CS224 - Fall 2020 - Lab #3 ((Version 1: October 20, 18:16 pm)

MIPS Assembly Language Programming
Bit Manipulation, Recursion, Linked Lists

Dates:

Section 1: Wednesday, 21 October, 13:30-16:20 in EA-Z04, Only odd no. students may come to the lab
Section 2: Friday, 23 October, 13:30-16:20 in EA-Z04, Only odd no. students may come to the lab
Section 3: Sunday, 25 October, 8:30-11:20 in EA-Z04, Only odd no. students may come to the lab
Section 4: Thusday, 22 October, 13:30-16:20 in EA-Z04, Only even no. students may come to the lab

Submission Date: October 31, Saturday, 23:59, for all lab days.
One piece of submission (no preliminary work and lab work distinction).

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

1. Lab Attendance Policy:

As you know all labs can be done online. On your lab day there will be a TA in the physical lab to answer your questions. Two TAs, one of them joining from the physical lab, will be available to answer your questions in zoom meeting. They will use zoom breakout room service and will be able answer two different students' questions at a given instant. For attendance tracking purposes make sure that you have some communication (zoom chat etc.) with your TAs on your lab day. If you need more help after or before your lab day please communicate with your TA by email.

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.

2. Important Implementation Requirement for Lab3: In this lab you are not allowed to use $t registers in the subprograms.

3. Purpose: More experience with MIPS assembly language programming with bit manipulation, recursion, and implementation of dynamic linked list structure.

4. Summary

There are five sub programs. **Program** 1: Bit processing, **Program** 2: Recursive program, **Program** 3: Display a linked list in reverse order with recursion, **Program** 4: Generate a copy of a linked list iteratively, **Program 5**: Generate a copy of a linked list recursively.

Each program is 20 points.

5. Testing

Test **Program** 1 and 2 using the same main with a simple console interface. For the linked list programs (**Program** 3 to **Program** 5) modify the given linked list program and provide a user interface. In the menu give the user the option of creating the linked list with their own input data values. For testing of the linked list program the user should enter the following number 5, 15, 20, 25, 30 (to make sure that you implement the linked list creation by getting the numbers from the user). 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."

Assume that all Inputs are Correct: In all lab works if it is not explicitly stated you may assume that program inputs are correct.

DUE DATE/TIME SAME FOR ALL SECTIONS

No late submission will be accepted.

Please upload your programs to Moodle by 23:59 on Saturday October 31. You may be given further instruction for uploading please check Moodle course interface.

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).

6. Lab Work

Note that in this lab you have one piece of work there is no division like preliminary work and lab work. For all sections the submission date is the same. You have to provide a neat presentation prepared in txt form. Provide following five lines at the top of your submission for your 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.

Program 1.

checkPattern: Write a **non recursive** subprogram to count the bit pattern stored in the rightmost n bits (window of size n bits) of a given input. Following are the inputs for the problem:

$a0 - pattern to search, for example 101 (stored as 0000000.....00101 in $a0)
$a1 - input to search, for example 10000111101101000101000101101000
$a2 - n

Note that, there is no need to check the validity of n (i.e., assume that $a2 is a valid input between 1 and 32). The search in $a1 must be performed from right to left. During pattern matching, bit pattern windows cannot overlap. For example, for the above sample input your program will divide the bit representation in $a1 as follows:

10000111101101000101000101101000 -> 10 000 111 101 101 000 101 000 101 101 000

The number of windows matching the given pattern ($a0, 101 for this example) is 5. Starting from right (least significant bits),

window 1 - 000 - not matching
window 2 - 101 - matching
...
window 11 - 10 - cannot match by definition

For n=4 the maximum match possible is 8, since you would first check the rightmost four bits then four bits that come before rightmost four bits, etc. For n=2, maximum match can at most be 16, for n=3 it can be at most 10.

Program 2.
recursiveSummation (20 points): Write a recursive MIPS subprograms that finds the summation of numbers from 1 to N. N is a positive number.

Program 3.
display_Reverse_Order_Recursively (20 points): Display the elements of the linked list in reverse order.

Program 4.
duplicateListIterative (20 points): Study the linked list program provided. Write a **non recursive** subprogram to duplicate a linked list. When called $a0 points to the original list. It returns the new list head in $v0.

Program 5.
duplicateListRecursive (20 points): Study the linked list program provided. Write a **recursive** subprogram to duplicate a linked list. When called $a0 points to the original list. It returns the new list head in $v0.

7. Submit your code to Moodle for MOSS similarity testing

Submit your MIPS codes for similarity testing to the Moodle > Assignment specific for your section. You will upload one file: **name_surname_SecNo_MIPS.txt** created in the relevant parts.