Unit : 3

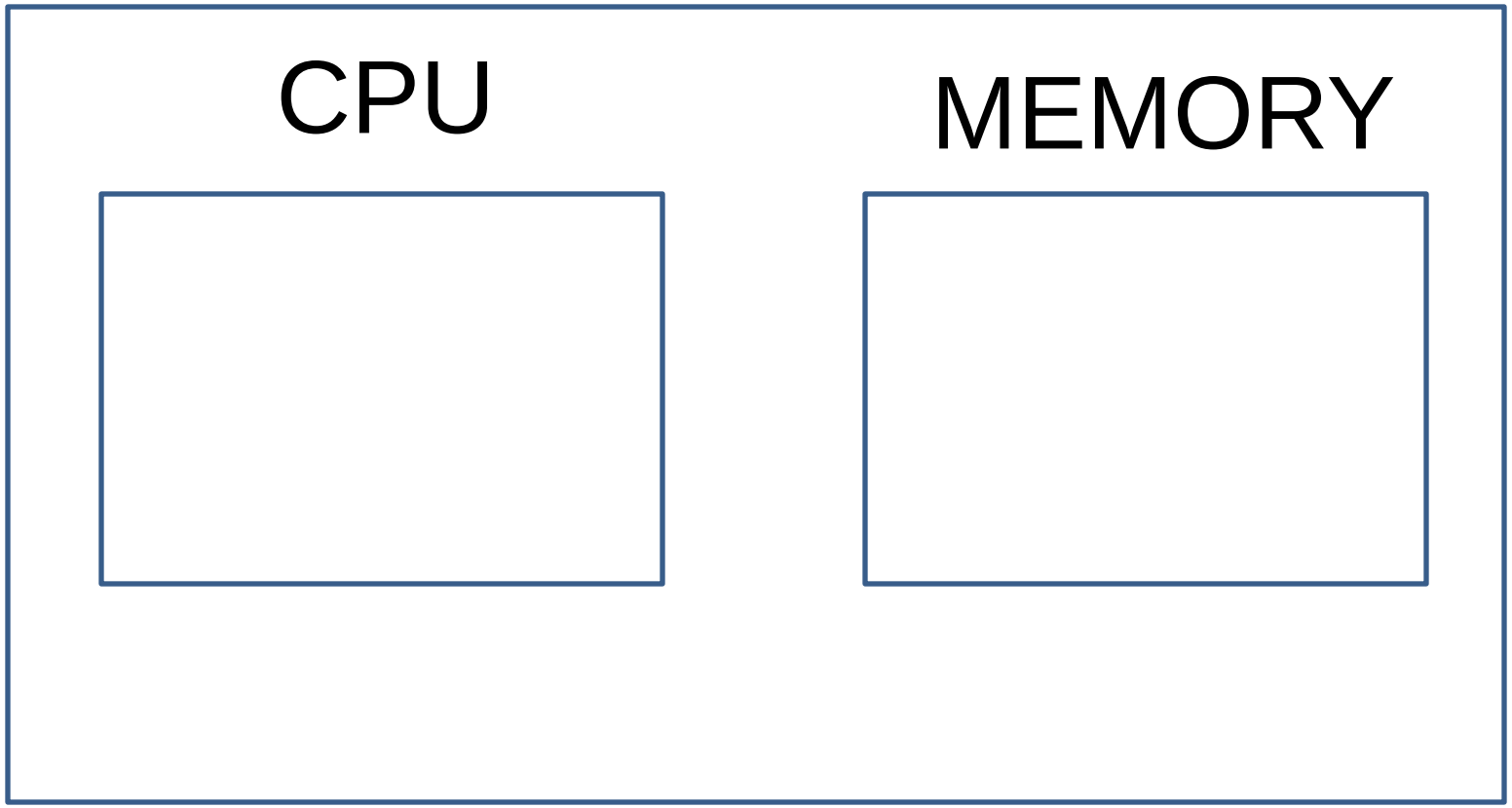
Memory Management

A. K. Panchasara

Sr. Lecturer in Computer Engineering

AVPTI-Rajkot

Main Components of Computer

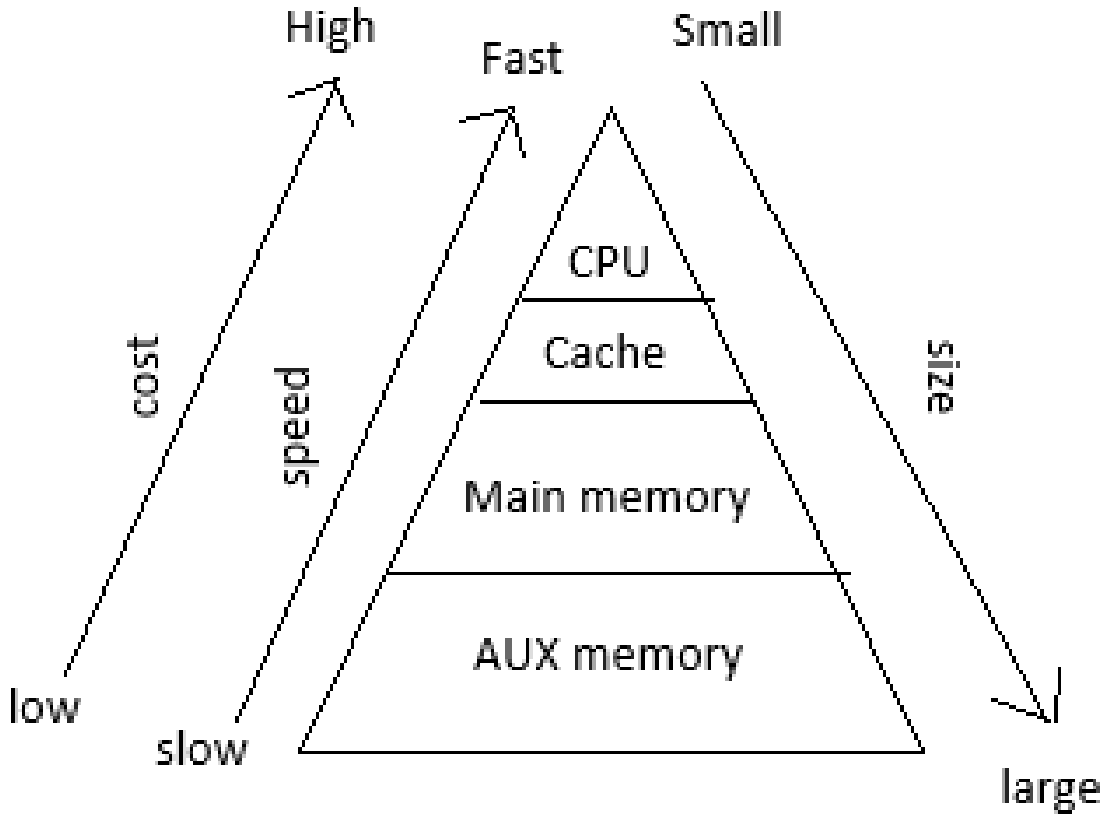


Expectations from Memory

1. Large in size (storage capacity)
2. Low access time (Fast)
3. Low Cost

Note: Size and Access time of memory are inversely proportional to each other.

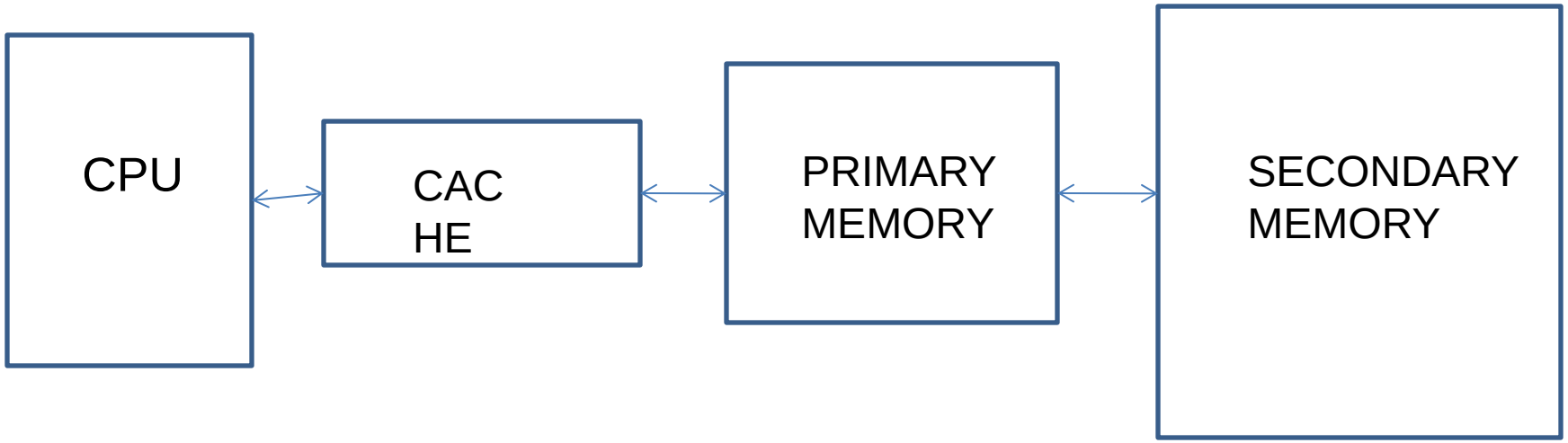
Memory Hierarchy



Comparison of Primary and Secondary Memory

Primary Memory	Secondary Memory
1. Directly accessible by CPU	1. Not directly accessible by CPU
2. Volatile in nature	2. Non- volatile
3. Access time is low	3. Access time is high
4. Cost is high	4. Cost is low
5. Also known as Main memory or Internal memory.	5. Also known as External memory or Auxiliary memory.
6. E.g. RAM and its variants	6. E.g. Hard Disk, Floppy Disk, Magnetic Tapes, etc.

Hierarchical connection of memory with CPU

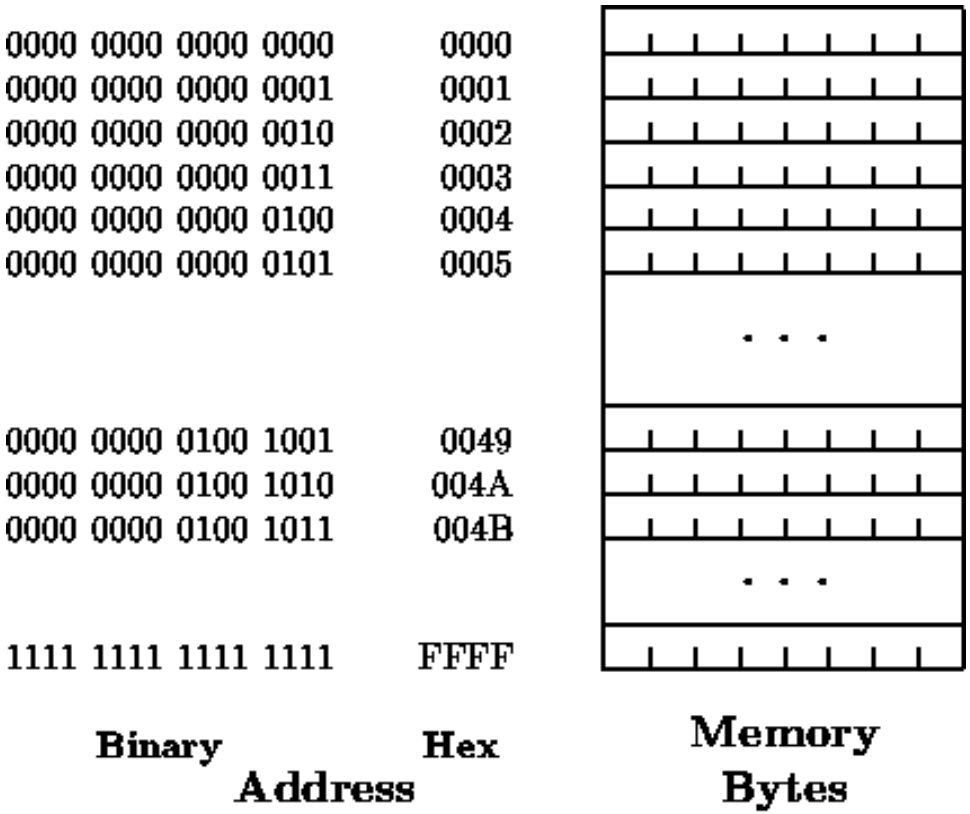


Introduction

- memory is second major component of any computer after processor.
- layers of memory (memory hierarchy)
- access info directly disks are slow and inefficient
- so, access from main memory is better than disk
- main memory is the physical memory and it is internal to the computer comes from chips
- it is slower and cheaper than cache memory and faster and costly than disks
- main memory is consists of RAM
- generally info stored on disks but access is much slower so, first it is brought into main memory then CPU can directly access it

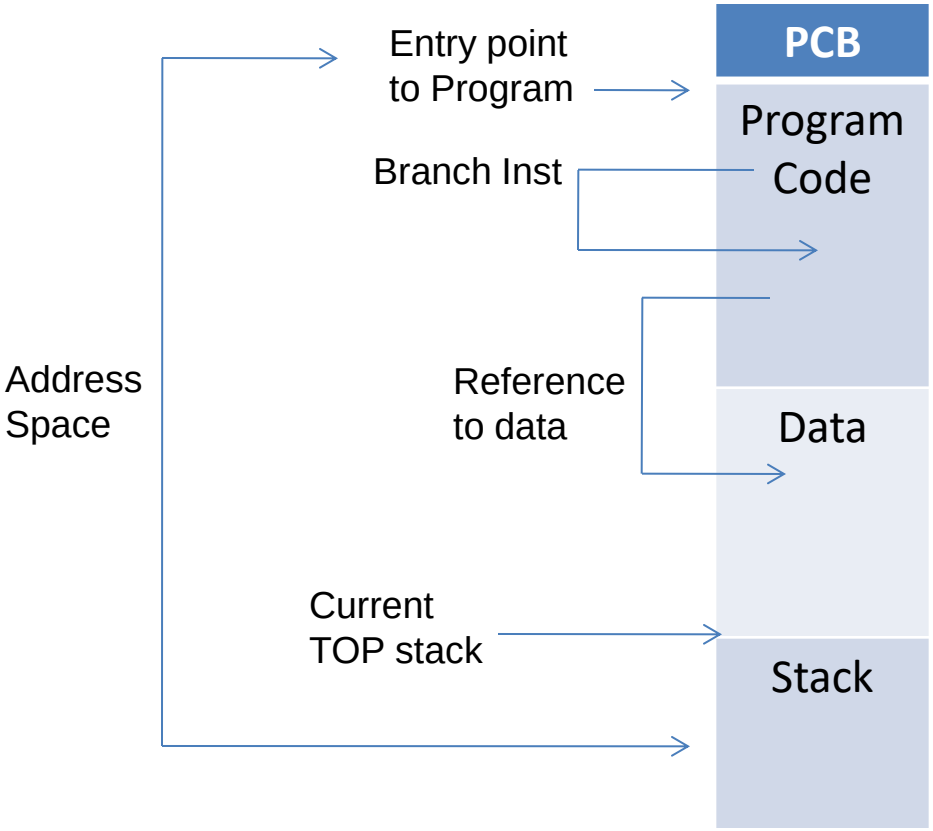
Introduction

- main memory has limited storage capacity
- entire memory can be considered as a sequential list of bytes, each byte has an address to locate info. **(physical address)**



Process

- it contains program code, data and stack



- Address space can be considered as a sequential list of bytes, each byte has an address to locate info. **(Logical address)**

Logical v/s Physical Address

- Logical Address
- as per the previous fig, address space considered as a sequential list of bytes, each byte has an address that is used to locate it → Logical Address.
- it is generated by CPU : CPU determines the location of each instruction and data.
- LAS → is a set of all logical addresses that can be referenced by a process
- Process has its own LAS.
- Logical addresses are limited by address size of the processors.
- Physical Address
- PAS-> set of all addresses occupied by a process in main memory during its execution.
- process can read and write its own physical address
- PAS may or may not be contiguous
- Physical address are limited to the amount of main memory
- There should be some mapping between Physical addresses and Logical addresses

Memory Management Unit [MMU]

(Memory Manager)

- *The mapping between physical address and logical address is done by memory manager*
- it is the part of the OS that deals with main memory.
- main goal → efficiently utilize the main memory among various simultaneously executing processes.
- Operations of Memory Manager :
  1. **Fetch :** when to move info from disk to main memory
  2. **Placement :** where to move info from disk to main memory
  3. **Replacement :** needed when sufficient memory is not available
  4. **Sharing :** allow more than one processes to share a piece of memory
  5. **Address translation :** translating logical address to physical address
  6. **Protection :** protect memory against unauthorized request for access

Memory Allocation

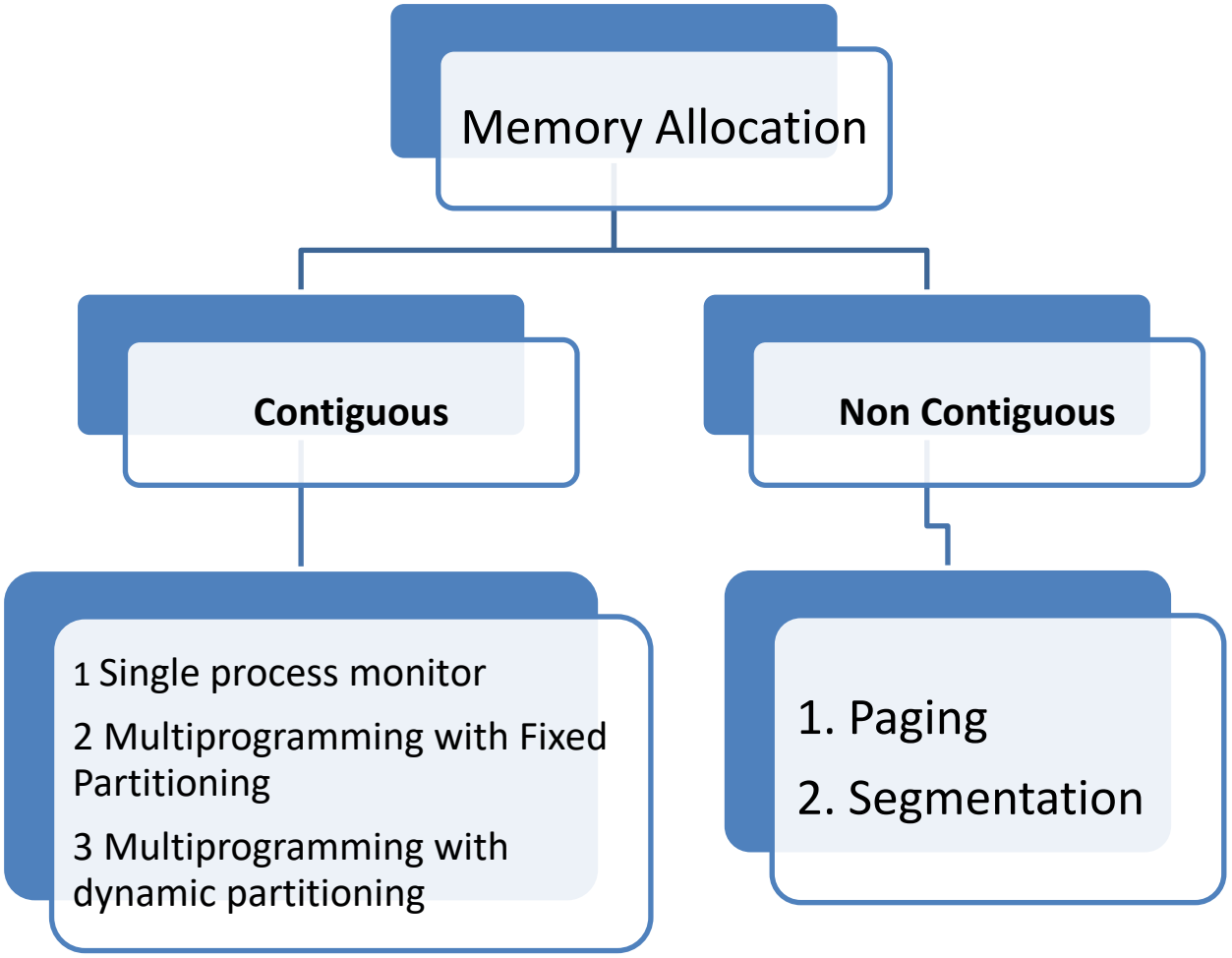
Why needed?

Answer:

Two goals :

1. High utilization
    1. Maximum possible memory should be utilized
    2. No single piece of memory should be wasted
2. High concurrency
    1. Maximum possible processes should be in the memory
    2. As more number of processes in memory, the CPU remains busy most of the time, resulting in better throughput.

Memory Allocation classification



Memory Allocation

Contiguous Memory Allocation

- simple and old method, generally not used by modern OS
- each process occupies a block of contiguous memory, entire process kept together
- when process comes into memory, enough space is to be found
- here, logical address space is not divided into partitions.

figure:

Advantages:

easy to understand and implementation

Disadvantages:

poor memory utilization

Memory Allocation

- **Non - Contiguous Memory Allocation**
  - Used by most modern OS
  - Logical address space of a process is divided into partitions, and for each partition contiguous chunk of free memory is allocated.
  - Physical memory will not be contiguous now.
  - Figure :
- Advantages:
  - better memory utilization
- Disadvantages:
  - complex to implement

Contiguous Memory Allocation

• Each process occupies a block of contiguous memory locations in main memory. Entire process is kept together.

• it can be used in three different ways.

1. Single Process Monitor

- It is simplest method
- Only single process is allowed to run
- Memory is shared between the process and OS
- Figure:

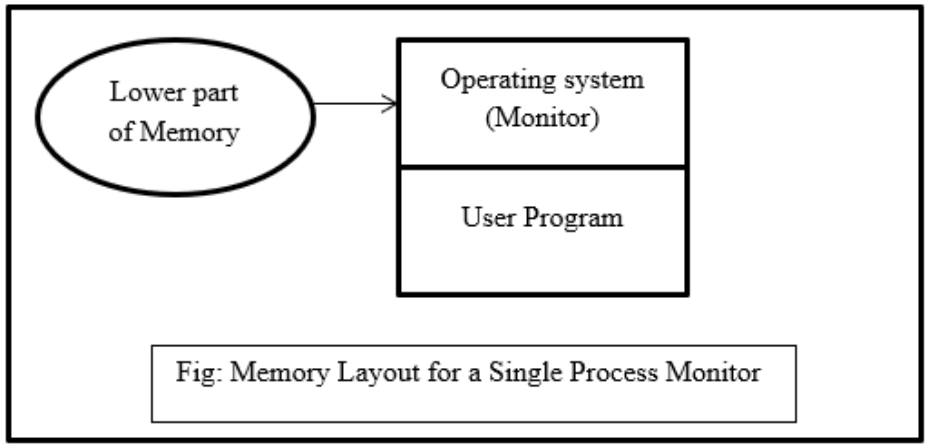


Fig: Memory Layout for a Single Process Monitor

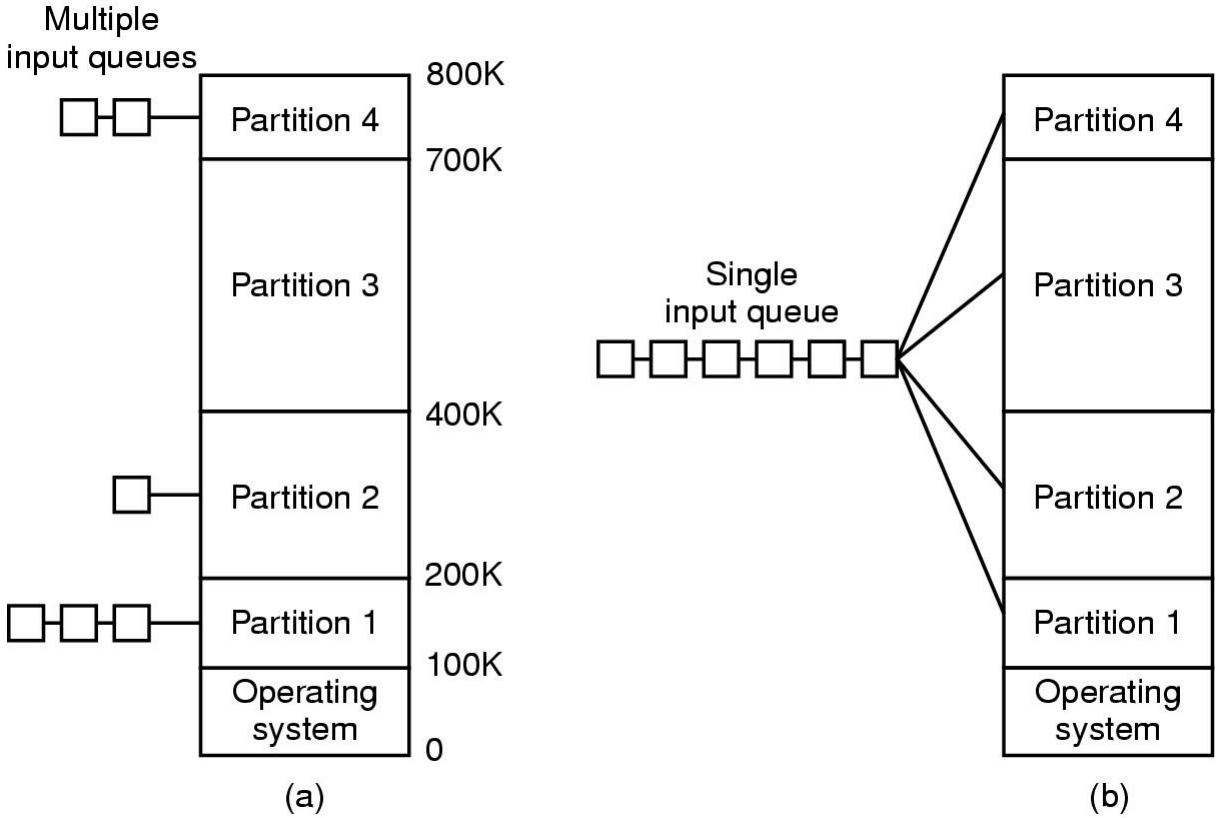
- Ex → MS-DOS, embedded systems, palmtop computers
- Procedure :
- Advantage : it is quite simple
- Disadvantage : only one process can be executed at a time

Contiguous Memory Allocation

2. Multiprogramming with Fixed (Static) partitions

- Allow multiple processes to execute simultaneously
- Memory is shared among OS and various simultaneously running processes
- CPU can be busy almost all the time due to multiprogramming
- Memory can be divided into fixed size partitions. Size can be equal or unequal for different partitions.
- Each partition can accommodate exactly one process i.e. only single process can be placed in one partition
- Procedure :
- Two ways to implement:
  1. Using Multiple input queue
  2. Using Single input queue

Contiguous Memory Allocation



Advantages	Disadvantages
- Easy implementation - Process overheads are low	- Cause Internal Fragmentation - Degree of multiprogramming is fixed

Contiguous Memory Allocation

3. Multiprogramming with dynamic partitions

- Allow multiple processes to execute simultaneously
- Memory is shared among OS and various simultaneously running processes
- Here, memory is **NOT** divided into fixed partitions. And number of partitions are also **NOT** fixed
- Process is allocated exactly as much memory as it required
- Initially entire memory is considered as a whole
- Procedure :

Contiguous Memory Allocation

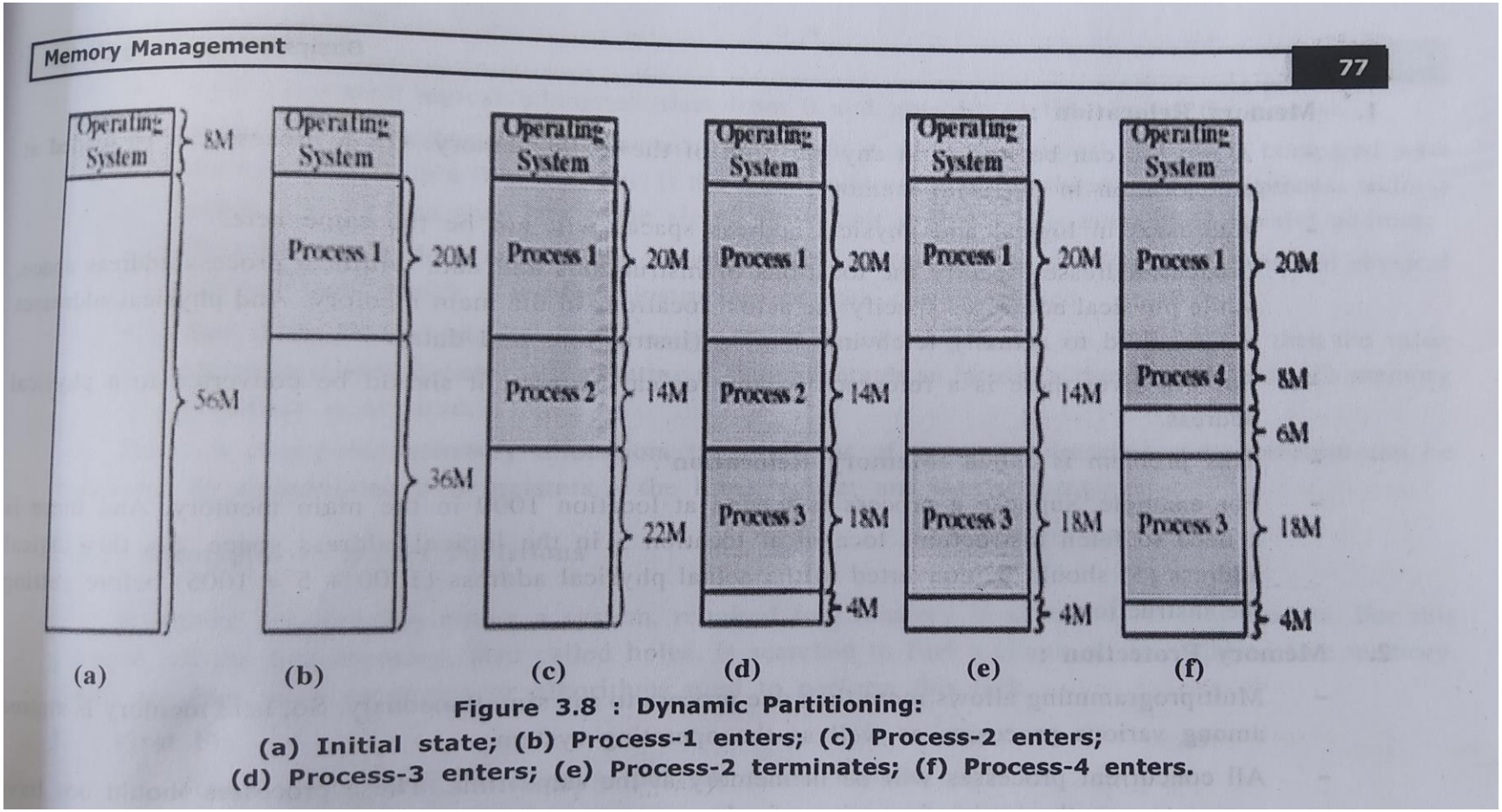


Figure 3.8 : Dynamic Partitioning:
(a) Initial state; (b) Process-1 enters; (c) Process-2 enters; (d) Process-3 enters; (e) Process-2 terminates; (f) Process-4 enters.

Advantages	Disadvantages
- Better memory utilization - Degree of multiprogramming is not fixed	- Cause the problem of external Fragmentation

Solution of this problem is defragmentation or compaction of the memory

Contiguous Memory Allocation

4. Memory relocation and protection

- These are the problems with multiprogramming

I. Memory Relocation :

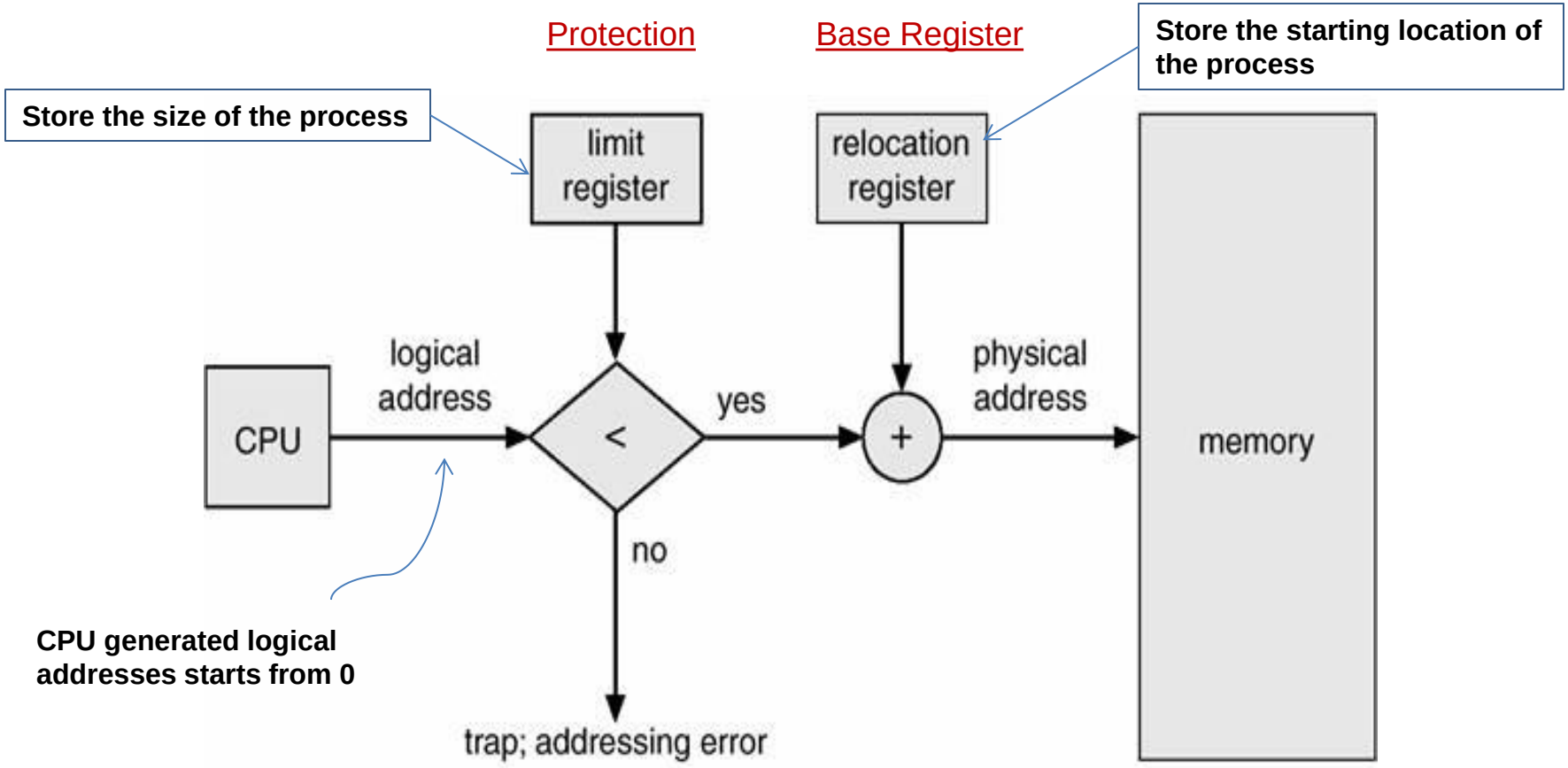
- Logical address specify the locations of instructions and data while physical address specify the actual location in main memory
- So, physical address are required to fetch the information
- So, whenever there is a reference to any logical address, it should be converted to physical address
- This problem is called → Memory Relocation

II. Memory Protection :

- As in multiprogramming, many processes are running concurrently in memory. They should not have access to unauthorized information of other processes. Each process should read and write only its own data.
- This problem is called → Memory Protection

Contiguous Memory Allocation

- Solution (memory Relocation and Protection):
- As in multiprogramming, many processes are running concurrently in



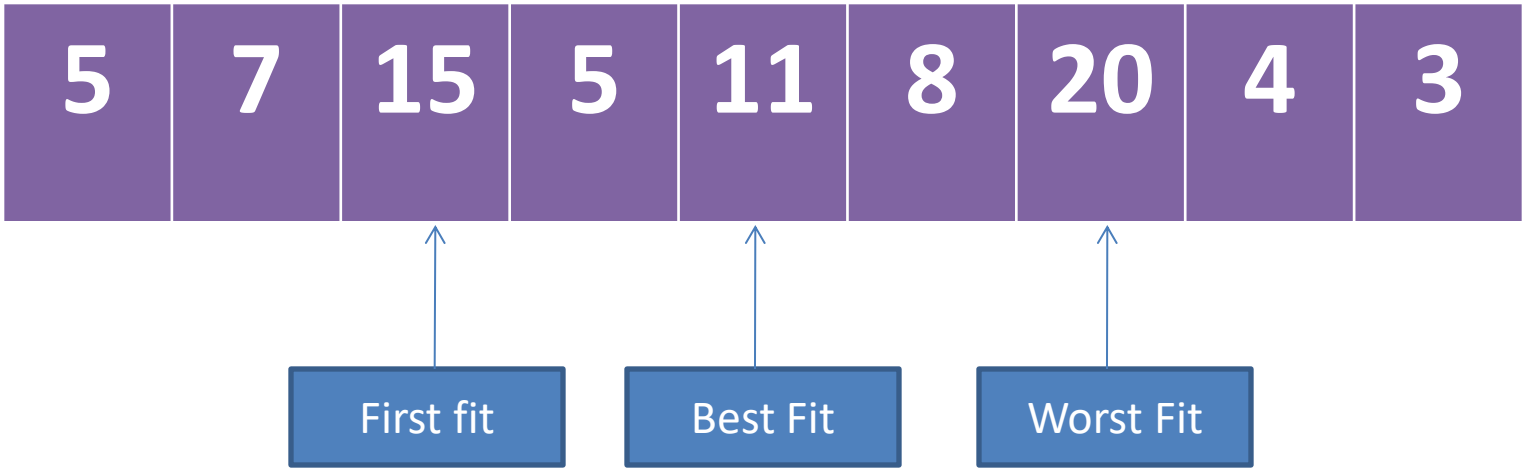
Contiguous Memory Allocation

- Strategies to select partitions: Four main strategies or algorithms to do this task

1. First fit
   - Search starts from starting location of the memory and find large enough hole that can hold the process
   - Search time is small and memory loss is higher in this case
2. Next fit
   - Search starts from where the previous search was ended
   - This is because to avoid repeating search of the memory
3. Best fit
   - Entire memory is searched here
   - The smallest hole which is large enough to hold the process is selected
   - Search time is high in this case and memory loss is less
   - It can cause external fragmentation
4. Worst fit
   - Entire memory is searched here
   - The largest hole which is large enough to hold the process is selected
   - This algorithms can be used with only with dynamic programming

Contiguous Memory Allocation

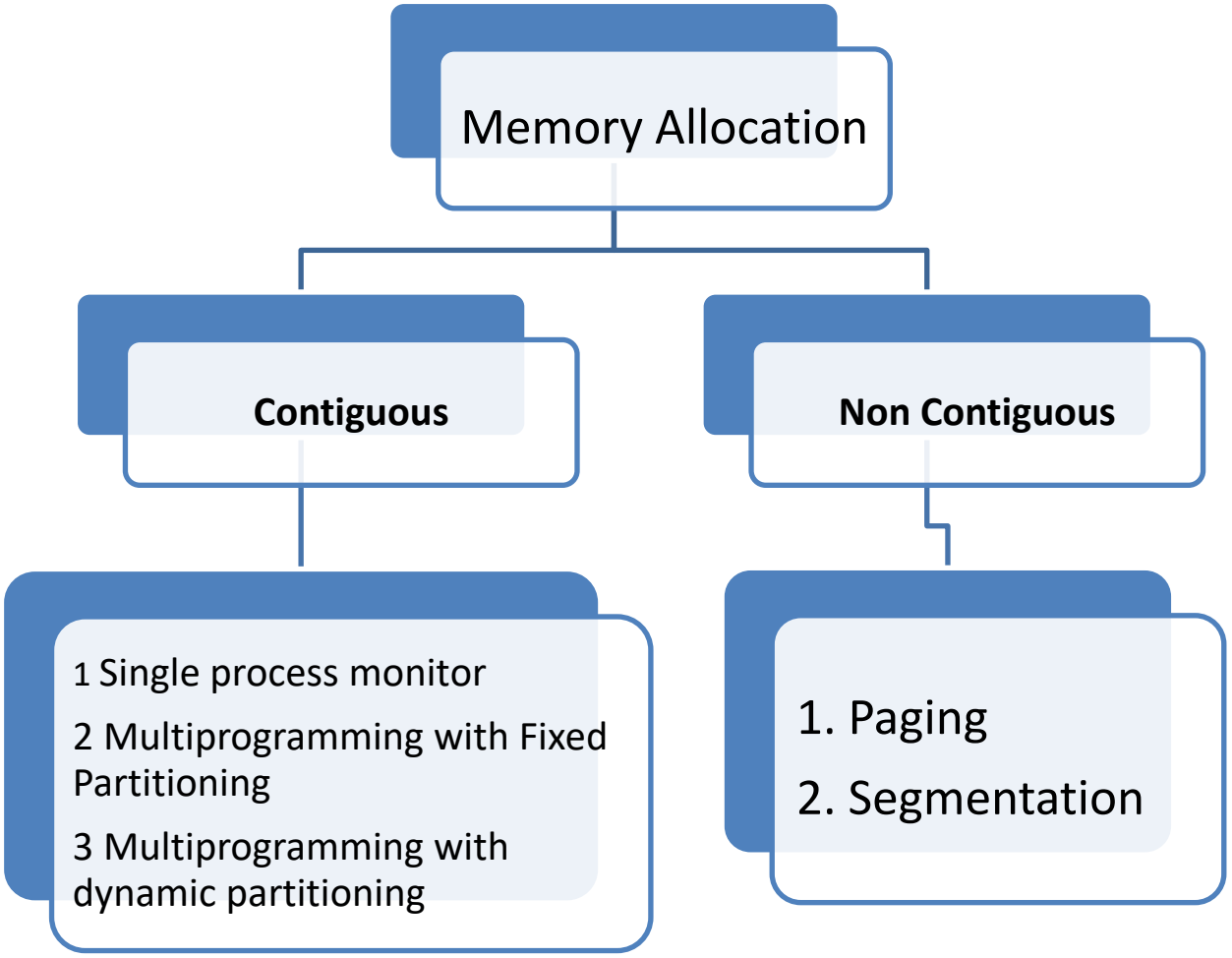
Finding the partition for selected by First fit, Best fit and Worst fit for process of 10 MB



Fragmentation

- Refers to the unused (free) memory that can't be allocated to any process
- Two kinds → External and Internal
- **External Fragmentation**
- Wastage of memory between partitions caused by non contiguous free space
- It's a problem with 'contiguous memory allocation with dynamic partition'
- Holes are generated due to memory allocation and de-allocation
- Inappropriate holes can't be used for allocating a block of memory, so external fragmentation generated
- Solution →compaction or de-fragmentation of the memory
- **Internal Fragmentation**
- Wastage of memory within partitions caused by the difference between the size of the partition and size of the process loaded.
- It's a problem with 'contiguous memory allocation with fixed partition'
- Solution → dynamic partition

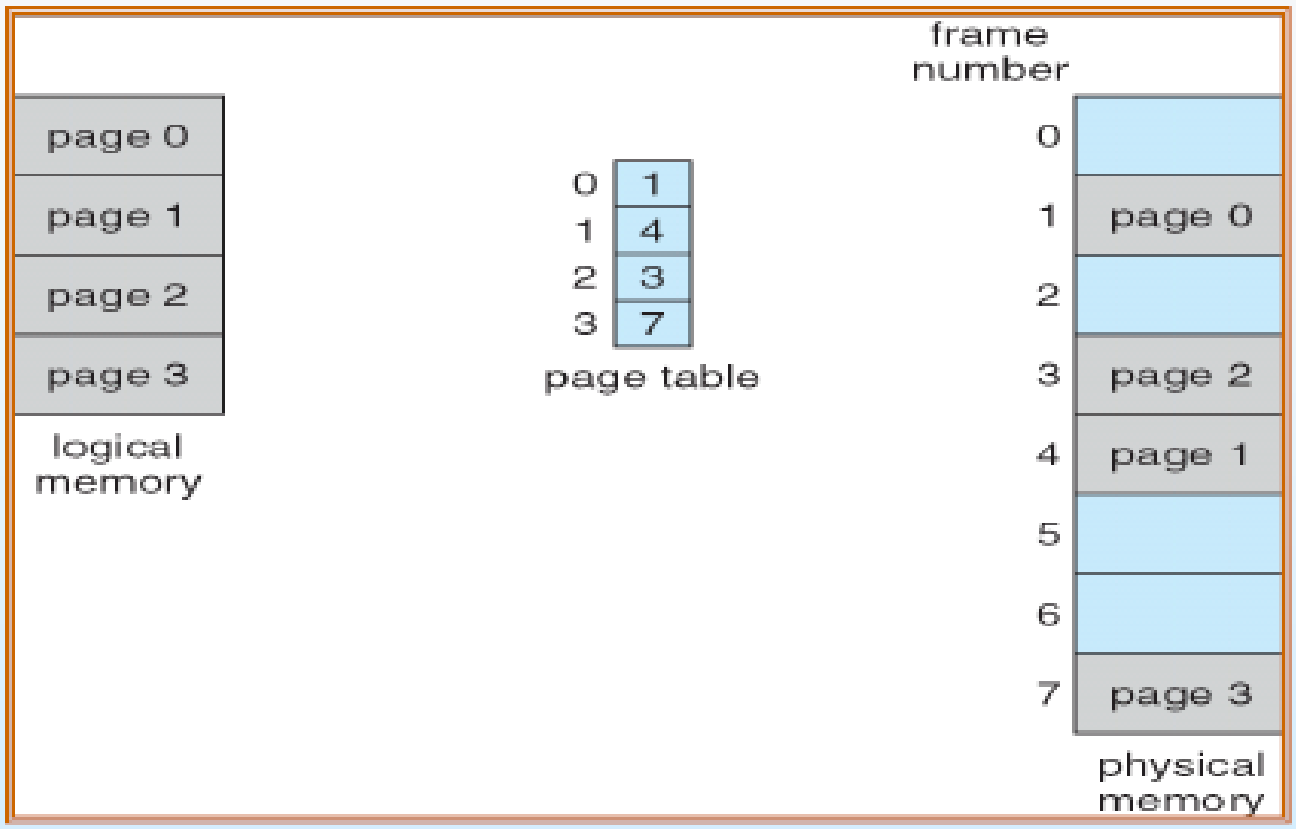
Memory Allocation classification



Non-Contiguous Memory Allocation : (1)Paging

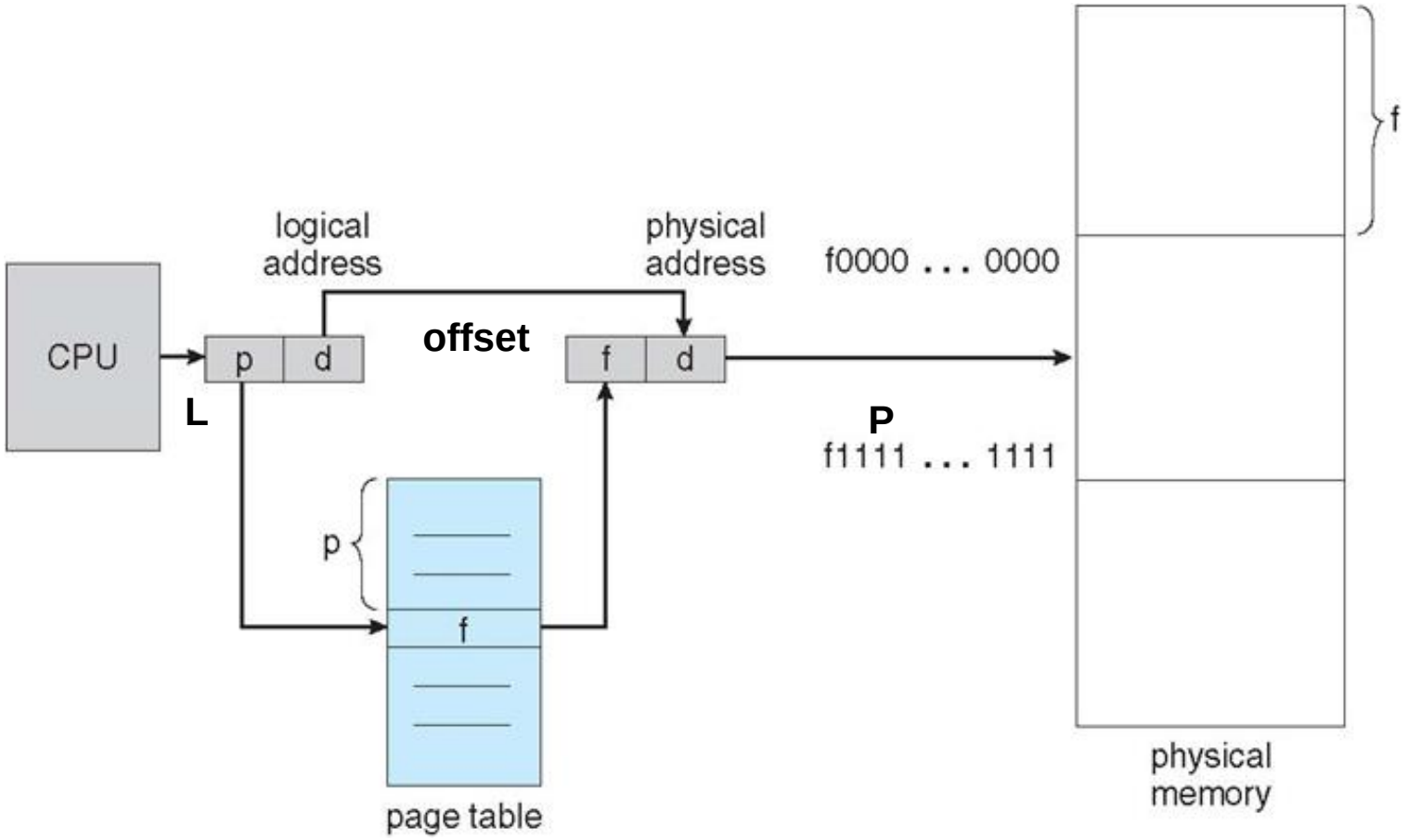
- Here, logical address space of process is divided into partitions
  (Pages are of same size : typical page size is power of 2)
- For each partition, a contiguous chunk of memory is allocated
- PAS is not contiguous, it will be scattered over entire memory
- Logical address space is divided into blocks of fixed sizes : Pages
- Physical address space is divided into blocks of fixed sizes : Frames
- Pages and frames will be of the same size
- Procedure :
- OS maintains a table called → page table for each process
- It contains the index of page no and information about frame no
- Logical address is divided into → page no and offset

Basic concept of Paging



Non-Contiguous Memory Allocation : (1)Paging

- Implementation



Logical Address (L)=	Page No (P)	Offset (d)
Physical Address(P)=	Frame No (P)	Offset (d)

Non-Contiguous Memory Allocation : (1)Paging

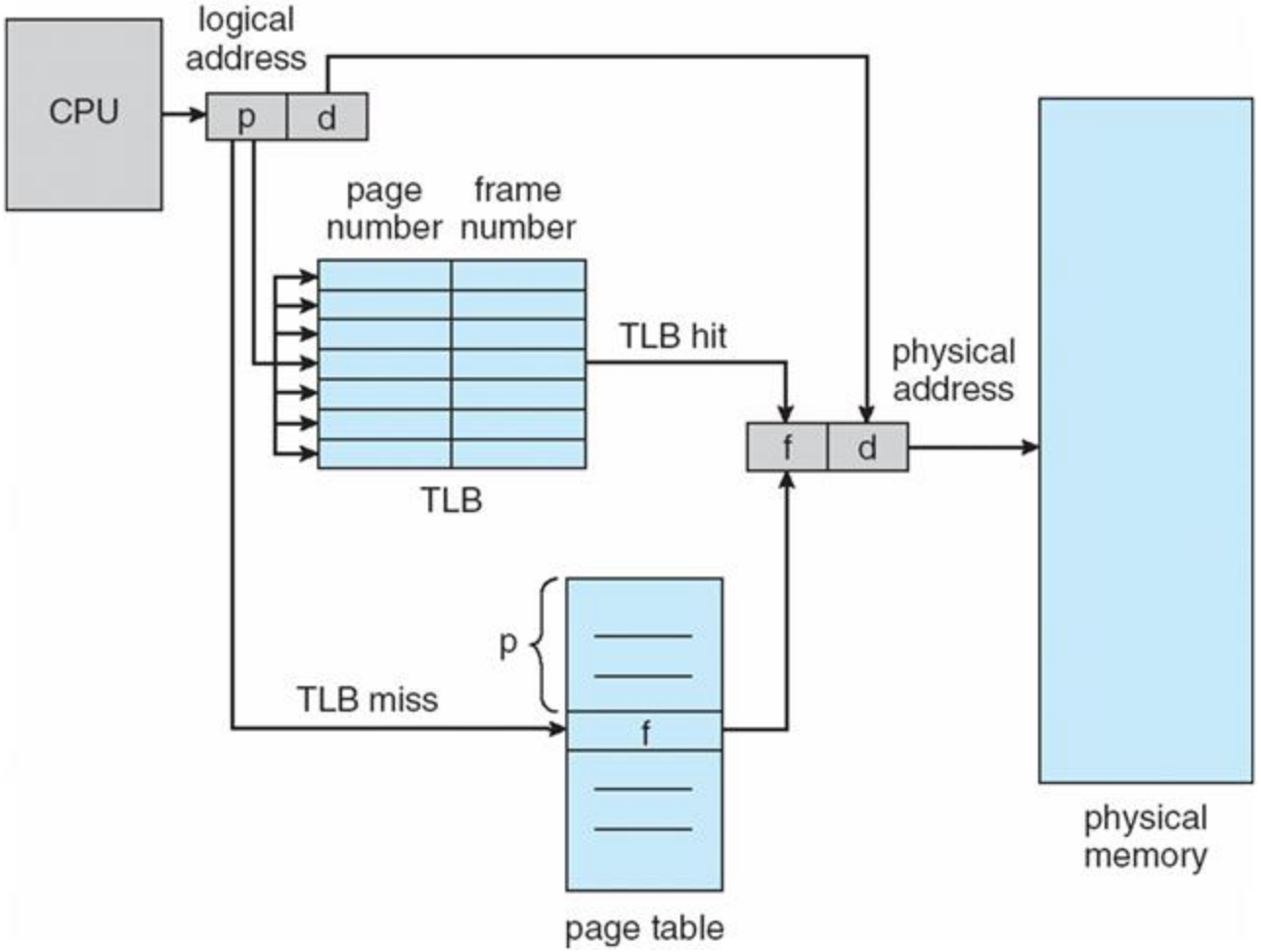
- Advantages of basic Paging method
  - Allocation and de-allocation is fast
  - Swap-in and swap-out operations are fast
  - No external fragmentation

- Disadvantages of basic Paging method
  - Additional memory reference is required
  - Size of page table may be too large to keep it in main memory
  - May be internal fragmentation

Paging : Variations

- Two main problems → high access time and large size of page table
- Most widely used variations are :

1. TLB (Translation Look-aside Buffer)



Paging : TLB

- Overcome of slower access
- TLB is a page-table cache (fast associative memory)
- All the table entries can be searched simultaneously within TLB
- Due to high cost, storage capacity of this memory is limited. So, subset of page table is kept in it.

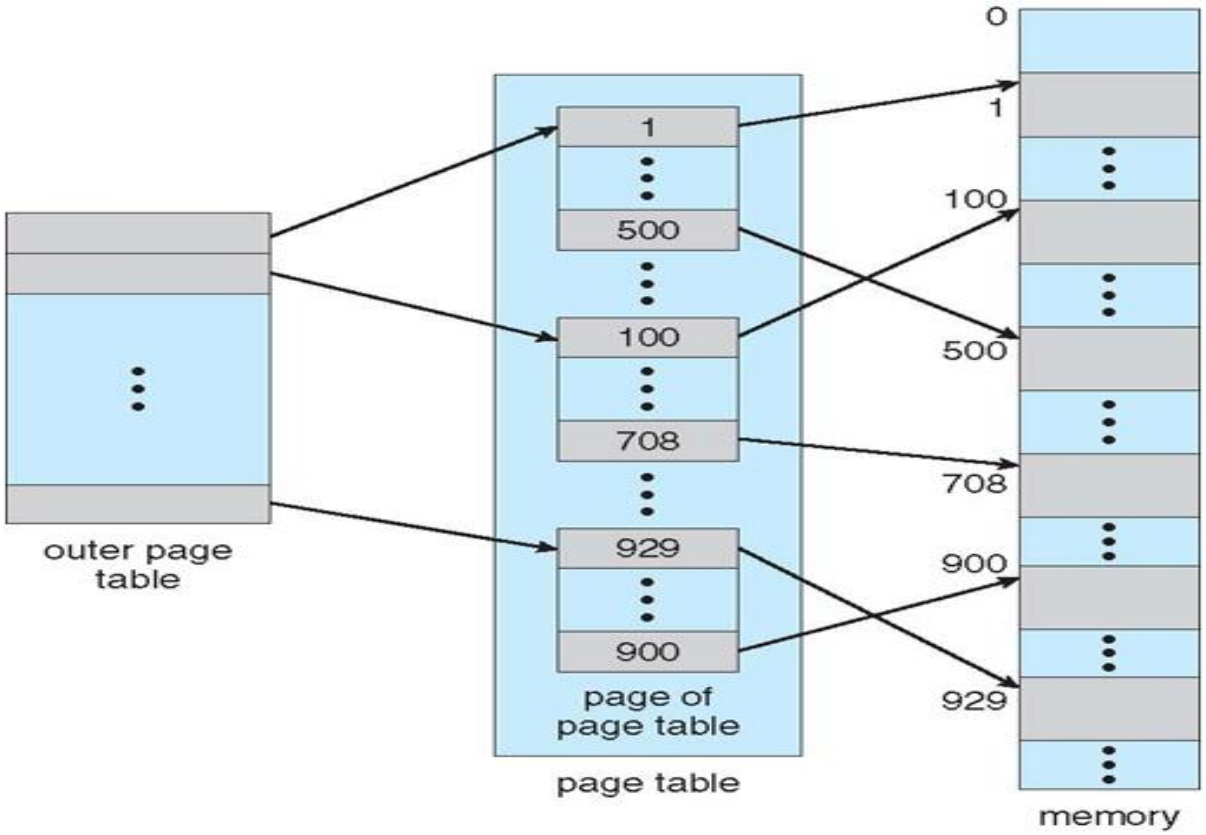
(generally, frequently use થતા pages ની entry TLB માં થાય જેથી access speed વધે.)

- Entry of TLB contains a page no and a frame no where the page is stored
- Working of TLB :
- → TLB hit
- → TLB miss
- When TLB is full, some existing entries are removed based on some replacement algorithms

Paging : Multilevel Page Tables

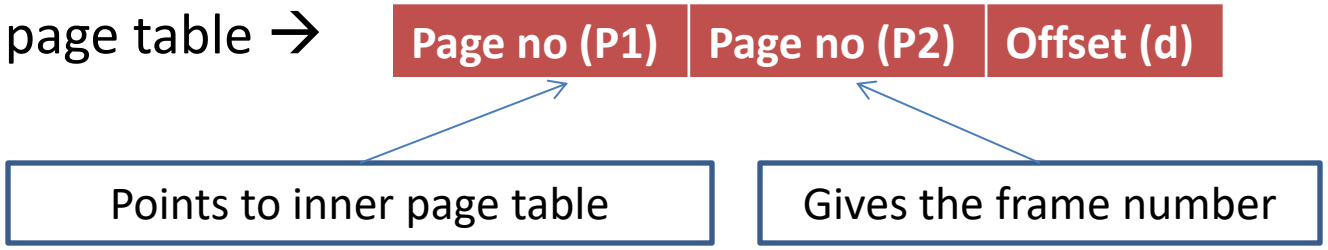
2. Multilevel Page tables

- Overcome the problem of large size tables
- Page table size increases due to increase in process size.
- And, entire page table should be in main memory
- Solution is → multilevel page tables



Paging : Multilevel Page Tables

- Entire page table is divided into various inner page tables, outer page table contains limited entries and inner page tables contain the actual frame numbers to instruction or data
- In two level page table →



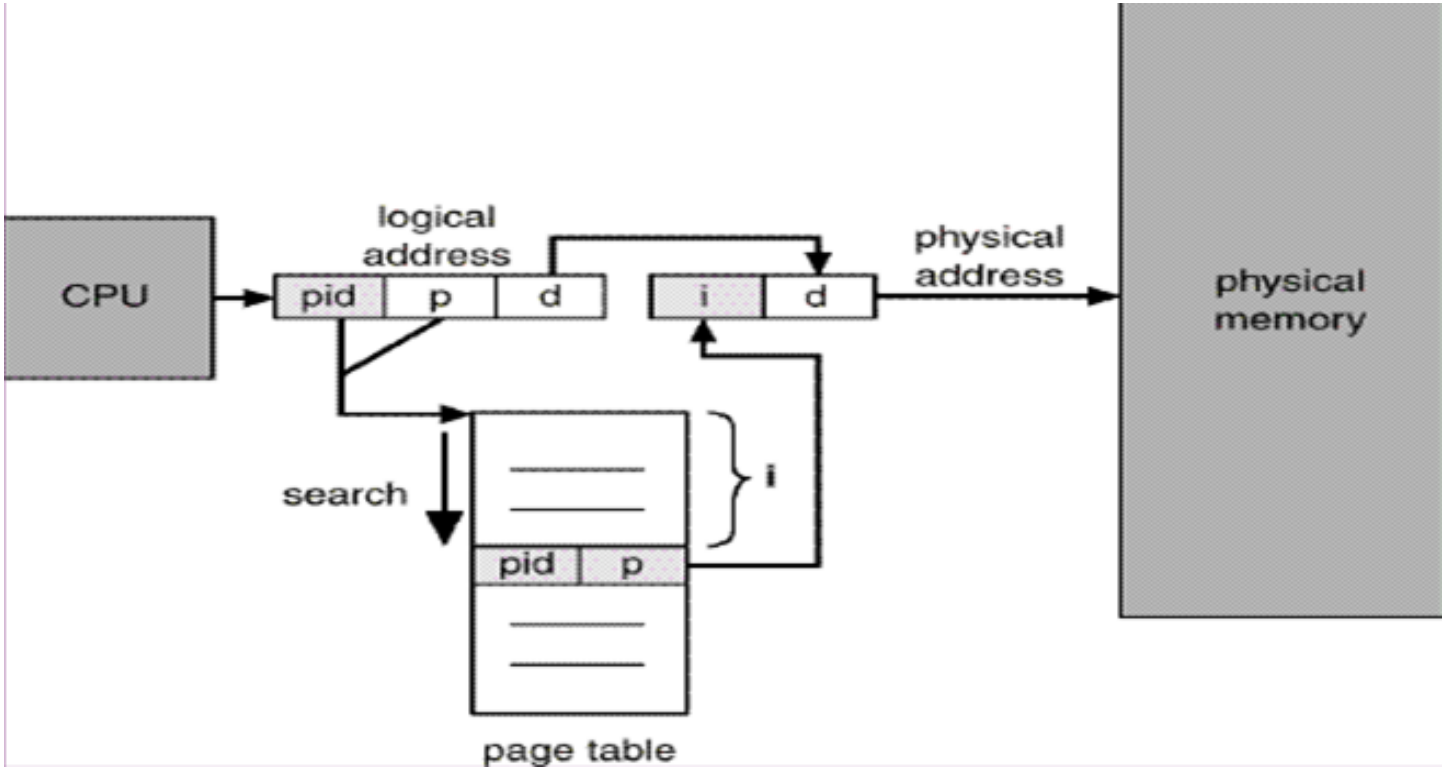
- Advantages :
- All the inner page tables need not to be in main memory simultaneously
- Disadvantage :
- Extra memory access required when increasing level of paging

Paging : Inverted Page Tables

2. Inverted Page tables (IPT)

- Overcome the problem of memory overhead due to large size of the page tables
- Generally size of the page table depends on the size of the process
- In IPT, there is one entry per frame of physical memory in table rather than one entry per page of logical address space
- So, there is a need of single system wide table rather than separate tables per processes
- Each entry in IPT is → Process Id and Page Number
- As CPU generate s process id, page number and offset. An IPT is searched to find a match a process id and page no and Frame number is determined

Paging : Inverted Page Tables



- Advantage :
- One page table for all processes
- Size of the page table is constant
- Disadvantage :
- Search time is very large, so translation becomes slow

Non-Contiguous Memory Allocation : (2) Segmentation

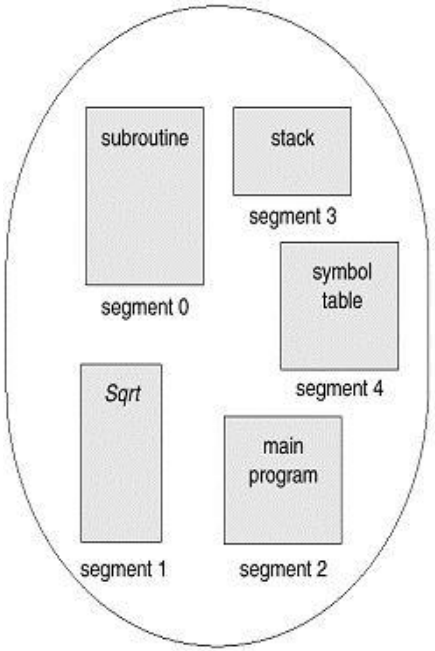
- Logical address contains → code, data and stack
- *Code can be comprise.. The main function, other user defined functions and library functions.*
- *Data can be local variables, global variables, symbol table etc..*
- Logical address space of a process is divided into blocks of ***varying size*** called *Segments*
- Each segment contains a logical unit of a process
- Each segment can ne considered as a completely independent address space
- All segments are of varying lengths depends upon the size of a logical unit
- All segments are given unique number to identify them
- OS maintains a table called **Segment Table** for each process
- Segment table contain info like → size of segment and location in the memory where the segment has been loaded

Non-Contiguous Memory Allocation : (2) Segmentation

- Logical address address is divided into two parts :
  I. Segment number : which identifies a segment
  II. An Offset : which gives the actual location within a segment

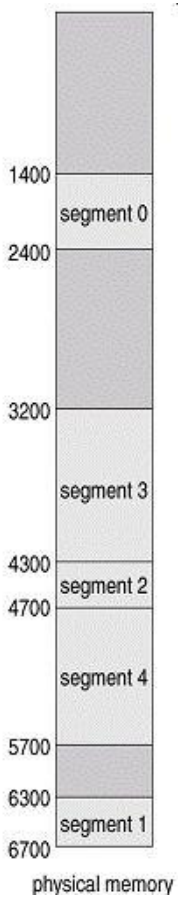
Segment Limit specifies the length of the segment

Segment Base specifies the starting physical address



	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

segment table

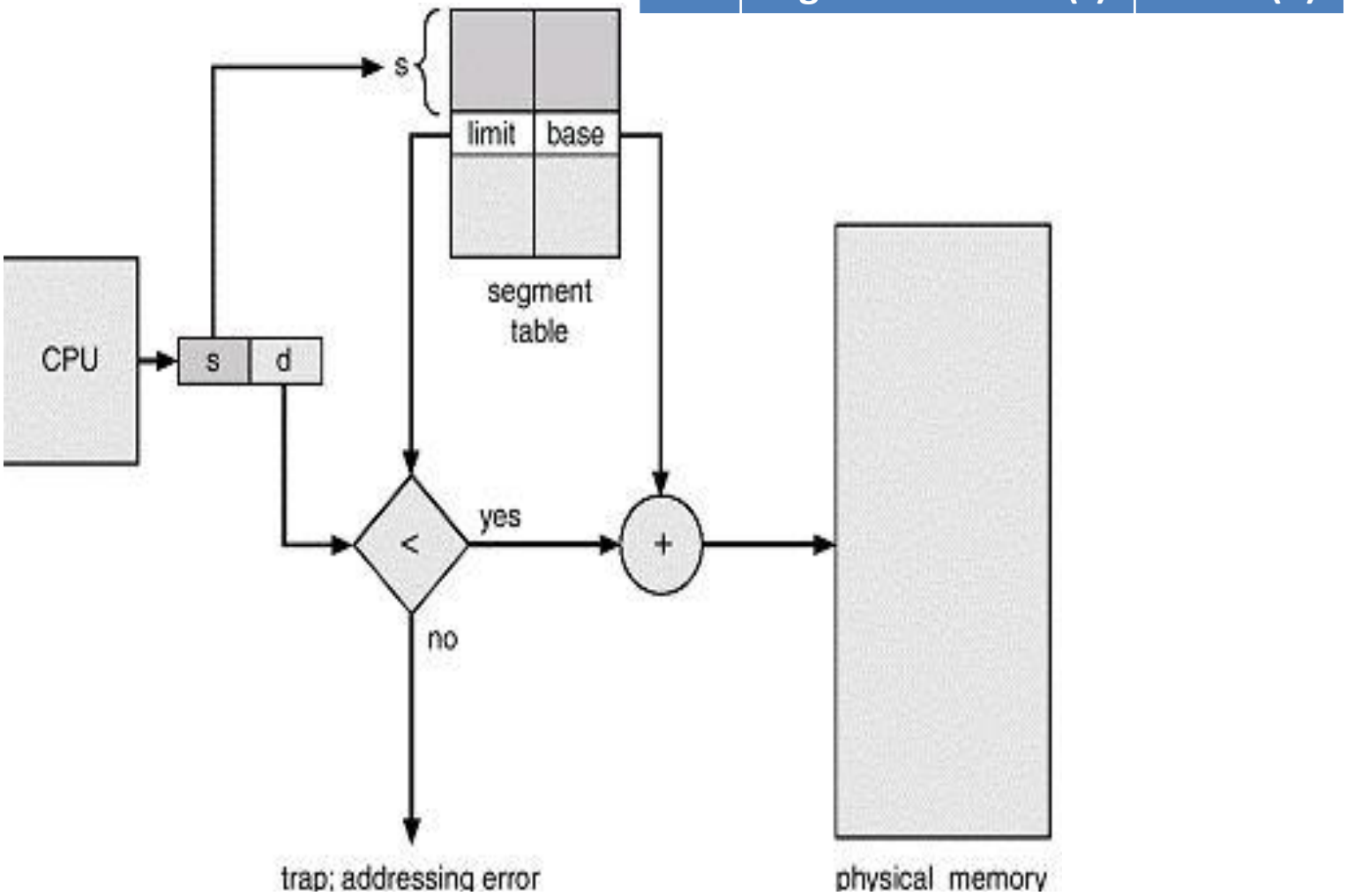


Segment table do segmentation

Non-Contiguous Memory Allocation : (2)
Segmentation

- Implementation :

L =	Segment Number(s)	Offset (d)



Non-Contiguous Memory Allocation : (2)
Segmentation

- Advantages :
- All segments are independent of each other
- No need to change or re-compile any segment while modifying other one
- Sharing of procedures and data among various processes is simple
- We can give different kind of protection to different segments
- There is no internal fragmentation
- Disadvantages :
- Difficult to allocate contiguous free memory to segments
- External fragmentation is possible

Swapping

- *Definition: It's a technique in which process is moved between main memory and disk*

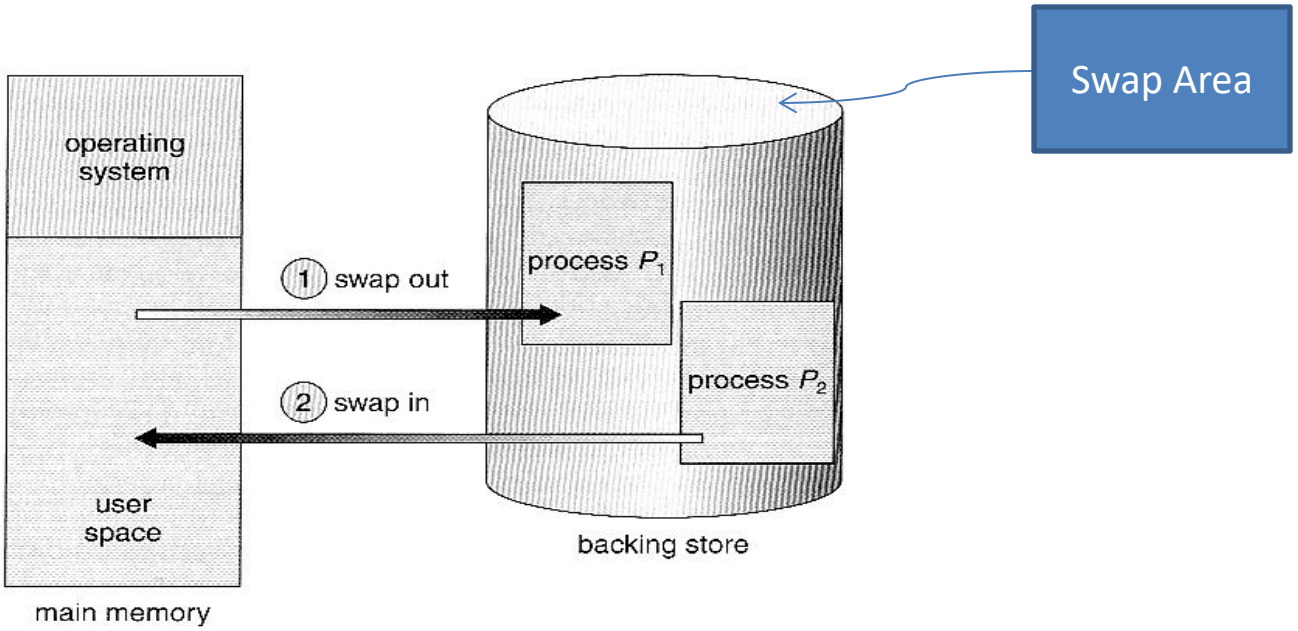


Figure 8.5 Swapping of two processes using a disk as a backing store.

- Disadvantage :
- Need of memory relocation
- External fragmentation

Virtual Memory

- It's a technique that allows a process to execute even though it is partially loaded in main memory
- It removes the requirement that an entire process should be in main memory
- Here, logical address space is referred as a virtual address space and virtual address space can be larger than physical memory
- Advantages :
  - A process larger than memory can also be execute
  - No need to keep entire process in memory
- Implementation :

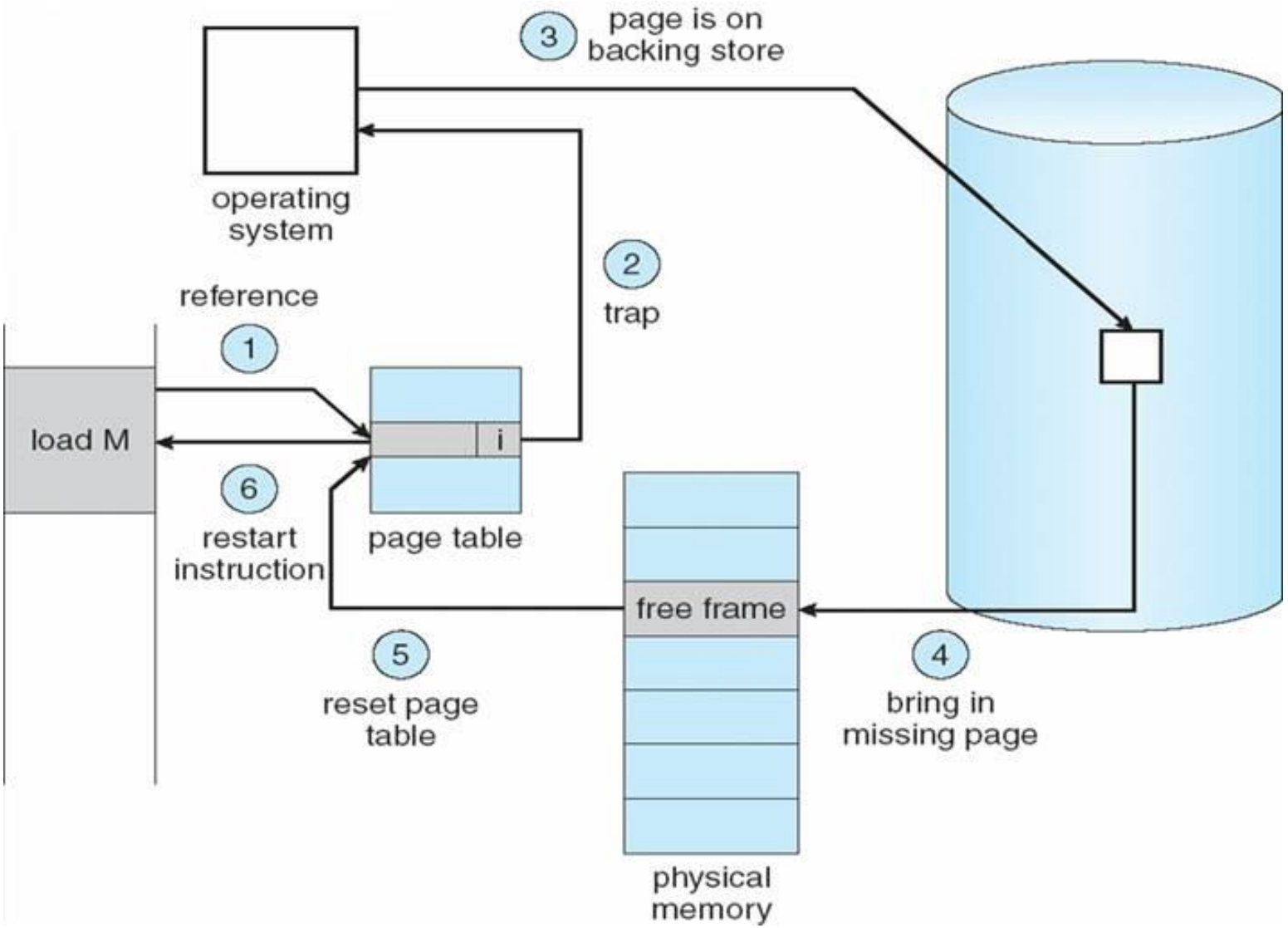
1. Demand paging
2. Demand segmentation
3. Segmentation with paging

Virtual Memory

1. Demand paging

- Pages are moved to memory as per requirement
- No all pages are moved together at the same time.
- Implementation :
  - It's similar to paging with swapping
  - Processes are kept on secondary memory (HDD)
  - Rather swapping the entire process into memory, only pages, which are required for execution, are swapped.
  - Page table includes *valid-invalid bit* for each page entry.
  - *Valid* → page is currently present in the memory
  - *Invalid* → page is not loaded in the memory
  - When process starts its execution, bit is set to *invalid* for all entries in the page table.
  - When page is loaded in memory, corresponding frame number is entered and bit is set to *valid*

Working of Demand paging



Virtual Memory

- Steps involved in Demand paging

1. Whenever any logical address is generated, page table is searched, if validity bit is set to *invalid*, it means page is not available in main memory. This situation is called **Page Fault**.
2. OS is requested to bring this page in memory from HDD
3. OS determines the page location on HDD
4. The page is brought into free frame in main memory
5. Frame number is entered in page table entry (page table is reset/updated by setting validity bit to *valid*)
6. Instruction is restarted

Page Replacement Algorithms

- In virtual memory, pages from the disk are brought into frames of main memory when page fault occurs.
- Two algorithms
  1. *FIFO (First In First Out)*
  2. *LRU (Least Recently Used)*

Thank you...