Justin Choi (jc8mc)
March 23rd 2017
CS 2150-106, Lab 7: IBCM

postlab7.pdf

I thought IBCM was a bit tough and hard since it was very new and I have never studied machine language, which is a very low-level representation of a program. Although there are about 15 opcodes that can be directly translated to higher-level pseudo code or C/C++ code, I personally had a hard time translating some parts of the pseudo code into machine code, IBCM. There are some limitations in IBCM such as loops and conditionals. The variable declaration part and constant part was easy enough since it followed a similar pattern to declaring variables in C/C++. However, I found the loops and conditionals to be a bit tedious and difficult since I had to trace by hand and see if it yielded the same results as a higher-level code in IBCM. If there was a direct opcode for loops and conditionals I think coding in IBCM would be more efficient and easier.

Although it was hard to translate loops and conditionals into opcode, I think it was an interesting learning experience since I could understand what was really going on in a loop and a conditional, under the hood.

The simulator made my life easier since I did not have to install unnecessary compilers or interpreters. However, I ran into problems when my .ibcm file was running. For example, when I was testing my IBCM code, if I had the wrong logic, and it came to a halt at an unexpected location, then it would just freeze the browser, making it impossible to know what happened at first. If it were to have capabilities like the clang++ compiler, which is more user-friendly since it directs you where and which code made the error, I think it would have been easier to use and fixing / debugging would have been easier.

Although there was slight learning curve to IBCM, I feel a bit better writing IBCM code. While you can write almost everything in IBCM, it would take a lot more time since if you want to fix one minor issue, you have shift all of the lines by one, which can be very tedious and error-prone. I would rather stick to writing in higher-level code.