ALU:

The ALU described in the first lab was used as the ALU in our processor.

IFU:

The instruction fetch unit was made following the design from lecture eight. The IFU takes in a clock signal, initialization control signal, 16-bit immediate, and two signals to check if the branch should be taken. The IFU outputs a 32-bit address that the instruction memory uses to get the current instruction. A 30-bit PC register that writes on each clock cycle was made, as well as an n-bit extender. The n-bit extender was chosen so that it could be used for the 30-bit immediate extension in the IFU and the immediate extension for i-type instructions as well as loads and stores. A 30-bit pc was used because the bottom bits of the address out are 00 in all cases. The upper 30 bits of the address out are PC+4 when the two branch signals (zero and branch in our implementation) are high, which is what should happen when a branch is not taken. When the branch is taken, the zero and branch signals are high, and the upper 30 bits of the target address are calculated by sign extending the immediate to 30 bits and, and adding it to PC+4. The main issues faced with the IFU was initializing the value in the PC. At first, we implemented the IFU as a closed loop, only taking in a clock signal, and there was no way to set a starting value. This was fixed by adding an initial value constant and an init signal, that is high for the first cycle and then low at all other times. This init signal is used as the select to a mux placed before the PC, which selects either the initial value or the new value of PC.

Control:

The control was divided into two components, the ALU control and the main control. Both were implemented using a PLA, because it would be simple to find the necessary output signals and would not require complicated logic. A PLA inverter and six-to-one AND gates were made to implement the PLA. The inverter works by taking in the six bits of data, as well as six bits that correspond to whether that bit of data needs to be inverted. If inv(i) is 1, then data(i) will be inverted before it is put into the AND gate. One note about this is that they are currently separate components (the PLA inverter, and the AND6to1), however, it would have been simpler to combine these two into one component for the PLA, rather than having one of each for each minterm, but I realized it too late. The current implementation still functions, properly but is not the most efficient in terms of the number of lines of code required. For the ALU control, the inputs are the opcode and the function code and the output is the necessary function for the ALU. Espresso was used to minimize the logic of the PLA’s. The R-type instructions looked at the function codes and found the output based on these, and the non R-type instructions looked at the opcode. Muxes were used to select either the output from the R-type PLA, or the non R-type PLA, with the select bit being all the opcode bits OR’d with each other, and the result inverted. This is because the opcode for R-type instructions is 000000, so the select will only be 1, when the instruction is R-type. The main issues we encountered with the ALU control, was typos in the .pla file used by Espresso. For the SLT and SLTU, we initially had the output as the subtraction ALU opcode, instead of the SLT or SLTU opcode. This generated the wrong PLA and had to be corrected.

The main control was implemented the same way as the ALU control for simplicity. The main control takes in the opcode and generates the nine control signals using a PLA. The output signals are: RegDst, RegWr, Branch, ExtOp, AluSrc, MemWr ,MemtoReg, BrSel1, BrSel0. Most of the control signals are the same as in the lecture; however, we needed to add a 2-bit branch select signal, that would feed in the appropriate value to the zero input of the IFU, depending on the type of branch. BNE, BEQ, and BGTZ all have different conditions in which they should branch, so they were fed into a 3-to-1 mux with BrSel as the two select bits, and the appropriate signal was passed to the IFU. We encountered problems similar to those from the ALU control when creating the main control. The main issues were typos or writing down the wrong control signal that needed to be generated before running Espresso. However, once the skeleton of the main control was made, these errors were not too difficult to fix. Both the controls, were able to be exhaustively tested due to the small number of inputs and outputs.