ECE-3120
Fall 2008

LAB 3 –PROGRAM LOOPS

The purpose of this lab is to introduce you to basic programming the 68HCS12 using program loops and arithmetic instructions. You will **NEED** to study about the arithmetic and branching instructions in Huang chapter 2 before attempting this program.

PRE-LAB:

Prepare pseudocode and the first draft of the program and calculate the expected results by hand. This must be completed before coming to the lab and shown to the lab instructor at the start of the lab session.

Approved: Lab TA _____ *Date* _____

PROGRAMMING ASSIGNMENT:

Write a fully-commented program for the HC12 board, including appropriate directives and labels for memory operands and constants, called **Loop.asm**. It must use a single program loop for most of the work. The program should do the following:

- Be able to process any array containing 1 to 255 signed data bytes as follows. Each byte can thus have a value between -128 and +127. This means that the program should be able to handle any size array in this range, with any set of values, by changing ONLY the directives that create the array values and array size, without modifying any instructions.
- Calculate the sum of all the positive numbers in the array.
- Calculate the sum of all the negative numbers in the array.
- Count the number of zeros in the array.
- Calculate the total sum of all numbers in the array.
- The code must start execution at $1000.

The program directives should initialize the following **signed decimal data byte array** in the data space (starting at $1200):
98, 130, -127, -3, 4, -15, -1, 0, 1, 140, 57, -55, 0, -99, in this order.

The program should store in memory:

- The sum of positive numbers in memory location $1300:$1301
- The sum of negative numbers in memory location $1302:$1303
- The number of zeros in memory location $1304
- The total sum in memory location $1306:$1307

Procedure: First, use D-Bug12 to fill memory locations $1000 through $3BFF with zeros. Then assemble, download, and debug/execute the program as follows.

a. Single-step through the program until it completes the first loop iteration and is ready to branch back to the start of the loop. Verify that each change is correct.
b. Then set a breakpoint to stop execution at the end of each loop and run through rest of the program to the end, pausing at each breakpoint to display the important results. Verify that each result is correct at the end of each loop and that all four final results are correct at the end of the program. Use the listing file to determine the breakpoint address. Remember that the instruction at the breakpoint address is NOT executed when the break occurs; it is only executed when you resume execution after the break.
c. Then reset the processor (which removes the breakpoint), download the program again, run it at full speed until it stops, and verify that the four final values are still correct.
d. When finished debugging and executing, copy the entire terminal window output and paste it into a Notepad or Word document for inclusion in the report. You should edit out mistakes and unnecessary repetitions before submission.

Approved: Lab TA _____ *Date* _____

Things to turn in as your Lab Report, attached in this order:

1. This assignment sheet, with your name at the top, signed by the TA where shown.

2. Your uncorrected pre-lab document (commented source code).

3. A printout of the final **Loop.asm and Loop.lst** files. You'll need to print the listing file in landscape mode to make it fit. Use Notepad to print them. Do NOT use MiniIDE to print the listing – it will not look right.

4. A printout from the terminal screen (method: highlight text, type ctrl+c, and paste into a Notepad or Word document, using Courier New as the font) that includes everything done. Add brief hand-written comments and highlight important results at each step in the procedure, showing the relevant memory contents and register contents.

5. Answer the following questions, in your document:

    1) What are the four final values, in both hex and decimal? Label each value clearly.
    2) Are the specified final answer sizes (bytes or words) really sufficient for all possible arrays, within our constraints? Explain.
    3) How many memory bytes, in both hex and decimal, are used by your program code (excluding data)? (Hint: use the listing file, Loop.lst.)
    4) Do you think loops are important for microcontrollers? Why or why not?
    5) Why do we use block fill? Could there be any repercussions if we didn't?