EMBSYS100 - AU19
ASSIGNMENT 06

Goal

The goals for the assignment this week:

1. Practice the use of the Cortex Microcontroller Software Interface Standard (CMSIS).
2. Gain more practice with the Cortex-M4 assembly language.
3. Become familiar with the linker Map file and use it to determine the resource usage.

Problems:

1. Use the CMSIS to implement code that blinks the user LED on the STM32 board.
    a. Create a new project.
    b. Create a main.c file and add it to the project.
    c. Add the files "**stm32f401xe.h**" & "**system_stm32f4xx.h**" to the folder where "main.c" is saved. *You should be able to get these files thru STM32CubeMX. You could also get them by downloading the zip file "CMSIS_STM32_Device_Specific_Files.zip" from canvas site under the link Assignment\A06 folder*
    d. Enable use of CMSIS in project options settings.
    e. Implement toggling of the LED using the CMSIS data structures.

2. Convert the blinking led program into assembly code.
    a. Go to the link Assignment\A06 folder and download the zip file "*Module07_Assignment06_Starter_Code.zip*". Use the skeleton files (**main.c**, **user_led.s**, and **delay.s**) inside that zip file.
    b. Create a new project and add the skeleton files to that project.
    c. Make sure to setup the project to connect to your board (follow instructions form **Module_02** if you forgot how to do that).
    d. Implement the function **control_user_led** in assembly.
        i. The function takes as input the led requested state (0 == OFF, 1 == ON) and the duration for holding the state.
        ii. The function returns void.
    e. Implement the function **delay** in assembly
        i. The function takes as input an integer value.
        ii. The function will decrement the value until it reaches 0
        iii. Then returns void.
    f. Call the "control_user_led" function from a while loop in main.
    g. For any C code, use only data types defined in the "stdint.h" file

    Hints:
    a. Implement delay in assembly first. Once it works, implement control_user_led function.

   b. Use your simple LED code that made use of the peripheral registers (**not** the bit-banding registers).
   c. It is ok to use a hard code value of the ODR (Output Data Register) address for GPIOA and store it into one of the CPU scratch registers.

3. Generate the map file for your program and provide details on:
   a. How much total ROM your program is occupying?
   b. How much total RAM your program is using?
   c. What part of your program is using the most ROM?
   d. What part of your program is using the most RAM?

4. **Bonus:** Anything that can be done to optimize the usage of ROM or RAM resources? Explain any options.

5. **Bonus:** Re-implement the **control_user_led** to use the bit-band region for accessing the Output Data Register (ODR) for GPIOA in order to toggle the LED ON/OFF. Hint: It is ok to use a hard code value of the ODR bit-band address for GPIOA and store it into one of the CPU scratch registers.

What to turn in and how:

- Check in all your homework in your repo under the folder "**assignment06**".
- Your folder should contain the following:
  - Turn in your source code files only (for example: main.c, …etc.) and any other files that you have authored.
  - Turn in answers to questions in markdown file format.
- Submit a link to your GitHub repo assignment:
  - Ex: "https://github.com/<account_id>/embsys100/assignment06"