BVRIT HYDERABAD College of Engineering for Women

(Approved by AICTE | Affiliated to JNTUH | Accredited by NAAC with Grade 'A' &NBA for CSE, ECE, EEE, & IT)
Bachupally, Hyderabad-500090

Department of CSE(Artificial Intelligence and Machine Learning)

OPERATING SYSTEM
PROJECT NAME: Virtual Memory Management

TEAM MEMBERS:

23wh5a6603 P.SNEHA
22wh1a6607 Yashaswiny Sripada
22wh1a6617 D.Sai Lasya
22wh1a6627 Chandrika
22wh1a6637 E.Lahari
22wh1a6647 A.Akshaya
22wh1a6657 K.Sreeja

SUBMITTED BY: 23wh5a6603 P.SNEHA

PROBLEM STATEMENT:

The problem statement for virtual memory management is to efficiently manage the memory allocation and deallocation for a computer system, allowing it to use secondary memory as if it were part of the main memory. Further explore virtual memory management by comparing and contrasting paging and segmentation. Delve into the trade-offs associated with each approach and discuss how modern operating systems combine these techniques for optimal memory utilization.

ABSTRACT:

Introduction:

Virtual memory is a memory management technique that allows a computer to use more memory than it physically has available. This project aims to implement a virtual memory management system in C language that enables efficient memory allocation and deallocation, as well as page replacement algorithms to manage the virtual memory.

Implementation Details:

The virtual memory management system will be implemented using a combination of hardware and software components. The hardware component will include a Memory Management Unit (MMU) that maps virtual addresses to physical addresses. The software component will include a page table that stores the mapping between virtual and physical addresses, as well as a page replacement algorithm to manage the virtual memory.

Performance Analysis:

The performance of the virtual memory management system depends on the page replacement algorithm used. The project will implement and compare the performance of the following page replacement algorithms:

First-In-First-Out (FIFO)

Least-Recently-Used (LRU)

Optimal

Conclusion:

The virtual memory management system implemented in this project will enable efficient memory allocation and deallocation, as well as page replacement algorithms to manage the virtual memory. The project will provide a solid foundation for understanding virtual memory management and its implementation in C language.

PROGRAM:

```
#include<stdio.h>
#include <stdlib.h>
#define PAGE_SIZE 4096
#define NUM_PAGES 10
typedef struct {
int page_id;
Int data[PAGE_SIZE];
} Page;
typedef struct {
 Page *pages[NUM_PAGES];
int page_table[NUM_PAGES];
} VirtualMemory;
VirtualMemory* initialize_virtual_memory() {
VirtualMemory*vm = (VirtualMemory*)malloc(sizeof(VirtualMemory)); for
(int i = 0; i < NUM_PAGES; i++)
{
vm->pages[i] = (Page*)malloc(sizeof(Page));
vm->pages[i]->page_id = i;
} for (int i = 0; i < NUM_PAGES; i++)
{
vm->page_table[i] = -1; // -1 indicates the page is not in physical memory }
return vm;
```

```
}
void load_page_into_memory(VirtualMemory *vm, int page_id)
{
printf("Loading Page %d into physical memory.\n", page_id);
vm->page_table[page_id] = page_id;
}
void access_memory_location(VirtualMemory *vm, int virtual_address) {
int page_id = virtual_address / PAGE_SIZE;
if (vm->page_table[page_id] == -1) {
load_page_into_memory(vm, page_id);
}
printf("Accessing memory location %d from Page %d in physical memory.\n", virtual_address, page_id);
}
void free_virtual_memory(VirtualMemory *vm)
{
for (int i = 0; i < NUM_PAGES; i++)
{
free(vm->pages[i]);
}
free(vm);
}
int main()
{
 VirtualMemory *vm = initialize_virtual_memory();
access_memory_location(vm, 2048);
access_memory_location(vm, 8192);
access_memory_location(vm, 5120);
free_virtual_memory(vm); return 0;
}
```

OUTPUT :

```
Loading Page 0 into physical memory.
Accessing memory location 2048 from Page 0 in physical memory.
Loading Page 2 into physical memory.
Accessing memory location 8192 from Page 2 in physical memory.
Loading Page 1 into physical memory.
Accessing memory location 5120 from Page 1 in physical memory.
```