CS 240 – Computer Organization

Lab 4 – Recursion, Nested Subroutines and Stacks– 105 points

This lab should be very simple and straight forward. The lab comes in three parts!

In this lab, you will learn how to use MIPS assembly instructions to:

1. Implement a simple recursion program.
2. Write a subroutine to make a call to another subroutine.
3. Use stacks to preserve information and retrieve them.

Download the lab file – lab4.s provided and fill in the subroutines. Please look at comments in the provided source code for more information.

Important note: ***Do not use the registers $s0 to $s7 in any of your codes.***

Your tasks:

1. Complete the function **gcd** to compute the greatest common divisor of two numbers. (30 points)

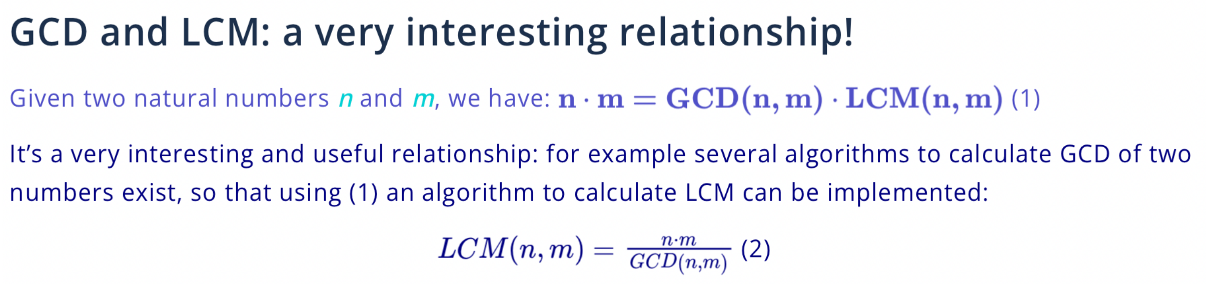
- You are given two 32-bit numbers A and B located in registers $a0 and $a1.
- Return the answer in $v0.
- You would use the Euclid's Algorithm to find the Greatest Common Divisor, a prototype of the GCD algorithm in C is given as follows:

```
int euclidGCD(int X, int Y) {
    if (Y == 0)
        return X;
    else
        return euclidGCD(Y, X % Y);
}
```

Note: X % Y would be the mod operation that would produce the remainder of the division.

2. Complete the function **lcm** to determine the least common multiple of two numbers. (30 points)

- You are given two 32-bit numbers A and B located in registers \$a0 and \$a1.
- Return the answer in \$v0.
- *You will use the relationship between GCD and LCM to tackle this problem*



GCD and LCM: a very interesting relationship!

Given two natural numbers n and m, we have: $\mathbf{n \cdot m = GCD(n, m) \cdot LCM(n, m)}$ (1)

It's a very interesting and useful relationship: for example several algorithms to calculate GCD of two numbers exist, so that using (1) an algorithm to calculate LCM can be implemented:

$$LCM(n, m) = \frac{n \cdot m}{GCD(n, m)} \quad (2)$$

- Make call to the GCD subroutine (part 1) to obtain the GCD information.
- Recall the rules of subroutines in MIPS
  - It's the responsibility of the caller to preserve all the \$t registers, \$v registers and the \$ra (return address).
  - It's the responsibility of the callee to preserve the \$s registers and \$a registers.

For this problem, you will use stack to preserve the values before invoking the nested subroutine call. In particular, your code has to store the return address in stack.

3. Complete the function **random_sum** to find the sum of smallest,largest, gcd and lcm. (35 points)

- Three 32-bit number is given to you in \$a0, \$a1 and \$a2
- Determine the largest of three numbers
- Then determine the smallest of there numbers.
- Then find the GCD and LCM of the two numbers using the subroutines you wrote for part 1 and 2.
- Find the sum of smallest number, largest number, GCD and LCM.
- **Use stacks to store register contents before making subroutine call.**
- Store the result in \$v0 register.

Important Things to Consider

1) Don't forget to update the student name and id variables. These are used for grading purposes. FAILURE TO ADD NAME AND ID RESULTS IN NO SUBMISSION

```
student_name: .asciiz "Your Name"
student_id: .asciiz "Your Student ID"
```

2) Since we use a computer program to test your code, you are **ONLY** allowed to modify the highlighted area in the source code.
For example:

```
count_ones:
move $t0, $a0
############## Part 1: your code begins here #################
Your stuff goes here :D
############## Part 1: your code ends here  ##################
move $v0, $t0
jr $ra
```

3) Do not use the $s registers in your code (subroutine) as they are used by the main program.

4) Make sure you preserve the return address in $ra register and other used $t , register before making a call to another subroutine

Submissions:

- Complete the provided lab4.s template and submit it to Gradescope.
- **Submit only lab4.s by itself (not in a folder/zip and with no other files)**

Gradescope will immediately show you your auto grading score, and you can resubmit as much as you need (grades will be synced to Canvas later). Note that we will use additional test cases other than the ones provided so make sure to test your code yourself. 5 points of each problem will be awarded for making a serious attempt at each task AND for adding comments.

CHEATING:

Submit your own work. While you may collaborate with other students, you are required to submit your own code. Any students found submitting identical/very similar code will be evaluated for academic dishonesty. DO NOT share/send your code to other students, this is also considered cheating. We use a similarity checker to evaluate all submissions.

New Anti-Cheat Measure:

The instructor and the TA reserve the right to call upon any student to explain their code after submission deadline. Failure to comply AND/OR if the student is unable to defend their work/code, the work will be considered as plagiarized; The student will earn no credit for the lab and may be reported for academic dishonesty.