

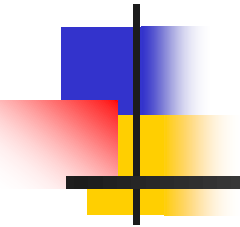


操作系统原理及应用

李 伟

xchlw@seu.edu.cn

计算机科学与工程学院、软件学院
江苏省网络与信息安全重点实验室



Chapter 1 Introduction



Outline

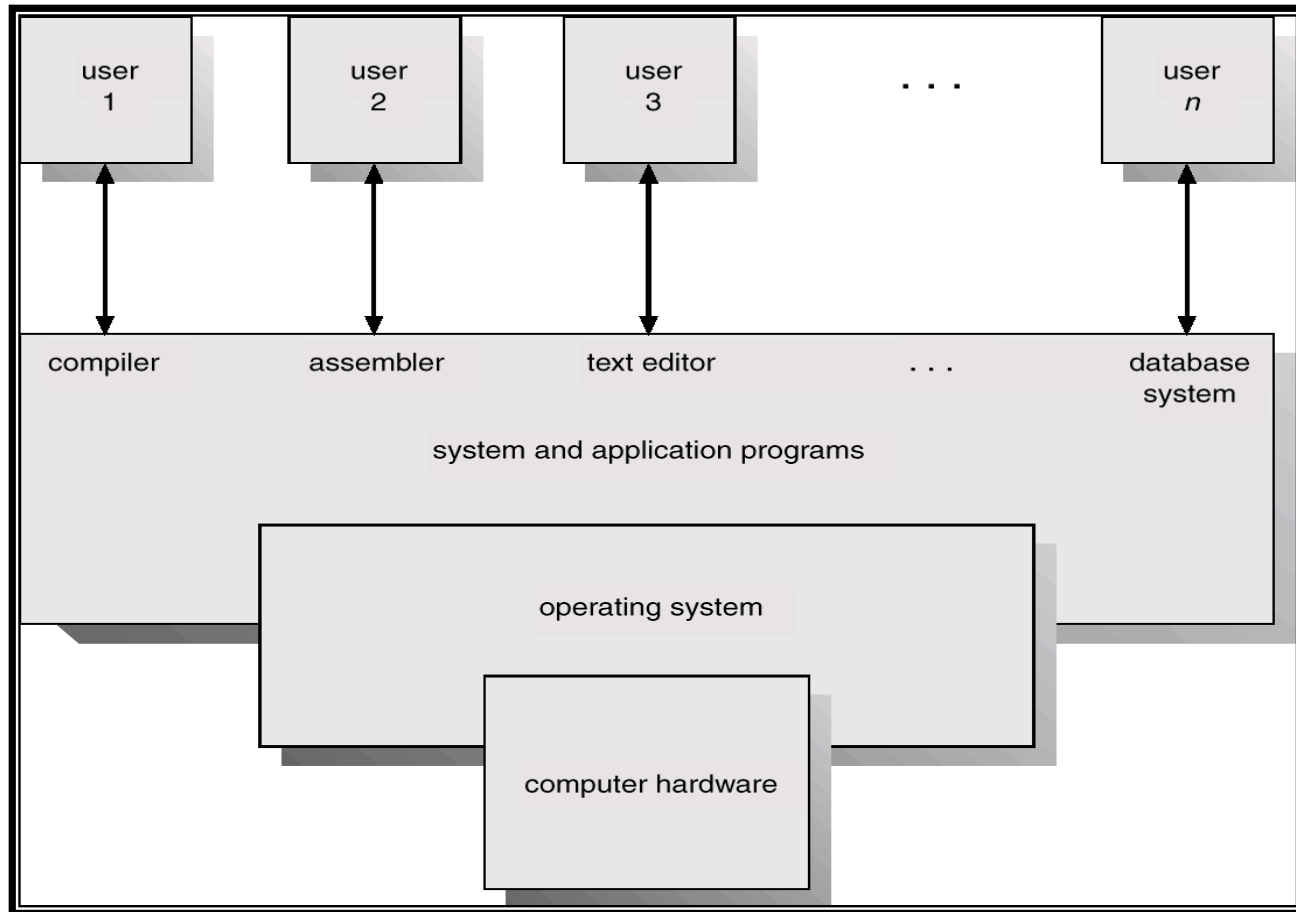
- **What is Operating System?**
- **OS-related Hardwares**
- **Development of Operating System**



Computer System Components

- **Hardware** – provides basic computing resources (CPU, memory, I/O devices, etc).
- **Operating system** – controls and coordinates the use of the hardware among the **various programs for the various users**.
- **Programs** – (compilers, database, video, games, email systems, QQ, etc) are used to solve user problems.
- **Users** (people, machines, other computers).

Abstract View of Computer System Components





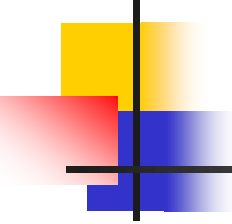
User View

- **PC Users:** an operating system is **a program** that provides an easy-to-use interface for using the hardware
- **Mainframe/Minicomputer Users:** an operating system is **a program** that helps maximize the system resource utilization
- **Workstation Users:** an operating system is **a program** designed to compromise between individual usability and resource utilization



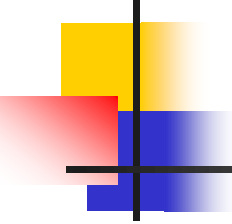
System View

- A ***Resource Allocator***: the operating system allocates and reclaims the system hardware resources to and from user programs
- A ***Control Program***: the operating system controls the execution of user programs to prevent errors and improper use of the computer



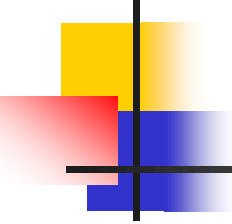
What is an Operating System?(1/3)

- There is ***no*** universal definition of what is an operating system.
- A common definition is that the operating system is the one ***kernel program*** running at all times on the computer. All other programs are systems/application programs.



What is an Operating System?(2/3)

- An operating system is **a set of programs** that acts as an intermediary between the user of a computer and the computer hardware.
 - Make the computer system convenient to use.
 - Execute user programs and make solving user problems easier.
 - Manage the computer hardware.



What is an Operating System?(3/3)

- General View

- A resource allocator, a control program, a kernel program

- The Goals of Operating System

- Primary Goal: *efficient* operation of the computer system.

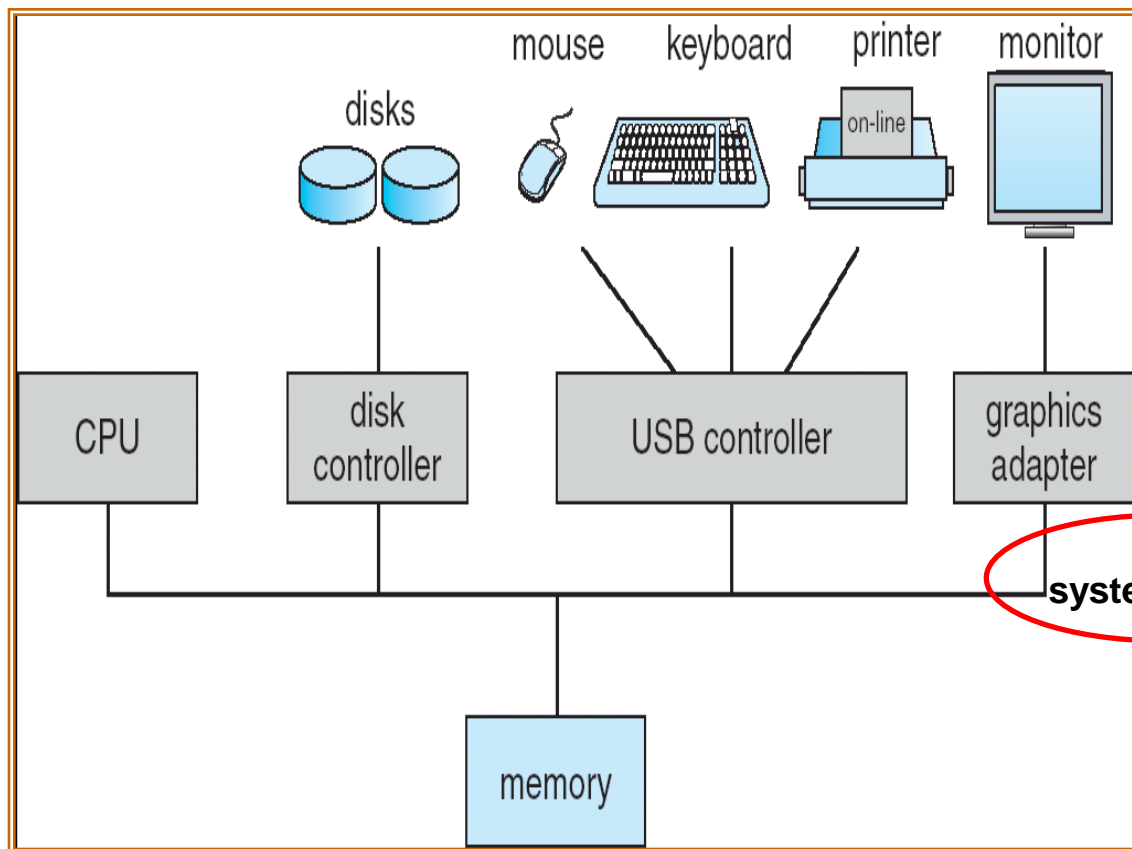
■ *The matter of what constitutes an operating system has become increasingly important.*



Outline

- **What is Operating System?**
- **OS-related Hardwares**
- **Development of Operating System**

Computer-System Architecture



- How does the OS get into system?
- How do we request for services?
- How does the OS know something has happened?



Computer Startup

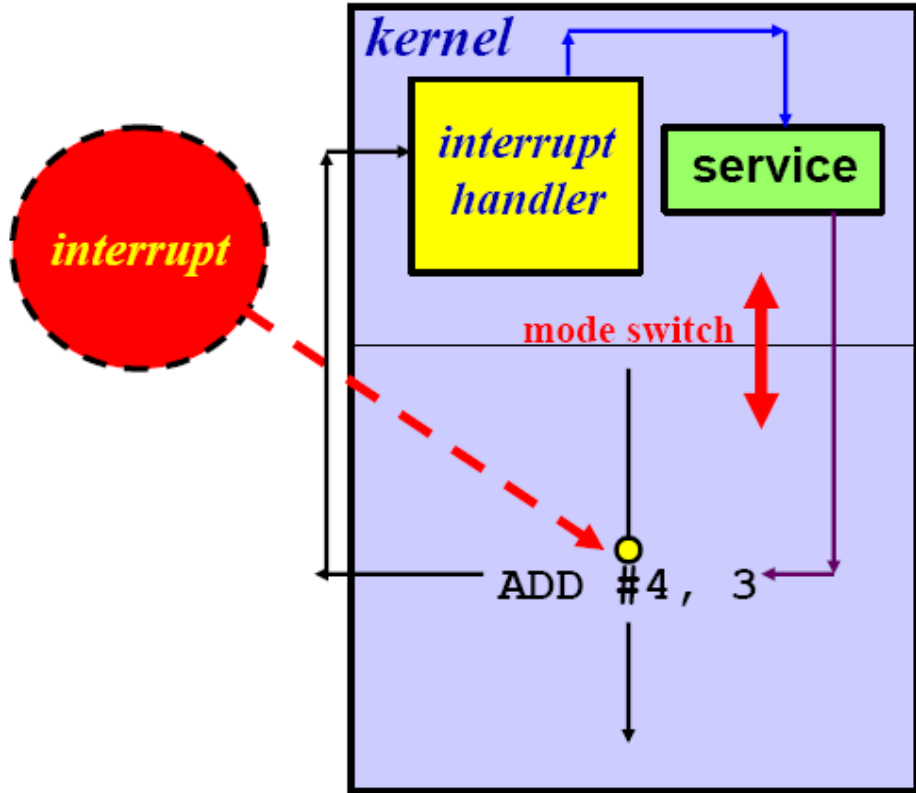
- **Bootstrap Program (引导程序)**
 - Loaded at power-up or reboot
 - Typically stored in **ROM (Read-Only Memory)** or **EPROM (Erasable Programmable ROM)** , generally known as **firmware (固件)**
 - Initializes all aspects of system
 - Loads the operating system kernel into memory, which starts executing “init” and waits for some event to occur



Interrupt

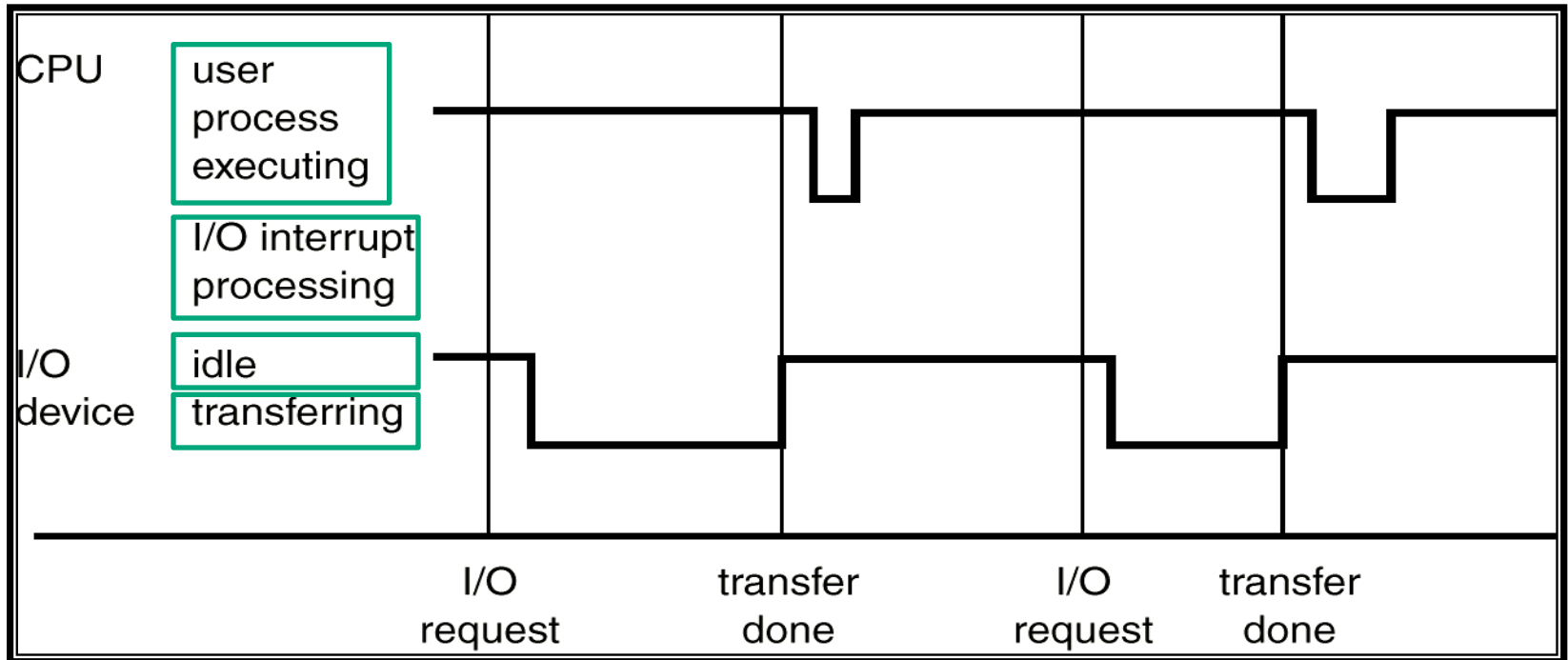
- An event that requires the attention of the OS is an ***interrupt***. These events include the completion of an I/O, a keypress, a request for service, a division by zero and so on.
- Interrupts may be generated by hardware or software. A **trap** is a software-generated interrupt caused either by an error or a user request.
- Modern operating systems are ***interrupt driven***, meaning the OS is in action only if an interrupt occurs.

What is Interrupt driven?



- OS is activated by an interrupt.
- The executing program is suspended.
- Control is transferred to OS.
- Program continues when the service completes.

Interrupt Time Line For a Single Process Doing Output





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*.



Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
 - *polling*
 - *vectored* interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

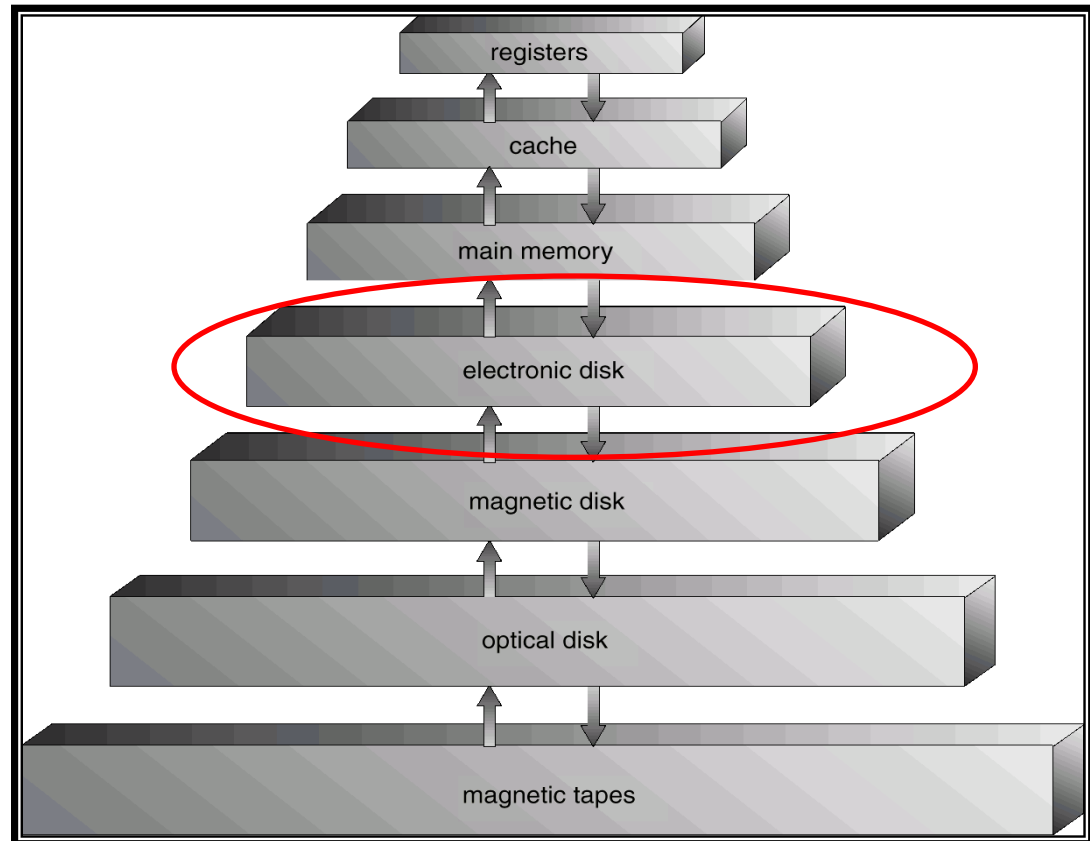


Storage Structure

- Computer programs must be in main memory to be executed.
- **Main memory** – only large storage media that the CPU can access directly.
 - too small to store all needed programs and data permanently
 - a volatile storage device
- **Secondary storage** – extension of main memory that provides large nonvolatile storage capacity.

Storage Hierarchy

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





Caching

- Use of high-speed memory to hold recently-accessed data.
- Requires a *cache management* policy.
- **Cache Coherency** ———this requires data that is simultaneously stored in more than one level to be *consistent*.



I/O Structure

- I/O devices and the CPU can execute concurrently.
- Each device controller is in charge of a particular device type and has a **local buffer**.

This form of interrupt-driven I/O is fine for moving small amounts of data but can produce high overhead when used for bulk data movement such as disk I/O.

- Device controller informs CPU that it has finished its operation by causing an **interrupt**.

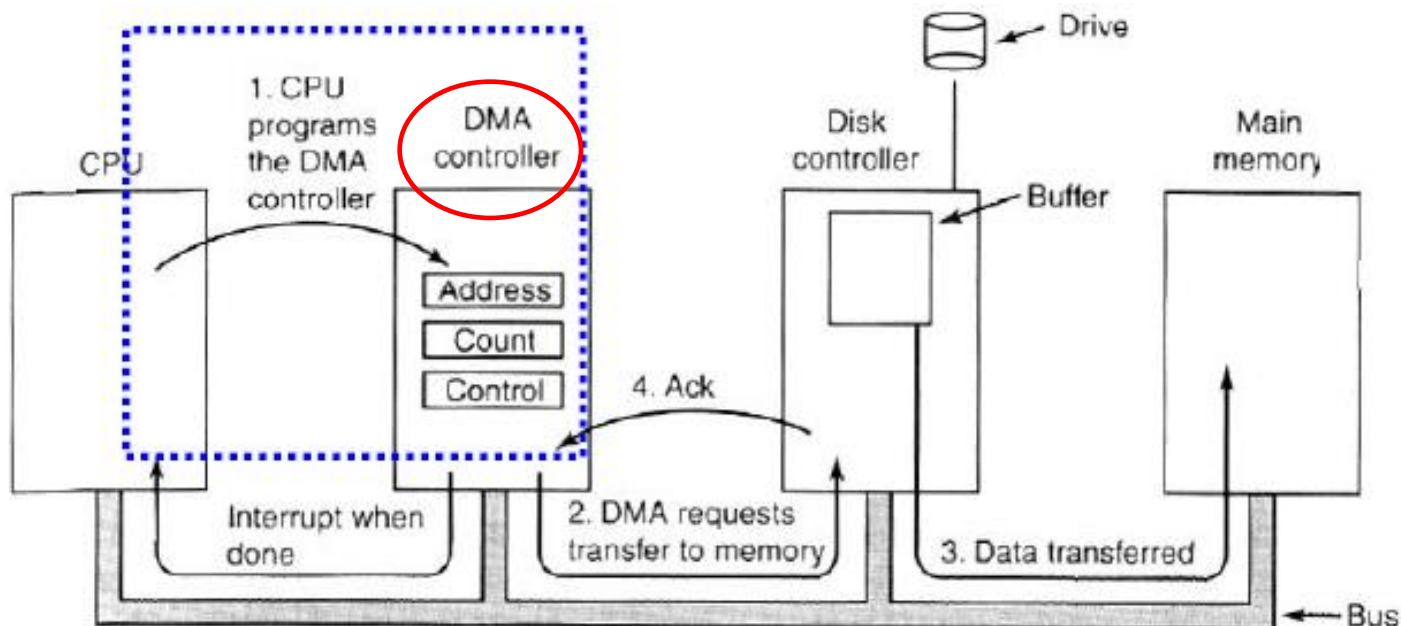


Direct Memory Access (DMA)

- Device controller directly transfers blocks of data between buffer storage and main memory **without CPU intervention.**
- Only one interrupt is generated per block, rather than the one interrupt per byte.
- Used for high-speed I/O devices able to transmit information at close to memory speeds.

Direct Memory Access (DMA)

The CPU gives the controller (1) **disk address**, (2) **memory address** for storing the block, and (3) a **byte count**. Then, the CPU goes back to work.





Hardware Protection

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



Dual-Mode Operation

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



Dual-Mode Operation

- ***Mode bit*** added to computer hardware to indicate the current mode: kernel (0) or user (1).
- When an interrupt or fault occurs, hardware switches from user mode to kernel mode.
- The dual mode of operation provides us with the means for protecting the operating system from errant user programs.

Privileged Instruction —Some of the machine instructions that may ***cause harm*** and can be executed only in ***kernel mode***.

user process

user process executing

calls system call

return from system call

user mode
(mode bit = 1)

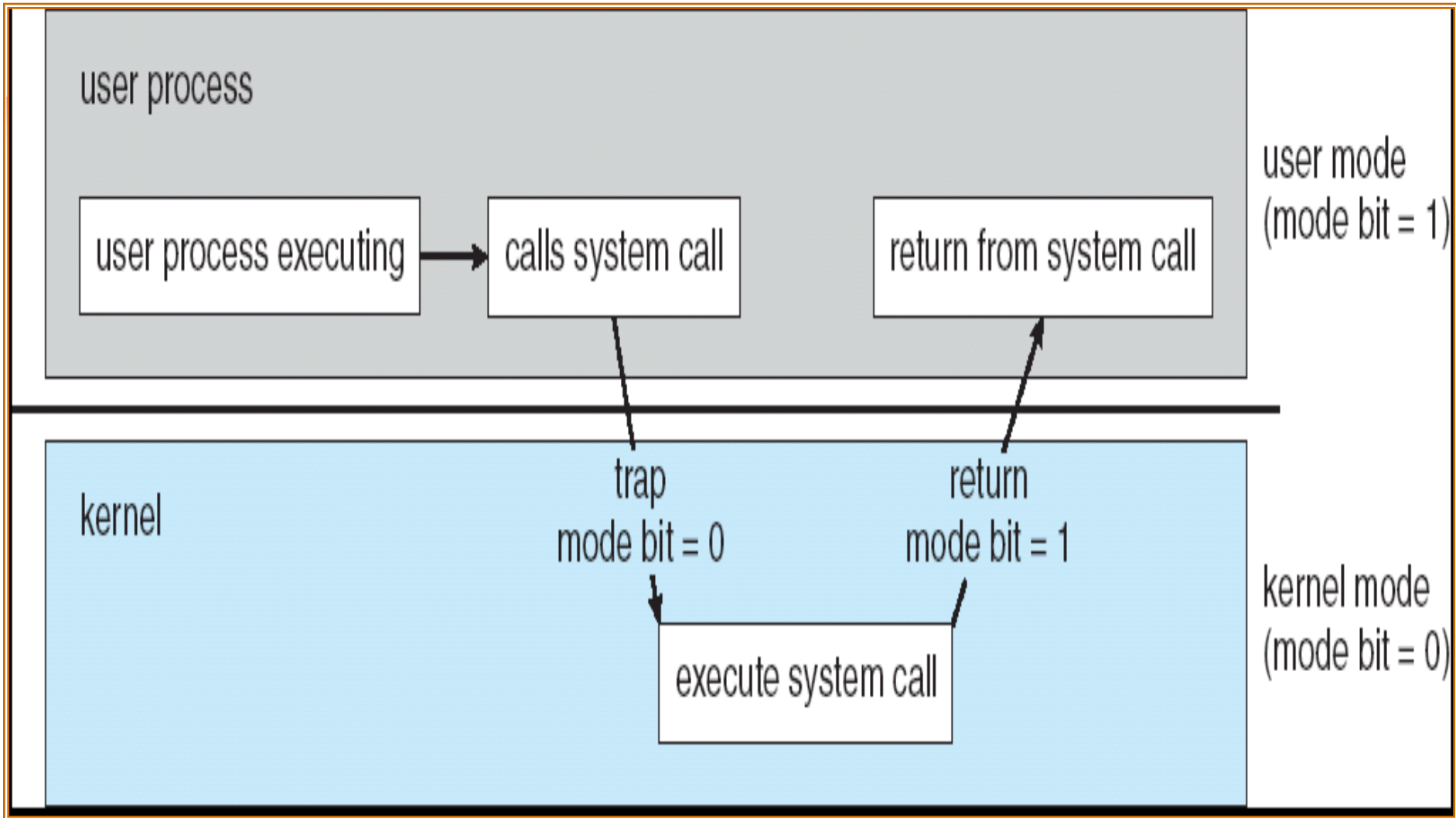
kernel

trap
mode bit = 0

return
mode bit = 1

execute system call

kernel mode
(mode bit = 0)





CPU Protection

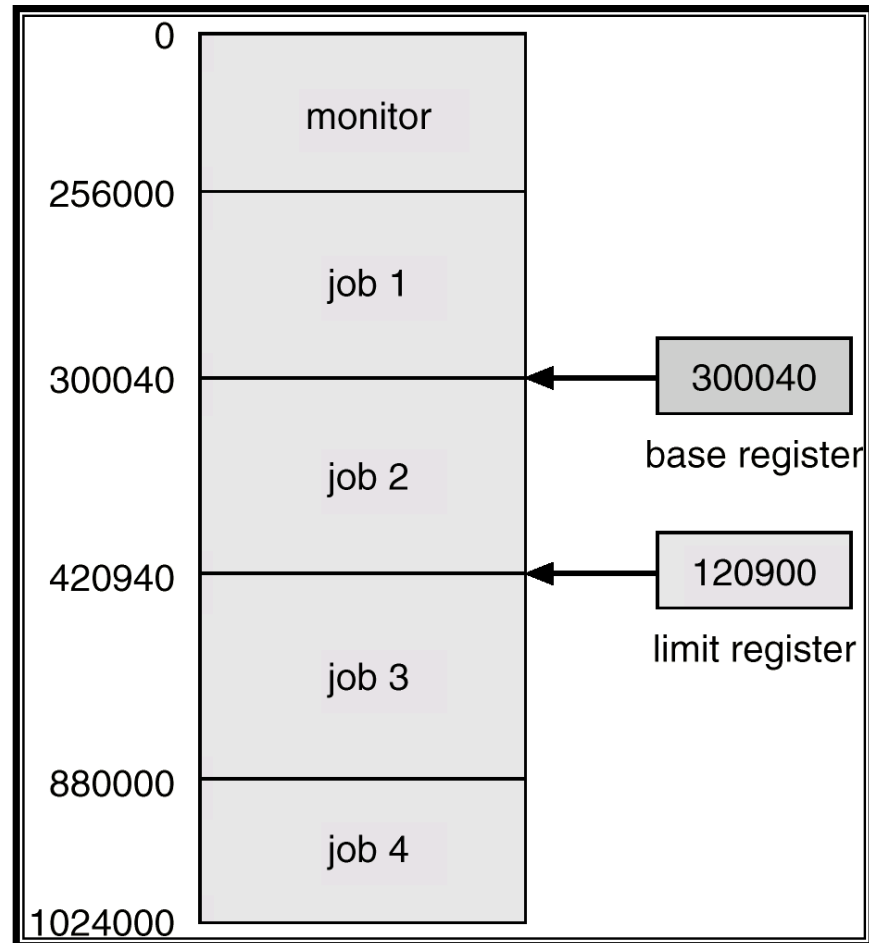
- ***Timer*** – interrupts computer after specified period to ensure operating system maintains control.
 - Timer is decremented every clock tick.
 - When timer reaches the value 0, an interrupt occurs.
- Timer commonly used to implement time sharing.
- Timer also used to compute the current time.
-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



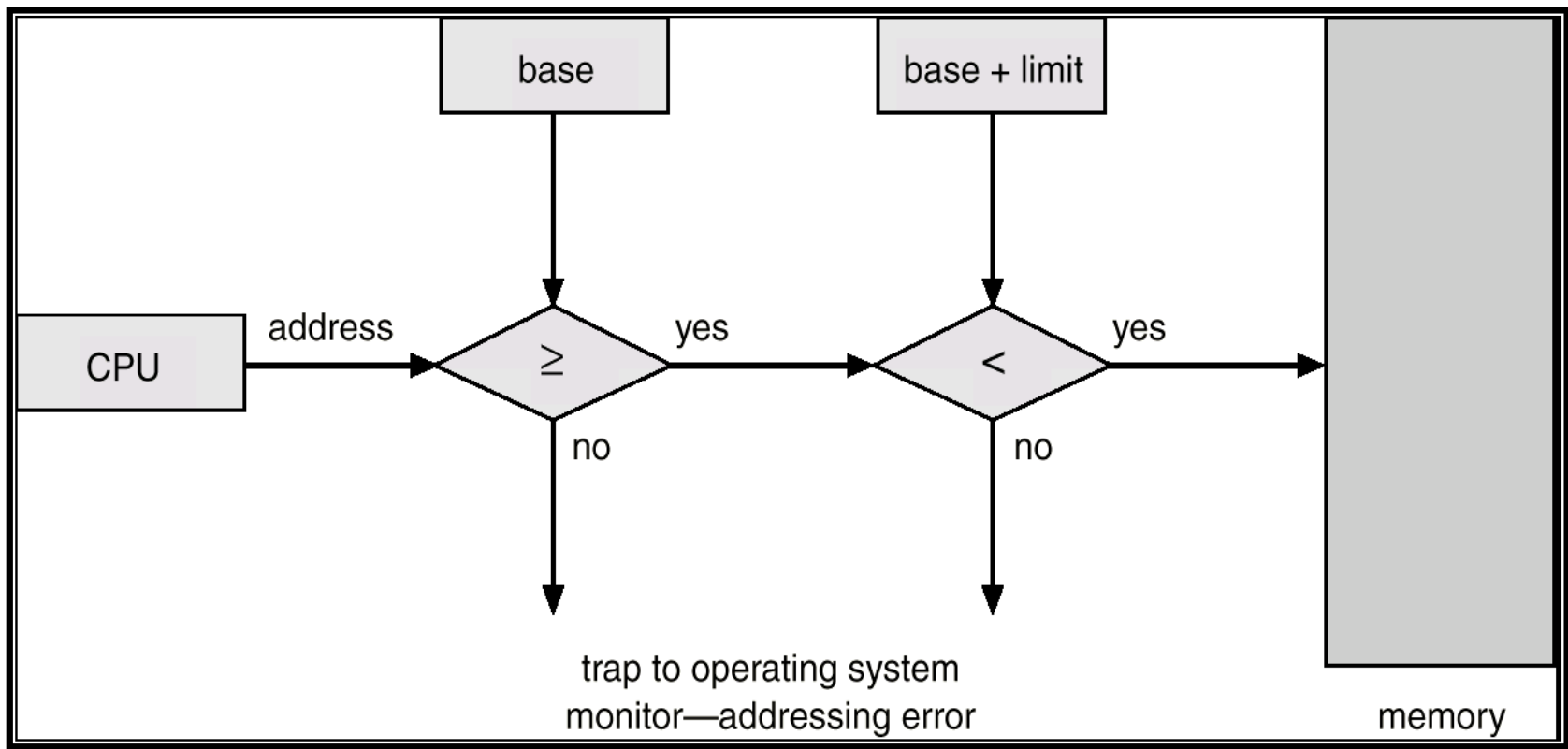
Memory Protection

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

Use of a Base and Limit Register



Memory Address Protection





Memory Protection

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



I/O Protection

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



Outline

- **What is Operating System?**
- **OS-related Hardwares**
- **Development of Operating System**

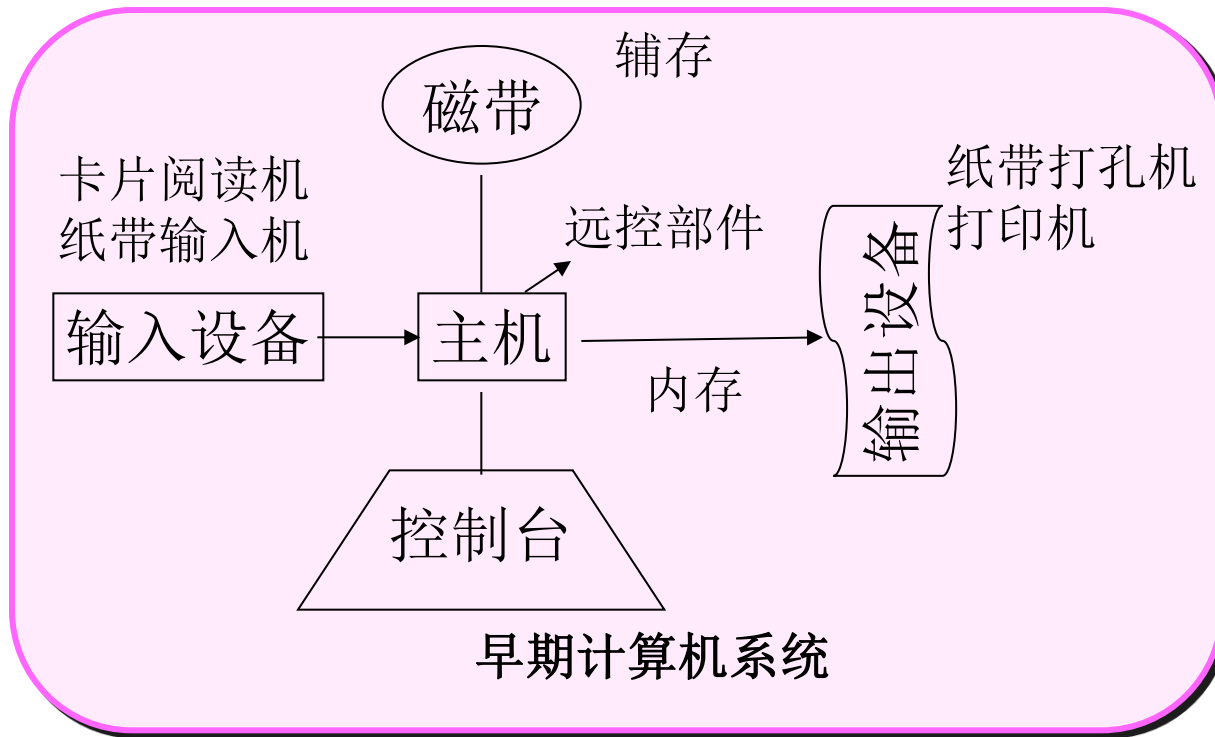


Development of OS

- **Mainframe Systems**
- **Desktop Systems**
- **Multiprocessor Systems**
- **Distributed Systems**
- **Clustered Systems**
- **Real-Time Systems**
- **Handheld Systems**

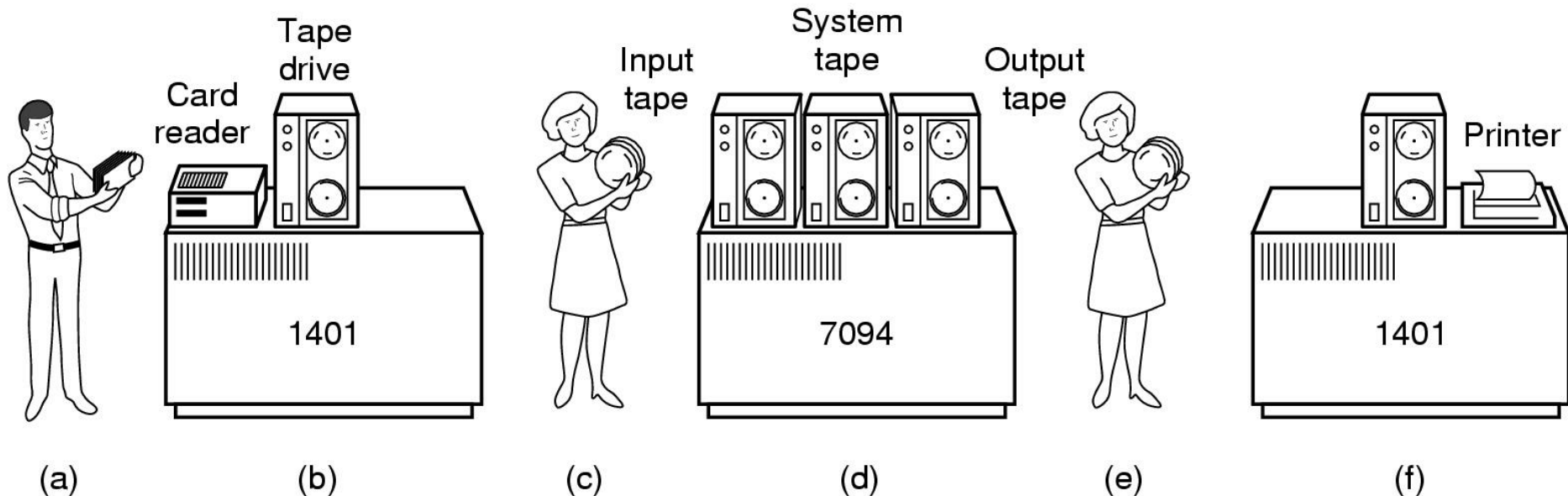
Mainframe Systems

No OS → Batch Systems → Multiprogramming Systems → Time Sharing Systems



- 用户独占全机资源
- 程序运行前准备时间过长
- 人机速度不匹配

Batch Systems (1/2)



- bring cards to IBM 1401
- IBM 1401 read cards to tape
- put tape on IBM 7094 which does computing
- put tape on IBM 1401 which prints output



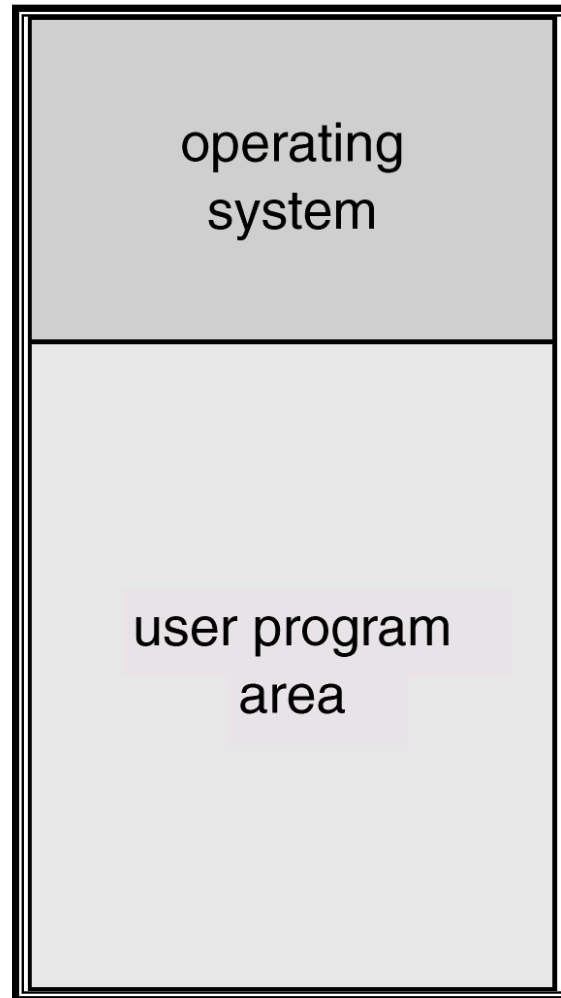
Batch Systems(2/2)

- Reduce setup time by batching similar jobs
- Automatic job sequencing – automatically transfers control from one job to another.
- Resident monitor
 - initial control in monitor
 - control transfers to job
 - when job completes control transfers back to monitor

First rudimentary operating system

Memory Layout for a Simple Batch System

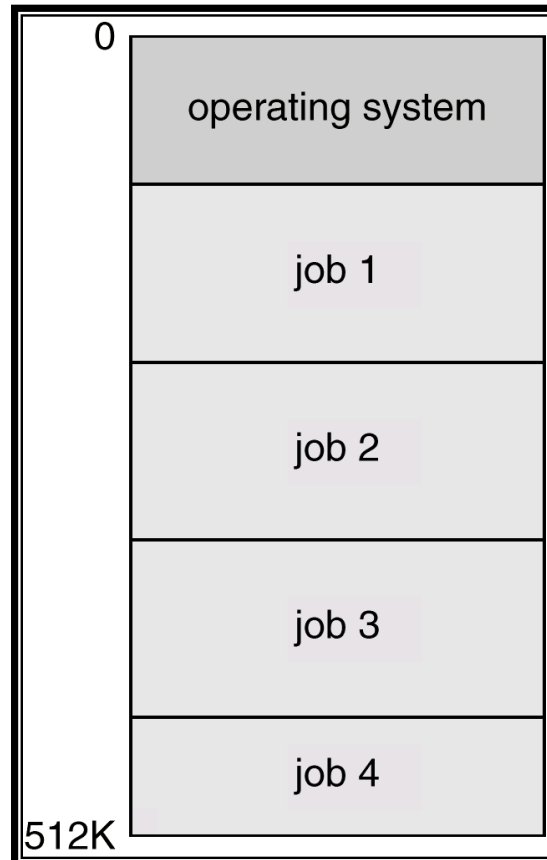




Multiprogramming Batch Systems

Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.

- Keeps several jobs in memory simultaneously
- Picks and begins to execute one
- If that job needs to wait, CPU is switched to another one



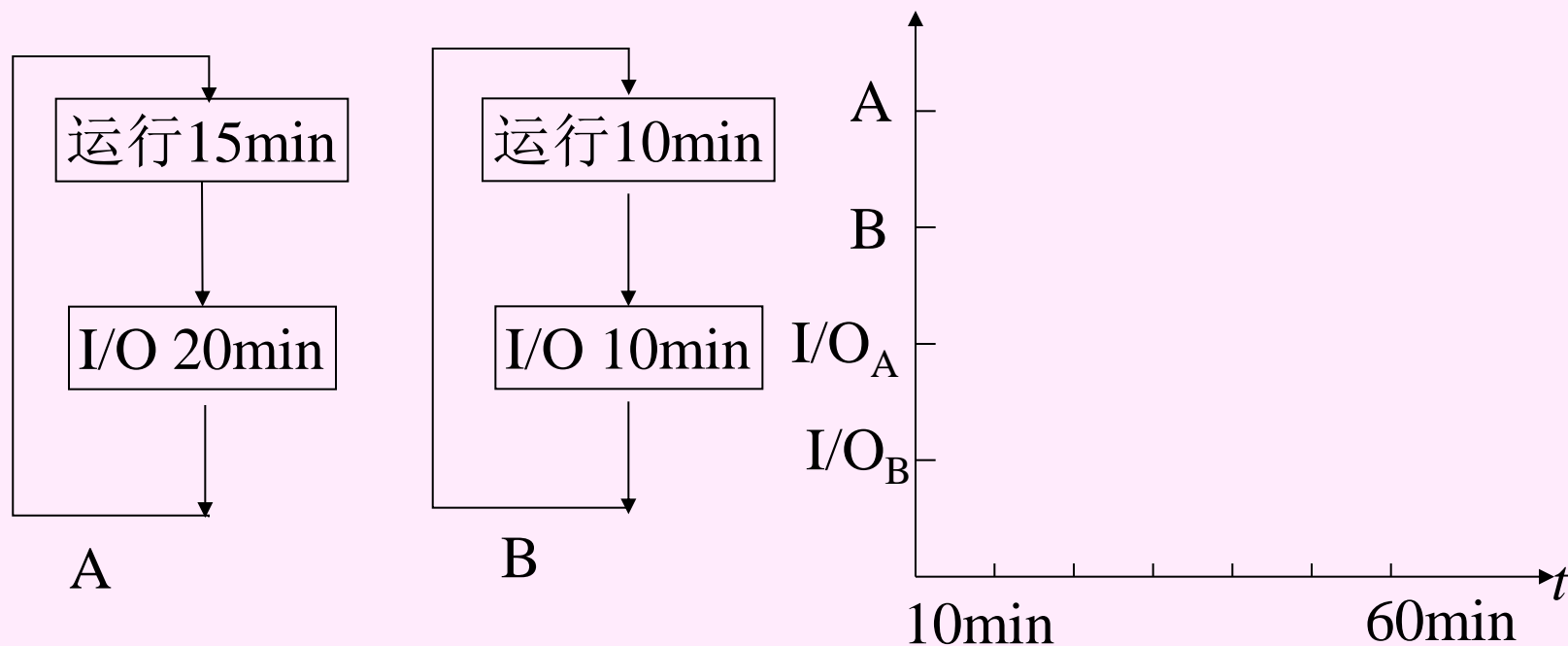
It's the first instance where the operating system must **make decisions for the users**

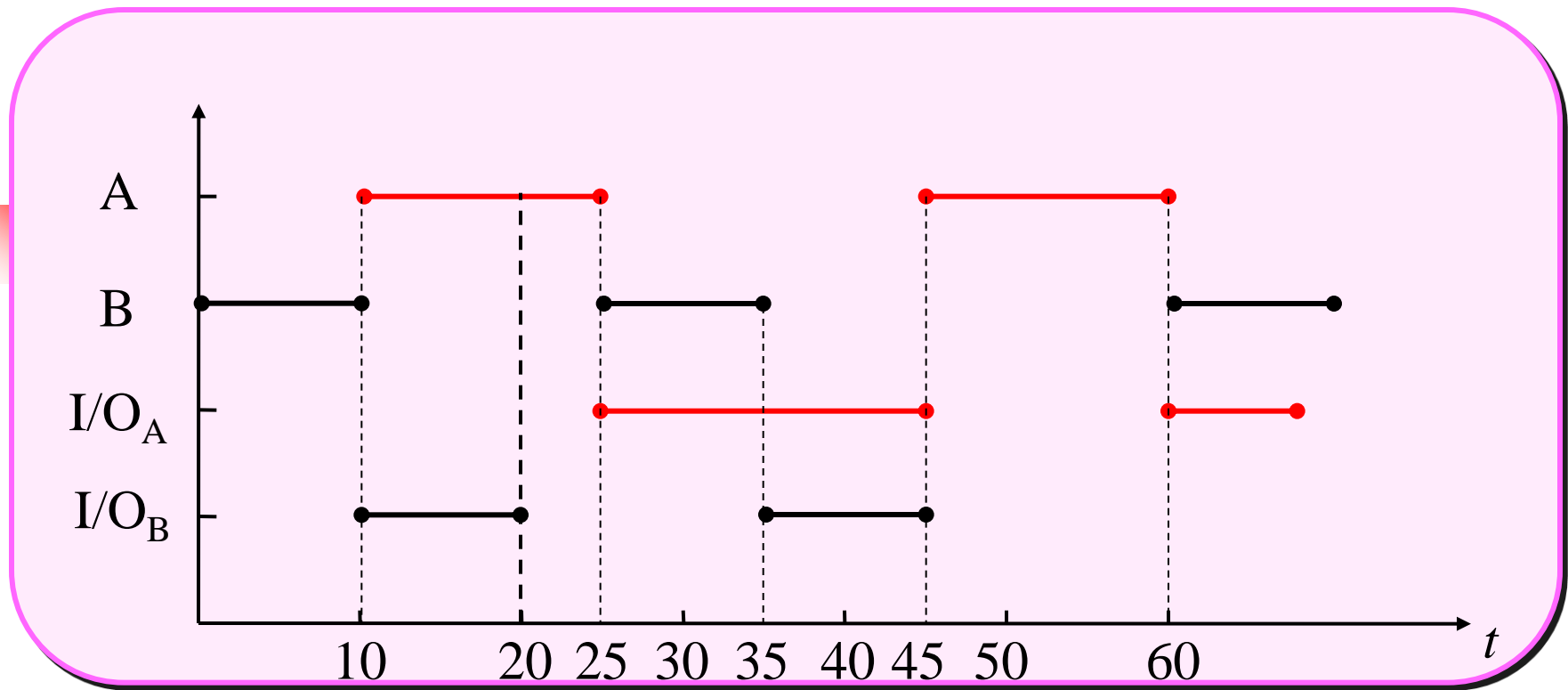


OS Features Needed for Multiprogramming

- **Memory management** – the system must allocate the memory to several jobs.
- **CPU scheduling** – the system must choose among several ready jobs to run.
- **Job scheduling** – the system must choose among ready jobs to be brought into memory.
- **Allocation of devices and I/O routine** supplied by the system.

例子：有两道程序A、B，按下图以多道程序方式运行，要求在右图画出它们的运行轨迹，并计算在60min内CPU的利用率，假设起始时首先运行B， I/O_A 和 I/O_B 不相同，并允许忽略监督程序切换A、B的时间。





$$P_{\text{CPU}} = \frac{60 - (45 - 35)}{60} \times 100\% = \frac{50}{60} \times 100\% = 83.3\%$$

若在单道程序系统中， **CPU**的利用率是多少？



Advantages and disadvantages of multiprogramming

提高CPU的利用率，充分发挥并行性，
这包括：程序之间、设备之间、设备与CPU之
间均并行工作。

用户独占资源，无交互性，延迟大



Time-Sharing Systems– Interactive Computing (1/2)

- Time-sharing operating system allows many users to share the computer simultaneously
 - Switches rapidly from one user to the next
 - Gives users the impression: **the entire computer system is dedicated for his use.**



Time-Sharing Systems– Interactive Computing (2/2)

- The CPU is multiplexed among several jobs that are kept in memory and on disk
- Sophisticated CPU-scheduling schemes are required for concurrent execution
- To obtain a reasonable response time, a job needs to be swapped in and out of memory to the disk (**virtual memory**).
- Disk management must be provided
- Job synchronization mechanisms are needed
- It may ensure that jobs do not get stuck in a deadlock



作业1

从技术发展的角度，简要介绍批处理系统、多道程序系统和分时系统各自的技术特性。（在该技术出现以前系统存在哪些问题，该技术是如何解决这些问题的）



Desktop Systems(1/2)

- **Personal Computers – computer system dedicated to a single user.**
- **I/O devices – keyboards, mouse, display screens, small printers.**
- **User convenience and responsiveness.**



Desktop Systems(2/2)

- Can adopt technology developed for larger

Single-Processor Systems:

there is one main CPU capable of executing a general-purpose instruction set.



Parallel Systems(1/2)

- **Multiprocessor systems** with more than one CPU in close communication.
- ***Tightly coupled system*** – processors share the computer bus, memory and a clock; communication usually takes place through the shared memory.



Parallel Systems (2/2)

- **Advantages of parallel system**
 - **Increased throughput:** Gets more jobs done
 - **Economy of scale:** Because of resource sharing, multiprocessor systems is cheaper than multiple single processor systems
 - **Increased reliability:** the failure of one processor will not halt the whole system



Types of Parallel Systems

- **Asymmetric multiprocessing**
 - Each processor is assigned a specific task.
 - A master processor controls the system. The other processors either look to the master for instructions or have predefined tasks.
 - Thus, this defines a **master-slave** relationship
 - Master processor schedules and allocated work to slave processors.
 - More common in extremely large systems



Types of Parallel Systems

- **Symmetric multiprocessing (SMP)**
 - Each processor runs an identical copy of the operating system.
 - Many processes can run at once without performance deterioration.
 - Must carefully control I/O to ensure that the data reach the appropriate processor
 - May result in inefficiencies
 - Most modern operating systems support SMP



Distributed Systems (1/3)

- **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.



Distributed Systems (2/3)

- A distributed system is a collection of physically **separate**, possible **heterogeneous** computer systems that are **networked** to provide the users with access to the various resources that the system maintains
- Advantages of distributed systems.
 - Resources Sharing
 - Computation speed up – load sharing
 - Reliability
 - **Communications**



Distributed Systems (3/3)

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



Clustered Systems (1/2)

- A clustered system has two or more individual systems **shared storage** and **closely linked** via LAN networking.
- The key of clustered system is **high availability**.



Clustered Systems (2/2)

- **Asymmetric Clustering**

- One machine is in **hot-standby** mode while others are running applications.
- The hot-standby machine (*i.e.*, does nothing but) monitors other machines and becomes active if one server fails.

- **Symmetric Clustering**

- Two or more hosts are running applications and monitor each other.
- This is more efficient as it uses all available hardware



Real-Time Systems (1/3)

- **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.**



Real-Time Systems (2/3)

- A real-time operating system has well-defined, **fixed time constraints**.
- Processing ***must*** be done within the defined constraints, or the system will fail.
- **Embedded systems** almost always run real-time operating systems



Real-Time Systems (3/3)

- Real-Time systems may be either *hard* or *soft* real-time.
 - *Hard Real-Time Systems* guarantee that critical tasks be completed on time.
 - *Soft Real-Time Systems* prioritize critical tasks. That is, a critical task get priority over other tasks, and retains that priority until it completes.



Handheld Systems

- **Personal Digital Assistants (PDAs)**
- **Pocket-PC**
- **Cellular telephones**
- **Advantages**
 - **Convenience and Portability**
- **Issues**
 - **Limited memory**
 - **Slow processors**
 - **Small display screens**