CSE 2312 Programming Assignment 3 – Fall 2021

Programming Lab Policies:

- Labs that fail to compile, or do not terminate correctly, will receive a zero.
- Labs that fail to compile, or do not terminate correctly, may not be resubmitted for a grade. This includes instances where students did not upload the correct file for grading.
- All labs must be submitted via the upload link on Canvas. Labs are not accepted through e-mail or Canvas Messenger.
- **Students must make a credible attempt to pass all programming labs to receive a passing grade in the course.**

Write ARM assembly implementations for all of the functions below. The functions must be present in a single .s file. Your function/procedure names must be identical to that presented below, as your implementations will be tested with generic C code used by the TAs.

All of your functions must return a value such that the program will run to completion with no segmentation faults. If a function cannot be successfully implemented, it still must return a valid value: **no function may be omitted**. Attempting to omit a function will result in a compile error.

A sample driver is included with this lab for student convenience. It is the student's responsibility to test their program with a sufficient range of values to ensure their code functions correctly. Programs will be tested with different values than are used in the sample driver.

Submit your assignment via the submission link on Canvas. The name of this file should be **lab#_lastname_loginID.s**. Example: If your name is John Doe and your login ID is jxd1234, your submission file name must be "lab#_Doe_jxd1234.s". Labs uploaded as a zip file, or labs that require a custom makefile, will receive a zero.

All questions worth eleven points unless otherwise noted.

1. float sumF32(float x[], uint32_t count);

// returns the sum of the elements in an array (x) containing count entries

2. float prodF32(float x[], uint32_t count);

// returns the product of the elements in an array (x) containing count entries

3. double dotpF64(double x[], double y[], uint32_t count);

// returns the dot product of two arrays (x and y) containing count entries

4. float maxF32(float x[], uint32_t count); * 12 points

// returns the maximum value in the array (x) containing count entries

5. double absSumF64 (double x[], uint32_t count);

// input: array (x) containing count double entries
// output: the absolute value of the sum of the elements in array (x) containing count entries

6. double sqrtSumF64(double x[], uint32_t count);

// input: array (x) containing count double entries

// output: the square root of the sum of the elements in array (x) containing count entries

7. char getDirection (BUSINESS business[], uint32_t index);

// input: array of BUSINESS structs and index of BUSINESS
// output: street direction of BUSINESS (e.g., 'N', 'S', 'E', 'W')

8. uint32_t getAddNo (BUSINESS business[], uint32_t index);

// input: array of BUSINESS structs and index of BUSINESS
// output: address number of business

9. char * getCity(BUSINESS business[], uint32_t index);

// input: array of BUSINESS structs and index of BUSINESS
// output: city name of BUSINESS

Bonus: void sortDecendingInPlace (int8_t x[], uint32_t count); * 15 points

// input: array (x) containing count entries
// output: array (x), with the values sorted in descending order
// Recommend selection sort

Notes:

- Students not pursuing the bonus points must still implement the sort function. Just return with `BX LR`.
- Flags from the VFP must be copied to the CPSR using the following instruction:
  `VMRS APSR_nzcv, FPSCR`
- BUSINESS will be provided in the sample driver.
- For a list of VFP instructions, go here:
  http://www.keil.com/support/man/docs/armasm/armasm_dom1361289934045.htm