Project 4: Interrupts

This project is due on **November 30, 2018** at **6 p.m.** and counts for 5% of your course grade. Late submissions are not accepted. If you have a conflict due to travel, interviews, etc., please plan accordingly and turn in your project early.

The code and other answers you submit must be entirely your own work, and you are bound by the Honor Code. You may consult with other students about the conceptualization of the project and the meaning of the questions, but you may not look at any part of someone else's solution or collaborate with others to develop a solution. You may consult published references, provided that you appropriately cite them (e.g., with program comments), as you would in an academic paper.

Solutions must be submitted electronically via **Moodle**, following the submission checklist below.

Introduction

In this project, you will write an Interrupt Service Routine (ISR) to accomplish similar tasks as in Project 2.

Objectives:

- Learn the value of interrupts and how they make your life easy
- See how interrupts can also make your life very, very difficult :)

Similar to Project 2, you will choose **ONE** of the two parts. The first option (Option 1) builds on the LED scrolling and push buttons. The second option (Option 2) uses the WS2812 LED strip used in Project 2.

Option 1. HELLO BUFFS

For this problem, you must use interrupts for both the timer and the push button. Read the whole description of this program before starting.
As in Project 2, you must make the HELLO BUFFS message scroll across the first four 7-segment displays. But, for this assignment, you need to use the **timer with interrupts** rather than a delay loop.

As an additional requirement, when the lower pushbutton (KEY1) is pressed, the message should scroll at a slower rate across the display. Each time the lower push button is pressed, scroll speed should decrease, but reach a minimum speed after 3 presses of the push button. So continued presses of KEY1 do not do anything. When the upper push button (KEY0) is pressed, the message scroll speed should speed-up. Similar to the case with KEY1, if you start at the default speed, three presses of KEY0 will speed-up the message to a maximum speed and no further.

Using the pushbuttons, you should be able to slow-down and speed-up the message scroll speed from the minimum to the maximum, and back down to the minimum. You need to decided how much faster and slower the message should scroll with each button push. At the fastest speed, you should still be able to clearly read the message, and at the slowest speed, the message should still scroll at a reasonable rate (you shouldn't have to wait for 10 seconds to see it completely scroll). In your push button ISR, consider if it is possible to see both push buttons having been pressed at the same time, and if so, what should you do about this case?

What to submit Include all of your .s files you create for this part (e.g. interrupt_example.s, exception_handler.s, interval_timer_ISR.s, pushbutton_ISR.s). You may build off the assembly interrupt example in the Monitor program (that we reviewed in class).

Option 2. LED strip

For this option, you will add push button functionality to the LED strips you implemented in Project 2. See the Project 2 description for how to setup the LED strip hardware.

You will need to handle button presses in an interrupt handler, but it's up to you if you want to implement the LED strip writing logic in an interrupt handler or not (hint: think carefully about if using the DE10-Lite's Interval Timer is a wise choice for timing 0.35 microseconds).

It's up to you what action the push buttons perform for your given pattern, but the buttons need to change the pattern in a noticeable way when pressed. For example, you could have the buttons determine the speed of your pattern, or you could have multiple patterns that can be switched when a button is pressed. Make sure to include a description (as a separate text file) of what behavior your pattern/buttons does.

What to submit Include all your .s files you create for this part, as well as a `description.txt` file that describes the behavior of your pattern/buttons.

Submission Checklist

You must do either Option 1 or Option 2, and submit the corresponding files in a tarball (`.tar.gz`) named
`project4.`*`your-identikey`*`.tar.gz`.
You can make a tarball with `$ tar czf project4-erwu1234.tar.gz ./file1.s ./file2.s ./file3.s`.

Upload this tarball to **Moodle**.