Dallas Delaney

2. **Descriptions of highlighted blocks**.
The blocks discussed are numbered and highlighted in orange in the datapath block diagram

1. This is an adder that takes the first 11 bits of the instruction (inst.) output from the IR and adds it to the PC value. Used in JSR
2. The lsb_write module is used for LDB. It takes inputs from the least significant bit of the alu output (calculated address) and the output of the MDR. The LSB dictates which byte of the input taken from the MDR will be used for the first 8 bits of the output. The last 8 bits of the output are always 0s.
3. This module zero-extends the first 5 bits taken from the instruction output of the IR. Used for immediate ADD/AND
4. This module zero-extends the first 4 bits taken from the instruction output of the IR. Used for immediate SHF
5. This offset mux decides which offset (pc+9 or pc+11) will be sent to the PC mux. Added for JSR
6. This module takes the first 8 bits of the instruction output from the IR and zero extends it to form a word. Used in TRAP

3. **An explanation of how you tested your design**.

Testing my design was a challenging process. I found it very helpful to test each instruction separately. This is because many instructions were dependent on other instructions previously executed in the code. For example, if I tried to test LEA, SHF, and ADD all at the same time by checking the register output at the end of execution it would be hard to tell what went wrong. LEA and SHF could be working fine but ADD could be broken and there would be no way to tell using this method. I would use the lc3b IDE and always check the register output from that and compare it against my waveform in ModelSim. To do this with more ease, I would always have the ‘opcode’ and ‘state’ signals showing so that I could keep up with the IDE’s execution.

The given test code did not always help in debugging and testing my code. This is because I did not write the instructions in any given order. The test code has a set order of instructions. If I wanted to test one instruction that I wrote using the test code and it showed up in a line of the test code that was after an instruction I hadn’t implemented yet, I wouldn’t be able to test it. This is why I wrote my own test code to test whatever instruction I was on. Making sure the instructions worked was easy this way. The registers shown on my waveform just had to match the registers from the lc3b IDE. Overall, testing my design was rather intuitive. All I had to do was follow the datapath until I found an unexpected signal value.