Operating Systems Design

19003028

Home Assignment-3

Ch. Lakshmi Sarayu

1A. If there are free page frames, grab one. If not, must evict something else. This is called page replacement.

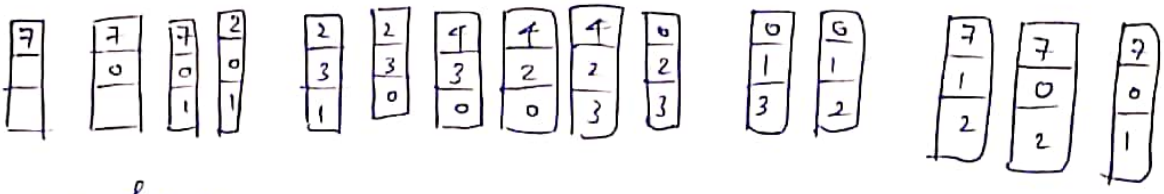
Given Reference string : 7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1

FIFO (First In First Out)

→ 3 frames (3 pages can be in memory at a time per process)

References string

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



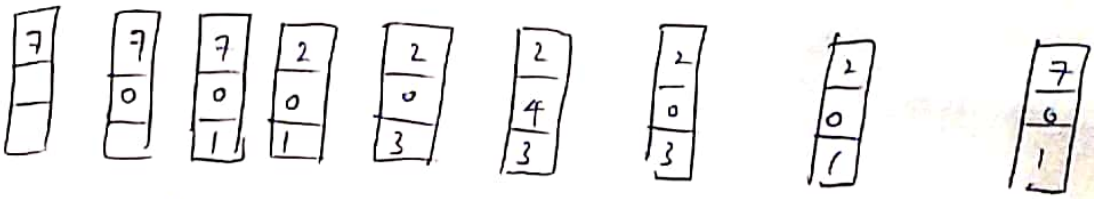
page frames

15 page-faults

optimal: Replace page that will not be used for longest period of time

Reference string

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



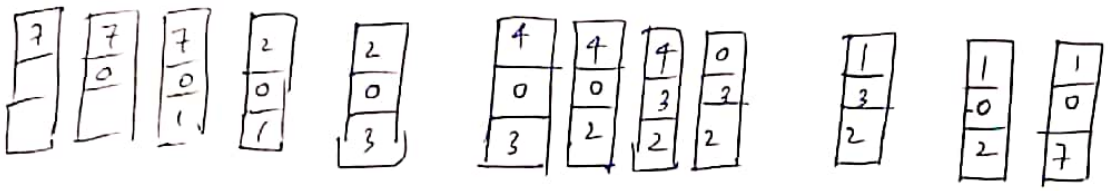
Page faimers.

Page faults

LRU (Least Recently Used):

Reference string.

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



12 page faults

→ LRU better than FIFO but worse than optimal

2. Ans: Given,

page reference string: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

LRU → Frame-1

20 page faults

Element	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame-1	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6

Frame-2

18 page faults

Element	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame-1	1		3		2		5		2		NF		7		3		1		3	
Frame-2		2		4		1		6		1		3		6		2		NF		6

Frame-3, 15 page faults

Element	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame-1	1			4			5			1			7			2		NF		
Frame-2		2			NF			6				3			NF				NF	
Frame-3			3			1			2		NF			6						6

Frames = 4, 10 page faults

Element	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3
Frame-1	1			1		NF				NF				6					
Frame-2		2			NF				NF		NF					NF		NF	
Frame-3			3				5					3			NF				NF
Frame-4				4				6					7				1		

Frames = 5, 5 page-faults

Element	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame-1	1					NF				NF							NF			
Frame-2		2			NF				NF		NF				NF	NF		NF		
Frame-3			3									NF			NF				NF	
Frame-4				4			5						7							
Frame-5								6						NF						NF

Frame = 6, 7 page faults

Element	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
Frame-1	1					NF				NF							NF			
Frame-2		2							NF		NF					NF		NF		
Frame-3			3									NF			NF				NF	
Frame-4				4																
Frame-5							5						7							
Frame-6								6						NF						NF

Frame = 7 , 7 page faults

Element	1	2	3	4	2	1	5	6	2	1	2	3	7	6	5	1	1	2	3	6
Frame-1	1					NF				NF							NF			
Frame-2		2			NF				NF		NF					NF		NF		
Frame-3			3									NF			NF				NF	
Frame-4				4																
Frame-5							5													
Frame-6								6						NF						NF
Frame-7													7							

Like above we will calculate faults for FIFO, OPTIMAL & clock replacement pages

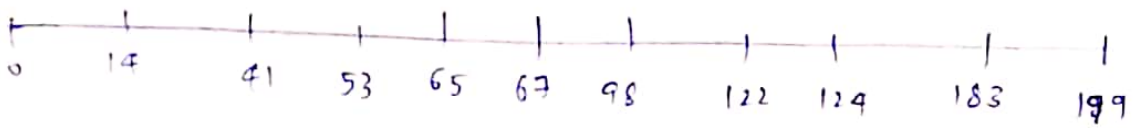
	No. of faults			
No of faults	LRU	FIFO	OPTIMAL	CLOCK
Frame-1	20	20	20	20
Frame-2	18	18	15	17
Frame-3	15	16	11	14
Frame-4	10	14	8	10
Frame-5	8	10	7	10
Frame-6	7	10	7	7
Frame-7	7	7	7	7

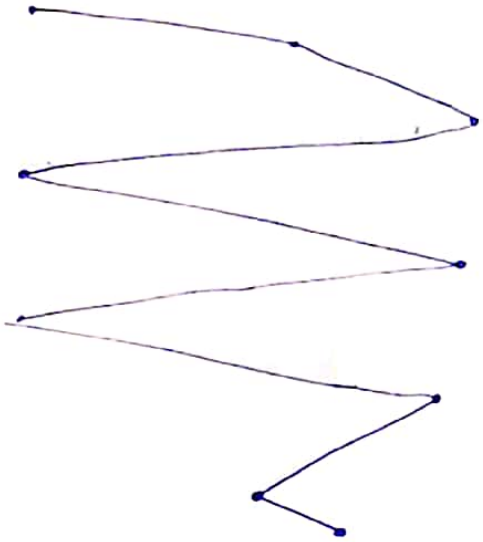
3Ans Total head movements incurred while serving these requests

$= (98-53) + (183-98) + (183-41) + (122-41) + (122-14) + (124-14) + (124-65) + (67-65)$

$= 45 + 85 + 142 + 81 + 108 + 110 + 59 + 2$

$= 632$





4Ans SCAN:

queue = 98, 183, 37, 122, 14, 124, 65, 67

Head starts at 53

Total head movements

$= (53-37) + (37-14) + (14-0) + (65-0) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124)$

$= 236$

4) C-SCAN:

queue = 98, 183, 37, 122, 14, 124, 65, 67

Head starts at 53.

Total head movements

= (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (199-183) + (199-0) + (14-0) + (37-14)

= 382

6) Look: Total head movements

(67-53) + (183-98) + (122-37) + (124-122) + (183-124) + (67-65) + (183-37) + (37-14)

= 416

7) C-Look: Total head movements:

(67-53) + (183-98) + (122-37) + (124-122) + (183-124) + (67-65) + (183-14) + (37-14)

= 439

5) If any computer program needs more RAM than there is a variable - it starts to use so called "page-file", a substitute of RAM that is actually a file on the harddisk. As the HDD is usually much ~~better~~ slower than RAM, the program performance get worse, but it still works.

No. of entries in pagetable = (virtual address spacesize) / (page size)

using above formula, we can see that there will be

$(2^{32-12}) = 2^{20}$ entries in page table.

No. of bits required to address 64MB physical memory = 26

so, there will be $2^{(26-12)} = 2^{14}$ page frames in. Therefore, each page contains 14 bits address of page frame & 1 bit for valid-invalid bit

so, memory is a byte addressable. so we take, each page table entry is 16 bits. i.e., 2 byte long size of page table.

(Total no of pagetable entries) * (size of pagetable entry)

$(2^{20} \times 2) = 2MB$//

9) (a) For 8 bit virtual address, and 4 bit page offsets (page size 16 bytes), the most significant 2 bits of a virtual address, will represent page number. so, reference string is 0,0,1,0,1,1,2,1,0,3 (repeated again)

(b) page faults with FIFO. 8 page faults on 0,1,2,3 (replaced 0) 0 (replaced 1), 1 (replaced 2), 2 (replaced 3), 3.

(c) page faults with LRU = 6. page faults on 0,1,2,3 (replaced 1), 2 (replaced 3), 3.

(d) The optimum algorithm will replace page least likely to be used in future, and would look like LRU above.

10) we have $t_x = h * t_h + (1-h) * t_m$

so, $t_h = \frac{t_m - t_x}{t_m - t_h}$

11) 20 = 010100 = 110100 = 52

40 = 101000 = 10111000 = 184

12) $0.6 * t_1 + 0.3 * t_2 + 0.1 * t_3$

13) page size = 2^{14} bytes

so, no of pagetable entries = $2^{48}/2^{14} = 2^{34}$

Each page can store 16KB/4 = 2^{12} page table entries

so, no. of innermost pages = $2^{34}/2^{12} = 2^{22}$

Now, pointers to all innermost pages must be stored in next level page table so, next level of pagetable has $2^{22}/2^{12} = 2^{10}$ pages

Finally, a single page can store all 2^{10} page table entries. so, outermost level has one page

so, total no. of pages the store pagetable entries is $2^{22} + 2^{10} + 1$

14) ceil $((64-12)/(12-2)) = 6$

2, 10, 10, 10, 10, 10 (starting from most significant to least)

Innermost level has 2^{52} PTE's, which fit in 2^{42} pages.

Next level has 2^{42} PTEs which require 2^{32} pages & so on.

Total pages = $2^{42} + 2^{32} + 2^{22} + 2^{12} + 2^2 + 1$

15) no. of PTEs per process = $2^{16}/2^8 = 2^8$

no. of PTEs per page = $2^8/2^2 = 2^6$

no. of inner page table pages = $2^8/2^6 = 2^2 = 4$.

which requires one outer page directory. so total pages = 4 + 1 = 5

1(c) page size = 8KB = 2^{13} B

Virtual address space size = 2^{46} B

PTE = 4B = 2^2 B

no. of pages or no. of entries in pagetable,

= (virtual address space size) / (page size)

= 2^{46} B / 2^{46} B

= 2^{33}

size of pagetable,

= (no. of entries in pagetable) * (size of PTE)

= $2^{33} * 2^2$ B = 2^{35} B

To create one more level, size of pagetable > page size

no. of pagetable in last level,

= 2^{35} B / 2^{13} B = 2^{22}

Base address of these tables are stored in pagetable (2^{nd} last level)

size of pagetable[second last level]

= $2^{22} * 2^2$ B = 2^{24} B

To create one more level,

size of pagetable[2^{nd} last level] > page size

No. of page tables in second last level = $2^{24} / 2^{13}$ B = 2^{11}

Base address of these tables are stored in page table [3^{rd} last level]

size of pagetable [3^{rd} last level]

= $2^{11} * 2^2$ B = 2^{13} B = page size

∴ 3 levels are required

17. Given :

Virtual address space = process size = 2^{16} bytes

physical address space = Main memory = 2^{16} bytes

process is divided into 8 equal size segments

pagetable size entry = 2 bytes = 16 bits

pagetable entry besides other information contains

(1 valid bit, 3 protection bits, 1 dirty bit

page size = 512 bytes.

No. of frames in Main memory

no. of frames in main memory = (size of main memory) / (page size)

$= 2^{16}$ bytes / 512 bytes

$= 2^{16}$ bytes / 2^{9} bytes

$= 2^{7}$ frames

Thus, no. of bits required for frame identification in pagetable entry = 7 bits.

No. of bits available for storing aging info

= no. of bits in pagetable entry - (no. of bits req for frame identification + 1 valid bit + 3 protection bits + 1 dirty bits)

= 16 bits - (7+1+3+1) bits

= 16 bits - 12 bits

= 4 bits

(b) Given,

virtual address space = 2^{16} bytes

physical address space = 2^{16} bytes

page table entry size = 2 bytes

Let page size = n bytes

since page table has to be stored into single page, we must have

size of page tables <= page size

size of each segment = process size / no. of segments

$\therefore 2^{16}$ bytes / 8 = 2^{13} bytes = 8KB

no. of pages of each segment = size of segment / page size

= 8 B / n bytes = (8K/n) pages

size of each page table = no. of entries in page table x page table entry size

= no. of pages the segment is divided x 2 bytes

= (8K/n) * 2 bytes

= (16K/n) bytes

page size = (16K/n) bytes <= n bytes

$(16K/n) <= n$

$n^2 >= 16K$

$n^2 >= 2^{14}$

$n >= 2^7$

min page size = 2^7 bytes = 128 bytes

Division for virtual address:

no. of segments the process is divided = 8

$= 2^3$

no. of bits = 3 bits

no of pages a segment is divided : segment size / page size

$= 8KB / 128 bytes$

$= 2^{13} bytes / 2^{7} bytes$

$= 2^{6} pages$

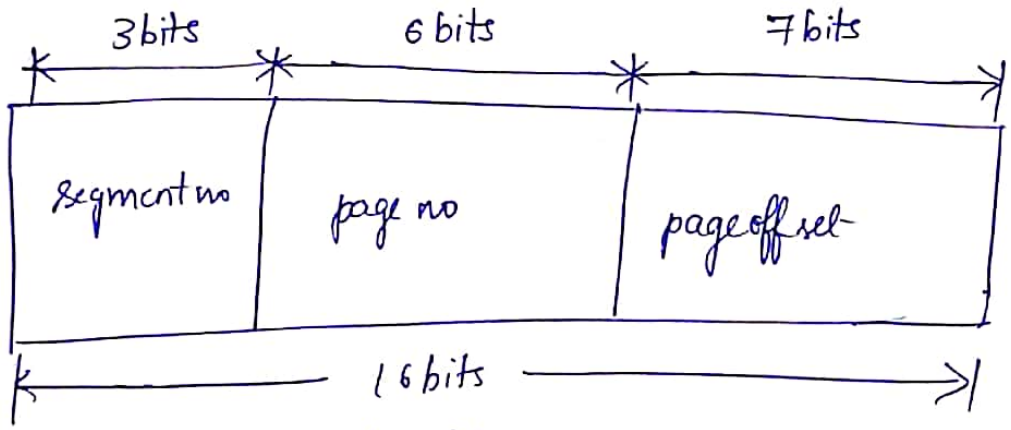
no. of bits = 6 bits

no. of bits req for page offset

page size = 128 bytes $= 2^{7}$ bytes

no. of bits = 7 bits

Thus, virtual address is divided as :-



virtual address