~~OS Assignment~~

OS Assignment-2

Ayush Gupta
2301010206
BTech CSE - D

Part-A

1). Address translation with multiple processes.
- Each process has its own page table.
- MMU + TLB translate virtual to physical addresses.
- On context switch, OS loads new page table base; ASIDs avoid TLB flush.
- Page faults handled by OS to bring pages from disk.

2) Layout causing both internal & external fragmentation + solutions.
- External: variable-sized allocations create small scattered holes.
- Internal: paging wastes space in last page (e.g. 4100B → 4+1 pages).
- Fixes: paging (removes external), segmentation + paging hybrid, buddy & slab allocators, coalescing free lists, memory compression (zram), variable page sizes, GC compaction.

3) Paging-based model & trade-offs
- Fixed-size pages (eg. 4KB), multi-level page tables, TLB cache, swapping.
- Small pages: less internal waste, more page table entries, more TLB misses.
- Large pages: fewer TLB misses. smaller page tables more internal waste.

- multi-level tables reduce memory use but slow on TLB miss.

4) OS-Hardware interaction in virtual memory

- Hardware: MMU, TLB, page table base register (CR3), ASIDs, page-fault trap.

- Steps: TLB hit → direct; TLB miss → page-table walk; if not present → page fault - OS loads page → update PTE → resume.

5. Given: Virtual address width = 16 bits → virtual address space size = 2^{16} bytes = 65536 bytes.

Page size = 1KB = 1024 bytes.

size per page-table entry (PTE) = 2 bytes.

a) No. of virtual pages:

$$\text{No. of pages} = \frac{\text{virtual address space}}{\text{page size}} = \frac{65536}{1024}$$

$$= 64 \text{ virtual pages.}$$

b) Page table size (bytes) = No. of pages × PTE size
= 64×2
= 128 bytes.

Page No:
Date: / /

Part-B

6. ~~Memory~~ Memory allocation simulation

Given: free memory = 1000 KB. Process - P1 = 212 KB, P2 = 417 KB, P3 = 112 KB, P4 = 426 KB.

First - fit
1. Start hole = 1000 KB
2. Place P1 (212 KB) → remaining hole = 1000 - 212 = 788 KB
3. Place P_2 (417 KB) ⇒ remaing hol = 788 - 417 = 371 KB
4. Place P_3 (112 KB) ⇒ remaining hole = 371 - 112 = 259 KB
5. Try P4 (426 KB) ⇒ 259 KB > 426 KB ⇒ can't place.

Result (First - fit): allocated P_1, P_2, P_3; free leftover = 259 KB; P4 unallocated

Best-fit: same sequence here (only one hole at each step) → 259 KB leftover (P4 unallocated).

Worst - fit: same result 1 259 KB leftover.

Conclusion: All three produces identical results for this arround pattern: 259 KB unused. P4 not placed.

7. Page Replacement

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

I'll give concise step outcomes (frame contents after each reference shows when a change happens).

Page No.: ______
Date: ___/___/______

FIFO (replace oldest)
Step through references - faults occur when page not in frames.

• After references and replacements in FIFO page faults = 10

Optimal (replace page whose next use is farthest)

Using future knowledge, Optimal choose best victim.

• Optimal page faults = 7.

LRU (replace least - recently used)

LRU page faults = 9

Algorithm	Page faults
FIFO	10
Optimal	7
LRU	9

8. Dirty page write overhead.

Given: disk WT = 10ms/page, memory write = 1ons/page, dirty fraction = 30%, pages replaced = 1000

Step-1 - No. of dirty pages:
$1000 \times 0.3 = 300$ pages.

Step-2 - Disk wait time total:
$300 \times 10 = 3000$ ms $= 3$ secs.

Page No:
Date: / /

Step-3 Memory write time (for comparison):

$300 \times 100 ns = 30000 ns = 0.00003 s$ (negligible)

Extra overhead ~0.3 secs. due to dirty-page disk writes for 1000 replacements (30% dirty)

9. Autonomous vehicle memory management

- Working set model: give real-time tasks (object detection) enough frames - their working set to prevent trashing.

- Policies:
  - lock critical pages in RAM (mlock), use WSclock or PFF control.
  - Reclaim from low-priority tasks if critical page-fault rate rises.
  - Use memory compression + quotas (cgroups) for infotainment.
  - Prioritized async writes to avoid blocking real-time tasks.

Checked 26/11/25