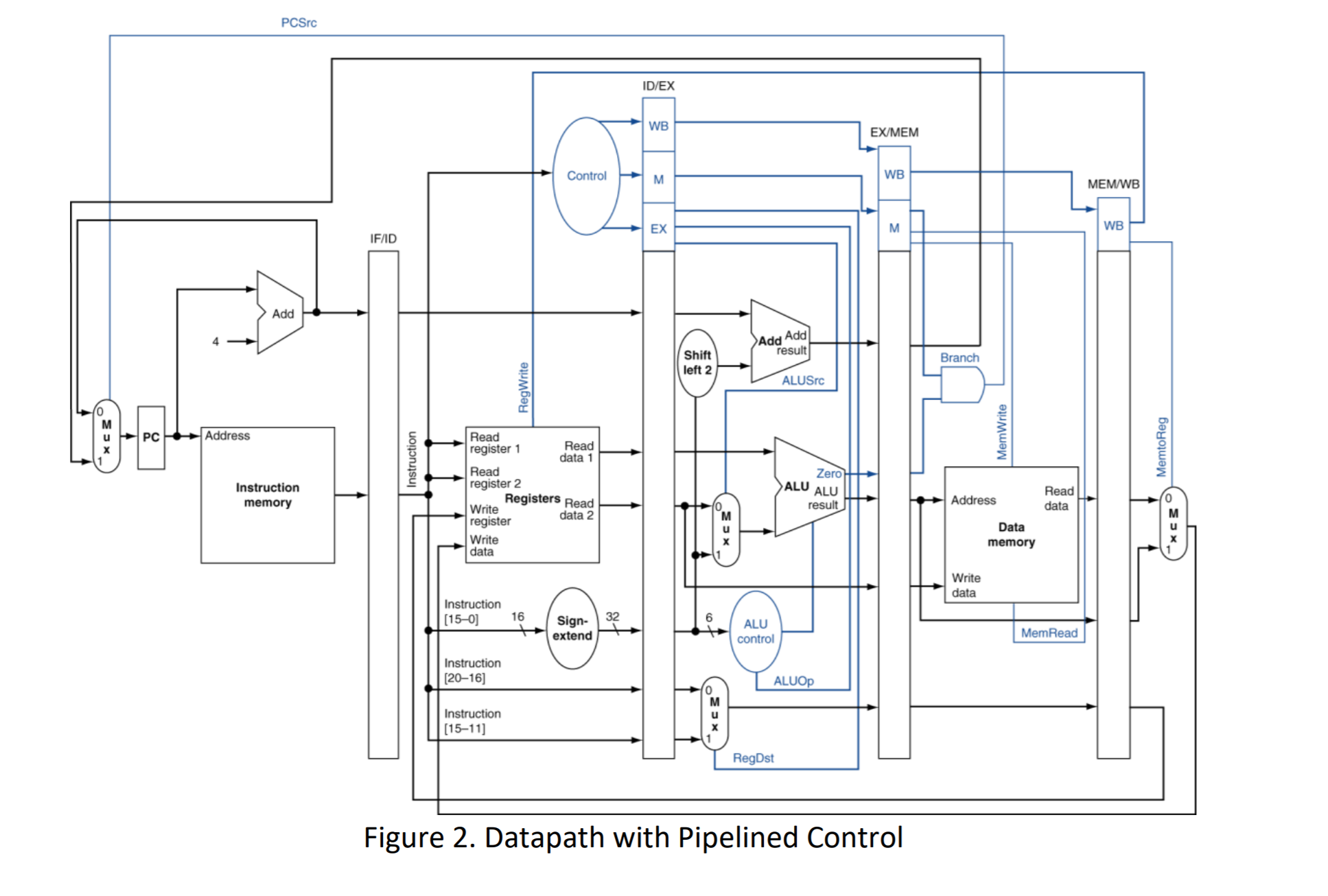
Project 2: 32-bit RISC Processor



Acknowledgement: I acknowledge all works including figures, codes and writings belong to

me and/or persons who are referenced. I understand if any similarity in the code, comments, customized program behavior, report writings and/or figures are found, both the helper (original work) and the requestor (duplicated/modified work) will be called for academic disciplinary action.

Signature: 

Abstract: The purpose of this project is to gain first hand experience in hardware design and the challenges that come with computer architecture.

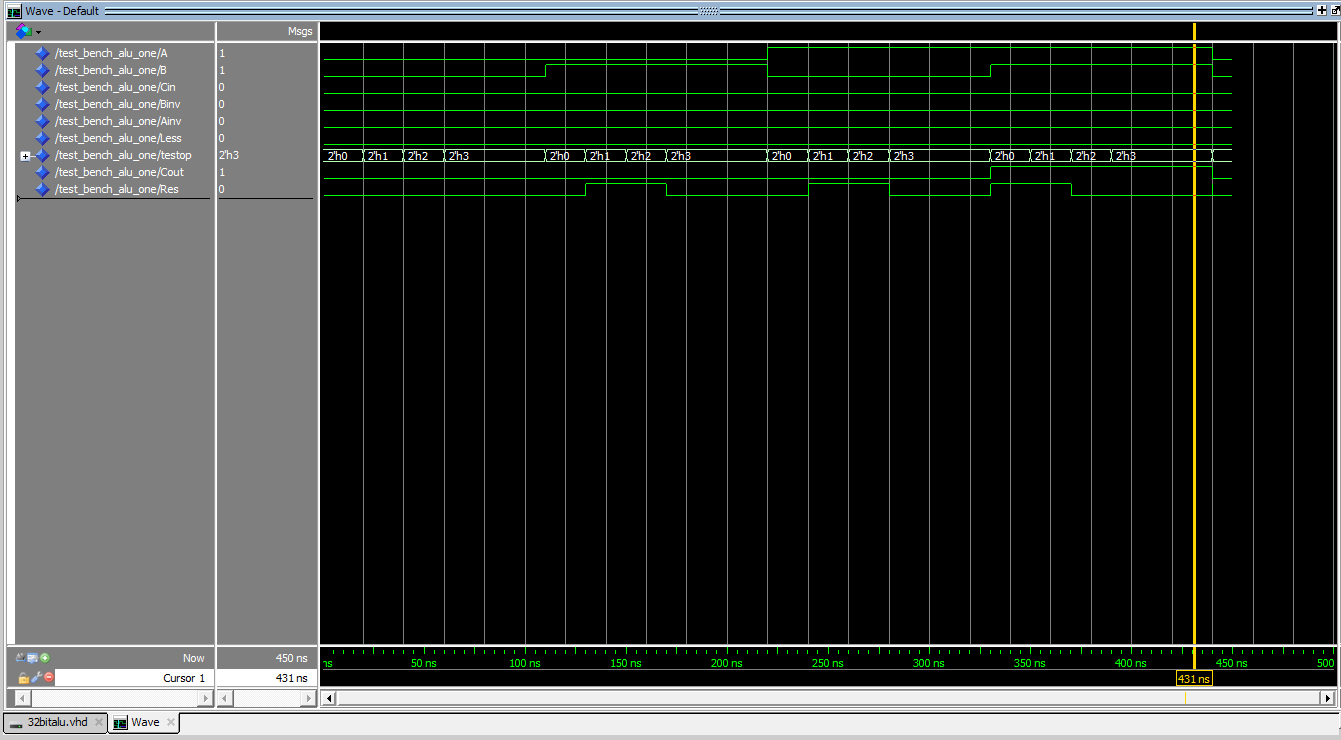
Introduction: This goal of this project is to design a 32-bit RISC processor with pipelining. This processor will be limited in functionality, but serves as a good way to gain experience with both VHDL and hardware design. The design elements for this project include a datapath and a control interface to complete a processing unit. The 32-bit adder design will be used in the creation of the processors ALU.

Background: The main design elements for this project are the datapath elements and the control interface. Hardware that needs to be created for the datapath include: PC, instruction memory, register file, ALU, and data memory. The PC and instruction memory will be used for instruction fetch, the register file will be used for instruction decode as well as write back, the ALU will be used for execution, and data memory will be used for memory access (reads/writes). Hardware that needs to be created for the control interface include: a main control unit, and an ALU control unit. The control unit is an element that will take the opcode of an instruction as input and will determine the signals needed for each stage of the datapath. It will also send a signal to the ALU control unit to tell the ALU what operation it needs to execute (and, or, add, etc...). For the control interface, the main control unit will produce several different control signals that need to be appropriately forward to the various datapath elements. This will be done by sending the signals needed to a register in between stages that can hold the control signals needed in the next stage until a clock signal is provided to forward needed signals to the next stage. This will be done with D-Flip Flops since they provide the ability to hold signals until a clock edge is triggered, then the previous signal will be forwarded to the next stage.

Results: Working components of this project include: 32-bit ALU, PC, Instruction Memory Unit, Data Memory Unit, Register File, Sign Extension Unit, ALU Control Unit, and Control Unit.

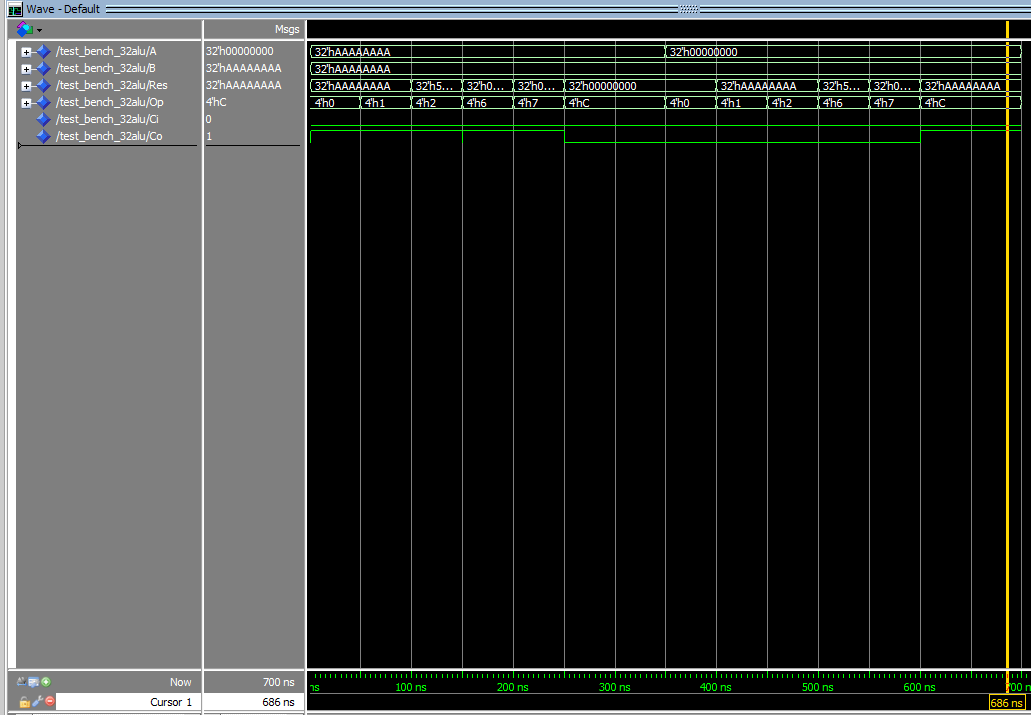
\*Note all graphs are uploaded to blackboard in full resolution

1-bit ALU



The 1-bit ALU was tested by using two different values for A and B inputs and evaluating all the different combinations of operations that it can handle.

32-bit ALU



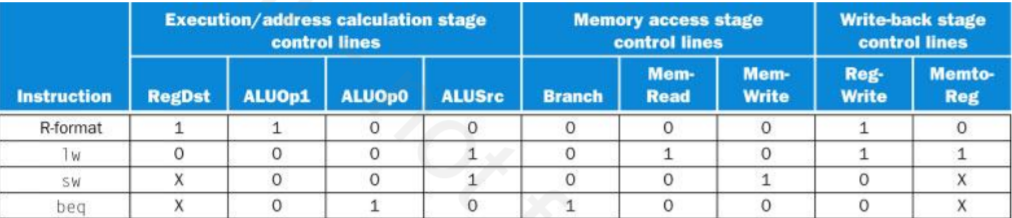
The 32-bit ALU was tested by using two different values for A and B and evaluating all combinations of the ALUop signal. This shows that all functionality works correctly even for SLT and Branch logic, which both utilize the subtraction logic.

ALU Control Unit

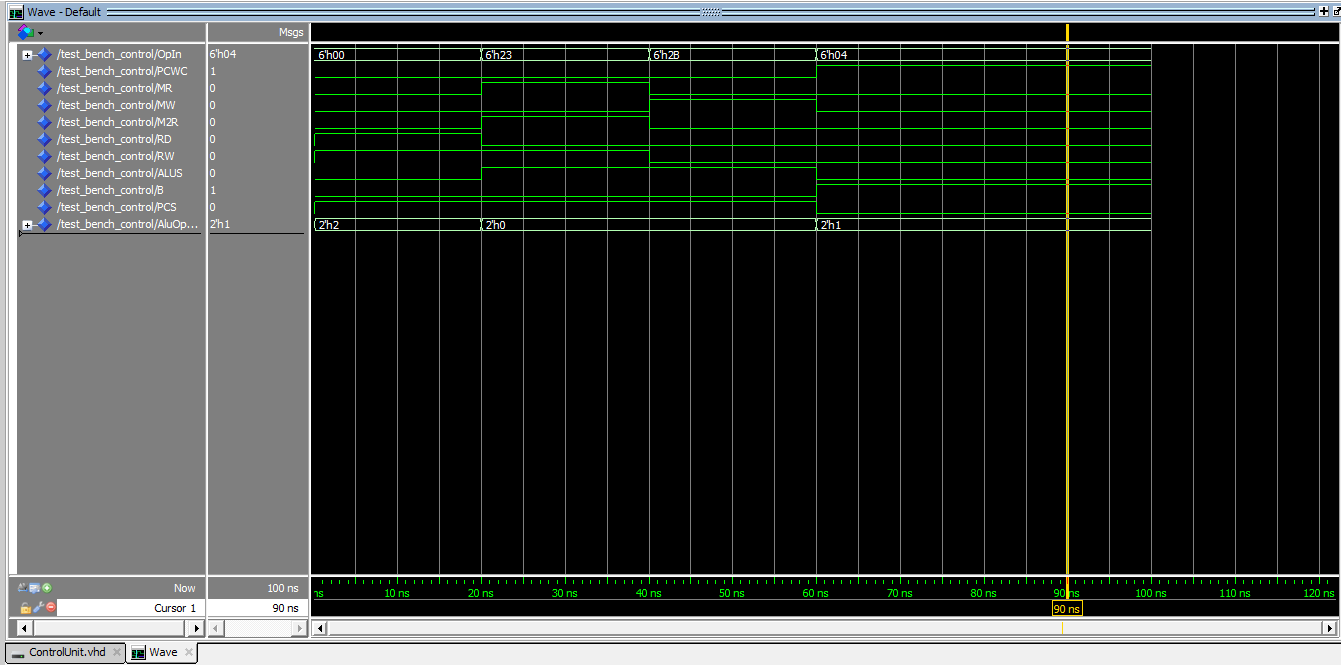


The ALU Control Unit was tested by evaluating all values of the instruction function codes that were given for project two and different values of ALUop signals from the main control unit. As seen, the ALU Control Unit correctly sends out the corresponding ALU control signals.

Control Unit

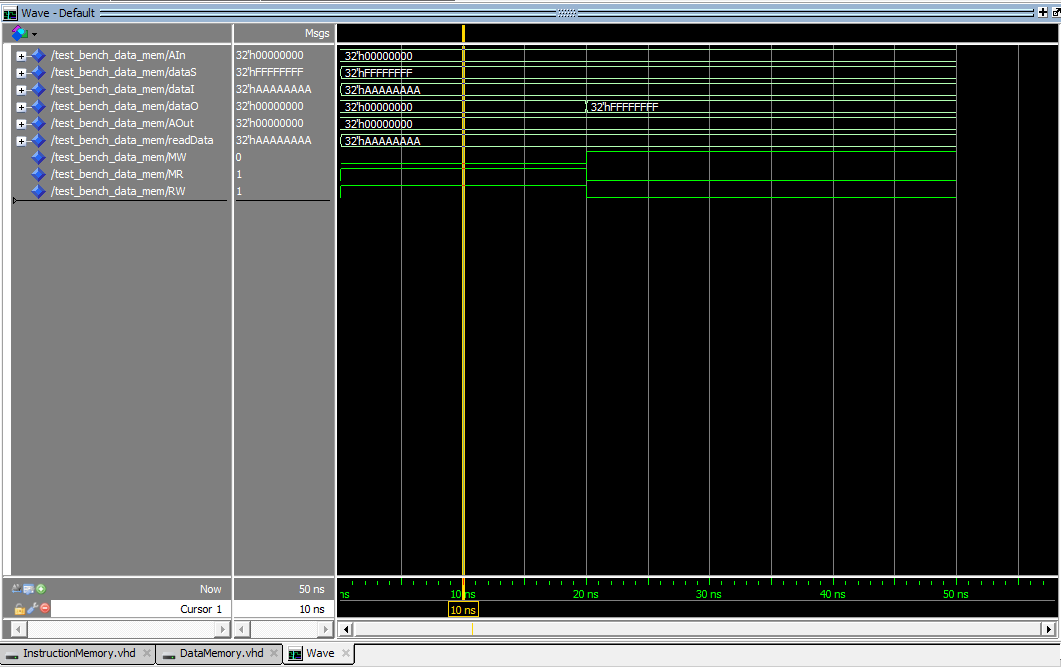


The logic for the control unit was based on this table of control signaled for a pipelined processor provided in the lecture 10 powerpoint.



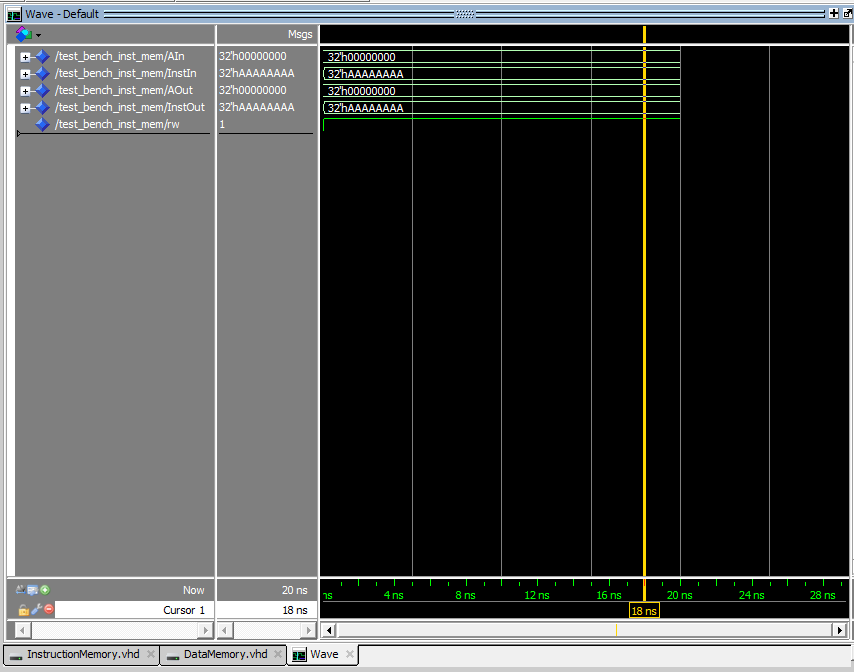
The Control Unit was tested by evaluating Opcodes for R, Store, Write, and Branch type instructions and their output signals verified.

Data Memory Unit:



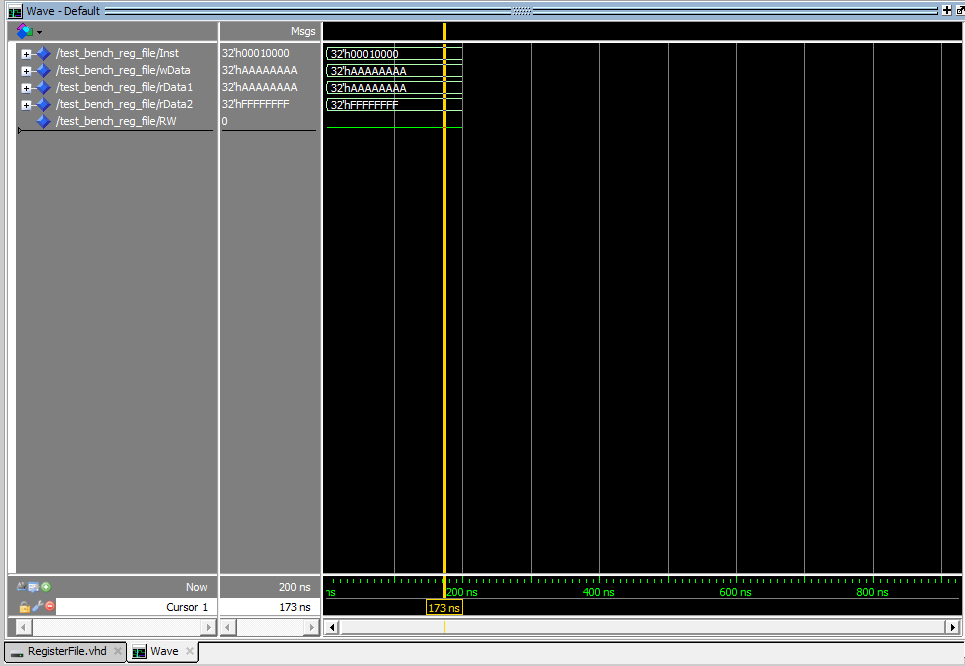
The data memory unit was tested by setting an input address, data source, and memory r/w signals. Since I did not create memory, the unit outputs a r/w signal, address, and write data for placement on a memory bus. From this, another input signal was data in from memory. In this test data at Ox0 in memory was OxAAAAAAAA. From the waveform it looks to be working correctly.

Instruction Memory Unit:



The instruction memory unit was tested by evaluating a test address Ox0 and test input instruction if there was memory, it also provides a r/w signal that is always set to ‘1’. The waveform shows a corresponding output.

Register File:



The register was tested for read and write register signals. Registers 0 and 3 were set to OxAAAAAAAA and Register 2 was set to OxFFFFFFFF. This was done in the register file VHDL file as an initial condition for testing. The testing was done with an instruction of Ox00010000 and write data of OxAAAAAAAA. Most of the features of this unit are working but I had to use constants for register numbers in testing because there were complications in trying to convert a bit vector input for register number to an array index.

Challenges: Some of the challenges I faced in project 2 include difficulty creating a working control interface. The units that involved combinational logic were intuitive to create, but when trying to tie all the components together to form a working CPU, I could not get the state elements working. Unfortunately my project is not functional as a CPU, I have included plans of how I would have completed the control in the background section of the report. Improvements I would implement if I could restart from scratch would be to create

a diagram of all the state elements that are involved between stages in the pipeline. For example, it would have helped my progress greatly if I created a diagram of all the registers and flip flops needed to hold the control signals at each stage.

Conclusion: This goal of this project was to gain hands on experience in hardware design and in that regard it has succeeded. The main shortcoming of my project is that I couldn’t get the control signals to propagate through the stages. While it was not possible to create a working CPU, each component was tested for functionality through using hierarchical design. For the ALU creation, a 1-bit ALU was created and tested for correct functionality. Then it was used in turn to create the 32-bit ALU. This made it easier to diagnose problems in the 32-bit ALU’s operation. This same principle was used in the other components, especially the low level components like 1 and 2 bit multiplexers.

List of References:

1. VHDL Tutorial by Peter J. Ashenden
   1. This tutorial was used to reference VLDL syntax and programming techniques.
2. ECE 485 Lecture 6-10 Slides by Dr. Won-Jae Yi
   1. The slides were referenced for the creation of the ALU design as well as the control interface.
3. Computer Organization and Design by David A. Patterson and John L. Hennessy
   1. Chapter 4 of the book on single cycle and multi cycle datapaths and control signals was utilized in the design of my control logic and datapath elements.

Appendix:

Source code attached as vhd files.