CEG 3310/5310 – Computer Organization

Lab 6 – TRAPs and the Runtime Stack

Learning Objectives

- Learn how to write your own trap service routines
- Learn how to write more complex programs that utilize the runtime stack

Overview

Using the LC-3 simulator, you will construct an assembly-level program that prompts for options when working with a linked list, like printing, adding, and removing items from the list. A sample execution and equivalent C program are included. The printing of integers and retrieving of integers will be done using custom TRAPs that you write.

Part 1

To implement input.asm you must do the following:

- You must read a user's input of 2 digit decimal numbers. You are encouraged to reuse code you wrote for Lab 3.
- Once you have the decimal value you must load the final value into R0
- For example: if a user types the ASCII characters “31” then at the end of your TRAP call R0 must contain the value “0x001F” in hexadecimal.
- This TRAP call should function properly even after being called multiple times, you will need to call this TRAP function multiple times to complete this lab.

To implement output.asm you must do the following:

- Using R0 you must print the decimal value loaded into R0 to the console
- For example if the hex value in R0 is “0x0028” then the console window should display the characters “40”

To allow these trap routines to be useful, you must add them to the trap vector table.

- To add a trap service routine to the trap vector table, it must have it’s own unique starting address (x4000 for input.asm and x5000 for output.asm)
- This starting address must be added to the trap vector table (x40 for input and x41 for output)
- You must submit multiple programs for your final lab submission

Part 2

Once your trap service routines are added to the trap vector table, you can implement the main program loop of your linked list program.

- Setup your runtime stack for the main function as described in the sample C code
    - Do not create custom subroutines for printf, scanf, malloc, or free. You must still be able to print information to the user and read the user's input.
- Prompt the user to make a selection from the choices of print, add, remove, or quit.
- If the user selects print, add, or remove setup your runtime stack to call the functions as described in the sample C code.
    - Implement the assembly equivalent of the functions for print, add, and remove, from the C code as subroutines.
    - Any required arguments must be passed to the functions through the runtime stack.
- If the user quits, pop your runtime stack and halt the processor.

Note

- You will have to submit multiple programs (at least 3 independent programs required i.e. main, input, and output. Using more than 3 programs is acceptable.)
- The add and remove functions must accept a double pointer (a pointer to a pointer) for your linked list. This allows removal of the head of the linked list and the creation of the head of the linked list when the head is nonexistent from within a subroutine.

GRADING

This lab is worth 5.00 points, distributed as follows:

Task	**Points**
Successfully implement input trap service routine	1.000
Successfully implement output trap service routine	1.000
Successfully implement print subroutine	1.000
Successfully implement add subroutine	1.000
Successfully implement remove subroutine	1.000
Total	5.00