CS224 - Fall 2016 - Lab #2 (Version 1: Oct. 21, 6:23 pm)

Creating and Running Simple MIPS Assembly Language Programs

Dates: Section 3, Wednesday, 2 November, 08:40-12:30
Section 1, Wednesday, 2 November 13:40-17:30
Section 4, Thursday, 3 November 13:40-17:30
Section 2, Friday, 4 November 13:40-17:30

Purpose: More experience with MIPS programming and subprograms.

ALL GROUPS: Dear students as announced in the classroom please bring and drop your preliminary work into the box provided in front of the lab by 5:00 pm Tuesday November 1.

At the end of the lab before submiting your code for MOSS similarity testing please READ the instructions given here

Students who do not follow the instructions will get 0.

Submit your MIPS codes for similarity testing to the Unilica > Assignment specific for your section. You will upload **one file**: name_surname_MIPS.txt created in the relevant parts. Be sure that the file contains exactly and only the codes which are specifically detailed below. 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).

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

We plan to run your code to make sure that it really works.

You have to show your work to your TA before by 12:00 in the morning lab and by 5:00 in the afternoon lab. Noe that you cannot wait for the last moment to do this. Make sure that you have shown your work to the Lab TA before uploading. If you wait for the last moment and show your work after the deadline time (12 or 5) 30 points will be taken off. If you upload before showing your work to the TA you will receive 0 (zero) for the lab work.

If we suspect that there is cheating we will send the work with the names of the students to the university disciplinary committee.

Part 1. Preliminary Work
(30 points)

You have to provide a neat presentation prepared by Word. 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 / Your Section No.
Fall 2016
Lab No.
Your Name

1. Write separate MIPS functions that

1. receives two numbers and returns its multiplication by successive addition recursively, note that m x n means add n m times.

2. receives an integer value in $a0 and returns it hexadecimal equivalent in a character string whose address is provided in $a1.

For unclear parts make proper assumptions and state them in your comments. The same is also true for the following parts.

Part 2. Recursive Programming
(20 points)

Write a recursive subprogram that receives two numbers and returns the result of division by successive subtractions recursively.

Part 3. MIPS: Creating Linked List Utility Routines
(50 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 and 2 of the 4 utility functions 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 4 utilities listed below will be provided—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.*

For this lab, you will write any two of the following linked list utility routines.

1. **Insert_end:** 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. **Insert_n:** Insert an element to the linked list at position n: the pointer to the linked list is passed in $a0, the integer value of the new element to be inserted is given in $a1, and the position is given in $a2. The first element is in position 1, the second element in position 2, etc. 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 correct position. Upon return, the 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. **Delete_n:** 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.

4. **Delete_x:** 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.

Part 4. Arrays and Subprograms
(0 points) Optional for Additional Challenge

Write a MIPS program with subprograms that provides a user interface. Follow the traditions of MIPS programmers using the stack. (Note that this was a part of Lab 2 and later it was deleted.)

The main program provides a prompt and user enters an asciiz string as an input and after this asks the user what is to be done and for this purpose provides the choices and gets the user choice, repeats asking the users to enter different choices.

1. counting number of smallcase letters in the string,
2. counting number of uppercase letters in the string,
3. replacing all uppercase letters with their "other end equivalent" of the alphabet (other end equivalent is defined as follows: for A it is Z, for B it is Y, for M it is N, and of course for Z it is A etc.).

After having several interactions with the user it generates a histogram of the choices made by the user. For example the user may have used the choice 1 twice and choice 2 four times then in the histogram to indicate the number of times these choices are used in the histogram you may display

choice 1: **

choice 2: ****

choice 3: -

Part 5. Cleanup

1) Erase all the files that you created.
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.