CSCE 451 Homework 1

Taylor Williamson

125007948

1/27/2020

Report

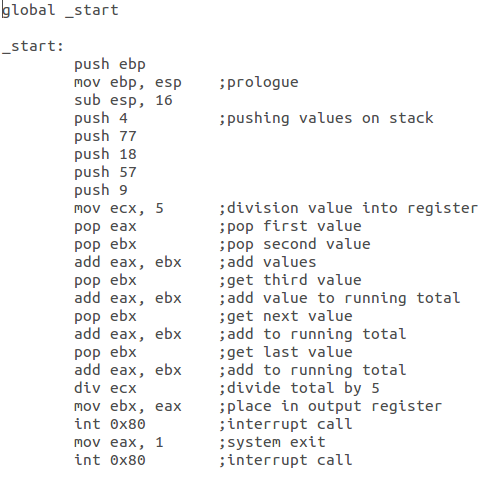
In the process of completing this assignment I had gone through several trials and tribulations in getting things to work as they should. I was frustrated at first as I believed that my logic was there and I was still getting segmentation fault errors. After looking at several examples from the book, online, and from the course so far, I could not seem to find what was wrong. I eventually discovered that my lack of an epilogue was the source of my issue. This would be my only critique of the course and this assignment so far, as there has only been one example (the Hello World example) where the proper epilogue had been demonstrated. Furthermore, it has been expressed through the lab examples that the prologue was more important when it did not seem that way when working on this assignment.

With those problems aside, I was able to successfully implement my logic into a working program. As I worked through the problems, I was confused about how I was supposed to have multiple parts in one file as when the file is executed, it only shows the final state of the output register. I know a print function can be called but I was still pretty confused on how that worked and didn’t want to mess with it. I struggled for a while with how I should be segmenting the code across the files. At first I attempted a file per part of the problem, but when I got to problem 3 I realized this wasn’t possible and also went back to combine all parts of problem 2 into the same file. To test code correctness, I used echo to output the return value at the completion of each problem section.

As I worked through the logic, I found myself utilizing loops excessively and that I was using them as a crutch and over simplifying them when I can combine and optimize their functionality. Ultimately, it took me a few tries and some reworkings of my logic to get a logical control flow that I was happy with. I found that stepping back and thinking about the code pattern helps me optimize and cut the fat off of what I would write if I just started hacking away at it. Furthermore, I found myself running out of registers, as I felt the need to add every non-stack value to a register, especially with the bounds of the loops. This was problematic as I quickly ran out of general purpose registers to use. I realized that the fixed bound of the loop had no need to be stored in a register and could just be compared directly with the dynamic register value.

Ultimately I enjoyed this assignment and look forward to enhancing my ability and understanding of assembly programming. I did have an understanding of assembly from a previous course, but I’m glad I was able to refresh those concepts as well as get good hands-on experience with real practice exercises. Now that I can draw insight from the point of view of the programmer, I believe my understanding of existing code will see great improvement.

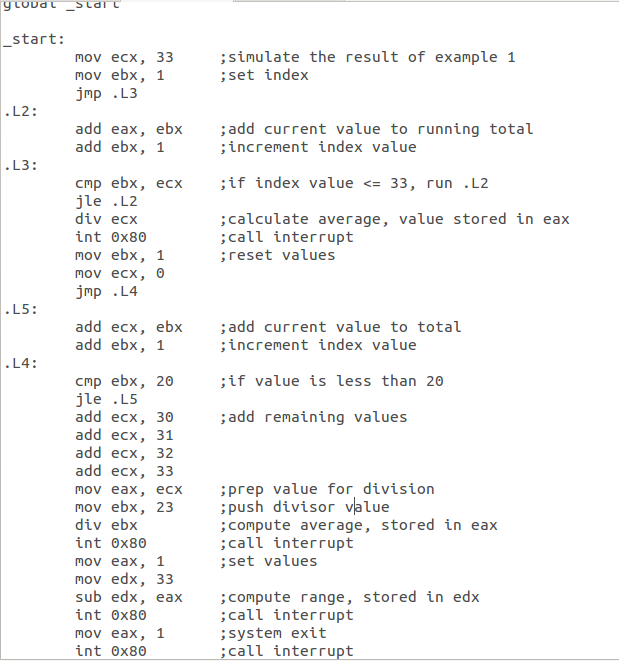
Example 1



To run this assembly code, I utilized the following commands:

* nasm -f elf32 ex1.asm -o ex1.o ; compile the program utilizing NASM
* ld -m elf\_i386 ex1.o -o ex1 ; execute the linking process
* ./ex1 ; execute the fully compiled program
* echo $? ; display result in terminal

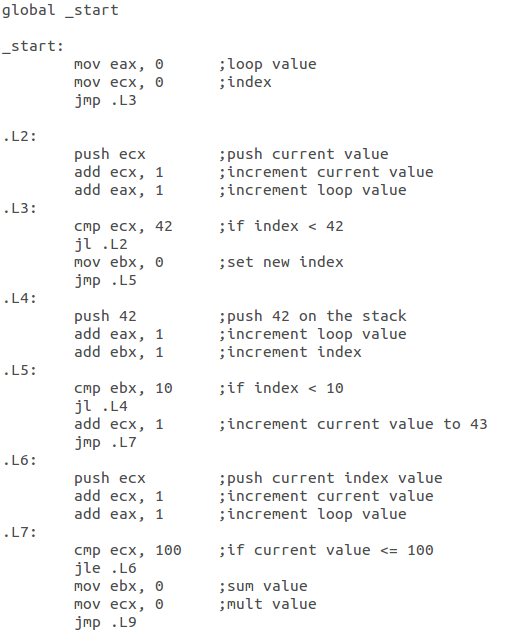
Example 2

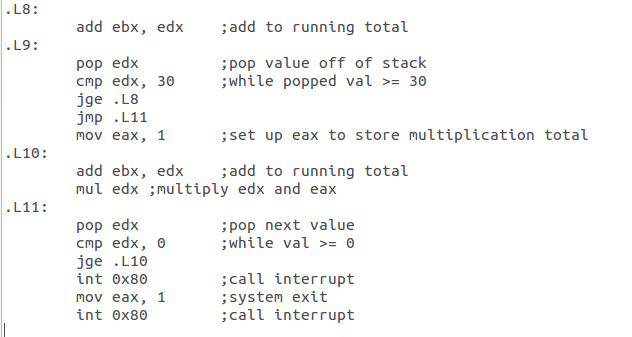


To run this assembly code, I utilized the following commands:

* nasm -f elf32 ex2.asm -o ex2.o ; compile the program utilizing NASM
* ld -m elf\_i386 ex2.o -o ex2 ; execute the linking process
* ./ex2 ; execute the fully compiled program
* echo $? ; display result in terminal to check return values

Example 3





To run this assembly code, I utilized the following commands:

* nasm -f elf32 ex3.asm -o ex3.o ; compile the program utilizing NASM
* ld -m elf\_i386 ex3.o -o ex3 ; execute the linking process
* ./ex3 ; execute the fully compiled program
* echo $? ; display result in terminal to check return values