Design Idea:

Kernel-Level Thread Scheduling – MP5:

I Have implemented the following:

- FIFO Scheduler

- Terminating of threads

- Option 1: Interrupt management

- Option 2: Round-Robin Scheduling using End-of-Quantum Timer

FIFO Scheduler:

A pointer-based list with the front and tail pointers is used to implement the ready queue. The queue is initially empty with its front and tail pointer pointing to null. A variable thread_count is used to maintain the number of threads in the system.

The Scheduler provides the following functionality:

- add(thread) – Adds the thread to the end of the ready queue after the tail node. If the front and tail pointer are null, they now point to the new thread added to the queue.

- resume(thread) – Adds thread to the end of the queue using add(thread) functionality.

- yield() – Scheduler passes the control to the thread at the front of the ready queue. It throws and assertion error if the queue is empty. If the ready queue only contains one thread, the front and tail nodes are made to point to null.

- terminate(thread) – Terminates the thread by removing it from the ready queue and yields to the next thread in queue.

Termination of threads:

Threads are terminated by thread_shutdown api in the thread class. The threads may be terminated after task completion or on suiciding. To handle this, the thread is removed from the ready queue by the scheduler before deleting the thread.

Test Screenshots / Output – FIFO Scheduler with terminating threads:

As seen below, every thread completes 10 ticks in a burst before passing the CPU to the next thread in queue. Also, threads 0 and 1 run for bursts 0 to 9 each, while the threads 2 and 3 run infinitely, unless the operating system is switched off at bursts 51 and 50 respectively.

```
<bochs:1> c
Installed exception handler at ISR <0>
Allocating Memory Pool... done
Installed interrupt handler at IRQ <0>
Constructed Scheduler.
Hello World!
CREATING THREAD 1...
esp = <2098100>
done
DONE
CREATING THREAD 2...esp = <2099148>
done
DONE
CREATING THREAD 3...esp = <2100196>
done
DONE
CREATING THREAD 4...esp = <2101244>
done
DONE
Added Thread To Ready Queue.
Added Thread To Ready Queue.
Added Thread To Ready Queue.
STARTING THREAD 1 ...
Thread: 0
FUN 1 INVOKED!
FUN 1 IN BURST[0]
FUN 1: TICK [0]
FUN 1: TICK [1]
FUN 1: TICK [2]
FUN 1: TICK [3]
FUN 1: TICK [4]
FUN 1: TICK [5]
FUN 1: TICK [6]
FUN 1: TICK [7]
FUN 1: TICK [8]
FUN 1: TICK [9]
```

```
Added Thread To Ready Queue.
Thread: 1
FUN 2 INVOKED!
FUN 2 IN BURST[0]
FUN 2: TICK [0]
FUN 2: TICK [1]
FUN 2: TICK [2]
FUN 2: TICK [3]
FUN 2: TICK [4]
FUN 2: TICK [5]
FUN 2: TICK [6]
FUN 2: TICK [7]
FUN 2: TICK [8]
FUN 2: TICK [9]
Added Thread To Ready Queue.
Thread: 2
FUN 3 INVOKED!
FUN 3 IN BURST[0]
FUN 3: TICK [0]
FUN 3: TICK [1]
FUN 3: TICK [2]
FUN 3: TICK [3]
FUN 3: TICK [4]
FUN 3: TICK [5]
FUN 3: TICK [6]
FUN 3: TICK [7]
FUN 3: TICK [8]
FUN 3: TICK [9]
Added Thread To Ready Queue.
Thread: 3
FUN 4 INVOKED!
FUN 4 IN BURST[0]
FUN 4: TICK [0]
FUN 4: TICK [1]
```

```
Added Thread To Ready Queue.
FUN 1 IN BURST[9]
FUN 1: TICK [0]
FUN 1: TICK [1]
FUN 1: TICK [2]
FUN 1: TICK [3]
FUN 1: TICK [4]
FUN 1: TICK [5]
FUN 1: TICK [6]
FUN 1: TICK [7]
FUN 1: TICK [8]
FUN 1: TICK [9]
Added Thread To Ready Queue.
FUN 2 IN BURST[9]
FUN 2: TICK [0]
FUN 2: TICK [1]
FUN 2: TICK [2]
FUN 2: TICK [3]
FUN 2: TICK [4]
FUN 2: TICK [5]
FUN 2: TICK [6]
FUN 2: TICK [7]
FUN 2: TICK [8]
FUN 2: TICK [9]
Added Thread To Ready Queue.
FUN 3 IN BURST[9]
FUN 3: TICK [0]
FUN 3: TICK [1]
FUN 3: TICK [2]
FUN 3: TICK [3]
FUN 3: TICK [4]
FUN 3: TICK [5]
FUN 3: TICK [6]
FUN 3: TICK [7]
FUN 3: TICK [8]
FUN 3: TICK [9]
Added Thread To Ready Queue.
FUN 4 IN BURST[9]
FUN 4: TICK [0]
FUN 4: TICK [1]
FUN 4: TICK [2]
FUN 4: TICK [3]
FUN 4: TICK [4]
FUN 4: TICK [5]
FUN 4: TICK [6]
FUN 4: TICK [7]
```

```
Added Thread To Ready Queue.
FUN 4 IN BURST[9]
FUN 4: TICK [0]
FUN 4: TICK [1]
FUN 4: TICK [2]
FUN 4: TICK [3]
FUN 4: TICK [4]
FUN 4: TICK [5]
FUN 4: TICK [6]
FUN 4: TICK [7]
FUN 4: TICK [8]
FUN 4: TICK [9]
Added Thread To Ready Queue.
FUN 3 IN BURST[10]
FUN 3: TICK [0]
FUN 3: TICK [1]
FUN 3: TICK [2]
FUN 3: TICK [3]
FUN 3: TICK [4]
FUN 3: TICK [5]
FUN 3: TICK [6]
FUN 3: TICK [7]
FUN 3: TICK [8]
FUN 3: TICK [9]
Added Thread To Ready Queue.
FUN 4 IN BURST[10]
FUN 4: TICK [0]
FUN 4: TICK [1]
FUN 4: TICK [2]
FUN 4: TICK [3]
FUN 4: TICK [4]
FUN 4: TICK [5]
FUN 4: TICK [6]
FUN 4: TICK [7]
FUN 4: TICK [8]
FUN 4: TICK [9]
Added Thread To Ready Queue.
FUN 3 IN BURST[11]
FUN 3: TICK [0]
FUN 3: TICK [1]
FUN 3: TICK [2]
FUN 3: TICK [3]
FUN 3: TICK [4]
FUN 3: TICK [5]
FUN 3: TICK [6]
```

Option 1 - Interrupt management:

- Interrupt is enabled on the starting a thread. This is also written to the EFLAGS while populating the thread stack by checking if the interrupt is enabled. (0=interrupt disabled, 1=interrupt enabled).

- Further, to ensure mutual exclusion, the interrupt is disabled and enabled before and after critical sections of manipulating the ready queue.

Test Screenshots / Output:

The below log shows clearly that the interrupt is enabled at thread start and disabled in the critical section. Once the critical section is over, the thread is enabled again.


```
Initial Interrupt State: 1
Disabled Interrupts Before Critical Section.
Enabled Interrupts after critical section.
Added Thread To Ready Queue.
```

Option 2 – Round-Robin Scheduling using End-of-Quantum Timer:

- End-of-Quantum Timer (EOQTimer) is implemented as a derived class of the SimpleTimer class. Upon completion of the quantum time (50ms / 5 clock ticks), the scheduler adds the current thread to the tail of the ready queue using the resume function. It also, yields the control from the current thread to the next thread at the front of the queue.

- When dispatching an interrupt, to let the interrupt controller (PIC) know that the interrupt has been handled, an EOI signal is sent and then the interrupt is handled by yielding to the next thread in ready queue.

- To implement Round Robin scheduling, the kernel is updated to use the EOQTimer instead of the SimpleTimer. Also, interrupt is enabled after the start of the first thread, instead of at the start of the operating system.

- Further the makefile is updated to compile the new EOQTimer files created.

Test Screenshots / Output:

As seen in the logs below:

- Every thread runs for the set quantum, before yielding to the next thread. Hence, instead of completing a burst (10 ticks) before giving up the CPU, they perform different number of ticks in each quantum. Thread 0, 1, 2 and 3 perform 3, 4, 2, and 3 ticks of burst 0 in their first time-quantum and then give up the CPU.

- In the next time quantum, these threads resume from where they left off, i.e., ticks 4, 5, 3, and 4 of bursts 0 respectively.

- Threads 0 and 1 terminate after completing 10 bursts at different time-quantums.

- Threads 2 and 3 are non-terminating threads and continue until the system is switched off.

- The non-terminating threads have completed different amounts of bursts, when the system was turned off. Specifically, threads 2 and 3 are in bursts 34 and 37 respectively.

```
Next at t=0
(0) [0x0000fffffff0] f000:fff0 (unk. ctxt): jmpf 0xf000:e05b              ; ea5be000f0
<bochs:1> c
Installed exception handler at ISR <0>
Allocating Memory Pool... done
SimpleTimer constructed
EOQTimer constructed/derived from  SimpleTimer
Installed interrupt handler at IRQ <0>
Constructed Scheduler.
Hello World!
CREATING THREAD 1...
esp = <2098100>
done
DONE
CREATING THREAD 2...esp = <2099148>
done
DONE
CREATING THREAD 3...esp = <2100196>
done
DONE
CREATING THREAD 4...esp = <2101244>
done
DONE
Initial Interrupt State: 0
Added Thread To Ready Queue.
Initial Interrupt State: 0
Added Thread To Ready Queue.
Initial Interrupt State: 0
Added Thread To Ready Queue.
STARTING THREAD 1 ...
Thread: 0
FUN 1 INVOKED!
FUN 1 IN BURST[0]
FUN 1: TICK [0]
FUN 1: TICK [1]
FUN 1: TICK [2]
FUN 1: TICK [3]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
Thread: 1
FUN 2 INVOKED!
FUN 2 IN BURST[0]
FUN 2: TICK [0]
FUN 2: TICK [1]
FUN 2: TICK [2]
FUN 2: TICK [3]
FUN 2: TICK [4]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
Thread: 2
FUN 3 INVOKED!
FUN 3 IN BURST[0]
FUN 3: TICK [0]
FUN 3: TICK [1]
FUN 3: TICK [2]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
Thread: 3
FUN 4 INVOKED!
FUN 4 IN BURST[0]
FUN 4: TICK [0]
FUN 4: TICK [1]
FUN 4: TICK [2]
FUN 4: TICK [3]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 1: TICK [4]
FUN 1: TICK [5]
FUN 1: TICK [6]
FUN 1: TICK [7]
FUN 1: TICK [8]
FUN 1: TICK [9]
FUN 1 IN BURST[1]
FUN 1: TICK [0]
FUN 1: TICK [
One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 2: TICK [5]
FUN 2: TICK [6]
FUN 2: TICK [7]
FUN 2: TICK [8]
```

```
FUN 2: TICK [9]
FUN 2 IN BURST[1]
FUN 2: TICK
One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 3: TICK [3]
FUN 3: TICK [4]
FUN 3: TICK [5]
FUN 3: TICK [6]
FUN 3: TICK [7]
FUN 3: TICK [8]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 4: TICK [4]
FUN 4: TICK [5]
FUN 4: TICK [6]
FUN 4: TICK [7]
FUN 4: TICK [8]
FUN 4: TICK [9]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
1]
FUN 1: TICK [2]
FUN 1: TICK [3]
FUN 1: TICK [4]
FUN 1: TICK [5]
FUN 1: TICK [6]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
 [0]
FUN 2: TICK [1]
FUN 2: TICK [2]
FUN 2: TICK [3]
FUN 2: TICK [4]
FUN 2: TICK [5]
FUN 2: TICK [6]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 3: TICK [9]
FUN 3 IN BURST[1]
FUN 3: TICK [0]
FUN 3: TICK [1]
FUN 3: TICK [2]
FUN 3: TICK [3]
FUN 3: TICK [4]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 4 IN BURST[1]
FUN 4: TICK [0]
FUN 4: TICK [1]
FUN 4: TICK [2]
FUN 4: TICK [3]
FUN 4: TICK [4]
FUN 4: TICK [5]
FUN 4: TICK [6]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 1: TICK [7]
FUN 1: TICK [8]
FUN 1: TICK [9]
FUN 1 IN BURST[2]
FUN 1: TICK [0]
FUN 1: TICK [1]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 2: TICK [7]
FUN 2: TICK [8]
```

```
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 1: TICK [8]
FUN 1: TICK [9]
FUN 1 IN BURST[9]
FUN 1: TICK [0]
FUN 1: TICK [1]
FUN 1: TICK [2]
FUN 1: TICK [3]
FUN 1: TICK [4]
FUN 1: TICK [5]
FUN 1: TICK [6]
FUN 1: TICK [7]
FUN 1: TICK [8]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 2: TICK [6]
FUN 2: TICK [7]
FUN 2: TICK [8]
FUN 2: TICK [9]
Disabled Interrupts Before Critical Section.
Enabled Interrupts After Critical Section.

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 3: TICK [5]
FUN 3: TICK [6]
FUN 3: TICK [7]
FUN 3: TICK [8]
FUN 3: TICK [9]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 4: TICK [8]
FUN 4: TICK [9]
FUN 4 IN BURST[10]
FUN 4: TICK [0]
FUN 4: TICK [1]
FUN 4: TICK [2]
FUN 4: TICK [3]
FUN 4: TICK [4]
FUN 4: TICK [5]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 1: TICK [9]
Disabled Interrupts Before Critical Section.
Enabled Interrupts After Critical Section.
Thread Terminated.
Yielding To Next Thread In Ready Queue.

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 3 IN BURST[9]
FUN 3: TICK [0]
FUN 3: TICK [1]
FUN 3: TICK [2]
FUN 3: TICK [3]
FUN 3: TICK [4]
FUN 3: TICK [5]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 4: TICK [6]
FUN 4: TICK [7]
FUN 4: TICK [8]
FUN 4: TICK [9]
FUN 4 IN BURST[11]
FUN 4: TICK [0]
FUN 4: TICK [1]
FUN 4: TICK [2]
FUN 4: TICK [3]
FUN 4: TICK [4]
FUN 4
One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
Thread Terminated.
Yielding To Next Thread In Ready Queue.
FUN 3: TICK [6]
```

```
One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 3: TICK [4]
FUN 3: TICK [5]
FUN 3: TICK [6]
FUN 3: TICK [7]
FUN 3: TICK [8]
FUN 3: TICK [9]
FUN
One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 4: TICK [3]
FUN 4: TICK [4]
FUN 4: TICK [5]
FUN 4: TICK [6]
FUN 4: TICK [7]
FUN 4: TICK [8]
FUN 4: TICK [9]
FUN 4 IN BURST[36]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
3 IN BURST[33]
FUN 3: TICK [0]
FUN 3: TICK [1]
FUN 3: TICK [2]
FUN 3: TICK [3]
FUN 3: TICK [4]
FUN 3: TICK [5]
FUN 3: TICK [6]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 4: TICK [0]
FUN 4: TICK [1]
FUN 4: TICK [2]
FUN 4: TICK [3]
FUN 4: TICK [4]
FUN 4: TICK [5]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 3: TICK [7]
FUN 3: TICK [8]
FUN 3: TICK [9]
FUN 3 IN BURST[34]
FUN 3: TICK [0]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 4: TICK [6]
FUN 4: TICK [7]
FUN 4: TICK [8]
FUN 4: TICK [9]
FUN 4 IN BURST[37]
FUN 4: TICK [0]
FUN 4: TICK [1]
FUN 4: TICK [2]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
FUN 3: TICK [1]
FUN 3: TICK [2]
FUN 3: TICK [3]
FUN 3: TICK [4]
FUN 3: TICK [5]

One time quantum has passed, yielding to next thread
Initial Interrupt State: 0
Added Thread To Ready Queue.
Yielding To Next Thread In Ready Queue.
====================================================================
Bochs is exiting with the following message:
[XGUI  ] POWER button turned off.
====================================================================
(0).[982892000] [0x000000100203] 0008:0000000000100203 (unk. ctxt): lea eax, ds:[edx+1]        ;
 8d4201
csce410@csce410-VirtualBox:~/Documents/OS/SnehaSingh_CSCE611/mp5csce410@csce410-VirtualBox:~/Do
```

Documents changed:

I have changed the following files:

- **eoq_timer.C (new file)** – as per instructions in Option 2
- **eoq_timer.H (new file)** – as per instructions in Option 2
- **interrupts.C** – as per instructions in Option 2
- **kernel.C** – as per instructions in the handout for the compulsory parts, options 1 & 2
- **makefile** – updated to compile eoq_timer
- **scheduler.C** – as per instructions in the handout for the compulsory parts & option 1
- **scheduler.H** – as per instructions in the handout for the compulsory parts & option 1
- **thread.C** – as per instructions in the handout for the compulsory parts, options 1 & 2