

Mathematics 3670: Laboratory 10

Pre-lab Exercises

1. Examine the assembly language program given in `lab10a.asm`. This program uses three different addressing modes: LD, LEA, and LDR. Using Appendix A as a reference, determine the difference between each of these.
2. The program in `lab10a.asm` is logically composed of two parts: lines 1–17 and lines 19–48. The program uses one integer variable (line 49) and one array of characters (line 50). Carefully trace the execution of lines 1–17. Make a table which shows how the registers R0, R1, R2, and R3 change as the loop makes progress. How will memory change as a result of executing this loop?
3. In a similar way, carefully trace the execution of lines 19–48, keeping track of all registers in use and showing how they change. How will memory change as a result of executing this loop? What output will be produced by the PUTS trap routine?
4. Using `lab10a.asm` as a model, design LC-3 assembly language programs which perform the following tasks:

- (a) Process a given string X, changing any lower-case letters to their equivalent upper-case counterparts. The change should be done “in place” — after your program runs, the original characters of X will be replaced with the revised string. For example, if this is done for `lab10a.asm`, then the string which begins at symbolic address X would be transformed to become:

MATH 3670 IS FUN!

- (b) Copy the string X to another group of memory locations, beginning at the symbolic address Y, filtering out all non-alphabetic characters. As a result, the characters stored in the destination will now have only characters between 'A' and 'Z'. Assume that X is no longer than 80 characters. You can declare space for Y as follows:

Y .BLKW 81 ; Up to 80 characters, plus the terminating NUL

- (c) Determine whether or not the string stored at Y is a palindrome. A **palindrome** is a string which reads the same in either direction — for example, “*Madam, I’m Adam*” and “*race car*” are two famous palindromes.

Since Y has already been stripped of non-alphabetic characters, your test would be applied to MADAMIMADAM and RACECAR in these two examples.

Lab Exercises

1. Obtain copies of this week's lab files and place them in your **lc3** folder.
2. Launch the LC-3 simulator, then perform the following actions:
 - (a) Assemble `lab10a.asm`.
 - (b) Load `lc3os.obj`.
 - (c) Load `lab10a.obj`.
 - (d) Set breakpoints at `x3000`, `ELOOP`, `ELOOP2` and `STOP`.
 - (e) Execute the program. Use the **Continue** button to quickly reach the program code, then single-step through the first loop, watching the register values change in response to the instructions. Don't rush: the goal is to understand each and every instruction in detail. How do your predictions from the pre-lab match the reality?
 - (f) In a similar way, step through the second loop. Once again, don't rush through this.
 - (g) Test this program using a variety of strings. To do this, you need to modify `lab10a.asm`, assemble it, and reload it. Your test cases should include both even and odd-length strings and short strings. Does the program behave correctly for all such strings?
3. Implement the three programs described in the pre-lab exercises, testing carefully as you go. Put your programs in `lab10b.asm`, `lab10c.asm`, and `lab10d.asm`. For the final program, you should be able to place an arbitrary string in `X` and your program should announce—yes or no—whether or not it is a palindrome. (You don't need to be able to enter the string at run-time; instead just enter it in the assembly language program.)

Submissions

Before submitting your work, make sure your name appears in each of the programs you wrote. Also, ensure that each of your programs is generously commented.

Create a **lab10** folder and place copies of all `.asm` programs you wrote in this folder. Submit the folder by dragging it onto the EIU submission icon.

Appendix

Contents of lab10a.asm

```

1      ;;
2      ;; Author: Bill Slough
3      ;;
4      ;; Determines the length of a string, then reverses and displays it.
5
6      .ORIG x3000
7
8      ;; Determine the length of the string X
9      LEA    R0,X          ; R0 = addr(X)
10     AND    R1,R1,#0      ; i = 0
11     LOOP   ADD    R2,R0,R1 ; while (X[i] != '\0')
12         LDR    R3,R2,#0  ;
13         BRz    ELOOP    ;
14         ADD    R1,R1,#1  ;      i = i + 1
15         BR     LOOP     ;
16     ELOOP  ; end while
17         ST     R1,N      ; N = length(X)
18
19     ;; Now reverse the string X
20     AND    R0,R0,#0      ; low = 0
21
22     LD     R1,N
23     ADD    R1,R1,#-1     ; high = N - 1
24
25     LOOP2  NOT     R2,R1  ; while (low < high)
26         ADD    R2,R2,#1  ;
27         ADD    R2,R0,R2  ;
28         BRzp   ELOOP2   ;
29
30         LEA    R2,X      ;
31         ADD    R2,R2,R0  ;      R2 = addr(X[low])
32
33         LEA    R3,X      ;
34         ADD    R3,R3,R1  ;      R3 = addr(X[high])
35
36         LDR    R4,R2,#0  ;      low_char = X[low]
37         LDR    R5,R3,#0  ;      high_char = X[high]
38
39         STR    R4,R3,#0  ;      X[high] = low_char
40         STR    R5,R2,#0  ;      X[low] = high_char
41
42         ADD    R0,R0,#1  ;      low++
43         ADD    R1,R1,#-1 ;      high--
44         BR     LOOP2    ;
45     ELOOP2 ; end while
46         LEA    R0,X
47         PUTS           ; write(X)
48     STOP   HALT      ;
49     N      .BLKW      1
50     X      .STRINGZ   "Math 3670 is fun!"
51     .END

```