CSC B58 0
Third lab practice session – Assembly Programming

Before we start working, here’s a quick reminder of the rules in effect while in the Embedded Systems Lab.

In-lab rules:

We have limited space, and you will be working with sensitive electronic components. Therefore:

- ***No food or drinks are allowed inside the lab.*** Be sure to eat properly before your session, you can step out for water if you need to. Notify your TA since he/she is the only person that can grant you access to the lab.
- ***No laptops, notebook computers, tablets, or other electronic gizmos*** allowed during the session.
- ***Bags and backpacks must be stored at the front or back of the room***.
- ***While in the lab, you’re responsible for taking good care of all equipment***.
- ***Treat everyone else in the lab with respect and consideration***.
- ***You must listen and follow all instructions provided by your TA***.

How your work will be marked:

Your TA will observe your work during the three hour period.

- 25% of the grade for this lab is given by attending the session and working hard .
- 75% is given for completing your work. This includes the parts of the report due ***before*** the session begins, your in-lab work, and the completed hand-in.

While each member of the team must individually show the TA they have completed the work due at the start of the lab. Each team will hand-in a ***single*** *completed handout.*

If you encounter any problems with the software or hardware, bring this to the attention of your TA immediately. If no solution can be found quickly, your TA may have to have you join another team for the duration of the session.

Third lab practice session – Assembly Programming

This lab is designed to Let you gain experience programming in MIPS 32 assembly. The code You will produce here is a bit more complex than what we have seen on the lectures, and the goal is to let you think about more challenging assembly coding problems in an environment where there is plenty of help available.

Please make sure you leave the lab feeling very comfortable with the main concepts involved in assembly programming, you will need this for assignment 3, and for the final exam.

Learning Objectives:

You will strengthen your understanding of memory organization

You will practice assembly programming concepts including: ***general program structure, memory management, system calls, conditional statements, loops, and pointers***

You will learn to use pointers as the main tool for accessing and modifying information

You will understand the relationship between variables, addresses, and memory contents

Skills Developed:

Writing MIPS 32 assembly programs

Using the SPIM simulator to test and debug assembly code

Thinking in terms of dynamic memory management, pointers, and addresses

Reference material:

This handout. ***Remember to hand one completed handout to your TA at the end of the session***

The links to MIPS 32 and SPIM information provided on the course web page

Student 1 Name (last,first): ________________________________

Student 1 student number: ________________________________

Student 2 Name (last, first): ________________________________

Student 2 student number: ________________________________

Lab session date and time:

You must complete all parts marked as 'prior to the lab session' before your appointed lab section starts.

Problem 1 - A linked-list to store strings

Call your program: linkedList_studentNo1_studentNo2.s

Your task here is to implement a program to create and manage a linked-list of strings. Your program should implement the following pseudo code:

1) The main procedure should

```
- Get a string from the user  - see 2)
Loop while the user's string is not empty:
    - Create a new linked-list node for this string  - see 3)
    - Store the string at the head of a linked-list
    - Prompt the user for another string

- Print all the strings in the linked list  - see 4)
```

2) Getting a string from the user

- This should be a ***separate function***. It must be called as a function.
- It ***must allocate*** memory for the new string using a system call
- Then it must prompt the user to enter a string, which will be stored in the newly allocated buffer
- The function should make the pointer to the buffer available to the main process somehow

Problem 1 - A linked-list to store strings (continued)

3) Allocating a new linked-list node

- Linked list nodes have two components:
  * A ***pointer to a string*** (this is provided by the function that you wrote for 2)).
  * A ***pointer to the next node*** in the linked-list

- Your function should ***allocate memory for a linked list node***. Then fill-in the values for the ***pointer to string, and pointer to the next node***.

The pointer to the string should already be available, the pointer to the next node is not, and should be initialized to something reasonable.

Note: The main procedure will be in charge of calling both the function that reads the strings, and the function that creates new linked-list nodes. Then ***it will do all the required linking***.

4) Printing the strings

This should be the easy part. Just traverse the linked list you created and print out all the strings

5) Think about clean-up

In the space below (or back of the page) explain ***in-detail*** what data allocated by your program should be released at the end, and how this would be done.

TA check – Verify the program works as specified: □

Part 2 - Challenging! do not work on this until your linked list is working perfectly

Call this program: sortedLL_studentNo1_studentNo2.s

Add to your linked list a new ***function*** that sorts the strings in the linked list ***alphabetically by the first letter in the string*** (don`t worry about ties being printed in the wrong order).

Your program should now print the strings in sorted order at the end. There are many ways to sort information, you are free to use any method you like.

TA check – Verify that the sorting works: □

CSC B58
Third lab practice session – Assembly Programming

Questions to be answered prior to the lab session:

1) What are the instructions on SPIM that are used to call a subroutine and to return from a subroutine?

2) What system call is used to allocate memory? what are the input parameters and return Values?

3) What system call is used to read a string from keyboard? Explain how it works (what the input and output parameters mean)

4) What system call is used to print strings? What are the input parameters?

5) Write pseudo-code for inserting a node at the head of a linked list

6) Write a simple MIPS 32 routine to allocate space for the new linked-list node. How much space is needed? what will you store there?

Wrapping up (10 minutes before the end of the session)

Make sure you have a completed handout to give your TA.

Submit your two programs by email with the subject line

'***CSCB58, Lab 3'*** (without the quotes!)

Inside the email, note the name and student numbers of each team member.

Don't forget to attach the .zip file!

13) Post-session feedback. Feel free to use simple yes/no answers, or to add comments where you feel they are relevant. Discuss as needed within each team.

- Did you complete the lab exercise in the 3-hour allotted time span?
  (if no, what did you complete, and what was left to do by the time the lab ended?)

- Did the exercise help you understand better each of the following:
  (please answer yes / no / somewhat)

  * Using functions in MIPS 32

  * Allocating memory and managing it using pointers

  * General program flow control (loops and conditional statements)

  * How to structure and implement a complex MIPS 32 program

- How confident are you after the session that you can manage data via pointers in other contexts? (e.g. programming in C)
  (very confident / somewhat confident / not confident)

- Any general comments about the session that you would like to share?