CS270 Recitation 8
"LC-3 Recursion Exercise"

Goals

- To understand purpose of the runtime stack.
- To be able to draw the contents of the runtime stack.
- To learn how to implement recursion in LC-3.

Background

The runtime stack is an area of memory used to keep track of a program's progress. When a function is called, a new activation record is added to the stack. Each activation record contains space for the function parameters, the return value, the return address, the frame pointer, and finally the local variables.

The return value is the value returned to the caller, for example using "return 1" in C. The return address (R7) is the address to the next instruction in the calling function. The stack pointer (R6) points to the newest value on the stack. Since the stack pointer changes often, the frame pointer (R5) is used to load and store parameters and locals. The frame pointer contains the address of the first allocated local variable.

An activation record does not automatically appear, it must be built using assembly instructions to push values onto the stack. Pushing a value onto the stack takes two instructions:

```
ADD R6,R6,#-1  ; decrement the stack pointer. The stack grows "up".
STR R0,R6,#0   ; store value (R0 here) at the location R6 points to.
```

When a function is called, the *caller*:

- Pushes the function parameters onto the stack.
- Transfers control to the *callee* (the called function) using JSR or JSRR.

Upon entry into a function, the *callee*:

- Allocates space for the return value by decrementing the stack pointer (in R6) by the number of locals.
- Pushes the return address (in R7) onto the stack.
- Pushes the caller's frame pointer (in R5) onto the stack. R5 becomes R6-1.
- Allocates space for any local variables by decrementing the stack pointer.

When a function call completes, its activation record must be removed from the stack. This is achieved by popping values off of the stack into the appropriate registers.

To pop a value off of the stack in LC-3, two instructions must be used.

```
LDR R0,R6,#0   ; load value from the location R6 points to.
ADD R6,R6,#1   ; increment the stack pointer, the stack shrinks down.
```

When a function returns, the *callee*:

- Writes the return value to the allocated location, usually the frame pointer + 3.
- De-allocates local variables by adding to the stack pointer.
- Restores the caller's frame pointer by popping it off of the stack into R5.
- Restores the return address by popping it off of the stack into R7.
- Returns control to the caller by executing RET.

When a function is returns, the *caller*:

- Pops the return value into a register for use later.
- Pops the arguments passed to callee off of the stack.

Assignment

Create a new directory called R8, all files should reside in this directory. Use wget to download r8.asm and r8.c from the course website, as shown below.

```
wget http://www.cs.colostate.edu/~cs270/.Fall12/recitations/R8/r8.asm
wget http://www.cs.colostate.edu/~cs270/.Fall12/recitations/R8/r8.c
```

For this assignment, you will be filling the provided table with values from the runtime stack of a program that computes a factorial. Use the LC-3 simulator to fill in at least three activation records in this table. Label which locations are used for the parameters, return value, return address, frame pointer, and locals. The C equivalent has been provided to help understand the program flow.

When you are finished, show the table to your TA.