Name:- Parmar Kalpesh Pravinbhai
Enrollment:- 9240058194
class:- MCA 1-C
Subject:- O.S.

1) Virtual Memory Architecture and Explantion.

- Virtual memory is a memory management technique that gives an application the impression that it has contiguous available memory, even if it is physically fragmented.

- It enables systems to use both physical memory (RAM) and disk storage to run application that require more memory that what is physically available. Virtual memory relies on hardware and software working together, with the following key components:

Virtual Address Space:

Each process is given its own virtual address space, which is a range of addresses that it can use. This space is independent of physical memory, allowing the process to operate without knowledge of actual memory allocation.

Paging:

Physical memory is divided into small, fixed-sized blocks called "pages". Similarly, virtual memory is divided into pages that map to these physical pages. If a required page is not in physical memory, the operating system swaps it from disk storage, a process called "paging" or "swapping".

Page Table:

Each process has a page table, a data structure that maps virtual addresses to physical addresses. The Page table allows the operating system to translate virtual addresses into physical ones.

Benefits of Virtual Memory:

- Allows programs to use more memory than physically available.
- Provides isolation between processes, enhancing security.
- Simplifies memory management by enabling dynamic allocation.
- Optimizes memory utilization by sharing pages between processes

2] What is Thread?

- A Thread is the smallest unit of execution within a process. Threads enable concurrent execution within a single process sharing resources like memory and file handles but having their own program counter, stack and register. Threads are particularly useful in scenarios where tasks can run simultaneously, such as handling multiple requests on a web server.

Types of Threads:

User Threads:

These are managed by user-level libraries, with no involvement from the operating System. They are faster to create and manage, but if one thread is blocked, all threads in the same process may stop.

Kernel Threads:

Managed directly by the operating system kernel. kernel threads can take advantage of multiple processors and can perform system calls. However they require more resources and are slower to create compared to user threads