

Information and Communication Technology

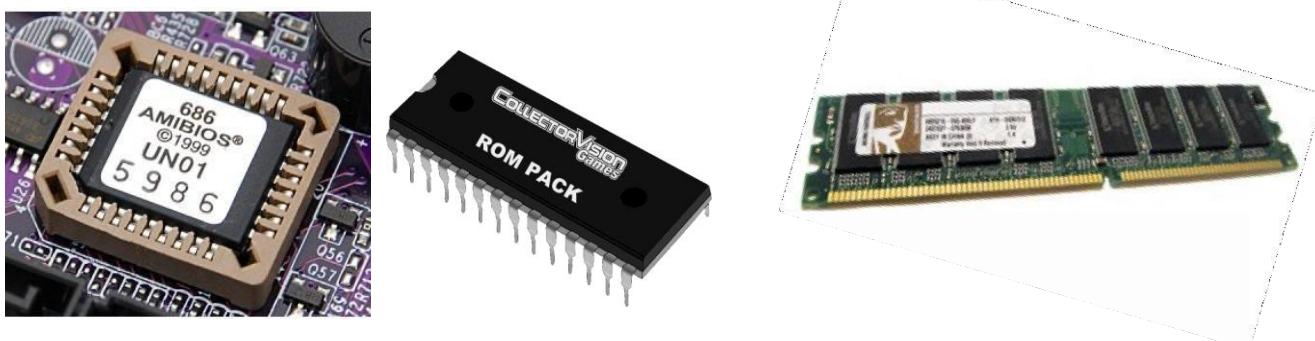
MEMORY MANAGEMENT & VIRTUAL MEMORY

Study Guide

**Assistant Prof. Sumersing Patil
CSE-AI&DS, PIET
Parul University**

What is Memory?

- › Computer memory is any **physical device capable of storing information temporarily or permanently**.
- › Types of memory
 1. **Random Access Memory (RAM)**, is a **volatile memory** that **loses its contents when the computer or hardware device loses power**.
 2. **Read Only Memory (ROM)**, is a **non-volatile memory**, sometimes abbreviated as NVRAM, is a memory that **keeps its contents even if the power is lost**.
 3. Computer **uses special ROM** called **BIOS (Basic Input Output System)** which **permanently stores the software** needed to access computer hardware such as hard disk and then load an operating system into RAM and start to execute it.



- › Computer memory is any **physical device capable of storing information temporarily or permanently**.
- › Types of memory
 3. **Programmable Read-Only Memory (PROM)**, is a memory chip on which you can store a program. But once the PROM has been used, you **cannot wipe it clean and use it to store something else**. Like ROMs, PROMs are non-volatile. E.g CD-R
 4. **Erasable Programmable Read-Only Memory (EPROM)**, is a special type of PROM that **can be erased by exposing it to ultraviolet light**. E.g

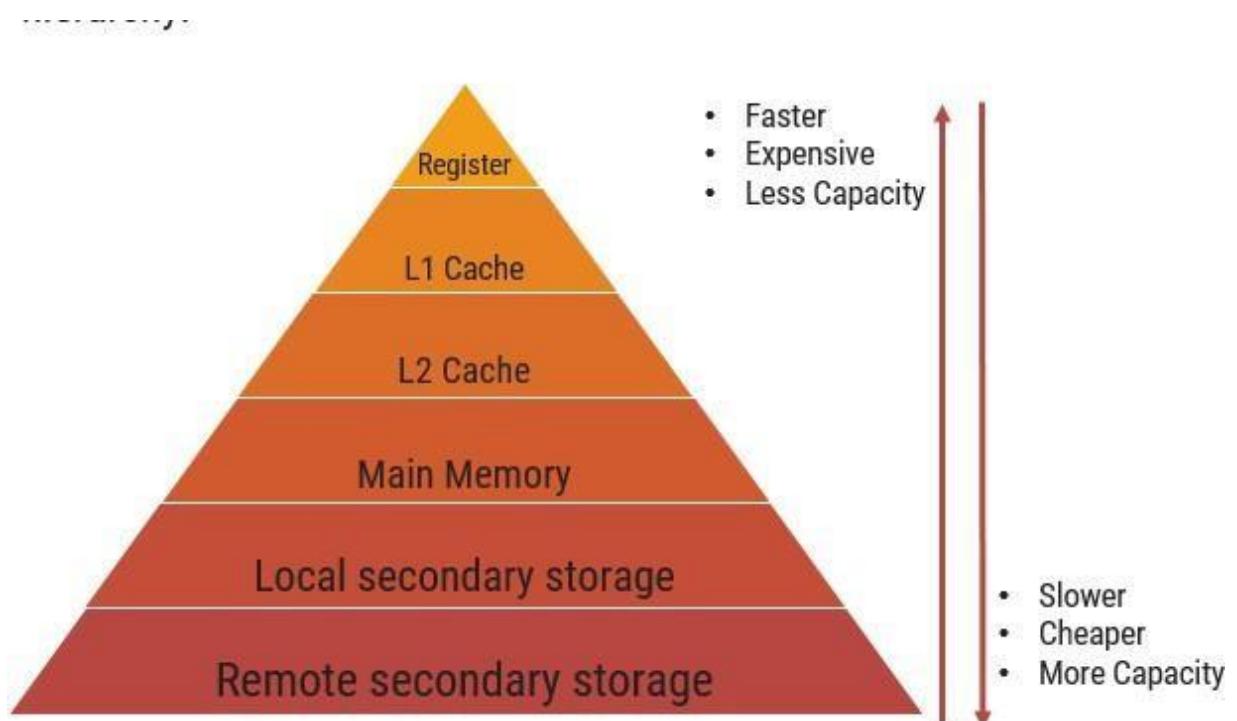
CD-RW

5. **Electrically Erasable Programmable Read-Only Memory (EEPROM)**, is a special type of PROM that can be erased by exposing it to an electrical charge. E.g Pendrive



What is Memory Hierarchy?

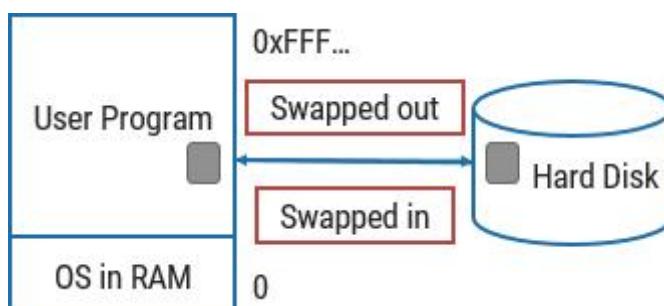
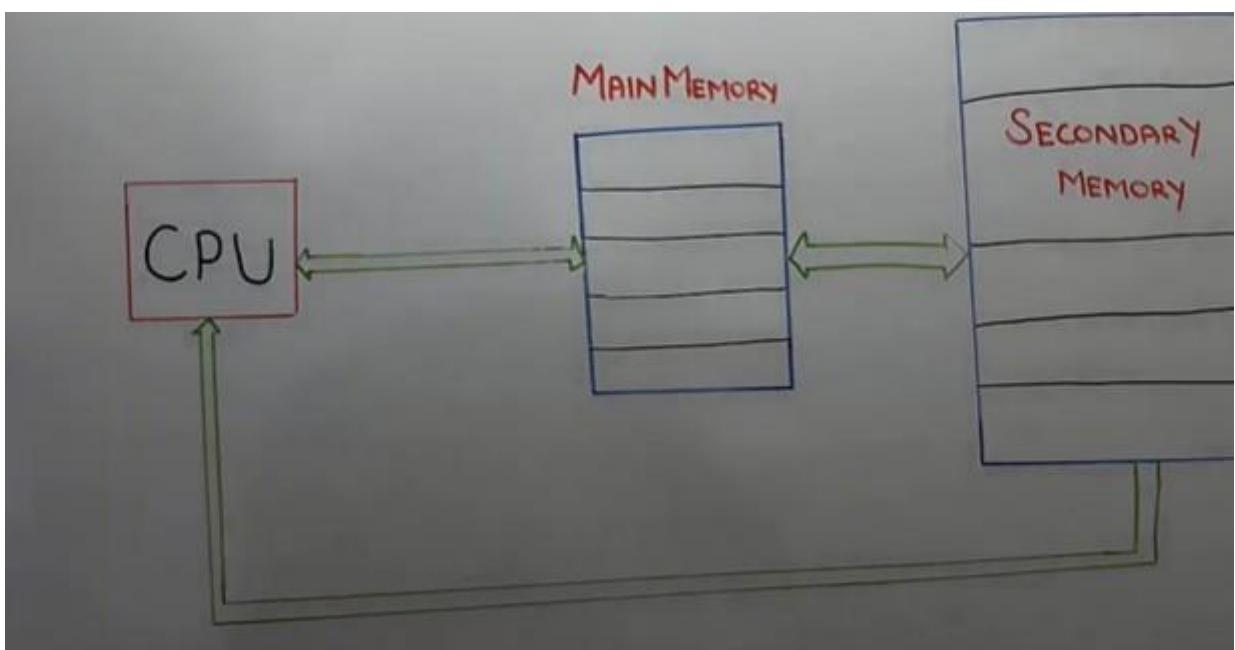
- The hierarchical arrangement of storage in current computer architectures is called the memory hierarchy.



- What if we want to run multiple programs?
4. OS saves entire memory on disk
 5. OS brings next program
 6. OS runs next program

- We can use swapping to run multiple programs concurrently.

- The process of bringing in each process in its entirety in to memory, running it for a while and then putting it back on the disk is called swapping.



Ways to implement swapping system

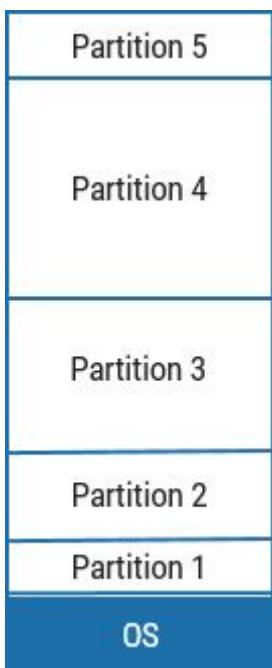
- Two different ways to implement swapping system
 7. Multiprogramming with **fixed partitions**
 8. Multiprogramming with **dynamic partitions**

Multiprogramming with fixed partitions

- Here memory is **divided into fixed sized partitions**.
- Size can be equal or unequal** for different partitions.
- Generally **unequal partitions are used** for better utilizations.

- Each partition can accommodate exactly one process, means only single process can be placed in one partition.

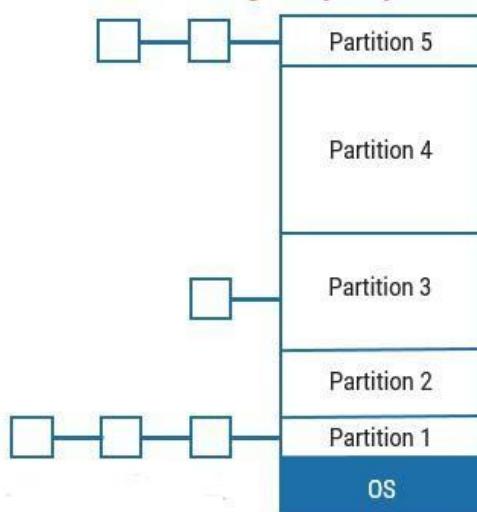
- ▶ The partition **boundaries** are **not movable**.
- ▶ Whenever any **program needs to be loaded in memory**, a **free partition big enough to hold the program** is found. This **partition will be allocated to that program or process**.
- ▶ If there is no free partition available of required size, then the **process needs to wait**. Such **process will be put in a queue**.



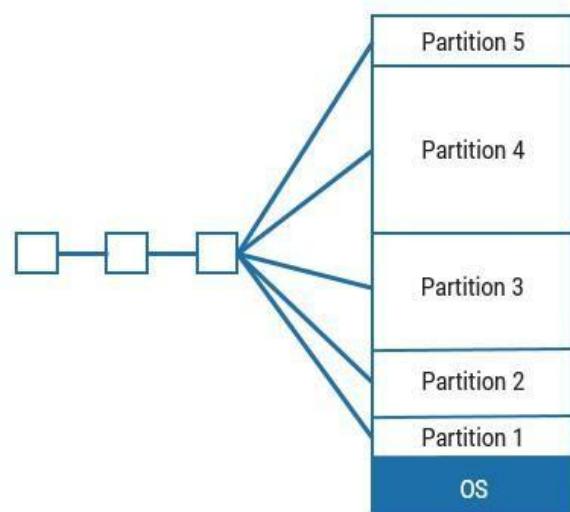
Multiprogramming with fixed partitions

- ▶ There are two ways to maintain queue

1. Using Multiple Input Queues



2. Using Single Input Queue

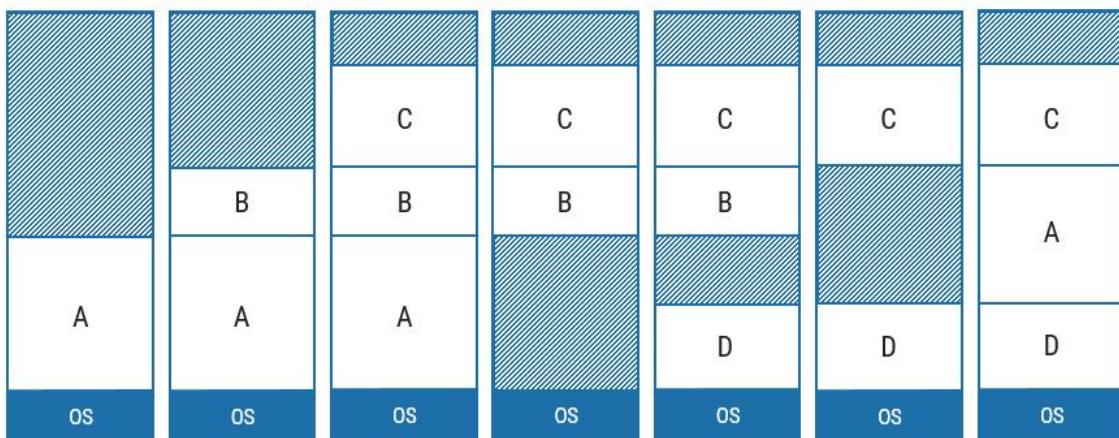


Multiprogramming with dynamic partitions

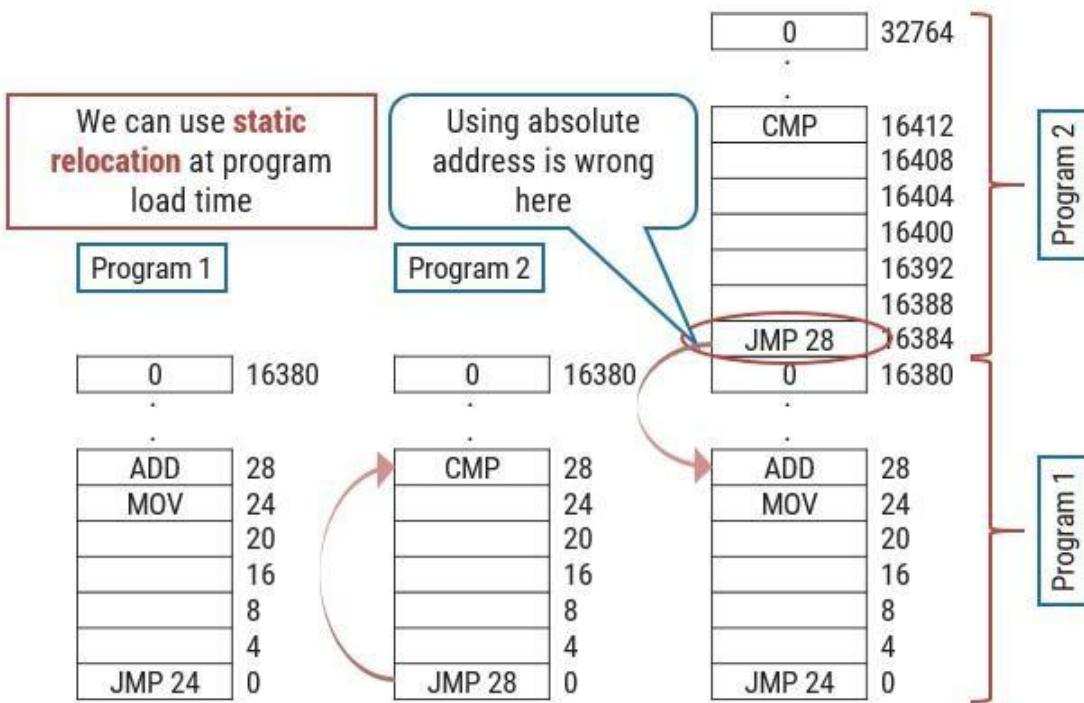
- › Here, memory is **shared among operating system and various simultaneously running processes.**
- › Initially, the **entire available memory is treated as a single free partition.**
- › **Process is allocated exactly as much memory as it requires.**
- › Whenever any **process enters** in a system, a **chunk of free memory big enough to fit the process is found and allocated.** The **remaining unoccupied space is treated as another free partition.**
- › If enough **free memory is not available to fit the process, process needs to wait until required memory becomes available.**
- › Whenever any **process gets terminate**, it **releases the space occupied.** If the **released free space is contiguous to another free partition, both the free partitions are merged together in to single free partition.**
- › **Better utilization** of memory than fixed sized size partition.



Multiprogramming with dynamic partitions



Multiprogramming without memory abstraction



Static relocation

When program was loaded at address 16384, the constant 16384 was added to every program address during the load process.

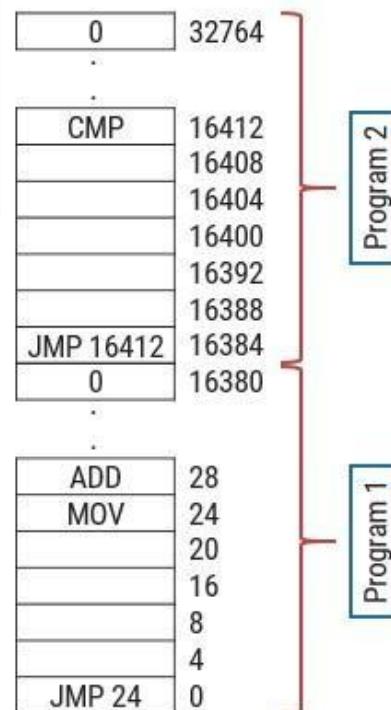
- Slow
- Required extra information from program

Program 1

0	16380
.	.
ADD	28
MOV	24
	20
	16
	8
	4
JMP 24	0

Program 2

0	16380
.	.
CMP	28
	24
	20
	16
	8
	4
JMP 28	0



Program 1



Logical and Physical address map

- An address space is set of addresses that a process can use to address memory.
- An address space is a range of valid addresses in memory that are available for a program or process.
- Two registers: **Base** and **Limit**

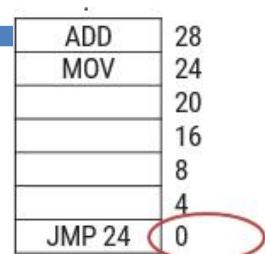
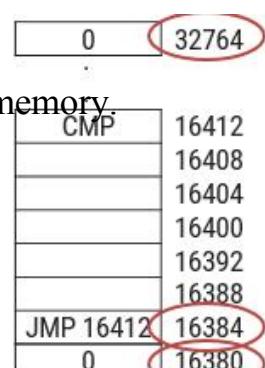
Base register: Start address of a program in physical memory.

Limit register: Length of the program.

- For every memory access

Base is added to the address

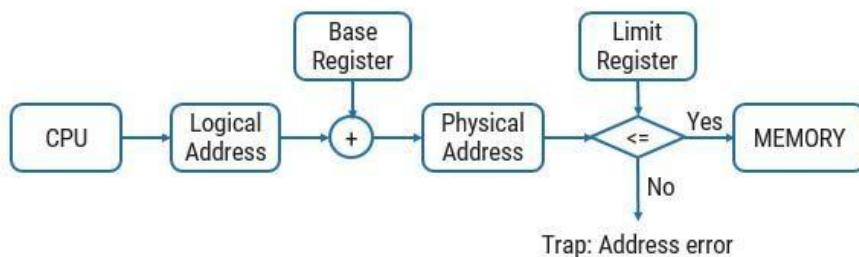
Result compared to Limit



- Only OS can modify Base and Limit register.

Dynamic relocation

- ▶ Dynamic relocation refers to **address transformations being done during execution of a program.**
- ▶ Steps in dynamic relocation
 1. Hardware **adds relocation register (base)** to virtual address to get a physical address
 2. Hardware **compares address with limit register**; address **must be less than or equal limit**
 3. If test fails, the **processor takes an address trap** and **ignores the physical address.**



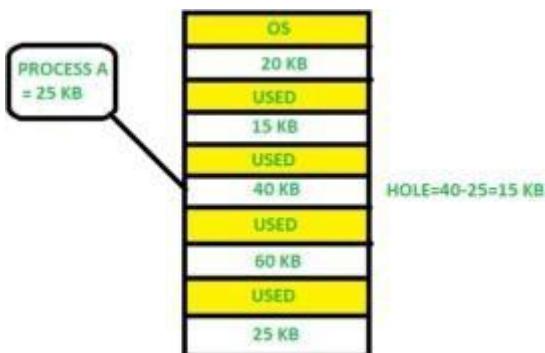
Memory allocation

- ▶ Four memory allocation algorithms are as follow
 9. First fit
 10. Next fit
 11. Best fit
 12. Worst fit

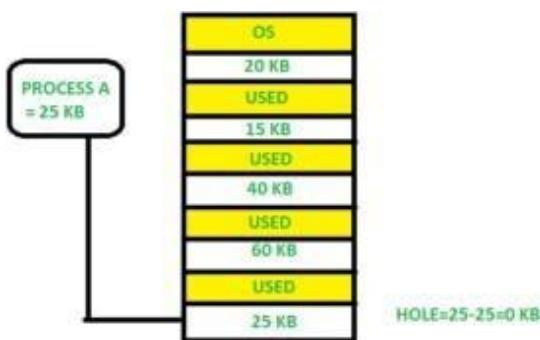
First fit

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 chooses the first available block that is large enough. Thus it allocates the first hole that is large

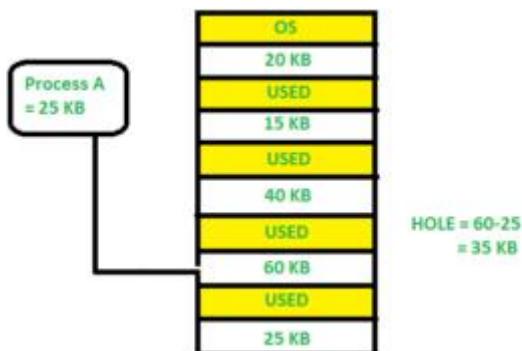
enough.



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 holes to find the smallest hole whose size is greater than or equal to the size of the process.



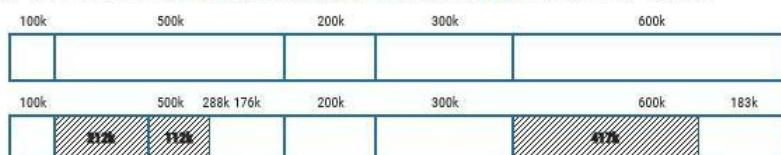
Worst Fit Allocate the process to the partition which is the largest sufficient among the freely available partitions available in the main memory. It is opposite to the best-fit algorithm. It searches the entire list of holes to find the largest hole and allocate it to process.



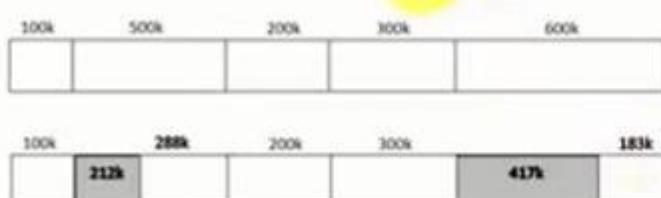
Next Fit: Next fit is similar to the first fit but it will search for the first sufficient partition from the last allocation point.

First fit

- ▶ Search starts from the starting location of the memory.
- ▶ First available hole that is large enough to hold the process is selected for allocation.
- ▶ The hole is then broken up into two pieces, one for process and another for unused memory.
- ▶ Example: Processes of **212K, 417K, 112K** and **426K** arrives in order.



- ▶ Here process of size **426k** will not get any partition for allocation.
- ▶ Fastest algorithm because it searches as little as possible.
- ▶ Memory loss is higher, as very large hole may be selected for small process.
- Example: Processes of **212K, 417K, 112K** and **426K** arrives in order.

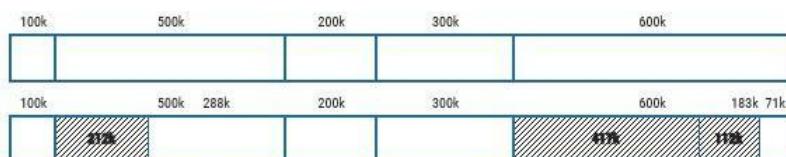




- Here process of size 426k will not get any partition for allocation.

Next fit

- It works in the same way as first fit, except that **it keeps the track of where it is whenever it finds a suitable hole**.
- The **next time when it is called to find a hole, it starts searching the list from the place where it left off last time**.
- Example: Processes of **212K, 417K, 112K** and **426K** arrives in order.



- Here process of size **426k will not get any partition** for allocation.
- Search time is smaller.**
- Memory manager must have to **keep track of last allotted hole to process**.
- It **gives slightly worse performance** than first fit.

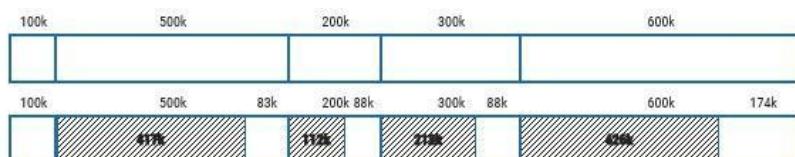
- Processes of **212K, 417K, 112K and 426K** arrives in order.



- Here process of size 426k will not get any partition for allocation.

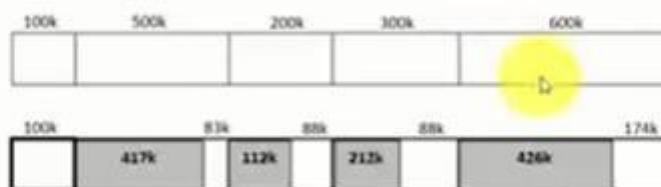
Best fit

- ▶ Entire memory is searched here.
- ▶ The **smallest hole, which is large enough to hold the process, is selected for allocation.**
- ▶ Example: Processes of **212K, 417K, 112K** and **426K** arrives in order.



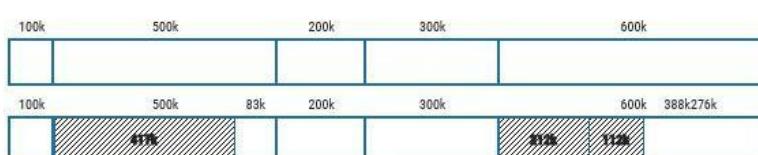
- ▶ Search time is high, as it searches entire memory every time.
- ▶ Memory loss is less.

- Processes of **212K, 417K, 112K and 426K** arrives in order.



Worst fit

- ▶ Entire memory is searched here also. The **largest hole, which is largest enough to hold the process, is selected for allocation.**
- ▶ Example: Processes of **212K, 417K, 112K** and **426K** arrives in order.

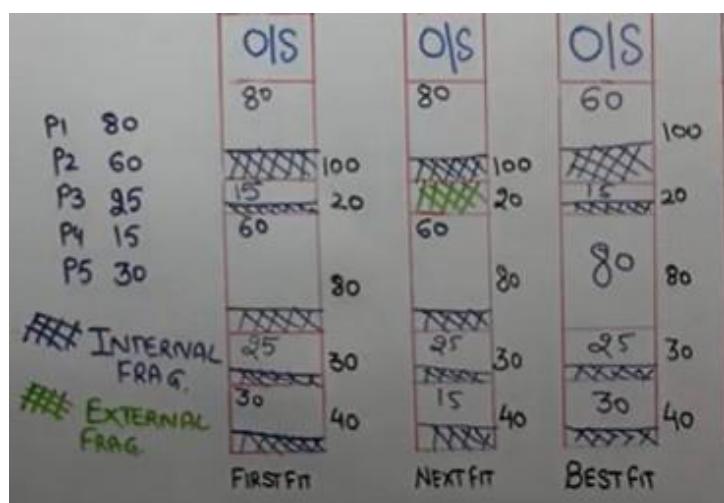
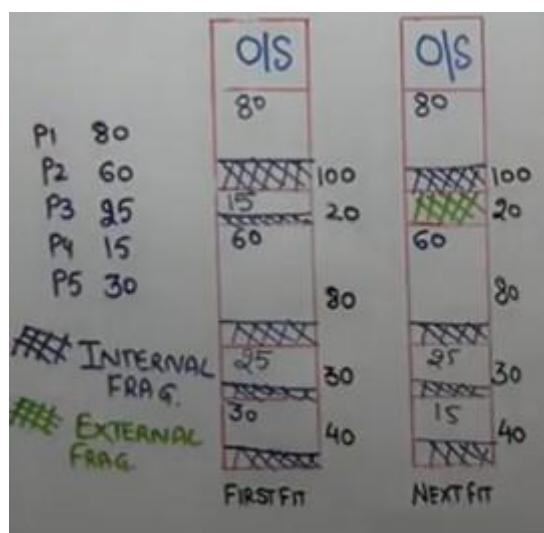
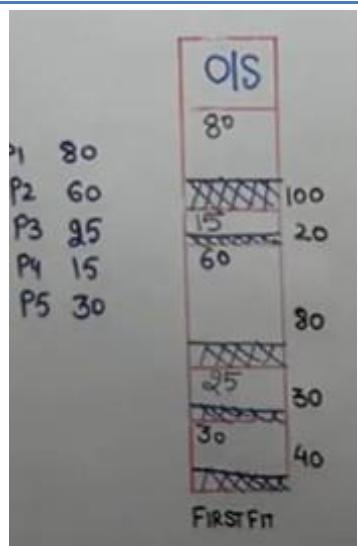


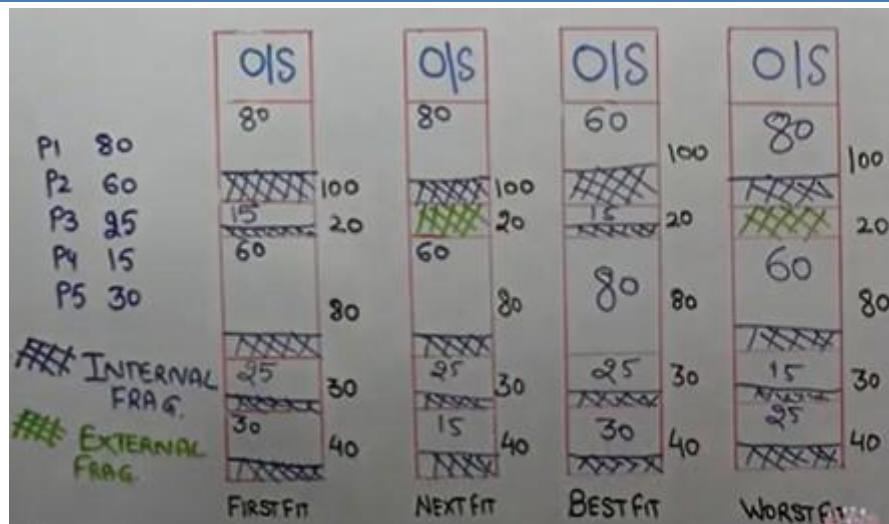
- ▶ Here process of size **426k will not get any partition** for allocation.
- ▶ Search time is high, as it searches entire memory every time.
- ▶ This algorithm can be used only with dynamic partitioning.



- Here process of size 426k will not get any partition for allocation.

	FIRST FIT	NEXT FIT	BEST FIT	WORST FIT
P1 80	80			
P2 60		60		
P3 95			95	
P4 15				15
P5 30				30
	100	100	100	100
	20	20	20	20
	30	30	30	30
	30	30	30	30
	40	40	40	40





1) FIRST FIT :- BEGIN TO SCAN MEMORY FROM START AND CHOOSE THE FIRST AVAILABLE Block SUITE ENOUGH TO ACCOMODATE PROCESS.

2) NEXT FIT : BEGINS AS FIRST FIT, WHEN CALLED NEXT TIME, IT STARTS SEARCHING WHERE IT LEFTS OFF.

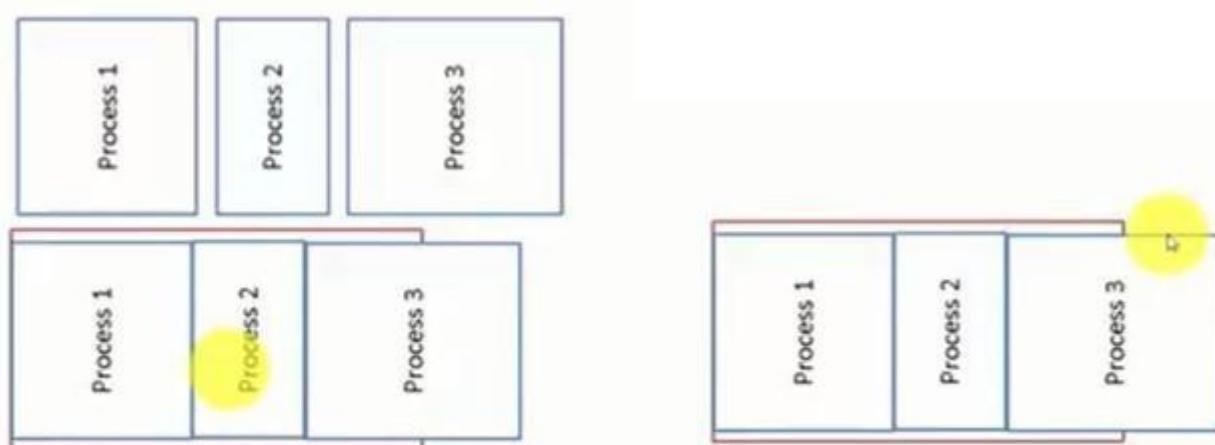
3) BEST FIT : SEARCH WHOLE PARTITION & SELECT BLOCK THAT IS CLOSEST IN SIZE TO THE REQUEST MEMORY.

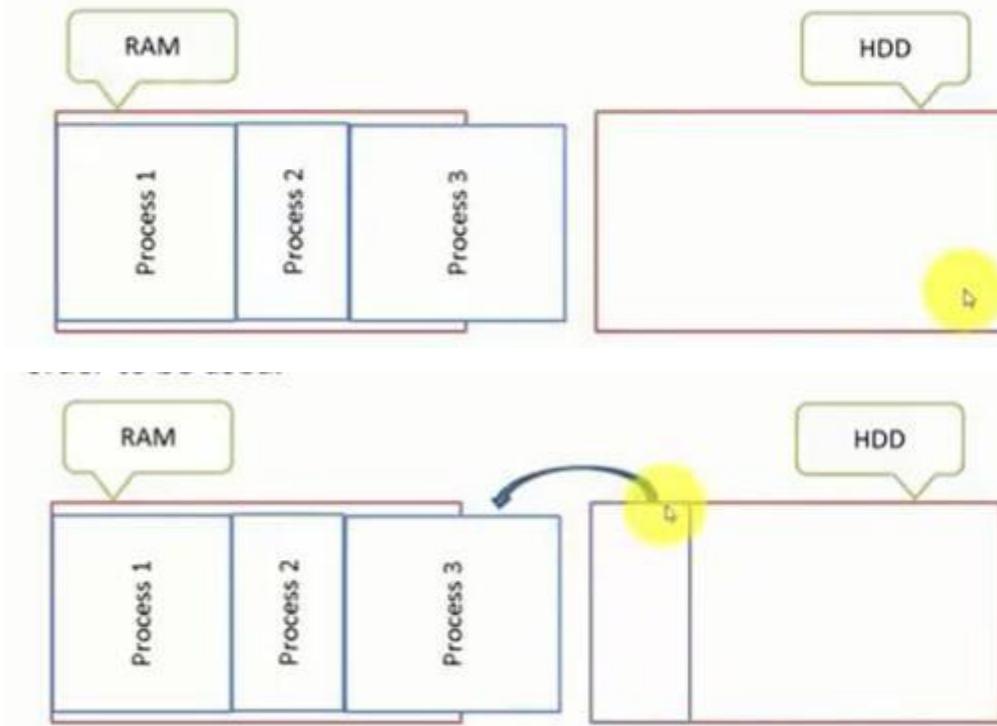
4) WORST FIT : SEARCH WHOLE PARTITIONS & SELECT THAT BLOCK THAT IS BIGGEST IN SIZE AND CAN ACCOMODATE THE PROCESS.



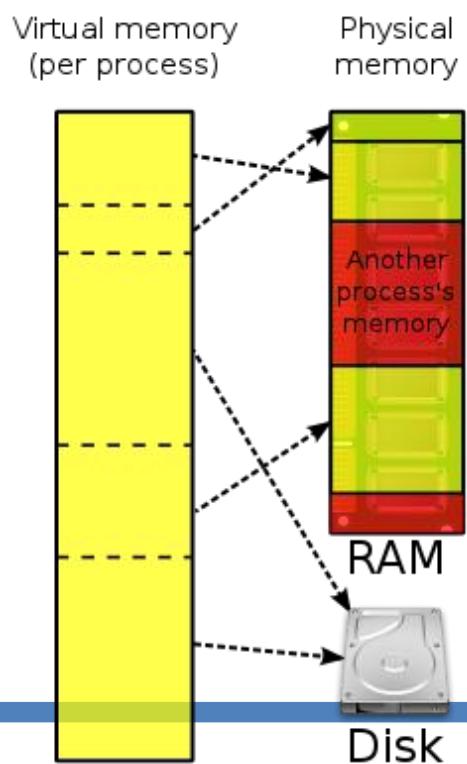
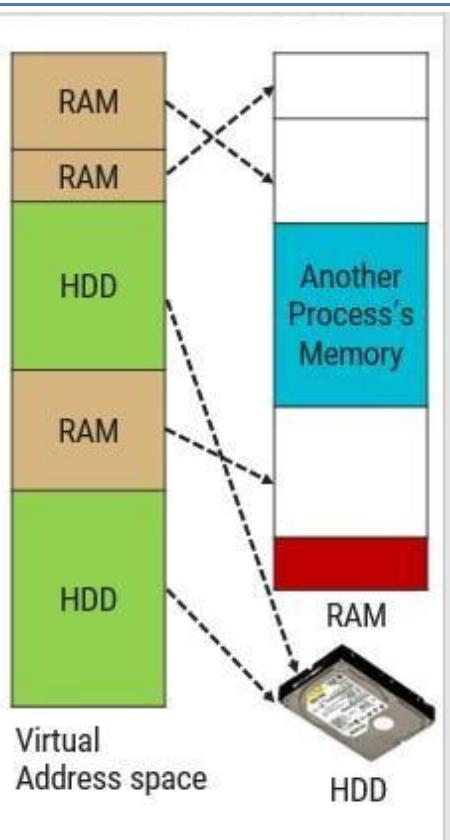
Virtual memory

- › Memory is **hardware that your computer uses to load the operating system and run programs.**
- › Computer **consists of one or more RAM chips** that each have several memory modules.
- › The amount of real **memory in a computer is limited** to the amount of RAM installed. Common memory sizes are **1GB, 2GB, and 4GB.**
- › Because your computer has a finite amount of RAM, **it is possible to run out of memory when too many programs are running at one time.**
- › This is where **virtual memory comes into the picture.**
- › Virtual memory **increases the available memory of your computer by enlarging the "address space,"** or places in memory where data can be stored.
- › It does this by **using hard disk space** for additional memory allocation.
- › However, since the **hard drive is much slower than the RAM,** data stored in **virtual memory must be mapped back to real memory in order to be used.**





- › Each program has its own address space, which is **broken up into pages**.
- › Each **page is a contiguous range of addresses**.
- › These **pages are mapped onto the physical memory** but, **to run the program, all pages are not required** to be present in the physical memory.
- › The operating system **keeps those parts of the program currently in use in main memory**, and the rest on the disk.
- › In a system using virtual memory, the **physical memory is divided into page frames** and the **virtual address space is divided into equally-sized partitions called pages**.
- › Virtual memory **works fine in a multiprogramming system**, with bits and pieces of many programs in memory at once.



What is Paging in OS?

- **Paging** is a storage mechanism that allows OS to retrieve processes from the secondary storage into the main memory in the form of pages.
- In the Paging method, the main memory is divided into small fixed-size blocks of physical memory, which is called frames.
- The size of a frame should be kept the same as that of a page to have maximum utilization of the main memory and to avoid external fragmentation.

Paging is used for faster access to data, and it is a logical concept.

- Logical Address or Virtual Address (represented in bits): An address generated by the CPU
- Logical Address Space or Virtual Address Space (represented in words or bytes): The set of all logical addresses generated by a program
- Physical Address (represented in bits): An address actually available on memory unit
- Physical Address Space (represented in words or bytes): The set of all physical addresses corresponding to the logical addresses

The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique.

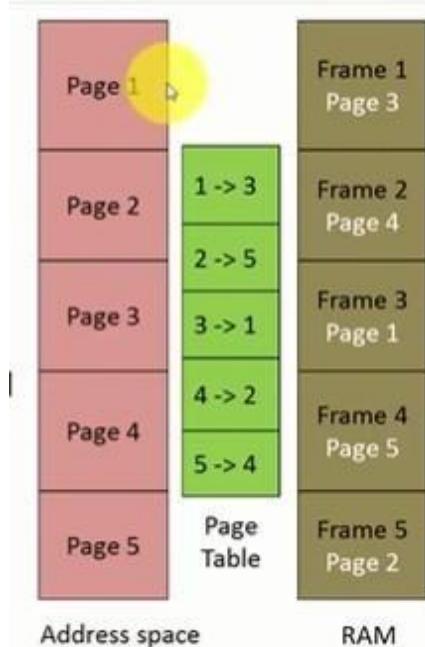
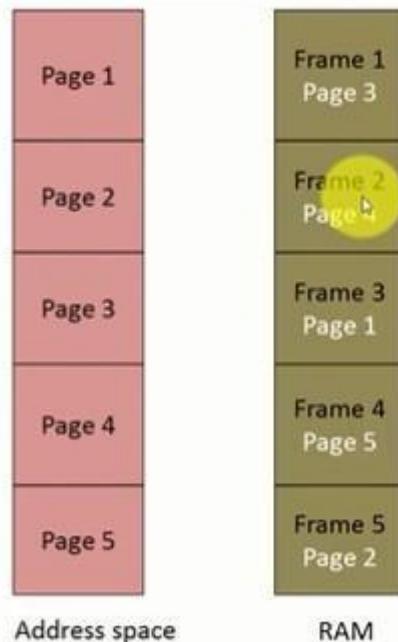
- The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.
- The Logical address Space is also splitted into fixed-size blocks, called **pages**.
- Page Size = Frame Size
- Address generated by CPU is divided into **Page number(p)**: Number of bits required to represent the pages in Logical Address Space or Page number
- **Page offset(d)**: Number of bits required to represent particular word in a page or page size of Logical Address Space or word number of a page or page offset.

Physical Address is divided into

- **Frame number(f)**: Number of bits required to represent the frame of Physical Address Space or Frame number.
- **Frame offset(d)**: Number of bits required to represent particular word in a frame or frame size of Physical Address Space or word number of a frame or frame offset.

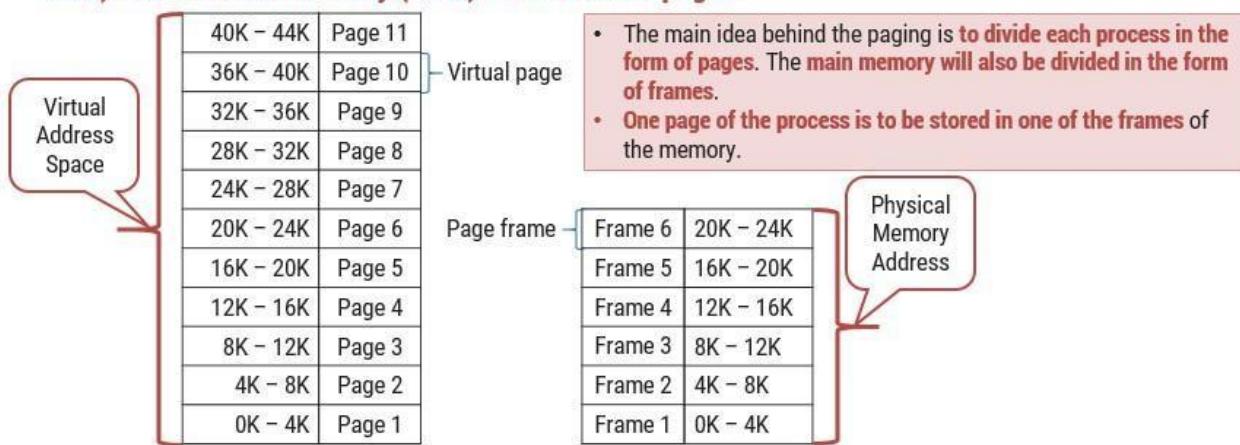
Paging

- The program generated address is called as **Logical Addresses** and form the Virtual Address Space.
- An address actually available on memory is called **Physical Address**.
- Virtual address space is divided into fixed-size partitions called pages.
- The corresponding units in the physical memory are called as page frames.
- The pages and page frames are always of the same size.
- (Page Size = Frame Size)



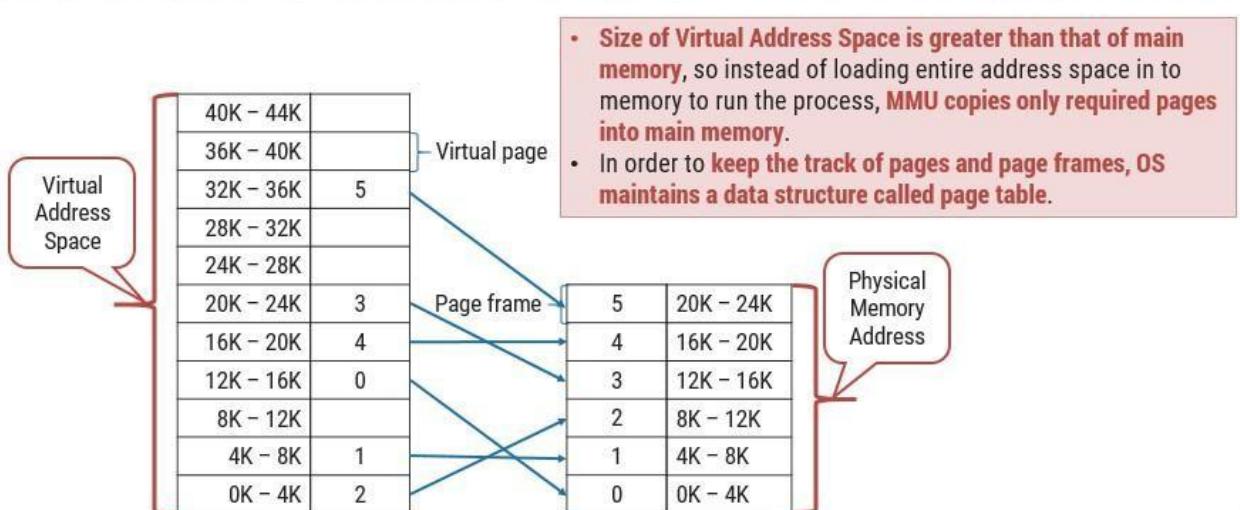
Paging

- Paging is a **storage mechanism used to retrieve processes from the secondary storage (Hard disk) into the main memory (RAM) in the form of pages.**



- Pages of the process are **brought into the main memory only when they are required** otherwise they reside in the secondary storage.
- The **sizes of each frame must be equal**. Considering the fact that the pages are mapped to the frames in Paging, page size needs to be as same as frame size.
- Different operating system defines different frame sizes.

Paging

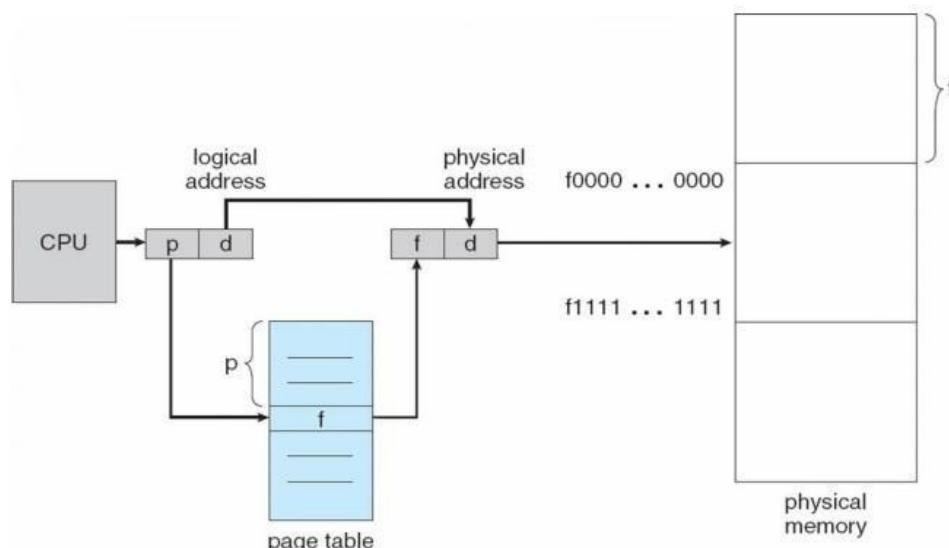


The hardware implementation of page table can be done by using dedicated registers.

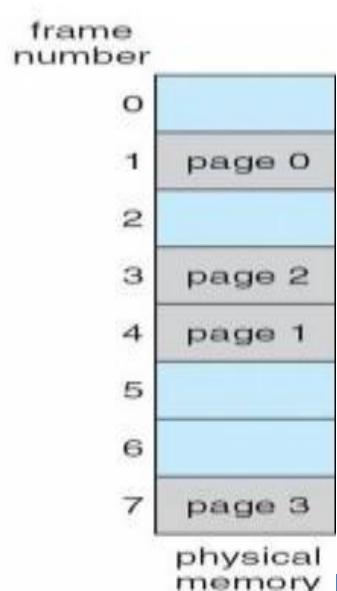
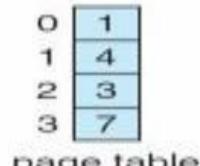
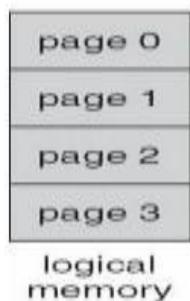
But the usage of register for the page table is satisfactory only if page table is small.

If page table contain large number of entries then we can use TLB(translation Look- aside buffer), a special, small, fast look up hardware cache.

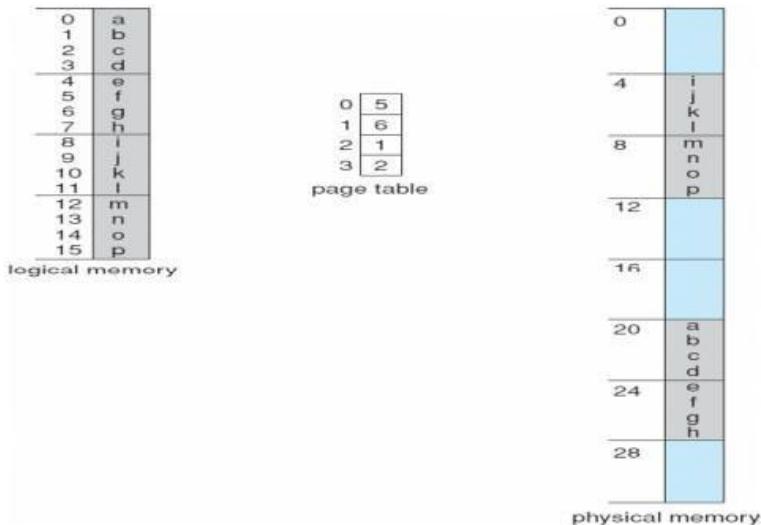
Paging Hardware



Paging Model of Logical and Physical Memory

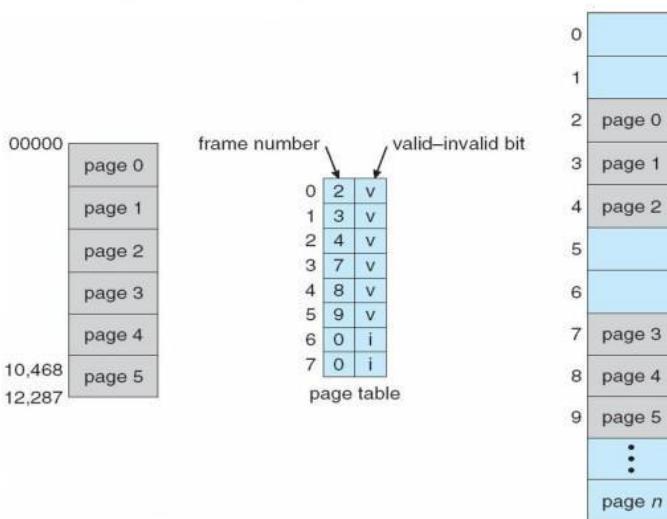


Paging Example



32-byte memory and 4-byte pages

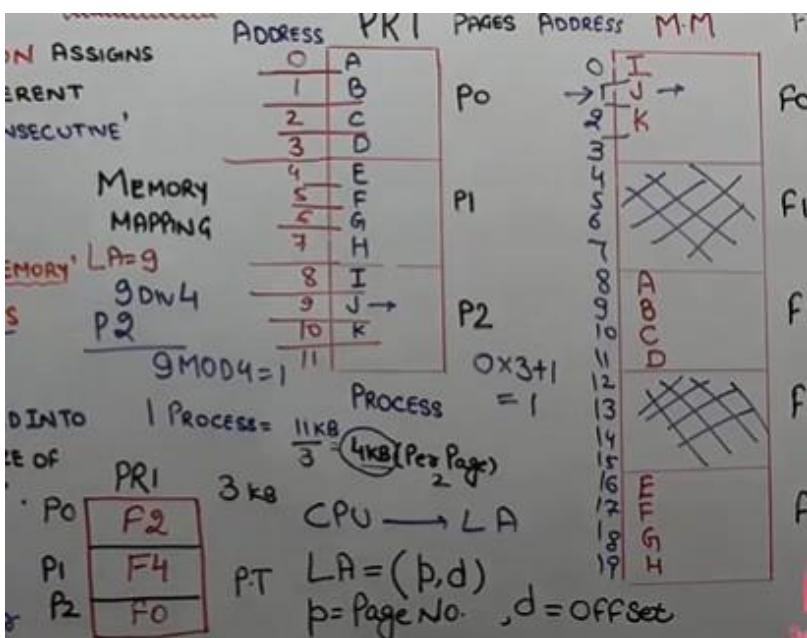
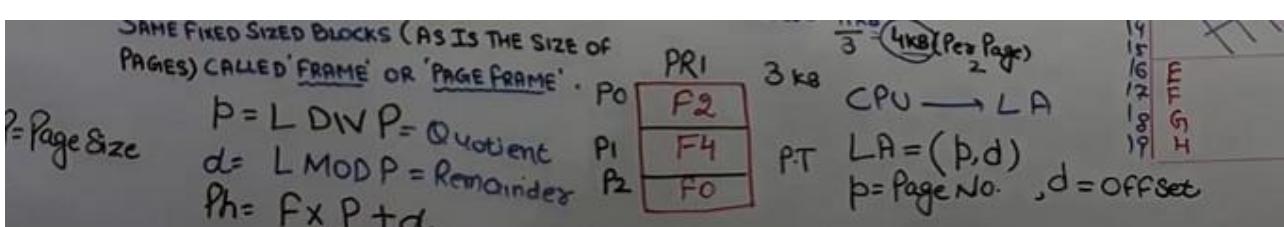
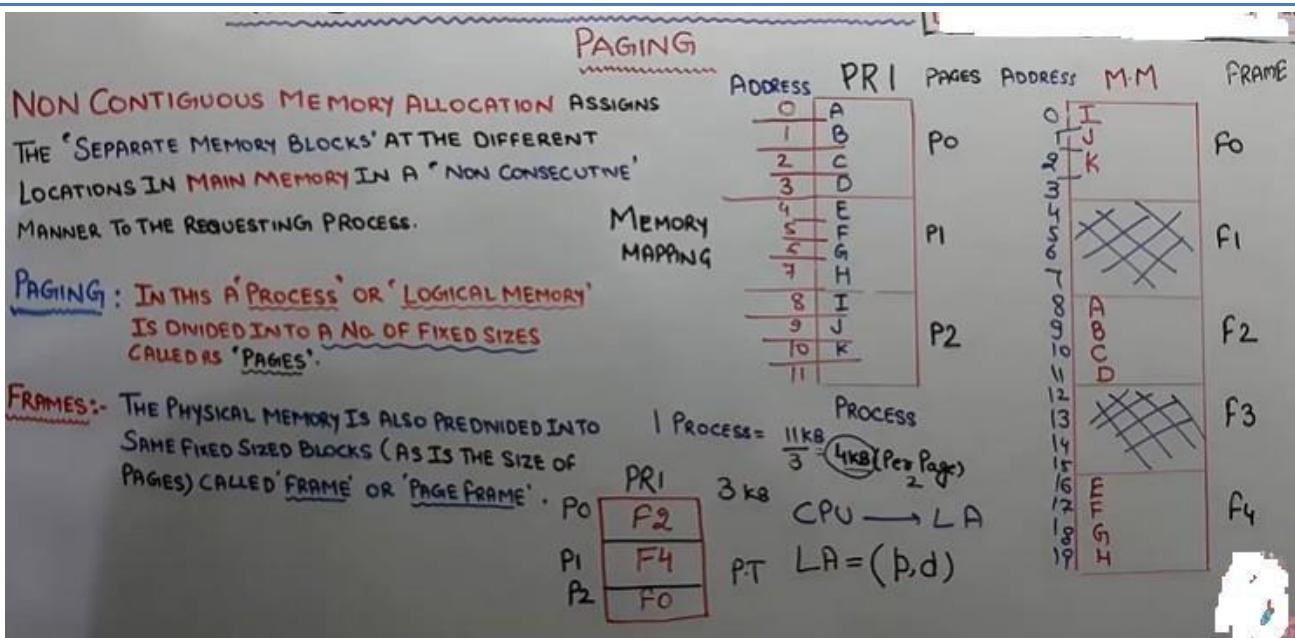
- “valid” indicates that the associated page is in the process’ logical address space, and is thus a legal page
- “invalid” indicates that the page is not in the process’ logical address space
- Valid (v) or Invalid (i) Bit In A Page Table**



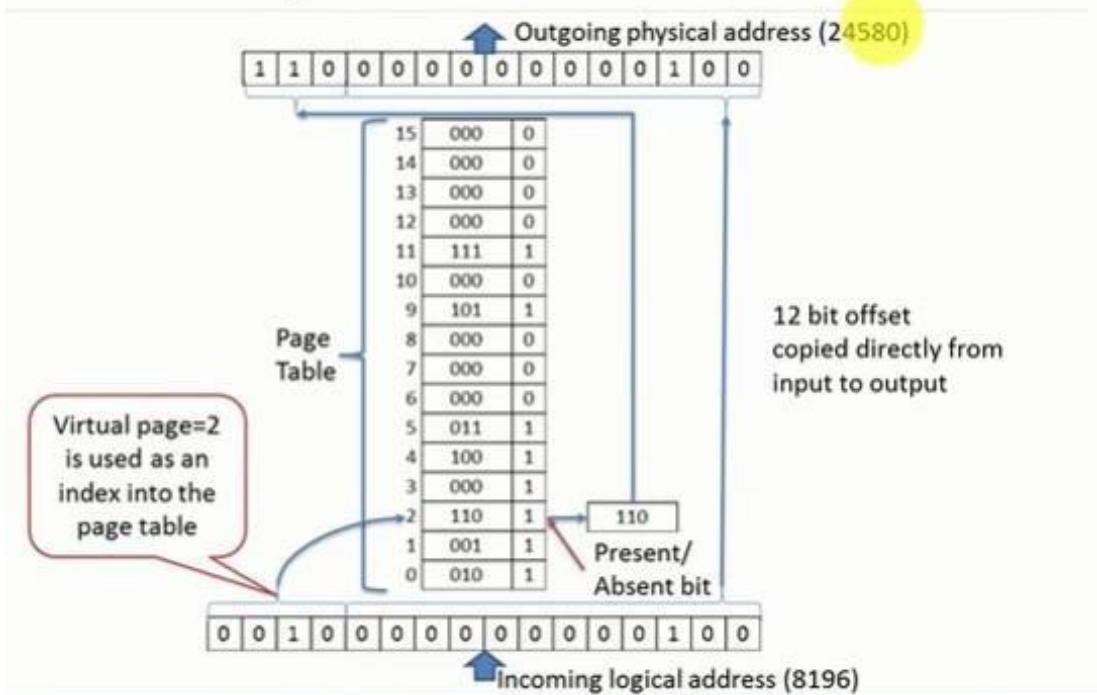
- **Logical Address is generated by CPU** while a program is running. The **logical address is virtual address as it does not exist physically** therefore it is also known as **Virtual Address**.
 - **Physical Address identifies a physical location of required data in a memory.** The user never directly deals with the physical address but can access by its corresponding logical address.
 - The **hardware device called Memory-Management Unit** is used for mapping (**converting**) logical address to its corresponding physical address.
-
- If the **present/absent bit is 0**, it is **page-fault**; a trap to the **operating system is caused to bring required page into main memory**.
 - If the **present/absent bit is 1**, required page is there with **main memory** and **page frame number found in the page table is copied to the higher order bit of the output register along with the offset**.
 - Together **Page frame number and offset creates physical address**.
 - Physical Address = Page frame Number + offset of virtual address.

Page table

- Page table is a **data structure which translates virtual address into equivalent physical address**.
- The virtual page number is used as an index into the Page table to find the entry for that virtual page and from the Page table physical page frame number is found.
- Thus the **purpose of page table is to map virtual pages onto page frames**.
- The page table of the process is **held in the kernel space**.



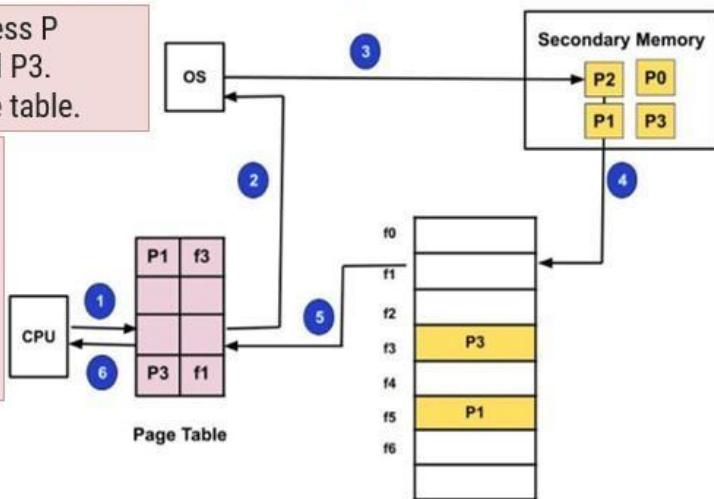
Internal operation of the MMU



Definitions (Demand paging)

► **Demand paging:** Demand paging is a **technique used in virtual memory** systems where the **pages** are brought in the **main memory** only when required or demanded by the CPU.

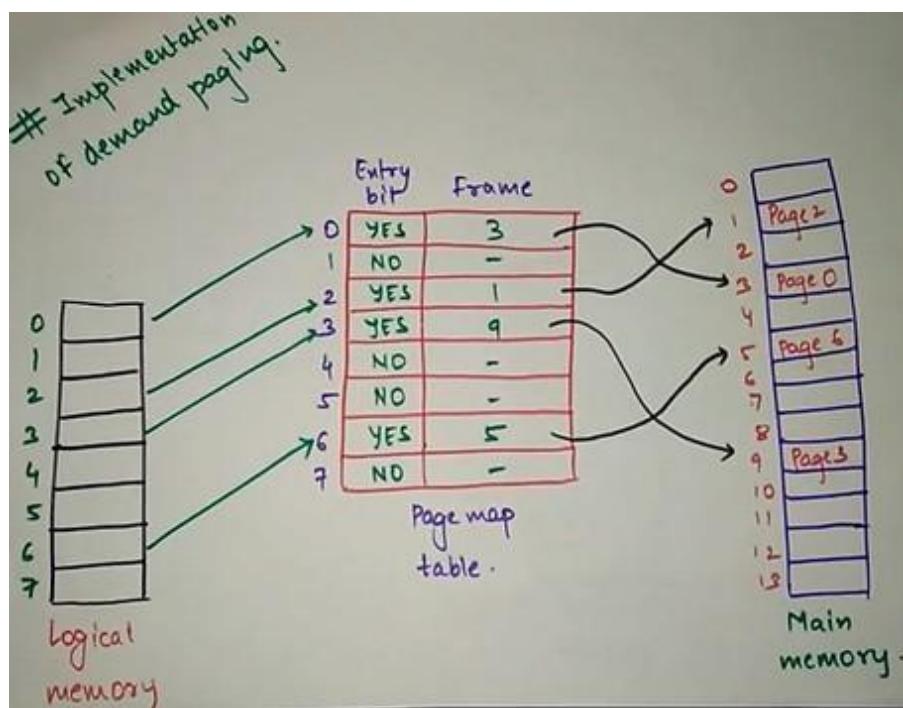
- Suppose we have to execute a process P having four pages as P0, P1, P2, and P3.
- We have page P1 and P3 in the page table.
- Now, if the CPU wants to access page P2 of a process P
 - search the page in the page table
 - P2 is not available in page table so page fault
 - OS bring P2 from HDD to RAM
 - OS will update page table
 - Process P will be executed.

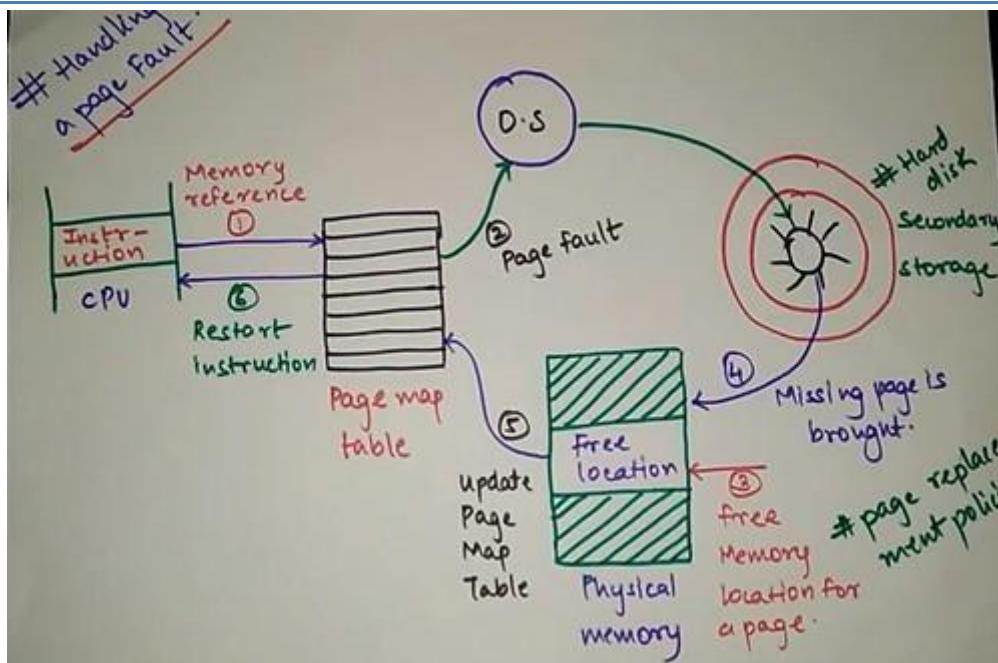


Advantages of Demand paging

- Advantages :

1. Not necessary to load an entire process.
2. Process can be larger than main memory size.
3. Few pages are present for a process, so we can accommodate multiple processes.
4. Time saving in case of unused page handling





Logical Address vs Physical Address

- ▶ Logical Address is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically therefore it is also known as Virtual Address.
- ▶ Physical Address identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address.
- ▶ The hardware device called Memory-Management Unit is used for mapping (converting) logical address to its corresponding physical address.

Page Replacement Algorithms

- › Following are different types of page replacement algorithms
 1. Optimal Page Replacement Algorithm
 2. FIFO Page Replacement Algorithm
 3. The Second Chance Page Replacement Algorithm
 4. The Clock Page Replacement Algorithm
 5. LRU (Least Recently Used) Page Replacement Algorithm
 6. NRU (Not Recently Used)

Optimal Page Replacement Algorithm

- › The moment a page fault occurs, some set of pages will be in the memory.
- › One of these pages will be referenced on the very next instruction.
- › Other pages may not be referenced until 10, 100, or perhaps 1000 instructions later.
- › Each **page can be labeled with the number of instructions that will be executed before that page is first referenced.**
- › The optimal page algorithm simply says that the **page with the highest label should be removed.**
- › The only problem with this algorithm is that it is **unrealizable.**
- › At the time of the page fault, the **operating system has no way of knowing when each of the pages will be referenced next.**

Optimal Page Replacement Algorithm

► Page Reference String:

- 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 0, 2, 0, 1, 7, 0, 1
- Three frames

Page Request	7	0	1	2	0	3	0	4	2	3	0	3	2	0	2	0	1	7	0	1
Frame - 1	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1
Frame - 2	0	0	0	0	0	0	4	4	4	4	0	0	0	0	0	0	0	0	0	0
Frame - 3		1	1	1	3	3	3	3	3	3	3	3	3	3	3	3	3	7	7	7
Page Faults (9)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F

FIFO Page Replacement Algorithm

- ▶ The first in first out page replacement algorithm is the simplest page replacement algorithm.
- ▶ The operating system **maintains a list of all pages currently in memory, with the most recently arrived page at the tail and least recent at the head.**
- ▶ On a page fault, the **page at head is removed and the new page is added to the tail.**
- ▶ When a page replacement is **required the oldest page in memory needs to be replaced.**
- ▶ The performance of the FIFO algorithm is **not always good because it may happen that the page which is the oldest is frequently referred by OS.**
- ▶ Hence removing the **oldest page may create page fault again.**

FIFO Page Replacement Algorithm

- ▶ Page Reference String:
 - 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
 - Three frames

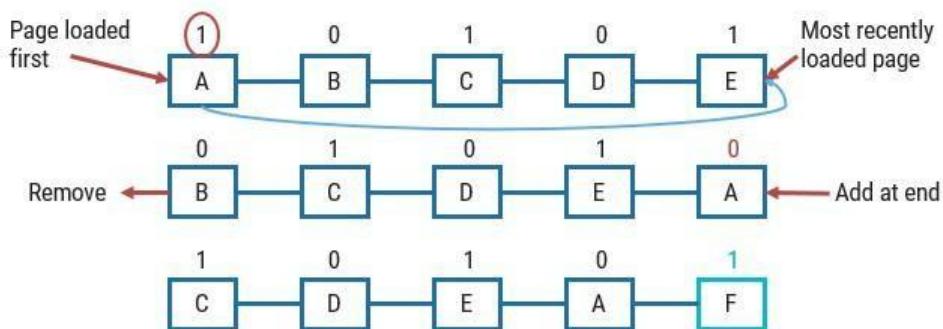
Page Request	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
Frame - 1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
Frame - 2	0	0	0	0	3	3	3	2	2	2	2	2	2	1	1	1	1	1	0	0
Frame - 3		1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1	
Page Faults (15)	F	F	F	F		F	F	F	F	F			F	F			F	F	F	

FIFO Page Replacement Algorithm

- ▶ The first in first out page replacement algorithm is the simplest page replacement algorithm.
- ▶ The operating system **maintains a list of all pages currently in memory, with the most recently arrived page at the tail and least recent at the head.**
- ▶ On a page fault, the **page at head is removed and the new page is added to the tail.**
- ▶ When a page replacement is **required the oldest page in memory needs to be replaced.**
- ▶ The performance of the FIFO algorithm is **not always good because it may happen that the page which is the oldest is frequently referred by OS.**
- ▶ Hence removing the **oldest page may create page fault again.**

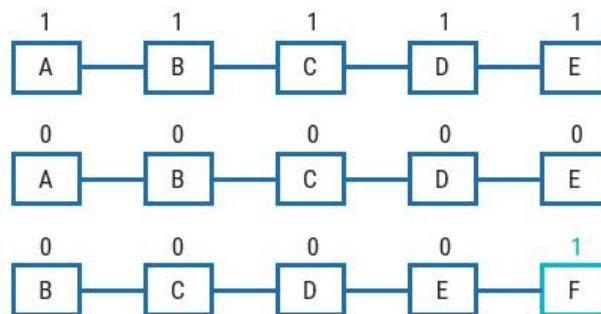
Second Chance Page Replacement Algorithm

- ▶ It is **modified form of the FIFO page replacement algorithm**.
- ▶ It looks at the front of the queue as FIFO does, but instead of immediately paging out that page, it **checks to see if its referenced bit is set**.
 - If it is not set (zero), the page is swapped out.
 - Otherwise, the referenced bit is cleared, the page is inserted at the back of the queue (as if it were a new page) and this process is repeated.



Second Chance Page Replacement Algorithm

- ▶ If all the pages have their referenced bit set, on the second encounter of the first page in the list, that page will be swapped out, as it now has its referenced bit cleared.
- ▶ If all the pages have their reference bit cleared, then second chance algorithm degenerates into pure FIFO.



Disadvantages of SC

- ▶ In second chance algorithm **pages are constantly moving around on its list**. So it is **not working efficiently**.
- ▶ A better approach is to **keep all the page frames on a circular list in the form of a clock**.

LRU (Least Recently Used) Page Replacement Algorithm

- ▶ A good approximation to the optimal algorithm is based on the observation that pages that have been heavily used in last few instructions will probably be heavily used again in next few instructions.
- ▶ When page fault occurs, **throw out the page that has been used for the longest time**. This strategy is called LRU (Least Recently Used) paging.
- ▶ To fully implement LRU, it is necessary to maintain a linked list of all pages in memory, with the **most recently used page at the front and the least recently used page at the rear**.
- ▶ The list must be updated on every memory reference.
- ▶ Finding a page in the list, deleting it, and then moving it to the front is a very time consuming operations.

LRU (Least Recently Used) Page Replacement Algorithm

- ▶ Page Reference String:
 - 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
 - Three frames

Page Request	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
Frame – 1	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
Frame – 2	0	0	0	0	0	0	0	0	3	3	3	3	3	3	3	0	0	0	0	0
Frame – 3		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7	7
Page Faults (12)	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F				

NRU (Not Recently Used) Page Replacement Algorithm

- ▶ NRU makes approximation to replace the page based **on R (referenced) and M (modified) bits.**
- ▶ When a process is started up, **both page bits for all pages are set to 0 by operating system.**
- ▶ Periodically, the **R bit is cleared**, to distinguish pages that have not been referenced recently from those that have been.
- ▶ When page fault occurs, the operating system inspects all the pages and **divide them into 4 categories based on current values of their R and M bits**
 - Case 0 : not referenced, not modified
 - Case 1 : not referenced, modified
 - Case 2 : referenced, not modified
 - Case 3 : referenced, modified
- ▶ The NRU (Not Recently Used) algorithm removes a page at random from the lowest numbered nonempty class.

NRU (Not Recently Used) Page Replacement Algorithm

- ▶ For example if,
 - Page-0 is of class-2 (referenced, not modified)
 - Page-1 is of class-1 (not referenced, modified)
 - Page-2 is of class-0 (not referenced, not modified)
 - Page-3 is of class-3 (referenced, modified)
- ▶ So lowest class **page-2 needs to be replaced by NRU.**

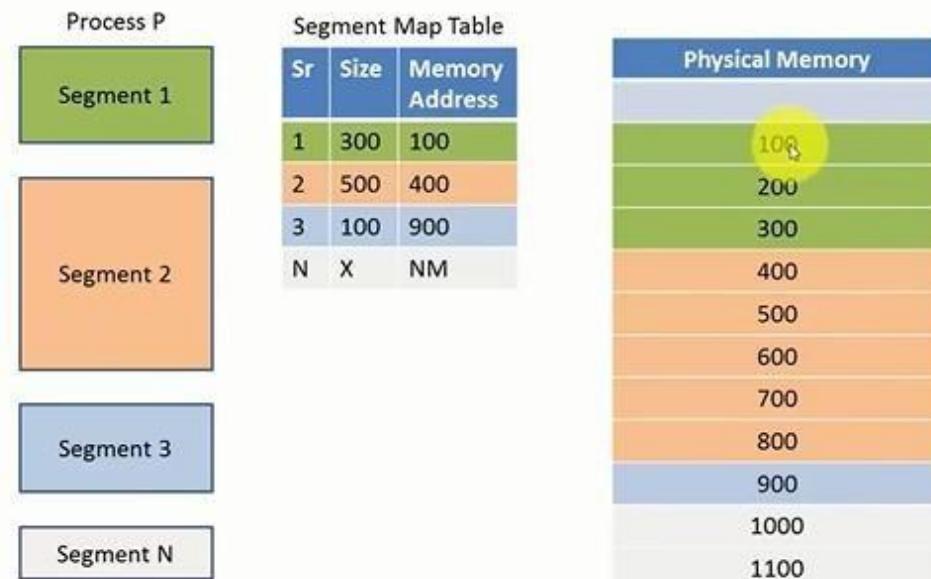
Segmentation

- ▶ Segmentation is a **memory management technique in which each job is divided into several segments of different sizes**, one for each module that contains pieces that perform related functions.
- ▶ **Each segment is actually a different logical address space of the program.**
- ▶ **When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory** though every segment is loaded into a contiguous block of available memory.
- ▶ Segmentation memory management works very similar to paging but here **segments are of variable-length where as in paging pages are of fixed size.**

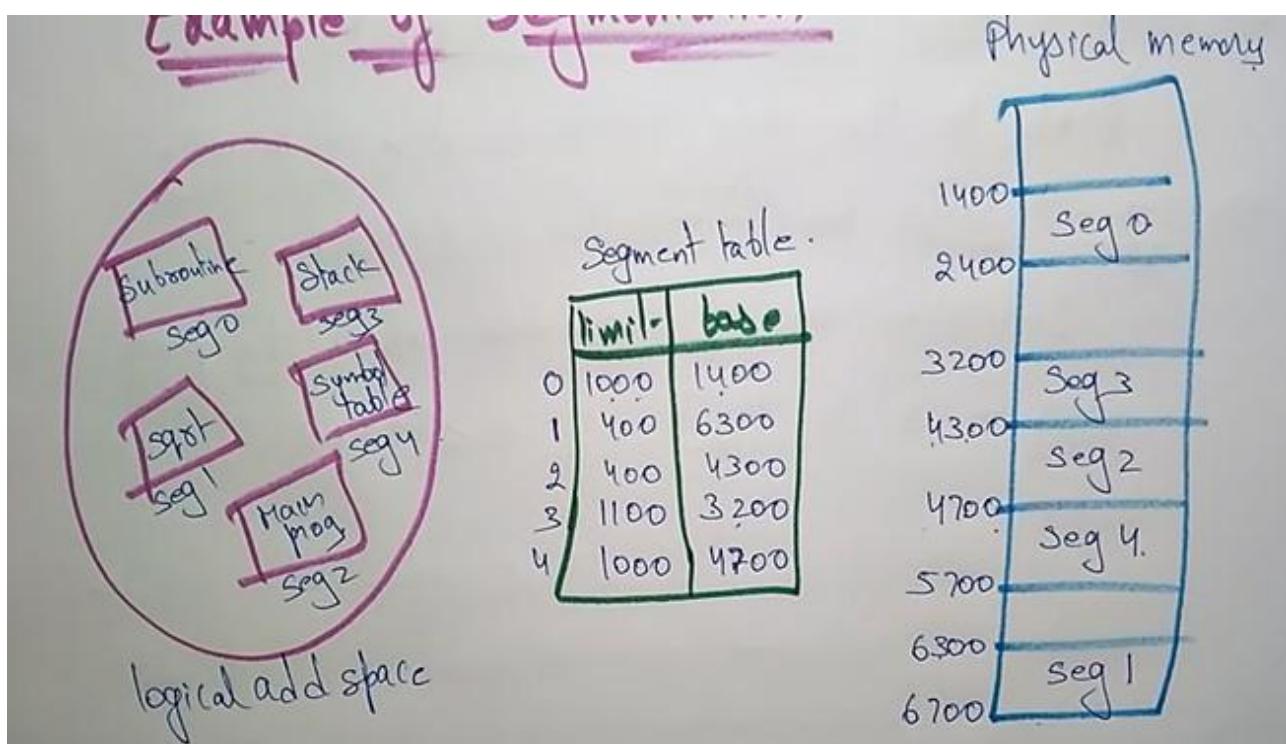
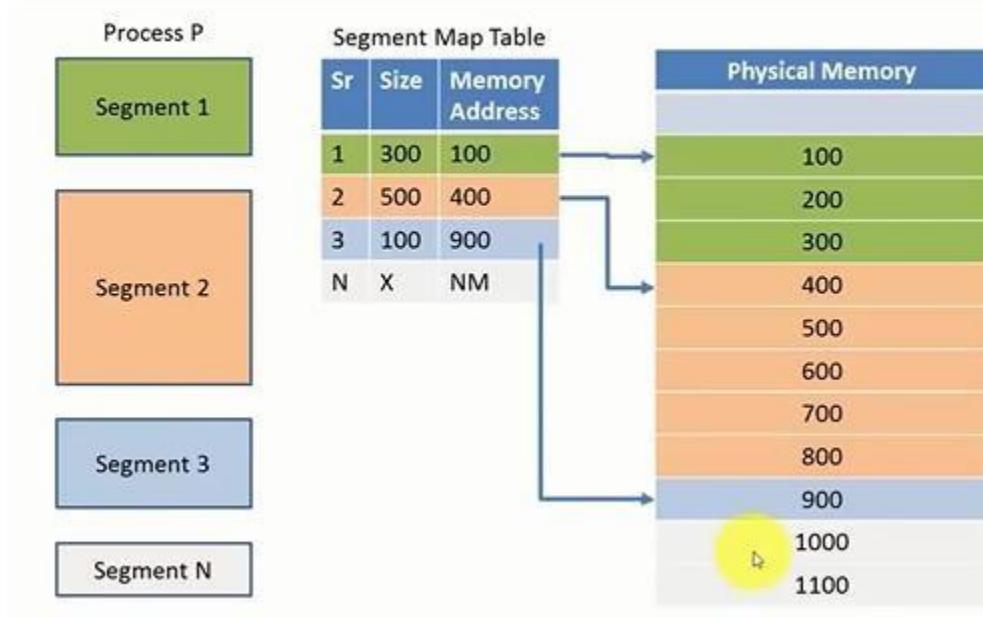
Segmentation

- ▶ A program segment contains the program's main function, utility functions, data structures, and so on.
- ▶ The **operating system maintains a segment map table for every process**.
- ▶ **Segment map table contains list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory.**
- ▶ For each segment, the **table stores the starting address of the segment and the length of the segment.**
- ▶ A reference to a memory location includes a value that identifies a **segment and an offset**

Segmentation



Segmentation



1. Consider (70120304230321201701) page reference string: How many page fault would occur for following page replacement algorithm. Consider 3 frames and 4 frames.
1. FIFO 2. LRU 3. Optimal

Belady's Anomaly (in FIFO Page Replacement Algorithm)

- Belady's anomaly is the **phenomenon in which increasing the number of page frames results in an increase in the number of page faults for certain memory access patterns.**
- This phenomenon is commonly **experienced when using the First-In First-Out (FIFO) page replacement algorithm.**
- In FIFO, the page fault may or may not increase as the page frames increase, but in Optimal and stack-based algorithms like LRU, as the page frames increase the page fault decreases.

Belady's Anomaly (in FIFO Page Replacement Algorithm)

► Page Reference String: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5 **Page Faults of 4 Frame > Page Faults of 3 Frame**

Three Frames	Page Request	1	2	3	4	1	2	5	1	2	3	4	5
	Frame - 1	1	1	1	4	4	4	5	5	5	5	5	5
Frame - 2		2	2	2	1	1	1	1	1	1	3	3	3
Frame - 3			3	3	3	2	2	2	2	2	2	4	4
Page Faults (9)	F	F	F	F	F	F	F			F	F		

Four Frames	Page Request	1	2	3	4	1	2	5	1	2	3	4	5
	Frame - 1	1	1	1	1	1	1	5	5	5	5	4	4
Frame - 2		2	2	2	2	2	2	2	1	1	1	1	5
Frame - 3			3	3	3	3	3	3	3	2	2	2	2
Frame - 4				4	4	4	4	4	4	4	3	3	3
Page Faults (10)	F	F	F	F				F	F	F	F	F	F