VIRTUAL MEMORY SIMULATOR

A MINI-PROJECT REPORT

Submitted by

SARAVANAN MD	**231901046**
UDAY KRISHNA N	**231901057**

in partial fulfilment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY)

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI
MAY 2025

BONAFIDE CERTIFICATE

Certified that this project report "**VIRTUAL MEMORY SIMULATOR**" is the Bonafide work of **"SARAVANAN MD (231901046), UDAY KRISHNA N (231901057)"** who carried out the project work under my supervision.

Submitted for the Practical Examination held on ______________

SIGNATURE

Ms.V.JANANEE

Assistant Professor (SG),

Computer Science and Engineering,

Rajalakshmi Engineering College,

(Autonomous),

Thandalam, Chennai - 602 105.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR.S.MEGANATHAN** and the chairperson **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

We express our sincere thanks to our Head of the Department **MR. Benedict JN**, for encouragement and being ever supporting force during our project work.

We also extend our sincere and hearty thanks to our internal guide **Ms.V.JANANEE,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1.SARAVANAN MD	231901046
2.UDAY KRISHNA N	231901057

ABSTRACT

Virtual memory management is a crucial component of modern operating systems, enabling efficient resource utilization by allowing processes to use more memory than physically available. This project implements a Virtual Memory Page Replacement Simulator using C programming to demonstrate the working of FIFO (First-In-First-Out) and LRU (Least Recently Used) page replacement algorithms. The simulator accepts a page reference string and a specified number of frames, then simulates the behavior of these algorithms while tracking page faults.

The project provides a step-by-step visualization of memory frame updates and highlights the efficiency of each algorithm in handling memory management. It helps users understand how page replacement policies affect system performance. The implementation can be extended to include other algorithms like Optimal Page Replacement. This project is ideal for students and learners looking to gain practical knowledge in operating system concepts, virtual memory, and memory management techniques.

TABLE OF CONTENTS

CHAPTER –1

INTRODUCTION

Virtual memory is a fundamental concept in modern operating systems, allowing efficient and secure memory management by providing an abstraction of more memory than is physically available. It enables features like multitasking, memory protection, and process isolation by mapping logical addresses used by applications to physical memory locations. Understanding the internal operations of virtual memory—especially page replacement strategies such as Least Recently Used (LRU) and First-In-First-Out (FIFO)—is crucial for students pursuing studies in system-level programming and computer architecture.
This project presents a **Virtual Memory Simulator** with a **real-time graphical user interface (GUI)** designed using GTK. The simulator visually demonstrates how virtual memory works, including page requests, page faults, and replacement strategies. By combining a powerful backend logic with an interactive and user-friendly interface, the project aims to enhance conceptual understanding through visualization. It serves as an educational tool to bridge the gap between theoretical learning and practical comprehension, helping users gain a deeper insight into the mechanics of memory management in operating systems.

1.1 Key Features:

1. **Virtual Memory Simulation**
   Simulates real-time memory management, including page loading, faults, and replacements using FIFO and LRU algorithms.
2. **Graphical User Interface (GUI)**
   Built using GTK in C, the GUI provides a clean, interactive visual display of memory frames and page replacement in action.
3. **Dual Modes of Operation**
   Supports both terminal-based (ncurses) and graphical (GTK) modes for flexible usage and demonstration.
4. **Real-Time Visualization**
   Updates frame status live as page requests are processed, showing which frames are ccupied or replaced.
5. **Educational Focus**
   Designed as a teaching aid for students learning Operating Systems concepts like paging, page faults, and memory allocation.
6. **Modular Design**
   Separation of core logic and interface code allows easy updates and extensions, such as adding more algorithms or features.
7. **Customizable Simulation**
   Page request patterns can be edited easily to simulate different types of memory access behaviors.

CHAPTER –2

ARCHITECTURE

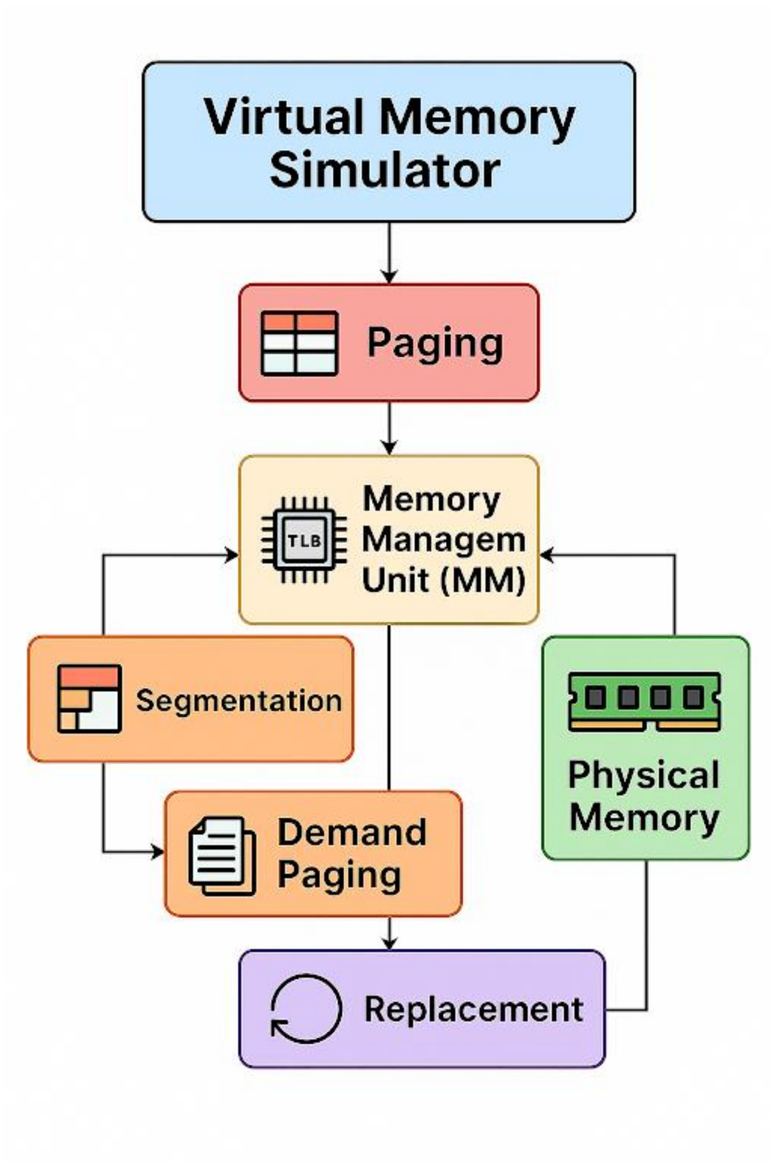

Figure 2.1 – Architecture of Virtual Memory Simulator

.

CHAPTER –3

SCOPE OF THE PROJECT

This project aims to simulate the internal functioning of an operating system's virtual memory management using real-time visualization. It models how memory frames are allocated to pages, handles page faults, and demonstrates page replacement using FIFO and LRU algorithms. The inclusion of a graphical user interface built with GTK in C makes the simulation interactive and educational, helping users observe and understand memory operations as they happen. By supporting both terminal and GUI modes, the project serves as a practical learning tool for students studying operating systems, computer architecture, and memory management.

CHAPTER –4

FLOW CHART

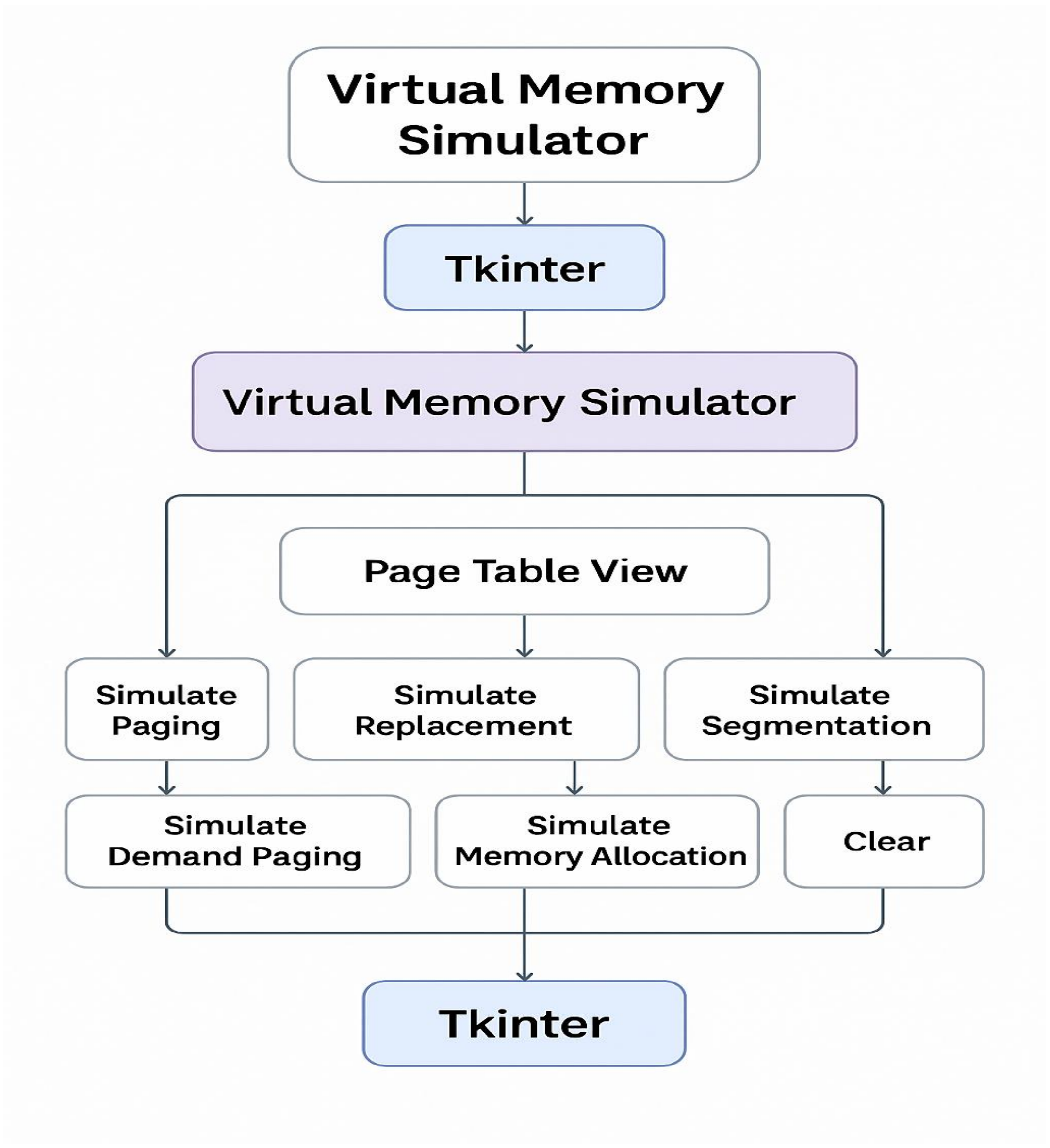

Figure 3.1 – Flowchart of Virtual Memory Simulator

1. VIRTUAL MEMORY SIMULATOR (MAIN APP)
entry point of the application using tkinter, the GUI library in python.

2. TKINTER INITIALIZATION
sets up the main window and widgets like labels, frames, and buttons.

3. PAGE TABLE VIEW
displays simulation results in a structured table format using ttk.treeview.

4. SIMULATION OPTIONS
Various simulation buttons trigger respective functions:

- Simulate paging: assigns pages to random memory frames.
- Simulate replacement: randomly selects a page replacement algorithm (fifo, lru, optimal).
- Simulate segmentation: displays segments like code/data/stack with base and limit.
- Simulate demand paging: shows which pages are loaded into frames dynamically.
- Simulate memory allocation: randomly selects allocation strategies (first-fit, best-fit, worst-fit).
- Clear: clears the table for a new simulation.

5. TKINTER GUI LOOP
Continuously runs the app, waiting for user interaction.

○

CHAPTER –5
PROGRAM CODE

def __init__(self, root):

Initializes the virtual memory simulator and sets up the GUI window.

def create_widgets(self):

Creates and arranges the graphical components like labels, buttons, and the memory simulation table**.**

def simulate_paging(self):

Simulates paging by randomly assigning page numbers to memory frames and displays them in the table.

def simulate_page_replacement(self):

Simulates page replacement using a random algorithm (FIFO, LRU, Optimal) and displays a message box with the selected algorithm.

def simulate_segmentation(self):

Simulates segmentation by generating segment names, base addresses, and limits, displaying them in the table**.**

def simulate_demand_paging(self):

Simulates demand paging by randomly loading pages into memory frames and displays their status in the table.

def simulate_memory_allocation(self):

Simulates memory allocation using a randomly chosen strategy (First-Fit, Best-Fit, Worst-Fit) and displays a message box.

def clear_table(self):

Clears all entries in the memory simulation table to reset the display for the next simulation.

CHAPTER –6

Comparison of Memory Management Algorithms

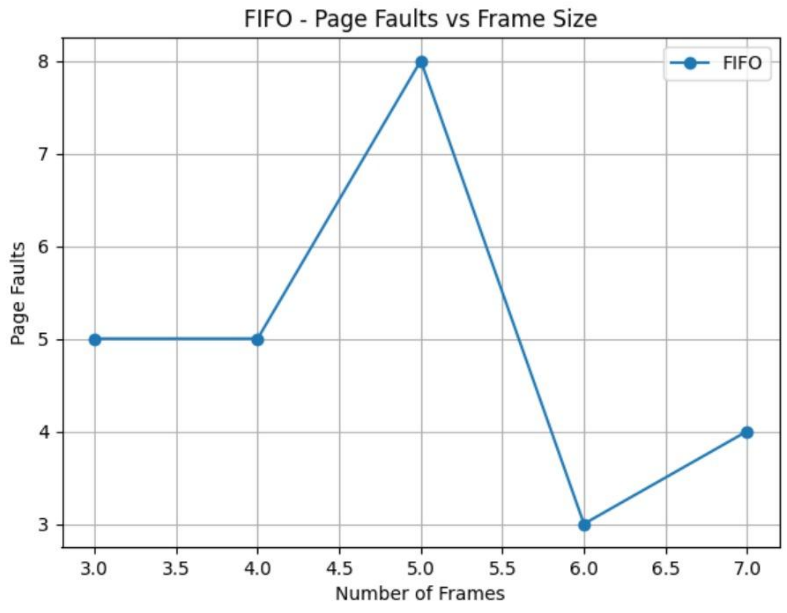

Figure 6.1 – FIFO Algorithm

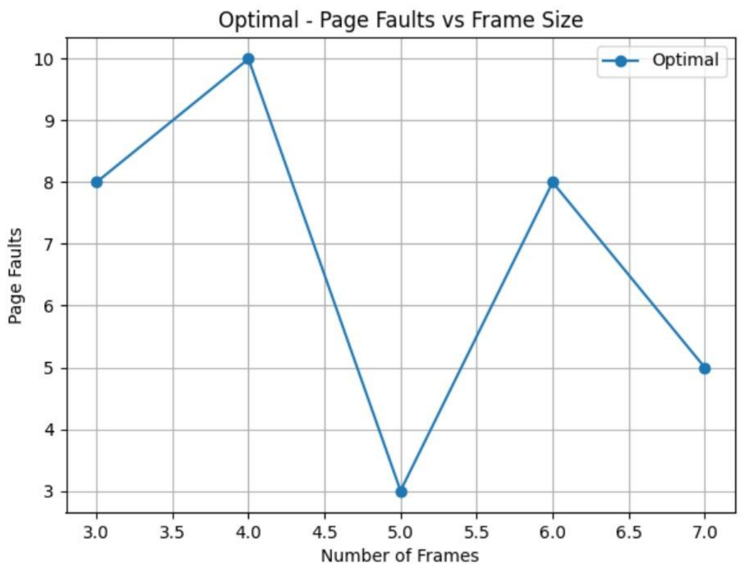

Figure 6.2 – Optimal Algorithm

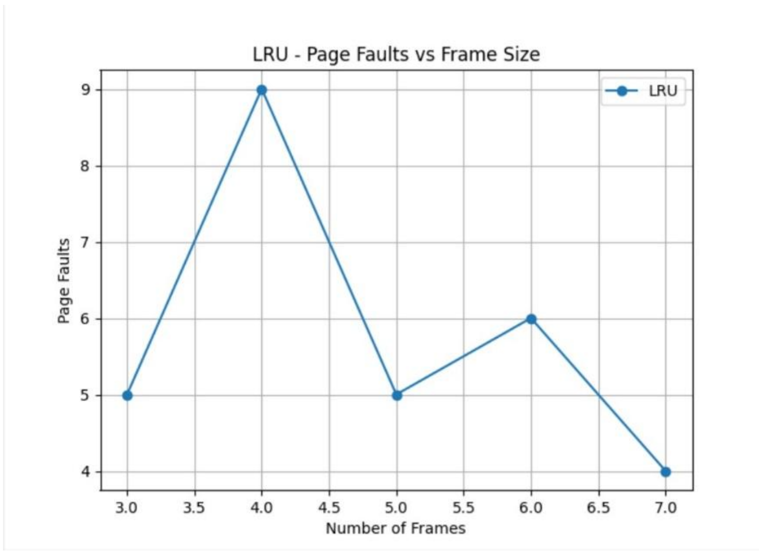

Figure 6.3 – LRU Algorithm

6.1 MEMORY VISUALIZATION

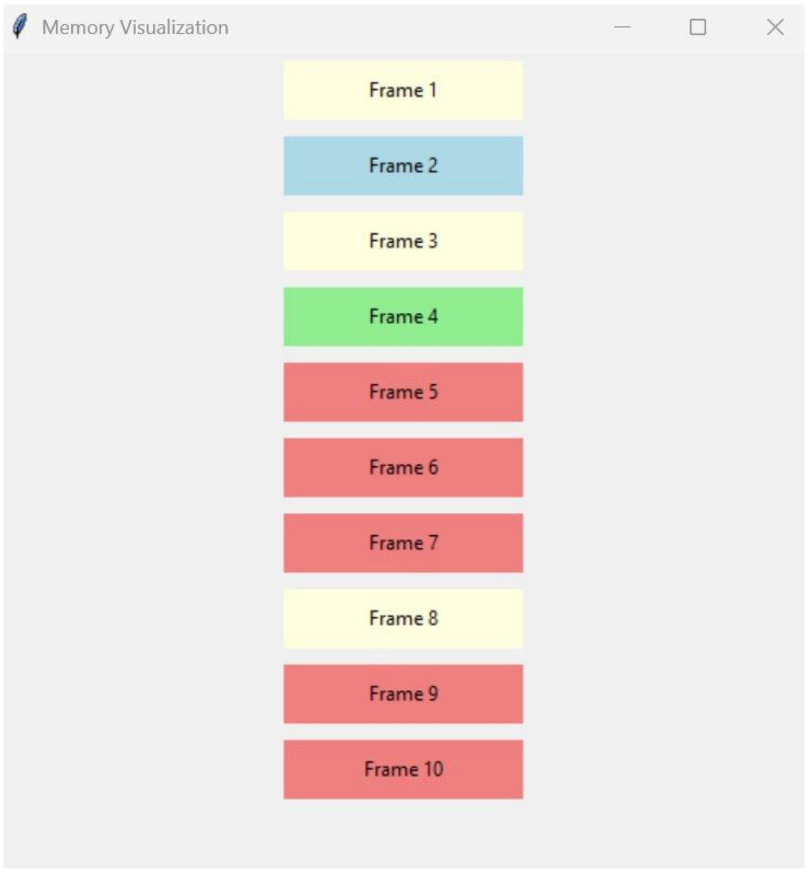

Figure 6.1.1 – Memory Visualization

CHAPTER – 7

CONCLUSION

The Virtual Memory Simulator effectively demonstrates the core concepts of memory management techniques such as paging, segmentation, demand paging, page replacement, and memory allocation. By providing a real-time, GUI-based simulation, it enables users—especially students and learners—to visualize how virtual memory works behind the scenes in an operating system. This interactive approach bridges the gap between theoretical learning and practical understanding, enhancing comprehension of complex OS concepts.

7.1 Advantages of this project include:

1. **GUI-Based Visualization**

   o Easy-to-use Tkinter interface makes the simulator intuitive and beginner-friendly.

2. **Educational Value**

   o Helps students understand and experiment with memory management techniques in real-time**.**

3. **Supports Multiple Algorithms**

   o Simulates various memory algorithms like FIFO, LRU, First-Fit, Best-Fit, etc., showing their differences and behavior.

4. **Better Concept Clarity**

   o Visual outputs such as tables and simulations improve understanding over traditional theoretical methods.

5. **Extensible Architecture**

   o The modular code allows easy additions of new algorithms or enhancements to the existing ones.

6. **Interactive Demonstrations**

   o Real-time feedback and dynamic updates make it perfect for academic presentations or live demos.

7. **Lightweight and Platform Independent**

   o Runs on any system with Python and Tkinter installed—no complex dependencies.

CHAPTER – 8
FUTURE WORKS

FUTURE WORKS:

Integration of Additional Algorithms

- Add support for advanced page replacement and memory allocation strategies like CLOCK, NRU (Not Recently Used), and Buddy System for deeper learning and comparison.

Performance Metrics Display

- Incorporate metrics such as page fault rate, memory utilization, and execution time to quantitatively compare algorithm efficiency.

User-Defined Inputs

- Allow users to input custom page reference strings, memory sizes, and allocation requests to make simulations more dynamic and personalized.

Animation and Step-by-Step Execution

- Introduce step-wise execution with animations to visualize how algorithms work internally at each step.

Save and Export Results

- Enable saving simulation data and exporting tables as PDFs or CSVs for reporting and documentation purposes.

Multi-Algorithm Comparison Graphs

- Implement graphical comparison between different algorithms to clearly show performance differences over various workloads.

Web-Based Version

- Develop a web-based simulator using frameworks like Flask or Django to make it accessible across platforms without the need for installation.

Memory Visualization with Real-Time Address Mapping

- Enhance the visual component by displaying how logical addresses are translated to physical addresses and how frames/segments are allocated.

Support for Multithreading or Multiprocessing Simulation

- Simulate concurrent processes accessing memory to demonstrate how OS handles memory management in multitasking environments.

CHAPTER – 9

REFERENCE

➢ REFERENCE:

- https://www.geeksforgeeks.org/virtual-memory-in-operating-system/
- https://www.studytonight.com/operating-system/paging-in-operating-system
- https://www.wiley.com/en-us/Operating+System+Concepts%2C+10th+Edition-p-9781119456339