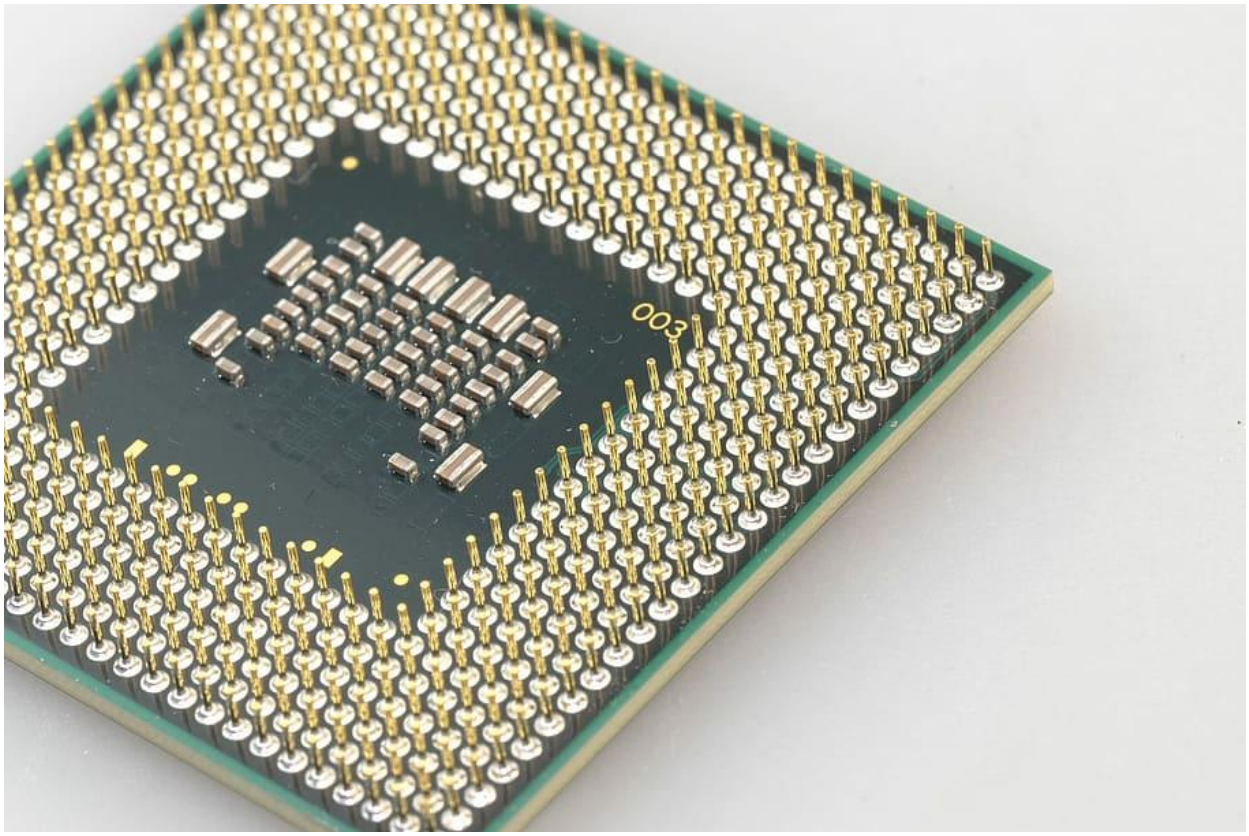


IIT Bombay CPU

4/12/2023

Sanjay Kumar Meena (22B3978)
Devtanu Barman(22B3904)
Ayush timan(22B1206)
Shreyash Wanjari (22B3926)

IIT Bombay CPU

Register 7 is utilized as an instruction register or program counter by the IITB-CPU, which uses 8 programmer registers to implement 15 instructions. These guidelines are implemented internally by 15 states, which are covered in more detail in the section that follows.

States

2.2 ADD and NAND State

1st State Read from memory and increment PC State

2nd State Read R type instruction State

3rd State Perform ALU operations

4th State Update programmer registers

5th State Read ADI type instructions

6th State Read LW, SW type instructions

7th State `Read from memory

8th State Store into memory

9th State Read LHI type instruction

10th State Branch to PC+ Immediate

11th State Store PC in register A

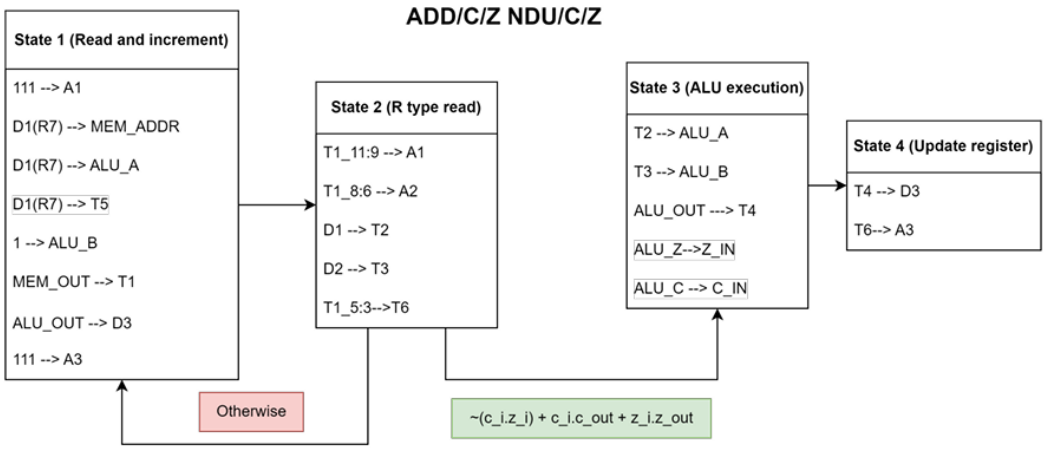
12th State Branch to register B

13th State Load from memory to register

14th State Update input to priority encoder

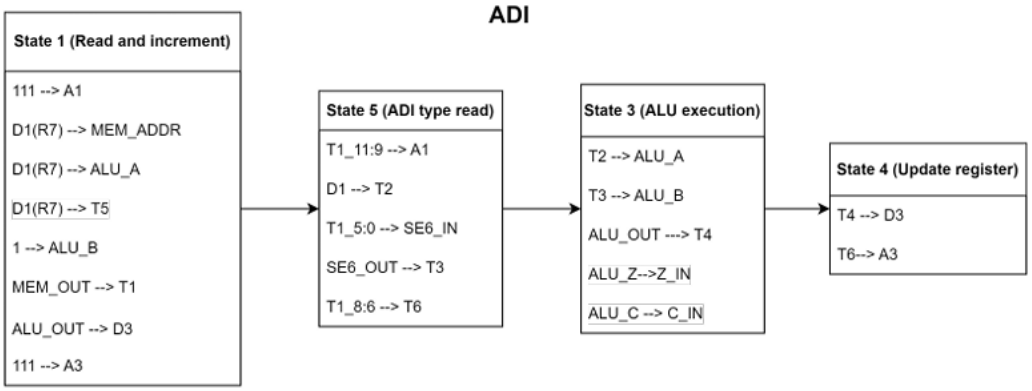
15th State Load from register to memory

AND and NAND



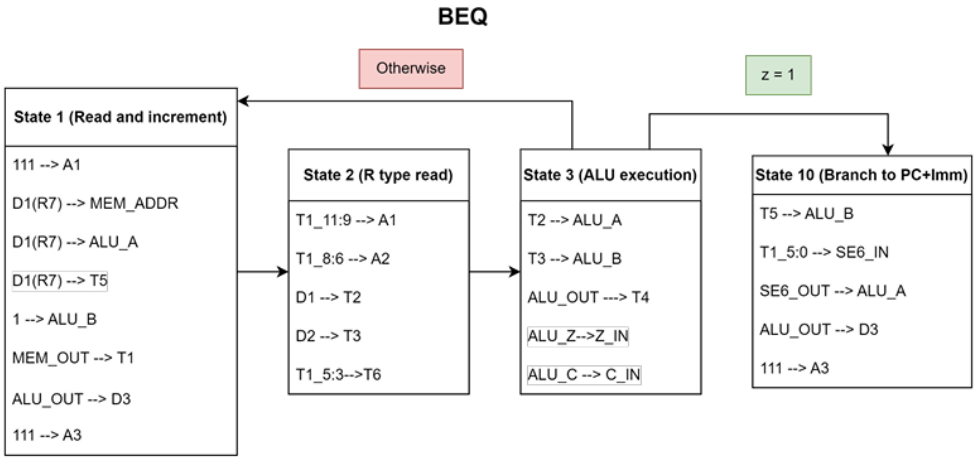
State flow

ADI



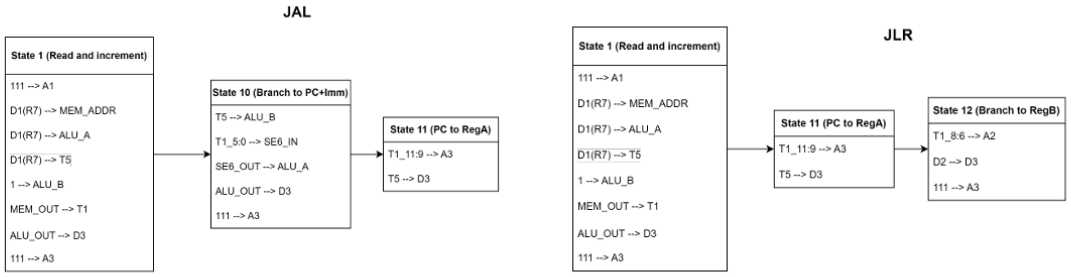
State Flow

BEQ



State Flow

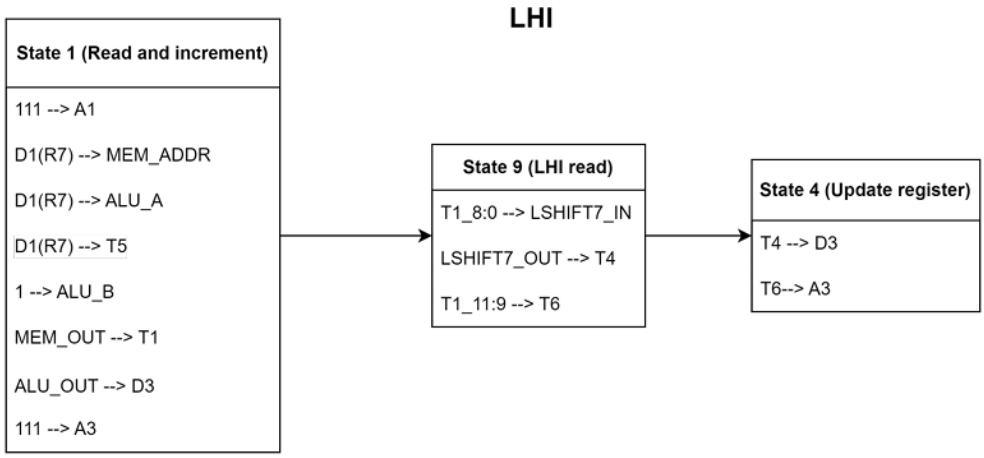
JAL & JLR



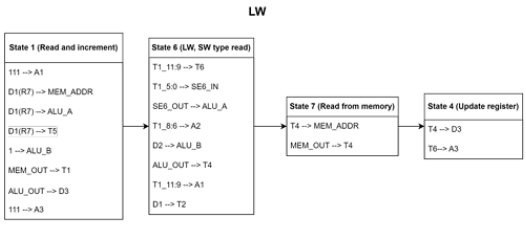
(a) JAL instruction state-flow

(b) JLR instruction state-flow

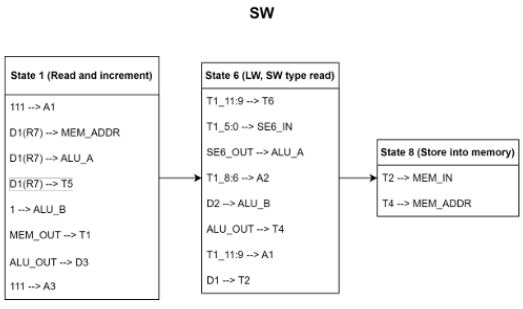
LHI



LW & SW

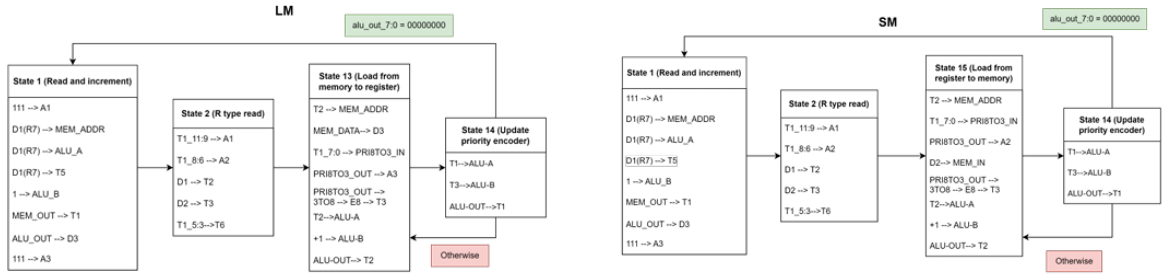


(a) LW instruction state-flow



(b) SW instruction state-flow

LM & SM



(a) LM instruction state-flow

(b) SM instruction state-flow

Three Control Signals and Transition Logic

State transition can be facilitated through behavioral modeling based on the values of:

1.Op Code (the four most important instruction bits)

2. C and Z priority bits (the two least important instruction bits)

3. C and Z flag (contents of flag register)

The ensuing control signals required definition:

Memory Writable (MWR)

MemoryReadable (MRD)

Programmer Register Writeable (PRWR)

T1WR and T2WR are the temporary registers that are writeable.

T3WR and T4WR are temporary registers that are writeable.

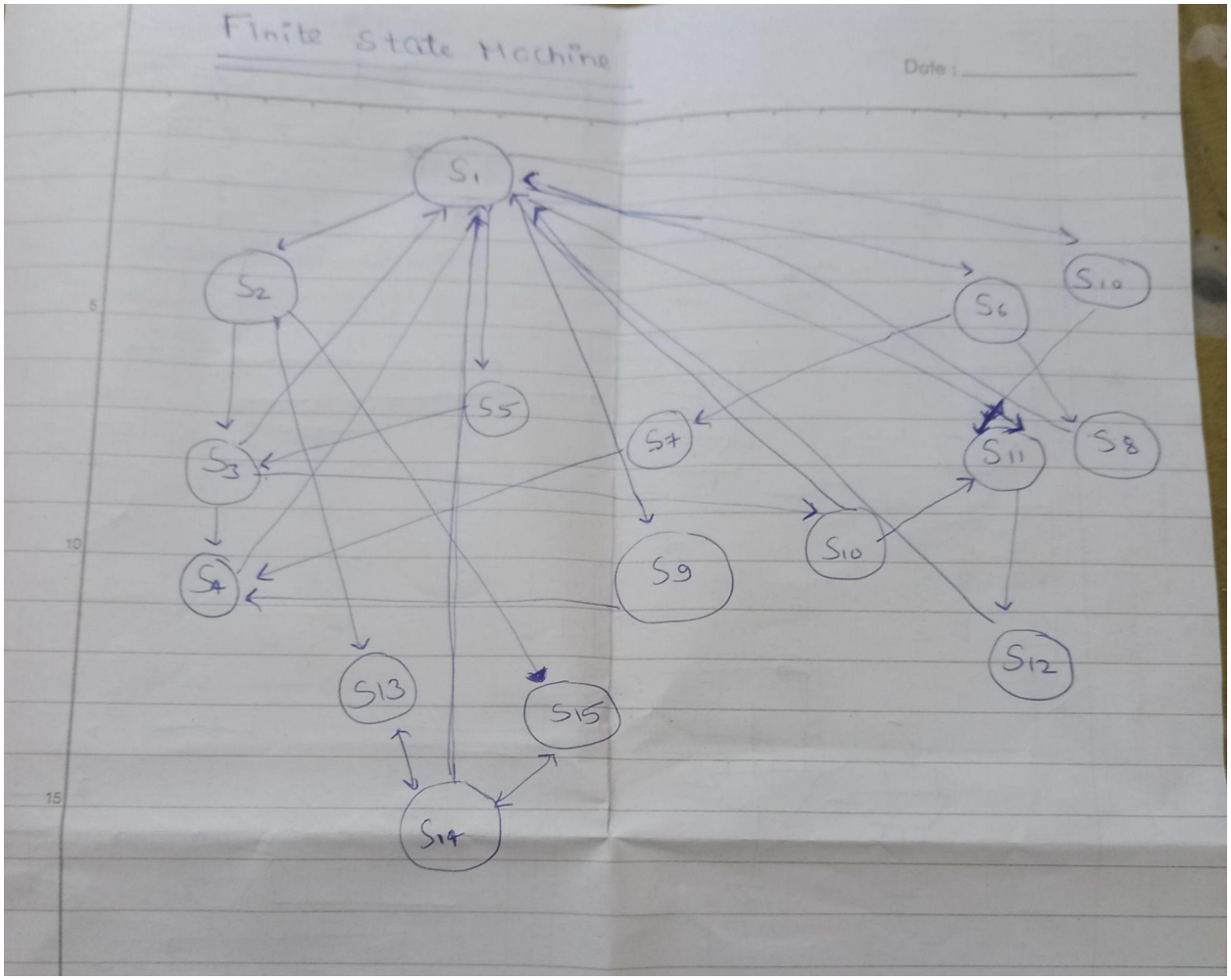
T5WR and T6WR are the temporary registers that are write-enabled.

Writeable flags: Zflag(ZEN) and Cflag(CEN)

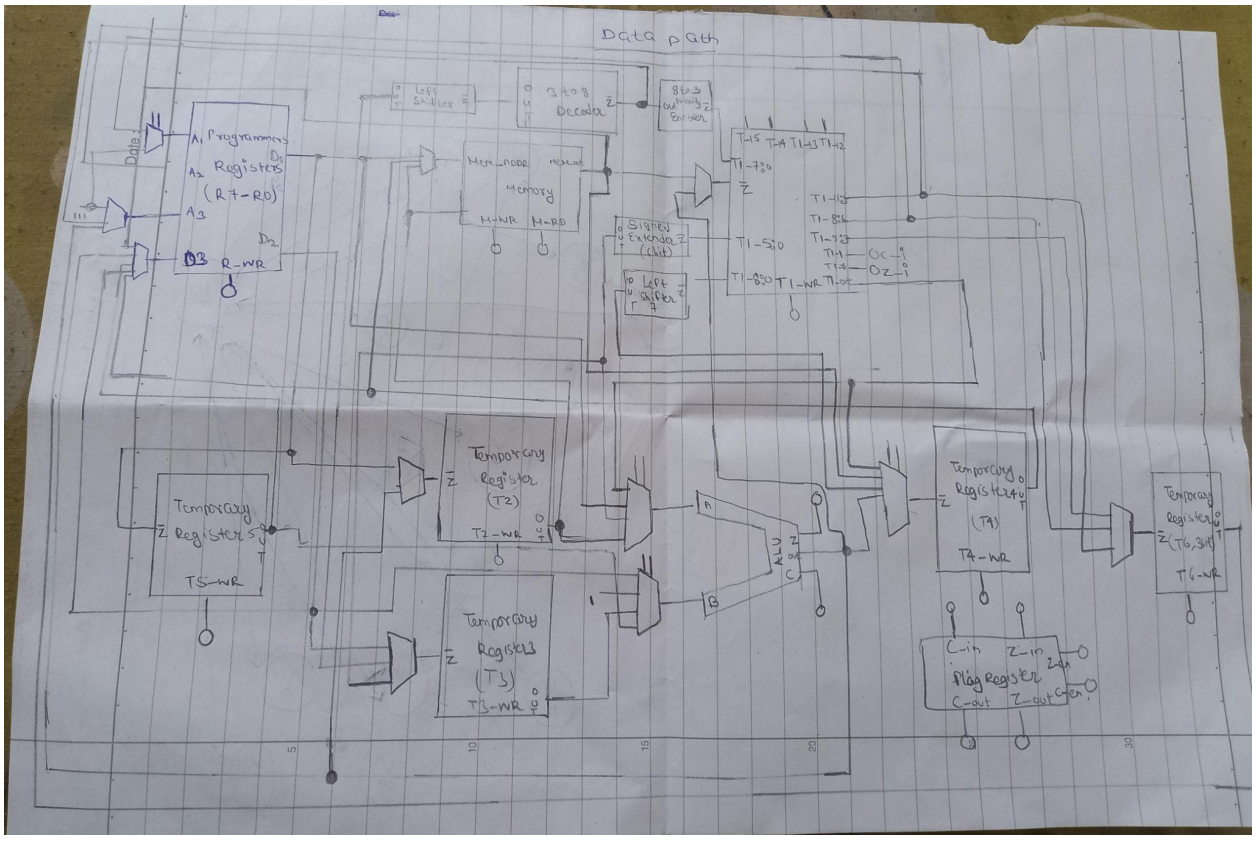
ALUSELECT0 and ALUSELECT 1 as the ALUselectsignal

Present State	Next State	Condition
State 1	State 2	!(op[3]) * !(op[2]) * !(op[0]) + (op[2] * op[1]) + (op[3] * op[2])
	State 5	!(op[3]) * !(op[2]) * !(op[1]) * op[0]
	State 6	!(op[3]) * op[2] * !(op[1])
	State 9	!(op[3]) * !(op[2]) * op[1] * op[0]
	State 10	op[3] * !(op[2]) * !(op[1]) * !(op[0])
	State 11	op[3] * !(op[2]) * !(op[1]) * op[0]
State 2	State 1	!(op[3]) * !(op[2]) * !(op[0])) * !(!(c_i *z_i) + c_i * c_f + z_i * z_f))
	State 3	(!(op[3]) * !(op[2]) * !(op[0])) * (!(c_i *z_i) + c_i * c_f + z_i * z_f)) + (op[3] * op[2])
	State 13	!(op[3]) * op[2] * op[1] * !(op[0])
	State 15	!(op[3]) * op[2] * op[1] * op[0]
State 3	State 1	op[3] * op[2] * !(z_f)
	State 4	!(op[3]) * !(op[2]) * !(op[1] * op[0])
	State 10	op[3] * op[2] * z_f
State 4	State 1	Unconditional
State 5	State 3	Unconditional
State 6	State 7	!(op[3]) * op[2] * !(op[1]) * !(op[0])
	State 8	!(op[3]) * op[2] * !(op[1]) * op[0]
State 7	State 4	Unconditional
State 8	State 1	Unconditional
State 9	State 4	Unconditional
State 10	State 1	op[3] * op[2]
	State 11	op[3] * !(op[2]) * !(op[1]) * !(op[0])
State 11	State 1	op[3] * !(op[2]) * !(op[1]) * !(op[0])
	State 12	op[3] * !(op[2]) * !(op[1]) * op[0]
State 12	State 1	Unconditional
State 13	State 14	Unconditional
State 14	State 1	T1_7:0 == 00000000
	State 13	(T1_7:0 != 00000000) * (!(op[3]) * op[2] * op[1] * !(op[0]))
	State 15	(T1_7:0 != 00000000) * (!(op[3]) * op[2] * op[1] * op[0])
State 15	State 14	Unconditional

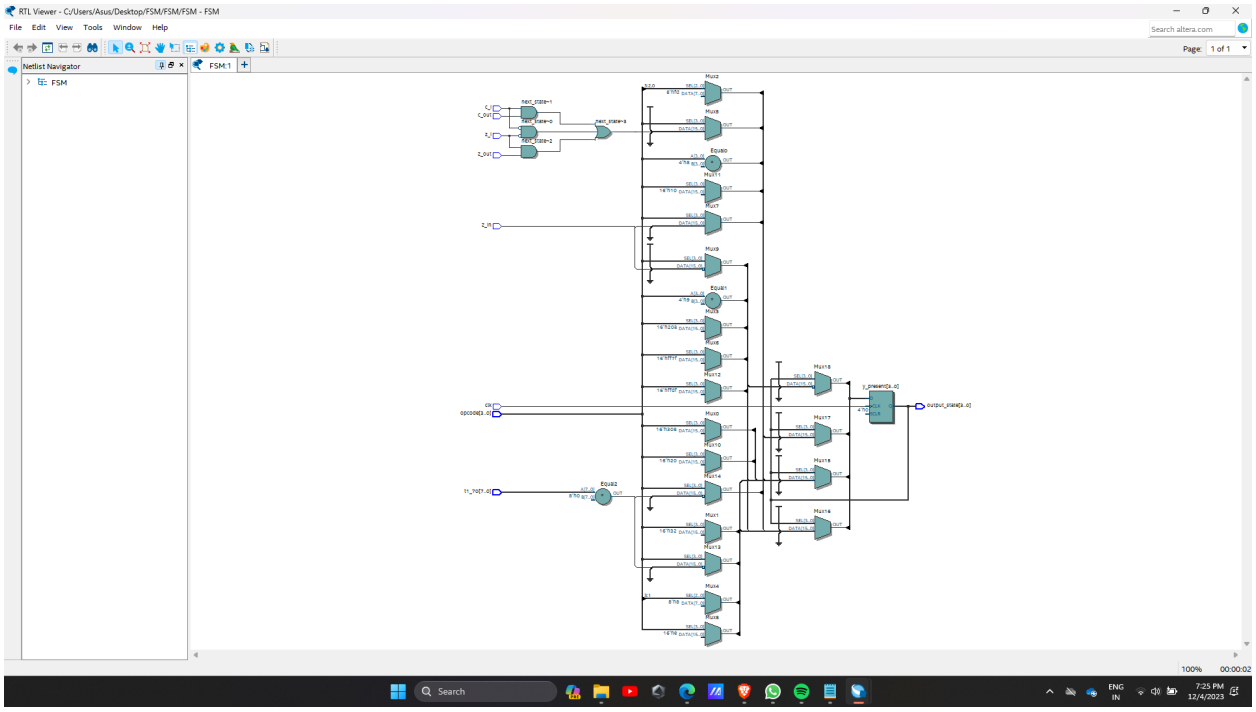
FINITE STATE MACHINE



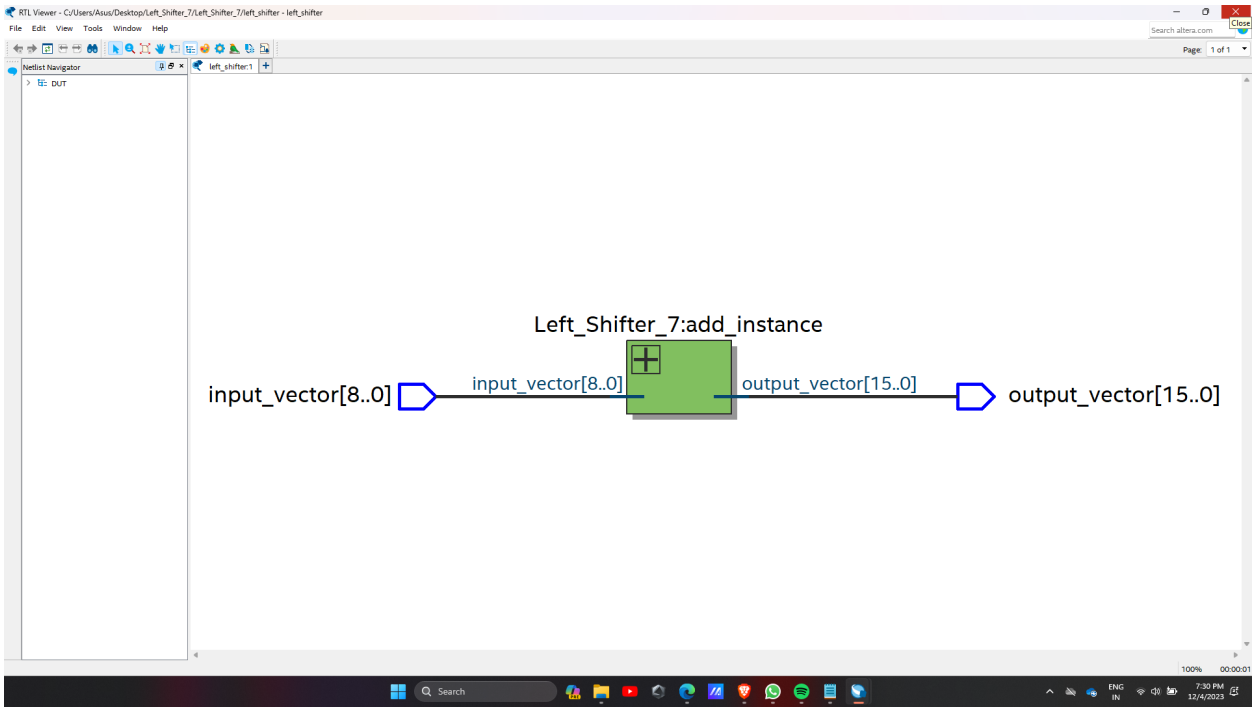
CPU DIAGRAM


Data path
Programmers Registers (R7-R0)
R_WR
Memory
M_WR
M_RD
3 to 8 Decoder
Left Shifter
Signed Extender (8bit)
Left Shifter
T1_WR
Temporary Register (T2)
T2_WR
Temporary Register (T3)
T3_WR
Temporary Register (T4)
T4_WR
Temporary Register (T5)
T5_WR
Temporary Register (T6)
T6_WR
ALU
Flag Register
C_in
Z_in
C_out
Z_out

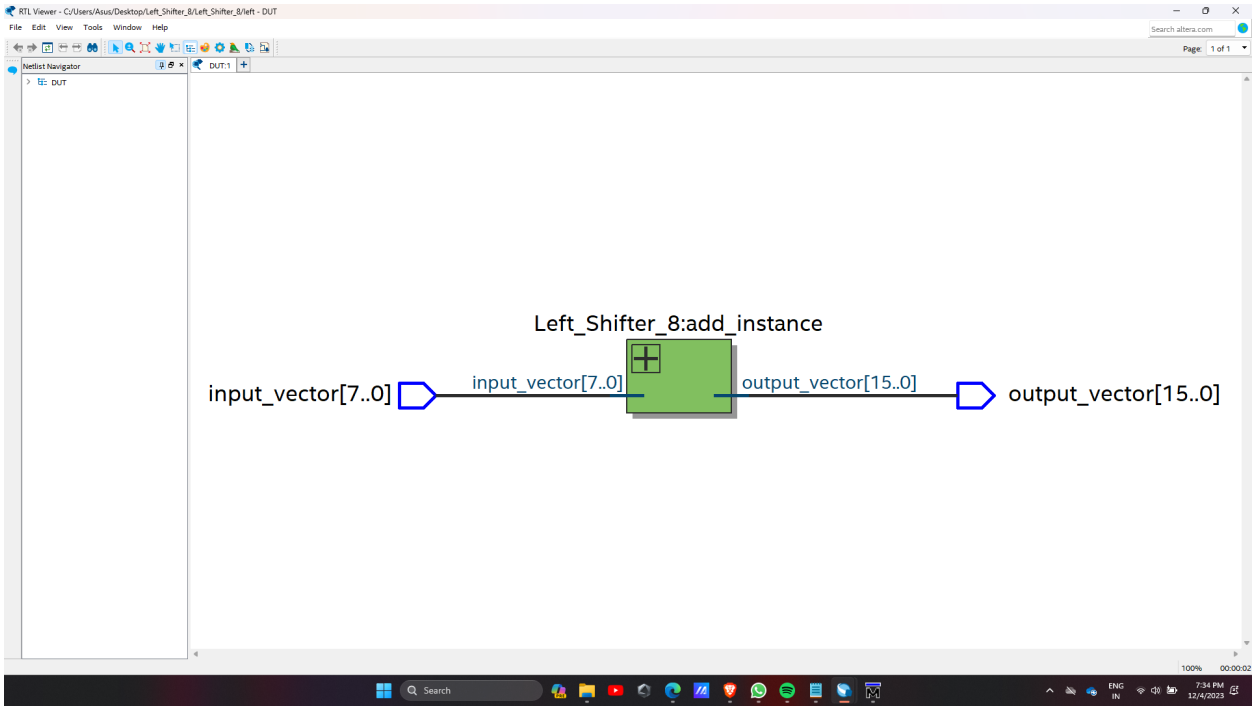
FSM:1



LEFT SHIFTER 7 : 1

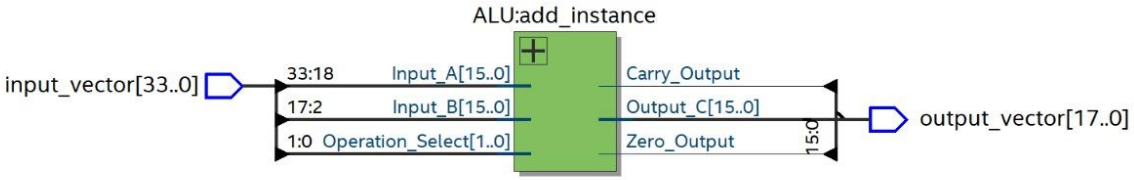


LEFT SHIFTER 8

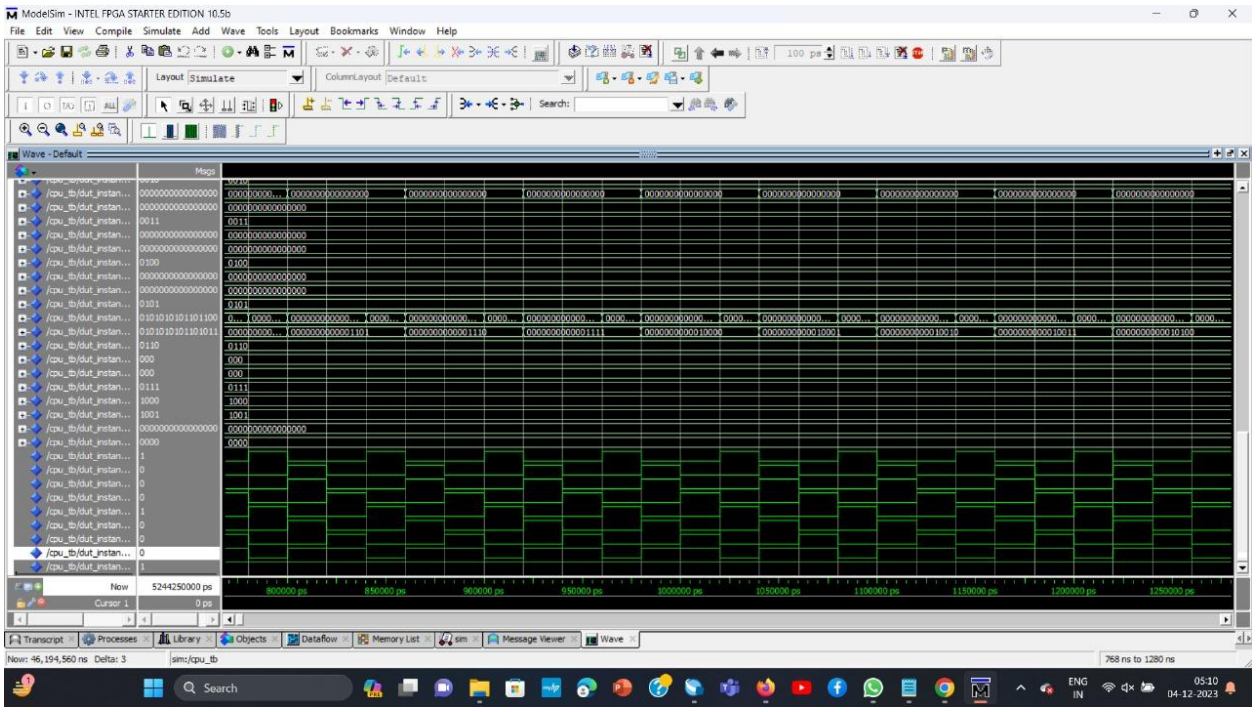


Currently we haven't used test vectors to check for the outputs of CPU and shifters

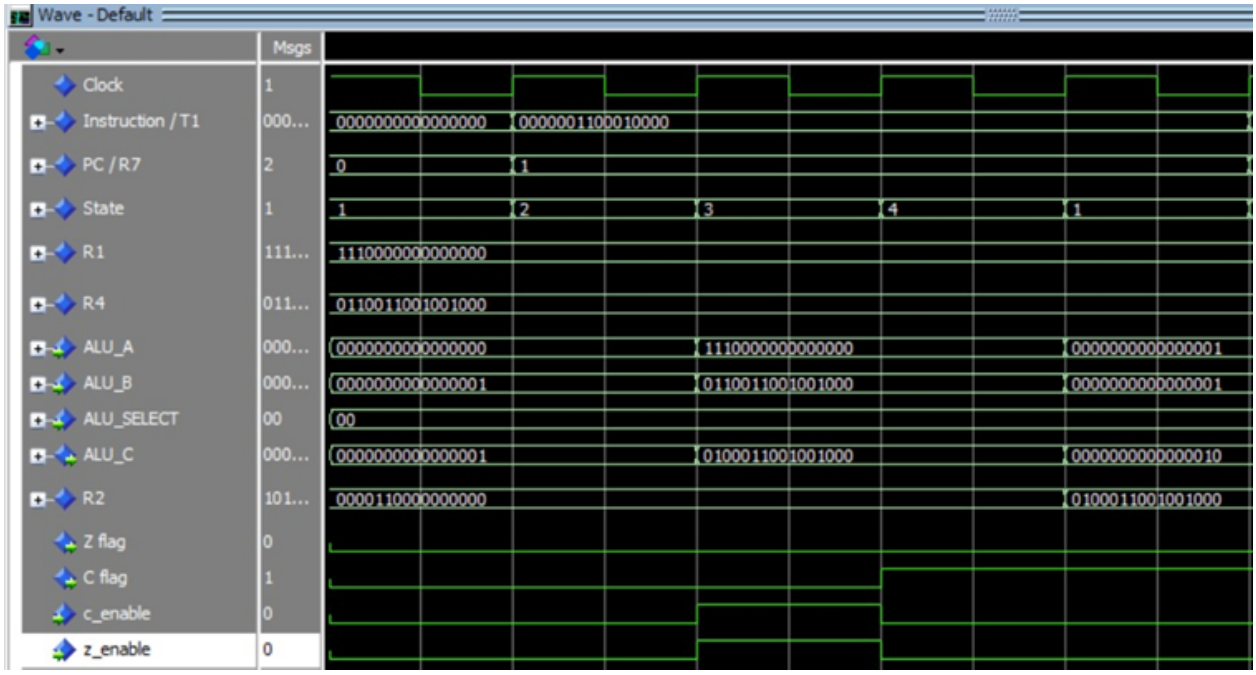
FOR ALU WE USED TEST INPUTS



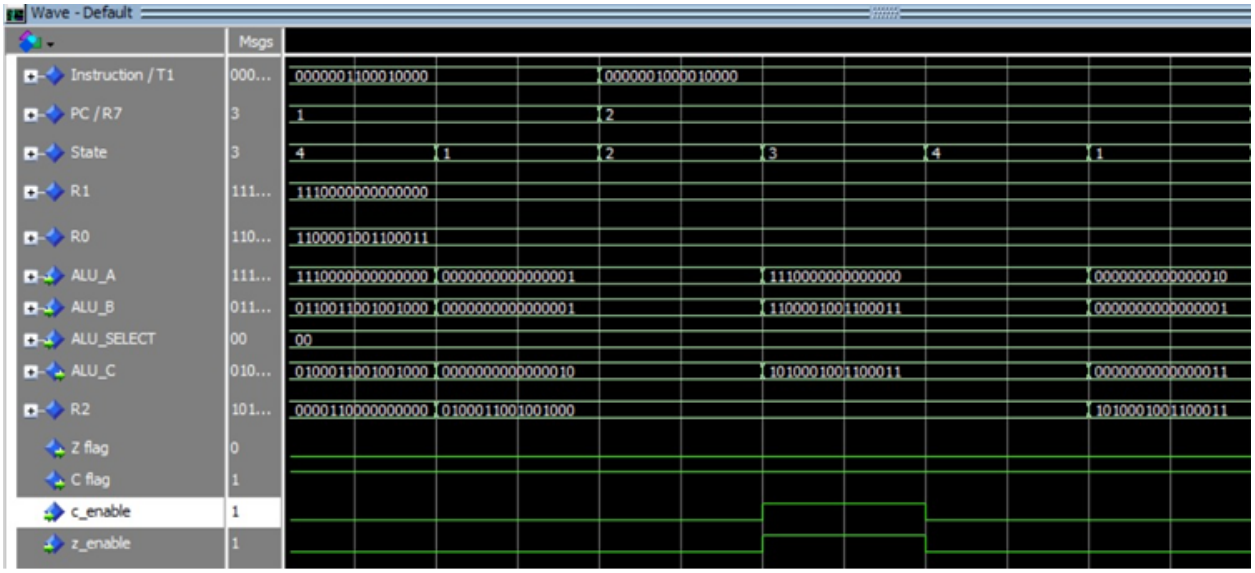
CPU WAVEFORM


ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Wave - Default
Msgs
Now
Cursor 1
Transcript
Processes
Library
Objects
Dataflow
Memory List
sim
Message Viewer
Wave
Search
ENG
IN
05:10
04-12-2023

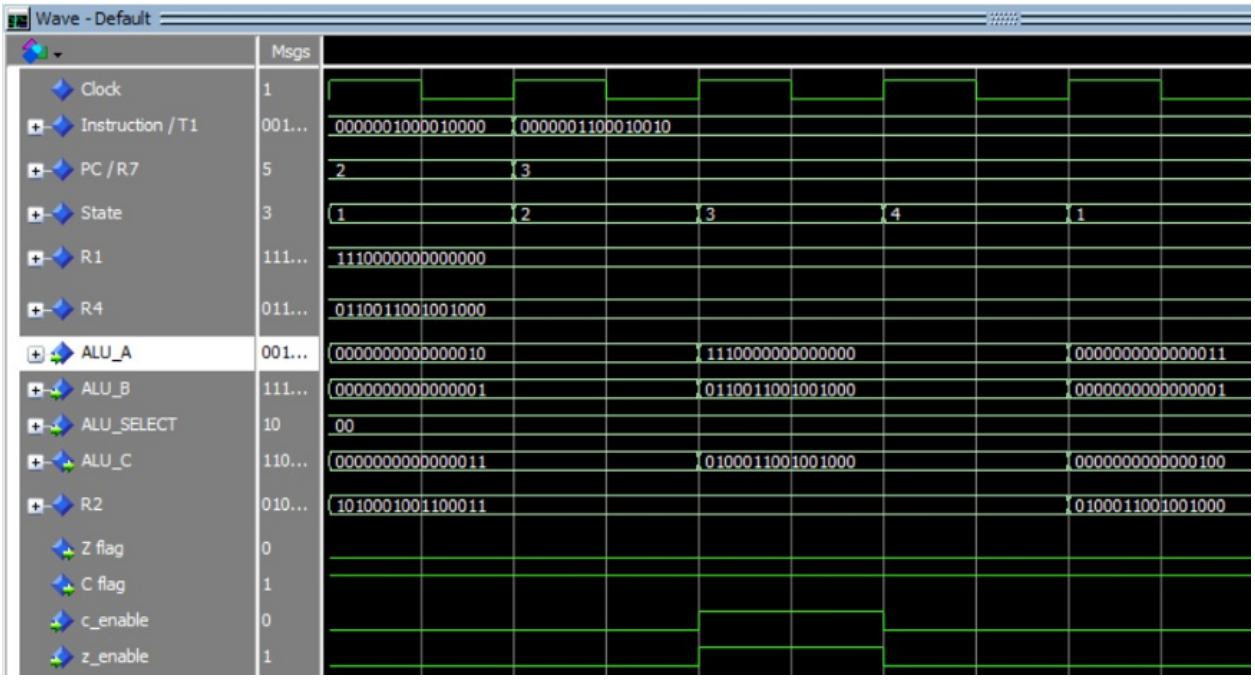
ADD R1 R4 R2


Wave - Default
Msgs
Clock
Instruction / T1
PC / R7
State
R1
R4
ALU_A
ALU_B
ALU_SELECT
ALU_C
R2
Z flag
C flag
c_enable
z_enable

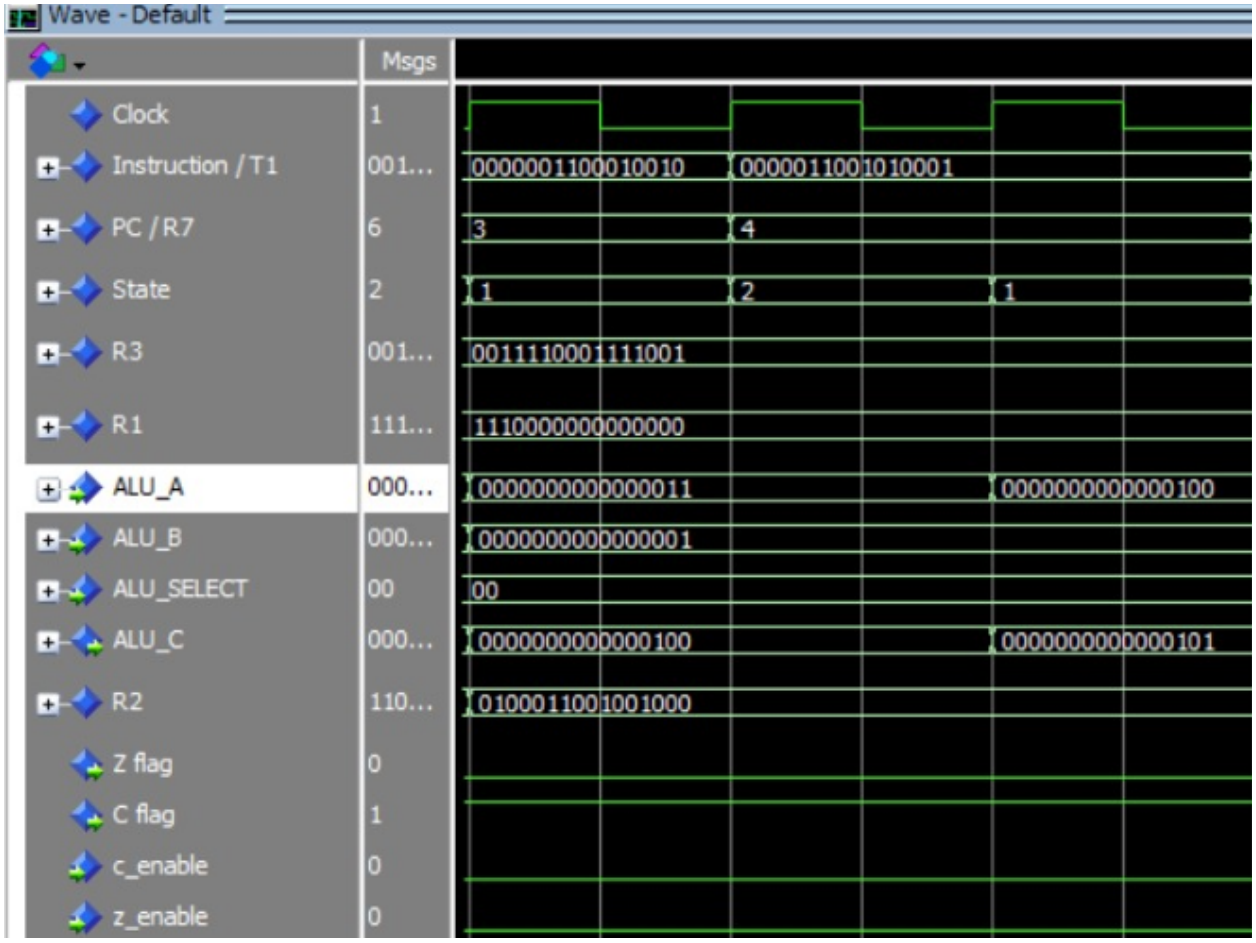
ADD R1 R0 R2



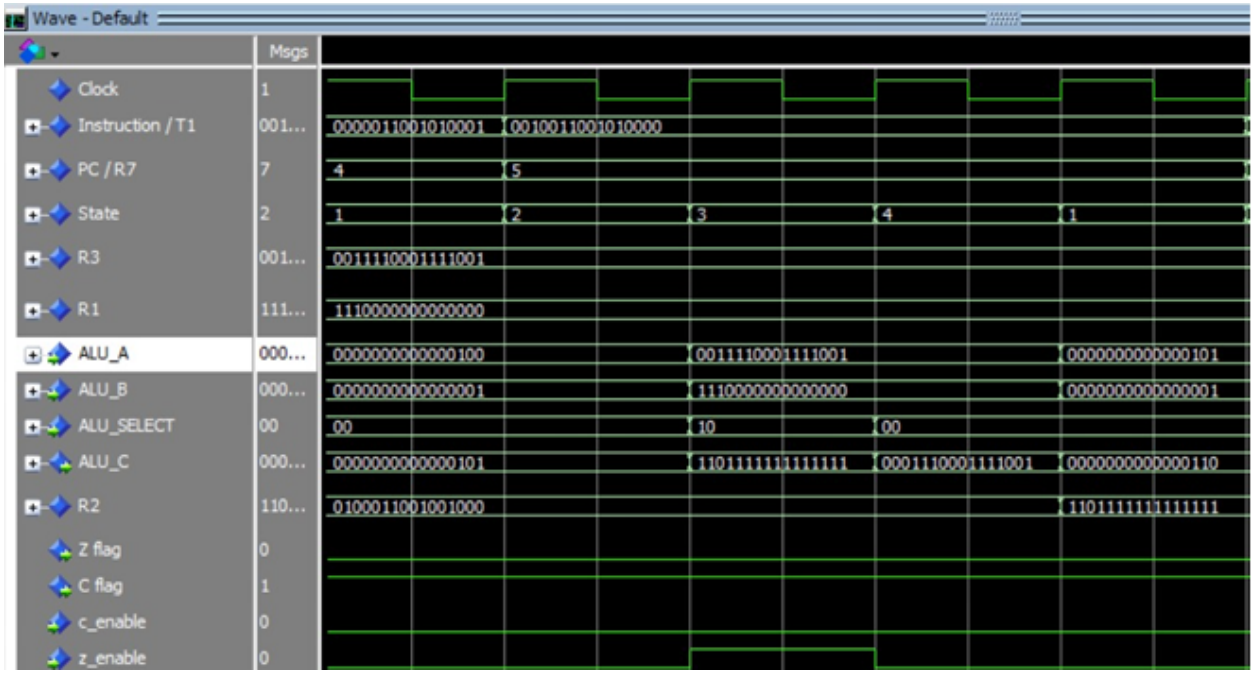
ADC R1 R4 R2



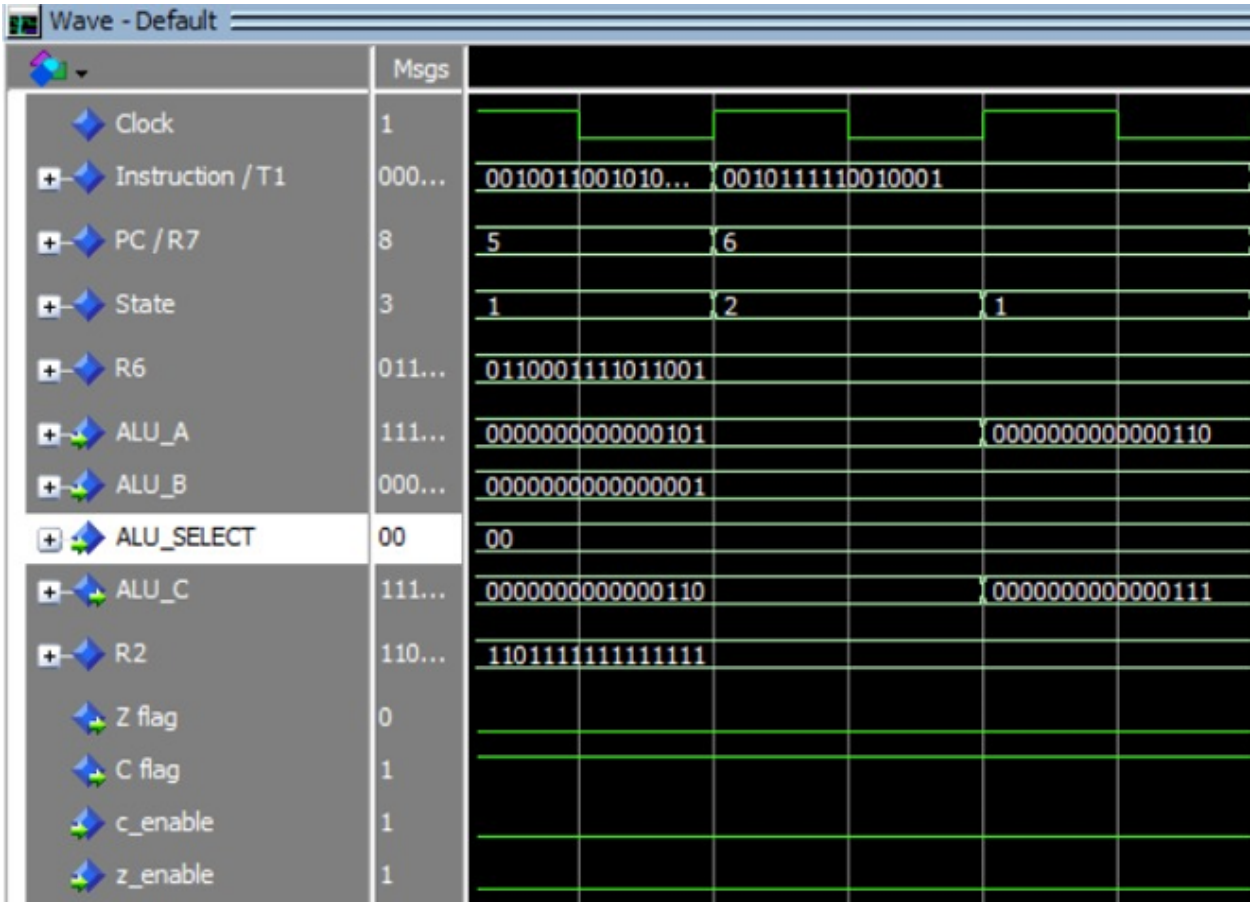
ADZ R3 R1 R2


Wave - Default
Msgs
Clock
1
Instruction / T1
001...
0000001100010010
0000011001010001
PC / R7
6
3
4
State
2
1
2
1
R3
001...
0011110001111001
R1
111...
1110000000000000
ALU_A
000...
0000000000000011
0000000000000100
ALU_B
000...
0000000000000001
ALU_SELECT
00
00
ALU_C
000...
0000000000000100
0000000000000101
R2
110...
0100011001001000
Z flag
0
C flag
1
c_enable
0
z_enable
0

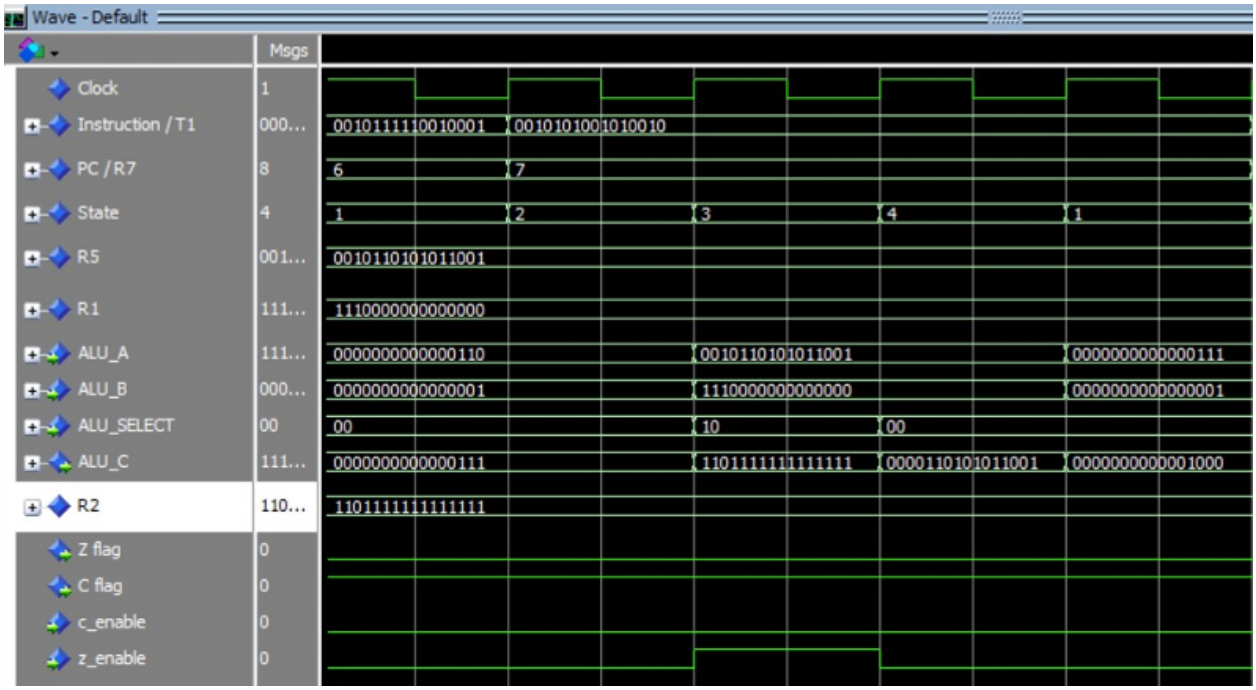
NDU R3 R1 R2


Wave - Default
Msgs
Clock
Instruction / T1
PC / R7
State
R3
R1
ALU_A
ALU_B
ALU_SELECT
ALU_C
R2
Z flag
C flag
c_enable
z_enable
0000011001010001
0010011001010000
0011110001111001
1110000000000000
0000000000000100
0000000000000001
00
0000000000000101
0100011001001000
1101111111111111
0001110001111001
0000000000000110

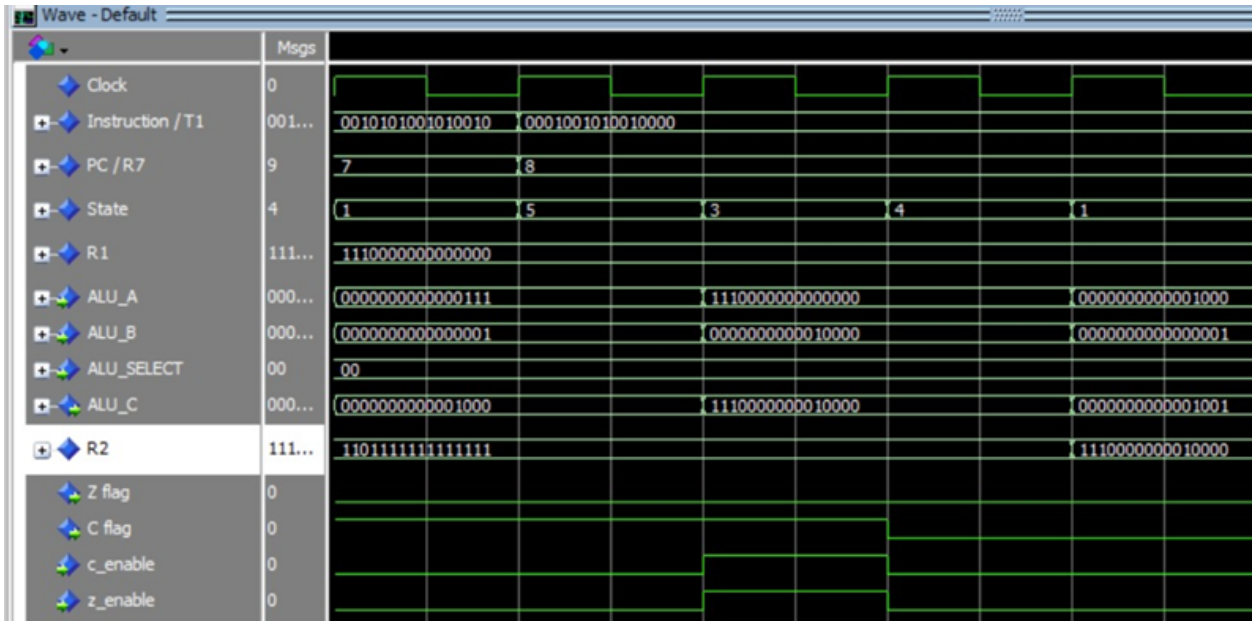
NDZ R7 R6 R2


Wave - Default
Msgs
Clock
1
Instruction / T1
000...
0010011001010...
0010111110010001
PC / R7
8
5
6
State
3
1
2
1
R6
011...
0110001111011001
ALU_A
111...
0000000000000101
0000000000000110
ALU_B
000...
0000000000000001
ALU_SELECT
00
00
ALU_C
111...
0000000000000110
0000000000000111
R2
110...
1101111111111111
Z flag
0
C flag
1
c_enable
1
z_enable
1

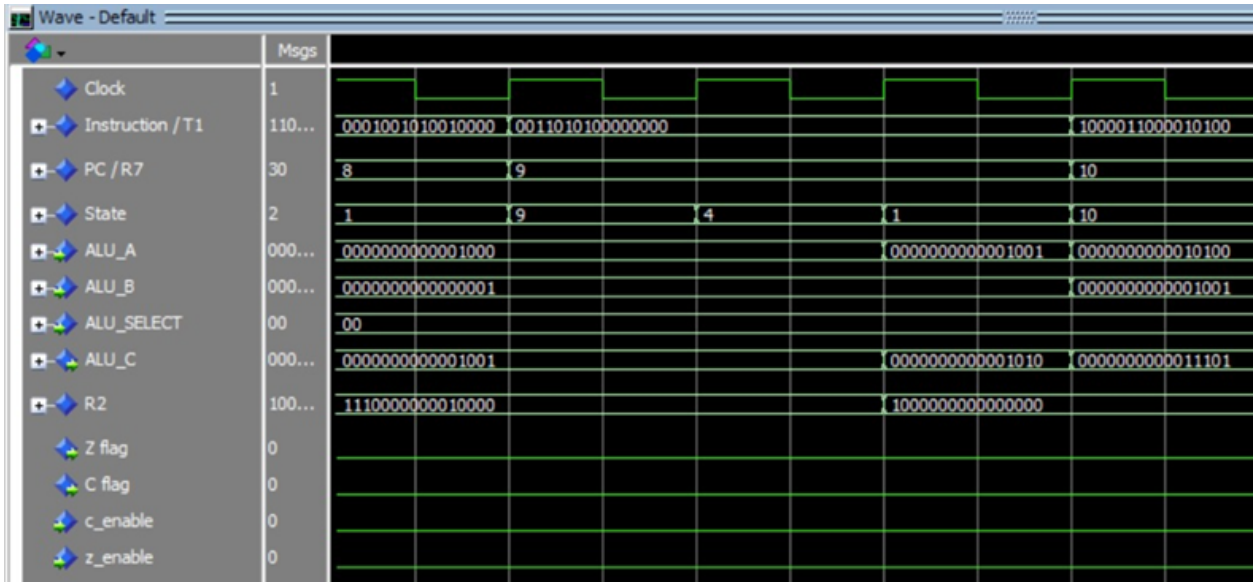
NDC R5 R1 R2



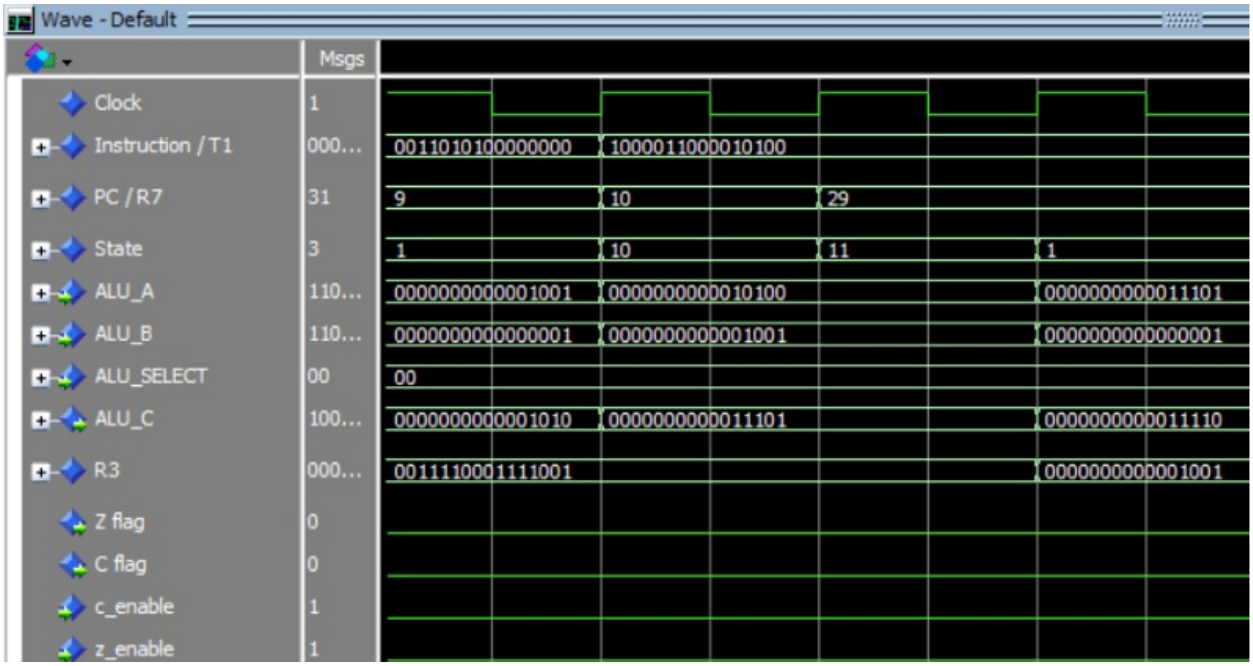
ADI R1 R2 010000



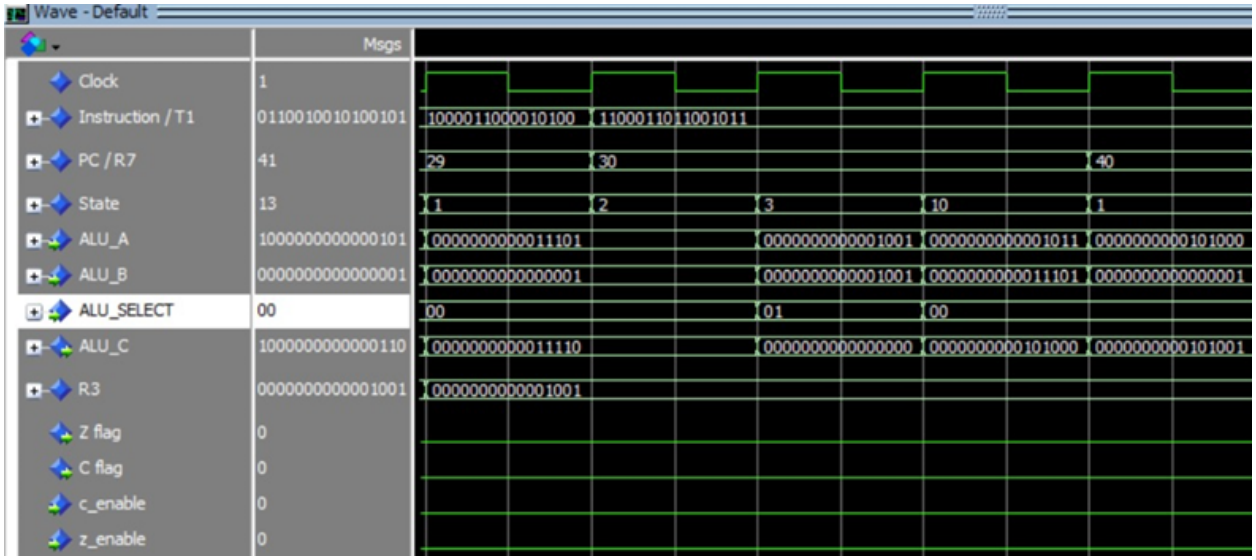
LHI R2 100000000



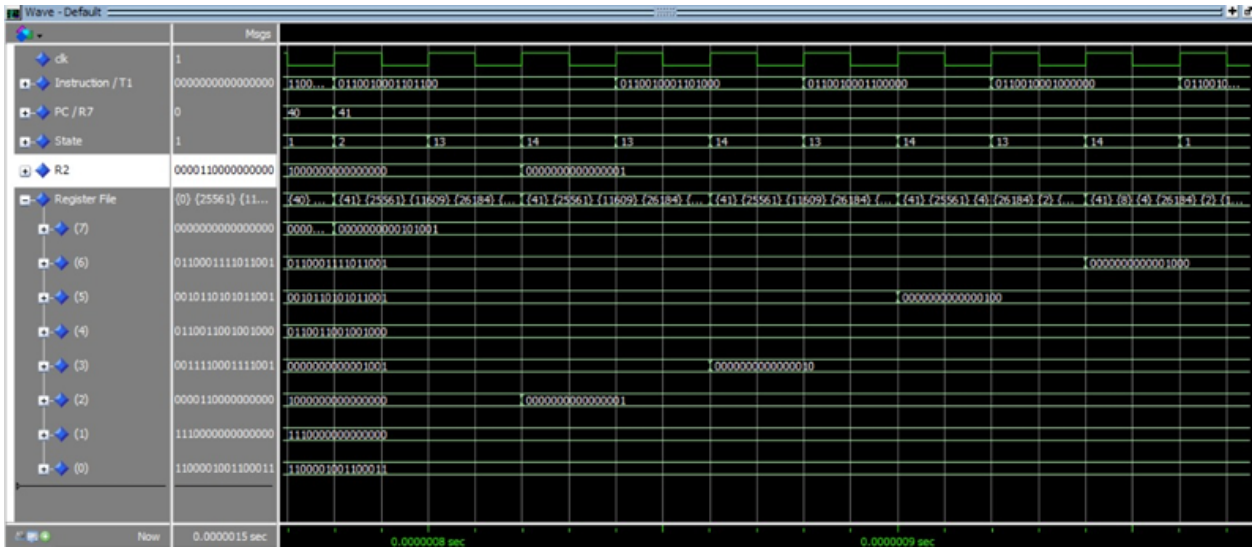
JAL R3 000010100



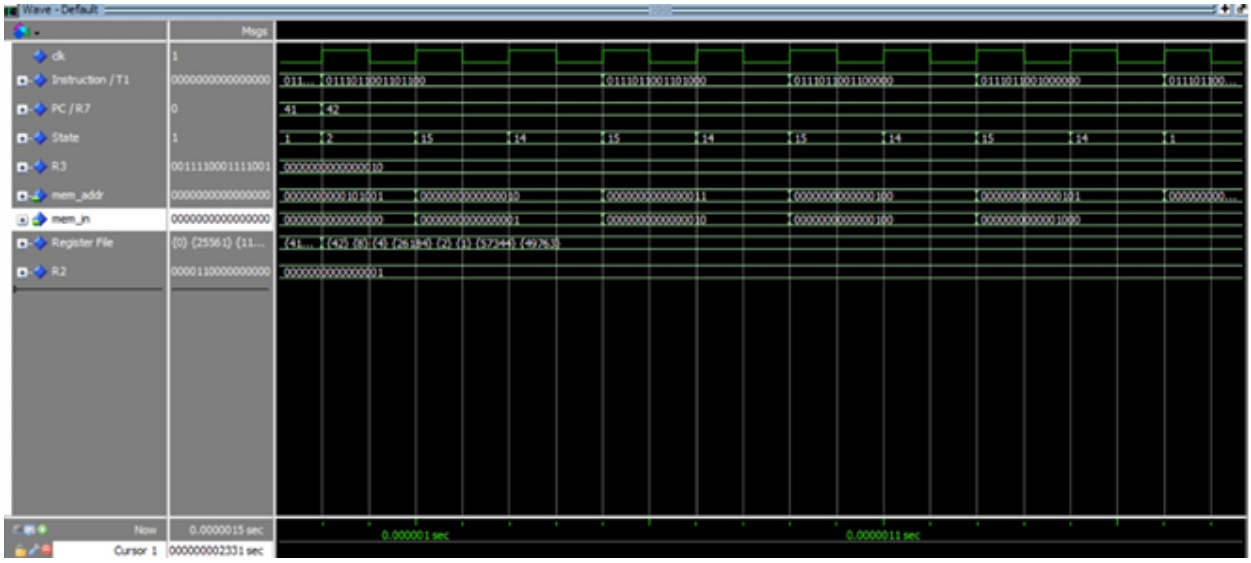
BEQ R3 R3 00000101101



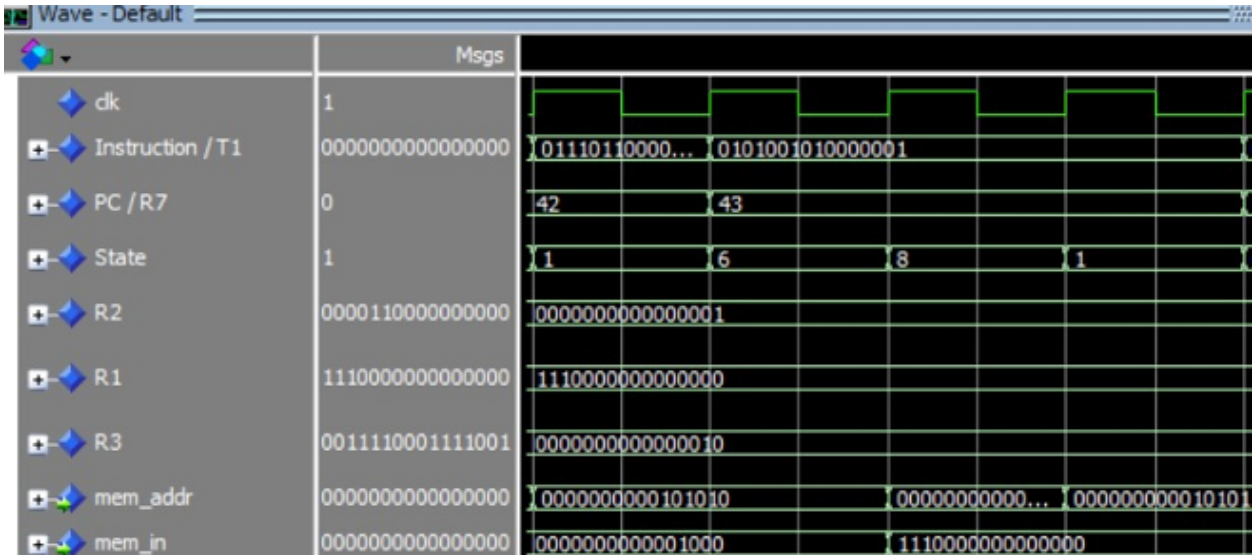
LM R2 010101100



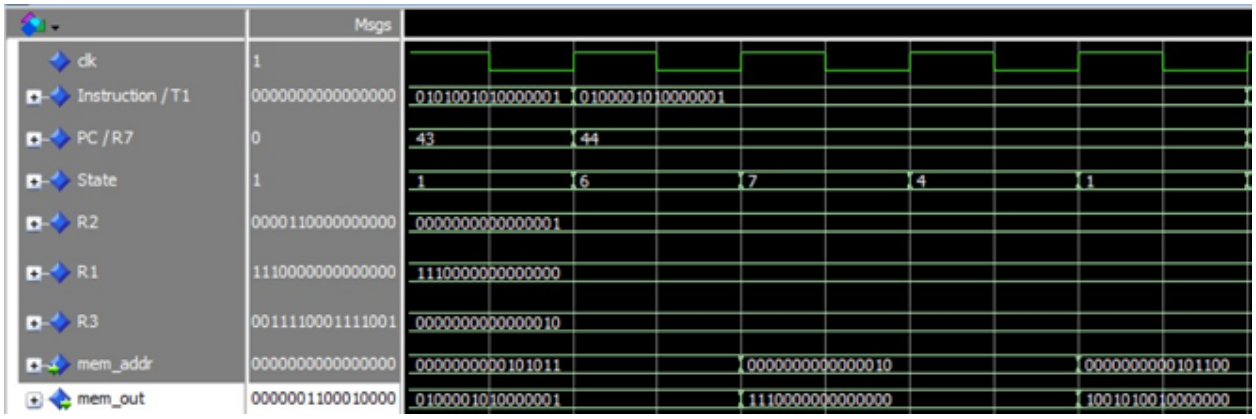
SM R3 010101100



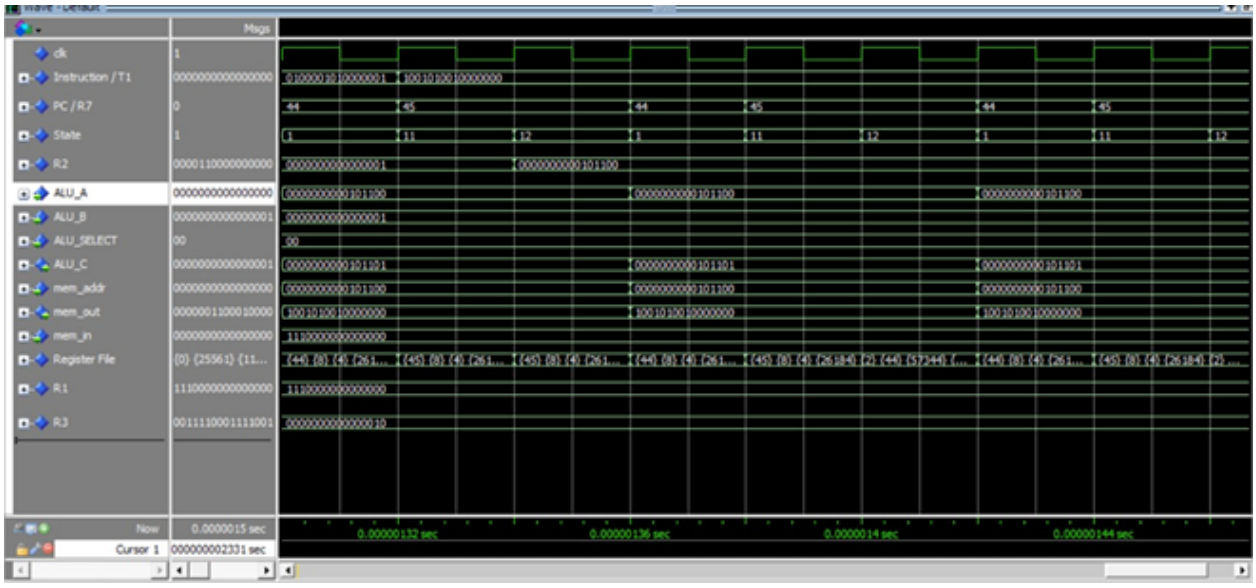
SW R1 R2 000001



LW R1 R2 000001



JLR R2 R2 000000



WORK DISTRIBUTION TEAM CODE -40

Devtanu Barman - Designed the left shifter, FSM and CPU main code and debugged the main code .

Shreyash Dnyanesh Wanjari - Significantly contributed to the creation and analysis of components like ALU , Signed Extender etc.

Sanjay Kumar Meena -Major contribution in writing the code for Data_Path and Data_Flow ,temporary register (3 bit & 16 bit), programmer_register and memory . contributed in debugging the main code with Devtanu.

Ayush Timan - Contributed in making state diagram and writing the code for 3to8_Decoder .