AOS Assignment No. 4

- **2 Marks Questions**

Q.1.) What is Page Replacement in OS?

Ans :- **Page Replacement** in an operating system is a memory management technique used in **virtual memory** when a requested page is not in RAM (a **page fault** occurs). The OS selects a page to **remove from memory** to make space for the new page. Common **page replacement algorithms** include **FIFO, LRU, and Optimal Page Replacement**.

Q.2.) What is virtual memory?

Ans :- **Virtual memory** is a memory management technique that extends the available physical RAM by using a portion of the **hard disk (swap space or page file)**. It allows programs to run even if they exceed the physical memory limit by swapping data between RAM and disk as needed.

Q.3.) Define Demand Paging in OS.

Ans :- **Demand Paging** is a virtual memory technique where pages are loaded into RAM **only when required** during execution, rather than preloading all pages. This reduces memory usage and improves efficiency, but may cause **page faults** when accessing pages not in memory.

Q.4.) What is physical address and logical address?

Ans :- **Logical Address**: The address generated by the CPU during program execution, also called a **virtual address**. It is mapped to a physical address by the **Memory Management Unit (MMU)**.

Physical Address: The actual address in **RAM** where data is stored. It is used by the hardware to access memory directly.

Logical Address → (MMU) → Physical Address

Q.5.) Define Page Fault and Page hit.

Ans :- **Page Fault**: A **page fault** occurs when a process tries to access a page that is **not currently in RAM**, requiring the operating system to fetch it from secondary storage (disk). This can cause delays in execution.

Page Hit: A **page hit** happens when the requested page is **already present in RAM**, allowing the process to access it immediately without any delay.

- **10 Marks Questions**

Q.1.) What is Frame allocation in virtual memory? Define its constraints?

Ans :- **Frame allocation** refers to the process of assigning a specific number of **fixed-size frames** (blocks of physical memory) to each process in a system using **paging**. The operating system manages the allocation to ensure efficient memory utilization and process execution.

Constraints of Frame Allocation

1. **Total Frame Availability** – The total number of frames in RAM is limited, so they must be distributed efficiently among processes.
2. **Minimum Frames per Process** – Each process must receive a minimum number of frames to function correctly and avoid excessive page faults.
3. **Thrashing Prevention** – If too few frames are allocated, frequent page faults can occur, causing **thrashing** (excessive swapping between disk and RAM).
4. **Local vs. Global Allocation** – The OS must decide whether to allocate frames **locally** (fixed per process) or **globally** (shared dynamically based on demand).
5. **Fairness & Priority** – High-priority processes may require more frames to ensure better performance compared to lower-priority ones.

Q.2.) Explain frame allocation algorithms.

Ans :- Frame allocation algorithms determine how **available frames (blocks of physical memory)** are distributed among processes in a **paging system**. The goal is to ensure **efficient memory usage** while minimizing **page faults**.

1. Equal Allocation

- Each process gets an **equal number of frames**, regardless of its size or priority.
- Simple to implement but may lead to **inefficient memory use** if some processes need more memory.

Example:
If there are **100 frames** and **5 processes**, each process gets **100/5 = 20 frames**.

Advantage: Fair and easy to implement.

Disadvantage: Not suitable when processes have different memory needs.

2. Proportional Allocation

- Frames are allocated **based on process size** (memory requirement).
- Larger processes get **more frames**, and smaller ones get **fewer frames**.

Example:
If total RAM has **100 frames** and two processes **P1 (size 200MB)** and **P2 (size 300MB)** exist:

- **P1 gets (200/500) × 100 = 40 frames**
- **P2 gets (300/500) × 100 = 60 frames**

Advantage: Better memory distribution based on needs.
Disadvantage: May still cause thrashing if demand changes dynamically.

3. Priority-Based Allocation

- Frames are allocated based on **process priority**.
- High-priority processes get **more frames**, while low-priority processes get **fewer frames**.

Example:
If **total frames = 100** and **P1 (high priority) gets 70 frames**, **P2 (low priority) gets 30 frames**.

Advantage: Ensures critical tasks get more memory.
Disadvantage: Low-priority processes may starve.

4. Global Allocation

- Frames are allocated **dynamically**, adjusting based on demand.
- If a process needs more frames due to frequent **page faults**, it can borrow frames from other processes.

Example:
If **P1 frequently page faults**, it can take frames from **P2** (if available).

Advantage: Adapts to changing memory needs.
Disadvantage: May cause **starvation** for some processes.

5. Local Allocation

- Each process is assigned **a fixed number of frames** that it can use.
- It cannot take frames from other processes, even if they are free.

Example:
If **P1 gets 40 frames** and **P2 gets 60 frames**, they must operate within their limits.

Advantage: Ensures predictable performance for each process.
Disadvantage: May waste memory if frames are not fully utilized.

Q.3.) Explain Optimal Page Replacement algorithm with an example.

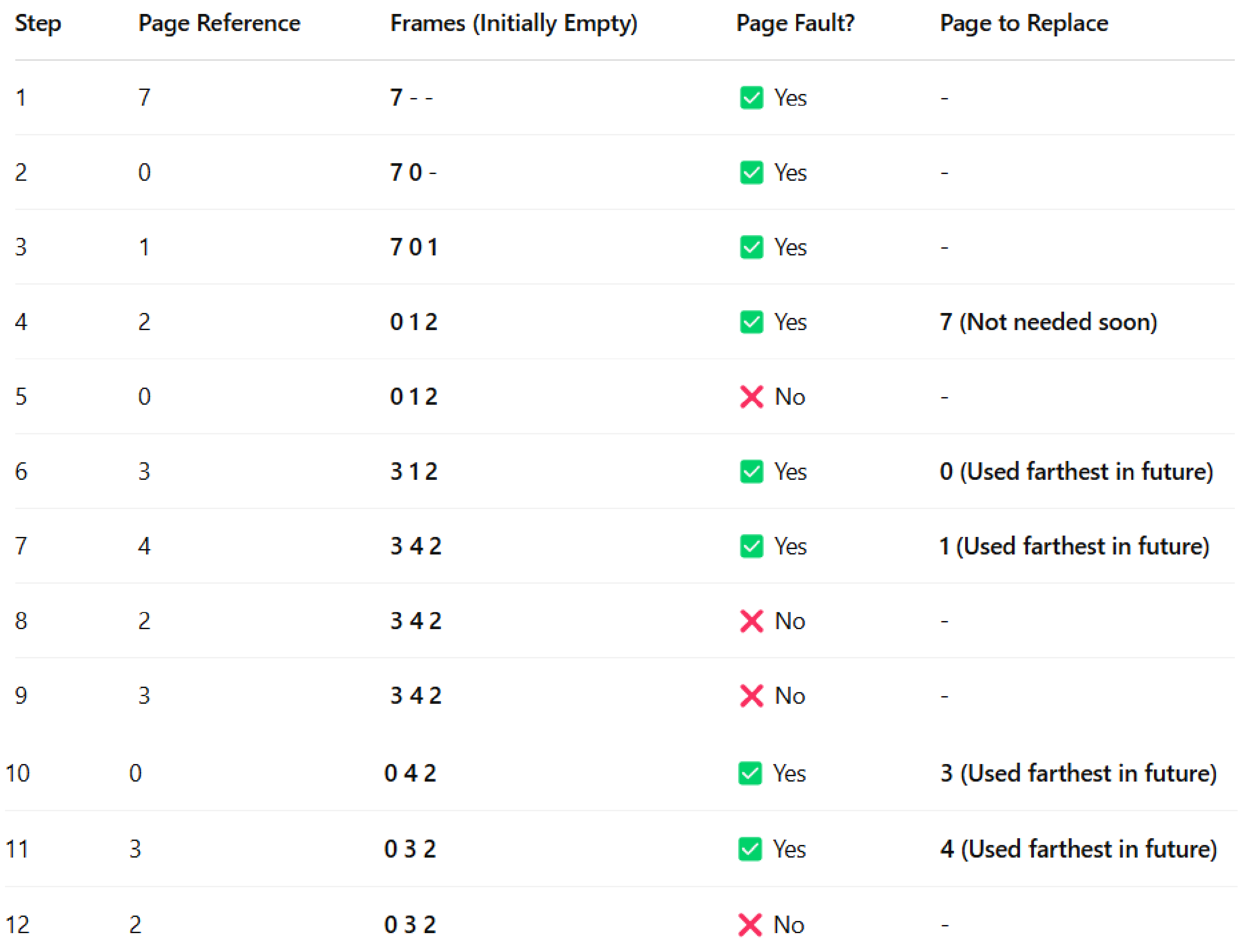
Ans :- The **Optimal Page Replacement** algorithm replaces the page that will **not be used for the longest time in the future**. It **minimizes** the number of page faults and is considered the **best** page replacement strategy, but it is not practically implementable because it requires **future knowledge of page references**.

Example of Optimal Page Replacement

Given:

- **Number of Frames:** 3
- **Page Reference String: 7, 0, 1, 2, 0, 3, 4, 2, 3, 0, 3, 2**

Step-by-Step Execution:



Step	Page Reference	Frames (Initially Empty)	Page Fault?	Page to Replace
1	7	**7 - -**	✅ Yes	-
2	0	**7 0 -**	✅ Yes	-
3	1	**7 0 1**	✅ Yes	-
4	2	**0 1 2**	✅ Yes	**7** (Not needed soon)
5	0	**0 1 2**	❌ No	-
6	3	**3 1 2**	✅ Yes	**0** (Used farthest in future)
7	4	**3 4 2**	✅ Yes	**1** (Used farthest in future)
8	2	**3 4 2**	❌ No	-
9	3	**3 4 2**	❌ No	-
10	0	**0 4 2**	✅ Yes	**3** (Used farthest in future)
11	3	**0 3 2**	✅ Yes	**4** (Used farthest in future)
12	2	**0 3 2**	❌ No	-

Total Page Faults: 8

The algorithm **minimizes page faults** because it always **removes the least-needed page**. However, it is **not practical** for real-world OS implementation since it requires **future knowledge of page accesses**.

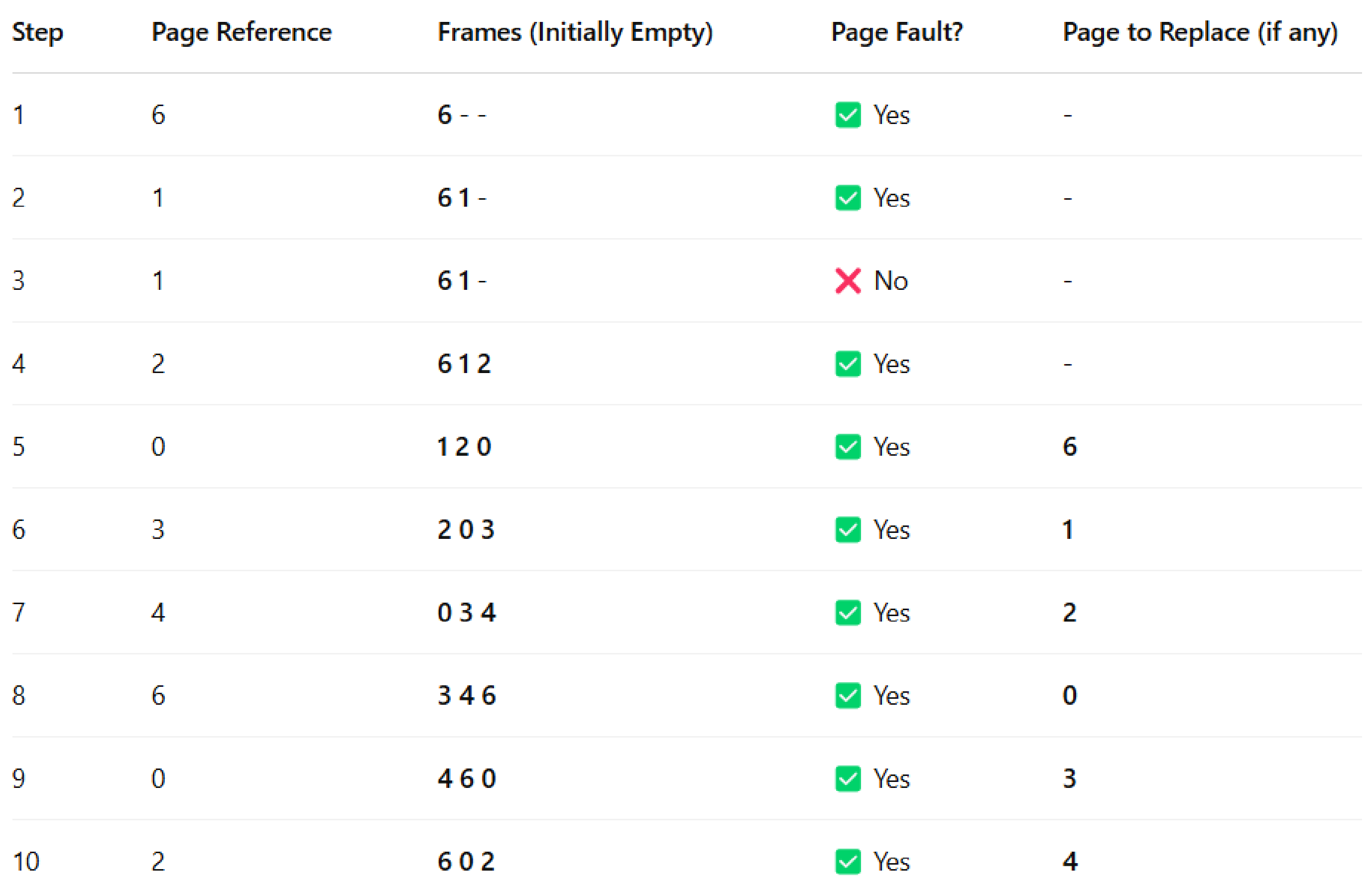
Q.4.) Consider the reference string 6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 4, 0 for a memory with three frames and calculate number of page faults by using LRU Page replacement algorithms.

Ans :- The **LRU (Least Recently Used)** algorithm replaces the page that has **not been used for the longest time** when a new page needs to be loaded into memory.

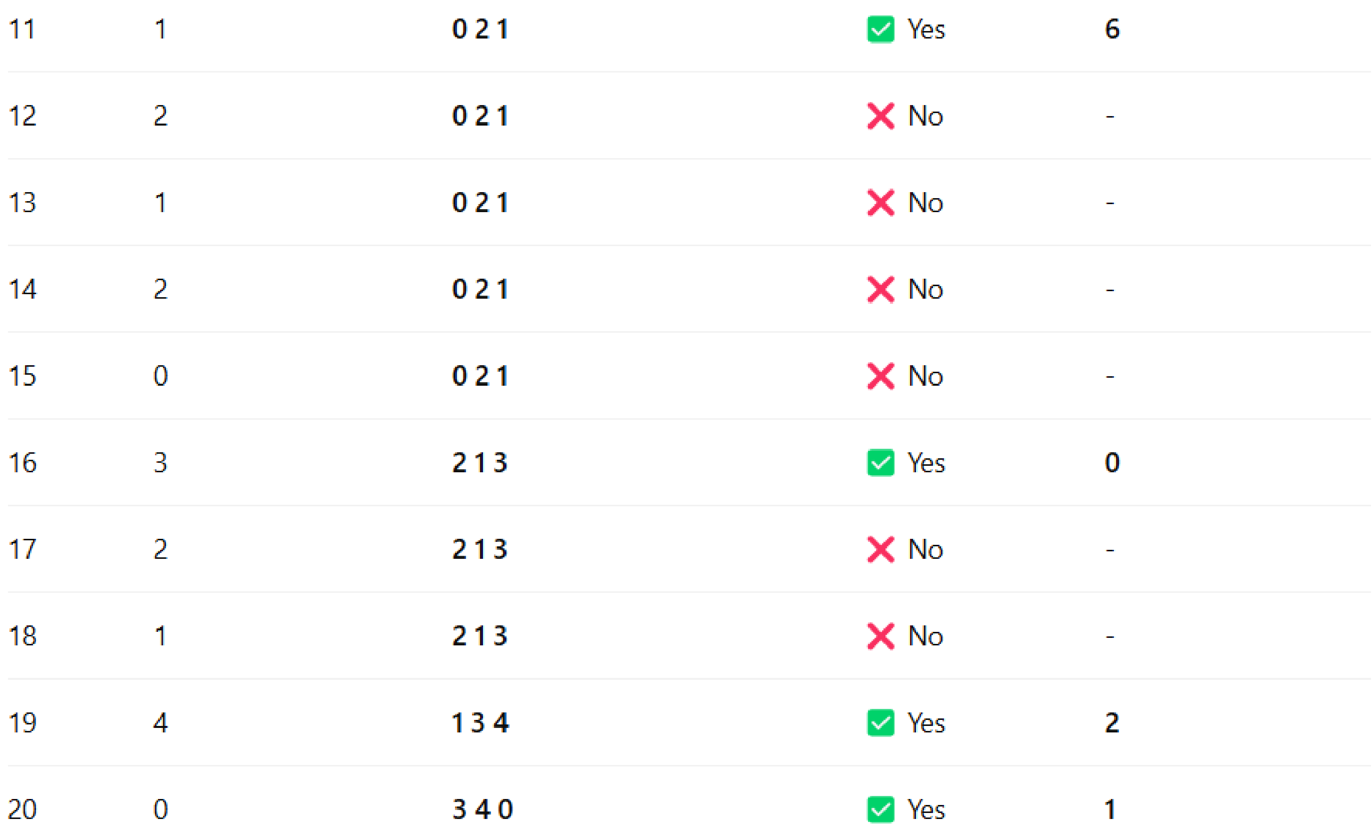
Given Data:

- **Page Reference String: 6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 4, 0**
- **Number of Frames: 3**

Step-by-Step Execution:



Step	Page Reference	Frames (Initially Empty)	Page Fault?	Page to Replace (if any)
1	6	**6 - -**	✅ Yes	-
2	1	**6 1 -**	✅ Yes	-
3	1	**6 1 -**	❌ No	-
4	2	**6 1 2**	✅ Yes	-
5	0	**1 2 0**	✅ Yes	**6**
6	3	**2 0 3**	✅ Yes	**1**
7	4	**0 3 4**	✅ Yes	**2**
8	6	**3 4 6**	✅ Yes	**0**
9	0	**4 6 0**	✅ Yes	**3**
10	2	**6 0 2**	✅ Yes	**4**



11	1	0 2 1	✅ Yes	6
12	2	0 2 1	❌ No	-
13	1	0 2 1	❌ No	-
14	2	0 2 1	❌ No	-
15	0	0 2 1	❌ No	-
16	3	2 1 3	✅ Yes	0
17	2	2 1 3	❌ No	-
18	1	2 1 3	❌ No	-
19	4	1 3 4	✅ Yes	2
20	0	3 4 0	✅ Yes	1

Total Page Faults: 12

- **LRU selects the least recently used page for replacement, which helps optimize memory utilization but requires tracking page usage history.**

Q.5.) Explain the types of Virtual Machine.

Ans :- A **Virtual Machine (VM)** is a software-based simulation of a physical computer that runs an operating system and applications. There are two main types of virtual machines:

1. System Virtual Machine

A **System VM** provides a **full virtualization of hardware**, allowing multiple operating systems to run independently on a single physical machine.

Examples:

- **VMware Workstation**
- **Oracle VirtualBox**

- **Microsoft Hyper-V**
- **KVM (Kernel-based Virtual Machine)**

Features:

- Provides complete **isolation** between VMs.
- Each VM runs its **own OS**.
- Uses a **Hypervisor** (Type 1 or Type 2) to manage VMs.
- Supports **resource sharing** (CPU, memory, storage).

2. Process Virtual Machine

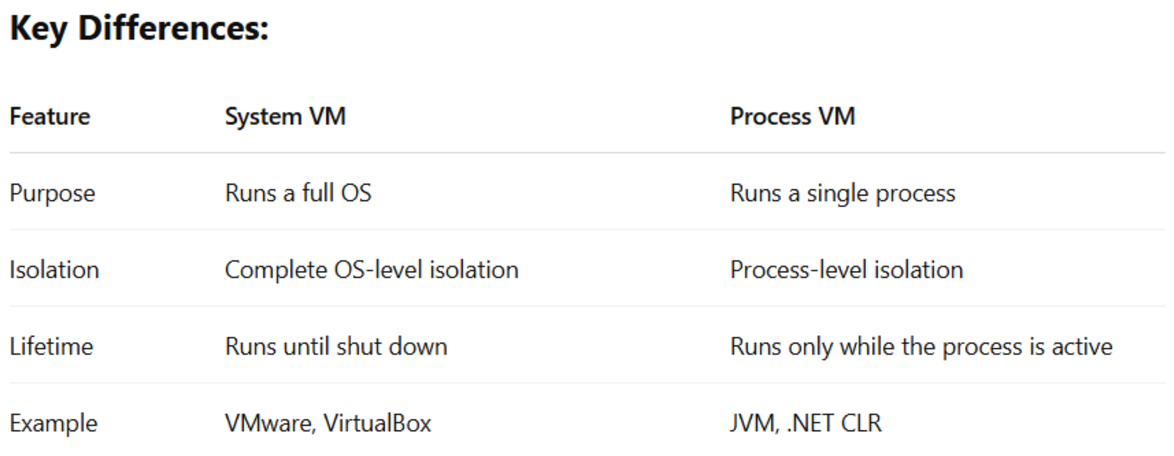
A **Process VM** runs a **single application or process** in an isolated environment. It is created when the process starts and destroyed when the process ends.

Examples:

- **Java Virtual Machine (JVM)** (Runs Java applications)
- **.NET Framework Common Language Runtime (CLR)** (For .NET applications)
- **Parrot Virtual Machine** (For dynamic languages like Perl, Python)

Features:

- Provides an **abstraction layer** for running applications.
- Ensures **cross-platform compatibility** (e.g., Java code runs on different OS).
- No need to install a separate OS inside the VM.
- Short-lived compared to system VMs.



Key Differences:

Feature	System VM	Process VM
Purpose	Runs a full OS	Runs a single process
Isolation	Complete OS-level isolation	Process-level isolation
Lifetime	Runs until shut down	Runs only while the process is active
Example	VMware, VirtualBox	JVM, .NET CLR