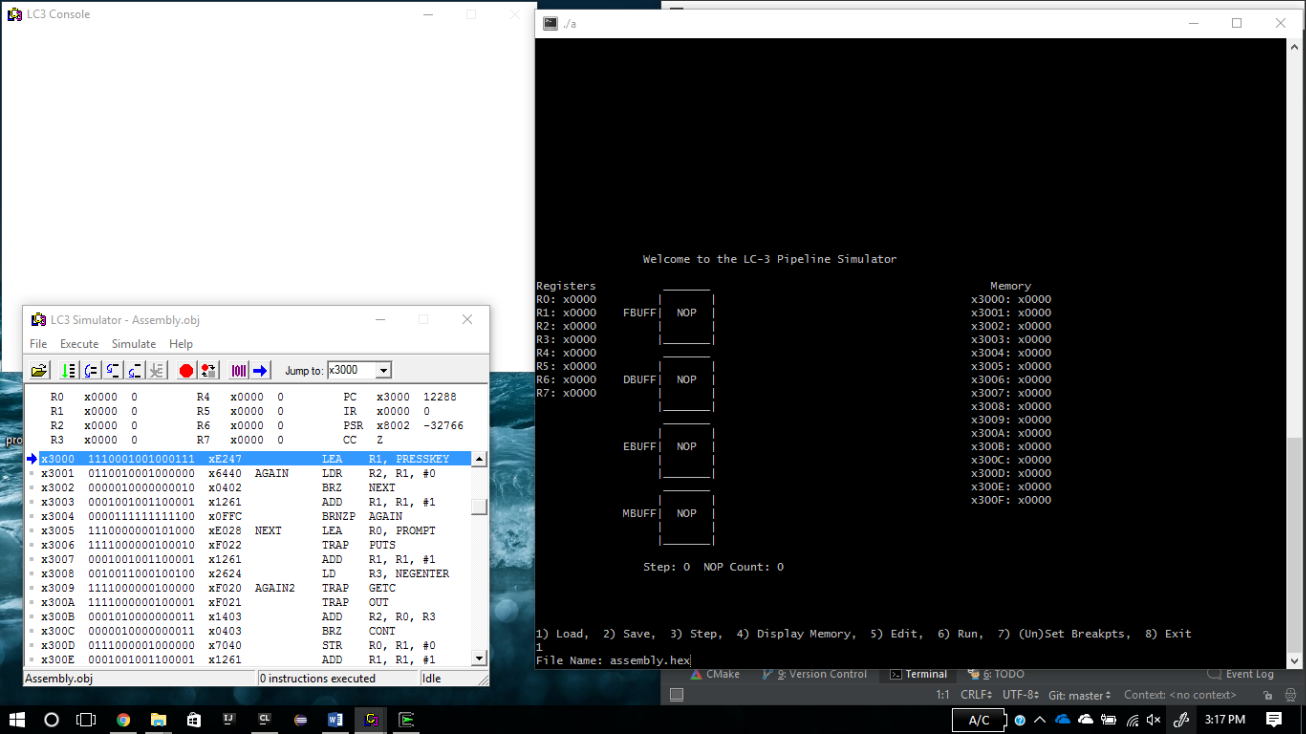
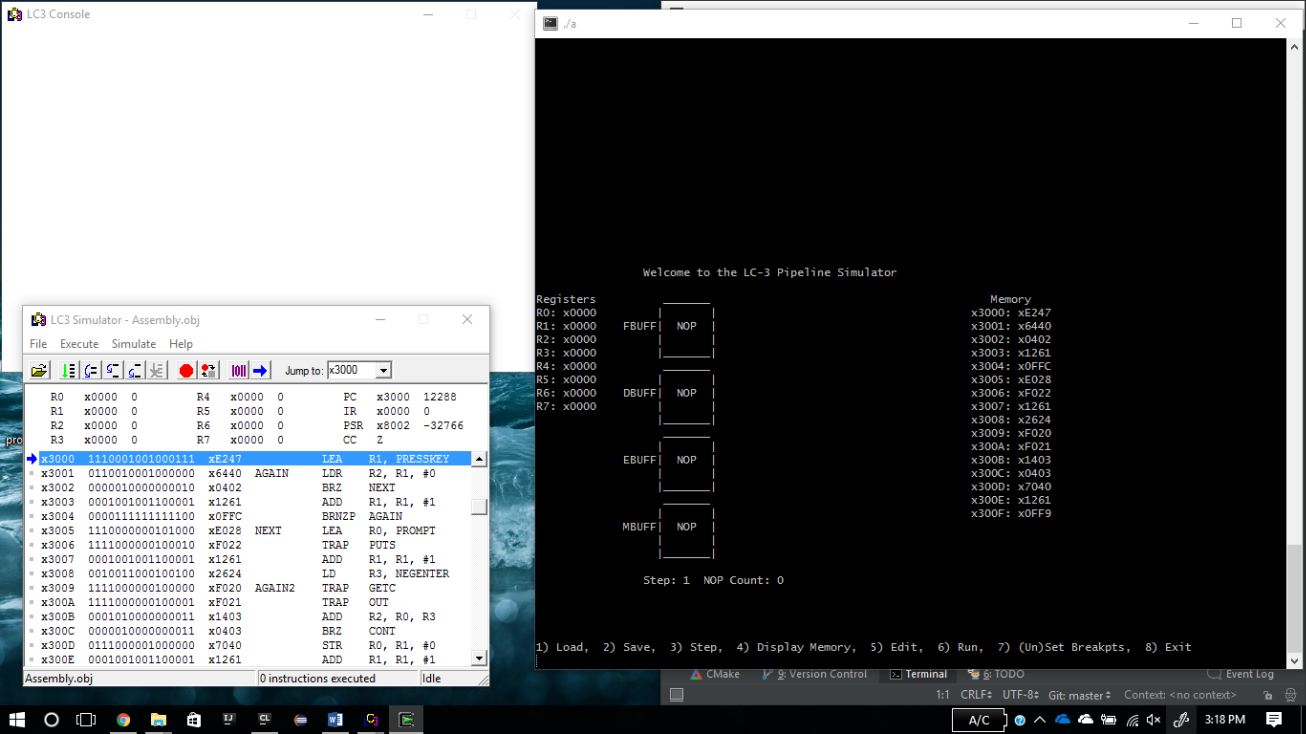
Load File:

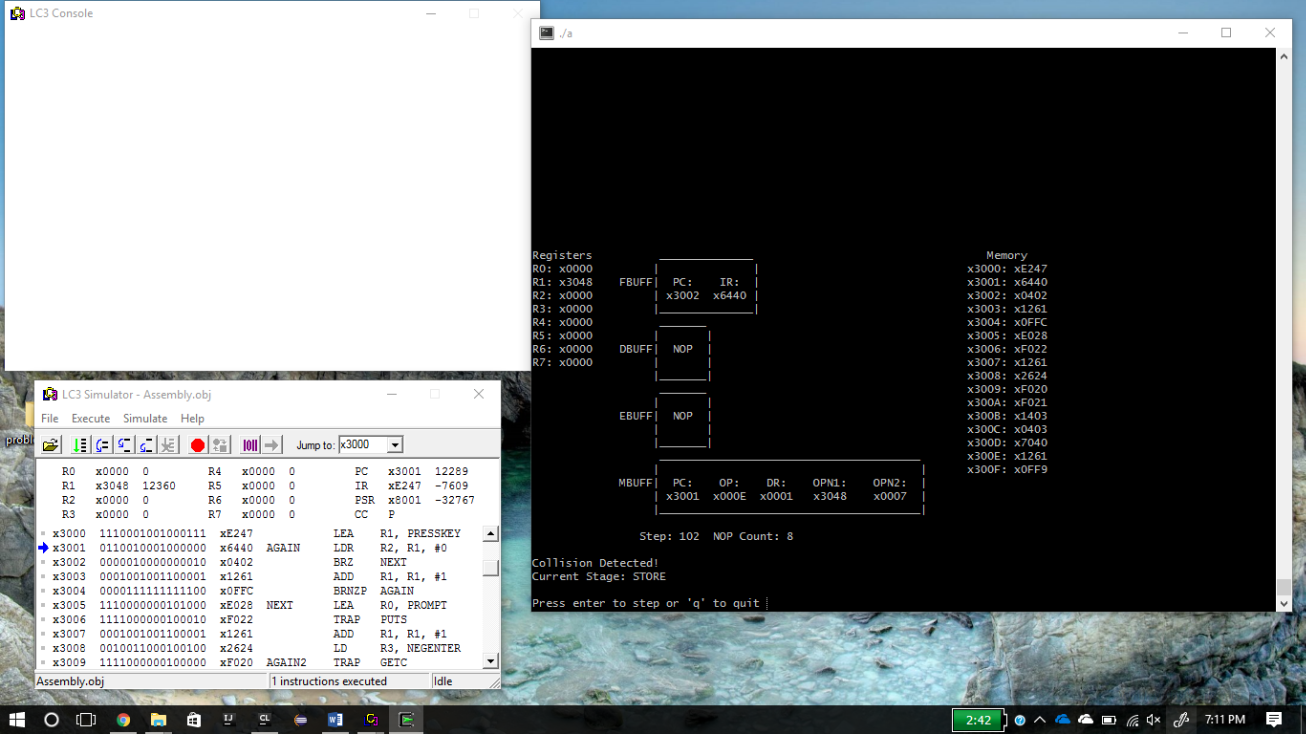




Step:

1st instructions

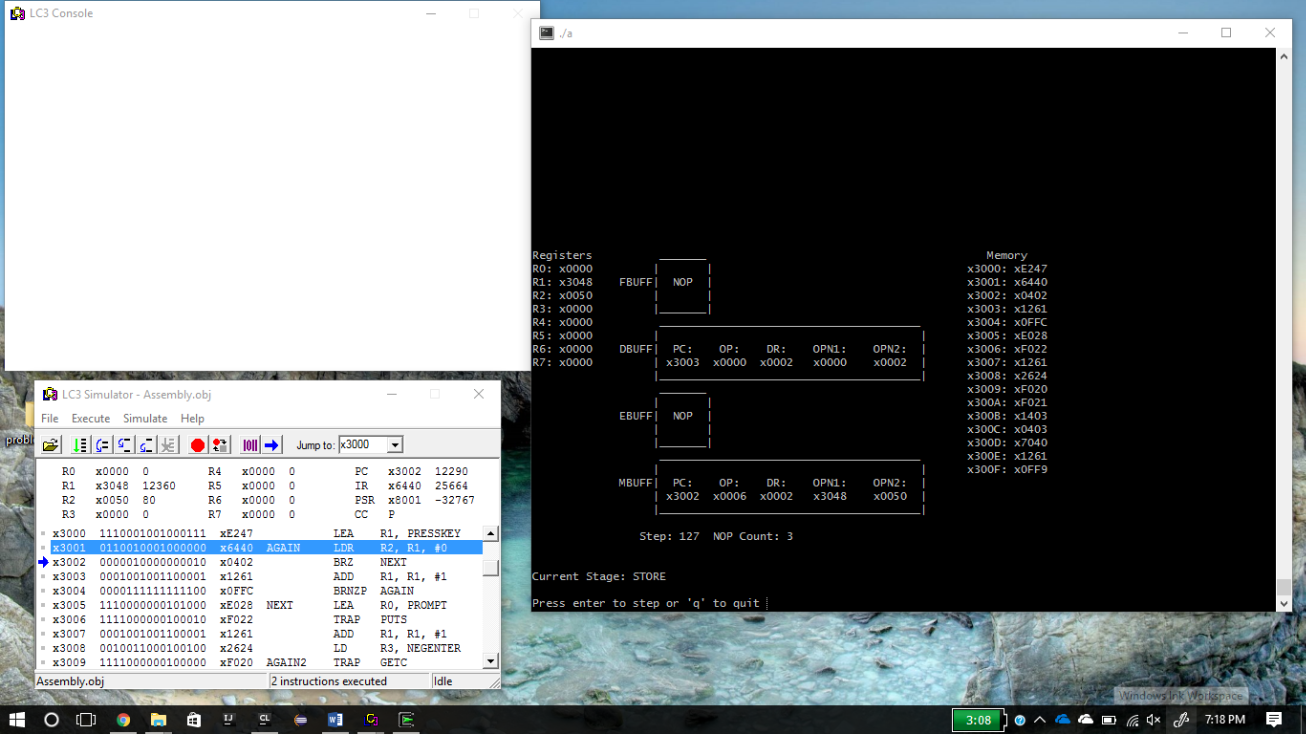
Load r1 with 3048 (102 steps)



2nd instruction

LDR r2, r1, 0

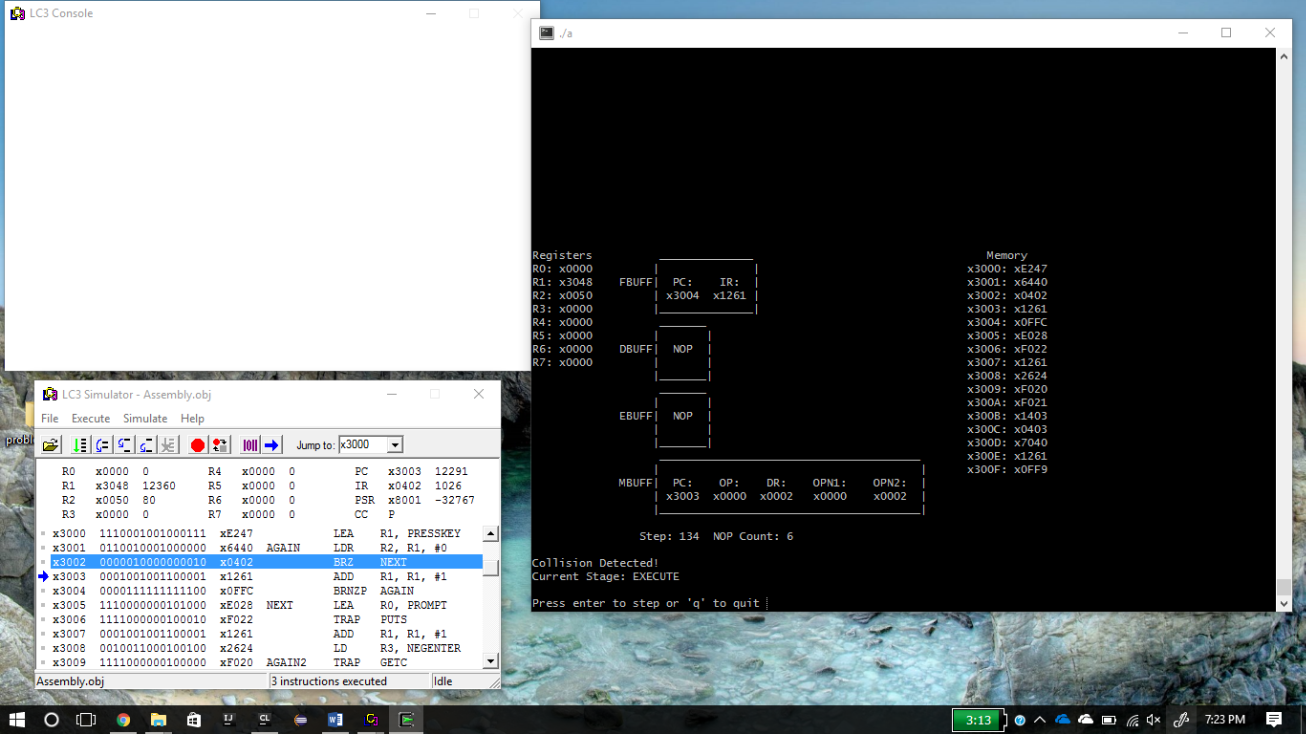
R2 contains x50 Step 127



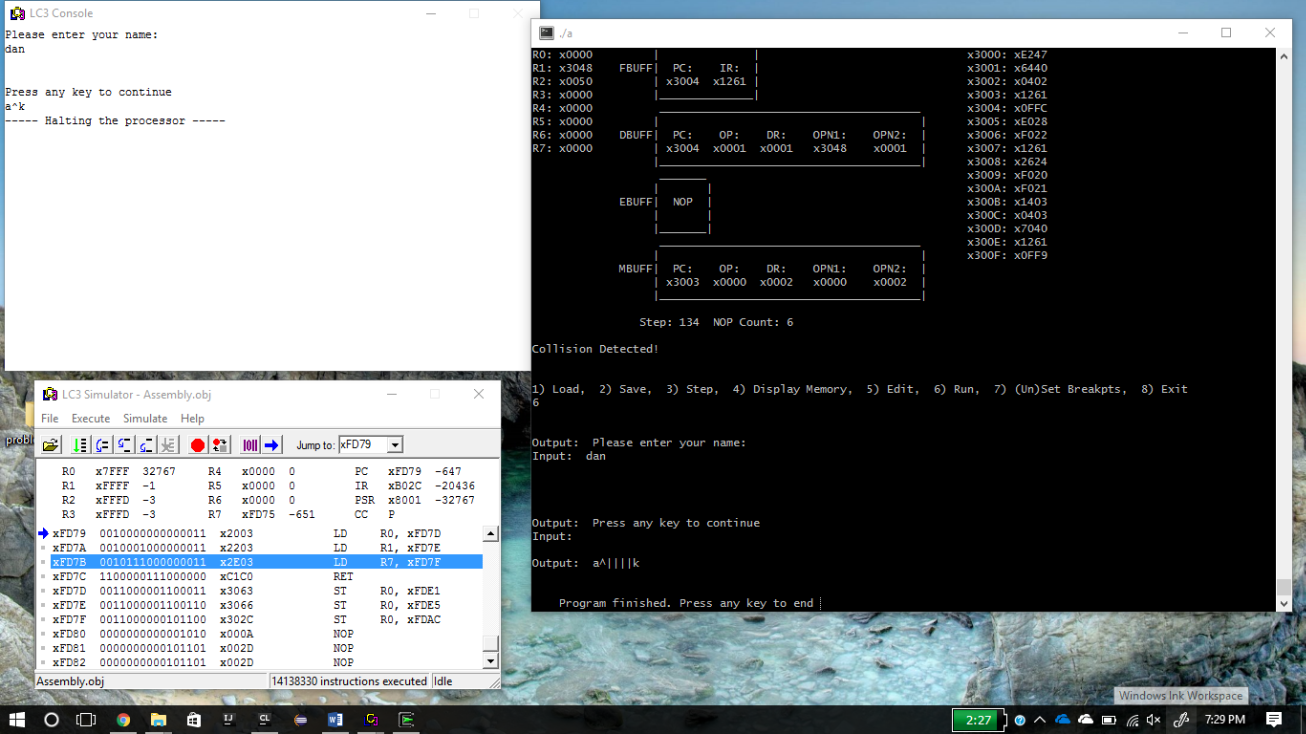
3rd instruction

Branch on zero.

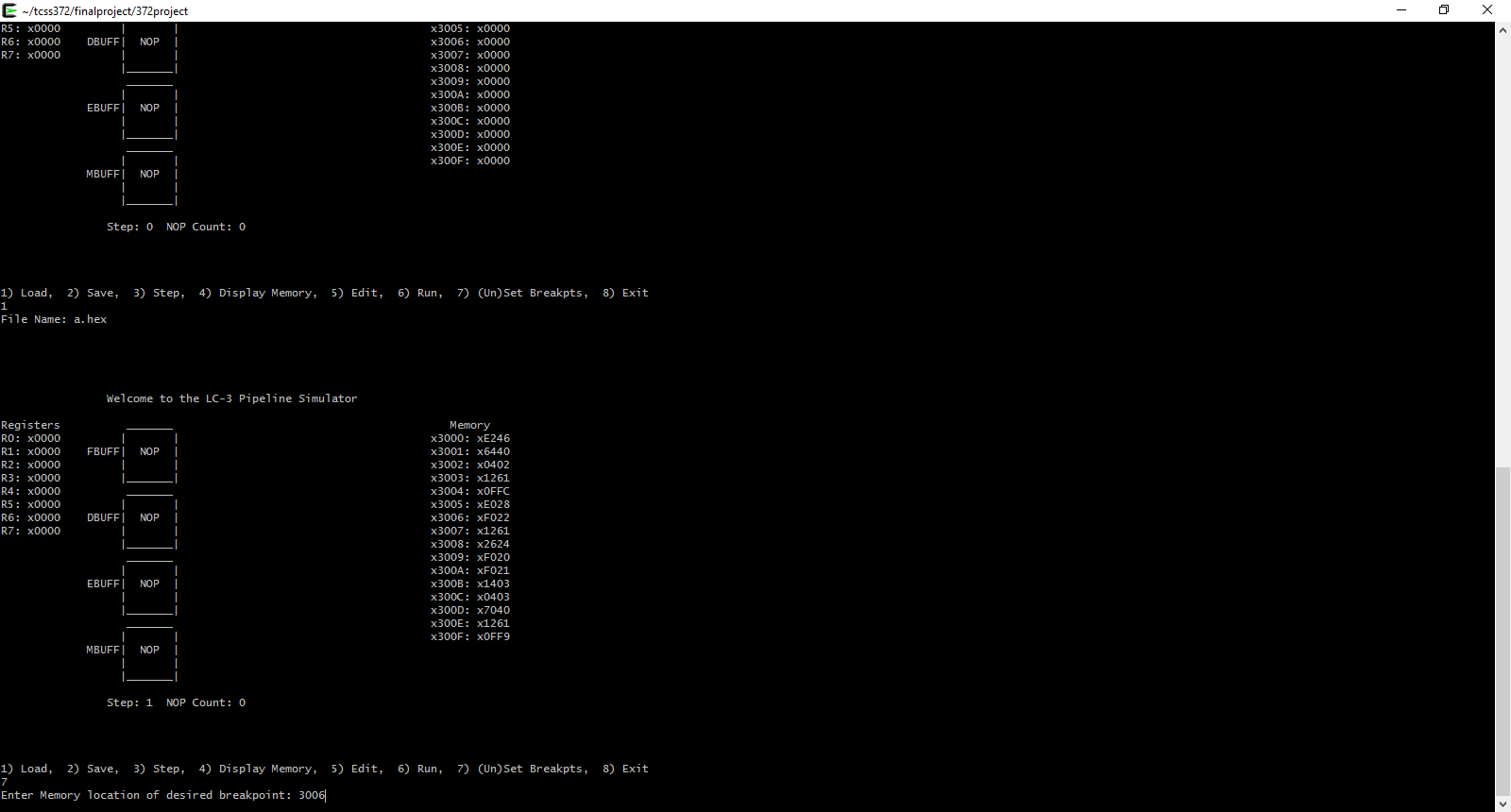
Step 134



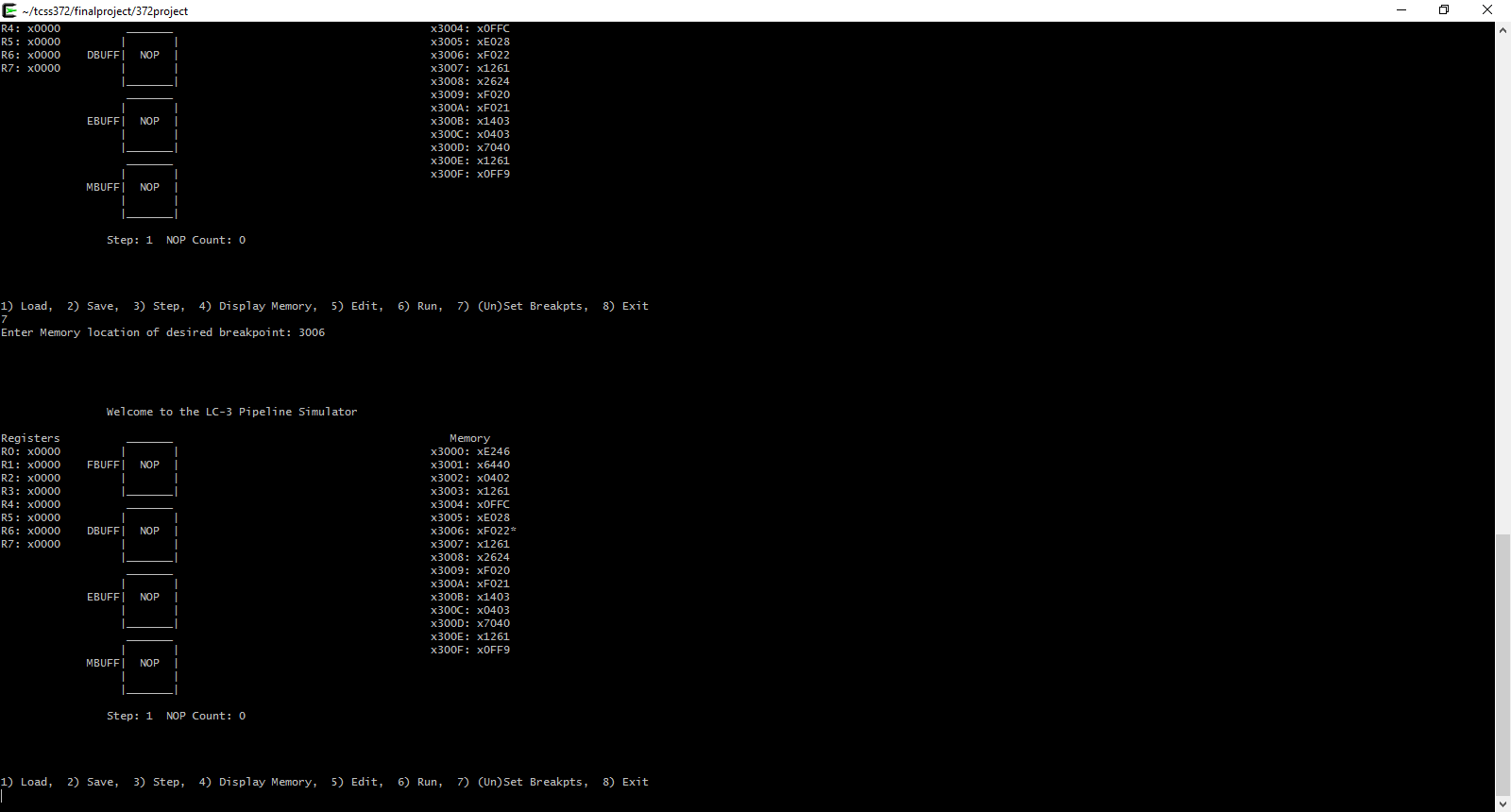
Program Run through - > Output



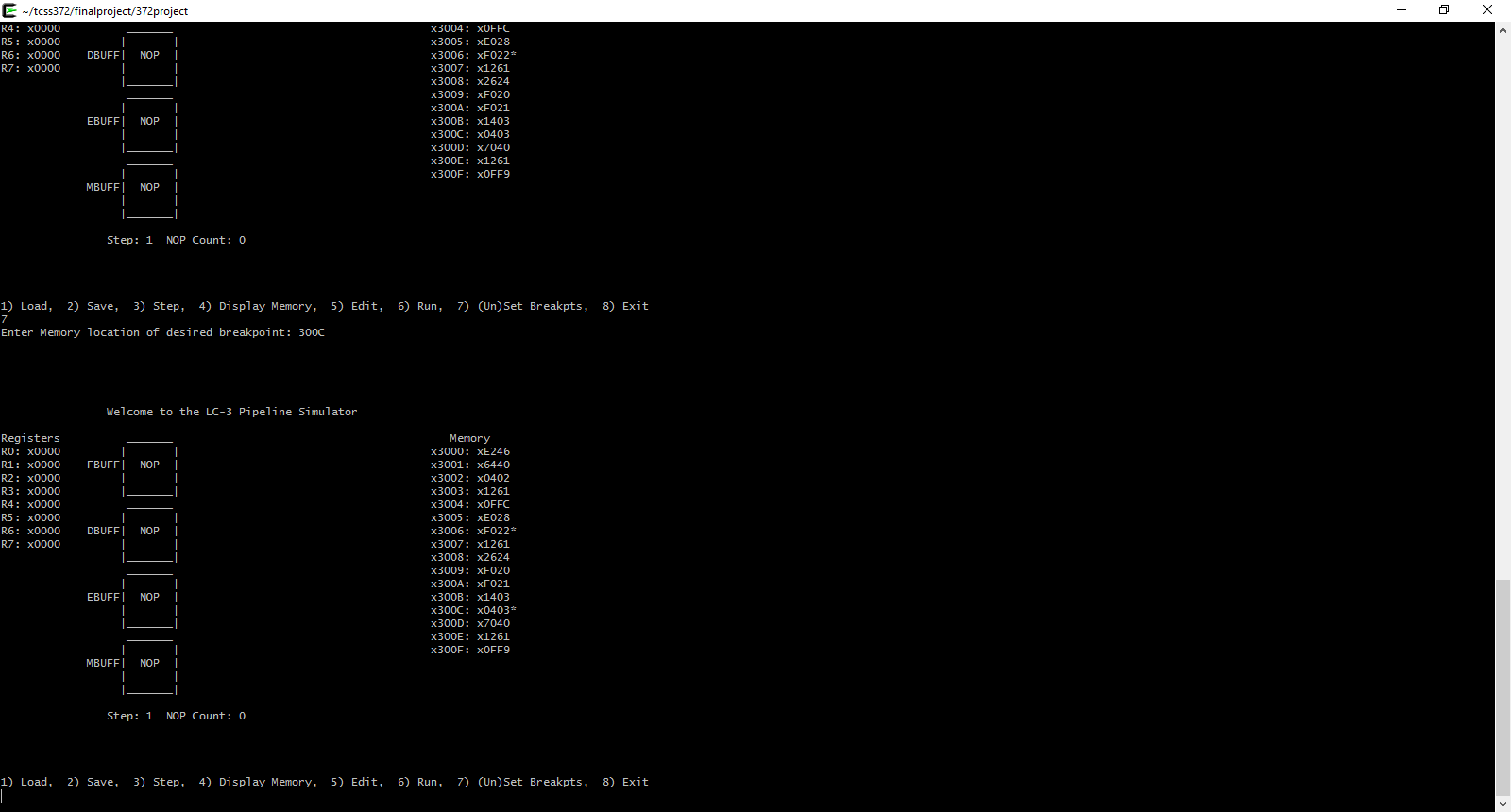
About to mark a breakpoint at address x3006



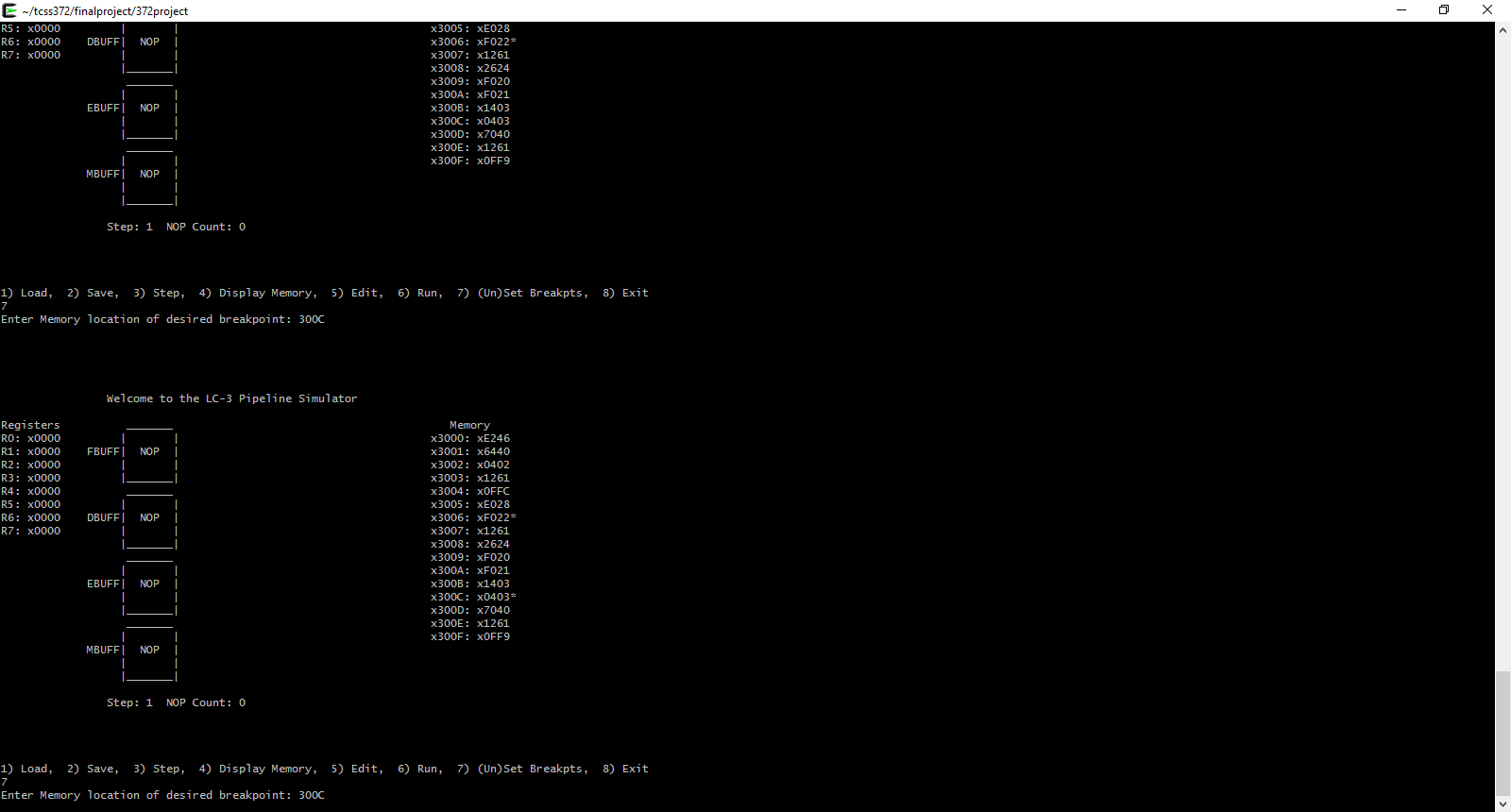
After marking a breakpoint at address x3006



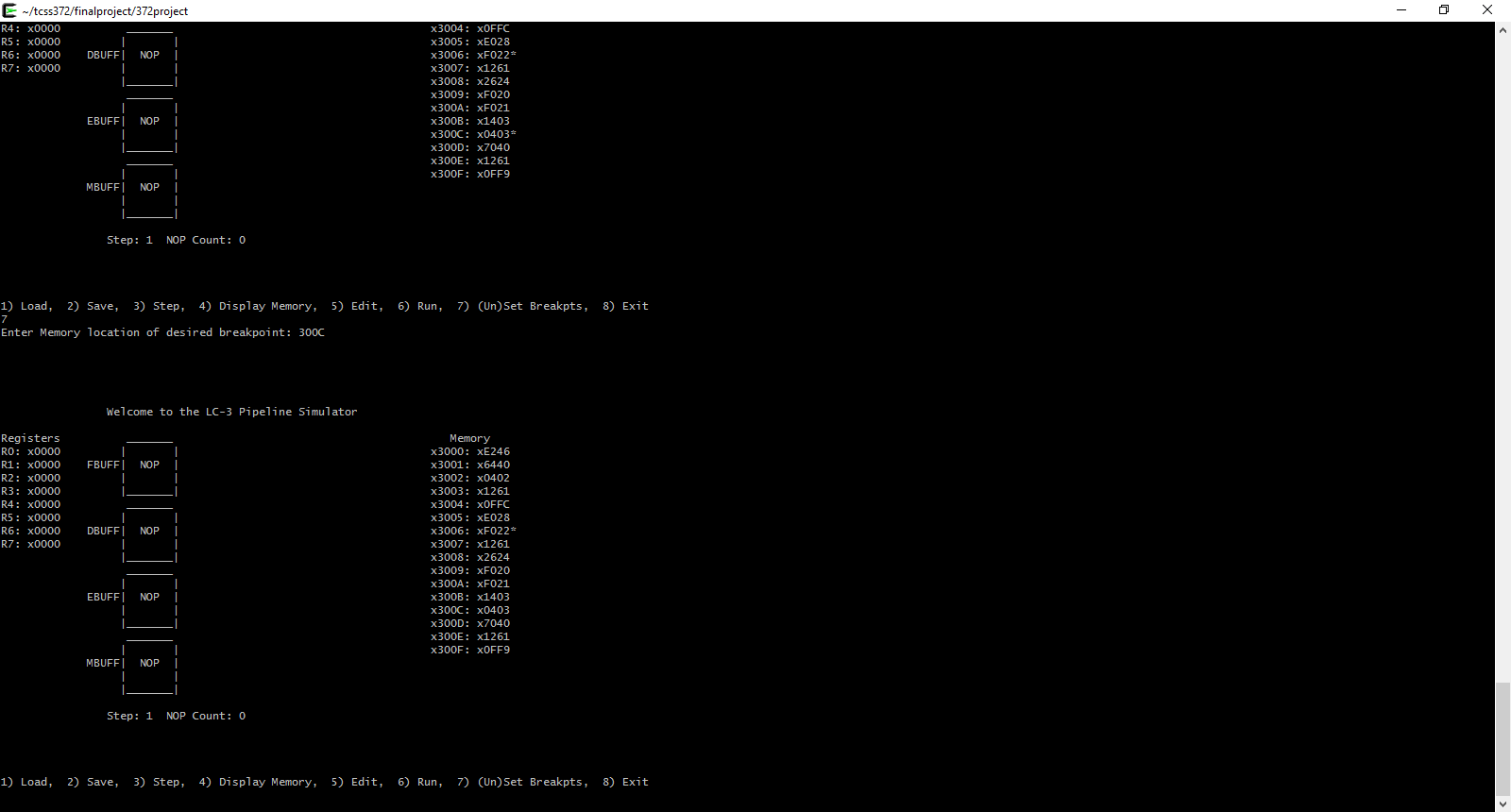
After marking another breakpoint



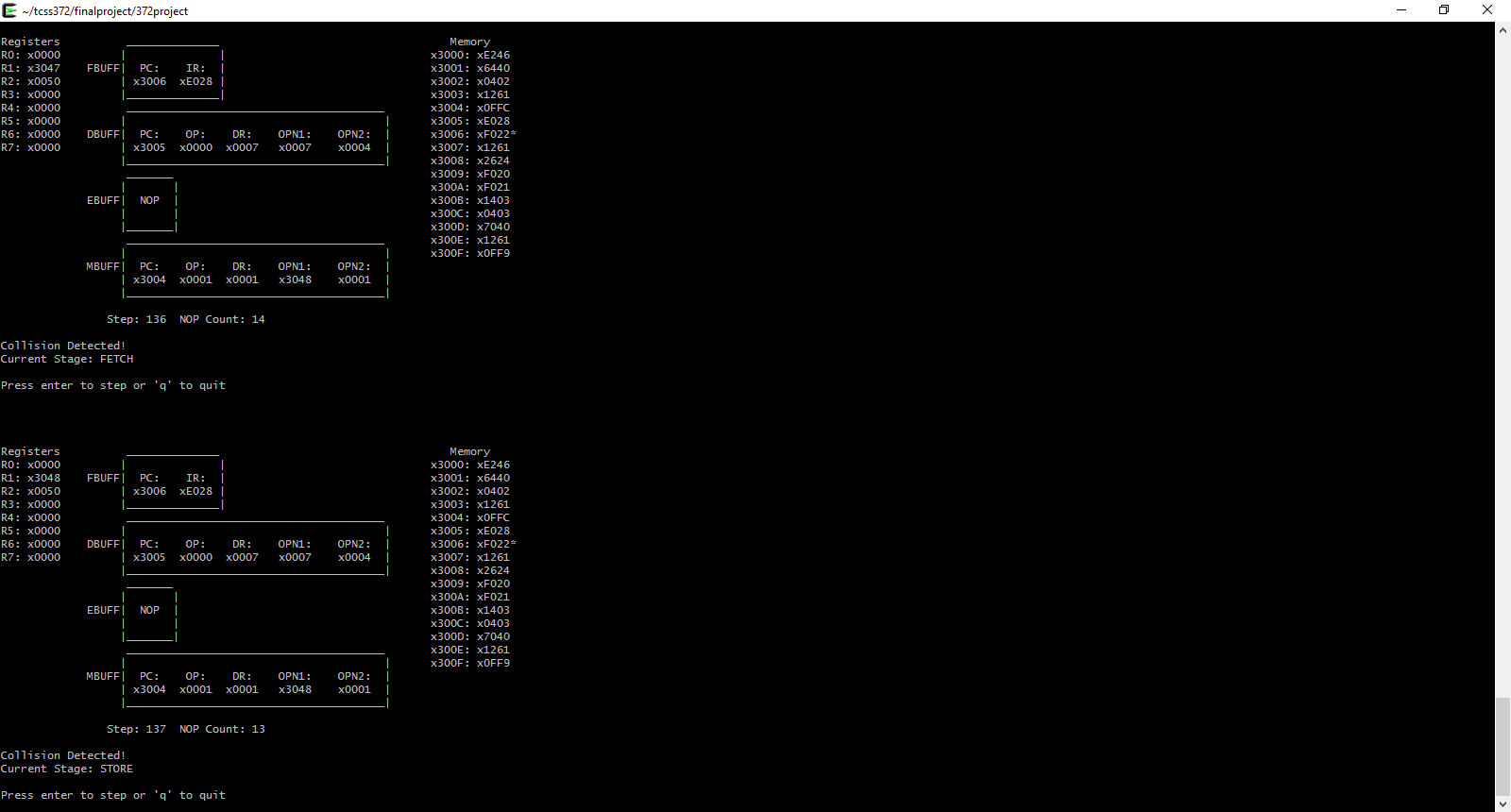
Before removing breakpoint at x300C



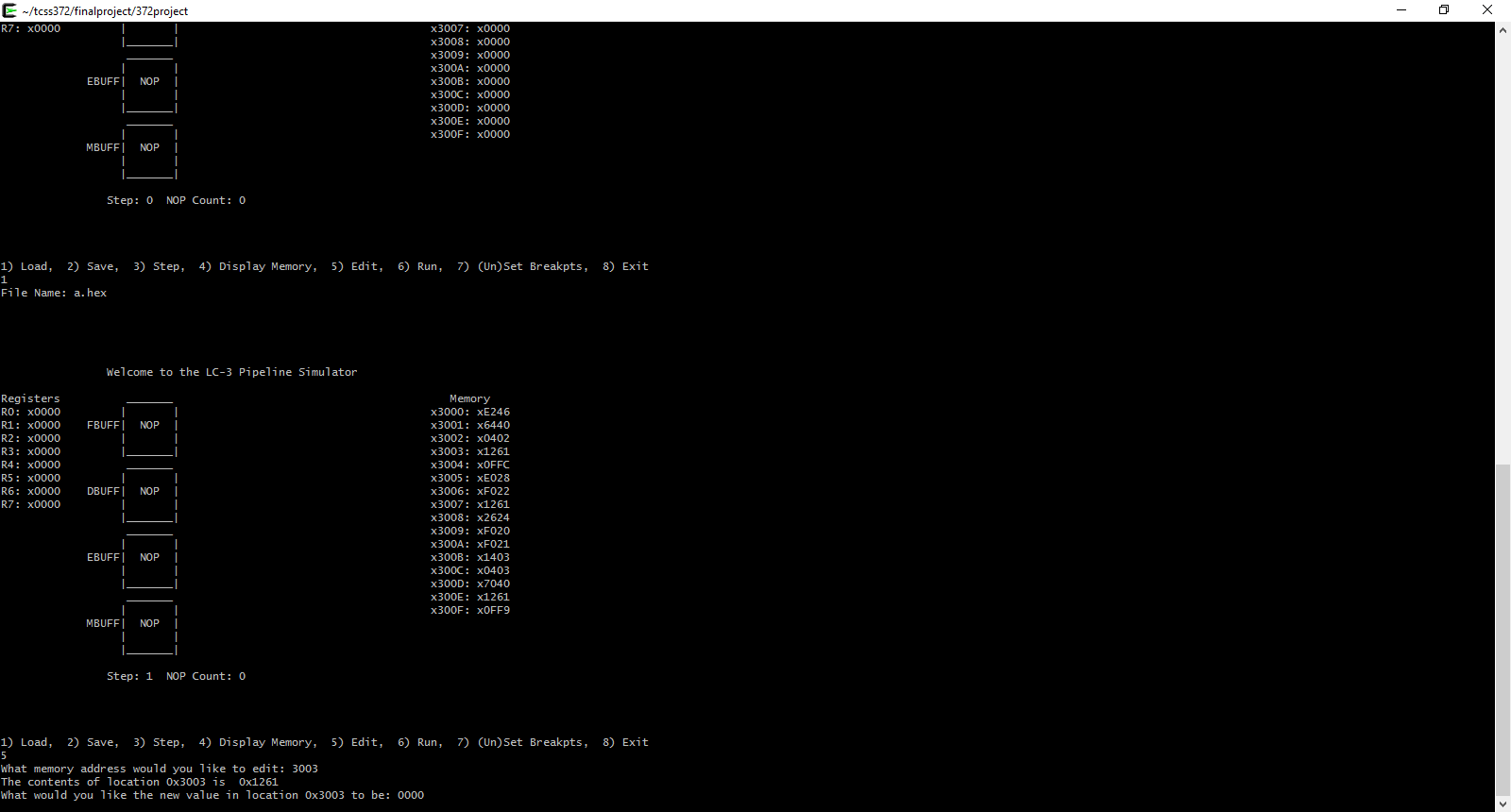
After removing breakpoint at x300C



After running with the breakpoint set

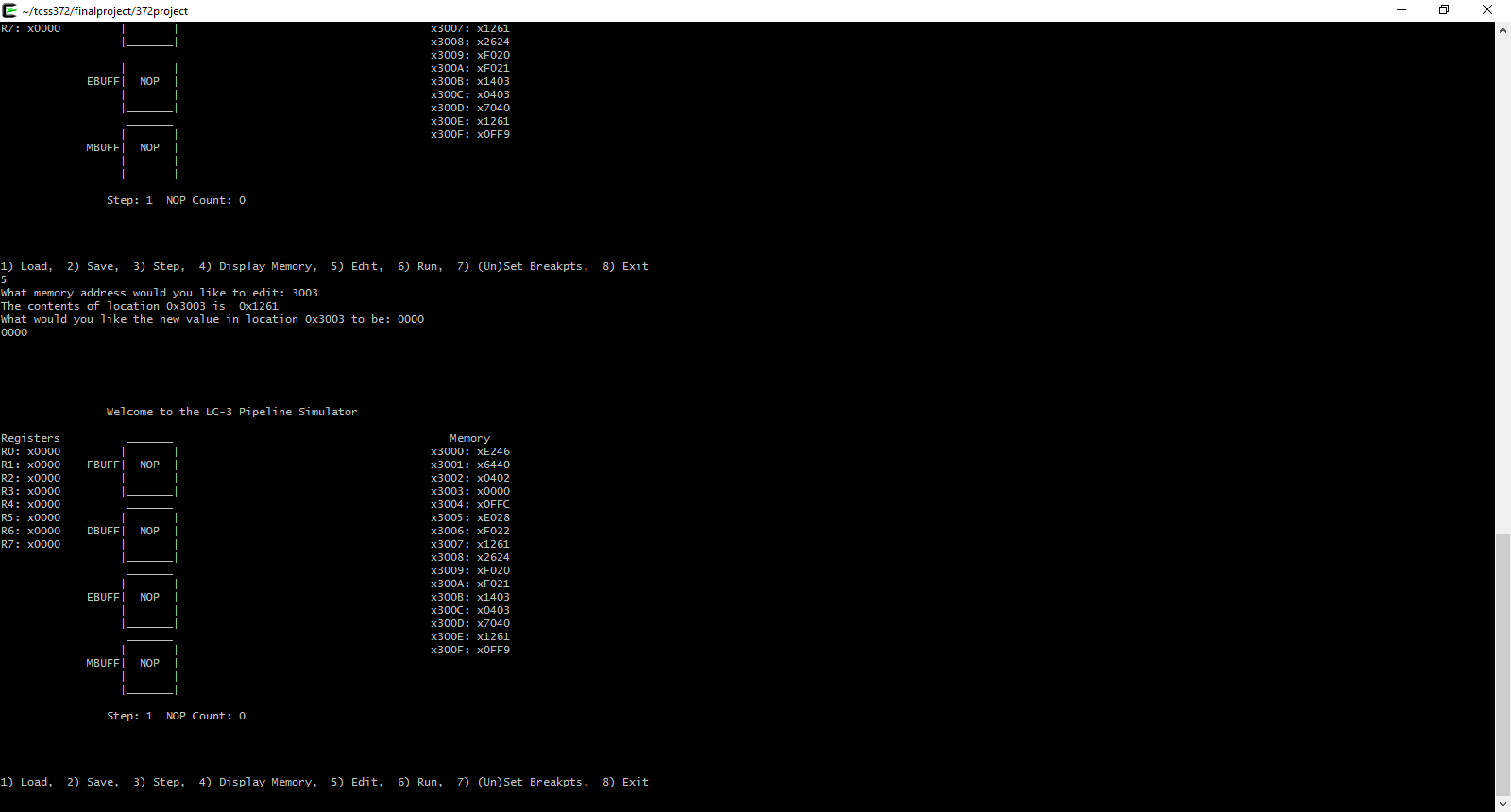


Before editing the memory address x3003

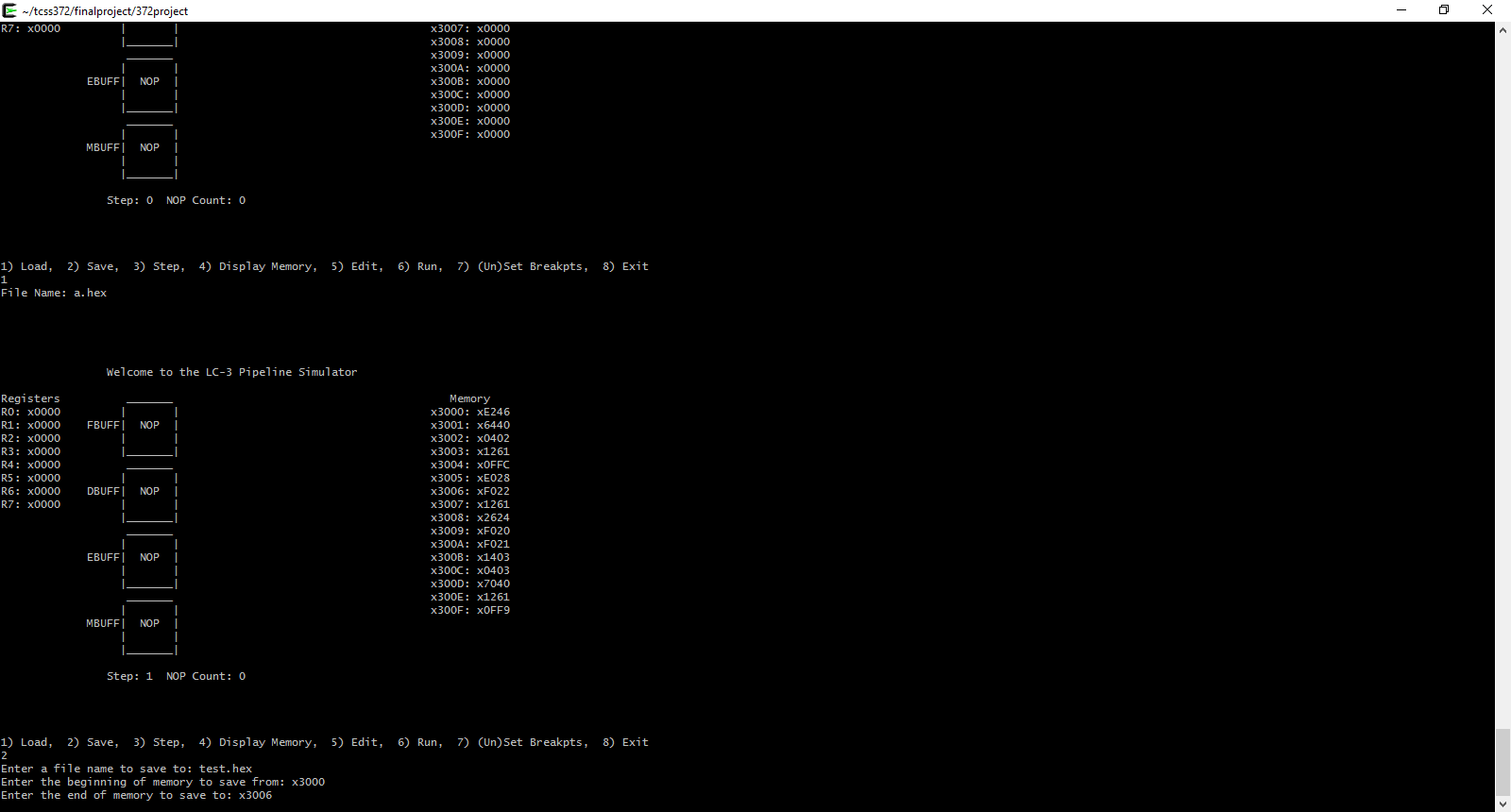


After editing the memory address x3003 from x1261 to x0000

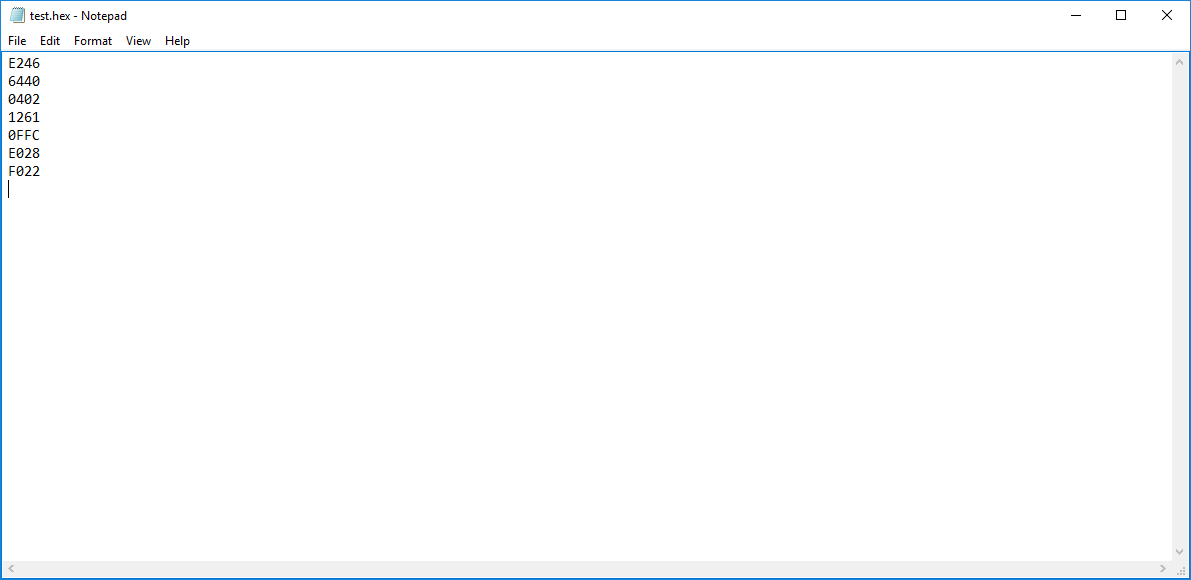
.



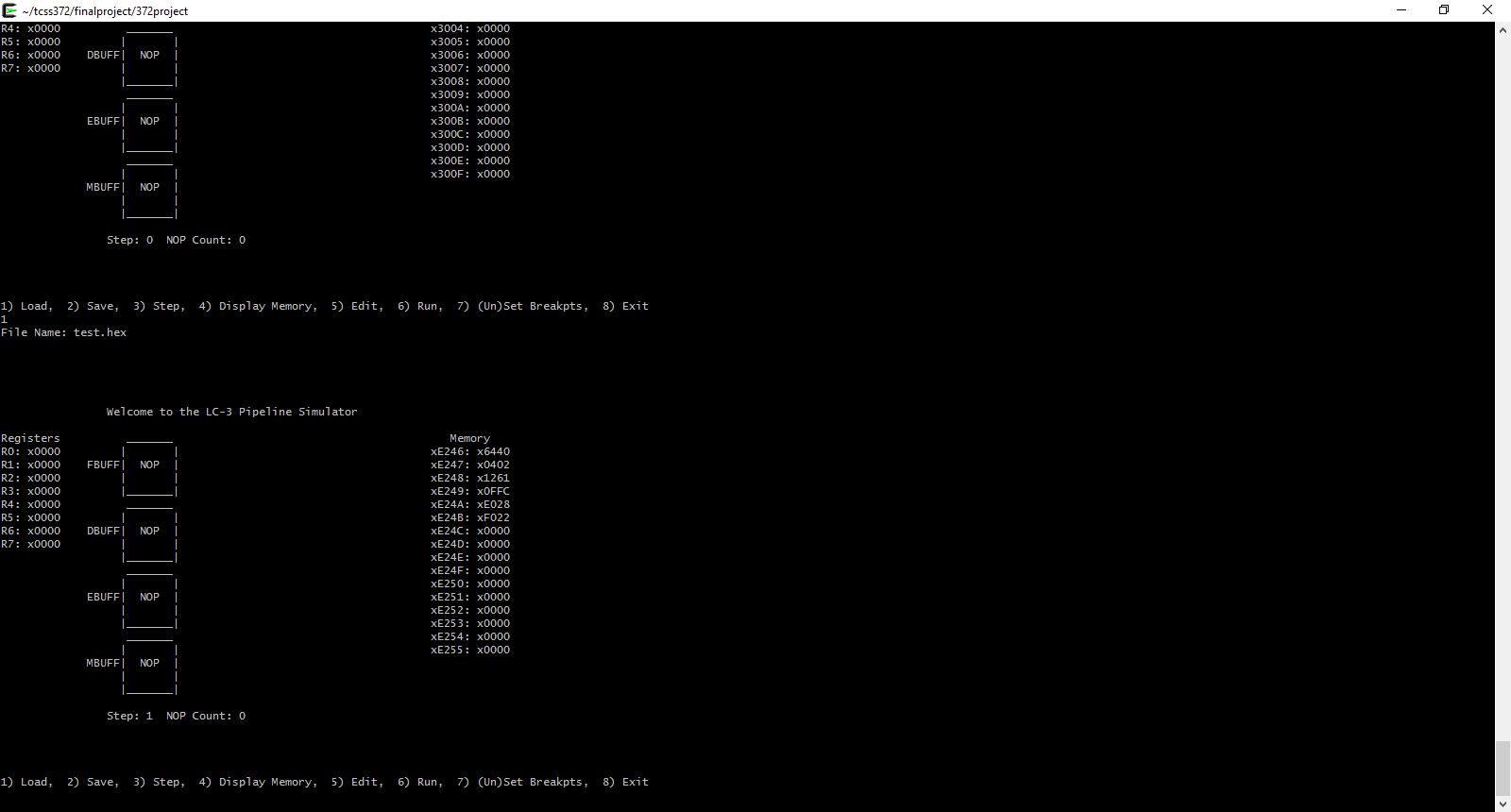
Before saving the current memory file



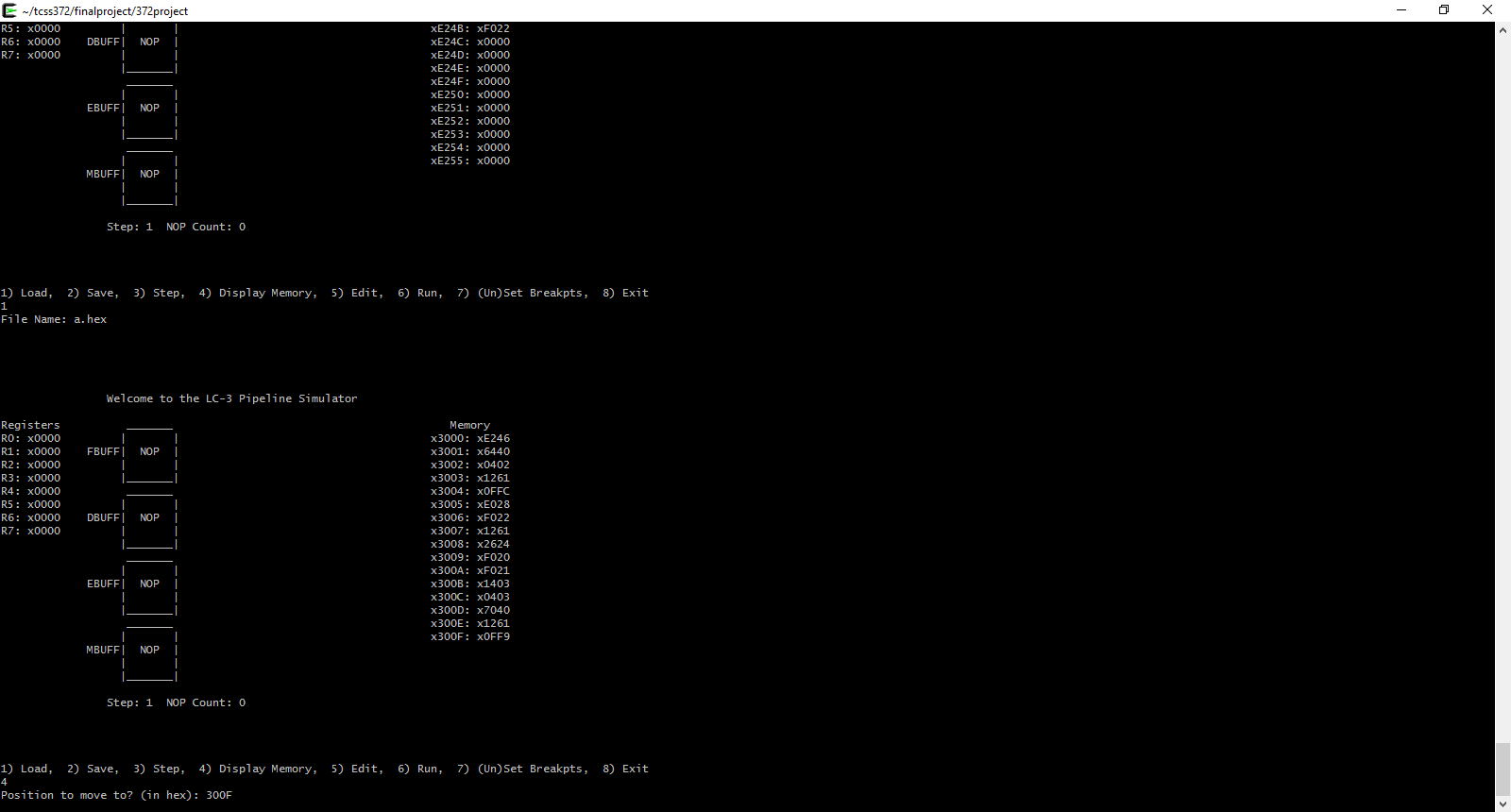
After writing to test.hex



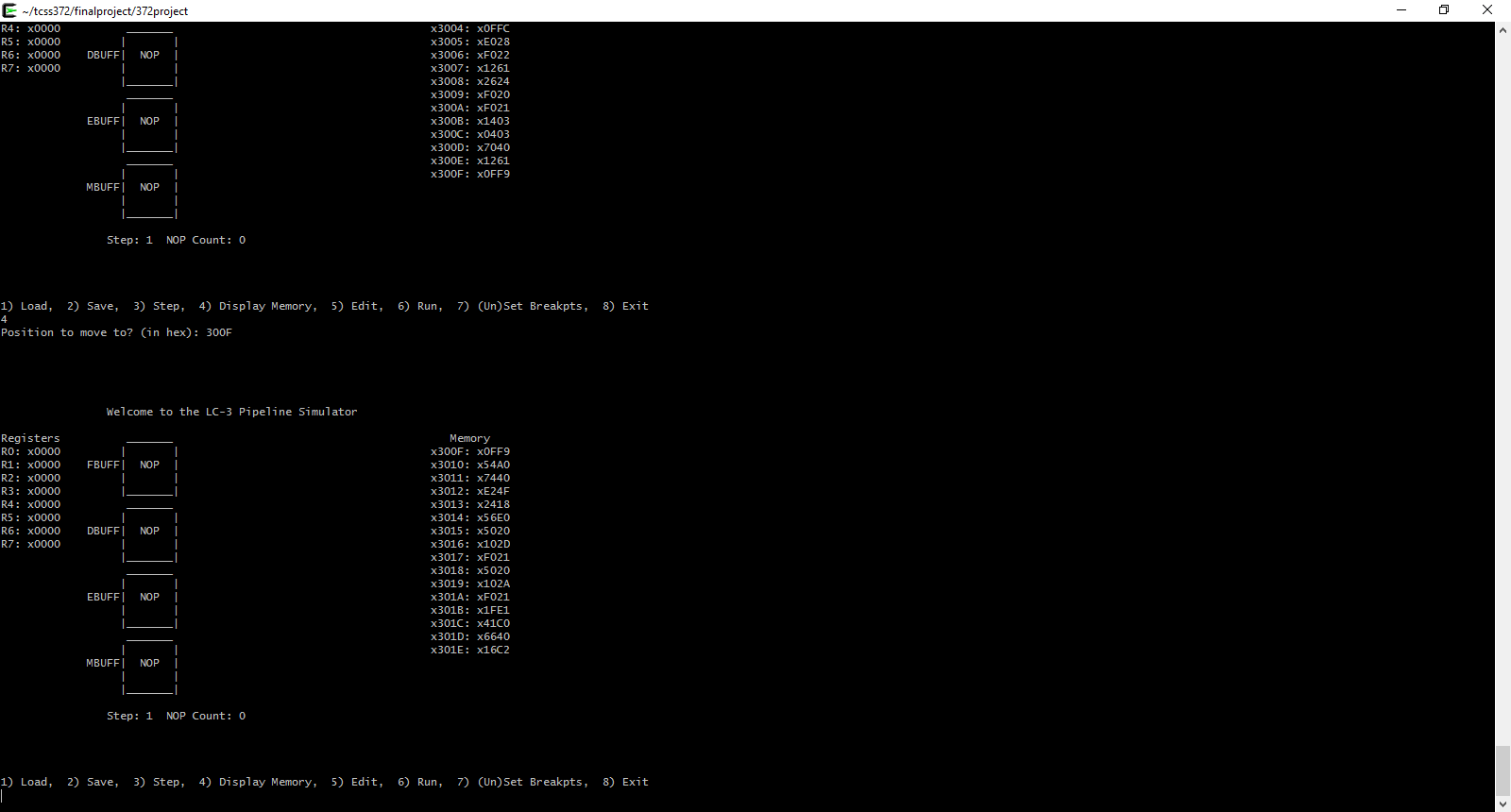
After loading test.hex back into the Simulator



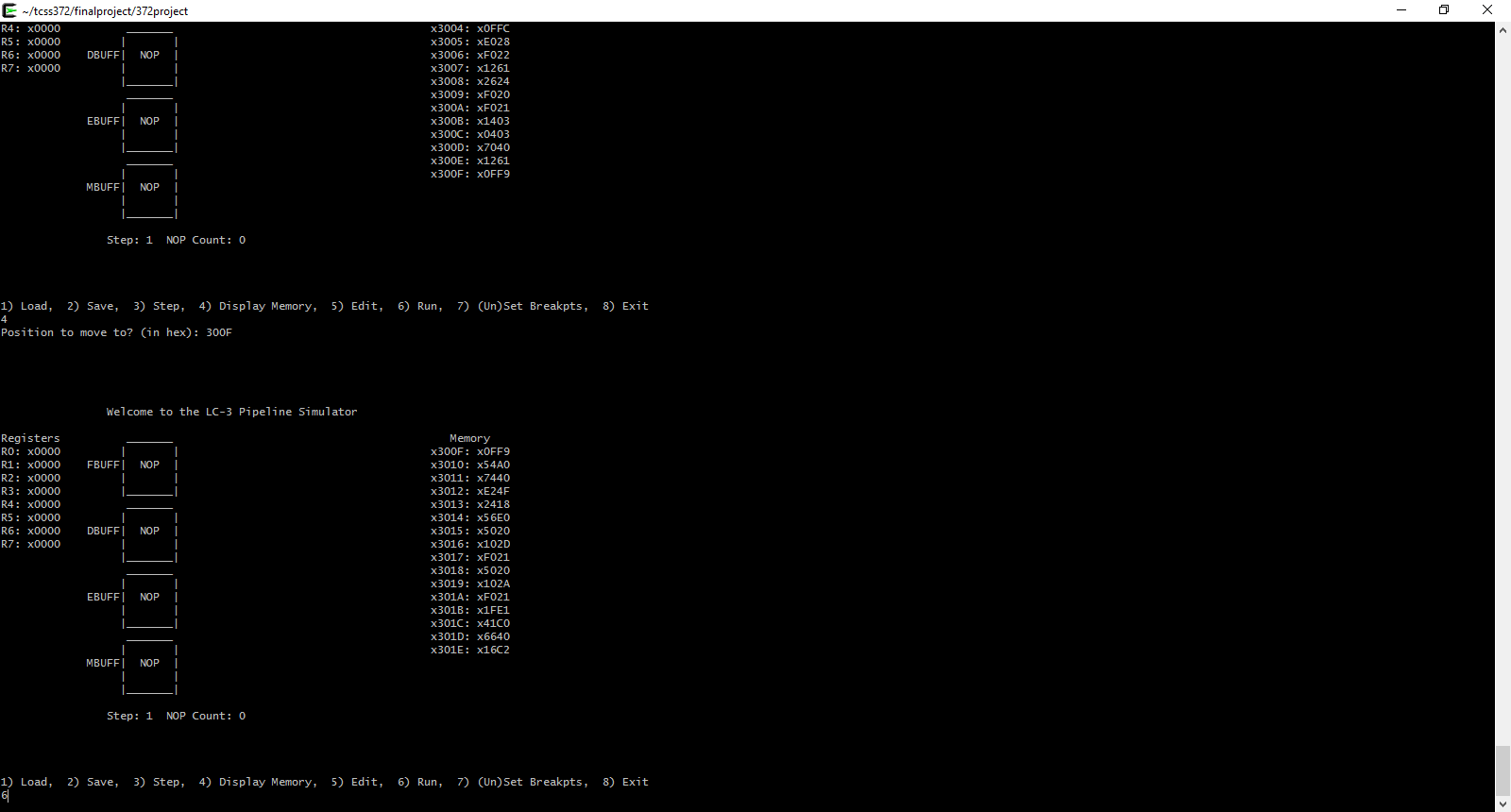
Before moving to x300F in memory



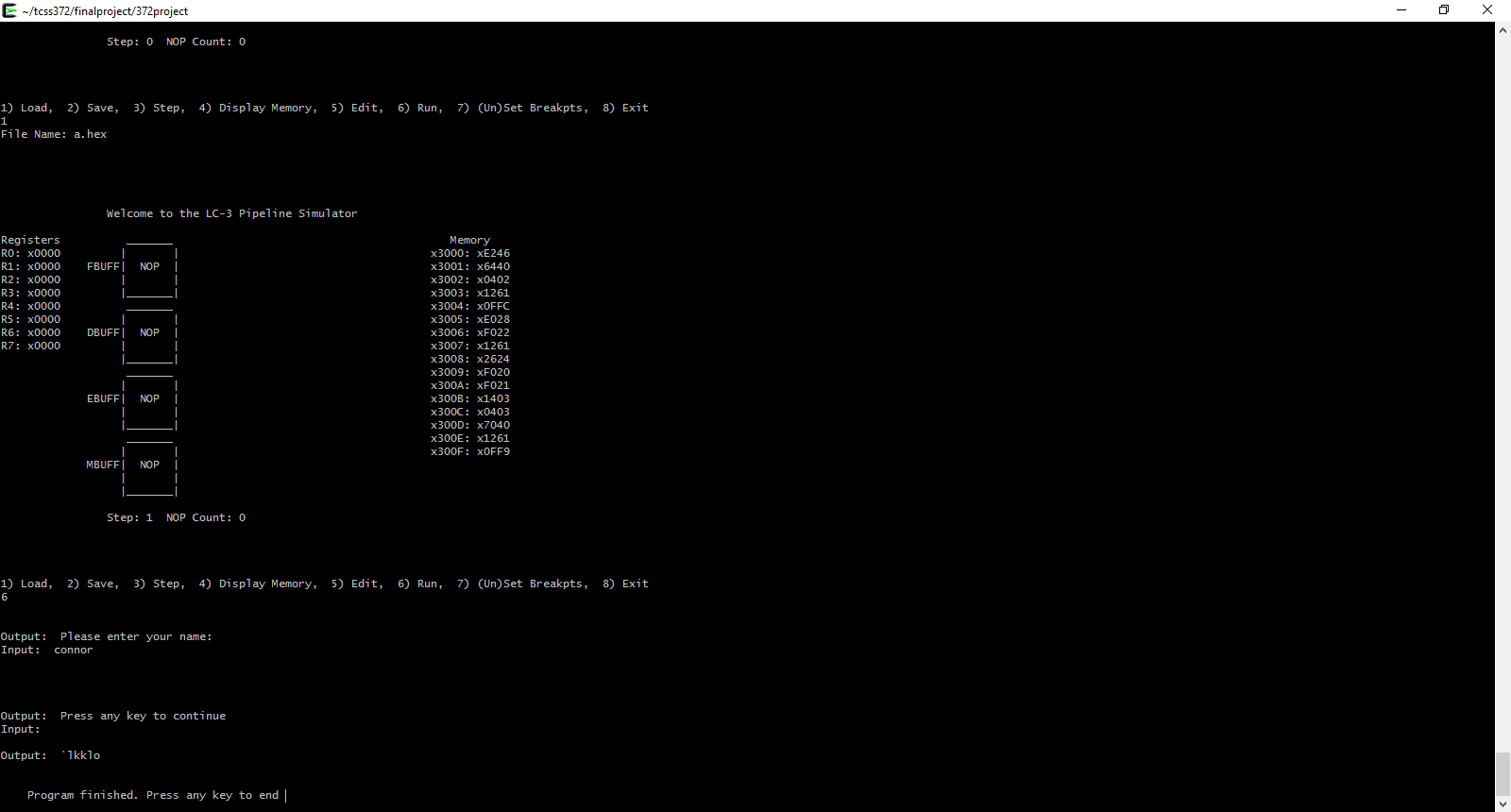
After moving to x300F in memory



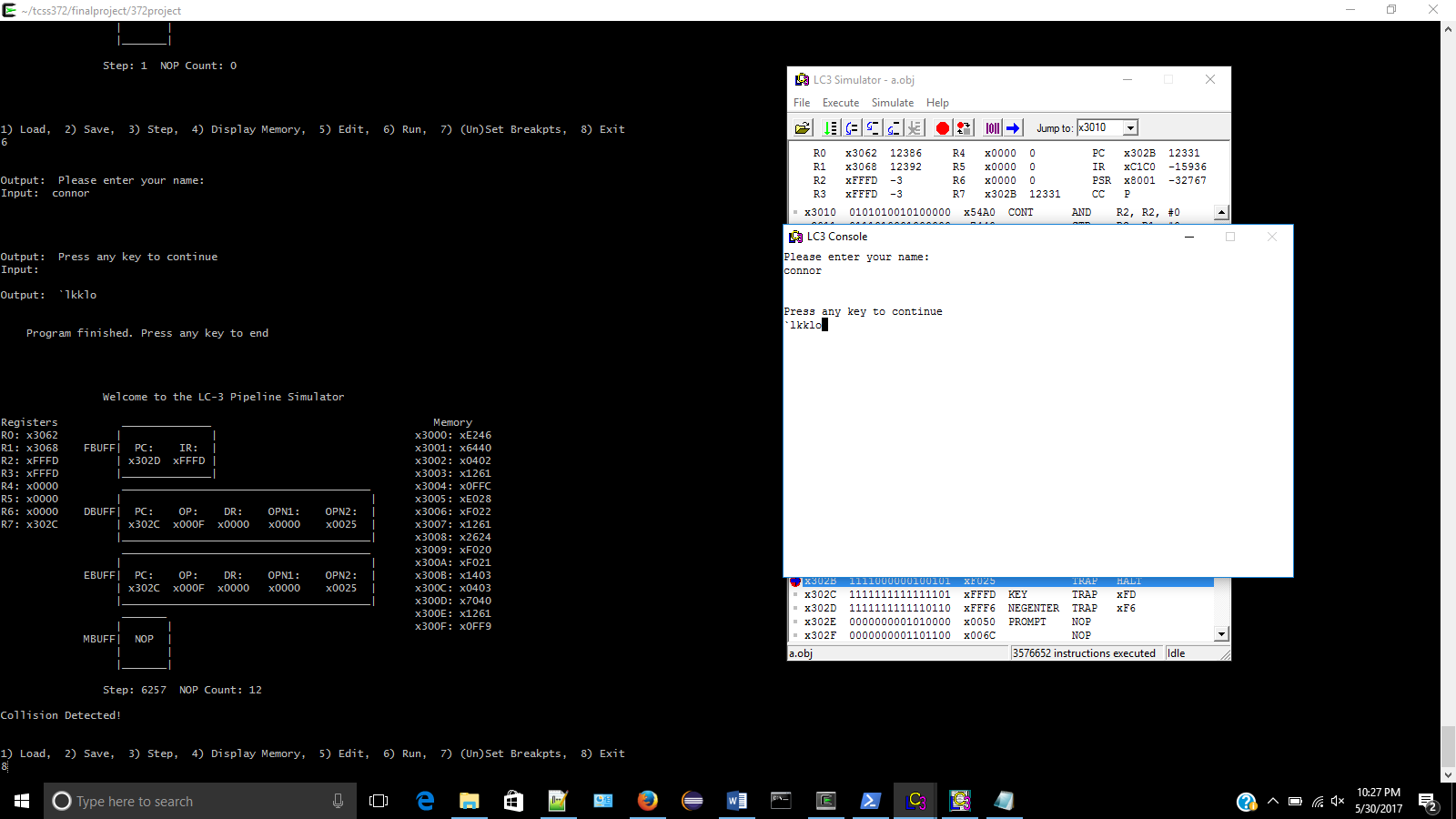
Before running the hex program from HW 4



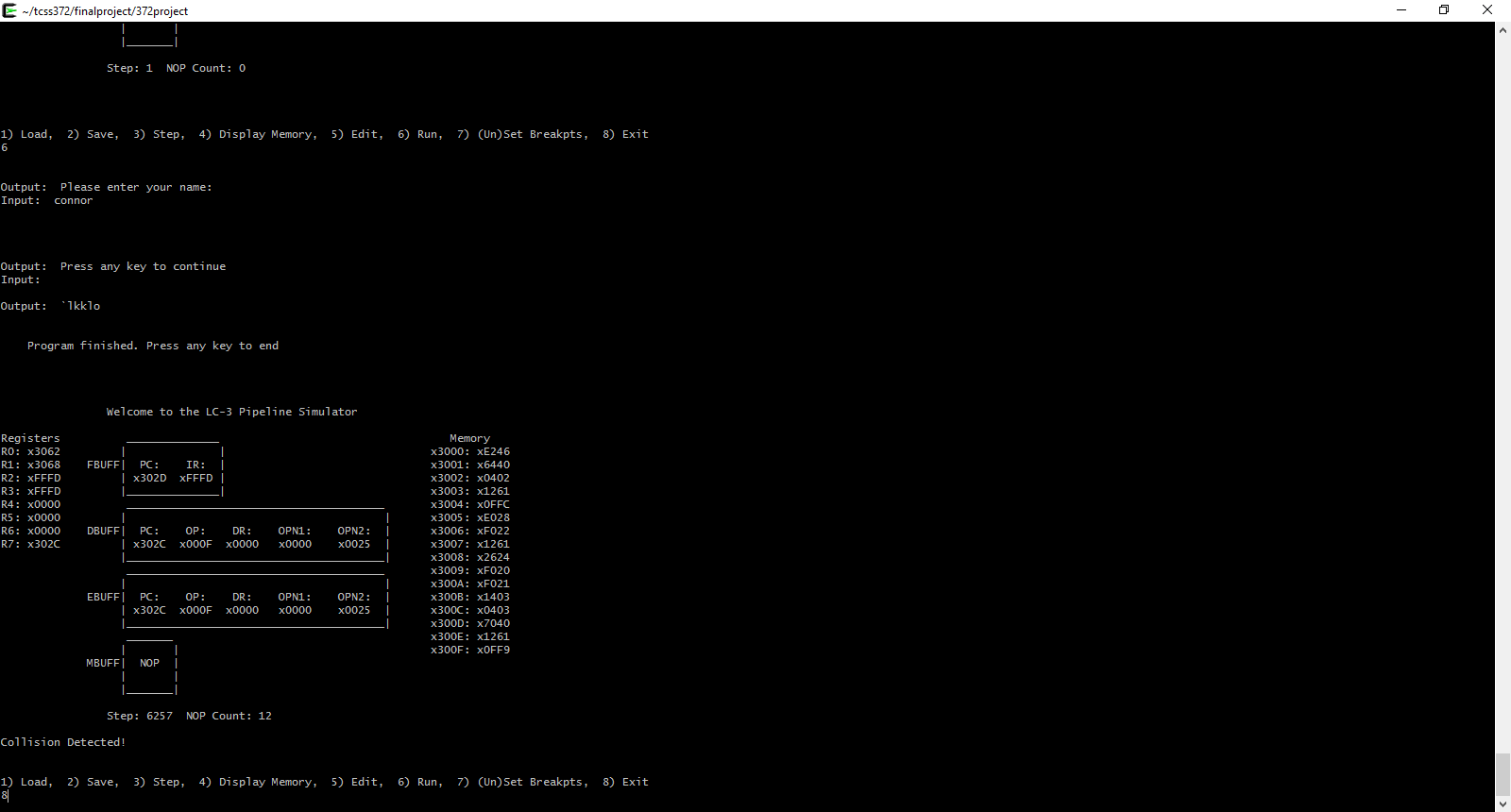
After choosing run and going through the I/O



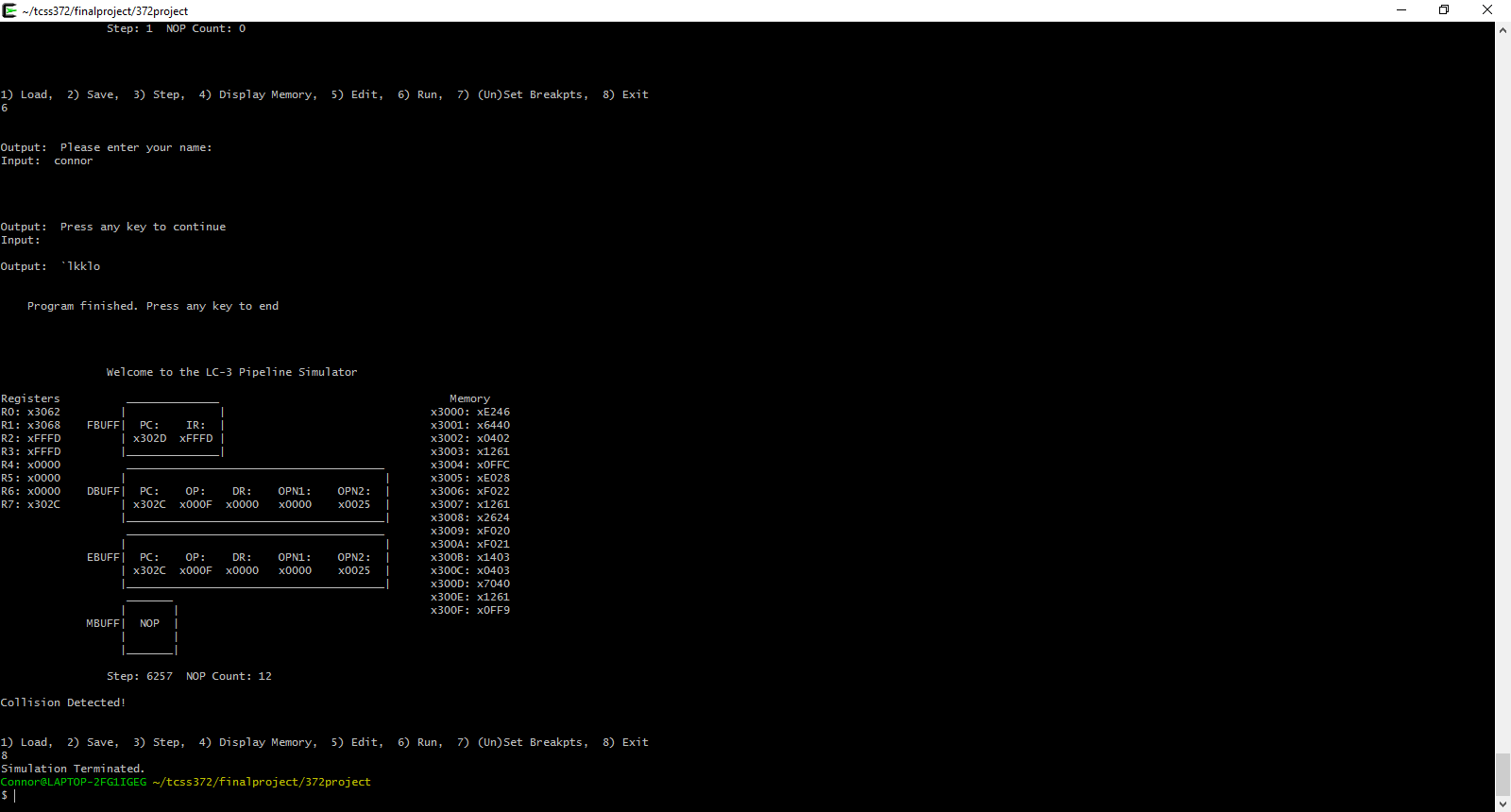
The LC-3 Simulator after running the program (registers included)



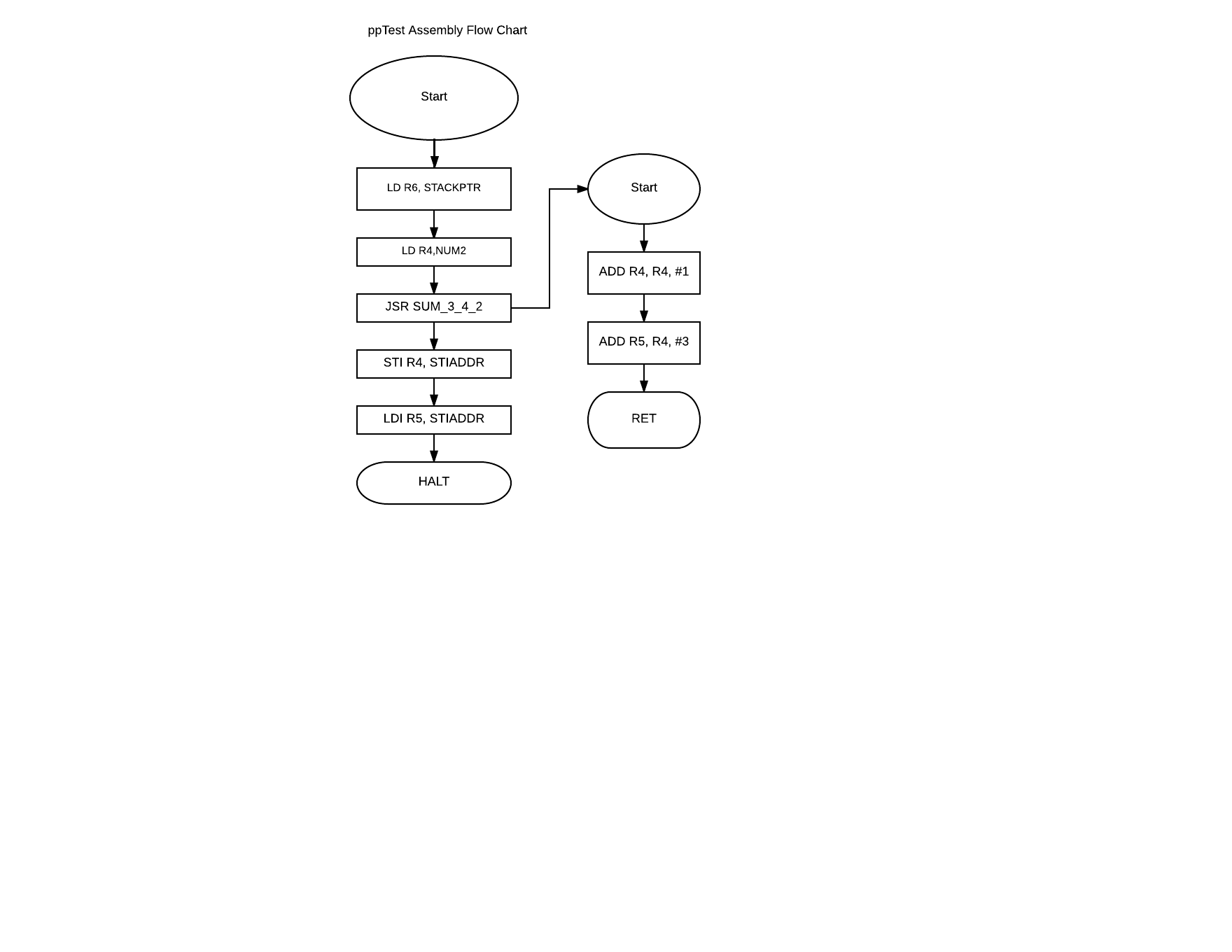
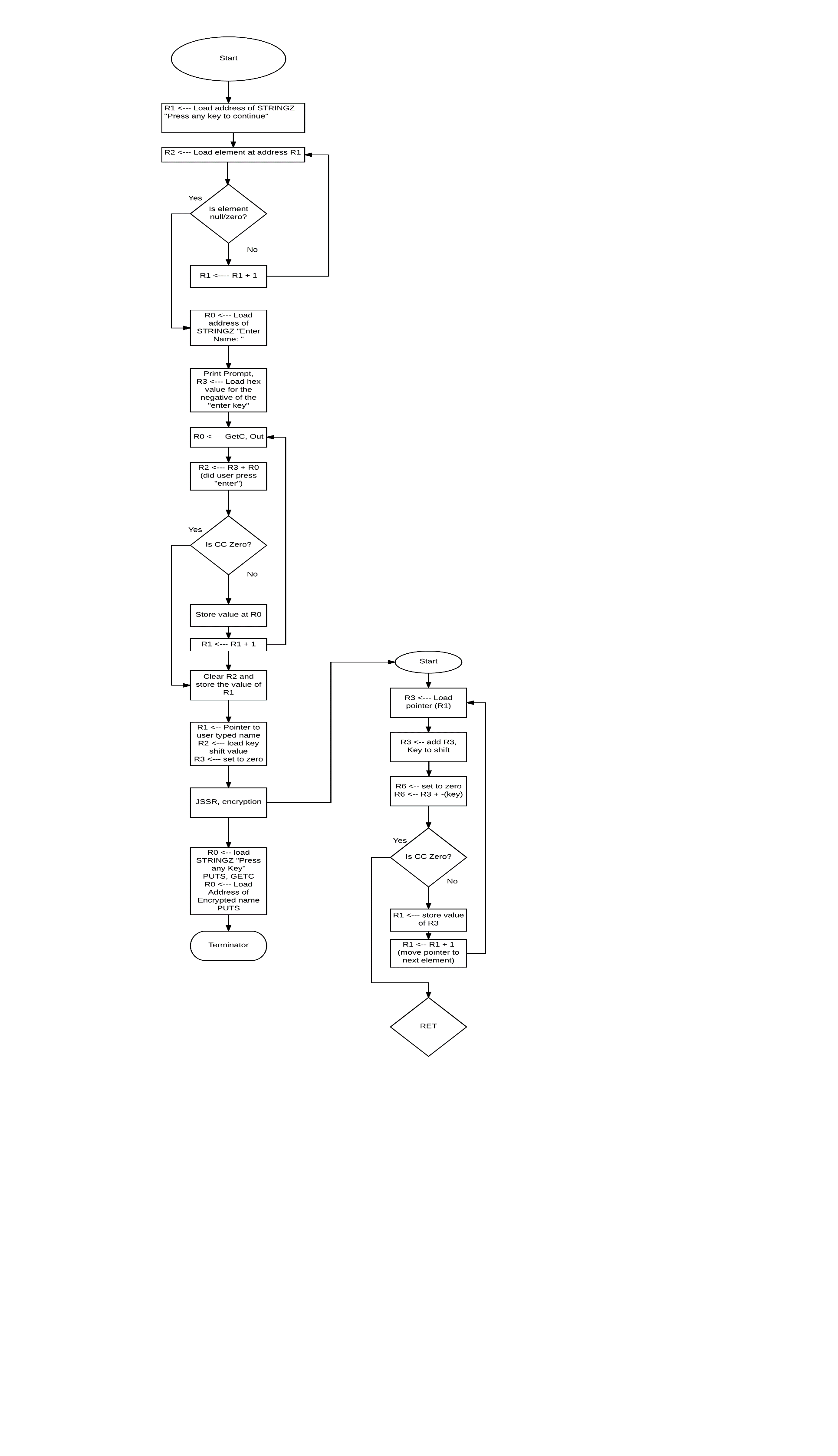
Our Pipeline simulator after running the program (the R7 is only different because we have a breakpoint in the LC-3 to stop the program before it reaches that). This is also what the program looks like before exiting the Simulator.



After exiting the Simulator.



Assembly.hex



This was quite an adventure. We decided early on that we were going to be implementing a Pipeline design into our Controller. It wasn’t easy though. Our first big problem was just trying to repurpose mine (Connor) and Daniel’s project from Homework 4. This was an issue because it was still using the MAR, MDR, PC, etc. registers in order to move data around so when we reversed our Controller (that is, to start with STORE as the first state and go backwards) and tried to add in the buffers, we kept forgetting to replace operations on said registers to the respective buffer at each position. Once we got that working though, we then had to confront the predecoder and accompanying prefetch along with simulating the time it takes to go to memory. Simulating memory access time wasn’t terribly difficult, but it took a bit longer to conceptually understand how the predecoder was able to look through the prefetch to find a collision. But we got it finished and working. After that was the Branch. I would list the JMP and JSRR in here as well but they weren’t nearly as difficult after we got the Branch working correctly. It was just that we were adding two numbers together that weren’t supposed to be added together. From there it was pretty smooth sailing. We got it able to successfully run through the Assembly program from the last assignment without a hitch. Then we started implementing the B grade project (the minimum requirements). Those weren’t terribly difficult either at first. We had a bit of a fire to put out the night of submission as it turns out I (Connor) had overridden some changes James had made with the (un)set breakpoints command on Git so it took some time to go through and fix everything.