Programming Assignment 1

ECE 109

DUE: March 26th, 2008 11:45pm via WolfWare Submit

Leave the program in a subdirectory csci/255/Prog1 of your CS account. Just like programs were submitted in CSCI 201

Overview:

In this programming assignment you will write a program in LC-3 that will add a list of numbers stored in memory and place that value in the register. This assignment will also introduce the use of the LC-3 Simulator for executing and debugging your programs.

Google for "lc3 simulator" or http://highered.mcgraw-hill.com/sites/0072467509/student_view0/lc-3_simulator.html

LC-3 Simulator:

Directions for downloading and installing the LC-3 Simulator are available at the problem session website at http://courses.ncsu.edu/ece109/common/index.html. For Windows the typical extract path is c:\lc3. For a lab computer you should extract it to the Desktop.

For Windows you will use LC3Edit to write your programs and assemble the program into an object file through this program. For Mac and Linux users, you will use a regular text editor and use the assembler tools to assemble your program into an object file.

Included at the website is an extensive tutorial for using the simulator, and more details will also be provided in problem session and class.

This program can be written in binary, hexadecimal, or assembly. You will submit either a **.bin**, **.hex**, or **.asm** file depending on which format you use to write it. You should NOT submit the .obj file.

Program Overview:

Often times in programs we write, we will have a list of data to manipulate stored somewhere in memory. In this program we will be given a starting address for a list of numbers stored in consecutive memory locations, and we will calculate the sum of all items in the list. The list will contain only positive numbers or zeroes. The end of the list is denoted by a negative number. Figure 1 shows an example of what the list will look like in memory.

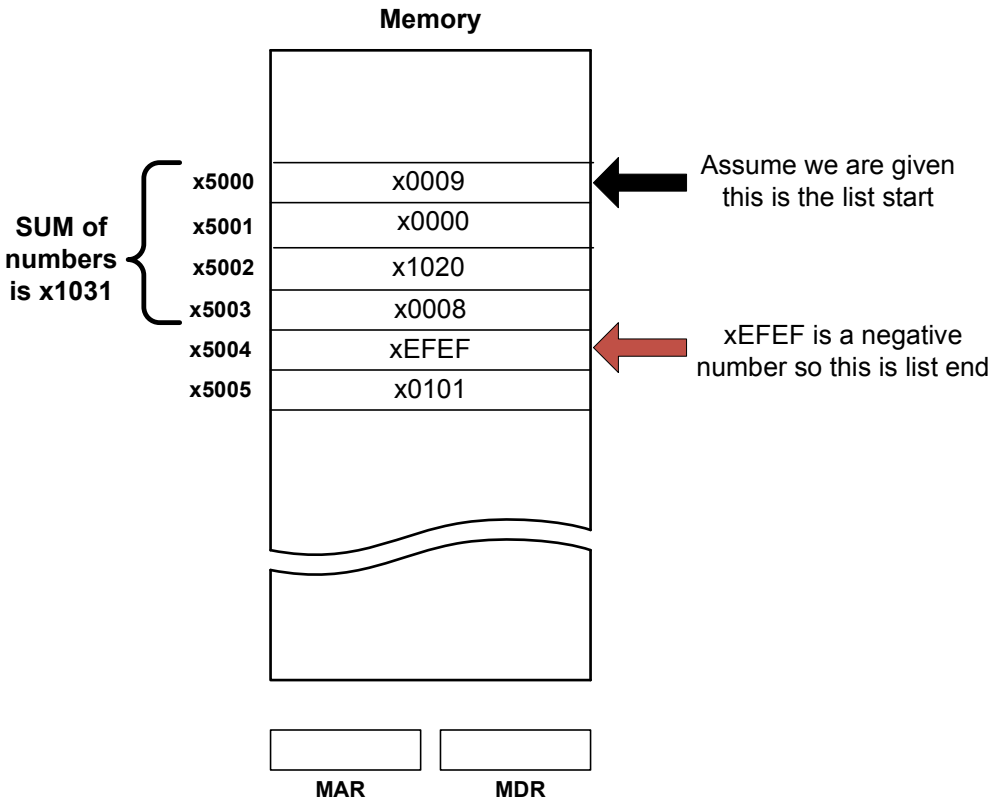
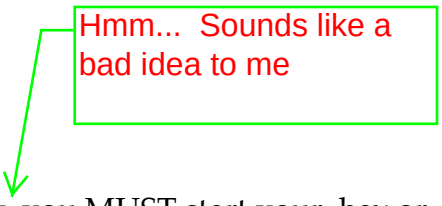


Figure 1 – Example of List in Memory

In Figure 1 we are given that the list starts at x5000. For the program **the start of the list** is placed into R1 prior to the beginning of program execution. To achieve this, before starting execution in your simulator, you should set the value in R1 to be the starting address of the list to sum. In Windows you simply double-click the register and manually set the value before running the program. A similar method is used for the linux and mac simulators.

In your program, the final sum value should be placed into R2. In essence your program will perform the following steps.

1. Clear the SUM register, R2.
2. Load the value from memory at the address in R1
3. If the value is negative jump to step 7
4. Add the loaded value to the running sum in R2
5. Increment R1
6. GOTO step 2
7. End Program



Writing the Program

If you are writing in hexadecimal or binary, you MUST start your .hex or .bin file with the starting address of the program. Your program should start at location x3000. So your program's first line should be either x3000 for hex or 0011000000000000 for binary.

If you are writing in assembly, the first line of your program should be .ORIG x3000. You must also end your program with .END

Writing programs in assembly will be covered both in class and in problem session.

You should also comment your program! In order to receive full credit you MUST comment your program. A program without comments will receive ZERO partial credit. To comment a program, use a semicolon to begin the comment, and the comment will extend until the end of line.

For example in a binary program where we simply add the values in R1 and R2 and place the result in R0.

```
0011 0000 0000 0000   ; Begin Program at x3000
0001 000 001 000 010  ; Performing the addition
1111 0000 00100101    ; HALT program
```

Comments should be descriptive and help you and the TA decipher your intentions as a program. If you use an instruction like AND R0,R1,R2 then a poor comment would be one that says, "ANDing values in R1 and R2 and placing result in R0," since this is already obvious by looking at the instruction. Comments are not required on every line. They should just clarify important points of your program.

Running the Program

Three files have been provided on the problem session website list1.obj, list2.obj, and list3.obj. Each list has a different starting address and a different final total. To run the program you should FIRST load the list object file you wish to test, then load your program's object file. Then set the value in R1 to correspond to the starting list address (See Table 1). After running your program, the value in R2 should match the list total given in Table 1.

You may also make lists of your own to test your program. When graded we will use some of the given lists as well as lists of our own to test your program. Be sure your program works for all cases, not just the given test cases!

List Name	Starting Address	SUM
list1.obj	x5000	x9549
list2.obj	x2000	x7335
list3.obj	x6000	x0000

In your UNCA CSCI account, leave your file in a directory called csci/255/Prog1

Submission and Grading

Your program is due on Wednesday March 26th, 2008 by 11:45pm via Wolfware Submit. Use http://submit.ncsu.edu to submit your .bin, .hex, or .asm file. DO NOT submit your .obj file.

Late submissions will be accepted for a penalty. Use the late submission link to submit a late assignment. Rounding up, late submissions will result in a deduction of 10 points per hour, for every hour past the due time.

The program will be graded via a detailed rubric to allow for partial credit. However, in general the following grading scheme will be used.

Program passes all test cases and is commented – 100%
Program passes all test cases and contains NO comments – 90%
Program assembles, does not pass test cases, contains comments, but shows significant effort – 50%
Program does not assemble, does not pass test cases, contains NO comments, shows significant effort – 0%
No submission / Significant Effort not shown – 0%

It will be left to the grader's discretion to define "significant effort."