The UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Comp 411 Computer Organization
Spring 2011

Lab #8: More pointers; function pointers
Issued Fri. 3/18/11; Due Fri. 3/25/11 (beginning of lab)

In this assignment, you will get more practice with pointers in C, including passing functions as arguments to other functions. You will essentially re-implement bubble sort from Lab 4, but this time a more general version that can work on arbitrary data types as long as a function to compare two objects is passed as an argument.

Exercise 1. Reading.
1. Read the entire Chapter 5 from the K&R book ("Pointers and Arrays") before even beginning this lab assignment. All of the material in that chapter is relevant. Section 5.11 in particular covers pointers to functions.

2. Read about MIPS floating-point registers and instructions. You can find these in the MIPS language reference in the textbook, or any number of websites. Specifically, you need to know how to compare two floating-point numbers, and how to move data between floating-point registers.

Exercise 2. Understand, compile and run bubble.c and bubble_generic.c
The first program, `bubble.c` is basically identical to the bubble sort implementation from Lab 4, except that the code has been partitioned into procedures. The second program, `bubble_generic.c` replaces the simple comparison operation "A > B" with a generic function "compare_fn" whose actual definition is an argument/parameter passed into the sort routine. Thus, the same bubble sort function can be used to sort floats in increasing order or in decreasing order simply by handing it a different compare function. Carefully study this generic implementation, compile it, and run it. (This implementation comes closer to how you can specify a "compare object/method" to a generic sort routine in C++ or Java.)

Exercise 3. Code bubble_generic in MIPS assembly
Take your C code for bubble_generic, and code the entire program in MIPS assembly. You should use single-precision floating-point operands and instructions in MIPS.

Submission: Submit your MIPS code and a copy of the program output for Exercise 3.