Objectives:

1. Parallel port input and output interfacing.
2. LED and 7-Segment LED Displays and software drivers.
3. DIP switch inputs and software drivers.
4. RTI Interrupt operation.

Note: The base address of the arrays and all other required address information will be given in the prelab assignment.

Hardware Configuration:
Interface one Seven Segment Display units and an 8-bit bar LED unit provided by the lab instructor to the General Purpose I/O ports. This is the same configuration as Lab 09.

Interface one 8-bit DIP switches provided by the lab instructor to the General Purpose I/O ports. This is the same configuration as Lab 09.

Task1:

1. Short version: Repeat Lab 9, Task 3, part 1 using the BLINKRTIS.s code from Task 1 instead of a software delay loop. Toggle the 7-segment dp every time the interrupt down-counter is reloaded.

Long Version: Write an assembly program that will sequentially display the decimal numbers (0-9) and Hexadecimal numbers (A-F) on the 7-Segment Display Units and also on the 8-bit bar LED unit with a delay of 1.00s. The delay should be implemented using the internal RTI and an RTI interrupt.

2. Repeat Lab 9, Task 3, part 2 using the BLINKRTIC.c code from Lab 9, Task 1 instead of a software delay loop. Use the structure of BLINKRTIC.c to perform the operations (set the flag in the interrupt, update the 7-segment as state 0x80 and set state 0x40, update the LED as state 0x40). Toggle the 7-segment dp every time the interrupt down-counter is reloaded. (part of state 0x80)

Task2:

Based on Task1, modify the code to implement a Modulo-256 counter that is supposed to updates the count every 100ms. Display the entire count on the LEDs but only the most significant 4-bits in hex on the 7-segment display. Toggle the 7-segment dp every time there is an interrupt. Generate both assembly language and C code.

Task3:

1. Using the code from Task2, write additional assembly code that will read the values of the DIP switch every 100 msec. (a) When the LSB (bit 0) of the dip switch is off, hold the value of the Modulo-256 counter and displays. When the switch is on, let the Modulo-256 counter count and show the count on the displays. (b) When the MSB (bit 7) of the dip switch is on, clear the Modulo-counter. (c) When the bit 3 of dip switch is on, count down.

2. Develop the C code from Task2 to perform the operations just described. Notes: read the dip switch position and toggle the dp as state 0x80 and set state 0x40. In state 0x40, determine if the count value should be zero. If so, states are done (0x00), if not, set state 0x20. In state 0x20, determine if you are counting up or down. If up, set
state 0x10. If down set state 0x08. In state 0x10, increment the 8-bit internal register and set state 0x04. In state 0x08, decrement the 8-bit internal register and set state 0x04. In state 0x04, output the value of the internal register to the LED and 7-segment display.

The programming table:

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 3</th>
<th>Bit 0</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Zero the counter</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>0</td>
<td>Hold count</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Count Up</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Count Down</td>
</tr>
</tbody>
</table>

**Task 4:**

1. Write an assembly program that will read the values of DIP Switch1 every 2 sec. Then, use the least significant three bits of DIP switch1 as a command, the 4 most significant bits of DIP switch1 as one operand, and the previous value of DIP switch1 as a second operand. With the command and two operands, performs the tasks described in the prelab. The full result must be displayed on the LED bar and the lower four bits of the result on the seven segment displays.

2. Develop C software routine to perform the same operations. The software delay loop should be implemented using embedded assembly for precise timing calculations. Note: In C, use a switch-case statement to perform the operations.