Memory Management & Virtual Memory

Unit - 4

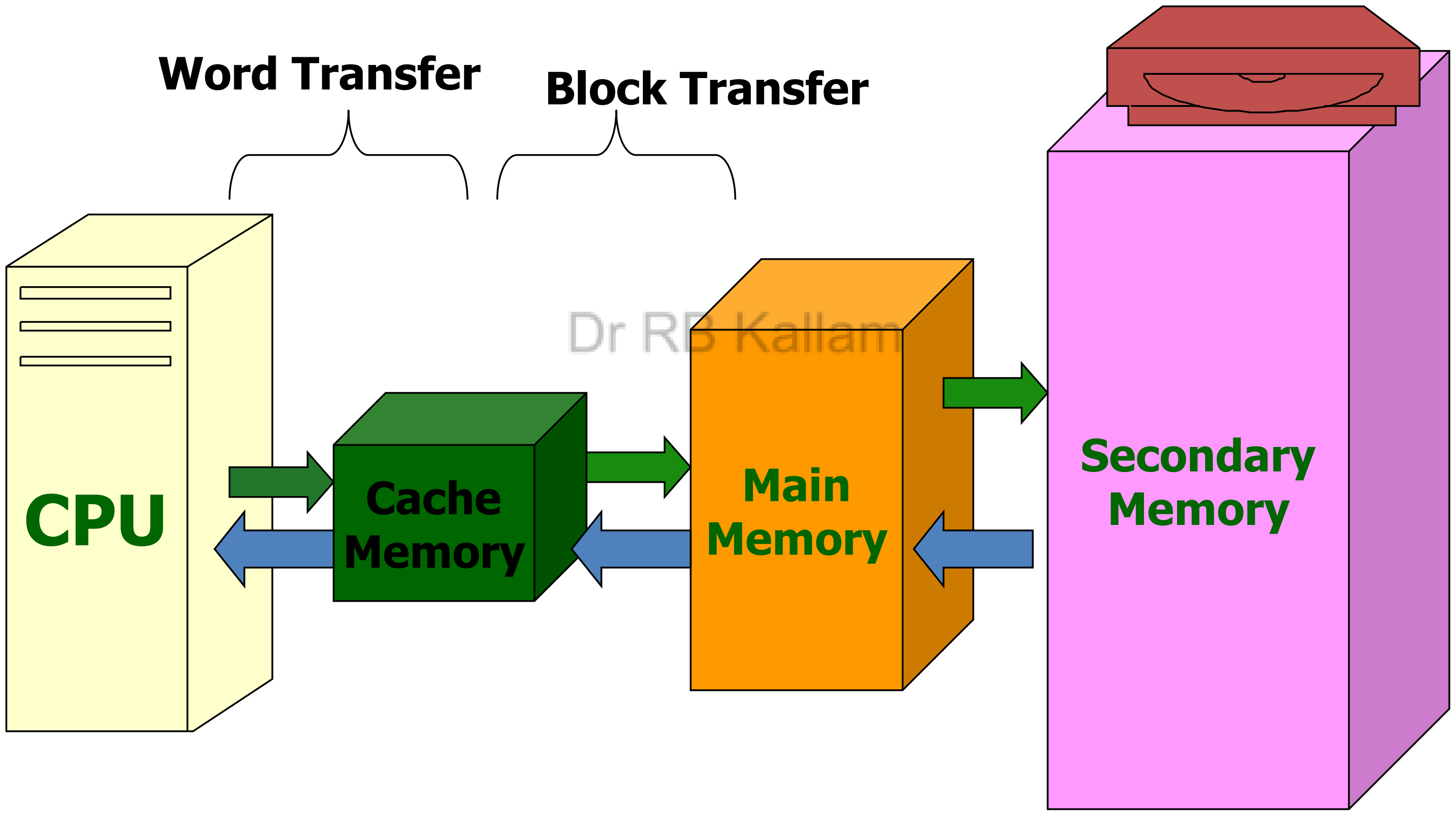
A Presentation by

Dr Capt Ravindra Babu Kallam

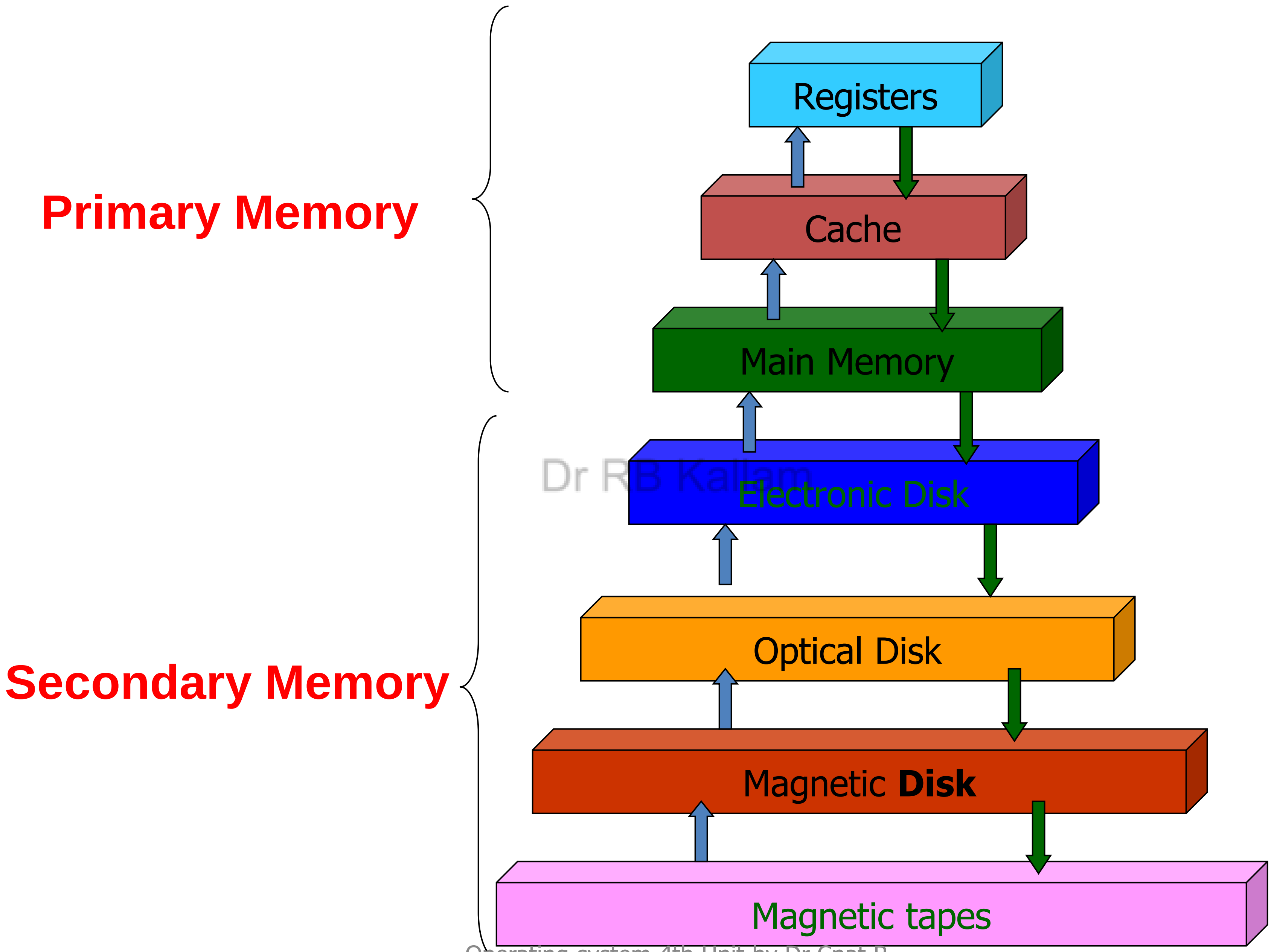
Prof &Head CSE, KITS(S)

Topics to be covered:

- Logical versus Physical Address Space, Swapping,
- Contiguous Allocation
- Paging, Segmentation
- Segmentation with Paging
- Virtual memory
  - Demand Paging, Page fault, Page Replacement
  - Thrashing
  - Page Replacement Algorithms.



Transition between memory's and CPU



Storage Device Hierarchy

Memory Management Requirements:

- **Relocation:**
  - Programmer does not know where the program will be placed in memory when it is executed .
  - While the program is executing, it may be swapped to disk due to several reasons and returned to main memory at a different location is called relocation.

- **Protection:**
  - Processes should not be able to reference memory locations in another process without permission.
  - It is also one of the major requirement in Memory management.
  - Memory protection requirement must be satisfied by the processor (hardware) rather than the operating system (software)

- **Sharing:**
  - A protection mechanism must have the flexibility to allow several processes to access the same portion of memory .
  - Processes that are cooperating on some task may need to share access to the same data structure.
  - -- The memory management system must therefore allow controlled access to shared areas of memory without compromising essential protection.
- **Logical organization:**

  - Main Memory in a computer system is organized as a linear, or one-dimensional order of a sequence of bytes or words. Secondary memory, at its physical level, is similarly organized.
  - Programs are written in modules
  - Modules can be written and compiled independently
  - Different degrees of protection given to modules (read-only, execute-only) and these modules can be shared among the processes

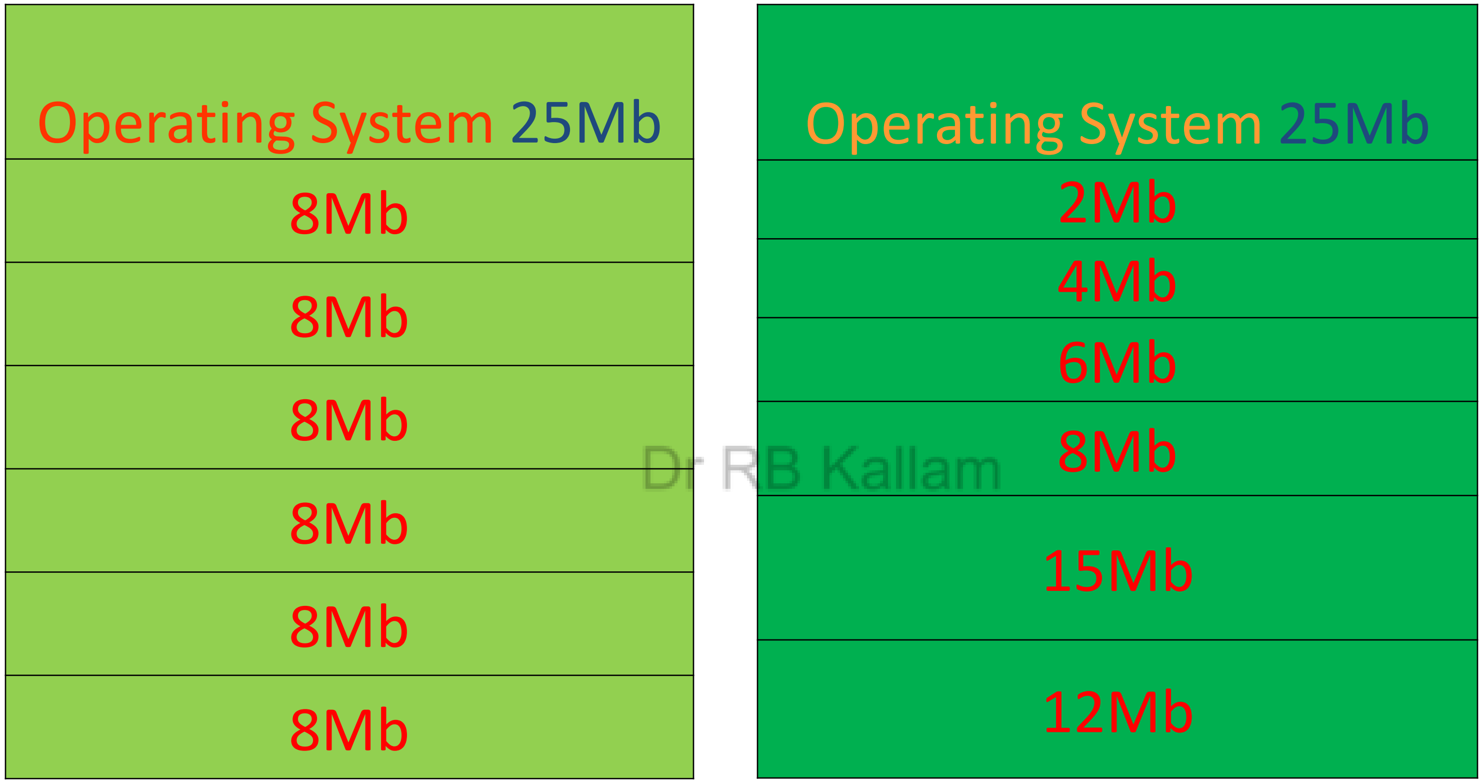
Physical organization:

- Computer memory is Physically organized into at least two levels, referred to as main memory and secondary memory.
- Main memory provides fast access at relatively high cost. In addition, main memory is volatile; that is, it does not provide permanent storage.
- Secondary memory is slower and cheaper than main memory and is usually not volatile.
- Thus secondary memory of large capacity can be provided for long-term storage of programs and data, while a smaller main memory holds programs and data currently in use for short duration.
- Memory available for a program plus its data may be insufficient, Overlaying allows various modules to be assigned the same region of memory

Loading Programs in to Main Memory:

- Fixed Partitioning or Static Partitioning
  - equal partitioning
  - unequal partitioning
    - **Internal fragmentation**
    - **Placement algorithms**
      - » ***First Fit***
      - » ***Next Fit***
      - » ***Best Fit***
      - » ***Worst Fit***
- Dynamic Partitioning
  - External fragmentation
  - Memory Compaction

Fixed Partitioning or Static Partitioning



a) Equal Size Partitions b) Un-Equal Size Partitions

Note: OS occupies some fixed portion of the main memory and the rest of the memory is available for use by multiple processes

Equal partitioning

- In this memory is divided in to equal size partitions
- Any process whose size is lees than or equal to the partition size can be loaded in to any available partition

Disadvantages:

-If a process is bigger than the partition we cannot allocate the space.

-If a process is smaller than the partition it leads to internal fragmentation .

Note: Internal Fragmentation: If a space is left unused with in the partition is called internal fragmentation.

Unequal partitioning

- To reduce the internal fragmentation in previous approach, now we divide the memory into unequal partitions as shown in the previous diagram.
- Use placement algorithms (First Fit, Next Fit and Best Fit) to allocate the memory partition to the processes.

Disadvantage:

-It suffers with the internal fragmentation

Placement algorithms

- First-Fit: It begins to scan memory from the beginning and chooses the first available block that is large enough.

- Next- Fit: It begins to scan memory from the location of the last placement and choose the next available block that is large enough.

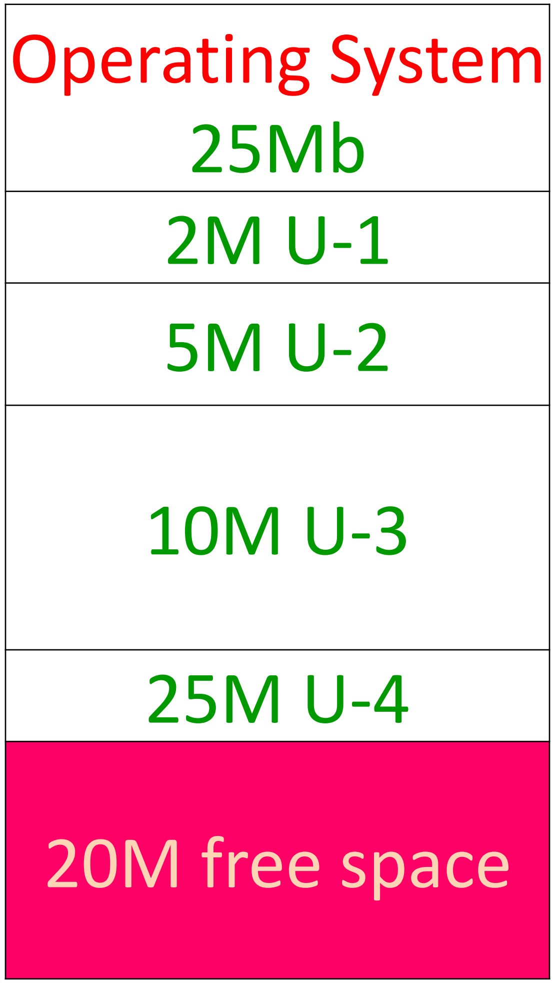
- Best Fit: It chooses the block that is closest in size to the request.
- Worst Fit: It chooses the largest leftover hole.

Dynamic Partitioning

- In this memory is allocated at run time.
- When a process is brought in to the main memory, it is allocated exactly as much memory as it requires.

Consider an example:

- As shown in the diagram, initially we could able to accommodate U1 to U4 in the main memory as per their requirements but we could not able to load U5, because it requires 22Mbytes and the available space is only 20Mb.
- Let us assume that U1 completed its execution and left the M.M and a hole of 2Mb has created and the total free space available in M.M is now 22Mb.
- Even though 22Mb exactly matches with U5 requirement we cannot accommodate U5 in M.M, because memory space is not available as a single block of 22Mb and we do not have a method to divide the process in to parts and it leads to External fragmentation



Ex:

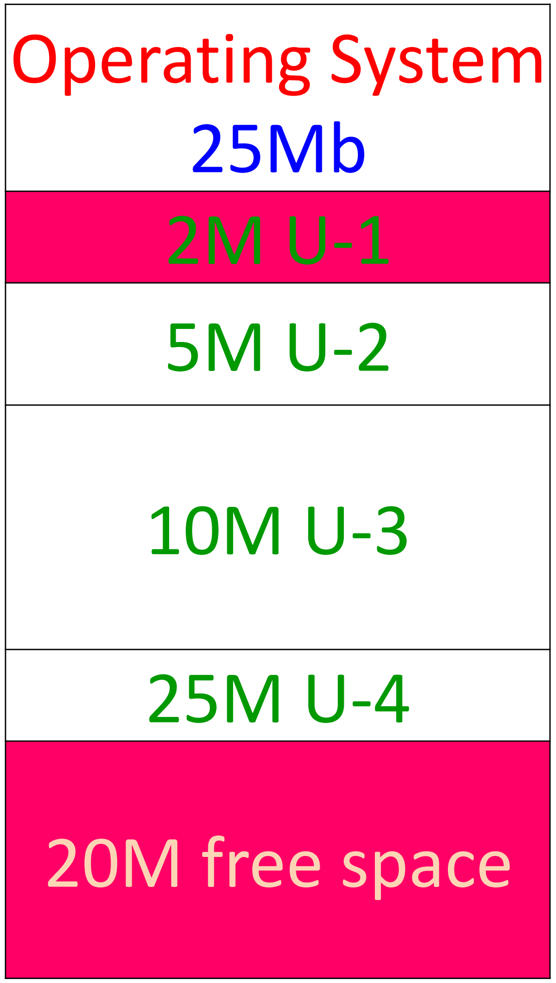
User 1 -- 2M

User 2 -- 5M

User 3 -- 10M

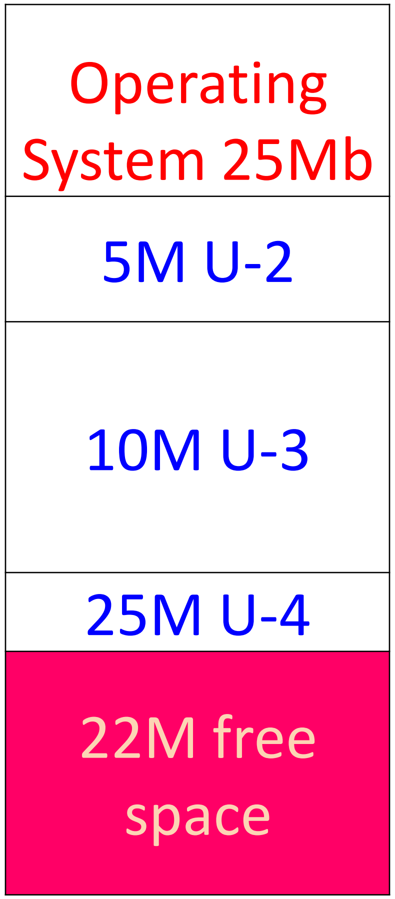
User 4 -- 25M

User5-- 22M



External fragmentation: If a block is left unused between two partitions is called External fragmentation, here 2Mb space is the external fragmentation.

- To avoid External fragmentation we have an approach called Memory Compaction.
- Memory Compaction: Collecting all unused blocks to a common location is called Memory compaction.
- With this, even though we can solve external fragmentation it is not an efficient method.
- Because it need dynamic relocation capability and it is a time consuming process.



- Simple Paging.
  - Page
  - Frame or Page frame
  - Page table,   Internal fragmentation
  - Logical to Physical address conversion
  - Multilevel page table
  - Inverted page table
  - TLB
- Simple Segmentation
  - External fragmentation
  - Memory Compaction
  - Logical to Physical address conversion
- Combined Paging and Segmentation

Simple Paging:

- It is similar to the static partitioning. It avoids external fragmentation and the need of compaction.
- The Main memory is partitioned into equal size chunks that are relatively small, and each process is also divided into small fixed size chunks of the same size.
- The chunks of the process is known as pages, could be assigned to available chunks of the memory is called frames or page frames.
- Consider an ex: process A has 3, B has 2, C has 4 and D has 4pages and available frames are 12 in M.M.
- Also assume that B has completed its task and swapped out as shown in fig c.
- For each user O.S maintains a separate page table for reference, which contains two entries, p# and F#
- The pages in the frames need not to be stored continuously.

Assigning Process pages to free frames in the main memory:

Frame #	M.M
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

a) Twelve available frames

Frame #	M.M
0	A0
1	A1
2	A2
3	B0
4	B1
5	C0
6	C1
7	C2
8	C3
9	
10	
11	

b) Load process A,B&C

Frame #	M.M
0	A0
1	A1
2	A2
3	
4	
5	C0
6	C1
7	C2
8	C3
9	
10	
11	

C) Swap out B

Frame #	M.M
0	A0
1	A1
2	A2
3	D0
4	D1
5	C0
6	C1
7	C2
8	C3
9	D2
10	D3
11	

D)Load Process D

Page tables:

P#	F#
0	0
1	1
2	2

Process table A

P#	F#
0	3
1	4

Process table B

F#
11

Free frame list

P#	F#
0	5
1	6
2	7
3	8

Process table C

P#	F#
0	3
1	4
2	9
3	10

Process table D

Logical to Physical address conversions:

Logical to Physical address translation in paging.

- Logical Address: It is a temporary address; a reference to the memory location independent of the current assignment of data to memory; a translation must be made to a physical address before the memory access.
- Physical Address: The absolute location of a unit of data in memory.

The following steps are needed for address translation:

- ➢ Extract the page number as the leftmost n bits of the logical address.
- ➢ Use the page number as an index into the process page table to find the frame number, k .
- ➢ Finally, The physical address is easily constructed by appending the frame number to the offset.

Address translation in paging

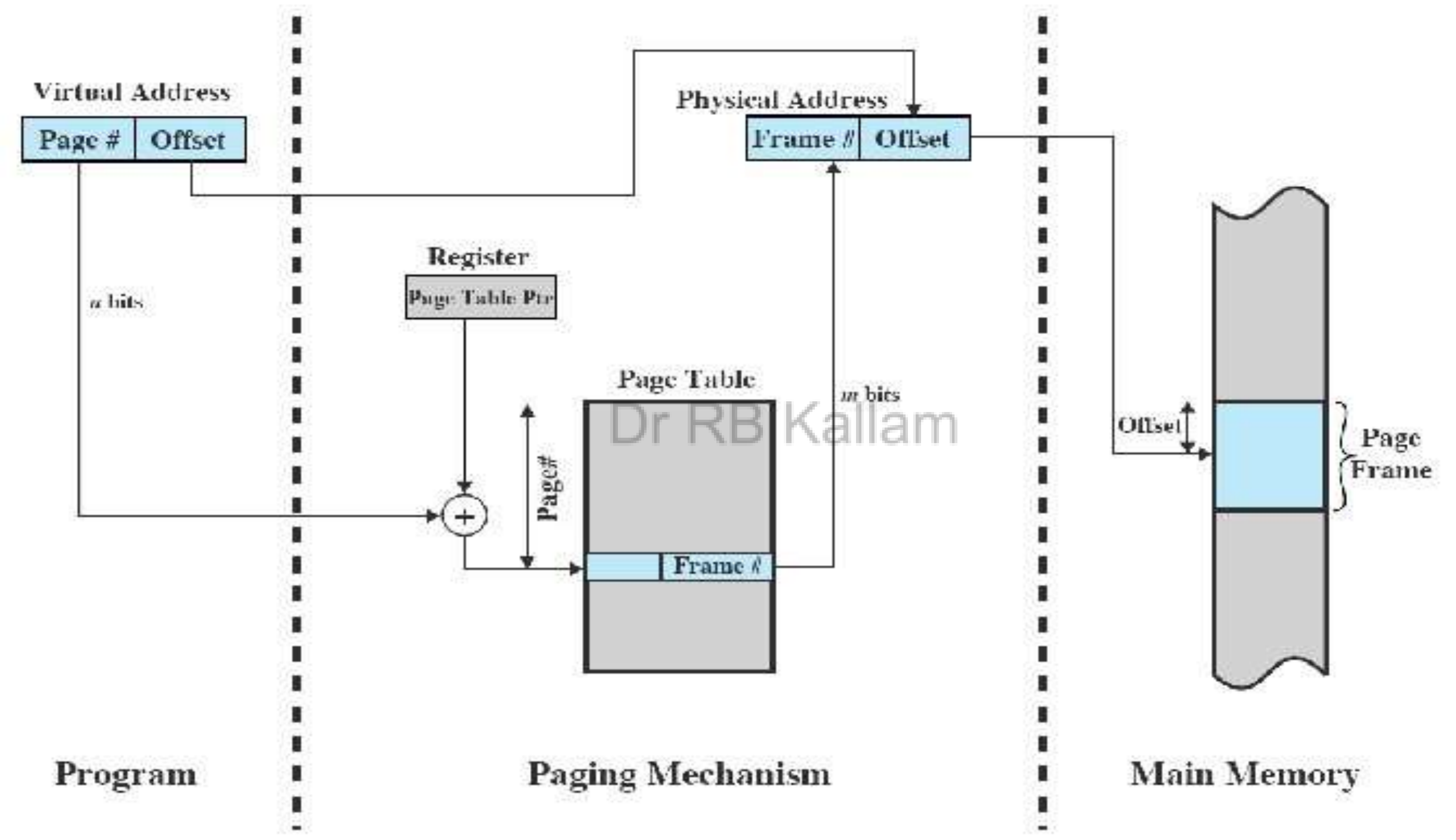
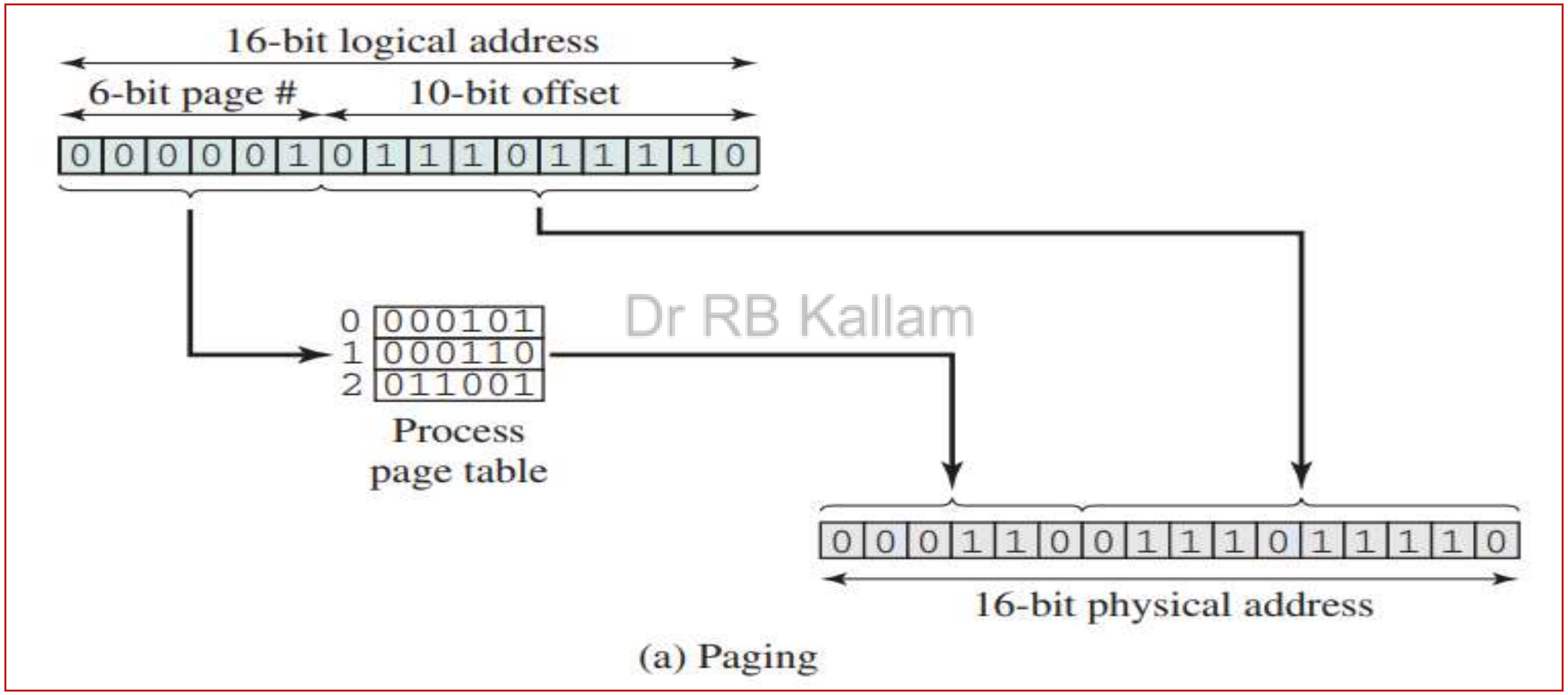


Figure 8.3 Address Translation in a Paging System

typically n > m

7

Ex: Let us assume that a user has 64pages, its corresponding frames in sequence are 000101, 000110, 011001, etc and the logical address is “0000010111011110” convert this into physical address.

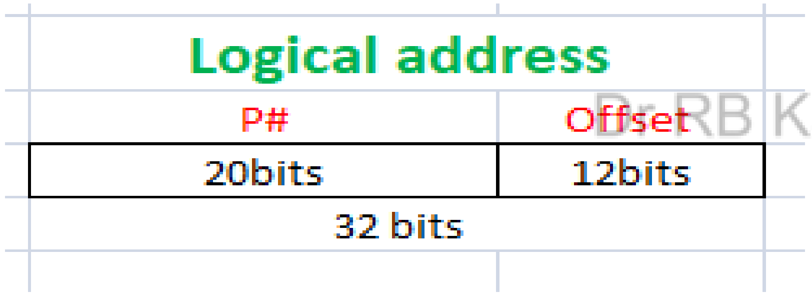


(a) Paging

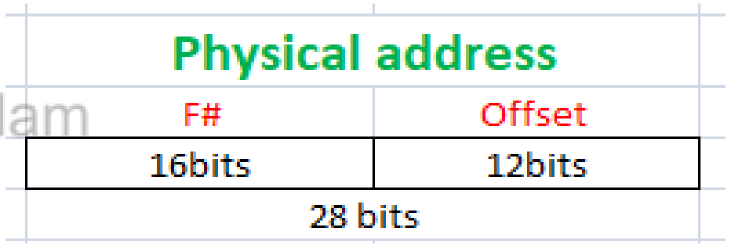
H/W:Ex2: Let us assume that a user has 7 pages, its corresponding frames in sequence are 001, 011, 010, etc and the logical address is “0000110111001110” convert this into physical address.

Multilevel Paging

- Most modern computer systems support a very large logical address space (2^{32} to 2^{64}). In such an environment the page table itself becomes excessively large.
- For example, consider a system with a 32-bit logical address space. If the page size in such a system is 4Kbytes (2^{12}), then a page table may consist of up to 1 million entries (2^{32} -2^{12} =2^{20}).



Logical address	
P#	Offset
20bits	12bits
32 bits	



Physical address	
F#	Offset
16bits	12bits
28 bits	

- If the physical address (Main memory) is of 2^{28} bits, because the page size is 4Kbytes each, then the frame size is also 4Kbytes which can be written as is 2^{12} . With this the offset value =12 and the frame number is in 28-12=16bits=2bytes.
- With this the frame number in the page table is of 2bytes and the page table is having 2^{20} entries, Hence the total buffer required for 1 page table to store is 2^{20} X2Bytes= 2MB, where as the size of the frame is only 4KB.
- So we cannot fit the page table in a single frame.

- To solve this problem a page table can be divided in to multiple level.
- In our example, the page number is further divided into a 9-bit page number (2MB/4KB= $2^{21}/2^{12}=2^9$)(Outer page table) and a 11-bit page offset ($2^{20}-2^9=2^{11}$)or (4KB/2B)(Inner page table). Thus, a logical address is as follows:

page number		Page Offset
P1	P2	d
09	11	12

- Where P1 is an index into the outer page table, and P2 is the displacement within the inner page table. The address-translation scheme for this architecture is shown in Figure below.

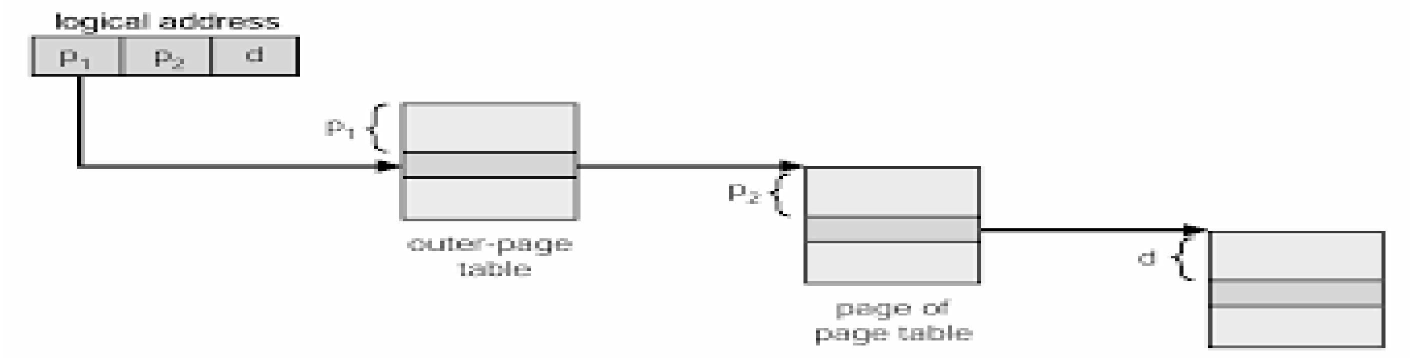


Figure . Address translation for a two-level 32-bit paging architecture.

Ex:2: Two level paging system given in Stallings

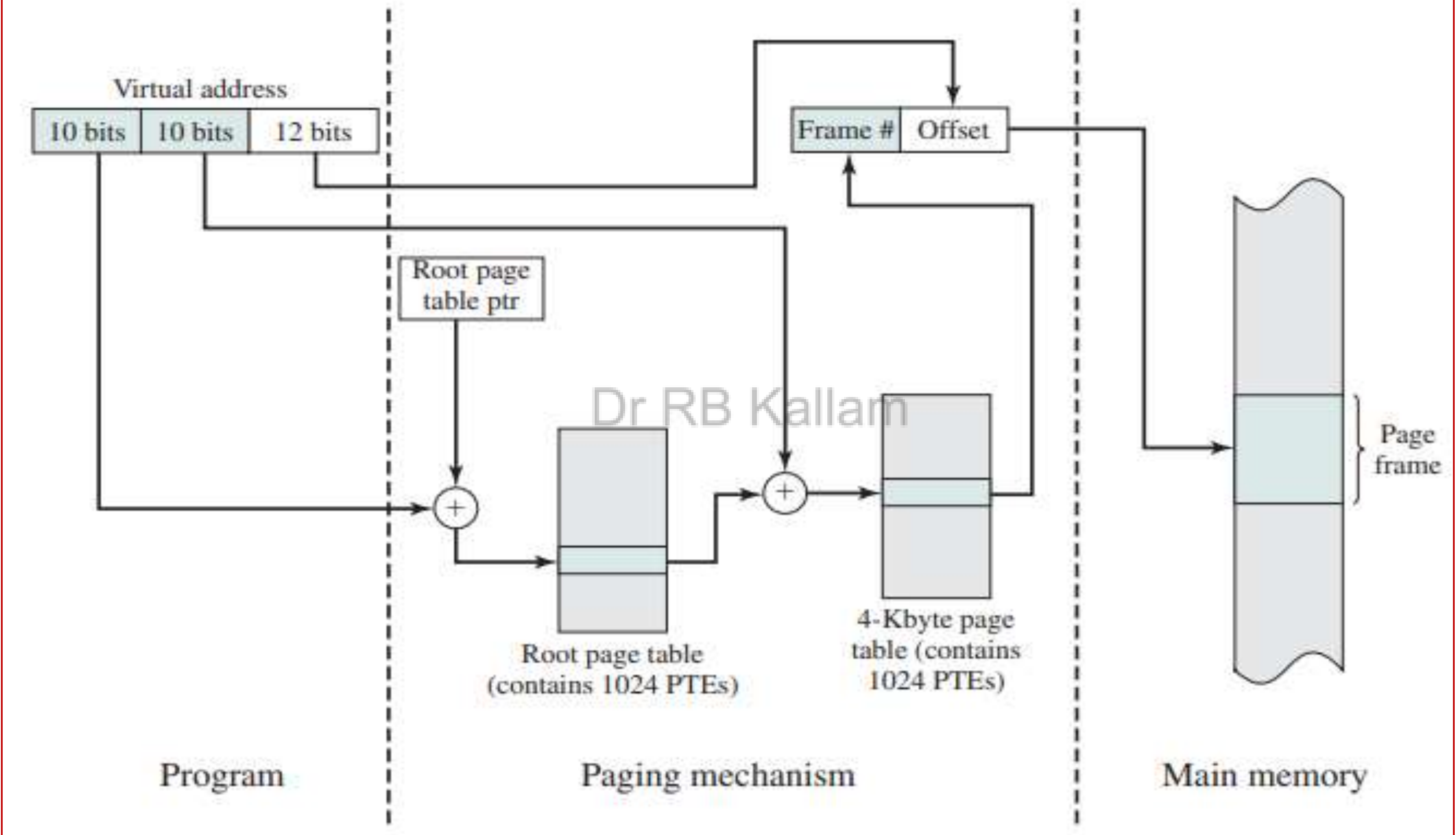
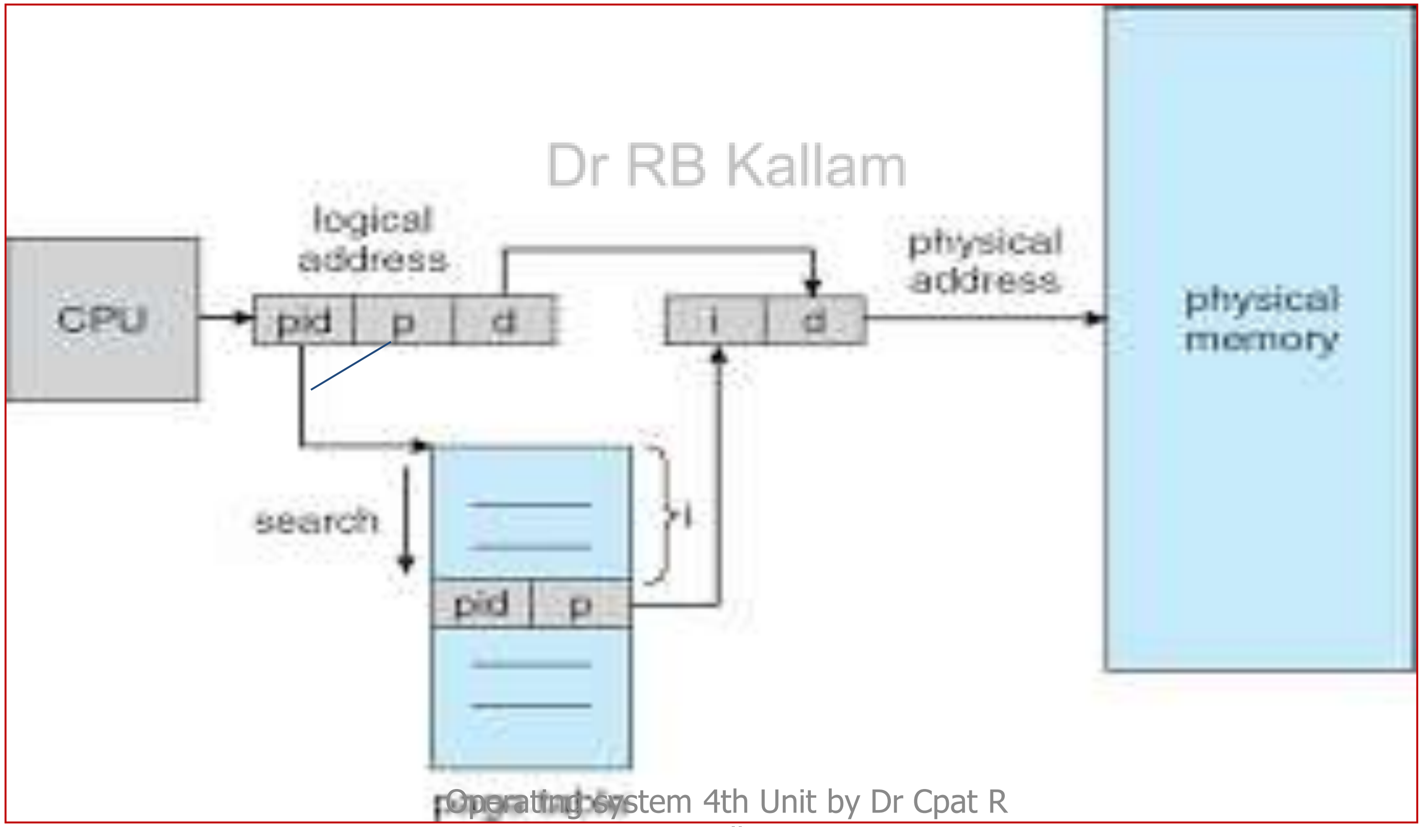


Figure 8.5 Address Translation in a Two-Level Paging System

Inverted page table

- Usually, each process has an associated page table. The page table has one entry for each page that the process is using. Storing all these page tables of each process in main memory leads to the wastage of memory.
- To solve this problem, an inverted page table is used.
- It is a global page table maintained by the operating system for all the processes.
- There is just one page table in the entire system, implying that additional information needs to be stored in the page table to identify page table entries corresponding to each process.
- It stores one entry per physical frame, and is a linear array where the contents at each location is <pid (process id), virtual page number> and the index of each location is the physical frame address.

- The inverted page table is then searched for a match. If a match is found—say, at entry i—then the physical address is generated as shown in the figure.
- If no match is found, then an illegal address access has been attempted.



Ruff page

Virtual Memory:

- **Virtual paging.**
  - Page fault

  - Fetch Policy

    Pre paging

    Demand Paging

  - Thrashing

Virtual Memory:

- One of the major resources in the computer is memory, to utilize the memory in efficient way and to increase the CPU utilization in multiuser OS, we need to share the Memory for more number of users.
- In the process of evaluation memory allocation, we have learned how to divide our process and memory into multiple partitions using paging and segmentation.
- As ours is a control flow computer, the CPU Executes instructions in continues fashion based on the program counter one after another. With this it is conformed that we need not to fetch the total pages of a process in to the memory.
- With this approach, each process is allowed to bring only a fixed number of pages in to the memory. Hence we can accommodate more number of processes in the main memory.
- While executing a process, If at all, if the required page is not available in the main memory, it may leads to **page fault**, then CPU demands for that page and it is known as **demand paging**.

- Using **swapping** we need to bring the required page in to the memory by sending one of the available pages out, if again page fault occurs, immediately after the previous swapping and it continuous further it may lead to **Thrashing**.
- To avoid thrashing we need to use the page replacement algorithms and should find the best algorithm with minimum page faults.

- An end user or computer operator may not to know what really happening inside the computer, because he could able to execute all processes concurrently, he may think that, he is having sufficient memory inside the computer. But in reality which is not there. This Illusion in related to memory is called Virtual memory, and if we apply paging it is called Virtual paging.

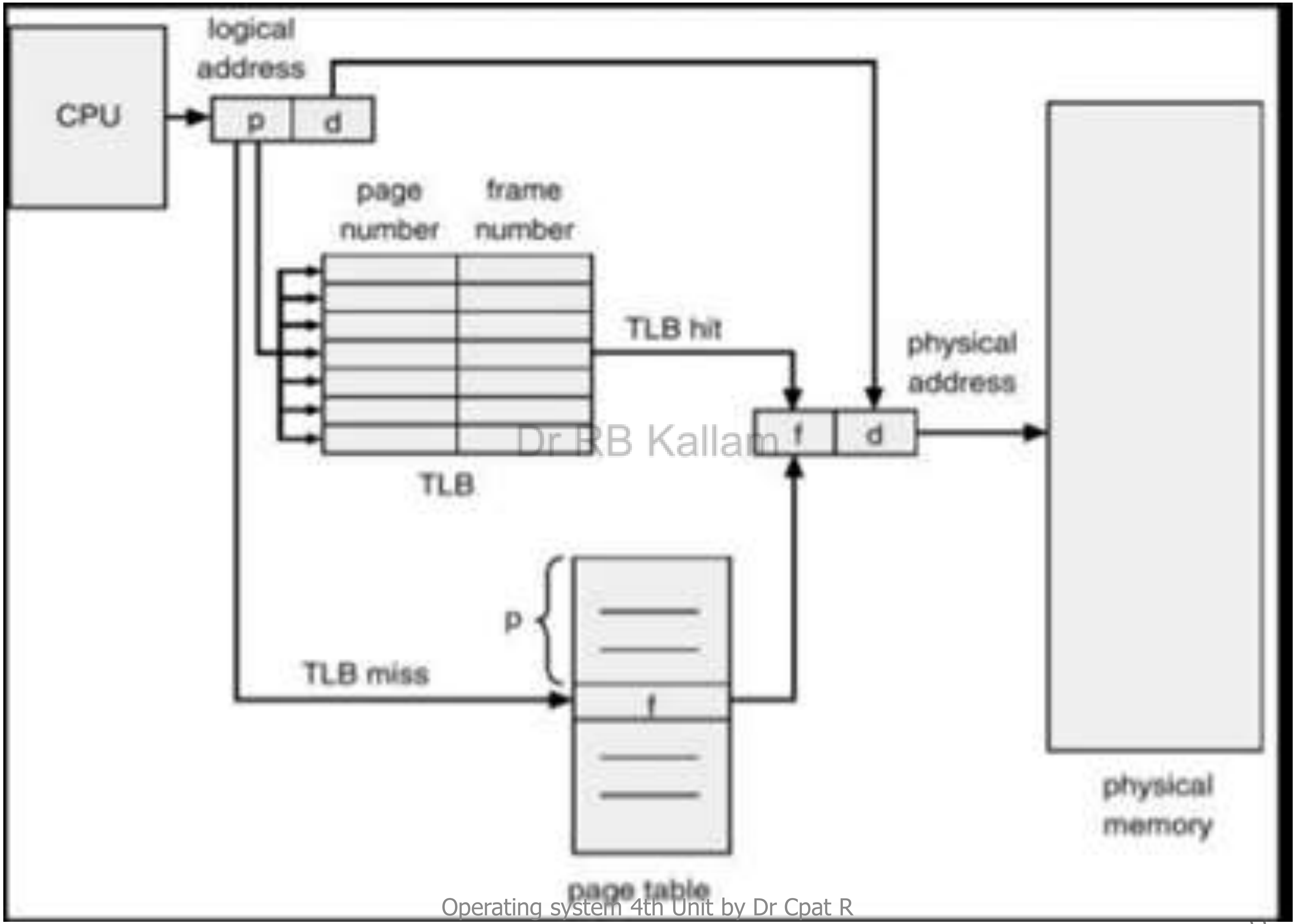
Ex:

- If we have two processes P1&P2; process P1 requires 100Kb and P2 also requires 100Kb and if the total available space is only 100Kb then how they will execute the programs in parallel?.
- To solve this problem by using paging technique we can divide P1 into 2 pages and P2 into 2 pages each of 50Kb and also memory into 2 frames of each 50Kb, by allocating 1 frame to each process and using swapping technique we can execute both the process in parallel and can get the results.
- But the end users will think that, there is sufficient buffer space that is 100Kb for each process because they could able to execute the program. To fulfull above requirement we should have 200Kb but in reality we have only 100Kb, this approach is called Virtual memory, as we have used paging it in turn called Virtual paging.

Translation look aside buffer (TLB)

- In principle every virtual memory reference can cause two physical memory accesses: one to fetch the appropriate page table entry and one to fetch the desired data.
- Thus a straight forward virtual memory scheme would have a effect of doubling the memory access time.
- To overcome this problem most virtual memory schemes make use of a special high speed cache for page table entries, usually called a TLB.
- This cache function in the same way as a memory cache and contains those page tables entries that have been most recently used.

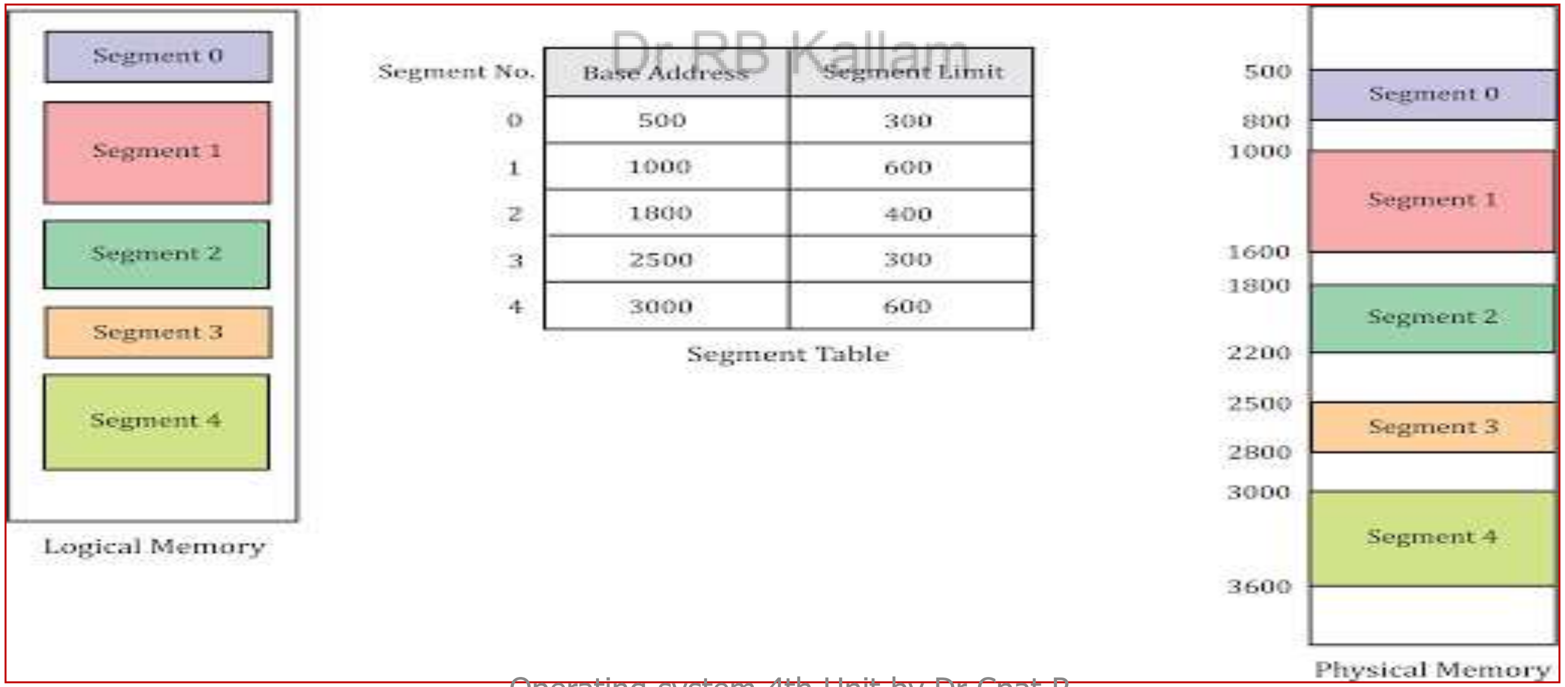
- Given a virtual address, the processer will first examine the TLB. If the desired page table entry is present (TLB hit), then the frame number is retrieved and the real address is formed .
- If the desired page table entry is not found (TLB miss), then the processor uses the page number to index the process page table in the main meory and examine the corresponding page table entry.
- If the "present bit" is set, then the page is in the main memory, and the processer can retrieve the frame number from the page table entry to form the real address. The processor also updates the TLB to include this new page table entry.
- Finally if the " Present bit" is not set, then the desired page not in main memory and a memory access fault, called a page fault, is issued.


CPU
logical address
p
d
page number
frame number
TLB hit
physical address
f
d
TLB
p
TLB miss
f
page table
physical memory

Segmentation:

- It is similar to dynamic partitioning
- A user program can be subdivided using segmentation, in which the program and its associated data are divided into a number of segments .
- It is not required that all segments of all programs be of the same length, although there is a maximum segment length.
- Due to the use of unequal-size segments, segmentation is similar to dynamic partitioning.
- The difference, compared to dynamic partitioning, is that with segmentation a program may occupy more than one partition, and these partitions need not be contiguous.

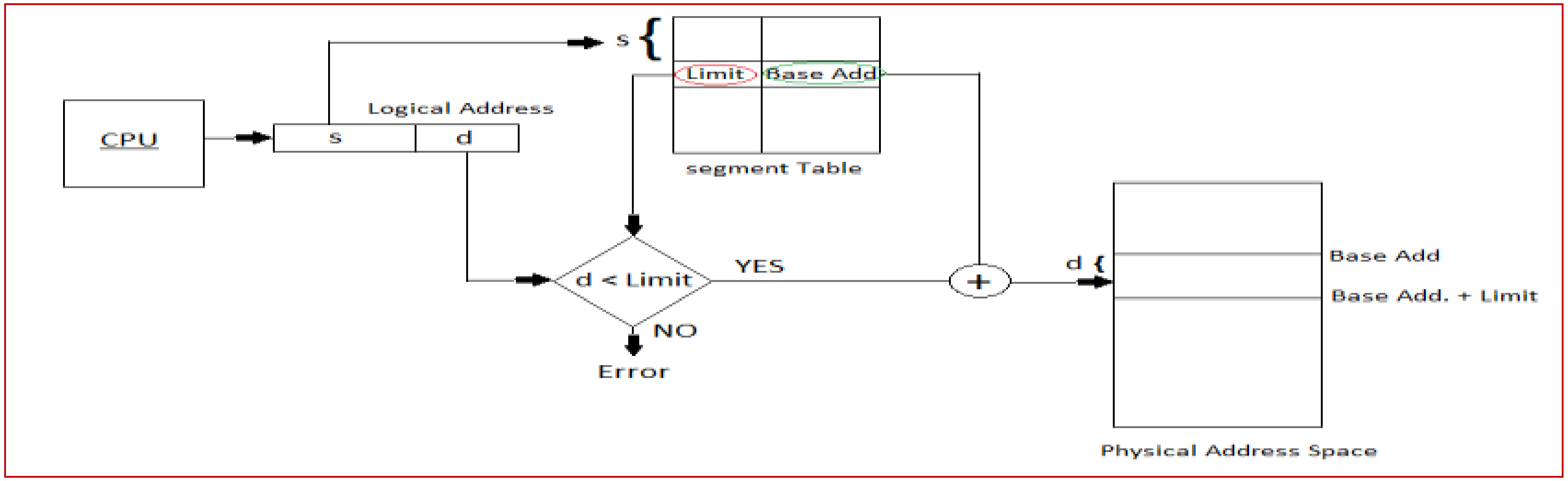
- With segmentation, sharing and different degree of protection is possible.
- A Segment table is maintained for each process and it contains three entries; s#, length of the seg., and Base address.
- As with paging, a logical address using segmentation consists of two parts, a segment number and an offset.



Segment No.	Base Address	Segment Limit
0	500	300
1	1000	600
2	1800	400
3	2500	300
4	3000	600

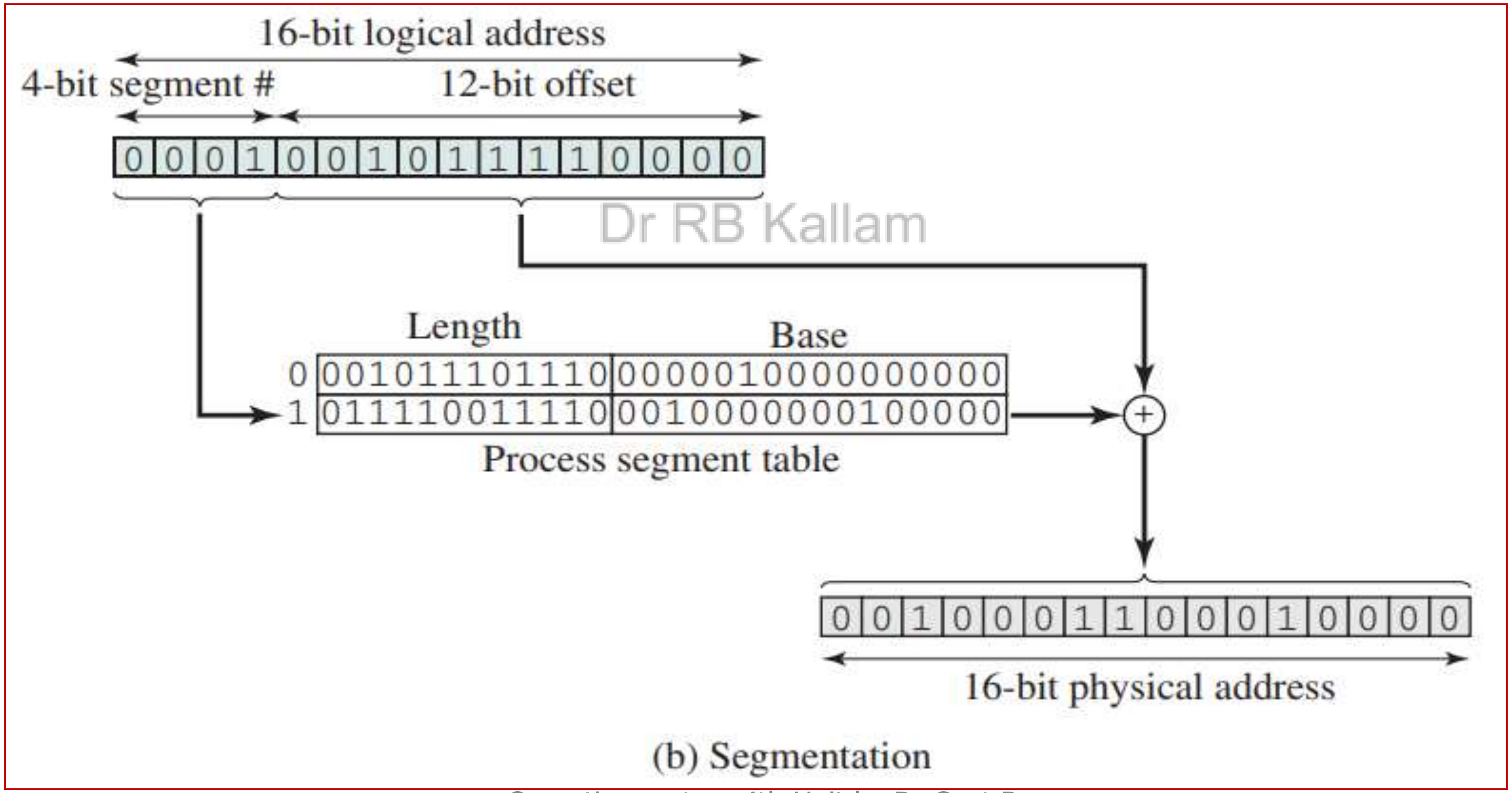
Segment Table

- Consider an address of n+m bits, where the leftmost n bits are the segment number and the rightmost m bits are the offset.
- For example if m = 12, the maximum segment size is 2^{12} = 4096.
- The following steps are needed for address translation:
  - Extract the segment number as the leftmost n bits of the logical address.
  - Use the segment number as an index into the process segment table to find the starting physical address/ base address of the segment.
  - Compare the offset, expressed in the rightmost m bits, to the length of the segment. If the offset is greater than or equal to the length, the address is invalid.
  - The desired physical address is the sum of the starting physical address of the segment plus the offset.

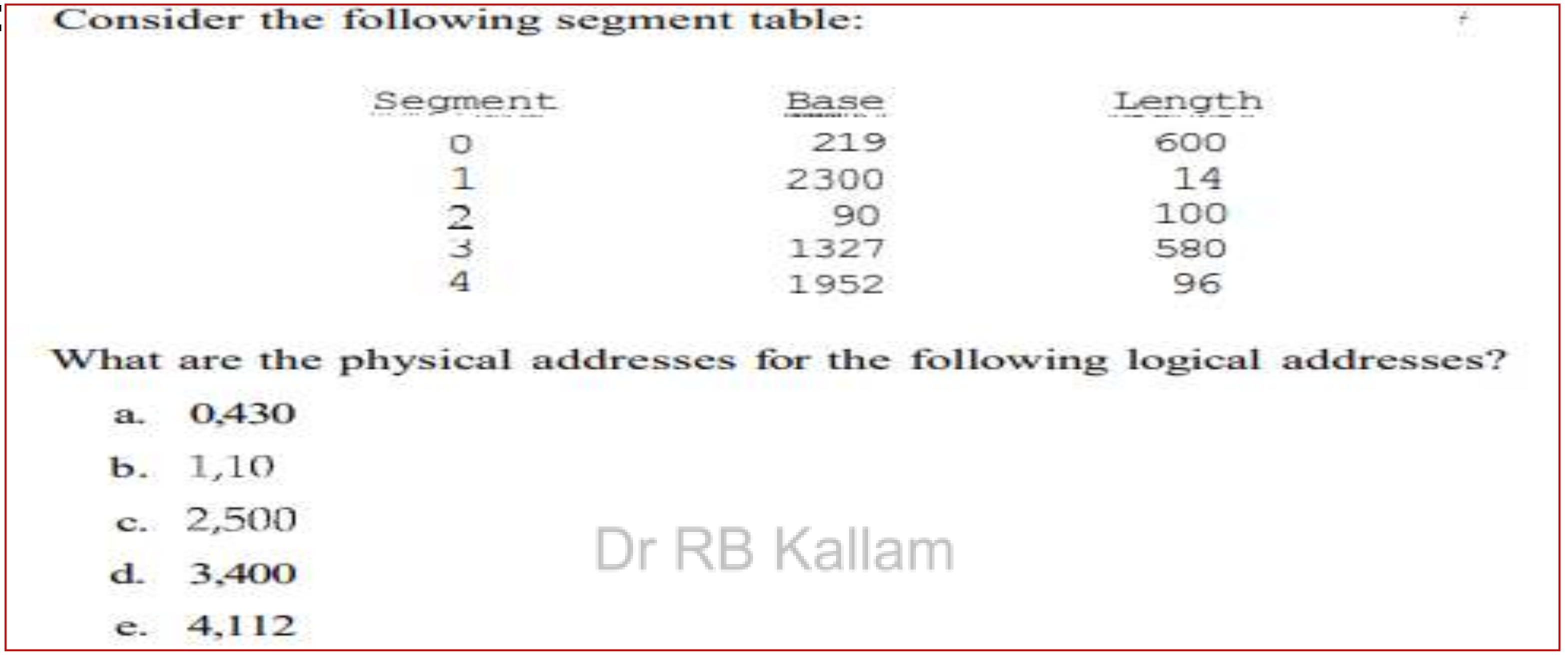


Translation in the Segmentation

- Consider an example, we have the logical address 0001001011110000, which is segment number 1, offset 752. Suppose that this segment is residing in main memory starting at physical address 0010000000100000. Then the physical address is 0010000000100000 + 001011110000 = 0010001100010000.



Ex 1:



Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

a. 0,430

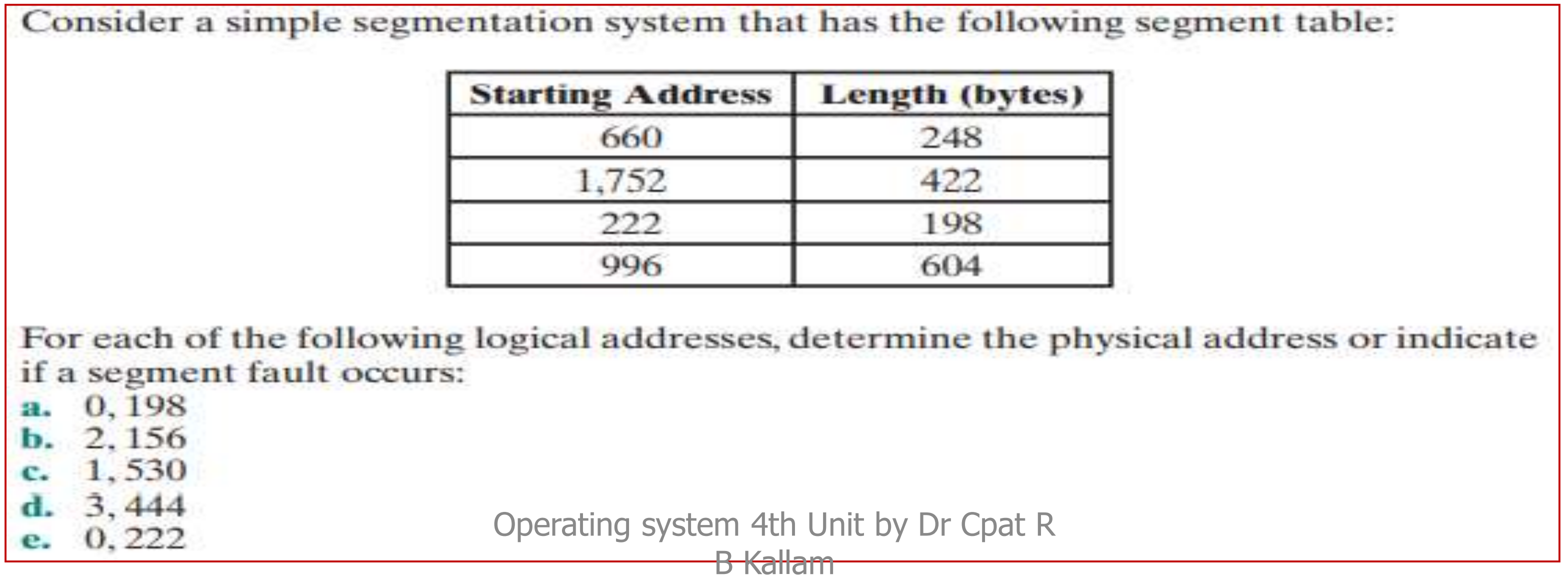
b. 1,10

c. 2,500

d. 3,400

e. 4,112

Ex 2:



Consider a simple segmentation system that has the following segment table:

Starting Address	Length (bytes)
660	248
1,752	422
222	198
996	604

For each of the following logical addresses, determine the physical address or indicate if a segment fault occurs:

a. 0, 198
b. 2, 156
c. 1, 530
d. 3, 444
e. 0, 222

- **Virtual Segmentation.**
  - Segmentation fault
  - Fetch Policy
    - Pre Segmentation
    - Demand Segmentation
  - Placement Policy
    - First Fit
    - Next Fit
    - Best Fit

Page replacement algorithms:

FIFO:

- This policy treats the page frames allocated to a process as a circular buffer, and pages are removed in round-robin style.
- All that is required is a pointer that circles through the page frames of the process.
- With this approach when a page must be replaced, the oldest page is chosen for replacement.

Optimal:

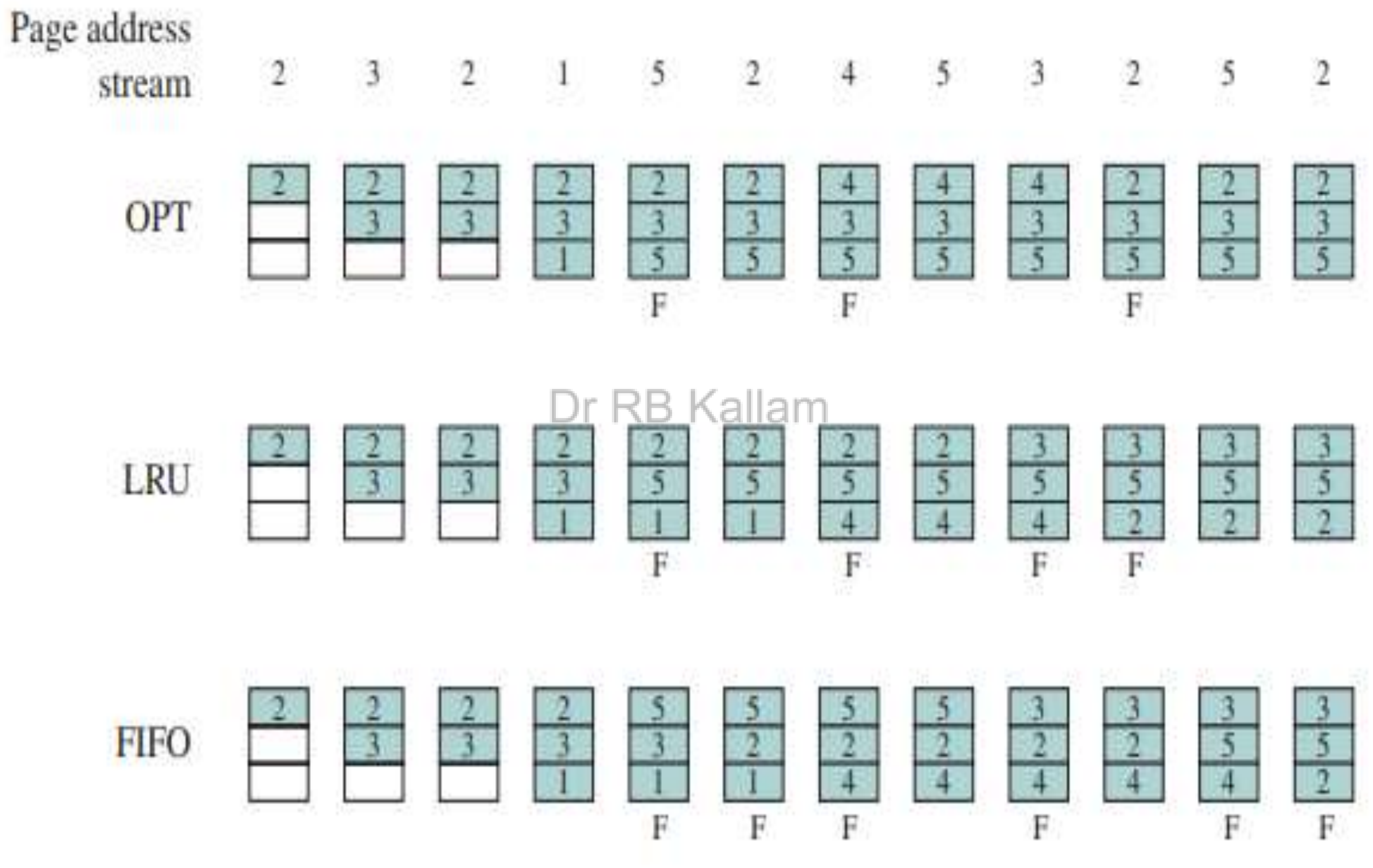
- This policy selects for replacement that page for which the time to the next reference is longest.
- Use of this page-replacement algorithm guarantees the lowest possible page fault rate for a fixed number of frames
- This policy is impossible to implement, because it would require the OS to have perfect knowledge of future events.

LRU (Least Recently Used):

- This policy replaces the page in memory that has not been referenced for the longest time.
- The problem with this approach is the difficulty in implementation.
- One approach would be to tag each page with the time of its last reference; this would have to be done at each memory reference, both instruction and data.
- It creates lot of overhead and expensive.

Ex: Let us assume that only three frames (fixed resident set size) are allocated for a process, the execution of the process requires reference to five distinct pages and the page address stream formed by executing the program is

2 3 2 1 5 2 4 5 3 2 5 2. Apply FIFO, Optimal and LRU page replacement algorithms and the number of page faults.



F = page fault occurring after the frame allocation is initially filled

Ruff Page

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

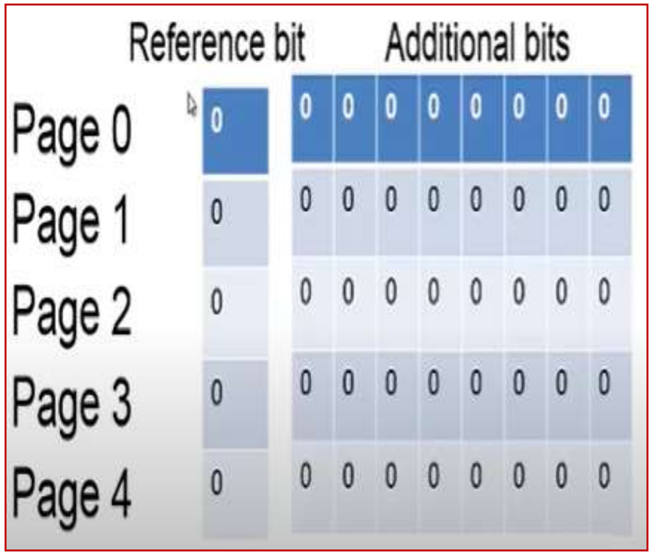
2	2	2	2								
	3	3	3								
			1								

- LRU Approximation Algorithms:
  - Additional Reference Bits Algorithm
  - Second Chance Algorithm
  - Enhanced Second Chance Algorithm
  - Counting Algorithms:
    - LFU Algorithm
    - MFU Algorithm
  - Page Buffering Algorithm

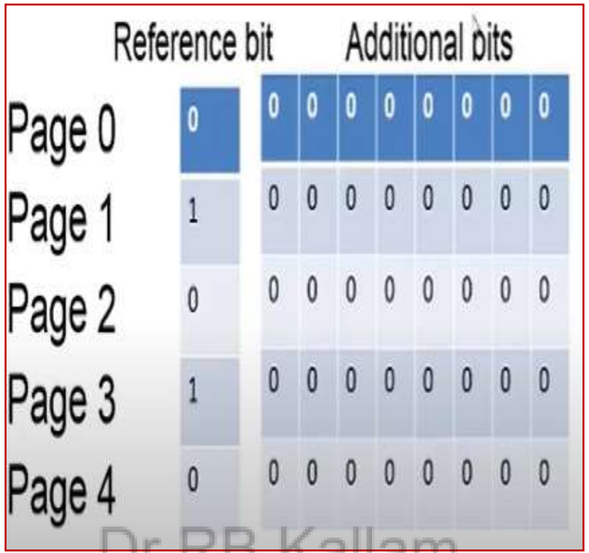
Additional Reference Bits Algorithm:

- We can gain additional ordering information by recording the reference bits at regular intervals.
- We can keep an 8-bit byte for each page in a table in memory
- At regular intervals (say for every 100 milliseconds), a timer interrupt transfers control to the OS.
- The OS shifts the reference bit for each page into the higher order bit of its 8 bit byte, shifting the other bits right 1 bit, discarding the lower order bit.
- These 8 bit shift register contain the history of page use for the last eight time periods.
- If the shift register contains 00000000, then the page has not been used for eight time periods.
- The page that is used at least once each period would have a shift register value of 11111111.
- *A page with a history register value of 11000100 has been used more recently than has one with 01110111. If we interpret these 8 bit bytes as unsigned integers, the page with the lowest number is the LRU page, and it can be replaced.*

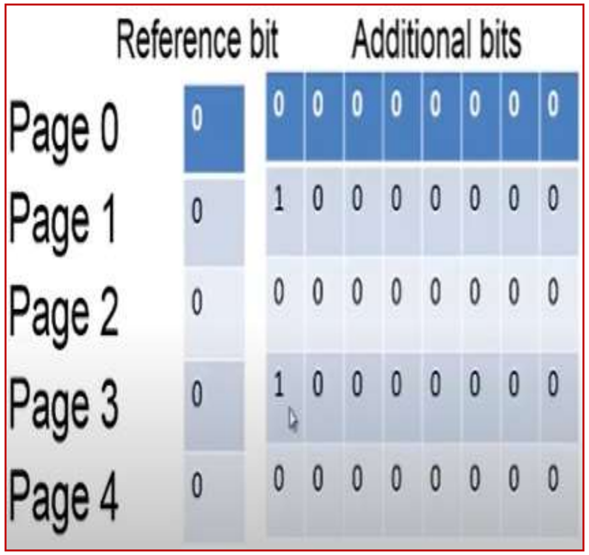
Example:



	Reference bit	Additional bits							
Page 0	0	0	0	0	0	0	0	0	0
Page 1	0	0	0	0	0	0	0	0	0
Page 2	0	0	0	0	0	0	0	0	0
Page 3	0	0	0	0	0	0	0	0	0
Page 4	0	0	0	0	0	0	0	0	0

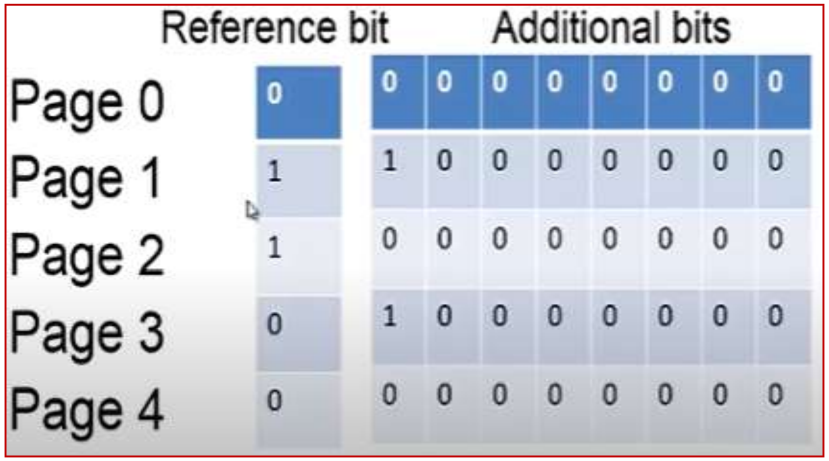


	Reference bit	Additional bits							
Page 0	0	0	0	0	0	0	0	0	0
Page 1	1	0	0	0	0	0	0	0	0
Page 2	0	0	0	0	0	0	0	0	0
Page 3	1	0	0	0	0	0	0	0	0
Page 4	0	0	0	0	0	0	0	0	0

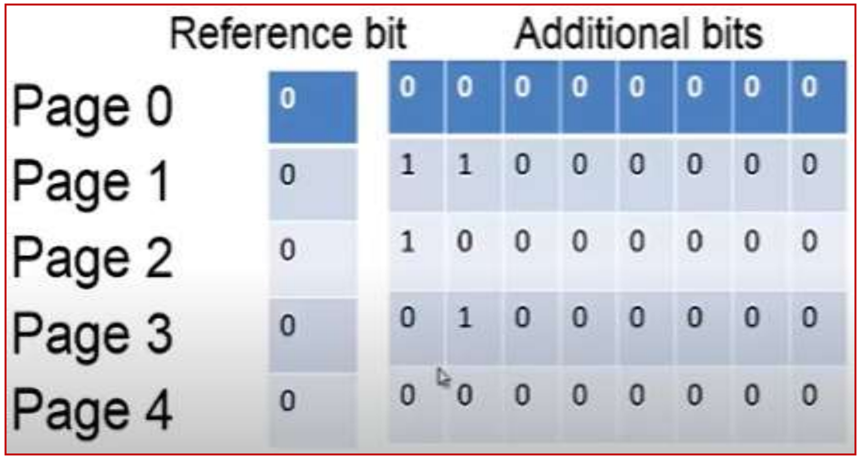


	Reference bit	Additional bits							
Page 0	0	0	0	0	0	0	0	0	0
Page 1	0	1	0	0	0	0	0	0	0
Page 2	0	0	0	0	0	0	0	0	0
Page 3	0	1	0	0	0	0	0	0	0
Page 4	0	0	0	0	0	0	0	0	0

Snapshot -1



	Reference bit	Additional bits							
Page 0	0	0	0	0	0	0	0	0	0
Page 1	1	1	0	0	0	0	0	0	0
Page 2	1	0	0	0	0	0	0	0	0
Page 3	0	1	0	0	0	0	0	0	0
Page 4	0	0	0	0	0	0	0	0	0

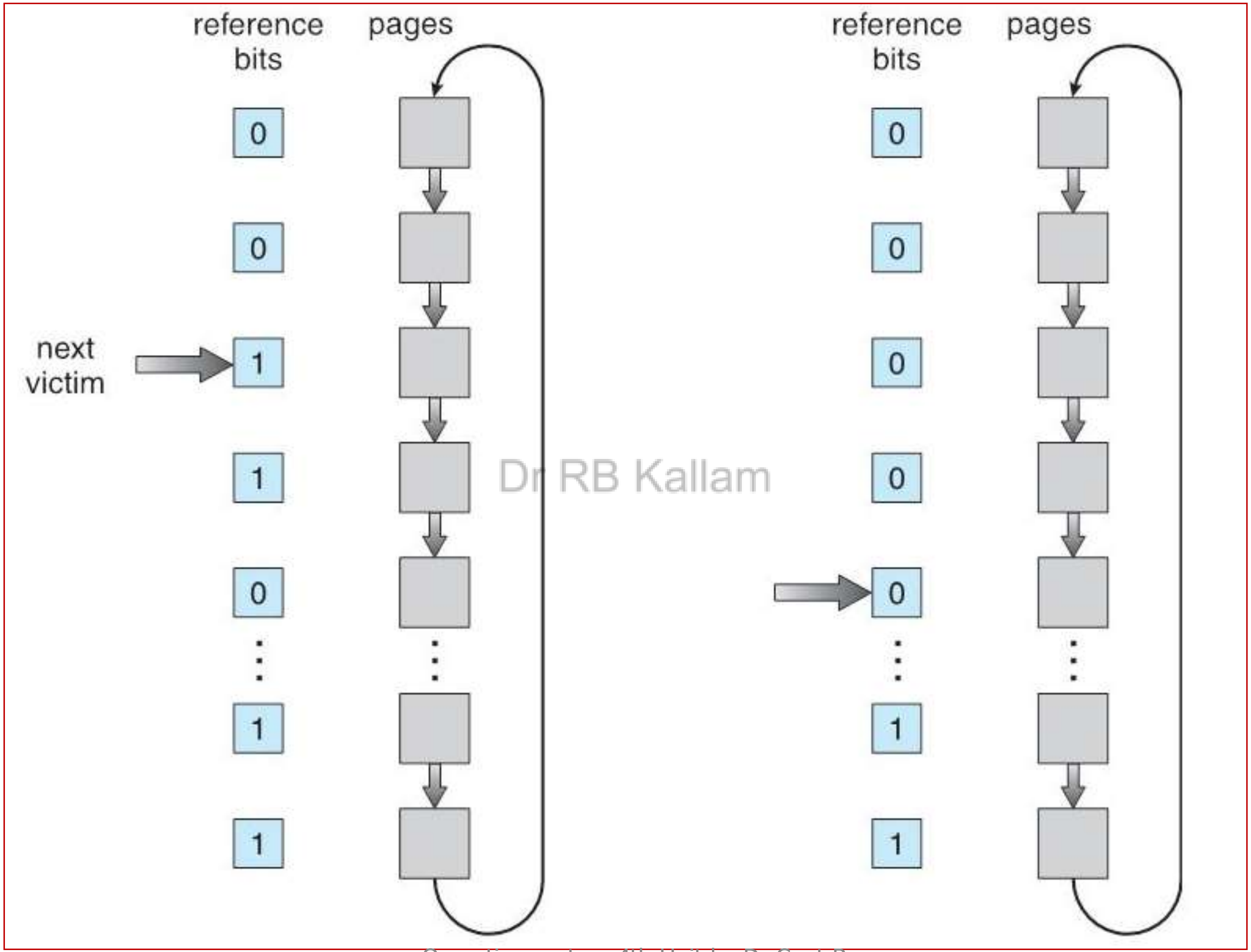


	Reference bit	Additional bits							
Page 0	0	0	0	0	0	0	0	0	0
Page 1	0	1	1	0	0	0	0	0	0
Page 2	0	1	0	0	0	0	0	0	0
Page 3	0	0	1	0	0	0	0	0	0
Page 4	0	0	0	0	0	0	0	0	0

Snapshot -2

Second Chance Algorithm (clock)

- It requires the association of an additional bit with each frame, referred to as a use bit.
- When a page is first loaded in to a frame in memory, the use bit for that frame is set to 1. When the page is subsequently referenced, its use bit is set to 1.
- For the page replacement algorithm, the set of frames that are candidates for replacement is considered to be a circular buffer with which a pointer is associated.
- When a page is replaced the pointer is set to indicate the next frame in the buffer.
- When it is time to replace a page, the OS scans the buffer to find a frame with a use bit set to 0.
- Each time it encounter a frame with a use bit of 1, it resets that bit to 0.
- If any of the frames in the buffer have a use bit of 0 at the beginning of this process, the first such frame encountered is chosen for replacement.
- If all the frames have a use bit of 1,then the pointer will make one complete cycle through the buffer, setting all the use bits to 0, and stop at its original position, replacing the page in that frame.


reference bits
pages
next victim
0
0
1
1
0
1
1
reference bits
pages
0
0
0
0
0
1
1

Enhanced Second Chance Algorithm

- The clock policy can be enhanced by considering both the reference bit and modify bit as an ordered pair.
- With these 2 bits, we have the following 4 possible classes:

1. (0,0) neither recently used nor modified – best page to replace.
2. (0,1) not recently used but modified – not quite as good, because the page will need to be written out before replacement.
3. (1,0) recently used but clean – probably will be used again soon.
4. (1,1) recently used and modified – probably will be used again, and write out will be needed before replacing it.

When a page has to be replaced, the first page encountered in the lowest non empty class is chosen.

Counting Algorithms:

- LFU Algorithm: The Least Frequently Used page replacement algorithm require that the page with the smallest count be replaced.
- MFU Algorithm: The Most Frequently Used page replacement algorithm require that the page with the greatest count be replaced.

Page Buffering Algorithm

- Maintaining pool of free frame list:
  - Systems commonly keep a pool of free frames.
  - When a page fault occurs, a victim frame is chosen as before.
  - However, the desired page is read into a free frame from the pool before the victim is written out.
  - This procedure allows the process to restart as soon as possible, without waiting for the victim page to be written out.
  - When the victim is later written put, its frame is added to the free-frame pool.
- Maintaining pool of modified frame list:
  - An expansion of free frame list is to maintain a list of modified pages.
  - Whenever the paging device is idle, a modified page is selected and is written to the disk.
  - Its modify bit is then reset. This scheme increases the probability that a page will be clean when it is selected for replacement and will not need to be written out.

Allocation of Frames:

- Minimum Number of frames:
  - We must also allocate at least a minimum number of frames.
  - The minimum number of frames is defined by the computer architecture.
- Allocation Algorithm
  - Equal allocation
  - Proportional allocation
- Global Versus Local Allocation

Allocation Algorithm:

- Equal allocation: It is easiest way to split *m* frames among *n* processes is to give everyone an equal share, *m / n* frames.

- Example:
  - Process-1 needs 10 frames
  - Process – 2 needs 127 frames
  - Available is only 62 frames
  - *With this approach each will get 31 frames*
  - *The process-1 does not need more than 10 frames, so the other 21 are strictly wasted.*

Proportional allocation: we allocate available memory to each process according to its size.

– Let the size of the virtual memory for process *Pi* be *Si*, and define

$$S = \sum Si$$

Then, if the total number of available frames is *m*, we allocate a_i frames to process *Pi*, where a_i is approximately

$$a_i = (Si / S)\ X\ m.$$

For previous example:

process-1 gets (10/137) X 62 = 4

process-2 gets (127/ 137) X 62 = 57

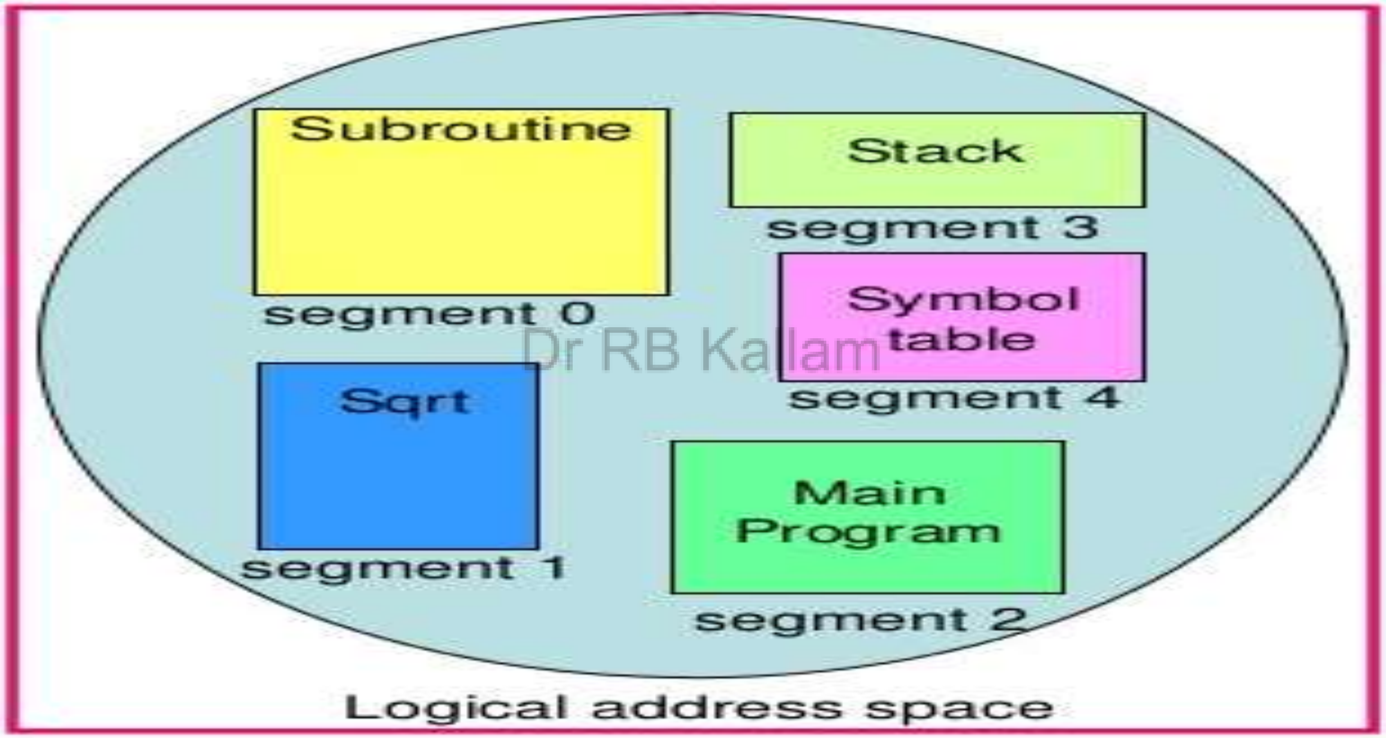
In this way both processes share the available frames according to their needs.

Global Versus Local Allocation

- Global replacement – process selects a replacement frame from the set of all frames; one process can take a frame from another.
- Local replacement – each process selects from only its own set of allocated frames.

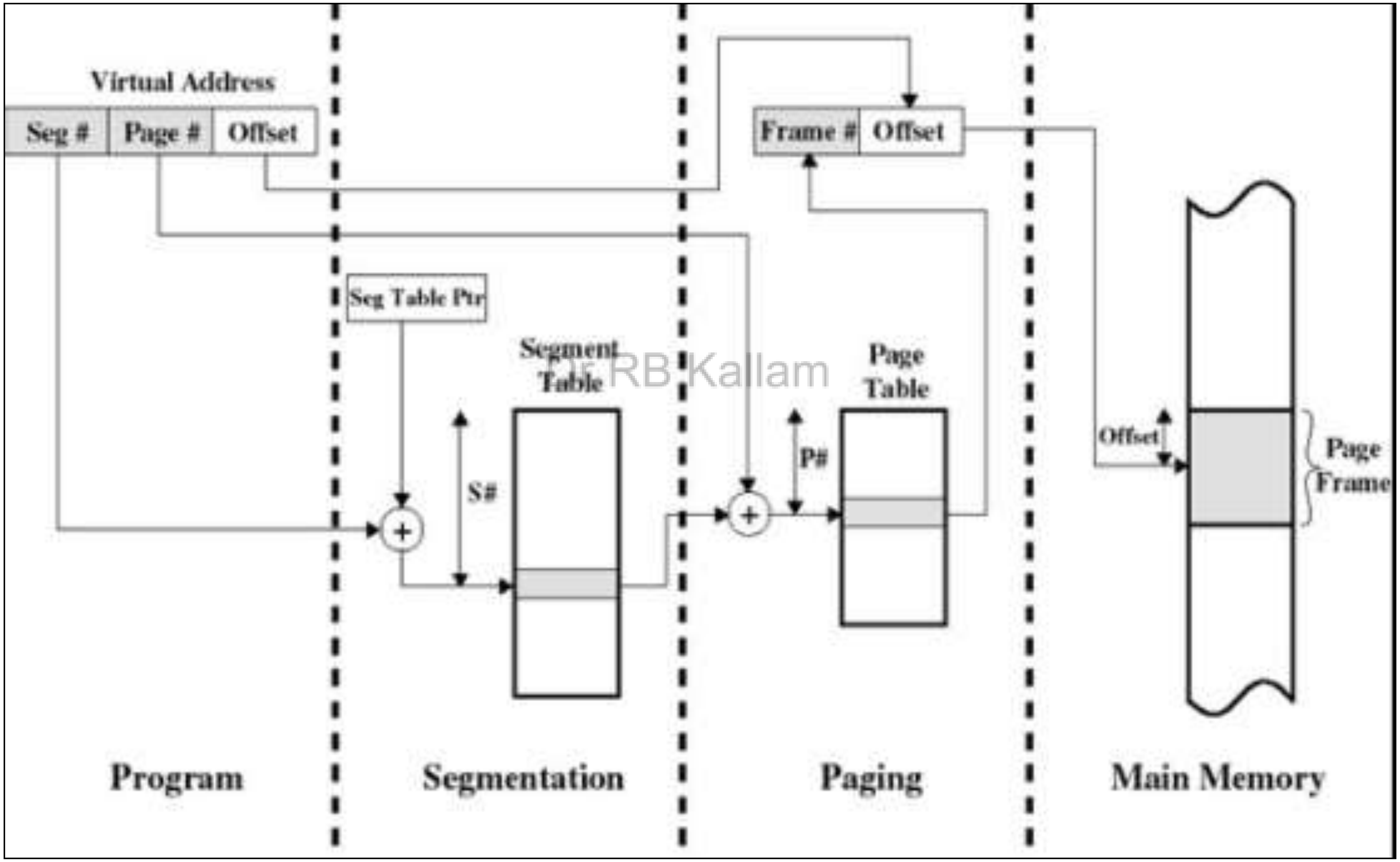
Combined segmentation and paging:

- To combine the advantages of both paging and segmentation some systems are equipped with segmentation with paging.
- In this the user address space is broken up into number of segments and each segment in turn broken up into number of fixed size pages, which are in equal in length to main memory frame.
- If the segment have length less than that of page , the segment occupies just one page.
- From the programmer's point of view the logical address is still consists of segment number and segment offset.
- From the system point of view the segment offset is viewed as page number and page offset for a page within the specified segment



segmentation

Combined segmentation and paging:



The End