EMBSYS100 - AU19
ASSIGNMENT 04

Goal

The goals for the assignment this week:

1. Practice the use of bit-banding region.
2. Examine the assembly code generated for a function with multiple parameters.
3. Go thru a design problem exercising key concepts we viewed so far (pointers, arrays, functions, design, API, testing…etc.)
4. Bonus: Apply usage of pointers and the understanding of Endianness.

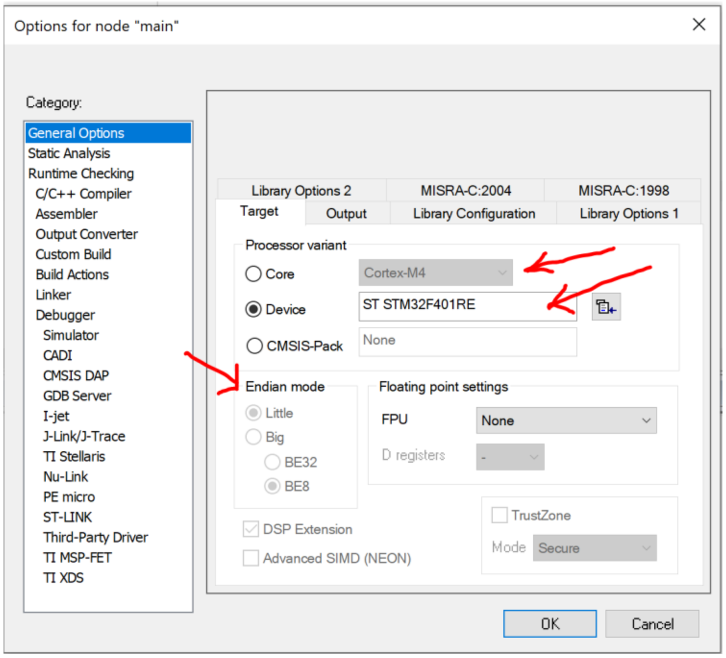
Problems:

1. Using bit-band region for peripherals:
    a. Convert the Blinking Led demo to use the corresponding bit-band address instead of the register address used in the peripheral region.
    b. What instructions does the compiler produce in assembly for the "writing" to the GPIO bit when using bit-band address?
    c. What were the instructions produced when writing to the GPIOx_ODR bit[5] directly?

2. Create a function with multiple arguments (5 arguments for example) and call that function from within another function. Trace thru the assembler and note:
    a. How does the **calling** function pass the values to the **called** function?
    b. What extra code did the compiler generate before calling the function with the multiple arguments?
    c. What extra code did the compiler generate inside the **called** function with the multiple list of arguments?
    d. Any other observations?

3. Following the queue data structure approach, design, implement and test a stack data structure:
    a. The following is the list of requirements:
        i. *The stack should have a predefined size*
        ii. *The stack supports "int" data types.*
        iii. *Provide a function to initialize the stack internals.*
        iv. *Provide a function to push an element onto the stack*
        v. *Provide a function to pop an element off the stack.*
        vi. *Provide a function that returns 1 if stack is empty.*
        vii. *Provide a function that returns 1 if stack is full.*
    b. Provide a list of the test cases and their implementation inside of main.c
    c. Separate the stack code from the rest of the test code (create stack.h & stack.c)

4. **<u>Bonus:</u>** Using the power of pointers and type casting, create a function that can determine if a computer is big-endian or little-endian. Test your function in the simulator and modify the Project Options (as shown in the figure below) against:
    a. Device STM32F401RE
    b. Cortex M4 (Little endian option)
    c. Cortex M4 (Big Endian option)



Processor variant

○ Core: Cortex-M4

● Device: ST STM32F401RE

○ CMSIS-Pack: None

Endian mode

● Little

○ Big

○ BE32

● BE8

Floating point settings

FPU: None

D registers: -

☑ DSP Extension

□ Advanced SIMD (NEON)

□ TrustZone

Mode: Secure

What to turn in and how:

- Check in all your homework in your repo under the folder "**assignment04**".
- Your folder should contain the following:
    - Turn in your source code files only (for example: main.c, stack.c, stack.h) 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/assignment04"