CSE 379
Lab #4
Part #2
Spring 2022

Objective
In this part of the lab, you will build off of what you started in Part #1, learning how to use general purpose I/O to interface hardware with the ARM processor and begin to create a library for your subroutines. You will utilize four LEDs on the Alice EduBase board, the four momentary push buttons on the Alice EduBase board, the RGB LED on the Tiva board, switch 1 (SW1) on the Tiva board, and the keypad on the Alice EduBase board.

Description
Write and test five ARM assembly language subroutines, called *read_tiva_push_button*, *read_from_push_btns*, *illuminate_LEDs*, *illuminate_RGB_LED*, and *read_from_keypad*. Once written, write a subroutine, called *lab4*, which has a menu that allows you to repeatedly test each of the subroutines until the user decides to exit the program. Note that instructions on how to use the program (such as what the menu choices are and what user input is expected) is required.

The Library
Incorporate the following subroutines into a separate library file called *library_lab_4.s*.

- *uart_init* initializes the user UART for use.
- *output_character* transmits a character passed into the routine in *r0* to *PuTTy* via the UART.
- *read_character* reads a character from *PuTTy* via the UART, and returns the character in *r0*.
- *read_string* reads a string entered in *PuTTy* and stores it as a null-terminated string in memory. The user terminates the string by hitting *Enter*. The base address of the string should be passed into the routine in *r4*. The carriage return should NOT be stored in the string.
- *output_string* displays a null-terminated string in *PuTTy*. The base address of the string should be passed into the routine in *r4*.
- *read_from_push_btns* reads the momentary push buttons, and returns the value read in *r0*. Push button 2 should correspond to the MSB and 5 to the LSB.
- *illuminate_LEDs* illuminates the four LEDs. The pattern indicating which LEDs to illuminate is passed into the routine in *r0*. Bit 3 corresponds to LED 3, bit 2 to LED 2, bit 1 to LED 1, and bit 0 to LED0.
- *illuminate_RGB_LED* illuminates the RBG LED. The color to be displayed is passed into the routine in *r0*. How the individual colors are encoded when passed into the routine in *r0* is up to you. You should provide for the RGB LED to be illuminated red, blue, green, purple, yellow., and white.
- *read_tiva_push_button* (originally labeled *read_from_push_btn* in Part #1, changed here for clarity for future use) reads from the momentary push button (SW1) on the Tiva board, and returns a one (1) in *r0* if the button is currently being pressed, and a zero (0) if it is not.
- *read_from_keypad* reads a keypress on the keypad, and returns the value corresponding to the key that was pressed in *r0*. Only keys 0 through 9 will be tested.

Skeleton Code

The following skeleton code shown below can be used to get you started with your library file.

```
        .text
        .global uart_init
        .global output_character
        .global read_character
        .global read_string
        .global output_string
        .global read_from_push_btns
        .global illuminate_LEDs
        .global illuminate_RGB_LED
        .global read_tiva_push_button
        .global read_keypad

uart_init:
        POP {lr}    ; Store register lr on stack

            ; Your code is placed here

        POP {lr}
        MOV pc, lr

output_character:
        PUSH {lr}   ; Store register lr on stack

            ; Your code is placed here

        POP {lr}
        MOV pc, lr

read_character:
        PUSH {lr}   ; Store register lr on stack

            ; Your code is placed here

        POP {lr}
        MOV pc, lr


read_string:
        PUSH {lr}   ; Store register lr on stack

            ; Your code is placed here

        POP {lr}
        MOV pc, lr
```

output_string:
```
      PUSH {lr}   ; Store register lr on stack

          ; Your code is placed here

      POP {lr}
      MOV pc, lr
```

read_from_push_btns:
```
      PUSH {lr}   ; Store register lr on stack

          ; Your code is placed here

      POP {lr}
      MOV pc, lr
```

illuminate_LEDs:
```
      PUSH {lr}   ; Store register lr on stack

          ; Your code is placed here

      POP {lr}
      MOV pc, lr
```

illuminate_RGB_LED:
```
      PUSH {lr}   ; Store register lr on stack

          ; Your code is placed here

      POP {lr}
      MOV pc, lr
```

read_tiva_pushbutton:
```
      PUSH {lr}   ; Store register lr on stack

          ; Your code is placed here

      POP {lr}
      MOV pc, lr
```

read_keypad:
```
      PUSH {lr}   ; Store register lr on stack

          ; Your code is placed here

      POP {lr}
      MOV pc, lr

      .end
```

To access these routines from *lab_4.s*, be sure to declare the labels as global, as shown below.

```
.text
.global uart_init
.global output_character
.global read_character
.global read_string
.global output_string
.global read_from_push_btns
.global illuminate_LEDs
.global illuminate_RGB_LED
.global read_tiva_push_button
.global read_keypad
.global lab4
```

```
lab4:
      PUSH {lr}   ; Store register lr on stack

          ; Your code is placed here

      POP {lr}
      MOV pc, lr

      .end
```

Partners

You will work with a partner in this lab. Your partner *MUST* be the same partner you worked with on lab #3.

Documentation

Your program must be clearly commented, and documentation must also be provided. The documentation must follow the guidelines covered in lecture (found on the *Lectures* webpage of the course website). Your comments should describe what *each* section of your program does. To receive full credit on your documentation, you must submit a draft of your flowchart before you start working on the lab in your regularly scheduled lab time on Wednesday, March 9 or Thursday, March 10.

Submissions

Your source code (C and assembly) must be submitted online using the submit command (submit_cse379 Your source code (C and assembly) and your documentation (as a PDF) must be submitted online using the submit command (*submit_cse379 lab_4_wrapper.c lab_4.s lab_4_library.s lab_4_documentation.pdf*) on *timberlake.cse.buffalo.edu* ***before 11:59 PM on Sunday, March 13, 2022***. Your documentation will be used along with the code you submitted when you perform the debug exercise for Lab #4.