



# **PUNJAB ENGINEERING COLLEGE (DEEMED TO BE UNIVERSITY) CHANDIGARH**



## **Assignment**

**Submitted By :**

**Sugam Arora**

**SID : 21105021**

**Branch : ECE**

**Date : 12th April, 2025**

## Assignment 8

Write a Python program to simulate memory management using **paging**. The program should:

1. Accept the **total size of physical memory** and the **size of each frame**.
2. Accept multiple processes with their **individual memory requirements**.
3. Divide each process into **pages**, allocate available **memory frames**, and construct a **page table** for each process.
4. If memory is full, simulate a simple **page replacement strategy** (e.g., FIFO or LRU).
5. Display the final **page table** and **frame allocation status** for all processes.

### Code :

```
from collections import deque
import math
```

```
class MemoryManager:
```

```
    def __init__(self, total_memory, frame_size):
        self.total_memory = total_memory
        self.frame_size = frame_size
        self.total_frames = total_memory // frame_size
        self.free_frames = list(range(self.total_frames))
        self.allocated_frames = {}
        self.page_tables = {}
        self.frame_queue = deque()
```

```
    def add_process(self, pid, process_size):
```

```
        if process_size <= 0:
            print(f"\n[Error] Process {pid} has invalid memory requirement.")
            return
```

```
        num_pages = math.ceil(process_size / self.frame_size)
        page_table = {}
        print(f"\nAllocating memory for Process {pid} ({num_pages} pages):")
```

```
        for page_num in range(num_pages):
```

```
            if self.free_frames:
                frame = self.free_frames.pop(0)
                print(f" Page {page_num} -> Frame {frame}")
                self.frame_queue.append((pid, page_num))
```

```

else:
    evicted_pid, evicted_page = self.frame_queue.popleft()
    frame = self.allocated_frames.get((evicted_pid, evicted_page))

    if frame is not None:
        print(f" [Replacement] Memory full! Replacing: Process {evicted_pid} Page {evicted_page} from Frame {frame}")
        self.page_tables[evicted_pid][evicted_page] = None
        del self.allocated_frames[(evicted_pid, evicted_page)]

    self.frame_queue.append((pid, page_num))

    page_table[page_num] = frame
    self.allocated_frames[(pid, page_num)] = frame

self.page_tables[pid] = page_table

def display_page_tables(self):
    print("\n=== Page Tables ===")
    for pid, table in self.page_tables.items():
        print(f"\nProcess {pid}:")
        for page_num, frame_num in table.items():
            if frame_num is not None:
                print(f" Page {page_num} -> Frame {frame_num}")
            else:
                print(f" Page {page_num} -> Not in memory")

def display_frame_allocation(self):
    print("\n=== Frame Allocation ===")
    occupied = set()
    for (pid, page), frame in self.allocated_frames.items():
        print(f"Frame {frame} -> Process {pid} Page {page}")
        occupied.add(frame)
    for frame in range(self.total_frames):
        if frame not in occupied:
            print(f"Frame {frame} -> Free")

if __name__ == "__main__":
    try:
        total_memory = int(input("Enter total physical memory size (in KB): "))
        frame_size = int(input("Enter frame size (in KB): "))
        if total_memory <= 0 or frame_size <= 0 or total_memory < frame_size:
            print("Invalid memory or frame size.")
        else:

```

```
mm = MemoryManager(total_memory, frame_size)
num_processes = int(input("Enter number of processes: "))
for _ in range(num_processes):
    pid = input("\nEnter Process ID: ")
    mem_req = int(input(f"Enter memory required by Process {pid} (in KB): "))
    mm.add_process(pid, mem_req)
mm.display_page_tables()
mm.display_frame_allocation()
except ValueError:
    print("Please enter only valid numeric inputs.")
```

### Results:

```
Enter total physical memory size (in KB): 1024
Enter frame size (in KB): 32
Enter number of processes: 8

Enter Process ID: 348592
Enter memory required by Process 348592 (in KB): 512

Allocating memory for Process 348592 (16 pages):
Page 0 -> Frame 0
Page 1 -> Frame 1
Page 2 -> Frame 2
Page 3 -> Frame 3
Page 4 -> Frame 4
Page 5 -> Frame 5
Page 6 -> Frame 6
Page 7 -> Frame 7
Page 8 -> Frame 8
Page 9 -> Frame 9
Page 10 -> Frame 10
Page 11 -> Frame 11
Page 12 -> Frame 12
Page 13 -> Frame 13
Page 14 -> Frame 14
Page 15 -> Frame 15
```

Enter Process ID: