Hunter Negron

How is recursion different in C and MIPS?

The implementation of recursion in MIPS and C is pretty different. In C, one only needs to write a base case (usually testing arguments with an if-statement). But in MIPS, the base case must also be tested, but you must use branches in place of a simple if-statement to run different code based on what the arguments are. In C, a recursive call is as simple as any other function call. In MIPS however, all of the important and registers must be saved onto the stack and then popped off once the recursive call returns.

How does the stack change when we execute recursive calls?

When a *normal* function call is made, the important registers are pushed onto the stack, the function is called with `jal`, which eventually returns, and the values are then popped off the stack. However, when a *recursive* call is made, the registers are pushed to the stack, but there is a good chance another recursive call is made before the original call returns, so the registers are once again pushed onto the stack. This can happen an arbitrary amount of times, so the stack grows pretty significantly when a series of recursive calls is made. This is often why a stack overflow occurs, because the base case may be misconfigured or too many recursive calls are made.