CS224 - Fall 2020 - Lab #2 (Version 2: October 10, 7:24)

MIPS Programming for Bit Manipulation and Dynamic Array Allocation

Dates:

Section 1: Wednesday, 14 October, 13:30-16:20 in EA-Z04, Only even no. students may come to the lab
Section 2: Friday, 16 October, 13:30-16:20 in EA-Z04, Only even no. students may come to the lab
Section 3: Sunday, 18 October, 8:30-11:20 in EA-Z04, Only even no. students may come to the lab
Section 4: Thusday, 15 October, 13:30-16:20 in EA-Z04, Only odd no. students may come to the lab

TAs:

Section 1 (Wed): Ege Berkay Gülcan, Zülal Bingöl
Section 2 (Fri): Pouya Ghahramanian, Yusuf Dalva
Section 3 (Sun): Yusuf Dalva, Zülal Bingöl
Section 4 (Thu): Berkay Gülcan, Eren Çalık

TA email Addresses:
Berkay Gülcan: berkay.gulcan@bilkent.edu.tr
Eren Çalık: eren.calik@bilkent.edu.tr
Pouya Ghahramanian: ghahramanian@bilkent.edu.tr
Yusuf Dalva: yusuf.dalva@bilkent.edu.tr
Zülal Bingöl: zulal.bingol@bilkent.edu.tr

Lab Attendance Policy:

All labs can be done online. Your id cards can be used to enter to the lab. If you want to come to the physical lab please follow the even/odd day policy: On even numbered days students with an even number student ID can come to the lab. Follow the same for odd numbered days.

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

Important Implementation Requirement for Lab2: In this lab you are not allowed to use $t registers in the subprograms. 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. Remember and use the rules of passing parameters (use $a0, $a1, $a2, $a3 registers as many needed) to subprograms and returning results (using $v0, $v1) to the caller.

For all parts provide a simple console interface that displays messages to the user to get inputs. The user interfaces you provide should keep doing until user wants to quit by a message such as "Do you want to continue? Enter 0 to stop." Learn how to read an integer number: Simple using a syscall.

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.

Summary

Preliminary Work:

Part 1 (25 points): Learning the principles of writing subprograms using bit manipulation.
Part 2 (30 points): Dynamic array processing with simple operations.

Lab Work:

Part 3 (30 points): Dynamic array processing with array compression by deleting array entries.
Part 4 (15 points): Dynamic array processing with array compression by deleting several numbers within a range using the program written for Part 3 as a subprogram.

Lab: (Note try and study lab part at home before coming to the lab. Make sure that you show the lab work to your TA before uploading in the lab.)

DUE DATE OF PART 1 & 2 (PRELIMINARY WORK): SAME FOR ALL SECTIONS

No late submission will be accepted.

a. Please upload your programs of preliminary work to Moodle by 13:30 on Wednesday October 14. You may be given further instruction for uploading please check Moodle course interface.
   Use filename **StudentID_FirstName_LastName_SecNo_PRELIM_LabNo.txt** Upload only the programs. Only a NOTEPAD FILE (txt file) is accepted. Any other form of submission receives 0 (zero).

DUE DATE PART 3 & 4 (LAB WORK): (different for each section) YOUR LAB DAY

a. You have to demonstrate (e.g. by using chat on Zoom) your lab work to the TA for grade by **11:00** in the morning lab and by **16:00** in the afternoon lab. Your TAs may give further instructions on this. If you wait idly and show your work last minute, minimum 20 points will be taken off from your grade. We will see it together how it works with Zoom.
b. At the conclusion of the demo for getting your grade, you will **upload your entire program work including Preliminary Work all parts** to the Moodle Assignment, for similarity testing by MOSS. See Part 5 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 & 2. Preliminary Work (55 points)

You have to provide a neat presentation prepared in txt form. Provide following five 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
Section No.
Your Full Name

Bilkent ID

Make sure that you identify what is what at the beginning of each program by using proper comments.
Important Note - Assume that all Inputs are Correct: In all lab works if it is not explicitly stated you may assume that program inputs are correct.

Part 1. bitComplementor: (25 points) Write a MIPS program on bit processing that
Complements n number of rightmost bits of an input register. The program provides a simple user interface to get the input number to be complemented and the value of n. It should display the input number and the output of the subprogram in hexadecimal and repeat this as long as user wants to continue. Provide a simple console interface.
Example: If the integer input value is in hex 0x000000CC and the value of n is 8 the generated output is 0x00000033. For the same input value if the value of n is 32 the output becomes 0xffffff33.

How to display an integer in hexadecimal: See Mars help menu on syscalls. For the bit manipulation instructions see the textbook and slides.

Part 2. arrayManipulator: (30 points) Write a MIPS program on dynamic array processing that
Performs array operations using the following subprograms in the order given below as long as the user wants to continue. Your main program must invoke the following subprograms in the order they are listed below.
createPopulateArray creates and populates an array of a certain size: the array size and array member values are provided by the user. Note that in MARS there is system call 9 that provides dynamic array allocation (like **new** in C++) but it has no garbage collection capability (**delete**); therefore, don't do anything for garbage collection.
checkdPalinrome checks if an array is a palindrome. You may use the program from your previous assignment. This time it is a subprogram.
~~**countFrequency**. Counts the frequency of each array element. This subprogram, for the array with the following values as given in the order [1, 4, 4,3] returns an array in the following order [1, 2, 2, 1] since the first element 1 appears once and the second and third elements appear twice etc. It returns the address of the created frequency array $v0.~~
countFrequency. Counts the number of occurrences of an array element value, whose index is given in $a2, in the array. The array beginning address and size -number of elements- are passed in $a0 and $a1, respectively.
Example: Consider an array of size 4 containing the values in the following order [1, 4, 4, 3], where at the index position 0 there is 1 and at the index positions 1 and 2 the array contains 4. If we invoke this subprogram $a2 containing 0 it returns 1, since at the index position 0 of the array the value 1 is stored and this value occurs only once in the array. If we invoke the subprogram with the same array and if $a2 contains 1 the subprogram returns 2, since in the array at the index position 1 we have 4 stored and this value (4) appears twice in the array.

Part 3 & 4. Lab Work (45 points)

In the implementation of the following subprograms you may assume that the user populates them with nonnegative numbers.

Part 3. compressArray (30 points) Write a MIPS program on dynamic array processing that

Deletes a certain value from an array generated by createPopulateArray subprogram and returns a newly generated array (pointed by $v0) and its size (in $v1) as the result. The original array is retained. If no occurrence of the number to be deleted is observed it returns the exact copy of the array as the result and its size (in $v1). To the subprogram pass the array beginning address and size in $a0 and $a1, respectively. Pass the value to be deleted in $a2.

Example: For the input array [1, 2, 3, 5, 7, 1, 2] if the number to be deleted is 1 it returns an array pointed by $v0 containing [2, 3, 5, 7, 2] and the size of this newly generated array 5, and this value is returned in $v1.

`

Part 4. compressMultiple: (15 points) Write a MIPS program on dynamic array processing that

Deletes multiple numbers from an array, where the numbers to be deleted are specified by given their range of values. The range is defined by a pair of numbers.

Example: If we want to delete all numbers within the range of 1 to 4, when you call the subprogram $a0: points to the array to be compressed, $a1: contains arrays size, $a2: contains 1 (beginning of the range), and $a2: contains 4 (end of the range). If the array contains [1, 5, 2, 4, 1] for the range defined as above after compression we obtain [5], so $v0 will point to this array and $v1 will contain 1 since the compressed array generated from the original array contains just 1 item. For the implementation you have use the method cm,pressArray by invoking it with the numbers to be deleted one by one.

Part 5. Submit Your Code for MOSS Similarity Testing

1. Submit your 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. Be sure that the file contains exactly and only the codes which are specifically detailed Part 1 to Part 4, yes including preliminary work 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 Moodle Assignment for similarity checking.*
5. 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 !

Part 6. Cleanup for Physical Lab

1. As defined earlier in Lab1 document.

Part 7. Lab Policies for Physical Lab

1. As defined earlier in Lab1 document.