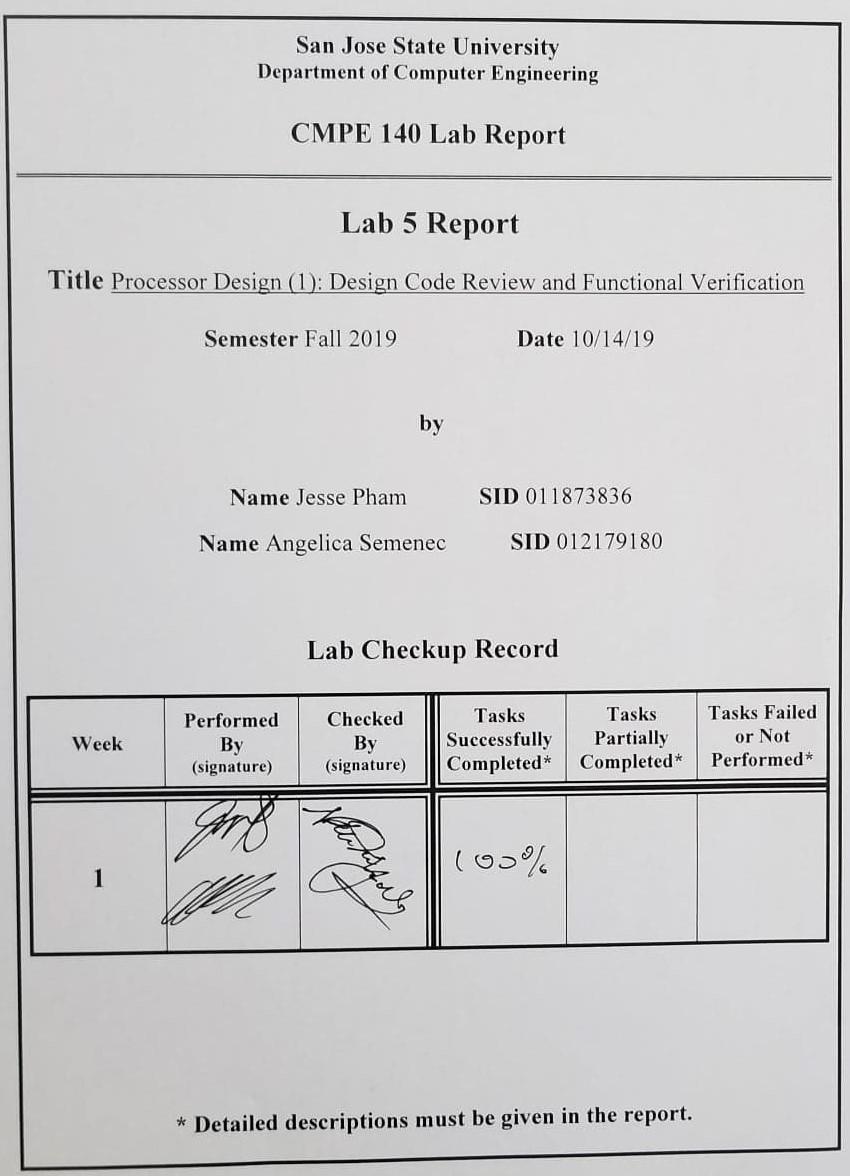
**Purpose**

The purpose of this lab is to gain experience with processor design by reviewing the RTL Verilog code to understand the single-cycle MIPS processor. A testbench for verifying the functionality of the processor is also included and will be used to learn basic techniques for functionally verifying the processor.

**Design Methodology**

The Verilog code containing all of the files making up the datapath, control unit, memory units, processor core, and complete processor as well as the testbench for functional verification were all provided. To understand the structure of the single-cycle MIPS processor, diagrams were created for each underlying segment of the architecture including the control unit, data path, memory units, processor core and complete processor. These diagrams show the interconnections that make up each segment and effectively show how the entire system works as a whole. The testbench that was provided shows the resulting waveforms generated from the program, which is populated with the hex value of MIPS instructions that was also provided.

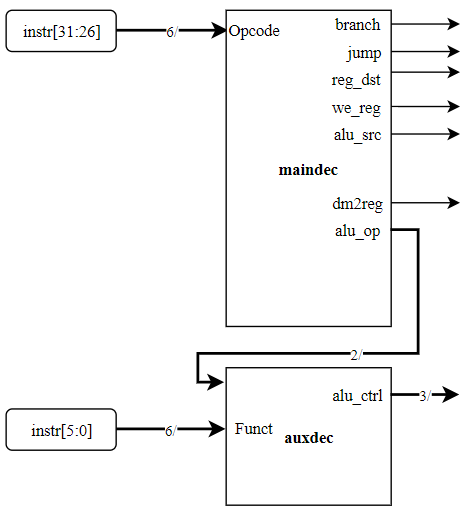
The machine code instructions from the MIPS program is provided in the file, “memfile.dat,” and was used for the *instruction memory* module to provide the Control Unit with the proper opcode and function if needed. The Control Unit provides the Datapath with the appropriate control signals to execute the instructions. These control signals are shown in *Figure 2* below, indicating where each signal is going in the Datapath as well as how they are being used by each module within the path. Using the testbench to construct the waveform simulation, the design’s functionality was verified by looking at the *we\_dm* instruction to locate when the store word instruction has been executed. After reading from memory, if the value that was stored previously is being outputted, then its functionality is working properly.

**Tasks Successfully Accomplished**

1. Created diagrams for the control unit, datapath, memory units, processor core, and complete processor.
2. Functionally verified the single-cycle MIPS processor using the provided testbench.

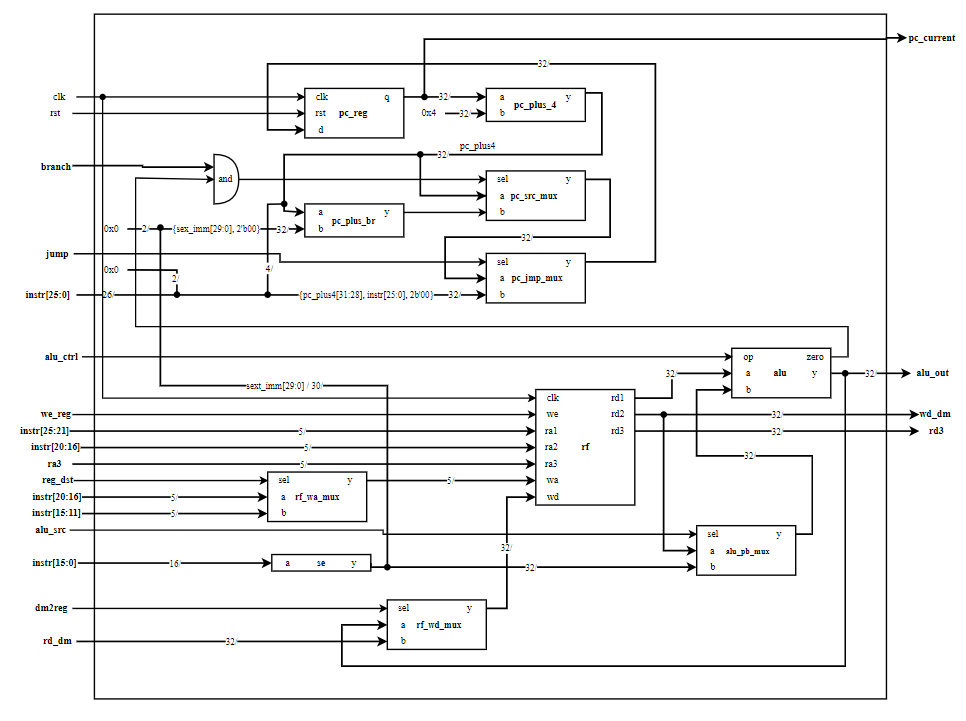
**Diagrams**

**Control Unit**

**

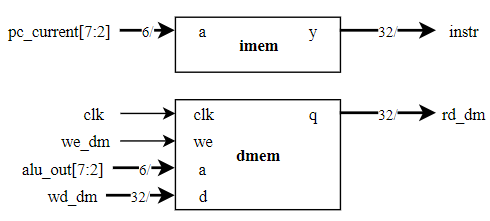
*Figure 1: The block diagram for the Control Unit. The opcode gets fed into the main decoder (maindec) to send the appropriate control signals to the Datapath Unit. The opcode will come from the last six bits from the imem module while the function input will be provided by the last six bits from the imem module.*

**Datapath**

**

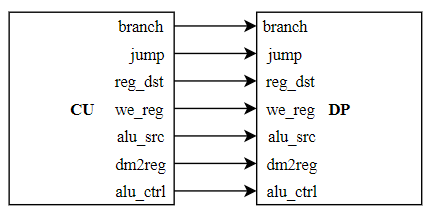
*Figure 2: The block diagram for the Datapath Unit which obtains its control signals from the Control Unit in the figure above. This module will handle the computations for the Program Counter logic, the Register File logic, the ALU logic, and the Memory logic.*

**Memory Units**

**

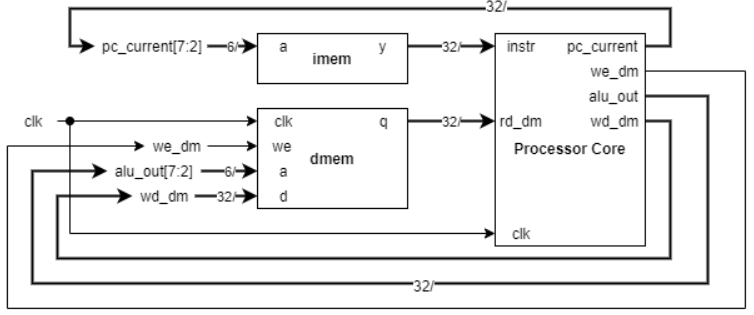
*Figure 3: Displaying the block diagram for the Instruction Memory (imem) and Data Memory (dmem). The imem module handles providing the Control Unit the appropriate instructions and the dmem module handles providing the appropriate values from the specified memory addresses*

**Processor Core**

**

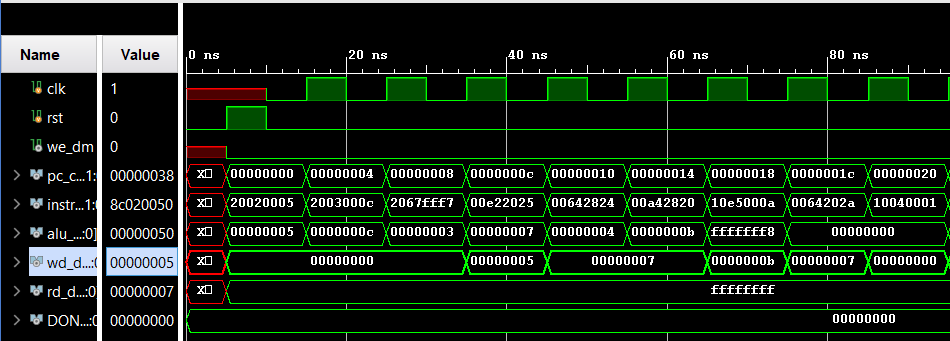
*Figure 4: The Processor Core diagram which conveys how the Control Unit (CU) is connected to the Datapath Unit (DP).*

**Complete Processor**

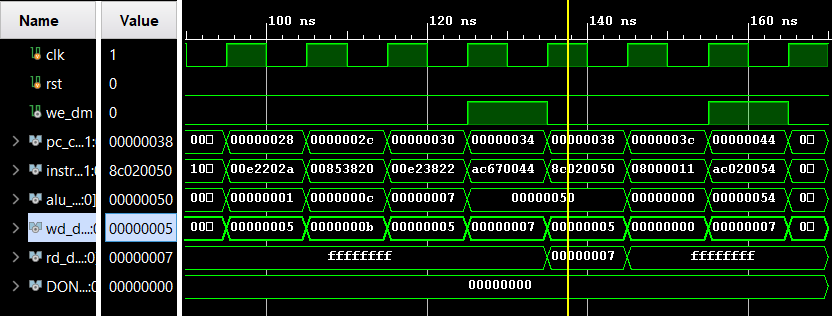
****

*Figure 5: Displays the Complete Processor Diagram which shows how each module interacts with one another. The Processor Core contains both the Control Unit and the Datapath Unit, the imem module contains the instructions, and the dmem module contains the data in memory.*

**Waveform**

****

*Figure 6: Showing the first half of the waveform simulation after running the testbench file, which captured the signals from from 0ns to 100ns.*

****

*Figure 7: Showing the second half of the waveform simulation after running the testbench file from 100ns to over 160ns. The data being read when program counter is 0x00000038 contains the value 710.*

**Discussion/Conclusion**

The lab experiment was successful in translating the MIPS instructions into machine code to be used in the simulated Control Unit and Datapath Unit for execution. The MIPS program provided in the lab manual created Machine Code instructions as hexadecimal values which were contained in a file called “memfile.dat.” This file was used as an input to our *imem* module and had its output instructions extracted into the Processor Core. We looked at how the store word (*sw*) MIPS instruction worked in this design and can verify its functionality using the waveform simulation. In *Figure 7* above, we can see that the signal *we\_dm* for write enable data memory goes HIGH to store the value 0x00000007 shown in signal *wd\_dm* for write data data memory. After it finishes its writing instruction, our test bench reads the value from the memory location and outputs the value 0x00000007, indicating that our store word instruction works as intended. Initially, the signals all had X’s and were marked red because the *rst* signal has not been activated. Before the *rst* signal, program counter will be at an arbitrary value, which will not allow the *imem* module to fetch the appropriate instructions. Once the *rst* signal is activated, the program counter is reset to 0 and will allow other signals to be properly initialized.

This lab provided experience in processor design using RTL Verilog design code. By creating the block diagrams for each module, we were able to gain a better understanding of their functionalities and how each module connects to one another to form the complete processor that can execute the proper MIPS instructions. The waveform simulation provides a great way to functionally verify the design and visually see the inputs and outputs.