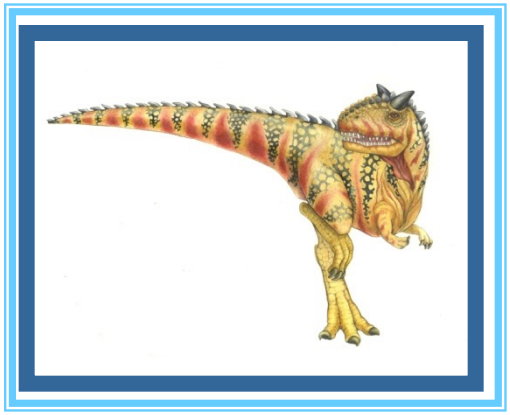
Chapter 13: I/O Systems



Chapter 13: I/O Systems

- I/O Hardware
- Software I/O Approaches
- Principles of I/O Software
- Application I/O Interface
- Transforming I/O Requests to Hardware Operations

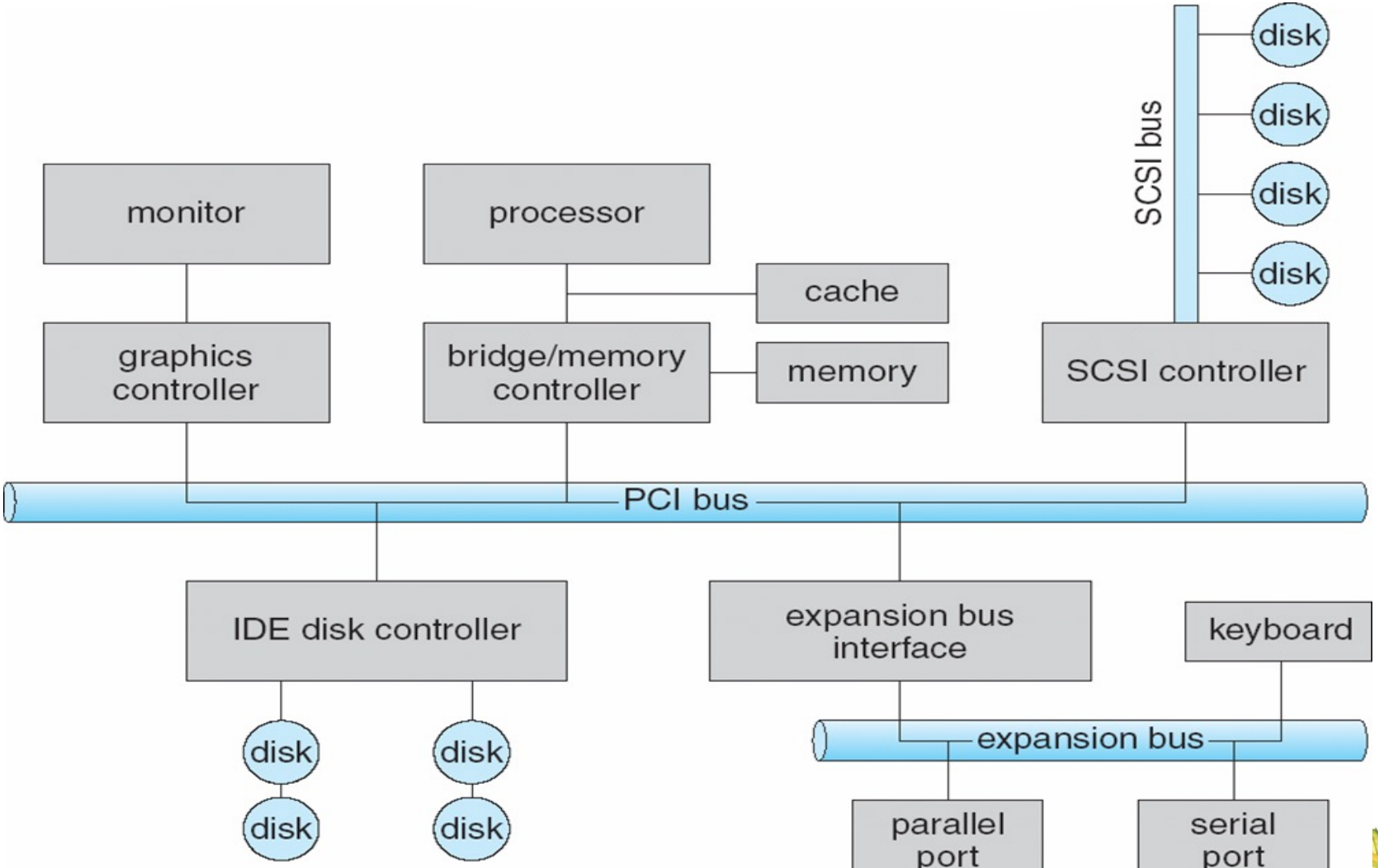
Objectives

- Explore the structure of an operating system's I/O subsystem
- Discuss the principles of I/O hardware and its complexity
- Provide details of the performance aspects of I/O hardware and software

I/O Hardware

- Incredible variety of I/O devices
- Common concepts
  - **Port**
  - **Bus**
  - **Controller** (**host adapter**)
- I/O instructions control devices
- Devices have addresses, used by
  - Direct I/O instructions
  - **Memory-mapped** I/O

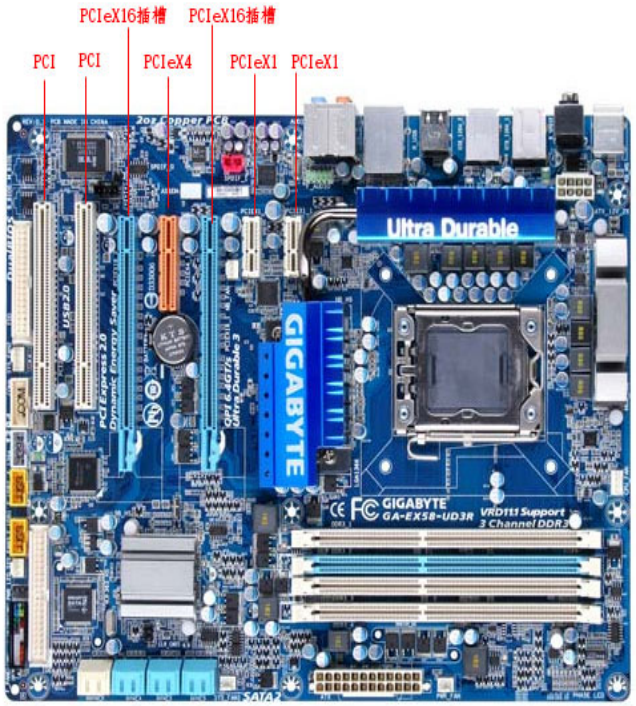
A Typical PC Bus Structure



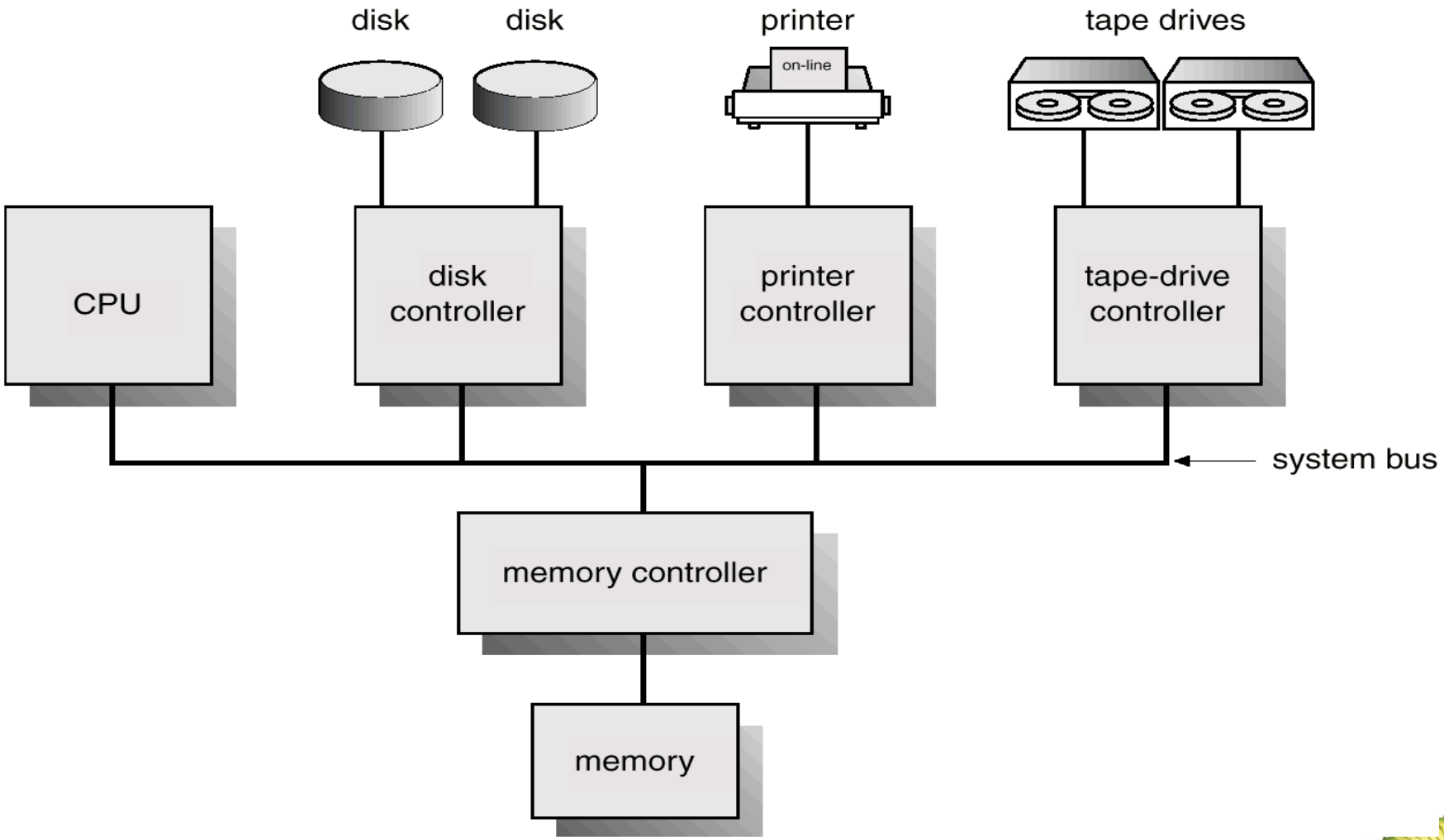
A Typical PC Bus Structure

PCI(Peripheral Component Interconnect)是英特尔推出的局部总线。地址与数据分时复用，支持即插即用 、中断共享等功能。**PCI Express**是新一代的总线接口，称之为第三代**I/O**总线技术。接口根据总线位宽不同而有所差异，包括**X1**、**X4**、**X8**以及**X16**，能够提供**8GB/s**的带宽，

PCI插槽



Device Controller



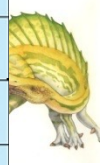
Device Controllers

- Controller's tasks
  - Control the physical operation of the device
  - convert serial bit stream to block of bytes
  - perform error correction as necessary
  - Exchange data with CPU via registers

◆可编址设备，连接多个设备时，每一个地址对应一个设备

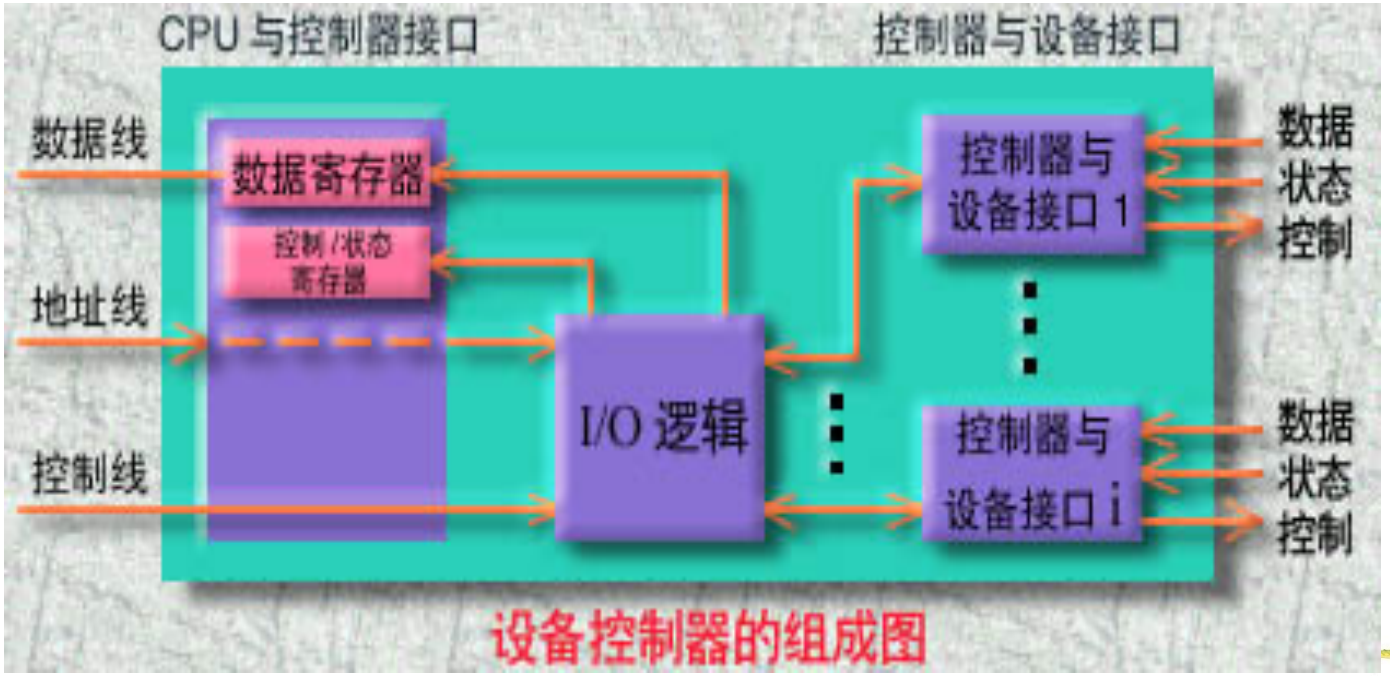
Device I/O Port Locations on PCs

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)



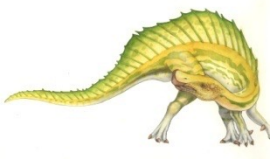
Device Controllers

- By writing into the registers: OS can command it to deliver or accept data, Or switch the device on or off
- By reading from the registers: OS can learn the status of the device
- In addition, controllers often have buffers
  - Which the OS can read or write, Eg: a video controller may have Video RAM,A data buffer programs can write or read



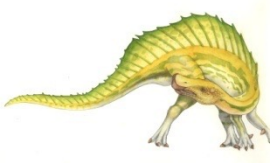
Software I/O Approaches

- Programmed I/O(Polling)
- Interrupt-driven I/O
- DMA I/O
- Channel I/O



Programmed I/O

- CPU wait for the I/O to complete
- This is called Polling or Busy waiting
  - simple to program



Programmed I/O(Polling)

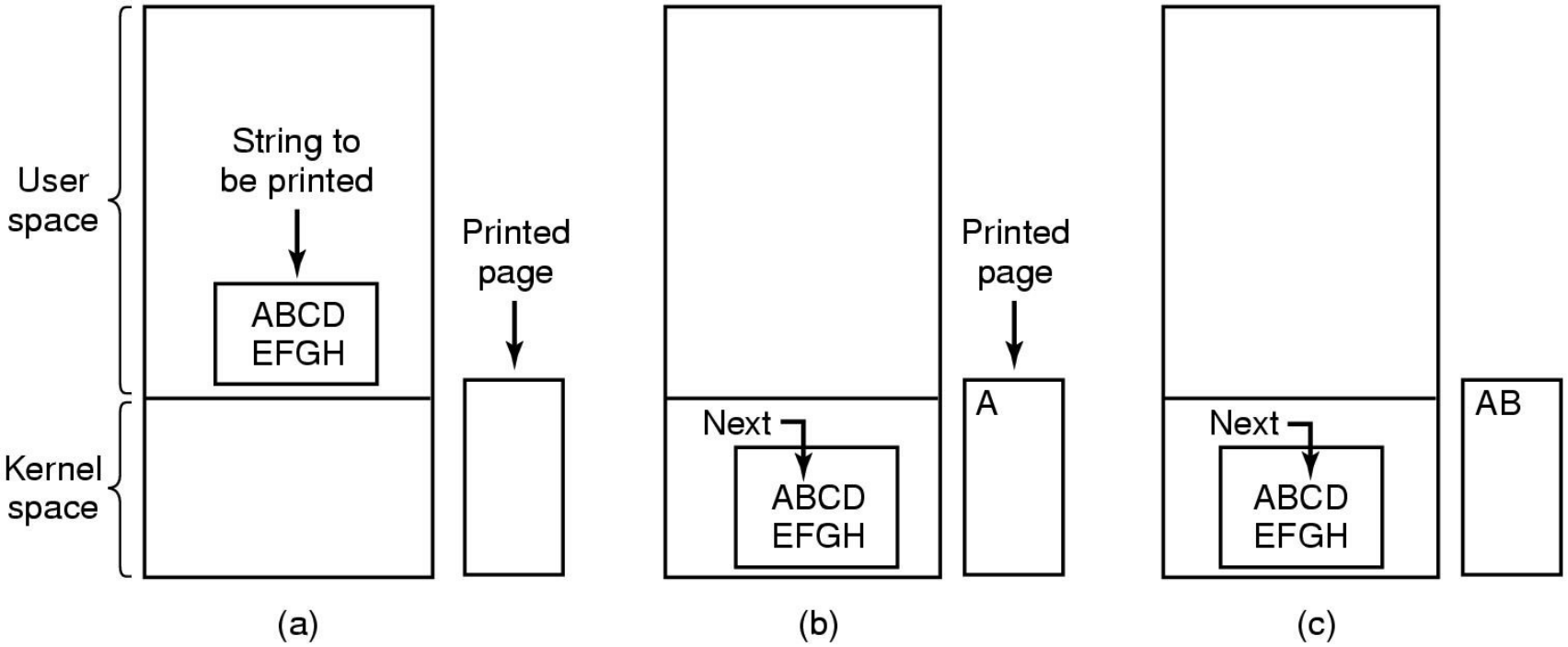
- The host repeatedly reads the *busy bit* until the bit becomes clear;
- The host issues an I/O instruction to controller;
- The controller sets the *busy bit;*
- The controller does the I/O operation by controlling the device;
- Once the I/O operation is done successfully, the controller clears the *busy bit* to inform the host;
- **Busy-wait** cycle to wait for I/O from device;

实现查询方式输入数据的程序如下:

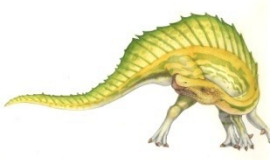
```
TEST: IN AL, STATUS;读入状态
      AND AL, 01H;测试是否准备好
      JE TEST;未好，继续测试
      IN AL, n;从端口n读入数据
      MOV BUFFER, AL;存数据到缓冲区
```

How about output?

Programmed I/O



Steps in printing a string

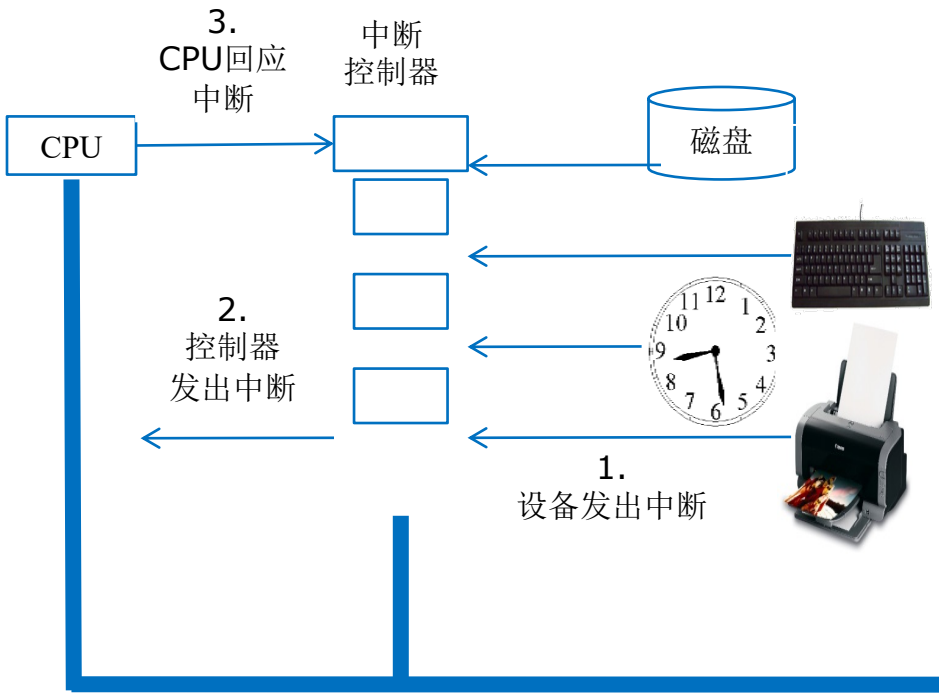


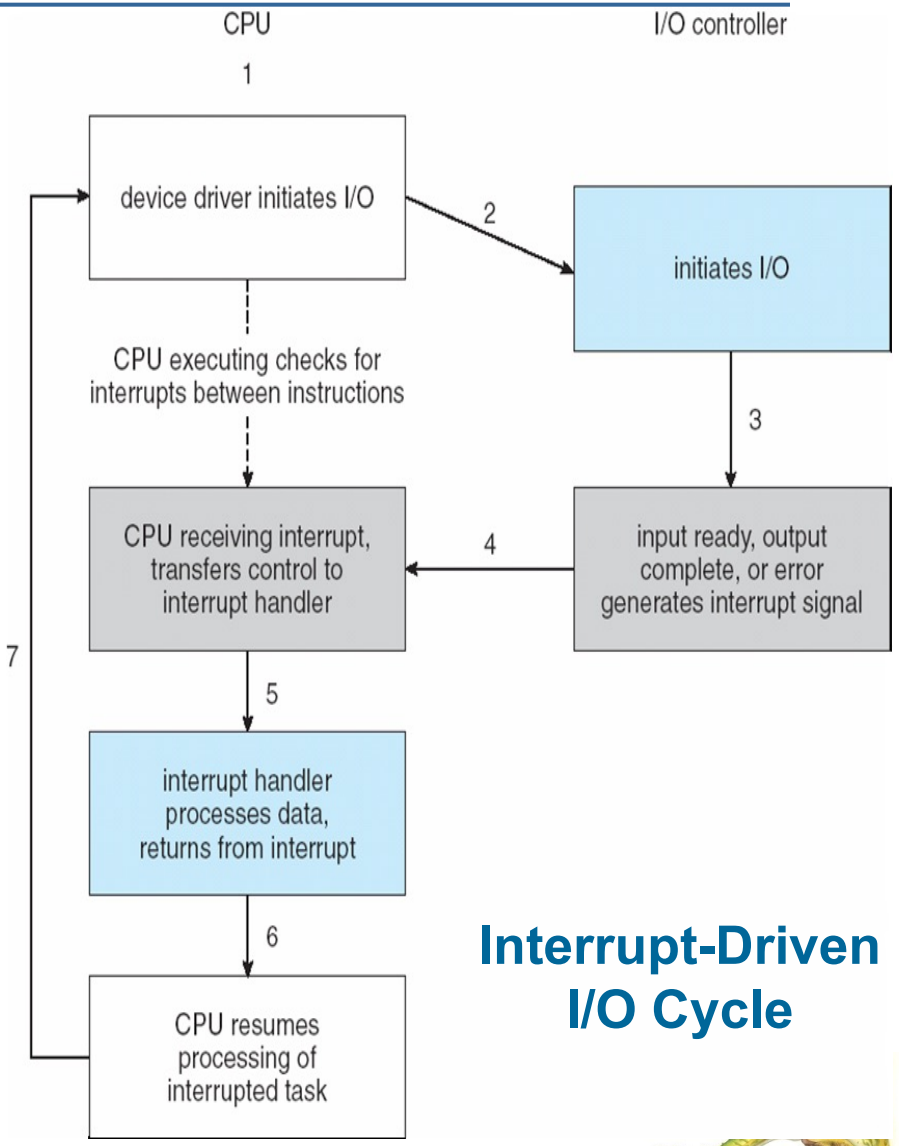
Interrupts

- **Interrupt----the hardware mechanism that enables a device to notify the CPU**
- The basic interrupt mechanism works as follows:
  - The device controller raises an interrupt by asserting a signal on the *interrupt request line*
  - CPU catches the interrupt and dispatches it to the interrupt handler
  - Interrupt handler performs necessary processing and executes a *return from interrupt* instruction to return CPU
- Interrupt vector to dispatch interrupt to correct handler
  - Based on priority
  - Some **nonmaskable**
- Interrupt mechanism also used for exceptions

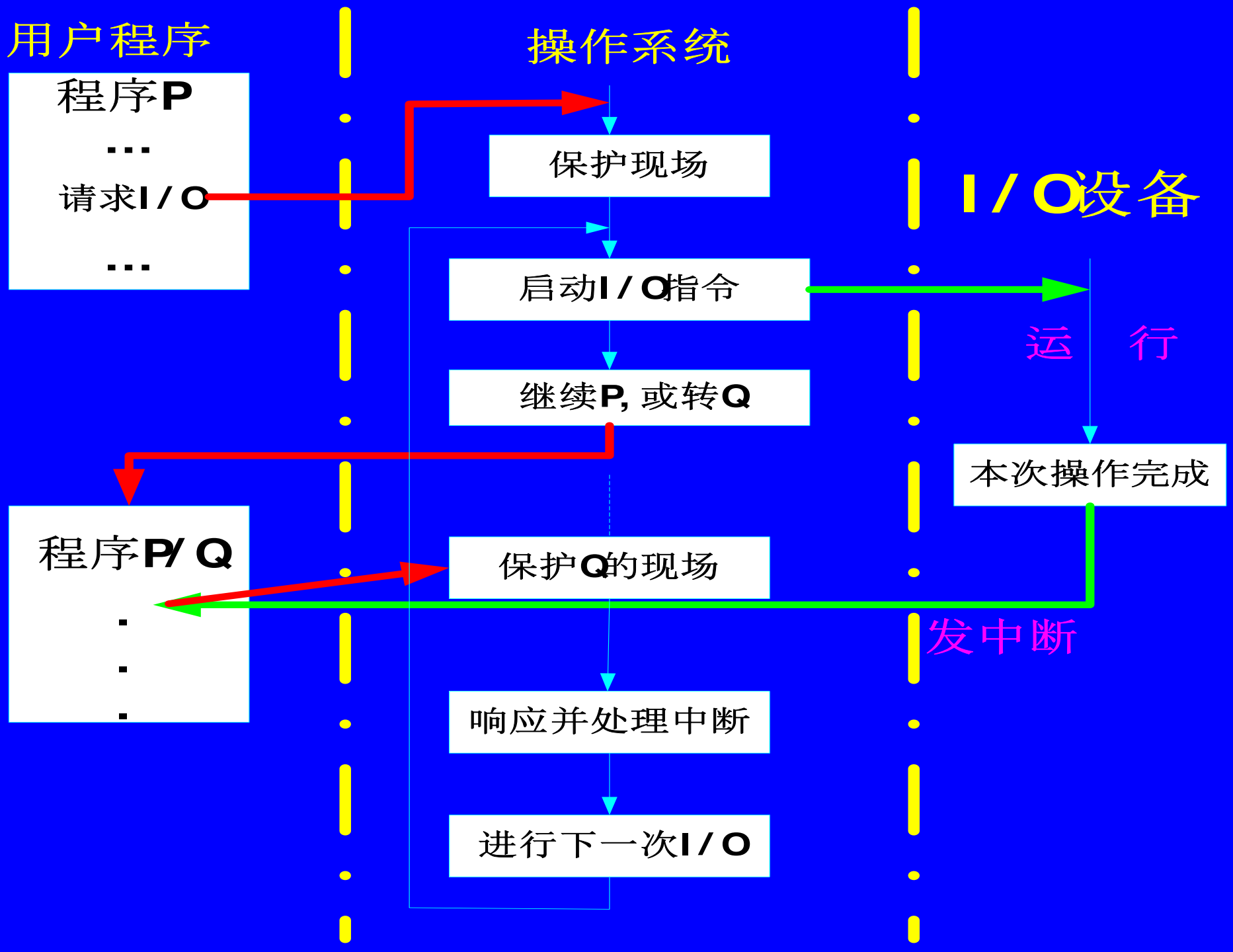
Interrupt-Driven I/O

- CPU setups and start I/O operation
- CPU goes off to do other things
- When I/O is done, CPU is interrupted
- CPU handles the interrupt
- CPU resumes interrupted operation



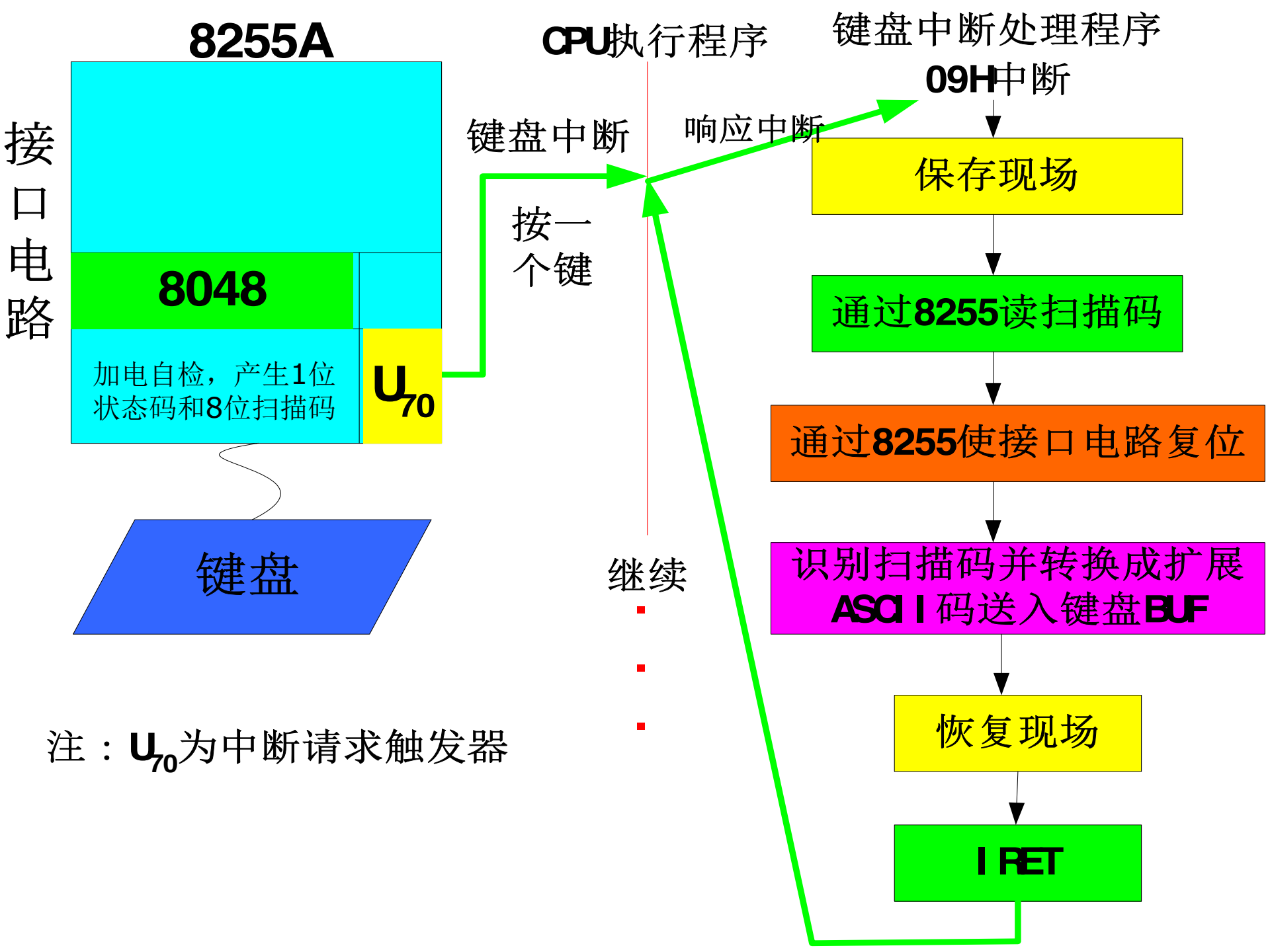


Interrupt-Driven I/O Cycle


用户程序
操作系统
程序P
…
请求I/O
…
保护现场
I/O设备
启动I/O指令
运 行
继续P, 或转Q
本次操作完成
程序P/Q
保护Q的现场
发中断
响应并处理中断
进行下一次I/O

Intel Pentium Processor Event-Vector Table

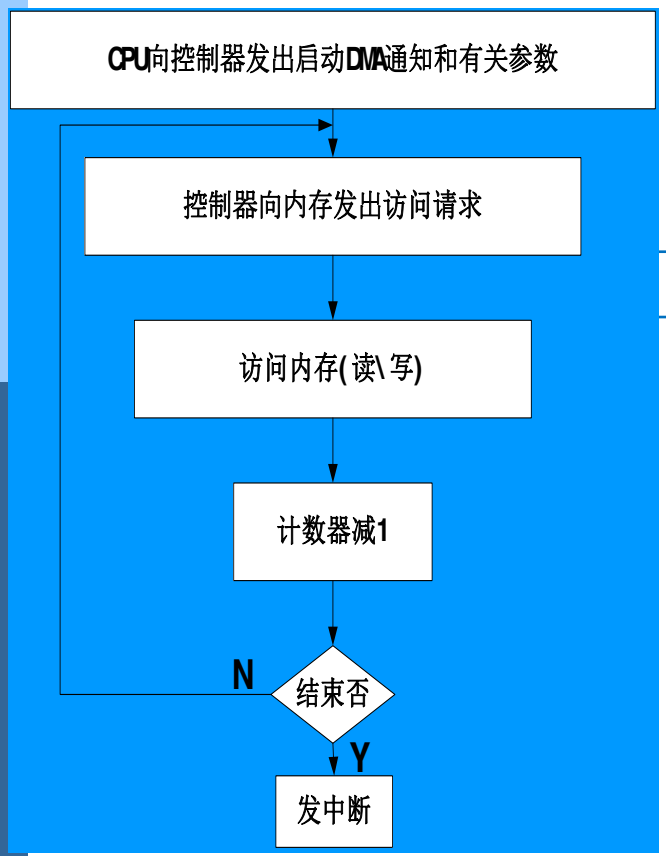
vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

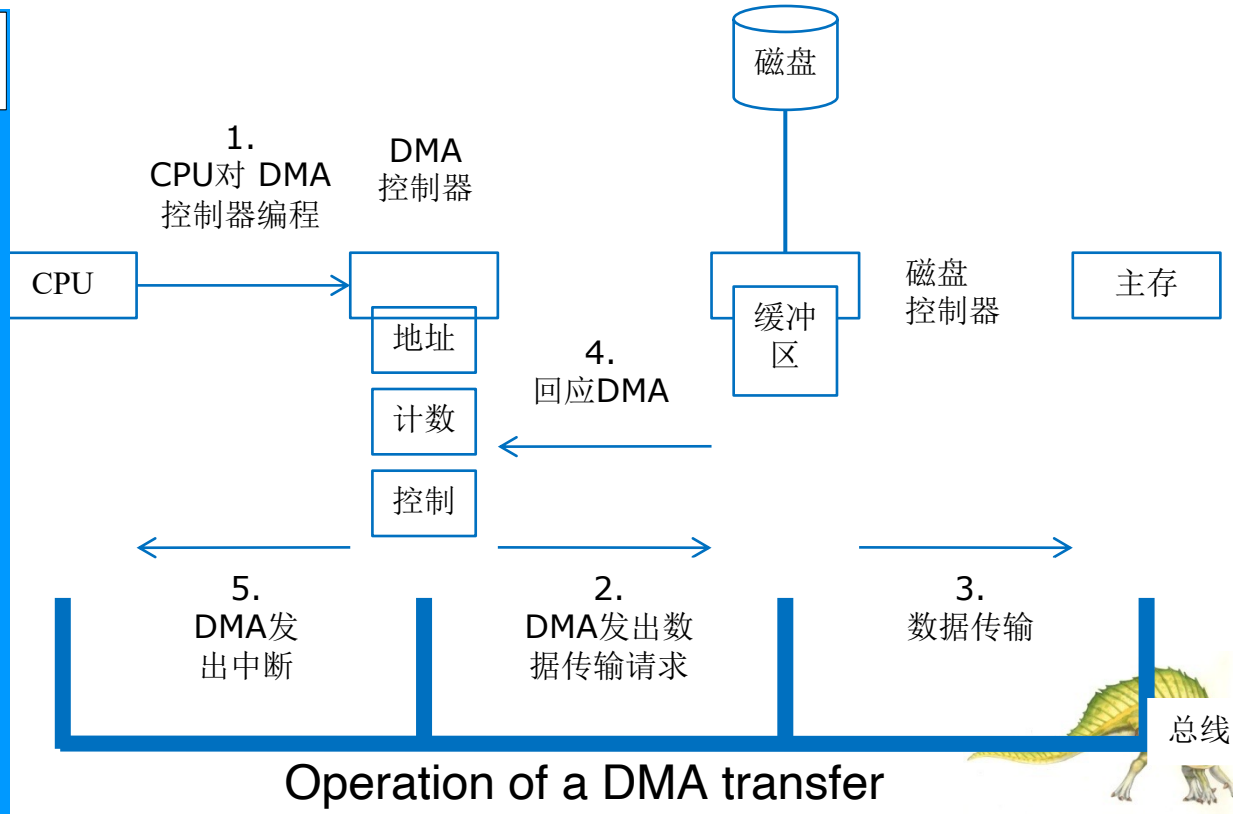


注：U_{70}为中断请求触发器

I/O Using DMA(Direct Memory Access)

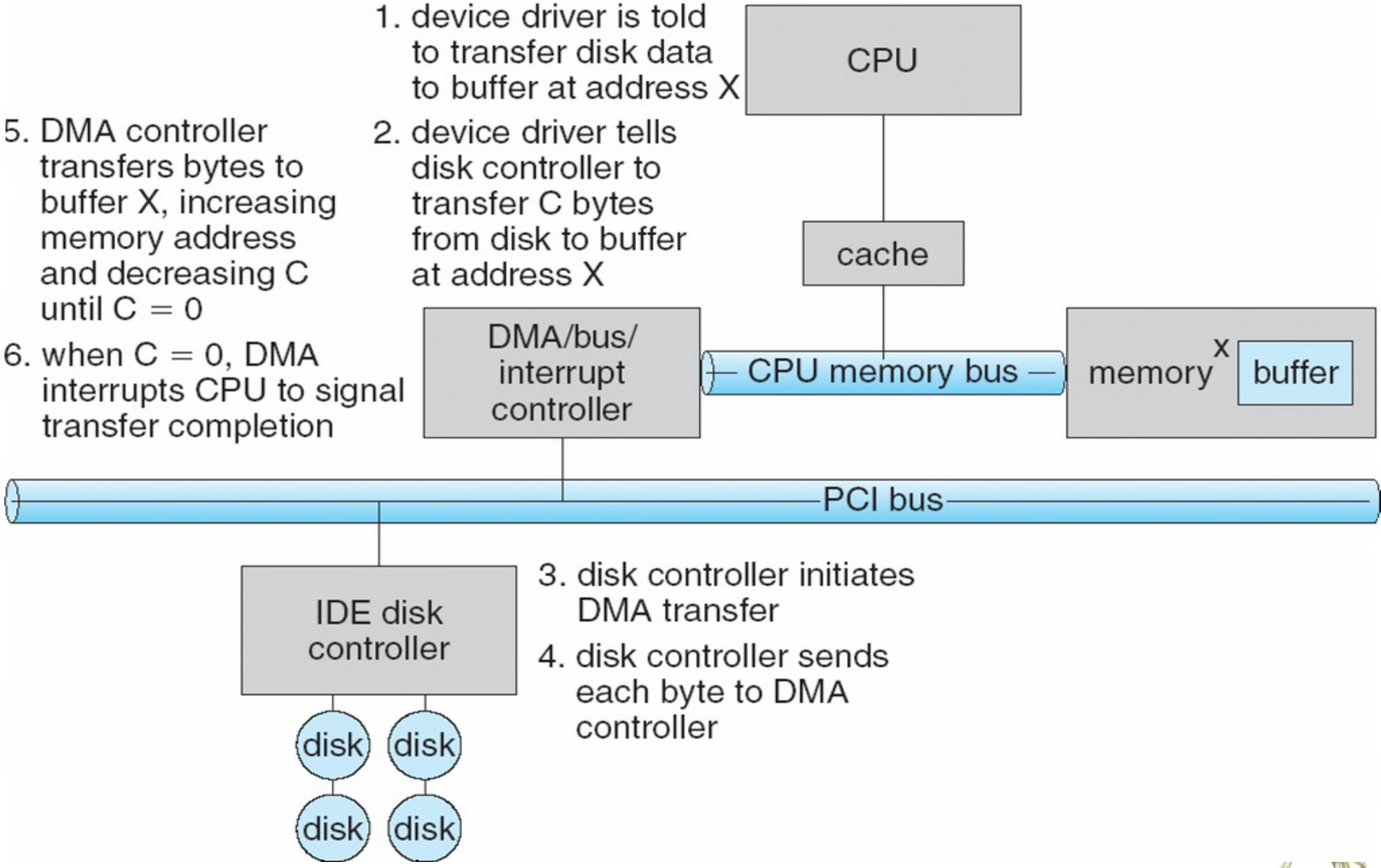
- An obvious con for interrupt-driven I/O is the frequent interrupts
- The solution is to use DMA, Idea is to let DMA controller handle the I/O one character at a time, In other words, busy wait with DMA.
- Bypasses CPU to transfer data directly between I/O device and memory

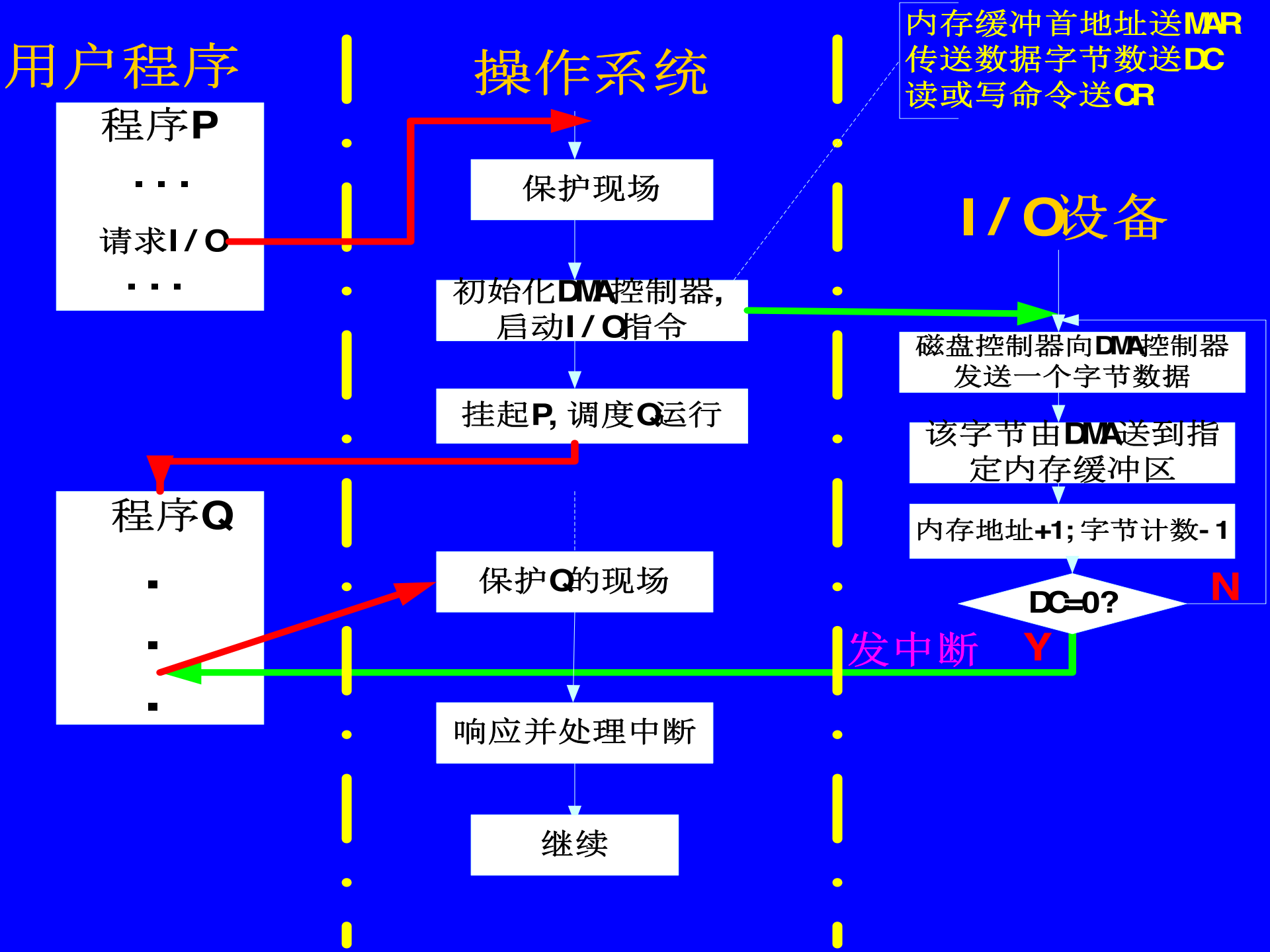




Operation of a DMA transfer

Six Step Process to Perform DMA Transfer



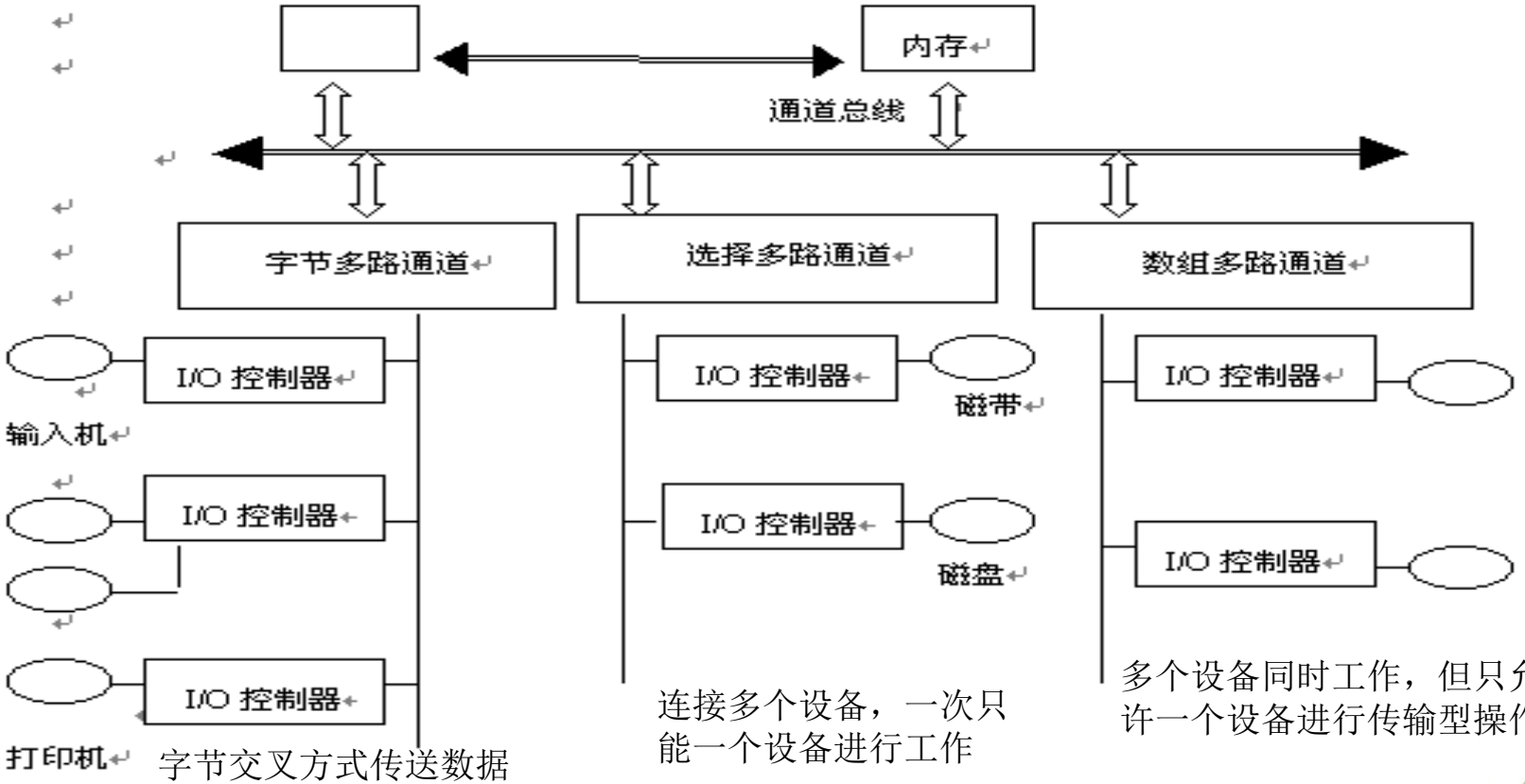

用户程序
操作系统
内存缓冲首地址送MAR
传送数据字节数送DC
读或写命令送CR
程序P
. . .
请求I/O
. . .
保护现场
初始化DMA控制器,
启动I/O指令
挂起P, 调度Q运行
I/O设备
磁盘控制器向DMA控制器
发送一个字节数据
该字节由DMA送到指
定内存缓冲区
内存地址+1; 字节计数-1
DC=0?
N
Y
发中断
程序Q
. . .
保护Q的现场
响应并处理中断
继续

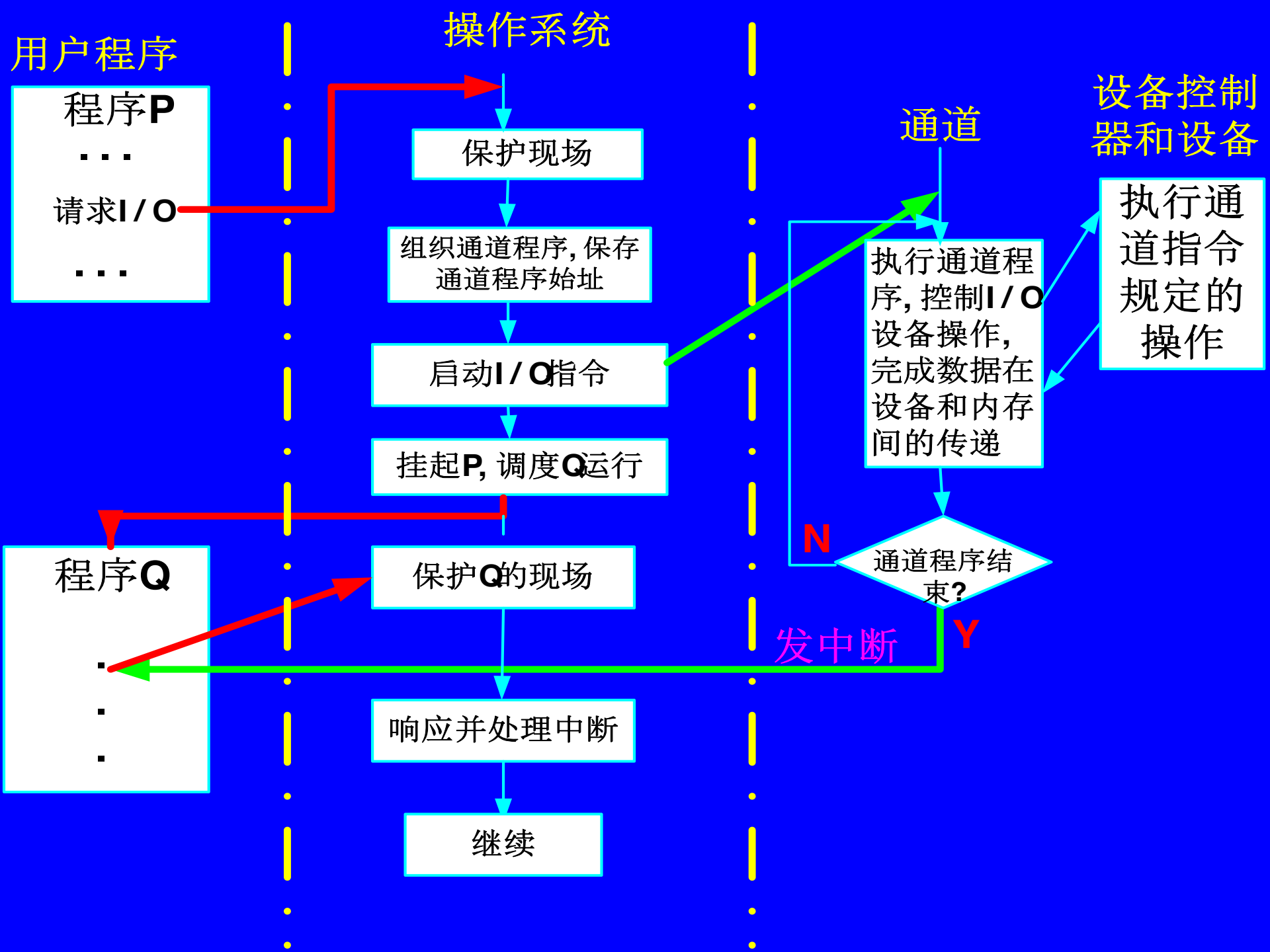
Issues with DMA

- How to access the bus?
  - Cycle stealing(利用CPU不访存的周期实现DMA操作，比如CPU在执行乘法指令时。当I/O设备发出DMA请求时，便挪用或窃取总线占用权一个或几个主存周期进行数据传输)
- Where to store the data?
  - Directly into memory or in DMA buffer?
- How to address the memory?
  - Virtual or physical address?

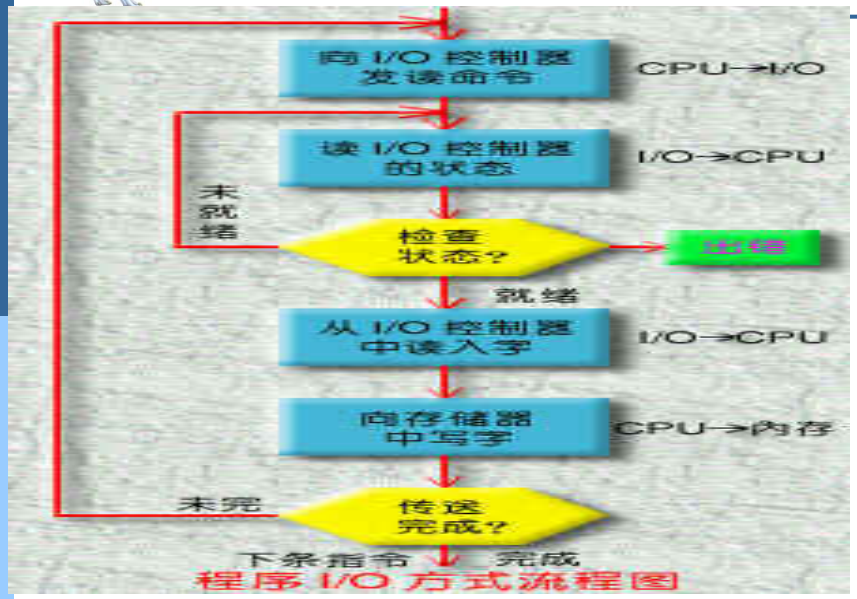
Channel I/O

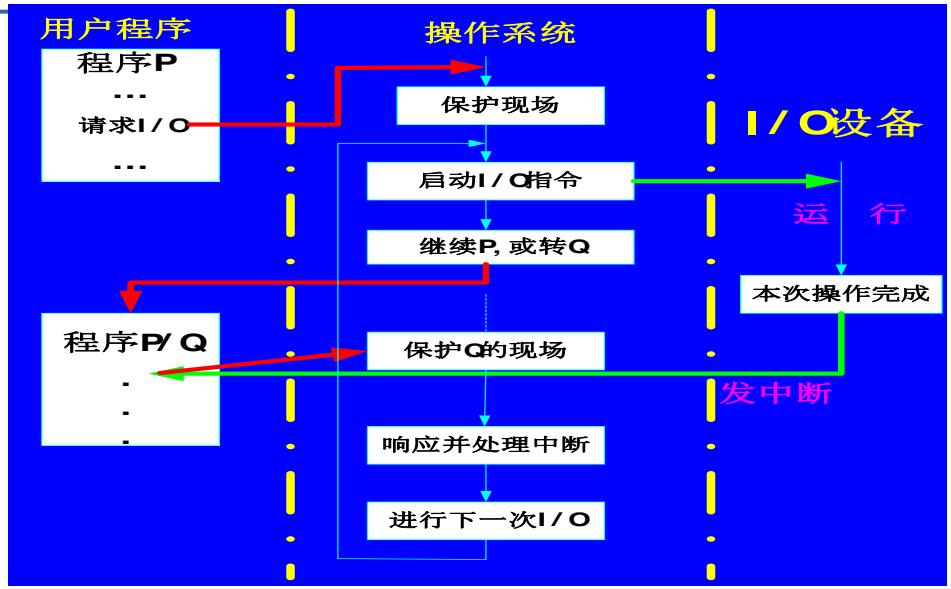
- A special processor, also called I/O processor;
- Executes channel program and completes a set of I/O operations at a time
- Transfers data between devices and memory

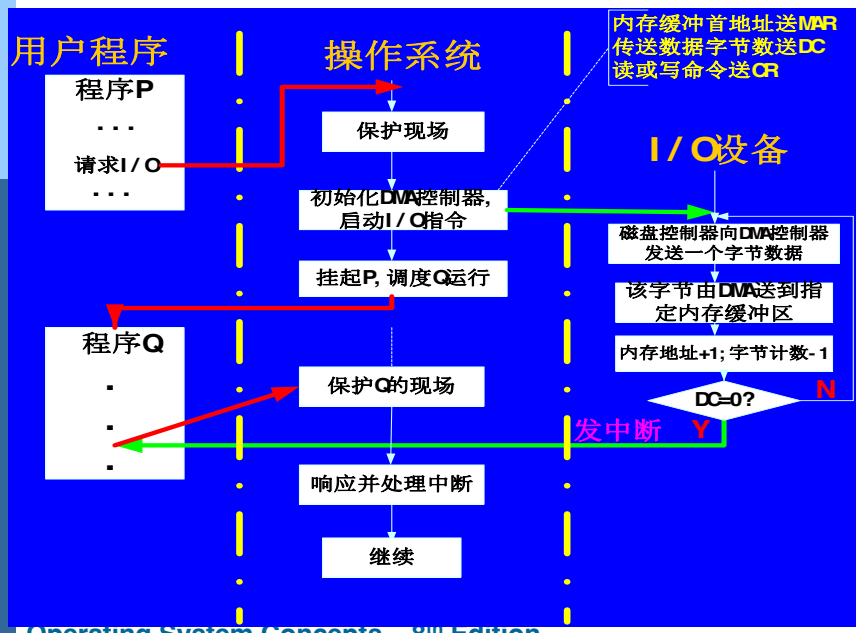


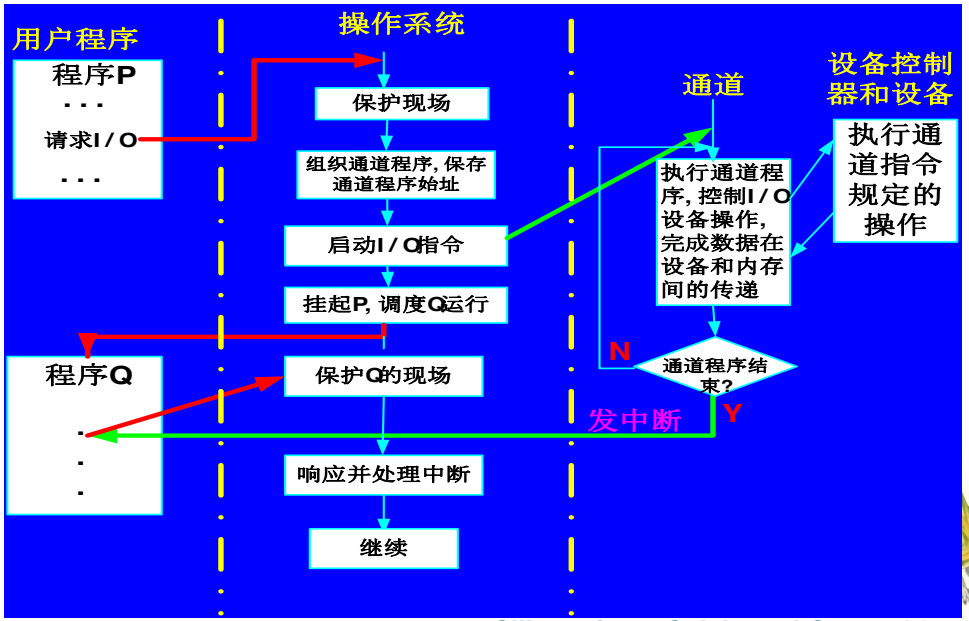

用户程序
操作系统
通道
设备控制器和设备
程序P
. . .
请求I/O
. . .
保护现场
组织通道程序，保存通道程序始址
启动I/O指令
挂起P，调度Q运行
执行通道程序，控制I/O设备操作，完成数据在设备和内存间的传递
执行通道指令规定的操作
N
通道程序结束?
Y
发中断
程序Q
. . .
保护Q的现场
响应并处理中断
继续

四种方式比较


向I/O控制器发读命令
CPU→I/O
读I/O控制器的状态
I/O→CPU
未就绪
检查状态?
出错
就绪
从I/O控制器中读入字
I/O→CPU
向存储器中写字
CPU→内存
未完
传送完成?
下条指令
完成
程序I/O方式流程图


用户程序
操作系统
I/O设备
程序P
...
请求I/O
...
保护现场
启动I/O指令
运行
继续P, 或转Q
本次操作完成
程序P/Q
保护Q的现场
发中断
响应并处理中断
进行下一次I/O

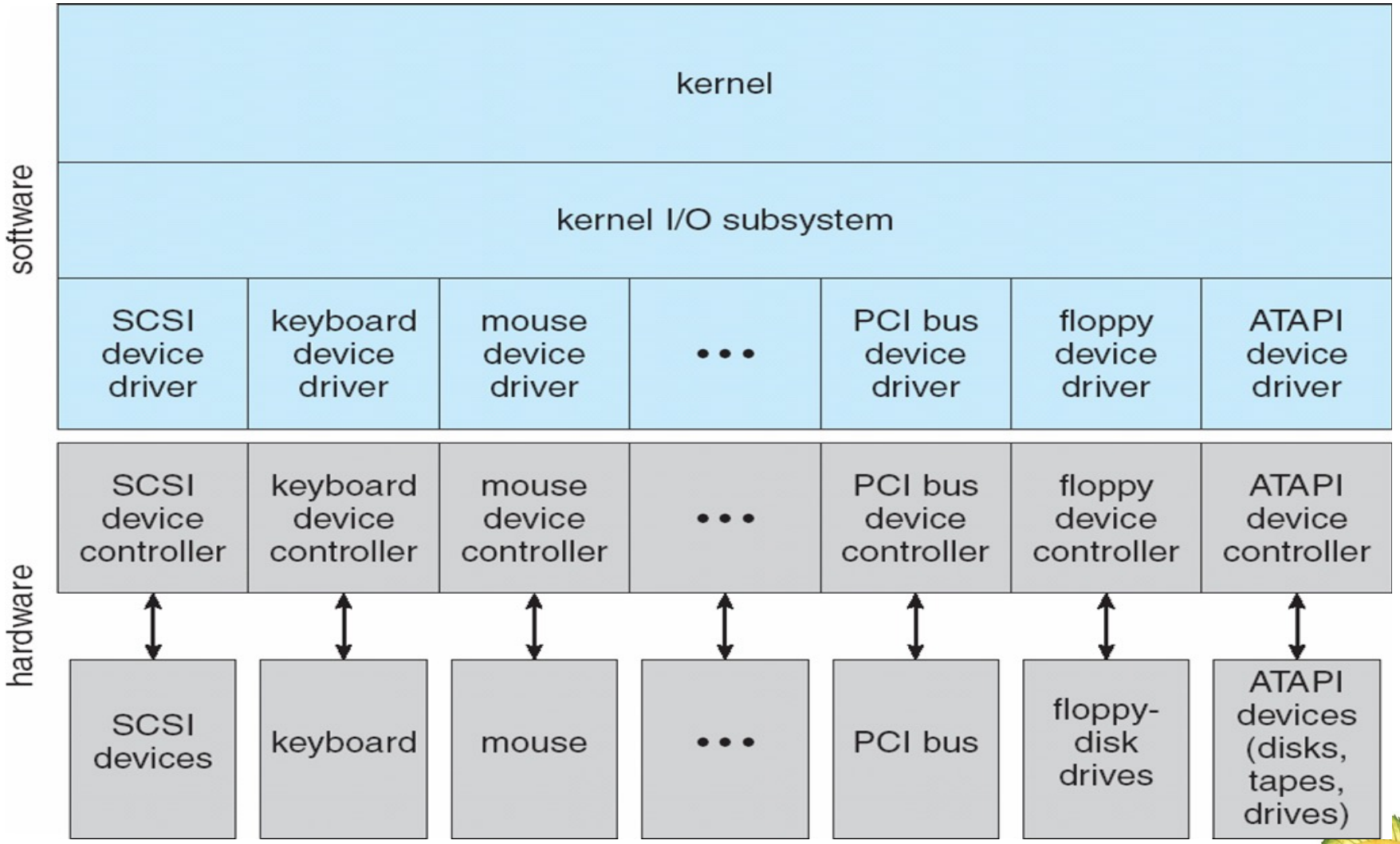

用户程序
操作系统
内存缓冲首地址送MAR
传送数据字节数送DC
读或写命令送CR
I/O设备
程序P
...
请求I/O
...
保护现场
初始化DMA控制器, 启动I/O指令
磁盘控制器向DMA控制器发送一个字节数据
该字节由DMA送到指定内存缓冲区
内存地址+1; 字节计数-1
DC=0?
N
Y
发中断
挂起P, 调度Q运行
程序Q
保护Q的现场
响应并处理中断
继续

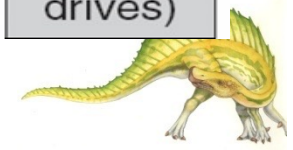

用户程序
操作系统
通道
设备控制器和设备
程序P
...
请求I/O
...
保护现场
组织通道程序, 保存通道程序始址
启动I/O指令
挂起P, 调度Q运行
执行通道程序, 控制I/O设备操作, 完成数据在设备和内存间的传递
执行通道指令规定的操作
N
通道程序结束?
Y
发中断
程序Q
保护Q的现场
响应并处理中断
继续

Application I/O Interface

- OS can abstract away the detailed differences in I/O devices by identifying a few general kinds.
- Each general kind is accessed through a standardized set of functions – an interface.
- Device-driver layer hides differences among I/O controllers from kernel

A Kernel I/O Structure





I/O device types

- **I/O设备类型繁多，从OS观点看，其性能指标有：**
  - 数据传输速率；数据的传输单位；设备共享属性等
- **按传输速率分类：**
  - 低速设备：每秒几个至数百个字节。例如：键盘、 鼠标器等设备。
  - 中速设备：每秒数千个至数万个字节。典型设备有行式打印机、激光打印机等。
  - 高速设备：传输速率在数百K个字节至数十兆字节。 磁盘机、光盘机等。
- **按信息交换的单位分类**
  - 块设备；字符设备；
- **按设备的共享属性分类**
  - 独占设备；
  - 共享设备；
  - 虚拟设备：在一类设备上模拟另一类设备，常用共享设备模拟独占设备，用高速设备模拟低速设备，被模拟的设备称为虚拟设备

Characteristics of I/O Devices

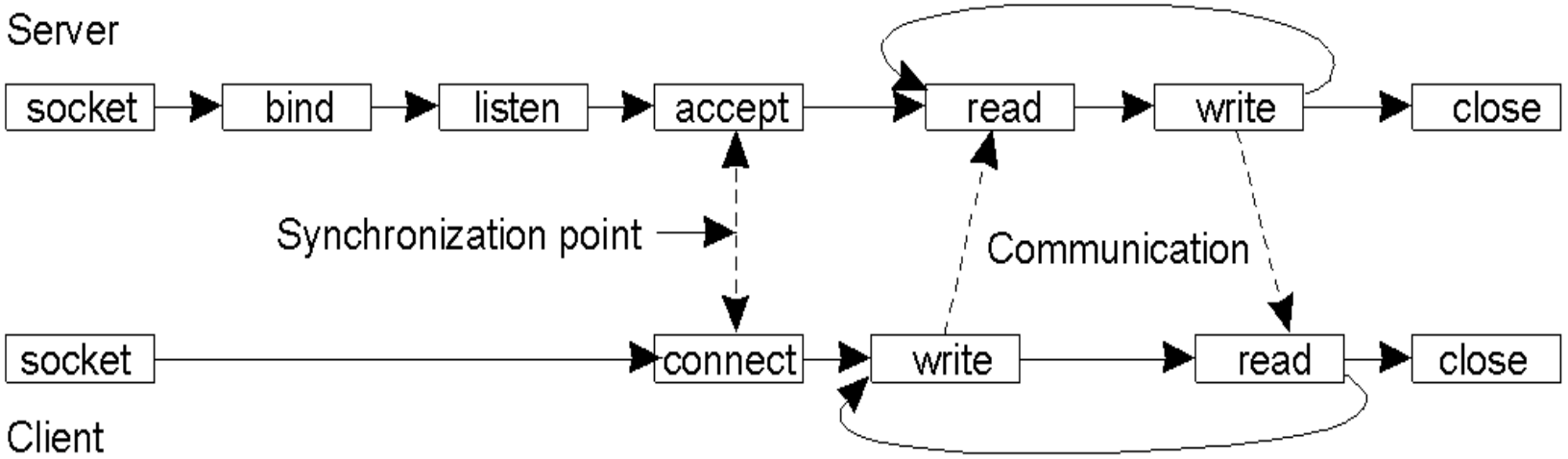
aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Block and Character Devices

- Block devices include disk drives
  - Commands include read, write, seek
  - Raw I/O or file-system access
  - Memory-mapped file access possible
- Character devices include keyboards, mice, serial ports
  - Commands include get, put
  - Libraries layered on top allow line editing(通过库行编辑)

Network Devices

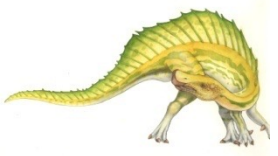
- Varying enough from block and character to have own interface
- Unix and Windows NT/9*x*/2000 include *socket* interface



- Also, many other approaches to inter-process communication and network communication
  - pipes, FIFOs, streams, message queues, mailboxes

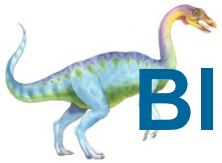
Clocks and Timers

- Hardware clocks and timers provide 3 basic functions:
  - Give the current time
  - Give the elapsed time
  - Set a timer to trigger operation X at time T
- The system calls implementing these functions are not standardized across operating systems.



Clocks and Timers

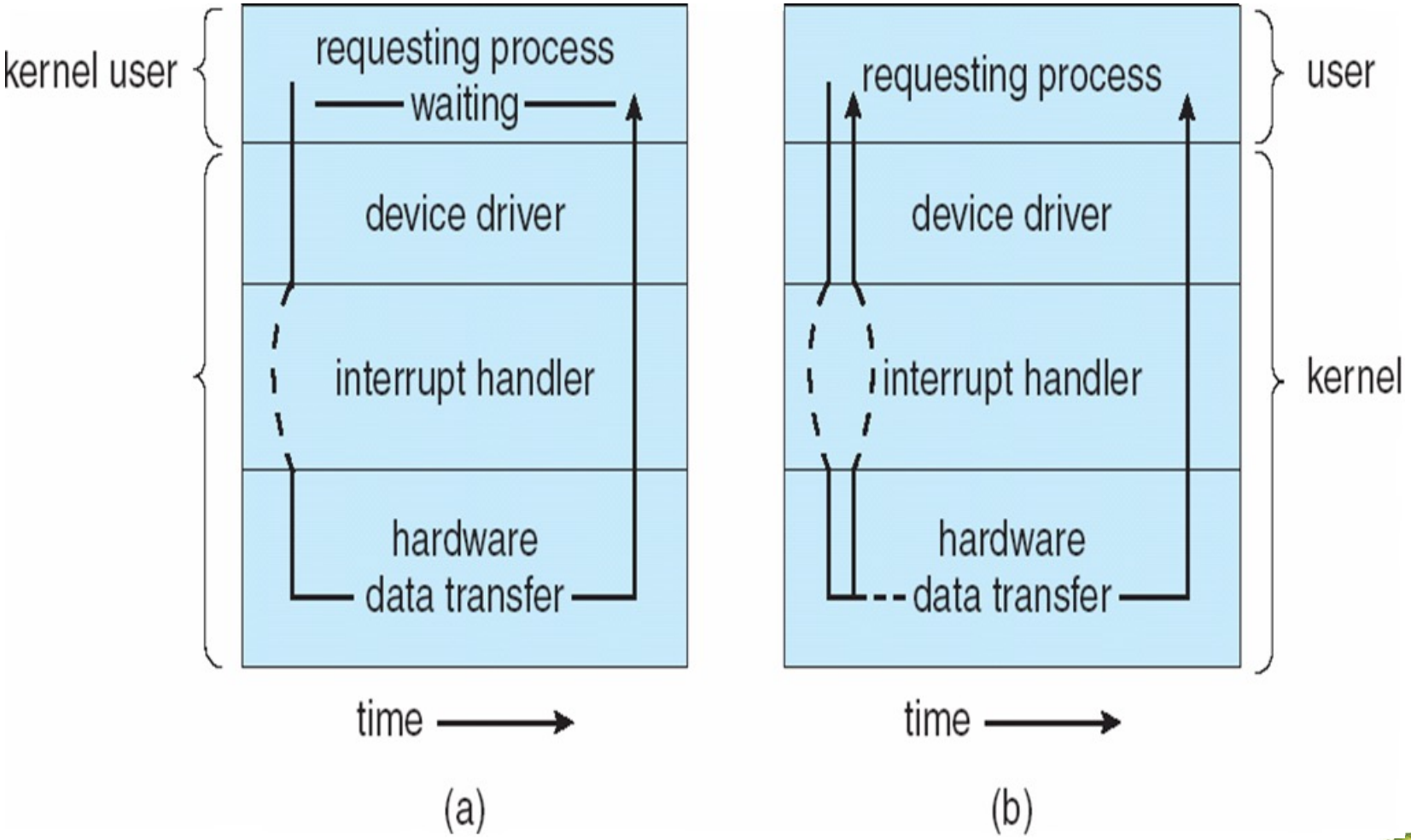
- **Programmable interval timer** used for timings, periodic interrupts. The hardware to measure elapsed time and trigger operations：
  - The scheduler uses it to generate an interrupt that will preempt a process at the end of its time slice
  - The disk subsystem uses it to invoke the flushing of dirty cache buffers to disk periodically
  - The network subsystem uses it to cancel operations that are proceeding too slowly because of network congestions or failures
- The OS may also provide an interface for user process to use timer.



Blocking and Nonblocking I/O(sys call)

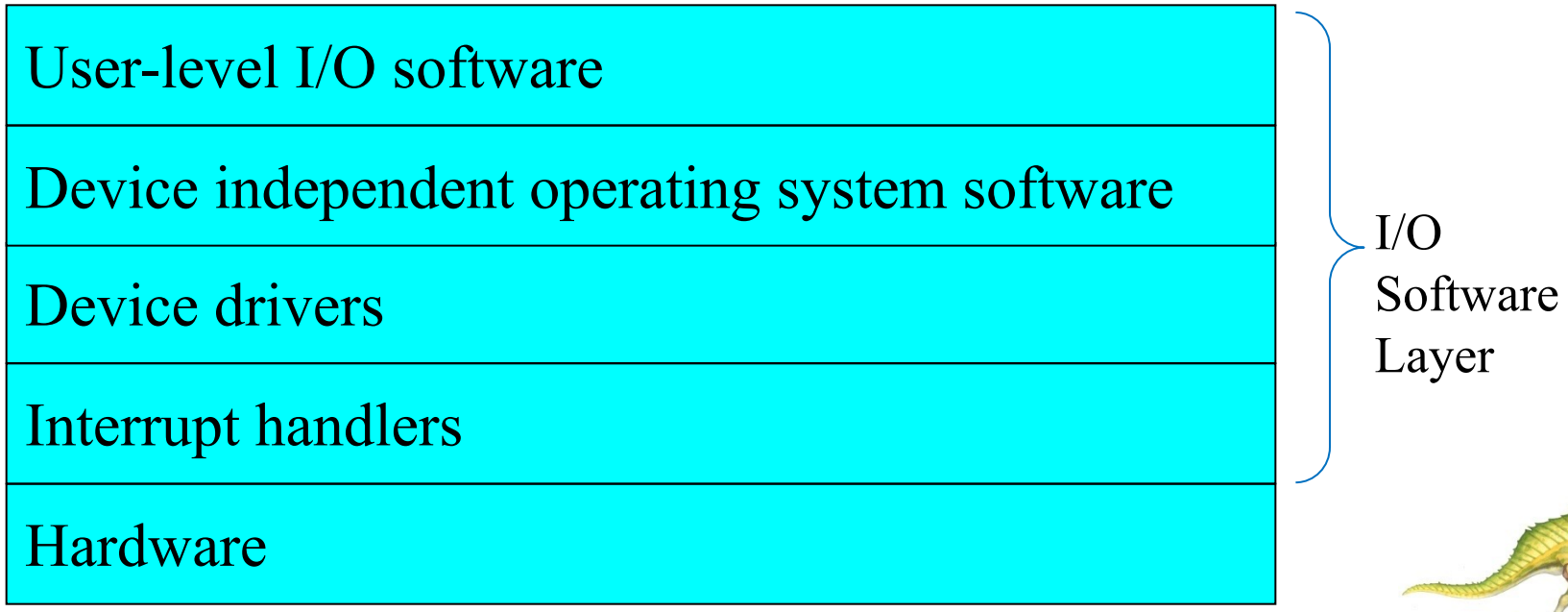
- **Blocking** - process suspended until I/O completed
  - Easy to use and understand
  - Insufficient for some needs
- **Nonblocking** - I/O call returns as much as available
  - User interface, data copy (buffered I/O)
  - Implemented via multi-threading
  - Returns quickly with count of bytes read or written
- **Asynchronous** - process runs while I/O executes(process)
  - Difficult to use
  - I/O subsystem signals process when I/O completed

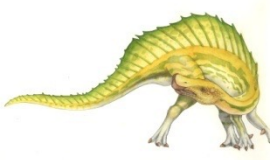
Two I/O Methods



I/O Software Layers

- IO software are organized into layers
- Each layer has a(from soft. Eng.):
  - Well-defined function to perform
  - Well-defined interface to the adjacent layer
- Exact division differs by systems





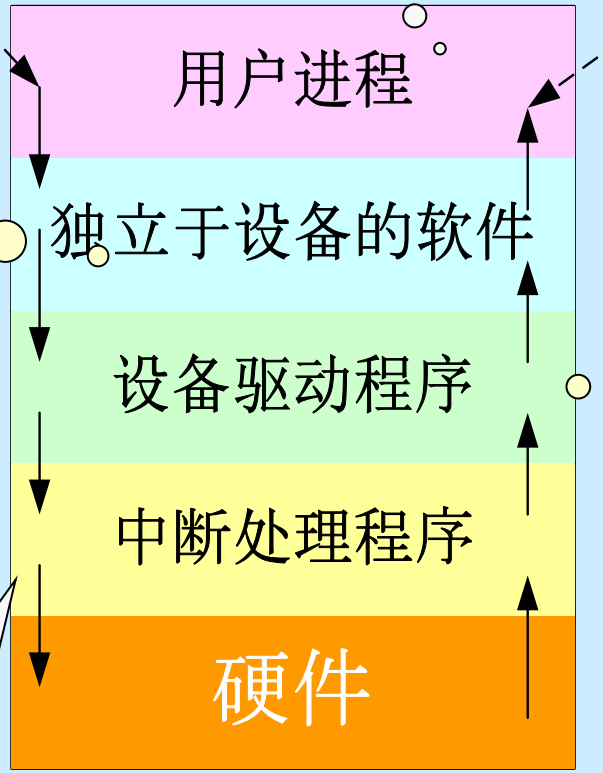
I/O系统调用，数据格式化,Spooling

设备命名、设备保护、成块处理、缓冲技术、设备分配与释放

设置设备寄存器、检查设备的执行状态

I/O请求

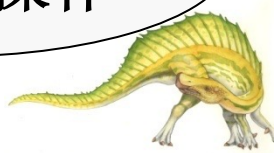
I/O完成后的回答



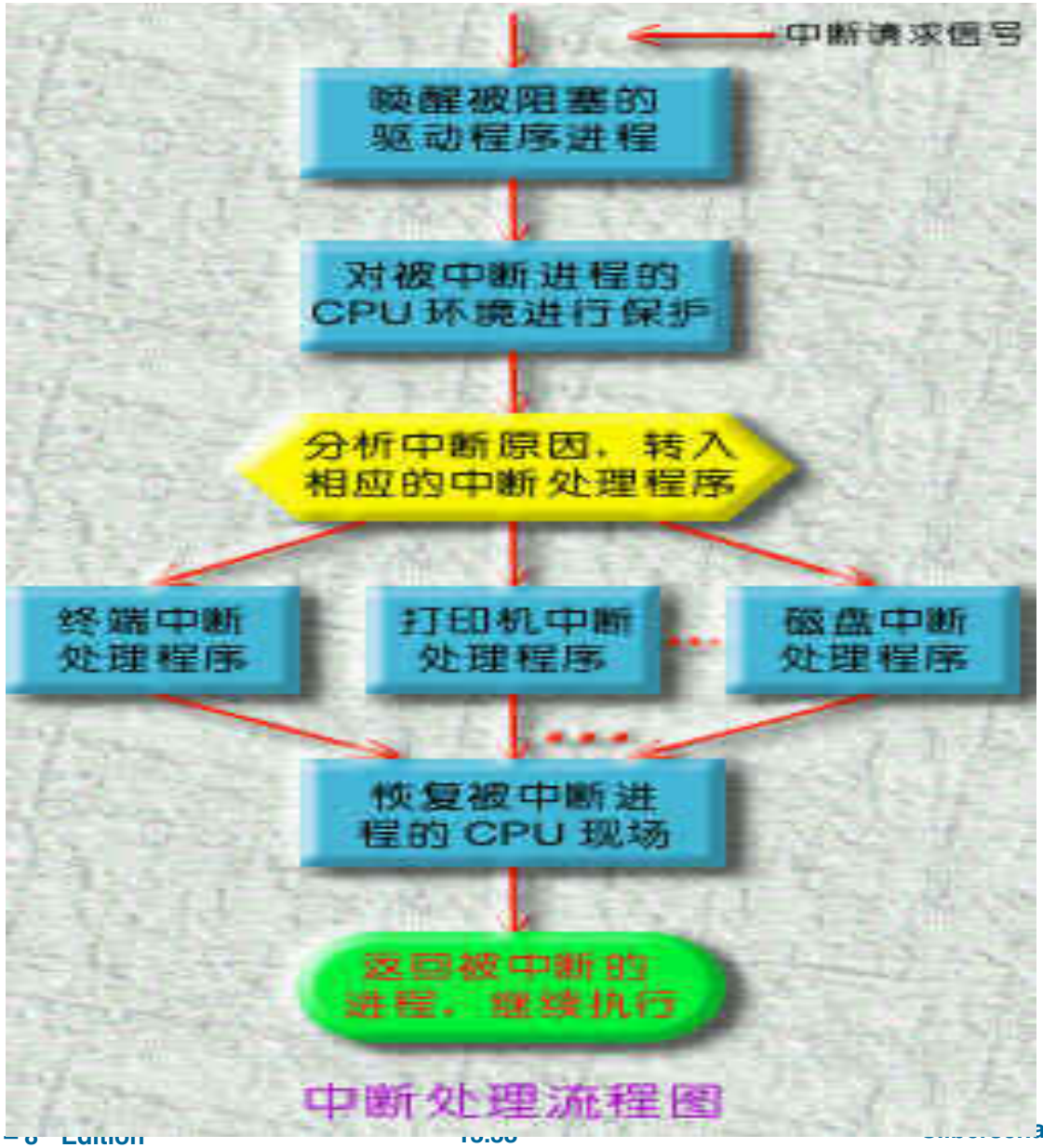
I/O软件的层次

唤醒设备驱动程序进程，进行中断处理

物理I/O操作

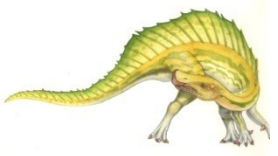


Interrupt Handlers



Device Drivers

- The software that talks to the controller
- Device specific, Tailored(定制)to individual device characteristics
- Written by device manufacturers
- Part of the OS Kernel
- Know about the details of the devices
  - Disk driver knows about sectors, tracks, cylinders, heads, arm motions, motor drives
  - Mouse driver knows about button pressed



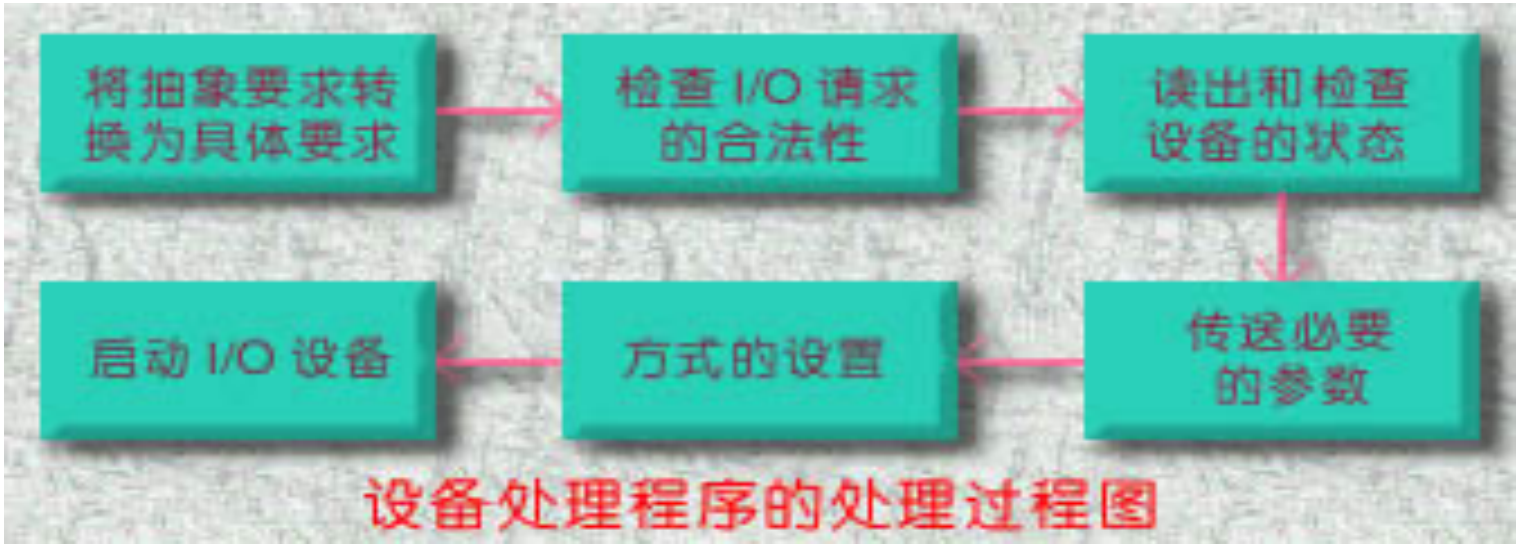
Device Drivers

- Device drivers' tasks:
  - Accept abstract read/write requests
    - from device-independent software above it
  - See that(确保)read/write requests carried out
  - Initialize device, turn on/off device
  - Power management for device
- Most OS define a standard interfaces for
  - Block devices
  - Character devices

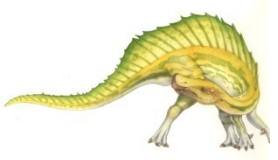
Device Drivers

- On UNIX,
  - OS is a single binary with device drivers compiled into it
  - Add new drivers or modify existing ones means recompile OS
  - But Solaris 10 has made device driver loadable afterwards
- In Windows, drivers are dynamically loaded

Device Driver Operation



- Block itself and wait for interrupt
- Upon waking up, check for I/O errors
- Pass data to the layer above it
- Return status to its caller



Device-Independent I/O Software

- Some I/O software is device specific， but some are not!
- Basic function is to:
  - Perform IO ops that are common to all devices
  - Provide uniform interface to user-level software

Device-Independent I/O Software

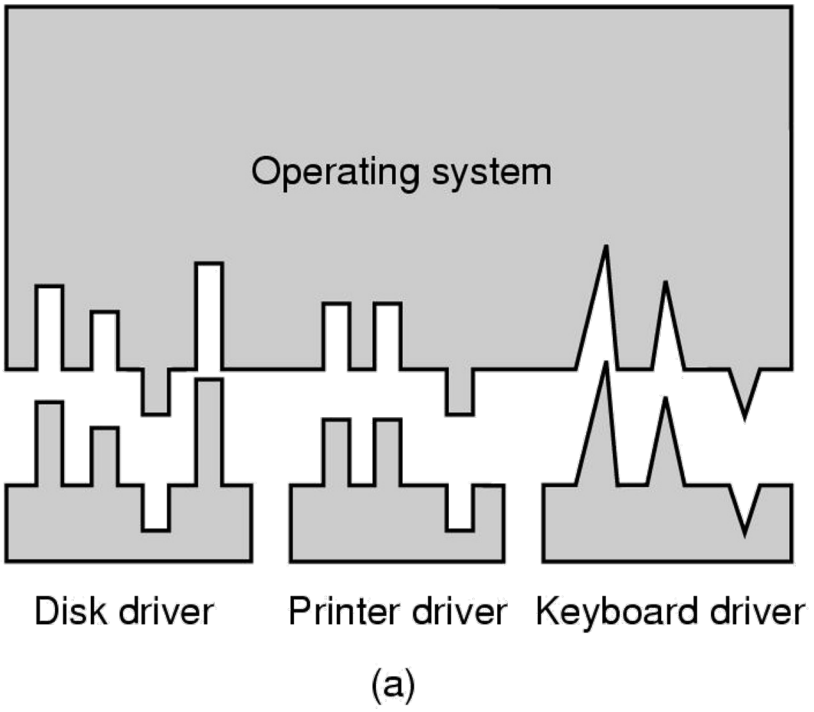
Uniform interfacing for device drivers
Buffering
Error reporting
Allocating and releasing dedicate devices
Providing a device-independent block size

Functions of the device-independent I/O software

Exact boundary between drivers and device independent software is device dependent

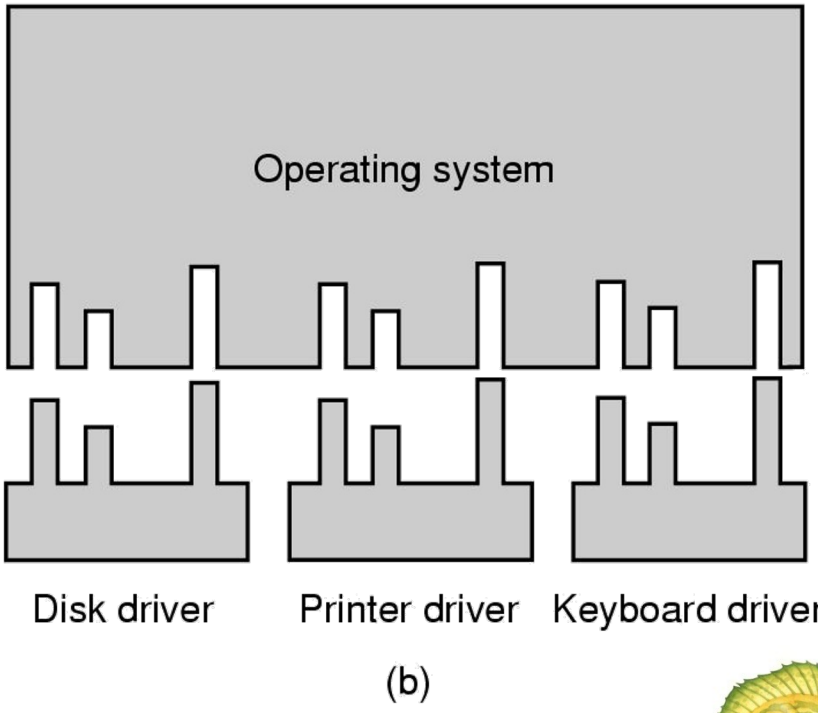
Uniform Interfacing

- Make all devices look the same or similar
  - By standardizing driver interface
- Specify the functions a driver provides and kernel function a driver can call



(a)

Without a standard driver interface

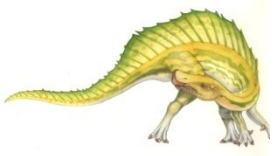


(b)

With a standard driver interface

Device Independence

- Problem:
  - lots of different brands of disks, disk interfaces,
  - And disk controllers
  - each with their own custom control interface
- Same is true of any I/O device
  - network card, video card, keyboard, mouse, etc.



Device Independence

- Add a “device driver” layer: to hide this diversity（差异）/complexity
- An abstraction makes things “better” : for things that use the abstraction

Rest of operating systems

Virtual interface ←

Device drivers

Physical interface ←

Millions of hardware devices

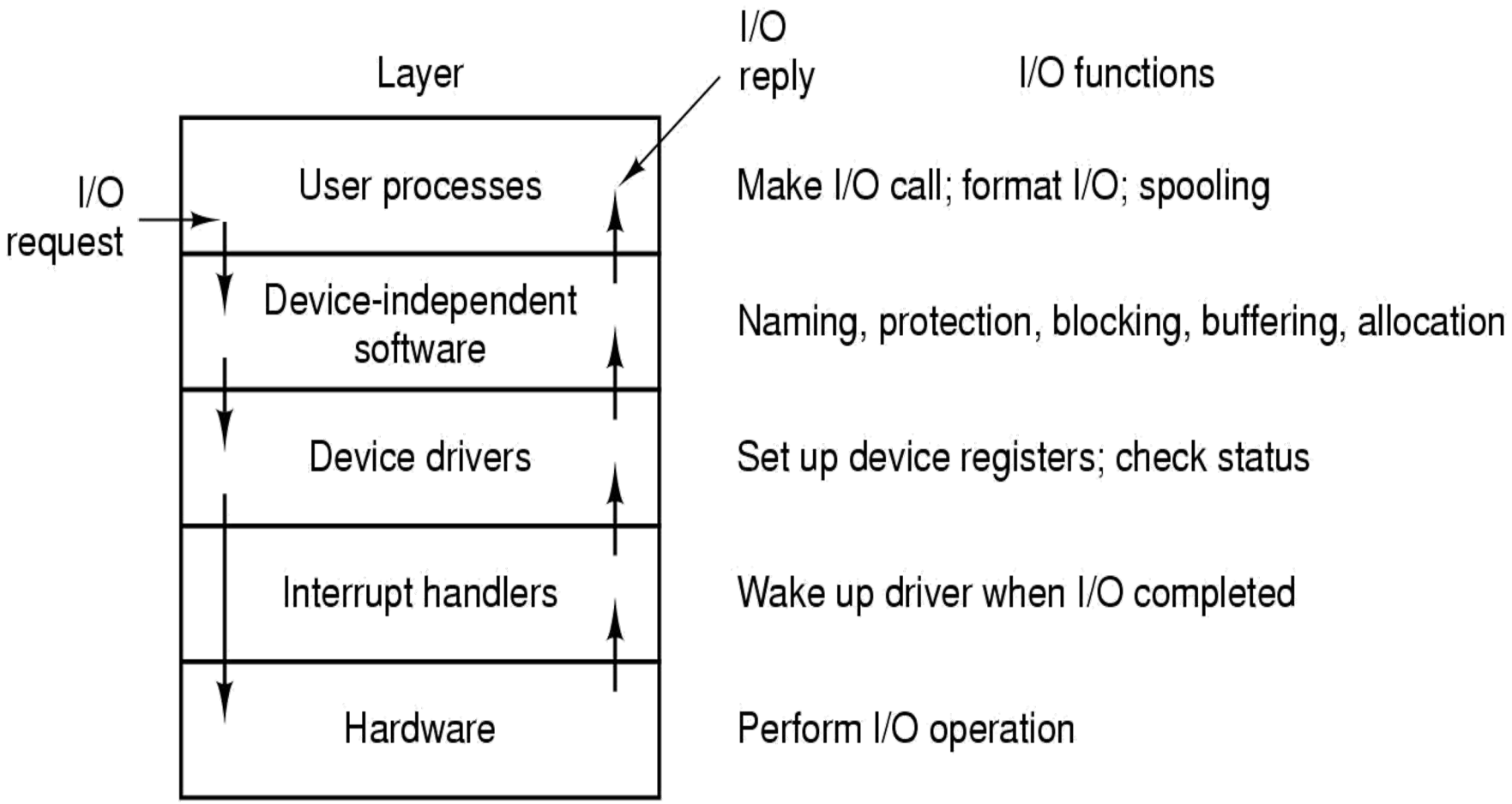
- How to use devices for user?
  - Logical Unit table: Maps logical devices to physical devices

逻辑设备名	物理设备名	驱动程序入口地址
/dev/tty	3	1024
/dev/print	5	2046
…	…	…

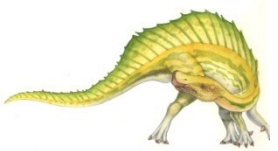
逻辑设备名	系统设备表指针
/dev/tty	3
/dev/print	5
…	…

LUT(Logical Unit Table)

User-Space I/O Software



I/O system Layers and their main functions

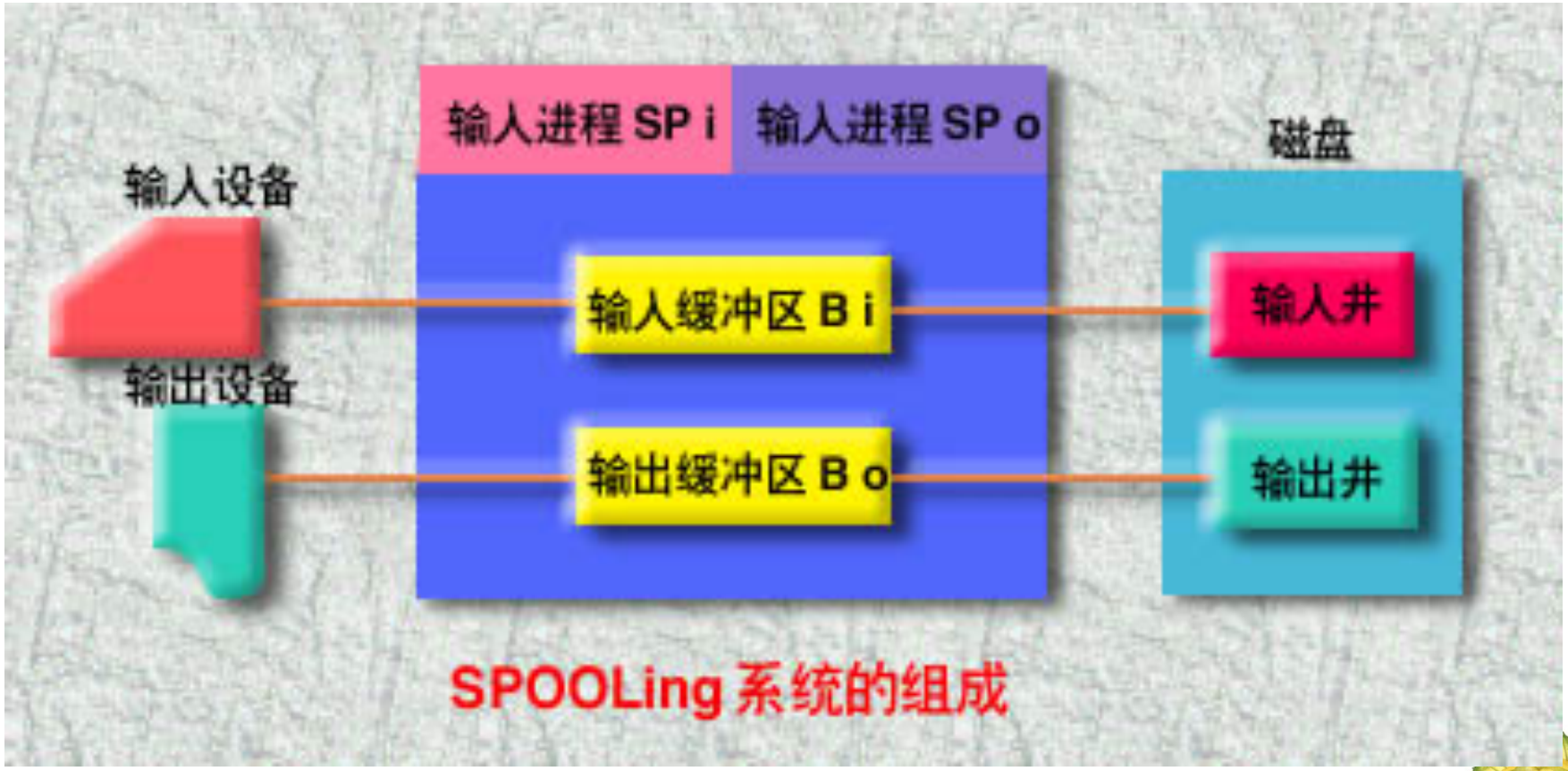


User-Space I/O Software

- It is often convenient for some part of I/O software to run in the user space(such library)
- Most OS do put part of IO in user space!
- Ex: Count = write(fd, buffer, nbytes)
- Library routine write will be linked with the program and contained in the binary program present in memory at run time

User-Space I/O Software

- Other I/O functions in user space include:
  - Spooling: Spooling daemon runs as a user process



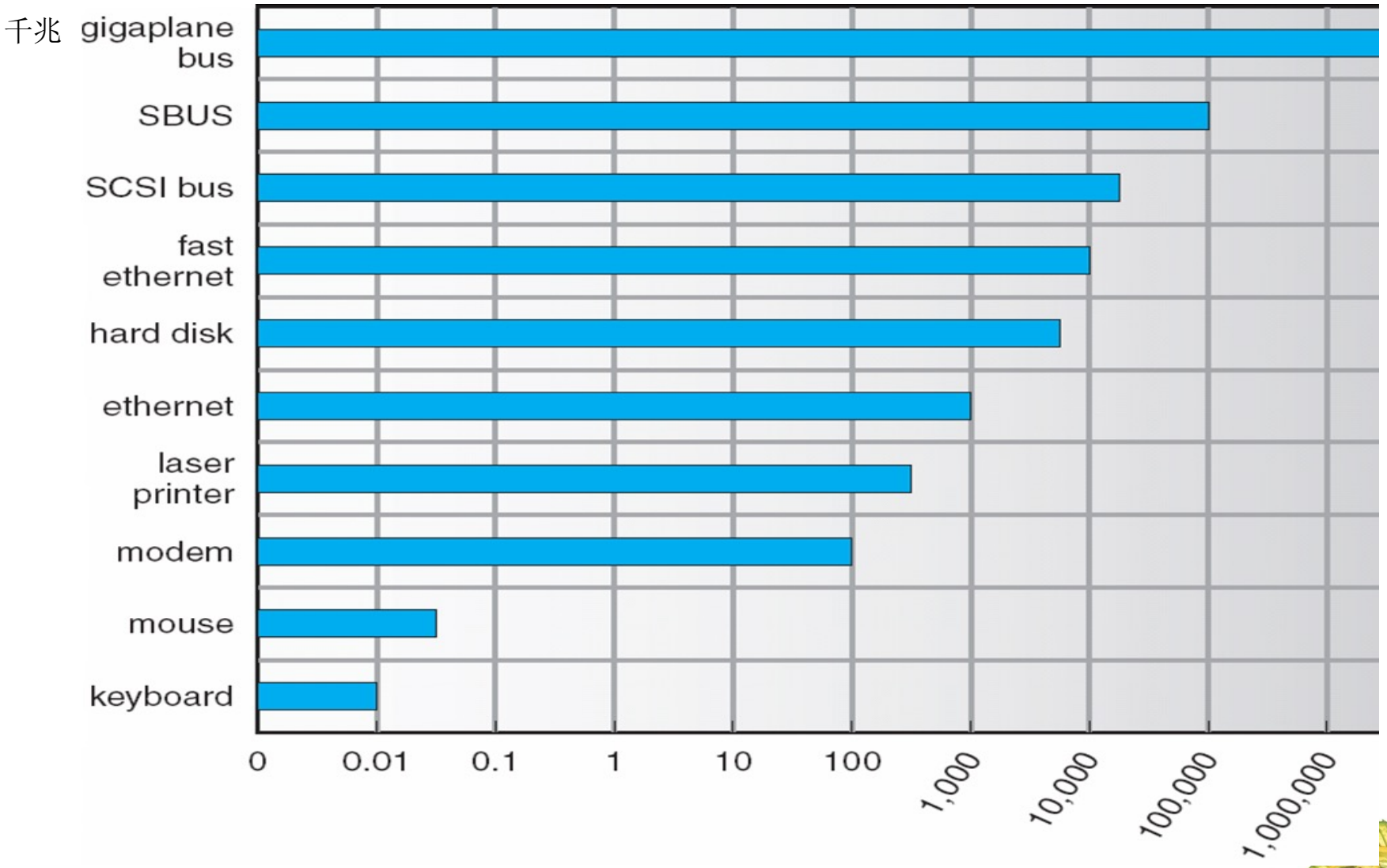
Spooling(Simultaneous Peripheral Operation On-Line)

Buffering

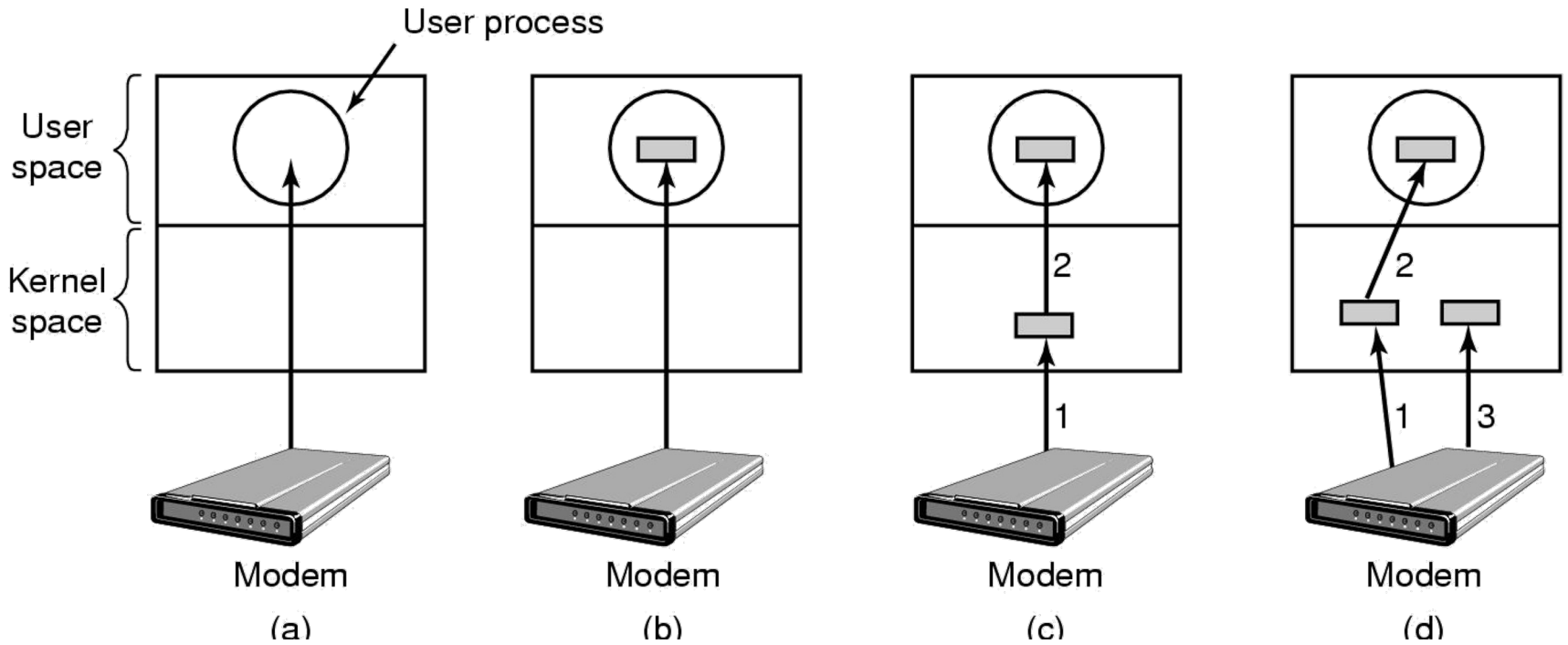
- Buffering--store data in memory while transferring between devices
  - To cope with device speed mismatch
  - To cope with device transfer size mismatch
  - To maintain “copy semantics”

（保证写入盘的数据就是Write()系统调用发生时的版本）

Sun Enterprise 6000 Device-Transfer Rates



Buffering



Unbuffered

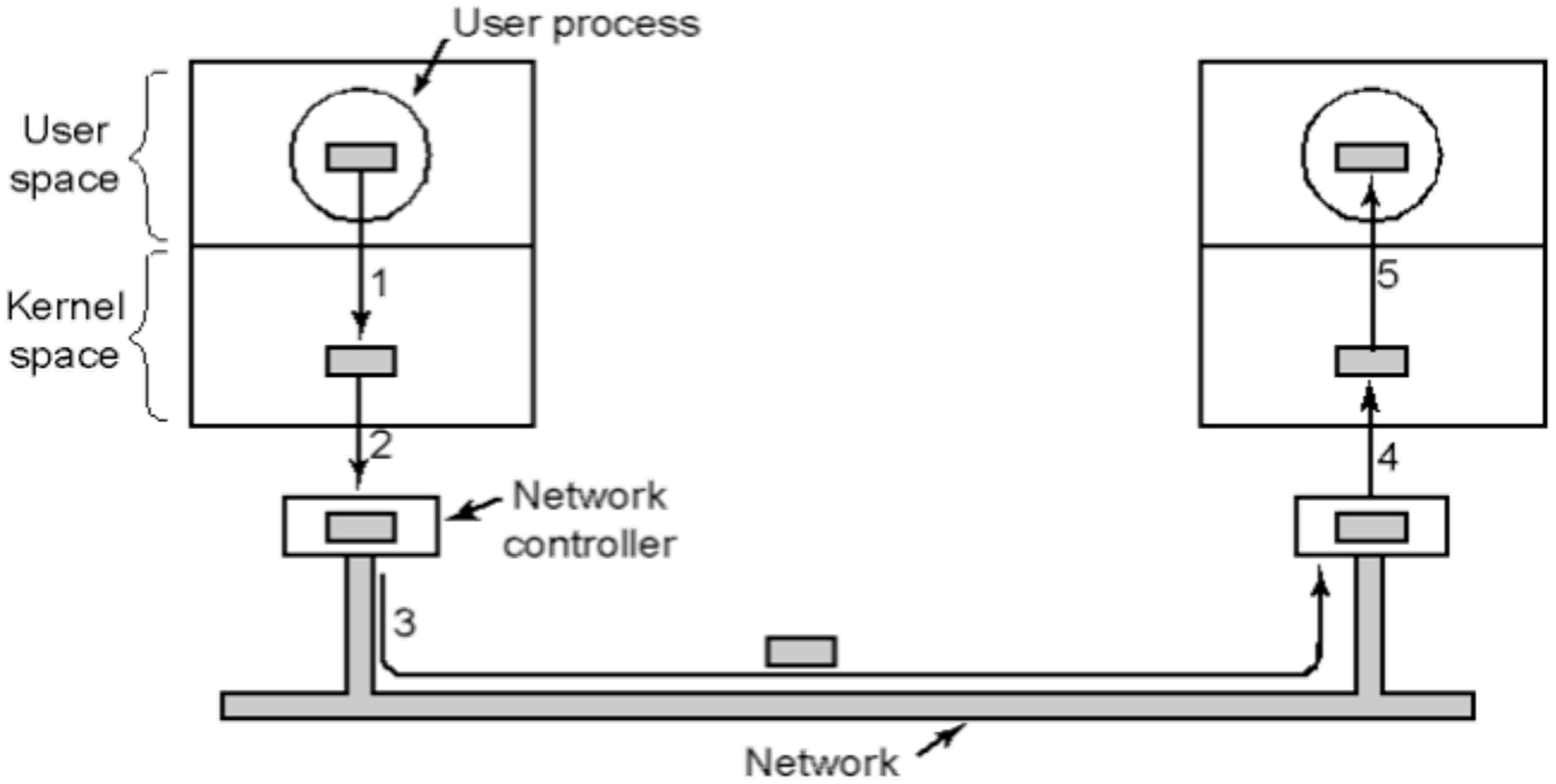
Buffer in user space

Buffer in kernel followed by copying to user space

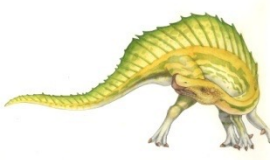
Double buffer in the kernel

Drawback of Buffering

- Slows down performance

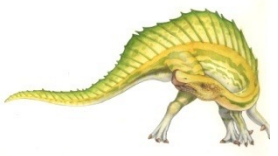


Networking may involve many copies



I/O Scheduling

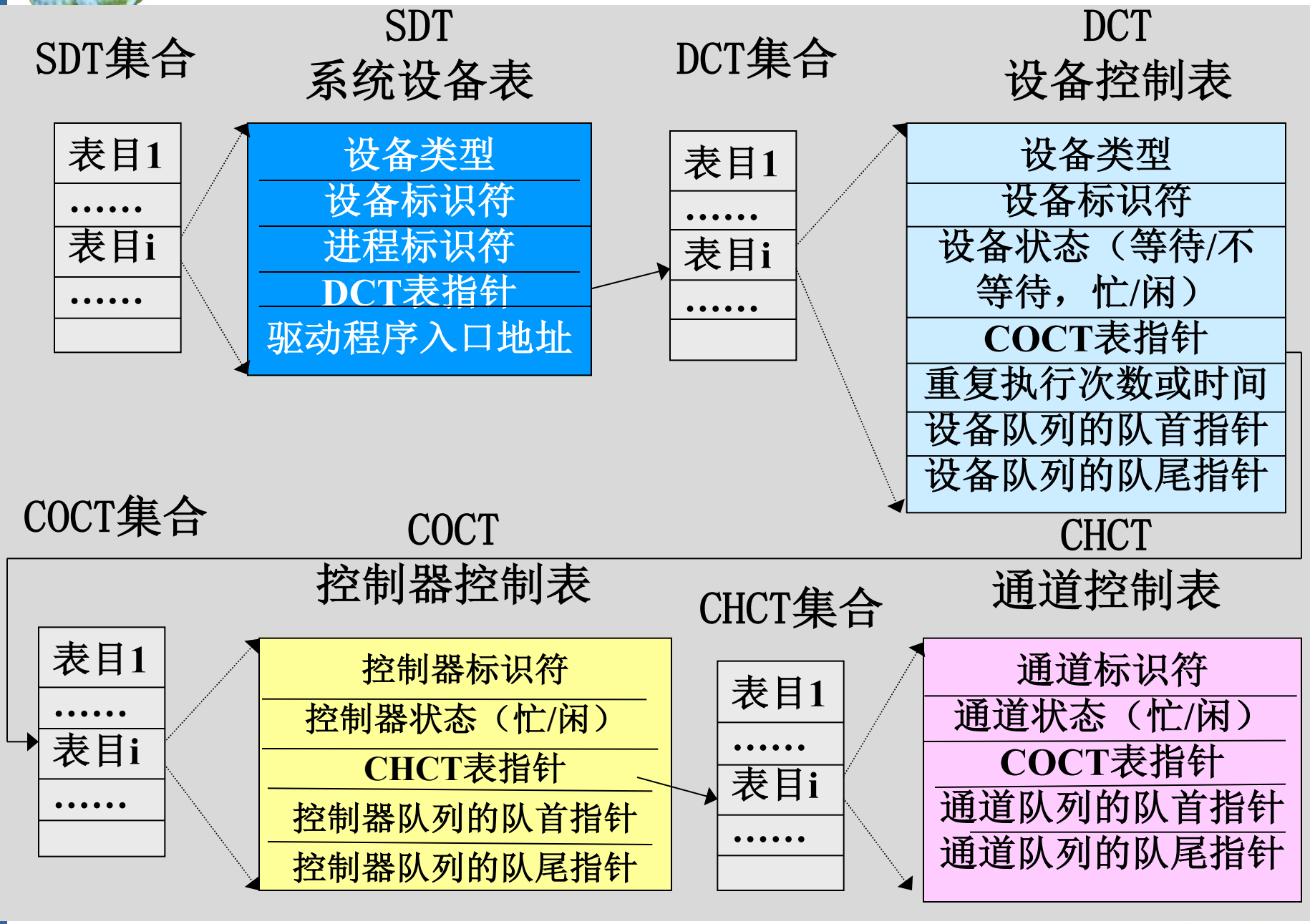
- I/O Scheduling
  - Some I/O request ordering via per-device queue
  - Some OSs try fairness
- Such as Disk Scheduling

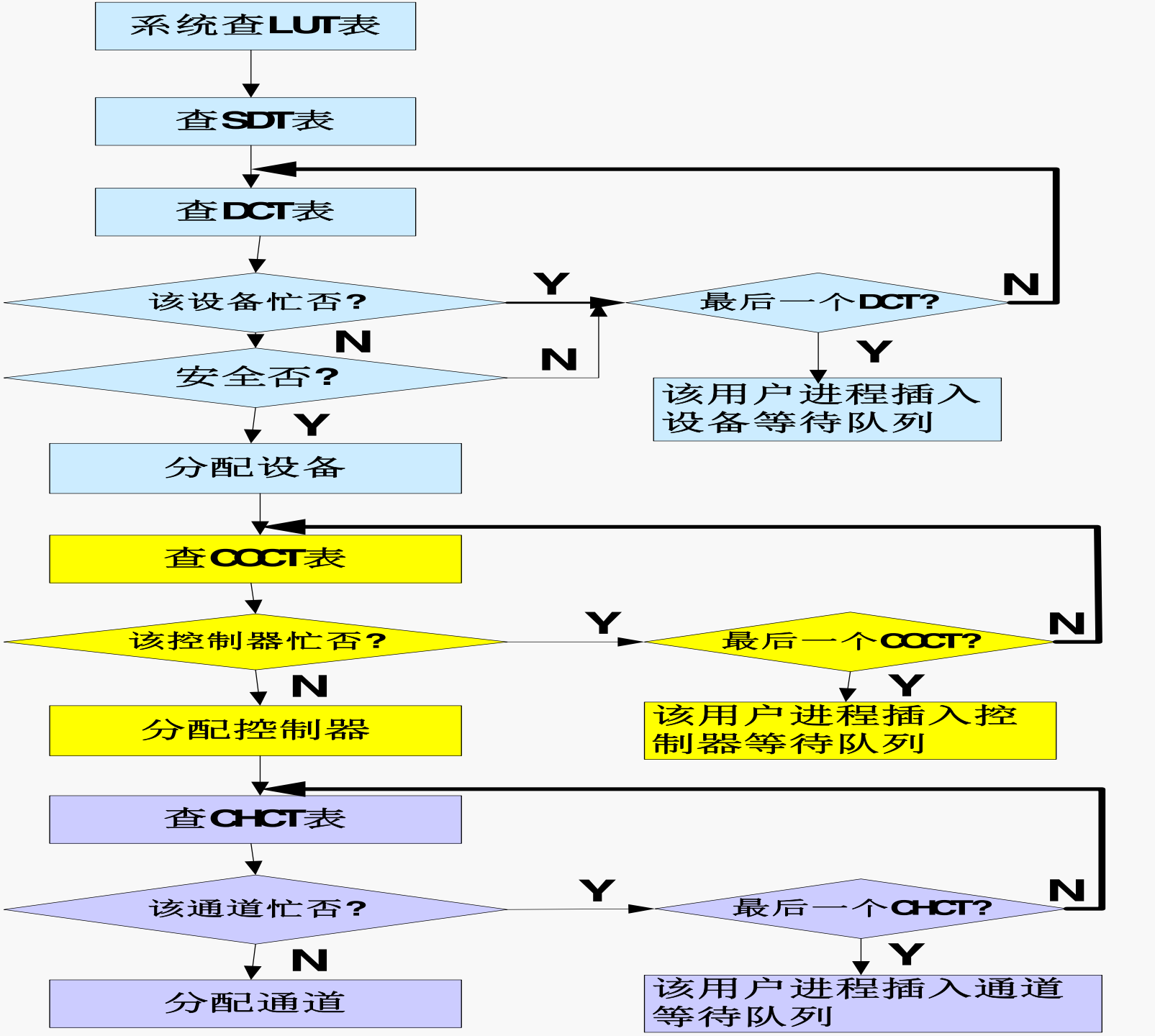


Device Allocation

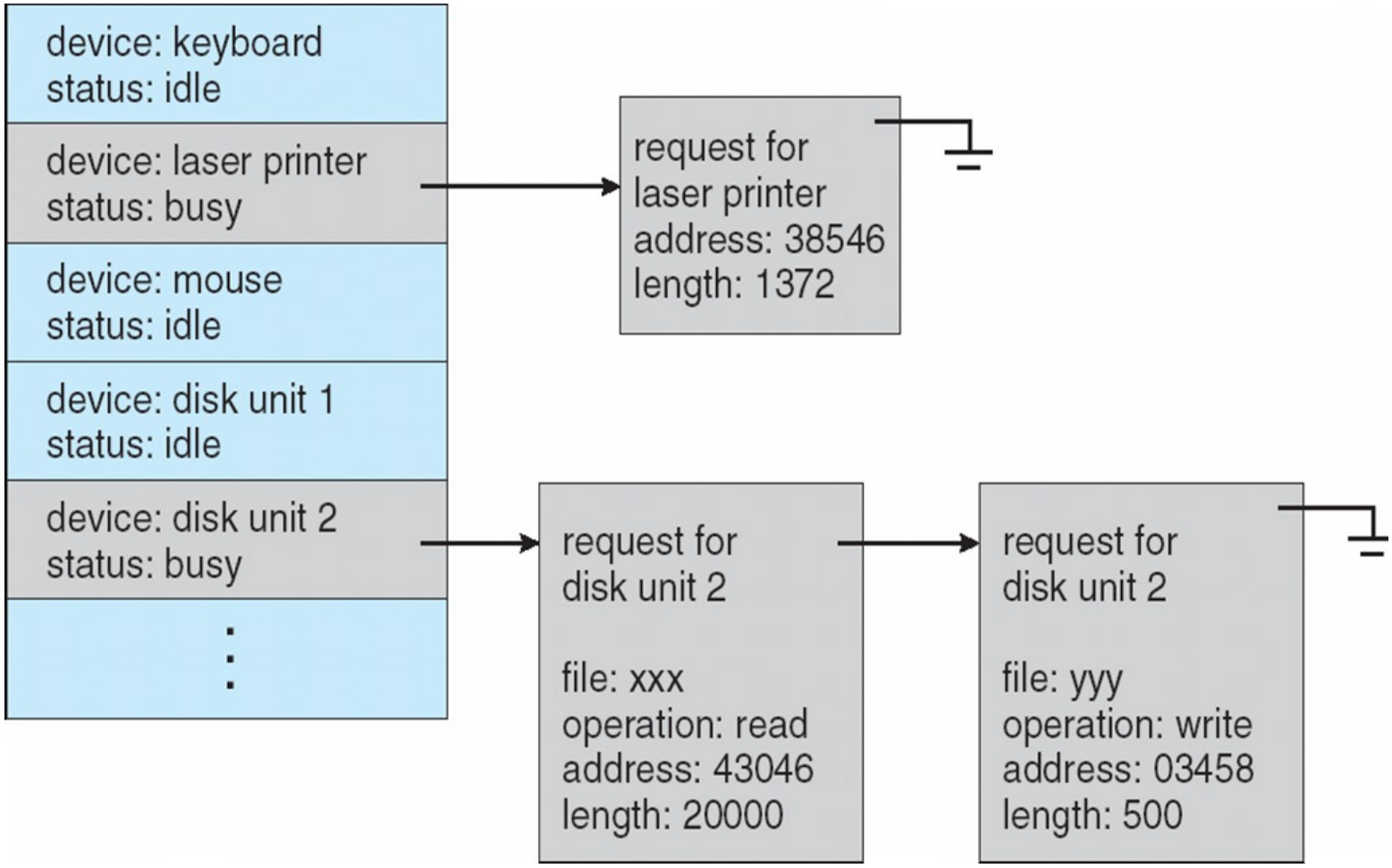
- **Device reservation** - provides exclusive access to a device
  - Another way to deal with concurrent device access.
  - System calls for allocation and deallocation

- **Data structures about device**
  - 设备控制表DCT
  - 控制器控制表COCT
  - 通道控制表CHCT
  - 系统设备表 SDT


SDT集合
SDT
系统设备表
DCT集合
DCT
设备控制表
表目1
……
表目i
……
设备类型
设备标识符
进程标识符
DCT表指针
驱动程序入口地址
表目1
……
表目i
……
设备类型
设备标识符
设备状态（等待/不等待，忙/闲）
COCT表指针
重复执行次数或时间
设备队列的队首指针
设备队列的队尾指针
COCT集合
COCT
控制器控制表
CHCT
通道控制表
CHCT集合
表目1
……
表目i
……
控制器标识符
控制器状态（忙/闲）
CHCT表指针
控制器队列的队首指针
控制器队列的队尾指针
表目1
……
表目i
……
通道标识符
通道状态（忙/闲）
COCT表指针
通道队列的队首指针
通道队列的队尾指针


系统查LUT表
查SDT表
查DCT表
该设备忙否？
Y
最后一个DCT?
N
N
Y
安全否？
N
该用户进程插入设备等待队列
Y
分配设备
查COCT表
该控制器忙否？
Y
最后一个COCT?
N
N
Y
分配控制器
该用户进程插入控制器等待队列
查CHCT表
该通道忙否？
Y
最后一个CHCT?
N
N
Y
分配通道
该用户进程插入通道等待队列

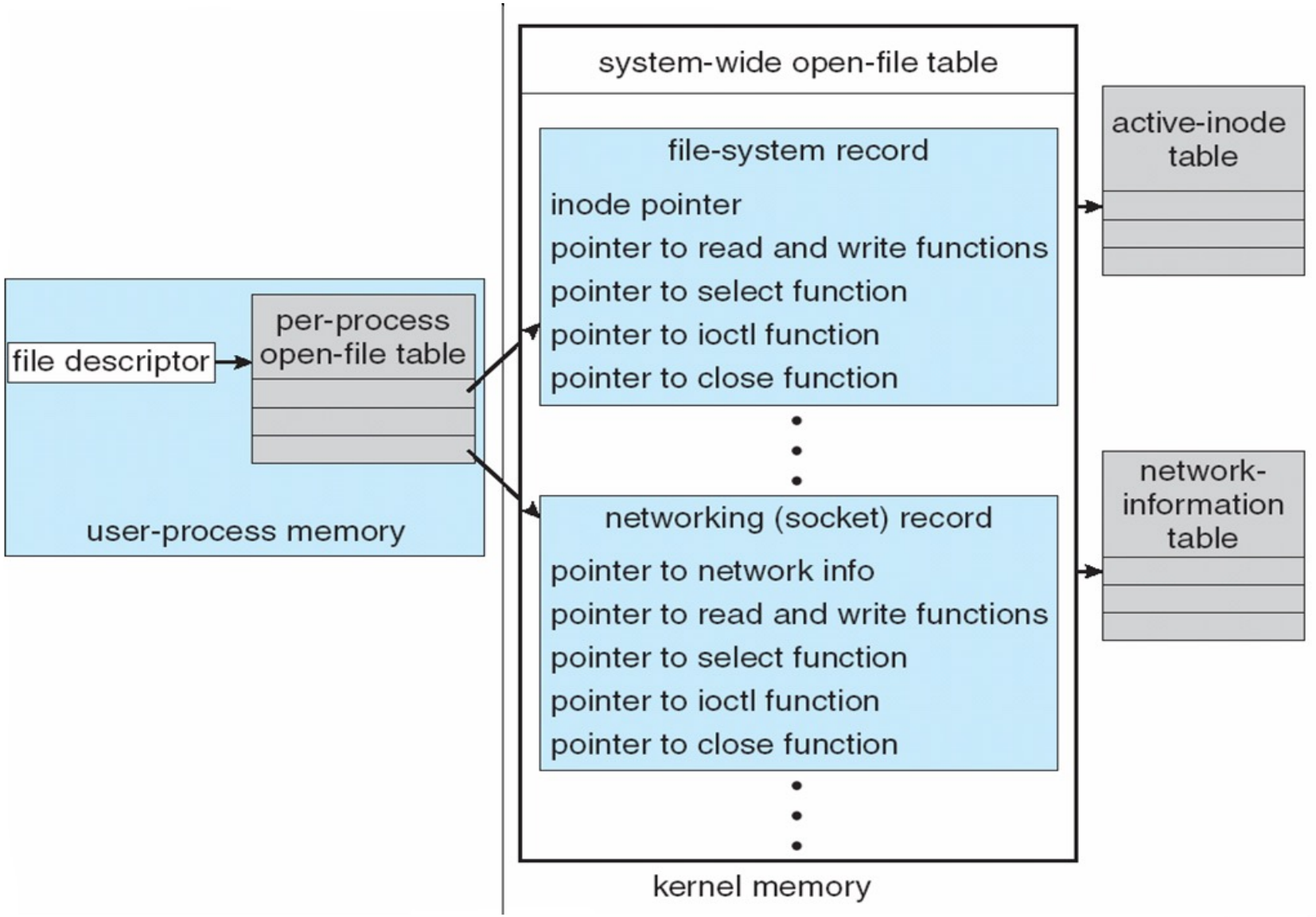
Device-status Table



Kernel Data Structures

- Kernel keeps state info for I/O components, including open file tables, network connections, character device state
- Many, many complex data structures to track buffers, memory allocation, “dirty” blocks
- Some use object-oriented methods and message passing to implement I/O
  - Windows NT, an I/O request is converted into a message through the kernel to the manager and then to the device driver.

UNIX I/O Kernel Structure

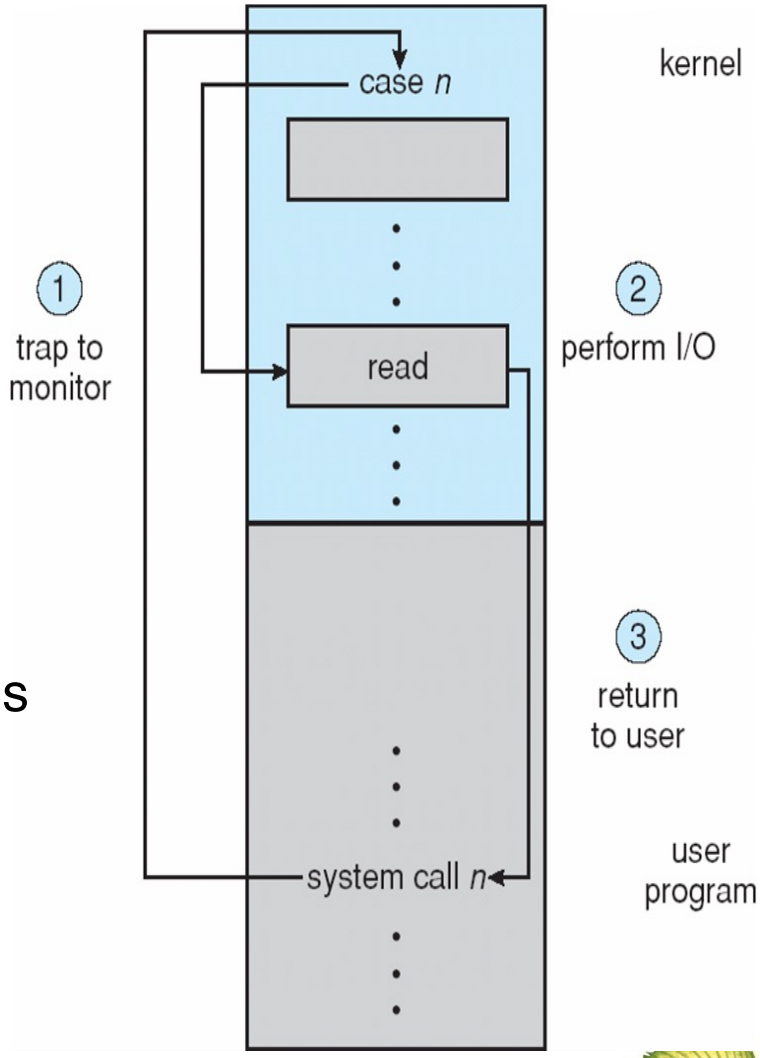


I/O Protection

- User process may accidentally or purposefully attempt to disrupt normal operation via illegal I/O instructions

(无意或有意地通过非法指令破坏正常操作)

  - All I/O instructions defined to be privileged
  - I/O must be performed via system calls
    - Memory-mapped and I/O port memory locations must be protected too



Use of a System Call to Perform I/O

Error Reporting

- Errors far more common in I/O
  - Due to interfacing with outside world
- Errors inside the computer case are rare
- Error reporting depends on types of error
- Programming errors
  - Process ask for something impossible
  - Read from an output device
- Actual I/O errors
  - Accessing a damaged part of disk

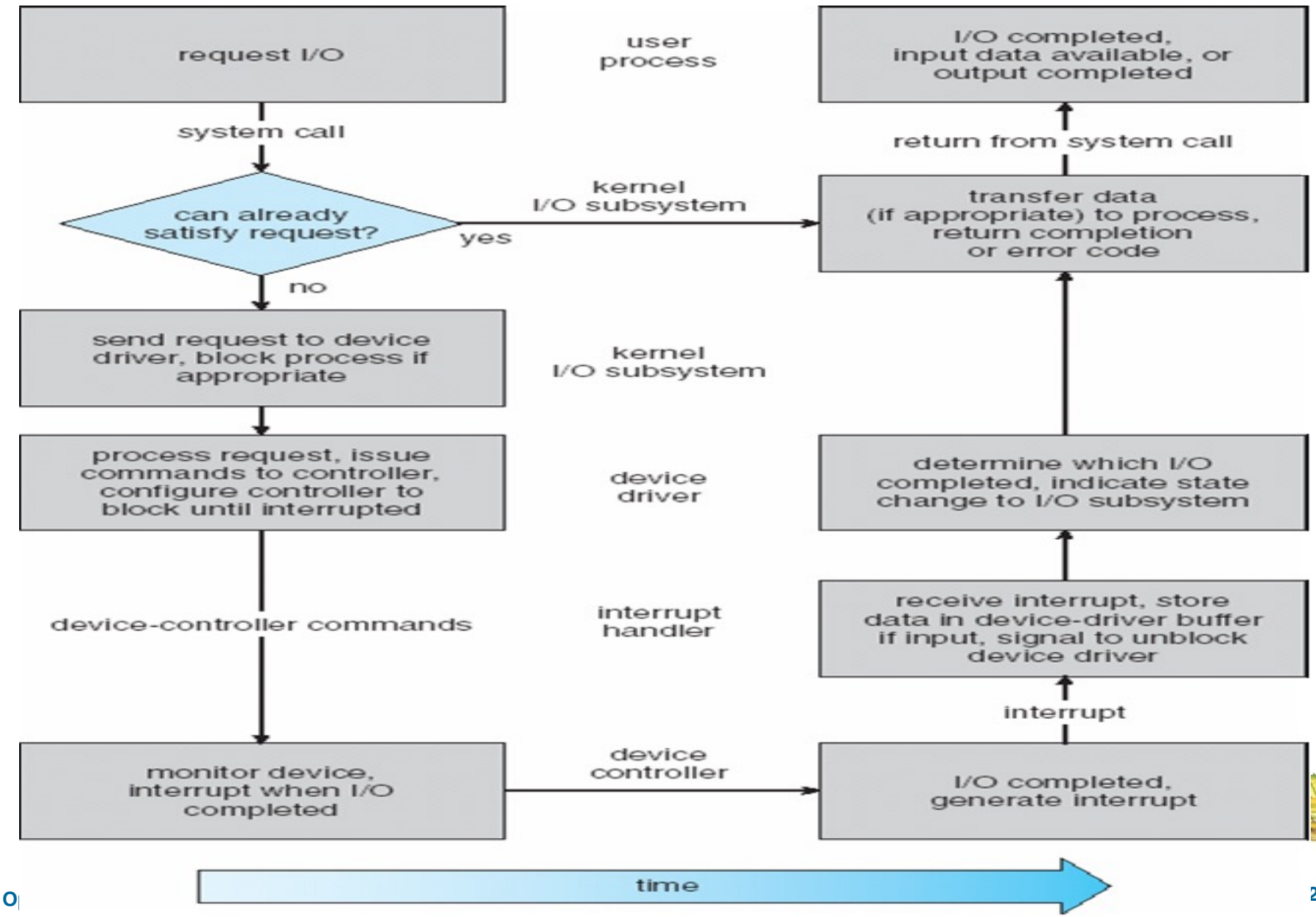
Error Reporting

- For programming errors, just report error back to the caller
- For actual I/O errors,
  - May take corrective measures（纠正措施）
  - May ask the caller what to do
  - May simply fail and return an error code

I/O Requests to Hardware Operations

- Consider reading a file from disk for a process:
  - Determine device holding file
  - Translate name to device representation
  - Physically read data from disk into buffer
  - Make data available to requesting process
  - Return control to process

Life Cycle of An I/O Request



A typical lifecycle of a blocking read request

- A process issues a blocking read system call to a file descriptor of a file that has been opened previously.进程发出阻塞读的系统调用

- **The system-call code in the kernel checks the parameters for correctness.in the case of input,if the data are already available in the buffer cache,the data are returned to the process and the I/O request is completed.**系统检查参数的合法性，如果要读的数据已经在缓冲区中，则将数据返回给用户进程。

- **If the data are not in the buffer cache, a physical I/O needs to be performed,so the process is removed from the run queue and is placed on the wait queue for the device, and the I/O request is scheduled .eventually,the I/O subsystem sends the request to the device driver.depending on the operating system, the request is sent via a subroutine call or via an in-kernel message.**否则，需启动设备进行物理**I/O**。进程进入设备的等待队列，该**I/O**请求等待调度。最终**I/O**子系统把这个**I/O**请求发给了设备驱动程序

A typical lifecycle of a blocking read request

- The device driver allocates kernel buffer space to receive the data, and schedules the I/O, eventually,the driver sends commands to the device controller by writing into the device control registers.设备驱动程序分配用于接收数据的内核缓冲，并调度I/O。最终，驱动程序把该读命令写入控制器的设备控制寄存器中
- **The device controller operates the device hardware to perform the data transfer.设备控制器操纵设备完成数据传输**
- **The driver may poll for status and data. Or it may have set up a DMA transfer into kernel memory. We assume that the transfer is managed by a DMA controller,which generates an interrupt when the transfer completes.驱动程序轮询设备状态或启动DMA传输。这里假定采用的是DMA方式**

A typical lifecycle of a blocking read request

- The appropriate interrupt handler receives the interrupt via the interrupt-vector table,store any necessary data,signals the device driver,and returns from the interrupt.中断处理程序接收到中断信号，保存相应的数据，并唤醒驱动程序

- **The device driver receives the signal,determines which I/O request completed,determines the request's status,and signals the kernel I/O subsystem that the request has been completed.驱动程序确定I/O请求完成了，并确定请求的状态，然后告诉I/O子系统I/O请求已完成**

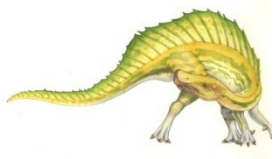
A typical lifecycle of a blocking read request

- The kernel transfers data or return codes to the address space of the requesting process,and moves the process from the wait queue back to the ready queue.内核把数据传给用户进程，把该进程从等待队列中移入就绪队列

- **The process is unblocked when it is moved to the ready queue.when the scheduler assigns the process to the CPU,the process resumes execution at the completion of the system call.**
  接下来，如果调度程序分派**CPU**给该进程，则该进程可以恢复执行

Assignments

1. 总结4种I/O控制方式的优缺点



End of Chapter 13