14/5

ASSIGNMENT - 2

Ans 1) Address translation
logical (virtual address) → MMU + Page table → Physical address
Each process has its own page table
TLB speeds up translation
If page not in memory → page fault → OS loads it

Ans 2) Internal Fragmentation :- It happens when a process doesn't use all the memory given to it.
External Fragmentation :- It occurs when free memory block are scattered around & can't be combined to fit a big process.

Memory Layout
M.M → 64Kb

	Size ( )	Process	Type of Fragmentation
P_1	10	OS	—
P_2	12	Process A (10Kb)	2Kb Internal
P_3	8	Free	—
P_4	16	Process B (16Kb)	—
P_5	6	Free	—
P_6	12	Process C (10Kb)	2Kb Internal

Here, we have 8Kb & 6Kb → 14Kb free but a new process D (12Kb) can't be allocated because no single contiguous block is large enough.
Paging can be used to mitigate fragmentation where we divide both memory & processes into fixed-size blocks (pages & frames)

Ans 3) Logical memory is divided into fixed-sized pages (eg 4kb) & physical memory into frames of the same size. Each process has a page table that maps its pages to available frames when a process accesses memory, the OS uses the page table (& the TLB) to translate virtual addresses to physical ones.

Trade offs	Benefits	Drawback
Memory overhead	Page tables enable flexible allocation	Each process needs its own pg table, consuming extra
Speed	TLB speeds up translation	TLB misses & multi-level pg table walks slow down access.
Fragmentation	Removes external fragmentation	Introduces internal fragmentation.

Ans 4) OS & hardware (MMU) work together to translate virtual addresses to physical addresses.
MMU → translates virtual → physical addresses.
TLB → caches recent address translations.
Page Table → stores mapping of page to fames.
Ex :- when a process accesses memory, the MMU checks the TLB; if not found, it looks up the page tables. If the page is not in memory hardware triggers a page fault, & the OS loads the page from disk and updates mappings.

Ans 5) Virtual address = 16 bits = 2^{16} bytes
Page size = 1kb = 2^{10} bytes
a) No. of virtual pages = $2^{16} - 2^{10} = 2^6$ bytes = 64 pages

b) Each Entry = 2 bytes

Page Table Size = $\underbrace{64}_{2^6} \times 2 = 128$ bytes

Ans 6)

	Size (Kb)
P1	212
P2	417
P3	112
P4	426

First Fit

P4 x
P3 ✓
P2 ✓
P1 ✓

→ 259 Kb unused

741 Kb used

Best Fit

P4 x
P3 ✓
P2 ✓
P1 ✓

unused = 259 kb

Used = 741 Kb

Worst Fit

P3 x
P1 ✓
P2 ✓
P4 ✓

→ unused = 45 Kb

used = 955 Kb

Among these three, worst-fit algo is the best utilization as it has very less unused space among these three.

Ans 7) For FIFO : 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2.

		1	1	1	1	0	0	0	3	3	3	3
	0	0	0	0	3	3	3	2	2	2	2	2
7	7	7	2	2	2	2	4	4	4	0	0	0

Page Fault = 10

For optimal : 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2.

		1	1	1	3	3	1	1	3	3	3	3
	0	0	0	0	0	0	0	0	0	0	0	0
7	7	7	2	2	2	2	4	4	4	2	2	2

Pg Fault = 7

For LRU: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2

	7	0	1	2	0	3	0	4	2	3	0	3	2
	0	0	0	0	0	0	0	0	3	3	3	3	
7	7	7	2	2	2	2	4	4	4	0	0	0	

Pg Fault = 9

Ans 8) Disk Write = 10 ms = 0.01 s
Memory write = 100 ns = 100×10^{-9} s = 0.0001 ms
Dirty Pages = 30%

a) For 1000 pages, Dirty pgs = 30% × 1000 = 300
Extra time per dirty pg = 10 ms − 0.0001 ms ≈ 10 ms
For 300 dirty pgs, extra time = 300 × 10 ms = 3000 ms
= 3 sec

b) Optimization technique that could reduce this overhead. Can be background page cleaning (write-behind) – asynchronously flush dirty pgs in idle time & write in batches.

Ans 9) (a) The OS uses the working set model to keep mission critical total active pages in memory, preventing thrashing less critical tasks get memory dynamically reduced when needed. A WS or WS clock replacement evicts pages from low-priority process first.

(b) Using priority – based dynamic allocation with reserved memory for real-time tasks ensures fast response for critical function while efficiently sharing leftover memory among the processes.