# MC48 Lab 1:
# Introduction to Assembly Language Programming

Max Hailperin

Due: October 12, 1995

The goal of this lab is to get familiar with the xspim or NeXTspim simulator, try out some of the simple programs we've been writing in class, and write and test a couple more programs of your own. In this lab, we'll steer clear of fancy things like recursion and other uses of memory; that's what the next lab is for.

Your lab write up can be short and sweet, but it should be English. You should communicate everything you did at some level of detail, but for the uninteresting parts, you can summarize in broad outline, while for the more interesting parts, you can go into detail. Specific details I will expect to see are the two programs you are asked to write, and test data and results from those two programs. Make sure your testing is systematic and thorough. Thorough doesn't mean using large quantities of weird numbers that still only test the same cases; rather, it means using carefully chosen tests to ensure that all cases are covered and all likely bugs exposed, while still allowing the answers to easily be hand-checked.

You can do this lab on either a NeXT or an SGI. If you use a NeXT, you will use the NeXTspim simulator, located in the /LocalApps directory. It is similar enough to the xspim simulator described in the book (but better) and similar enough to other NeXTstep applications, that you should be able to figure it out; let me know if not. If you use an SGI, then you will use xspim, as described in the book. The simplest way to get at it is probably to use the "Find an icon" menu item to find the xspim icon, then drag it onto your desktop; thereafter you can just double-click it to run it. Let me know if you have troubles with this or other SGI/X-windows user interface stuff.

Start by stepping through simple programs, like one to add two numbers or the one from class for calculating the absolute value of a number. This

part of the lab is to make sure you are comfortable with SPIM.

Now you have two programs to write and test and debug (if necessary). You can do these in whichever order seems more natural for you.

1. Write an adding machine program which repeatedly reads in integers and adds them into a running sum. It should stop when it gets an input that is 0, printing out the sum at that point. To read in and print out integers, you will need to use two of the system calls describe on pages A-45 and A-46. You can use another of them to exit the program at the end. You should put the label `main`: at the beginning of your program. If you are having trouble with details like how to use the system calls, you should be particularly quick to ask for help; I don't want you getting hung up on that kind of thing.

2. Write a program that reads in three integers and prints out the sum of the largest two of the three. You'll need to use the system calls, as above. You can break ties arbitrarily. (Hint: you may find it easier to think about which one number is smallest, not which two are largest.)