Lab 17 in Winter 2003

Lab 16: Introduction to MC68000 and the Mas Assembler

Revised 5/16/00

Introduction

In this lab, you will work with the MAS Assembler to learn about computer operation and how to work with assembly language programs. You should familiarize yourself with the *Macintosh Assembly System* manual, particularly Ch. 12, and bring this manual to lab. You should also bring the excerpts from the *Programmer's Reference Manual* by Motorola.

1. Tutorial

The MAS Assembler has a tutorial and extensive help. You should spend some time working through the tutorial (Ch. 2,3, and 4). The amount of time depends on how familiar you are with the Macintosh user interface and computer programming. The MAS "help" facility has a full list of assembler commands and the built in MAS subroutines. You may want to look at this, but the information is also in the manuals. Be sure to learn how to use the debugger since this allows you to observe all aspects of the computer in operation.

2. Counting Program

In the tutorial, you looked at a program, 'Countdown.a,' that counted down from a number that you entered:

```
;     Fasten your seat belts.  The countdown begins. Get ready
;     to blast off into the friendly world of MAS.

      xref    strout, decin, decout, newline, stop

start:  lea msg1,a0
      move.w      #34,d0
      jsr   strout
      jsr   decin         ;  enter the count
loop: jsr   decout        ;  the countdown loop
      jsr   newline
      dbra  d0,loop
      lea     msg2,a0
      move.w  #17,d0
      jsr     strout
      jsr   newline
      jsr   stop

      data
msg1: dc.b  'Enter the time for the countdown: '
msg2:   dc.b    'Welcome to MAS!!!'
      even

      end
```

Run this program using the debugger and observe the contents of registers A0 and D0 as you single-step through the instructions. Also observe the branch operation in action. Look at the memory locations containing `msg1` and `msg2`. Do you recognize how the ASCII characters are stored as numbers? Do you see how register A0 keeps track of the location of the beginning of the message strings?

See if you can modify this program so it will count **up** to the number that you entered instead of down. Thereare several ways to do this. Some of the commands that one can use are: add, compare, and branch. The specific instructions that a previous TA used were: addi, cmp, and bpl, respectively. In your lab notebook, include your program with full comments. Comment every line, yes every line, and make special note of the lines that you added or removed.

3. Program to Read and Store ASCII Characters

Examine the operation of the program to read and store 20 ASCII characters in the attached note (and repeated below). Use the debugger to observe how indirect addressing works to store characters in successive memory locations "automatically" (i.e., in the `move.b` instruction in the loop).

```
; Program to read and store 20 ASCII characters

       xref     getchar, stop        ;indicates that these are external

start: lea      msg,a1               ; put address of msg in a1
       move.w   nchar,d1             ; number of bytes to read
       jmp      enter                ; enter loop at end
loop:  jsr      getchar              ; getchar puts the character in d0
       move.b   d0,(a1)+             ; move the character to memory
enter: dbra     d1,loop              ; subtract 1 from d1
                                     ;   and see if done
       jsr      stop                 ; goes here when d1 equals -1
       data
msg:   ds.b     100                  ; set aside 100 bytes of storage
nchar: dc.w     20                   ; number of characters to read

       end
```

How could you modify this program to change any upper case alphabetic character to lower case before storing it? To begin with, assume that all the charecters are alphabetic so you don't have to check that this is the case. *Hint:* look at the bit pattern of an upper case character compared with its lower-case equivalent.

As a further challenge if time permits, try to figure out how you could test that you were dealing with an upper case alphabetic character so the input could be any ASCII character. There is an example of a test for numerical characters in the final program in the attached note (you could use appropriately modified code "in-line" rather than worrying about the subroutine used in the last example).