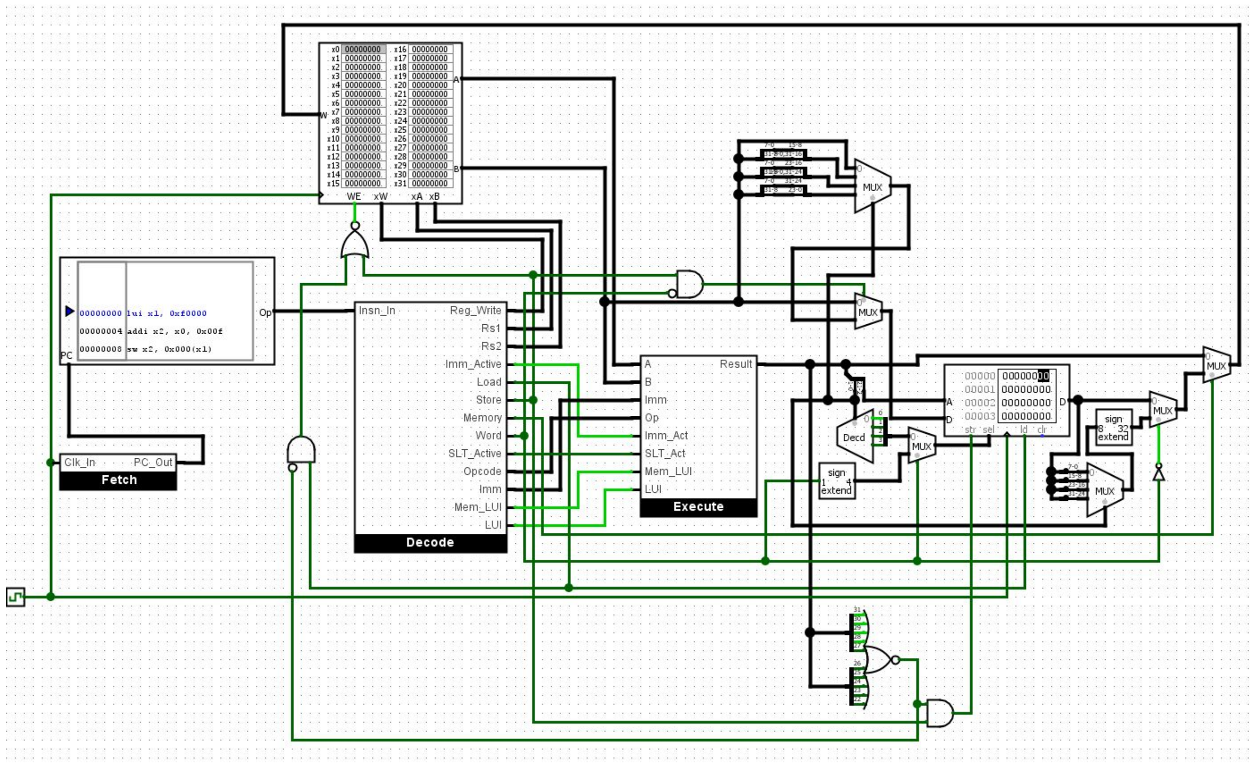
Austin Brown

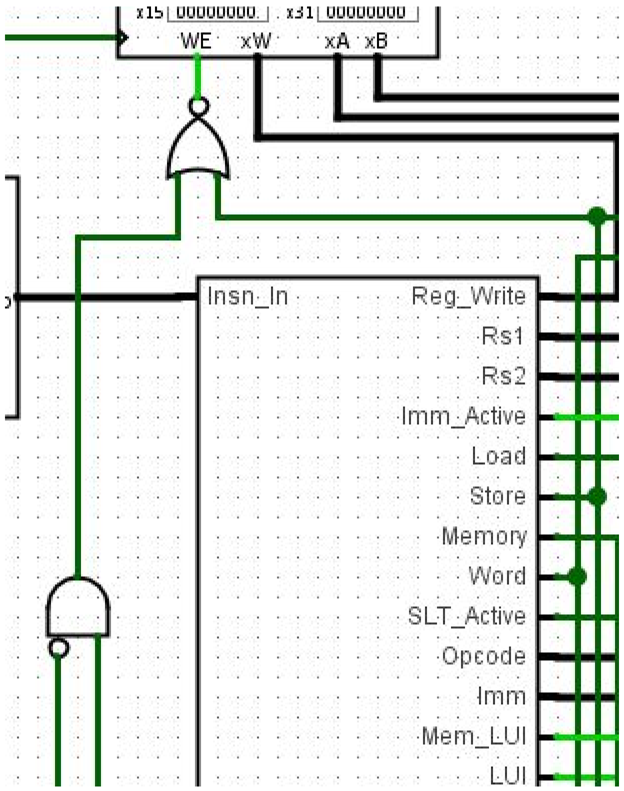
CS 3410 Project 2 Documentation

RISCV32



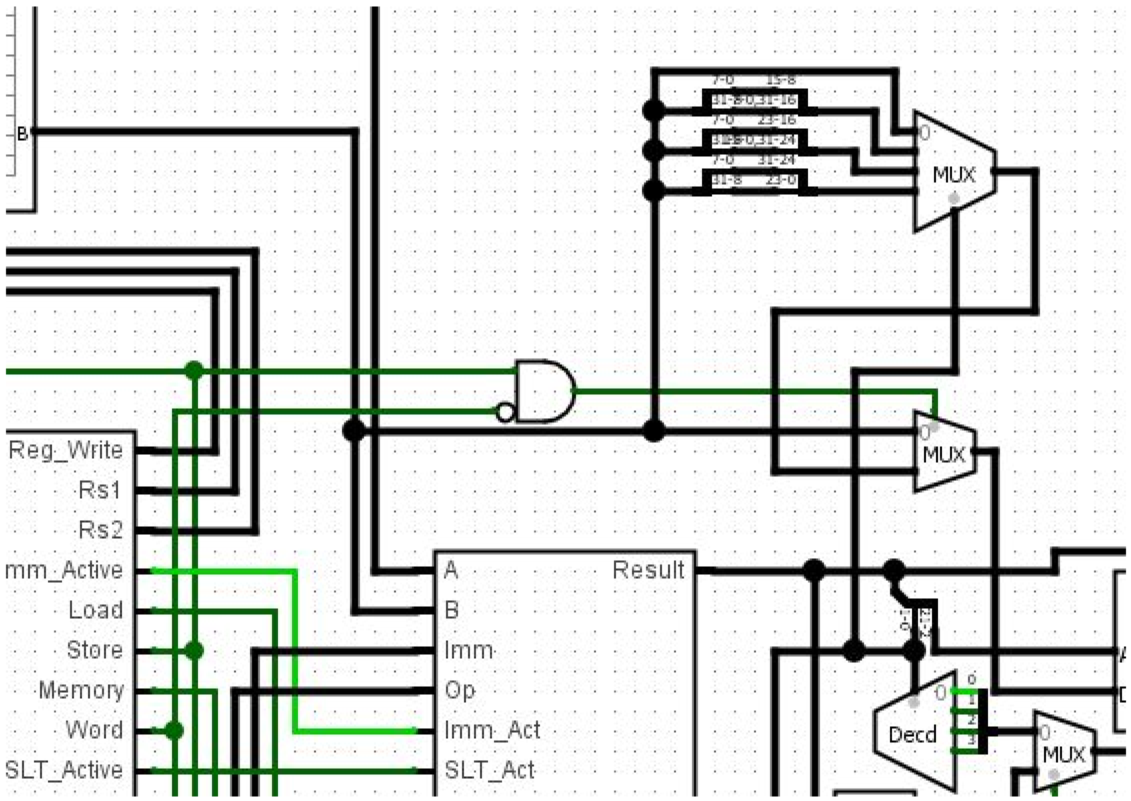
The RISC-V single-cycle processor handles some of the functions that are a part of the actual RISC-V processor. The RISCV32 circuit is broken up into five stages: Fetch, Decode, Execute, Memory, and Write back. The function of each stage and the internals will be discussed later. For now, the certain parts of the RISCV32 circuit that may be confusing will be discussed.

Austin Brown



The few gates here determine when the register file will have write-enable activated. The AND gate on the left will be activated if both a load is occurring and the calculated memory address is outside the limits of the RAM. It will then connect to a NOR gate which has a second input that is active when a load is occurring. Essentially, if either of these two conditions exist, it will prevent any values from being written to the register file.

Austin Brown



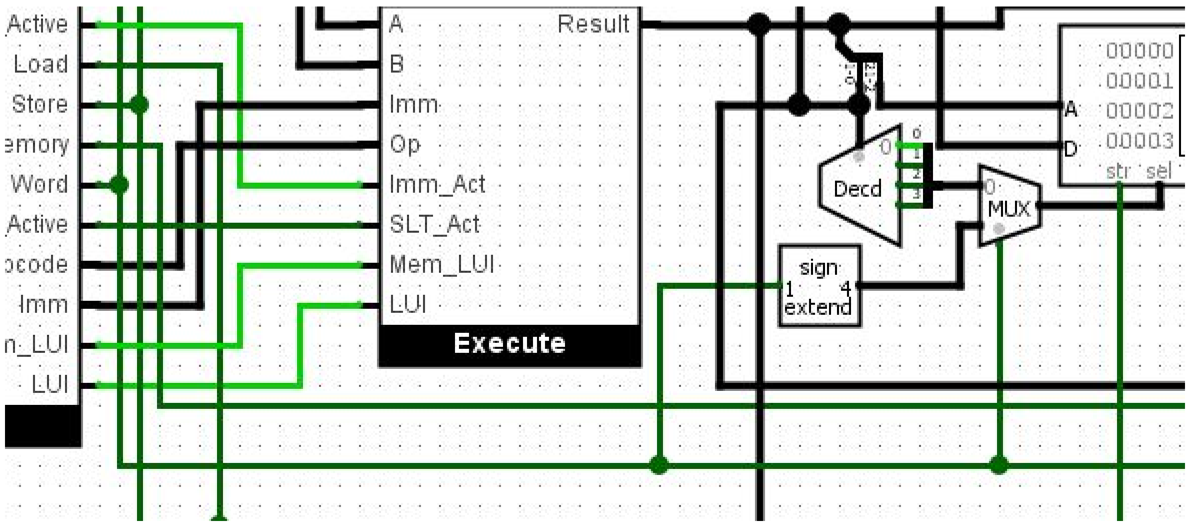
This part of the circuit assists with store bytes. The AND gate determines if there is SB function called. If so, the setup on muxes come into play. The top mux moves the byte around to the necessary storage space in RAM.

Selector Bits 0-1 of calculated address	Output where ** will be written to
00	000000**
01	0000**00
10	00**0000
11	**000000

After the top mux handles the necessary byte shifting, the lower mux handles the necessary change if a SB is called. After that, the data will be sent to the RAM.

Austin Brown

Selector SB active	Data Input
0	Unchanged register B
1	Edited SB input



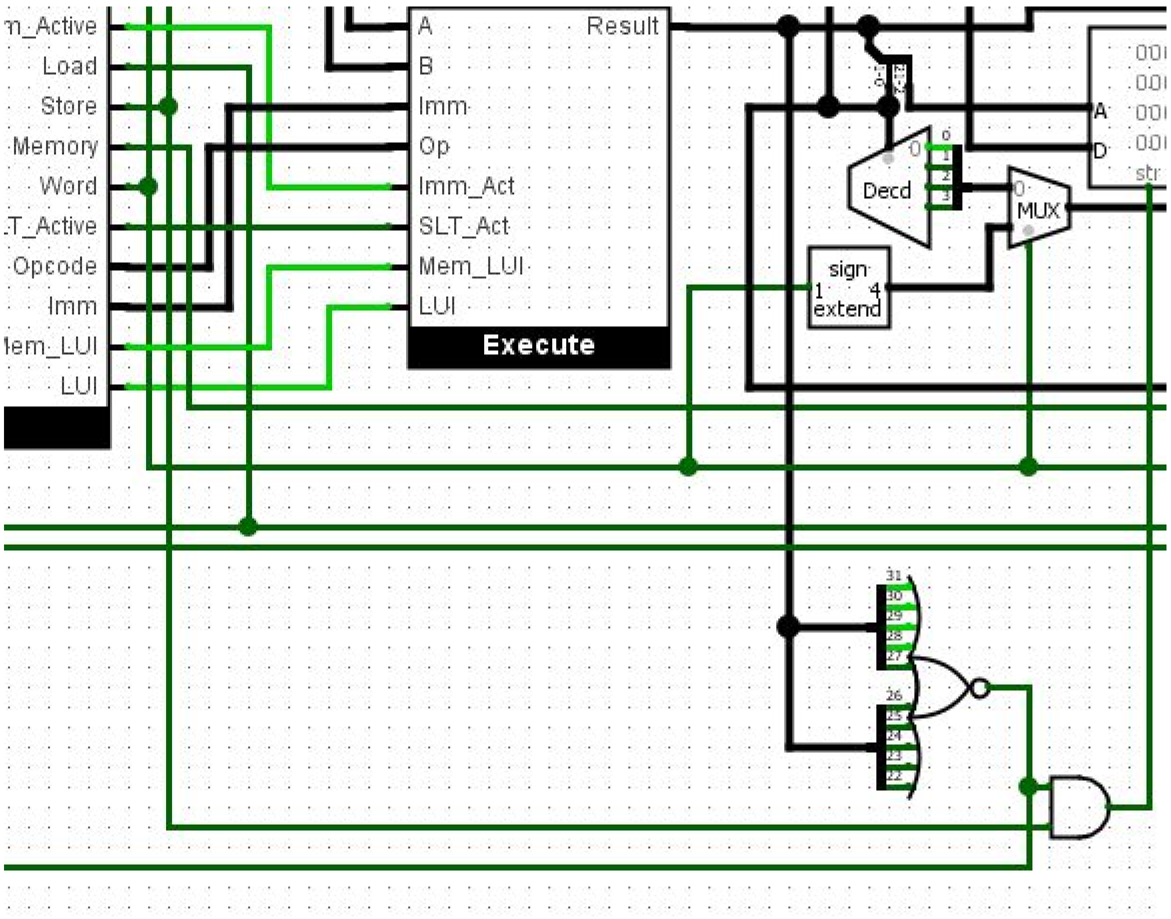
This area of the circuit determines the selected bytes of the RAM. The decoder helps with the byte functions. By determining the 0-1 bits of the calculated address, it will create a 4-bit signal for what byte needs to be selected.

Selector 0-1 bits of calculated address	Output
00	0001
01	0010
10	0100
11	1000

Austin Brown

This input will be connected to the mux that determines if a word or byte function is called. The other input for the mux is necessary for word functions. If a word function is called, all the selector bits will be set to 1 and be passed to the RAM.

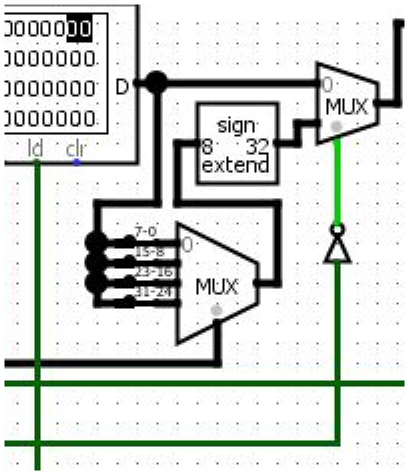
Selector Word	Output to RAM
0	Decoder-determined output
1	1111



The bottom NOR and AND gates help determine when the RAM needs to store a value.

Austin Brown

The NOR gate determines if any of the 22-31 bits of the calculated address are 1, which indicates an out-of-bounds address. The AND gate takes this input and the Store input to determine if a valid store function is called and needs to set the RAM str input to 1.



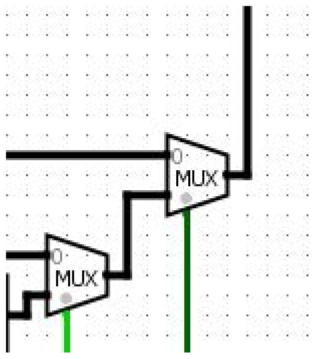
This part of the circuit assists with the necessary output for LB. If a LB function is called, the output byte from RAM must be shifted to lower 8 bits for the register and sign extended. The 4-input mux will take the output byte and move it to the lower 8 bits.

Selector 0-1 bits of calculated address	Output
00	1st byte
01	2nd byte
10	3rd byte
11	4th byte

Once the output goes through the signed bit extender, it will go to the 2-input mux that has the untouched input from RAM. If a byte function is called, it will output the adjusted output. Otherwise, it will output the unaffected value from RAM

Austin Brown

Selector Word	Output
0	Adjusted value
1	Unadjusted value

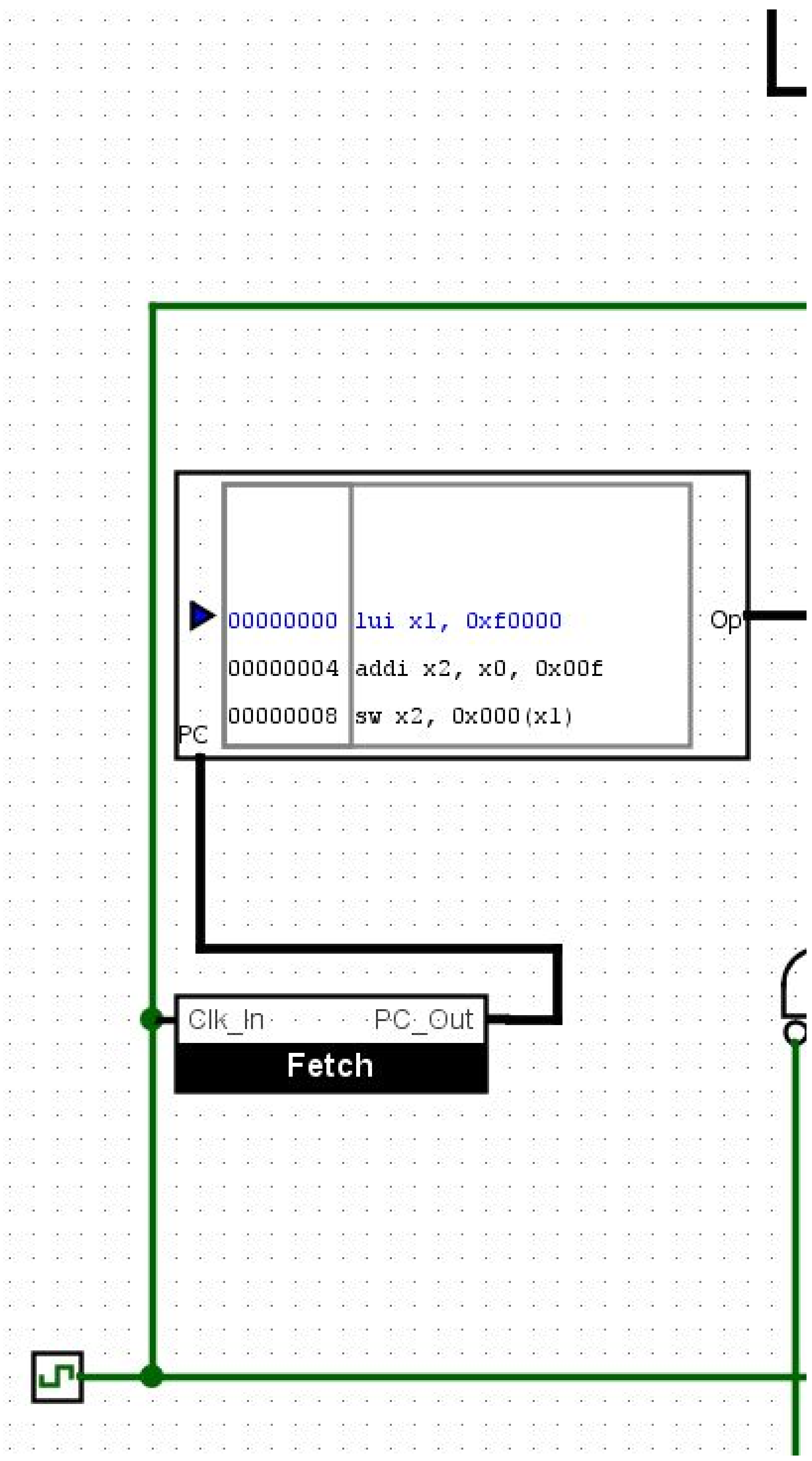


The far right mux will display either the value from memory or the ALU.

Selector Memory	Output
0	ALU
1	Memory

Fetch

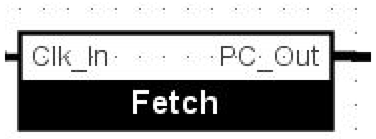
Austin Brown

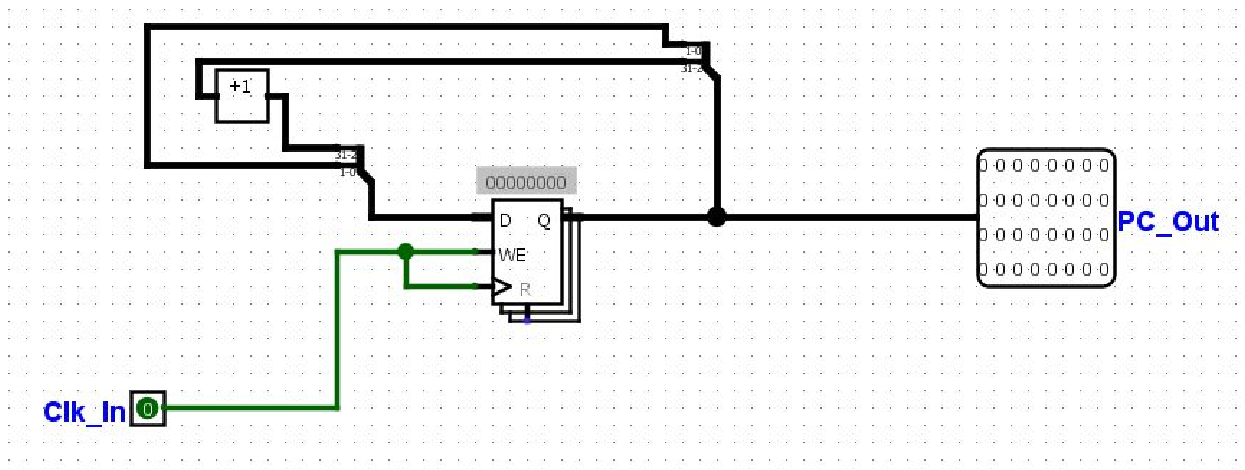

```
00000000 lui x1, 0xf0000
00000004 addi x2, x0, 0x00f
00000008 sw x2, 0x000(x1)
```

Fetch handles the retrieval of instructions from program memory.

Fetch sub-circuit

Austin Brown





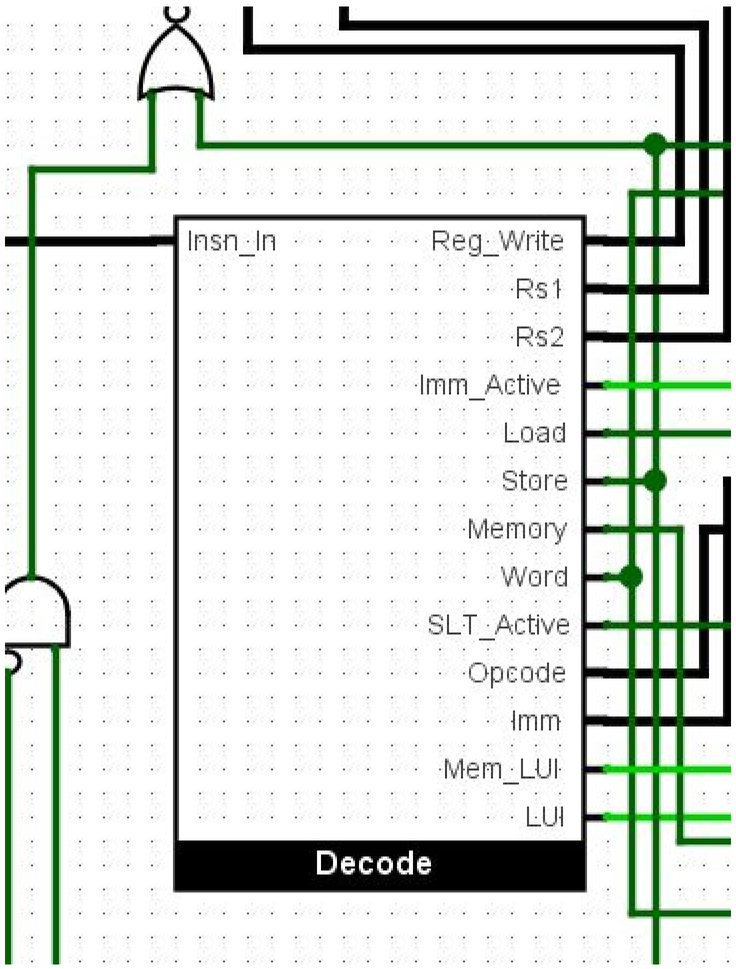
The Fetch sub-circuit handles the PC value. The incrementer is used on bit-2 so that it consistently increases the PC by 4. The register will store the PC value and will change its value on the rising-edge of a clock cycle.

Decode

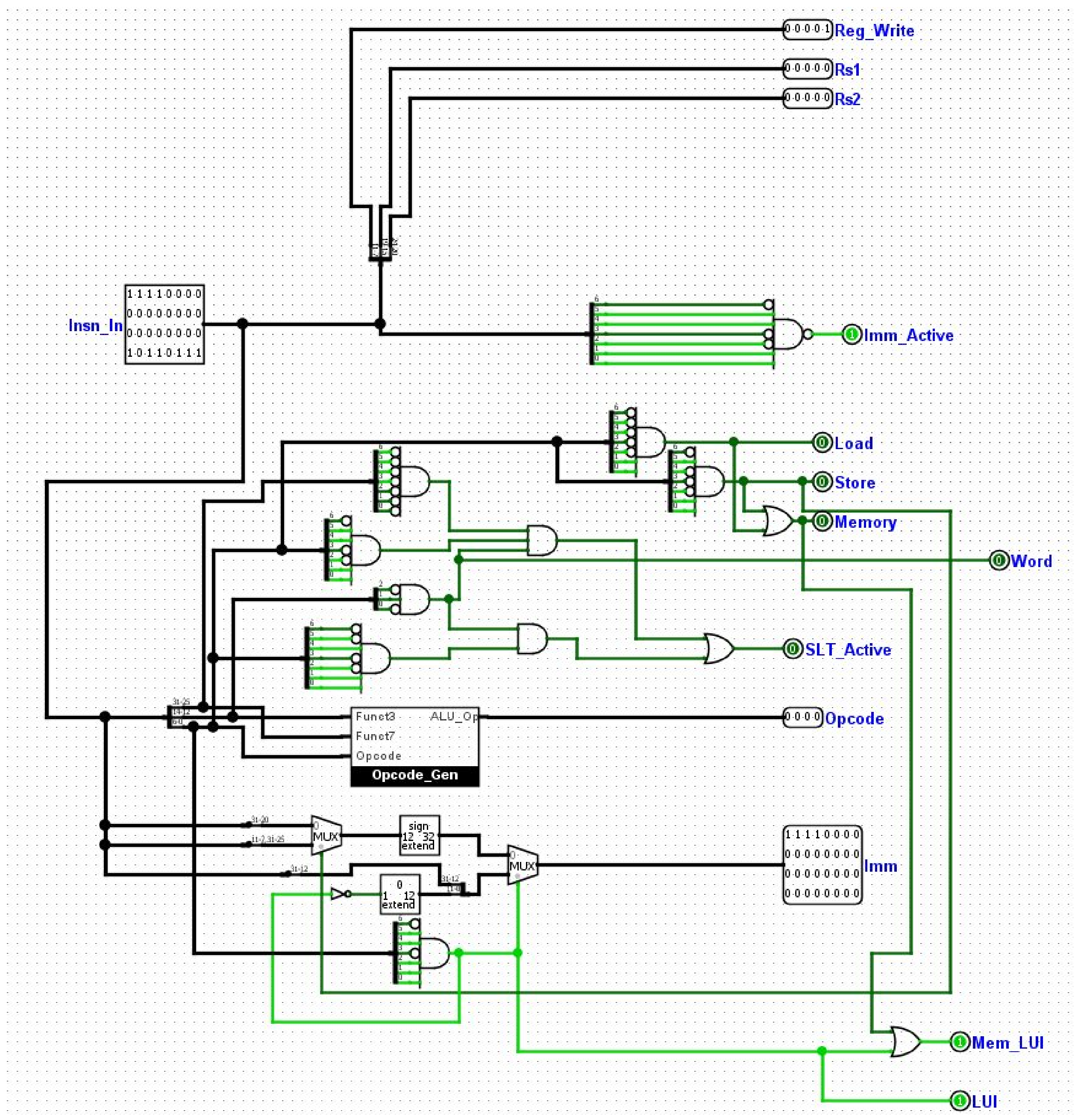
Decode handles the deconstruction of the instruction to allow the rest of the circuit to handle its operations.

Decode sub-circuit

Austin Brown



Austin Brown



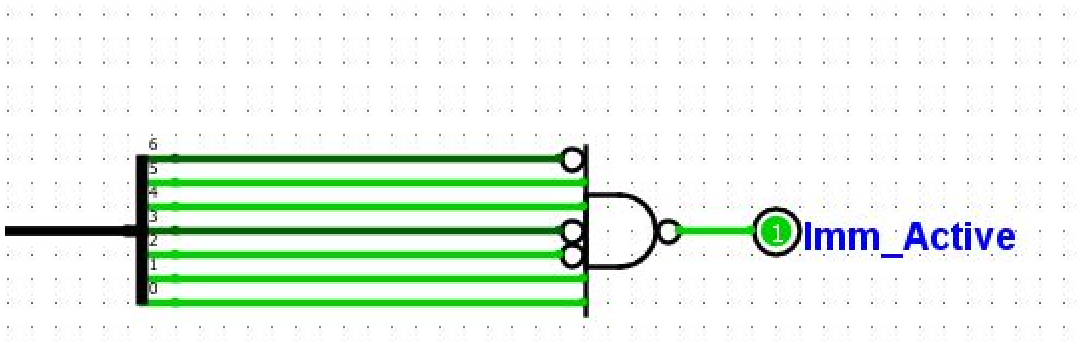
The Decode sub-circuit handles the necessary breakdown of the instruction input.

The following outputs are generated by this sub-circuit:

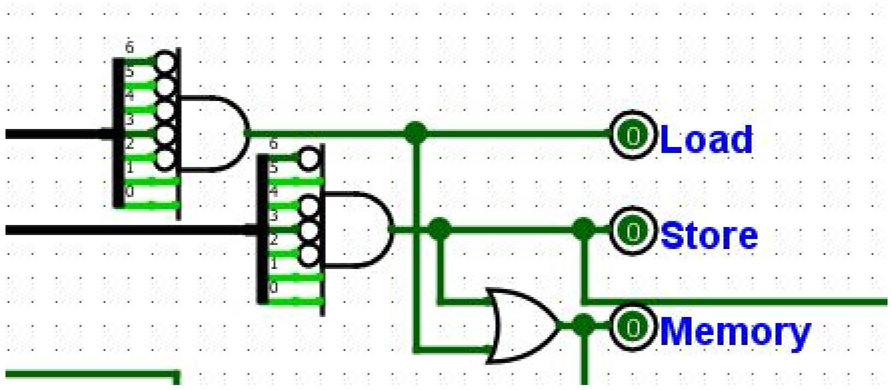
- Reg_Write: Address of register file that needs to be written to
- Rs1: Address of register A of register file
- Rs2: Address of register B of register file

Austin Brown

- Imm_Active: Determines if an immediate needs to be used
- Load: Determines if a load function is called
- Store: Determines if a store function is called
- Memory: Determines if a function that needs the RAM is called
- Word: Determines is a memory function is using a word or byte
- SLT_Active: Determines if a SLT or SLTI is called
- Opcode: Determines the necessary opcode for the ALU
- Imm: Immediate
- Mem_LUI: Determines if a memory function or LUI is called
- LUI: Determines if LUI is called

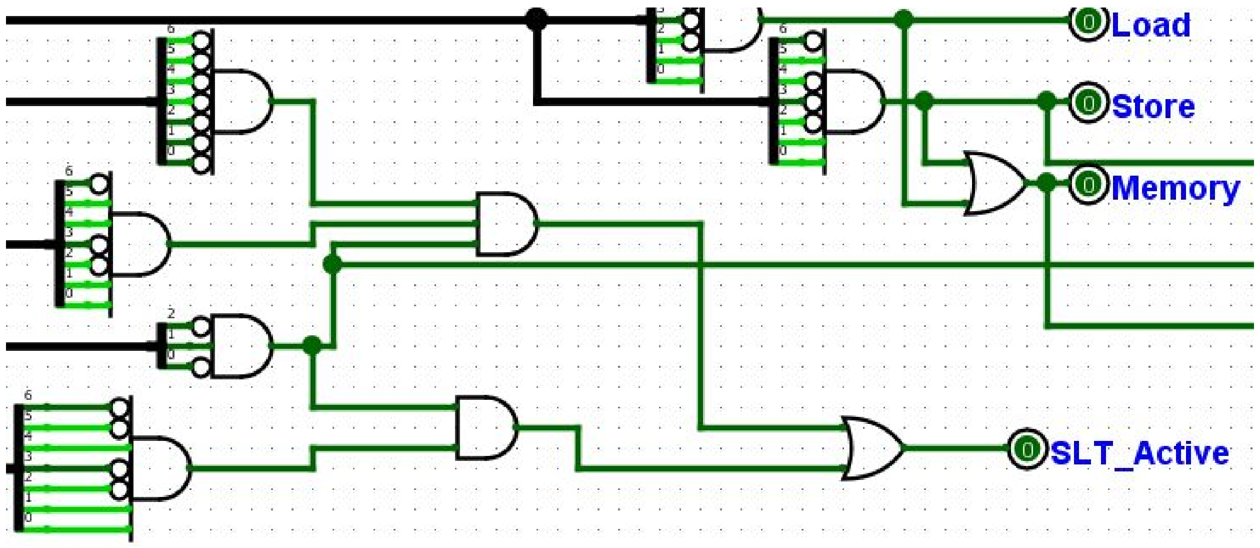


The AND gate checks if an R function is used or not. If an R function is not called, an immediate must be used and Imm_Active is on.

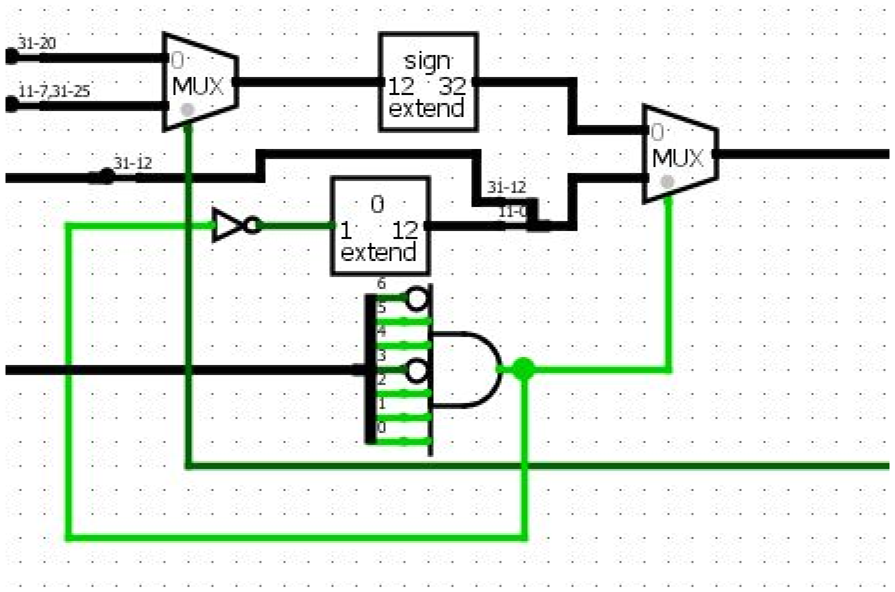


Austin Brown

This part of the circuit checks the opcode to see if an LB/LW or SB/SW function is called. If one of them is called, then memory must be used.



This part of the circuit checks if an SLT or SLTI function is called. It will check that the Funct3, Funct7(for SLT only), and opcode match the values for SLT or SLTI.



This area of the circuit helps generate the immediate. The first mux determines if a I or S function is called.

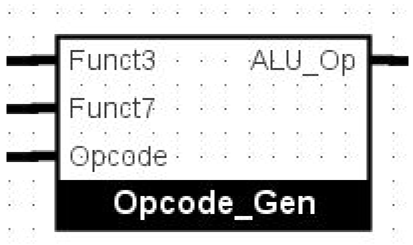
Austin Brown

Control Store	Function type
0	I
1	S

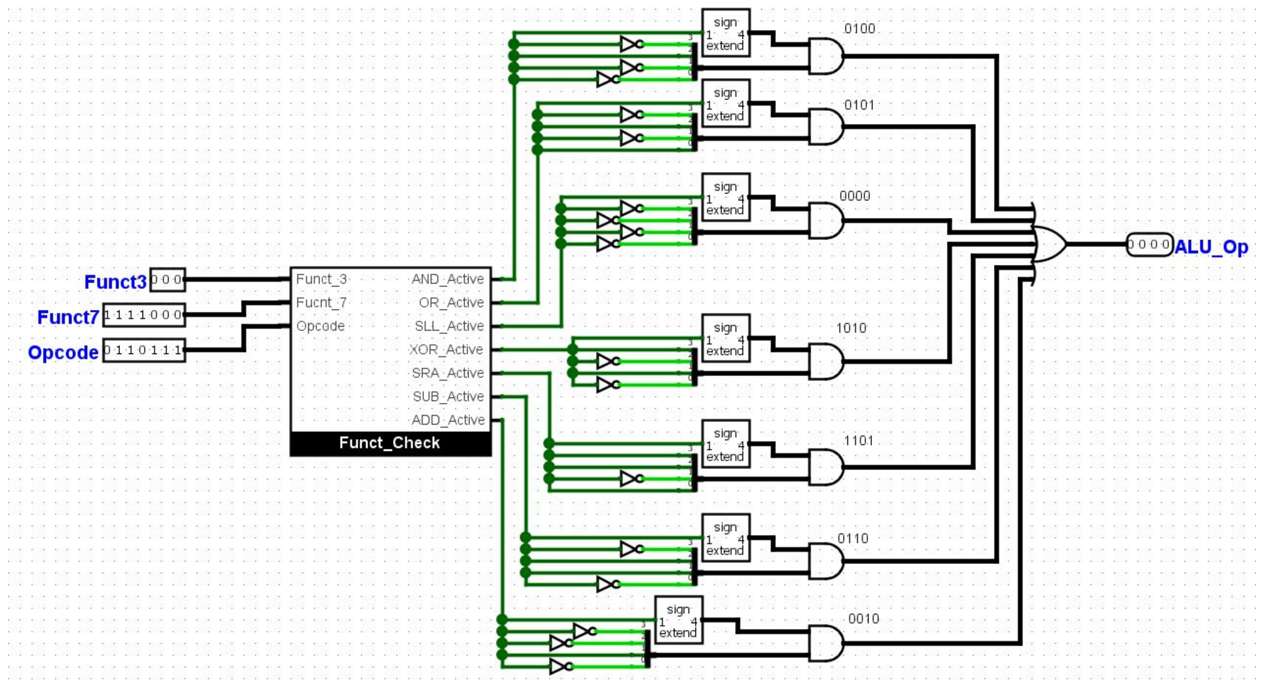
The output of the first mux will be sign-extended to 32 bits and will be passed onto the second mux. The other input of the second mux is the immediate of an LIU instruction. The second mux will determine if an I/S or LIU function has been called

Control LIU called	Immediate used
0	I/S
1	LIU

Opcode_Gen

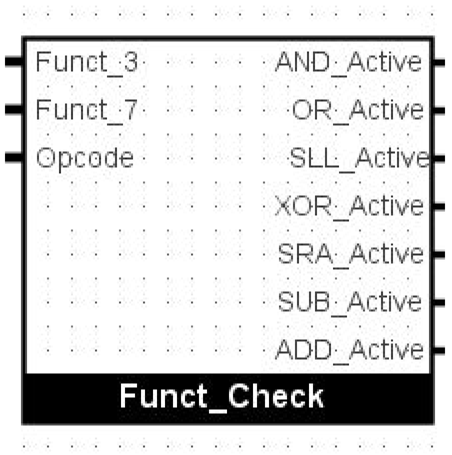


Austin Brown

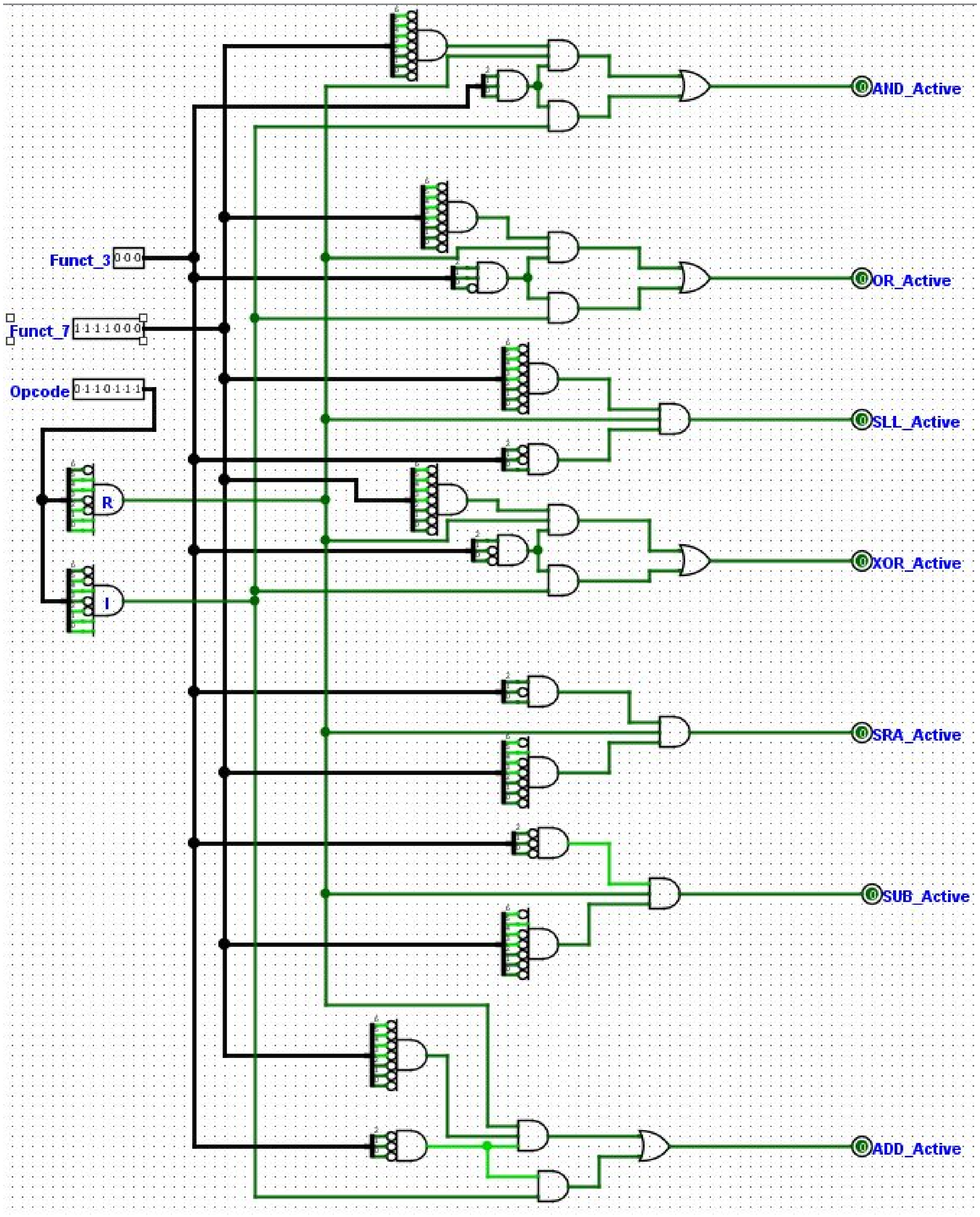


Opcode_Gen creates the ALU opcode from Funct3, Funct7, and Opcode. After going through Funct_Check to see which function is active, it will then be sent to two different parts. One part creates the relevant opcode for ALU and the other is bit extended to allow the ALU opcode to pass through the AND gate.

Funct_Check



Austin Brown

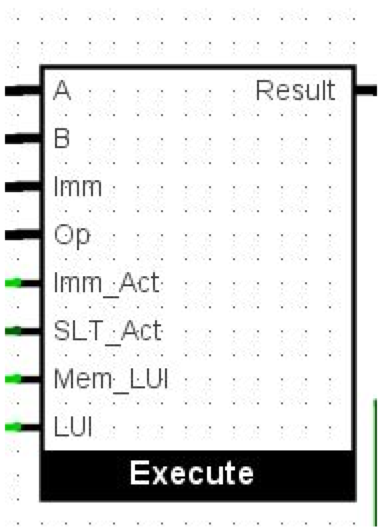


Funct_Check checks if an ALU function is called. Funct_3, Funct_7, and Opcode are used to check if a relevant function is called. The R and I function opcode is calculated in one place right under the inputs since it would be redundant to include them for each function.

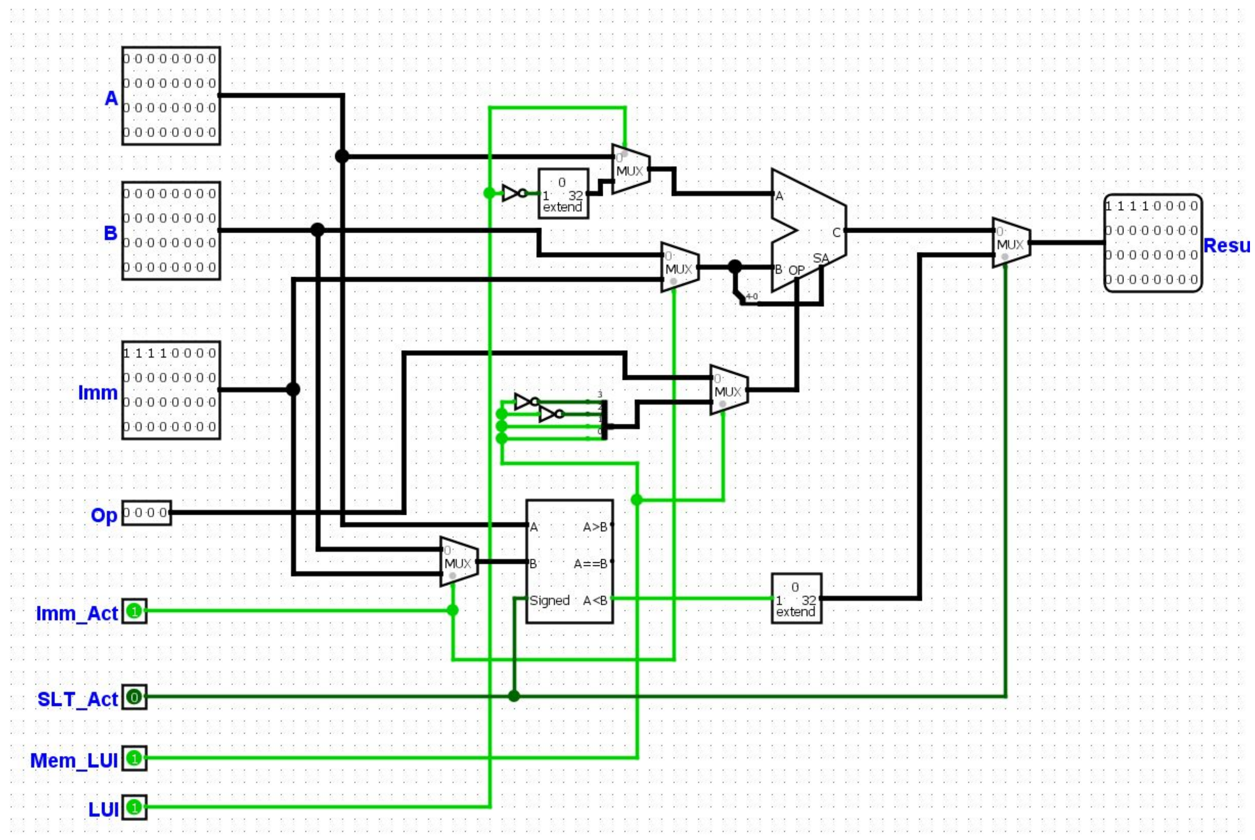
Execute

The Execute stage handles all the necessary calculations for functions and can prepare the memory address for the RAM.

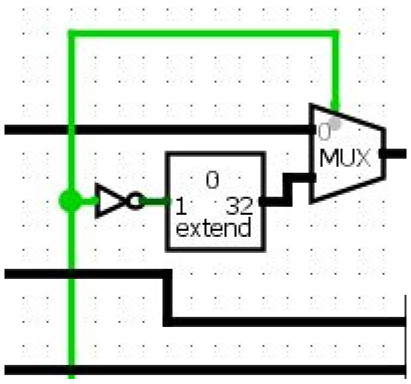
Execute sub-circuit



Austin Brown



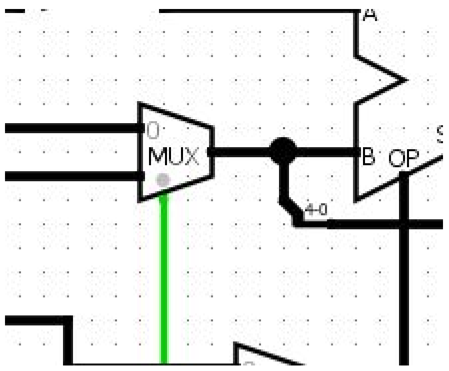
The Execute sub-circuit handles the calculations of the SLT(I) and ALU functions.



This part of the circuit helps ensure that the A input will be set to 0 if a LUI function is called.

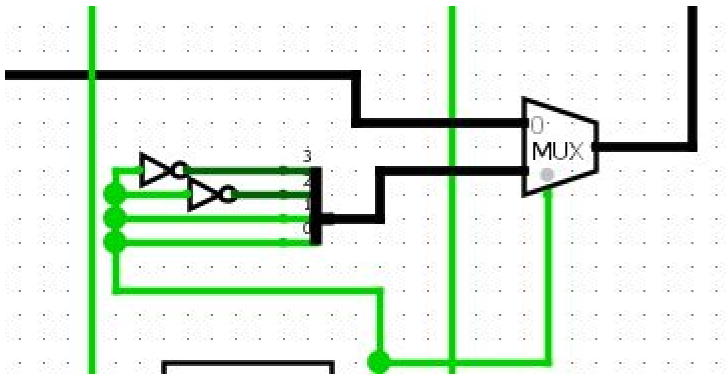
Austin Brown

Selector LUI	Output
0	A
1	0x00000000



The mux before the ALU will ensure that the called function gets the immediate or B as needed.

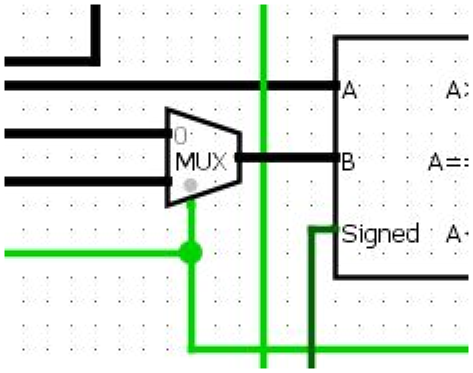
Selector Imm_Act	Output
0	B
1	Imm



Austin Brown

This part of the circuit ensures that the ALU uses ADD if a memory or LUI function is called.

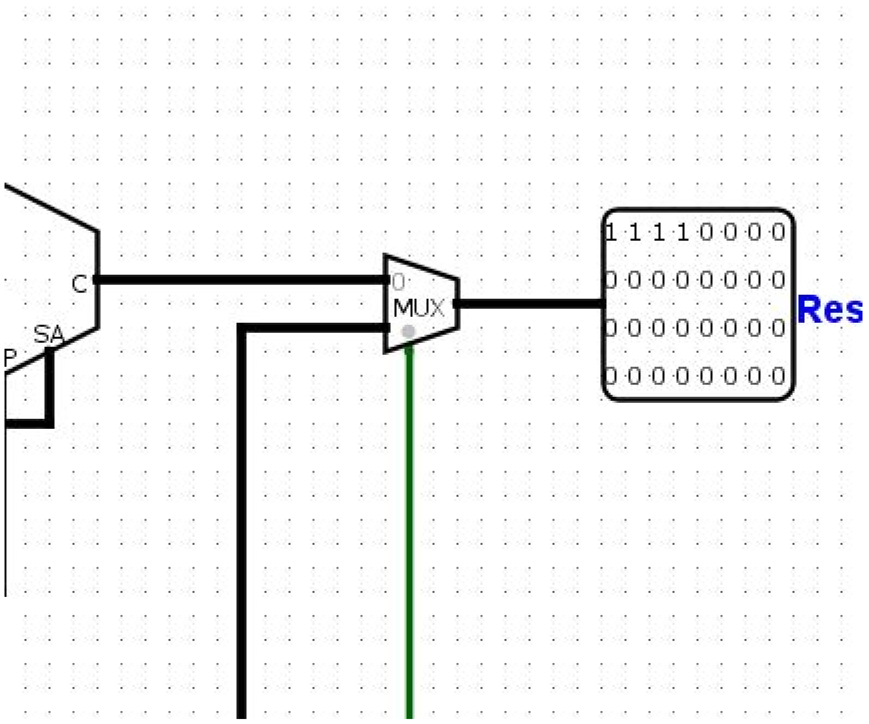
Selector Mem_LUI	Output
0	Normal Opcode
1	0011 ADD



The mux here chooses between B or immediate for the SLT and SLT(I) functions.

Selector Imm_Act	Output
0	B
1	Imm

Austin Brown

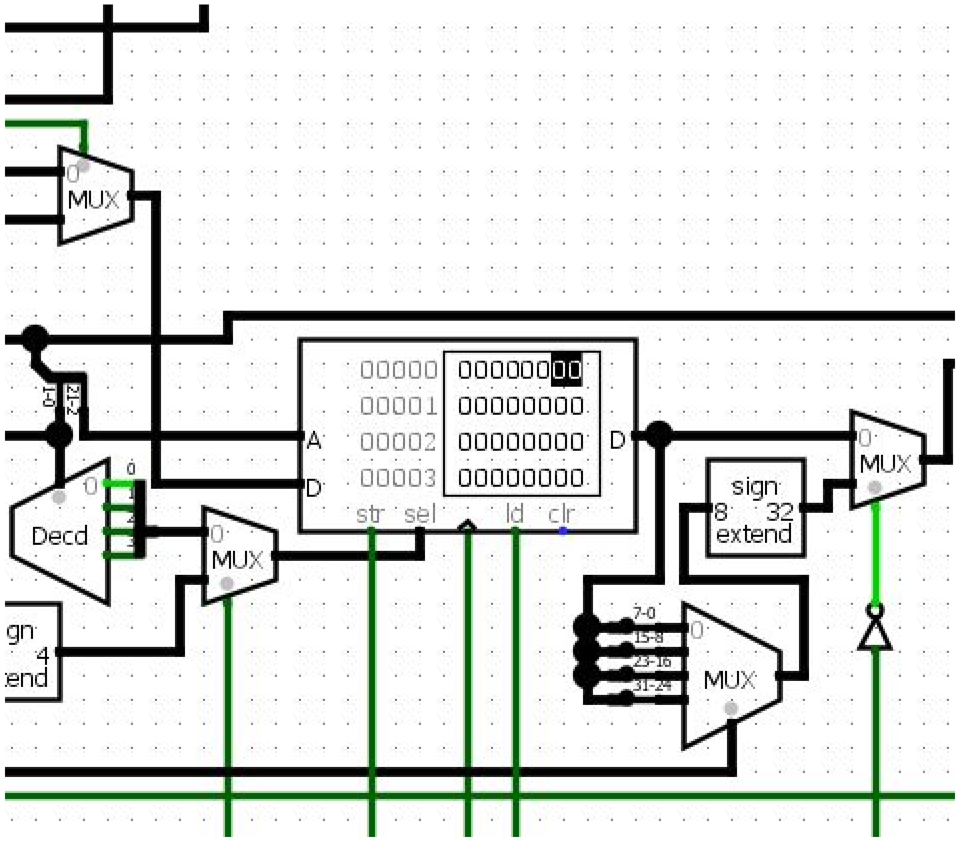


The mux shown here chooses between the result of the ALU and the result of the SLT(I).

Selector SLT_Act	Output
0	ALU
1	SLT(I)

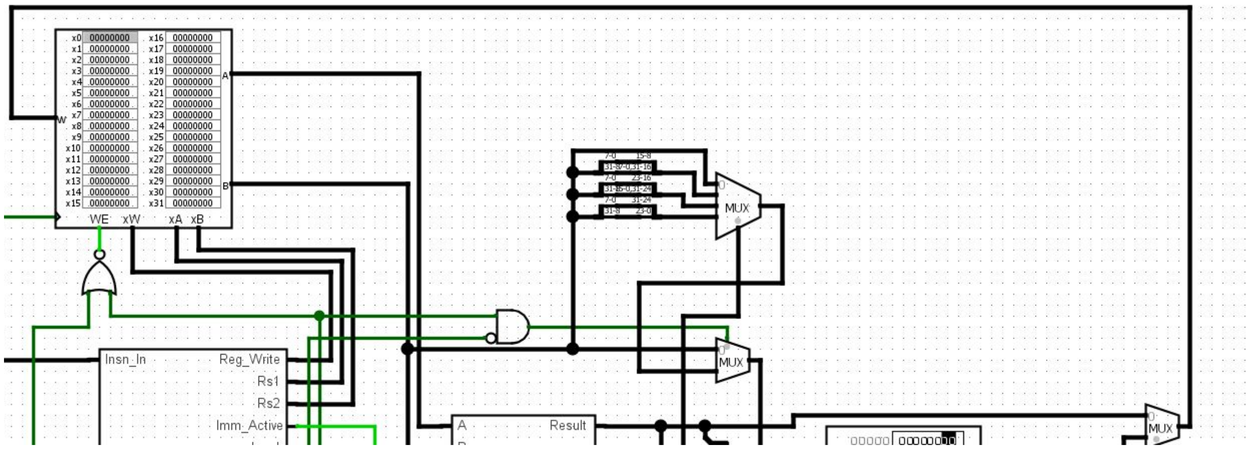
Memory

Austin Brown



Memory handles the I and U functions that need to access RAM.

Write Back



Write Back handles the writing of the desired value to the desired register.

Austin Brown

Testing

Testing was designed to cover the different edge cases and some random cases for each function. Since we were provided a working ALU, edge cases on it weren't necessary. The primary goal of testing with the ALU was to make sure it worked with the rest of the circuit. The edge cases that needed to be checked involved the parts that needed to be specially implemented to work, the SLT(I) and RAM. Once those edge cases were checked, each function would be tested once to ensure proper operability within the circuit. When it came to the random cases, it was determined that three cases of each function would satisfy testing requirements and ensure that the circuit operated as expected.