**Unit 4**

*1) Page Replacement Algos! FIFO, LRU, Optimal*

*2) Explain Buddy System with the help of neat diagram and example.  
  
Ans*. Buddy System is a memory allocation technique used in computer OS to allocate and manage memory efficiently. This technique by dividing the memory into fixed-size blocks, and whenever a process requests memory, the system finds the smallest available block that can accommodate the requested memory size.

It splits memory blocks, called “buddies,” to minimize fragmentation and ensure efficient allocation. When a process is deallocated, its buddy can be merged back into a larger block, reducing wasted space.

Below are the steps involved in the Buddy System Memory Allocation Technique:

* The first step includes the division of memory into fixed-sized blocks that have a power of 2 in size (such as 2, 4, 8, 16, 32, 64, 128, etc. ).
* Each block is labeled with its size and unique identification.
* Initially, all the memory blocks are free and are linked together in a binary tree structure, with each node representing a block and the tree’s leaves representing the smallest available blocks.
* When a process requests memory, the system finds the smallest available block that can accommodate the requested size. If the block is larger than the requested size, the system splits the block into two equal-sized “buddy” blocks.
* The system marks one of the buddy blocks as allocated and adds it to the process’s memory allocation table, while the other buddy block is returned to the free memory pool and linked back into the binary tree structure.
* When a process releases memory, the system marks the corresponding block as free and looks for its buddy block. If the buddy block is also free, the system merges the two blocks into a larger block and links it back into the binary tree structure.

The Buddy System technique has several advantages, including efficient use of memory, [reduced fragmentation](https://www.geeksforgeeks.org/what-is-fragmentation-in-operating-system/), and fast allocation and deallocation of memory blocks. However, it also has some drawbacks, such as internal fragmentation, where a block may be larger than what the process requires, leading to a waste of memory. Overall, the Buddy System is a useful memory allocation technique in operating systems, particularly for embedded systems with limited memory.

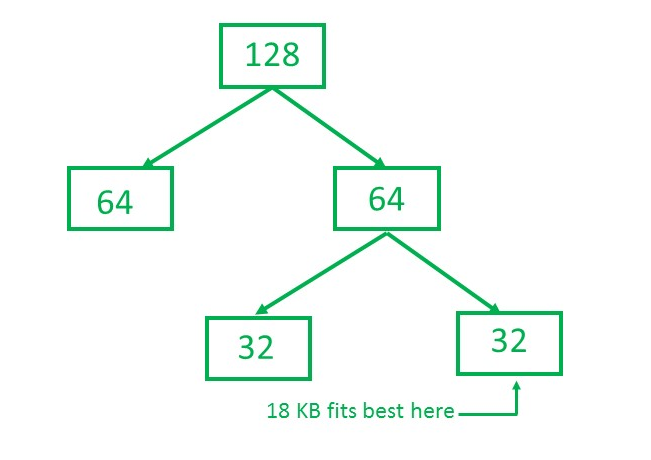
* **If 2 U-1 <S<=2 U :**Allocate the whole block
* **Else:**Recursively divide the block equally and test the condition at each time, when it satisfies, allocate the block and get out of the loop.

The system also keeps a record of all the unallocated blocks and can merge these different-sized blocks to make one big chunk.

**Example of Buddy system**

Consider a system having buddy system with physical address space 128 KB.Calculate the size of partition for 18 KB process.

**Solution:**



So, size of partition for 18 KB process = 32 KB. It divides by 2, till possible to get minimum block to fit 18 KB.

Application:

*The Buddy System is normally used in running structures, mainly in environments wherein memory is allotted and deallocated frequently, including* [*multitasking*](https://www.geeksforgeeks.org/difference-between-multitasking-multithreading-and-multiprocessing/) *structures or structures with varying memory needs. It is frequently hired in reminiscence allocators, which includes the pal allocator in the Linux kernel.*

3) *What is segmentation? How address Translation is performed in segmentation system?*

Ans.

**Segmentation in Operating Systems**

Segmentation is a memory management technique where a program is divided into logically related variable-sized blocks called segments. Each segment represents a distinct logical unit, such as a code section, data section, or stack. Unlike paging, which divides memory into fixed-size blocks, segmentation is based on the logical structure of a program.

**Key Characteristics of Segmentation**

1. Logical Division: Memory is divided into segments based on functionality, like code, data, heap, and stack.
2. Variable Size: Segments can vary in size depending on the size of the logical unit.
3. Address Space: Each segment has a unique segment number and its own offset.
4. Protection: Different protection levels can be assigned to each segment, enabling efficient access control.

**Address Translation in a Segmentation System**

Segmentation uses a segment table for address translation. This table contains information about each segment, such as its base address and limit.

**Steps in Address Translation**

1. Logical Address: A logical address in a segmentation system is represented as a pair: (Segment Number, Offset). Example: (Segment 2, Offset 120).
2. Segment Table Lookup:
   * Segment Number: Used to index into the segment table.
   * The segment table contains:
     + Base Address: Starting physical address of the segment in memory.
     + Limit: The size of the segment.
3. Validation:
   * Check if the Offset is within the Limit for the segment. If:
     + Offset < Limit: Proceed.
     + Offset >= Limit: Trigger a segmentation fault (invalid access).
4. Physical Address Calculation:
   * The physical address is computed by adding the Base Address of the segment to the Offset: Physical Address = Base Address + Offset.

**Example of Address Translation**

Given:

* Logical Address: (Segment 2, Offset 150)
* Segment Table: Segment | Base Address | Limit 0 | 1000 | 400 1 | 2000 | 600 2 | 3000 | 500

Steps:

1. Segment Table Lookup:
   * Segment 2:
     + Base Address = 3000
     + Limit = 500
2. Validation:
   * Offset = 150
   * Since 150 < 500, the access is valid.
3. Physical Address Calculation: Physical Address = Base Address + Offset = 3000 + 150 = 3150

Result: The physical address corresponding to the logical address (Segment 2, Offset 150) is 3150.

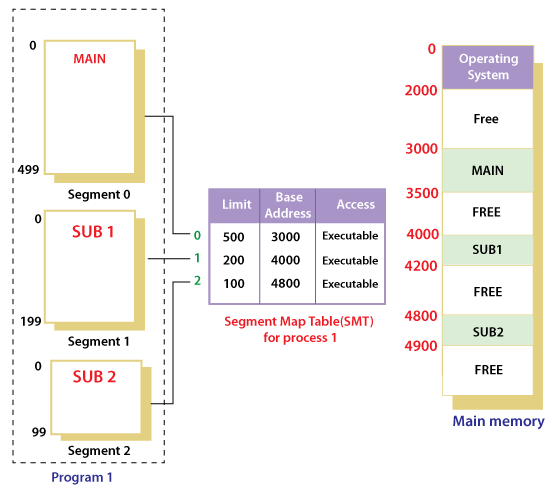
**Advantages of Segmentation**

1. Provides a logical organization of memory compared to paging.
2. Supports protection and sharing at the segment level.
3. Allows dynamic growth of segments like stacks or heaps.
4. No internal fragmentation
5. Average Segment Size is larger than the actual page size.
6. Less overhead
7. It is easier to relocate segments than entire address space.
8. The segment table is of lesser size as compared to the page table in paging.

**Disadvantages of Segmentation**

1. Can lead to external fragmentation because segments are of variable size.
2. Segment Table Overhead: Requires extra memory and processing to maintain the segment table.
3. it is difficult to allocate contiguous memory to variable sized partition.
4. Costly memory management algorithms.

Segmentation is often combined with paging in modern systems for efficient memory management.

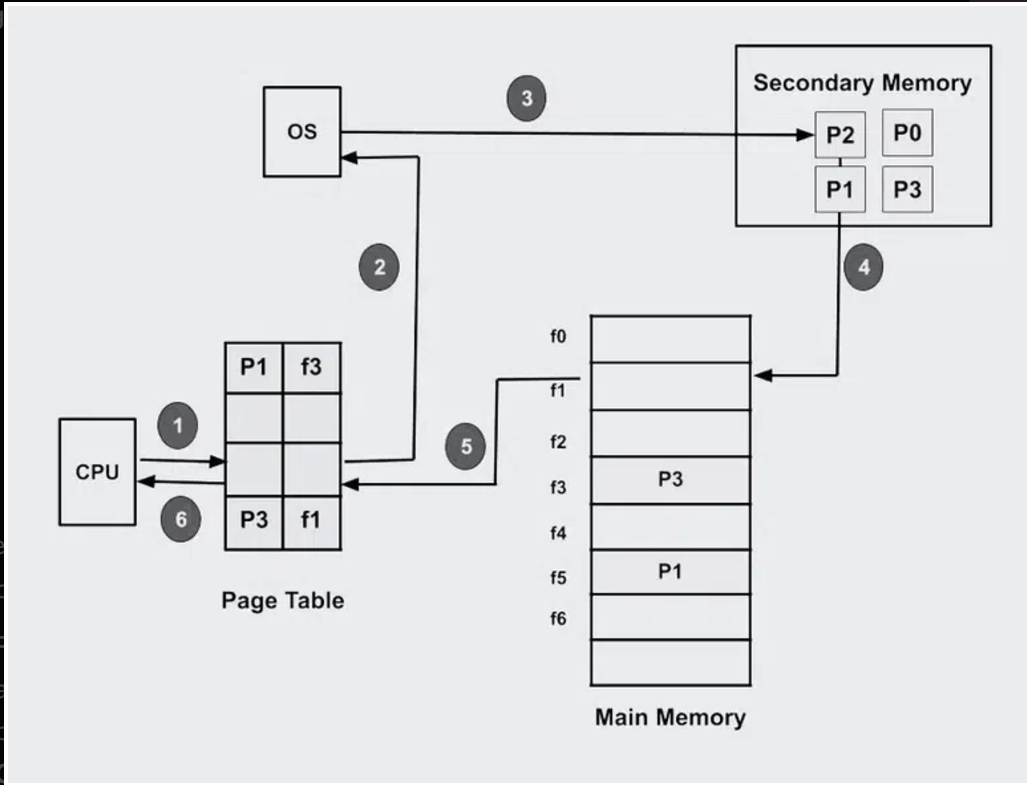


4) *Explain Demand paging with the help of neat diagram.*

Ans. <https://www.youtube.com/watch?v=ROVIH4m0LyU>

Demand paging is a technique used in virtual memory systems where pages enter main memory only when requested or needed by the CPU. In demand paging, the operating system loads only the necessary pages of a program into memory at runtime, instead of loading the entire program into memory at the start. A page fault occurred when the program needed to access a page that is not currently in memory.

The operating system then loads the required pages from the disk into memory and updates the page tables accordingly. This process is transparent to the running program and it continues to run as if the page had always been in memory.



The [operating system](https://www.geeksforgeeks.org/what-is-an-operating-system/)‘s demand paging mechanism follows a few steps in its operation.

* **Program Execution:** Upon launching a program, the operating system allocates a certain amount of memory to the program and establishes a process for it.
* **Creating Page Tables:**To keep track of which program pages are currently in memory and which are on disk, the operating system makes [page tables](https://www.geeksforgeeks.org/page-table-entries-in-page-table/) for each process.
* **Handling Page Fault:**When a program tries to access a page that isn’t in memory at the moment, a page fault happens. In order to determine whether the necessary page is on disk, the operating system pauses the application and consults the page tables.
* **Page Fetch:** The operating system loads the necessary page into memory by retrieving it from the disk if it is there.
* The page’s new location in memory is then reflected in the page table.
* **Resuming The Program:**The operating system picks up where it left off when the necessary pages are loaded into memory.
* **Page Replacement:** If there is not enough free memory to hold all the pages a program needs, the operating system may need to replace one or more pages currently in memory with pages currently in memory. on the disk. The page replacement algorithm used by the operating system determines which pages are selected for replacement.
* **Page Cleanup:** When a process terminates, the operating system frees the memory allocated to the process and cleans up the corresponding entries in the page tables.

**Advantages of Demand Paging**

1. **Efficient Memory Utilization**: Only required pages are loaded into memory.
2. **Reduced Load Time**: Programs load faster as not all pages are fetched initially.
3. **Support for Large Programs**: Allows execution of programs that require more memory than physically available.

**Disadvantages of Demand Paging**

1. **Page Fault Overhead**: Frequent page faults can slow down execution.
2. **Latency**: Accessing a page from secondary storage introduces delay.
3. **Thrashing**: Excessive page faults due to limited memory can degrade system performance.

5) *What are the distinctions among logical, relative and physical addresses?*

*Ans.*

**1. Logical Address**

* **Definition**: The address generated by the CPU during program execution. It is a virtual address in the context of the program's view of memory.
* **Context**: Also called a virtual address.
* **Translation**: Mapped to a physical address by the memory management unit (MMU) using segmentation or paging.
* **Scope**: Logical addresses are process-specific, meaning each process has its own logical address space.
* **Usage**: Used by the user/programmer to access memory.

**Example:**

If a process accesses memory at logical address 200, the CPU works with this address, but it needs to be translated to a physical address for actual memory access.

**2. Relative Address**

* **Definition**: An address specified relative to a specific starting point, such as the beginning of a program, segment, or block.
* **Context**: Often used in programs as offsets from a base address.
* **Translation**: Converted into a logical address by adding the relative address to the base address of the segment or block.
* **Scope**: Process-specific, but depends on the base address provided by the operating system or compiler.
* **Usage**: Commonly used for relocation or in position-independent code.

**Example:**

If the base address of a segment is 5000 and a relative address is 120, the logical address becomes 5120 (5000 + 120).

**3. Physical Address**

* **Definition**: The actual address in the main memory hardware where data is stored or retrieved.
* **Context**: The memory address seen by the hardware after translation from a logical address.
* **Translation**: Obtained by the memory management unit (MMU) using the page table or segment table to map logical or relative addresses.
* **Scope**: System-wide; independent of processes.
* **Usage**: Used by the hardware (RAM) to access memory locations.

**Example:**

If the MMU maps logical address 5120 to frame 2 at physical address 10240, this is the address used by the memory hardware.

**Scenario:**

* A program is allocated a **segment** with a base address of 4000 in physical memory.
* Inside the program, it references an **instruction** or **data** at an **offset** of 120 bytes from the start of the segment.
* The logical address generated by the CPU for this reference is (Segment 1, Offset 120).

**1. Logical Address**

* **Definition**: The address generated by the CPU that the program "sees" when it accesses memory.
* **Example**: (Segment 1, Offset 120)
  + **Segment 1** refers to a segment assigned to the program.
  + **Offset 120** refers to the distance from the beginning of the segment.
* **Explanation**: The logical address (1, 120) needs to be mapped to an actual physical address by adding the segment's base address (from the segment table).

**2. Relative Address**

* **Definition**: An address specified as an offset relative to a program's base or starting point.
* **Example**: 120 (relative to the start of the segment or program).
  + This is the offset provided by the program for accessing memory within its segment.
* **Explanation**: The relative address is independent of the segment's location in physical memory. It becomes meaningful only when combined with the program's base address, translating into a logical or physical address.

**3. Physical Address**

* **Definition**: The actual address in physical memory where data or instructions reside.
* **Example**: 4000 (Base) + 120 (Offset) = 4120.
  + The physical address 4120 represents the location in the computer's main memory where the instruction or data is stored.
* **Explanation**: After the operating system or memory management unit (MMU) maps the logical address (1, 120) using the base address of the segment (4000), it resolves into the physical address 4120.

*6) Explain Belady’s anomaly with suitable example.*

Ans. **Belady's Anomaly**

**Belady’s Anomaly** is a counterintuitive phenomenon in page replacement algorithms, where increasing the number of page frames in memory can lead to **more page faults** rather than fewer. It is primarily observed in the **First-In-First-Out (FIFO)** page replacement algorithm.

**Explanation of Belady’s Anomaly**

* Normally, increasing the number of page frames should decrease page faults because more pages can be stored in memory.
* However, with Belady's anomaly, the replacement order of FIFO can cause more page faults when additional frames are added, as the algorithm simply removes the oldest page without considering future use.

**Example to Illustrate Belady's Anomaly**

**Page Reference String**: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

**Case 1: 3 Page Frames**

We simulate the FIFO algorithm with 3 frames:

1. Initially empty: Add 1, 2, 3 → **Page faults: 3**.
2. Add 4 (replace 1) → **Page faults: 4**.
3. Access 1 (replace 2) → **Page faults: 5**.
4. Access 2 (replace 3) → **Page faults: 6**.
5. Add 5 (replace 4) → **Page faults: 7**.
6. Access 1, 2, 3, 4, 5: All cause page faults because the memory does not have these pages anymore → **Page faults: 12**.

**Total Page Faults (3 frames)**: **9**

**Case 2: 4 Page Frames**

We simulate the FIFO algorithm with 4 frames:

1. Initially empty: Add 1, 2, 3, 4 → **Page faults: 4**.
2. Access 1, 2: Present in memory → **Page faults: 4**.
3. Add 5 (replace 3) → **Page faults: 5**.
4. Access 1, 2: Present in memory → **Page faults: 5**.
5. Add 3 (replace 4) → **Page faults: 6**.
6. Add 4 (replace 5) → **Page faults: 7**.
7. Access 5 (replace 3) → **Page faults: 8**.

**Total Page Faults (4 frames)**: **10**

**Observing the Anomaly**

* With **3 frames**, the total page faults are **9**.
* With **4 frames**, the total page faults increase to **10**.

This is **Belady's Anomaly**: Adding more frames (4 instead of 3) caused **more page faults**, which is contrary to the expectation.

**Why Does Belady’s Anomaly Occur?**

1. **FIFO Nature**: The FIFO algorithm replaces the oldest page without considering whether it will be used in the future.
2. **Replacement Decisions**: With more frames, the order of page replacement changes, leading to more faults in some cases.

**Avoiding Belady’s Anomaly**

Algorithms like **Least Recently Used (LRU)** and **Optimal Page Replacement** do not suffer from Belady's anomaly because they consider future or recent usage of pages while making replacement decisions.

7) Consider six memory partitions of size 100 KB, 300 KB, 50 KB, 200 KB,150 KB and 200 KB. These partitions need to be allocated to processes of sizes 200 KB, 100 KB, 50 KB in that order. Perform the allocation of processes using dynamic partitioning algorithms given be low and comment on internal and external fragmentation [12] i) ii) iii) First Fit Algorithm Best Fit Algorithm Worst Fit Algorithm  
  
