OS Assignment - 01
Made By - Arsh Hasan,
Roll no - 2100100321
B Tech Comp. Engineering
II^nd yr, CE41

Ans(1) :- Technically, it is possible to operate a computer without an operating system, but it would be very difficult and limited in functionality.
(2) You would have to manually manage hardware resources and run programs directly from command line or by writing code in assembly or machine language
(3) This would require deep understanding of computer architecture & programming hence would be very - time - consuming and error-prone
(4) With no GUI, you would have to interact with computer solely through a command line interface.

Q2 :- The major drawback of multiprogrammed batch systems was the lack of user/programmer interaction with their jobs. How can you overcome this?

Ans :- These are four ways to overcome this limitations?
(1) Time Sharing (multiuser) system: In a time-sharing, multiple users can share the resources of the computer simultaneously. The system switches rapidly btw users, giving each one a small amount of time to interact with the computer. This allows users to interact with their jobs in real time.
(2) Interactive programming environment: The environment provide a set of tools facilities that allows users to interact with their programs during the development process.
(3) Job Control language: Batch systems can be digital to include Job Control language (JCL), which is a scripting language used to specify and control batch jobs. JCL can be used to provide users with greater control over their jobs, allowing them to specify options such as I/O files,

T. pages : 27

, resource allocation, program parameters etc.

(4) Job Monitoring: provides users with feedback on the status of their jobs. This can include real-time progress reports, notifications of errors or warnings, information on job completion & output. Hence, batch systems can be made more user-friendly and interactive.

Q3. The response time is the major requirement of a multiuser time-sharing OS. What are the things that need to be improved for this requirement from a system designer's viewpoint?

Ans - There are five ways to improve the response time :-

(1) Efficient Scheduling: The system designer needs to implement an efficient process scheduling algorithm that prioritizes tasks based on their priority. The algorithm should be designed to minimise the average response time of the system.

(2) Resource allocation: The system designer needs to allocate system resources such as CPU time, memory, I/O devices efficiently to reduce contention & avoid system overload, using a good resource allocation algorithm.

(3) System Performance Monitoring: to identify bottlenecks and optimise system parameters. Performance metrics such as CPU utilisation, Memory usage, I/O performance should be analysed to identify issues and optimise the system for fast response time.

(4) Load Balancing: helps in distributing the workload evenly among system resources. Load balancing helps avoid resource contention & ensures that all users get an equal share of system resources, hence improving response time.

(5) Caching: Caching techniques helps to store frequently accessed data in memory or disk to reduce response time, by reducing the time it takes to access data.

Q4. Is time-sharing OS suitable for real-time systems?

Ans - (1) A time-sharing OS is generally not suitable for real-time systems because it is designed to share system resources among multiple users and

Ans- applications, which can result in unpredictable delays in responding to real time events.

(2) In time-sharing OS, CPU time scheduling is not deterministic, which means that the system cannot guarantee that a task will be executed within a specific time frame.

(3) This unpredictability makes time-sharing systems unsuitable for real-time systems, where responses times needs to be deterministic & bounded.

Q5. Examine the following conditions and find appropriate OS for them:-

(a) In a LAN, users want to share some costly resources like laser printers.

↳ Network Operating System: allows multiple users to share network resources and manages their access.

(b) Multiple users on a system want quick response on their terminals.

↳ Multiuser time-sharing Operating System: allows multiple users access and quicker response time.

(c) Railway Reservation System.

↳ Real-Time Operating System: manages real-time response and precise timing for reservation processing.

(d) A user wants to work with multiple jobs on his system.

↳ Multi-tasking Operating System: allows multiple programs to run simultaneously on a single computer.

(e) In a network system you want to transfer file and log on to some node.

↳ Network Operating System: provides monitoring and logging features to track file usage and identify issues.

(f) There are some jobs in the system which does not want user interaction.

↳ Batch Processing Operating System: jobs are executed in the background without user intervention.

(9) Washing Machine.

↳ <u>Embedded Operating System</u>: limited processing power and memory. Controls all washing machine features/functions!

Q6. Explore the features of operating system being used in recent design of smartphones.

<u>Ans</u> – Some key features are:-

(1) <u>User interface</u>: Modern smartphone's OS have user friendly interfaces that make it easy for users to navigate through the phone's features and applications. They are designed to be intuitive & responsive, with features like touch-screen gestures.

(2) <u>App Store</u>: Most smartphone's OS come with an app store that allows users to install apps from a centralized location. These app stores offer a wide range of apps that can be used to perform various tasks.

(3) <u>Multi-Tasking</u>: Smartphones are designed to perform multiple tasks at once, by allowing users to run multiple apps simultaneously. This is achieved through a combination of hardware & software features that optimize performance & memory usage.

(4) <u>Security</u>: Smartphone OS have advanced security features that protect user data and prevent unauthorised access, like encryption, remote wipe, biometric authentication etc.

(5) <u>Connectivity</u>: Smartphones are designed to be always connected, & modern OS support a range of connectivity options, like Wi-Fi, Bluetooth, NFC, & Cellular Networks!

(6) <u>Customization</u>: Smartphone OS allows users to customize their devices to suit their individual preferences, by custom wallpapers, widgets, app icons.

(7) <u>Voice assistants</u>: Many smartphone OS include voice assistants like Apple's Siri, Google Assistant, Amazon's Alexa. These assistants use natural language processing & machine learning algorithms to understand user commands.

Ans - ⑧ Augmented Reality: Apple's iOS, includes features for Augmented Reality (AR). AR allows users to overlay digital information on the real world, creating interactive & immersive experiences.

Q7. Do all operating systems contain shell?

Ans - No, not all operating systems contain a shell. A shell is a command line interface (CLI) that allows users to interact with an OS by typing commands. While many popular OS like macOS include a shell by default, there are other OS's that do not have a shell. Example: Microsoft Windows has a GUI that is designed to be more user friendly, although it also includes a PowerShell that can be used by more advanced users. In addition some OS are designed for specialized applications, such as embedded systems, may not include a shell, as they are designed to run specific applications without user interaction.

Q8: "Multi-programming is inherent in multi-user & multi-tasking systems". Explain how?

Ans - ① Multi-programming is inherent in multi-user & multi-tasking systems because these systems are designed to allow multiple users to run multiple tasks concurrently on a single computing device.

② In both multi-user & multi-tasking systems, the OS uses techniques such as time-sharing & process scheduling to manage allocation of resources & ensure that each user gets a fair share.

③ These techniques involve dividing the available time into small time slices & allocating them to different programs in a round-robin fashion, allowing each program to run for a short time before switching to next.

(4) Overall, the ability to run multiple programs simultaneously is a key feature of OS's, enabling effective & efficient use of computing resources.

Q9. There are three jobs running in a multi-programming system with the following requirements:-

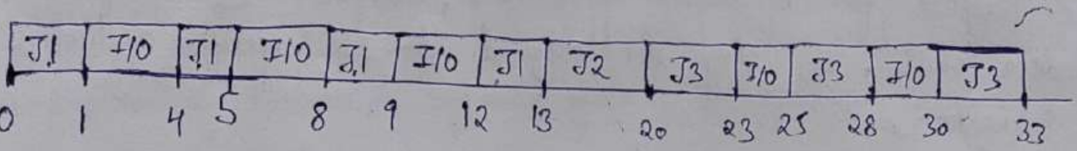
job 1: requires disk after every 1 min, device service time-including wait and access = 3 min, total processing time = 4 min.

job 2: does not require any I/O, total processing time = 7 mins.

job 3: requires printer after every 3 min, device service time including wait and access = 2 min, total processing time = 9 min.

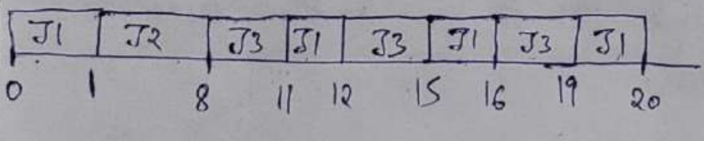
Prepare a timing chart showing the CPU and I/O activities of the jobs. Compute the total time for monoprogramming and multiprogramming and then compare the results.

Ans - In monoprogramming environment,



total = 33 minutes time.

In multiprogramming environment,



total time = 20 minutes.

The timing chart shows and proves that the job execution times are decreased in case of multiprogramming systems than monoprogramming systems as multiple jobs can be executed simultaneously, in multiprogramming systems.

Name : Dhruv Mishra
Class : CE41
Roll No : 2100100 3036
Subject : OS Assignment - 2 (Chapter 2)

Q1. The interrupt number of a hardware interrupt is 8. At what location in the IVT its ISR address will be found?

Ans - In x86 architecture, IVT starts at memory location 0x0000 and ends at 0x03FF. Each entry in IVT is 4 bytes long, with the first 2 bytes storing the segment address of the ISR and second 2 bytes storing the offset address of ISR.

∴ Memory Address = IVT start address + offset of interrupt vector '8'

= 0x0000 + 32 (8*4)

= 0x0020.

Q2. Is nested interrupt possible? If yes, how are they handled?

Ans - Yes, nested interrupts are possible in some processors and microcontrollers. Nested interrupts occur when an interrupt is triggered while the processor is already servicing another interrupt.

When a nested interrupt occurs, the processor first saves the current context of the interrupted ISR on the stack. It then switches to the ISR of the high priority interrupt and services it. Once the high priority interrupt is serviced, the processor restores the saved context of the previous ISR from the stack and resumes its execution.

To handle nested interrupts properly, the processor must be able to handle interrupt priorities and mask lower-priority interrupts while servicing a higher-priority interrupt. In addition, the ISR stack must have enough space to store the context of all nested interrupts.

Q3. All I/O instructions are privileged. Then, how does a user access the devices?

Ans - In most modern OS, user programs cannot access hardware devices directly. Instead, they must use System Calls or device drivers to interact indirectly

With the hardware devices.
When a user program makes a system call, the processor switches to kernel mode, which gives the OS full control over the system resources, including hardware devices. The OS then performs the requested operation on behalf of the user program and returns control back to the user program.
Device Drivers are responsible for managing the low level details of device access, such as communicating with the device, handling interrupts, & buffering data.
When a user program requests a device operation, the OS uses the appropriate device driver to perform the operation on the device.

Q4. Which of the following instructions should be privileged:-

(a) Switch from user mode to kernel mode.
↳ Privileged as only code running in kernel mode, such as the OS kernel, has the privilege to execute this instruction.

(b) Updating base and limit register
↳ Privileged as only the OS kernel or device driver running in kernel mode has the privilege to modify base and limit register values.

(c) Clear memory location
↳ Not Privileged as it is a common operation that can be performed by any program or process with access to the memory location.

(d) Set value of timer
↳ Privileged, in case of most computer architectures as timer is a hardware device that generates interrupts at regular intervals to allow the OS to perform scheduling & other time-based tasks.
Not Privileged, in case of embedded devices, where timer can be implemented entirely in user mode. (as software timer).

(e) Read a clock
↳ Privileged, in case of most computer architectures as clock as hardware device provides the system with a reference for timekeeping and synchronization.
Not Privileged, in case of embedded devices, clock may be as software based timer that is implemented entirely in user mode.

aces,
ection on
user program,
of data access,
utilities data,

(f) Interrupts are disabled.
↳ Privileged as that can only be executed in kernel mode.

(g) Executing a loop to enter user data
↳ Not Privileged as it is common operation that can be performed by any program or process with access to the user I/O streams.

(h) Load a value in processor register.
↳ Not Privileged as it is common operation that can be performed by any program or process with appropriate privileges to access the register in question / problem given.

(i) Abort a process.
↳ Privileged as that can only be executed by the OS kernel.

(j) Read input from keyboard.
↳ Not Privileged typically as it is common operation that can be performed by any program that has access to keyboard input stream.

(k) Send a file to printer to print.
↳ Privileged typically as it is implemented as a system call, which allows user-level programs to request the OS kernel to print a particular file.

(l) A global variable in the user process reinitialized
↳ Not Privileged as it is common operation that can be performed by any program that has access to the global variable.

Q5. Inter-sector and Inter-track gaps are used on the disk to avoid errors. How do these gaps affect storage utilization on the disk?

Ans - These gaps have an impact on storage utilization on the disk. Since each gap occupies a certain amount of physical space, it reduces the amount of available space that can be used for storing data. In general, larger gaps tend to provide better error tolerance and reliability, but at cost of reduced storage of utilising.

Q6. Study the DOS and Windows operating systems with reference to dual mode protection and find out which operating system provides a better protection in terms of multi-tasking.

Ans - DOS (Disk Operating System) is an early OS that was developed by Microsoft in 1980s. It was designed for single user, single tasking systems and did not have a true dual mode protection mechanism. Instead, it relied on rudimentary memory protection scheme that separated system memory from application memory to prevent programs from accidently overwriting critical system data. However, this scheme was not foolproof and did not provide effective protection against malicious software or user errors.

In contrast, Windows provides a robust dual-mode protection mechanism that seperates user-mode and kernel-mode code and data. This protection mechanism, which is implemented through the Windows NT architecture, allows multiple programs to run simultaneously in user mode while preventing them from accessing system resources or data in kernel mode. This mechanism also provides better protection against malicious software and user errors, as it isolates user programs from critical system data and prevents unauthorized access.

In terms of multi-tasking, Windows provides better support than DOS.
Overall, Windows provides better protection in terms of dual mode protection and multi-tasking than DOS.

Name: Dhruv Mishra
Class: CE41
Roll No: 2100103036
Subject: OS Assignment-3 (Chapter 4)

Q1. Would microkernel architecture work well for design of an object-oriented OS? Justify your answer.

Ans - ① Microkernel architecture can work well for the design of an object oriented OS, as it promotes modularization and abstraction of functionality, which are important concepts in object-oriented design.

② In microkernel architecture, the kernel provides a minimal set of services, such as process management and memory management, while other services, such as file systems and device drivers, are implemented as seperate modules running outside the kernel. This allows for better separation of concerns & easier maintenance of the system.

③ Microkernel architecture allows for greater flexibility and customization of the OS, as different modules can be swapped in and out as needed.

④ This can be especially useful for embedded systems or other specialized environments where specific functionality may be required.

Q2. Design a format of message in message-passing system of microkernel architecture.

Ans - In a microkernel based message-passing system, the format of the message should include the following components:-

① Message Type: This field specifies the type of the message, such as a request, a response or an error message.

② Source and Destination Processes: These fields identify the source and destination processes of the message, respectively. The source process sends the message, while the destination process receives the message.

③ Payload: This field contains the data that is being sent with the message. The size & structure of the payload may vary depending on the type of message.

(4) Message ID: This field contains a unique identifier for the message. This can be used to track the progress of the message through the system, or to match requests with responses.

(5) Timestamp: This field contains a timestamp indicating when the message was sent. This can be useful for tracking message delays and for debugging purposes.

(6) Security Information: This field contains any security-related information, such as authentication or authorization data, that is required for the message to be processed.

(7) Additional Metadata: Depending on the specific requirements of the system, additional metadata may be included in the message format. For example, this could include information about the priority of the message, or the expected response time.

Q3. How is reliability increased in microkernel architecture?.

Ans - A few ways to improved reliability:-

(1) Modularity: In microkernel architecture, the OS is divided into small, independent modules, each of which performs a specific task. This modular design makes it easier to isolate and fix bugs or faults within a single module without affecting the entire system.

(2) Fault Isolation: In a microkernel based system, each module run in its own protected memory space & communicates with other modules through message passing. This ensures that a fault or error in one module cannot affect other modules or the entire system. If a module crashes, it can be restarted or replaced without affecting rest of the system.

(3) Verification & Validation: Since the microkernel provides a small set of services, it is easier to verify & validate the correctness of these services. The modular design allows for testing & verification of each module in isolation, which can help to identify & fix faults early in development process.

(4) Redundancy: Microkernel based systems can be designed to include redundancy mechanisms, such as hot-swappable components. This can help to ensure that

Ans – system remains operational even in the presence of hardware or software failures.

Q4. Explore some research issues in designing an exokernel.

Ans – In designing an exokernel, there are several research issues that need to be addressed:–

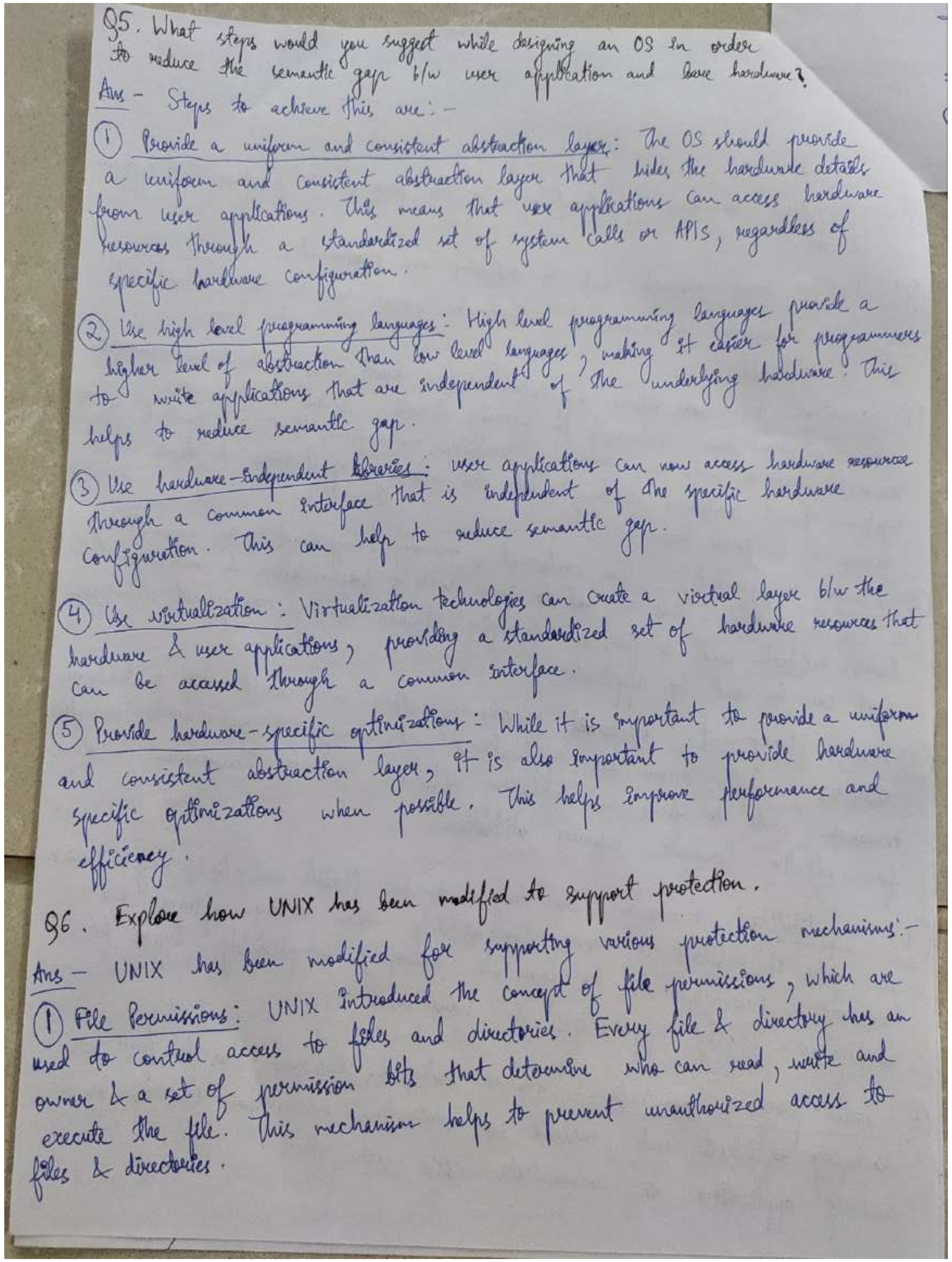
① Security: Exokernels allow applications to have direct access to hardware resources, which can make them vulnerable to security threats. To address this issue, research needs to focus on developing security mechanisms that can protect the system against malicious attacks, while still allowing applications to have direct access to hardware resources.

② Memory Management: In exokernel, memory management is the responsibility of the application. This can lead to issues such as fragmentation & memory leaks. Research needs to focus on developing efficient memory management techniques that can be used by applications.

③ Resource Management: Exokernel provides applications with direct access to hardware resources, which means that applications must manage these resources themselves. Research needs to focus on developing efficient resource management techniques for effective hardware resources utilization.

④ Compatibility: Exokernels are designed to be flexible and light-weight, which can make it difficult to ensure compatibility with existing software and hardware. Research has to focus on developing compatibility layers for working with existing applications & hardware.

⑤ Inter-process Communication: In an exokernel, Research has to focus on developing efficient and reliable communication mechanisms that can be used by applications to communicate with each other.



Q5. What steps would you suggest while designing an OS in order to reduce the semantic gap b/w user application and bare hardware?

Ans – Steps to achieve this are:–

① Provide a uniform and consistent abstraction layer: The OS should provide a uniform and consistent abstraction layer that hides the hardware details from user applications. This means that user applications can access hardware resources through a standardized set of system calls or APIs, regardless of specific hardware configuration.

② Use high level programming languages: High level programming languages provide a higher level of abstraction than low level languages, making it easier for programmers to write applications that are independent of the underlying hardware. This helps to reduce semantic gap.

③ Use hardware-independent libraries: user applications can now access hardware resources through a common interface that is independent of the specific hardware configuration. This can help to reduce semantic gap.

④ Use virtualization: Virtualization technologies can create a virtual layer b/w the hardware & user applications, providing a standardized set of hardware resources that can be accessed through a common interface.

⑤ Provide hardware-specific optimizations: While it is important to provide a uniform and consistent abstraction layer, it is also important to provide hardware specific optimizations when possible. This helps improve performance and efficiency.

Q6. Explore how UNIX has been modified to support protection.

Ans – UNIX has been modified for supporting various protection mechanisms:–

① File Permissions: UNIX introduced the concept of file permissions, which are used to control access to files and directories. Every file & directory has an owner & a set of permission bits that determine who can read, write and execute the file. This mechanism helps to prevent unauthorized access to files & directories.

Ans – (2) User and Group ID's : UNIX also introduced the concept of user and group IDs, which are used to identify users & groups on the system. Each user & group has a unique ID, and file permissions can be set for specific users & groups.

(3) Access Control Lists (ACLs): Some variants of UNIX, such as Solaris, support ACLS. ACLs provide a finer level of control over file permissions, allowing specific users & groups to be granted or denied access to files & directories.

(4) Security Extensions: Some UNIX variants, such as AIX, include security extensions that provide additional security features, such as mandatory access control & auditing. MAC allows system administrators to define security policies that restrict the actions of users & applications.

(5) Sandboxing: More recent UNIX variants, such as iOS, have introduced sandboxing mechanisms that isolate applications from each other & from the rest of the system. Sandboxing helps to prevent applications from accessing unauthorized resources & from interfering with other applications.

Name: Dhruv Mishra
Class: CE41
Roll No: 2100103036
Subject: OS Assignment-4 (Chapter 5)

Q1. In a large OS, if there is a single queue for maintaining the blocked processes, it will be inefficient to search a required process that needs to be awakened on occurrence of an event. What can be the remedies for this inefficiency?

Ans- Some common remedies are:-

(1) Multiple Queues: There can be separate queues for I/O events, timer events, & synchronization events. This would reduce the time required to search for a required process.

(2) Priority Queues: Processes with higher priority can be placed at the front of the queue, making them easier to find and awaken when necessary.

(3) Indexed Queues: can be used to store blocked processes, where the index is based on the event that causes the process to block, leading to efficient searching.

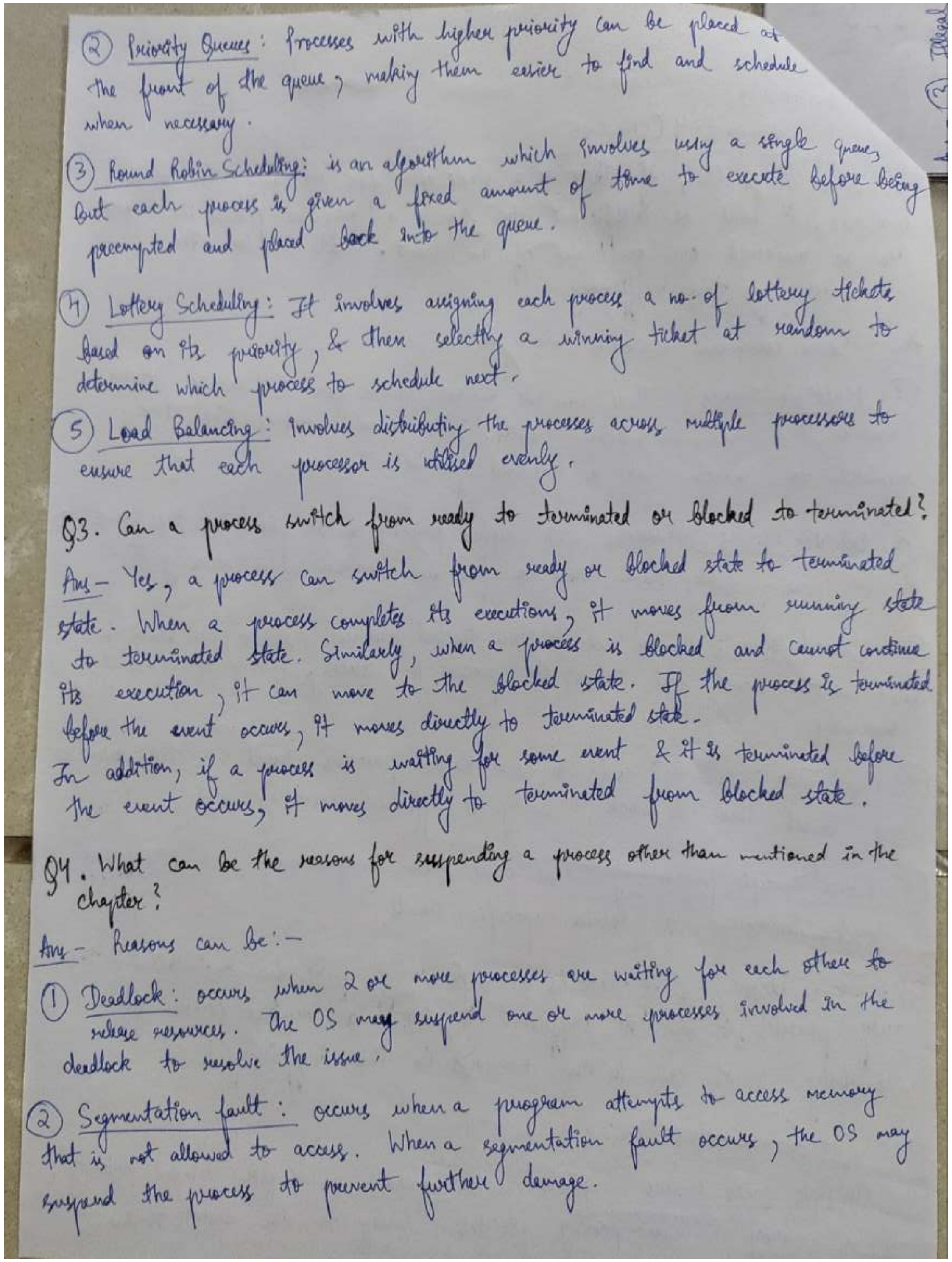
(4) Hash Tables: can be used to store blocked processes based on the event that caused them to block, leading to efficient searching.

(5) Event-specific programming: or event driven programming, where processes register themselves, to receive specific events.

Q2. In a large OS, if there is a single ready queue for maintaining the ready processes, it will be inefficient to search a required process for scheduling. What can be the remedies to this inefficiency?

Ans- Some remedies are:-

(1) Multiple Ready Queues: There can be seperate queues for high-priority, medium-priority & low-priority processes, leading to less search time.



② Priority Queues : Processes with higher priority can be placed at the front of the queue, making them easier to find and schedule when necessary.

③ Round Robin Scheduling: is an algorithm which involves using a single queue, but each process is given a fixed amount of time to execute before being preempted and placed back into the queue.

④ Lottery Scheduling: It involves assigning each process a no. of lottery tickets based on its priority, & then selecting a winning ticket at random to determine which process to schedule next.

⑤ Load Balancing: Involves distributing the processes across multiple processors to ensure that each processor is utilised evenly.

Q3. Can a process switch from ready to terminated or blocked to terminated?

Ans – Yes, a process can switch from ready or blocked state to terminated state. When a process completes its executions, it moves from running state to terminated state. Similarly, when a process is blocked and cannot continue its execution, it can move to the blocked state. If the process is terminated before the event occurs, it moves directly to terminated state.
In addition, if a process is waiting for some event & it is terminated before the event occurs, it moves directly to terminated from blocked state.

Q4. What can be the reasons for suspending a process other than mentioned in the chapter?

Ans – Reasons can be :–

① Deadlock : occurs when 2 or more processes are waiting for each other to release resources. The OS may suspend one or more processes involved in the deadlock to resolve the issue.

② Segmentation fault : occurs when a program attempts to access memory that is not allowed to access. When a segmentation fault occurs, the OS may suspend the process to prevent further damage.

Ans - (3) Illegal instruction: exception occurs when a program tries to execute an invalid machine instruction. When an illegal instruction exception occurs, the OS may suspend the process to prevent further damage.

(4) Stack overflow: occurs when a program tries to allocate more memory on the stack than is available. When a stack overflow occurs, the OS may suspend the process to prevent further damage.

Q5. Is it necessary that every event causing interrupt will change the state of the running process?

Ans - No, it is not necessary that every event causing interrupt will change the state of the running process. Maskable interrupts can be handled without changing the state of the running process.

Maskable Interrupts: The processor can continue executing the current process while the interrupt is pending. Once the processor is ready to handle the interrupt, it can save the current state of the running process, execute the interrupt handler, & then return to the interrupted process.

Q6. The suspended queue is maintained on the hard disk & therefore, I/O operations are required with it. Does it have any impact on the performance of process management because all other queues are managed in the main memory?

Ans - If the suspended queue were to be stored on the hard disk instead of main memory, it could have significant impact on the performance of the process management. Accessing data from hard disk is much slower than accessing data from main memory, due to much longer seek and access times involved. This could result in increased overhead and longer wait times for processes that are waiting in suspended queues.

However, even if the suspended queue were stored in stored on the hard disk, the impact on performance will be minimal, since the number of processes in the suspended queue are small compared to number of processes in ready & blocked queues.

Q7. How does a blocked or blocked-suspended process know about the completion of the I/O device or the resource for which it is waiting?

Ans – There are few ways that the OS can signal a blocked or blocked-suspended process about the completion of an I/O operation such as :–

① Interrupts : Many I/O devices generate interrupts when they have completed an operation. When an interrupt occurs, the OS can check which process was waiting for the I/O operation & wake it up.

② Signals : The OS can send a signal to the waiting process when the I/O operation becomes available. The process can then wake up & resume execution.

③ Polling : In some cases, the OS has to poll the device or resource periodically to check if it has become available. When it is available, the OS can wake up the waiting process.

Q8. What event handler would be executed in the following cases:–

ⓐ The running process has finished its execution before completion of its time slice.

↳ The OS's scheduler will preempt the running process and select the next process to run based on its priority.

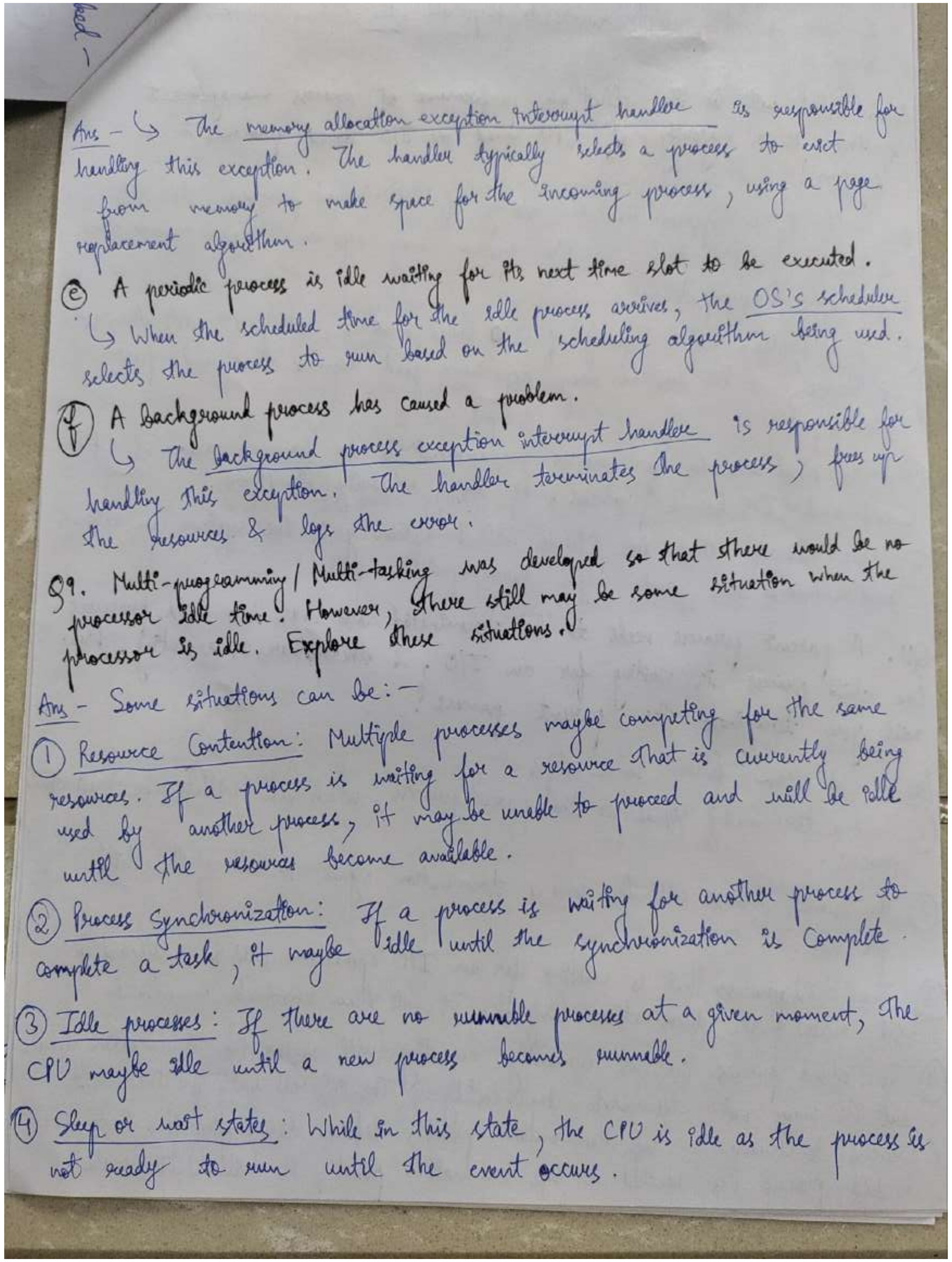
ⓑ The running process tries to access a memory location that it is not allowed to access.

↳ The memory access violation interrupt handler is responsible for handling this exception & takes appropriate action. The handler will terminate the offending process & free up any resources used by it.

ⓒ If there is a failure in reading or writing an I/O device.

↳ The I/O error interrupt handler is responsible for handling the exception & takes appropriate action. The handler will terminate the offending process & free up any resources used by it.

ⓓ A process is ready to execute but there is no space in the main memory.



Ans - ↳ The memory allocation exception interrupt handler is responsible for handling this exception. The handler typically selects a process to evict from memory to make space for the incoming process, using a page replacement algorithm.

(e) A periodic process is idle waiting for its next time slot to be executed.
↳ When the scheduled time for the idle process arrives, the OS's scheduler selects the process to run based on the scheduling algorithm being used.

(f) A background process has caused a problem.
↳ The background process exception interrupt handler is responsible for handling this exception. The handler terminates the process, frees up the resources & logs the error.

Q9. Multi-programming / Multi-tasking was developed so that there would be no processor idle time. However, there still may be some situation when the processor is idle. Explore these situations.

Ans - Some situations can be:-

① Resource Contention: Multiple processes maybe competing for the same resources. If a process is waiting for a resource that is currently being used by another process, it may be unable to proceed and will be idle until the resources become available.

② Process Synchronization: If a process is waiting for another process to complete a task, it maybe idle until the synchronization is complete.

③ Idle processes: If there are no runnable processes at a given moment, the CPU maybe idle until a new process becomes runnable.

④ Sleep or wait states: While in this state, the CPU is idle as the process is not ready to run until the event occurs.

(5) <u>System Overhead</u>: In some cases, the CPU maybe idle if there is not enough work to keep all the system processes busy.

Q10. What will be the effect on performance of process management if there is a majority of CPU-bound or I/O bound processes in a system?

<u>Ans</u> – (1) If a system has a majority of CPU-bound processes, the CPU utilization will be high, & the system may experience high response time and poor throughput.

(2) If a system has a majority of I/O bound processes, the CPU utilization will be lower, & the system may experience good response time and poor throughput.

(3) The ideal scenario for process management is to have a balance of CPU-bound and I/O bound processes, by using scheduling algorithms.

(4) This ensures that the CPU is utilised efficiently & the system can achieve good response time and high throughput.

Q11. A parent process needs to be terminated but has two child processes. One child process is waiting for an I/O, & the other is executing. How will you terminate the parent process?

<u>Ans</u> – We can follow some steps:–

(1) Send a terminating signal to the parent process, which will initiate the termination process.

(2) The parent process will then send a termination signal to both of its child processes

(3) The child process that is waiting for an I/O operation will be unblocked & will receive the termination signal. It will then terminate immediately.

(4) The child process that is currently executing will receive the termination signal, but it may not terminate immediately. The OS will wait for the child process to reach a safe termination point, before terminating it. Once the child process has reached a safe termination point, it will terminate.

<u>Ans</u>- (5) Once all

Ans- (5) Once all the child processes have been terminated, the parent process will be terminated.

Q12. How will you design the device queues where processes are waiting to access them? Can the priority of a process be incorporated here?

Ans - (1) One common design for device queues is a First-In-First-Out (FIFO) queue, where processes are added to the end of the queue as they arrive, & are removed from the front of the queue as the device becomes available.

Another design is a priority queue, where processes are assigned priorities based on their importance or urgency, & the highest priority process is granted access to the device first.

In addition to incorporating process priority, device queues can also be designed to handle other aspects of I/O device management, such as device scheduling, buffer management & error handling.

Q13. Context switch time is a pure overhead. What can be the factors that can increase this overhead?

Ans- Some factors can influence this :-

(1) Size of the process's context: The larger the amount of data that needs to be saved & restored for a process, the longer the context switch time.

(2) Number of processes running: The more processes that are running simultaneously, the more frequent the context switches, increasing the overhead.

(3) Type of processor: Different processors have different architectures that affect the context switch time. Some processors have specialised instructions that can speed up context switching.

(4) Operating System Design: An OS that uses a large no. of kernel-level interrupts may have a higher context switch time than an OS that uses user level interrupts.

⑤ <u>I/O operations :</u> I/O operations can also increase the context switch time, thus adding overhead to the system.

Q14. Consider Brain Teaser Problem 9 in Chapter 1 & illustrate the state transitions for all the processes in the system.

<u>Ans</u> - <u>In multiprogramming environment</u>,

```
⇒  | J1 | J2      | J3    | J1 | J3    | J1 | J3    | J1 |   → showing job state transitions.
   0    1         8       11   12      15   16      19   20
```

Let J_1 = time taken for Job 1 execution.
J_2 = time taken for Job 2 execution.
$J3$ = time taken for Job 3 execution.
∴ <u>total time for execution = 20 mins</u>

Q15. Again Consider brainteaser problem 9 in Chapter 1. Assume that there is a context switch time of 2 mins & then prepare a timing chart showing the CPU and I/O activities of the jobs in multi-programming with this context switch time. Compute the total time for execution.

<u>Ans</u> - <u>In multiprogramming environment</u>,

```
⇒  | J1 | CST | J2      | CST | J3    | CST | J1 | CST | J3    | CST | J1 | CST | J3    | CST | J1 |
   0    1     3         10    12      15    17   18    20      23    25   26    28      31    33   34
```

Let J1 = time taken for Job 1 execution.
J2 = time taken for Job 2 execution.
J3 = time taken for Job 3 execution.
CST = Context Switch Time b/w jobs.

∴ <u>total time for execution = 34 mins.</u>

Q16. In a system, Process A is running & Processes B, C are in a blocked state. An I/O Completion signal arrives for Process B. The priority of Process B is the highest. Describe the sequence of actions in the operating system.

Ans – Actions : –

① The OS will receive the I/O completion signal for Process B and update the state of Process B to reflect that it is now ready to run. However, since Process A is currently running, the OS cannot immediately switch to B.

② The OS will check the priorities of the processes in the system to determine which one should be given control of the CPU next. Therefore, Process B is selected to run next.

③ The OS will preempt process A, saving its state, & switch to process B. Since B was previously blocked, the OS will restore its state from point at which it was blocked and resume its execution.

④ Process B will continue executing until it either completes or becomes blocked again. If it completes, the OS will select another process to run based on their priority. If it becomes blocked again, the OS will repeat the process of selecting a new process to run based on priority.

⑤ If B completes, & there are no other processes waiting to run, the OS may put the CPU into an idle state until another process becomes ready.

⑥ If B becomes blocked again, & there are other processes waiting to run, the OS will repeat the process of selecting a new process to run based on priority.

Q17. In a system, Process A is running & Processes B, C are in a blocked state. Process D is ready to execute but not in the ready queue as there is no space. What will you do to accommodate this ready process D? Describe the sequence of actions in the OS.

Ans – Actions :

(1) The OS will check if there is any process in the ready queue that has a lower priority than D. If so, the OS will preempt that process & make space in the ready queue for D.

(2) If there is no process in the ready queue that has a lower priority than D, the OS will check if any process in the blocked state is waiting for a resource that is currently held by A. If so, the OS will preempt A, save its state, & unblock the waiting process. This will make space for D in ready queue.

(3) If there are no blocked processes waiting for a resource held by A, the OS will check if any blocked process has a higher priority than A. If so, the OS will preempt A & allow the blocked process to run. This will make space for D in ready queue.

(4) If none of the above options are available, the OS will wait until a process in the ready queue completes & then remove it from queue, creating space for D.

(5) Once there is space in ready queue, the OS will add D to the queue & mark it as ready to execute.

(6) The OS will then use its scheduling algorithm to select the next process to run from ready queue, which may be D if it has highest priority.

(7) If D is selected to run, the OS will restore its state & begin its execution.

(8) If another process is selected to run, the OS will restore its / save the state of D & put it back into ready queue, & then switch to the newly selected process.

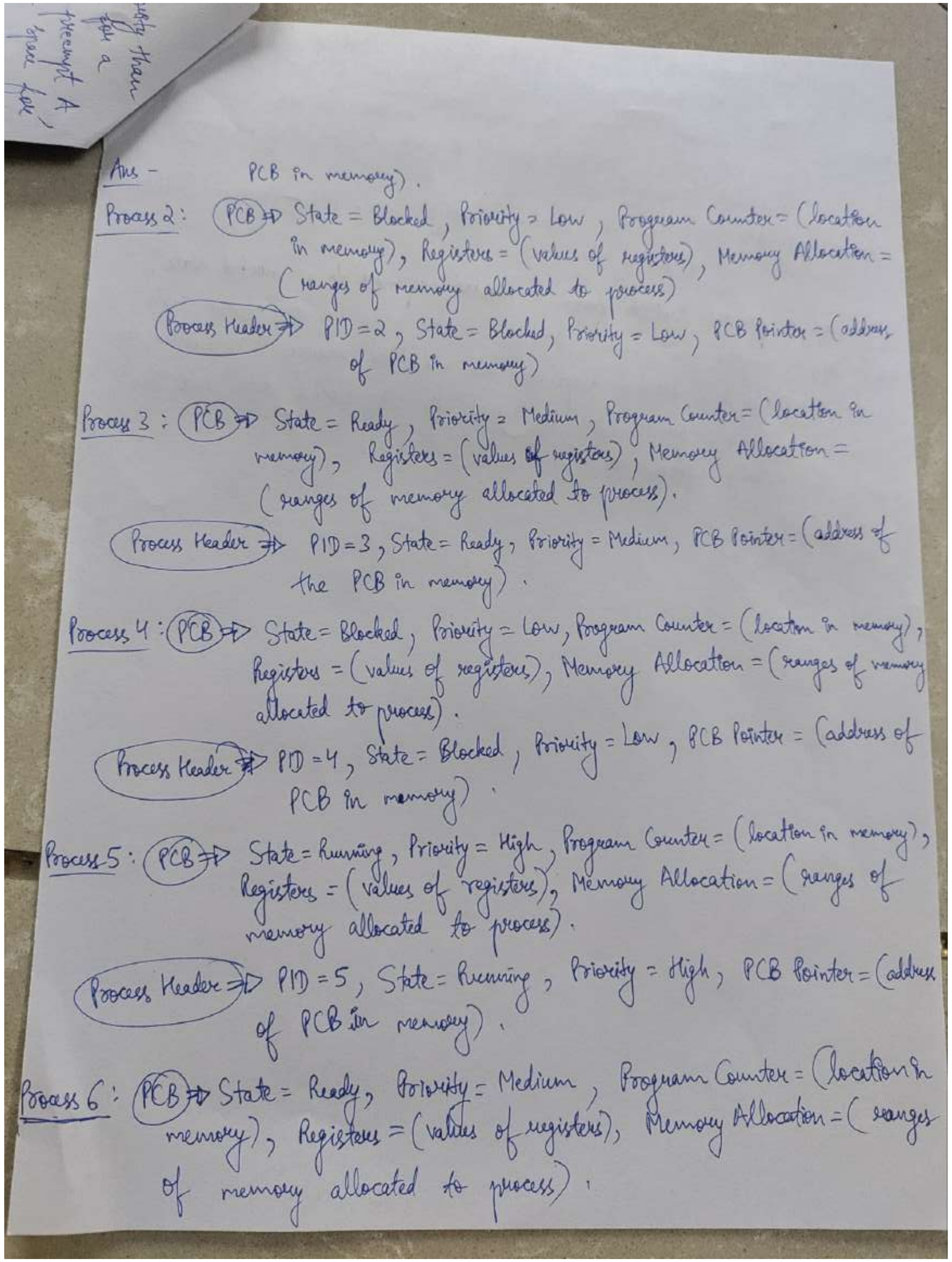
(9) The OS will continue selecting processes based on its scheduling algorithm.

Q18. In a system, Process 5 is running & Processes 2, 4, 7 are in a blocked state. Processes 1, 3, 6 are in ready state. Show the PCBS and process headers of all these processes in the memory.

Ans - <u>Assuming a basic OS</u> :-

Process 1: (PCB) ⇒ State = Ready, Priority = High, Registers = (values of registers), Program Counter = (location in memory), Memory Allocation = (ranges of memory allocated to process).

(Process Header) ⇒ PID = 1, State = Ready, Priority = High, PCB Pointer = (address of



than
for a
preempt A,
'grace fee'

Ans - PCB in memory).

Process 2: (PCB) ⇒ State = Blocked, Priority = Low, Program Counter = (location in memory), Registers = (values of registers), Memory Allocation = (ranges of memory allocated to process)

(Process Header) ⇒ PID = 2, State = Blocked, Priority = Low, PCB Pointer = (address of PCB in memory)

Process 3: (PCB) ⇒ State = Ready, Priority = Medium, Program Counter = (location in memory), Registers = (values ~~of~~ registers), Memory Allocation = (ranges of memory allocated to process).

(Process Header) ⇒ PID = 3, State = Ready, Priority = Medium, PCB Pointer = (address of the PCB in memory).

Process 4: (PCB) ⇒ State = Blocked, Priority = Low, Program Counter = (location in memory), Registers = (values of registers), Memory Allocation = (ranges of memory allocated to process).

(Process Header) ⇒ PID = 4, State = Blocked, Priority = Low, PCB Pointer = (address of PCB in memory).

Process 5: (PCB) ⇒ State = Running, Priority = High, Program Counter = (location in memory), Registers = (values of registers), Memory Allocation = (ranges of memory allocated to process).

(Process Header) ⇒ PID = 5, State = Running, Priority = High, PCB Pointer = (address of PCB in memory).

Process 6: (PCB) ⇒ State = Ready, Priority = Medium, Program Counter = (location in memory), Registers = (values of registers), Memory Allocation = (ranges of memory allocated to process).

Process Header ⟹ PID=6, State = Ready, Priority = Medium,
PCB Pointer = (address of PCB in memory).

Process 7: (PCB⟹) State = Blocked, Priority = Low, Program Counter = (location in memory), Registers = (values of registers), Memory Allocation = (ranges of memory allocated to process).

Process Header ⟹ PID = 7, State = Blocked, Priority = Low, PCB – Pointer = (address of PCB in memory).

Q19. Can a process switch from a blocked-suspended to a blocked state?

Ans – Yes, a process can switch from a blocked-suspended to blocked state. If a process is in a blocked-suspended state and the resource it is waiting for becomes available, process can be moved to the blocked state until the resource is actually allocated to it.