Eza Rasheed

er6qt

03-20-19

postlab7.pdf

IBCM is surely an interesting and unique machine code, but unfortunately, it is not meant to be implemented by someone like me. I found it quite difficult, more nitpicky than C++. IBCM requires a lot of attention to detail and all steps must be well thought out before coding, or else it is a major pain to debug. This was a hurdle for TAs as well because everyone's code is so unique to their implementation and so, therefore, half of the TAs I went to for help found it difficult to find my bugs. This ofcourse made IBCM harder for me since I couldn't get specific help like on other assignments. The first complication I went through when coding IBCM for the first time was remembering the hex digits for the specific commands (ex. store, load, add, jmpl). I would always have to look back or would mistakenly type the wrong hex digit for a command, which would mess everything up. Thankfully, though, I memorized them by the end of the tedious in-lab. Another obstacle was changing, adding, or removing instructions when you forgot to do a step, or didn't need a specific step anymore. It required rewriting a lot of the other instructions and modifying many memory addresses. I had to do this a couple of times between the loops in the in-lab bubblesort. Coding loops and jumping was very confusing to me, and took me a good eight hours to do in the in-lab. As for the simulator, I have no critiques. Once I understood how it worked, I found it simple to use and I liked being able to step through the instructions like I use to do on the Java Tutor and Python Tutor websites for my previous CS classes. One suggestion I would make, though, is that when you go through the code, it would be helpful to see the updated memory addresses being outputted on the side so that you can see what is being changed as you are stepping through your program. Overall, I feel alright about IBCM. I still need more time to get use to its syntax, and definitely need to brush up on it before the exam.