COMP-SCI-431 Intro Operating Systems

Lecture 1 – Introduction

Adu Baffour, PhD

University of Missouri-Kansas City

Division of Computing, Analytics, and Mathematics

School of Science and Engineering aabnxq@umkc.edu

# Lecture Objectives

* To understand the fundamental role of an operating system in computer systems.
* To describe the structure of an operating system, including its key components and how they interact.
* To illustrate how system calls are used to provide operating system services.
* To examine the evolution of operating systems, exploring their historical development and the current scope of their functionalities.

# Outline

1.1 The role of an operating system

1.2 The OS structure

1.3 The evolution and scope of OSs

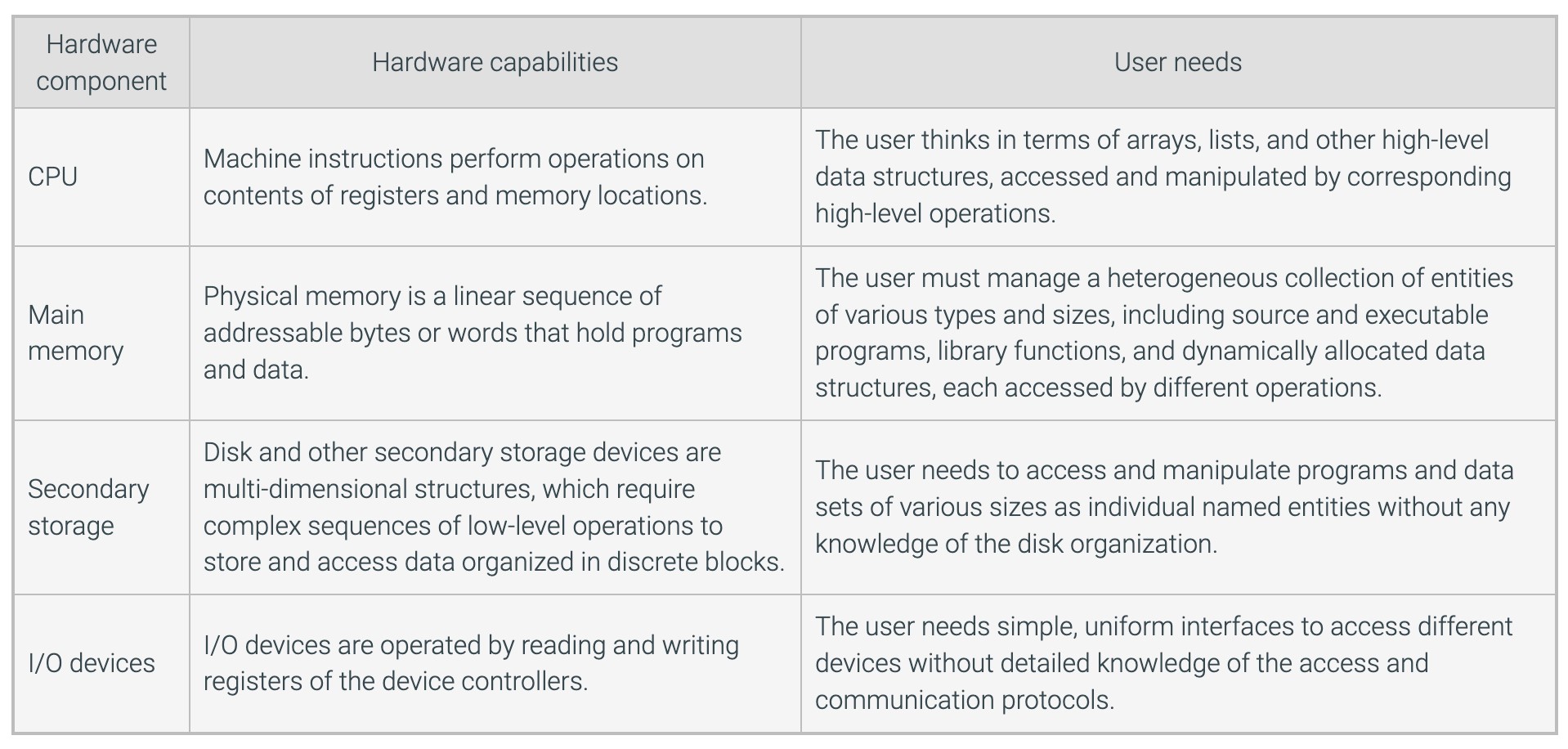
# Outline

1.1 The role of an operating system

## 1.2 The OS structure 1.3 The evolution and scope of OSs

### Bridging the hardware/user gap

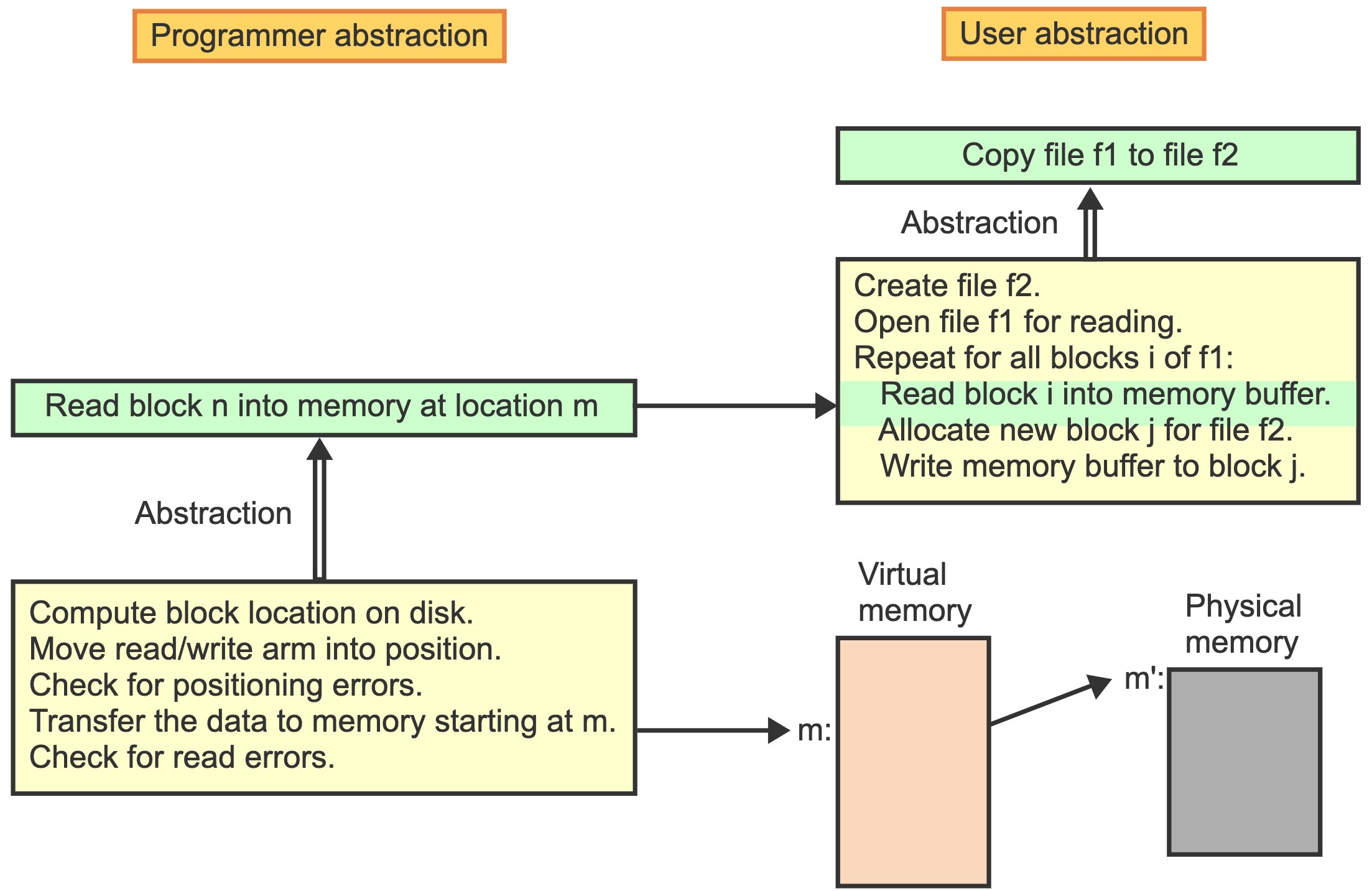
* The operating system (OS) is the software that runs on the bare hardware of a computer and provides essential support for users to develop and use applications most efficiently and safely.
* The mismatch between hardware capabilities and user needs



### The OS as an extended machine

* OSs use abstraction extensively by creating hierarchies of objects where multiple operations at one level are combined into a single operation at a higher level.
* Thus hiding the implementation details and making the operation easier to use.
* An OS provides efficient high-level functions and virtual entities that liberate the programmer and the user from understanding details of memory, disk, I/O management, and other internal processes.

Principles of abstraction and virtualization



### The OS as a resource manager

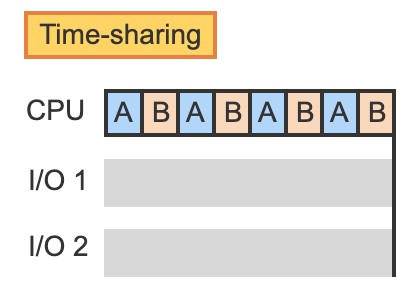
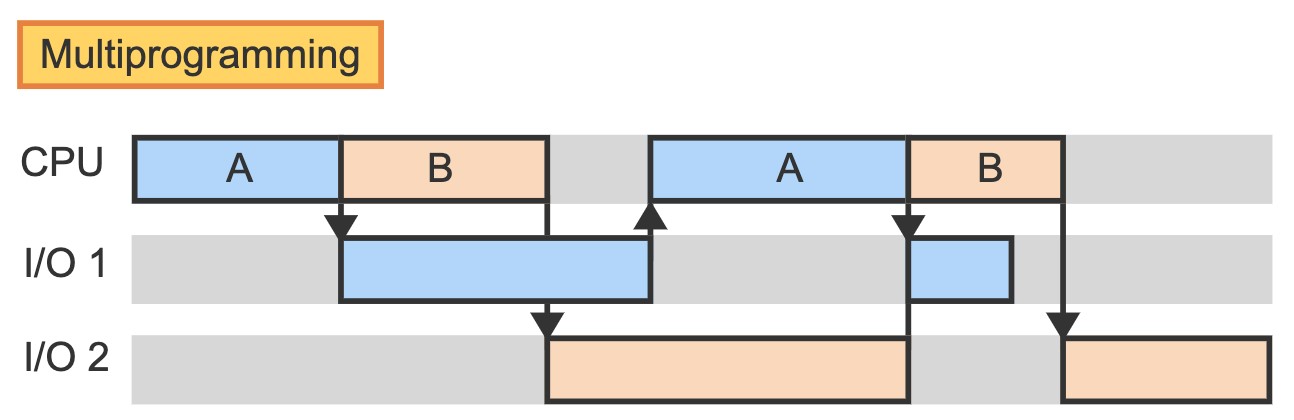
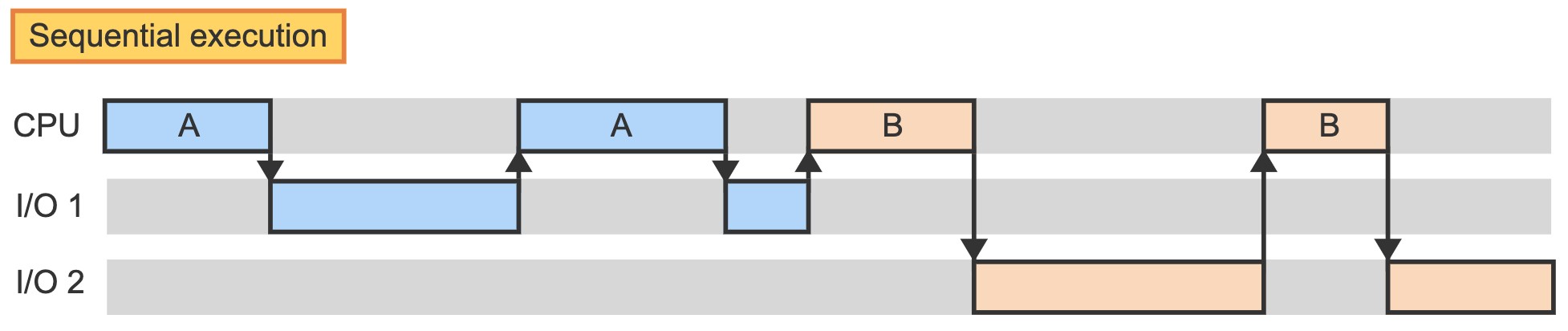
* One of the main tasks of an OS is to optimize the use of all computational resources to ensure good overall performance.
* A program typically alternates between phases of input, computation, and output.
* CPU is underutilized during the I/O phases, while the I/O devices are idle during the compute phase.
* Concurrency - the OS strives to keep the CPU, main memory, and all storage and I/O devices busy by overlapping independent operations whenever possible.

THE GOAL OF RESOURCE MANAGEMENT IS TO KEEP THE CPU BUSY WHEN THERE ARE TASKS TO BE DONE (i.e. optimize cpu resources)

### The OS as a resource manager

* Multiprogramming is a technique that keeps several programs active in memory and switches execution among the different programs to maximize the use of the CPU and other resources.
* Time-sharing (multitasking) is an extension of multiprogramming where the CPU is switched periodically among all active computations to guarantee acceptable response times to each user.

Principles of multiprogramming and time-sharing



# Outline

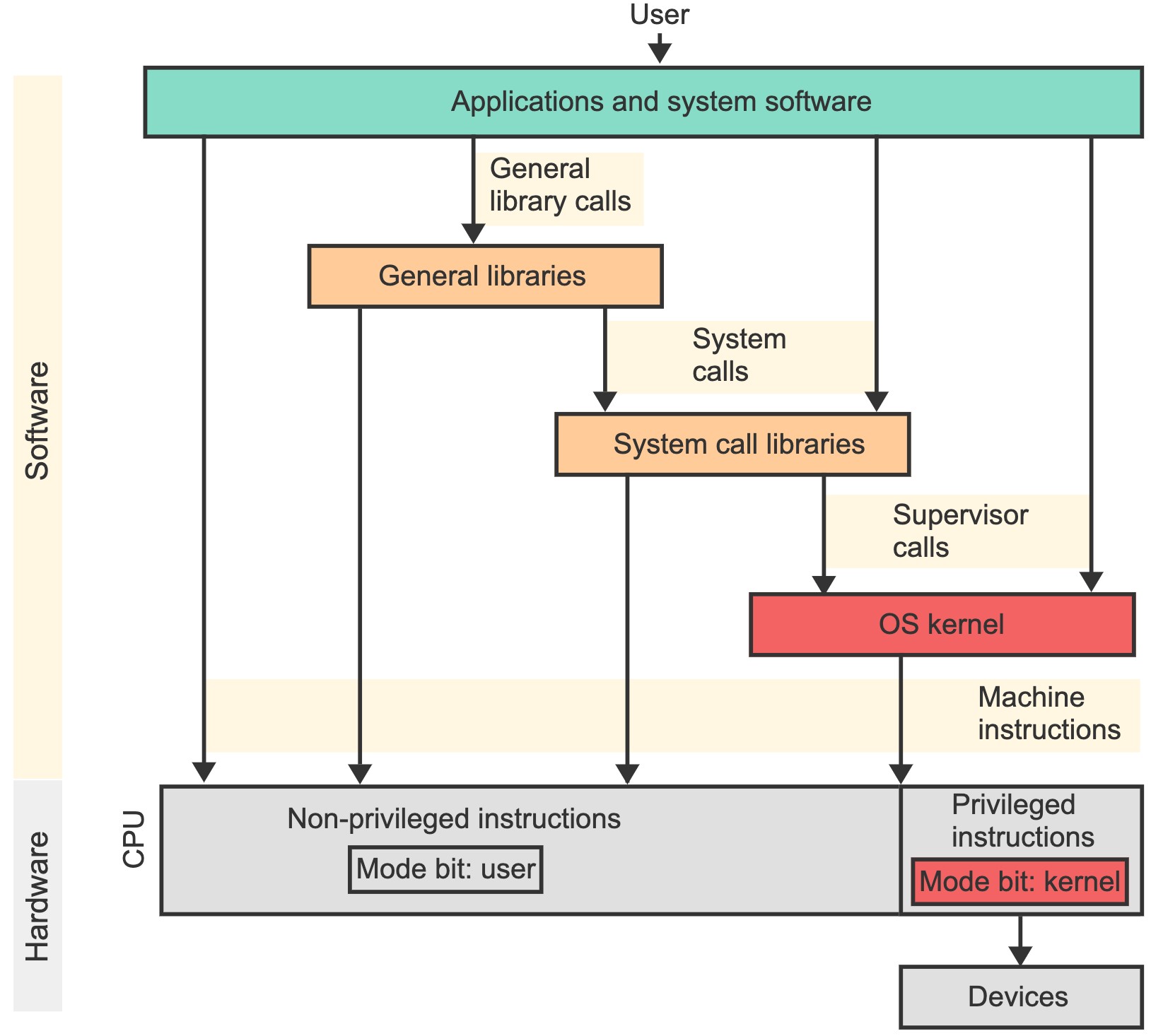
1.1 The role of an operating system

1.2 The OS structure

## 1.3 The evolution and scope of OSs

**A hierarchical organization**

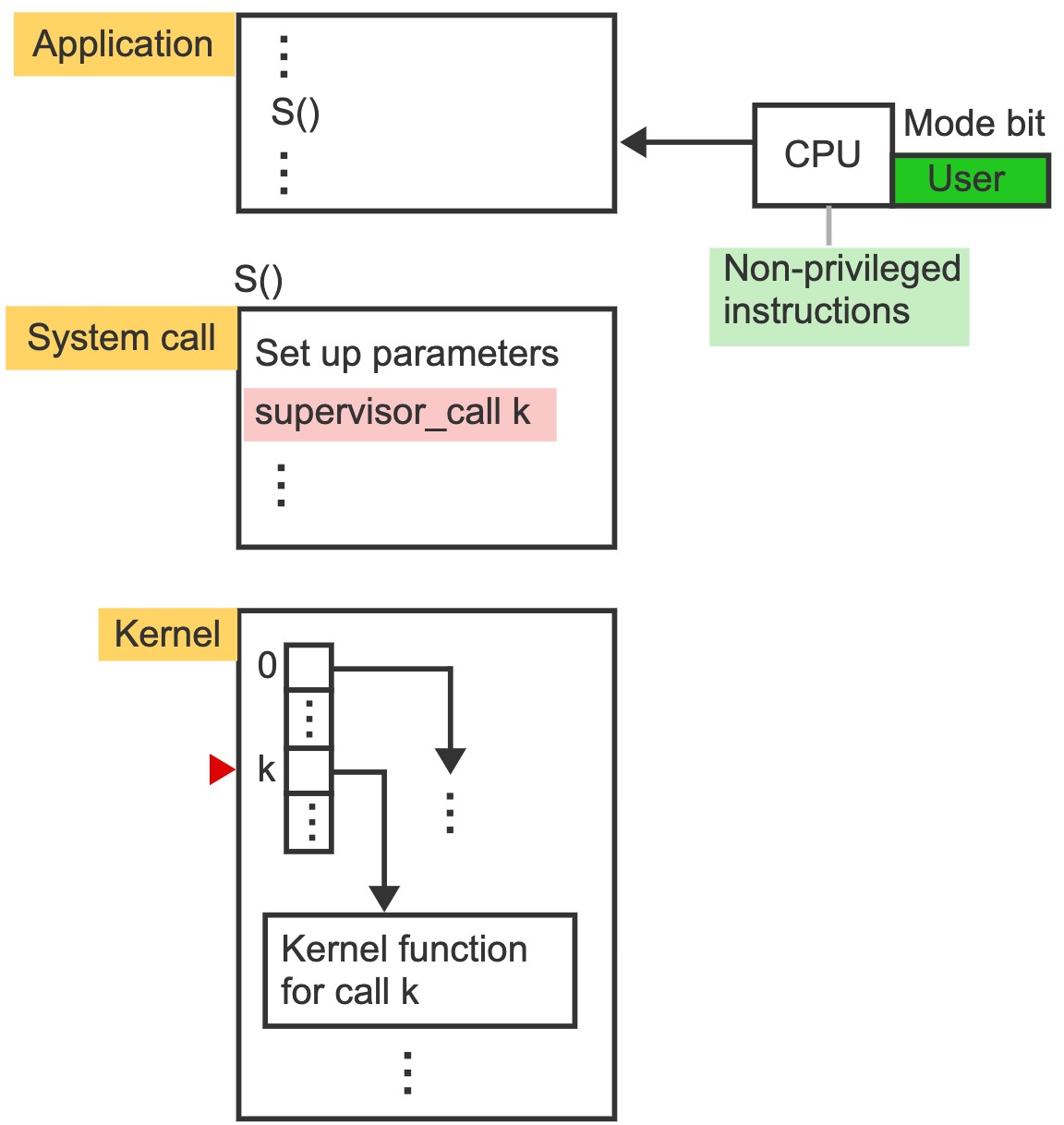
* The kernel of an OS is the minimal set of functions necessary to manage the system resources safely and efficiently.
* The CPU's instruction set is divided into privileged and non-privileged to address security issues.
* A privileged instruction performs critical operations that access I/O devices and the CPU's status and control registers. Only the OS kernel is allowed to execute privileged instructions.
* CPU operates in two different modes – kernel mode and user mode, which are indicated by a particular mode bit.
* ***Kernel mode*** is the CPU state where privileged and non-privileged instructions may be used.
* ***User mode*** is the CPU state where only non-privileged instructions may be used.
* Any attempt to execute a privileged instruction in user mode automatically transfers control to the kernel.

The OS hierarchy 

**System calls and supervisor calls.**

* A ***system call*** is a request from an application for an OS service.
* A ***supervisor call*** (***kernel call***) is a privileged instruction that automatically transfers execution control to a well-defined location within the OS kernel.
* A supervisor call is like a function call with two special features:
* The call switches execution from user to kernel mode by setting the mode bit in the CPU.
* Matches the function to be invoked using an index into a branch vector. Thus, kernel-mode execution is limited to only well-defined entry points within the kernel.
* When the kernel function terminates, control is returned to the invoking library function in user mode.

Execution of a system call



### Interrupts and traps

* An ***interrupt*** is an event triggered by an external device’s hardware signal that diverts the current execution of a program to a predefined location in the kernel to respond to an event.
* The two most common uses of interrupts are as follows:
* Signal to the OS the completion of an I/O operation. The I/O device generates the interrupt.
* Implement time-sharing. A countdown timer generates the interrupt.
* A ***trap*** (an internal interrupt) is triggered by the currently executing instruction. E.g., Dividing by zero.
* Executing a supervisor call instruction is not an error but causes a trap since the primary purpose is to transfer control to the kernel when requesting a service.
* An ***interrupt handler*** is a kernel function invoked whenever an interrupt occurs that determines the cause of the interrupt and invokes the appropriate kernel function to respond.

### Interface

* The OS starts a graphical user interface or a shell when a user logs in.
* A ***graphical user interface*** (***GUI***) presets various icons on the screen, which the user can click on in different ways to invoke services associated with the icons or to reveal pull-down menus for additional tasks.
* The ***OS shell*** is a command interpreter that accepts and interprets textual commands issued by the user via a keyboard.

# Outline

## 1.1 The role of an operating system 1.2 The OS structure

1.3 The evolution and scope of OSs

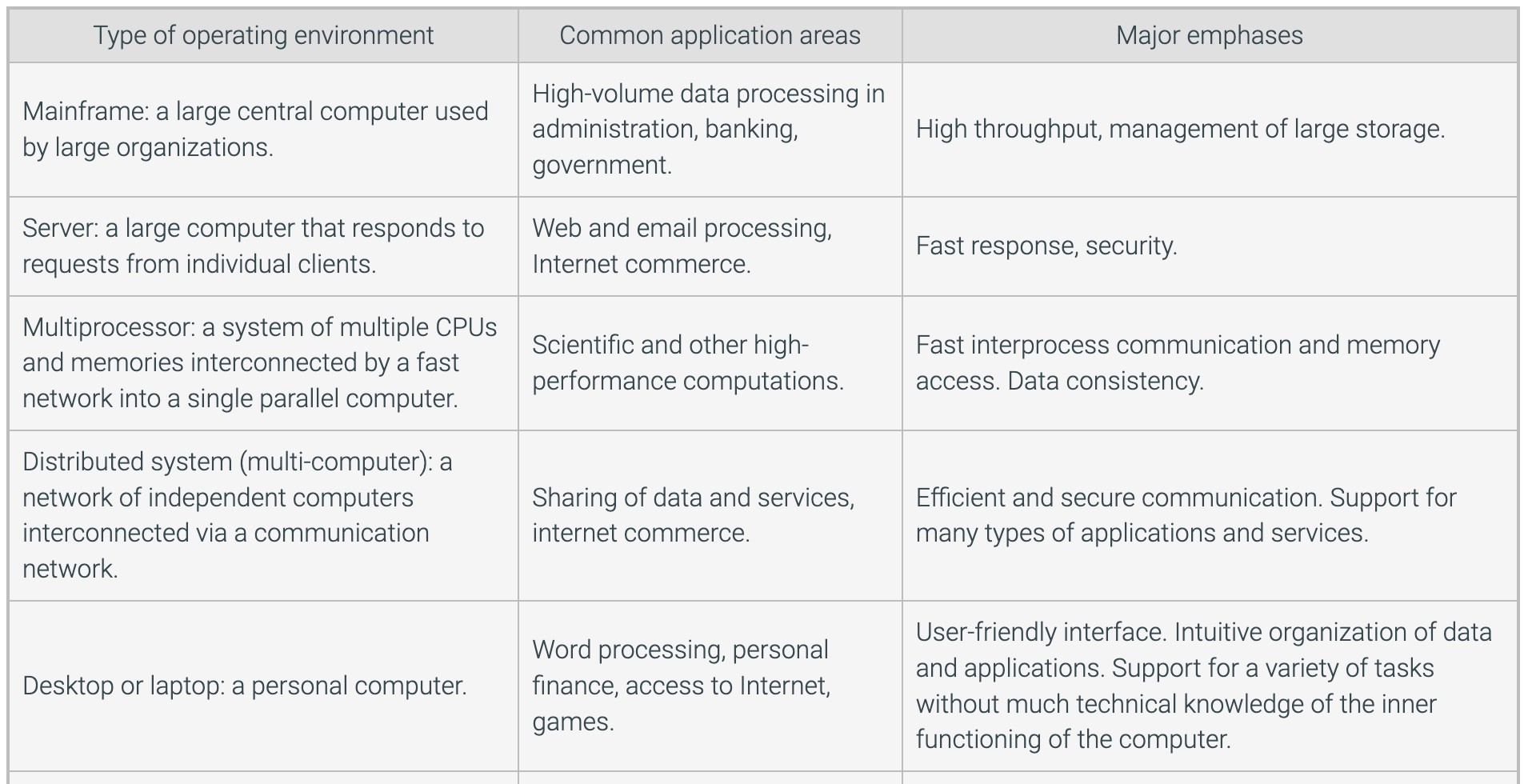
### The five generations of computer systems

* Moore's law, formulated by the scientist Gordon Moore, is the observation that the number of transistors in an integrated circuit doubles about every two years.
* 1st generation: Vacuum tubes with no OS
* 2nd generation: Transistors replaced vacuum tubes as smaller and faster switches. Batch

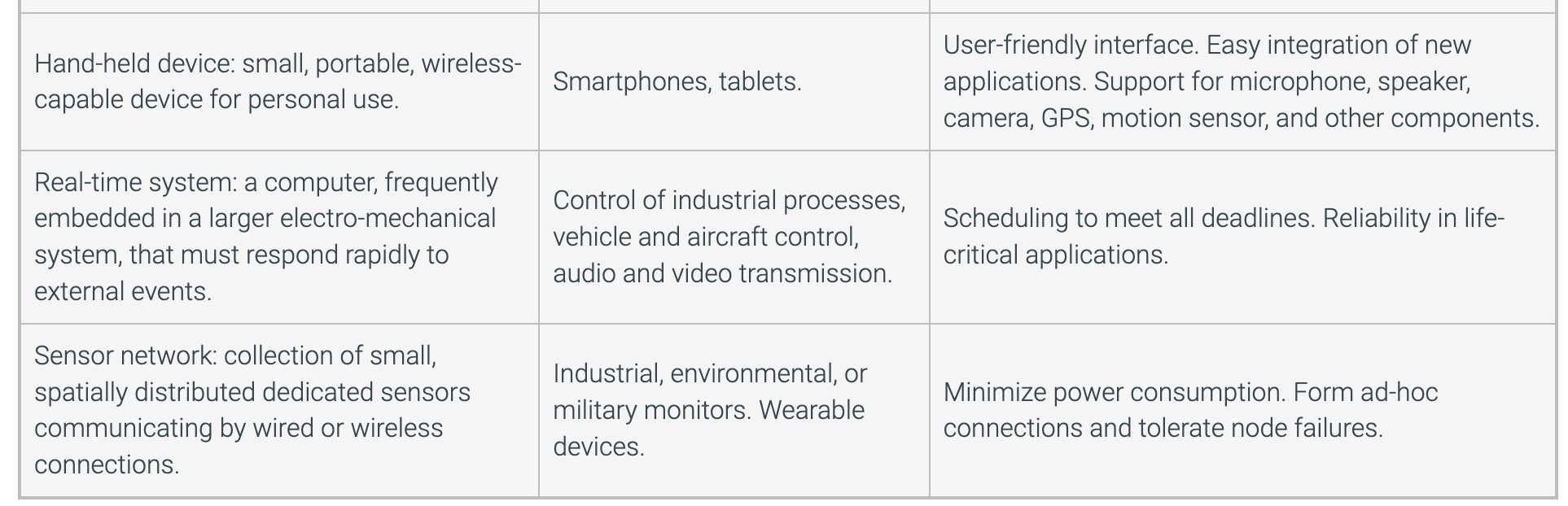
OS

* 3rd generation: Integrated circuits allowed the development of microchips to replace individual transistors—interactive multi-user OS.
* 4th generation: Very-large-scale integration (VLSI) allowed the placement of a complete microprocessor on a single chip, leading to the development of personal computers (PCs) —desktop and laptop OS.
* 5th generation: Networking hardware enabled the harnessing of the power of multiple computers —OSs for supercomputers, distributed systems, and mobile devices.

### Types of OSs



### Types of OSs



End of Lecture

Thank you

Any questions?