**CSCE 611**

**Gautham Srinivasan (UIN: 927008557)**

**MP5 Design Document**

**Objective:**

The objective of machine problem 5 is to design FIFO and Round Robin scheduler (**option 2**) with correct handling of interrupts (**option 1**).

**Following are the files modified:**

Scheduler.C

Scheduler.H

Thread.C

Simple\_timer.C

Kernel.C

**First In First Out Scheduler (FIFO):**

FIFO scheduling algorithm schedules and executes the job which came first.

**Detailed implementation:**

**Scheduler.C**

1. **Scheduler() –**

Initialize the head and tail of ready to NULL as the queue is empty.

1. **Yield () –**

When this function is called, the current thread gives up the CPU. The head of the queue is moved to the next node and the first head node is deleted. Then the dispatch\_to function is called to execute the next thread.

**Option 1: Correct handling of interrupts**

Disable the interrupts if it is enabled at the beginning of the yield function and enable the interrupts if it is disabled at the end of yield function.

1. **Resume () –**

This function creates the new node in the queue list and adds them to the end of the queue if the queue is already present or adds a head node only.

**Option 1: Correct handling of interrupts**

Disable the interrupts if it is enabled at the beginning of the resume function and enable the interrupts if it is disabled at the end of resume function.

1. **Add () –**

This function resumes the thread.

1. **Terminate () –**

This function gets called when the thread exits. It cleans up by deleting the node in the queue and calls yield () to dispatch the next thread.

**Thread.C**

1. **Thread\_shutdown () –**

This function disables the interrupts and calls terminate and yield function from scheduler.C.

1. **Thread\_start () –**

This function enables the interrupts.

**Round Robin Scheduler (Option 2):**

Round robin scheduler is same as FIFO scheduler except we need to add end of quantum using a timer. Hence, round robin scheduler inherits FIFO scheduler and yield (), resume (), add () and terminate () functions are the same. In my implementation, I have added 50 ms quantum.

**Detailed implementation:**

**Simple\_timer.C**

1. **Handle\_interrupts () –**

This function is called when the timer expires and the CPU schedules the next thread. Hence, it calls resume () and yield () from the scheduler.C.

**Kernel.C**

I have uncommented \_USES\_SCHEDULER\_ macro to enable scheduling and when using round robin scheduler, I have commented out normal FIFO scheduler and add round robin scheduler with 50 ms quantum.