#BTCS2400
Operating System

Unit – 1
Introduction of Operating System

Subject Faculty: Prof. Aanchal Phutela
Computer Science & Engineering
aanchal.phutela@galgotiasuniversity.edu.in

Disclaimer

- ***It is hereby declared that the production of the said content is meant for non-commercial, scholastic and research purposes only.***

- ***We admit that some of the content or the images provided in this channel's videos may be obtained through the routine Google image searches and few of them may be under copyright protection. Such usage is completely inadvertent.***

- ***It is quite possible that we overlooked to give full scholarly credit to the Copyright Owners. We believe that the non-commercial, only-for-educational use of the material may allow the video in question fall under fair use of such content. However we honor the copyright holder's rights and the video shall be deleted from our channel in case of any such claim received by us or reported to us.***

Reference Books

1. Operating Systems: Internals & Design Principles, 9th Edition, William Stallings, Pearson Education India
2. Operating System Concepts, 9th edition Peter B. Galvin, Greg Gagne, Abraham Silberschatz, John Wiley & Sons, Inc.
3. Modern Operating Systems-By Andrew S. Tanenbaum (PHI)

Topics to be covered

- Computer system overview and architecture
- Definition Operating System (OS)
- Objectives / Goals of Operating System (OS)
- Operating Systems (OS) services
- Generations of Operating Systems (OS)
- System calls
- Views of Operating Systems
- Types of Operating Systems (OS)
- Multiprogramming v/s Multiprocessing v/s Multitasking
- Structures of OS

Basic elements of computer

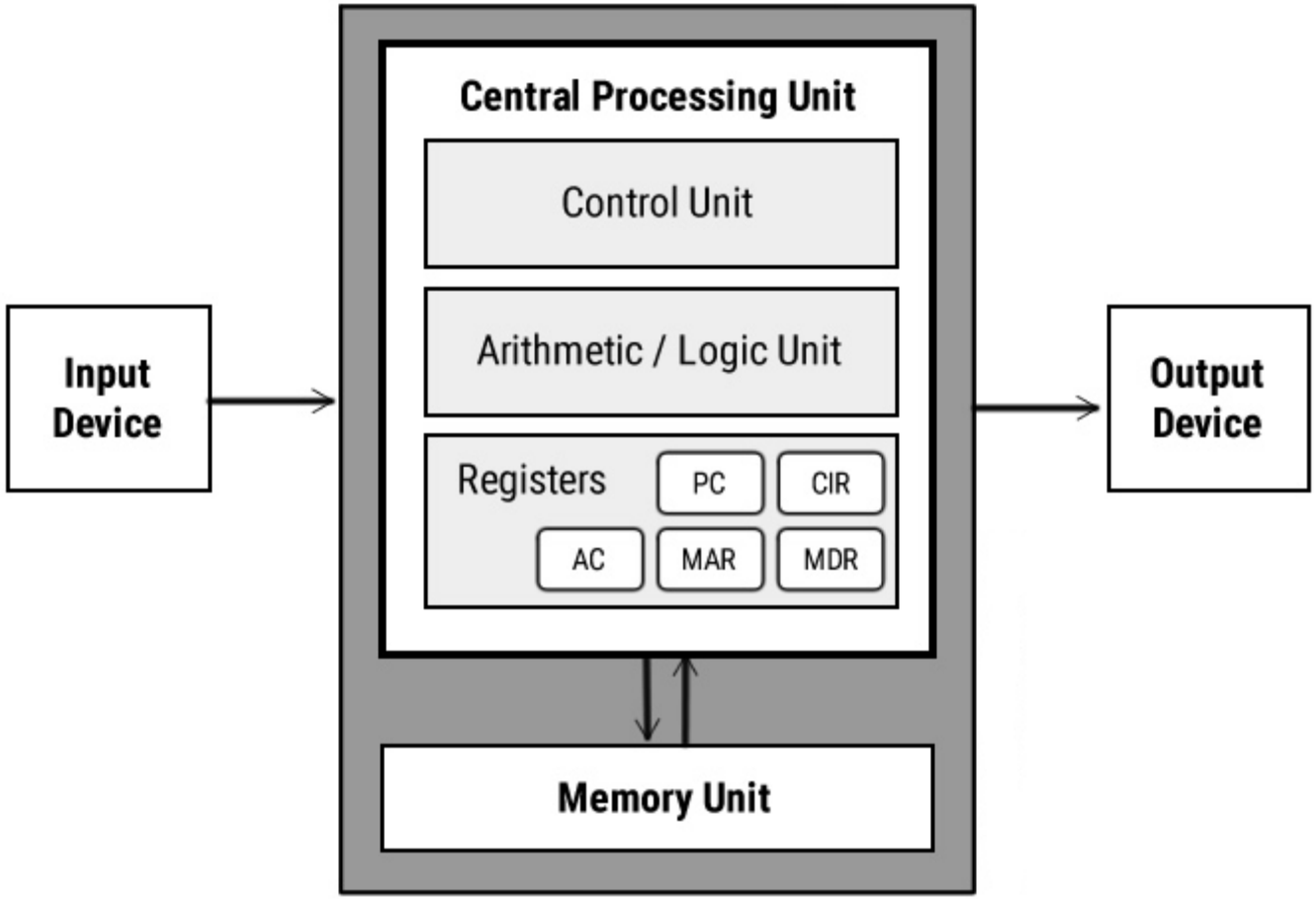
Processor

I/O Modules

Main Memory

System Bus

Computer system architecture



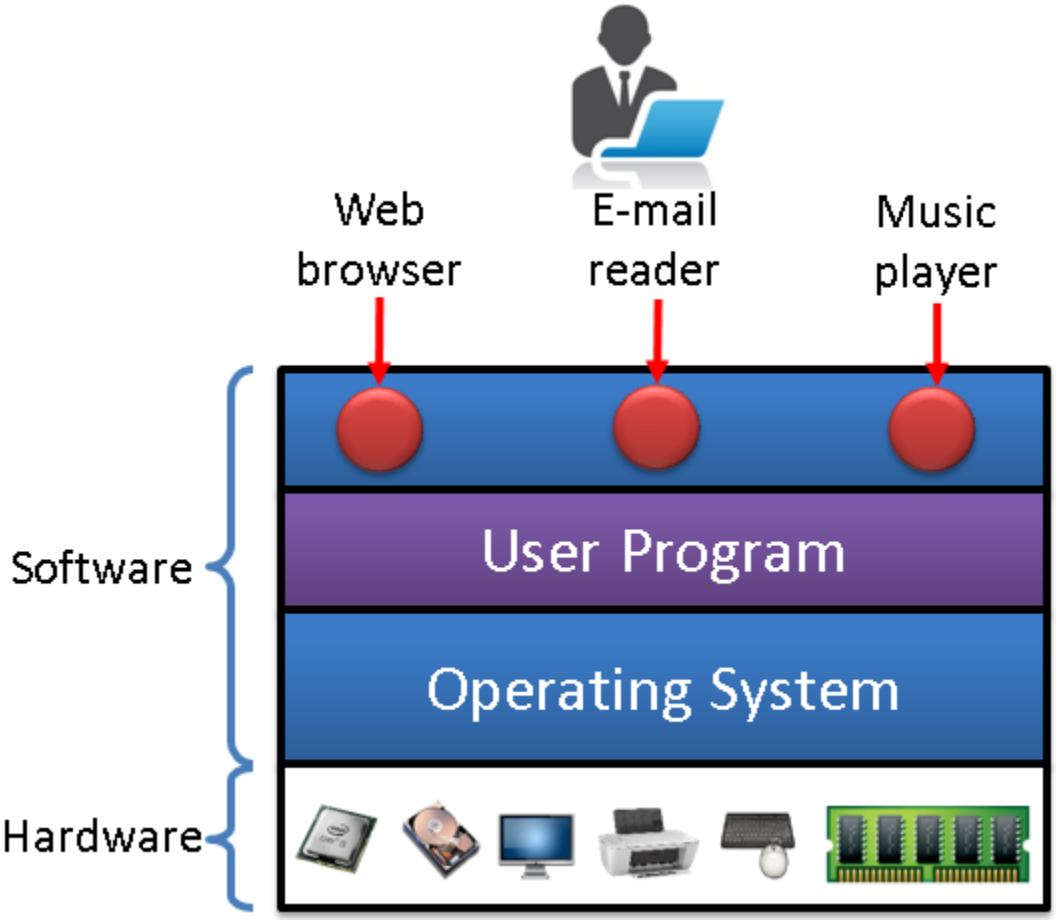
What is Operating System (OS)?

An Operating System is a program that acts as an intermediate/interface between a user of a computer and the computer hardware.

Definition

- It manages hardware resources and provides various service to users
- It provides a user-friendly environment in which a user may *easily* develop and execute programs.
- Otherwise, hardware knowledge would be mandatory for computer programming.
- So, it can be said that an OS *hides the complexity* of hardware from uninterested users.

Where does the OS fit in?



- OS **lies between hardware and user program**.
- It **acts as an intermediary** between the user and the hardware.

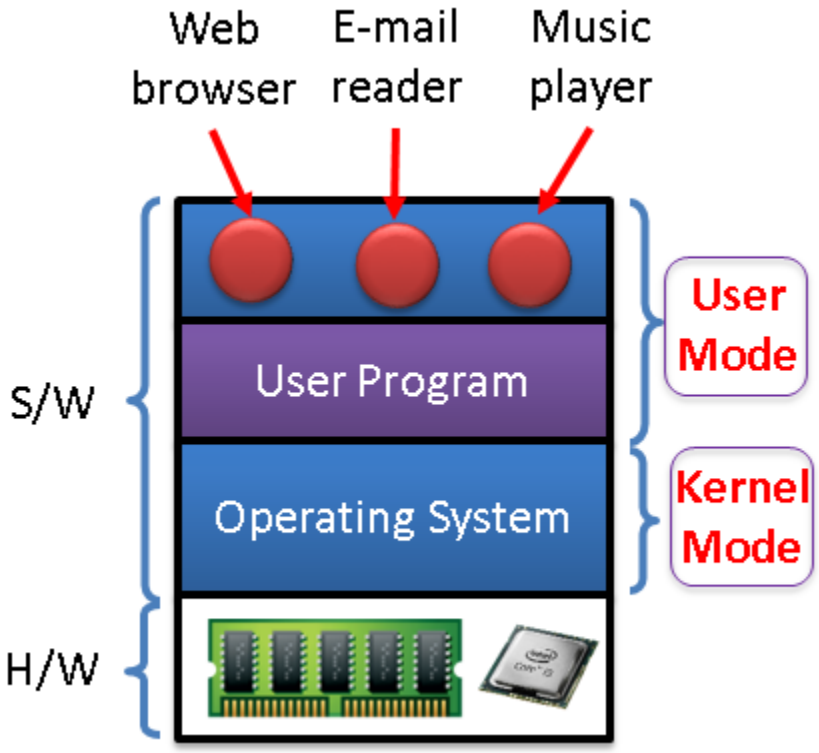
Modes of operation of computer

1. Kernel Mode

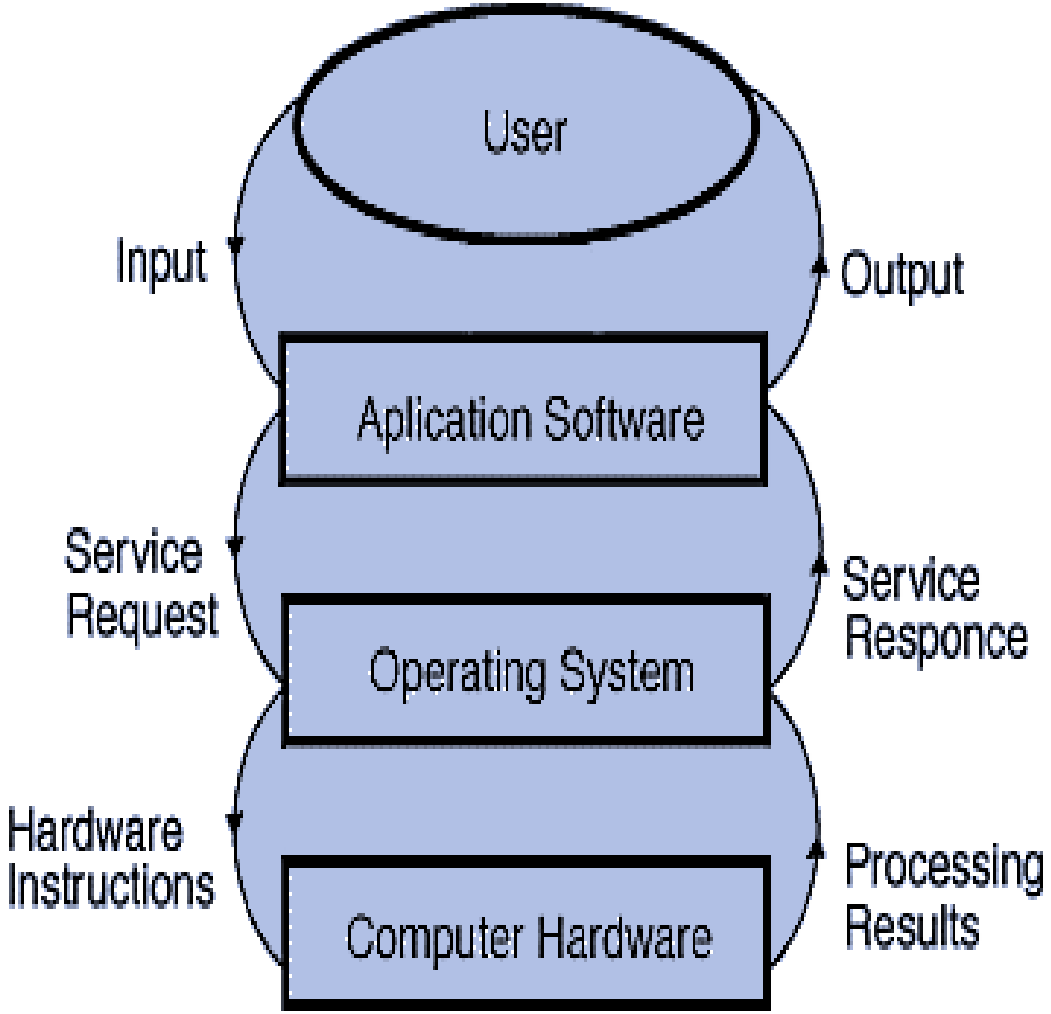
- has complete access to all the hardware
- here OS have complete access to all the hardware and can execute any instruction on that a machine is capable of executing.
- has high privileged (rights)

2. User Mode

- can execute only subset (few) of the machine instructions
- has less privileged (rights)

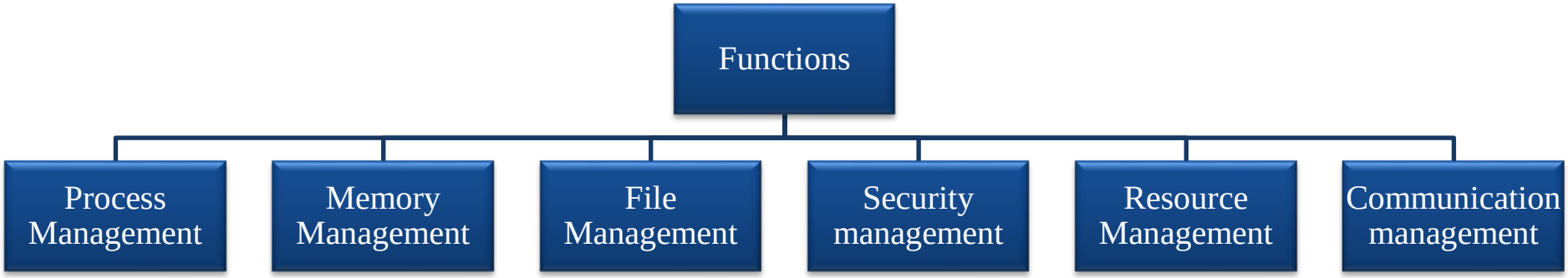


Flow of Communication



Functions/Services/Tasks of OS

- An Operating System (OS) is a collection of software that
  - manages hardware resources
  - provides various service to the users



Functions/Services/Tasks of OS

1. Process Management

- *By process management OS manages many kinds of activities:*
  - All process from start to shut down i.e. open, save, copy, install, print.
  - Creation and deletion of user and system processes.

Functions/Services/Tasks of OS

2. Memory Management

- *The major activities of an operating regard to memory-management are:*
  - Decide which process are loaded into memory when memory space becomes available.
  - Allocate and deallocate memory space as needed.

Functions/Services/Tasks of OS

3. File Management

- *The file management system allows the user to perform such tasks:*
  - Creating files and directories
  - Renaming files
  - Coping and moving files
  - Deleting files

Functions/Services/Tasks of OS

4. Security Management

- *By security management OS manages many tasks such as:-*
  - Alert messages
  - Virus protection
  - Dialogue boxes
  - Firewall
  - Passwords

Functions/Services/Tasks of OS

5.Resource Management

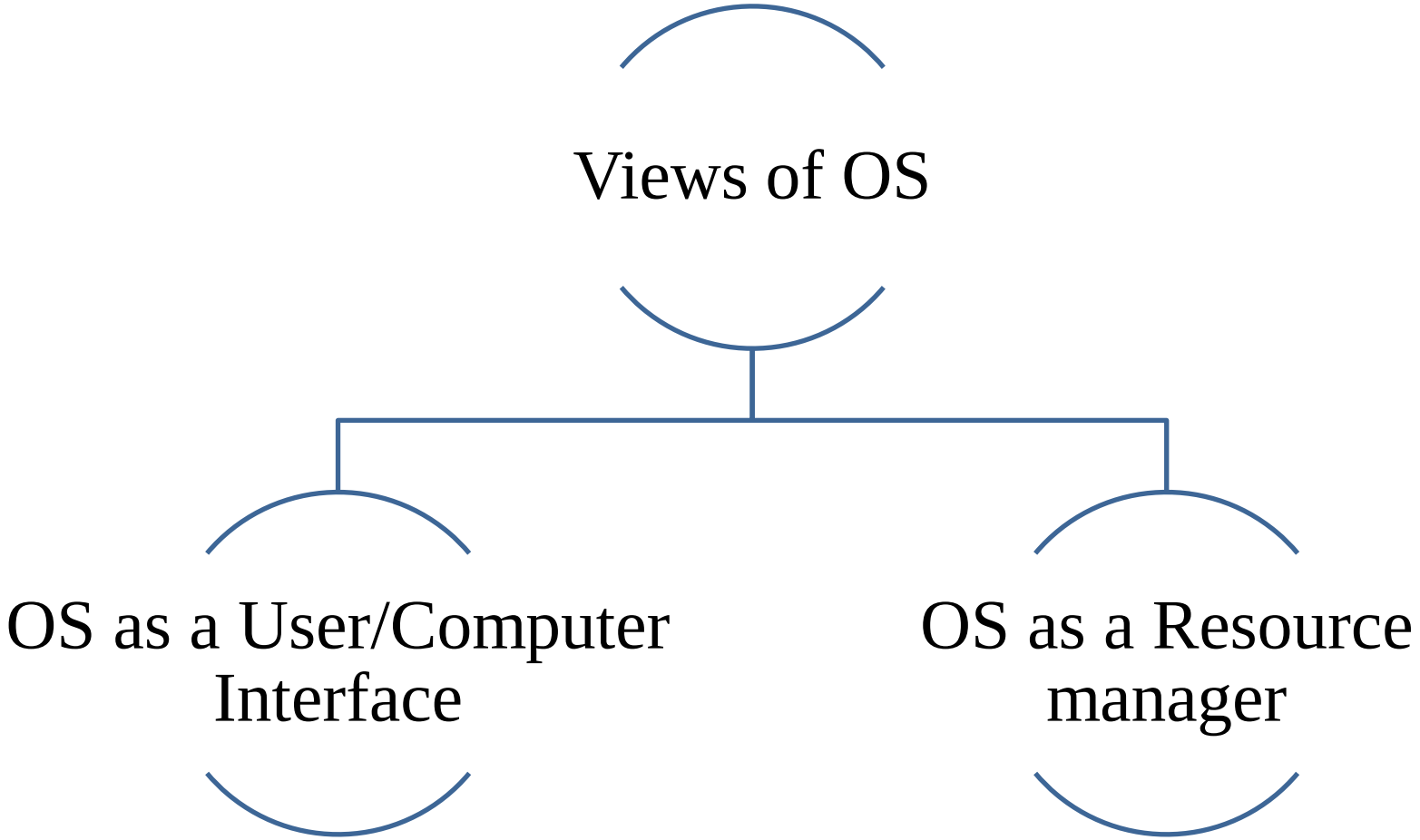
- *To manage Resources, OS perform many tasks such as:*
  - Install drivers required for input and output, memory, power.
  - Coordination among peripherals.

Functions/Services/Tasks of OS

6. Communication Management

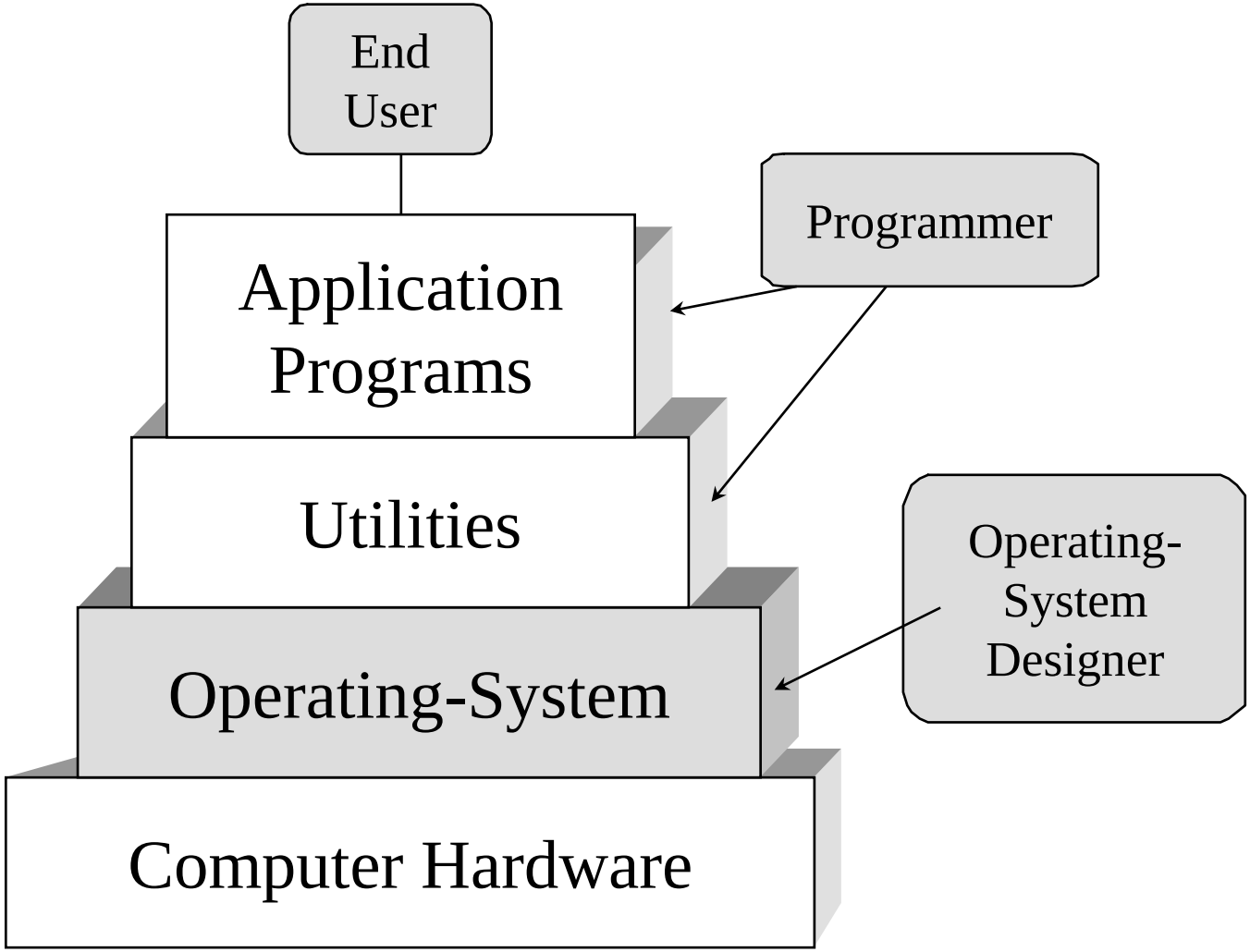
- *For proper coordination OS performs communication management:*
  - Communication between *user-application software-hardware.*
  - One computer to another via LAN or WAN.

Different views of OS



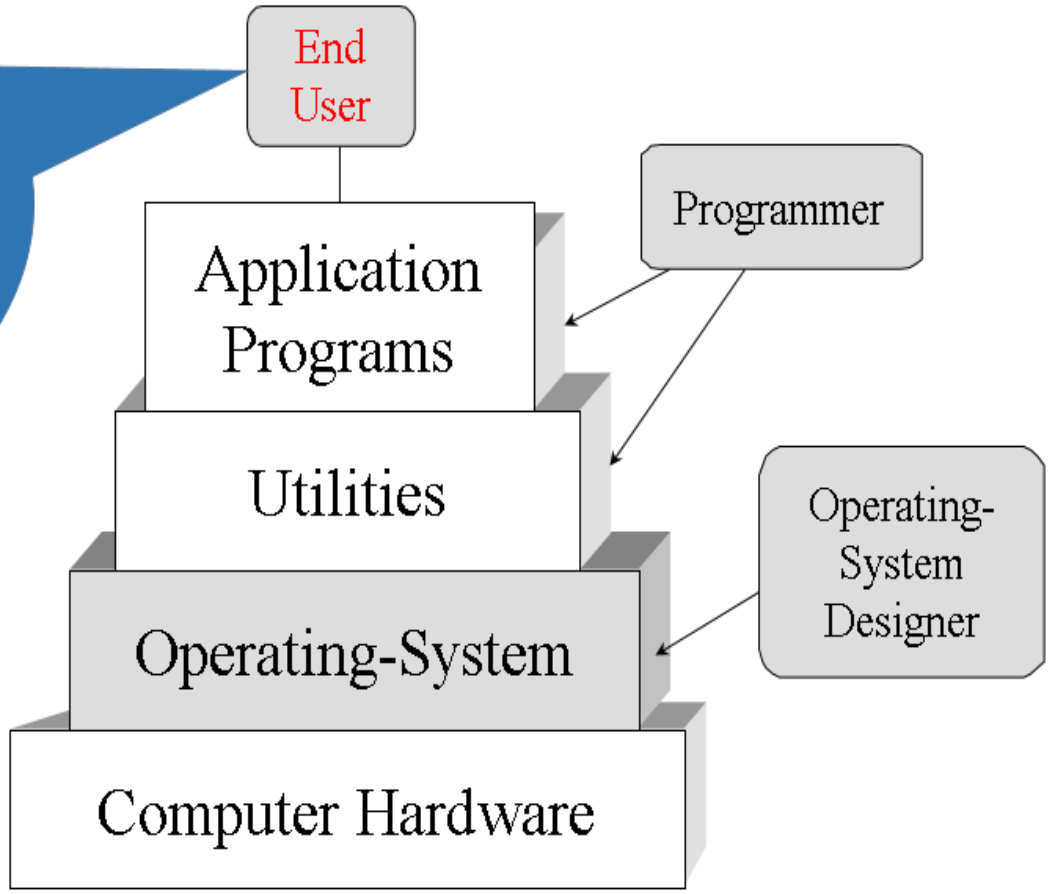
Different views of OS

- OS as a User/Computer Interface:

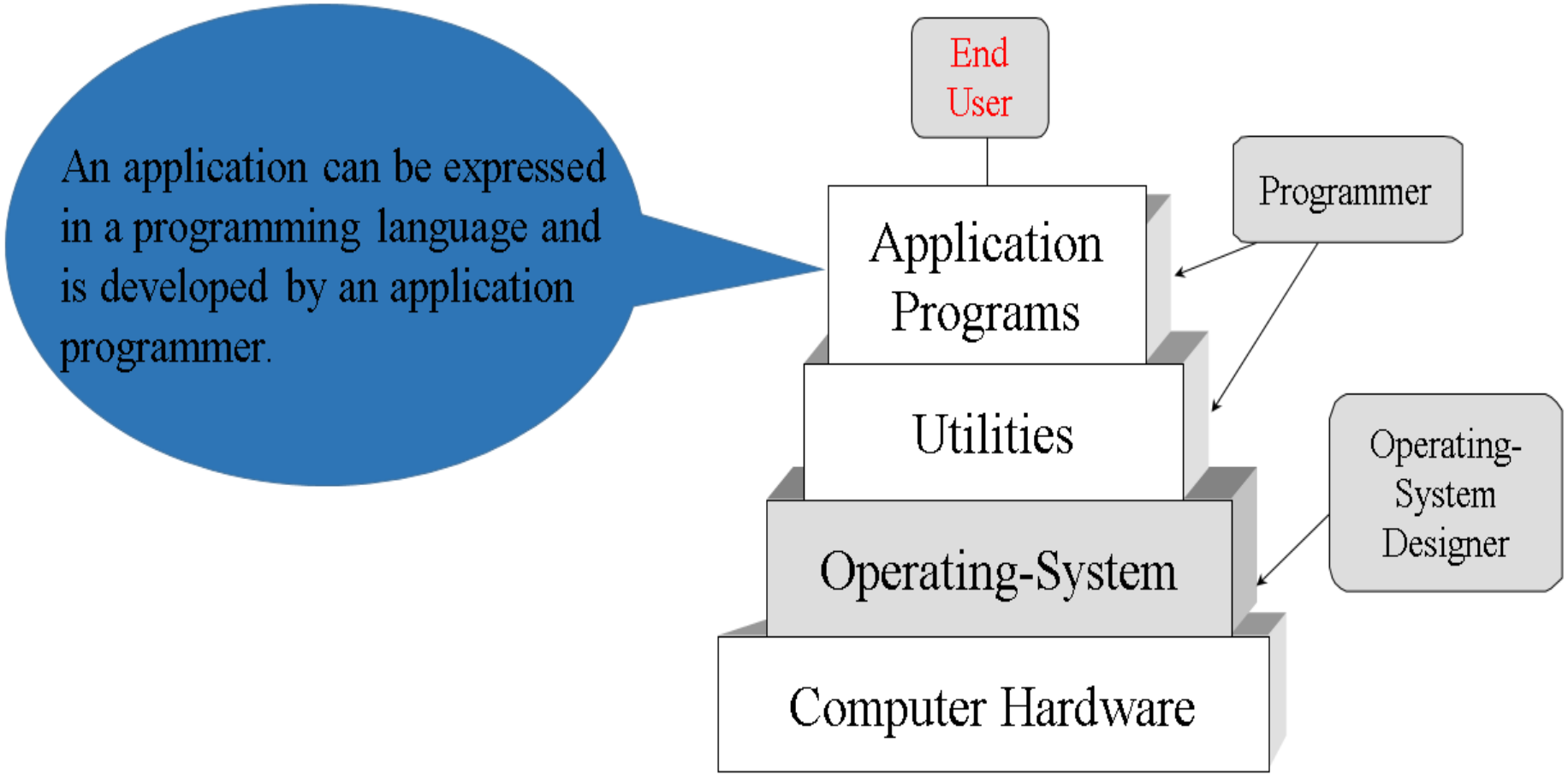


Different views of OS

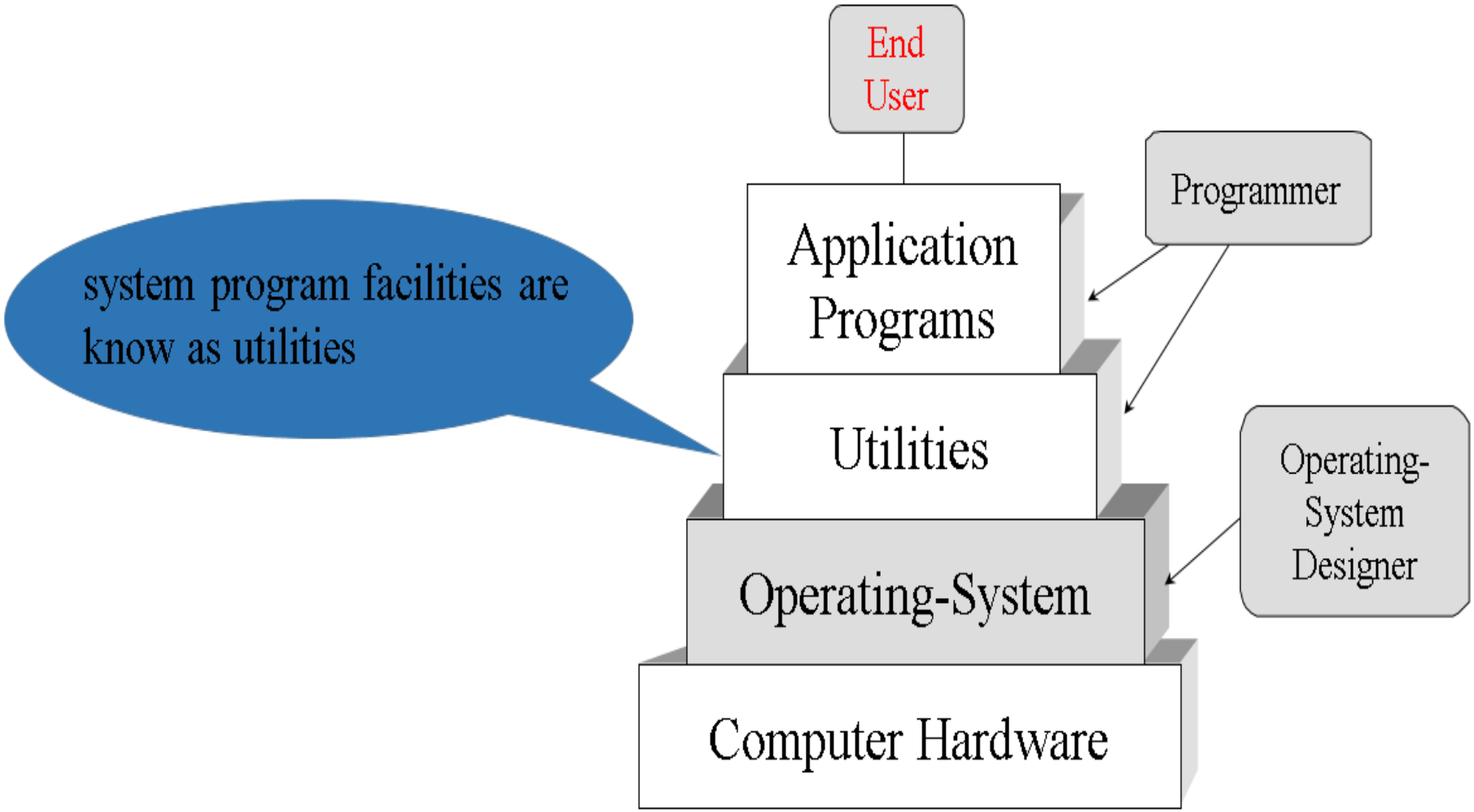
End user only see the computer system in terms of set of application, he/she is not concern with the details of the computer hardware.



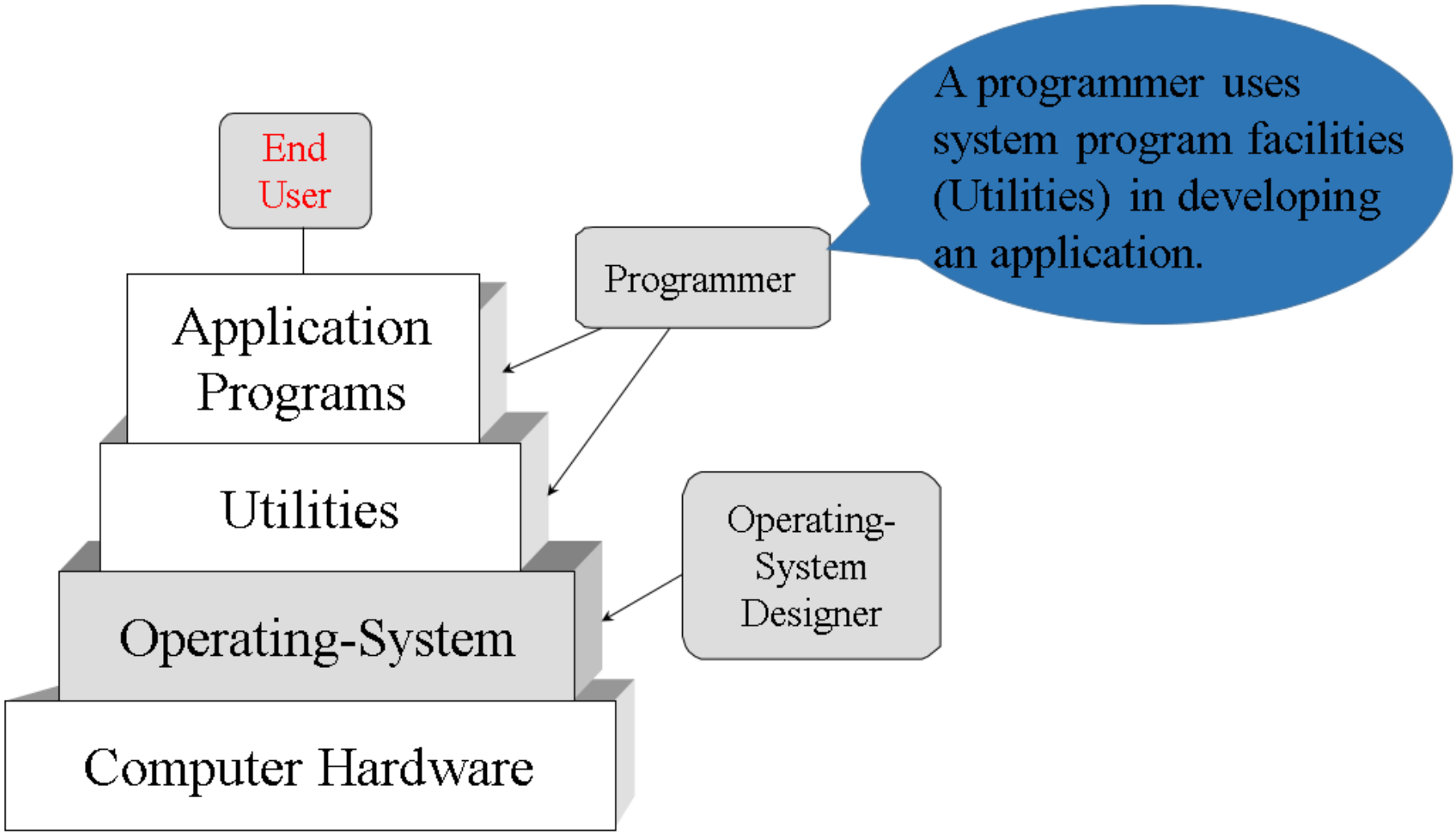
Different views of OS



Different views of OS

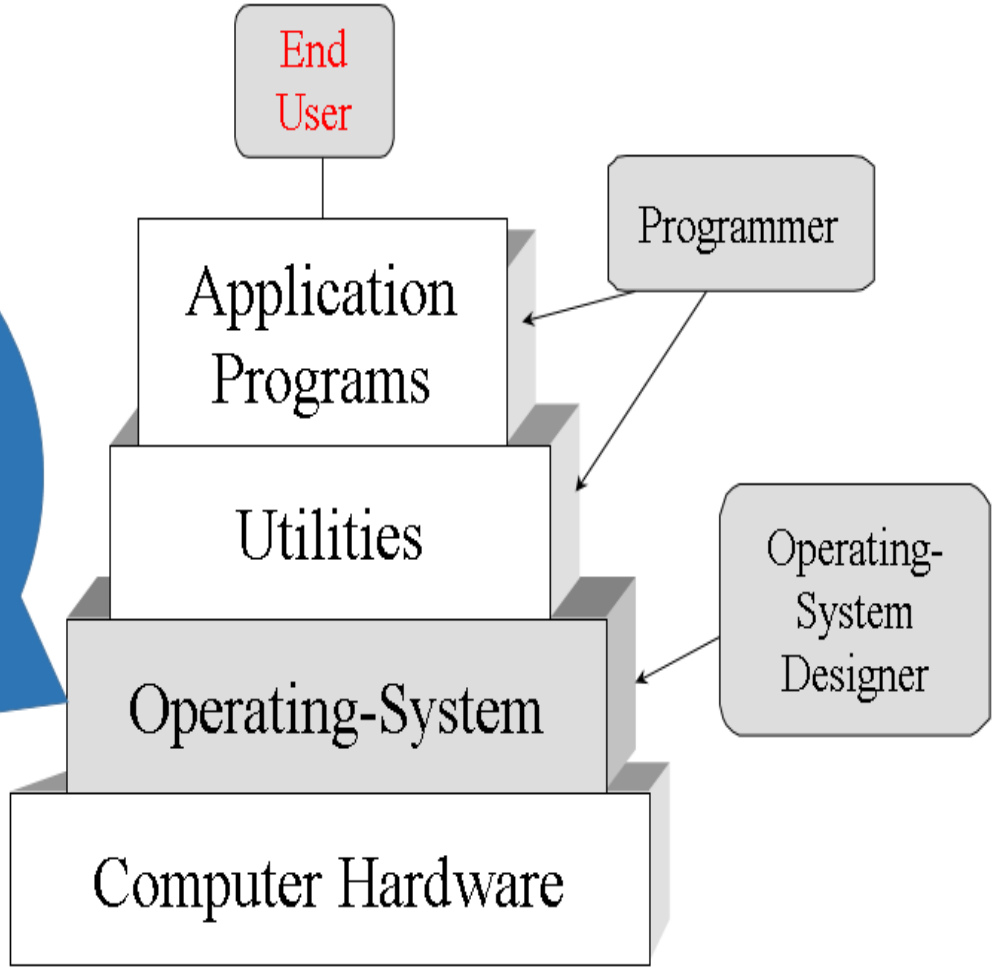


Different views of OS



Different views of OS

OS is made up of most important collection of system programs
OS act as mediator.
Make easier for the **Programmer** and **application program** to access the facilities and services of OS.



OS as a Resource Manager

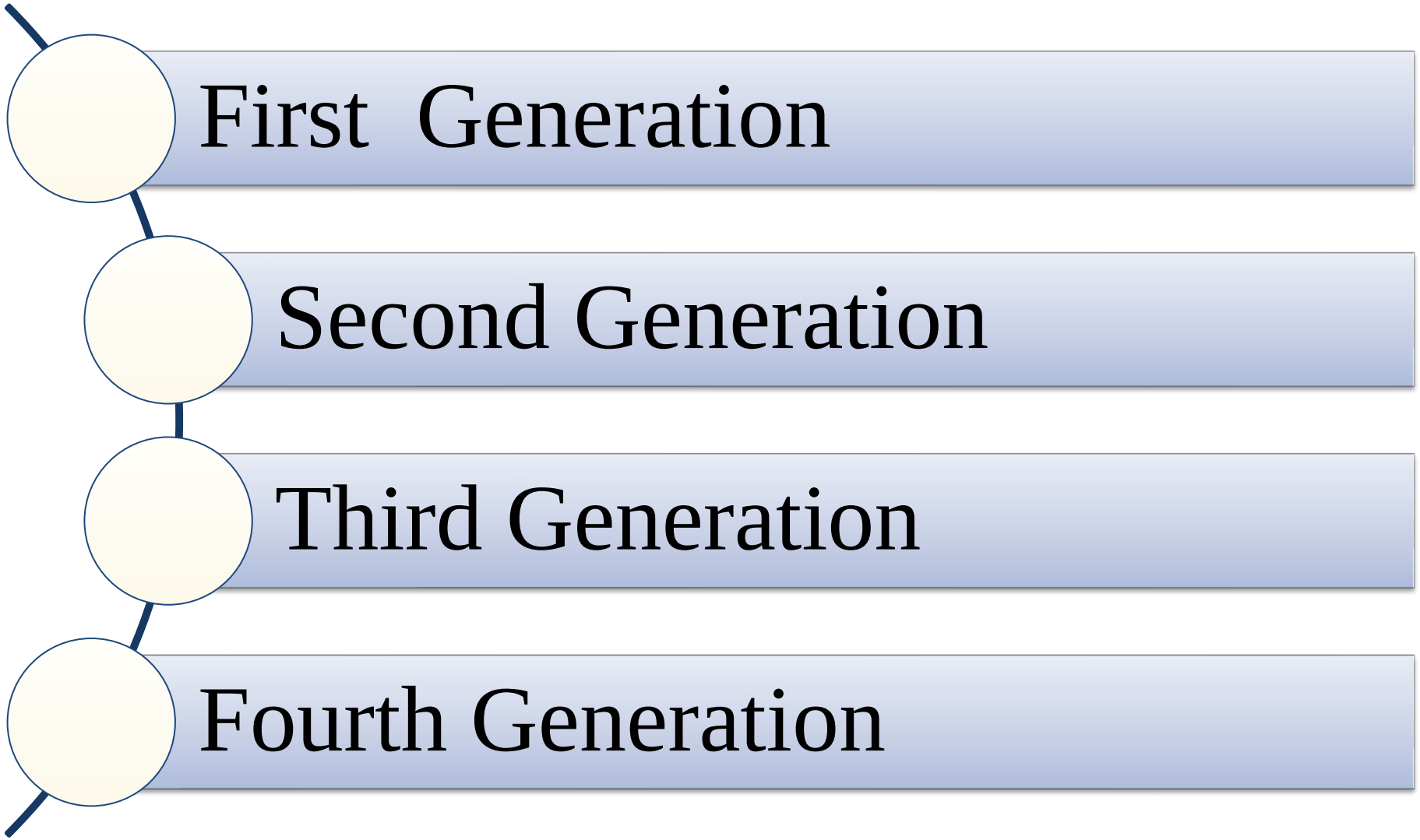
OS as a Resource manager (OS from system view)

- A computer is a set of resources which preform the function such as store, process and transfer the data.
- If a computer system is used by multiple application(or users),then they will compete for these resources.
- OS is responsible for managing the resources to control this functions.

Goals/objective of OS

- Control/execute user/application programs.
- Make the computer system convenient to use.
- Ease the solving of user problems.
- Use the computer hardware in an efficient manner.

Generations of Operating systems



History of OS (First generation)

- First generation (1945-1955)
  - **Vacuum tubes and plug-boards** are used in these systems.

Vacuum tubes

Plug board

First generation

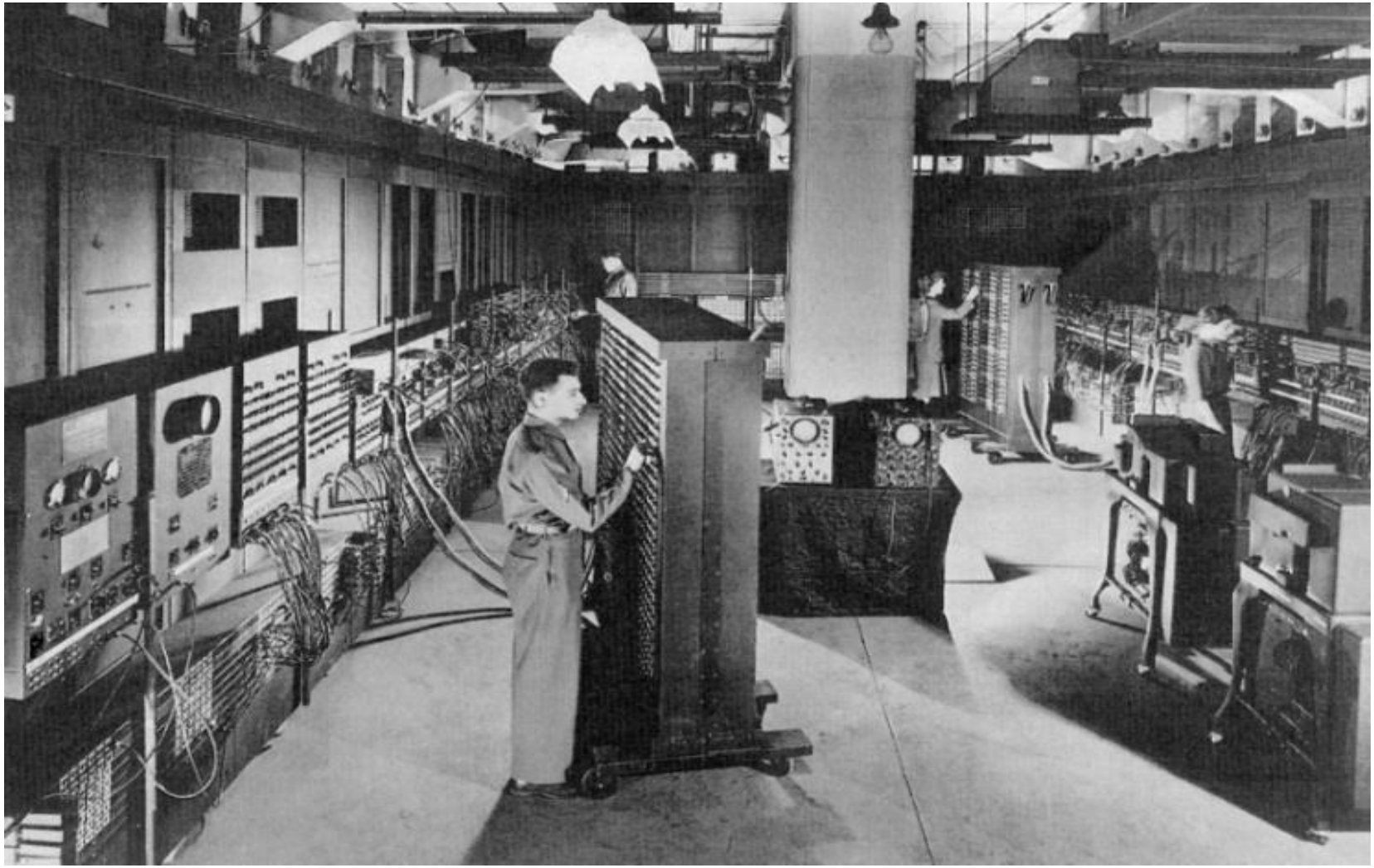
First Generation (1945-1955)

- Technology : Vacuum Tubes
- Operating System : Not present
- Language : Machine language

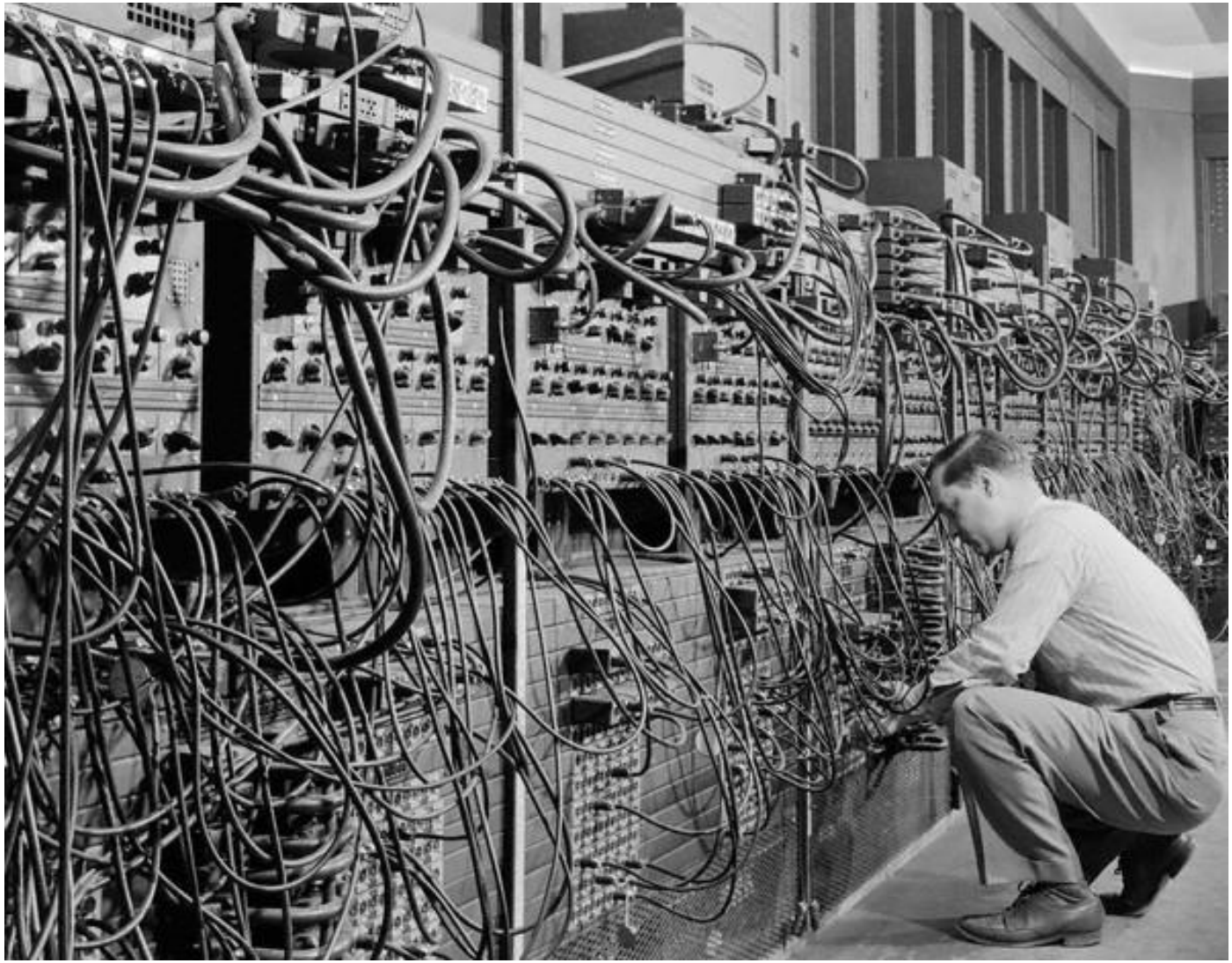
First generation

Mechanical Components are replaced by Electronic Components (Vacuum Tubes)

Vacuum Tubes

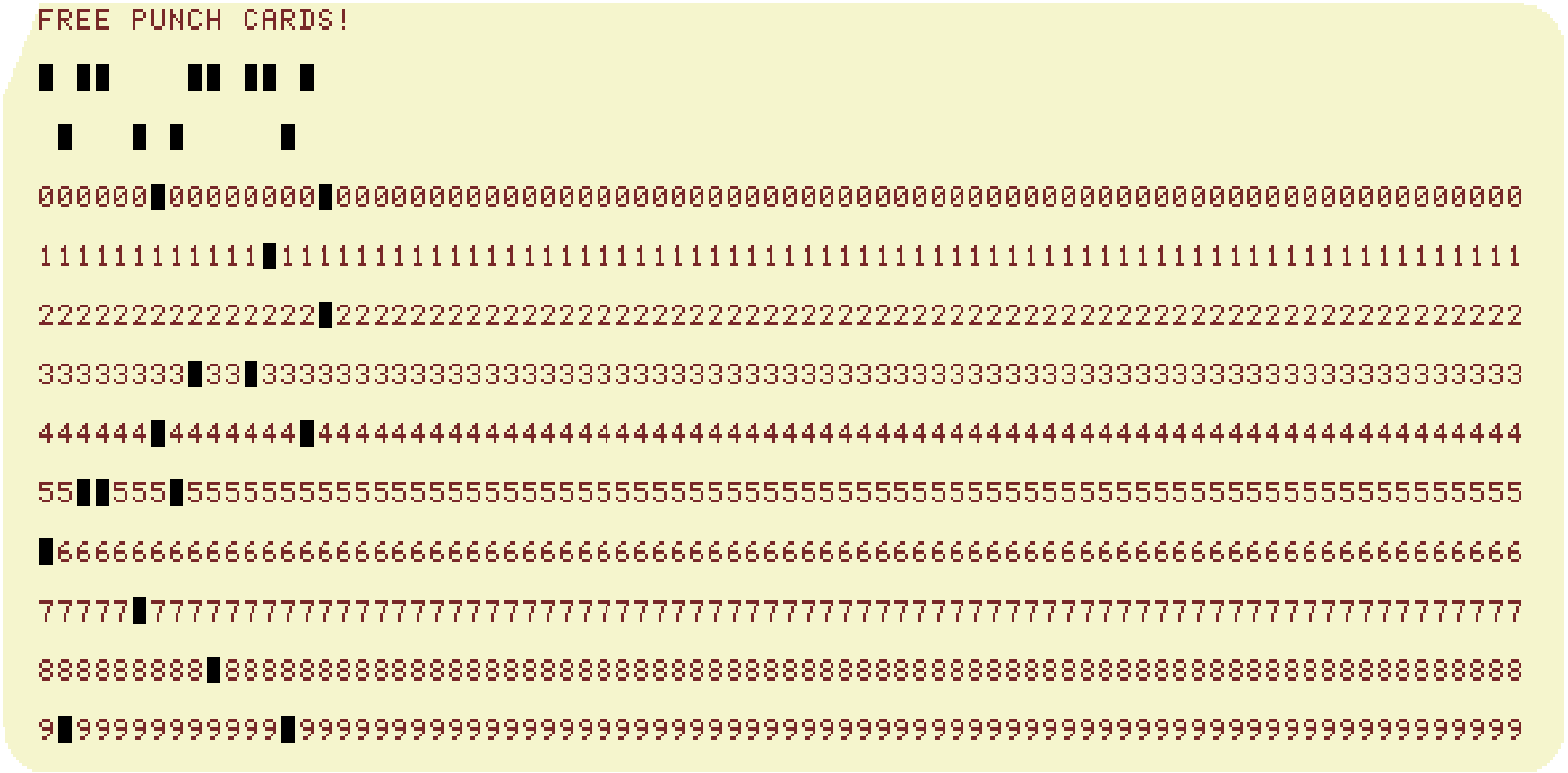


First Generation Computers



coding by cable connections (plug board)

Punch Card


FREE PUNCH CARDS!
000000 00000000 0000000000000000000000000000000000000000000000000000000000000000000000000000
111111111111 1111111111111111111111111111111111111111111111111111111111111111111111111111
222222222222222 2222222222222222222222222222222222222222222222222222222222222222222222222
33333333 33 3333333333333333333333333333333333333333333333333333333333333333333333333333
444444 4444444 4444444444444444444444444444444444444444444444444444444444444444444444444
55 55 555 5555555555555555555555555555555555555555555555555555555555555555555555555555
6666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
77777 7777777777777777777777777777777777777777777777777777777777777777777777777777777777
88888888 8888888888888888888888888888888888888888888888888888888888888888888888888888888
9 99999999999 9999999999999999999999999999999999999999999999999999999999999999999999999

First Generation

Working

- Computer run one job at time. Programmers have to enter the plug board or punch card into the computer, run it, record the result. (might require rewiring!).

Problem

- lots of wasted computer time!
- CPU was idle during first and last steps
- Computers were ***very*** expensive!

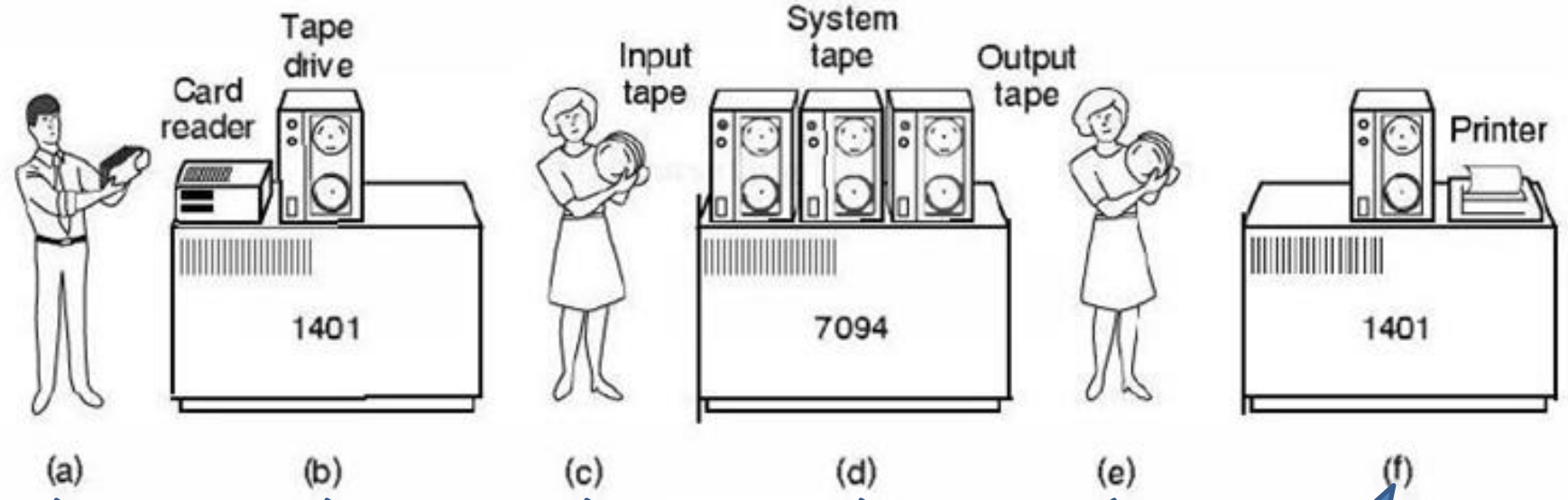
First Generation

Features of First Generation Computers

- They used vacuum tubes as main electronic component.
- They were large in size, slow in processing and had small storage capacity.
- They consumed lots of electricity and produced excessive heat.
- They were less reliable than later generation computers
- They used machine level language for programming

History of OS (Second generation)

- Second generation (1955-1965)
  - **Transistors** are used in these systems
  - The machine that are produced are called mainframes.
  - **Batch systems** was used for processing.



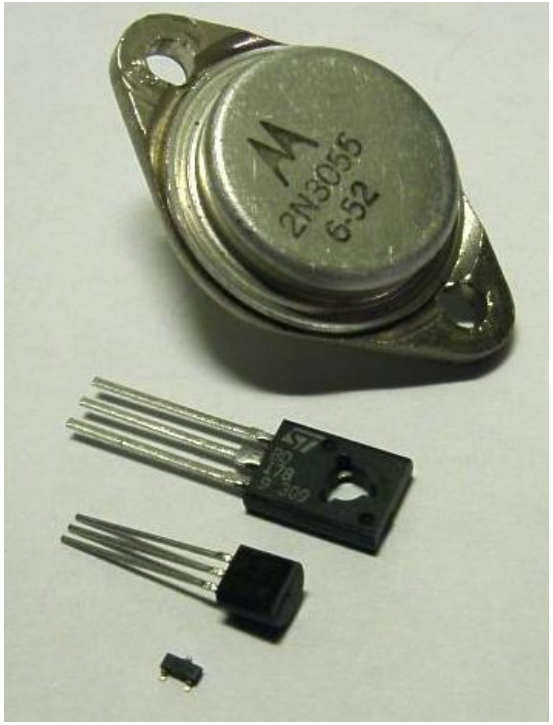
Second Generation

Second Generation (1955-65)

- Technology : Transistors
- Operating system : Present
- Language : Assembly and High level language

Second Generation

- Around 1955, Transistors were introduced.
- Operating systems were designed which is know as FMS (Fortran Monitor System) and IBMSYS.
- Used assembly language. Used FORTRAN as high level language.

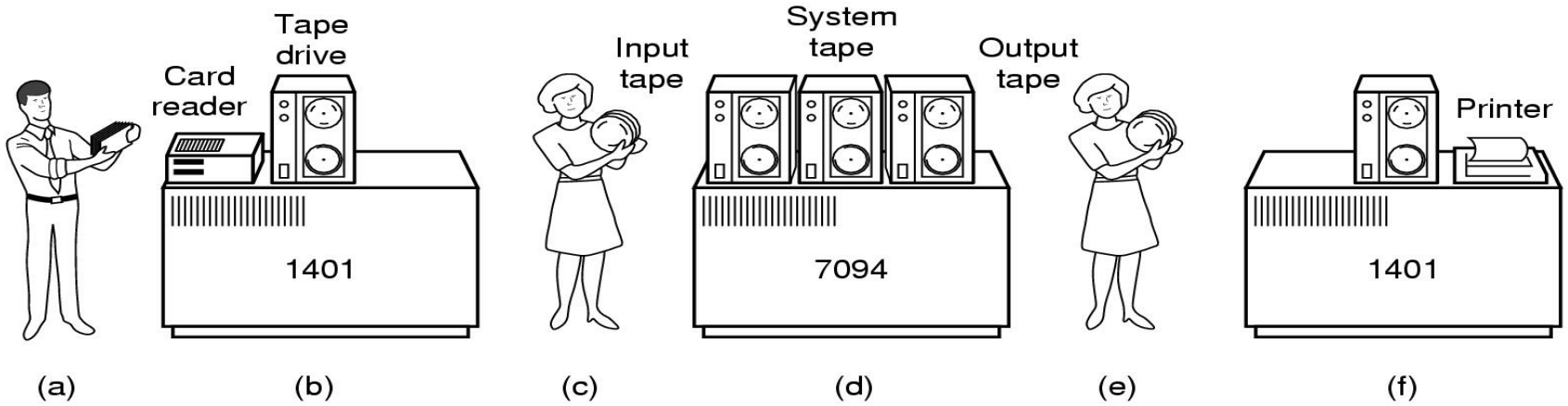


Transistors

Second Generation

Batch system

- To reduce the time new methodology is adopted know as batch system.
- To execute the program two computers were used IBM 1401 for reading cards, copying tapes, and printing output, and IBM 7094 for real computing (numerical calculation).



Batch system: Working

- Collect a tray full of jobs and read them onto magnetic tape using small computer such as IBM 1401.
- Then magnetic tapes are mounted on a tape drive., operator load special program which read the first job from tape and run it.
- Output was written onto second tape, after each job finished, the OS automatically read next job from the tape.
- Output tape inserted into IBM 1401 for printing.

Second Generation

Features of Second Generation Computers

- Second generation computers used transistors as their main electronic component.
- These computers were much smaller, reliable and more powerful
- Apart from machine language, assembly language were developed and used in this generation
- Some high level languages like COBOL & FORTRAN were introduced towards the end of second generation
- Printers, tape storage, disk storage, memory were started from second generation of computers
- Processing speed improved to microseconds

History of OS (Third generation)

- Third generation (1965-1980)
  - Integrated circuits (**IC's**) are used in place of transistors in these computers.
  - It provides **multiprogramming** (the ability to have several programs in memory at once, each in its own memory partition).

Third Generation

Third Generation(1965-80)

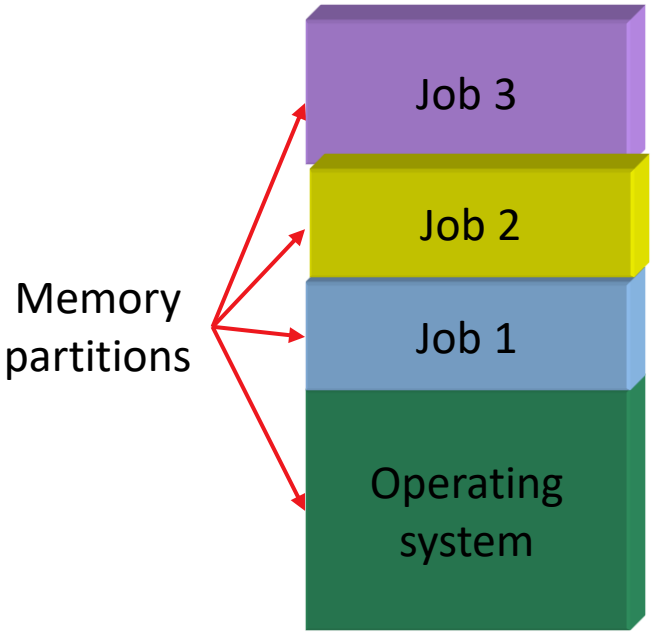
- Technology : Integrated Circuits
- Operating Systems : Present
- Language : High level language

44

Third Generation

Third Generation Computers

Third Generation



Multiprogramming

- Partition the memory into several pieces with different job in each partition.
- While one job was waiting for I/O to complete, another job could use CPU.
- whenever running job finished, the OS load new job from the disk into the empty partition of memory. This is know as **SPOOLING (Simultaneous Peripheral Operation On Line).**

History of OS (Fourth generation)

- Fourth generation (1980-present)
  - Personal Computers
  - **LSI** (**Large Scale Integration**) circuits, chips containing thousands of transistors are used in these systems.

Fourth Generation

Fourth Generation(1980-90)

- Technology : LSI
- Operating systems : Present
- Language : High level language

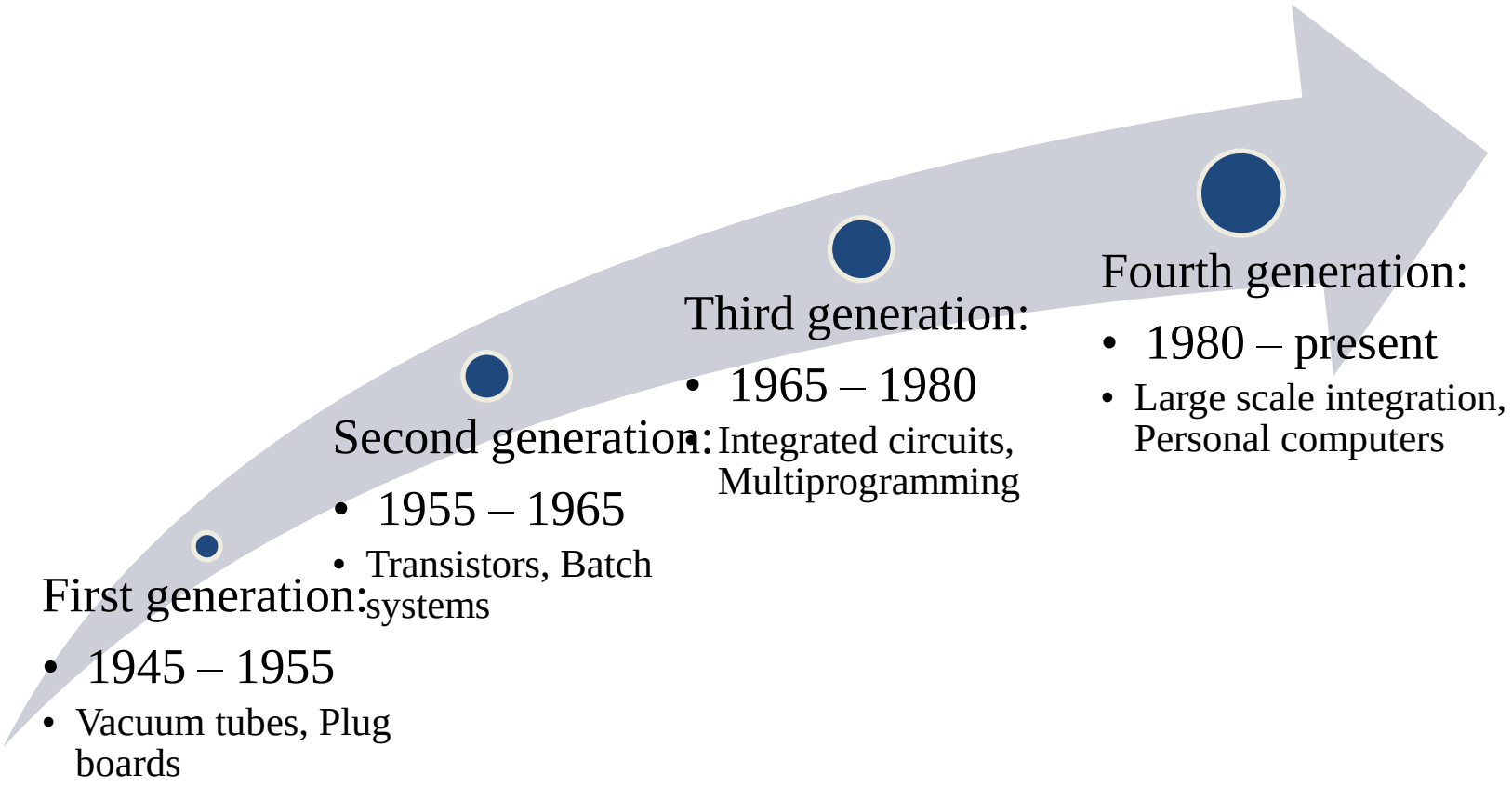
48

Fourth Generation

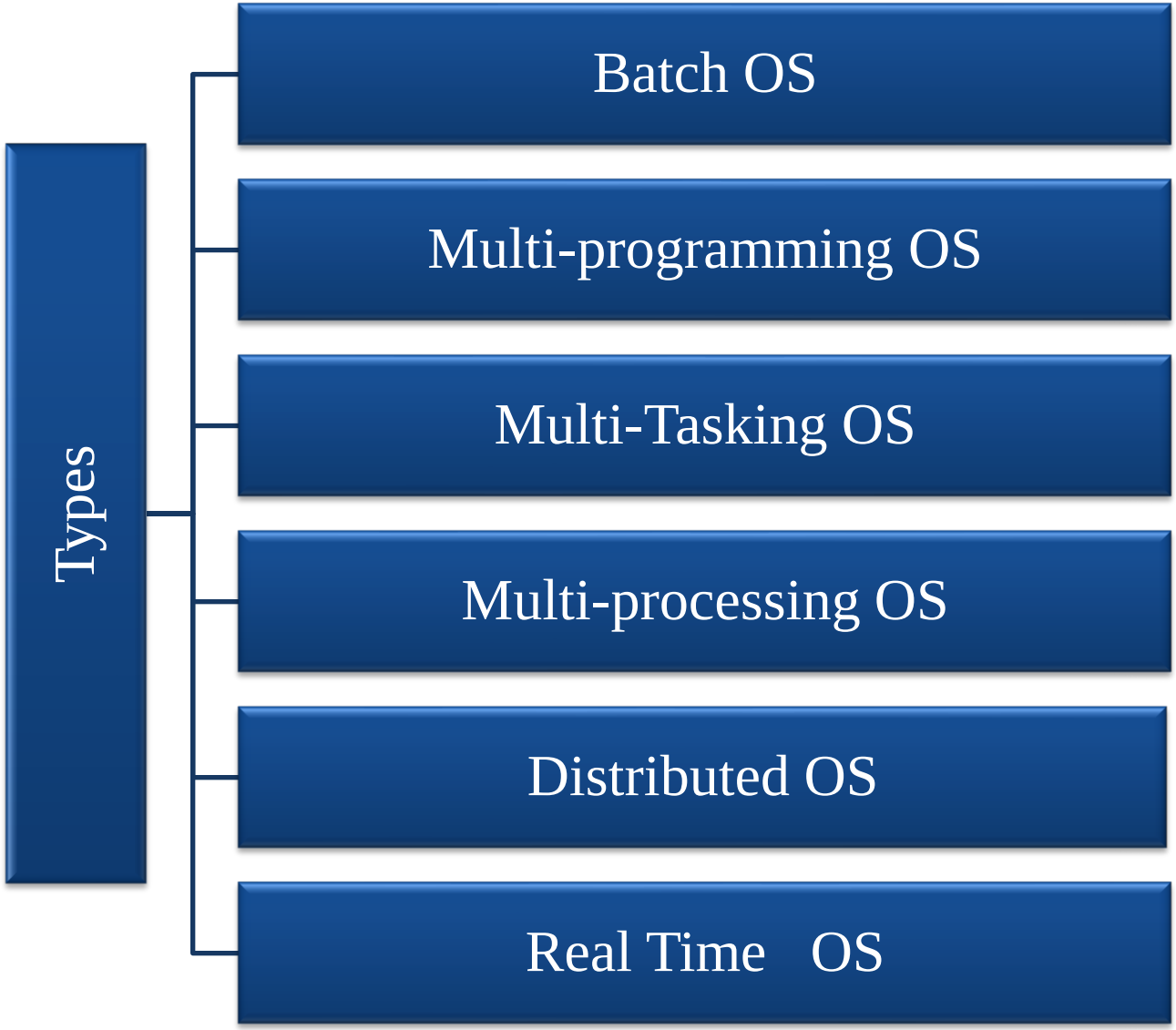
- Large Scale Integrated Circuits (LSIC's) were introduced.
- Intel came out with 8080, first microcomputer with a disk.
- Kindall wrote disk based OS called CP/M (Control Program for Microcomputers).
- Tim Paterson had developed OS know as Disk Operating System (DOS). Microsoft revised system and was known as MS-DOS.

LSIC

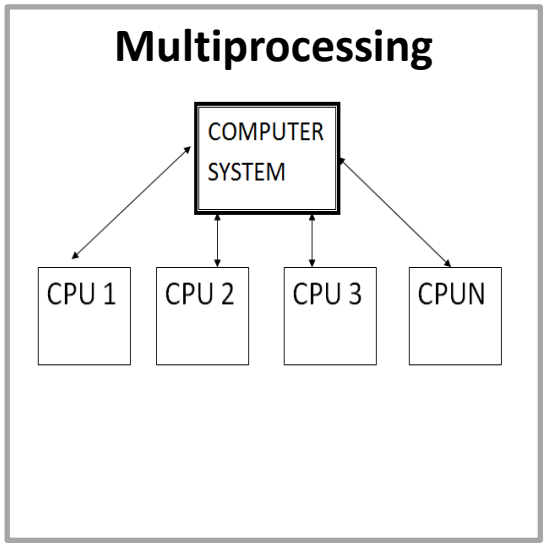
Generations of Operating systems: Summary

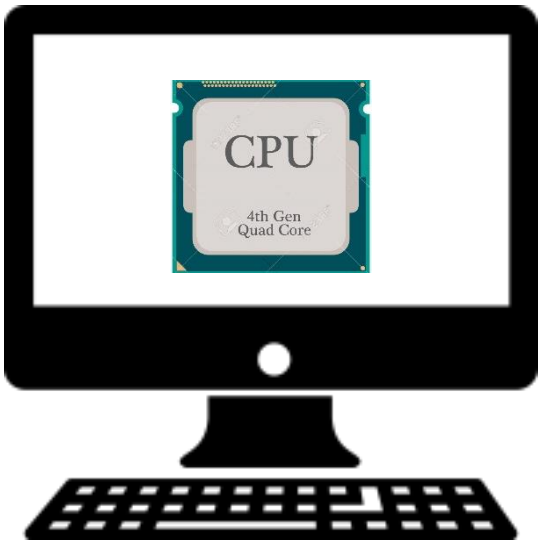


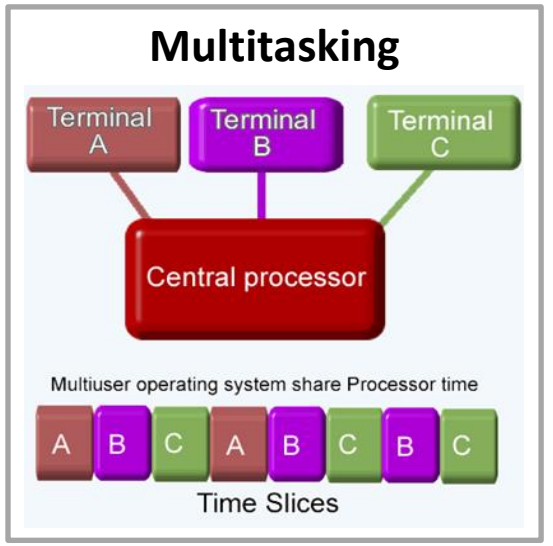
Types of Operating Systems

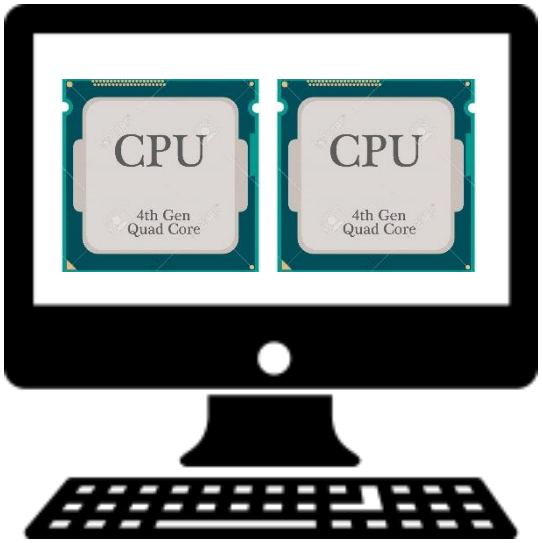


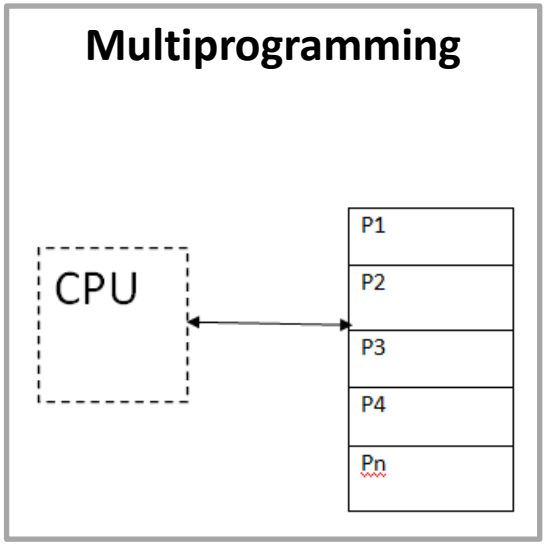
Multiprogramming v/s Multiprocessing v/s Multitasking

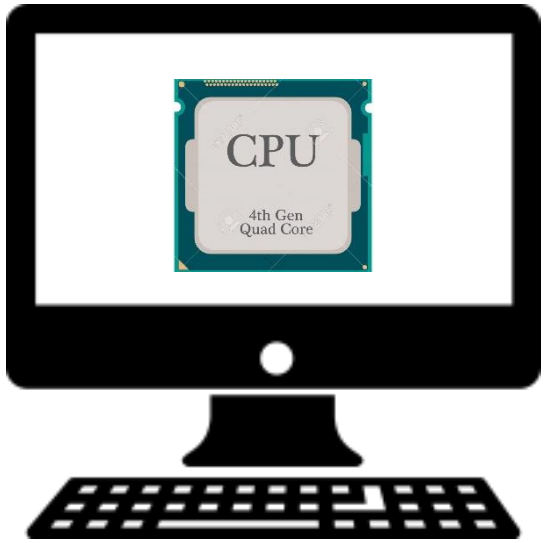












Multiprogramming v/s Multiprocessing v/s Multitasking

Multiprogramming	Multiprocessing	Multitasking
The concurrent residency of more than one program in the main memory is called as multiprogramming.	The availability of more than one processor per system, which can execute several set of instructions in parallel is called as multiprocessing.	The execution of more than one task simultaneously is called as multitasking.
Number of processor: one	Number of processor: more than one	Number of processor: one
One process is executed at a time.	More than one process can be executed at a time.	One by one job is being executed at a time.

Real Time OS

- A system is said to be Real Time if it is required to complete it's work & deliver it's services on time.
- A real time OS defines the completion of job within the *rigid time constraints* otherwise job looses its meaning.
- Example – Flight Control System, Air line reservation system.
  - All tasks in that system must execute on time.
- **The Real Time OS is of two types:**

1. Hard Real Time OS
2. Soft Real Time OS

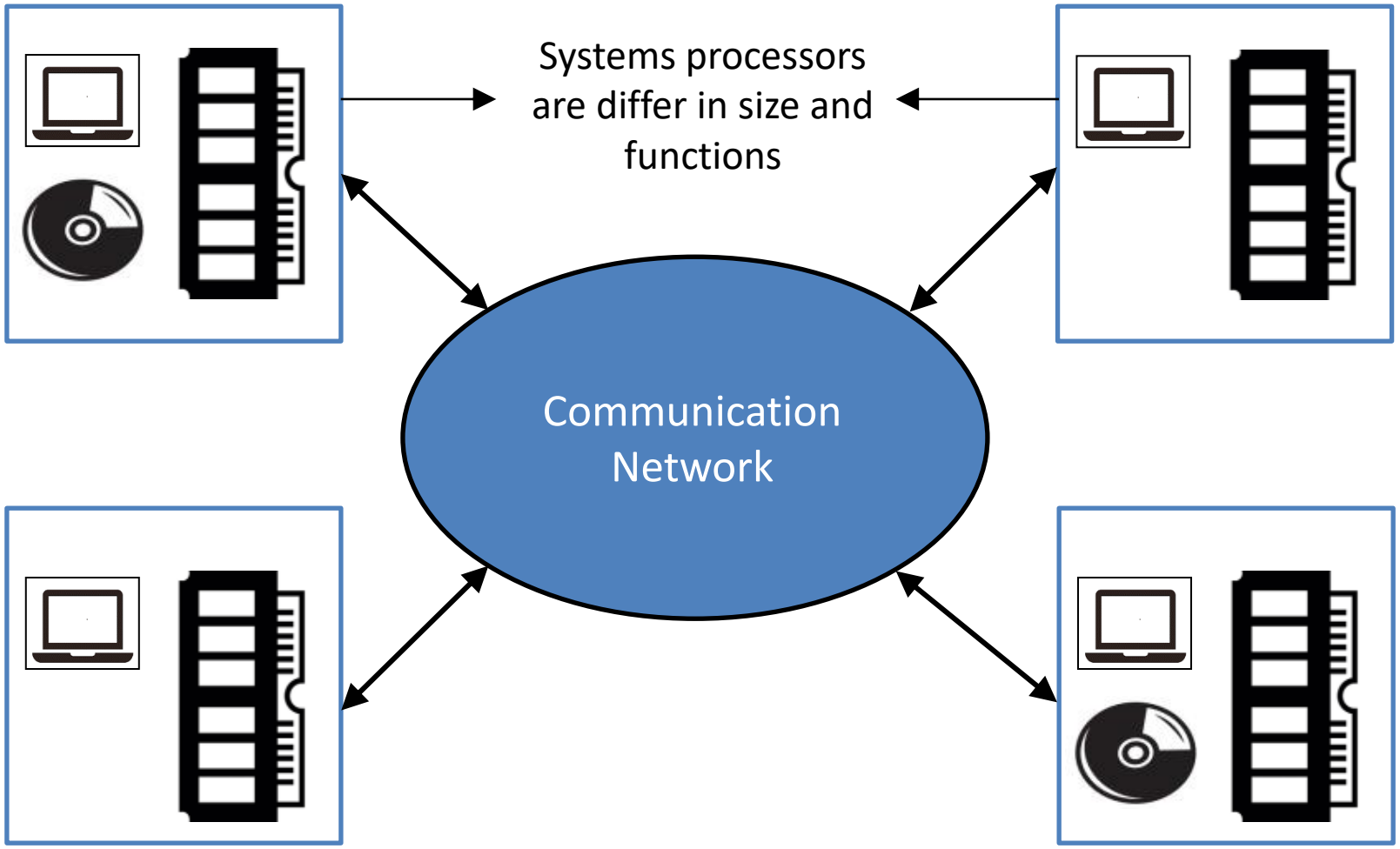
Real Time OS

- **Hard Real time**
  - It ensures the complication of critical tasks within the well defined constraints.
  - It can not afford missing even a single deadline, single miss can lead to the critical failure.
  - Example: Flight controller system.

- **Soft Real Time**
  - Late completion of jobs is undesirable but not fatal(does not leads to any critical failure) .
  - System performance degrades as more & more jobs miss deadlines.
  - Example: Online Databases, DVD player cannot process a frame.

Distributed Operating System

- "A Distributed system is collection of independent computers which are connected through network."



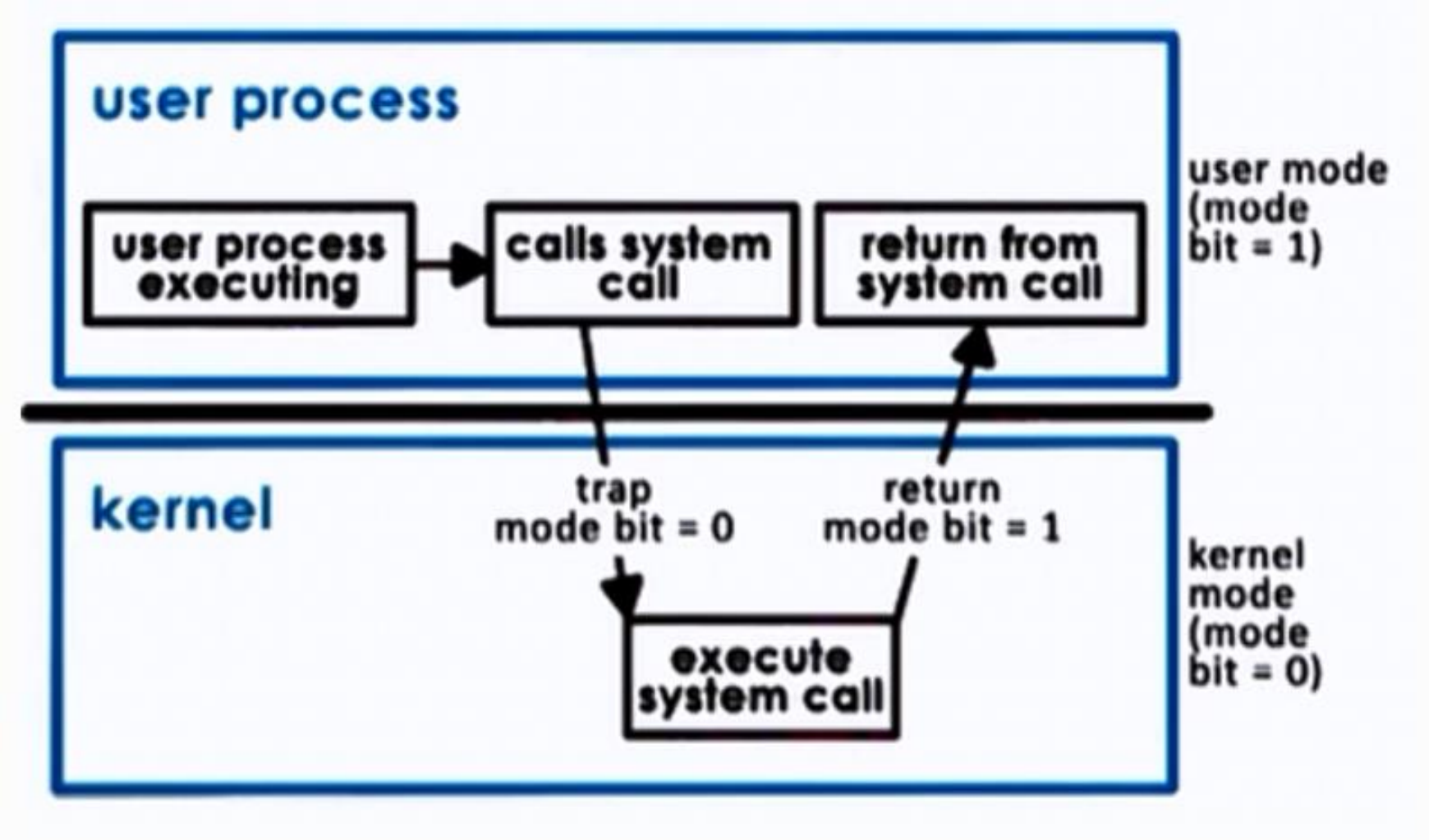
Examples of Distributed OS

- **Web Search Engines:**
  - Major growth industry in the last decade.
  - 10 billion per month for global number of searches.
  - e.g. Google distributed infrastructure

Distributed OS

Feature	Description
Resource sharing	Improves resource utilization
Reliability	Availability of services and resources
Computation speed up	Parts of communication system can be execute in different computer system.
Communication	Provides means of communication between remote entities.
Incremental Growth	Processing power can be enhanced.

User Mode vs. Kernel Mode



System Call

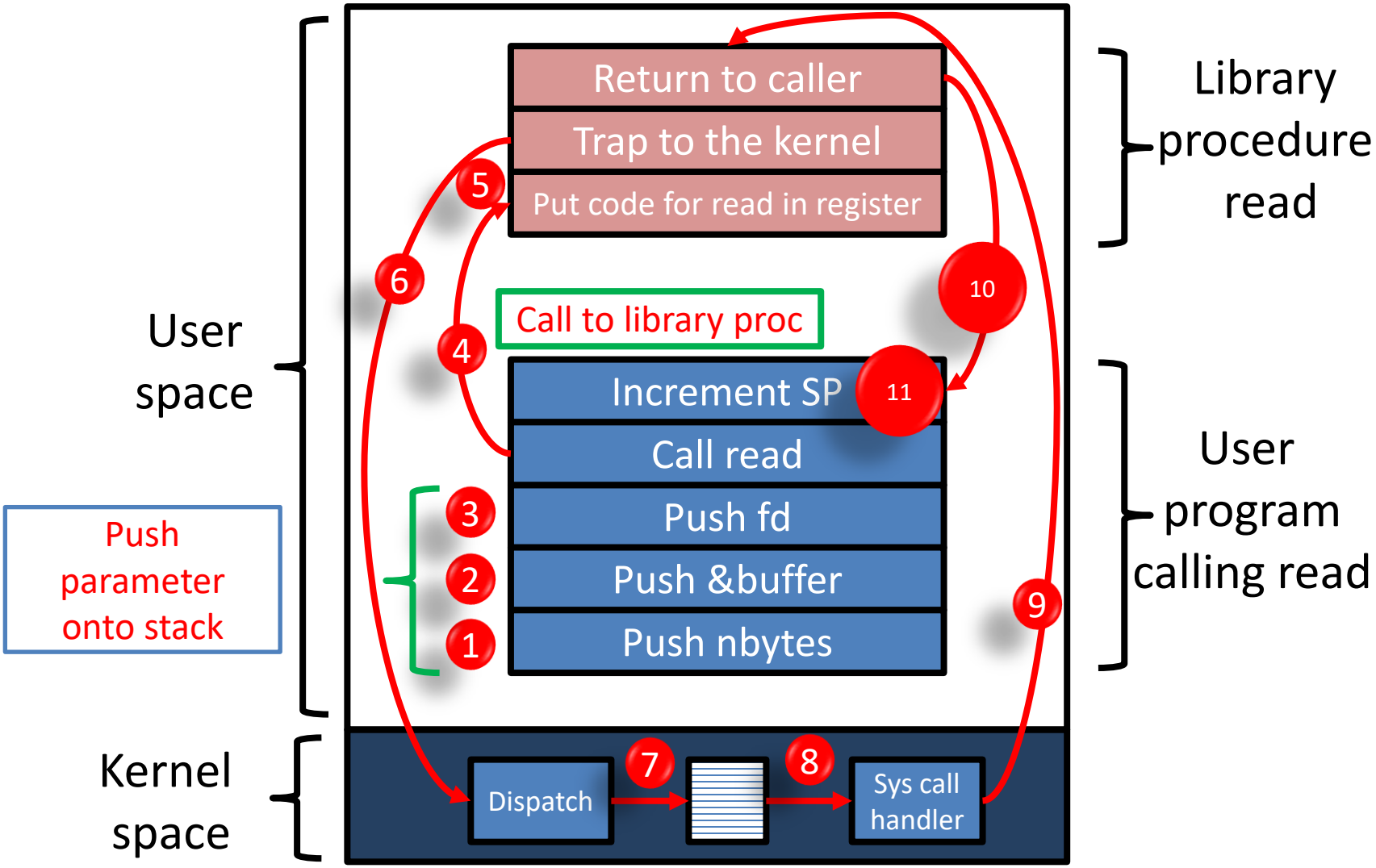
- A system call is a *request* that a program makes to the OS.
- OS and user program communicate with help of system call. System call are provided by the OS.
- Example : User program needs to read the file from the hard disk but as the user program is working in the user mode it can not have direct access to any computer hard ware, thus program makes system call instruction to transfer control to the operating system.
- Then OS process the system call and returns control to the instruction.

System Call

Read system call

- count = read (fd, buffer, nbytes);
- Where
  - fd specifies the file,
  - Buffer tells where the data are to be put,
  - nbytes specifies number of bytes to read
  - Count will have number of bytes which actually read by this system call.

Read System Call



Read System Call

- **Step-1,2,3** calling program first pushes the parameters onto the stack The first and third parameters are called by value, but the second parameter is passed by reference, meaning that the address of the buffer (indicated by &) is passed, not the contents of the buffer.

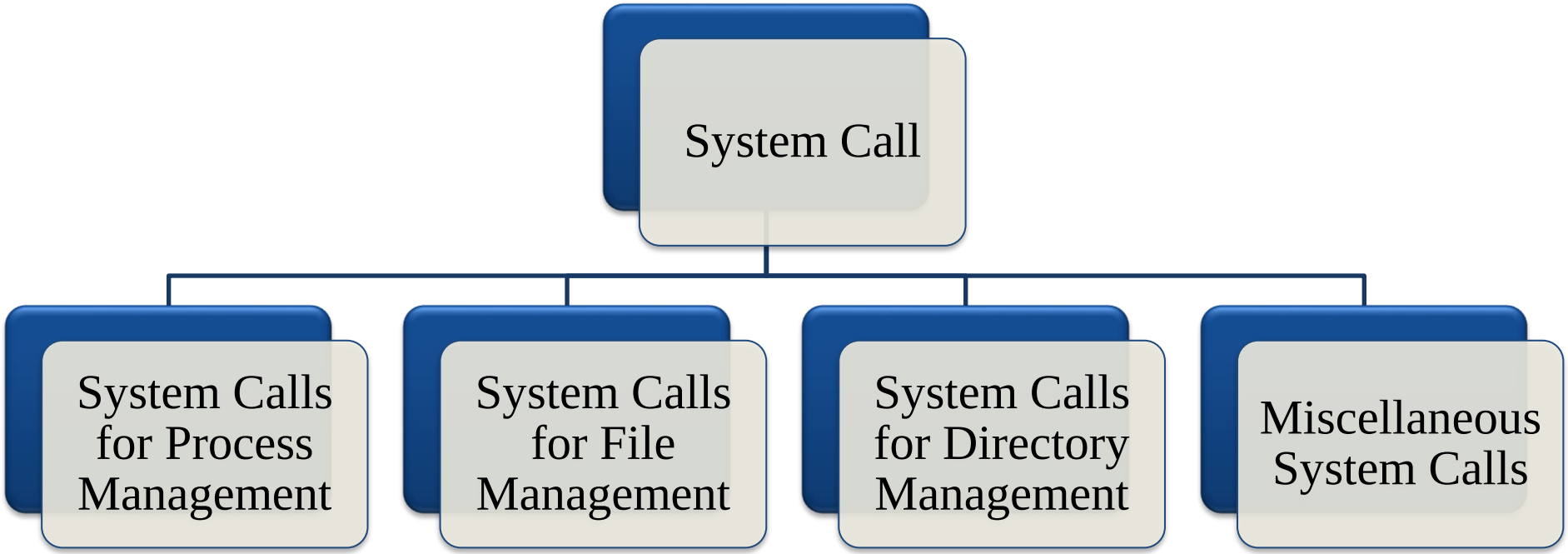
- **Step-4** Then comes the actual call to the library procedure. This instruction is the normal procedure call instruction.

- **Step-5** The library procedure, possibly written in assembly language, typically puts the system call number in a place where the operating system expects it, such as a register.

Read System Call

- **Step-6.** Then it executes a TRAP instruction to switch from user mode to kernel mode and start execution at a fixed address within the kernel.
- **Step-7.** The kernel code that starts examines the system call number and then dispatches to the correct system call .
- **Step-8**. At that point the system call handler runs.
- **Step-9.** Once the system call handler has completed its work, control may be returned to the user-space library procedure .
- **Step-10.** This procedure then returns to the user program in the usual way procedure calls return.
- **Step-11**. To finish the job, the user program has to clean up the stack, as it does after any procedure call.

System Call


System Call
System Calls for Process Management
System Calls for File Management
System Calls for Directory Management
Miscellaneous System Calls

System Calls for Process Management

Systems call for process management:

- pid = fork()
- pid = waitpid(pid, &statloc, options)
- s = execve(name, argv, environp)
- exit(status)

System Calls for Process Management

pid = fork()

- Fork is the only way to create new process in POSIX.
- Creates an exact duplicate of the original process.
- It will return PID which id of child.

pid = waitpid(pid, &statloc, options)

- After creation of the child process **parent waits for the child to execute the command,**
- Then read the next command when child terminates.
- **Pid is the id of child**,
- Statloc will set to the child's exit status,
- Various options can be specified with help of the third parameter.

System Calls for File Management

fd = open (file, how,...)

- Open a file for reading, writing or both.
- Specifies the file name to be opened and any of the following code *O_RDONLY*, *O_WRONLY*, or *O_RDWR*, *O_CREAT*.

s = close(fd)

- Close an open file.

n = read(fd, buffer, nbytes)

- Read data from a file into a buffer.

n = write(fd, buffer, nbytes)

- Write data from a buffer into a file.

System Calls for File Management

position = lseek(fd, offset, whence)

- Move the file pointer for reading or writing thus it can begin anywhere in the file.
- Fd is **file descriptor**
- Offset **shows the file position**
- Whence tells whether the **file position** is relative to beginning, or current position, or end of the file.

s = stat(name, &buf)

- Get a **file's status information** such as type of file, size, time of last modification and other information.
- Name is the file name
- Pointer to structure where this needs to be put.

System Calls for Directory Management

s = mkdir(name, mode)

- Create a new empty directory.

s = rmdir(name)

- Remove an empty directory.

System Calls for Directory Management

s = unlink(name)

- Remove a directory entry

s = mount(special, name, flag)

- Allows two file systems to be merged into one.
- It mounts a storage device or file system, making it accessible and attaching it to an existing directory structure.
- Special is the name of block special file for drive.
- Name is destination space where we want to mount file.
- Flag is Mode of mount read-write or read only.

Miscellaneous System Calls

s = chdir(dirname)

- Change the working directory

s = chmod(name, mode)

- Change a file's protection bits.
- Name is the name of the file
- Mode includes the read-write –execute bits for user, owner, group.

Miscellaneous System Calls

s = kill(pid, signal)

- Send *a signal* to a process. The kill system call is the way users and user processes send signals. If a process is prepared to catch a particular signal, then when it arrives, a signal handler is run.
- If the process is not prepared to handle a signal, then its arrival kills the process (hence the name of the call).
- Common uses are : to kill the process with pid, or to force process *N* to read its configuration file.

seconds = time(&seconds)

- Returns current time in second.

System Calls: Interrupts

- An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.
- The processor responds by suspending its current activities, saving its state, and executing a function needed by interrupt.
- This interruption is temporary, and, after the interrupt handler finishes, the processor resumes normal activities.
- There are two types of interrupts: hardware interrupts and software interrupts.

Interrupts

Hardware interrupt

pressing a key on the keyboard or moving the mouse triggers hardware interrupts that cause the processor to read the keystroke or mouse position.

Software interrupt

errors or events occurring during program execution that are cannot be handled within the program itself.

For example, if the processor's arithmetic logic unit is commanded to divide a number by zero.

Operating system structures

1. Simple Structure
2. Monolithic systems
3. Layered systems
4. Microkernel structure
5. Modular structure

Simple Structure

- MS-DOS is an operating system created for personal computers. It was developed by Microsoft. It is a classic example of an operating system with a layered structure.
- MS-DOS operating system is split into various layers and each of the layers have different functionalities.
- Layering provides a distinct advantage in the MS-DOS operating system because all the layers can be defined separately and interact with each other as required.
- Also, it is easier to create, maintain and update the system if it is done in the form of layers. Change in one layer specification does not affect the rest of the layers.
- However, the layers in MS-DOS are not that sharply defined and the layer specifications often bleed into each other.

SIMPLE STRUCTURE

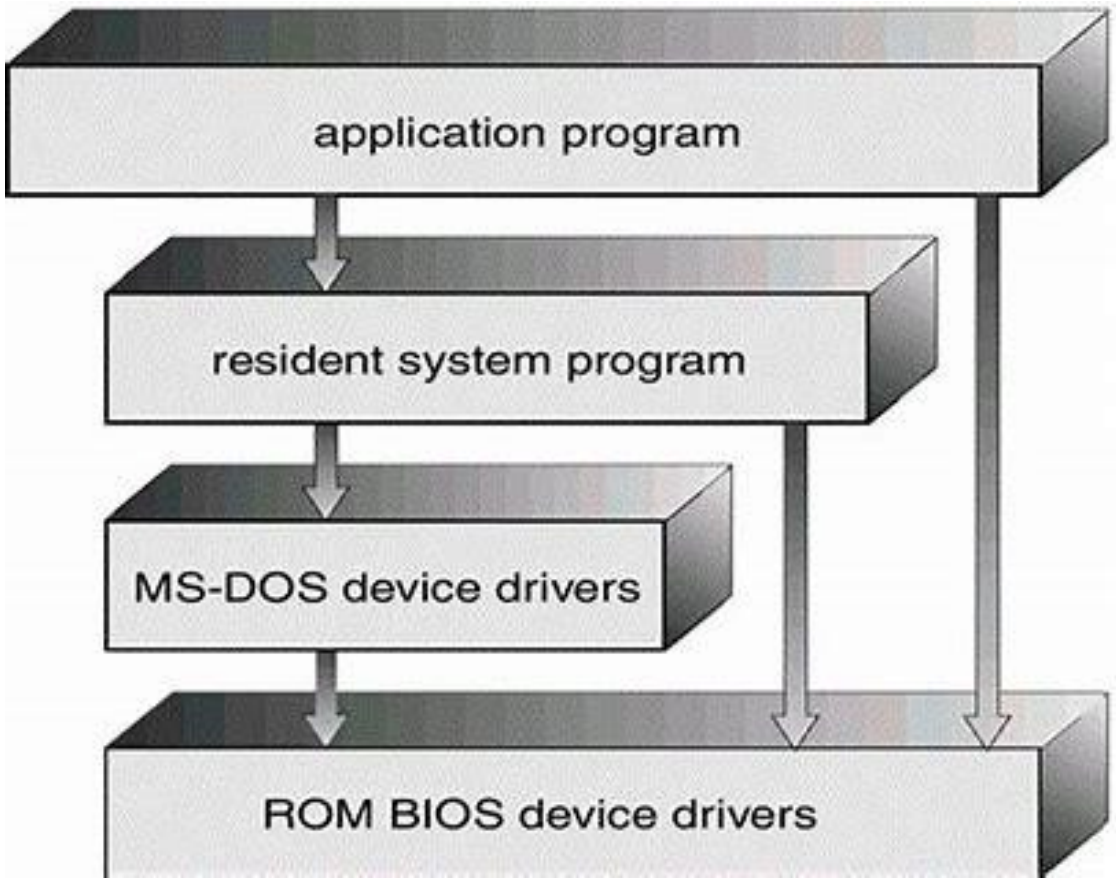
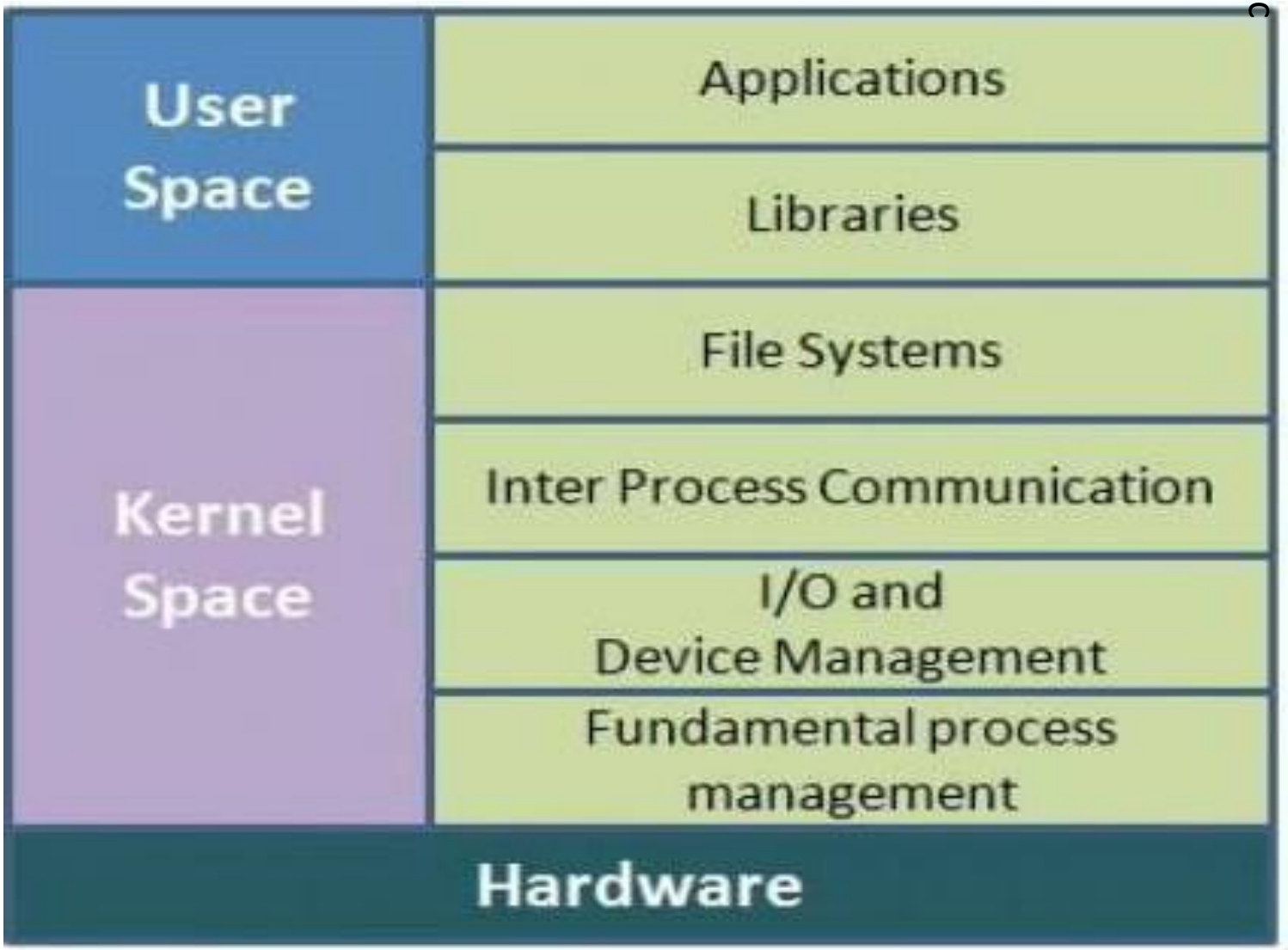


Fig: MS-DOS layer Structure

Monolithic Structure

- Monolithic systems – basic structure:
- the entire operating system is working in kernel space.
- A set of primitives or system calls implement all operating system services such as process management, concurrency, and memory management.
- Device drivers can be added to the kernel as modules.
- **user services** and **kernel services** are implemented under same address space. As both services are implemented under same address space, this makes operating system execution faster.
- If any service fails the entire system crashes, and it is one of the drawbacks of this kernel. The entire operating system needs modification if user adds a new service.
- Acts as a virtual machine which controls all hardware parts
- Examples: Linux, Unix like kernels, BSD, OS/360

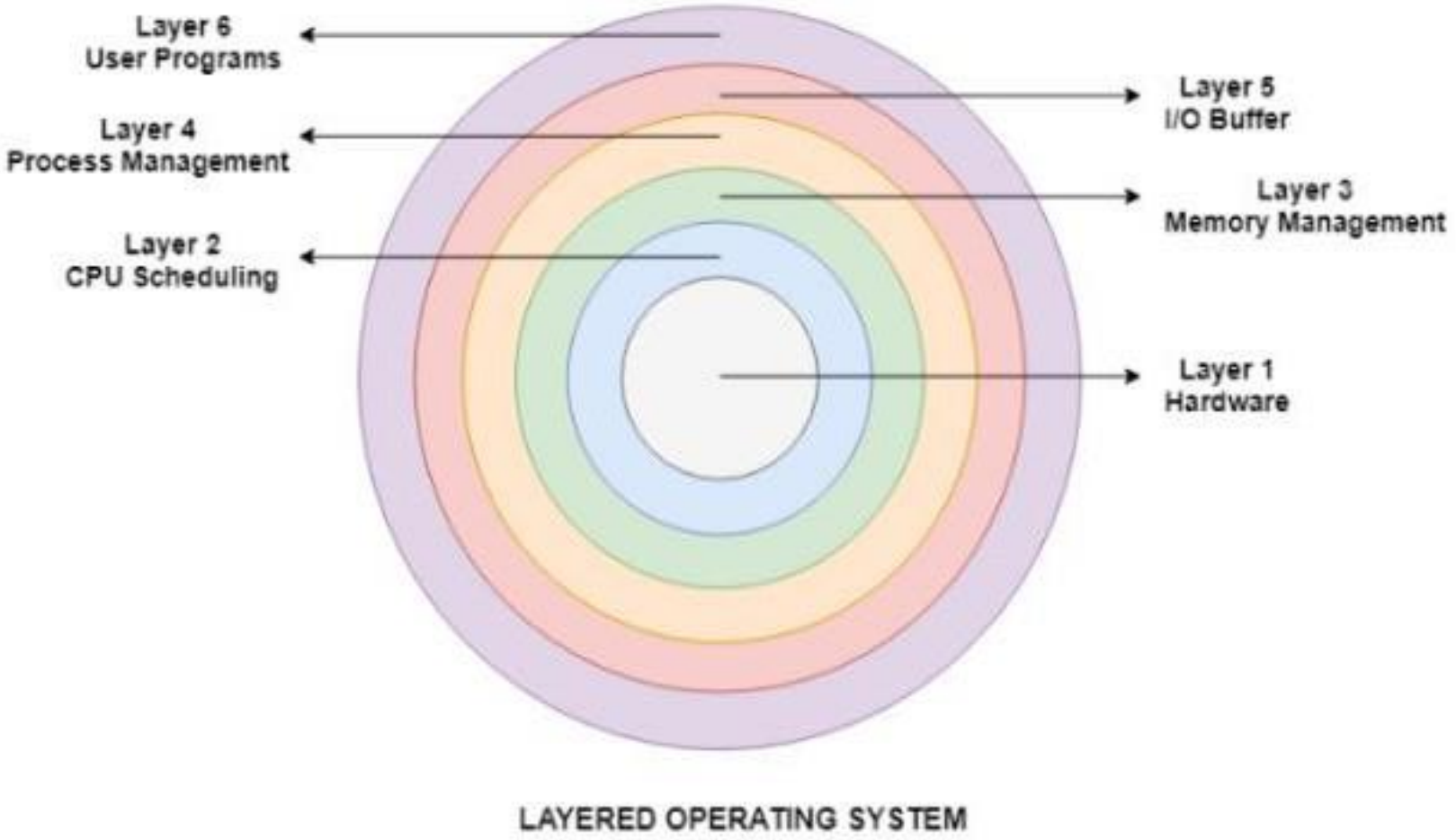
MONOLITHIC STRUCTURE



Layered Structure

- A layered system consists of a **series of layers**, each of which depends only on the correct operation of the layer immediately beneath it.
- The lowest layer represents the hardware interface, and the highest layer the application interface.
- All the layers can be defined separately and interact with each other as required. Also, it is easier to create, maintain and update the system if it is done in the form of layers.
- Change in one layer specification does not affect the rest of the layers.
- **A key problem in a layered system is deciding on the ordering of the layers.** (This is critical because of the requirement that each layer can ultimately only use services provided by layers below it - so the ordering cannot have any cycles.)
- In a strictly-layered structure, efficiency can also become a problem because when a higher-level layer requires a lower-level operation the request must work its way down layer by layer.
- Examples OS/2, window NT

LAYERED STRUCTURE



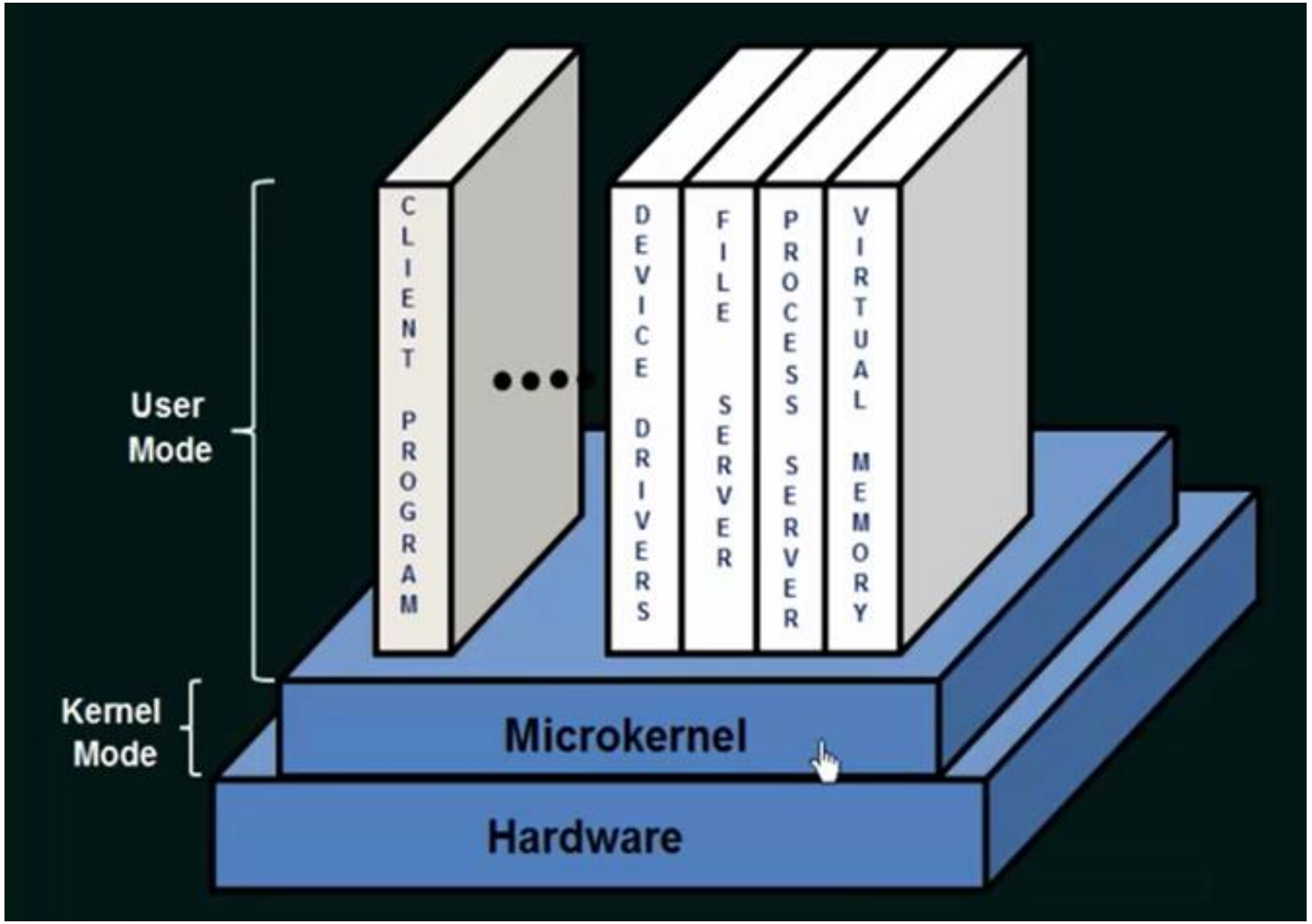
LAYERED OPERATING SYSTEM

Microkernel Structure

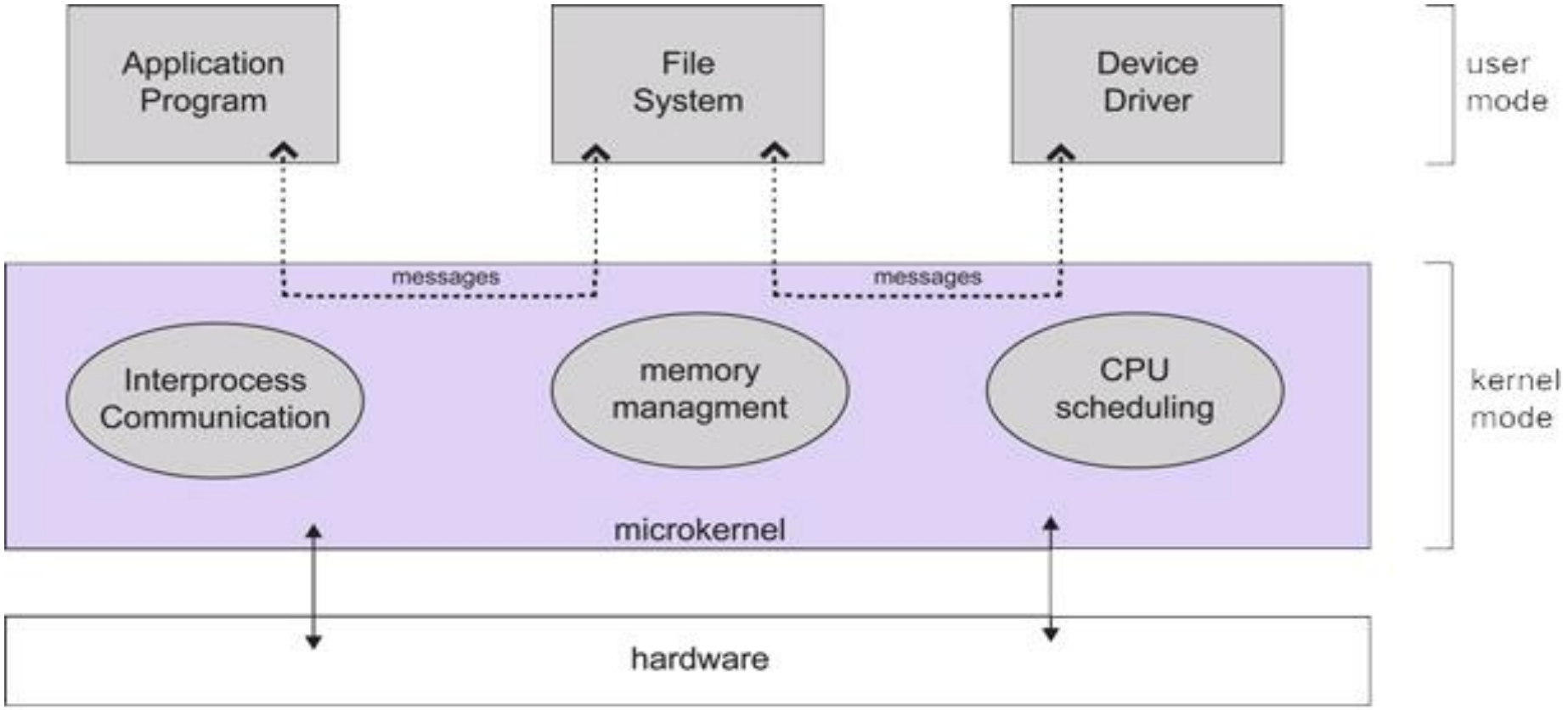
- Kernel is made as small as possible only the most important services are put inside the kernel and rest of the OS service are present in the system application program.
- The user can easily interact with those not-so important services within the system applications kernel is solely responsible for the three most important services of operating system namely:
  - Inter-Process communication
  - Memory management
  - CPU scheduling
- Microkernel and system applications can interact with each other by *message passing* as and when required.
- This is extremely advantageous architecture since burden of kernel is reduced and less crucial services are accessible to the user and hence security is improved too. It is being highly adopted in the present-day systems.

- MacOSX, Eclipse IDE is a good example of Microkernel Architecture.

MICROKERNEL STRUCTURE



MICROKERNEL STRUCTURE

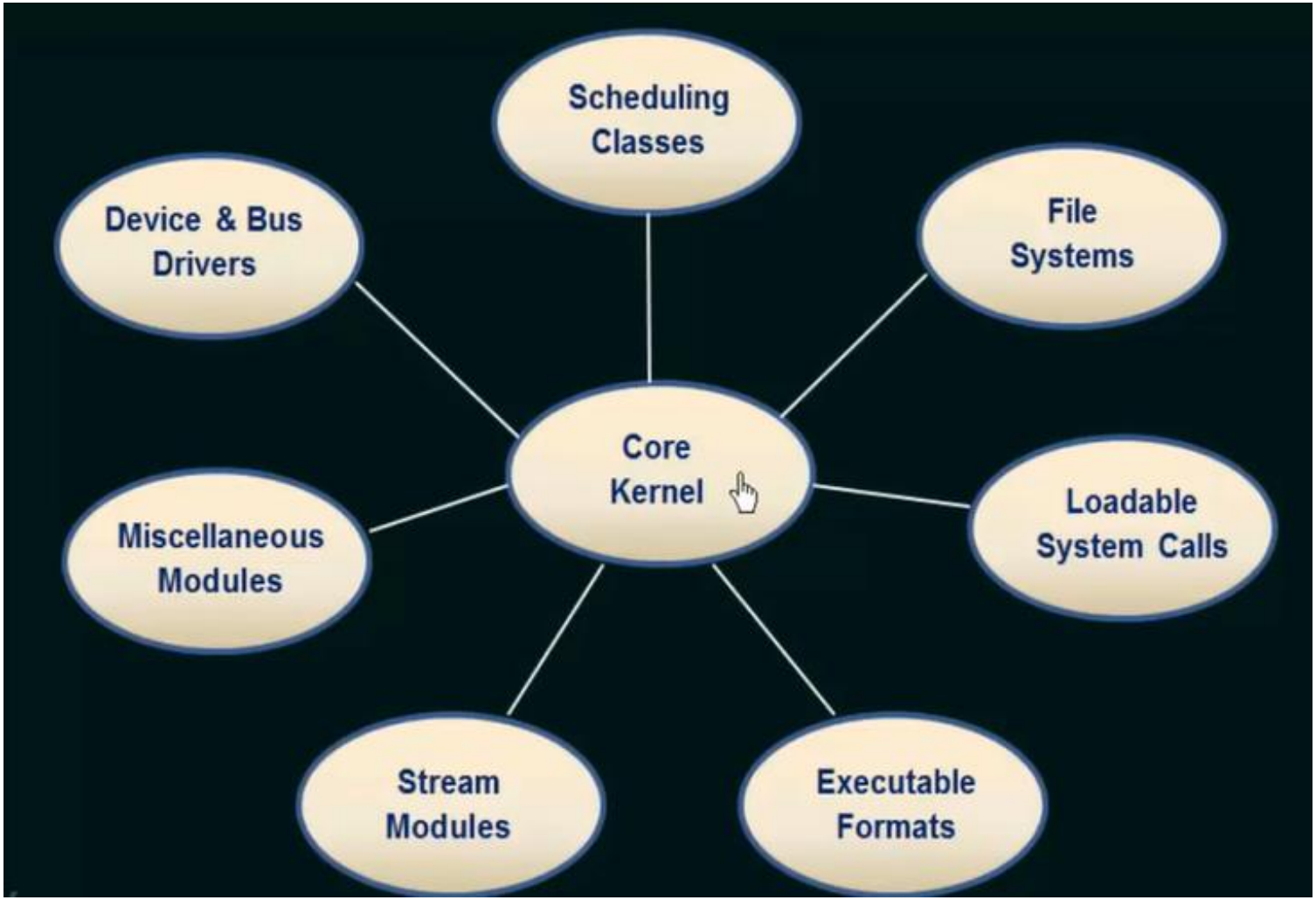


Architecture of a typical microkernel

Modular Structure

- A modular operating system is built with its various functions broken up into distinct processes, each with its own interface.
- The primary benefit of the modular approach is that each process operates independently, and if one of them fails or needs an update, it won't affect any of the other functions.
- The main elements of a modular operating system are a kernel and a set of dynamically loadable applications with their own discrete memory spaces. The kernel is protected from service and application failures.
- The Linux kernel is modular, which means it can extend its capabilities through dynamically-loaded kernel modules.

MODULAR STRUCTURE



Questions asked in GTU Exams

1. What is Operating System? Discuss role/functions of OS as a resource manager.
2. Write different operating system services and describe it with suitable example.
3. Enlist various generations of operating system and explain them with advantages and disadvantages in detail.
4. Describe RTOS its types and DOS with suitable examples.
5. Identify the differences between Multi-Programming, Multi-tasking, and Multiprocessing System.
6. What are System Calls? Explain various types of system calls with appropriate examples.
7. What is an interrupt and context switching? How it is handle by operating system.
8. Explain Monolithic, Micro Kernel and Layered system of Operating System Structure.


THANK
YOU