Are You Ready?

Lets Begin the Unlimited Learning

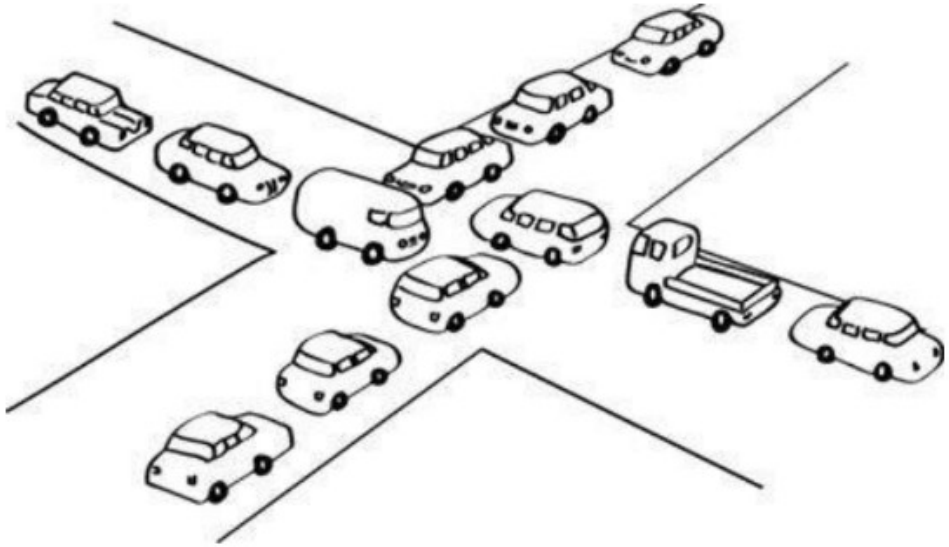
Operations on Resources

3 operations on resources:

1. Request
2. Use
3. Release

Deadlock

If two or more processes are waiting for such an event which is never going to occur

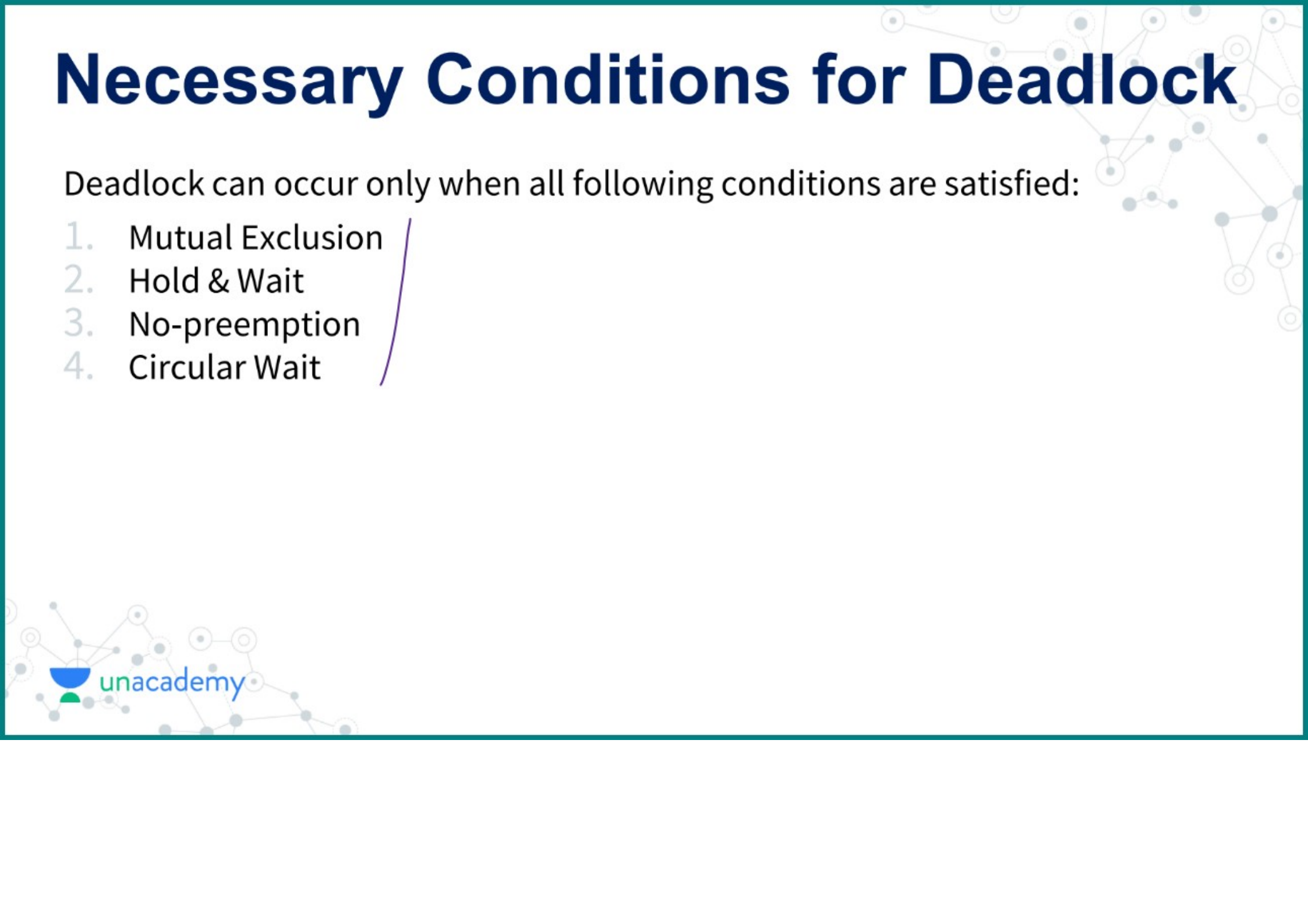


unacademy

Necessary Conditions for Deadlock

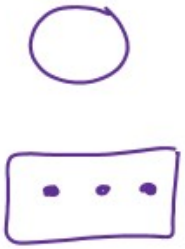
Deadlock can occur only when all following conditions are satisfied:

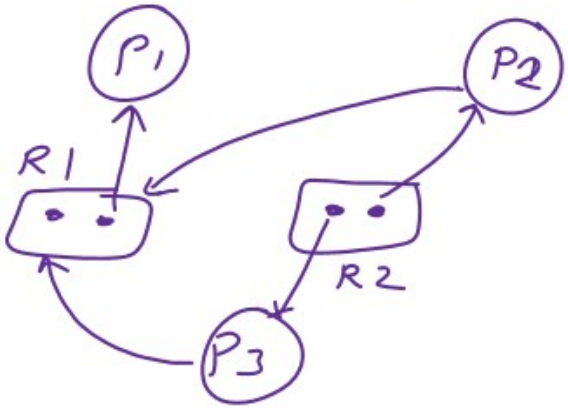
1. Mutual Exclusion
2. Hold & Wait
3. No-preemption
4. Circular Wait



Resource Allocation Graph

Nodes
- 1. Process
- 2. Resource





Edges
- Allocation — from instance to Process
- Request — —11— Process to Resource

unacademy

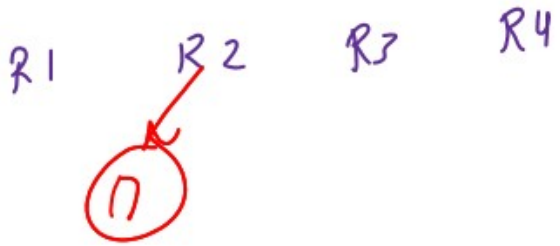
Recovery From Deadlock

1. Make Sure that deadlock never occur
   - Prevent the system from deadlock or avoid deadlock
2. Allow deadlock, detect and recover
3. Pretend that there is no any deadlock

Deadlock Prevention

Prevent any of four necessary conditions to occur

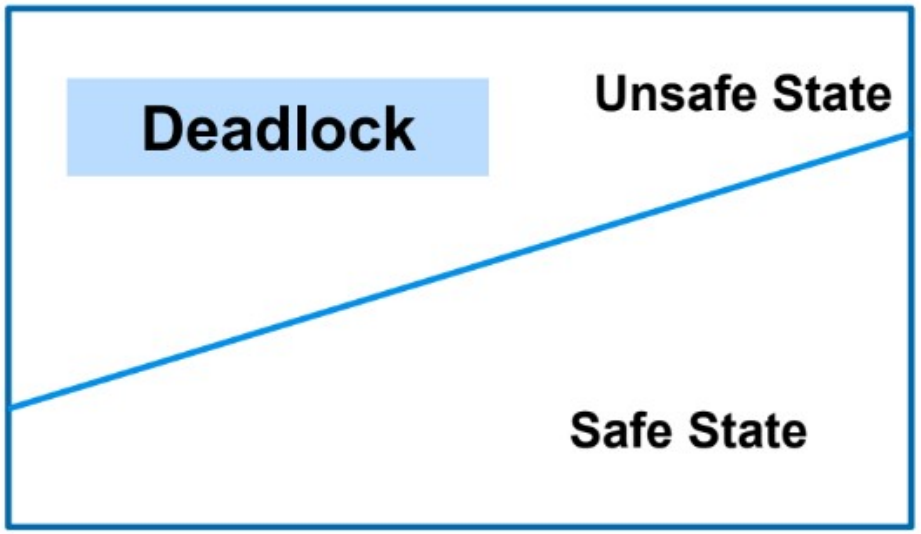
1. Mutual Exclusion
2. Hold & Wait
3. No Preemption
4. Circular Wait



unacademy

Deadlock Avoidance

In deadlock avoidance, the OS tries to keep system in safe state



unacademy

Banker's Algorithm

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety

→ safety

→ Resource Request

unacademy

Banker's Algorithm

Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	~~3 3 2~~
P_1	2 0 0	3 2 2	P1 5 3 2
P_2	3 0 2	9 0 2	P3 7 4 3
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Need

A	B	C
7	4	3
1	2	2
6	0	0
0	1	1
4	3	1

Safe seq. = P1, P3, P2, P0, P4

unacademy

Resource Request Algorithm

1. $Req_i \leq Need_i$ ✓
2. $Req_i \leq Available$
3. $Allocat^n_i = Allocat^n_i + Req_i$

   $Available = Available - Req_i$

   $Need_i = Need_i - Req_i$
4. Safety Algo → safe ⇒ granted
   → unsafe ⇒ rejected

Question

What will happen if process P1 requests one additional instance of resource type A and two instances of resource type C?

$Req_1 < 1, 0, 2 >$

Banker's Algorithm

Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	~~3 3 2~~ 2 3 0
P_1	~~3 2 0 0 2~~	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

+2 2
0 2 0

safe → Granted

Recovery From Deadlock

1. Make Sure that deadlock never occur
   - Prevent the system from deadlock or avoid deadlock
2. Allow deadlock, detect and recover
3. Pretend that there is no any deadlock

Deadlock Detection

1. When all resources have single instance
2. When resources have multiple instances ⇒ Banker's algo

Deadlock Detection

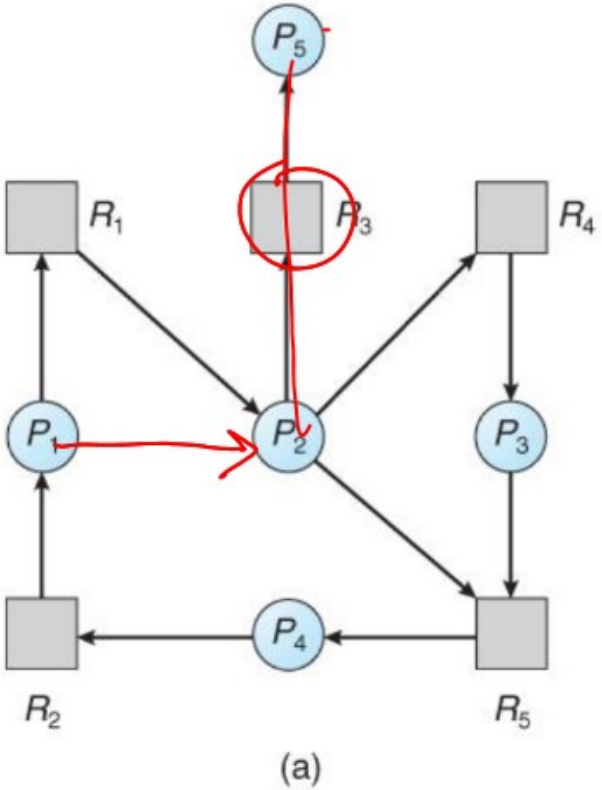
When all resources have single instance:

Deadlock detection is done using wait-for-graph

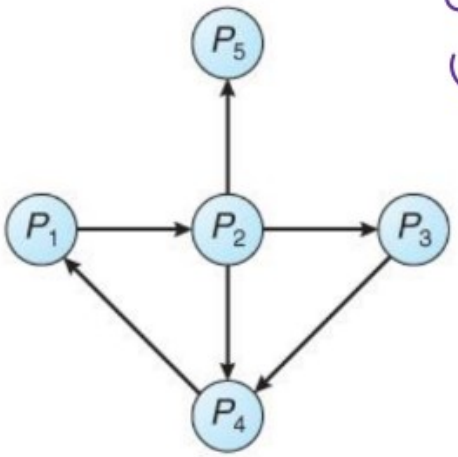
Wait For Graph

It is created from resource allocation graph

Wait For Graph



(a)

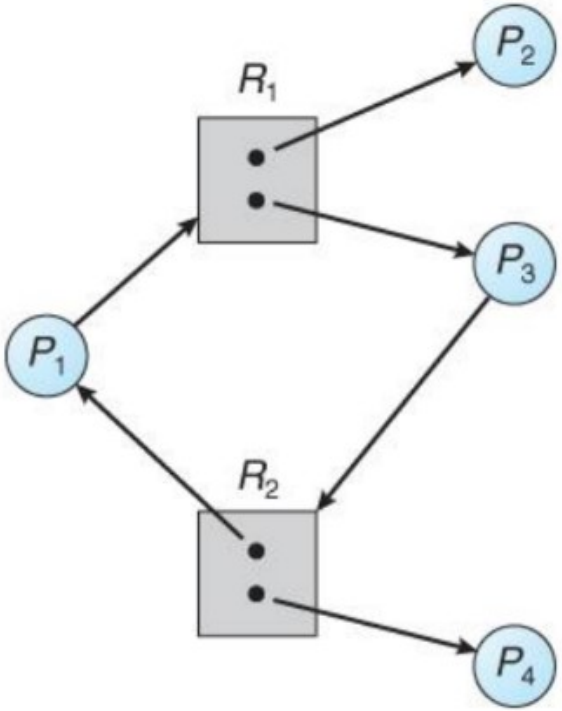


(b)

cycle
⇓
deadlock

unacademy

Wait For Graph: Example



	Allocath		Req.	
	R1	R2	R1	R2
P1	0	1	1	0
P2	1	0	0	0
P3	1	0	0	1
P4	0	1	0	0

unacademy

Deadlock Detection

When resources have multiple instance:

Deadlock detection is done using a specific algorithm

Deadlock Detection Algorithm

	Allocation	Request	Available
	A B C	A B C	A B C
P_0	0 1 0	0 0 0	~~0 0 0~~
P_1	2 0 0	2 0 2	~~0 1 0~~
P_2	3 0 3	0 0 0	~~3 1 3~~
P_3	2 1 1	1 0 0	~~5 1 3~~
P_4	0 0 2	0 0 2	~~7 3 4~~
			7 3 6

Question

Consider a system with 3 processes A, B and C. All 3 processes require 4 resources each to execute. The minimum number of resources the system should have such that deadlock can never occur? 10

n processes

k resources

$n(k-1)+1$

Recovery From Deadlock

There are three basic approaches to recovery from deadlock:

1. Inform the system operator and allow him/her to take manual intervention
2. Terminate one or more processes involved in the deadlock
3. Preempt resources.

Process Termination

1. Terminate all processes involved in the deadlock
2. Terminate processes one by one until the deadlock is broken

Process Termination

Many factors that can go into deciding which processes to terminate next:

1. Process priorities.
2. How long the process has been running, and how close it is to finishing.
3. How many and what type of resources is the process holding
4. How many more resources does the process need to complete
5. How many processes will need to be terminated
6. Whether the process is interactive or batch

Resource Preemption

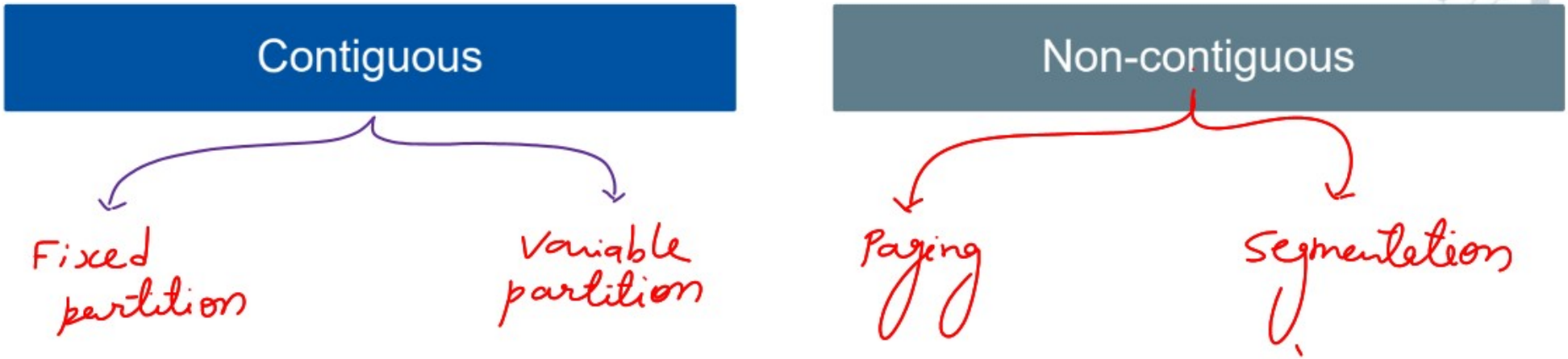
Important issues to be addressed when preempting resources to relieve deadlock:

1. Selecting a victim
2. Rollback
3. Starvation

Memory Management

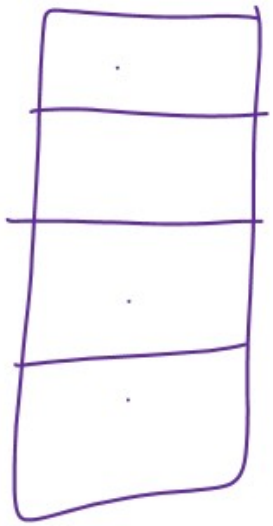
- Module of OS
- Functions:
  1. Memory allocation
  2. Memory deallocation
  3. Memory protection
- Goals:
  1. Maximum Utilization of space
  2. Ability to run larger programs with limited space

Memory Management Techniques



unacademy

Fixed Partition Contiguous MMT

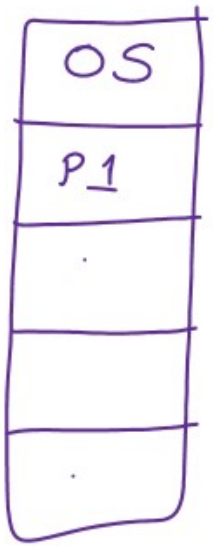


unacademy

Partition Allocation Policy

- First fit
- Best fit
- worst fit

Variable Partition Contiguous MMT



External Fragmentation ⇒ Soln
↳ Compaction

Non-Contiguous MMT

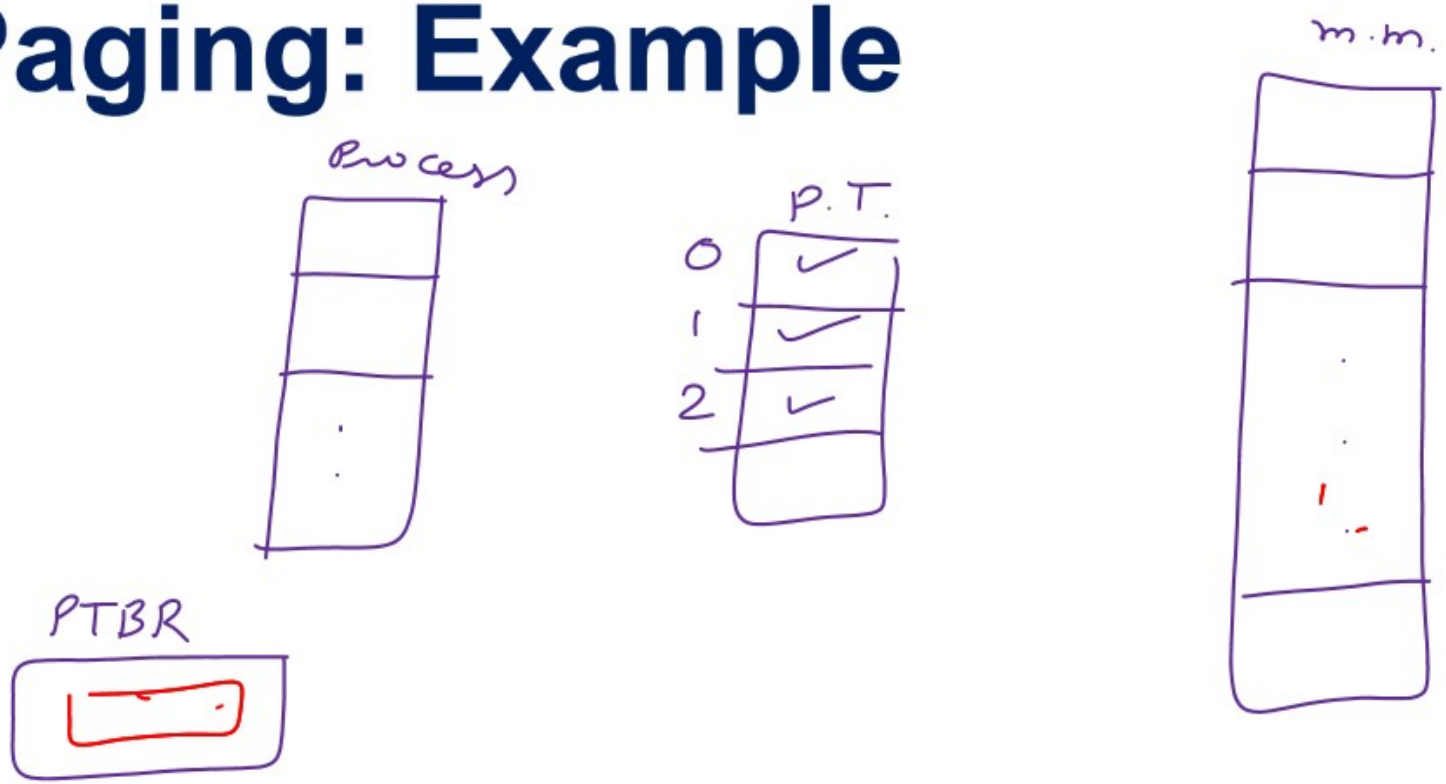
- Process is scattered in memory, not allocated at one area
- Two techniques:
  - Paging: Scattered in same size of memory areas
  - Segmentation: Scattered in variable size of memory areas

Paging

- Process is divided in equal size of pages
- Physical memory is divided in same equal size of frames
- Processor will have a view of process and its pages
- Pages are scattered in frames
- Page table is used to map a process page to a physical frame

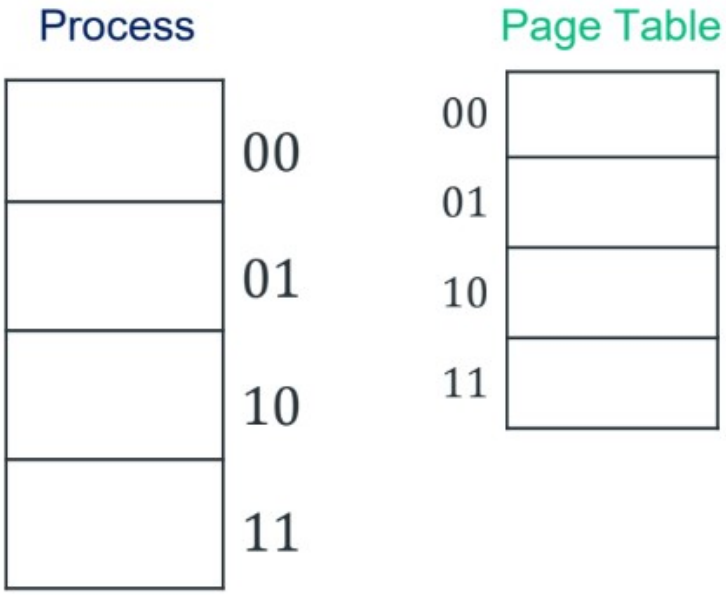
- 1 P.T. entry = frame no. + protection bits
  (translatn)
- No. of entries in PT = No. of pages in process

Paging: Example

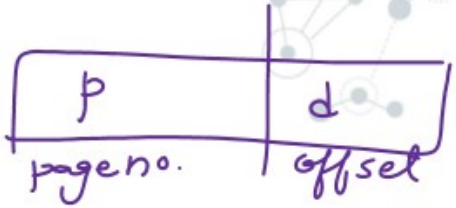


unacademy

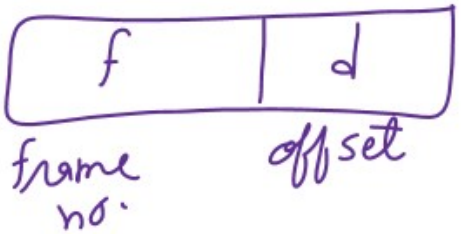
Paging



L.A.



P.A.



unacademy

Paging

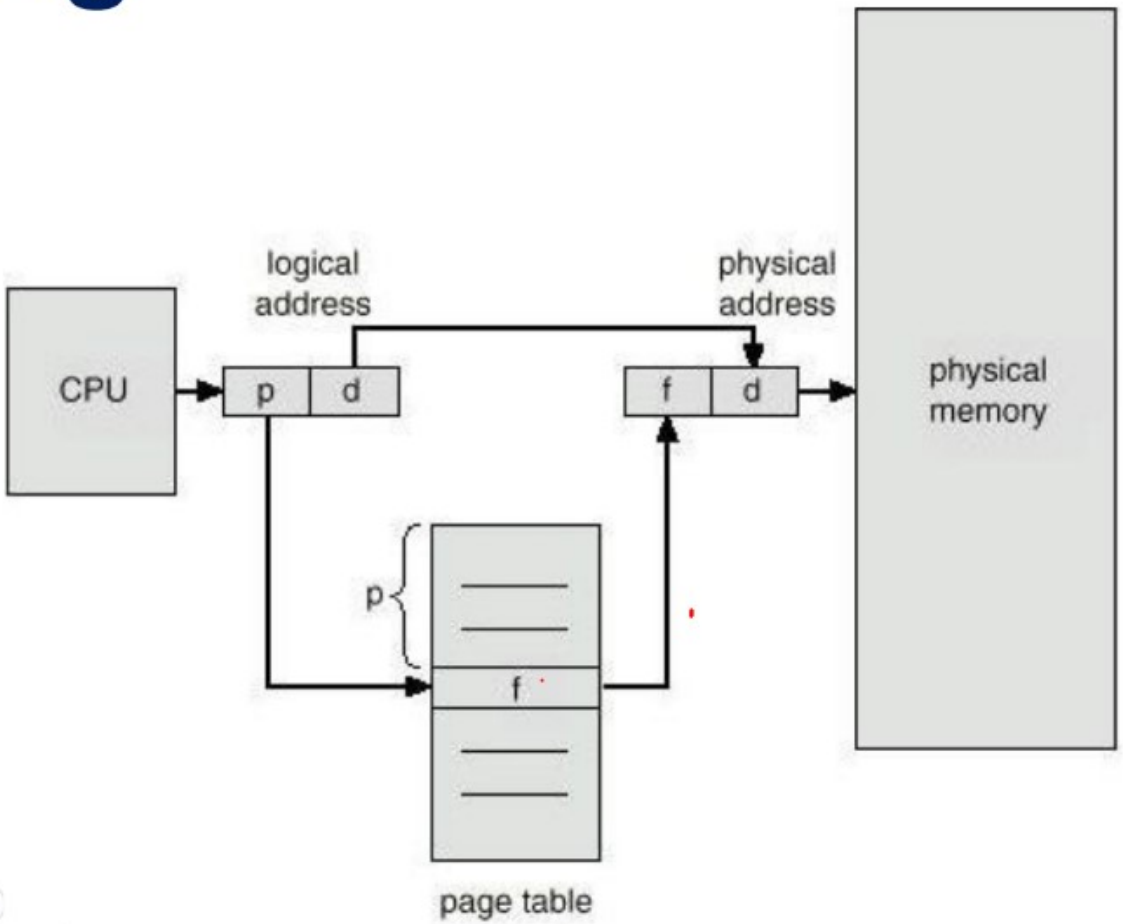
$$\text{P.T. size} = \text{no. of entry} * \text{1 entry size}$$

$$= \text{no. of pages} * (f + \text{extra bits})$$

$$\text{no. of pages} = \frac{\text{L.A.S.}}{\text{Page size}}$$

$$\text{no. of frames} = \frac{\text{P.A.S.}}{\text{page size}}$$

Paging

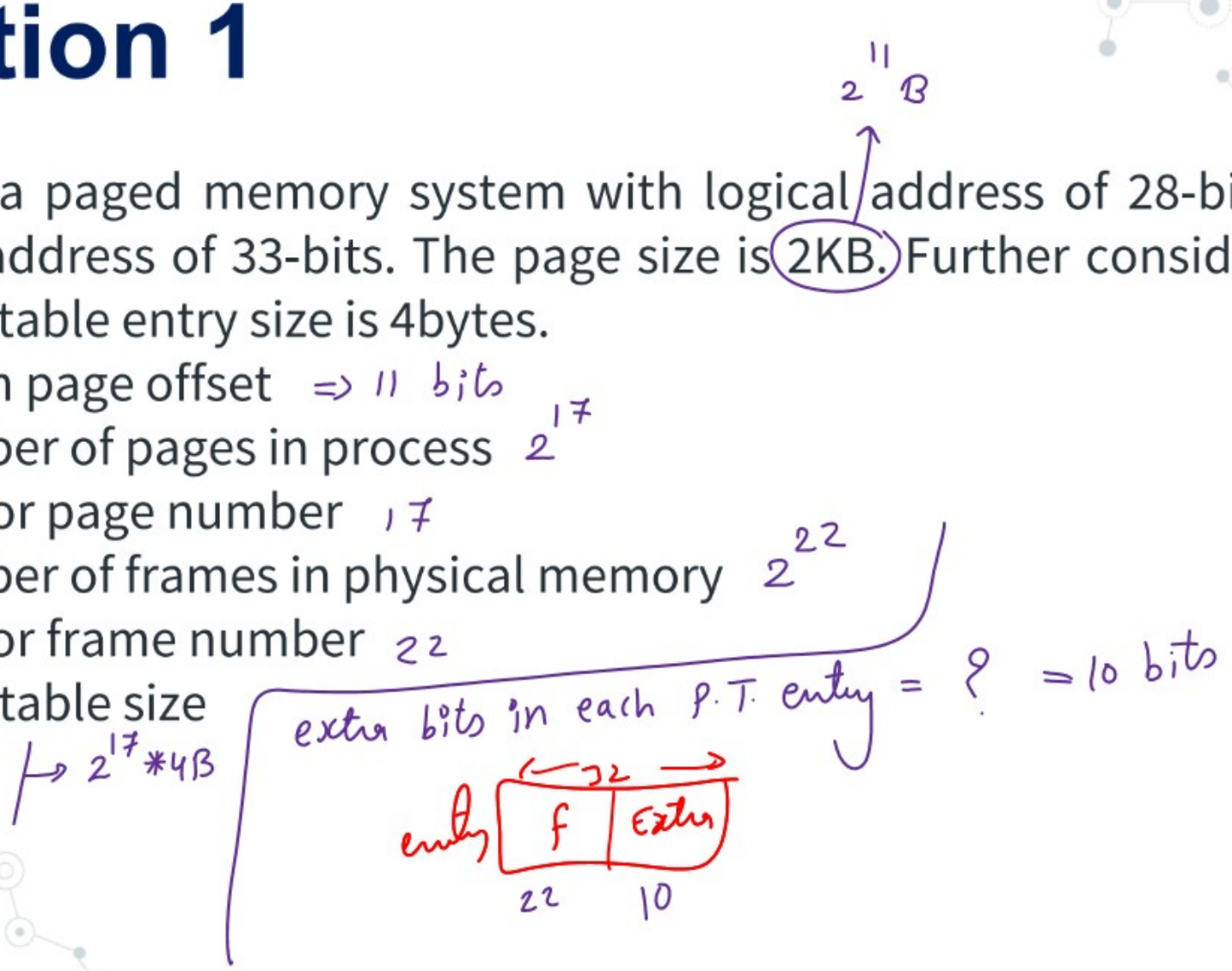


unacademy

Question 1

Consider a paged memory system with logical address of 28-bits and physical address of 33-bits. The page size is 2KB. Further consider that one page table entry size is 4bytes.

1. Bits in page offset
2. Number of pages in process
3. Bits for page number
4. Number of frames in physical memory
5. Bits for frame number
6. Page table size



Time Required in Paging

$E.M.A.T. = 2 * t_{mm}$

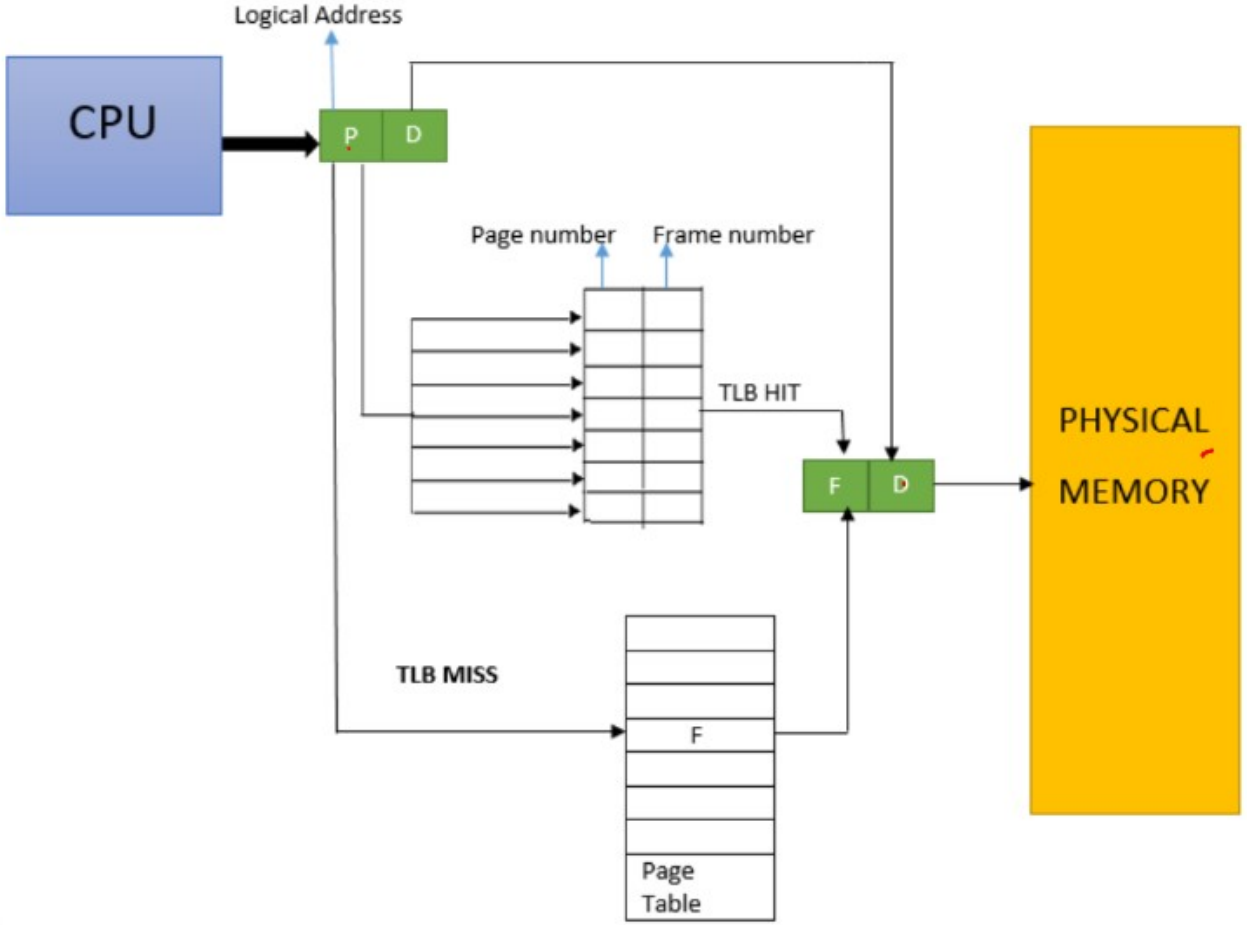
if P.T. is Registers

$= t_{mm}$

TLB

A translation lookaside buffer is a memory hardware that is used to reduce the time taken to access a user memory location

TLB



unacademy

Effective Access Time With TLB

$$E.M.A.T. = t_{TLB} + t_{mm} + (1-H)\, t_{mm}$$

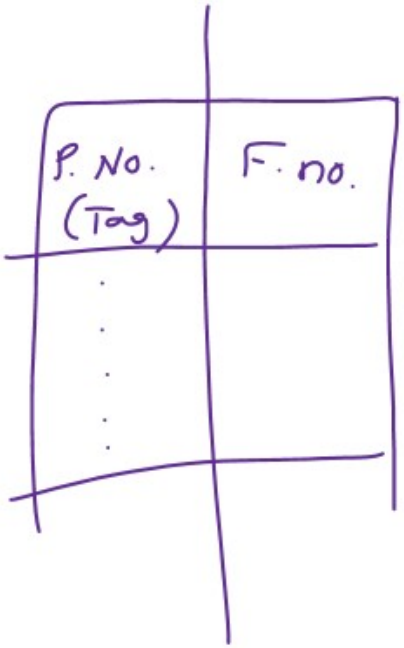
TLB

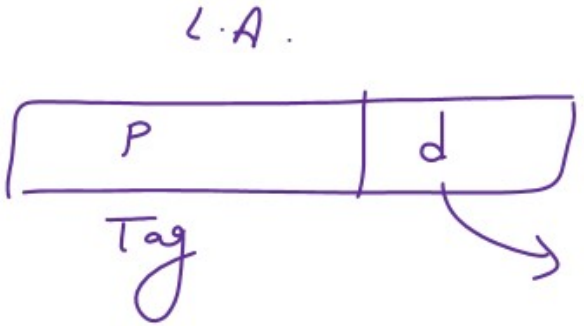
A translation lookaside buffer is a memory hardware that is used to reduce the time taken to access a user memory location

TLB Mappings

Full Associative Mappings

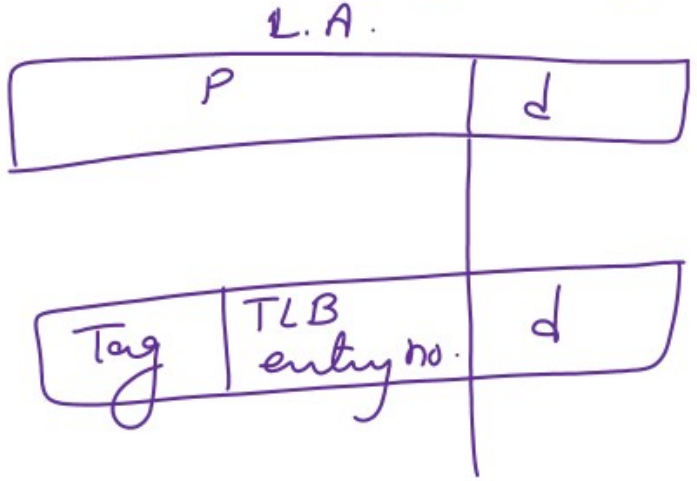
Costly



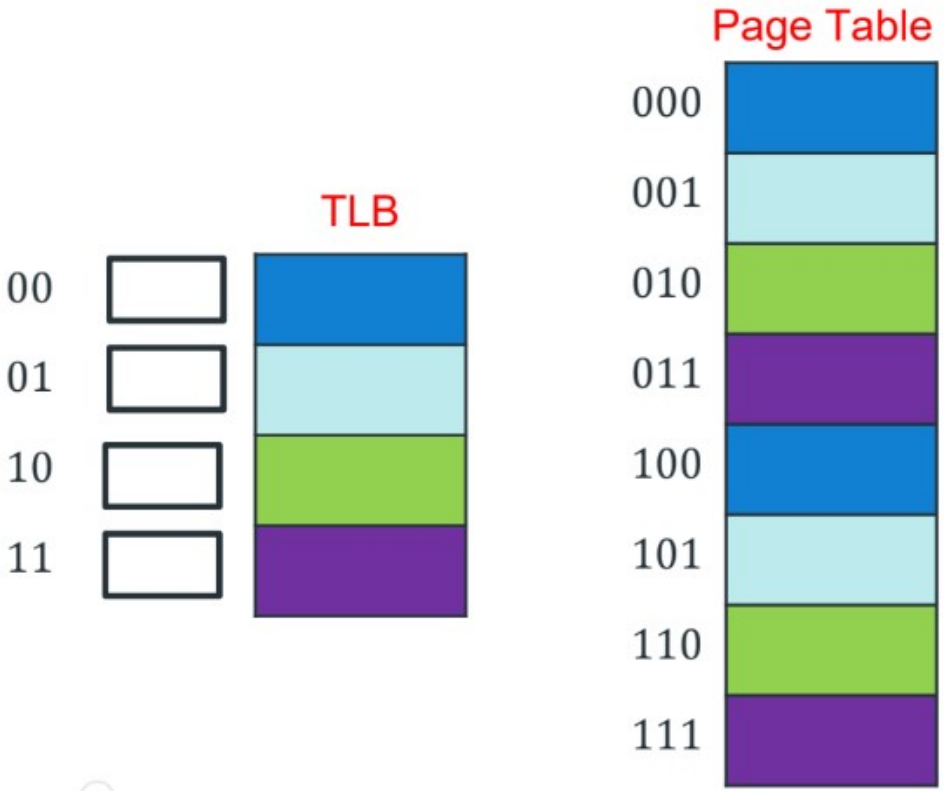


unacademy

Direct Mappings

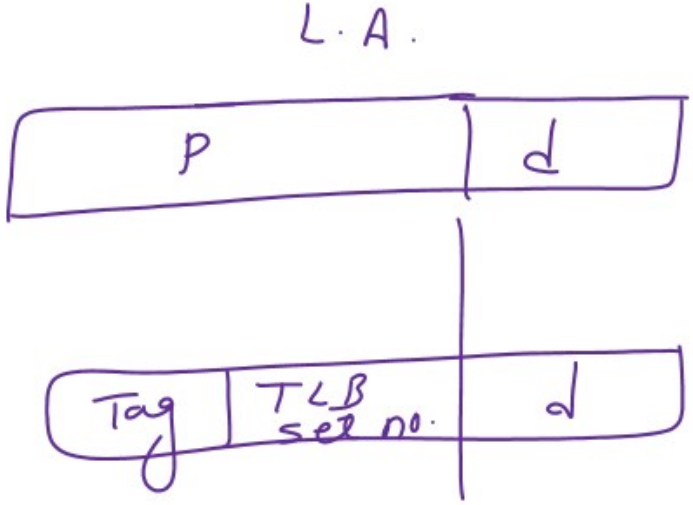


Problem With Direct Mappings



unacademy

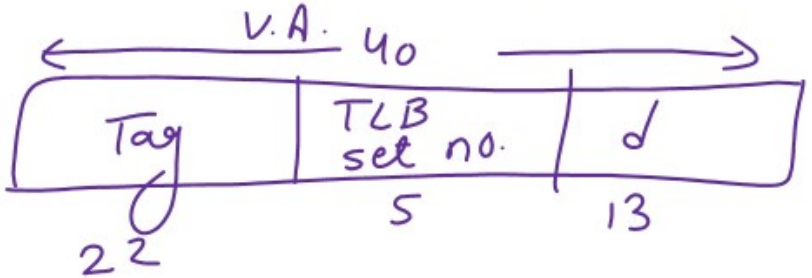
Set Associative Mappings



$$\text{no. of sets in TLB} = \frac{\text{no. of entries in TLB}}{\text{associative}}$$

Question GATE-2015

A computer system implements a 40 bit virtual address, page size of 8 kilobytes, and a 128-entry translation look-aside buffer (TLB) organized into 32 sets each having four ways. Assume that the TLB tag does not store any process id. The minimum length of the TLB tag in bits is _____?



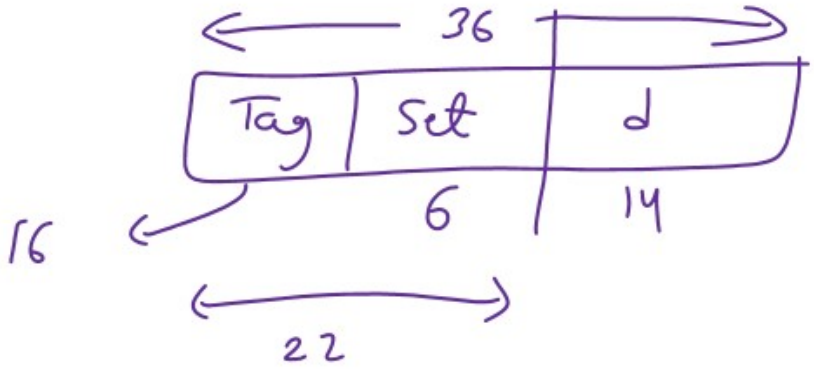
(A) 20
(B) 10
(C) 11
(D) 22

unacademy

Question

Ans = 16

A Computer system implements a 36-bits virtual address, page size of 16Kbytes and a 256-entry TLB organized into 64 sets each having 4-ways. Assume that the TLB tag does not store any process-id. The minimum length of the TLB tag in bits is _____?



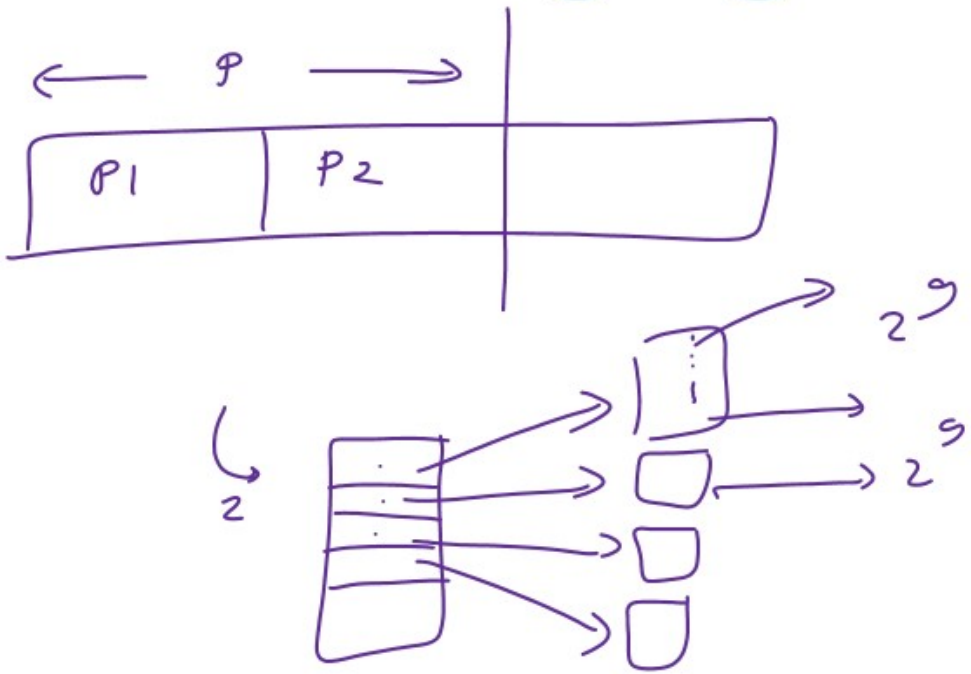
unacademy

Optimal Page Size

$$\sqrt{2 * \text{Process size} * \text{P.T. entry size}}$$

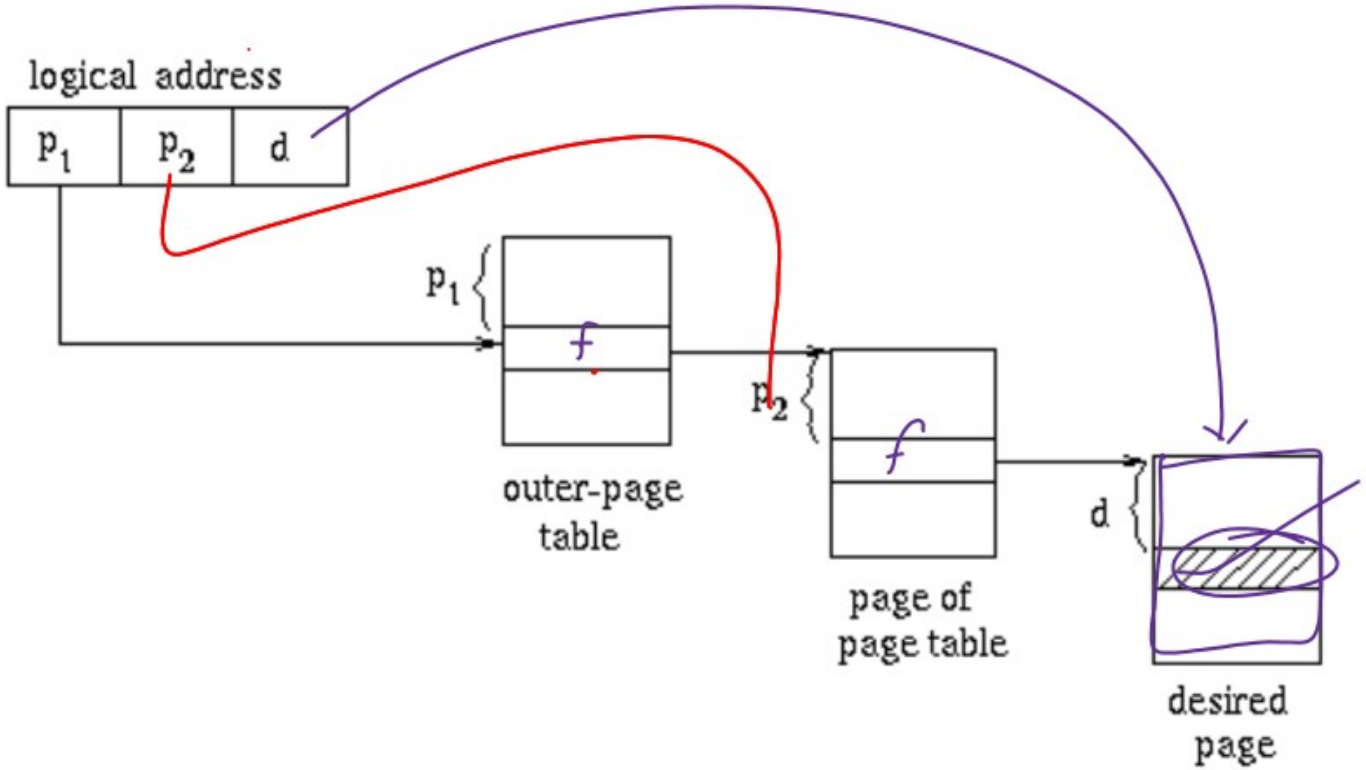
Page Table in Memory

Multilevel Paging



unacademy

Multilevel Paging



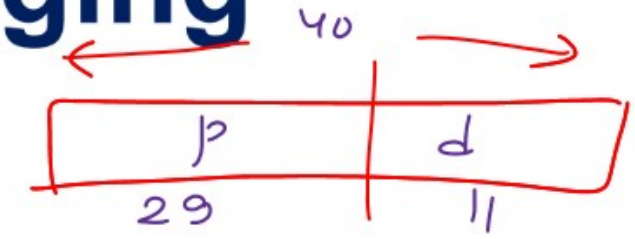
unacademy

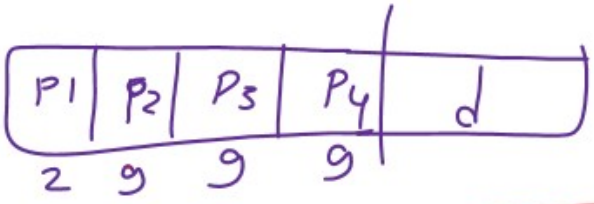
Multilevel Paging

Page size = 2KB

P.T. ent. = 4B

L.A. = 40-bits





no. of entries in 1 page $= \frac{2KB}{4B}$

$= 2^9$

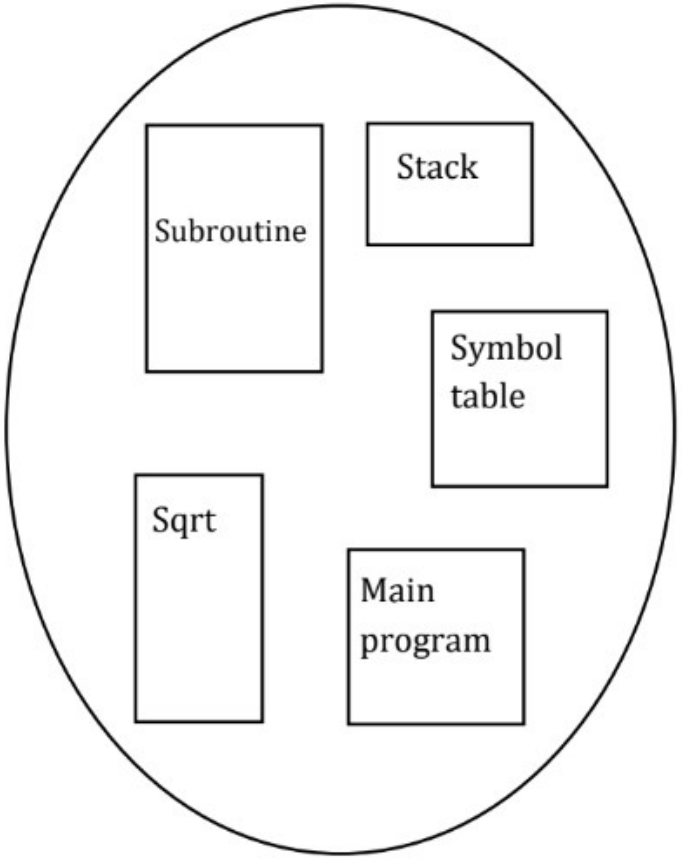
first level	2nd level	3rd	4th
1	4	$4*2^9$	$4*2^9*2^9$

unacademy

Segmentation

- Divide Process in logically related partitions (Segments)
- Segments are scattered in physical memory

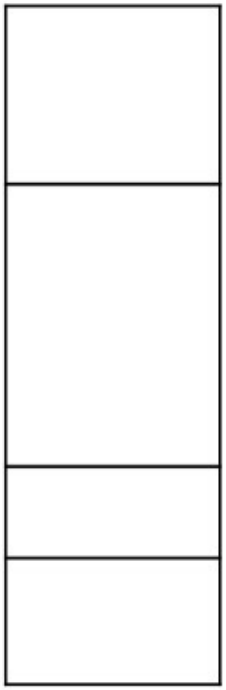
Segmentation



Logical address

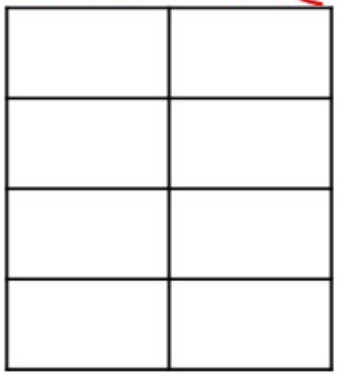
unacademy

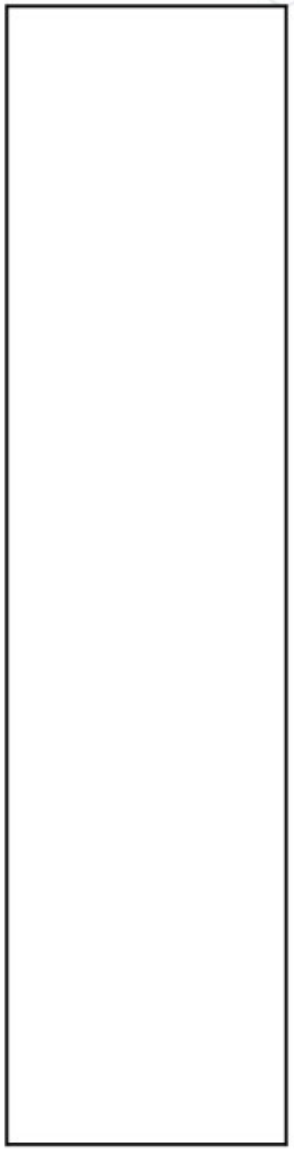
Segmentation



S. T.

Limit Base





unacademy

Segmentation

- Size of segment can vary, so along with base, keep limit information also
- Limit defines max number of words within the segment

Segmentation

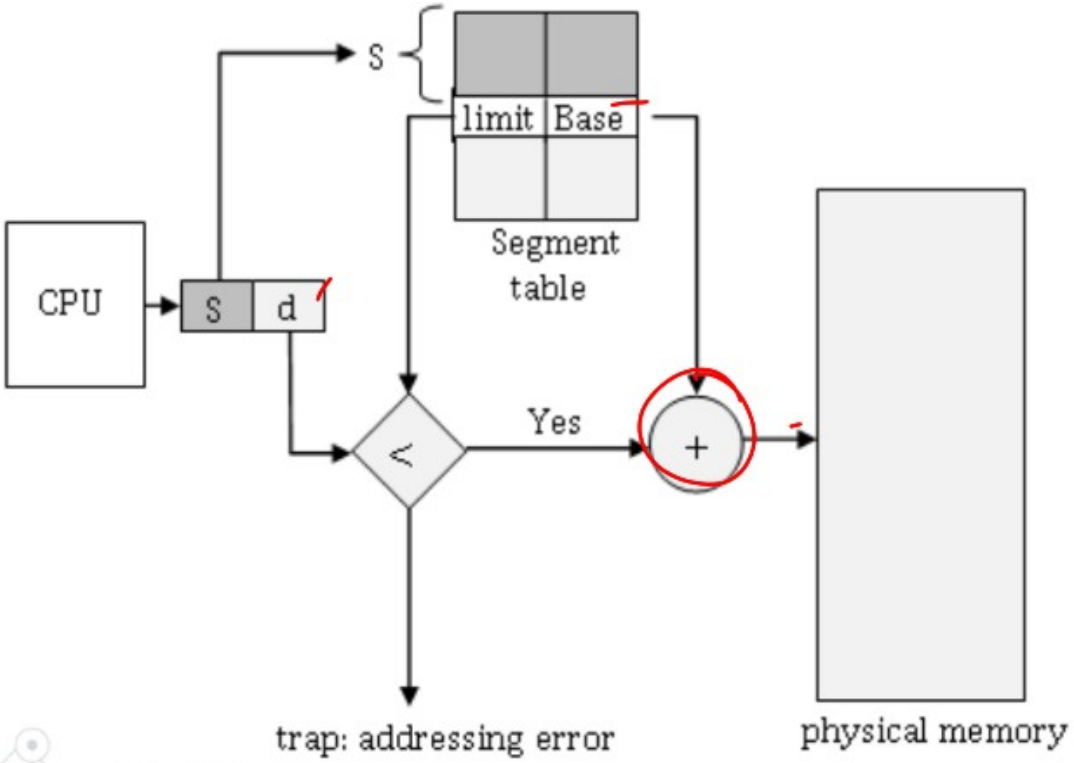
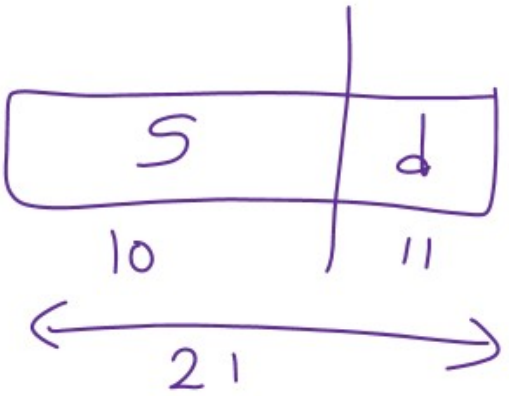


Fig. Segmentation hardware

Question

d = 11

- Maximum segment size = 2KB
- Number of segments in process = 2^{10}
- Logical address = _____ bits ??

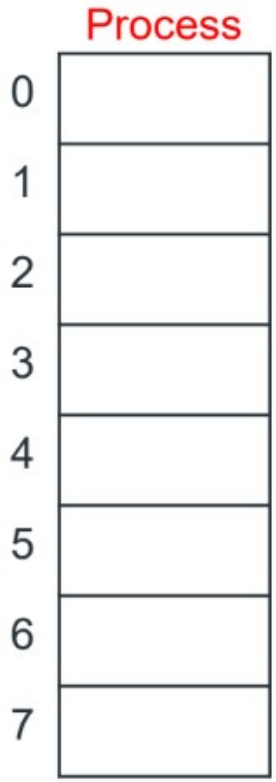


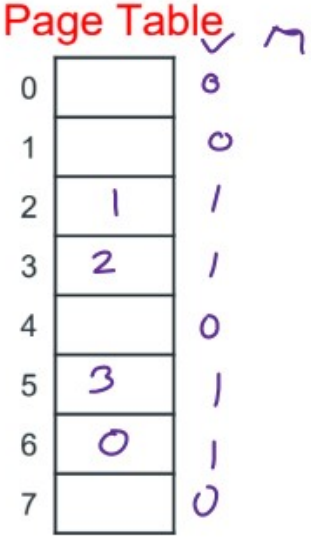
unacademy

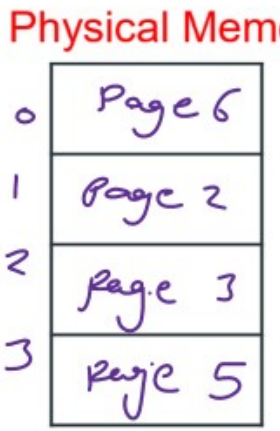
Virtual Memory

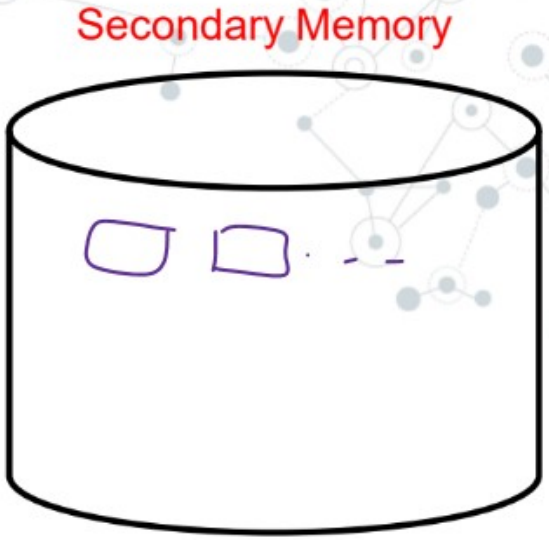
- Feature of OS
- Enables to run larger process with smaller available memory

Virtual Memory









unacademy

Demand Paging

- Bring pages in memory when CPU demands

Page Replacement Policies

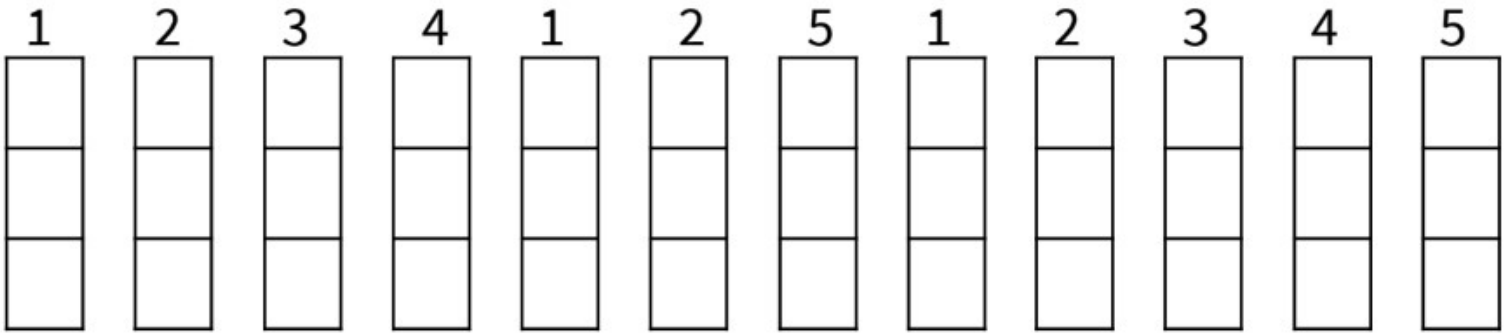
1. First In First Out (FIFO)
2. Optimal Policy
3. Least Recently Used (LRU)
4. Least Frequently Used (LFU)
5. Most Frequently Used (MFU)
6. Last In First Out (LIFO)

First In First Out (FIFO)

Page fault rate = $\frac{\text{no. of P.F.}}{\text{Total references}}$

Assume:

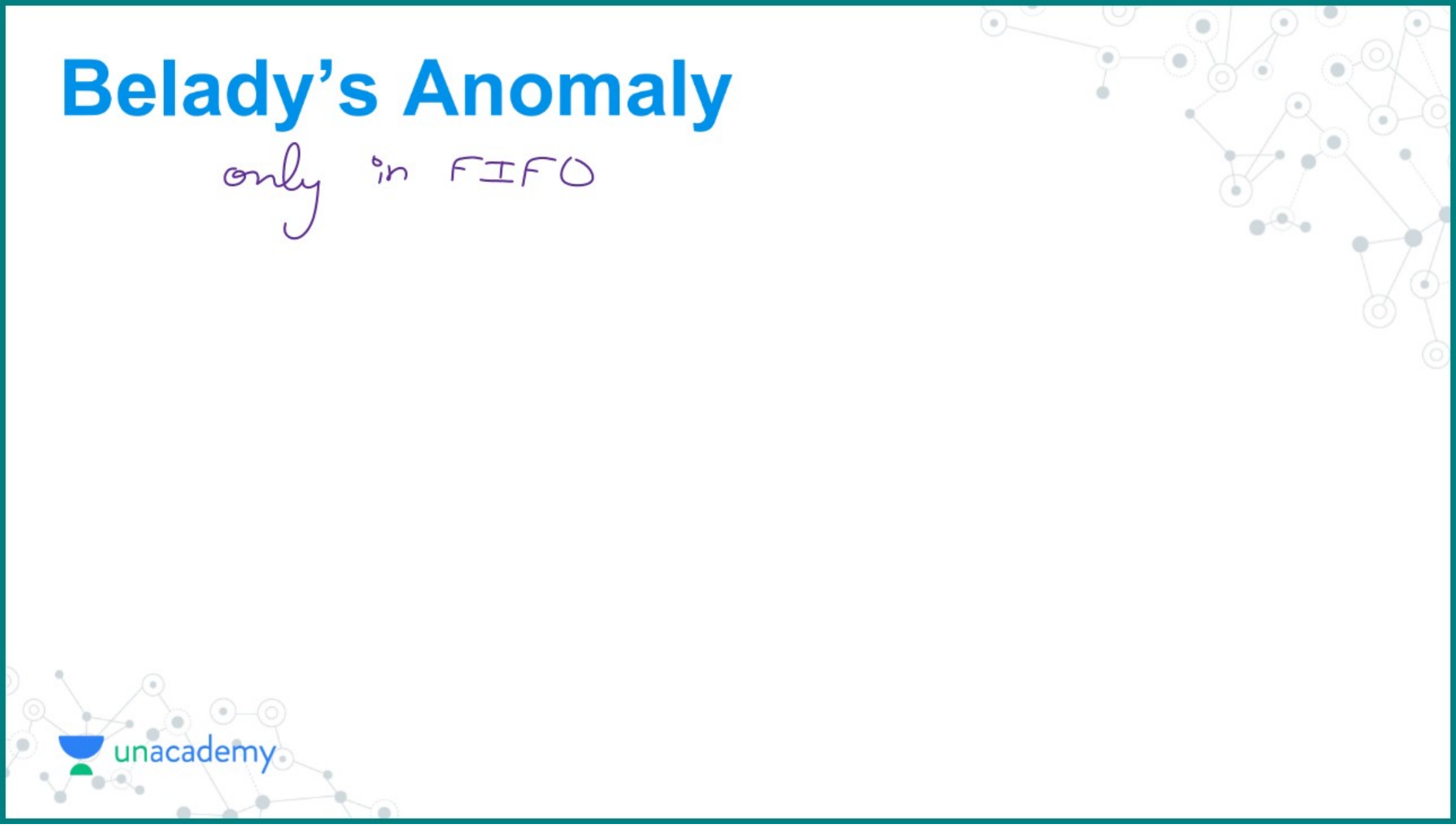
- Number of frames = 3 (All empty initially)
- Page reference sequence: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



unacademy

Belady's Anomaly

only in FIFO



First In First Out (FIFO)

Advantages

1. Simple and easy to implement.
2. Low overhead.

Disadvantages:

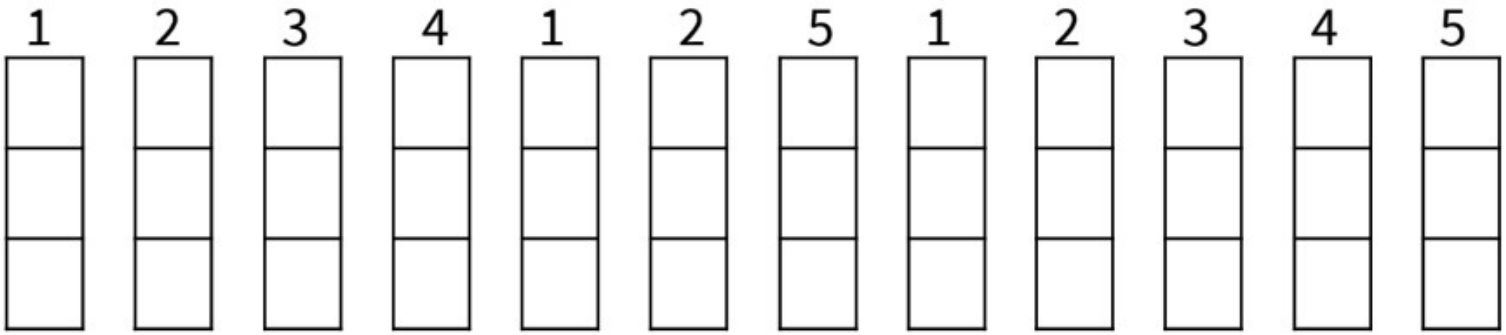
1. Poor performance.
2. Doesn't consider the frequency of use or last used time, simply replaces the oldest page.
3. Suffers from Belady's Anomaly

Optimal Policy

↳ replace a page which is going to refered very late in future

Assume:

- Number of frames = 3 (All empty initially)
- Page reference sequence: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



unacademy

Optimal Policy

Advantages

1. Easy to Implement ✓
2. Simple data structures are used ✓
3. Highly efficient => min page fault

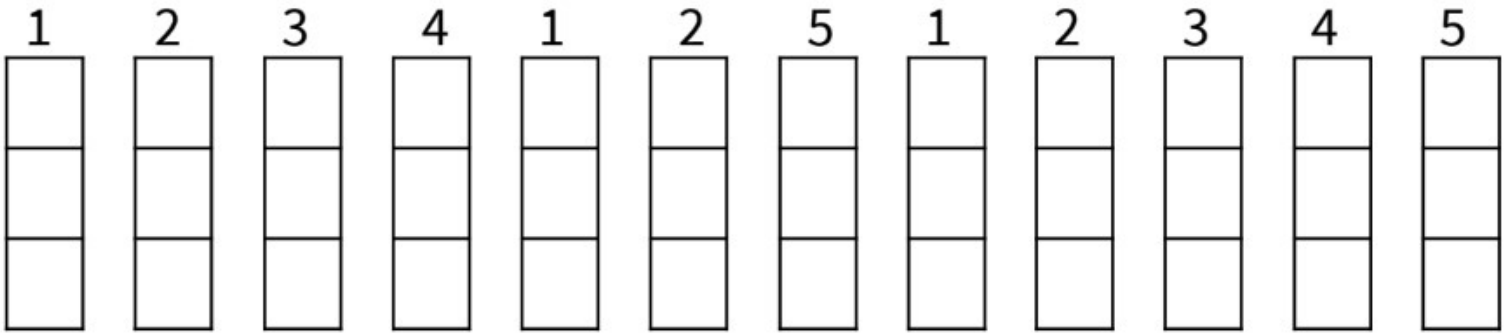
Disadvantages:

1. Requires future knowledge of the program ✓
2. Time-consuming

Least Recently Used (LRU)

Assume:

- Number of frames = 3 (All empty initially)
- Page reference sequence: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



unacademy

Least Recently Used (LRU)

Advantages

1. Efficient.
2. Doesn't suffer from Belady's Anomaly

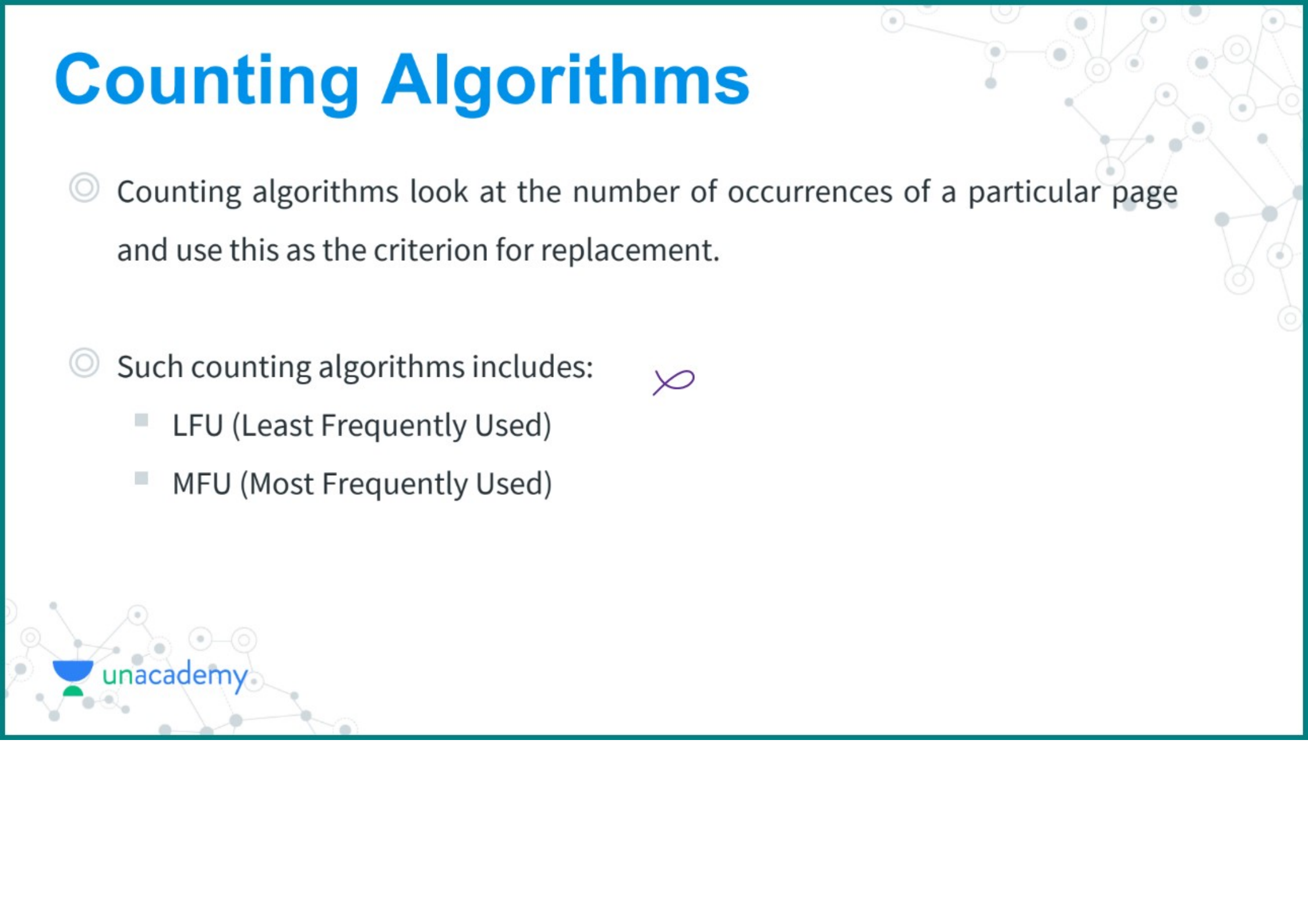
Disadvantages:

1. Complex Implementation
2. Expensive
3. Requires hardware support

Counting Algorithms

- Counting algorithms look at the number of occurrences of a particular page and use this as the criterion for replacement.

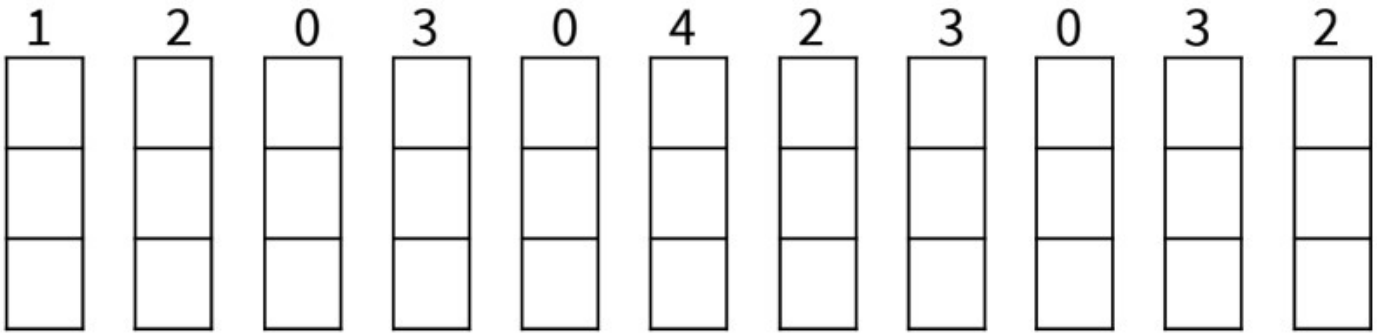
- Such counting algorithms includes:
  - LFU (Least Frequently Used)
  - MFU (Most Frequently Used)



Least Frequently Used (LFU)

Assume:

- Number of frames = 3 (All empty initially)
- Page reference sequence: 1 2 0 3 0 4 2 3 0 3 2

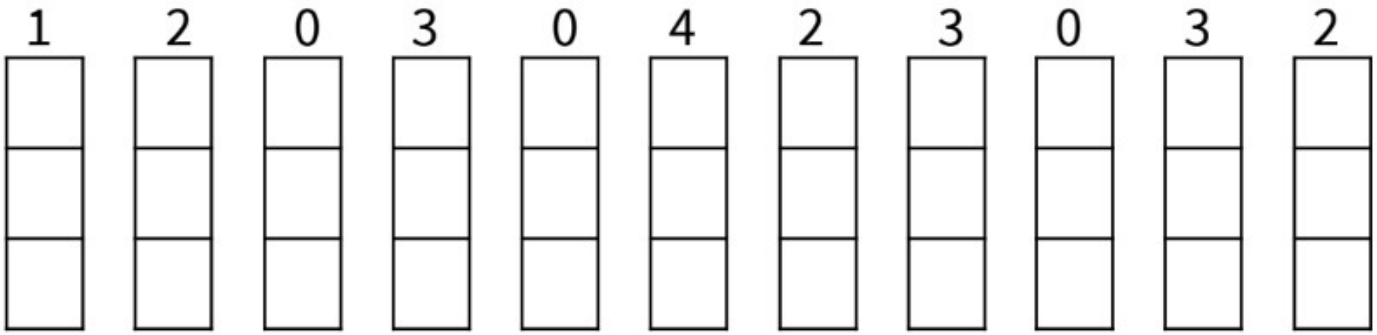


unacademy

Most Frequently Used (MFU)

Assume:

- Number of frames = 3 (All empty initially)
- Page reference sequence: 1 2 0 3 0 4 2 3 0 3 2

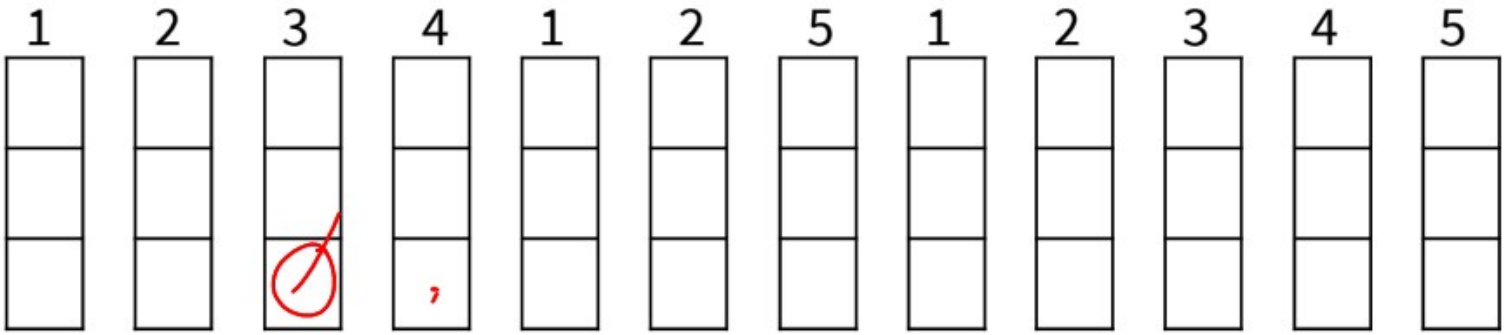


unacademy

Last In First Out (LIFO)

Assume:

- Number of frames = 3 (All empty initially)
- Page reference sequence: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



unacademy

Frame Allocation

How many frames do we allocate per process?

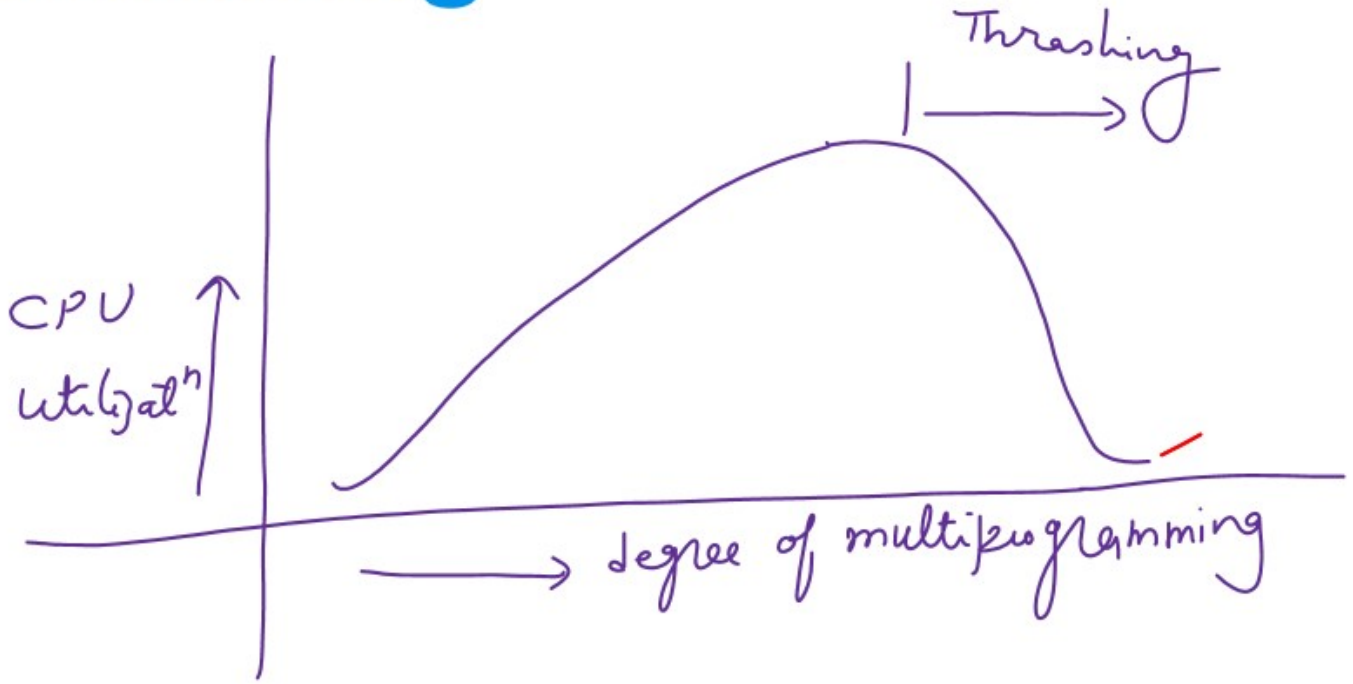
- If it is a single-user, single-tasking system, it's simple – all the frames belong to the user's process

Frame Allocation

1. Equal Allocation
2. Proportional Allocation

Local Vs Global Allocation

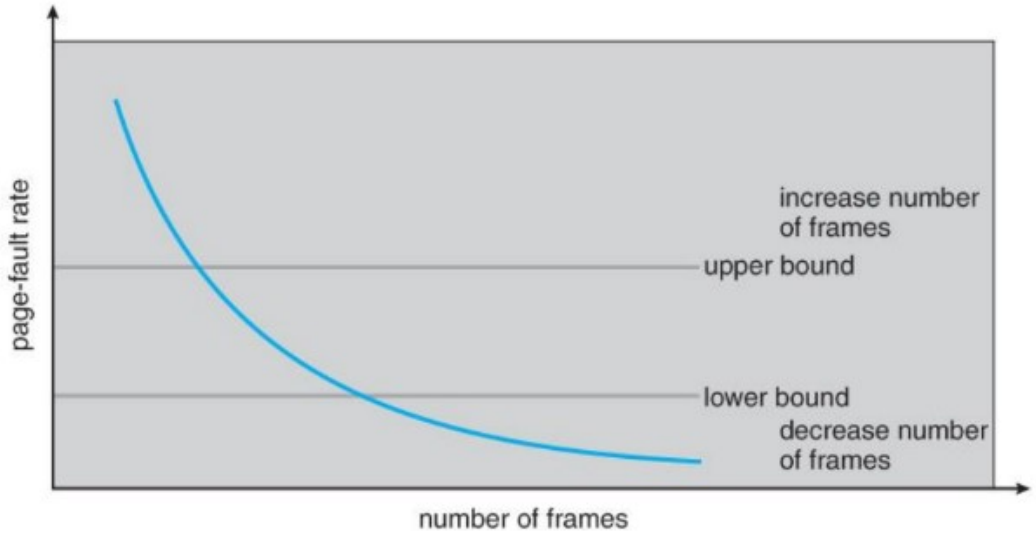
Thrashing



unacademy

How to Handle Thrashing

1. Working Set Model
2. Page Fault Frequency



unacademy

Let's Take a Break

See you in 10min