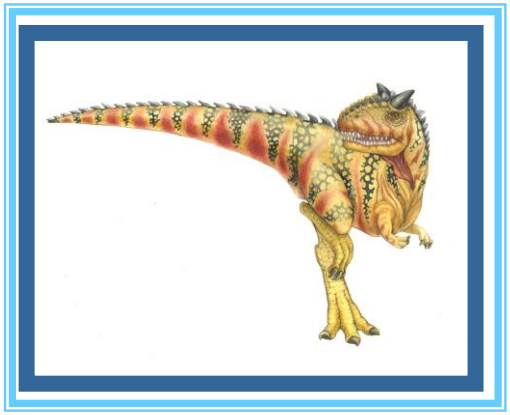
Memory Management
Day6: Sep 2021

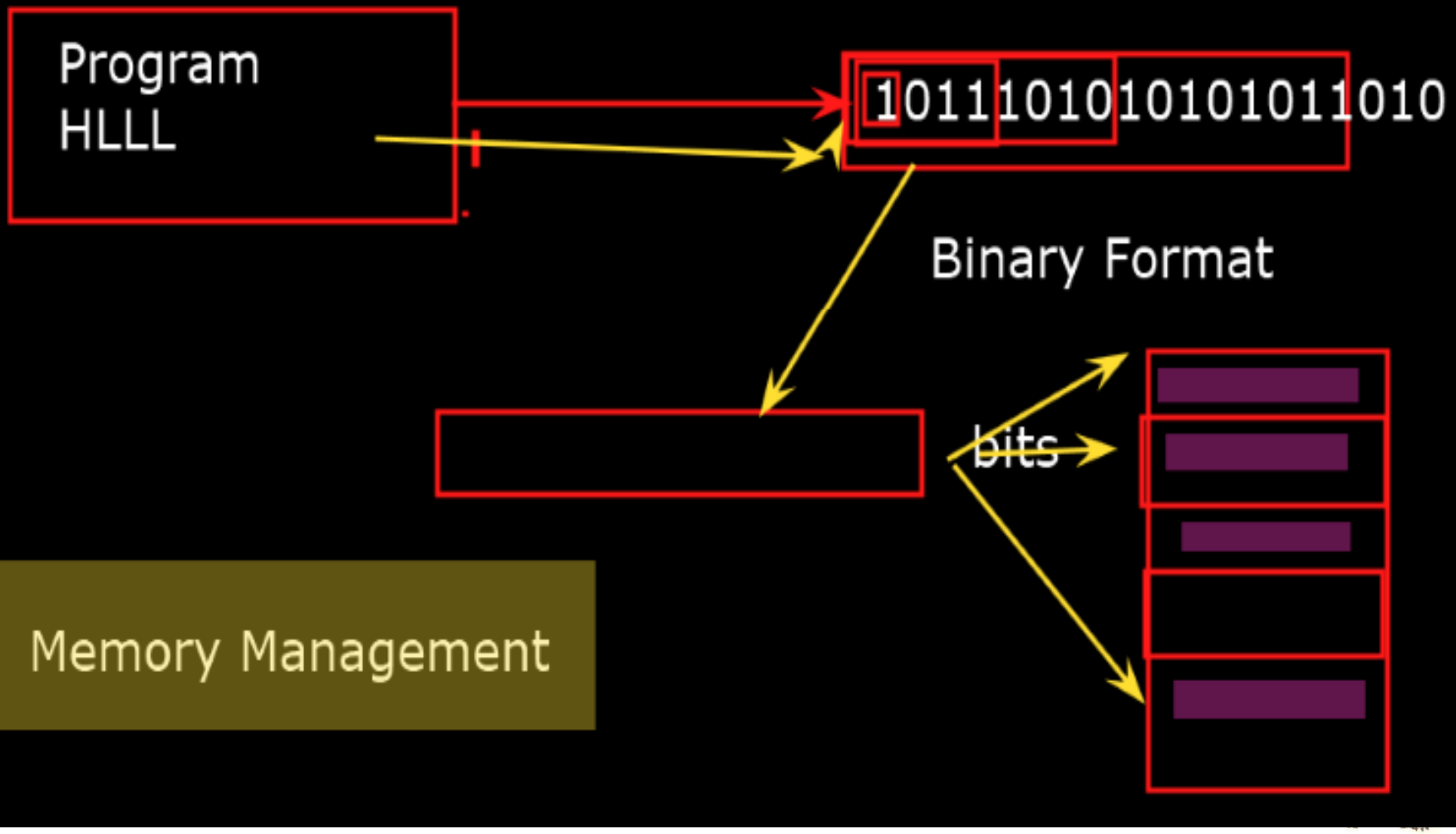
Kiran Waghmare



- Memory Management
  - continuous & dynamic
  - best fit, worst fit, first fitexternal internal fragmentation
  - segmentation
  - paging
  - concept of dirty bit
  - shared pages & reentrant
  - Throttling

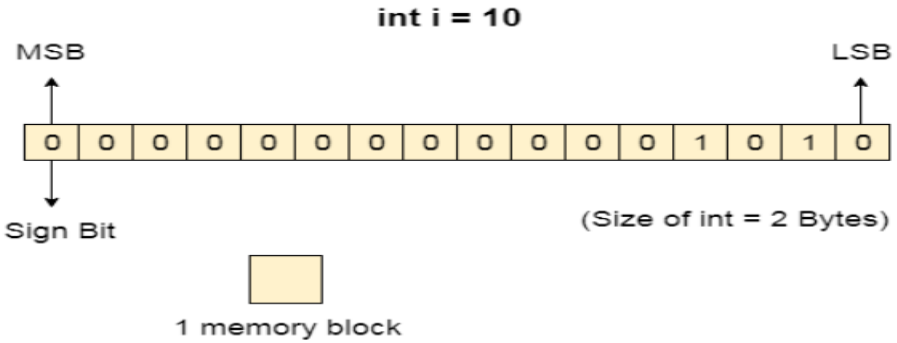
Memory:

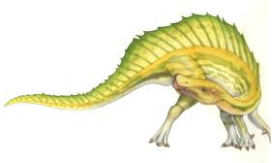
-collection of some data which is represented in the binary format.



What is Memory?

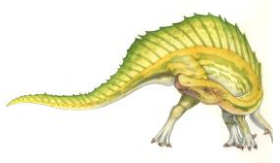
- Computer memory can be defined as a **collection of some data represented in the binary format.**
- On the basis of various functions, memory can be classified into various categories.
- Machine **understands only binary language that is 0 or 1**.
- Computer converts every data into binary language first and then stores it into the memory.





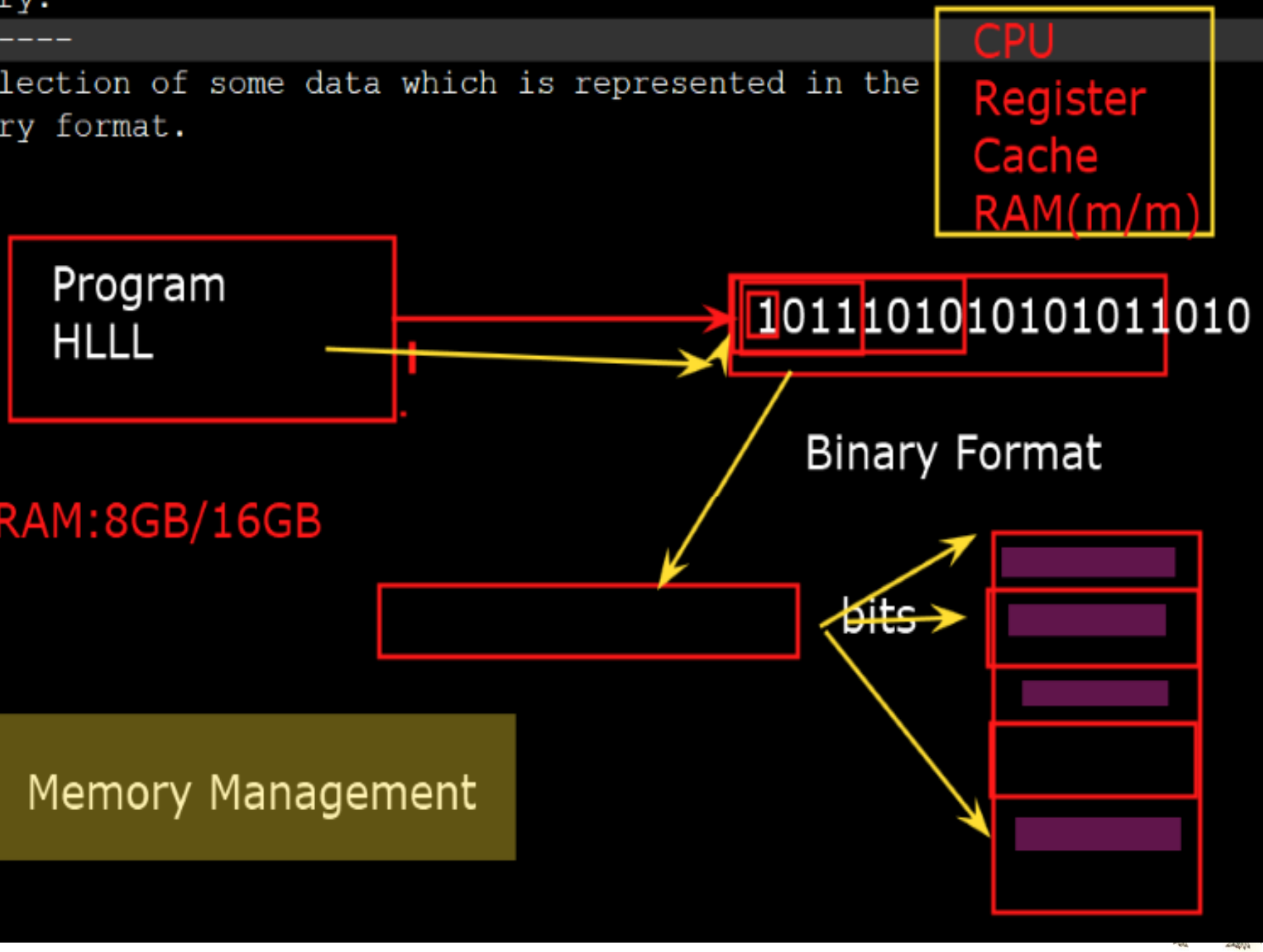
What is Main Memory:

- The main memory is **central to the operation of a modern computer**.
- Main Memory is a **large array of words or bytes, ranging in size from hundreds of thousands to billions.**
- Main memory is a **repository of rapidly available information** shared by the CPU and I/O devices.
- Main memory is the place where **programs and information are kept** when the processor is effectively utilizing them.
- Main memory is **associated with the processor, so moving instructions and information i**nto and out of the processor is extremely fast.
- Main memory is also known **as RAM(Random Access Memory).**
- This memory is a **volatile memory**.
- RAM lost its data when a power interruption occurs.



Memory:

-collection of some data which is represented in the binary format.

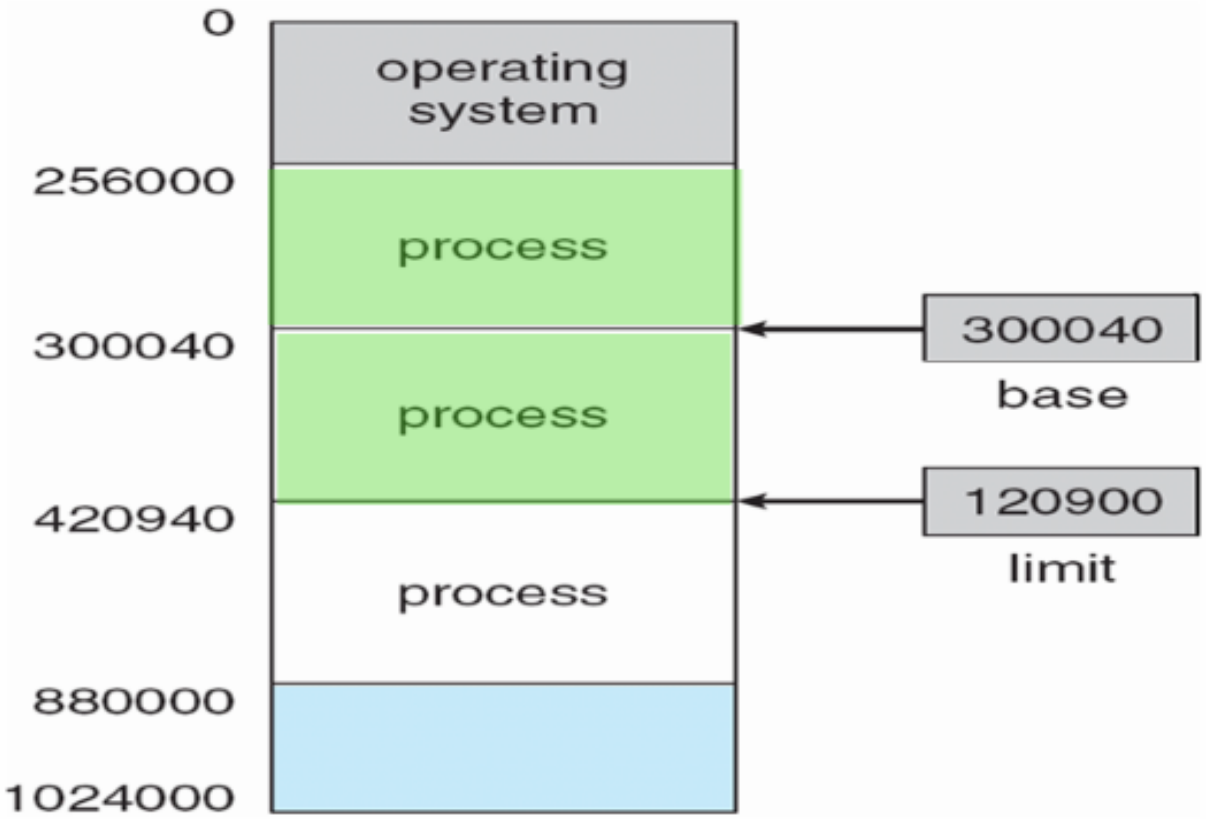


Background

- Program must **be brought (from disk) into memory** and placed within a process for it to be run
- **Main memory and registers are only storage CPU** can access directly
- Register access in **one CPU clock** (or less)
- Main memory **can take many cycles**
- **Cache** sits between main memory and CPU registers
- Protection of memory required to ensure correct operation

Base and Limit Registers

- A pair of **base** and **limit** registers define the logical address space



Memory:

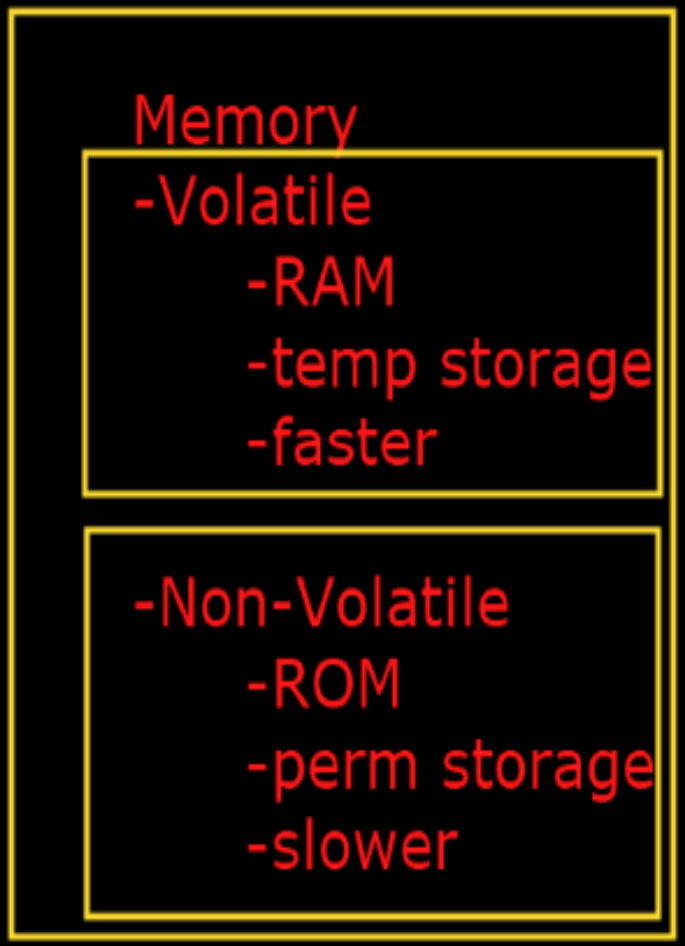
-collection of some data which is represented in the binary format.

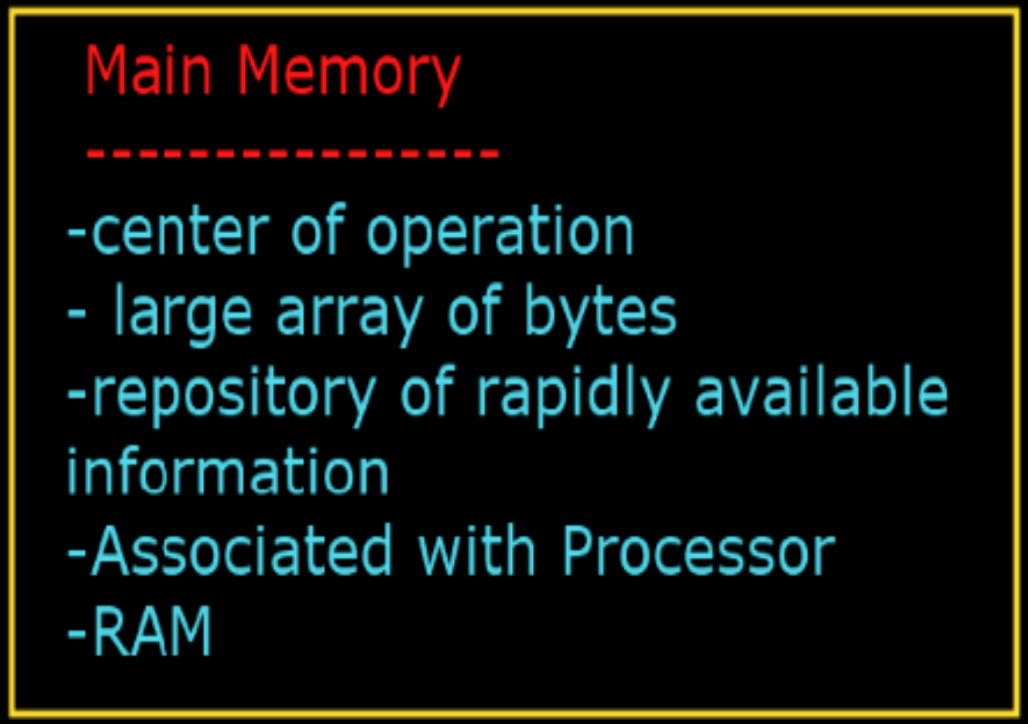
Memory
-Volatile
-RAM
-temp storage
-faster

-Non-Volatile
-ROM
-perm storage
-slower

Memory:

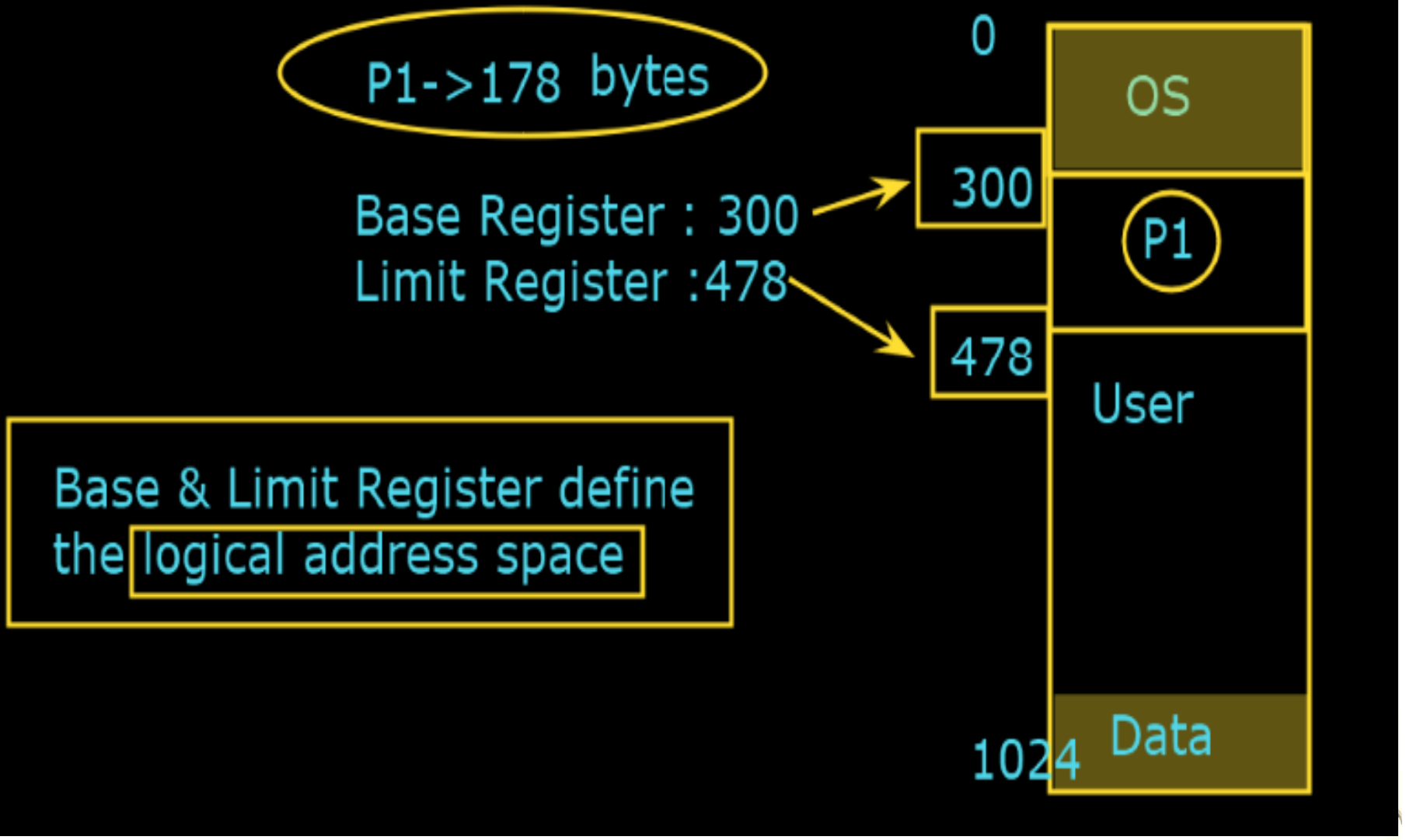
-collection of some data which is represented in the binary format.





Memory:

-collection of some data which is represented in the binary format.

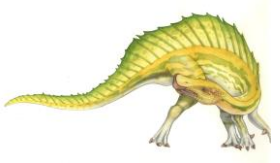


Need for Memory Management in OS

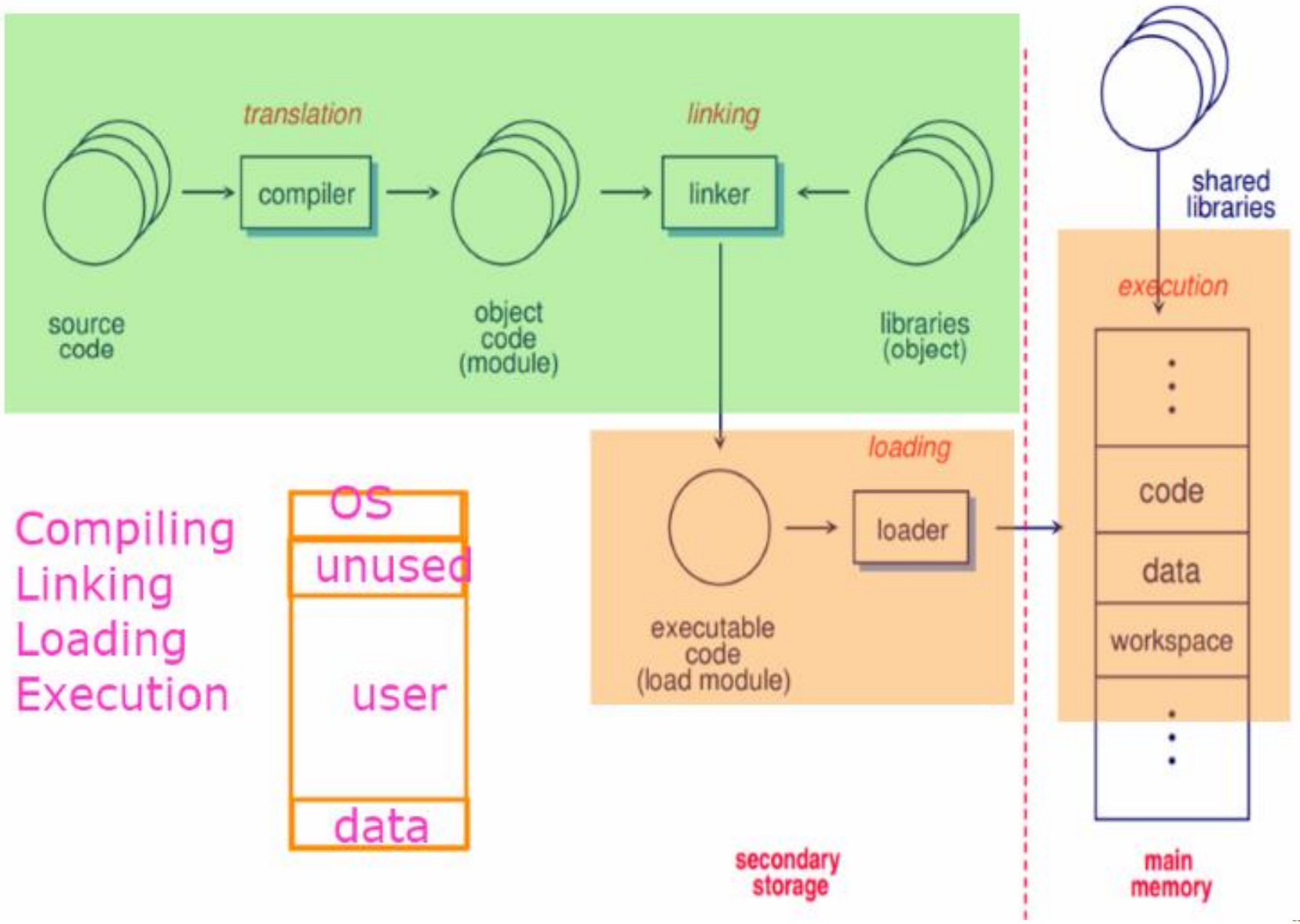
- Memory management technique is needed for the following reasons:
  - This technique **helps in placing the programs in memory** in such a way so that memory is utilized at its fullest extent.
  - This technique **helps to protect different processes from each other** so that they do not interfere with each other's operations.
  - It helps to **allocate space to different application routines**.
  - This technique allows you **to check how much memory needs** to be allocated to processes that decide which processor should get memory at what time.
  - It **keeps the track of each memory location** whether it is free or allocated.
  - This **technique keeps the track of inventory whenever memory gets freed or unallocated** and it will update the status accordingly.

Why Memory Management is required:

- **Allocate and de-allocate memory** before and after process execution.
- To **keep track of used memory** space by processes.
- To **minimize fragmentation** issues.
- To **proper utilization** of main memory.
- To **maintain data integrity** while executing of process.



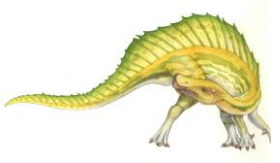
From *source* to *executable* code



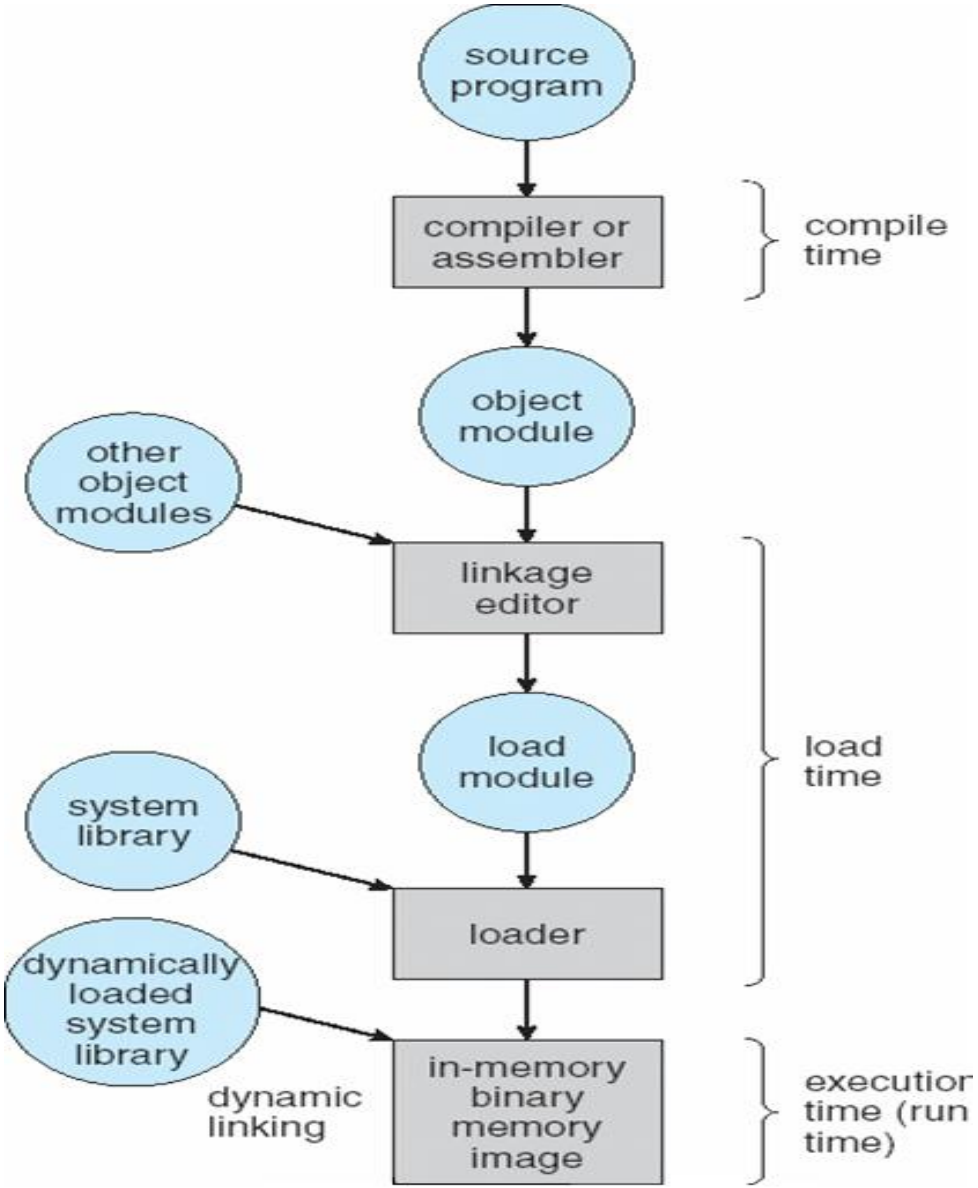
Compiling
Linking
Loading
Execution

Binding of Instructions and Data to Memory

- Address binding of instructions and data to memory addresses can happen at three different stages
  - **Compile time**: If memory location known a priori, **absolute code** can be generated; must recompile code if starting location changes
  - **Load time**: Must generate **relocatable code** if memory location is not known at compile time
  - **Execution time**: Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., base and limit registers)



Multistep Processing of a User Program

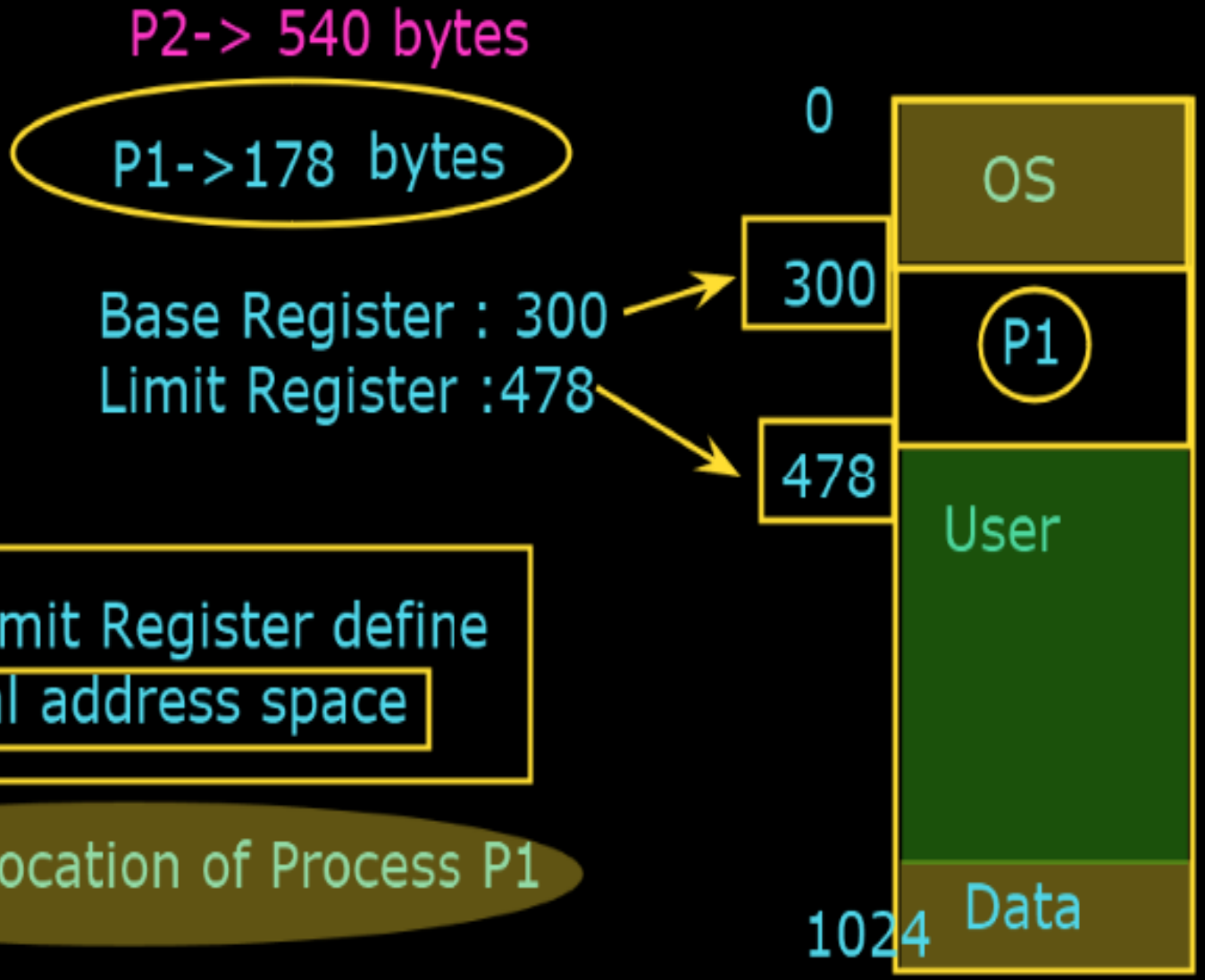


Logical vs. Physical Address Space

- The concept of a logical address space that is bound to a separate **physical address space** is central to proper memory management
  - **Logical address** – generated by the CPU; also referred to as **virtual address**
  - **Physical address** – address seen by the memory unit
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme

Memory:

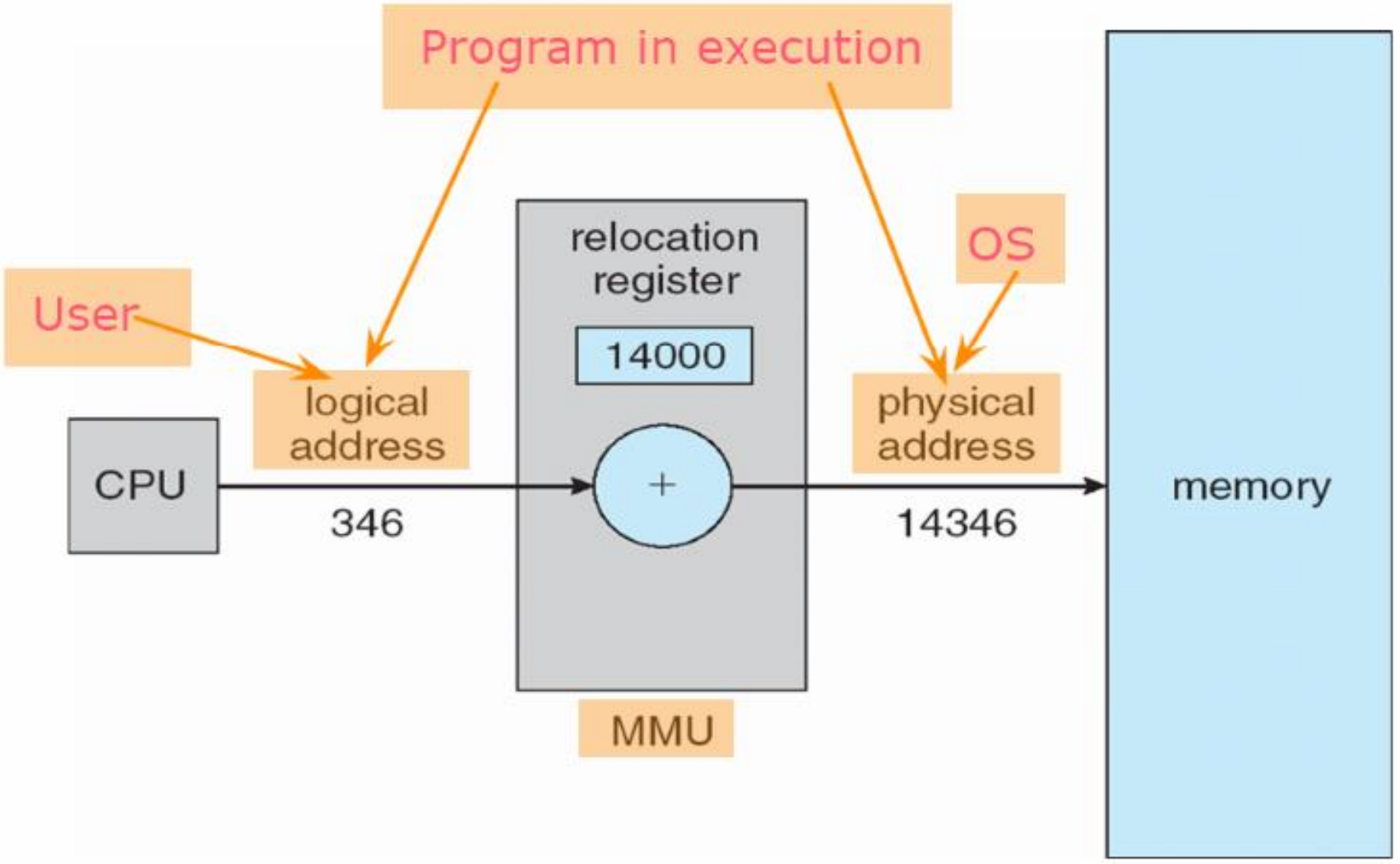
-collection of some data which is represented in the binary format.



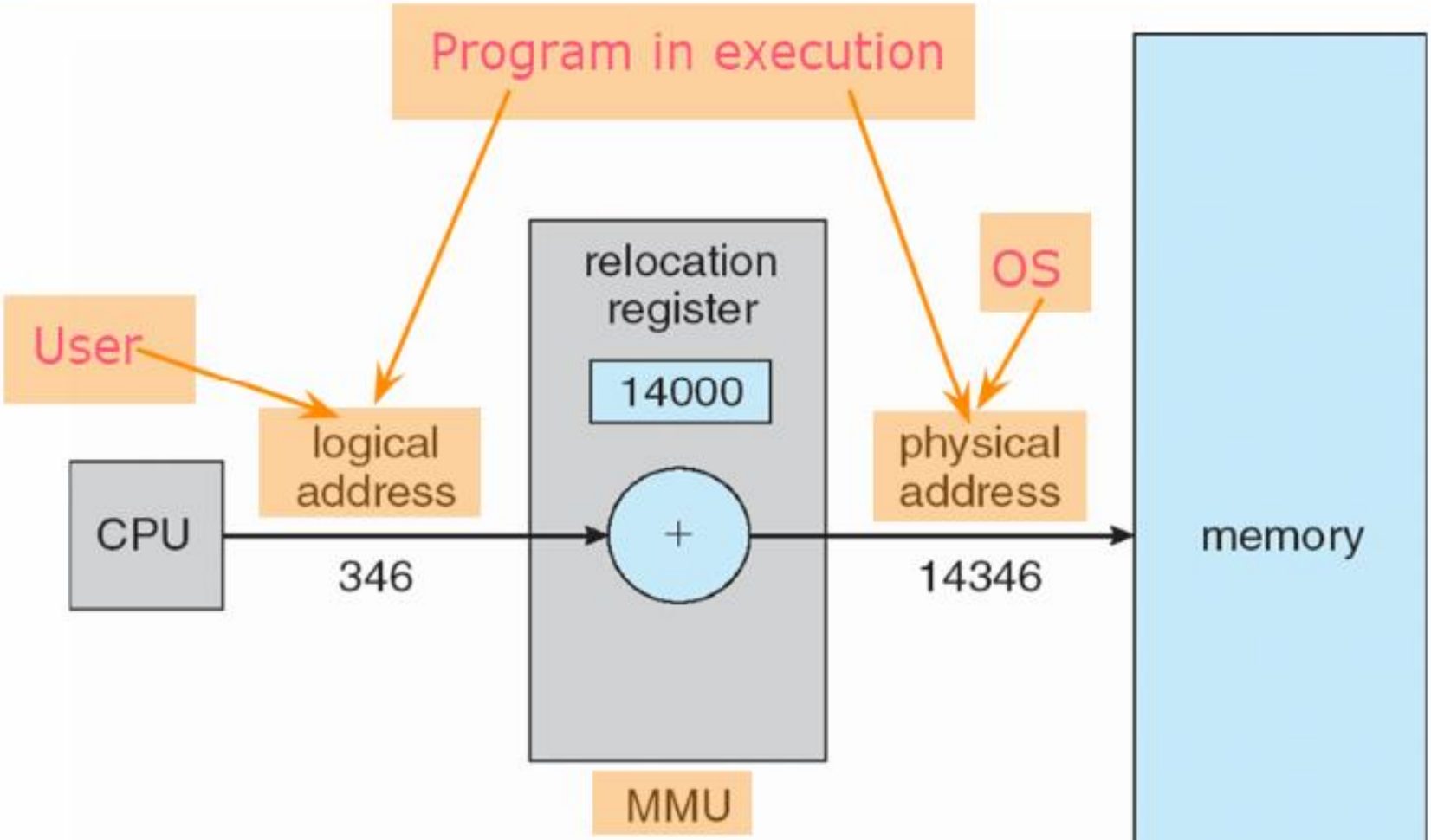
Memory-Management Unit (MMU)

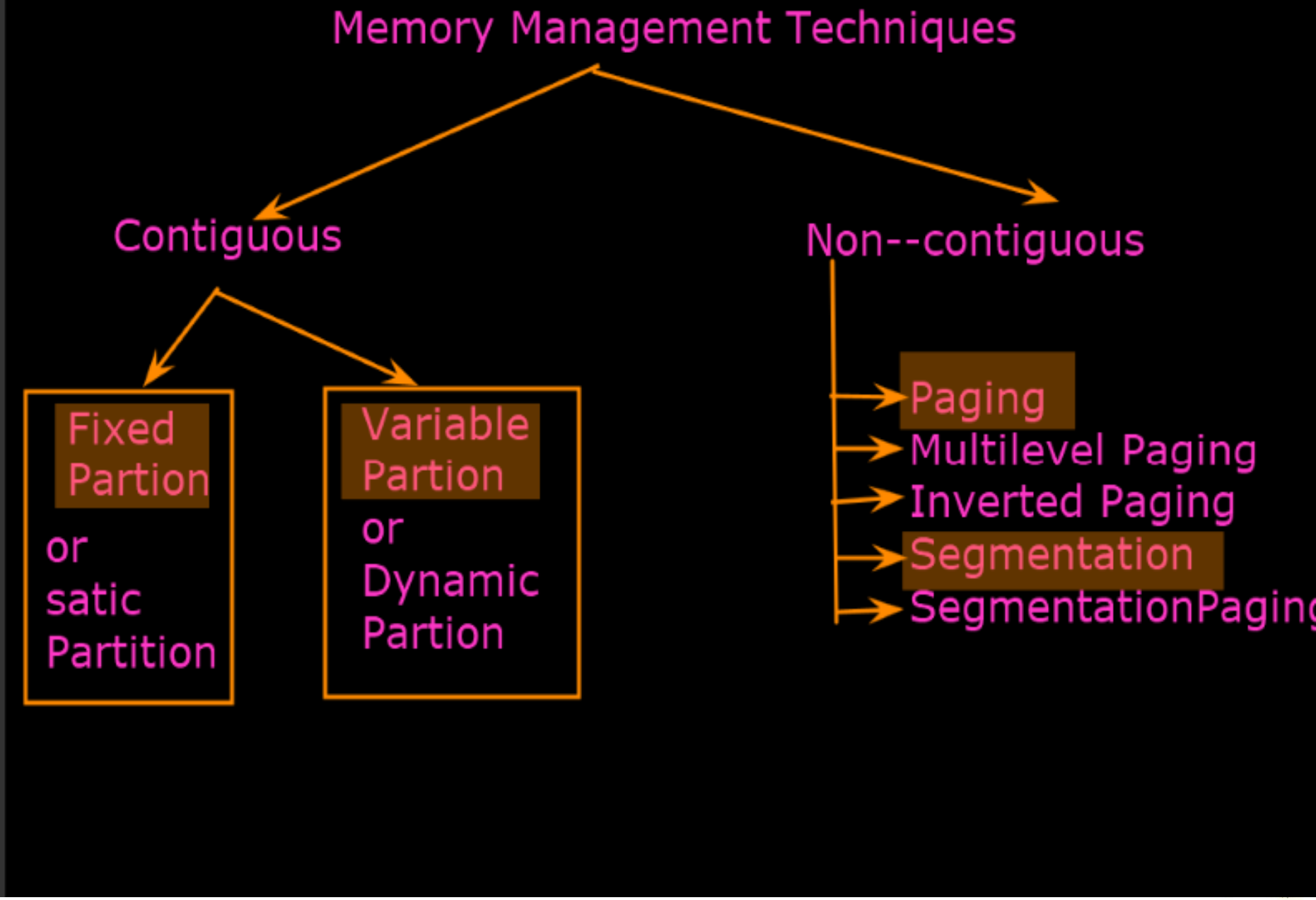
- Hardware device that **maps virtual to physical address**
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- The user program deals with ***logical* addresses;** it never sees the *real* physical addresses

Dynamic relocation using a relocation register



Dynamic relocation using a relocation register





Contiguous Allocation

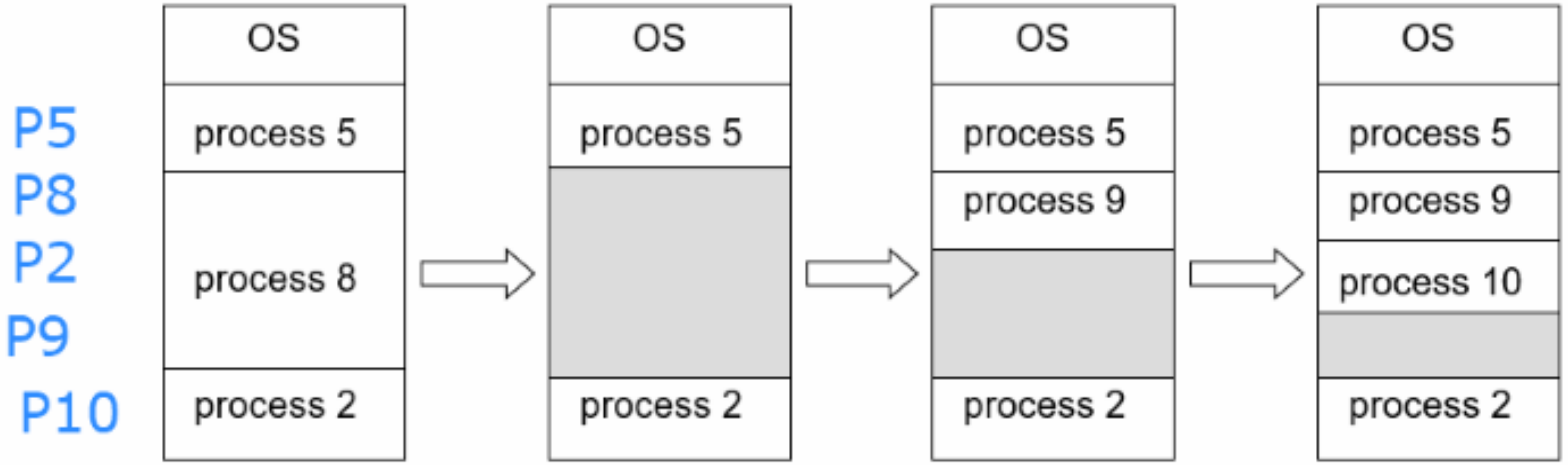
- Main memory usually into two partitions:
  - Resident operating system, usually held in low memory with interrupt vector
  - User processes then held in high memory

- Relocation registers used to protect user processes from each other, and from changing operating-system code and data
  - Base register contains value of smallest physical address
  - Limit register contains range of logical addresses – each logical address must be less than the limit register
  - MMU maps logical address *dynamically*

Contiguous Allocation (Cont)

- Multiple-partition allocation
  - Hole – block of available memory; holes of various size are scattered throughout memory
  - When a process arrives, it is allocated memory from a hole large enough to accommodate it
  - Operating system maintains information about:
    a) allocated partitions b) free partitions (hole)

First-fit
Best-fit
Worst-fit
Next-fit



Dynamic Storage-Allocation Problem

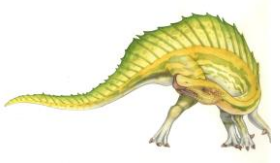
How to satisfy a request of size *n* from a list of free holes

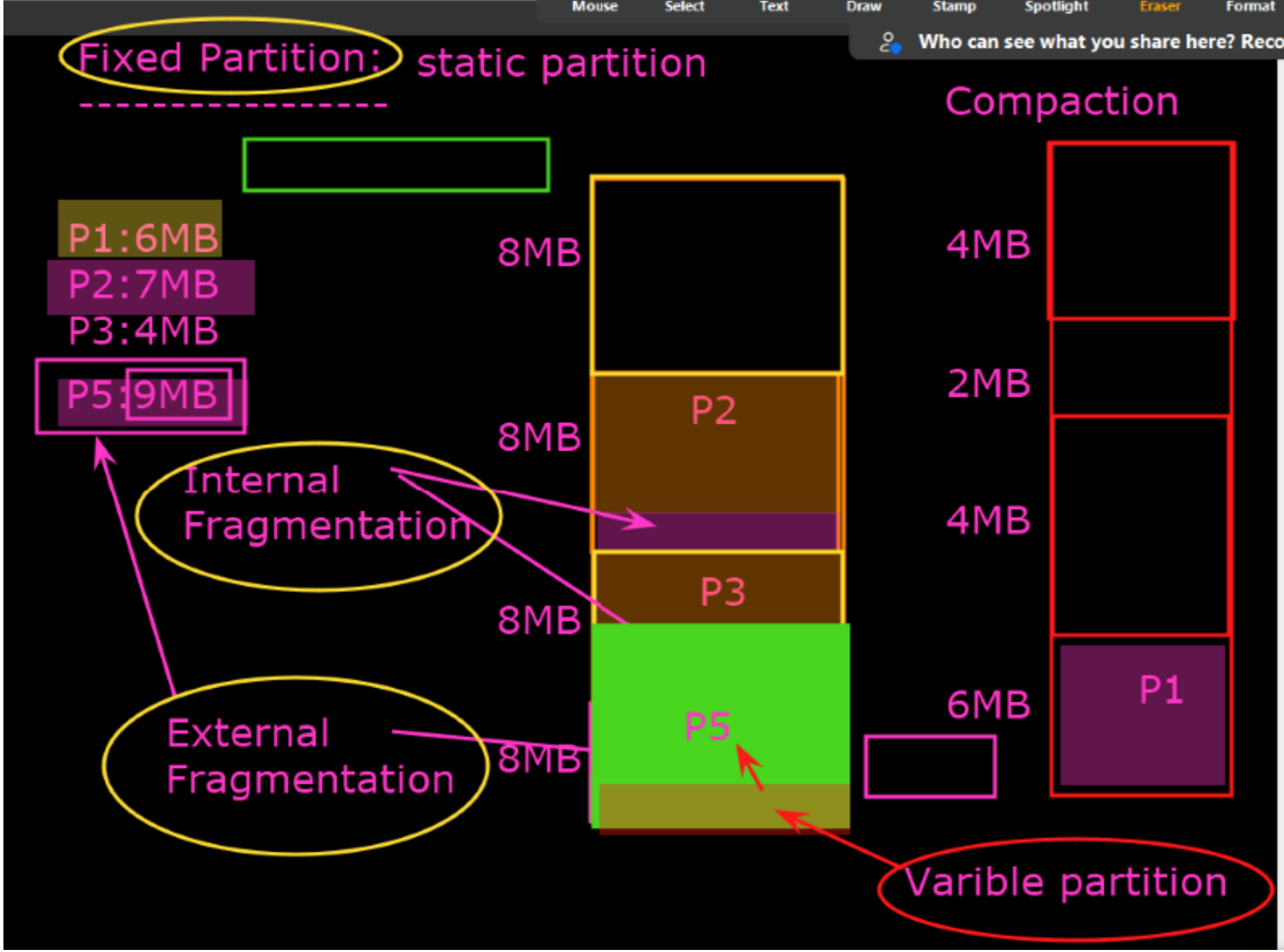
- **First-fit**: Allocate the *first* hole that is big enough
- **Best-fit**: Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size
  - Produces the smallest leftover hole
- **Worst-fit**: Allocate the *largest* hole; must also search entire list
  - Produces the largest leftover hole

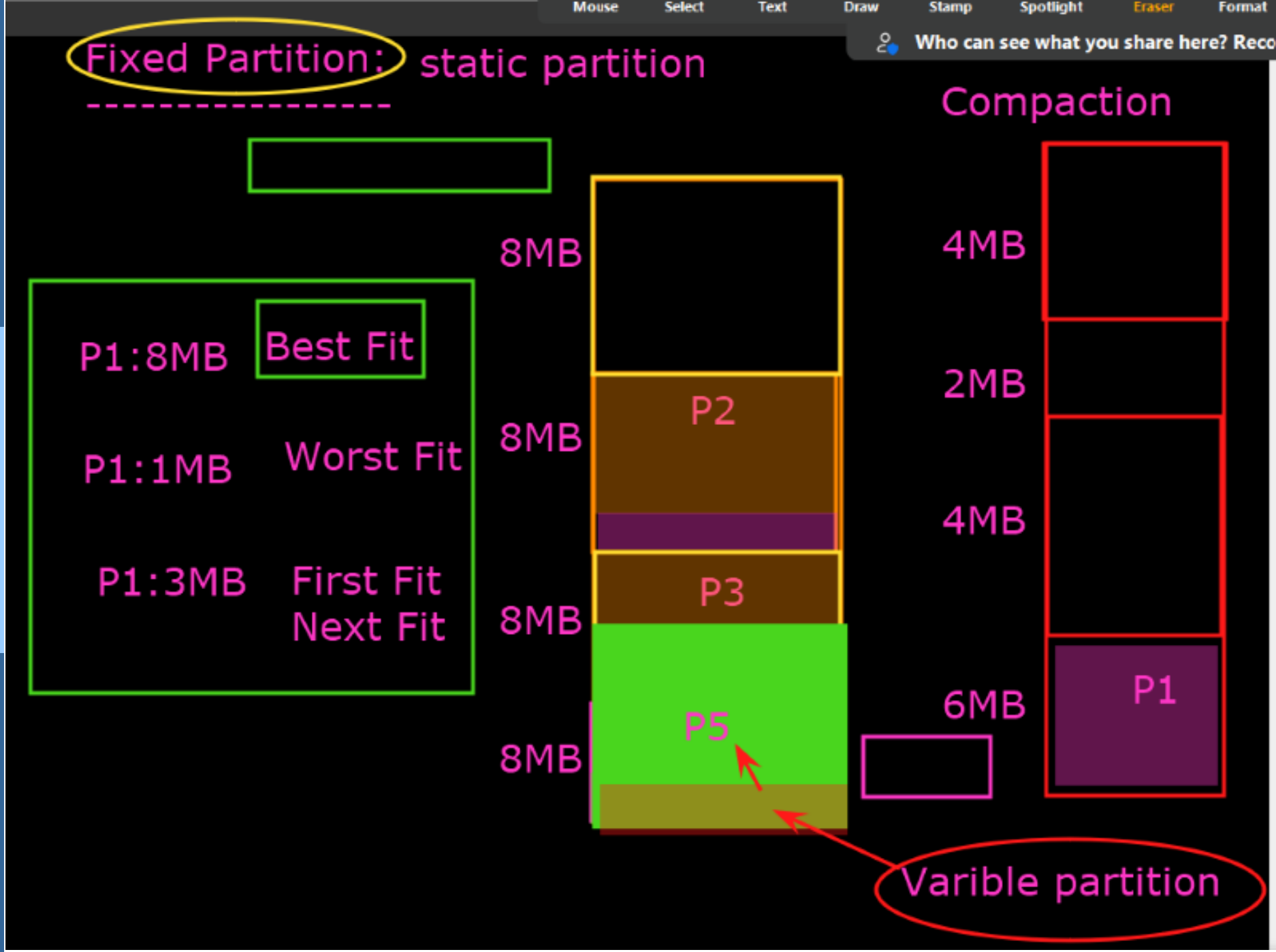
First-fit and best-fit better than worst-fit in terms of speed and storage utilization

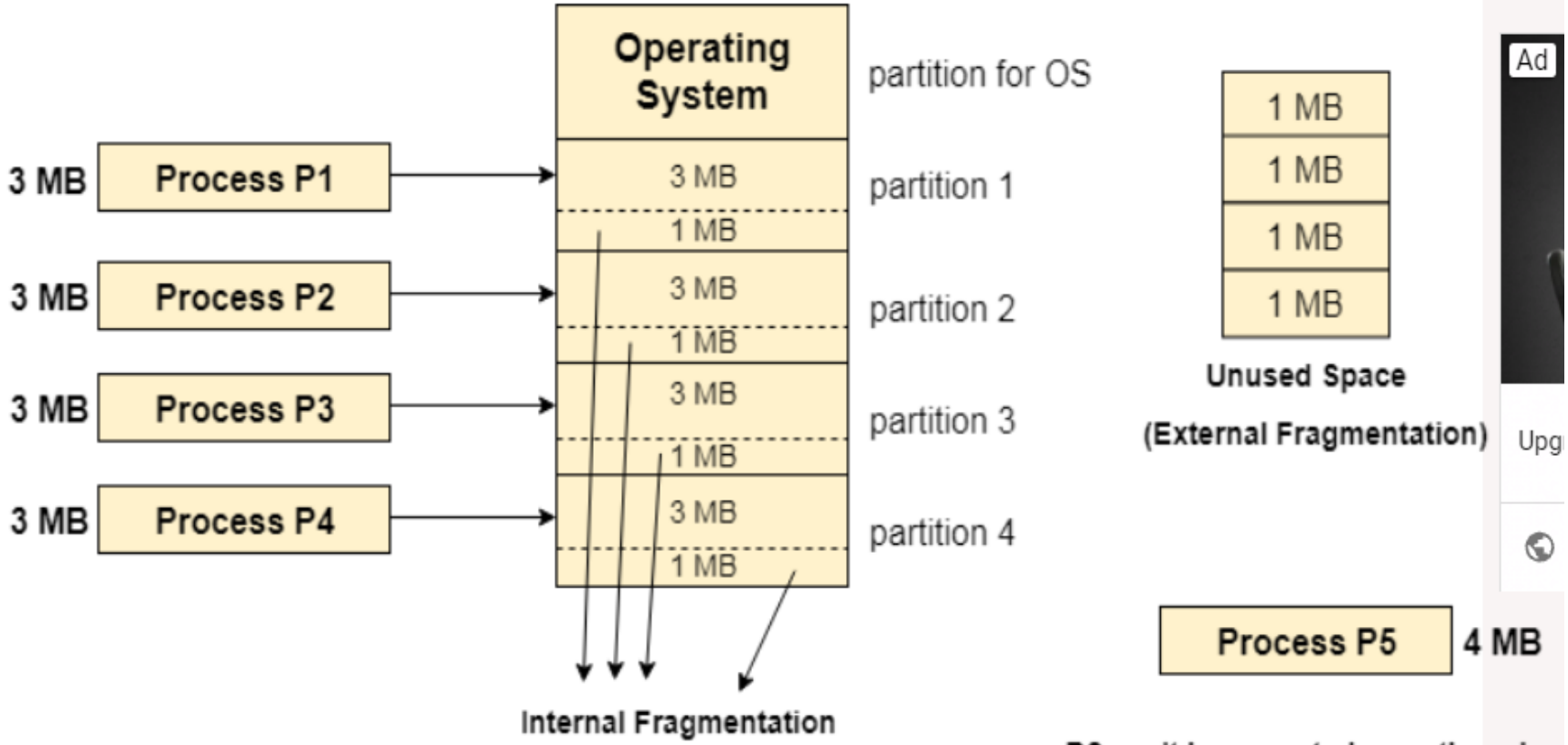
Contiguous Memory Allocation :

- The main memory should **oblige** both the operating system and the different client processes.
- Therefore, the **allocation of memory** becomes an important task in the operating system.
- The memory is **usually divided into two partitions**: one for the resident operating system and one for the user processes.
- We **normally need several user processes** to reside in memory simultaneously.
- Therefore, we need to consider how to allocate available memory to the processes









Fixed Partitioning

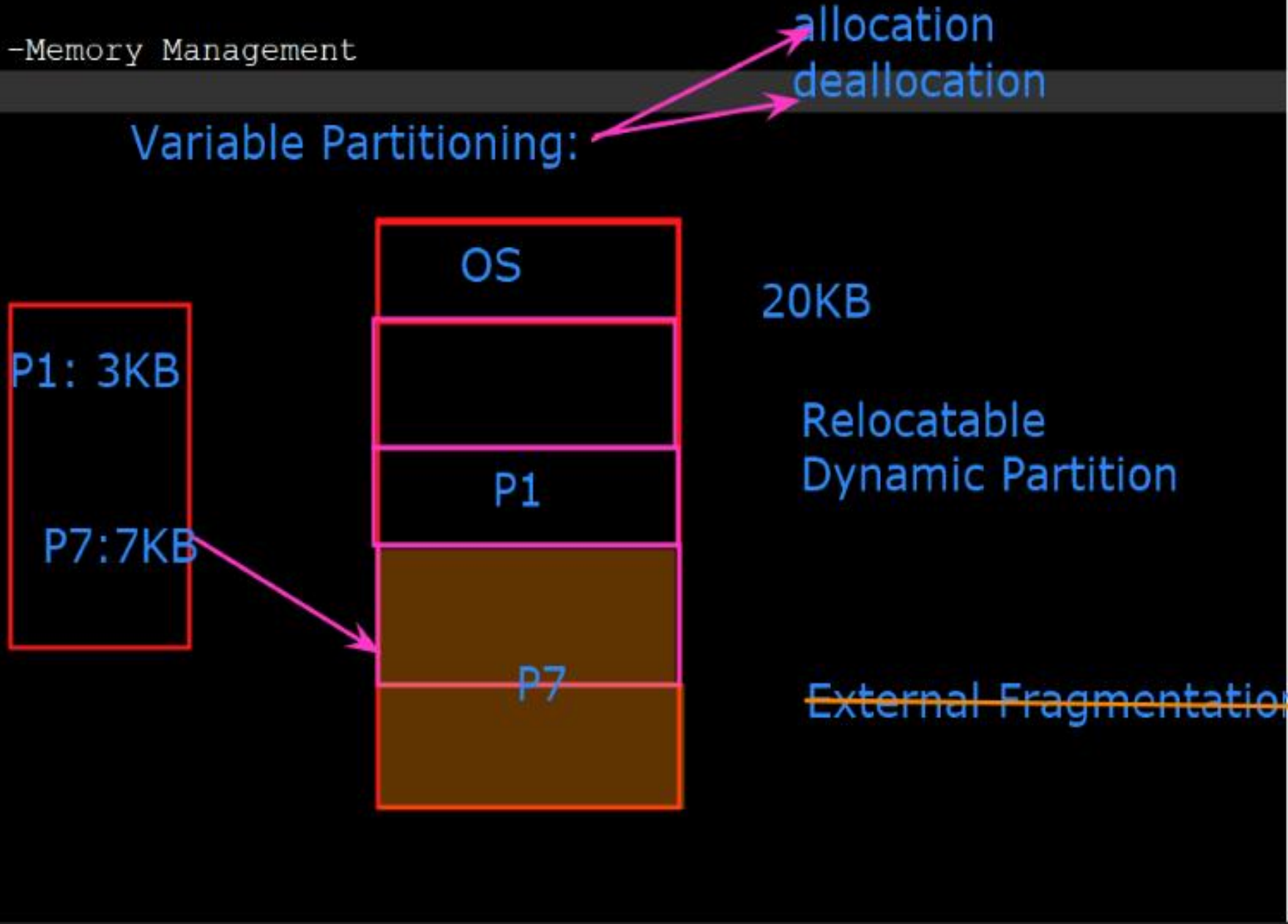
(Contiguous memory allocation)

Day 6: 19-03-2022

==========

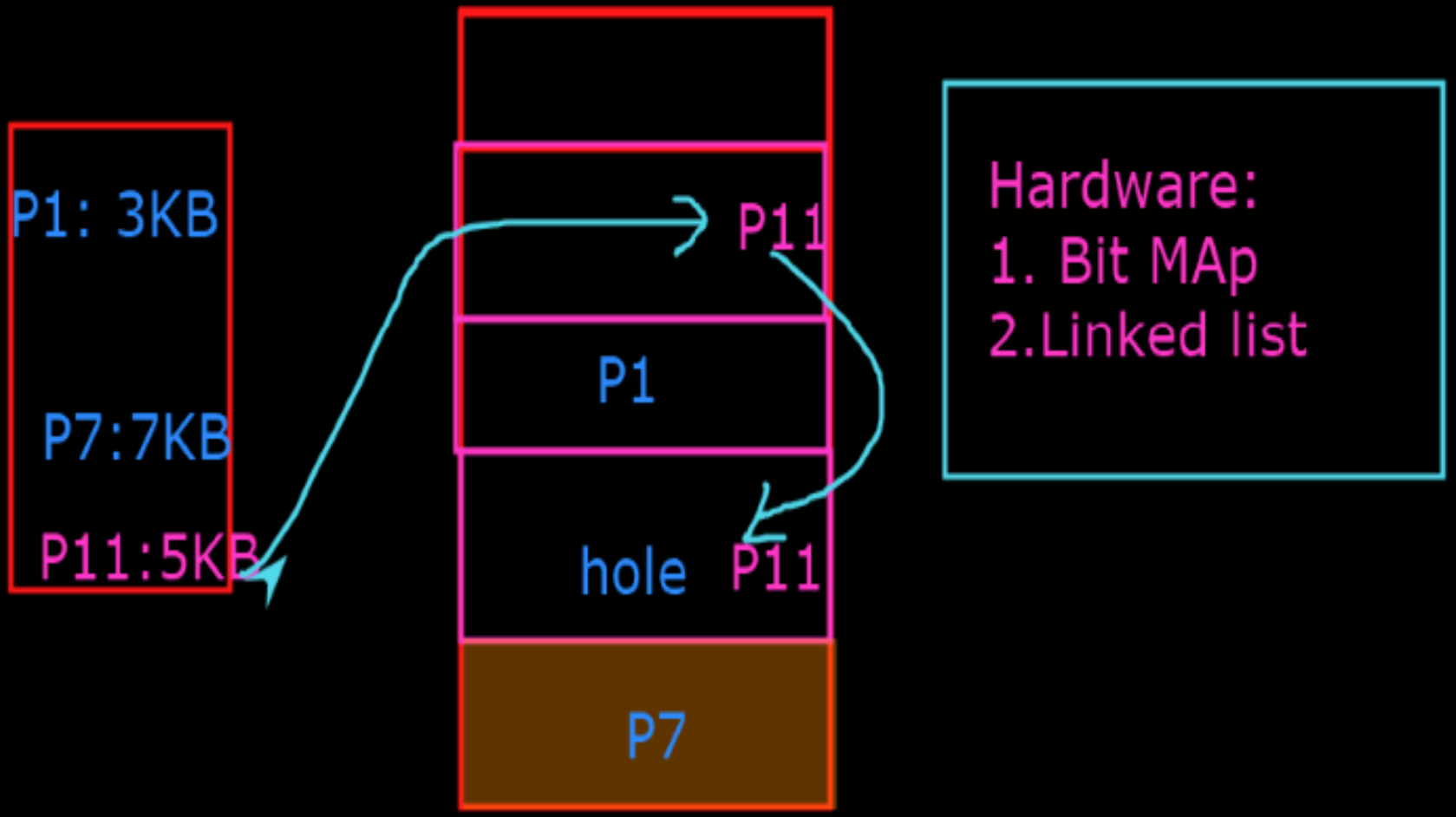
Topics:

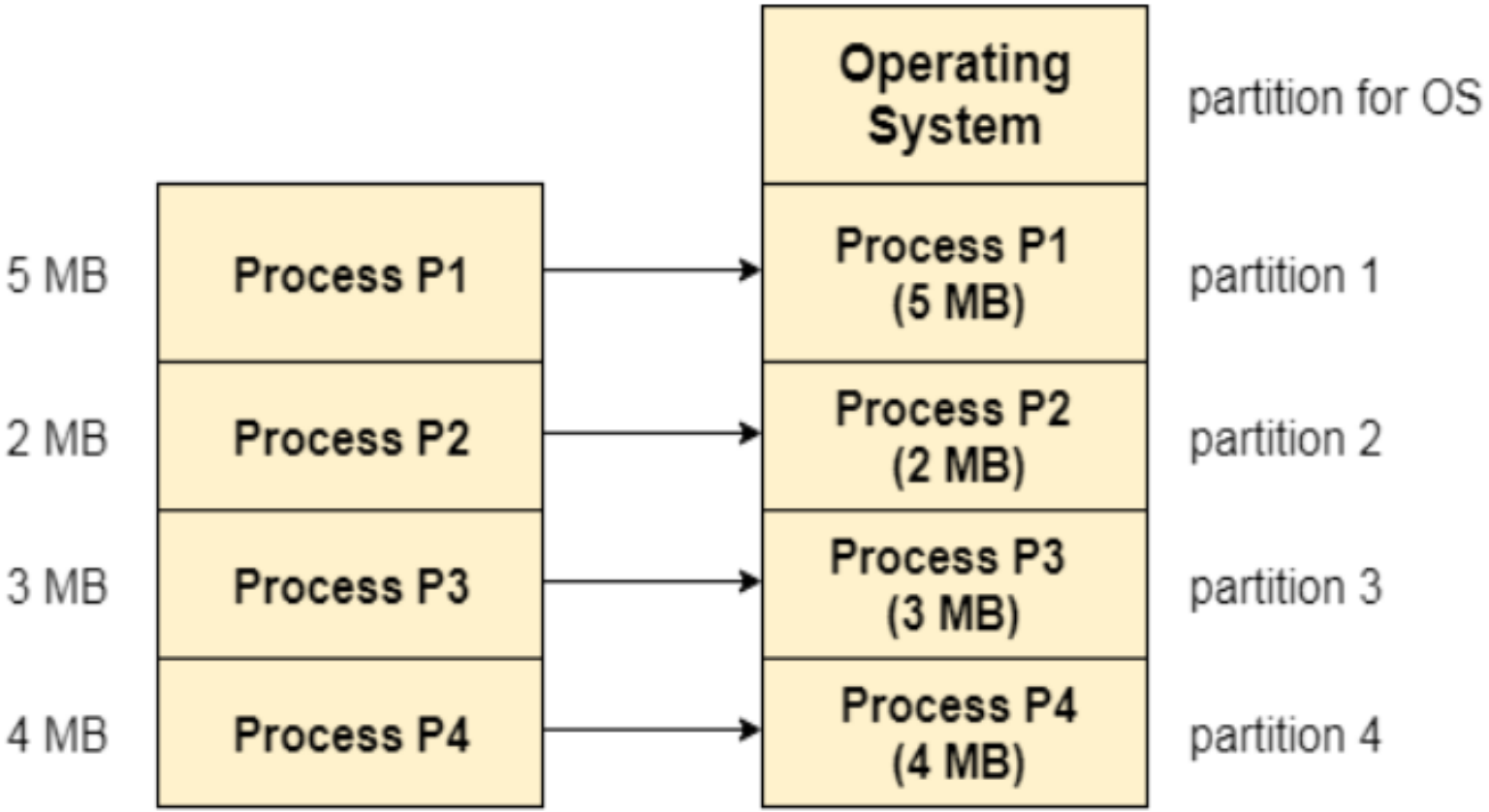
-Memory Management



allocation
deallocation

Variable Partitioning:



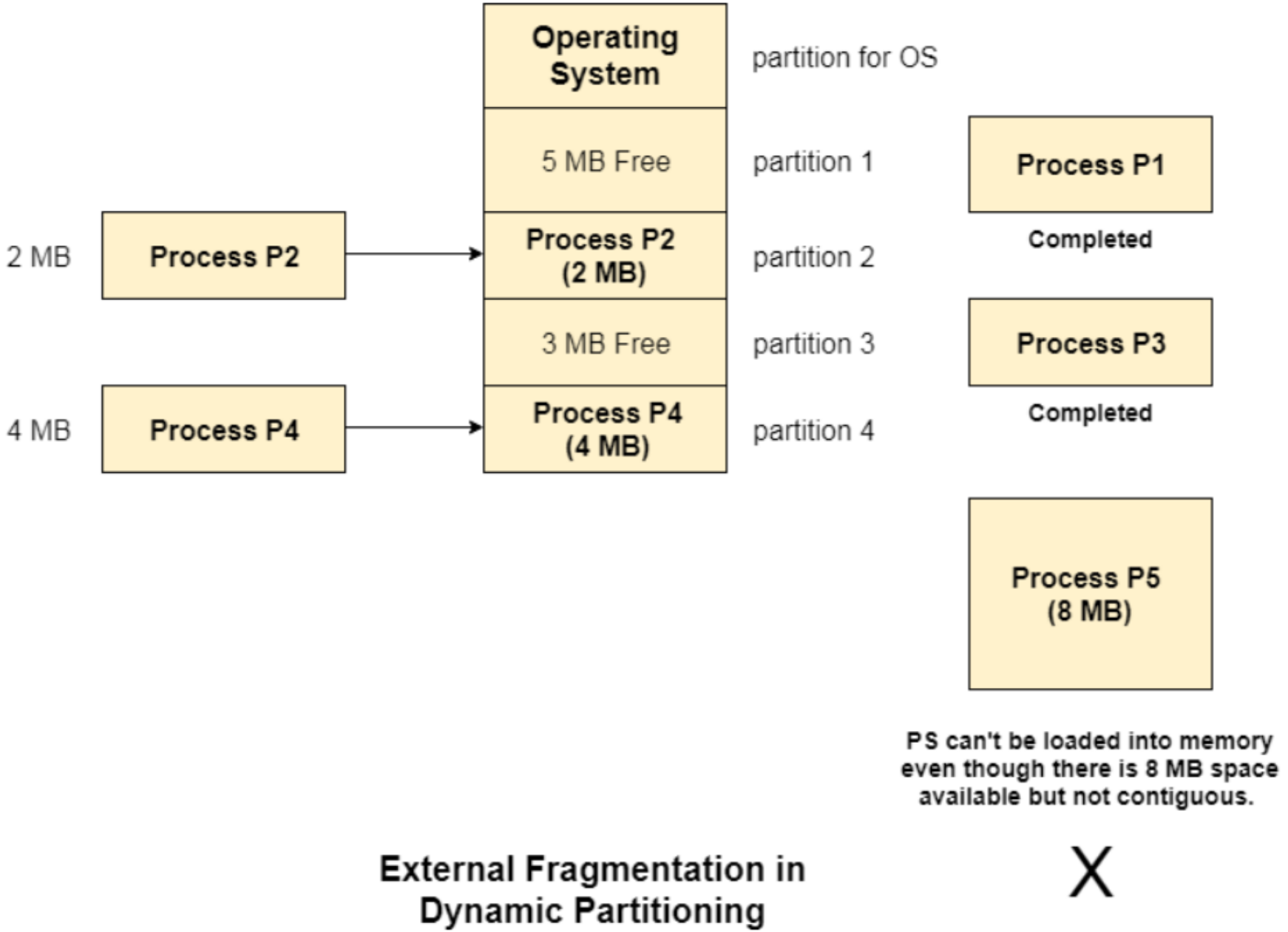


Dynamic Partitioning

(Process Size = Partition Size)

Fragmentation

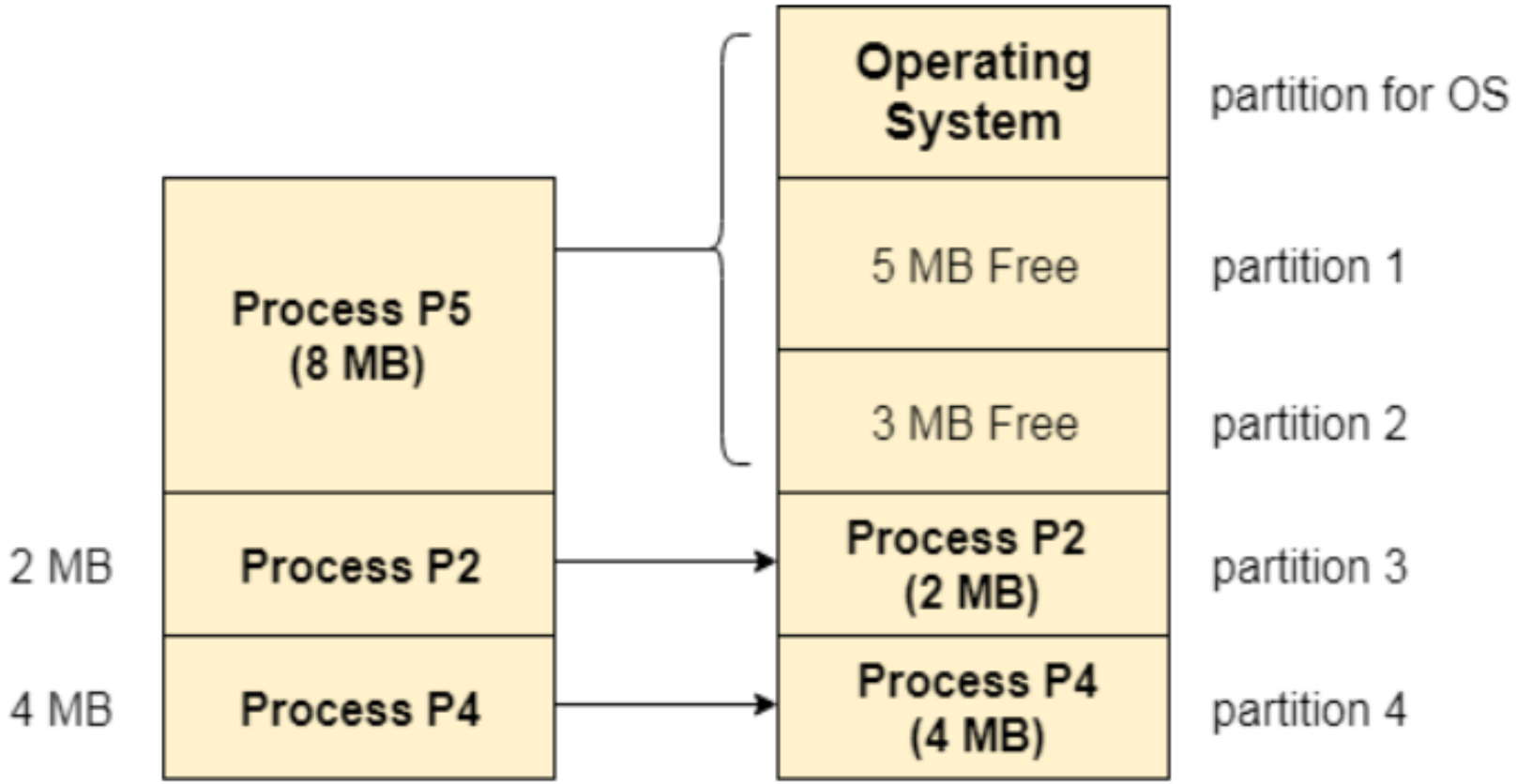
- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used
- Reduce external fragmentation by **compaction**
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is **possible *only* if relocation is dynamic**, and is done at execution time
  - I/O problem
    - Latch job in memory while it is involved in I/O
    - Do I/O only into OS buffers



External Fragmentation in Dynamic Partitioning

Compaction

- We got to know that the **dynamic partitioning suffers from external fragmentation**. However, this can cause some serious problems.
- To avoid compaction, we need to change the rule which says that the process can't be stored in the different places in the memory.
- We can also **use compaction to minimize the probability of external fragmentation**. In compaction, all the free partitions are made contiguous and all the loaded partitions are brought together.
- By applying this technique, we can store the bigger processes in the memory.
- The free partitions are merged which can now be allocated according to the needs of new processes. This technique is also called defragmentation.



Now PS can be loaded into memory because the free space is now made contiguous by compaction

Compaction

Bit Map for Dynamic Partitioning

- The Main concern for dynamic partitioning is keeping track of all the free and allocated partitions. However, the Operating system uses following data structures for this task.
  - Bit Map
  - Linked List

Page Fault Algorithms: FIFO,OPR,LRU

Sequence: 7,0,1,2,0,3,0,4,2,3,0,3,1,2,0 frame:3

Calculate : Hit, Miss (page fault)

FIFO

f3			1	1	1	1	0	0	0	3	3	3	2	2
f2		0	0	0	0	3	3	3	2	2	2	1	1	1
f1	7	7	7	2	2	2	2	4	4	4	0	0	0	0
	★	★	★	★	✓	★	★	★	★	★	★	★	★	✓

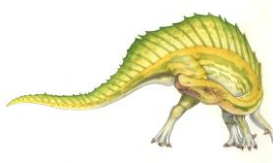
Hit=3
Miss=12

Hit Ratio = 3/15*100=

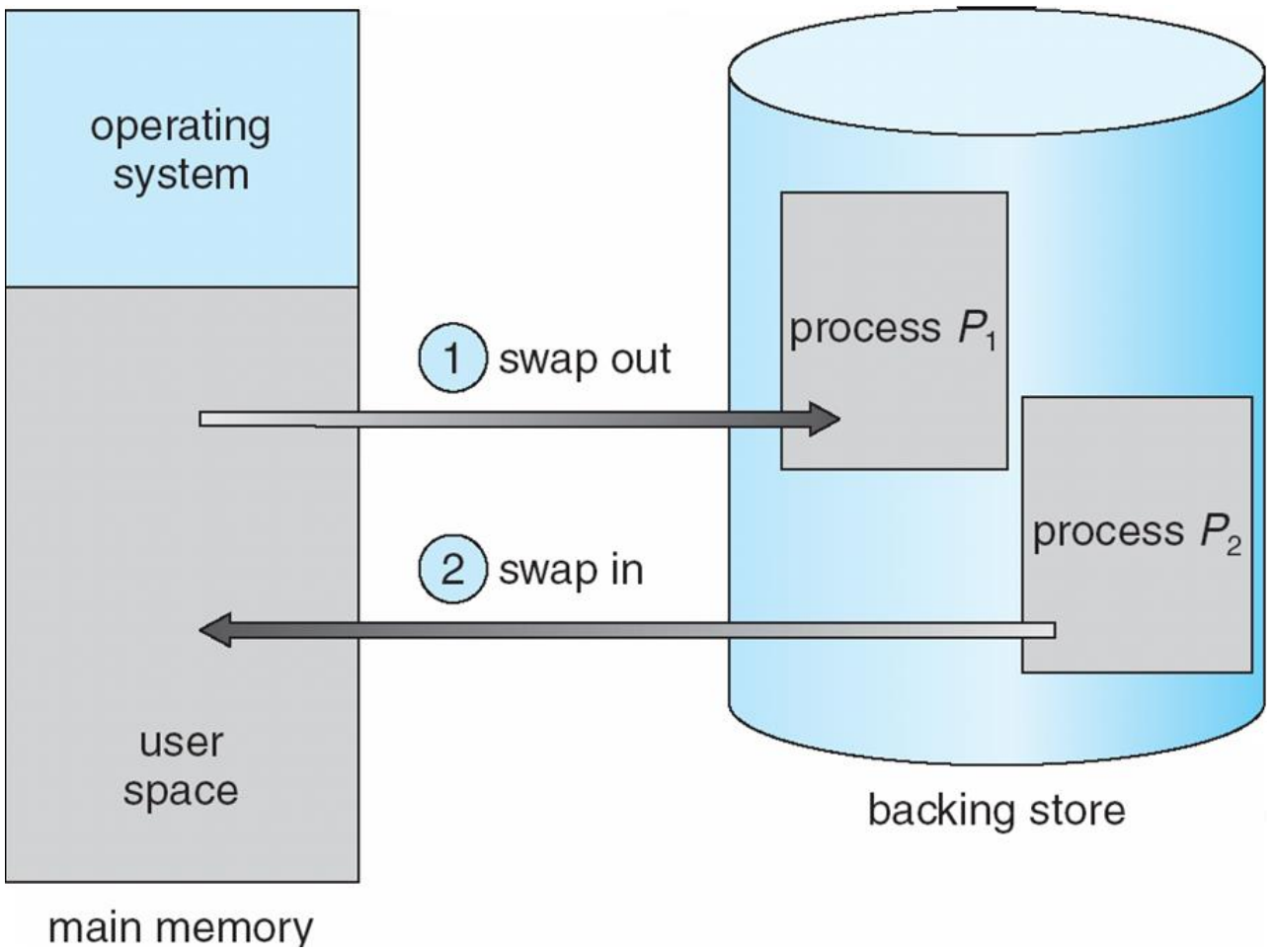
Miss Ratio = 13/15*100=

Swapping

- A process **can be swapped temporarily** out of memory to a backing store, and then brought back into memory for continued execution
- **Backing store** – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
- **Roll out, roll in** – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed
- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped
- Modified versions of swapping are found on many systems (i.e., UNIX, Linux, and Windows)
- System maintains a **ready queue** of ready-to-run processes which have memory images on disk

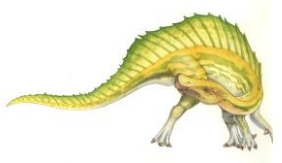


Schematic View of Swapping



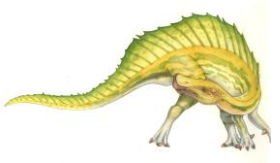
Swap In and Swap Out in OS

- The procedure by which any process gets removed from the hard disk and placed in the main memory or RAM commonly known as Swap In.
- On the other hand, Swap Out is the method of removing a process from the main memory or RAM and then adding it to the Hard Disk.
- **Advantages of Swapping**
- The swapping technique **mainly helps the CPU to manage multiple processes within a single main memory**.
- This technique **helps to create and use virtual memory**.
- With the help of this technique**, the CPU can perform several tasks** simultaneously. Thus, processes need not wait too long before their execution.
- This technique is **economical.**
- This technique can be easily applied **to priority-based scheduling** in order to improve its performance.

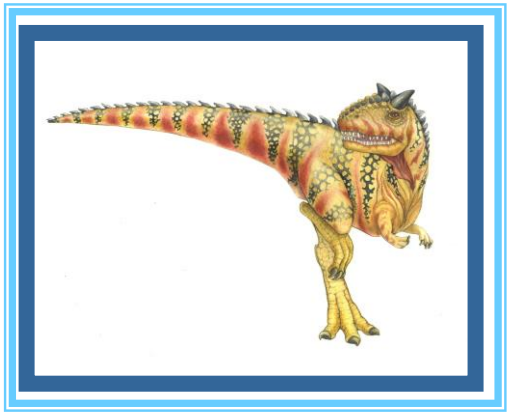


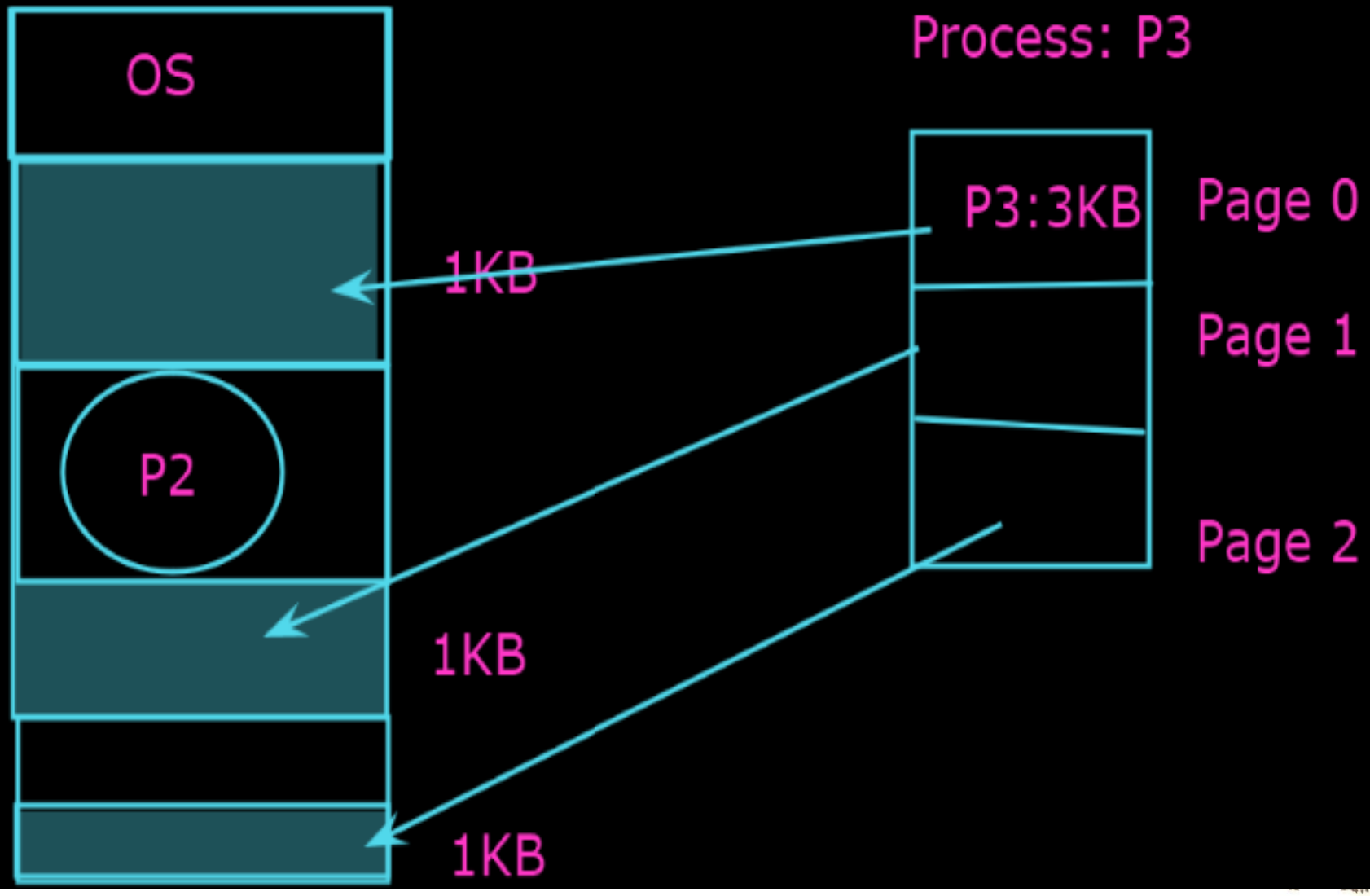
Disadvantages of Swapping

- There may occur inefficiency in the case if a resource or a variable is commonly used by those processes that are participating in the swapping process.
- If the algorithm used for swapping is not good then the overall method can increase the number of page faults and thus decline the overall performance of processing.
- If the computer system loses power at the time of high swapping activity then the user might lose all the information related to the program.



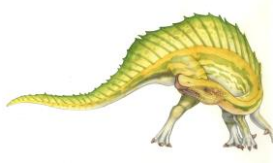
Paging and Segmentation



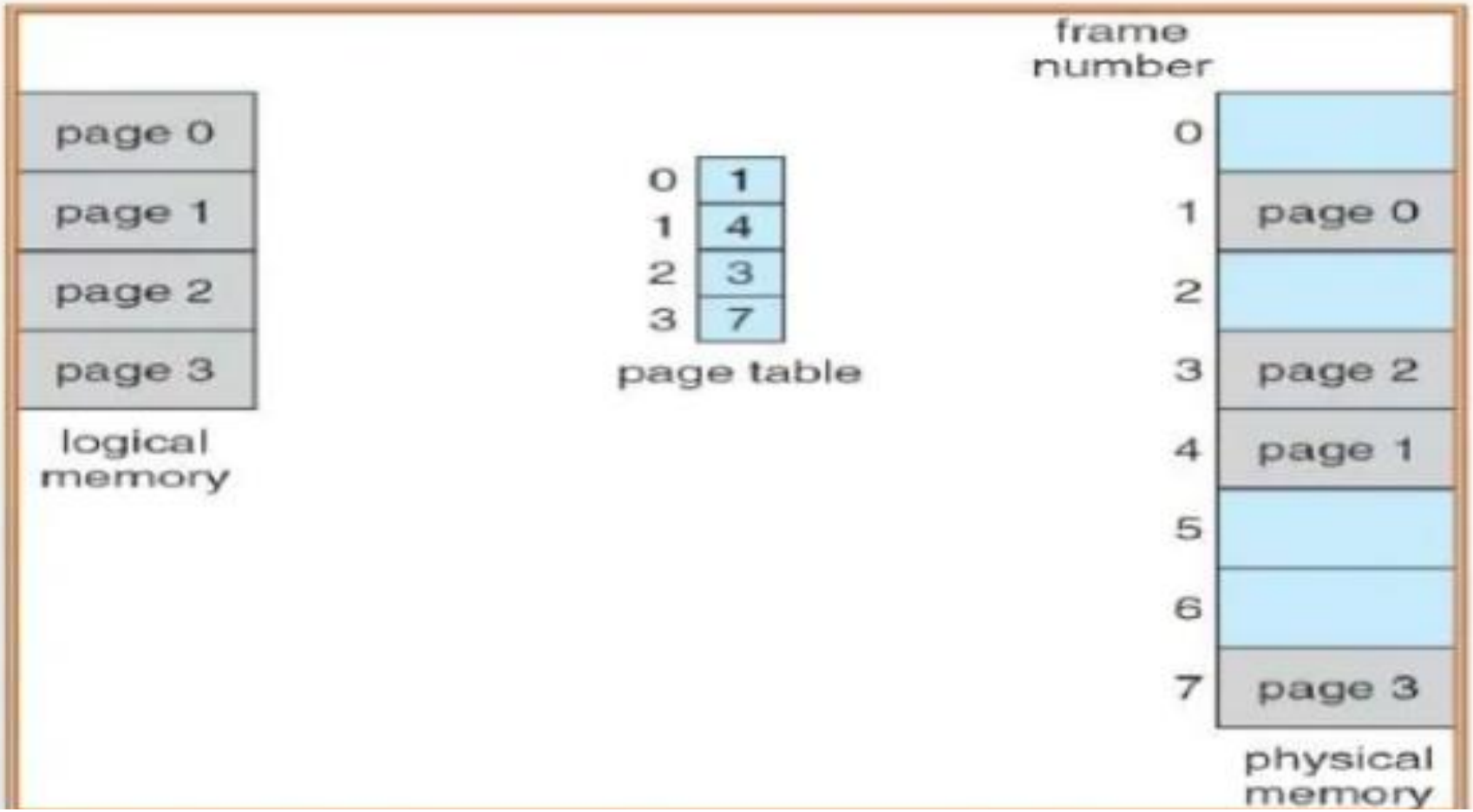


Need for Paging

- Disadvantage of Dynamic Partitioning
  - 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.
  - We **need to find out a mechanism which can load the processes in the partitions** in a more optimal way. Let us discuss a dynamic and flexible mechanism **called paging.**
- **Need for Paging**
- Lets consider a process P1 of size 2 MB and the main memory which is divided into three partitions. Out of the three partitions, two partitions are holes of size 1 MB each.
- P1 needs 2 MB space in the main memory to be loaded. **We have two holes of 1 MB each but they are not contiguous.**



Paging Example



Paging

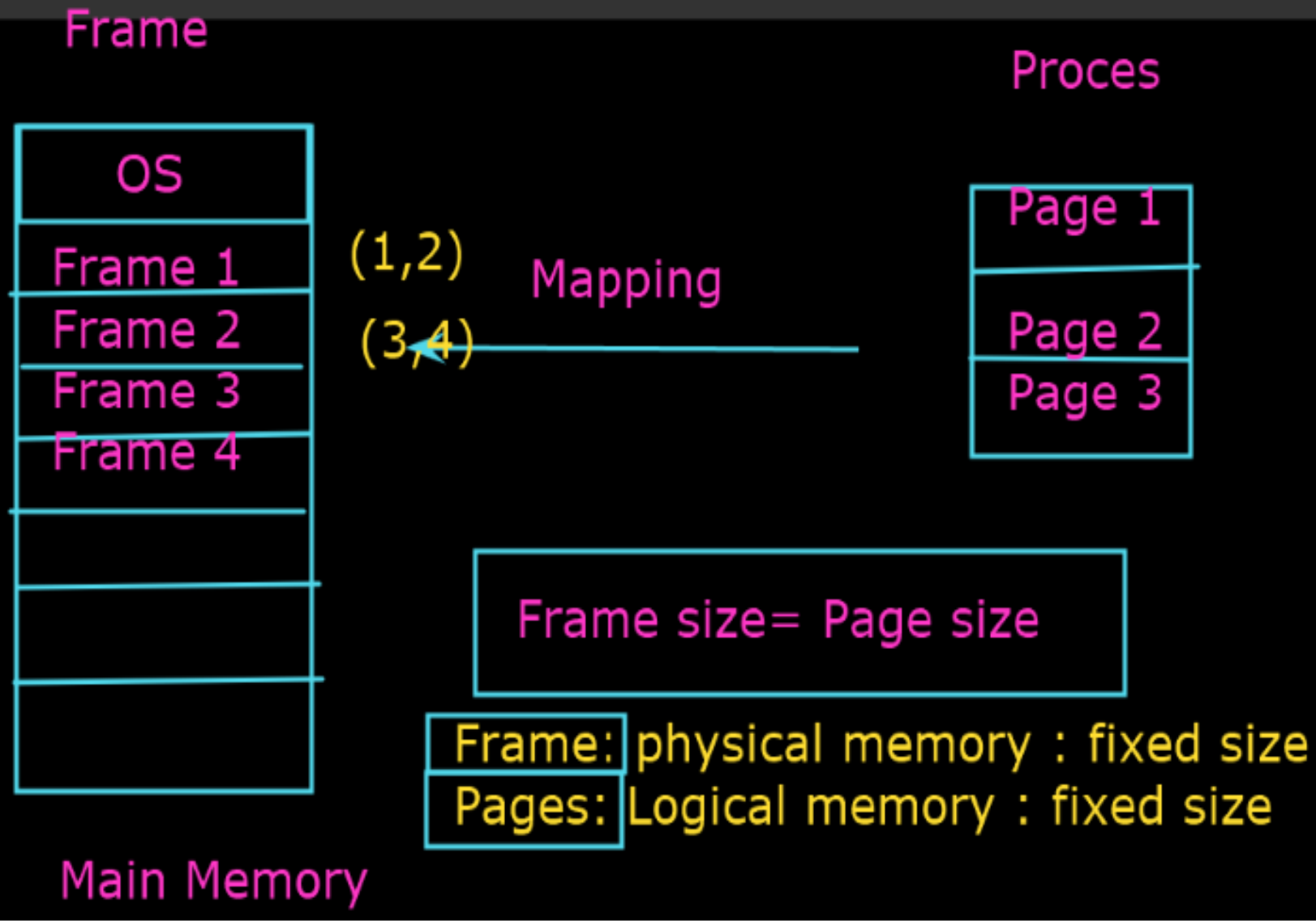
- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available
- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 8,192 bytes)
- Divide logical memory into blocks of same size called **pages**
- Keep track of all free frames
- To run a program of size ***n*** pages, need to find *n* free frames and load program
- Set up a page table to translate logical to physical addresses
- Internal fragmentation

==========

Topics:

-Memory Management

Paging

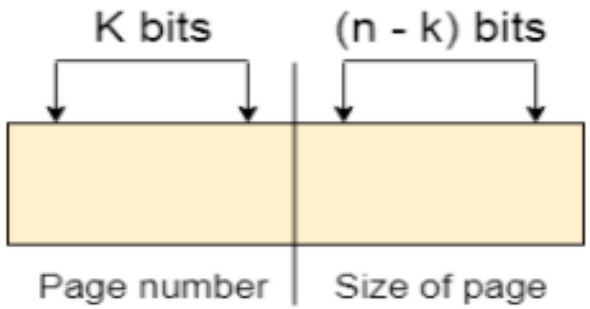


Memory Management Unit

- The purpose of Memory Management Unit (MMU) is **to convert the logical address into the physical address.**
- The logical address is the address generated by the CPU for every page while the physical address is the actual address of the frame where each page will be stored.
- When a page is to be accessed by the CPU by using the logical address, the operating system needs to obtain the physical address to access that page physically.
- The logical address has two parts.
  - Page Number
  - Offset
- Memory management unit of OS needs to convert the page number to the frame number.

1 Bit	2 Bits		3 Bits		
0	0	0	0	0	0
1	0	1	0	0	1
	1	0	0	1	0
	1	1	0	1	1
			1	0	0
			1	0	1
			1	1	0
			1	1	1

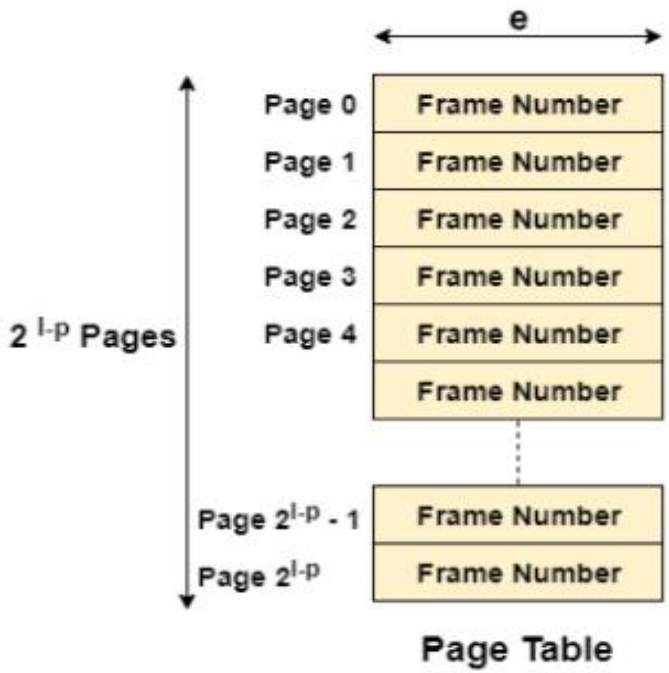
these n bits can be divided into two parts, that are, **K** bits and **(n-k)** bits.

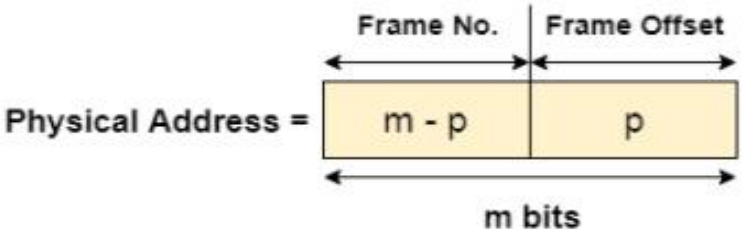


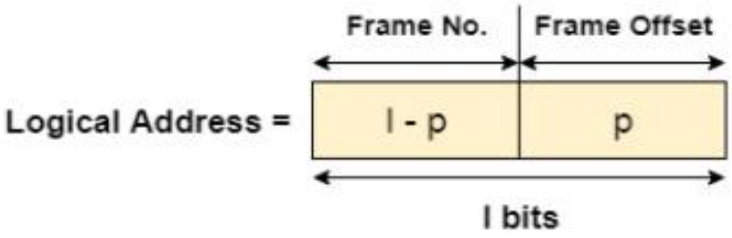
$$2^n = 2^n \times 2^{n-k}$$

Page Table

- Page Table is a data structure used by the virtual memory system to store the mapping between logical addresses and physical addresses.



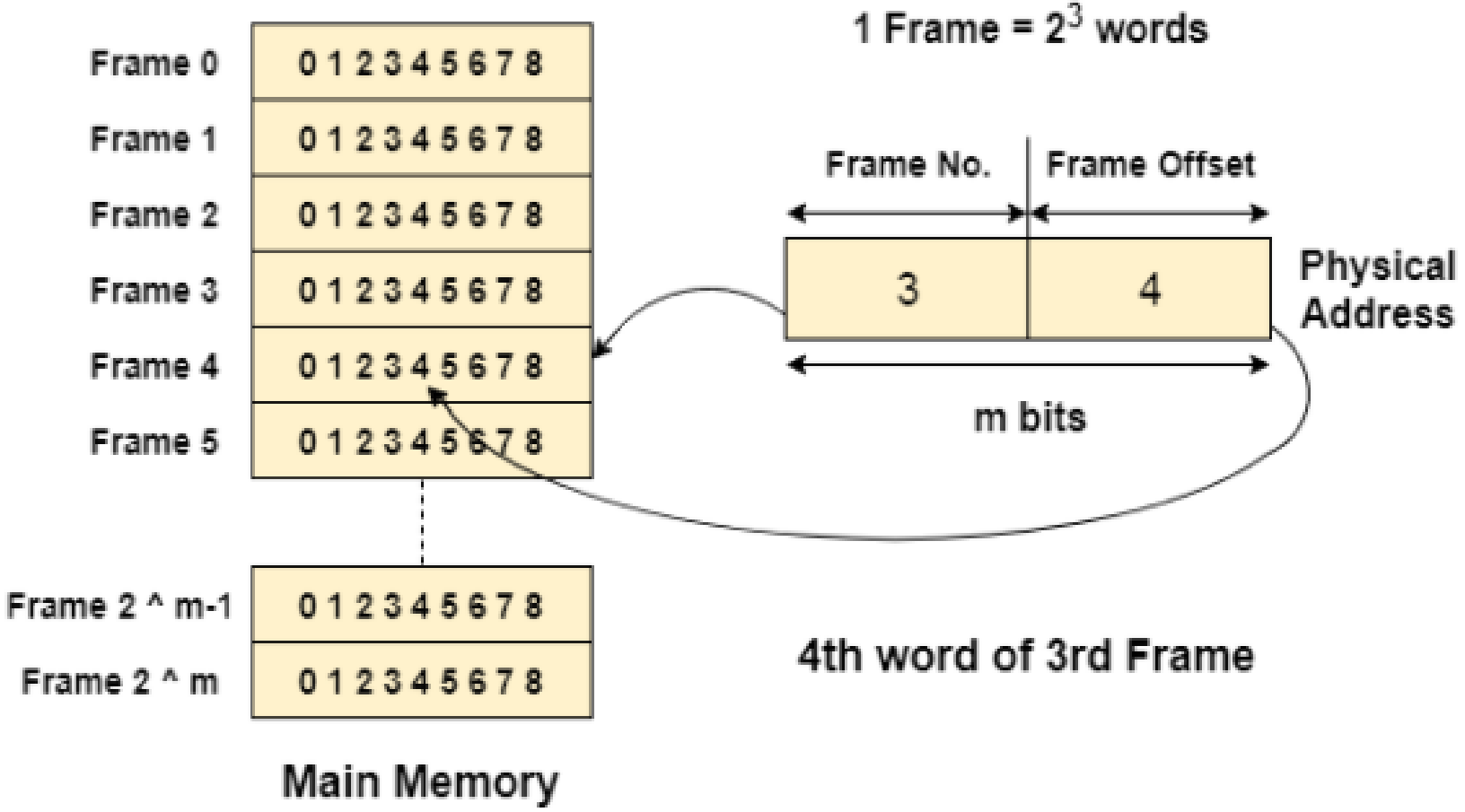


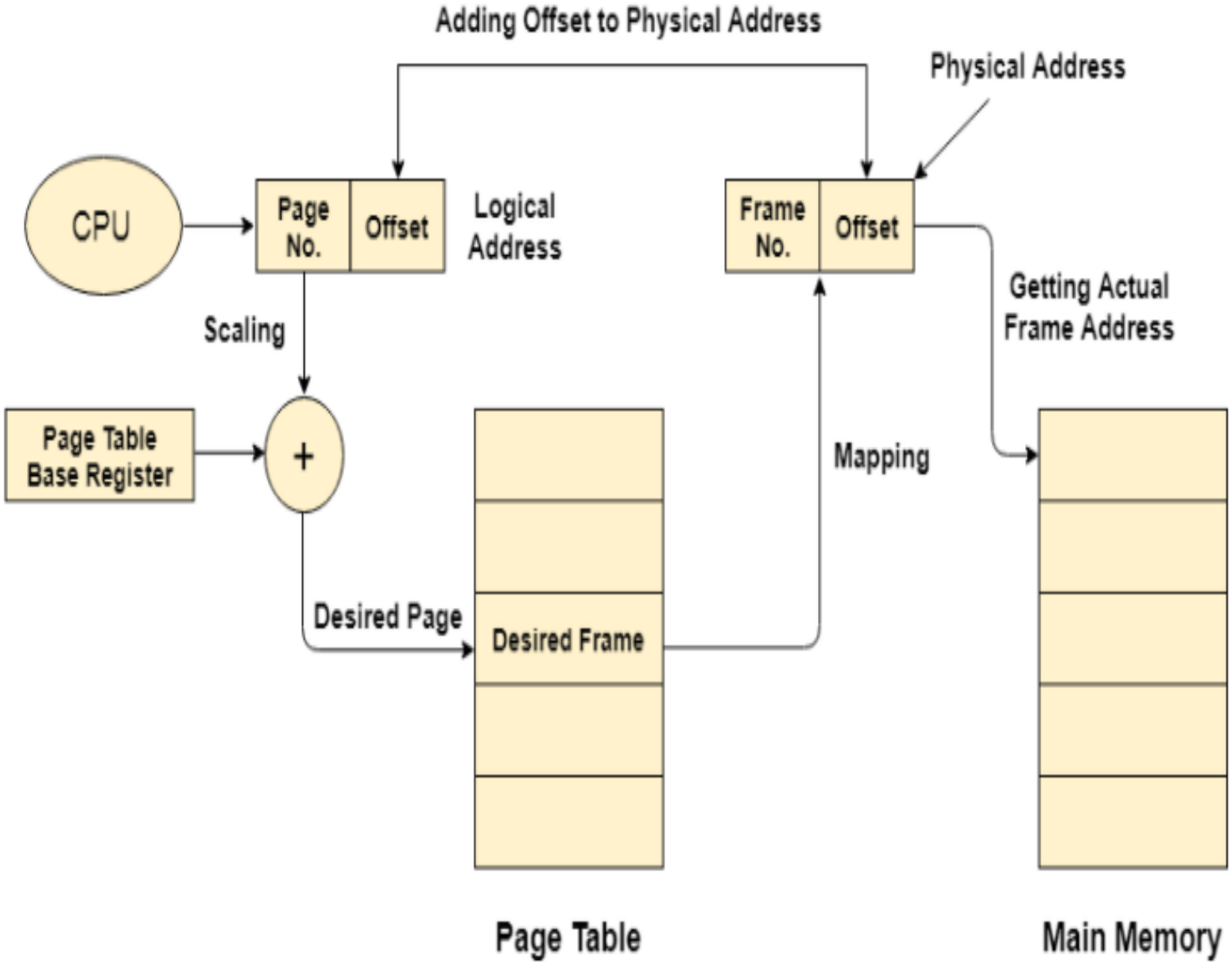


No. of entries in Page Table = No. of the pages in the process

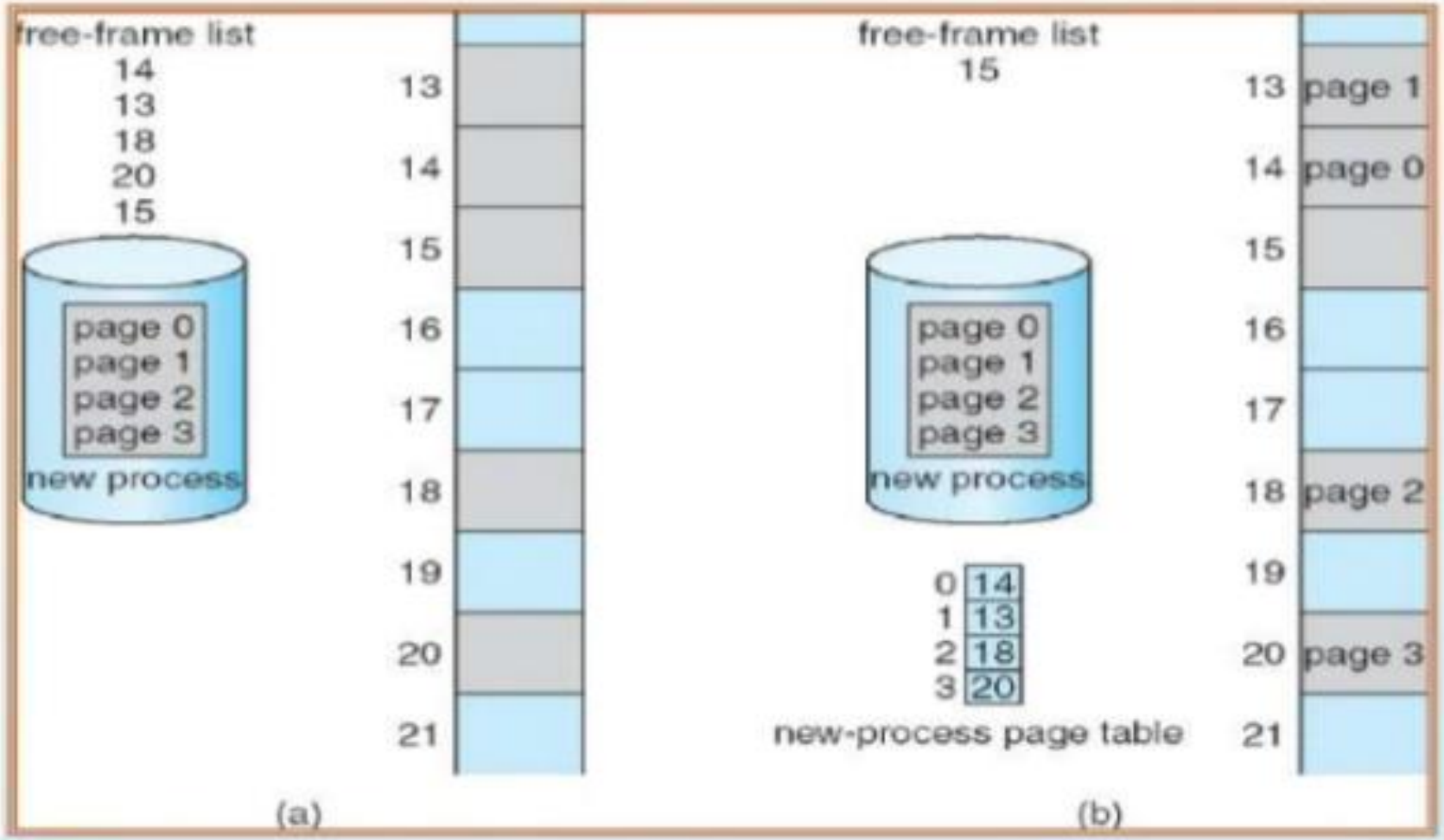
Page Table Size = 2^{l-p} X e bytes

e = m-p (Frame Size) bits



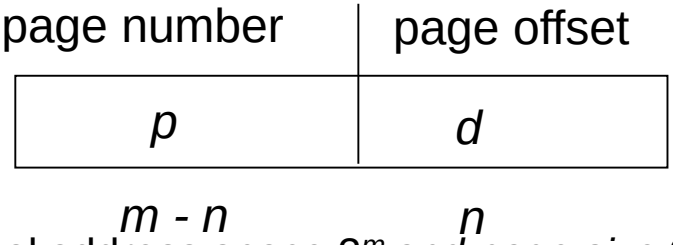


Paging Example



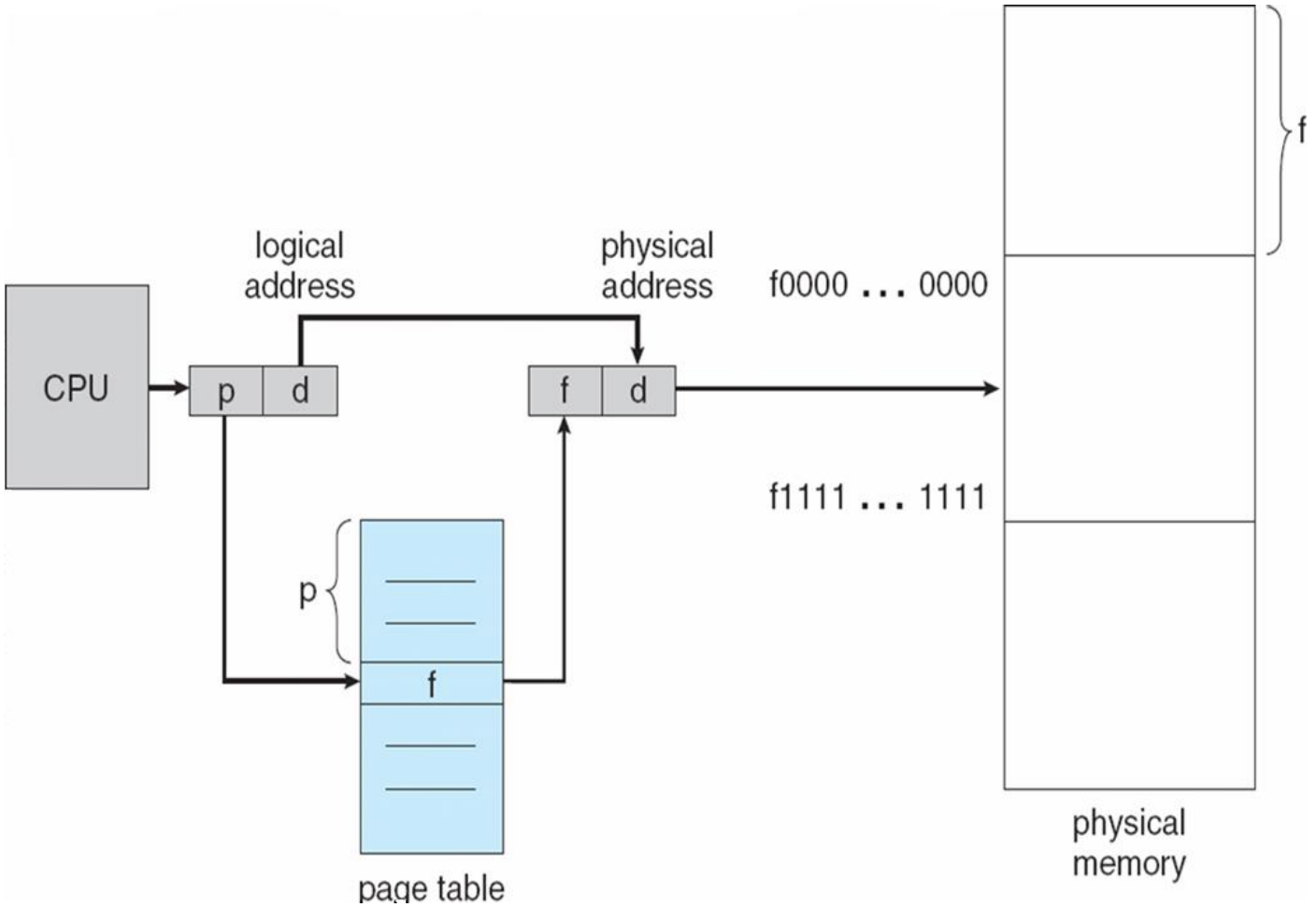
Address Translation Scheme

- 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

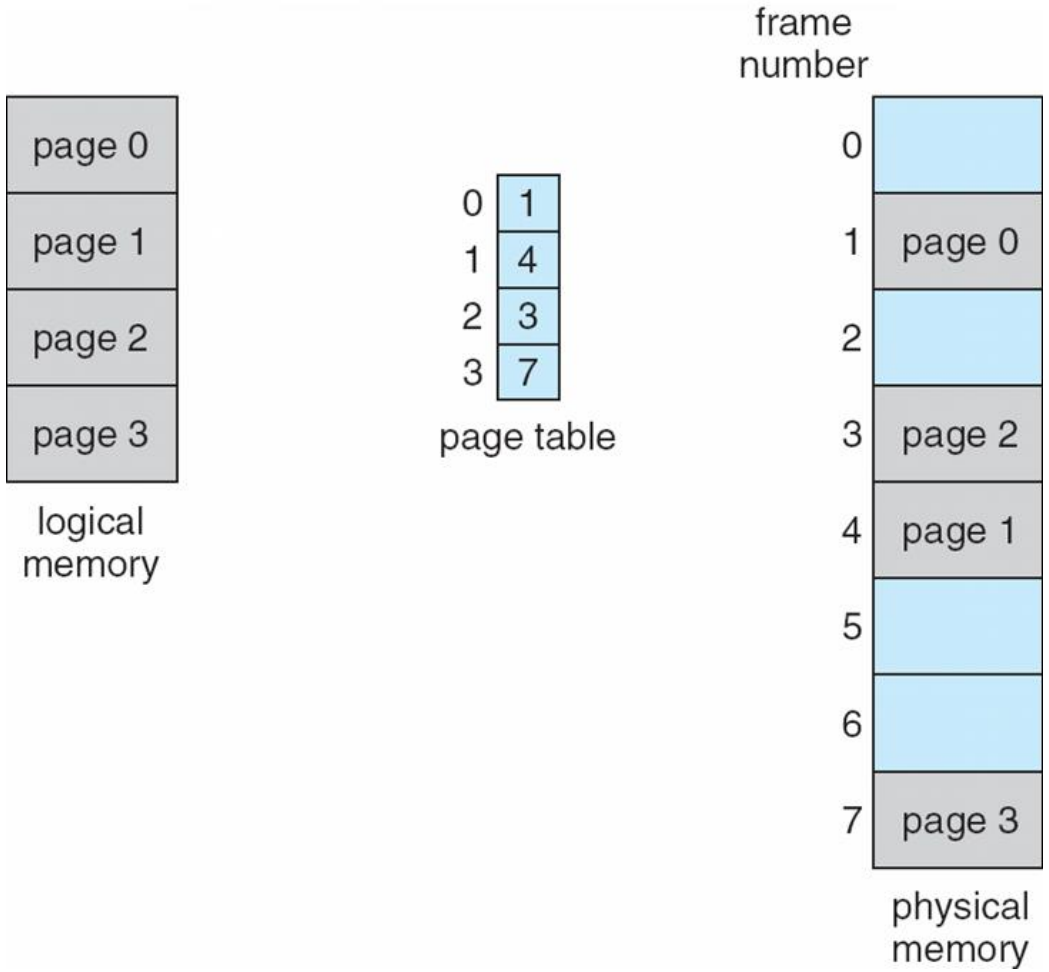


- For given logical address space 2^m *and page size* 2^n

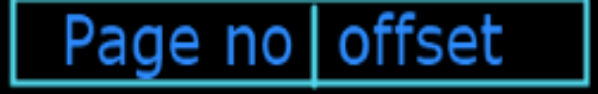
Paging Hardware

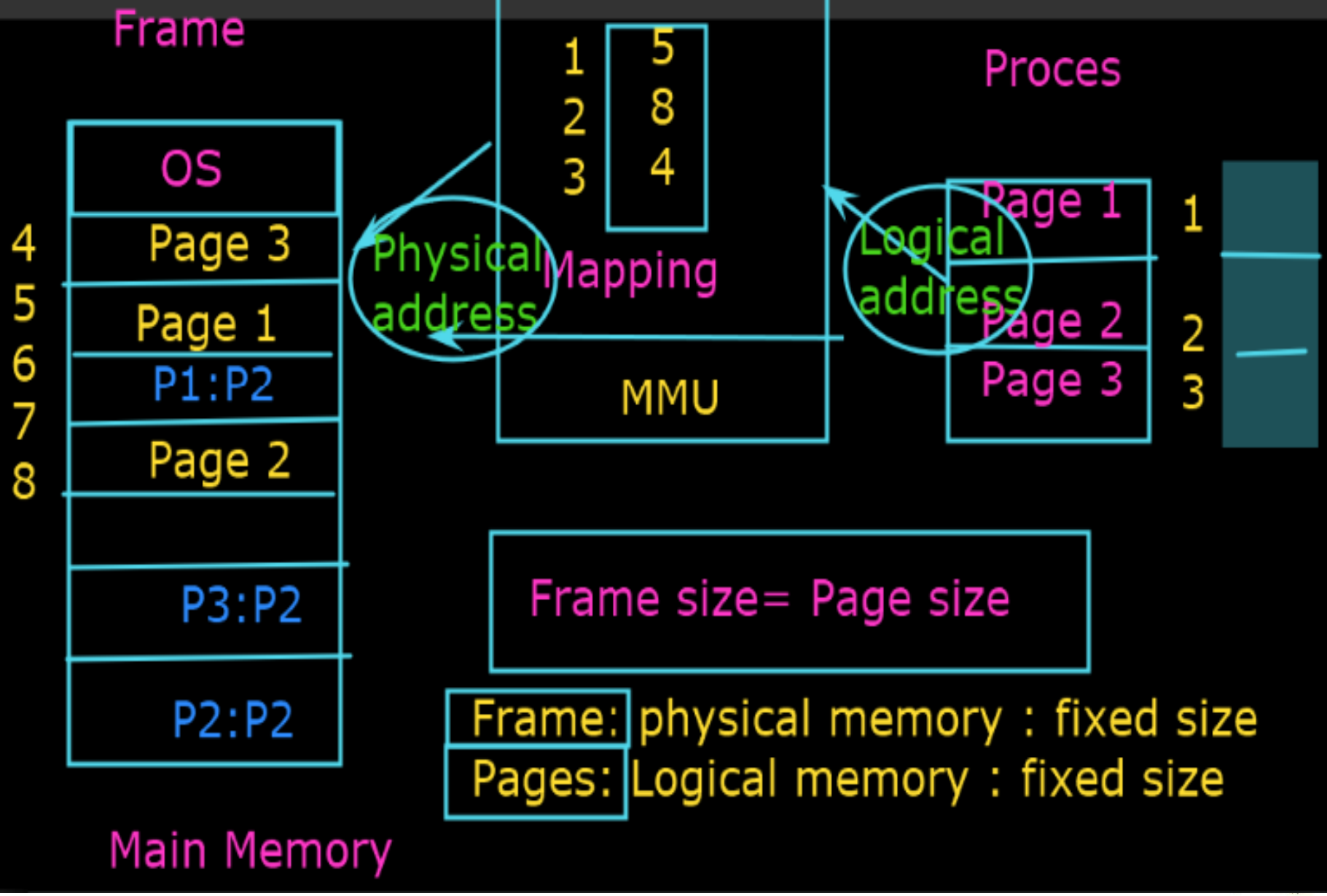


Paging Model of Logical and Physical Memory

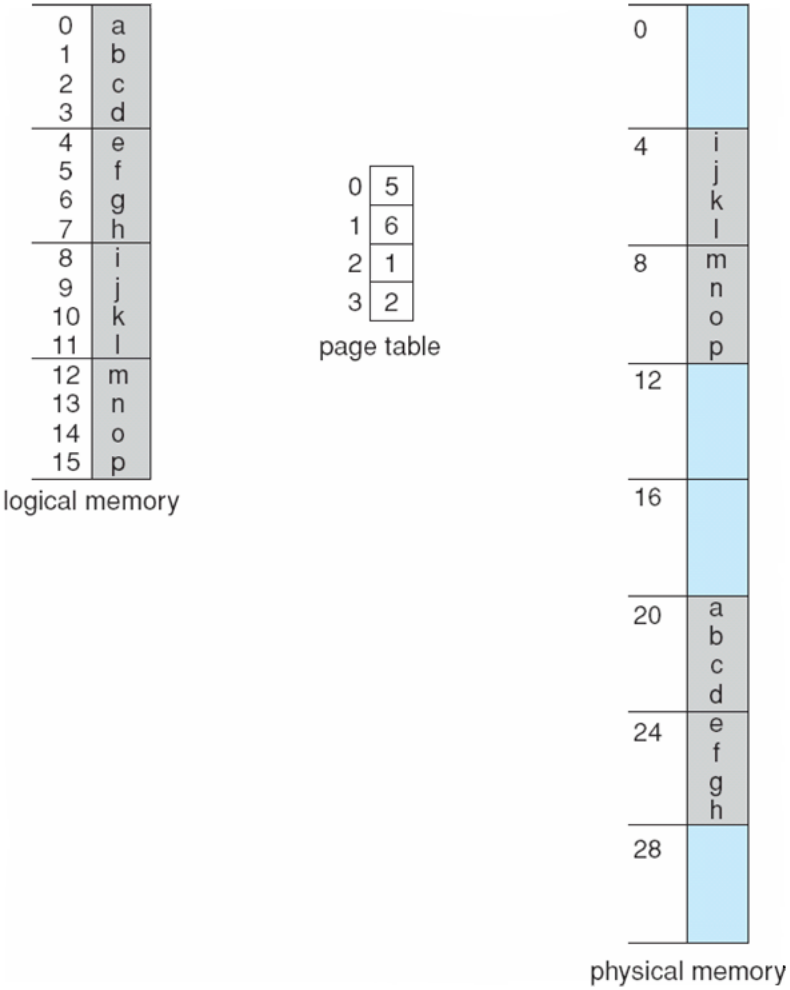


Paging



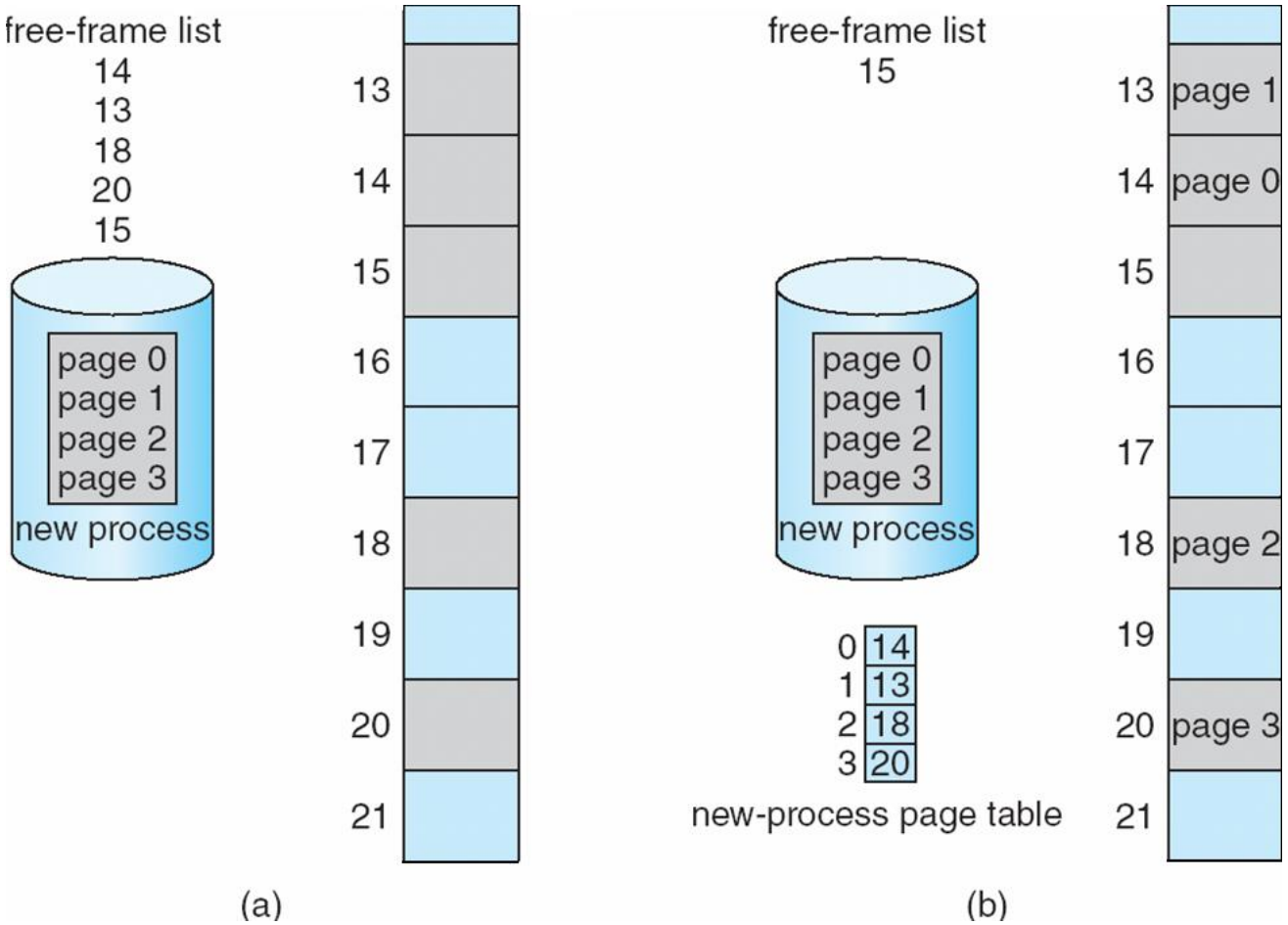


Paging Example



32-byte memory and 4-byte pages

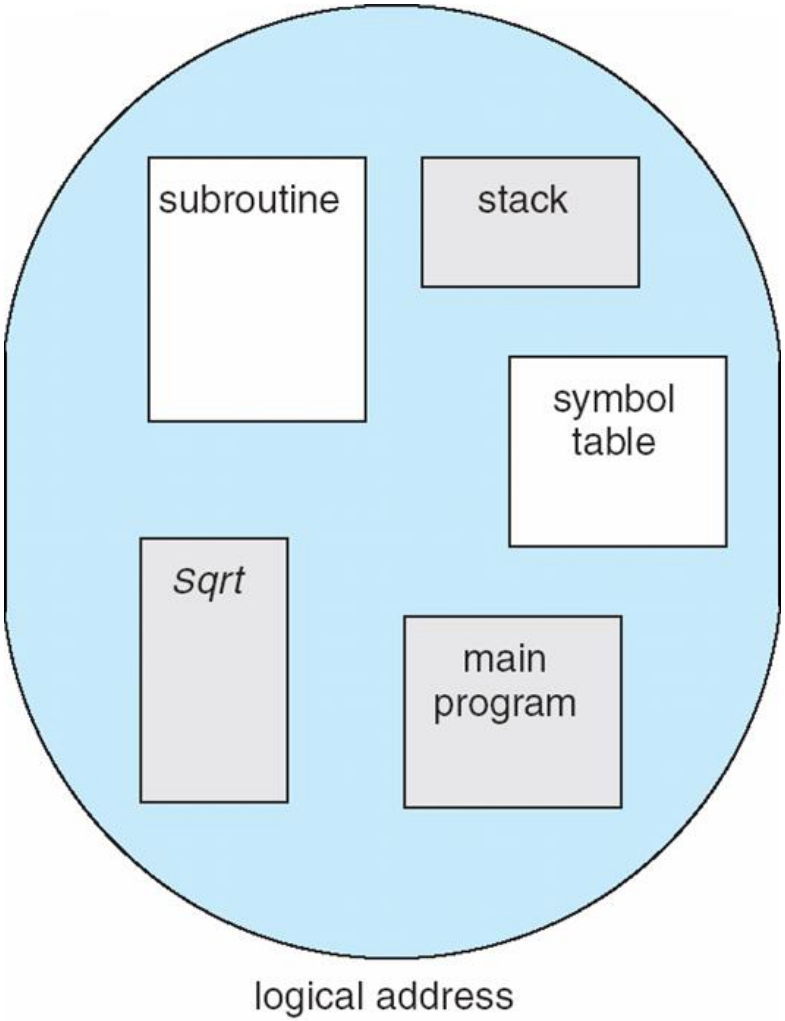
Free Frames



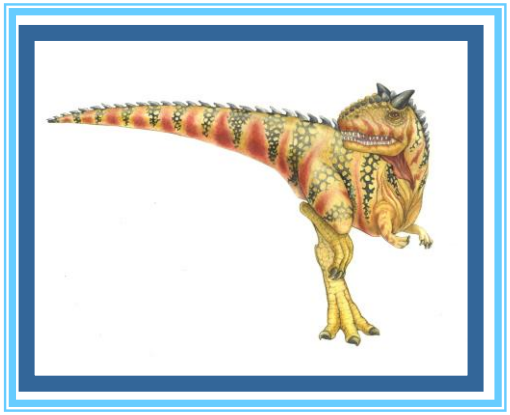
Before allocation

After allocation

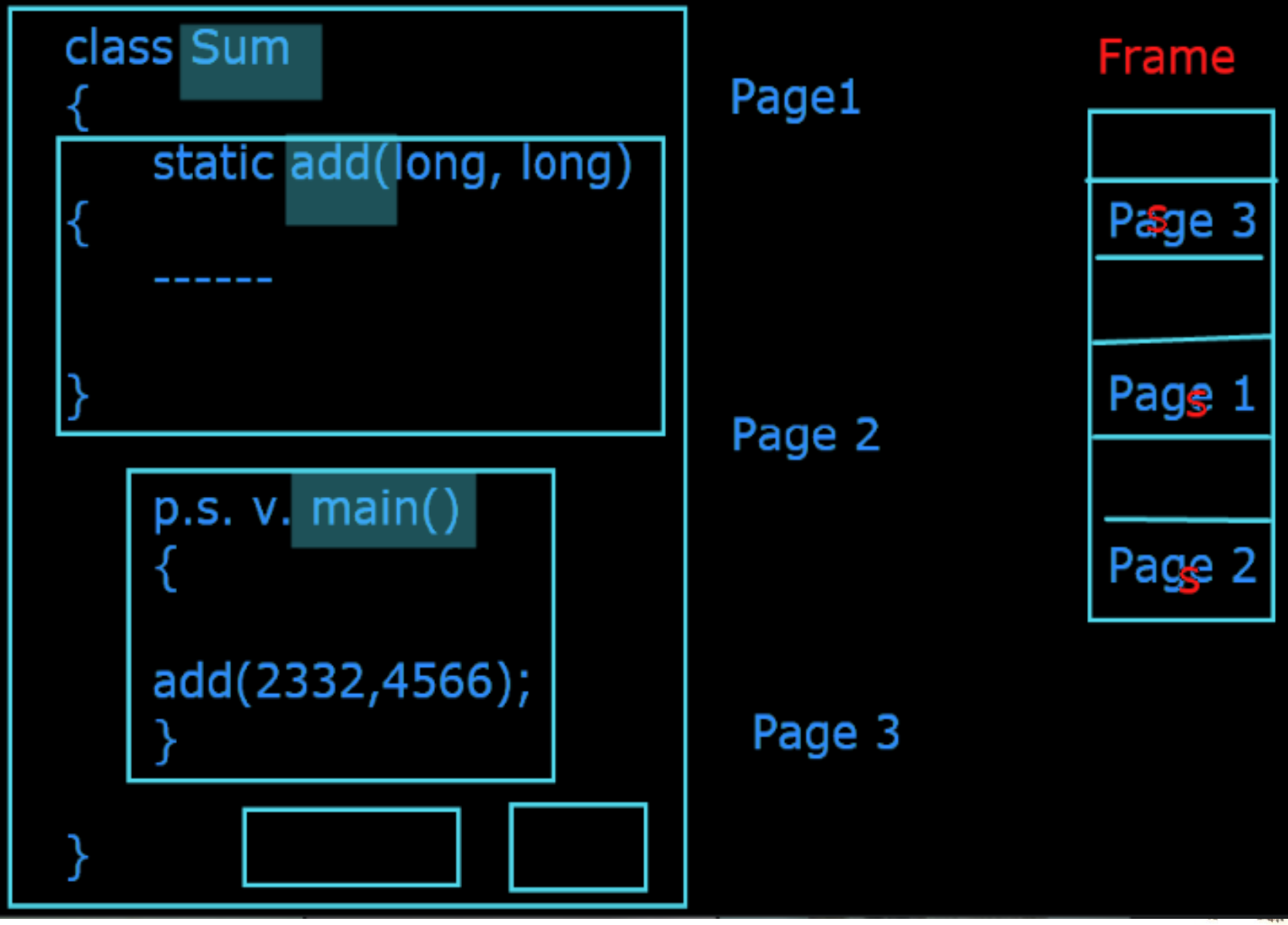
User's View of a Program



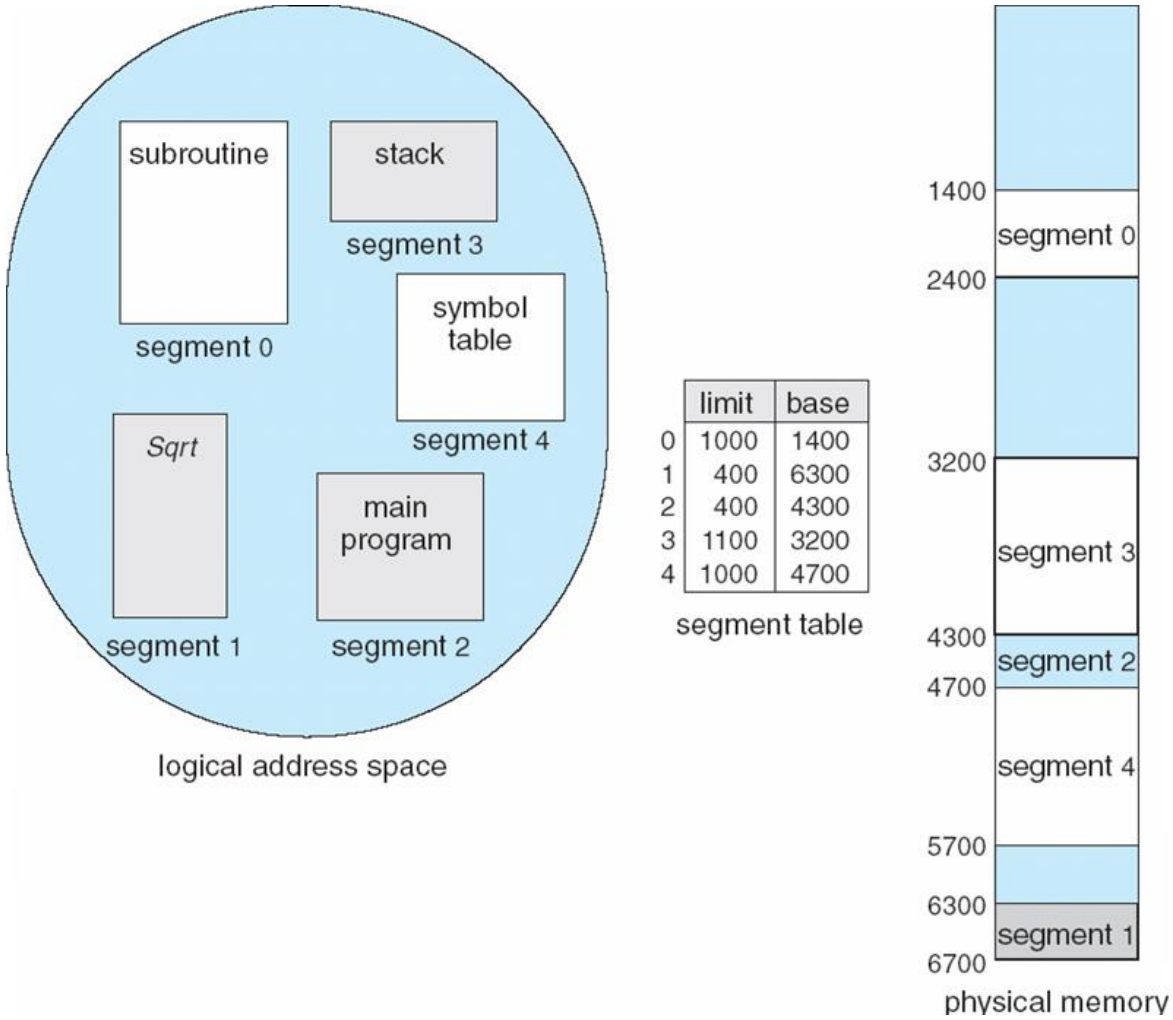
Segmentation

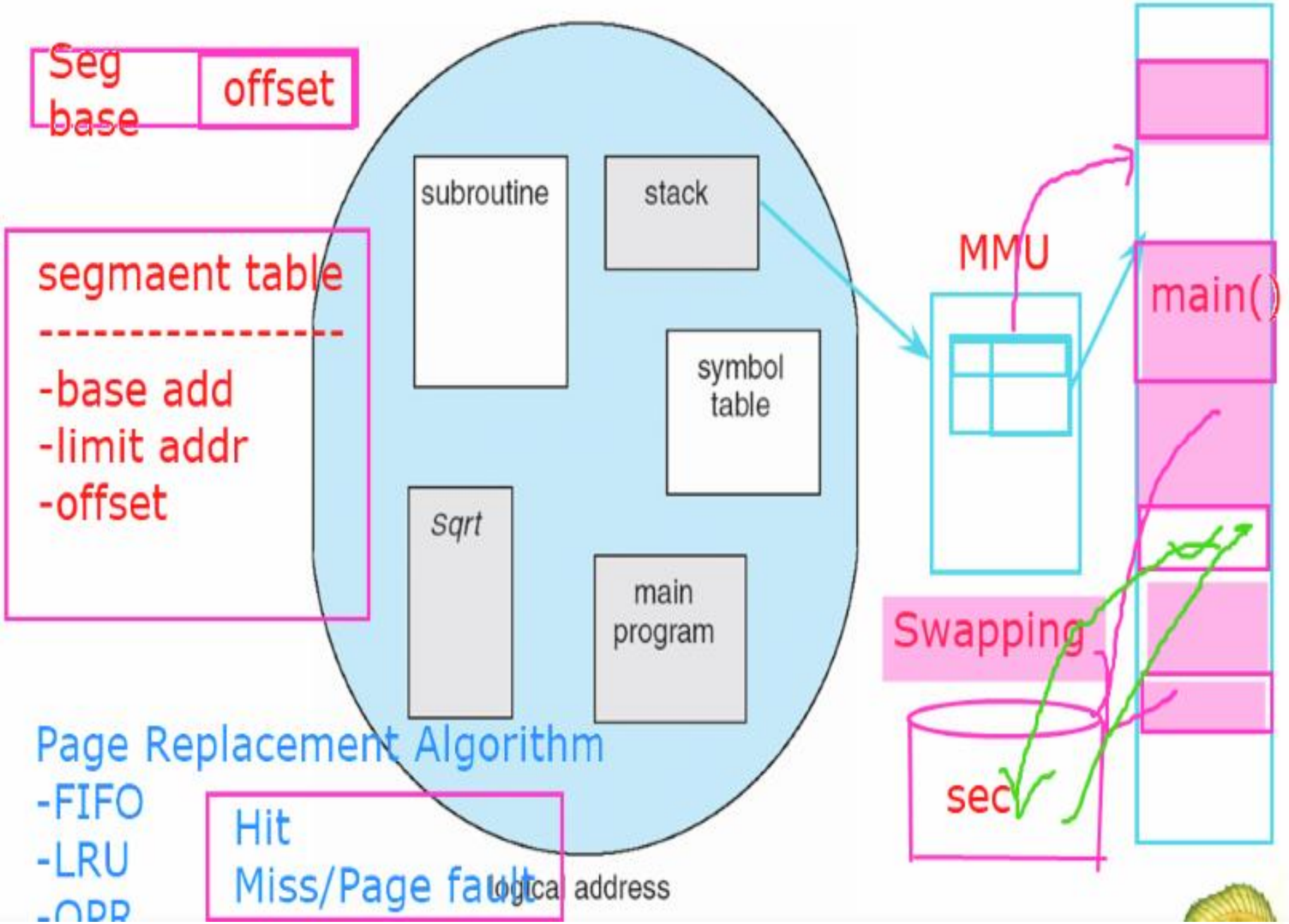


Segmentation



Example of Segmentation




Seg base
offset
segmaent table
-base add
-limit addr
-offset
subroutine
stack
symbol table
Sqrt
main program
MMU
main()
Swapping
sec
Page Replacement Algorithm
-FIFO
-LRU
Hit
Miss/Page fault
logica address

Segmentation

- Memory-management scheme that supports user view of memory
- A program is a collection of segments
  - A segment is a logical unit such as:

    main program

    procedure

    function

    method

    object

    local variables, global variables

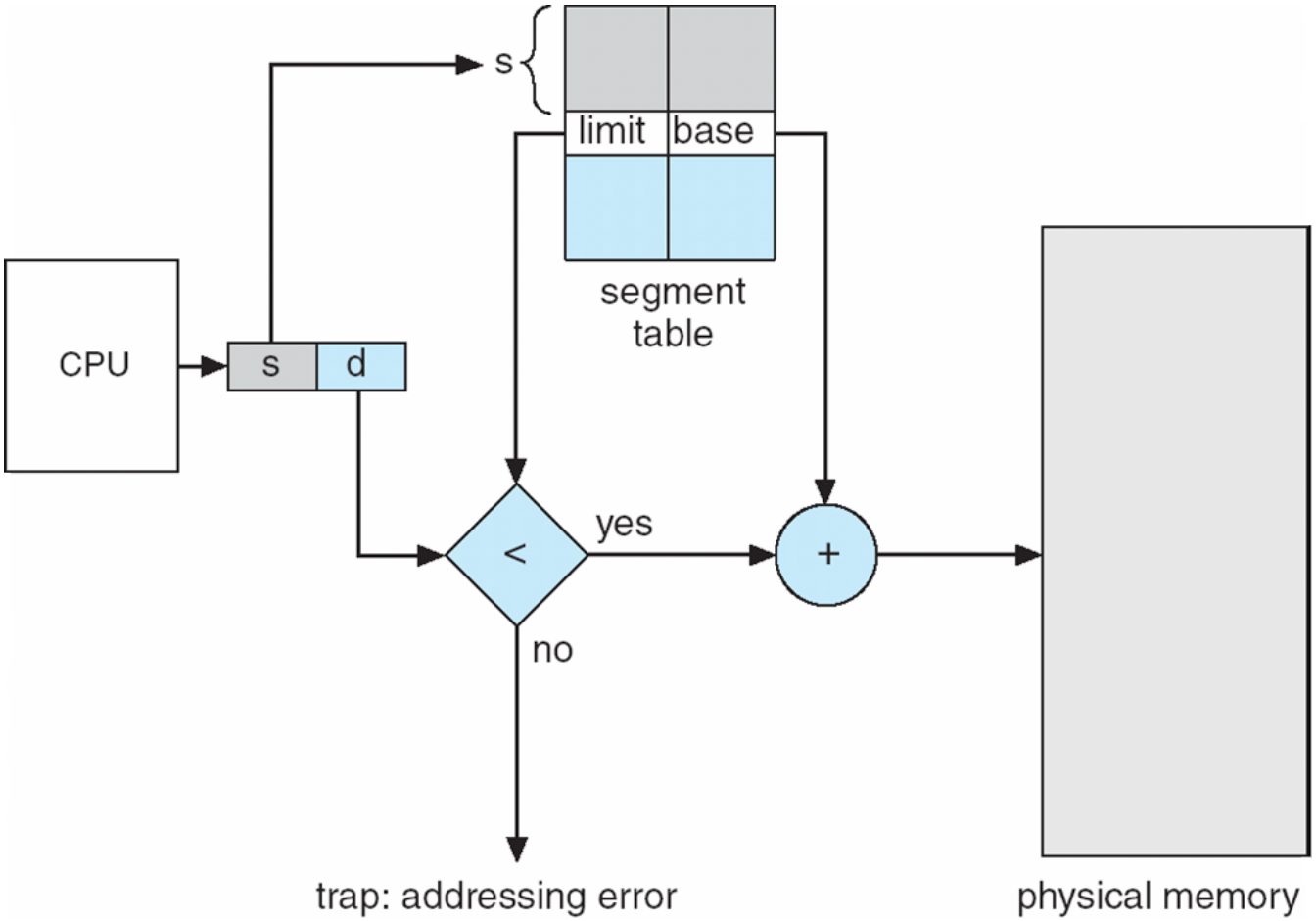
    common block

    stack

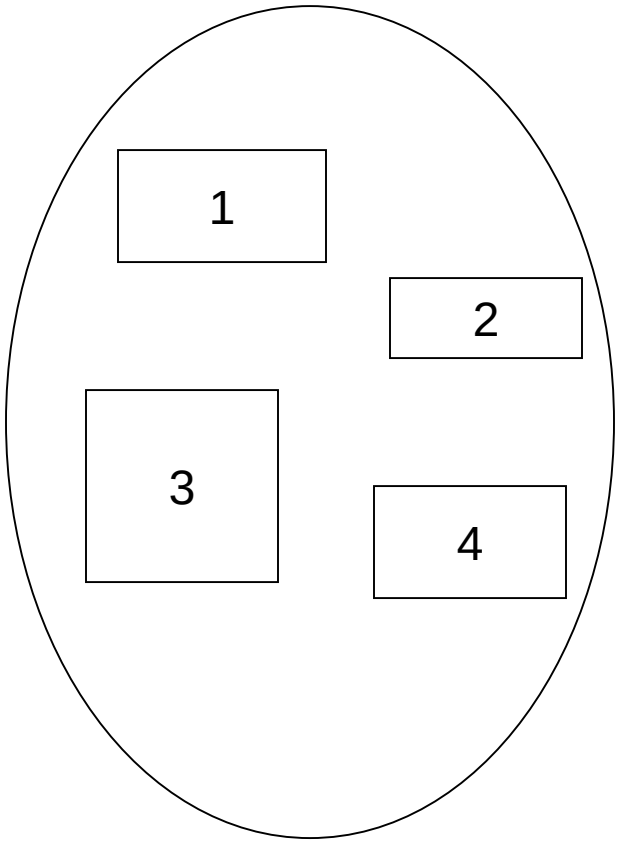
    symbol table

    arrays

Segmentation Hardware



Logical View of Segmentation



user space

physical memory space

Segmentation Architecture

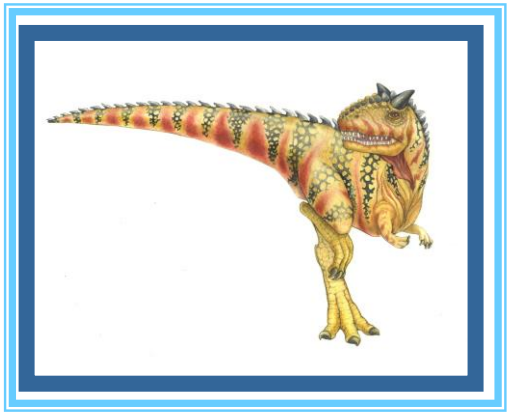
- Logical address consists of a two tuple:

  <segment-number, offset>,

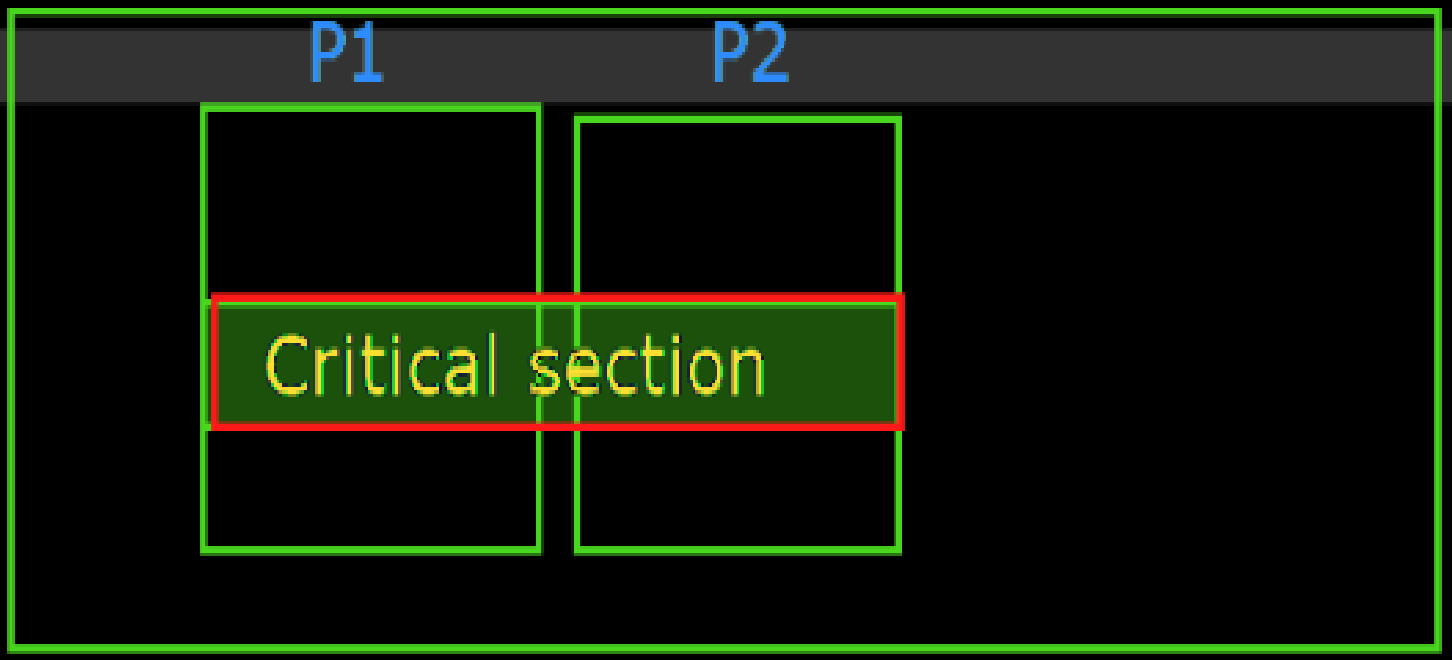
- **Segment table** – maps two-dimensional physical addresses; each table entry has:
  - **base** – contains the starting physical address where the segments reside in memory
  - **limit** – specifies the length of the segment
- **Segment-table base register (STBR)** points to the segment table's location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

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

End of Chapter 8



Process Synchronization:



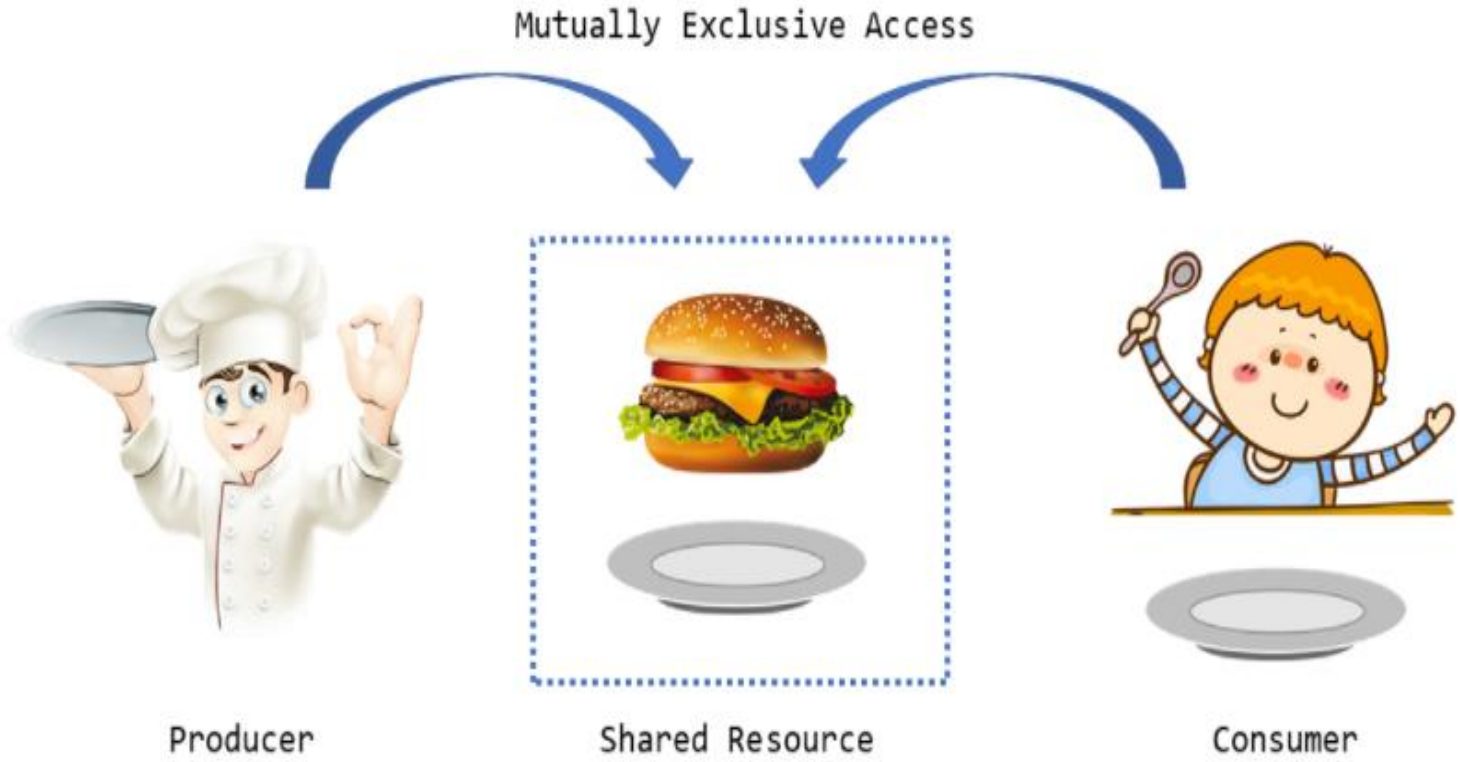
Producer - Consumer Problem
Dinning Philosophers
Reader Writer Problem

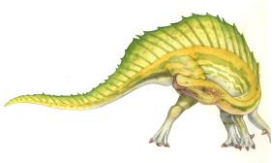
-Memory Management

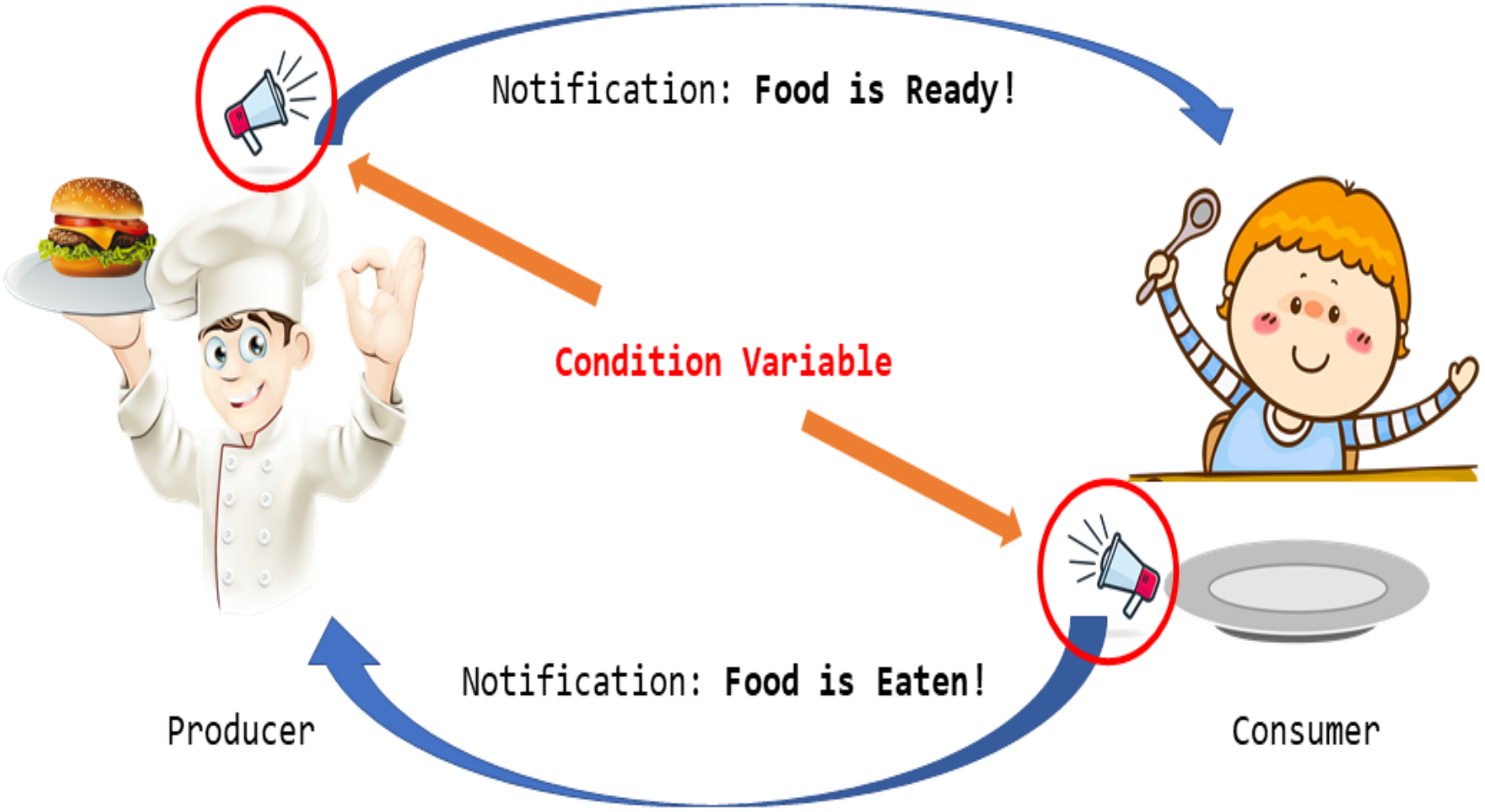
Process Synchronization:

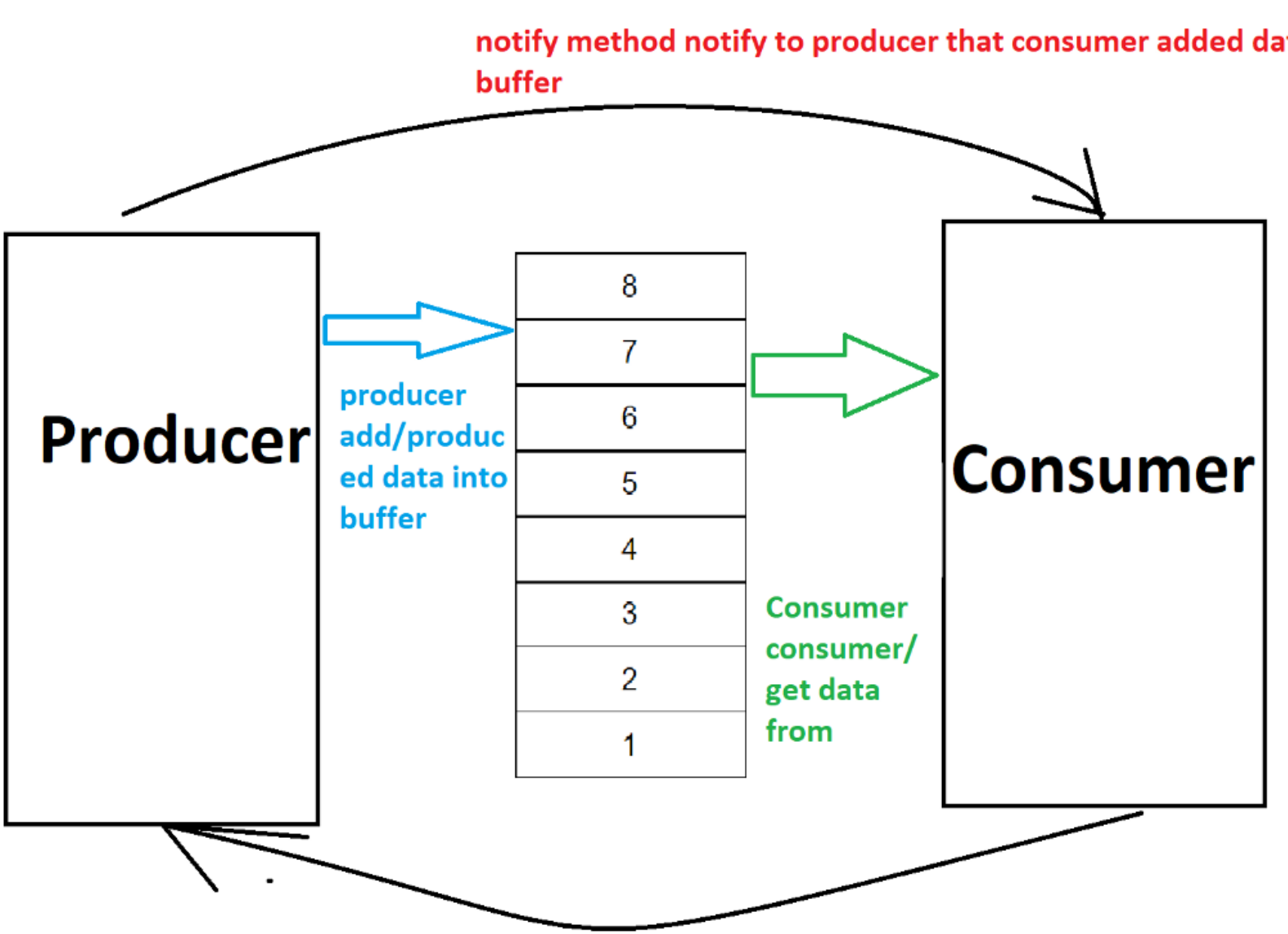
Producer - Consumer Problem
Dinning Philosophers
Reader Writer Problem

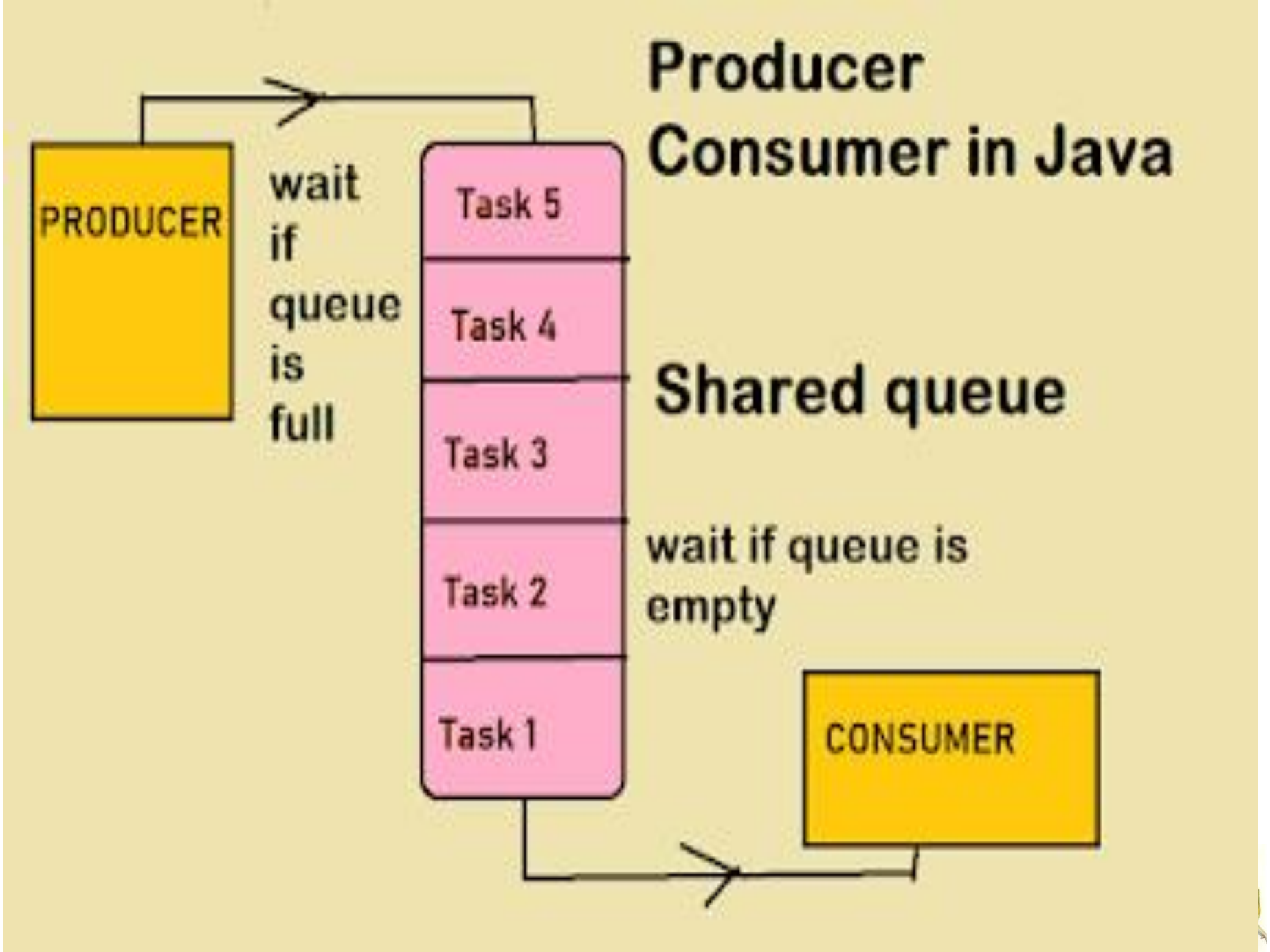
Solutions
---
set & Test
Lock
semaphores

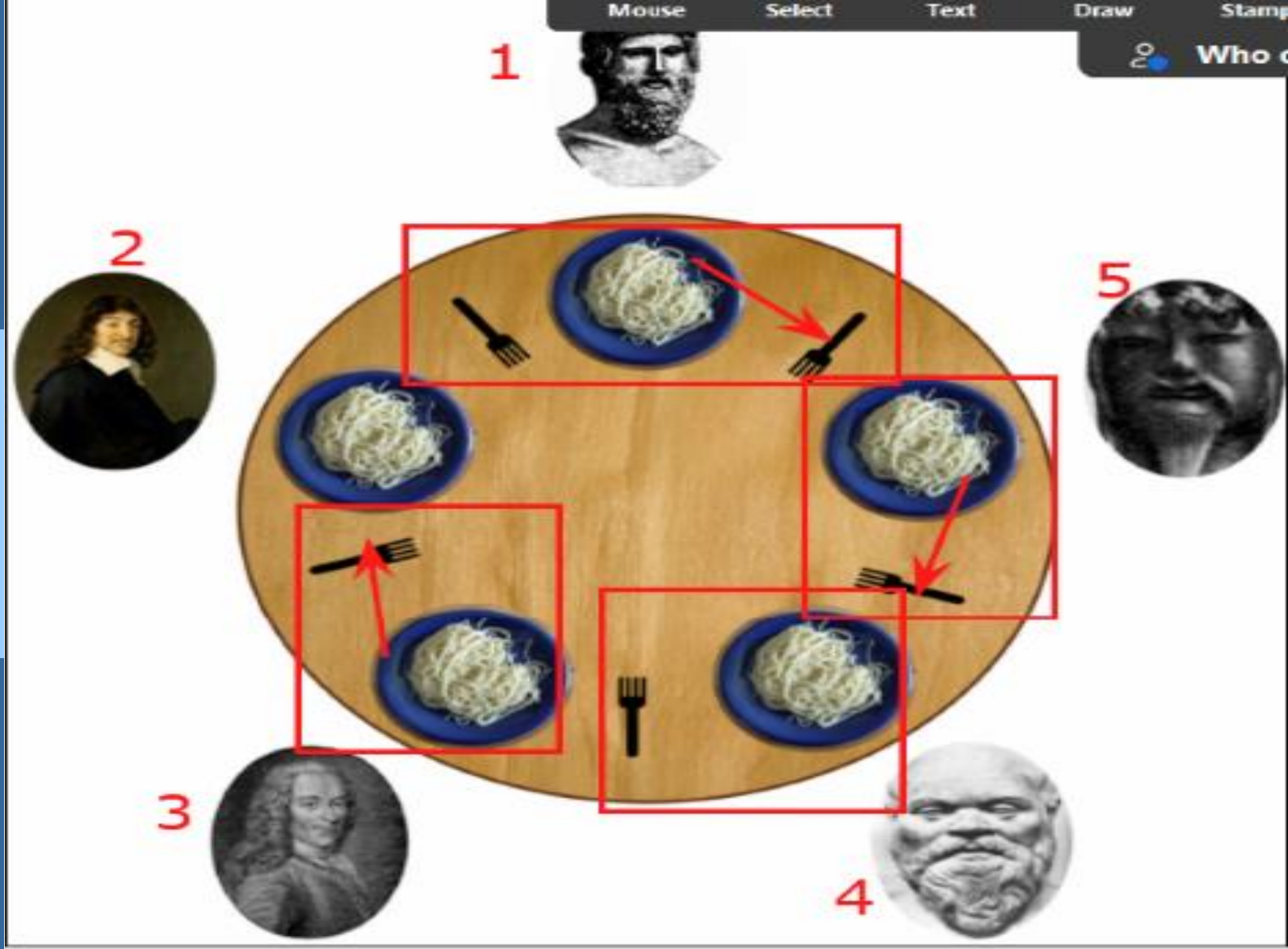





Notification: Food is Ready!
Condition Variable
Notification: Food is Eaten!
Producer
Consumer


notify method notify to producer that consumer added data on buffer
Producer
producer add/produced data into buffer
8
7
6
5
4
3
2
1
Consumer consumer/ get data from
Consumer
notify method call and notify to producer consumer consumes the data.


Producer
Consumer in Java
PRODUCER
wait
if
queue
is
full
Task 5
Task 4
Task 3
Task 2
Task 1
Shared queue
wait if queue is
empty
CONSUMER


Mouse
Select
Text
Draw
Stamp
Who
1
2
3
4
5