CS224 - Spring 2021 - Lab #1 (Version 1: February 8, 22:43)

Creating and Running Simple MIPS Assembly Language Programs

Dates:

Section 1: Mon, 15 Feb, 8:30-12:20 in EA-Z04
Section 2: Wed, 17 Feb, 13:30-17:20 in EA-Z04
Section 3: Tue, 16 Feb, 13:30-17:20 in EA-Z04
Section 4: Mon, 15 Feb, 13:30-17:20 in EA-Z04
Section 5: Fri, 19 Feb, 8:30-12:20 in EA-Z04
Section 6: Fri, 19 Feb, 13:30-17:20 in EA-Z04

TAs:

Section 1: TBA
Section 2: TBA
Section 3: TBA
Section 4: TBA
Section 5: TBA
Section 6: TBA

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
Mehmet Semi Yenimol (x1): semi.yenimol@bilkent.edu.tr
Yusuf Dalva (x2): yusuf.dalva@bilkent.edu.tr
Ziya Erkoç (x2): ziya.erkoc@bilkent.edu.tr
Zülal Bingöl (x2): zulal.bingol@bilkent.edu.tr

Lab Attendance and Rotation Policy:

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 lab work.

Please note that labs will be done with rotation. - This gives us a scheduling flexibility.- For example, Lab1 and Lab2 starts on Monday (the following Friday becomes the last lab day), whereas Lab3 and Lab6 on Tuesday (Monday becomes the last lab day), Lab4 on Friday (Wednesday becomes the last lab day), and Lab5 on Wednesday (Tuesday becomes the last lab day). See the course syllabus for the lab days.

Purpose: This lab tries to achieve the following purposes. **1**. Learning CS224 lab rules and procedures. **2.** Introduction to MIPS assembly language programming and MARS environment (please see the self study appendix given at the end of this document). **3.** Learning simple array processing. **4.** Using multiplication and division and HI and LO registers.

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 (25 points): Testing if an array is symmetric. In a symmetric array when you fold it in the middle the overlapping array elements have the same value. For example the arrays{**1**}, { 2, **5**, 2} and {4, 7, **2**, 7, 4} are symmetric but the array {1, **5**, 4}is not symmetric. Note that in the example arrays the folding points is in bold and underlined.

Part 2 (25 points): Arithmetic expression calculation.

Lab Work:

Part 3 (25 points): Finding average, maximum, and minimum values of an array.

Part 4 (25 points): Implementing an arithmetic expression.

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 15.

b. Please note that the submission closes sharp at 9:30 am and no late submissions will be accepted. You can make resubmissions 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. Please familiarize yourself with the Moodle course interface, find the submission entry early, and avoid sending an email like "I cannot see the submission facility." (As of now it is not yet openned.)

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 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. 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 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 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. (25 points) Accessing array elements in MIPS: Write a program to check if an array is a symmetric array.

- Define an array and state its size in a separate variable and test if it is a symmetric array. Display array elements and print a message such as "The above array is symmetric". If not symmetric "The above array is not symmetric". For array declaration and for declaring its size you may use lines similar to the following. By changing these declarations you may use your program with different arrays.
- array: .word 1, 2, 1
- arrsize: .word 3

Part 2. (25 points) Calculating an arithmetic expression in MIPS: Write a program to evaluate the following expression given below (all numbers are integers): x= a * (b - c) % d

- Read a, b, c, d.
- Perform the computation and print the result. (%: modulo operation)
- Provide a simple user interface for input and output.

Table 1. Explanation for divide and multiplication instructions.

Instruction Example	How it Works
div $t1,$t2	Divides $t1 by $t2 then sets LO to quotient and HI to remainder (use mfhi to access HI, mflo to access LO). Note that LO and HI are two registers used for multiplication and division.
mult $t1,$t2	Multiplies $t1 and $t2 and sets HI to high-order 32 bits, LO to low-order 32 bits of the product of $t1 and $t2 (use mfhi to access hi, mflo to access lo)
mfhi $t1	Moves from HI register : Set $t1 to contents of HI
mflo $t1	Moves from LO register : Set $t1 to contents of LO

Part 3 & 4. Lab Work (50 points)

Part 3. Accessing array elements in MIPS: Write a program to find min, maximum, and average value of the elements of an array.

(25 points)

- Define an array and state its size in a separate variable as you have done in Part 1.
- Find minimum, maximum, and average value of the array elements. Use integer arithmetic.
- Display array elements and results as follows

  Memory Address Array Element
  Position (hex) Value (int)
  ========== ==========
  ....
  ....
  Average: ...
  Max: ...
  Min: ...

Hint. syscall with the code value 34 is used for printing integer numbers as a hexadecimal number. Remember that memory address are integer numbers. See the code segment below.

```
# Print the contents of $t0 as an hexadecimal number.
        li      $v0, 34
        add     $a0, $zero, $t0
        syscall
```

Part 4. Using MIPS for Mathematical Calculations (25 Points)

Write a program that prompts the user for one or more integer input values, reads these values from the keyboard, and computes a mathematical formula such as (***a different formula may be given by your TA***):

$$A= (B * C + D / B - C) \% B$$

When the computation is finished, the program should print the result along with an explanatory comment to the user via the display. Note that all is done using integer instructions (no floating point).

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

1. Attendance is mandatory. **The preliminary work is graded only if you submit your lab work with the observation 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. Students will earn their own individual lab grade. The questions asked by the TA will have an effect on your individual 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.

APPENDIX
SELF STUDY: ASSEMBLING AND DEBUGGING A PROGRAM

A. Examining the Controls and Options in Mars

1. Open the MARS simulator. Take a few minutes to explore each of the windows (Edit & Execute; MARS Messages & Run I/O; Registers, Coproc 1 and Coproc 2). Look at each of the controls on the pull-down menus from the task bar at the top (File, Edit, Run, Settings, Tools, Help) and discuss what you think it does. Change the run speed to 1 instuction/sec using the slide bar at the top.

2. Now, starting from the left, slowly put the mouse over each of the icons in the row across the top, to see the action that it will cause (but don't click the icon). Determine which actions, represented by the icons, also can also be selected from a pull-down menu. Click the "?" icon (or choose it from the Help pull-down menu) and in the Help window that open, click each of the tabs and briefly look at the Help contents for that topic. Note that the MIPS tab offers a box with scroll bar, and 6 tabs of its own. Similarly, the MARS tab opens a window with 8 tabs. Be sure to look at each of these.

3. On the top menu, Tools offers a list of tools in the pull-down menu. Open each of these and look at it briefly to begin to understand the range of additional capabilities of MARS

B. Using a Simple Program in Mars

4. Load in Program1 (Generates "hello world"), using File > Open, or the appropriate icon button. In the Edit window, examine this program, and learn what it does. Try to understand how it does it.

5. Assemble the program, using Run > Assemble , or the appropriate icon button. In the Execute window, examine the code portion of memory (called Text Segment) and the data portion of memory (called the Data Segment). Note the beginning address of each, and the contents of each. Knowing that 2 hex characters represent one byte, and remembering that ASCII characters are each coded as one byte, determine which characters are stored in which locations in the Data Segment. Examine the initial values of the registers—which ones are non-zero? Make a note of their values. Read the messages in the MARS Messages window. Check the Run I/O window.

6. Set the Run Speed bar to 1 instruction/second. Now run the assembled program, using Run > Go, or the appropriate icon button. What happens to the yellow highlight bar in the Text Segment during execution? What is written in the MARS Messages window? In the Run I/O window? Compare the final values of the non-zero registers—did you expect to find $1, $4, and $2 changed by the program? What is the final value of the $pc register?

7. Now edit the Program1 so that it produces a different output: Hello <a different name> , and make it print out that name.

C. Finding and Fixing Errors in Mars

8. Load in Program2 (Conversion program), assemble it, and run it, providing the necessary input from the keyboard. What does this program do? Examine the MIPS assembly program to understand how it does it.

9. Edit the program by changing li $v0, 5 to li v0, 5 . Then assemble it and read the error message. Then fix the error and re-assemble it.

10. Edit the program by changing li $v0, 5 to $li v0 . Then assemble it and read the error message. Then fix the error and re-assemble it. Now run the program, at 1 instr/sec, and make note of the value of $v0 after the 2nd syscall is completed.

11. Edit the program by changing mul $t0, $v0, 9 to mul $t0, $t0, 9 . Then assemble it and run it, and explain why the output value is what it is. Then fix the error and re-assemble it and re-run it, to verify that Program2 again works correctly.

12. Change the Run speed back up to the maximum. Assemble the program, and instead of hitting the normal Run button, instead hit the “Run 1 step at a time” icon (looks like >1) repeatedly, to single-step through the program. Notice that you can go slowly, examining the memory and registers after each step, or go quickly. This single-step mode is good for debugging.