CS224 - Spring 2019 - Lab #2 (Version 1: Feb. 25, 8:45 pm)

MIPS Assembly Language Programming Using Subprograms

Dates: Section 1, Monday, 4 March, 13:40-17:30
Section 2, Wednesday, 6 March, 13:40-17:30
Section 3, Tuesday, 5 March, 13:40-17:30
Section 4, Thursday, 7 March, 13:40-17:30
Section 5, Friday, 8 March, 8:40-12:30
Section 6, Friday, 8 March, 13:40-17:30
Lab Location: EA-Z04

Purpose: Understanding principles of passing arguments to, and receiving results from, subprograms; bit manipulation, experience with dynamic storage allocation, and code generation for control instructions.

Summary
Part 1 (30 points): Preliminary Report/Preliminary Design Report: Learning principles of writing subprograms. Generating object code for control instructions.
Part 2 (70 points): Lab: Mapping a given algorithm to assembly language program.

DUE DATE OF PART 1: SAME FOR ALL SECTIONS:

a. Please bring and drop your printed preliminary work into the box(es) provided in front of TA office room number EA-407 by 13:30 on Monday March 4.
b. Please **upload your programs of Part 1 (PRELIMINARY WORK)** to the Unilica by 13:30 on Monday March 4. Use filename **StudentID_FirstName_LastName_SecNo_PRELIM_LabNo.txt** <u>Upload only the programs. Only a NOTEPAD FILE (txt file) is accepted.</u> Your paper submission for preliminary work must match MOSS submission.

No late submission will be accepted.

DUE DATE PART 2: (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 lab (only lab) program work** to the Unilica Assignment, for similarity testing by MOSS. See Part 3 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.

For further possible instructions etc. please follow the possible updates on the Unilica course web page.

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
Spring 2019
Lab No.
Your Full Name/Bilkent ID

1. Write MIPS assembly language programs as described below.
The main program calls the subprogram interactWithUser and then stops.

Important. Only use the $s registers during the implementation of all subprograms to practice MIPS programming conventions. **Warning**: If you do not follow this requirement you will lose half of the points for that program. Using $s registers in subprograms means that you have to save them to stack and restore them back from stack before going back to the caller. Make sure that you also save/restore any other register that need to be saved.

a. **(2 points) interactWithUser**: Write a subprogram, called interactWithUser, that provides three menu options.

First option asks the user enter an octal number in the form of a string. It passes its address to the subprogram convertToDec, defined below, that gets the decimal equivalent of the number from the subprogram, and prints it.

Second option asks the user enter a number writes the number in hex, then calls the subprogram reverseNumber and prints the reversed number in hex. See the example below.

Third option returns the control to the main program. Otherwise, the control stays in interactWithUser to perform the first or second option.

b. **(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 "14", and returns its decimal ($14_8 = 12_{10}$) equivalent in register $v0.

c. **(10 points) reverseNumber**: Write a subprogram, called reverseNumber, that receives a decimal number (in $a0) and reverses its bytes and returns as its result (in $v0). For example, if the number received is AABBCCDD in hex it returns DDCCBBAA. For hex display see the related syscall. In the implementation of reverseNumber use *shift* and logical bit manipulation instructions such as *and* etc. as needed.

2. **(8 points)** Generating machine instructions. Give the object code in hexadecimal for the be, bne, j, and jr instructions of the following code segment. Briefly show your work.

```
... # some other instructions
```

```
again:
                add ... # there is an instruction here and meaning is insignificant
                add ... # likewise for the other similar cases
                add ...
                add ...
                beq   $t0, $t1, next
                bne   $t2, $t3, again
                add ...
                add ...
                jr     $ra
next:
                j          again
```

Assume that the label again is located at memory location 10 01 00 40_{16}. If you think that you do not have enough information to generate the code, explain why.

Part 2. Lab Work: Writing MIPS assembly language programs using arrays (70 points)

Important. In the implementation of subprograms defined below you can only use $s registers. **Warning**: If you do not follow this requirement you will lose half of the points for that program.

1. **(15 points) readArray**: Write a subprogram, called readArray, that asks the user the size of an integer array and get the values of its members from the user in a loop. It makes sure that only numbers within the range of 0 to 100 [0, 100] are entered. Returns the beginning address of the array in $v0, and array size in terms of number of integers in $v1. The array the dynamically allocated using syscall 9.

Extra challenge (optional): Rather than getting the values from the user, generate random integer numbers in the range of [0, 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) insertionSort**: Write a subprogram, called insertionSort, that sorts an integer array in increasing/ascending order using the insertion 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.

For sorting use the following algorithm (taken from Wikipedia). (If you think that the algorithm is incorrect, fix it.)

```
i ← 1
while i < length(A)
   j ← i
   while j > 0 and A[j-1] > A[j]
```

```
    while j > 0 and A[j-1] > A[j]
        swap A[j] and A[j-1]
        j ← j - 1
    end while
    i ← i + 1
end while
```

3. (15 points) medianMode: Write a subprogram, called medianMode, to return the median value of a sorted integer array. It receives array address and array size. If there is more than one mode it returns the smaller value. For example, the array 3, 3, 4, 4, 5 has two modes 3 and 4; for this array it returns 3.

For definitions of mode and median see https://www.purplemath.com/modules/meanmode.htm

4. (15 pts.) main: Write a main program 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.

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 and only the **LAB work**. 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 in Part 2, only the lab work. 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.

Part 5. 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.

1