CMR INSTITUTE OF TECHNOLOGY

SCHEME AND SOLUTION

Internal Assessment Test I – March 2017

Sub:	OPERATING SYSTEMS					**Code:**10EC65		
Date:	30 / 03 /2017	Duration:	90 mins	Max Marks:	50	**Sem:**	VI A & B	**Branch:ECE**

Note: Answer any five questions:

1.	Mention the goals and operations of an operating system. Discuss kernel based and microkernel based operating systems with neat diagrams.	10M

Mentioning 3 Goals and 3 operations- 4M
Explanation about kernel based operating systems with diagrams- 3M
Explanation about microkernel based operating systems with diagrams-3M

- Fundamental goals of an operating system
  - Efficient use of computer resources
  - User convenience
  - Noninterference in the activities of its users

- Principal functions of OS:
  - Program management
  - Resource management
  - Security and protection

Kernel-Based Operating Systems

- Historical motivations for kernel-based OS structure were OS portability and convenience in design and coding of non kernel routines .*Mechanisms* implemented in kernel, *policies* outside and Kernel-based OSs have poor extensibility

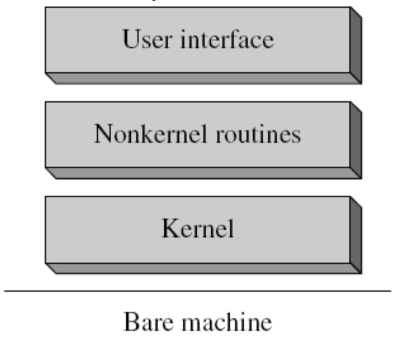

Figure 4.6 Structure of a kernel-based OS.

- *Dynamically loadable kernel modules*
  - Kernel designed as set of modules
    - Modules interact through interfaces
  - *Base kernel* loaded when system is booted
    - Other modules loaded when needed
      - Conserves memory
  - Used to implement device drivers and new system calls
- *User-level device drivers*
  - Ease of development, debugging, deployment and robustness
  - Performance is ensured through HW and SW means

Microkernel-Based Operating Systems

- The *microkernel* was developed in the early 1990s to overcome the problems concerning portability, extensibility, and reliability of kernels

- A microkernel is an essential core of OS code
  - Contains only a subset of the mechanisms typically included in a kernel
  - Supports only a small number of system calls, which are heavily tested and used
  - Less essential code exists outside the kernel
- Microkernel does not include scheduler and memory handler
- They execute as servers

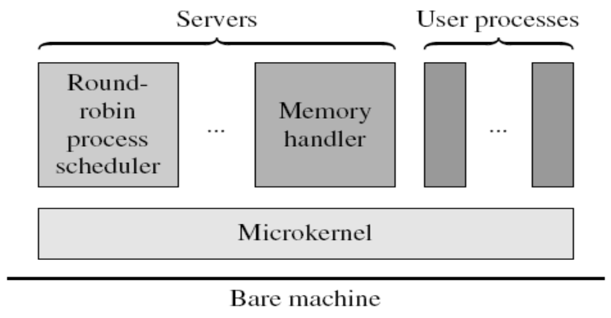

Figure 4.7 Structure of microkernel-based operating systems.

2. (a) Briefly explain the 3 problems faced in the layered design of operating systems. 4M

Explanation about the 3 problems- 3M

- Problems:
  1.System operation slowed down by layered structure
  2 Difficulties in developing a layered design
    - Problem: ordering of layers that require each other's services
      - Often solved by splitting a layer into two and putting other layers between the two halves
  3 Stratification of OS functionalities
    - Complex design
    - Loss of execution efficiency
    - Poor extensibility

(b) Explain the various resource allocation strategies of an operating system and illustrate using a resource table for I/O devices. 6M

Mentioning 3 stratergies-2M
Example of resource allocation table-2M
Explanation of strtergies-6M

- Resource allocations and deallocations can be done with a resource table
  - Entry: name, address and status of a resource unit
  - Constructed by the boot procedure, maintained during operation

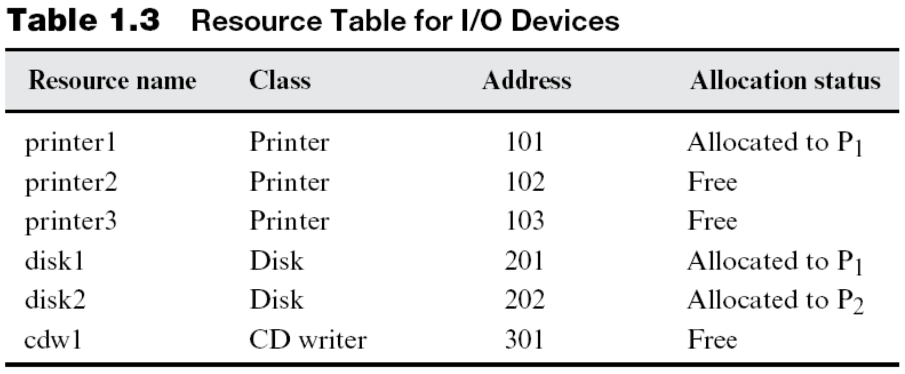

Table 1.3 Resource Table for I/O Devices

Resource name	Class	Address	Allocation status
printer1	Printer	101	Allocated to P_1
printer2	Printer	102	Free
printer3	Printer	103	Free
disk1	Disk	201	Allocated to P_1
disk2	Disk	202	Allocated to P_2
cdw1	CD writer	301	Free

- Popular resource allocation strategies:
  - *Resource partitioning*
    - OS decides *a priori* what resources to allocate to each user program; divides system resources into *partitions*
      - A resource partition is a collection of resources
    - Resource table contains entries for partitions
    - Simple to implement, but lacks flexibility
  - *Pool-based*
    - OS allocates resources from a pool of resources
      - Consults table and allocates the resource if it is free
    - Less overhead of allocating and deallocating resources
    - Achieves more efficient use of resources
- A *virtual resource* is a fictitious resource
  - Abstract view of a resource taken by a program
  - Supported by OS through use of a real resource
  - Same real resource may support several virtual ones
  - Started with the use of virtual devices
    - E.g., a print server
- Provides effect of having more resources
- Most OSs provide *virtual memory*
  - May execute a program bigger than size of RAM
- Some OSs create *virtual machines*
  - Each virtual machine can be allocated to a user

3. Explain the structure of virtual machine operating system VM/370, also explain its diverse purposes, and the arrangements for virual machines without VM OS. 10M

Explanation of VM OS with diagram-4M(2+2)
Minimum 3 Diverse purposes-2M
VMs are also used without a VM OS-4M

3M

- Different classes of users need different kinds of user service
- Virtual machine operating system (VM OS) creates several virtual machines
  - A virtual machine (VM) is a virtual resource
  - Each VM is allocated to one user, who can use any OS
    - Guest OSs run on each VM
- VM OS runs in the real machine
  - schedules between guest OSs
- Distinction between privileged and user modes of CPU causes some difficulties in use of a VM OS

Example: Structure of VM/370

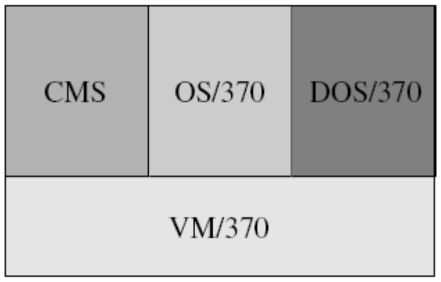

Figure 4.5 Virtual machine operating system VM/370.

- Virtualization: mapping interfaces and resources of a VM into interfaces and resources of host machine
  - Full virtualization may weaken security
  - Paravirtualization replaces a nonvirtualizable instruction by easily virtualized instructions
    - Code of a guest OS is modified to avoid use of nonvirtualizable instructions, done by:
      - Porting guest OS to operate under VM OS
      - Or, using dynamic binary translation for kernel of a guest OS
- VMs are employed for diverse purposes:
  - Workload consolidation
  - To provide security and reliability for applications that use the same host and the same OS
  - To test a modified OS on a server concurrently with production runs of that OS
  - To provide disaster management capabilities
    - A VM is transferred from a server that has to shutdown to another server available on the network

- VMs are also used without a VM OS:
  - Virtual Machine Monitor (VMM)
    - Also called a hypervisor
    - E.g., VMware and XEN
  - Programming Language Virtual Machines
    - Pascal in the 70s
      - Substantial performance penalty
    - Java
      - Java virtual machine (JVM) for security and reliability
      - Performance penalty can be offset by implementing JVM in hardware

4. Explain the architectural support and techniques of multiprogramming systems with appropriate examples, diagrams, and necessary timing diagrams. 10M

Explaning 3 features of architectural support-3M
Explaning 2 techniques of multiprogramming-3M
Diagram depicting operation -2M
Timing diagram-2M

Multiprogramming Systems
- Provide efficient resource utilization in a noninteractive environment
- Uses DMA mode of I/O 2M
  - Can perform I/O operations of some program(s) while using the CPU to execute some other program
    - Makes efficient use of both the CPU and I/O devices
- Turnaround time of a program is the appropriate measure of user service in these systems

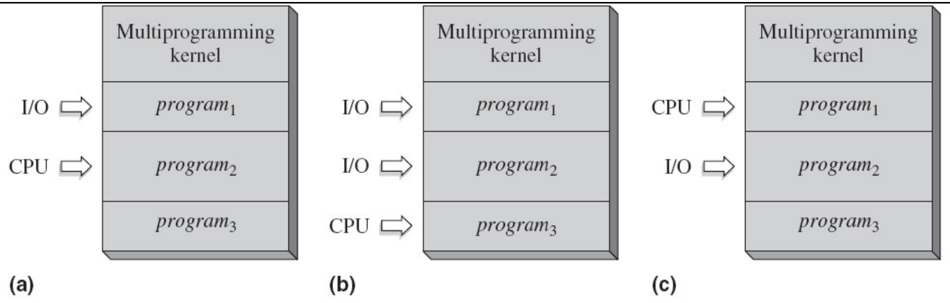

Figure 3.3 Operation of a multiprogramming system: (a) *program*$_2$ is in execution while *program*$_1$ is performing an I/O operation; (b) *program*$_2$ initiates an I/O operation, *program*$_3$ is scheduled; (c) *program*$_1$'s I/O operation completes and it is scheduled.

Table 3.4 Architectural Support for Multiprogramming

Feature	Description
DMA	The CPU initiates an I/O operation when an I/O instruction is executed. The DMA implements the data transfer involved in the I/O operation without involving the CPU and raises an I/O interrupt when the data transfer completes.
Memory protection	A program can access only the part of memory defined by contents of the *base register* and *size register*.
Kernel and user modes of CPU	Certain instructions, called *privileged instructions*, can be performed only when the CPU is in the kernel mode. A program interrupt is raised if a program tries to execute a privileged instruction when the CPU is in the user mode.

- An appropriate measure of performance of a multiprogramming OS is throughput
  - Ratio of the number of programs processed and the total time taken to process them
- OS keeps enough programs in memory at all times, so that CPU and I/O devices are not idle
  - Degree of multiprogramming: number of programs
  - Uses an appropriate program mix of CPU-bound programs and I/O-bound programs
  - Assigns appropriate priorities to CPU-bound and I/O-bound programs

Table 3.5 Techniques of Multiprogramming

Technique	Description
Appropriate *program mix*	The kernel keeps a mix of CPU-bound and I/O-bound programs in memory, where • A *CPU-bound program* is a program involving a lot of computation and very little I/O. It uses the CPU in long bursts—that is, it uses the CPU for a long time before starting an I/O operation. • An *I/O-bound program* involves very little computation and a lot of I/O. It uses the CPU in small bursts.
Priority-based preemptive scheduling	Every program is assigned a priority. The CPU is always allocated to the highest-priority program that wishes to use it. A low-priority program executing on the CPU is preempted if a higher-priority program wishes to use the CPU.

Priority of Programs
In multiprogramming environments, an I/O-bound program should have a higher priority than a CPU-bound program
When an appropriate program mix is maintained, an increase in the degree of multiprogramming would result in an increase in throughput.

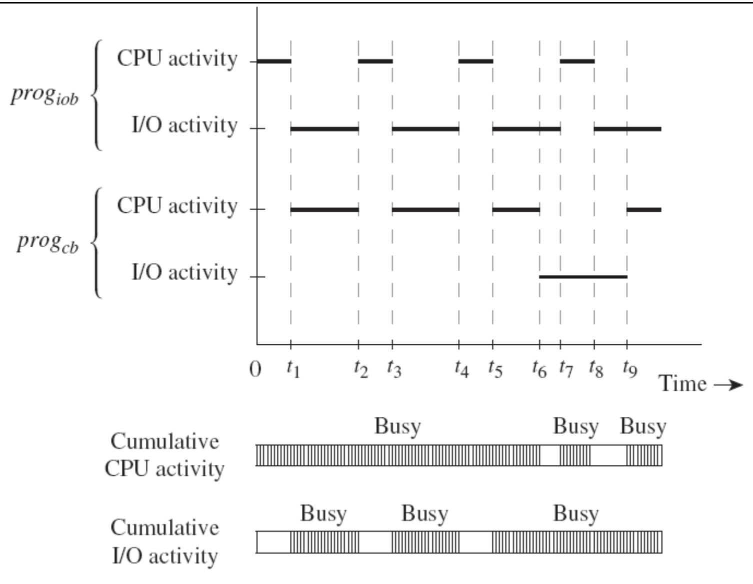

Figure 3.4 Timing chart when I/O-bound program has higher priority.

5. Consider two processes P_1 and P_2 whose CPU burst times are 16 and 30 ms respectively, while the I/O bursts are 110 and 60 ms, with time slice 10ms. Illustrate the operation of time sharing system with necessary timing diagram. Also explain time sharing operating systems with respect to memory management. 10M

Solving the instance with table-3M
Timing diagram-2 M
Mentioning about swapping -1M
Diagram and explanation about the memory management-4M

- Provide a quick response to user subrequests
  - *Round-robin scheduling with time-slicing*
    - Kernel maintains a *scheduling queue*
    - If *time slice (δ)* elapses before process completes servicing of a subrequest, kernel preempts it, moves it to end of queue, and schedules another process
      - Implemented through a timer interrupt

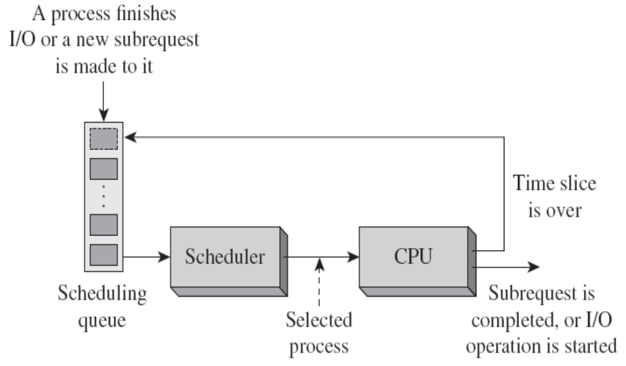
A process finishes I/O or a new subrequest is made to it
Time slice is over
Scheduler
CPU
Scheduling queue
Selected process
Subrequest is completed, or I/O operation is started

Figure 3.6 A schematic of round-robin scheduling with time-slicing.

Operation of Processes P1 and P2 in a time sharing system

Time	Scheduling List	Scheduled Program	Remarks
0	P1,P2	P1	P1 is preemted at 10ms
10	P2,P1	P2	P2 is preempted at 20ms
20	P1,P2	P1	P1 starts I/O at 26ms
26	P2	P2	P2 is preempted at 36ms
36	P2	P2	P1 starts I/O at 46ms
46	-	-	CPU is idle

Thus the response times are 136ms and 106ms respectively

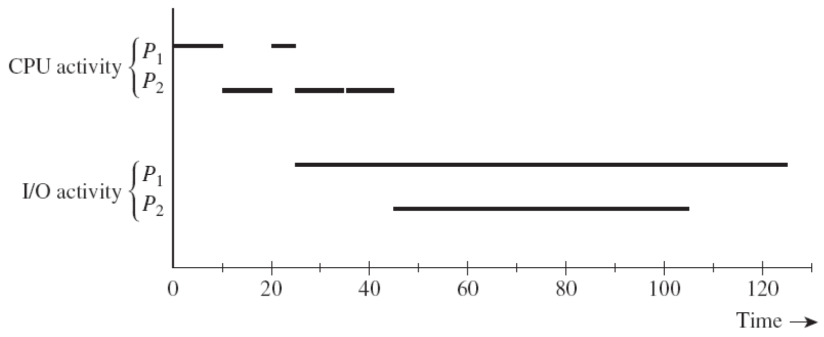

Figure 3.7 Operation of processes P_1 and P_2 in a time-sharing system.

Swapping of Programs

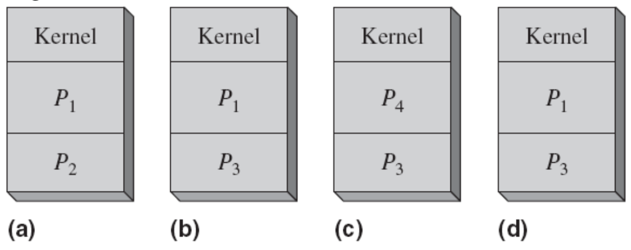

Figure 3.8 Swapping: (a) processes in memory between 0 and 105 ms; (b) P_2 is replaced by P_3 at 105 ms; (c) P_1 is replaced by P_4 at 125 ms; (d) P_1 is swapped in to service the next subrequest made to it.

- Kernel performs *swap-out* and *swap-in* operations
- The process of tempororilp removing the process from the memory is called swapping.

6. Explain layered design operating systems with a neat diagram. How it is superior compared to monolithic structure? 10M

Explanation about the semantic gap with diagram-3M
Explanation about the Layered structure with diagram-4M
Problems of monolithic structure-3M

Semantic Gap: The mismatch between the nature of operations needed in the application and the nature of operations provided in the machine.

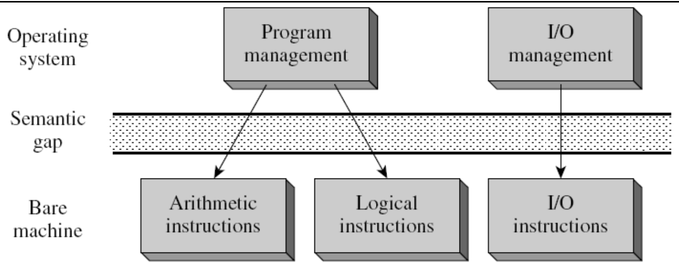

Figure 4.3 Semantic gap.

- Semantic gap is reduced by:
  - Using a more capable machine
  - Simulating an *extended machine* in a lower layer
- Routines of one layer must use only the facilities of the layer directly below it
  - Through its interfaces only

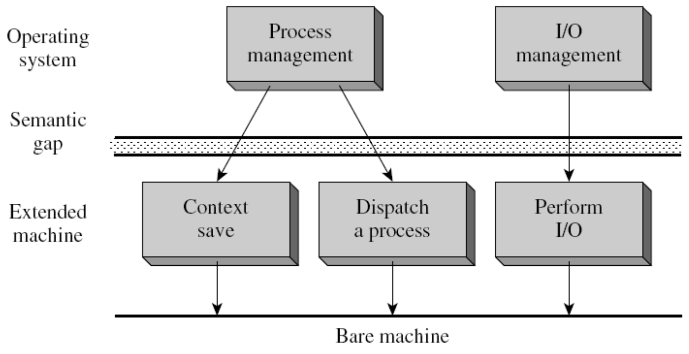

Figure 4.4 Layered OS design.

- Problems:
  - System operation slowed down by layered structure
  - Difficulties in developing a layered design
    - Problem: ordering of layers that require each other's services
      - Often solved by splitting a layer into two and putting other layers between the two halves
  - Stratification of OS functionalities
    - Complex design
    - Loss of execution efficiency
    - Poor extensibility
- **Problems with the monolithic structure:**
  - Sole OS layer had an interface with bare machine
    - Architecture-dependent code spread throughout OS
      - Poor portability
    - Made testing and debugging difficult
      - High costs of maintenance and enhancement
- Alternative ways to structure an OS:
  - *Layered structure*
  - *Kernel-based structure*
  - *Microkernel-based OS structure*

7. a) Write short notes on: 6M

1) Direct and Indirect naming 4M

Direct naming explanation-1.5M
Indirect naming explanation-1.5M

In *direct naming*, sender and receiver processes mention each other's name
In *symmetric naming*, both sender and receiver processes specify each other's name

1) In *asymmetric naming*, receiver does not name process from which it wishes to receive a message; kernel gives it a message sent to it by *some* process
2) In *indirect naming*, processes do not mention each other's name in **send** and **receive** statements

.

3) Key techniques of distributed operating system and benefits of it.

Techniques-1.5M
Benefits-1.5M

Table 3.9 **Key Concepts and Techniques Used in a Distributed OS**

Concept/Technique	Description
Distributed control	A control function is performed through participation of several nodes, possibly *all* nodes, in a distributed system.
Transparency	A resource or service can be accessed without having to know its location in the distributed system.
Remote procedure call (RPC)	A process calls a procedure that is located in a different computer system. The RPC is analogous to a procedure or function call in a programming language, except that the OS passes parameters to the remote procedure over the network and returns its results over the network.

Table 3.8 **Benefits of Distributed Operating Systems**

Benefit	Description
Resource sharing	Resources can be utilized across boundaries of individual computer systems.
Reliability	The OS continues to function even when computer systems or resources in it fail.
Computation speedup	Processes of an application can be executed in different computer systems to speed up its completion.
Communication	Users can communicate among themselves irrespective of their locations in the system.

(b)Define the following
1)Throughput 2)Priority 3)Swapping of programs 4)Time slice
Definition -1M each

Throughput:The average number of jobs,programs,processes or subrequests completed by a system in unit time.

Priority:A tie-breaking criterion under which a scheduler decided which request should be scheduled when many requests await service.

Swapping:The techique of temporarily removing a process from memeory of a computer.

Time Slice:The largest amount of CPU time any time shared process can consume when scheduled to execute on the CPU.
