CMPT350: Lab Demo 4

Functions with parameters and an introduction to the stack
1%
Due: Wednesday, October 12 @ 16:50

Demo 3 introduced the concept of procedures, but without arguments and return values , they aren't very practical. In this lab we will learn how to pass function arguments and return values from a function to compute and verify the determinants of 2x2 matricies

Laboratory Procedure:

Matrix Determinants

Let M be a 2x2 matrix, defined as

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Then, the *determinant* of M, $|M|$, is given by

$$|M| = ad - bc$$

Procedure

From the Lab Demo 4 assignment, download the demo4.s skeleton file to begin working from. Create a MIPS program that takes as input 5 integers, and processes them with two functions, `det`, and `det_verifier`. These functions have the following specifications:

- `det`
  - Takes 4 arguments: a,b,c,d, the entries of the matrix.
  - Returns one value, the determinant of the matrix.
- `det_verifier`
  - Takes 5 arguments: a,b,c,d, the entries of the matrix, and the guessed value of the determinant.
  - Computes the value of the determinant and returns one value:
    - 0, if the guessed value is incorrect.
    - 1, if the guessed value is correct.

The program should compute the value of the matrix using `det`, print it to the screen, and check if the user guessed the correct determinant using `det_verifier`. Both functions should compute the determinant from a,b,c, and d.

Assignment Requirements:

Ensure your program meets all the following requirements:

- Two functions are implemented; one just to compute the determinant (`det`), and one to compute the determinant and compare it with a provided value (`det_verifier`).
- The functions return program flow to where they were called from (I.e., you don't jump at the end of the function to a fixed label)
- The program respects assembler conventions for using registers with functions (see below).
- The procedure terminates cleanly
- The program is well documented

*** **NOTE:** Your code **must run** within the **laboratory environment**. Failure to do so will result in a **mark of ZERO** for the program. ***

Submission

When you feel your program meets all of the above requirements, call the lab instructor over and they will review your program with you. Please wait until you are confident in your program's results. (Feel free to ask for clarification on anything at any time though!). Your program should be named using your name and end in a .s extension, for example, BakerDemo2.s

After demonstrating your program to the lab coordinator, create a zip archive with your code submission and trace file included. An easy way to do this is with the `zip` command. For example, if your solution file and trace file are in a folder called `baker`, then the command `zip -r baker.zip baker/` will create a zip file containing your submission files called baker.zip (Replace this with your name when creating your zip file). Upload this zip file to Moodle.

Assembler Caller vs. Callee Conventions

When a function is called, there are two parties involved. The *caller* is the piece of code from which the function is called, and the *callee* is the code within the function. To make function writing as easy as possible, conventions have been established to determine what should be expected to happen when a function call occurs.

Caller-Saved Registers (May be modified by the function)	$t registers, $a registers, $v registers
Callee-Saved registers (Must be untouched or restored by the function)	$s registers, $ra
Argument registers	Only $a registers are used for arguments. Extra data needed must be stored on the stack.

Hints

- This assignment sounds much harder than it really is! The only math operators we need are `mul` and `sub.` The `det` function only needs to be 4 instructions long, and `det_verifier` isn't much longer
- We only have 4 argument registers, `$a0-$a3,` so to provide more arguments we need to store memory on the stack – make use of the stack pointer (`$sp`)