Lecture: 32

Demand Paging

Paging: Divide memory into fixed-sized pages, map to frames.

Up to now, we assumed it was all in memory.

GO CLASSES

Virtual Memory

In Chapter 8, we discussed various memory-management strategies used in computer systems. All these strategies have the same goal: to keep many processes in memory simultaneously to allow multiprogramming. However, they tend to require that an entire process be in memory before it can execute.

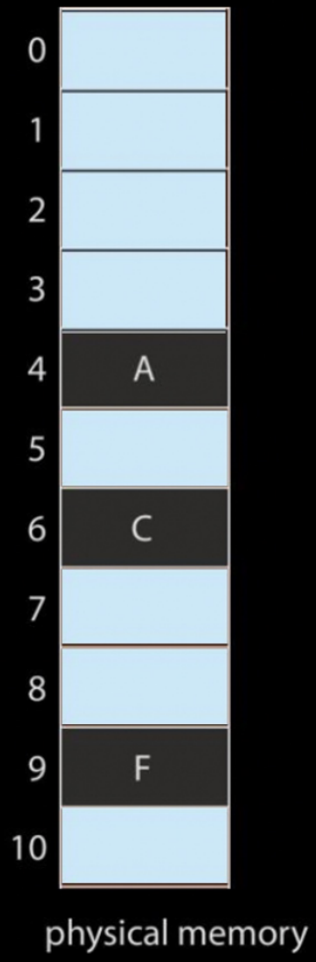
Virtual memory is a technique that allows the execution of processes that are not completely in memory. One major advantage of this scheme is that programs can be larger than physical memory. Further, virtual memory abstracts main memory into an extremely large, uniform array of storage, separating logical memory as viewed by the user from physical memory. This technique frees programmers from the concerns of memory-storage

Suppose we need 50 pages to run the job then we need 50 available frames otherwise we can't run job.

Suppose we need 50 pages to run the job then we need 50 available frames otherwise we can't run job.

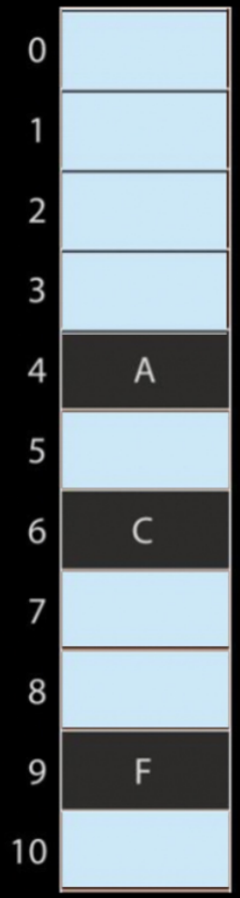
Demand paging says that even if we can load only one page in the main memory then also job can be executed.

Since we are not putting all pages in main memory hence some of the pages will also reside in secondary storage.

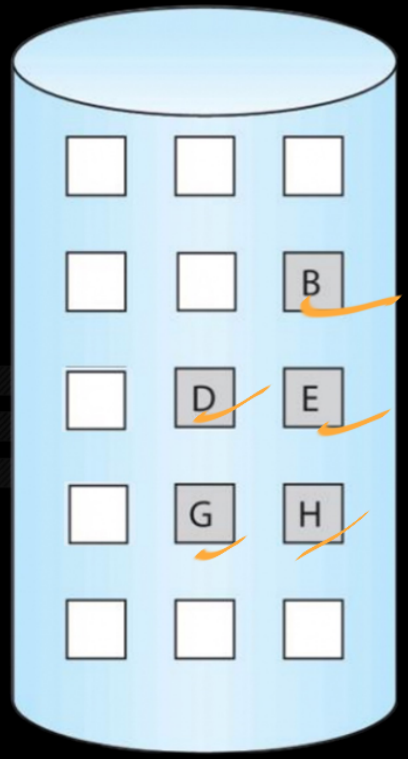


Where are the other pages ?

logical memory / virtual memory



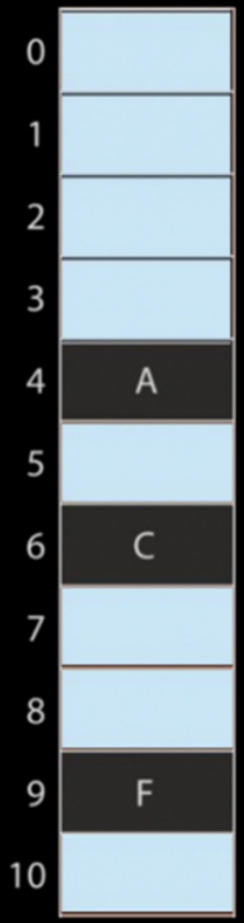
physical memory



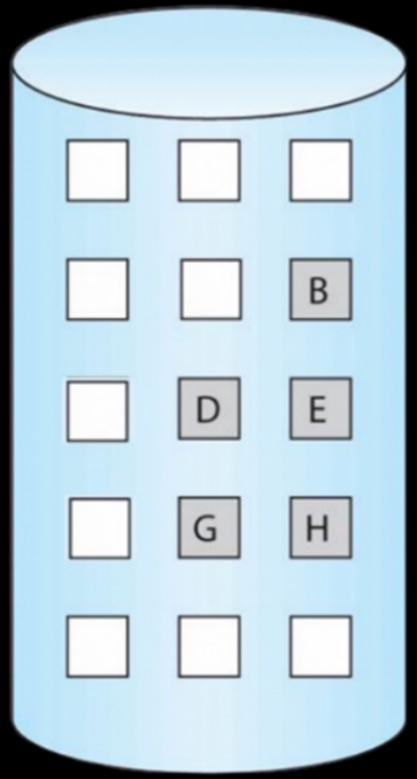
Operating Systems

logical memory

page table



physical memory



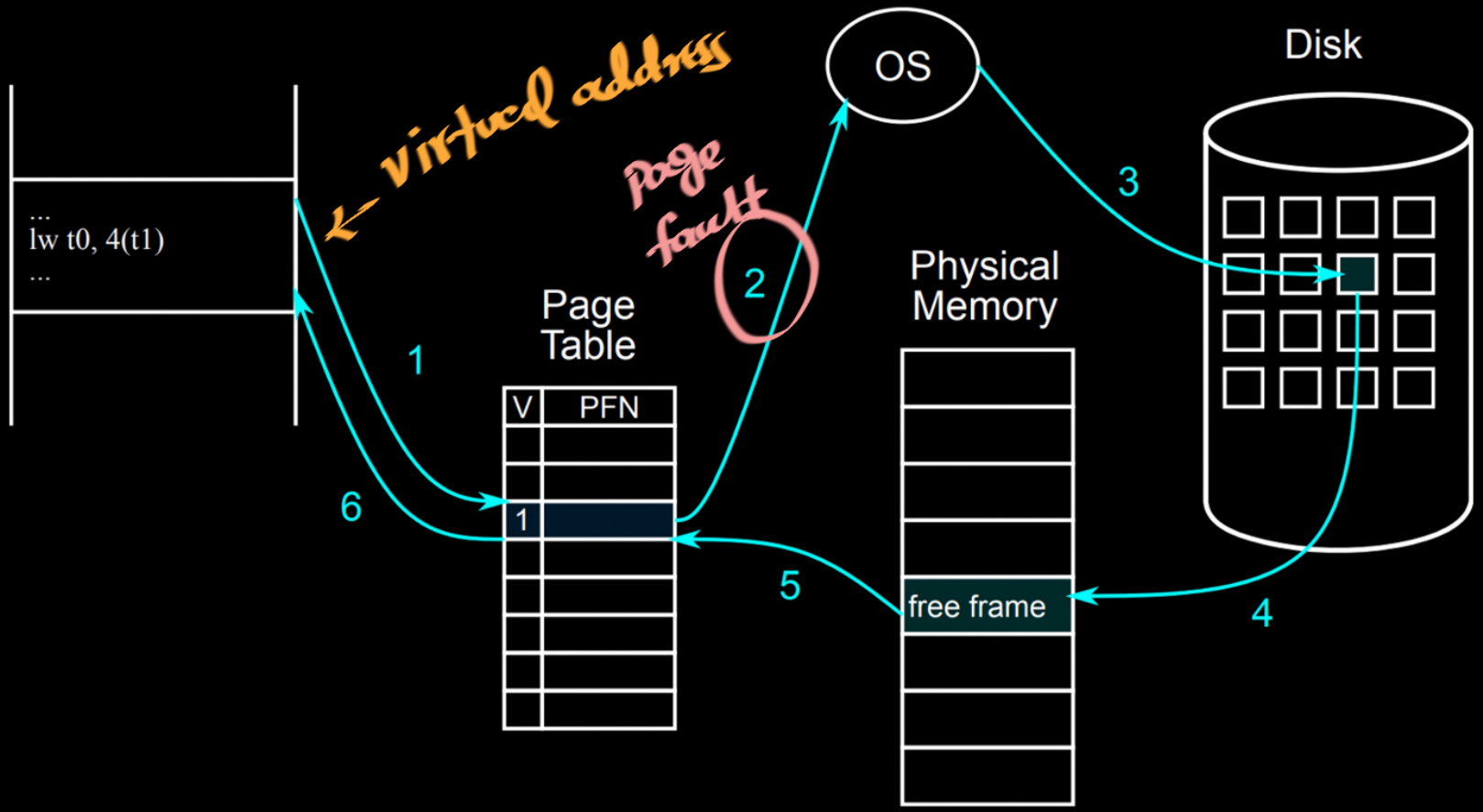
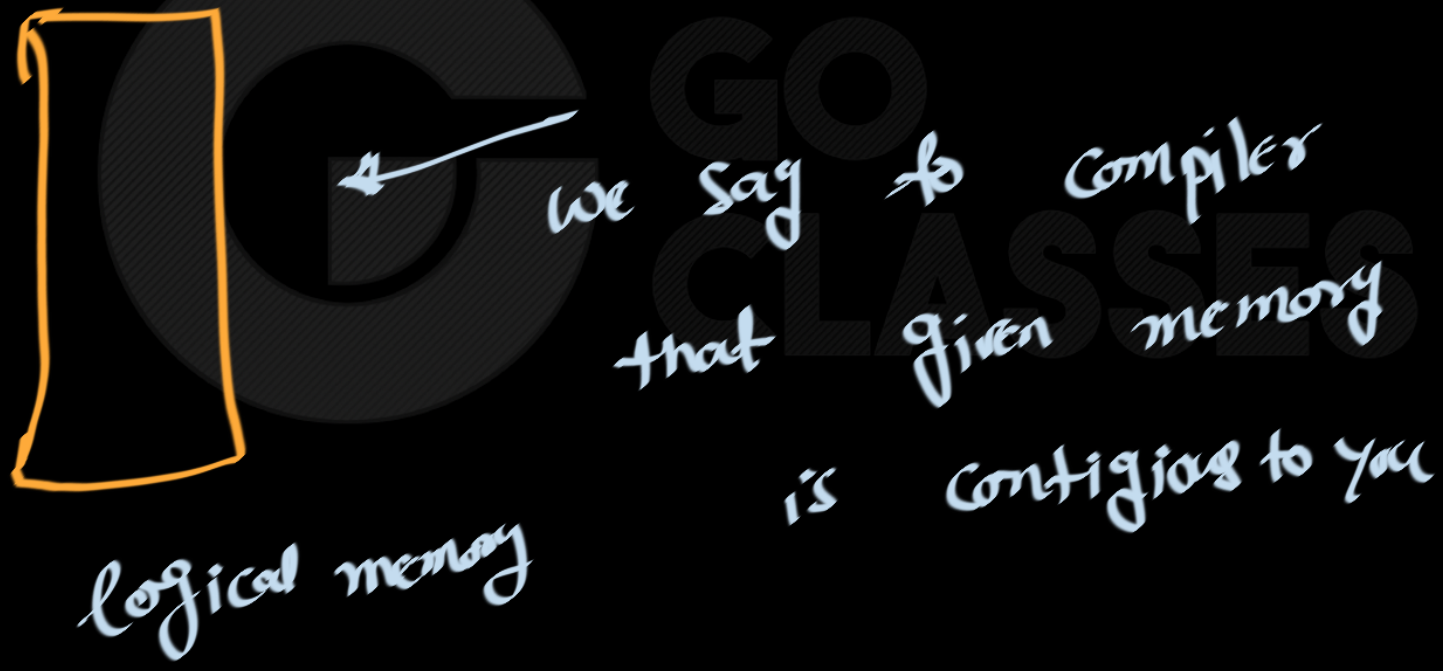


Figure: A typical sequence of execution when page is not in MM

"Demand Paging allow us to store pages to disk."

[It means we can run processes larger than main memory: This is idea of virtual memory]

Logical Memory and Virtual memory



Logical Memory and Virtual memory

Logical memory is an idea that says a program "logically" uses addresses from 0 to Max.

Relocatable code

Virtual memory just says that whatever it is logically using can be even bigger than physical memory.

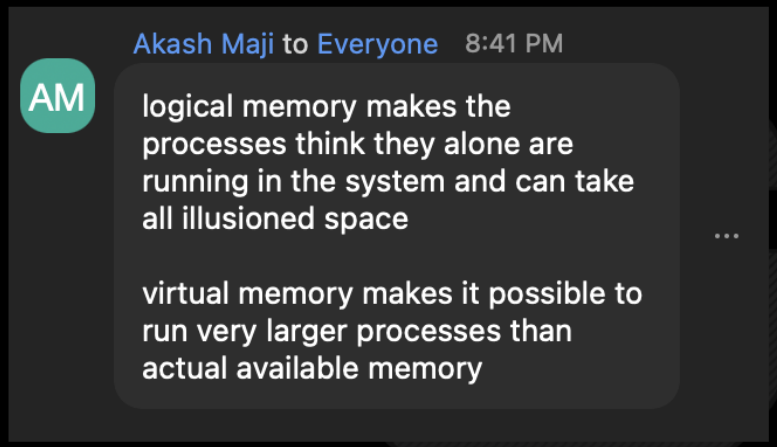
Logical Memory and Virtual memory

Logical memory is an idea that says a program "logically" uses addresses from 0 to Max.

Relocatable code

Virtual memory just says that whatever it is logically using can be even bigger than physical memory.

How to even do it ? – using Demand paging or Demand Segmentation.


Akash Maji to Everyone 8:41 PM
AM
logical memory makes the processes think they alone are running in the system and can take all illusioned space
virtual memory makes it possible to run very larger processes than actual available memory

12. Virtual Memory

(A) is an extremely large main memory

(B) is an extremely large secondary memory

(C) allows execution of processes that may not be completely in memory

(D) is a type of memory used in supercomputers

https://www.cse.iitb.ac.in/~krithi/courses/684/osquiz0.pdf

12. Virtual Memory

(A) is an extremely large main memory

(B) is an extremely large secondary memory

(C) allows execution of processes that may not be completely in memory

(D) is a type of memory used in supercomputers

Answer: C

https://www.cse.iitb.ac.in/~krithi/courses/684/osquiz0.pdf

Virtual memory is a technique that allows the execution of processes that are not completely in memory. One major advantage of this scheme is that programs can be larger than physical memory. Further, virtual memory abstracts main memory into an extremely large, uniform array of storage. This technique frees programmers from the concerns of memory-storage limitations.

In this chapter, we discuss virtual memory in the form of demand paging and examine its complexity and cost.

Source: Galvin

Process / virtual memory

MM

What are we achieving by putting part of the program in MM?

→ We are able to run large process

→ Higher degree of multi programming

GO CLASSES

With virtual memory,

we are not ONLY able to run large Process, also we are able to run MANY Processes.

The ability to execute a program that is only partially in memory would confer many benefits:

- A program would no longer be constrained by the amount of physical memory that is available. Users would be able to write programs for an extremely large *virtual* address space, simplifying the programming task.
- Because each user program could take less physical memory, more programs could be run at the same time, with a corresponding increase in CPU utilization and throughput but with no increase in response time or turnaround time.

Source: Galvin

It is desirable to be able to execute a process whose logical address space is larger than the available physical address space. Virtual memory is a technique that enables us to map a large logical address space onto a smaller physical memory. Virtual memory allows us to run extremely large processes and to raise the degree of multiprogramming, increasing CPU utilization. Further, it frees application programmers from worrying about memory availability.

Virtual memory is commonly implemented by demand paging.

Source: Galvin

GO CLASSES

"VM is most amazing invention in Computer Science history"

– Computer Systems : Programmers perspective.

Separation means these is no relationship between size of logical memory and size of physical memory

Virtual memory involves the separation of logical memory as perceived by users from physical memory. This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available (Figure 9.1). Virtual memory makes the task of programming much easier, because the programmer no longer needs to worry about the amount of physical memory available; she can concentrate instead on the problem to be programmed.

Source: Galvin

Figure 9.1 is on Next slide.

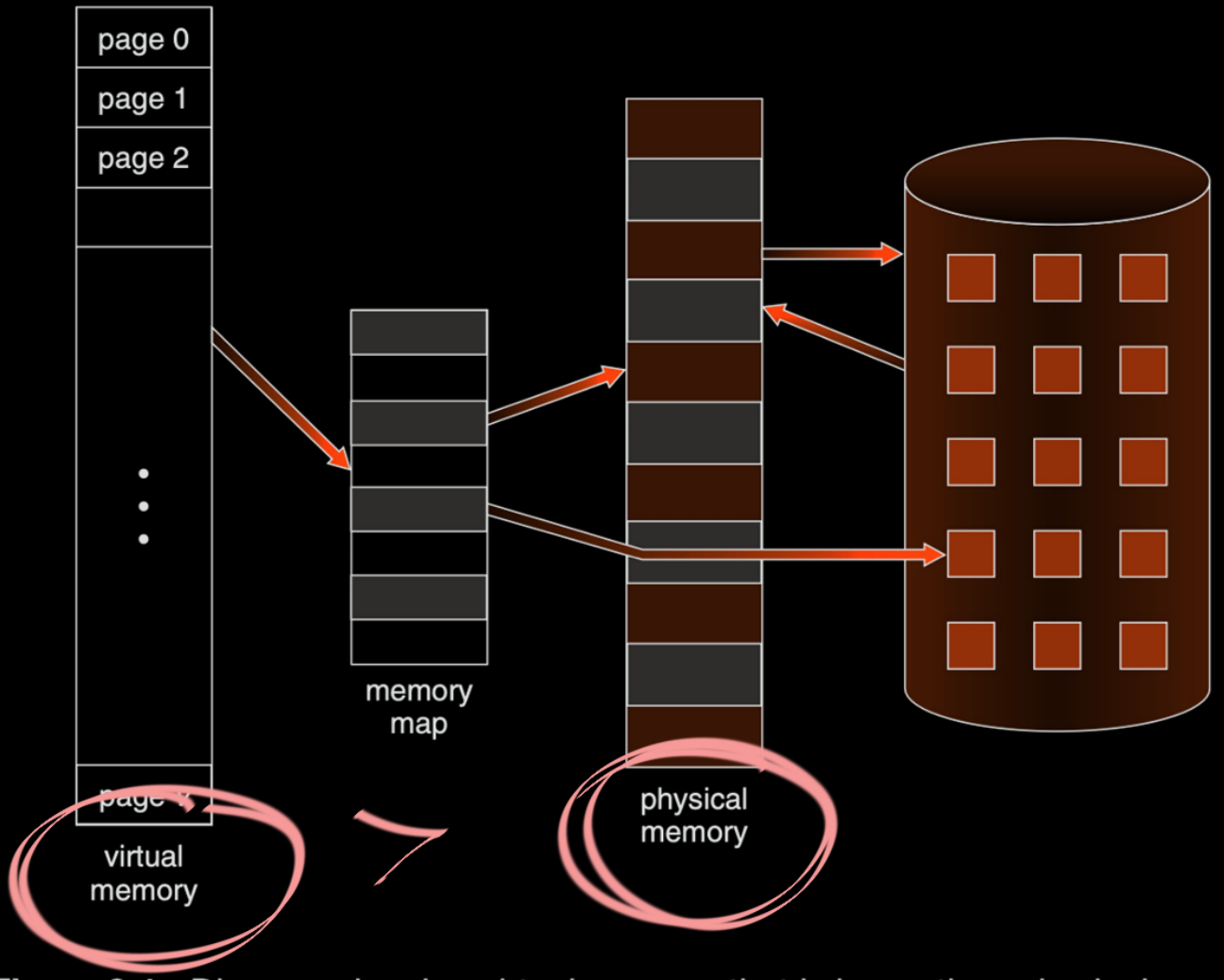


Figure 9.1 Diagram showing virtual memory that is larger than physical memory.

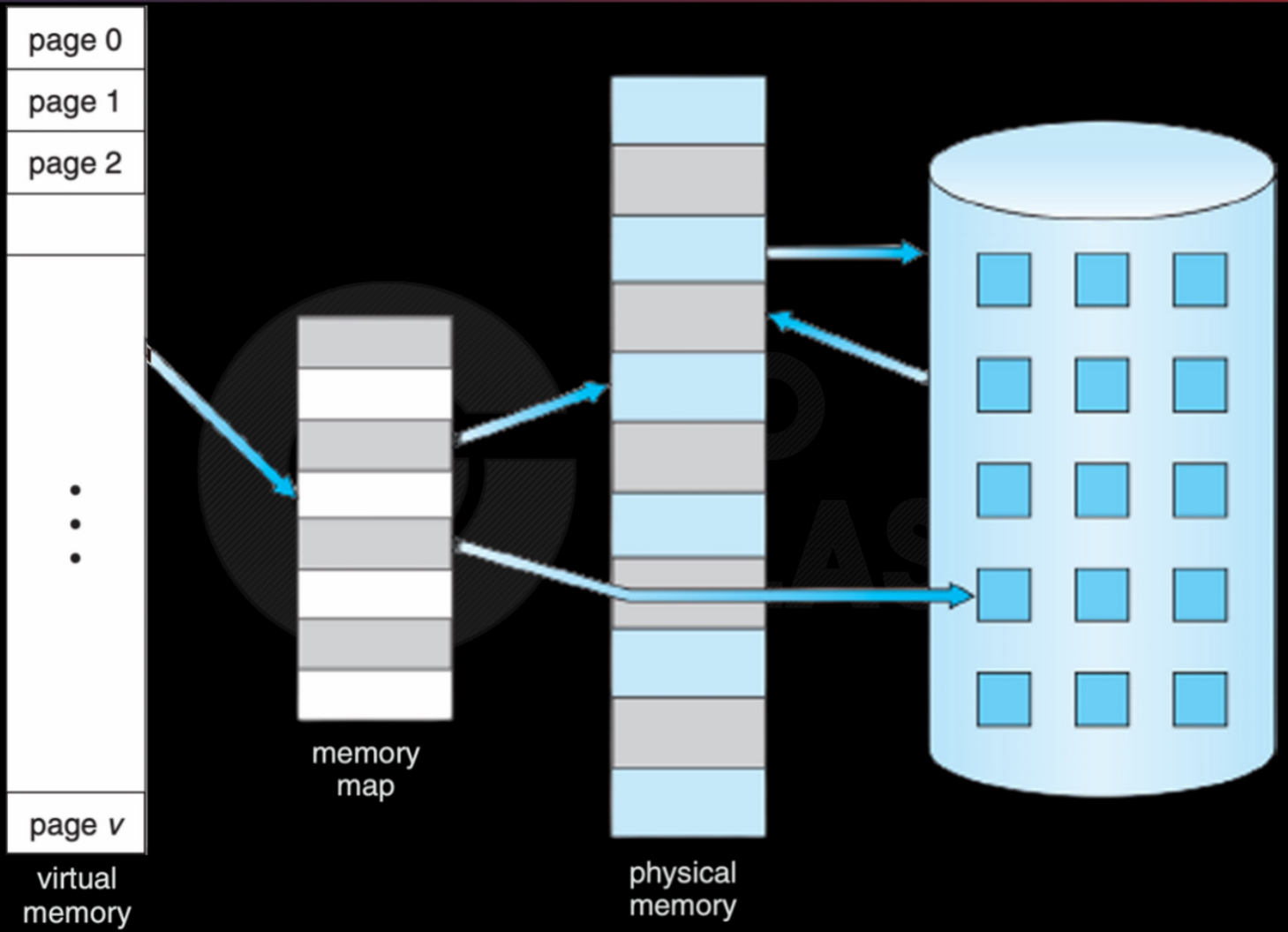


Figure 9.1 Diagram showing virtual memory that is larger than physical memory.

The **virtual address space** of a process refers to the logical (or virtual) view of how a process is stored in memory. Typically, this view is that a process begins at a certain logical address—say, address 0—and exists in contiguous memory, as shown in Figure 9.2.

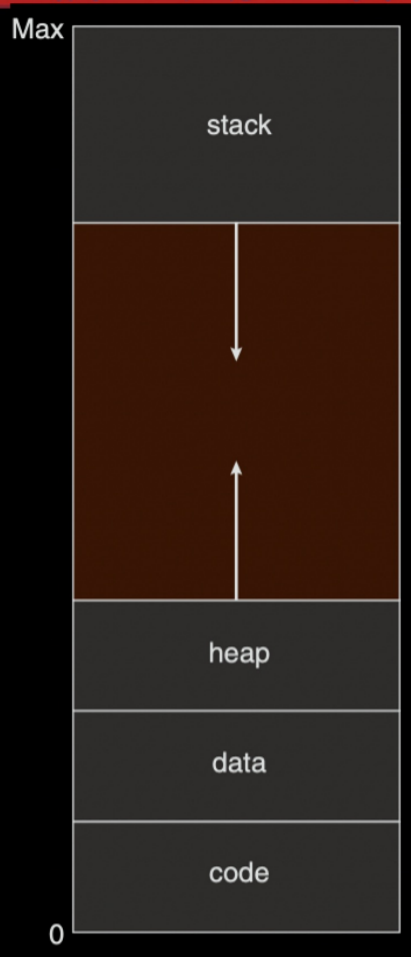
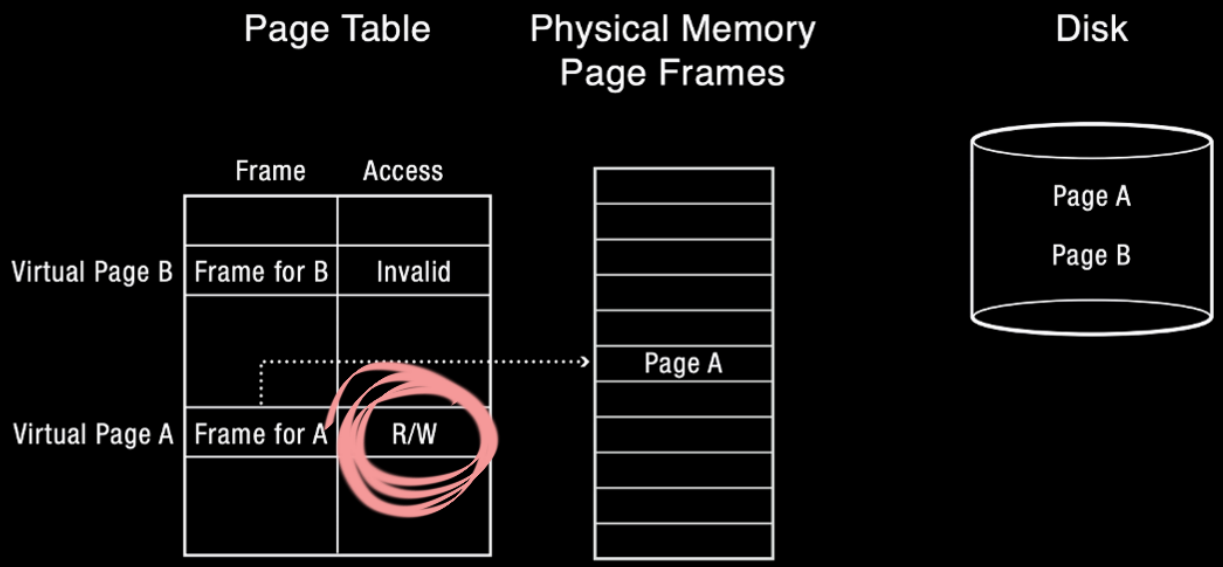


Figure 9.2 Virtual address space.

What if Page we are looking for is not in MM ?

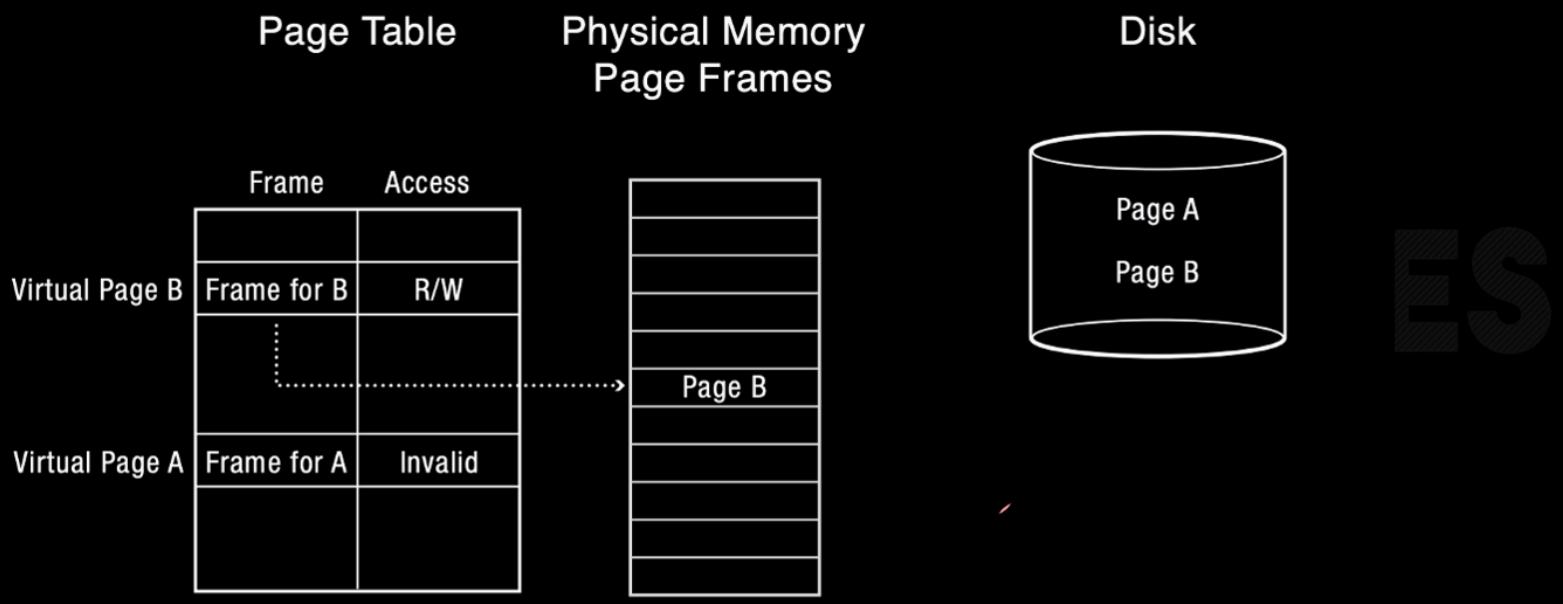
If page is not present in the memory then CPU generates an interrupt called page fault interrupt.

Demand Paging (Before)



https://courses.cs.washington.edu/courses/cse451/17au/caching2.pdf

Demand Paging (After)



Demand Paging says

Keep some pages in Physical mem.

Some pages in Secondary mem

Bring page from secondary memory to MM whenever needed.

GO CLASSES

Pure Demang Paging says

You start with 0 pages in MM, keep all pages in Secondary storage initially.

GATE CSE 1995 | Question: 2.7

52

The address sequence generated by tracing a particular program executing in a pure demand based paging system with 100 records per page with 1 free main memory frame is recorded as follows. What is the number of page faults?

$0100, 0200, 0430, 0499, 0510, 0530, 0560, 0120, 0220, 0240, 0260, 0320, 0370$

A. 13
B. 8
C. 7
D. 10

gate1995 | operating-system | page-replacement | normal

Page Size
S = 100 records

GATE CSE 1995 | Question: 2.7

52

The address sequence generated by tracing a particular program executing in a pure demand based paging system with 100 records per page with 1 free main memory frame is recorded as follows. What is the number of page faults?

$0100, 0200, 0430, 0499, 0510, 0530, 0560, 0120, 0220, 0240, 0260, 0320, 0370$

A. 13
B. 8
C. 7
D. 10

gate1995 | operating-system | page-replacement | normal

Page Size = 100 records

total 7 page faults

https://gateoverflow.in/2619/gate-cse-1995-question-2-7

60

Best answer

- 0100 — 1 page fault. Records 0100 — 0199 in memory
- 0200 — 2 page faults. Records 0200 — 0299 in memory
- 0430 — 3 page faults. Records 0400 — 0499 in memory
- 0499 — 3 page faults. Records 0400 — 0499 in memory
- 0510 — 4 page faults. Records 0500 — 0599 in memory
- 0530 — 4 page faults. Records 0500 — 0599 in memory
- 0560 — 4 page faults. Records 0500 — 0599 in memory
- 0120 — 5 page faults. Records 0100 — 0199 in memory
- 0220 — 6 page faults. Records 0200 — 0299 in memory
- 0240 — 6 page faults. Records 0200 — 0299 in memory
- 0260 — 6 page faults. Records 0200 — 0299 in memory
- 0320 — 7 page faults. Records 0300 — 0399 in memory
- 0370 — 7 page faults. Records 0300 — 0399 in memory

So, (C) - 7 page faults.

answered May 31, 2015 • edited Jun 29, 2018 by **kenzou**

comment Follow share this

Arjun

GATE IT 2007 | Question: 12

42

The address sequence generated by tracing a particular program executing in a pure demand paging system with 100 bytes per page is

$0100, 0200, 0430, 0499, 0510, 0530, 0560, 0120, 0220, 0240, 0260, 0320, 0410.$

Suppose that the memory can store only one page and if x is the address which causes a page fault then the bytes from addresses x to $x + 99$ are loaded on to the memory.
How many page faults will occur?

A. 0
B. 4
C. 7
D. 8

Page Size = 100 bytes

https://gateoverflow.in/3445/gate-it-2007-question-12

GATE IT 2007 | Question: 12

42

The address sequence generated by tracing a particular program executing in a pure demand paging system with 100 bytes per page is

$0100, 0200, 0430, 0499, 0510, 0530, 0560, 0120, 0220, 0240, 0260, 0320, 0410.$

Suppose that the memory can store only one page and if x is the address which causes a page fault then the bytes from addresses x to $x + 99$ are loaded on to the memory.
How many page faults will occur?

A. 0
B. 4
C. 7
D. 8

Page Size = 100 bytes

100 – 199

430 – 430+99 = 529

220 – 319

https://gateoverflow.in/3445/gate-it-2007-question-12

72

Best answer

0100 - page fault, addresses till 199 in memory

0200 - page fault, addresses till 299 in memory

0430 - page fault, addresses till 529 in memory

0499 - no page fault

0510 - no page fault

0530 - page fault, addresses till 629 in memory

0560 - no page fault

0120 - page fault, addresses till 219 in memory

0220 - page fault, addresses till 319 in memory

0240 - no page fault

0260 - no page fault

0320 - page fault, addresses till 419 in memory

0410 - no page fault

So, 7 is the answer- (C)

answered Nov 29, 2014 • edited Jun 29, 2018 by **kenzou**

comment Follow share this

Arjun

Performance of Demand Paging

Performance of Demand Paging

- Effective access time for demand-paged memory can be computed as:

  eacc = (1–p)(macc) + (p)(pfault)

  where:

  p = probability that page fault will occur

  macc = memory access time

  pfault = time needed to service page fault

No page fault

(1–p) macc + p × pfault

Question

p: Prob of page fault

- Memory access time = 200 nanoseconds
- Average page-fault service time = 8 milliseconds

- EAT =

Question

p: Prob of page fault

- Memory access time = 200 nanoseconds
- Average page-fault service time = 8 milliseconds
- EAT = $(1-p)\,200 + p \times 8 \times 10^6$

- Memory access time = 200 nanoseconds = 200×10^{-9} sec.
- Average page-fault service time = 8 milliseconds

- EAT = (1 – p) x 200 + p (8 milliseconds)
  = (1 – p x 200 + p x 8,000,000
  = 200 + p x 7,999,800
- If one access out of 1,000 causes a page fault, then EAT = 8.2 microseconds.

8.2×10^{-6} sec

$p = \frac{1}{1000}$

$\frac{2 \times 10^{-6}}{200 \times 10^{-9}} \quad 10^{-3}$

$\frac{2.8 \times 10^{-6}}{}$

- Memory access time = 200 nanoseconds
- Average page-fault service time = 8 milliseconds

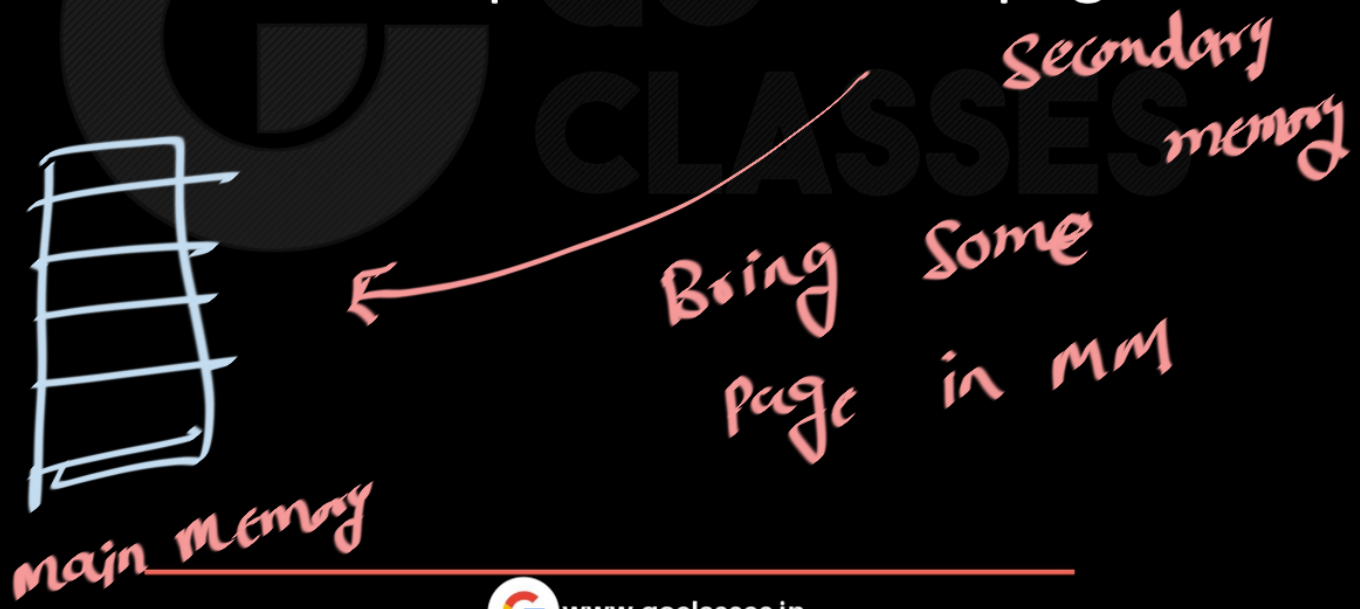
- EAT = (1 – p) x 200 + p (8 milliseconds)
  = (1 – p x 200 + p x 8,000,000
  = 200 + p x 7,999,800
- If one access out of 1,000 causes a page fault, then
  EAT = 8.2 microseconds.
  This is a slowdown by a factor of 40!!

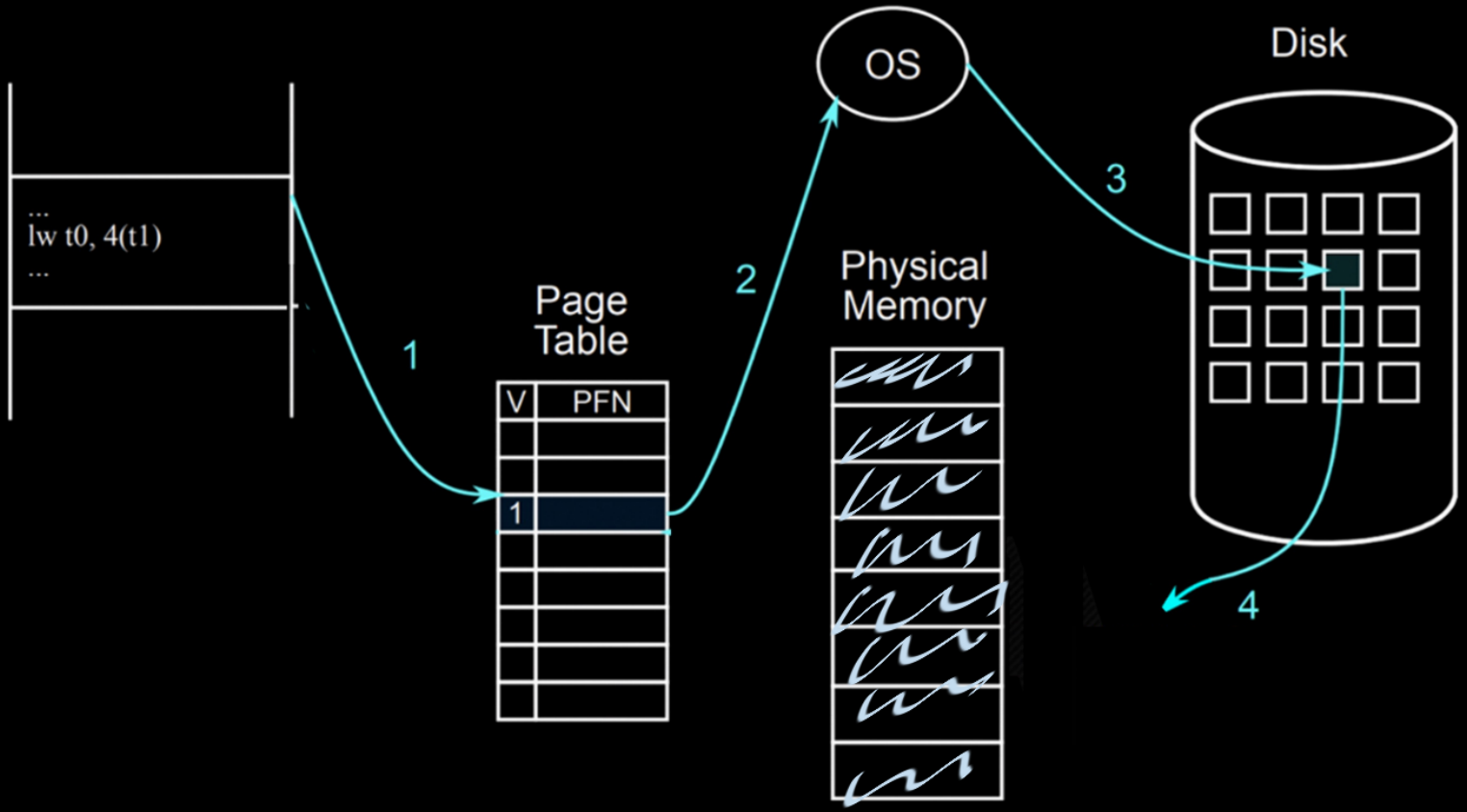
$p = \frac{1}{1000}$

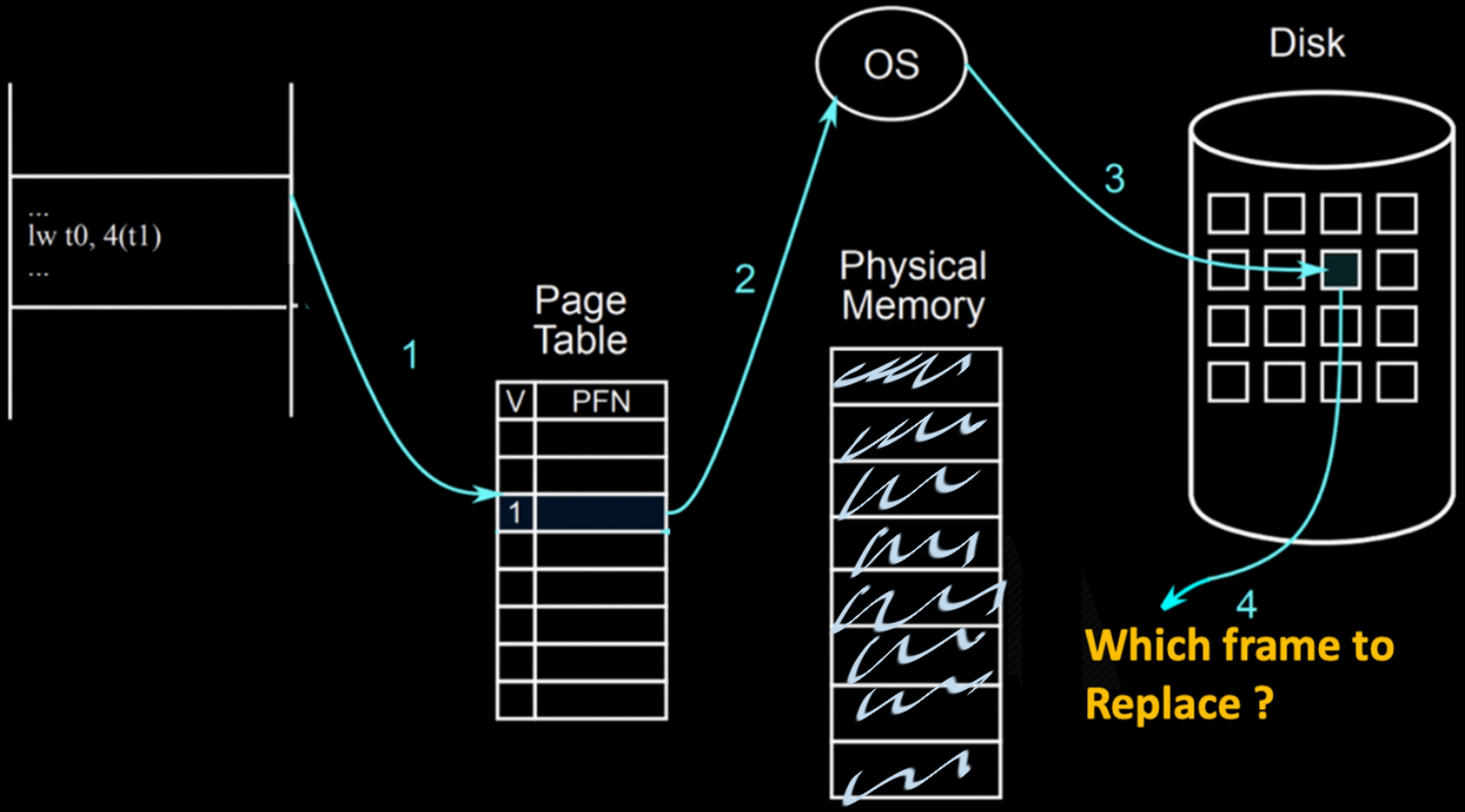
we want this prob to be as low as possible

After page fault, we load page to MM.

But, It may so happen that MM doesn't have any space so we need to replace one of the page.




...
lw t0, 4(t1)
...
1
Page Table
V
PFN
1
2
OS
Physical Memory
3
Disk
4


...
lw t0, 4(t1)
...
1
Page Table
V
PFN
1
2
OS
3
Disk
Physical Memory
4
Which frame to Replace ?

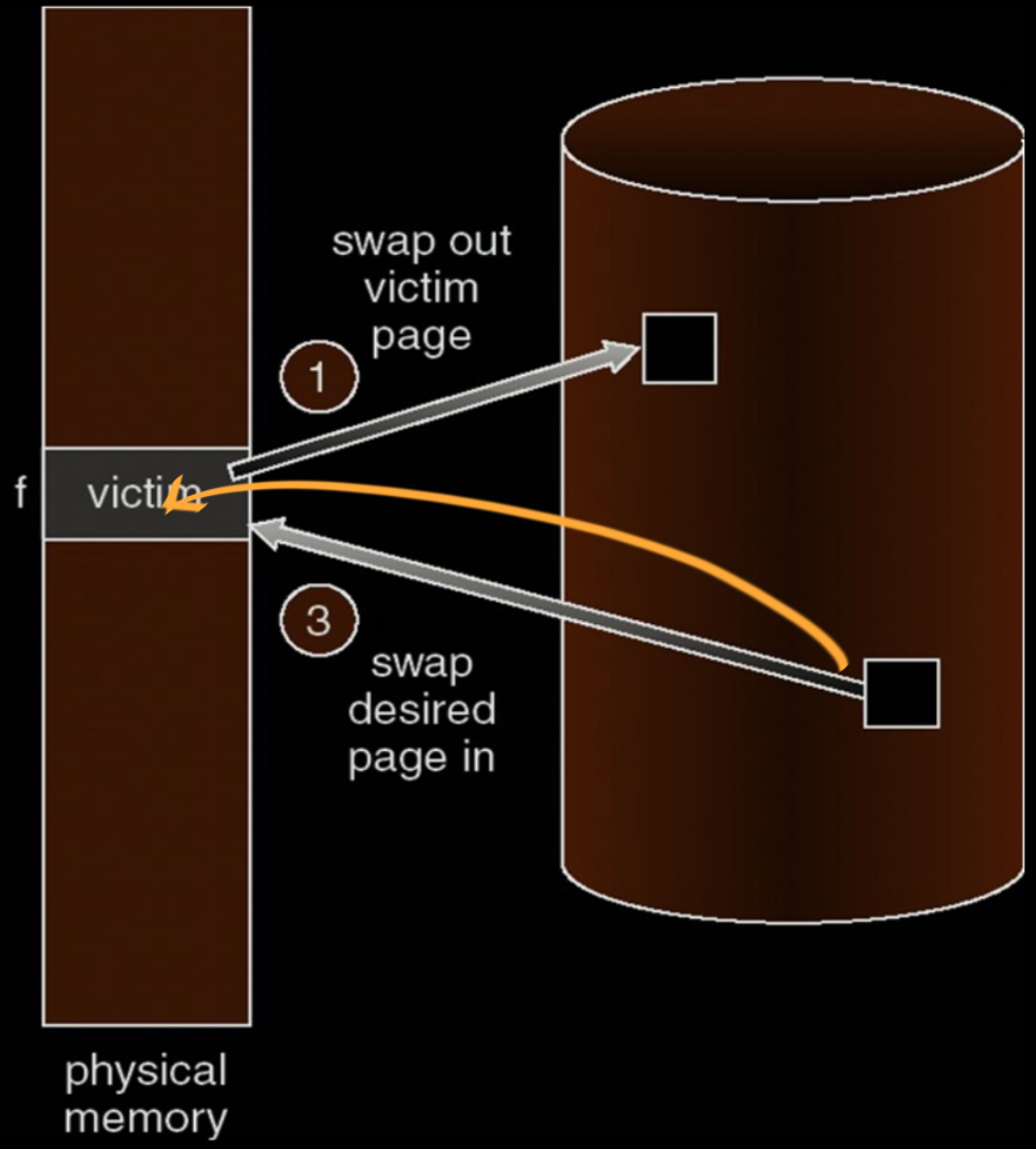
What Happens if There is no Free Frame?

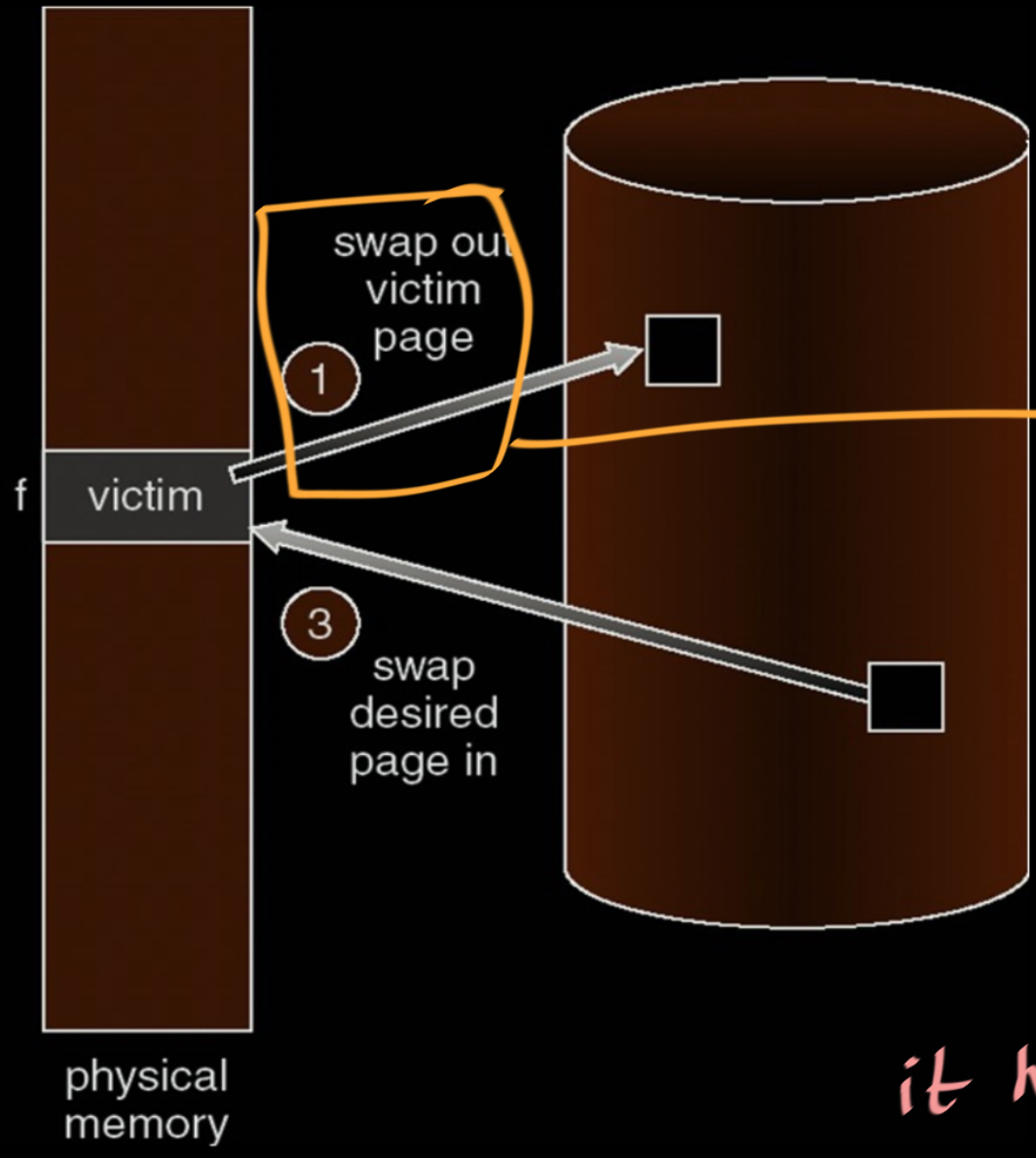
- Page replacement – find some page in memory, but not really in use, page it out
  - Algorithm – replace the page?
  - Performance – want an algorithm which will result in minimum number of page faults

Which “Used” Page Frame To Replace?

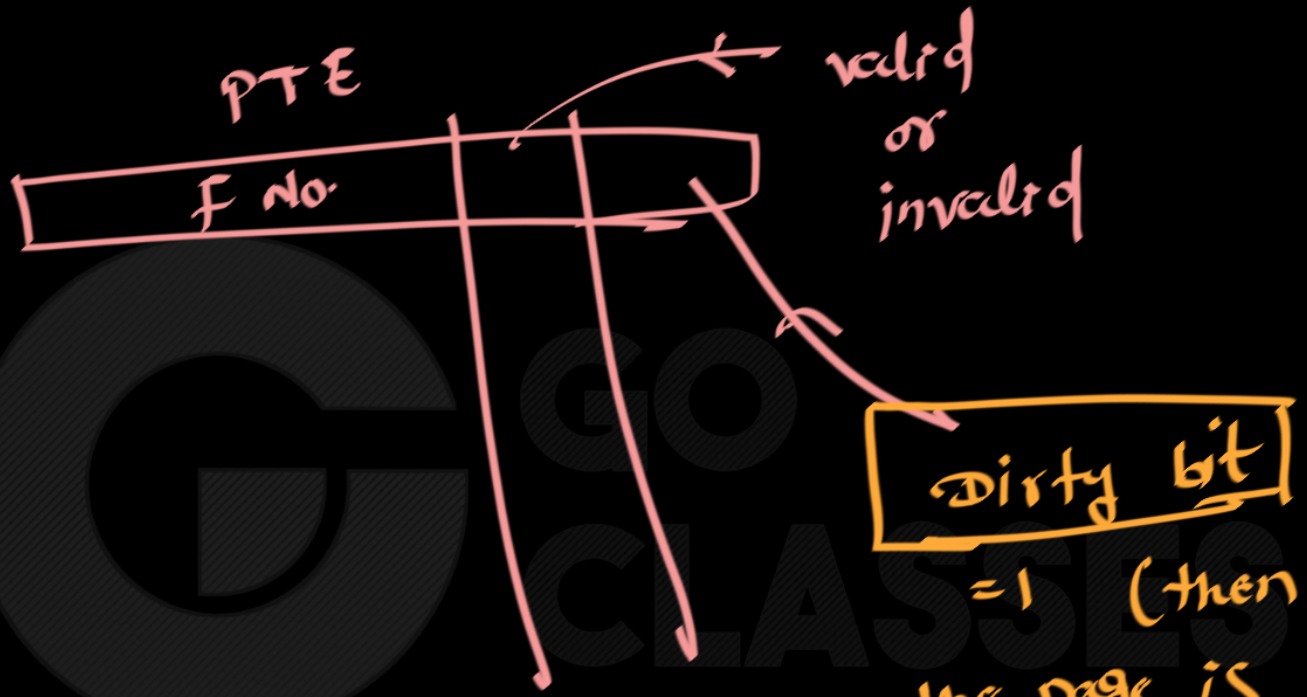
There are many algorithms…

Page Replacement Algorithms





we need to write the Victim page into Secondary memory only if it has been modified.


PTE
F no
valid
or
invalid
Dirty bit
=1 (then
the page is
modified and
should be written back
to hard disk)

VM Page Replacement

- Things are not always available when you want them
  - It is possible that no unused page frame is available
  - VM needs to do page replacement

we need to bring that page in MM

https://www.cs.princeton.edu/courses/archive/fall10/cos318/lectures/VMPaging.pdf

GO CLASSES

- On a page fault
  - If there is an unused frame, get it
  - **If no unused page frame available,**
    - **Find a used page frame**
    - **If it has been modified, write it to disk**
    - **Invalidate its current PTE and TLB entry**

Page Replacement

valid bit = i

VM Page Replacement

- Things are not always available when you want them
  - It is possible that no unused page frame is available
  - VM needs to do page replacement
- On a page fault
  - If there is an unused frame, get it
  - **If no unused page frame available,**
    - **Find a used page frame**
    - **If it has been modified, write it to disk**
    - **Invalidate its current PTE and TLB entry**

    Page Replacement
  - Load the new page from disk
  - Update the faulting PTE and remove its TLB entry
  - Restart the faulting instruction
- General data structures
  - A list of unused page frames
  - A table to map page frames to PID and virtual pages, why?

https://www.cs.princeton.edu/courses/archive/fall10/cos318/lectures/VMPaging.pdf

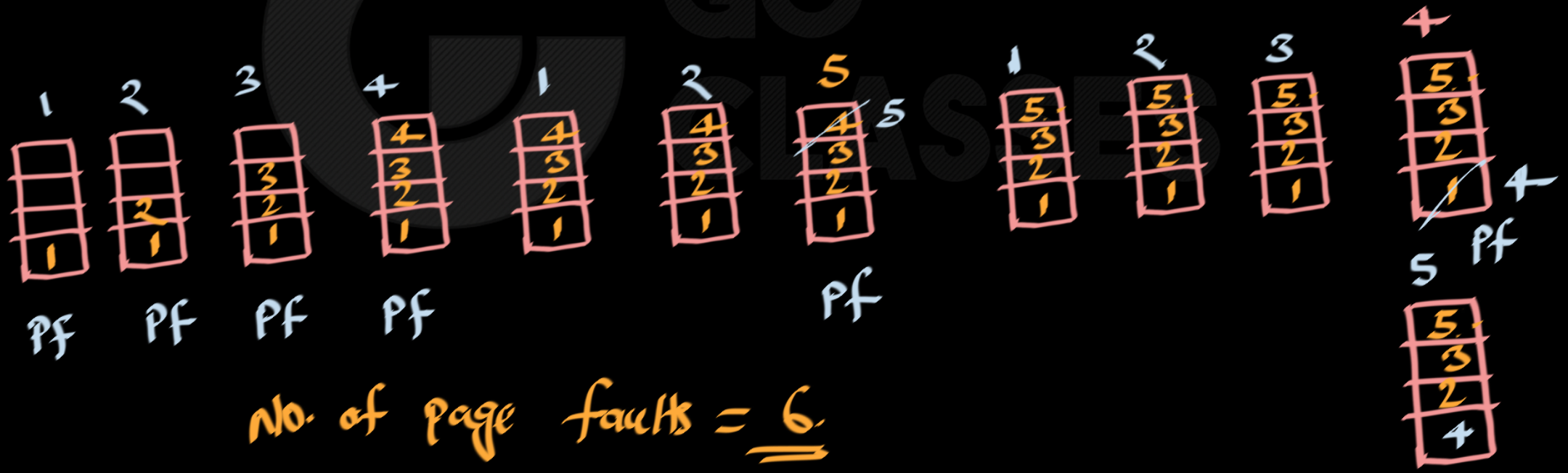
Optimal Algorithm

↳ Replace the page which we will not be using in future

Optimal Algorithm

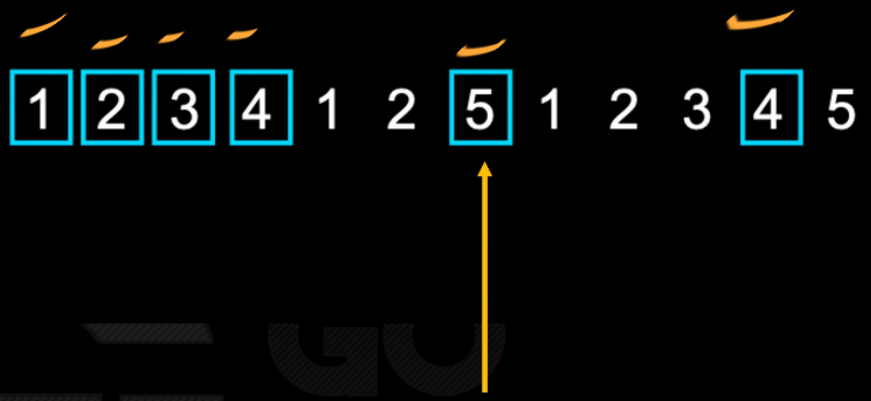
- Replace the page that won't be used for the longest time
- Can not be used in practice because we need to know all references in the future.

- Example
  - 4 page frames
  - Reference string: 1 2 3 4 1 2 5 1 2 3 4 5

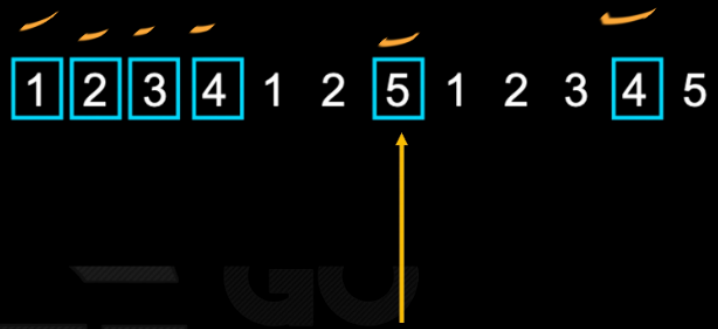


- Example
  - Reference string:
  - 4 page frames
  - 6 faults

Answer



- Example
  - Reference string:
  - 4 page frames
  - 6 faults



Answer

Here we are taking the decision based on future

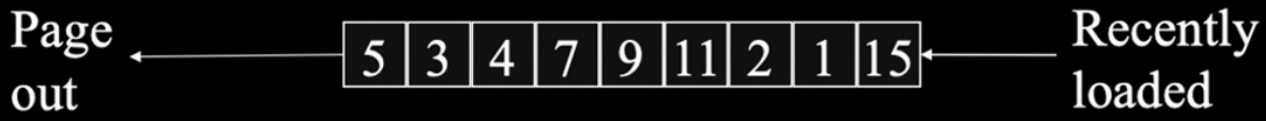
hence not Practical.

But it is useful for Comparision Purpose.

Optimal

First-In-First-Out (FIFO)

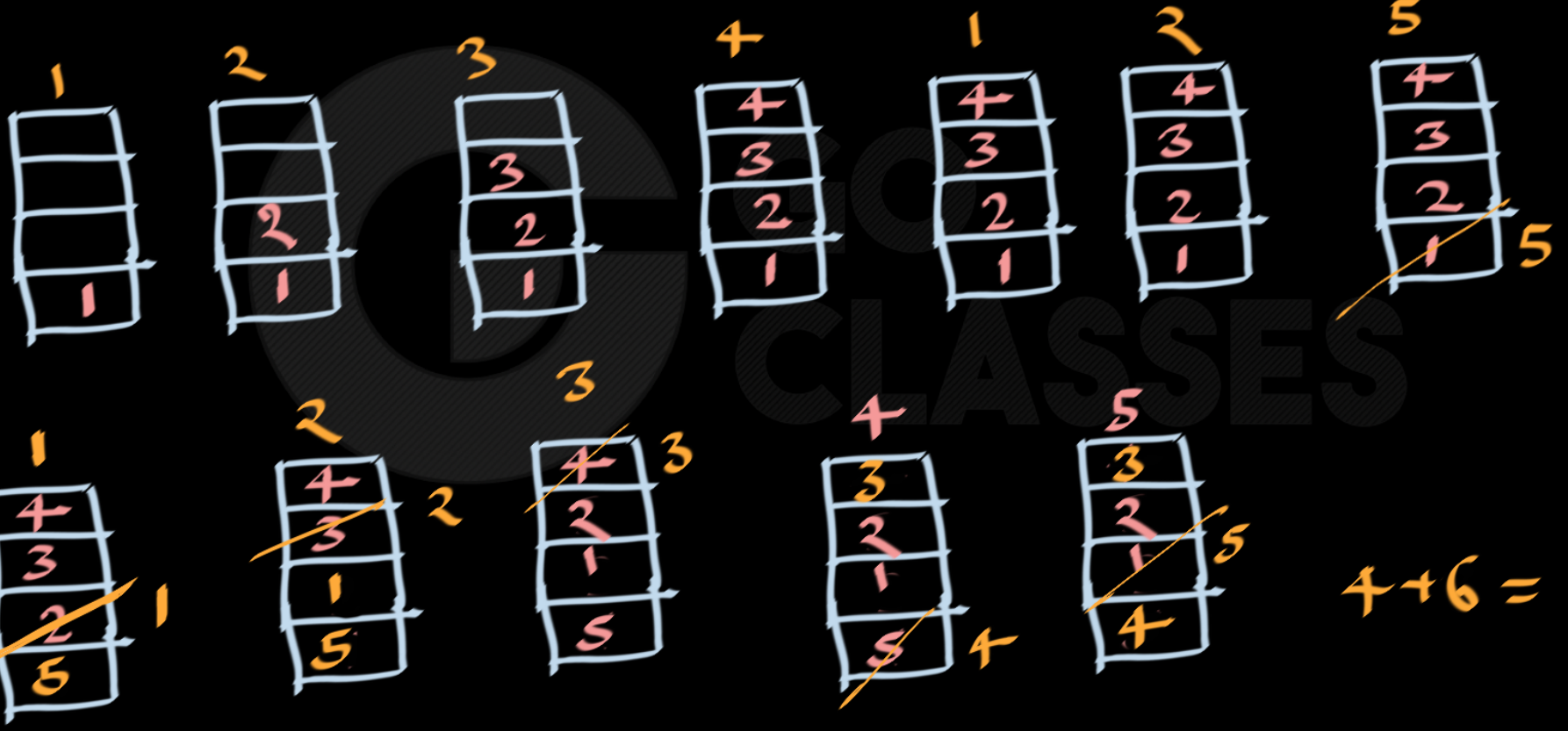
First-In-First-Out (FIFO)



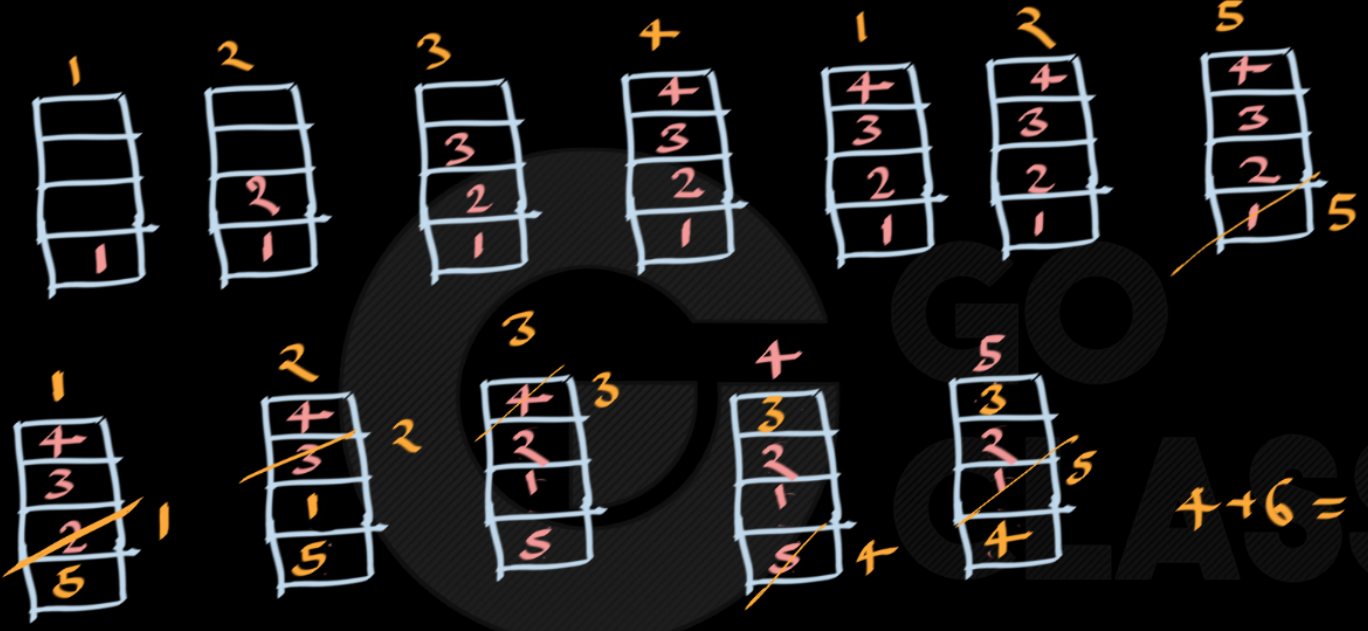
- Algorithm
  - Throw out the oldest page

- Example
  - 4 page frames
  - Reference string: 1 2 3 4 1 2 5 1 2 3 4 5 ___

- Example
  - 4 page frames
  - Reference string: 1 2 3 4 1 2 5 1 2 3 4 5



- Example
  - 4 page frames
  - Reference string: 1 2 3 4 1 2 5 1 2 3 4 5



4+6 = 10 page faults

using queue it is BETTER to Solve FIFO question.

- Example
  - 4 page frames
  - Reference string
  - 10 page faults

Fewer Frames → More Page Faults?

- Consider the following with 4 page frames
  - Algorithm: FIFO replacement
  - Reference string: 1 2 3 4 1 2 5 1 2 3 4 5
  - 10 page faults

what if we have lesser frames, say 3
→ will there be more page faults or lesser?

GO CLASSES

what if we have lesser frames, say 3

→ will there be more page faults or lesser?

Fewer Frames → More Page Faults?

- Consider the following with 4 page frames
  - Algorithm: FIFO replacement
  - Reference string: 1 2 3 4 1 2 5 1 2 3 4 5
  - 10 page faults
- Same string with 3 page frames
  - Algorithm: FIFO replacement
  - Reference string: 1 2 3 4 1 2 5 1 2 3 4 5
  - **9 page faults!**
- This is so called "Belady's anomaly" (Belady, Nelson, Shedler 1969)

Belady's Anomaly

If you have more page frames (i.e., more memory)...
You will have fewer page faults, right???

Belady's Anomaly

If you have more page frames (i.e., more memory)...
You will have fewer page faults, right???

GO CLASSES

1 2 3 4 1 2 5 1 2 3 4 5

Case 1:

3 frames available

Case 2:

4 frames available

$memory_1 \subseteq memory_2$

Case 2:

$memory_2$

1 2 3 4 1 2 5 1 2 3 4 5

Case 1:

3 frames available

Case 2:

4 frames available

Case 3:

1 2 3 4 1 2 5 1 2 3 4 5

Case 1:

3 frames available

Case 2:

4 frames available

Case 2:

GO CLASSES

1 2 3 4 1 2 5 1 2 3 4 5

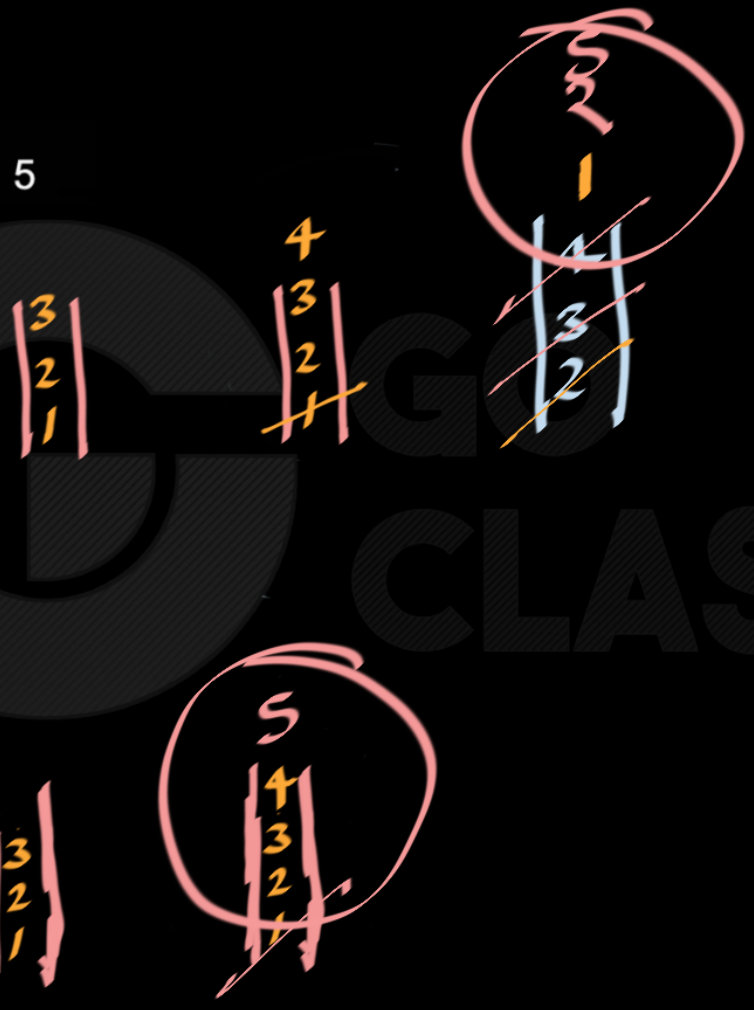
Case 1:

3 frames available

Case 2:

4 frames available

Case 2:



Consider FIFO page replacement
Look at this reference string
012301401234

Case 1:
3 frames available

Case 2:
4 frames available

All pages frames initially empty

	0	1	2	3	0	1	4	0	1	2	3	4
Youngest page	0	1	2	3	0	1	4	4	4	2	3	3
		0	1	2	3	0	1	1	1	4	2	2
Oldest page			0	1	2	3	0	0	0	1	4	4
	P	P	P	P	P	P	P			P	P	

9 Page faults

FIFO with 3 page frames

	0	1	2	3	0	1	4	0	1	2	3	4
Youngest page	0	1	2	3	3	3	4	0	1	2	3	4
	0	1	2	2	2	2	3	4	0	1	2	3
Oldest page		0	1	1	1	1	2	3	4	0	1	2
			0	0	0	0	1	2	3	4	0	1
	P	P	P	P			P	P	P	P	P	P

10 Page faults

FIFO with 4 page frames

All pages frames initially empty

	0	1	2	3	0	1	4	0	1	2	3	4
Youngest page	0	1	2	3	0	1	4	4	4	2	3	3
		0	1	2	3	0	1	1	1	4	2	2
Oldest page			0	1	2	3	0	0	0	1	4	4
	P	P	P	P	P	P	P			P	P	

9 Page faults

FIFO with 3 page frames

memory$_1$

memory$_1$ is Not subset of memory$_2$

	0	1	2	3	0	1	4	0	1	2	3	4
Youngest page	0	1	2	3	3	3	4	0	1	2	3	4
		0	1	2	2	2	3	4	0	1	2	3
Oldest page			0	1	1	1	2	3	4	0	1	2
				0	0	0	1	2	3	4	0	1
	P	P	P	P			P	P	P	P	P	P

10 Page faults

memory$_2$

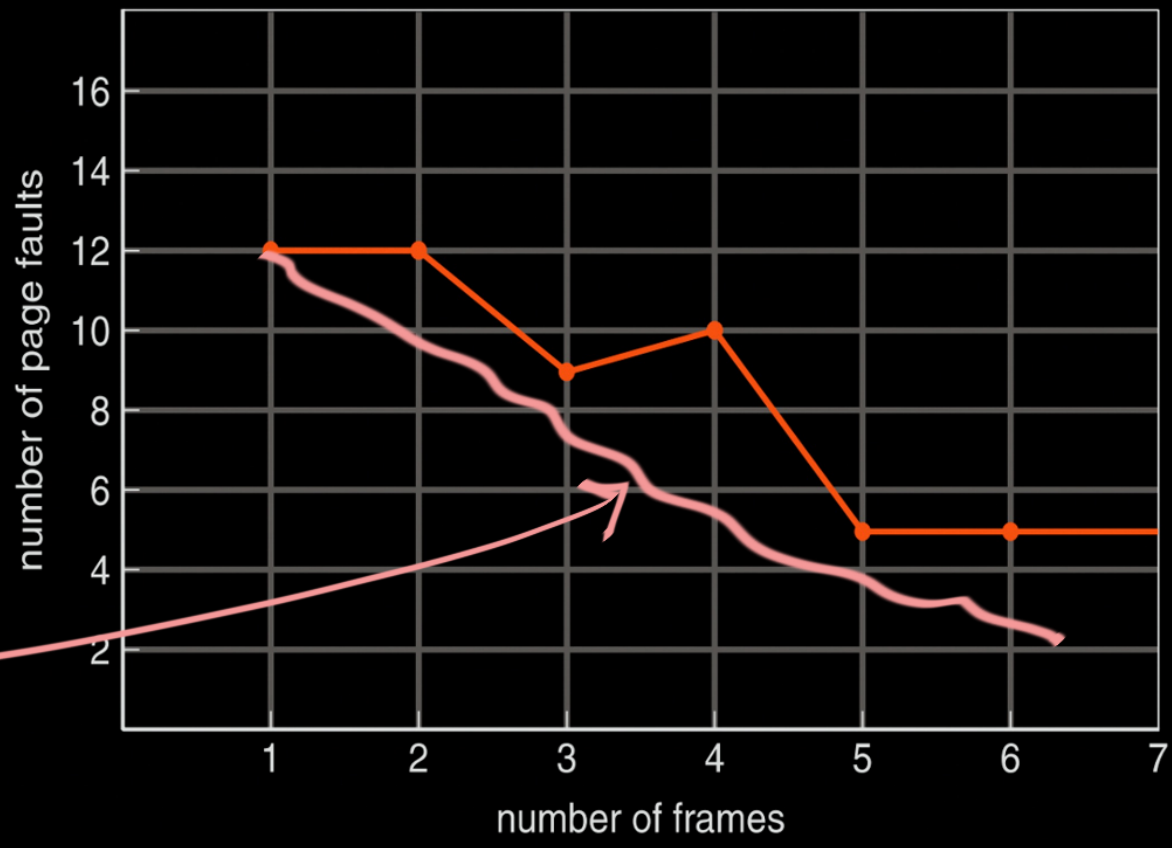
FIFO with 4 page frames

"Belady's anomaly"

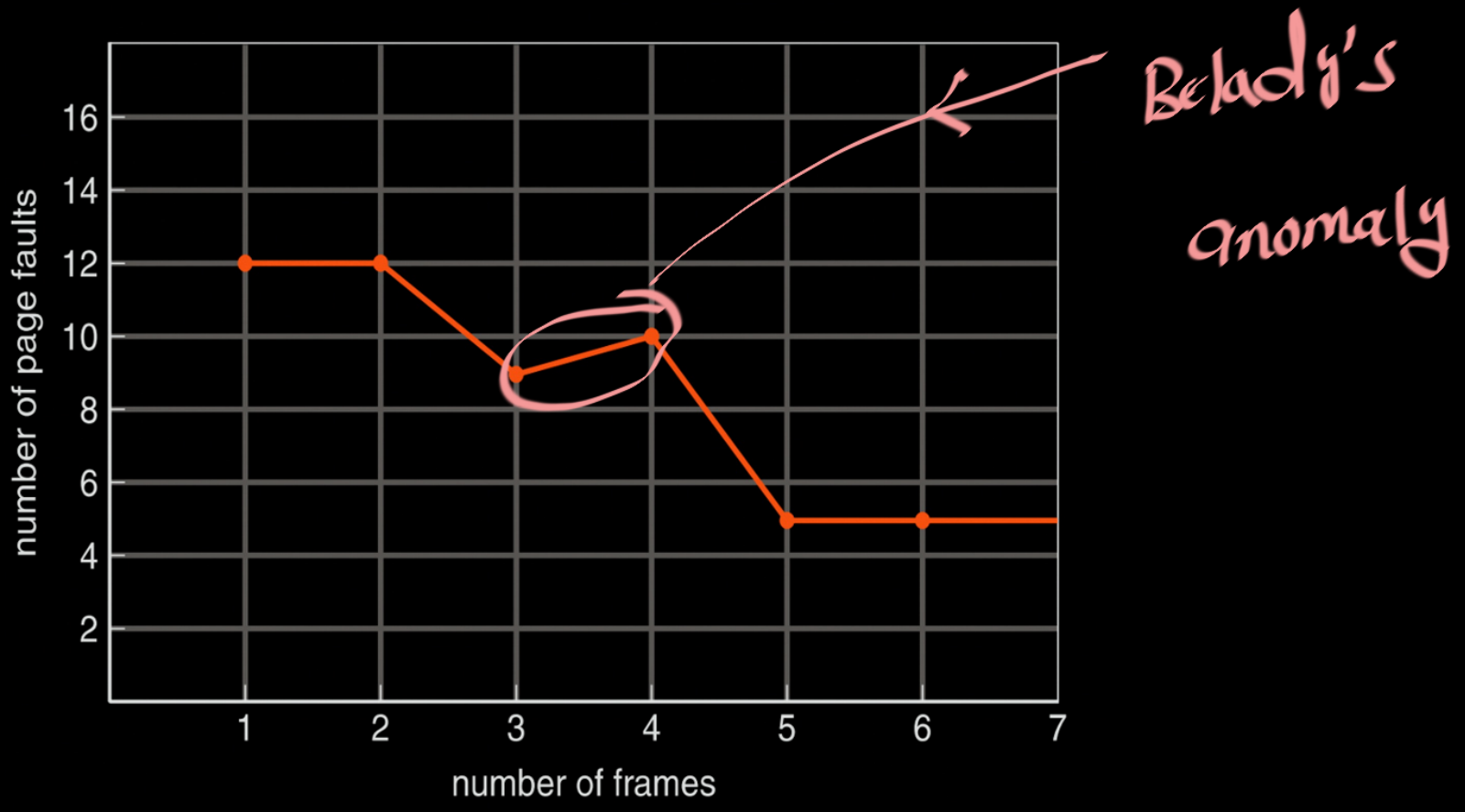
- Why??? **Increasing the number of page frames affects the order in which items are removed.**
- For certain memory access patterns, this can actually increase the page fault rate!

Subset Property get violated

FIFO Illustrating Belady's Anomaly



FIFO Illustrating Belady's Anomaly



GO CLASSES

optimal

fifo

Least Recently Used

Least Recently Used

- Algorithm
  - Replace page that hasn't been used for the longest time
- Example
  - 4 page frames
  - Reference string: 1 2 3 4 1 2 5 1 2 3 4 5

GO CLASSES

LRU

1 2 3 4 1 2 5 1 2 3 4 5

1,2,3,4

1

Rough work

1
4
3
2

GO CLASSES

LRU

1 2 3 4 1 2 5 1 2 3 4 5

1,2,3,4

1

2
1
4
3

Rough work

GO CLASSES

LRU

1 2 3 4 1 2 5 1 2 3 4 5

1,2,3,4

1

5
2
1
4
3

Rough work

GO CLASSES

LRU

1 2 3 4 1 2 5 1 2 3 4 5

1,2,3,4

1

1
5
2
4
3

Rough work

GO CLASSES

LRU

1 2 3 4 1 2 5 1 2 3 4 5

1,2,3,4

1

2
1
5
4
3

Rough work

GO CLASSES

LRU

1 2 3 4 1 2 5 1 2 3 4 5

1,2,3,4

1

3
2
1
5
4
3

Rough work

GO CLASSES

LRU

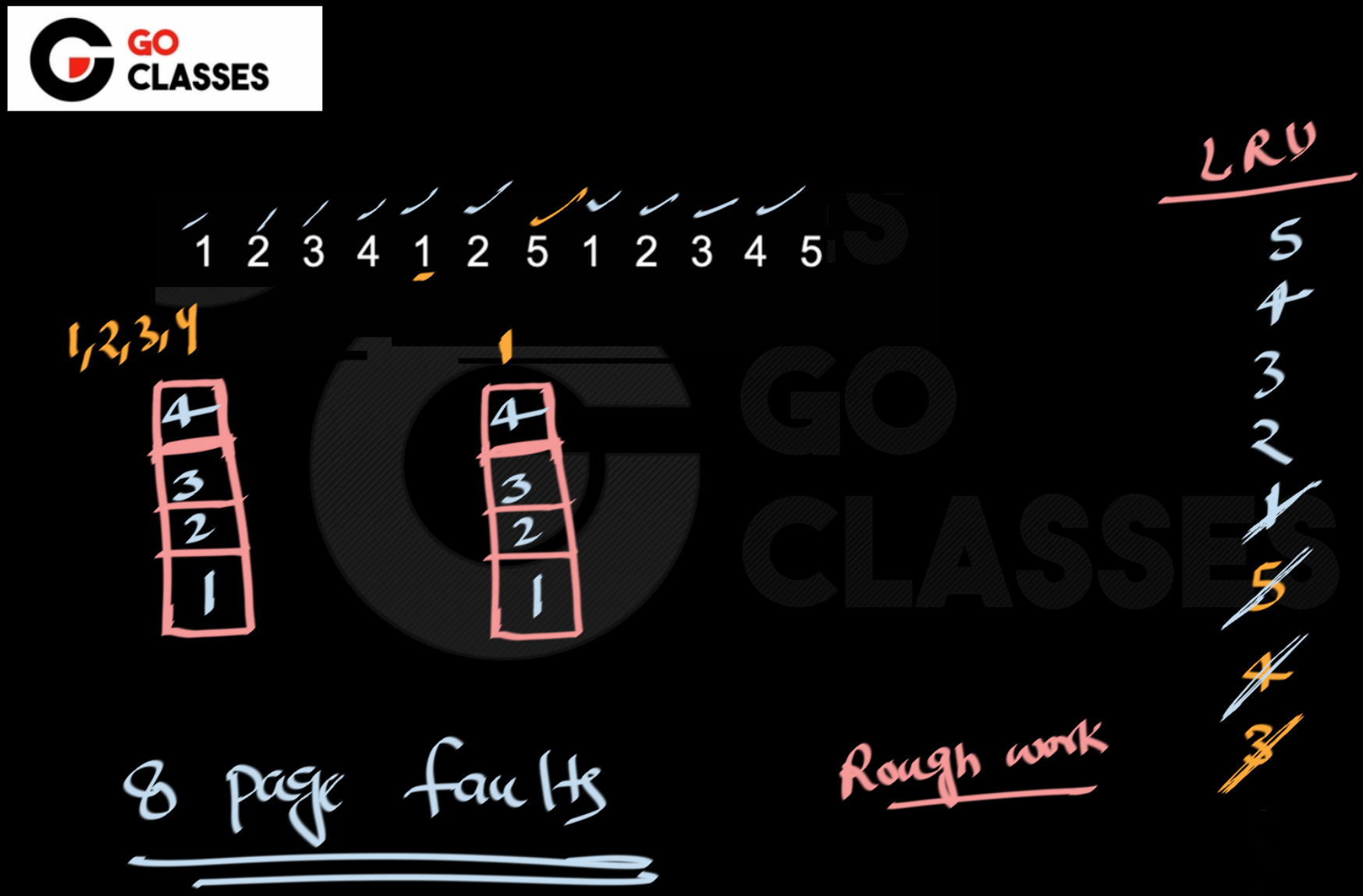
1 2 3 4 1 2 5 1 2 3 4 5

1,2,3,4

1

4
3
2
1
5
4
3

Rough work


1 2 3 4 1 2 5 1 2 3 4 5
1,2,3,4
1
4
3
2
1
4
3
2
1
8 page faults
Rough work
LRU
5
4
3
2
1
5
4
3

GO CLASSES

Which data structure to use to implement

→ use linked list in a way that if something old comes we remove from LL and put it on the end of LL

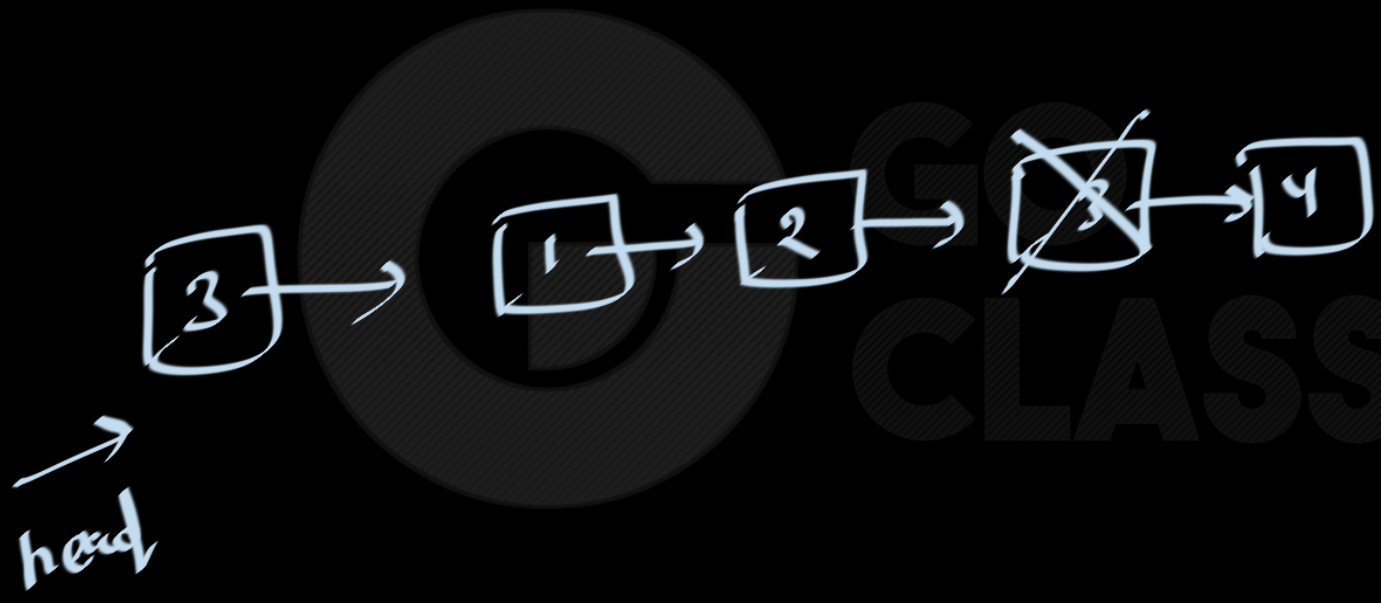
and we always remove from front of LL.

→ if something new comes, put it at end

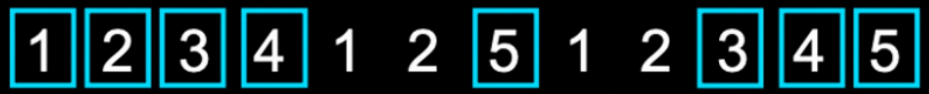
GO CLASSES

1 2 3 4

1 2 3 4 3



- Example
  - 4 page frames
  - Reference string:
  - 8 page faults



Least Recently used

Most Recently used

Variation

Counting Algorithms

- Keep a counter of the number of references that have been made to each page.
  - LFU (least frequently used) algorithm
    - replaces page with smallest count.
    - Rationale : frequently used page should have a large reference count.
      - Variation - shift bits right, exponentially decaying count.
  - MFU (most frequently used) algorithm
    - replaces page with highest count.
    - Based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

- **Least Frequently Used (LFU) Algorithm**: replaces page with smallest count
- **Most Frequently Used (MFU) Algorithm**: based on the argument that the page with the smallest count was probably just brought in and has yet to be used

GATE CSE 2010 | Question: 24

23

A system uses FIFO policy for system replacement. It has 4 page frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order and then accesses the same 100 pages but now in the reverse order. How many page faults will occur?

A. 196
B. 192
C. 197
D. 195

gatecse-2010 operating-system page-replacement normal

FIFO

https://gateoverflow.in/2203/gate-cse-2010-question-24

40

Best answer

Answer is (**A**).

When we access 100 distinct page in some order (for example $1, 2, 3 \ldots 100$) then total number of page faults $= 100$. At last, the 4 page frames will contain the pages $100, 99, 98$ and 97. When we reverse the string $(100, 99, 98, \ldots, 1)$ then first four page accesses will not cause the page fault because they are already present in page frames. But the remaining 96 page accesses will cause 96 page faults. So, total number of page faults $= 100 + 96 = 196$.

answered Oct 29, 2014 • edited Apr 24, 2021 by **gatecse**

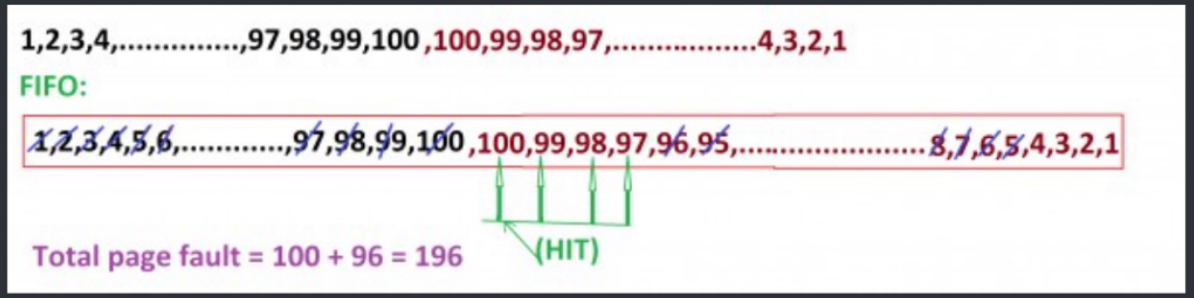
comment Follow share this

neha pawar

3 Comments

Lakshman Bhaiya commented Jan 7, 2019 edited Dec 20, 2019 by **Lakshman Bhaiya**

Page fault $= 100 + 96$ (reverse order) $= 196$



7

GATE CSE 1993 | Question: 21

14

The following page addresses, in the given sequence, were generated by a program:

$1\ 2\ 3\ 4\ 1\ 3\ 5\ 2\ 1\ 5\ 4\ 3\ 2\ 3$

This program is run on a demand paged virtual memory system, with main memory size equal to 4 pages. Indicate the page references for which page faults occur for the following page replacement algorithms.

A. LRU
B. FIFO

Assume that the main memory is initially empty.

gate1993 operating-system page-replacement normal descriptive

LRU
FIFO

https://gateoverflow.in/2318/gate-cse-1993-question-21

18

LRU : $1, 2, 3, 4, 5, 2, 4, 3, 2$

FIFO : $1, 2, 3, 4, 5, 1, 2, 3$

answered Apr 26, 2015 • edited Jun 29, 2018 by **kenzou**

comment Follow share this

Digvijay Pandey

Best answer

GATE CSE 1995 | Question: 1.8

19

Which of the following page replacement algorithms suffers from Belady's anamoly?

A. Optimal replacement

B. LRU

C. FIFO

D. Both (A) and (C)

gate1995 operating-system page-replacement normal

FIFO

https://gateoverflow.in/2595/gate-cse-1995-question-1-8

GATE CSE 2002 | Question: 1.23

19

The optimal page replacement algorithm will select the page that

A. Has not been used for the longest time in the past
B. Will not be used for the longest time in the future
C. Has been used least number of times
D. Has been used most number of times

gatecse-2002 operating-system page-replacement easy

Optimal

GATE CSE 1994 | Question: 1.13

33

A memory page containing a heavily used variable that was initialized very early and is in constant use is removed then

A. LRU page replacement algorithm is used

B. FIFO page replacement algorithm is used

C. LFU page replacement algorithm is used

D. None of the above

gate1994 operating-system page-replacement easy

LRU
LFU
FIFO

https://gateoverflow.in/2454/gate-cse-1994-question-1-13

Answer: B

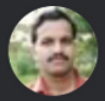
38

FIFO replaces a page which was brought into memory first will be removed first so since the variable was initialized very early. it is in the set of first in pages. so it will be removed answer: (B) if you use LRU - since it is used constantly it is a recently used item always. so cannot be removed. If you use LFU - the frequency of the page is more since it is in constant use. So cannot be replaced.

Best answer

answered Oct 9, 2014 • edited Apr 23, 2021 by **Lakshman Bhaiya**

comment Follow share this



Sankaranarayanan P.N

GATE CSE 2007 | Question: 82

15

A process has been allocated 3 page frames. Assume that none of the pages of the process are available in the memory initially. The process makes the following sequence of page references (reference string): $\mathbf{1, 2, 1, 3, 7, 4, 5, 6, 3, 1}$

If optimal page replacement policy is used, how many page faults occur for the above reference string?

A. 7
B. 8
C. 9
D. 10

Optimal

https://gateoverflow.in/1274/gate-cse-2007-question-82

23

Best answer

Optimal replacement policy means a page which is "farthest" in the future to be accessed will be replaced next.

Frame 0	1				
Frame 1	~~2~~	~~7~~	~~4~~	~~5~~	6
Frame 2	3				

3 initial page faults for pages $1, 2, 3$ and then for pages $7, 4, 5, 6 \implies 7$ page faults occur.

Answer is **(A)**.

GATE CSE 2012 | Question: 42

21

Consider the virtual page reference string

$$1, 2, 3, 2, 4, 1, 3, 2, 4, 1$$

on a demand paged virtual memory system running on a computer system that has main memory size of 3 page frames which are initially empty. Let LRU, FIFO and $\mathrm{OPTIMAL}$ denote the number of page faults under the corresponding page replacement policy. Then

A. $\mathrm{OPTIMAL} < \mathrm{LRU} < \mathrm{FIFO}$
B. $\mathrm{OPTIMAL} < \mathrm{FIFO} < \mathrm{LRU}$
C. $\mathrm{OPTIMAL} = \mathrm{LRU}$
D. $\mathrm{OPTIMAL} = \mathrm{FIFO}$

gatecse-2012 | operating-system | page-replacement | normal

Optimal
LRU
FIFO

https://gateoverflow.in/2150/gate-cse-2012-question-42

18

Page fault for LRU $= 9$, FIFO $= 6$, OPTIMAL $= 5$

Answer is (**B**).

GATE CSE 2014 Set 1 | Question: 33

23

Assume that there are 3 page frames which are initially empty. If the page reference string is $1, 2, 3, 4, 2, 1, 5, 3, 2, 4, 6$ the number of page faults using the optimal replacement policy is_____.

gatecse-2014-set1 | operating-system | page-replacement | numerical-answers

Optimal

https://gateoverflow.in/1805/gate-cse-2014-set-1-question-33

Correct Answer: 7.

7

In Optimal page replacement a page which will be farthest accessed in future will be replaced first.

Here, we have 3 page frames. Since, initially they are empty the first 3 distinct page references will cause page faults.

Best answer

After 3 distinct page accesses we have : Page Frames | 1 | 2 | 3 | Next Use Order | 2 | 1 | 3 |.

Based on the Next Use Order, the next replacement will be 3. Proceeding like this we get

Request	Page Frames	Next Use Order	
4	1 2 4	2 1 4	– Miss.
2	1 2 4	2 1 4	– Hit.
1	1 2 4	2 4 1	– Hit.
5	5 2 4	2 4 5	– Miss.
3	3 2 4	2 4 3	– Miss.
2	3 2 4	4 3 2	– Hit.
4	1 2 4	4 3 2	– Hit.
6	1 2 6	2 1 6	– Miss.

(When multiple pages are not going to be accessed again in future, replacing **any** of them is allowed in Optimal page replacement algorithm)

Now, counting the misses which includes the 3 initial ones we get number of page faults as $3 + 4 = 7$.

Correct Answer: 7.

answered Jul 14, 2019

comment Follow share this

Arjun

GATE CSE 2014 Set 2 | Question: 33

68

A computer has twenty physical page frames which contain pages numbered 101 through 120. Now a program accesses the pages numbered $1, 2, \ldots, 100$ in that order, and repeats the access sequence **THRICE**. Which one of the following page replacement policies experiences the same number of page faults as the optimal page replacement policy for this program?

A. Least-recently-used
B. First-in-first-out
C. Last-in-first-out
D. Most-recently-used

gatecse-2014-set2 | operating-system | page-replacement | ambiguous

LIFO
LRU
MRU

https://gateoverflow.in/1992/gate-cse-2014-set-2-question-33

76

Best answer

It will be **(D)** i.e Most-recently-used.

To be clear "repeats the access sequence THRICE" means totally the sequence of page numbers are accessed 4 times though this is not important for the answer here.

If we go optimal page replacement algorithm it replaces the page which will be least used in near future.

Now we have frame size 20 and reference string is

$$1, 2, \ldots, 100, 1, 2, \ldots, 100, 1, 2, \ldots, 100, 1, 2, \ldots, 100$$

First 20 accesses will cause page faults - the initial pages are no longer used and hence optimal page replacement replaces them first. Now, for page 21, according to reference string page 1 will be used again after 100 and similarly 2 will be used after 1 so, on and so the least likely to be used page in future is page 20. So, for 21^{st} reference page 20 will be replaced and then for 22^{nd} page reference, page 21 will be replaced and so on which is MOST RECENTLY USED page replacement policy.

PS: Even for Most Recently Used page replacement at first all empty (invalid) pages frames are replaced and then only most recently used ones are replaced.

GATE CSE 2014 Set 3 | Question: 20

LRU

17

A system uses 3 page frames for storing process pages in main memory. It uses the Least Recently Used (**LRU**) page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below?

$4, 7, 6, 1, 7, 6, 1, 2, 7, 2$

https://gateoverflow.in/2054/gate-cse-2014-set-3-question-20

Answer: 6

23

Best answer

Total page faults $= 6$.

4	7	6	1	7	6	1	2	7	2
		6 F	6	6	6	6	6	7 F	7
	7 F	7	7	7	7	7	2 F	2	2
4 F	4	4	1 F	1	1	1	1	1	1

$\Longrightarrow$ 6 faults

Another way of answering the same.

6	6	~~6~~ 7	7
7	~~7~~ 2	2	2
~~4~~ 1	1	1	1

$\Longrightarrow$ 3 faults + 3 initial access faults = 6 page faults

OR

~~6~~ 7
~~7~~ 2
~~4~~ 1

$\Longrightarrow$ 3 faults + 3 initial access faults = 6 page faults

GATE CSE 2015 Set 1 | Question: 47

24

Consider a main memory with five-page frames and the following sequence of page references: $3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3$. Which one of the following is true with respect to page replacement policies First In First Out (FIFO) and Least Recently Used (LRU)?

A. Both incur the same number of page faults
B. FIFO incurs 2 more page faults than LRU
C. LRU incurs 2 more page faults than FIFO
D. FIFO incurs 1 more page faults than LRU

gatecse-2015-set1 | operating-system | page-replacement | normal

FIFO
LRU

https://gateoverflow.in/8353/gate-cse-2015-set-1-question-47

32

Best answer

Requested Page references are $3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3$ and number of page frames is 5.

In FIFO Page replacement will take place in sequence in pattern First In first Out, as following

Request	3	8	2	3	9	1	6	3	8	9	3	6	2	1	3
Frame 5						1	1	1	1	1	1	1	1	1	1
Frame 4					9	9	9	9	9	9	9	9	2	2	2
Frame 3			2	2	2	2	2	2	8	8	8	8	8	8	8
Frame 2		8	8	8	8	8	8	3	3	3	3	3	3	3	3
Frame 1	3	3	3	3	3	3	6	6	6	6	6	6	6	6	6
Miss/hit	F	F	F	H	F	F	F	F	F	H	H	H	F	H	H

Number of Faults $= 9$. Number of Hits $= 6$

Using Least Recently Used (LRU) page replacement will be the page which is visited least recently (which is not used for the longest time), as following:

Request	3	8	2	3	9	1	6	3	8	9	3	6	2	1	3
Frame 5						1	1	1	1	1	1	1	2	2	2
Frame 4					9	9	9	9	9	9	9	9	9	9	9
Frame 3			2	2	2	2	2	2	8	8	8	8	8	1	1
Frame 2		8	8	8	8	8	6	6	6	6	6	6	6	6	6
Frame 1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Miss/hit	F	F	F	H	F	F	F	H	F	H	H	H	F	F	H

Number of Faults $= 9$. Number of Hits $= 6$

So, both incur the same number of page faults.

Correct Answer: A

GATE CSE 2016 Set 1 | Question: 49

59

Consider a computer system with ten physical page frames. The system is provided with an access sequence $(a_1, a_2, \ldots, a_{20}, a_1, a_2, \ldots a_{20})$, where each a_i is a distinct virtual page number. The difference in the number of page faults between the last-in-first-out page replacement policy and the optimal page replacement policy is_____.

gatecse-2016-set1 | operating-system | page-replacement | normal | numerical-answers

LIFO
Optimal

https://gateoverflow.in/39711/gate-cse-2016-set-1-question-49

74

Answer is 1.

In LIFO first 20 are page faults followed by next 9 hits then next 11 page faults. (After a_{10}, a_{11} replaces a_{10}, a_{12} replaces a_{11} and so on)

In optimal first 20 are page faults followed by next 9 hits then next 10 page faults followed by last page hit.

Best answer

answered Feb 13, 2016 • edited Jun 29, 2018 by **kenzou**

comment Follow share this

Krishna murthy

GATE IT 2008 | Question: 41

59

Assume that a main memory with only 4 pages, each of 16 bytes, is initially empty. The CPU generates the following sequence of virtual addresses and uses the Least Recently Used (LRU) page replacement policy.

$0, 4, 8, 20, 24, 36, 44, 12, 68, 72, 80, 84, 28, 32, 88, 92$

How many page faults does this sequence cause? What are the page numbers of the pages present in the main memory at the end of the sequence?

A. 6 and $1, 2, 3, 4$
B. 7 and $1, 2, 4, 5$
C. 8 and $1, 2, 4, 5$
D. 9 and $1, 2, 3, 5$

gateit-2008 operating-system page-replacement normal

LRU

https://gateoverflow.in/3351/gate-it-2008-question-41

75

Best answer

At first we have to translate the given virtual addresses (which addresses a byte) to page addresses (which again is virtual but addresses a page). This can be done simply by dividing the virtual addresses by page size and taking the floor value (equivalently by removing the page offset bits). Here, page size is 16 bytes which requires 4 offset bits. So,

$$0, 4, 8, 20, 24, 36, 44, 12, 68, 72, 80, 84, 28, 32, 88, 92 \implies 0, 0, 0, 1, 1, 2, 2, 0, 4, 4, 5,$$

We have 4 spaces for a page and there will be a replacement only when a 5^{th} distinct page comes. Lets see what happens for the sequence of memory accesses:

Incoming Virtual Address	Page Address	No. of Page Faults	Pages in Memory in LRU Order
0	0	1	0
4	0	1	0
8	0	1	0
20	1	2	0, 1
24	1	2	0, 1
36	2	3	0, 1, 2
44	2	3	0, 1, 2
12	0	3	1, 2, 0
68	4	4	1, 2, 0, 4
72	4	4	1, 2, 0, 4
80	5	5	2, 0, 4, 5
84	5	5	2, 0, 4, 5
28	1	6	0, 4, 5, 1
32	2	7	4, 5, 1, 2
88	5	7	4, 1, 2, 5
92	5	7	4, 1, 2, 5

So, (**B**) choice.

GATE CSE 2017 Set 1 | Question: 40

42

Recall that Belady's anomaly is that the page-fault rate may *increase* as the number of allocated frames increases. Now, consider the following statements:

- S_1: *Random page replacement* algorithm (where a page chosen at random is replaced) suffers from Belady's anomaly.
- S_2: *LRU page replacement* algorithm suffers from Belady's anomaly.

Which of the following is CORRECT?

A. S_1 is true, S_2 is true
B. S_1 is true, S_2 is false
C. S_1 is false, S_2 is true
D. S_1 is false, S_2 is false

gatecse-2017-set1 page-replacement operating-system normal

LRU

https://gateoverflow.in/118323/gate-cse-2017-set-1-question-40

Answer: B

69

Best answer

A page replacement algorithm suffers from Belady's anamoly when it is not a stack algorithm.

A stack algorithm is one that satisfies the inclusion property. The inclusion property states that, at a given time, the contents(pages) of a memory of size k page-frames is a subset of the contents of memory of size $k+1$ page-frames, for the same sequence of accesses. The advantage is that running the same algorithm with more pages(i.e. larger memory) will never increase the number of page faults.

Is LRU a stack algorithm?

Yes, LRU is a stack algorithm. Therefore, it doesn't suffer from Belady's anamoly.

Ref : Ref1 and Ref2

Is Random page replacement algorithm a stack algorithm?

No, as it may choose a page to replace in FIFO manner or in a manner which does not satisfy inclusion property. This means it could suffer from Belady's anamoly.

$\therefore$ (**B**) should be answer.