CS224 - Fall 2019 - Lab #2 (Version 2: October 16, 16:22)

Creating and Running Simple MIPS Assembly Language Programs

Dates: Section 1, Wednesday, 23 October, 13:40-17:30
Section 2, Friday, 25 October, 13:40-17:30
Section 3, Wednesday, 23 October, 8:40-12:30
Section 4, Thursday, 24 October, 13:40-17:30
Section 1, 2, 3, 4 Prelim. Report Deadline: Wednesday October 23, 10:40
Lab Location: EA-Z04

NO LATE SUBMISSION IS ACCEPTED

Purpose: 1. Understanding preliminary principles of using stack for saving s registers, passing arguments to and receiving results from subprograms, and dynamic storage allocation. 2. Code generation for branch, jump, load address and load word.

You are obliged to read this document word by word and are responsible for the mistakes you make by not following the rules. Your programs should be reasonably documented and must have a neat syntax in terms of variable names and spacing. In all labs if it is not specified assume that inputs are always correct.

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, j, la, lw instructions.
Part 2 (70 points): Learning principles of writing subprograms. Involves dynamic storage allocation for arrays, getting the elements of an integer array from user, sorting numbers, finding the min, max and mode values, and providing a user interface.

DUE DATE OF PART 1: SAME FOR ALL SECTIONS:

a. Please bring and drop your hardcopy (printed copy) of the preliminary work into the box(es) provided in the lab by 10:40 on Wednesday October 23, 2019.
Please **upload your programs of Part 1 (PRELIMINARY WORK)** to the Unilica by 10:40 on Wednesday October 23. 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. Any format other than txt gets 0 points.

b. To get credit for preliminary work you have to submit its hard copy and upload its txt version to unilica. **No late submission will be accepted**.

DUE DATE PART 2-5: (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 entire program work** to the Unilica Assignment, for similarity testing by MOSS. See Part 6 below for details.

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 / 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
Fall 2019
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 number in base 8 in the form of a string, for example, like "20", and returns its decimal ($20_8 = 16_{10}$) equivalent in register $v0. It assumes that the number passed is a legal octal number. Be sure that the program works for longer octal numbers as well.

A sketch of this convertToDec is as follows.

```
main:
            ...
            la   $a0, octalNo
            jal   convertToDec
    # result comes in $v0
             ...
    # stop execution here by syscall
```

```
convertToDec:
              ...
               jr   $ra
              .data
octalNo:   .asciiz "20"
```

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 it generates an error message and ensures that a proper input is received. It passes this address to the subprogram defined above convertToDec, gets the result from it, prints it, and returns the decimal value back to the caller, i.e. the main program. How to read a string: See MIPS system calls on the web to understand how to read a string or use MARS help menu.

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. Make sure that you also save/restore any other register that need to be saved.

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

(**Each instruction: 2 points**) Give the object code in hexadecimal for the following branch (be, bne), jump (j) and load instructions. Note that the meaning of the code segment is insignificant.

```
           ... # some other instructions
again:     add  $t0, $t1, $t2
           add  $t0, $t0, $t3
           add  $t0, $t0, $t4
           add  $t0, $t0, $t5
           beq   $t0, $t6, next
           bne  $t0, $t6, again
           add  $t0, $t0, $t5
next:      j         again
           la    $t0, array2
           lw    $t1, array2
           ...
           .data
array1:    .space   100
array2:    .word    10, 20, 30
```

Assume that the label **again** is located at memory location 00 40 00 A0$_{16}$ and the array **array1** starts at memory location 10 01 00 00$_{16}$. 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. Lab Work: Writing MIPS assembly language programs (70 points)

In this part when needed you have to follow the conventions of professional MIPS programmers and use stack.

1. **(10 points) readArray**: Write a subprogram, called readArray, that asks the user the size of an integer array and gets the values of array members from the user in a loop. It returns the beginning address of the array in $v0, and array size in terms of number of integers in $v1.

Hint.

```
        li $a0, 20 #allocate enough space for 5 integers
        li $v0, 9 #syscall 9 (sbrk)
        syscall
# beginning address of the allocated space is returned in $v0
```

2. **(25 points) bubbleSort**: Write a subprogram, called bubbleSort, that sorts an integer array in decreasing/descending order using the bubble sort algorithm and using the **absolute values of the numbers stored**. (When -5, 1, 4, 2 is sorted after sorting the array contains -5, 4, 2, 1.) The subprogram receives the beginning address of the array in $a0, and the array size in $a1. The array size can be 0 or more.

3. **(10 points) thirdMinThirdMax**: Write a subprogram, called thirdMinThirdMax, that returns the third minimum and third maximum numbers of an integer array. For example, for the array {13, 5, 1, 3, 8, 7} 3th minimum is 5, and 3th maximum is 7. You may assume that the array has at least 3 elements. The input array may or may not be sorted. Use $a registers for passing parameters and use $v0 and $v1 to return the thirdMin and thirdMax values.

4. **(10 points) mode**: Write a subprogram, called mode, to return the mode (the most frequently appearing) value of a sorted array. If there are more than one candidate it returns the minimum of them. It receives array address and array size in $a0 and $a1 registers, respectively.

5. **(5 points) print**: Write a subprogram, called print, to print the content of the array. Use $a registers for passing parameters and don't forget that array size can be 0.

6. **(10 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

1. Submit your MIPS codes for similarity testing to the Unilica > Assignment specific for your section.
2. You will upload one file. 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 Part 1 to Part 6, 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.*
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.