C Programs in Assembly and The Runtime Stack

Programming Lab #5
CEG 3310/5310: Computer Organization

PURPOSE

In this lab you will learn how C programs look in assembly. This will show you how higher level languages are actually running on your machine at a low level.

ASSIGNMENT

You will act as a "human compiler" converting the C code provided in "lab-5-c-code.c" into assembly code. You must implement the functions using subroutines and a properly organized runtime stack. The runtime stack format that you must use is defined in "Runtime Stack Lab 5.pdf".

When your program reaches the HALT instruction in main, you should have the correct output stored in the stack.

For example:

If the array "ar" contains the elements: {2, 3, 5, 0, 1} then the sumOfSquares subroutine should return:

$2^2+3^2+5^2+0^2+1^2=4+9+25+0+1=39$

You must use a subroutine called "square" to calculate the square of each element in the array, so given an input of "2" the square subroutine should return: 4

The output of "square" is then returned to "sumOfSquares" where a running total is kept, adding each square after every loop. sum = 4 on first loop, sum = 13 on second loop, sum = 38 on third loop and so on.

GETTING STARTED

Load the sample code "lab-5.asm" into your LC3 simulator and run it, notice how the runtime stack is getting setup for main and how the jump to sumOfSquares is called.

Load the C code "lab-5-c-code.c" into your C compiler and run it. You must understand what this C code is doing in order to complete your lab assignment.

Open the "Runtime Stack Lab 5.pdf" file and as you write your subroutines, format your stack to meet this document exactly.

IMPLEMENTATION

Write your assembly code in the "lab-5.asm" file provided:

- Implement the main function (it is already implemented for you) in assembly.
- Implement the sumOfSquares function as an LC3 assembly subroutine.
- Implement the square function as an LC3 assembly subroutine.
- For main and each subroutine you must maintain the runtime stack as defined in "Runtime Stack Lab 5.pdf".
- When your assembly code halts, the correct sum of squares calculation must be stored in the main return value location in memory.

GRADING

- main() function in assembly generates the correct runtime stack up to the first subroutine call
    - 5 points
- sumOfSquares() function in assembly generates the correct runtime stack up to the first JUMP instruction in the sumOfSquares subroutine
    - 5 points
- square() function in assembly generates the correct runtime stack up to the first JUMP instruction in the square subroutine
    - 5 points
- sumOfSquares in assembly performs functionally identical to the sumOfSquares() function in the provided C code, taking the proper inputs, performing the task, and returning the proper output
    - 10 points
- square in assembly performs functionally identical to the sumOfSquares() function in the provided C code, taking the proper inputs, performing the task, and returning the proper output
    - 10 points
- The correct calculation of the sum of each squared array value is stored in main() return value location of the runtime stack even when a different array is used:
  (for example {1, 5, 10, 2, 19, 10} = 591)
    - 10 points
- Total
    - 45 points