

CSCI-21 Lab #9, due 11/29/22

Use the "frame-based linkage convention" (argument in \$a0 register, local variables on the stack) for this. If the subroutine needs to save the parameter, put it into a local variable on the stack and reload it after the nested call.

Part 1:

Write a program that computes the value of triangle numbers using a recursive subroutine. The definition of a triangle number is:

$$\text{Triangle}(n \leq 1) = 1$$
$$\text{Triangle}(n > 1) = n + \text{Triangle}(n-1)$$

For example:

$$\text{Triangle}(1) = 1$$
$$\text{Triangle}(2) = 3$$
$$\text{Triangle}(3) = 6$$
$$\text{Triangle}(4) = 10$$

For this, each subroutine invocation has its own argument n which is passed in \$a0, and will be stored in its stack frame in a local variable. Then it gets the return value from the next invocation (in \$v0) and adds it to its n (as was stored in the stack frame) before putting the result into \$v0 to return to the caller.

Your program will prompt the user for n and calculate and print the value returned from $\text{Triangle}(n)$. Test with several values, including a large value like 100.

Part 2:

The square of an integer n is equal to $\text{Triangle}(n) + \text{Triangle}(n-1)$. Write a recursive subroutine using the Triangle subroutine from part 1 to calculate $\text{Square}(n)$. Write a program to prompt for n and print the result of $\text{Square}(n)$. The definition of a square number is:

$$\text{Square}(n \leq 1) = 1$$
$$\text{Square}(n > 1) = \text{Triangle}(n) + \text{Triangle}(n-1)$$

Call Square using a stack frame, and have Square call Triangle twice, passing its argument in the \$a0 register. For the recursion, create automatic local variables on the stack for the parameter and both recursive results in the variables, then retrieve the values to add together to return to the caller.

For both programs, use addu to ignore the errors caused by potential overflow. You may assume that the user will enter a reasonable value (greater than or equal to 1). Test with several values, including a relatively large value like 100.