Rohini Dhankhar 230101026 6
B.tech Cse Core-D

Operating System Assignment - 2

PART- A

Ans. 1) Logical Address → Physical Address
- CPU generates a logical address (virtual address).
- The Memory Management Unit (MMU) translates it into a physical address.
- Uses page table (in paging) or segment table (in segmentation).

Example:
Logical Address = Page No, offset)
Page Table → frame No
Physical Address = (Frame No ; offset)

Ans. 2)
- Internal fragmentation : A 200KB block given to a 180 KB process leaves 20KB wasted inside.
- External fragmentation: Free blocks [100 KB , 300KB] process requires 250 KB → cannot fit despite enough total free memory
- Solution (beyond compaction):
  - Paging (removes external fragmentation).
  - Segmentation with paging
  - Dynamic allocation (Buddy system , slab Allocation)

Ans 3) • Memory divided into fixed-size pages (e.g. 4KB).
• Process is divided into pages & stored in free frames

• Trade-offs:
• Extra memory overheard (page tables)
• Access speed reduced due to page table lookup, improved with TLB.
• No external fragmentation, but last page may have internal fragmentation.

Ans 4) • Hardware support:
• Page Table Base Register (PTBR)
• Translation Lookaside Buffer (TLB)
• Protection bits in page table entries
• Interaction:
• If page is present → physical address generated.
• If not present → page fault → OS loads required page from disk → updates page table

Ans 5) • Virtual address = 16 bits → address space = 2^{16} = 65,536 bytes
• Page size = 1KB = 2^{10} bytes
• No. of virtual pages = $2^{16}/2^{10}$ = 2^6 = 64 pages
• Page table entries = 64, each of 2 bytes → Page Table Size = 128 bytes

PART - B

Ans 6) • Given:

- Total Free memory = 1000 KB
- Processes:
  - P_1 = 212 KB
  - P_2 = 417 KB
  - P_3 = 112 KB
  - P_4 = 426 KB

a) First-Fit Allocation

- P_1 → allocated (212 KB) → remaining = 788 KB
- P_2 → allocated (417 KB) → remaining = 371 KB
- P_3 → allocated (112 KB) → remaining = 259 KB
- P_4 → requires 426 KB → cannot fit (only 259 KB left)

Unused memory = 259 KB

b) Best-Fit Allocation

- P_1 → allocated (212 KB) → remaining = 788 KB
- P_2 → allocated (417 KB) → remaining = 371 KB
- P_3 → allocated (112 KB) → remaining = 259 KB
- P_4 → ~~me~~ needs 426 KB → cannot fit

Unused memory = 259 KB

c) Worst-Fit Allocation.

- P_1 → allocated (212 KB) → remaining = 788 KB
- P_2 → allocated (417 KB) → remaining = ~~259~~ 371 KB
- P_3 → allocated (112 KB) → remaining = 259 KB
- P_4 → needs 426 KB → cannot fit

Unused Memory = 259 KB

Final Results

- First fit unused : 259 KB
- Best fit unused : 259 KB
- Worst fit unused : 259 KB

Conclusion:

All three methods give the same utilization in this case. None can allocated P_4.

Ans-7 Reference string : 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2

Frames : 3.

a) FIFO
- Replaces oldest page
- Total 8 page faults

b) Optimal
- Replaces page not needed for longest time
- Total 6 page faults (best)

c) LRU
- Replaces least recently used page
- Total 9 page faults.

b) Summary:
- FIFO = 8
- Optimal = 6
- LRU = 9

c) Best performance & Belady's anomaly:
- Optimal performs best (minimum faults)
- LRU is a practical approximation
- FIFO can suffer from Belady's Anomaly, while Optimal & LRU do not

Ans-8 Given: disk write 10 ms, memory write = 100 ns, 30% of replaced pages are dirty, 1000 pages replaced.

a) Additional time overhead due to dirty pages

- Dirty pages = 30% of 1000 = 300 pages.
- Time to write one dirty page to disk = 10 ms
- Total disk write time = 300 × 10 ms = 3000 ms = 3 s
- Memory write contribution: 300 × 100 ns = 30,000 ns = 0.00003 s - negligible

Answer. 3.0 seconds extra overhead (dominant cost from disk writes)

b) Optimization techniques

- Use write-back with dirty bit (write only when needed)
- Perform asynchronous / background writes so eviction doesn't block.
- Batch writes together to reduce I/O overhead.

These reduce the extra 3 seconds of overhead

Ans-9) Given: multiple memory-intensive tasks (object detection - mission-critical, route planning, infotainment - less critical)

a) Using working set model & page replacement

- Working set: Monitor each task's working set (recent pages it actively uses)
- Guarantee frames to critical task: Reserve enough frames for object detection equal to its working set so it won't thrash
- Replacement policy: Use Working-Set-aware LRU (or true Working set algo) that keeps pages for tasks whose working set is active; prefer

evicting pages of low priority tasks (infotainment).

- Result : Object detection keeps required pages in memory / low page faults ; less critical tasks adapt or are throttled

b) Memory allocation strategy

- Priority - based dynamic allocation with quotas.
- Reserve a minimum guaranteed frame quota for real - time, safety tasks (object detection, sensor fusion)
- Allow remaining frames to be shared dynamically using a global LRU / working set policy.
- If memory pressure rises, demote or throttle non - critical tasks (reduce their quota / lower fidelity).
- Justification : Ensures real - time responsiveness & avoid thrashing for critical tasks while keeping overall memory utilization efficient & fair.