Assignment 5 - Due: Wednesday, April 18, 2018 10am (before class)

This is assignment consists of two major parts: writing assembly code and written questions (two problems from the text plus an additional question).

To hand in – the following 5 files (the first 4 are for the Program 1 part, the last one is for the questions). Note: you can use address_map_nios2.s and not need to hand it in (assume we have it):

- <lastname>_<firstname>_interrupt_buffs.s,
- <lastname>_<firstname>_exception_handler.s,
- <lastname>_<firstname>_interval_timer.s and
- <lastname>_<firstname>_pushbutton_ISR.s
- <lastname>_<firstname>_assignment5.pdf for the answers to the questions from the text and additional problem.

Program 1 – using interrupts.

For this problem, you must use interrupts for both the timer and the pushbutton. Read the whole description of this program before starting.

NOTE: EXAM 2 will be the same week as this is due – START EARLY!

As in Assignment 3, 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 pushbutton 1 is pressed, the message should scroll at a slower rate across the display. Each time pushbutton 1 is pressed, scroll speed should decrease, but reach a minimum speed after 3 presses of the pushbutton. So continued presses of pushbutton 1 do not do anything.

When pushbutton 2 is pressed, the message scroll speed should speed-up. Similar to the case with pushbutton 1, if you start at the default speed, three presses of pushbutton 2 will speed-up the message to a maximum speed and no further.

Using pushbuttons 1 and 2 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 pushbutton ISR, consider if it is possible to see both pushbuttons having been pressed at the same time, and if so, what should you do about this case?

Finally, since the message will scroll at 7 speeds, use the red LEDs to indicate the current speed. Use LEDs 7 through 1 for this. The initial, medium speed should be indicated by $LEDG_4$, the maximum speed should be indicated by LED_7 while the slowest speed is indicated by LED_1.

To do this assignment, (i) look at the interrupt example that can be created when you create a new project (selecting include sample code), (ii) look at the assignment 3 solutions posted to D2L if you struggled with assignment 3, (iii) leverage your code from assignment 3, as that's familiar to you. You can use the interrupt example sample code as the starting point, or your assignment 3 code. And use the *DE10-Lite Computer Manual* as a guide. You should have at exactly four files to hand in (all preprended with your name as described at the top of this document): **interrupt_buffs.s**, **exception_handler.s**, **interval_timer.s** and **pushbutton_ISR.s**. The interrupt_buffs.s code contains initialization code and the idle loop; it would be very similar to the interrupt_example.s code we saw in class.

When setting up your project for this assignment, on the memory setting screen there is an area for 'Linker Section Presets' – set this to Exceptions.

When using the interval timer, if the timer is already counting down, you can't just write a new count value to the timer. You need to first stop the timer, write the value, and then start the timer again. This means writing the correct bit mask to the timer's control register.

In your .data section, you may be mixing .byte and .word directives to allocate space for variables. Try to group the .byte and .word allocations together; and before the .word directive use ".align 4" on a line by itself. This ensures that your word variables are properly aligned on word boundaries.

In Figure 11 of the DE10-lite Manual, you'll notice code to set up the stack (this is done because there is no function calling _start, so this is where it gets initialized). The sample code has equally valid code:

```
_start:
/* set up stack pointer */
movia sp, 0x03FFFFFC /* put stack at top of SDRAM */
```

Problems from the text:

Please do problems: Q3-28 and Q3-29 (4th ed) (which are problems Q3-26 and Q3-27 in the 3rd ed, or Q3-21 and Q3-23 in the 2nd edition). For Q3-29, there is a typo in the text. It should read "the two-way, set-associative cache with *four sets*" instead of *four banks*.

Additional Problem:

Assume that a system has a two-level cache: The level 1 cache has a hit rate of 90% and the level 2 cache has a hit rate of 97%. The level 1 cache access time is 4 ns, the level 2 access time is 15 ns, and the main memory access time is 80 ns. What is the average memory access time?