Virtual Memory Management Tool

A simulation-based tool that visualizes virtual memory concepts like paging, segmentation, and page replacement. Designed to enhance understanding of memory management through interactive experimentation.

Introduction

Purpose of the Project

- This project aims to simulate and visualize the behavior of virtual memory management in modern operating systems.
- It provides an interactive platform to understand complex concepts like paging, segmentation, demand paging, and memory allocation.
- Users can explore how memory is organized, how page faults occur, and how different page replacement algorithms function.

Motivation

- Concepts like paging, segmentation, and page faults are difficult to grasp through theory alone.
- Many students find it challenging to visualize how virtual memory works behind the scenes.
- This tool is developed to bridge that gap by turning abstract ideas into visual, interactive experiences, making learning more effective and engaging.

Objectives

Simulate Paging and Segmentation

Provide a visual and interactive demonstration of how paging and segmentation divide and manage memory.

Visualise Page Faults and Demand Paging

Show how and when page faults occur, and how pages are loaded on demand during execution.

Allow User Input for Custom Memory Allocation

Let users define memory sizes, process requirements, and simulate real-world memory scenarios.

Demonstrate Memory Fragmentation

Simulate both internal and external fragmentation during allocation and deallocation processes.

Evaluate and Compare Page Replacement Algorithms

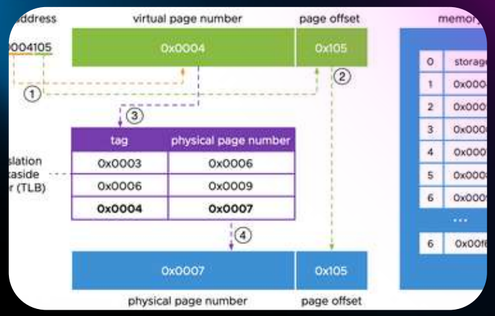
LRU (Least Recently Used)

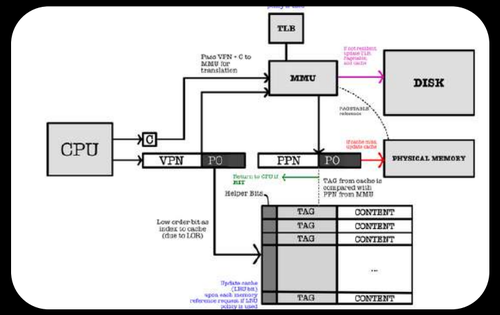
Optimal Replacement Algorithm

What is Virtual Memory?

Definition

Virtual memory is a memory management technique that gives an application the illusion of having a large, continuous block of memory, even if the physical memory is limited. It uses both RAM and disk space to extend available memory.



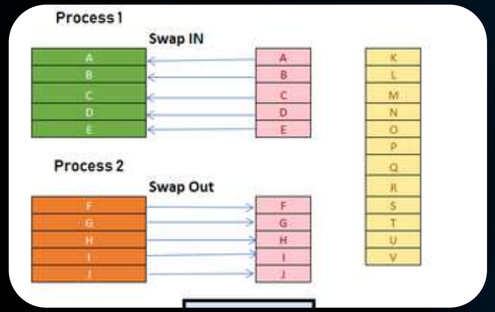


Virtual Memory vs Physical Memory

- Physical Memory: Actual RAM installed in the system.
- Virtual Memory: Logical memory created by the OS using paging and segmentation, partially stored on disk.

Benefits of Virtual Memory

- Enables multitasking by allowing multiple processes to run concurrently.
- Supports efficient memory utilization by loading only required pages.
- Isolates process memory, ensuring better security and stability.



Paging and Segmentation Overview

Paging

- Memory is divided into fixed-size blocks:
  a. Pages (logical memory)
  b. Frames (physical memory)
- Pages are mapped to frames using a page table.
- Eliminates external fragmentation but may cause internal fragmentation if page size > process requirement.
- Efficient memory use with fixed-size blocks, commonly used in modern OS.

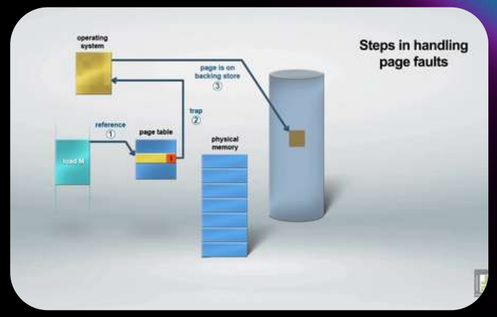
Segmentation

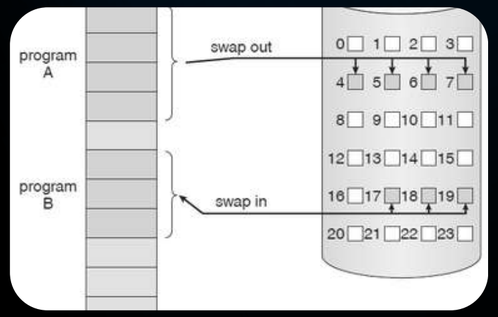
- Memory is divided into variable-sized logical units, like:
  - Code, Stack, Heap, Data
- Each segment has its own base and limit.
- Allows better logical organisation of memory.
- Can result in external fragmentation due to variable segment sizes.

Page Faults and Demand Paging

What is a Page Fault?

- A page fault occurs when a program tries to access a page that is not currently in physical memory (RAM).
- The operating system must then fetch the required page from secondary storage (disk) and load it into memory.



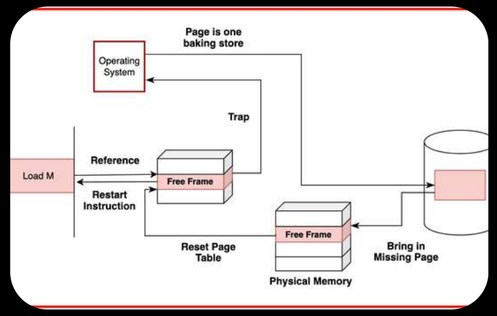


What is Demand Paging?

- A memory management strategy where pages are loaded into memory only when they are needed, not in advance.
- Reduces memory usage and improves performance for large programs.

How Our Tool Simulates It

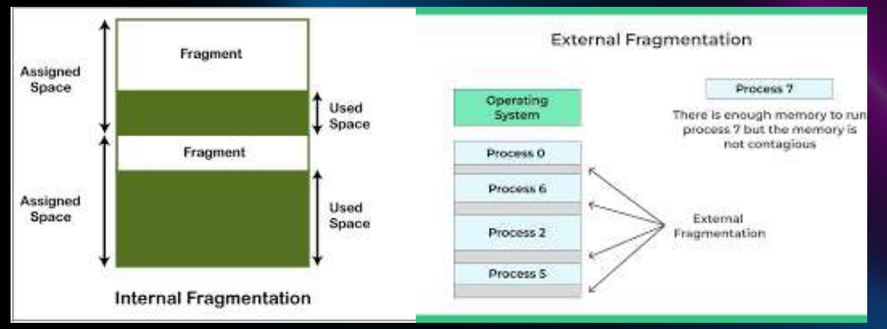
- Tracks when a page fault occurs during program execution.
- Visual indicators highlight page loading steps and fault handling.
- Users can observe demand paging in action with real-time memory state updates.



Memory Allocation and Fragmentation

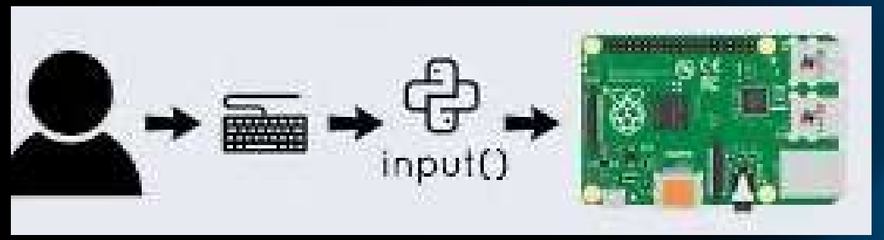
Types of Fragmentation

Internal Fragmentation:
Occurs when memory is allocated in fixed-sized blocks, and the process doesn't use the entire block, leaving unused space inside.
External Fragmentation:
Happens when there’s enough total memory, but it is scattered in small chunks, making it hard to allocate to larger processes.



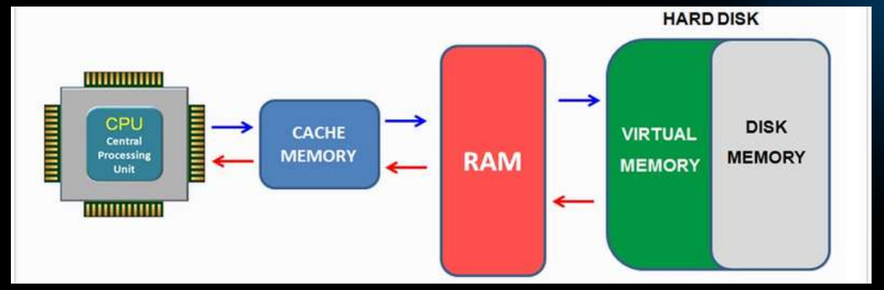
User Input in the Tool

- Define custom total memory size.
- Specify individual process memory requirements.
- Observe how different allocations affect memory utilization.



Tool Simulates:

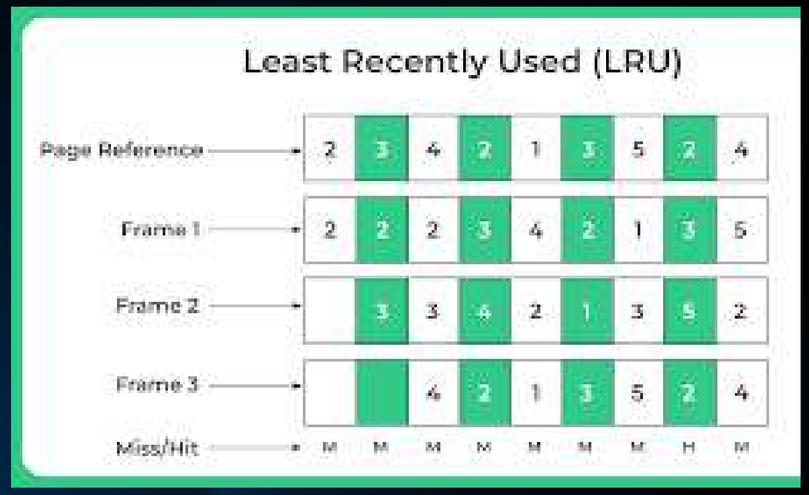
- Real-time memory allocation and free space tracking.
- Highlights when and where fragmentation occurs.



Page Replacement Algorithms

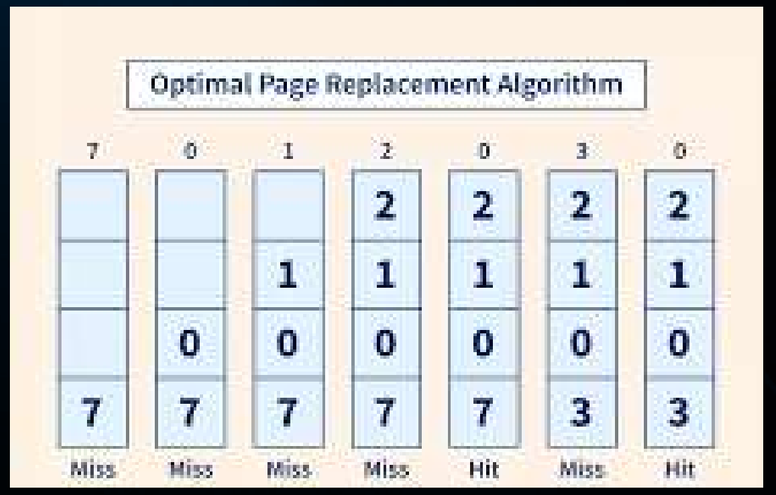
LRU (Least Recently Used):

- Replaces the least recently accessed page in memory.
- Based on the assumption that recently used pages will likely be used again.
- Requires tracking of usage history (e.g., timestamps or stack simulation).



Optimal Replacement:

- Replaces the page that will not be used for the longest time in the future.
- Yields minimum number of page faults, but requires future knowledge (ideal for simulation/comparison purposes).



Features of the Tool

User-Friendly Interface

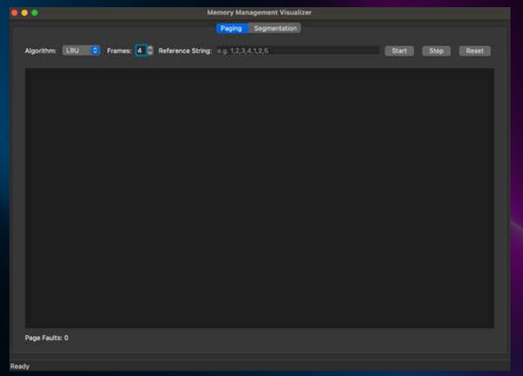
- Simple and interactive Graphical User Interface (GUI).
- Allows users to easily input memory size, number of processes, and algorithm choice.

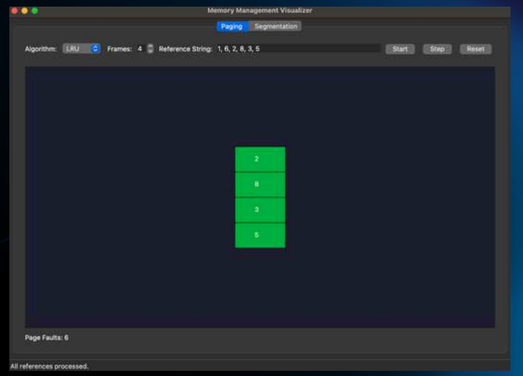
Real-Time Simulation Includes

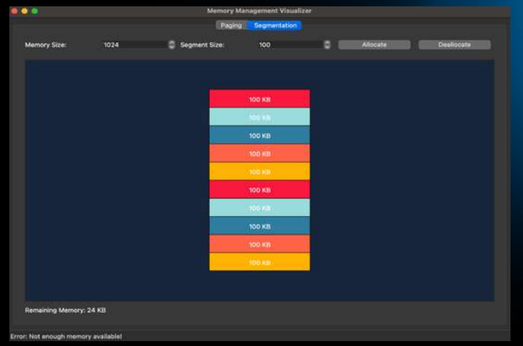
- Paging and Segmentation: Dynamic visualization of how memory is divided and managed.
- Page Faults: Instant feedback when a fault occurs, with visual indicators.
- Demand Paging: See how pages are loaded only when required.
- Fragmentation Effects: Visual tracking of internal and external fragmentation during memory allocation.

Algorithm Comparison Graph

- Displays page fault counts and efficiency metrics for LRU and Optimal algorithms.
- Helps users understand the performance difference through visual insights.







Technologies Used

Programming Language: Python

Used as the core language to implement the simulation logic, memory handling, and interactive GUI behavior.

Libraries and Tools:

- **PyQt5**
  - Framework for building the Graphical User Interface (GUI).
  - Enables creation of interactive windows, buttons, labels, and graphics views.
- **QtWidgets (from PyQt5)**
  Provides essential GUI elements like:
  - QMainWindow – Main application window
  - QPushButton – Button controls
  - QLabel – Text and image labels
  - QGraphicsView – Custom graphical memory visualization
- **QtGui (from PyQt5)**
  Supplies graphical styling and rendering elements:
  - QColor – For defining and customizing colors
  - QBrush – For rendering filled shapes and memory blocks
- **sys**
  Used to manage system-level operations such as application startup and shutdown.

Challenges and Solutions

Challenges Faced:

- **Accurate Simulation Logic:**
  Simulating realistic behaviors of paging, segmentation, and page replacement required precise memory modeling and attention to detail in data flow.

- **GUI Integration:**
  Integrating real-time simulation with a dynamic GUI using PyQt5 posed challenges in managing layout updates, performance, and user interactivity.

Solutions Implemented:

- **Modular Design Approach:**
  Divided the tool into separate logic, UI, and visualization components, making the system easier to build, test, and maintain.
- **Visual Feedback with Data Structures:**
  Used clear visual representations of memory states linked directly to underlying data structures, improving both accuracy and user understanding.

Learning Outcomes

Conceptual Understanding:

- Gained in-depth knowledge of virtual memory management, including how operating systems handle memory efficiently.
- Developed a strong grasp of paging and segmentation techniques used to manage memory allocation and access.
- Understood how page faults occur and how demand paging optimizes memory usage.
- Explored and compared page replacement strategies like LRU and Optimal, learning their strengths and trade-offs.

Practical Skills:

- Strengthened programming skills using Python and PyQt5 for real-time simulation and GUI development.
- Enhanced debugging and problem-solving abilities by tackling complex logic integration with interactive UI components.
- Learned the importance of modular design and data visualization in making technical concepts user-friendly.

Future Scope

Support for Additional Page Replacement Algorithms:

Provide a visual and interactive demonstration of how paging and segmentation divide and manage memory.

Multi-Threaded Simulation:

Show how and when page faults occur, and how pages are loaded on demand during execution.

Web-Based Version:

Let users define memory sizes, process requirements, and simulate real-world memory scenarios.

These additions will expand the tool’s capability, improve learning engagement, and promote accessibility beyond desktop users.

CONCLUSION

Designed and developed a virtual memory management tool that simulates paging, segmentation, page faults, and demand paging. Supports user-customised memory inputs, real-time visualizations, and page replacement algorithm comparisons(LRU & Optimal). Bridges the gap between theoretical operating system concepts and practical understanding. Makes abstract memory management topics interactive, visual, and easier to grasp. An effective educational aid for students and teachers. Promotes deeper learning through hands-on experimentation and intuitive visuals.


Thank you