALU.v
- test_ALU.v
And your project report.
Because maybe some of you will use other modules(adder, etc.), you can handle more than two verilog file. However, I will modify your test file with my own code to test your file.
I will give you example test bench and you should extend the test bench by yourself and analyze the final result in your report.

Grading

Support the Arithmetic/Logic operations - 60%
- Basic operations with value of general registers
- Load and store
- Special operations with other registers (multiplication with Hi, Lo, etc.)

With a clear data flow from instruction to ALU - 20%
- Control part
- Sign extension
- Zero extension

With special handling for flags - 10%
- negative flag
- Zero flag
- Overflow detection

Project report - 10%