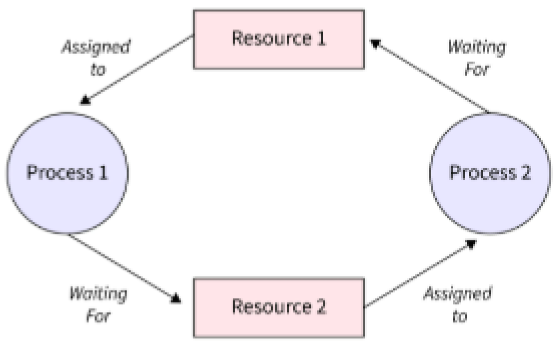
Deadlocks

Define Deadlock.

A deadlock is a situation in an operating system where two or more processes get stuck. Each process is holding a resource and waiting for another resource that is already being used by another process. As a result, none of them can move forward or finish their tasks.



What are the necessary conditions for a deadlock to occur? Name these conditions and provide an elaboration.

1. **Mutual Exclusion**: One resource must be used by only one process at a time. If another process wants that same resource, it has to wait until the first process is done and releases it.
2. **Hold and Wait**: A process is already using one or more resources and is also waiting to get more resources that are currently being used by other processes.
3. **No Preemption**: Resources can't be taken away by force. A process can release a resource only when it wants to — usually after finishing its task.
4. **Circular Wait**: There must be a set of processes where each one is waiting for a resource that another process in the same set is holding. For example, P1 is waiting for a resource held by P2, P2 is waiting for one held by P3, and so on, until one process is waiting for a resource held by P1 — forming a circle.

Describe different recovery techniques from deadlock.

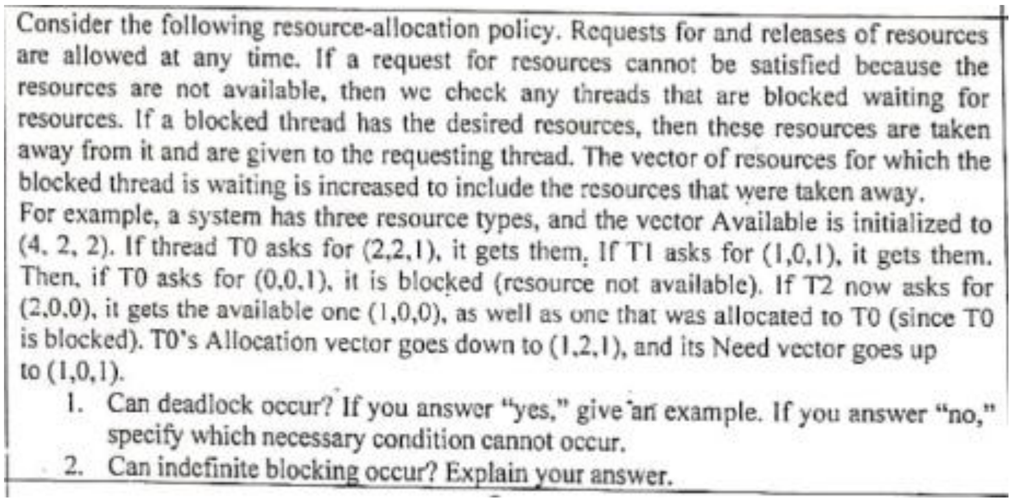
There are four primary methods of deadlock recovery: process termination, resource preemption, priority inversion, and rollback.

1. Process Termination: The system stops one or more of the deadlocked processes to fix the problem. This frees up the resources they were using so other processes can continue. But it can cause loss of data or unexpected behavior since the process ends suddenly.

2. Resource Preemption: The system takes resources away from some running processes and gives them to others to break the deadlock. The affected process is paused until it can get the needed resources again. This can slow things down and make resource use less efficient.

3. Priority Inversion: Here, the system temporarily gives higher priority to the process holding a needed resource so it can finish faster and release the resource. This helps avoid deadlock. But it might slow down other lower-priority processes and cause delays.

4. Rollback: The system moves one or more processes back to a previous safe state before the deadlock happened. It does this using saved logs or checkpoints, then restarts the process from there. This can take time and may lose some data if not done properly.


Consider the following resource-allocation policy. Requests for and releases of resources are allowed at any time. If a request for resources cannot be satisfied because the resources are not available, then we check any threads that are blocked waiting for resources. If a blocked thread has the desired resources, then these resources are taken away from it and are given to the requesting thread. The vector of resources for which the blocked thread is waiting is increased to include the resources that were taken away.
For example, a system has three resource types, and the vector Available is initialized to (4, 2, 2). If thread T0 asks for (2,2,1), it gets them. If T1 asks for (1,0,1), it gets them. Then, if T0 asks for (0,0,1), it is blocked (resource not available). If T2 now asks for (2,0,0), it gets the available one (1,0,0), as well as one that was allocated to T0 (since T0 is blocked). T0's Allocation vector goes down to (1,2,1), and its Need vector goes up to (1,0,1).

1. Can deadlock occur? If you answer "yes," give an example. If you answer "no," specify which necessary condition cannot occur.
2. Can indefinite blocking occur? Explain your answer.

1) No, **deadlock cannot occur**. The policy violates the "no preemption" necessary condition for deadlock. In deadlock, the four necessary conditions are:

1. Mutual exclusion
2. Hold and wait
3. No preemption
4. Circular wait

In this policy, if a process asks for a resource that isn't available, the system can take that resource away from a blocked process and give it to the one that's waiting. So, resources can be forcibly taken back, which breaks the no preemption rule. Thus, deadlock is impossible.

2) Yes, **indefinite blocking can occur**. Indefinite blocking (starvation) is possible because a thread might repeatedly lose its resources to other threads and never make progress. For example:

- Suppose thread T0 is blocked and its resources are taken by T1.

- Later, T1 releases resources, but before T0 can acquire them, another thread T2 requests and takes them.
- This could repeat indefinitely, with T0 always being preempted by other threads.

Memory Management

Define contiguous memory allocation and explain its main advantages and disadvantages. Provide examples to illustrate external and internal fragmentation.

Contiguous memory allocation is a memory management technique where each process is allocated a single continuous block of memory in the main memory (RAM).

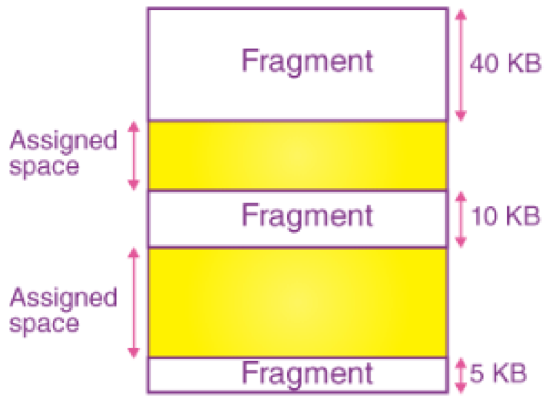
Advantages:	**Disadvantages:**
1. It is simple to set up and manage. 2. It allows quick access because the data is stored in order. 3. It gives better speed and performance with less delay.	1. It wastes memory due to scattered free space (external fragmentation). 2. It is hard to increase the program size after memory is assigned. 3. Some parts of memory may stay unused. 4. It is not flexible for programs that need changing memory sizes.

External Fragmentation: This happens when free memory is available but split into small pieces scattered across the system. Even though the total free memory is enough, none of the pieces are big enough by themselves to fit a process. Example:

- A Process needs 50KB of contiguous memory space

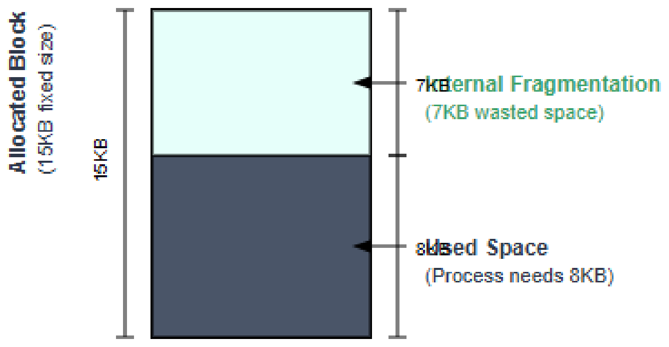
- Available free fragments: 40KB + 10KB + 5KB = 55KB total
- Even though there's enough total free memory (55KB > 50KB), the process cannot be allocated because no single block is large enough. The largest available fragment is only 40KB, which is insufficient for the 50KB requirement

This is called external fragmentation because the free space is broken up outside of the used blocks and can’t be joined together easily.



External Fragmentation

Internal Fragmentation Example



Internal Fragmentation: This happens when the system gives more memory to a process than it really needs, and the remaining space inside that block is wasted. Example:

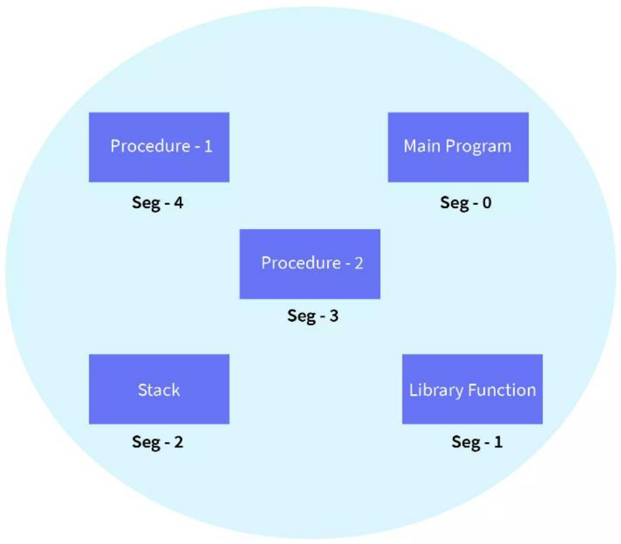
- Suppose the system always gives memory in fixed-size blocks of 15KB.
- A small process needs only 8KB, but it still gets the full 15KB block.
- That means 7KB is wasted inside the block (15KB - 8KB).

This unused space is called internal fragmentation because it’s inside the allocated memory block and can’t be used by others.

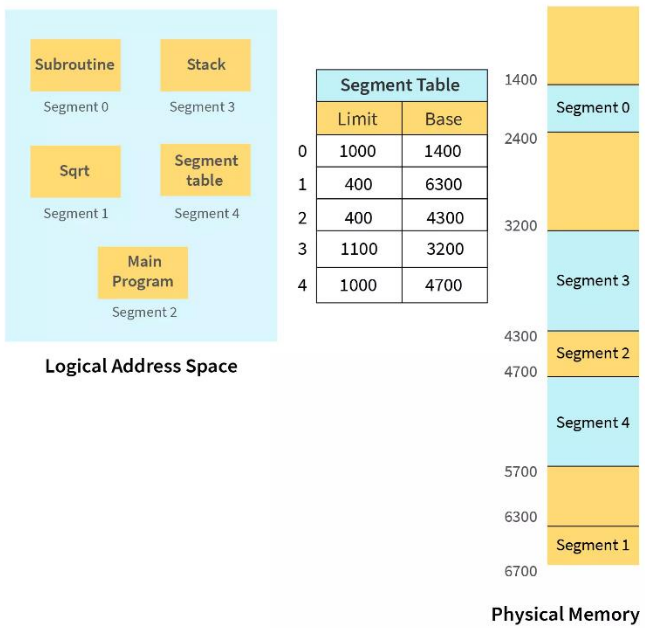
Define segmentation. Explain the user and logical view of segmentation.

Segmentation is a memory management technique in operating system where a process is split into different parts called segments. Each segment is based on what the program does, like code, data, stack, or heap, and these parts are kept in separate places in memory.

User View of Segmentation: From the user’s point of view, a program is not just one big block of memory. Instead, it is divided into different parts like code, global data, stack, and heap. These parts are called segments, and each one has a different task. They can grow or shrink separately as needed. This setup matches how programs are actually written and makes it easier for programmers to organize and manage their code.



Logical View of Segmentation: From the logical point of view, each segment has a segment number, a starting address (called base), and a size (called limit). When a program wants to access memory, it gives the segment number and the position (offset) inside that



Segment Table

	Limit	Base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

segment. The system checks if the offset is within the segment's size. If it is, the system allows access by converting it to a real memory address. If not, it throws an error like a segmentation fault. This way, the system keeps memory access safe and controlled.

Compare the memory organization schemes of contiguous memory allocation and paging with respect to the following issues: External fragmentation, Internal fragmentation.

Contiguous memory allocation and paging are two common memory organization techniques, and they differ significantly in how they handle fragmentation.

Contiguous memory allocation means each process gets one large block of memory placed next to each other in the system's memory. This makes it easy and fast for the system to access and manage memory. However, as processes keep starting and ending, small gaps of free memory appear between used blocks. These scattered free spaces may not be useful if a new process needs a big block of memory. This problem is called **external fragmentation**—even though there's enough total free memory, it's not in one big block. Also, if the memory block given to a process is larger than what it actually needs, the unused space inside the block gets wasted. This is called **internal fragmentation**, but it doesn't happen as much in contiguous memory allocation.

Paging, on the other hand, splits both the process's memory and the physical memory into small fixed-size chunks called pages and frames. Paging solves the **external fragmentation** problem because pages don't need to be next to each other in physical memory. The system can put any page into any free frame, which uses memory more efficiently. However, paging can cause **internal fragmentation** because the last page of a process might not fill up the whole frame, leaving some unused space inside. Usually, this wasted space is small because pages are small, but it can be big when many processes are running.

In summary, Contiguous memory allocation often has external fragmentation but less internal fragmentation. Paging removes external fragmentation but can have some internal fragmentation.

Virtual Memory

What is paging? Explain the concept of free frames in memory management and how they are used for process allocation. Provide an example of a scenario where you have 10 free frames initially. Two processes, Process A and Process B, request memory allocation. Describe the state of free frames before and after the allocation, and discuss the significance of efficient frame allocation in a multi-process environment.

Paging is a memory management technique in operating systems where the logical memory (used by a program) is divided into fixed-size blocks called pages, and the physical memory (RAM) is divided into fixed-size blocks called frames.

Free frames are the empty blocks in the computer’s physical memory (RAM) that are not being used yet. When a process needs to be loaded, the operating system uses these free frames to store the process’s data. Since processes are divided into smaller parts called pages, each page goes into one free frame.

The operating system keeps a list called a **frame table** to track which frames are free and which are already taken.

When a process needs memory:

- The process is split into pages.
- The OS finds a free frame for each page.
- Each page is placed into one of those free frames.
- The **page table** (a record for each process) keeps track of which page is in which frame.

Initial condition:

- Total free frames = 10
- Frame size = 1 page per frame

➤ Process A: Requires 4 pages = 4 frames

➤ Process B: Requires 5 pages = 5 frames

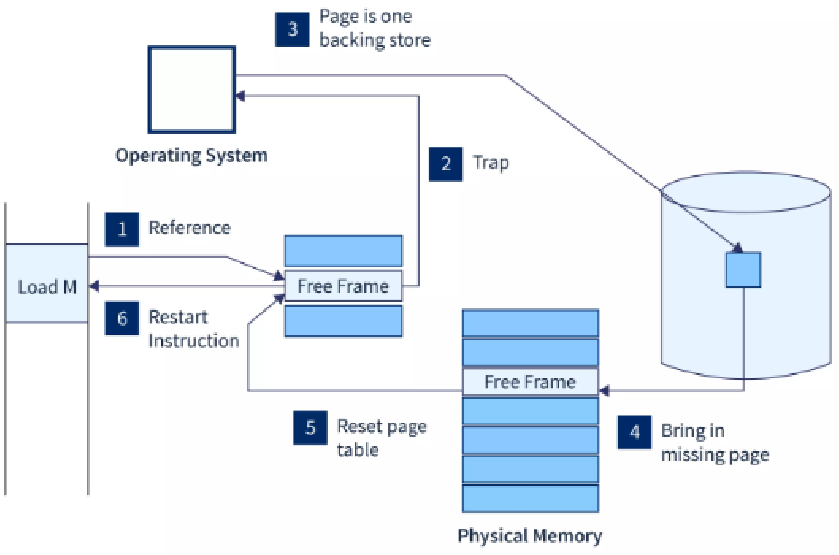
Before Allocation		**After Allocation**	
Frame No.	**Status**	**Frame No.**	**Status**
0	Free	0	Process A
1	Free	1	Process A
2	Free	2	Process A
3	Free	3	Process A
4	Free	4	Process B
5	Free	5	Process B
6	Free	6	Process B
7	Free	7	Process B
8	Free	8	Process B
9	Free	9	Free

Importance of Efficient Frame Allocation:

1. Stops too much swapping (thrashing) that slows everything down (সোয়াপিং কমায়ে performance ডাউন কমায়)
2. Lets more programs run at the same time without running out of memory (একসাথে বেশী প্রোগ্রাম চলতে দেয়)
3. Gives system or important processes the memory they need first (যেটার গুরুত্ব বেশী সেটাকে প্রাধান্য দেয়)
4. Fewer page faults means faster program performance (performance বাড়ায়)

Discuss the impact of page faults on the overall performance of a computer system, including under what circumstances page faults occur. Explain the key responsibilities and steps the operating system takes to handle a page fault in a virtual memory system—such as checking page validity, locating the required page, selecting victim pages if needed, and updating the page table. Additionally, describe how page fault handling mechanisms like demand paging help minimize the negative effects of page faults. Provide an example scenario with a diagram to illustrate the entire page fault handling process.

A page fault happens when a program tries to use a part of memory (a page) that isn't currently in the computer's main memory (RAM).



Circumstances Under Which Page Faults Occur

- A page that is not currently in RAM.
- A page marked as invalid or missing in the page table.
- A page where access is not allowed (like writing to a read-only page).

Key Responsibilities and Steps the OS Takes to Handle a Page Fault

When a page fault happens, the OS does these steps:

- **Interrupt Handling:** The hardware stops the CPU and tells the OS about the page fault.
- **Check Page Validity:** The OS checks if the page is valid.
  - If it's invalid, the OS may stop the program.
  - If valid but not in memory, the OS continues.
- **Locate Required Page:** The OS finds where the page is stored on disk.
- **Find a Free Frame:** The OS looks for a free frame in RAM.
  - If no free frame is available, it picks a page to remove using a method like LRU or FIFO.
  - If the victim page has been changed, it writes it back to disk.
- **Load Page:** The OS reads the needed page from disk into the free frame in RAM.
- **Update Page Table:** The OS updates the page table to mark the page as now in memory and shows the frame number.
- **Restart Instruction:** The OS restarts the program instruction that caused the page fault.

Impact of Page Faults on Overall Performance

1. Page faults make programs run slower because data comes from the hard drive.
2. They increase disk I/O, which slows down the system performance.
3. Too many page faults can make the system very slow (thrashing).
4. They make the CPU wait longer, reducing its efficiency.
5. Apps take more time to respond, which affects the user experience.

How Demand Paging Helps Minimize Negative Effects of Page Faults

Demand paging is a smart memory management method which minimize the negative effects of page faults

1. Demand paging loads only the pages that are needed, saving memory.
2. It lowers page faults by keeping important pages in memory.
3. It helps programs start faster by not loading everything at once.
4. It makes better use of RAM by skipping unused pages.
5. It improves system speed by using memory in a smart way.
6. It reduces unnecessary disk access

Example: Let’s say a program needs 10 pages, but only 4 frames are free in memory. With demand paging, the OS only loads the pages the program actually uses. If the program tries to access page 5, and it’s not in memory — that’s a page fault. So, the OS loads page 5 into a free frame on demand.

Explain the paging hardware with TLB.

Describe the role of the Translation Lookaside Buffer (TLB) in a paging system. How does it improve the efficiency of address translation?

Paging hardware is a memory management technique in operating systems where the logical memory (used by a program) is divided into fixed-size blocks called pages, and the physical memory (RAM) is divided into fixed-size blocks called frames.

Explain how the logical address is mapped to physical address with TLB.

How Paging Hardware Works with TLB

1. The CPU creates a logical address, which has two parts: the page number and the offset inside that page.
2. The system first looks for the page number in TLB
    - If it finds it there, it quickly gets the frame number.
    - If it doesn't find it, it looks up the page table in regular memory to find the frame number, then adds this info to the TLB for faster access next time.
3. After finding the frame number, the system combines it with the offset to get the exact physical memory address.
4. Finally, the system uses this physical address to access the data in memory.

Efficiency Improvement in Address Translation

1. Without TLB, the system checks memory two times to find each address.
2. TLB is a small, fast memory that stores recently used address info.

3. If TLB has the needed info, it gives the physical address quickly without using the page table.
4. This saves time by avoiding extra memory checks.
5. It makes memory access faster and helps the system run better.

Analyze thrashing effect in the paging system.

Thrashing happens when a computer spends most of its time moving data between memory and disk instead of running programs. This usually occurs because there isn't enough free memory for the running programs, causing lots of page faults.

How Thrashing Happens:

1. Many programs run at the same time and fight for limited memory space.
2. Each program needs a certain number of memory frames to run smoothly without too many page faults.
3. If all programs together need more memory than available memory, the system keeps swapping pages in and out of memory.
4. This causes lots of page faults, making the CPU busy handling swapping instead of running programs.

What Happens When Thrashing Occurs:

1. The system slows down a lot because the hard drive is working too much.
2. The CPU spends most of its time fixing page faults instead of running programs.
3. The CPU often waits for pages to load, so it's not used efficiently.

4. Programs freeze or become very slow and unresponsive.

Explain the working procedure of Hashed Page Tables with an example.

A hashed page table is a special way for the operating system to manage memory, especially in systems that use a large virtual address space like 64-bit systems.

⚙ How It Works (Step by Step)

1. **Take the Virtual Page Number (VPN):** From the virtual address, the system first extracts the VPN.
2. **Use a Hash Function:** It then runs this VPN through a hash function to find where to look in the hash table.
3. **Check the Table:** At the calculated index in the table:
   - ❖ If there's a matching VPN, it gives us the related Physical Frame Number (PFN).
4. **Handle Collisions:** If there is collison, the system goes through the list at that index to find the right VPN.
5. **Get the Physical Address:** Once the PFN is found, it's combined with the offset from the virtual address to get the final physical address.

Page size = 4KB = 4096 bytes
Hash table size = 4, Hash function = VPN % 4
Hash table:
Index 0: —
Index 1: VPN=5 → PFN=10
Index 2: VPN=2 → PFN=7
Index 3: —

Virtual address: 0x2400 = 9216
VPN = 9216 ÷ 4096 = 2, Offset = 9216 % 4096 = 0
Hash index = 2 % 4 = 2 → HashTable[2] = VPN 2 → PFN 7
Physical address = 7 × 4096 + 0 = **28672**

Answer: 0x2400 maps to physical address **28672**

Explain the LRU (Least Recently Used) page replacement algorithm in the context of memory management. Using a provided sequence of page references and a memory capacity of 4 pages, demonstrate how the LRU algorithm works to manage pages in memory. Calculate the page fault rate for the given sequence. Discuss the advantages and limitations of the LRU algorithm in comparison to other page replacement algorithms. you have the following page reference string: 2321524532

The Least Recently Used (LRU) page replacement algorithm helps the operating system to decide which page to remove from memory when it's full and a new page needs to come in. It removes the page that hasn't been used for the longest time by checking which pages were used recently.

Example: Page reference string: 2 3 2 1 5 2 4 5 3 2
Memory can hold 4 pages.

Step	**Page**	**Memory after step**	**Page Fault?**	**Page Removed**
1	2	2	Yes	-
2	3	2, 3	Yes	-
3	2	2, 3	No (আগে থেকে আছে তাই)	-
4	1	2, 3, 1	Yes	-
5	5	2, 3, 1, 5	Yes	-
6	2	2, 3, 1, 5	No	-

7	4	2, 1, 5, 4	Yes	3 (zeta sobcheye aage use hoiche, 3 ekhane seta… 2, 3 num step e o use hoiche)
8	5	2, 1, 4, 5	No	-
9	3	2, 4, 5, 3	Yes	1
10	2	4, 5, 3, 2	Yes	-

Total page faults: 7

Page fault rate = 7 faults / 10 page references = 70%

Algorithm	**Simple?**	**Works well?**	**When used?**
FIFO	Yes	Not really	Very simple systems
LRU	Medium	Good	Most OS use
LFU	Hard	Sometimes	Rarely in OS
Optimal	No	Best	Only theoretical

Illustrate with an example how the Enhanced Second Chance Page Replacement Algorithm works.

Enhanced Second Chance is an improved version of the Second Chance page replacement algorithm. Enhanced Second Chance uses **two bits per page**:

- **Reference bit (R)**: 1 if recently accessed, 0 otherwise
- **Modify bit (M)**: 1 if page is dirty, 0 if clean

Priority Classes for Replacement

1. **(0,0)** - Not used, clean → **Replace first**
2. **(0,1)** - Not used, dirty → **Replace second**
3. **(1,0)** - Used, clean → **Replace third**
4. **(1,1)** - Used, dirty → **Replace last**

Example

Memory: 3 frames
Reference string: 1, 2, 3, 4

Step 1-3: Load pages 1, 2, 3

Frame 0: Page 1 (1,0)

Frame 1: Page 2 (1,0)

Frame 2: Page 3 (1,0)

Clock hand → Frame 0

Step 4: Reference page 4 (Page fault)

Clock search from Frame 0:

- Frame 0: Page 1 (1,0) → Give second chance, set to (0,0)
- Frame 1: Page 2 (1,0) → Give second chance, set to (0,0)
- Frame 2: Page 3 (1,0) → Give second chance, set to (0,0)
- Frame 0: Page 1 (0,0) → **Replace with Page 4**

Final State

Frame 0: Page 4 (1,0)

Frame 1: Page 2 (0,0)

Frame 2: Page 3 (0,0)

Clock hand → Frame 1

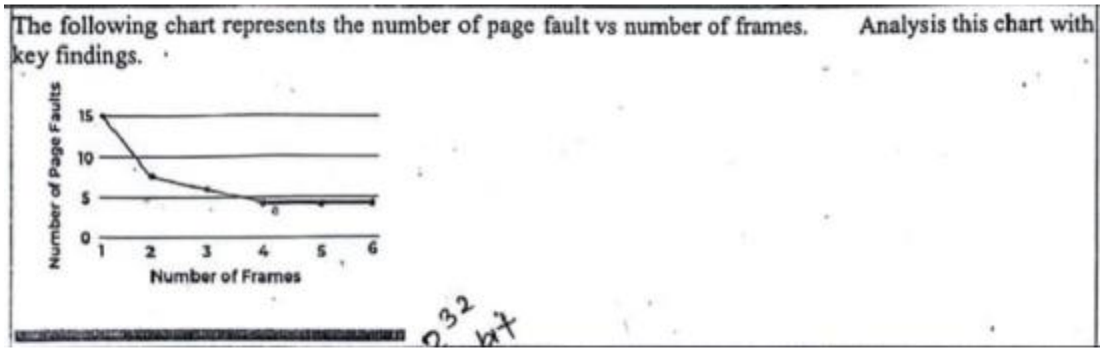
In spite of limited physical memory, we can execute a program whose logical memory size is greater than physical memory of the system. How is it possible?

It is possible because of virtual memory. The operating system loads only the parts of the program that are needed into physical memory using methods like paging or segmentation, while keeping the rest stored on the disk. This lets the program run even when its full size is bigger than the physical memory available.

Considering factors like page table size overhead and potential reduction in page faults, discuss the potential benefits and drawbacks of using a larger page size (8 KB) compared to a smaller one (2 KB).

Using a bigger page size like 8 KB means fewer pages are needed to cover a program's memory. This makes the page table smaller and saves memory. It can also lower the number of page faults because each page holds more data, which can speed things up sometimes. But bigger pages also cause more wasted space inside each page, called internal fragmentation. Plus, when a page fault happens, moving a bigger page takes more time.

On the other hand, smaller page sizes like 2 KB use memory more efficiently with less waste inside pages, but they need bigger page tables and might cause more page faults since less data loads each time.


The following chart represents the number of page fault vs number of frames. Analysis this chart with key findings.

Key Findings:

- When we increase the number of frames from 1 to 6, page faults go down.
- This shows the **principle of locality** — the more frames, the less page faults.
- After 4 frames, the drop in faults becomes smaller, showing **less benefit** as we add more frames.
- There's **no Belady's Anomaly** — adding more frames didn't increase page faults.

Conclusion: Giving more frames to a process helps improve performance, but after a certain number, the improvement becomes less noticeable.

File Systems

Explain the different file access methods used in operating systems and file systems. Compare and contrast the characteristics of sequential access, direct access, and indexed access methods. Provide examples of scenarios where each access method is most suitable and discuss the advantages and limitations of each in terms of data retrieval and storage.

1. Sequential Access: Data is read one by one in order, from start to end. Easy to set up. Good for batch jobs (like payroll). We can't jump to a specific part.

Example: Reading a log file or processing bank transactions in order.

Pros: Works well for reading the whole file. Takes little system resources.

Cons: Not good if we need to jump to a random part. Slow if the file is updated often.

2. Direct (Random) Access: We can go straight to the data using its position. Uses file address or offset. Great for quick lookups.

Example: Booking a seat in an airline system or searching in a database.

Pros: Fast to find what we need. Easy to change or update parts.

Cons: Needs more effort to set up. Takes more space to store addresses.

3. Indexed Access: We use an index (like a table of contents) to find data faster. Mixes both sequential and direct access. Finds data using a key or pointer.

Example: File systems like FAT or NTFS, or when searching in a big database.

Pros: Works well with large data. Lets us read in order or jump to parts.

Cons: Needs extra space for the index. May slow down if the index gets too big.

What is file allocation method? Describe Linked Allocation and Indexed Allocation method.

File Allocation Method: This is how the operating system decides to store a file on the disk. It shows how file data is spread across disk blocks and how the system keeps track of those blocks.

Linked Allocation: In linked allocation, a file is stored as a chain of disk blocks that can be anywhere on the disk. Each block holds some data and a pointer (or link) to the next block. The system remembers the first block's address in the directory. This method stops external fragmentation and makes it easy to add more blocks at the end. But it's slow if you want to jump to a certain part of the file because you have to follow the chain from the start. Also, the pointer takes up some space in each block.

Indexed Allocation: Indexed allocation keeps all the addresses of a file's blocks in one special block called the index block. The directory points to this index block. Since the index lists all the blocks, the system can quickly find any block directly. It avoids external fragmentation and works well for both reading in order and jumping around. The downside is that the index block needs extra space, and for very big files, you might need multiple levels of indexing.

Illustrate how File-Allocation Table works.

1. FAT is a table on the disk that keeps track of all the blocks. Each spot in the table tells you which block comes next for a file or if it's the last block.
2. The directory remembers where the first block of every file is located.
3. When reading a file, the system starts at the first block and follows the FAT chain from block to block until it reaches the end.
4. When writing or adding data, the system updates the FAT to connect new blocks to the file's chain.

Protection and Security

What are some best practices for securing a network against threats?

1. **Use Strong Login Checks:** Make sure passwords are strong and ask for extra proof, like a code from your phone, to log in.
2. **Keep Everything Updated:** Regularly update your software and devices with the latest security fixes.
3. **Use Firewalls and Antivirus:** Set up firewalls to stop bad access and antivirus programs to catch viruses and malware.
4. **Encrypt Data:** Protect important data by turning it into code during sending and saving.
5. **Limit User Permissions:** Give people only the access they need to do their work, no more.
6. **Watch Network Traffic:** Keep an eye on network activity to spot anything unusual or suspicious.
7. **Back Up Data Often:** Save copies of your data regularly so you can restore it if something goes wrong, like a virus attack.

How can program threats impact a computer system or network?

Program threats can harm a computer system or network by doing bad things like deleting or stealing data, making the system slow, or letting attackers take control. These threats can spread viruses, worms, or malware that interrupt normal work, cause data loss, or open security gaps for more attacks, which can damage the system's safety and privacy.

Explain the concept of "breach of availability" in computer security. How can unauthorized destruction of data and denial of service (DoS) attacks impact the availability of a system or service? Provide examples of scenarios where these security violations might occur and

discuss strategies for maintaining system availability despite potential threats.

A breach of availability happens when a system, network, or service is stopped from being used or accessed by the people who should be able to use it. This means users can't get to the data or tools they need, which causes problems in normal work. Availability is one of the key parts of security, along with keeping data private (confidentiality) and correct (integrity).

When someone destroys data without permission, it hurts availability because important information is lost or damaged. This makes the data unusable for the right users and can stop work from going on, causing downtime and loss of trust. A Denial of Service (DoS) attack also affects availability by sending too many fake requests to a system or network. This overload uses up resources like internet bandwidth or the computer's processor, making the system slow or even crash. Because of this, real users can't connect or use the service until the attack is stopped.

For example, a hacker deleting important database files is data destruction, or a group of infected computers flooding a website with traffic is a DoS attack. To keep systems available, companies use backups to restore data, firewalls and security tools to block attacks, load balancers to manage high traffic, and extra systems so the service keeps working even if one part breaks.

How does Trojan Horse destroy a computer system?

1. A Trojan Horse tricks users by looking like a safe or useful program, so people install it without knowing it's harmful.

2. After it’s inside, it quietly runs bad code that can start damaging the system without the user noticing.
3. It can delete important system files or mess up data, which may cause programs or even the whole operating system to stop working or crash.
4. The Trojan can secretly steal personal information like passwords, bank details, or private files and send them to attackers.
5. It often creates hidden entry points called backdoors, letting hackers control the computer from far away or put in more malware.
6. By running these harmful processes, Trojans can make the computer slow, crash often, or behave unpredictably.

What is the goal of protection? What do you know about firewall to protect system?

Goal of Protection

1. Stop people from accessing resources without permission
2. Keep data private and safe
3. Make sure data is not changed or damaged
4. Control what users can do to prevent misuse
5. Allow only real, authorized users to access the system

A firewall is a tool, either hardware or software, that watches and controls the data coming in and going out of a network. It works by following security rules set beforehand. The main job of a firewall is to build a wall between a safe internal network (like your computer or company network) and outside networks (like the internet) that might be risky. Firewalls protect your system by blocking bad access but letting good traffic through. They check things like IP addresses, ports, and types of data to stop hackers, viruses, or unwanted programs from getting in. Because of this, firewalls are very important for keeping computers and networks safe from cyber attacks.

Trojan horse, Spyware, Ransomware, and Trap door are popular malware to breach the security of a system. Create a security plan to protect against these threats.

1. **Install and Update Antivirus Software:** We should use trusted antivirus programs like Norton or McAfee. We need to keep them running with real-time scanning and update them regularly to catch new threats.
2. **Use a Firewall**: We must turn on our firewall to block unwanted access from outside. We should also set up firewalls on each computer to protect every system.
3. **Keep Our Software and OS Updated**: We need to install the latest updates and security patches for our operating system and apps. It's important to disable or remove services we don't need so hackers have fewer ways to get in.
4. **Be Careful with Emails and Websites**: We shouldn't open email attachments or click links if we don't trust them. Using spam filters helps block harmful emails. We should visit only safe websites that use HTTPS.
5. **Learn and Control Access**: We must learn about malware risks and practice safe computing. We should give users only the access they need (least privilege). Disabling autorun on USB drives helps stop Trojans from spreading.
6. **Backup Data and Use Encryption**: We should regularly back up important files (offline or in the cloud) so we can recover if ransomware attacks. Encrypting sensitive data stops spyware from stealing our information.
7. **Use Intrusion Detection and Prevention Systems**: We can set up systems that watch for suspicious activity and block

attacks. Keeping logs helps us find hidden trap doors or unauthorized access.

Explain the terms 'WORMS' and 'VIRUSES' with reference to system threats. How do computer worms work?

Worms are harmful programs that make copies of themselves and spread across computer networks on their own. They don't need to attach to any other file to work. Worms usually find weak points in a system's security and move from one computer to another. They can use up system resources, slow down the network, and cause big problems—often without the user even knowing.

Viruses are harmful programs that attach themselves to real files or software. When someone opens or runs the infected file, the virus gets activated. It can damage or delete data, affect how the system works, and spread to other files. Unlike worms, viruses usually need the user to do something, like open a file, to spread—but once they do, they can cause a lot of damage.

1. Worms look for weak spots in networks or systems, like outdated software or open ports.
2. Once a worm gets in, it makes many copies of itself without needing anyone to do it.
3. These copies spread to other systems through things like emails, file sharing, or network links.
4. Worms can slow down the network, use up system resources, or bring in other harmful programs like spyware or ransomware.

Analyze the principle of least privilege regarding system protection.

The principle of least privilege is a basic but very important idea in operating system security. It means every user, program, or process should get only the access it needs to do its job—nothing more. For example, if a user only needs to read a file, they shouldn’t be able to change or delete it. Or if a program is meant just for printing, it shouldn't be able to see or use private files.

This helps keep the system safe. Even if a hacker or malware gets in through a low-level account, they won’t be able to do much damage because they won’t have permission to access sensitive parts. It also makes things easier to manage, as the system only has to keep track of a small set of permissions. Plus, it supports good access control practices and helps follow security rules. In short, this principle protects the system by giving only the needed access—no extra power.

List security violation categories with examples.

Breach of confidentiality: Reading or sharing data without permission.
Example: Someone looking at private files they shouldn’t.

Breach of integrity: Changing data without permission.
Example: Changing a bank balance when you’re not allowed.

Breach of availability: Destroying or blocking data or resources without permission.

Example: Deleting important system files or making a service stop working by flooding it with requests (called a DoS attack).

Theft of service: Using system resources without permission. *Example:* Someone using a company’s computers for their own work without approval.

Denial of service (DoS): Stopping real users from accessing resources.
Example: Overloading a network with too much traffic so the server slows down or crashes.