DATE: / /
PAGE:

Webinar - (111) : 31st July 2021 (Saturday)

Non-contiguous Memory Allocation

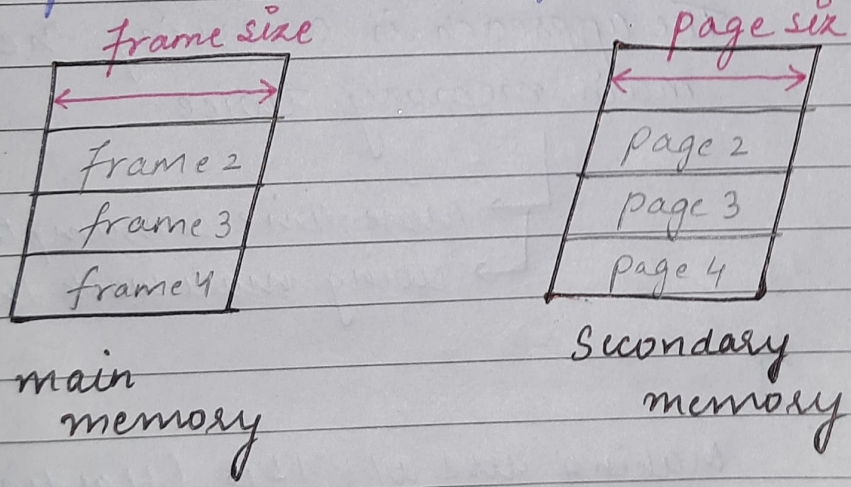
↳ a process is stored in diffrent parts in main memory
↳ External Fragmentation takes place

(#) PAGGING is done to remove External fragmentation

PAGGING TECHNIQUE

↳ divide secondary memory into equal parts where each part is known as page.

↳ the main memory is also divided into equal parts of same size as of pages. Here each part is known as a frame



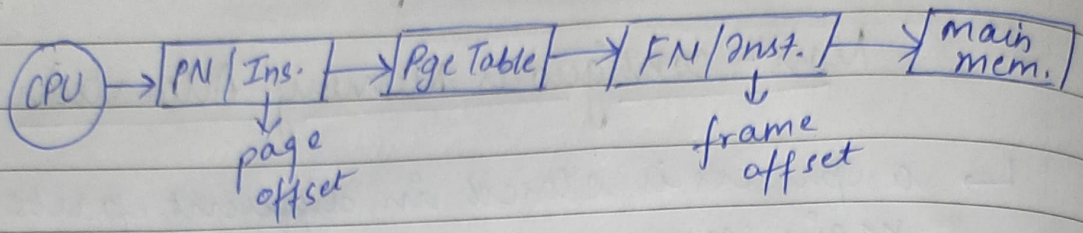
↳ frame size == page size

↳ Now CPU sends logical address but main memory needs physical address.

logical address ⇒ | PageNo | Instruction |

Physical address ⇒ | Frame No | Instruction |

DATE: / /
PAGE:



What is Page Table?

↳ Map page No to frame No.

P0	F0
P1	F2
P2	F3
P3	F1

* Frame 0 stored at page 0
* Frame 2 stored at Page 1
and so on....

So | P0 | 55 | ⟹ | F0 | 55 |

and | P1 | 55 | ⟹ | F2 | 55 | and so on...

Problem Here!

The approach is accessing the main memory twice

↳ More time consumption
↳ doing more work than req.

Solution!

Making use of TLB (Hardware)
(Translation lookaside Buffer)

↳ Initially Empty
↳ Get updated after page table
↳ Is checked before ~~TLB~~ Page Table.

Page Replacement Algorithm

DATE: / /
PAGE:

FIFO Optimal LRU.

First In First Out

- oldest page / front page removed first.

Example: 7, 6, 1, 2, 6, 3, 6, 4, 2, 3, 6, 3, 1, 2, 6

Element	Frames(4)	Result
7	7	Miss! fault+1
6	7 6	Miss! fault+1
1	7 6 1	Miss! fault+1
2	7 6 1 2	Miss! fault+1
6	7 (6) 1 2	Hit! ✓
3	(3) 6 1 2	Miss! fault+1
6	3 (6) 1 2	Hit! ✓
4	3 (4) 1 2	Miss! fault+1
2	3 4 1 (2)	Hit! ✓
3	(3) 4 1 2	Hit! ✓
6	3 4 (6) 2	Miss! fault+1
3	(3) 4 6 2	Hit! ✓
1	3 4 6 (1)	Miss! fault+1
2	(2) 4 6 1	Miss! fault+1
6	2 4 (6) 1	Hit! ✓

No. of page faults => 9
No. of Hits => 6

Advantages
↳ simple

Disadvantages
↳ Efficiency is low
↳ Slow.

Optimal Page Replacement

DATE: / /
PAGE:

↳ Need knowledge of all the pages beforehand.

↳ The page which will be used the furthest in future is replaced.

Example

7, 6, 1, 2, 6, 3, 6, 4, 2, 3, 6, 3, 2, 6, 1, 7, 6, 1

Element	7	6	1	2	6	3	6	4	2	3	6	3	2	6	1	7	6	1
F4				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
F3			1	1	1	1	1	4	4	4	4	4	4	4	4	7	7	7
F2		6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
F1	7	7	7	7	7	3	3	3	3	3	3	3	3	3	1	1	1	1
Hit/Miss	X	X	X	X	✓	X	✓	X	✓	✓	✓	✓	✓	✓	X	X	✓	✓

↳ 7 is needed after 1, 6, 2
Hence 7 is replaced!
[First we see that after this element 6 comes first
then 2
then 1
and then 7]

↳ Now 2, 4, 3 all are not need so replace any

No. of Hits => 10
No. of Miss => 8

Advantages
↳ Easy to implement
↳ Data structure used are light

Disadvantages
↳ Need to know about future requests
↳ Error handling is tough.

Least Recently Used (LRU)

DATE: / /
PAGE:

↳ The page which is least used ~~from the frames is replacement~~.

Example

7, 6, 1, 2, 6, 3, 6, 4, 2, 3, 6, 3, 2, 6, 1, 7, 6, 1

Element	7	6	1	2	6	3	6	4	2	3	6	3	2	6	1	7	6	1
F4				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
F3			1	1	1	1	1	4	4	4	4	4	4	4	1	1	1	1
F2		6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
F1	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	7	7	7
Hit/Miss	X	X	X	X	✓	X	✓	X	✓	✓	✓	✓	✓	✓	X	X	✓	✓

↳ 7 is the least used before this no.

No. of Miss ⇒ 8
No. of Hits ⇒ 10

Advantages:
↳ 1. Easy to choose pages.

Disadvantages
↳ Extra Data Structure