Ishika
2301010242
Btech cse (Sec-4)

CLASSTIME Pg. No.
Date / /

Assignment - 2

PART-A (Short Answer Type):

1. In a multitasking system each process generates logical addresses which the OS and hardware translate to physical addresses using the MMU (Memory Management Unit). The logical address = < page number, offset>. The page number is mapped via the page table to a physical frame, while the offset remains unchanged.
(Illustration: Logical → Page Table → Physical Frame → Physical Address)

2. Suppose memory has free blocks: [200 KB, 300 KB, 500 KB] and processes need 180 KB, 220 KB, 250 KB.

- Internal Fragmentation: occurs when a 200 KB block is given to a 180 KB process (20 KB wasted).
- External Fragmentation: free memory exists but scattered (e.g., 80 KB free in pieces, not usable).
Mitigation: Modern OS uses paging, segmentation with paging, buddy system or slab allocation to reduce fragmentation instead of just compaction.

3. In paging, memory is divided into fixed-size pages and loaded into frames. This avoids external fragmentation, but the trade-offs are:
- Memory overhead: Page tables consume extra memory.

- Speed : Address translation adds delay (improved with TLB caching).
- Fragmentation : Only internal fragmentation (last page not fully filled).

4. The OS maintains page tables, while hardware (MMU) performs translation - TLB (Translation Lookaside Buffer) speeds up lookups, while protection bits (read/write/execute) enforce security. Example: if a page is not in memory, hardware raises a page fault, and OS loads the required page from disk.

5. Given :
   Virtual address = 16 bits
   Address space = 2^{16} = 65,536 bytes = 64 KB
   Page size = 1 KB
   = 2^{10} bytes

(a) Number of virtual pages = 64 KB / 1 KB = 64 pages
(b) Page table size = 64 enteries x 2 bytes
= 128 bytes

PART-B (Application / Numerical Based) :

6. Free memory = 1000 KB. Processes : P_1 = 212, P_2 = 417, P_3 = 112, P_4 = 426

(a) First - Fit :
P_1 (212 KB) → allocated at start (uses first available block)
P_2 (417 KB) → next available block after P_1 leaving 371 KB free.

P3 (112 KB): allocated to the first available block (371 KB) leaving 259 KB free.
P4 (426 KB): Cannot be allocated as no block is large enough.

Best - Fit Allocation:
Best Fit selects the smallest free block that fits the process.
P1 (212 KB) → fits into initial block.
P2 (417 KB) → next smallest block fitting 417 KB
P3 (112 KB) → smallest block fitting 112 KB → allocated
P4 (426 KB) → not allocated

Worst - Fit allocation:
Worst - Fit allocates the largest available block first.
P1 (212 KB) → largest block used
P2 (417 KB) → largest remaining block used
P3 (112 KB) → next largest block
P4 (426 KB) → not allocated

(b) All strategies left 259 KB unused memory in this scenario.

(c) Best - Fit is generally preferred bcoz it minimizes leftover gaps hence better utilization in most scenarios.

7. Reference string = 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2
No. of frames = 3

(a) 1. FIFO (First - In First - Out):
Page faults = 10
Pages are replaced in the order when entered memory.
2. Optimal LRU (Least Recently Used):
The page that has not been used for the longest time is replaced.
Total page faults = 9
3. Optimal:
The page whose next use is farthest in the future (or never used again) is replaced.
Total page faults = 7

(b) FIFO : 10 page faults
LRU : 9 page faults
Optimal : 7 page faults

(c) Optimal performs best as it replaces the page not needed for longest time. FIFO shows Belady's anomaly (more frames may increase faults) while LRU approximates optimal but depends on past usage.

8. (a) Total pages replaced = 1000
Percentage of dirty pages = 30% → 300 dirty pages
Overhead per dirty page = 10 ms
Total overhead = 300 x 10 ms = 3000 ms = 3 seconds

(b) To reduce this delay, the OS can use a write - back policy with a dirty - bit. In this method

only modified pages are written back to disk when necessary. Additionally, techniques like buffer caching and asynchronous writes allow overlapping of I/O with computations, further minimizing overhead.

9. (a) The OS tracks the working set of critical tasks (e.g. object detection) and ensures their pages remain in memory, preventing thrashing. Less critical tasks (like infotainment) adapt by releasing frames during high demand. A priority-based page replacement policy ensures real-time tasks are not delayed.

(b) A dynamic priority-based allocation with real-time reservations is suitable. Critical tasks receive guaranteed minimum frames for responsiveness, while remaining memory is shared among non-critical tasks. This balances real-time safety with overall system efficiency.