March 26, 2014

CS160 ARM Programming Assignment 2

SORTING

Regular assignment: 150 points

Problem description

In this lab you will write three ARM assembly language subroutines for sorting numbers. You will use a C++ program for tying them all together. See your textbook and ARM lab manuals for details of ARM assembly language techniques.

Getting more specific

Download the zip file for your choice of platform (Windows or Linux) attached to this assignment on Blackboard. This contains the files necessary to complete the assignment, namely, the source files `sortmain.cpp` and `sortfuncs.s` which you will be modifying, and the executable `sort-solution` that you can run to see how the program should function. You are strongly encouraged to test your output against this executables to make sure that your program is displaying the correct behavior and formatting.

For 150 regular credit points, your job is to do the following:

(a 10 pts) Explain in your own words how `sortmain.cpp` fills in the three arrays. See www.plusplus.com/reference/X for details on the functions rand/srand (X=cstdlib), memcpy (X=cstring), and time (X=ctime). Place your explanation in a file called `sortmain.txt`.

(b 15 pts) Subroutine `sort1` works as follows. Each pair of elements from the N element long arrays listA and listB are compared. If the value of the element from listA is less than the value of the element from listB, the former is replaced by the latter. Write a few lines of C++ code that outlines how to achieve this result. Place this code in a comment field above the `sort1` ARM subroutine in the `sortfuncs.s` file. Hint: Follow the format of the pseudo code for the `sort2` ARM subroutine in the same file.

(c 50 pts) Write the ARM subroutine `sort1`. Include comments that refer back to the C++ code from (b). Test the code on different sized input arrays.

(d 25 pts) Subroutine `sort2` is given an N element long array which it sorts using the C++ code listed below and in the `sortfuncs.s` file. Explain in your own words how `sort2` which implements a so-called selection sort works. Place your explanation in the `sortmain.txt` file.

(e 50 pts) Write the ARM subroutine `sort2`. Include comments that refer back to the C++ code mentioned in (d). Test the code on different sized input arrays.

For bragging rights, develop and implement subroutine `sort3` which implements a so-called insertion sort. See Wikipedia for an algorithmic explanation. Basically, `sort3` is a more efficient version of `sort2`. During each pass thru a sublist, `sort2` swaps list entries LIST[k] and LIST[j] whenever LIST[k] $>$ LIST[j]. An alternative strategy is to keep track of the address of the largest value in the sublist and to perform, at most, one swap at the end of the sublist search. As before, write C++ pseudo code that outlines what needs to be done before you try to write the ARM assembly code.

Subtle and not-so-subtle hints

- Pay attention to how the C++ code passes arguments to the ARM subroutines and how it expects to receive arguments back.
- See the ARM Programming Techniques manual for a detailed description of ARM assembly language programming techniques. This manual is available from Blackboard under Course Materials.

  Keep in mind that the GNU assembler uses different directives.

  Do not print this document as it is 250+ pages long. View and print individual pages as you need to. Chapters 2–5 and 7 are probably those of most interest to you.

C++ code for selection sort

```
for (j=n-1; j>0; j=j-1) {
    for (k=j-1; k>=0; k=k-1;) {
        if (list[k] > list[j]) {
            tmp = list[k];
            list[k] = list[j];
            list[j] = tmp;
        }
    }
}
```

Compiling and assembling your code

```
arm-elf-g++ -c -mcpu=arm7tdmi -g sortmain.cpp -o sortmain.o
arm-elf-as -mno-fpu -marm7tdmi --gdwarf2 sortfuncs.s -o sortfuncs.o
arm-elf-g++ -o sort sortmain.o sortfuncs.o -lc
```

Due date and submission policy

Submit the `sortmain.txt` and `sortfuncs.s` files via Blackboard by the time specified under the ARM2 Lab Assignment tab.