CS224 - Spring 2019 - Lab #3 (Version 2: March 7, 5:12 pm)

(Relatively Big) MIPS Assembly Language Programming Recursion, Floating Point Numbers, Linked Lists

Dates: Section 1, Monday, 11 March, 13:40-17:30
Section 2, Wednesday, 13 March, 13:40-17:30
Section 3, Tuesday, 12 March, 13:40-17:30
Section 4, Thursday, 14 March, 13:40-17:30
Section 5, Friday, 15 March, 8:40-12:30
Section 6, Friday, 15 March, 13:40-17:30
Lab Location: EA-Z04

Purpose: More experience with MIPS assembly language programming using recursion, and linked lists. Gaining experience in MIPS program analysis by adding more to the utilities of an existing program. Floating point numbers problem solving.

Summary
Part 1 (30 points): Learning the principles of recursive programming and the implementation of dynamic link list data structure by program analysis in MIPS assembly language. Floating point number representation.

Part 2 (70 points): More experience with linked lists and recursive programming.

Refer to the linked list program given in the folder that contains this document. It can be used for creating and printing a linked list. Extend that program as suggested below.

TRY TO FINISH PART 2 BEFORE COMING TO THE LAB. MAKE SURE THAT YOU DO THE DEMO TO YOUR TA.

DUE DATE OF PART 1: SAME FOR ALL SECTIONS:

1. Please bring and drop your printed preliminary work into the box(es) provided in front of TA office room number EA-407 by 13:30 on Monday March 11.

2. Please **upload your programs of Part 1 (PRELIMINARY WORK)** to the Unilica by 13:30 on Monday March 11. Use filename **StudentID_FirstName_LastName_SecNo_PRELIM_LabNo.txt** Upload only the programs. Only a NOTEPAD FILE (txt file) is accepted. Your paper submission for preliminary work must match MOSS submission.

No late submission will be accepted.

DUE DATE PART 2: (different for each section) YOUR LAB DAY
You have to demonstrate your lab work to the TA for grade by **12:15** in the morning lab and by **17:15** in the afternoon lab. Your TAs may give further instructions on this. If you wait idly and show your work last minute, 20 points may be taken off from your grade.

At the conclusion of the demo for getting your grade, you will **upload your lab (only lab) program work** to the Unilica Assignment, for similarity testing by MOSS. See Part 3 below for details.

For further possible instructions etc. please follow the possible updates on the Unilica course web page. In the lab your TAs may also give further instructions.

Part 1. Preliminary Work / Preliminary Design Report (30 points)

You have to provide a neat presentation prepared by Word or a word processor with similar output quality such as Latex as you were asked in Lab 1. At the top of the paper on left provide the following information and staple all papers. In this part provide the program listings with proper identification (please make sure that this info is there for proper grading of your work, otherwise some points will be taken off).
CS224
Section No.: Enter your section no. here
Spring 2019
Lab No.
Your Full Name/Bilkent ID

For the linked list programs only upload the utilities that you have implemented do not upload the program segments provided to you.

1. **(10 points) recursiveDivision**: Write a recursive MIPS subprogram that performs integer division of two positive numbers by successive subtractions. As a challenge you may want to implement it to return the quotient in $a0 and remainder in $a1. You may assume that inputs are OK, i.e. two positive numbers as expected. Likewise in the following programs also assume that input is OK.

2. **(10 points) multiplyDigits**: Write a recursive MIPS subprogram that finds the multiplication of the digits of a positive integer. For example for 127 it returns 14.

3. **Delete_x (10 points)**: Study the linked list program provided and the linked list explanation provided below in Part 2. Delete all elements from the linked list with the value x: the pointer to the linked list is passed in $a0, and the integer value of the element to be deleted is given in $a1. Return the number of counted nodes in $v0. Return the list head pointer in $v1. Are you able to return the deleted node(s) back to the heap? If not include a comment in the program to explain why.

4. **Floating Point Numbers Problem Solving (0 points, Optional)**:

   For each show your work briefly.

a. **Convert** the following negative number - 125.25 to IEEE 754 standard
   Single precision representation:
   Double precision representation:

b. **Convert** the above decimal number to another floating point representation. In this representation For single precision use bias of 125:
   For double precision it use bias of 1021:

c. **Consider** the following 32 bit hexadecimal number 0xc2030000. Assuming that it is a
   Floating point number give its decimal equivalent:
   Integer number give its decimal equivalent:

d. **Give** an example of your own to show that when you add a small floating number and a large floating number you may lose precision. In your example use decimal numbers in normalized form and assume that you can only have three digits after the decimal point.

Part 2. MIPS: Creating Linked List Utility Routines (70 points)

In this part, you will write the utility functions defined below for linked lists in MIPS assembly language. Your utilities will be combined with other utility programs such as create_list and display_list, and called by a main program. [Any code other than the you are asked to write in this lab will be provided (see the Unilica folder)—you don’t need to write it]

In memory, the linked lists your utilities will work with are implemented as follows: each element consists of 2 parts: pointerToNext, and value. Each part is a 32-bit MIPS word in memory. The two parts are located in successive word addresses, with the pointerToNext being first. For example, if the byte address of the pointerToNext

is 100, then the byte address of the value will be 104. *Remember, MIPS memory is byte addressable.*

In the last element of the linked list, the pointerToNext has value 0, in other words it is the null pointer. This means that there is not any next element. *Do not forget that the last element still has a value.*

Write the following linked list utility routines. Follow MIPS programming conventions in terms of using the stack and registers. Ignore the other methods not implemented. Modify the menu to include the following utilities.

1. **Insert_n (10 points):** Insert an element to the linked list as the nth element (n > 1): the pointer to the linked list is passed in $a0, and the integer value of the new element to be inserted is given in $a1. The register $a2 contains the value of n. The first element is in position 1, the second element in position 2, etc. If there is no nth element it is inserted to the end of the linked list. This utility will request space in memory from the system with syscall 9 and use it to create a new element, then insert the new element correctly into the linked list. The pointer to the head of the linked list should be the same as it was before the call. If the original linked list is empty no insertion is done. Return value in $v0= 0 if successful if the insertion is done, -1 otherwise.
2. **AddNodes (10 points):** Add the contents (value part) of the node, whose position is given as a parameter, with the value of the next consecutive node and delete the consecutive one. For example, for the linked list <1, 5, 7, 3, 17> if the node position is given as 2, after using the utility the linked list becomes <1, 12, 3, 17>. Return value in $v0 =0 if operation is successful, -1 otherwise. Why -1: the indicated node number may not exist, etc.
3. **SwapNodes (10 points)**: Swap two consecutive nodes. Note that this is not swapping values, it is swapping nodes. The node number of the initial node is given as the input of the utility. Return value in $v0 =0 if successful if the insertion is done, -1 otherwise.
4. **CountCommonValues (10 points)**: Find the number of common values in two sorted linked lists. For example, for the lists <1, 3, 4, 8, 9> and <0, 3, 5, 9, 17, 19> it returns 2, since the common values are 3 and 9.
5. **Delete_n (15 points)**: Delete an element from the linked list at position n: the pointer to the linked list is passed in $a0, and the position of the element to be deleted is given in $a1. If there is no nth character it must delete the **LAST** element of the linked list. Return value in $v0 =0 if successful, -1 if not (this happens if the linked list is empty). In either case, the return value in $v1 contains the pointer to the head of the linked list.

6. **FindSumInRange (15 points)**: Find the summation of the linked list elements within a value range by writing a **recursive** program. For example, if the value range is [5, 10] for the following linked list <1, 7, 5, 2, 10, 2> this utility returns 22.

   For other interesting linked list utilities visit https://www.geeksforgeeks.org/data-structures/linked-list/ Knowing and implementing some of them may be useful in your future job interviews.

Part 3. Submit your code for MOSS similarity testing

1. Submit your MIPS codes for similarity testing to the Unilica > Assignment specific for your section.
2. You will upload one file and only the **LAB work**. Use filename **StudentID_FirstName_LastName_SecNo_LAB_LabNo.txt**
3. Upload only the programs.
4. Only a NOTEPAD FILE (txt file) is accepted.
5. Be sure that the file contains exactly and only the codes which are specifically detailed in Part 2, only the lab work. Check the specifications! *Even if you didn't finish, or didn't get the MIPS codes working, you must submit your code to the Unilica Assignment for similarity checking.*
6. Your codes will be compared against all the other codes in the class, by the MOSS program, to determine how similar it is (as an indication of plagiarism). So be sure that the code you submit is code that you actually wrote yourself!
7. All students must upload their code to Unilica > Assignment while the TA watches.
8. Submissions made without the TA observing will be deleted, resulting in a lab score of 0.

Part 4. Cleanup

1. After saving any files that you might want to have in the future to your own storage device, erase all the files you created from the computer in the lab.
2. When applicable put back all the hardware, boards, wires, tools, etc where they came from.
3. Clean up your lab desk, to leave it completely clean and ready for the next group who will come.

Part 5. Lab Policies

1. You can do the lab only in your section. Missing your section time and doing in another day is not allowed.
2. Students will earn their own individual lab grade. The questions asked by the TA will have an effect on your individual lab score.
3. Lab score will be reduced to 0 if the code is not submitted for similarity testing, or if it is plagiarized. MOSS-testing will be done, to determine similarity rates. Trivial changes to code will not hide plagiarism from MOSS—the algorithm is

quite sophisticated and powerful. Please also note that obviously you should not use any program available on the web, or in a book, etc. since MOSS will find it. The use of the ideas we discussed in the classroom is not a problem.

4. You must be in lab, working on the lab, from the time lab starts until your work is finished and you leave.
5. No cell phone usage during lab.
6. Internet usage is permitted only to lab-related technical sites.
7. For labs that involve hardware for design you will always use the same board provided to you by the lab engineer.