3140702
Operating System

Unit – 6
Memory Management

Subject Faculty: Suhag Baldaniya

Disclaimer

- ***It is hereby declared that the production of the said content is meant for non-commercial, scholastic and research purposes only.***

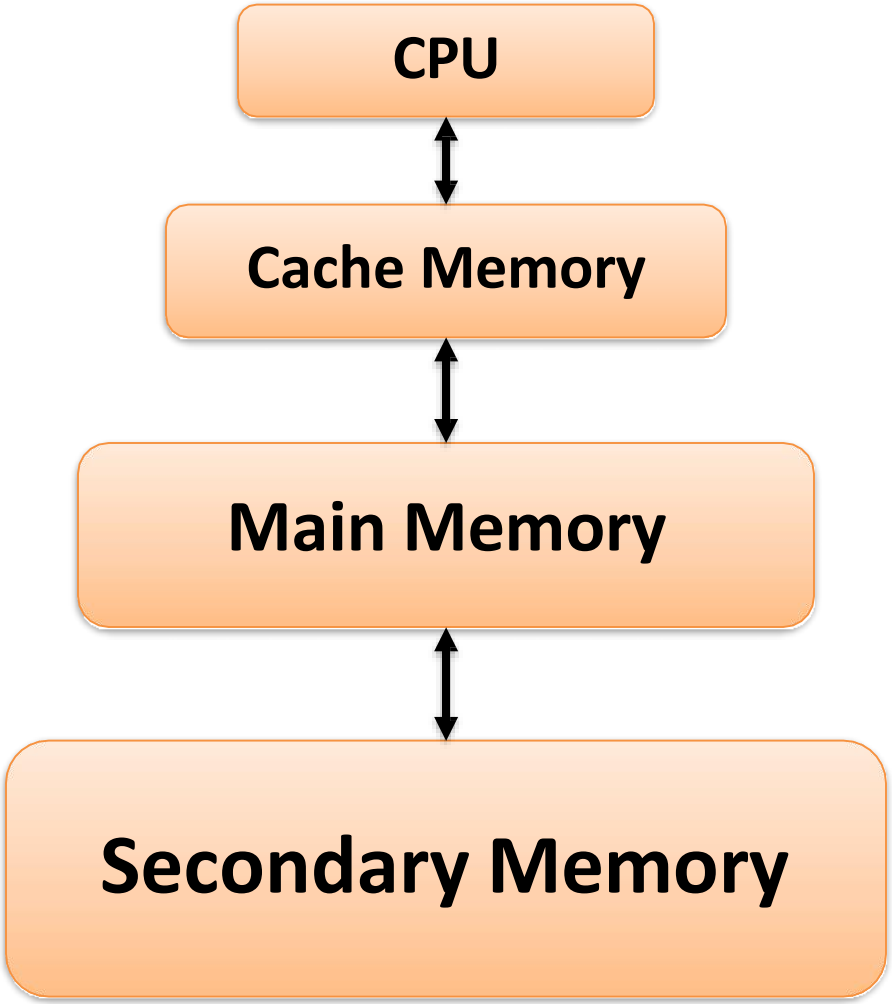
- ***We admit that some of the content or the images provided in this channel's videos may be obtained through the routine Google image searches and few of them may be under copyright protection. Such usage is completely inadvertent.***

- ***It is quite possible that we overlooked to give full scholarly credit to the Copyright Owners. We believe that the non-commercial, only-for-educational use of the material may allow the video in question fall under fair use of such content. However we honor the copyright holder's rights and the video shall be deleted from our channel in case of any such claim received by us or reported to us.***

Topics to be covered

- Basics of Memory Management
- Memory partitioning: Fixed and Variable Size Partitioning
- Memory Allocation Strategies (First Fit, Best Fit, and Worst Fit)
- Swapping and Fragmentation
- Paging and Demand Paging
- Segmentation
- Concepts of Virtual Memory
- Page Replacement Policies (FIFO, LRU, Optimal, Other Strategies)
- Thrashing

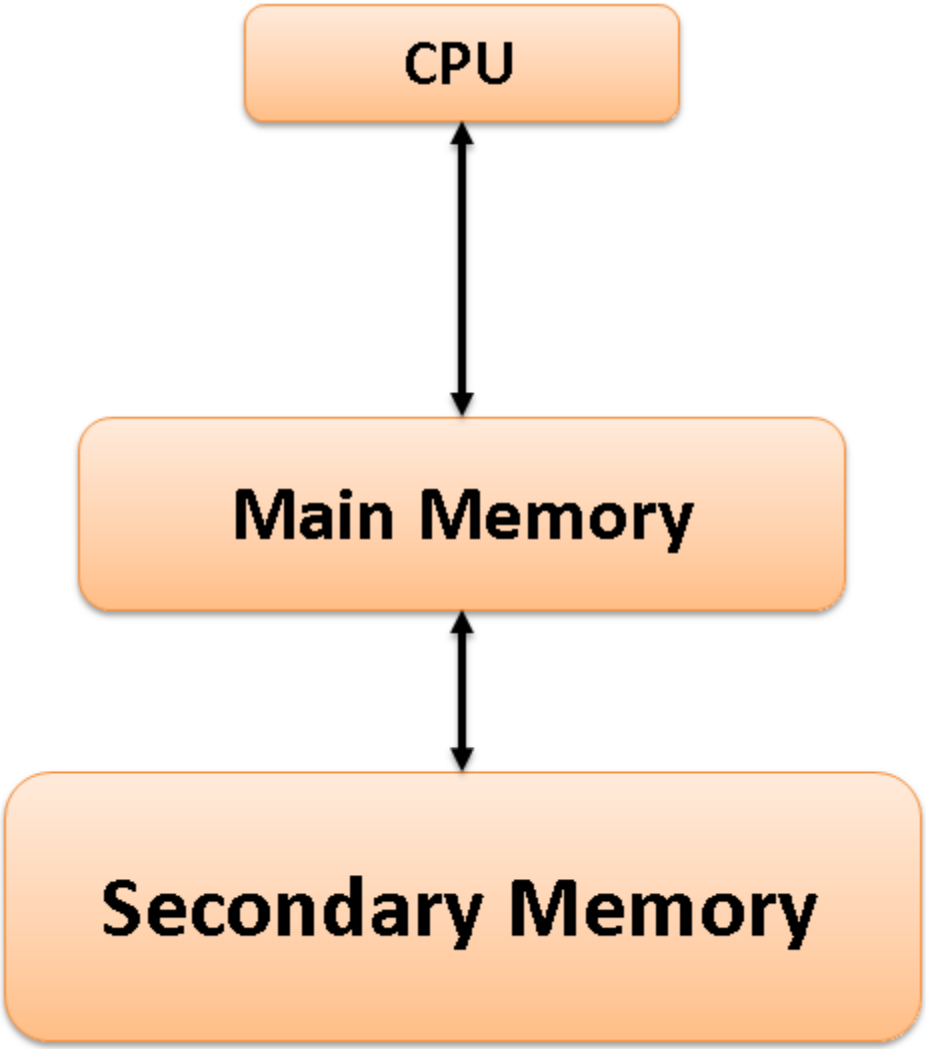
Hierarchy of Memory

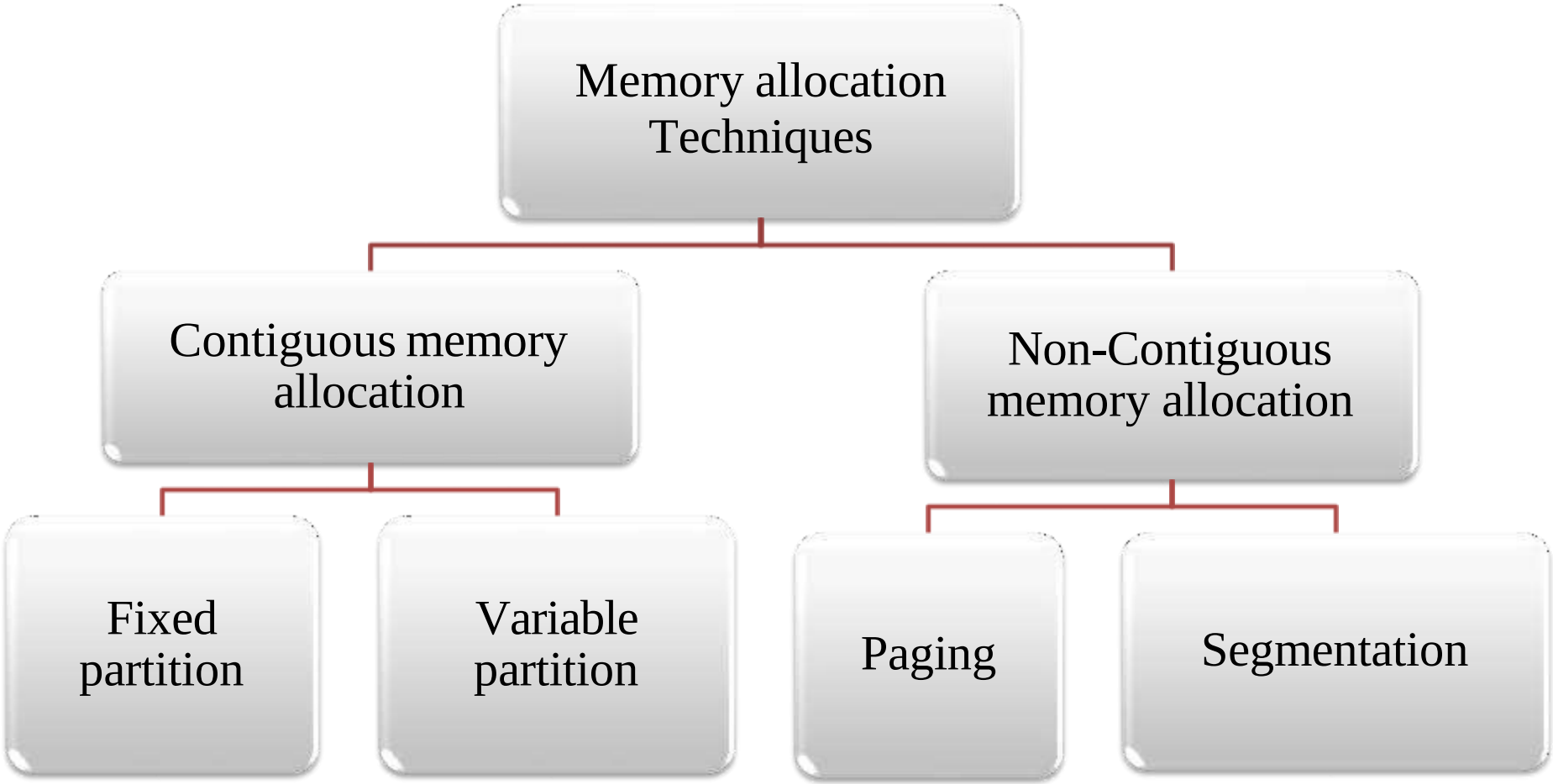


Hierarchy of Memory

Now, here the OS have two important responsibilities:

1. **Space Allocation:** Now OS will decide which process from the SM will get which area in MM.
2. **Address Translation** i.e. from logical address to physical address.



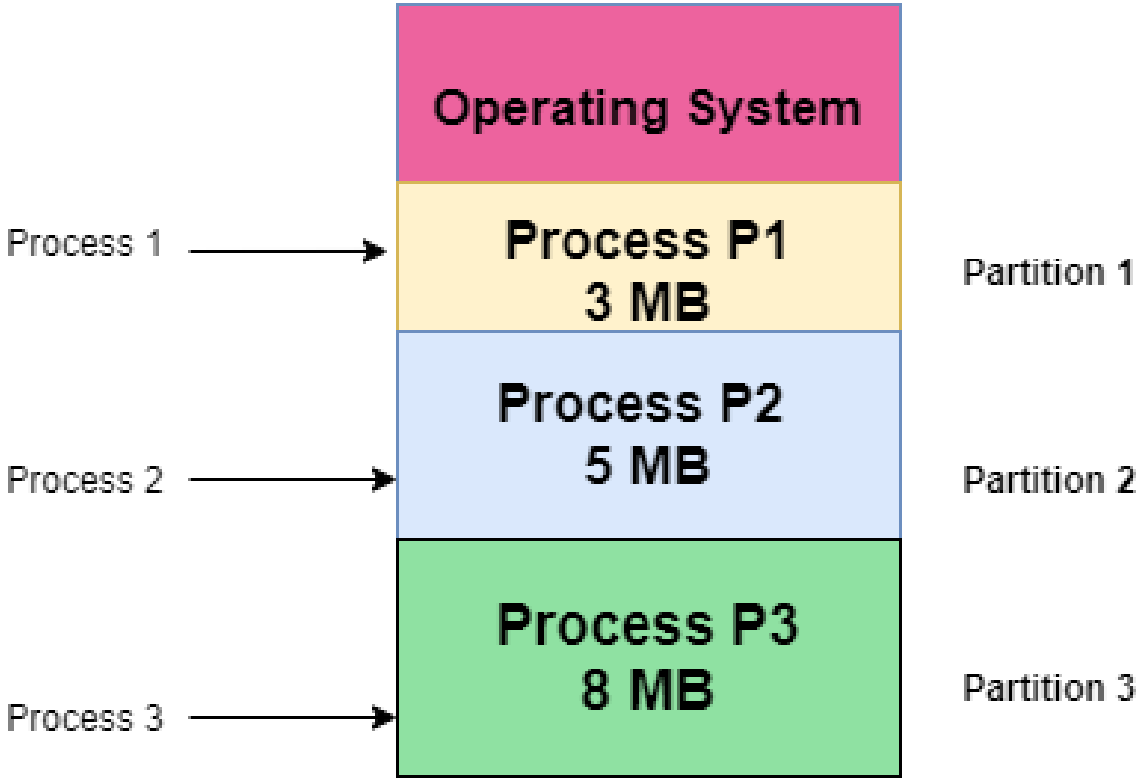

Memory allocation Techniques
Contiguous memory allocation
Non-Contiguous memory allocation
Fixed partition
Variable partition
Paging
Segmentation

Memory Allocation Techniques

Marwadi University

Contiguous Memory allocation

- Simple and old method.
- Here each process occupies contiguous block of main memory.
- When process is brought in memory, a memory is searched to find out a chunk of free memory having enough size to hold a process.



Size of Partition =Size of Process

Contiguous M/m allocation: Fixed Partition

Marwadi University

Multi programming with fixed partition

- Numbers of **partitions are fixed.**
- Here, memory is divided **into fixed size partition.**
- **Each partition** may contain **exactly one process**.
- Size of each partition is **not requires to be same.**
- When a partition is free, process is selected from the input queue and it is loaded into free partition.

Contiguous M/m allocation: Fixed Partition

Marwadi University

Multi programming with fixed partition:

- **Advantage:**
  - Implementation is simple.
  - Processing overhead is low.
- **Disadvantage:**
  - Limit in **process size**.
  - **Degree of multiprogramming** is also limited.
  - Causes **External fragmentation** because of contiguous memory allocation.
  - Causes **Internal fragmentation** due to fixed partition of memory.

Contiguous M/m allocation: Variable Size Partition

Marwadi University

Multi programming with variable/dynamic partition

- Here memory **is not divided into fixed partition**, also the number of partition is not fixed.
- Only required memory is allocated to process at runtime.
- Whenever any process enter in a system, a chunk of memory big enough to fit the process is found and allocated. And the remaining unoccupied space is treated as another free partition.
- When process get terminated it releases and the space occupied and it that free partition is contiguous to another free partition then that both free partition can be merge.

Contiguous M/m allocation: Variable Size Partition

Marwadi University

Multi programming with variable/dynamic partition:

- **Advantage:**
  - No internal fragmentation.
  - No limitation on number of processes.
  - No limitation on process size.
- **Disadvantage:**
  - Causes **External fragmentation:**
    - Memory is allocated when process enters into system, and deallocated when terminates. This operation may leads to small holes in the memory.
    - This holes will be so small that no process can be loaded in it..
    - But total size of all holes may be big enough to hold any process.

Difference

Parameter	Fixed partitioning	Variable partitioning
Division	Main memory is divided into fixed sized partitions.	Main memory is not divided into fixed sized partitions.
Implementation	It is easy to implement.	It is less easy to implement.
Process	Only one process can be placed in a partition.	In variable partitioning, the process is allocated a chunk of free memory.
Utilize	It does not utilize the main memory effectively.	It utilizes the main memory effectively.
Fragmentation	There is presence of internal fragmentation and external fragmentation.	There is external fragmentation.
Degree	Degree of multi-programming is less.	Degree of multi-programming is higher.
Limitation	There is limitation on size of process.	There is no limitation on size of process.

Memory Allocation Strategies

Marwadi University

- In **Partition Allocation**, when there is **more than one partition freely available** to accommodate a process's request, **a partition must be selected.**
- To **choose a particular partition**, a partition allocation method is needed. A partition allocation method is **considered better** if it **avoids internal fragmentation.**
- There are different Memory allocation Algorithm are:

1. First Fit
2. Best Fit
3. Worst Fit

Memory Allocation Strategies

Marwadi University

1. First Fit:

- In the first fit, the **partition is allocated** which is **the first sufficient block from the top** of Main Memory.
- It scans **memory from the beginning** and **choose the first available block that is large enough.** Thus it allocates the first partition that is large enough.

Memory Allocation Strategies

Marwadi University

1. First Fit:

Process	Memory Block
P1-90	20
P2-50	100
P3-30	40
P4-40	200
	10

Memory Allocation Strategies

Marwadi University

2. Best Fit:

- Allocate the process to the partition which is the **first smallest sufficient partition** among the free available partition.
- It searches the entire list of partition to find the **smallest partition whose size is greater than or equal** to the **size of the process**.

Memory Allocation Strategies

Marwadi University

2. Best Fit:

Process	Memory Block
P1-40	100
P2-10	50
P3-30	30
P4-60	120
	35

Memory Allocation Strategies

Marwadi University

3. Worst Fit:

- Allocate the process to the partition which is the **largest sufficient among the freely available partitions** available in the main memory.
- It is opposite to the best-fit algorithm.
- It searches the entire list of partitions to **find the largest partition** and allocate it to process.

Memory Allocation Strategies

Marwadi University

3. Worst Fit:

Process	Memory Block
P1-40	100
P2-10	50
P3-30	30
P4-60	120
	35

Memory Allocation Strategies

Marwadi University

Consider five memory partitions of size 100 KB, 500 KB, 200 KB, 450 KB and 600 KB in same order. If sequence of requests for blocks of size 212 KB, 417 KB, 112 KB and 426 KB in same order come, then which of the following algorithm makes the efficient use of memory?

A.Best fit algorithm

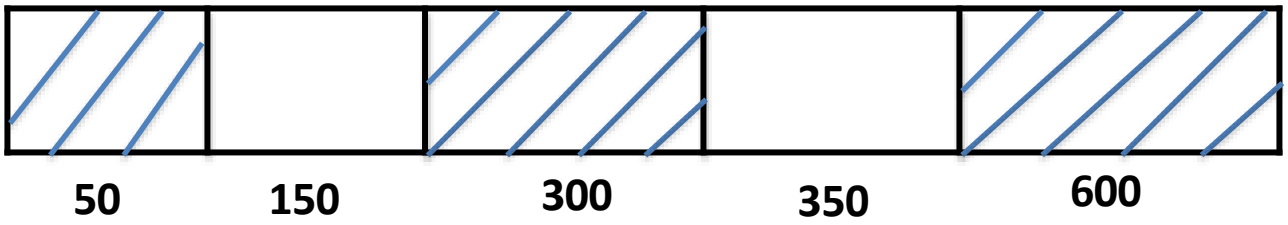
B.First fit algorithm

C.Worst fit algorithm

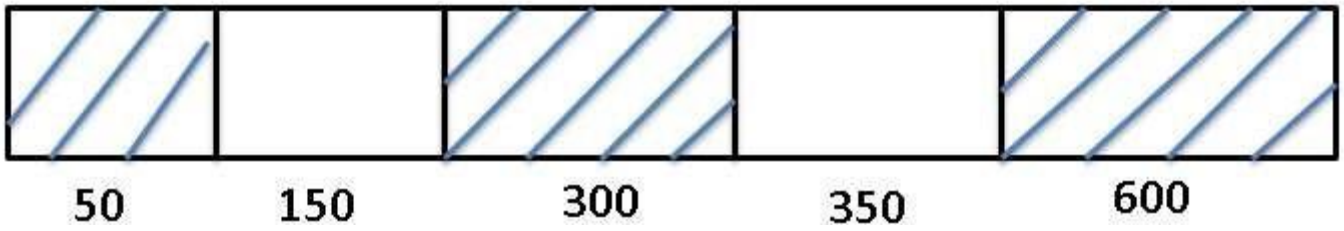
D.Both worst fit and best fit results in same

Variable Size Partitioning

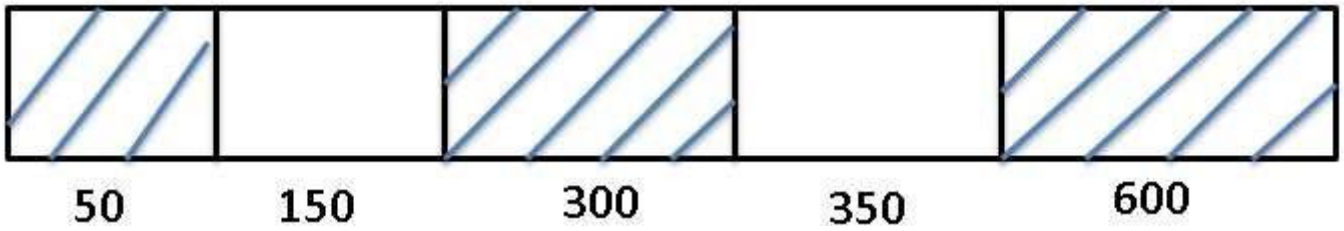
- P1=300
- P2=25
- P3=125
- P4=50



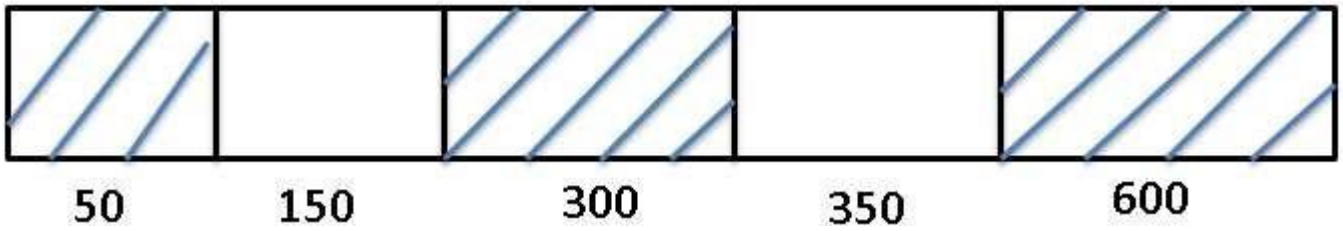
First Fit



Best Fit



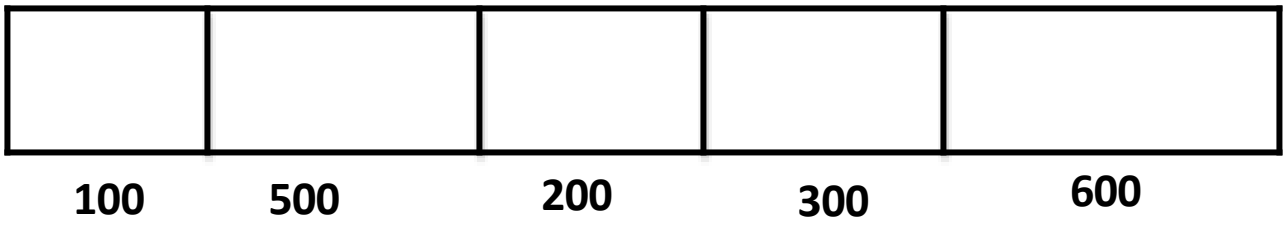
Worst Fit



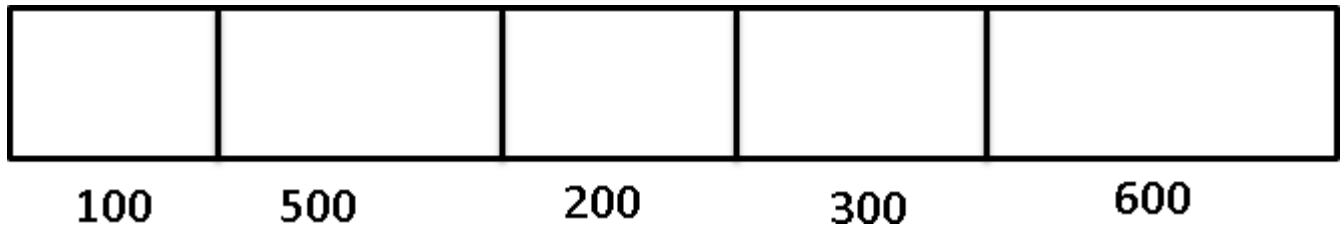
Fixed Size Partitioning

- P1=210K
- P2=400K
- P3=110K
- P4=450K

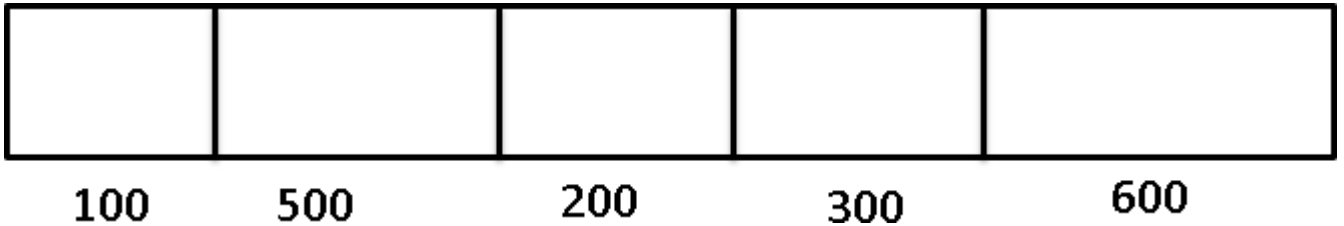
First Fit



Best Fit

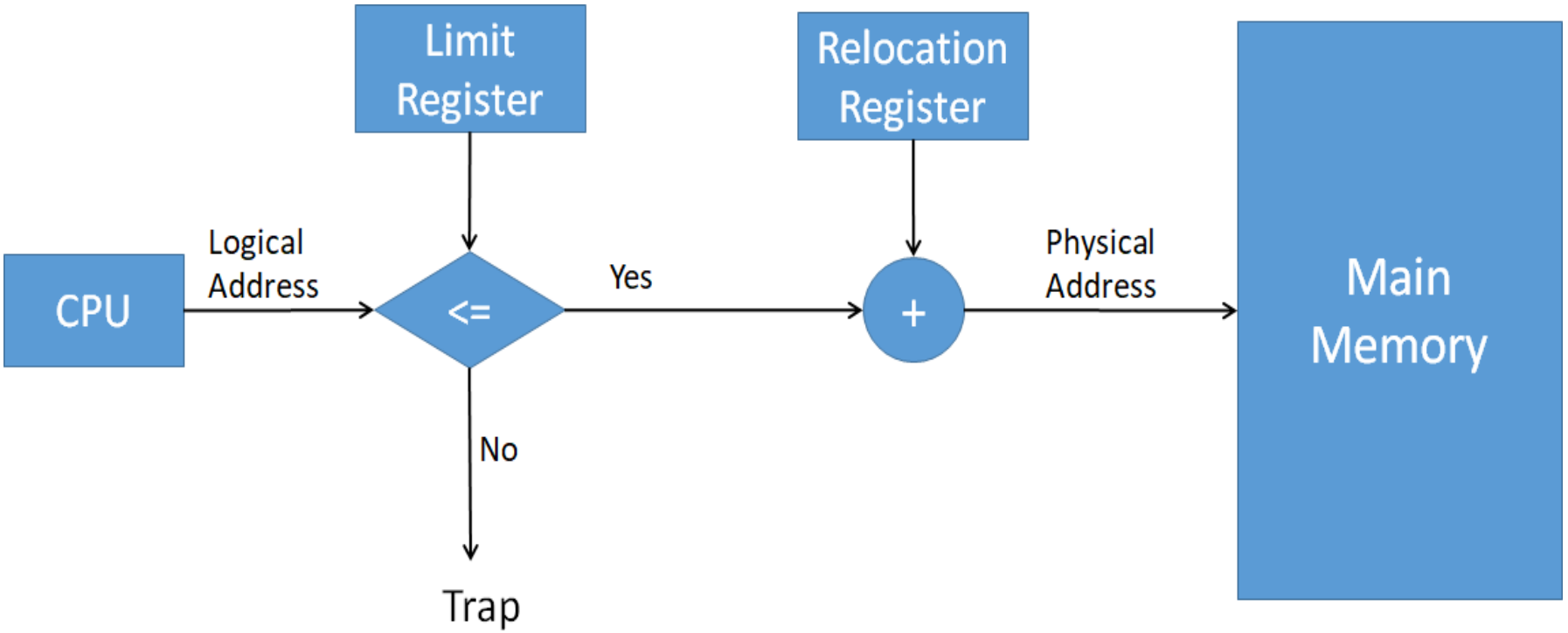


Worst Fit

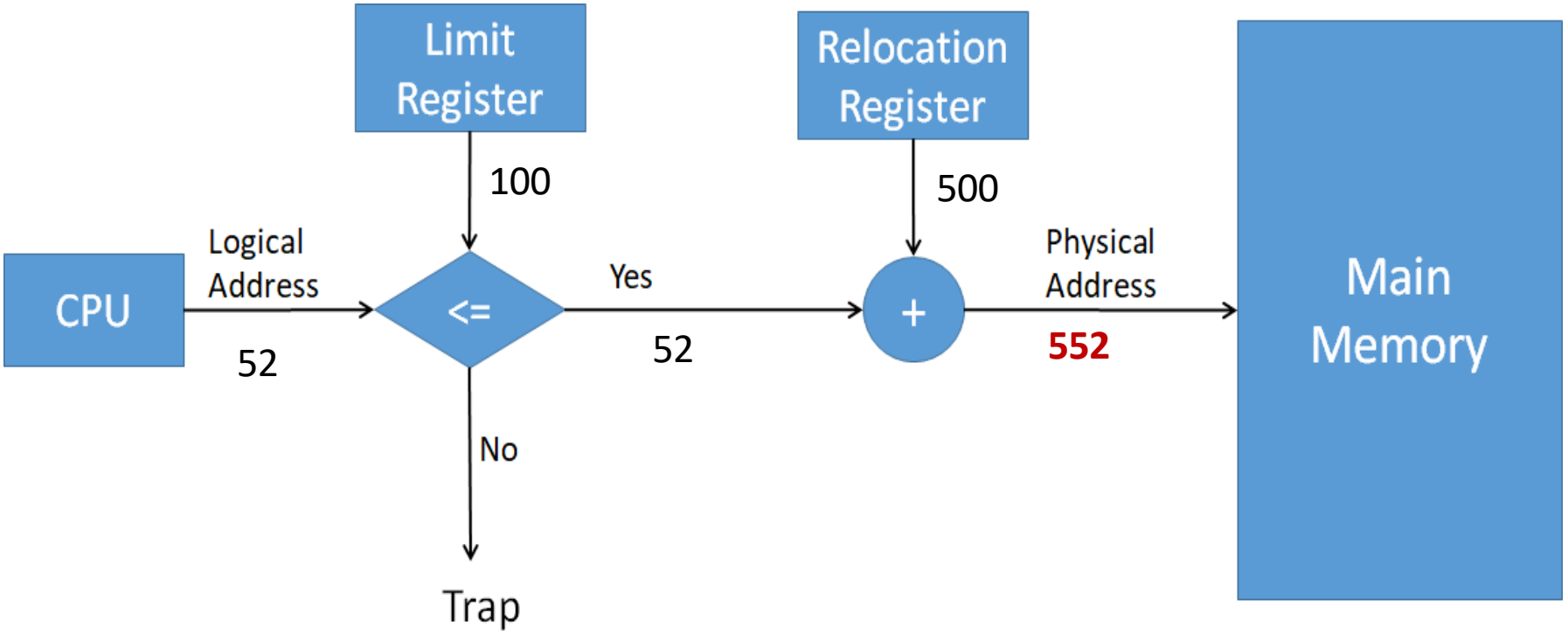


Logical to Physical Address Mapping

- **Relocation Register :** Special purpose register in CPU which holds the base address of the process in main m/m.
- **Limit register:** The size of a process is stored in this register.



Logical to Physical Address Mapping



For e.g. Values:
Limit Register=100
Relocation Register = 500

Non-Contiguous M/m Allocation

Marwadi University

- Non-Contiguous memory allocation techniques are basically of two types:

1. **Paging**
2. **Segmentation**

- The main disadvantage of Dynamic Partitioning is **External fragmentation.**
- Although, this can be **removed by Compaction** but as we have discussed earlier, the compaction makes the system inefficient.
- That's why we come up with an idea of non-contiguous memory allocation.

Paging – Basic Idea and its Need

Marwadi University

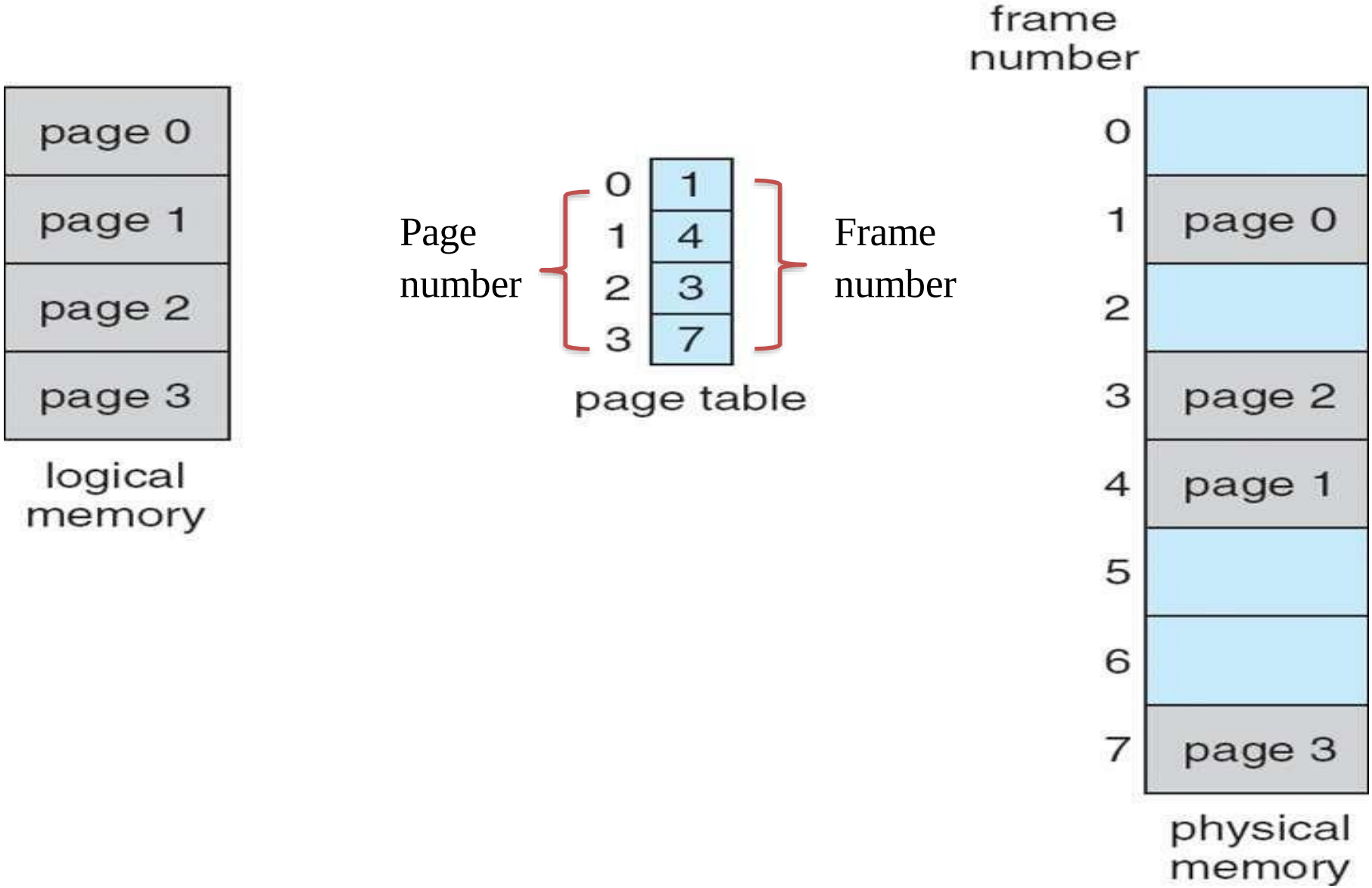
- **Physical memory (Main Memory)**is divided into fixed sized block called **frames.**
- **Logical address space (Secondary Memory)** is divided into blocks of fixed size called **pages.**
- **Page and frame will be of same size.**
- Whenever a process needs to get execute on CPU, its pages are moved from **hard disk** to available **frame in main memory.**

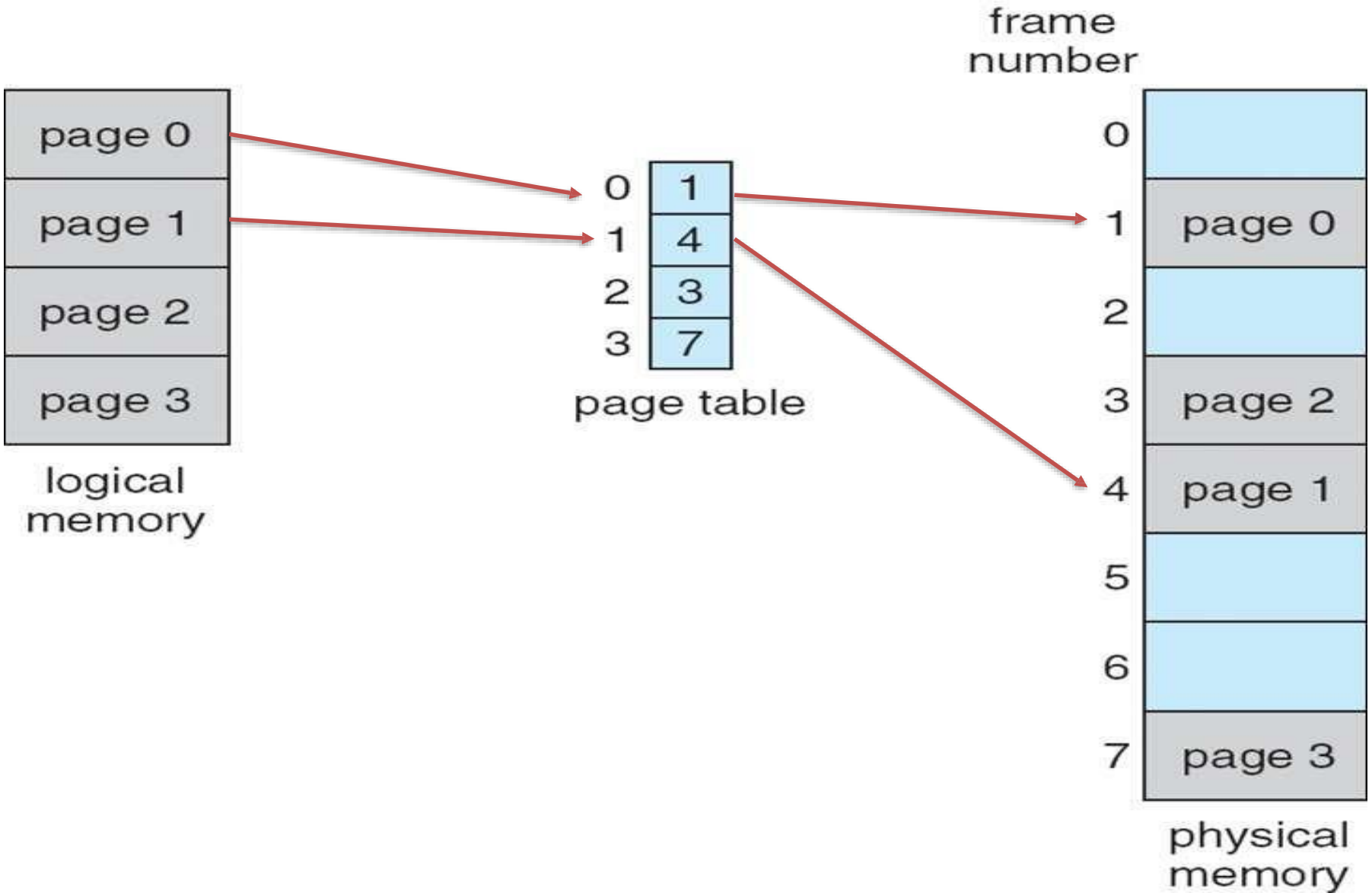
Paging

Thus here **memory management task** is:

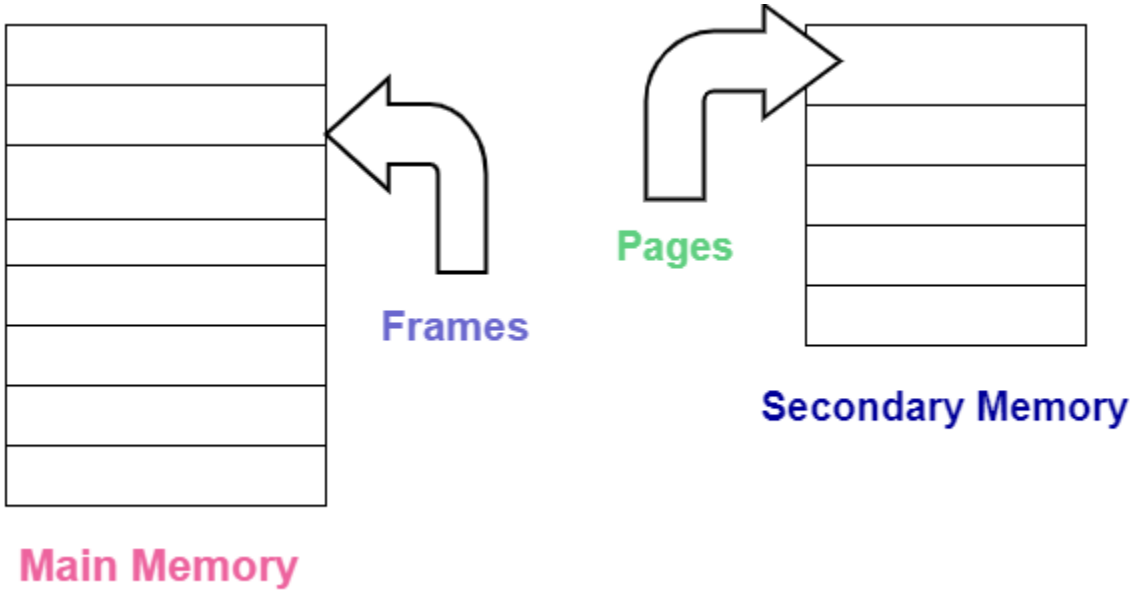
- ✓ to **find free frame** in main memory,
- ✓ **allocate appropriate frame** to the page,
- ✓ **keeping track** of which **page belong** to **which frame.**

- OS maintains a **table called page table,** for each process.
- Page table is index by page number and stores the information about frame number.

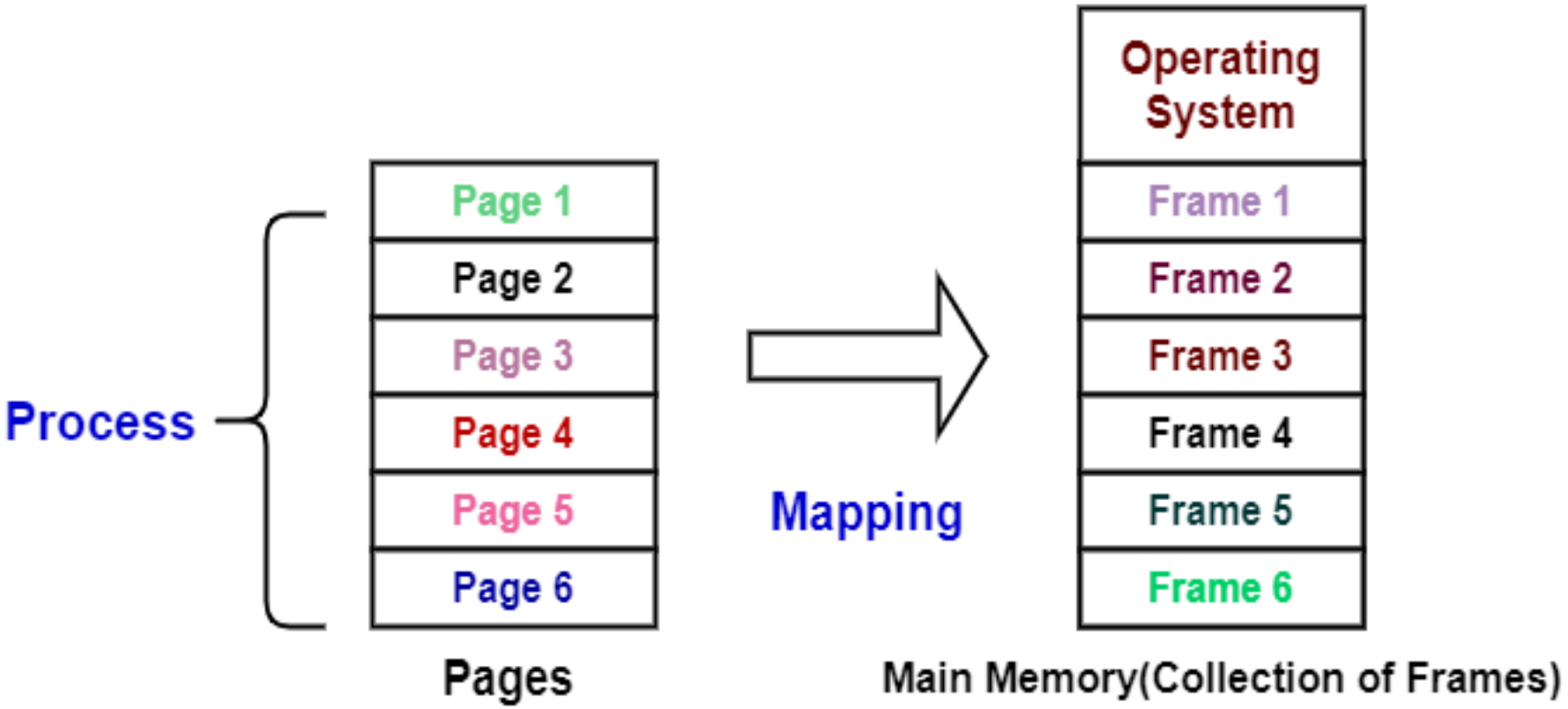

frame number
page 0
page 1
page 2
page 3
logical memory
Page number
0 1
1 4
2 3
3 7
Frame number
page table
0
1 page 0
2
3 page 2
4 page 1
5
6
7 page 3
physical memory


frame
number
page 0
page 1
page 2
page 3
logical
memory
0
1
2
3
1
4
3
7
page table
0
1
2
3
4
5
6
7
page 0
page 2
page 1
page 3
physical
memory

- The paging technique divides the physical memory(main memory) into fixed-size blocks that are known as **Frames** and also divide the **logical memory(secondary memory) into blocks of the same size** that are known as **Pages.**
- This technique keeps the track of all the free frames.
- The Frame has the same size as that of a Page.
- A frame is basically a place where a (logical) page can be (physically) placed.



- Pages of a process are brought into the main memory only when there is a requirement otherwise they reside in the secondary storage.
- One page of a process is mainly stored in one of the frames of the memory. Also, the pages can be stored at different locations of the memory but always the main priority is to find contiguous frames.


Process
Page 1
Page 2
Page 3
Page 4
Page 5
Page 6
Pages
Mapping
Operating System
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Main Memory(Collection of Frames)

Non-Contiguous M/m Allocation

Marwadi University

Page allocation in Paging system

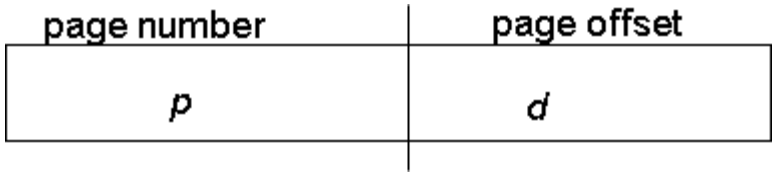
- Here one *free frame list* is maintain.
- When a process arrives in the system to be executed, size of process is expressed in terms of number of pages.
- Each page of the process needs one frame.
- Thus if process requires n pages then n frames must be free in memory.

Before allocation

After allocation

- Translation of Logical Address into Physical Address

- The CPU always generates a logical address.
- In order to access the main memory always a physical address is needed.
- The **logical address generated by CPU always consists of two parts:**
- Page Number(p)
- Page Offset (d)



- **Page Number** is used to specify the specific page of the process from which the CPU wants to read the data and it is also used as an index to the page table.

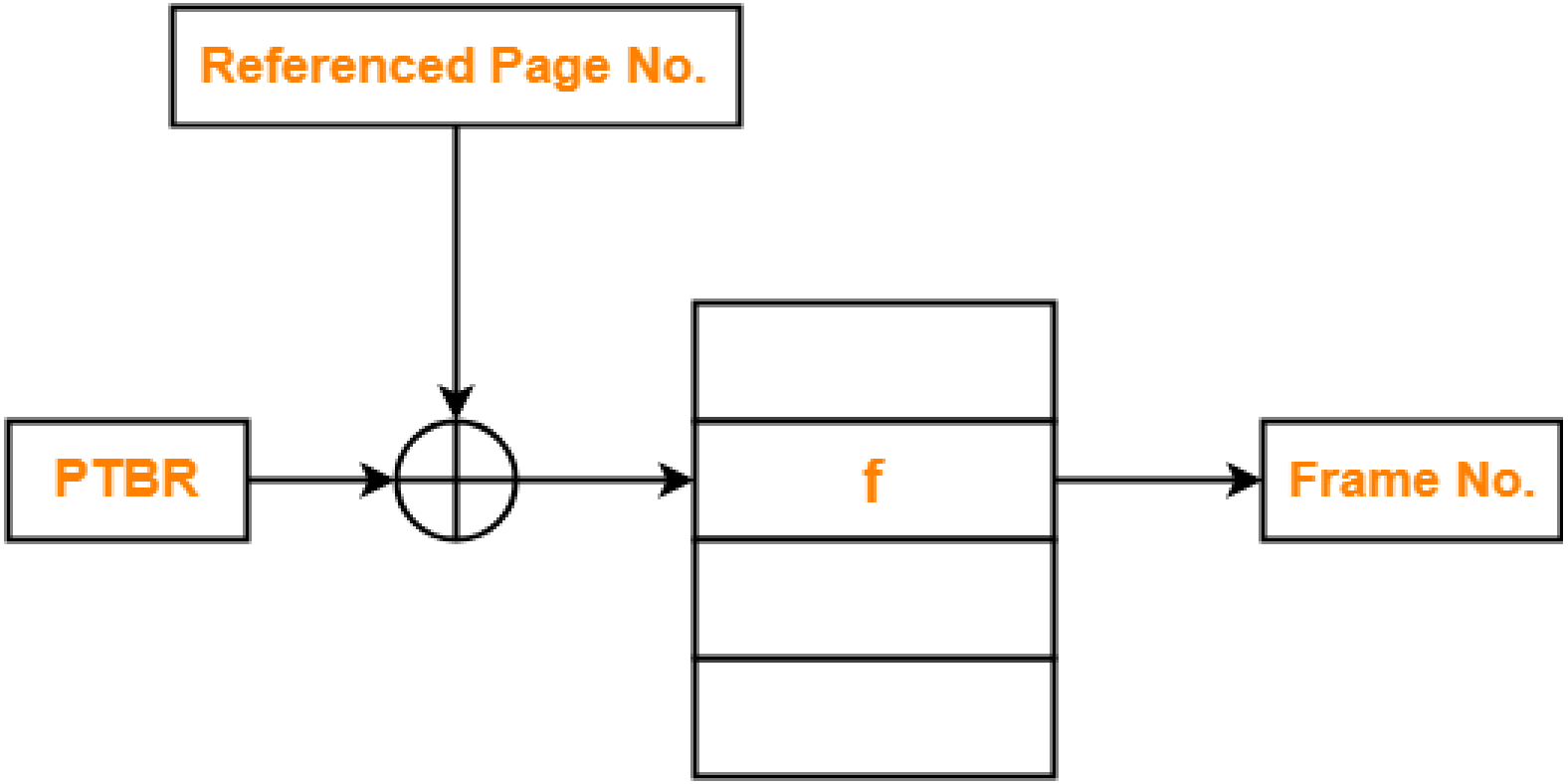
- **Page offset** is mainly used to specify the specific word on the page that the CPU wants to read.

Page Table-

- Page table is a data structure.
- It maps the page number referenced by the CPU to the frame number where that page is stored.

Characteristics-

- Page table is stored in the main memory.
- Number of entries in a page table = Number of pages in which the process is divided.
- Page Table Base Register (PTBR) contains the base address of page table.
- Each process has its own independent page table.

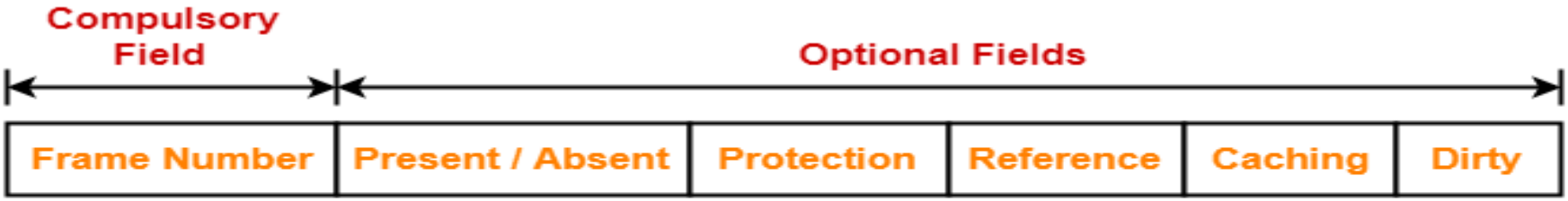


Obtaining Frame Number Using Page Table

- Page Table Base Register (PTBR) provides the base address of the page table.
- The base address of the page table is added with the page number referenced by the CPU.
- It gives the entry of the page table containing the frame number where the referenced page is stored.

Page Table Entry-

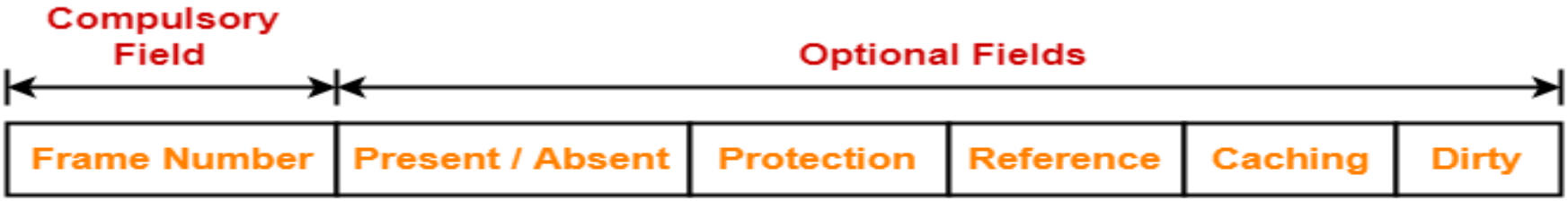
- A page table entry contains several information about the page.
- The information contained in the page table entry varies from operating system to operating system.
- The most important information in a page table entry is frame number.



Page Table Entry Format

1. Frame Number-

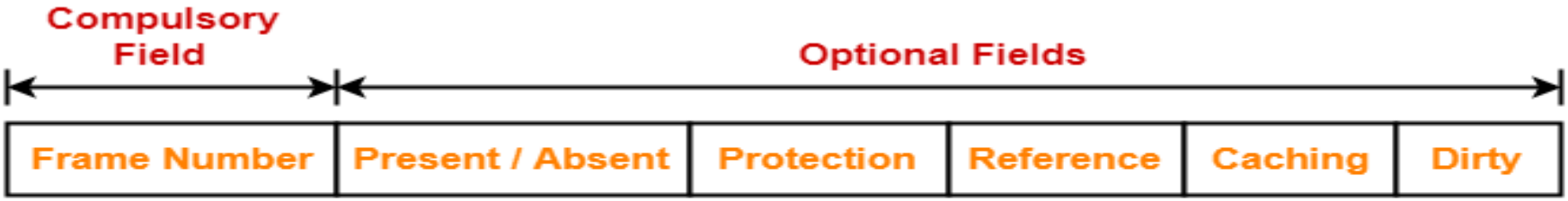
- Frame number specifies the frame where the page is stored in the main memory.
- The number of bits in frame number depends on the number of frames in the main memory.



Page Table Entry Format

2. Present / Absent Bit-

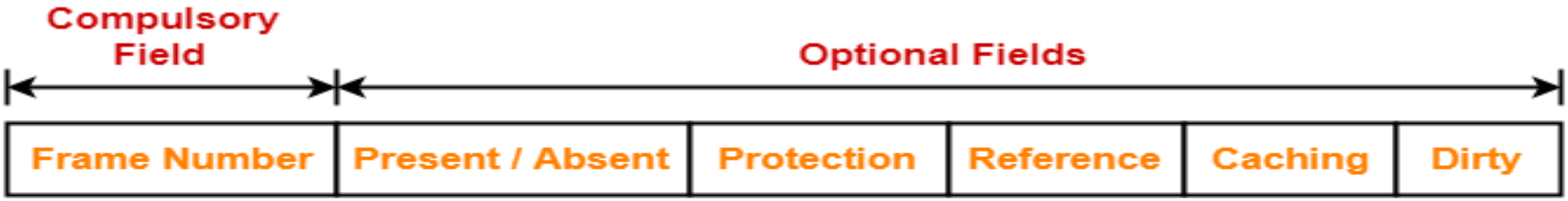
- This bit is also sometimes called as **valid / invalid bit**.
- This bit specifies whether that page is present in the main memory or not.
- If the page is not present in the main memory, then this bit is set to 0 otherwise set to 1.



Page Table Entry Format

3. Protection Bit-

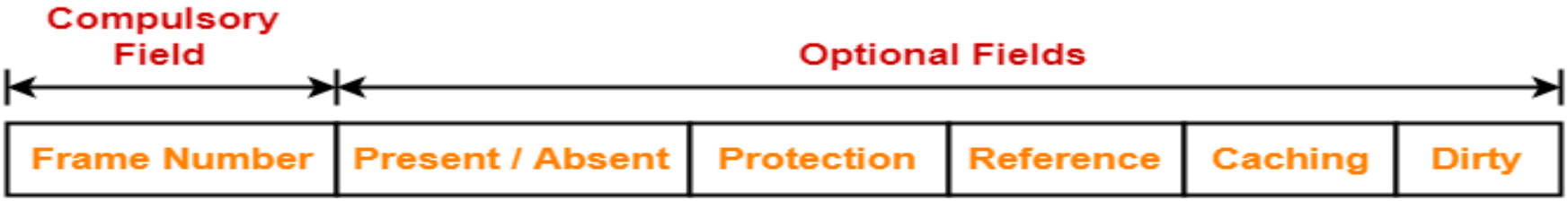
- This bit is also sometimes called as "**Read / Write bit**".
- It specifies the permission to perform read and write operation on the page.
- If only read operation is allowed to be performed and no writing is allowed, then this bit is set to 0.
- If both read and write operation are allowed to be performed, then this bit is set to 1.



Page Table Entry Format

4. Reference Bit-

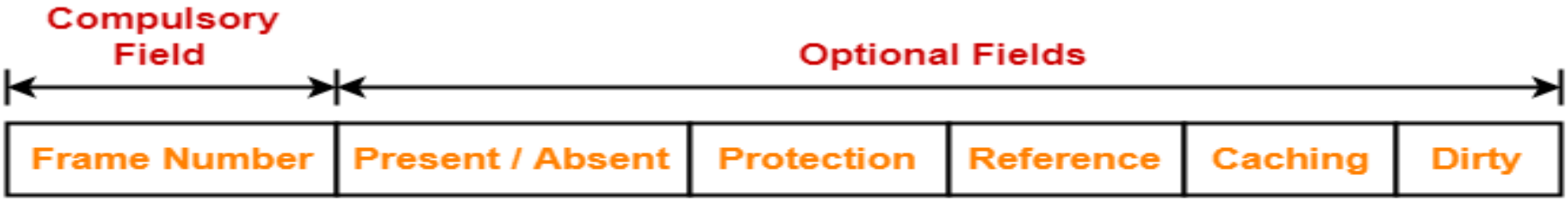
- Reference bit specifies whether that page has been referenced in the last clock cycle or not.
- If the page has been referenced recently, then this bit is set to 1 otherwise set to 0.



Page Table Entry Format

5. Caching Enabled / Disabled-

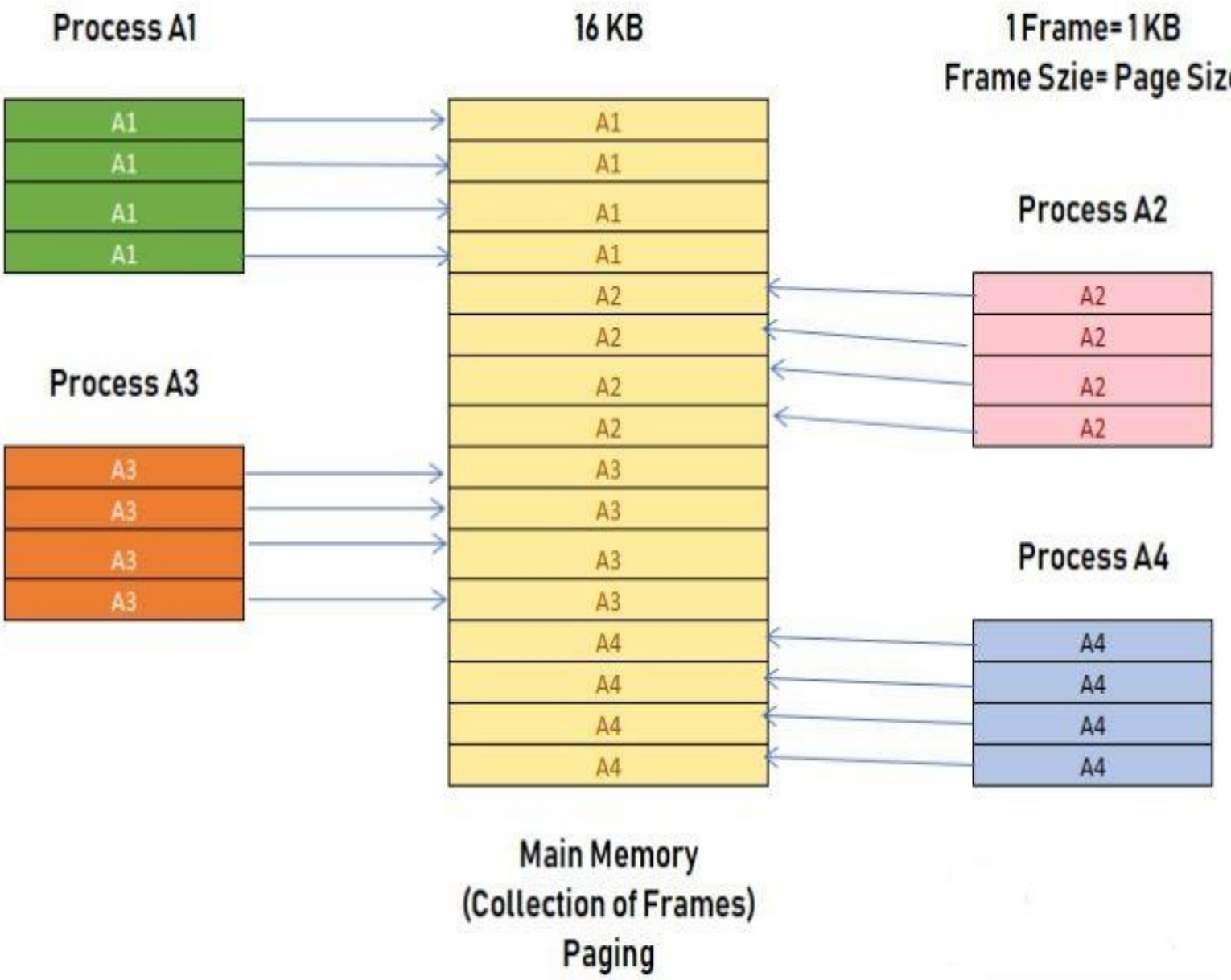
- This bit enables or disables the caching of page.
- Whenever freshness in the data is required, then caching is disabled using this bit.
- If caching of the page is disabled, then this bit is set to 1 otherwise set to 0.

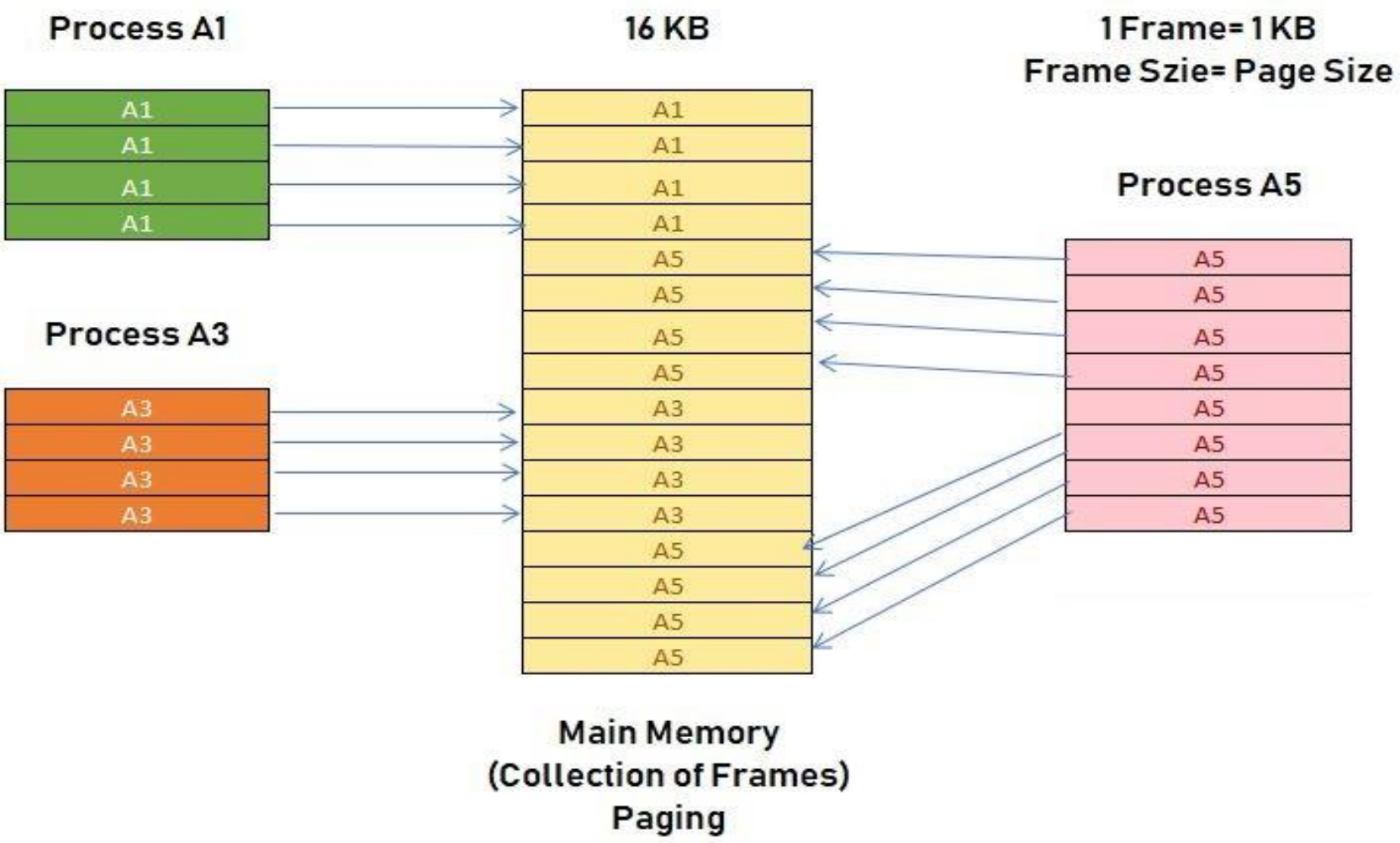


Page Table Entry Format

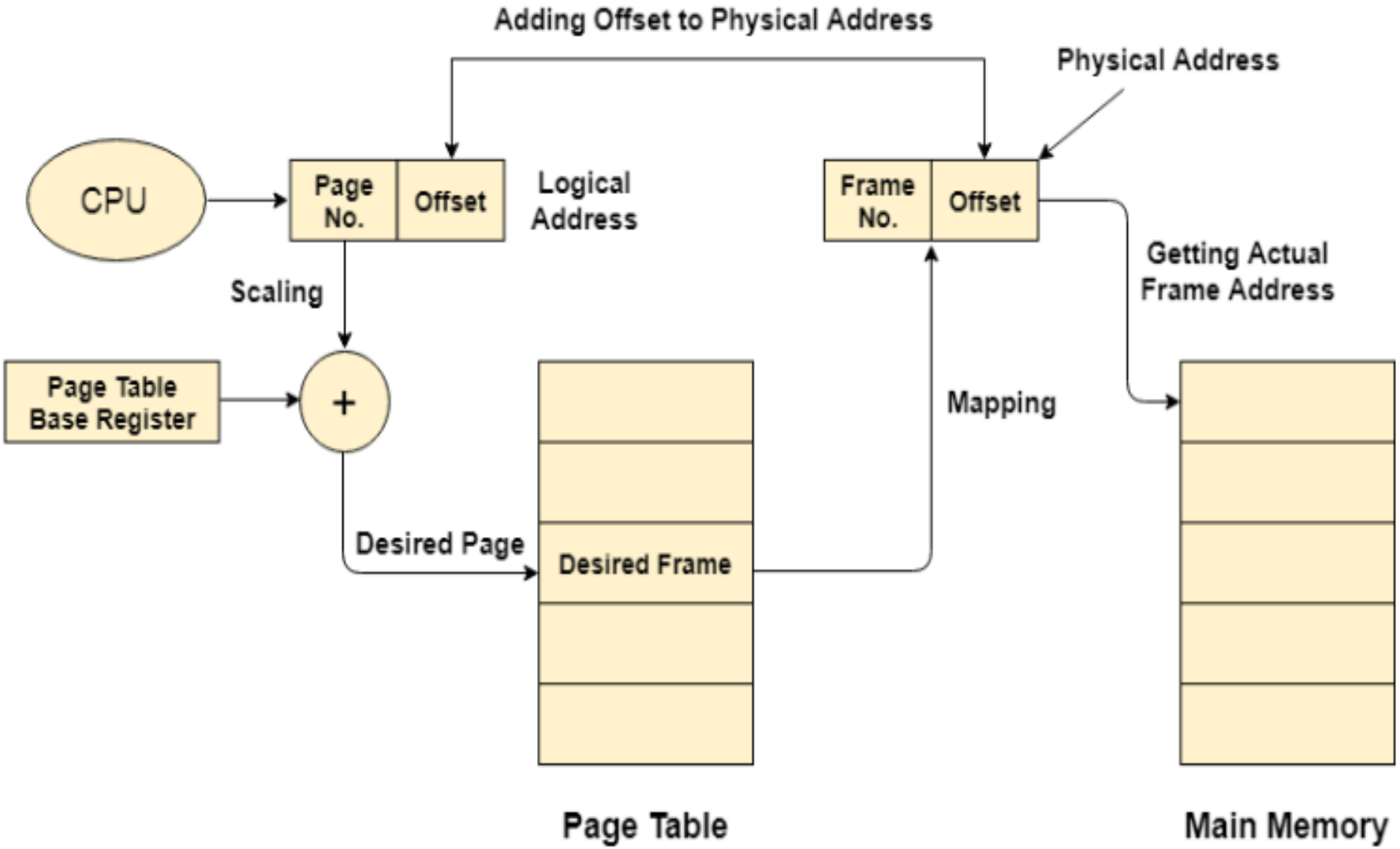
6. Dirty Bit-

- This bit is also sometimes called as "**Modified bit**".
- This bit specifies whether that page has been modified or not.
- If the page has been modified, then this bit is set to 1 otherwise set to 0.


Process A1
16 KB
1 Frame = 1 KB
Frame Szie = Page Size
A1
A1
A1
A1
A1
A1
A1
A1
Process A2
A2
A2
A2
A2
A2
A2
A2
A2
Process A3
A3
A3
A3
A3
A3
A3
A3
A3
Process A4
A4
A4
A4
A4
A4
A4
A4
A4
Main Memory
(Collection of Frames)
Paging


Process A1
16 KB
1 Frame= 1 KB
Frame Szie= Page Size
A1
A1
A1
A1
A1
A1
A1
A1
A5
A5
A5
A5
A3
A3
A3
A3
A3
A3
A3
A3
A5
A5
A5
A5
Process A3
Process A5
A5
A5
A5
A5
A5
A5
A5
A5
Main Memory
(Collection of Frames)
Paging

Address Mapping in Paging



Paging: Hardware support

Marwadi University

- Modern OS uses variations of Paging which are:
  - **Translation look aside buffer**
  - **Hierarchical paging**
  - **Inverted page table**

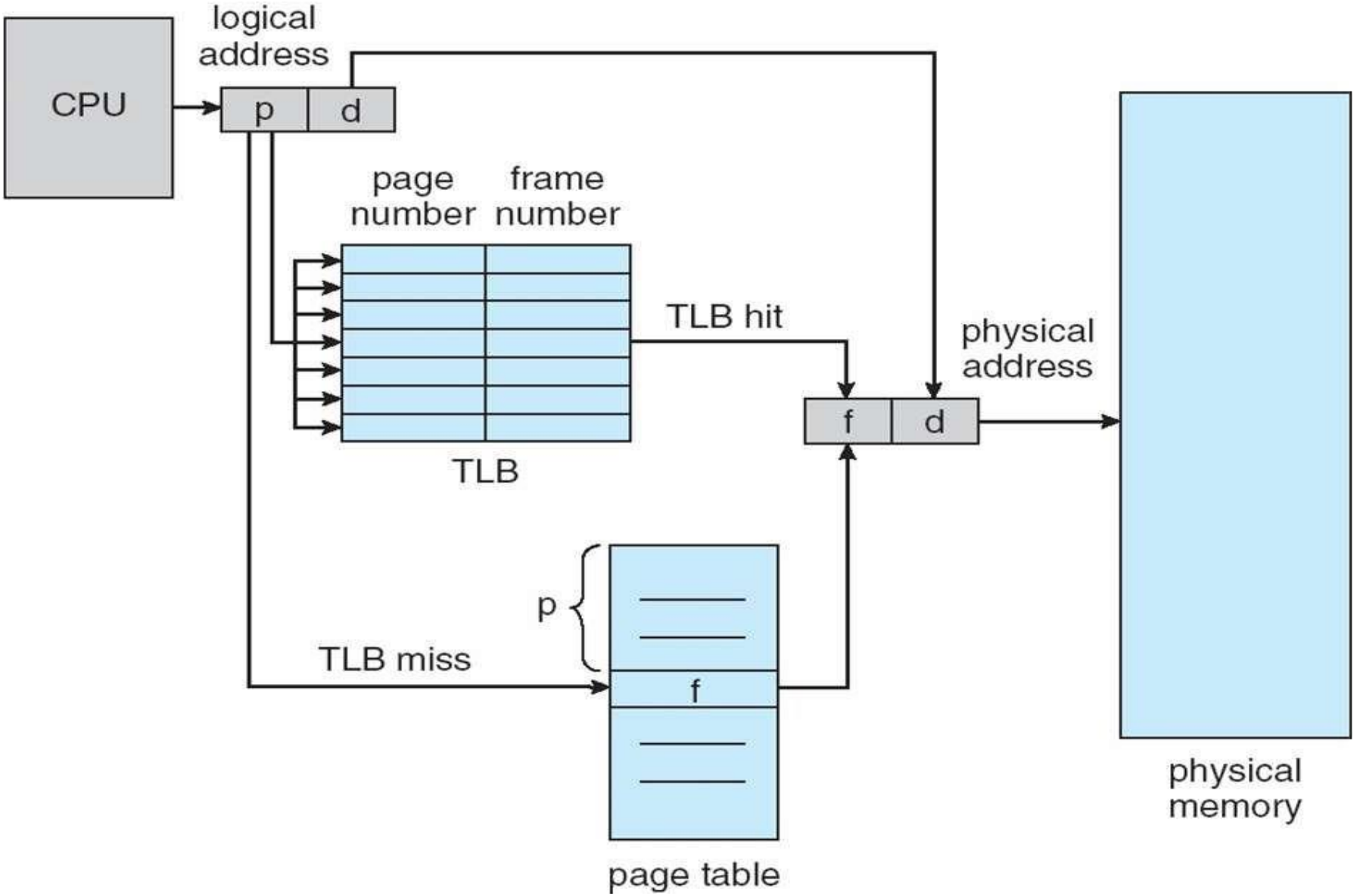
Translation look aside buffer

- Used to overcome the slower access problem.
- TLB is page table cache, which is implemented in fast associative memory.
- Its cost is high so capacity is limited, thus only subset of page table is kept in memory.
- Each TLB contains a page number and a frame number where the page is stored in the memory.

TLB - Working

- Whenever a logical address is generated, the page number of logical address is searched in the TLB.
- If the page number is found then it is know as TLB hit. In this case corresponding frame number is fetched from TLB entry and used to get physical address.
- If a match is not found then it is termed as TLB miss, in this case a memory reference to that page must be made. page table is used to get the frame number. And this entry id moved to TLB.
- If TLB is full while moving the entry then some of the existing entry in the TLB ae removed. (strategy can be Least Recently Used(LRU) to random).

TLB - Working


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

Hierarchical Page Table

Marwadi University

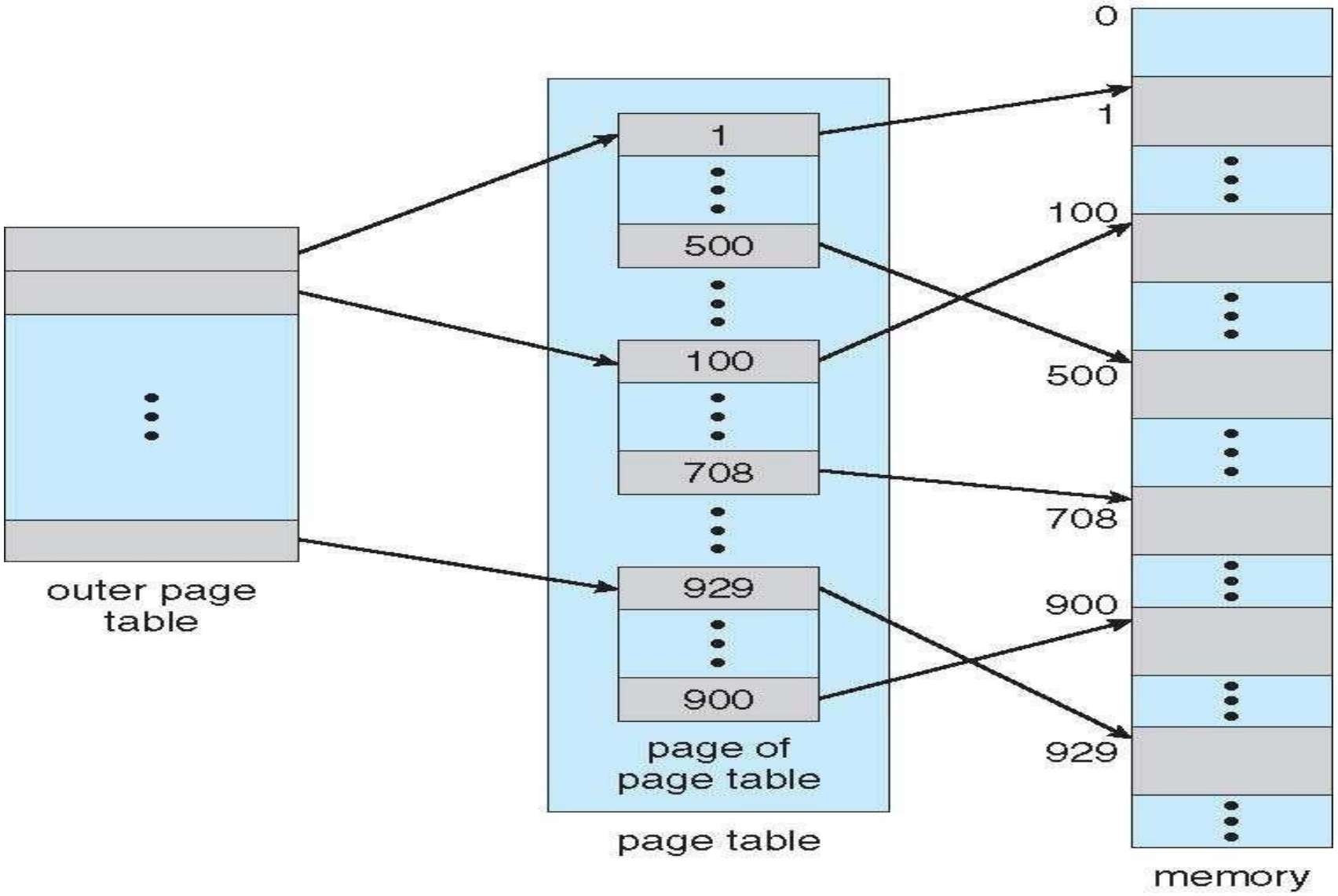
- Here the page table is divided into various **inner page table.** Also the extra **outer page table** is maintained.
- The outer table contains very limited entry, and points to the inner page table.
- And the inner page table in turn gives actual frame number.
- Here logical address is divided into three parts:

  Page number (p1) (outer page table page number)

  Page number (p2) (page table page number)

  Offset (d)

Hierarchical Page Table



Hierarchical Page Table

- **Advantage:**

All the inner pages need not be in memory simultaneously, thus reduce memory overhead.

- **Disadvantage:**

Extra memory access is required in each level of paging. Here total three memory access is required to get data.

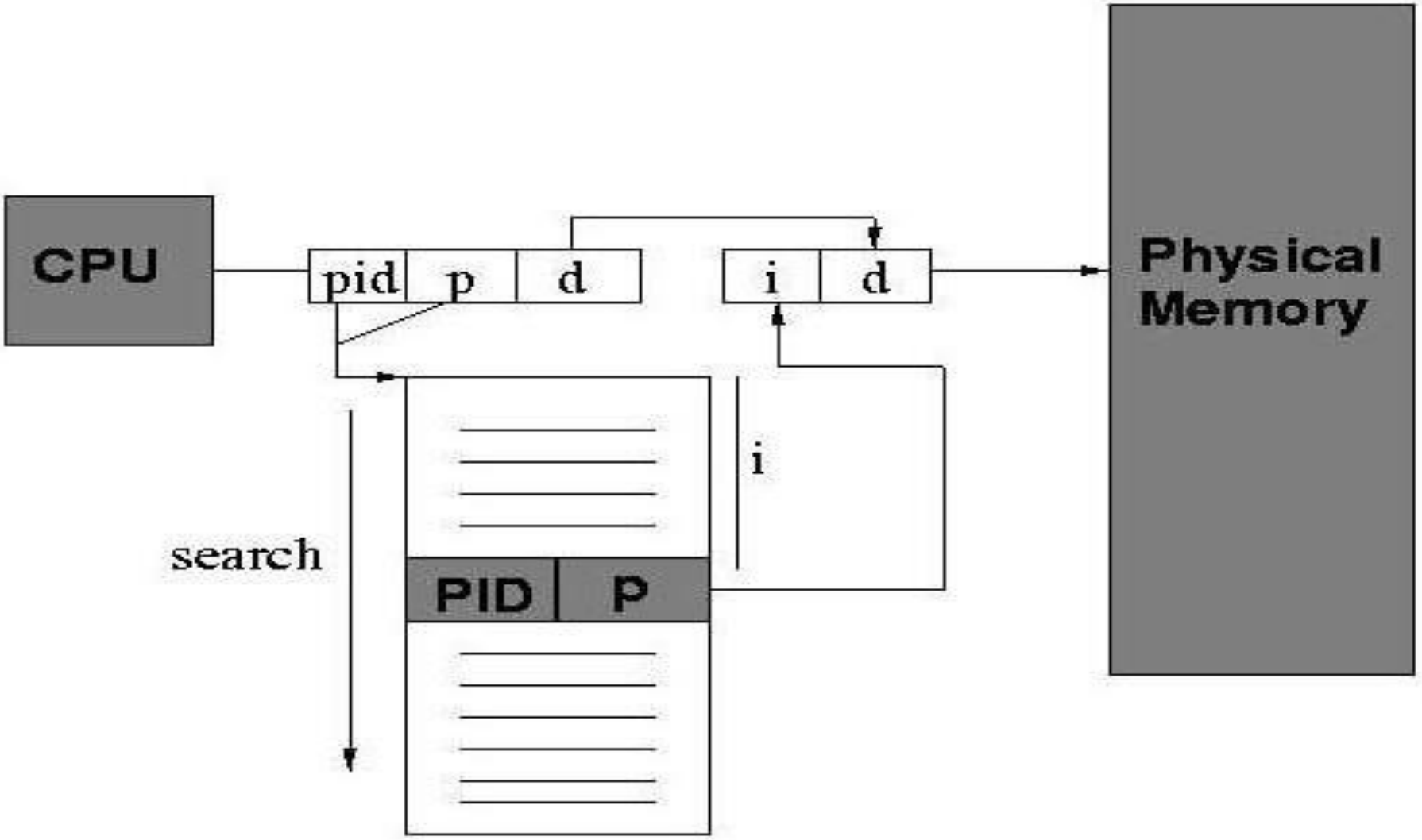
Inverted Page Table

- Is also used to **overcome the problem** of memory overhead **due to large size of page table.**
- In the previous approaches every process has a individual page table. If the size of process is high then the number of entry in page table increases. Thus it become difficult to store more than one entry for single process.
- Inverted page table solve this problem: there is one entry per frame of physical memory in table, rather than one entry per page of logical address space.
- So the size of page table is depends on the size of physical memory.
- It will create one system wide table known as **global table.**

Inverted Page Table

- We organize the inverted page table as a hash table.
- We use hash function for given (process id, page number), we will apply hash function and look for match of process id and page number.
- If we have a match, translate the address. If not, use collision resolution technique (rehash, search, linear probing) and search again.
- 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(i.e. PID).

Inverted Page Table



Aadvantages of Paging

1) Easy to use memory management algorithm
2) No need for external Fragmentation
3) Swapping is easy between equal-sized pages and page frames.

Disadvantages of Paging

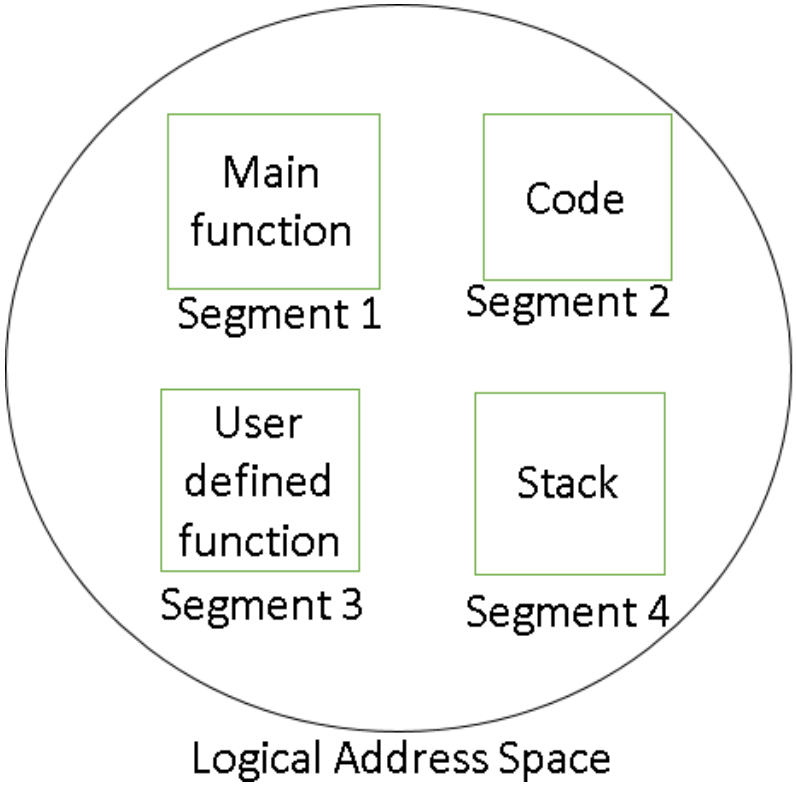
1) May cause Internal fragmentation
2) Page tables consume additional memory.
3) Multi-level paging may lead to memory reference overhead.

Segmentation

- Here the **logical address space** of a process is divided into **blocks of varying size, called segments.**
- **Each segment** contains a **logical unit** of process.
- When ever a process is to be executed, **its segments** are moved from **secondary storage** to the **main memory.**
- **Each segment** is allocated a ***chunk of free memory*** *of the* ***size equal to that segment.***
- OS maintains **one table** known as ***segment table,*** for each process. It includes **size of segment** and **location in memory** where the segment has been loaded.

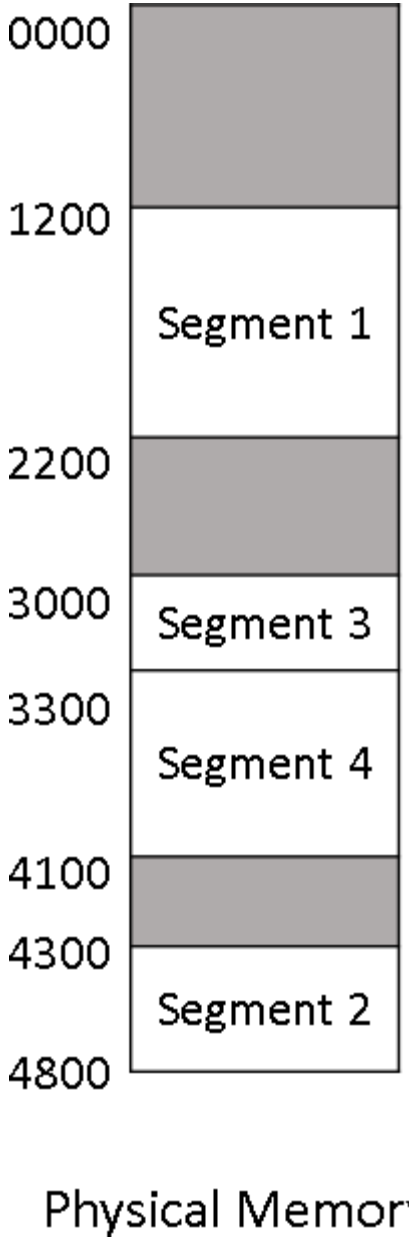
Segmentation

- Logical address is divided in to two parts:
  1. **Segment number:** identifier for segment.
  2. **Offset:** actual location within a segment.

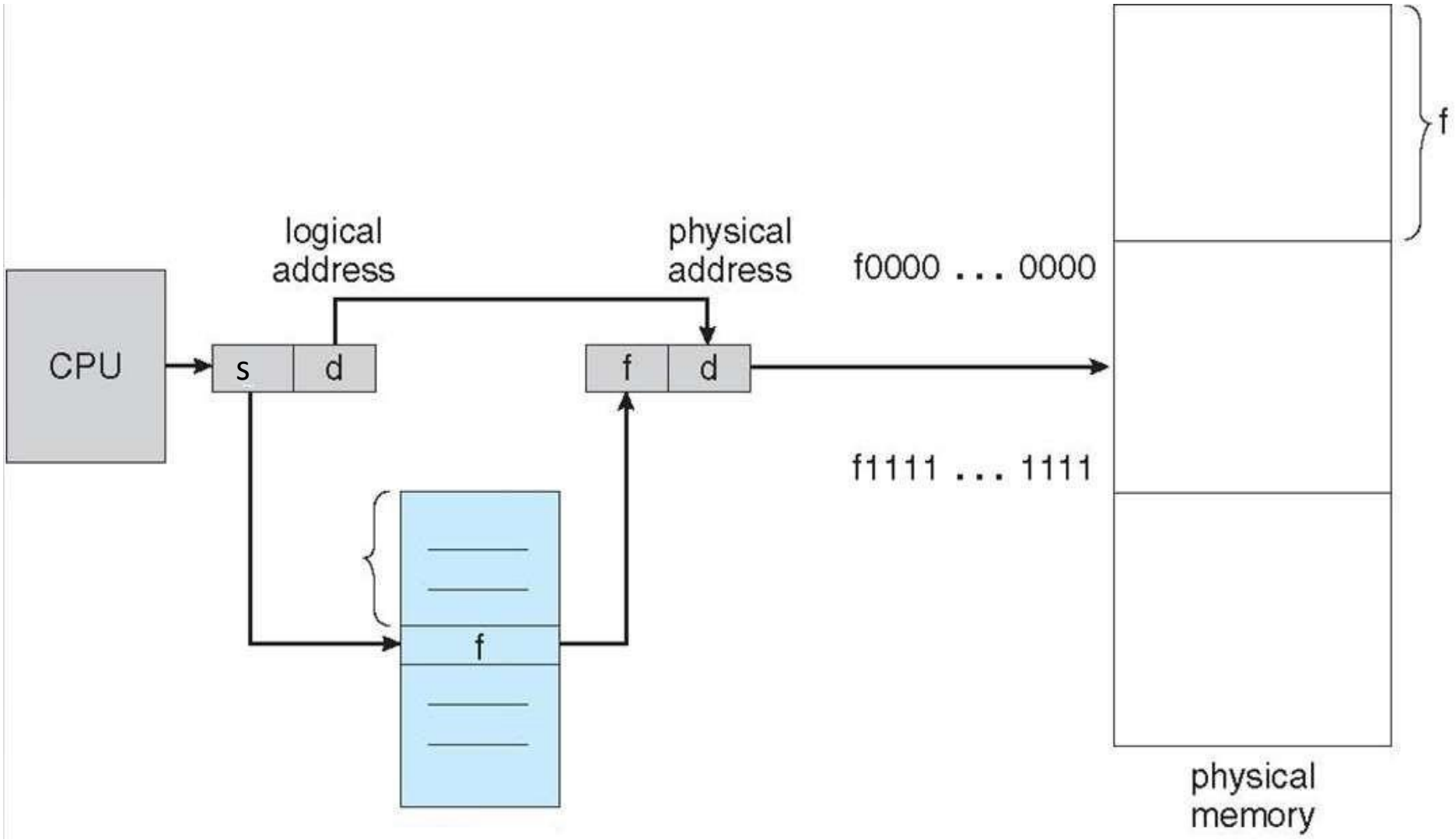


Segment No.	Size of Segment	Base address
1	1000	1200
2	500	4300
3	300	3000
4	800	3300

Segment Table



Address Mapping in 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.

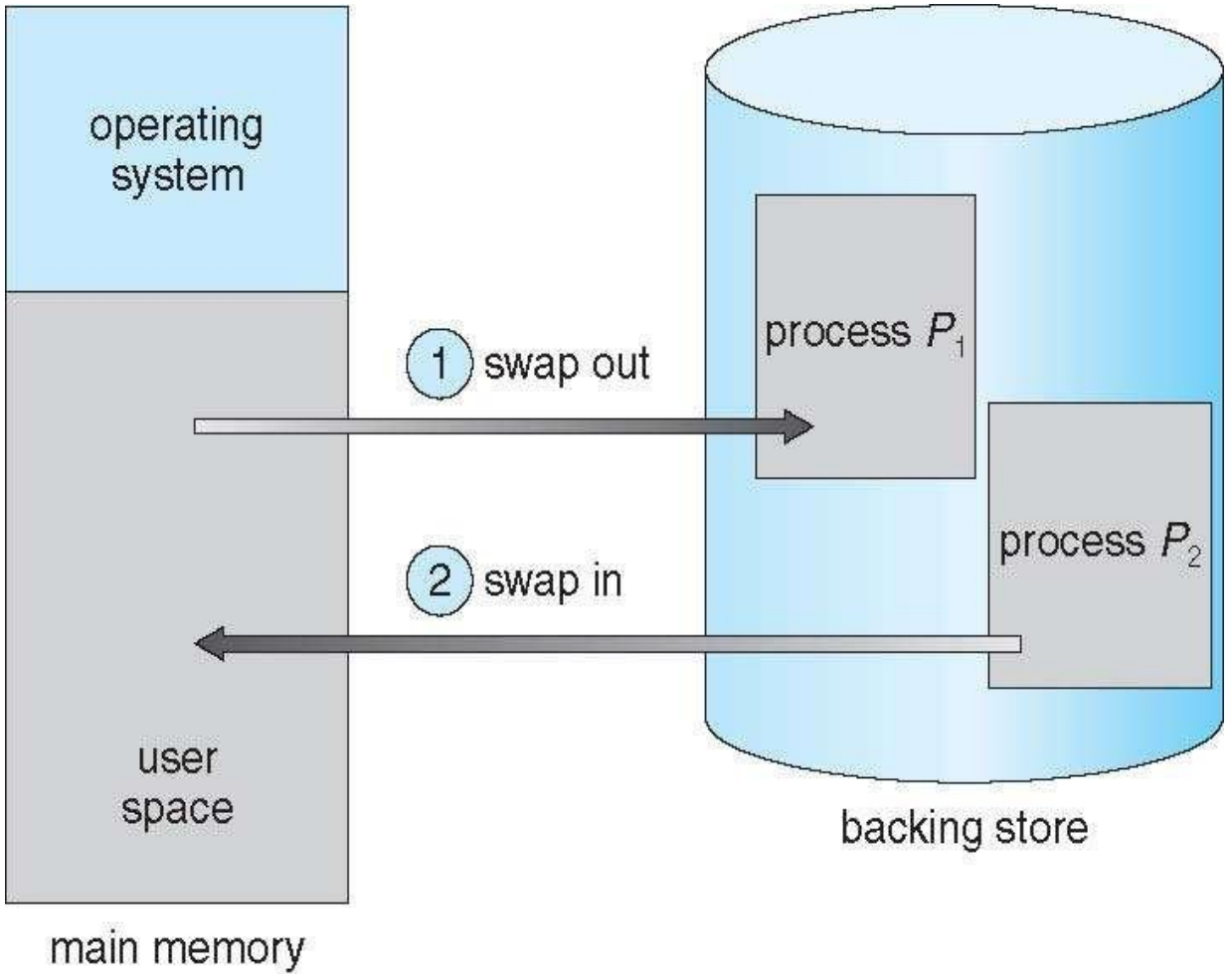
Swapping

- Example : A system has a physical memory of size 32-MB. Now suppose there are 5 process each having size 8MB that all want to execute simultaneously. How it is possible???
- **The solution is to use swapping.** Swapping is technique in which **process are moved** between **main memory** and **secondary memory or disk.**
- Swapping use some portion of secondary memory as backing store known as swapping area.
- **Operation of moving process** from **memory to swap area** is called **“swap out".** And **moving from swap area** to **memory** is known as **“swap in”.**

Swapping

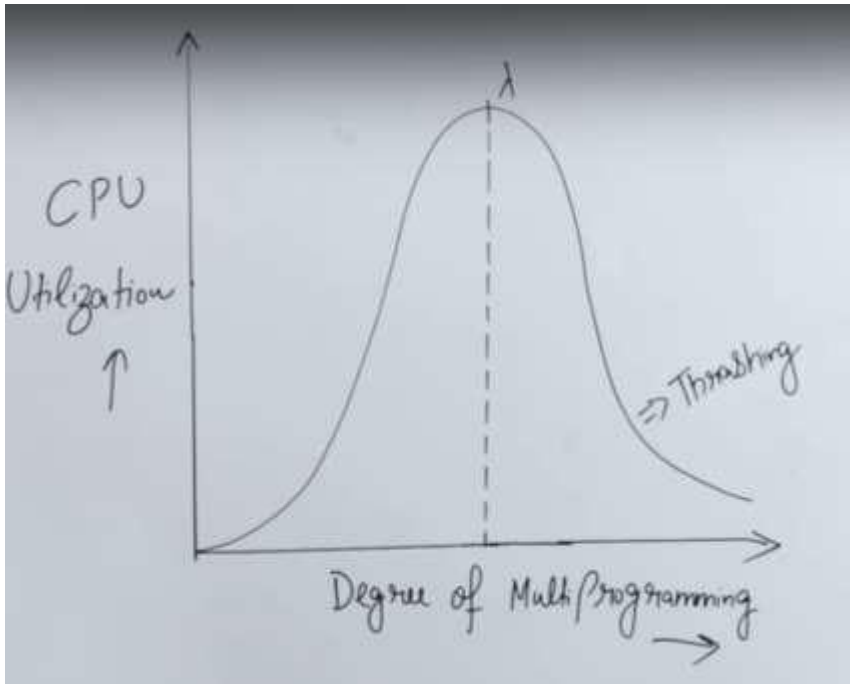
- Example : A system has a physical memory of size 32-MB. Now suppose there are 5 process each having size 8MB that all want to execute simultaneously. How it is possible???
- **The solution is to use swapping.** Swapping is technique in which **process are moved** between **main memory** and **secondary memory or disk.**
- Swapping use some portion of secondary memory as backing store known as swapping area.
- **Operation of moving process** from **memory to swap area** is called **“swap out".** And **moving from swap area** to **memory** is known as **“swap in”.**

Swapping


operating system
user space
main memory
1 swap out
2 swap in
process P_1
process P_2
backing store

Thrashing

- CPU utilization is directly linked to degree of multi programming.
- As RAM is limited, so we are using paging concept here.
- For e.g. I have 100 processes and each process is divided into some number of pages.
- Degree of multiprogramming is maximum, if I will place one page of each process into RAM.
- Degree of multiprogramming is achieved here as every process have its one page available in the RAM, but it causes maximum page faults.


CPU
Utilization
Thrashing
Degree of Multiprogramming

Thrashing

- Due to this **performance** of the **system is decreased** .
- After a certain limit λ thrashing occurs.
- **To avoid this problem :**

i. **Increase the main memory size**

ii. **Efficiently use long term scheduler**.

P1 – page1
P2– page1
P3 – page1
P4 – page1
…
P100 – page1

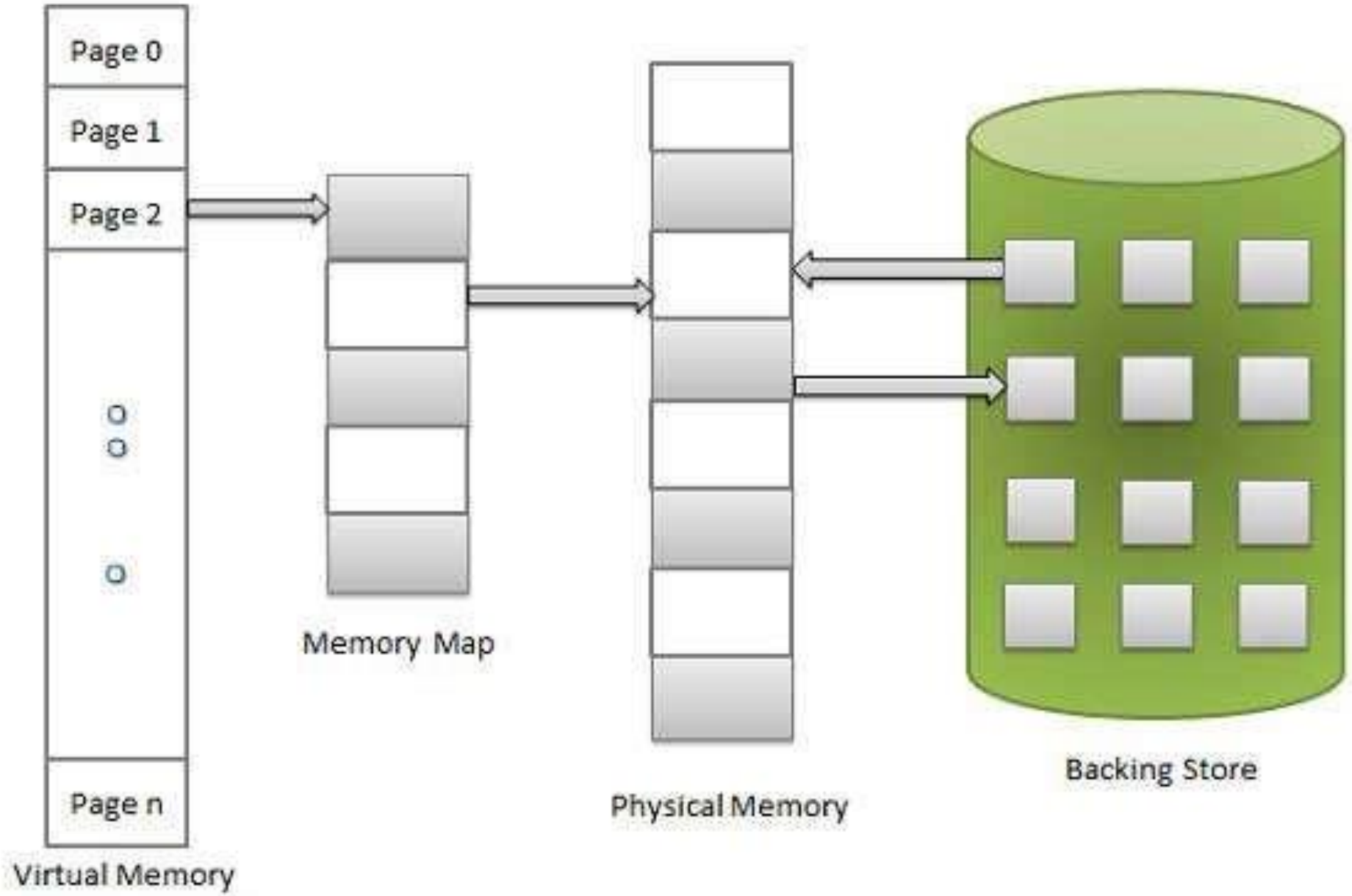
Virtual Memory

- A virtual memory is technique that allows a process to execute even though it is partially loaded in main memory.
- The basic idea behind virtual memory is that the combined size of the program, data, and stack may exceed the amount of physical memory(main memory) available for it.
- The operating system keeps those parts of the program currently in use in main memory, and the rest on the disk.
- These program-generated addresses are called virtual addresses and form the virtual address space.
- **MMU (Memory Management Unit)** maps the virtual addresses onto the physical memory addresses

Advantage of Virtual Memory

Marwadi University

- Less number of I/O would be needed to load or swap each user program into memory.
- A program would no longer be constrained by the amount of physical memory that is available. User would be able to write programs for an extremely large virtual address space.
- Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

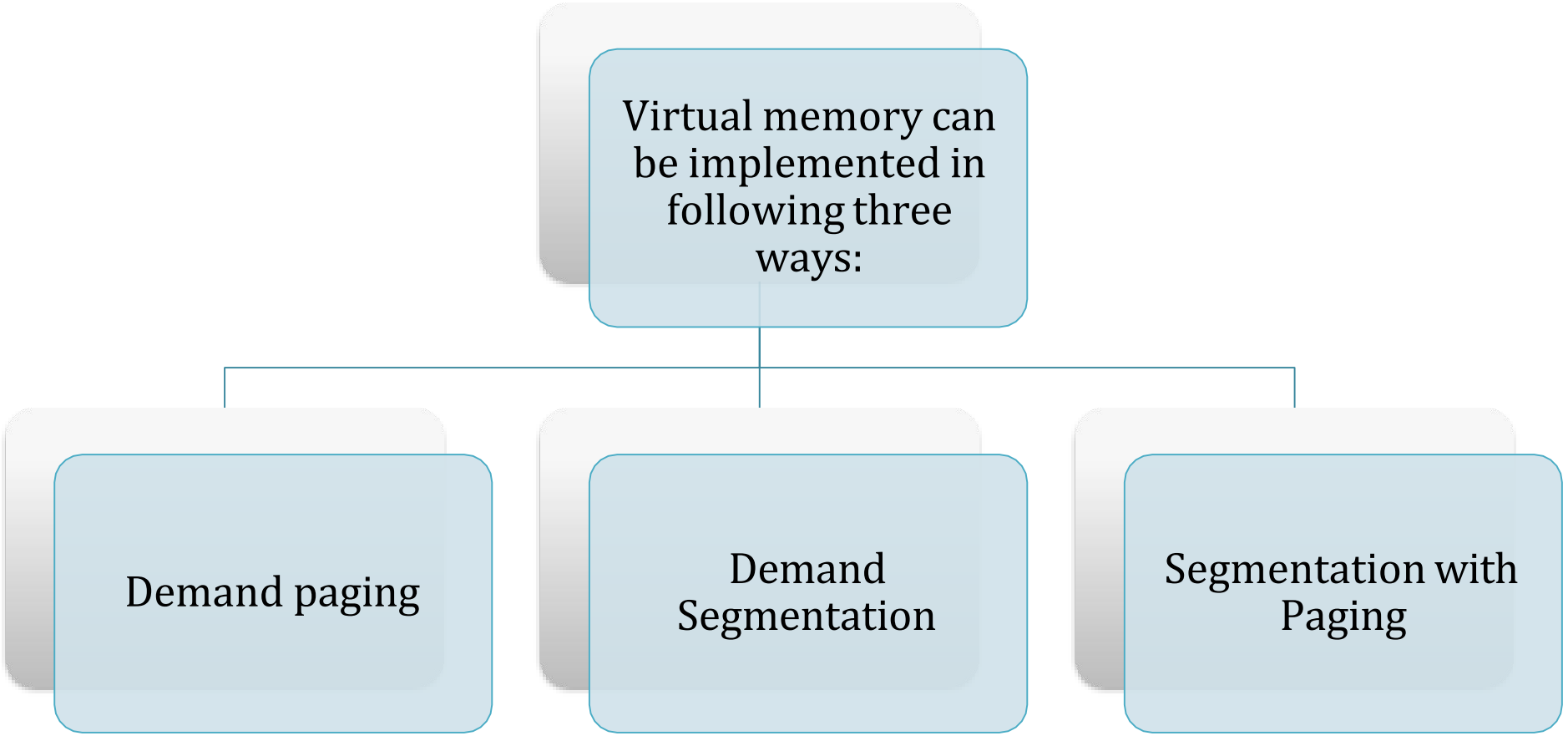

Page 0
Page 1
Page 2
Page n
Virtual Memory
Memory Map
Physical Memory
Backing Store

Virtual Memory: Hardware and control structures

Marwadi University

- Virtual memory involves the separation of logical memory perceived(aware) by user from physical memory.
- This separation allows extremely large virtual memory to be provided for programmers when only smaller amount of physical memory is available.
- Thus programmer need not to worry about the amount of memory available.

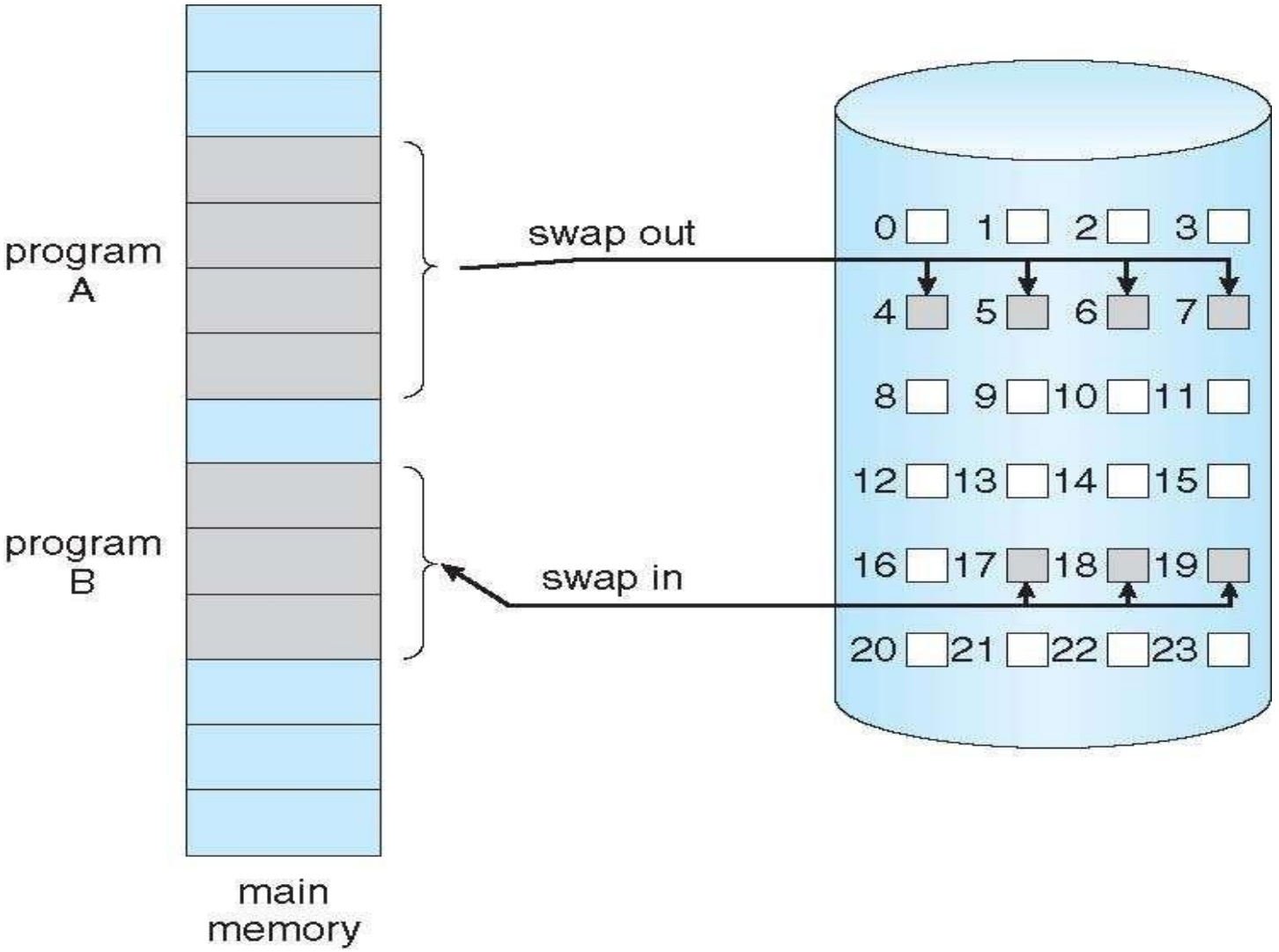
Virtual Memory



Demand Paging

- Demand paging is similar to a paging system with swapping where processes may reside in secondary memory. When we want to execute a process, we swap it into the main memory.
- Rather than swapping entire process into memory we use lazy swapper.
- A lazy swapper never swaps page into memory, unless that page will be needed.
- If some process is needs to be swap in, then pager(page table handler) brings only those pages which will be used by process.
- Thus avoid reading unused pages and decrease swap time and amount of physical memory needed.

Demand Paging



Demand Paging- Page Fault

If a process tries to access a page that is not in main memory then it causes page fault.

Pager will generate trap to the OS, and tries to swap in.

Page table includes the **valid-invalid bit** for each page entry.

If the bit is valid then page is currently available in to the memory.

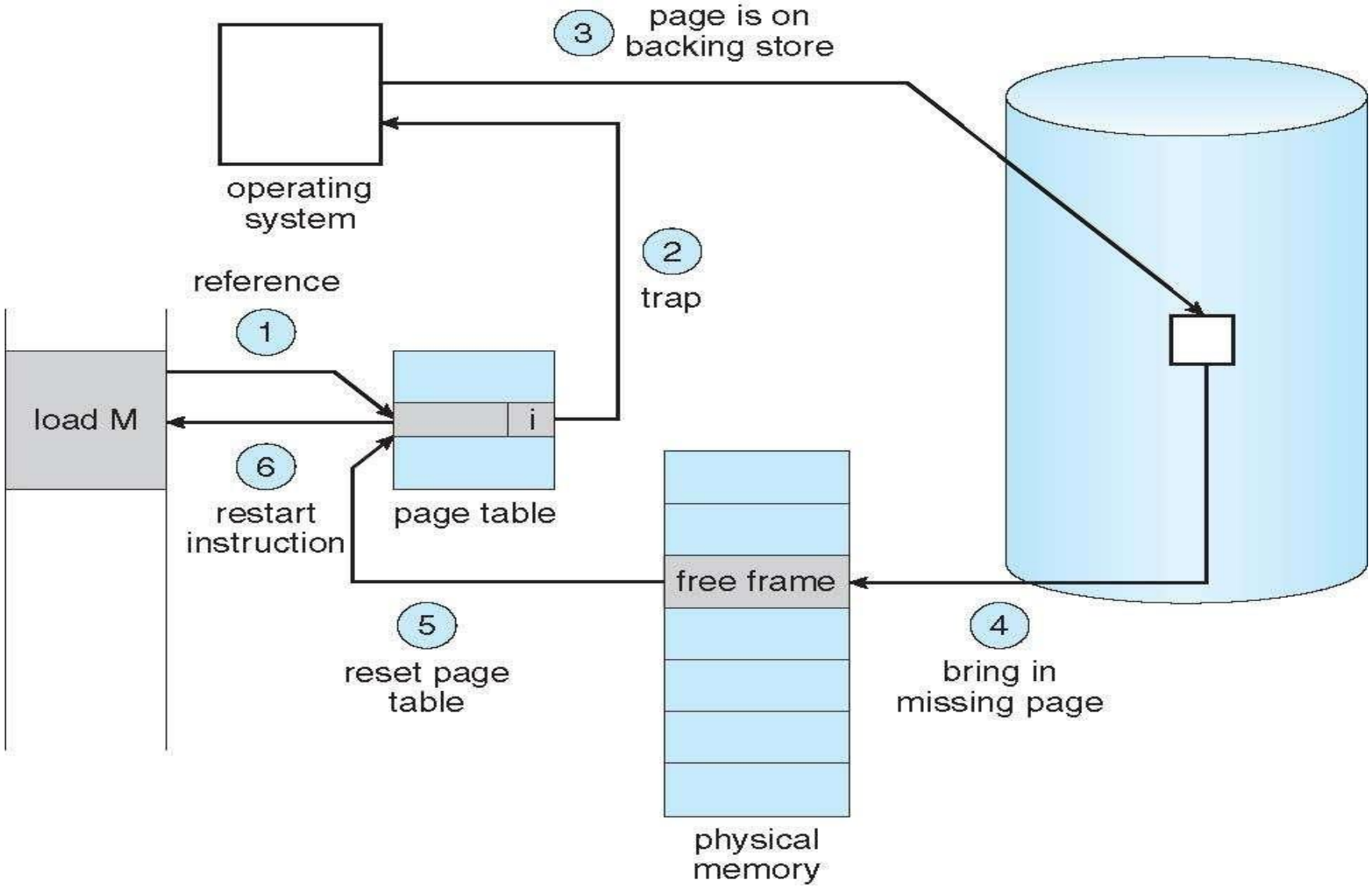
If it is set to invalid then page is either invalid or not present in main memory.

Demand Paging – Page Fault

Following steps are followed to manage page fault:

1. Check page table for the process to determine whether the reference is valid or invalid.
2. If the page is invalid then terminate the process, but if the page is valid but currently not available in main memory, then generate trap instruction.
3. OS determines the location of that page on swap area.
4. Then it will use free frame list to find out free frame. OS will schedule disk operation to read desired page into newly allocated memory.
5. When disk read is complete modify the page table and set reference bit to valid.
6. Restart the instruction.

Demand Paging – Page Fault



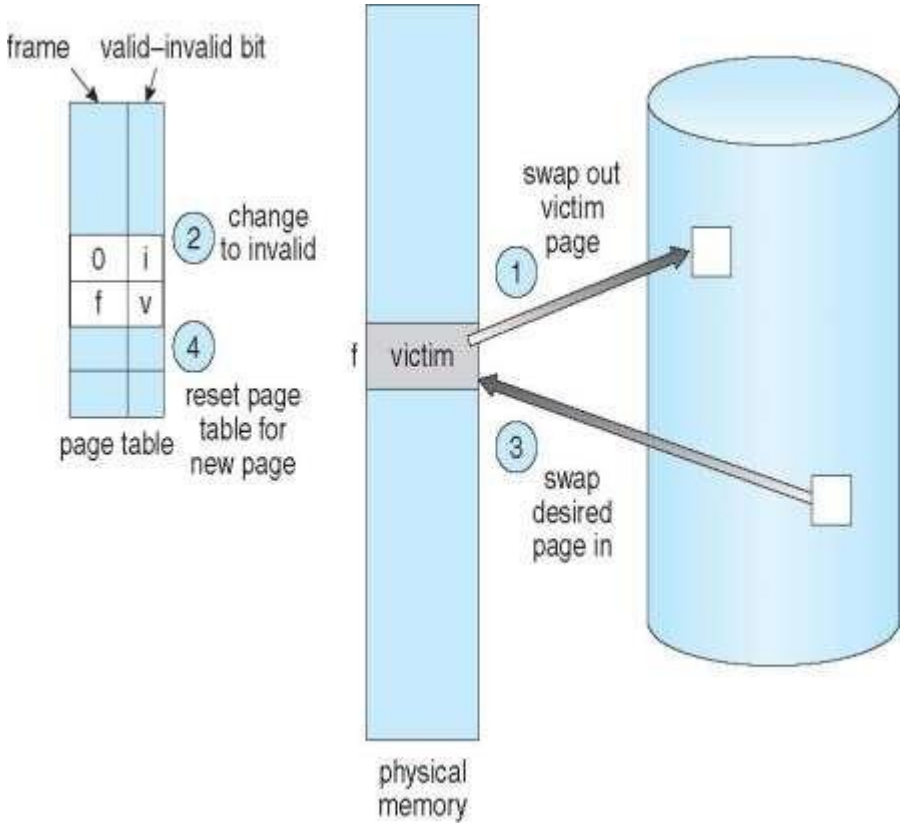
Page Replacement Policies - Basics

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
     - Write victim frame to disk if dirty

3. Bring the desired page into the (newly) free frame; update the page and frame tables

4. Continue the process by restarting the instruction that caused the trap



First In First Out (FIFO)

- The simplest page replacement algorithm.
- When the page must be replaced the oldest page is chosen.
- one FIFO queue is maintained to hold all pages in memory.
- Replace the page which at the top of the queue and add new pages from rear end (tail)of the queue.

Reference String : 2 3 2 1 5 2 4 5 3 2 5 2

- 3 frames (3 pages can be in memory at a time per process)

- **Reference String : 7,0,1,2,0,3,0,4,2,3,0,3,1,2,0**
- 3 frames (3 pages can be in memory at a time per process) FIFO

F3			1	1	1	1	0	0	0	3	3	3	3	2	2
F2		0	0	0	0	3	3	3	2	2	2	2	1	1	1
F1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0
	F	*	*	*	hit	*	*	*	*	*	*	2hit	*	*	Hit

First In First Out (FIFO)

- Reference String : **7,0,1,2,0,3,0,4,2,3,0,3,1,2,0**
- 3 frames (3 pages can be in memory at a time)

F3			1	1	1	1	0	0	0	3	3	3	3	2	2
F2		0	0	0	0	3	3	3	2	2	2	2	1	1	1
F1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0
	F	F	F	F	Hit	F	F	F	F	F	F	hit	F	F	hit

Page faults/Page Miss = 12 **Page hit** = 3

- **Miss Ratio** = Num of miss/ num of reference => (12/15)*100= 80%
- **Hit Ratio** = Num of hit/ num of reference => (03/15)*100= 20%

First In First Out (FIFO)

- Reference string: **1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5**
- 3 frames (3 pages can be in memory at a time per process)

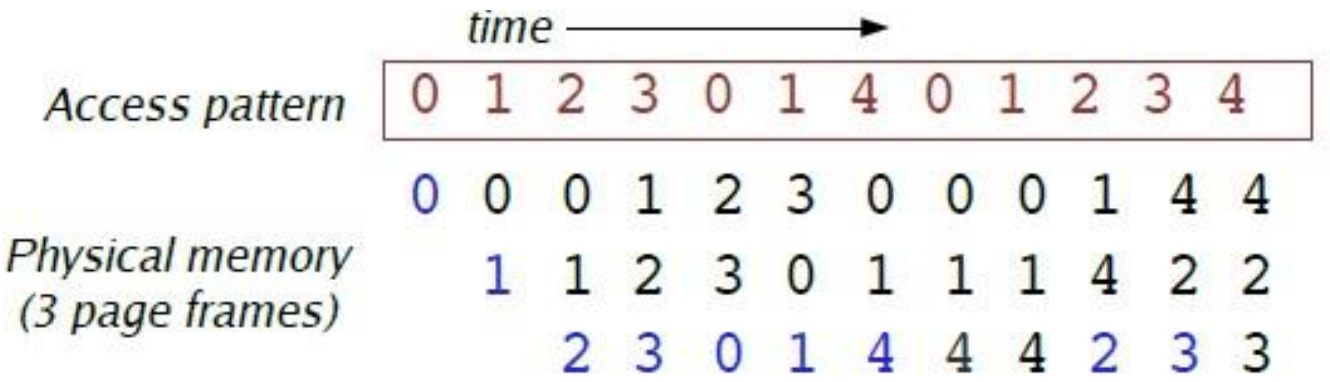
9 page faults

- **Find out the total page faults in the given reference string?**

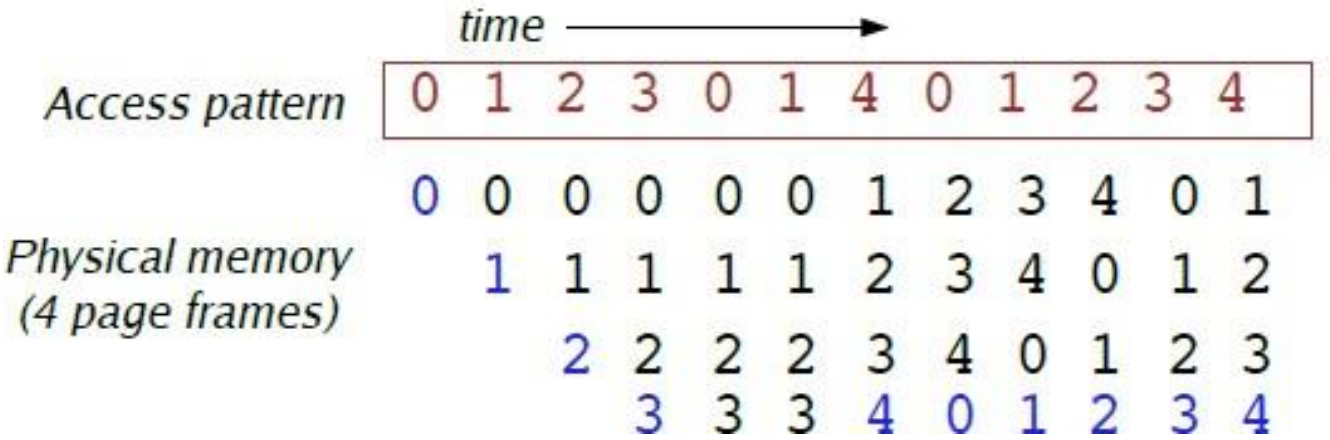
By using 4 frames

- **10 page faults and 2 page hits**

Belady's Anomaly in FIFO



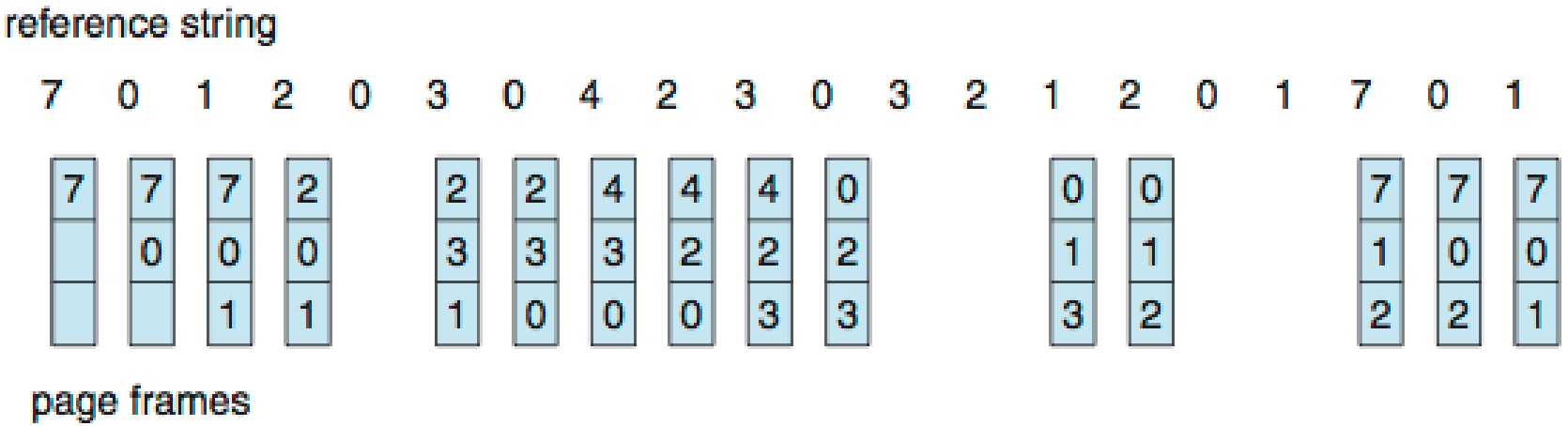
9 page faults!



10 page faults!

First In First Out (FIFO)

- Reference string: **7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1**
- 3 frames (3 pages can be in memory at a time per process)



- 15 page faults

First In First Out (FIFO)

Marwadi University

Advantage

- Very simple.
- Easy to implement.

Disadvantage

- A page fetched into memory a long time ago may have now fallen out of use.
- This reasoning will often be wrong, because there will often be regions of program or data that are heavily used throughout the life of a program.
- Those pages will be repeatedly paged in and out by the FIFO algorithm.

Least Recently Used (LRU)

Marwadi University

It is based on the observation that if pages that have been *heavily used in the last few instructions will probably be heavily used again in the next few.*

Conversely, pages that have not been used for ages will probably remain unused for a long time.

This idea suggests a realizable algorithm: when a page fault occurs, throw out the page that has been *unused for the longest time*.

Least Recently Used (LRU)

- Example-1 Reference string: 1, 2, 3, 4, 1, 2, **5**, 1, 2, **3**, **4**, **5**
- **8 page faults**

(replace the least recently used pages in past)

F1	**1**	**1**	**1**	**1**	**1**	**1**	**5**
F2	2	2	2	2	2	2	2
F3	3	**3**	5	5	**5**	4	4
f4	4	4	4	**4**	3	3	3
	4*	2hit	*	2hit	*	*	*

Least Recently Used (LRU)

- Example-2
- Reference string: **7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1**
- 3 frames (3 pages can be in memory at a time per process)

F1	7	2	2	2	2	4	4	4	0	0	1	1	1	1	1	1
F2	0	0	0	0	0	0	0	3	3	3	3	3	0	0	0	0
F3	1	1	1	3	3	3	2	2	2	2	2	2	2	2	7	7
	3*	*	hit	*	hit	*	*	*	*	4hit	*	hit	*	hit	*	2hit

Least Recently Used (LRU)

Marwadi University

1. Reference string: **1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6**
2. Reference string: **0 1 7 2 3 2 7 1 0 3**

- 4 frames (4 pages can be in memory at a time per process)

- 10 page faults

- 3 frames (3 pages can be in memory at a time per process)

- 15 page faults

Most Recently Used (MRU)

Idea of MRU- Replace the page which is most recently used in past.

- Example
- 1. Reference string: **7,0,1,2,0,3,0,4,2,7,3**
- 2. Reference string: **0 1 7 2 3 2 7 1 0 3**

3 frames (3 pages can be in memory at a time per process)

F1	7	7	7	7	7	7	7	3
F2	0	0	0	3	0	4	4	4
F3	1	2	2	2	2	2	2	2
	3*	*	hit	*	*	*	2hit	*

Optimal Page Replacement

Marwadi University

- Its best page replacement policy.
- The Optimal policy selects for replacement the page that will *not be used for longest period* of time.
- Impossible to implement (need to know the future) but serves as a standard to compare with the other algorithms we shall study.

Optimal Page Replacement

- **For example: 3 frames**
- Reference string: **1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6**

F1	1	1	1	1	1	1	3	3	3	3	3	3	6
f2	2	2	2	2	2	2	2	7	7	2	2	2	2
f3	3	4	4	5	6	6	6	6	6	6	1	1	1
	3*	*	2hit	*	*	3hit	*	*	2hit	*	*	2hit	*

Optimal Page Replacement

- Reference string : **1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5**(check next pages)
- 4 frames

F1	1	1	1	1	4	
F2	2	2	2	2	2	
F3	3	3	3	3	3	
f4	4	4	5	5	5	
	4*	2 hit	*	3hit	*	

Optimal Page Replacement

Marwadi University

- Need an approximation of how likely each frame is to be accessed in the future
- If we base this on past behavior we got a way to track future behavior
- Tracking memory accesses requires hardware support to be efficient

Optimal Page Replacement

Marwadi University

Advantage:

- Lowest page faults.
- Can Improves performance of system as it reduces number of page faults so requires less swapping.

Disadvantage:

- Very difficult to implement.


THANK
YOU