CHAPTER 1

Question 1: **What are the two main functions of an operating system?**

An operating system provides the users with an extended (i.e., virtual) machine, and it must manage the I/O devices and other system resources.

Question: **What is multiprogramming?**

Multiprogramming is the rapid switching of the CPU between multiple processes in memory. It is commonly used to keep the CPU busy while one or more processes are doing I/O. It also allows two or more people to use the same computer simultaneously. Furthermore, it permits a single user to start up multiple, independent processed at the same time.

Question**: What is spooling? Do you think that advanced personal computers will have spooling as a standard feature in the future?**

Input spooling is the technique of reading jobs, onto the disk, so that when the currently executing processed are finished, there will be work waiting for the CPU. Output spooling consists of first copying printable files to disk before printing them, rather thatn printing directly as the output is generated. Input spooling on a personal computer is not very likely since the jobs are normally created on the disk, but output spooling is.

Question 5: **On early computers, every byte of data read or written was handled by the CPU (i.e., there was no DMA). What implication does it have on multiprogramming?**

It makes multiprogramming less favorable since it is no longer the case that when one process does I/O the CPU is completely free to work on other processes.

Question 12: **Which of the following instructions should be allowed only in kernel mode?**

(a) Disable all interrupts.

(b) Read the time-of-day clock.

(c) Set the time-of-day clock.

(d) Change the memory map.

Choices (a), (c) and (d) should be restricted to kernel mode.

Questions 17: **What is a trap instruction? Explain its use in operating systems.**

A trap instruction switches the execution mode of a CPU from the user mode to the kernel mode. This instruction allows a user program to invoke functions in the operating system kernel.

Question 28. **To a programmer, a system call looks like any other call to a library procedure. Is it important that a programmer know which library procedures result in system calls? Under what circumstances and why?**

As far as program logic is concerned, it does not matter whether a call to a library procedure results in a system call. But if performance is an issue, if a task can be accomplished without a system call the program will run faster. Every system call involves overhead time in switching from the user context to the kernel context. Furthermore, on a multiuser system the operating system may schedule another process to run when a system call completes, further slowing the progress in real time of a calling process.