Assignment SPIM 3

Due Date: April 26

Purpose
In this assignment, you will try out SPIM subroutines, in the form of recursion. To do this properly, you must store local variables on the stack. This also forces you to take care of the $sp and $fp pointers as well as the return address register, $ra. Follow the in-class and text examples carefully!

You may work on this problem with one partner. Turn in only one solution that includes both names.

Problem

The Tower of Hanoi is a famous computing problem. In this puzzle, there are \( n \) disks of different sizes and three pegs. Initially, all of the disks are on one peg, \( A \) or the source, in order of size, with the largest on the bottom (see diagram). The goal is to move all of the disks from peg \( A \) to peg \( B \) (the destination). The catch is that only one disk may be moved at a time and a disk can never be placed on top of a smaller disk. In order to do this, a third peg, \( C \) or spare is required. So for the initial condition shown in the diagram, a sequence of moves would be:

Move \( A \) to \( B \)
Move \( A \) to \( C \)
Move \( B \) to \( C \)
Move \( A \) to \( B \)
Move \( C \) to \( A \)
Move \( C \) to \( B \)
Move \( A \) to \( B \)

Note that for only three disks, seven \((2^3 - 1)\) moves were required! Legend has it that when 64 disks were moved by priests in a temple in India, the world would end.\(^1\)

Your job is to write a recursive solution to the Tower of Hanoi puzzle in SPIM.

Input
The program should prompt for \( n \), an integer, representing the number of disks on peg \( A \). Assume that \( n \geq 1 \).

Output
The output should be a list of moves, similar to those shown in the above example. After all the moves are displayed, the total number of moves should be shown.

\(^1\)I don’t think we need to worry about this happening anytime soon.
Specifics

- You must do this recursively. In order for this to work properly, you must store local values in the stack before each new recursive call. Follow the sample program done in class. Note that it doesn’t hurt to allocate more storage than you actually need.

- The total number of moves also must be calculated recursively.

- Do not use global variables. This means, do not simply use global registers. You can easily store the number of moves in a globally-used register, but you will not receive any points for this.

- For your own sanity, use good commenting style.

Notes

- Find a recursive algorithm in a text or online; no need to reinvent the solution.

- Your best bet is to write a short Python/C++ version first, then translate this to SPIM. Note that this is not a very long program, but it is somewhat tricky.

- I also suggest working on the total number of moves and the moves themselves separately; that is, make the program work for the moves first, and then figure out how to get the total number of moves displayed.

- Many solutions can be found on the interwebs. My advice: Don’t copy them:
  1. It’s not always clear that a given solution is correct.
  2. You will not learn how to program procedures properly, and therefore you may not be able to figure out problems on the final exam.
  3. Interweb solutions may not include finding the total number of moves recursively. This portion will count for a significant number of points.
  4. You’re a Computer Science major! This should be (somewhat?) fun!

- As usual, turn in your source code to me via email. Name the file as before: firstInitial,lastNameSPIM3.a, as in mgousieSPIM3.a.

- Turn in a printed copy in class on April 27.

Worst comes to the worst.
– Cervantes (1547-1616)