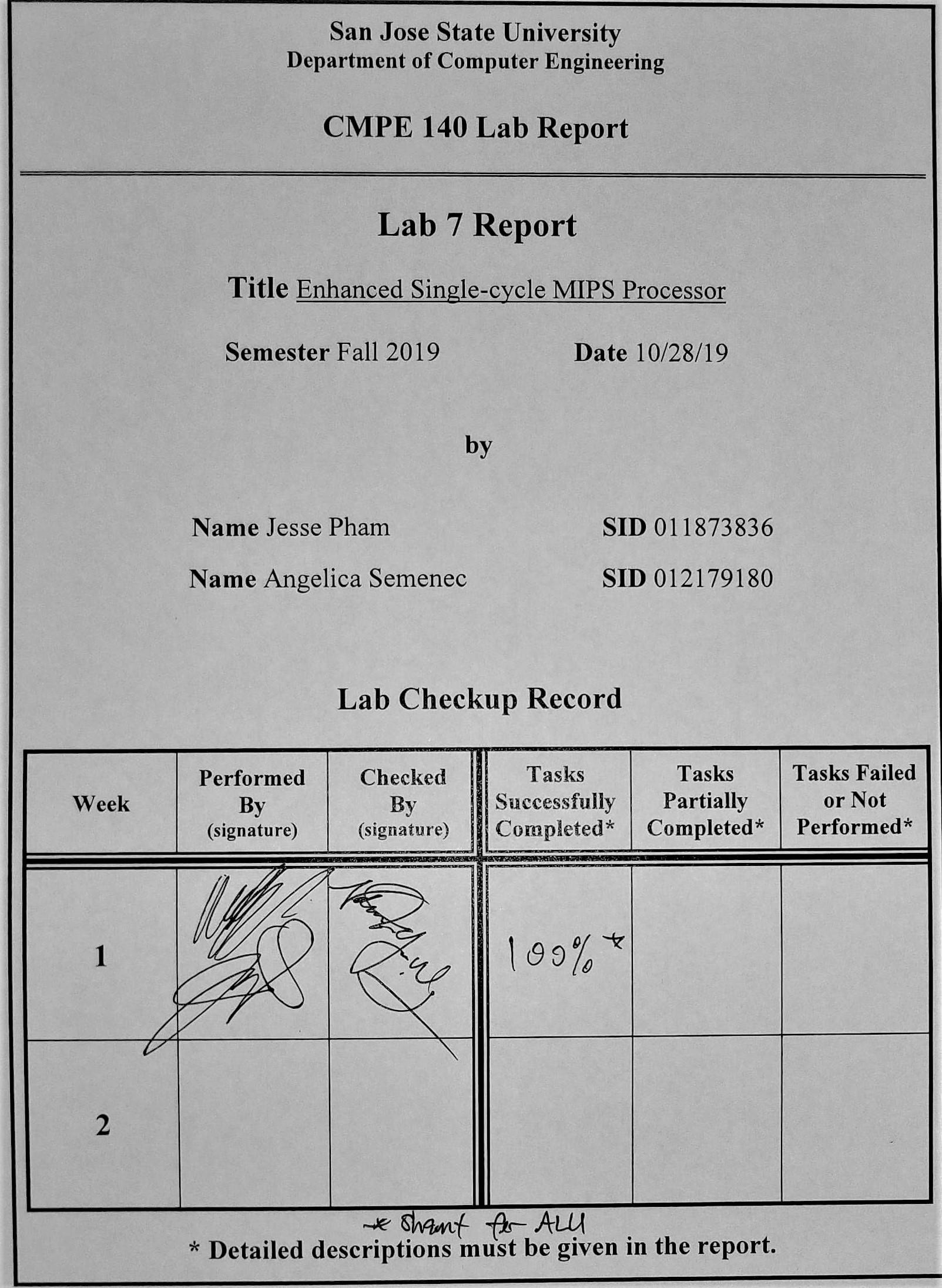
**Purpose**

The purpose of this lab is to extend the design of the single-cycle MIPS processor from Lab 05 and 06 to incorporate additional MIPS instructions. The instructions to be added are *MULTU, MFHI, MFLO, JR, JAL, SLL,* and *SLR*. This will provide students experience in developing an extended MIPS processor and to also reinforce lessons in the lecture.

**Design Methodology**

Examining the source code for the Verilog implementation of the main decoder and the auxiliary decoder, a Truth Table was created in order to keep track of what control signals are being sent which is shown in Table 01 and Table 02 below. The R-Type instructions (*MULTU, MFHI, MFLO, JR, SLL,* and *SRL*) only requires control signals *reg\_dst* for register destination and *we\_reg* to enable the register for a writing operation. The *alu\_op* control signal is 0x2 because in the auxiliary decoder, a value of 0x0 is to add while 0x1 is to subtract, however, the new instructions will be neither and will require a different functionality. These functionalities are unique and will therefore have unique control signals for the output of the auxiliary decoder (*alu\_ctrl*). The *alu\_ctrl* signal was initially 3 bits, however, our design has increased it to 4 bits to be able to provide each instruction with a different alu\_ctrl value. The *JAL* instruction is similar to the *addi* instruction since it essentially adds 4 to the program counter and stores it into the return address. This instruction will also need to activate the *JUMP* control signal as well as the *LINK* control signal.

The MIPS program was provided within the lab manual and was assembled to obtain the machine code. The memory configuration was configured to have Data Segment begin at address 0x00. The machine code was then pasted into the *memfile.dat* to be used as the input for the design.

**Tasks Successfully Completed**

* Completed the Control Unit's Truth Table
* Drafted an extended MIPS microarchitecture for both the Datapath and Control Unit

**Control Unit Test Table**

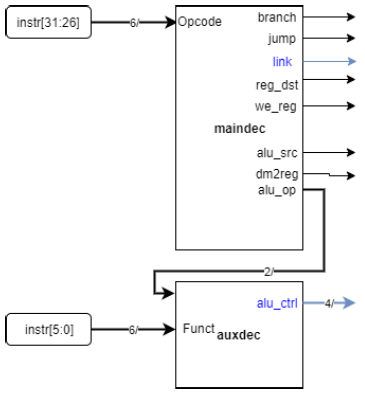
Table 01: Displaying the Truth Table for the Main Decoder. For the R-Type instruction, this includes the required instructions MULTU, MFHI, MFLO, JR, SLL, and SRL. The JAL instruction will use the newly implemented *link* control signal to enable the PC + 4 to register as well as the address $ra.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **maindec** | | | | | | | | | | |
|  | Input | Output |  |  |  |  |  |  |  |  |
| **Instruction** | **opcode** | **branch** | **jump** | **link** | **reg\_dst** | **we\_reg** | **alu\_src** | **we\_dm** | **dm2reg** | **alu\_op[1:0]** |
| R-Type\* | 0x00 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0x2 |
| addi | 0x08 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0x0 |
| beq | 0x04 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0x1 |
| j | 0x02 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0x0 |
| sw | 0x2b | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0x0 |
| lw | 0x23 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0x0 |
| jal\*\* | 0x03 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0x0 |

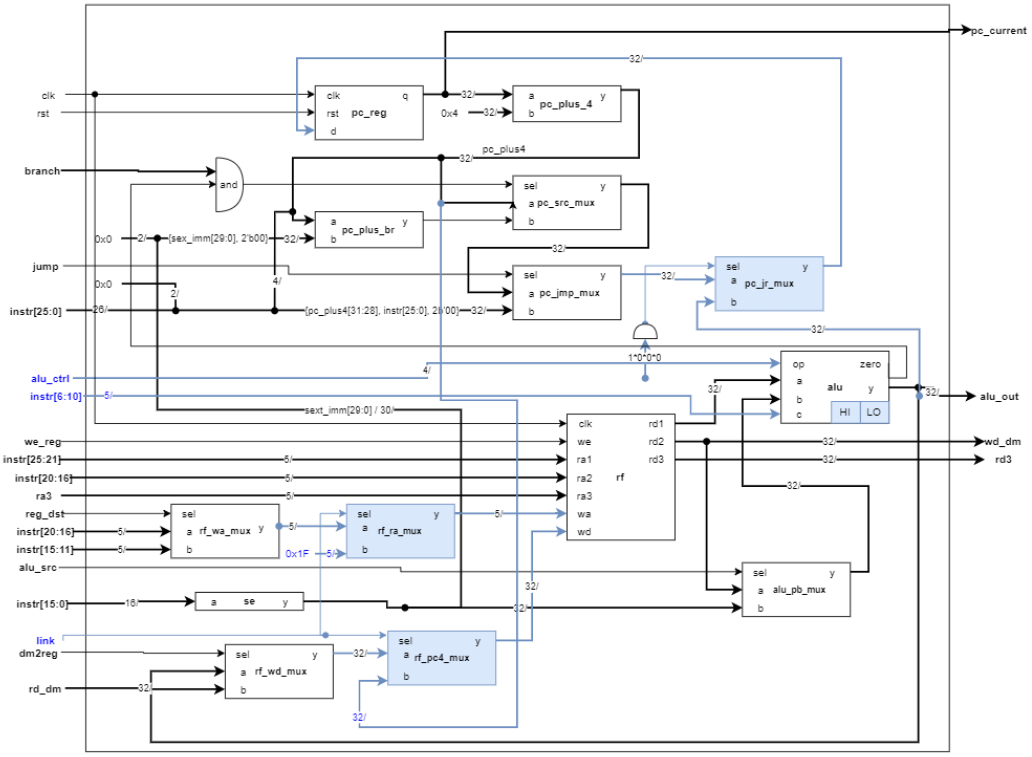
Table 02: Displaying the Truth Table for the Auxiliary Decoder. In this design, the *alu\_ctrl* signal will be four bits rather than three. Each of the R-Type instructions will have a unique *alu\_ctrl* signal. The JAL instruction will be similar to the *addi* instruction since it simply adds the value 4 to the program counter and stores it in $ra.

|  |  |  |  |
| --- | --- | --- | --- |
| **auxdec** | | | |
|  | Input |  | Output |
| **Instruction** | **alu\_op[1:0]** | **funct[5:0]** | **alu\_ctrl[3:0]** |
| addi | 0x0 | x | 0x2 |
| beq | 0x1 | x | 0x6 |
| j | 0x0 | x | 0x2 |
| sw | 0x0 | x | 0x2 |
| lw | 0x0 | x | 0x2 |
| multu | 0x2 | 0x19 | 0x3 |
| mfhi | 0x2 | 0x10 | 0x4 |
| mflo | 0x2 | 0x12 | 0x5 |
| jr | 0x2 | 0x08 | 0x8 |
| sll | 0x2 | 0x00 | 0x9 |
| srl | 0x2 | 0x02 | 0xa |
| jal | 0x0 | x | 0x2 |

**Diagrams**

****

*Figure 01: The newly designed control signals, link and alu\_ctrl, are colored blue. The only difference is the addition of a link signal and the addition of one bit to the alu\_ctrl signal to provide unique controls for the new R-Type instructions.*

**

*Figure 02: Displaying the new extended MIPS design. The signals alu\_ctrl, instr[6:10], and link are different control signals being introduced to the original datapath. Additionally, new modules are also added into the datapath, such as rf\_ra\_mux to store the PC in $ra when JAL is called, rf\_pc4\_mux, and pc\_jr\_mux. HI and LO registers have also been implemented in the ALU module to hold the upper and lower bits for instructions MFHI and MFLO.*

**Source Code**

|  |
| --- |
| **maindec.v** |
| module maindec (  input wire [5:0] opcode,  output wire branch,  output wire jump,  output wire link,  output wire reg\_dst,  output wire we\_reg,  output wire alu\_src,  output wire we\_dm,  output wire dm2reg,  output wire [1:0] alu\_op  );  reg [8:0] ctrl;  assign {branch, jump, link, reg\_dst, we\_reg, alu\_src, we\_dm, dm2reg, alu\_op} = ctrl;  always @ (opcode) begin  case (opcode)  6'b00\_0000: ctrl = 10'b0\_0\_0\_1\_1\_0\_0\_0\_10; // R-type  6'b00\_1000: ctrl = 10'b0\_0\_0\_0\_1\_1\_0\_0\_00; // ADDI  6'b00\_0100: ctrl = 10'b1\_0\_0\_0\_0\_0\_0\_0\_01; // BEQ  6'b00\_0010: ctrl = 10'b0\_1\_0\_0\_0\_0\_0\_0\_00; // J  6'b10\_1011: ctrl = 10'b0\_0\_0\_0\_0\_1\_1\_0\_00; // SW  6'b10\_0011: ctrl = 10'b0\_0\_0\_0\_1\_1\_0\_1\_00; // LW  6'b00\_0011: ctrl = 10'b0\_1\_1\_0\_1\_0\_0\_0\_00; // JAL  default: ctrl = 10'bx\_x\_x\_x\_x\_x\_x\_xx;  endcase  end  endmodule |

|  |
| --- |
| **auxdec.v** |
| module auxdec (  input wire [1:0] alu\_op,  input wire [5:0] funct,  output wire [3:0] alu\_ctrl  );  reg [3:0] ctrl;  assign {alu\_ctrl} = ctrl;  always @ (alu\_op, funct) begin  case (alu\_op)  2'b00: ctrl = 4'b0010; // ADD  2'b01: ctrl = 4'b0110; // SUB  default: case (funct)  6'b10\_0100: ctrl = 4'b0000; // AND  6'b10\_0101: ctrl = 4'b0001; // OR  6'b10\_0000: ctrl = 4'b0010; // ADD  6'b10\_0010: ctrl = 4'b0110; // SUB  6'b10\_1010: ctrl = 4'b0111; // SLT  6'b01\_1001: ctrl = 4'b0011; // MULTU  6'b01\_0000: ctrl = 4'b0100; // MFHI  6'b01\_0010: ctrl = 4'b0101; // MFLO  6'b00\_1000: ctrl = 4'b1000; // JR  6'b00\_0000: ctrl = 4'b1001; // T  6'b00\_0010: ctrl = 4'b1010; // SRL  default: ctrl = 4'bxxxx;  endcase  endcase  end  endmodule |

**Discussion/Conclusion**

The lab experiment allows students to demonstrate their understanding of how to properly implement additional instructions to an already existing design. Three new modules were added into the datapath, rf\_ra\_mux, pc\_jr\_mux, and rf\_pc4\_mux. Within the ALU, a HI register and a LO register was also implemented to allow instructions *MFHI* and *MFLO* to output the values within the HI and LO registers. Within the main decoder and the auxiliary decoder, changes were made in order to send the appropriate control signals to the datapath. These changes are shown in the source codes above. The extended design for this lab was implemented in Verilog. The provided MIPS source code was assembled and the machine code was used as the input to validate the design. After eye balling the test results, the design appears to be functioning as expected. The experiment is ongoing, however, has already provided multiple opportunities for students to learn more regarding Verilog RTL design.