INDEX

NAME: KIRAN M.

UNIT - 1

INTRODUCTION TO OPERATING SYSTEMS, SYSTEM STRUCTURES

Chapter 1: Introduction

Objectives: * To provide a grand tour of the major components of operating systems.

* To describe the basic organization of computer systems.

● What is an Operating System? What Operating Systems do

→ An operating system is a program that manages the computer hardware.

→ It provides a basis for application programs and act as an intermediary between the computer user and the computer hardware.

→ Mainframe os are designed primarily to optimize utilization of hardware.

→ personal computer (PC) os support complex games, business applications, and so on.

→ Os for hand-held computers are designed to provide an environment in which a user can easily interface with the computer to execute programs.

→ Operating system goals:

* Execute user programs and make solving user problems easier.
* Make the computer system convenient to use.
* Use the computer hardware in an efficient manner.

A computer system can be divided roughly into four components:

* The Hardware - provides basic computing resources.

≫ Central processing unit (CPU), memory, I/O devices.

* The Operating system - Controls and coordinates use of hardware among various applications and users.

* The Application Programs - define the ways in which these (system) resources are used to solve the computing problems of the users.

≫ Word processors, spreadsheets, compilers, web browsers

* The Users - People, machines, other computers.

Views of OS.

Iser View - The user view of the computer depends on the interface used.

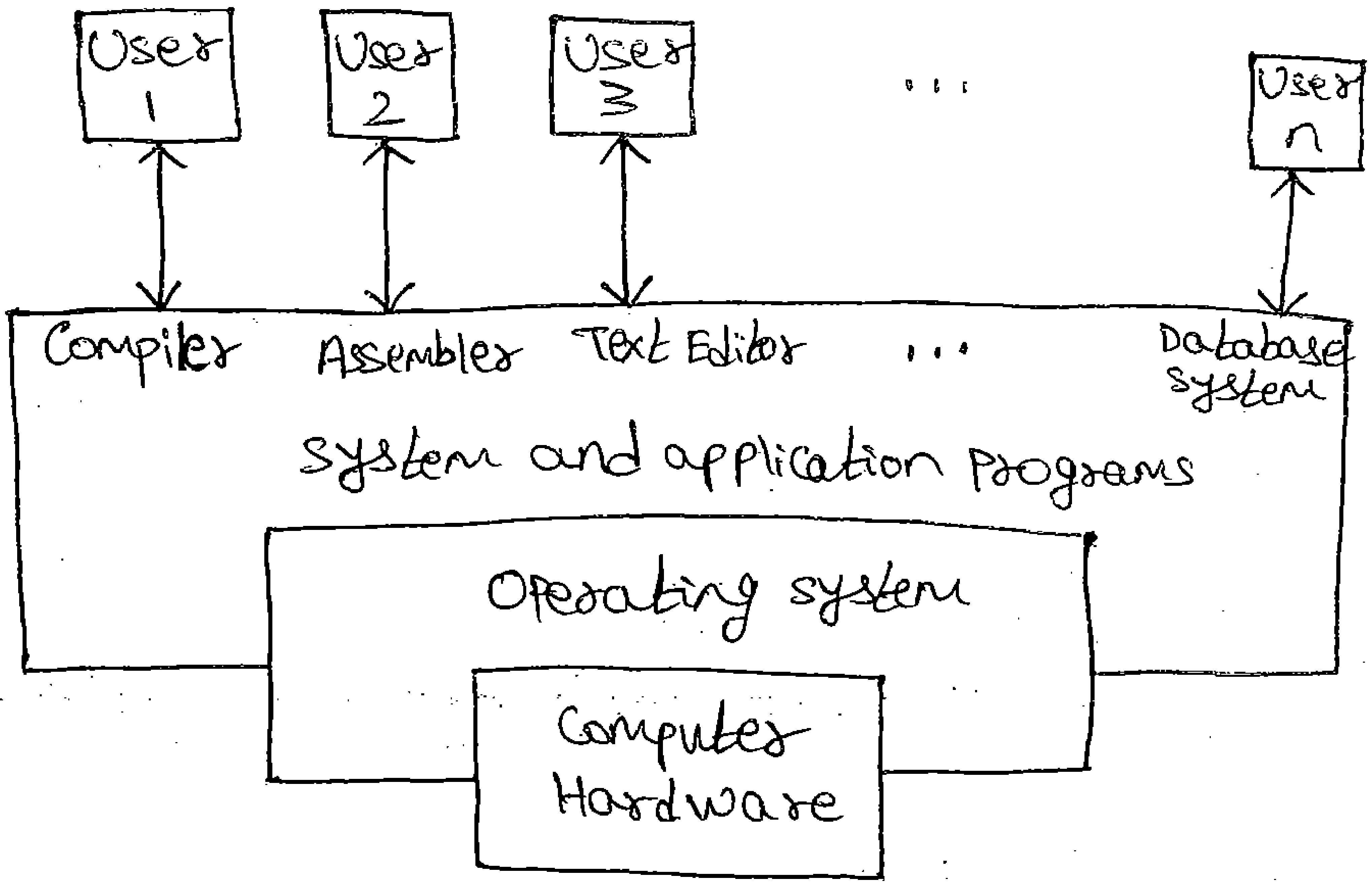
(i) Some users may use PC's - In this, the system is designed so that only one user can utilize the resources and mostly for ease of use where the attention is mainly on performances and not on the resource utilization.

(ii) Some users may use a terminal connected to a mainframe (or) a minicomputer.

iii) Other users may access the same computer through other terminals. These users may share resources and exchange information. — In this, OS is designed to maximize resource utilization - so that all available CPU time, memory & I/O are used efficiently.

iv) Other users may sit at workstations, connected to the networks of other workstation & servers - In this, OS is designed to

Abstract view of the components of a computer system.



System View - From the computer's point of view, the OS is the program most intimately involved with the hardware.

) We can view an OS as a resource allocator - A computer system has many resources that may be used to solve a problem: CPU Time, memory space, I/o devices The OS acts as a manager of these resources. The OS must decide how to allocate these resources to programs and the users so that it can be operate the computer system efficiently and fairly.

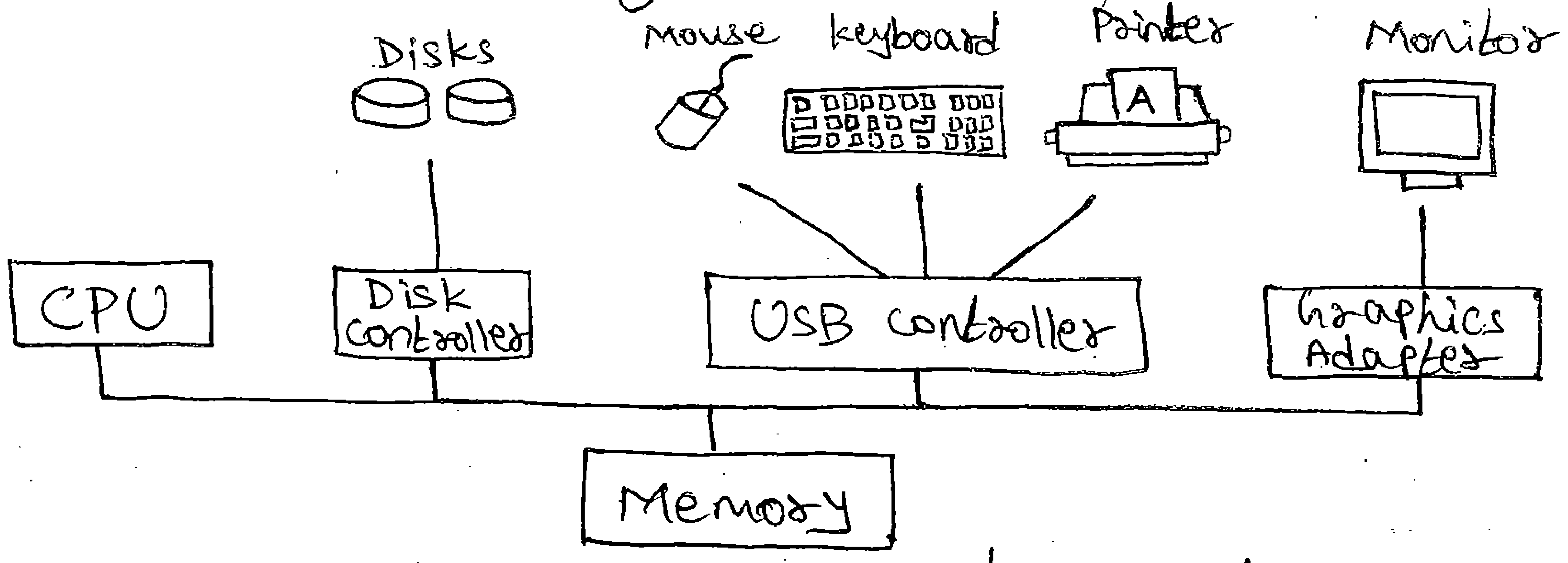
) A different view of an OS is that it need to control various I/o devices and user programs i.e, an OS is a control program used to manage the execution of user program to prevent errors and improper use of the computer.

A more common definition, the one that we usually follow, is that the operating system is the one program running at all times on the computer - usually called the kernel.

Two other types of programs, along with the kernel:

System programs - associated with the OS but not part of the kernel

Computer-System Organization



A modern computer system

- Bootstrap program - Initial program loaded at power-up (or) reboot.
  - Typically stored in ROM (or) EEPROM (Electrically erasable programmable read-only memory), generally known as Firmware.
  - Initializes all aspects of system, from CPU registers to device controllers to memory contents.
  - Loads operating system kernel and starts execution.

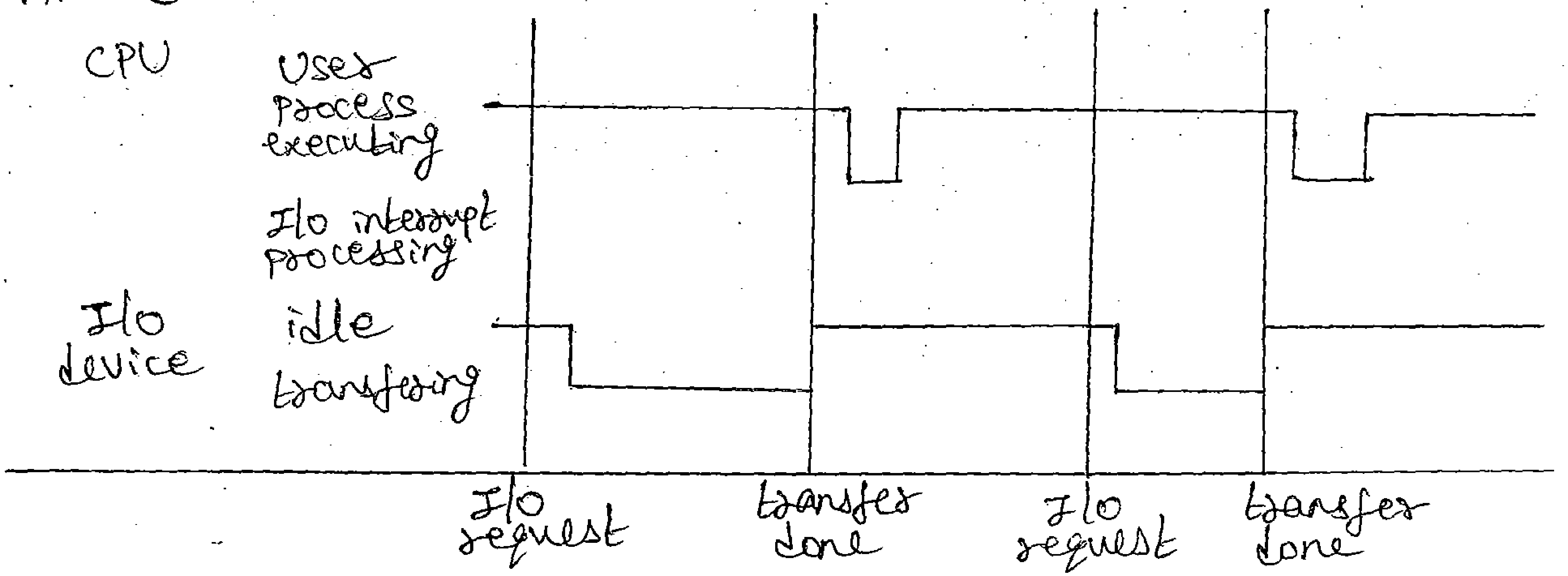
Computer-System Operation

- One (or) more CPUs, device controllers connected through common bus providing access to shared memory.
- Concurrent execution of CPUs and devices competing for memory cycles.
- I/O devices and the CPU can execute concurrently.
- Each device controller is in charge of a particular device type. Ex:- Disk Drives, audio devices and video displays.
- Each device controller has a local buffer.
- CPU moves data from/to main memory to/from local buffers.
- I/O is from the device to local buffers of controller.

device controller informs CPU that it has finished its operation

→ Common Functions of Interrupts.

- * Interrupt transfers control to the interrupt service routine generally through the interrupt vector, which contains the addresses of all the service routines.
- * Interrupt architecture must save the address of the interrupted instruction.
- * Incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt.
- * A <u>trap</u> is a software-generated interrupt caused either by an error (or) a user request.
- * An OS is interrupt driven.



Interrupt time line for a single process doing output

* Storage structure

> The CPU can load instructions only from memory, so any programs to run must be stored there.

> General-purpose computers run most of their programs from rewriteable memory, called Main Memory (RAM).

> Main Memory is implemented in a semiconductor technology called Dynamic Random-Access Memory (DRAM).

> Ideally, we want the programs and data to reside in main memory permanently. This arrangement usually is not possible for the following two reasons:

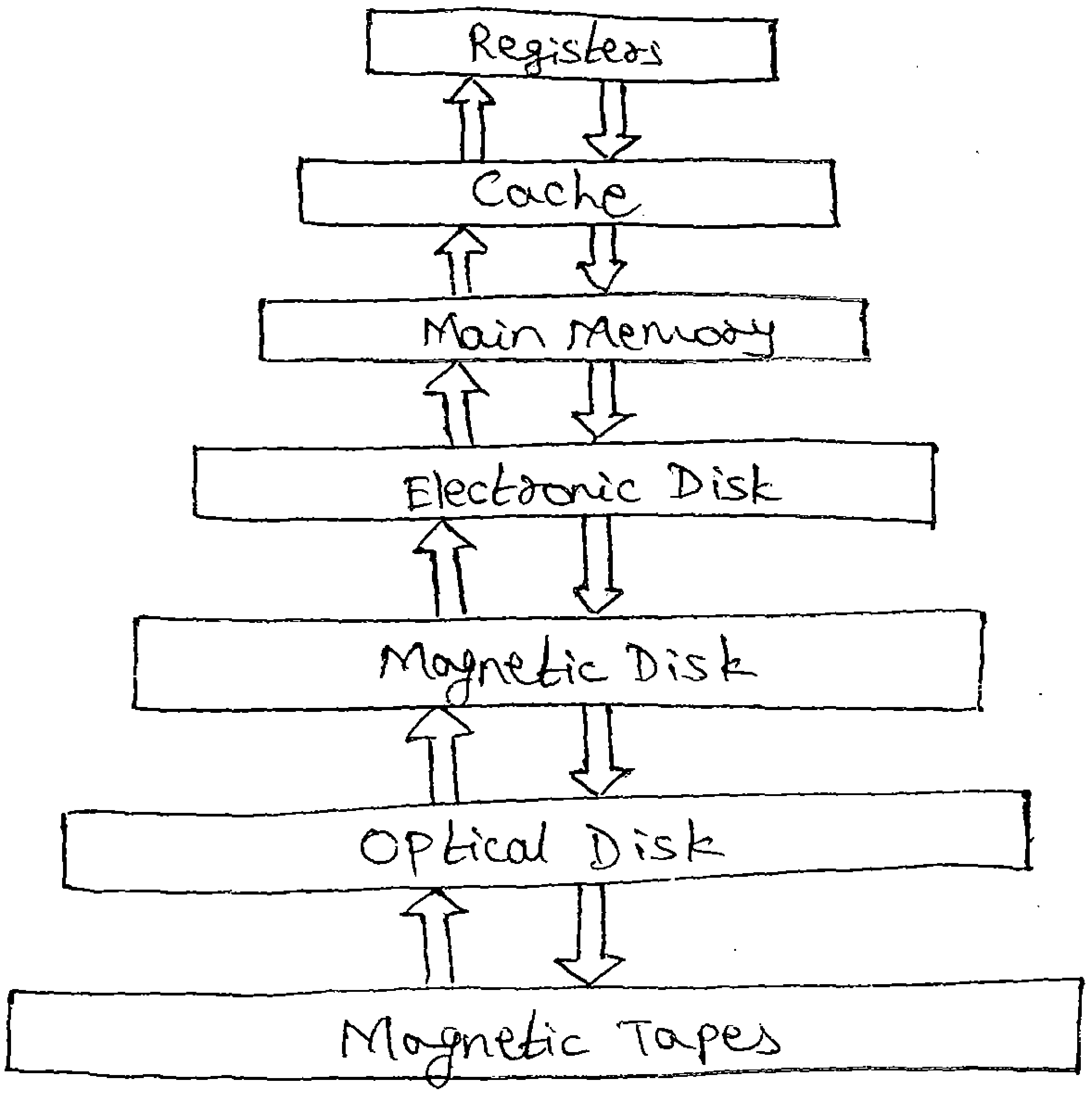
- * Main memory is usually too small to store all needed programs and data permanently.
- * Main memory is a volatile storage device that loses its contents when power is turned off (or) otherwise lost.

> Thus, most computer systems provide secondary storage as an extension of main memory.

> Main requirement for secondary storage is that it be able to hold large quantities of data permanently [non-volatile]

› The most common secondary-storage device is a magnetic disk.

- ≫ Provides storage for both programs and data.
- ≫ Rigid metal (or) glass platters covered with magnetic recording material.
- ≫ Disk surface is logically divided into tracks, which are subdivided into sectors.
- ≫ The disk controller determines the logical interaction between the device and the computer.



Storage Device Hierarchy

→ storage systems organized in hierarchy according to:

- * speed
- * Cost
- * Volatile.

→ The higher levels are expensive, but they are fast.

→ As we move down the hierarchy, the cost per bit generally decreases, whereas the access time generally increases.

→ storage systems in a hierarchy above the electronic disk are volatile, whereas those below are non-volatile.

→ Electronic disk can be designed to be either volatile (or) non-volatile.

- * During normal operation, it stores data in a large DRAM array, which is volatile.
- * It contains hidden magnetic hard disk & a battery for backup power.
- * If external power is interrupted, the electronic-disk controller copies the data from RAM to the magnetic disk.
- * when external power is restored, the controller copies the data back into DRAM.

→ Another form of electronic disk is flash memory.

→ Caching - copying information into faster storage system main memory can be viewed as a cache for secondary storage.

I/O structure

→ After I/O starts, control returns to user program only upon I/O completion.

- » wait instruction idles the CPU until the next interrupt.
- » wait Loop (contention for memory access)

≫ At most one I/O request is outstanding at a time no simultaneous I/O processing.

› After I/O starts, control returns to user program without waiting for I/O completion.

≫ System call – Request to the OS to allow user to wait for I/O completion.

≫ Device-status table – Contains entry for each I/O device indicating its type, address & state.

≫ OS indexes into I/O device table to determine device status and to modify table entry to include interrupt.

> Direct Memory Access (DMA) structure.

* Used for high-speed I/O devices able to transmit information at close to memory speeds.

* Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.

* Only one interrupt is generated per block, rather than the one interrupt per byte.

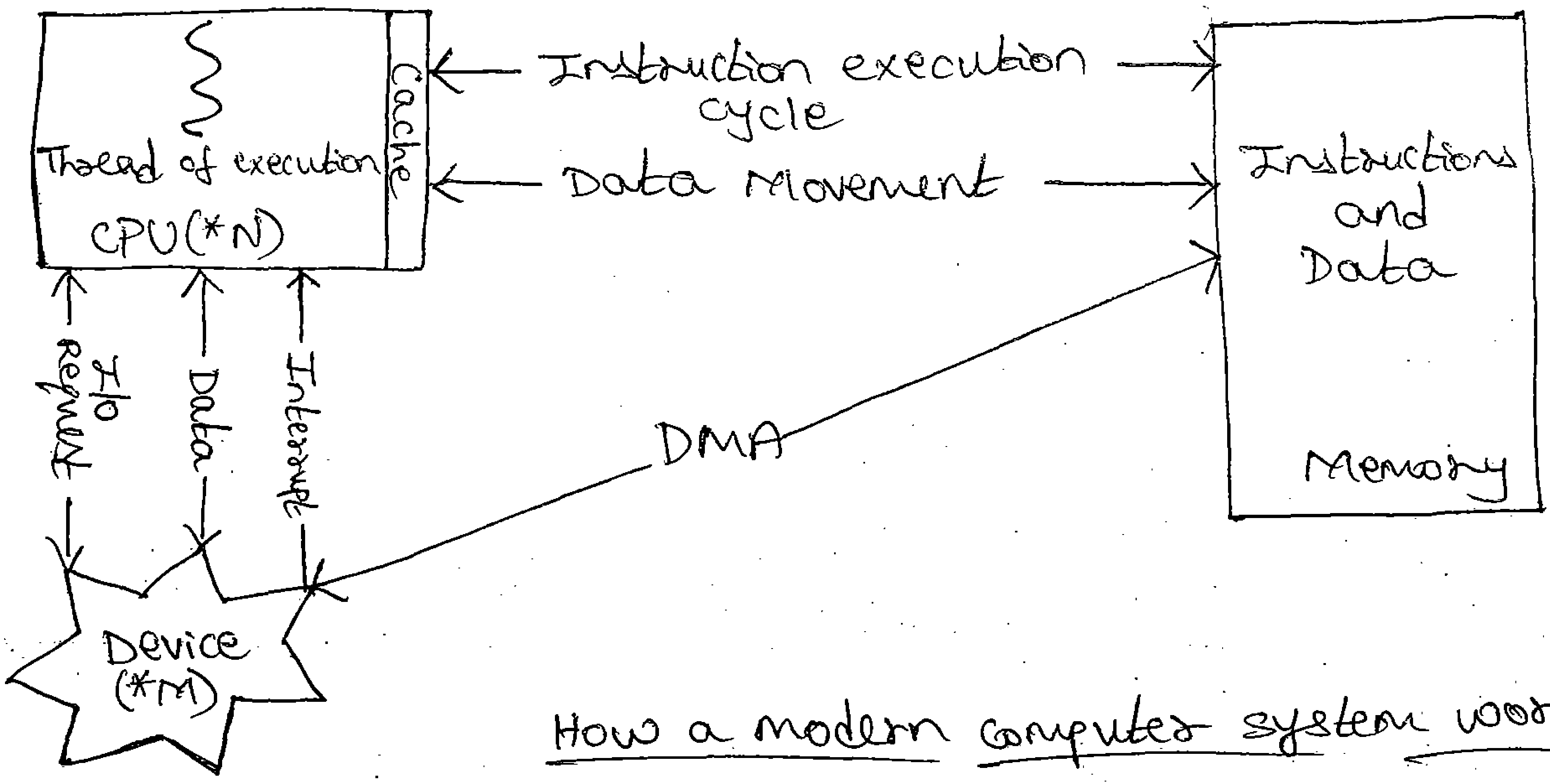
› Computer-System Architecture

* Single-processor systems

→ Most systems use a single general-purpose processor (from PDAs through mainframes).

→ There is one main CPU capable of executing a general-purpose instruction set, including instructions from user processes.

→ Most systems have special-purpose processors as well.



How a modern computer system works.

Multiprocessor Systems

→ Growing in use and importance.
→ Also known as Parallel systems, tightly-coupled systems.
→ Three main advantages:

* Increased throughput - By increasing the number of processors we can get more work done in less time. When multiple process cooperate on task, a certain amount of overhead is incurred in keeping all parts working correctly.

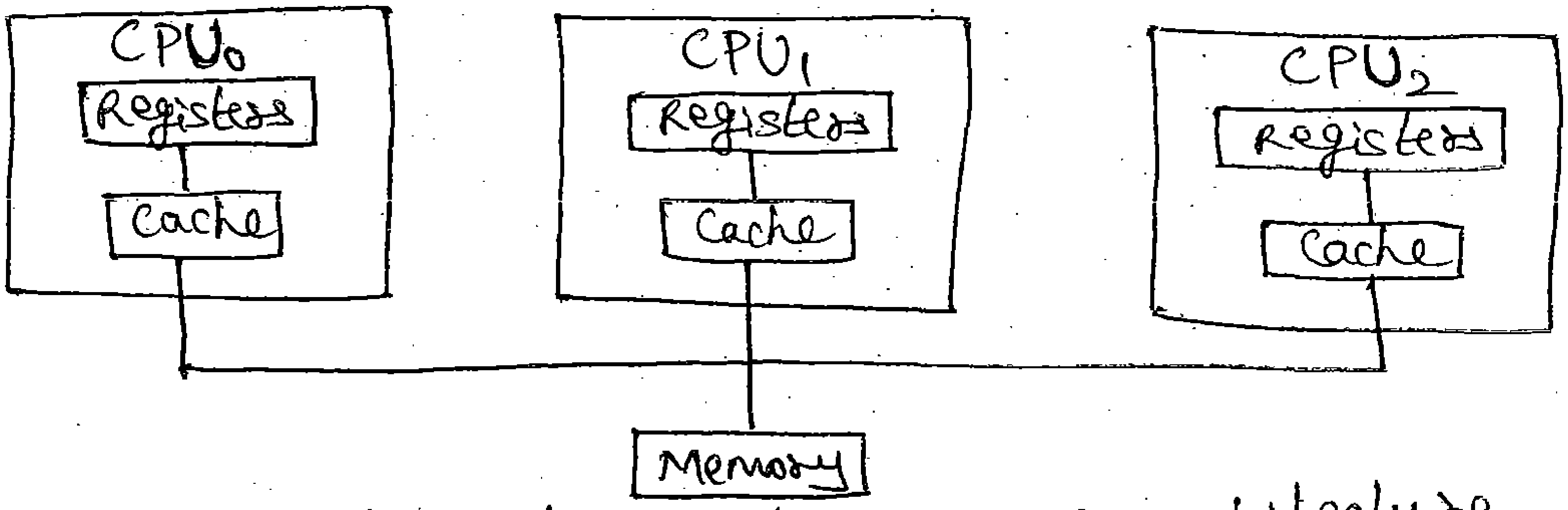
* Economy of scale - Multiprocessor system can save more money than multiple single processor, since they share peripherals, mass storage and power supplies. If many programs operate on same data, they will be stored on one disk & allprocessors can share them instead of maintaining data on several systems.

* Increased Reliability - If a program is distributed properly on several processors, then the failure of one processor will not halt the system but it only slows down.

→ Two types:

(i) Asymmetric multiprocessing - Each processors is assigned a specific task. It uses a master-slave relationship. A master processor controls the system. The master processors schedules and allocates work to slave processors.

(ii) symmetric multiprocessing - Each processors runs an identical copy of OS and they communicate with one another as needed. All the CPU shares the common memory.

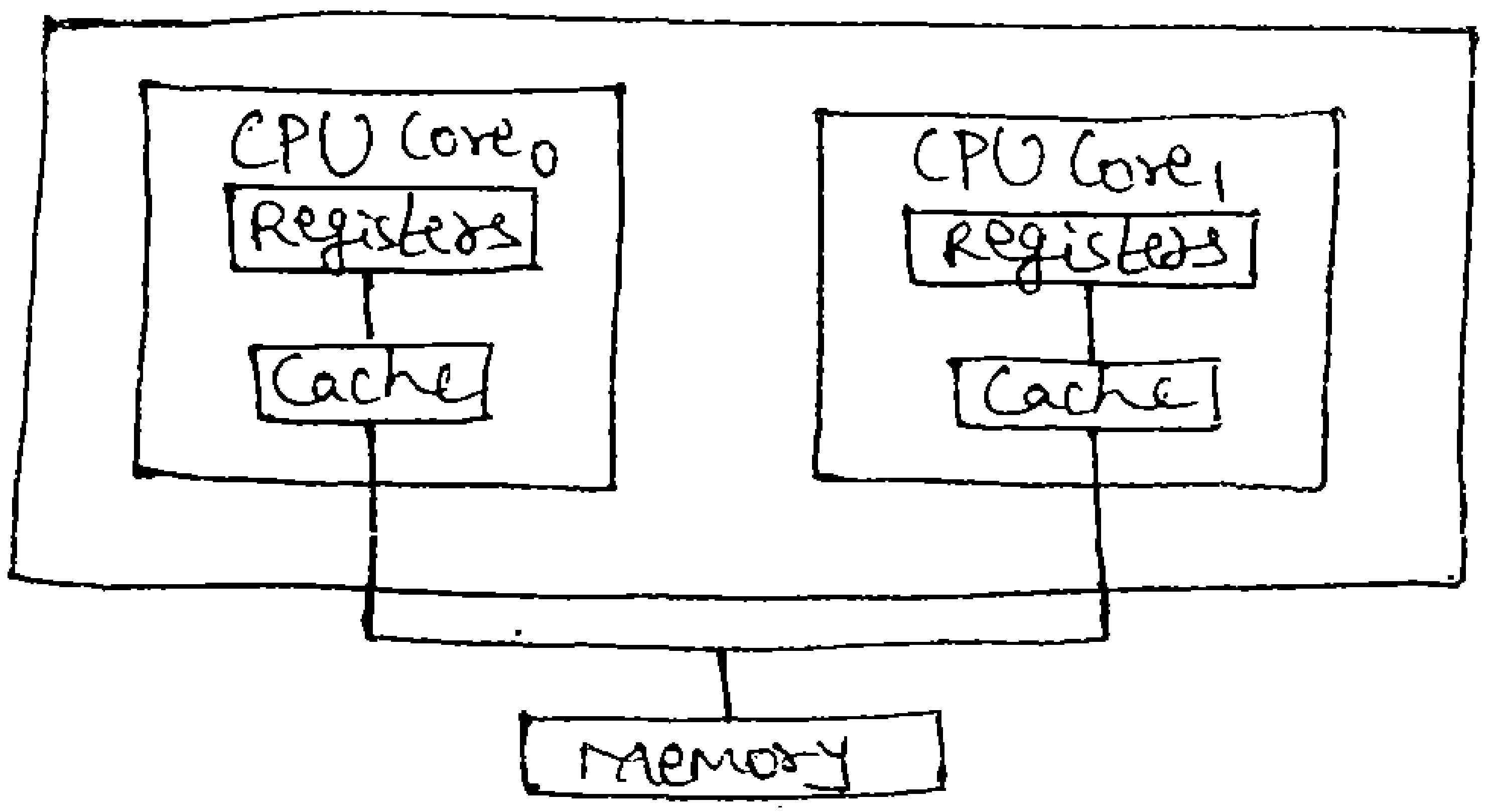


Symmetric multiprocessing architecture

> The difference between symmetric & asymmetric multiprocessing may result from either hardware (or) software.

> special hardware can differentiate the multiple processors, (or) the software can be written to allow only one master and multiple slaves.

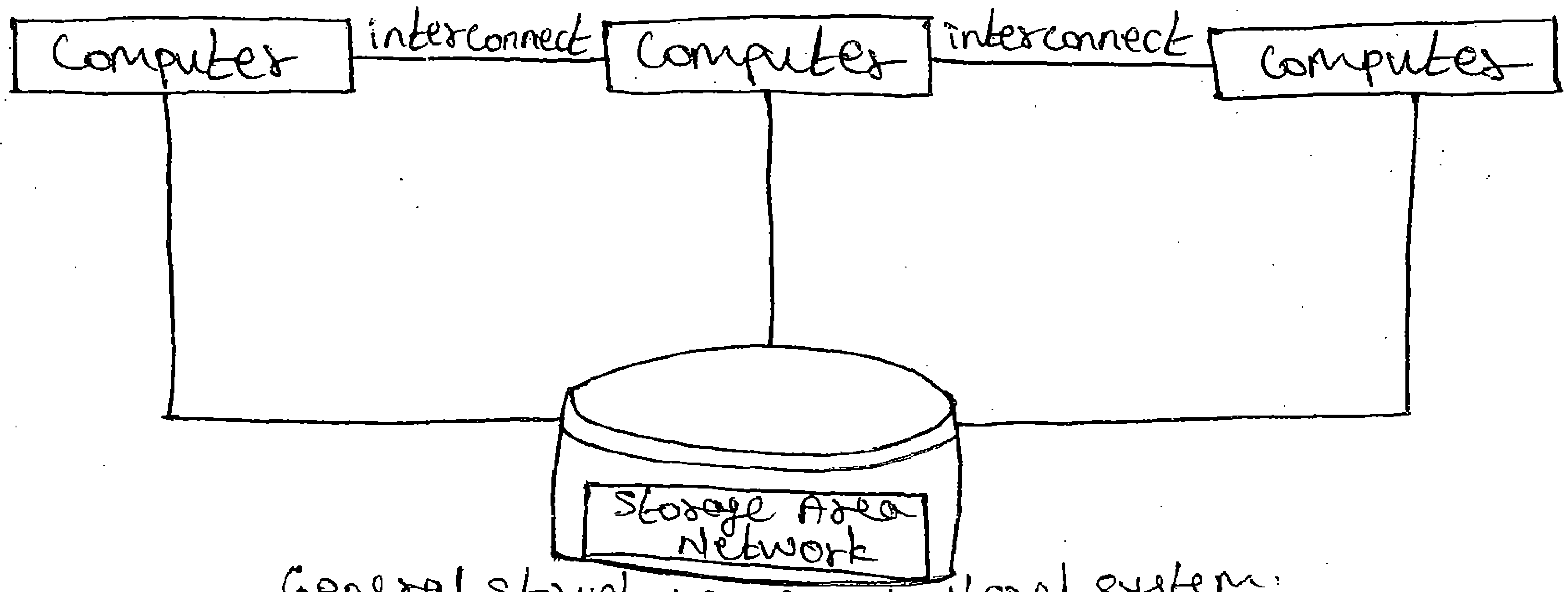
> Example: Sun's OS, SunOS Version 4 provided asymmetric multiprocessing, whereas version 5 (solaris) is symmetric on the same hardware.



Clustered Systems

→ Like multiprocessor systems, but multiple systems working together.

- » Usually sharing storage via storage-area network (SAN)
- » Provides a high-availability service which survives failures.
  - (i) Asymmetric clustering - Has one machine in hot-standby mode while the other is running the applications.
  - (ii) Symmetric clustering - Has multiple nodes running applications, monitoring each other.
- » Some clusters are ~~a~~for High-performance computing (HPC)
- » Applications must be written to use parallelization

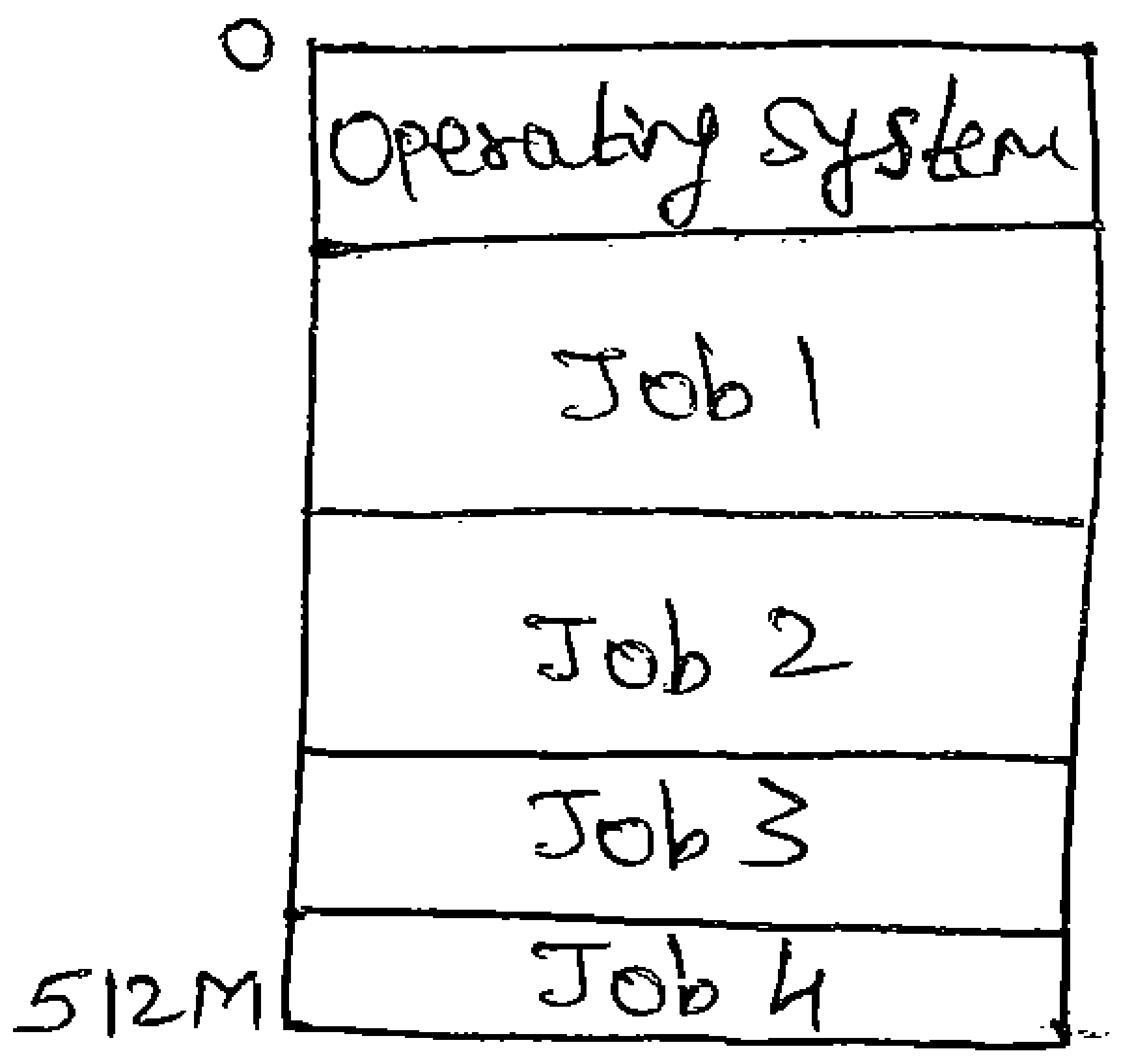


General structure of a clustered system.

Operating-system structure

→ Multiprogramming needed for efficiency.

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



Memory Layout for a multiprogramming system

Time sharing (or multitasking) - is a 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.
- * If several jobs ready to run at the same time ⇒ CPU scheduling.
- * If processes don't fit in memory, swapping moves them in and out to run.
- * Virtual memory allows execution of processes not completely in memory.

Operating-System Operations

Modern OS are interrupt driven by hardware.

Software error (or) request creates exception (or) Trap.

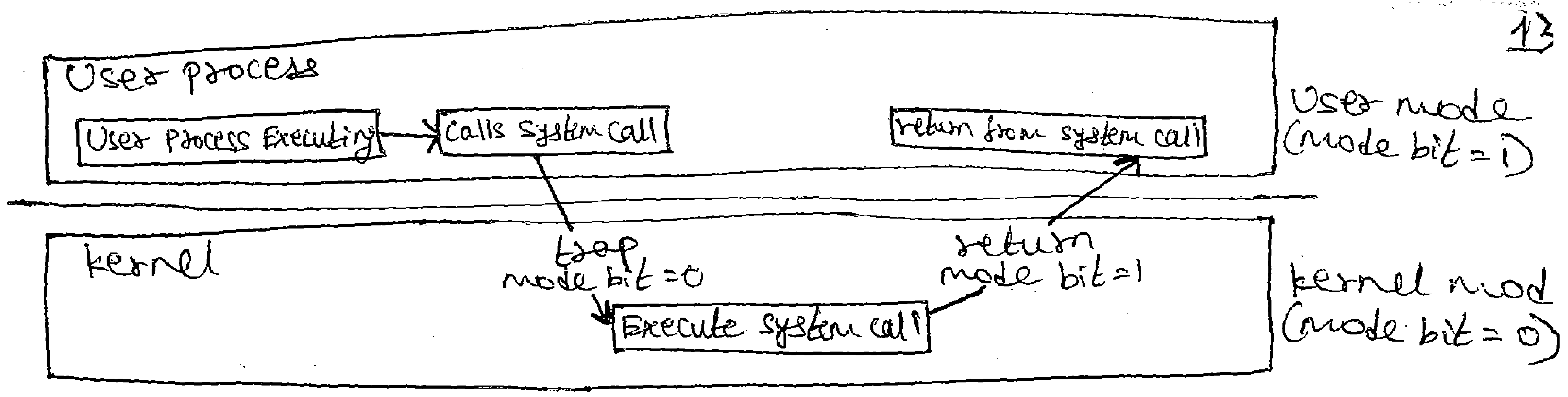
Ex:- Division by zero, request for OS service.

Other process problems include infinite loop, processes modifying each other (or) the OS.

Dual-Mode operation

Allows OS to protect itself and other system Components.

Two separate modes of operation: User mode and



Transition from user to kernel mode.

→ Mode bit provided by hardware

* Provides ability to distinguish when system is running user mode (or) kernel Mode
* Some instructions designated as privileged, only executable in kernel mode.
* System call changes mode to kernel, return from call resets it to user.

Timer

→ Timer to prevent infinite loop/process hogging resource

* set interrupt after specific period.
* OS decrements counter
* When counter reaches zero, an interrupt occurs.
* set up before scheduling process to regain control (or) terminate program that exceeds allotted time.

Process management

→ A process is a program in execution.

→ It is a unit of work within the system.

→ program is a passive entity, process is an active entit

→ Process needs resources to accomplish its task

>> CPU, memory, I/O devices, files.

>> Various initialization data (input) may be passed along.

→ Process termination

- A single-threaded process has one program counter specifying location of next instruction to execute.
  - ≫ Process executes instructions sequentially, one at a time, until completion.

- Multi-threaded process has multiple program counters, each pointing to the next instruction to execute for a given thread.

All the processes (those execute system code / user code) can potentially execute concurrently by multiplexing on a single CPU.

- OS is responsible for the following activities of the Process management:

* Scheduling processes and threads on the CPUs.
* Creating and deleting both user and system processes.
* Suspending and resuming processes.
* Providing mechanisms for process synchronization.
* Providing mechanisms for process communication.
* Providing mechanisms for Deadlock handling.

Memory Management

- All data in memory before and after processing.
- All instructions in memory in order to execute.

Memory management determines what is in memory when,

≫ optimizing CPU utilization & computer response to users.

Memory management activities:

* keeping track of which parts of memory are currently

* Deciding which processes (or parts thereof) and data to move into and out of memory.

* Allocating and deallocating memory space as needed.

Storage Management - To make computer system convenient for

-> OS provides uniform, logical view of information storage

-> Abstracts physical properties to logical storage unit - File

-> Maps files into physical media and accesses these files via the storage devices. Example:- Book and Ebook.

File-system Management

-> Most visible components of an OS.

-> stores information on several types of physical media like Magnetic Disks, Magnetic Tapes, Optical Disks etc.

-> A file is logical collection of information.

-> File consists of both program & data. Data files may be numeric, alphabets, alphanumeric (or) binary.

-> Files normally organized into directories.

-> OS activities include:

* Creating & deleting files.

* Creating & deleting directories to organize files.

* Supporting primitives for manipulating files and directorie

* Mapping files onto secondary storage.

* Backing up files on stable (non-volatile) storage media.

Mass-storage Management

-> Usually disks used to store data that does not fit in main memory (or) data that must be kept for a "long" period of time.

-> Proper Management is of central importance.

OS activites:

- Free-space management
- Storage allocation
- Disk scheduling

ome storage need not be fast:

Tertiary storage includes optical storage, magnetic tape.

Still must be managed - by OS (or) applications.

Varies between WORM (write-once, read-many-times) & RW (read-write).

ching

an important principle, performed at many levels in computer (hardware, OS, software).

formation in use copied from slower to faster storage mporarily.

ster storage (cache) checked first to determine if information there:

» If it is, information used directly from the cache (fast)

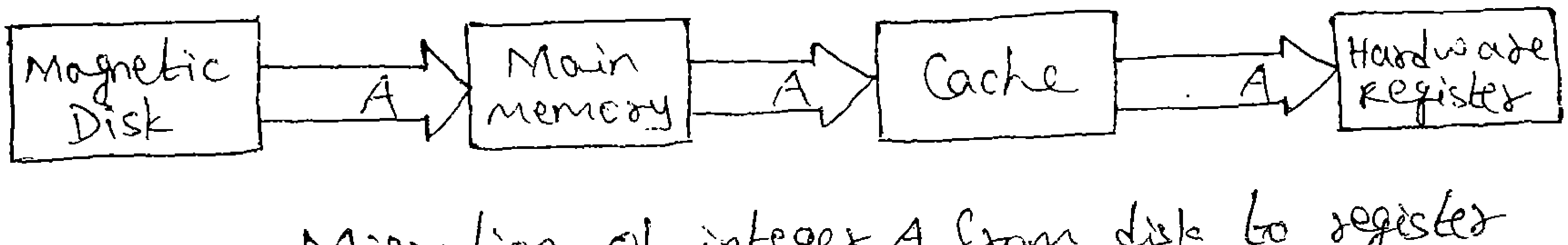
» If not, data copied to cache & used there.

he smaller than storage being cached

› Cache management important design problem.

› Cache size & replacement policy.

lti-tasking environments must be careful to use most ent value, no matter where it is stored in the storage rarchy.



Migration of integer A from disk to register

→ Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache.

→ Distributed environment situation even more complex.
  » several copies of a datum can exist.

I/O Systems

→ One purpose of OS is to hide peculiarities of hardware devices from the user.

→ I/O subsystem consists of several components:
  * A memory-management component that includes buffering, caching & spooling (overlapping of output of one job with input of other jobs).
  * A general device-driver interface.
  * Drivers for specific hardware devices.

---

⑪ Protection and Security

→ Protection – Any mechanism for controlling access of processes (or) users to resources defined by the OS

→ Security – Defense of the system against internal and external attacks.
  » Huge range, including denial-of-service, worms, viruses, identity theft, theft of service.

→ Systems generally first distinguish among users, to determine who can do what:
  * User identities (user IDs, security IDs) include name and associated number, one per user.
  * User ID then associated with all files, processes of that user to determine access control.

* Group identifier (group ID) allows set of users to be defined & controls managed, then also associated with process, file.

* Privilege escalation allows user to change to effective ID with more rights.

) Distributed Systems

> Collection of separate, possibly heterogeneous, systems networked together.

> A network is a communication path b/w two (or) more systems.

* Local Area Network (LAN) - room, floor, building
* Wide Area Network (WAN) - buildings, cities, countries
* Metropolitan Area Network (MAN) - Link buildings within a city
* Wireless/small-Area network (WAN) - BlueTooth

> Network OS provides features between systems across network.

> Communication scheme allows s/ms to exchange messages

> Illusion of a single system.

) Special-Purpose Systems.

Real-Time Embedded Systems

> Most prevalent form of computers.

> Vary considerable, special purpose, limited purpose OS, real-time OS.

Multimedia systems

> streams of data must be delivered according to time restrictions - frames of video.

Handheld systems

> PDAs - personal Digital Assistants, smart phones, limited CPU, memory, power

⑪ Computing Environments

Traditional Computing

→ Blurring over time

→ Office Environment

  * PCs connected to a network, terminals attached to mainframe (or) minicomputers providing batch and timesharing.

  * Today, portals allowing networked and remote systems access to same resources.

→ Home Networks

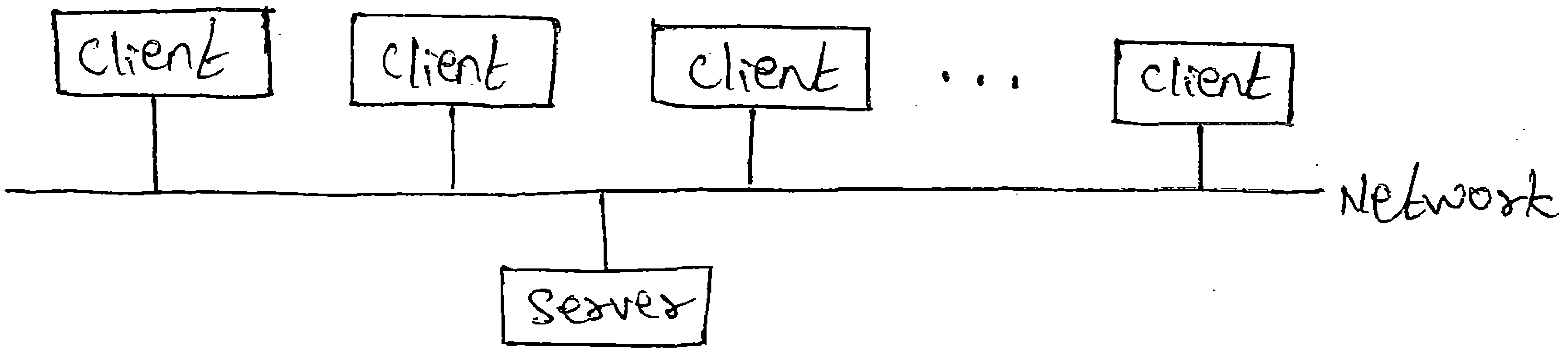
  * Used to be single system, then modems.

  * Now firewalled, networked.

Client-Server Computing

→ Dumb terminals supplanted by smart PCs.

→ many systems now servers, responding to requests generated by clients



General structure of a client-server system.

  * Compute~~r~~-server system provides an interface to client to request services (i.e, database).

  * File-server system provides interface for clients to store and retrieve files.

Peer-to-Peer Computing (P2P)

→ Another model of distributed systems.

→ P2P does not distinguish clients and servers

  * Instead all nodes are considered peers.

* May each acts as client, server (or) both
* Node must join P2P network.
  >> Registers its service with central lookup service on network (or)
  >> Broadcast request for service & respond to requests for service via Discovery Protocol.

> Examples include Napster & Gnutella.

---

* Web-Based Computing

> Web has become ubiquitous

> PCs most prevalent devices.

> More devices becoming networked to allow web access.

> New category of devices to manage web traffic among similar servers: Load Balancers.

> Use of OS like Windows 95, client-side, have evolved into Linux & windows XP, which can be clients & servers.

> Open-Source Operating Systems.

> Open-Source OS are those made available in source-code format rather than as compiled binary code (closed-source)

> Ex:- Linux:- Open-Source OS,
   Microsoft Windows:- Closed-Source OS.

. Counter to the copy protection & Digital Rights management (DRM) movement.
Started by Free software Foundation (FSF), which has "copyleft" GNU public License (GPL).
Examples include GNU/Linux & BSD UNIX (including core of Mac OS X) & many more.

Chapter 2: System Structures

Objectives: * To describe the services, an operating system provides to users, processes & other systems.

* To discuss the various ways of structuring an operating system.

* To explain how operating systems are installed and customized and how they boot.

Operating-System Services

→ An OS provides an environment for execution of programs and services to programs and users.

→ One set of OS services provides functions that are helpful to the user:

* User interface - Almost all operating systems have a user interface (UI)

  >> Varies between DTrace command-line interface (CLI) Graphical user interface (GUI), Batch interface.

* program execution - The system must be able to load a program into memory and to run that program, end execution, either normally (or) abnormally (indicating erro

* I/O operations - A running program may require I/O, which may involve a file (or) an I/O device

* File-system manipulation - The file system is of particular interest. Programs need to read & write files & directories, create & delete them, search them list file Information, permission management.

* Communications - Processes may exchange information, on the same computer (or) between computers over

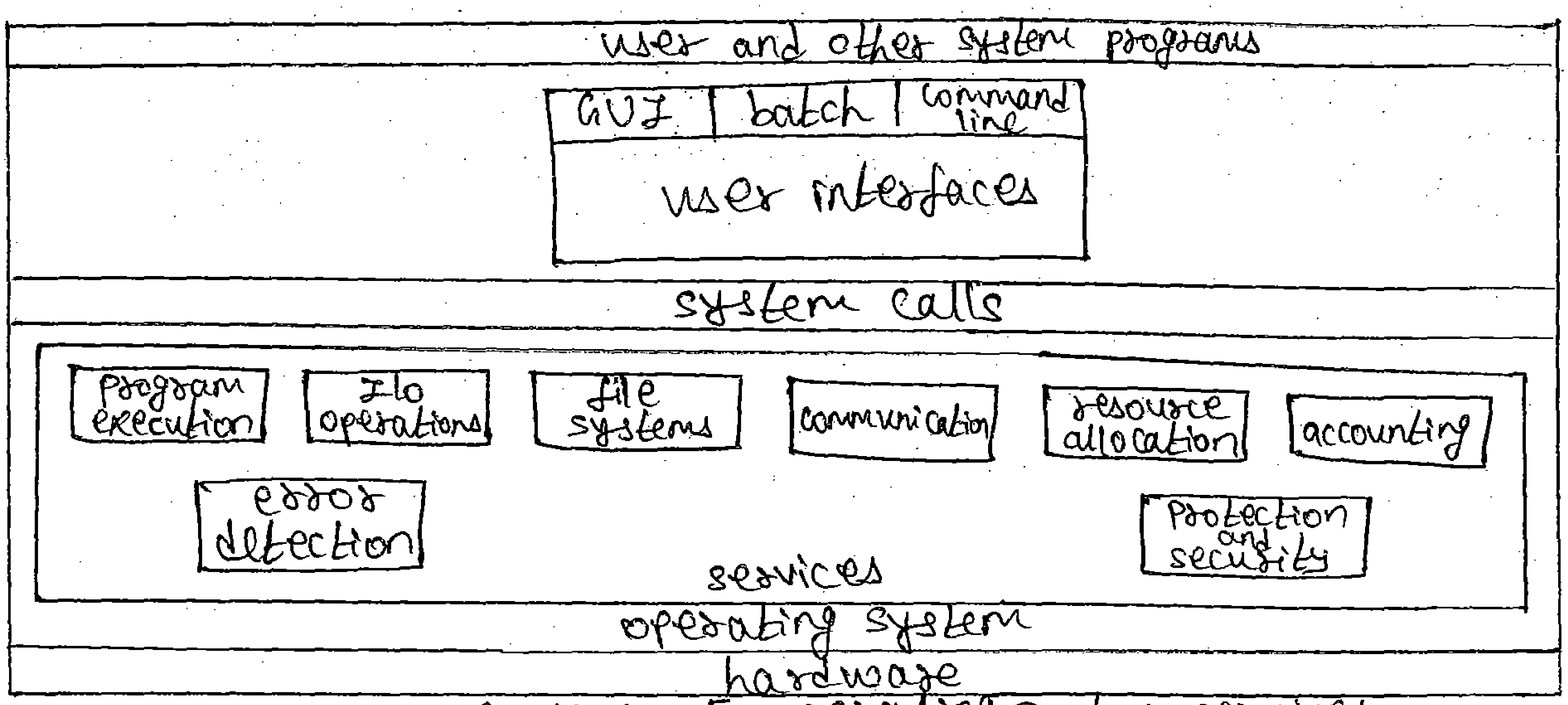
>> Communications may be via shared memory (or) through message passing (Packets moved by the OS).

* Error Detection - The OS needs to be constantly aware of possible errors.

>> may occur in the CPU & memory hardware, in I/O devices, in user program.

>> for each type of error, OS should take the appropriate action to ensure correct & consistent computing.

>> Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system.



A view of operating system services

Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing.

* Resource allocation - When multiple users (or) multiple jobs running concurrently, resources must be allocated to each of them.

>> Many types of resources - some (such as CPU cycles, main memory, & file storage) may have special allocation code, others (such as I/O devices) may have general request & release code.

* Accounting - To keep track of which users use how much & what kinds of computer resources.

* Protection and security - The owners of information stored in a multiuser (or) networked computer system may want to control use of that information, concurrent processes should not interfere with each other.

  > Protection involves ensuring that all access to system resources is controlled.

  > Security of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts.

  > If a system is to be protected & secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

---

⑩ User Operating-System Interface

Command Interpreter

> Command Line Interface (CLI) (or) command interpreter allows direct command entry

> Sometimes implemented in kernel, sometimes by systems program.

> Sometimes multiple flavors implemented - shells

> Ex:- Bourne shell, C shell, Bourne-Again shell, korn shell etc.

> Primarily fetches a command from user and executes it

  > Sometimes commands built-in, sometimes just names of programs.

Graphical User Interfaces (GUI)

> User friendly desktop metaphor interface.

> Usually mouse, keyboard, & monitor.

> Icons represent files, programs, actions etc.

→ Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory; known as folder).

→ Invented at Xerox PARC in early 1970s.

→ First GUI appeared on the Xerox Alto computer in 1973.

→ Many systems now include both CLI & GUI interfaces.

* Microsoft Windows 10 is GUI with CLI "command" shell.
* Apple Mac OS X as "Aqua" GUI interface with UNIX kernel underneath & shells available.
* Solaris is CLI with optional GUI interfaces (Java Desktop, K Desktop Environment (KDE)).

---

2) System Calls

> Provides an interface to the services made available by an OS.

> Typically written in a high-level language (C or C++)

> Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use

› Three most common APIs are:

>> Win32 API for Windows.

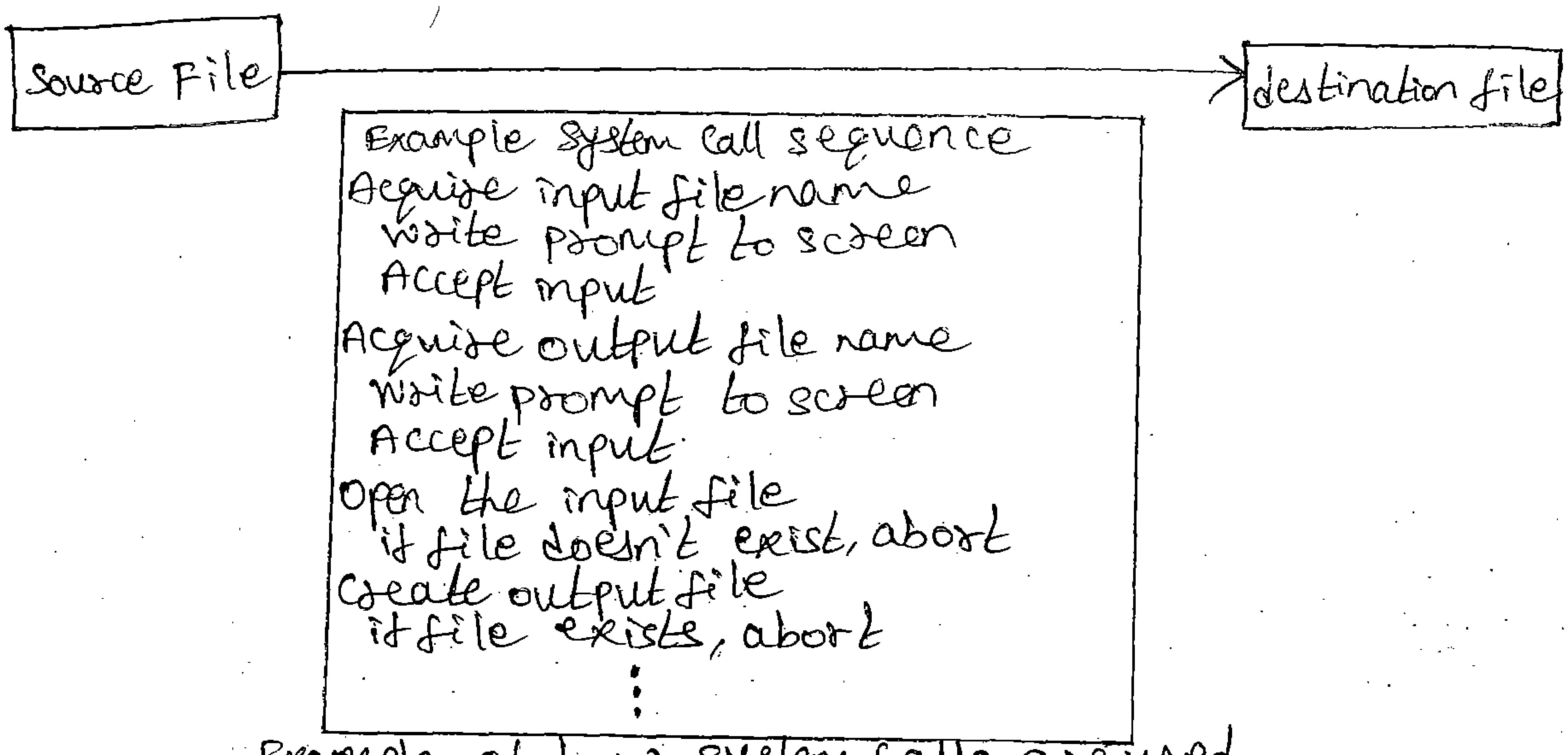
>> POSIX API for POSIX-based systems (UNIX, Linux, Mac OS X).

>> Java API for Java Virtual Machine (JVM).

Why use APIs rather than system calls?

* Program portability
* System calls can often be more detailed & difficult to work.

System call sequence to copy the contents of one file to another file.



```
Example system call sequence
Acquire input file name
  write prompt to screen
  Accept input
Acquire output file name
  write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
        ⋮
```

Example of how system calls are used

→ Example of standard API.

* Consider the ReadFile() function in the Win32 API.

- A function for reading from a file.

```
return value
     ↓
   BOOL    ReadFile c      (HANDLE        file,         ⎤
                ↑           LPVOID        buffer,       ⎥
                            DWORD         bytes To Read,⎥ Parameters
             function name  LPDWORD       bytes Read,   ⎥
                            LPOVERLAPPED  ovl);         ⎦
```

→ A description of the parameters passed to ReadFile()

* HANDLE file - the file to be read

* LPVOID buffer - a buffer where the data will be read into and written from.

* DWORD bytesToRead - the number of bytes to be read into the buffer.

* LPDWORD bytesRead - the number of bytes read during the last read.

* LPOVERLAPPED ovl - indicates if overlapped I/O is being used.

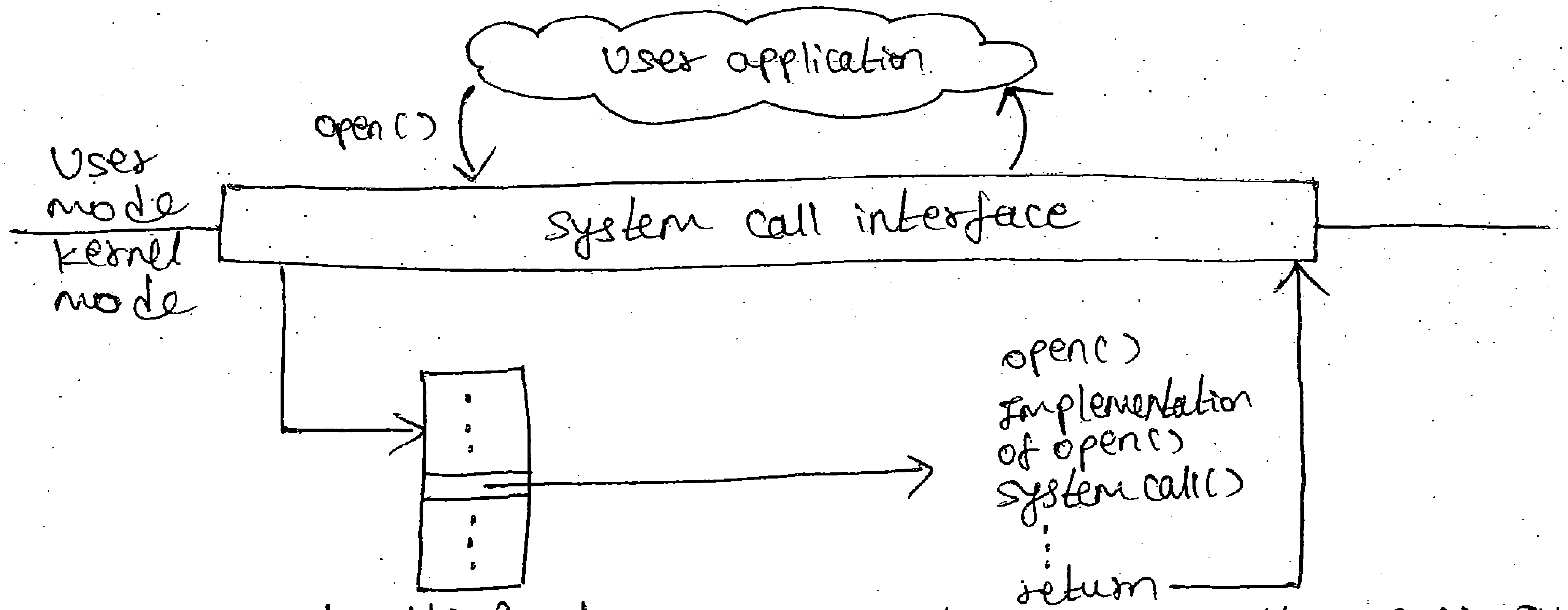
→ System Call Implementation.

* Typically, a number associated with each system call.

* System-call interface maintains a table indexed according

* The system call interface invokes intended system call in OS kernel & returns status of the system call and any return values.

* The caller need know nothing about how the system call is implemented.
  - >> Just needs to obey API & understand what OS will do as a result call.
  - >> Most details of OS interface hidden from programmer by API. - Managed by run-time support library.

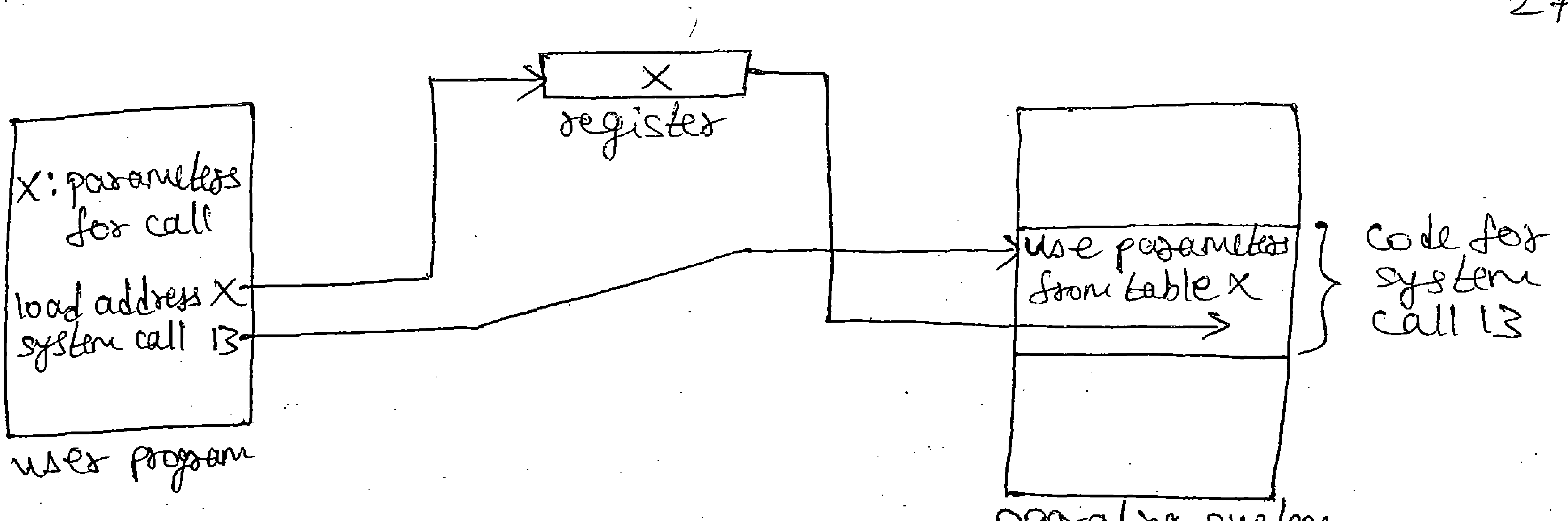


The handling of a user application invoking the open() system call.

→ System call Parameter Passing

* Often, more information is required than simply identity of desired system call.

* Three general methods used to pass parameters to the OS:
  - >> Simplest - pass the parameters in registers.
  - >> Parameters stored in a block, or table, in memory & address of block passed as a parameter in a register.
  - >> Parameters placed, or pushed, onto the stack by the program and popped off the stack by the OS.



passing of parameters as a table

⑥ Types of System Calls

(i) Process Control

- * end, abort
- * load, execute
- * create process, terminate process
- * get process attributes, set process attributes
- * wait for time
- * wait event, signal event
- * allocate and free memory.

i) File management

- * create file, delete file
- * open, close file
- * read, write, reposition
- * get file attributes, set file attributes.

i) Device management

- * request device, release device
- * read, write, reposition
- * get device attributes, set device attributes
- * logically attach (or) detach devices.

) Information maintenance

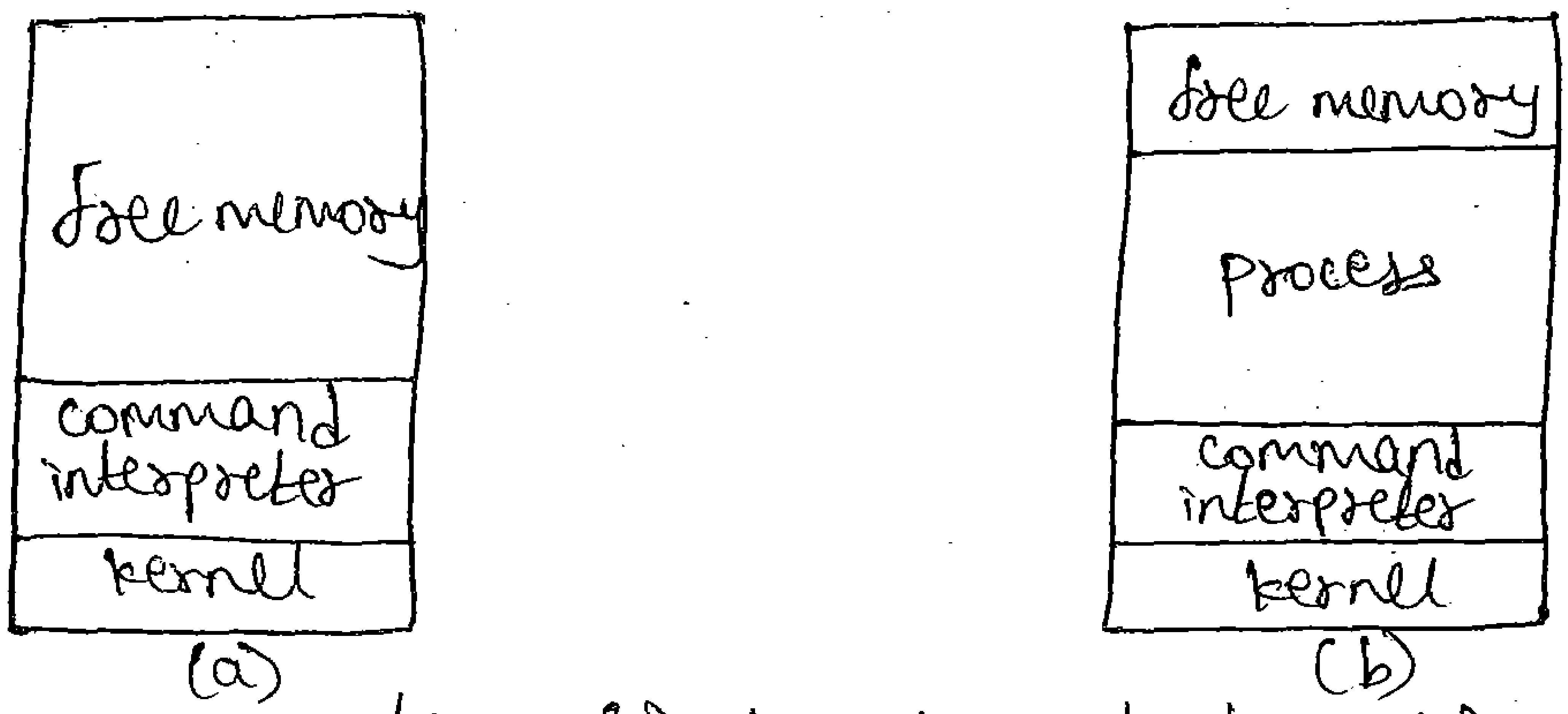
- * get time (or) date, set time (or) date.
- * get system data, set system data.

(v) Communications

- * create, delete communication connection
- * send, receive messages
- * transfer status information
- * attach (or) detach remote devices.

→ Example :- MS-DOS execution.

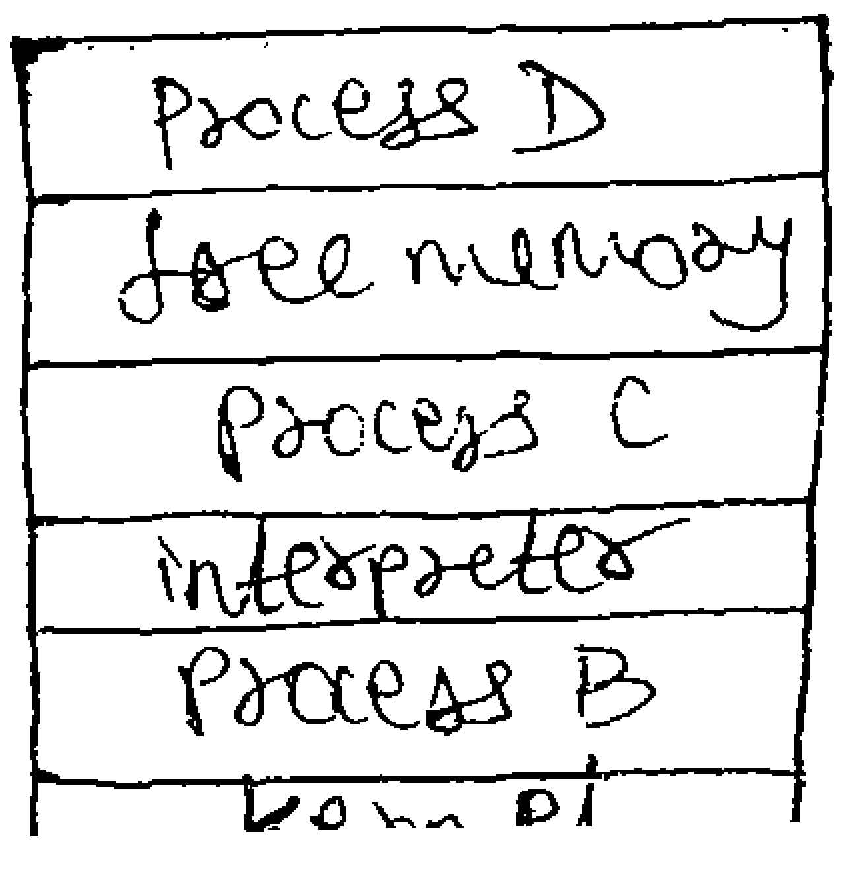
- * Single-tasking
- * shell invoked when system booted
- * simple method to run program
  - ⇒ No process created
- * Single memory space
- * Loads program into memory, overwriting all but the kernel.
- * Program exit → shell reloaded.



MS-DOS execution. (a) At system startup (b) Running a program

› Example :- FreeBSD

- * Unix variant, multitasking.
- * User login → invoke user's choice of shell.
- * Shell executes fork() system call to create process.
  - ⇒ Executes exec() to load program into process.
  - ⇒ shell waits for process to terminate (or) continues with user cmds.
- * Process exits with code of 0 - no error (or) >0 - error code.



⑩ System Programs

→ System programs provide a convenient environment for program development and execution. They can be divided into:

(i) File management - These programs create, delete, copy, rename, print, dump, list & generally manipulate files and directories.

(ii) Status information - Some ask the system for info - date time, amount of available memory, disk space, no. of user.

- * Others provide detailed performance, logging and debugging information.
- * Typically, these programs format and print the output to the terminal (or) other output devices.
- * Some systems implement a registry - used to store and retrieve configuration information.

(iii) File modification - Text editors to create & modify files Special commands to search contents of files (or) perform transformations of the text.

(iv) Programming-language support - Compilers, assemblers, debuggers and interpreters sometimes provided.

(v) Program loading and execution - Absolute loaders, relocatable loaders, linkage editors, over-lay loaders, debugging systems for higher-level and machine language.

(vi) Communications - Provide the mechanism for creating virtual connections among processes, users and computer systems.

- * Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another.

→ In addition, Application programs include web browsers, word processors and text formatters, spreadsheets, compilers &

Operating-system Design and Implementation

→ Design and Implementation of OS not "solvable", but some approaches have proven successful.

→ Internal structure of different OS can vary widely.

→ start by defining goals and specifications.

→ Affected by choice of hardware, type of system

→ User goals and System goals.

- * User goals - OS should be convenient to use, easy to learn, reliable, safe and fast.
- * System goals - OS should be easy to design, implement, and maintain, as well as flexible, reliable, error-free & efficient.

→ Mechanisms and Policies

- * Mechanisms determine how to do something?
- * Policies determine What will be done?
- * The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later.
- * Microkernel-based OS take the separation of mechanism and policy to one extreme by implementing a basic set of primitive building blocks.

→ Implementation

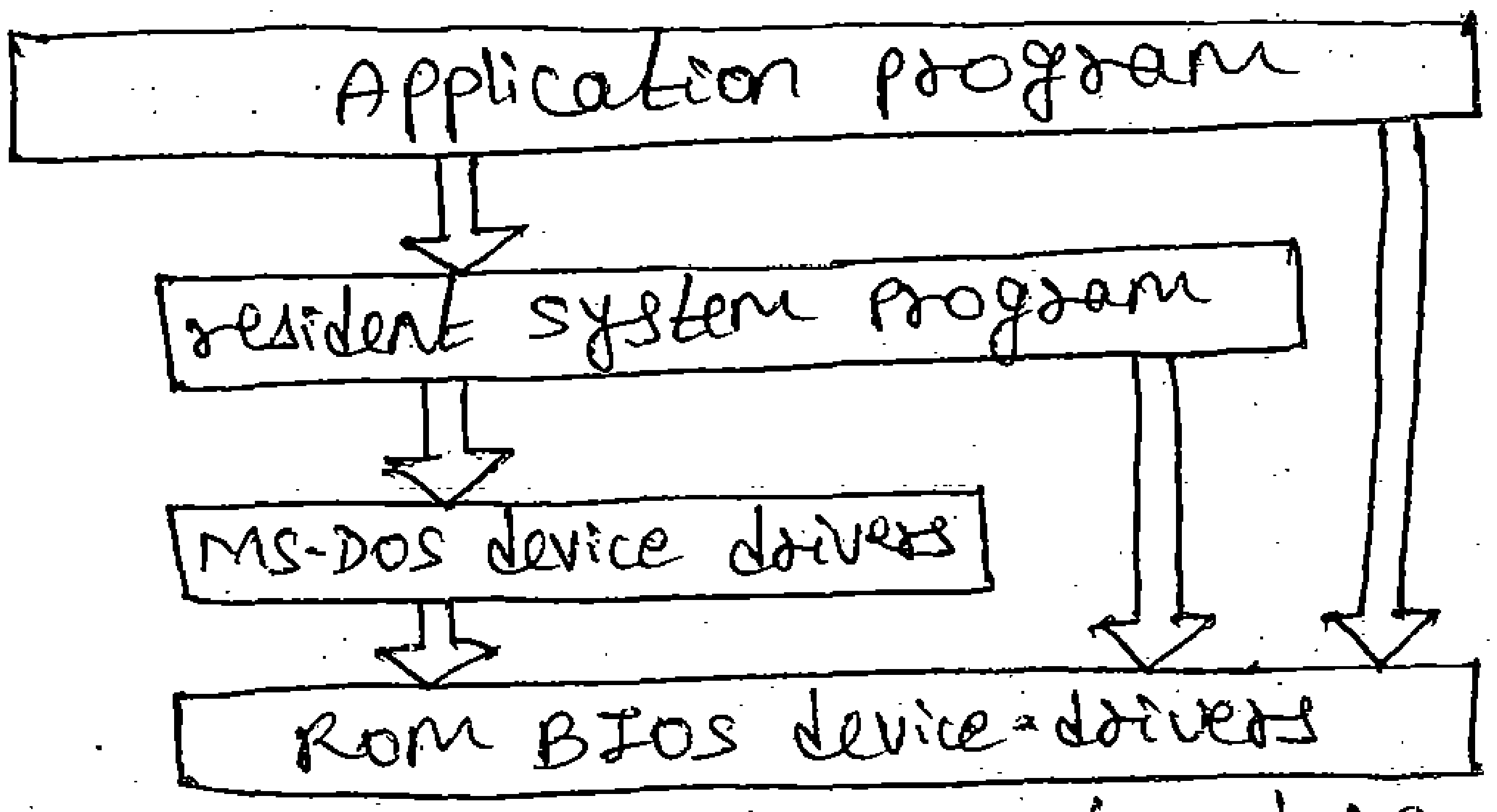
- * Once an OS is designed, it must be implemented.
- * Emulators are programs that duplicate the functionality of one system with another system.

⑩ Operating-System structure

Simple structure

→ MS-DOS - written to provide the most functionality in the least space.

- * Not divided into modules.
- * Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated.

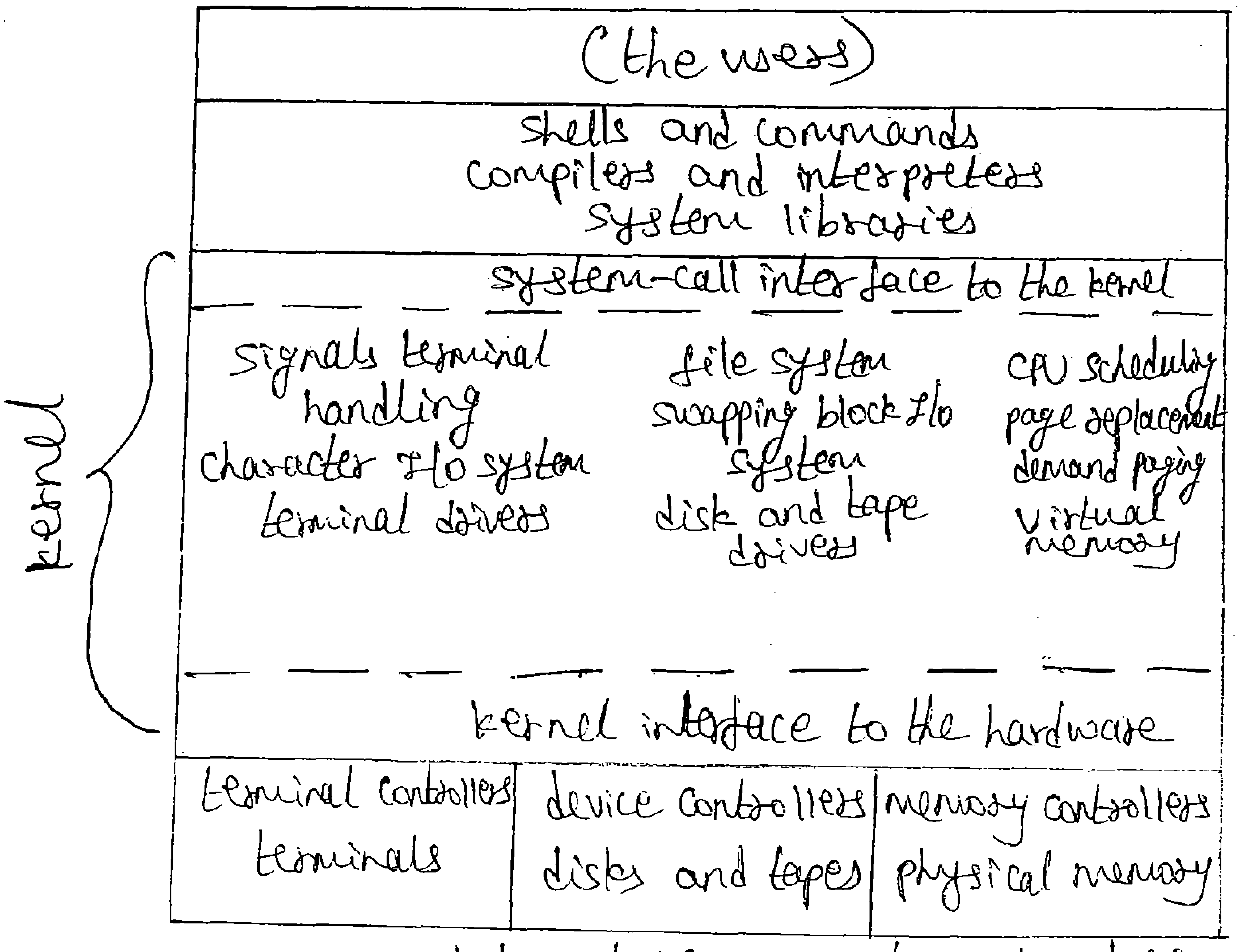


MS-DOS layer structure

Layered Approach

→ The OS 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.



Traditional UNIX system structure

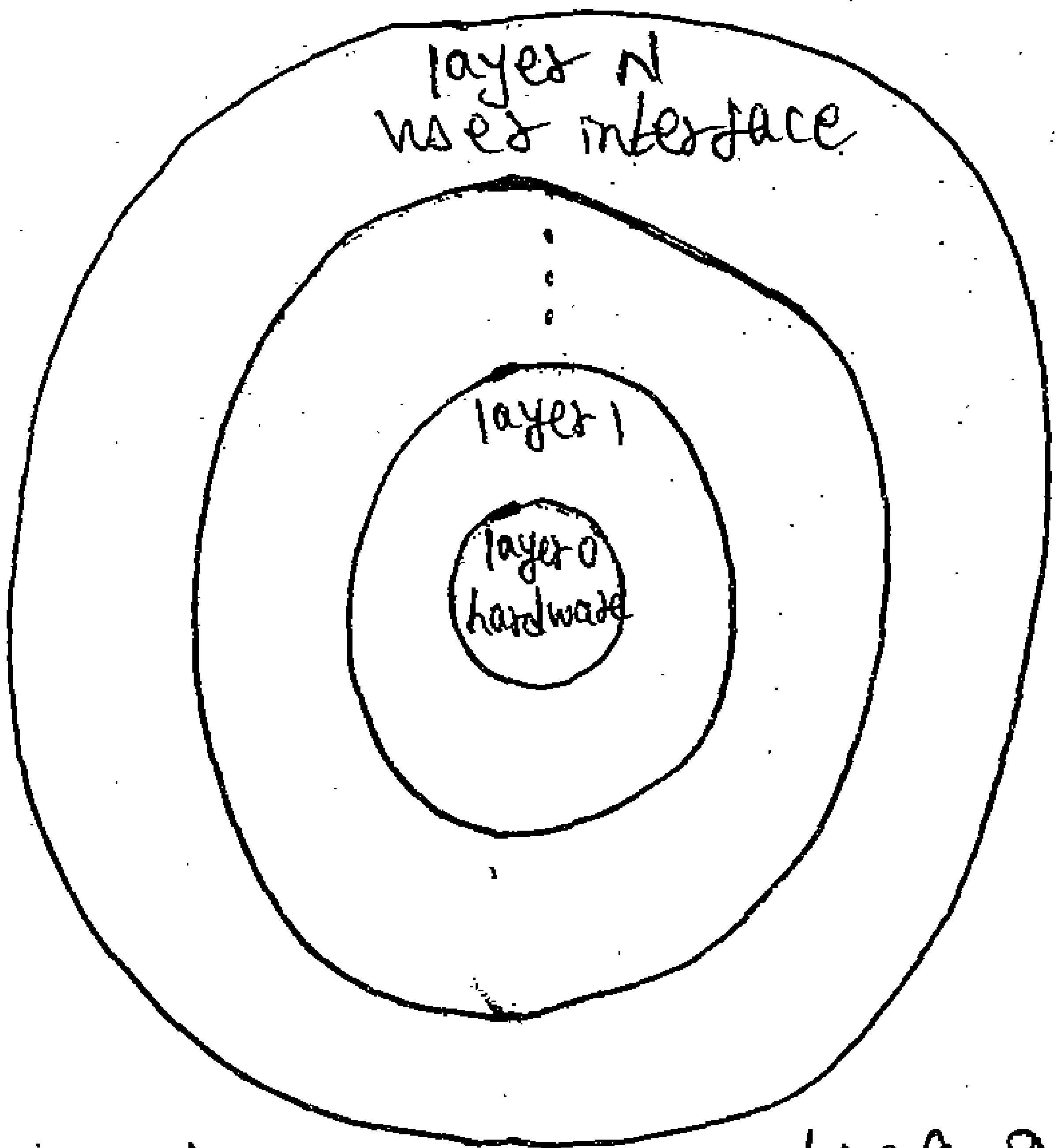
→ Another example of limited structuring is the original UNIX OS.

* The Unix OS consists of two separable parts:

≫ Systems programs.

≫ The kernel.

- consists of everything below the system-call interface and above the physical hardware.
- Provides the file system, CPU scheduling, memory management and other OS functions; a large number of functions for one level.



A layered operating system

Microkernels

→ Moves as much from the kernel into "user" space.

→ Communication takes place between user modules using message passing.

→ Benefits:

* Easier to extend a microkernel.

* Easier to port the OS to new architectures.

* More reliable (less code is running in kernel mode)

* More secure.

→ Detriments/Dangerous: Performance overhead of user space to kernel space communication.

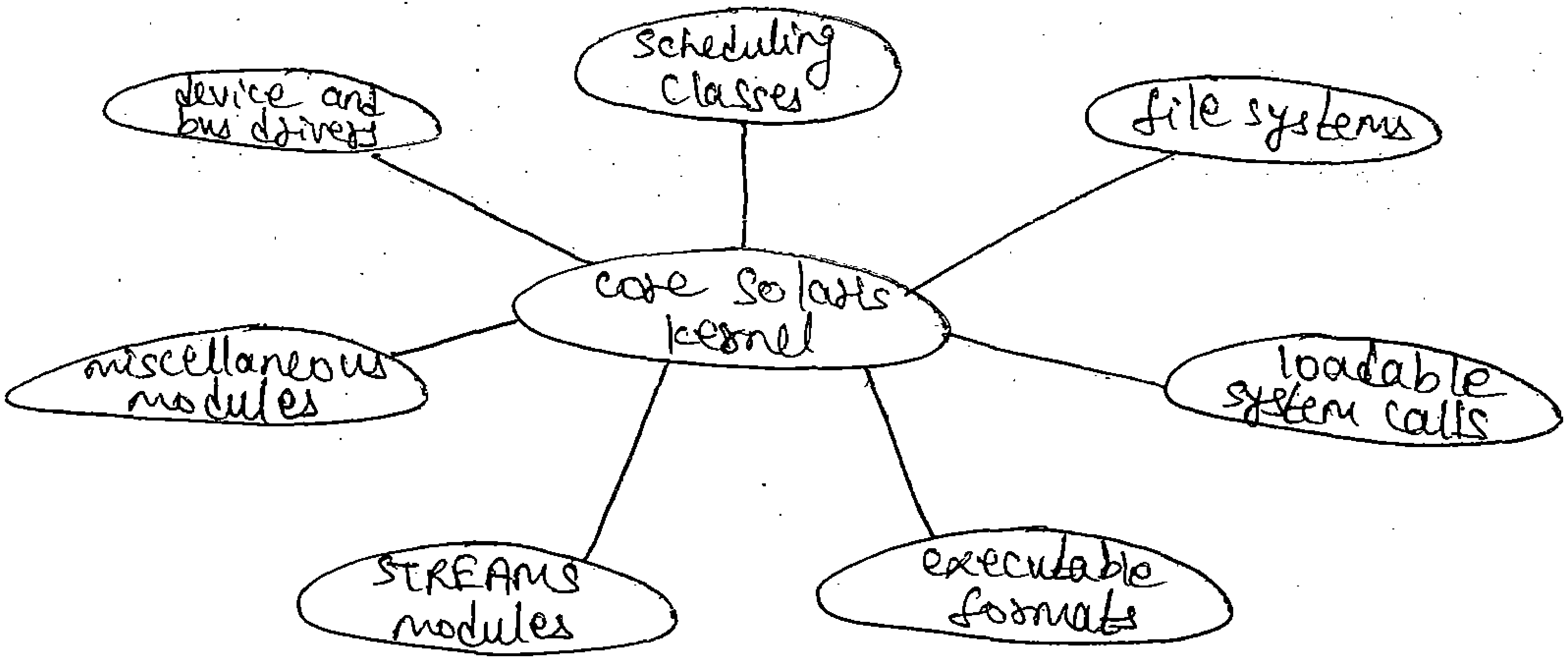
Modules

→ Most modern operating systems implement kernel modules.

* Uses object-oriented approach.
* Each core component is separate.
* Each talks to the others over known interfaces.
* Each is loadable as needed within the kernel.

> Overall, similar to layers but with more flexible.

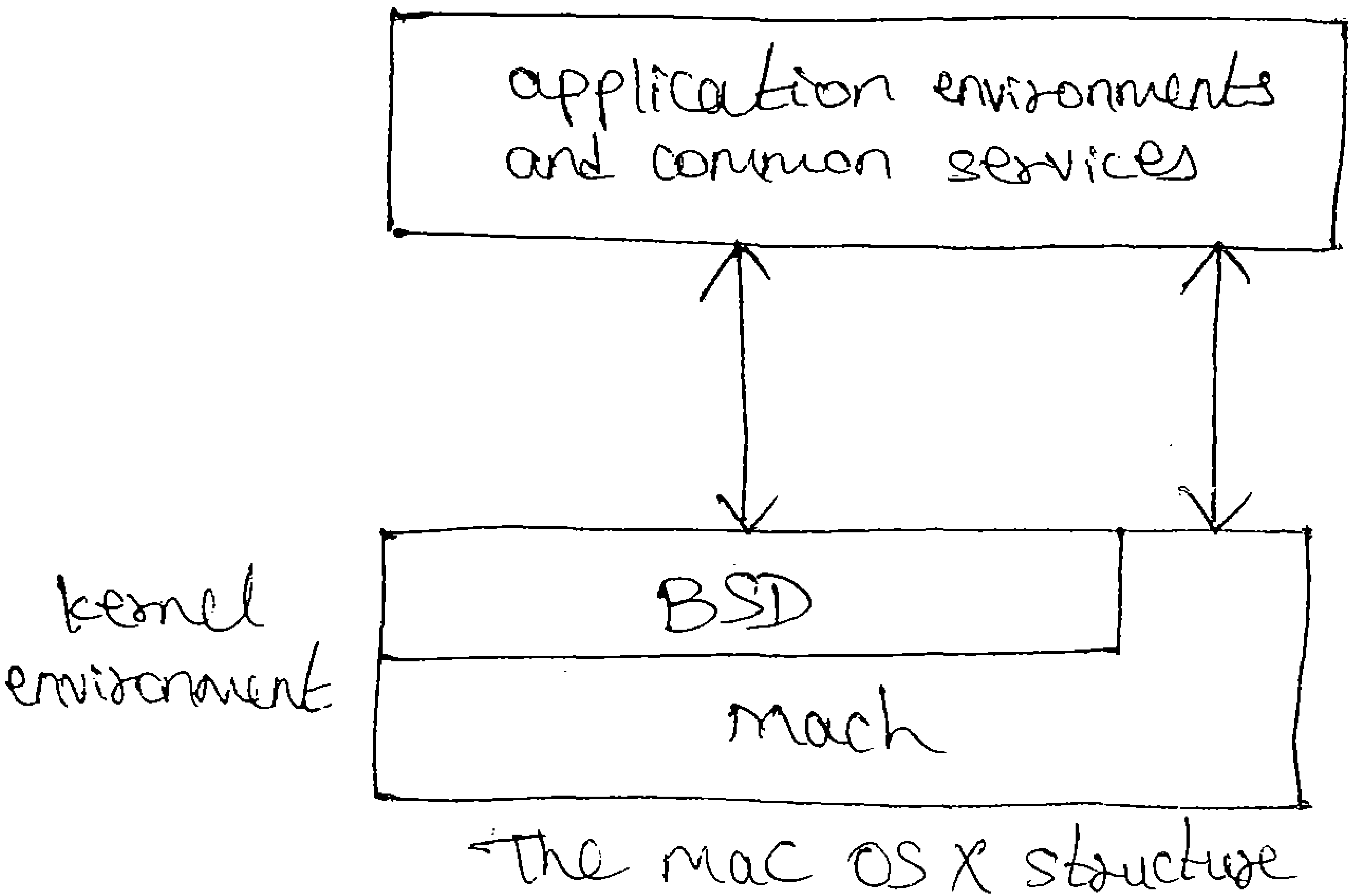
> Ex!- Solaris operating system structure, is organized around a core kernel with seven types of loadable kernel modules.



Solaris loadable modules

> Ex!- Apple Mac OS X os uses a hybrid structure. It is a layered system in which one layer consists of the Mach microkernel.

> kernel environment consists primarily of Mach microkernel and the BSD kernel.



The mac OS X structure

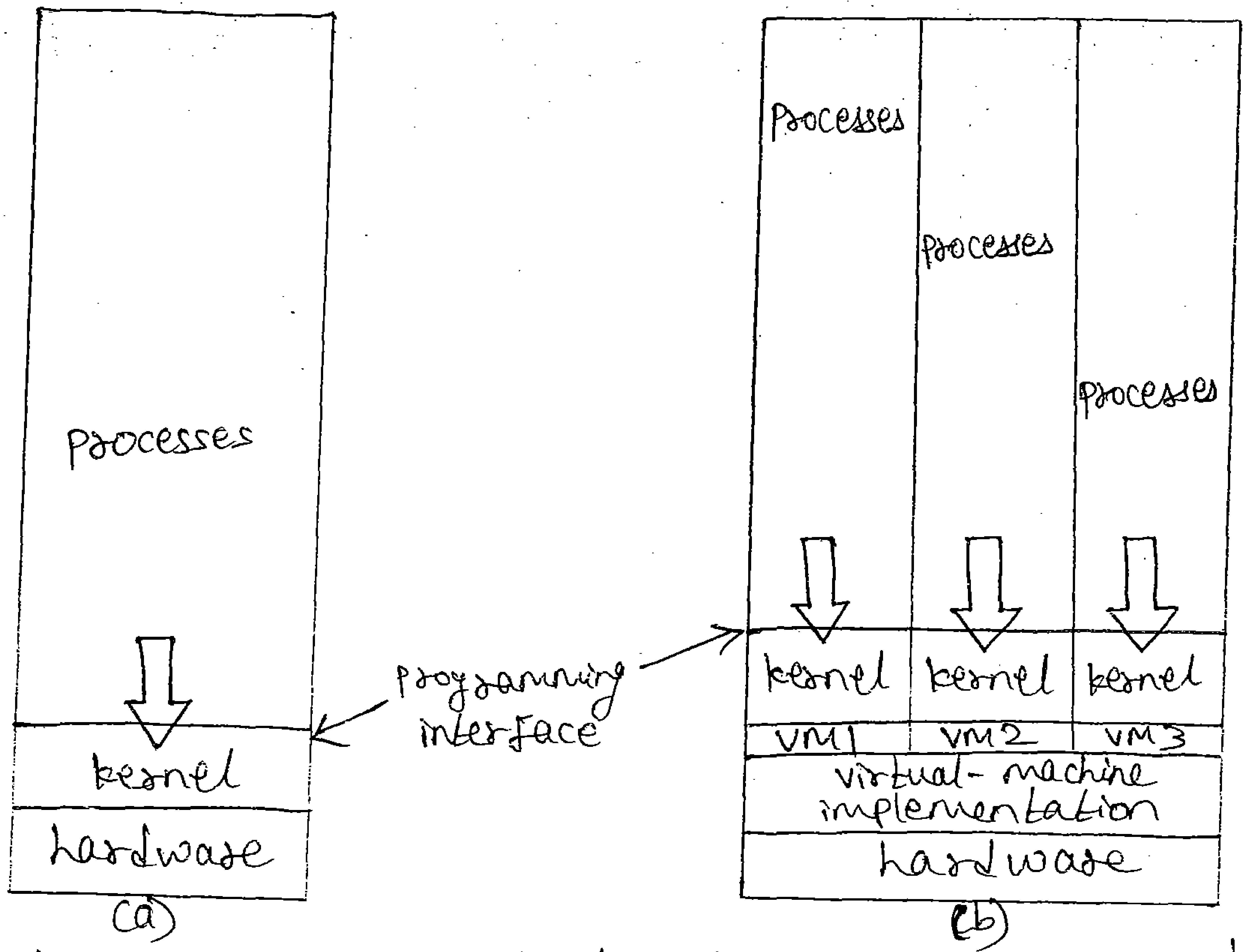
⑫ Virtual Machines

→ A virtual machine takes the layered approach to its logical conclusion. It treats hardware and the OS kernel as though they were all hardware.

→ A virtual machine provides an interface identical to the underlying base hardware.

→ The OS host creates the illusion that a process has its own processor and (virtual) memory.

→ Each guest provided with a (virtual) copy of underlying computer.



system models. (a) Non-virtual machine (b) virtual machine

Virtual Machines History and Benefits.

→ First appeared commercially in IBM mainframes in 197

→ Fundamentally, multiple execution environments (different OS) can share the same hardware.

→ Protect from each other.

→ Commutate with each other, other physical systems via networking.

→ Useful for development, testing

→ Consolidation of many low-resource use systems ~~via networking~~ onto fewer busier systems.

→ "open virtual Machine Format" standard format of virtual machines, allows a vm to run within many different virtual machine (host) Platforms.

Simulation

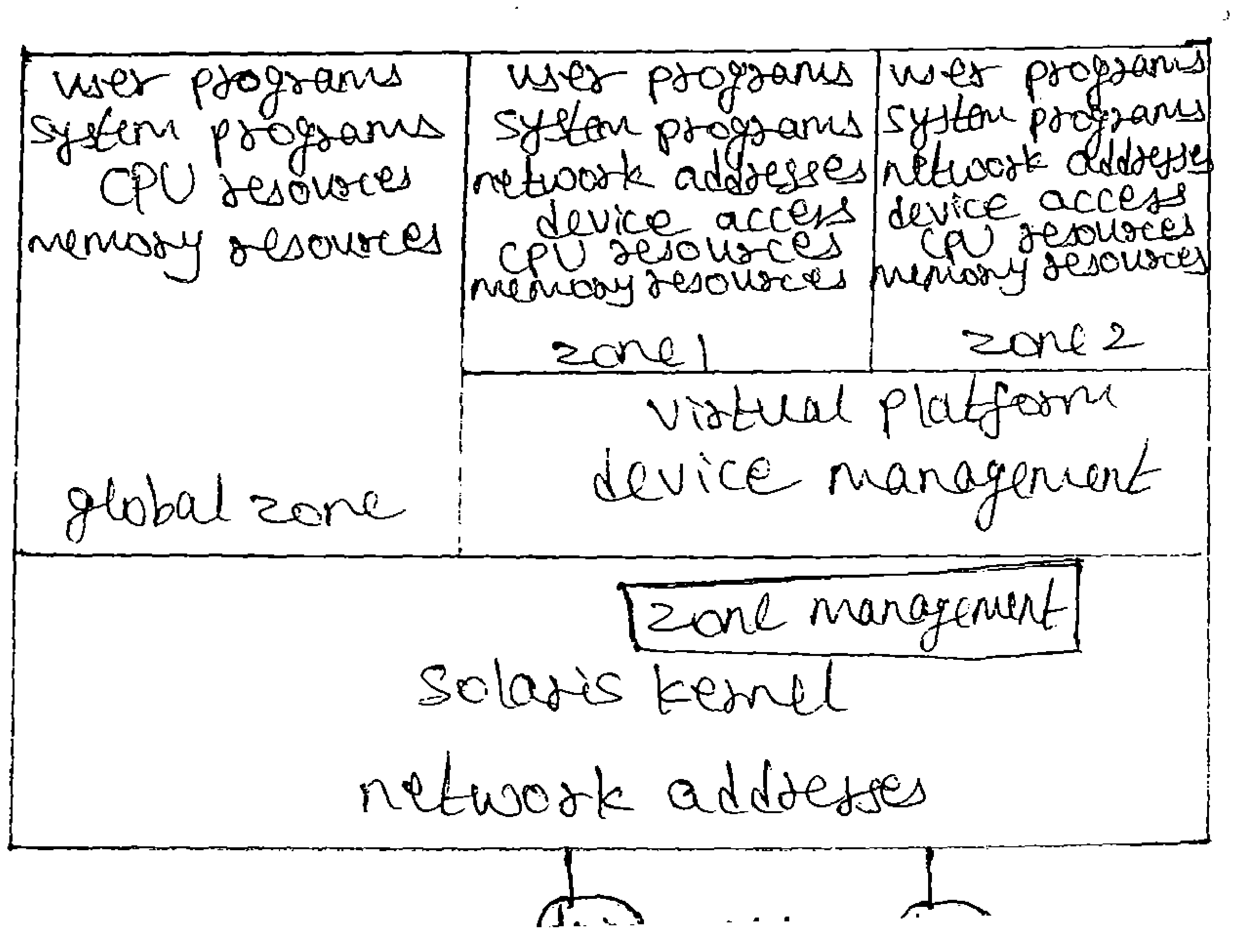
→ Another methodology is simulation, in which the host system has one system architecture and the guest system was compiled for a different architecture.

Para-virtualization

→ Presents the guest with a system that is similar but not identical to the guest's preferred system.

→ Guest must be modified to run on the paravirtualized hardware.

→ Guest can be an OS, (or) in the case of Solaris 10 applications running in containers



Implementation

-> Difficult to implement - must provide an exact duplicate of underlying machine.

-> Typically runs in user mode, creates virtual user mode and virtual kernel mode.

-> Timing Can be an issue - slower than real machine

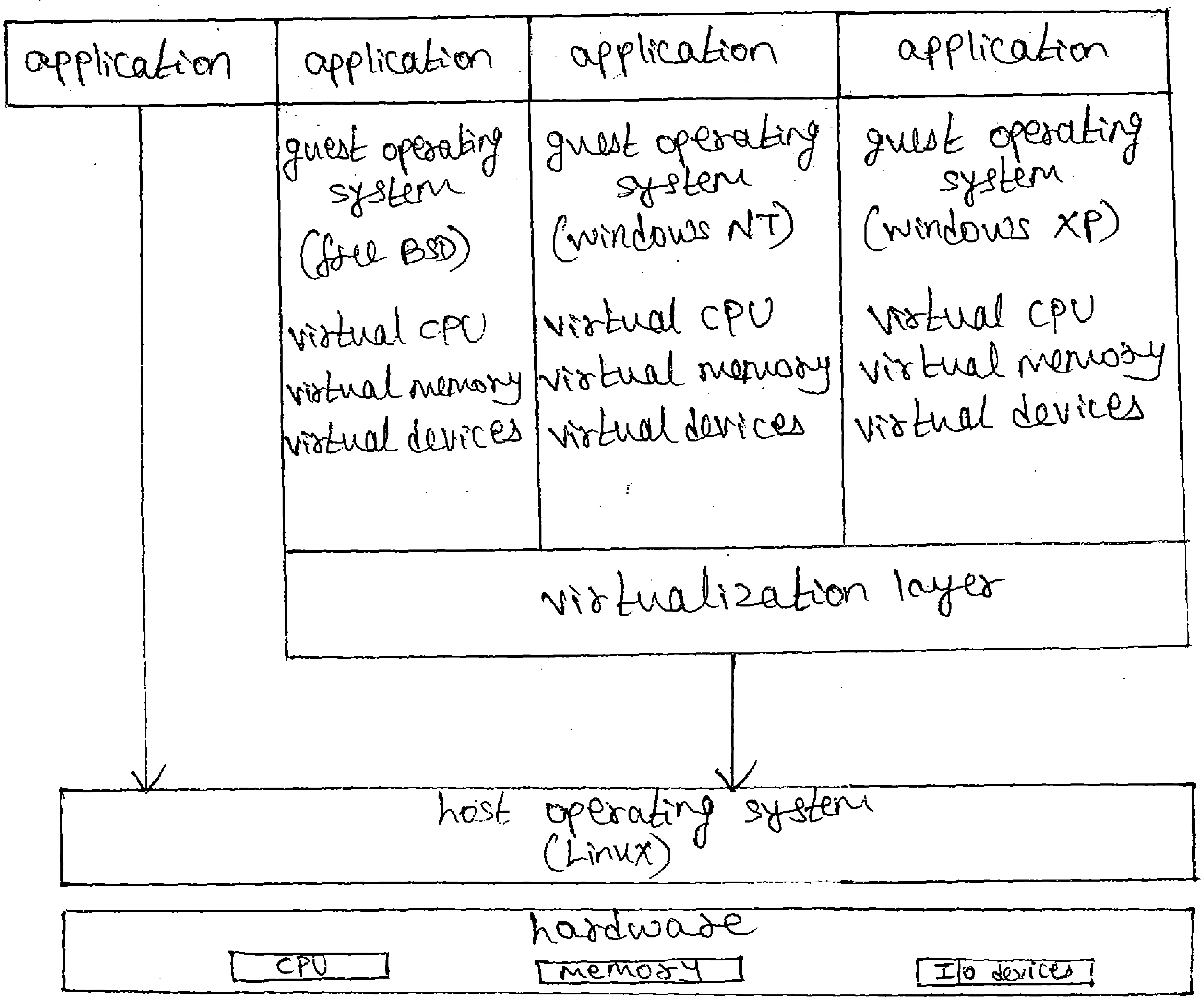
-> Hardware support needed

  * More support -> better virtualization.

    Ex!- AMD virtualization technology provides "host" and "guest" modes.

VMware

-> VMware Workstation is a popular commercial application that abstracts Intel X86 and compatible hardware into isolated virtual machines.



The Java Virtual Machine. (JVM)

> Java is a popular object - oriented programming language introduced by Sun Microsystems in 1995.

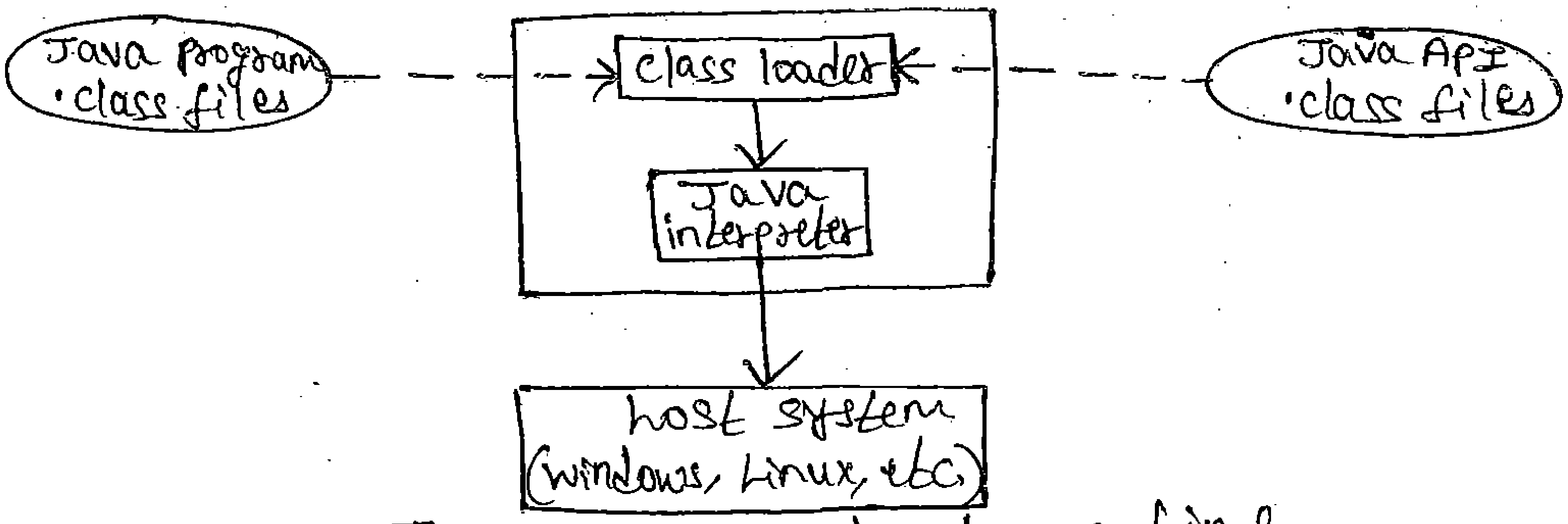
> Java provides a specification for a JVM.

> Compiled Java programs are platform-neutral bytecodes executed by a JVM.

> JVM consists of -class loader -class verifier -runtime interpreter.

> JVM automatically manages memory by performing garbage collection.

> Just-In-Time (JIT) compilers increase performance



The Java virtual machine

Operating-System Debugging

> Debugging is finding and fixing errors (or) bugs.

- OS generate log files containing error information.

> Failure of an application can generate core dump file capturing memory of the process.

- OS failure can generate crash dump file containing kernel memory.

Beyond crashes, performance tuning can optimize system performance.

Kernighan's Law - " Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as

→ DTrace tool in Solaris, FreeBSD, Mac OS X allows live instrumentation on production systems.

* Probes fire when code is executed, capturing state data and sending it to consumers of those probes.

Operating-System Generation

→ Operating systems are designed to run on any of a class of machines; the system must be configured for each specific computer site.

→ System Generation (SYSGEN) program obtains information concerning the specific configuration of the hardware system

System Boot

→ After an OS is generated, it must be made available for use by the hardware.

→ Booting - starting a computer by loading the kernel.

→ Bootstrap program (or) bootstrap loader - locates the kernel, loads it into memory, and starts it.

→ Sometimes two-step process where boot block at fixed location loads bootstrap loader.

→ when power initialized on system, execution starts at a fixed memory location.

* Firmware used to hold initial boot code.