CS224 - Spring 2017 - Lab #2 (Version 3[1], March 5, 9:06 am; Version 2[2]: March 3, 7:37 pm)

Creating and Running Simple MIPS Assembly Language Programs

Dates: Section 1, Monday, 6 March, 13:40-17:30
Section 3, Tuesday, 7 March, 13:40-17:30
Section 2, Wednesday, 8 March, 13:40-17:30
Section 4, Thursday, 9 March, 13:40-17:30
Section 5, Friday, 10 March, 8:40-12:30
Section 6, Friday, 10 March, 13:40-17:30

Purpose: Understanding preliminary principles of passing arguments to and receiving results from subprograms, and branch and jump instruction code generation.

Summary
Part 1 (30 points): Preliminary Report/Preliminary Design Report: Learning principles of writing subprograms. Involves converting octal number to decimal. Generating object code for beq, bne, and j instructions.
Part 2 (15 points): Learning principles of writing subprograms. Involves getting the elements of an integer array from user, sorting numbers, finding the min, max and median values, providing user interface.

DUE DATE OF PART 1: SAME FOR ALL SECTIONS Dear students please bring and drop your preliminary work into the box provided in front of the lab before 12:59 pm on Monday March 6. ~~For late submissions there will be a substantial penalty.~~ No late submissions will be accepted. Please **upload your preliminary work** for MOSS similarity by 14:00 o'clock on Monday March 6. Please see Part 3 for further instructions on MOSS submission.

LAB WORK SUBMISSION TIMING: You have to show your lab work to your TAs by **12:15** in the morning lab and by **17:15** in the afternoon lab. Your TAs may give further instructions on this. Please upload your lab work for MOSS similarity testing and follow the instructions of your TAs.

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

[1] Changes are highlighted: involves correction of two obvious typos. If you see any other such obvious typos in this and in the upcoming lab assignments please correct without asking.
[2] Change is highlighted.

You have to provide a neat presentation prepared by a word processor. 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 2017
Lab No.
Your Full Name/Bilkent ID

1. **(20 points)** Write MIPS assembly language programs as described below.

a. **(10 points) convertToDec**: Write a subprogram, called convertToDec, that receives the beginning address of an asciiz string that contains a octal number in the form of a string, for example, like "17", and returns its decimal ($17_8 = 15_{10}$) equivalent in register $v0. If the number stored in the input string is not a proper octal number it returns a negative value in $v0. A sketch of this convertToSubprogram is as follows.

```
__start:
            ...
            la   $a0, octalNo
            jal   convertToDec
    # result comes in $v0
            ...
    convertToDec:
            .data
    octalNo:   .asciiz "17"
```

b. **(10 points) interactWithUser**: Write a subprogram, called interactWithUser, that asks the user enter an octal number in the form of a string, makes sure that it is a proper octal number if not generates an error message and ensures that a proper input is received. It passes this address to the subprogram defined above convertToDec, if necessary modify it, and gets the result from it and returns the decimal value back to the caller, i.e. the main program. The main program is the one that contains the label __start . How to read a string: See MIPS system calls on the web to understand how to read a string. Invoke interactWithUser from the main program that contain the label __start.

Use the $s registers during the implementation of the above subprograms. Using $s registers means that you have to save them to stack and restore them back from stack. (Question: Why do we use the $s registers? Answer: To have practice about the use of stack in MIPS.) Make sure that you also save/restore any other register that need to be saved.

2. **(10 points)** Generating machine instructions

(**Branch instructions 3 points each, jump instruction 4 points**) Give the object code in hexadecimal for the following branch (be, bne) and jump (j) instructions.

```
            ... # some other instructions
again:      add ... # there is an instruction here and meaning is insignificant
            add ... # likewise for the other similar cases
            beq   $t0, $t1, next
            bne  $t0, $t1, again
            add ...
            add ...
            add ...
next:       j         again
```

Assume that the label again is located at memory location 10 01 00 20_{16}. If you think that you do not have enough information to generate the code explain. See slide number 117 in the new Chap 6 slides of the textbook for the MIPS memory map (available at our unilica web site, Documents folder).

Part 2. Writing MIPS assembly language programs (70 points)

In this part when needed please use the stack.

1. **(10 points)** Write a subprogram, called readArray, that asks the user the size of an integer array and get the values from the user in a loop. Returns the beginning address of the array in ~~$a0~~ $v0, and array size in terms of number of integers in ~~$a1~~ $v1.

Hint.

```
    li $a0 8 #enough space for two integers
    li $v0 9 #syscall 9 (sbrk)
    syscall
# address of the allocated space is now in $v0
```

Extra challenge (optional): Rather than getting the values from the user, generate random integer numbers with the maximum value of 100. For an even more challenge use byte size integers rather than word size, and therefore use byte oriented instructions when needed.

Note: For the optional extra challenge no extra credit will be given. Please do this optional part after doing what is required above.

2. **(25 points) bubbleSort**: Write a subprogram, called bubbleSort, that sorts an integer array in increasing/ascending order using the bubble sort algorithm. The subprogram receives the beginning

address of the array in $a0, and the array size in $a1. The array size can be 1 or more, this also applies to other subprograms.

3. **(10 points) minMax**: Write a subprogram, called minMax, that returns the minimum and maximum numbers of an integer array. The input array may or may not be sorted. Use $a registers for passing parameters and use $v0 and $v1 to return the min and max values.

4. **(10 points) median**: Write a subprogram, called median, to return the median value of a sorted array. It receives array address and array size.

5. **(15 pts.) monitor**: Write a monitor program that provides a user interface to use the above subprograms in an interactive manner. The main program, the one that contains __start, provides a simple user interface that will use the above subprograms in the way that you imagine. A simple interface is enough if you like you may make it fancy.

Part 3. 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 Part 1 and Part 2, 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 Part 2 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 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.

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.