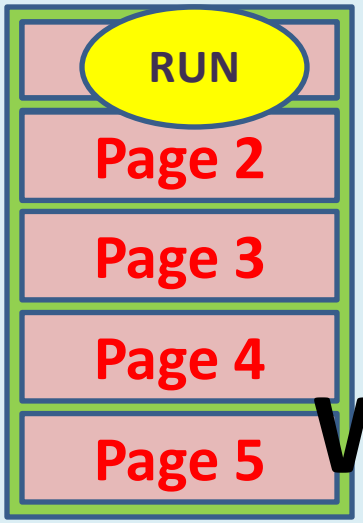
Memory Management

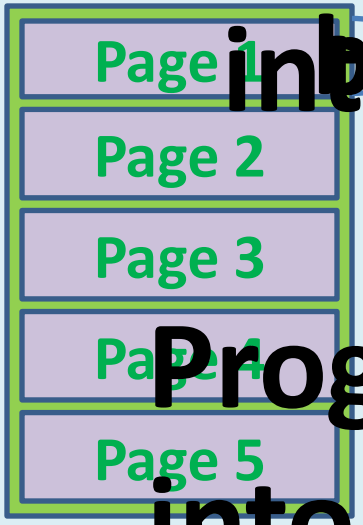
Paging	Segmentation
Fixed Sized Partitions	Variable Sized Partitions
Physical Partitioning	Logical Partitioning
Implicit	Explicit
Done by OS	Done by Programmer
Slower Than Segmentation	Faster than Paging

Paging



10 KB

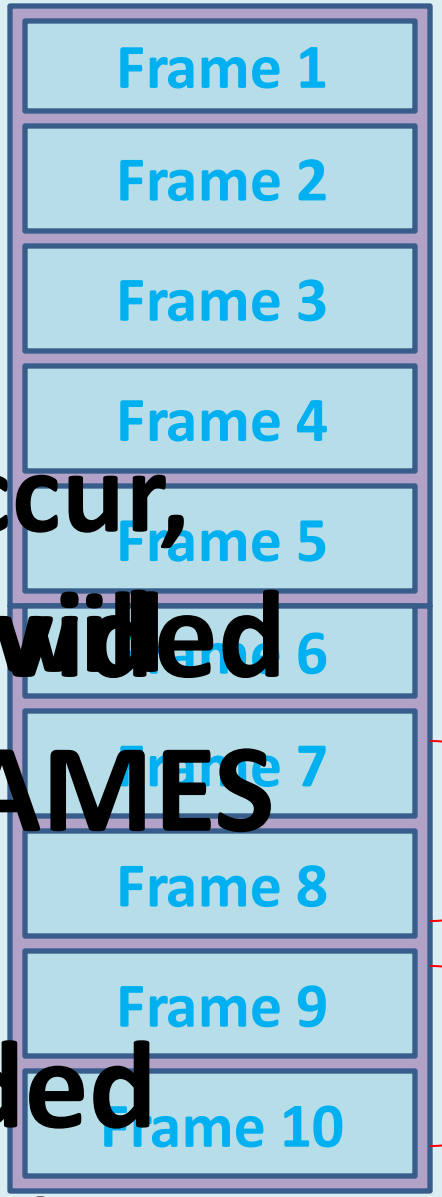
Program 1 in Secondary Memory



10 KB

Program 2 in Secondary Memory

Page Fault/ Miss



P1
P2
P3

Primary Memory (RAM)

When Page Miss occur, RAM will also be divided into same sized FRAMES

Program will be divided into fixed sized partitions

Page Relocation

Advantage of Paging

– Alleviate the Problem of Fragmentations

Translation of Page Numbers in Memory Frames

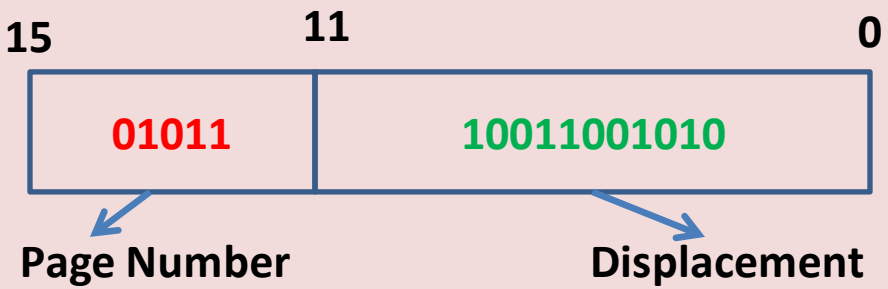
Page Relocation can be done by "Addressing of pages"

Disadvantage of Paging

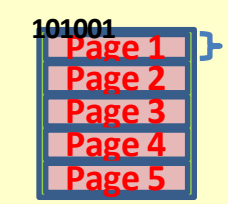
- Relocation Problem

Page Address

Address of Page = (P, D)



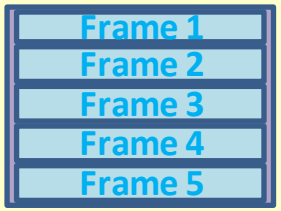
For 16 bit Computer



Page Address = Program Address + Page Displacement

Page	Displacement
1	10 KB
2	20 KB
3	30 KB
4	40 KB
5	50 KB

Page Address Table



Primary Memory (RAM)

Frame	Displacement
1	30 KB
2	20 KB

Frame Address Table

Page No	Frame No
1	
2	
3	
4	
5	

Page Table

No Problem on Execution of Page 1

No Problem on Execution of Page 2

Page Miss on execution of Page 3

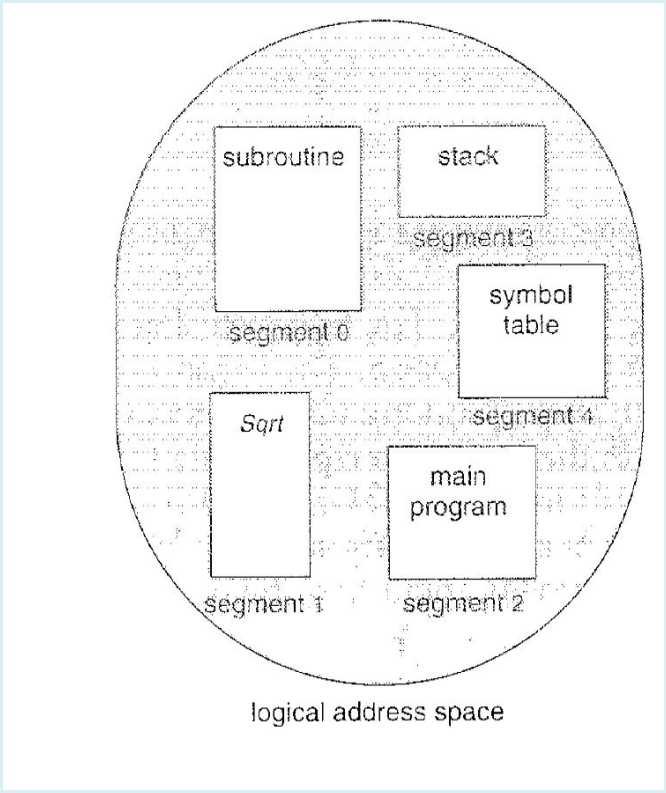
Page Swap will be done

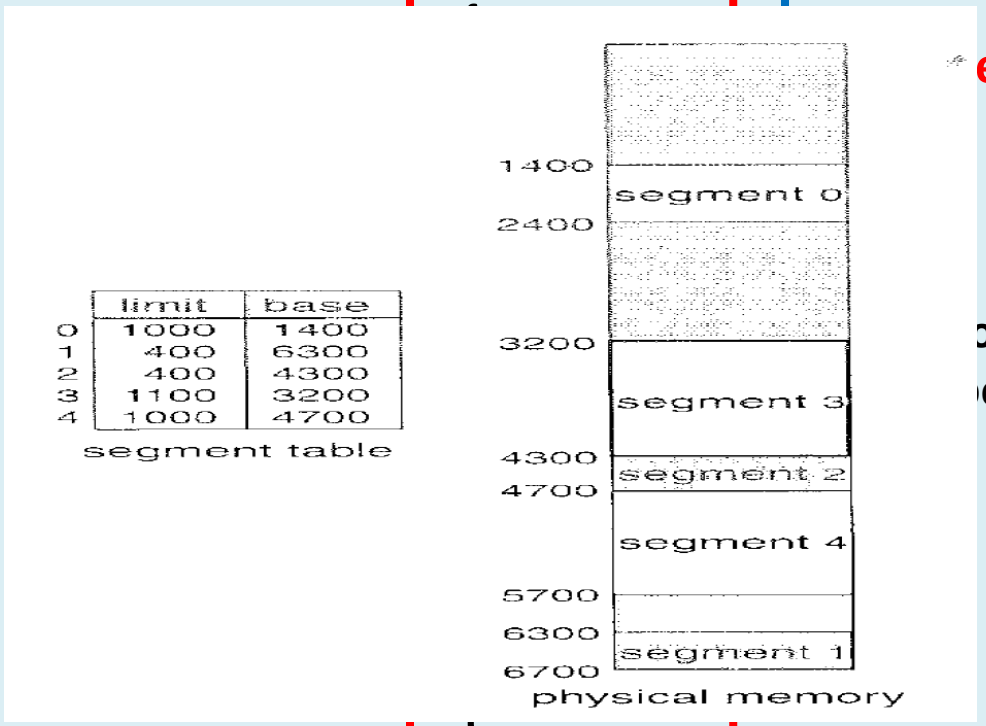
Segmentation

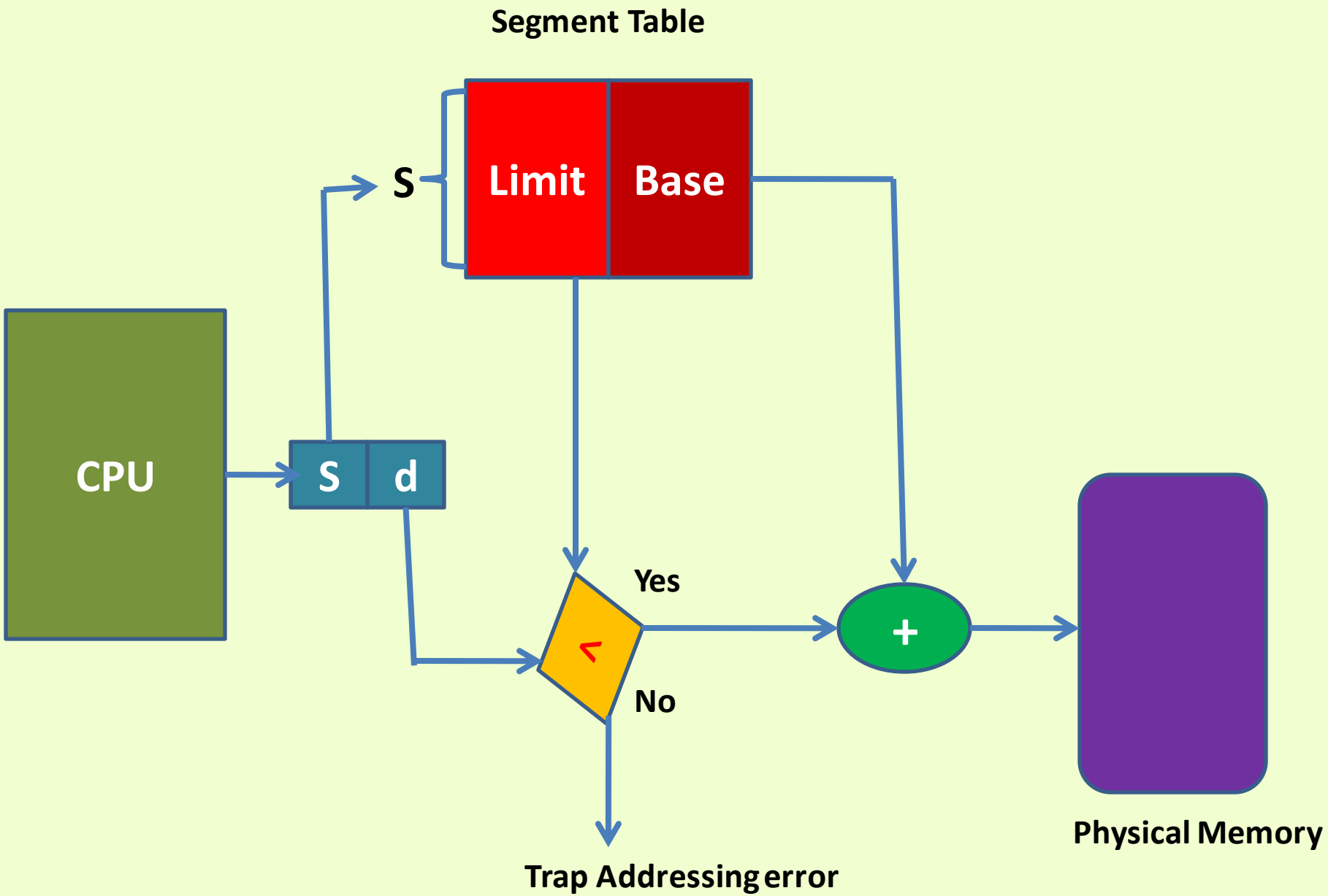
It is Logical Partitioning of the Program.

The Size of segments will be vary

Blocks in Program (like Branching & looping constructs as well as Functions) can not be partitioned






Segment Table
S
Limit
Base
CPU
S
d
Yes
<
No
+
Physical Memory
Trap Addressing error

Virtual Memory

Virtual Memory is a technique that allow the execution of processes that are not completely in memory

The program would no longer be constrained by the amount of physical memory.

User would be able to write programs for an extremely large virtual address space

Advantage

Programs can be larger than physical memory

Increase in CPU utilization and throughput

But no increase in response time and turn around time

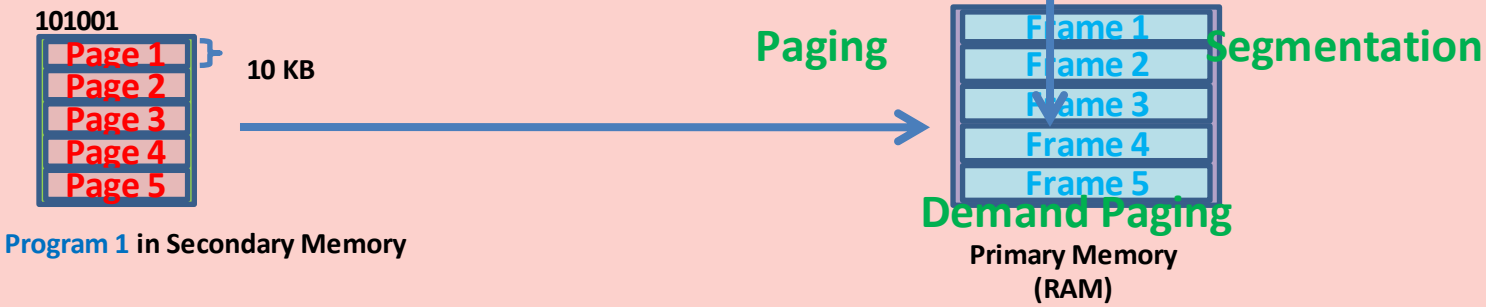
Disadvantage

Complex and costly

Virtual Memory

Place entire logical space in Physical memory

Place a part of logical space in Physical Memory



With Demand Paging Virtual Memory

Pages are only loaded when they are demanded during program execution

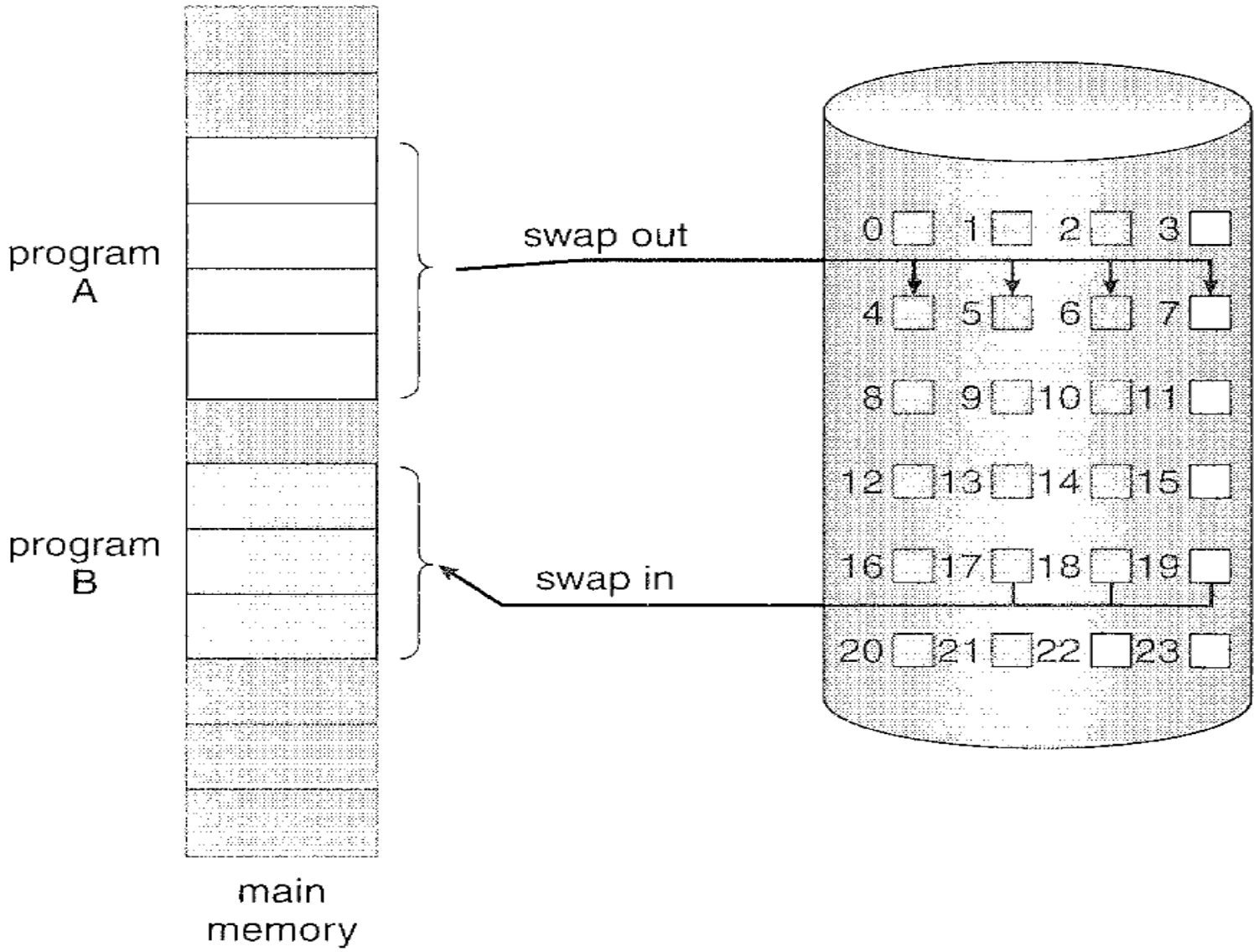


Figure 9.4 Transfer of a paged memory to contiguous disk space.

When the Process is to be swapped in,

The PAGER guesses which page will be used
before the process is swapped out again ,

Instead of swapping in a whole process

The Pager brings only those necessary pages into memory.

Thus avoids reading into pages that will not be used anyway

Deceasing the swap time and the amount of physical memory needed

Requirement of Demand Paging is Same as Paging and Segmentation i.e. –
- Page Table
- Secondary Memory

Page Replacement

WHAT IS?

If no frame is free ,

OS find one which is not currently being used and free it

HOW TO ?

We can free a page by writing its contents to Swap space and changing the page table

We can now use the freed frame to hold the page for which the process faulted

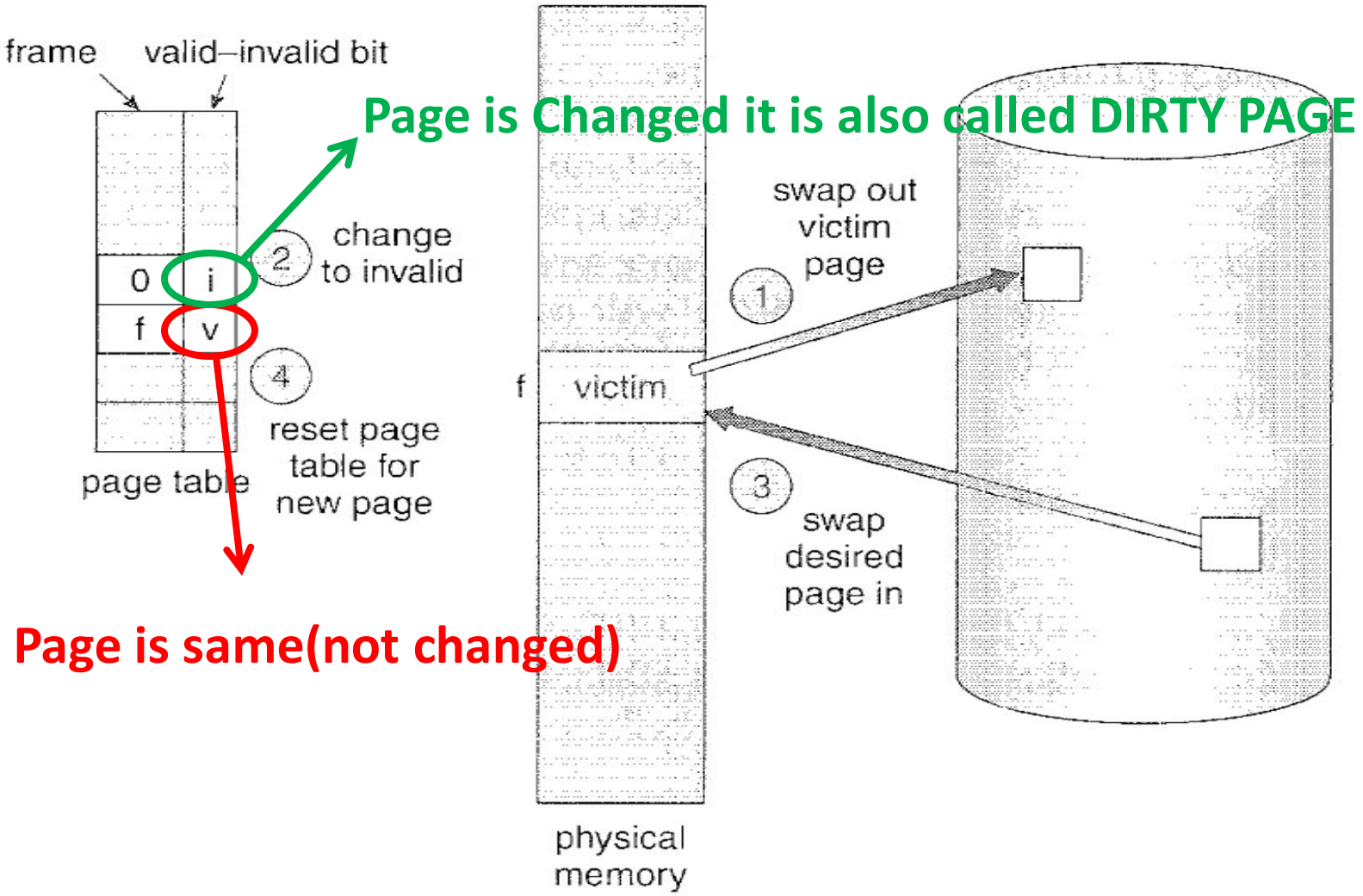


Figure 9.10 Page replacement.

Page Replacement Procees

1. Find the location of desired page on the 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
   - Write the victim frame to the disk; change the page and frame table accordingly

3. Read the desired page into the newly freed frame.; change page and frame table

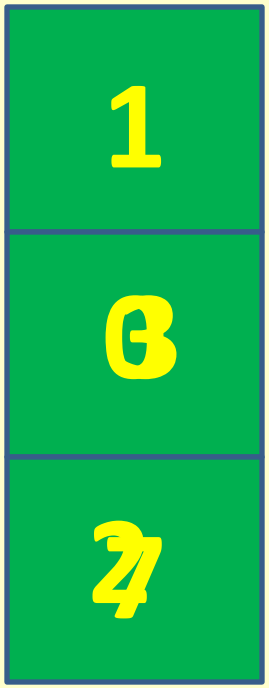
4. Restart the user process.

Page Replacement Algorithms

FIFO page Replacement Algorithm

When the Page is replaced – the OLDEST page is chosen

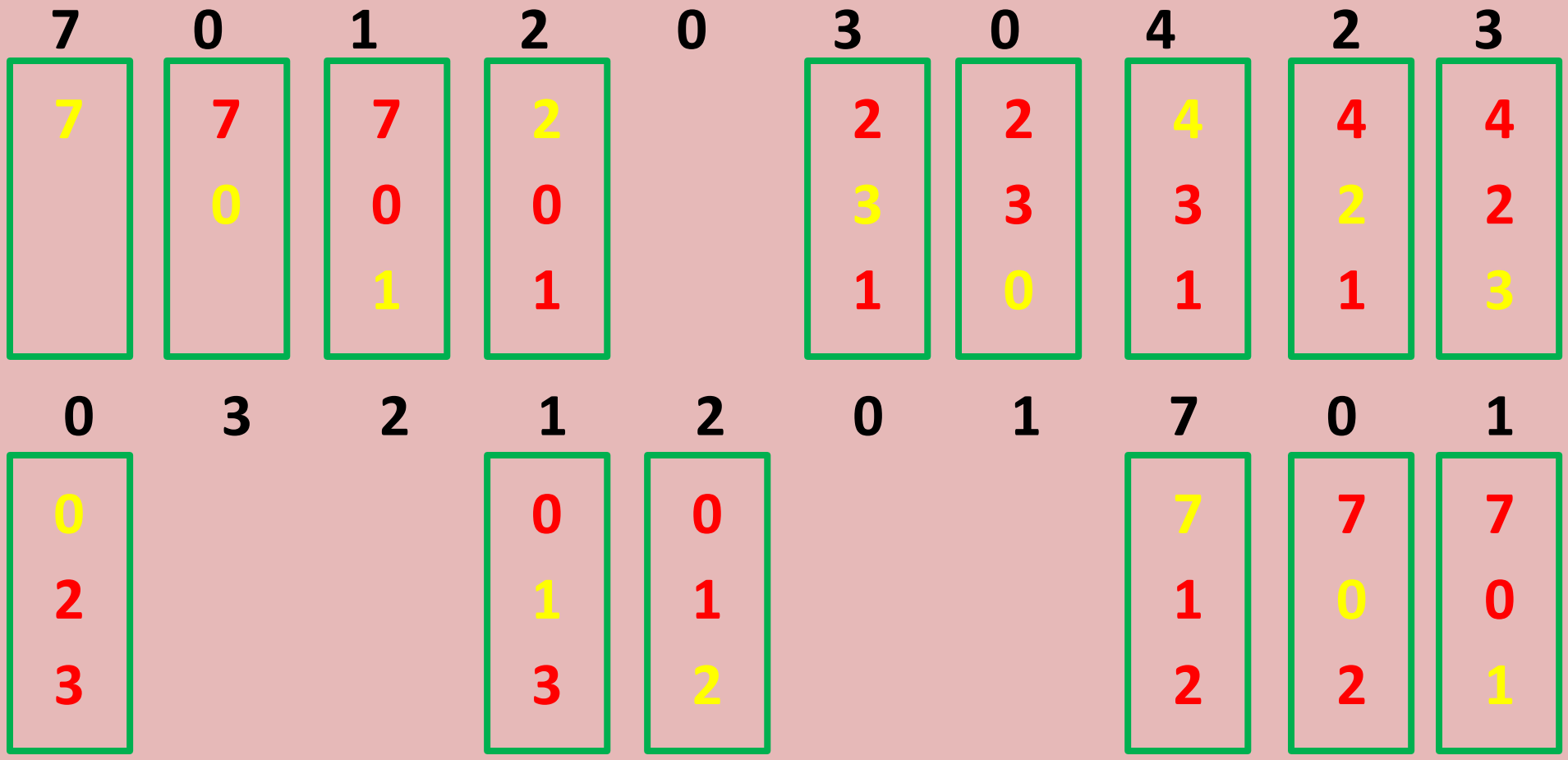
Reference String : 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



Page Fault

Page HIT

First IN page

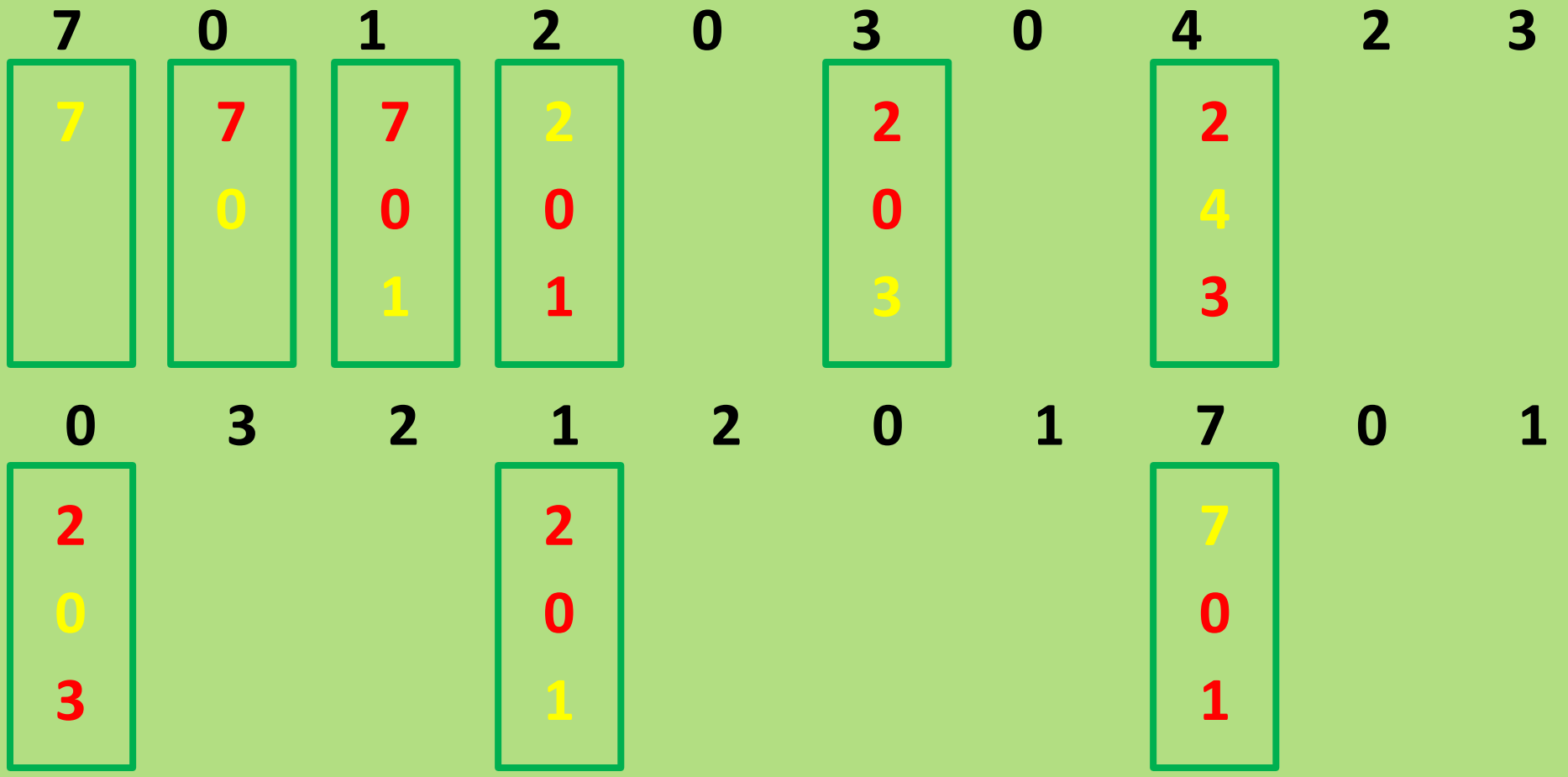

7 0 1 2 0 3 0 4 2 3
7
7 0
7 0 1
2 0 1
2 3 1
2 3 0
4 3 1
4 2 1
4 2 3
0 3 2 1 2 0 1 7 0 1
0 2 3
0 1 3
0 1 2
7 1 2
7 0 2
7 0 1

Page Replacement Algorithms

Optimal page Replacement Algorithm

Replace the Page that WILL be NOT USED for LONGER PERIOD of TIME

Reference String : 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



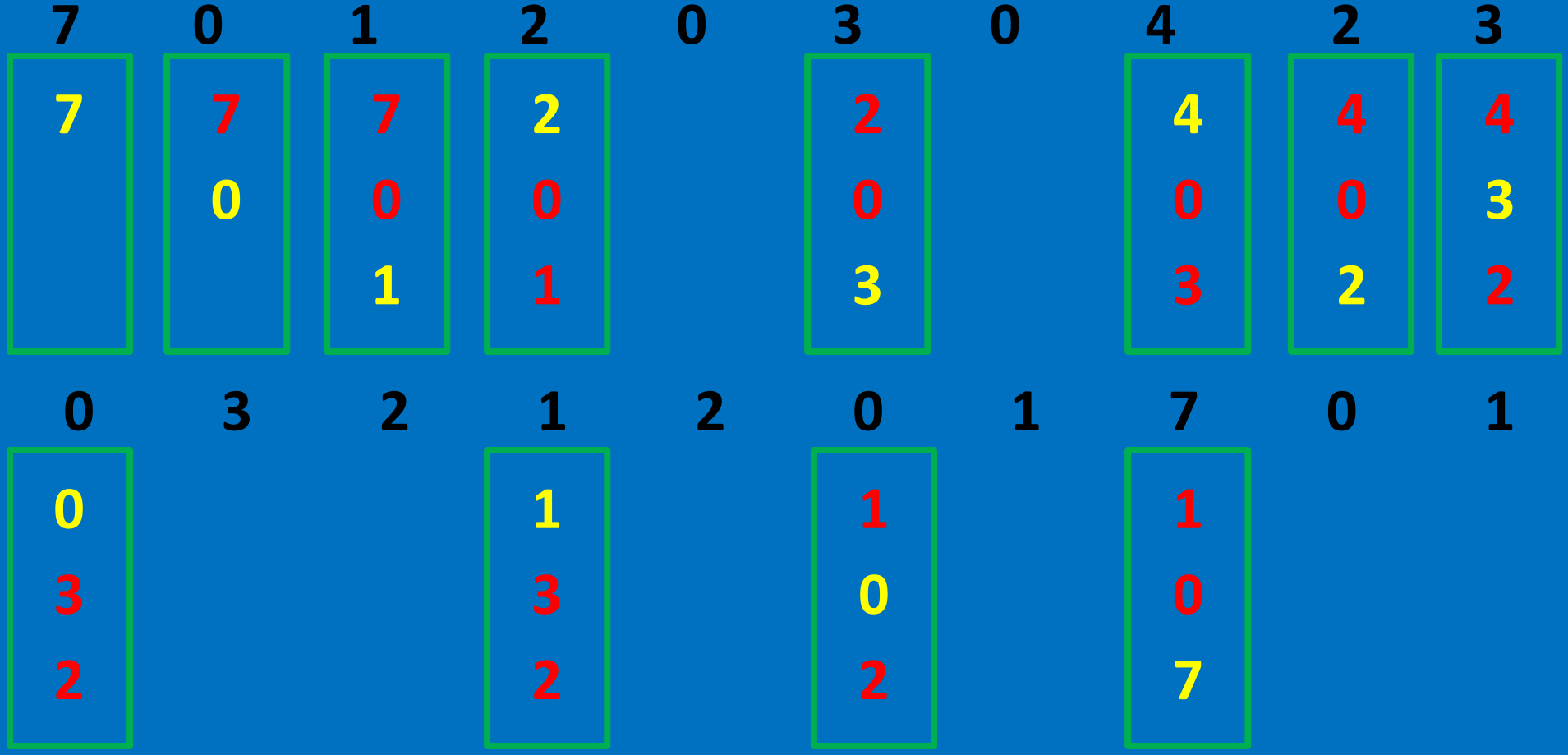
SEE THE RIGHT

Page Replacement Algorithms

LRU (Least Recently Used) page Replacement Algorithm

Replace the Page that is NOT USED for LONGER PERIOD of TIME

Reference String : 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1


7 0 1 2 0 3 0 4 2 3
7
7 0
7 0 1
2 0 1
2 0 3
4 0 3
4 0 2
4 3 2
0 3 2 1 2 0 1 7 0 1
0 3 2
1 3 2
1 0 2
1 0 7

SEE THE LEFT

Thrashing

High Page Fault activity is called THRASHING

Thrashing decreases the performance of the system

How big Problem is THRASHING ????????

See

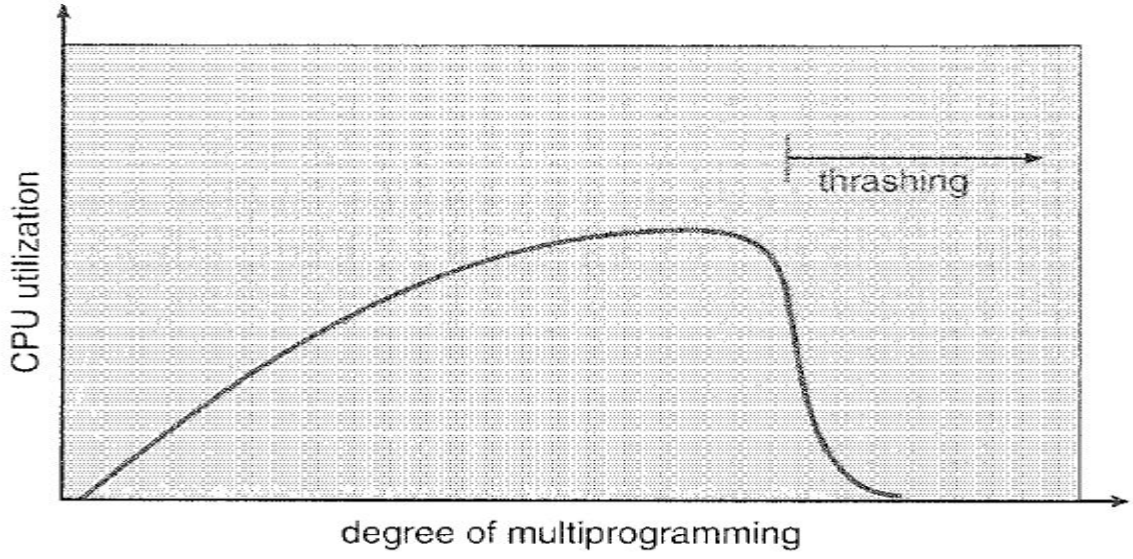


Figure 9.18 Thrashing.

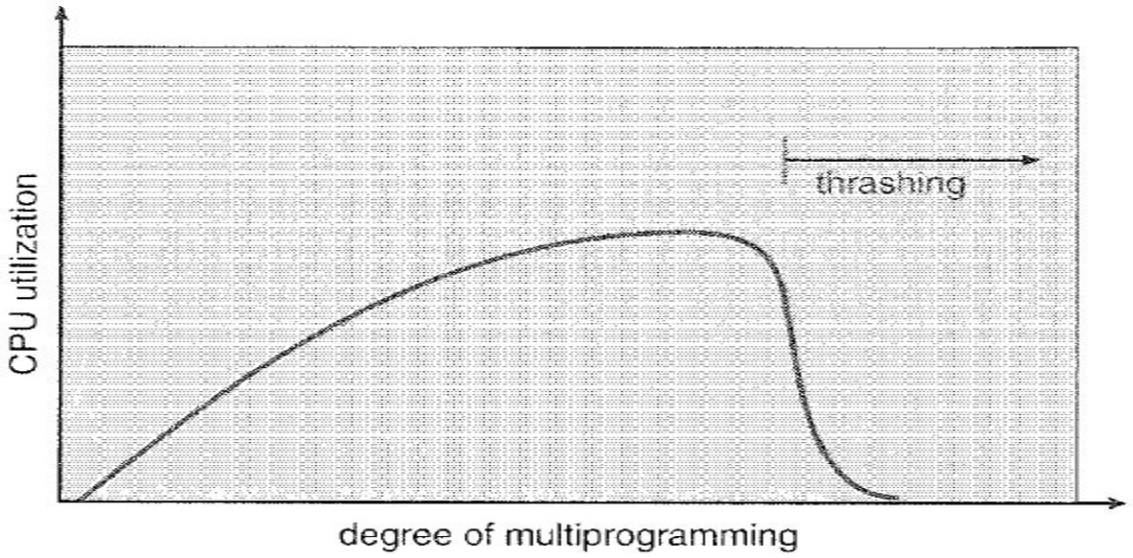


Figure 9.18 Thrashing.

This phenomenon is illustrated in Figure 9.18, in which CPU utilization is plotted against the degree of multiprogramming. As the degree of multiprogramming increases, CPU utilization also increases, although more slowly, until a maximum is reached. If the degree of multiprogramming is increased even further, thrashing sets in, and CPU utilization drops sharply. At this point, to increase CPU utilization and stop thrashing, we must *decrease* the degree of multiprogramming.

How to Limit Multiprogramming ?????

Thrashing can be handled by-

Local Replacement Algorithm

With local replacement

if one process starts thrashing,

it can not steel frames of other process

To prevent thrashing, we must provide a process with as many frames it needs

But How we know about, How many frames it needs ??????????

The Locality Model

The locality model states that , as a process executes,
- it moves from locality to locality

A locality is a set of pages that are actively used together

For example if a Function is called , it defines a new locality. When we exit the function the process leaves this locality, since the local variables and functions are no longer in use.

Suppose we allocate a process, enough frames according to its locality,

It will fault for pages in its locality until all these pages are in memory, then it will not fault again until it changes locality

BREADLY's ANOMALY

PAGE FAULT RATE > PAGE MISS RATE

CPU Utilization is inversely proportional to PAGE FAULT

CPU Utilization	**1/ Page Fault**

Increase in PAGE FAULT will decrease the CPU Utilization

OR

CPU Utilization	**1/ Thrashing**