CS224 - Spring 2018 - Lab #2 (Version 2: Feb. 23, 6:57 pm)

Creating and Running Simple MIPS Assembly Language Programs

Dates: Section 1, Monday, 26 February, 13:40-17:30
Section 3, Tuesday, 27 February, 13:40-17:30
Section 2, Wednesday, 28 February, 13:40-17:30
Section 4, Thursday, 1 March, 13:40-17:30
Section 5, Friday, 2 March, 8:40-12:30
Section 6, Friday, 2 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 and call conventions. Involves converting hexadecimal number to decimal. Generating object code for j instructions.
Part 2 (70 points): Learning principles of writing subprograms. Involves generating the elements of an integer array using a random number generator, sorting numbers...

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 February 26. 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 February 26. 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

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

a. **(10 points) convertHexToDec**: Write a subprogram, called convertHexToDec, that receives the beginning address of an asciiz string, in register $a0, that contains a hexadecimal number in the form of a string, for example, like "1A", and returns its decimal ($1A_{16}$= 26_{10}) equivalent in register $v0. If the number stored in the input string is not a proper hexadecimal number it returns -1 in $v0. A sketch of this convertHexToDec is as follows.

```
            .text
              ...
              la   $a0, hexNo
              jal   convertHexToDec
    # result comes in $v0
              ...
    convertHexToDec:
              ...
              .data
    hexNo:   .asciiz "1A"
```

b. **(10 points) interactWithUser**: Write a subprogram, called interactWithUser, that asks the user enter an hexadecimal number in the form of a string, makes sure that it is a proper hexadecimal number if not generates an error message and ensures that a proper input is received. It passes this address to the subprogram defined above convertHexToDec, 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 comes right after the assembler directive .text. 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.

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
(**jump instructions 10 points**) Give the object code in hexadecimal for the following jump (j) instructions.

```
            ...
previous:   nop
            nop
            nop
            nop
            j       next
            nop
            add     $t0 ,$t1, $t2
next:       j       previous
```

Assume that the label previous is located at memory location 10 01 00 $1C_{16}$. If you think that you do not have enough information to generate the code explain. See the textbook and 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 initializes the array entries with random numbers between 0 and 100. Returns the beginning address of the array in $a0, and array size in terms of number of integers in $a1. For random number generation see the related syscall, stackoverflow etc.

Hint.

```
li $a0, 8 #enough space for two integers
li $v0, 9 #syscall 9 (sbrk: allocate heap memory)
syscall
```

address of the allocated space is now in $v0.

Make sure that you display the contents of the array with proper headings etc. For this write another subprogram.

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) noOfUniqueElements**: Write a subprogram, called noOfUniqueElements, to return the number of unique elements of an array. The input array can be sorted or unsorted. It receives array address and array size. For example the array 1, 3, 1, 5, 6 contains 4 unique elements.

5. **(15 pts.) monitor**: Write a monitor program, i.e. a program that controls subprograms, that provides a user interface to use the above subprograms in an interactive manner. The main program, 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 with error checks.

Extra challenge (optional: brings no extra credit): To challenge yourself use byte size integers rather than word size, and therefore use byte oriented instructions when needed. If you choose this way just do everything with byte instructions: one version no word version.

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 to Part 5, 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 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.