# Chapter 13: I/O Systems

# Chapter 13: I/O Systems

- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- Streams
- Performance

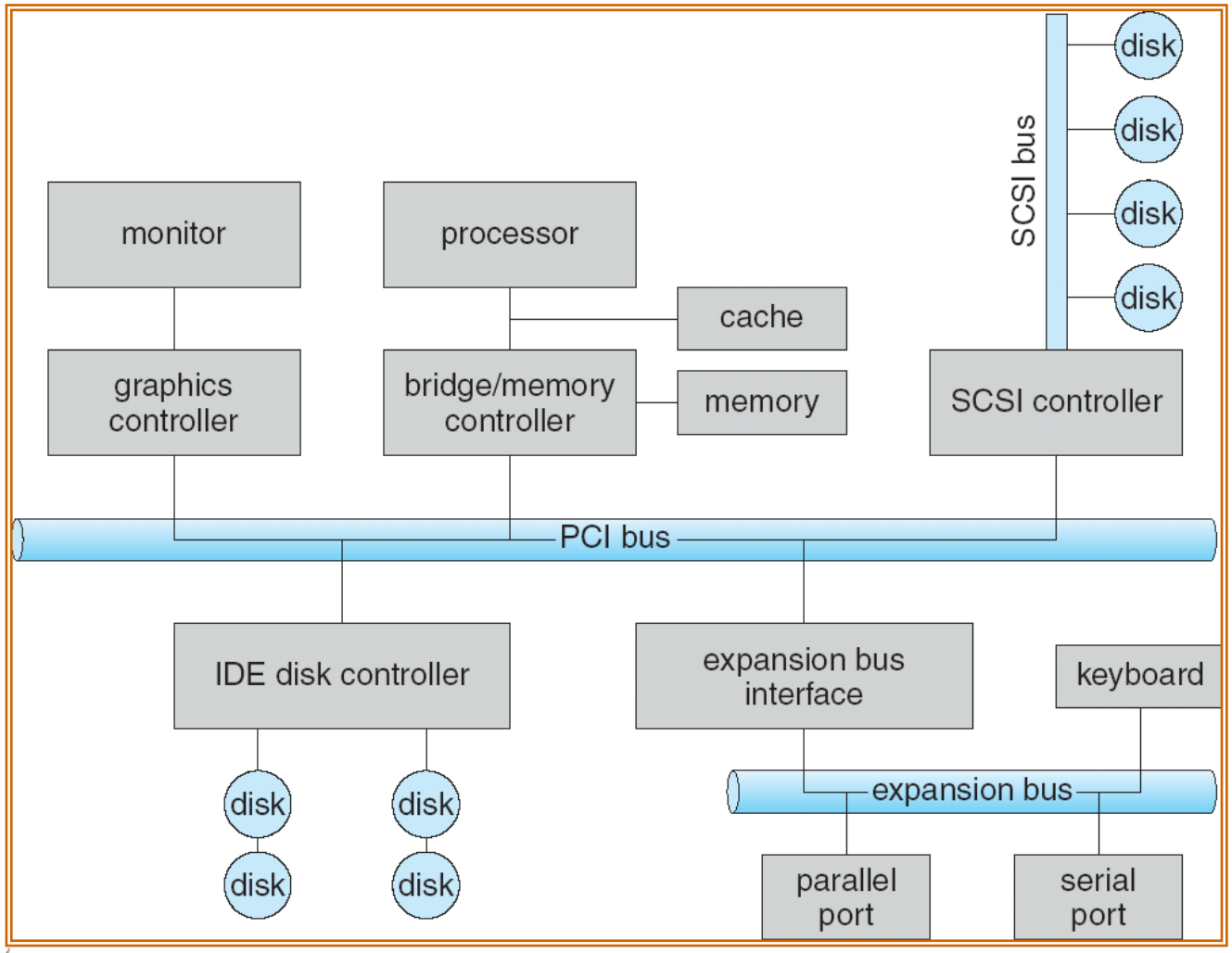
# Objectives

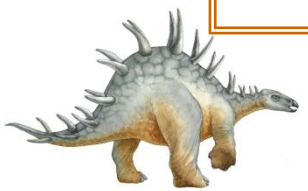
- Explore the structure of an operating system's I/O subsystem
- Discuss the principles of I/O hardware and its complexity
- Provide details of the performance aspects of I/O hardware and software

# I/O Hardware

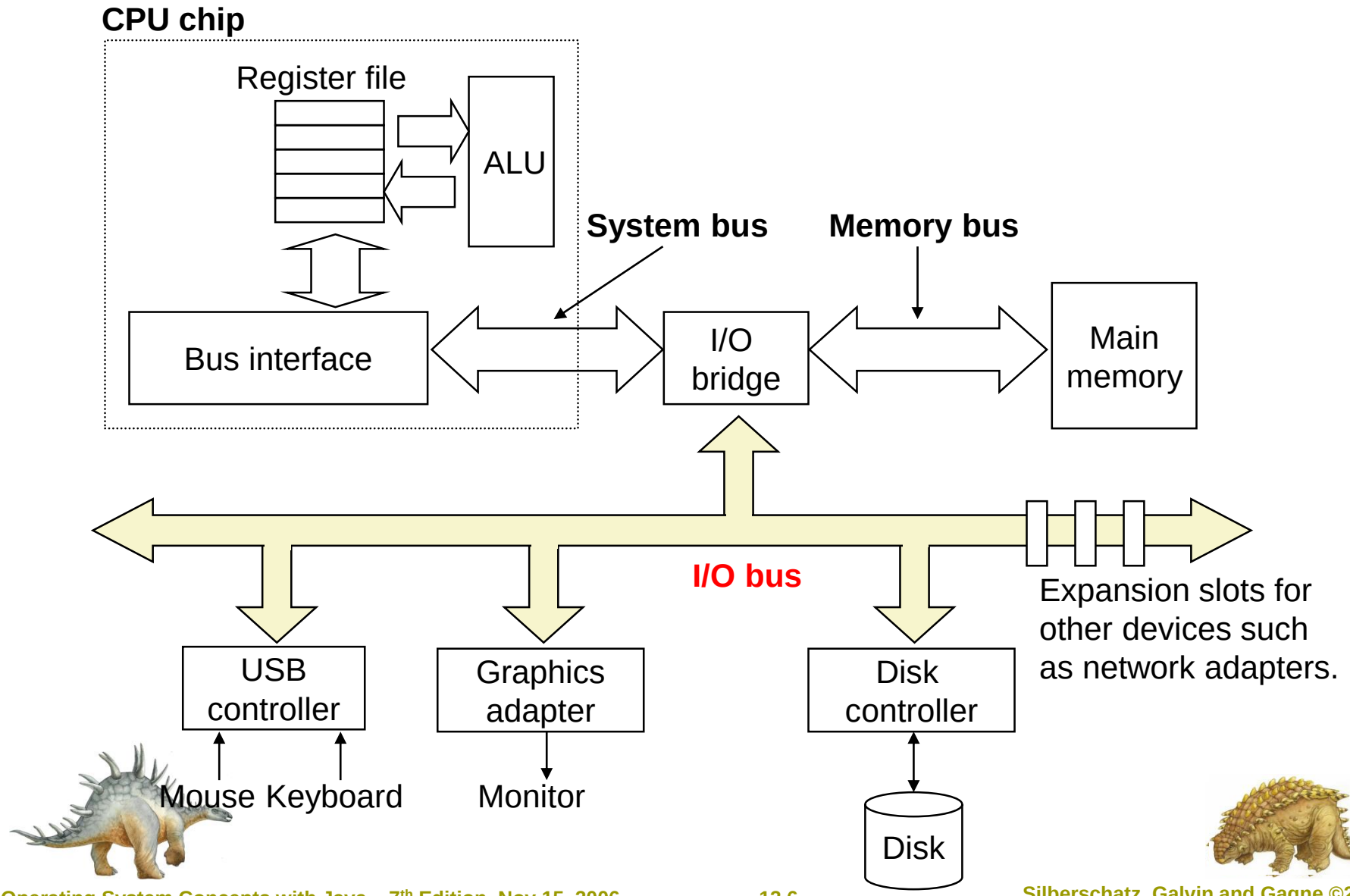
- Incredible variety of I/O devices
- Common concepts
  - **Port**
  - **Bus** (**daisy chain** or shared direct access)
  - **Controller** (**host adapter**)
- I/O instructions control devices
- Devices have addresses, used by
  - Direct I/O instructions（专有I/O通道）
  - **Memory-mapped I/O**（将I/O映射到内存里面，从而使得I/O和内存管理得到统一）

# A Typical PC Bus Structure



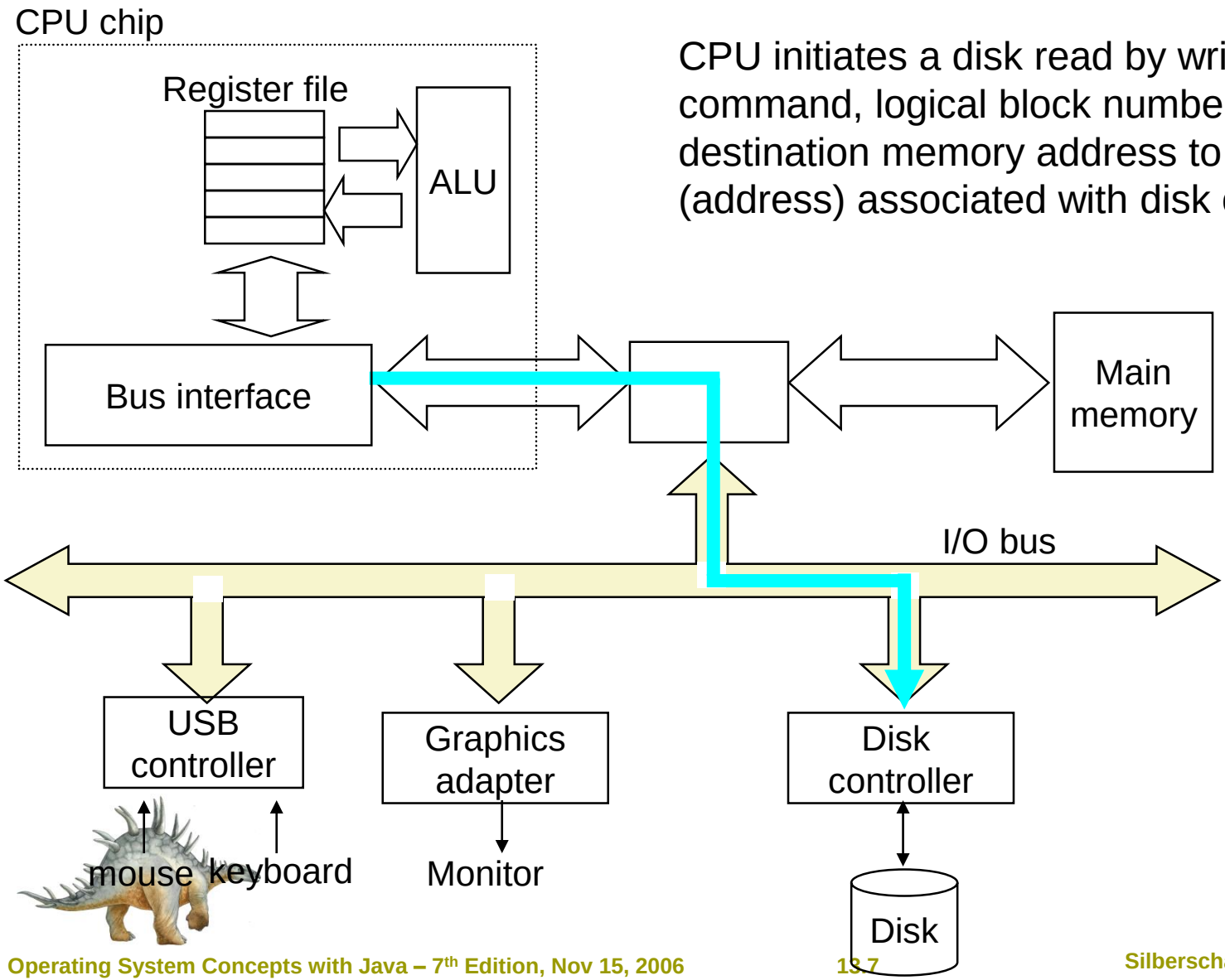



# I/O Bus

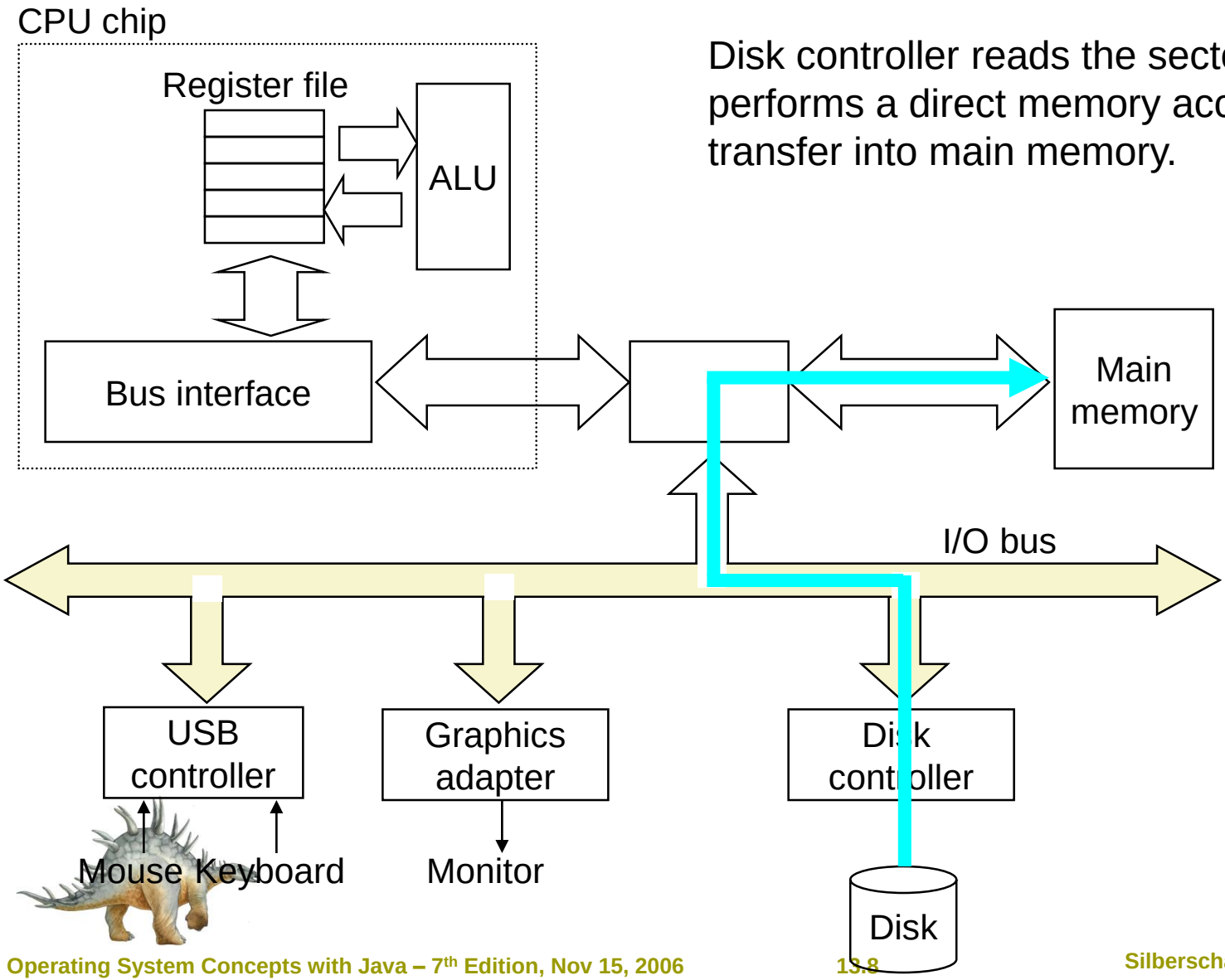

CPU chip
Register file
ALU
System bus
Memory bus
Bus interface
I/O bridge
Main memory
I/O bus
Expansion slots for other devices such as network adapters.
USB controller
Graphics adapter
Disk controller
Mouse
Keyboard
Monitor
Disk

# Reading a Disk Sector (1)

CPU initiates a disk read by writing a command, logical block number, and destination memory address to a port (address) associated with disk controller.



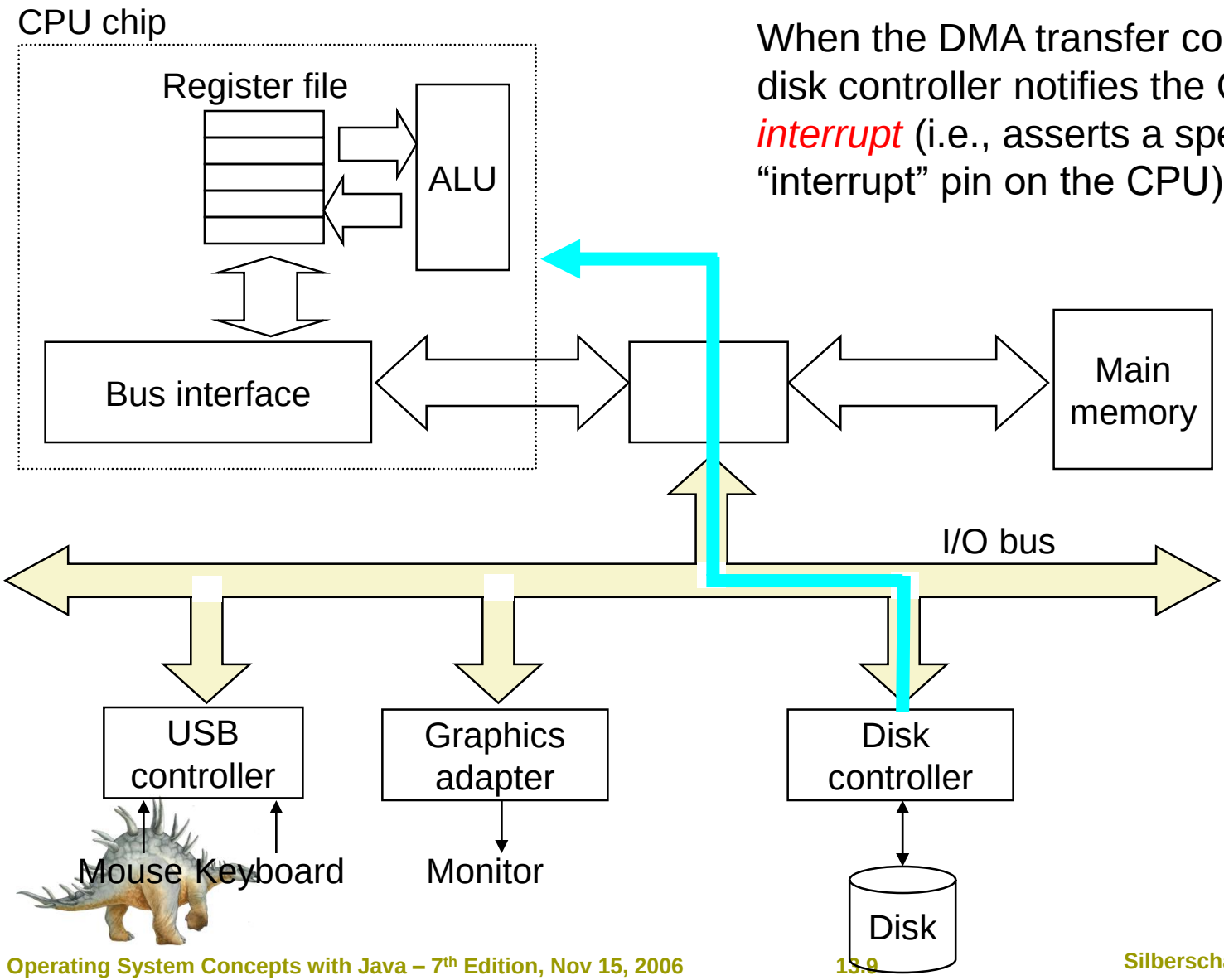
# Reading a Disk Sector (2)




Disk controller reads the sector and performs a direct memory access (DMA) transfer into main memory.

# Reading a Disk Sector (3)

When the DMA transfer completes, the disk controller notifies the CPU with an *interrupt* (i.e., asserts a special "interrupt" pin on the CPU)



# Device I/O Port Locations on PCs (partial)

| 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) |

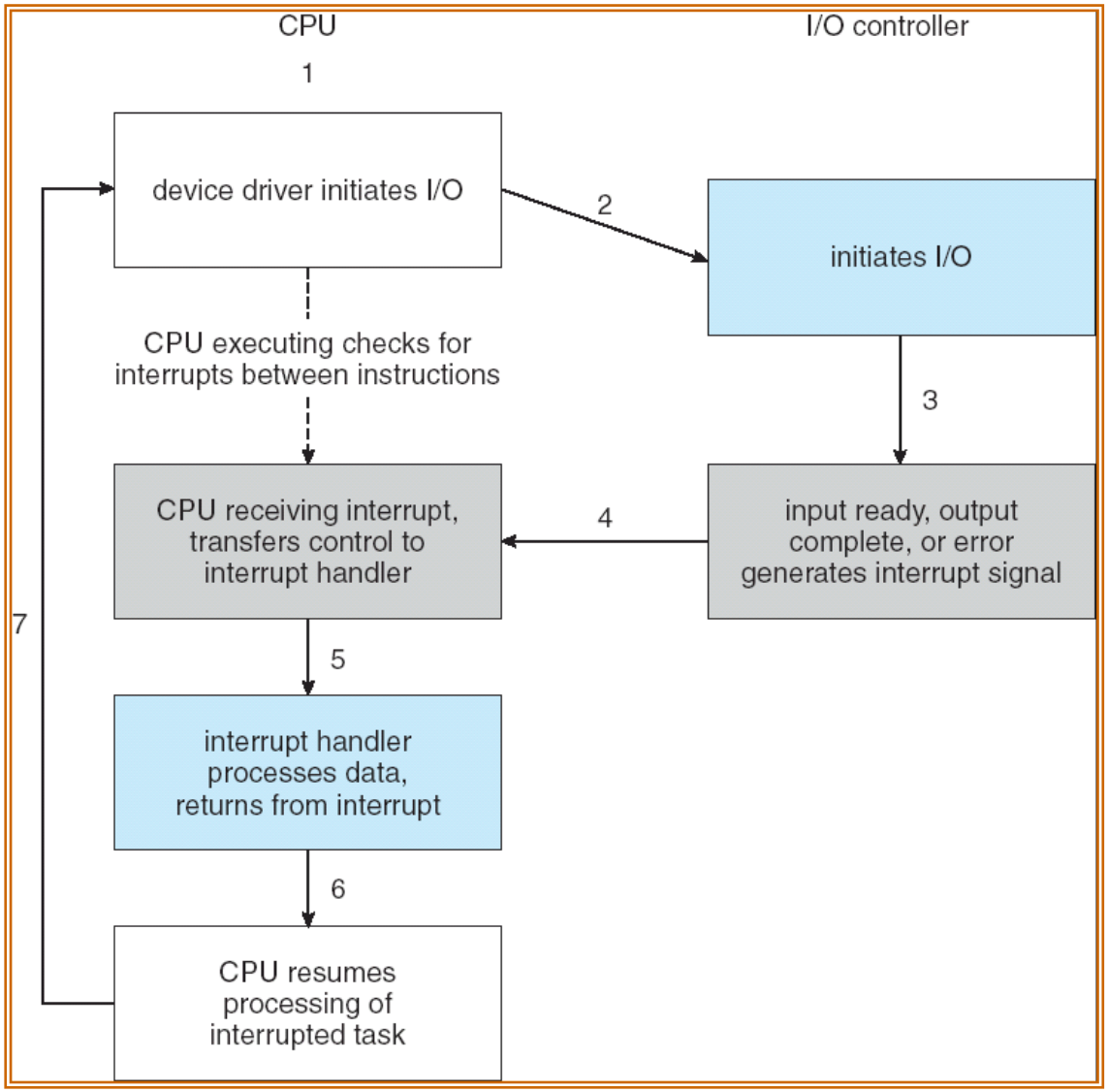
# Polling

- Determines state of device
  - command-ready
  - busy
  - Error
- **Busy-wait** cycle to wait for I/O from device

# Interrupts

- CPU **Interrupt-request line** triggered by I/O device
- **Interrupt handler** receives interrupts
- **Maskable** to ignore or delay some interrupts
- Interrupt vector to dispatch interrupt to correct handler
  - Based on priority
  - Some **nonmaskable**
- Interrupt mechanism also used for exceptions

# Interrupt-Driven I/O Cycle



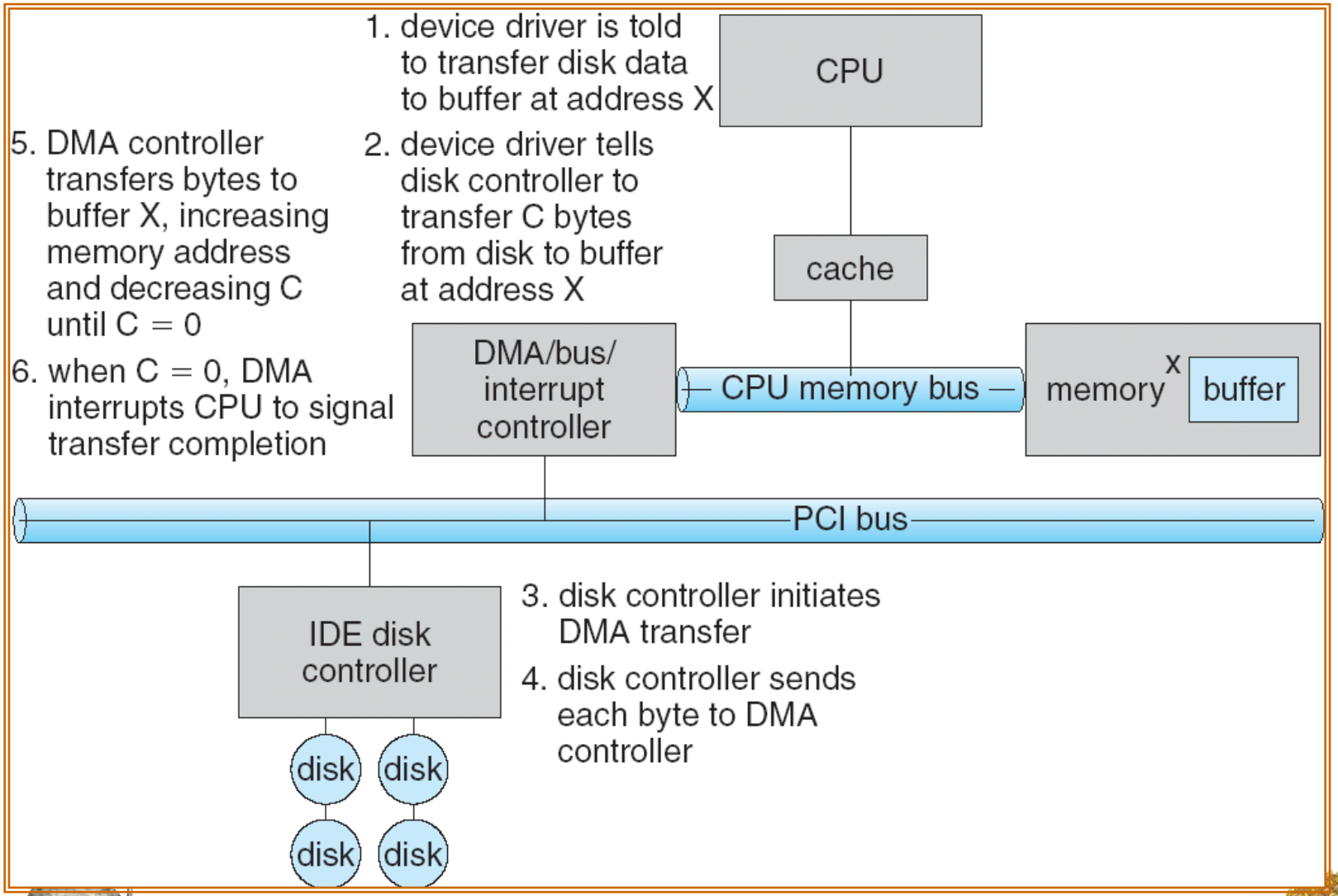

# Intel Pentium Processor Event-Vector Table

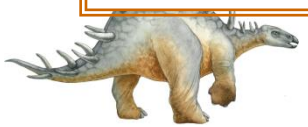
| vector number | description |
|---|---|
| 0 | divide error |
| 1 | debug exception |
| 2 | null interrupt |
| 3 | breakpoint |
| 4 | INTO-detected overflow |
| 5 | bound range exception |
| 6 | invalid opcode |
| 7 | device not available |
| 8 | double fault |
| 9 | coprocessor segment overrun (reserved) |
| 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 |

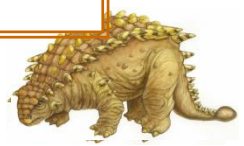
# Direct Memory Access

- Used to avoid **programmed I/O** for large data movement
- Requires **DMA** controller
- Bypasses CPU to transfer data directly between I/O device and memory

# Six Step Process to Perform DMA Transfer



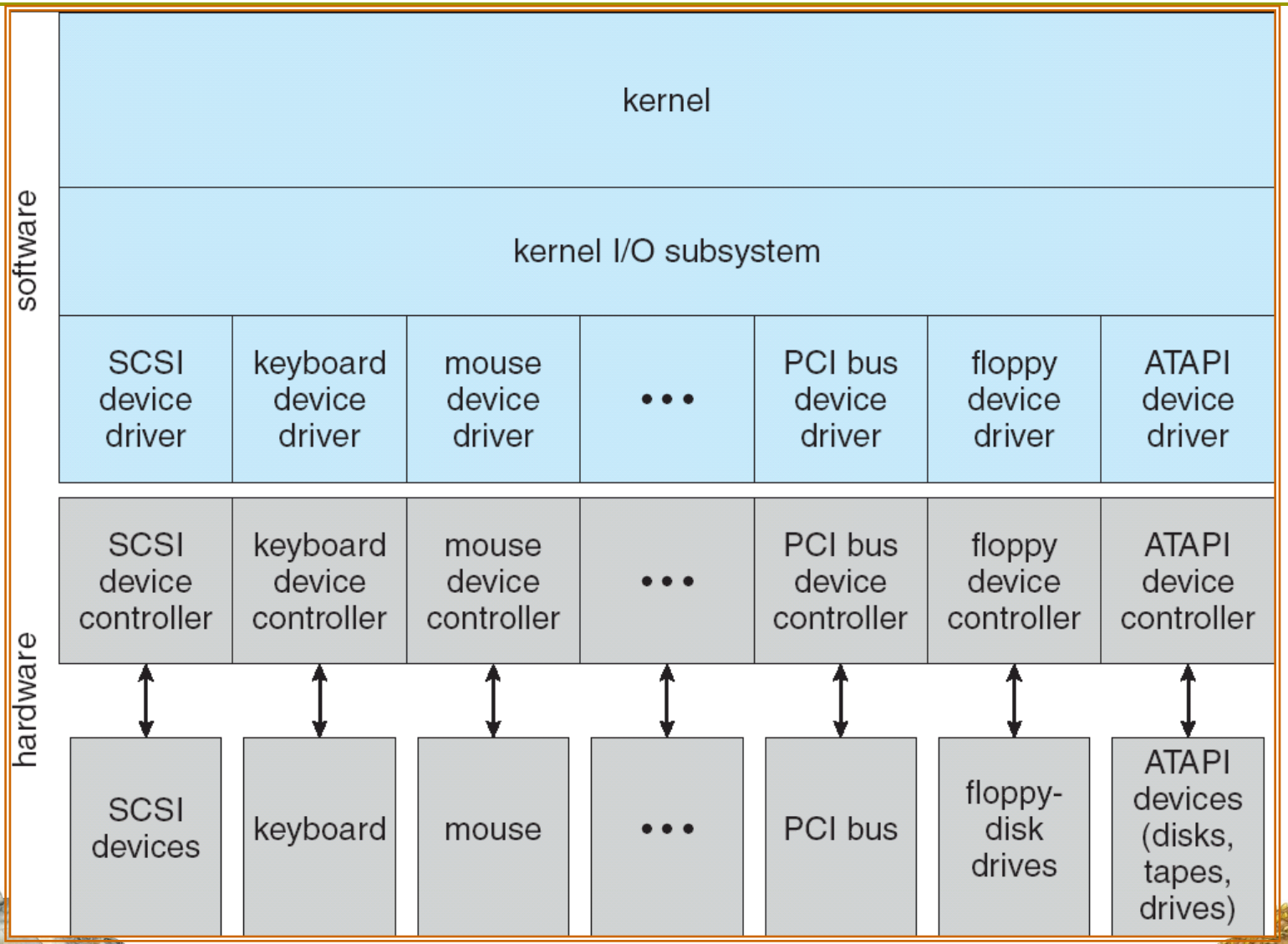





# Application I/O Interface

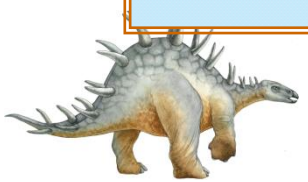
- I/O system calls encapsulate device behaviors in generic classes
- Device-driver layer hides differences among I/O controllers from kernel
- Devices vary in many dimensions
  - **Character-stream** or **block**
  - **Sequential or random-access**
  - **Sharable or dedicated**
  - **Speed of operation**
  - **read-write, read only,** or **write only**

# A Kernel I/O Structure



# Characteristics of I/O Devices

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

# Network Devices

- Varying enough from block and character to have own interface
- Unix and Windows NT/9*x*/2000 include socket interface
  - Separates network protocol from network operation
  - Includes select functionality
- Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes)

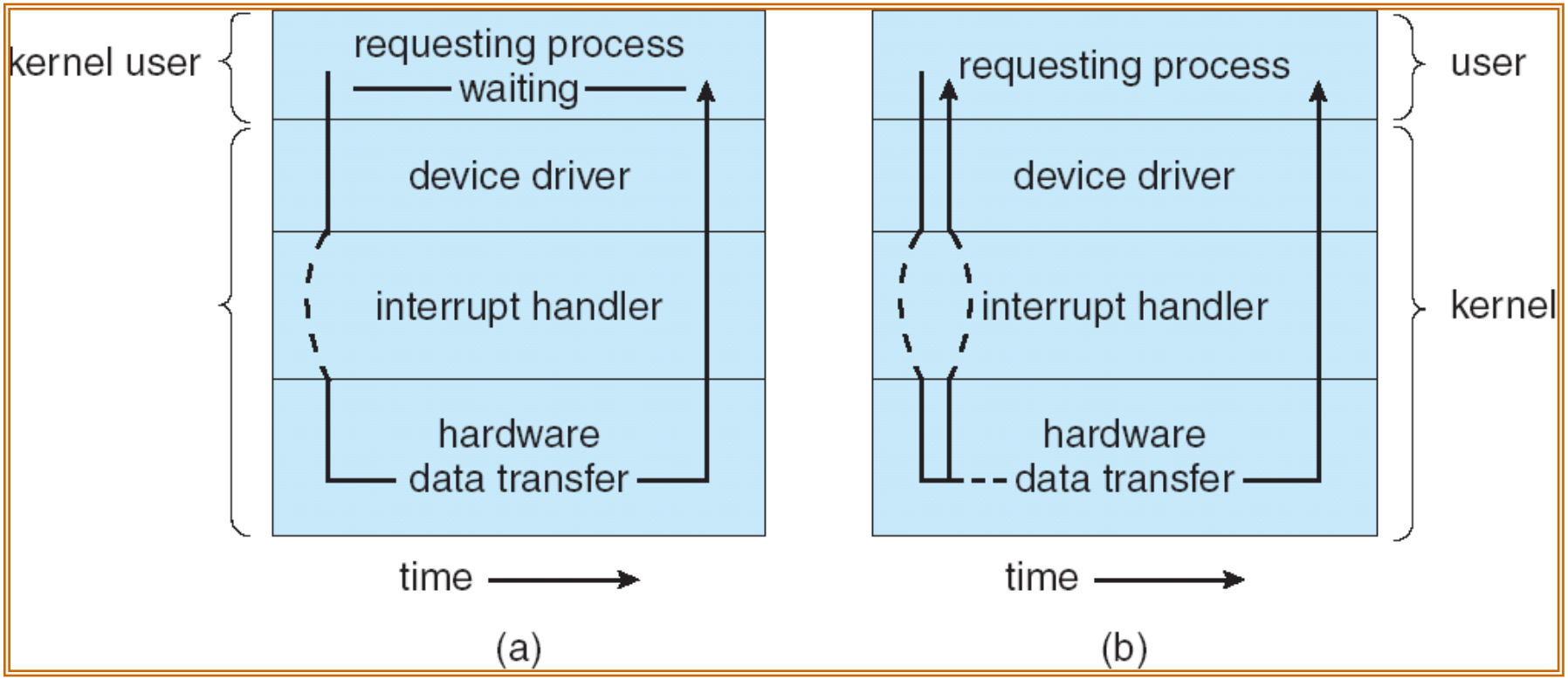
# Clocks and Timers

- Provide current time, elapsed time, timer
- **Programmable interval timer** used for timings, periodic interrupts
- ioctl (on UNIX) covers odd aspects of I/O such as clocks and timers

# Blocking and Nonblocking I/O

- **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
  - Difficult to use
  - I/O subsystem signals process when I/O completed

# Two I/O Methods



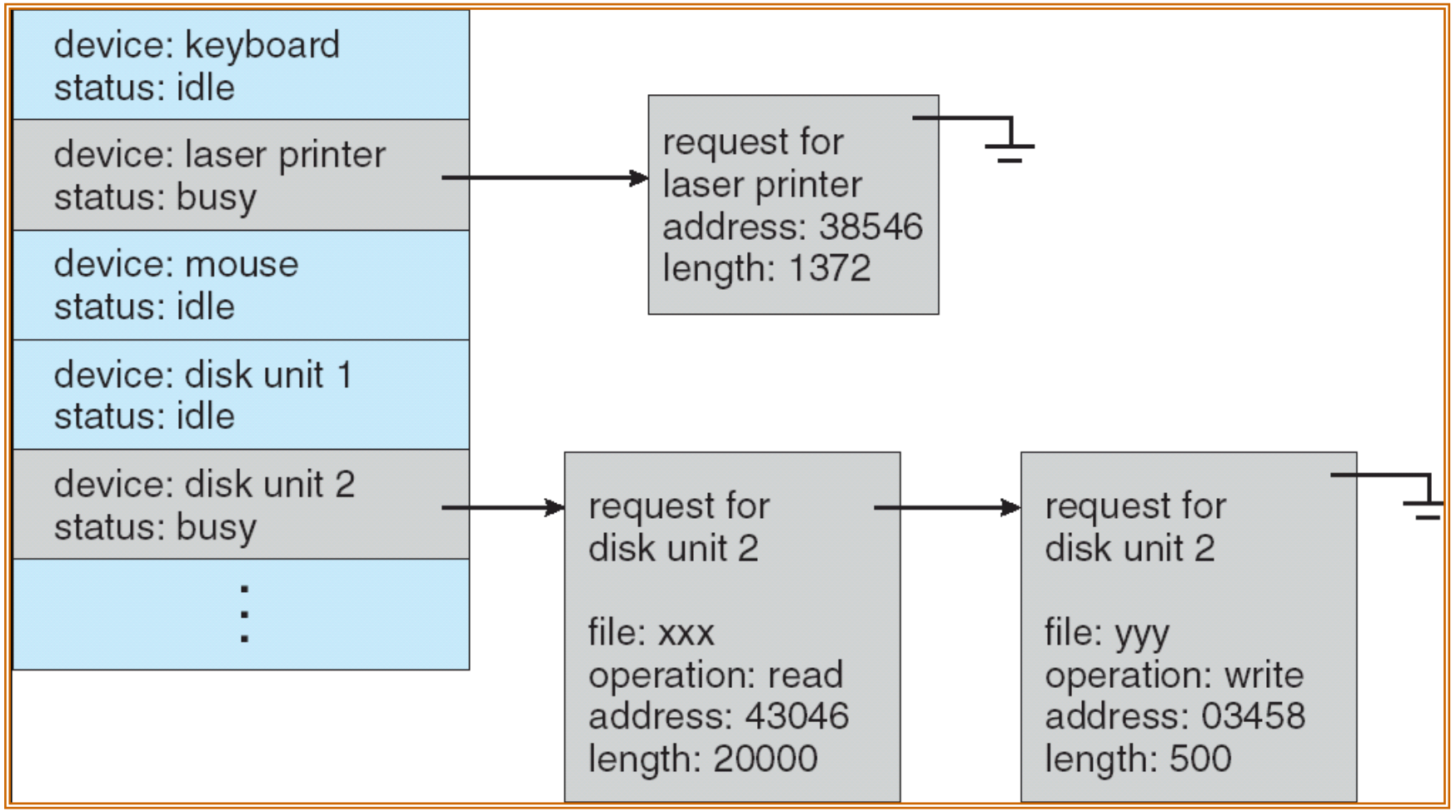

Synchronous

Asynchronous

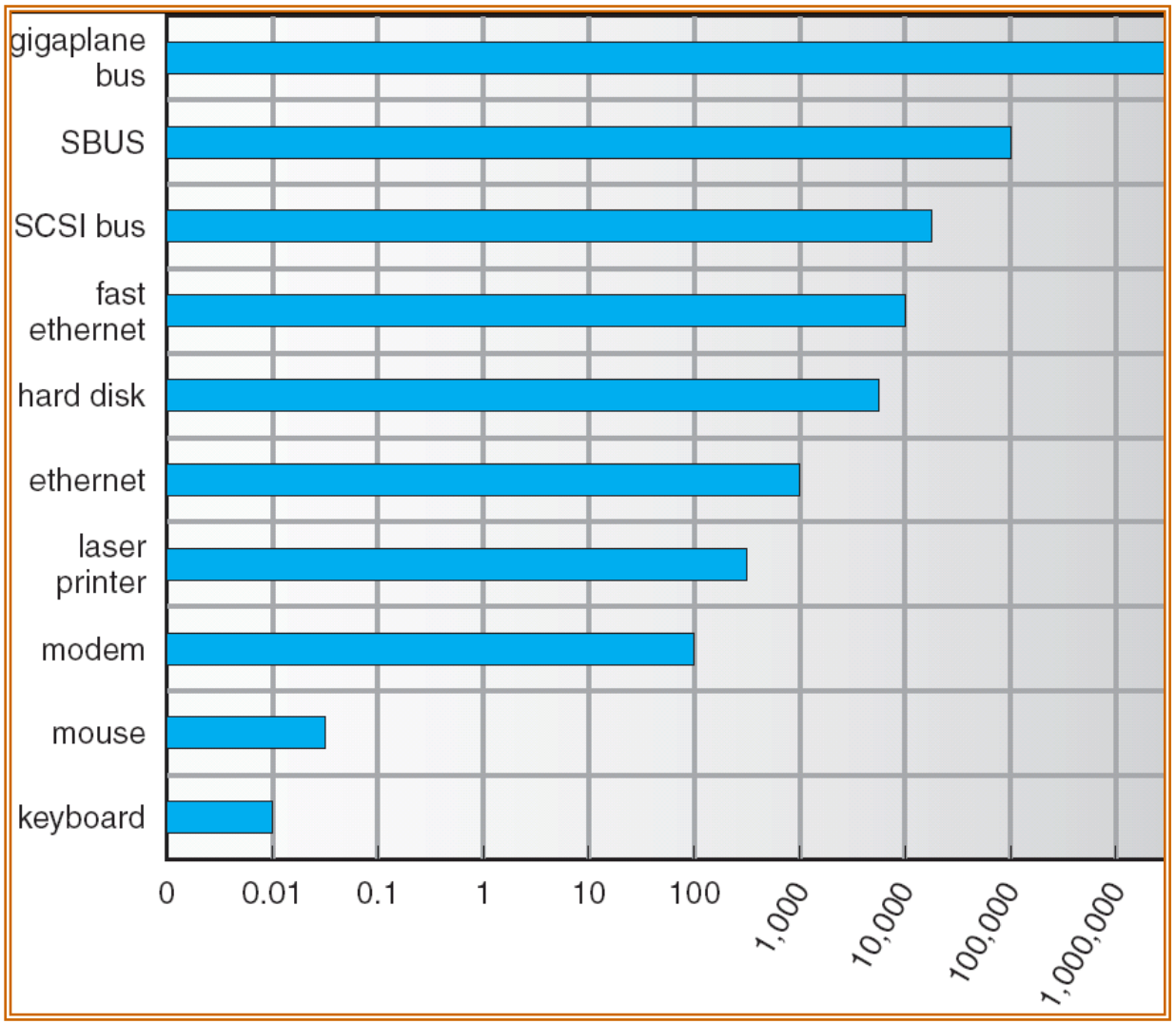
# Kernel I/O Subsystem

- **Scheduling**
  - Some I/O request ordering via per-device queue
  - Some OSs try fairness
- **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”

# Device-status Table




# Sun Enterprise 6000 Device-Transfer Rates




# Kernel I/O Subsystem

- **Caching** - fast memory holding copy of data
  - Always just a copy (buffer)
  - Key to performance
- **Spooling** - hold output for a device
  - If device can serve only one request at a time
  - i.e., Printing
- **Device reservation** - provides exclusive access to a device
  - System calls for allocation and deallocation
  - Watch out for deadlock

# Error Handling

- OS can recover from disk read, device unavailable, transient write failures
- Most return an error number or code when I/O request fails
- System error logs hold problem reports

# Improving Performance

- Reduce number of context switches
- Reduce data copying
- Reduce interrupts by using large transfers, smart controllers, polling
- Use DMA
- Balance CPU, memory, bus, and I/O performance for highest throughput

# End of Chapter 13

# I/O管理

设备管理是指操作系统对除了CPU和内存以外的所有输入/输出设备的管理，诸如设备控制器、通道、中断控制器等等，因此为了提高计算机系统的整体效率，除了需要对中央处理器CPU合理调度、对内存合理使用之外，对系统中的设备也要实施行之有效的管理，这样才能真正发挥计算机系统的整体效率。

# I/O外设分类

## 人可读；

- 计算机用户间的交互；
- 打印机；
- 视频终端；
- 键盘、鼠标；

## 通信；

- 远程设备间通信；
- 数字线路驱动器；
- 调制解调器；

## 机器可读；

- 电子设备间通讯；
- 磁盘、磁带设备；
- 传感器；
- 控制器；
- 传动器；

# I/O功能的组织

- 可编程I/O

  处理器代表进程给I/O模块发送一个I/O命令，进程进入忙等待，操作完成后，才可继续执行；

- 中断驱动I/O

- 直接存储访问（DMA）

  DMA模块控制主存和I/O模块间数据交换，只有当整个数据块结束后，处理器才被中断；

# I/O功能的发展

1. **处理器直接控制外部设备；**

   由 CPU 负责在机器内存与设备控制器数据寄存器之间进行数据传送

2. **非中断可编程I/O**

3. **中断驱动I / 0**

   为了减少设备驱动程序不断地询问控制器状态寄存器的开销当 I/O 操作结束后，由设备控制器主动通知设备驱动程序

4. **DMA**

5. **通道**

6. **I/O处理器**

# 中断方式传送

为了减少CPU测试等待时间和CPU与外设的并行工作能力，引入了中断控制传输方式。

- 进程CPU发START命令启动外设准备数据；
- 进程放弃CPU，进程调度选择其它进程占有CPU；
- 外设准备好数据，通过中断控制器向CPU发中断请求信号，CPU接到该信号且在允许中断的情况下，响应该中断请求，转中断处理程序，在中断处理程序中完成数据输入。

# DMA方式

## 存储器直接存取方式；

内存和外设之间开辟直接的数据交换通路，由DMA控制器完成数据交换。

DMA方式 在传送开始需要CPU做一些初始化和传输结束做一些善后处理工作之外，在整个数据传输过程中，不需要CPU任何干预。

1. 进程要求输入数据，CPU将输入数据的内存地址及个数送入相应寄存器；
2. 进程进入等待状态，进程调度程序调度其他进程；
3. 在DMA控制器控制下，将数据寄存器的数据不断写入内存；

# DMA方式

4. 传送完毕，DMA控制器向CPU发出中断请求，CPU转中断处理程序；
5. 中断处理程序结束，CPU返回被中断的进程继续执行。


# DMA方式

1. 负责在高速外围设备与内存之间成批量的数据传输工作，但是不对数据作再加工处理，I/O操作类型简单；
2. 需要使用专门的DMA控制器：控制、状态寄存器、传送字数寄存器、内存地址寄存器和数据缓冲寄存器。
3. 采用“偷窃总线控制权”，不用CPU干预
4. 每传送一个数据并不产生中断，只有本次DMA传送的数据全部完毕时，才产生中断。

# DMA方式与中断的主要区别

- 中断方式是在数据缓冲寄存区满后，发中断请求， CPU 进行中断处理，DMA方式则是在所要求传送的数据块全部传送结束时要求 CPU 进行中断处理，大大减少了 CPU进行中断处理的次数。
- 中断方式的数据传送是由 CPU 控制完成的而 DMA方式则是在DMA控制器的控制下不经过 CPU 控制完成的。

# 操作系统设计问题-设计目标

## 1. 提高使用效率：

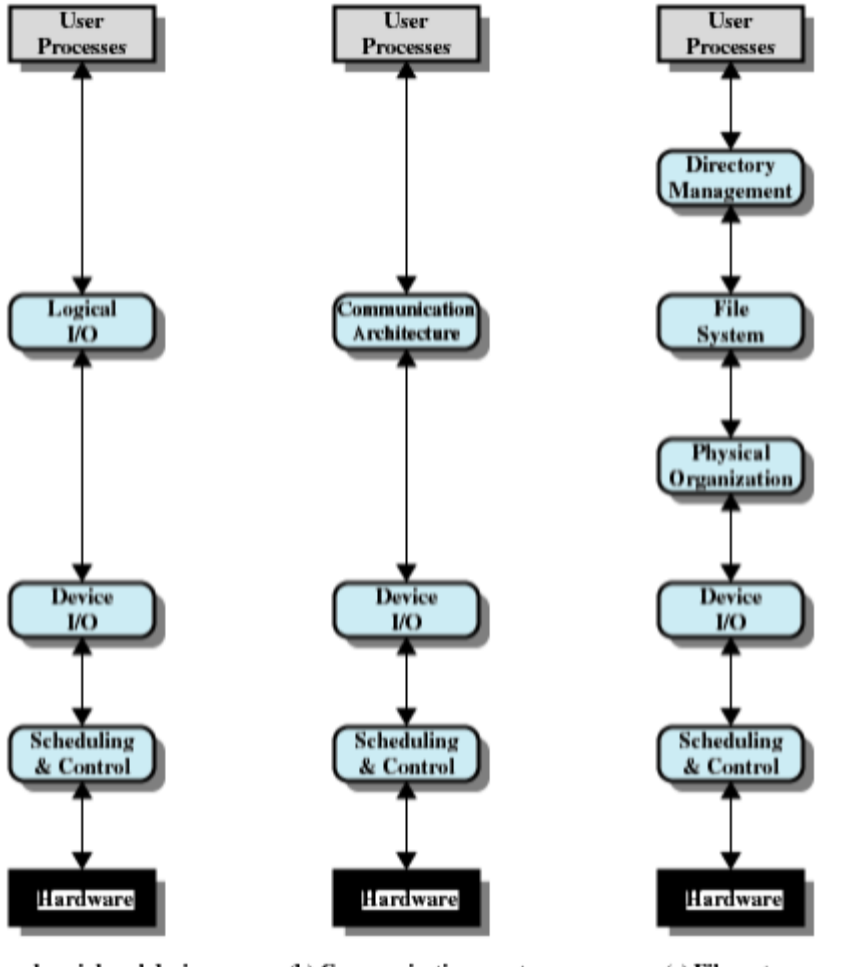
提高CPU与外设、外设与外设之间的并行工作的能力。中断和通道技术的引入提供了这种并行性。

## 2. 通用性：

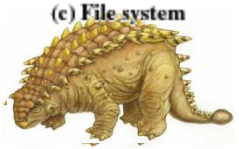
是使用户能独立于具体设备的复杂物理特性而方便的使用设备。也就是说避开各种物理设备的具体使用细节，呈现在用户面前的是简单、易操作的逻辑设备

# I/O功能逻辑结构

- **逻辑I/O：**把设备当作一个逻辑资源来处理，并不关心实际控制设备的细节。
- **设备I/O：**请求的操作和数据被转换成适当的I/O指令序列、通道命令和控制命令。
- **调度和控制：**关于I/O操作的排队、调度和控制实际发生在这一层，该层与I/O模块和设备硬件真正发生交互的软件层。




(a) Local peripheral device (b) Communications port (c) File system



# I/O缓冲

为解决计算机外设的和中央处理器之间速度不匹配，所产生的“瓶颈”现象，提高系统性能，因此引入了缓冲技术。

- 改善中央处理器与外围设备之间速度不配的矛盾，
- 提高CPU和I/O设备的并行性。

- **面向块的I/O设备；**
- **面向流的I/O设备；**

# I/O缓冲-缓冲区管理

## 单缓冲

- 当用户进程发出I/O请求时，操作系统在主存的系统空间为该操作分配一个缓冲区，可以实现预读和滞后写。

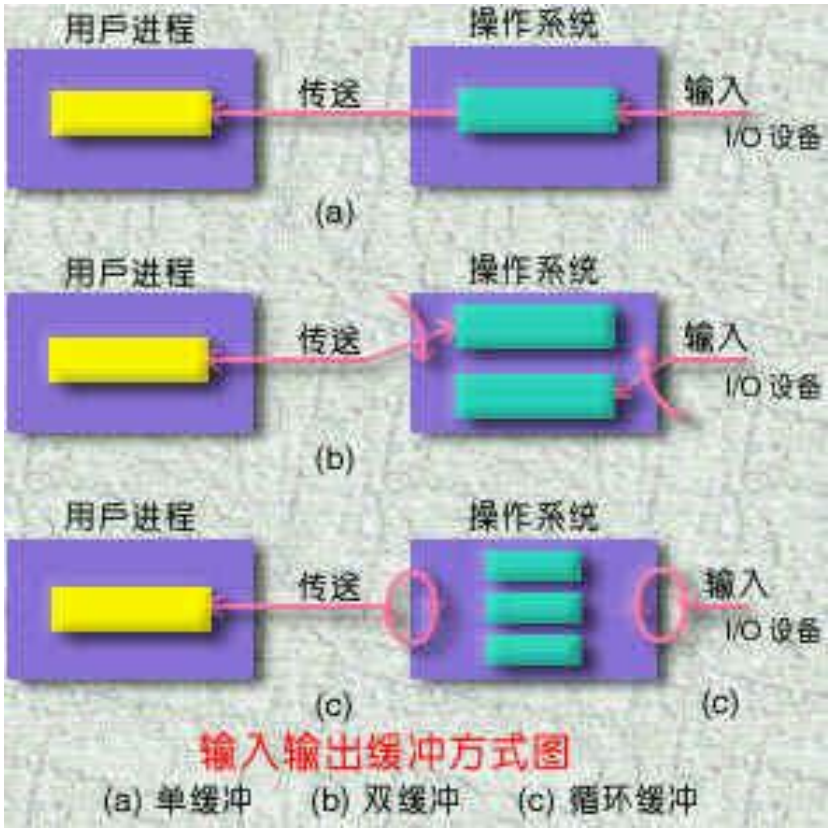
## 双缓冲

- 可以实现用户数据区—缓冲区之间交换数据和缓冲区—外设之间交换数据的并行。

## 循环缓冲

- 多个缓冲区连接起来统一管理

# I/O缓冲-缓冲区管理




生产者/消费者模型
面向块：时间为MAX[C，T]+M

生产者/消费者模型
面向块：时间为MAX[C，T]

有界缓冲区
生产者/消费者模型