Lab #9 – C Interface with Pointers

Goals:

Write assembly functions on the Discovery Board that interface with pointers and arrays of bytes in a C Main program

For each step enter the necessary responses into the Canvas assignment text entry tool to let the TA follow your work.

Remember the 'SVC #0' debugging instruction available with your Discover Board library! That can be inserted in your assembly code any time you need to clarify exactly what is in the registers!

Refer to the Lecture slides for the following:

Part 1:
a) Find the ASCII character table on slide 48 of the Week 1 slides. Find the single bit position in the binary codes that determines whether the characters 'A' through 'Z' and 'a' through 'z' are either upper case or lower case.

b) Download Lab09-ASCIIMain.c and Lab09-ascii.s from the Canvas files area, and in the LabCode folder. Complete the assembly code for the three functions `lo2up`, `up2lo`, and `findchr` in the assembly file. Note the prototypes for these functions in the C main program that declare the number and types of the function arguments and return values.

The `lo2up` function takes a single argument of type pointer to char that points to an array of character variables that forms a character string in the C main program. You can expect that after the sequence of ASCII characters in the array, there will be a "null byte", that is, a byte of value 0, that terminates the string. The function must use the pointer argument to step through the array elements in sequence and turn any lower case ASCII character found into its corresponding upper case character. Form a loop in the assembly function that tests each character in the array, makes a change in the array when needed, and terminates when a '0' byte is encountered, then returns to the calling program. To make the change from lower to upper case, use a bitwise logical assembly instruction operating on the string character at the bit position you identified in a). Any character not falling in the ASCII code range 'a' through 'z' must **not** be altered. Note that you will need to use the form of the load and store register instructions that read and write single bytes to and from memory.

The `up2lo` function is similar to `lo2up`, and can be written easily by first starting with a copy of `lo2up` and edited so that any upper case ASCII character is changed into its lower case equivalent. To make the change from upper to lower case, use a bitwise logical assembly instruction operating on the string character at the bit position you identified in a). Any character not falling in the ASCII code range 'A' through 'Z' must **not** be altered.

The findchr function is similar to the C library function `strchr`. It expects two function arguments, a character pointer pointing to a string array, and a single character. A loop in the function tests each character in the string array in sequence, starting from the beginning of the string, searching for an occurrence of the second argument character. If the search character is found somewhere in the string before the null character is encountered, the function returns the pointer to that first occurrence of the search character in the string. If no occurrence is found, the function must return the "null pointer", or an address of 0, to the main program to signal a search failure.

The C main program will test your three functions. A character string will be passed to the two case conversion functions `lo2up` and `up2lo`, and after each function returns the string is printed so you may verify the proper case changes. Then two different searches are performed on a character string by the `findchr` function, one for the character ‘a’ and another for ‘p’, and the return value pointer is tested for the null pointer after each call. The first search should succeed, and the character that the return value points to is displayed. The second search should fail, and the C program report that result as well.

Upload your code and a cell phone photo of your display screen showing the string processing.

Part 2: Time to work on the current programming project assignment.