Operating System (315319)

Model Question Paper — Set B · Model Answers

Q.1

a) States of a process. 2 M

1. **New** — process is being created.
2. **Ready** — waiting in main memory to get the CPU.
3. **Running** — instructions are being executed.
4. **Waiting / Blocked** — waiting for an I/O or an event.
5. **Terminated** — process has finished execution.

(Any four states, ½ M each)

b) File management system calls. 2 M

1. **create()** / open() — create or open a file.
2. **read()** — read data from a file.
3. **write()** — write data into a file.
4. **close()** — close an opened file.
5. **delete / unlink()**, lseek() (reposition), get/set file attributes (stat(), chmod()).

(Any four, ½ M each)

c) Turnaround time and response time. 2 M

i) Turnaround time: The interval from the time of submission (arrival) of a process to the time of its completion. TAT = Completion time − Arrival time. It includes waiting time, execution time and I/O time.

ii) Response time: The time from the submission of a request until the **first response** is produced (i.e. the process gets the CPU for the first time), not the time to output the full result.

d) Preemptive vs non-preemptive scheduling. 2 M

Preemptive scheduling	Non-preemptive scheduling
CPU can be taken away from a running process (on time-slice expiry or arrival of a higher-priority process).	Once the CPU is given to a process, it keeps it until it terminates or goes to the waiting state.
Better response time; suitable for time-sharing and real-time systems.	Simple, low overhead; a long process can make others wait.
Examples: Round Robin, SRTN, preemptive priority.	Examples: FCFS, SJF (non-preemptive), non-preemptive priority.

(Any two points, 1 M each)

e) Fragmentation and its types. 2 M

Fragmentation: When processes are loaded and removed from memory, the free memory is broken into small pieces that cannot be used effectively; this wastage of memory is called fragmentation.

Types: 1) **Internal fragmentation** — unused space *inside* an allocated partition/block (allocated memory is larger than requested). 2) **External fragmentation** — enough total free memory exists, but it is not contiguous, so a request cannot be satisfied.

f) File operations. 2 M

1. **Create** a file
2. **Write** a file
3. **Read** a file
4. **Reposition** within a file (seek)
5. **Delete** a file
6. **Truncate** a file; also open, close, append, rename

(Any four, ½ M each)

g) Purpose of ps command and options. 2 M

ps (process status) displays information about the active processes — PID, terminal (TTY), CPU time and command name. It gives a snapshot of current processes.

Option	Meaning
ps -e (or -A)	show all processes of the system
ps -f	full format listing (UID, PID, PPID, C, STIME, TTY, TIME, CMD)
ps -u username	show processes of the given user
ps -l	long format (state, priority, nice value …)

(Purpose 1 M, any two options 1 M)

Q.2

a) Time-sharing OS vs real-time OS. 4 M

Time-sharing OS	Real-time OS
CPU time is shared among many interactive users using time slices (Round Robin).	Tasks must be completed within strict time limits (deadlines).
Main aim: quick response to every user and fair sharing of CPU.	Main aim: meeting deadlines; predictable, deterministic behaviour.
Missing a short delay is acceptable.	Missing a deadline may cause failure (hard RTOS).
Used for general-purpose multi-user computing.	Used in embedded / control systems.
Example: UNIX, Linux multi-user systems.	Example: VxWorks, FreeRTOS, QNX; air-bag, missile control.

(Any four points, 1 M each)

b) OS activities: file management and I/O management. 4 M

i) File management — the OS is responsible for:

- Creating and deleting files.
- Creating and deleting directories to organise files.
- Supporting primitives for manipulating files and directories (read, write, rename, copy).
- Mapping files onto secondary storage (allocation of disk blocks).
- Backing up files on stable (non-volatile) storage media; access control / protection of files.

ii) I/O system management — the OS hides the details of hardware devices from the user and is responsible for:

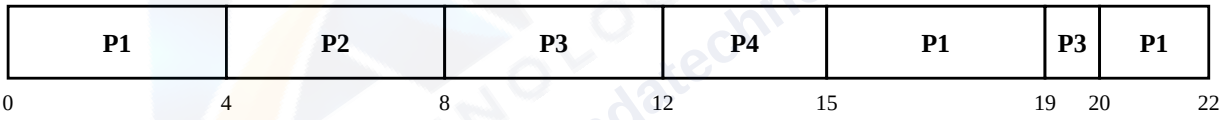
- A memory-management component including **buffering, caching and spooling**.
- A general **device-driver interface**.
- **Drivers** for specific hardware devices.
- Allocating and de-allocating devices to processes; handling interrupts from devices.

(Each component 2 M)

c) Round Robin scheduling (q = 4 ms). 4 M

Round Robin (RR) is a preemptive scheduling algorithm designed for time-sharing systems. Each process gets the CPU for a small fixed time called **time quantum (time slice)**. If the process does not finish within the quantum, it is preempted and put at the **tail of the ready queue** (circular FIFO queue). The CPU then goes to the next process. No process waits more than (n − 1) × q time units for its next turn.

Round Robin (time quantum = 4 ms)



Gantt chart

Process	BT	Completion time (CT)	TAT = CT	WT = TAT − BT
P1	10	22	22	22 − 10 = 12
P2	4	8	8	8 − 4 = 4
P3	5	20	20	20 − 5 = 15
P4	3	15	15	15 − 3 = 12

Average waiting time = 43 / 4 = **10.75 ms**

(Explanation 2 M, Gantt chart and average WT 2 M)

d) Two-level directory structure. 4 M

In a **two-level directory**, each user has a separate **User File Directory (UFD)** that lists only that user's files. The system has a **Master File Directory (MFD)** which is indexed by user name (or

account number); each entry points to the UFD of that user. When a user refers to a file, only his own UFD is searched. A file is identified by path name: /user/file.

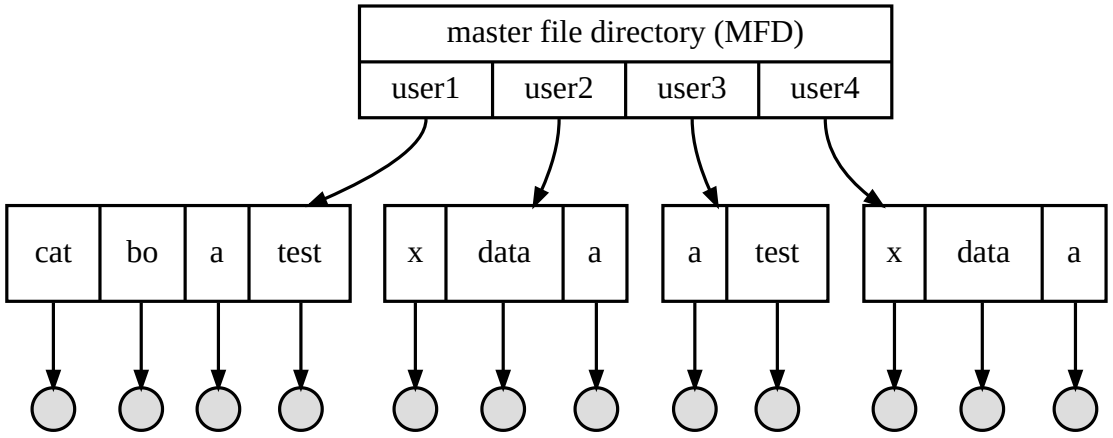


Fig.: Two-level directory structure (MFD → user file directories → files)

Advantages: 1) Different users can have files with the **same name** (e.g. 'a' in user1, user3, user4). 2) Searching is efficient — only one UFD is searched. 3) Isolates users from one another, giving protection/privacy.

Disadvantages: users cannot group their own files (no sub-directories); sharing files between users is difficult.

(Explanation 2 M, diagram 1 M, advantages 1 M)

Q.3

a) Scheduler; long-term and short-term schedulers. 4 M

Scheduler: A scheduler is the part of the OS that **selects processes** from the various scheduling queues (job queue, ready queue, device queues) according to some policy. Types: long-term, short-term and medium-term scheduler.

Long-term scheduler (job scheduler)	Short-term scheduler (CPU scheduler)
Selects processes from the job pool on disk and loads them into memory (ready queue).	Selects a process from the ready queue and allocates the CPU to it.
Controls the degree of multiprogramming.	Controls which ready process runs next.
Executes infrequently (minutes); can be slower.	Executes very frequently (every few ms); must be very fast.
Should select a good mix of I/O-bound and CPU-bound processes.	Works with the dispatcher, which does the context switch.
May be absent / minimal in time-sharing systems.	Present in all multiprogramming systems.

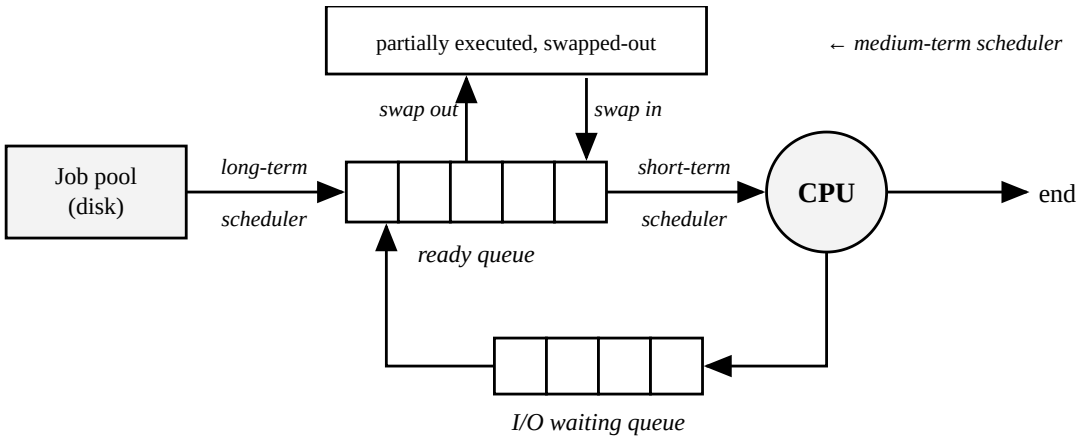


Fig.: Queueing diagram showing long-term, short-term and medium-term schedulers

(Definition 1 M, each scheduler 1½ M)

b) User-level vs kernel-level threads.

4 M

User-level threads	Kernel-level threads
Created and managed by a thread library in user space; kernel is not aware of them.	Created and managed directly by the operating system (kernel).
Creation and switching are fast (no system call / mode switch).	Creation and switching are slower (need system calls).
If one thread makes a blocking system call, the whole process is blocked.	If one thread blocks, the kernel can schedule another thread of the same process.
Cannot run in parallel on multiple CPUs (kernel sees one process).	Threads of one process can run in parallel on multiprocessors.
Portable; can run on any OS.	Specific to the operating system.
Example: POSIX Pthreads (user library), Java green threads.	Example: Windows threads, Linux kernel threads.

(Any four points, 1 M each)

c) Virtual memory.

4 M

Virtual memory is a technique that allows the execution of processes that are **not completely in main memory**. It separates the logical (user) memory from physical memory, so a program can be larger than physical memory. Only the needed pages are kept in RAM; the remaining pages stay on the disk (backing store) and are brought in on demand (**demand paging**). A reference to a page not in memory causes a **page fault**.

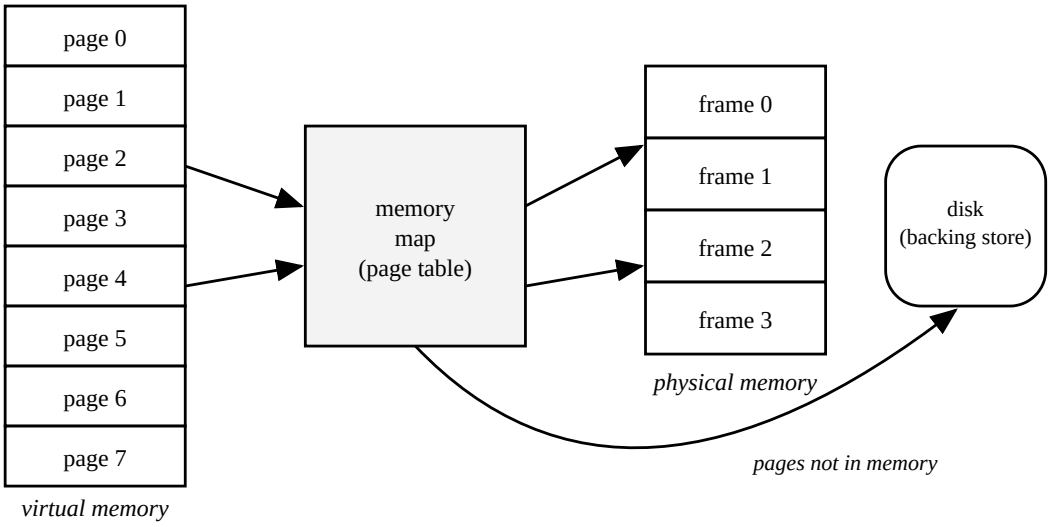


Fig.: Virtual memory larger than physical memory

Advantages:

- A program is no longer limited by the size of physical memory.
- More programs can run at the same time — higher degree of multiprogramming, better CPU utilisation and throughput.
- Less I/O is needed to load or swap programs, so programs start faster.
- Allows sharing of files and memory (shared libraries) between processes.

(Explanation 2 M, diagram 1 M, advantages 1 M)

d) Sequential and direct access methods. 4 M

1) Sequential access: Information in the file is processed **in order, one record after the other**. A read operation (read next) reads the next record and advances the file pointer; write next appends at the end. The file can be reset (rewound) to the beginning. It is based on the tape model. **Example:** editors and compilers reading source files; reading a log file; audio on a cassette tape.

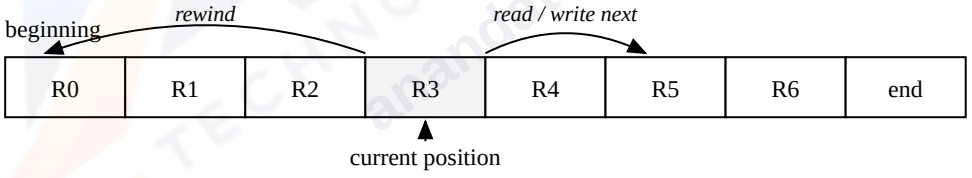


Fig.: Sequential-access file

2) Direct (relative) access: The file is viewed as a numbered sequence of fixed-length **blocks / records**, and any block can be read or written **directly** in any order, without reading the previous ones (read n, write n, where n is the relative block number). It is based on the disk model. **Example:** database systems; airline-reservation system — the record of flight 713 is read directly from block 713.

(Each method with example 2 M)

Q.4

a) Variable (dynamic) partitioning. 4 M

In **variable partitioning**, memory is not divided in advance. Partitions are created **dynamically at load time**, and each process is given exactly as much memory as it requires. Initially all user memory is one large free block (**hole**). When a process arrives, a large enough hole is found and allocated; when it terminates, its memory becomes a hole and adjacent holes are merged. The OS keeps a table of allocated and free partitions and uses first fit / best fit / worst fit to choose a hole.

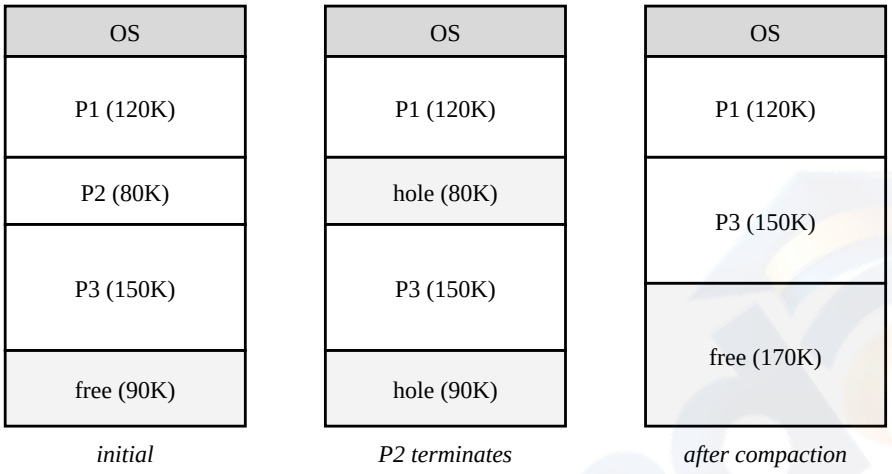


Fig.: Variable partitioning — a 160K process cannot be loaded until compaction

No internal fragmentation occurs, but after many loads and exits free memory is split into small holes — **external fragmentation**.

Removing external fragmentation: 1) **Compaction** — shuffle the memory contents so that all free memory is placed together in one large block (possible only if relocation is dynamic, done at execution time). 2) Allow the logical address space to be non-contiguous — **paging** and **segmentation**.

(Explanation 2 M, diagram 1 M, removal 1 M)

b) Deadlock prevention: hold-and-wait and circular wait. 4 M

Deadlock prevention ensures that at least one of the four necessary conditions can never hold.

i) Denying hold and wait: Whenever a process requests a resource, it must not hold any other resource. Two protocols: (a) a process must **request and be allocated all its resources before it begins execution**; (b) a process may request resources only when it holds **none** — it must release all held resources before requesting more. *Drawbacks:* low resource utilisation and possible starvation. Example: a process that copies from DVD to disk and then prints must request DVD, disk and printer at the start.

ii) Denying circular wait: Impose a **total ordering** on all resource types — give each type a unique number, e.g. F(tape) = 1, F(disk) = 5, F(printer) = 12 — and require every process to request resources only in **increasing order** of these numbers. A process holding resource Rj can request Ri only if F(Ri) > F(Rj). Then a cycle of waiting processes can never form, so circular wait is impossible.

(Each method 2 M)

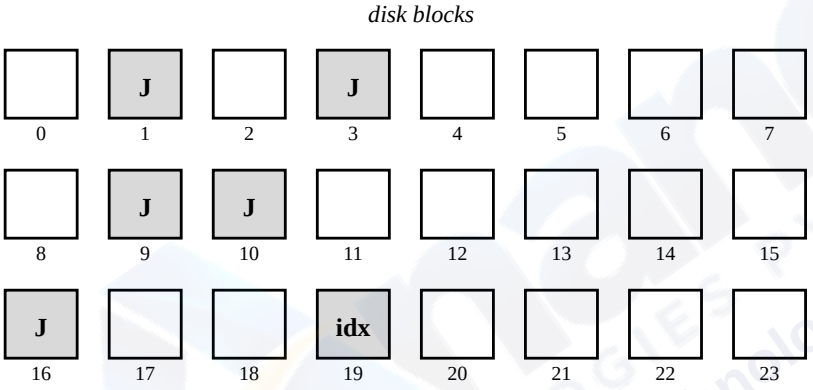
c) Four CPU scheduling criteria. 4 M

1. **CPU utilisation:** keep the CPU as busy as possible (40% to 90% in real systems) — maximise.
2. **Throughput:** number of processes completed per unit time — maximise.
3. **Turnaround time:** time from submission to completion of a process (TAT = CT − AT) — minimise.
4. **Waiting time:** total time spent waiting in the ready queue (WT = TAT − BT) — minimise.
5. **Response time:** time from submission of a request until the first response is produced — minimise (important for interactive systems).

(Any four, 1 M each)

d) Indexed allocation; difference from linked allocation. 4 M

In **indexed allocation**, all the pointers (block addresses) of a file are brought together into one block called the **index block**. Each file has its own index block; the i-th entry of the index block points to the i-th block of the file. The directory entry contains the address of the index block. It supports direct access without external fragmentation.



File	Index block
jeep	19

index block 19:

entry	block
0	9
1	16
2	1
3	10
4	3

Fig.: Indexed allocation of file 'jeep'

Linked allocation	Indexed allocation
Each block contains a pointer to the next block of the file.	All pointers are kept together in an index block.
Directory stores start and end block.	Directory stores the address of the index block.
Supports only sequential access efficiently.	Supports both sequential and direct access.
If one pointer is lost/damaged, the rest of the file is lost.	Index block damage affects the file, but blocks are independent.
No index overhead; pointer space in every block.	Extra index block per file (wasted for small files).

(Explanation with diagram 2 M, differences 2 M)

e) Bitmap free-space management. 4 M

In the **bitmap (bit vector)** method, each disk block is represented by **one bit**: if the block is **free, the bit is 1**; if the block is **allocated, the bit is 0** (standard textbook convention). The bitmap is

kept in memory; to find a free block the OS scans for the first 1 bit (the first non-zero word). Advantage: simple and efficient to find the first free block or n consecutive free blocks. Disadvantage: the whole bit vector must be kept in main memory, which is large for big disks.

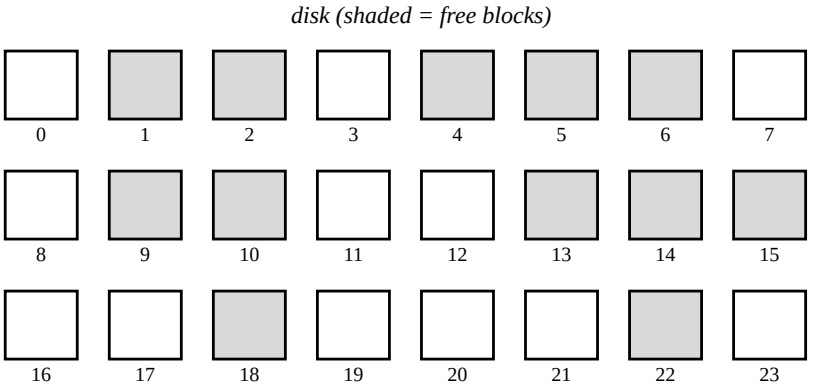


Fig.: Free blocks on the disk

Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Bit	0	1	1	0	1	1	1	0	0	1	1	0	0	1	1	1	0	0	1	0	0	0	1	0

Bit vector: 01101110 01100111 00100010

(Some books use the opposite convention, 0 = free; then every bit is complemented: 100100011001100011011101.)

(Explanation 2 M, bit vector 2 M)

Q.5

a) Six services of operating system. 6 M

1. **User interface:** OS provides a command-line interface (CLI, e.g. bash shell), batch interface, or graphical user interface (GUI, e.g. Windows desktop) to interact with the system.
2. **Program execution:** the system loads a program into memory and runs it; the program ends normally or abnormally (with an error indication). Example: running a.out.
3. **I/O operations:** a running program may need I/O from a file or device (keyboard, printer). Users cannot control I/O devices directly, so the OS performs I/O for them.
4. **File-system manipulation:** programs need to read, write, create, delete, search files and directories, list file information and manage permissions.
5. **Communication:** processes exchange information on the same computer or across a network, by shared memory or message passing.
6. **Error detection:** OS detects errors in CPU, memory, I/O devices (e.g. parity error, printer out of paper) or user programs (divide by zero, illegal memory access) and takes proper action.
7. **Resource allocation, accounting, protection and security** — allocating CPU, memory and files to multiple users/jobs, keeping usage records, and controlling access (passwords).

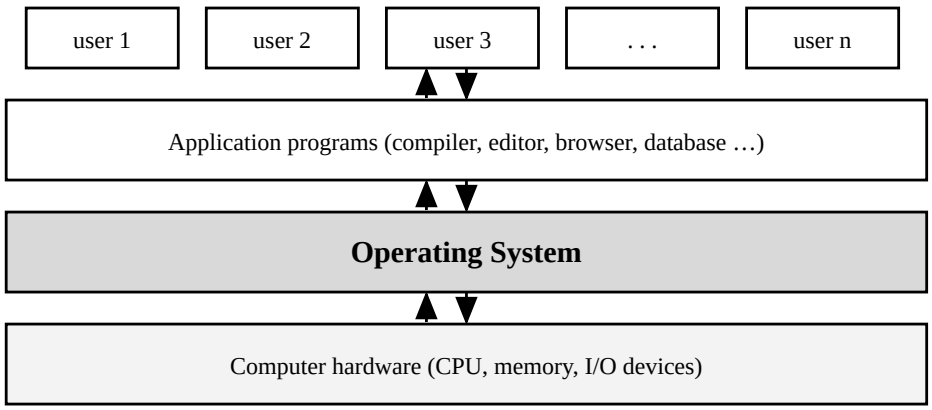


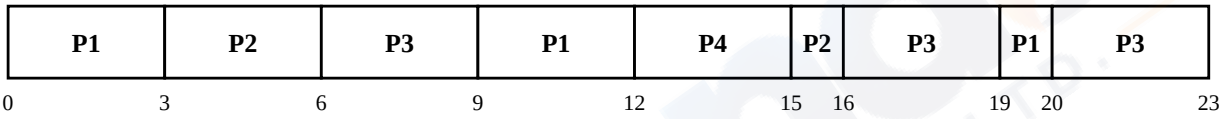
Fig.: Abstract view of a computer system

(Any six services, 1 M each)

b) Round Robin (q = 3) and SRTN. 6 M

Round Robin (time quantum = 3 ms)

At t = 3, P2 and P3 have arrived; they join the ready queue before the preempted P1. A newly arrived process is placed in the queue before the process whose time slice has just expired.



Gantt chart

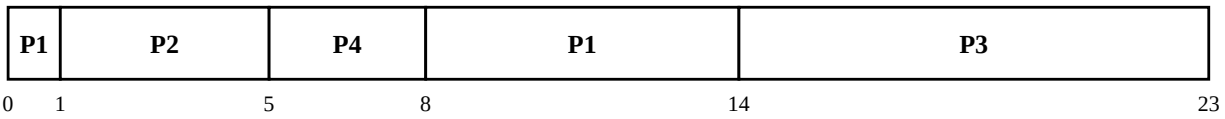
Process	AT	BT	Completion time (CT)	TAT = CT − AT	WT = TAT − BT
P1	0	7	20	20 − 0 = 20	20 − 7 = 13
P2	1	4	16	16 − 1 = 15	15 − 4 = 11
P3	2	9	23	23 − 2 = 21	21 − 9 = 12
P4	4	3	15	15 − 4 = 11	11 − 3 = 8

Average turnaround time = 67 / 4 = **16.75 ms**

Average waiting time = 44 / 4 = **11 ms**

SRTN (Shortest Remaining Time Next, preemptive SJF)

At every arrival, the process with the shortest remaining time gets the CPU (P1 is preempted at t = 1 because P2 needs 4 < 6).



Gantt chart

Process	AT	BT	Completion time (CT)	TAT = CT − AT	WT = TAT − BT
P1	0	7	14	14 − 0 = 14	14 − 7 = 7

P2	1	4	5	5 − 1 = 4	4 − 4 = 0
P3	2	9	23	23 − 2 = 21	21 − 9 = 12
P4	4	3	8	8 − 4 = 4	4 − 3 = 1

Average turnaround time = 43 / 4 = **10.75 ms**

Average waiting time = 20 / 4 = **5 ms**

Algorithm	Average WT (ms)	Average TAT (ms)
RR (q = 3)	11	16.75
SRTN	5	10.75

(Each algorithm: Gantt chart 1 M, table 1 M, averages 1 M)

c) Paging with paging hardware. 6 M

Paging is a non-contiguous memory-management scheme that allows the physical address space of a process to be non-contiguous. **Physical memory** is divided into fixed-size blocks called **frames**, and **logical memory** into blocks of the same size called **pages** (size is a power of 2, e.g. 1 KB – 8 KB). When a process is executed, its pages are loaded into any available frames. A **page table** for each process stores the frame number of each page.

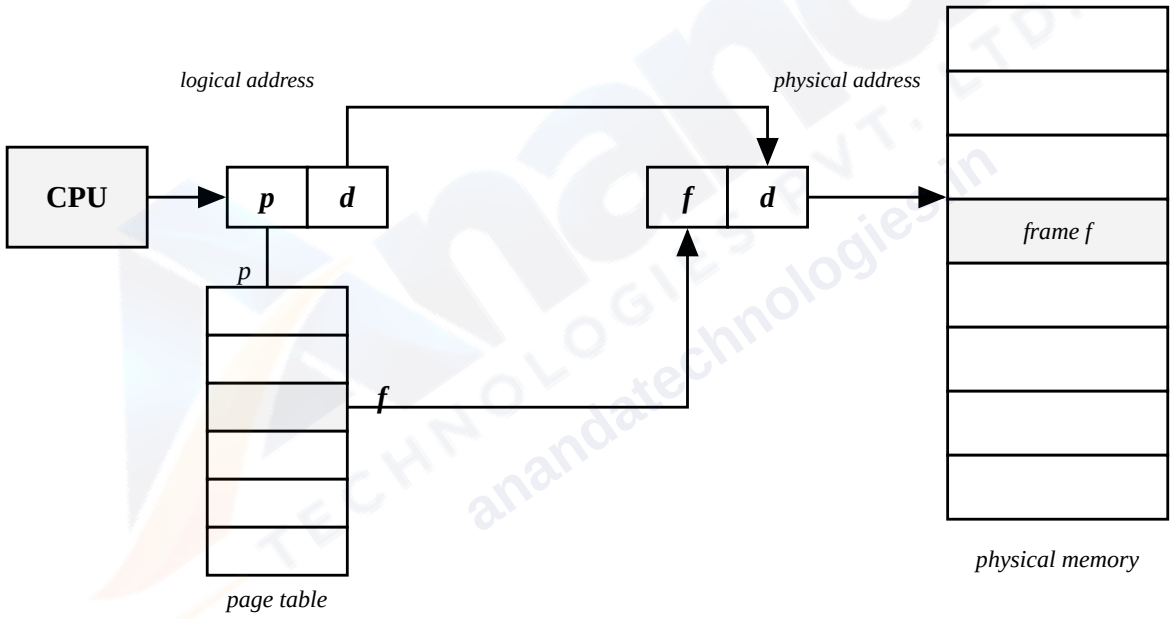


Fig.: Paging hardware

Working (address translation):

1. Every address generated by the CPU (logical address) is divided into two parts: **page number (p)** and **page offset (d)**.
2. The page number p is used as an index into the **page table**, which gives the base address (**frame number f**) of that page in physical memory.
3. The frame number f is combined with the offset d to form the **physical address** (f, d) = f × page size + d, which is sent to the memory unit.
4. Paging has no external fragmentation; the last page may have internal fragmentation.

Example: page size = 1024 bytes, logical address = 2500.
Page number p = 2500 div 1024 = **2**; offset d = 2500 mod 1024 = **452**.
Page 2 is in frame 5, so physical address = 5 × 1024 + 452 = **5572**.

(Explanation 2 M, diagram 2 M, working / example 2 M)

Q.6

a) Distributed OS and mobile OS. 6 M

Distributed operating system: A distributed system is a collection of **physically separate, possibly heterogeneous computers** (nodes/sites) connected by a **communication network**, which appears to its users as a single coherent system. Each node has its own processor and local memory (loosely coupled); they communicate through the network (LAN/WAN). The distributed OS manages resources across all nodes and lets users access remote resources as if they were local.

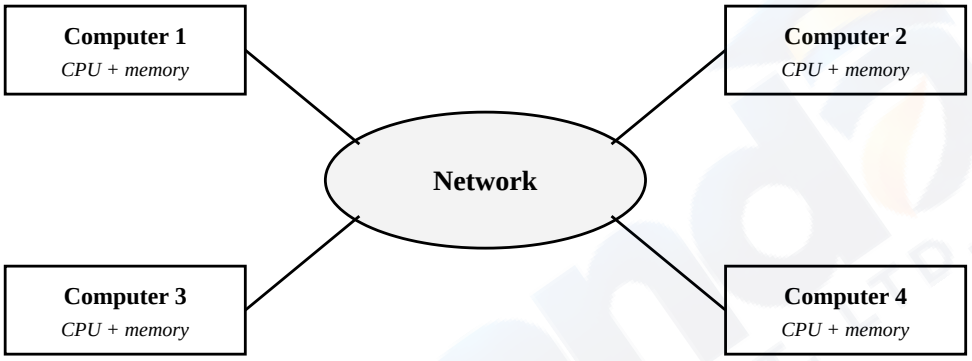


Fig.: Distributed operating system

Advantages: 1) **Resource sharing** — files, printers, data on one site can be used from another. 2) **Computation speed-up** — a job is divided and run in parallel on several sites (load sharing). 3) **Reliability** — if one site fails, the others continue. 4) **Communication** — users at different sites exchange information (mail, file transfer). **Examples:** LOCUS, Amoeba; cluster / cloud systems.

Mobile operating system: An OS designed for smartphones, tablets and other hand-held devices. It manages the device hardware (touch screen, camera, GPS, sensors, cellular/Wi-Fi radio) and provides features such as touch-based GUI, app store, power (battery) management, limited-memory management, notifications and security/sandboxing of apps.

- **Android** (Google) — open-source, based on the Linux kernel; apps written in Java/Kotlin; used by many manufacturers (Samsung, Xiaomi …).
- **iOS** (Apple) — proprietary, used only on iPhone/iPad; based on Darwin (UNIX-like) kernel; apps written in Swift/Objective-C.
- Others: HarmonyOS, KaiOS.

(Distributed OS with diagram and advantages 3 M, mobile OS with examples 3 M)

b) LRU and Optimal page replacement (3 frames). 6 M

Reference string (15 references): 6, 1, 2, 1, 4, 6, 3, 1, 2, 4, 6, 2, 3, 1, 4; frames = 3.

LRU (Least Recently Used)

Rule: replace the page that has not been used for the longest time in the past.

Ref.	6	1	2	1	4	6	3	1	2	4	6	2	3	1	4
F1	6	6	6	6	4	4	4	1	1	1	6	6	6	1	1
F2		1	1	1	1	1	3	3	3	4	4	4	3	3	3
F3			2	2	2	6	6	6	2	2	2	2	2	2	4
Fault	F	F	F	H	F	F	F	F	F	F	F	H	F	F	F

F = page fault, H = hit. **Number of page faults (LRU) = 13** (hits = 2); page fault ratio = 13/15 = 0.87

Optimal (OPT)

Rule: replace the page that will not be used for the longest time in future (if several pages are never used again, the oldest of them is replaced).

Ref.	6	1	2	1	4	6	3	1	2	4	6	2	3	1	4
F1	6	6	6	6	6	6	3	3	3	3	3	3	3	1	1
F2		1	1	1	1	1	1	1	2	2	2	2	2	2	4
F3			2	2	4	4	4	4	4	4	6	6	6	6	6
Fault	F	F	F	H	F	H	F	H	F	H	F	H	H	F	F

F = page fault, H = hit. **Number of page faults (OPT) = 9** (hits = 6); page fault ratio = 9/15 = 0.60

Result: LRU = 13 page faults; Optimal = 9 page faults. Optimal gives the minimum number of page faults.

(Each algorithm 3 M)

c) Banker's algorithm — safe sequence. 6 M

Step 1: Need matrix (Need = Max − Allocation)

Process	Allocation (A B C)	Max (A B C)	Need (A B C)
P0	1 1 2	4 3 3	3 2 1
P1	2 1 2	3 2 2	1 1 0
P2	4 0 1	9 0 2	5 0 1
P3	0 2 0	7 5 3	7 3 3
P4	1 1 2	3 2 3	2 1 1

Step 2: Safety algorithm. Work = Available = (2, 1, 0); Finish[i] = false for all i. Find a process with Finish = false and Need ≤ Work; then Work = Work + Allocation, Finish = true. Repeat.

Process	Need ≤ Work	Result	New Work = Work + Allocation
P0	(3, 2, 1) ≤ (2, 1, 0) ?	No — wait	—
P1	(1, 1, 0) ≤ (2, 1, 0) ?	Yes	(2, 1, 0) + (2, 1, 2) = (4, 2, 2)
P2	(5, 0, 1) ≤ (4, 2, 2) ?	No — wait	—
P3	(7, 3, 3) ≤ (4, 2, 2) ?	No — wait	—

P4	(2, 1, 1) ≤ (4, 2, 2) ?	Yes	(4, 2, 2) + (1, 1, 2) = (5, 3, 4)
P0	(3, 2, 1) ≤ (5, 3, 4) ?	Yes	(5, 3, 4) + (1, 1, 2) = (6, 4, 6)
P2	(5, 0, 1) ≤ (6, 4, 6) ?	Yes	(6, 4, 6) + (4, 0, 1) = (10, 4, 7)
P3	(7, 3, 3) ≤ (10, 4, 7) ?	Yes	(10, 4, 7) + (0, 2, 0) = (10, 6, 7)

All processes can finish. **The system is in a SAFE state. Safe sequence: <P1, P4, P0, P2, P3>**

(Need matrix 2 M, safety steps 3 M, safe sequence 1 M)

Note: This is a **model question paper prepared by Ananda Technologies** only for practice and exam preparation, based on the pattern of previous MSBTE question papers and the K-scheme syllabus. It is **not an official MSBTE paper**, and it does **not** mean that these questions will come in the exam. Answers follow standard textbook points; marks shown are indicative. Free for students — anandatechnologies.in