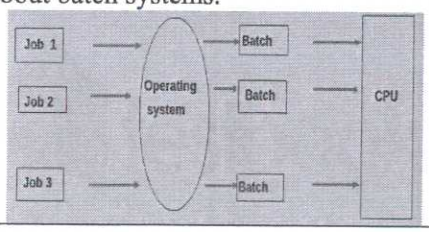
A

Scoring Indicators

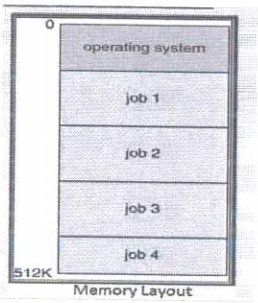
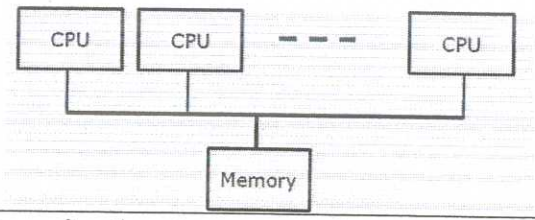
COURSE NAME : OPERATING SYSTEM

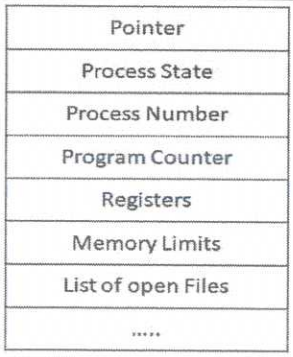
COURSE CODE : 5132 REVISION(2021) **QID : 2109230288**

Q No	Scoring Indicators	Split score	Sub Total	Total score
	PART A			**9**
I. 1	An operating system (OS) is a software program that serves as an intermediary between computer hardware and user applications, providing essential services to manage resources and facilitate efficient interaction.	1	1	
I. 2	True.	1	1	
I. 3	A process is a program in execution.	1	1	
I. 4	Ready state.	1	1	
I. 5	Throughput.	1	1	
I. 6	Logical Address.	1	1	
I. 7	Virtual Memory.	1	1	
I. 8	File type can be represented by file extension	1	1	
I. 9	A relative path to a file is a description of the file's location in relation to the current directory or another specified reference point.	1	1	
	PART B			24
II. 1	1. **Process Management:** • Process management involves the creation, scheduling, and termination of processes (programs in execution).It allocates resources such as CPU time, memory space, and I/O operations to processes **2.Memory Management:** • Memory management involves the allocation and deallocation of memory space to processes. • allocation and deallocation to minimize wastage. **3.File System Management:** • File system management involves organizing and managing files on storage devices, such as hard drives or SSDs. .	1 mark each or (any 3can give marks)	3	
II. 2	Batch systems are a type of operating system that manages and executes batches of tasks or jobs without user interaction during their execution. Here are some key points about batch systems: 	Expl with fig(3 marks) Otherwise 2 marks	3	

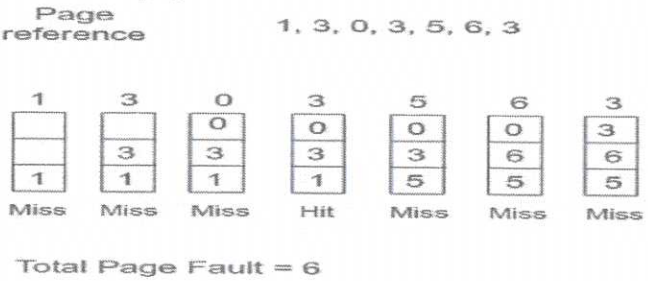
II. 3	A context switch refers to the process in which a computer's operating system saves and restores the state of a CPU (Central Processing Unit) for one process or thread and proceeds to execute the state of another process or thread. This allows multiple processes or threads to share a single CPU, giving the illusion of concurrent execution. Save Context: Load Context: Update PCB		3	
II. 4	Contiguous memory allocation is a memory management technique in which a process is allocated a block of contiguous (adjacent and uninterrupted) memory cells or segments. In this approach, the entire process must reside in a single continuous block of memory.	Expl with fig(3 marks) Otherwise 2 marks	3	
II. 5	The Translation Lookaside Buffer (TLB) is a special cache used in paging systems to improve memory access time. It contains page table entries that have been most recently used. When the processor receives a virtual address, it examines the TLB to determine if the corresponding page table entry is present (TLB hit). If it is, the frame number is retrieved, and the real address is formed. If the page table entry is not found in the TLB (TLB miss), the page number is used as an index to process the page table. The TLB helps reduce the effective access time by avoiding an additional memory reference to the page table. It stores frequently accessed page table entries, allowing for faster retrieval of frame numbers and real addresses1. By caching these entries, the TLB minimizes the need for repeated memory lookups, improving overall system performance.	3	3	
II. 6	In memory management, external fragmentation occurs when free memory blocks are scattered throughout the memory space, making it challenging to allocate large contiguous blocks for incoming processes1. This can lead to inefficient memory utilization and performance degradation1. External fragmentation happens when a dynamic memory allocation method allocates some memory but leaves a small amount of memory unusable2. The quantity of available memory is substantially reduced if there is too much external fragmentation2. There is enough memory space to complete a request, but it is not contiguous	3	3	
II. 7	Thrashing is a phenomenon that occurs in computer systems, particularly in operating systems with virtual memory, when the system spends the majority of its time swapping data between main memory (RAM) and the disk, instead of executing useful processes. Thrashing occurs when the system is over-committed in terms of memory. This happens when the total memory requirements of all active processes exceed the available physical RAM. Thrashing is a critical performance issue that occurs in systems with virtual memory. It can severely degrade system performance and must be addressed through various measures, including increasing physical memory, optimizing algorithms, and closely monitoring system performance.	3	3	
II. 8	A tree-structured directory offers a more organized and scalable approach to file management compared to single-level or two-level structures. It provides a hierarchical arrangement, allowing for logical grouping and categorization of files. This aids in efficient retrieval and navigation,	3	3	

	particularly in systems with numerous files and users. The tree structure also promotes user isolation, ensuring that each user has their designated section, enhancing security and reducing the likelihood of conflicts. Additionally, it accommodates scalability, allowing for the addition of new branches or subdirectories as the system expands. This structure simplifies file naming, as files can have straightforward, unique names within their specific branch. Overall, the tree-structured directory enhances efficiency, organization, and user-friendliness in file management.			
II.9	Sequential file organization is the simplest type of file organization in which the files are sequentially stored one after the other. Instead of storing the various records of the files in rows and column format (tabular form), it stores the records in a single row. This method is used for large files that are typically processed in their entirety.	3	3	
II.10	The time taken by the head to move to the appropriate cylinder or track is called seek time. Once the head is at right track, it must wait until the desired block rotates under the read-write head. This delay is latency time. Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head.	3	3	
	PART C			42
III.	**1.Assembler:** An assembler is a program that translates assembly language code into machine code or object code. Assembly language is a low-level programming language that uses mnemonics to represent instructions that can be directly executed by a computer's CPU. Assemblers are specific to the architecture of a particular CPU. They convert assembly language code into the binary machine language instructions understood by that CPU. The output of an assembler is an object file that can be further linked to create an executable program. **2.Compiler:** A compiler is a language translator that converts high-level source code written in languages like C, C++, Java, etc., into machine code. The compilation process involves several stages, including lexical analysis, syntax analysis, semantic analysis, and code generation. A compiler generates an executable file, which can be run directly by the computer's operating system without further translation. Once compiled, the program is independent of the platform and can be executed on any machine that has an appropriate runtime environment. **3.Interpreter:** An interpreter is a program that directly executes high-level source code without prior compilation. It reads and interprets the source code line by line at runtime and executes the corresponding machine-level operations. Interpreted languages, like Python, JavaScript, and Ruby, use interpreters. These languages do not produce an executable file; instead, they rely on the interpreter to execute the code. Interpreted languages are generally more flexible and suitable for tasks like scripting and rapid prototyping. **Comparison:** • Assemblers work with assembly language and convert it into	2.5+2.5+2	7	7

	machine code specific to a particular CPU architecture. • Compilers work with high-level programming languages and generate machine code that can run on various platforms without further translation. • Interpreters directly execute source code, without generating an independent executable file. Assemblers, compilers, and interpreters serve as language translators, converting source code into a form that a computer's hardware can execute. The choice between them depends on the programming language used and the specific requirements of the task at hand.			
IV	A multiprogrammed operating system is designed to maximize CPU utilization by allowing multiple programs (or processes) to be in various stages of execution simultaneously.  Memory Layout A multiprocessor system, also known as a parallel processing system, consists of multiple processors (or cores) that share a common memory and can execute multiple tasks simultaneously. Parallel Execution, Increased Processing Power... Execution, Efficient Use of Resources,Reduced Wait Times, Increased Throughput....  	Explanation with diagram 7 marks otherwise 5 marks	7	7
V	Deadlock is a situation in a computing system where two or more processes are unable to proceed because each is waiting for the other to release a resource, or more than two processes are waiting for resources in a circular chain. Four necessary conditions for deadlock to occur are: **1.Mutual Exclusion:** One or more resources must be non-shareable (or only one process at a time can use the resource). This means that if a process holds a resource, others cannot use it until the holding process releases it. **2.Hold and Wait (or Resource Holding):** A process must be holding at least one resource and waiting to acquire additional resources. This means that a process can request additional resources while still holding some, and it will be allowed to wait until those resources become	Def 1 mark Each condition 2 marks	7	7

	available. **3.No Preemption:** Resources cannot be forcibly taken away from a process; they can only be released voluntarily by the process holding them. This implies that a process cannot be interrupted and have its resources reallocated to another process. **4.Circular Wait:** There must be a circular chain of two or more processes, each of which is waiting for a resource held by the next member of the chain. This creates a cycle of dependencies, where each process is waiting for a resource that is held by another process in the cycle. When all these conditions are met, a deadlock can occur. In a deadlock, no processes can proceed, and the system may require intervention (e.g., manual termination of processes or the allocation of additional resources) to recover.			
VI	 **Process Control Block** The Process Control Block (PCB) is a crucial data structure used by the operating system to manage information about a running process. It contains essential details regarding the current state of a process, allowing the operating system to efficiently manage and control the execution of multiple processes. The structure of a PCB typically includes the following components: **1.Process ID (PID):**A unique identifier assigned to each process in the system. This number distinguishes one process from another and is used by the operating system for process management. **2.Program Counter (PC):**Also known as the instruction pointer, it keeps track of the address of the next instruction to be executed by the CPU. **3.CPU Registers:**These registers hold the current values of various CPU registers (like accumulator, stack pointer, etc.). They represent the process's state at a given moment. **4.CPU Scheduling Information:**nformation about the process's priority, scheduling state (e.g., ready, running, waiting), and any other scheduling-related data. **5.Memory Management Information:**his includes details about the process's memory usage, such as the base address, limit, page tables, and pointers to memory segments. **6.Accounting Information:**Information related to resource usage by the process, like CPU time used, execution history, and any other statistics. **7.I/O Status Information:**This component keeps track of the processes'	Fig 3 marks Expla 4 marks	7	7

	I/O requests, including pending I/O operations, open file descriptors, and other I/O-related data. **8.File Information:**Details about the files and resources associated with the process, such as open file handles, file pointers, etc.. **9.Process ID (PPID):**The identifier of the parent process that spawned or created the current process. **10.State of the Process:**Indicates whether the process is in a running, ready, or waiting state. The PCB is maintained by the operating system and is stored in the system's memory. When a context switch occurs (i.e., when the CPU switches from executing one process to another), the information in the PCB is crucial for restoring the state of the new process.			
VII	Round Robin (RR) scheduling is a pre-emptive scheduling algorithm in which each process is assigned a fixed time slice or quantum. When a process's time slice expires, it's temporarily suspended, and the CPU is allocated to the next ready process in the queue. This continues until all processes have completed execution. Here's an explanation of Round Robin scheduling with a Gantt chart: Consider the following processes with their arrival times (AT) and burst times (BT): Process Arrival Time Burst Time P1 0 5 P2 1 3 P3 2 2 P4 3 1 Time Quantum (TQ) = 2 **Gantt Chart:** | P1 | P2 | P3 | P4 | P1 | P2 | P3 | P4 | P2 | P3 | P2 |	Def 3.5marks Eg 3.5 marks	7	7
VIII	A Resource Allocation Graph (RAG) is a directed graph used to represent the allocation of resources and the relationships between processes and resources in a system. It is a crucial tool in deadlock detection in operating systems. In a Resource Allocation Graph, nodes represent processes and resources. There are two types of nodes: 1. **Process Nodes (P):** Represent processes in the system. 2. **Resource Nodes (R):** Represent resources that can be allocated to processes. Edges in the graph represent relationships: • **Request Edge (R -> P):** Represents a process P requesting resource R. • **Assignment Edge (P -> R):** Represents a process P being allocated resource R. The given resource allocation graph is single instance with a cycle contained in it.There are no instances available currently and both the	4+3	7	7

	processes require a resource to execute.Therefore, none of the process can be executed and both keeps waiting infinitely.Thus, the system is in a deadlock state.			
IX	This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal. Consider page reference string 1, 3, 0, 3, 5, 6, 3 with 3 page frames.Find the number of page faults.  Initially, all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.** when 3 comes, it is already in memory so —> **0 Page Faults.** Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault.** 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.** Finally, when 3 come it is not available so it replaces 0 **1 page fault.**	7	7	7
X	Segmentation is a memory management technique that divides the physical memory into variable-sized segments, each representing a logical unit of a program (such as a procedure, function, or data structure). Each segment is assigned a unique identifier or segment number. Segmentation allows a program to be subdivided into meaningful, independent modules, making it easier to manage and execute. Key features of segmentation: **Variable Segment Sizes:** Segments can vary in size and represent different parts of a program, such as code, data, stack, and more. **Logical Units:** Segments correspond to meaningful units of a program, making it easier for programmers to understand and manage memory allocation. **Dynamic Growth:** Segments can dynamically grow or shrink based on the needs of the program. This allows for flexibility in memory allocation. **Fragmentation:** Segmentation can lead to both internal and external fragmentation. Internal fragmentation occurs when there is unused space within a segment, while external fragmentation occurs when there are unused segments scattered throughout memory. **Segment Table:** The operating system maintains a segment table that keeps track of the base address and limit (size) of each segment in physical memory. **Protection and Access Control:** Segmentation allows for access control by assigning privileges to segments. This helps prevent unauthorized access to critical parts of a program. **Differences Between Segmentation and Paging:**	7	7	7

	Unit of Allocation: Segmentation divides memory into variable-sized segments, each representing a meaningful unit of a program. Paging divides memory into fixed-sized blocks called pages. **Management of Address Space:** Segmentation is more flexible in managing address spaces, as segments can vary in size and represent different program components. Paging divides the address space into fixed-sized pages, which can lead to more efficient use of memory, but may not align with the logical structure of a program. **Handling of Fragmentation:** Segmentation can lead to both internal and external fragmentation, especially if segments grow and shrink dynamically. Paging can lead to internal fragmentation, as the last page of a program may not be fully utilized. **Hardware Support:** Segmentation requires hardware support for managing segment tabl Paging relies on hardware support for page tables. **Translation Process:** In segmentation, translation of logical addresses to physical addresses involves using a segment table to find the base address and limit of the segment. In paging, translation involves using a page table to map page numbers to frame numbers. In summary, segmentation and paging are both memory management techniques, but they differ in how they divide and manage the address space of a program. Segmentation is more flexible and aligns better with the logical structure of a program, while paging offers more efficient use of memory but may not always match the program's natural units.			
XI	**First Fit:** In the First Fit memory allocation strategy, the operating system allocates the first available block of memory that is large enough to accommodate the process. It starts searching from the beginning of the available memory and allocates the first block that meets the size requirements of the process. If the allocated block is larger than needed, the remaining portion becomes a free block. First Fit is simple and easy to implement, but it can lead to more external fragmentation over time as memory becomes fragmented. **Best Fit:** The Best Fit memory allocation strategy aims to find the smallest available block of memory that can accommodate the process. It searches through all the available blocks of memory and selects the one that is closest in size to the process's requirements without being smaller. This strategy is more efficient in terms of memory utilization and can help reduce external fragmentation. However, it may require more time to search through the available blocks, which can lead to slower allocation. **Worst Fit:** In the Worst Fit memory allocation strategy, the operating system allocates	7/3	7	7

	the largest available block of memory to the process. It looks for the largest free block and allocates it to the process. This can potentially lead to more fragmentation, as smaller blocks may be left unused. Worst Fit is generally not as efficient as Best Fit in terms of memory utilization, but it can help in situations where large blocks of memory are needed.			
XII	**Compile-Time Binding (Static Binding):** • **Definition:** Compile-time binding refers to the process of associating a method call with the corresponding method implementation during the compilation phase. • **Timing:** This binding occurs at compile time, which means it is done before the program is run. **Link-Time Binding (Static Linking):** • **Definition:** Link-time binding involves linking external references or libraries with the program code during the linking phase. • **Timing:** This binding occurs during the linking phase, which is typically after the compilation phase but before the program is run. **Run-Time Binding (Dynamic Binding):** • **Definition:** Run-time binding refers to the process of determining the method to call at runtime based on the actual type of an object. • **Timing:** This binding occurs during runtime, while the program is running. **Comparison:** • **Flexibility:** • Compile-time and link-time bindings are static and lack flexibility at runtime. • Run-time binding provides the highest degree of flexibility and dynamic behavior. **Efficiency:** • Compile-time and link-time bindings are generally more efficient in terms of execution speed because the method call is directly mapped to the appropriate implementation. • Run-time binding may involve dynamic lookup, which can introduce a slight overhead. **Error Detection:** • Compile-time binding catches errors related to method signatures or undefined references during compilation. • Run-time binding may result in runtime errors if a method is missing or incompatible. The choice of binding scheme depends on the specific requirements of the application. Compile-time binding is suitable for situations where efficiency is critical and the program structure is fixed. Run-time binding is preferred when flexibility, polymorphism, and dynamic behavior are more important than speed. Link-time binding serves as a compromise between the two, allowing for some degree of flexibility while still providing efficiency.	Split marks in to each parts	7	7
XIII	Contiguous file allocation is a file allocation technique used in operating systems, where a file is stored in a contiguous block of storage space on a	3+4	7	7

	storage medium (such as a disk). In this method, the file occupies a single, uninterrupted block of consecutive storage locations. The starting location and size of the file are stored in a file allocation table. **Advantages:** Simplicity and Efficiency: Fast Access: Reduced Disk Head Movement: . Good Performance for Large Files: **Drawbacks:** Fragmentation: Wastage of Space: Limited Flexibility: Difficulty in File Growth: Comparison with Other Allocation Techniques: **Linked Allocation:** Contiguous allocation is more efficient in terms of disk access speed since files are stored as a single block, whereas linked allocation involves traversing a chain of pointers, which can be slower. However, linked allocation does not suffer from fragmentation issues. **Indexed Allocation:** Indexed allocation provides more flexibility for file growth and does not suffer from fragmentation. However, it requires additional space for maintaining an index block. **Multilevel Indexing:** Multilevel indexing allows for efficient access to large files and can dynamically allocate space for new files. However, it involves complex data structures and can have higher overhead. Contiguous file allocation is efficient for accessing and managing large files but can lead to fragmentation and may not be suitable for systems with dynamic storage needs. Other techniques like linked allocation, indexed allocation, and multilevel indexing offer different trade-offs and may be more suitable for certain scenarios.			
XIV	Given the request queue: 3, 10, 1, 5, 8, 12, and the initial head position of 7, we can calculate the seek distances: 1. Seek from 7 to 3: $\|7 - 3\| = 4$ tracks 2. Seek from 3 to 10: $\|3 - 10\| = 7$ tracks 3. Seek from 10 to 1: $\|10 - 1\| = 9$ tracks 4. Seek from 1 to 5: $\|1 - 5\| = 4$ tracks 5. Seek from 5 to 8: $\|5 - 8\| = 3$ tracks 6. Seek from 8 to 12: $\|8 - 12\| = 4$ tracks Total Seek Time = $4 + 7 + 9 + 4 + 3 + 4 = 31$ tracks So, the total seek time for the FCFS disk scheduling algorithm in this case is 31 tracks.	3+4	7	7