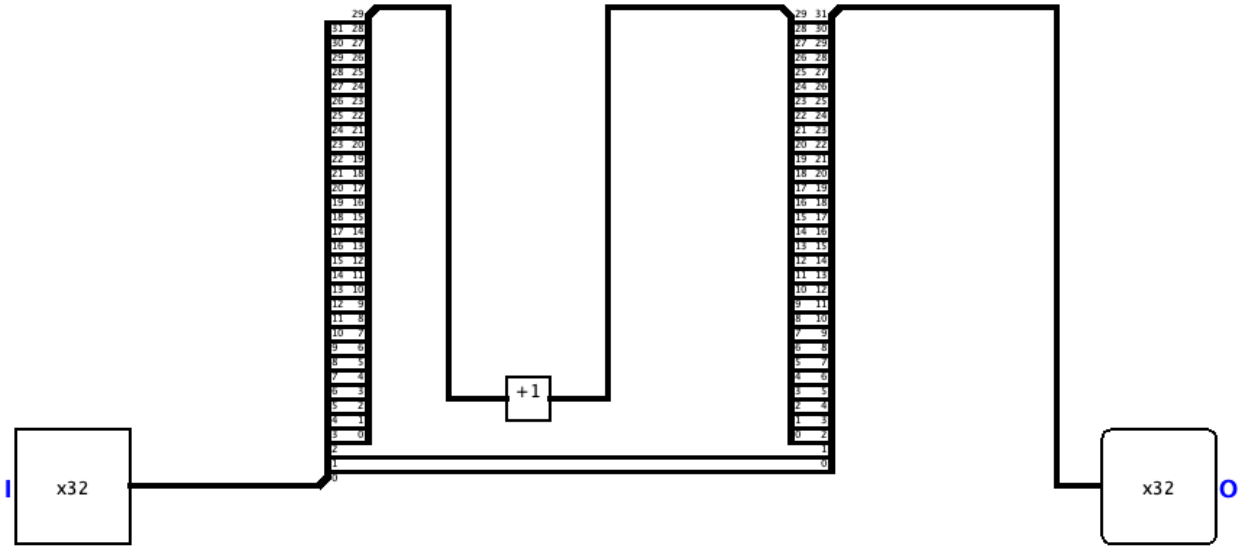
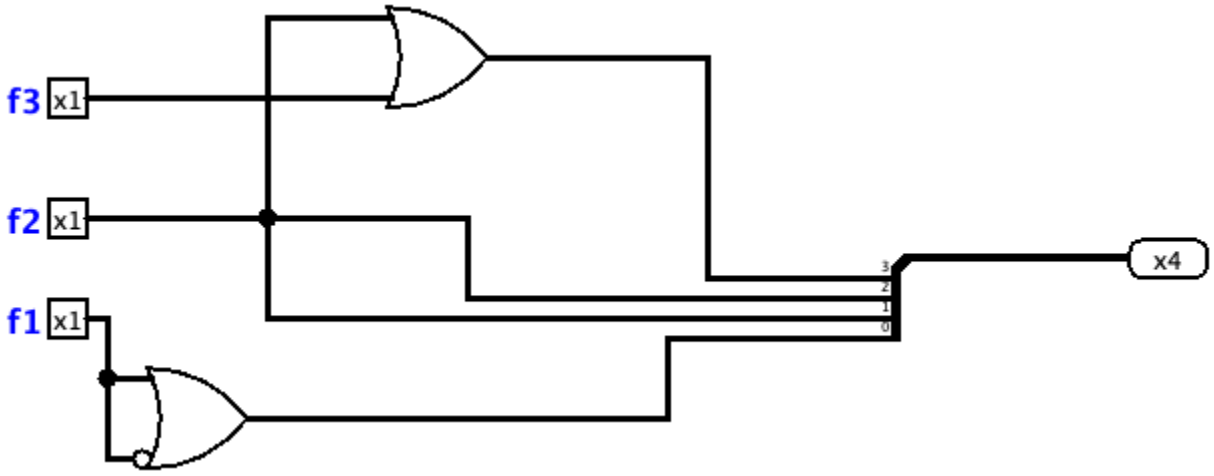
Michael Torku
CS3410
Project 2

DESIGN DOC

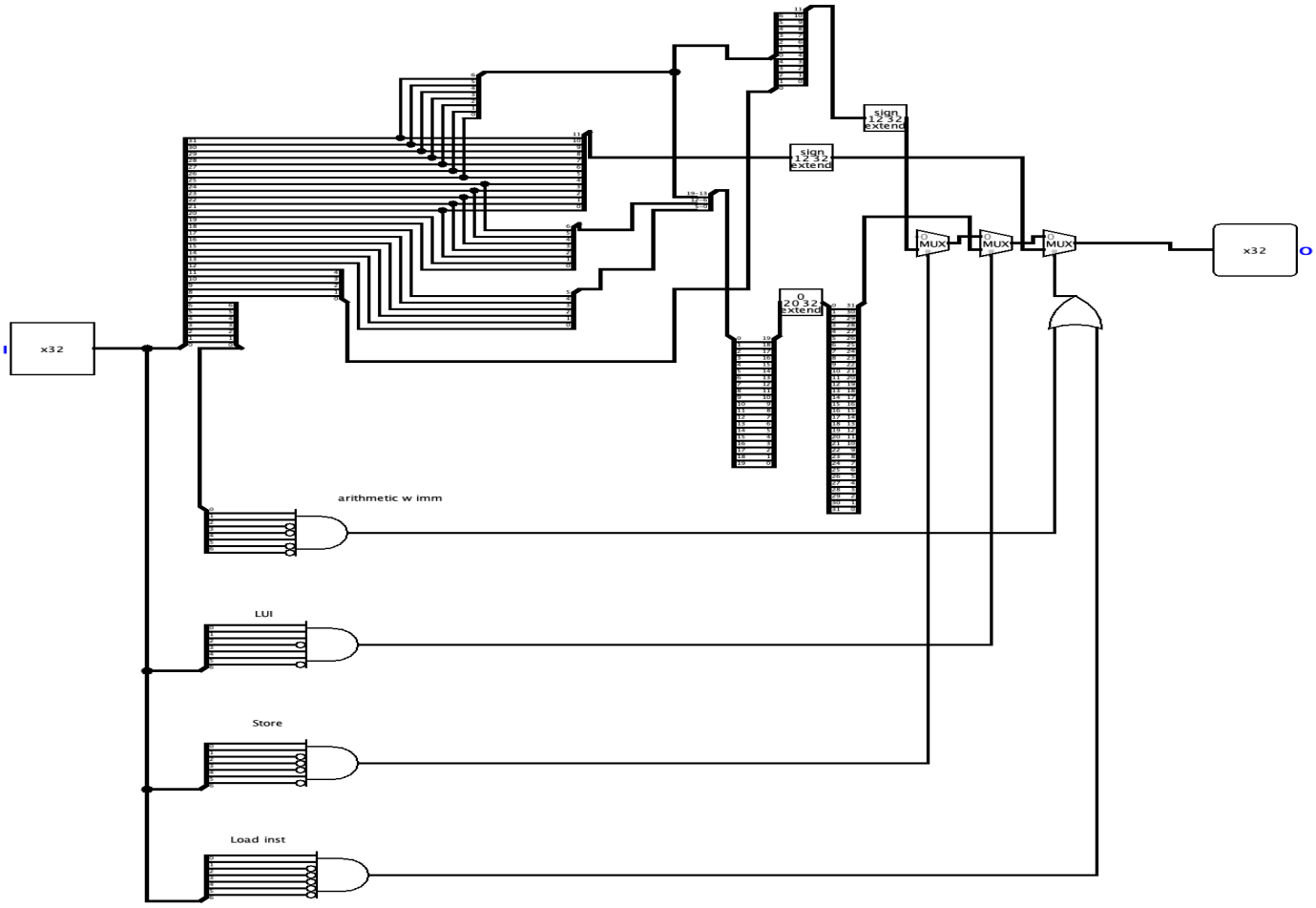
ADD4: This subcircuit handles incrementing the program counter value by 4. A splitter is used to abstract the 30 most significant bits and add 1 to the value (using an incrementer) another splitter is used to join the result of said addition with the two least significant bits of the original input value to get a 32 bits value which is 4 bits greater than the original input value.



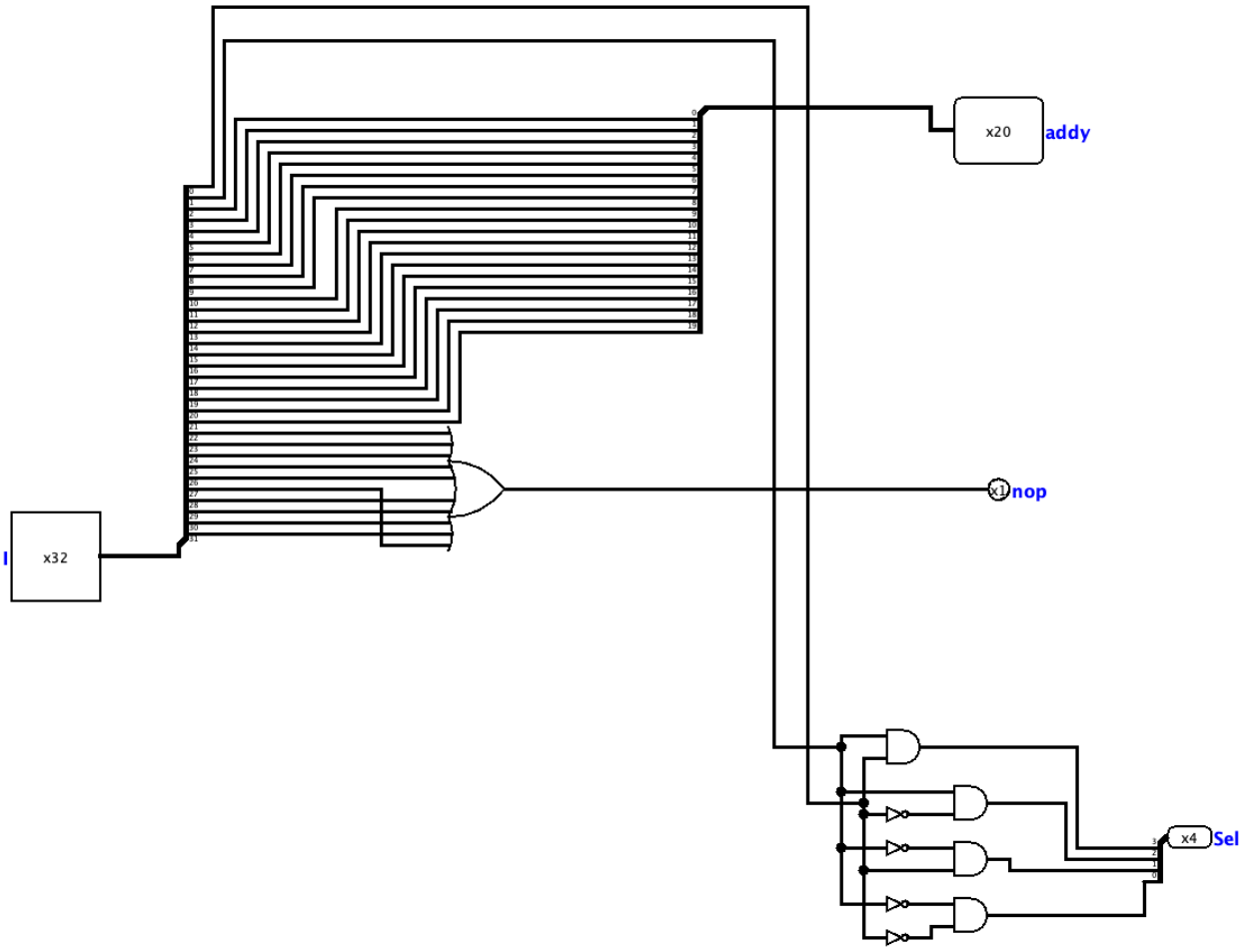
SelComb: This subcircuit is used to determine the byte(s) to be selected for the memory component - it is only used if a full word is being loaded or stored.



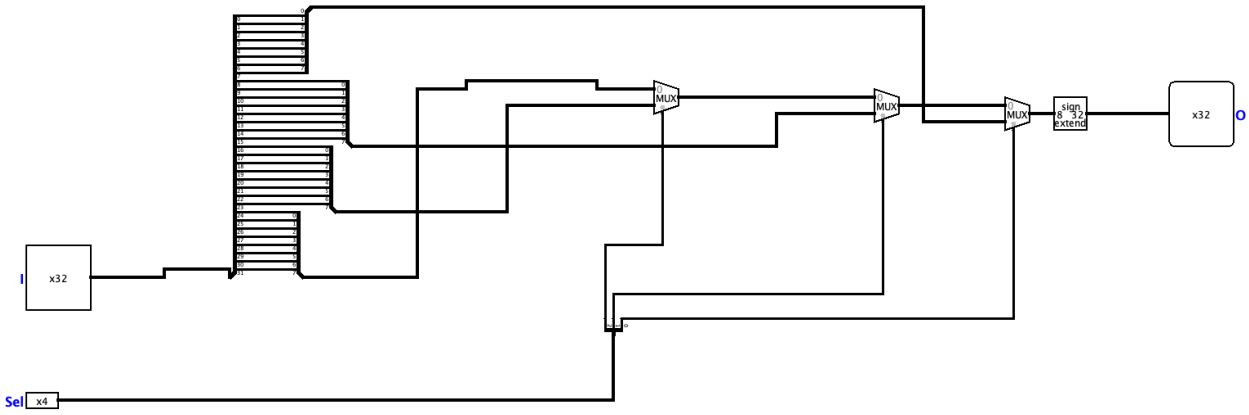
IMMGEN: This subcircuit is used to extract the value of an immediate for any operation in which an immediate is used (addi, lui, etc). The four AND gates at the bottom are used to determine the type of immediate to be taken (they are used as the selector bits for the mix) and are an assertion of the kind of operation being executed (e.g. 0010011 implies that the immediate of an arithmetic operation is being requested - this is found in the first 12 bits) so the right immediate is passed through the muxes and sent to the output O.



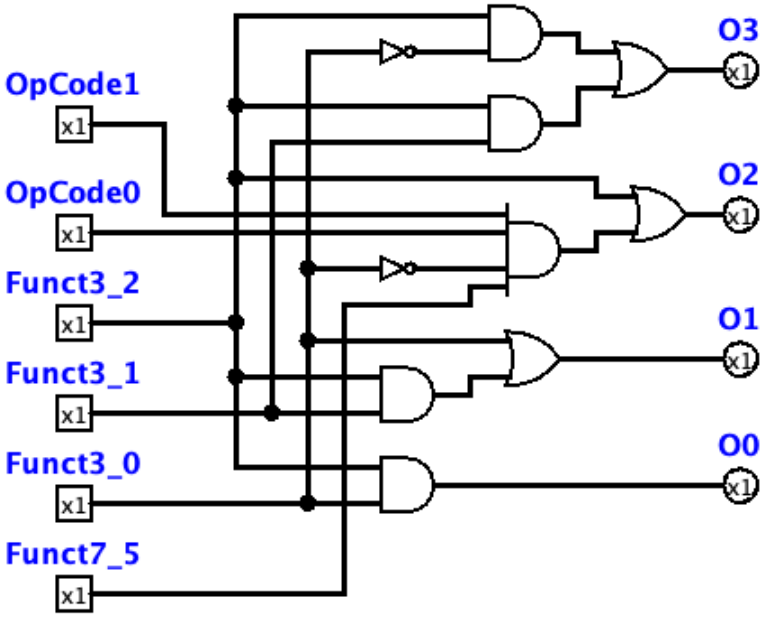
NOPCHECK: This subcircuit is used to determine the select bits of memory by when a byte is being loaded or stored. It uses the last two bits to determine the location of the byte being stored or loaded by using a small decoder at the bottom of the circuit which outputs a 4 bit value with a 1 at the location of the byte being loaded or stored. This value is provided as Sel to the memory component if the operation being executed is a store.The last 10 bits are used to handle the case of a nop - if there is a non-zero value in the 10 most significant bits of the input, then that implies that no operation taking place (a nop) and so send an output 1 to indicate that there is a nop (other components ensure there is no loading or storing of data from or to the memory component (respectively) in this case).



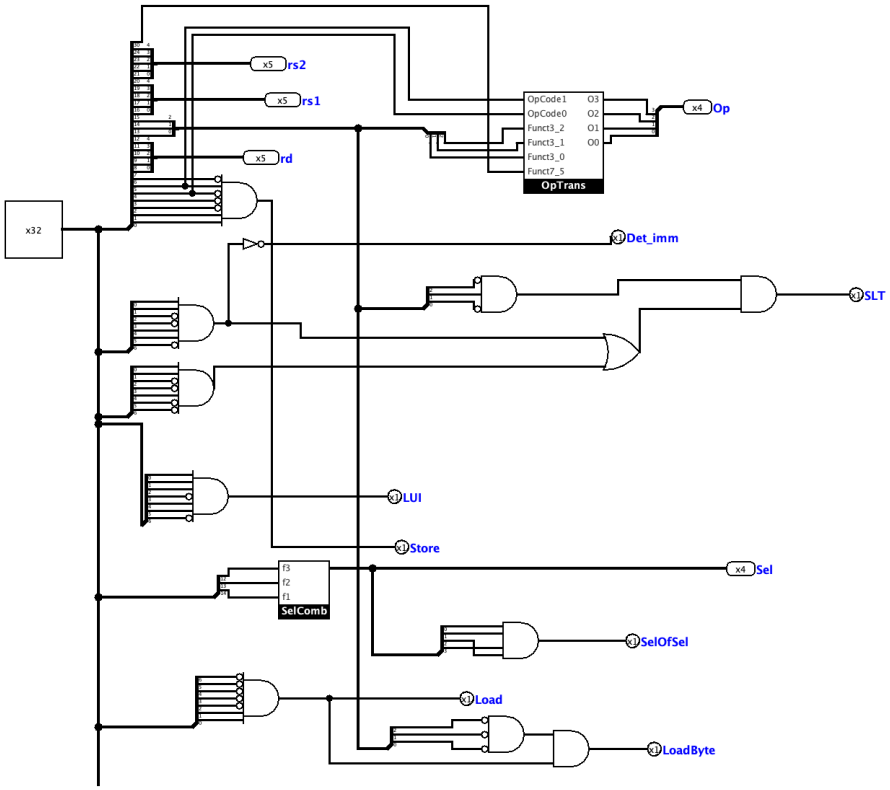
SPLIT: This sub-circuit is used to form a valid 32 bit value from a load byte operation (a byte is 8 bits) so when a byte is loaded from memory as output, the rest of the 24 bits of the 32 bit value is undefined. This subcircuit takes in the data output of the memory as well as the select bits which were sent to the memory component to determine the byte being selected. It uses Sel to determine the location of the valid bits in the input and with the use of muxes, is able to select the correct byte (based on Sel) and sign extend that value to get a valid/well-defined 32 bit value as output



OpTrans: This sub-circuit is used to translate the The 2nd MSB (most significant bit) of the opcode and the 3rd MSB of the opcode as well as the three bit funct3 value of the instruction and the 5th bit of the funct7 value of the instruction to determine the ALU opcode of the instruction being executed. It was created using combinational analysis, and it provides (as output) the ALU opcode which the ALU uses to determine the operation it needs to execute.



OpSplitter: This sub circuit manages the majority of the RISC-V processor's functionality. Firstly it splits the input instruction into a rs1 (an input register - where applicable), rs2 (an input register - where applicable), and rd (a destination register). It also retrieves the ALU opcode with the OpTrans component as discussed above. It creates an output Det_imm which determines whether an immediate value should be used over the value in rs2 by asserting whether the instruction opcode does not match an R-type instruction (the only instruction type where an immediate is not used). It also creates SLT output, which checks the opcode and funct3 values to determine if the program the instruction is an SLT or SLTI operation. It also creates an LUI output by asserting if the instruction opcode matches the LUI opcode. It also creates a store output which asserts if the instruction opcode matches the store opcode. It also creates a load output which asserts if the instruction opcode matches the load opcode.
The sel output it creates is the output from the Selcomb subcircuit discussed above (and is only used when a word is being loaded or stored). SelofSel is an output used to decide whether the Sel output above is used or the nopCheck sub circuit's output val "sel" used - hence the name Selector of Selectors => sel of sel. The load byte output checks if the opcode matches the load operation and funct3 of the instruction matches the load byte instruction's funct3 value - essentially asserting that the instruction is a load byte instruction.



RISCV: This is the overarching circuit of the RISCV processor. It feeds the different register values obtained from OpSplitter into the register file. It uses mux to determine whether rs2's value or the immediate should be used in computation; it does this using

Det_imm as the selector bit. It also uses SLT as a selector to choose between the ALU output or the (sign-extended) blackbox comparator output as the result of computation. It also uses LUI from OpSplitter to determine whether to use the Immediate or result of previous selection as the output of computation. It then chooses between the output of the memory component or the previously discussed computation using load or store (the output bits from OpSplitter indicating if the operation is a load or a store) as the selector bit to form the writeback value. Closer to the memory component, we choose between the raw output value or the sign extended output value (from the split subcircuit) using the loadbyte output of OpSplitter as a selector bit for the mux.
For the str input of the mux, The circuit sends in a 1 if the operation is a store operation and there is not a nop. For the ld input of the mux, The circuit sends in a 1 if the operation is a load operation and there is not a nop. Rs2 is the data input to the memory component (as detailed in the RISCV ISA)