**Lab 3 - Implementation of a MIPS like processor**

Design and implement (in Verilog) datapath and control unit for a single cycle MIPS like processor (including instruction memory) which has two classes of instructions. The two classes of instructions along with the example usage and instruction decoding to be used are as below

1. **Immediate Type**

Example: li r1, constant 🡪 loads immediate signed value specified in the instruction to the register R1

**Constant**

**Rdst**

**Opcode**

**6-bit Opcode 5-bit destination 21-bit**

1. **Register Type (R-type)**

Example: add r1, r2, r3 🡪 adds the contents of registers r2 and r3. The result of addition is written in to the register r1

**Function**

**Shamt**

**Rsrc2**

**Rsrc1**

**Rdst**

**Opcode**

**6-bit Opcode 5-bit 5-bit 5-bit 5-bit 6-bit**

**Destination Source1 Source2 Shift Amount**

Assume there are 32 32-bit general purpose registers indicated by r0, r1, r2…r31 and corresponding register numbers (00000), (00001)………..(11111).

Assume the Opcode for Immediate type and R-type instructions as below

|  |  |
| --- | --- |
| **Instruction Class** | **Opcode** |
| Immediate type | 111111 |
| Register Type | 000000 |

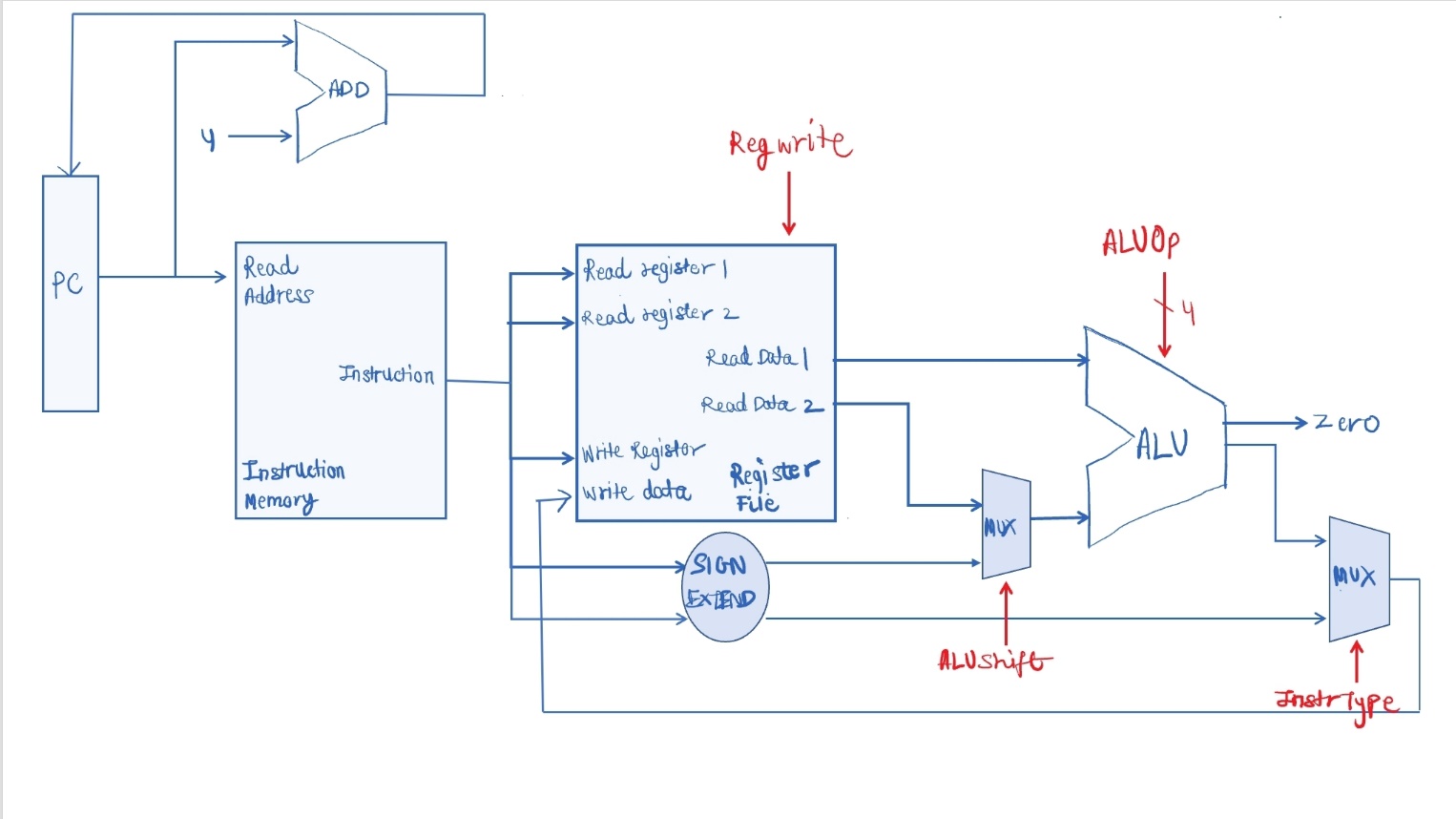
Additionally R-type instructions have multiple variations defined by their function codes. The R-type instructions should include **add**, **sub**, **AND**, **OR**, **srl** (Shift right logical), **sll** (shift left logical) .The different R-type instructions that the processor should support are tabulated below.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **R-type Instruction** | **Example usage** | **Opcode** | **Rdst** | **Rsrc1** | **Rsrc2** | **shamt** | **Function** |
| **add** | **add r0, r1, r2** | **000000** | **00000** | **00001** | **00010** | **00000** | **100000** |
| **sub** | **sub r4, r5, r6** | **000000** | **00100** | **00101** | **00110** | **00000** | **100010** |
| **AND** | **and r8, r9, r10** | **000000** | **01000** | **01001** | **01010** | **00000** | **100100** |
| **OR** | **and r9, r8, r10** | **000000** | **01001** | **01000** | **01010** | **00000** | **100101** |
| **sll** | **sll r11, r6, 6** | **000000** | **01011** | **00110** | **00000\*** | **00110** | **000000** |
| **srl** | **srl r13, r9, 10** | **000000** | **01101** | **01001** | **00000\*** | **01010** | **000010** |

\*Second source is not used for shift operations

The processor module should have only two inputs CLK and Reset. When Reset is activated the Processor starts executing instructions from 0th location of instruction memory.

1. **Draw the block level design of the processor (datapath + control unit) for above specifications. (you can modify the design given in the class ppts and copy the image of final design here)**

**Answer:**  

The red signals are the control signals generated by the Main Control unit. The signals are RegWrite, ALUShift, ALUOp, and InstrType.

1. **List the different blocks that will be required for implementation of datapath of the above processor.**

Answer: The different blocks that are required for datapath are:

* Instruction Memory
* Register File
* ALU
* Multiplexers
* Register Memory
* Adder
* Sign Extender

1. **Most of the datapath blocks that are listed above have already been implemented as part of previous labs. Implement the blocks which have not been implemented in the previous labs and copy the images of those Verilog codes here.**

Answer: The data path blocks that are being used are Instruction Memory/Fetch unit, the Register File unit, ALU unit, multiplexers and Sign Extension Unit. The Instruction, Register and ALU unit have already been implemented as part of previous labs. The codes of multiplexers and sign-extension units are of one line only, and hence have been included in the Processor module itself, and can be referred to from there.

1. **Assume Main control unit generates all the control signals. List different control signals that will be required for the above processor. Also specify the value of the control signals for different instructions.**

Answer:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Control Signal Name 🡪** | **ALUOp** | **InstrType** | **RegWrite** | **ALUShift** |
| **li r1, 8** | XXXX | 1 | 1 | 0 |
| **add r0, r1, r2** | 0010 | 0 | 1 | 0 |
| **sub r4, r5, r6** | 0100 | 0 | 1 | 0 |
| **and r8, r9, r10** | 0000 | 0 | 1 | 0 |
| **and r9, r8, r10** | 0000 | 0 | 1 | 0 |
| **sll r11, r6, 6** | 0011 | 0 | 1 | 1 |
| **srl r13, r9, 10** | 0111 | 0 | 1 | 1 |

1. **Implement the main control unit and copy the image of Verilog code of Main control unit here.**

**Answer:**  

1. **Implement complete processor in Verilog (using all the datapath blocks and main control unit as modules). Copy the image of Verilog code of the processor here.**

**Answer:** 

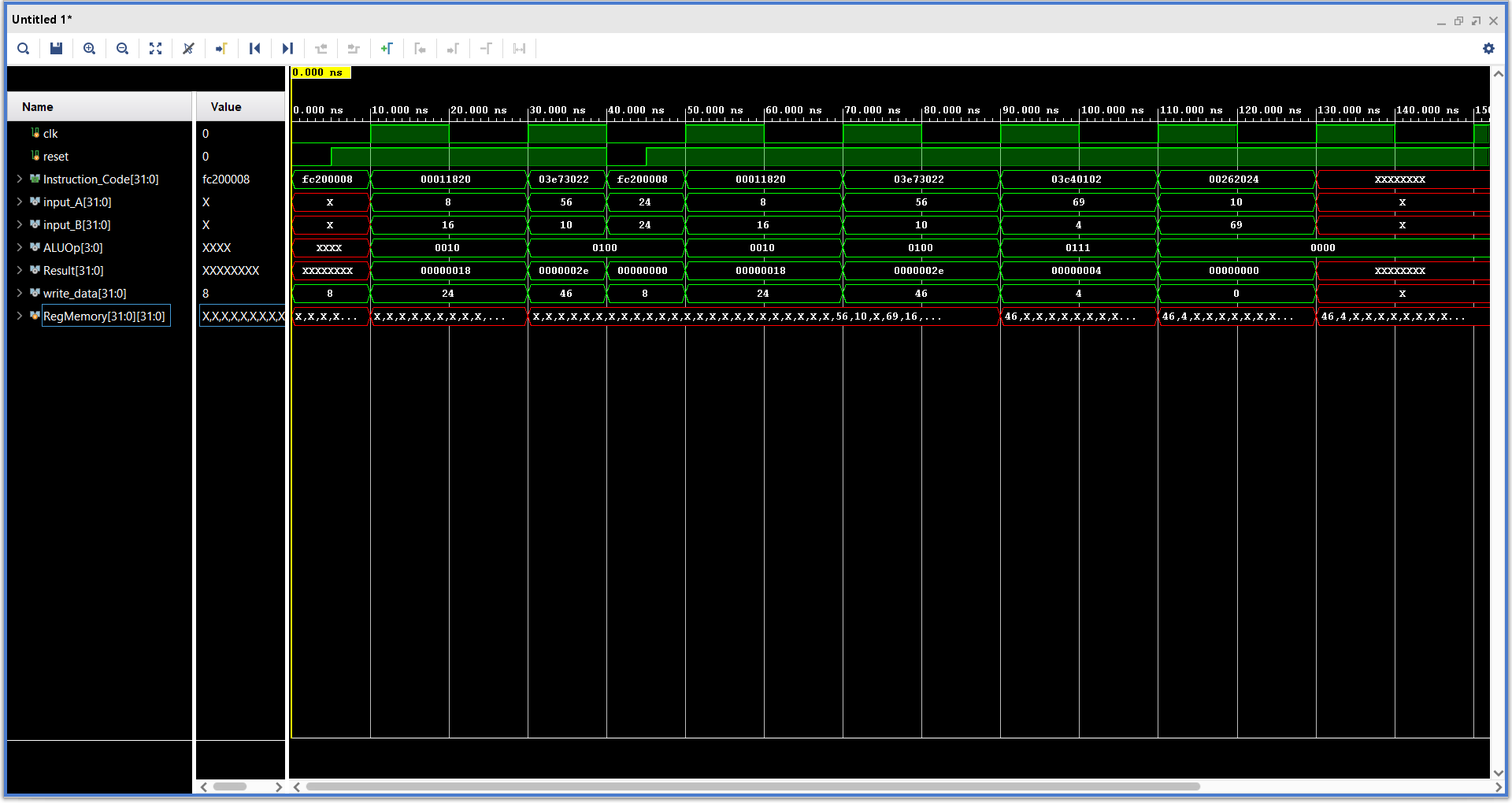
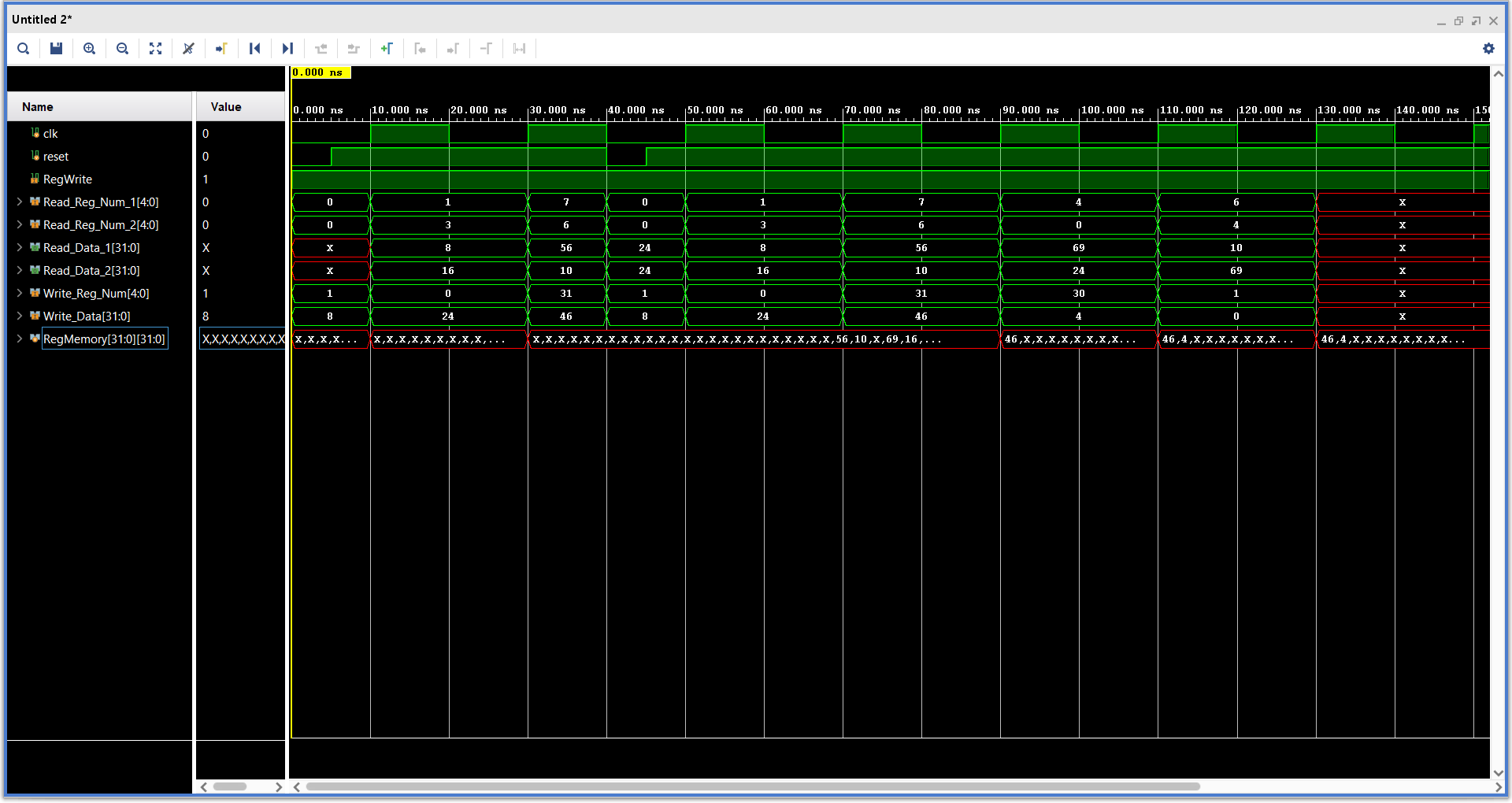
1. **Test the processor design by initializing the instruction memory with a set of instructions (at least 5 instructions). List below the instructions you have used to initialize the instruction memory. Verify if the register file is changing according to the instructions. (Register file contains unknowns, you can initialize the register file or you can load values into the register file using li instruction specified earlier).**

Answer:The instructions that I have used to initialize the instruction memory with are:

* li R1, 8
* add R0, R1, R3
* sub R31, R7, R6
* srl R30, R4, 4
* AND R1, R6, R4

It has been verified that the register file also changes according to changes in the instructions.

1. **Once design, test and verification are**

Copy verified **Register file** waveform here:

The first waveform window shows the ALU operations waveforms. The second one shows the verified Register File waveforms.