SIMPLERISC CPU State-Control Report

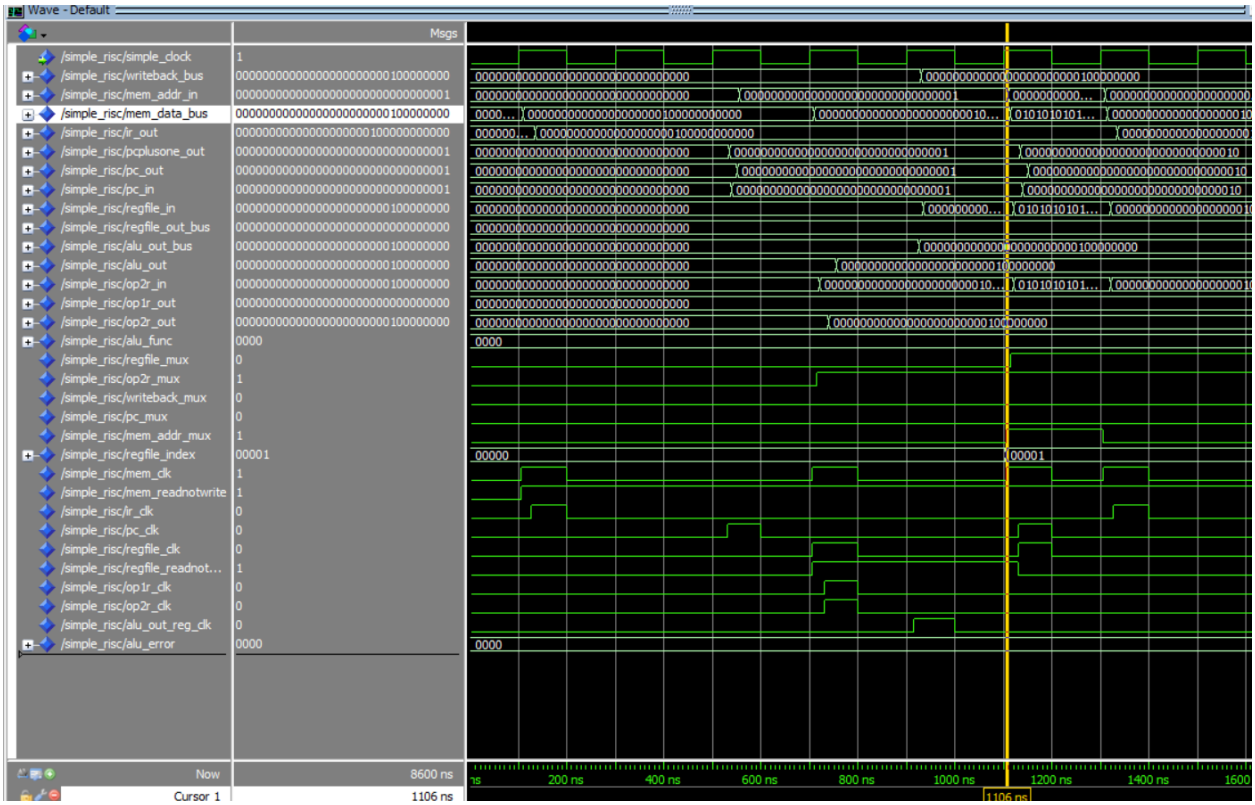
Student: Yinbo Chen

This report contains all 18 control states. I performed 5 operations based on the time sequence. There are LD, ADD, STO, JMP, and JZ. The structure shows below.

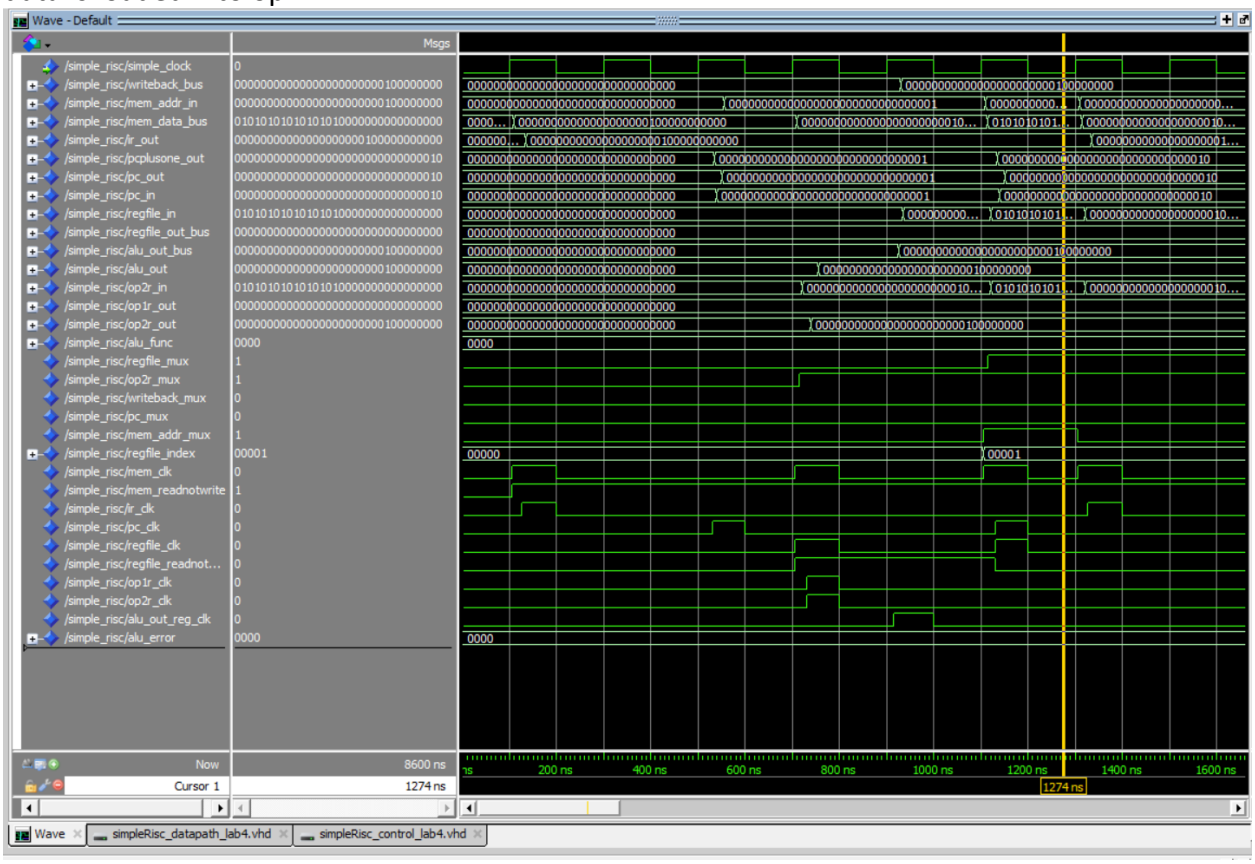
```
   data_memory(0) :=  "00000000000000000000100000000000";   -- LD R1,R0(100)
   data_memory(1) :=  "00000000000000000000000100000000";
   data_memory(2) :=  "00000000000000000001000000000000";   -- LD R2,R0(101)
   data_memory(3) :=  "00000000000000000000000100000001";
   data_memory(4) :=  "00001000001000100001100100000000";   -- ADD R3,R1,R2
   data_memory(5) :=  "00000100011000000000000000000000";   -- STO R3,R0(102)
   data_memory(6) :=  "00000000000000000000000100000010";
   data_memory(7) := "00001100100000000000000000000000";    -- JMP R4(00000010)
   data_memory(8) := X"00000010";
   data_memory(16):=  "00010000100001010000000000000000";   -- JZ R5,R4(00000000)
   data_memory(17) := X"00000000";

-- data for the first two loads to use
    data_memory(256) := X"55550000";
    data_memory(257) := X"00005555";
    data_memory(258) := X"ffffffff";
```

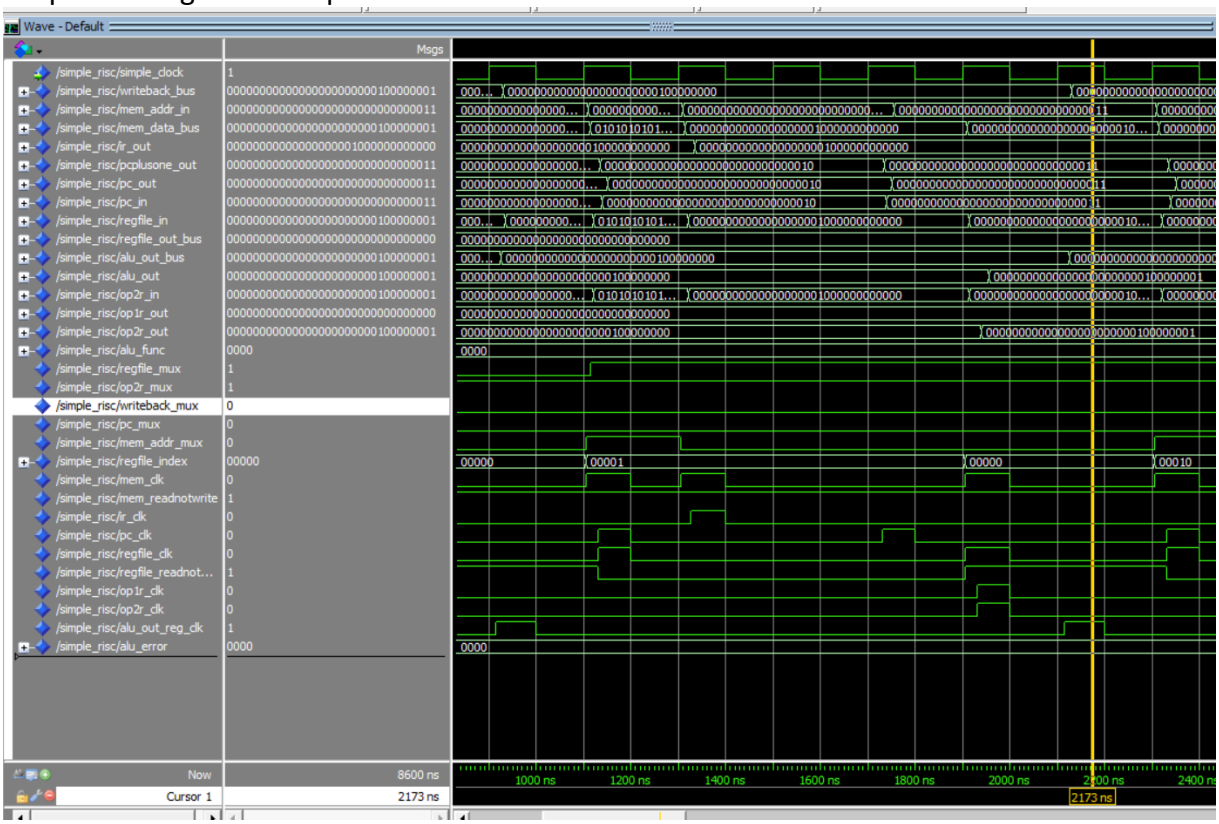
Based on the instruction. The data_memory(0) has already been loaded into the writeback_bus. You can find the data info shown below. This image also shows that the pc_out is 1. It's reading the data from data_memory(1) and the result shows in the mem_data_bus.



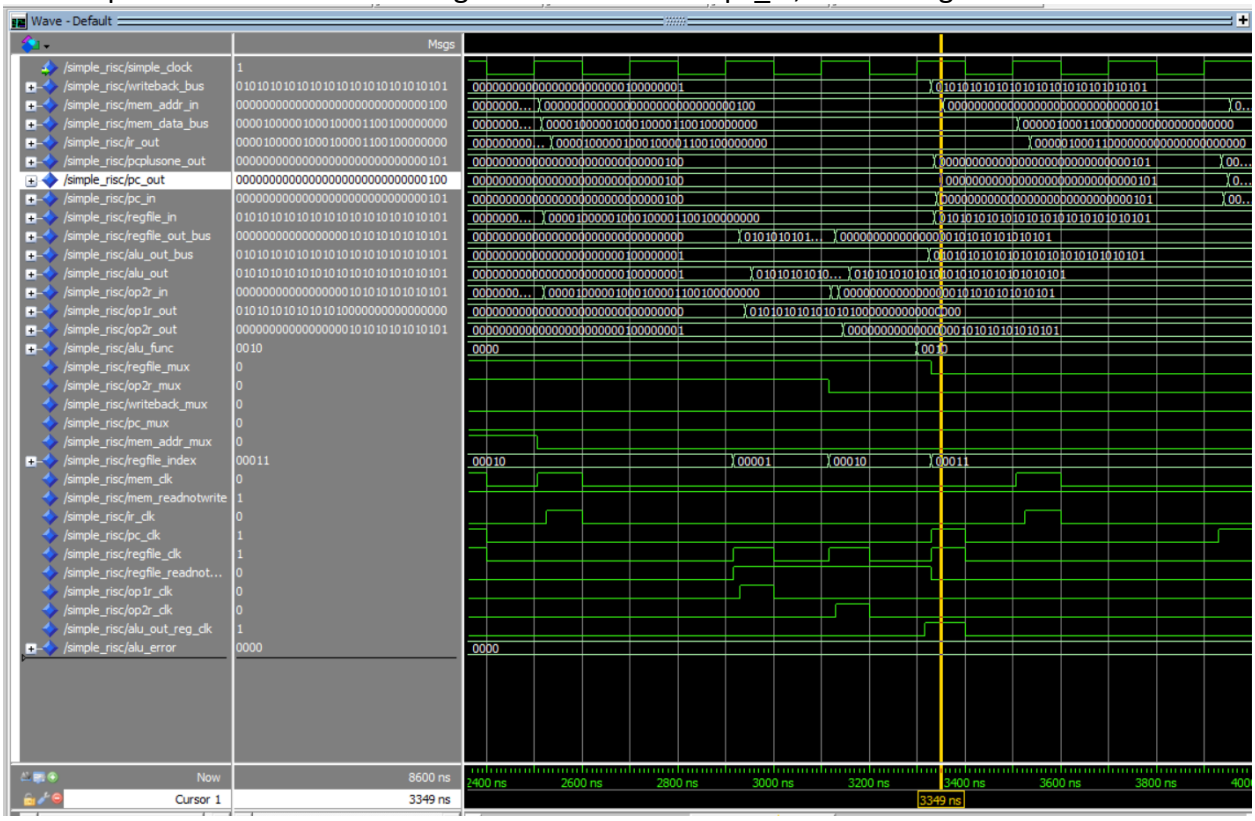
In this picture, the mem_data_bus has the data X55550000 from the data_memory(256). This data is loaded into op2r.



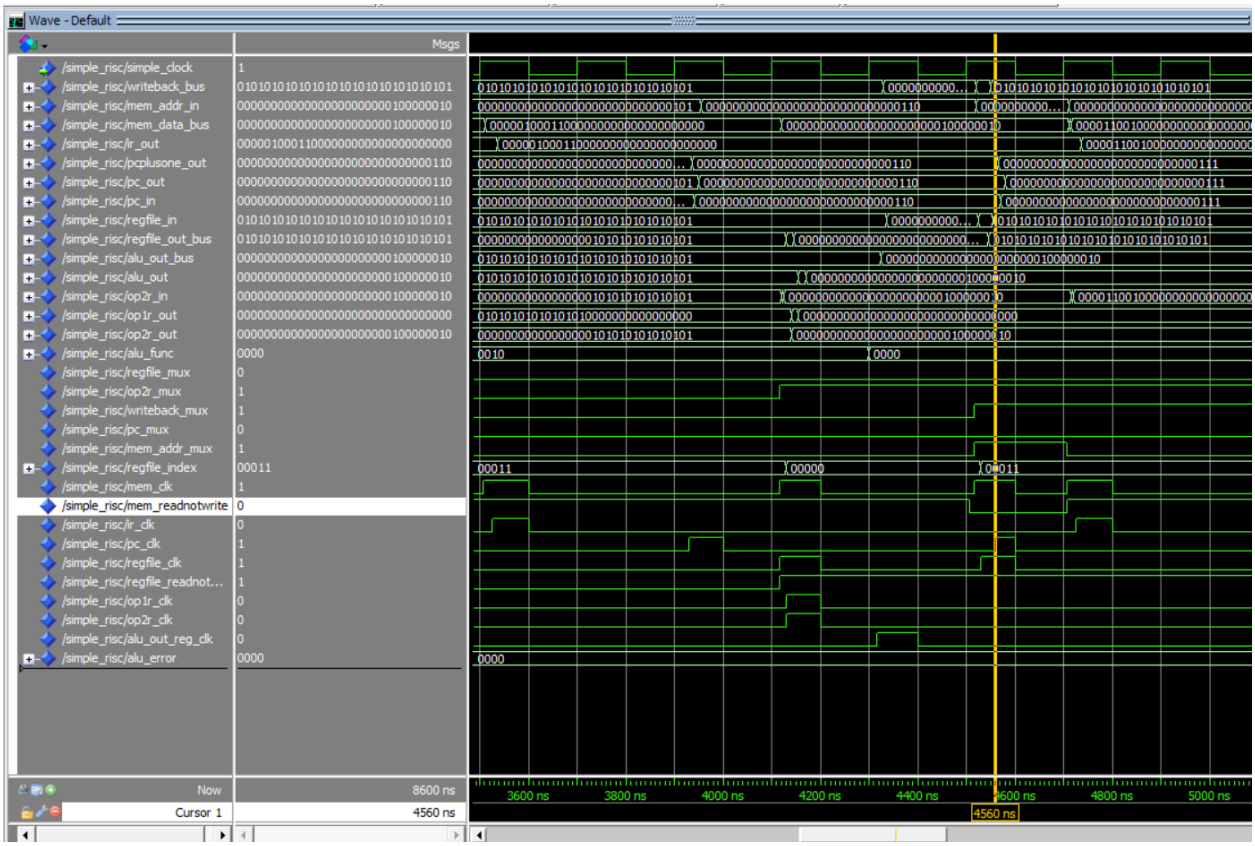
Currently, it's reading the information from data_memory(3) which you can find the binary value at pc_out. The data from this location goes into mem_data_bus. After this step, it's ready for performing the add operation.



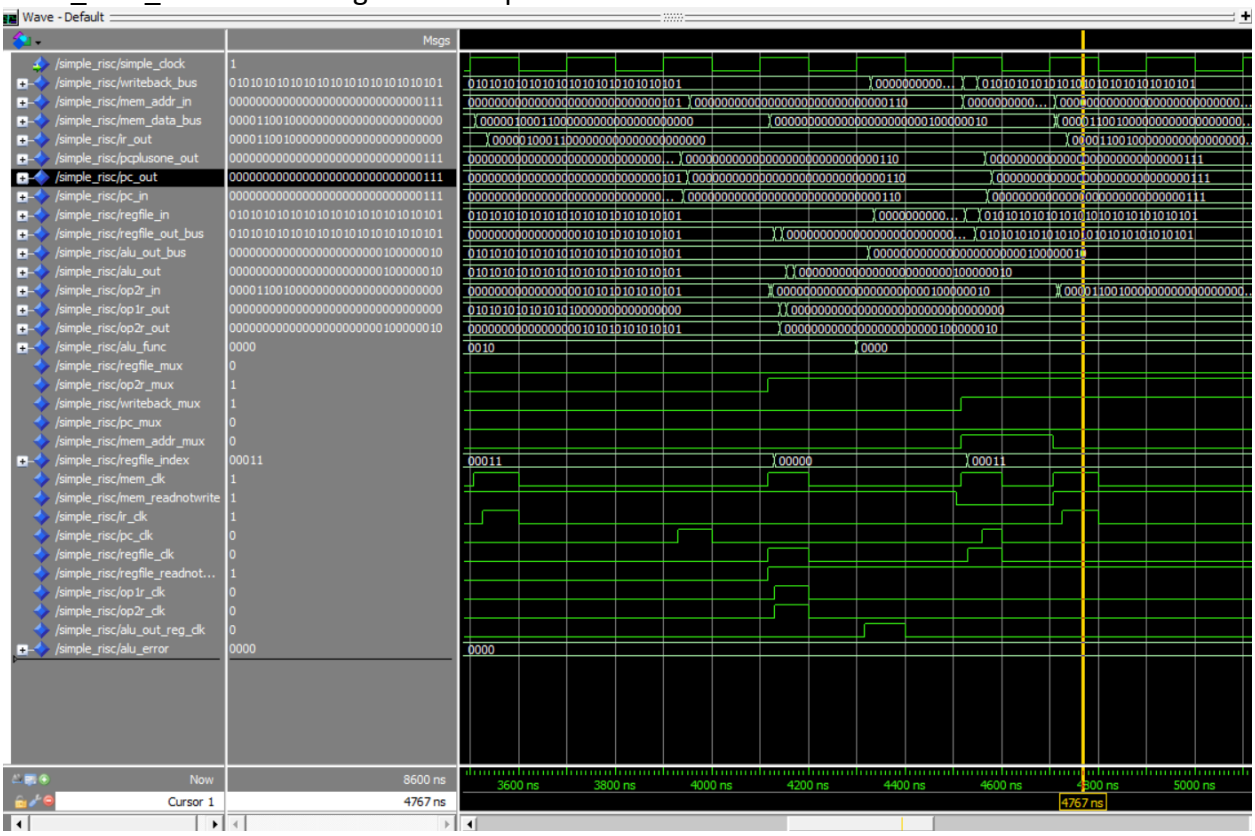
This step shows the adding result. You can read the instruction binary codes from mem_data_bus which is the same as data in the data_memory(4). The pc_out also shows the memory location which is at 4. From op1r_out and op2r_out, you can find the data read from data_memory(256) and (257). The adding result goes into writeback_bus and is ready for the next step to store the result that signal can be found in pc_in, it's reading the next instruction.



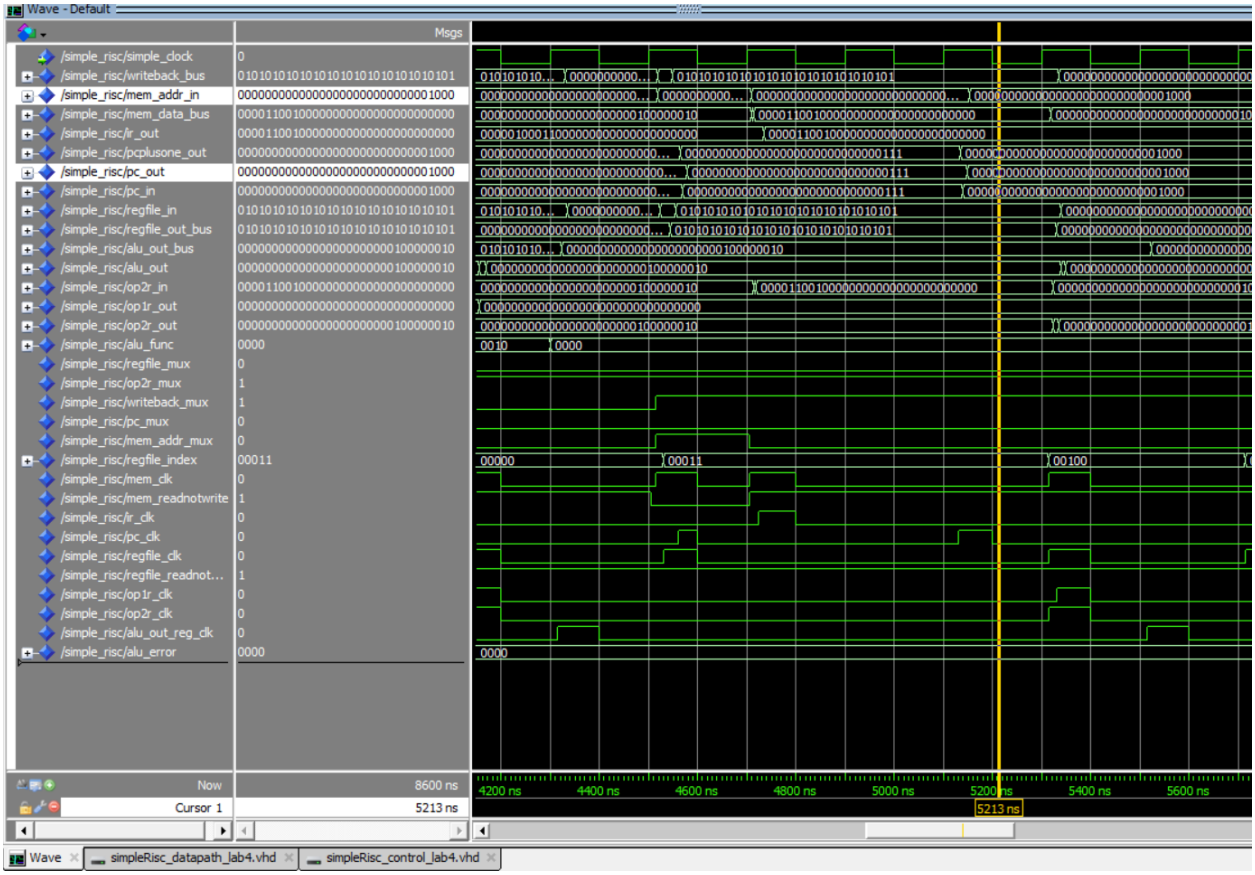
This figure shows the store operation. You can find the writeback_bus keeps the calculated result from the previous adding step. The mem_readnotwrite signal goes to 0. It's writing the data.



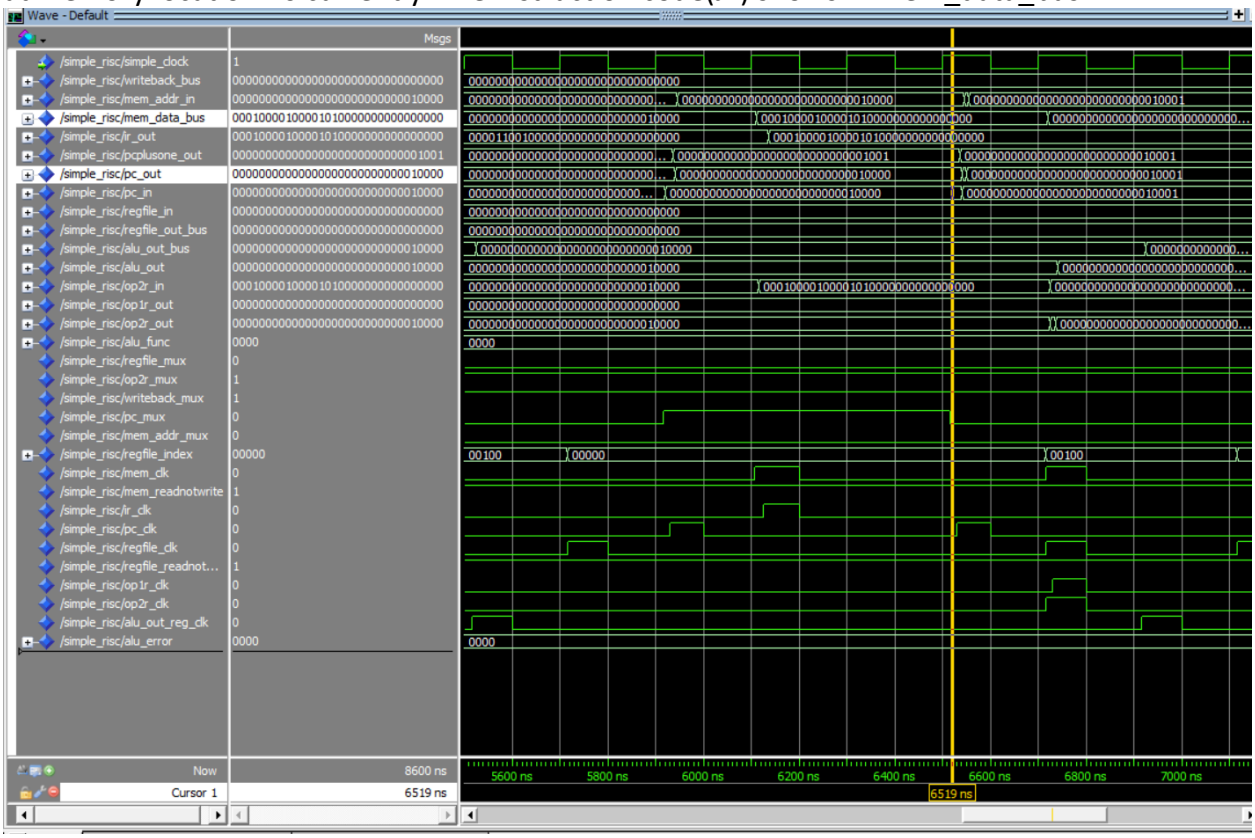
The program pointer continually counts the instruction. Pc_out shows binary value 7 that means it reads operation code from data_memory(7). The code info can be read in mem_data_bus. It's starting the JMP operation.



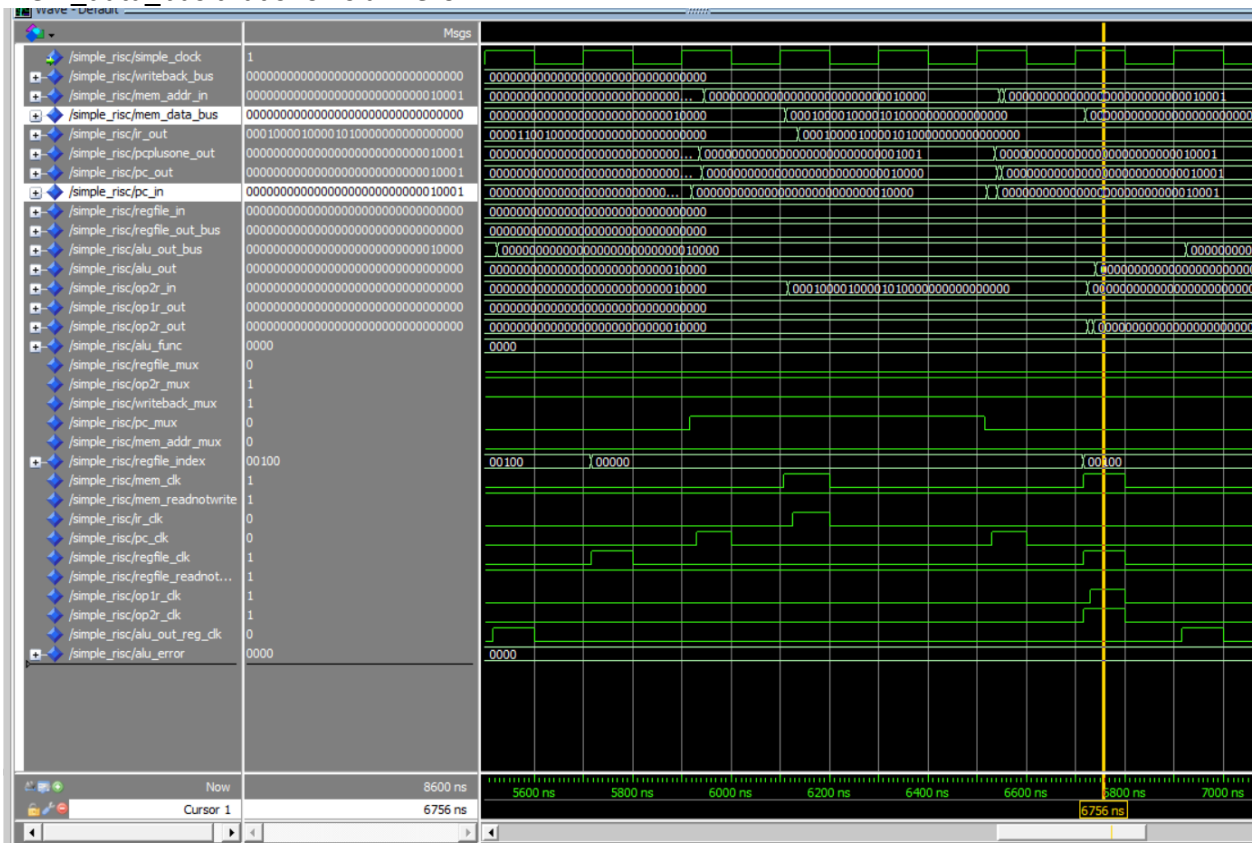
Instruction counter at location 8. (pc_out is binary value 8) The value in mem_addr_in hasn't changed due to the time delay. It should be as same as the value in the data_memory(8). We will observe the value at the next step.



As mentioned in the last step. You can see the value in the mem_addr_in is the same as the value shown in data_memory(8). Also, you will find the instruction counter has already increased at 16 from the pc_out channel. It means it performed JMP operation and the counter at memory location 16 currently. The instruction code(JZ) shows in mem_data_bus.



The program counter is at memory location 17. Then read the data at data_memory(17) into mem_data_bus that shows all zero.



In the final step, PC_in goes to 0, reads data from data_memory(0), and puts operation code into mem_data_bus. It finished a cycle. From this point, the program will continually execute the instructions shown above.