**CSE 2421 LAB 4**

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
* watch condition flags being set

**Reminders and Grading Criteria**

* This is an individual lab. No partners are permitted
* Since this lab includes the source code, you do not have to worry about assembler errors or warnings.

**Lab Description**

1. Pull out your copy of Bryant/O’Halloran and review section 3.4.2 with special attention to the

information in Figures 3.4, 3.5, and 3.6.

2. Download from Carmen a copy of the file lab4.s and then do the following:

A. Create a lab4Readme file from the template supplied on Carmen.

B. Use gcc –g –lc –m64 –o lab4 lab4.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 lab4Readme 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. Bring up gdb debugger using the command **gdb lab4**.

F. Set a breakpoint using: **break Label1**

G. Start the program using: **run**

H. When the program stops, say **tui reg general** so that you can see all of the registers.

I. 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.

J. You should be able to see the contents of all 16 integer registers and how they change as you

execute each assembler instruction.

K. 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

lab4Readme 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.

L. Delete the comment # in front of the first 4 x86-64 instructions and try to reassemble the

program. What happens?

M. 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 error message consistent? What do you think this means with respect to

the push/pop instructions with those suffixes?

N. Answer the questions/write the required paragraphs at the end of the lab4Readme template.

**Requirements**

Create a lab4Readme file using the following as a template. The template is all the text within the two

solid lines. A copy of this template will be available on Carmen.

**Lab Submission**

For lab 4 the only file that you have to submit is lab4Readme. A copy of the lab4Readme

template will be available on Carmen, so you can submit a Word or PDF version of it after you

complete the questions. A PDF version is preferred.