CS224 - Spring 2021 - Lab #2 (Version 1: February 16, 11:33)

MIPS Assembly Language Programming with Subprograms and Bit Manipulation

Dates:
Section 1: Mon, 22 Feb, 8:30-12:20 in EA-Z04
Section 2: Wed, 24 Feb, 13:30-17:20 in EA-Z04
Section 3: Tue, 23 Feb, 13:30-17:20 in EA-Z04
Section 4: Mon, 22 Feb, 13:30-17:20 in EA-Z04
Section 5: Fri, 26 Feb, 8:30-12:20 in EA-Z04
Section 6: Fri, 26 Feb, 13:30-17:20 in EA-Z04

TAs:
Section 1: Zülal Bingöl, Ergün Batuhan Kaynak
Section 2: Zülal Bingöl, Kenan Çağrı Hırlak
Section 3: Alper Şahıstan, Kenan Çağrı Hırlak
Section 4: Ege Berkay Gülcan, Ergün Batuhan Kaynak
Section 5: Hüseyin Eren Çalık, Yusuf Dalva
Section 6: Ziya Erkoç, Yusuf Dalva

TA name (x No of labs) email address:
Alper Şahıstan (x1): alper.sahistan@bilkent.edu.tr
Ege Berkay Gülcan (x1): berkay.gulcan@bilkent.edu.tr
Ergün Batuhan Kaynak (x2): batuhan.kaynak@bilkent.edu.tr
Hüseyin Eren Çalık (x1): eren.calik@bilkent.edu.tr
Kenan Çağrı Hırlak (x2): cagri.hirlak@bilkent.edu.tr
Yusuf Dalva (x2): yusuf.dalva@bilkent.edu.tr
Ziya Erkoç (x1): ziya.erkoc@bilkent.edu.tr
Zülal Bingöl (x2): zulal.bingol@bilkent.edu.tr

Lab Attendance and Rotation Policy Reminder and Time for the Next Lab:
All labs will be done online and attendance is mandatory. You have to attend the online labs and submit the lab work by following the instructions of your TA. Before submission your TA will do a brief interview with you to understand your knowledge of the work you plan to submit. Note that preliminary works are submitted before all labs and are graded only if you attend the lab and submit your lab work.

Next Lab - Lab3 Days: Due to the lab rotation policy the first Lab3 session will be with Section 3 on March 2. The lab days for Sections 2, 5 and 6 are on the following days of the same week. The last Lab3 session will be on March 8 with Section 1 and Section 4. See the course syllabus on Moodle for the lab days.

Purpose: This lab aims learning **1**. Implementation of subprograms. **2.** Use of stack in subprograms. **3**. Dynamic storage allocation. **4.** Bit manipulation.

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 must have a neat syntax in terms of variable names, comments and spacing as you have learned in CS101 and as suggested by in class discussions and the programs in the textbook.

Summary

Preliminary Work:

Part 1 (30 points) **Process Static Array**: Write a program with three subprograms for integer array processing. The subprograms to be written prints an array, checks if the array is symmetric, and finds maximum and minimum elements of an array.

Part 2 (20 points) Reverse Bits of a Number: Write a subprogram that receives a decimal number and reverses the order of its bits and returns the reversed form as its result.

Lab Work:

Part 3 (30 points) Process Dynamic Array: It is similar to Part 1 of Preliminary Work, but this time array size is dynamically determined during execution time. The difference is this time you will have an additional subprogram that gets the array size and array elements from the user.

Part 4 (20 points): Count Bit Pattern: Write a subprogram to count a given bit pattern stored in a register. (Remember: registers store integer numbers in binary form.)

Note try, study, and aim to complete lab part at home before coming to the online 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 9:30 am on Monday Feb 22.

b. Please note that the Moodle submission closes sharp at 9:30 am and no late submissions will be accepted. You can make resubmissions before the system closes, so do not wait the last moment. Submit your work earlier and change your submitted work if necessary. Note that only the last submission will be graded.

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

d. 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 3 & 4 (LAB WORK): (different for each section) YOUR LAB DAY

a. You have to demonstrate (using Zoom) 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 Part 3 & 4** to the Moodle Assignment, for similarity testing by MOSS. See Part 5 below for details.

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

d. We use zoom to track your lab attendance.

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 (50 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 No.
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 Notes

1. **Assume that all Inputs are Correct**: In all lab works, if it is not explicitly stated, you may assume that program inputs are correct.
2. **In the subprograms you are not allowed to use $t registers.** Instead of $t registers use $s registers and make sure that you push $s registers you use to the stack. Before returning to the caller make sure that you restore the values of the $s registers from the stack.

Part 1 (30 points) **Process Static Array**: Write a program with three subprograms for an integer array defined in the data segment. You have to pass array beginning address and the array size to a subprogram in $a0 (la $a0, array), and $a1 (lw $a1, arraySize) registers.

PrintArray: Prints the contents of an array.
CheckSymmetric: Checks if an array is symmetric (returns 1 in $v0 if symmetric, returns 0 otherwise).
FindMinMax: Finds minimum and maximum elements of an array and returns them, respectively, in $v0 and $v1.
Provide a simple user interface when appropriate.

Assume that the variables array and arraySize are defined in the data segment like as follows.

```
array:      .word 10, 20, 30
```

arraySize: .word 3

For parameter passing use the argument registers where $a0: points to the beginning of the array, $a1: number of elements in array. Your program should work for an array of size 0 or more. If the array size is 0, it means that there will be no array definition in the data segment and the contents of $a1 will be equal to 0.)

Part 2 (20 points) Reverse Bits of a Number: Write a subprogram that receives a decimal number (in $a0) and reverses the order of its bits and returns it as its result (in $v0). For example, if the number received in $a0 is 0XAABBCCDD in hex, the subprogram returns 0XDDCCBBAA in $v0. For hex display see the related syscall that prints an integer in hexadecimal notation. In the implementation use *shift* and logical bit manipulation instructions such as *and* etc. as needed. Provide a simple user interface where appropriate.

Part 3 & 4. Lab Work (50 points)

Part 3 (30 points) Process Dynamic Array: It is similar to Part 1. The difference is this time you will have an additional subprogram, with the name getArray, that gets the array size from the user, performs a dynamic storage allocation (dynamic means it is done during execution time) to the array, and gets the array elements from the user. The subprogram getArray returns the array beginning address in $v0 and array size in $v1 to the main. Before going back to main getArray invokes PrintArray to print the array elements. Therefore, getArray is a subprogram that calls another subprogram

The main program receives the array beginning address and array size, respectively, in $v0 and $v1 and invokes CheckSymmetric, FindMinMax and outputs their results with proper output messages. Provide a simple user interface where appropriate.

You have to pass array beginning address and the array size to a subprogram in $a0 and $a1 registers. Your program should work for an array of size 0 or more.

For dynamic array allocation use the syscall 9. Before invoking the syscal,l put the number of bytes to be allocated to $a0, make sure that you have 9 in $v0 then perform syscall. The syscall returns the beginning address of the array into $v0. Consider the example in the following code segment to allocate an array of size 4 words..

```
li  $a0, 16  # The bytes size of the memory location to be allocated
li  $v0, 9   # Used for dynamic storage allocation
syscall      # Perform storage allocation by using values stored in $a0 (memory size) and $v0 (action to be performed by syscall).
add $t0, $zero, $v0  # $t0 <== $v0
# $t0 points to the beginning of the array.
```

Part 4 (20 points): Count Bit Pattern: Write a subprogram to count the number of occurrences of the bit pattern stored in $a0 in $a1. The bit pattern to be searched is stored in the least significant bit positions of $a0, the bit pattern length is given in $a2. Provide a simple user interface in main, display numbers in hex so that it will be easy to see if your program works properly. Initialize register contents by decimal easy to comprehend numbers. For example, for

$a0: 0X00 00 00 FF (integer value 255), **$a1**: 0XFF FF FF FF (integer value -1), and **$a2**: 8

The subprogram returns 4 since 0XFF appears in $a1 four times. This example shows that the bit pattern search will be done in discrete non overlapping steps. Search the bit pattern by considering the least significant bits of $a1, then the next possible bit positions etc. In this example first consider bit positions [0, 7] of $a1 then [8, 15] etc.), perform your pattern matching from right to left (from least significant bit position to most significant bit position).

Aim to complete lab part at home before coming to the online lab.

Part 5. Submit Your Code 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. 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. Lab Policies (Reminder)

1. As indicated in Lab1: Attendance is mandatory and the preliminary work is graded only if you submit your lab work with the observation/permission of your TA.
2. You can do the lab only in your section. Missing your section time and doing in another day is not allowed.
3. The questions asked by the TA will have an effect on your lab score.
4. 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.
5. Your lab attendance is tracked by the Zoom system. Please also see lab policies no. 1 item.