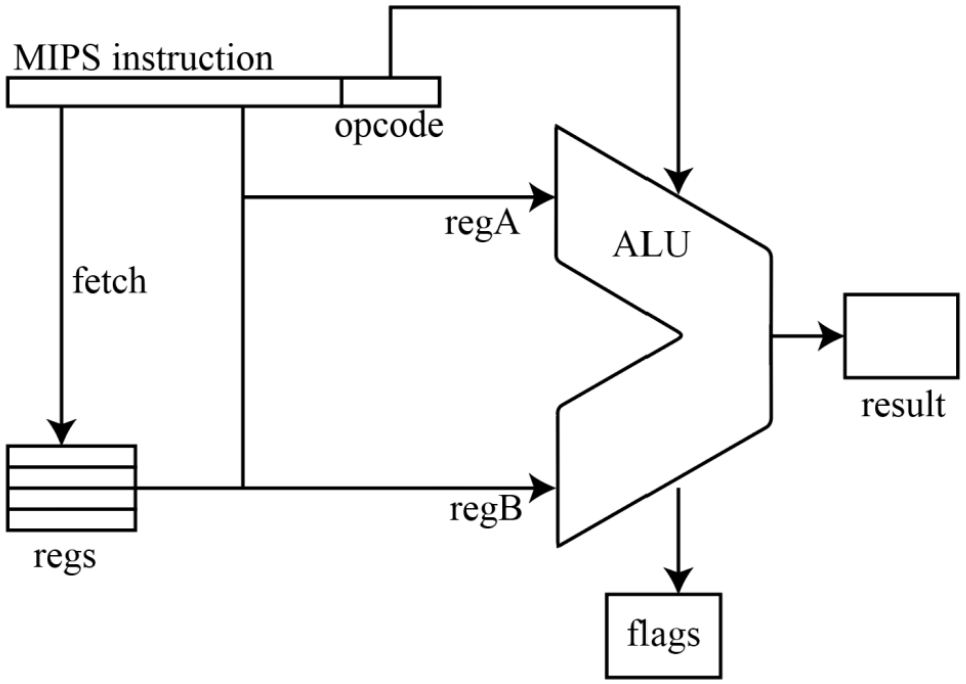
Assignment #3 Documentation

118010488

Hyeonwook Kim

Overview

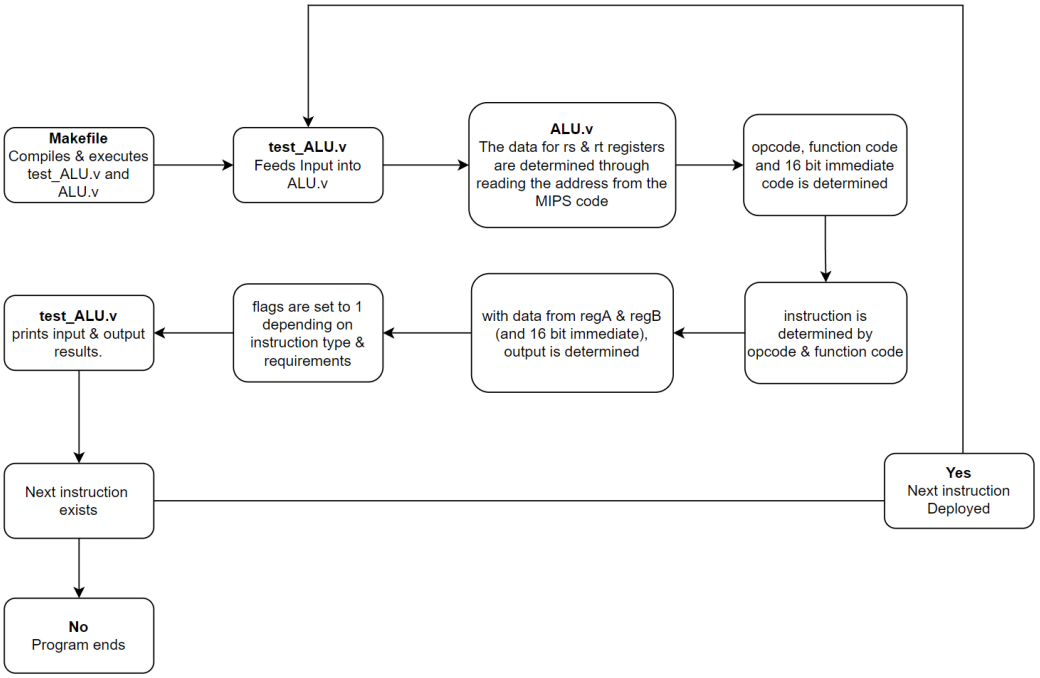
The following programming project is an implementation of the Arithmetic and Logic Unit or ALU. Icarus Verilog was used to simulate the ALU and the registers. Included is ALU.v, test_ALU.v and Makefile. ALU.v takes a 32 bit MIPS instruction and 2 register data inputs (regA, regB) and outputs a 32 bit result and a 3 bit flag code.



When ALU.v is activated with a MIPS instruction and 2 register data values, the code first determines the address of registers rs and rt. instruction[25:21] and instruction[20:16] is read. If the address is 000000, the register is assigned the data of regA. If the address is 000001, the data of regB is assigned. The opcode, function code and 16 bit immediate is also determined.

Depending on the value of the opcode and / or function code, the type of instruction is determined by the module. The input data of regA, regB (and the 16 bit immediate for I type instructions) are used to get the output. For the overflow flag, add , addi , and sub triggers the flag. For the zero flag, beq and bne triggers the second flag. slt , slti , sltiu , sltu instructions use the negative flag.

Workflow



The makefile compiles the code and initializes test_ALU. test_ALU deploys a series of inputs with MIPS code and register data for regA and regB into ALU.v. For each input ALU.v receives, case() is used to determine the register data for rs and rt through the address written in the instruction MIPS code. After this, the opcode, function code and 16 bit immediate are also determined through the 32 bit instruction input. Then the instruction type is determined through the opcode and function code.

Here, case() is also used again. An opcode of 000000 will make the module determine the instruction to be an R type, and another series of case() lines searches the function code and determines the instruction. Otherwise, the opcode is used to determine the I type instructions.

Once determined, the data from regA, regB and/or 16 bit immediate is used to generate the correct output. Not all outputs are complete outputs. For example, instructions such as *bne* outputs the 16bit immediate multiplied by 4, since PC-4 is added outside the MIPS ALU. Likewise, *lw* and *sw* only outputs the address designated by the 16 bit immediate, and does not store or load the word themselves, since this is a task done outside the ALU.

Depending on the instruction type, flags are also triggered. For example, the add instruction triggers the overflow flag when the addition of two numbers of equal sign results in the opposite sign. Once the flags and output bits are determined,

test_ALU prints the inputs and outputs, and the next input is fed into the alu module until all instructions have been completed. The test_ALU.v file contains instruction inputs for all the possible instruction types. Icarus Verilog was used for compilation.