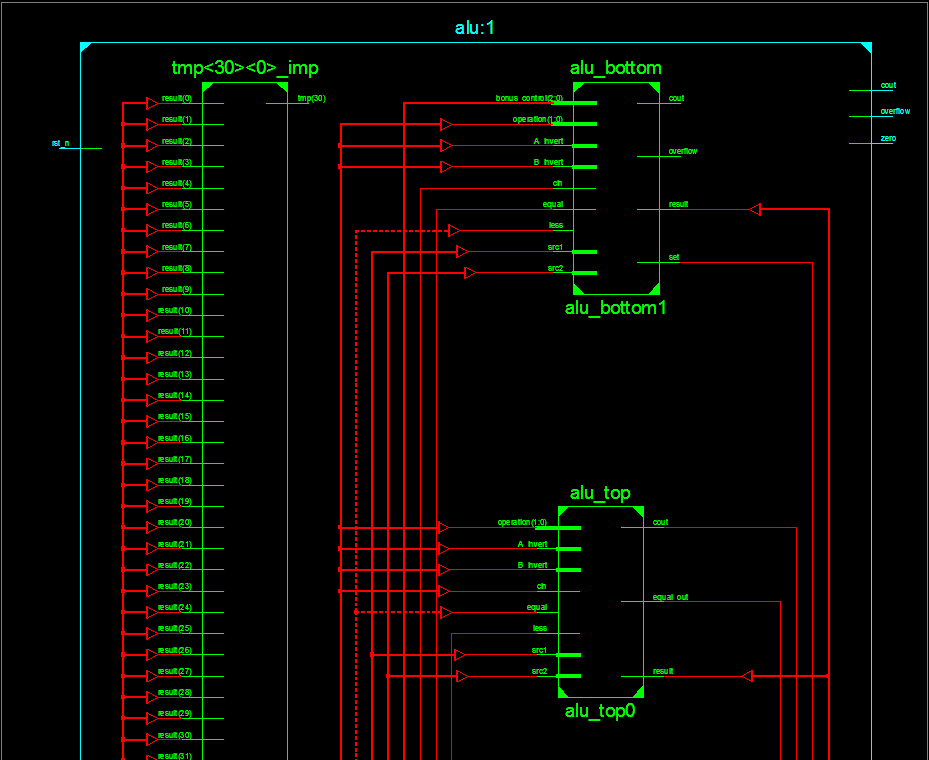
**Computer Organization**

**Architecture diagram:**

Partially amplified.



Whole diagram



**Detailed description of the implementation:**

Inside the 1-bit alu module:

Inverter using xor.

And, or gates with gate-level description.

Nand gate from or, nor gate from and.

Full adder from xors, ands, and ors.

The set bit from the arithmetic result in the last 1-bit alu. (To be used as the less bit in the first alu, others take in 1’b0 instead.)

Multiplexer to choose the result from all results produced by the alu.

Or gate to see if the addition outcome and the previous outcomes were all 0s, and pass on the result to the next alu.

In the bottom 1-bit alu:

Gate-level description for overflow bit.

Multiplexer to choose the set bit using previous “equal” results and the addition result.

In the 32-bit alu:

Connected 32 1-bit alus (carry-in and carry-out)

A 32-to-1 or gate tree (with a not gate) to produce the zero flag.

Decode A\_invert, B\_invert, cin, and operation message from ALU\_control.

Set all outputs to 0 if rst\_n is 0.

All of the above and basically everything required.

**Problems encountered and solutions:**

We used the zero flag as an indicator for Seq, but it created a circuit loop thus wouldn’t work.

Emailed TA, added a wire through the 32 1-bit ALUs to collect the message indicating equal.

**Lesson learnt (if any):**

Gate-level descriptions are frustrating.