# Memory Management

**Prof J P Misra**
**BITS, Pilani**

# We have learnt

- Concept of multiprogramming
  - It is defined for uniprocessor systems
  - More than on program can reside in memory but at any time only one can be excuting .
- Concept of process
  - Process is an instance of executing program
  - Process can be in various state at different point of time
- Memory is one of the important resource which OS need to manage

# What is Memory

- Memory is a device which can store information. It supports two operations
  - Store ( write )
  - Load (Read)
- Memory devices take some finite amount of time to perform read / write operation

# Memory subsystem design requirement

- Storage capacity ( larger the better)
- Speed of access ( faster access)
- Cost of storage ( lower the better)

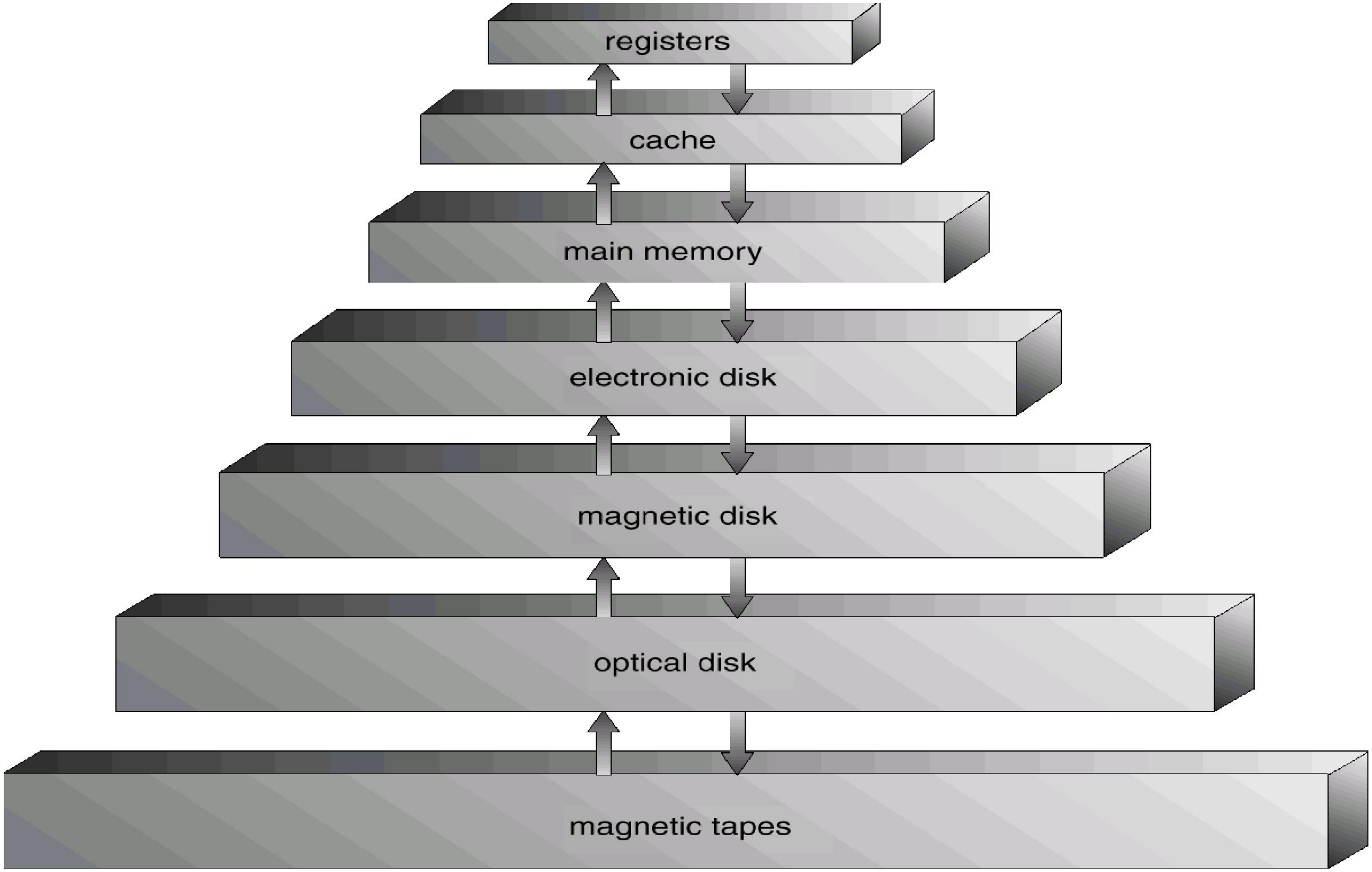
# Types of memory

- Based on persistence of information
  - Volatile
  - Non volatile
- Based on technology/ material used
  - Semiconductor memory
    - RAM (Random access memory)
    - ROM (Read only memory )
  - Magnetic memory
    - Hard disk
    - Magnetic tape
  - Optical memory
    - CD/DVD

# Memory Management

- Three design constraints of memory subsystem
  - large Size
  - high Speed
  - low Cost
- Across the spectrum of the technologies following relationship holds
  - Smaller access time , greater per bit cost
  - Greater capacity, smaller per bit cost
  - Greater capacity , greater access time

# To meet the contradictory  design requirement , organize memory in hierarchical manner



- As one goes down the Hierarchy , the following conditions occur:
  - Decreasing cost per bit
  - Increasing capacity
  - Increasing access time

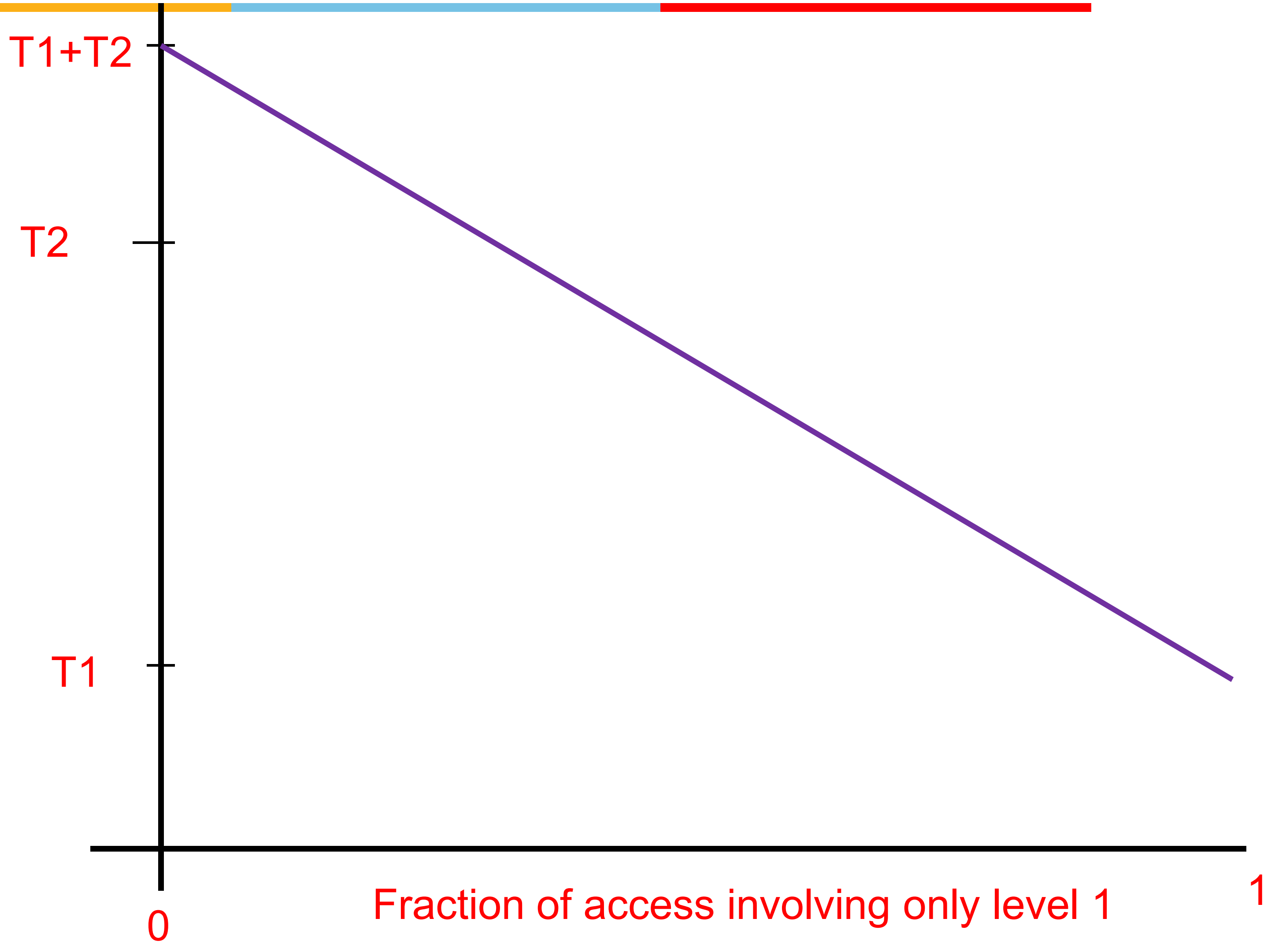
# Example (two level memory)

Processor has access to two level of memory

- Level 1 contains 1000 words and access time(T1) is 0.1 Micro second
- Level 2 contains 100,000 words and access time (T2) is 1 Micro second

If the word is found in level 1

- then it is accessed in 0.1 Micro sec
- else 1.1 micro sec


T1+T2
T2
T1
0
1
Fraction of access involving only level 1

# Observations

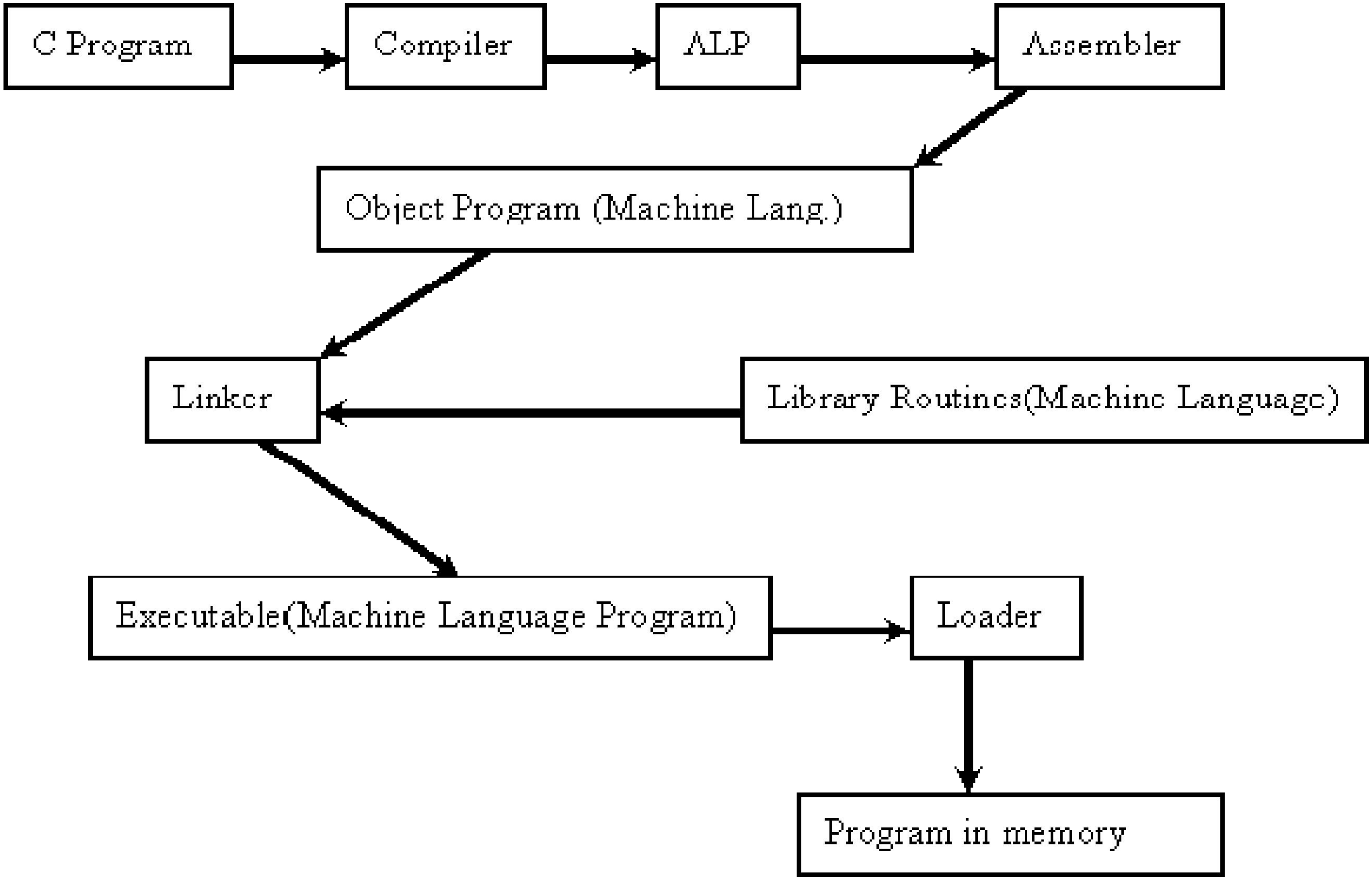
- We observe that if memory access at level 2 is less frequent then overall access time is close to level 1 access time

- The basis for validity of this condition is a principal known as locality of reference

- During course of execution of program , memory references for both data and instruction tends to cluster

# Memory Management Requirement

- Relocation : ability to load and execute when programs are  loaded at different location at different point of time.
- Protection
- Sharing
- Logical organization
- Physical organization



C Program

Compiler

ALP

Assembler

Object Program (Machine Lang.)

Linkcr

Library Routincs(Machinc Languagc)

Executable(Machine Language Program)

Loader

Program in memory

# Binding of Instructions and Data to Memory

**Address binding of instructions and data to memory addresses can occur at three different point of time.**

- **Compile time**: If memory location known a priori, (starting location) absolute code can be generated; must recompile code if starting location changes. Example MS DOS .COM format programs.
- **Load time**: Must generate *relocatable* code if memory location is not known at compile time. Final binding is delayed until load 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*). Most general purpose operating systems use this method.

# Loading Program into main memory

it is assumed that OS occupies some fixed portion of memory and rest is available to user processes.

Based on requirement and functionality different memory management methods are adopted.

Issues:

# Memory management issues

- contiguous  memory allocation ?
- How much memory to allocate to each process ?
- How to partition memory ?
  - Static  /   dynamic partitioning
  - Equal  /  unequal partition

# Fixed Partitioning

- Main memory is divided into number of fixed size partition at system generation time.
  - A processes can be loaded into a partition of equal or greater size

Limitation:

- The maximum number of processes that can be in system becomes fixed at time of system generation
- Could lead to inefficient utilization of  memory .

# Equal size fixed Partition

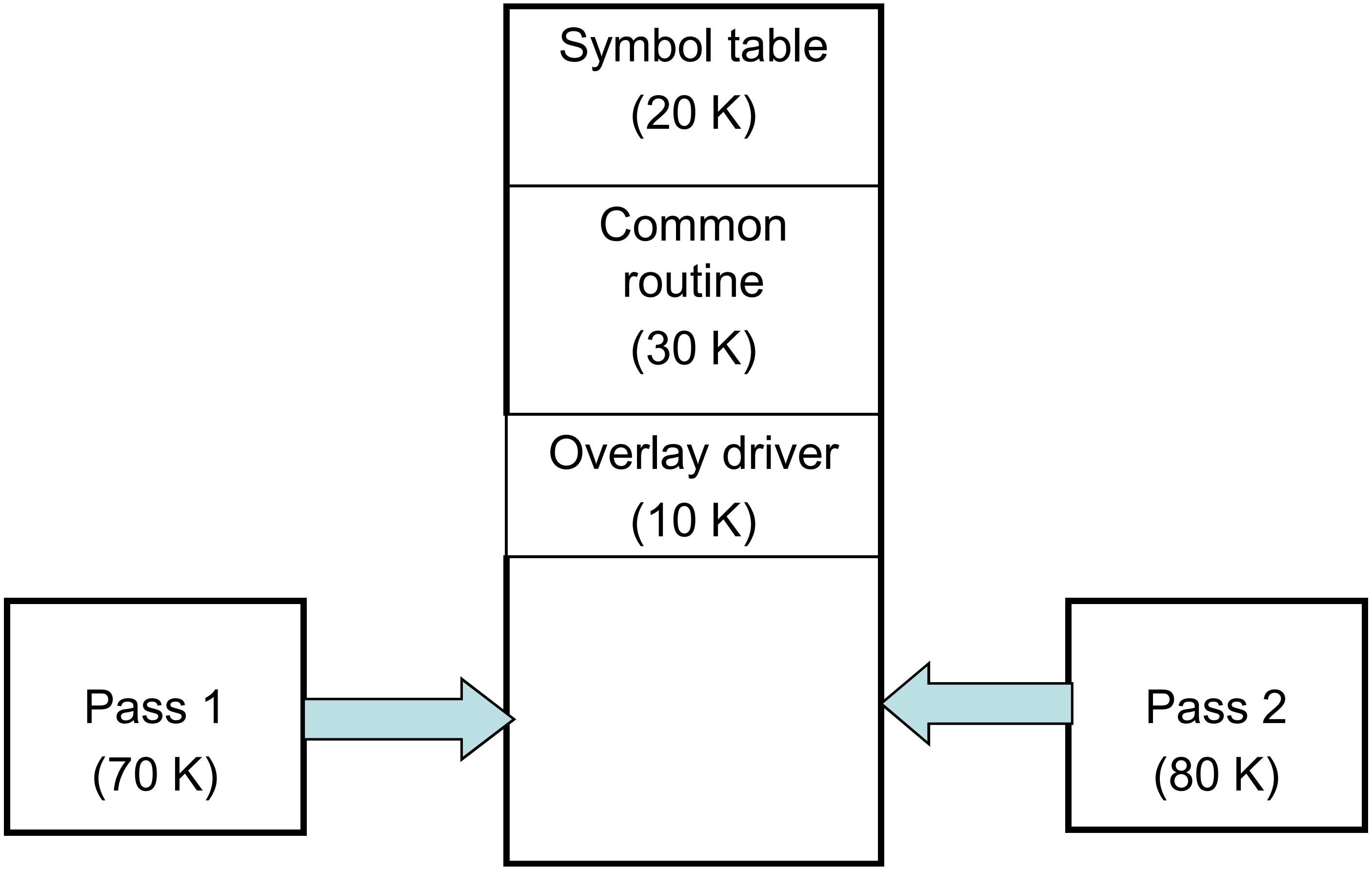
- Easy to implement
- If total user memory space is X and size of partition is Y, ( Y< X) then number of partitions in system will be X / Y. This is the maximum number of processes that can be loaded in the memory at any given time
- If program size is smaller than the size of partition, the remaining space remains unutilized
- A program may be too big to fit into a partition

# Overlays

- Needed when process is larger than amount of memory allocated to it.

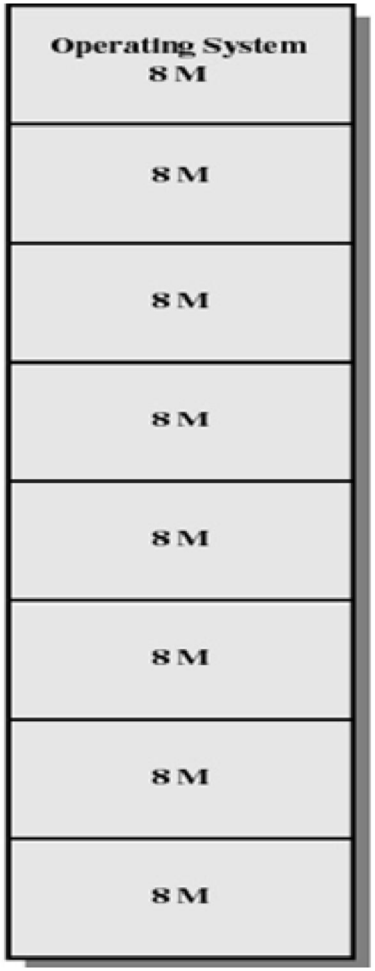
- Keep in memory only those instructions and data that are needed at any given time

- Implemented by user, no special support provided from operating system,
- programming of overlay structure is complex

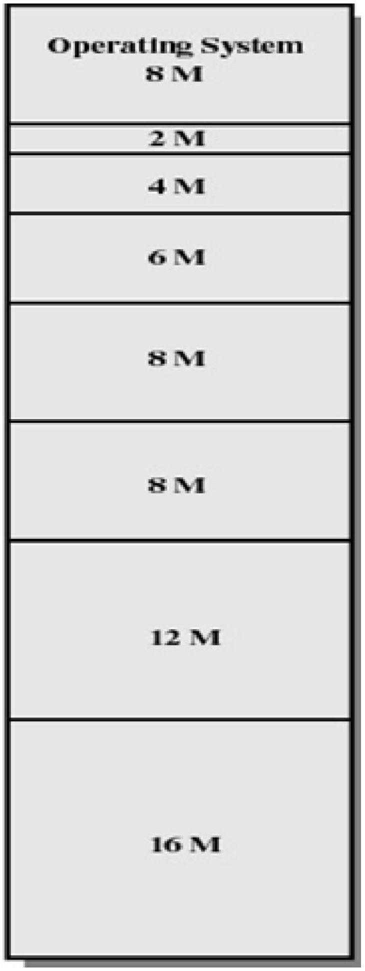

Symbol table
(20 K)
Common
routine
(30 K)
Overlay driver
(10 K)
Pass 1
(70 K)
Pass 2
(80 K)

# Unequal size partition

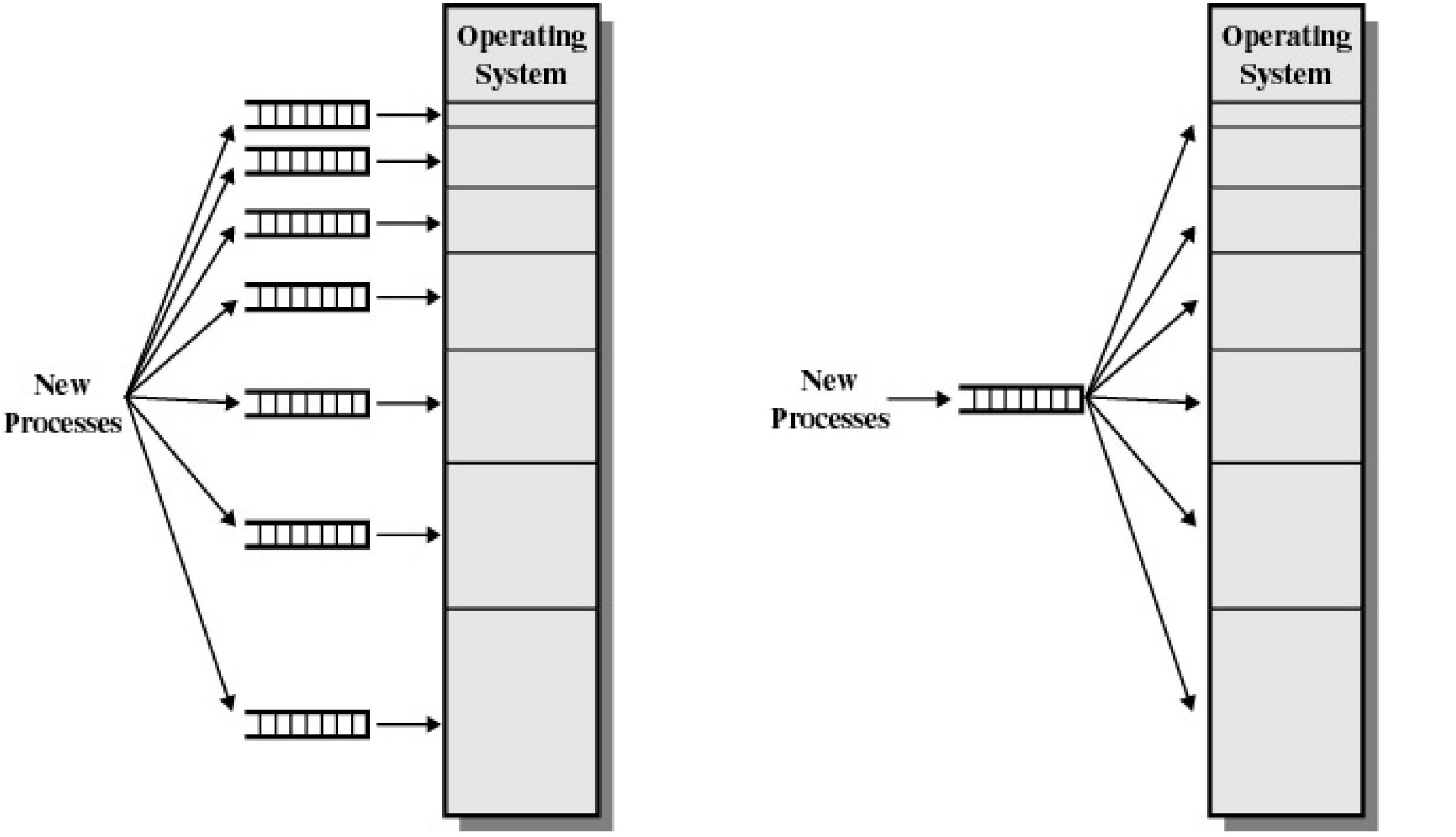
- We create fixed number of unequal size partition
- Program is loaded into best fit partition
  - processes are assigned in such a way as to minimize wasted memory within a partition
- queue for each partition



Equal size partition



Unequal size partition



(a) One process queue per partition

(b) Single process queue

innovate achieve lead

# Thank You