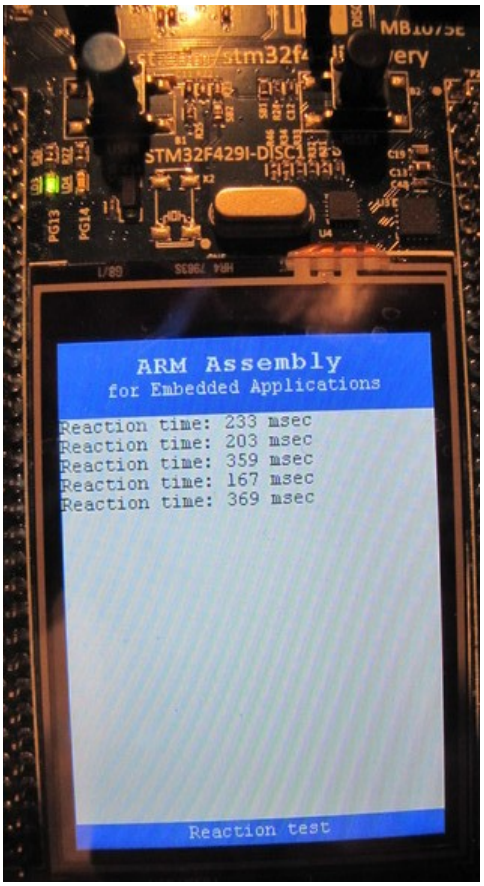
CS2240 Fall 2022
Programming Project #4 – Reaction Timer
Due: Monday, December 5, 2022, 11:59pm

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-random.s code from Lab #8 and your Lab11-delay.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. Your new files will contain several functions to interface the red and green 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-ReactionTimer.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 there is one primary infinite loop in the main program that runs a test of your manual reaction time each iteration. At the top of each loop the red user LED is turned on using a function `set_LEDs()` that you must complete in the template source file. Then your `random()` function is called to return a 32-bit word of random bits. A logical bitwise AND operation is performed to "mask off" all but the lower 12 bits, and force all the top 20 bits to be zero. This will leave a random number between 0 and 2^{12}-1, or 4095. Then 1000 is added to produce a random number between 1000 and 5095. Your `delay()` function is then called, with that random number as an argument, to cause the program execution to delay between 1 and 5.095 seconds. After this random time delay has expired, the user LEDs are changed so that the red LED turns off and the green LED turns on. This is the user's signal to press the blue push button on the board. The processor cycle counts are sampled in the main program before and after a call to your function `wait_for_press()` that you must complete. This function must simply continue in an iterative loop until loading from IO port A indicates the button has been *pressed*. The difference in cycle counts between the call and return of this function is a measurement of the user's reaction time between seeing the green LED turn on and pressing the blue button. This cycle count difference is divided by 180,000 to convert to milliseconds and reported on the board screen as the reaction time. To reset the timer at the bottom of the main program loop, a fixed delay of 50 msec is called, and then

your new function `wait_for_release()` is called. This is similar to your `wait_for_press()` function, but with an iterative loop that repeats until an IO port A load indicates the blue button has been *released*. Then the testing loop in the C program repeats for another random time delay and another reaction time test.

Note that just after the green LED is turned on in the main program, an additional function is called 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. This is used to test to make sure the user has not pressed the button at some point during the red LED time, and is truly waiting for the green LED to turn on before pressing the button.

So in summary, you need to complete two functions in the Project4-LED.s source file, `setup_LEDs()` and `set_LEDs()`, 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. The template assembly source files also contain comment lines with more details of what is expected in each of these functions.

Run your code and verify that it tests your reaction time. Your TA achieved a reaction time of only 139 milliseconds, and you may consider that the time to beat. Upload your code to Canvas and a brief video clip of your timer functioning. Make sure, however, that any debugging breakpoints you inserted for debugging your code have been removed in the version you submit.