I. Learner Objectives:

At the conclusion of this lab, participants should be able to:

- Handle timer interrupts
- Handle external interrupts
- Take a similar solution to a problem written in C and write a solution in MIPS 32 assembly
- Design and implement subroutines in MIPS32 assembly
- Interface with basic peripheral modules in MIPS32 assembly

II. Prerequisites:

Before reading this lab, participants should be able to:

- Summarize the unit related to Timers
- Explain the unit related to External Interrupts
- Outline the lab related to ports
- Summarize the unit related to registers

III. Keywords:

Interrupts

IV. Pre-lab:

None.

V. Background:

This lab will introduce you to the wonderful world of interrupts with the PIC32MX4. The idea behind the lab is to introduce you to basic low level design and programming with interrupts. You will be using a switch peripheral module LED peripheral module, on-board buttons, and on-board LEDs to construct a binary up-down counter. This problem is similar to the one you solved in Lab 3, with a few modifications.

In this lab you will use MPLAB and the C preprocessor to create a basic binary up-down counter. In order to use the C preprocessor and assembly you will need to create a project which uses the C compiler and a .S (uppercase S) assembly source file. The program should display the current value of the counter on the LEDs. You will use the on-board buttons of the Cerebot MX4cK to start, stop, and clear the counter. The current 4-bit result of the counter should be displayed to the on-board LEDs. The counter value should increment or decrement every one second. You must use a timer and timer interrupts to determine when a second has expired. The example provided here will help you get started with timers and interrupts. You must use external interrupt 1 (INT1) to grab the attention of the CPU and display the current value of the on-

boards LEDs to an external display device, which in this case is your peripheral LED module. The example provided here will help you get started with external interrupts. The external interrupt should only interrupt the processor on a rising edge. Refer to the Cerebot MX4cK Reference Manual document to determine the best ports to plug in your switches and peripheral LEDs.

You may include the following in your main.S file to configure SYSCLK and PBCLK.

```
// SYSCLK = 80 MHz (8 MHz Crystal/ FPLLIDIV * FPLLMUL / FPLLODIV)
// PBCLK = 40 MHz
// Primary Osc w/PLL (XT+,HS+,EC+PLL)
// WDT OFF
// Other options are don't care

#pragma config FNOSC  = PRIPLL // Required
#pragma config FPLLMUL  = MUL_20
#pragma config FPLLIDIV  = DIV_2
#pragma config FPBDIV  = DIV_2 // Divide SYSCLK by 2 ==> PBCLK
#pragma config FPLLODIV  = DIV_1
```

Recall the SYSCLK is the frequency at which the CPU will operate. The I/O ports operate based on SYSCLK. Peripherals such as timers operate based on the PBCLK.

Requirements:

- Pressing on-board BTN1 must start the up counter (counts 0 → 15)
- Pressing on-board BTN2 must start the down counter (counts 15 → 0)
- Pressing BTN1 and BTN2 together once must stop the counter
- Pressing BTN1 and BTN2 together twice must clear the counter (counter = 0)
- Flipping a switch, corresponding to INT1, from low to high forces an interrupt in order to display the current value of the counter to the peripheral LEDs

Hardware Required:

- 1 - Digilent Cerebot MX4cK Embedded Controller Board
- 1 – USB A -> micro B programming cable
- 1 – Switch module
- 1 – Peripheral LED module

Have fun, be creative, and ask your TA questions!!!

VII. Questions:

1. Explain how you were able to achieve one second increments/decrements with your assembly solution. As part of your answer provide the basic mathematical computation(s) and value(s) of register(s) required to acquire your one second interval.

2. Explain trade-offs between using a 16-bit versus a 32-bit timer to solve this problem.