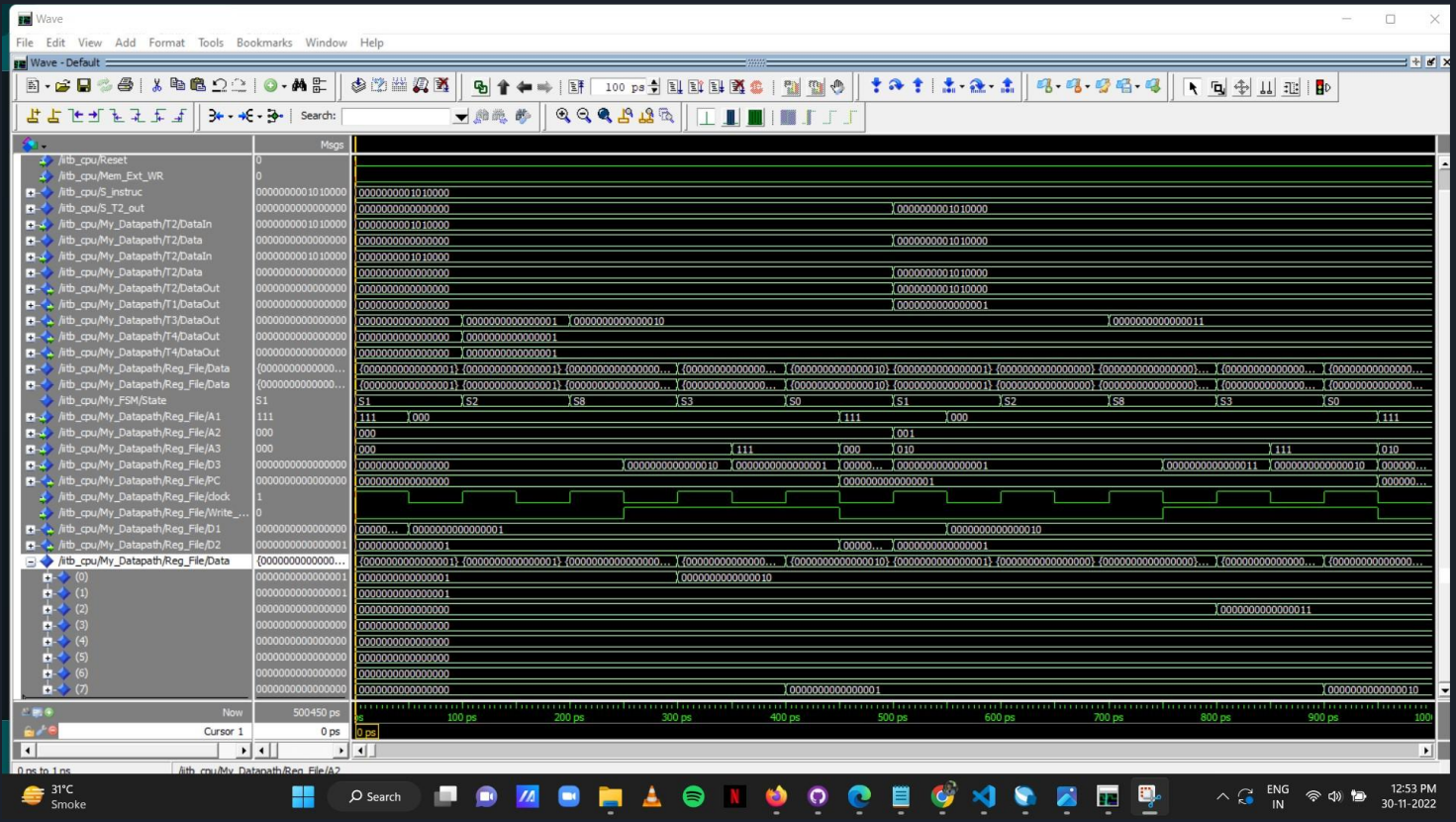
RTL Simulations

-Harshraj Chaudhari(210040060)
-Sarvadnya Desai(210040138)
-Ishaan Manhar(210070033)
-Akshat Taparia(210110014)

All the images in this presentation are included in the github repository under RTL folder. For zooming in and out refer the same.

ADD Instruction



ADD Instruction

Notice the Registers T3 and T4 have initial value zero.

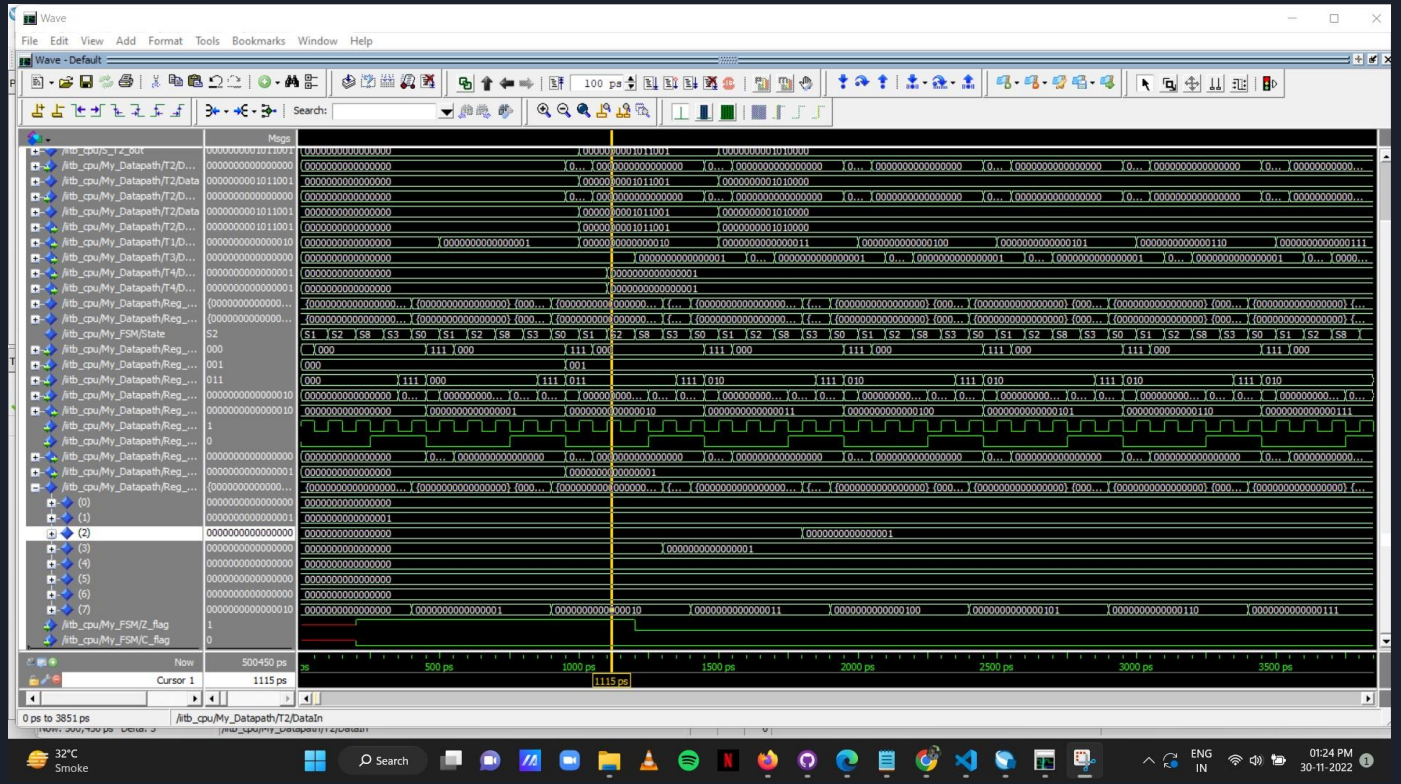
Then we see that they are fed the value 1 and 1 as a part of the instruction.

A moment later we see a binary 2 and then a binary 3 in the T3 register.

This is showing the update.

Also observe the D1 and D2 RTL simulation.

ADZ instruction



ADZ Instruction

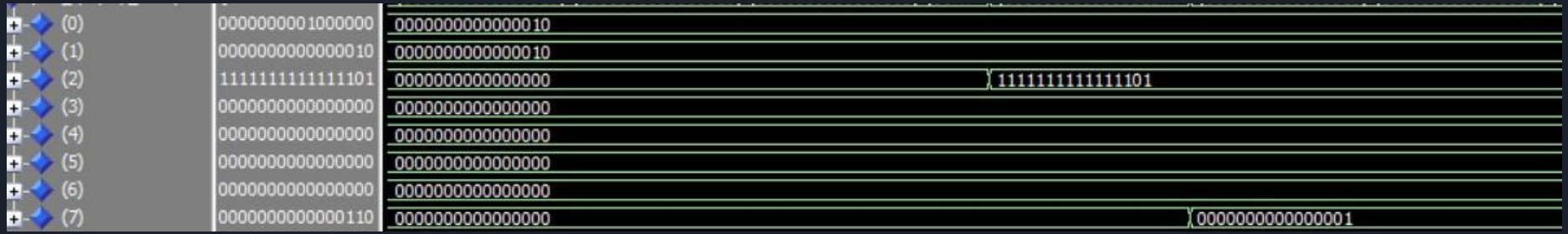
Observe that the first addition doesn't occur. This is because after setting the zero flag too the first instruction doesn't generate a zero flag.

The second instruction however after generation of the zero flag , does execute and the values of T3 and T4 are updated.

Also observe the D1 and D2 registers which fetch the data but the computation is not output for the same.

Add R1 with R2 and store in R3. (Executed successfully when zero flag is set.)

NDU Instruction

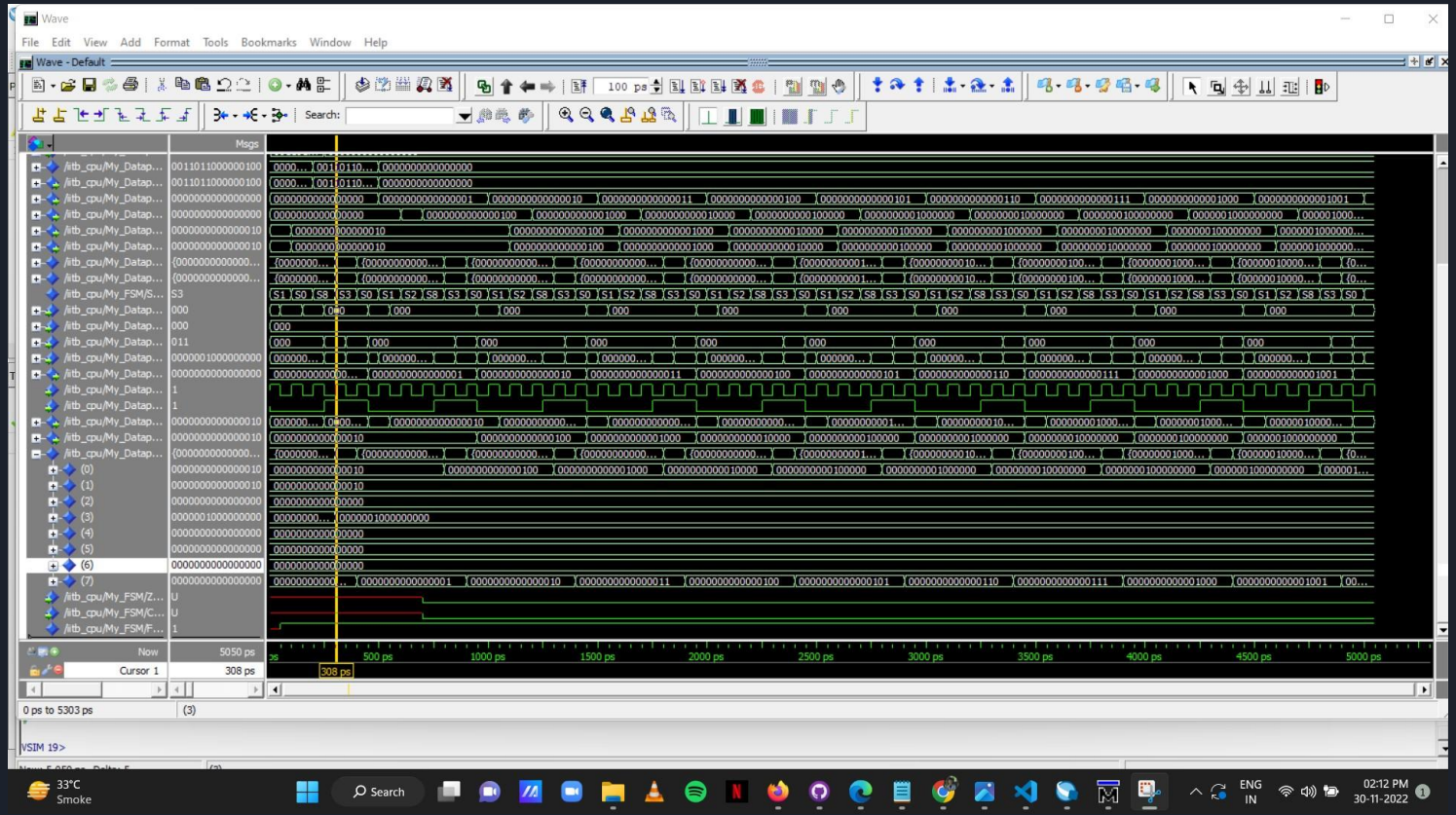


NDU Instruction

We have NANDed R0 and R1 and stored it in R2.

See the updated values.

LHI Instruction

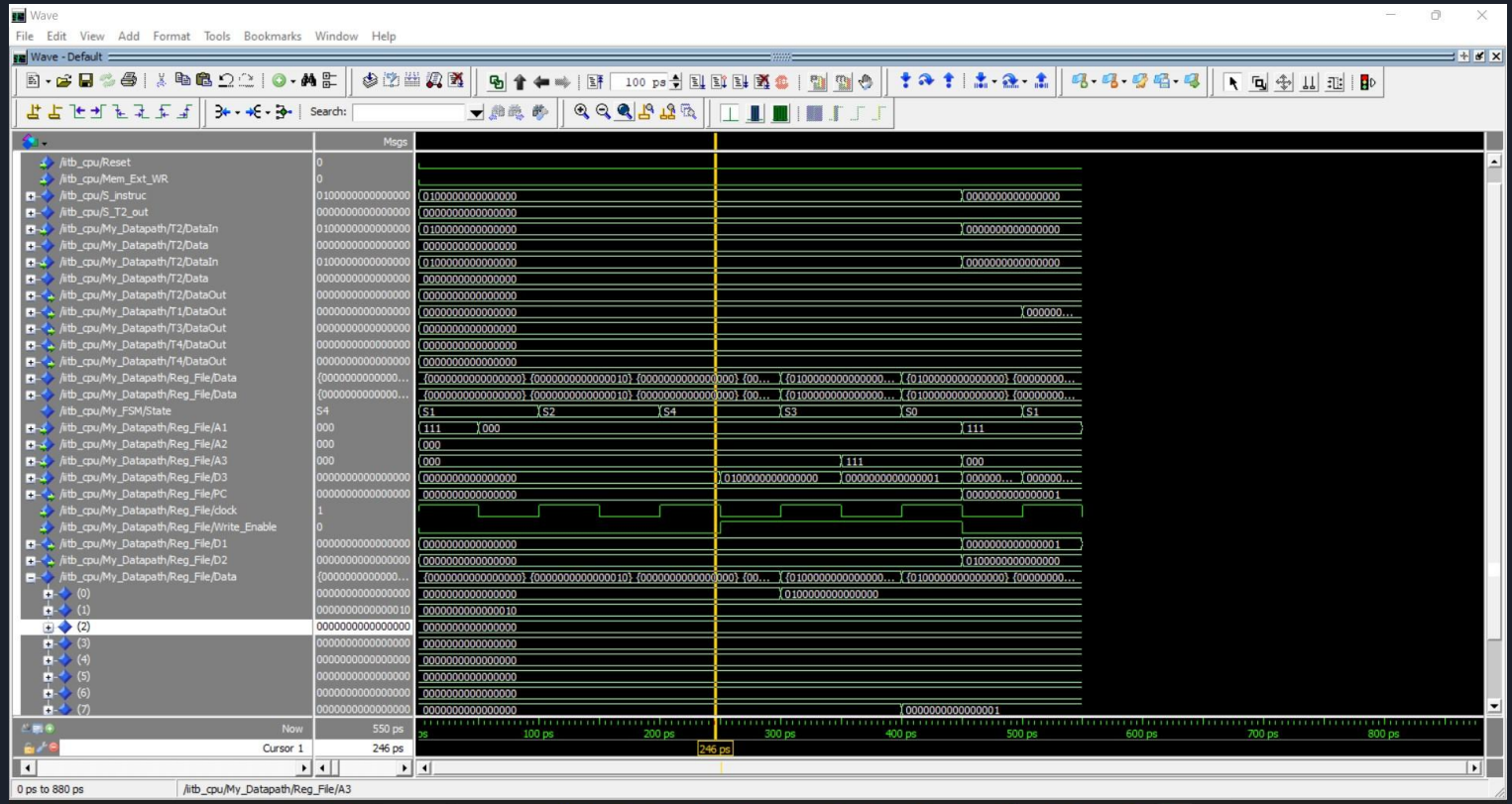


LHI Instruction

Observe R3 in the RTL simulation as soon as clock tick arrives we get a 7 bit shifted Immediate updated on R3.

Observe this in the drop down menu in the RTL simulation.

LW Instruction

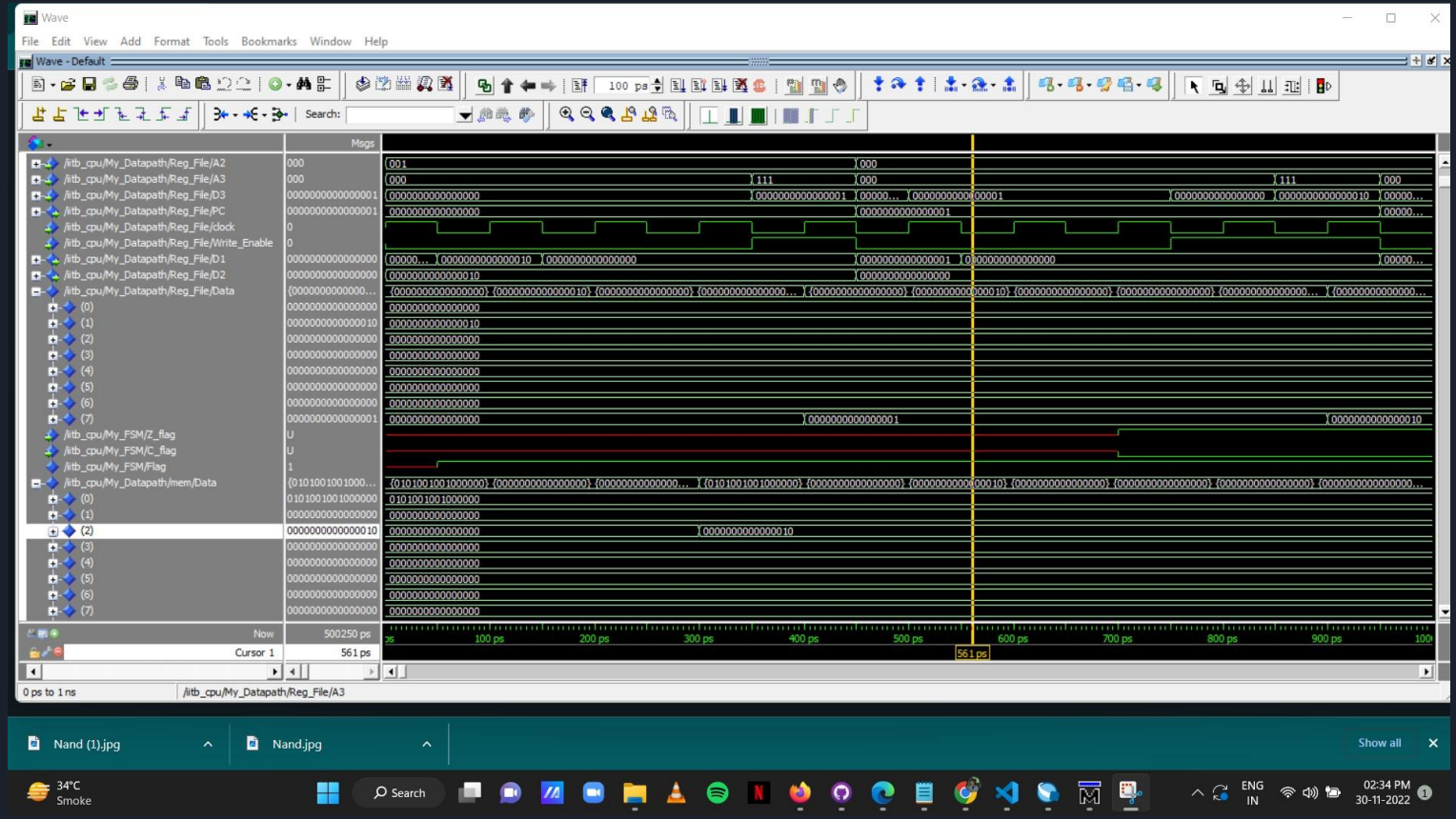


LW Instruction

We load the instruction stored at location 0 offset by zero onto the R0.

Observe the change in value of R0 in the simulation.

SW instruction



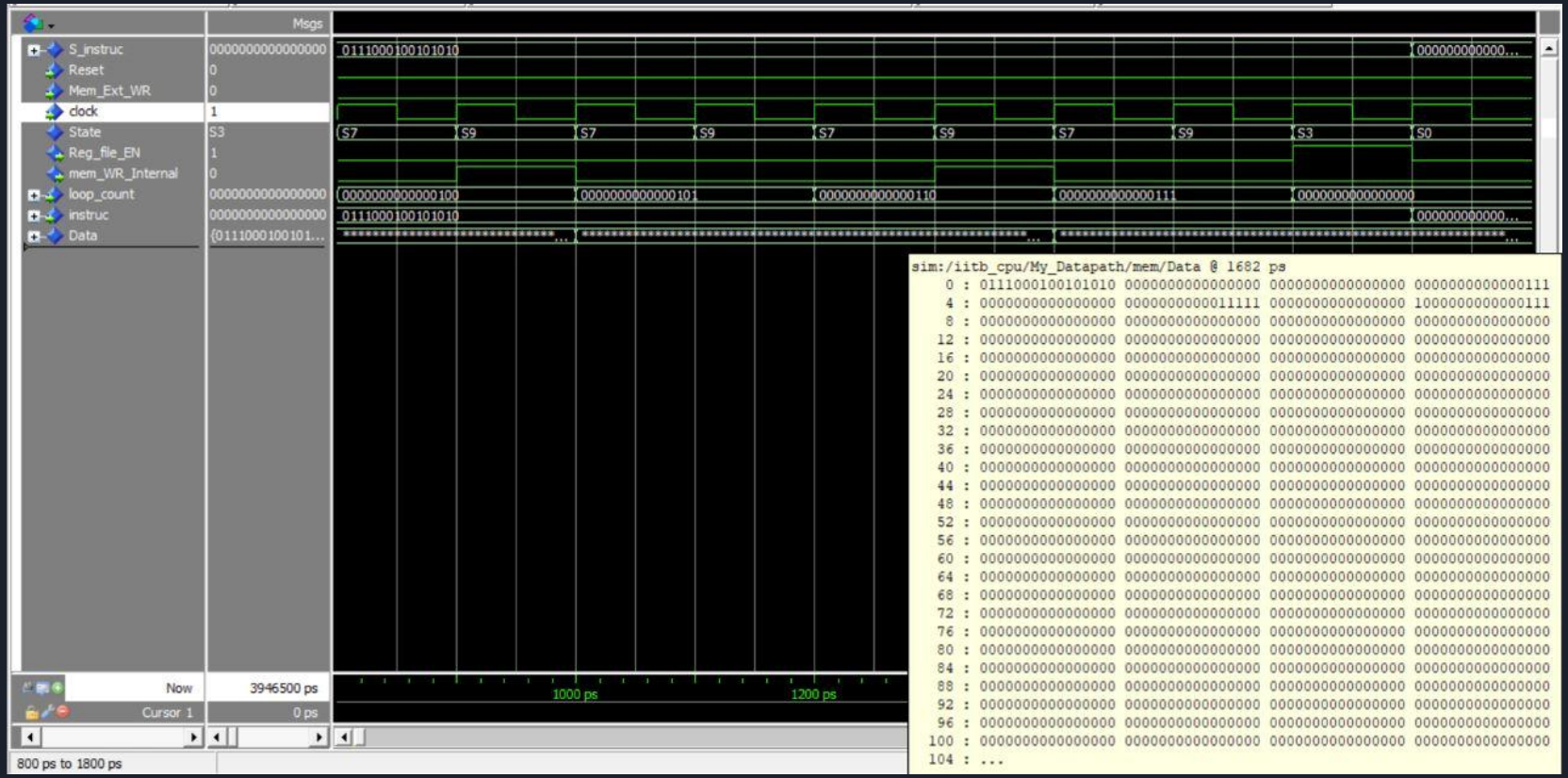
SW instruction

Store the value of R1 where R1 is pointing.

This was the instruction loaded on this example.

We can see the same on the RTL simulation.

SM Instruction



SM Instruction

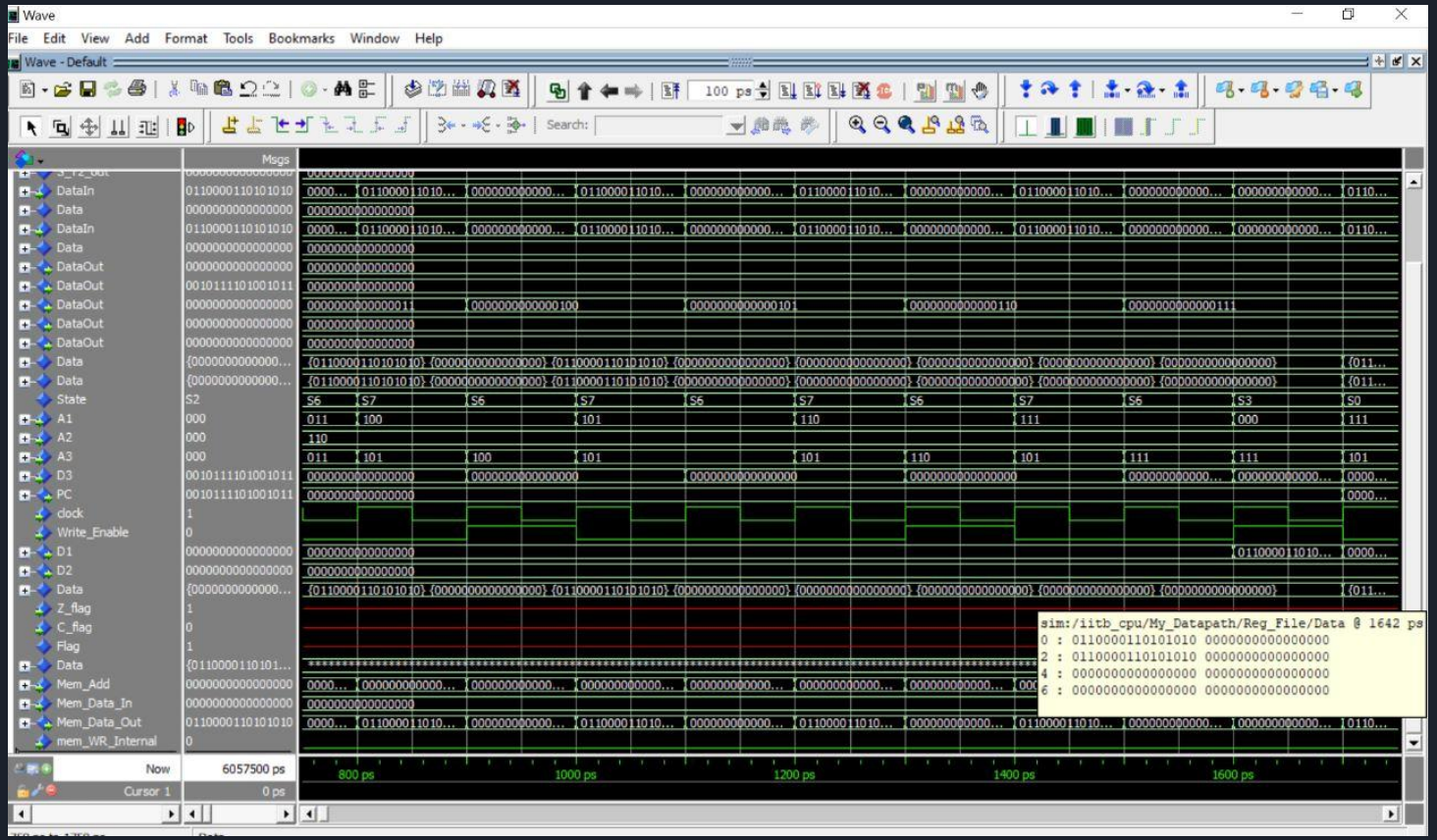
We provided the address of the 2nd register in the memory as the start address and then moved further from left to right.

We see that only locations at index 2,4,6 get written with values.

This is the selective storing feature we observe.

These values are corresponding to the values stored in R2 R4 and R6 of the programmer's register.

LM Instruction



LM Instruction

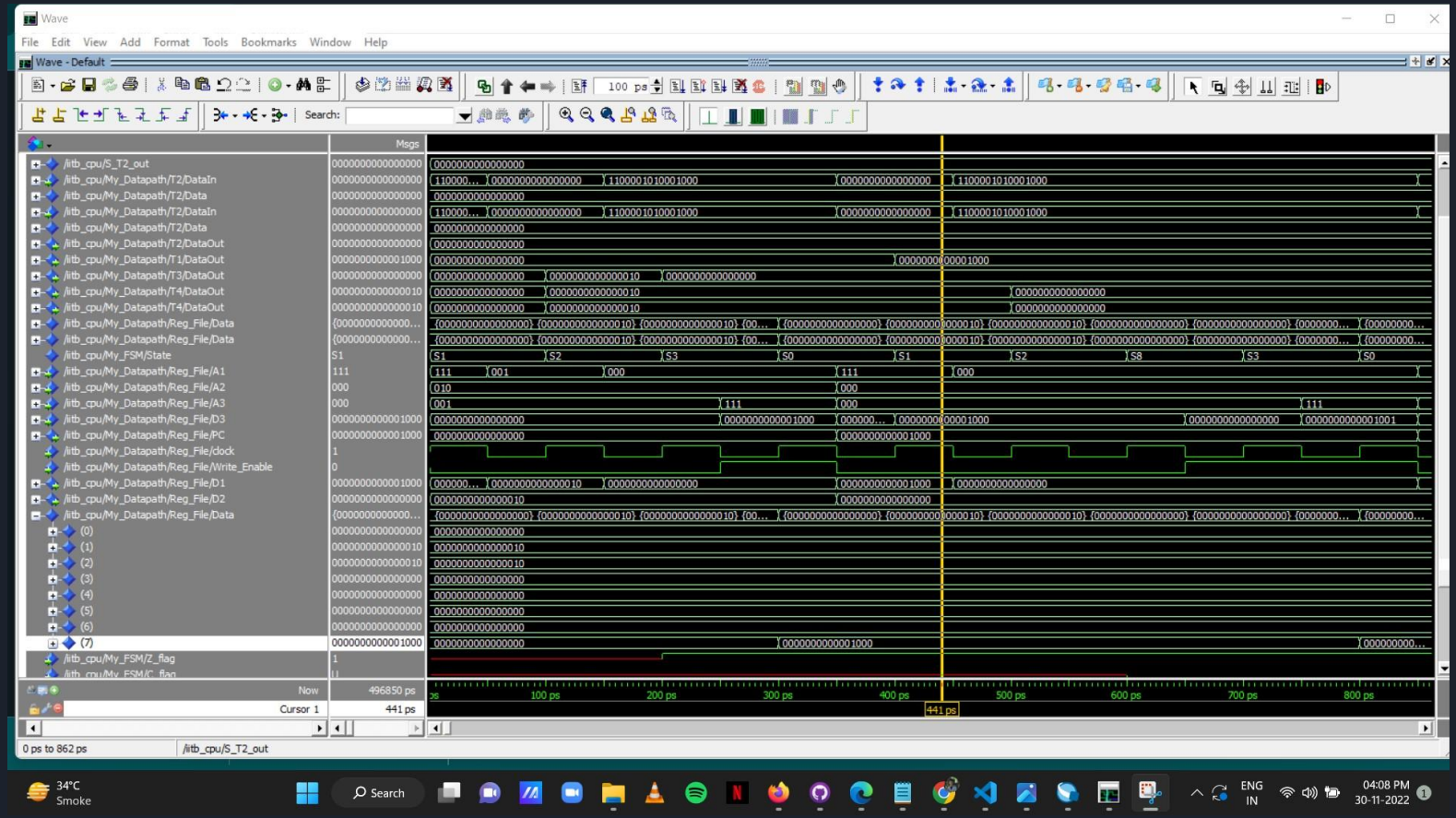
We see that we wish to load on R0,R2, R4,R6 using the instruction LM in the given example.

We see that the initial location has the instruction as the 16-bit value on it. We thus see this loaded on R2 and R0.

However for R4 and R6 we see that the value at the offset locations is 0 in 16-bits and thus that gets loaded onto the 2 registers.

Refer the White pop-up register storage in the image.

BEQ instruction



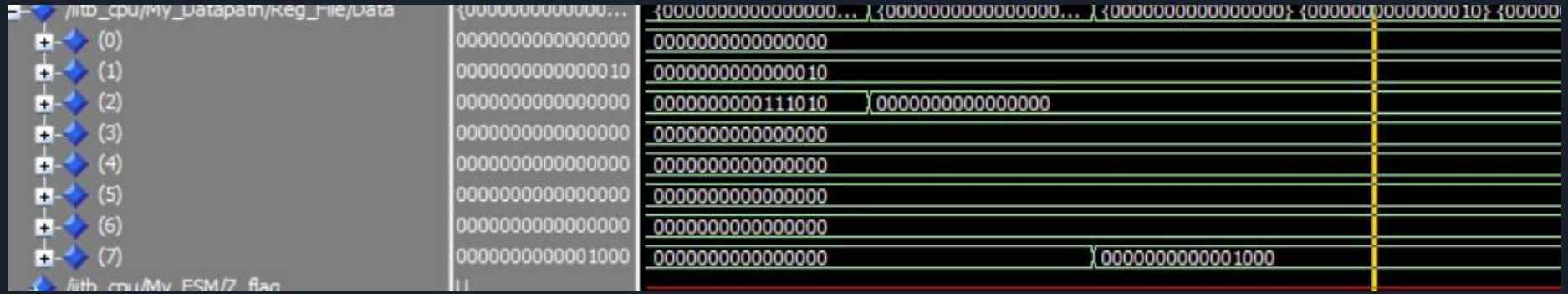
BEQ Instruction

We observe a positive case of branching.

Notice T3 and T4. T3 goes 0 after we compare the values in T3 and T4 which are both 0000000000000010.

And thus a successful branching occurs.

JAL Instruction



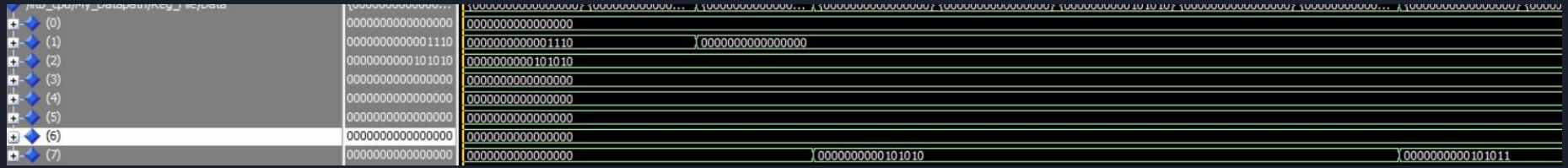
JAL Instruction

Observe the contents of R2 are modified.

Also R7 branches to an address with Imm+PC i.e. 0000000000001000 here.

See (2) and (7) in RTL simulation.

JLR Instruction



JLR Instruction

We put the program counter inside register R1(001)

We see the value of R7 chance to value of R2(010) after a clock ticks.

See the RTL simulation of (1),(2),(7) in the photo for this instruction.