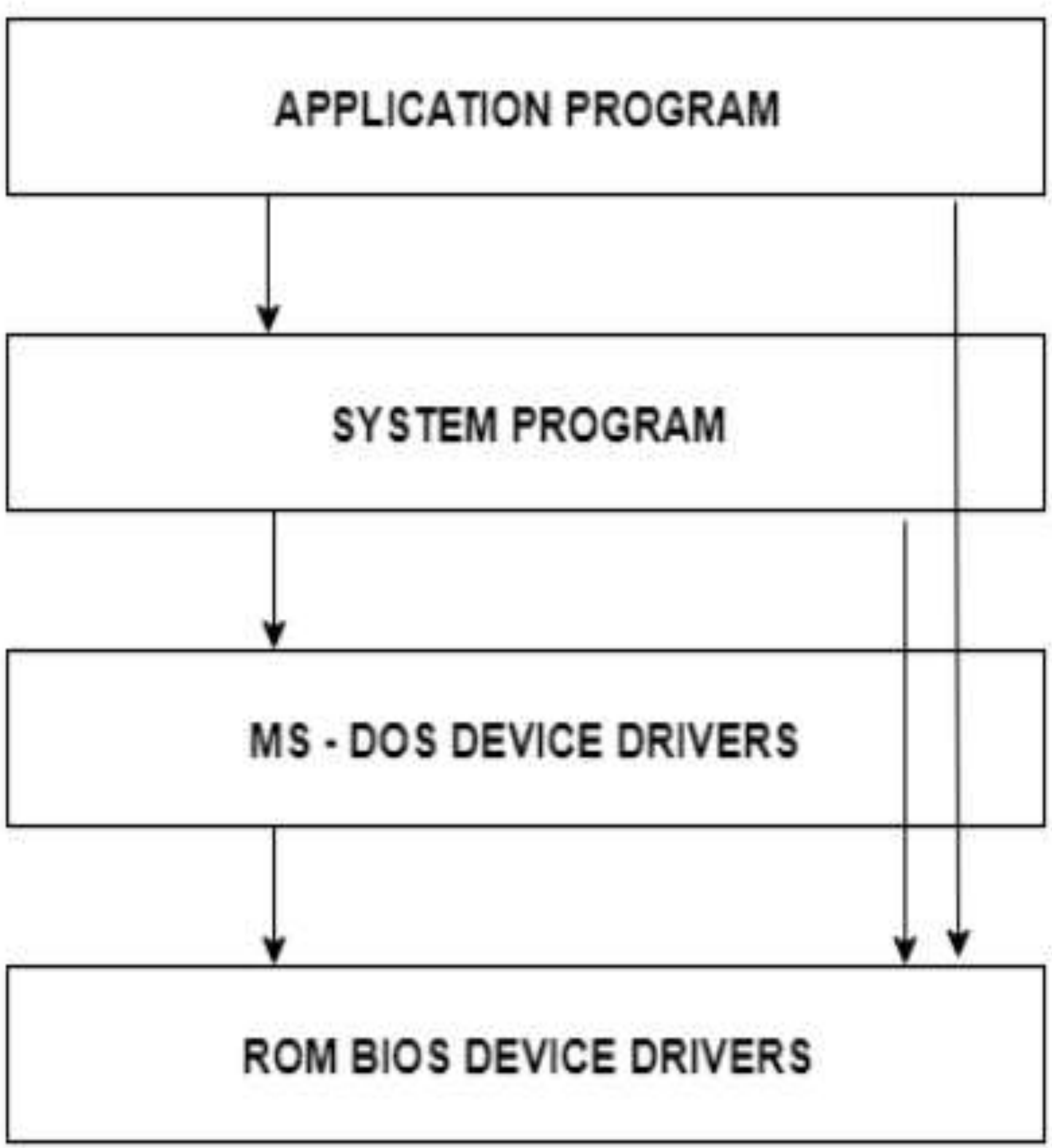
Operating System Structure

- An operating system is a construct that allows the user application programs to interact with the system hardware.
- Since the operating system is such a complex structure, it should be created with most care so it can be used and modified easily.
- An easy way to do this is to create the operating system in parts. Each of these parts should be well defined with clear inputs, outputs and functions.

Simple Structure

- There are many operating systems that have a rather simple structure. These started as small systems and rapidly expanded much further than their scope.
- A common example of this is MS-DOS. It was designed simply for a recess amount for people. There was no indication that it would become so popular.
- An image to illustrate the structure of MS-DOS is as follows:



APPLICATION PROGRAM

SYSTEM PROGRAM

MS - DOS DEVICE DRIVERS

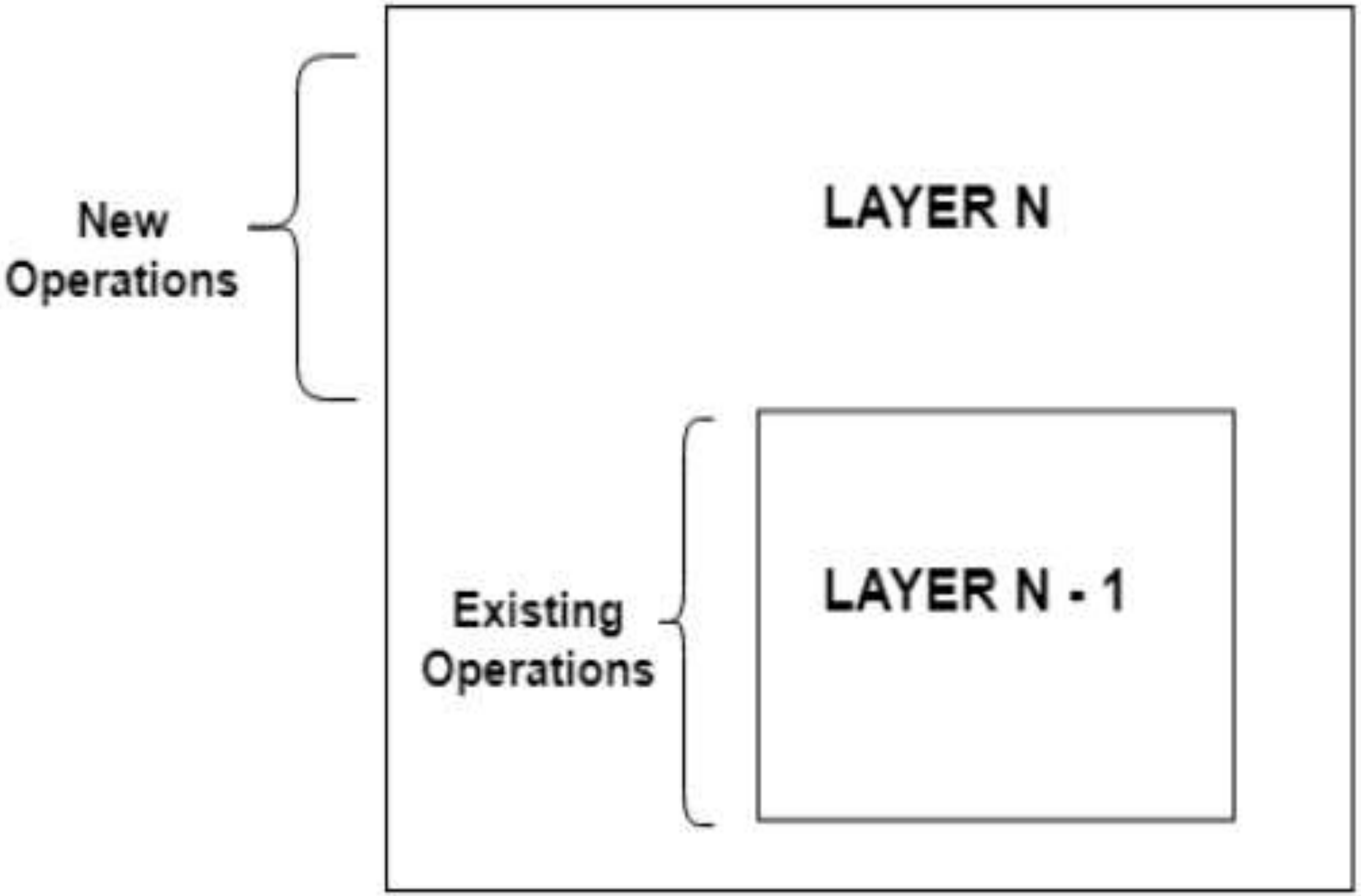
ROM BIOS DEVICE DRIVERS

MS-DOS STRUCTURE

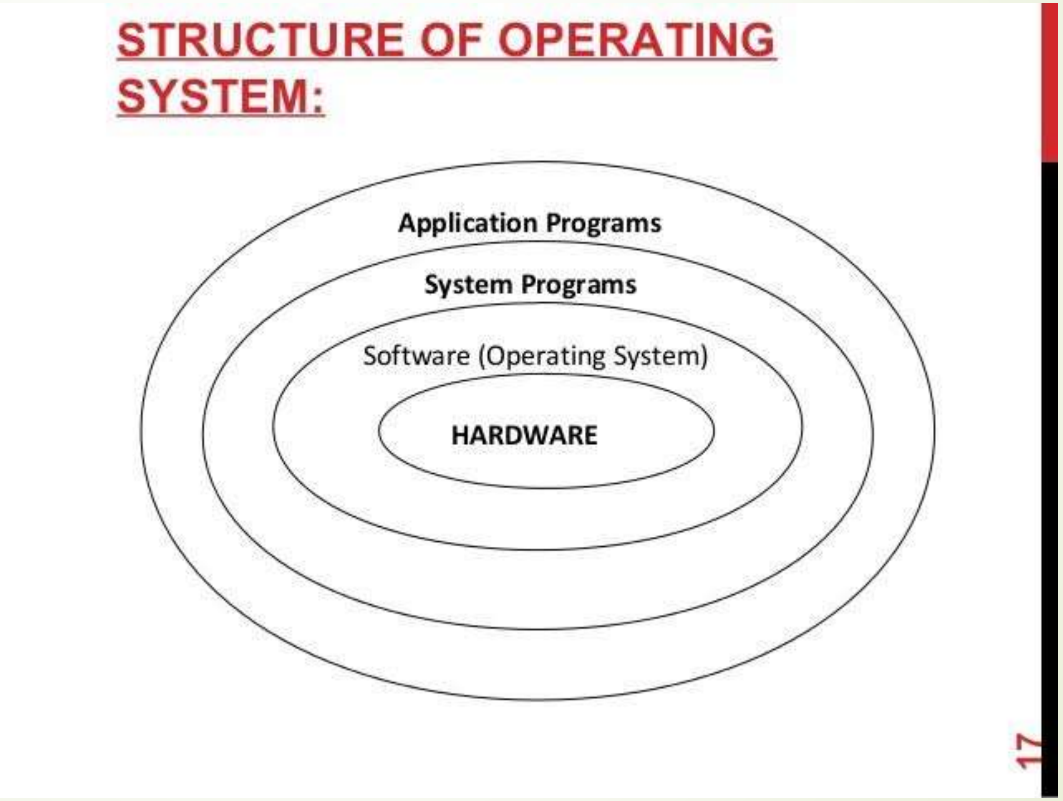
- It is better that operating systems have a modular structure, unlike MS-DOS. That would lead to greater control over the computer system and its various applications.
- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers

Layered Structure

- One way to achieve modularity in the operating system is the layered approach. In this, the bottom layer is the hardware and the topmost layer is the user interface.
- An image demonstrating the layered approach (views) is as follows:



Layered Structure of Operating System



STRUCTURE OF OPERATING SYSTEM:

Application Programs

System Programs

Software (Operating System)

HARDWARE

- **Multiprogramming** needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs (job pool) in system is kept in memory
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job

- **Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory ⇨**process**
  - **Job scheduling:** Which jobs to bring to memory from job pool on disk.

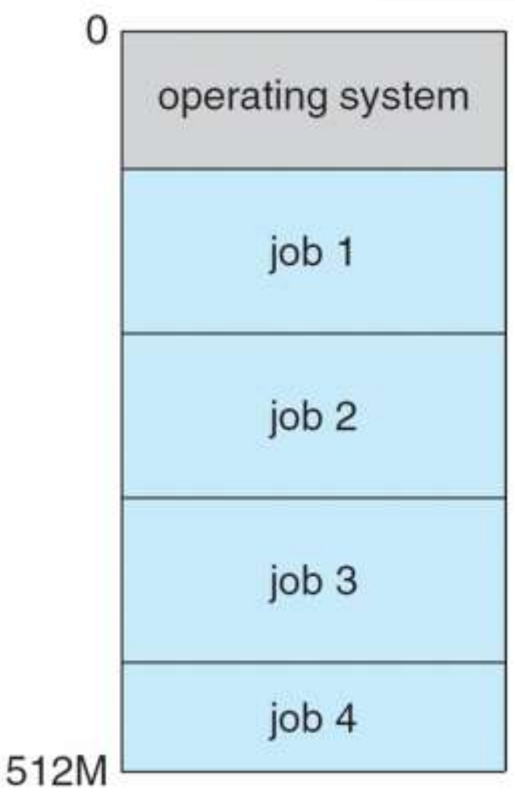
  (The **job pool** contains both **jobs** that are currently executing and **jobs** that have been scheduled but are not yet being executed**)**

  - If several jobs ready to run at the same time ⇨ **CPU scheduling**

  (CPU scheduling is a **process** which allows one **process** to use the **CPU** while the execution of another **process** is on hold(in waiting state) . The aim of **CPU scheduling** is to make the system efficient, fast and fair.**)**

  - **Virtual memory** allows execution of processes not completely in memory

Memory Layout for Multiprogrammed System


0
operating system
job 1
job 2
job 3
job 4
512M