All,

I would like to clear the water on a few specifications on this project:

1. The PSW is used for interruption/exceptions/traps of programs, conditional branches, and privileged instructions. That being stated, the PSW is a register that stores a 16 bit value that symbolizes the current status of execution. The values of this register will only require a change (store a new set of 16 bits) when the S bit is set (the S bit will be set if the Z or N bits are different from the current PSW Z and N bits). It is also used for other things not mentioned in this project, i.e. BRZ or BRN (Branch on Zero or Branch on Negative).

2. A REGISTER IS SIMPLY 16 LATCHES (or D Flip-Flops) THAT REPRESENT 16-BITS, there is no need to create your own. Potentially use a black box or use a model from the Simulation/CAD software you are using (I suggest Logisim, it's a simple, multi-platform tool).

3. Also, remember that the 16 bit BUS can be 16 wires that are independent of each other. During simulation it's advised that you have a mechanism in place to control what "goes on" the BUS at a time (only one 16 bit value can be using the BUS at a time, otherwise there will be catastrophe :( ).

One of the best things that Electrical Engineers created is the Tri-State Buffer!!!!! This device is a 2 terminal device, that, when the top terminal is turned HIGH will allow the input terminal to flow through the output. Likewise, when the top terminal is LOW, the input is simply grounded, not allowing flow. This device is perfect for solving the problem that number 3 represents.

It's coming down to ~2 weeks until the project is due and I have already met with a few of the teams. I am open to having some out of office hour meetings, but a reminder that I have a busy schedule with research and may not be able to respond or meet at your digression. I have a few possible office hours on Mondays from ~12pm to ~3pm and can work things out.

A few notes regarding the ALU:

Remember that it is a computation ONLY unit. It's objective is to perform an arithmetic operation on 2 operands, i.e. Add, Subtract

When I say Add, Subtract, these are NOT the instructions (opcode ADD, SUB...), they are simply the logic operation used. You will use these in almost EVERY instruction

A few notes involving the Hardwired Control Unit and project in general:

Remember that the goal of this project is to simulate/describe the operation and optimization of your design for a Hardwired Control Unit.

Take your time and think about what this is; Don't dive in to the layout of the machine until you know what you're control signals are first!

Please remember: Take the PC on the BUS! Increment it and use this ("Effective Address") to Fetch an Instruction from Main Memory (MAR - Address Register - Read ONLY), Decode it and store it in the IR, this will contain your instruction (Z, N, OPCODE, RS1, RS2, and so on...), Execute the "instruction" by passing the decoded OPCODE to the Hardwired Control Unit and send the correct control signals for the specified instruction clock cycle.

Please read a little bit about this example below (#7).

Do it, it will help, below (#7).

I promise, below (#7).

A good example of a Hardwired Control Unit can be found here – <https://users.ece.cmu.edu/~koopman/misc/byte87a.pdf>

I will not be available on the week of the Fourth of the July after that Monday. So come to my office hour please, I will make sure to make it from 12 - 245pm for this case.