CS224 - Fall 2024 - Lab #2 (Version 1: October 7, 18:43)

Writing Subprograms and Array & Bit Processing in MIPS Assembly Language

Dates: (TAs - Tutor)

Section 1: Wed, 9 Oct, 13:30-17:20 in EA-Z04 (TA: Kadri, Soheil; Tutor: ...)
Section 2: Thu, 10 Oct, 13:30-17:20 in EA-Z04 (TA: Onur, Soheil; Tutor: ...)
Section 3: Thu, 10 Oct, 8:30-12:20 in EA-Z04 (TA: Onur, Pouya; Tutor: Mert)
Section 4: Fri, 11 Oct, Fri 13:30-17:20 in EA-Z04 (TA: Berkan, Kadri; Tutor: Umut)

TA Full Name (email address: @bilkent.edu.tr)
Berkan Şahin
M. Kadri Gofralılar (kadri.gofralilar)
Onur Yıldırım (o.yildirim)
Pouya Ghahramanian (ghahramanian)
Sepehr Bakhshi (sepehr.bakhshi) (Coord.)
Soheil Abadifard (soheil.abadifard)

Tutor Full Name (email address: @ug.bilkent.edu.tr)
Deniz Çatakoğlu (deniz.catakoglu)
Mert Emre Yamalı (emre.yamali)
Umut Başar Demir (basar.demir)

Purpose: **1.** Understanding principles of using stack for saving $s and $ra registers. **2.** Passing arguments to and receiving results from subprograms. **3.** Understanding bit manipulation. **4.** Learning dynamic storage allocation and array processing.

Note that Bilkent tries to make sure that students follow international standards in terms of their professional activities such as following the rules. Please do not expect that the rules will be made flexible in nonstandard ways, which are seen in some countries that you may be familiar with.

Summary

Preliminary Work (50 points)

Learning the principles of writing MIPS subprograms, creating dynamic size arrays, accessing array elements. It involves:

1. Generating an array,
   populating it and
   creating a frequency table for array's content.

Lab Work (50 points)

Learning principles of writing subprograms and bit manipulation.

1. Hamming distance calculation.
2. Reversing the bits of a register.

Implementation Notes and Requirements for Subprograms

You are not allowed to use $t registers in the subprograms. In subprograms only use $s registers to get used to the MIPS software development traditions. When you enter to a subprogram save $s registers you use in the subprogram to the stack. When necessary save $ra register too. For passing arguments to subprograms use argument ($a) registers and for returning results to the caller use $v0 and $v1 registers. If you do not follow this rule 10 points will be taken off from your grade.

You can only use a limited number of pseudo instructions in your lab work; such as: lw and sw instructions (like: lw $t0, a; sw $t0, a), load address (like: la $t0, a), and conditional instructions (like: bgt, blt ...). If you do not follow this rule some points will be taken off from your grade.

Important Notes for All Labs About Attendance, Performing and Presenting the Work

You are obliged to read this document word by word and are responsible for the mistakes you make by not following the rules.

1. Not attending to the lab means 0 out of 100 for that lab. If you attend the lab but do not submit the preliminary part you will lose only the points for the preliminary part.
2. Try to complete the lab part at home before coming to the lab. Make sure that you show your work to your TAs and answer their questions to show that you know what you are doing before uploading your lab work and follow the instructions of your TAs.
3. In all labs if you are not told you may assume that inputs are correct.
4. In all labs when needed you have to provide a simple user interface for inputs and outputs.
5. Presentation of your work

   You have to provide a neat presentation prepared in txt form. Your programs must be easy to understand and well structured.

   Provide following six lines at the top of your submission for preliminary and lab work (make sure that you include the course no. CS224, important for ABET documentation).

   CS224
   Lab No.
   Section No.
   Your Full Name
   Bilkent ID
   Date

   Please also make sure that your work is identifiable: In terms of which program corresponds to which part of the lab.
6. **If we suspect that there is cheating we will send the work with the names of the students to the university disciplinary committee. You can experiment with ChatGPT for learning; however, you cannot use/modify the code generated by it. Such an act is classified as plagiarism. Note that MOSS is capable of detecting ChatGPT code. Furthermore remember that, the code you use from**

ChatGPT can also be used by another student in the course. Make sure that the code you submit is really yours and has been internalized.

DUE DATE PRELIMINARY WORK: SAME FOR ALL SECTIONS

NO late submission will be accepted. Please do not try to break this rule and any other rule we set.

a. Please upload your programs of preliminary work to Moodle by 13:30 on Wednesday, 16 October, 2024.

b. Please note that the submission closes sharp at 13:30 and no late submissions will be accepted. You can make resubmissions so do not wait for the last moment. Submit your work earlier and change your submitted work if necessary. Note that only the last submission will be graded.

c. Please familiarize yourself with the Moodle course interface, find the submission entry early, and avoid sending an email like "I cannot see the submission ınterface." (As of now it is not yet opened.)

d. Do not send your work by email attachment they will not be processed. They have to be in the Moodle system to be processed.

e. Use filename **StudentID_FirstName_LastName_SecNo_PRELIM_LabNo.txt** Only a NOTEPAD FILE (txt file) is accepted. Any other form of submission receives 0 (zero).

DUE DATE PART LAB WORK: (different for each section) YOUR LAB DAY

a. You have to demonstrate your lab work to your TA for grading. Do this by **12:00** in the morning lab and by **17:00** in the afternoon lab. Your TAs may give further instructions on this. If you wait idly and show your work last minute, your work may not be graded.

b. At the conclusion of the demo for getting your grade, you will **upload your Lab Work** to the Moodle Assignment, for similarity testing by MOSS. See below for the details of lab work submission.

c. Try to finish all of your lab work before coming to the lab, but make sure that you upload your work after making sure that it is analyzed by your TA and/or you are given the permission by your TA to upload.

Part 1. Preliminary Work (50 points)

1. Find Number of Occurrences. (50 points)

This program involves practice of writing subprograms and dynamic array creation. The subprograms to be written are: **CreateArray**, **InitializeArray**, and **FindFreq**. See the following for their definitions. You may have additional subprograms as needed.

A. Main program invokes **CreateArray**.

The subprogram **CreateArray** performs the following. It

1. Asks the user to enter the size of the array to be created then allocates storage (using syscall 9).

2. Calls another subprogram: **InitializeArray** to initialize the array content with positive integers by using a simple user interface.

3. After initialization it prints the contents of the array (if you like you may write a subprogram for this).

4. Returns array address and array size to Main (in $v0 and $v1 registers).

B. Main invokes **FindFreq**.
The subprogram **FindFreq** performs the following. It
Receives the array address, array size, and address of FreqTable from Main in argument ($a) registers and finds number of times the numbers 0 to 9 appear in the array and stores this frequency information into FreqTable.

FreqTable .word 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Defined in data segment and assume that it is only accessible by Main (its address will be passed to FindFreq).
FreqTable 1st word contains the number times the number 0 appears in the array
...
FreqTable 10th word contains the number times the number 9 appears in the array
FreqTable 11th word contains the number times any number other than 0 to 9 appears in the array

For example, if the array created by the user contains (1, 2, 3, 56, 90, 7, 8, 1); 1 appears twice and any number other than 0 to 9 appears twice.

C. Main prints FreqTable with a proper format. If you prefer you may write a subprogram for this.

Part 2. Lab Work (50 points)

1. Hamming Distance Between Registers (25 points)
Write a subprogram that finds the Hamming Distance between two registers. Your program must provide a subprogram to gets values to be assigned to the registers from the user; display register values in hex and also display the Hamming distance calculated. The user interface should ask the user if user wants to continue etc.

Your program cannot be one single main program you have to use subprograms in your implementation as needed. You have to decide the subprograms. Get the value from the user and provide reasonable output that will enable us to follow your program. Have at least two reasonable subprograms.

For questions regarding the Hamming distance calculation for registers see the examples given in wikipedia.

2. Reverse Register Content (25 points)
Write a program that reverses the content of a register bit by bit. For example, for 0XCF 00 00 01 the reversed number is 0X80 00 00 F3
You have to use subprograms in your implementation as needed. You have to decide the subprograms. Provide reasonable user interface. The user interface should ask the user if user wants to continue with another number. Have at least two reasonable subprograms.

Part 3. Submit Lab Work for MOSS Similarity Testing

1. Submit your Lab Work MIPS codes for similarity testing to Moodle.
2. You will upload one file. Use filename **StudentID_FirstName_LastName_SecNo_LAB_LabNo.txt**
3. Only a NOTEPAD FILE (txt file) is accepted. No txt file upload means you get 0 from the lab. Please note that we have several students and efficiency is important.
4. *Even if you didn't finish, or didn't get the MIPS codes working, you must submit your code to the Moodle Assignment for similarity checking.*
5. Comparison of your programs with other students's programs: The MOSS plagiarism detection tool determines how similar they are to other programs (as an indication of plagiarism). So be sure that the code you submit is code that you actually wrote yourself ! You are not allowed to use web resources that solves the assigned programs.
6. The effectiveness of MOSS for chatbot code is quite good, with a detection rate of much higher than 50%. (The answer is provided by chatbot.)

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. The questions asked by the TA will have an effect on your 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 works for ChatGPT code too. 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.