CHAPTER 1 INTRODUCTION TO OPERATING SYSTEMS

by
Prof. Chetan K. Solanki

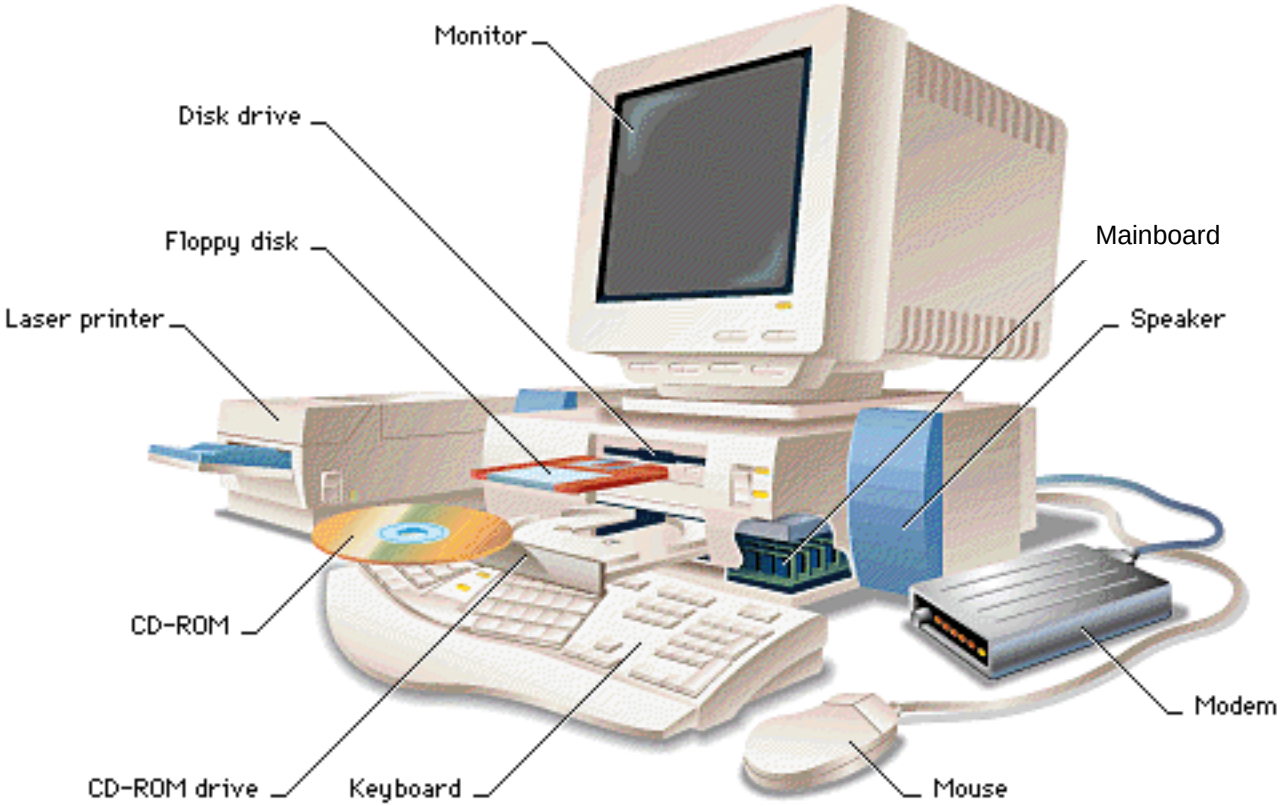
1.1 General Definition

- An OS is a program which acts as an *interface* between computer system, users and the computer hardware.
- 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.

1.1 General Definition

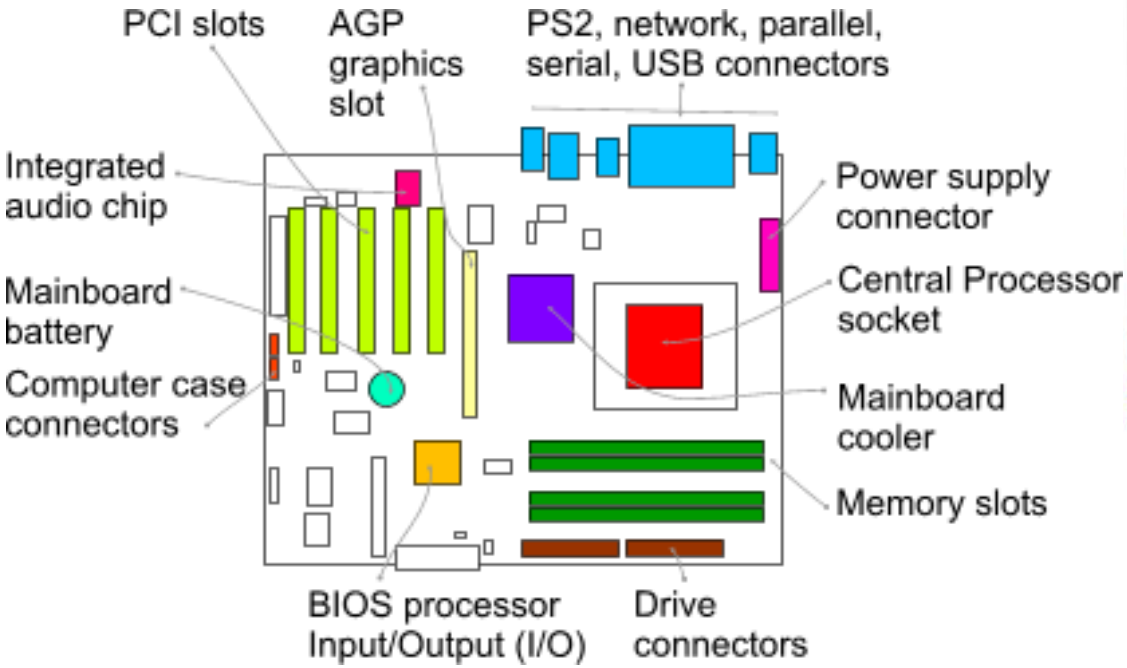
- In general, a computer system has some resources which may be utilized to solve a problem. They are
  - Memory
  - Processor(s)
  - I/O
  - File System
  - etc.

1.1 General Definition



1.1 General Definition

1.1 General Definition



mainboard

1.1 General Definition

processor

1.1 General Definition

RAM

What is OS?

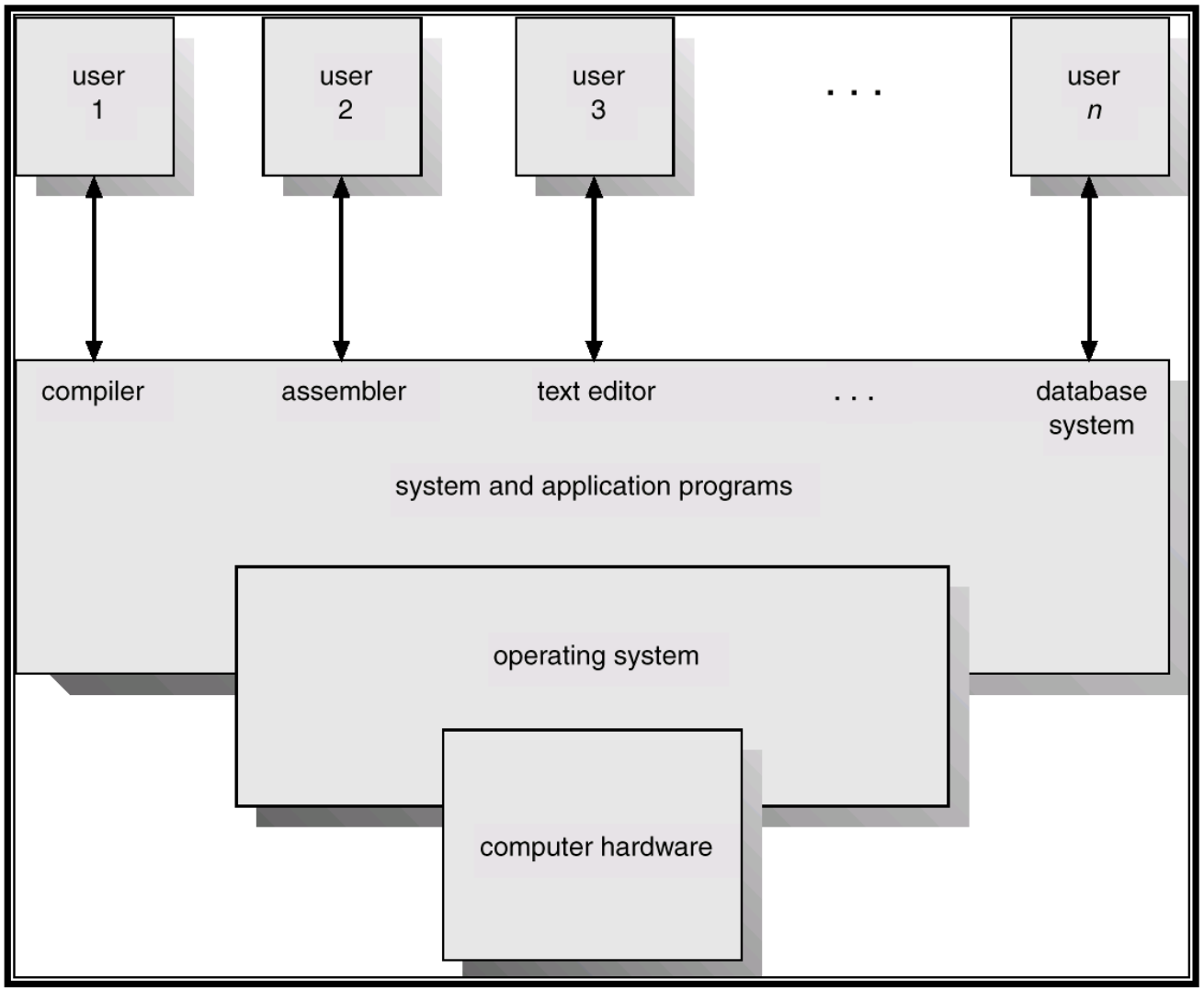
- An interface between Hardware and User Programs
- An abstraction of the hardware for all the (user) processes
  - Hide the complexity of the underlying hardware and give the *user* a better view of the computer

- => A MUST!

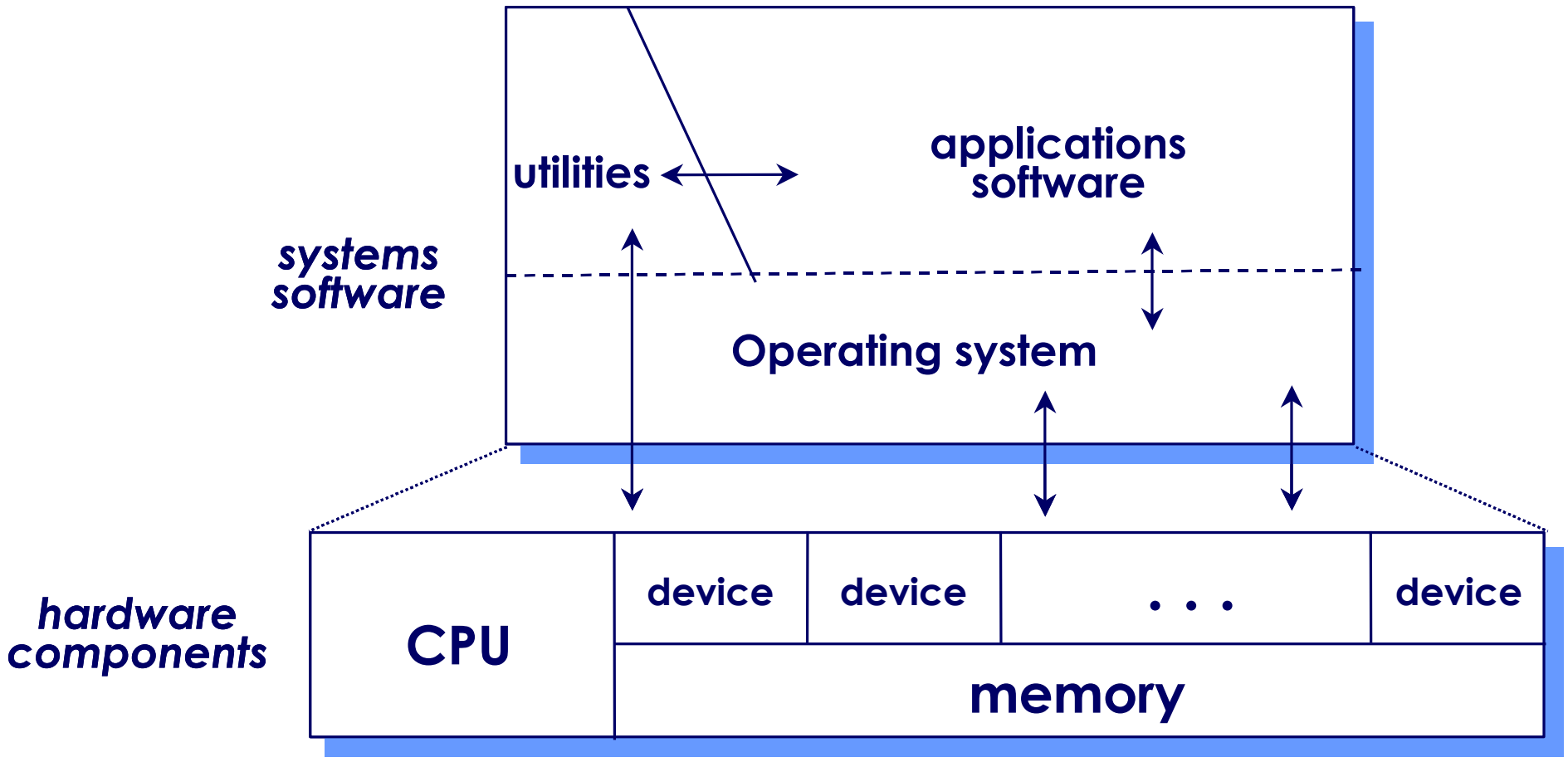
Computer System Components

1. Hardware – provides basic computing resources (CPU, memory, I/O devices).
2. Operating system – controls and coordinates the use of the hardware among the various application programs for the various users.
3. Applications programs – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).
4. Users (people, machines, other computers).

Abstract View of System Components



The OS

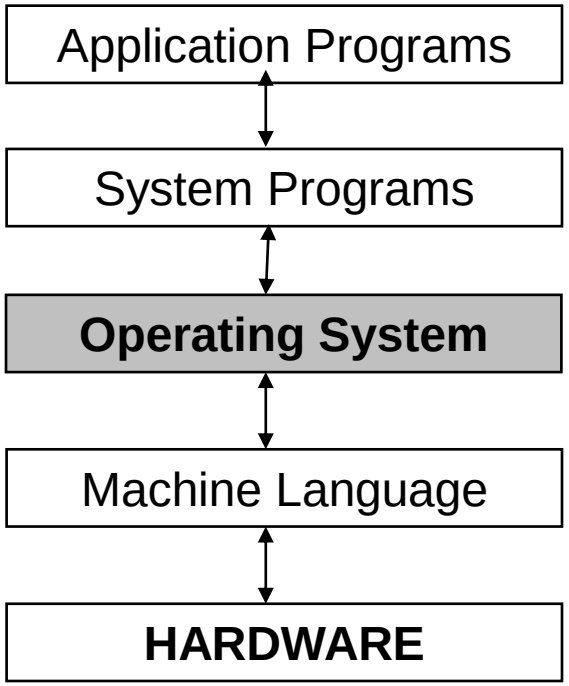

utilities
applications software
systems software
Operating system
hardware components
CPU
device
device
. . .
device
memory

1.1 General Definition

- The OS manages these resources and allocates them to specific programs and users.
- With the management of the OS, a programmer is rid of difficult hardware considerations.
- An OS provides services for
  - Processor Management
  - Memory Management
  - File Management
  - Device Management
  - Concurrency Control

1.1 General Definition

- Another aspect for the usage of OS is that; it is used as a *predefined library* for hardware-software interaction.
- This is why, system programs apply to the installed OS since they cannot reach hardware directly.



1.1 General Definition

- Since we have an already written library, namely the OS, to add two numbers we simply write the following line to our program:

```
c = a + b ;
```

1.1 General Definition

- in a system where there is no OS installed, we should consider some hardware work as:

  (Assuming an MC 6800 computer hardware)

  LDAA $80 → Loading the number at memory location 80
  LDAB $81 → Loading the number at memory location 81
  ADDB → Adding these two numbers
  STAA $55 → Storing the sum to memory location 55

- locations and used our hardware knowledge of the system.ware knowledge of the system.

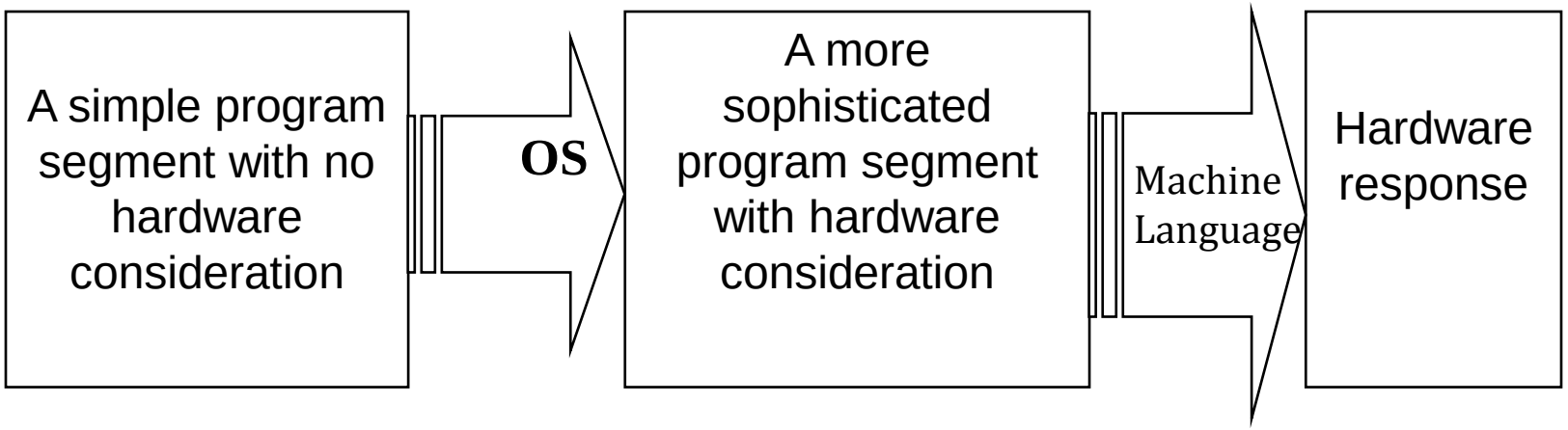
1.1 General Definition

- In an OS installed machine, since we have an intermediate layer, our programs obtain *some advantage of mobility* by not dealing with hardware.
- For example, the above program segment would not work for an 8086 machine, where as the

  "c = a + b ;"

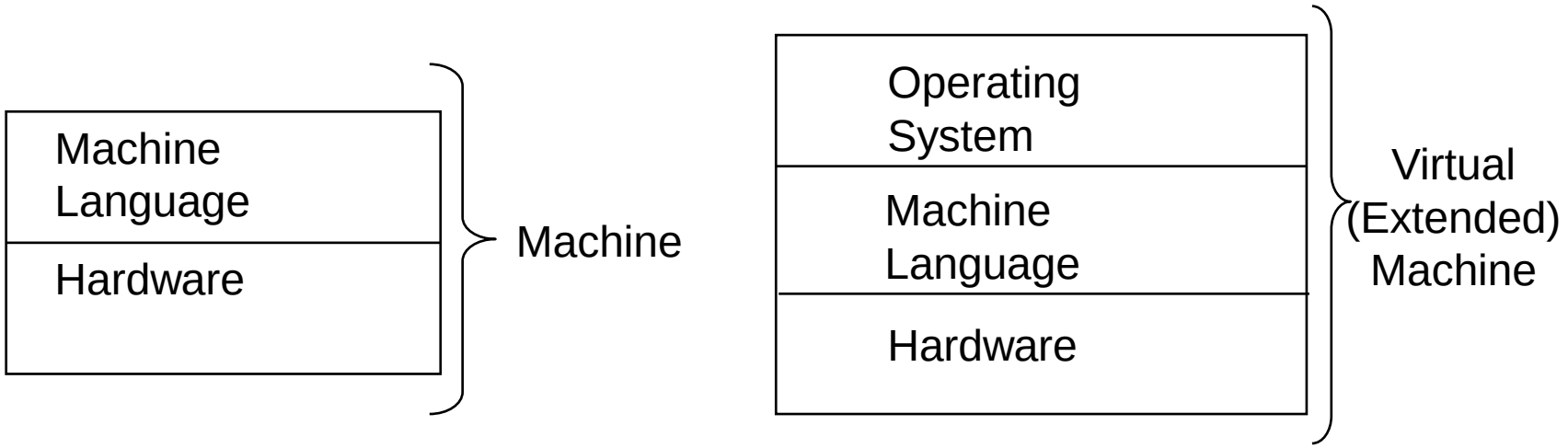
  syntax will be suitable for both.

1.1 General Definition



1.1 General Definition

- With the advantage of easier programming provided by the OS, the hardware, its machine language and the OS constitutes a new combination called as a **virtual (extended) machine**.

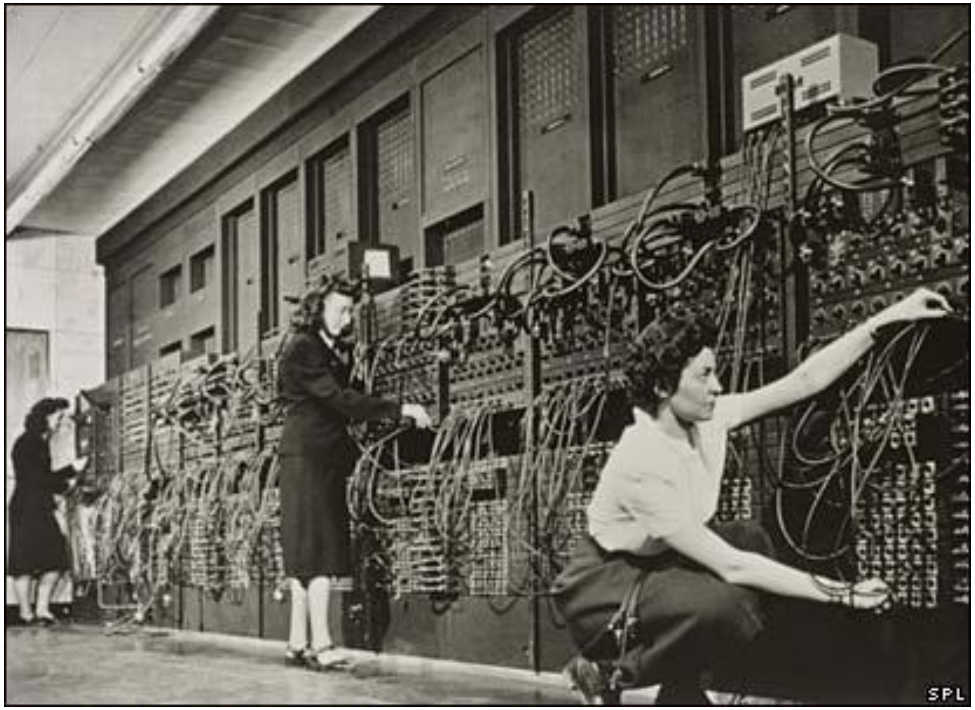


1.1 General Definition

- In a more simplistic approach, in fact, OS itself is a program.
- But it has a priority which application programs don't have.
- OS uses the **kernel mode** of the microprocessor, whereas other programs use the **user mode**.
- The difference between two is that; all hardware instructions are valid in kernel mode, where some of them cannot be used in the user mode.

1.2 History of Operating Systems

- It all started with computer hardware in about 1940s.



ENIAC 1943

1.2 History of Operating Systems

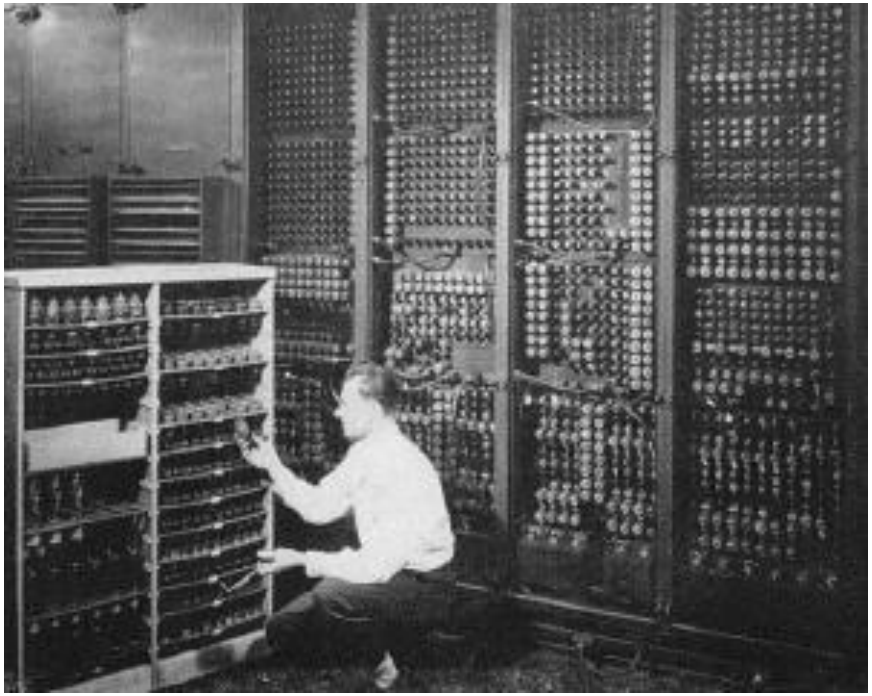
- ENIAC (Electronic Numerical Integrator and Computer), at the U.S. Army's Aberdeen Proving Ground in Maryland.
  - built in the 1940s,
  - weighed 30 tons,
  - was eight feet high, three feet deep, and 100 feet long
  - contained over 18,000 vacuum tubes that were cooled by 80 air blowers.

1.2 History of Operating Systems

- Computers were using vacuum tube technology.

ENIAC's vacuum tubes

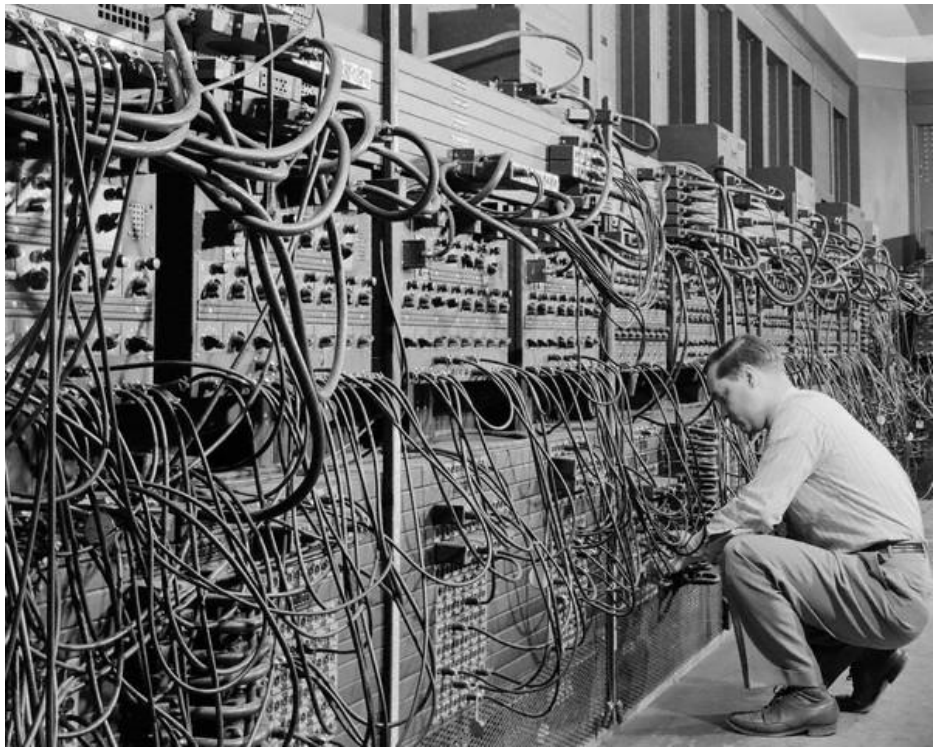
1.2 History of Operating Systems



ENIAC's backside

1.2 History of Operating Systems

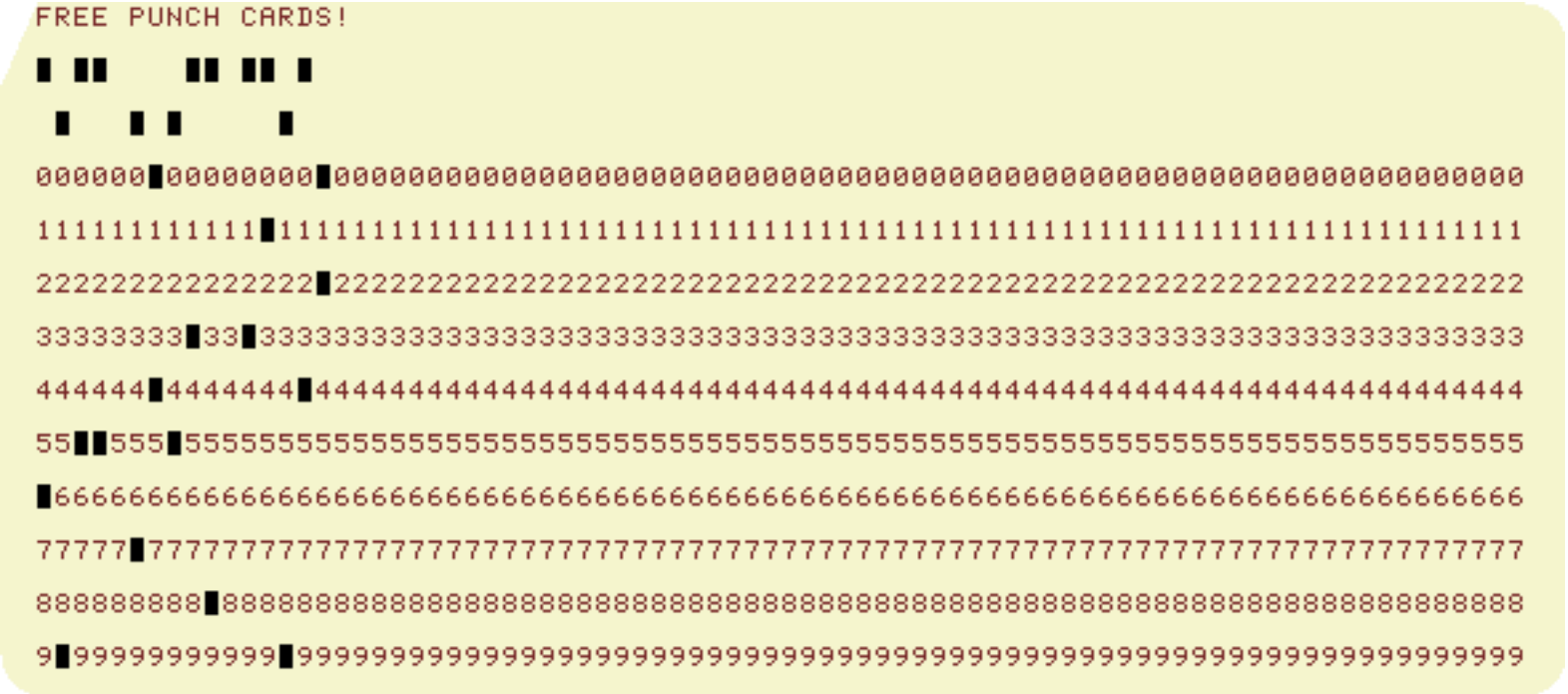
Programs were loaded into memory manually using switches, punched cards, or paper tapes.



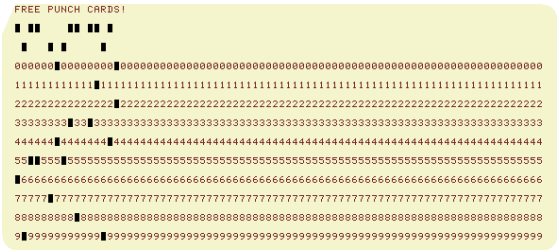
ENIAC : coding by cable connections

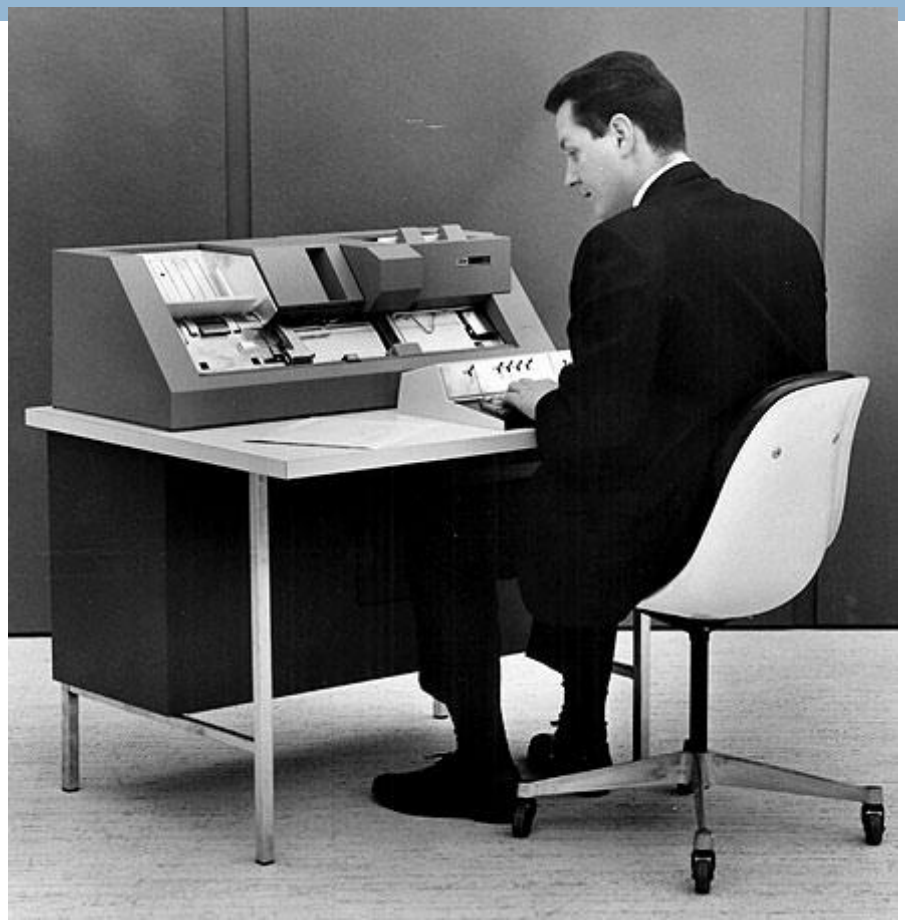
1.2 History of Operating Systems

punch card

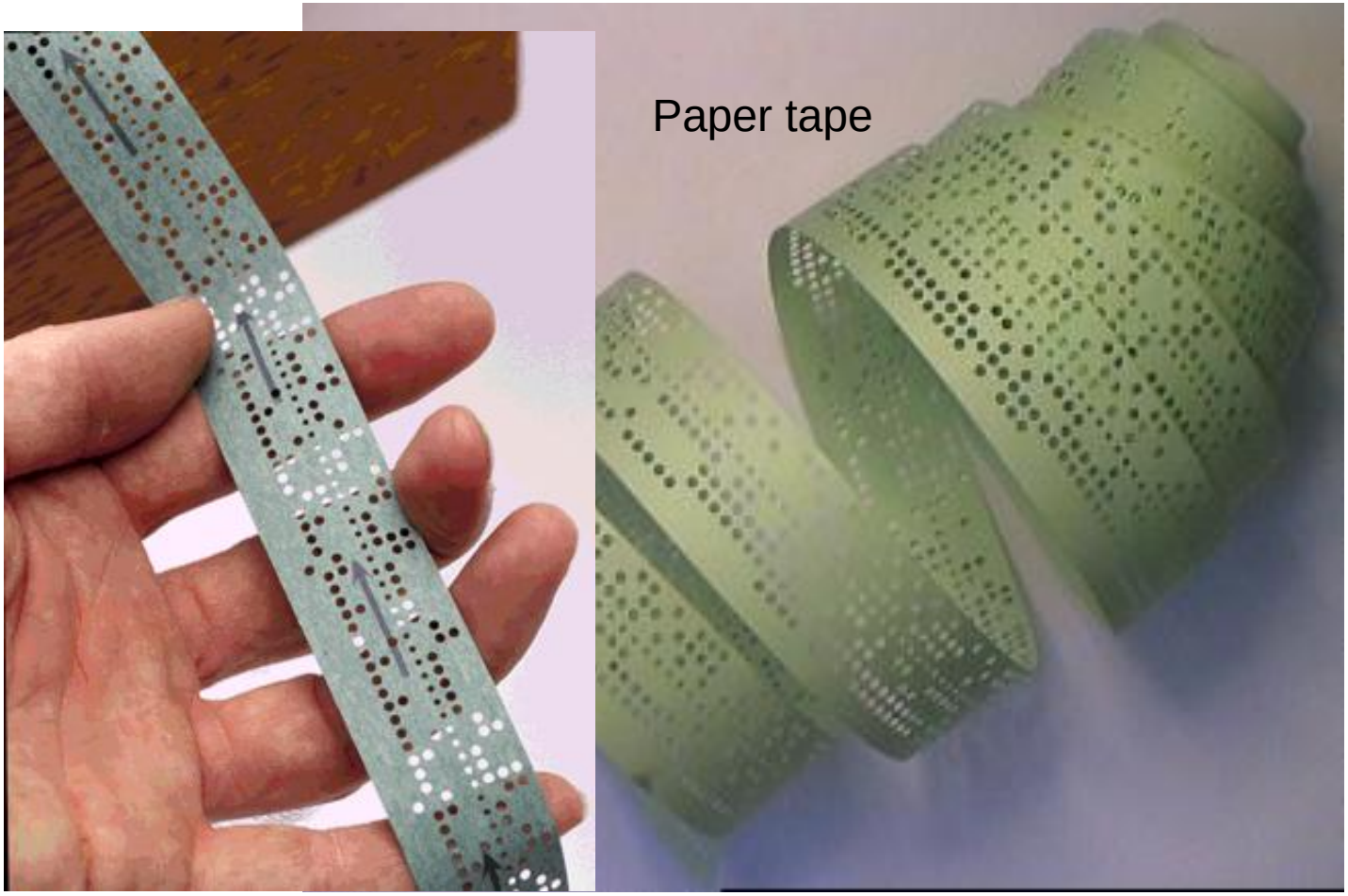


1.2 History of Operating Systems

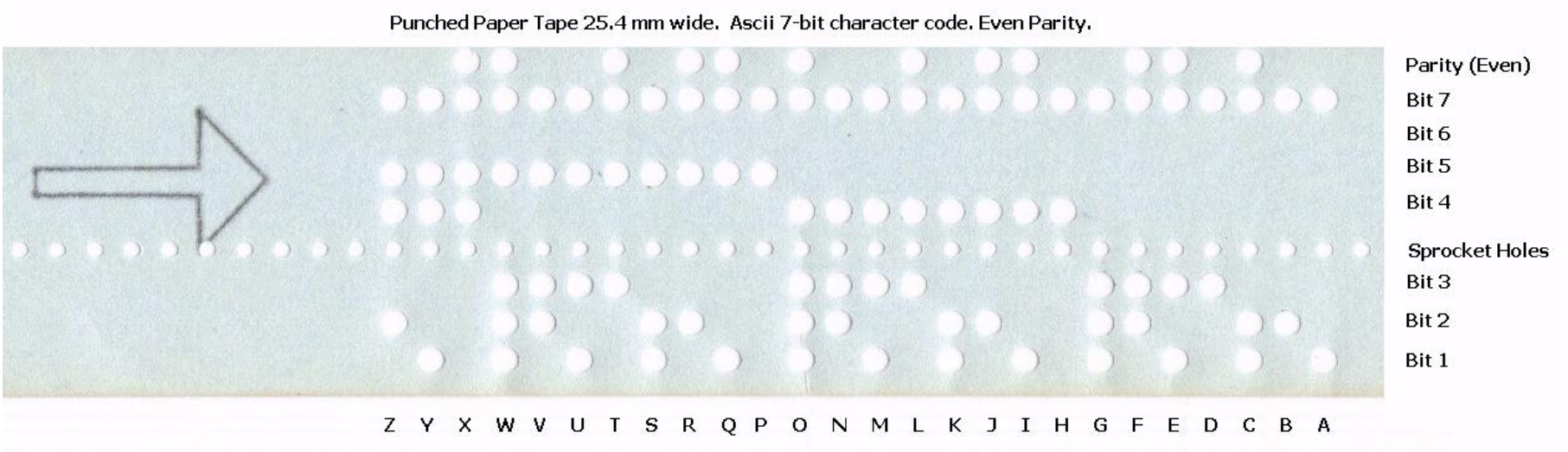




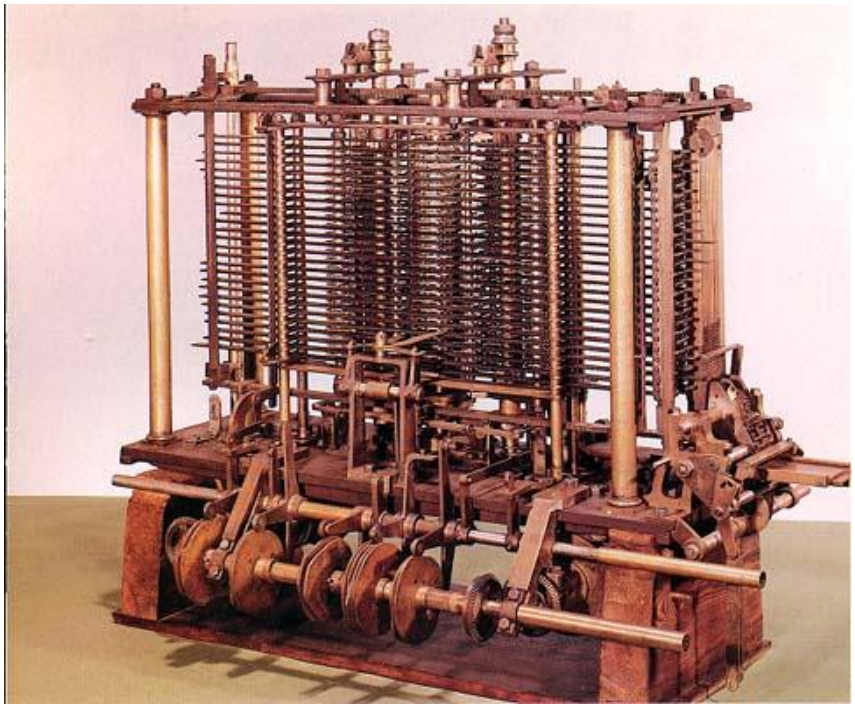
1.2 History of Operating Systems



1.2 History of Operating Systems



1.2 History of Operating Systems

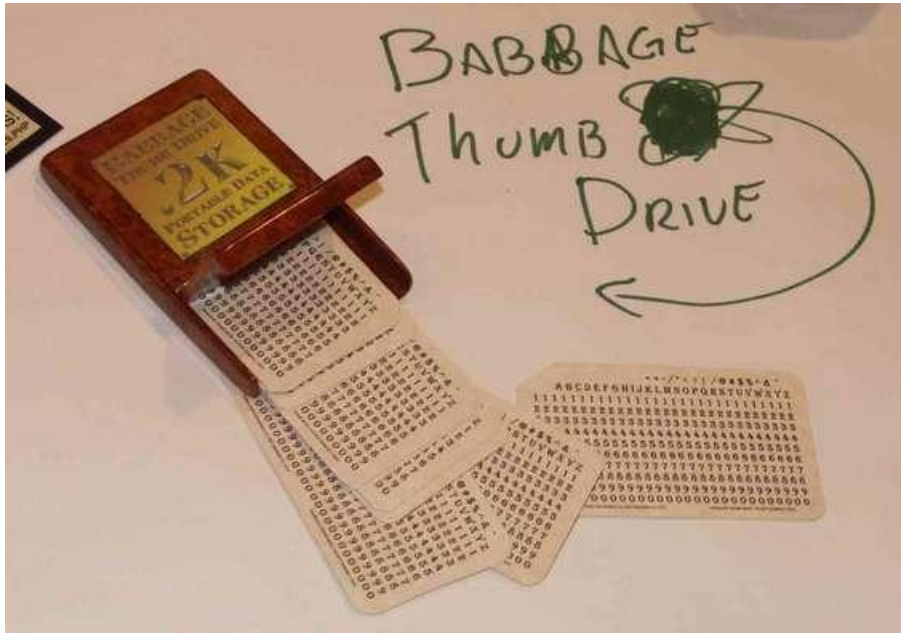


Babbage's analytical engine
(designed in 1840's by Charles Babbage, but cold not be constructed by him.
An earlier and simpler version is constructed in 2002, in London)

http://www.computerhistory.org/babbage/

1.2 History of Operating Systems

- Ada Lovalence (at time of Charles Babbage) wrote code for analytical engine to compute Bernulli Numbers



1.2 History of Operating Systems

- As time went on, card readers, printers, and magnetic tape units were developed as additional hardware elements.
- Assemblers, loaders and simple utility libraries were developed as software tools.
- Later, off-line spooling and channel program methods were developed sequentially.

1.2 History of Operating Systems

Commodore PET, 1977

History of the OS

- Two distinct phases of history
  - Phase 1: Computers were expensive
    - Goal: Use computer's time efficiently
    - Maximize throughput (I.e., jobs per second)
    - Maximize utilization (I.e., percentage busy)
  - Phase 2: Computers are inexpensive
    - Goal: Use people's time efficiently
    - Minimize response time

First commercial systems

- 1950s Hardware
  - Large in size, expensive, and slow
  - Input/Output: Punch cards and line printers
- Goal of OS
  - Get the hardware working
  - Single operator/programmer/user runs and debugs interactively
- OS Functionality
  - Standard library only (no sharing or coordination of resources)
  - Monitor that is always resident; transfer control to programs
- Advantages
  - Worked and allowed interactive debugging
- Problems
  - Inefficient use of hardware (throughput and utilization)

Inexpensive Peripherals

- 1960s Hardware
  - Expensive mainframes, but inexpensive keyboards and monitors
  - Enables text editors and interactive debuggers
- Goal of OS
  - Improve user's response time
- OS Functionality
  - Time-sharing: switch between jobs to give appearance of dedicated machine
  - More complex job scheduling
  - Concurrency control and synchronization
- Advantage
  - Users easily submit jobs and get immediate feedback

Inexpensive Personal Computers

- 1980s Hardware
  - Entire machine is inexpensive
  - One dedicated machine per user
- Goal of OS
  - Give user control over machine
- OS Functionality
  - Remove time-sharing of jobs, protection, and virtual memory
- Advantages
  - Simplicity
  - Works with little main memory
  - Machine is all your own (performance is predictable)
- Disadvantages
  - No time-sharing or protection between jobs

Inexpensive, Powerful Computers

- 1990s+ Hardware
  - PCs with increasing computation and storage
  - Users connected to the web
- Goal of OS
  - Allow single user to run several applications simultaneously
  - Provide security from malicious attacks
  - Efficiently support web servers
- OS Functionality
  - Add back time-sharing, protection, and virtual memory

Current Systems

- Conclusion: OS changes due to both hardware and users
- Current trends
  - Multiprocessors
  - Networked systems
  - Virtual machines
- OS code base is large
  - Millions of lines of code
  - 1000 person-years of work
- Code is complex and poorly understood
  - System outlives any of its builders
  - System will always contain bugs
  - Behavior is hard to predict, tuning is done by guessing

Batch Processing

- Goal of OS: Better throughput and utilization
- Batch: Group of jobs submitted together
  - Operator collects jobs; orders efficiently; runs one at a time
- Advantages
  - Reducing setup costs over many jobs
  - Operator more skilled at loading tapes
  - Keep machine busy while programmer thinks
  - Improves throughput and utilization
- Problems
  - User must wait until batch is done for results
  - Machine idle when job is reading from cards and writing to printers

Spooling

- Hardware
  - Mechanical I/O devices much slower than CPU
  - Read 17 cards/sec vs. execute 1000 instructions/sec
- Goal of OS
  - Improve performance by overlapping I/O with CPU execution
- Spooling: Simultaneous Peripheral Operations On-Line
  1. Read card punches to disk
  2. Compute (while reading and writing to disk)
  3. Write output from disk to printer
- OS Functionality
  - Buffering and interrupt handling
- Problem
  - Machine idle when job waits for I/O to/from disk

Multiprogramming Operating Systems:

- Finally, the idea of **multiprogramming** came.
- Multiprogramming means sharing of resources between more than one processes.
- By multiprogramming the CPU time is not wasted, because, while one process moves on some I/O work, the OS picks another process to execute till the current one passes to I/O operation.

Multiprogrammed Batch Systems

- Observation: Spooling provides pool of ready jobs
- Goal of OS
  - Improve performance by always running a job
  - Keep multiple jobs resident in memory
  - When job waits for disk I/O, OS switches to another job
- OS Functionality
  - Job scheduling policies
  - Memory management and protection
- Advantage: Improves throughput and utilization
- Disadvantage: Machine not interactive

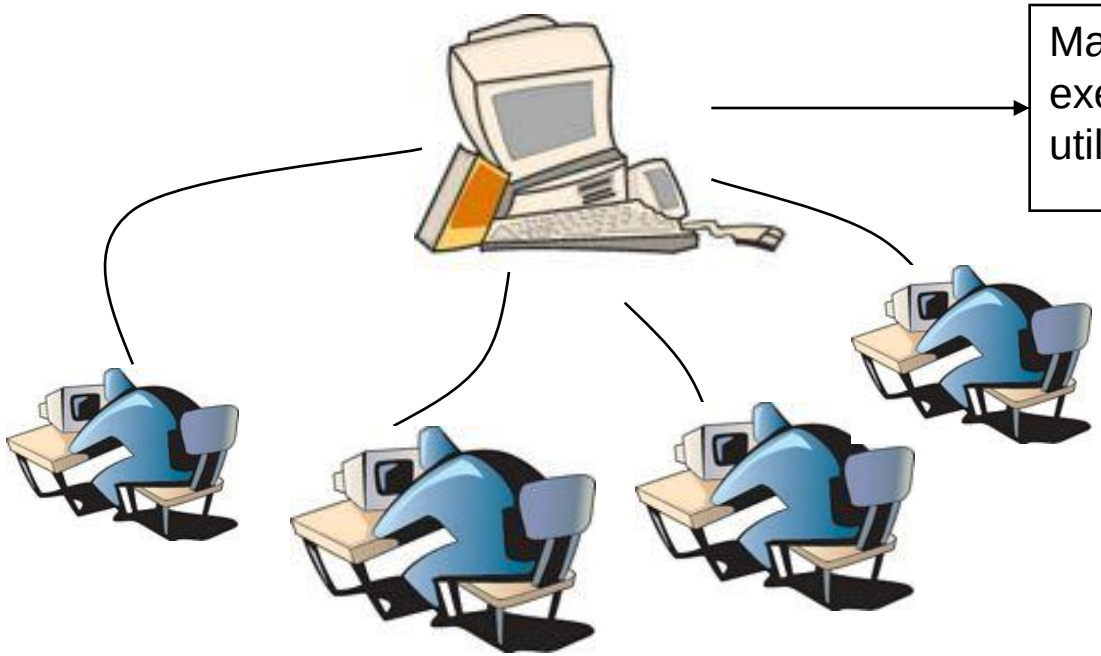
Multi-tasking and Single-tasking Operating Systems

- When a single program is allowed to run at a time, the system is grouped under the single-tasking system category.
- while in case the operating system allows for execution of multiple tasks at a time, it is classified as a multi-tasking operating system.
- In multitasking, the operating system slices the CPU time and dedicates one slot to each of the programs.

Multi-user and Single-user Operating Systems

- Computer operating systems of this type allow multiple users to access a computer system simultaneously.
- Time-sharing systems can be classified as multi-user systems as they enable a multiple user access to a computer through time sharing.
- Single-user operating systems, as opposed to a multi-user operating system, are usable by only one user at a time.
- Being able to have multiple accounts on a Windows operating system does not make it a multi-user system. Rather, only the network administrator is the real user.
- But for a Unix-like operating system, it is possible for two users to login at a time and this capability of the OS makes it a multi-user operating system.

1.2 History of Operating Systems



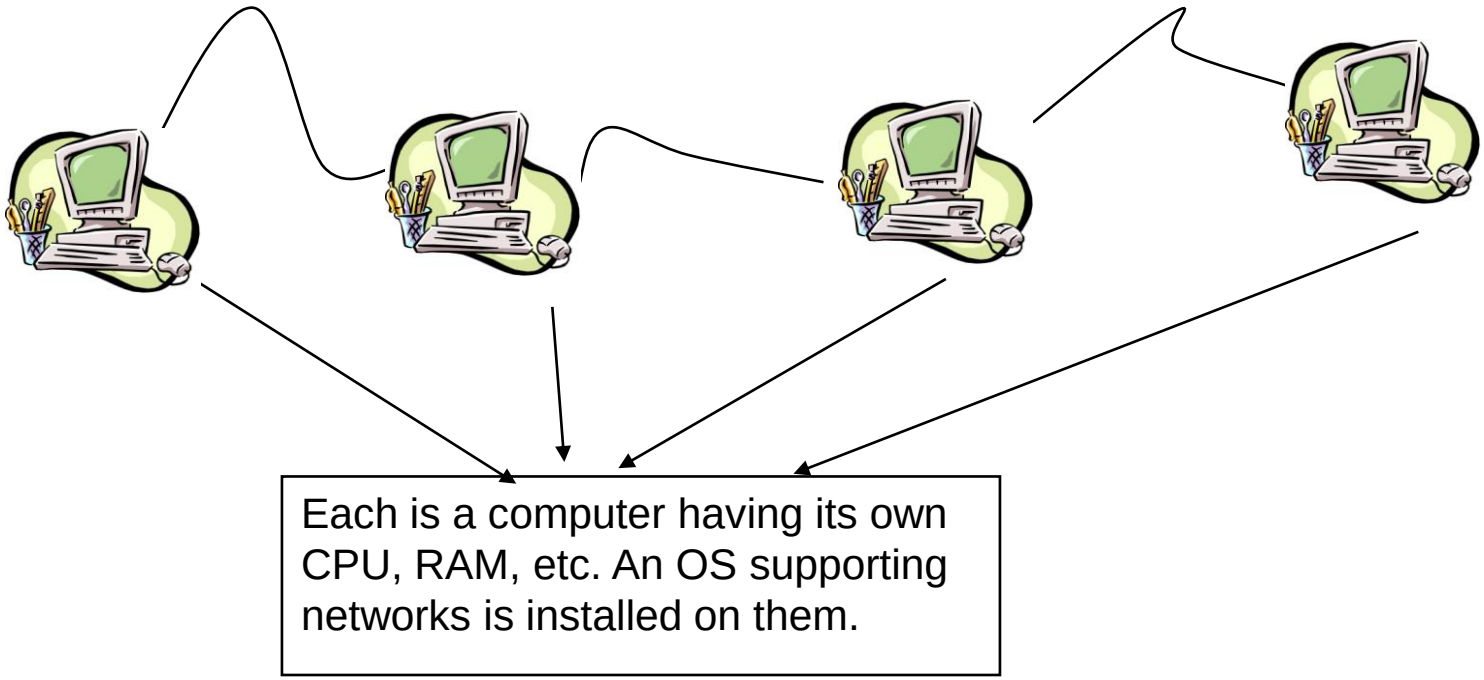
Main computer; having a CPU executing processes by utilization of the OS, (e.g. UNIX).

Terminals are connected to the main computer and used for input and output. No processing is made. They do not have CPUs.

Network Operating Systems

- Use of the networks required OSs appropriate for them.
- In **network systems**, each process runs in its own machine but the OS have access to other machines.
- By this way, file sharing, messaging, etc. became possible.
- In networks, users are aware of the fact that s/he is working in a network and when information is exchanged. The user explicitly handles the transfer of information.

Distributed Operating Systems



Distributed Operating Systems

- **Distributed systems** are similar to networks. However in such systems, there is no need to exchange information explicitly, it is handled by the OS itself whenever necessary.

- With continuing innovations, new architectures and compatible OSs are developed. But their details are not in the scope of this text since the objective here is to give only a general view about developments in OS concept.

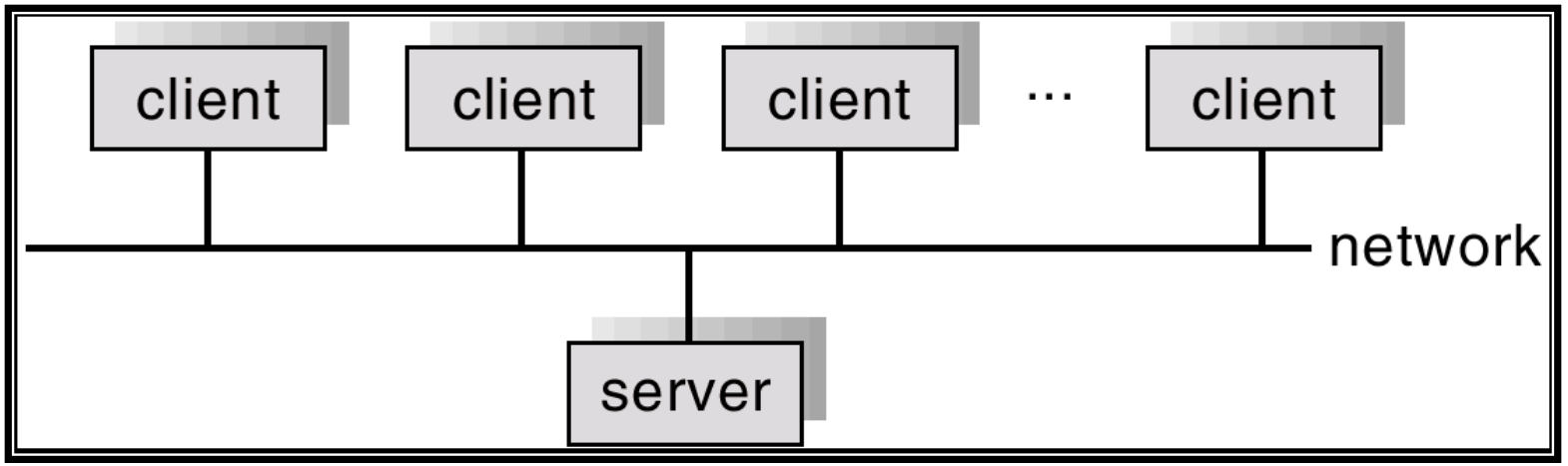
Distributed Operating Systems

- Distribute the computation among several physical processors.
- *Loosely coupled system* – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- Advantages of distributed systems.
  - Resources Sharing
  - Computation speed up – load sharing
  - Reliability
  - Communications

Distributed Operating Systems

- Requires networking infrastructure.
- Local area networks (LAN) or Wide area networks (WAN)
- May be either client-server or peer-to-peer systems.

General Structure of Client-Server



Clustered Systems

- Clustering allows two or more systems to share storage.
- Provides high reliability.
- *Asymmetric clustering*: one server runs the application while other servers standby.
- *Symmetric clustering*: all N hosts are running the application.

Real-Time Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints.
- Real-Time systems may be either *hard* or *soft* real-time.

Real-Time Systems (Cont.)

- Hard real-time:
  - Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM)
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- Soft real-time
  - Limited utility in industrial control of robotics
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.

Handheld Systems

- Personal Digital Assistants (PDAs)
- Cellular telephones
- Issues:
  - Limited memory
  - Slow processors
  - Small display screens.

OS Services

The operating system provides certain services to the program and user

- These services include

User interface

Program execution

I/O operation

File system management

Communications

Error detections and handlings

Resources allocations

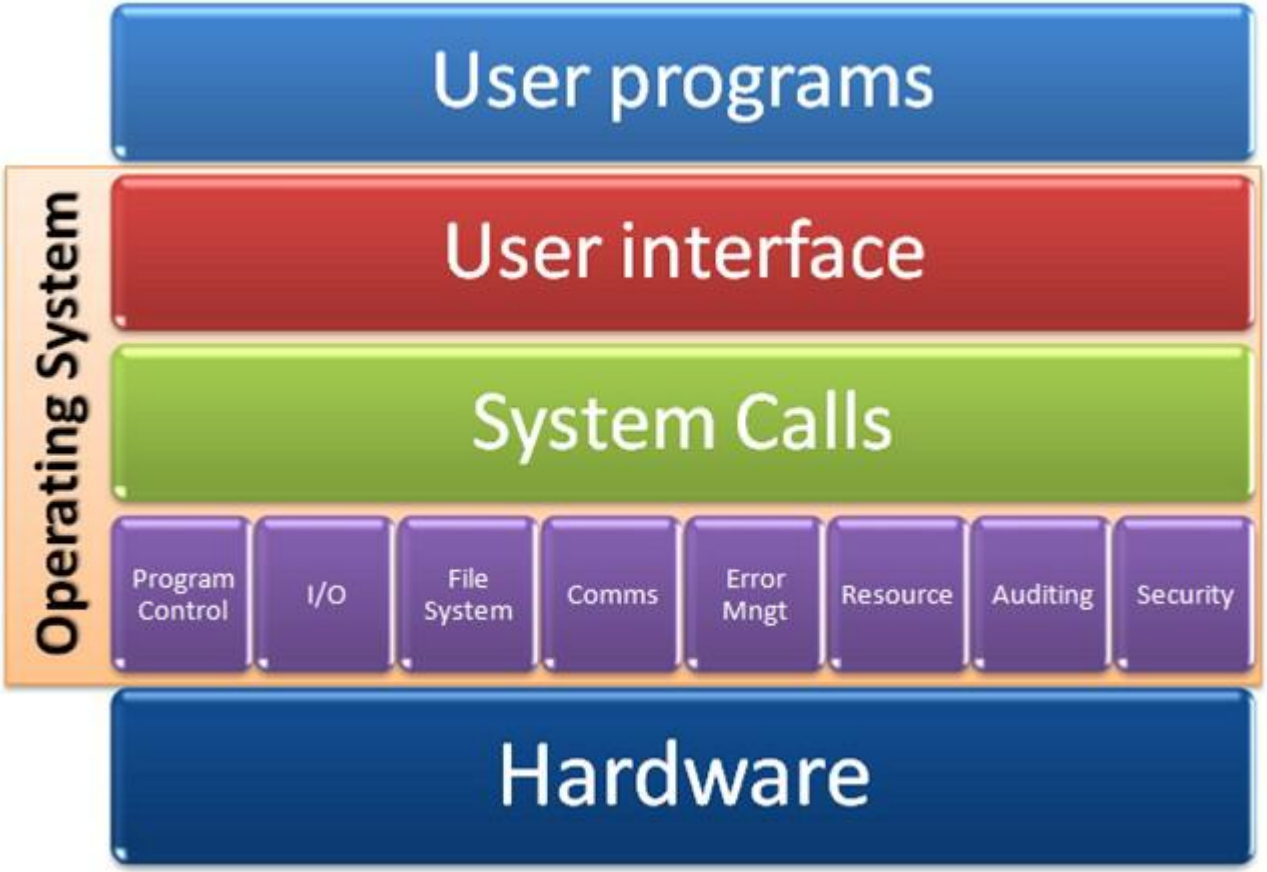
Accounting

Protection

Command interpretation

Resource management

OS Services (Cont…)



OS System Calls

- What are System Calls?
  - System Calls provide the Interface between a process and the Operating System.
  - These calls are generally available as Assembly language instruction.
  - System Calls can also be made directly through HLL programs for certain systems.
  - UNIX System calls can be invoked directly from a C or C++ program.

OS System Calls (Cont…)

- Categories of System Calls
  - System calls can be grouped into five major categories as follows.
    - Process control
    - File management.
    - Device management
    - Information Maintenance and Communication

OS System Calls (Cont…)

- Process control
  - End, abort
  - Load, execute
  - Create process, terminate process Get process, terminate process
  - Wait for time
  - Allocate and free memory
- File management
  - Create file, delete file
  - Open , close
  - Read, write, reposition.
  - Get file attributes, set file attributes

OS System Calls (Cont…)

- Device management
  - Request Device, release device
  - Read, write, reposition.
  - Get device attributes and set device attributes Logically attach or detach devices
- Information Maintenance and Communication
  - Get time or date, Set time of date
  - Logically attach or detach devices
  - Information maintenance
  - Get system data, Set Systems data
  - Get process, file of device attributes Set process, file or device attribute

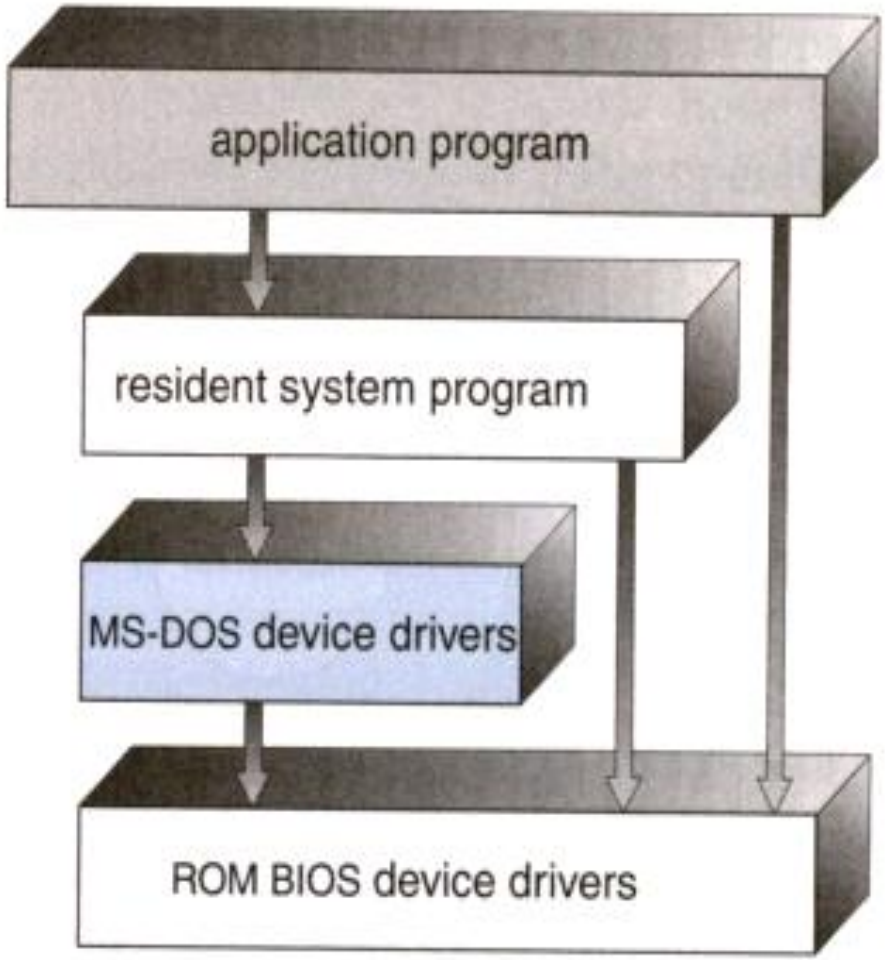
OS Structure: Simple Structure

• Operating systems such as MS-DOS and the original UNIX did not have well-defined structures.

•There is no CPU Execution Mode (user and kernel), and so errors in applications can cause the whole system to crash.

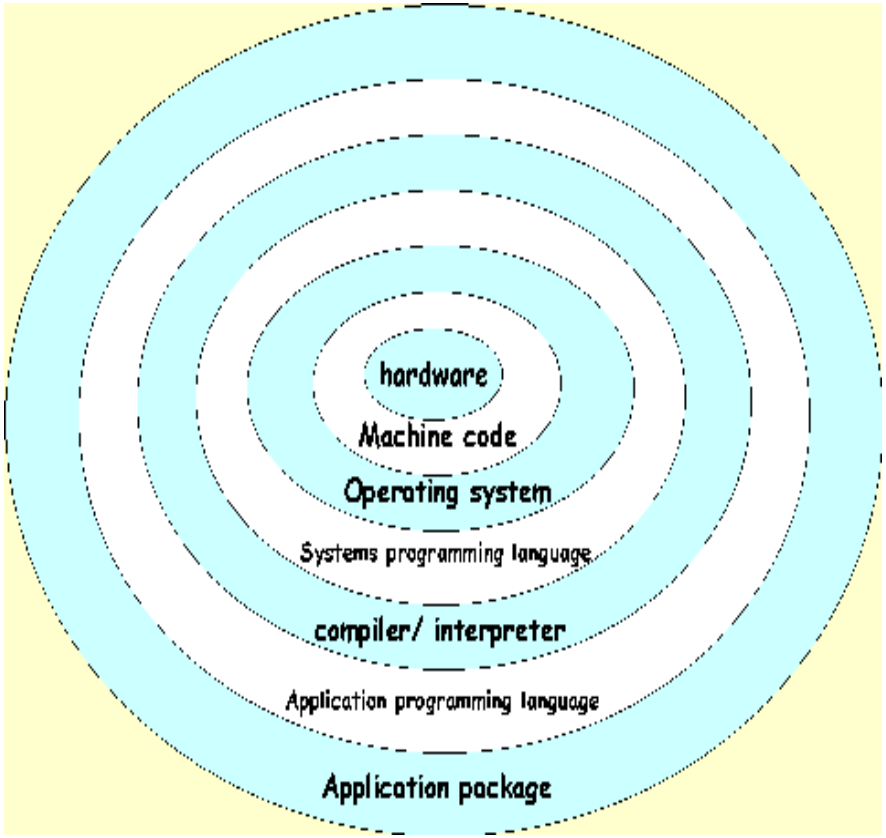
• written to provide the most functionality in the least space

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



OS Structure: Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers.
- The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface / Application Package.
- Modularity is the advantage of this layered system.
- With modularity, layers are selected such that each uses functions
- (operations) and services of only lower-level layers

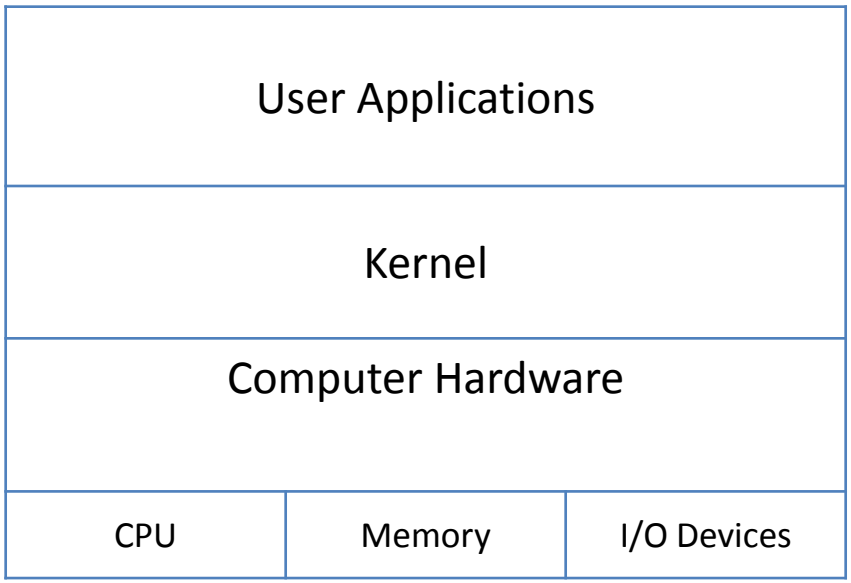


Layered Approach

- Advantages of layered apporach:
  - modularization makes debugging much easier.
  - Design and implementation makes simple.
  - it provides transparency between the layers.
- Disadvantages of layered apporach:
  - Less efficient as system call takes longer time.
  - Interaction between the layers and parameters passing is difficult.

Kernel

- is a software code that resides in the central core of a OS.
- it has complete control over system
- it is different than the shell.
- the shell is the outermost part of an OS and a program that interacts with user commands.



Kernel

- it does not interact directly with the user, but it interacts with the shell, other programs and hardware.
- when OS boots, kernel is the first part of the OS to load into memory and remains in the memory for the entire duration of the computer session.
- kernel code is usally loaded into a protected area of memory.
- kernel performs its tasks like execution processes and handling interrupts in kernel space.
- memory is divided into system area and user area
- kernel content will change according to the OS but it includes following:
  - scheduler
  - supervisor
  - Interrupt handler
  - memory manager

Types of Kernel

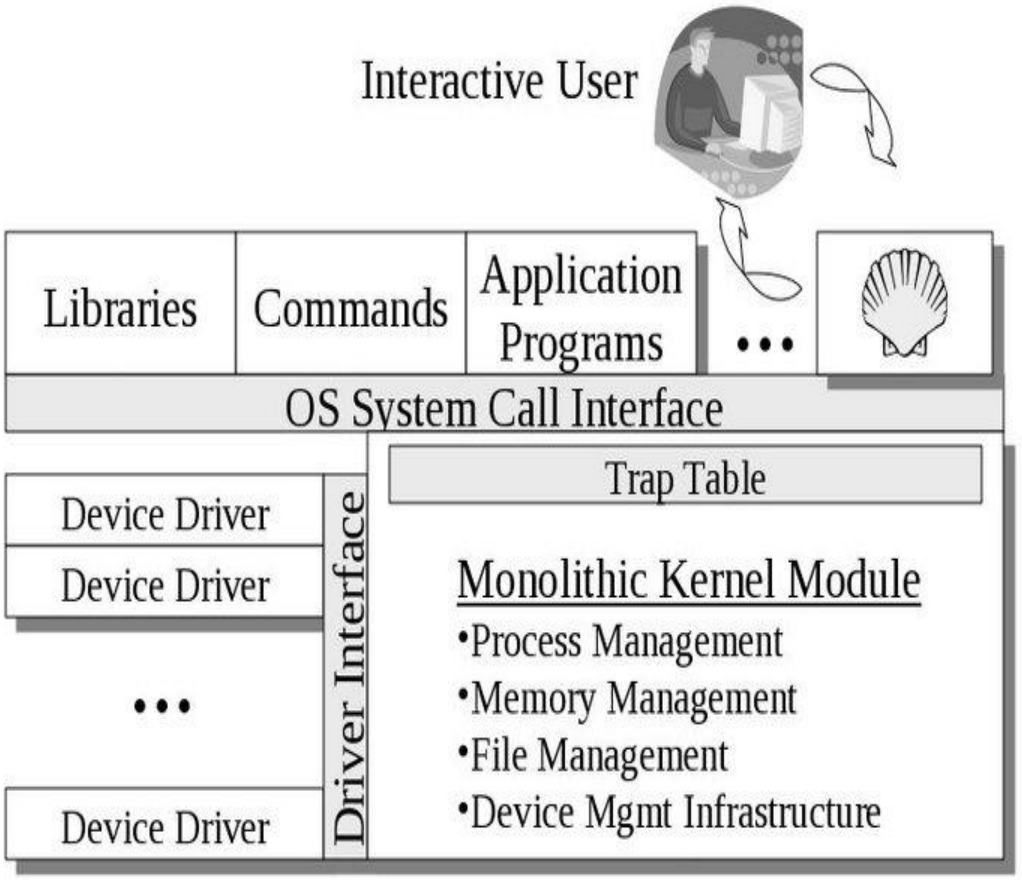
- Monolithic kernel
- Microkernel

Continue...

• Functionality of the OS is invoked with simple function calls within the kernel, which is one large program.

• Device drivers are loaded into the running kernel and become part of the kernel.

• Linux and Unix are the example of OS having this type of kernel



continue...

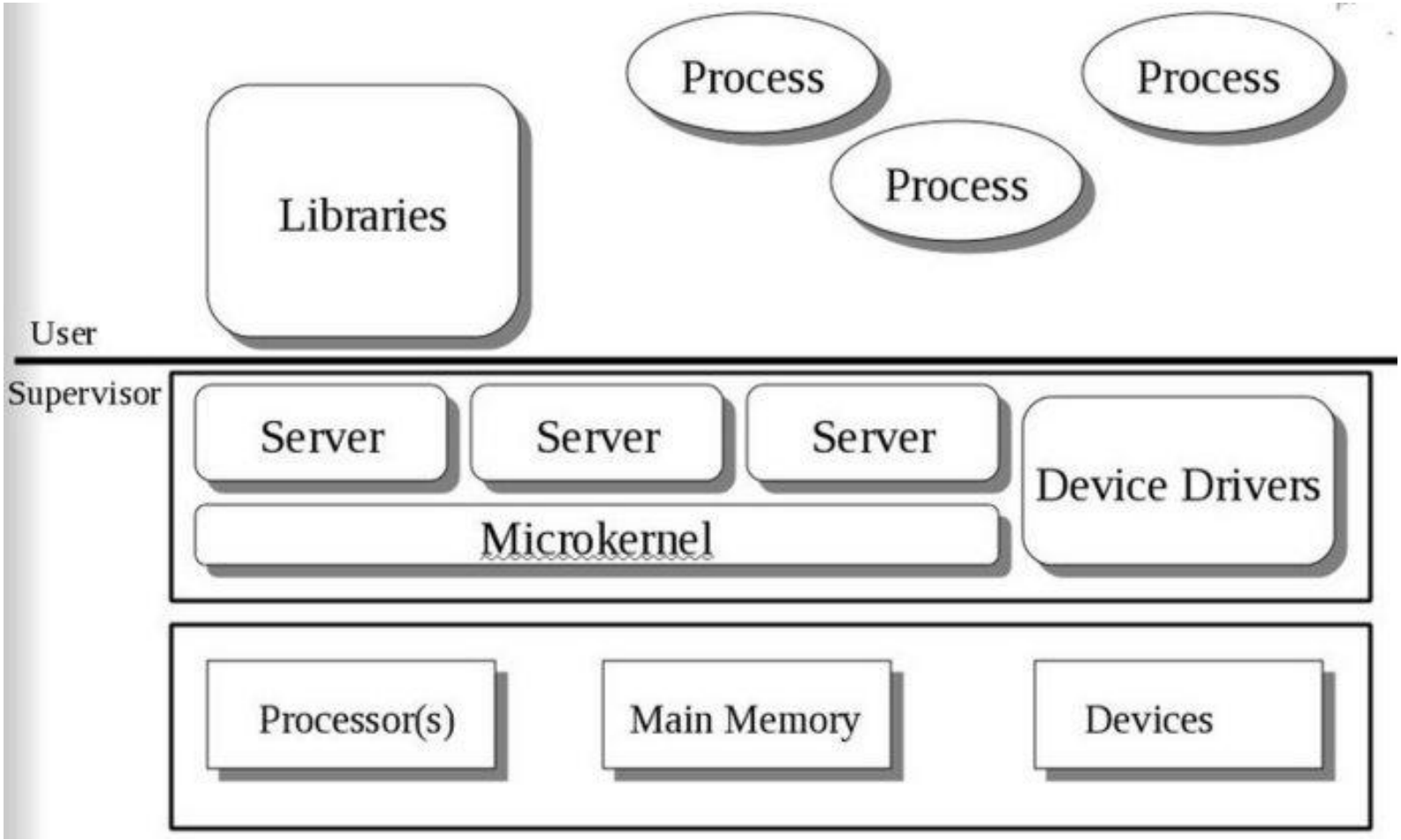
- Advantages:
  - simple to design and implement
  - simplicity provides speed on simple h/w
  - it can be expanded using moduler system.
  - time tested and design well known.
- Disadvantages:
  - runtime loading and unloading is not possible because of moduler system.
  - if code size increases, maintainance is difficult.
  - fault tolerance is low.

Microkernel

- This structures the operating system by removing all nonessential portions of the kernel and implementing them as system and user level programs.
  - Generally they provide minimal process and memory management, and a communications facility.
  - Communication between components of the OS is provided by message passing.

- The benefits of the microkernel are as follows:
  - Extending the operating system becomes much easier.
  - Any changes to the kernel tend to be fewer, since the kernel is smaller.
  - The microkernel also provides more security and reliability.
  - Main disadvantage is poor performance due to increased system overhead from message passing.

continue...


Process
Process
Process
Libraries
User
Supervisor
Server
Server
Server
Device Drivers
Microkernel
Processor(s)
Main Memory
Devices

Comparision of Monolithic and micro kernel

Sr. No.	Monolithic kernel	Microkernel
1	kernel size is large.	kernel size is small.
2	OS is complex to design	OS is easy to design, implement and install.
3.	request may be serviced faster.	request may be serviced slower than monolithic.
4.	all the OS services are included in the kernel.	kernel provides only IPC, and low level device mgmt.
5.	no message passing and no context switching are required while kernel is perfroming job.	it requires message passing and context switching.

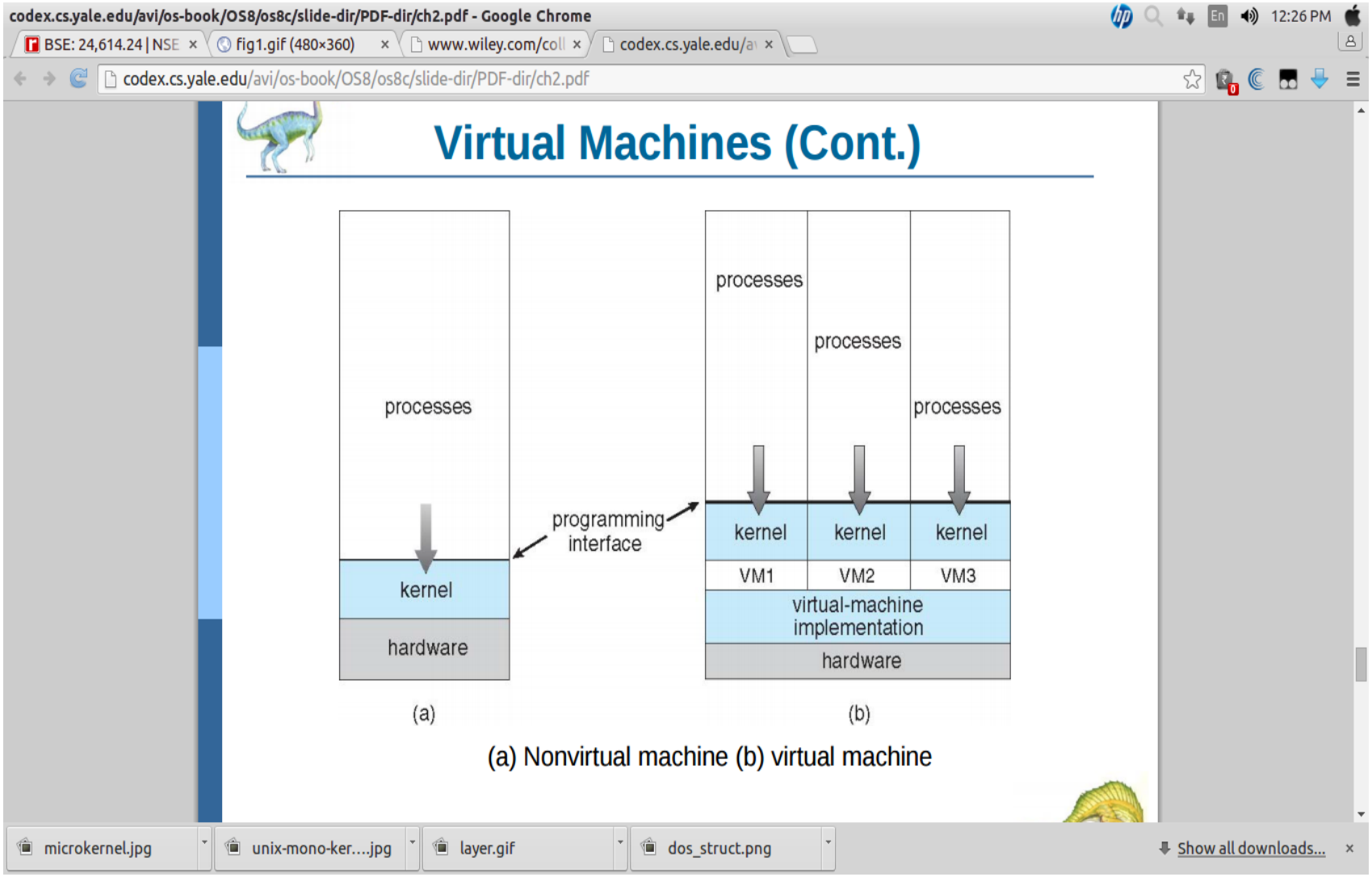
Virtual Machines

- A virtual machine takes the layered approach to its logical conclusion.
- It treats hardware and the operating system kernel as all hardware.
- A virtual machine provides an interface identical to the underlying bare hardware.
- The operating system host creates the illusion that a process has its own processor and virtual memory.
- Each guest provided with a (virtual) copy of underlying computer.

Virtual Machines History and Benefits

- First appeared commercially in IBM mainframes in **1972**
- Fundamentally, multiple execution environments (different operating systems) can share the same hardware
- Protect from each other
- Some sharing of file can be permitted,
- controlled Communication with each other, other physical systems via networking
- Useful for development, testing
- Consolidation of many low-resource use systems 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

Virtual Machines History and Benefits



Migration of Operating-System Concepts and Features