CS270 Recitation 10
"LC-3 Input/Output/Functions"

Goals

To improve your understanding of LC-3 input, output, and function calls, via a practice session.

The Assignment

Make a subdirectory called R10 for the recitation assignment. Download the R10.asm source file as a starting point for the exercise.

http://www.cs.colostate.edu/~cs270/CurrentSemester/recitations/R10/R10.asm

The program you are starting with reads a string in hexadecimal, echoes the string, then converts it to an integer. All code is in the main program and the conversion function is incomplete. The object of this recitation is to:

1) Complete the code to convert from a string in hexadecimal format to an integer.
2) Modularize the program to implement the code as two separate functions, using parameter passing in registers.

The first function should have the label InputHex. It should take one parameter that is the memory address of the string in the R0 register, and should return after exactly 4 characters have been input by the user. Do not modify the data labels in the provided code. The main program will print the prompt shown in italics: *Input: 0x*ABCD

The second function should have the label ConvertHex. It should take one parameter that is the memory address of the string in the R0 register, and return the converted integer value in R1. See the specification below for extra conditions on the input and output.

Discussion

The input function is accomplished via TRAP instructions that call system function to read and write characters. For previous assignments in this class, we used a memory protocol to pass parameters, now we are implementing parameter passing using registers. Compare the two methods and the TA will discuss the differences. Is there a problem with both methods that will be solved with a stack solution?

Algorithm

The algorithm for converting a hexadecimal string is as follows:

1) Initialize the result to zero.
2) For each character in the string:
    a. Multiply the result by 16.
    b. Convert the next character to a temporary value .
    c. '0' to '9' maps to 0..9
    d. 'A' to 'F' maps to 10..15
    e. Ignore all other values
    f. Add the temporary value to the result.
3) Return the result.

Specification

You can assume the user will only try to input 4 characters, and that all characters will be legal hexadecimal digits, and that only uppercase letters will be entered. Note that constants that are defined, these may come in useful.

Code Listing

```
; Recitation 10
;-------------------------------------------------------------------------------
; Begin reserved section: do not change ANYTHING in reserved section!

                .ORIG x3000
                BR Main

; Variables
String          .BLKW 4               ; 4 characters
Value           .FILL 0               ; 1 integer
Prompt          .STRINGZ "Input 0x"   ; Prompt

; Constants
DigitStart      .Fill -48            ; ASCII '0' (negative)
DigitEnd        .Fill -57            ; ASCII '9' (negative)
LetterStart     .Fill -65            ; ASCII 'A' (negative)
LetterEnd       .Fill -90            ; ASCII 'F' (negative)

; End reserved section: do not change ANYTHING in reserved section!
;-------------------------------------------------------------------------------

Main            LEA R0,Prompt         ; prompt pointer
                TRAP x22              ; print prompt

InputHex        LEA R1,String         ; initialize pointer
                AND R2,R2,#0          ; initialize counter
                ADD R2,R2,#4

InputLoop       TRAP x20              ; read character
                TRAP x21              ; echo character
                STR R0,R1,#0          ; store character
                ADD R1,R1,#1          ; increment pointer
                ADD R2,R2,#-1         ; decrement counter
                BRp inputLoop         ; loop

ConvertHex      LEA R0,String         ; initialize pointer
                AND R2,R2,#0          ; initialize counter
                ADD R2,R2,#4
                AND R1,R1,#0          ; initialize result

ConvertLoop                           ; Your code here!

                ADD R0,R0,#1          ; increment pointer
                ADD R2,R2,#-1         ; decrement counter
                BRp ConvertLoop       ; loop

Return          ST R1,Value           ; store result
                HALT

;-------------------------------------------------------------------------------
                .END
```