

OBJECTIVE

To design and simulate a Virtual Memory Manager that efficiently handles memory allocation using different page replacement algorithms — minimizing page faults and improving system performance.

SYSTEM WORKFLOW

- User inputs page reference string & frame size. The VMM simulates page loading and replacement.
- Tracks page faults, hits, and efficiency for each algorithm.
- Displays comparative performance metrics using charts.

PERFORMANCE METRICS

- Page Faults: Number of times a page is not found in memory.
- Hit Ratio: Fraction of times required page is already in memory.
- Execution Time: Simulation time per algorithm.

CONCEPT OVERVIEW

Virtual Memory:
A memory management technique that gives an illusion of a large main memory by using disk space as an extension of RAM.

Page Replacement Algo Implemented:

1. FIFO (First In, First Out)
2. LRU (Least Recently Used)
3. Optimal Algorithm

Paging:
The process of dividing physical and logical memory into fixed-size blocks called pages and frames to efficiently manage memory.

TECHNOLOGY STACK

- Language: Python
- Libraries: Matplotlib
- Concepts: Paging, Replacement Policies

EXAMPLE OUTPUT

```
Enter page reference string (space separated): 7 0 1 2 0 3 0 4 2 3 0 3 2
Enter frame size: 3

Performance Summary:
FIFO: Faults = 10, Hits = 3, Hit Ratio = 0.23
LRU: Faults = 9, Hits = 4, Hit Ratio = 0.31
Optimal: Faults = 7, Hits = 6, Hit Ratio = 0.46
```

CONCLUSION

- The project successfully demonstrates how different page replacement algorithms affect system performance.
- Optimal Algorithm yields the best results, but LRU is most practical for real-world systems.