Assignment -10 Operating System
Name: Veeransh Shah
Reg ID: 221070063

Aim:
mplement Paging Technique of memory management

Theory:

Paging is a fundamental memory management technique used in computer systems to manage how data is stored and retrieved in physical memory. This technique involves breaking the memory into fixed-sized blocks, known as "pages," in the virtual memory and corresponding "frames" in the physical memory. The primary advantages of paging include:

- Elimination of External Fragmentation: Since pages and frames are of the same size, the system can efficiently utilize memory without leaving unallocated space that's too small to be useful.
- Simpler Memory Allocation: Allocating memory in fixed-sized chunks simplifies the management of memory, making it easier to allocate and deallocate memory dynamically.
- Virtual Memory Implementation: Paging supports the use of virtual memory, where pages can be swapped in and out of physical memory to disk, allowing a system to use more memory than what is physically available.

The core component of paging is the "page table," which maps virtual addresses to physical addresses. Each entry in the page table corresponds to a page and contains the frame number where that page resides in the physical memory. If a page is not currently in memory (i.e., a page fault occurs), the system must retrieve it from disk, which involves more complex management techniques such as page replacement algorithms.

Conclusion of the Simulation Code

The provided Python script simulates a very simplified version of paging:

- Memory Management Unit (MMU): The simulation includes a class that functions as a basic MMU, managing the mapping of virtual pages to physical frames through a page table.
- Page Table: Implemented as a dictionary where keys are page numbers and values are the corresponding frame numbers. This setup helps to translate logical addresses to physical addresses.
- Address Translation: The script includes functionality to translate a logical (virtual) address to a physical address using the page table. It checks if the page is present in the table and calculates the physical address if it is; otherwise, it reports a page fault.
- Allocation and Page Fault Simulation: The code simulates memory allocation for pages and handles basic page faults when a non-allocated page is accessed.

Code:

```
# Page simulation in python
class MemoryManagementUnit:
    def __init__(self, page_size, num_pages):
        self.page_size = page_size
        self.num_pages = num_pages
        self.page_table = {i: None for i in range(num_pages)}

    def allocate_page(self, page_number, frame_number):
        """Allocate a page to a specific frame in memory."""
        if self.page_table[page_number] is None:
            self.page_table[page_number] = frame_number
            print(f"Page {page_number} allocated to frame {frame_number}.")
        else:
            print(f"Page {page_number} is already allocated to frame
{self.page_table[page_number]}.")

    def translate_address(self, logical_address):
        """Translate a logical address to a physical address."""
        page_number = logical_address // self.page_size
        offset = logical_address % self.page_size
```

```
        if page_number in self.page_table and self.page_table[page_number]
is not None:
            frame_number = self.page_table[page_number]
            physical_address = frame_number * self.page_size + offset
            print(f"Logical address {logical_address} -> Physical address
{physical_address}")
            return physical_address
        else:
            print(f"Page fault at page number {page_number}!")
            return None


if __name__ == '__main__':
    mmu = MemoryManagementUnit(page_size=256, num_pages=16)


    mmu.allocate_page(0, 5)
    mmu.allocate_page(1, 8)


    mmu.translate_address(100)
    mmu.translate_address(300)
    mmu.translate_address(1024)



    # MemoryManagementUnit Class: This class represents a simple MMU
(Memory Management Unit).
    # Page Table: A dictionary is used to simulate a page table, where keys
are page numbers and values are frame numbers (None if not yet mapped).
    # allocate_page Method: This method simulates allocating a page to a
specific frame in memory.
    # translate_address Method: This method converts a logical address to a
physical address using the page table. It checks if the page exists and
has been allocated. If not, a "page fault" is reported.
```

Output:

Conclusion:

This code provides a foundational understanding of how paging works but is abstracted from the complexities of a real system, such as handling multiple processes, dealing with I/O operations for paging in and out, and implementing sophisticated page replacement strategies. For educational purposes, this script effectively demonstrates the core concepts of paging and can serve as a stepping stone to more complex memory management simulations.