

SU 2020 CSE 2421 LAB5

Due Date: Sunday, 7/26/2020 11:30 p.m.

IMPORTANT: READ THESE INSTRUCTIONS AND FOLLOW THEM CAREFULLY.

Objectives:

- learn how to use gdb with an assembler program
- learn how different register sizes affect results
- understand how the movzXX and movsXX instructions work
- understand how the pushX and popX instructions work (and which ones don't)
- watch condition flags being set
- watch shifting (all 3 kinds) happen

REMINDERS and GRADING CRITERIA:

- This is an individual lab. No partners are permitted
- You should aim to always hand an assignment in on time or early. If you are late, you will receive 75% of your earned points for the designated grade as long as the assignment is submitted by 11:30 pm the following day, based on the due date given above. If you are more than 24 hours late, you will receive a zero for the assignment and your assignment will not be graded at all.
- Since this lab includes the source code, you do not have to worry about assembler errors or warnings.

<https://powcoder.com>

Add WeChat powcoder

LAB DESCRIPTION

1. From Bryant/O'Halloran textbook, read the first paragraph of section 3.3 (Data Formats) and check out Figure 3.1. Note the different choices for the suffix of assembler instructions. Then, read section 3.4.2 with special attention to the information in Figures 3.4, 3.5, and 3.6. In addition, take a look at Figure 3.10. **If you do not do this reading assignment, the rest of this lab will be harder to figure out.**
2. Remember we said in class that the **movX** (where X is q, l, w, or b) instruction is actually more like a copy instruction than a move.
3. Download from Carmen a copy of the file lab5.s and then do the following:
 - A. Create a lab5Readme file from the template supplied on Carmen.
 - B. Use `gcc -g -lc -m64 -o lab5 lab5.s` to assemble the code.
 - C. **Create as large a window on stdlinux as you can.** This is important.
 - D. Have a copy of the lab5Readme file beside you so that you can fill in the values of the registers as each instruction executes. **IMPORTANT:** Remember that you are being asked what the values are for particular registers ***AFTER*** the instruction on the line executes.

- E. EVEN MORE IMPORTANT: **specify all register values in hexadecimal unless another format is specifically asked for.**
- F. Bring up gdb debugger using the command ***gdb lab5.***
- G. Set a breakpoint using: **break Label1**
- H. Start the program using: **run**
- I. When the program stops, say **tui reg general** so that you can see all of the registers.
- J. Use the **step** or **next** command to move forward one assembler instruction at a time. As you do so, note how the values in the **%rip** register change.

*Hint: you can use **stepi** and **nexti** to traverse the assembly code instead of step and next in the debugger.*

- K. You should be able to see the contents of all 16 integer registers and how they change as you execute each assembler instruction.
- L. Note the suffix of each instruction as it executes as well as the way each specific register is referenced. Then, write the contents of %rax (or %rdx), in hexadecimal within the lab5Readme file in the appropriate location. Keep in mind that **%rax** is an 8 byte register, so each value you record should be an **8 byte value**. **Take care to recognize unprinted leading 0's if the gdb display does not print an 8 byte value. (i.e. 0xffffffff as a value in an 8 byte register is really 0x00000000ffffffff).** Note what changes within each 8 byte register as instructions with suffixes other than 'q' are used.
- M. Delete the comment # in front of the first 4 x86-64 instructions and try to reassemble the program. What happens?
- N. Put the comment # back in front of those first 4 x86-64 instructions and then delete the comment # in front of the second 4 x86-64 instructions. Try to reassemble the program. What happens? Is the error message consistent? What do you think this means with respect to the push/pop instructions with suffixes 'b' and 'l'?
- O. Answer the questions/write the required paragraphs at the end of the lab5Readme template.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

REQUIREMENTS

Create a lab5Readme file using the lab5Readme template available on Carmen. Complete the answers to all of the questions within the comments on the right side of each instruction and also the questions at the bottom of the file.

LAB SUBMISSION

For lab 5 the only file that you have to submit is lab5Readme. A copy of the lab5Readme template will be available on Carmen.