CS2240 Spring 2023
Programming Project #4 – The Green Square Game
Due: Tuesday, May 2, 2023, 11:59pm

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

First collect some assembly source code files you have written for previous labs. You will need your Lab08-Part1Random.s code from Lab #8 and your Lab11-Part1Delay.s code from Lab #11. To these two source code files, you will be adding two new assembly source code files you'll complete for this project, Project4-LED.s and Project4-button.s, and compiling them all together with the main C program Project4-GreenSquare.c. Your two new files will contain several functions to interface the green and red user LEDs and the blue user push button on the Discovery Board. Although there are 6 functions you will complete in these two source files, each function is quite simple and straightforward.

Download the C code file Project4-GreenSquare.c and the two template assembly source files Project4-LED.s and Project4-button.s. First, inspect the main C program for this project. At the beginning of the C code two setup functions are called, `setup_LEDs()` and `setup_button()`. These are functions you must complete in the template source files so that they store the proper bits in the memory-mapped configuration words for IO ports G and A to interface to the LEDs and the blue push button on the Discovery Board, as discussed in class.

Then the remainder of the main C program implements the logic for a rudimentary video game that exercises your reaction time, as shown in class. There are 16 possible positions for filled squares to be drawn on the graphics display screen, and squares are drawn one at a time at a sequence of randomly selected positions. At all but one of the positions the square will be filled red, but there is a hidden position where the square will be filled green, and that hidden position will occasionally drift around the screen. The game will start off at Level 1, with a fairly slow progression of squares. You can advance the level by pressing the blue pushbutton on the board only during the time the green square is showing. As the game level is raised the rate of progression of squares around the display gets somewhat faster. You win the game when you accurately time your pressing of the button while the green square is showing at Level 5. The LEDs on the board will flash a signal that confirms what color square was being shown when the button is pressed, and also with an alternating flash pattern when the game is won.

In the Project4-button.s template source file, the four required functions are described in comment lines. In addition to the setup function described above, complete the function named `button()`. This function should simply return the current state of the button, 0 for released and 1 for pressed, with no iterative wait loop. Then two other functions are needed, `wait_for_release()` and `wait_for_press()` . These must implement an iterative loop that repeats indefinitely until an IO port A load indicates the blue button has been *released* or *pressed*, respectively.

In the Project4-LED.s template source file, the two required functions are also described in comment lines. In addition to the setup function described above, complete the function named `setLEDs()`. This function should accept two unsigned integer arguments, which may either be the values 0 or 1 corresponding to an LED being *off* or *on*, respectively. The first argument controls the green LED, and the second argument controls the red LED.

So in summary, you need to complete two functions in the Project4-LED.s source file, `setupLEDs()` and `setLEDs()`, and four functions in the Project4-button.s file, `setup_button()`, `button()`, `wait_for_press()`, and `wait_for_release()`. The prototypes for these functions, and also your `random()` and `delay()` functions from previous labs, are at the top of the main C code, and show the number and types of the expected function arguments and return values, if any. Your `random()` function will let the C main program assign positions for drawing the squares, and your `delay()` function will let the C main program sequence the squares at the proper rate for each game level.

Run your code and verify that it provides the C code its needed interface to the LEDs and pushbutton so that the game is run properly. Upload your code to Canvas and a brief video clip of your game functioning. Make sure, however, that any debugging breakpoints you inserted for debugging your code have been removed in the version you submit.