Course Project 2022

EE224 - Digital Design

IITB CPU

Alisha: 21D070011
Maalavika C S: 210070050
Shubhhi Singh: 210070084
Vedika Rathi: 21D070089

Course Instructor : Prof. Virendra Singh

{Link to Repository}

COMPONENTS USED

1. Memory
2. Register File, which includes the PC Register
3. Four Temporary registers
4. ALU with ADD, NAND, SHIFT7 operations
5. SE7 (To extend 9 bits to 16 bits)
6. SE10 (To extend 6 bits to 16 bits)
7. LMSM Block (To produce address of registers for the LM and SM instructions)
8. One 8x1 16-bit MUX
9. Four 4x1 16-bit MUX
10. One 4x1 3-bit MUX
11. Three 2x1 16-bit MUX
12. One 1x2 16-bit DEMUX

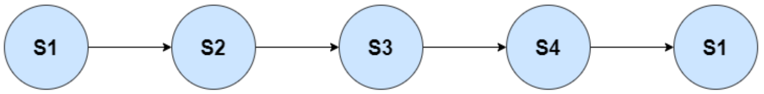
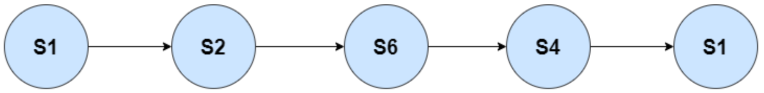
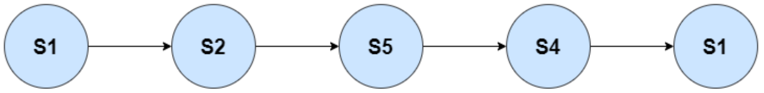
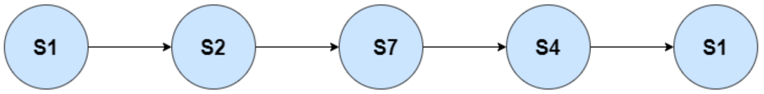
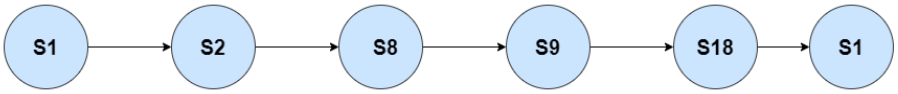
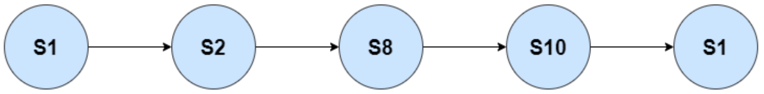
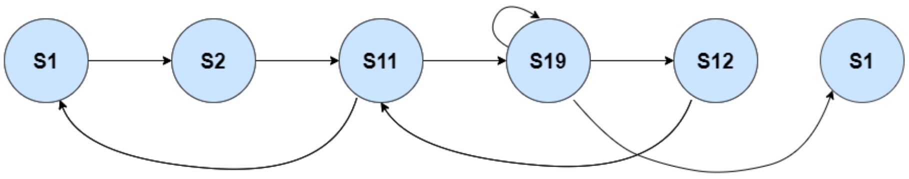
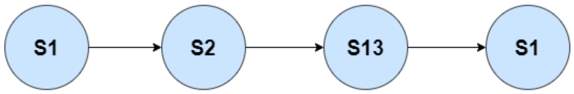
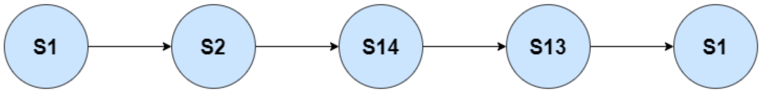
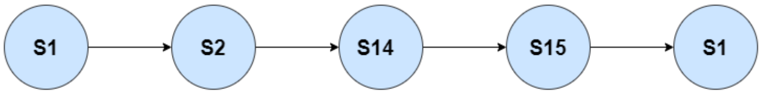
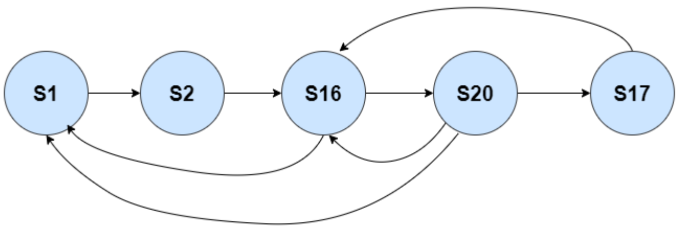
For the datapath and control path, we have incorporated both in one file, named as ‘main.vhdl’. We have 20 states, which have been defined below.

For the simulation, there is 1 DUT file, 1 Testbench, and a ‘TRACEFILE.txt’. There has been a dummy output mapped to ‘0’ always, and the inputs in the TRACEFILE are clock and reset.

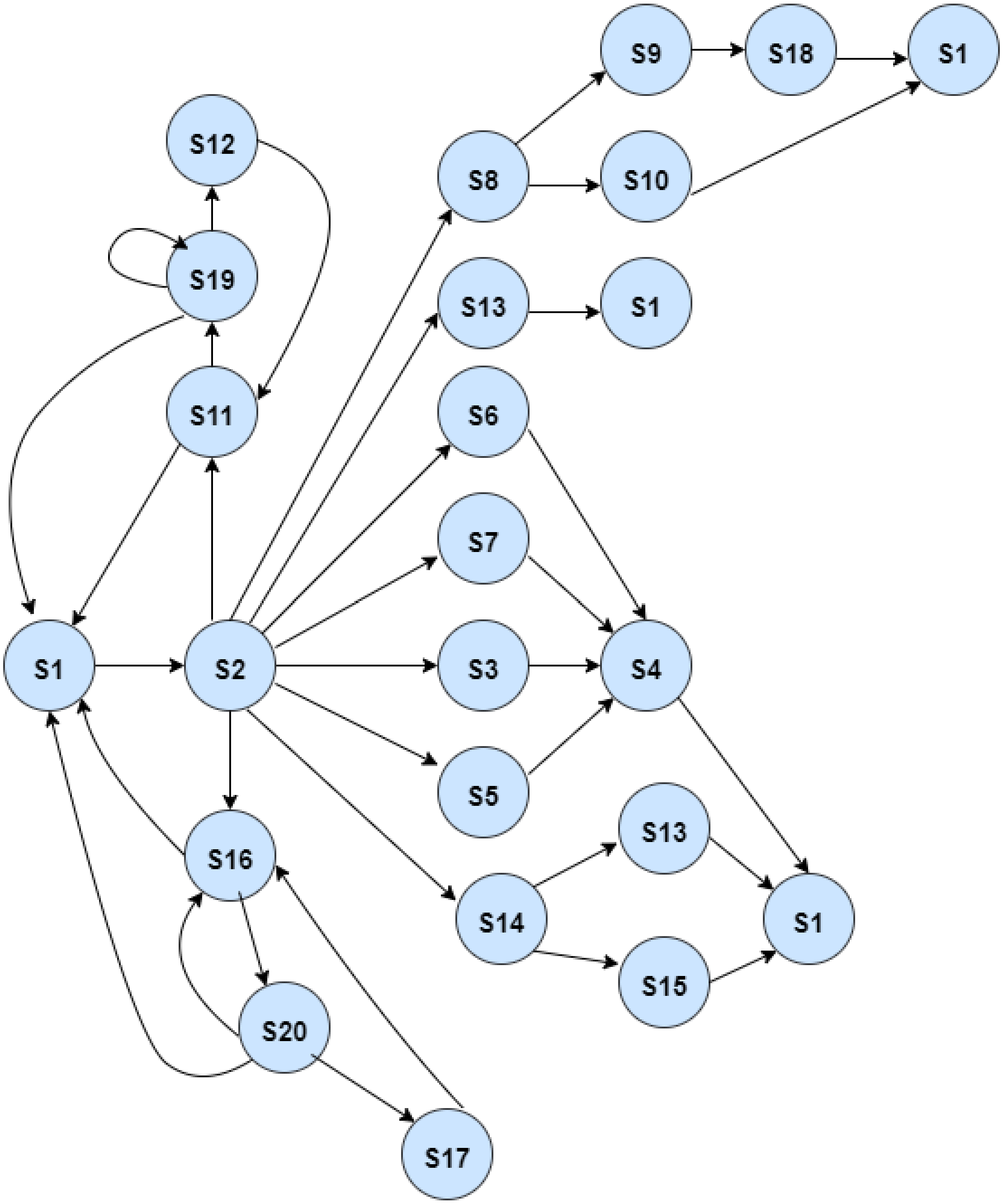
To check the various instruction outputs in the RTL Simulation, one has to run the analysis and synthesis with the DUT as top level entity, and run RTL simulation. The memory and register files need to be initialized as per need, and the signals for the present state, the values in the various intermediate signals as well a-s the memory and register file can be viewed in the ModelSim Window.

The **14 instructions** have been checked by us and it yields the **correct intermediate signals** and results. The RTL Simulations screenshots with sample instructions have been given at the end of the document.

STATE FLOW DIAGRAM

ADD,ADC,ADZ	
ADI	
NDU,NDC,NDZ	
LHI	
LW	
SW	
LM	
BEQ	
JAL	
JLR	
SM	

State Diagram Net



HARDWARE FLOWCHARTS

S1	PC → M_ADD, T4, ALU_A M_DATA → T1 +1 → ALU_B *Memory is a two-dimensional integer indexed array. To move to next mem address, integer 1 is added.* ALU_X → PC	ADD MEM_R T1_W PC_W T4_W
S2	T1_11-9 → RF_A1 RF_D1 → T2 T1_8-6 → RF_A2 RF_D2 → T3 T1_5-0 → SE10	T2_W T3_W

S3	If (t1_op(1 downto 0) = “00”) then
	T2 → ALU_A
	T3 → ALU_B
	ALU_X → T2
	    Update C and Z
	Elsif (t1_op(1 downto 0) = “10”) then
	  If (carry_present = ‘1’) then
	    T2 → ALU_A
	    T3 → ALU_B
	    ALU_X → T2
	    Update C and Z
	  Else
	    T1_5_3 → RF_A1
	    RF_D1 → T2
	Elsif (t1_op(1 downto 0) = “01”) then
	  If (zero_present = ‘1’) then
	    T2 → ALU_A
	    T3 → ALU_B
	    ALU_X → T2
	    Update C and Z
	  Else
	    T1_5_3 → RF_A1
	    RF_D1 → T2
	Else
	  null
		ADD
		T2_W

S4	Y → RF_A3 T2 → RF_D3	RF_W

S5	If (t1_op(1 downto 0) = “00”) then
	T2 → ALU_A
	T3 → ALU_B
	ALU_X → T2
	      Update C and Z
	Elsif (t1_op(1 downto 0) = “10”) then
	  If (carry_present = ‘1’) then
	    T2 → ALU_A
	    T3 → ALU_B
	    ALU_X → T2
	    Update C and Z
	  Else
	    T1_5_3 → RF_A1
	    RF_D1 → T2
	Elsif (t1_op(1 downto 0) = “01”) then
	  If (zero_present = ‘1’) then
	    T2 → ALU_A
	    T3 → ALU_B
	    ALU_X → T2
	    Update C and Z
	  Else
	    T1_5_3 → RF_A1
	    RF_D1 → T2
	Else
	  null
		NAND
		T2_W

S6	SE10 → ALU_B T2 → ALU_A ALU_X → T2 UPDATE C,Z	ADD T2_W
S7	T1_8-0 → SE7 SE7 → ALU_A ALU_X → T2	SHIFT7 T2_W
S8	T3→ ALU_A SE10 → ALU_B ALU_X → T3 *If next state is S9, update Z*	T3_W ADD
S9	T3 → M_ADD M_DATA → T3	T3_W MEM_R
S10	T3 → M_ADD T2 → M_INP	MEM_W
S11	T2 → M_ADD M_DATA → T3	MEM_R T3_W

S12	T2 → ALU_A +1 → ALU_B *Memory is a two-dimensional integer indexed array. To move to next mem address, integer 1 is added.* ALU_X → T2	T2_W ADD
S13	If (t1_op(14) = ‘1’) then If (T2 = T3) then T4 → ALU_A SE10 → ALU_B ALU_X → PC Else Null Else T4 → ALU_A SE7 → ALU_B ALU_X → PC	ADD PC_W
S14	T4 → RF_D3 T1_11_9 → RF_A3	RF_W
S15	T3 → PC	PC_W
S16	R_ADD → RF_A1 RF_D1 → T3	T3_W
S17	T2 → ALU_A +1 → ALU_B *Memory is a two-dimensional integer indexed array. To move to next mem address, integer 1 is added.* ALU_X → T2	T2_W ADD
S18	T1_11_9 → RF_A3 T3 → RF_D3	RF_W

S19	R_ADD → RF_A3
	T3 → RF_D3
		if (t1_8_0 (lmsm_count) = '0') then
		RF_W<='0';
		else
		RF_W<='1';

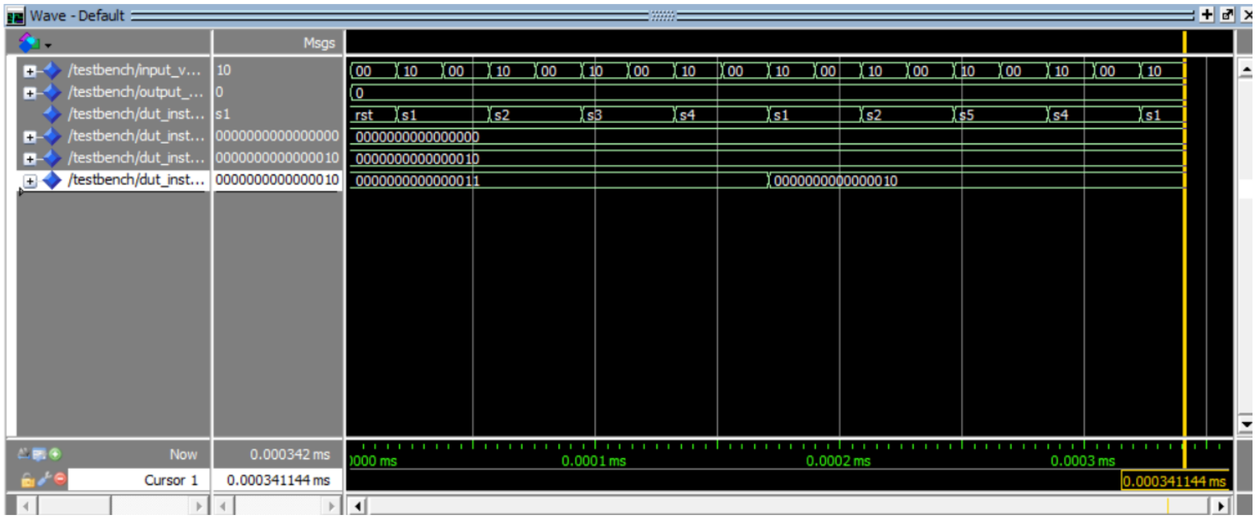
S20	T2 → M_ADD
	T3 → M_INP
		if (t1_8_0 (lmsm_count) = '0') then
		MEM_W<='0';
		else
		MEM_W<='1';

RTL SIMULATIONS

The sample instruction for each of the given instructions is stored in the array of index 1 in memory. Before running the sample instructions, we need to have suitable values stored in the registers and memory.

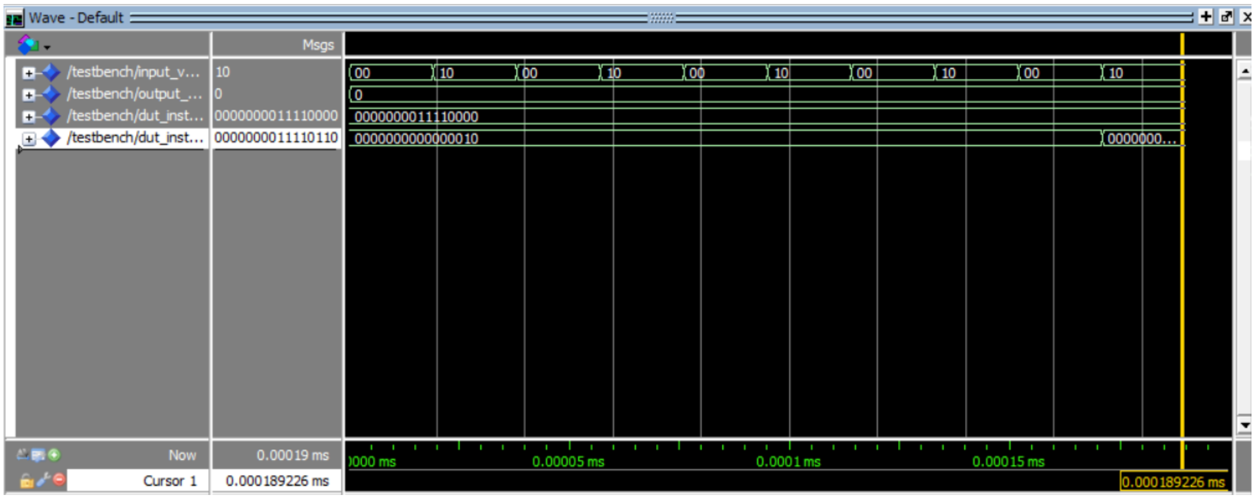
1) ADD + NDC - 0000000001010000,0010000001010010

First instruction simply adds the values in registers(0) and registers(1), and stores it in registers(2), as seen in the following. The second instruction being NDC, causes no change in the registers(2), since carry is zero due to the previous instruction.



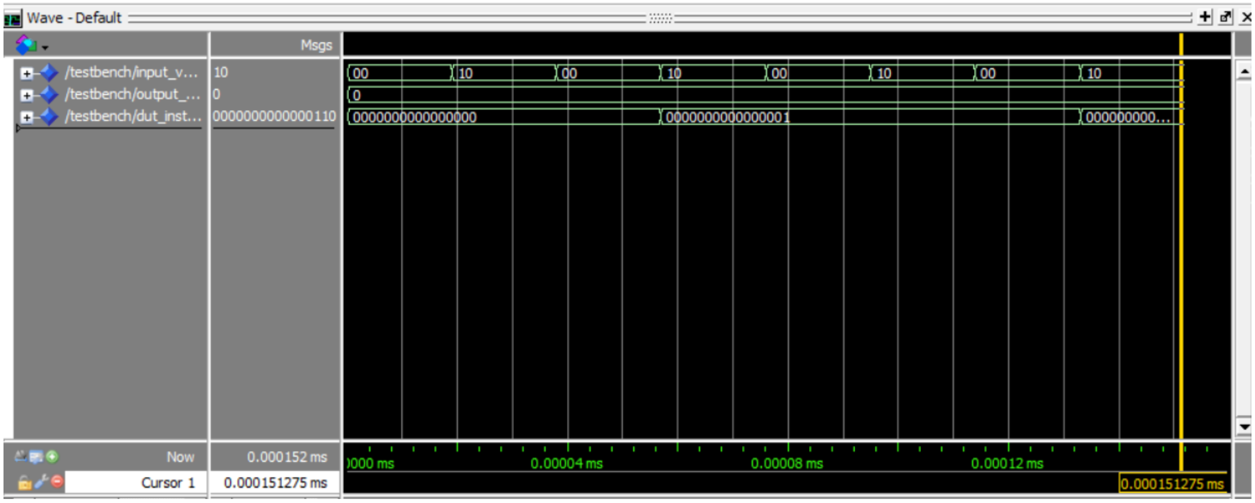
2) ADI - 0001000001000110

Signals visible here are registers(0) and registers(1).
The content of registers(0) is added to sign extended(000110), and stored in registers(1).



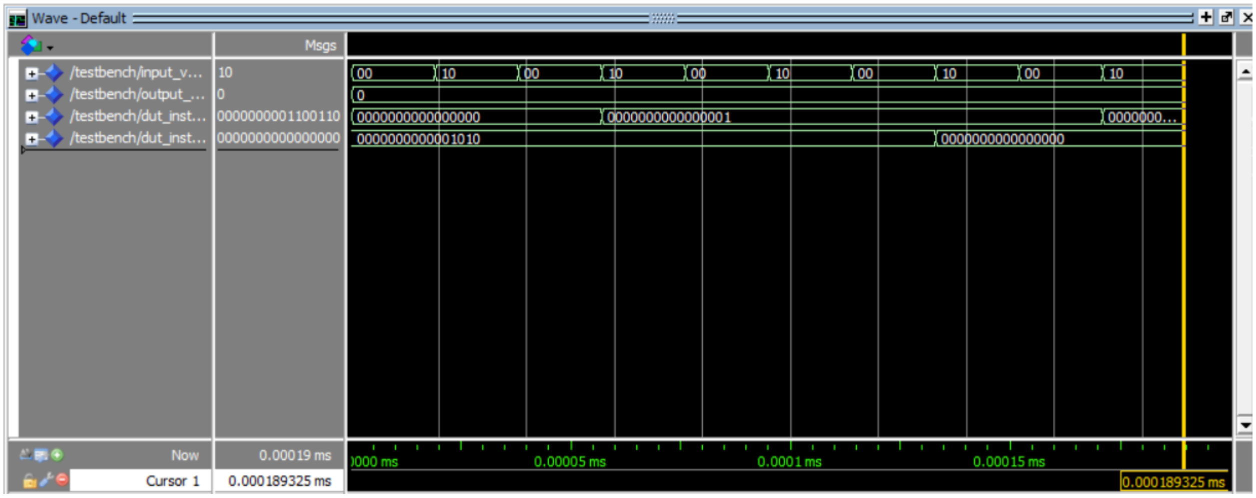
3) BEQ - 1100000001000110

Signal visible here is pc_op(the address for the next instruction in memory).
The values of registers(0) and registers(1) is compared, and when it is found to be equivalent, the pc_op signal changes to the sign extended(000110).



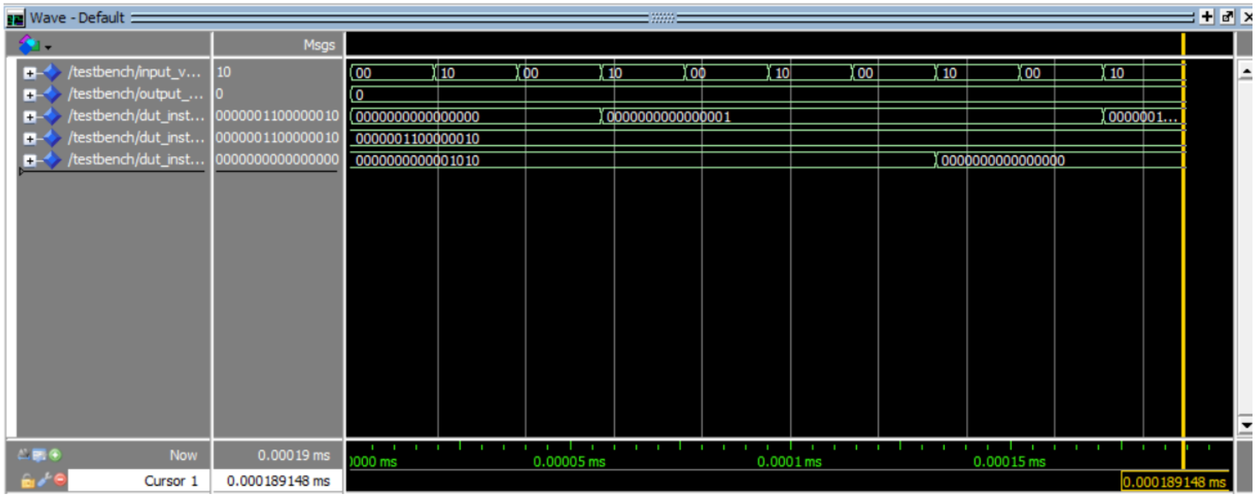
4) JAL- 1000000001100110

Signals visible are pc_op(address for the next instruction) and registers(0).
The pc_op signal changes to sign extended(001100110), and the value stored in registers(0) is set to the address of the instruction, in this case 0000000000000000.



5) JLR- 1001000001000000

Signals visible are pc_op(address for the next instruction), registers(1), and registers(0).
The pc_op signal changes to the value stored in registers(1), which is 0000001100000010, abd the address of the JLR instruction gets stored in registers(0), which is 0000000000000000.

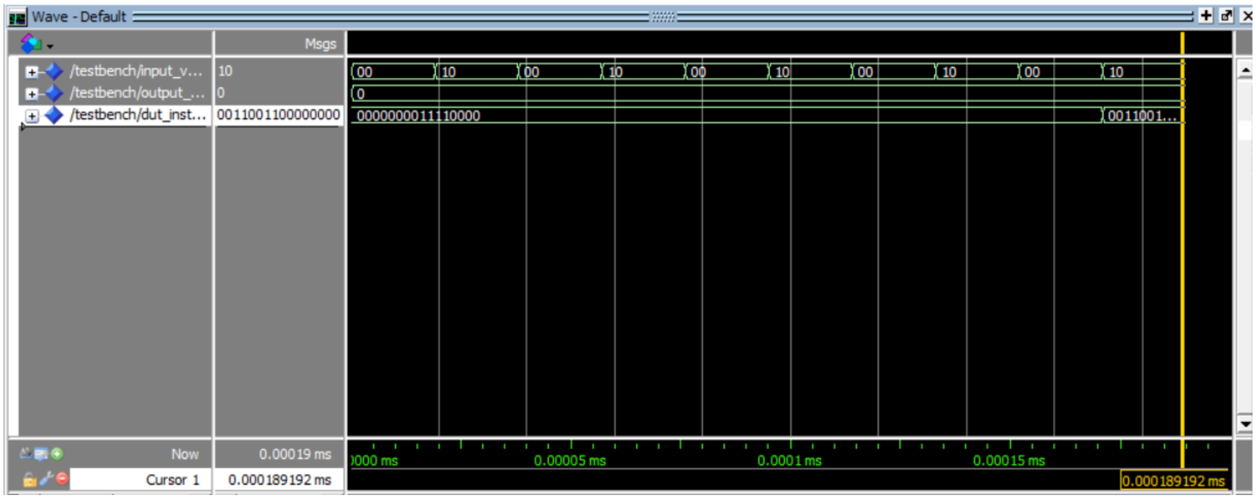


Wave - Default

	Msgs
/testbench/input_v...	10
/testbench/output_...	0
/testbench/dut_inst...	0000001100000010
/testbench/dut_inst...	0000001100000010
/testbench/dut_inst...	0000000000000000
Now	0.00019 ms
Cursor 1	0.000189148 ms

6) LHI- 0011000001100110

Signal visible is registers(0).
The 9 Imm bits, 001100110 are stored in registers(0), with 0000000 added towards the right of the Imm.

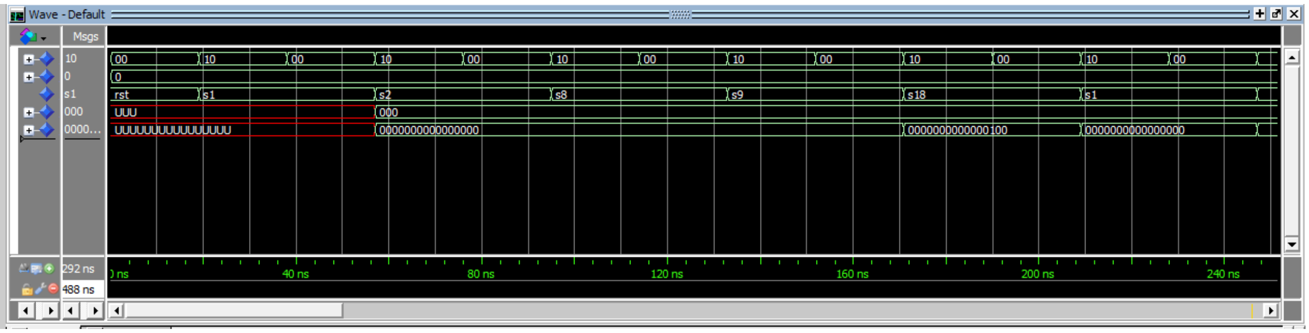


Wave - Default

	Msgs
/testbench/input_v...	10
/testbench/output_...	0
/testbench/dut_inst...	0011001100000000
Now	0.00019 ms
Cursor 1	0.000189192 ms

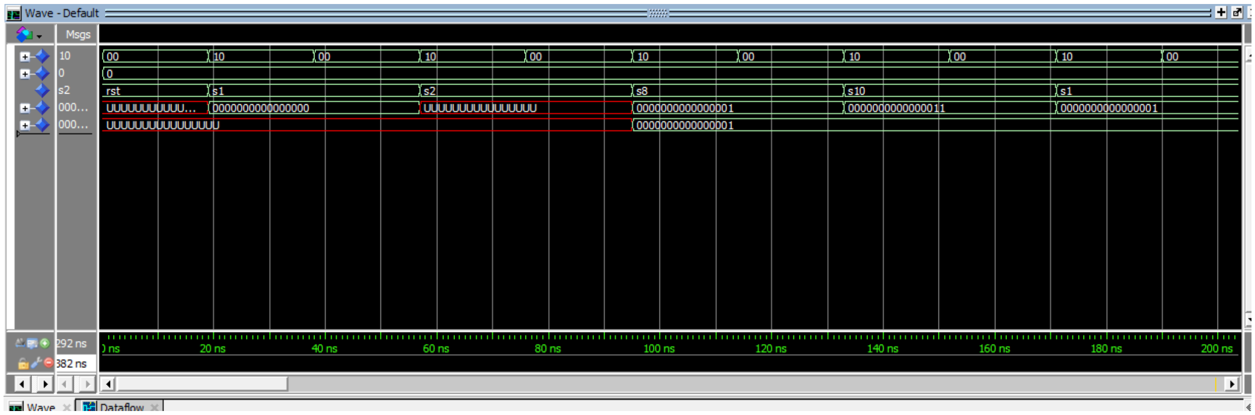
7) LW - 0100000001000001

Intermediate signals visible here are state_present, y (address of the register which we update), m5_op (data which we enter into the register).
The memory address is calculated by adding “000001” with contents of RB (“001”). The content of this memory address in stored in RA (“000”).



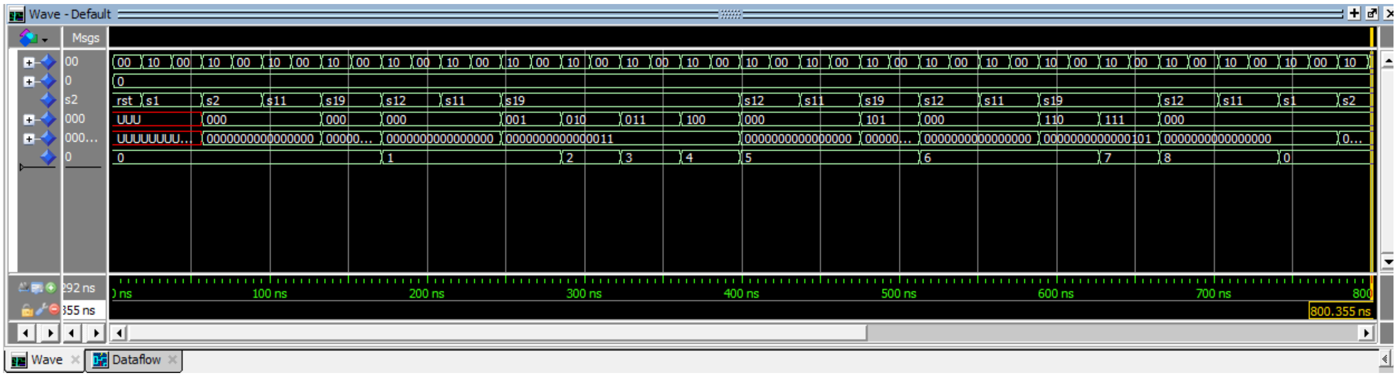
8) SW - 0101000001000001

Intermediate signals visible are state_present, m2_op (address of the memory to be updated), m3_op (data which we enter into the memory).
Value present in RA (“000”) is stored in memory. The memory address is again calculated by adding immediate (“000001”) to content of RB (“001”).



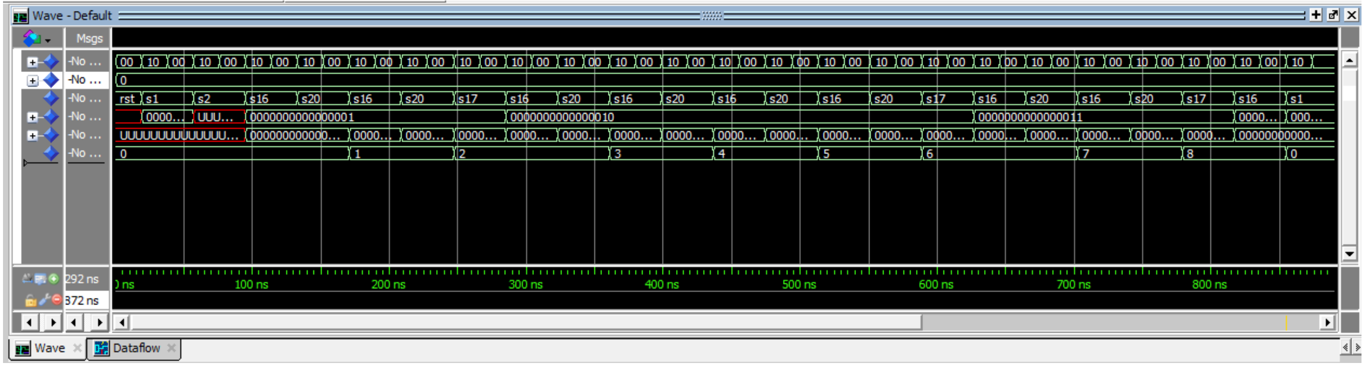
9) LM - 0110000010110001

Intermediate signals visible here are state_present, y (address of the register which we update), m5_op (data which we enter into the register), lmsm_count.
Since there are four 1’s in the immediate, only 4 registers are updated. The first memory address is present in RA (“000”) and data from consecutive memory addresses are stored in the register whose immediate has a 1.



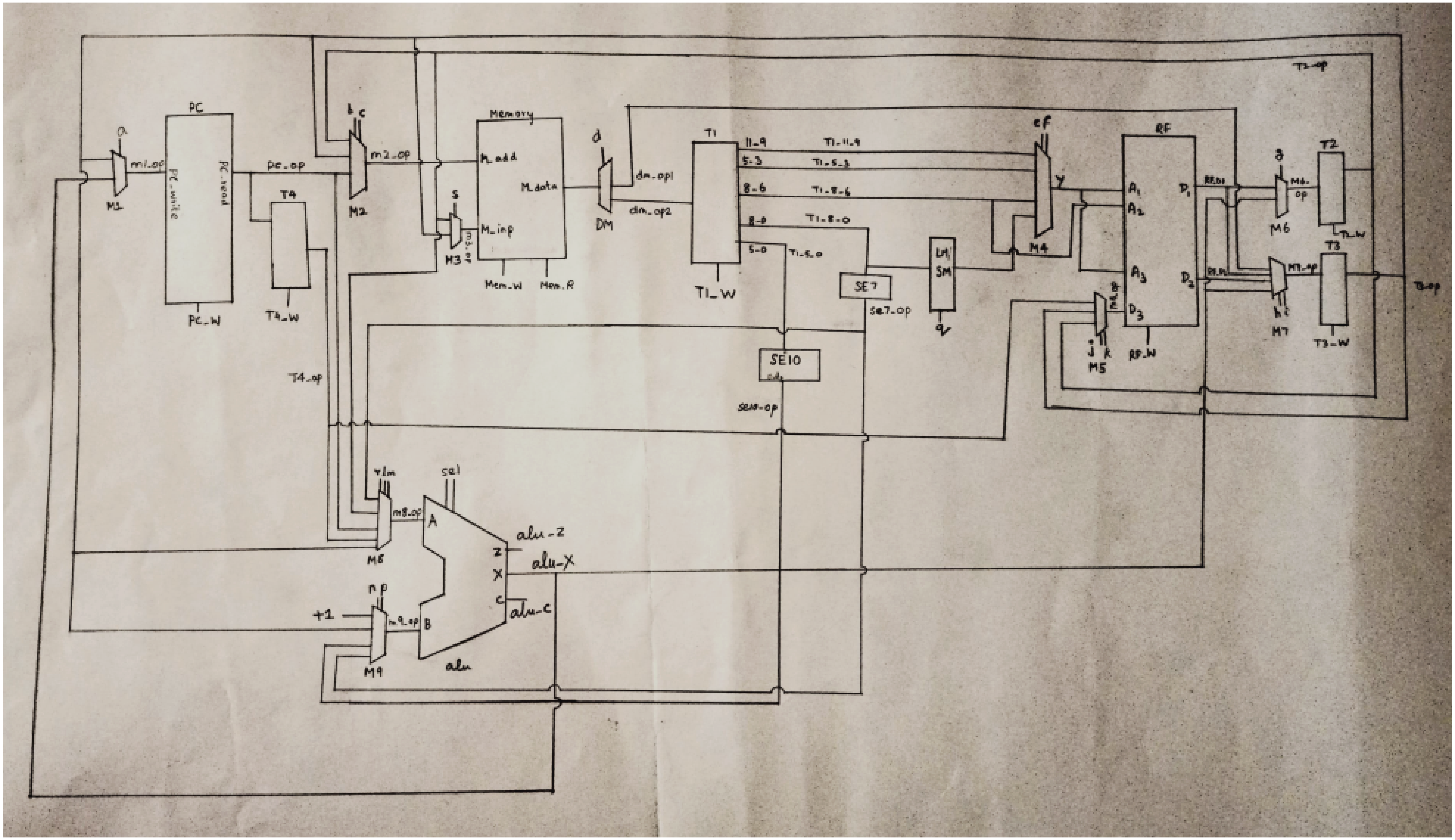
10) M - 0111000010100010

Intermediate signals visible are state_present, m2_op (address of the memory to be updated), m3_op (data which we enter into the memory), lmsm_count.
Since there are only 3 1’s in the immediate, only data from these particular registers are stored in memory. The first memory address is present in RA (“000”) and data from the next register is stored in consecutive memory address.



HARDWARE DESIGN - COMPONENTS AND CONNECTIONS

(Components, signals, connections are all marked in this image)



a
M1
m1_op
PC
PC_write
PC_read
PC_W
pc_op
T4
T4_W
T4_op
b c
M2
m2_op
s
M3
m3_op
Memory
M_add
M_data
M_inp
Mem_W
Mem_R
d
DM
dm_op1
dm_op2
T1
T1_W
11-9
5-3
8-6
8-0
5-0
T1_11_9
T1_5_3
T1_8_6
T1_8_0
T1_5_0
SE7
se7_op
SE10
se10_op
LM/SM
q
e f
M4
y
RF
A1
A2
A3
D1
D2
D3
RF_W
RFD1
RFD2
g
M6
M6_op
T2
T2_W
T2_op
h i
M7
M7_op
T3
T3_W
T3_op
j k
M5
r l m
M8
m8_op
sel
A
B
Z
X
C
alu
alu_z
alu_X
alu_c
n p
+1
M9
m9_op