CSE 590
Computer Architecture

Project 1
Report

Submitted By
Karan Manchandia
(karanman)
Person # 50290755

Status of Demo:

- Simulation demo successfully completed.
- Demo on FPGA board successfully completed.

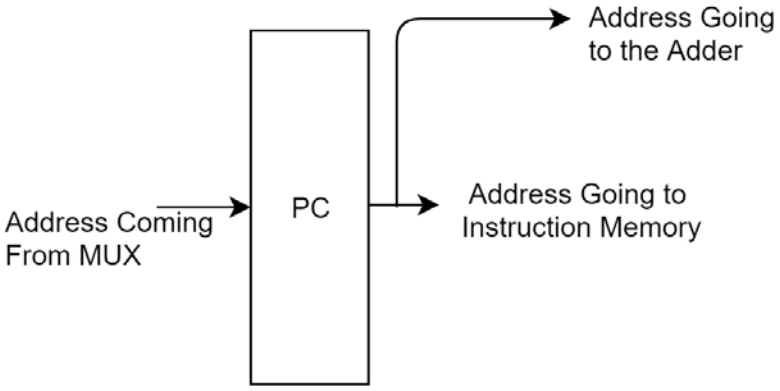
Individual Contribution in Group Members:

- Both the members of the team contributed equally to every section of the project. Most of the parts of the project were done together in the University Library. The work was equally distributed in each of the following steps of the project:
- Drawing the block diagram of the MIPS processor.
- Designing an 8-bit microprocessor using Verilog HDL.
- Designing individual components using behavioral Modelling.
- Simulation.
- Testing the Project on the FPGA Board.

Mandatory Components of the MIPS processor:

Program Counter:

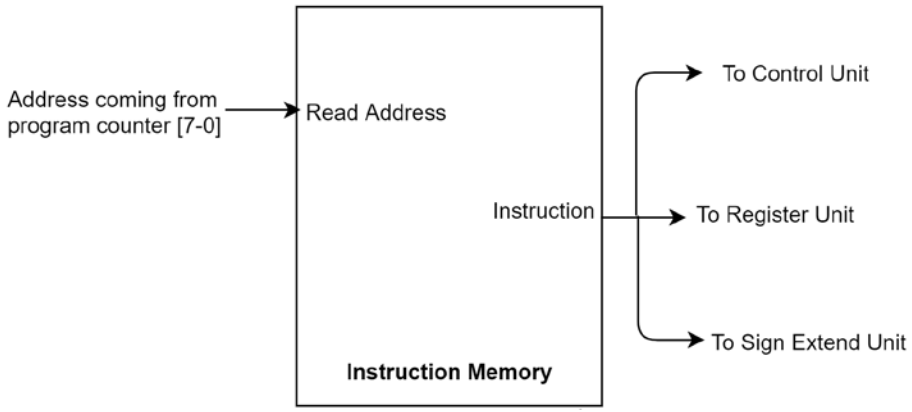
- A program counter is simply defined as a register structure that contains the address pointer value to the current instruction.
- The 8-bit MIPS processor in the program counter takes the input from the MUX and gives the address to the instruction memory and the adder unit.



- In each clock cycle the value in the program counter is read into the instruction memory and the program counter is updated to point to the next instruction.

Instruction Memory:

- The 8-bit address from the program counter comes to the instruction memory. The instruction memory fetches the instruction in the form of an 8-bit array data.



- So, we can say that instruction memory is just responsible for storing the 8-bit instruction in the form of an array.
- In this project the instruction format for the R-type, I-type and J-type instructions are designed as per the project guidelines.
- The R-type instructions include all the integer arithmetic and bitwise operations along with nonbranching compare instructions such as slt, sgt and seq. The R-type instructions that we are implementing in this project are add and sub.
- The R-type instruction format given in the project description is:

Opcode			**Rt/Rd**	**Rs**	**Unused**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

- Load (lw) store(sw) and addi instructions are I-type instructions. The I-type instruction format given in the project description is:

Opcode			**Rd**	**Rs**	**Immediate**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

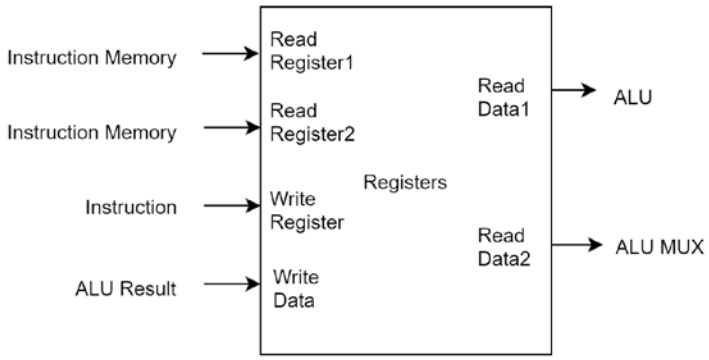
- Jump(jmp) is J-type instruction. The J-type instruction format given in the project description is:

Opcode			**Address**				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

- In the R-type instruction the opcode specifies the type of the instruction. Rt/Rd tells which register is the target register and Rs specifies the second operand.
- In the I-type instruction the 3 bits of the operand specifies whether it is a addi, lw or sw instruction. Rt specifies which register is the first operand and Rs specifies which register is the second operand.

- In the J-type instruction the 3-bit opcode specifies the type of the instruction. Address specifies the address of the location where the program counter has to jump.

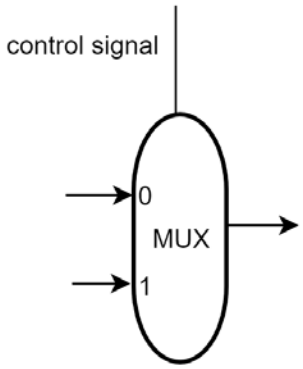
Register:



- The register file is a small set of high-speed storage cells.
- A MIPS processor has 32 general purpose registers. So, it takes 5 bits to specify which register to use.

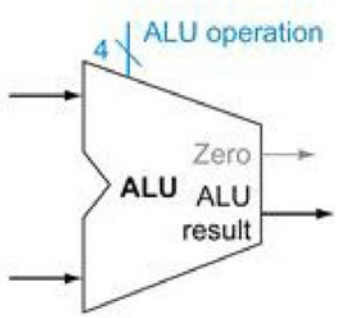
Multiplexer (MUX):

- The multiplexer, shortened to "MUX", is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal.
- A MUX has two input signals, a select signal to decide which input is to be given as a output and one output signal.
- There are 4 MUX used in our processor design. All the MUXs are with different inputs and different control signals which are ALUsrc, MemToReg, Jump, RegToDest.



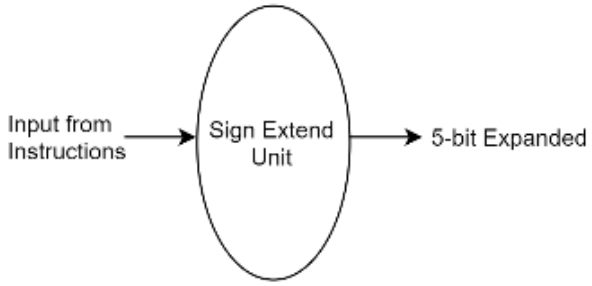
ALU:

- An ALU in general is a combinational circuit capable of computing a variety of arithmetic and logical functions.



- The first input to the ALU is the first operand data and the second input is the second operand data and the output is supplied to the data memory in case when and address is involved such as in case of ADD instruction or the jump MUX in case of the jump instruction execution.
- As per description of our project the ALU performs two kind of operations on the 5-bit operands involved, one is the bitwise addition and the second is the bitwise subtraction.
- When LoadWord or StoreWord instructions are getting executed the ALU is responsible for adding data from one of the register and the corresponding offset.
- When the ALU performs ADD or SUB instructions, it uses the data from both the registers. The second input coming into the ALU is from ALU MUX which decides to take data from read register 2 or the sign extend unit data during the execution of ADDI instruction.

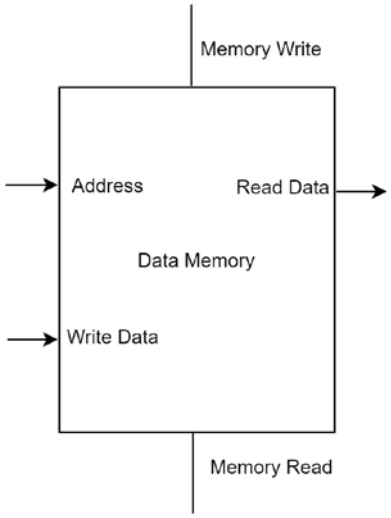
Sign Extend Unit:



- In general, sign extension is the operation of increasing the number of bits in a binary number while preserving the sign and the value. The input to the sign extend unit is coming from instruction memory.
- The function of the sign Extend unit is to append two 0's in front of the 3-bit address to convert it into a 5-bit address. Since, only zeros are added to the left there is no change in sign or in value.

Data Memory:

- Data Memory gets an address as the input and gives the read Data as the output. It also gets write data as an input for writing to the memory location.



- The Memory Write and Memory Read Signals tells the Data Memory the type of operation to be performed.
- Data Memory can store the data from 0 to 256. Therefore, the data memory has 8-bit memory address.
- To read the data memory, set memory read to 1.
- To write the data memory, set memory write to 1.

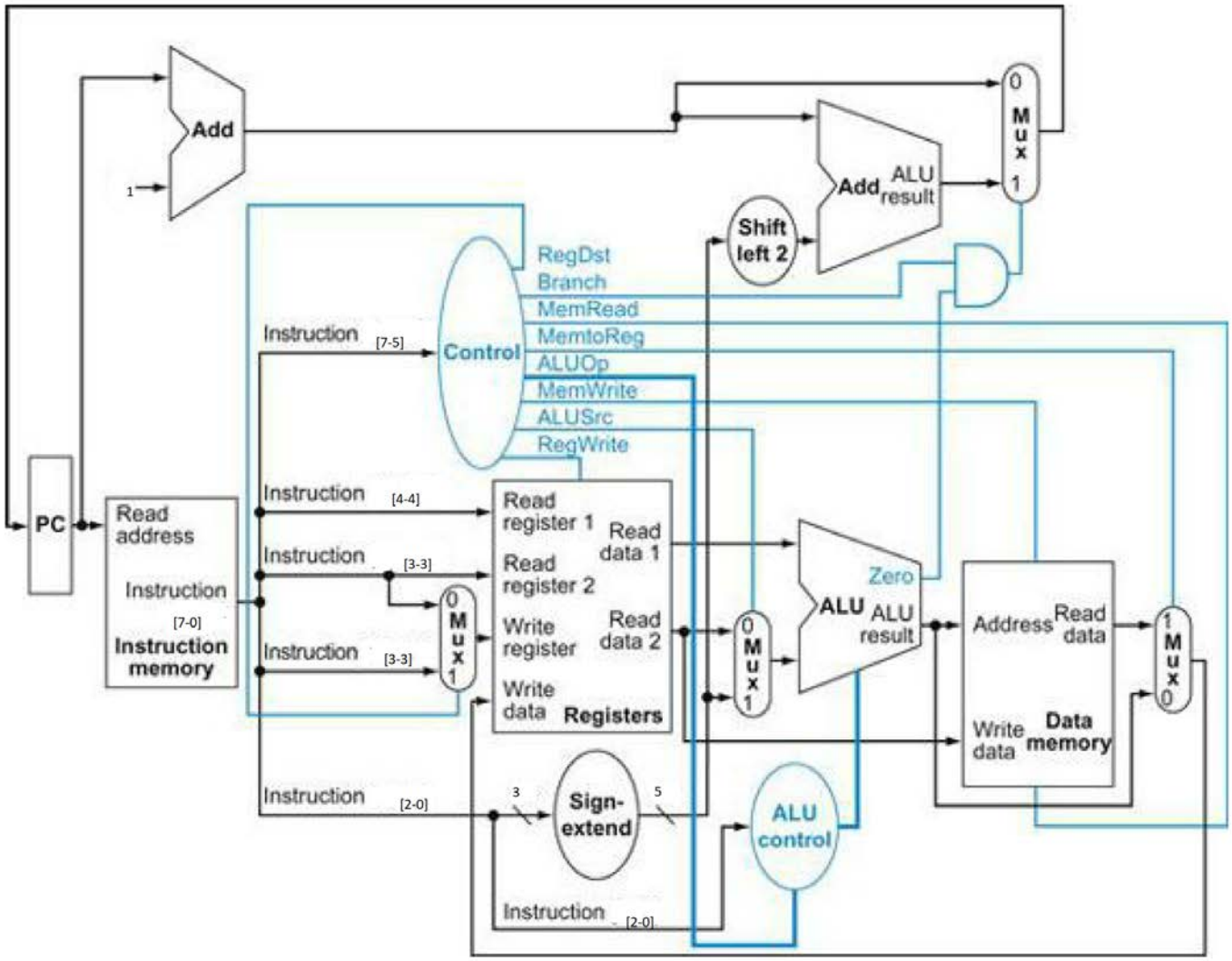
Control Unit:

- The control unit is responsible for giving control signal to all the other units of the MIPS processor. So, the control unit functions to make sure that all the instructions are executed correctly.
- The control unit uses a 3-bit instruction word and a instruction opcode to generate the control signals. After the control signals are generated these are sent to the units where those are inputs.

The control signal table is shown below:

Operation	Opcode	RegDst	RegWrite	Jump	AluSrc	AluOp	MemRead	MemWrite	MemToReg
add	000	1	1	0	0	0	0	0	0
addi	001	1	1	0	1	1	0	0	0
sub	010	1	1	0	0	0	0	0	0
lw	011	0	1	0	1	1	1	0	1
sw	100	0	0	0	1	1	0	1	0
jmp	101	0	0	1	0	0	0	0	0

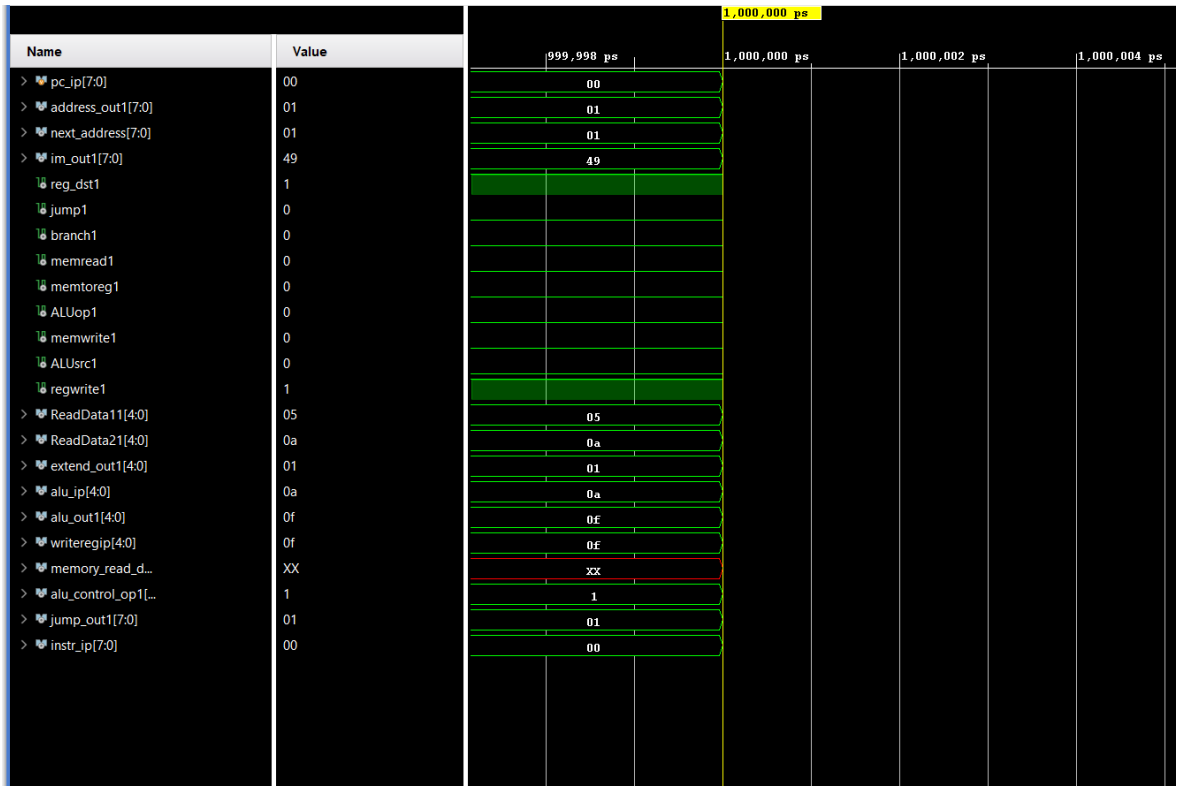
8 Bit MIPS Processor:



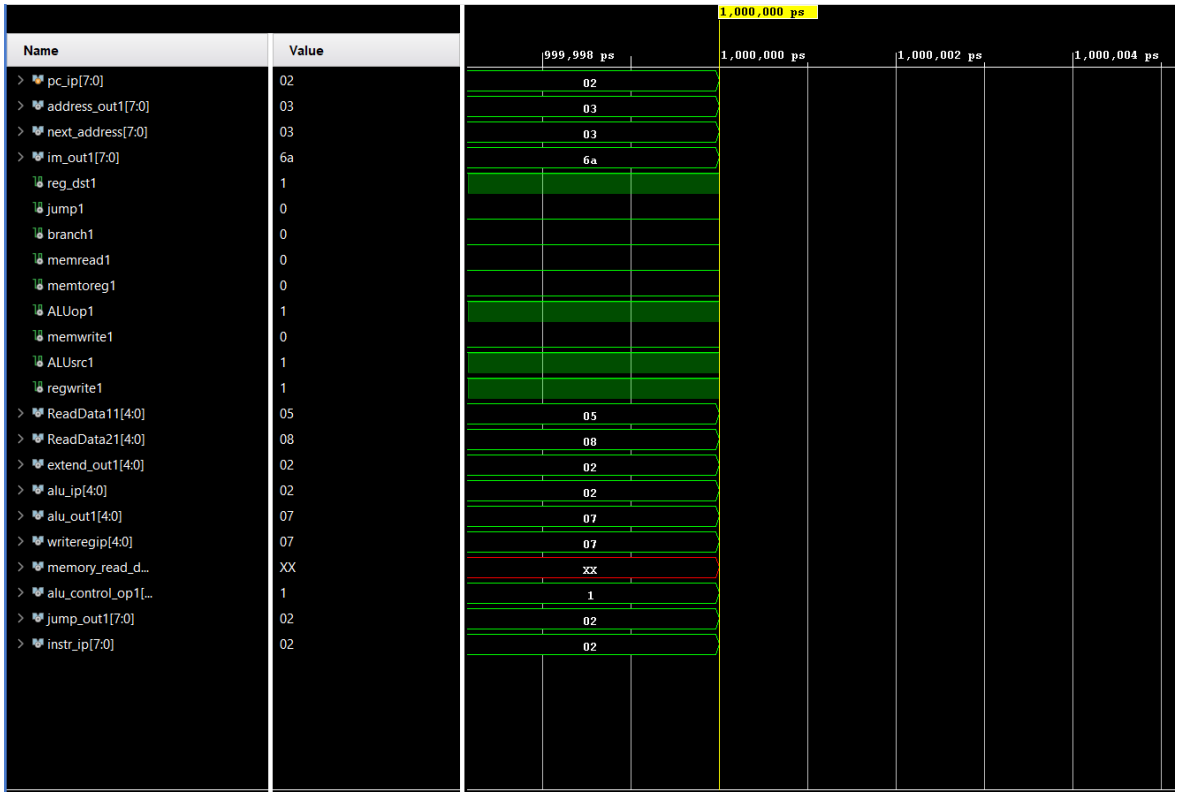
Simulation Results of Each Instructions:

ADD Instruction:

ADD R0 R1
Instruction – 8’b00001001
rt – R0 – 01010 (Content of R0 is 10)
rs – R1 – 00101 (Content of R1 is 5)
Write Register – 01111 (Value written is 15 which is 0f in hexadecimal, which is Alu_out)



ADDI Instruction:
ADDI R0 R1 2
Instruction - 8'b00101010
rs – R0 – 0111 (The result is 7, shown in alu_out in screenshot)
rt – R1 – 00101 (The content of R1 is 5)
immediate – 010 (The value of immediate is 2)



SUB Instruction:

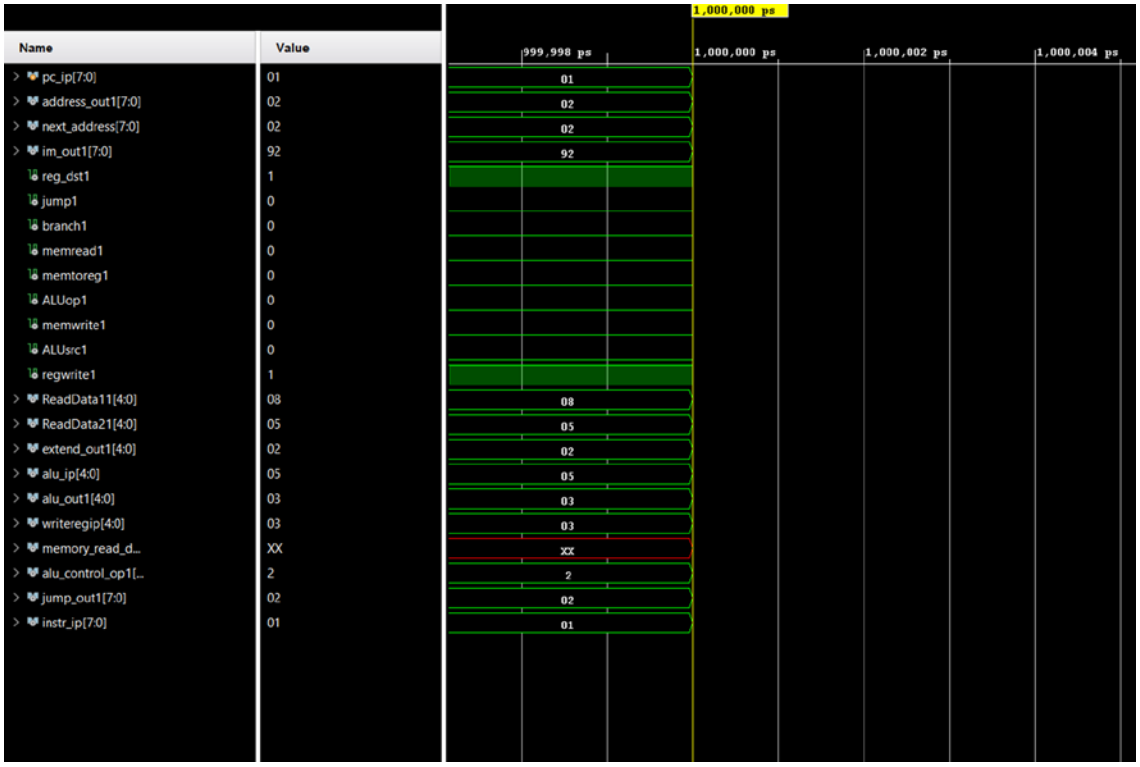
SUB R0 R1

Instruction - 8'b01010010

rs – R0 – 01000 (Register R0 contains 8)

rt – R1 – 00101 (Register R1 contains 5)

Write register - 00011 (The result is 3, shown by Alu_out in screenshot)



LW Instruction:

lw R0 2(R1)

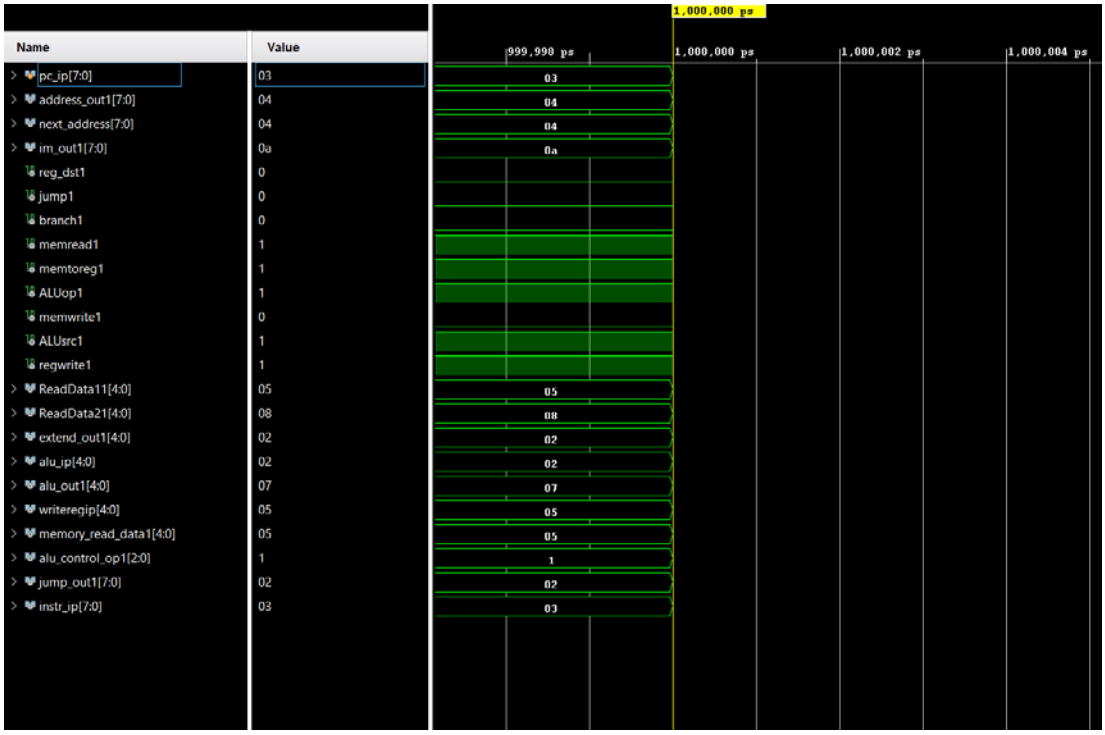
Instruction - 8'b01101010

rs – R0 – 01000

rt – R1 – 00101

immediate – 00010

stored in memory address 7 – 5'b00101



SW Instruction

SW R0 2(R1)

Instruction - 8'b10001010

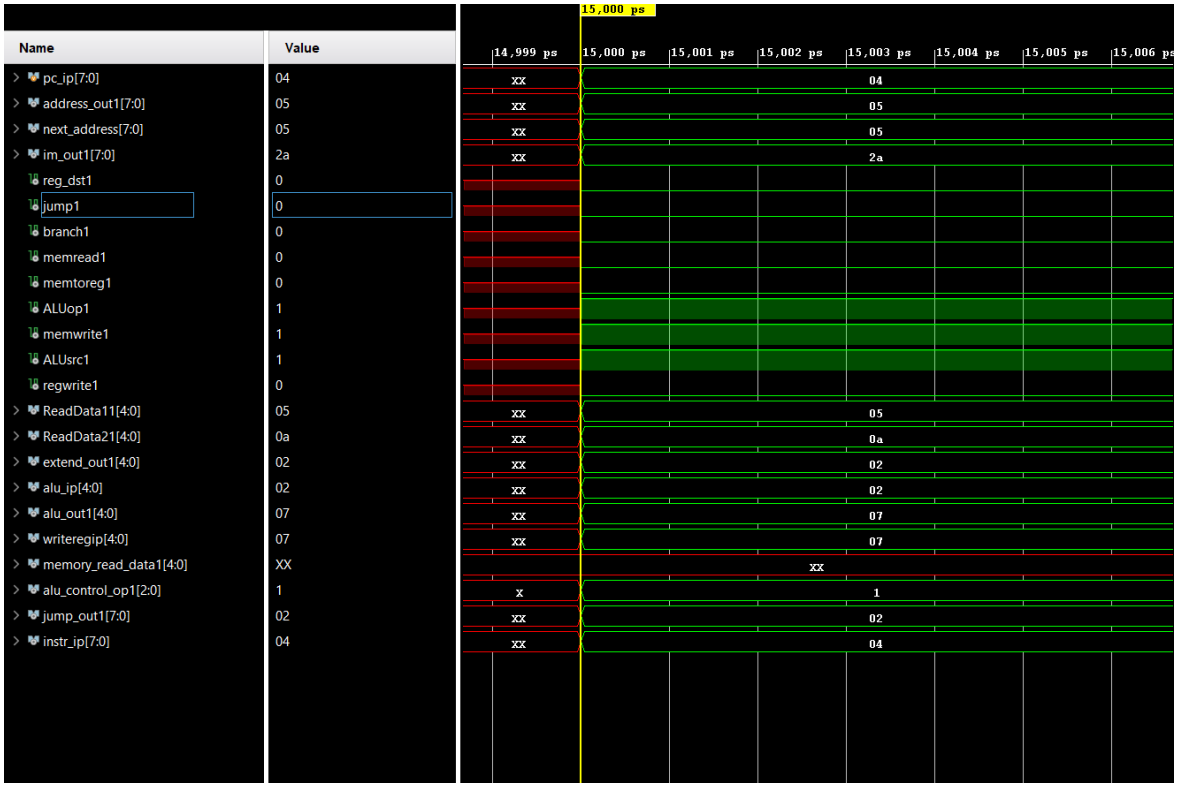
rs – R0 – 01000

rt – R1 – 00101

immediate – 00010

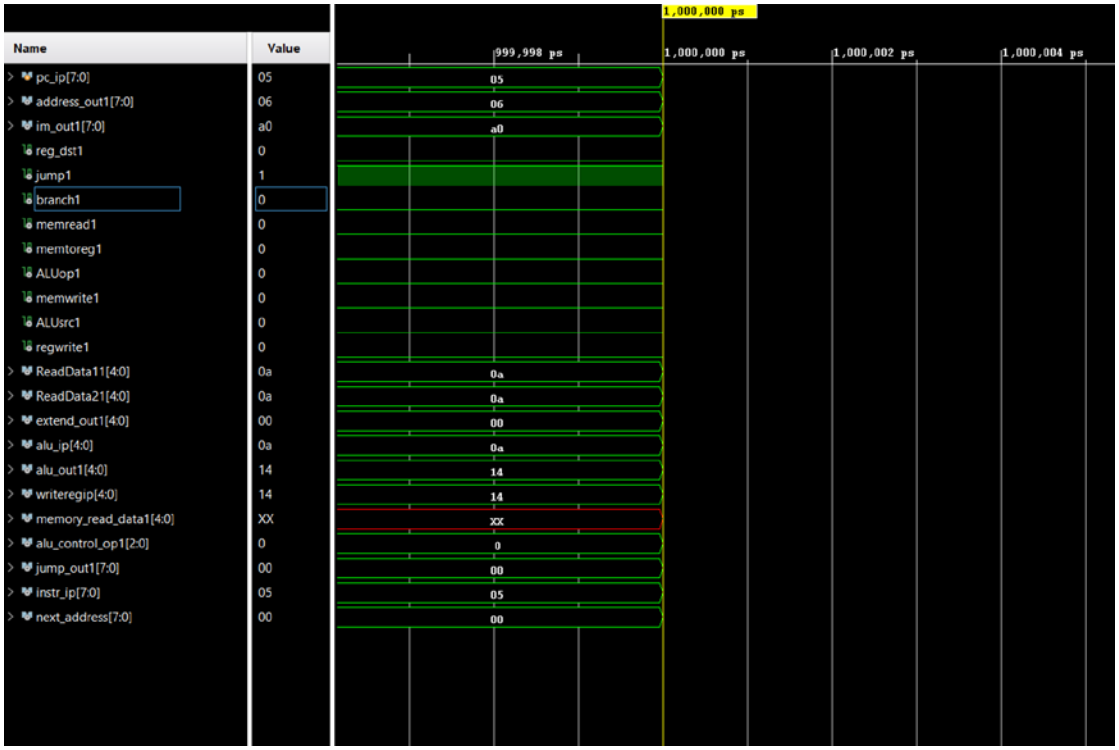
JMP Instruction:

Instruction - 8'b10100000



Jump Instruction:

Jmp 000



- The instruction executed is the jump 000. The result would be the selection of the next address. This address is specified in the immediate part of the jmp instruction.

References:

- MK Computer Organization and Design 5th Edition, David A. Patterson and John L. Hennessy.
- Computer Architecture, A Quantitative Approach, David A. Patterson and John L. Hennessy.
- https://www.xilinx.com/support/download.html
- https://reference.digilentinc.com/_media/reference/programmable-logic/basys-3/basys3_rm.pdf
- http://www.asic-world.com/verilog/veritut.html