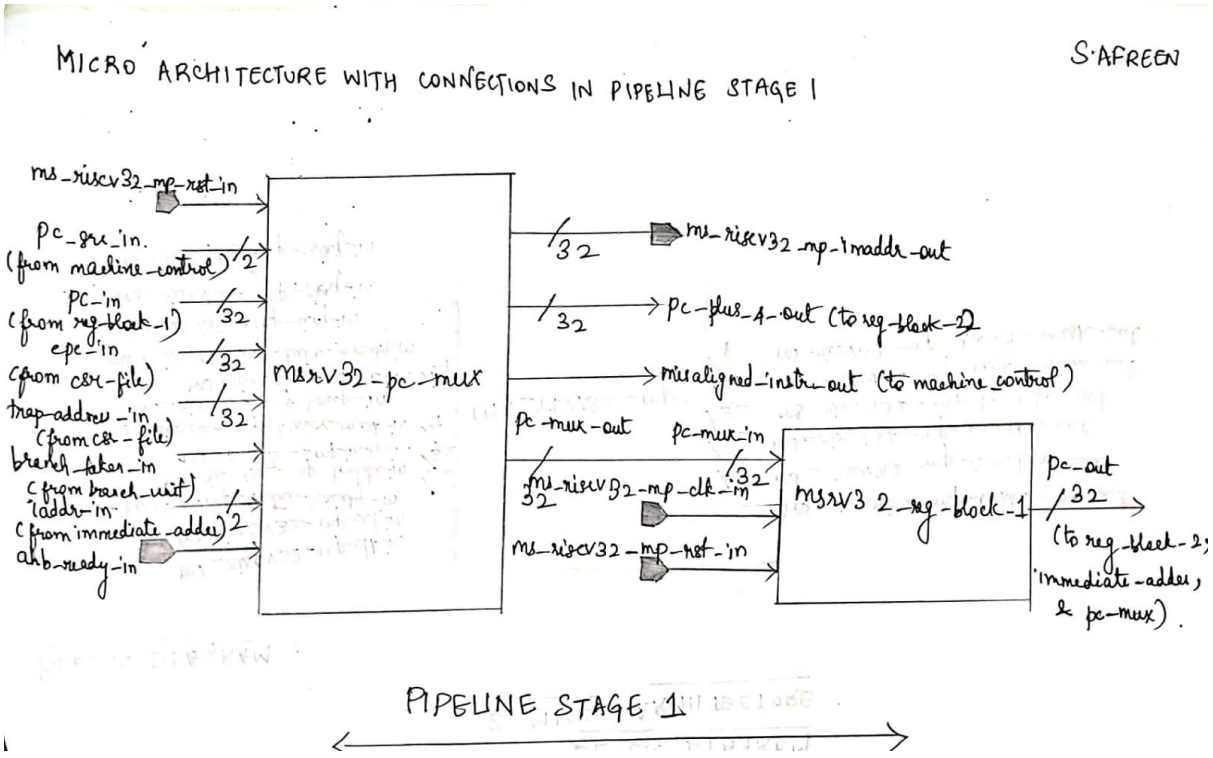
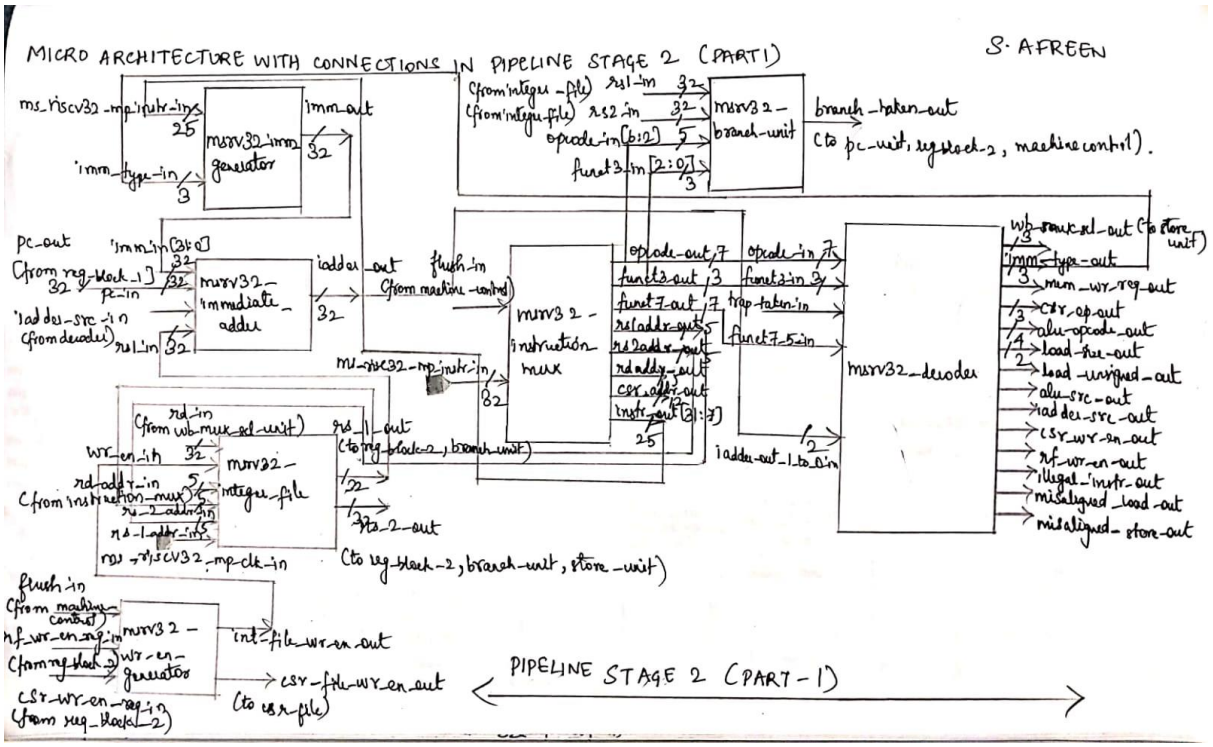
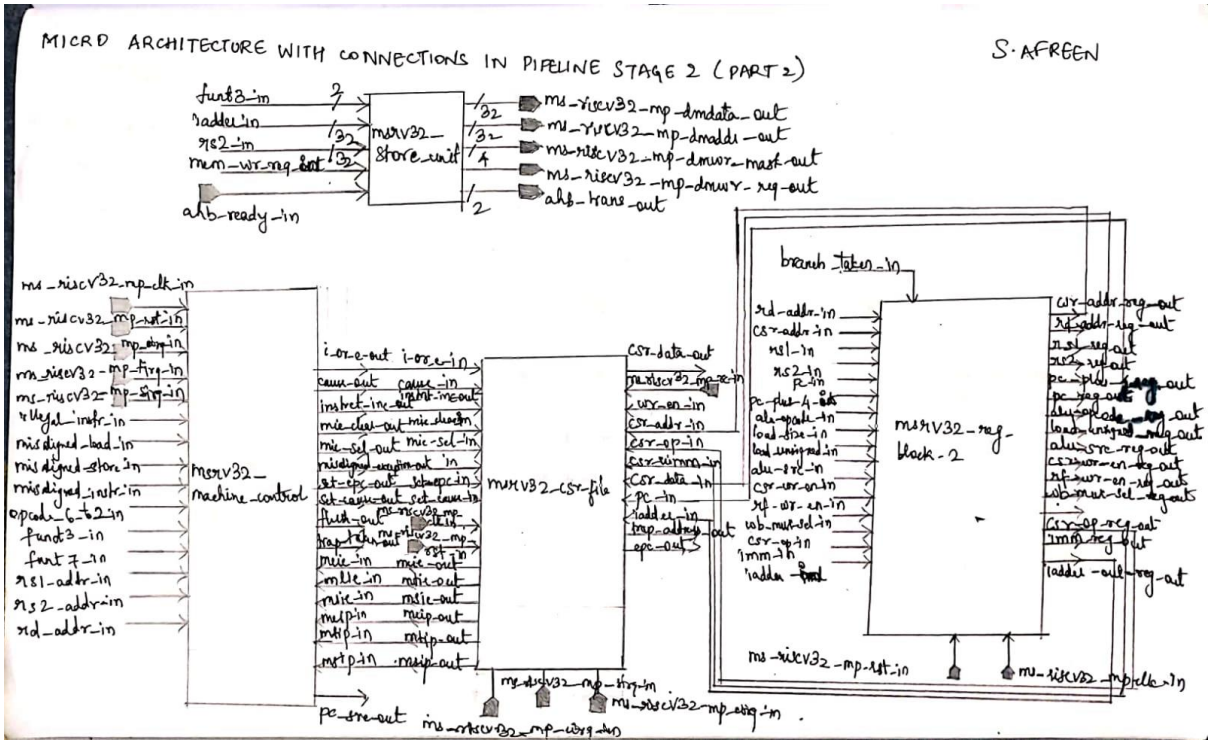
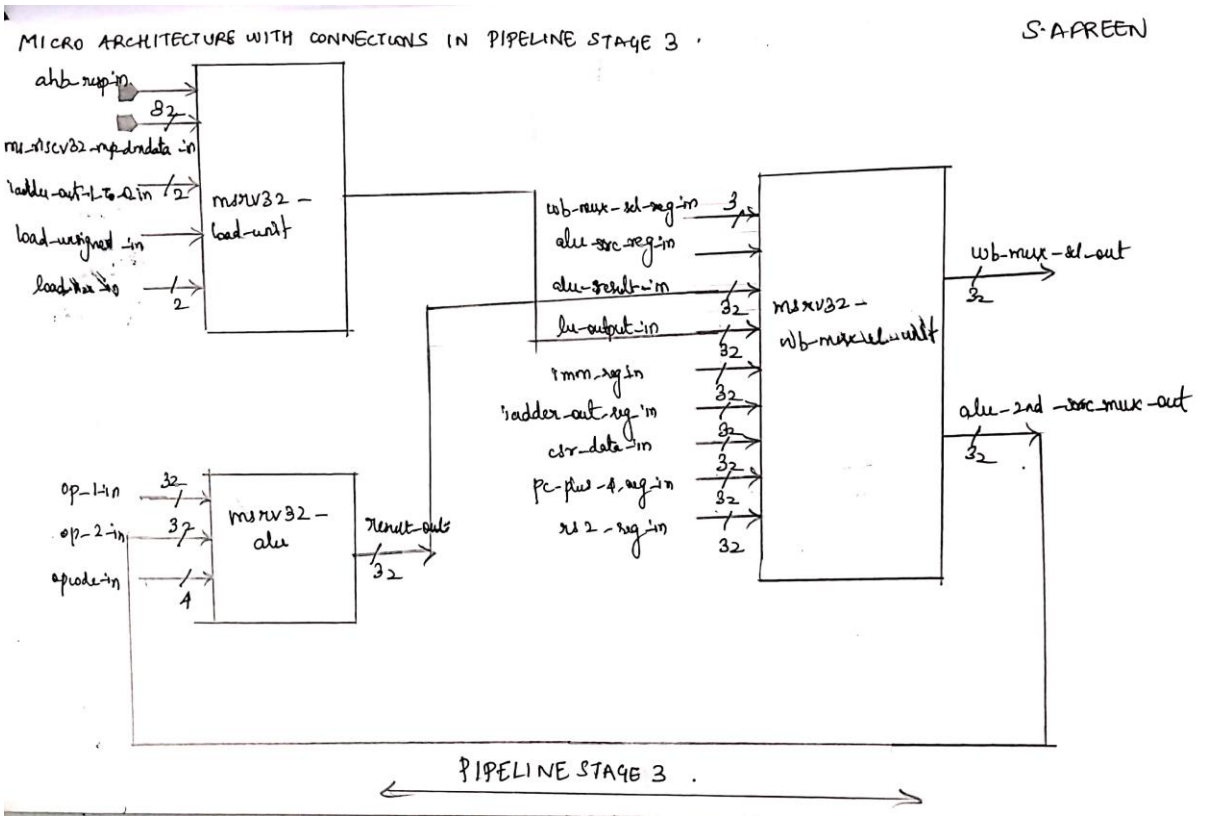
Project: RISC-V RV32I Multi stage pipeline processor RTL Design

FUNCTIONALITY:






MICRO ARCHITECTURE WITH CONNECTIONS IN PIPELINE STAGE 2 (PART 2)
S·AFREEN
msrv32_store_unit
msrv32_machine_control
msrv32_csr_file
msrv32_reg_block_2
branch_taken_in
ahb_ready_in


MICRO ARCHITECTURE WITH CONNECTIONS IN PIPELINE STAGE 3.
S·AFREEN
msrv32_load_unit
msrv32_wb_mux_sel_unit
msrv32_alu
PIPELINE STAGE 3

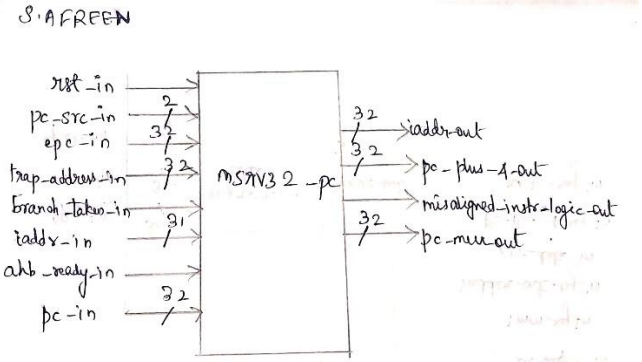
1. TOP MODULE BLOCK DIAGRAM:

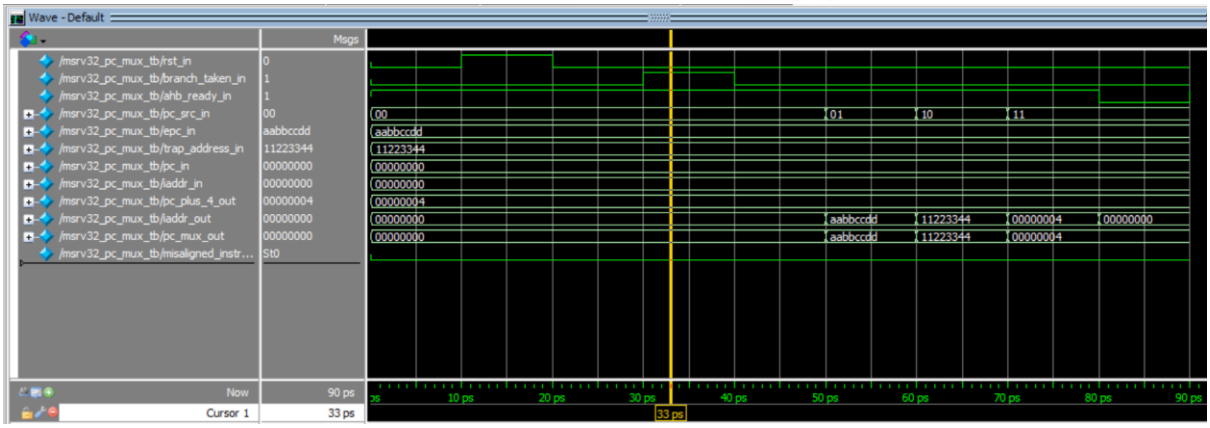
The top module contains inputs namely, system clock, system reset, data obtained from external memory, signal indicating that the instruction memory is ready to sample the address and start sending the instruction, signal indicating that the data memory is ready to sample data during write operation (from processor), signal indicating that the data memory is ready to drive data during read operation (to processor), signal indicating whether the transfer is a non-sequential transfer or an IDLE transfer, signal containing instruction obtained from the external memory, signal containing current value read from the real time counter, signal indicating an external interrupt request, signal indicating a timer interrupt request, signal indicating a software interrupt request, and outputs namely, signal indicating request to write data, signal containing the address of the instruction that the core wants to obtain from the memory, signal containing the address of the memory position in which the data will be stored (for write operation) and signal containing address of the memory position where the data to be obtained is present (for read operation), signal containing the data to be stored in memory, and signal containing mask of 4 byte-write enable bits.

```
# ------------------------------------------------------
#            JAL is done succesfully
# ------------------------------------------------------
#
#
# env_cfg_db_h.no_of_iterations = 200
#
#
# JAL Command Working Perfectly
#
#  the following list of command are working fine
#        '{"JAL"}
#  and test result is
#
#
# -------------------------------------------
# 8888888b.      d8888  .d8888b.   .d8888b.
# 888   Y88b    d88888 d88P  Y88b d88P  Y88b
# 888    888   d88P888 Y88b.      Y88b.
# 888   d88P  d88P 888  *Y888b.    *Y888b.
# 8888888P*  d88P  888     *Y88b.     *Y88b.
# 888       d88P   888       *888       *888
# 888      d8888888888 Y88b  d88P Y88b  d88P
# 888     d88P     888  *Y8888P*   *Y8888P*
#
```

2. PC MUX:

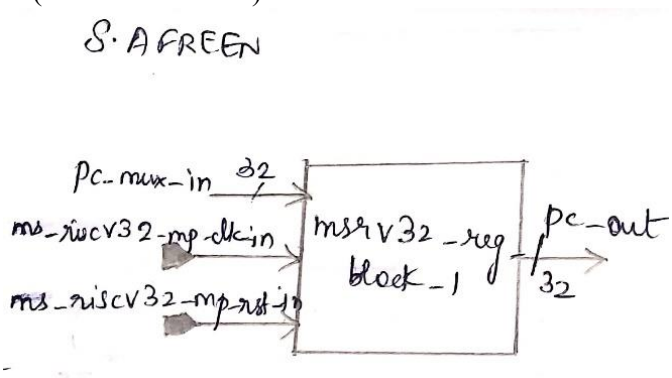
The main functionality of this module is to hold the address of the next instruction to be executed. Signal i_addr_out is used to interface with instruction memory. The program counter will be loaded according to the current state of the core. Signal pc_plus_4_out is assigned with an initial value. Misaligned instruction logic out signal generation is enabled by performing AND operation between branch_taken_in and the first bit of next_pc. iaddr_out is assigned to BOOT_ADDRESS if the reset is high else it checks if ahb_ready_in is high so that i_addr_out gets pc_mux_out.

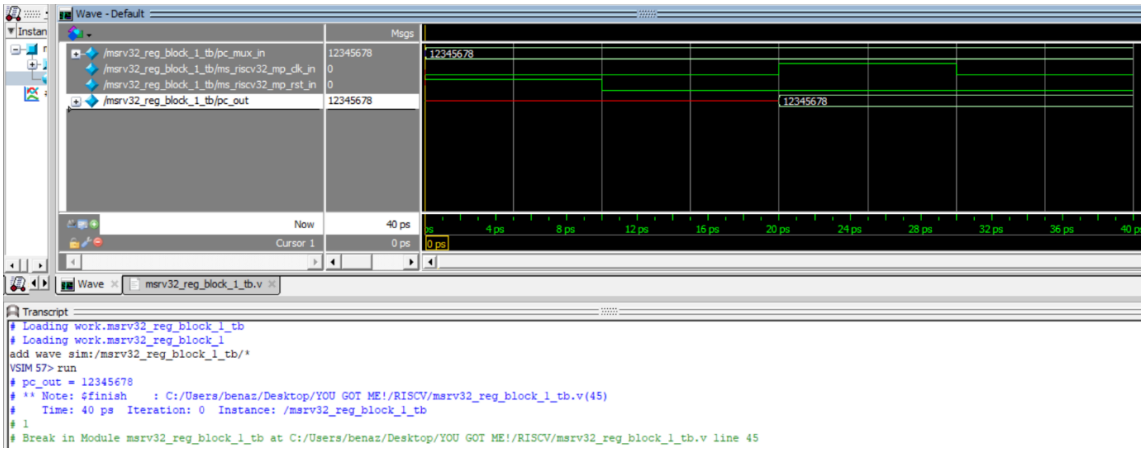




3. REG BLOCK 1:

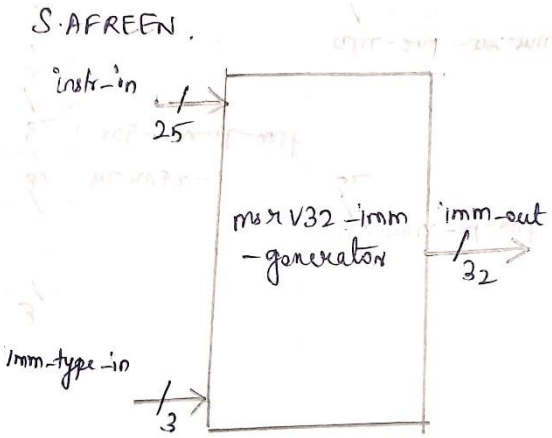
The main functionality of this module is to register the pc_mux_in and to produce an output at the posedge of clk (if there is no reset).

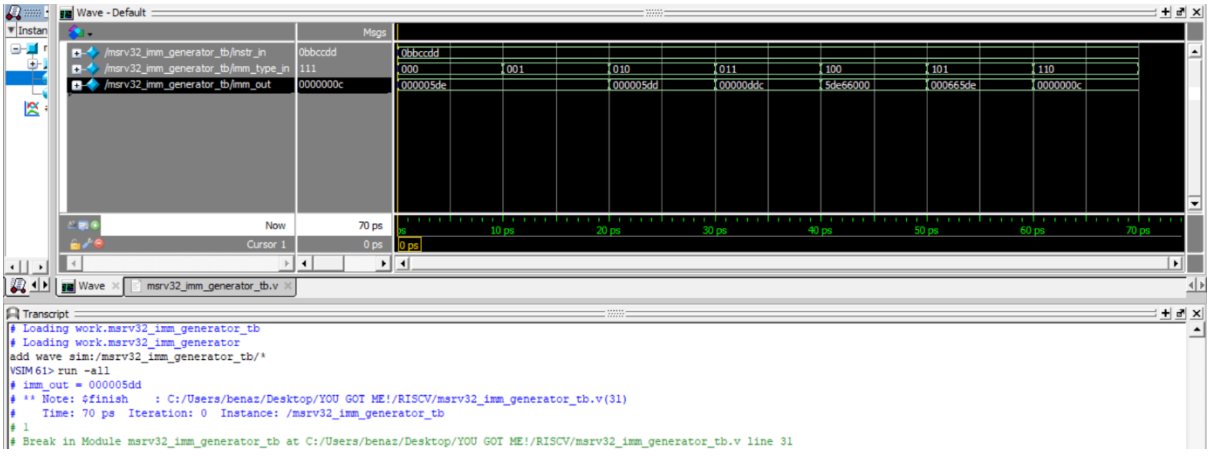




4. IMMEDIATE GENERATOR:

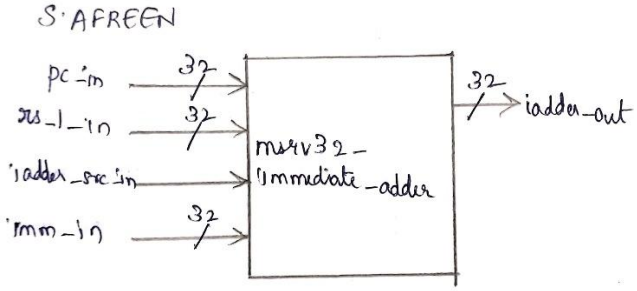
The main functionality of this module is to rearrange the immediate bits present in the instruction and if required sign extension is done to obtain a 32-bit value. This module is controlled by the imm_type_out signal that is generated by the Control Unit. The imm_type_in is generated by the Decoder Unit. Opcode [6:4] is used to distinguish between different types of instruction.

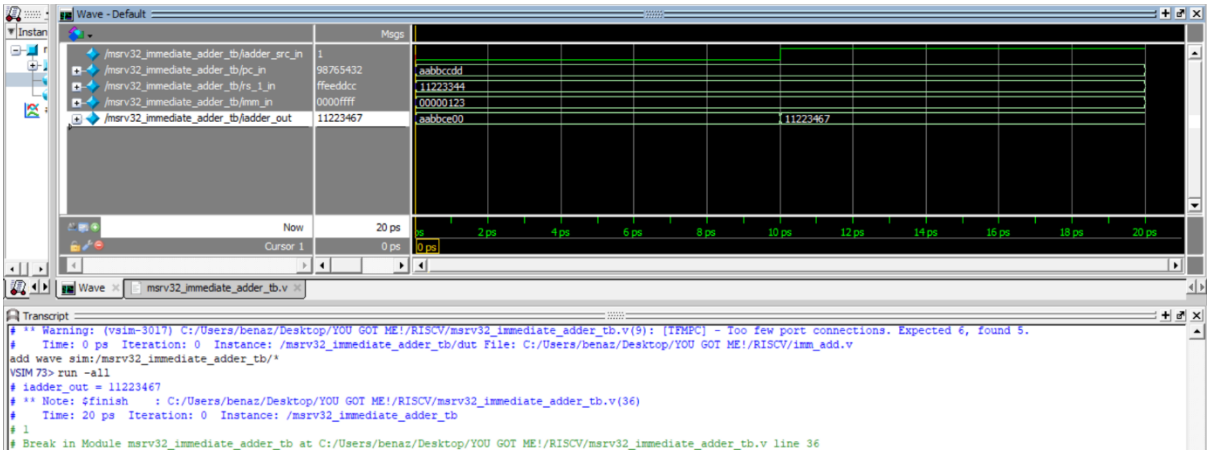




5. IMMEDIATE ADDER:

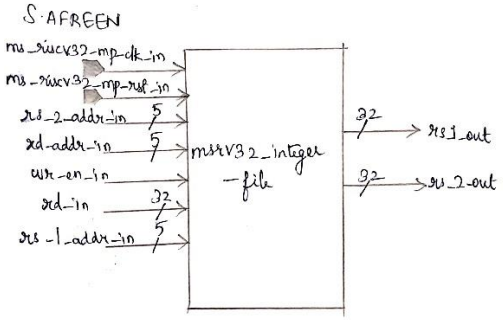
The main functionality of this module is to add the immediate value with rs1 if the core wants to perform load, store, jalr instructions. If the core wants to perform branch instruction and jal then the immediate value will be added to a PC. So, the core will jump on the new address with the help of Program Counter. The output of the module will be 32 bit address depending on the instruction format.

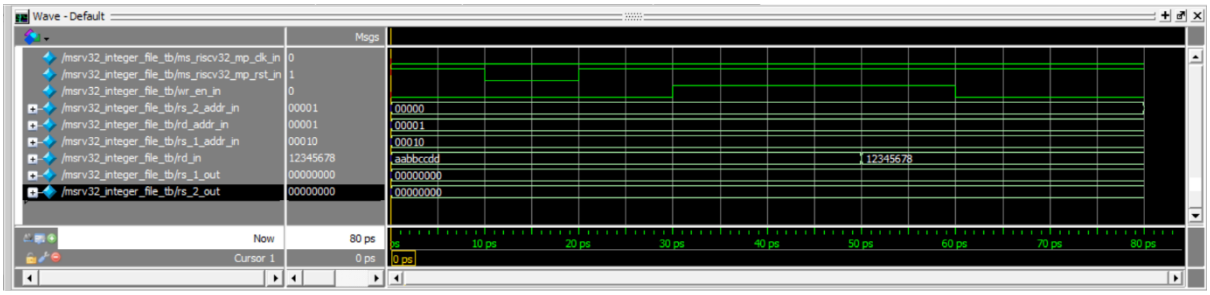




6. INTEGER FILE:

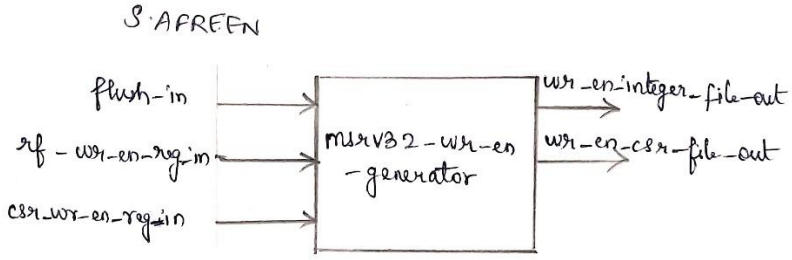
The main functionality of this module is to support read and write operations and it has 32 general purpose registers. Read operations are requested by pipeline stage 2, and data is provided from one or two registers. Write operations are requested by pipeline stage 3 and provides the data to the selected register from Write back multiplexer. reg_file[0] is hardwired with 0 and hence no write operations are permitted.

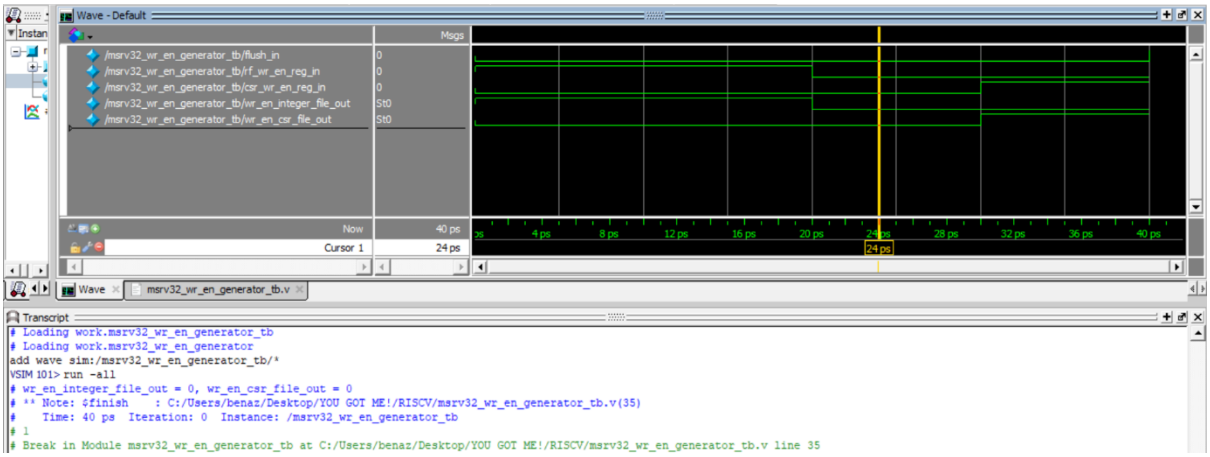




7. WRITE ENABLE GENERATOR:

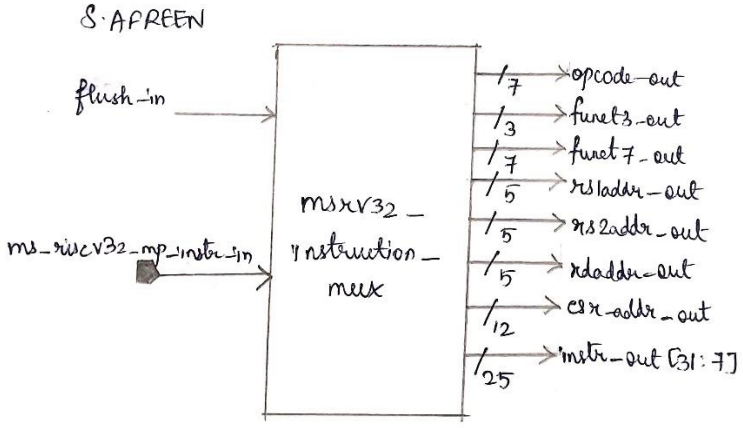
The main functionality of this module is to flush the output when set from the Machine Control. Whenever flush_in goes low output is obtained either from register file write enabled pipelined output or control status register write enable pipelined. If flush_in goes high, the output is assigned the value 1.

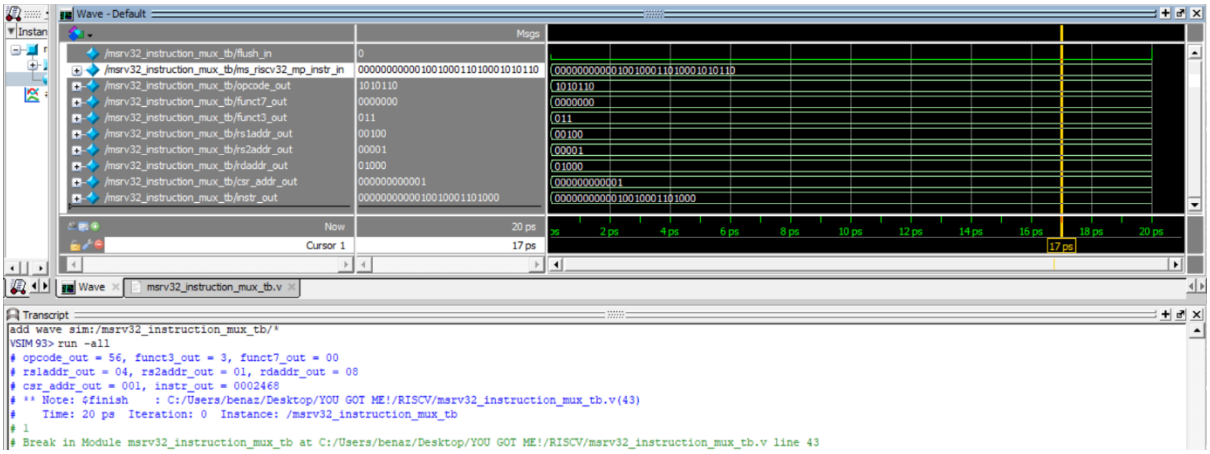




8. INSTRUCTION MUX:

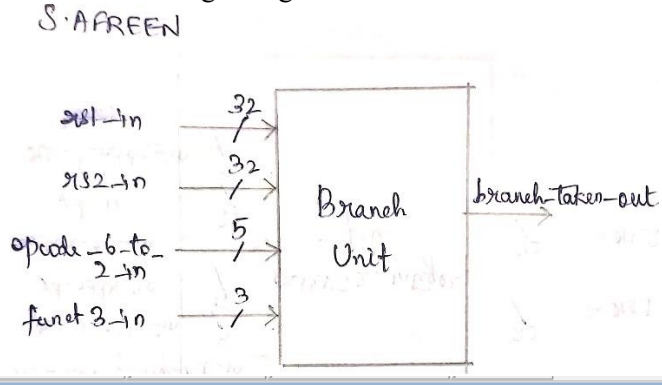
The main functionality of this module is to provide instructions to the fields that are used by other modules to perform their functionality. The module depends on 'flush' to provide the fields. Various locations of instr_mux is assigned to different outputs of the same module.

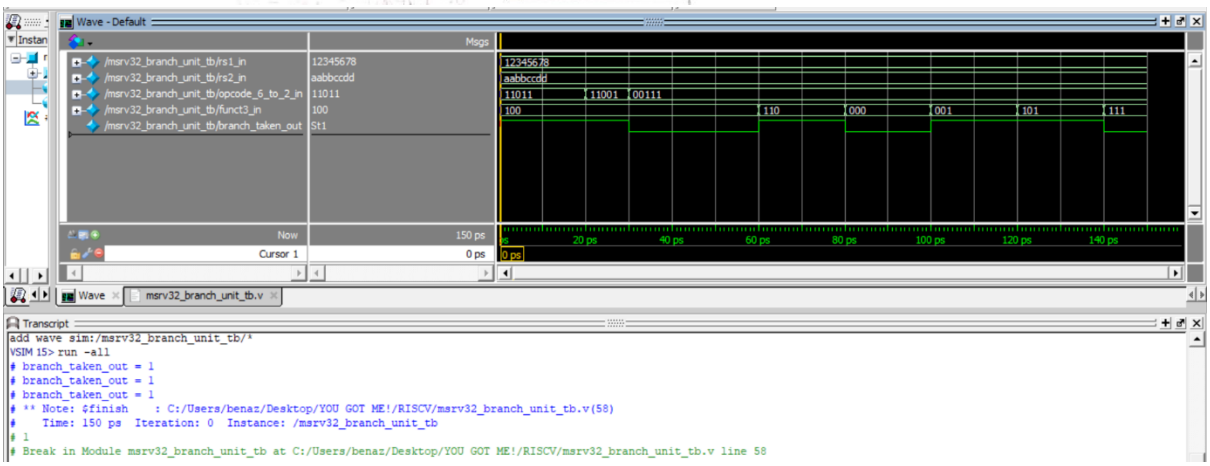




9. BRANCH UNIT:

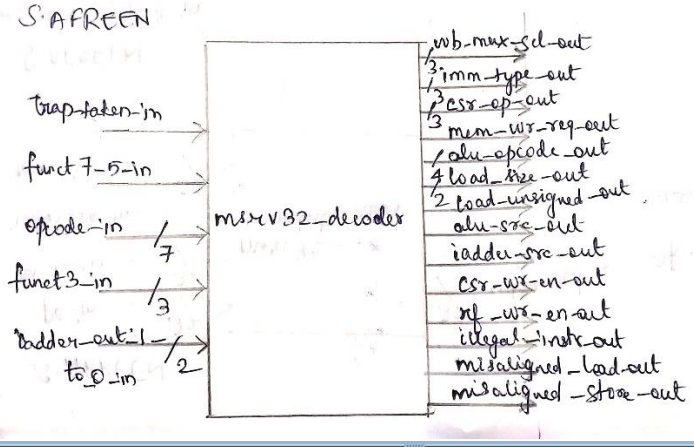
The main functionality of this module is to decide whether a branch instruction must be taken or not. The branch is decided based on the values of opcode and funct3 instruction fields as it receives two operands from the Integer Register File.

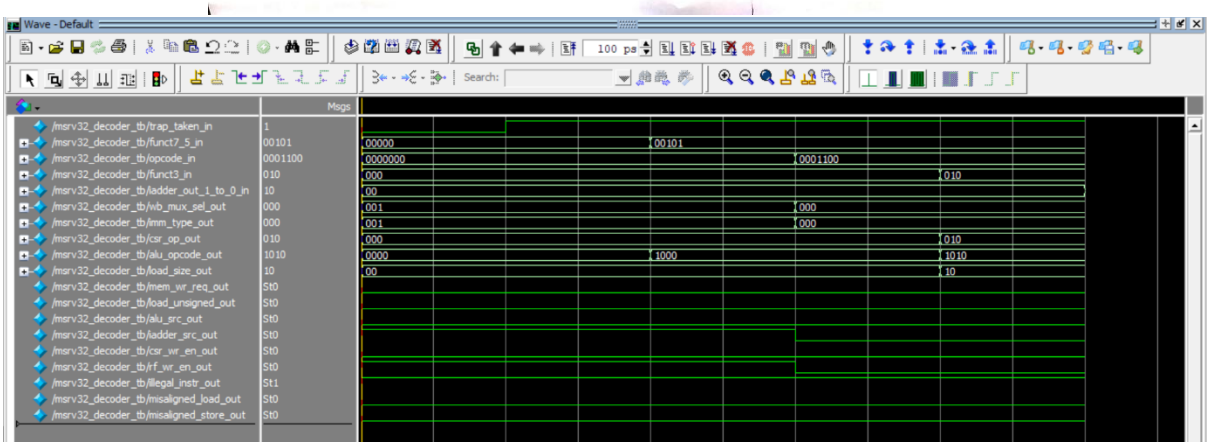




10. DECODER:

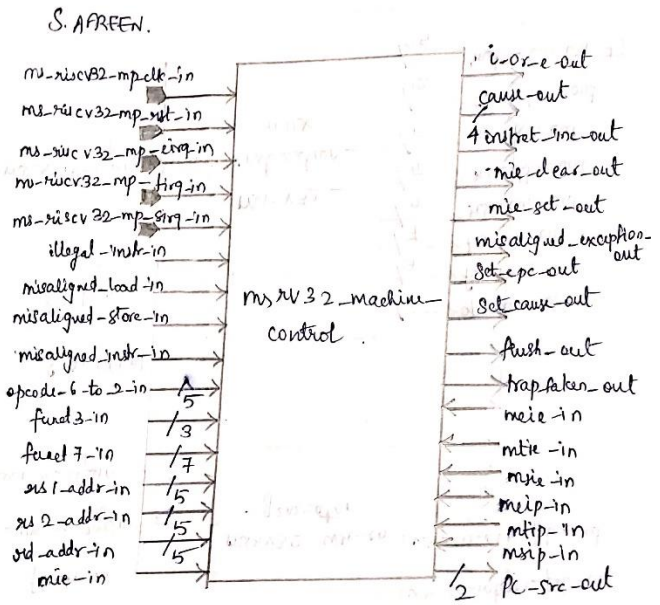
The main functionality of this module is to decode the instruction and generate the signals that control the memory, Load Unit, Store Unit, ALU, Integer File, CSR File, Immediate Generator, and Write Back Multiplexer. The output logic is defined that carries out various operations.





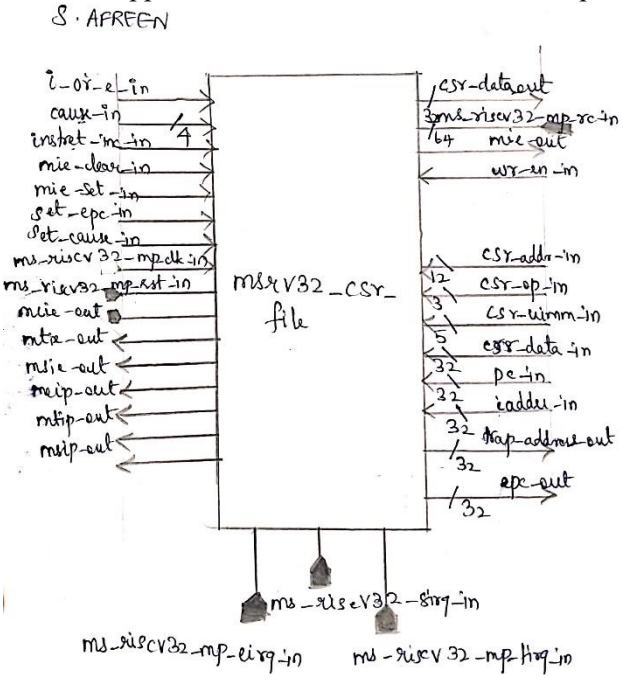
11. MACHINE CONTROL:

The main functionality of this module is to assert the outputs of internal control logic. Output logic is also defined by asserting parameters high according to conditions satisfied by the inputs.



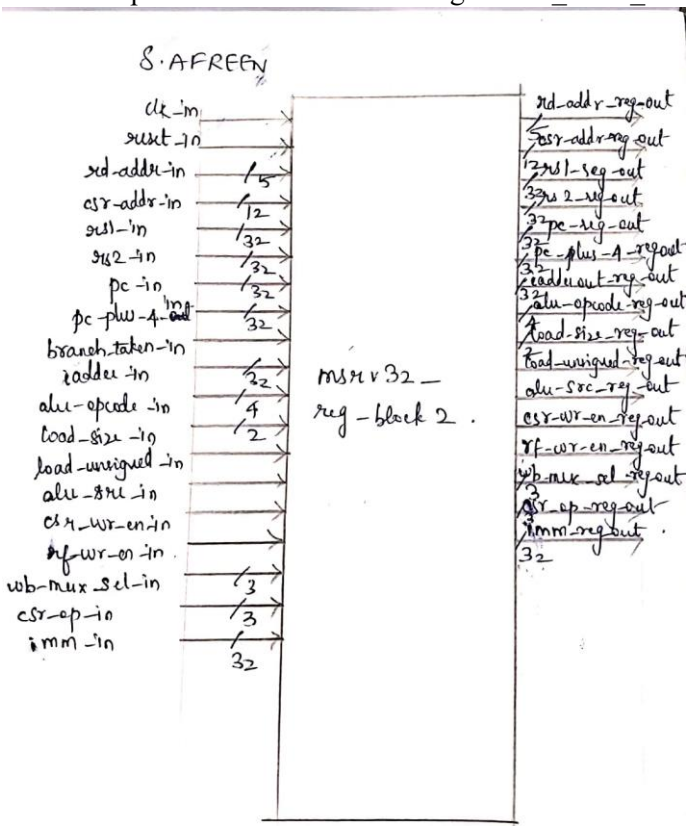
12. CSR FILE:

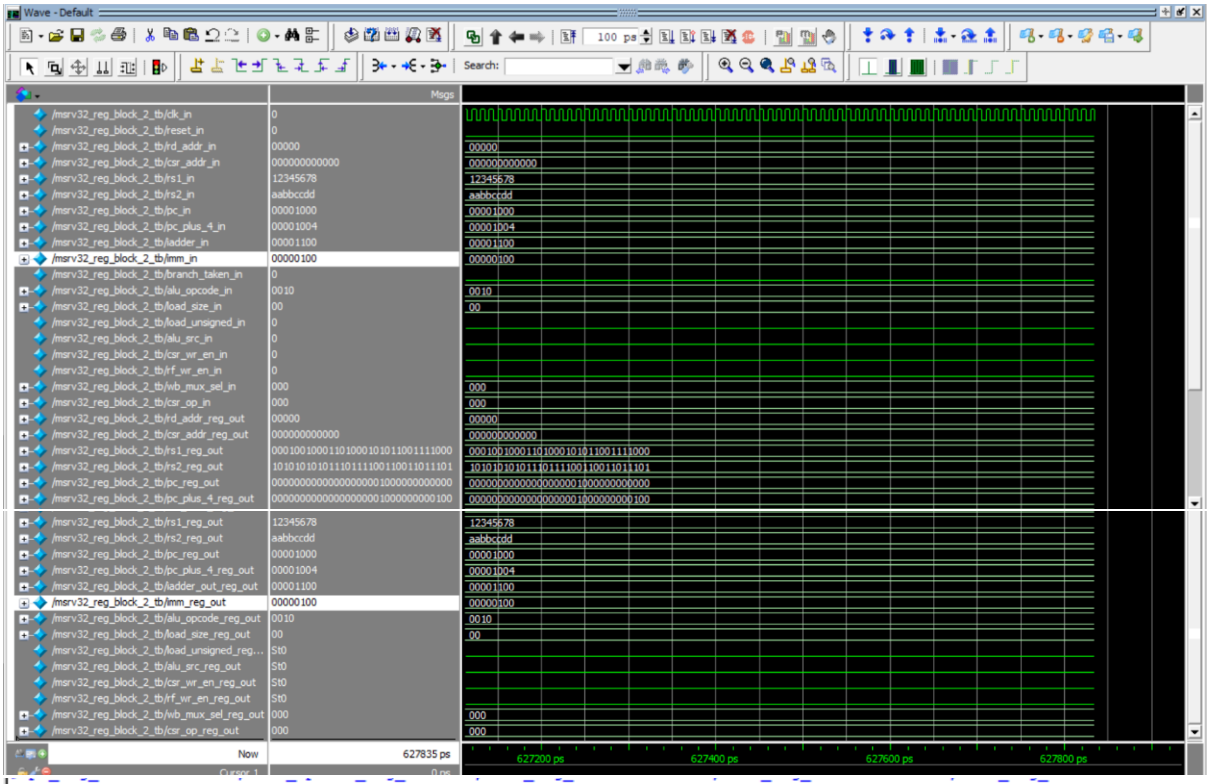
The main functionality of this module is to implement M-mode using the control and status registers. The registers can be applied with read/write, set, and clear operations.



13. REG BLOCK 2:

The main functionality of this module is to register all the inputs and produces outputs at the posedge of the clk. It also implements a 2:1 MUX having branch_taken_in as a select line.

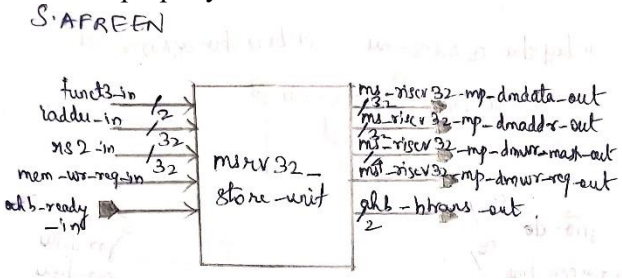


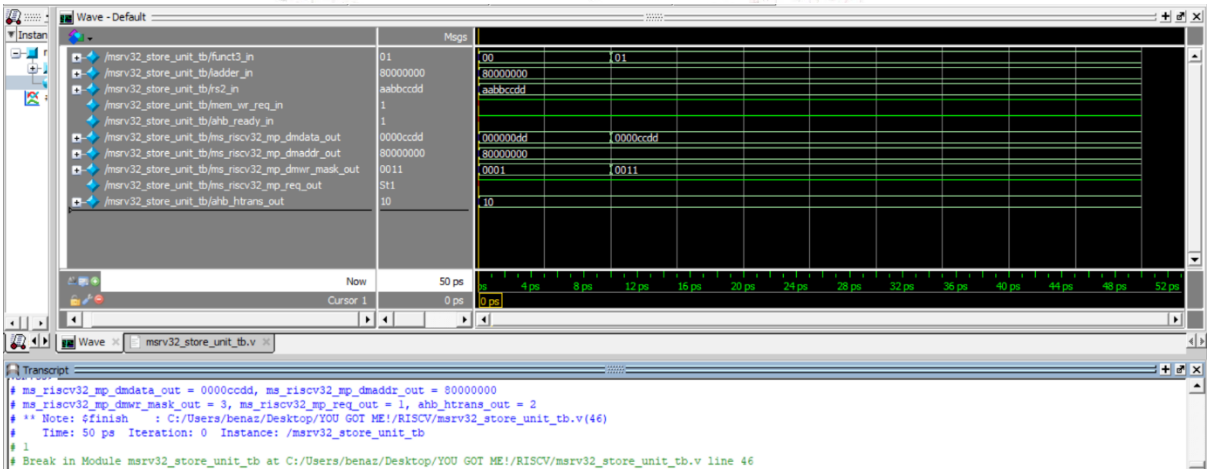



```
# pc_reg_out = 00001000, alu_opcode_reg_out = 2, imm_reg_out = 00000100, rs1_reg_out = 12345678, rs2_reg_out = aabbccdd
# pc_reg_out = 00001000, alu_opcode_reg_out = 2, imm_reg_out = 00000100, rs1_reg_out = 12345678, rs2_reg_out = aabbccdd
```

14. STORE UNIT:

The main functionality of this module is to drive the signal that connects with the external data memory. This module ensures that the data to be written is placed in the right position whereas other bits are masked properly.

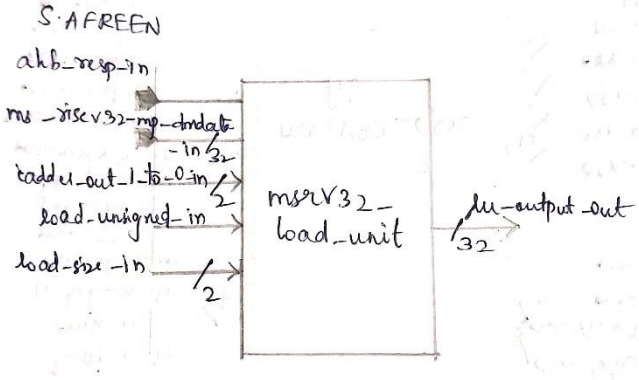


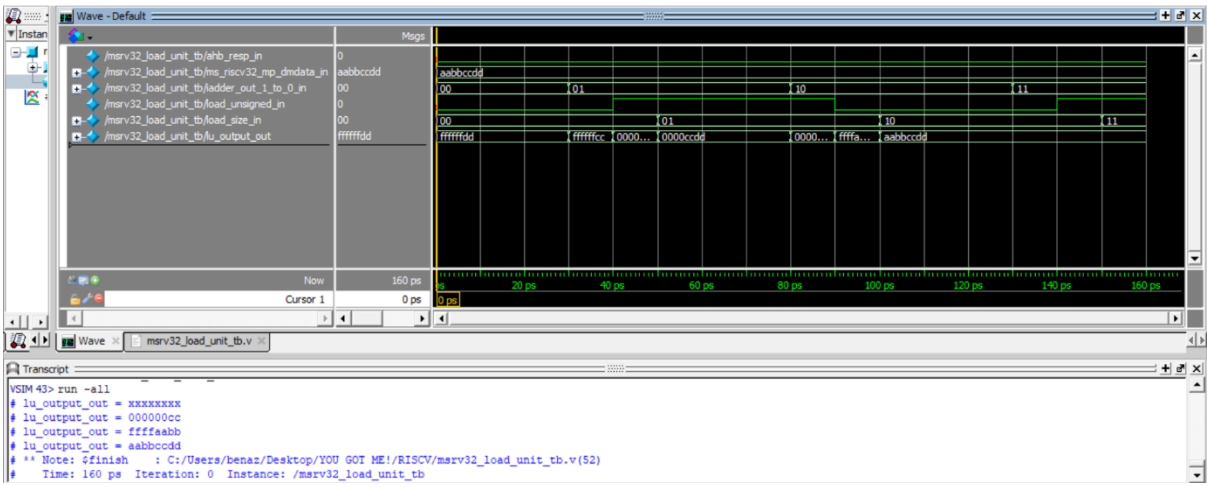



```
# ms_riscv32_mp_dmdata_out = 0000ccdd, ms_riscv32_mp_dmaddr_out = 80000000
# ms_riscv32_mp_dmwr_mask_out = 3, ms_riscv32_mp_req_out = 1, ahb_htrans_out = 2
# ** Note: $finish    : C:/Users/benaz/Desktop/YOU GOT ME!/RISCV/msrv32_store_unit_tb.v(46)
#    Time: 50 ps  Iteration: 0  Instance: /msrv32_store_unit_tb
# 1
# Break in Module msrv32_store_unit_tb at C:/Users/benaz/Desktop/YOU GOT ME!/RISCV/msrv32_store_unit_tb.v line 46
```

15. LOAD UNIT:

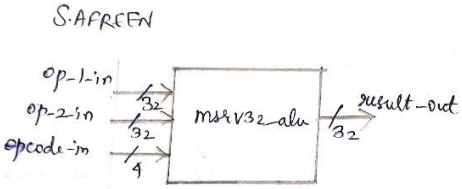
The main functionality of this module is to read the data input signal and based on the load instruction type a 32-bit value is formed. The formed 32-bit value on the output can then be written in the Integer Register File.

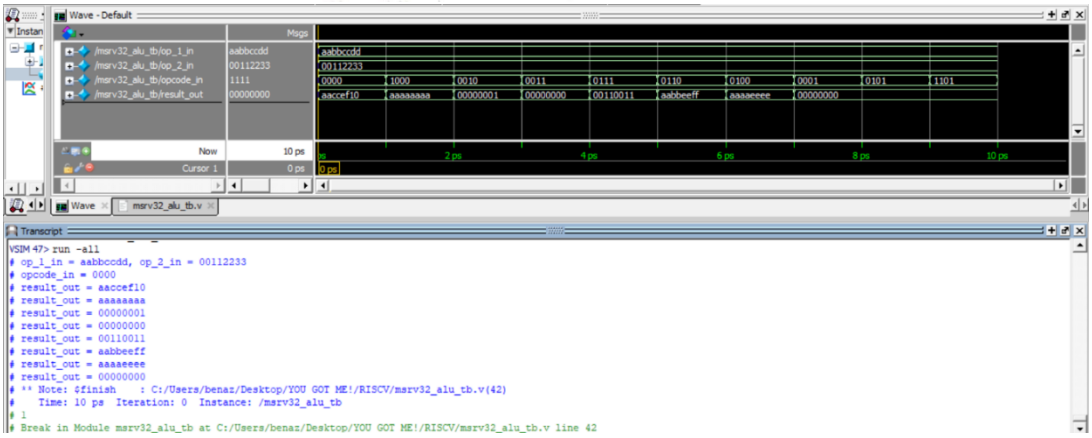




16. ALU:

The main functionality of this module is to apply ten logical and arithmetic operations to two 32-bit operands in parallel, to give output based on the result selected by opcode_in. The opcode values are assigned to facilitate instruction decoding.





17. WB MUX SELECTION UNIT:

The main functionality of this module is based on the wb_mux_sel_reg_in signal according to which wb_mux_out signal is assigned. The inputs are ALU output, Load unit, Load unit immediate value, RS1+IMM/PC+IMM, CSR, PC+4 (for jal, jalr)