CSE490/590 COMPUTER ARCHITECTURE

Project - I

Instructor: Dr. Ramalingam Sridhar

April 14, 2021

DESIGN OF AN 8-BIT NON-PIPELINED PROCESSOR USING VERILOG

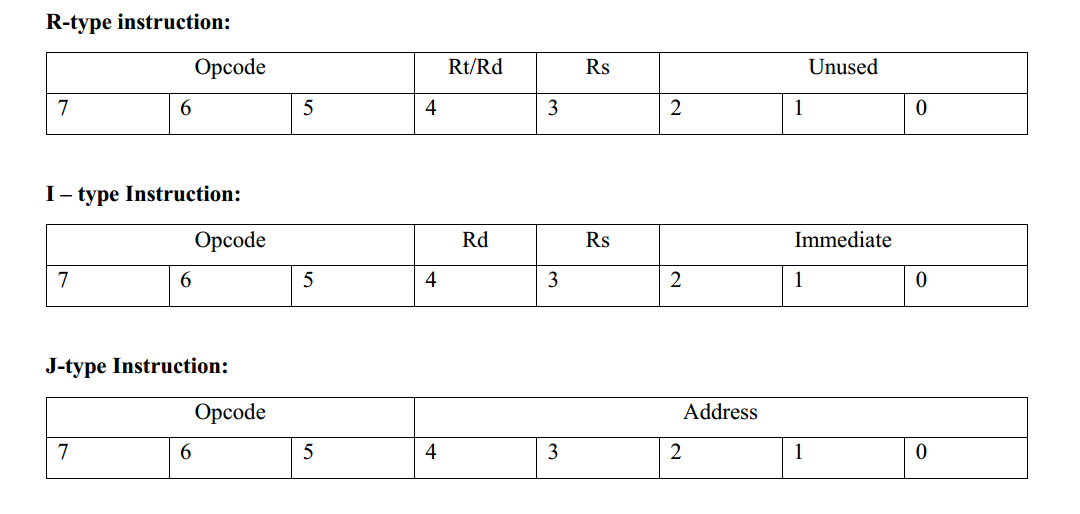
By

Manasa Challa – [manasasa@buffalo.edu](mailto:manasasa@buffalo.edu) (50356441)

Nishant Kapoor– [nkapoor2@buffalo.edu](mailto:nkapoor2@buffalo.edu) (50354536)

**Objectives**

Main objective of this project is to model and design an 8-bit microprocessor using Verilog HDL by using Structural Verilog modelling. The individual components were to be designed using behavioral modelling. The 8-bit instruction formats for R-type, I-type and J-type instructions are given below:

**Figure 1: Types of Instructions to be implemented**

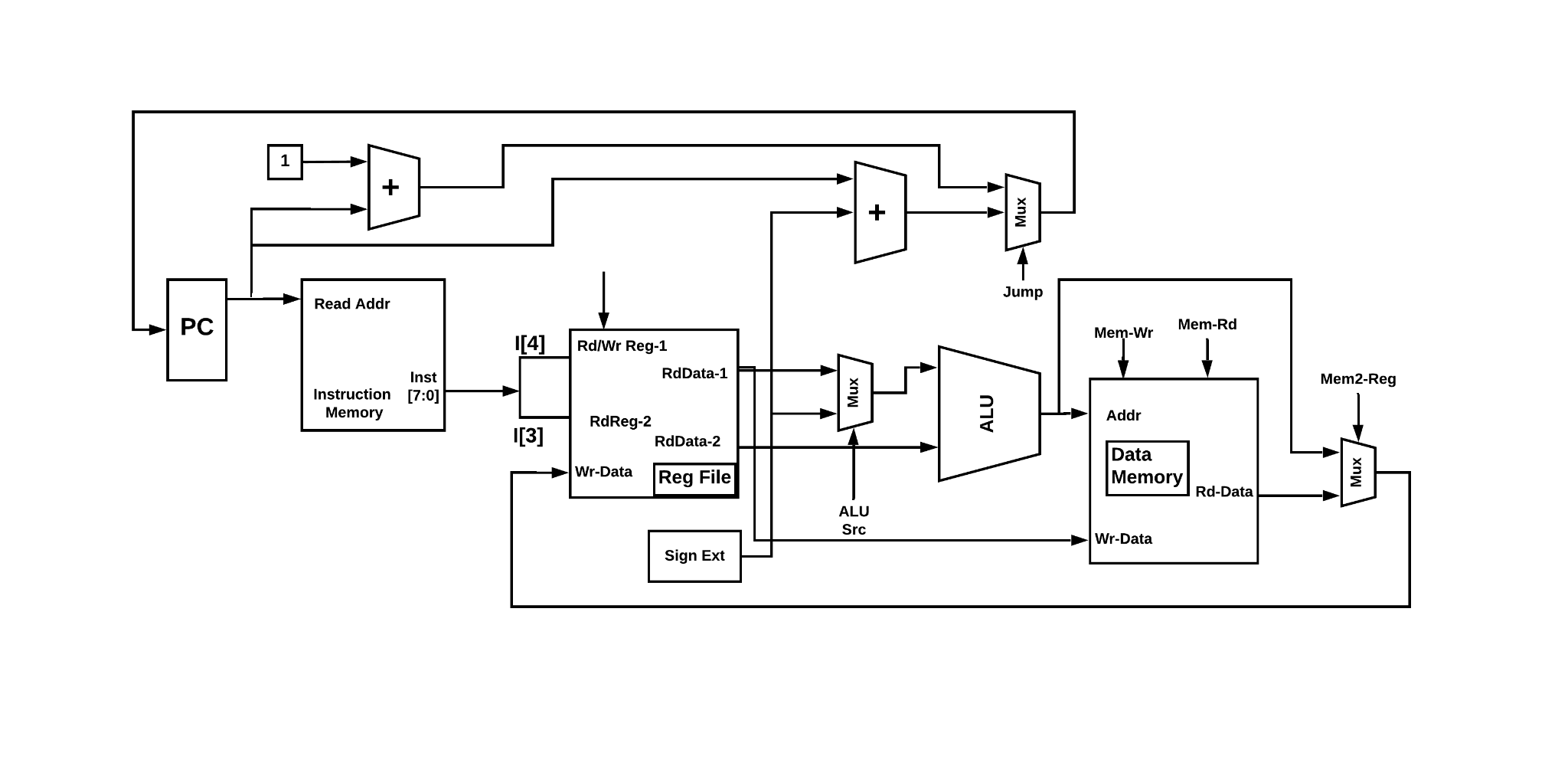
lw , sw, jmp, add, addi and sub instructions are implemented in our design.

**Design**

Design has the Following mandatory components.

**Data Path :-**

Following figure shows the datapath of our design :

****

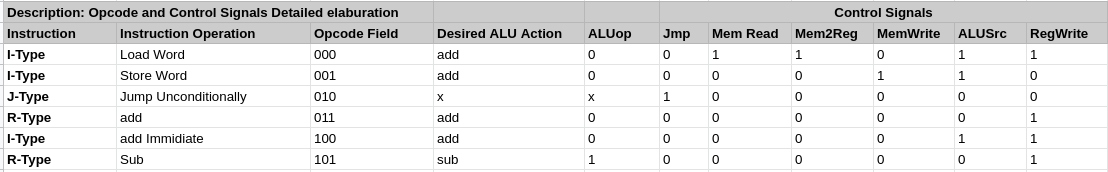
**Figure : Datapath and control signals**

**Instruction memory**

The instruction memory is used to store the instructions of programs that are executed on the processor. Instruction memory automatically jumps to the next instruction when execution of the current instruction is completed. Program counter is responsible for moving the index of instruction memory to the next instruction to be executed.

**Control Unit**

Control unit generates the control signal required for muxes, refile and data memory according to the given instruction. It takes the opcode of the instruction and generates the seven outputs, control signals and ALUOp. Control unit generates the following signals according to the opcodes, processor is processing at current time.



**Table 1 : Instructions, Opcodes and respective control signals**

**Register file**

Our processor has 2 × 8-bit registers ($to and $t1). At posedge of clock, if the write-enable of the regfile is high, input write data is written on the destination address register. Whenever there are read data addresses, data is read from the registers and placed on the read data ports. We have the same address for the first source register and destination register.

**Data memory**

The data memory is used to store the data. The data memory will have the following signals:

1. **Clock** : This will be the clock generated by the system.
2. **Write Enable / Read Enable :** If write enable signal is high, then we will be able to write data at the positive edge of the clock. If the read enable signal is high, then we will be able to read data at the positive edge of the clock. These will be single bit signals.
3. **Address Port** : This will be the 8-bits signal where we will provide the address of the memory from where we want to read or write data.
4. **Write data / Read data port** : Write data is an 8-bit input port that will be used to place the data present on it into the memory. Read data is an 8-bit output port that will be used to get the data from the memory.

**ALU (Arithmetic Logic Unit)**

The ALU will be able to perform the following operation:

1. Addition
2. Subtraction

ALUop signal input will be generated from the main controller that consists of purely combinational logic. On the basis of this signal, the ALU performs either the addition or subtraction.

**Program Counter (PC)**

The program counter, PC, is a special-purpose register that is used by the processor to hold the address of the next instruction to be executed. The logic implemented automatically updates the PC to point to the next instruction. The PC can also have an address of jump target address calculated with the pc and immediate value if the jmp control signal is asserted.

**Sign Extender**

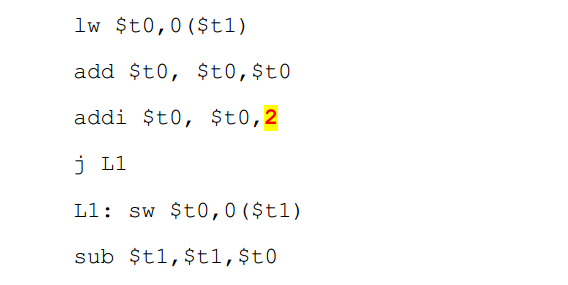
This logic will be present here to sign extend the immediate value that will be provided in the instruction, before sending to the ALU for any arithmetic operation being performed on it.

**Multiplexers**

The design has three muxes in it. They are there to route the data depending upon the control signals.

**Test Plan**

In order to test the functionality of our Design, 8-bit microprocessor, we need to check all six instructions: lw , sw, jmp, add, addi and sub implemented in our design to give expected results. The detailed test plan is given below. For that, we have run the sample program given in requirements.



**Figure : Test Program**

**Test Plan Implementation:**

To implement the following test plan we have converted the program into binary code according to the opcodes and different fields and then executed it. Figure below shows the binary converted instruction stored in memory:

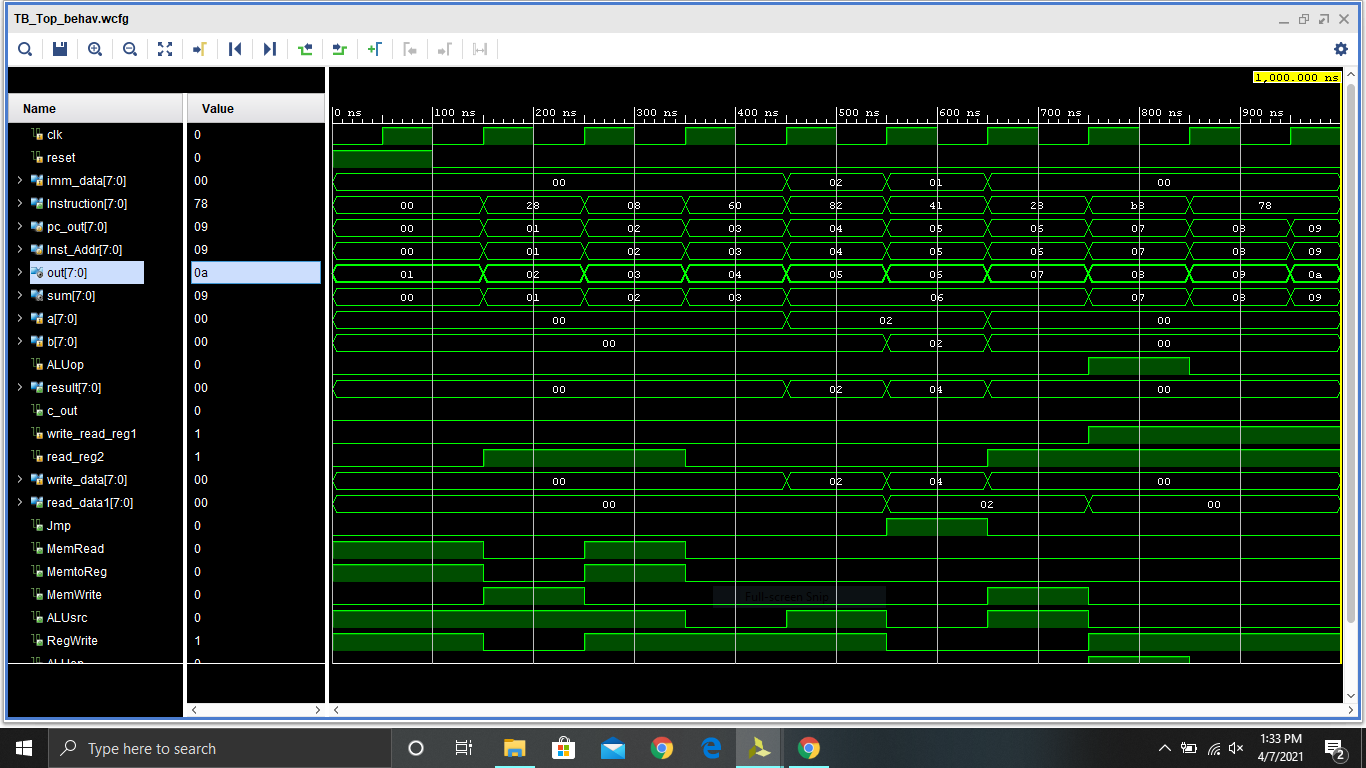
Graphical user interface, text, application

Description automatically generated

**Figure : Instruction Memory Snippet**

**Simulation Results**

After running the above program on our design we synthesis the design, ran the simulation and got the results shown in the screen shots from Vivado attached below:



**Figure : Simulation Results**