# CEG 3310/5310 – Computer Organization

# Lab 4 – C Programs in Assembly and The Runtime Stack

## Learning Objectives

- Learn how to implement a runtime stack in assembly language
- Learn how a C program gets translated to assembly and machine language

## Overview

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.

## The Program

You will act as a “human compiler” converting the C code provided in “lab4.c” into assembly code. You must implement the functions using subroutines and a properly organized runtime stack.

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

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 “5” the square subroutine should return: 25

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.

- Implement the main, sumOfSquares, and square function in assembly.
- Implement a proper runtime stack.
- Translate the C code directly to assembly as best as possible.
- Do not attempt to rewrite stdio.h or print statements in assembly.

## Sample Code

Load the C code “lab4.c” into your C compiler and run it. You must understand what this C code is doing in order to complete your lab assignment.

## Grading

This lab is worth 5.00 points, distributed as follows:

| Task | Points |
|---|---|
| Successfully implement main function in assembly | 1.25 |
| Successfully implement sumOfSquares function in assembly | 1.25 |
| Successfully implement square function in assembly | 1.25 |
| Successfully implement a proper runtime stack | 1.25 |
| Total | 5.00 |