Task 5:

Building the instruction logic was divided into 3 parts. The first part I worked on was the actual program counter which essentially is just seeing if clear is enabled and if so it sets the program counter to zero and if not, leaves it as it is. The add_program module I designed checks to see if a jump/branch is enabled and if so, it will add 1 and the offset(4 in this projects case) to the program counter, if not it will just add 1 to the counter. The instruction memory was a fairly simple task. I started by creating 8 memory spaces and using the 32 bits of each space to write the code for each instruction. Then from there it's just a simple assignment statement where the input address indexes into the memory and is outputted.

Task 6:

The difficulty of task 6 mostly came from deciding what each instruction would need each individual wire to do. The instruction would be the only input and from there it would be broken up into 4 parts to be sent to other modules by the control unit. After breaking up these parts I hardcoded the values for each OP-CODE and used a case statement to see what OP-CODE was being used. From there I just went though each possible case and manually set each wire equal to 1 or 0 depending on what operations would be needed by that specific instruction.