CMPT350: Lab Assignment 2

Recursive functions and using the stack with MIPS 5%

Due: Thursday, October 20, 17:00

This lab assignment will have you implement recursive functions in MIPS.

Laboratory Procedure:

On the course Moodle page, look for the Laboratory 2 assignment in the Laboratories section. Inside are two premade skeleton MIPS files, `laboratory2_fact.s` and `laboratory2_collatz.s` Save these somewhere handy on your machine. You may use both the command line version, `spim,` and the GUI version, QtSpim to run your program. As well, you will need a text editor to edit your programs.

Laboratory2_fact.s

In `laboratory2_fact.s,` write a MIPS program which will **recursively** compute the factorial of a non-negative value provided by the user. The factorial of n is defined as

$$n! = \prod_{i=1}^{n} i$$

with the special case that 0! = 1

Laboratory2_collatz.s

In `laboratory2_collatz.s,` write a MIPS program which will **recursively** compute the Collatz number and print a Collatz sequence of a positive value provided by the user. The *Collatz sequence* is a sequence recursively defined as

$$C_{n+1} = \left\{ \begin{array}{ll} \frac{C_n}{2}, & \text{if } C_n \text{ is even} \\ 3C_n + 1, & \text{if } C_n \text{ is odd} \end{array} \right\}$$

for any positive C_1. We will consider the Collatz sequence terminated when C_n= 1.

The Collatz conjecture states that for any positive value, its Collatz sequence will stabilize at some finite value n with C_n= 1. This index n is C_1's *Collatz number*. (N.B. This has been shown empirically to be true for all values up to ~2^{68} . However, it has not been proven generally, and in fact is considered by prominent mathematicians to even be "completely out of reach of present day mathematics"[1])

For example, let C_1= 10. Then,

$$
\begin{aligned}
C_1 &= 10 \text{ (even)} \\
C_2 = \frac{C_1}{2} &= 5 \text{ (odd)} \\
C_3 = 3C_2 + 1 &= 16 \\
C_4 = \frac{C_3}{2} &= 8 \\
C_5 = \frac{C_4}{2} &= 4 \\
C_6 = \frac{C_5}{2} &= 2 \\
C_7 = \frac{C_6}{2} &= 1
\end{aligned}
$$

Therefore, the Collatz number of 10 is 7, and the Collatz sequence for 10 is 10, 5, 16, 8, 4, 2, 1.

Assignment Requirements:

- **Basics**
  - Run your program successfully using either QtSpim or the `spim` command line tool.
  - Sufficient commenting is present in the program, including the preamble comment block at the beginning of the file. **This preamble block must contain a line which contains the contents "# Author: <YourName>"**
  - The program terminates properly.
  - Output looks "proper" (e.g. appropriate newlines, easy to read)
  - Appropriate data management is used (respecting Caller-Callee conventions, using memory vs. registers appropriately, etc.)
- **Laboratory2_fact.s**
  - The program prompts the user for a non-zero input value.
  - The program computes the factorial of the provided input value
  - The program computes and returns the factorial using a function

[1] Lagarias, Jeffrey C., ed. (2010). *The Ultimate Challenge: The 3x + 1 Problem.* American Mathematical Society. ISBN 978-0-8218-4940-8. Zbl 1253.11003

    - The program computes the factorial **recursively**, using repeated function calls.
    - The function displays the result after being returned by the function call
- **Laboratory2_collatz.s**
    - The program prompts the user to input a positive number to begin the Collatz sequence,
    - The program uses a function to **recursively** compute the Collatz sequence and number for the given input.
    - The program computes and displays the Collatz sequence for the given input (you do not need to save this sequence, just display the sequence values as you compute them)
    - The function returns the Collatz number for the provided input.

*** **NOTE:** Your code **must be interpreted successfully and execute** within the **laboratory environment, using the command line version of SPIM**. Failure to do so will result in a **mark of ZERO** for the program. ***

Creating a trace file

As part of all demo and laboratory submissions, you must submit what's called a trace file. This is a text file you'll create that in essence replicates what your terminal looks like when you demonstrate your programs. To assist with this, a tool called `script2` has been provided for you. To create a trace file, simply run `script2` in your terminal, and then proceed to use the terminal as normal to navigate to and run your programs. When you have demonstrated everything, press CTRL+D to signal that you are done, and `script2` will exit, creating a file called typescript in the directory you ran `script2` from. This is your trace file. **Please rename your tracefile to include your name, e.g. baker_typescript**

Submission:

Your programs should be named using your name and end in a .s extension. For example, BakerLaboratory2.s

When you feel your programs meet all of the above requirements, create a zip archive with your code submission and trace file(s) 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.

Example Output:

Example screenshots are posted under the Laboratory 2 section on Moodle.

Hints:

- Don't forget where you've been – you MUST use the stack when writing a recursive function.
- Don't overcomplicate the arithmetic – arithmetic pseudoinstructions are perfectly acceptable. The pseudoinstructions for multiply and divide are:
    - `mul $t0, $t1, $t2`
    - `div $t0, $t1, $t2`
- Use `andi` to find the parity of a value:
    - `andi $t1, $t0, 1`
- Consider using subroutines/extra labels to divide up your recursive functions. E.g the Collatz function may use the `collatz_basecase`, `collatz_evencase`, and `collatz_oddcase` labels to branch and jump (but not jump and link!) to. Labelling your labels with the name of the function they should belong to helps organize your program.