CS224 - Spring 2018 - Lab #3 (Version 1: March 3, 1:01 pm)

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

Dates: Section 1, Monday, 12 March, 13:40-17:30
Section 3, Tuesday, 13 March, 13:40-17:30
Section 2, Wednesday, 14 March, 13:40-17:30
Section 4, Thursday, 15 March, 13:40-17:30
Section 5, Friday, 16 March, 8:40-12:30
Section 6, Friday, 16 March, 13:40-17:30

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

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.
Part 2 (70 points): Understanding bit manipulation operations by representing an integer number as a single precision floating point number and more experience with linked lists and recursive programming.

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

DUE DATE/TIME OF PART 1 --SAME FOR ALL SECTIONS

a. Please drop your written Preliminary Design Report into the box provided in front of the lab by 13:40 on Monday March 12. No late submissions will be accepted!
b. Please **upload your Preliminary Design Report** to the Unilica Assignment for Preliminary Work by 13:40 on Monday March 12. Use filename **name_surname_SecNo_PRELIM.txt**

DUE TIME OF PART 2—DIFFERENT FOR EACH SECTION:

a. You have to demonstrate your Part 2 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 your TA may not accept it. Make sure that you follow your TAs' instructions.
b. At the conclusion of the demo for getting your grade, you will **upload your lab work** to the Unilica Assignment, for similarity testing by MOSS. Please see the related section below for further instructions on MOSS submission.

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. 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.: ...
Spring 2018
Lab No.:
Your Full Name/Bilkent ID

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

1. **(10 points) recursiveMultiplication**: Write a recursive MIPS subprogram that performs multiplication of two positive numbers by successive additions. Note that 5 X 4= 5 + 5 + 5 + 5. Error check is optional. For **recursiveMultiplication** and **recursivesummation** provide a simple user interface to test them under the same main program..

2. **(10 points) recursiveSummation**: Write a recursive MIPS subprograms that finds the summation of numbers from 1 to N. Error check is optional.

3. **Delete_x (10 points)**: Study the linked list program provided and the linked list explanation provided below in Part 3. Delete an element from the linked list with 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. If there is more than one element containing the value x, the first occurrence of the value is the element to be deleted. Return value in $v0 =0 if successful, -1 if not. In either case, the return value in $v1 contains the pointer to the head of the linked list. Are you able to return the deleted node back to the heap? If not include a comment in the program to explain why. Modify the given linked list program to include this option.

Part 2. Converting An Integer Number To Floating Point Number (15 points)

Write an interactive program that asks the user to enter a positive integer and number converts this number to floating point representation by using bit manipulations and prints both integer number and its floating point equivalent. Provide a user interface that allows the user enter integers as long as the user wants to continue. For example, if the user enters 10 which is 1010 in binary, you have to remember that its scientific notation is 1.010×2^3 hence in a register like $t0 you have to construct the bit pattern "0**1000 0010** 01000000000000000000000" sign bit:0, exponent: **1000 0010**, mantissa: 01000000000000000000000. Note that in the exponent we use excess 127 notation by following the IEEE 754 standard and it is $127= 7F_{16}$, $7F + 3= 82_{16}$, in binary it is **1000 0010**. Use proper subprograms and

make sure that you use the conventions of MIPS programming by properly using the stack and registers as needed. As an extension you may accept negative integers too.

Hint: In order to print an integer number stored in a variable, as a floating point number you may use the following code segment.

```
        l.s     $f0, a  # $f.. are floating point registers
        cvt.s.w $f12, $f0
        li      $v0, 2
        syscall  # Prints 10.0
        .data
    a: .word 10
```

Part 3. MIPS: Creating Linked List Utility Routines (55 points)

Linked lists are important dynamic data structures, useful to a variety of algorithms because of their ability to grow and shrink. Utilities libraries for linked lists are therefore useful, so that common functions can be available to users, pre-written, tested and ready to go. In this lab, 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 possibilities.

1. **Insert_end (15 points):** Insert an element to the linked list at end: 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. This utility will request space in memory from the operating system, use it to create a new element, and then insert the new element correctly into the linked list at the end. Upon return, the value in $v0 =0 if successful, -1 if not. In either case, the pointer to the head of the linked list should be the same as it was before the call.

2. **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. The first element is in position 1, the second element in position 2, etc. Return value in $v0 =0 if successful, -1 if not. In either case, the return value in $v1 contains the pointer to the head of the linked list.

3. **Display_Reverse_Order (10 points)**: Display the elements of the linked list in reverse order. This means that if the list contains -> 10 -> 20 -> 30 it displays the list in the reverse order 30, 20, 10. This must be a non-recursive subprogram.

4. **Display_Reverse_Order_Recursively (15 points)**: Display the elements of the linked list in reverse order. This must be a recursive subprogram: it must call itself with jal.

Part 4. Submit your code for MOSS similarity testing

Submit your MIPS codes for similarity testing to the Unilica > Assignment specific for your section. You will upload one file: **name_surname_SecNo_MIPS.txt** created in the relevant parts. Be sure that the file contains exactly and only the codes which are specifically detailed above, including Part 1 programs (your paper submission for preliminary work must match MOSS submission). 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.* 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 ! All students must upload their code to Unilica > Assignment while the TA watches. Submissions made without the TA observing will be deleted, resulting in a lab score of 0.

Part 5. 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.

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.