Operating System

BIM IV Semester

Credits: 3
Lecture Hours:48

Er. Santosh Bhandari,
(Master Computer Science)

Unit-4

Memory Management:

Introduction

Definition: Method of managing primary memory.
Goal: Efficient organization of memory

-The task of subdividing the memory among different processes is called Memory Management.
-Memory management is a method in the operating system to manage **operations between main memory and disk** during process execution.

Introduction

Why Memory Management is Required?

- To achieve a degree of **multiprogramming** and proper utilization of memory, memory management is important.
- Allocate and de-allocate memory before and after process execution.
- To keep track of used memory space by processes.
- To minimize fragmentation issues.
- To proper utilization of main memory.
- To maintain data integrity while executing of process.

-To achieve a degree of multiprogramming, we need to increase the size of RAM

Logical and Physical Address Space

Logical Address Space:

-An address generated by the CPU is known as a “Logical Address”.

-It is also known as a Virtual address.

-Logical address space can be defined as the size of the process.

-A logical address can be changed.

Physical Address Space:

-An address seen by the memory unit (i.e. the one loaded into the memory address register of the memory) is commonly known as a “Physical Address”.

-A Physical address is also known as a Real address.

-The set of all physical addresses corresponding to these logical addresses is known as Physical address space.

Static and Dynamic Loading

Loading a process into the main memory is done by a loader. There are two different types of loading :

Static Loading: Static Loading is basically loading the entire program into a fixed address. It requires more memory space.

Dynamic Loading: The entire program and all data of a process must be in physical memory for the process to execute. So, the size of a process is limited to the size of physical memory.

Monoprogramming vs. Multi-programming

Monoprogramming:

-only one program runs at a time.

-Memory contains only one program at a time

-CPU utilization is low

-Fixed-size partition is used.

E.g. Old mobile OS

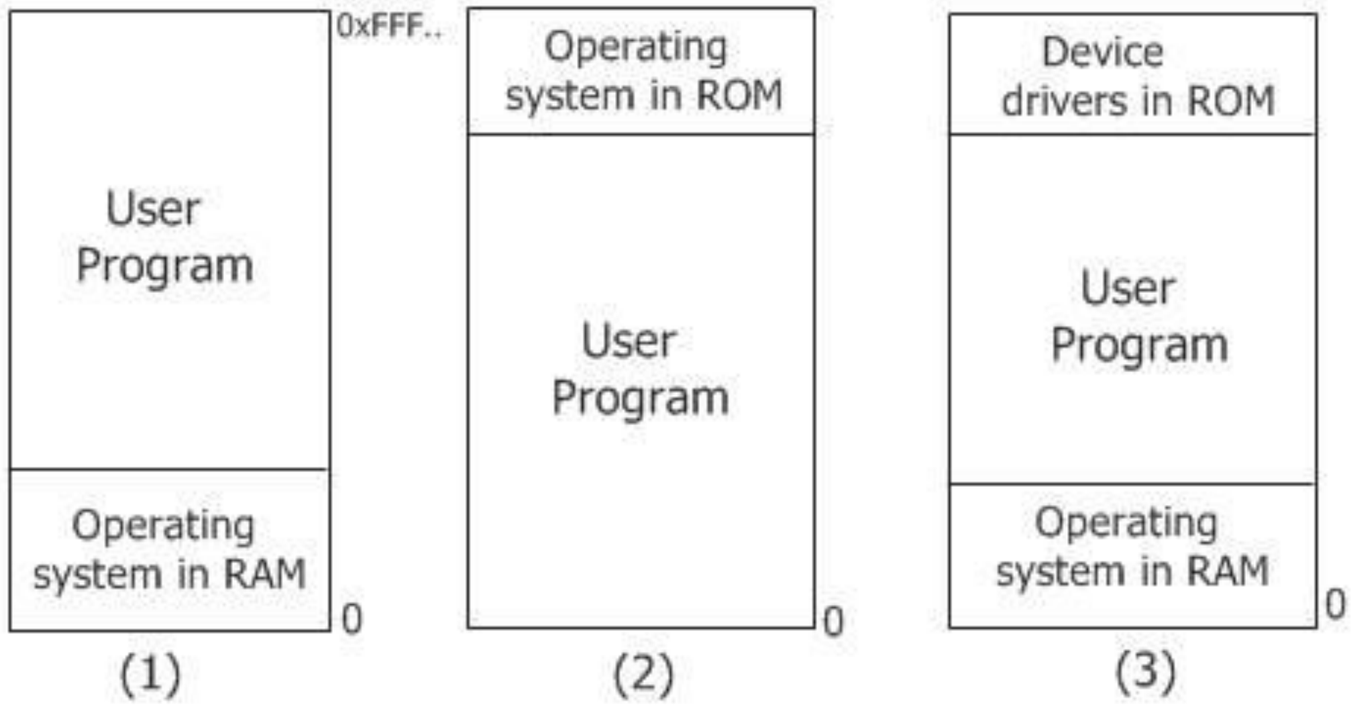
Monoprogramming vs. Multi-programming

There are three different takes on the monoprogramming model.

Model 1 (Fig 1): The OS located at the bottom of the memory in RAM

Model 2 (Fig 2): OS located at the beginning of the memory that is stored in ROM.

Model 3 (Fig 3): the device drivers are located at the top of the memory in a ROM, whereas the remaining components of the system are located below the device drivers in RAM).



Monoprogramming vs. Multi-programming

- The first model is one that is utilized on mainframes and minicomputers.
- The second model is implemented on a select number of embedded systems and palmtop computers.
- The third model was the one that early personal computers like MS-DOS used.

Monoprogramming vs. Multi-programming

Advantages of Monoprogramming:

Simplicity: Because they do not need to manage multiple programs or allocate resources to different processes, monoprogramming operating systems are relatively simple.

-The operating system has complete control over the computer's resources because only one program runs at a time.

-This makes allocating resources and prioritizing tasks easier, which can lead to improved performance and faster processing.

-Security: Because no other programs are running that could interfere with the system or cause security breaches.

Monoprogramming vs. Multi-programming

Disadvantages of Monoprogramming:

-Monoprogramming operating systems can only run one program at a time and have limited functionality.

-Wasted Resources: The computer's resources might not be fully utilized because only one program is running at once.

-This may result in a waste of resources such as memory, CPU cycles, and other resources.

Poor User Experience

Monoprogramming vs. Multi-programming

Multi-programming:

-The concurrent application of more than one program in the main memory is known as multiprogramming.

-CPU is single

-The number of users is one at a time.

-The memory can hold several programs at a time.

-The resources are allocated to different programs.

Eg: Window

Requirements of Memory Management System

Memory management keeps track of the status of each memory location, whether it is allocated or free.

-It allocates the memory dynamically to the programs at their request and frees it for reuse when it is no longer needed.

Requirement for memory management:

1. Relocation
2. Protection
3. Sharing
4. Logical Organization
5. Physical organization

Requirements of Memory Management System

Relocation:

- The term program relocatability refers to the ability to load and execute a given program into memory.
- Memory management technique in which the system stores and retrieves data from secondary storage for use in main memory is called paging.
- In reality, the program may be loaded at different memory locations, which are called physical addresses.
- Relocation is way to map virtual addresses into physical addresses.

Requirements of Memory Management System

Types of relocation:

There are two types of relocation in memory management.

1. Static Relocation

Static relocation is performed before or during the loading of the program into main memory, by a relocating linker/ loader.

2. Dynamic Relocation

-Mapping from the virtual address space to the physical address space is performed at run-time.

-This runtime mapping from virtual address to physical address is performed by a hardware device known as a memory management unit.

Requirements of Memory Management System

Protection:

-Memory protection is a hardware mechanism that separates different parts of memory and ensures that each process can only access its own memory space.

-This prevents one process from accessing or modifying the memory of another process, thereby protecting the system from malware and other malicious attacks.

-The primary goal of safeguarding memory is to avert an application from accessing RAM without permission.

Requirements of Memory Management System

Different Ways of Memory Protection

1. Segmentation

Memory is segmented into sections, every single one which can have a separate set of access rights. An OS kernel segment, for instance, might be read-only, whereas a user data segment could have been designated as read-write.

2. Paged Virtual Memory

Memory is divided into pages in paged virtual memory, and each page can be saved to its own place in physical memory.

Requirements of Memory Management System

3. Protection keys

Each RAM page has a set of bits called encryption keys.

Accessibility to the page can be controlled using these bits.

Swapping

-To increase CPU utilization in multiprogramming, a memory management scheme known as swapping can be used.

-Swapping is the process of bringing a process into memory and then temporarily copying it to the disc after it has run for a while.

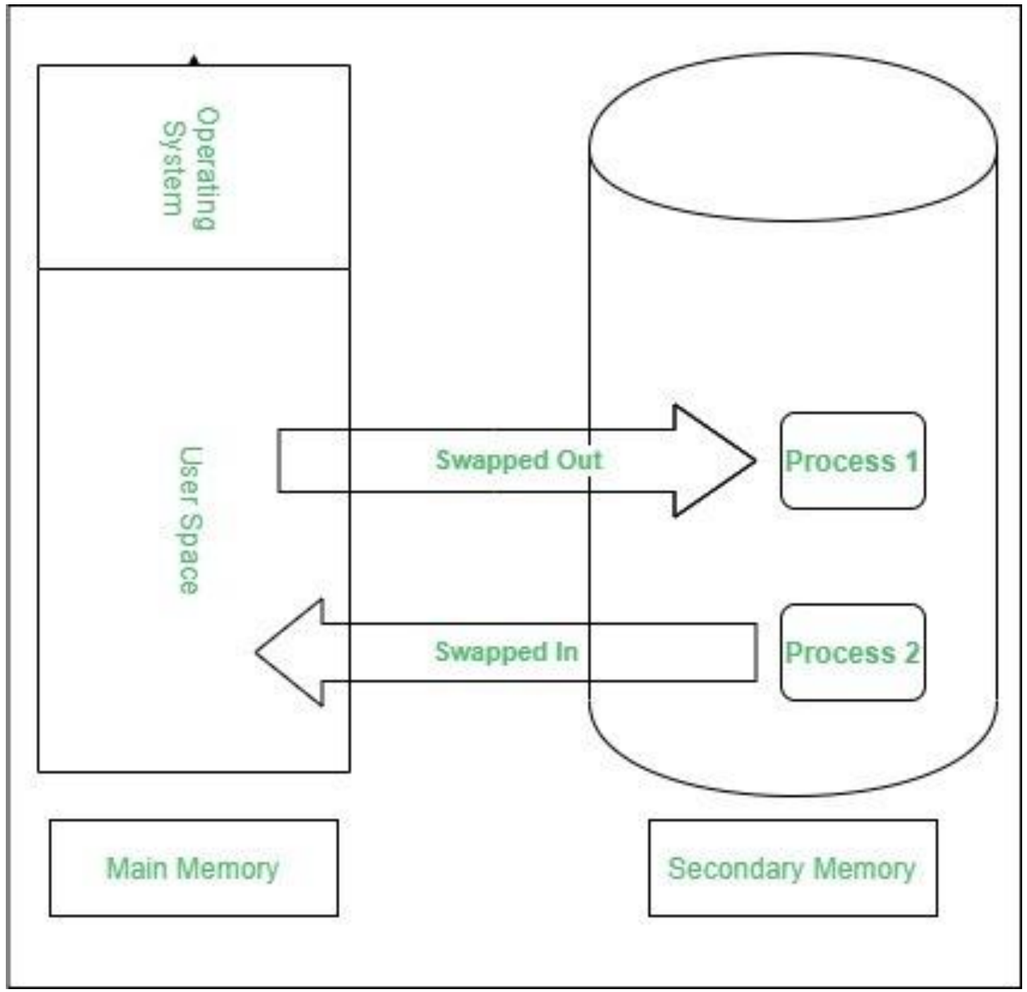
-The purpose of swapping in an operating system is to access data on a hard disc and move it to RAM

Swapping

Two types:

Swap-out: is a technique for moving a process from RAM to the hard disc.

Swap-in: is a method of transferring a program from a hard disc to main memory, or RAM.



Swapping

Advantages

1. If there is low main memory so some processes may has to wait for much long but by using swapping process do not have to wait long for execution on CPU.
2. It utilize the main memory.
3. Using only single main memory, multiple process can be run by CPU using swap partition.
4. The concept of virtual memory start from here and it utilize it in better way.
5. This concept can be useful in priority based scheduling to optimize the swapping process.

Free space management

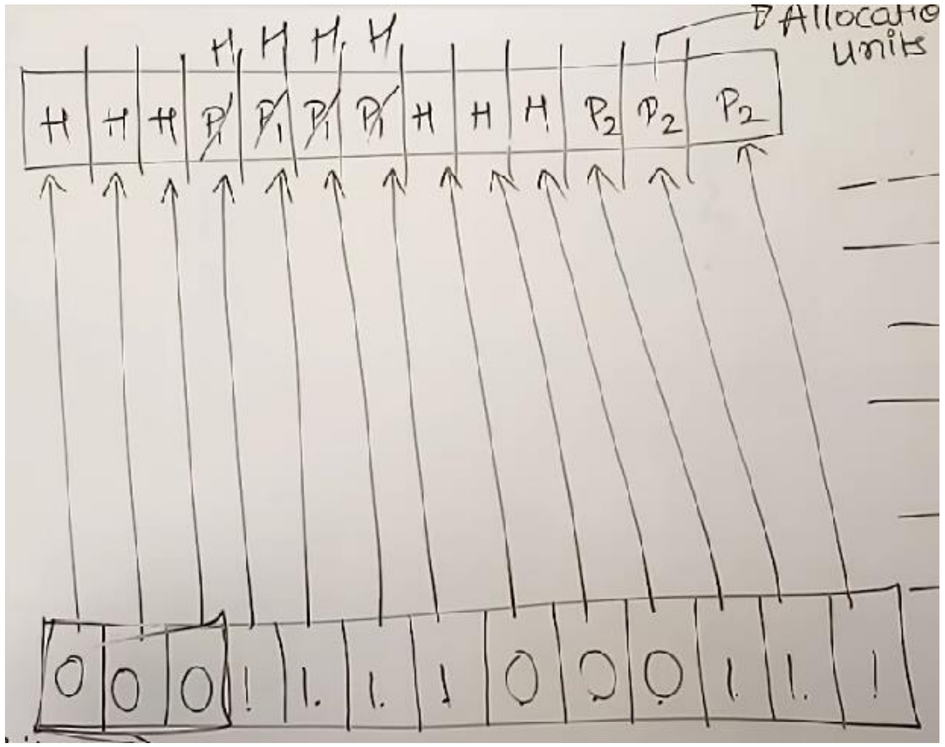
Free space management is a critical aspect of operating systems as it involves managing the available storage space on the hard disk or other secondary storage devices.

OS uses various free space management techniques:

1. Bit-MAP
2. Linked List

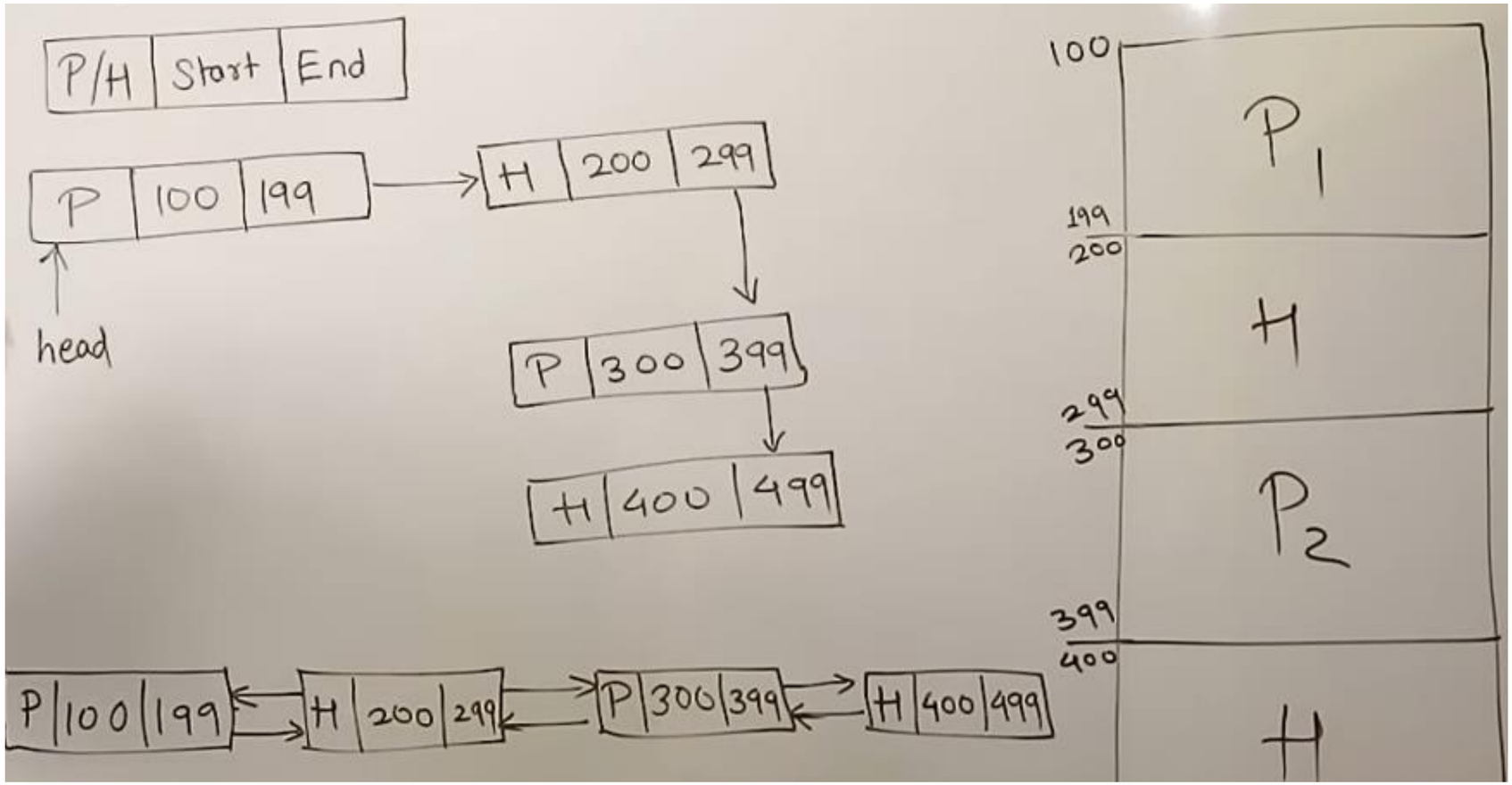
Bit-MAP

- It is data structure used in memory allocation
- With bitmap, memory is divided into allocation units.
- Corresponding to each allocation unit there is a bit in a bitmap.
- Bit is 0 if the unit is free (hole).
- Bit is 1 if unit is occupied (process).
- If a new process arrives, it searches for a sequence of holes in a data structure.
- If any process is completed then the process becomes hole.



Linked List

-It is a non-contiguous allocation
-In this approach, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block.



Linked List

Advantages:

-File size can increase

Disadvantage:

-Large seek time

-Direct access difficult

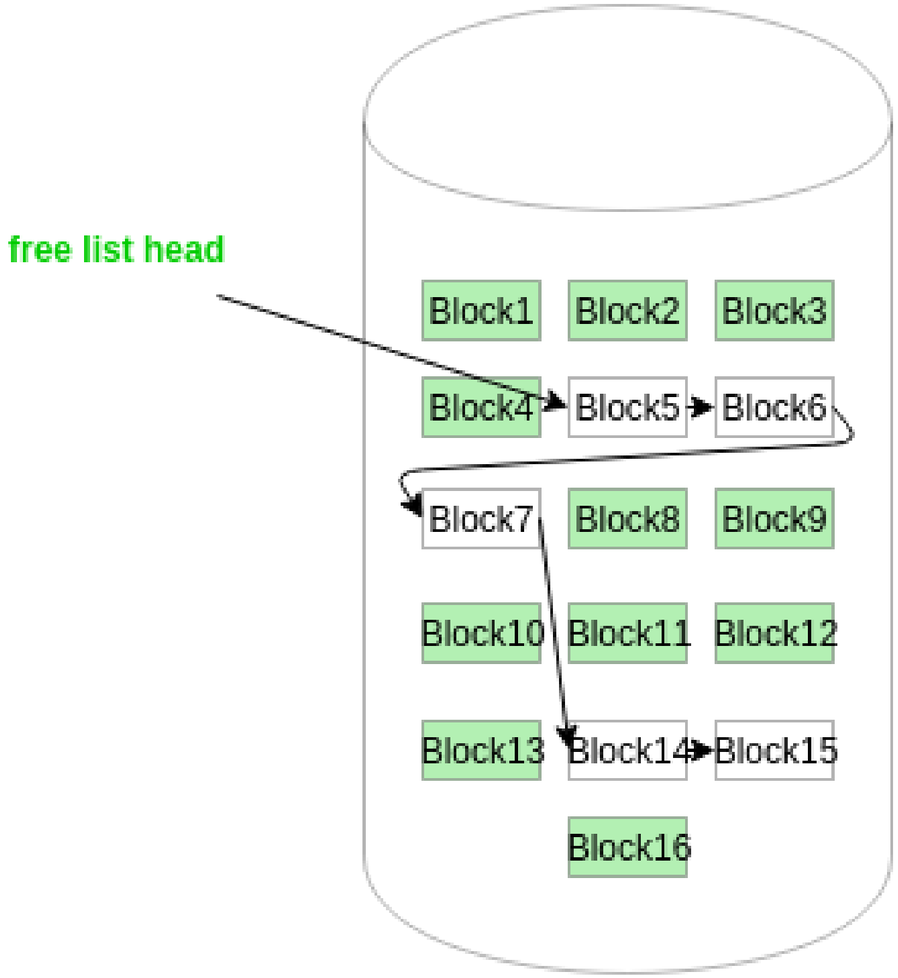


Figure - 2

Memory management techniques

1. Contiguous Memory Allocation

- Static portioning/Fix-size Partitioning Method
- Dynamic portioning/Variable Partitioning/Flexible Partitioning

2. Non-contiguous memory allocation

- Paging (Multilevel paging, inverted paging)
- Segmentation (Segmented paging)

Fixed-size partitioning:

-In fixed partitioning, the memory is divided into fixed-size partitions, and each partition is assigned to a process.

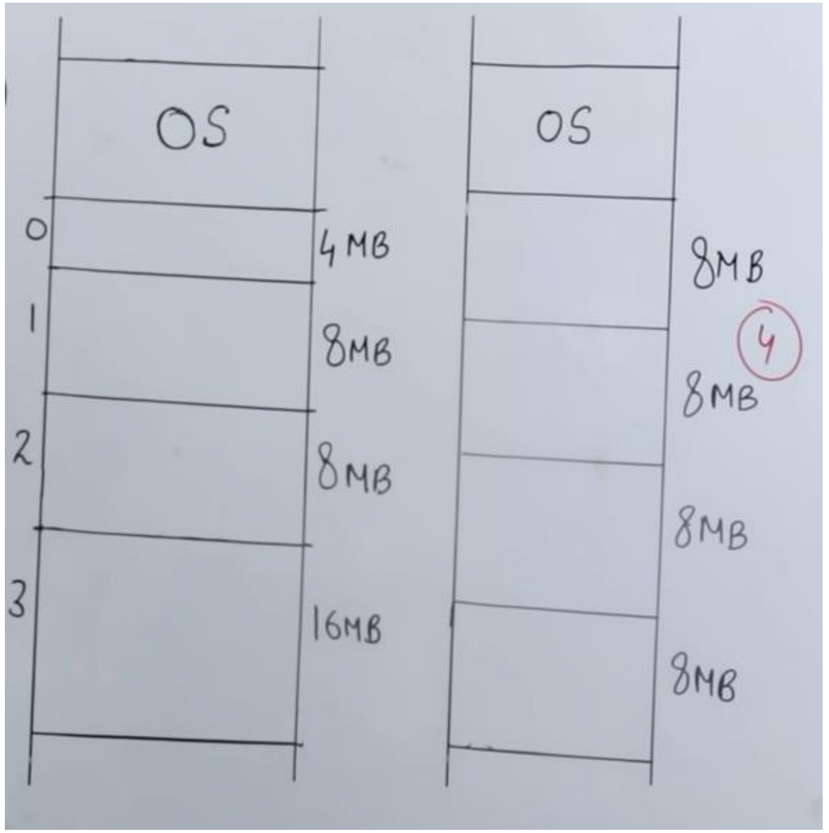
-Number of partitions are fixed

-Size of each partition may or may not be same

-Each process in this method of contiguous memory allocation is given a fixed size continuous block in the main memory.

-In contiguous memory allocation spanning is not allowed.

Spanning: divide the process into small parts and store them in different memory partitions.

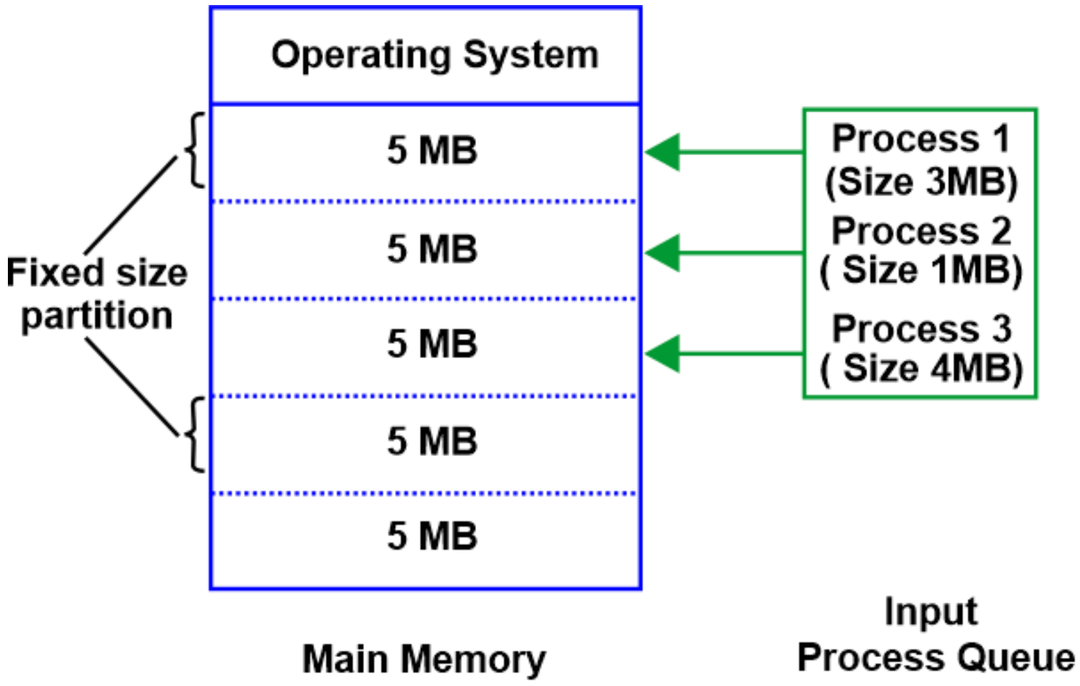


Fixed-size partitioning:

-If process P1 arrives and its size is 3 MB. It fits into the first partition which size is 5 MB. Now 2 MB is a waste. Which is **internal fragmentation**

-If the process size is larger than the memory partition, then we can't accommodate it.

-If number of processes are greater than the number of memory partitions, then the process can't accommodate.



Fixed-size partitioning:

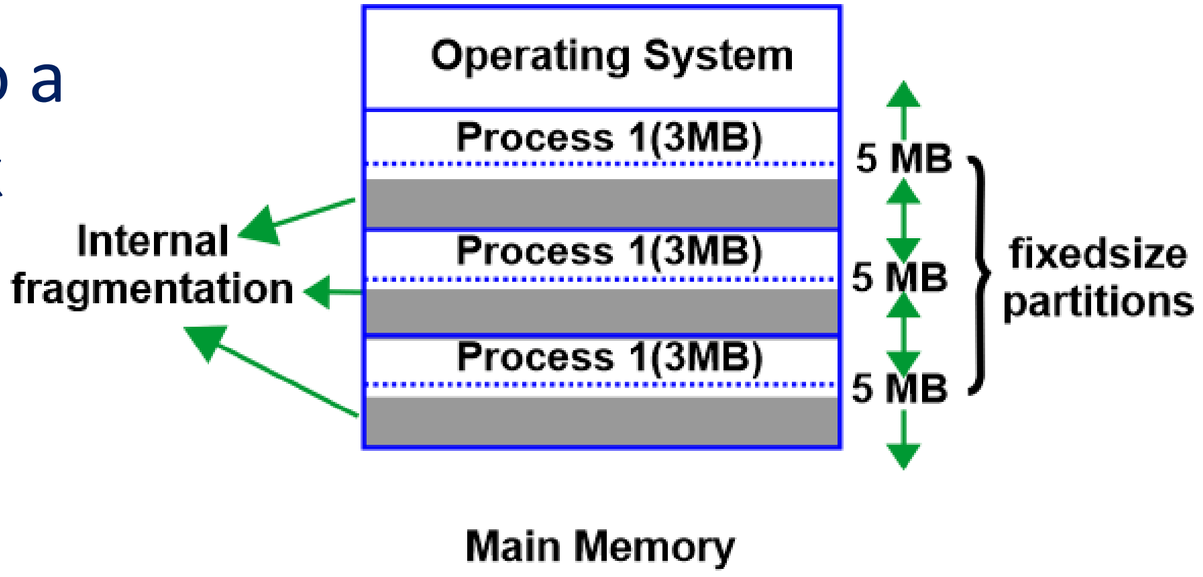
Advantages

- This strategy is easy to employ because each block is the same size.
- It is simple to keep track of how many memory blocks are still available, which determines how many further processes can be allocated memory.
- This approach can be used in a system that requires multiprogramming since numerous processes can be maintained in memory at once.

Fixed-size partitioning:

Disadvantages:

- **Internal fragmentation**
- We won't be able to allocate space to a process whose size exceeds the block since the size of the blocks is fixed.
- External Fragmentation: if we add all waste space (2+2+2), the total available space will be 6MB in this example. If new process arrives which size is 6MB but we can’t store this process. Which is external fragmentation.



Dynamic or Variable or Flexible Partitioning

-It is a part of the Contiguous allocation technique.

-It is used to alleviate the problem faced by Fixed Partitioning.

Features:

- Initially, RAM is empty and partitions are made during the run-time according to the process's need instead of partitioning during system configuration.
- The size of the partition will be equal to the incoming process.
- The partition size varies according to the need of the process.
- The number of partitions in RAM is not fixed and depends on the number of incoming processes and the Main Memory's size.

Dynamic or Variable or Flexible Partitioning

Advantages:

-There will be no unused space left in the partition.

-A process can be loaded until the memory is empty.

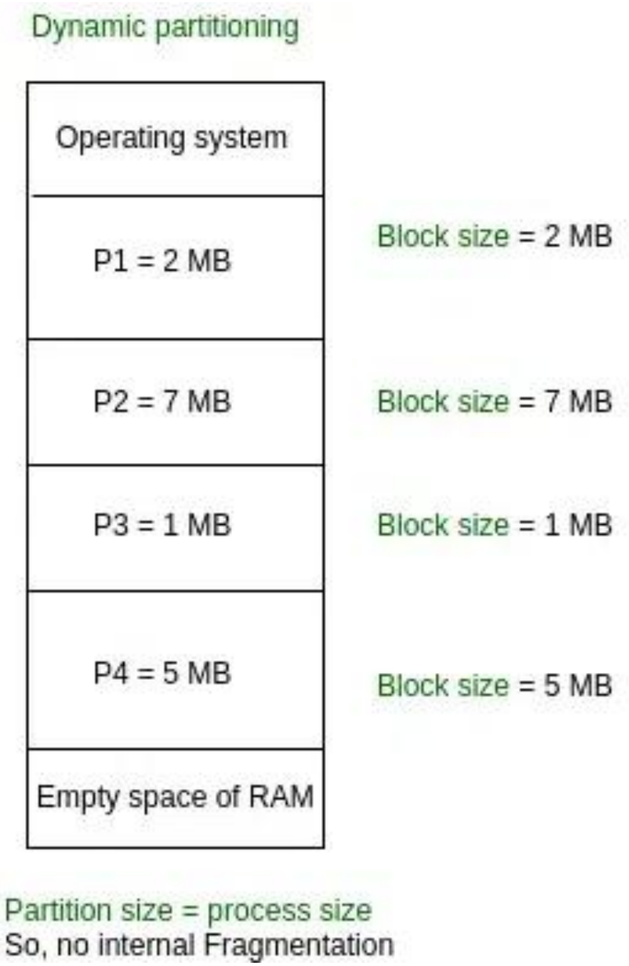
-No Limitation on the size of the Process

-No internal fragmentation

Disadvantage:

-Difficult Implementation, since memory allocation is performed during run time.

-External fragmentation



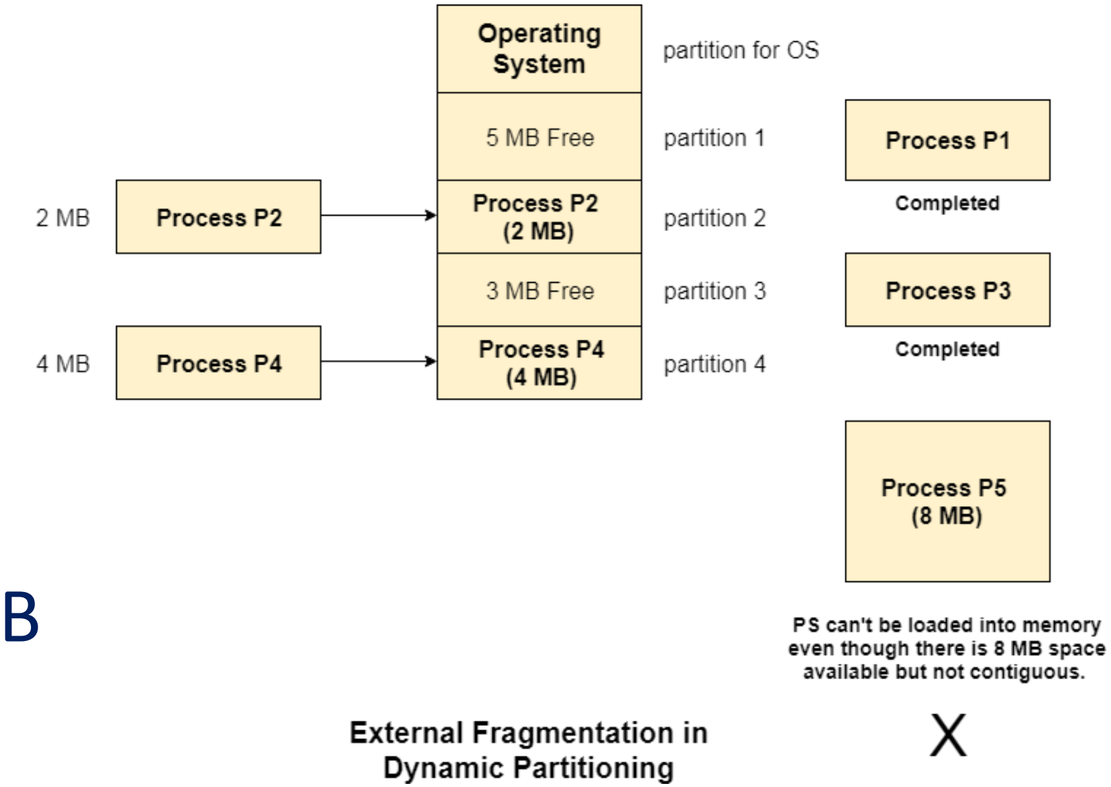
Dynamic or Variable or Flexible Partitioning

Disadvantage:

-External fragmentation: consider three processes P1 (1 MB), P2 (3 MB) and P3 (1 MB) are being loaded in the respective partitions of the main memory.

After some time P1 and P3 got completed and their assigned space is freed.

Now there are two unused partitions (1 MB and 1 MB) available but they cannot be used to load a 2 MB process in the memory since they are not contiguously located.

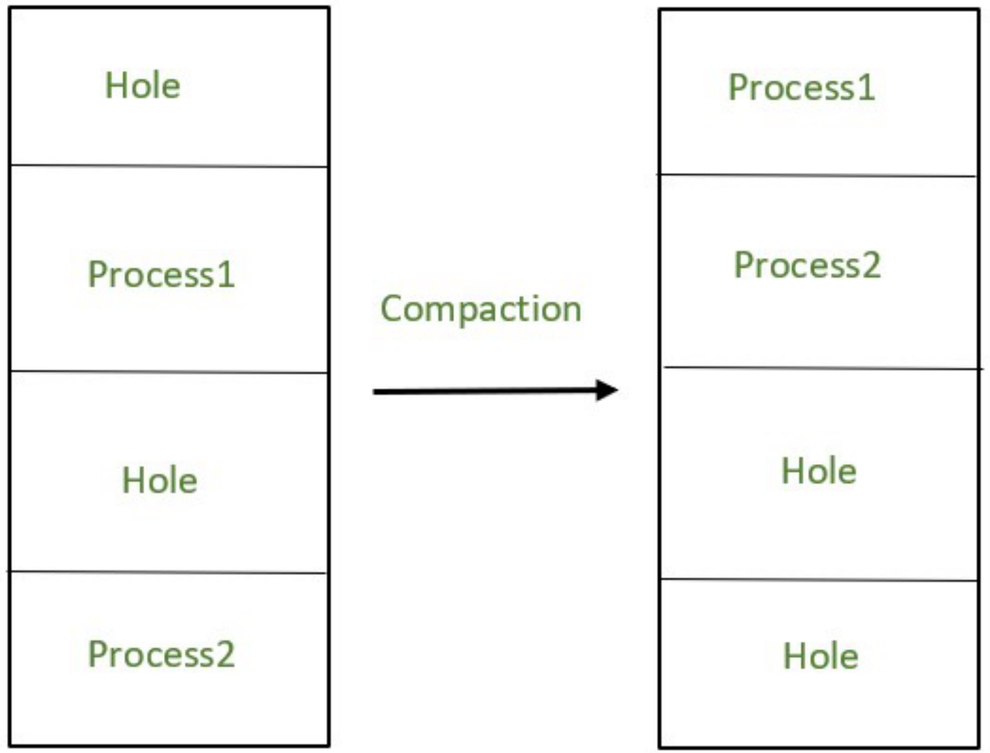


External Fragmentation in Dynamic Partitioning

Compaction technique

-To remove this external fragmentation we use the compaction technique.
-Compaction is a technique to collect all the free space together and all occupied space together. Free memory can be used to run other processes.

BUT….
This technique is not efficient, because we need to stop the running process to bring it together.



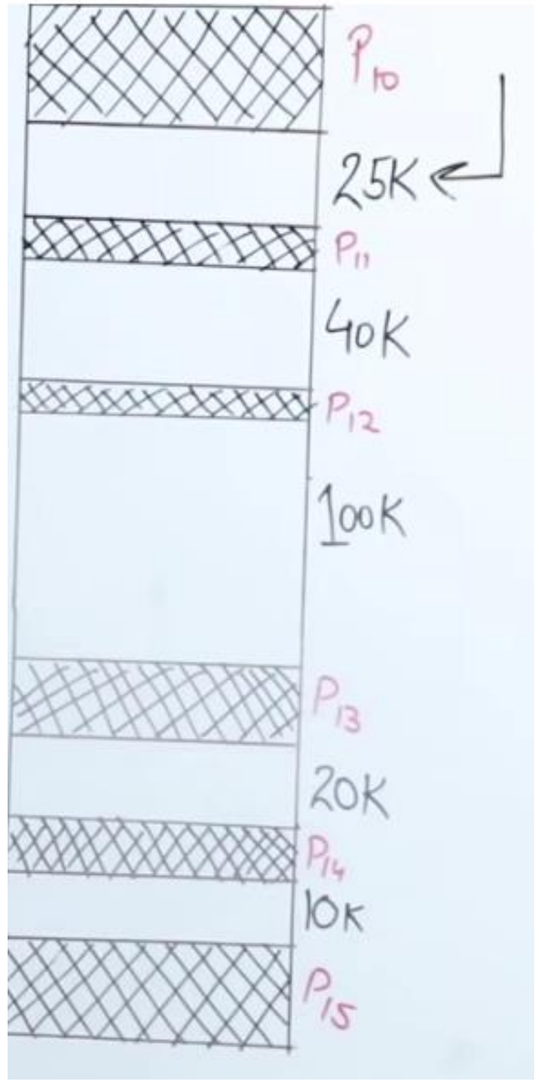
Contiguous memory allocation techniques: Algorithms

First-Fit: Allocate the first hole that is big enough

Next fit: Same as first fit but start search always from last allocated hole.

Best Fit: Allocate the smallest hole that is big enough

Worst Fit: Allocate the largest hole.



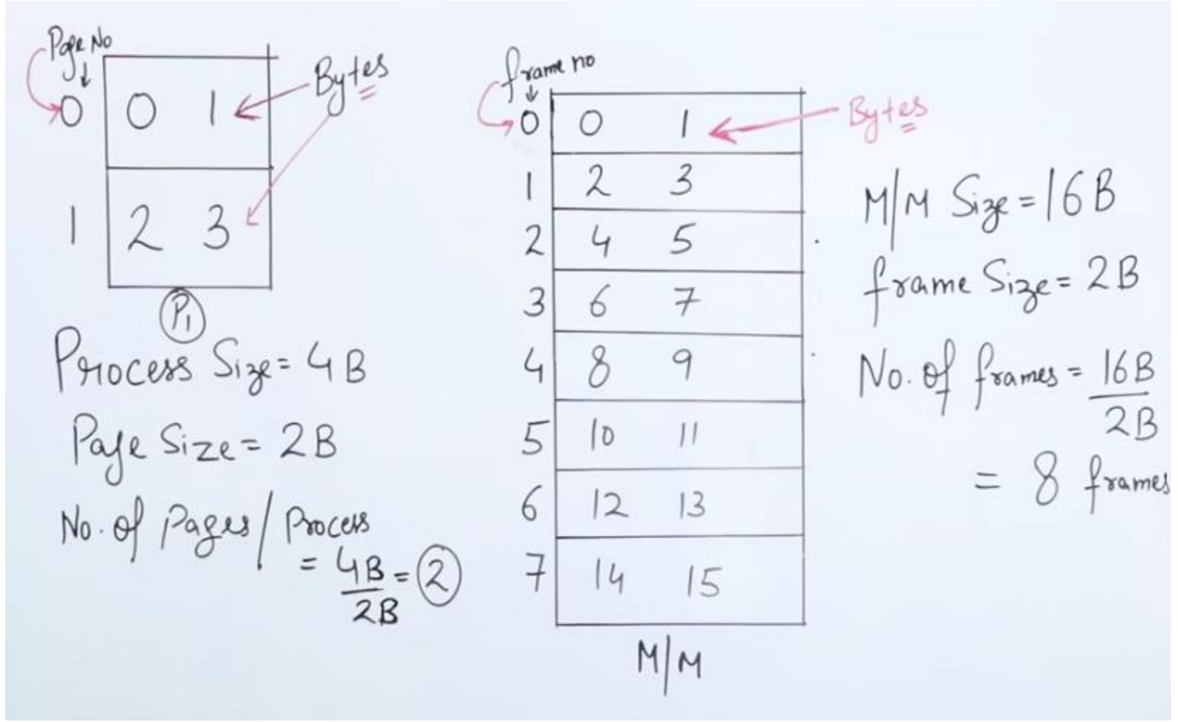
Non Contiguous memory: Virtual memory

Paging:

-In paging, process is divide in to equally size block called pages and memory is divided into fixed-size blocks called frames, and pages are stored in to the frame.

-We divide process in equal page size and feet into to frame of the main memory.

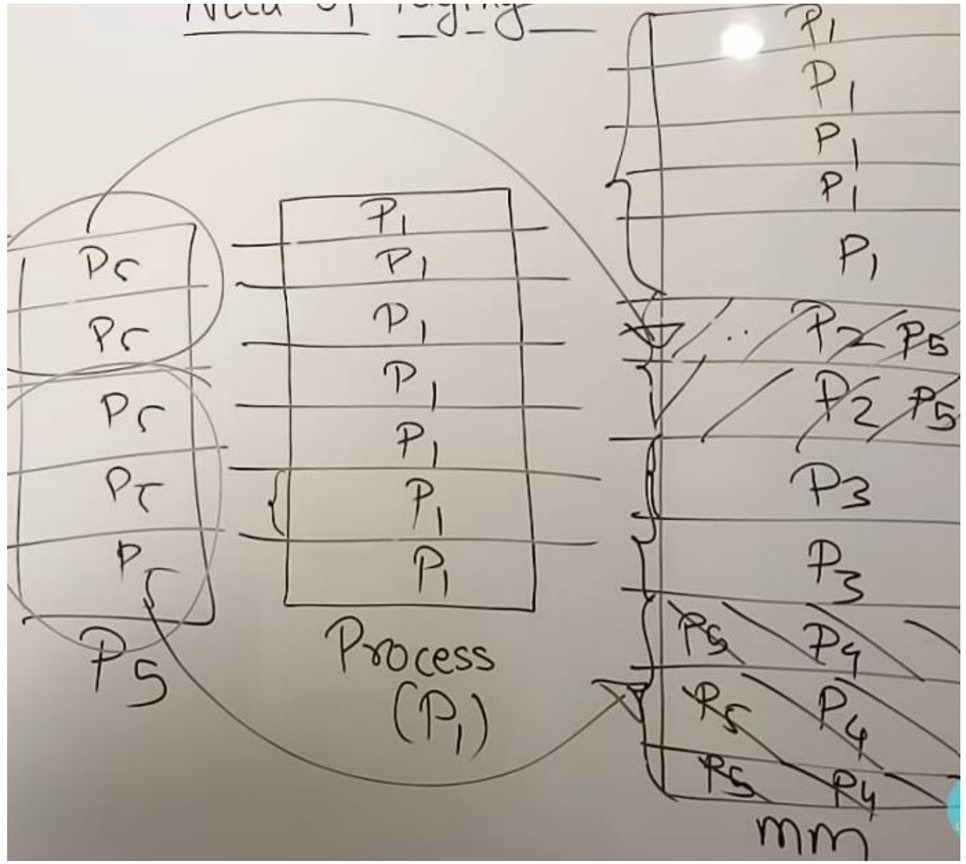
-Page size == frame size (Page size and frame size must be the same)



-Each page is of the same size, and the size is a power of 2, such as 4KB or 8KB.

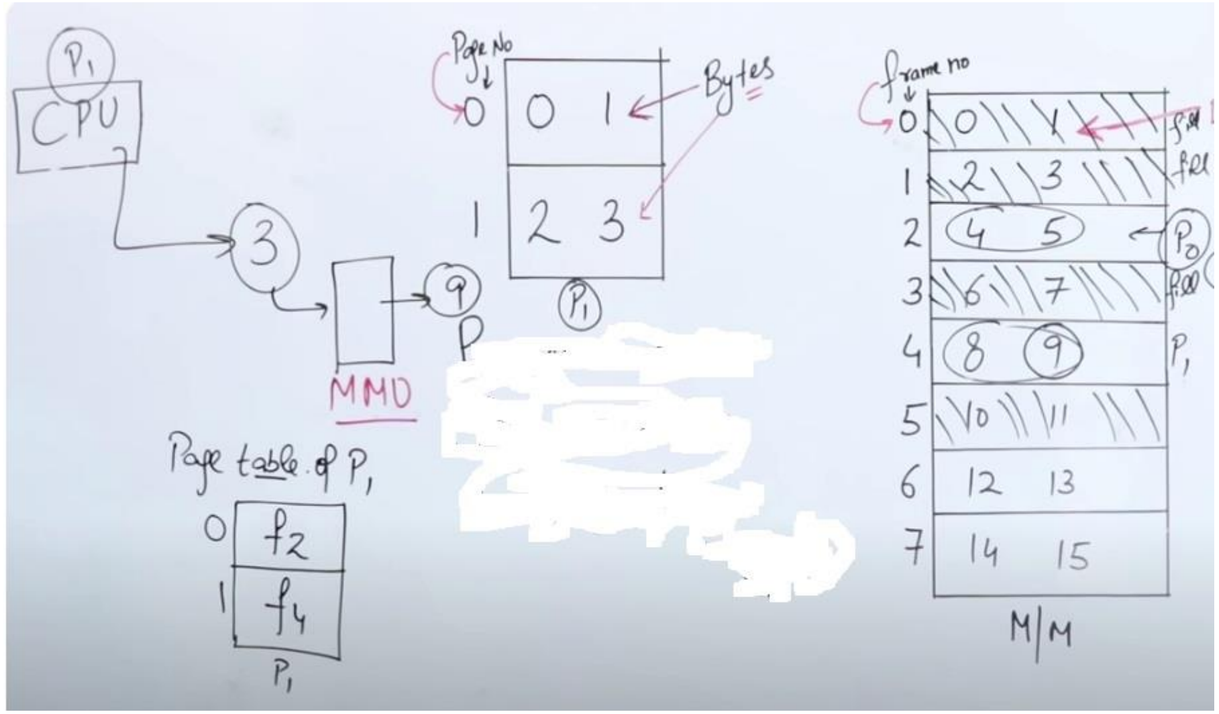
Non Contiguous memory: Virtual memory

-Paging removes External fragmentation:



Mapping

-The CPU always accesses the processes through their logical addresses. However, the main memory recognizes physical addresses only.

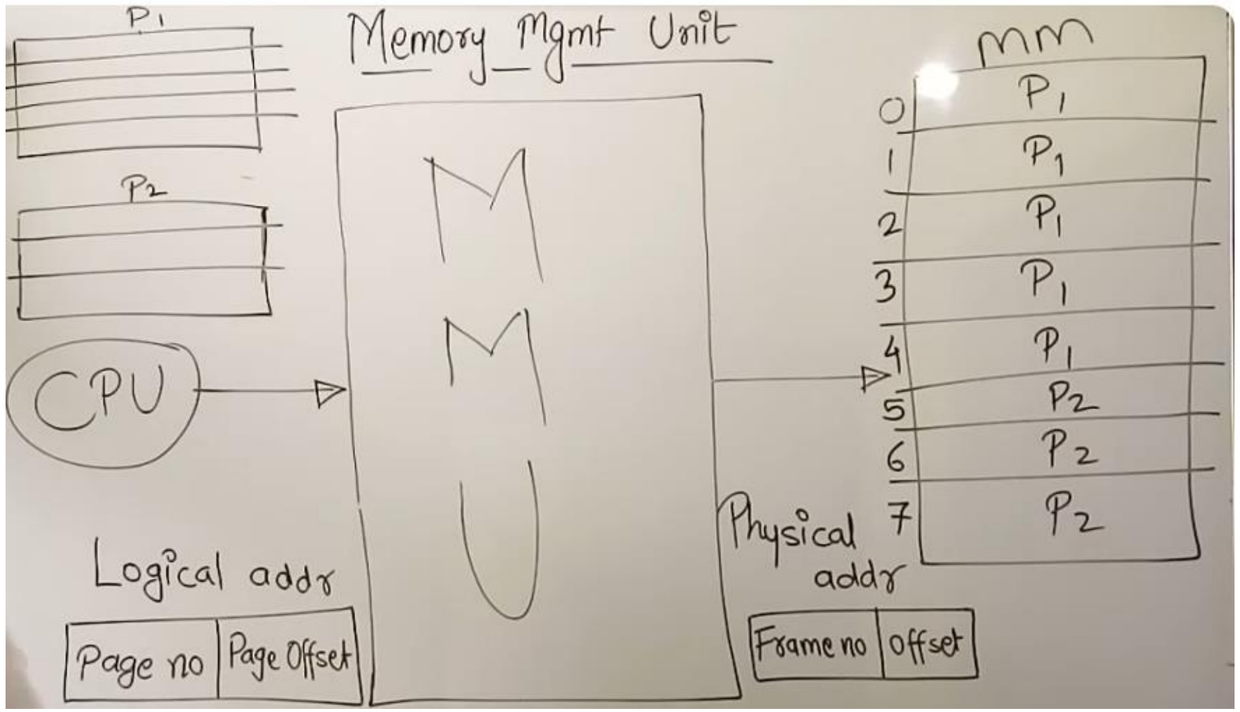


-To map the CUP address (logical) and main memory (physical address) it is required to convert a logical address to a physical address. Which is done by MMU.

-Memory Management Unit (MMU), maintains a page table

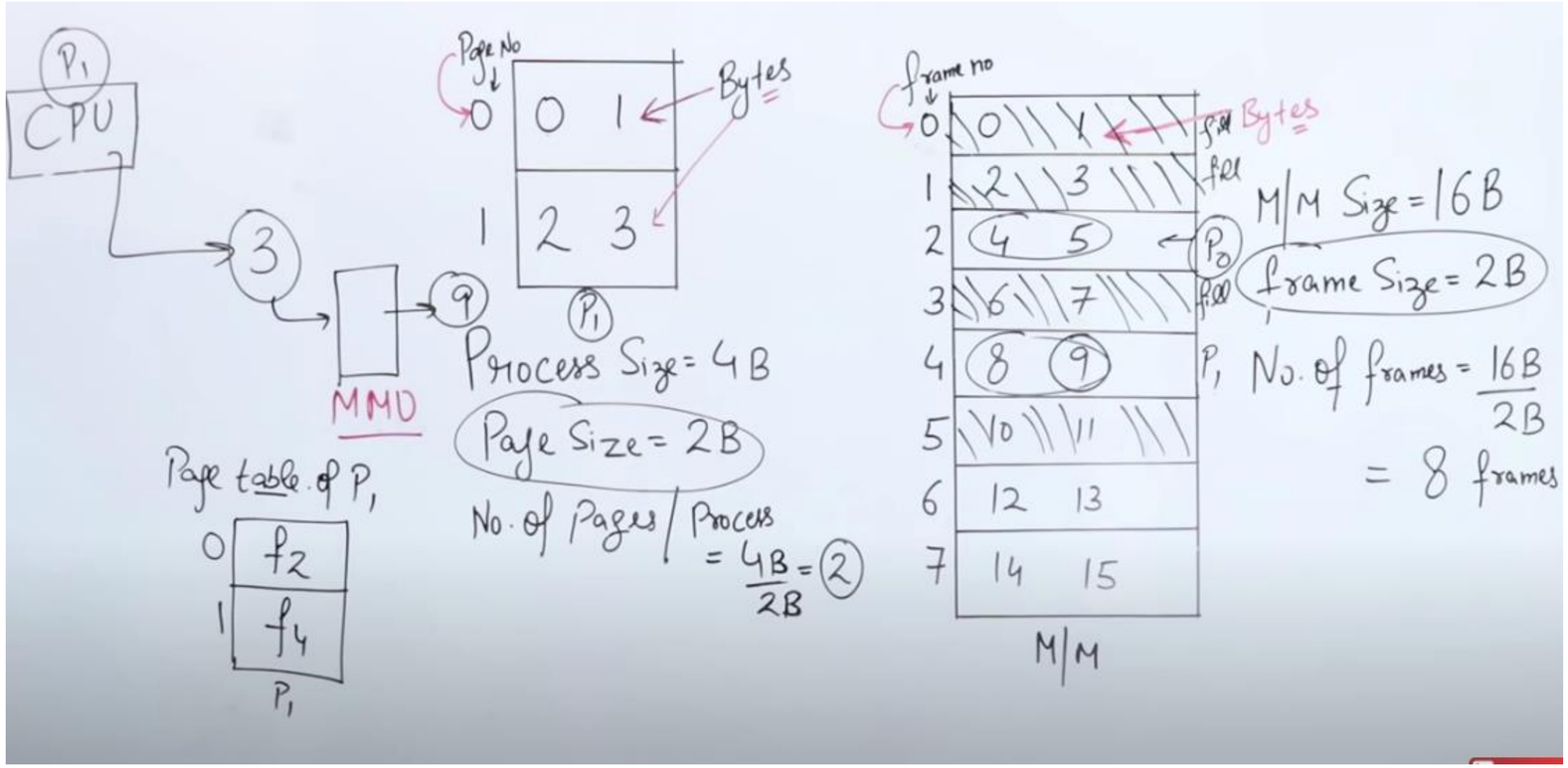
Memory Management Unit (MMU)

-CPU generates logical address
-Logical address contains two fields
[page no |page offset]
-offset= size of page
-Actual data is stored in Main memory which have different addresses.
-MMU converts logical address to physical address so that CPU can access process in main memory.
-MMU use **page table** to convert the address.

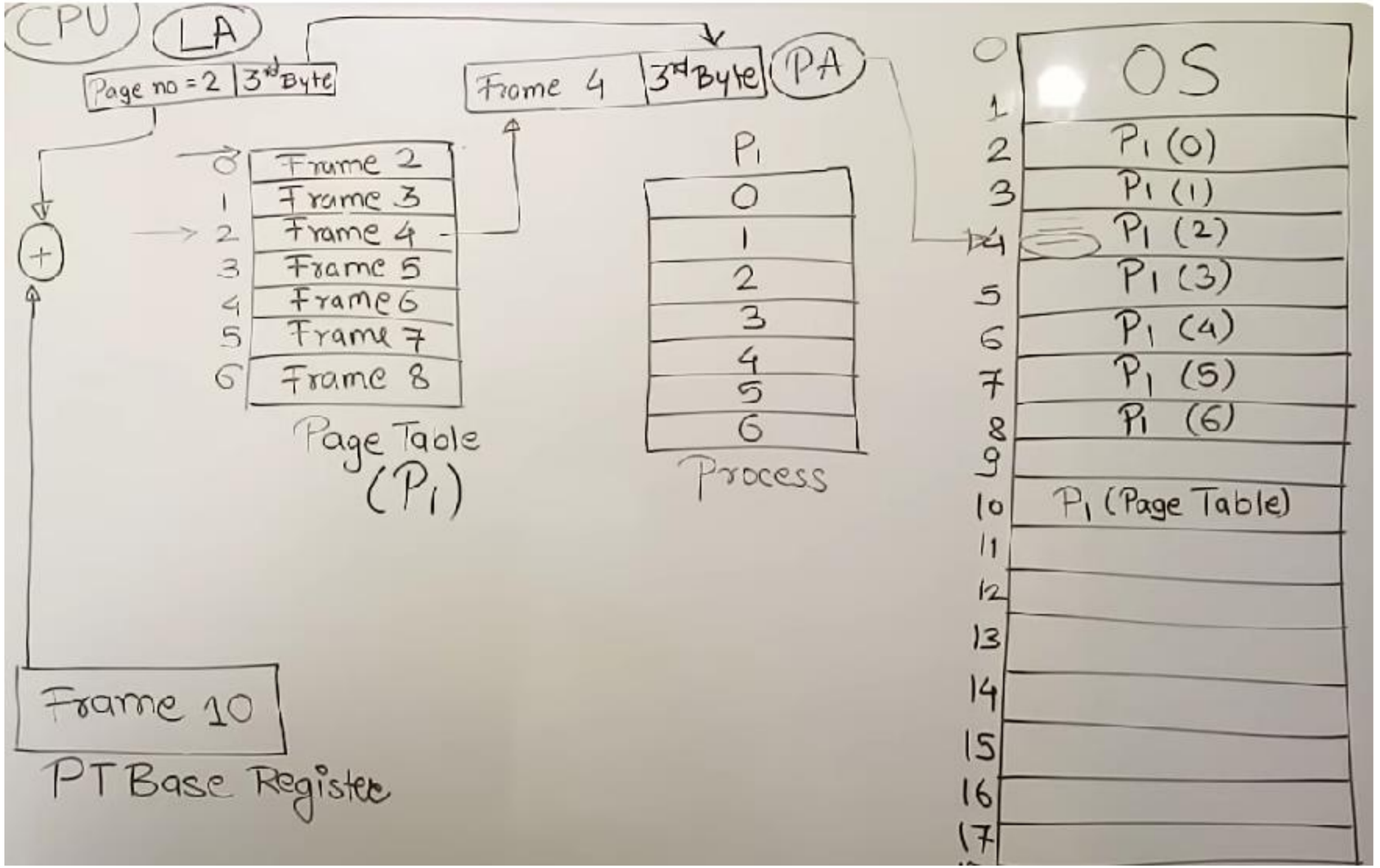


Page table

-Page table contains frame number. Which represents where this page is present in the main memory.
-Every process will have its page table.
-Page Table is a data structure used by the virtual memory system to store the mapping between logical addresses and physical addresses.



Page table Working



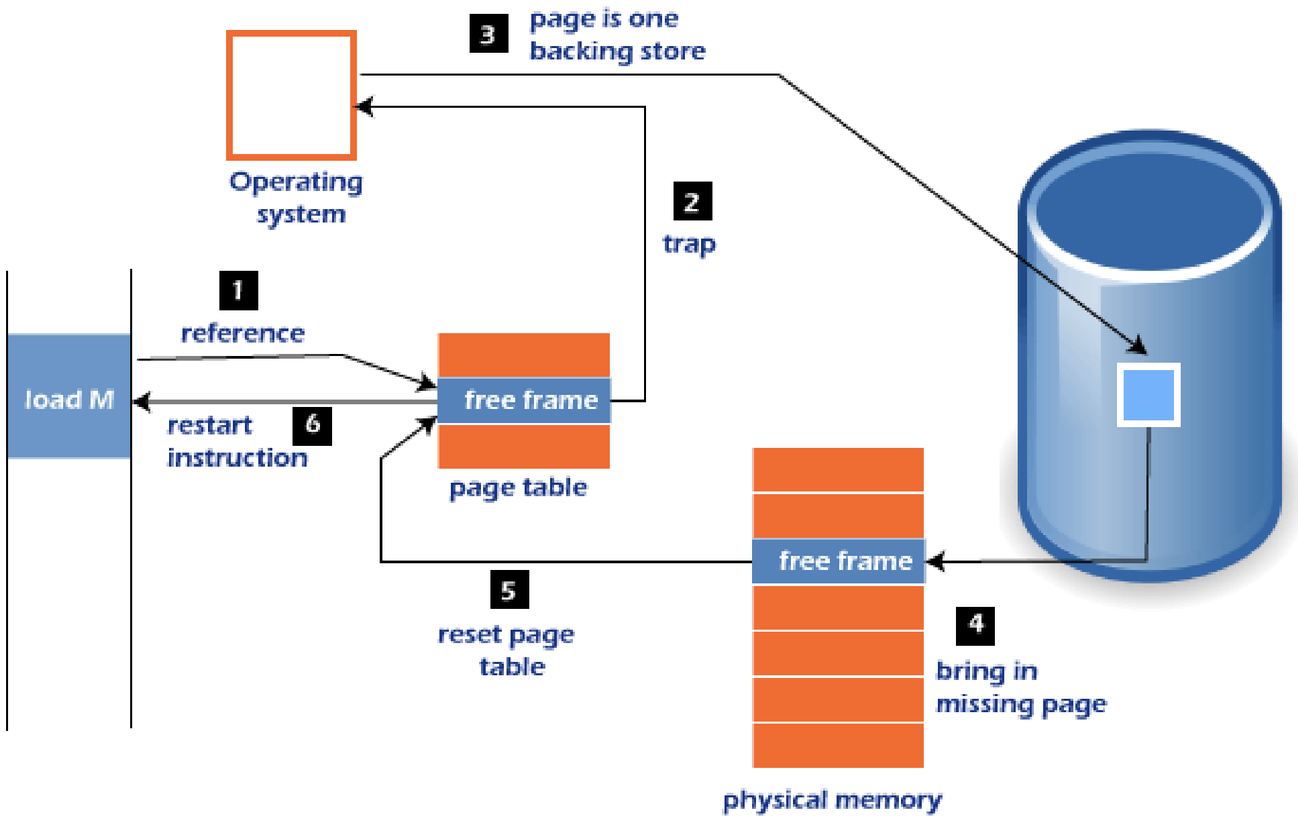
Page fault

-If the CPU is demanding a specific page for execution. But this page is not available in the main memory, this is called a page fault.
-If the frame number is absent in the page table, this is called a trap.
-The page fault primarily causes an exception, which is used to notify the operating system to retrieve the "pages" from virtual memory to continue operation.
-Once all of the data has been placed into physical memory, the program resumes normal operation.
-The Page fault process occurs in the background, and thus the user is unaware of it.

Handling Page fault

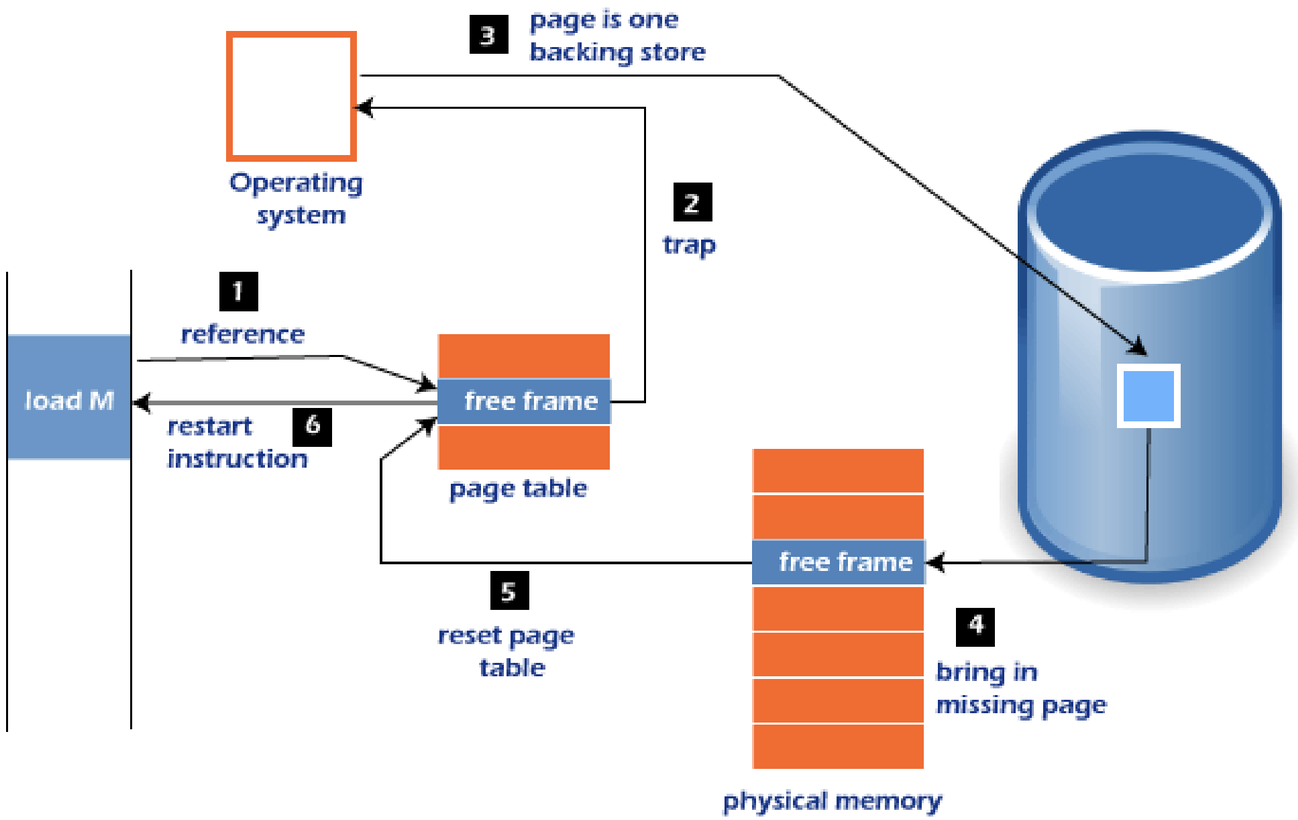
6 steps:

1. Firstly, an internal table for this process to assess whether the reference was valid or invalid memory access.
2. If the reference becomes invalid, the system process would be terminated. And it generates error (trap) and sends to OS.
3. After that, the free-frame list finds the free frame in the system.



Handling Page fault

4. Now, the disk operation would be scheduled to get the required page from the disk.
5. When the I/O operation is completed, the process's page table will be updated with a new frame number, and the invalid bit will be changed. Now, it is a valid page reference.
6. If any page fault is found, restart these steps from starting.



Page Fault Terminology

There are various page fault terminologies in the operating system.

1. Page Hit: When the CPU attempts to obtain a needed page from main memory and the page exists in main memory (RAM), it is referred to as a "PAGE HIT".

2. Page Miss: If the needed page does not exist in the main memory (RAM), it is known as "PAGE MISS".

3. Page Fault Time: The time it takes to get a page from secondary memory and recover it from the main memory after loading the required page is known as "PAGE FAULT TIME".

Page Fault Terminology

4. Hard Page Fault: If a required page exists in the hard disk's page file, it is referred to as a "HARD PAGE FAULT".

5. Soft Page Fault: If a required page is not located on the hard disk but is found somewhere else in memory, it is referred to as a "SOFT PAGE FAULT".

TLB (Translation Lookaside Buffer)

Drawbacks of Paging:

1. Size of Page table can be very big and therefore it wastes main memory.
2. CPU will take more time to read a single word from the main memory.

To solve this problem we use TBL

TLB (Translation Lookaside Buffer)

Translation look aside buffer (TLB)

-A Translation look aside buffer can be defined as a memory cache which can be used to reduce the time taken to access the page table again and again.

-It is a memory cache which is closer to the CPU and the time taken by CPU to access TLB is lesser then that taken to access main memory.

-TLB is faster and smaller than the main memory but cheaper and bigger than the register.

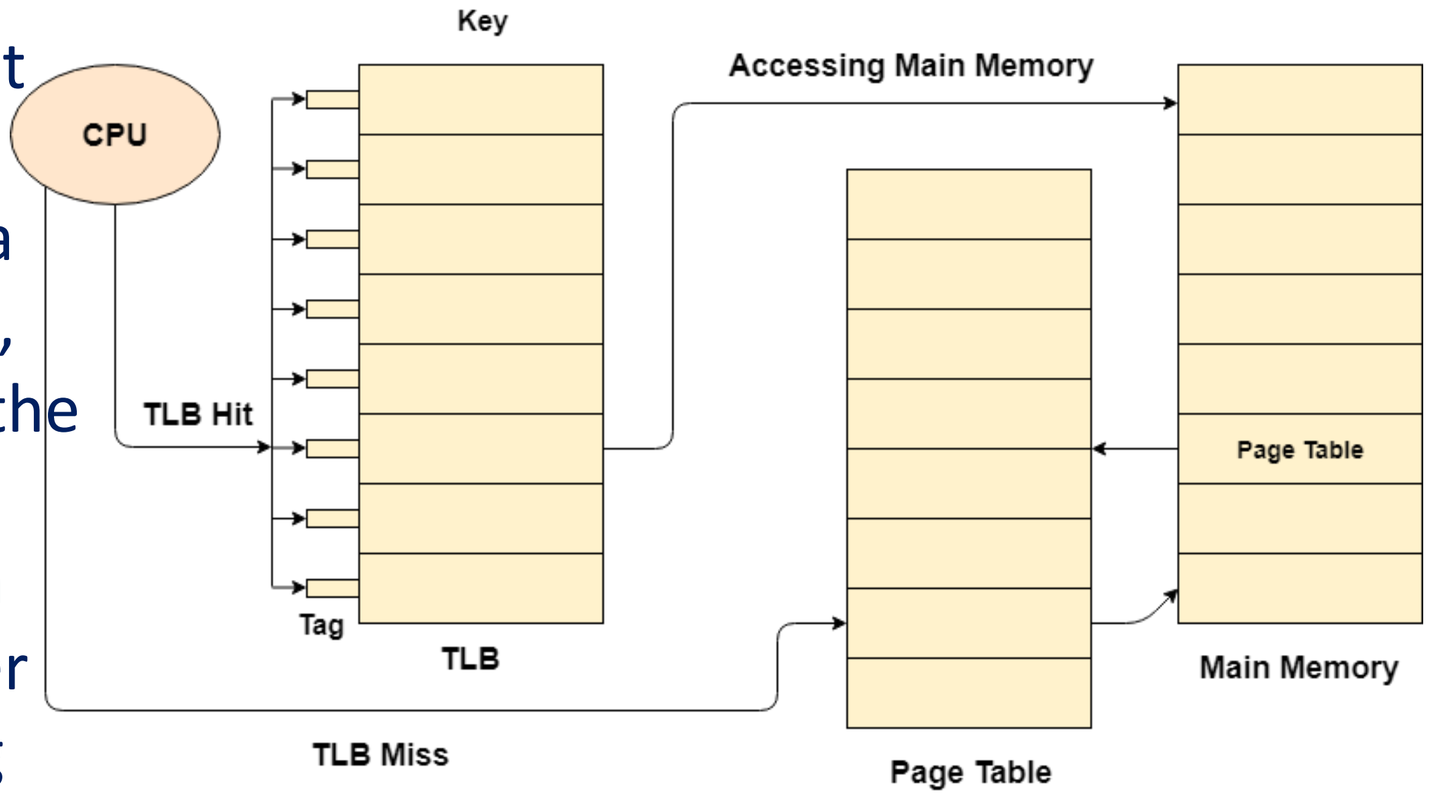
TLB (Translation Lookaside Buffer)

How TLB works:

-TLB contains page table entries that have been most recently used.

-the processor examines the TLB if a page table entry is present (TLB hit), the frame number is retrieved and the real address is formed.

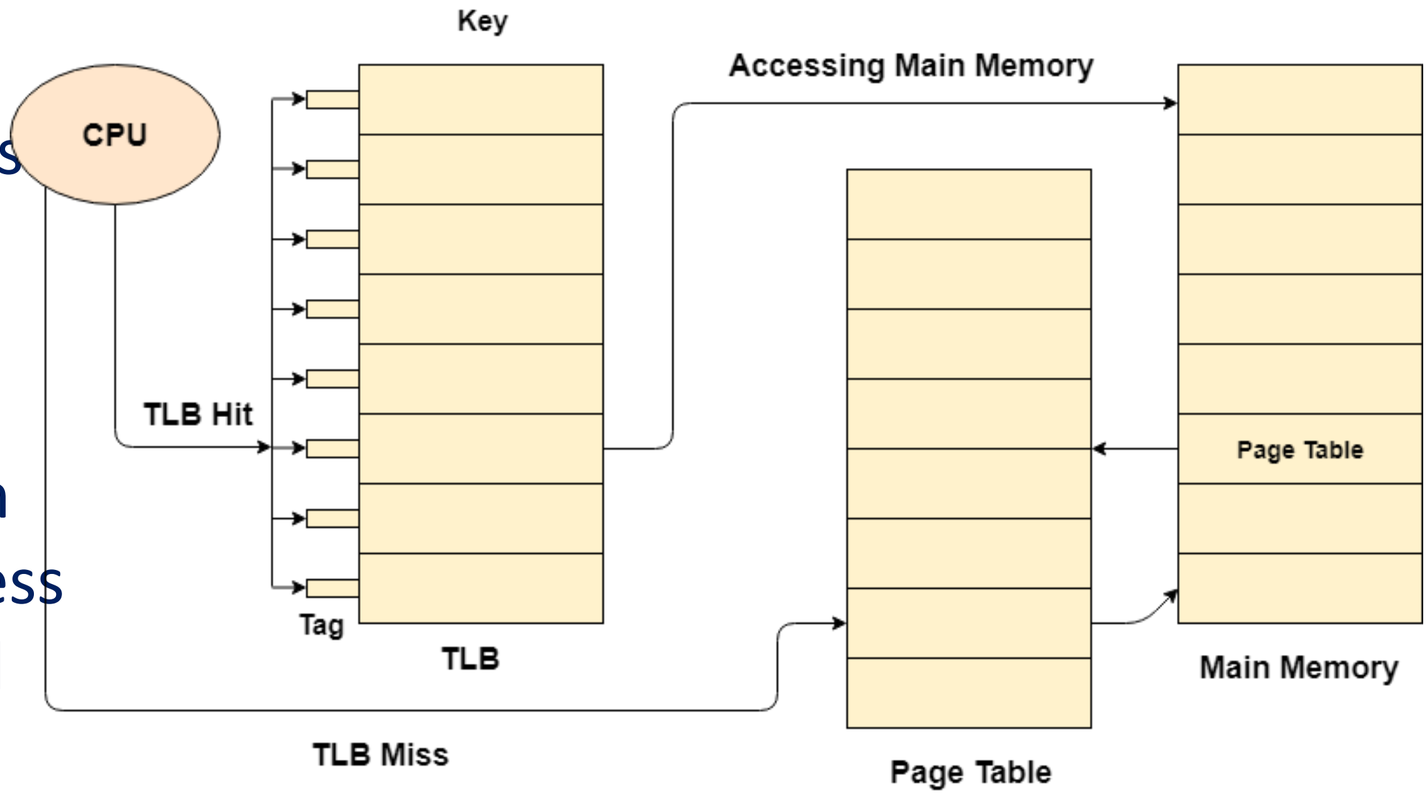
-If a page table entry is not found in the TLB (TLB miss), the page number is used as an index while processing the page table.



TLB (Translation Lookaside Buffer)

TLB hit is a condition where the desired entry is found in TLB. If this happens then the CPU simply access the actual location in the main memory.

However, if the entry is not found in TLB (TLB miss) then CPU has to access page table in the main memory and then access the actual frame in the main memory.



TLB (Translation Lookaside Buffer)

Steps in TLB hit

1. CPU generates a virtual (logical) address.
2. It is checked in TLB (present).
3. The corresponding frame number is retrieved, which now tells where the main memory page lies.

Steps in TLB miss

1. CPU generates a virtual (logical) address.
2. It is checked in TLB (not present).
3. Now the page number is matched to the page table residing in the main memory
4. The corresponding frame number is retrieved, which now tells where the main memory page lies.
5. The TLB is updated with new PTE (if space is not there, one of the replacement techniques comes into the picture i.e. either FIFO, LRU or MFU etc).

TLB (Translation Lookaside Buffer)

If the probability of TLB hit is P% (TLB hit rate) then the probability of TLB miss (TLB miss rate) will be (1-P) %.

Therefore, the effective access time can be defined as;

$EAT = P\ (t + m) + (1 - p)\ (t + k.m + m)$

Where, p → TLB hit rate, t → time taken to access TLB, m → time taken to access main memory k = 1, if the single level paging has been implemented.

TLB (Translation Lookaside Buffer)

Numerical:

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, calculate the effective memory access time (in milliseconds).

Solution:

TLB hit ratio (p) = 0.6

Therefore, TLB miss ratio (1-p) =(1-0.6)= 0.4

Time taken to access TLB (t) = 10 ms

Time taken to access main memory (m) = 80 ms

EAT = P (t + m) + (1 - p) (t + k.m + m)

Effective Access Time (EAT) = 0.6 (10 + 80) + 0.4 (10 +1* 80 + 80)

=90 X 0.6 + 0.4 X 170 = 122 (ms)

Page replacement algorithm

-A page replacement algorithm is needed to decide which page needs to be replaced when a new page comes in.
-Page replacement becomes necessary when a page fault occurs and there are no free page frames in memory.

There are three types of Page Replacement Algorithms.
1. Optimal Page Replacement Algorithm
2. First In First Out Page Replacement Algorithm
3. Least Recently Used (LRU) Page Replacement Algorithm

First In First Out (FIFO)

In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

First In First Out (FIFO)

Example 1: Consider page reference string 7, 0, 1,2,0, 3, 0, 4, 2,3,0,3,1,2,0 with 3-page frames.

Find

a. Number of page faults

b. Number of page hits.

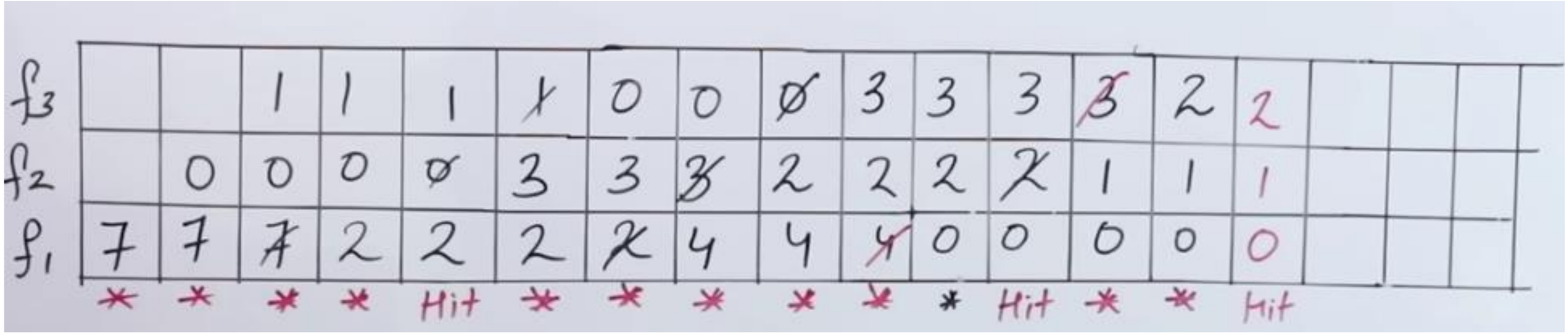
c. Hit ratio

d. Page fault ratio

Hit ratio = number of hit/total references*100

Page fault ratio = number fault/total references*100

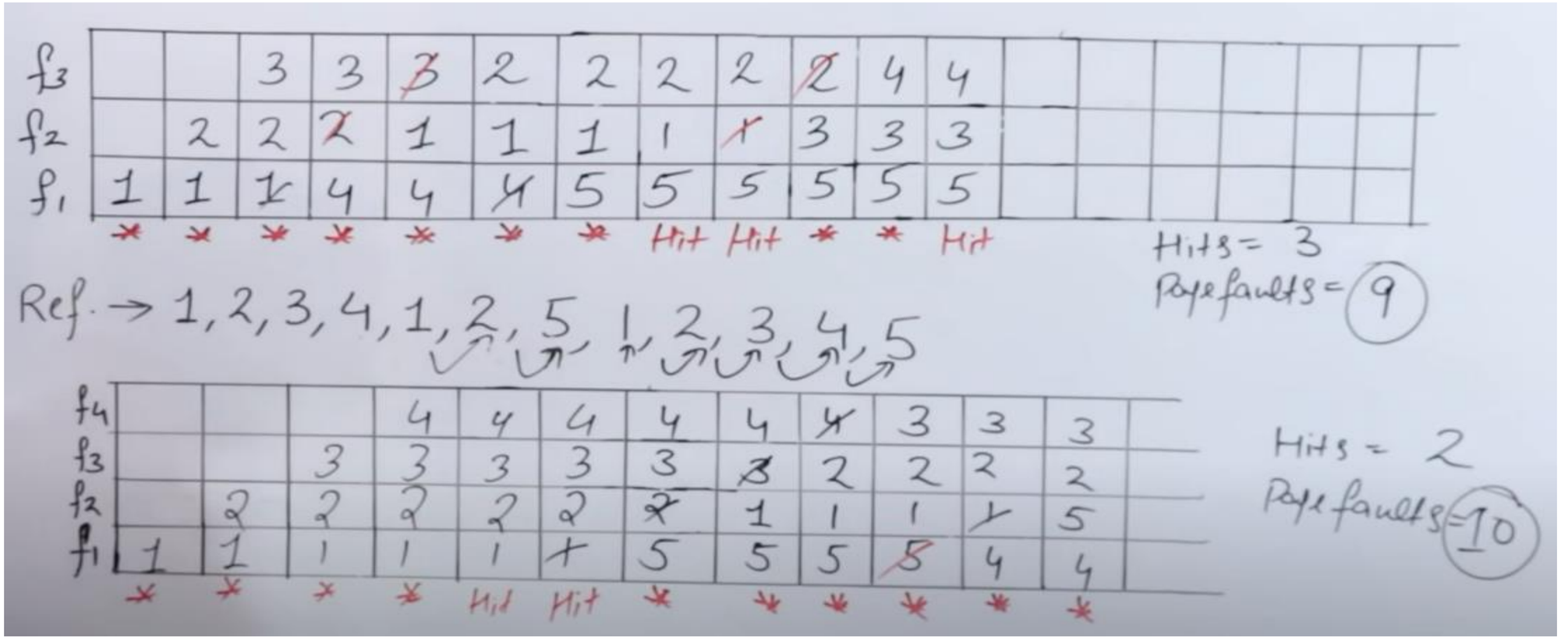
First In First Out (FIFO)



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

a. Number of page faults = 12
b. Number of page hits. = 3
c. Hit ratio = (3/15)*100% = 20%
d. Page fault ratio = (12/15)*100% = 80%

Belady's anomaly: First In First Out (FIFO)



Since we have increased the number of frames for the same reference. But the number of page faults has increased. Which is not expected. This is Blady's anomaly. FIFO is suffering from this algorithm.

Optimal Page Replacement Algorithm

In this algorithm, pages are replaced that would not be used for the longest duration of time in the future.

Key: Which is not used in the longest dimension in the future

Example:

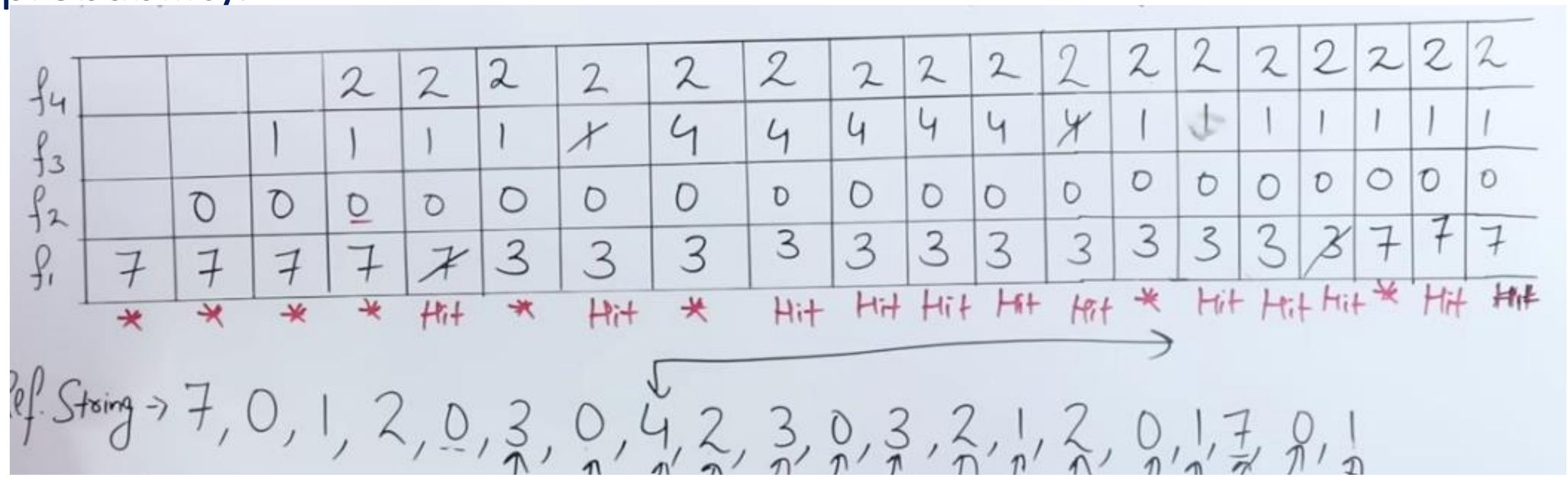
Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1,2,0,1,7,0,1 with 4-page frame. Find

1. Number of page faults,
2. Number of page hits,
3. Page fault rate,
4. Page fault probability.

Optimal Page Replacement Algorithm

Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1,2,0,1,7,0,1 with 4-page frame.

1. Number of page faults,
2. Number of page hits,
3. Page fault rate,
4. Page fault probability.



Least Recently Used (LRU) Algorithm

In this algorithm, the page will be replaced which is least recently used.

Key: page will be replaced which is least recently used.

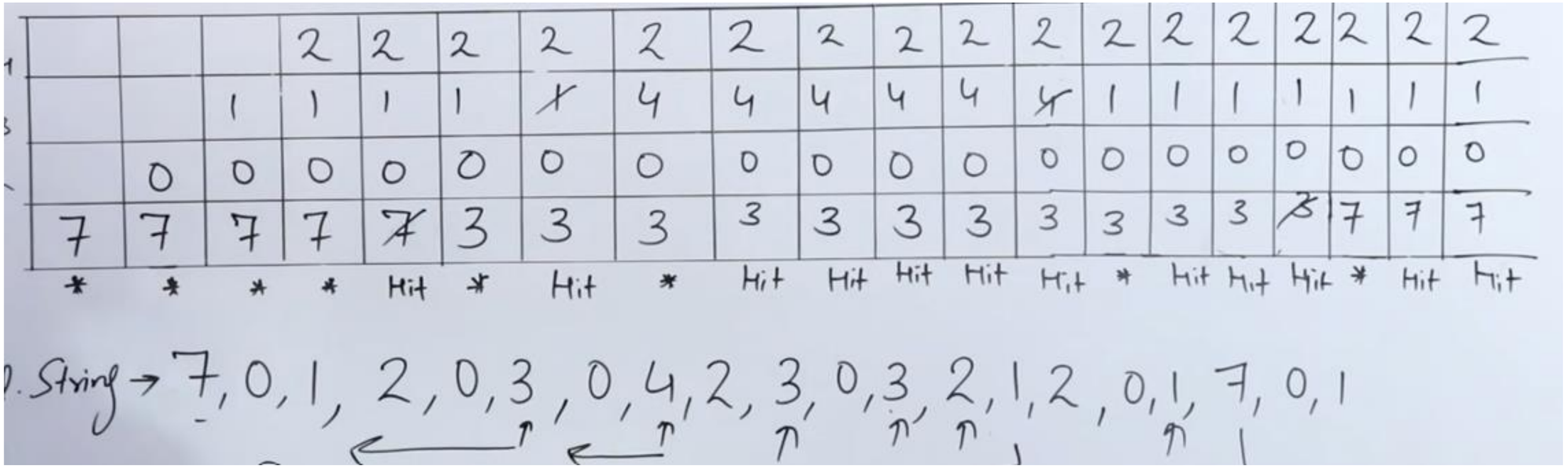
Example-3: Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1,2,0,1,7,0,1 with 4-page frames. Find.

1. Number of page faults,
2. Number of page hits,
3. Page fault rate,
4. Page fault probability.

Least Recently Used (LRU) Algorithm

Example-3: Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1,2,0,1,7,0,1 with 4 page frames.

1. Number of page faults,
2. Number of page hits,
3. Page fault rate,
4. Page fault probability.



Second Change Page Replacement(SCP)

- Second chance Page Replacement Algorithm is a modification of FIFO algorithm.

- In this algorithm, a variable counter called the Reference bit (R) is introduced

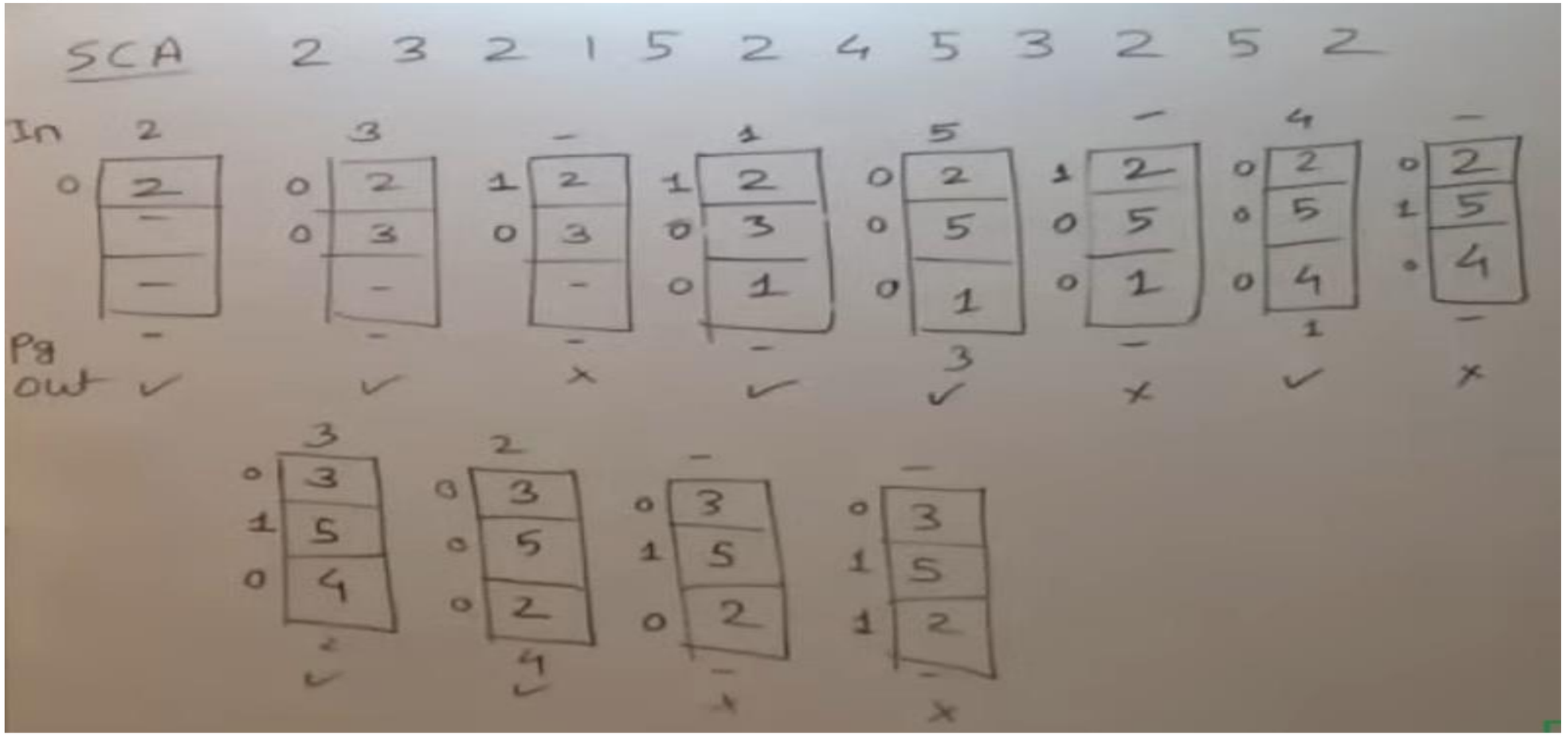
Second Change Page Replacement(SCP)

Consider the reference string:2 3 2 1 5 2 4 5 3 2 5 2. With no. of frames given as 3, find the no. of page faults and the page fault rate.

S.N	• 2	3	2	1	5	2	4	5	3	2	5	2
F3				1	1	1	4	4	4	2	2	2(1)
F2		3	3	3	5	5	5	5(1)	5(1)	5(0)	5(1)	5(1)
F1	2	2	2(1)	2(1)	2(0)	2(1)	2(0)	2(0)	3	3	3	3
Page F/H	F	F	H(sc)	F	F	H(sc)	F	H(sc)	F	F	H(sc)	H(sc)
Total pf= 7 Page Fault Rate = Total Page fault / No. of pages in reference string (7/12)												

Second Change Page Replacement(SCP)

Consider the reference string:2 3 2 1 5 2 4 5 3 2 5 2. With no. of frames given as 3, find the no. of page faults and the page fault rate.

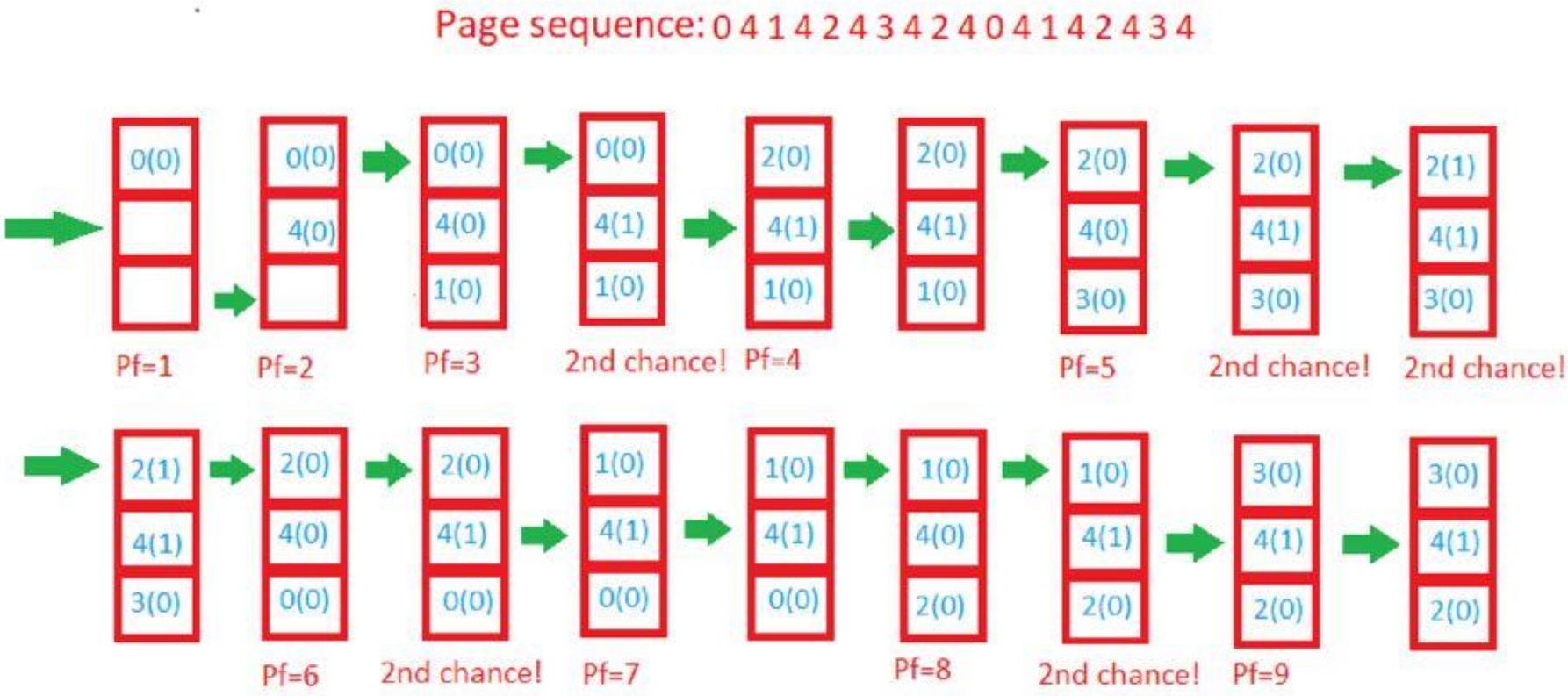


Clock Page Replacement(SCP)

Consider the reference string:1,2,3,4,3,1,2,1,5,1. With no. of frames given as 3, find the no. of page faults and the page fault rate.

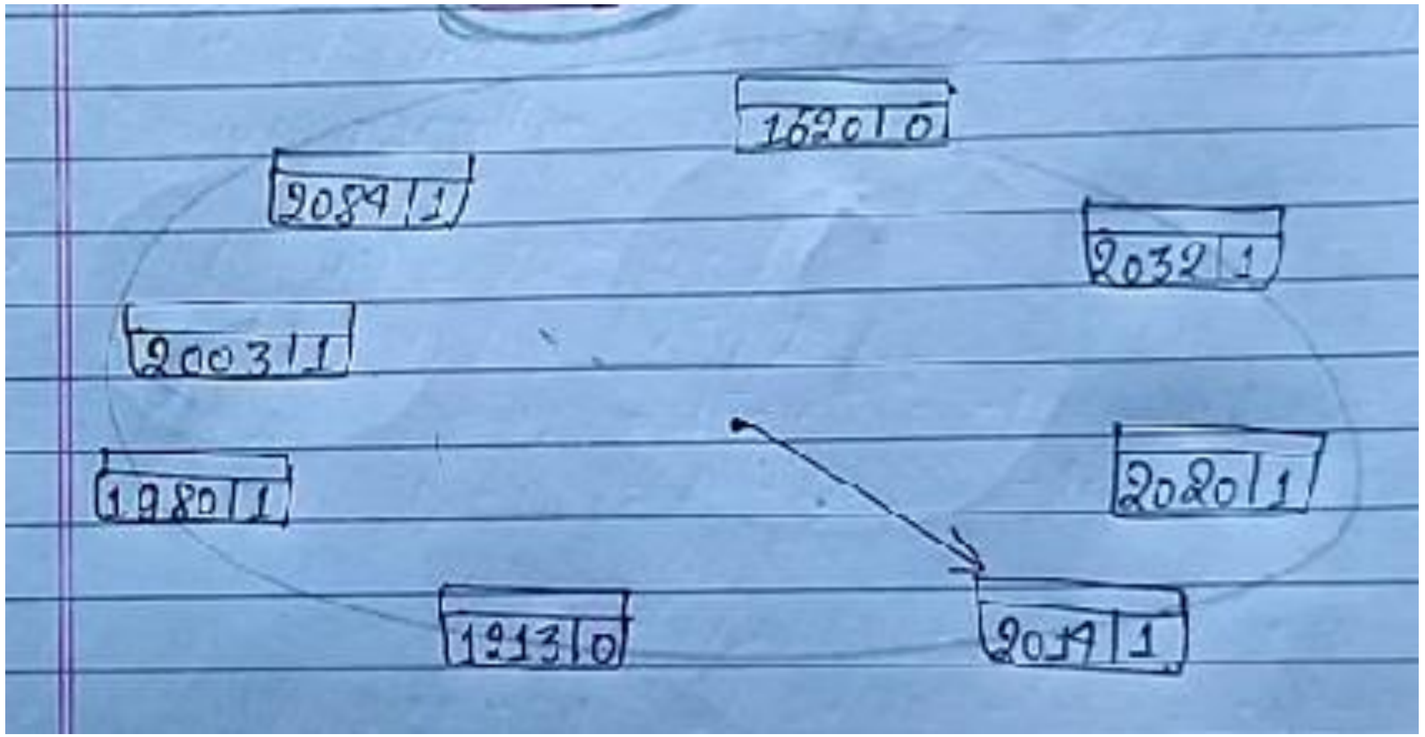
Clock Page Replacement(SCP)

Consider the reference string:0,4,1,4,2,4,3,4,2,4,0,4,1,4,2,4,3,4. With no. of frames given as 3, find the no. of page faults and the page hit rate.



WS Clock Page Replacement(SCP)

Insert 2204 in the appropriate location in the following diagram.



Solution:
we replace 1213 with 2204

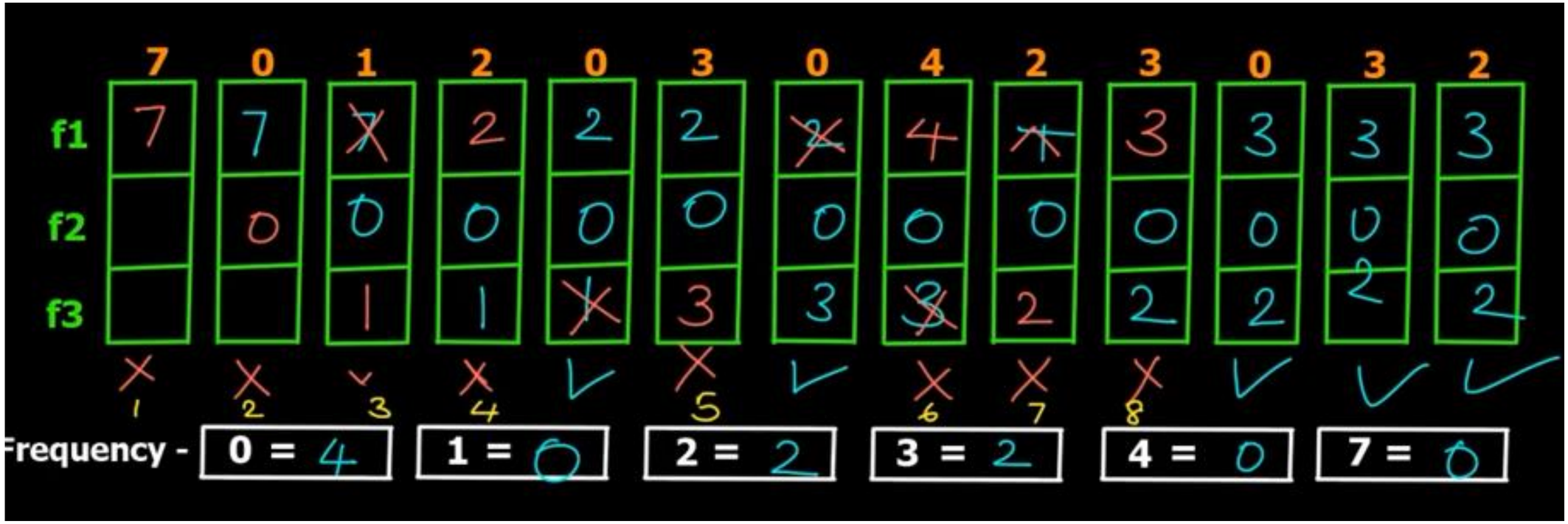
Least Frequently Used (LFU)

The following reference string: 7 0 2 4 3 1 4 7 2 0 4 3 0 3 2 7
Consider the page frame size to be three.
Find the number of page faults using LFU.

String	7	0	2	4	3	1	4	7	2	0	4	3	0	3	2	7
F3			2	2	2	1	1	1	2	2	2	3	3	3	3	3
F2		0	0	0	3	3	3	7	7	0	0	0	0	0	2	7
F1	7	7	7	4	4	4	4	4	4	4	4	4	4	4	4	4
Miss/Hit	M	M	M	M	M	M	H	M	M	M	H	M	H	H	M	M

Least Frequently Used (LFU)

The following reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2
Consider the page frame size to be three.
Find the number of page faults using LFU.



Concept of Locality of Reference

-The locality of reference is the behavior of the processor.
-Process tries to access the same process multiple times within a short period.
-Locality of reference refers to the tendency of the computer program to access the same set of memory locations for a particular time period.

Cache Operation: It is based on the principle of locality of reference. There are two ways with which data or instruction is fetched from main memory and get stored in cache memory.

Concept of Locality of Reference

1. Temporal Locality – Temporal locality means current data or instruction that is being fetched may be needed soon. So we should store that data or instruction in the cache memory so that we can avoid again searching in the main memory for the same data.
(if same address is referred soon)

2. Spatial Locality – Spatial locality means instruction or data near to the current memory location that is being fetched, may be needed soon in the near future. This is slightly different from the temporal locality. Here we are talking about nearly located memory locations while in temporal locality we were talking about the actual memory location that was being fetched.
(if nearby address is referred soon)

Segmentation

Paging is closed to OS, segmentation is closed to user.
Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process.

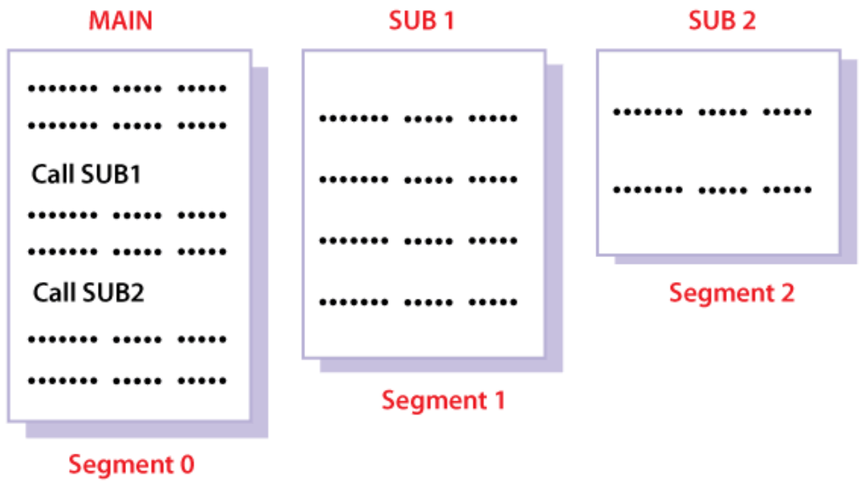
The details about each segment are stored in a table called a segment table. Segment table is stored in one (or many) of the segments.

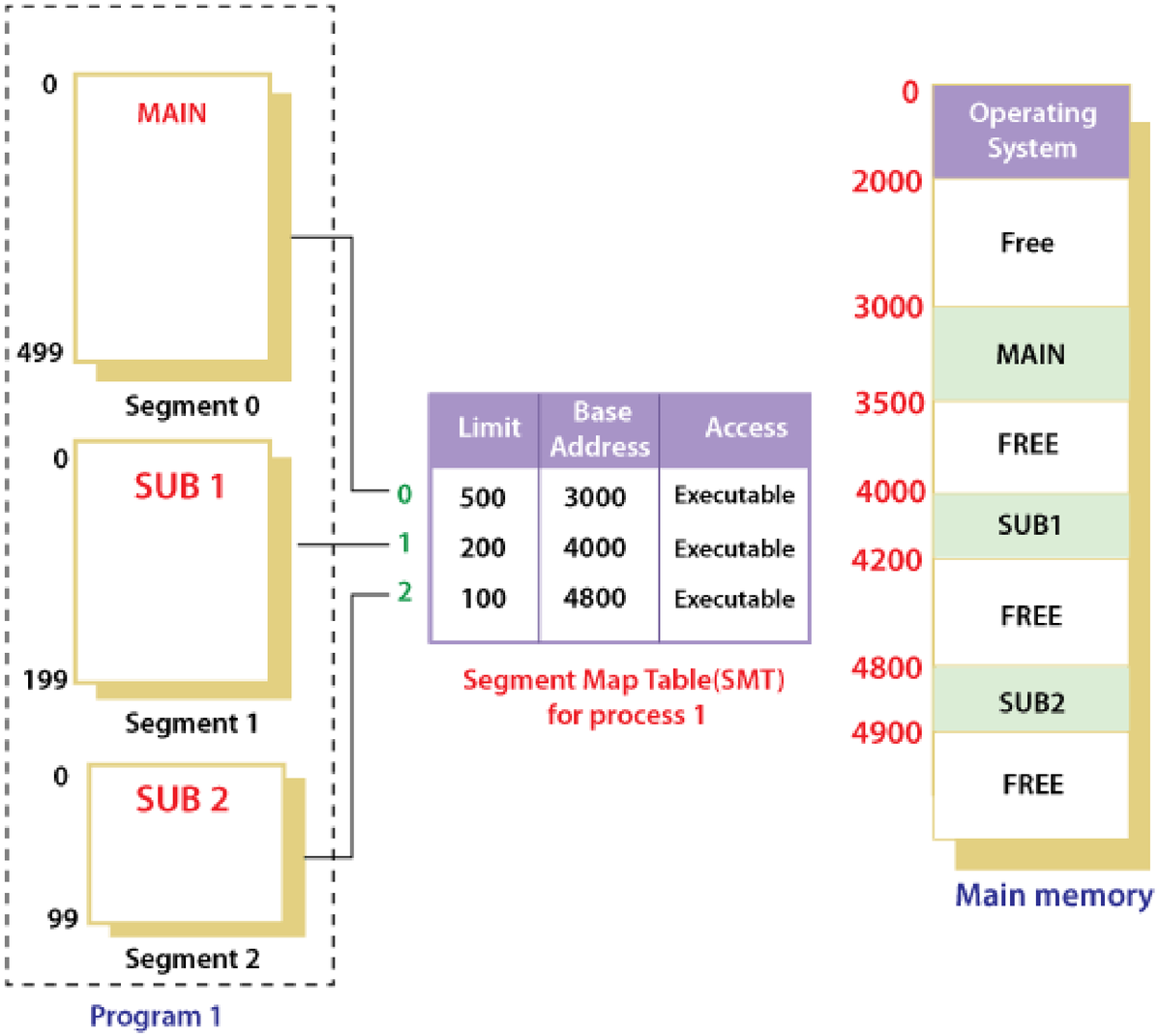
Segment table contains mainly two information about segment:

Base: It is the base address of the segment
Limit: It is the length of the segment.

Segmentation

SS





Segmentation

Advantages of Segmentation

- No internal fragmentation
- Average Segment Size is larger than the actual page size.
- Less overhead
- It is easier to relocate segments than entire address space.
- The segment table is of lesser size as compared to the page table in paging.

Disadvantages

- It can have external fragmentation.
- it is difficult to allocate contiguous memory to variable-sized partition.
- Costly memory management algorithms.

Find me

9851083215

Santosh.it288@mail.com

www.phtechno.com

Kathmandu