Modules:

Move – The module takes R2 as an input, then set the output, which is R1 to the input.

Not – Using parameterizing the basic logic not gate, it takes each bit as input and go through the not gate and coming out as a result.

Add – Basically I used a Full adder which I used in previous lab. All the function of this adder is equivalent.

Subtract – I used Full adder but as an input b is negated. In other words, b = b’ + 1. A + (b’ + 1) is the same thing as subtraction. The function is same as the adder.

Or – Take a and b as two inputs and compare each bit. If one of them is 1, then output is 1. If none of them is 1, then output is zero.

Multiplexer – Multiplexer takes as input the output of the command modules, as well as the ALU opcode as select. Basically, it decides which function should be operated based on opcode of ALU.

ALU – ALU combines all the function that we need. We just need to provide correct opcode to ALU then the result will come out.

Register – It’s D flip-flop registers that we used in pre-lab.

Top Module – Assign zero flag (knowing the output of Register is 0 or 1), overflow (unable to express in given bits), and c\_out. Also, top module stores the value that came out of ALU into the register. Without this process, there will be no value available at the end.

DD function. DD function.

A picture containing text, electronics, computer

Description automatically generated

I compared all the possible combinations including the corner cases which was given in the lab instruction. All the functions are working properly. The verification and flags are also working properly. From left to right, I tested MOV, NOT, AND, OR, SUB, ADD function.

At the bottom of the page, I inserted two pictures to show the design of the ALU model that I create.

Diagram

Description automatically generatedDiagram, schematic

Description automatically generated