Workshop 5

Introduction

We have been using a simple way to print out the results to see if the calculations are correct or not. It turns out that there is a better way of doing the test of various functions of a project. This is the unit test. In this lab, we will use unity (http://www.throwtheswitch.org/unity [http://www.throwtheswitch.org/unity]) as a framework for unit test. We will demonstrate the usage of unity and then we will use unity test functions to facilitate TDD (test driven development) of a few functions in assembly.

We will also demonstrate how to evaluate the performance of the code. This will be the main topic of Lab 10 for this week.

TDD is just a way of software development. We need to have the requirements to develop the software. In this workshop, we look at different ways of swapping numbers and use these functions to sort an array in ascending and descending order.

We have learned in class that to swap two variables, we need to call the swapping function using references (pointers) of these two variables; within the function, we need to use an intermediate variable to hold the value of the first variable while saving the other to the first variable's address. This is the `swap_c_1` function which is provided. A former student of this class sent in a link regarding the implementation of the swap function without using the intermediate variable, as shown in https://teaearlgraycold.me/xor-swap-explained-visually/ [https://teaearlgraycold.me/xor-swap-explained-visually/].

In this workshop, we test if these functions work correctly by using unit test. The code of the `addSwap` and `xorSwap` functions in the link are modified a bit to obtain `swap_c_2` and `swap_c_3`, respectively, which are also given. You will see that they pass the unit test with a couple of test cases.

Also provided are functions to do the sorting of array elements by calling the swap functions. By using these swap functions, we wrote two functions to sort an array in ascending and descending order.

Tasks for the workshop

Download and unzip the file to your workspace. You should be able to see that there are two build targets in the project. See the other pdf file as for how to add more targets if you are interested.

You should be able to build both the **Application** and **Unit test** targets. You will be able to run both, although the former is not functioning well due to the missing of assembly functions.

Task 1 (20 points) is to add test functions and test cases for the assembly functions. You need to add test functions to all the four assembly functions in the assembly source file. You need to study the given example to see the pattern and write your test code as illustrated in the class.

Tasks 2 to 4 (20 points each) is to implement the three swap_asm function in assembly. Read the C function carefully and implement them in assembly. Keep your code as short as possible.

Team working and submission of the work

You are supposed to do the assignment in teams of two members. Besides the 80 points for programming, there are 20 points for following the assignment/submission requirements:

- Compress your project into a zip file `cec320_sec_x_ws5_lastname1_firstname1__lastname2_firstname2.zip` and submit it online. For the student ID used in the code, use the ID of the first student.
- You need also submit a pdf file which contains the screenshots of your test functions and the three assembly functions and the screenshots of the running results. Use the same naming convention as above C file but with a suffix of `pdf` for this file.