SNS COLLEGE OF TECHNOLOGY

An Autonomous Institution

Coimbatore-35

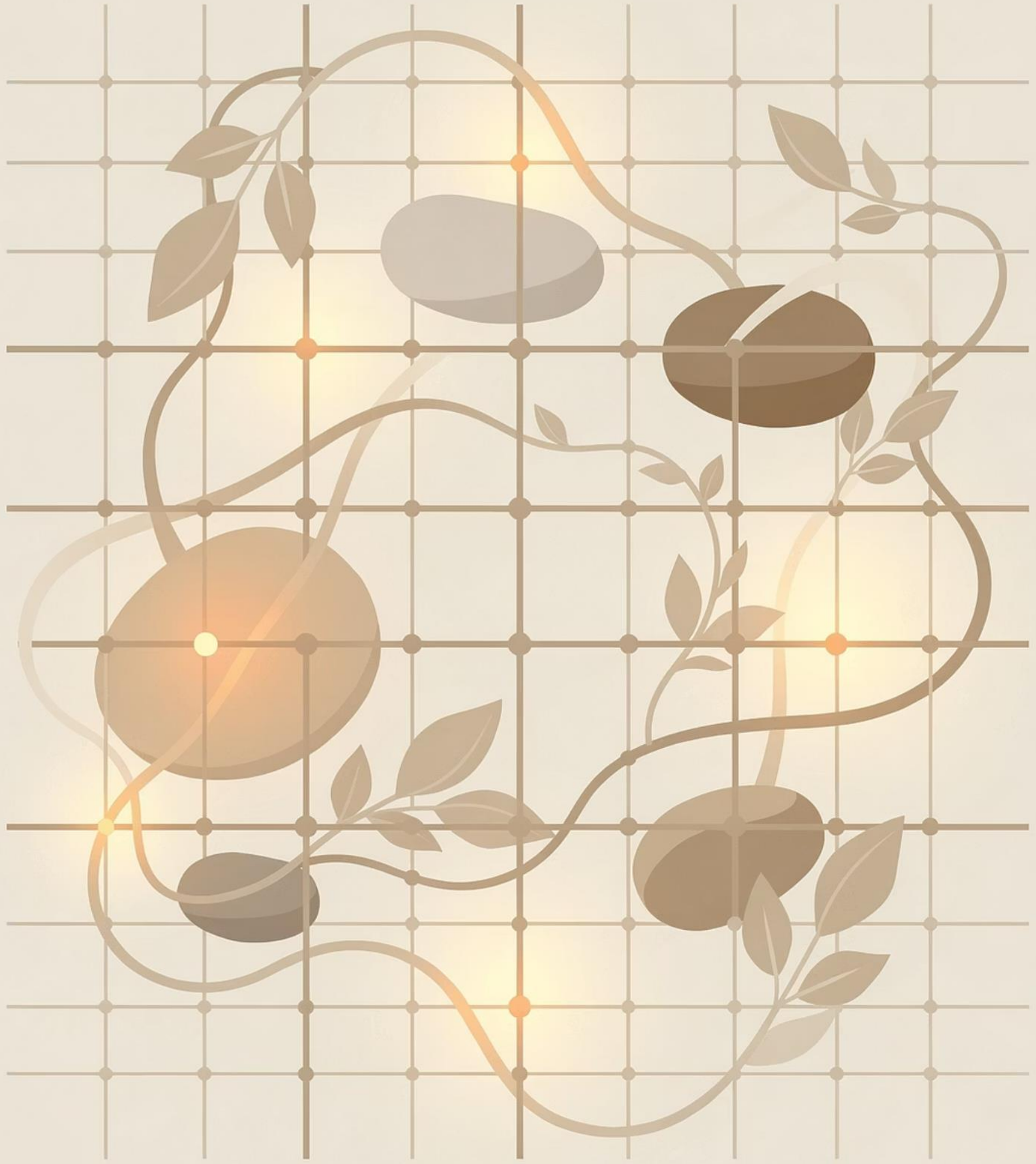
Department of Computer Science and Engineering

23CST206–OPERATING SYSTEMS AND VIRTUALIZATION

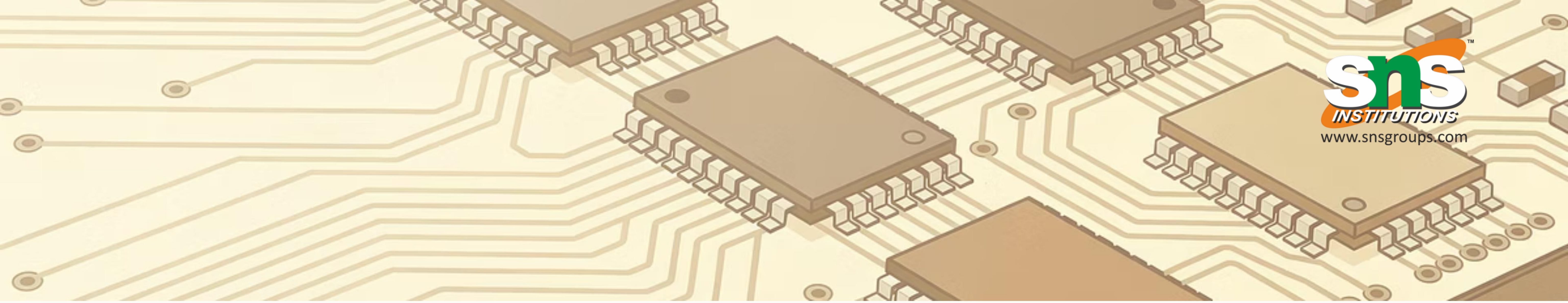
B.E- CSE /IV SEMESTER

UNIT – III MEMORY MANAGEMENT

Topic 3:Paging



Paging - Structure of the Page Table



INTRODUCTION

What is Paging?

Paging is the process of moving parts of a program, called pages, from secondary storage into main memory (RAM). The main idea is to break a program into smaller fixed-size blocks called pages.

Non-Contiguous Allocation

Process doesn't need contiguous memory space.

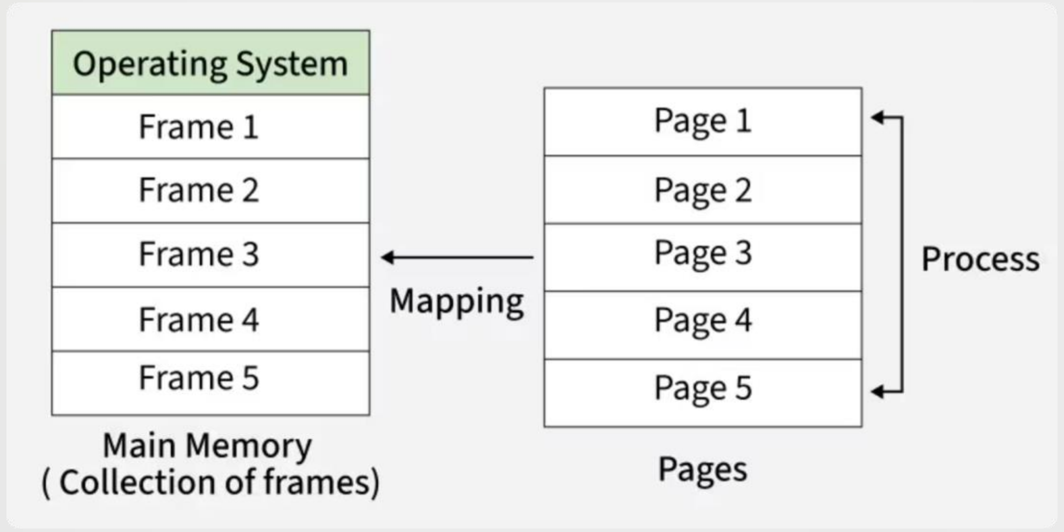
On-Demand Loading

Pages load when needed, allowing more processes to run.

Simplified Allocation

Memory allocated in fixed-size pages.

Paging Memory Management Technique



Paging is a memory management technique in which:

- Logical (virtual) memory is divided into fixed-size blocks called **pages**
- Physical memory is divided into fixed-size blocks called **frames**
- Pages are mapped to frames using a **page table**

Paging in Memory Management

Paging addresses common challenges in allocating and managing memory efficiently. Why paging is needed as a memory management technique:

Memory Fragmentation

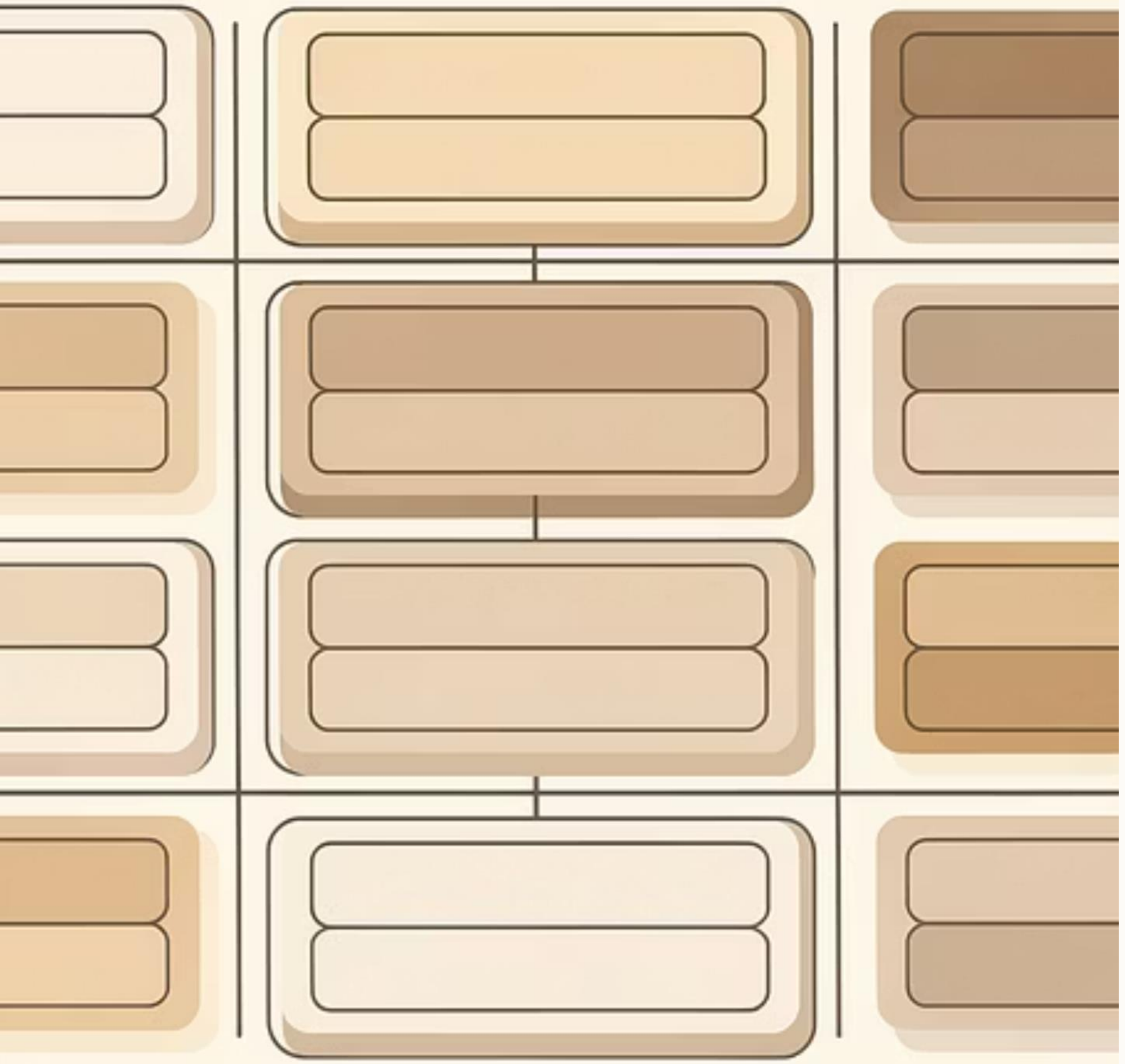
Dynamic Process Size

Logical (Virtual) Address Space

The set of all addresses generated by a process during program execution.

Physical Address Space

The set of actual memory addresses available in main memory (RAM).



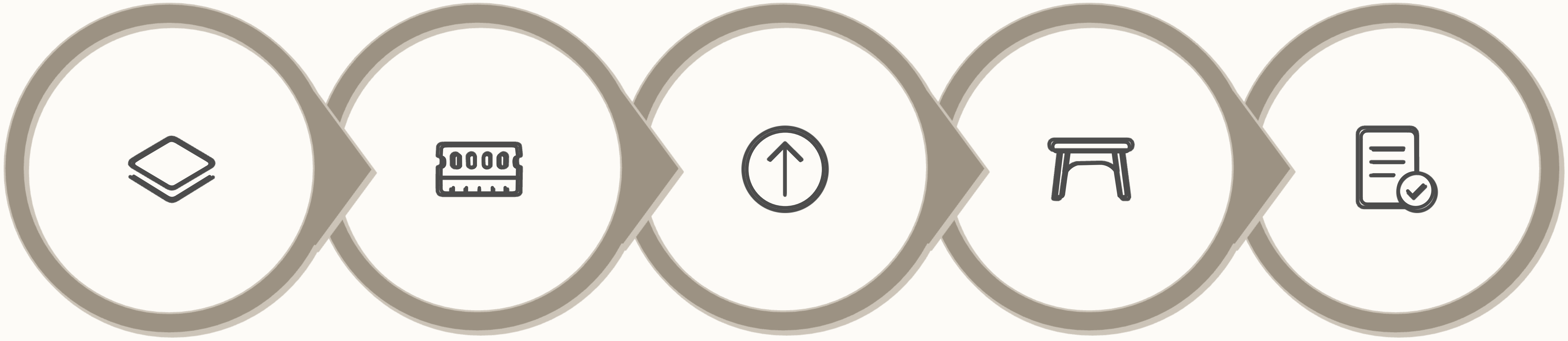
Working of Paging

Simple Explanation

Paging is a memory management technique in which logical memory is divided into fixed-size pages and physical memory is divided into fixed-size frames. Pages of a process can be stored in any available frame, not necessarily in contiguous memory.

PROCESS

How Paging Works



Divide Logical

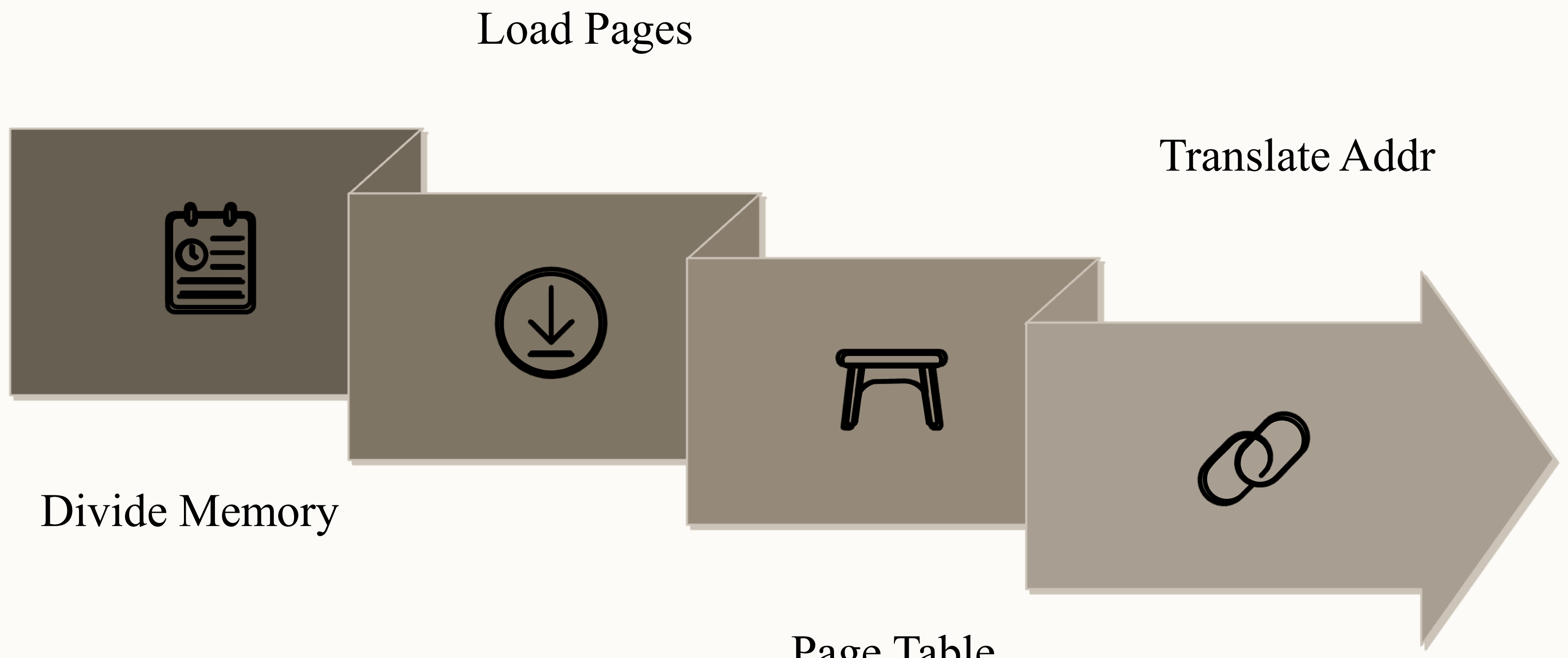
Divide Physical

Load Pages

Create Page Table

Address Translation

Steps Involved in Paging



Address Structure in Paging

Logical Address

Page Number (p) and Offset (d)

Offset (d)

Shared between logical and physical

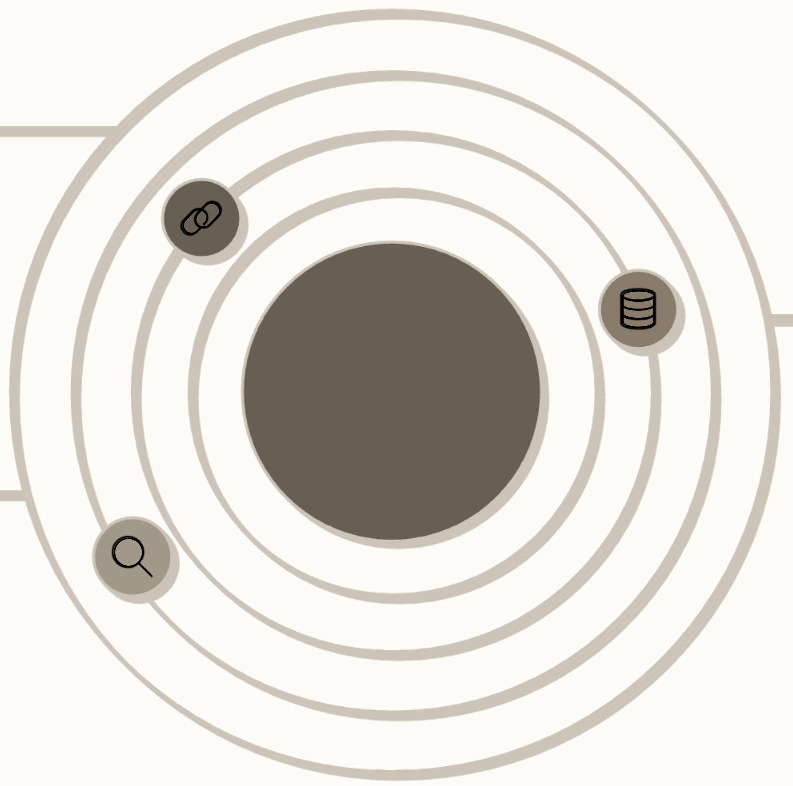
Physical Address

Frame Number (f) and Offset (d)

CORE CONCEPT

What is a Page Table?

A page table is a data structure maintained by the operating system for each process that stores the mapping between logical pages and physical memory frames.



Address Mapping

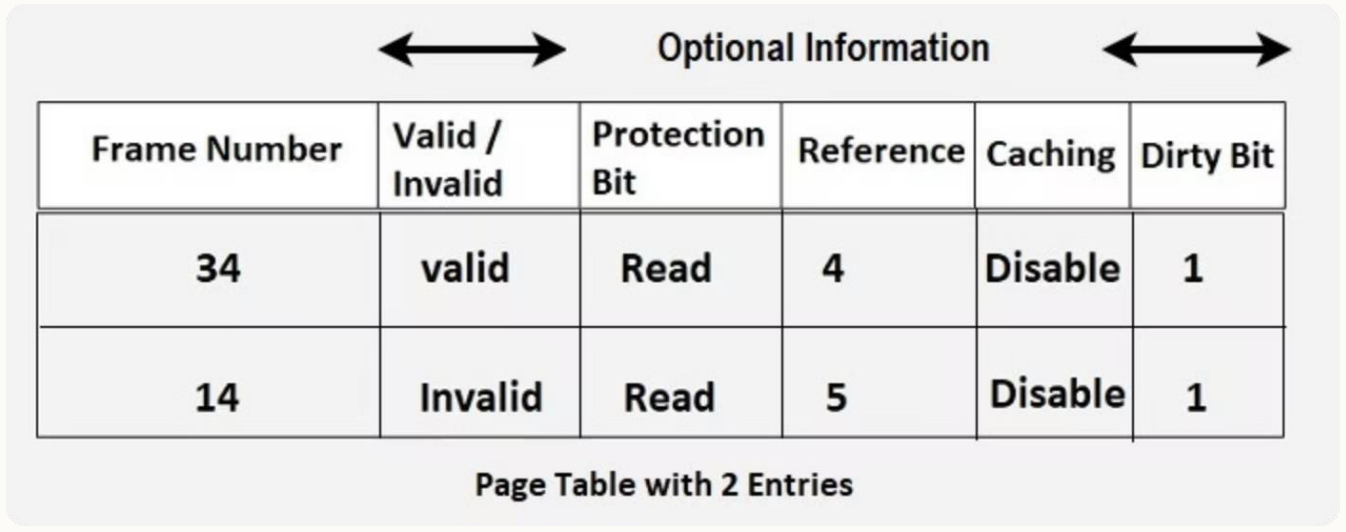
Maps logical pages to physical frames

Page Information

Stores frame, disk location, and permissions

Quick Lookup

Enables OS to locate pages in memory fast

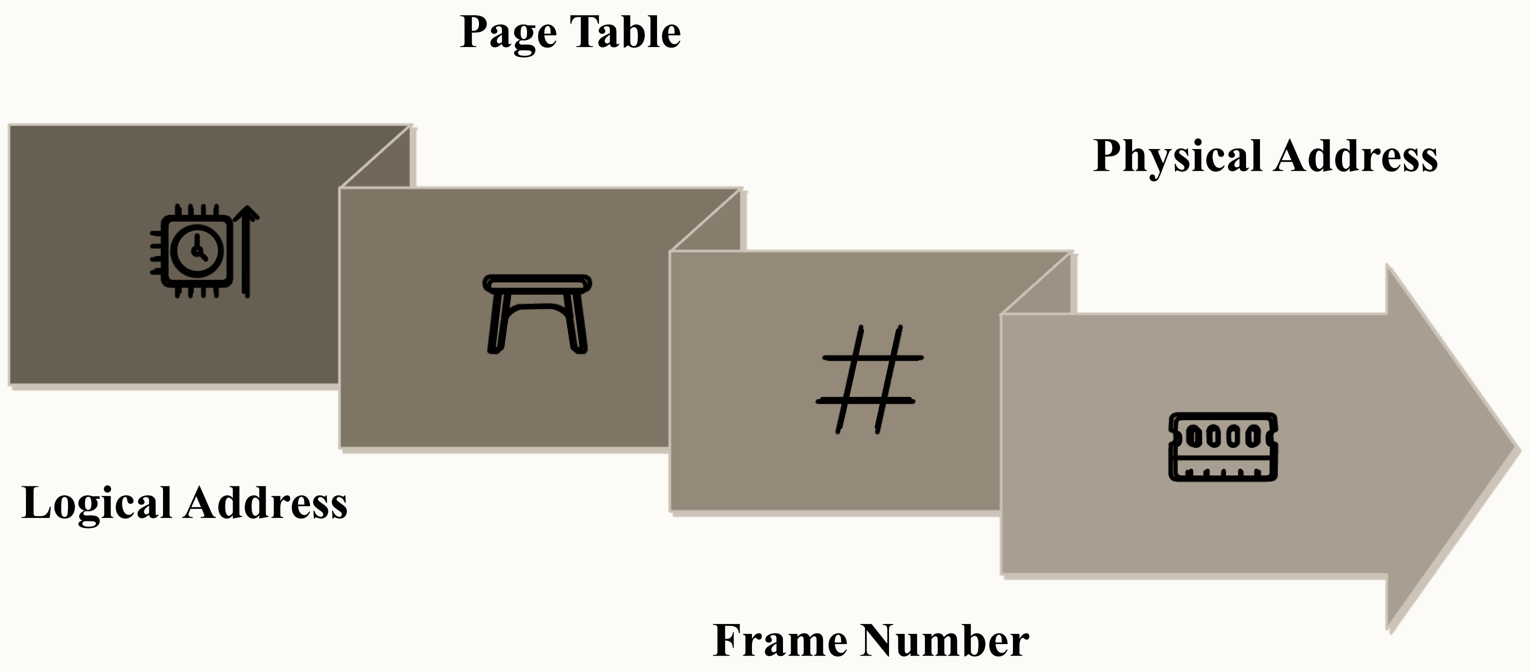


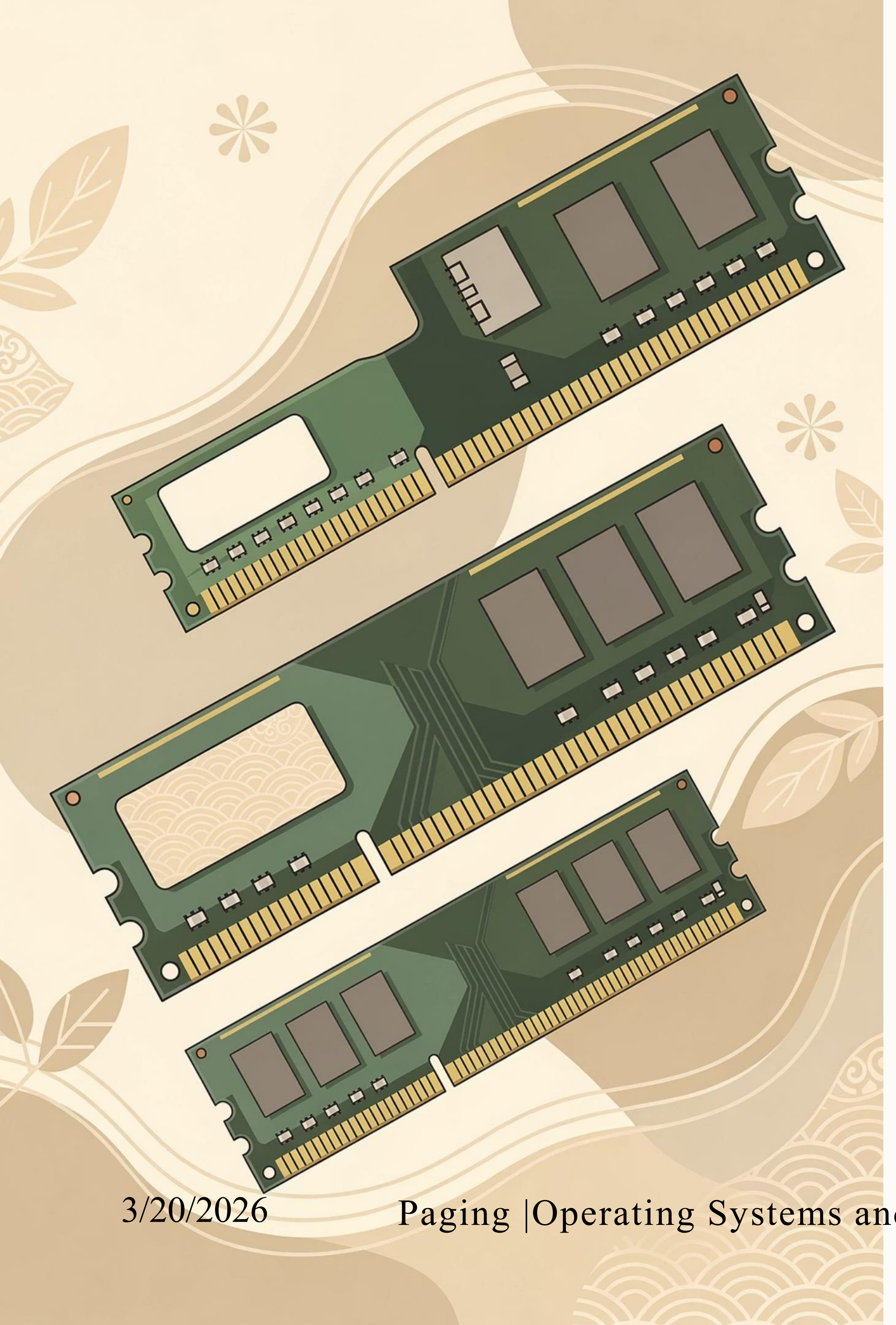
	⟵ Optional Information	⟶			
Frame Number	Valid / Invalid	Protection Bit	Reference	Caching	Dirty Bit
34	valid	Read	4	Disable	1
14	Invalid	Read	5	Disable	1

Page Table with 2 Entries

Purpose of the Page Table

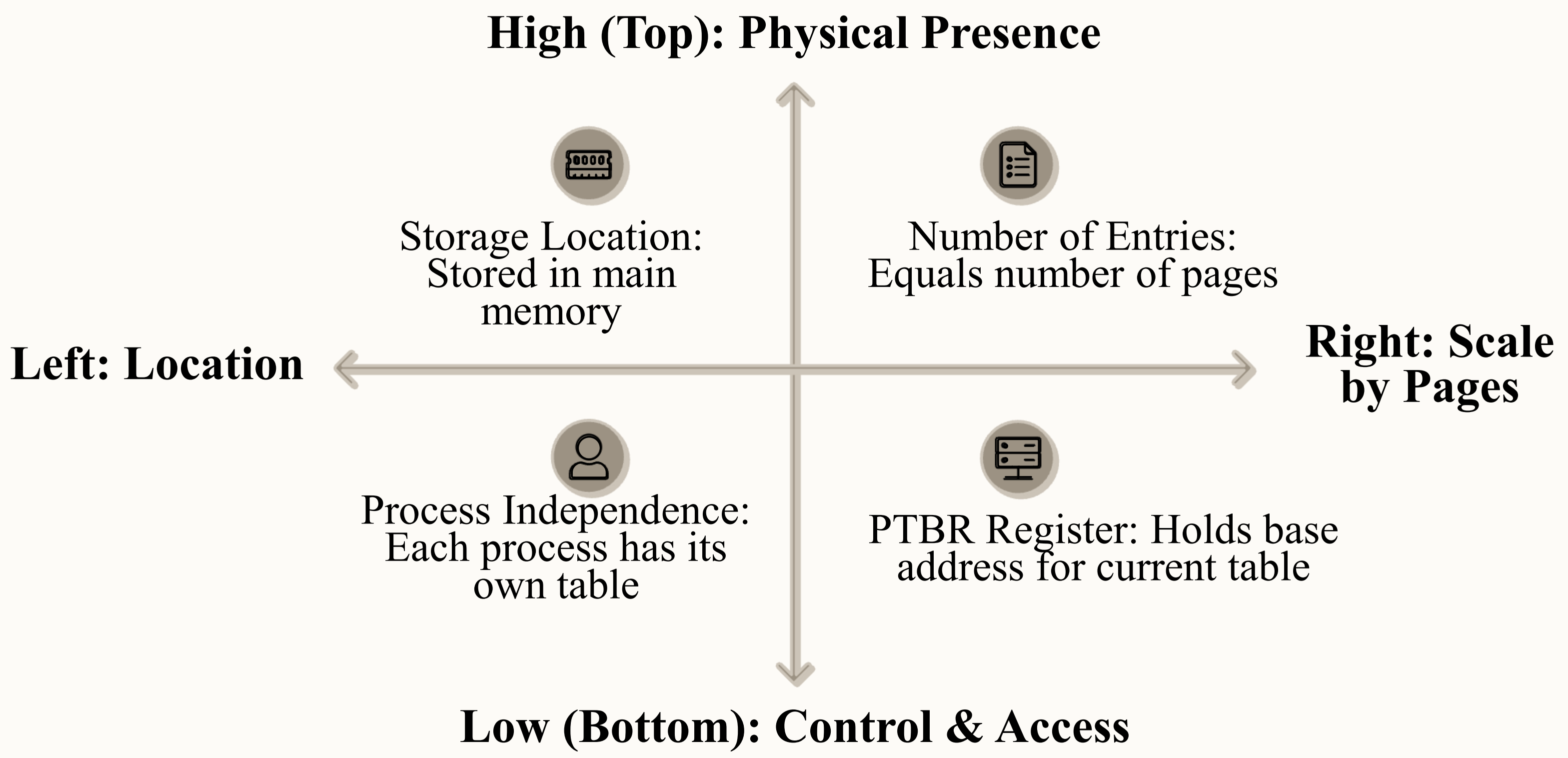
The main purpose of a page table is to keep track of the mapping between logical addresses used by a program and physical addresses in the computer's memory.



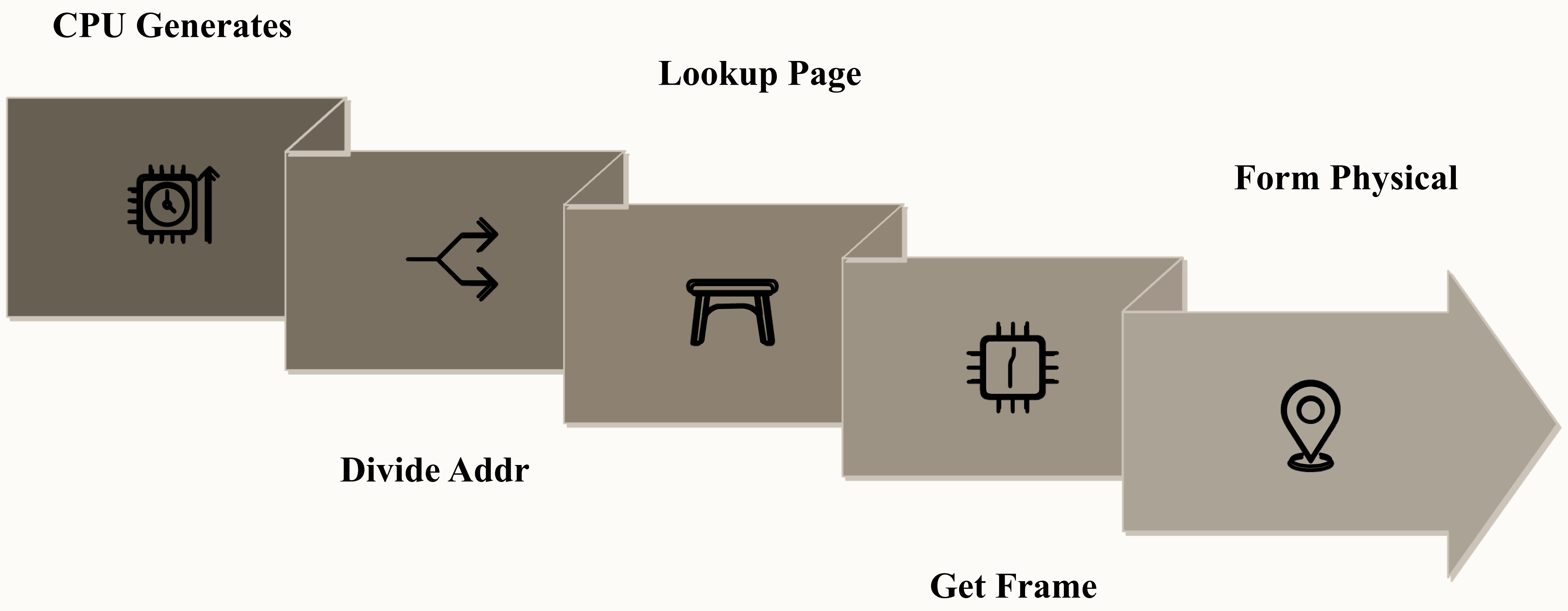


www.snsgroups.com

Characteristics of the Page Table


High (Top): Physical Presence
Storage Location: Stored in main memory
Number of Entries: Equals number of pages
Left: Location
Right: Scale by Pages
Process Independence: Each process has its own table
PTBR Register: Holds base address for current table
Low (Bottom): Control & Access

How Page Table Works



What Does a Page Table Contain?

Each page table entry (PTE) typically includes:

Page Number

Identifies the logical page

Frame Number

Physical memory location

Valid/Invalid Bit

Indicates if page is present or not

Protection Bits

Read/write/execute permissions

Dirty Bit

Indicates if page was modified

ADVANCED TECHNIQUES

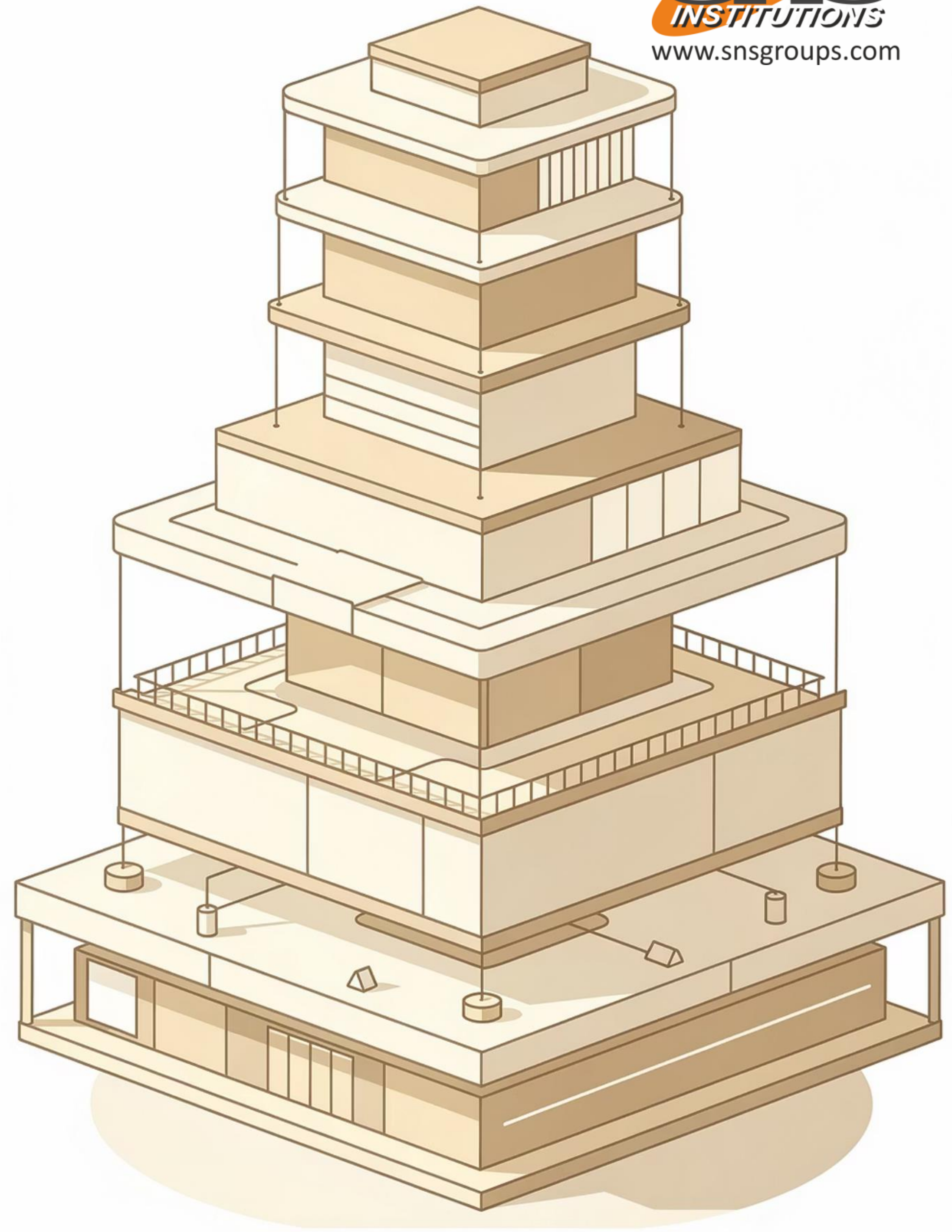
Techniques for Structuring the Page Table

Common techniques used for structuring the page table:

Hierarchical Paging

Hashed Page Table

Inverted Page Table



Hierarchical Paging (Multilevel Paging)

Definition: Hierarchical Paging divides a large page table into multiple smaller page tables, arranged in levels.

Problem Solved

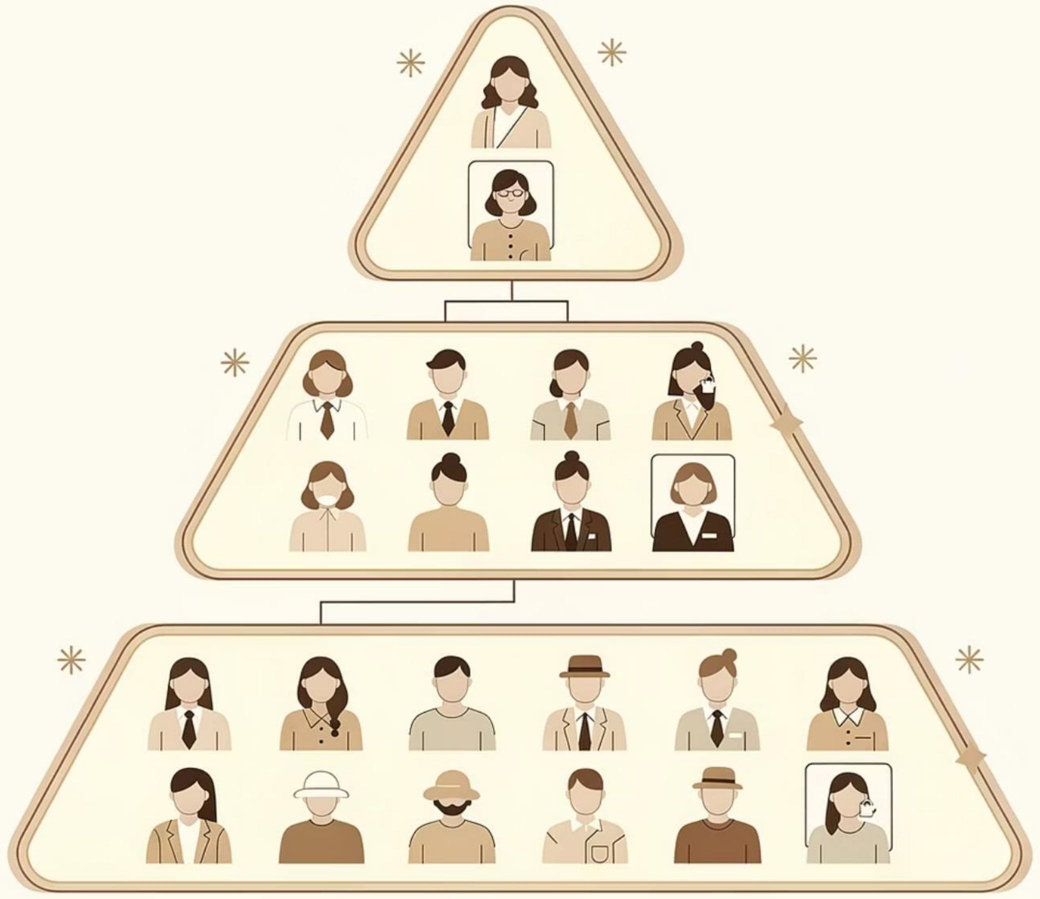
Page table may be too big to fit in contiguous space, so we use a hierarchy with several levels.

Approach

Logical address space is broken up into multiple page tables.

Implementation

One of the simplest techniques using two-level or three-level page tables.



Two Level Page Table

Consider a system having 32-bit logical address space and page size of 1 KB:

Initial Division

- **Page Number: 22 bits**
- **Page Offset: 10 bits**

After Paging the Page Table

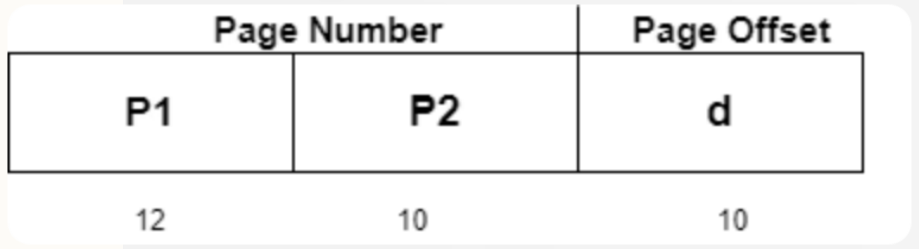
- **Page Number: 12 bits**
- **Page Offset: 10 bits**

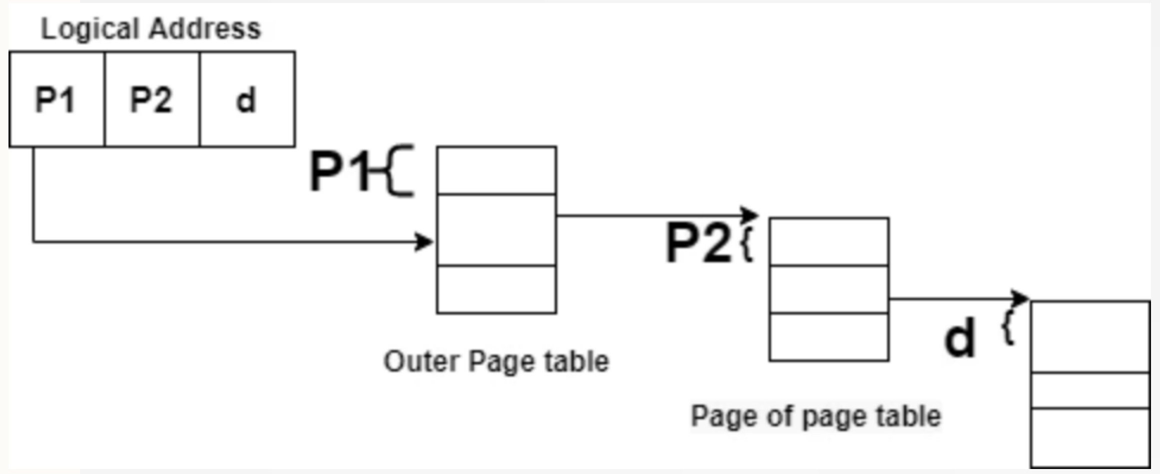
Logical Address Structure

P1: Index into Outer Page table

P2: Displacement within page of Inner page table

Address translation works from outer page table inward (forward-mapped).





Three Level Page Table

For a system with 64-bit logical address space, a two-level paging scheme is not appropriate. With page size of 4KB, dividing the outer page table results in a three-level page table.

1. **Two-Level Issue**

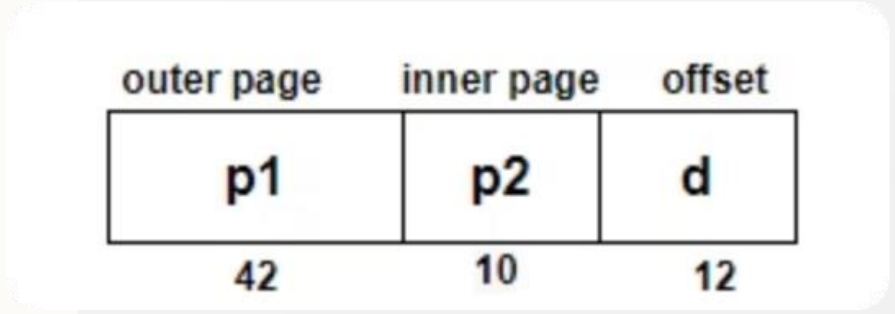
   64-bit address space creates excessively large outer page table

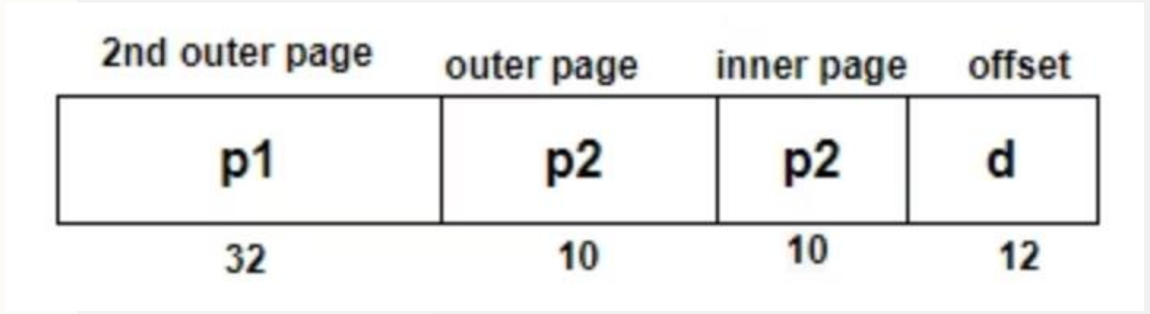
2. **Solution**

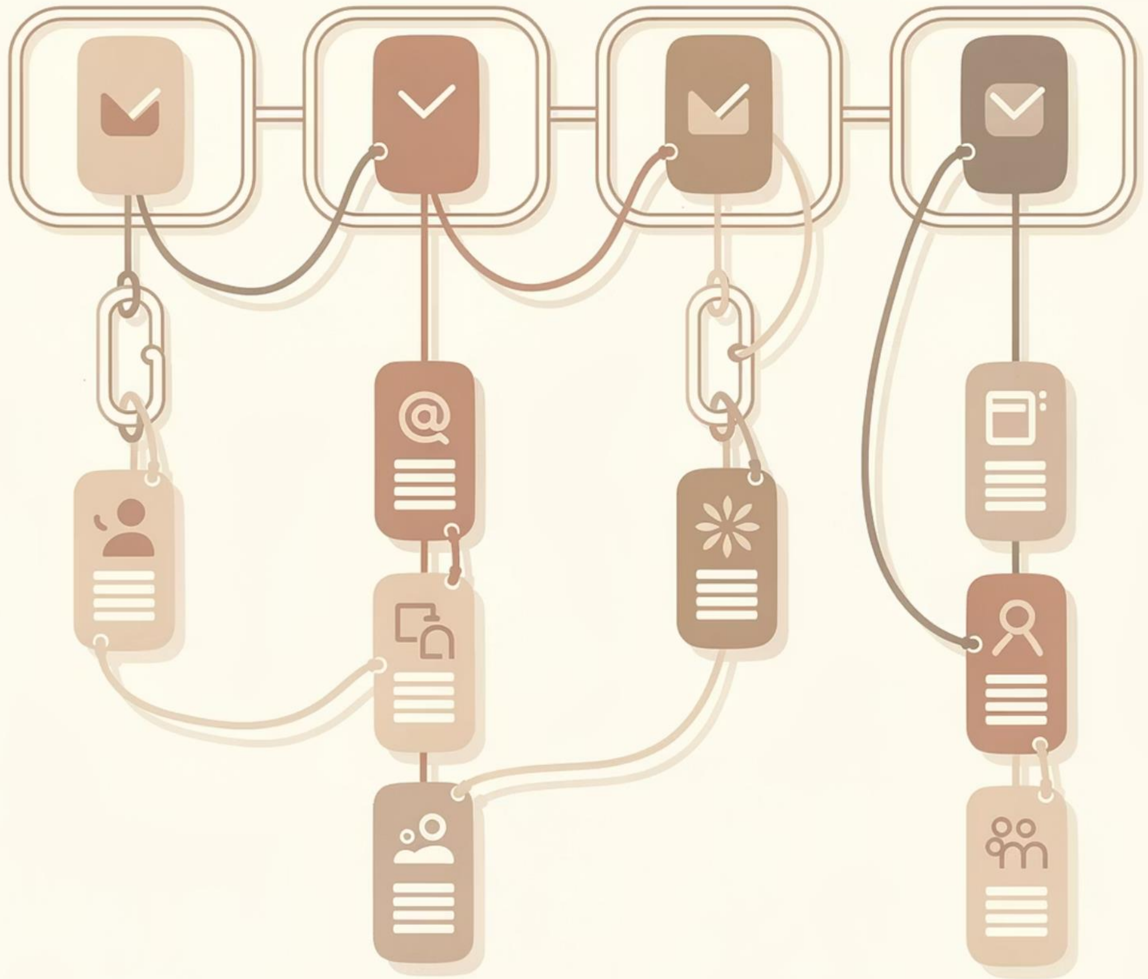
   Divide the outer page table into additional level

3. **Result**

   Three-level page table structure manages large address spaces efficiently







Hashed Page Tables

This approach is used to handle address spaces larger than 32 bits. The virtual page number is hashed into a page table containing a chain of elements hashing to the same location.

1 **Virtual Page Number**

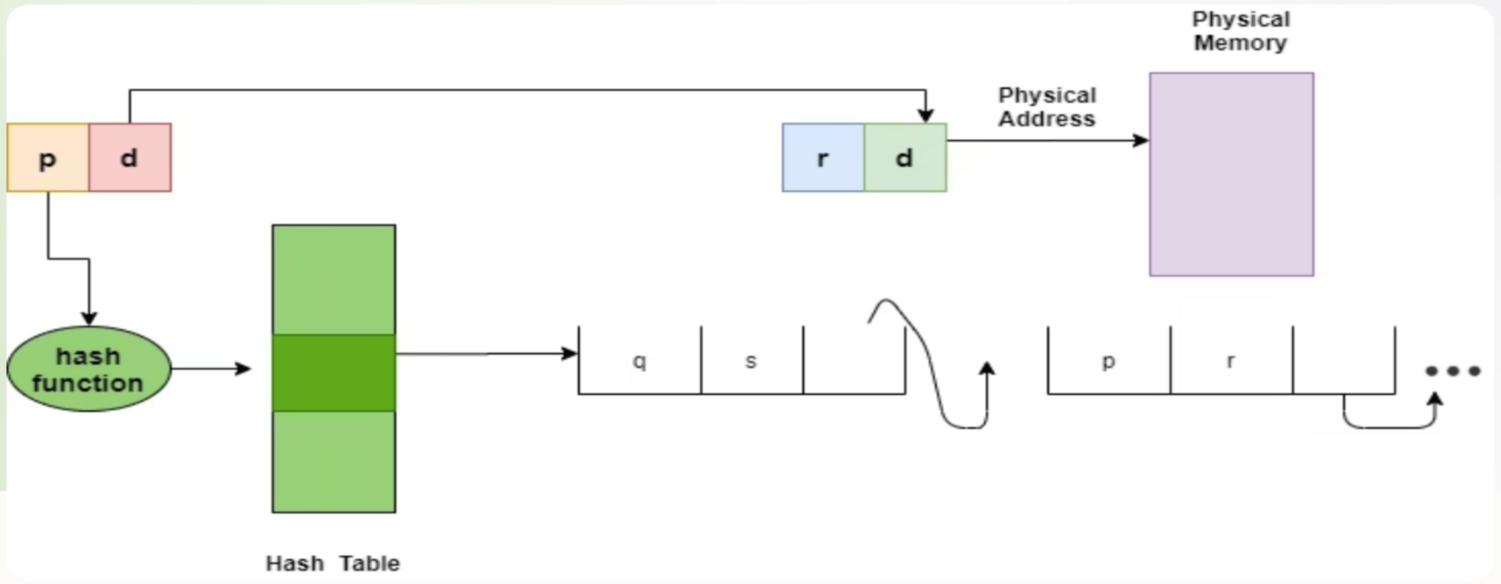
The virtual page number of the address

2 **Mapped Page Frame**

The value of the mapped page frame

3 **Pointer**

A pointer to the next element in the linked list



Hashed Page Table Address Translation

Translation Process

Virtual page numbers are compared in the chain searching for a match.

If match is found, corresponding physical frame is extracted.

Clustered Page Tables

Variation for 64-bit address space where each entry refers to several pages (e.g., 16) rather than 1.

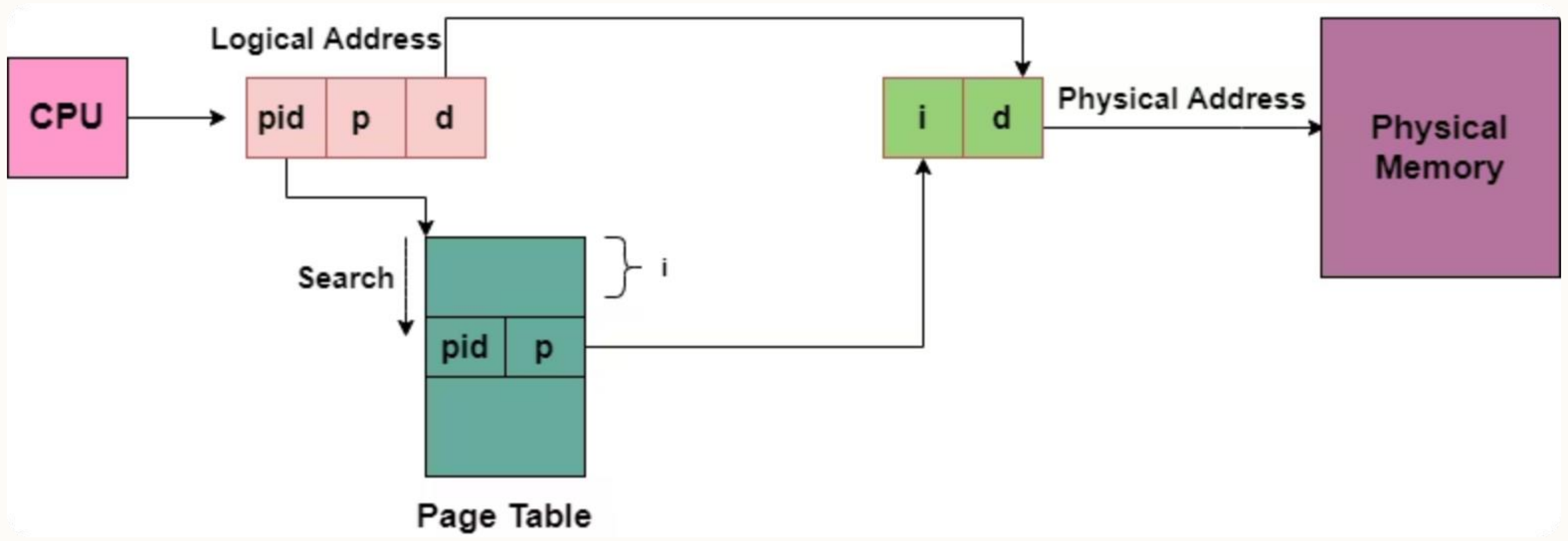
Used for sparse address spaces with non-contiguous, scattered memory references.

Inverted Page Tables

The Inverted Page table combines a page table and a frame table into a single data structure.

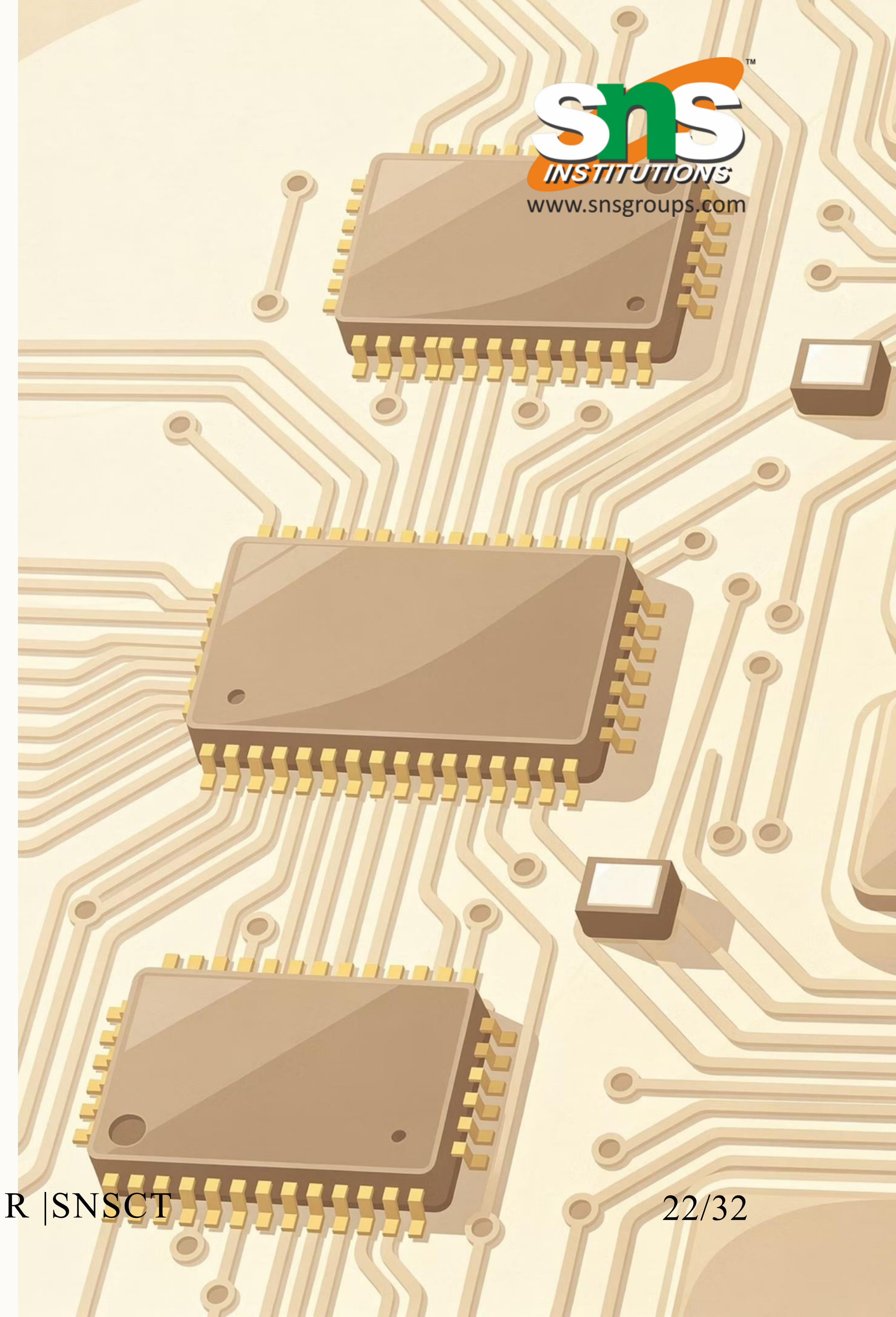
Structure

Trade-off



Paging – Structure of the Page Table

A design thinking approach to understanding memory management



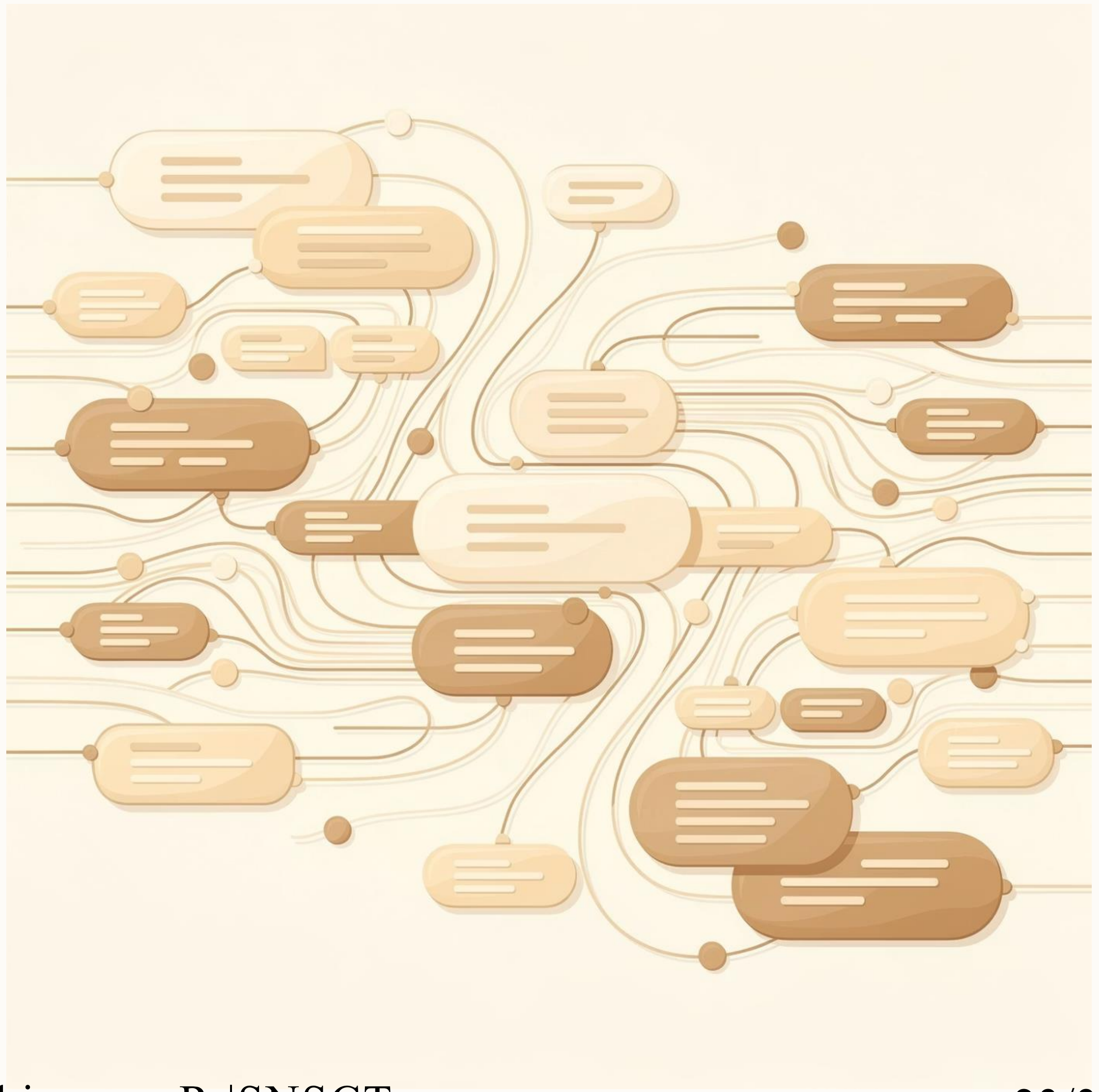
EMPATHIZE

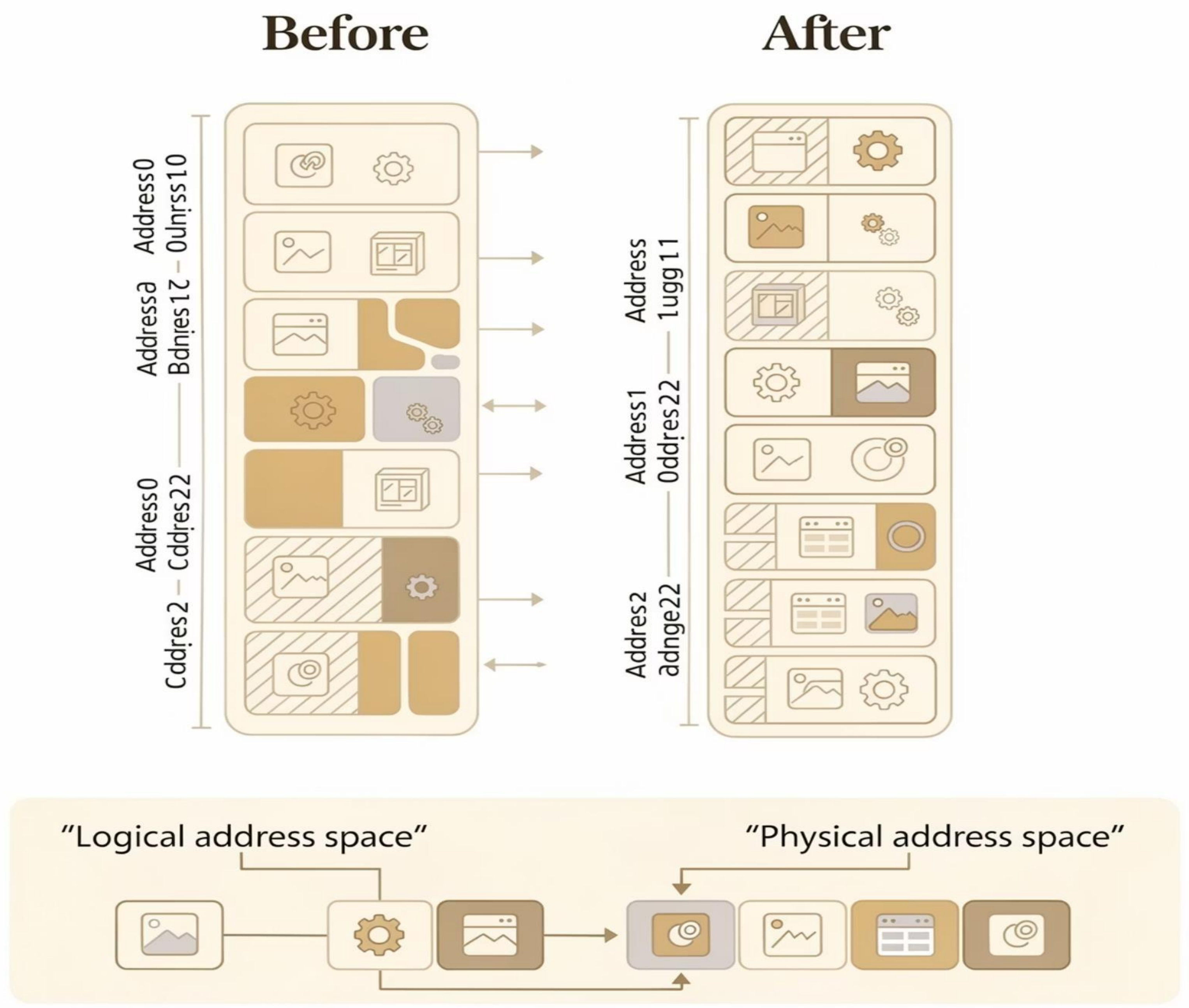
Problem Statement

Modern operating systems must run multiple large programs simultaneously in limited main memory. If programs are loaded contiguously, memory gets wasted due to external fragmentation, and processes may fail even when enough total memory exists.

The Core Challenge

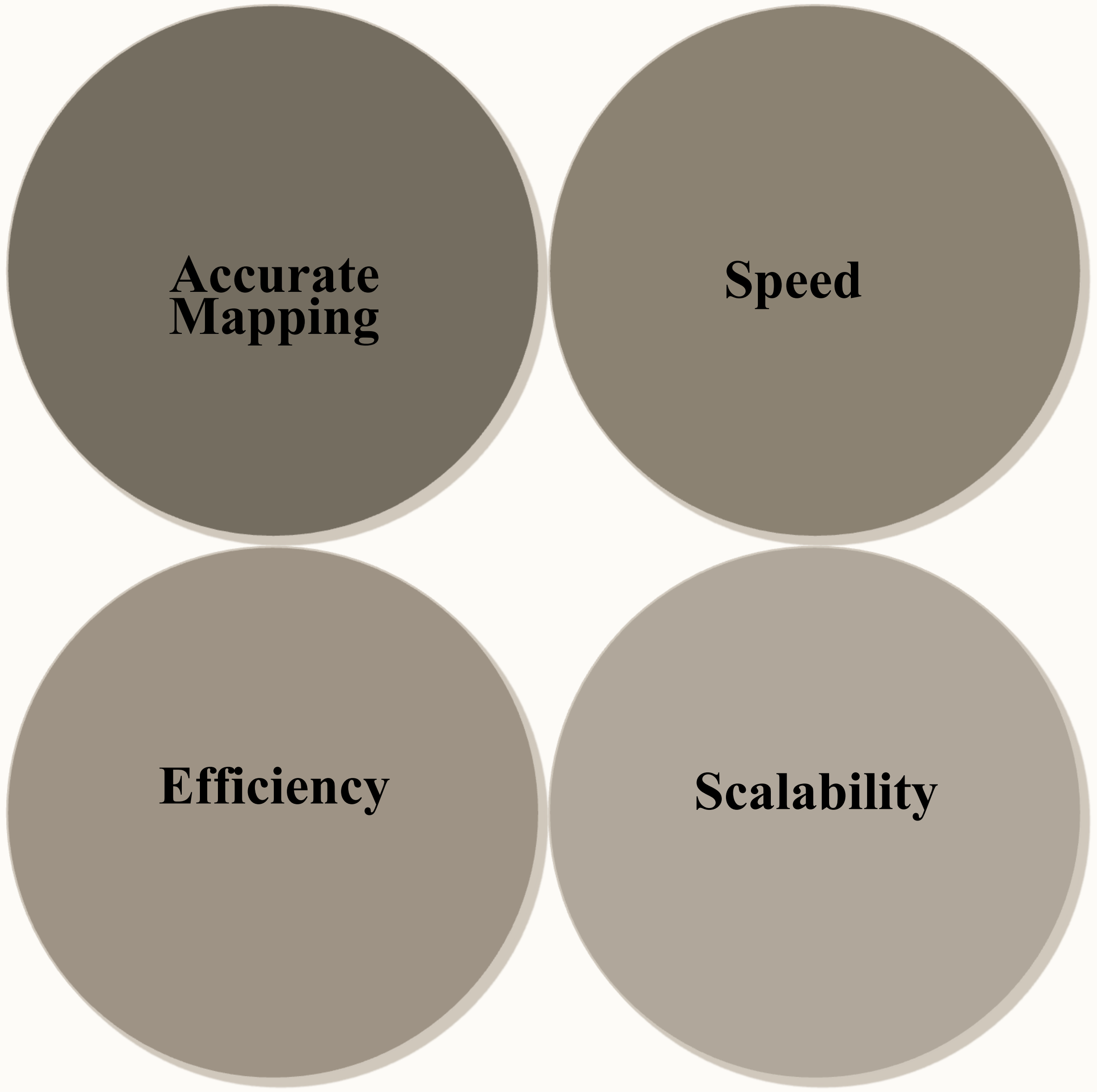
How can the OS efficiently store and access a process divided into pages, while quickly translating logical addresses into physical addresses without wasting memory or slowing down execution?




Before
After
Address0
0unrss10
Addressə
Bdnies12
Address0
Cddres22
Cddres2
Address
1ugg11
Address1
0ddres22
Addres2
adnge22
"Logical address space"
"Physical address space"

DEFINE

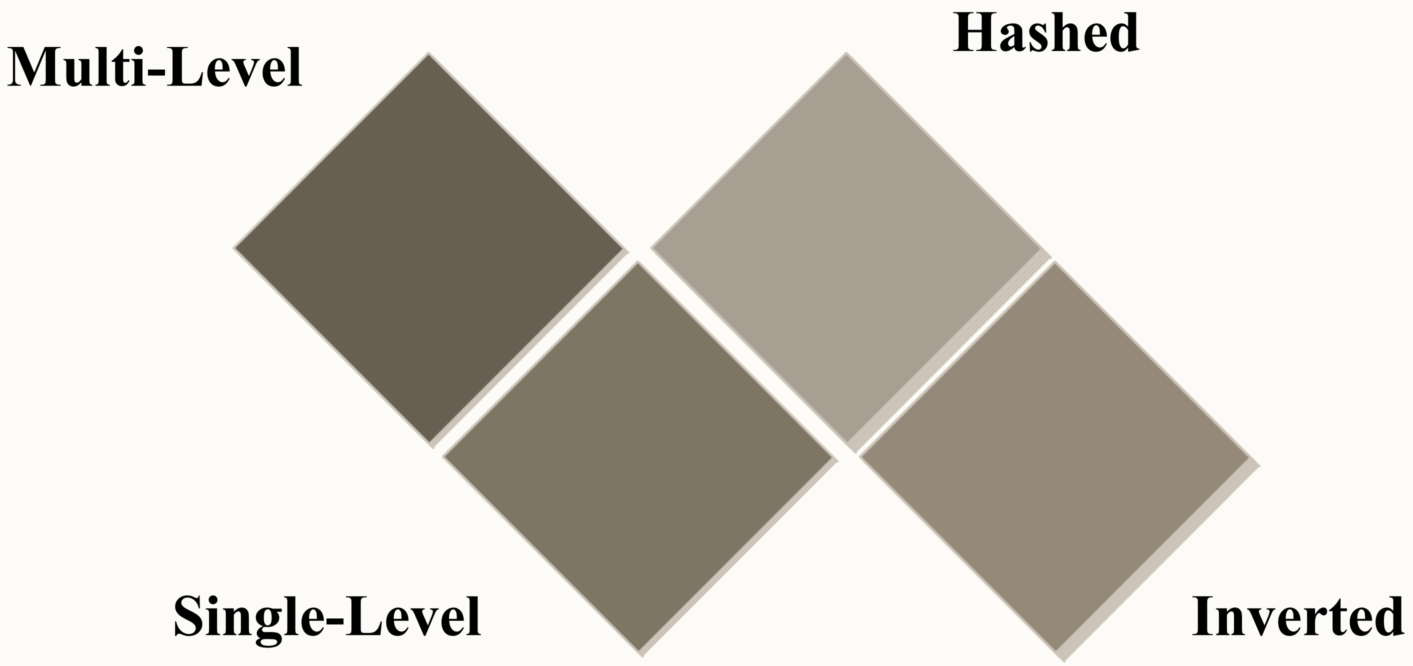
Problem Definition Stage



IDEATE

Solution Exploration Stage

Multiple architectural approaches exist for structuring page tables, each with distinct trade-offs:



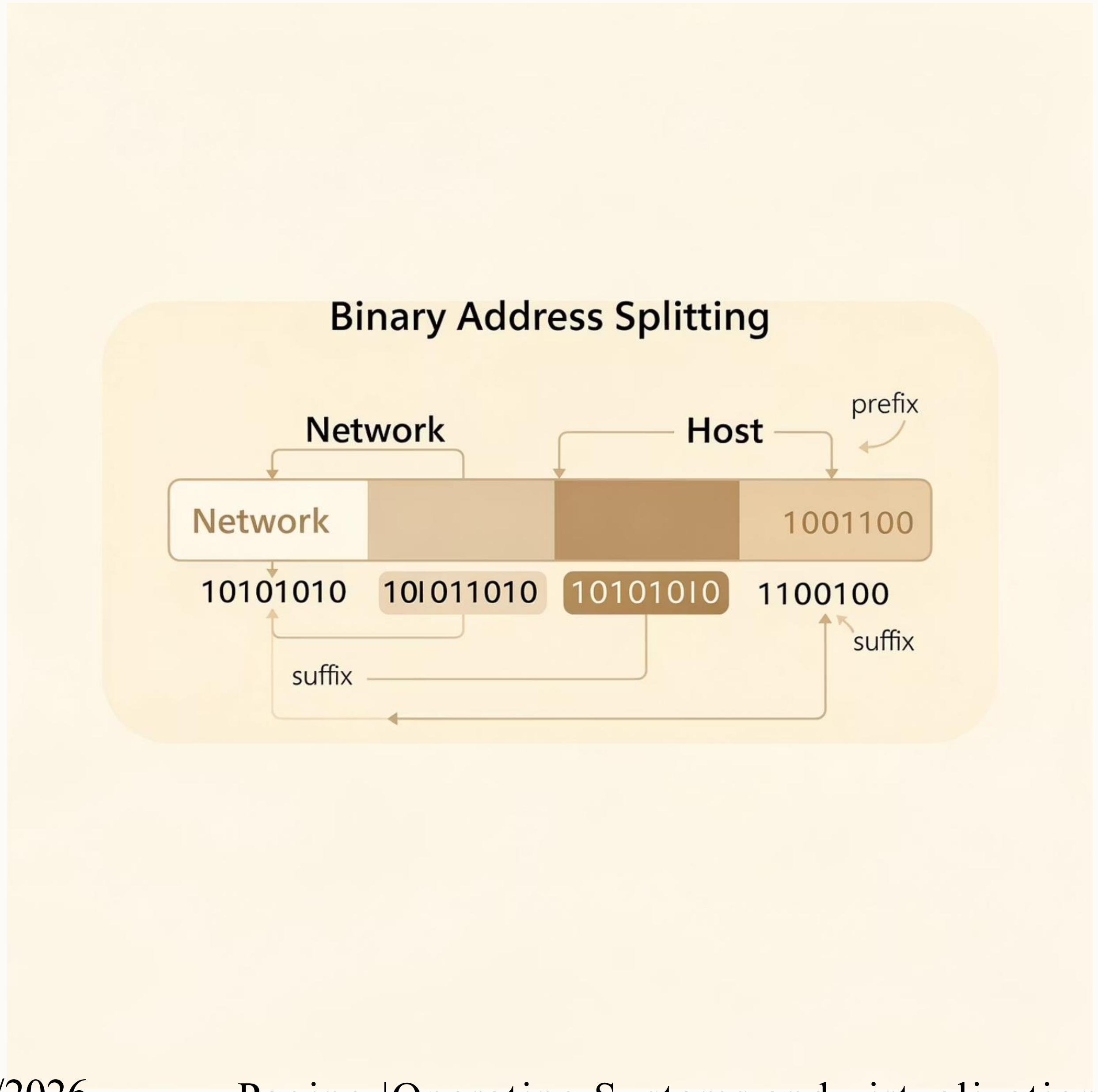
PROTOTYPE

Structure of a Page Table Entry

Each Page Table Entry (PTE) contains multiple fields that enable effective memory management and protection:

Field	**Description**
Page Number	Logical page number for indexing
Frame Number	Corresponding physical frame in main memory
Valid/Invalid Bit	Indicates whether page is currently in memory
Protection Bits	Defines Read, Write, and Execute permissions
Dirty Bit	Flags page modification for write-back decisions
Reference Bit	Used by page replacement algorithms

Logical Address Format



A logical address generated by the CPU consists of two distinct components that work together for address translation:

Page Number

Indexes into the page table to locate the corresponding entry

Offset

Specifies the exact position within the selected page

Address Translation Process

The memory management unit translates logical addresses to physical addresses through a systematic process:

1

CPU Generation

CPU generates a logical address during program execution

2

Page Table Lookup

Page number indexes the page table to find the entry

3

Frame Retrieval

Frame number is obtained from the page table entry

4

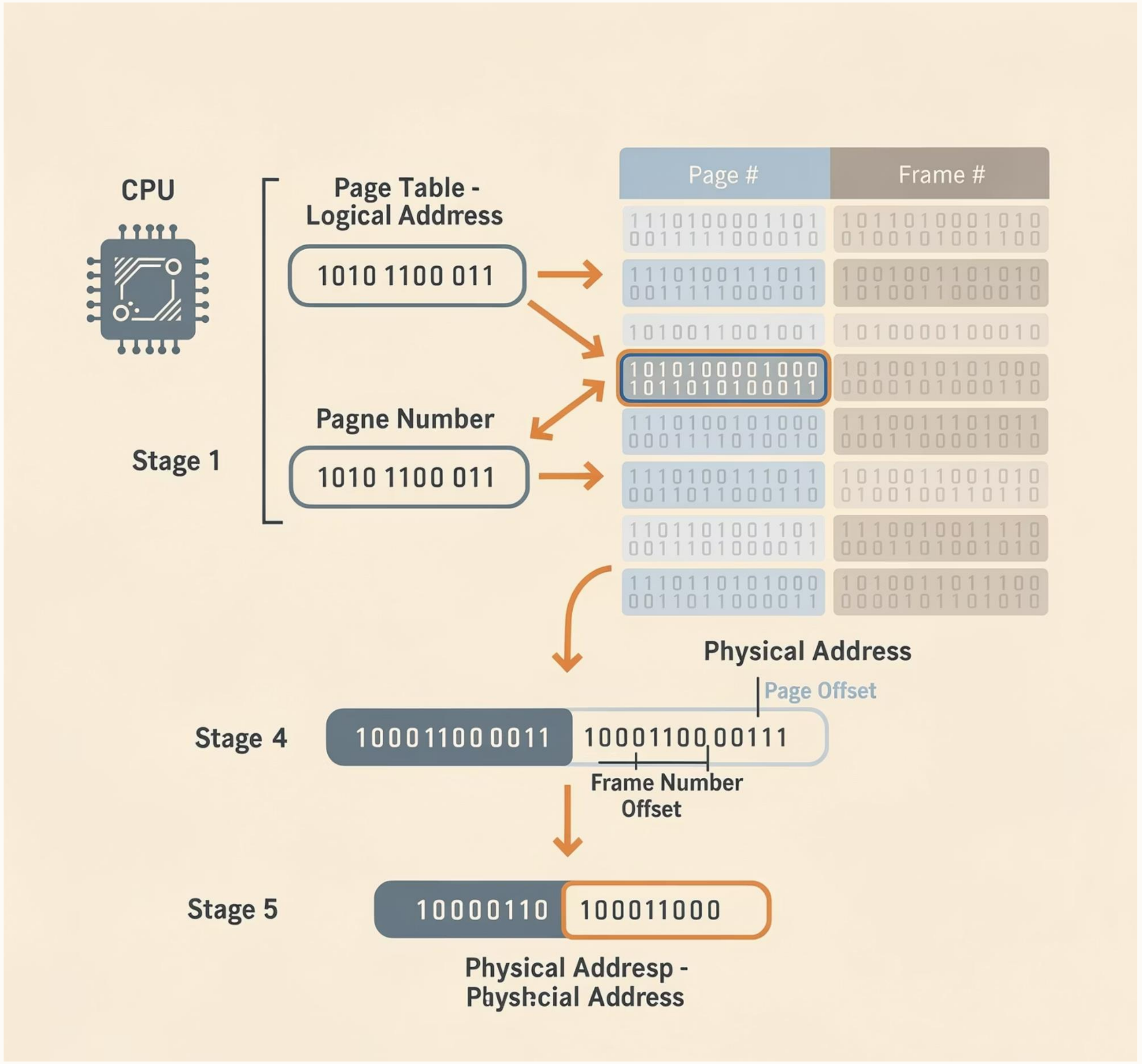
Physical Address

Physical address = Frame number concatenated with Offset

5

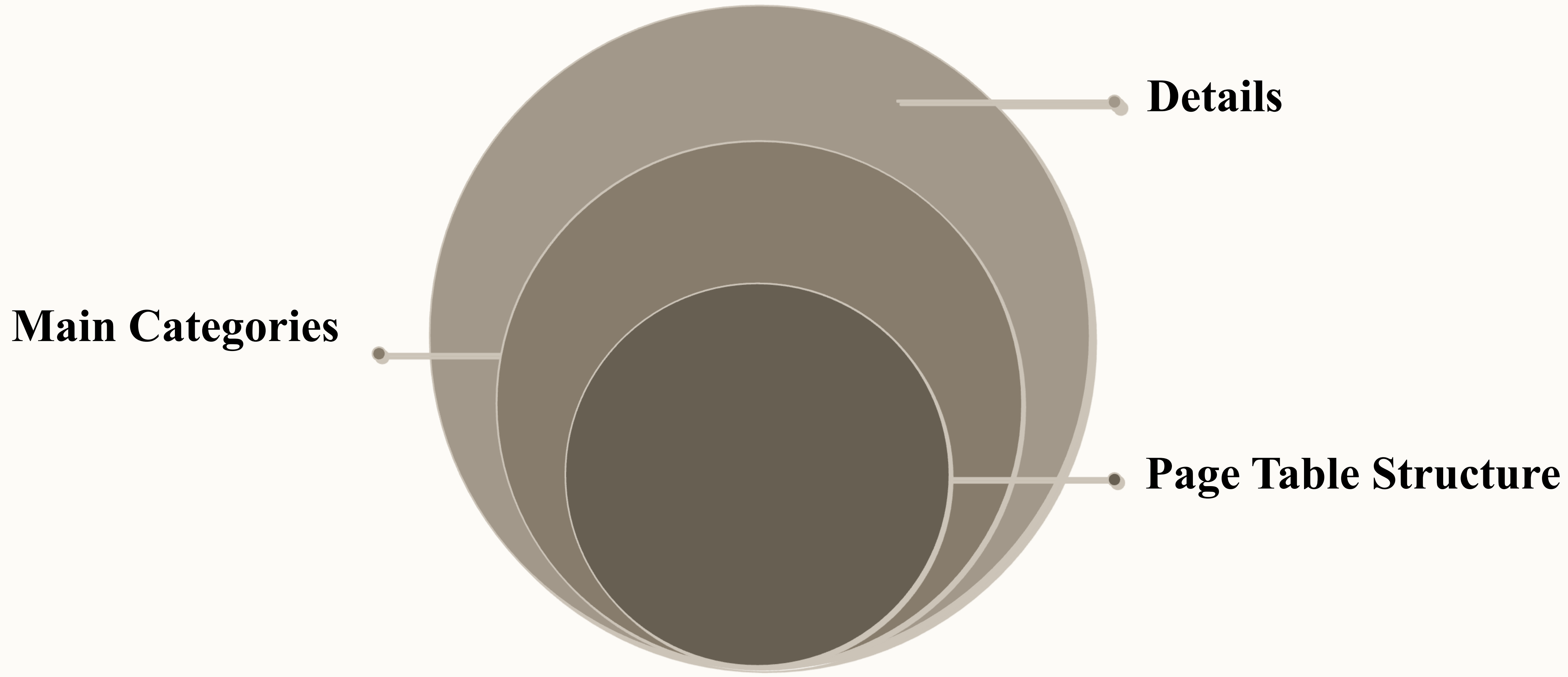
Exception Handling

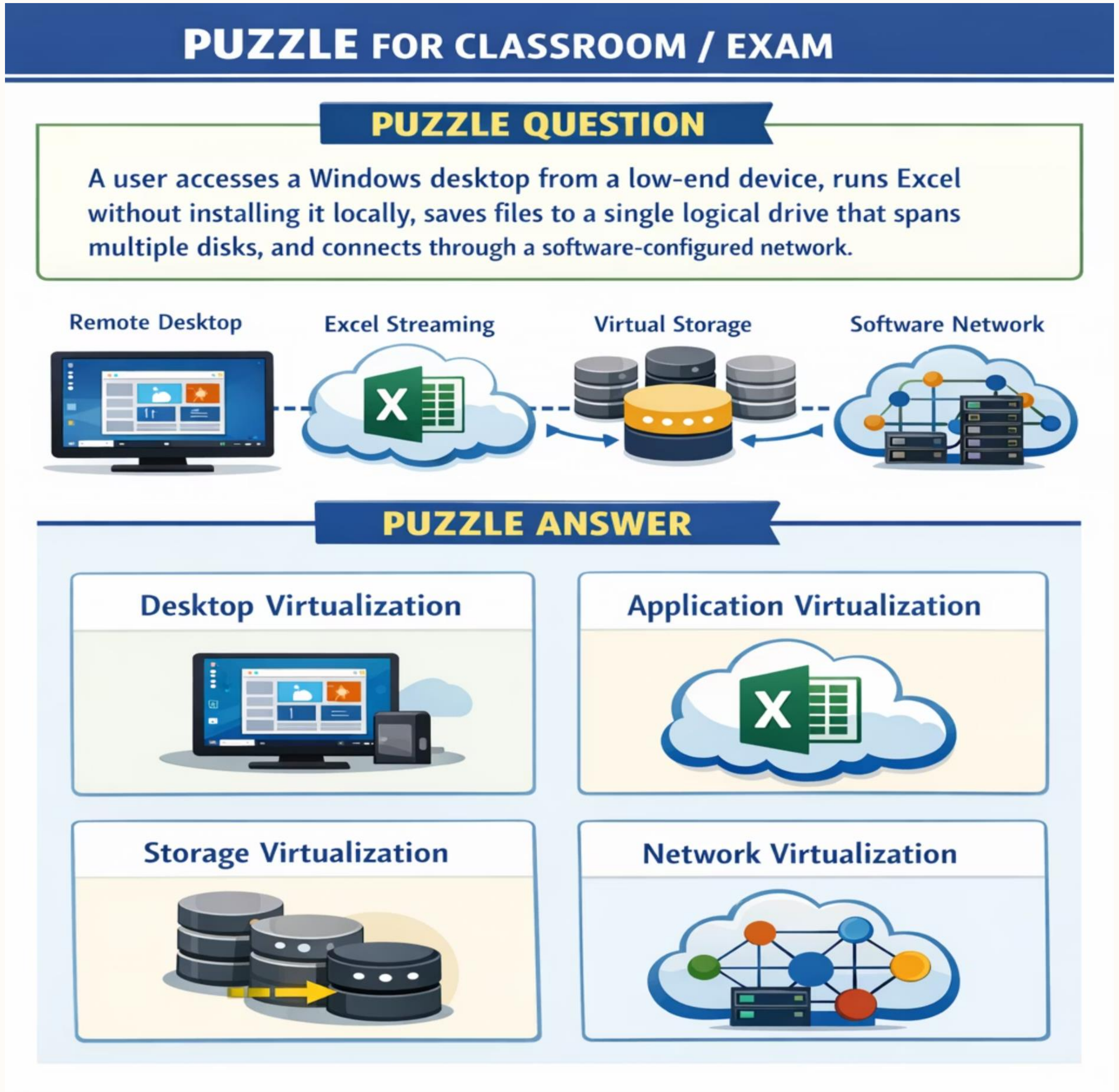
If page not present, trigger Page Fault interrupt



Page #	Frame #
1110100001101 0011111000010	1011010001010 0100101001100
1110100111011 0011111000101	1001001101010 1010011000010
1010011001001	1010000100010
1010100001000 1011101010011	1010010101000 0000101000110
1110100101000 0001111010010	1110011101011 0001100001010
1110100111011 0011011000110	1010011001010 0100100110110
1101101001101 0011101000011	1110010011110 0001101001010
1110110101000 0011011000011	1010011011100 0000101101010

Page Table Structure Mind Map





PUZZLE FOR CLASSROOM / EXAM

PUZZLE QUESTION

A user accesses a Windows desktop from a low-end device, runs Excel without installing it locally, saves files to a single logical drive that spans multiple disks, and connects through a software-configured network.

Remote Desktop
Excel Streaming
Virtual Storage
Software Network

PUZZLE ANSWER

Desktop Virtualization

Application Virtualization

Storage Virtualization

Network Virtualization