CS270 Programming Assignment 8
"Recursive Magic"

Goals

For this assignment you will write programs in C and LC-3 assembly code. Both programs will perform the identical recursive algorithm. The goals of this programming assignment are:

1. To solidify your understanding of the stack memory model.
2. To understand how recursion is implemented in C and at the assembly level.
3. To extend your assembly coding skills.

The Assignment

STEP ONE: C Program

Write a C program called *pa8.c* that prints a 16-bit hex number input by the user. The number must be positive, meaning it must be in the range 0x0000 to 0x7fff. Use the recursive algorithm as specified later in this document. The C program will consist of a main entry point and the two functions described below. Do not use any global variables. To get started, implement a main program that inputs a hex number (%x) from the user and calls the following function to print it:

void printHexValue(unsigned int value);

The main program should print an error when the value is out of the range shown above. You cannot use any system calls such as printf or sprintf. Instead, create a recursive function. If the value is 16 or greater, make a recursive call to *printHexValue()* with a parameter equal to the input value divided by 16. Then print the input value modulo 16, i.e. the last digit of the number. This is also the base case when the input value is less than 16. Test your program with at least the input values 0x1234, 0x6789, 0x789a, and 0x3bcf. Make sure that the output has only lowercase letters.

void printHexDigit(unsigned int iDigit);

This function prints a single ASCII character representing the input value. Print "0" to "9" for the values 0 to 9, and lowercase "a" to "f" for the values 10 to 15.

STEP TWO: LC-3 Program

Copy the program from the following link:

http://www.cs.colostate.edu/~cs270/.Spring14/assignments/PA8/pa8.asm

Convert the example program from an iterative method to a recursive method. The PRINT function is equivalent to the *printHexValue()* function , and must be modified to call itself in exactly the same manner as the C program from the previous step. The program that you have been given implements a function called DIVIDE that divides the numerator by a power of two denominator. You must implement the OUTPUT function to output a single hex digit, corresponding to *printHexDigit()* in your C code. You may use these functions in your code. Please leave the PARAM value in address x3001 so that we can test your program. No error handling is required in the assembly program, so the input data value must be between 0x0000 and 0x7fff.

Your OUTPUT and modified PRINT functions must follow exactly the stack protocol shown below, which is the one used by the LC-3 C compiler:

1. The caller pushes parameters onto the stack.
2. The callee makes room for the return value, if one exists.
3. The callee pushes the return address of the caller from R7.
4. The callee pushes the frame pointer of the caller from R5.
5. The callee sets up its own frame pointer.
6. The callee executes its function code.
7. The callee stores the return value, if one exists, onto the stack.
8. The callee pops the frame pointer of the caller to R5.
9. The callee pops the return address of the caller to R7.
10. The callee returns using the RET command.
11. The caller pops the return value, if one exists, off the stack.
12. The caller cleans up the stack to remove the parameters it pushed.

This stack protocol is shown in the code provided in pa8.asm. We recommend that you completely understand the provided code before you make changes. This assignment does not require very many lines of code, but you will have to make the modifications carefully. During lab hours we will be glad to explain the provided code, but we will not show you how to modify the code for the assignment.

Submission Instructions

Submit pa8.asm to the Checkin tab for preliminary testing.

Grading Criteria

To grade the assignment, we will examine the C program and make sure that you have followed directions and implemented the print function recursively (25 points). We will examine the assembly program to make sure that the print function is implemented recursively (25 points). We will test the assembly program with four legal integer values (40 points). The remainder of the grade will be given for style considerations: commenting, indentation, naming, and following directions (10 points). The preliminary testing will also test several legal integer values, but will not verify that your program is recursive or that it implements the stack protocol correctly. Make sure that R6 is correctly restored after your program executes.