Assignment - II

Address translation with multiple processes. In modern systems with multiprogramming, multiple processes coexist in memory. Each process is given a logical (virtual) address space, while the actual data resides in physical memory (RAM). The translation is managed by Mumory management Unit (monu) with help of page tables. (i) Logical Address Generation (ii) mone translation via page table (iii) Physical Address Cornation

(B) Access to Physical Memory

2. Memory loyant with fragmentation.

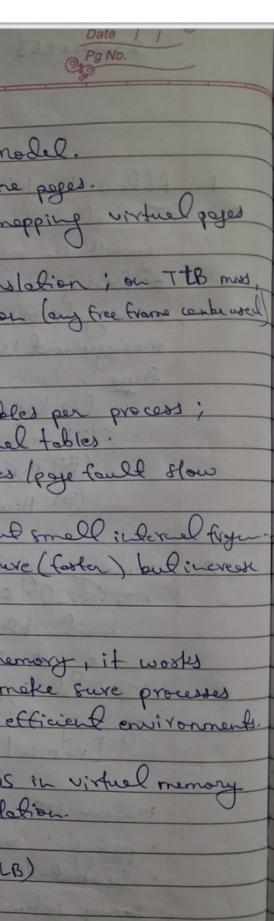
. Internal Fragmentation: Fixed partitions wester space

eg. Block = 8 KB, Process needs 6KB -> 2KB wested.
* External Evagnus delier: free memory is stateved.

cg. layout Tromb | Grees mb | 20 mb | Freezons | 15 mb | Gree 6 mb

modern 05 solutions (byond compaction)

- · peging: removes external fragmutation.
 · Segmentation + Paging: Reduces both types
- · Buddy system: Split menges memory in powers of two
- · Sbb allocation: Efficient terral memory use
- · Vivtuel memory: Non-contiguous allocation avoids external Evagmentation



3. Paging-based memory Allocation model.

. memory is divided into fixed sine pages.

. Each process has a page table mapping virtual pages

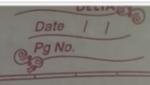
- physical frames.

. mmv + Tib & hardle fast translation; on Ttb miss.

. eliminates external fragmentation (any free frame cambinosed) Trade - offs · memosy overhead: large page tables per process;
mitigales with multi-level tables. - Speed: The hits are fost; musses / page foult slow done execution · No external fragmentation; but small; ideral fryu-· Huge pages : Peduce TLB pressure (foster) bulineresse internel meste. 4. When an OS manages a virtual memory, it works closely with the hordware to make sure processes run in Holated, protected, and efficient environments. Interaction blu Herdware and OS in virtual memory

(i) virtual to physical Address translation. (ii) Page tables (iii) Translation lookeside Butter (TLB) (1) Memory Protection
(2) Page toulls and Demand paging
(2) Herdware support for Holetion

Key Hordwore Structures
· mmv : houldes alless translation
· mmu : handles alleress translations mapping virtual popul to · Page tables: Os managed structures mapping virtual popul to · TLB: handware cache for East translations. Physical translations.
· TLB: herdware cache for East translations. I physical trans
5. vitual allows sine = 16 bits page sine = 1 KB = 1024 bytes = 210 bytes (medice) Page table cutry (PTE) fine = 2 bytes
page size = 1 KB = 1629 bytes 2 botes
Page table cutry (PTE) fire -
. 0 00 101 100
(a) virtual odl = 16 bits total Page Fire = 210 -> 10 bits used for offert within a page
Page 6-e - 2 -3 (0 B) & 15/2 -
Remaining bits for pope no = 16-13
So, pgr no-field = 6 bits page offeel field = 10 bits
no- of virtual pages = 2 = 64 virtual pages
no- of virtues page
1 0 0 0 0 00 and in some table
(b) Foil virtual page reads one entry in page table
no. of entries = 64
Fool enly = 2 bytes
0 - CV X 2 - 128 bules =
PART - R total give = C4 x2 = 128 bytes
6. Memory Allocation Simulation
A system has 1000 kB of Free mamory.
Process Sine (KB)
P1 212
P2 417
P3 112
P., 926



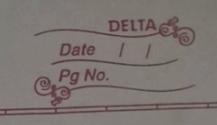
Using first-Fit, Best-Fit and Worst-Fit Est- Ct Stort = 1000 KB Place P1 (212 KB) -> remaining = 1000 - 212 = 788 KB Place P2 (417KB) -) remaining = 788-417 = 371 KB Place P3 (112 KB) -> remaining = 371-112 = 259 KB Try P4 (426KB) -> 259KB 2426KB => court place Result: Allocated Pr, P2, P3; free = 259 XB Best- Et: Seme seguence here (only one at call step) -> 259 KB Jeffouer (Py unallocated) Worst-fit: some result -> 259 KB Jeffour Conclusion: All three produces identical results for this allocation. 259 KB which · Py not placed 7. Page reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2 and 3 frames GIFO Ref 7 0 1 2 0 3 0 4 2 3 F1 7 7 7 2 2 2 2 4 4 4 F2 - 0 0 0 0 3 3 3 2 2 F3 - - 1 1 1 1 0 0 0 3 Foult F F F F H F F F Total (IFO page foults = 10

Date | |

Pg No.

										@Pg N	0.			100
-	- ,	0 1	0.0	0.	, ,	0.0	1							100
- '	Optime	el (belle	and	3	-0	,	Fair	00		200			20
Re	C	E	2	2	9	-3		Tan	_					
7	1000	7	4	0	01	-	2112 115	7		0.				
6	0	7	7	1000	920		254	600	_	- 1				
١	\	7	03	0	= 6	200	Me	049	T	70				
3	2	2		0		'			-	1.70				
	0	2	end	0	12/20	1	Veni	1	-	Here	-			
9	3	2		0	Sage.	3	39	S. Line	F	10	7-11			
	0	2		0		3			H					
1	4	4	100	2	-46	0		4 =	+	9 /-	100			
	2	4	600	2	2	0)		H	17	12:00			
	2 000	4	01	2	74	0100	3	-	F	1	sanaan	,		
	0	0		-	2		3		F					
	3	0		4.	2		3	= 1	H	at 5	10967			
	2	6			2	0.0	3	0	+	1		21		
		0		0		6	204	1	7	Cmil	- 6 ogs	ible)	
	Total	10	byu	معلا	Code	Co	1	64	470	10	2.7			
	,	. 0	_	0	0 .	1200	10	14-11						
1	RU (least	Ree	ent	4	080								
			1000	224			0	_	4	2	3	0	3	2
P	Ref	7	0	1	2	0	3	0	4		4		0	
F	- June	7 7	7	17	2	2	7	-	10	0	3	2	3	3
	Fz		0	0						_	2	2	2	2
	F3	7	-	1	1	10	3	3	3	2	2	E	14	H
	fould	F	F	E	F	H	F	H	O.F.	OF.	F		- ' '	11
110														
	1-	0_0	100	001	•	aul d	3 =	9	v 7	22	0000			
	Order	1		13		4)	M	:						
	Order	1 6:	15est	,	2000	2 1)	19)	2	EIE	0 ((0)			
	Op	timel	(7) <	-									
									THE RESERVE OF THE PERSON NAMED IN					

Dist write time = 10 ms per page memory unite time = 100 ns per page replaced pages that one disty = 30.1. (a) Additional time overhead due to dirty poper when replaint 1000 poper 0.30×1000 = 300 Disty pg.s = Dist time = 300×10 ms = 3000 ms = 3 second 1000 x 100 ns = 100,000 nd = 0.1 md memory time = Total time spend = disk time + mem So, additionel overhed = 3 sec. total (dirk+ mem) = 3000 ms + 0. 1 - 3000.1 ms. (b) Proposed optimination to reduce this overhead. Best single practice optimization: Background pre-claiming & preton clear victims.
Two linked ideas that one commonly used together: · page-cliener deemon (bedeground write-beck) · Clear first victim selection (culoual clock/modificallet)



9. Autonomous Velicle Case Study or Working set model + replecement policy.

. OS tracks recent active poses par Jask. for object delection: Allocate stable working sit Con informent: Allows Clexible replacement es it adopts to avoilable memory. Memory Allocation Strategy

Voer priority-bosed dynamic allocation

Real-time responsiveness ensured by working

real time shedule.