# **BCIT**

**Comp 4735 Operating Systems**

**Instructor: Mirela Gutica**

**Fall 2015**

Mark: \_\_\_\_\_\_\_\_ 100

Assignment 1

Note: To receive any credit whatsoever, your answers must be legible and readily readable in the judgement of the grader. Add brief explanatory comments as necessary to make sure your answers are clear and unambiguous to the grader. **When you solve a problem, show all the steps, similar with the examples in the lectures**. Just the answer will not give you credits for a problem. **It is required to have a professional layout for the assignment**.

The assignment should be handed-in (on D2L) no later than **11:30pm,** **Tuesday, October 13, 2015. No late assignments will be accepted.**

If not specified, a question weights **10 marks**.

**Answer to questions:**

(20p)

1. Consider the Intel i7 processor (64 and IA-32 architectures). Use latest documentation for this processor to answer the following questions:
   1. What data types are supported by the Intel processors? Enumerate.
   2. State and describe the stack manipulation instructions. State the role of each instruction.
   3. The CALL instruction (used to call a function/procedure) has two flavours: *near call* and *far call*. What is the difference between the two?
   4. Besides the method of passing parameters to the stack, the documentation indicates two other methods. Give a short description for each.
   5. Give descriptions for interrupts and exceptions underlying the difference between them.
   6. What exception classes exist for this processor?
      1. Give a short description for each.
      2. Explain if the OS terminates the process or if the problem is correctable.

Hint: Find the answers in the technical documentation (e.g., <http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf> ).

1. Give all four situations and reasons for a process to be moved to the state Ready/ Suspend. For each case, give an example. Refer to the figure 3.9b from the textbook – Process State Transition Diagram with Suspended States.

(20p)

1. Consider an operating system that executes within user processes. Consider (1) a procedure call that passes in two arguments and returns a result, and (2) a system call that passes in one argument of type char and displays the character (e.g., “E”) on the screen. Both the procedure and the system call are executed in process P1, and each take a little less than a time slice. Consider that after P1, process P2 is in the ready queue. After execution, P1 will join the ready queue. No other processes are in the ready queue.

Using diagrams and explanations answer the following questions:

1. Enumerate the steps performed in the case of the procedure call. Assume that the procedure is called during the first half of a time slice and returns in the same time slice.
2. Enumerate the steps performed in the case of the procedure call. Assume that the procedure is called during the second half of a time slice and returns in the next time slice assigned to P1.
3. Enumerate the steps performed in the case of the system call. Consider that the system call is called during the second half of a time slice and finishes in the next time slice assigned to P1.
4. Which is more expensive: the procedure call or the system call if a) return is in the same time slice b) return is in a different time slice? Why? Elaborate.
5. Disuses what an operating system should do when a new process is created and when a process terminates.
6. In many modern operating systems, when an interrupt or a system call in a process transfers control to the operating system, a kernel stack area separate from the process’s stack area is created. Why? Give at least two reasons. Elaborate.
7. Contrast and compare ULT and KLT. What solution is better and in what circumstances?
8. Web servers are multithreaded environments. Explain the consequences of a Web server that is single-threaded.
9. Consider an operating system that has only processes and doesn’t have kernel-level threads implemented. What advantages and disadvantages do you see in comparison with an operating system that has processes and kernel-level threads implemented?