NUMERICALS

On a simple paging system with 2^{24} bytes of physical memory , 256 pages of logical address space , and a page size of 2^{10} bytes,

How many bits in the physical address specify the page frame

Solution of the problem mentioned in previous slide

- A 24-bit address is required to cover 2^{24}-bytes physical address space . Since pages , and therefore page frames , are 2^{10} bytes, the last bits in the physical address specify the page offset .The remaining 14 bits specify the page frame number.

NUMERICALS

On a simple paging system with 2^{24} bytes of physical memory , 256 pages of logical address space , and a page size of 2^{10} bytes,

How many entries are there in page table(how long is the page table)

Solution of the problem mentioned in previous slide

- The page table must contain an entry for each page . Since there are 256 pages in the logical address space , the page table must be 256 entries long.

NUMERICALS

On a simple paging system with 2^{24} bytes of physical memory , 256 pages of logical address space , and a page size of 2^{10} bytes,

How many bits are needed to store an entry in the page table (how wide is the page table)? Assume each page entry contain a valid/invslid bit in addition to the page frame number

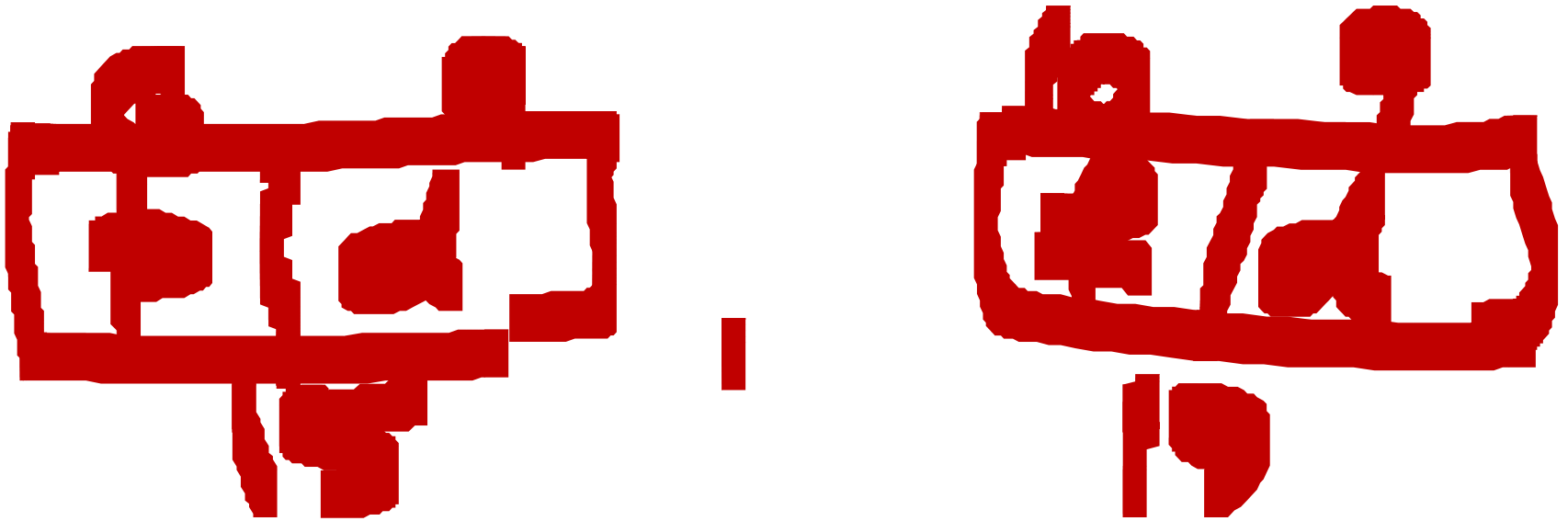
Solution of the problem mentioned in previous slide

- Each entry contains 1 bit to indicate if the page is valid and 14 bits to specify the page frame number.

NUMERICALS

On a simple paging system with page table containing 64 entries of 11 bits (including valid/ invalid bit) each, and a page size of 512 bytes

How many bits in the logical address space specify the page number



Solution of the problem mentioned in previous slide

- Since there are $64=2^6$ pages, 6 bits of logical address space are required to specify the page number .

NUMERICALS

On a simple paging system with page table containing 64 entries of 11 bits (including valid/ invalid bit) each, and a page size of 512 bytes

How many bits in the logical address space specify the offset within the page

Solution of the problem mentioned in previous slide

- Since a page is 512 =2^9 bytes long , 9 bits are required to specify the offset within the page

NUMERICALS

On a simple paging system with page table containing 64 entries of 11 bits (including valid/ invalid bit) each, and a page size of 512 bytes

How many bits are in a logical address

Solution of the problem mentioned in previous slide

- 15.
- The 6 MSB of the logical address would specify the page number , the 9 LSB would specify the page offset , making the total number of bits 15.

NUMERICALS

On a simple paging system with page table containing 64 entries of 11 bits (including valid/ invalid bit) each, and a page size of 512 bytes

What is the size of the logical address space

Solution of the problem mentioned in previous slide

- A 15- bit address creates an address space of 2^{15}=32,768

NUMERICALS

On a simple paging system with page table containing 64 entries of 11 bits (including valid/ invalid bit) each, and a page size of 512 bytes

How many bits in the physical address space specify the page frame number

Solution of the problem mentioned in previous slide

- A page table entry contains the frame number and a valid/invalid bit. If the total number of bits is 11 , the number of bits in a page frame number are 10.

NUMERICALS

On a simple paging system with page table containing 64 entries of 10 bits (including valid/ invalid bit) each, and a page size of 512 bytes

How many bits in the physical address space specify the offset within the page frame

Solution of the problem mentioned in previous slide

- The page frame offset is the same as the page offset, a 9-bit value.

NUMERICALS

On a simple paging system with page table containing 64 entries of 11 bits (including valid/ invalid bit) each, and a page size of 512 bytes

How many bits are in the physical address

Solution of the problem mentioned in previous slide

- 19.
- The 10 MSB of the physical address would specify the page frame number , the 9 LSB would specify the page offset , making the total number of bits 19.

NUMERICALS

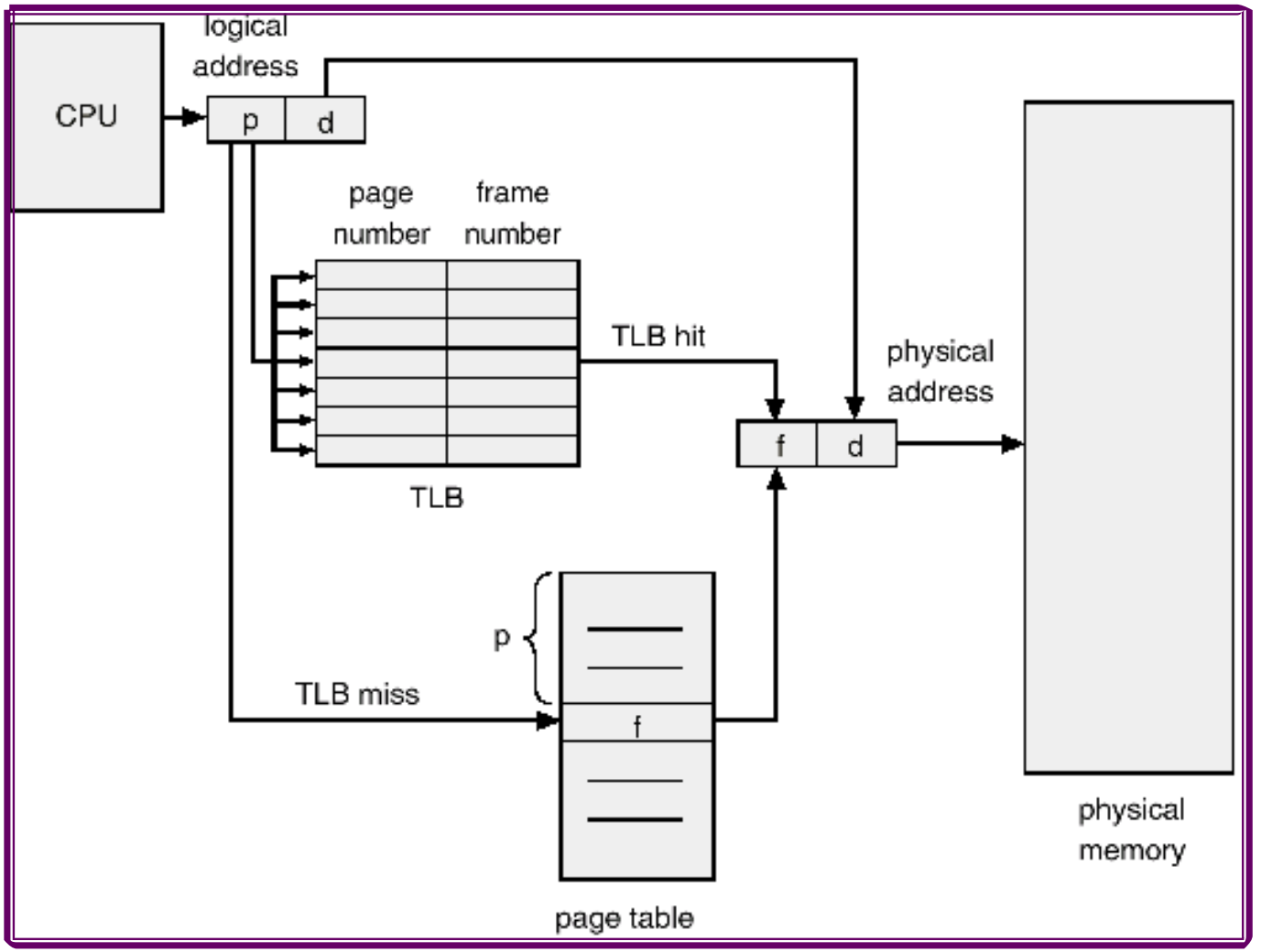
On a simple paging system with page table containing 64 entries of 11 bits (including valid/ invalid bit) each, and a page size of 512 bytes

What is the size of the physical address space

Solution of the problem mentioned in previous slide

- A 19-bit address creates an address space of 2^{19}.

Paging Hardware With TLB



Effective Access Time

- Associative Lookup = ε time unit
- Assume memory cycle time is 1 microsecond
- Hit ratio – percentage of times that a page number is found in the associative registers; ration related to number of associative registers.
- Hit ratio = α
- Effective Access Time (EAT)

$$EAT = (1 + \varepsilon)\,\alpha + (2 + \varepsilon)(1 - \alpha)$$

$$= 2 + \varepsilon - \alpha$$

Example-1 : Find the Average Effective Memory Access Time when 80 percent is the hit ratio and it take 20 nanoseconds to search TLB and 100 nanoseconds to access memory (**Hint :** then take 120 nanoseconds to access mapped memory). In Case of Miss, It also takes 100 nanosecond to access frame number. The Total time taken in case of Miss is (20 + 100 + 100 = 220 nanosecond)

- Solution:

Average Access Time = 0.80 * (20+ 100) + 0.20 * (20 + 100 + 100)

= 0.80 * 120 + 0.20 * 220

= 96 + 44 = **140 nanoseconds**

Example:-2

- Consider a paging hardware with a TLB. Assume that the entire page table and all the pages are in the physical memory. It takes 10 milliseconds to search the TLB and 80 milliseconds to access the physical memory. If the TLB hit ratio is 0.6, the effective memory access time (in milliseconds) is _____.

- **Solution:**

Effective Access Time =

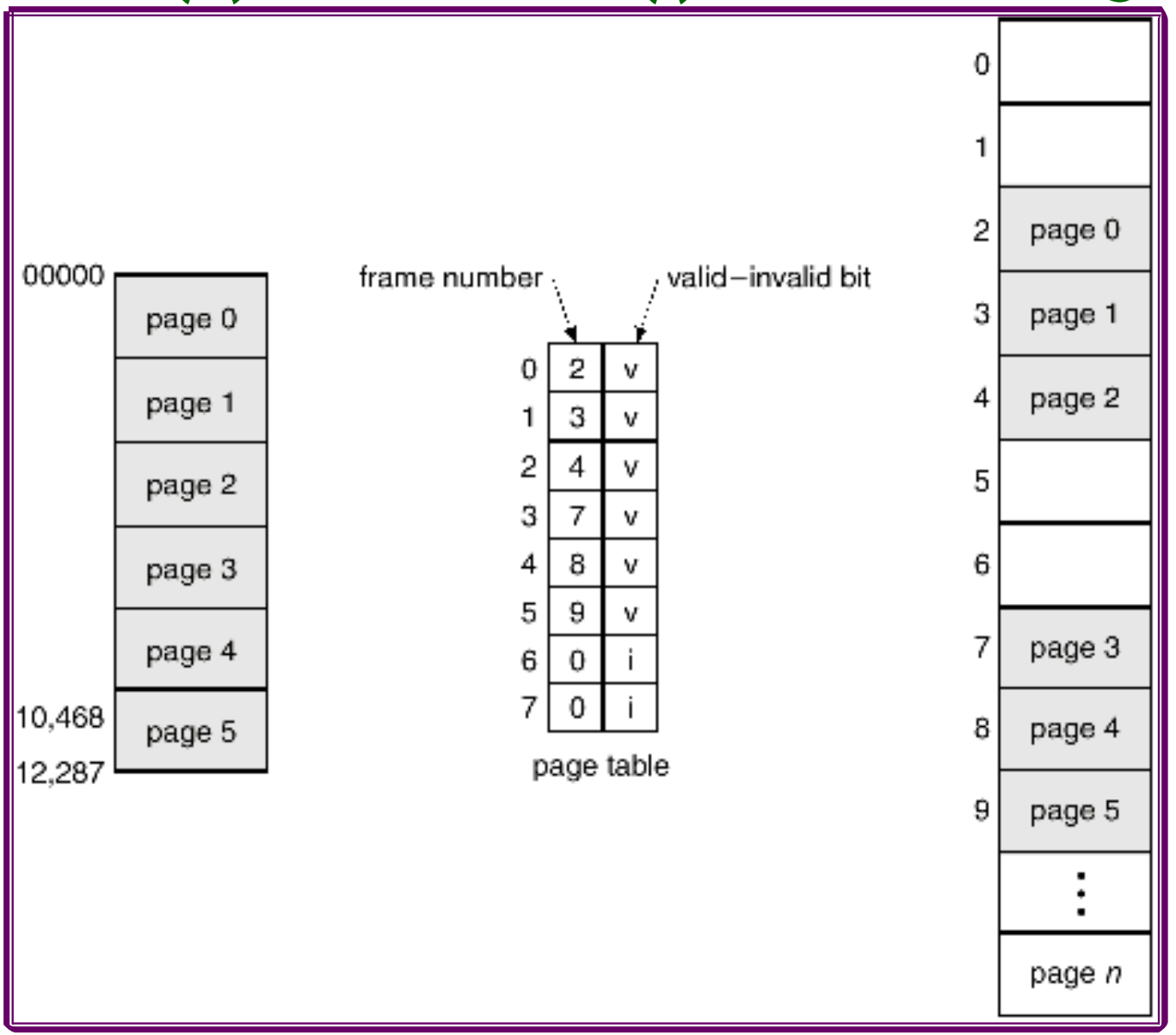
hit ratio * (TLB access time + Main memory access time) +
(1 – hit ratio) * (TLB access time + 2 * main memory time)

= 0.6*(10+80) + (1-0.6)*(10+2*80)
= 0.6 * (90) + 0.4 * (170)
= 122

Memory Protection

- Memory protection implemented by associating protection bit with each frame.
- *Valid-invalid* bit attached to each entry in the page table:
  - "valid" indicates that the associated page is in the process' logical address space, and is thus a legal page.
  - "invalid" indicates that the page is not in the process' logical address space.

Valid (v) or Invalid (i) Bit In A Page Table



Suppose, memory space is partitioned into 8 fixed size slots of 8 MB each. Assume 8 processes are currently requesting memory usage with sizes indicating as [2 MB, 4 MB, 3 MB,7 MB, 9 MB, 6 MB, 1 MB, 8 MB]. Calculate the internal fragmentation if Single partition is allocated to one process only. Also Calculate internal partition when multiple partition can be allocated to one process only.

Virtual Memory

- *Main Reference from chapter 9 of "Operating System Concepts" by Galvin 5th and 8th Edition.*

hitendra.garg@gmail.com

Chapter 9: *Virtual Memory*

- Background
- Demand Paging
- Page Replacement && its algorithms.
- Allocation of Frames
- Thrashing

- **Already have some flavor of Virtual Memory during *OVERLAYS*.**

hitendra.garg@gmail.com

Objectives

- To describe the benefits of a virtual memory system
- To explain the concepts of demand paging, page-replacement algorithms, and allocation of page frames.

Virtual Memory

- It is an illusion of memory which is different from the “real” memory, i.e, RAM, existing in a system. A user or his/her application program sees only the virtual memory. The memory may be much larger in size than the real memory. The kernel implements the illusion using a combination of hardware and software means.
- The basic of a virtual memory (**VM**) implementation is **noncontiguous memory allocation**. **Noncontiguous memory allocation permits part of a program to be loaded into 2 or more non-adjacent areas of memory for execution**. This technique reduces the problem of memory fragmentation since a free area of memory can be reused even it is not large enough to hold an entire program.
- **Either the case of paging or segmentation or paged segmentation, it is required that whole of the process must be in main memory which is not the case in virtual memory.**

Virtual Memory

- Another important aspect of a VM implementation is an **arrangement which enables a program to execute even when its entire code and data are not present in the memory**.
- The basic idea is to **load only the required portions of the code and data in memory at any time**. **This arrangement enables execution of a program whose side exceeds the size of the memory.** The kernel uses this idea to reduce memory allocation to programs in general-that is, **even programs which can fit into memory are not loaded fully into memory**. **This strategy increases the number of programs which can be accommodated in memory even further**.

hitendra.garg@gmail.com

Background

- Virtual memory – separation of user logical memory from physical memory.
  - Only part of the program needs to be in memory for execution
  - Logical address space can therefore be much larger than physical address space
  - Allows address spaces to be shared by several processes

  Virtual memory can be implemented via:
  - **Demand paging [bring pages of the process from the secondary storage device onto the RAM (Physical memory) when required(on demand)]**
  - **Demand segmentation [bring segments of the process from the secondary storage device onto the RAM (Physical memory) when required(on demand)]**

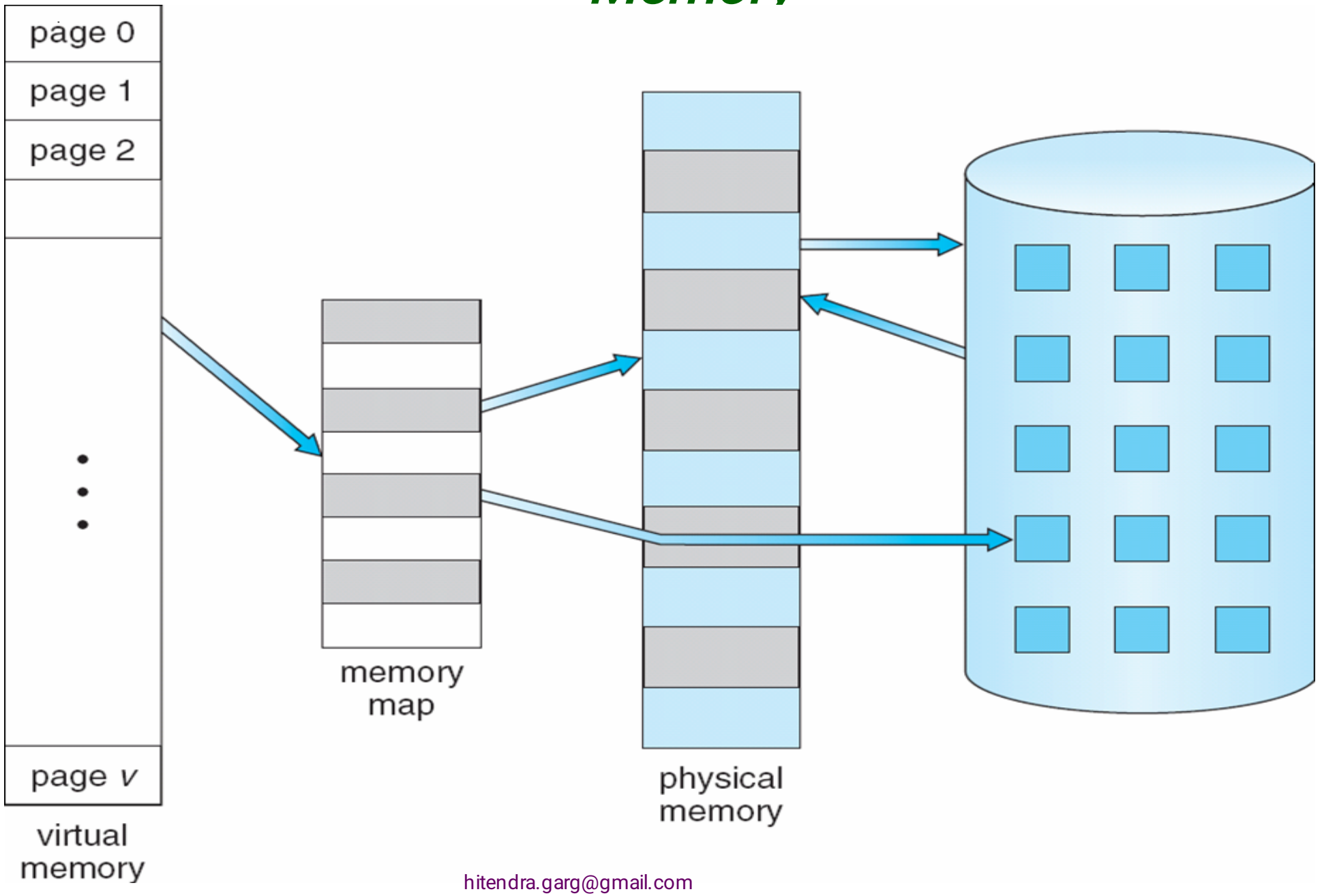
Virtual Memory

Virtual memory is a technique that allows a process to execute in the main memory space which is smaller than the page size.

Only a part of the process needs to be loaded in the main memory for it to execute.

Sharing of address space among several processes.

Virtual Memory That is Larger Than Physical Memory



hitendra.garg@gmail.com

Demand Paging

- Bring a page into memory only when it is needed
  - Less I/O needed[All of the features of S/W is rarely used i. e., bring into main memory, for example exception handling.]
  - Less memory needed
  - Faster response [Lesser response time.]
  - More users[High Degree of multiprogramming]

Demand Paging

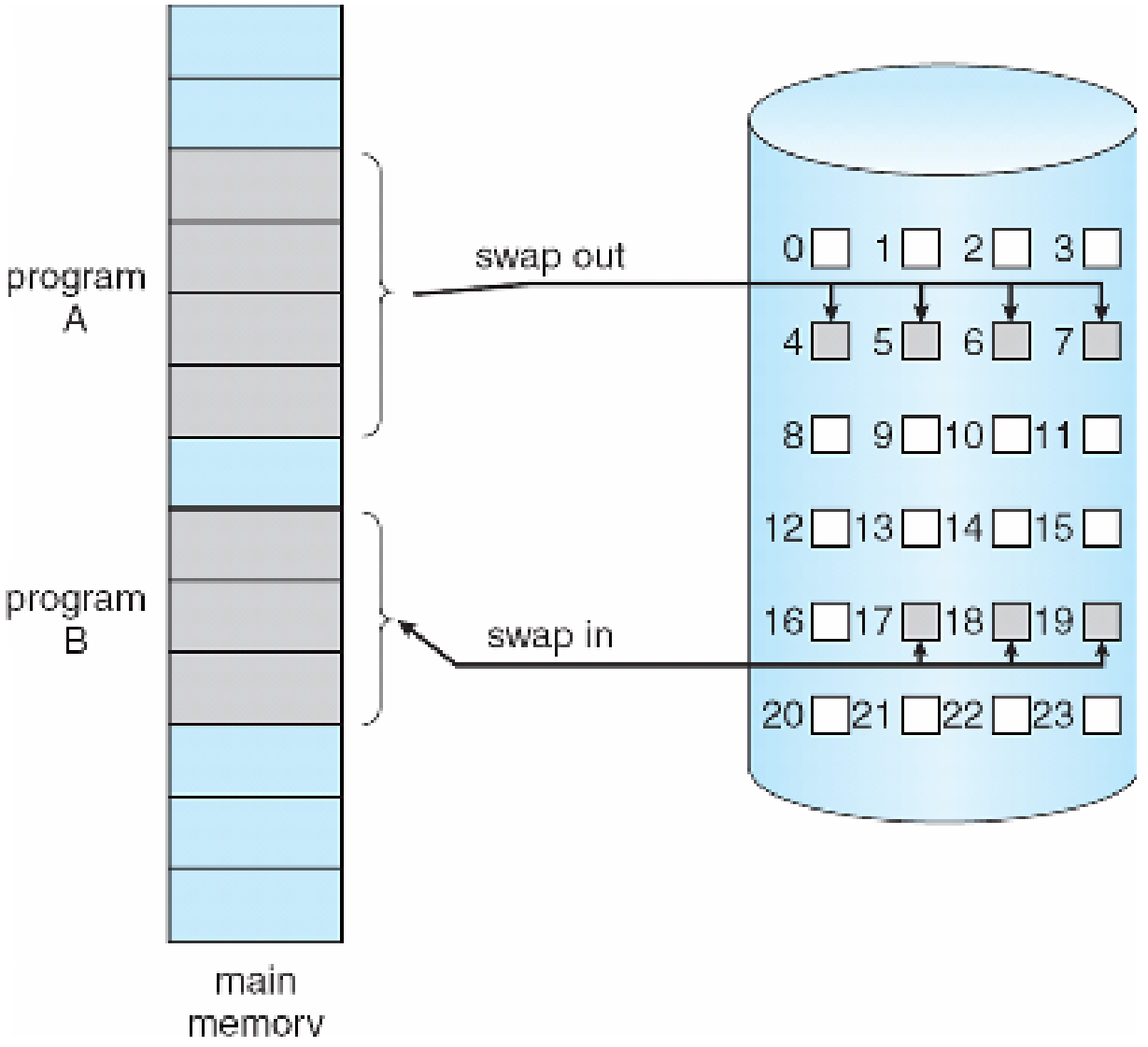
Page is needed ⇒ reference to it

During translation from logical to physical address translation

- invalid reference ⇒ abort[demanded page by the process is not in process address space, it is not the part of its address space.]
- not-in-memory ⇒ Page Fault ⇒ bring to memory[demanded page by the process is valid and is in process address space, but not presently available in primary memory[RAM]. Trap for the page fault[Due to Trap , common service code will execute and check whether it is due to **invalid reference or due to not-in-memory**].[Page Fault ?].]

- No free frame ⇒**Swapping[out and in]**[Page Replacement ?]
- Lazy swapper – never swaps a page into memory unless page will be needed
  - Swapper that deals with pages is a pager

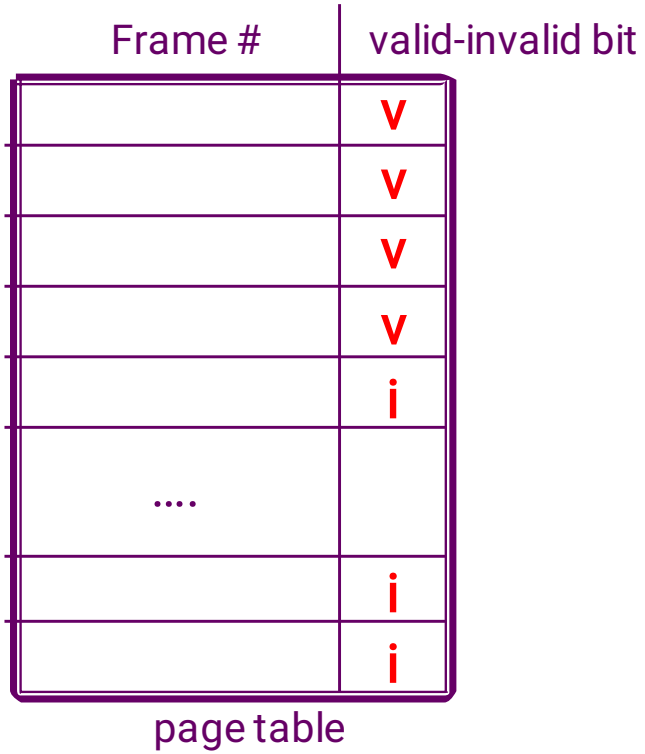
Transfer of a Paged Memory to Contiguous Disk Space[Swapping]



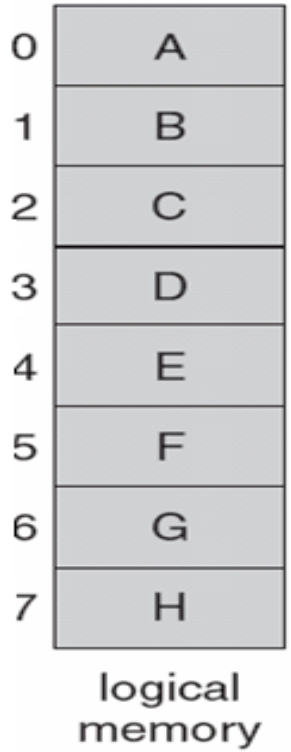
hitendra.garg@gmail.com

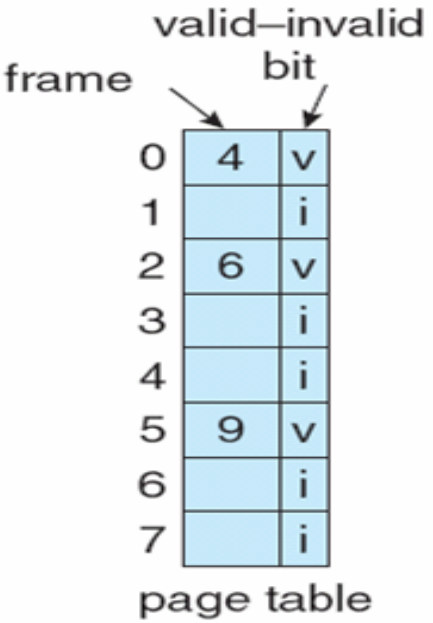
Valid-Invalid Bit

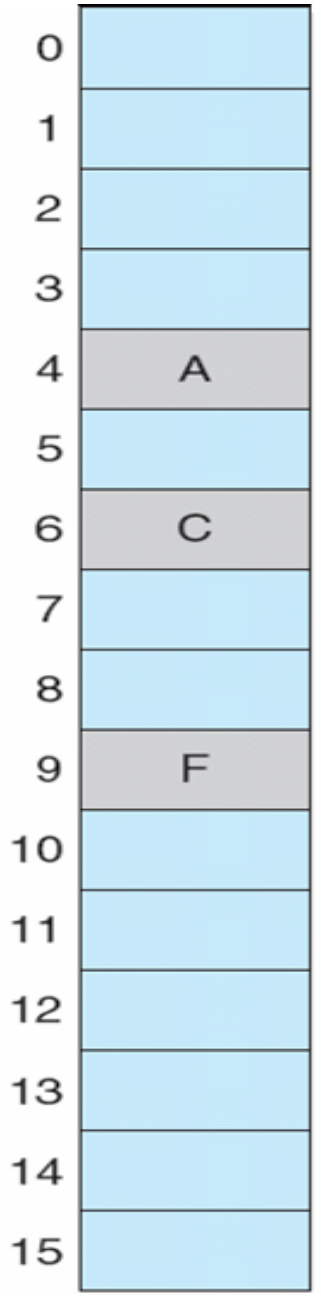
- With each page table entry a valid–invalid bit is associated (v ⇒ in-memory, i ⇒ not-in-memory)
- Initially valid–invalid bit is set to i on all entries
- During address translation, if valid–invalid bit in page table entry is I ⇒ page fault
- Example of a page table snapshot:

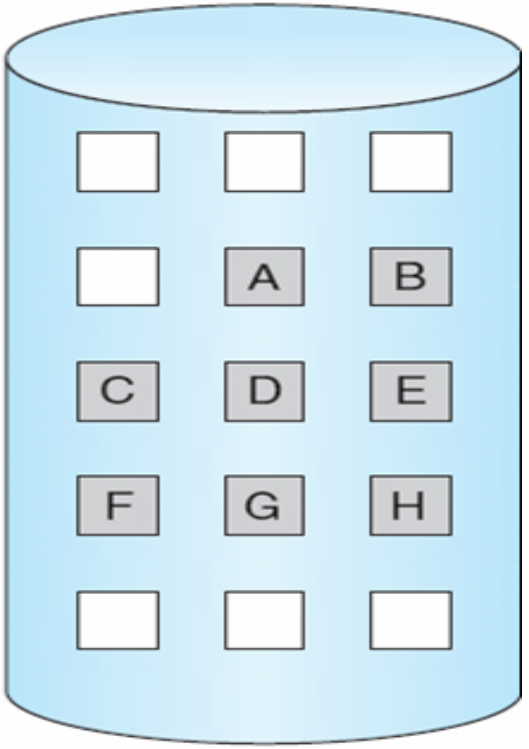


Page Table When Some Pages Are Not in Main Memory









hitendra.garg@gmail.com

Page Fault

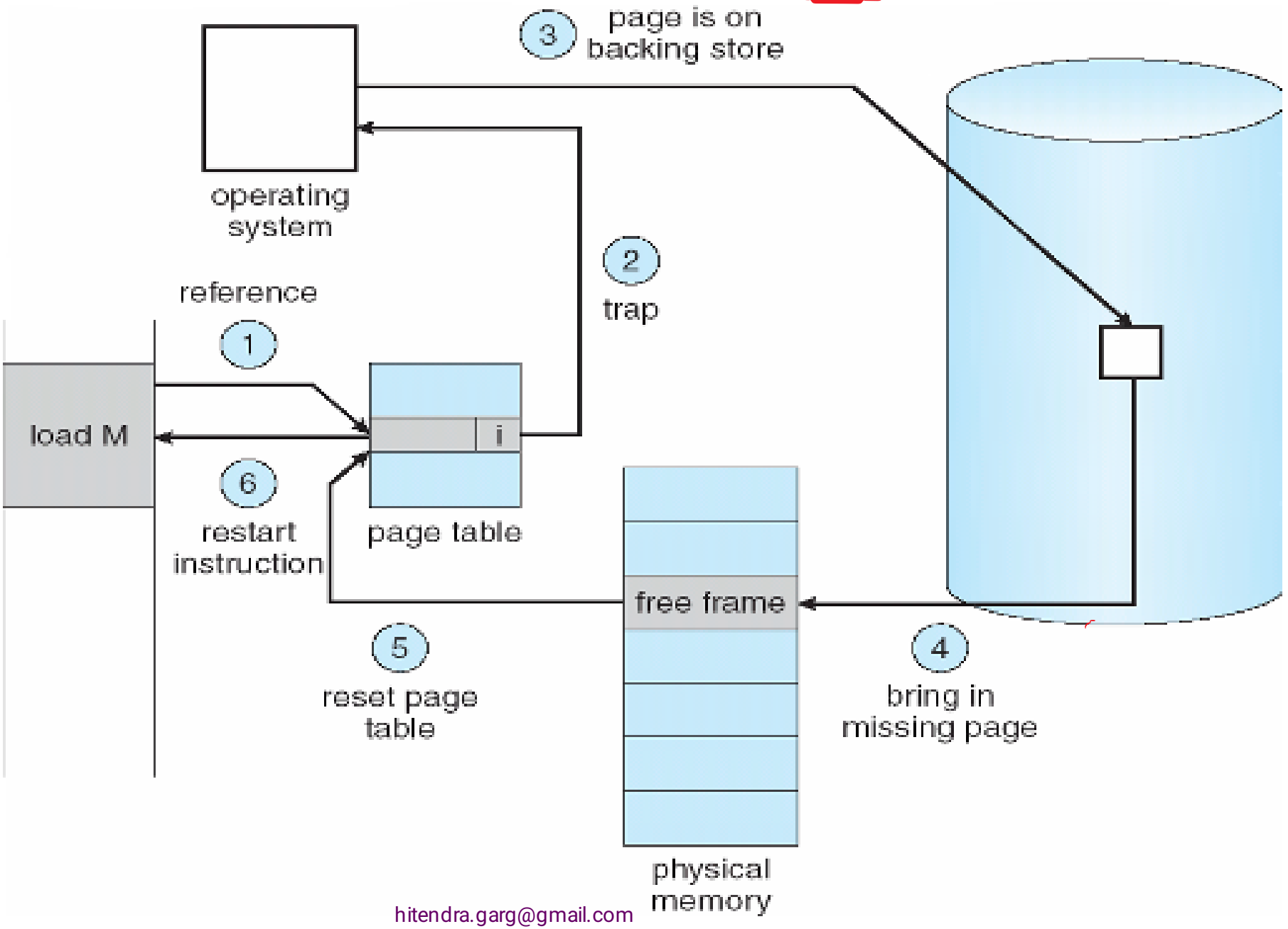
- If there is a reference to a page, first reference to that page will trap to operating system:

  page fault

1. Operating system looks at another table to decide:
   - Invalid reference ⇒ abort[trap to OS ⇒abort process]
   - Just not in memory ⇒ page fault ⇒ service page fault
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

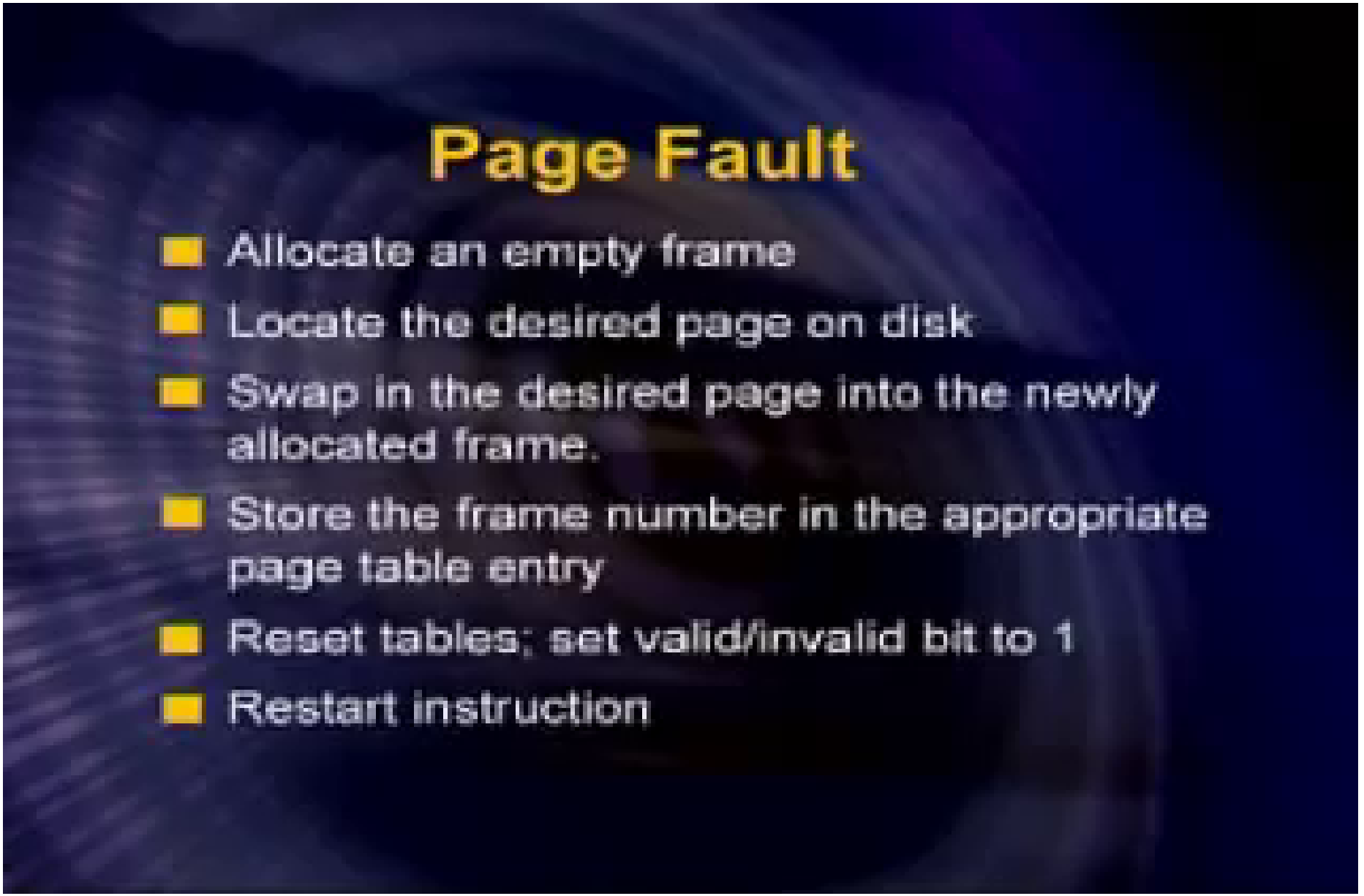
hitendra.garg@gmail.com

Steps in Handling a Page Fault



hitendra.garg@gmail.com

Steps in Handling a Page Fault/Servicing the Page Fault



hitendra.garg@gmail.com

Performance of Demand Paging [Contribution Of Page Fault]

- Page Fault Rate $0 \leq p \leq 1.0$
  - if $p = 0$ no page faults
  - if $p = 1$, every reference is a fault

hitendra.garg@gmail.com

Performance of Demand Paging [Contribution Of Page Fault]

- **Effective Access Time** (EAT)

  EAT = (1 − *p*) x memory access

  \+ *p* (page fault service time i.e, time needed to service the page fault)

Page fault = 0.7 → 0.3

Memory Access time = 100 ns

Page fault service time = 200 ns

EAT = ?

0.3 x 100 + 0.7 x 200

= 30 + 140 = 170 ns

hitendra.garg@gmail.com

Performance of Demand Paging [Contribution Of Page Fault]

- Effective Access Time (EAT)

  EAT = (1 − *p*) x memory access
  + *p* (page fault overhead
  + swap page out
  + latency time
  + swap page in
  + restart overhead
)

Page-fault service time

hitendra.garg@gmail.com

Performance of Demand Paging
[Contribution Of Page Fault]

Memory Access Time =100 nanoseconds

Page Fault Service Time =25 milliseconds

$T_{effective}$=**Effective Access Time=(1-p)*100 +p*(25 milliseconds)**

=(1-p)*100 +p*(25000000)

= 100 + (24999900 *p) [Effective Access Time is directly proportional to the page fault rate "**p**"]

If 1 access out of 1000[place p=1/1000 in above equation**] causes a page fault , effective access time is *25 microseconds* , a slow down by a factor of 250[**How Bad is it ?? , not acceptable ,so minimize the page fault**].**

hitendra.garg@gmail.com

Page Replacement Algorithms

- To choose the victim page[right now discussion is on **local replacement** only]
- Want lowest page-fault rate[means choosing a victim that should not give rise to page fault itself.]
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string

- In all our examples, the reference string is

  1, 2, 3, 4, 5, 3, 4 , 1, 6, 7, 8, 7, 8, 9, 5, 4, 5, 4, 4

Number of frames allocated to this corresponding process =4

hitendra.garg@gmail.com

Page Replacement Algorithms

- In all our examples, the reference string is

  1, 2, 3, 4, 5, 3, 4 , 1, 6, 7, 8, 7, 8, 9, 5, 4, 5, 4, 4

FIFO

Number of frames allocated to this corresponding process =4

1 2 3 4 5 3 4 1 6 7 8 7 8 9 5 4 5 4 4

hitendra.garg@gmail.com

Page Replacement Algorithms

- In all our examples, the reference string is

  1, 2, 3, 4, 5, 3, 4 , 1, 6, 7, 8, 7, 8, 9, 5, 4, 5, 4, 4

Number of frames allocated to this corresponding process =4

														F	F			
1	2	3	4	5	3	4	1	6	7	8	7	8	9	5	4	5	4	4

hitendra.garg@gmail.com

Page Replacement Algorithms

- In all our examples, the reference string is

  1, 2, 3, 4, 5, 3, 4 , 1, 6, 7, 8, 7, 8, 9, 5, 4, 5, 4, 4

Number of frames allocated to this corresponding process =4

F

1 2 3 4 5 3 4 1 6 7 8 7 8 9 5 4 5 4 4

hitendra.garg@gmail.com

Page Replacement Algorithms

- In all our examples, the reference string is

  1, 2, 3, 4, 5, 3, 4 , 1, 6, 7, 8, 7, 8, 9, 5, 4, 5, 4, 4

Number of frames allocated to this corresponding process =4

1 2 3 4 5 3 4 1 6 7 8 7 8 9 5 4 5 4 4

hitendra.garg@gmail.com

Graph of Page Faults Versus The Number of Frames Allocated to a Process.

More the number of free frames allocated to a process, less and less is going to be a page fault rate.

hitendra.garg@gmail.com

FIFO ALGORITHM

hitendra.garg@gmail.com

First-In-First-Out (FIFO) Algorithm

In previous slide , no pre paging is done.

In previous slide , with 4 allocated frames to a corresponding process , 12 page fault occurs.

- *Belady's Anomaly: more frames ⇒ more page faults*
- *Belady's Anomaly: more memory ⇒ more page faults; [Not Logical]*
- *Not in all Page Replacement Algorithms but found in FIFO ; that is why FIFO is rarely used.*
- *"The page fault frequency of a program increases as its memory allocation is increased."*
- *"For some Page Replacement Algorithms , the page fault rate may increase as the number of allocated frames increases."*

Example Depicting Belady's Anomaly

FIFO Algorithm

- Number of frames allocated = 3
- Reference string:

  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- Number of page faults = 9

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

hitendra.garg@gmail.com

Example Depicting Belady's Anomaly

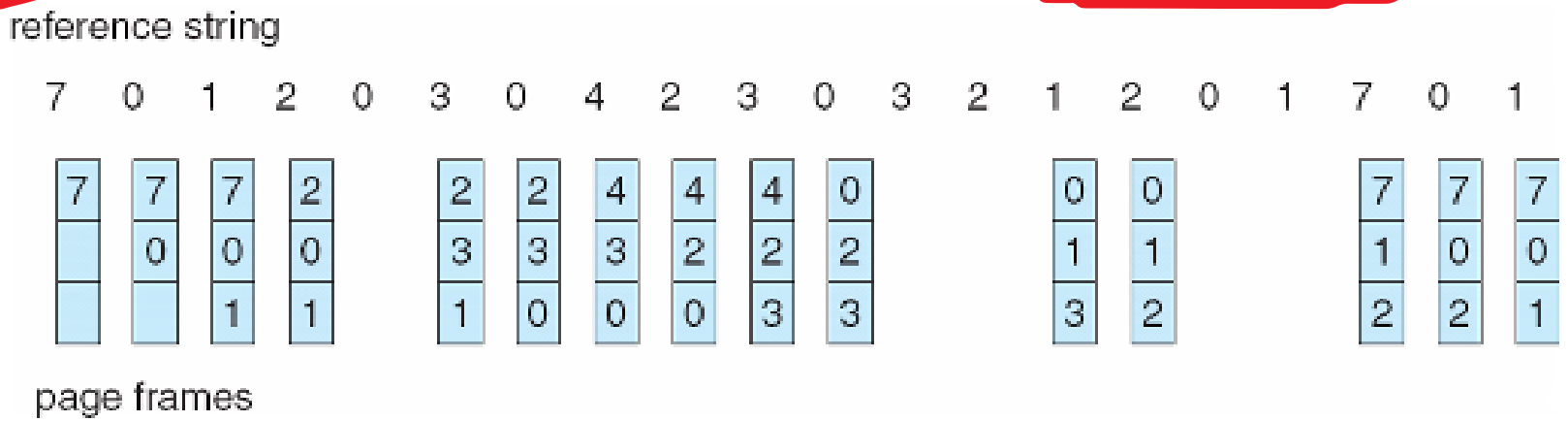
FIFO Algorithm

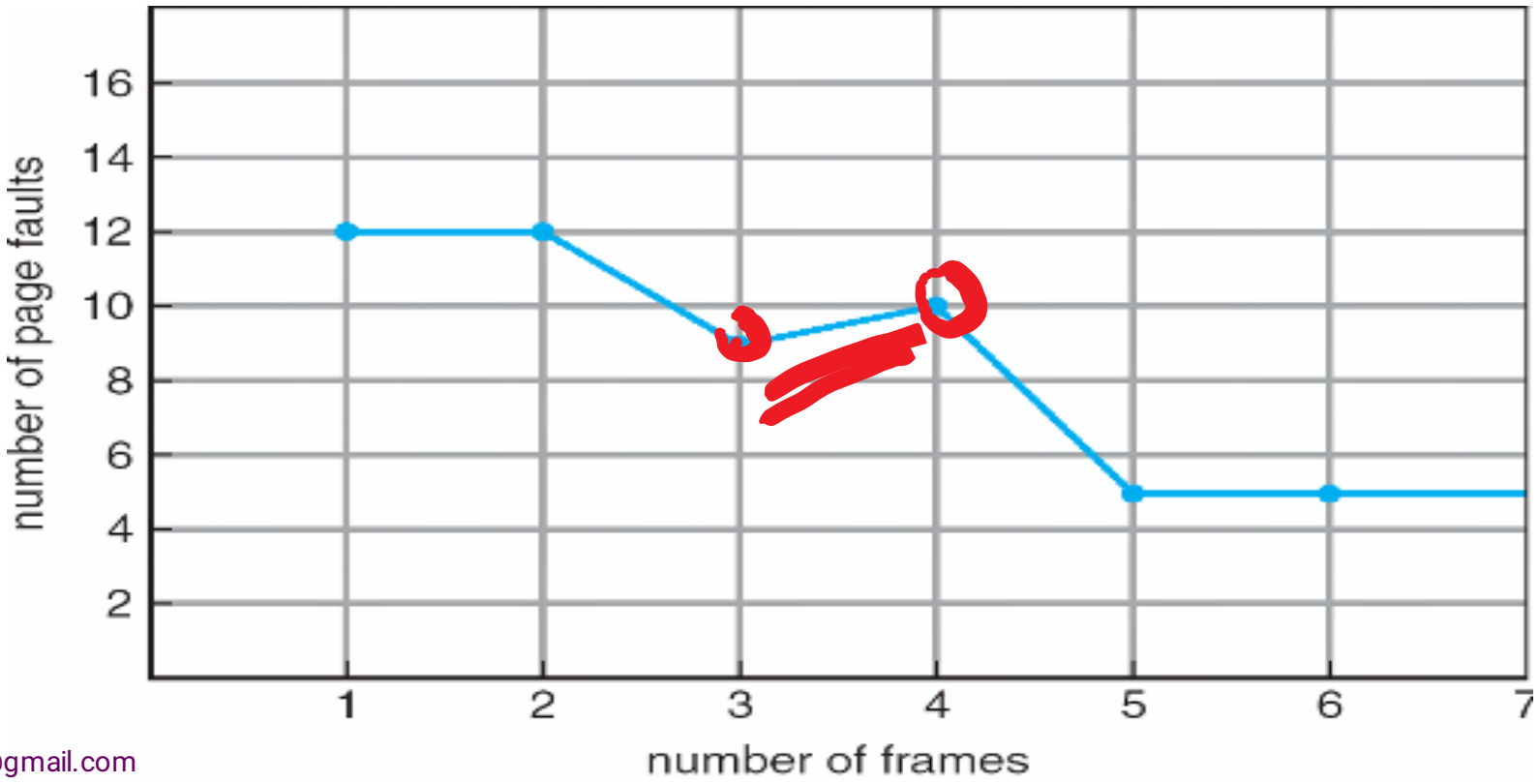
- Number of frames allocated = 4
- Reference string:

  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- Number of page faults = 10

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

FIFO Illustrating Belady's Anomaly





hitendra.garg@gmail.com

Stack Replacement Algorithms

- A class of page replacement algorithms with the following property

" Set of pages in the memory with "n" frames is a subset of the set of pages in memory with "n+1" frames"

Do not suffer from Belady's Anomaly.

Example LRU.

FIFO does not guarantee this.

Stack replacement algorithms are those algorithms which guarantee that with "n" free frames whatever the set of pages the process have in main memory , this set would be the subset of pages in memory if there are "n+1" free frames.

Example Depicting Belady's Anomaly

LRU Algorithm

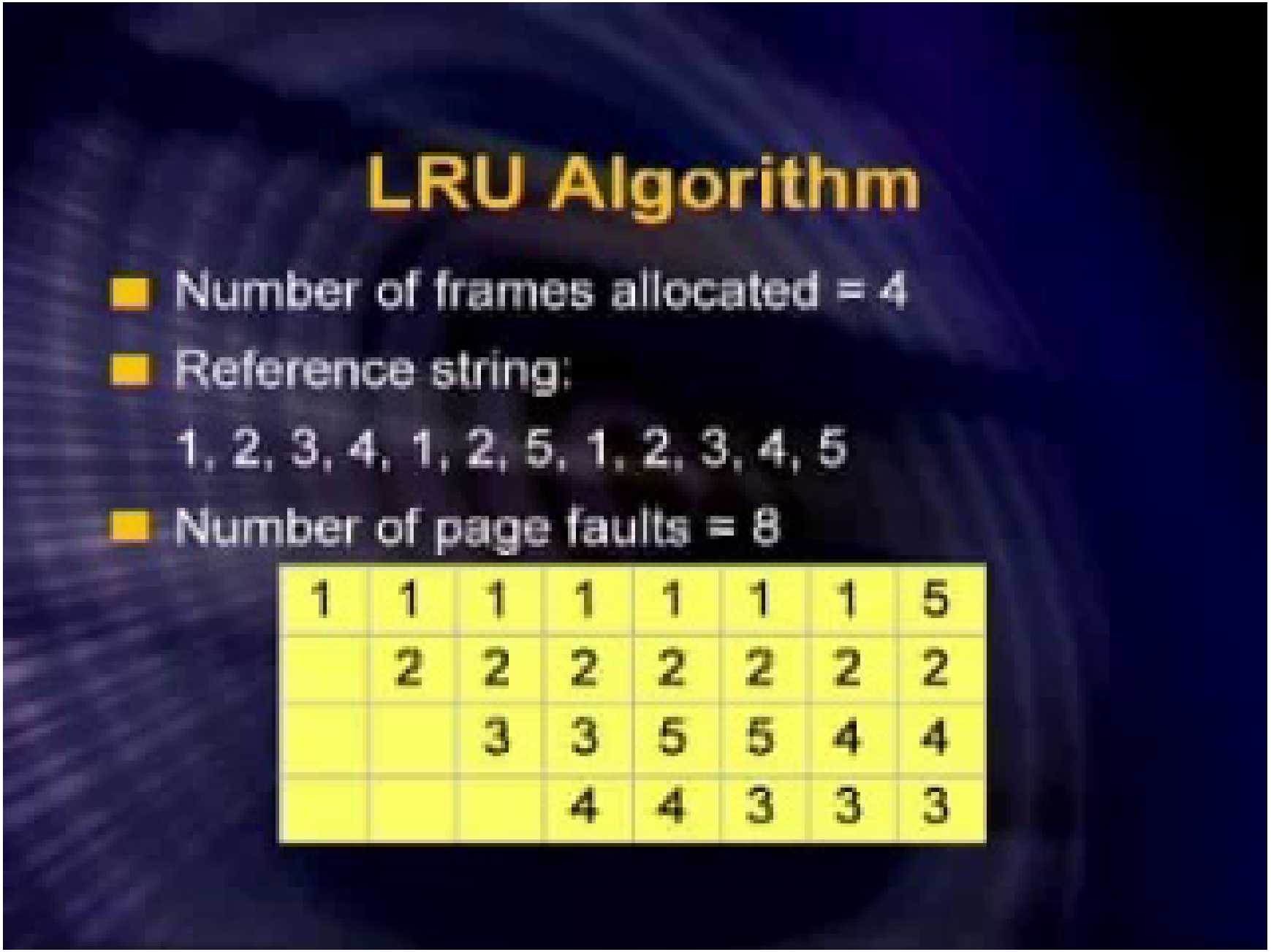
- Number of frames allocated = 3
- Reference string:

  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- Number of page faults = 10

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

hitendra.garg@gmail.com

Example Depicting Belady's Anomaly



hitendra.garg@gmail.com

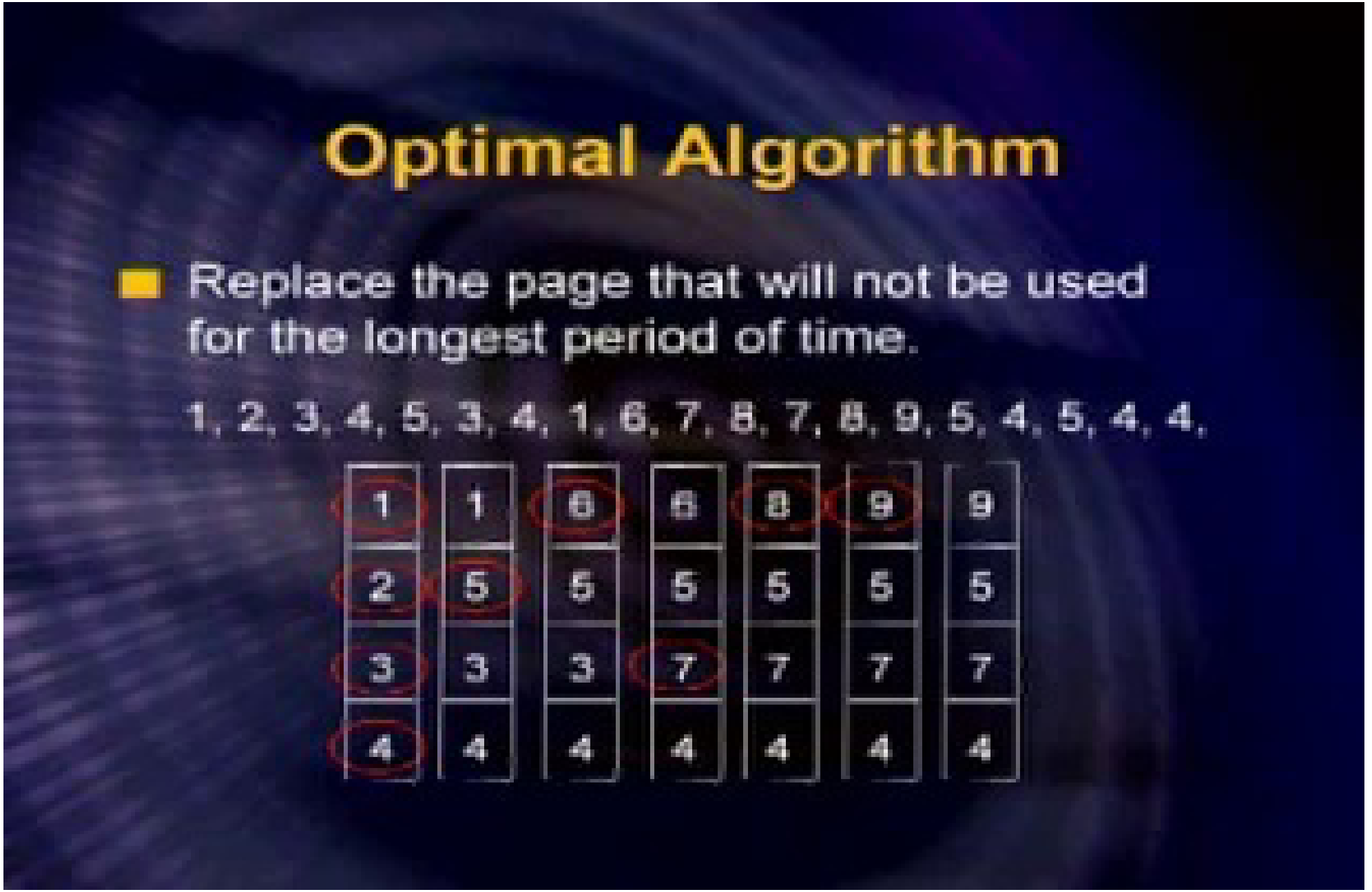
Optimal Algorithm

- **Replace page that will not be used for longest period of time**.
- In case of tie in this algorithm, we use FIFO.
- Best Algorithm, but not practical and implementable[Why ???].
- But how will you predict the future ???[The pages and the sequences required in the future????]. We will know the requirements only when this process is executed.

How do you know this?

- **Used for measuring how well your algorithm performs**[served as a bench mark.].

Optimal Algorithm [Page Fault=9 in this case]



hitendra.garg@gmail.com

Least Recently Used (LRU) Algorithm

- ***Replace the page that has not been used for the longest period of time.***

In optimal move forward towards future to check future references to find the page[victim to be replaced] which will be called/used in the last while in LRU move backward towards past to find the page[victim to be replaced] which was called/used in the first among the pages still in the frames[main memory] i.e. both are mirror image of each other.

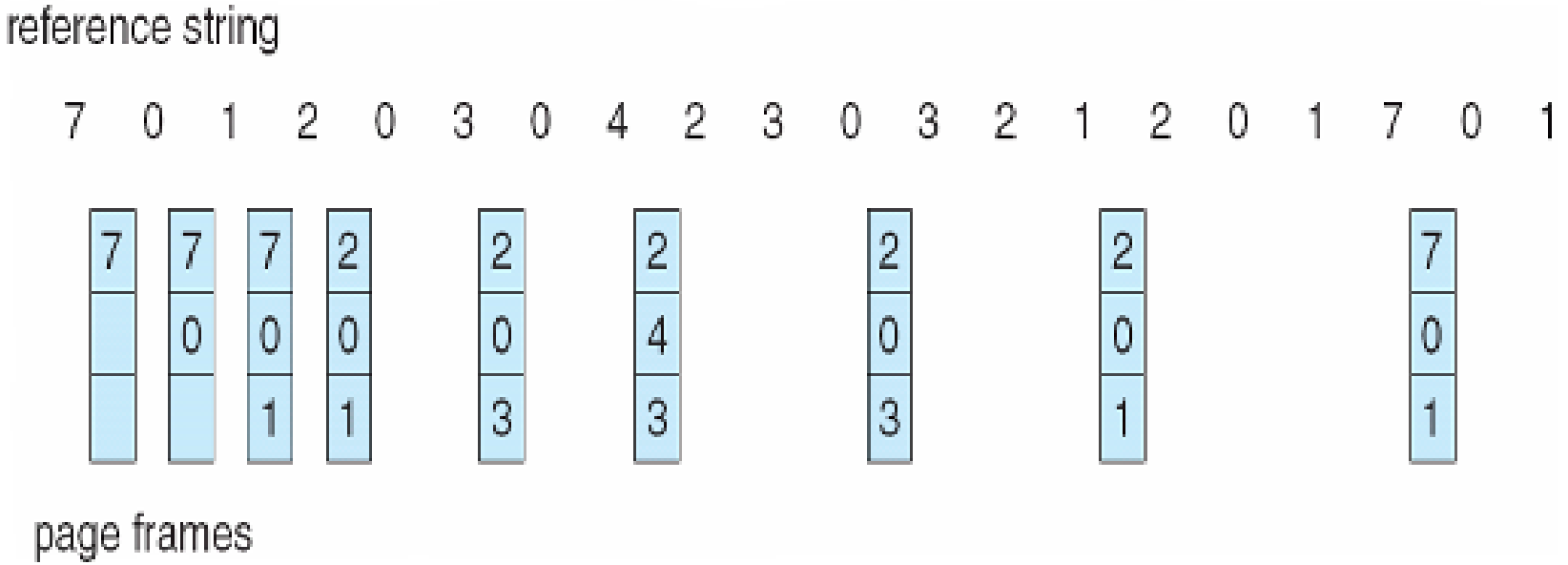
Counter implementation

- Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter
- When a page needs to be changed, look at the counters to determine which are to change

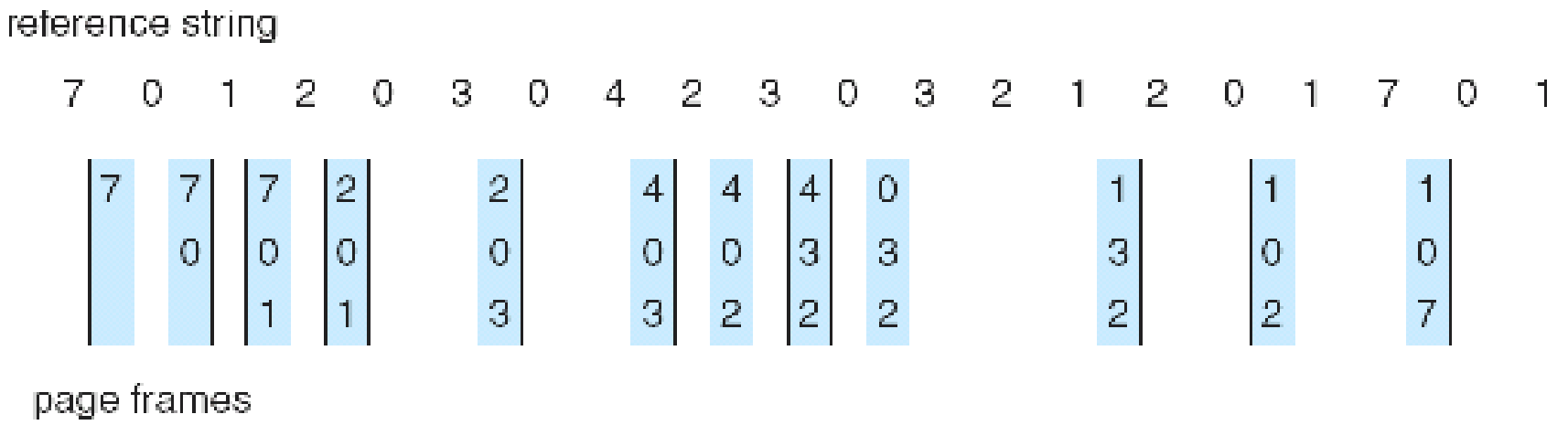
4 Free Frames; 12 Page Faults

hitendra.garg@gmail.com

Optimal Page Replacement



Least Recently Used (LRU) Page Replacement
3 Free Frames; 12 page fault



hitendra.garg@gmail.com

Q. A virtual memory has an address space 18K words, a memory space has 9K words and block sizes of 3K words. The following reference generated by CPU

3 4 2 6 4 7 1 3 2 6 3 5 1 2 3

Show each page reference change if the replacement algorithm used is (a)FIFO (b)LRU © Optimal. Also state which algorithm is better on the basis of page fault?

3 4 2 6 4 7 1 3 2 6 3 5 1 2 3

Q. A virtual memory has a page size of 1K words. There are eight pages and four blocks. The memory page table contains the following entries:

Page	Block/Frame
0	3
1	1
4	2
6	0

Make a list of all virtual addresses (In decimal) that will cause a page fault if used by the CPU.

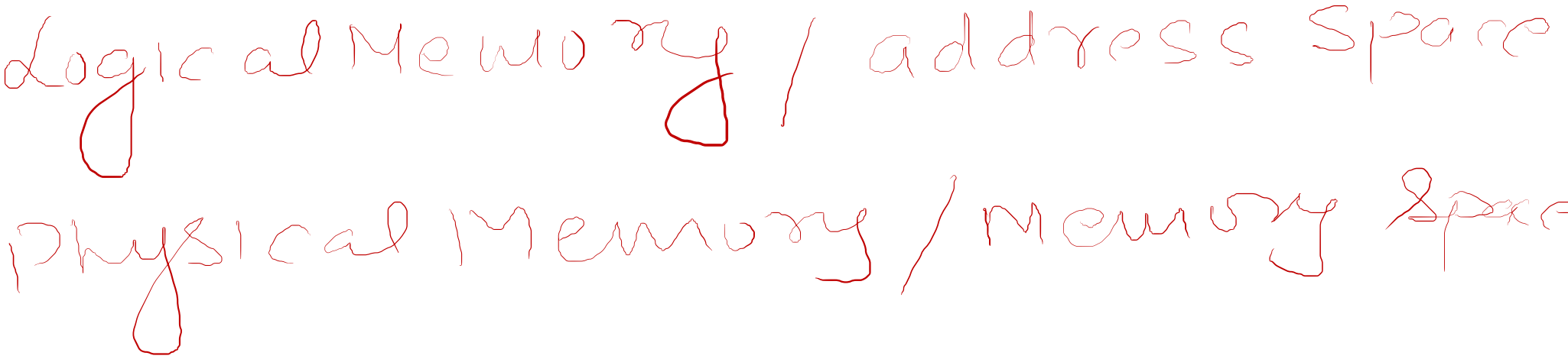
Q. A virtual memory system has an address space of 8K words, a memory space of 4K words, and page and block sizes of 1K words The following page reference changes occur during given time interval.

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

Determine the four pages that are resident In main memory after each page reference change if the replacement algorithm used Is (a) FIFO; (b) LRU.

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

An address space is specified by 24 bits and the corresponding memory space by 16 bits. How many words are there in the address space. How many words are there in the memory space. If a page consists of 2K words, how many pages and blocks are there in the system?



Q. The logical address space in a computer system consists of 128 segments each segment can have up to 32 pages of 4K words In each. Physical memory consists of 4K block of 4K words In each. Formulate the logical and physical address formats.

Consider a swapping system in which memory consists of the following hole sizes in memory order: 10 MB, 4 MB, 20 MB, 18 MB, 7 MB, 9 MB, 12 MB, and 15 MB. Which hole is taken for successive segment requests of

(a) 12 MB
(b) 10 MB
(c) 9 MB

for first fit? Now repeat the question for best fit, worst fit, and next fit. (2)

A computer system has a 36-bit virtual address space with a page size of 8K, and 4 bytes per page table entry.

a) How many pages are in the virtual address space?

b) What is the maximum size of addressable physical memory in this system?

A Computer provides each process with 65536 bytes of address space divided into pages of 4096 bytes. A particular program has instruction of 32768 bytes , a data size of 16386 bytes and a stack size of 15870 bytes. Will this program fit in the address space? If the page size were 512 bytes , would it fit? Remember that a page may not contain part of two different segments.

A CPU generates 32-bit virtual addresses. The page size is 4 KB. The processor has a translation look-aside buffer (TLB) which can hold a total of 128 page table entries and is 4-way set associative. The minimum size of the TLB tag is:
(A)11 bits (B) 13 bits (c) 15 bits (D) 20 bits

Answer Size of a page = 4KB = 2^12
Total number of bits needed to address a page frame = 32 – 12 = 20
If there are ‘n’ cache lines in a set, the cache placement is called n-way set associative.
Since TLB is 4 way set associative and can hold total 128 (2^7) page table entries, number of sets in cache = 2^7/4 = 2^5.
So 5 bits are needed to address a set, and
15 (20 – 5) bits are needed for tag.

Page Fault & No Free Frame

Page Replacement

Replacement Algorithm(minimize number of page fault ; minimize the page fault rate).

Performance with replacement.

Victim Selection

Criteria

Local Vs Global.

hitendra.garg@gmail.com

Basic Page Replacement

1. If no free frame is available on a page fault , replace a page in memory to load the desired page.
2. Page fault service routine is modified to include page replacement.
3. Replacement can be of
    1. several pages[by the swapper-medium term scheduler in case of page of process of high priority to be brought into main memory]
    2. Choosing a victim[one page only that depends upon whether the victim is a “dirty ” page or not.]. If victim is a dirty page , it will be swapped out ; otherwise the page to be brought in is overwritten over it.
    3. Choosing a 1 page victim, that victim can be of the same process whose page is to be brought in[Local replacement] or any other process in RAM[Global replacement] .

- The page-fault service routine is now modified to include page replacement:

1. Find the location of the desired page on the disk.
2. Find a free frame:
   - a. If there is a free frame, use it.
   - b. Otherwise, use a page-replacement algorithm to select a *victim frame.*
   - c. Write the victim page to the disk; change the page and frame tables accordingly.
3. Read the desired page into the (newly) free frame; change the page and frame tables.
4. Restart the user process.

Basic Page Replacement

1. Use modify(dirty) bit to reduce overhead of page transfers- only modified pages are written to disks.
    1. Set by hardware when data is written to a page.
    2. Checked by OS at page replacements.

What happens if there is no free frame?

- Page replacement – find some page in memory, but not really in use, swap it out
  - algorithm
  - performance – want an algorithm which will result in minimum number of page faults
- Same page may be brought into memory several times

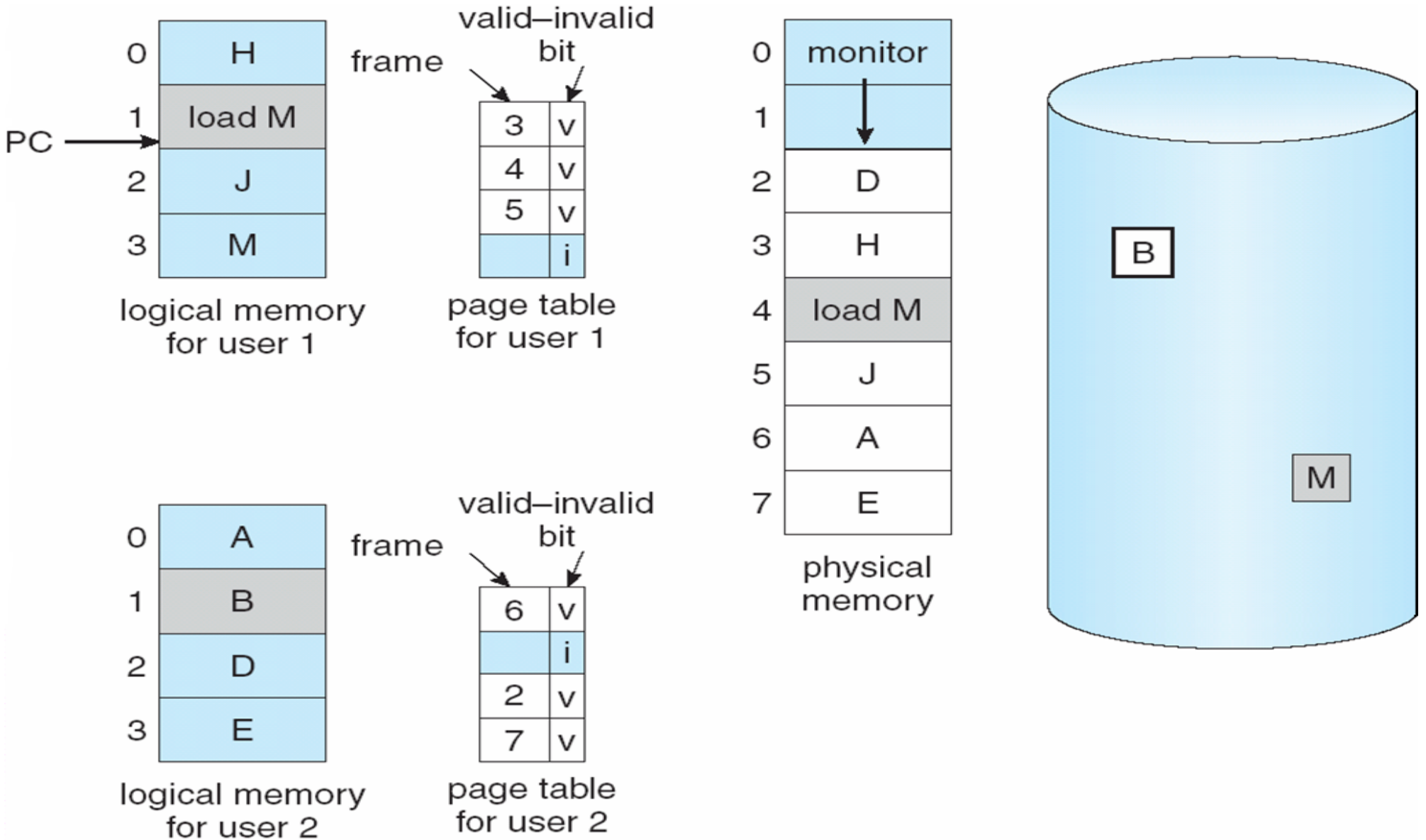
Basic Page Replacement

1. Find the location of the desired page on disk

2. 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 a victim frame
   [*Different methods to choose a victim when there is no free frame available, with the objective of minimizing the rate of page fault , are various page replacement algorithms*]
3. Bring the desired page into the (newly) free frame; update the page and frame tables

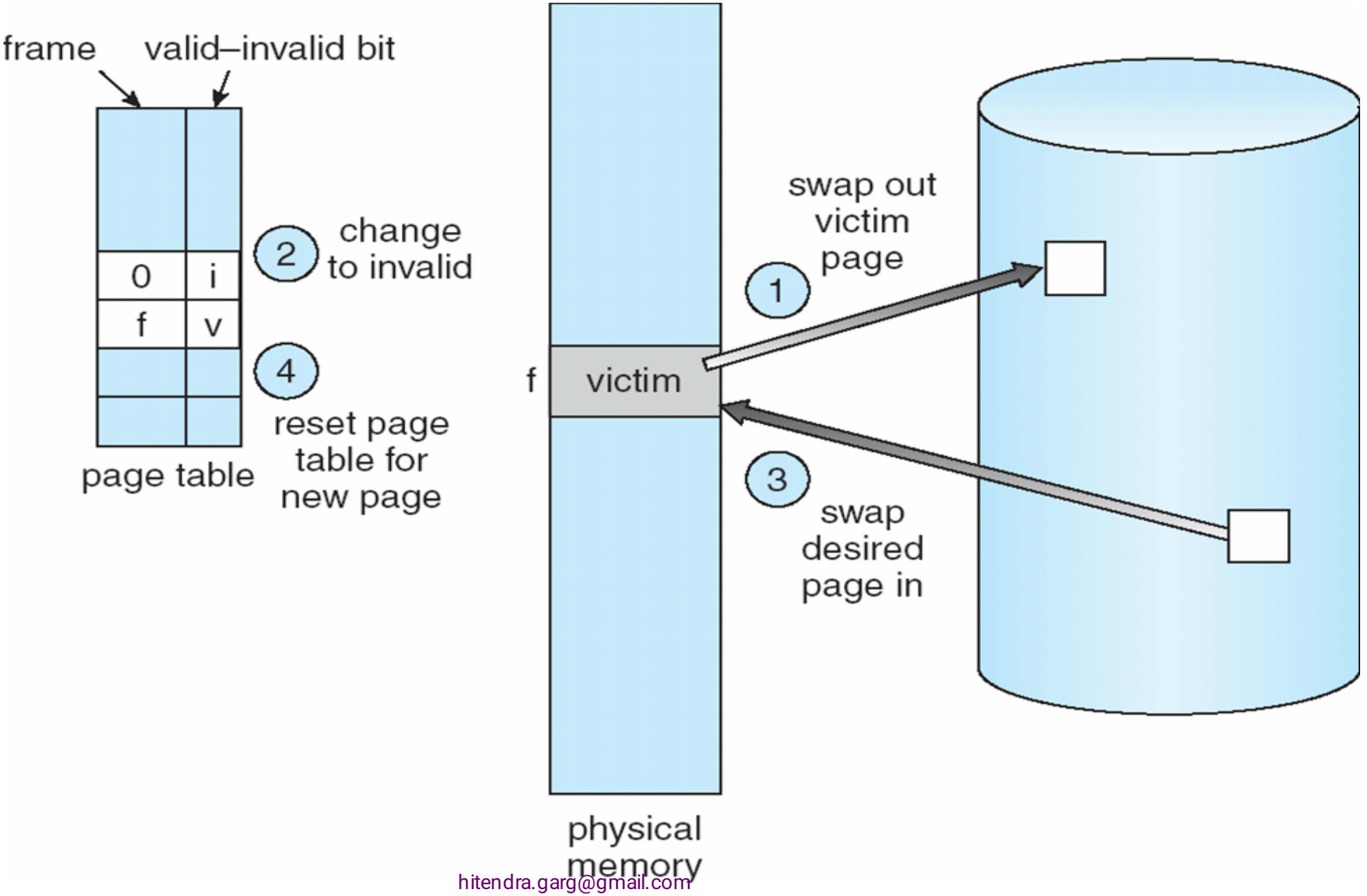
4. Restart the process

Need For Page Replacement



hitendra.garg@gmail.com

Page Replacement



Page Replacement

- Prevent over-allocation of memory by modifying page-fault service routine to include page replacement
- Use **modify (dirty) bit** to reduce overhead of page transfers – only modified pages are written to disk
- Page replacement completes separation between logical memory and physical memory – large virtual memory can be provided on a smaller physical memory