DEPARTMENT OF AIML
23CST202- OPERATING SYSTEMS
II YEAR IV SEM AIML-B
UNIT 3-MEMORY MANAGEMENT
TOPIC –PAGING,SEGMENTATION

PAGING

Paging is a memory management scheme that eliminates the need for a contiguous allocation of physical memory. The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as paging. The basic purpose of paging is to separate each procedure into pages.

The mapping between logical pages and physical page frames is maintained by the page table, which is used by the memory management unit to translate logical addresses into physical addresses. The page table maps each logical page number to a physical page frame number. By using a Page Table, the operating system keeps track of the mapping between logical addresses (used by programs) and physical addresses (actual locations in memory).

Why Paging is used for memory Management?
Paging is a memory management technique that addresses common challenges in allocating and managing memory efficiently. Here we can understand why paging is needed as a Memory Management technique:

- **Memory isn't always available in a single block:** Programs often need more memory than what is available in a single continuous block. Paging breaks memory into smaller, fixed-size pieces, making it easier to allocate scattered free spaces.
- **Processes size can increase or decrease:** programs don't need to occupy continuous memory, so they can grow dynamically without the need to be moved.

Terminologies Associated with Memory Control

- **Logical Address or Virtual Address:** The Logical Address, also known as the Virtual Address, is the address generated by the CPU when a program accesses memory.
- **Logical Address Space or Virtual Address Space:** The Logical Address Space, also known as the Virtual Address Space, refers to the set of all possible logical addresses that a process can generate during its execution. It is a conceptual range of memory addresses used by a program and is independent of the actual physical memory (RAM).
- **Physical Address:** A Physical Address is the actual location in the computer's physical memory (RAM) where data or instructions are stored. It is used by the memory hardware to access specific data in the system's memory.
- **Physical Address Space:** The Physical Address Space refers to the total range of addresses available in a computer's physical memory (RAM). It represents

the actual memory locations that can be accessed by the system hardware to store or retrieve data.

Important Features of Paging in PC Reminiscence Management

- **Logical to bodily address mapping:** In paging, the logical address area of a technique is divided into constant-sized pages, and each web page is mapped to a corresponding physical body within the main reminiscence. This permits the working gadget to manipulate the memory in an extra flexible way, as it is able to allocate and deallocate frames as needed.
- **Fixed web page and frame length:** Paging makes use of a set web page length, which is usually identical to the size of a frame within the most important memory. This facilitates simplifying the reminiscence control technique and improves device performance.
- **Page desk entries:** Each page within the logical address area of a method is represented through a page table entry (PTE), which contains facts approximately the corresponding bodily body in the predominant memory. This consists of the frame range, in addition to other manipulate bits which can be used by the running machine to manage the reminiscence.
- **A number of page desk entries:** The range of page desk entries in a manner's page desk is identical to the wide variety of pages inside the logical deal with the area of the technique.
- **Page table stored in important memory:** The web page desk for each system is typically saved in important reminiscence, to allow for green get right of entry to and change by the operating device. However, this could additionally introduce overhead, because the web page table must be updated on every occasion a system is swapped in or out of the main memory.

How Paging Works?

Paging is a method used by operating systems to manage memory efficiently. In paging, the physical memory is divided into fixed-size blocks called page frames, which are the same size as the pages used by the process. The process's logical address space is also divided into fixed-size blocks called pages, which are the same size as the page frames.

When a process requests memory, the operating system allocates one or more page frames to the process and maps the process's **logical pages** to the physical page frames. When a program runs, its pages are loaded into any available frames in the physical memory.

This approach prevents fragmentation issues by keeping memory allocation uniform. Each program has a page table, which the operating system uses to keep track of where each page is stored in physical memory. When a program accesses data, the system uses this table to convert the program's address into a physical memory address.

Paging allows for better memory use and makes it easier to manage. It also supports virtual memory, letting parts of programs be stored on disk and loaded into memory only when needed. This way, even large programs can run without fitting entirely into main memory.

- If Logical Address = 31 bit, then Logical Address Space = 231 words = 2 G words (1 G = 230)

- If Logical Address Space = 128 M words = 27 * 220 words, then Logical Address = log2 227 = 27 bits
- If Physical Address = 22 bit, then Physical Address Space = 222 words = 4 M words (1 M = 220)
- If Physical Address Space = 16 M words = 24 * 220 words, then Physical Address = log2 224 = 24 bits

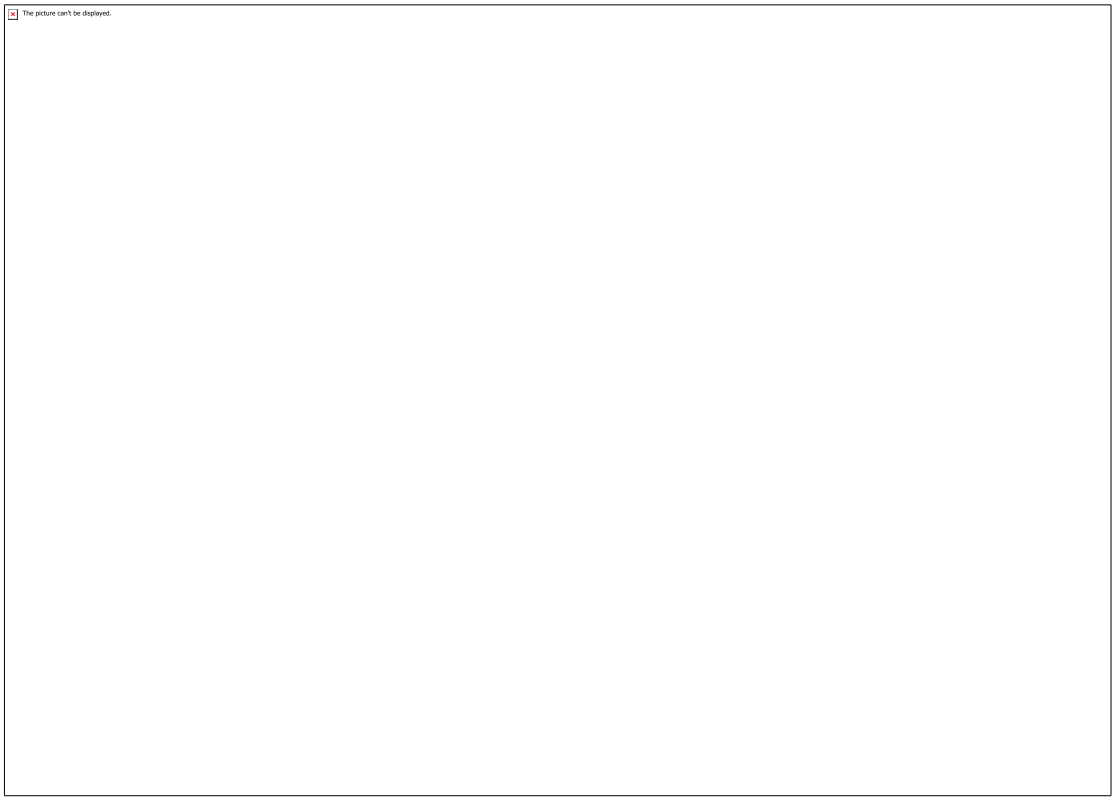
The mapping from virtual to physical address is done by the Memory Management Unit (MMU) which is a hardware device and this mapping is known as the paging technique.

- The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.
- The Logical Address Space is also split into fixed-size blocks, called **pages**.
- Page Size = Frame Size

Example

- Physical Address = 12 bits, then Physical Address Space = 4 K words
- Logical Address = 13 bits, then Logical Address Space = 8 K words
- Page size = frame size = 1 K words (assumption)

Number of frames = Physical Address Space / Frame Size = 4K / 1K = 4 = 22
Number of Pages = Logical Address Space / Page Size = 8K / 1K = 23



The address generated by the CPU is divided into

1. **Page number(p):** Number of bits required to represent the pages in Logical Address Space or Page number
2. **Page offset(d):** Number of bits required to represent a particular word in a page or page size of Logical Address Space or word number of a page or page offset.

A **Physical Address** is divided into two main parts:

1. **Frame Number(f):** Number of bits required to represent the frame of Physical Address Space or Frame number frame
2. **Frame Offset(d):** Number of bits required to represent a particular word in a frame or frame size of Physical Address Space or word number of a frame or frame offset.

So, a physical address in this scheme may be represented as follows:

Physical Address = (Frame Number << Number of Bits in Frame Offset) + Frame Offset
where "<<" represents a bitwise left shift operation.

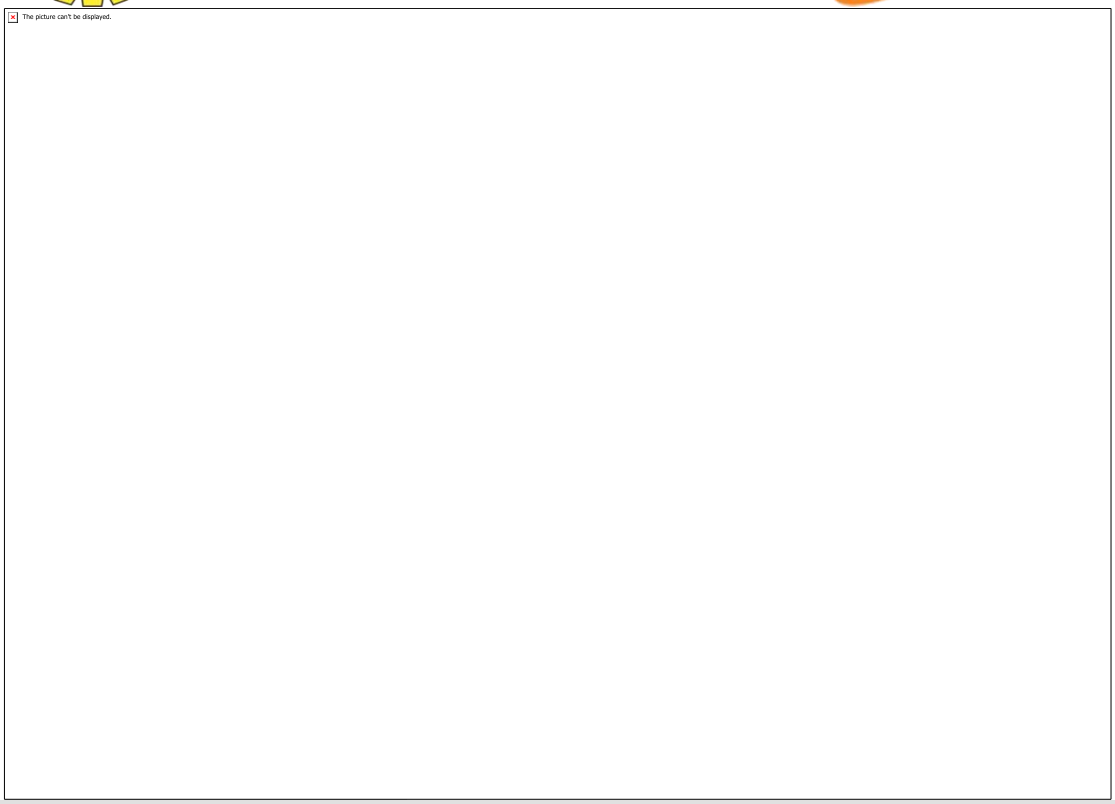
- The TLB is associative, high-speed memory.
- Each entry in TLB consists of two parts: a tag and a value.
- When this memory is used, then an item is compared with all tags simultaneously. If the item is found, then the corresponding value is returned.

Paging is a memory management technique used in operating systems to manage memory and allocate memory to processes. In paging, memory is divided into fixed-size blocks called pages, and processes are allocated memory in terms of these pages. Each page is of the same size, and the size is typically a power of 2, such as 4KB or 8 KB.

Hardware implementation of Paging

The hardware implementation of the page table can be done by using dedicated registers. But the usage of the register for the page table is satisfactory only if the page table is small. If the page table contains a large number of entries then we can use TLB(translation Look-aside buffer), a special, small, fast look-up hardware cache.

- The TLB is associative, high-speed memory.
- Each entry in TLB consists of two parts: a tag and a value.
- When this memory is used, then an item is compared with all tags simultaneously. If the item is found, then the corresponding value is returned.



Main memory access time = m
If page table are kept in main memory,
Effective access time = m(for page table) + m(for particular page in page table)

Advantages of Paging

- **Eliminates External Fragmentation:** Paging divides memory into fixed-size blocks (pages and frames), so processes can be loaded wherever there is free space in memory. This prevents wasted space due to fragmentation.
- **Efficient Memory Utilization:** Since pages can be placed in non-contiguous memory locations, even small free spaces can be utilized, leading to better memory allocation.
- **Supports Virtual Memory:** Paging enables the implementation of virtual memory, allowing processes to use more memory than physically available by swapping pages between RAM and secondary storage.
- **Ease of Swapping:** Individual pages can be moved between physical memory and disk (swap space) without affecting the entire process, making swapping faster and more efficient.
- **Improved Security and Isolation:** Each process works within its own set of pages, preventing one process from accessing another's memory space.

Disadvantages of Paging

- **Internal Fragmentation:** If the size of a process is not a perfect multiple of the page size, the unused space in the last page results in internal fragmentation.
- **Increased Overhead:** Maintaining the Page Table requires additional memory and processing. For large processes, the page table can grow significantly, consuming valuable memory resources.

- **Page Table Lookup Time:** Accessing memory requires translating logical addresses to physical addresses using the page table. This additional step increases memory access time, although Translation Lookaside Buffers (TLBs) can help reduce the impact.
- **I/O Overhead During Page Faults:** When a required page is not in physical memory (page fault), it needs to be fetched from secondary storage, causing delays and increased I/O operations.
- **Complexity in Implementation:** Paging requires sophisticated hardware and software support, including the Memory Management Unit (MMU) and algorithms for page replacement, which add complexity to the system.

What is Memory Management Unit (MMU)?

A memory management unit (MMU) is a technique used to convert logical address to physical address. Logical address is the address generated by CPU for each page and physical address is the real address of the frame where page is going to be stored.

Whenever a page has to be accessed by CPU using the logical address, it requires physical address for accessing the page. Logical address comprises of two parts: Page Number and Offset.

SEGMENTATION

A process is divided into Segments. The chunks that a program is divided into which are not necessarily all of the exact sizes are called segments. Segmentation gives the user's view of the process which paging does not provide. Here the user's view is mapped to physical memory.

Types of Segmentation in Operating Systems

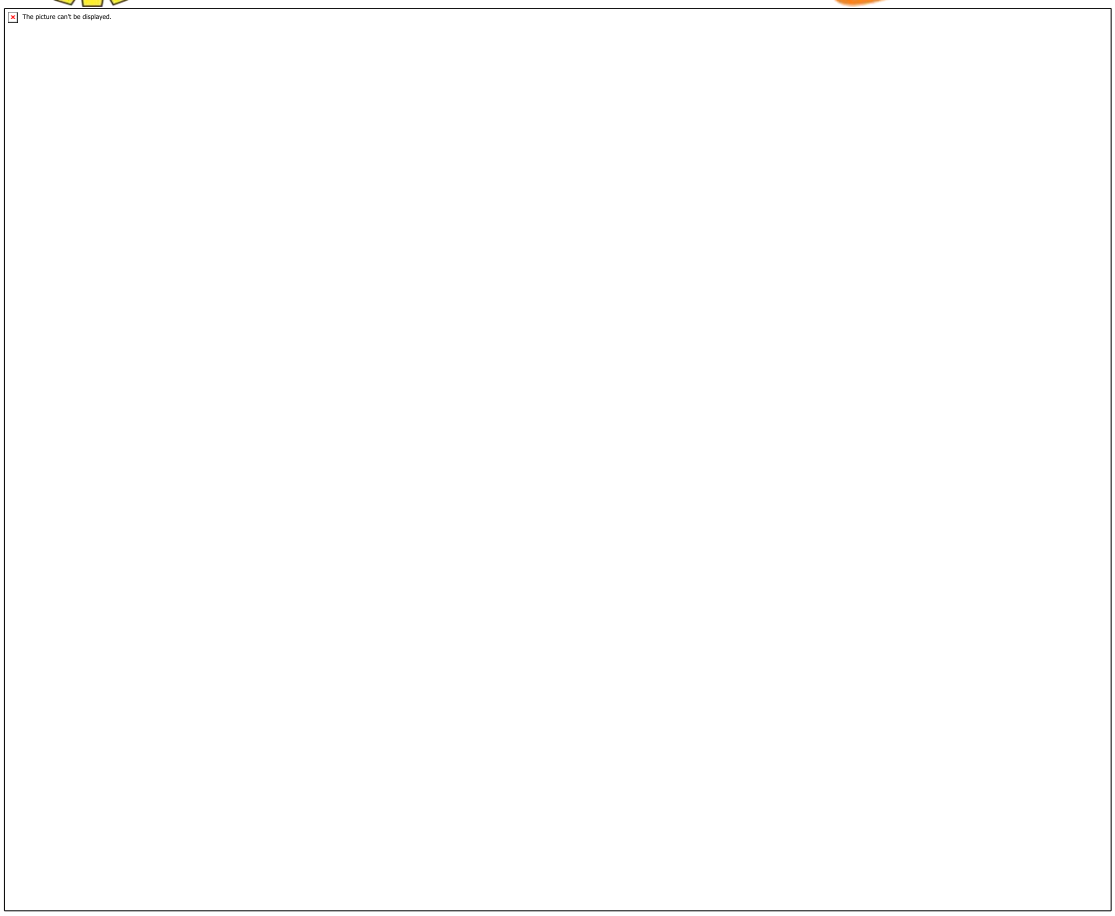
- **Virtual Memory Segmentation:** Each process is divided into a number of segments, but the segmentation is not done all at once. This segmentation may or may not take place at the run time of the program.
- **Simple Segmentation:** Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

There is no simple relationship between logical addresses and physical addresses in segmentation. A table stores the information about all such segments and is called Segment Table.

What is Segment Table?

It maps a two-dimensional Logical address into a one-dimensional Physical address. It's each table entry has:

- **Base Address:** It contains the starting physical address where the segments reside in memory.
- **Segment Limit:** Also known as segment offset. It specifies the length of the segment.



Segmentation

Translation of Two-dimensional Logical Address to Dimensional Physical Address.

Translation

The address generated by the CPU is divided into:

- **Segment number (s):** Number of bits required to represent the segment.

- **Segment offset (d):** Number of bits required to represent the position of data within a segment.

Advantages of Segmentation in Operating System

- **Reduced Internal Fragmentation** : Segmentation can reduce internal fragmentation compared to fixed-size paging, as segments can be sized according to the actual needs of a process. However, internal fragmentation can still occur if a segment is allocated more space than it is actually used.
- Segment Table consumes less space in comparison to Page table in paging.
- As a complete module is loaded all at once, segmentation improves CPU utilization.
- The user's perception of physical memory is quite similar to segmentation. Users can divide user programs into modules via segmentation. These modules are nothing more than separate processes' codes.
- The user specifies the segment size, whereas, in paging, the hardware determines the page size.
- Segmentation is a method that can be used to segregate data from security operations.
- **Flexibility:** Segmentation provides a higher degree of flexibility than paging. Segments can be of variable size, and processes can be designed to have multiple segments, allowing for more fine-grained memory allocation.
- **Sharing:** Segmentation allows for sharing of memory segments between processes. This can be useful for inter-process communication or for sharing code libraries.
- **Protection:** Segmentation provides a level of protection between segments, preventing one process from accessing or modifying another process's memory segment. This can help increase the security and stability of the system.

Disadvantages of Segmentation in Operating System

- **External Fragmentation** : As processes are loaded and removed from memory, the free memory space is broken into little pieces, causing external fragmentation. This is a notable difference from paging, where external fragmentation is significantly lesser.
- Overhead is associated with keeping a segment table for each activity.
- Due to the need for two memory accesses, one for the segment table and the other for main memory, access time to retrieve the instruction increases.
- **Fragmentation:** As mentioned, segmentation can lead to external fragmentation as memory becomes divided into smaller segments. This can lead to wasted memory and decreased performance.
- **Overhead:** Using a segment table can increase overhead and reduce performance. Each segment table entry requires additional memory, and accessing the table to retrieve memory locations can increase the time needed for memory operations.
- **Complexity:** Segmentation can be more complex to implement and manage than paging. In particular, managing multiple segments per process can be challenging, and the potential for segmentation faults can increase as a result.