Memory Management

Memory Management

- Background
- Swapping
- Contiguous Allocation
- Paging
- Segmentation
- Segmentation with Paging

Address Binding

- A program resides on a disk as a binary executable file.
- To execute the program, bring it to memory and placed within a process.
- The processes on the disk that are waiting to be brought to memory for execution form an input queue.
- Processes are selected from input queue, load it to memory, execute the process and terminated by freeing the memory.

- Every byte in the memory has specific address defined by the h/w is known as physical address. Whenever a program brought into memory for execution it occupies certain number of memory locations. The set of all physical addresses used by the program is known as physical address space.
- After compilation program can run some specific address known as logical address . The set of all logical addresses used by the program known as logical address space.

- When a user program is brought into main memory for execution its logical addresses must be mapped to physical addresses.The mapping from logical address to physical address is known as address binding.

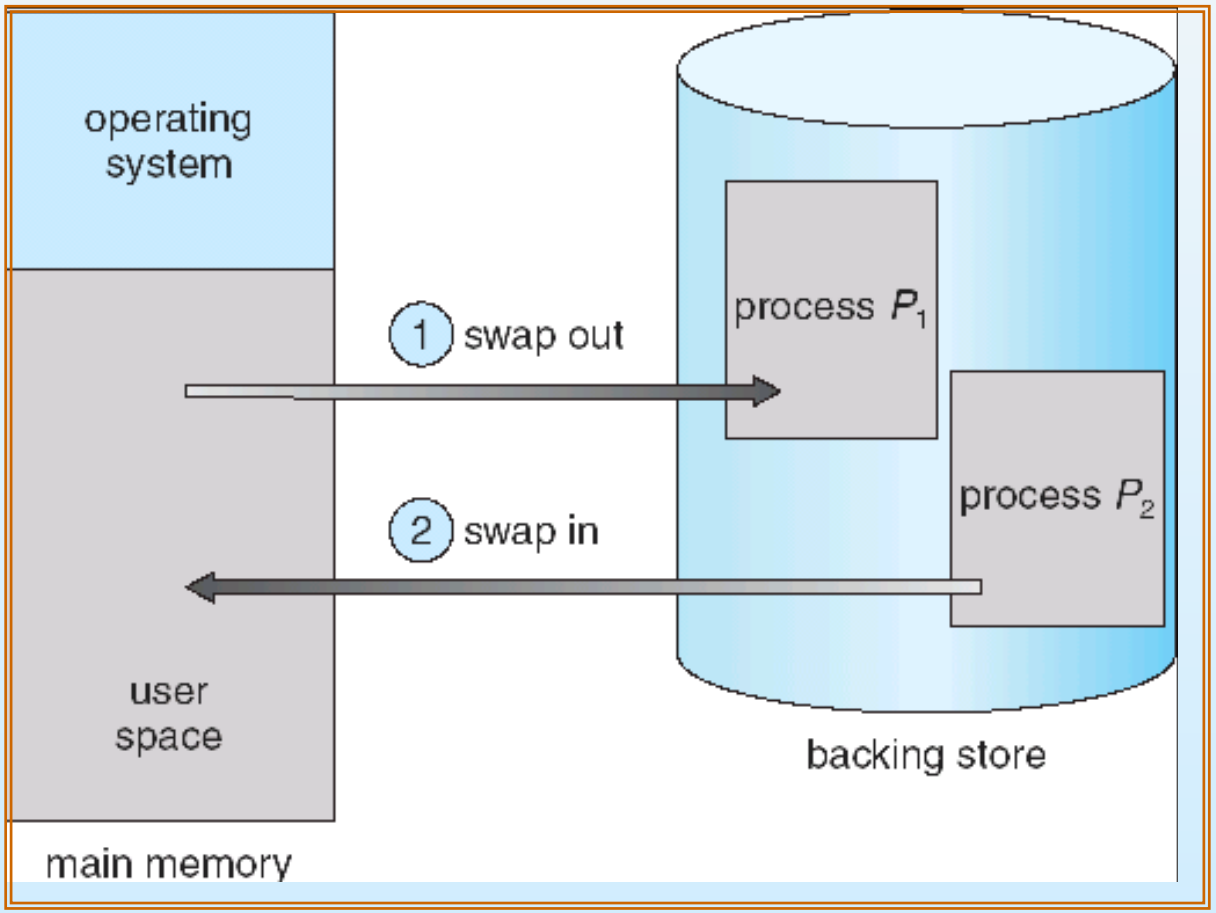
Address binding of instructions and data to memory addresses can
happen at three different stages

- **Compile time**: If memory location is known a priori program will occupy in the main memory. *absolute code* can be generated; logical same as physical address.
- **Load time**: Must generate *relocatable code at compile time* if memory location is not known at compile time, and convert into absolute code at the load time.
- **Execution time**: Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Program generates relocatable code at compile time and convert into absolute code at run time.

Swapping

- A process must be in memory to be executed.
- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution.
- Eg. Round Robin CPU scheduling algorithm
  - If time quantum of process P1 expires, memory manager swap out process P1 from memory and swap in next process.

Schematic View of Swapping

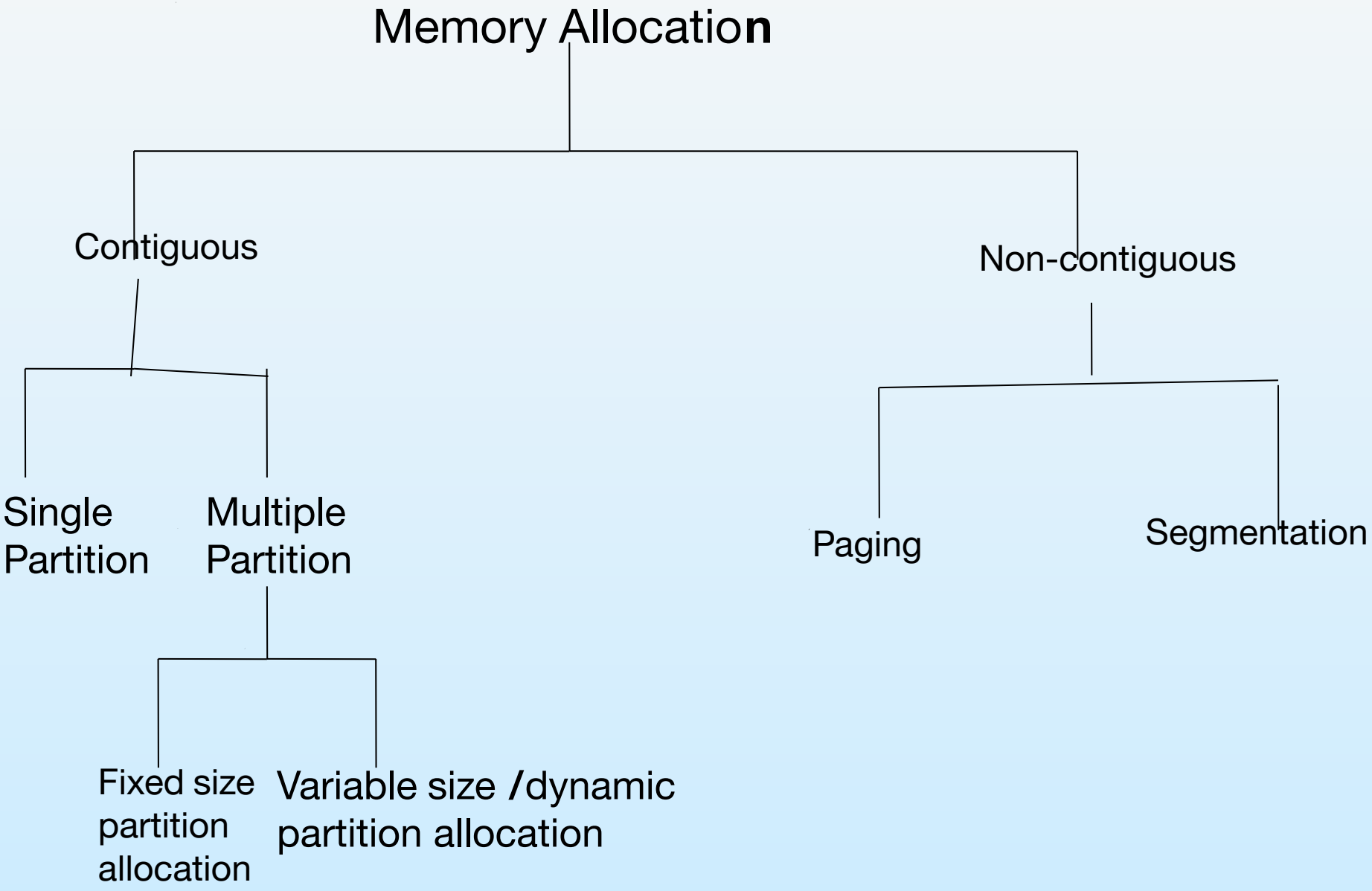


Memory management strategies

Strategies to manage shared memory

- Continuous memory allocation
- Non continuous memory allocation

Memory allocation or management techniques



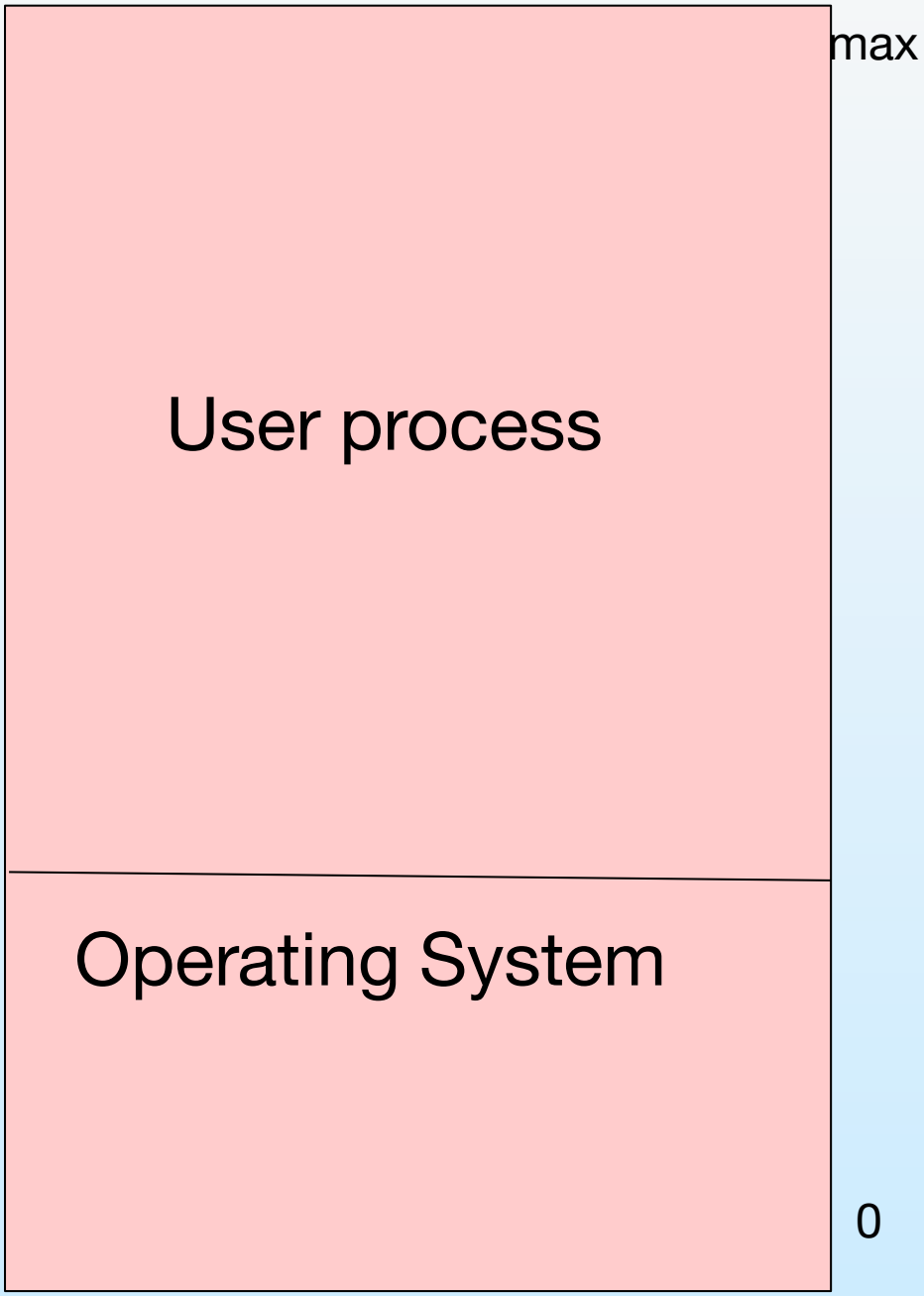
Contiguous Allocation

- Each process is allocated to a single continuous part of the memory.
- Two types

1. Single partition
2. Multiple partition

Single Partition Allocation

- The main memory is divided into two parts- one of them is permanently allocated to OS and other part is to user process.
- In this scheme Operating system is residing in lower part of memory and user processes are executing in higher memory.
- Here only one process can be executed at a given time & that is loaded to main memory. When another process arrives, the new process overwrites the old one.


max
User process
Operating System
0

- **Advantages**
  - It is simple.
  - It is easy to understand and use.
- **Disadvantages**
  - It leads to poor utilization of processor and memory.
  - Users job is limited to the size of available memory & only one at a time.

Multiple-partition allocation

- Dividing the memory into several partitions. In multiple-partition method, when a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process.
- Available partition of memory is called *hole*.

- Os keeps a table which tracks available memory parts
- Operating system maintains information about:
  a) allocated partitions b) free partitions (hole)

Multiple-partition allocation

Two Types

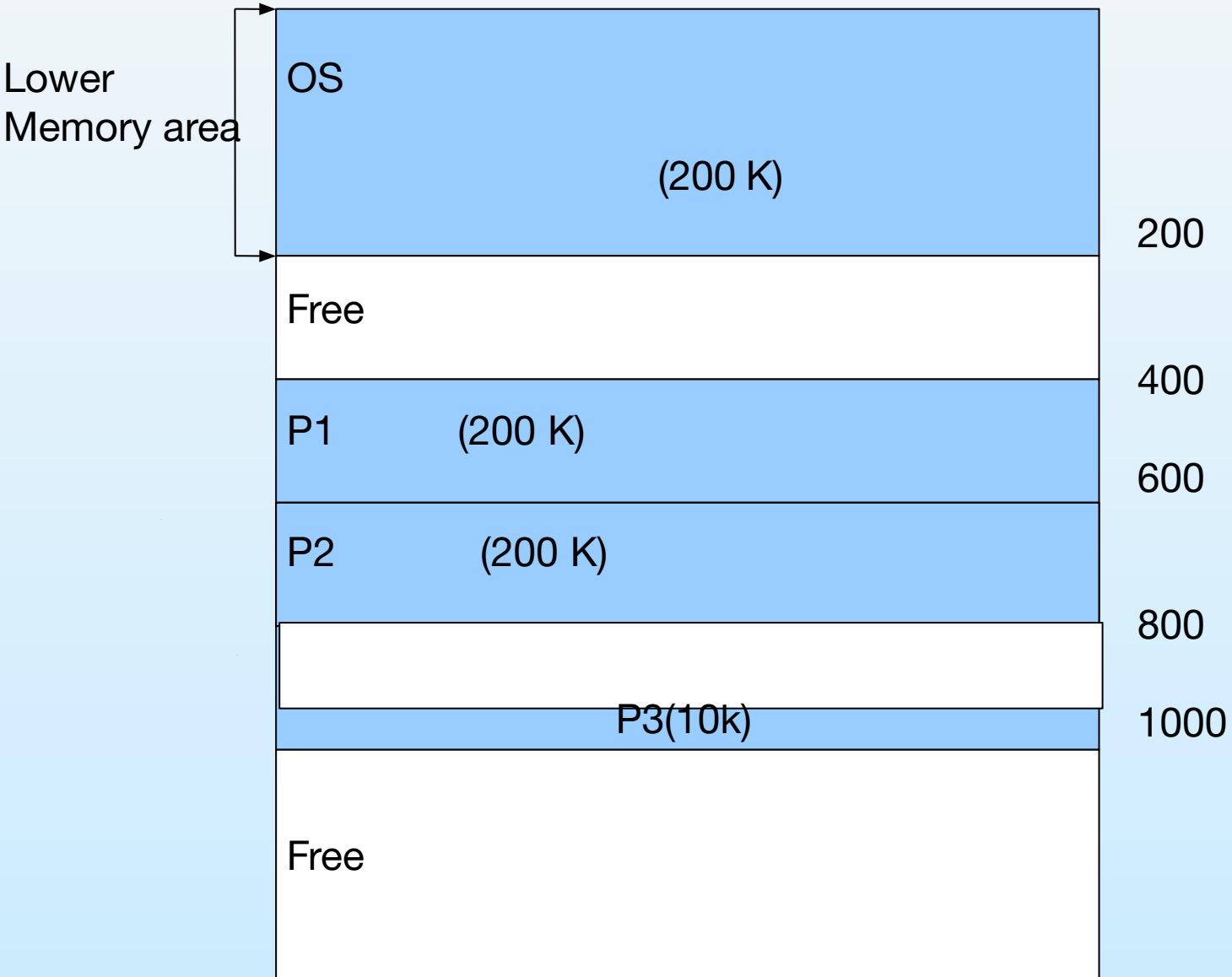
1. Fixed Equal-size Partitions
2. Fixed Variable Size Partitions

Fixed Equal-size Partitions

- Each partition is of fixed size and can contain only one process. Whenever a partition is free a process whose size is less than or equal to any partition is selected from the input queue and loaded into the partition.
- When the process terminates the partition becomes free to be allocated.

- Here is one problem that is memory utilization is not efficient. Any process regardless how small it is, occupies an entire partition which leads to the wastage of memory.This phenomenon which result in the wastage of memory within partition is called internal fragmentation.

a)Fixed equal-size partitions



Fixed Variable Size Partitions

- A separate queue is maintained for each partition. whenever a process arrives , it is placed into the input queue of the smallest partition large enough to hold it.

Partition selection algorithm

How to satisfy a request of size *n* from a list of free holes

- **First-fit**: Allocate the *first* hole that is big enough.
  - Searching can start either at the beginning of the set of holes or where the previous first-fit search ended.
- **Best-fit**: Allocate the *smallest* hole that is big enough
  - Must search entire list, unless ordered by size. Produces the smallest leftover hole.
- **Worst-fit**: Allocate the *largest* hole
  - Must also search entire list, unless it is sorted by size. Produces the largest leftover hole.

First-fit and best-fit better than worst-fit in terms of speed and storage utilization

- At a certain point of time there will be a set of holes of various sizes scattered in the memory. There is a possibility that the total available memory is large enough to accommodate the waiting process. But it cannot be utilized as it is scattered. This wastage of memory is called external fragmentation.

- To get rid of this problem it is relocate some or all free partitions of the memory to one end to make a large hole. This technique of reforming the storage is known as compaction.
- Compaction result the memory partition into two-used part and free part.

Fragmentation

- External Fragmentation
- Internal Fragmentation

Relocation and Protection

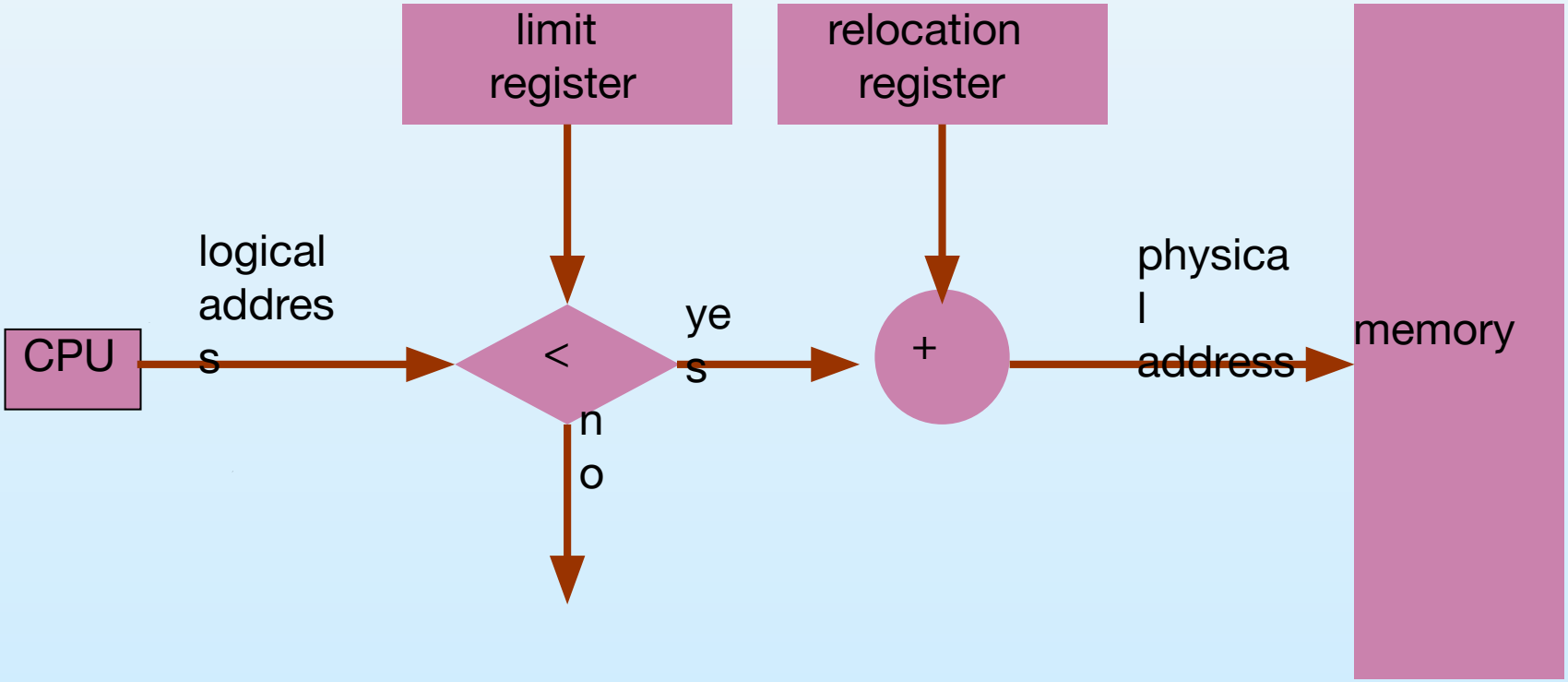
- Relocation-register scheme used to protect user processes from each other, and from changing operating-system code and data
- Relocation register contains the starting address of the partition into which process is to be loaded.

- Memory Management Unit (MMU) maps logical address dynamically by adding relocation register.

- Whenever a process contain an instruction to call a procedure at absolute address &when this address is loaded into the partition the address may change. Thus it should not call the actual instruction .This problem is the relocation problem. It can be solved using relocation register. MMU maps the logical address to physical. This physical is calculated by adding the logical address with relocation register.

- With the relocation register the problem of relocation is solved. There is a possibility to exceed the maximum size of partition.To avoid this limit register is used which stores the range of the logical address.
- Each logical address is checked against this register to ensure that it does not attempt to access the memory address outside the allocation partition.

Memory Protection



Non contiguous allocation

- These are main methods

1. paging

2. segmentation

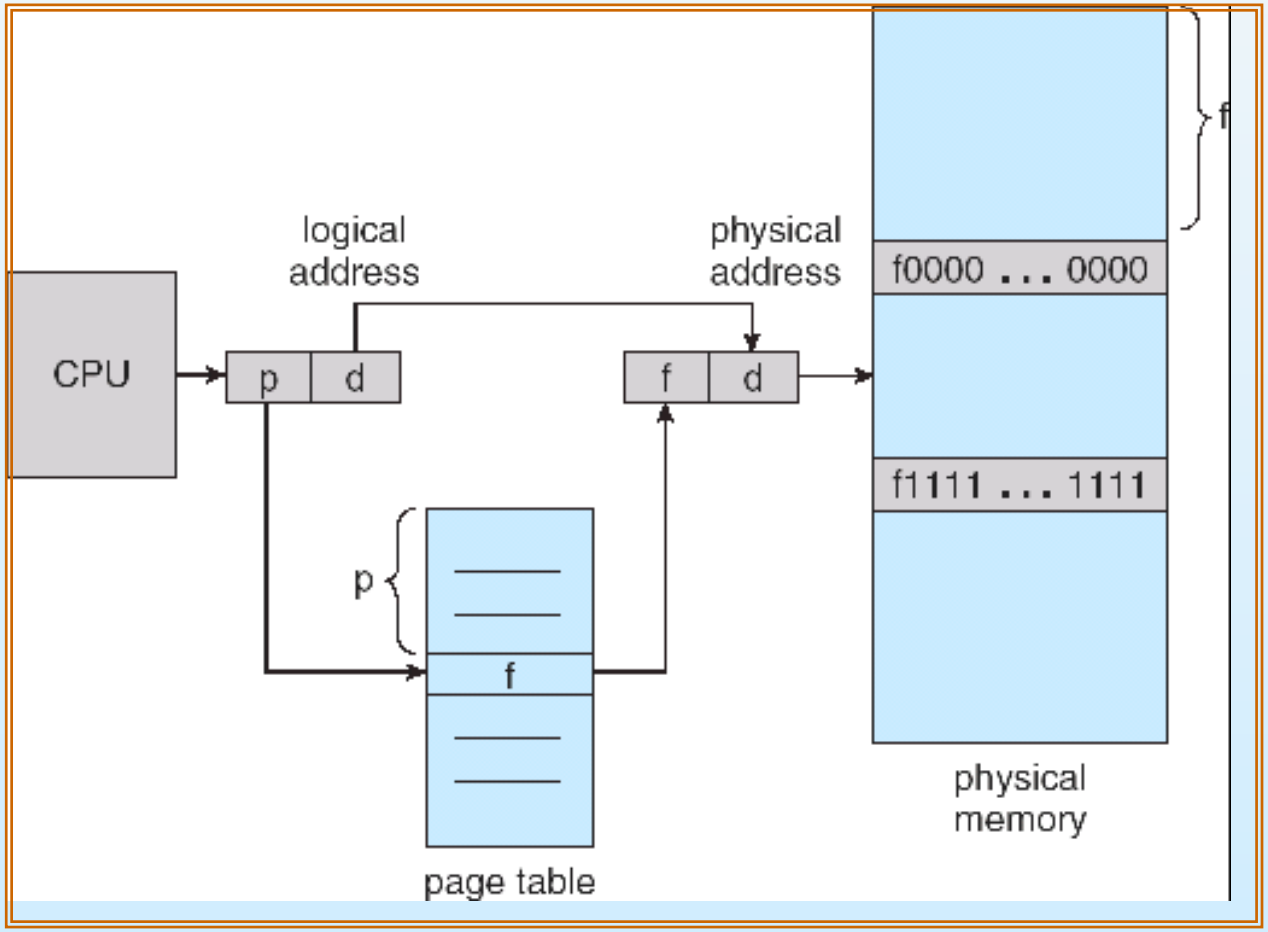
Paging

- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available
- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 8192 bytes)
- Divide logical memory into blocks of same size called **pages**.
- Keep track of all free frames
- Set up a page table to translate logical to physical addresses
- Mapping between virtual address space and physical address space
- Internal fragmentation

Address Translation Scheme

- Address generated by CPU is divided into:
  - *Page number (p)* – used as an index into a *page table* which contains base address of each page in physical memory
  - *Page offset (d)* – combined with base address to define the physical memory address that is sent to the memory unit

Paging Hardware Support

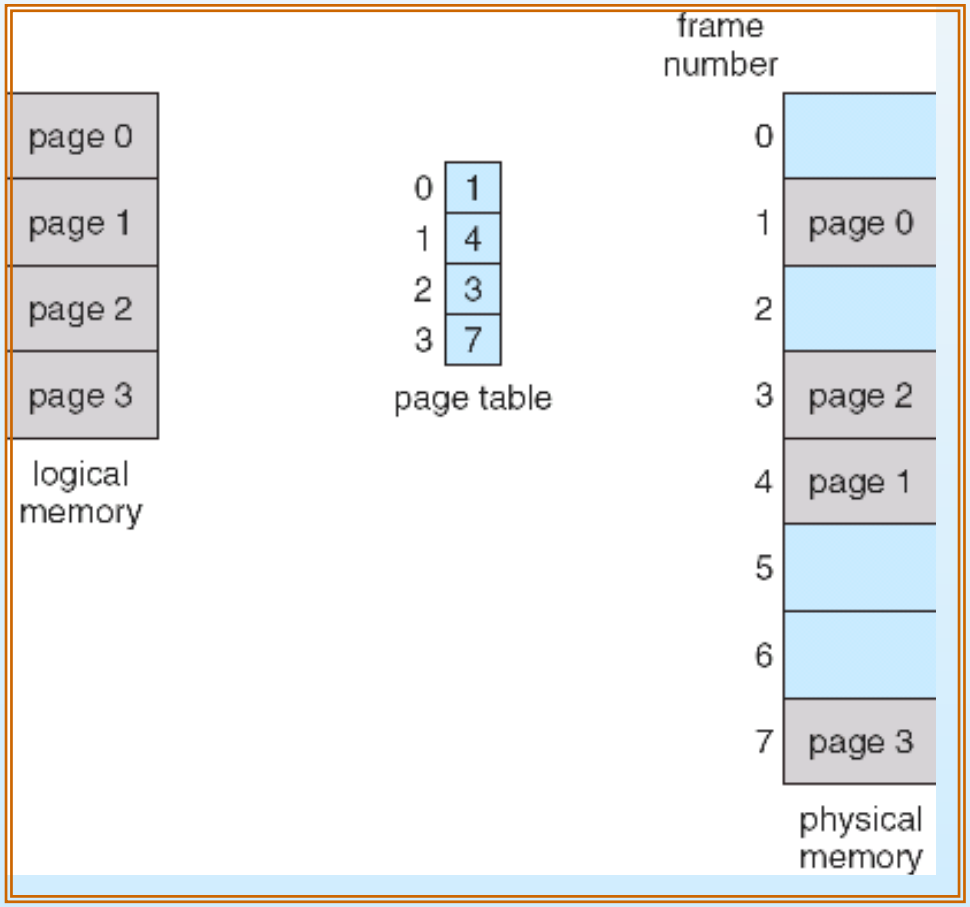


Paging operation- principle

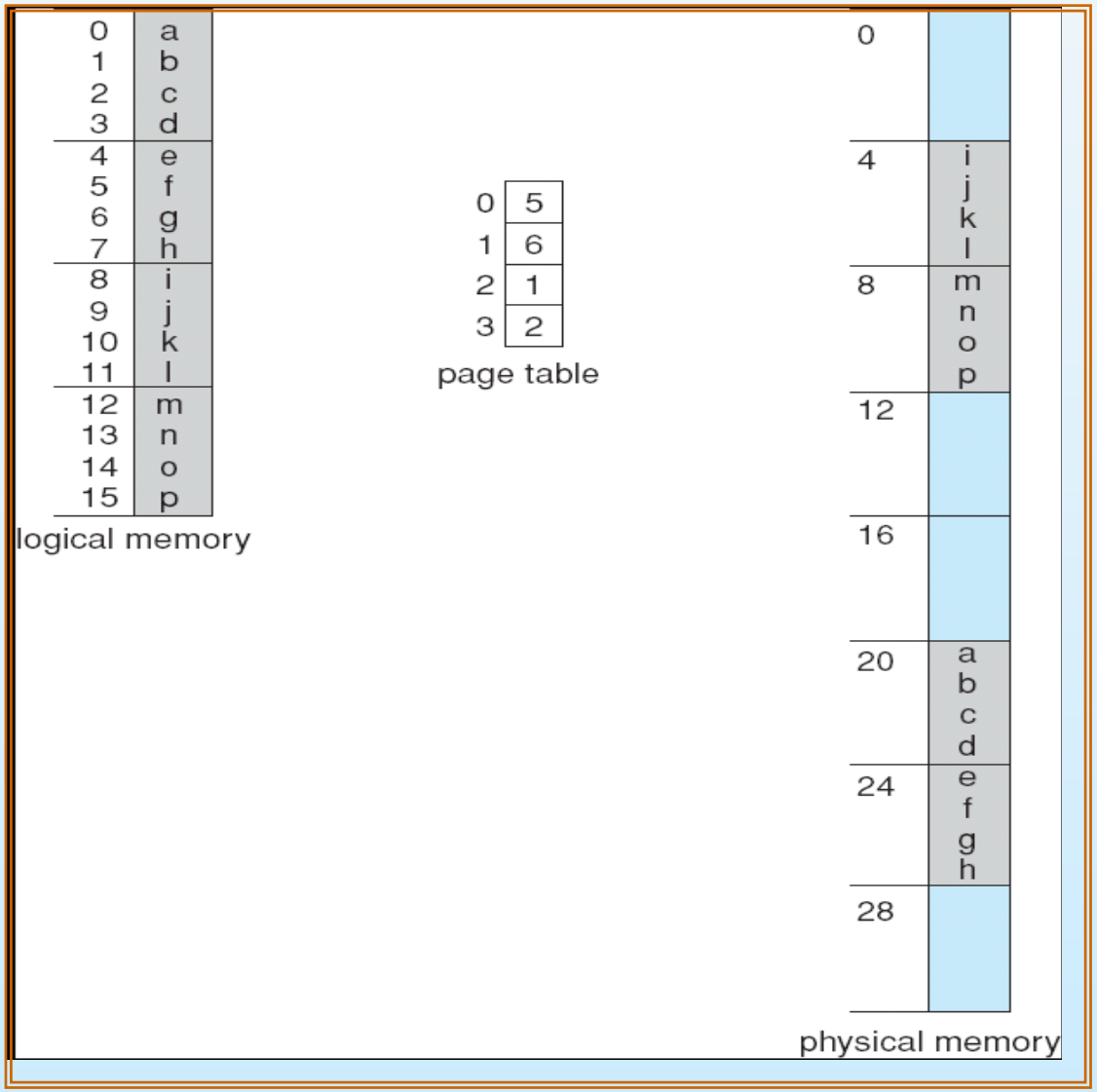
During the process execution, CPU generates a logical(virtual) address ,that comprises of page number(p) and offset (d) with in the page. P is used to index in to the page table and fetch corresponding frame number .The physical address is obtained by combining frame number(f) with the offset (d)

- Note that page size and frame size is defined between 512 bytes to 4KB depending on computer
- Page size is usually a power of 2(2^n)bytes

Paging Example



Paging Example



Segmentation

- A segment can be defined as a logical grouping of instructions (ie sub routine , array.....etc)
- Memory-management scheme that supports user view of memory
- A program is a collection of segments. No ordering among segments.
- A segment is a logical unit such as:

  main program,

  procedure,

  function,

  method,

  object,

  local variables, global variables,

  common block,

  stack,

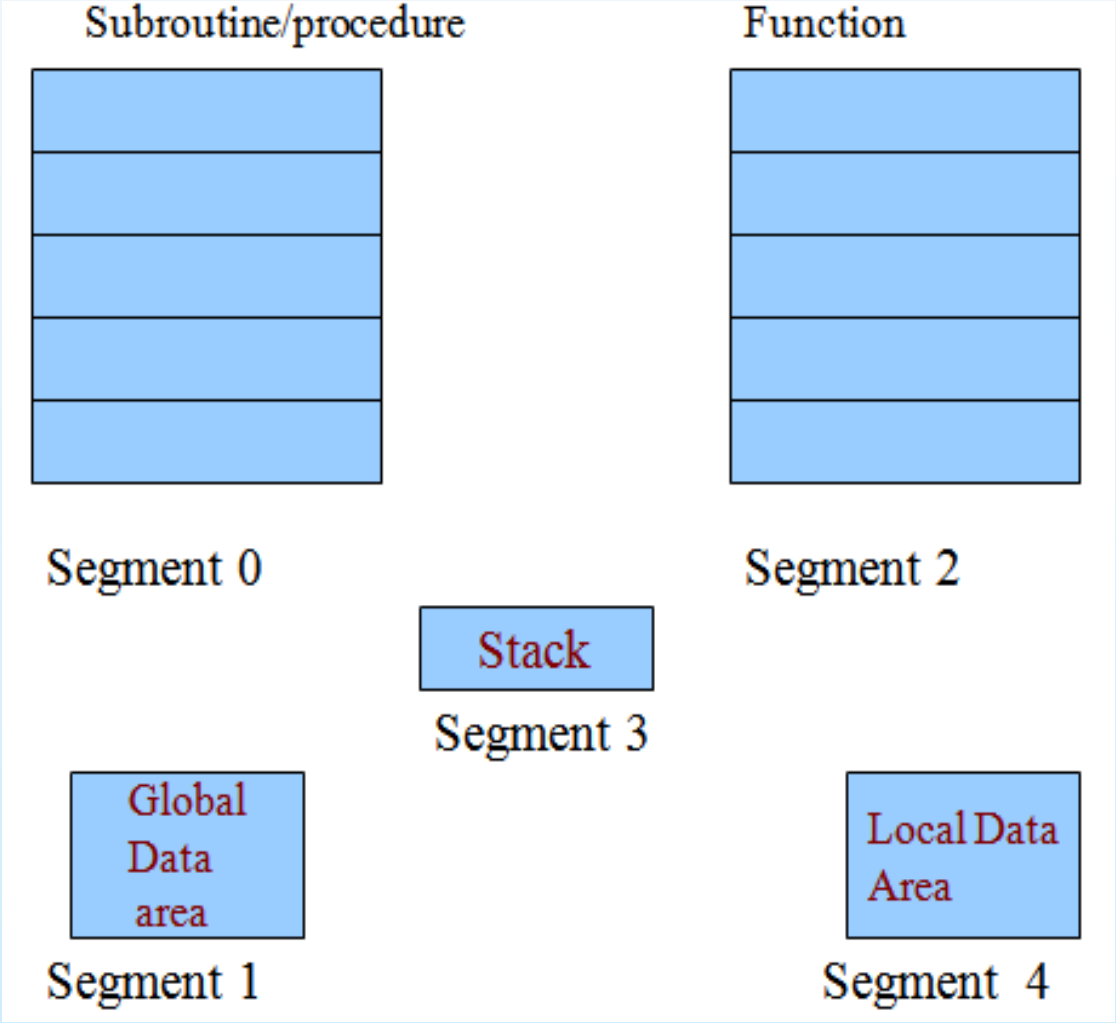
  symbol table, arrays

Segmentation

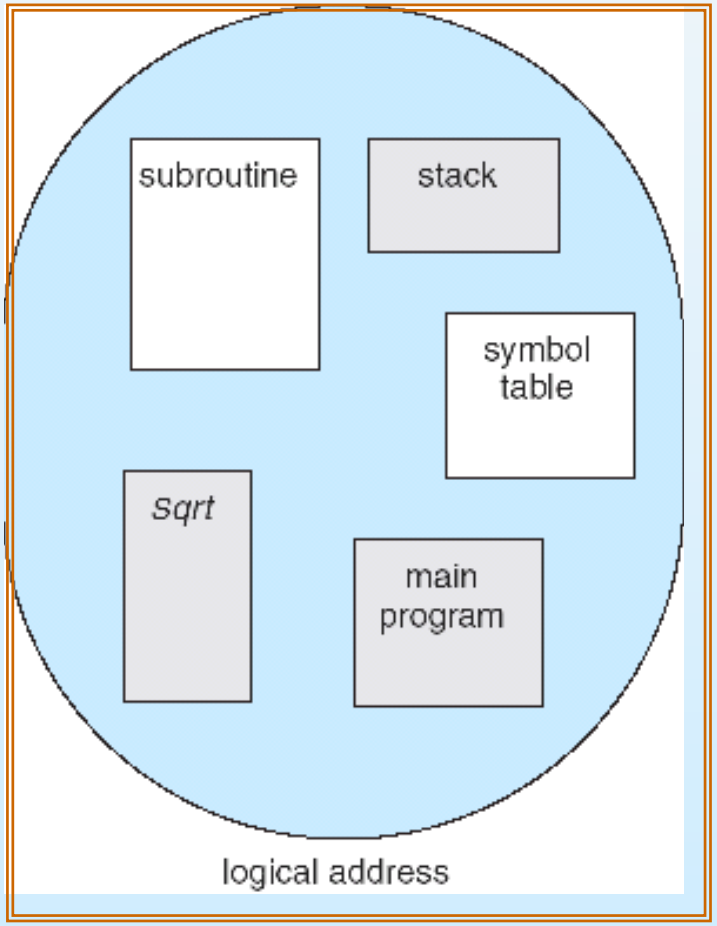
- Segments are formed at program translation time by grouping together logically related entities.

Note: The process of formation of these segments vary from one system to another

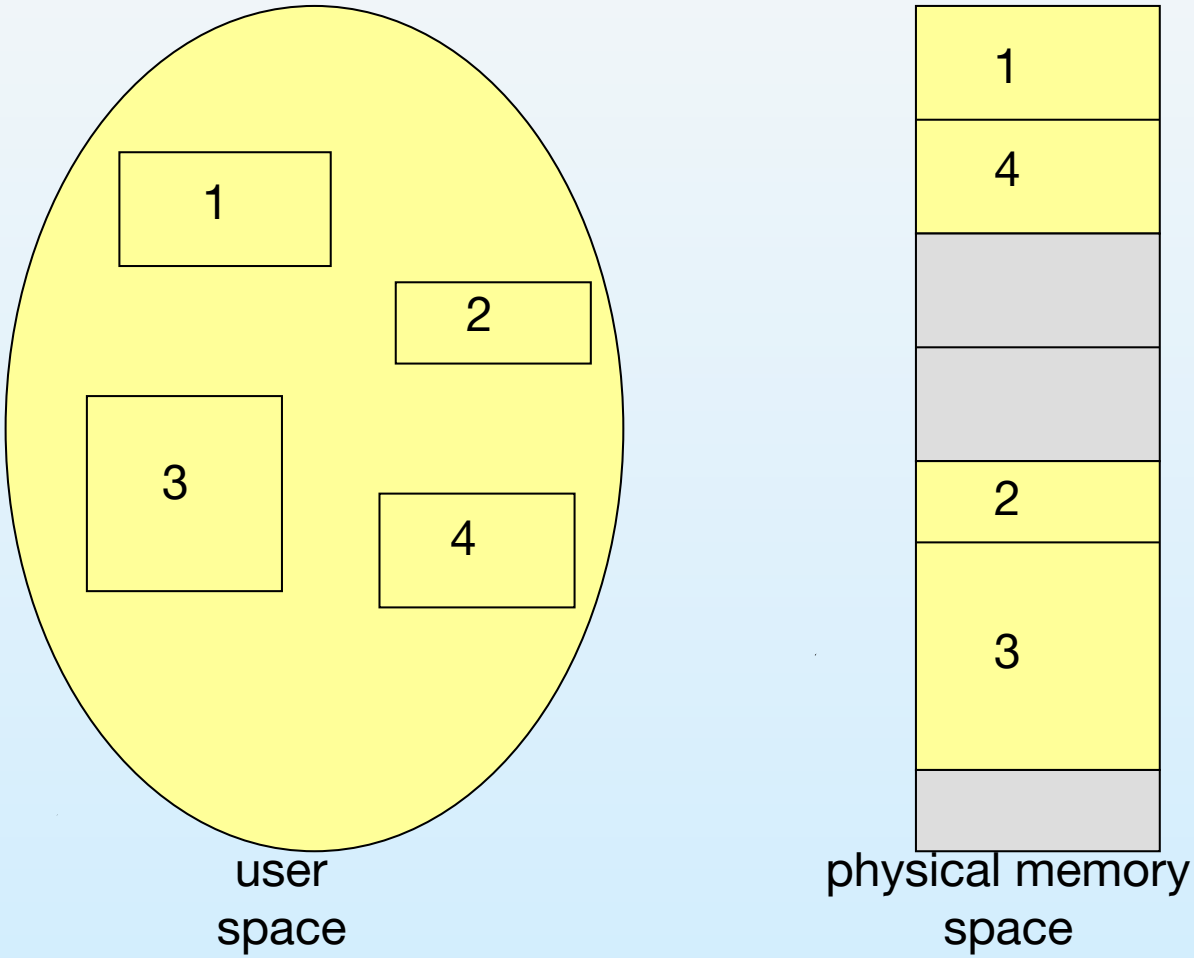
User's view of a program



User's View of a Program



Logical View of Segmentation



Segmentation Architecture

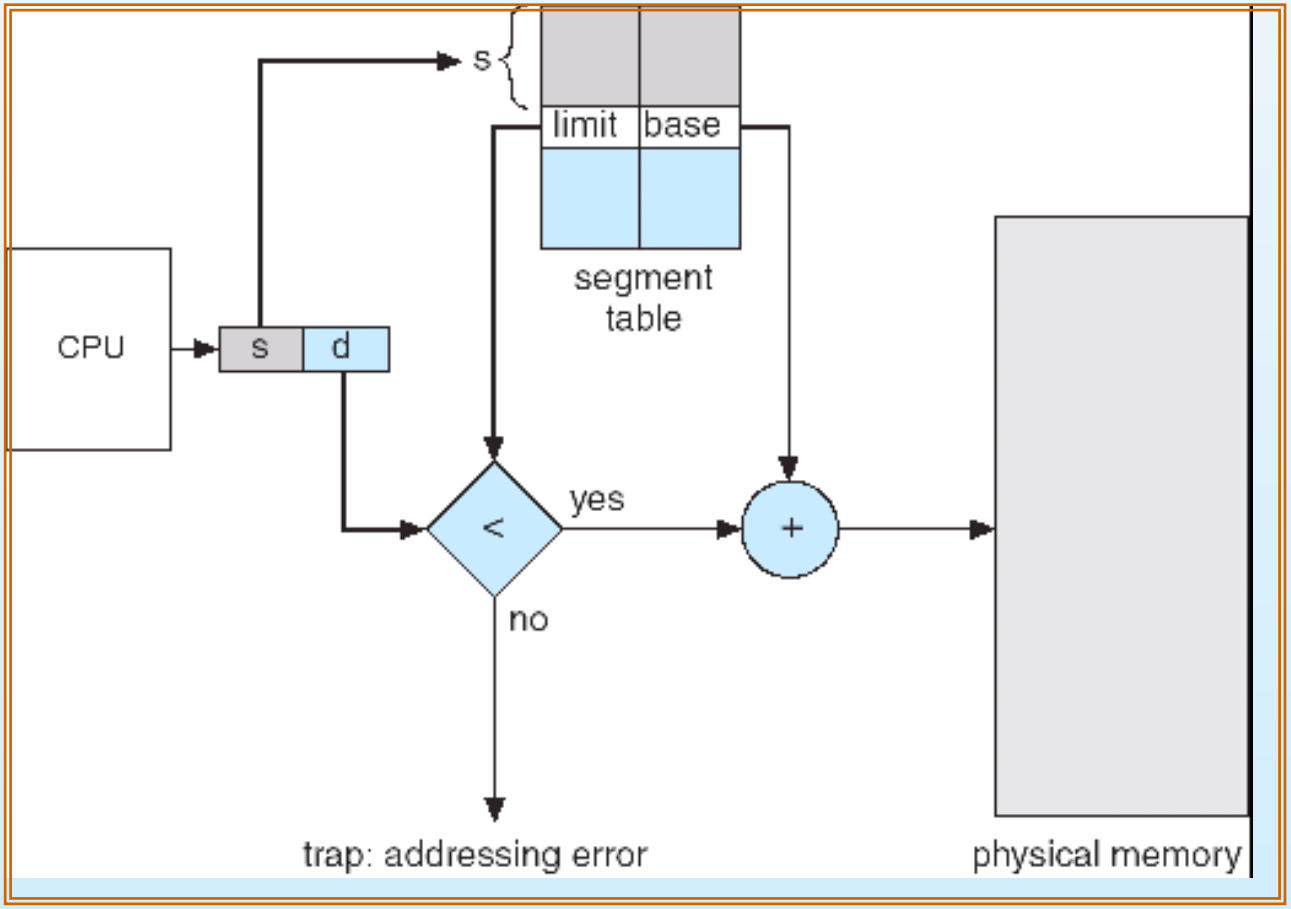
- Logical address consists of a two tuple:

  <segment-number, offset>,

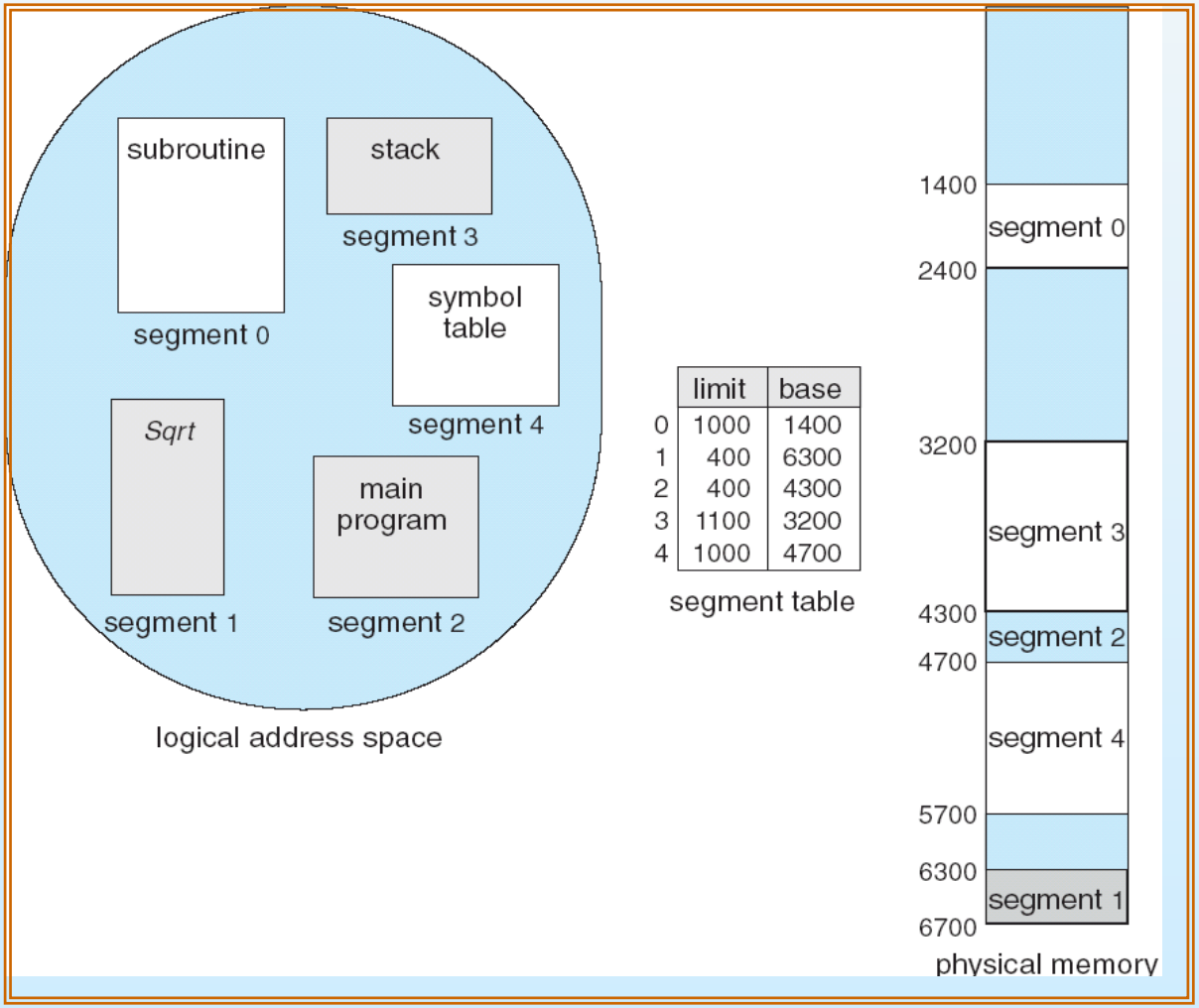
- **Segment table** – maps two-dimensional physical addresses; each table entry has:
  - *base* – contains the starting physical address where the segments reside in memory
  - *limit* – specifies the length of the segment
- *Segment-table base register (STBR)* points to the segment table's location in memory
- *Segment-table length register (STLR)* indicates number of segments used by a program;

  segment number *s* is legal if *s* < STLR

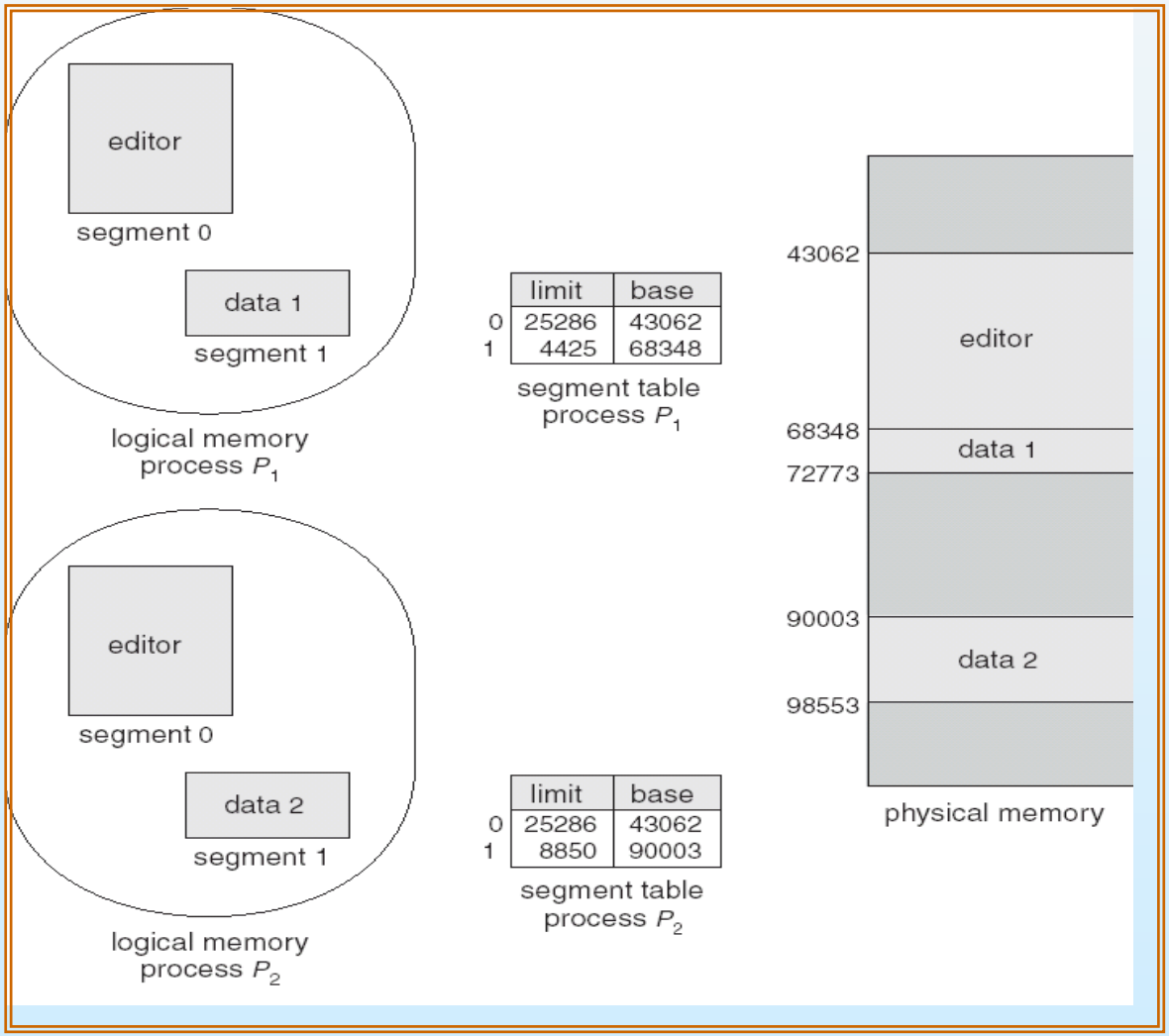
Address Translation Architecture



Example of Segmentation



Sharing of Segments



Segmentation

- Advantages:
  - Sharing of code or data is possible
  - Protection bits are associated with segments
  - Segments can grow dynamically also
  - It supports virtual memory
  - Dynamic linking and loading is possible here

Segmentation

- Disadvantages:

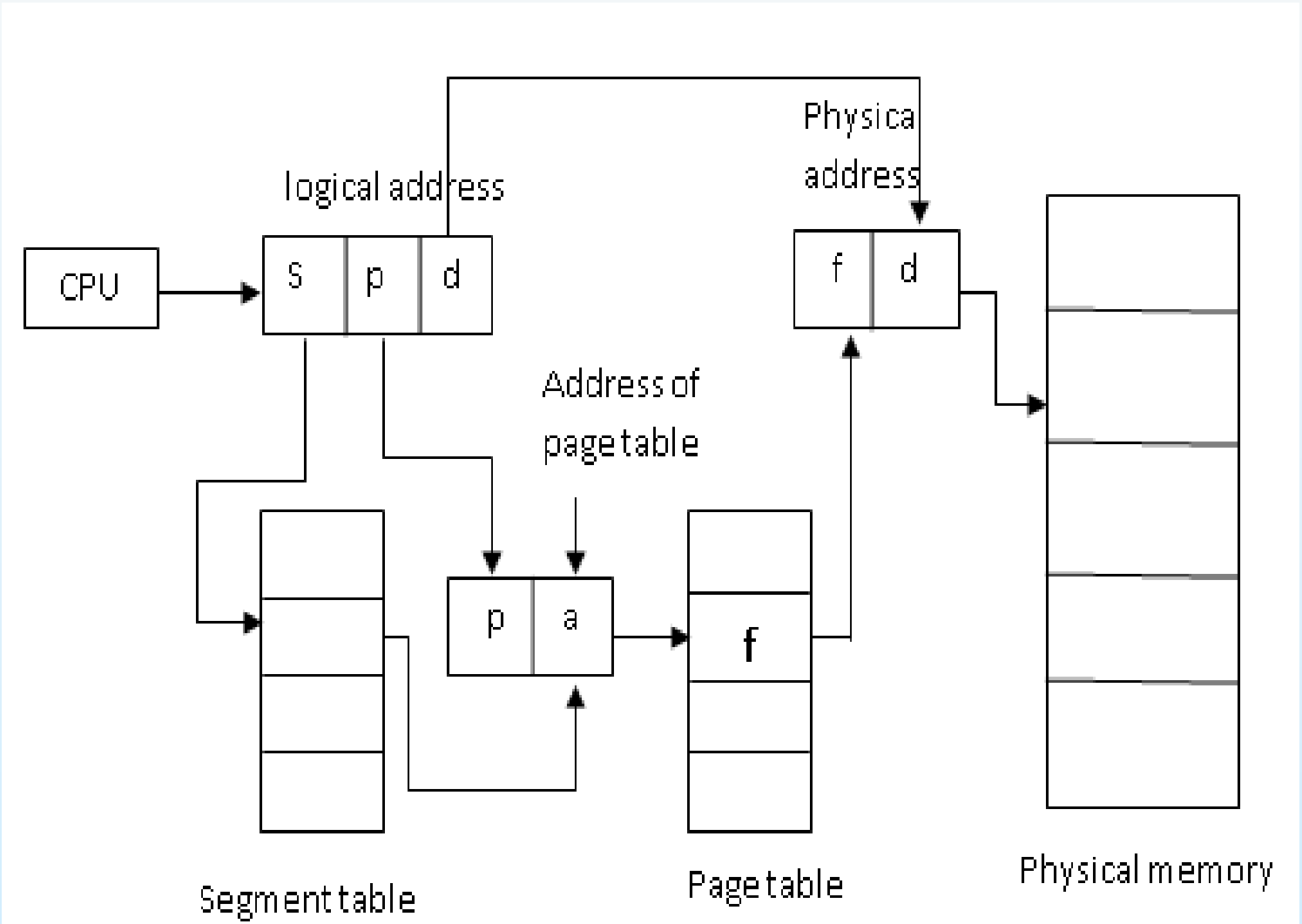
Mapping requires two memory references as like paging (segment table info / data). It will slows the system

- Dynamic growth will leads to store the segment table in main memory . (registers are not feasible here). So high capacity H/W support is needed

Segmentation with Paging

- Combine the advantages of both paging & segmentation together into a single scheme.
- Each segment divided into a number of pages
- A page table maintained for each segment & it keep track of these pages
- Logical address : segment number (s) & segment offset
- Segment offset : page number (p) & page offset (d)
- Segment table entry : segment base, segment limit & address of segment’s page table (a)

- MMU uses the segment number as an index to the segment table to find the address of the page table.
- The page number of logical address is attached to the high-order end of the page table address & used as an index to page table to find the page table entry.
- The physical address formed by attaching the frame number from page table entry to the high-order end of the page offset.


Physica
address
logical address
CPU
S
p
d
f
d
Address of
page table
p
a
f
Segment table
Page table
Physical memory