Atharv Tekurkar, Ansh Singhal
CSC270 - Final Project: 16-bit CPU

- **A 16-bit CPU with the basic components:**
  - ➔ A "basic" ALU capable of addition, subtraction, and logical operations.
  - ➔ Support for both R-type and immediate instructions.
  - ➔ A register file with a minimum of 8 registers.
  - ➔ Instruction Memory (and program counter) and Data Memory.
  - ➔ An assembler/linker

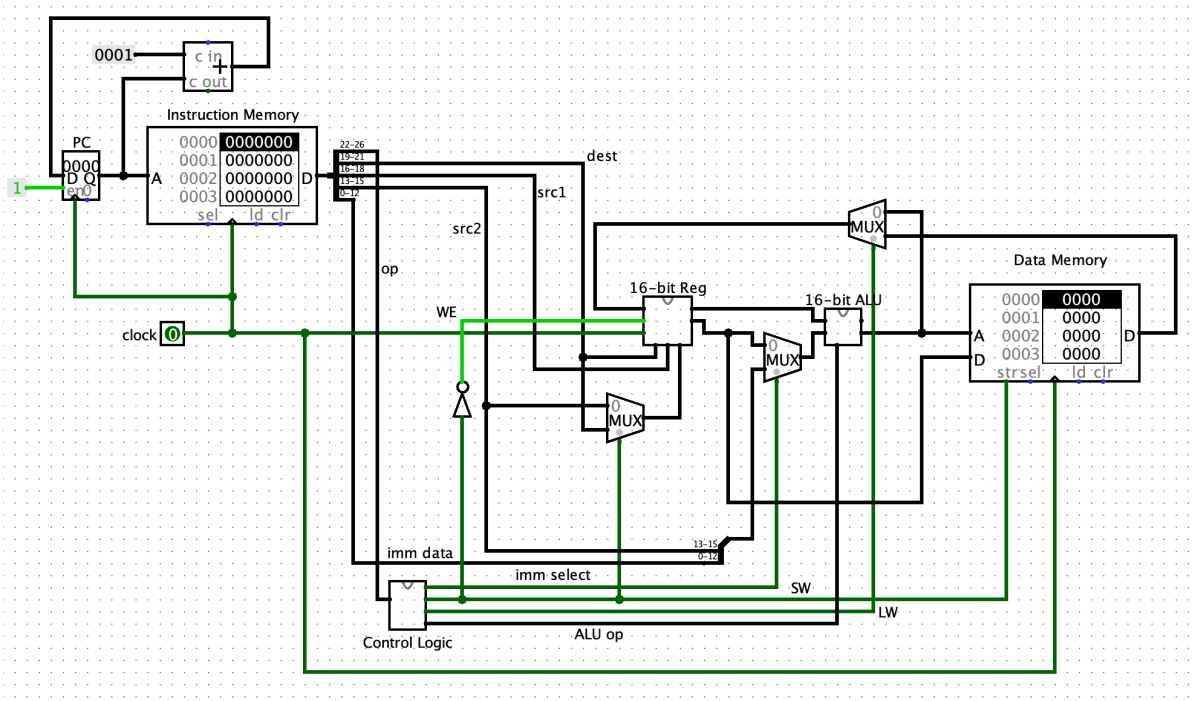

Fig: 16-bit CPU

Assembler:
https://github.com/anshsinghal2002/16-bit-mips-assembly/blob/master/assembler.py

The Control Logic used for the above circuit was a simple one that used the 5 MSB of the 27-bit instruction word as the Control Logic OP Code, which branched them out accordingly.
The ALU used can only perform 4 functions - ADD (00), AND (01), OR (10), and SUB (11), which was given by the 2 LSB of the Control Logic OP Code.

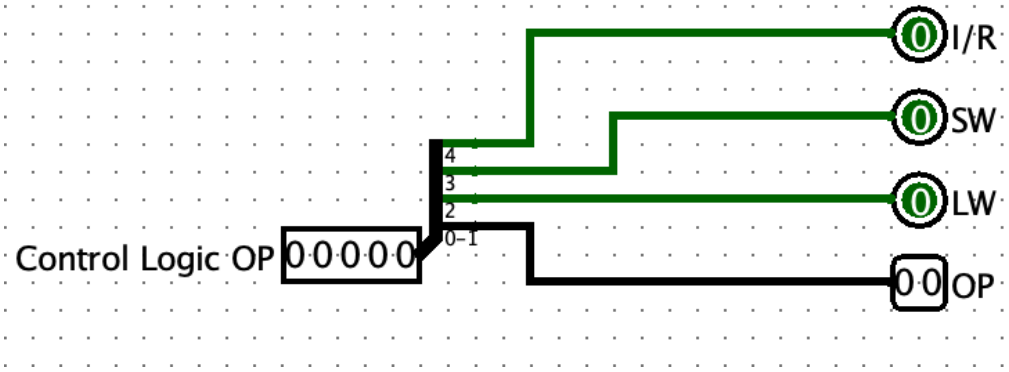

Fig: Control Logic

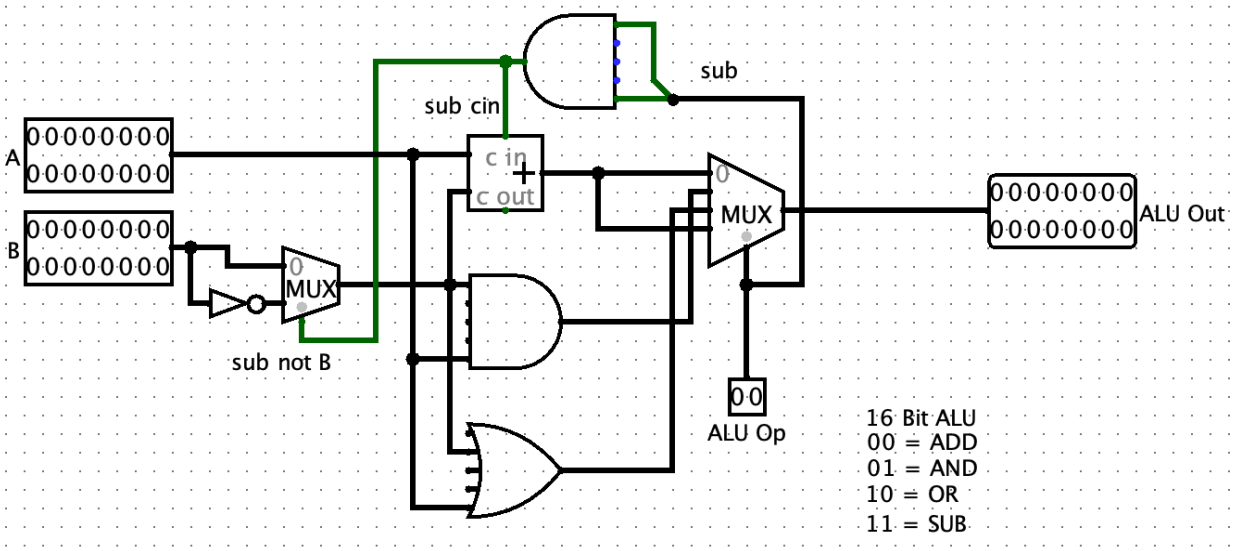

Fig: 16-bit ALU

Basic R-type test results:
https://drive.google.com/drive/folders/1SW5NqPS7dA3fy9OksBHAgjrOzpnmssPw?usp=share_link

Basic SW-LW test results:
https://drive.google.com/drive/folders/1xssT3mKd1GLoXTWJUSS_LyXyQjza3YWW?usp=share_link

- **Branch-if-equal (beq)**

  In order to add the option of beq statements, a new control logic was designed that used the 6 MSB as the op code. An ALU Control Decoder was added to the circuit, which controlled the ALU function to be used, by using the 2 ALU OP Code bits and the 6 Function Code bits ($Instr_{0\text{-}5}$). The ALU was updated such that it could now perform more than 8 operations if required, but only the initial 4 were required for the branch.

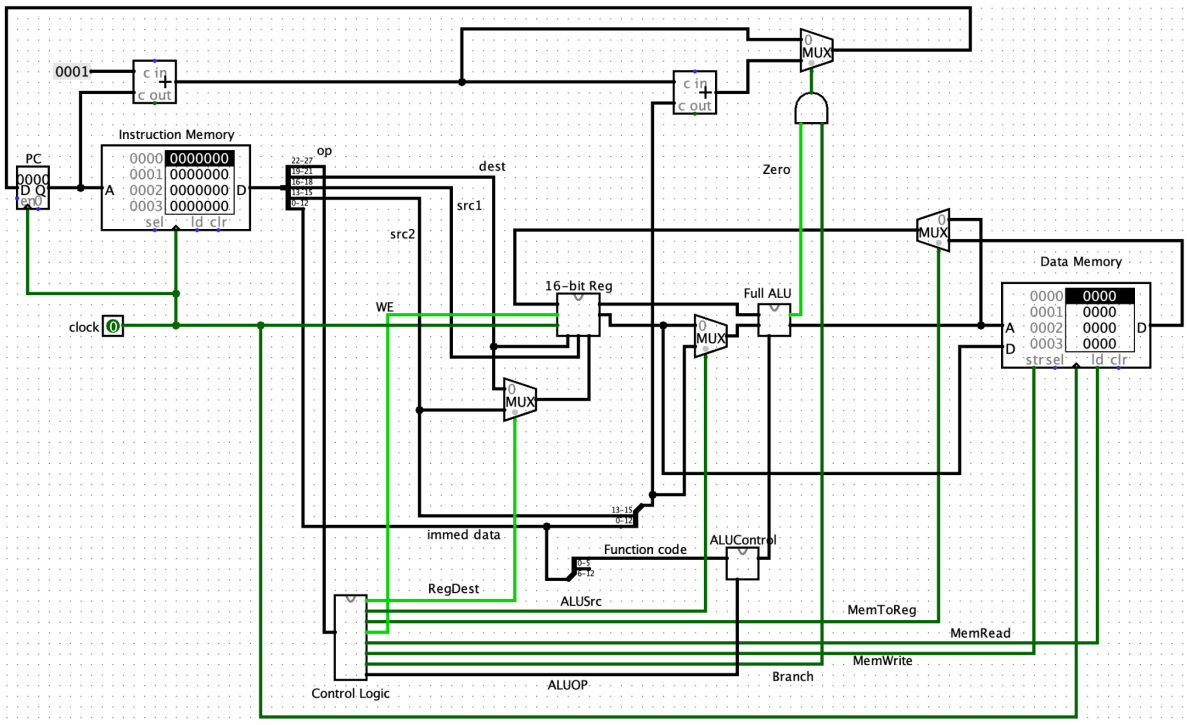

Fig: 16-bit CPU with branch

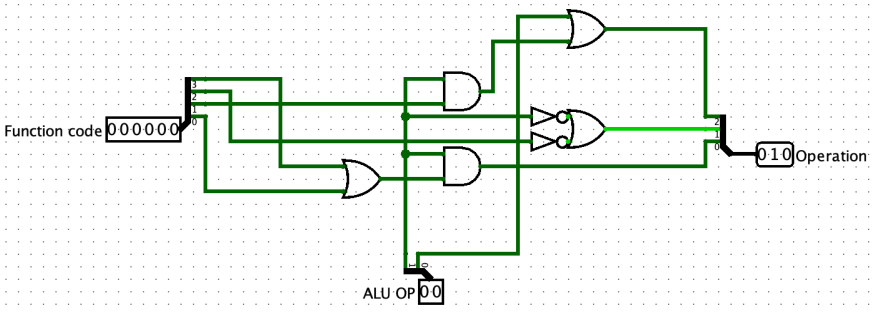

Fig: ALU Control

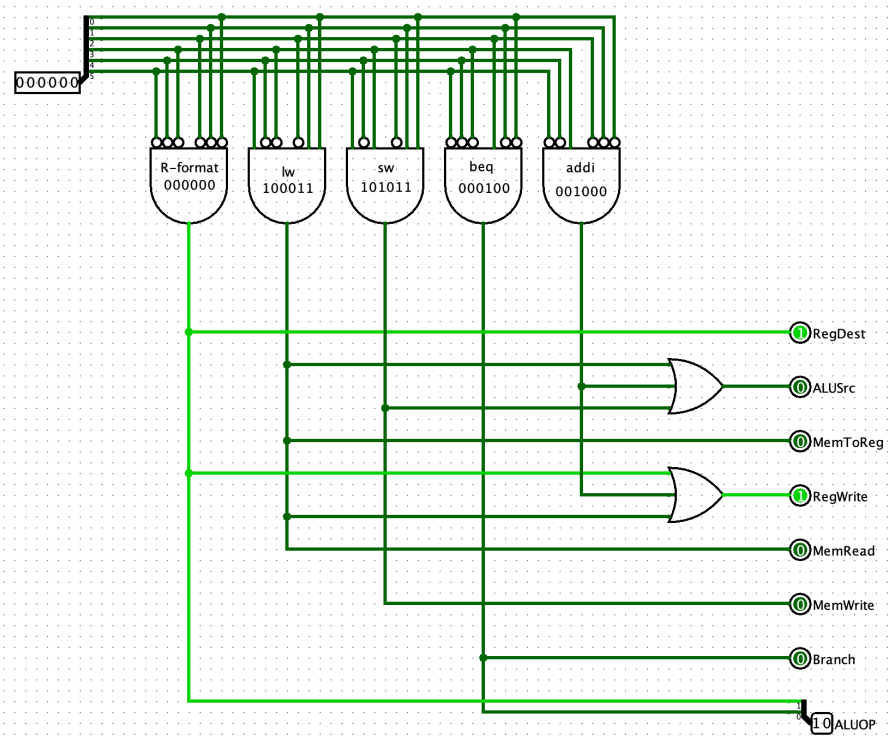

Fig: Updated Control Logic

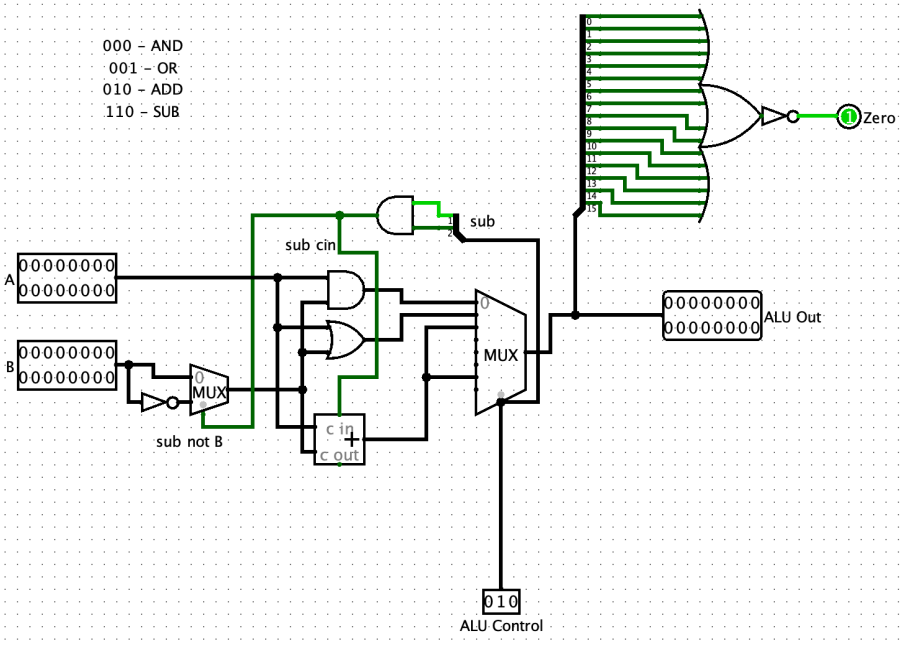

Fig: Full ALU

Branch test results:
https://drive.google.com/drive/folders/1xssT3mKd1GLoXTWJUSS_LyXyQjza3YWW?usp=share_link

- **Jump**

  The control logic was updated to support jump instructions.

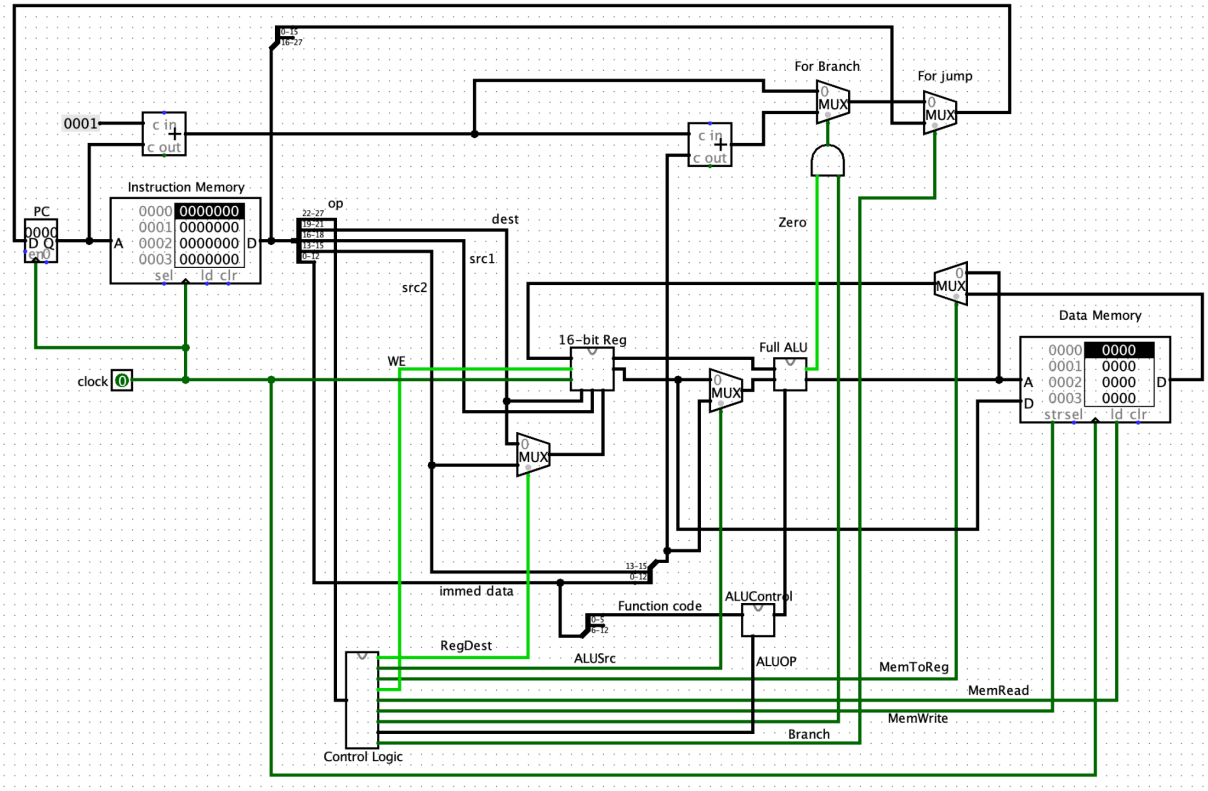

Fig: 16-bit CPU w/ branch, jump

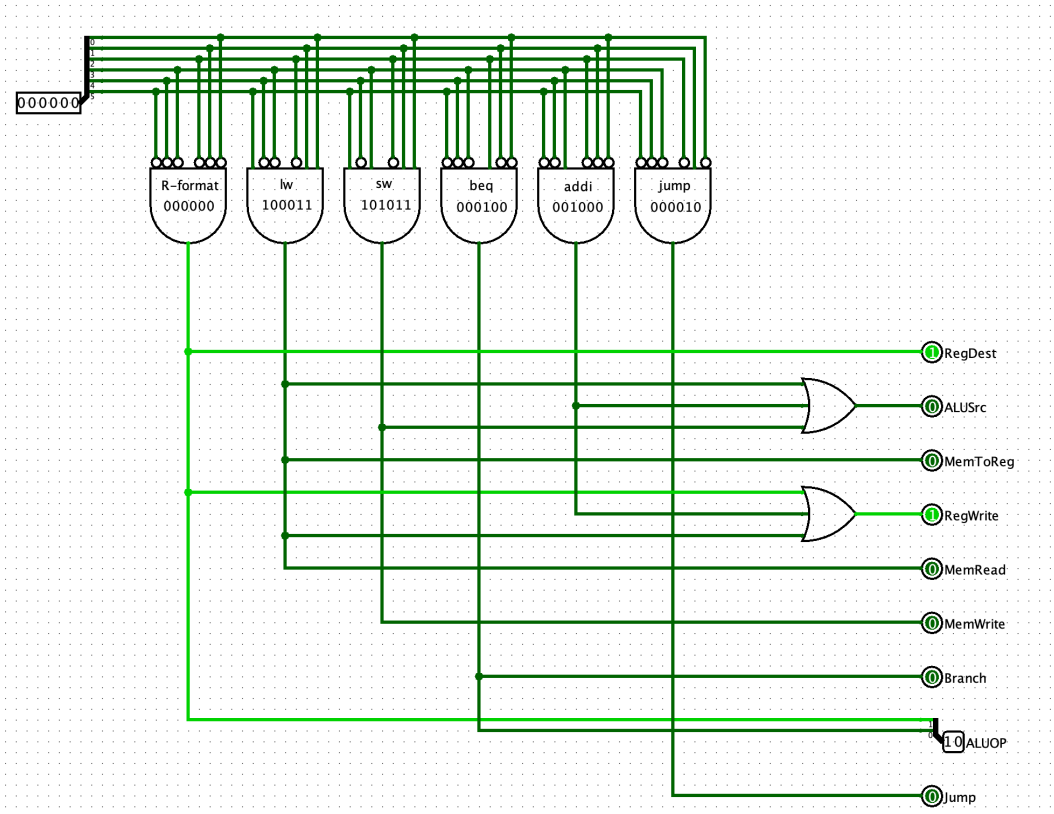

Fig: Control Logic with jump

Jump test results:
https://drive.google.com/drive/folders/12TIDixrDPC0-vXa5jLZnTUFwz6B1OzQL?usp=share_link

- **slt, sgt**

  The Full ALU was updated to add the option of slt/sgt function. The ALU control code for the same was set to be 111 and the function code was set as 101010. The Control Logic OP code stays the same as the one for the registers - 000000 - as this operation deals with only registers.
  In case of sgt, the assembler was edited such that it performs the same function as slt, after switching the src1 and src2.

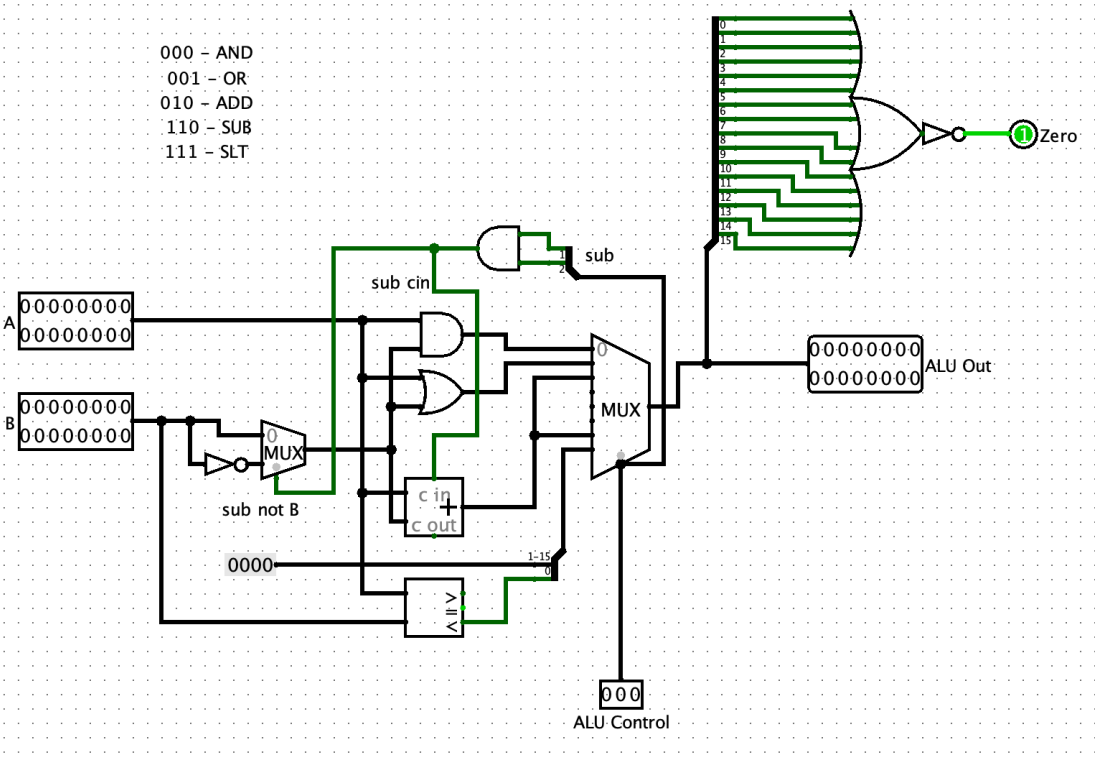

Fig: Full ALU with slt

Slt test results:
https://drive.google.com/drive/folders/1qY4xmPotxaSZAz-Khrw6jCEqHwWEnOdT?usp=share_link

- **labels**
  We reformatted the linker to first clean the input file, which includes removing all comments and converting all labels to differences between instructions

- **blt, bgt**

  In the linker, we added functionality so that if blt/bgt are called it compiles an slt/sgt and beq instruction. Note: Since we didn't have any spare register this relies on changing the value of $1.