IITB RISC 2022 MULTICYCLE IMPLEMENTATION

Karrthik Arya 200020068

Kaishva Shah 200020066

Ronil Mandavia 20D180020

Siddhant Das 20D070075

1 Information about the design

Owing to the fixed encoding in RISC architecture we have made the PC move 2 bytes at a time. To command the PC to stop we added a state with state id 111111 which serves as an infinite loop. The instruction register IR stores the current instruction. Eight 16-bit registers from R0-R8 were implemented using an array in regs.vhdl. The writing operation for all memory elements was done on the negative clock edge. Sign extender and shifters are used as required to send the input to ALU in the required format. ALU was made using behavioral modeling and it can perform addition, subtraction, NAND, and compare operations as well as check the zero and carry flags. An FSM was made to control the flow of states depending on the current instruction. A state signal has been sent to every component to control which operation is performed by them in that state. A clock of frequency 100MHz has been used.

2 Feeding Instructions

IITB-RISC-22 is a 16-bit microprocessor having 64 bytes of data memory and 64 bytes of instruction memory. Instructions can be given to the processor using the mem.vhdl file in which the instructions can be directly fed into the instruction memory array.

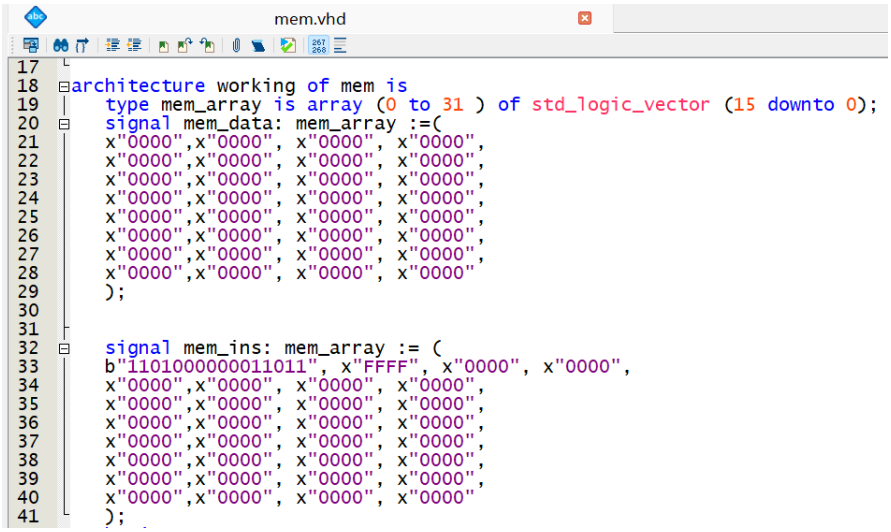

Figure 2.1: Instruction feeding

It is important to note that an end instruction, given by x"FFFF", has to be input at the appropriate location the program is expected to end as seen in the above example. Also the project statement had the same OP CODE for LHI and ADI. Hence we have taken the OP CODE for LHI as 0011 and kept the OP CODE for ADI as 0000.

3 Altering Data Memory

Data memory, similar to instruction memory, is also of 64 bytes and can be accessed in the mem.vhdl file.

The picture shown has value 1,2,3,4,5 stored at addresses 0,1,2,3,4 respectively.
Every element in the array is of 2 bytes, making the array of size 32.

4 Register Access

Eight 16-bit registers from R0-R8 were implemented using an array in regs.vhdl. They can be accessed from the same.

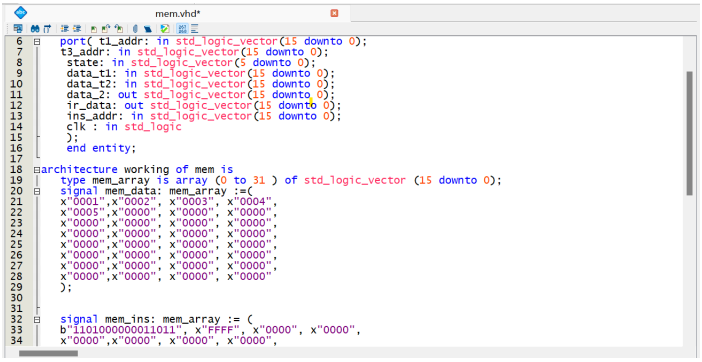

Figure 3.1: Altering data memory

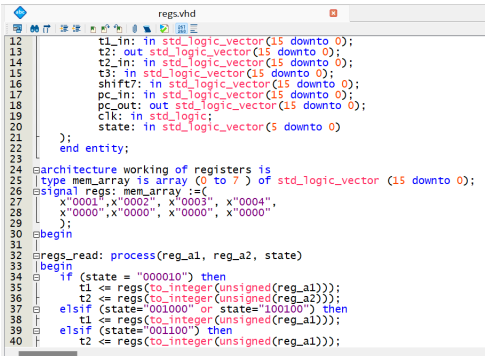

Figure 4.1: Register

The picture shown has value 1,2,3,4 stored in R0,R1,R2,R3 respectively.

5 CHECKING VALUES IN RTL SIMULATION

After compiling successfully, we run RTL simulation. This is where we check if our program has given the output expected out of it.

The value stored in any storage component or any signal can be accessed in RTL simulation.

5.1 CHECKING VALUE STORED IN DATA MEMORY

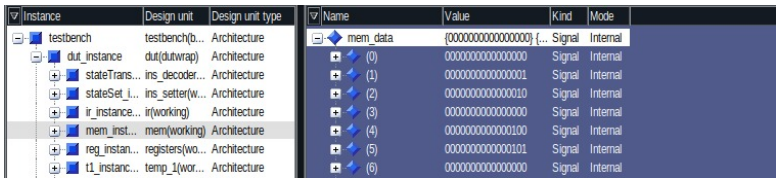

Figure 5.1: RTL Simulation - Memory data

5.2 CHECKING VALUE STORED IN REGISTERS

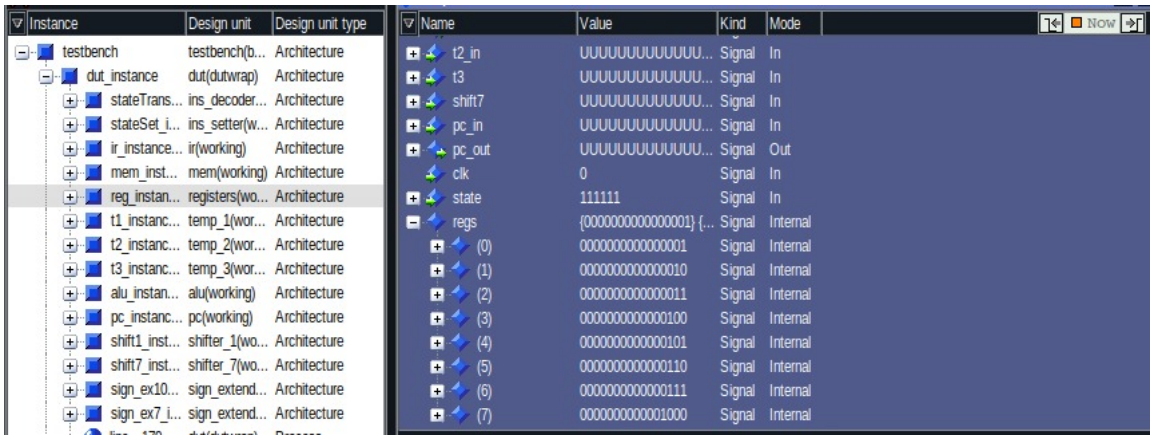

Figure 5.2: RTL Simulation - Registers

5.3 CHECKING VALUE STORED IN INSTRUCTION MEMORY

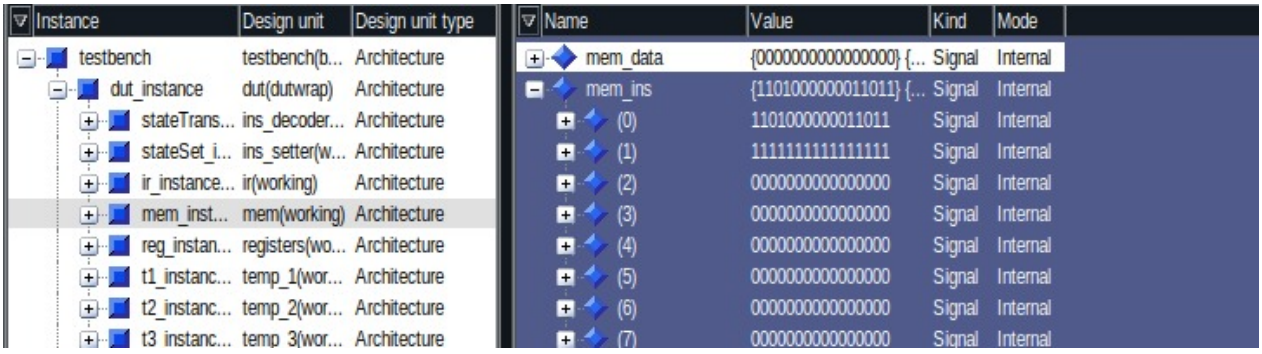

Figure 5.3: RTL simulation - Memory instruction

5.4 Checking value of Program Counter

As seen in the instruction memory the second instruction is the end instruction and hence the value of program counter is set to 2.

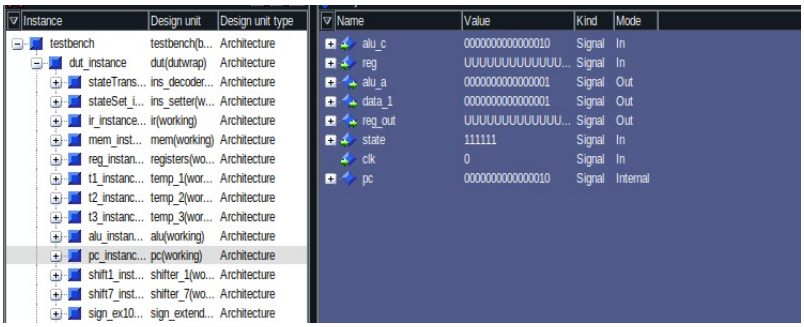

Figure 5.4: RTL Simulation - Program Counter