Chapter 13 I/O Systems

Contents

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

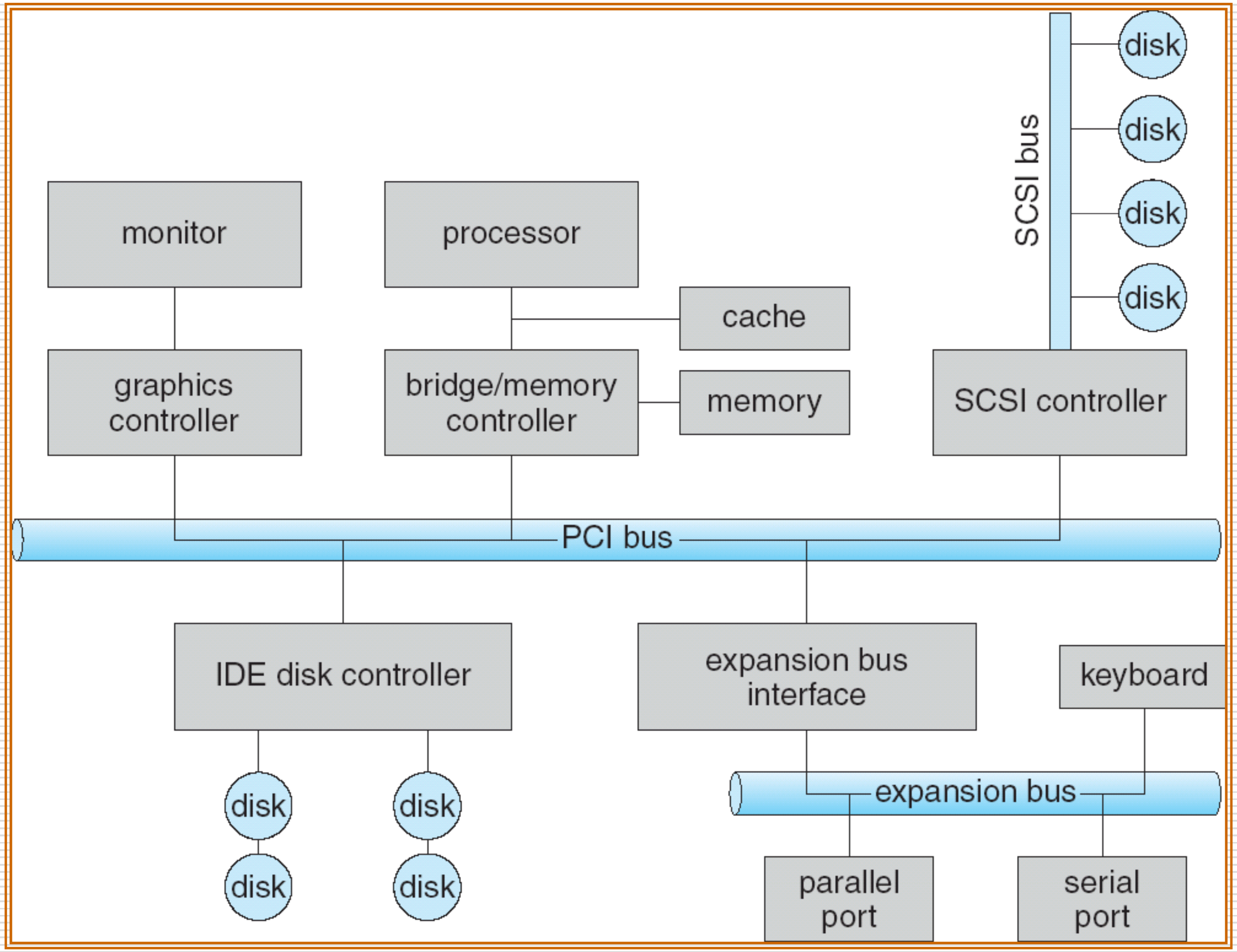
- Incredible variety of I/O devices
  - Function
  - speed
- Device driver
  - Present a uniform device-access interface to the I/O subsystem, much as systems calls

I/O Hardware

- 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
  - **Memory-mapped** I/O

A Typical PC Bus Structure

- Peripheral component interconnection



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)

I/O port

- PCs use I/O instructions to control some devices.
- An I/O port typically consists of four registers
  - The data-in register is read by the host to get input
  - The data-out register is written by the host to send output
  - The status register contains bits that be read by the host.
  - The control register can be written by the host to start a command or to change the mode of a device.

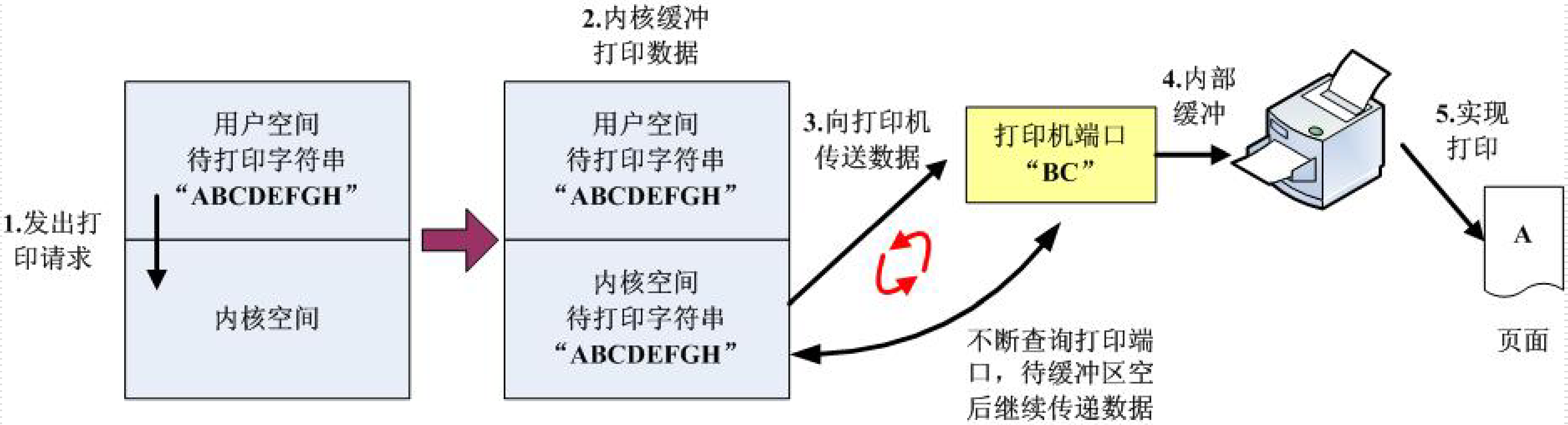
Polling

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

Polling

- 1. The host repeatedly reads the *busy* bit until that bit becomes clear.
- 2. The host sets the *write* bit in the *command* register and writes a byte into the *data-out* register.
- 3. The host sets the *command-ready* bit.
- 4. When the controller notices that the *command-ready* bit is set, it sets the *busy* bit.
- 5. The controller reads the command register and sees the write command. It reads the *data-out* register to get the byte, and does the I/O to the device.
- 6. The controller clears the *command-ready* bit, clears the *error* bit in the status register to indicate that the device I/O succeeded, and clears the *busy* bit to indicate that it is finished

Example



- 由操作系统的“服务程序”负责将用户数据传送至打印机端口
- 服务程序顺序传送打印数据，填满接口缓冲区后就等待（空循环）
- 每次循环中都检查接口缓冲区是否可用，一旦可用就继续传送数据
- 数据传送完成后“服务程序”结束，用户进程继续运行
- 缺点：靠CPU以“忙等待”的形式与打印机进行通信，浪费CPU资源

Interrupts

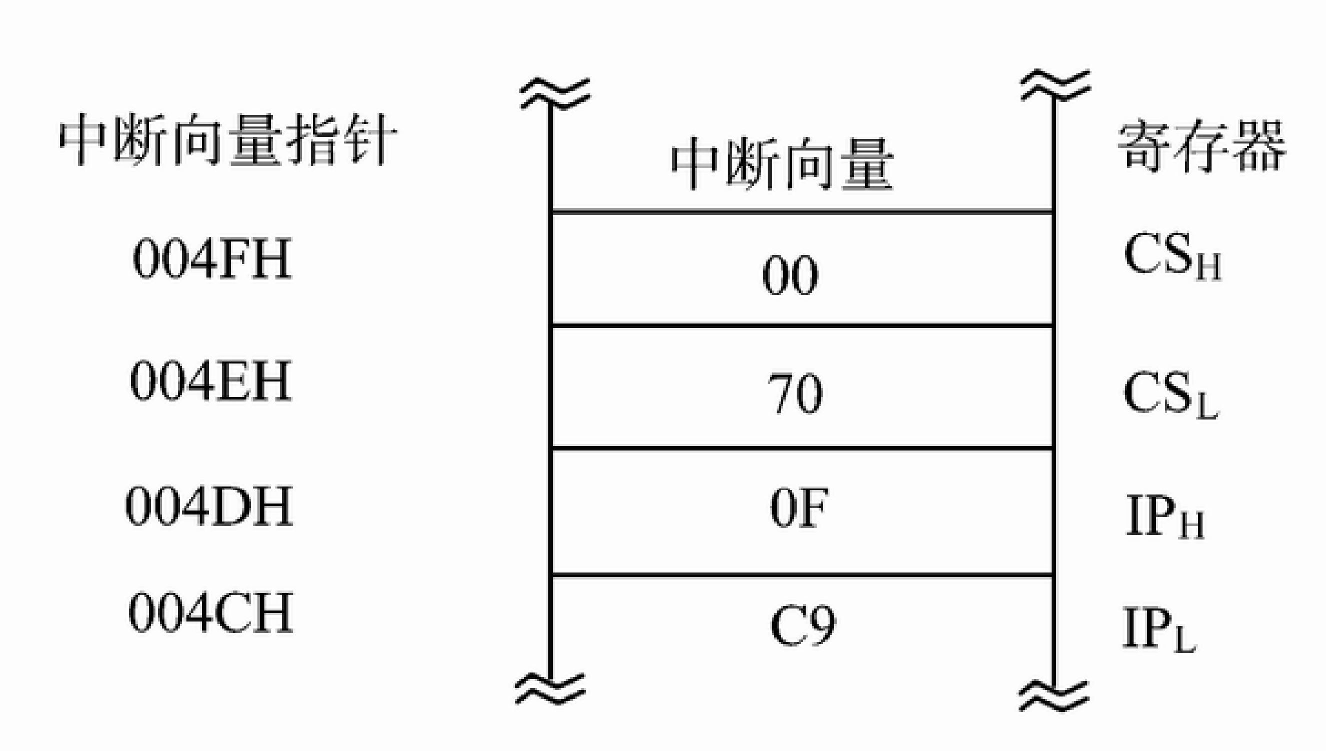
- CPU **Interrupt-request line** triggered by I/O device
- **Interrupt handler** receives interrupts
- **We need more sophisticated interrupt-handling features**
  - we need ability to defer interrupt handling during critical processing
  - we need an efficient way to dispatch to the proper interrupt handler for a device without first polling all the devices to see which one raised the interrupt.
  - we need multilevel interrupts, so that the OS can distinguish between high-and low- priority interrupts and can respond with the appropriate degree of urgency.
- **Two interrupt request lines:**
  - **Nonmaskable interrupt**
  - **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

Interrupts

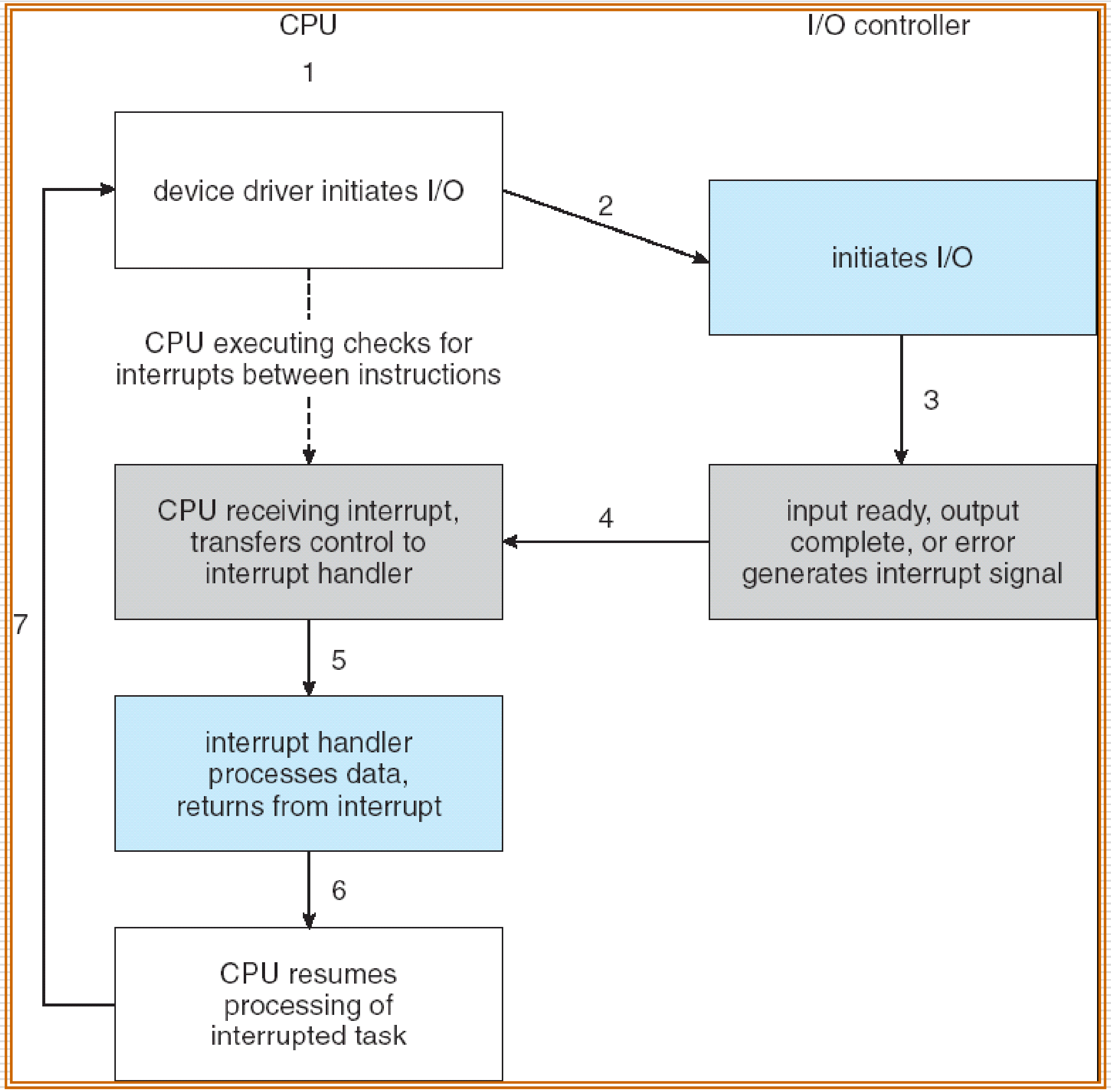
- 中断向量表
  - 每一个中断服务程序都有一个唯一确定的入口地址，我们把系统中所有的中断向量集中起来放到存储器的某一区域内，这个存放中断向量的存储区就叫中断向量表，换言之，每一个中断服务程序与该表内的一个中断向量建立一一对应的关系
- 将中断类形号N×4，得到中断向量的第一个字节（即存放IP的低8位）的指针，即向量地址=0000：N×4。

Interrupts

- 例如，软盘"INT 13H"，它的中断向量为"0070H（CS）：0FC9H（IP）"，当处理中断时，CPU根据类型号（13H）乘4后得到中断向量的第一个字节的指针，即：13H×4=004CH。从它开始连续4个字节单元中用来存放"INT 13H"的中断向量（即入口地址）



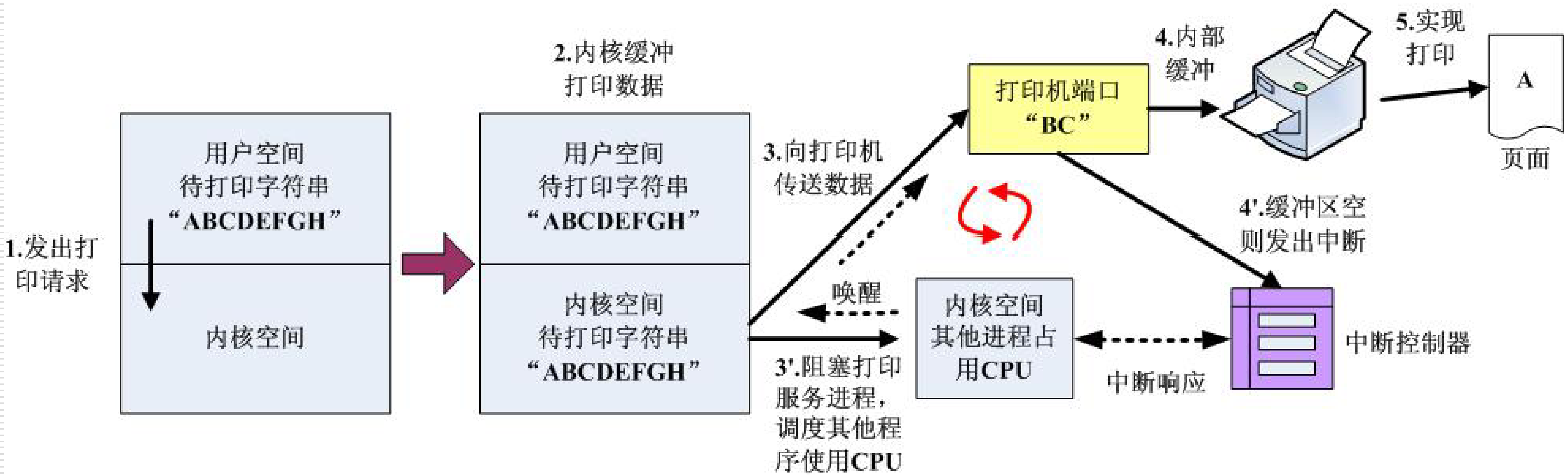
Interrupt-Driven I/O Cycle



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

Example

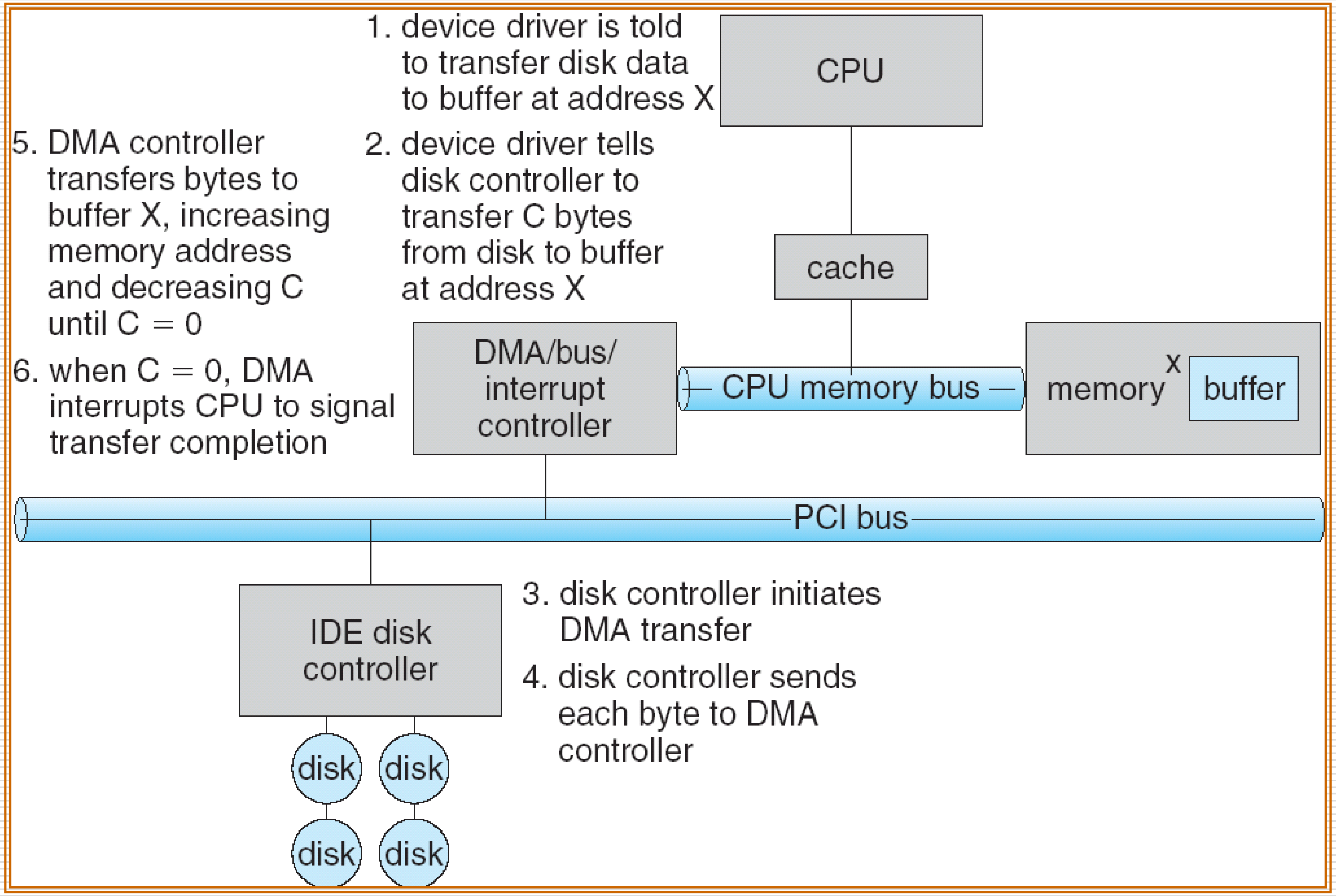


- “打印服务程序”只将最开始的数据传送至打印机端口，然后阻塞
- CPU可继续调度其他进程运行，不浪费CPU时间
- 一旦打印缓冲区空后，打印机端口发出硬件中断
- CPU响应中断，恢复“打印服务程序”运行，继续传送数据
- 缺点：虽然节省了CPU资源，但是中断响应也消耗较大的系统资源

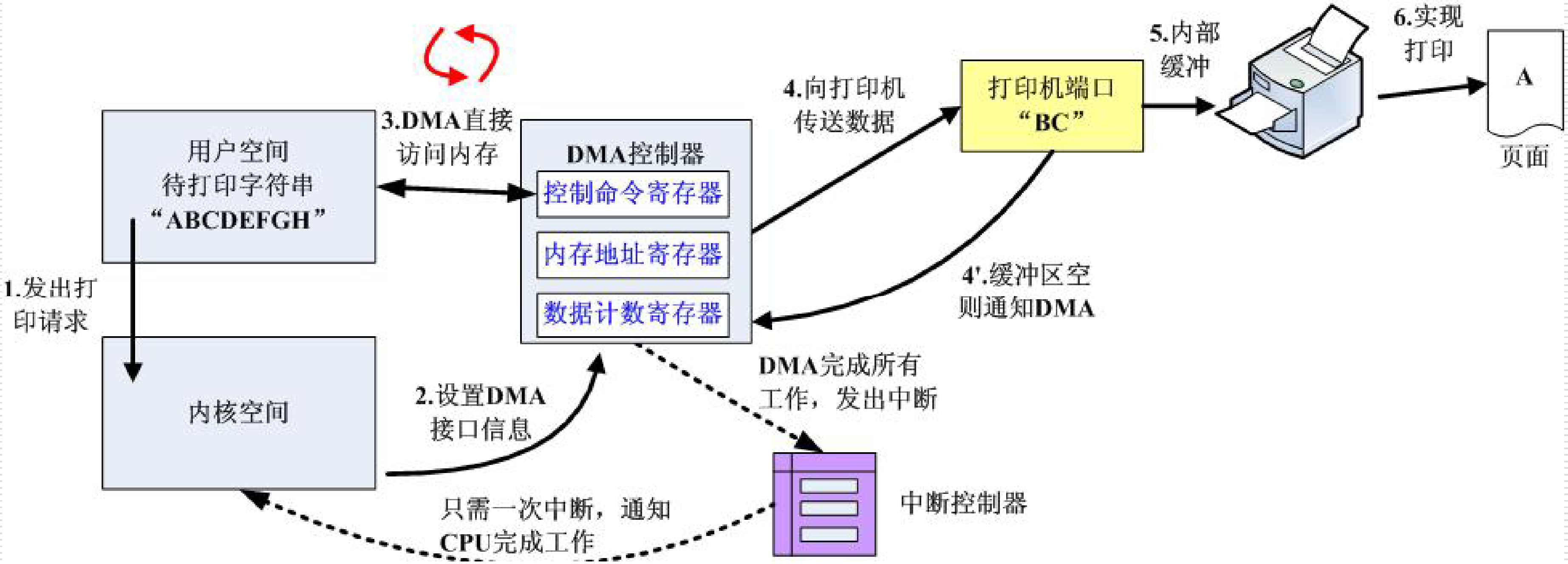
Direct Memory Access

- 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



Example

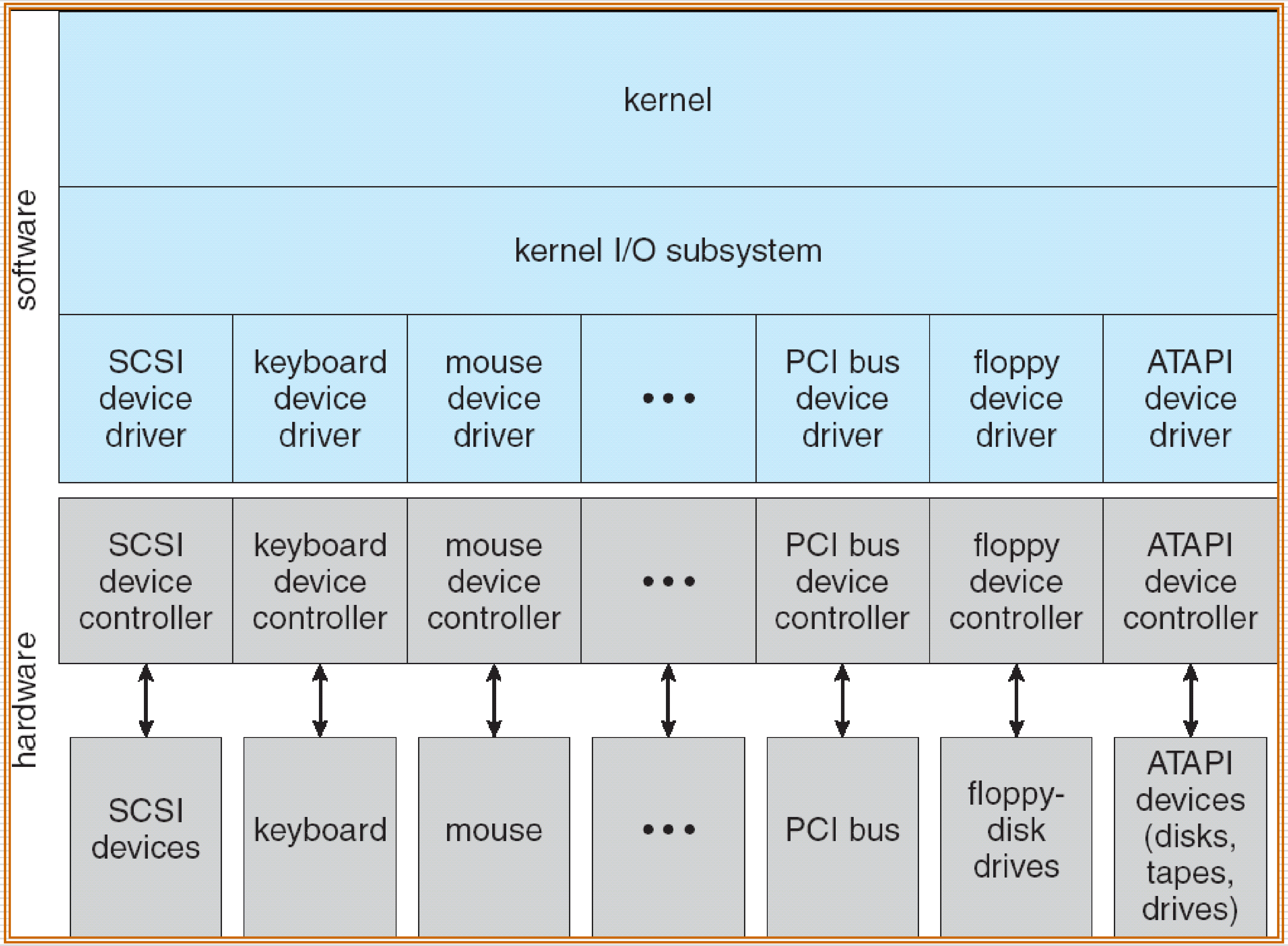


- 用户进程发出系统调用后进入阻塞态，CPU直接设置DMA端口
- CPU与DMA并行工作，DMA负责将用户数据传送给打印机
- 当DMA完成所有工作后，向CPU发出中断，CPU响应后唤醒用户进程
- 优点：只有一次中断、DMA与CPU并行提高了系统运行效率
- 缺点：DMA速度较慢，如果CPU并不繁忙，那么DMA机制并无太大意义

13.3 Application I/O Interface

- I/O system calls encapsulate device behaviors in generic classes
- 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**
- Device-driver layer hides differences among I/O controllers from kernel

A Kernel I/O Structure



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

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)

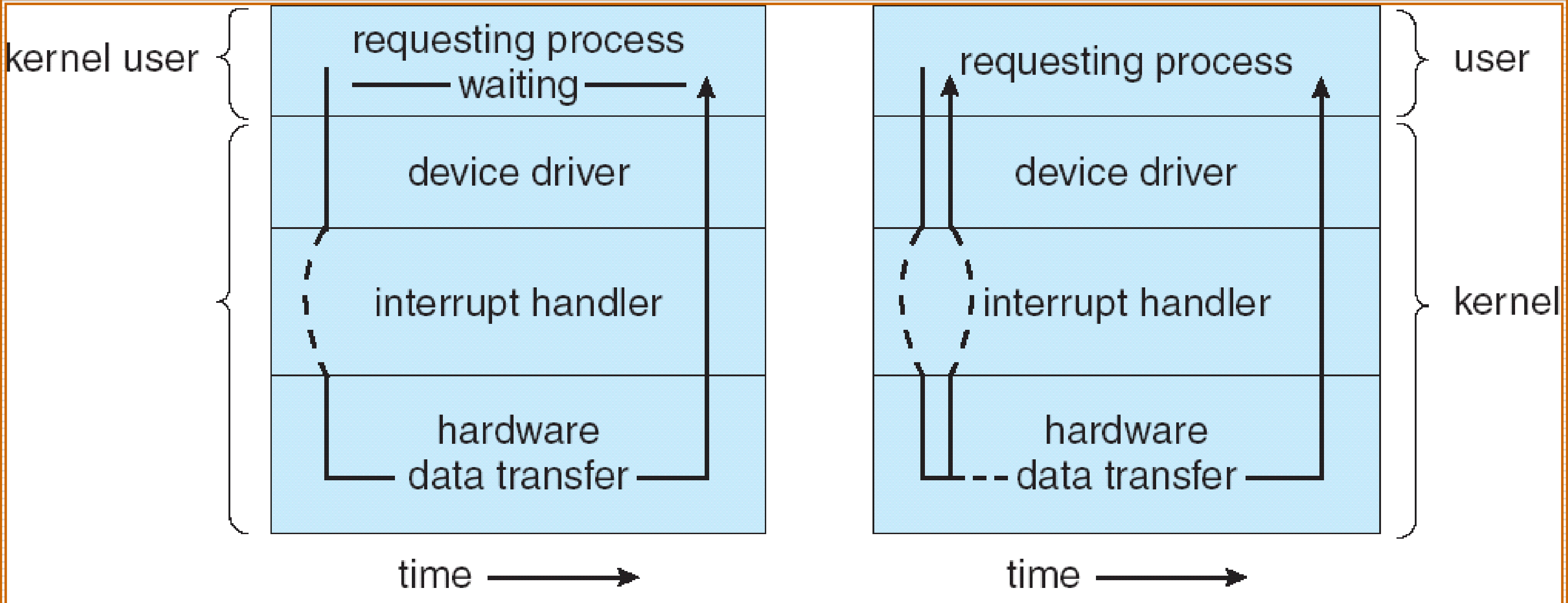
Clocks and Timers

- 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

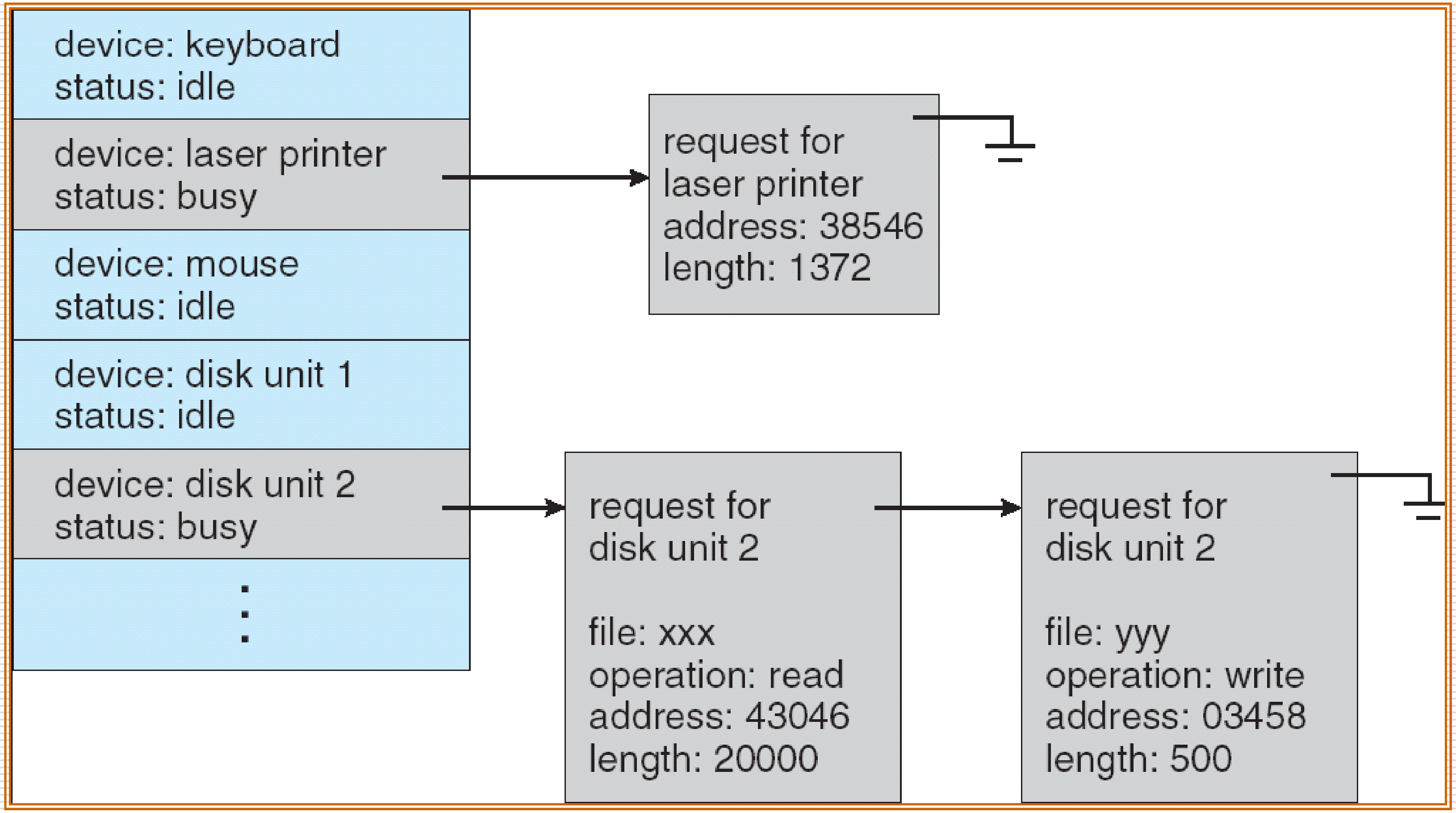


(a) (b)

13.4 Kernel I/O Subsystem

- Kernels provide many services related to I/O:
  - scheduling,
  - buffering,
  - caching,
  - spooling,
  - device reservation,
  - and error handling.
- I/O Scheduling
  - To schedule a set of I/O requests means to determine a good order in which to execute. Is it important?
- Implementation
  - Some I/O request ordering via per-device queue
  - Some OSs try fairness

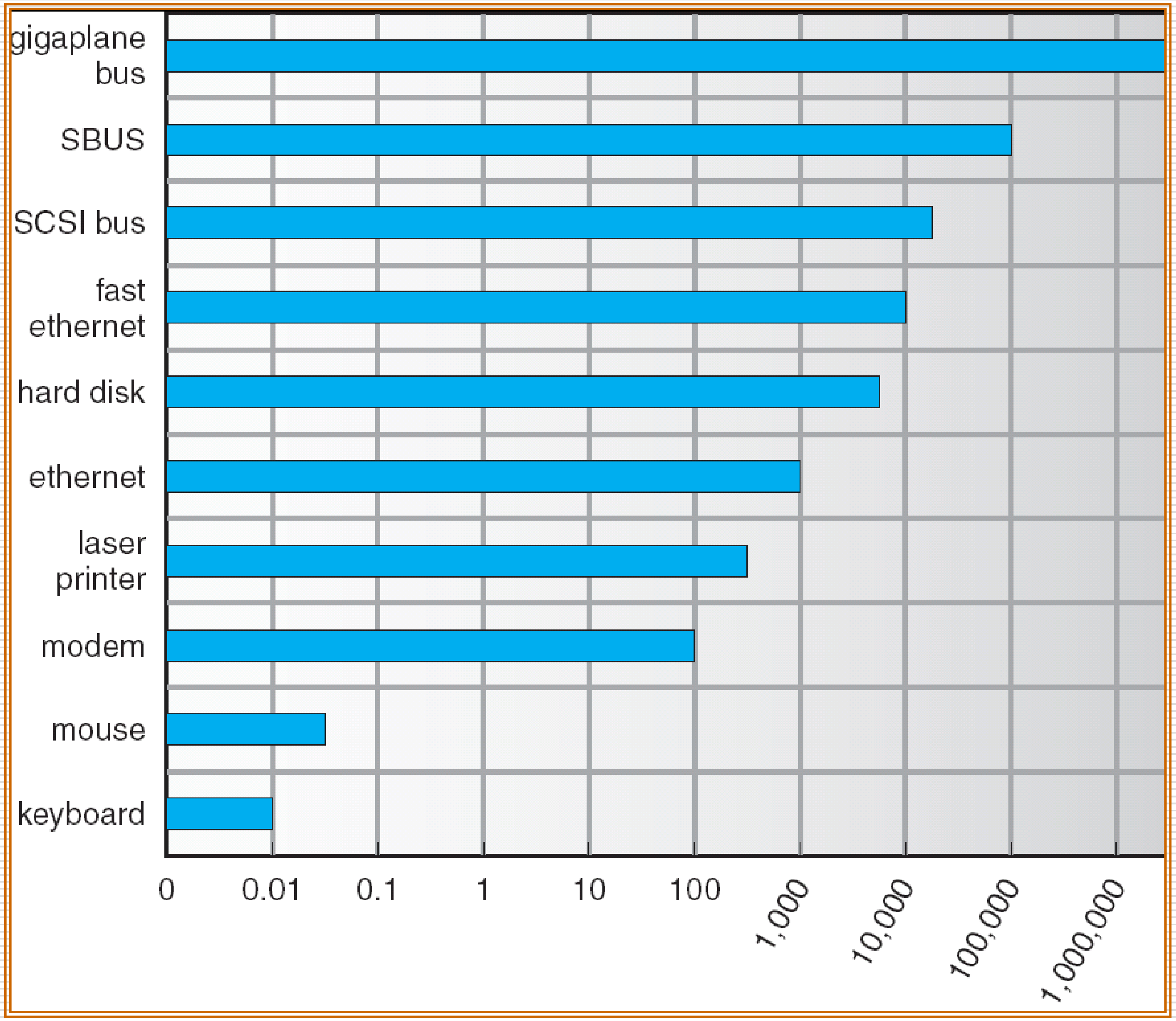
Device-status Table



Buffering(缓冲)

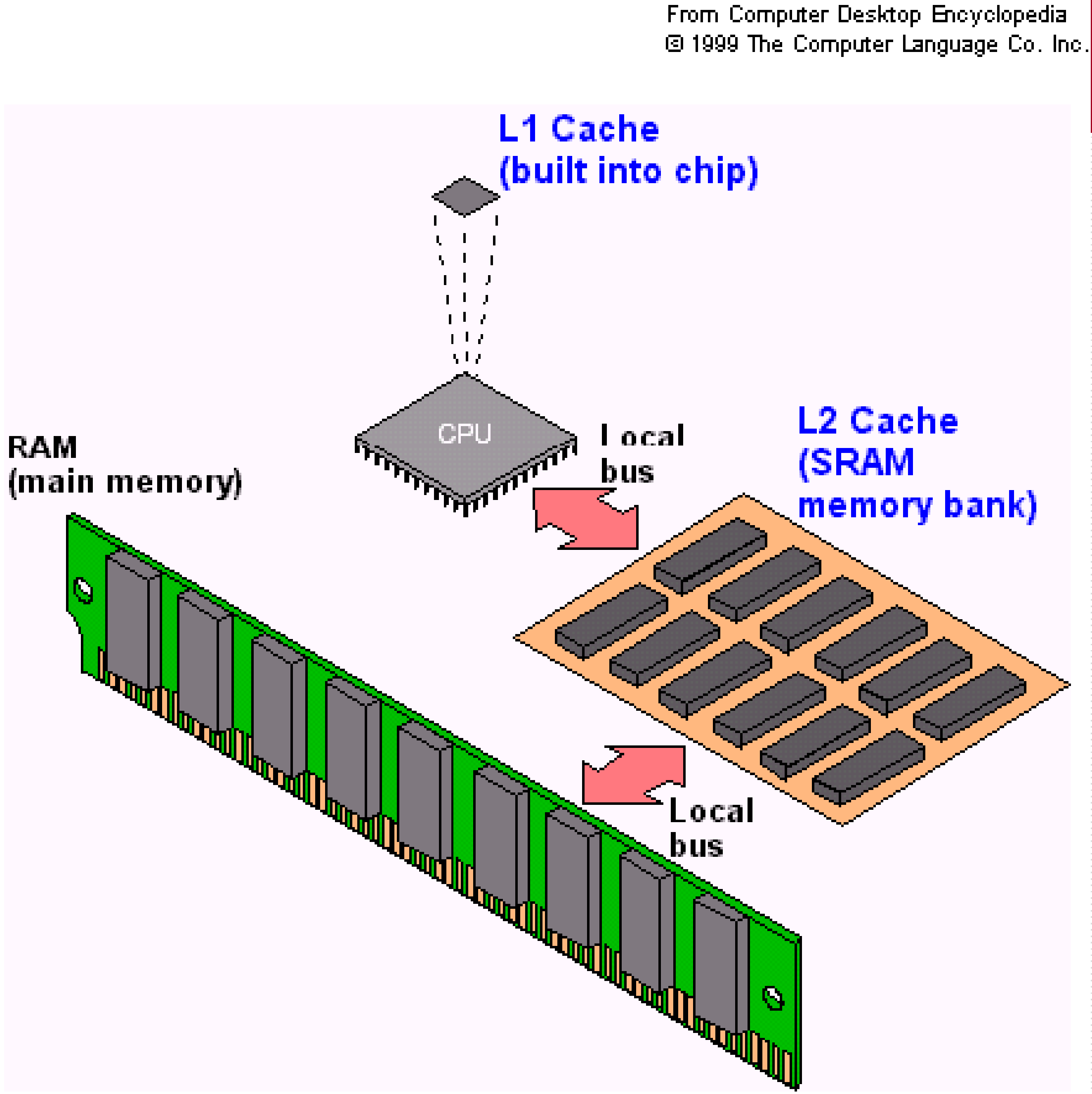
- A buffer is a memory area that stores data while they are transferred between two devices or between a device and an application.
- Buffering - store data in memory while transferring between devices
  - To cope with device speed mismatch
  - To cope with device transfer size mismatch
  - To maintain “copy semantics”

Sun Enterprise 6000 Device-Transfer Rates



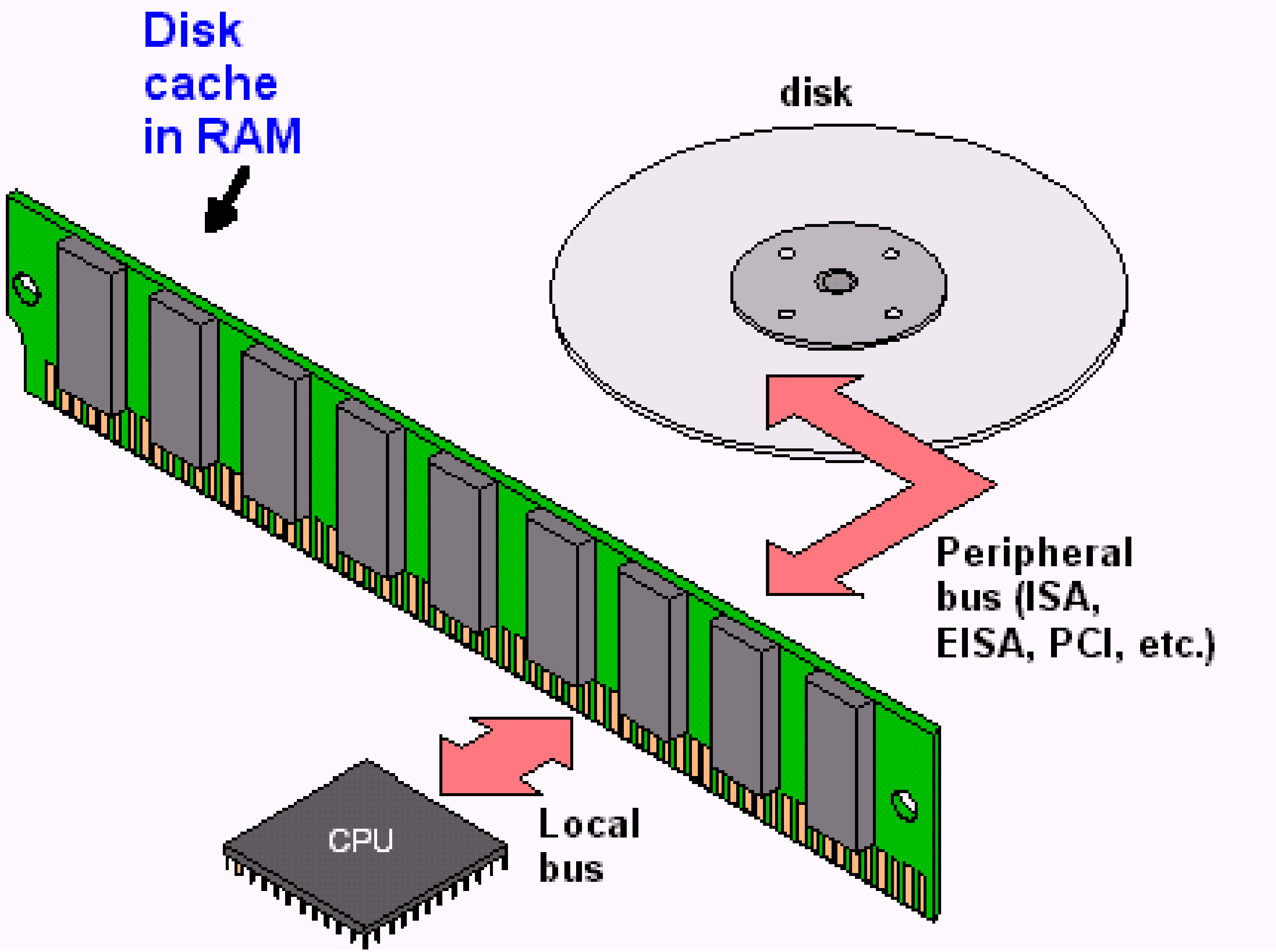
Caching (高速缓存)

- **Caching** - fast memory holding copy of data
- Reasons to use it
  - Speed
  - Data size
- Differences between caching and buffering
  - A buffer may hold the only existing copy of a data item.
  - A cache is just holds a copy on faster storage of an item.



From Computer Desktop Encyclopedia
© 1999 The Computer Language Co. Inc.
L1 Cache
(built into chip)
CPU
Local
bus
L2 Cache
(SRAM
memory bank)
RAM
(main memory)
Local
bus

From Computer Desktop Encyclopedia



Spooling

- **Spooling** - hold output for a device
  - Simultaneous peripheral Operations On-Line
  - If device can serve only one request at a time
  - Its speed is very slow
  - i.e., Printing

Device reservation

- **Device reservation** - provides exclusive access to a device
  - System calls for allocation and deallocation
  - Watch out for deadlock

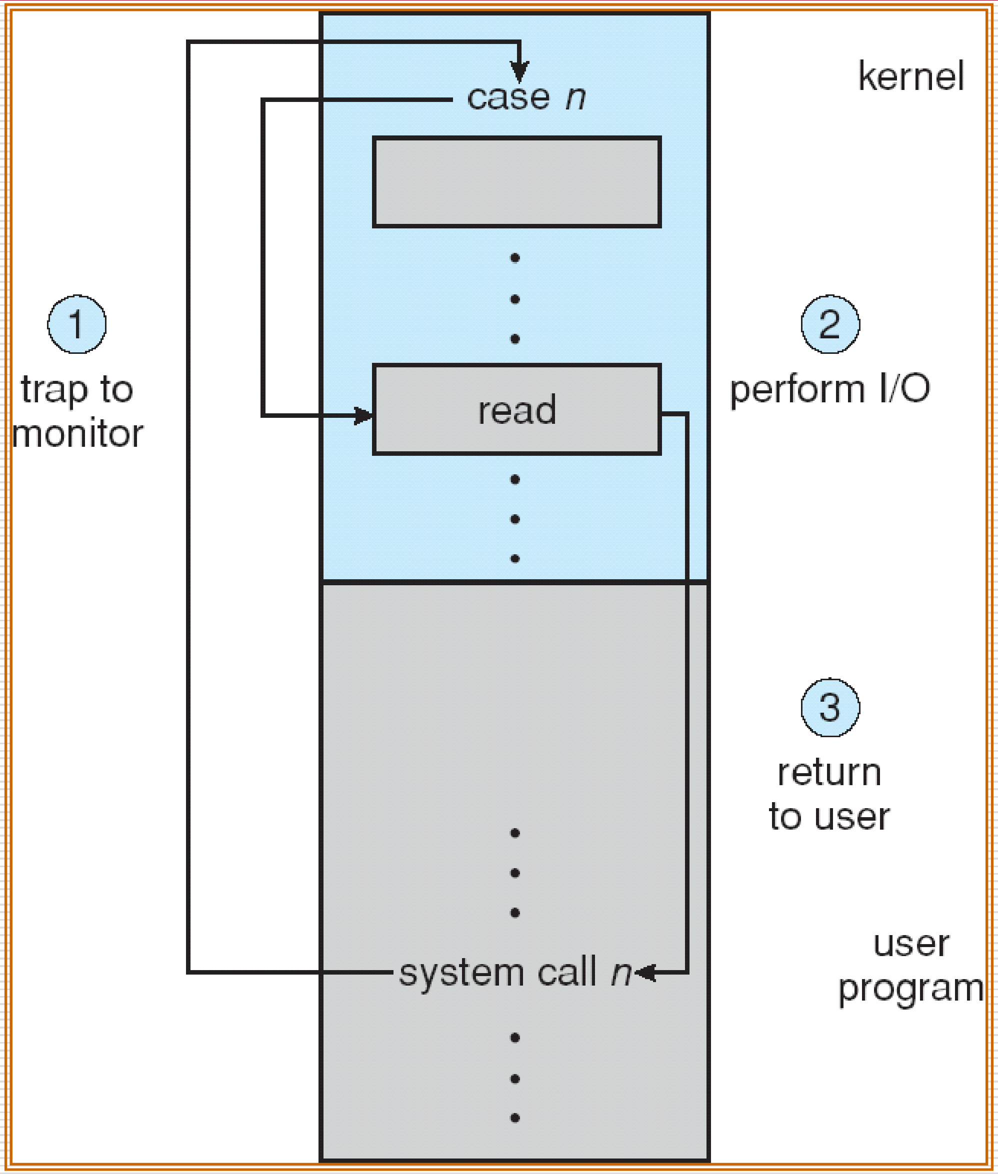
Error Handling

- 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

I/O Protection

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

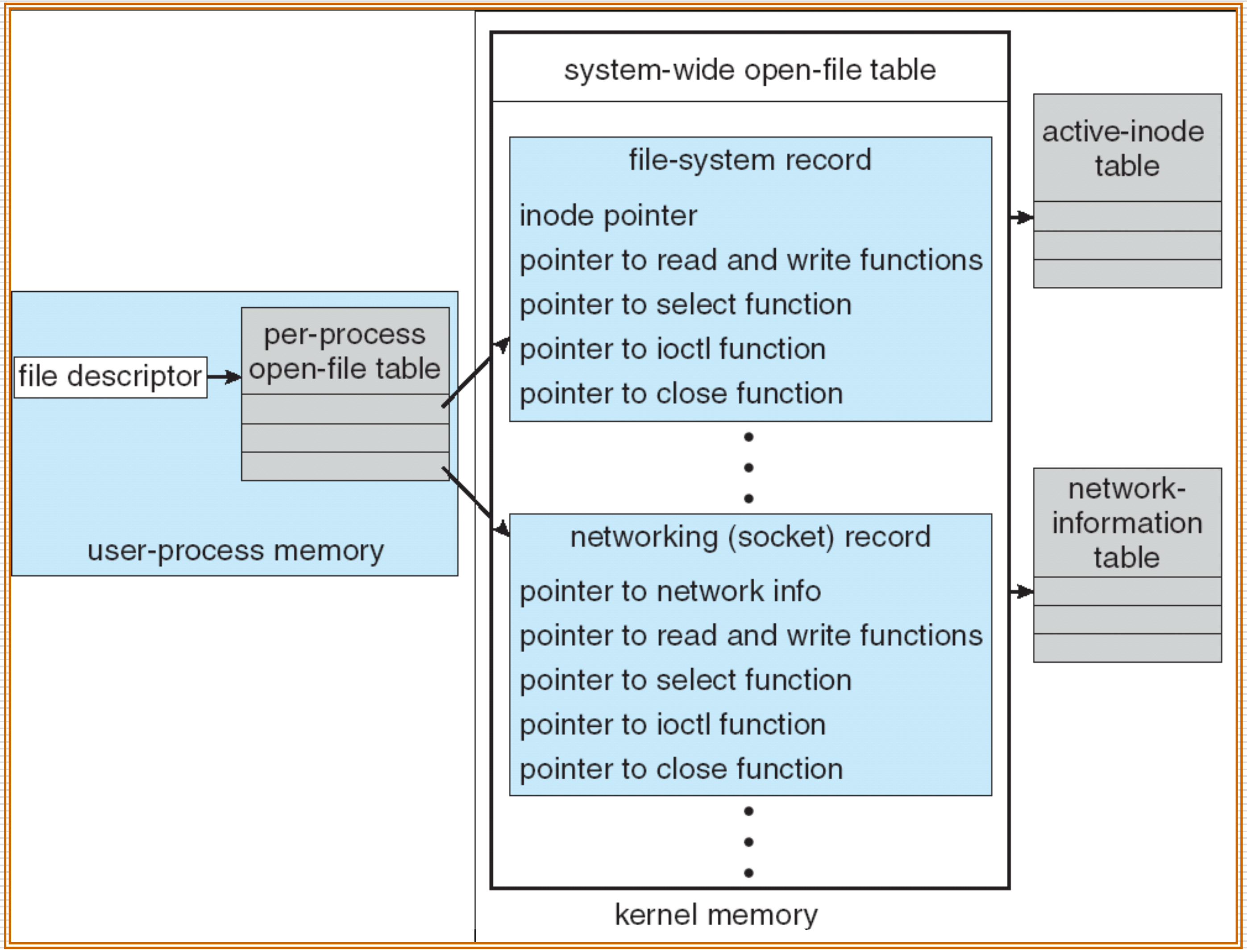
Use of a System Call to Perform I/O



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

UNIX I/O Kernel Structure



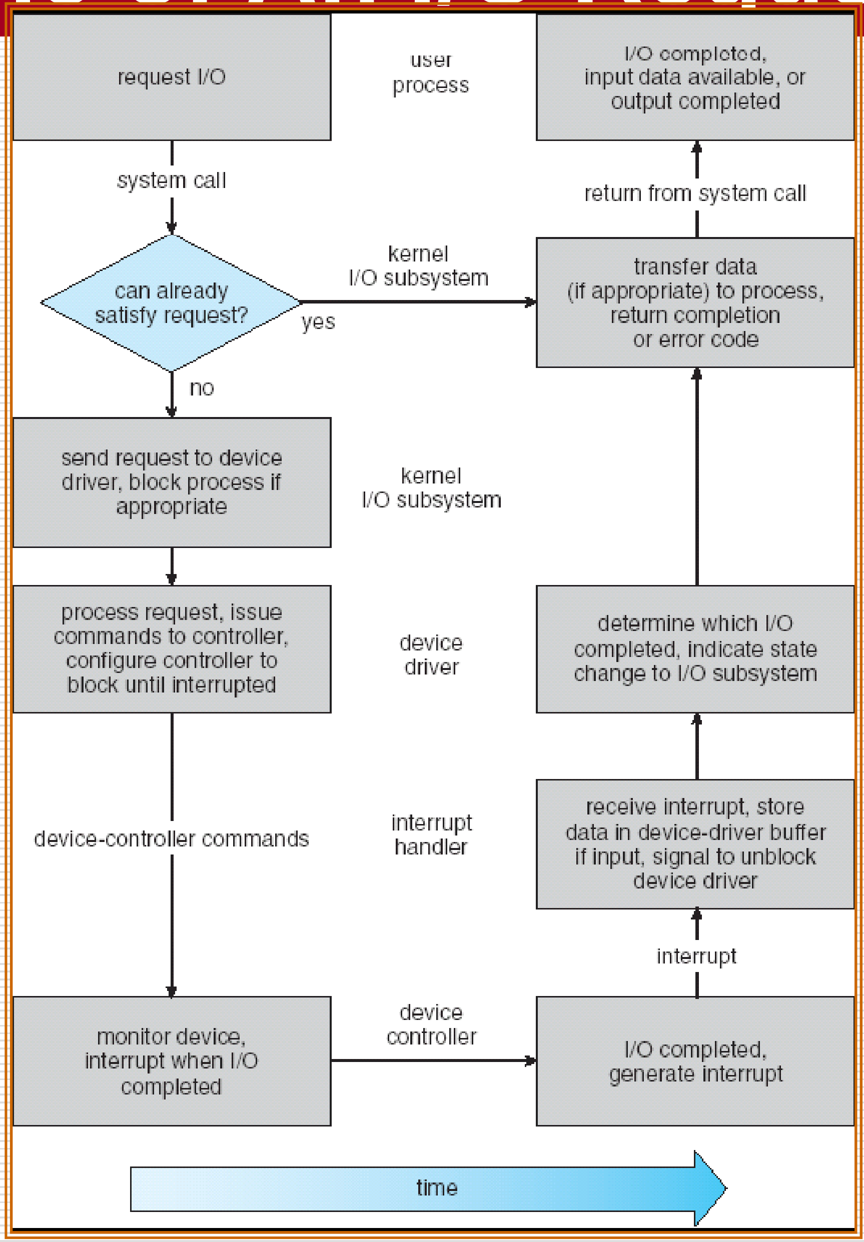
Kernel I/O subsystem Summary

- The I/O subsystem supervises the following procedures:
  - Management of the name space for files and devices
  - Access control to files and devices
  - Operation control
  - File-system space allocation
  - Devices allocation
  - I/O scheduling
  - Device-status monitoring, error handling, and failure recovery
  - Device-driver configuration and initialization

13.5 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



STREAMS

- **STREAM** – a full-duplex communication channel between a user-level process and a device in Unix System V and beyond

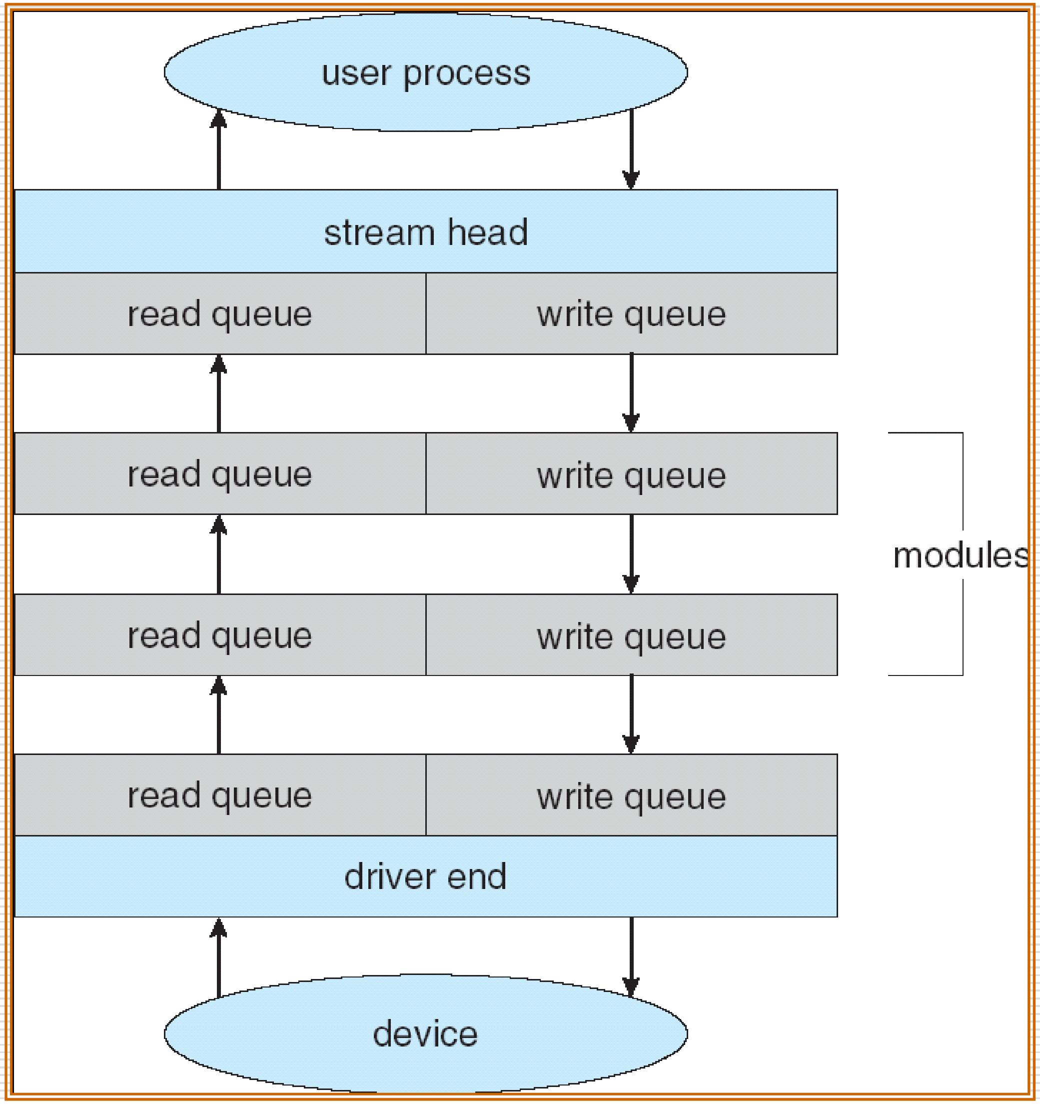
- A STREAM consists of:

  - STREAM head interfaces with the user process
  - driver end interfaces with the device
  - zero or more STREAM modules between them.

- Each module contains a **read queue** and a **write queue**

- Message passing is used to communicate between queues

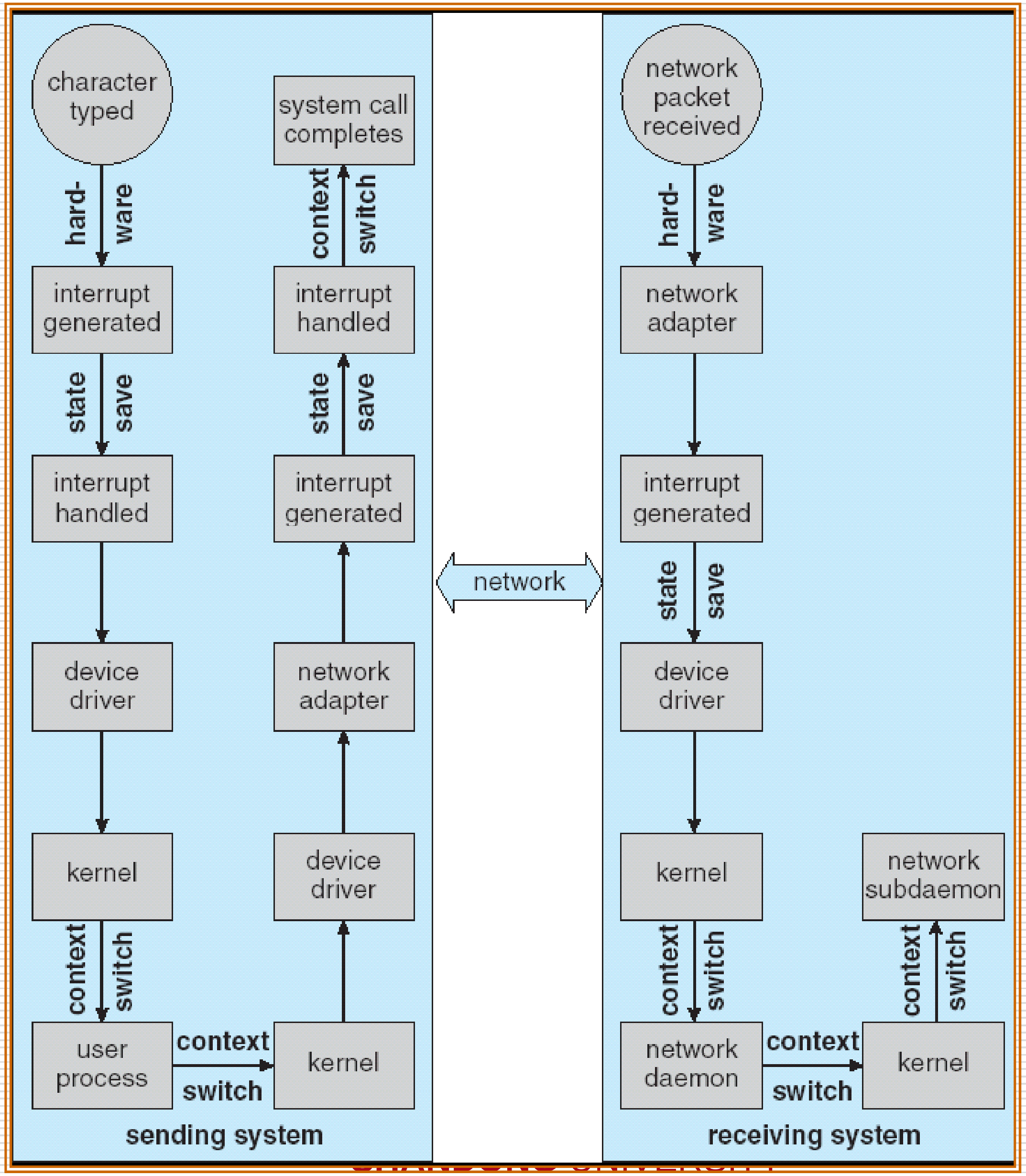
The STREAMS Structure



13.7 Performance

- I/O a major factor in system performance:
  - Demands CPU to execute device driver, kernel I/O code
  - Context switches due to interrupts
  - Data copying
  - Network traffic especially stressful

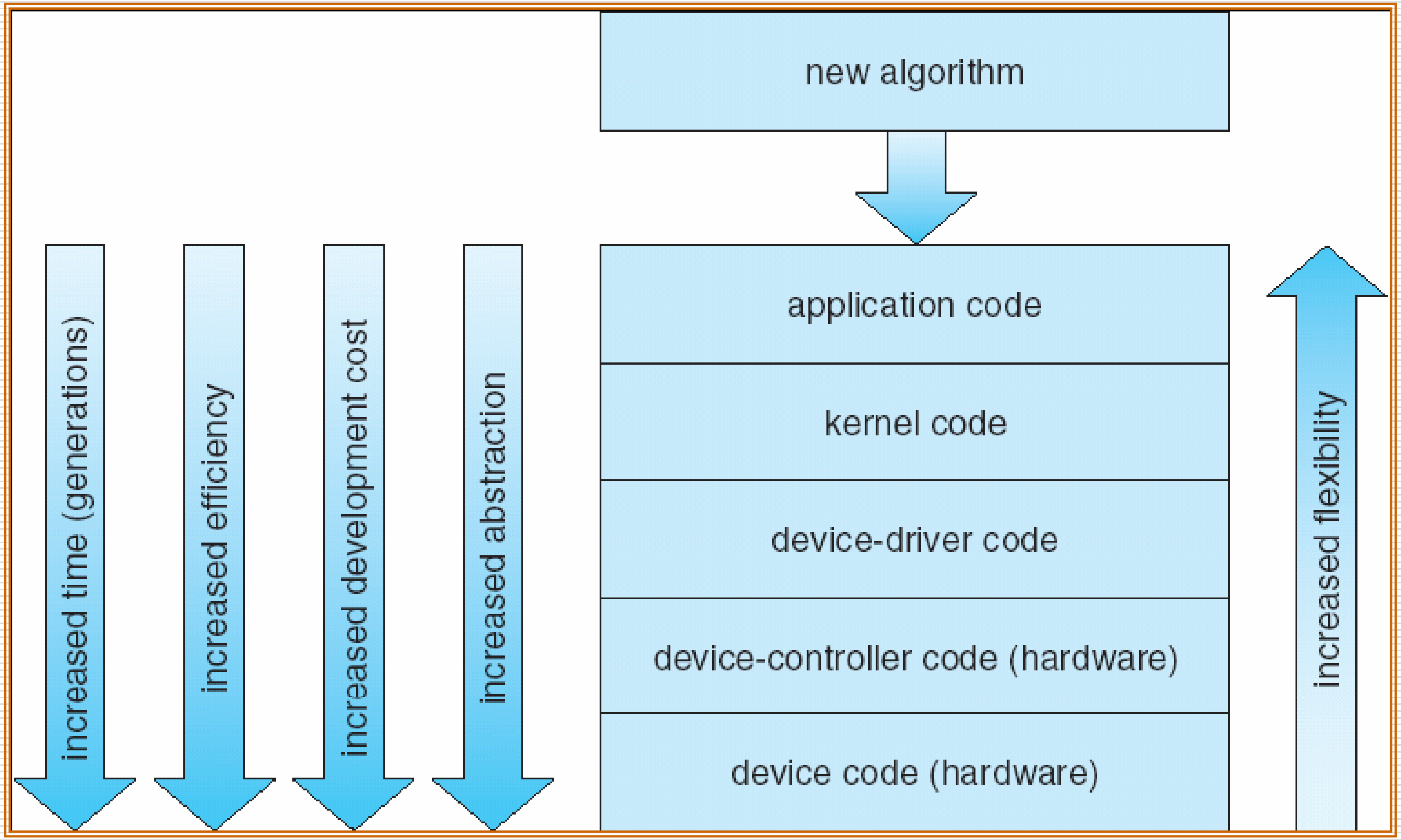
Intercomputer Communications



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

Device-Functionality Progression



Assignment

- 3

End of Chapter 13

Any Question?