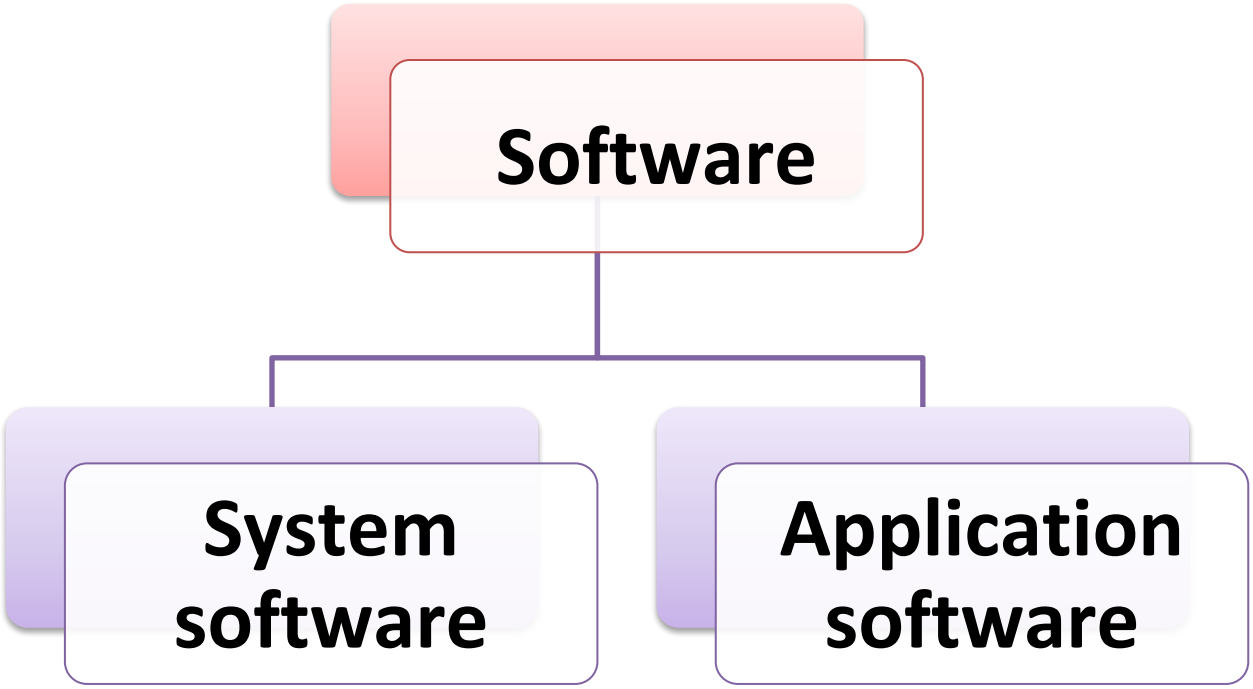
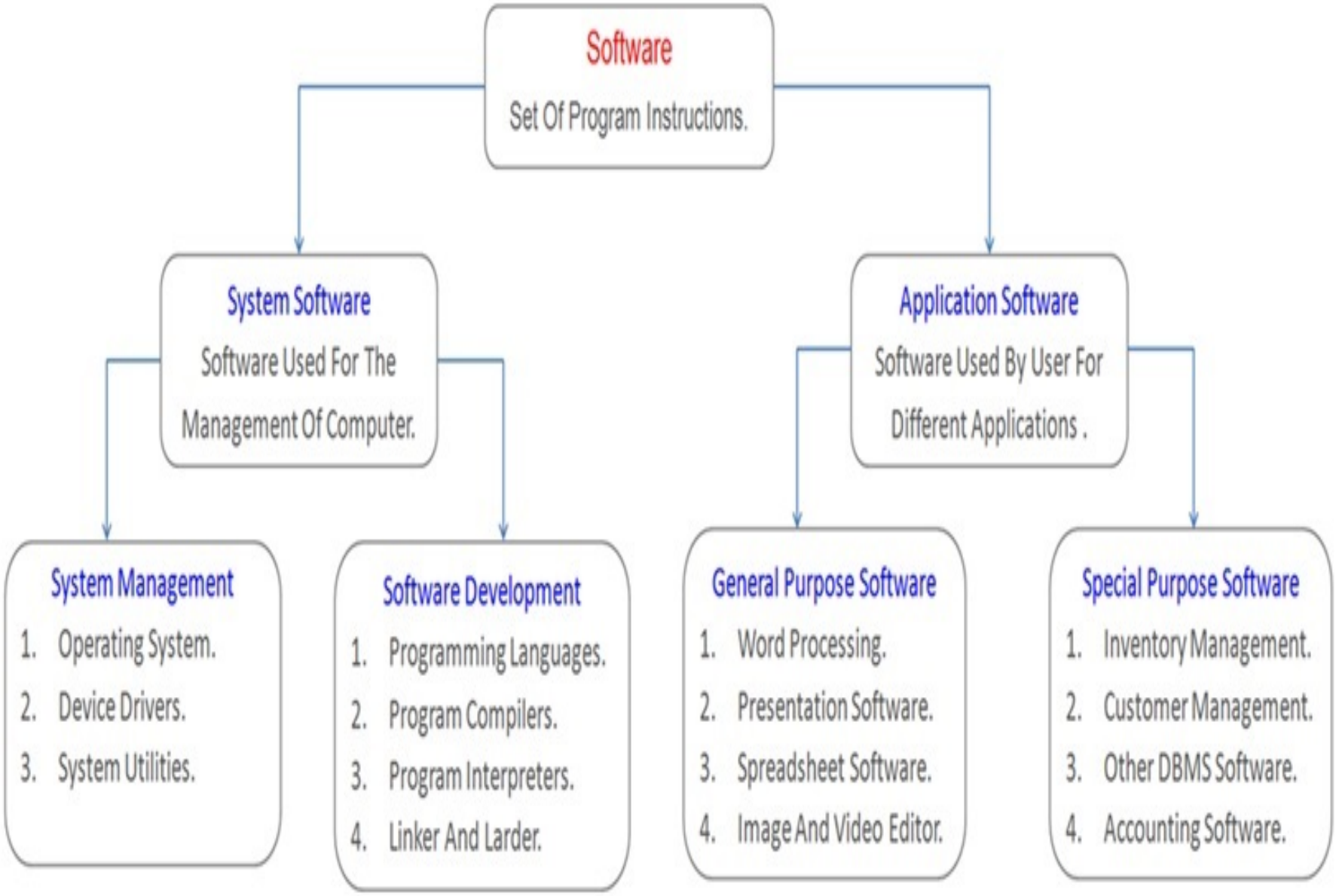
Operating System

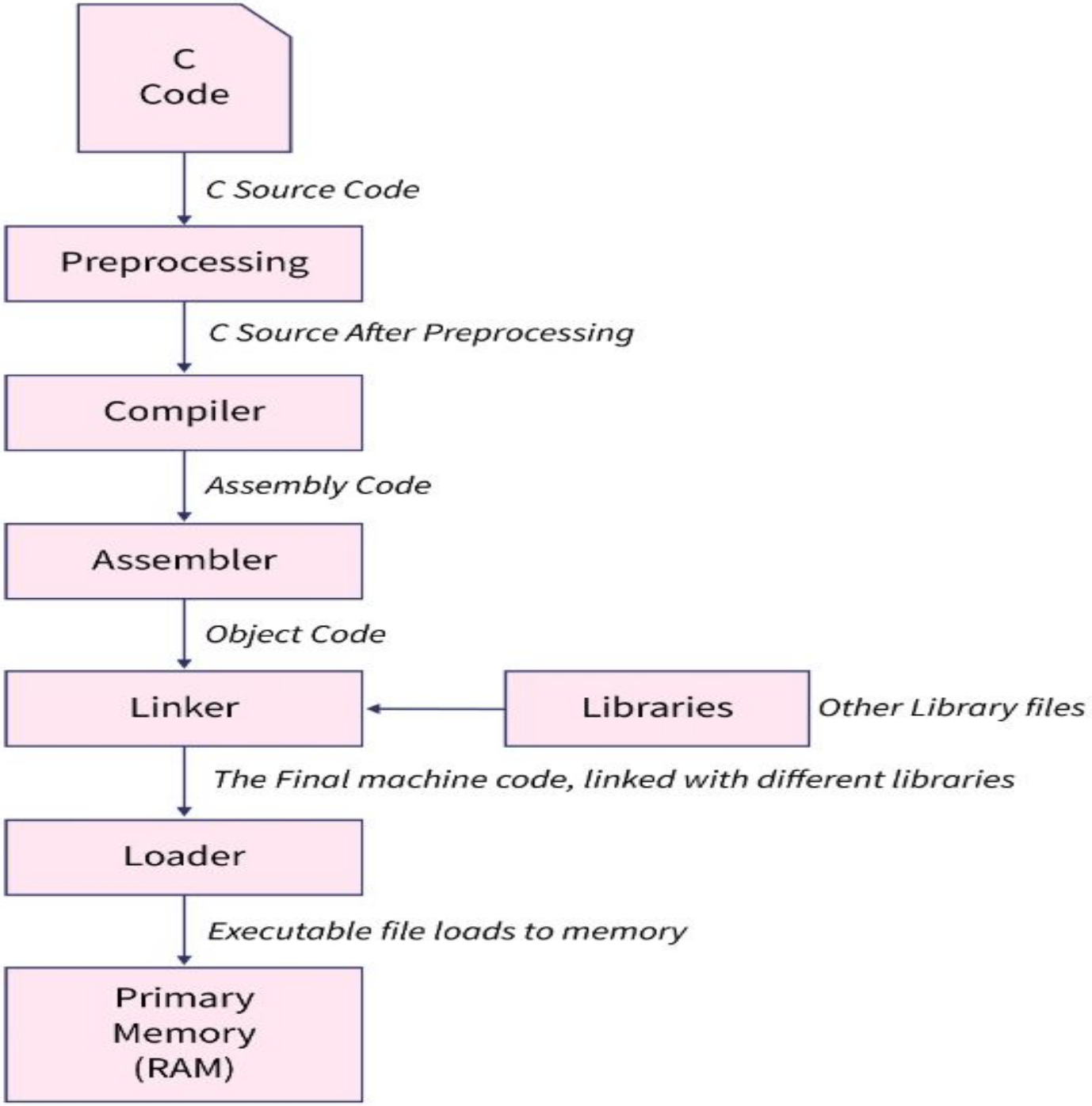
Software-

- Software is a set of instructions, data or programs used to operate computers and execute specific tasks.
- It is a collection of programs



System Software	Application Software
System Software maintain the system resources and gives the path for application software to run.	Application software is built for specific tasks.
Low level languages are used to write the system software.	While high level languages are used to write the application software.
Without system software, the system can’t run.	While without application software system always runs.
System software runs when the system is turned on and stop when the system is turned off.	While application software runs as per the user’s request.
System Software programming is complex than application software.	Application software programming is simpler as comparison to system software.

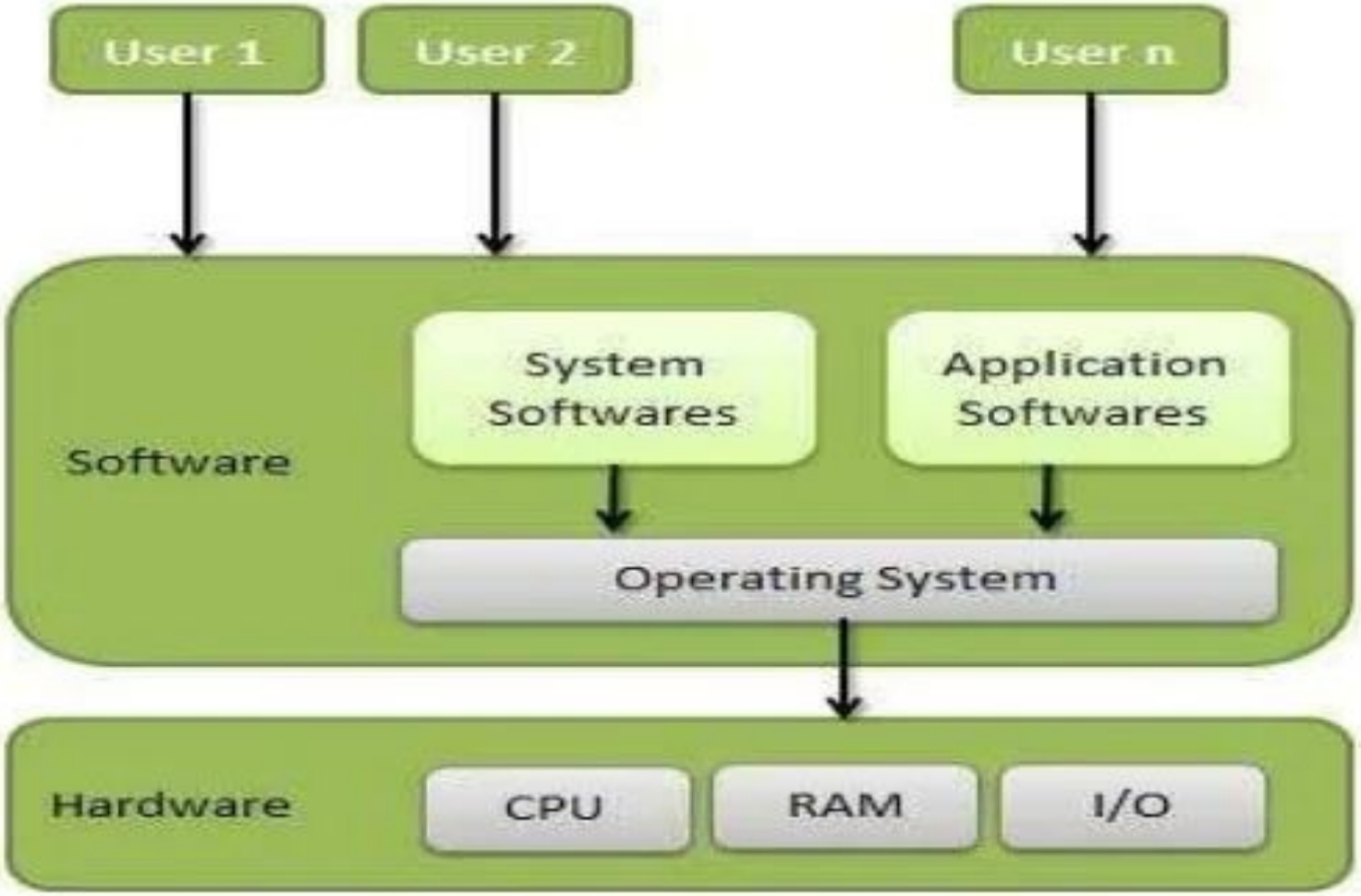

Software
Set Of Program Instructions.
System Software
Software Used For The Management Of Computer.
Application Software
Software Used By User For Different Applications .
System Management
1. Operating System.
2. Device Drivers.
3. System Utilities.
Software Development
1. Programming Languages.
2. Program Compilers.
3. Program Interpreters.
4. Linker And Larder.
General Purpose Software
1. Word Processing.
2. Presentation Software.
3. Spreadsheet Software.
4. Image And Video Editor.
Special Purpose Software
1. Inventory Management.
2. Customer Management.
3. Other DBMS Software.
4. Accounting Software.


C
Code
C Source Code
Preprocessing
C Source After Preprocessing
Compiler
Assembly Code
Assembler
Object Code
Linker
Libraries
Other Library files
The Final machine code, linked with different libraries
Loader
Executable file loads to memory
Primary
Memory
(RAM)

What is an Operating System?

- An operating system is a program (or set of programs) that manages the computer hardware
- It also provides a basis for running application programs and acts as an intermediary between the computer user and the computer hardware
- Some operating systems are designed to be convenient, others are designed to be efficient, and still others are a combination of both

Operating System Diagram :-

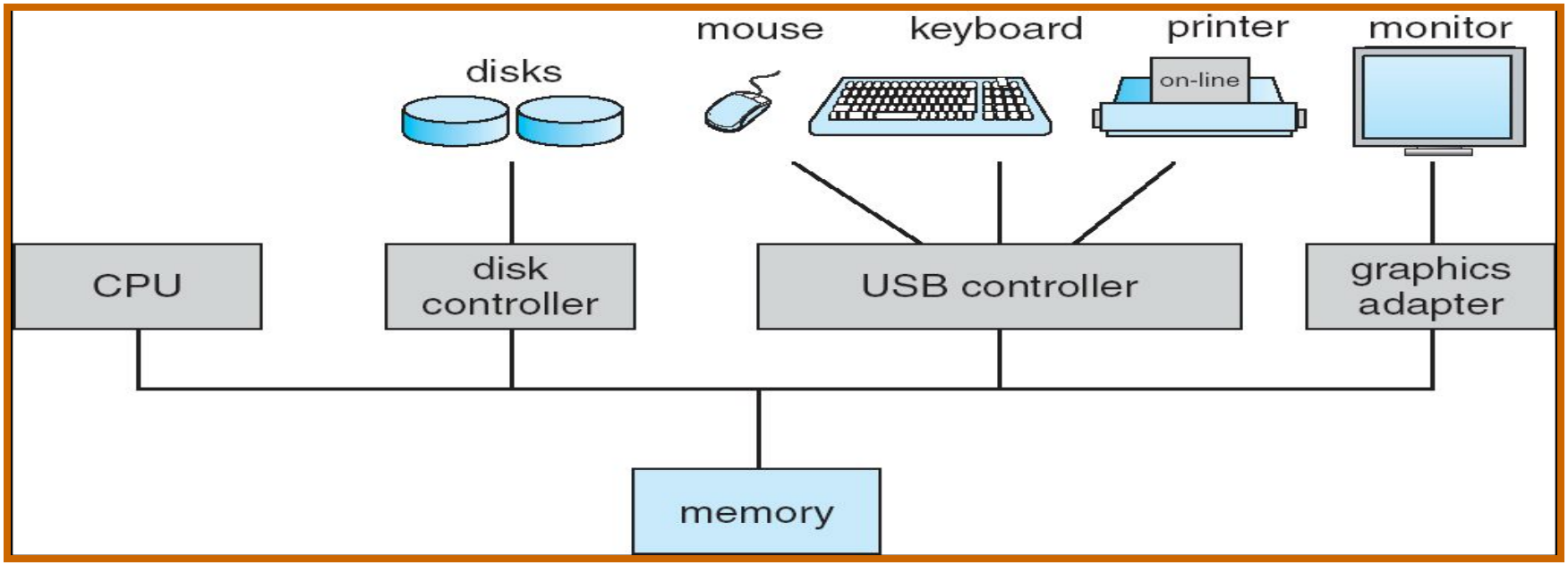


System View of an Operating System

- The operating system is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- The operating system is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

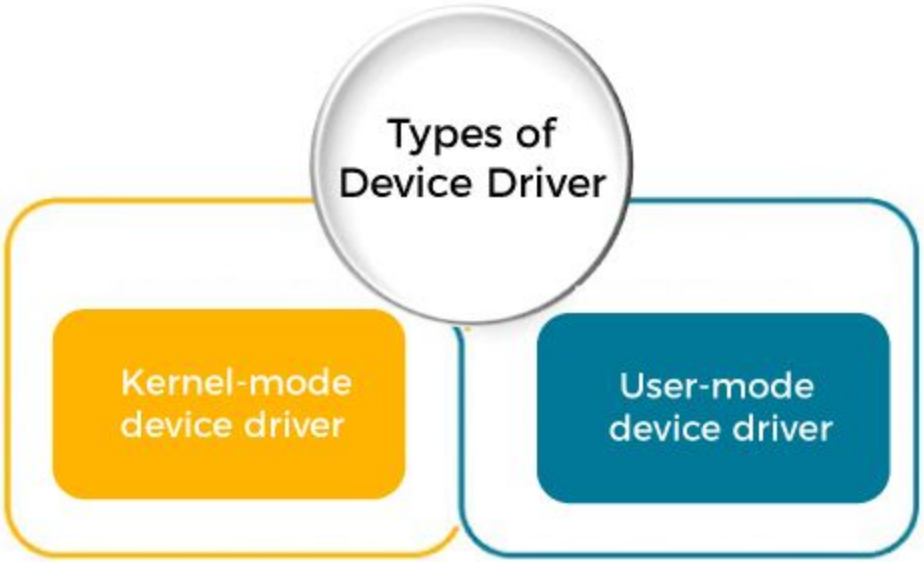
Computer Startup

- POST(power on self test)
- Bootstrap program(BIOS)



Drivers

- A device driver is a computer program that operates or controls a particular device attached to a computer
- Drivers are hardware-dependent and operating-system-specific.



Interpreter and Compiler

Interpreter translates just one statement of the program at a time into machine code.	Compiler scans the entire program and translates the whole of it into machine code at once.
An interpreter takes very less time to analyze the source code. However, the overall time to execute the process is much slower.	A compiler takes a lot of time to analyze the source code. However, the overall time taken to execute the process is much faster.
An interpreter does not generate an intermediary code. Hence, an interpreter is highly efficient in terms of its memory.	A compiler always generates an intermediary object code. It will need further linking. Hence more memory is needed.

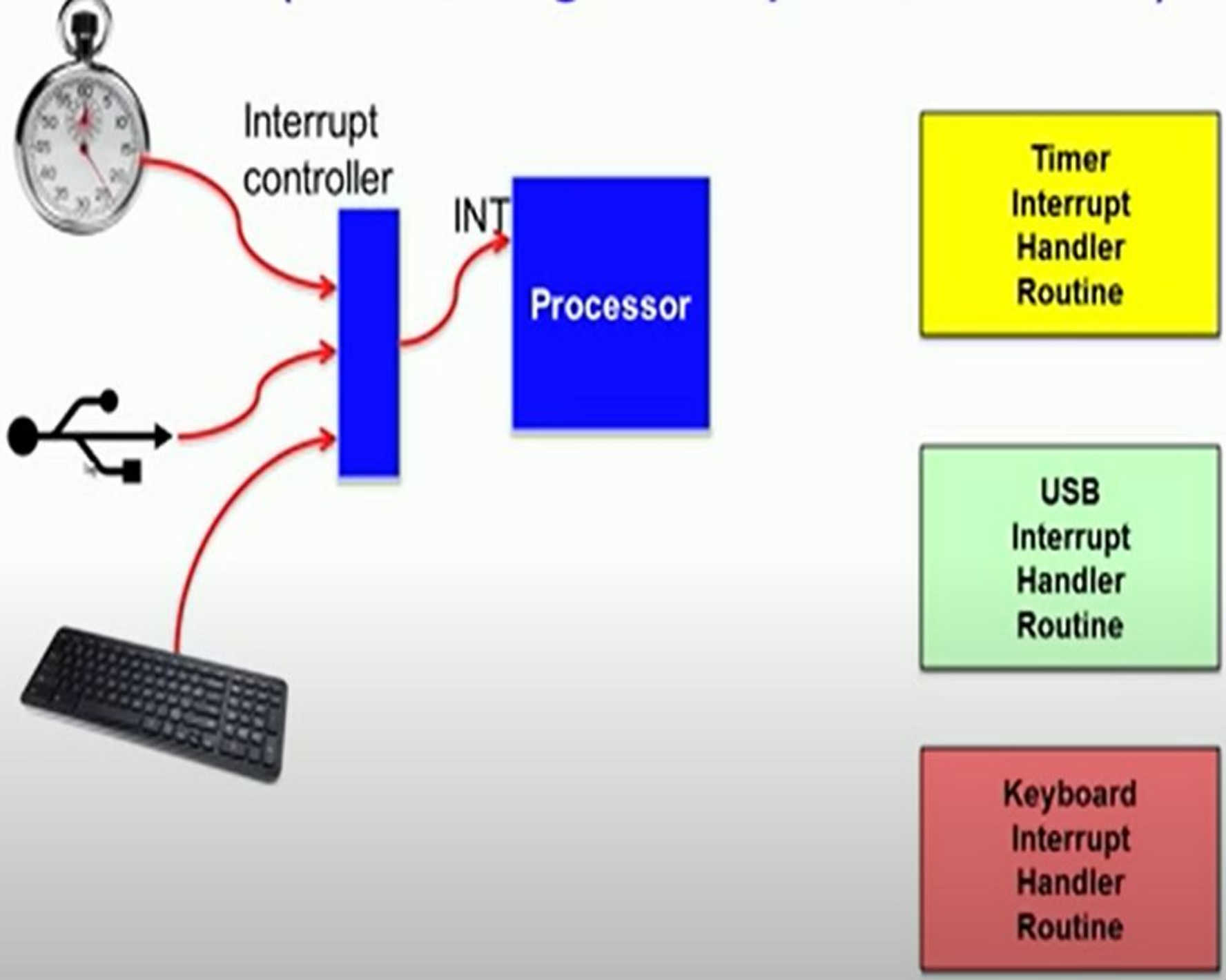
Keeps translating the program continuously till the first error is confronted. If any error is spotted, it stops working and hence debugging becomes easy.	A compiler generates the error message only after it scans the complete program and hence debugging is relatively harder while working with a compiler.
Interpreters are used by programming languages like php and Python for example.	Compliers are used by programming languages like C and C++ for example.

Interrupt

An interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention.

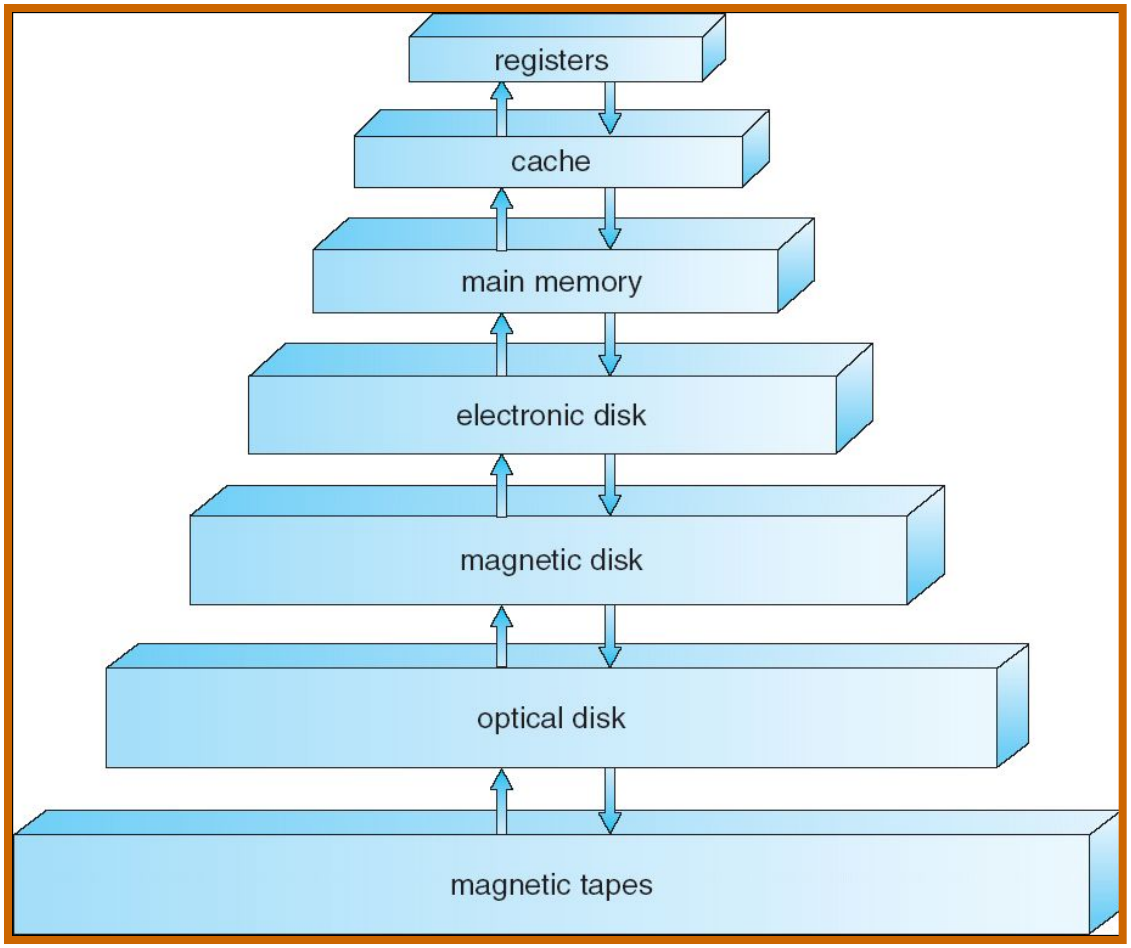
Two types of interrupt

- **Hardware Interrupt:**
  - Raise by hardware device
  - by sending signal to CPU through system bus
- **Software Interrupt:**
- Raised by user program
- by executing special operation called system call


Interrupt
controller
INT
Processor
Timer
Interrupt
Handler
Routine
USB
Interrupt
Handler
Routine
Keyboard
Interrupt
Handler
Routine

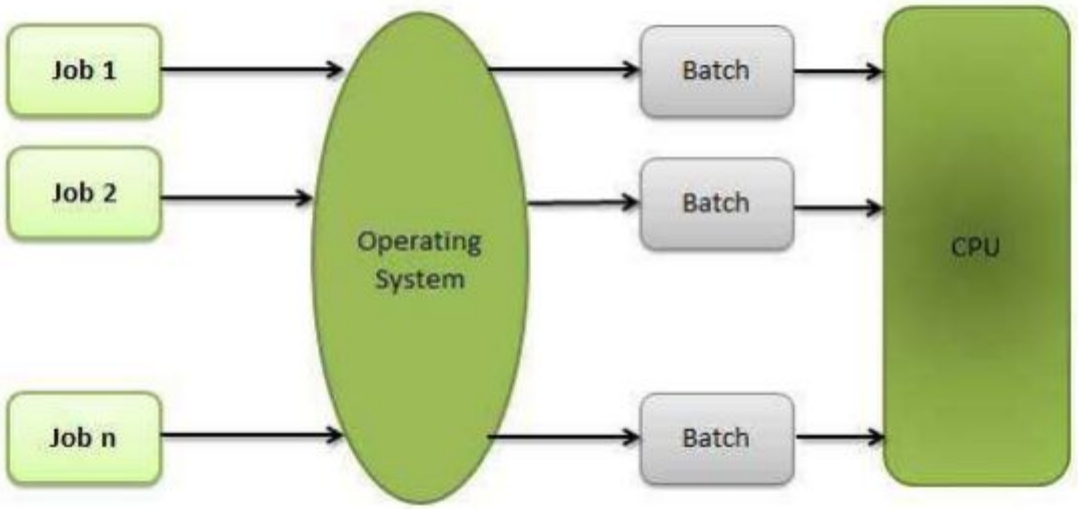
Storage Structure

- Storage systems organized in hierarchy.
  - Speed
  - Cost
  - Volatility



Types of Operating System

Batch System



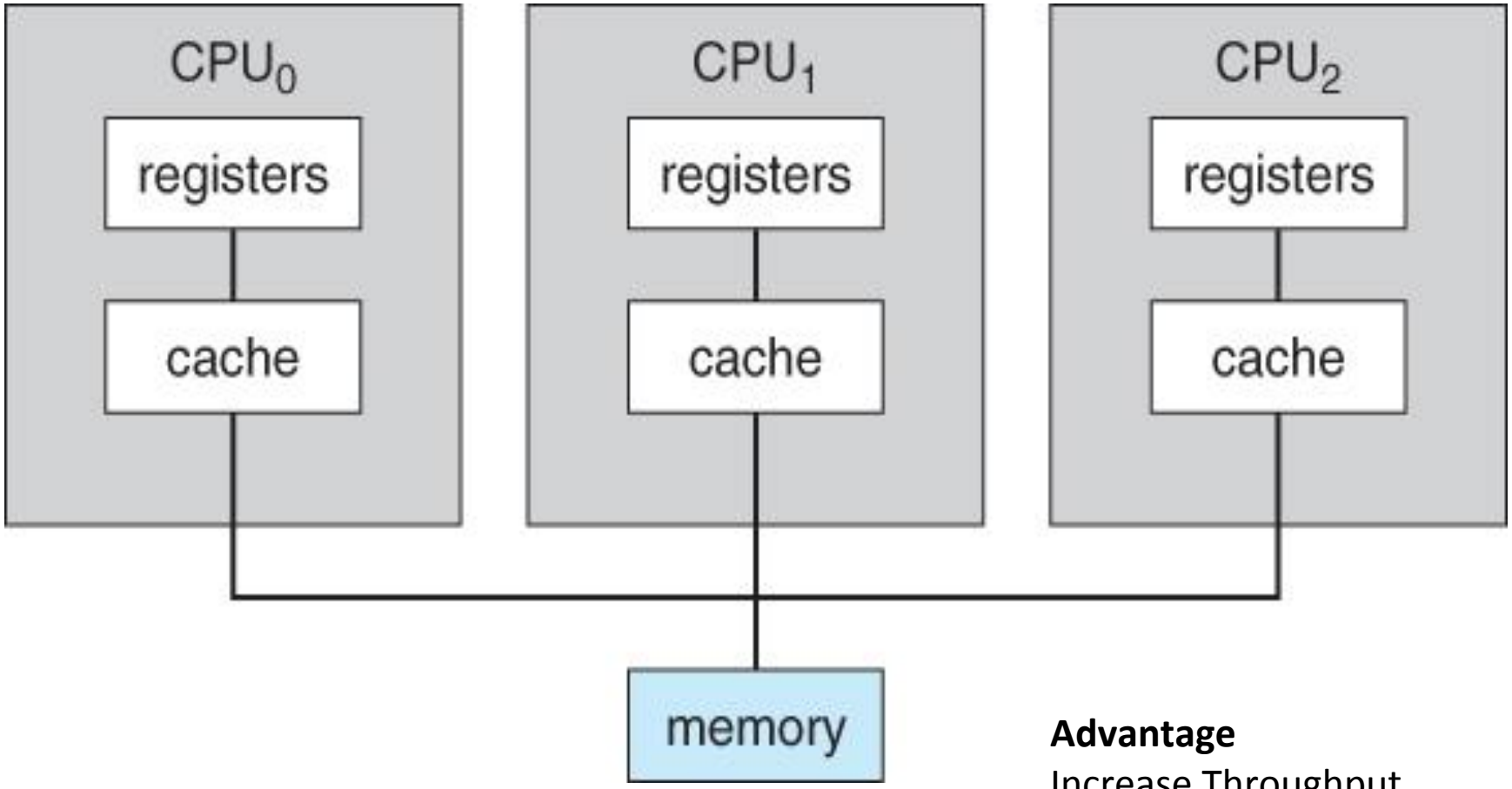
Disadvantage:

No human interaction

CPU Ideal

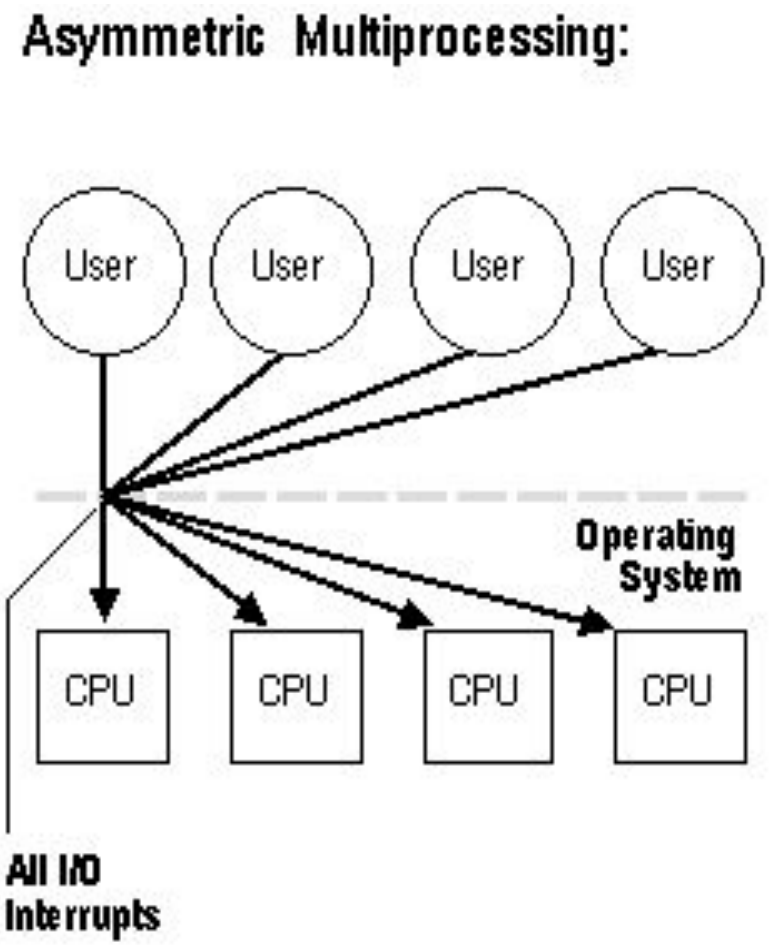
Cant set priority

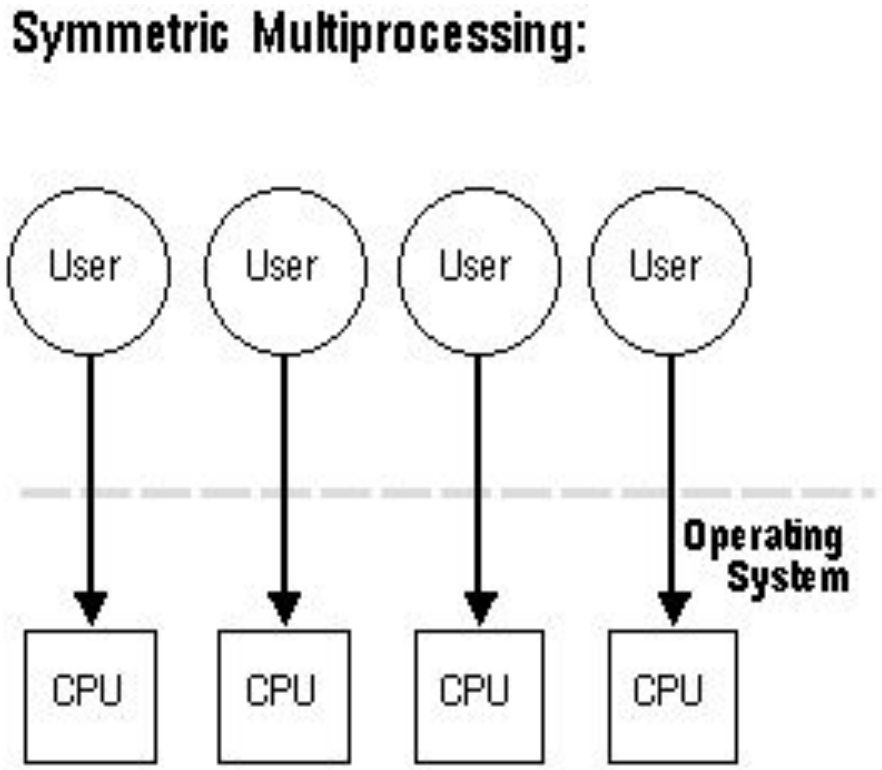
Multi-Processor System(parallel system or tightly coupled system)



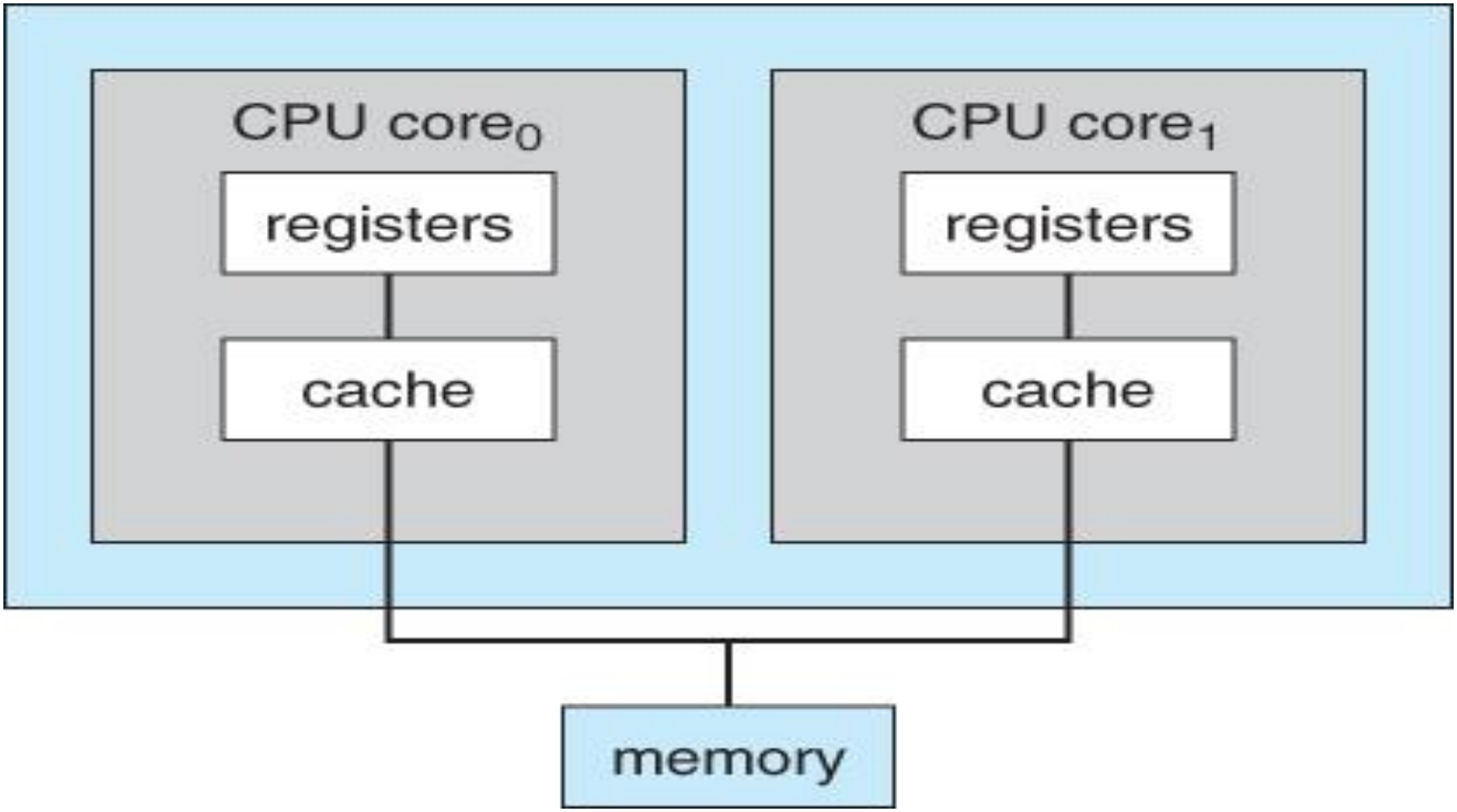
Advantage
Increase Throughput
Increase Reliability
Economy of Scale

Asymmetric vs symmetric multiprocessing

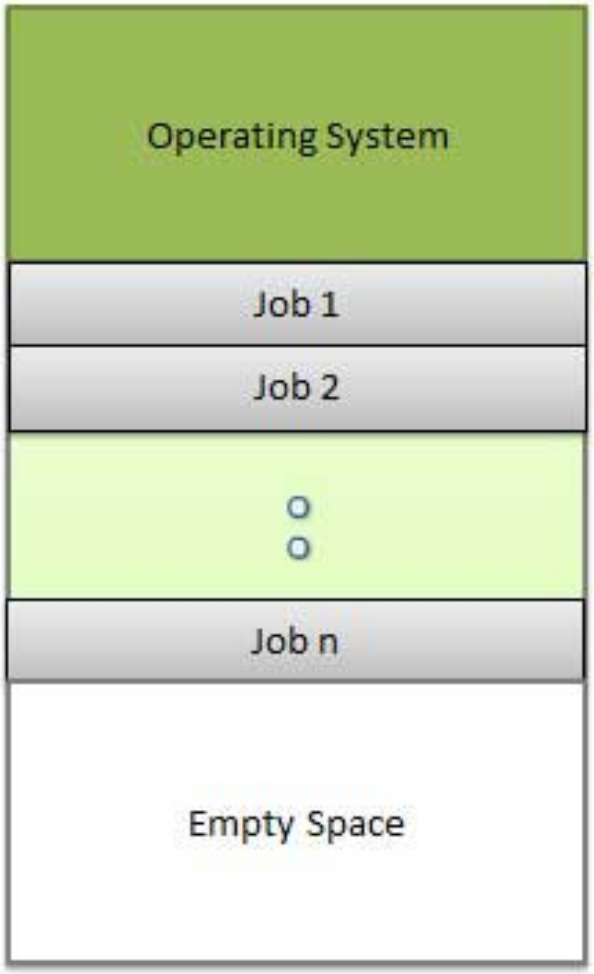




Two cores place on single chip

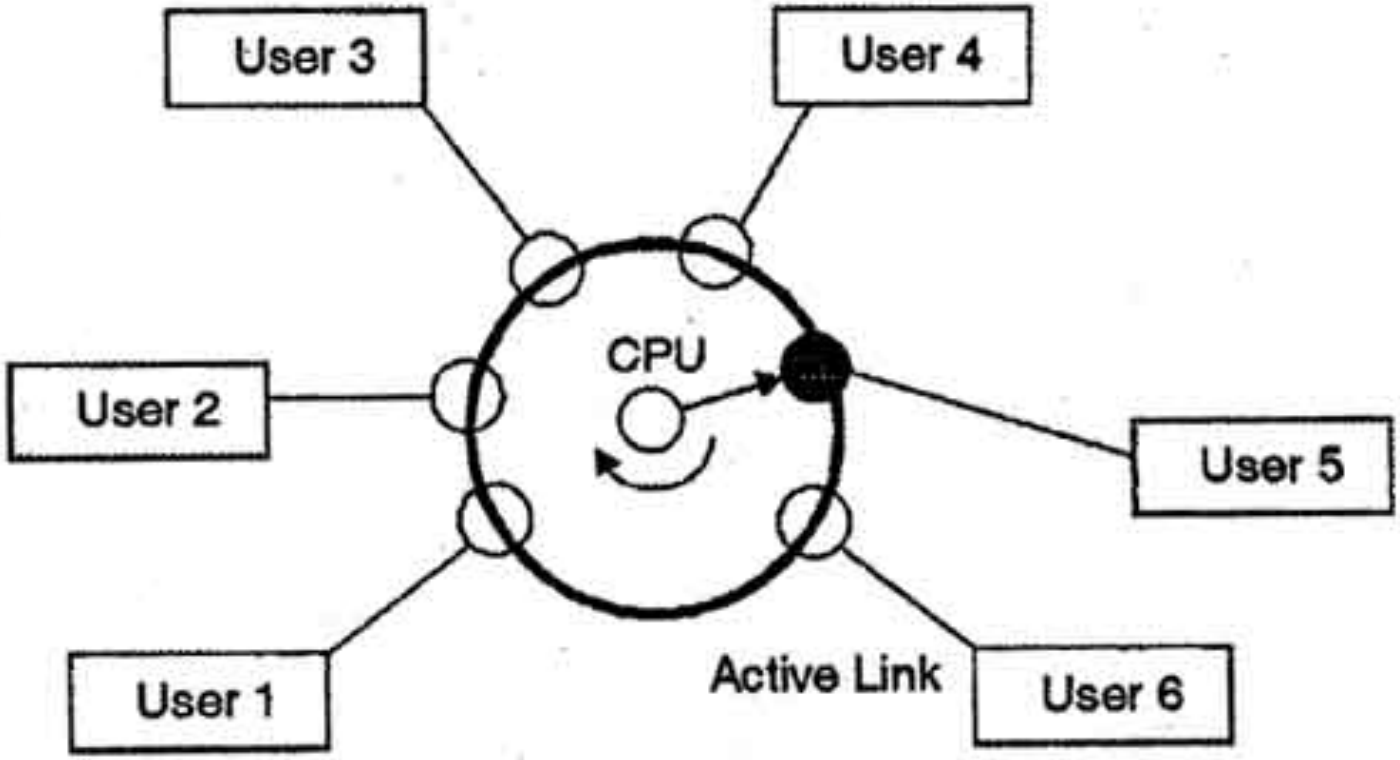


Multiprograming OS



Degree of Multiprogramming:- No of process in main memory

Time Sharing OS



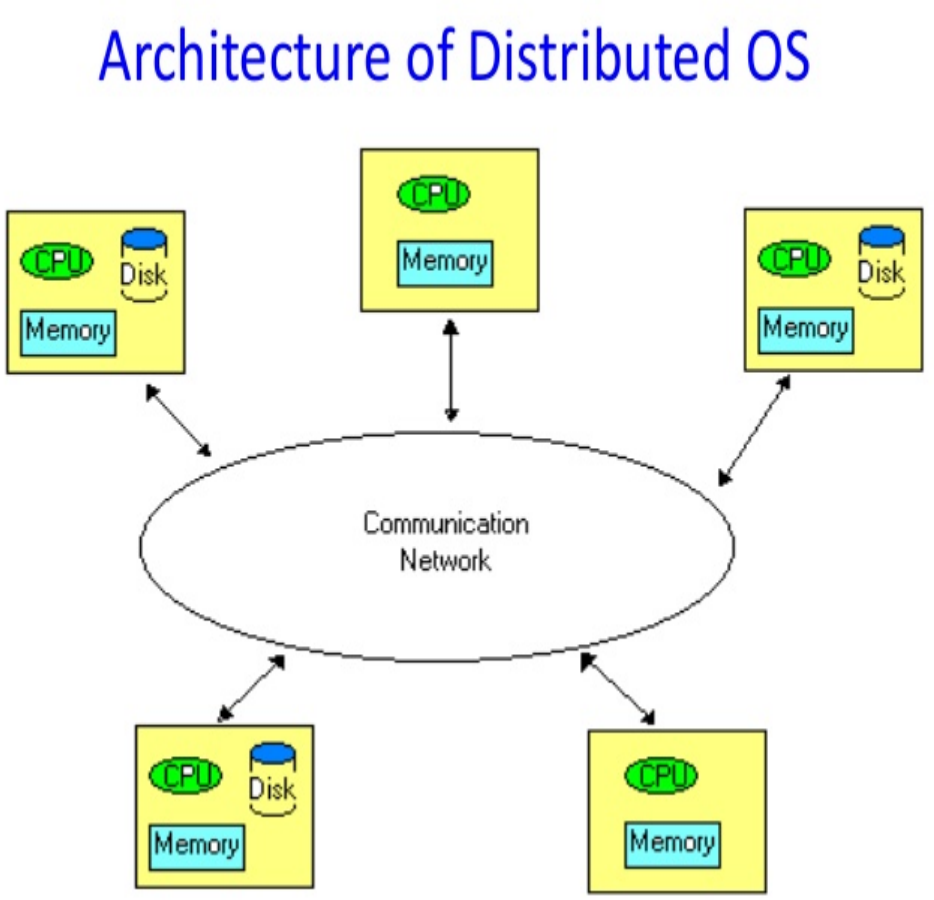
Advantage:-

- Minimum Response Time
- Reduce CPU ideal time
- Avoid Duplication of Software

Distributed OS(Loosely Coupled System)

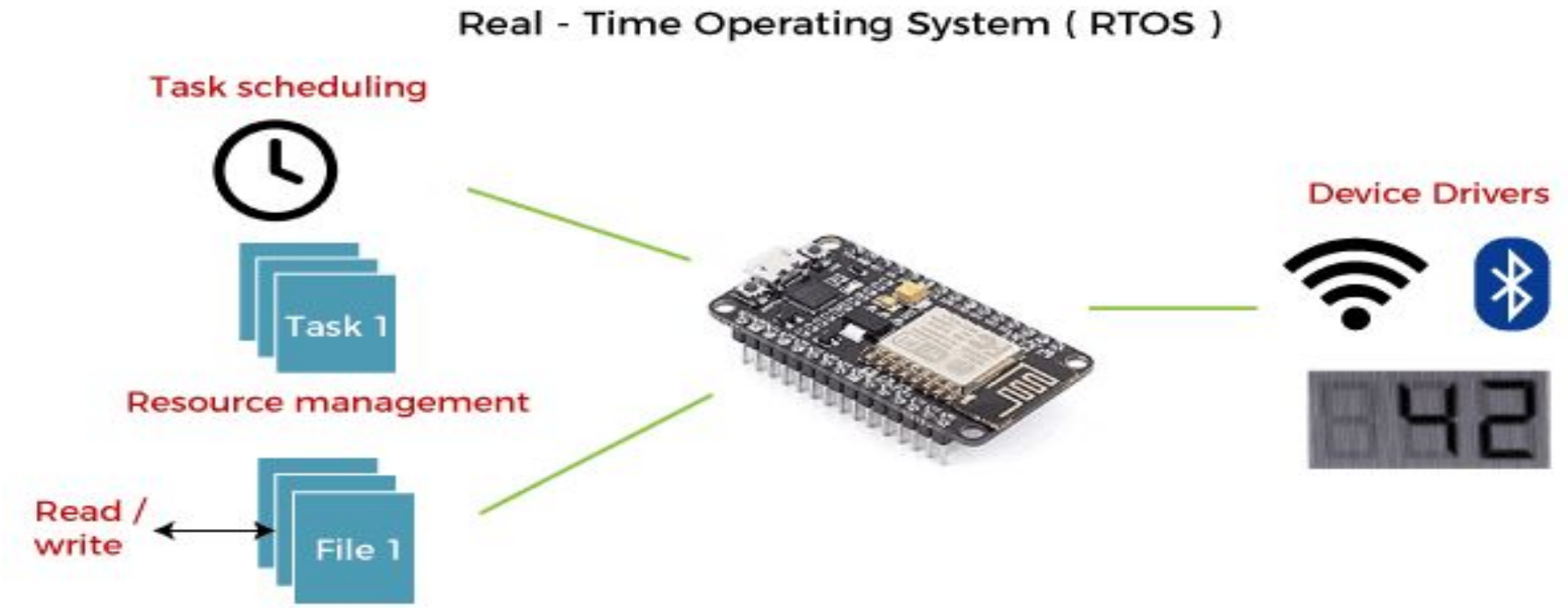
Advantage:-

- Scalability
- Fault Tolerance
- Heterogeneity
- Resource Sharing



Real Time OS

- **It** is a special-purpose operating system used in computers that has strict time constraints for any job to be performed.
- Whenever an event external to the computer occurs, it is communicated to the computer with the help of some sensor used to monitor the event.
- The sensor produces the signal that is interpreted by the operating system as an interrupt.
- On receiving an interrupt, the operating system invokes a specific process or a set of processes to serve the interrupt.
- Response time is fixed



Two type

- Hard Real time
- Soft Real time

Operating System Services

Following are the services provided by an operating system

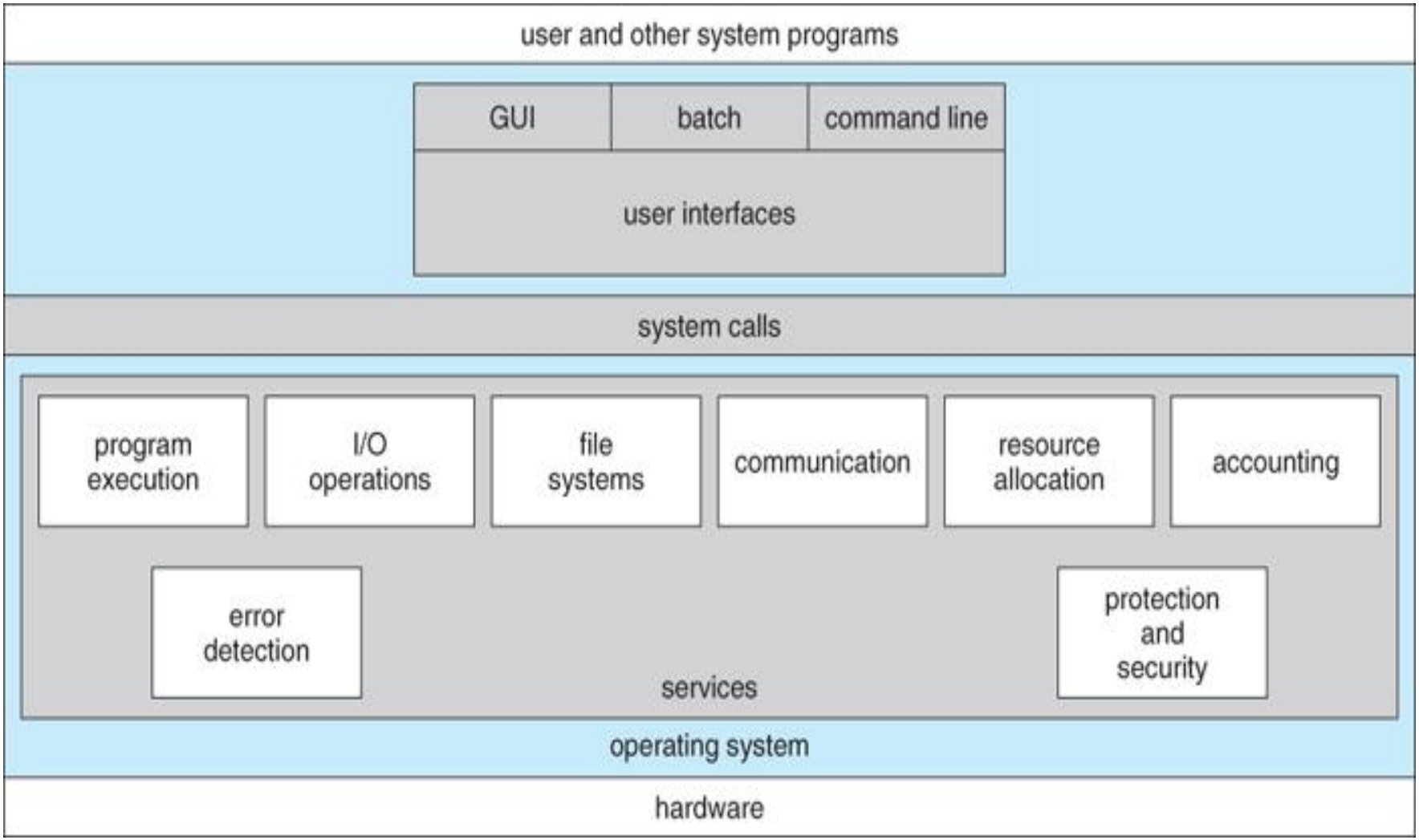
- User Interface
  - CLI
  - Batch
  - GUI
- Program execution
- Control Input/output devices
- Error Detection and Response
  - CPU & Memory
  - I/O Devices
  - User Program
- Accounting
- Security and Protection
- File Management
- Communication

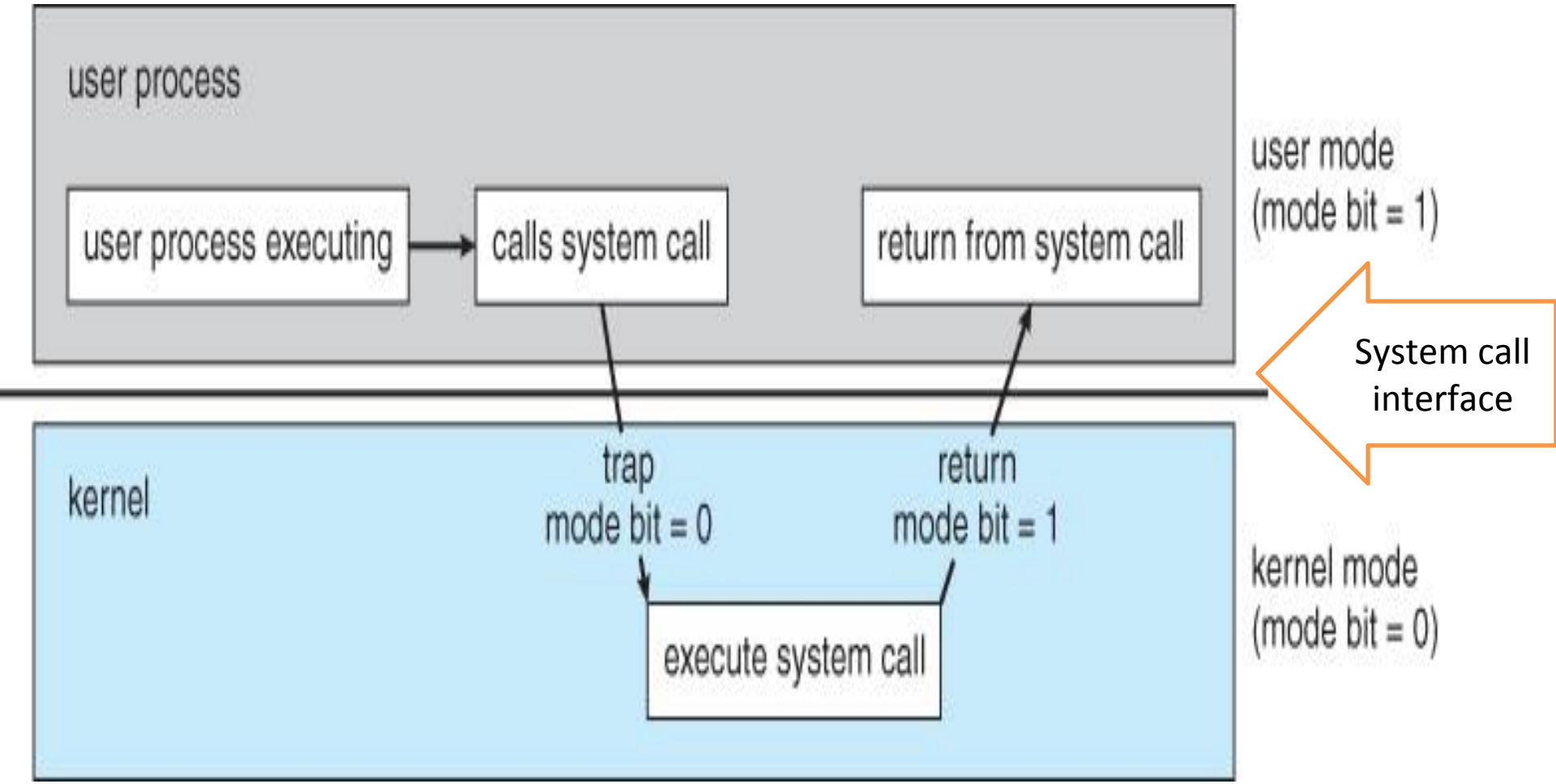
System Call:-

- Interface to Operating system Services
- These calls are generally available as routines written in C and C++, although certain low-level tasks (for example, tasks where hardware must be accessed directly), may need to be written using assembly-language instructions.
- Application programmer write program using application programming interface
- **3 Common API**
  - **Win 32 API**
  - **POSIX API**
  - **JAVA API**
- In background these API invoke system call
- Why use API Instead of System Call

  -Portability

  -Difficult to work with system call


user and other system programs
GUI
batch
command line
user interfaces
system calls
program execution
I/O operations
file systems
communication
resource allocation
accounting
error detection
protection and security
services
operating system
hardware


user process
user process executing
calls system call
return from system call
user mode
(mode bit = 1)
System call
interface
kernel
trap
mode bit = 0
return
mode bit = 1
execute system call
kernel mode
(mode bit = 0)

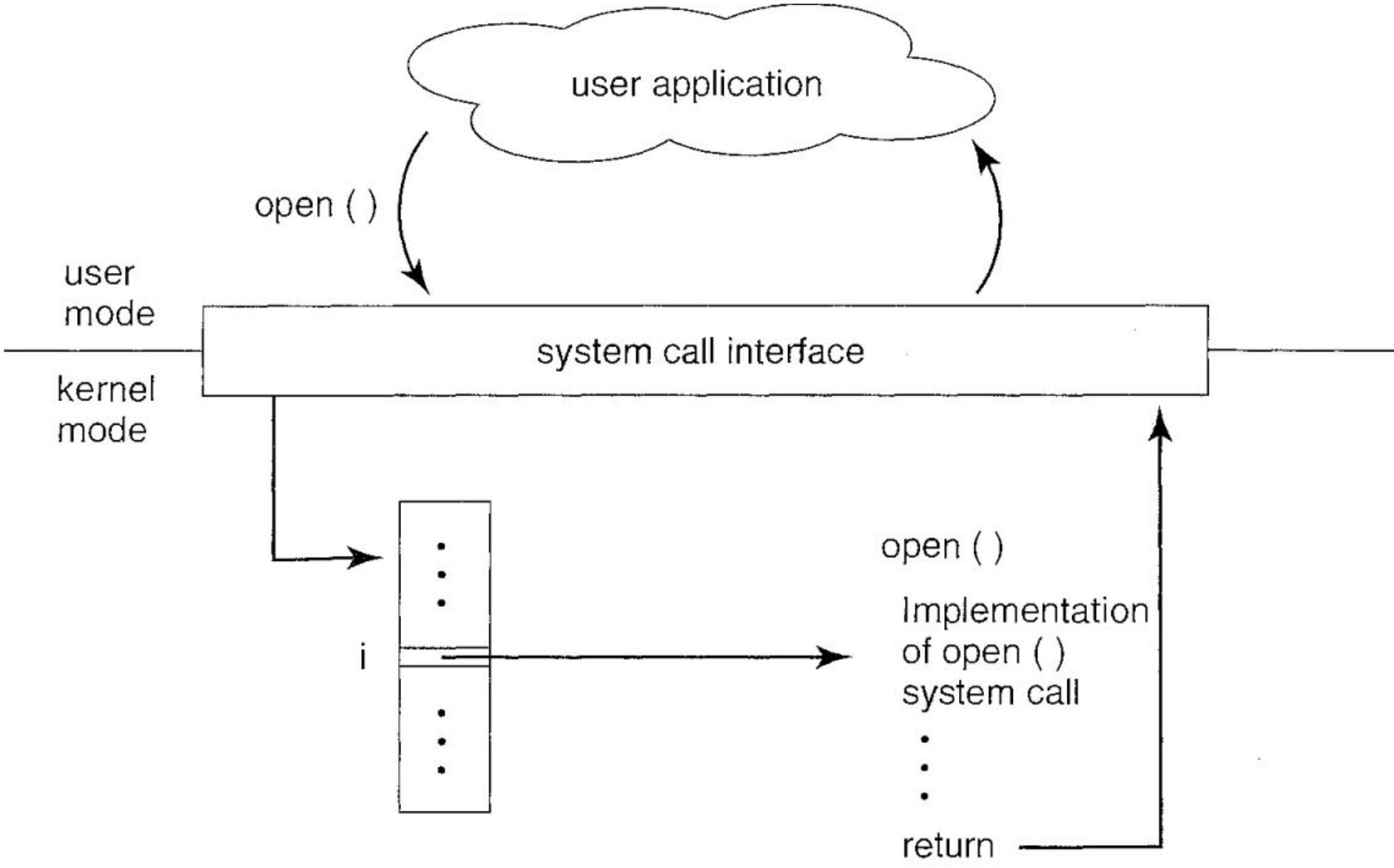
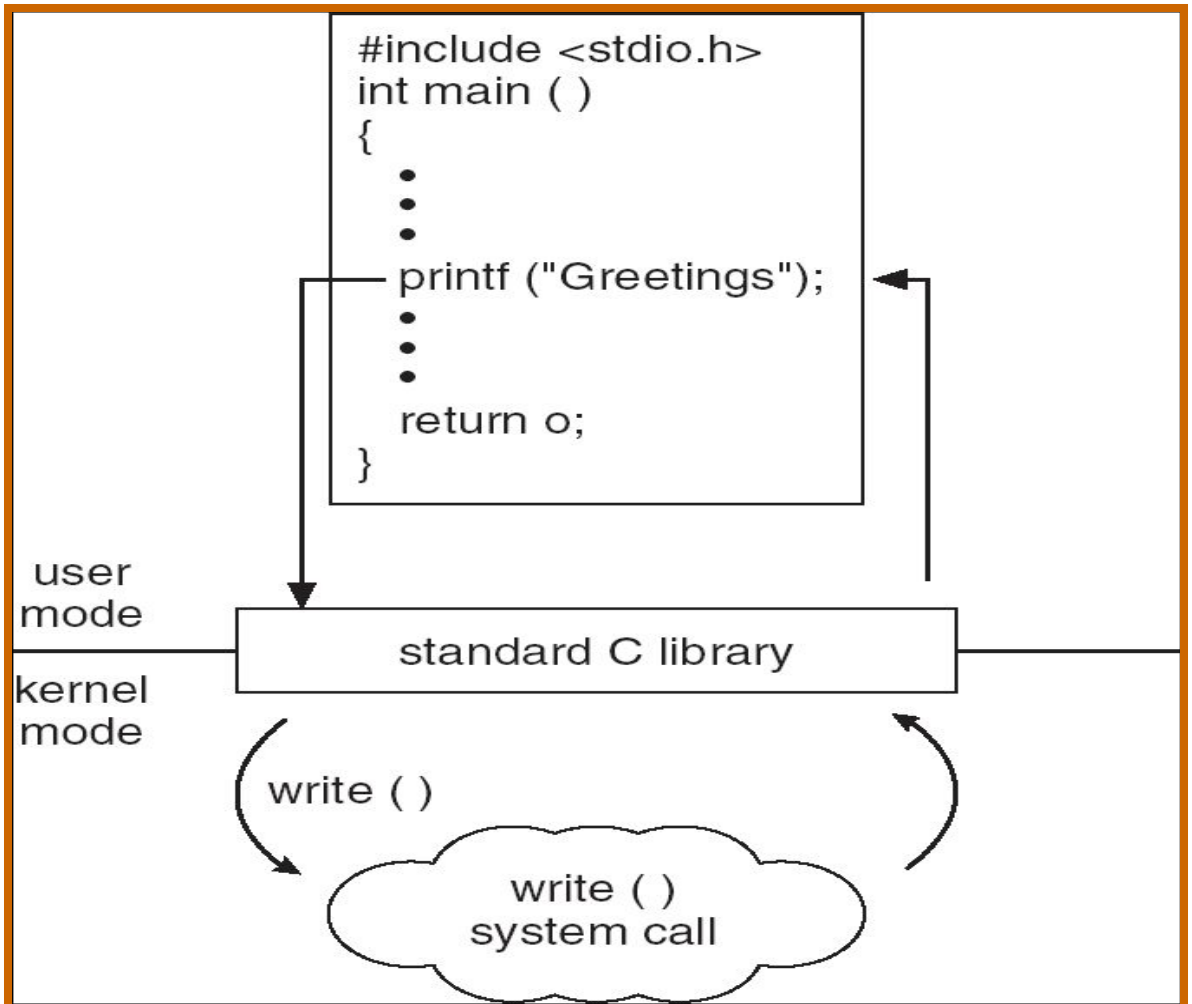


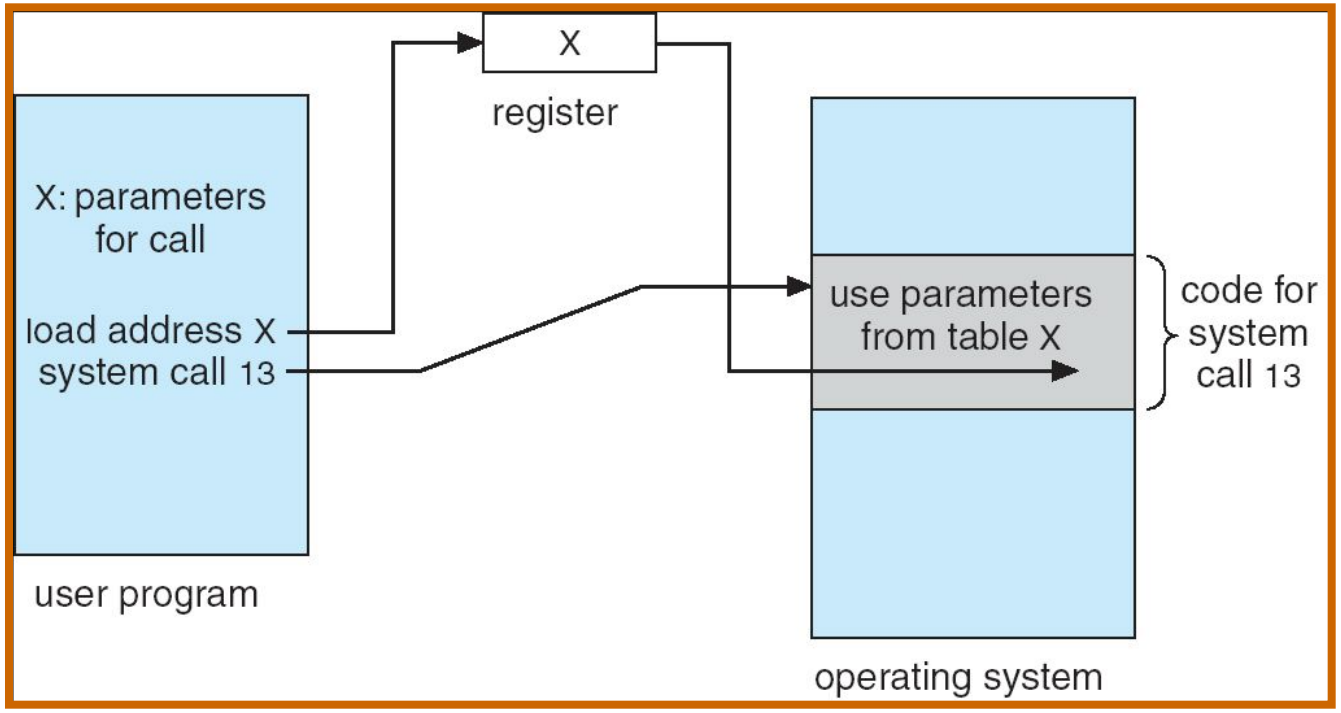
Figure 2.6 The handling of a user application invoking the `open()` system call.

Standard C Library Example

- C program invoking `printf()` library call, which calls the write() system call



Parameter Passing via Table



Five Major Categories

- System calls can be grouped into five major categories:
  - Process control
    - Load, execute, end, abort, create process, get/set process attributes, wait for time/signal, allocate/free memory
  - File management (manipulation)
    - Create/delete/open/close/read/write a file, get/set file attributes
  - Device management
    - Request/release device, read/write data, get/set attributes
  - Information maintenance
    - Get/set time or date, get/set system data, get/set attributes for process/file/device
  - Communications
    - Create/delete connection, send/receive messages, attach/detach devices

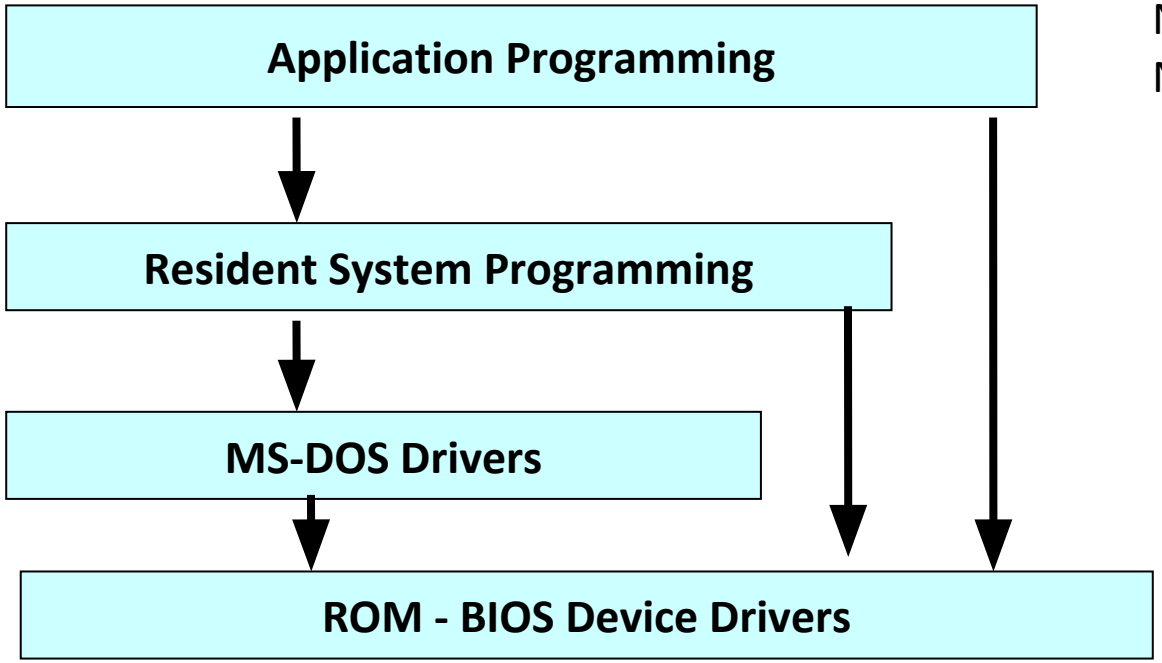
EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

	Windows	Unix
Process Control	`CreateProcess()`	`fork()`
	`ExitProcess()`	`exit()`
	`WaitForSingleObject()`	`wait()`
File Manipulation	`CreateFile()`	`open()`
	`ReadFile()`	`read()`
	`WriteFile()`	`write()`
	`CloseHandle()`	`close()`
Device Manipulation	`SetConsoleMode()`	`ioctl()`
	`ReadConsole()`	`read()`
	`WriteConsole()`	`write()`
Information Maintenance	`GetCurrentProcessID()`	`getpid()`
	`SetTimer()`	`alarm()`
	`Sleep()`	`sleep()`
Communication	`CreatePipe()`	`pipe()`
	`CreateFileMapping()`	`shmget()`
	`MapViewOfFile()`	`mmap()`
Protection	`SetFileSecurity()`	`chmod()`
	`InitlializeSecurityDescriptor()`	`umask()`
	`SetSecurityDescriptorGroup()`	`chown()`

OPERATING SYSTEM STRUCTURES

A SIMPLE STRUCTURE:

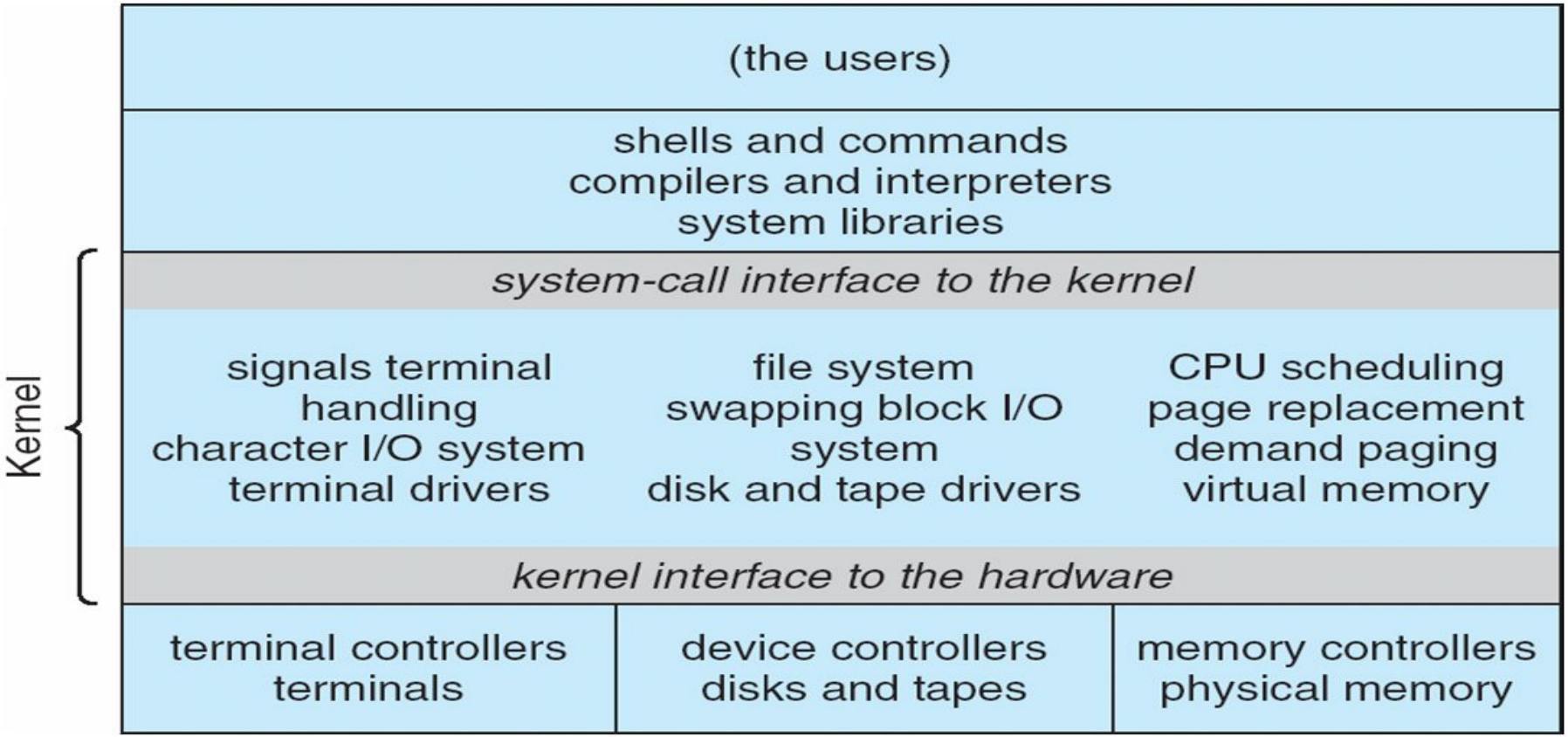
Example of MS-DOS.



Disadvantage:-

Not Well Protected
Not Well structure
Not well define

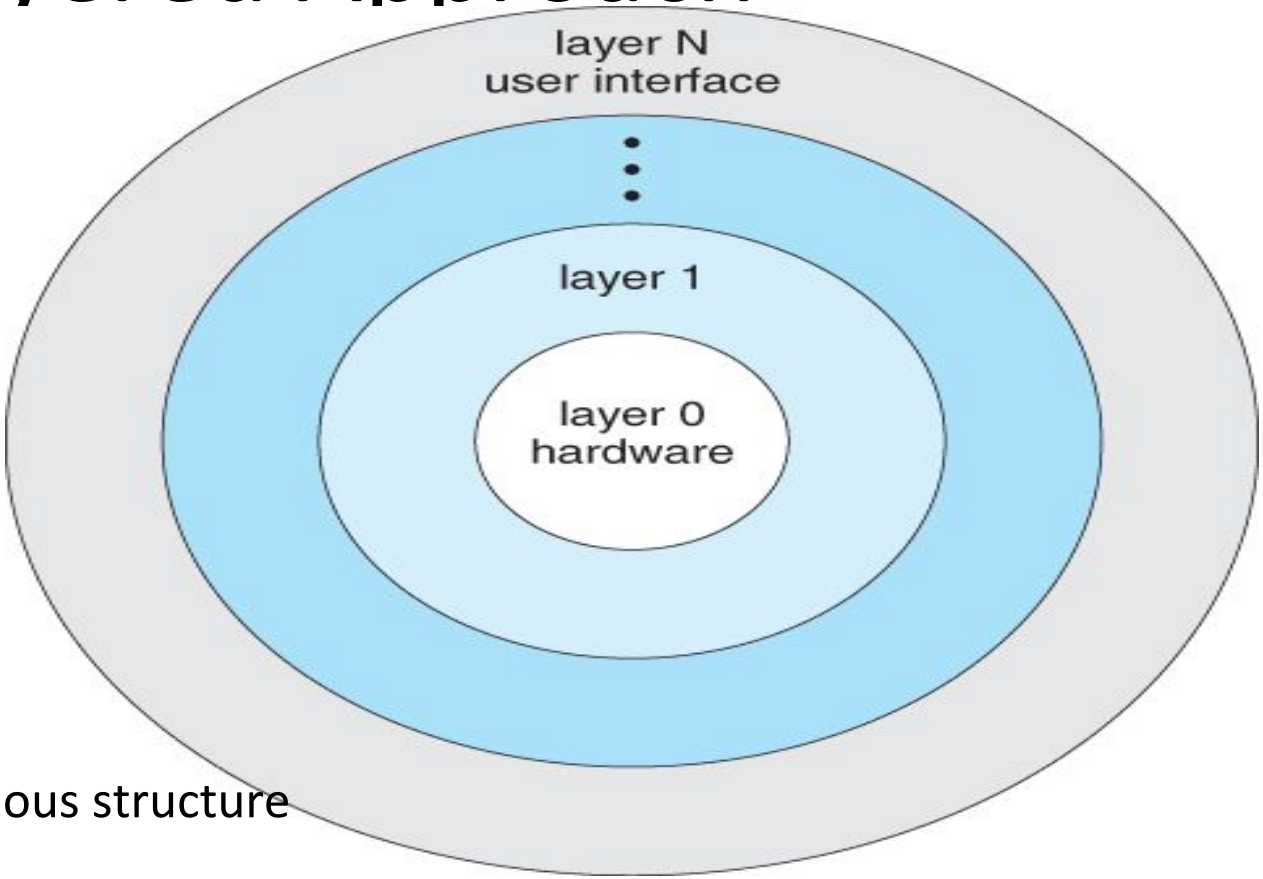
Traditional UNIX System Structure(Monolithic kernel)



Disadvantage:-

As all services at one level so Implementation and maintenance difficult

Layered Approach



Advantage:-

Provide solution to all previous structure

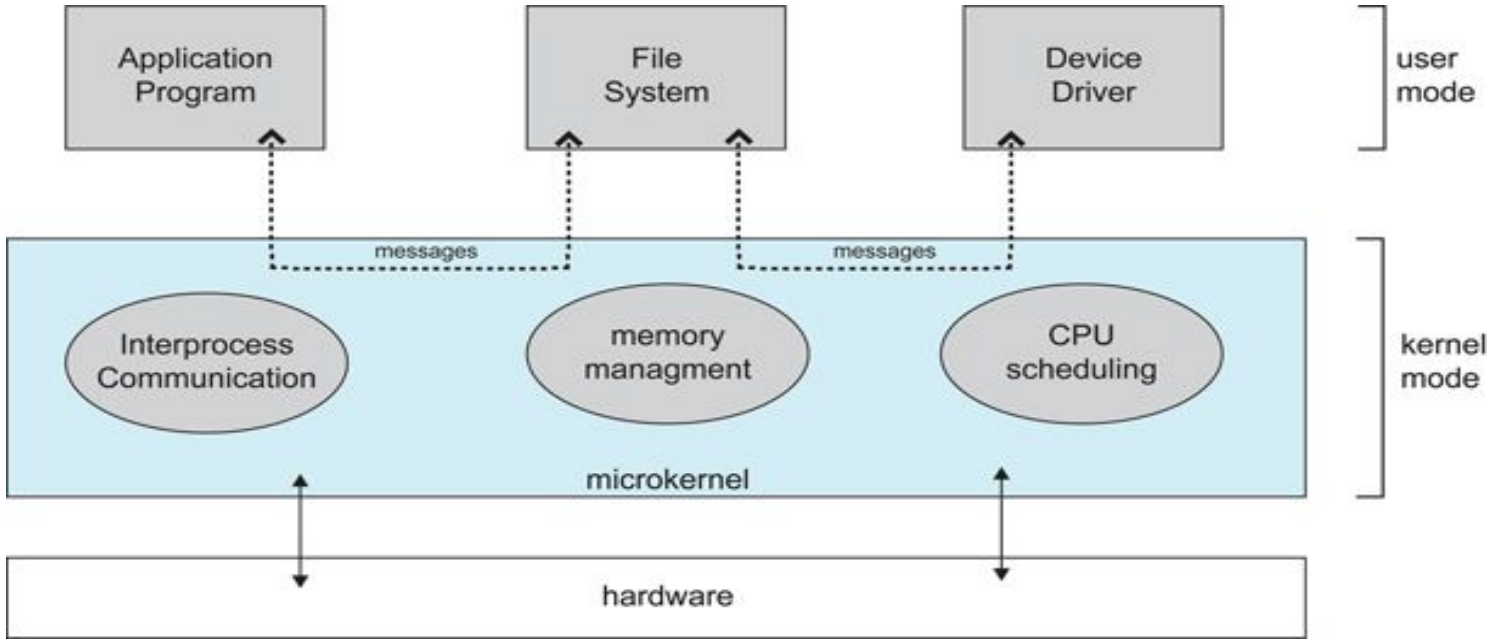
Hardware is protected

Disadvantage:-

Need to carefully decide sequence of layers

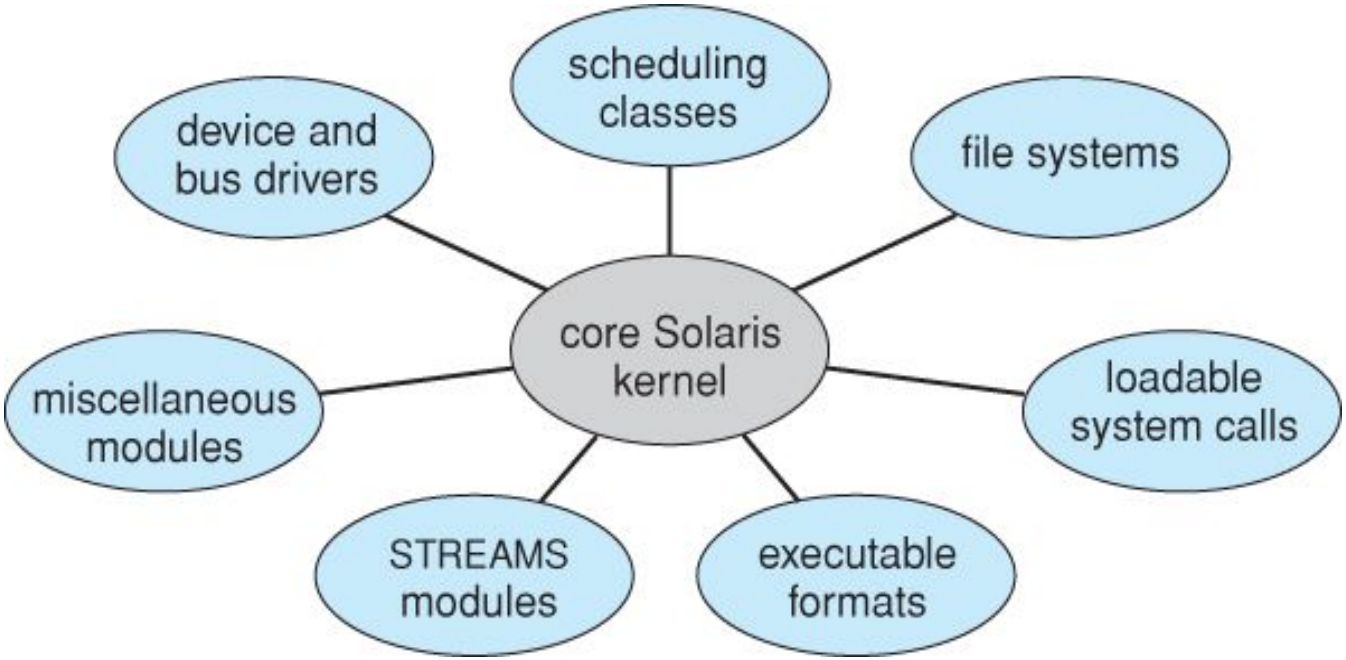
Response time is high

Architecture of a typical microkernel



- **Advantage:-**
- Remove non essential component from kernel that's why microkernel
- Not easily crash the system
- **Disadvantage**
- Maximum overhead

Module



- Best Structure
- Use object oriented concept
- Remove the dis-advantage of all previous structure

Hardware Protection

Hardware Protection

- Dual-Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection

Dual-Mode Operation

- Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.
- Provide hardware support to differentiate between at least two modes of operations.
  1. *User mode* – execution done on behalf of a user.
  2. *Monitor mode* (also *kernel mode* or *system mode*) – execution done on behalf of operating system.

Dual-Mode Operation

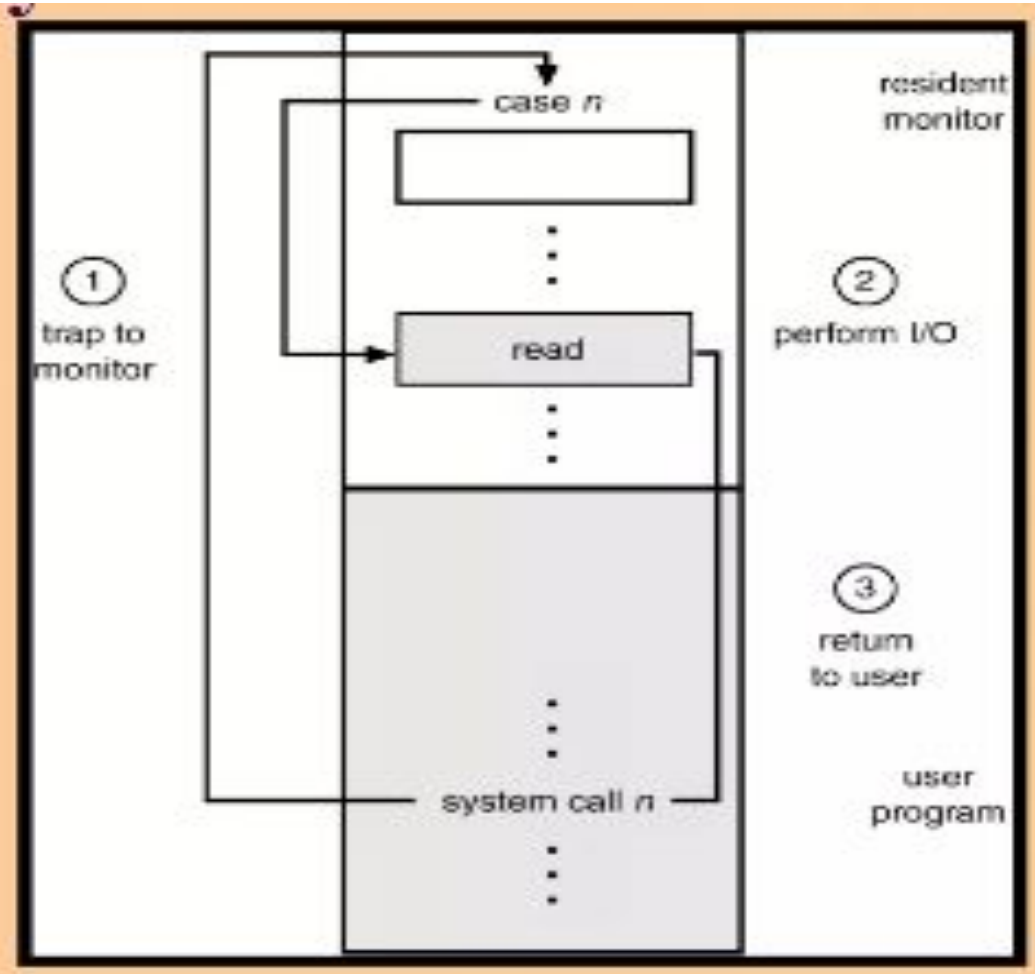
- *Mode bit* added to computer hardware to indicate the current mode: monitor (0) or user (1).
- When an interrupt or fault occurs hardware switches to monitor mode Interrupt/fault

Privileged instructions can be issued only in monitor mode.

I/O Protection

- All I/O instructions are privileged instructions.
- Must ensure that a user program could never gain control of the computer in monitor mode (I.e., a user program that, as part of its execution, stores a new address in the interrupt vector).

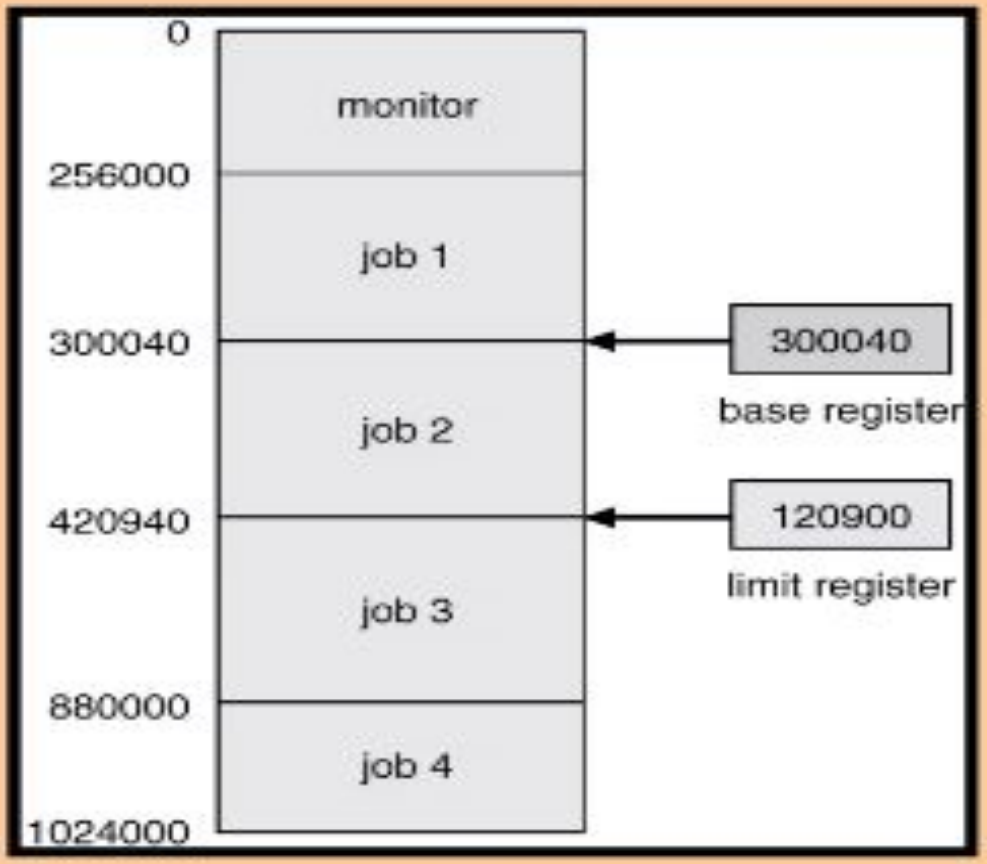
I/O Protection



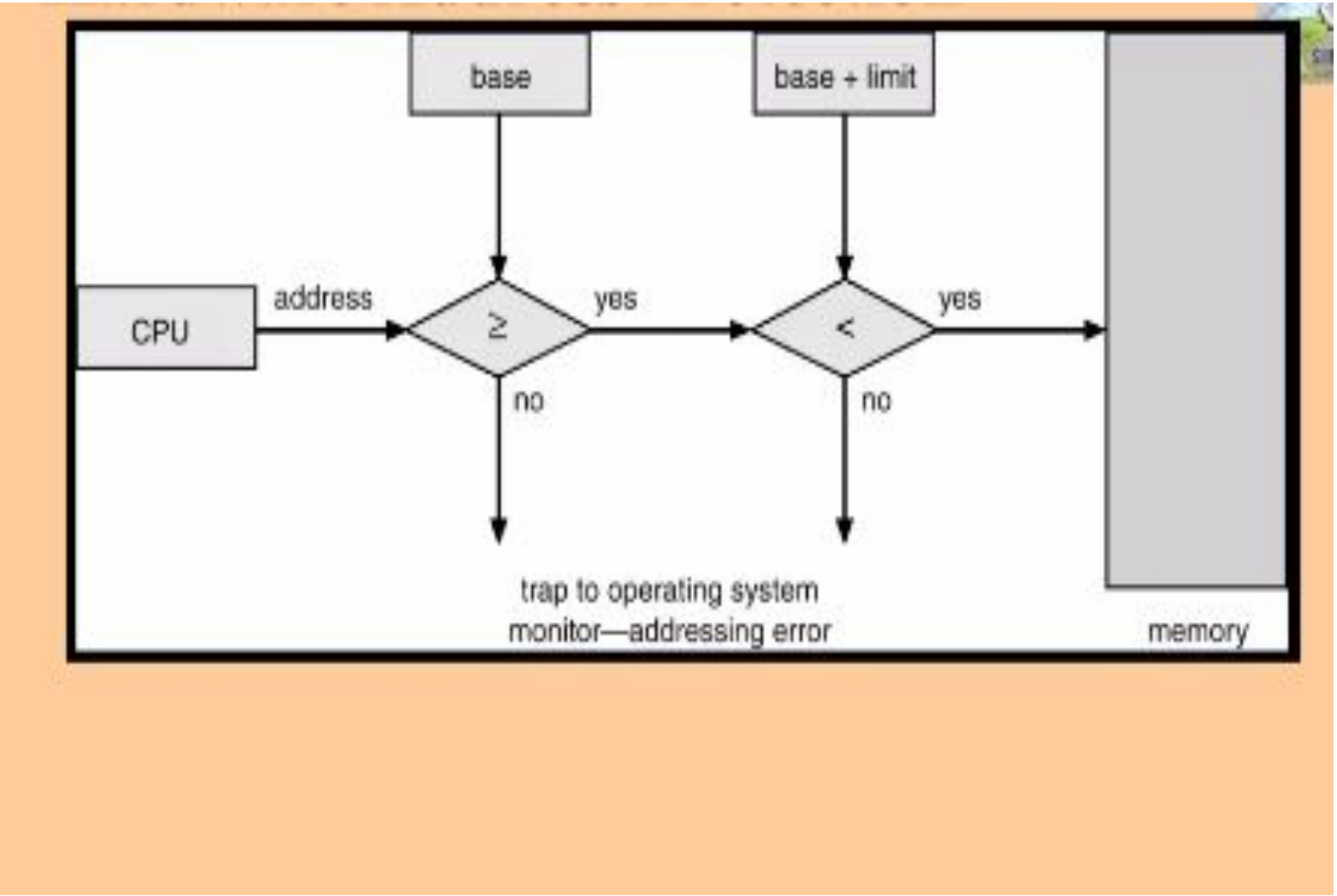
Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
  - **Base register** – holds the smallest legal physical memory address.
  - **Limit register** – contains the size of the range
- Memory outside the defined range is protected.

Memory Protection



Memory Protection



Memory Protection

- When executing in monitor mode, the operating system has unrestricted access to both monitor and user's memory.
- The load instructions for the *base* and *limit* registers are privileged instructions.

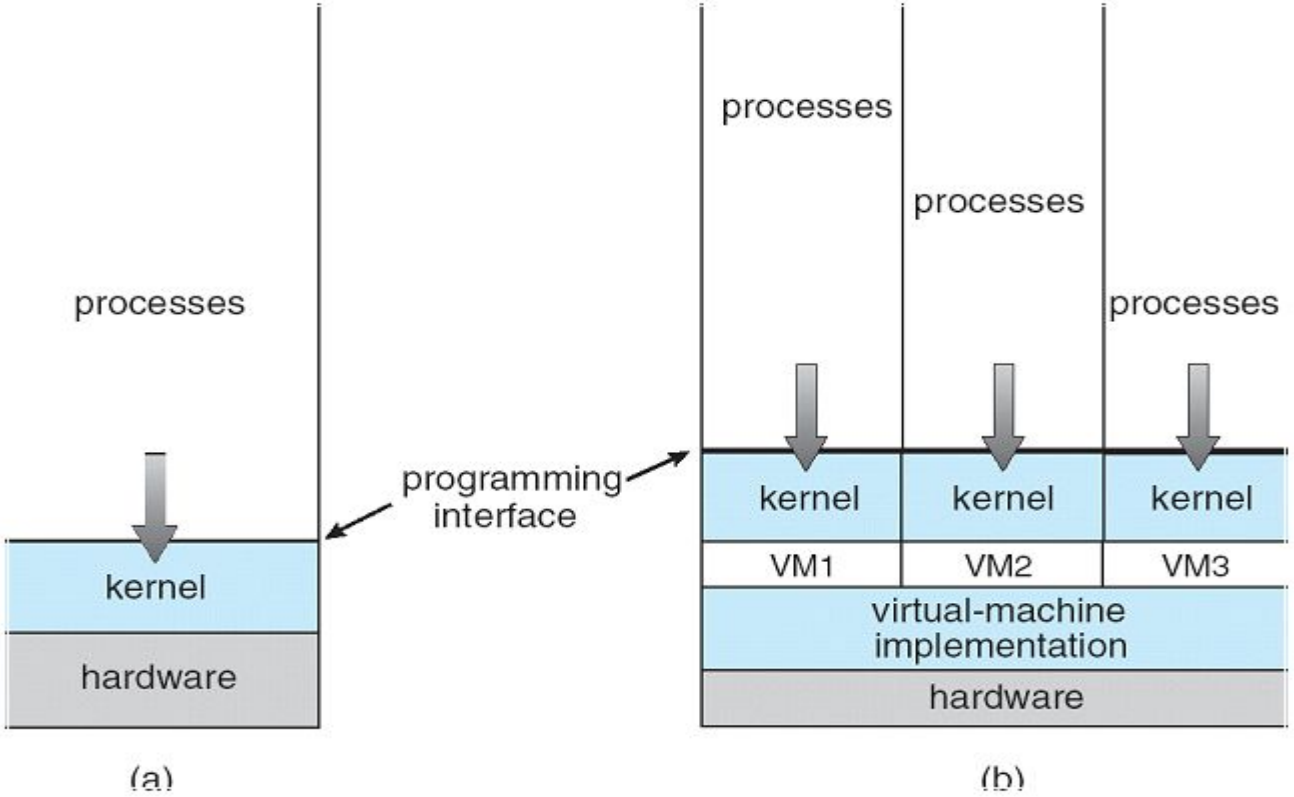
CPU Protection

CPU Protection

- *Timer* – interrupts computer after specified period to ensure operating system maintains control.
  - Timer is decremented every clock tick.
  - When timer reaches the value 0, an interrupt occurs.
- Timer commonly used to implement time sharing.
- Time also used to compute the current time.
- Load-timer is a privileged instruction.

Virtual Machine

- The fundamental idea behind virtual machine is to abstract the hardware of single computer(CPU , Memory ,I/O Devices etc .)into several different execution environment, thereby creating a illusion that each separate execution environment is running its own private computer



- Virtual Machine software- Run in Kernel Mode
- Virtual Machine- Run in User Mode

- Consequently we must have
- A virtual user mode
- A virtual kernel mode

Both of which run on physical user mode

Types of virtual machine :

- System virtual machines - Hardware virtual machine

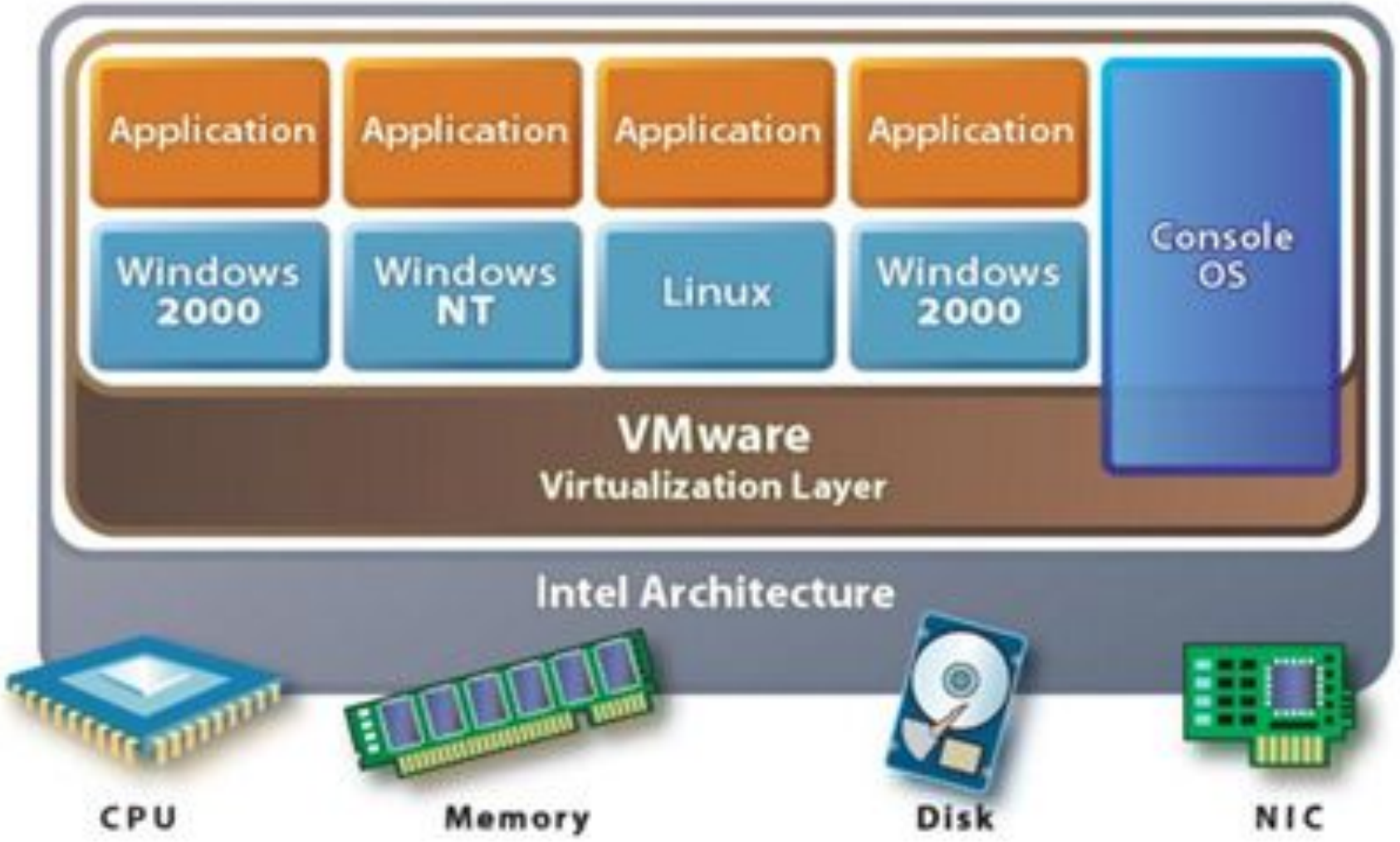
  Provides a complete system platform environment which supports the execution of a complete operating system (OS).

- Process virtual machine - Application virtual machine

  Provides a platform-independent programming environment that abstracts away details of the underlying hardware or operating system from software or application runtime.

- Example:
  - Hardware virtual machine: VMWare, Xen, VirtualBOX .
  - Application virtual machine: Java Virtual Machine, .NET Framework

Vmware Architecture



JVM

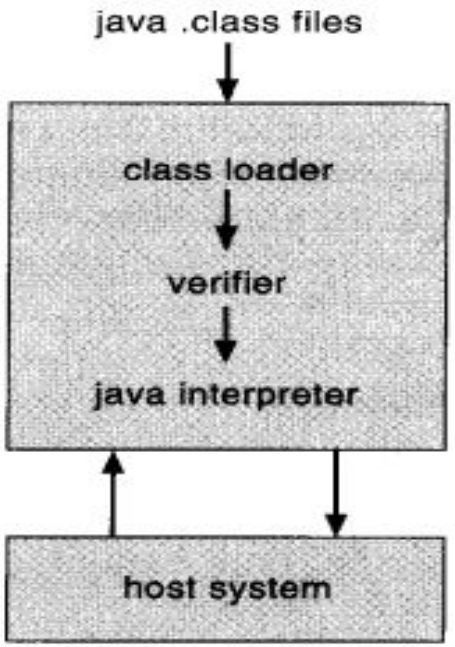


Figure 3.12 The Java virtual machine.

Advantage

- Isolation
- Portable
- Protection