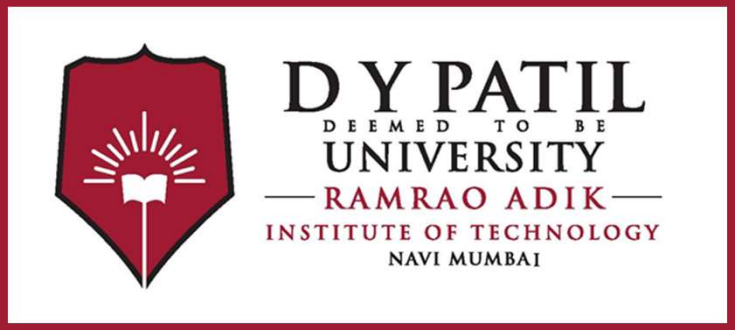



# Subject Name: Operating Systems

## Unit: 5 Unit Name: Memory Management

Faculty Name: Ms. Puja Padiya

# Index

# Lecture: Paging

## Virtual Memory in OS

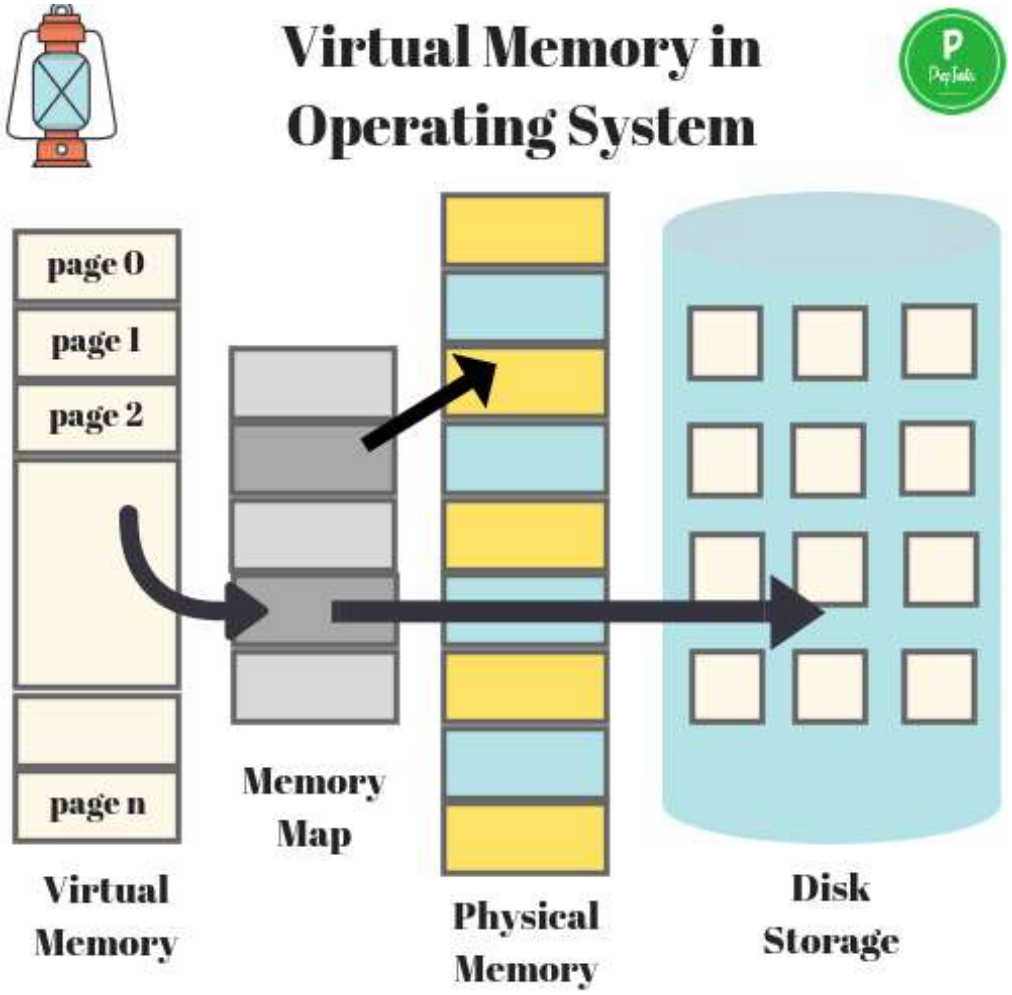
- The Primary Memory (such as RAM) is expensive and limited in the system, however, the secondary memory is cheaper and large and can be extended easily.

- Virtual memory concept offers unique solution for apparent primary memory expansion, in this we secondary memory can be accessed as if it were the part of the main memory. One of the techniques to extend this is to transfer the inactive RAM storage to secondary storage temporarily.

- It essentially with the help of Operating System, and the software and hardware installed on the system, helps to access more memory than what actually is installed on the system, by transferring data from the RAM to the to disk.

# Virtual Memory in OS

- **Primary Memory – RAM**
- The entry application or program in the system first loads into the RAM (Random Access Memory), this makes the application faster to load and access, than the secondary memory and is readily available
- **Secondary Memory – Hard Disk and Storage devices**



## In what Form Virtual Memory is stored?

- The memory is stored in form of units that we call as pages in Operating System, these are atomic units where large programs can be stored.

- For a program there may a large set of storage in physical addresses. All of those are not required to be loaded as only certain programs are used, for an application.

- Imagine playing a game like FIFA on your desktop. The game itself maybe stored in hard disk (secondary device), but while playing all the installed space would not be loaded into RAM. But only certain programs that are required for the application (Game) to run .

## Benefits of having Virtual Memory

- Large programs can be written, as virtual space available is huge compared to physical memory.
- Less I/O required, leads to faster and easy swapping of processes.
- More physical memory available, as programs are stored on virtual memory, so they occupy very less space on actual physical memory.

## How it works?

- The Addressing scheme that is provided by the system and total available space in the secondary storage device is what limits the size of the virtual storage, not the size of Primary storage as RAM. If the storage is unlimited (as in the case of cloud systems), some applications can behave as if they have unlimited primary memory available to them.

- We use both of these of the system –

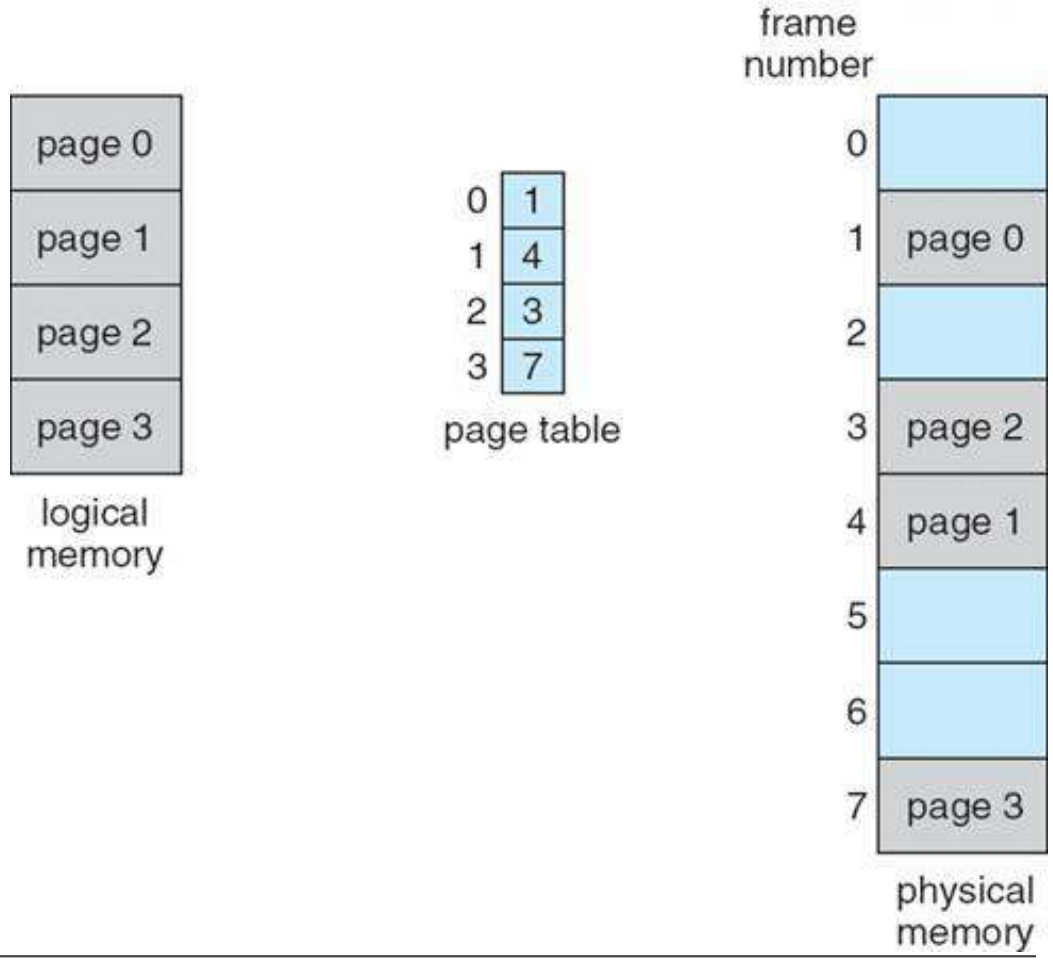
- Hardware
- Software

## Memory Management- Paging

- Paging is a **memory management** scheme that allows processes **physical memory to be discontinuous**
- Divide **physical memory** into **fixed-sized blocks** called ***frames*** (size is power of 2, between 512 bytes and 8,192 bytes)
- Divide **logical memory** into blocks of same size called ***pages***
- **Any page** ( from any process ) can be **placed into any available frame**
- The ***page table*** is used to look up what frame a particular page is stored in at the moment.
- Keep track of all **free frames**
- To run a program of size ***n* pages**, need to find ***n* free frames** and load program
- Set up a ***page table*** to **translate *logical*** to ***physical*** addresses
- Produce **Internal fragmentation**

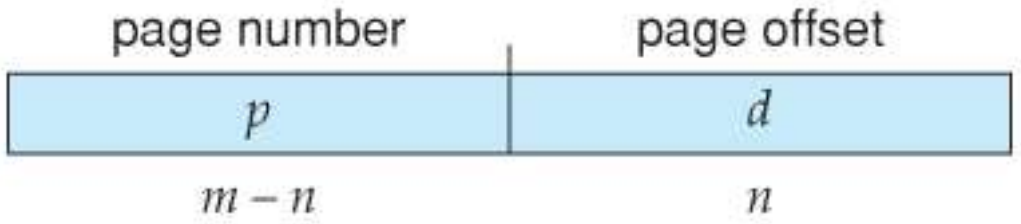
## Memory Management- Paging

- In the following example, for instance, **page 2** of the program's logical memory is currently **stored in frame 3** of physical memory:



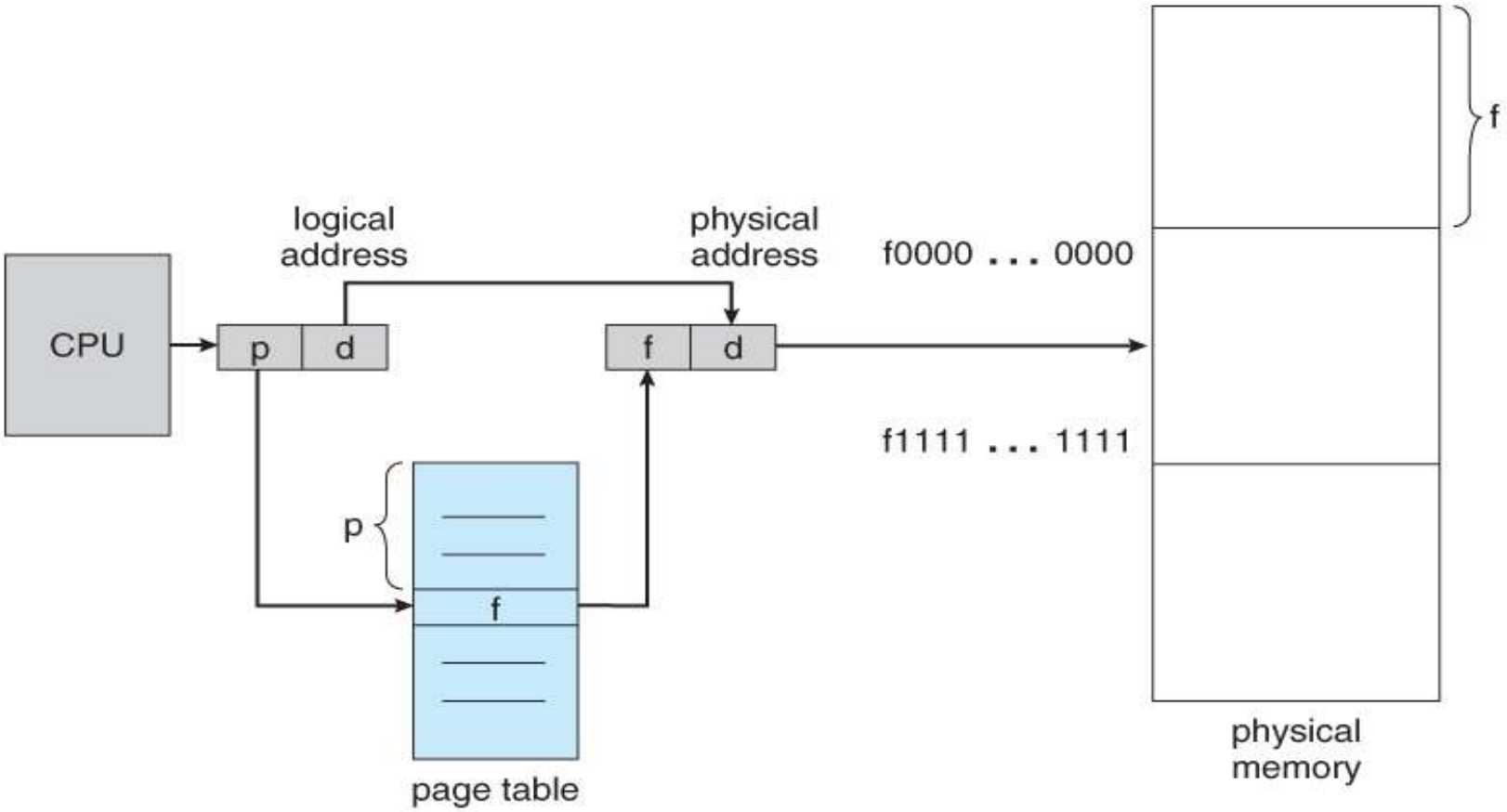
## Memory Management- Paging

- A **logical address** consists of two parts: A **page number** in which the address resides, and **an offset** from the beginning of that page.
  - The **number of bits in the page number** limits **how many pages** a single process can address.
  - The **number of bits in the offset** determines the **maximum size of each page**, and should correspond to the system frame size.



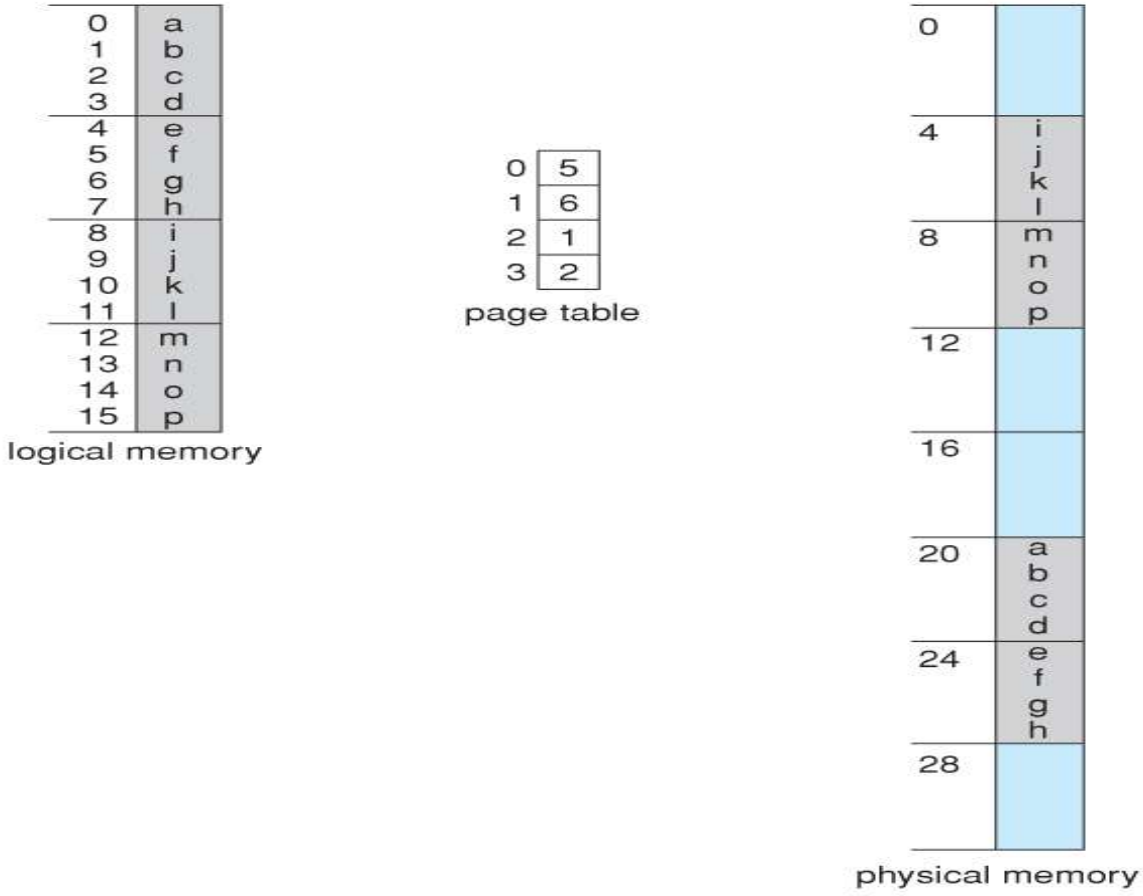

- The **page table** maps the **page number to a frame number**, to **yield a physical address** which also has two parts:
  - The **number of bits in the frame number** determines **how many frames** the system can address, and
  - the **number of bits in the offset** determines the **size of each frame**.

- **Address Translation**



## Memory Management- Paging

- Consider the following micro example, in which **a process has 16 bytes of logical memory**, mapped in **4 byte pages into 32 bytes of physical memory**.



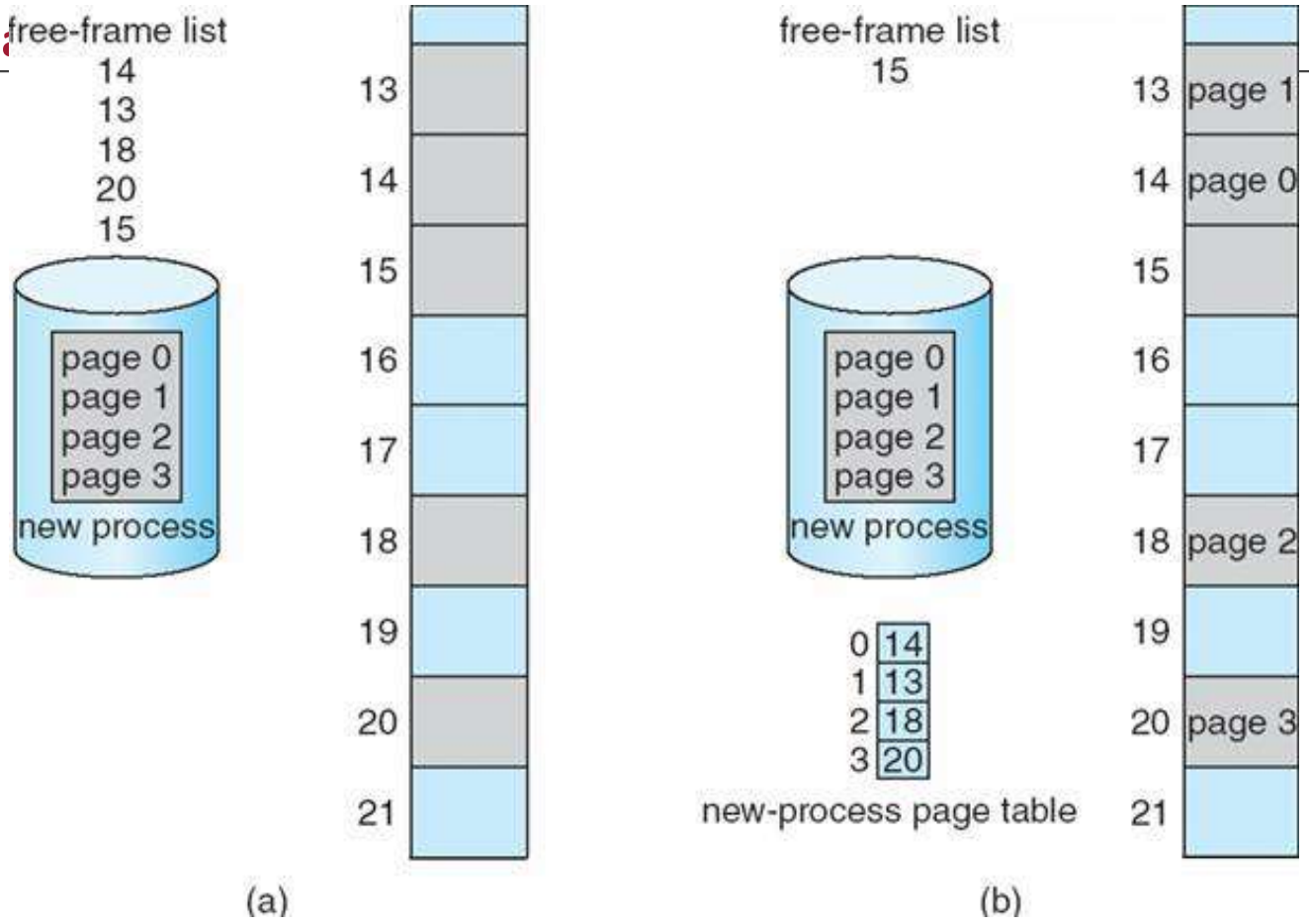
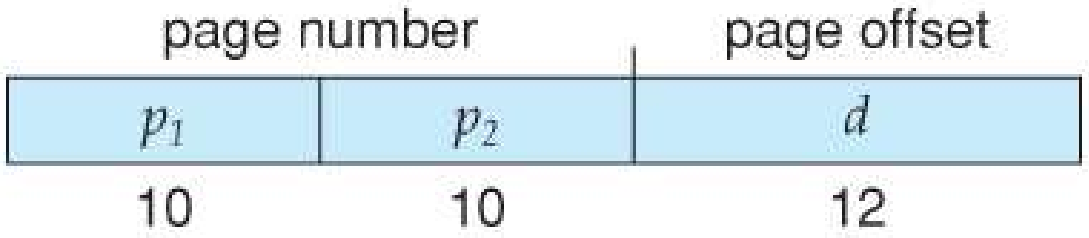


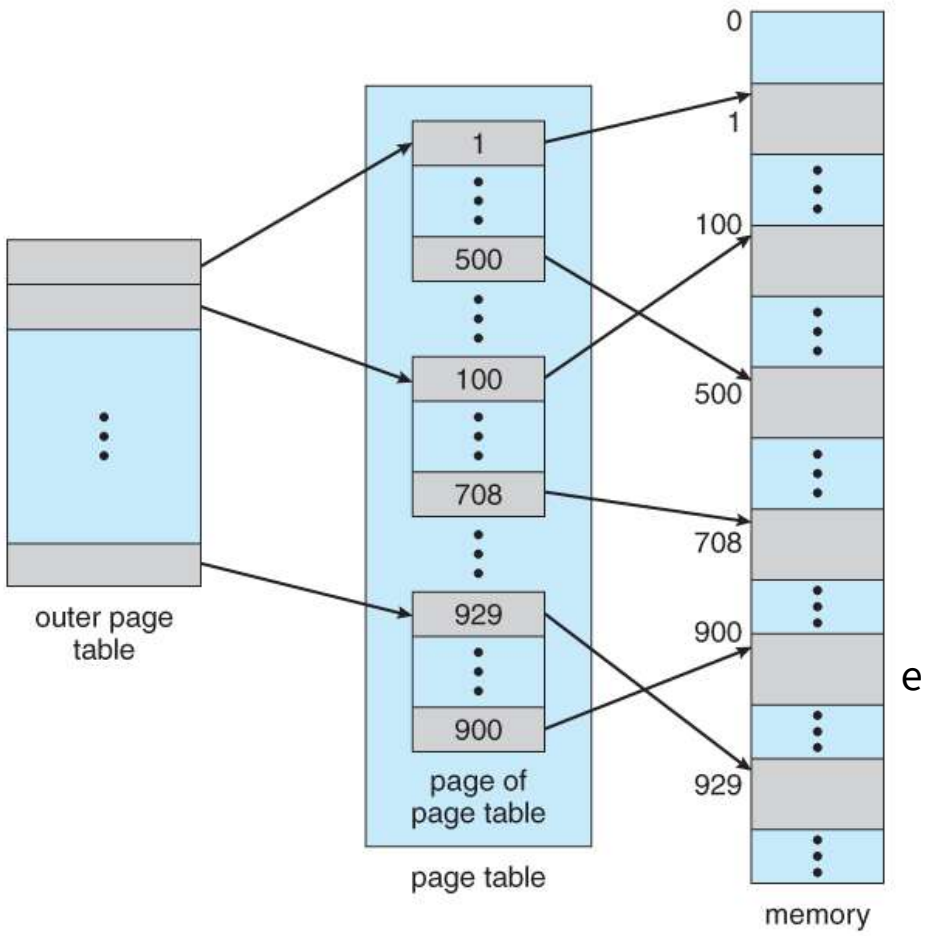

Figure: Free frames (a) before allocation and (b) after allocation

## Memory Management- Structure of the Page Table

- **Hierarchical Paging**
  - Most modern computer systems support **logical address spaces** of 2^32 to 2^64.
  - With a 2^32 address space and 4K ( 2^12 ) page sizes, this leave 2^20 entries in the page table.
  - At **4 bytes per entry, this amounts to a 4 MB** page table, which is too large to reasonably keep in contiguous memory.
  - Thus, with 4K pages, this would take **1024 pages just to hold the page table**!
  - One option is to use a **two-tier paging system**, i.e. to page the page table.



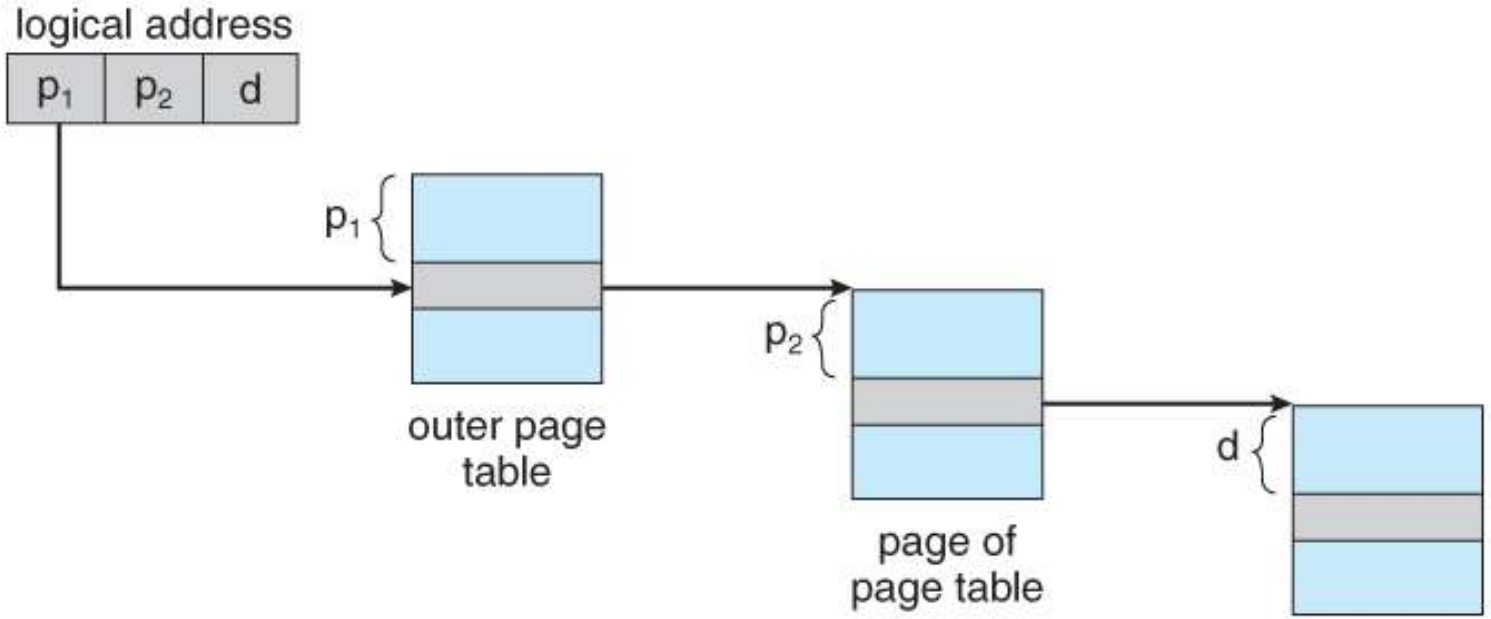
# Memory Management- Hierarchical Paging




e

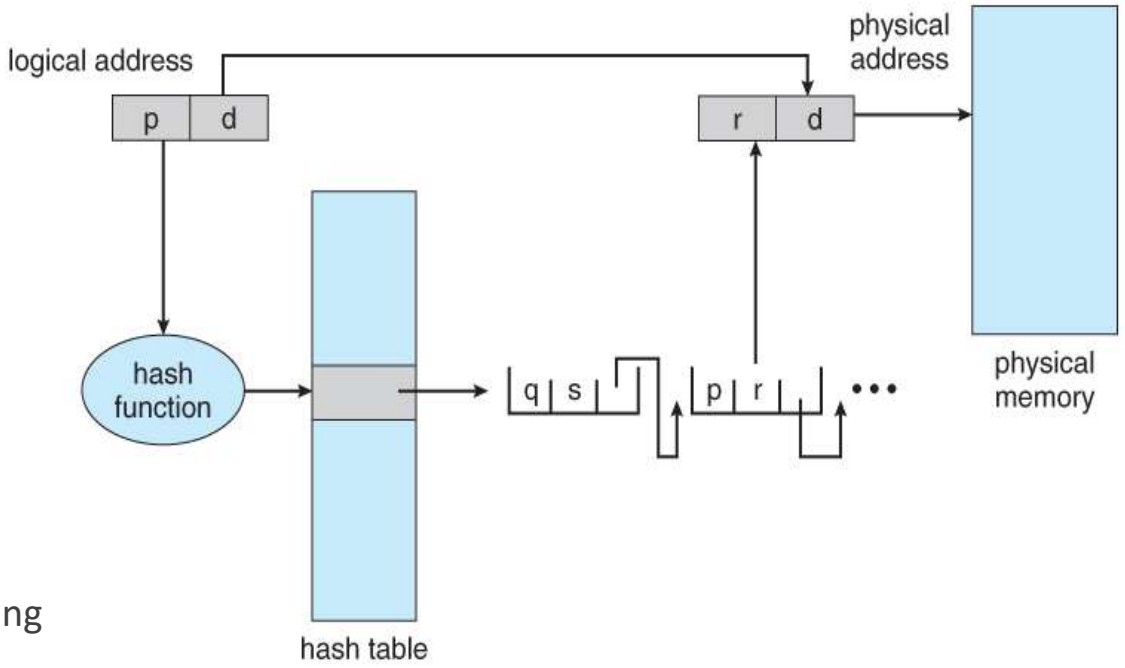
## Memory Management- Hierarchical Paging

- **Address Translation**



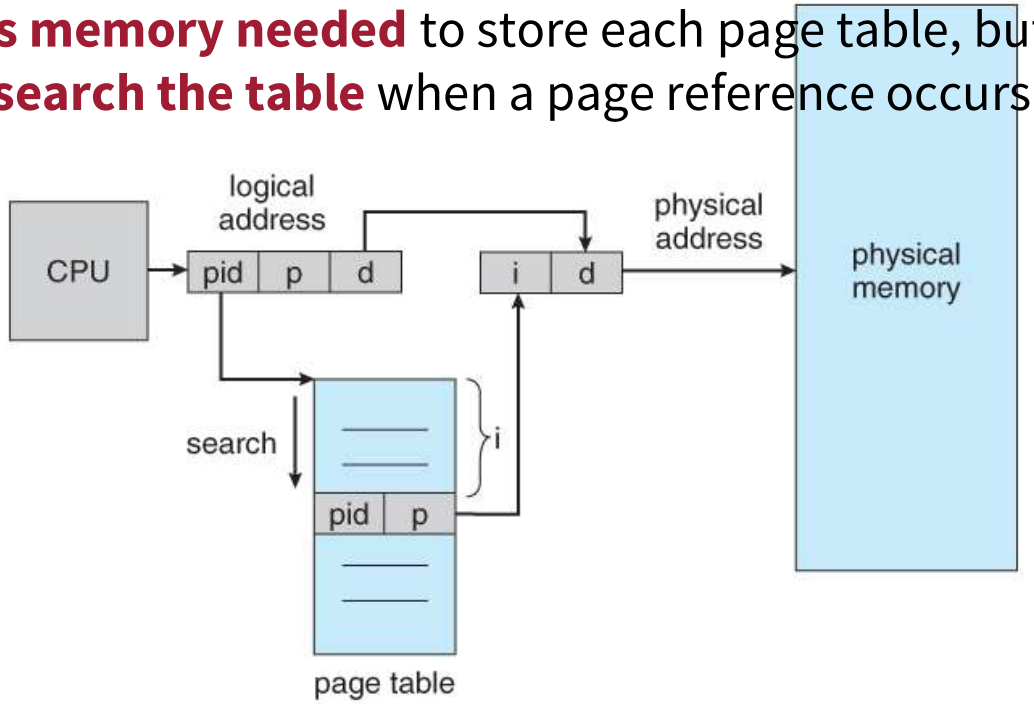
## Memory Management- Structure of the Page Table

- **Hashed Page Tables**
  - The **virtual page number** is **hashed** into a **page table**
    - This page table contains **a chain of elements hashing to the same location**
  - **Virtual page numbers** are **compared** in this chain searching **for a match**
    - If a **match is found**, the **corresponding physical frame is extracted**



## Memory Management- Structure of the Page Table

- **Inverted Page Tables**
  - Another approach is to use an ***inverted page table***.
  - Instead of a **table listing all of the pages** for a **particular process**, an inverted page table lists all of the **pages currently loaded in memory**, for all processes.
  - **Decreases memory needed** to store each page table, but **increases time needed to search the table** when a page reference occurs






D Y PATIL RAMRAO ADIK INSTITUTE OF TECHNOLOGY NAVI MUMBAI

# Lecture:

## Demand Paging

# Hardware and Control Structures

- Two characteristics fundamental to memory management:
    1) all memory references are logical addresses that are dynamically translated into physical addresses at run time
    2) a process may be broken up into a number of pieces that don't need to be contiguously located in main memory during execution
- If these two characteristics are present, it is not necessary that all of the pages or segments of a process be in main memory during execution

# Terminology

| | |
|---|---|
| **Virtual memory** | A storage allocation scheme in which secondary memory can be addressed as though it were part of main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites, and program-generated addresses are translated automatically to the corresponding machine addresses.The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of secondary memory available and not by the actual number of main storage locations. |
| **Virtual address** | The address assigned to a location in virtual memory to allow that location to be accessed as though it were part of main memory. |
| **Virtual address space** | The virtual storage assigned to a process. |
| **Address space** | The range of memory addresses available to a process. |
| **Real address** | The address of a storage location in main memory. |

## Execution of a Process

- Operating system brings into main memory a few pieces of the program
- Resident set - portion of process that is in main memory
- An interrupt is generated when an address is needed that is not in main memory (segment/page fault)
- Operating system places the process  in a blocking state

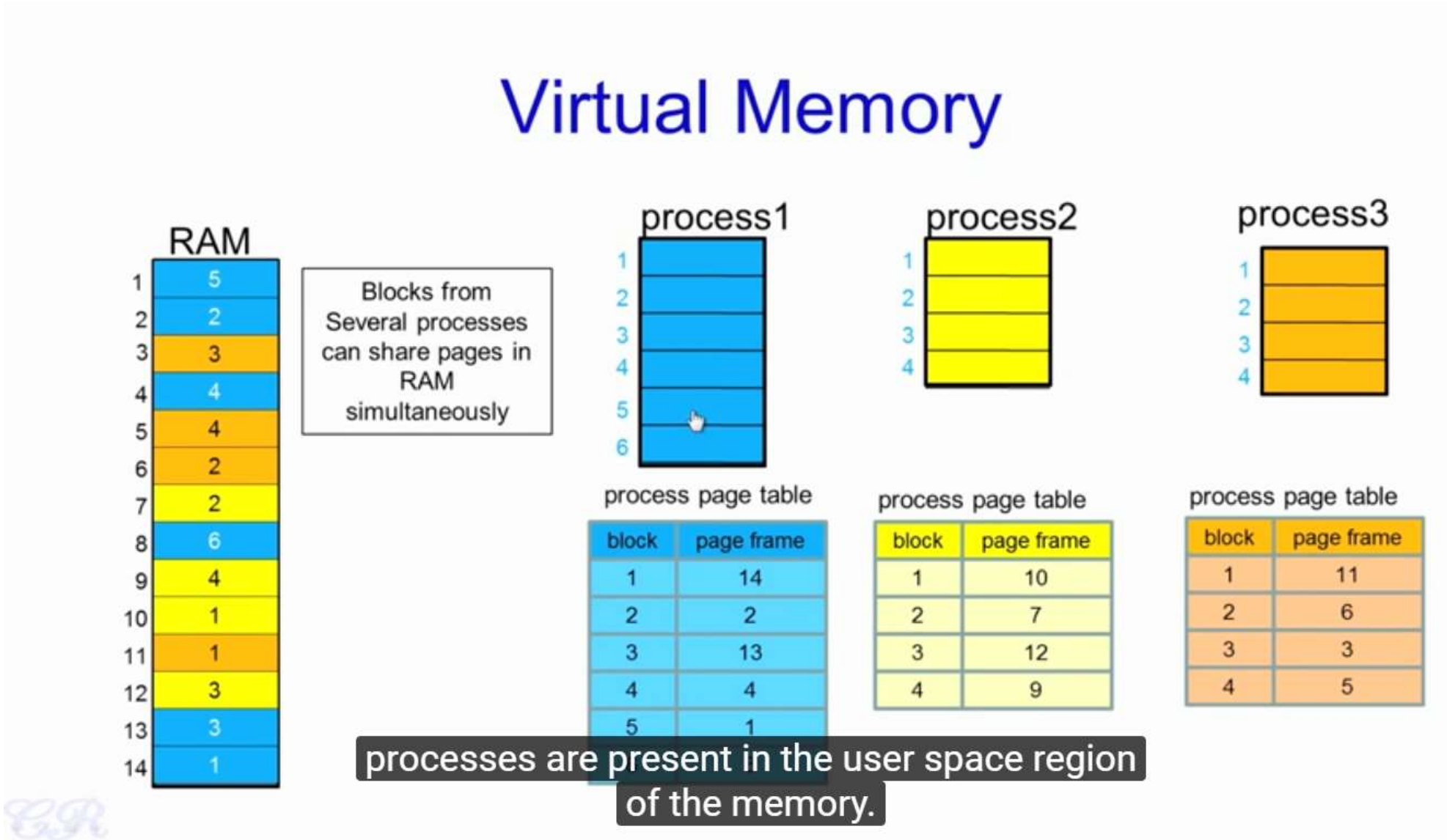
## Execution of a Process

- Piece of process that contains the logical address is brought into main memory
  - operating system issues a disk I/O Read request
  - another process is dispatched to run while the disk I/O takes place
  - an interrupt is issued when disk I/O is complete, which causes the operating system to place the affected process in the Ready state

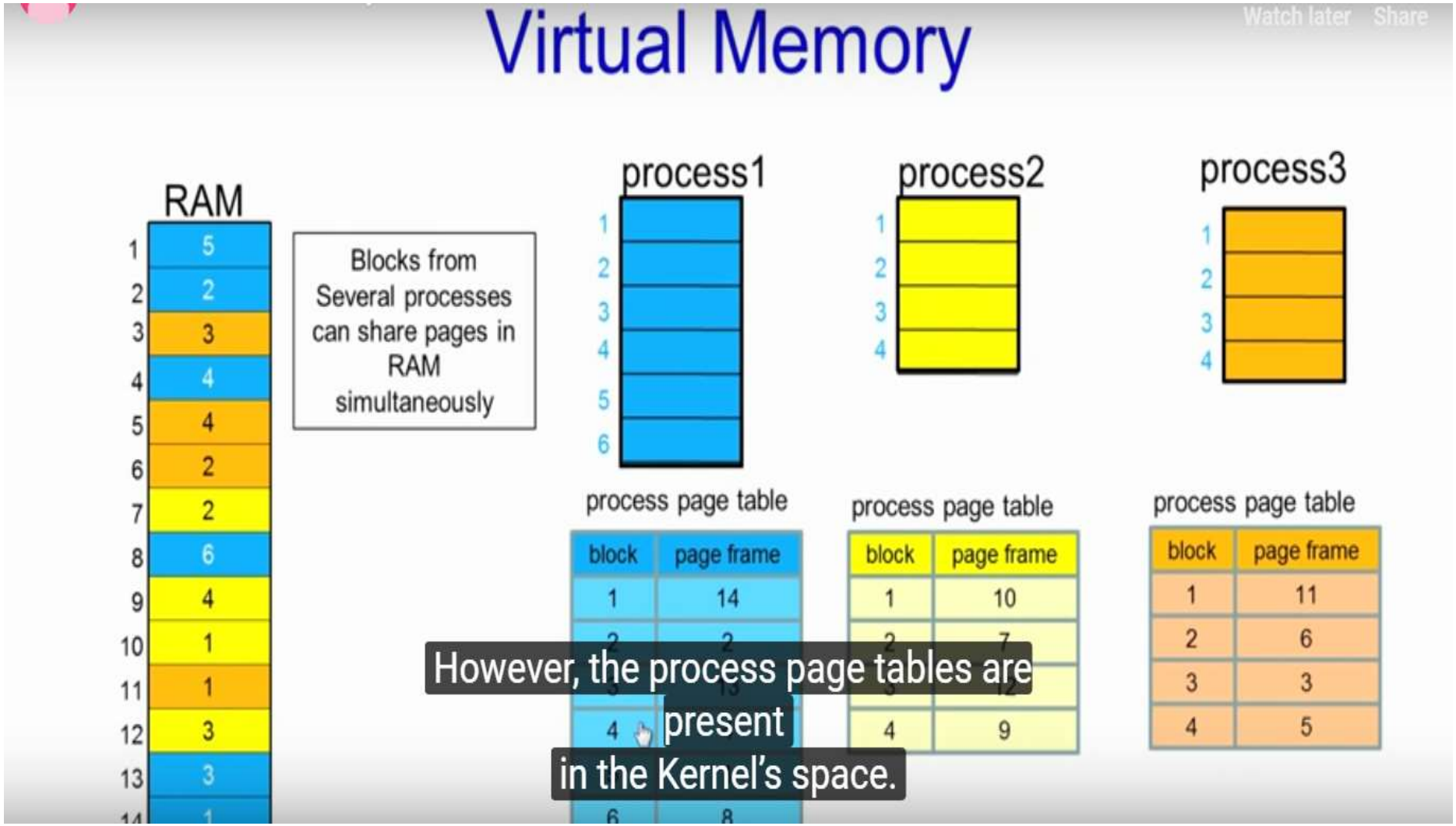
## Support Needed for Virtual Memory

**For virtual memory to be practical and effective:**

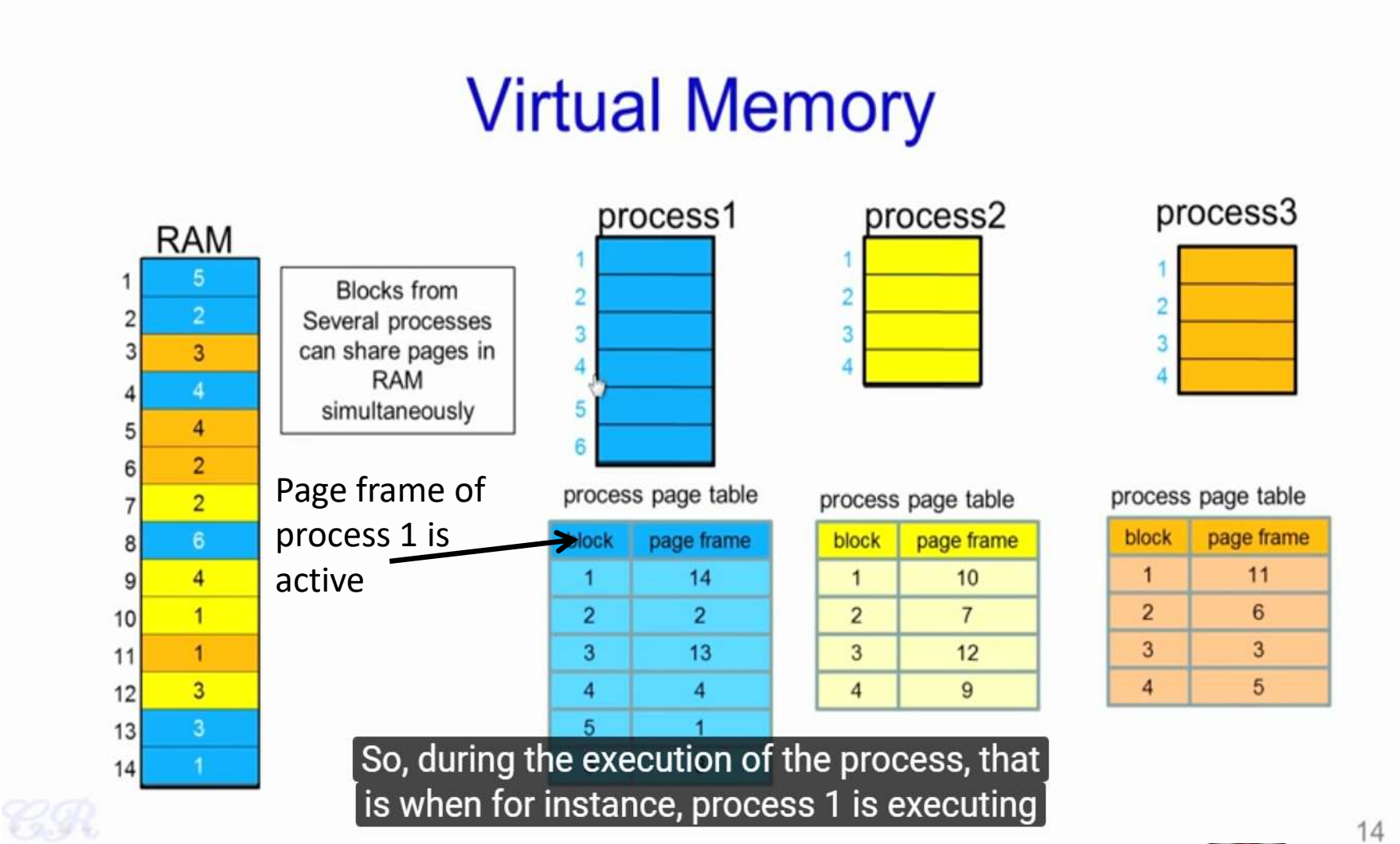
- hardware must support paging and segmentation
- operating system must include software for managing the movement of pages and/or segments between secondary memory and main memory

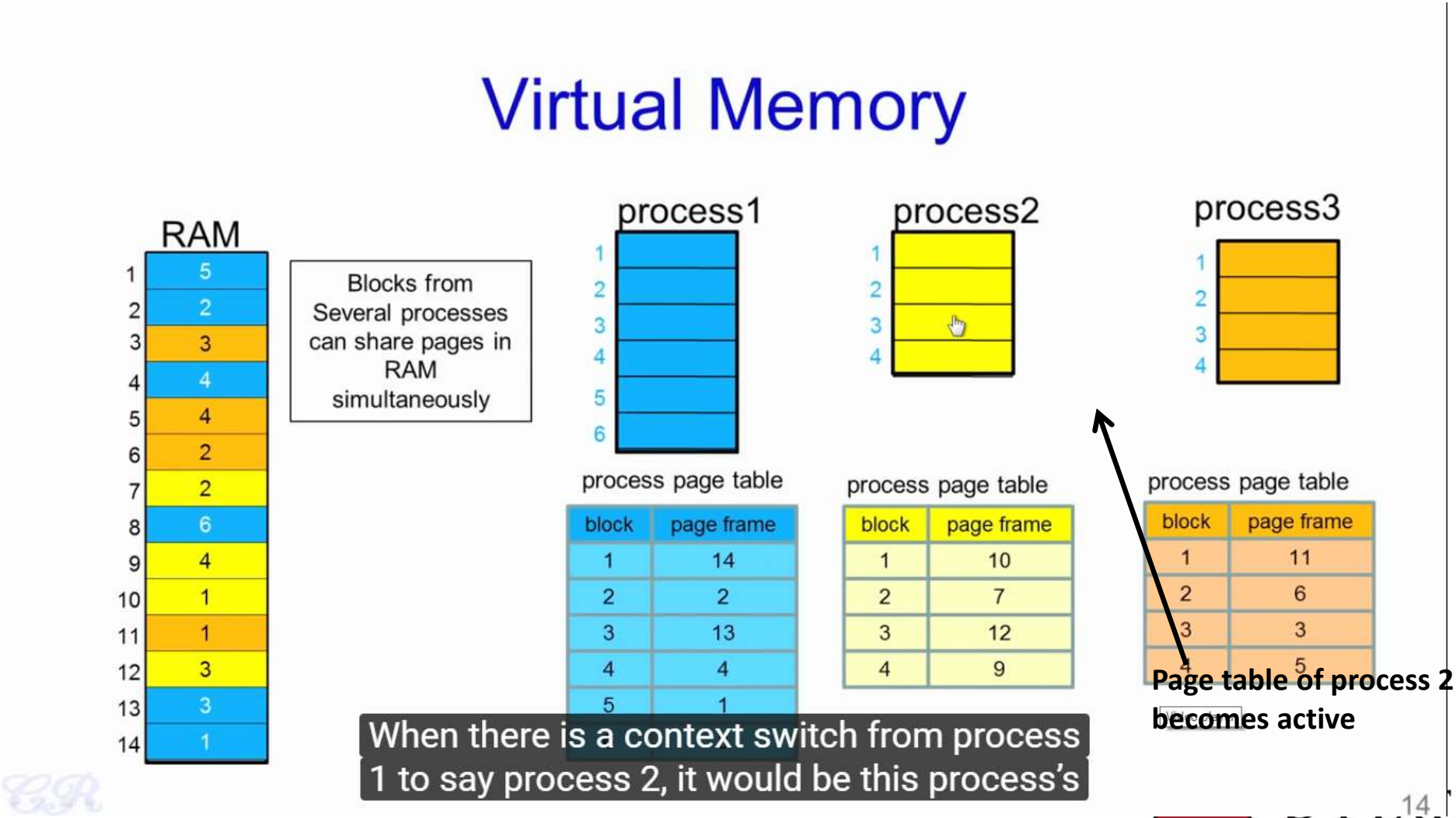

Virtual Memory
RAM
1 5
2 2
3 3
4 4
5 4
6 2
7 2
8 6
9 4
10 1
11 1
12 3
13 3
14 1
Blocks from Several processes can share pages in RAM simultaneously
process1
1
2
3
4
5
6
process2
1
2
3
4
process3
1
2
3
4
process page table
block | page frame
1 | 14
2 | 2
3 | 13
4 | 4
5 | 1
process page table
block | page frame
1 | 10
2 | 7
3 | 12
4 | 9
process page table
block | page frame
1 | 11
2 | 6
3 | 3
4 | 5
processes are present in the user space region of the memory.

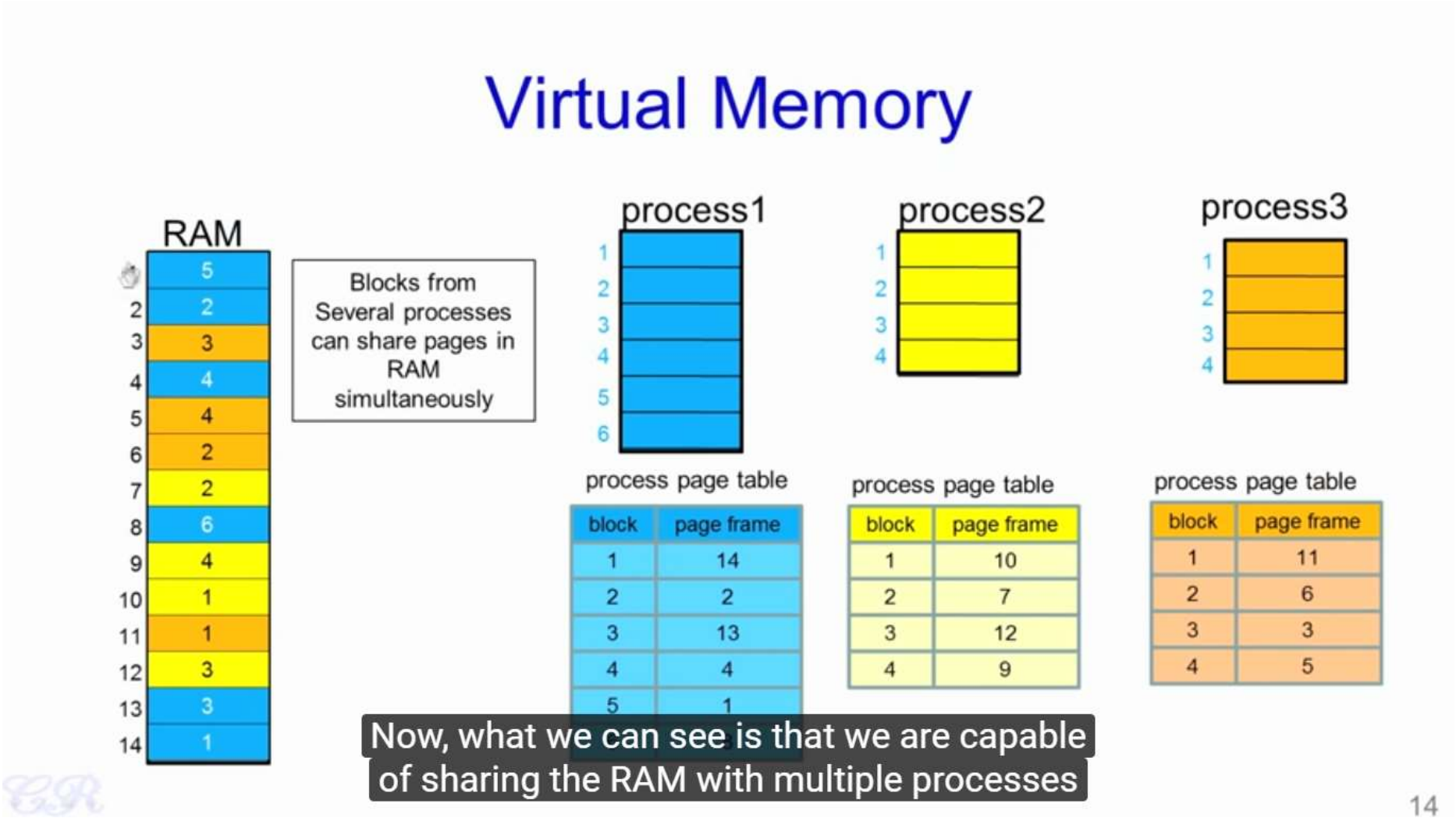
# Virtual Memory



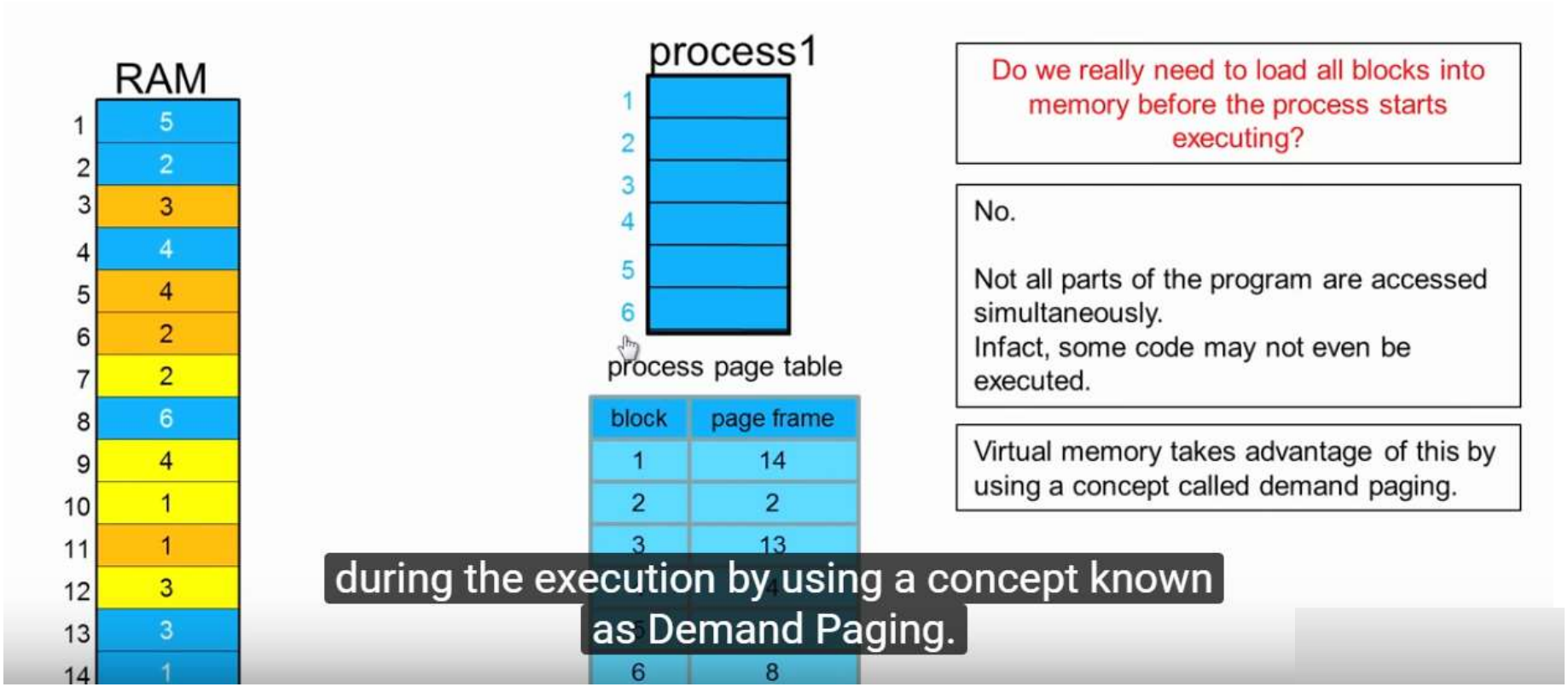
# Virtual Memory





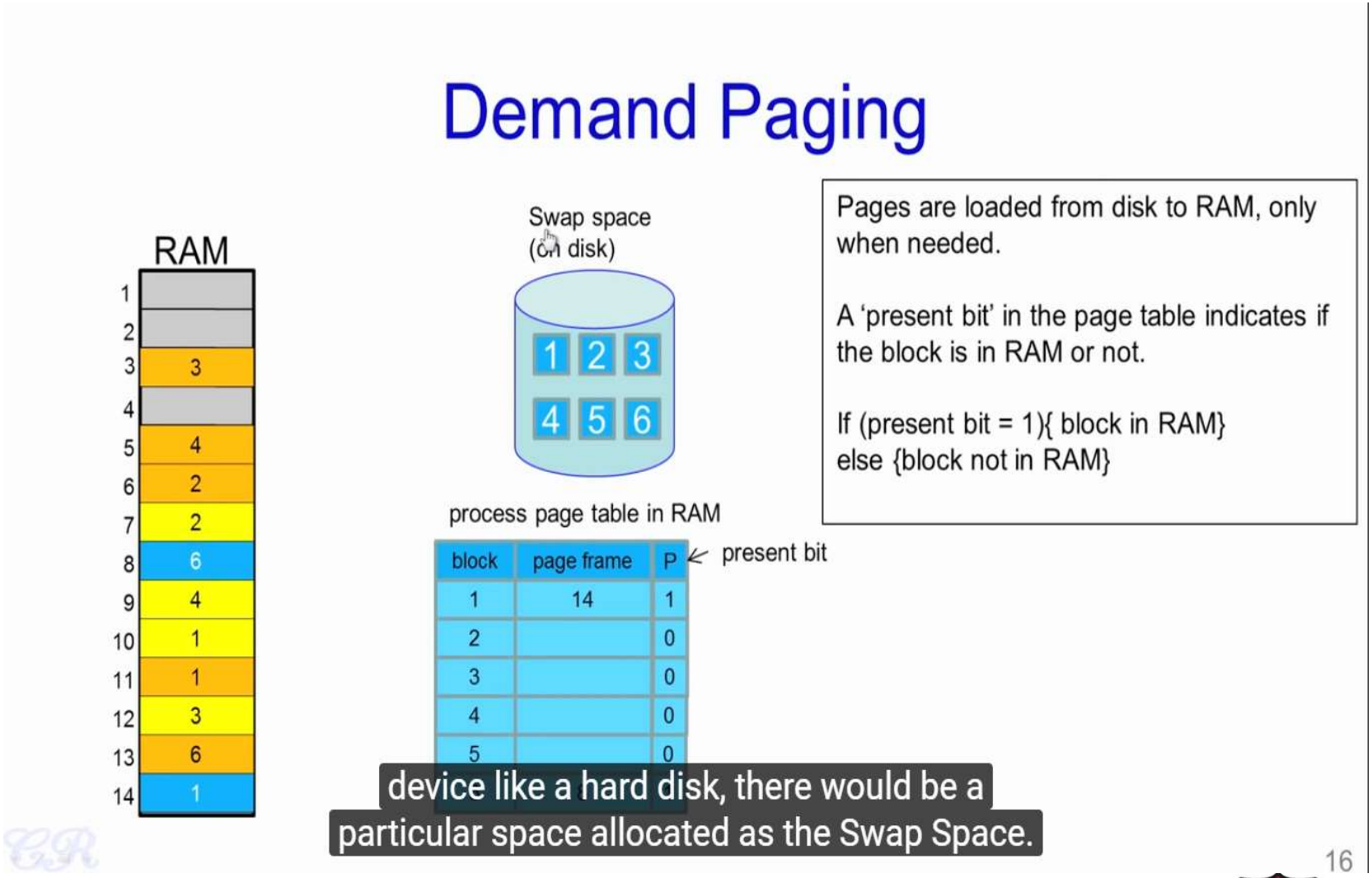


# Virtual Memory

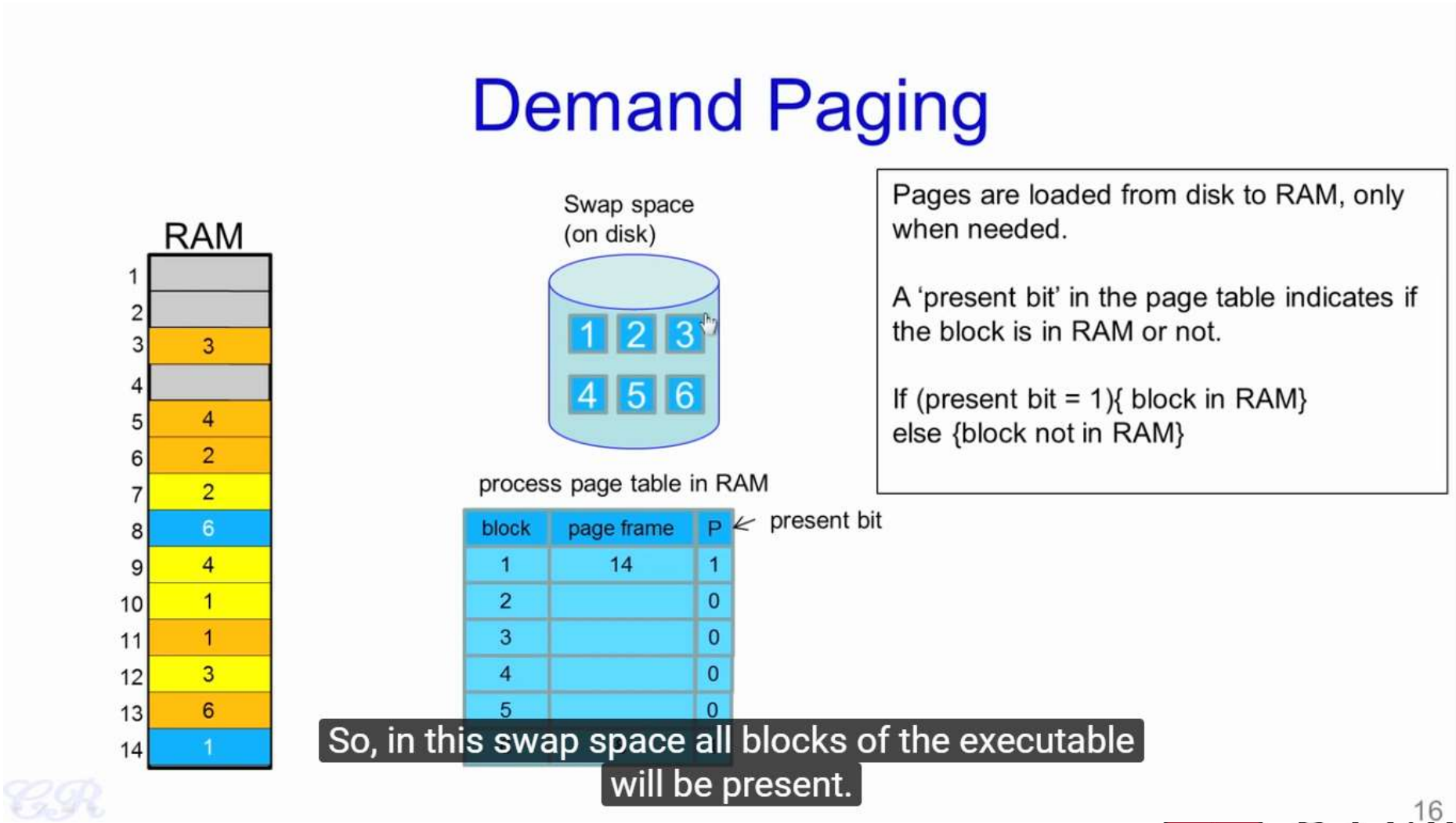


## Demand Paging

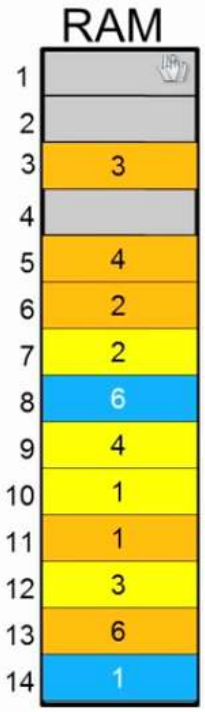
- Bring a page into memory only when it is needed
  - Less I/O needed
  - Less memory needed
  - Faster response (no need to wait for all pages to load)
  - More users
- Page is needed ⇒ reference to it
  - invalid reference ⇒ abort
  - not-in-memory ⇒ bring to memory
- **Lazy swapper** – never swaps a page into memory unless page will be needed
  - Swapper that deals with pages is a **pager**

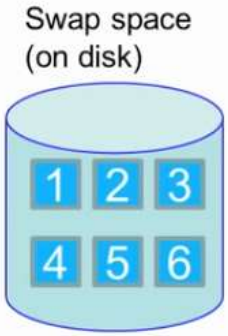


# Demand Paging



# Demand Paging




process page table in RAM

| block | page frame | P |
|---|---|---|
| 1 | 14 | 1 |
| 2 | | 0 |
| 3 | | 0 |
| 4 | | 0 |
| 5 | | 0 |

present bit

Pages are loaded from disk to RAM, only when needed.

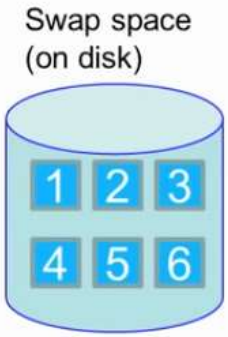
A 'present bit' in the page table indicates if the block is in RAM or not.

If (present bit = 1){ block in RAM}
else {block not in RAM}

And only on demand, will blocks be loaded from the swap space into the RAM.

16

# Demand Paging


Pages are loaded from disk to RAM, only when needed.

A 'present bit' in the page table indicates if the block is in RAM or not.

If (present bit = 1){ block in RAM}
else {block not in RAM}

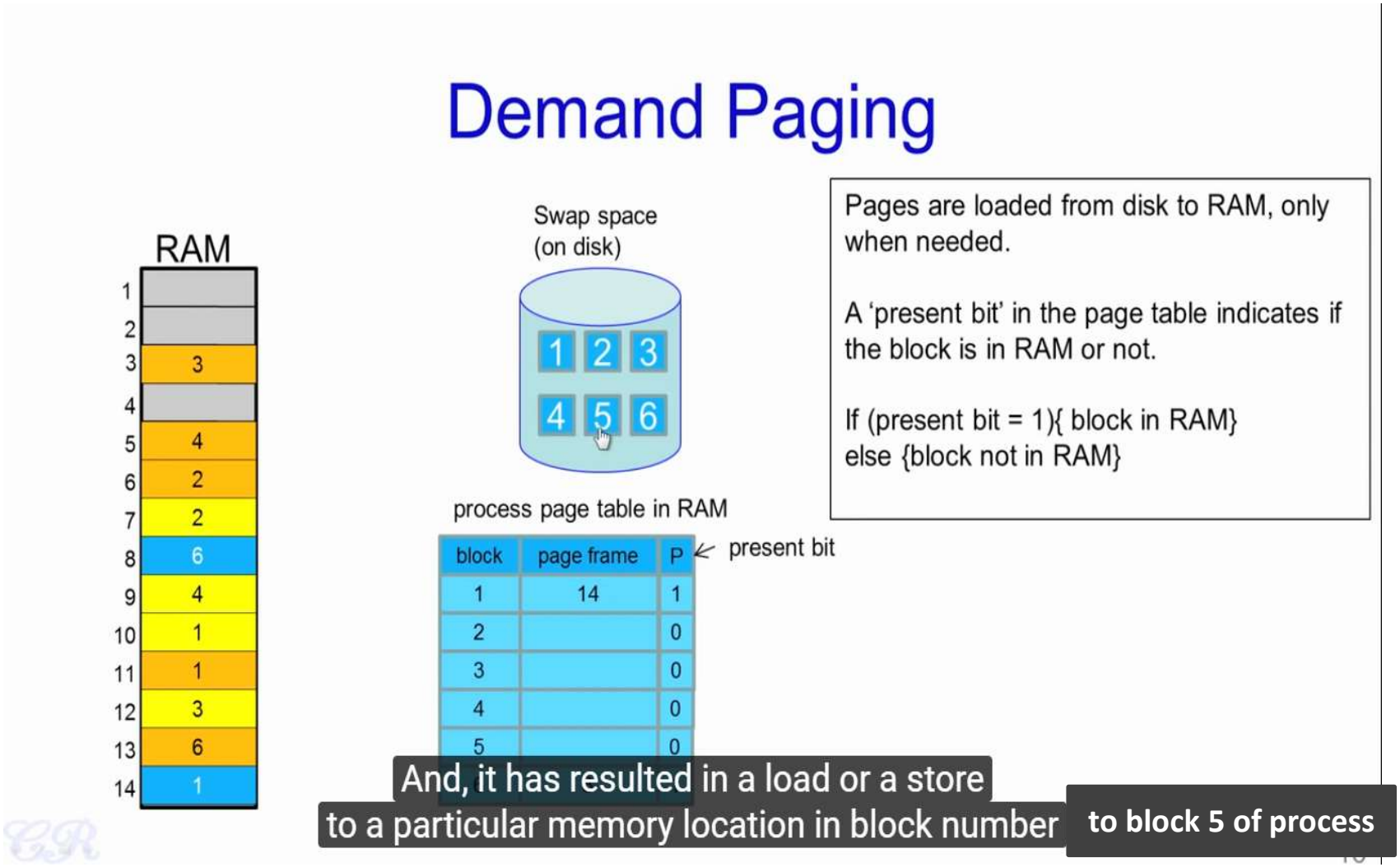
process page table in RAM

| block | page frame | P |
|---|---|---|
| 1 | 14 | 1 |
| 2 | | 0 |
| 3 | | 0 |
| 4 | | 0 |
| 5 | | 0 |

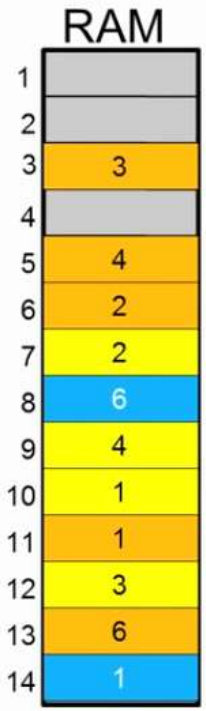
present bit

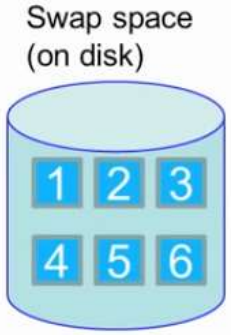
So, now let us assume that there is an instruction which is executing in page frame 8 that corresponds to block 6 of process

# Demand Paging



# Demand Paging




Pages are loaded from disk to RAM, only when needed.

A 'present bit' in the page table indicates if the block is in RAM or not.

If (present bit = 1){ block in RAM}
else {block not in RAM}

process page table in RAM

| block | page frame | P |
|---|---|---|
| 1 | 14 | 1 |
| 2 | | 0 |
| 3 | | 0 |
| 4 | | 0 |
| 5 | | 0 |

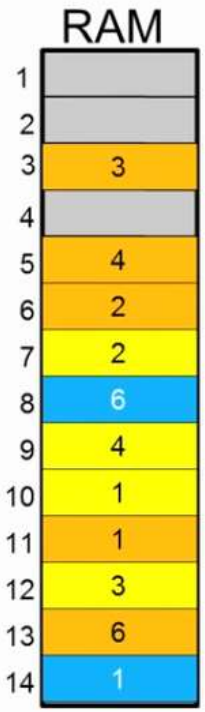
present bit

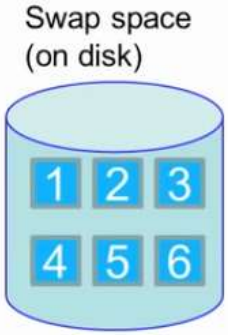
to block number 5 and see that the present bit is set to 0.

16

So, this would result in something known as a Page fault trap, also called as the Page fault interrupt

# Demand Paging




Pages are loaded from disk to RAM, only when needed.

A 'present bit' in the page table indicates if the block is in RAM or not.

If (present bit = 1){ block in RAM}
else {block not in RAM}

process page table in RAM

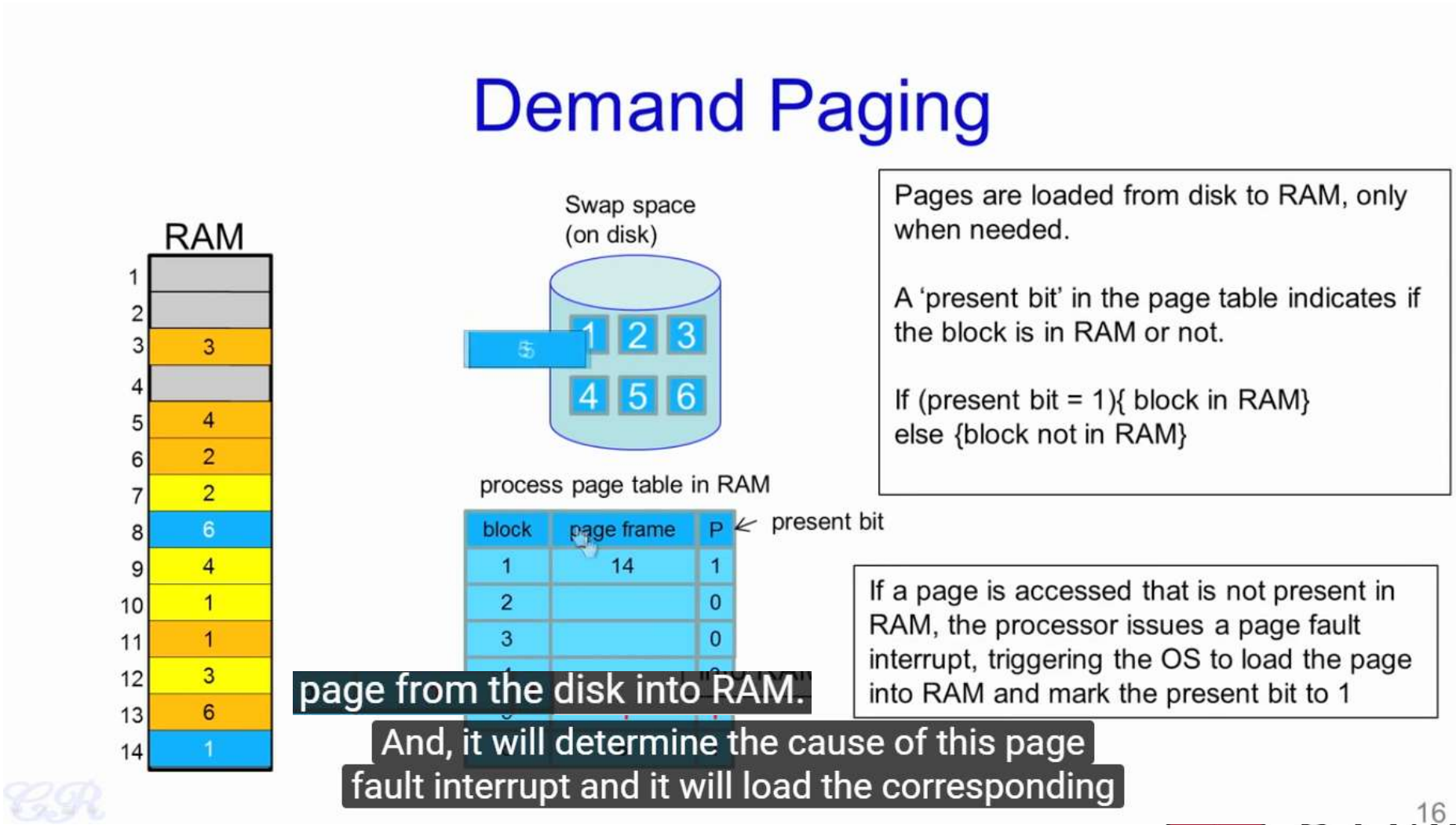
| block | page frame | P |
|---|---|---|
| 1 | 14 | 1 |
| 2 | | 0 |

present bit

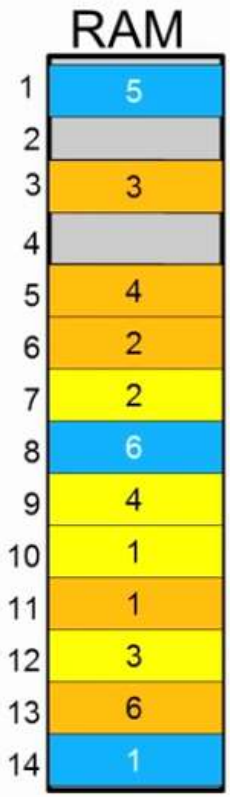
Now, the operating system will then detect that this was a page fault interrupt.
Now, the page fault interrupt would then trigger the operating system to execute.

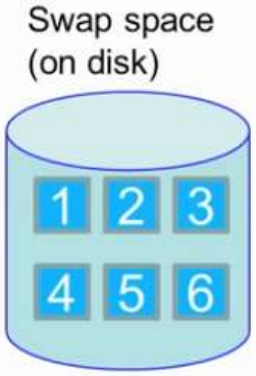
16

# Demand Paging



# Demand Paging




Pages are loaded from disk to RAM, only when needed.

A 'present bit' in the page table indicates if the block is in RAM or not.

If (present bit = 1){ block in RAM}
else {block not in RAM}

process page table in RAM

| block | page frame | P |
|---|---|---|
| 1 | 14 | 1 |
| 2 | | 0 |
| 3 | | 0 |
| 4 | | 0 |
| 5 | 1 | 1 |

present bit

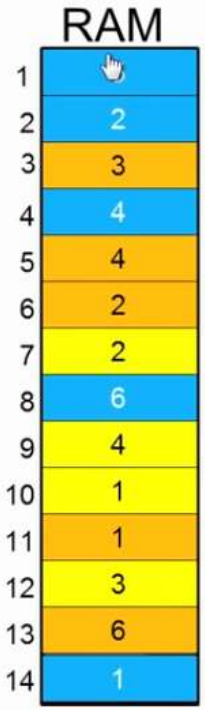
If a page is accessed that is not present in RAM, the processor issues a page fault interrupt, triggering the OS to load the page into RAM and mark the present bit to 1

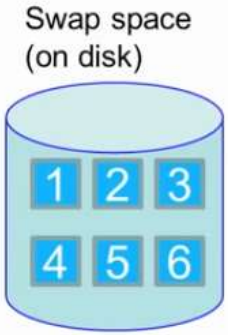
Consequently, it will also modify the process's page table.

Now, all later accesses to this particular block 5 will not cause a page fault because

16

# Demand Paging




process page table in RAM

| block | page frame | P |
|---|---|---|
| 1 | 14 | 1 |
| 2 | 2 | 1 |
| 3 | | 0 |
| 4 | 4 | 1 |
| 5 | 1 | 1 |

If there are no pages free for a new block to be loaded, the OS makes a decision to remove another block from RAM.

This is based on a replacement policy, implemented in the OS.

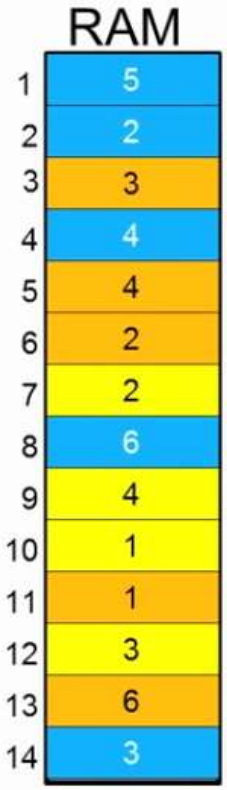
Some replacement policies are
* First in first out
* Least recently used
* Least frequently used

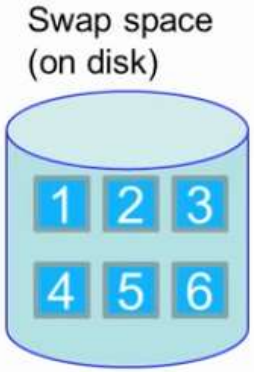
The replaced block **may** need to be written back to the swap (swap out).

So now we have reached a state where every page frame in the RAM occupies some block.

17

# Demand Paging




process page table in RAM

| block | page frame | P | D |
|---|---|---|---|
| 1 | 14 | 0 | 1 |
| 2 | 2 | 1 | 1 |
| 3 | 14 | 1 | 0 |
| 4 | 4 | 1 | 1 |
| 5 | 1 | 1 | 0 |
| 6 | 8 | 1 | 1 |

D-bit or the dirty bit.

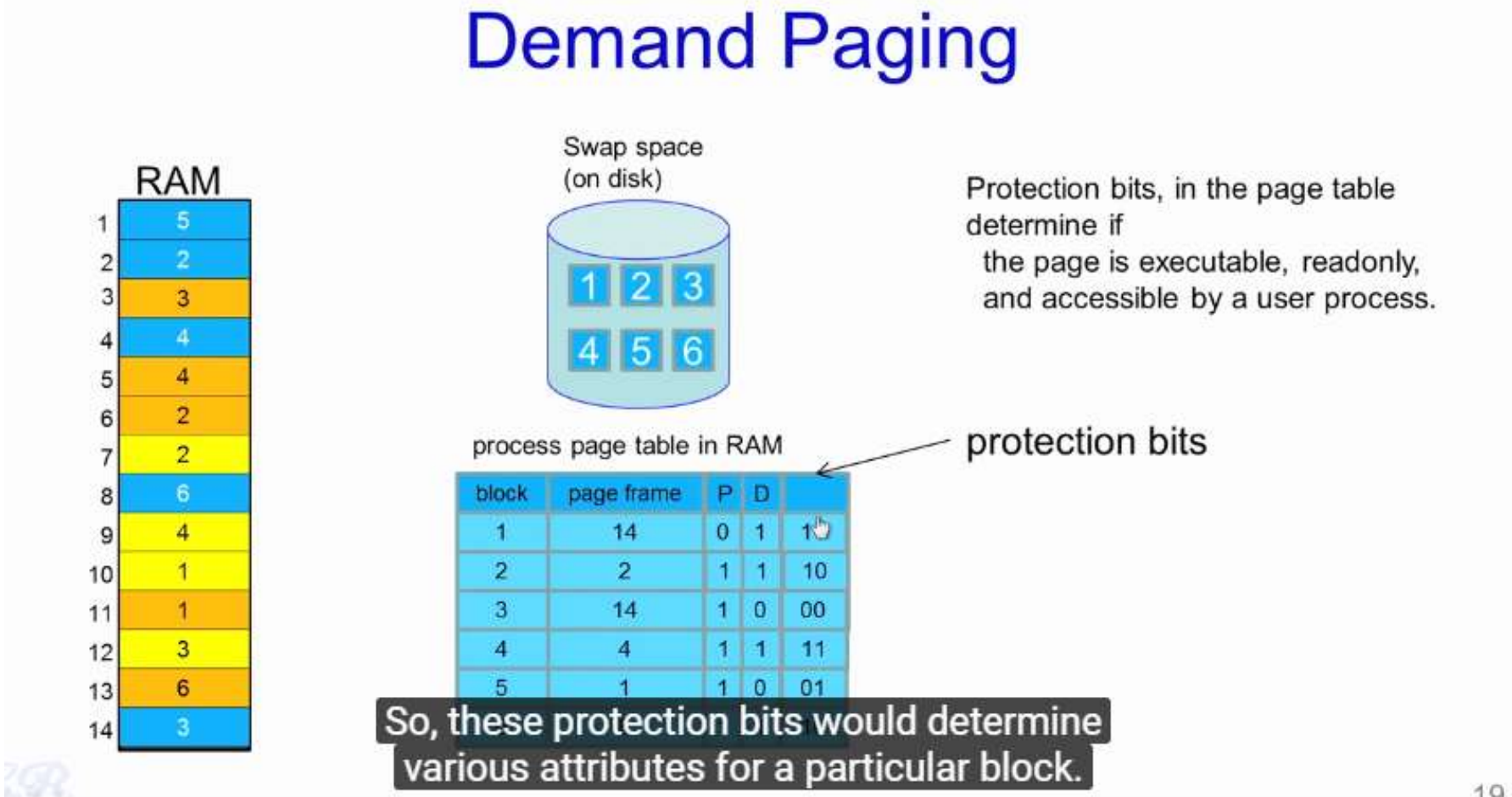
The **dirty bit**, in the page table indicates if a page needs to be written back to disk

If the dirty bit is 1, indicates the page needs to be written back to disk.

So, the dirty bit essentially indicates whether the contents of a block in the RAM has been modified with respect to its contents in the disk or in the swap space.

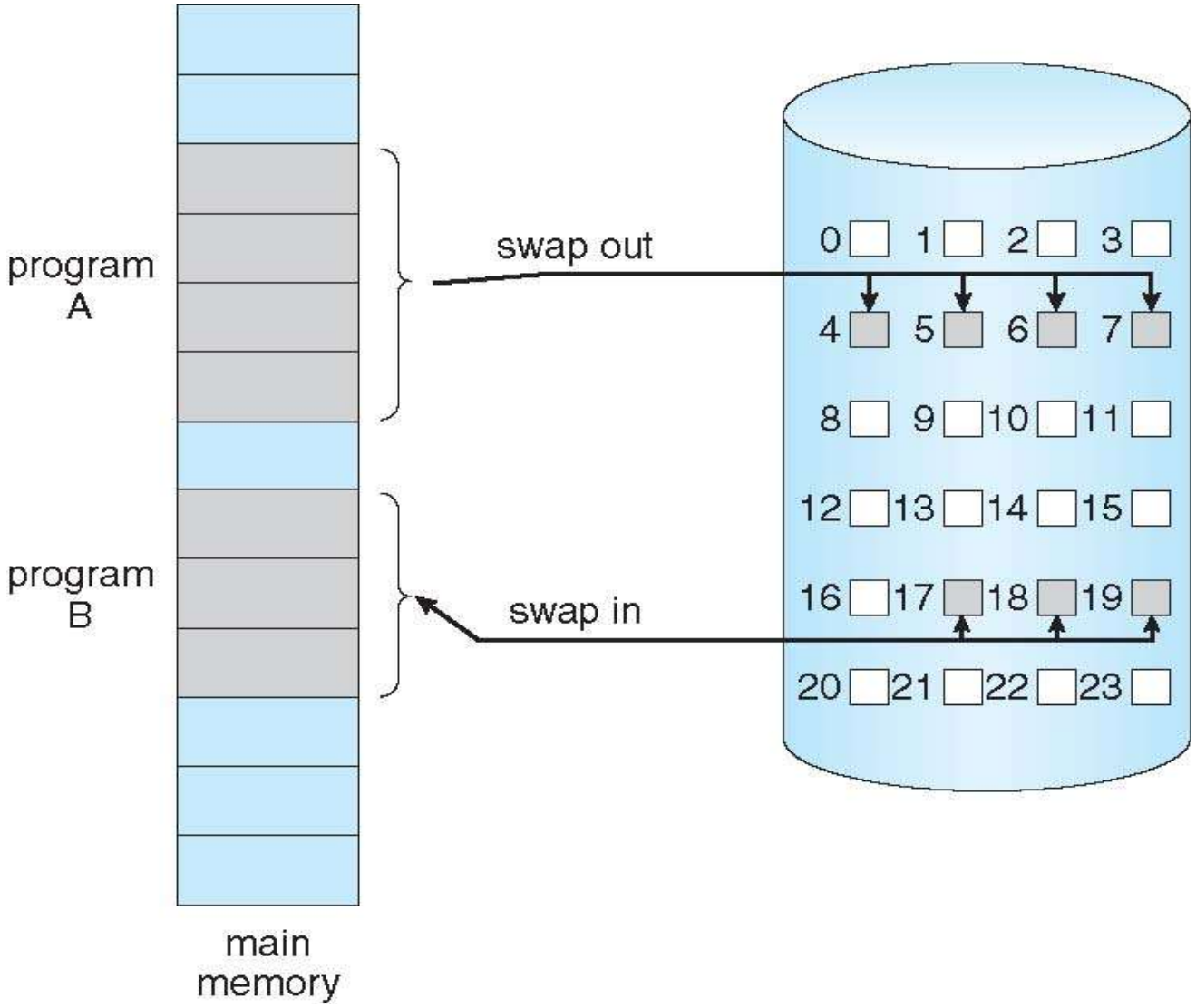
18

# Demand Paging

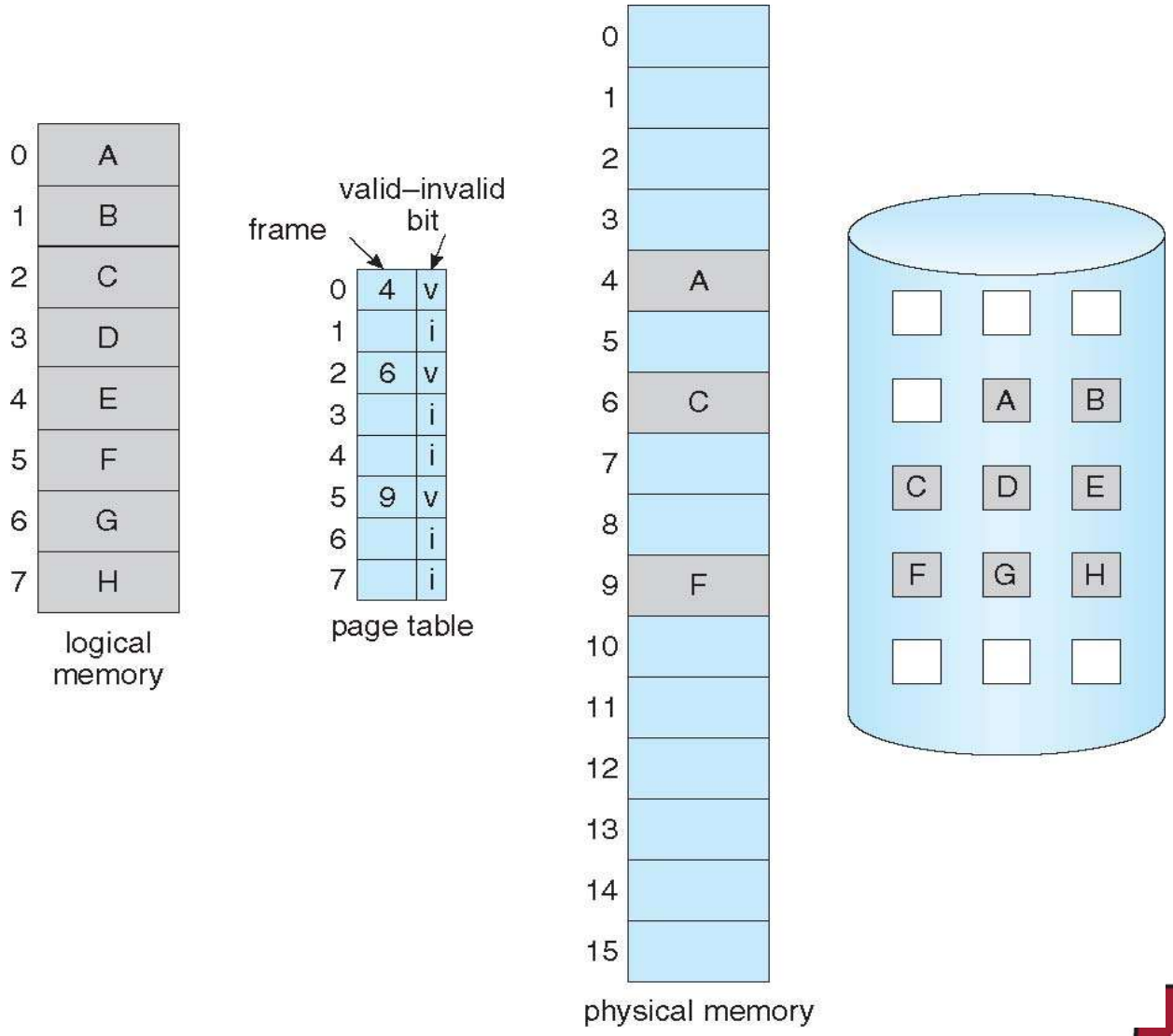



Attributes such as read only, executable, read/write

# Transfer of a Paged Memory to Contiguous Disk Space



# Page Table When Some Pages Are Not in Main Memory



## Page Fault

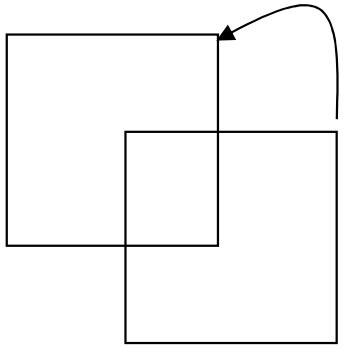
- If there is a reference to a page, first reference to that page will trap to operating system:

  **page fault**

1. Operating system looks at another table to decide:
   - Invalid reference ⇒ abort
   - Just not in memory
2. Get empty frame
3. Swap page into frame
4. Reset tables
5. Set validation bit = **v (or present bit p=1)**
6. Restart the instruction that caused the page fault

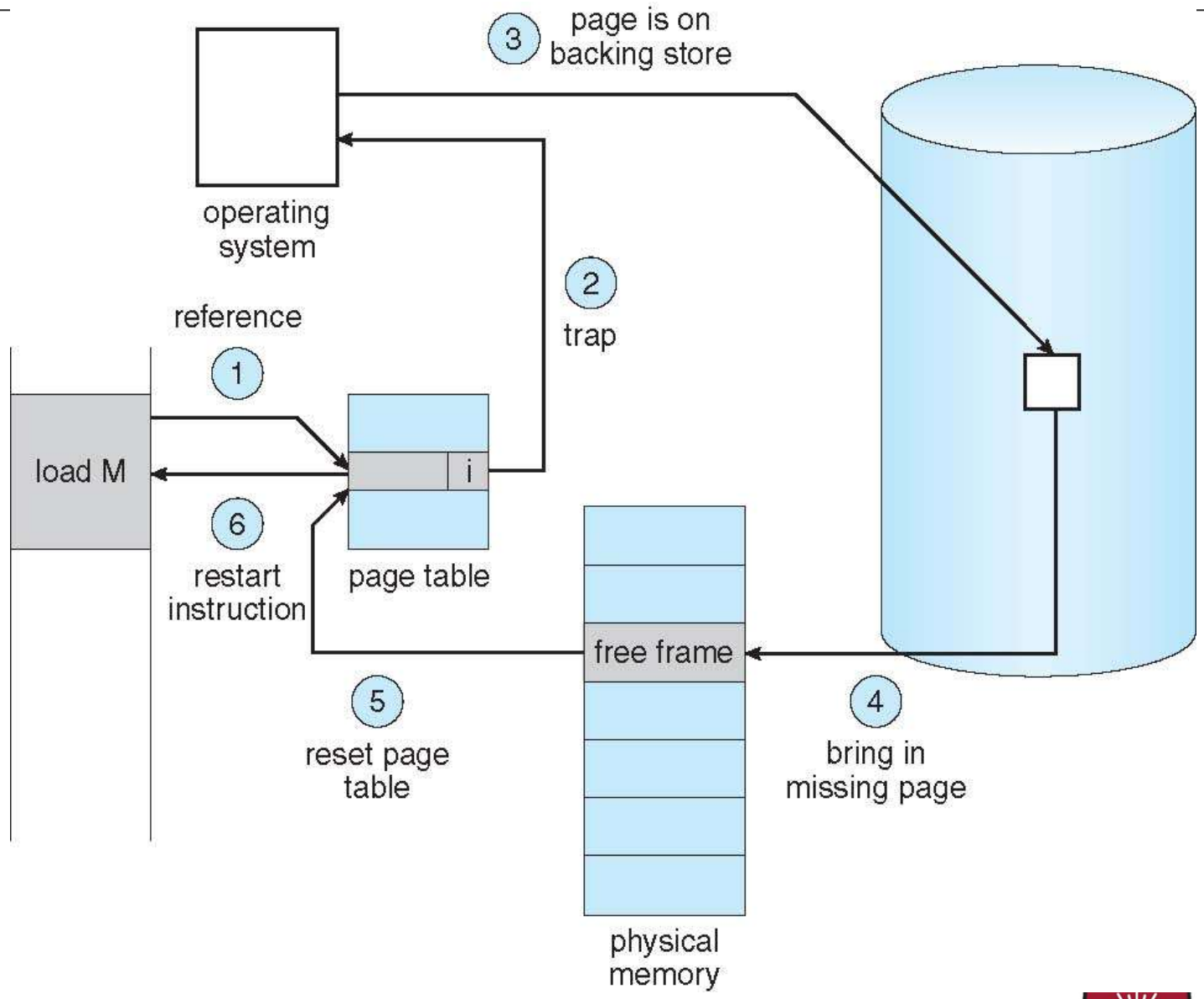
## Page Fault (Cont.)

- Restart instruction
  - block move



  - auto increment/decrement location

# Steps in Handling a Page Fault



## Page Fault Service Time

- Whenever a page fault occurs, the CPU takes considerable amount of time to perform the above mentioned steps. This time taken by the CPU **is called as page fault service time.**
    - Access time for Main memory : m
    - Service time for Page fault: s
    - Rate for Page fault Page fault rate is : p
    - **Effective memory access time = (p*s) + (1-p)*m**

## Performance of Demand Paging

- Page Fault Rate $0 \leq p \leq 1.0$
  - if $p = 0$ no page faults
  - if $p = 1$, every reference is a fault
- Effective Access Time (EAT)

$$\begin{aligned} \text{EAT} = (1 - p) \times \text{memory access} \\ + p\ (\text{page fault overhead} \\ + \text{swap page out} \\ + \text{swap page in} \\ + \text{restart overhead} \\ ) \end{aligned}$$

## Demand Paging Example

- 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!!

To keep slowdown to 10% p < 0.0000025 (1 in ~400K)

## Copy-on-Write

- Copy-on-Write (COW) allows both parent and child processes to initially *share* the same pages in memory
  If either process modifies a shared page, only then is the page copied
- COW allows more efficient process creation as only modified pages are copied
- Free pages are allocated from a **pool** of zeroed-out pages

# **Lecture:** Page Replacement Strategies: FIFO, Optimal, LRU

## Memory Management- Page Replacement

- In order to make the **most use of virtual memory**, we load several processes into memory at the same time.

- Since we only load the pages that **are actually needed by each process** at any given time, there is room to load many more processes than if we had to load in the entire process.

- **Find some page in memory that isn't being used right now, and swap that page only out to disk, freeing up a frame that can be allocated to the process requesting it. This is known as page replacement, and is the most common solution.**

## Basic Page Replacement

- Find the **location of the desired page on the disk**, either in swap space or in the file system.
  - **Find a free frame: If there is a free frame, use it.**
  - If there is no free frame, **use a page-replacement algorithm to select an existing frame to be replaced, known as the victim frame**.
  - Write the victim frame to disk. Change page tables to indicate that this **page is no longer in memory.**
  - Read in the **desired page and store it in the frame**. Adjust all related page and frame tables to indicate the change.
  - **Restart the process that** was waiting for this page.

## Page Replacement Algorithms

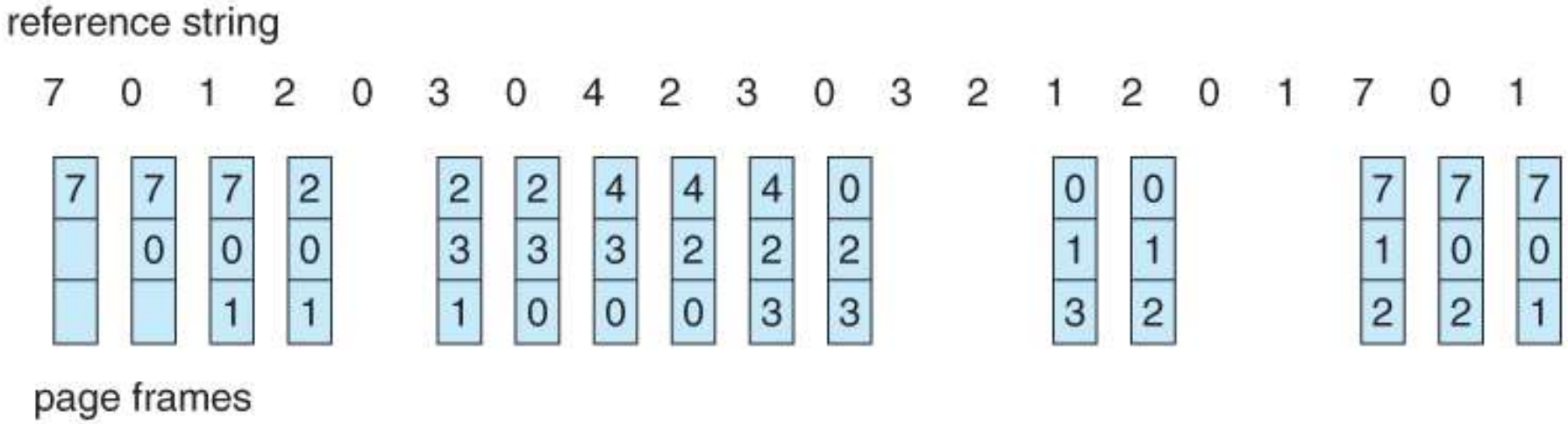
- We want to have the **lowest page-fault rate**.
- **Evaluate algorithms** by running it on a particular string of memory references and
- Compute the **number of page faults** on that string.

## Algorithms:

- **FIFO**: First-In-First-Out
- **Optimal** (replace page that will ***not be used for longest time*** in the future).
- **LRU**: Least Recently Used.
- **LFU**: Least frequently used,
- **MFU**: Most Frequently used.

## FIFO Page Replacement

- A simple and obvious page replacement strategy is ***FIFO***, i.e. first-in-first-out.
- As **new pages** are brought in, they are **added to the tail of a queue**, and the page at the **head of the queue is the next victim**. In the following example, 20 page requests result in 15 page faults:
- **FIFO** suffers from **belady's anamoly**



## First In First Out:-

- Incoming page Steam – 7, 0, 1, 2, 0 , 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
- Number of frames = 3

| **Incoming Page Stream ->** | **7** | **0** | **1** | **2** | **0** | **3** | **0** | **4** | **2** | **3** | **0** | **3** | **2** | **1** | **2** | **0** | **7** | **0** | **1** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Existing at position 1 | 7 | 7 | 7 | 2 | | 2 | 2 | 4 | 4 | 4 | 0 | | | 0 | 0 | | 7 | 7 | 7 |
| Existing at position 2 | | 0 | 0 | 0 | | 3 | 3 | 3 | 2 | 2 | 2 | | | 1 | 1 | | 1 | 0 | 0 |
| Existing at position 3 | | | 1 | 1 | | 1 | 0 | 0 | 0 | 3 | 3 | | | 3 | 2 | | 2 | 2 | 1 |
| Page Fault | x | x | x | x | | x | x | x | x | x | x | | | x | x | H | x | x | x |

## Belady's Anomaly

Generally, on increasing the number of frames to a process' virtual memory, its execution becomes faster as less number of page faults occur.

Sometimes the reverse happens, i.e. **more number of page faults occur** when **more frames** are allocated to a process.

This most unexpected result is termed as **Belady's Anomaly**.

**Bélády's anomaly** is the name given to the phenomenon where increasing the number of page frames results in an increase in the number of page faults for a given memory access pattern.

# Belady's Anomaly

```
1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
```

**Case-1:** If the system has 3 frames, the given reference string on using FIFO page replacement algorithm yields a total of 9 page faults. The diagram below illustrates the pattern of the page faults occurring in the example.

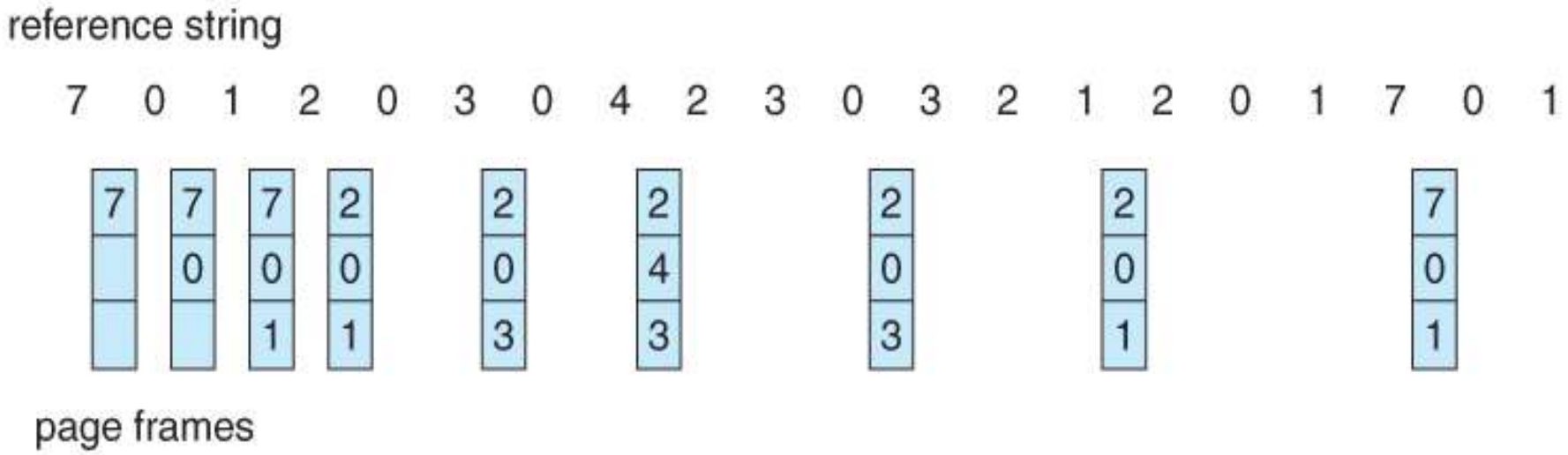
| 1 | 1 | 1 | 2 | 3 | 4 | 1 | 1 | 1 | 2 | 5 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 3 | 4 | 1 | 2 | 2 | 2 | 5 | 3 | 3 |
| | | 3 | 4 | 1 | 2 | 5 | 5 | 5 | 3 | 4 | 4 |
| PF | PF | PF | PF | PF | PF | PF | X | X | PF | PF | X |

**Case-2:** If the system has 4 frames, the given reference string on using FIFO page replacement algorithm yields a total of 10 page faults. The diagram below illustrates the pattern of the page faults occurring in the example.

| 1 | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 4 | 5 | 1 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 2 | 3 | 4 | 5 | 1 | 2 | 3 |
| | | 3 | 3 | 3 | 3 | 4 | 5 | 1 | 2 | 3 | 4 |
| | | | 4 | 4 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |
| PF | PF | PF | PF | X | X | PF | PF | PF | PF | PF | PF |

## Optimal Page Replacement

- This algorithm is simply "Replace the **page that will not be used for the longest time** in the future.
- Figure below shows that by applying **OPT** to the **same reference string** used for the **FIFO** example, the **minimum number of possible page faults** is 9.



# Lecture:
# Page Replacement Strategies: LRU

## LRU Page Replacement:-

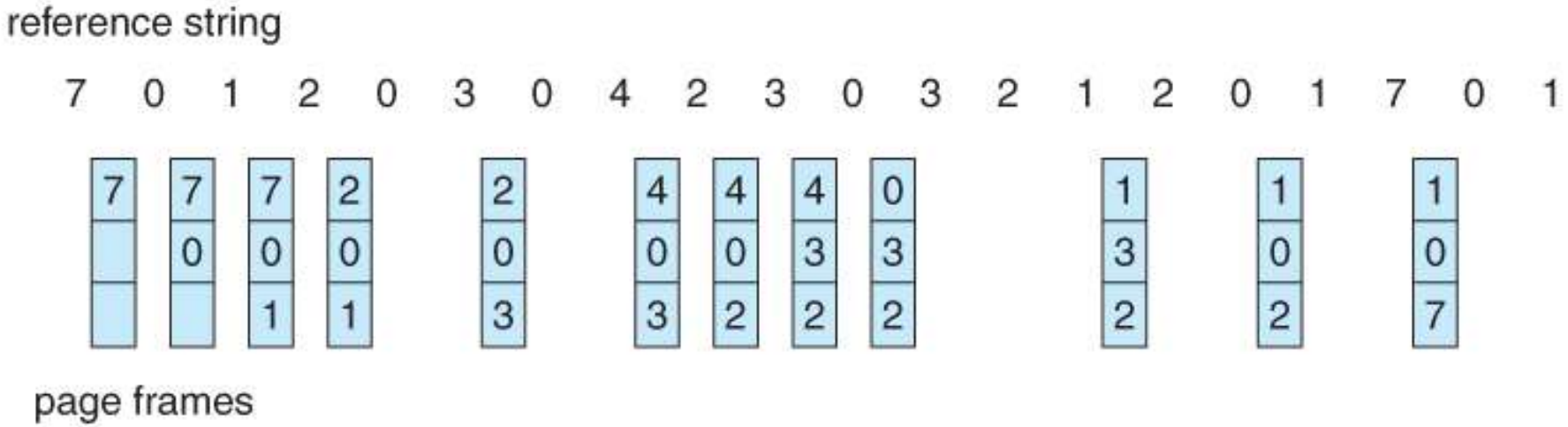
- The prediction behind ***LRU*****,** the ***Least Recently Used*****,** algorithm is that the page that **has not been used in the longest time** is the one that will not be used again in the near future.

- Some view LRU as analogous to OPT, except looking backwards in time instead of forwards.

- Figure below illustrates **LRU** for our sample string, **yielding 12 page faults**, ( as compared to **15 for FIFO and 9 for OPT**. )

- As the name suggests, this algorithm works on the principle of “Least Recently Used”.

- It replaces the page that has **not been referred by the CPU for the longest time.**



## Practice Problems - examples

- Consider the page reference string of size 12:

- 1, 2, 3, 4, 5, 1, 3, 1, 6, 3, 2, 3 with frame size 4(i.e. **maximum 4 pages in a frame**).
  - Also calculate the hit ratio and miss ratio.

## Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

# Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 |
|---|

| 1 |
|---|
| |
| |
| |

| M |
|---|

# Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 |
|---|---|

| 1 | 1 |
|---|---|
| | 2 |
| | |
| | |
| M | M |

# Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 |
|---|---|---|

| 1 | 1 | 1 |
|---|---|---|
| | 2 | 2 |
| | | 3 |
| | | |

| M | M | M |
|---|---|---|

**M = Miss**
**H = Hit**

# Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 |
|---|---|---|---|

| 1 | 1 | 1 | 1 |
|---|---|---|---|
|  | 2 | 2 | 2 |
|  |  | 3 | 3 |
|  |  |  | 4 |
| M | M | M | M |

**M = Miss**
**H = Hit**

# Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 |
|---|---|---|---|---|
|  | 2 | 2 | 2 | 2 |
|  |  | 3 | 3 | 3 |
|  |  |  | 4 | 4 |
| M | M | M | M | M |

**M = Miss**
**H = Hit**

# Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 |
|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 |
|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 |
| | | 3 | 3 | 3 | 3 |
| | | | 4 | 4 | 4 |

| M | M | M | M | M | M |
|---|---|---|---|---|---|

**M = Miss**
**H = Hit**

# Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 |
|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 |
|---|---|---|---|---|---|---|
|  | 2 | 2 | 2 | 2 | 1 | 1 |
|  |  | 3 | 3 | 3 | 3 | 3 |
|  |  |  | 4 | 4 | 4 | 4 |
| M | M | M | M | M | M | H |

**M = Miss**
**H = Hit**

# Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 |
|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 |
|---|---|---|---|---|---|---|---|
|  | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
|  |  | 3 | 3 | 3 | 3 | 3 | 3 |
|  |  |  | 4 | 4 | 4 | 4 | 4 |
| M | M | M | M | M | M | H | H |

**M = Miss**
**H = Hit**

# Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 |
|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 |
|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 6 |
| | | | 4 | 4 | 4 | 4 | 4 | 4 |
| M | M | M | M | M | M | H | H | M |

**M = Miss**
**H = Hit**

# Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 |
|---|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 5 |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 6 | 6 |
| | | | 4 | 4 | 4 | 4 | 4 | 4 | 3 |
| M | M | M | M | M | M | H | H | M | M |

**M = Miss**
**H = Hit**

# Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 5 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 6 | 6 | 6 |
| | | | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 |
| M | M | M | M | M | M | H | H | M | M | M |

**M = Miss**
**H = Hit**

# Solution- First in First out (FIFO) page replacement policy

Total number of references = 12

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 5 | 2 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 6 | 6 | 6 | 6 |
| | | | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 |
| M | M | M | M | M | M | H | H | M | M | M | H |

**M = Miss**
**H = Hit**

## Solution- First in First out (FIFO) page replacement policy

**Calculating Hit ratio-**

**Total number of page hits** = Total number of references – Total number of page misses or page faults

= 12– 9

= 3

**Total Page Fault = 9**

**Page Fault ratio = 9/12 i.e. total miss/total possible cases**

Thus, Hit ratio = Total number of page hits / Total number of references

= 3 / 10

= 0.3 or 30%

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 |
|---|

| 1 |
|---|
| |
| |
| |

| M |
|---|

M = Mis
H = Hit

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 |
|---|---|

| 1 | 1 |
|---|---|
| | 2 |
| | |
| | |
| M | M |

M = Miss
H = Hit

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 |
|---|---|---|

| 1 | 1 | 1 |
|---|---|---|
|  | 2 | 2 |
|  |  | 3 |
|  |  |  |
| M | M | M |

M = Miss
H = Hit

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 |
|---|---|---|---|

| 1 | 1 | 1 | 1 |
|---|---|---|---|
| | 2 | 2 | 2 |
| | | 3 | 3 |
| | | | 4 |
| M | M | M | M |

M = Miss
H = Hit

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 |
|---|---|---|---|---|
|  | 2 | 2 | 2 | 2 |
|  |  | 3 | 3 | 3 |
|  |  |  | 4 | 4 |
| M | M | M | M | M |

M = Miss
H = Hit

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 |
|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 |
|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 |
| | | 3 | 3 | 3 | 3 |
| | | | 4 | 4 | 4 |
| M | M | M | M | M | M |

M = Miss
H = Hit

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 |
|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 |
|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 | 1 |
| | | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 4 | 4 | 4 |
| M | M | M | M | M | M | H |

M = Miss
H = Hit

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 |
|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 |
|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 | 1 | 1 |
| | | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 4 | 4 | 4 | 4 |
| M | M | M | M | M | M | H | H |

**M = Miss**
**H = Hit**

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 |
|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 |
|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 4 | 4 | 4 | 4 | 6 |
| M | M | M | M | M | M | H | H | M |

M = Miss
H = Hit

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 |
|---|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 5 |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 4 | 4 | 4 | 4 | 6 | 6 |
| M | M | M | M | M | M | H | H | M | H |

M = Miss
H = Hit

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 5 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 4 | 4 | 4 | 4 | 6 | 6 | 6 |
| M | M | M | M | M | M | H | H | M | H | M |

**M = Miss**
**H = Hit**

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 5 | 2 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 4 | 4 | 4 | 4 | 6 | 6 | 6 | 6 |
| M | M | M | M | M | M | H | H | M | H | M | H |

M = Miss
H = Hit

# Solution-LRU page replacement policy

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 5 | 2 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 4 | 4 | 4 | 4 | 6 | 6 | 6 | 6 |
| M | M | M | M | M | M | H | H | M | H | M | H |

M = Miss
H = Hit

**Total Page Fault = 8**

**Page Fault ratio = 8/12**

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 |
|---|---|

| 1 | 1 |
|---|---|
|  | 2 |
|  |  |
|  |  |

| M | M |
|---|---|

M = Miss
H = Hit

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 |
|---|---|---|

| 1 | 1 | 1 |
|---|---|---|
|  | 2 | 2 |
|  |  | 3 |
|  |  |  |
| M | M | M |

M = Miss
H = Hit

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 |
|---|---|---|---|

| 1 | 1 | 1 | 1 |
|---|---|---|---|
|  | 2 | 2 | 2 |
|  |  | 3 | 3 |
|  |  |  | 4 |
| M | M | M | M |

M = Miss
H = Hit

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|
|  | 2 | 2 | 2 | 2 |
|  |  | 3 | 3 | 3 |
|  |  |  | 4 | 5 |
| M | M | M | M | M |

M = Miss
H = Hit

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 |
|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|
|  | 2 | 2 | 2 | 2 | 2 |
|  |  | 3 | 3 | 3 | 3 |
|  |  |  | 4 | 5 | 5 |
| M | M | M | M | M | H |

M = Miss
H = Hit

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 |
|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 2 | 2 |
| | | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 5 | 5 | 5 |
| M | M | M | M | M | H | H |

M = Miss
H = Hit

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 |
|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| | | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 5 | 5 | 5 | 5 |
| M | M | M | M | M | H | H | H |

M = Miss
H = Hit

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 |
|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 |
|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 5 | 5 | 5 | 5 | 5 |
| M | M | M | M | M | H | H | H | M |

M = Miss
H = Hit

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 |
|---|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 5 | 5 | 5 | 5 | 5 | 5 |
| M | M | M | M | M | H | H | H | M | H |

M = Miss
H = Hit

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| M | M | M | M | M | H | H | H | M | H | H |

M = Miss
H = Hit

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| M | M | M | M | M | H | H | H | M | H | H | H |

M = Miss
H = Hit

# Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| M | M | M | M | M | H | H | H | M | H | H | H |

M = Miss
H = Hit

## Solution Optimal Page Replacement

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 2 | 3 | 4 | 5 | 1 | 3 | 1 | 6 | 3 | 2 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| | | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| M | M | M | M | M | H | H | H | M | H | H | H |

M = Miss
H = Hit

**Total Page Fault = 6** **Page Fault ratio = 6/12**

## Belady's Anomaly

- In the case of LRU and optimal page replacement algorithms, it is seen that the number of **page faults will be reduced if we increase the number of frames.**

- **However, Belady found that, In FIFO page replacement algorithm, the number of page faults will get increased with the increment in number of frames.**

- This is the strange behavior shown by FIFO algorithm in some of the cases. This is an Anomaly called as **Belady's Anomaly.**

## Let's examine through example :

- The reference String is given as 0 1 5 3 0 1 4 0 1 5 3 4. Let's analyze the behavior of FIFO algorithm in two cases.
- **Case 1: Number of frames = 3**

| Request | 0 | 1 | 5 | 3 | 0 | 1 | 4 | 0 | 1 | 5 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Frame 3 | | | 5 | 5 | 5 | 1 | 1 | 1 | 1 | 1 | 3 | 3 |
| Frame 2 | | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 5 | 5 | 5 |
| Frame 1 | 0 | 0 | 0 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 |
| Miss/Hit | Miss | Miss | Miss | Miss | Miss | Miss | Miss | Hit | Hit | Miss | Miss | Hit |

**Number of Page Faults = 9**

## Cont.

- **Case 2: Number of frames = 4**

| Request | 0 | 1 | 5 | 3 | 0 | 1 | 4 | 0 | 1 | 5 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Frame 4 | | | | 3 | 3 | 3 | 3 | 3 | 3 | 5 | 5 | 5 |
| Frame 3 | | | 5 | 5 | 5 | 5 | 5 | 5 | 1 | 1 | 1 | 1 |
| Frame 2 | | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 4 |
| Frame 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 4 | 4 | 3 | 3 |
| Miss/Hit | Miss | Miss | Miss | Miss | Hit | Hit | Miss | Miss | Miss | Miss | Miss | Miss |

**Number of Page Faults = 10**

Therefore, in this example, the number of page faults is increasing by increasing the number of frames hence this suffers from **Belady'sAnomaly.**

## Memory Management- Page Replacement

- Counting-based Algorithms
  - Keep a counter of the number of references that have been made to each page.
  - Two possibilities: Least/Most Frequently Used (LFU/MFU).
  - LFU Algorithm: replaces page with smallest count; others were and will be used more.
  - MFU Algorithm: based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

## Sample questions

Q.2. a. Calculate hit and miss for the following string using page replacement policies – FIFO, LRU and Optimal. Compare it for the frame size 3 & 4.

1 2 3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4 3 May-2017 [10]

(b) Calculate hit and miss percentage for the following string using page replacement policies FIFO, LRU and Optimal. Compare it for the frame size 3 and 4. **10**

2,0,3,0,4,2,3,0,3,2,7,2,0,7,5,0,7,5,7,0

May-2016

What is paging ? Explain LRU, FIFO, OPT page replacement policy for the given page frame sequences. Page frame size is 4.

2, 3, 4, 2, 1, 3, 7, 5, 4, 3, 2, 3, 1

Calculate page hit and page miss. May-2015

# **Lecture:** Thrashing

## Hardware and Control Structures

- Two characteristics fundamental to memory management:
    1) all memory references are logical addresses that are dynamically translated into physical addresses at run time
    2) a process may be broken up into a number of pieces that don't need to be contiguously located in main memory during execution
- If these two characteristics are present, it is not necessary that all of the pages or segments of a process be in main memory during execution

# Terminology

| | |
|---|---|
| **Virtual memory** | A storage allocation scheme in which secondary memory can be addressed as though it were part of main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites, and program-generated addresses are translated automatically to the corresponding machine addresses.The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of secondary memory available and not by the actual number of main storage locations. |
| **Virtual address** | The address assigned to a location in virtual memory to allow that location to be accessed as though it were part of main memory. |
| **Virtual address space** | The virtual storage assigned to a process. |
| **Address space** | The range of memory addresses available to a process. |
| **Real address** | The address of a storage location in main memory. |

## Execution of a Process

- Operating system brings into main memory a few pieces of the program
- Resident set - portion of process that is in main memory
- An interrupt is generated when an address is needed that is not in main memory (segment/page fault)
- Operating system places the process  in a blocking state

## Execution of a Process

- Piece of process that contains the logical address is brought into main memory
  - operating system issues a disk I/O Read request
  - another process is dispatched to run while the disk I/O takes place
  - an interrupt is issued when disk I/O is complete, which causes the operating system to place the affected process in the Ready state

## Support Needed for Virtual Memory

**For virtual memory to be practical and effective:**

- hardware must support paging and segmentation
- operating system must include software for managing the movement of pages and/or segments between secondary memory and main memory

## Demand Paging

- Bring a page into memory only when it is needed
  - Less I/O needed
  - Less memory needed
  - Faster response (no need to wait for all pages to load)
  - More users
- Page is needed ⇒ reference to it
  - invalid reference ⇒ abort
  - not-in-memory ⇒ bring to memory
- **Lazy swapper** – never swaps a page into memory unless page will be needed
  - Swapper that deals with pages is a **pager**

## Page Fault

- If there is a reference to a page, first reference to that page will trap to operating system:

  **page fault**

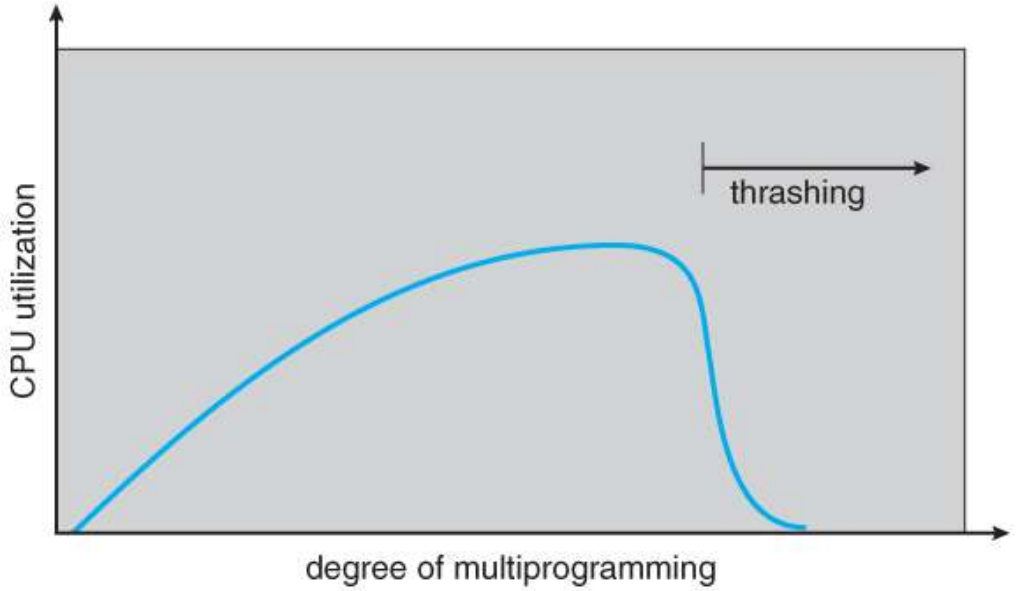
1. Operating system looks at another table to decide:
   - Invalid reference ⇒ abort
   - Just not in memory
2. Get empty frame
3. Swap page into frame
4. Reset tables
5. Set validation bit = **v**
6. Restart the instruction that caused the page fault

## Memory Management- Virtual Memory- Allocation of Frames

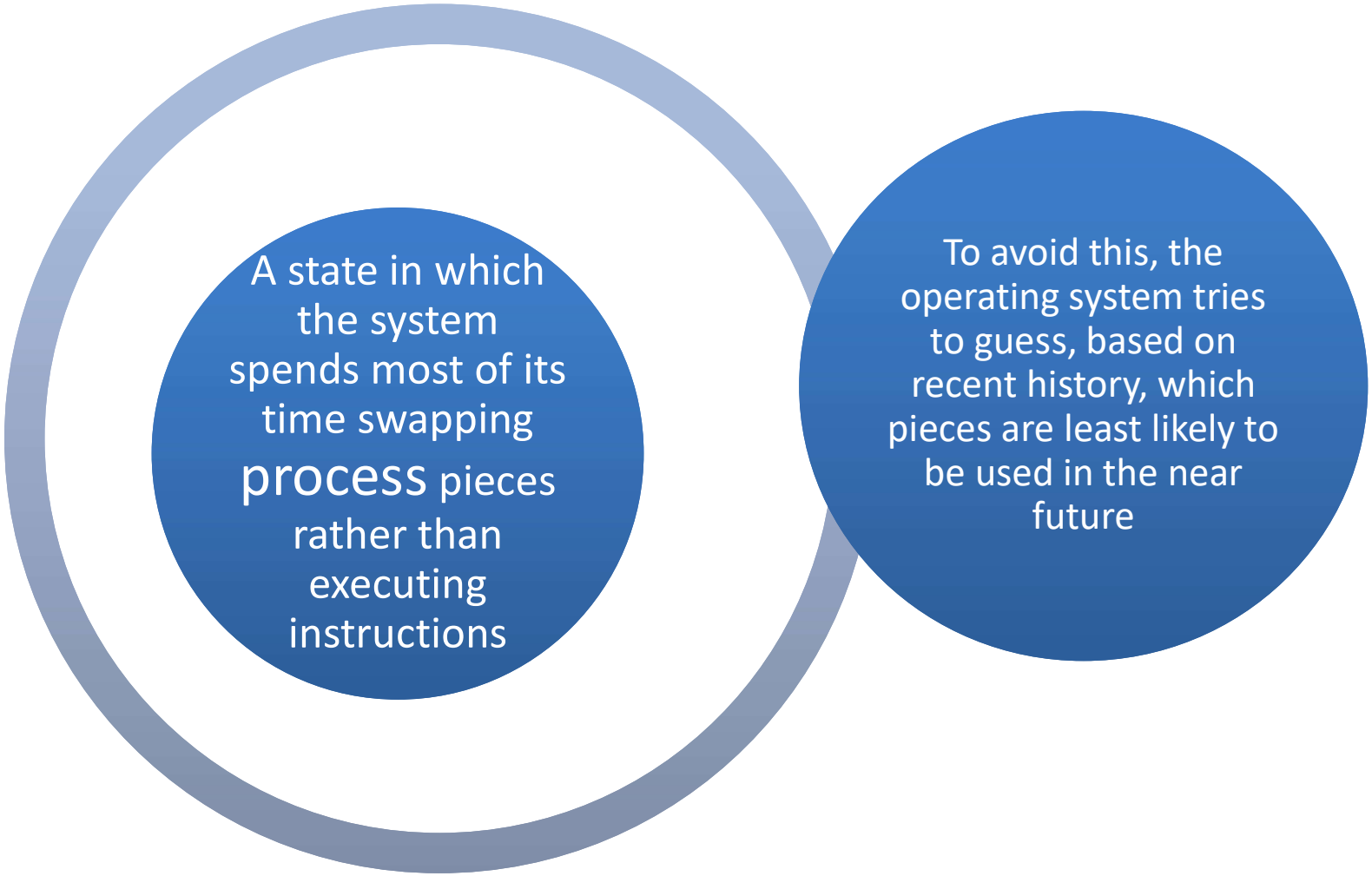
- **Global versus Local Allocation**
  - **With local replacement**, the number of pages allocated to a process is fixed, and page **replacement occurs** only **amongst the pages allocated** to this process.
  - With **global replacement**, **any page may be a potential victim**, whether it currently belongs to the process seeking a free frame or not.
  - Local page replacement **allows processes to better control their own page fault rates**, and leads to more consistent performance of a given process over different system load levels.
  - Global page replacement **is overall more efficient**, and is the more **commonly used approach**.

## Memory Management- Virtual Memory- Thrashing

- If a process **cannot maintain its minimum required number of frames**, then it **must be swapped out**, freeing up frames for other processes. This is an intermediate level of CPU scheduling.
- But what about a process **that can keep its minimum**, but **cannot keep all of the frames that it is currently using** on a regular basis? In this case **it is forced to page out pages that it will need again** in the very near future, leading to **large numbers of page faults.**
- A process that is spending more time paging than executing is said to be ***thrashing***
-



# Thrashing



## Memory Management- Virtual Memory- Thrashing

- **Cause of Thrashing**
  - Early process scheduling schemes would **control the level of multiprogramming** allowed **based on CPU utilization**, **adding in more processes** when **CPU utilization was low**.
  - The **problem is that** when **memory filled up** and **processes started spending** lots of time **waiting for their pages to page in**, then **CPU utilization would lower**, **causing** the **schedule to add** in even more **processes** and **exacerbating** the problem! Eventually the system would essentially grind to a halt.
  - Local page replacement policies **can prevent one thrashing process** from **taking pages away from other processes**, but it still **tends to clog up the I/O queue**, thereby slowing down any other process that needs to do even a little bit of paging ( or any other I/O for that matter. )

## Principle of Locality

- Program and data references within a process tend to cluster.
- Only a few pieces of a process will be needed over a short period of time.
- Therefore it is possible to make intelligent guesses about which pieces will be needed in the future.
- Making good guesses avoids thrashing

## Summary

- Memory Management
  - Background
    - Memory Protection
    - Address Binding
    - Logical Address Space
    - Physical Address Space
    - Memory mgmt. unit
  - Swapping
  - Contiguous Memory Allocation
    - Allocation Algorithms (Best, First, Worst)
    - Fragmentation
  - Non- Contiguous Memory Allocation
    - Paging
    - Structure Of page Table
    - Segmentation
- Virtual Memory
- Demand Paging
- Page Replacement Algorithms
  - FIFO
  - Optimal
  - LRU
  - LRU Approximation
    - Second Chance
    - Enhanced Second Chance
  - Counting Based
- Allocation Of Frames
  - Minimum No. of frames
  - Allocation Algorithms
  - Local v/s Global Allocation
- Thrashing

## Previous Year Questions

Explain the effect of page frame size on performance of page replacement algorithms. 5

Explain Thrashing. 5

[June 2018]

Explain paging hardware with TLB along with protection bits in page table. [June 2018] 10

ı What is paging? Explain LRU, FIFO and Optimal page replacement policy for the following string. Page frame size is 4. 10
1,2,3,4,5,3,4,1,6,7,8,7,8,9,7,8,9,5,4,5,4,2

[May2019]

Given memory partitions of 150k,500k,200k,300k,550k(in order) how would each of the first fit, best fit and worst fit algorithm places the processes of 220k,430k,110k,425k(in order).Evaluate, which algorithm makes most efficient use of memory? 10

[May2019]

## Previous Year Questions

What is paging? Explain LRU, FIFO and Optimal page replacement policy for the following string. Page frame size is 4. Calculate the hit ratio for the same.
1,2,3,4,5,3,4,1,6,7,8,7,8,9,7,8,9,5,4,5,4,2 [Dec 2018] **10**

Explain virtual memory concept with respect to paging, segmentation and TLB. [Dec 2018] **10**

Explain different types of memory fragmentation. [Dec 2019] [8]

Compare the performance of FIFO, LRU and Optimal based on number of page hit for the following string. Frame size = 3; String (pages): 1 2 3 4 5 2 1 3 3 2 4 5 [12]
[Dec 2019]

# Thank You

D Y PATIL
DEEMED TO BE
UNIVERSITY
RAMRAO ADIK
INSTITUTE OF TECHNOLOGY
NAVI MUMBAI