**EC 413 Lab 6 Report – Brendan Shortall, U71270986**

For SLT I added another if block under the R-type instructions that checks the instruction and outputs an unused function value 3’d5. From there I just added another statement checking if func == 3’d5, if so compute the slt (a<b).

For addi I began in the control block where I added a statement to check if the instruction is == 6'b001000. This indicates we want to add and immediate. From there I set ALUSrc and RegWrite to 1 and the rest to 0. This indicates that the ALU input will be an immediate instead of a register value and that it will write to a register rather than to memory. I also set the ALUop to 2’b11 which will indicate I type operations. Within the ALUcontrol I set a block to look for an aluop == 11. If it is this type, it outputs a function = 3’d0 (add). I did not add any other statements within this block since addi is the only i-type instruction I will be implementing. From there the implementation of addi should be done. The alu will add as normal, with an immediate input rather than reg input. It will then write to register as instructed by the RegWrite wire.

For the jump functionality I began by adding a jump output into the control module. This will be zero unless a jump instruction is seen. I added a branch within the control that accepts a jump instruction, which sets only jump=1, and aluop = 2’b10. I set aluop to this value arbitrarily because it does not matter what the value is. I then created another shift module to shift the last 26 bits of the instruction 2 bits to the left. I then concatenated the 4 MSB of the current instruction to the 28bits of the shifted instruction. From there, I added a new mux with the select = to the jump output from the control unit. If jump == 0, the expected address is passed to the PC If jump == 1, the address passed to the PC is the one we just computed.

For the branch instructions (beq and bne) I began by updating the control unit to accommodate instructions referring to branches. I then added an ALU to compute if the two inputs (read data 1 and 2) are equal. I used the instruction[31:26] to select which operation to output. If its BEQ the alu will output 1 if they are equal, and opposite if BNE. The output of this alu is then anded with the previous output of the and gate that checks for zero flag and branch. The output of this new and gate replaces where the old one was, going into the mux that selects if the expected next instruction is chosen or the branch instruction.

For lui I began by adding an LUI output to the control module that is 0 unless the lui operation is called, I also added a module inside the control to look for the lui instruction and set the parameters accordingly. I then created the shifted immediate using an assign line. From there I created a mux to select between ALU output and the shifted immediate. The output of this mux is then fed to the existing mux that selects memory data or alu data and feeds to write data input of the reg file.