OS Project 2 : Part 1 : Submission

Name : Pradeep Kumar

Group Member : Vibhor Gupta

Question 1: What happens when a thread is terminated? Take a look at ThreadFinish in the file Thread.k. How is the thread put to sleep with no hope of ever being awakened?

Solution: Thread termination happens in the function ThreadFinish () since all the operations have been completed in this thread and the final think for this thread that is left is to clean up and reclaim the Thread object. This method is called but However, since the thread is already running we cannot reclaim the thread object, so we just mark the thread and add it to the threadsToBeDestroyed list and then we put the thread to sleep and the thread get's blocked with no hope of ever being awakened.

```
threadsToBeDestroyed.AddToEnd (currentThread)

currentThread.Sleep ()

-- Execution will never reach the next instruction

endFunction
```

When the next thread starts the clean-up happens in the method run and we check the threadsToBeDestroyed list remove the thread objects from the list in a while loop and set the status of the thread as unused and we reclaim the thread object. So basically since no function wakes up any thread from **threadsToBeDestroyed** queue hence the thread goes to indeterminate sleep.

```
From Run() method

while ! threadsToBeDestroyed.IsEmpty ()

  th = threadsToBeDestroyed.Remove()

  th.status = UNUSED
```

Question 2: In TimerInterruptHandler, there is a call to Yield. Why is this there? Try commenting this statement? What happens? Make sure you understand how Yield works here.

This routine is called when a timer interrupt occurs. As soon as this function is invoked the interrupts are DISABLED. Upon return from this function the execution returns to the interrupted process, which necessarily had interrupts ENABLED.

The method yield actually modifies the status of the current thread to ready and then it begins running next thread which is present in the ready list (if there is any) and also it adds the current thread to ready list at the end.

This method should only be called on the current thread. The current thread yields the processor to other threads by executing currentThread.Yield (). This method may be invoked with or without interrupts enabled. Upon return, the interrupts will be in the same state; however. since other threads are given a chance to run and they may allow interrupts, interrupts handlers may have been invoked before this method returns.

If we comment out the statement to invoke yield() it stops time slicing and Threads will then execute until they call "Yield" explicitly, or until they call "Sleep.

The output from script :

```
pradeep@pradeep-VirtualBox:~/Project2_part1/Proj2 done$ script output_project2_part1
Script started, output log file is 'output_project2_part1'.
pradeep@pradeep-VirtualBox:~/Project2_part1/Proj2 done$ make
kpl System -unsafe
asm System.s
kpl List -unsafe
asm List.s
kpl Thread -unsafe
asm Thread.s
asm Switch.s
kpl Synch
asm Synch.s
kpl Main -unsafe
asm Main.s
asm Runtime.s
lddd System.o List.o Thread.o Switch.o Synch.o Main.o Runtime.o -o os
kpl Game -unsafe
asm Game.s
lddd System.o List.o Thread.o Switch.o Synch.o Runtime.o Game.o -o game
pradeep@pradeep-VirtualBox:~/Project2_part1/Proj2 done$ blitz -g os
Beginning execution..
====================  KPL PROGRAM STARTING  ====================
Example Thread-based Programs...
Initializing Thread Scheduler...


-- You should see 70 lines, each consecutively numbered. --

LockTester-A = 1
LockTester-A = 2
LockTester-B = 3
LockTester-C = 4
LockTester-D = 5
```

LockTester-E = 6
LockTester-A = 7
LockTester-F = 8
LockTester-G = 9
LockTester-C = 10
LockTester-B = 11
LockTester-D = 12
LockTester-A = 13
LockTester-F = 14
LockTester-E = 15
LockTester-G = 16
LockTester-C = 17
LockTester-D = 18
LockTester-A = 19
LockTester-F = 20
LockTester-B = 21
LockTester-G = 22
LockTester-E = 23
LockTester-C = 24
LockTester-D = 25
LockTester-A = 26
LockTester-F = 27
LockTester-G = 28
LockTester-B = 29
LockTester-C = 30
LockTester-E = 31
LockTester-D = 32
LockTester-A = 33
LockTester-F = 34
LockTester-G = 35
LockTester-C = 36
LockTester-B = 37
LockTester-D = 38
LockTester-A = 39
LockTester-F = 40
LockTester-E = 41
LockTester-G = 42
LockTester-C = 43
LockTester-B = 44
LockTester-D = 45
LockTester-F = 46
LockTester-A = 47
LockTester-G = 48
LockTester-E = 49
LockTester-C = 50
LockTester-D = 51
LockTester-B = 52
LockTester-F = 53
LockTester-A = 54
LockTester-G = 55
LockTester-C = 56
LockTester-D = 57
LockTester-E = 58

```
LockTester-B = 59
LockTester-F = 60
LockTester-G = 61
LockTester-D = 62
LockTester-C = 63
LockTester-E = 64
LockTester-F = 65
LockTester-G = 66
LockTester-B = 67
LockTester-E = 68
LockTester-B = 69
LockTester-E = 70

*****  A 'wait' instruction was executed and no more interrupts are scheduled... halting emulation!  *****

Done!  The next instruction to execute will be:
000EC8: 09000000       ret
Number of Disk Reads    = 0
Number of Disk Writes   = 0
Instructions Executed   = 424451
Time Spent Sleeping     = 0
   Total Elapsed Time  = 424451
pradeep@pradeep-VirtualBox:~/Project2_part1/Proj2 done$ exit
exit
Script done.
pradeep@pradeep-VirtualBox:~/Project2_part1/Proj2 done$
```