CS 273 Lab 5: More Conditionals, Branches, and Loops

Dept. of Computer Science, NMSU

Fall 2006

Due Date: Next class after your lab session.

In this lab you will handle multi-byte values – in other words, values which are more than 8 bits. In part one, all the values are 16 bits, or 2 bytes, but in part two your result will be a 4-byte value (32 bits). On the HC11, and other Motorola CPU's, all multi-byte values are stored with their most significant byte in the lowest address. For example, if you have

```
result      RMB         4
```

in your program, then the highest order bits are at the address `result`, on down to the lowest order bits at the address `result+3`. This storage scheme for multi-byte values is called *big-endian*, because the big end comes first in memory. Some CPU's, such as all Intel x86 (and thus Pentium) CPU's, use *little-endian* storage, which is exactly the opposite. Thus, in this lab you must produce results in big-endian order, since your program is running on a Motorola HC11.

Problem 1

Write a program which will add two 16-bit unsigned values and produce a 16-bit unsigned result (in memory). There are some restrictions on how you program this. First, the HC11 does have an instruction that does this already (`ADDD`), but you may NOT use this in your solution. You also may not use the `ABX` or `ABY` instructions. Finally, you must only use one of the 8-bit registers A or B, but not both.

Your program must also use the I/O features of the simulator (and HC11). In particular, you must read in the input data from address $1004, and you must output the result data to address $1003. An input action is a load-instruction, and an output is a store-instruction. The order of input must be: first input value, high-order byte; first input value, low-order byte; second input value, high-order byte; second input value, low-order byte. Thus, you will probably need to store the input data in some variables in order to use it.

HINTS: Don't forget to declare your variables in memory as

```
RMB     2
```

so that it is a 2-byte variable. Furthermore, you can use expressions such as `label+1` on instructions. So, for example, the instruction

```
LDAA label+1
```

would load A with the byte at the location AFTER the location pointed to by the label. As a last hint, this program CAN (and should) be done without any branches: look for some *special* add instructions. Find the right instruction.

Questions (as part of your lab report):

1. For the input values decimal 37298 and 17567, write down the hex values these numbers convert to, and the resulting sum your program produces, in hex and decimal. Also write down what the C bit in the condition code register is when your program completes the summation.
2. Do the same for input values 79 and 91.
3. Do the same for input values 37298 and 31333.

Problem 2

Using what you learned in Problem 1, now you must write a program that will multiply two 16-bit unsigned numbers and produce a 32-bit unsigned result. As with problem 1, the two input values will be input from address $1004, but the result does not need output and should only end up in a 4-byte variable in RAM. You can use whatever registers you need, but you cannot use the `MUL` instruction and you cannot use the `ADDD` instruction. (HINTS: `LDD` and/or `STD` can make programming a bit easier, and don't overlook the X register and the instructions that use it. Also remember that a multiply can be viewed simply as successive adds.) The naive solution will result in a program that takes many thousands of cycles, especially for the input values of question 1 below. This is all you have to do for this lab. But a solution can be created which takes only perhaps a hundred or a couple of hundred cycles. For 2 extra credit points, devise and submit a second solution to this problem which, even for the inputs for question 1, takes less than a thousand cycles. (Hint: a web search for multiplication algorithms might help. But be sure to write your own HC11 solution!)

Questions (as part of your lab report)

1. For the input values decimal 3941 and 467, write down the resulting product your program produces, in hex and decimal. Also write down what the C bit in the condition code register is when your program completes the multiplication. How many cycles did your program take to complete? If this was running on a 2M Hz HC11 CPU, how much real time is that?
2. For the input values decimal 37298 and 17567, do the same as you did for question 1.

Lab Turn-in

For this lab, you must hand in the following:

1. Your source code for Problem 1
2. Your source code for Problem 2
3. A lab report in the template format including
   (a) answers for all the questions in each part, and
   (b) an estimation of the time it took you to complete this lab.

Submit these through the Web Submission Page at

`http://www.cs.nmsu.edu/~joemsong/273/upload/`

Due Date: Next class after your lab session.