操作系统

宫晓利、蒲凌君、张久武

gongxiaoli@nankai.edu.cn、pulingjun@nankai.edu.cn、
zhangjiuwu@nankai.edu.cn

南开大学 计算机学院 网络空间安全学院

章节7：设备管理

设备有什么特点？ 外设

- 以鼠标、键盘、显示器为例
- 速度慢 操作外设是特权态的事情
- 不易共享
- 每类设备都有自己的工作逻辑 **工作原理不同**
- 同一设备，由不同厂商生产，也会有一定的差别

- 应用程序需要为每一款设备调整吗？
- 应用程序需要知道其他人在使用设备吗？

示例：电阻式触摸屏工作原理

硬件：设置适配器
软件：驱动器

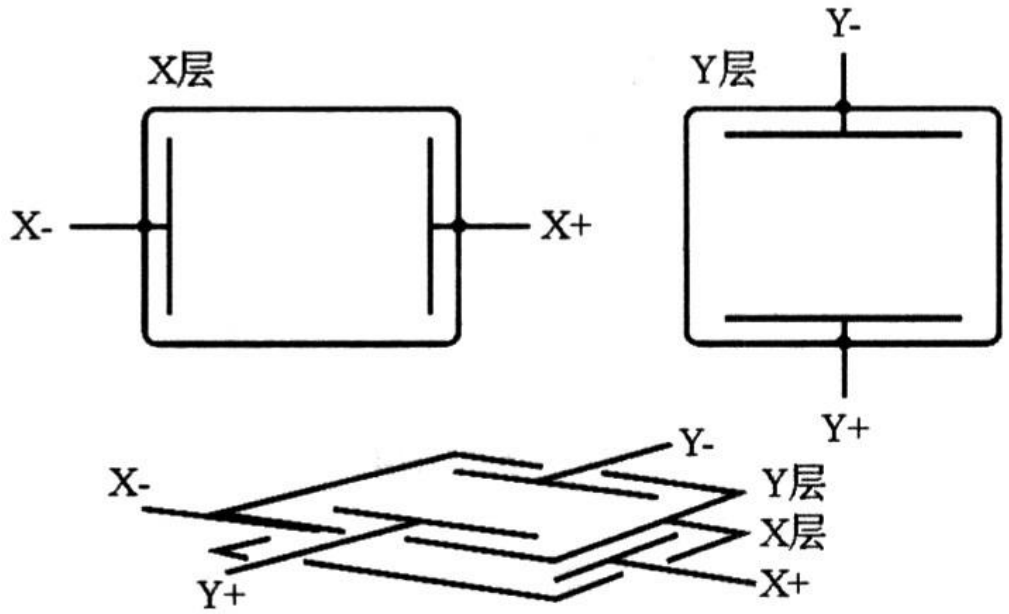
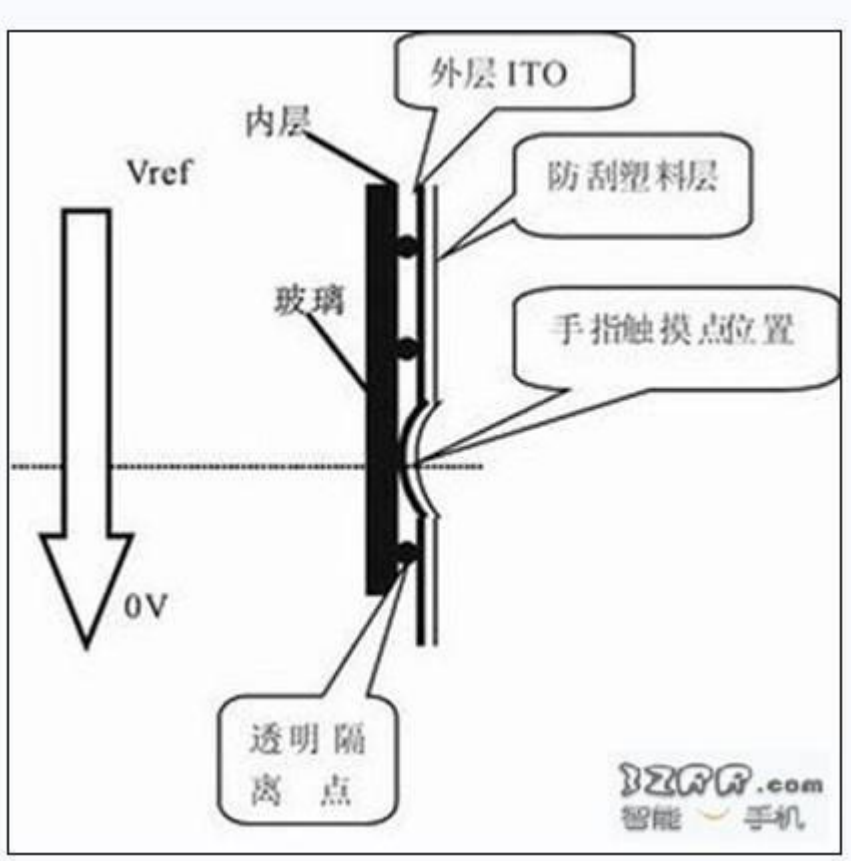
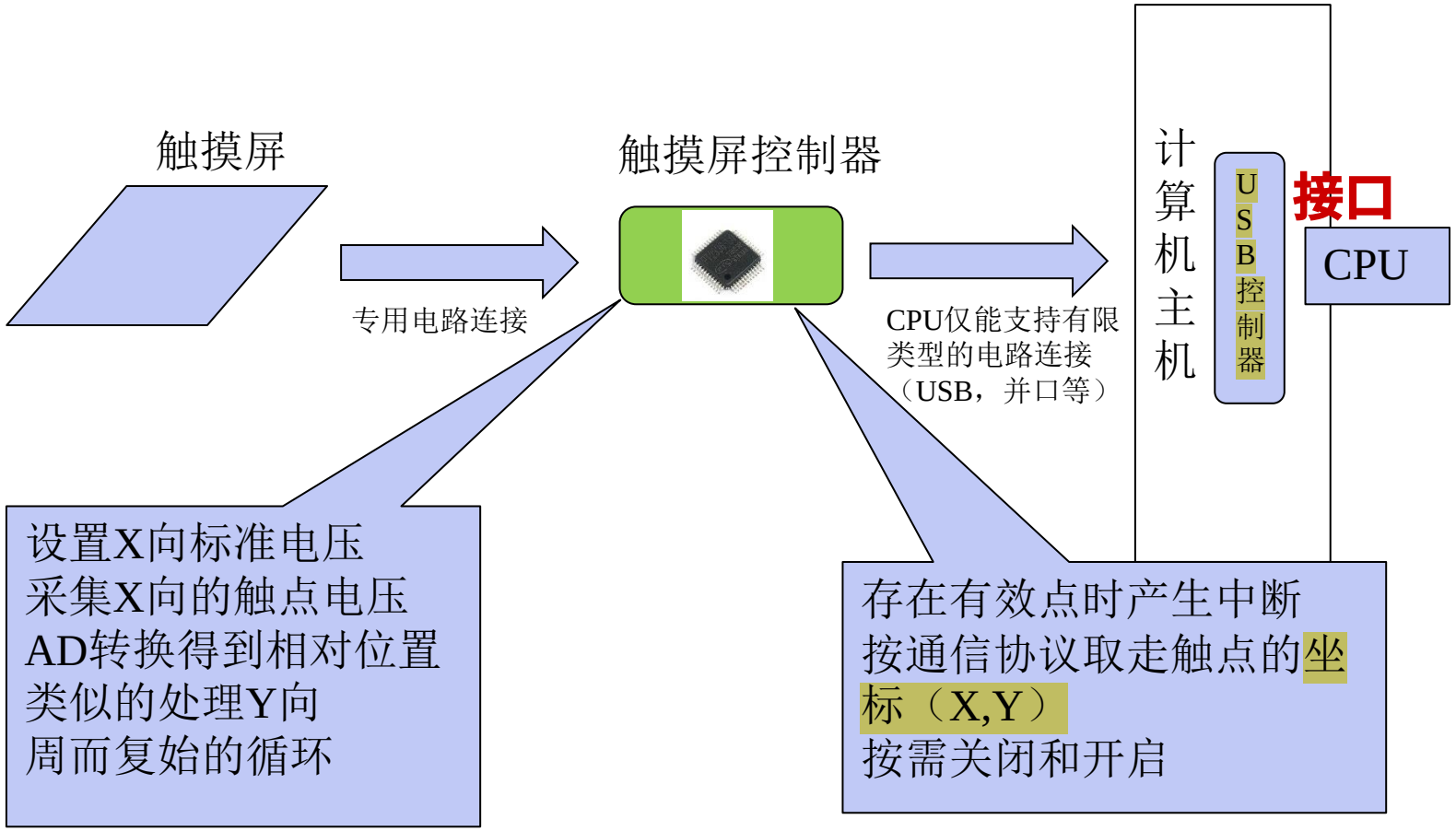


图1　四线制电阻式触摸屏原理图

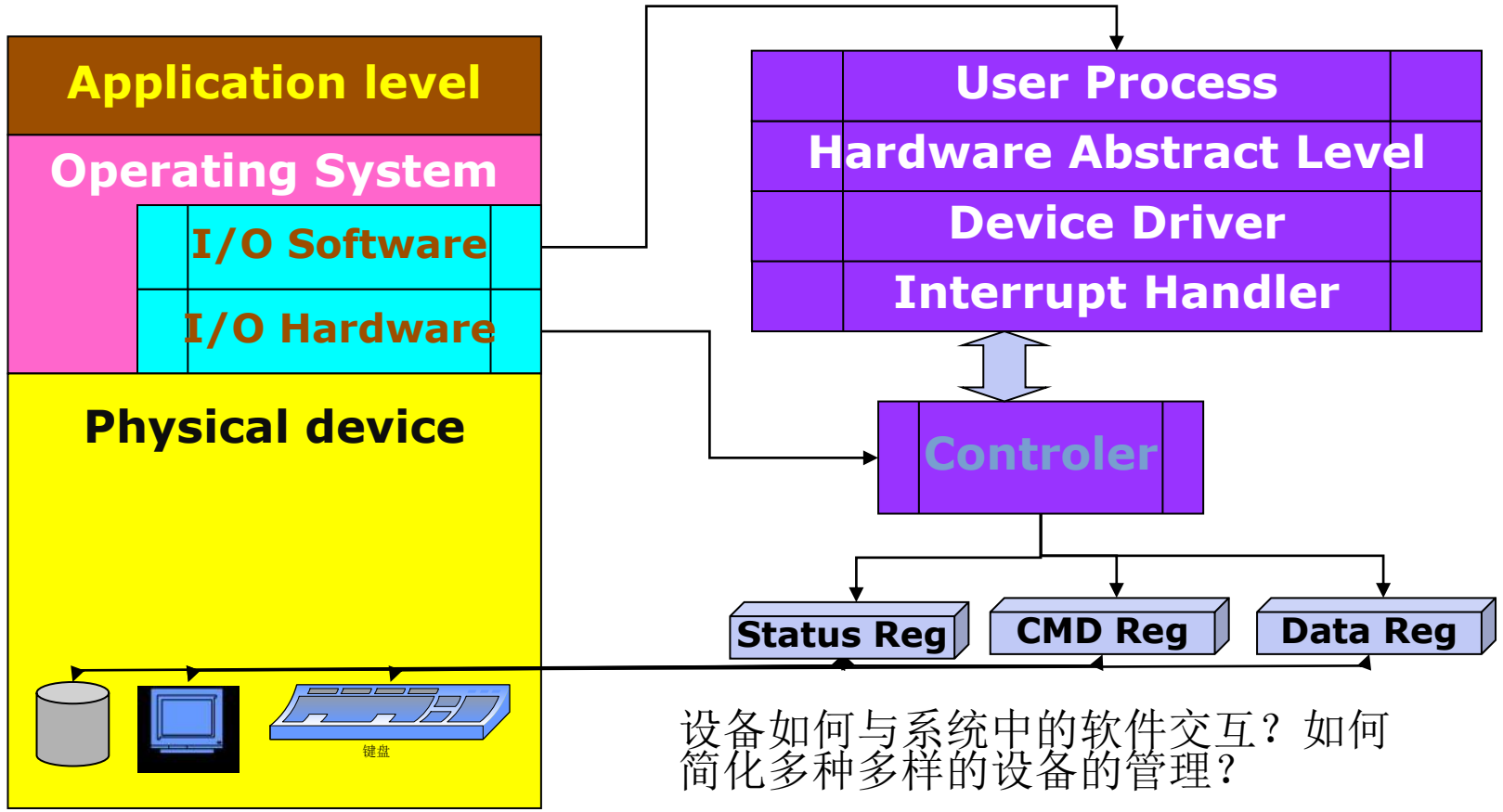


示例：触摸屏与计算机连接示意



Architecture of Device Management System

进程如何简单易用的操作设备？如何在多个进程之间共享设备？



设备如何与系统中的软件交互？如何简化多种多样的设备的管理？

分类和抽象：Categories of Devices

- Classified by Function
  - Storage device: Temporarily or permanently
  - I/O device: Human-Computer interactive
  - Communication device: Data exchange and transfer
- Classified by Data management method
  - Block stream: using block in different size as the basic unit for data management
  - Byte stream: using byte as the basic unit for DM
- Classified by Device assignment
  - Monopolization device: Low-speed I/O device
  - Sharing device: high-speed I/O device
  - Virtual device: simulating hardware by software
- Classified by working mode
  - Logical device: data structure maintained by OS
  - Physical device: different kinds of hardware

Discussion of Devices Management

- Complexity of Device
  - Different devices have different working mode
  - Different devices use different data format
  - Different devices support different interfaces
- Important issues about devices
  - Speed: the bottleneck of computer
  - HCI: operation mistake caused by user
  - Compatibility: device-independent & OS-independent
- Key strategy of device management
  - Efficient and reasonable: harmonize the speed difference between CPU, RAM and devices. Control and manage devices in more efficient way.
  - Convenient: compatible, safe and stable
  - Standardization: the basement of IT industry

设备管理的重要思路

- 标准化 **但不代表没有各自的名字**
  - 为了减少应用软件开发人员的工作，上层应尽少感知硬件的差异，例如，所有控制光标的设备，传给上层的数据都一样
  - 为了减少操作系统开发人员的工作，OS应尽量减少为不同硬件做出的修改，例如，所有的鼠标的控制逻辑都一样
- 分类处理
  - 设备的类型过于多样和复杂，不可能所有的设备都归类到同一标准上
- 灵活性
  - 能够支持某类设备的新款式，例如，鼠标增加了新按键、键盘加了控制灯
  - 能够支持新类型的设备，例如，条码枪 模拟成一个键盘

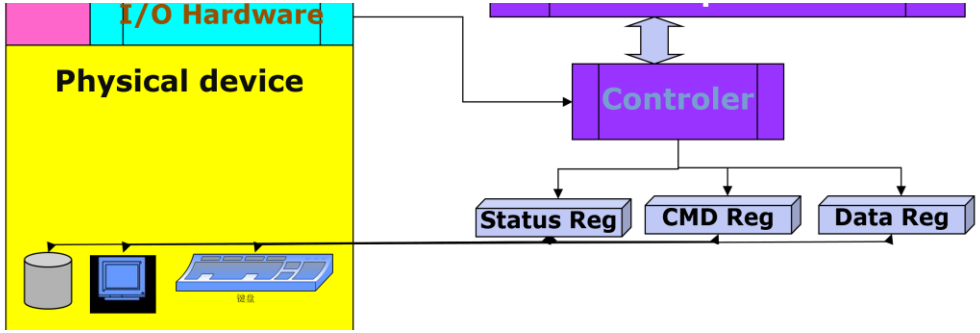
Purpose of Devices Management

- Device-independent programming interface
  - Hide the difference between hardware components
  - Provide simple and universal control methods for user
  - Maintain the safety and stability of user process
- Efficient management strategy
  - Allocation and releasing: like process scheduling
  - Performance enhancement: improve data transferring speed and make devices more adaptive for CPU and RAM
  - Protection: internal and external protection, deadlock
- Difficulties of devices management
  - Port address management
  - Control mode design
  - deadlock: unreasonable device request or assignment

设备的工作

转变为电信号——在外设里做

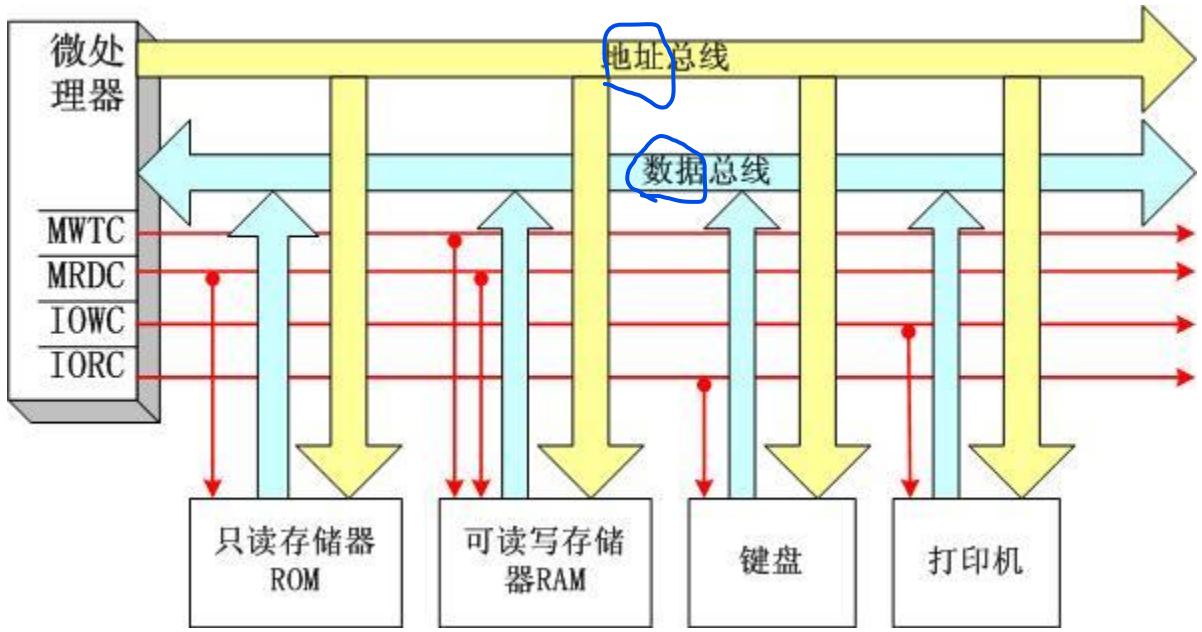
- 将外部的机械变化（如开关按下抬起）、模拟信号变化（如温度高低变化）等信息，转变为电信号（如电路通断、电压高低等），进而转变成为计算机可以处理的数字信号（按键编码、温度值等）的过程
- 设备完成上述过程后，将所得到的数据，放于自自的寄存器、数据buffer等结构中，供CPU读取，并准备好控制接口，供CPU发布控制命令
- 受限于CPU与其他器件连接的方式，设备制造商需要为设备制造“适配器”，以方便完成电路连接和数据格式连接



Communication between CPU and devices

- CPU如何读写设备提供的数据接口？

通过总线



How to access devices?

- I/O port
  - ID of the devices' registers
  - Computer maintains a I/O port list for I/O communication
  - Disadvantage: separates memory space and devices' registers
- Memory-mapped I/O
  - All devices' registers are mapped into memory space
  - Each register is assigned a unique memory address
  - Advantage: uniform address format

Dependent address : I/O port

- Advantages
  - Separate I/O address from memory address, special I/O instructions are designed to access I/O port
  - It is very easy to distinguish I/O access and memory access, the cost of management is little
- Disadvantage
  - Only simple instructions can be used for I/O communication, complex I/O programming is difficult
  - The programming model and address-mapping method are different, it is not convenient for programmers

I/O**也有**read**和**write**，后面接地址**

I/O控制器	I/O地址	中断向量
时钟	040-043	8
键盘	060-063	9
辅助RS-232	2F8-2FF	11
硬盘	320-32F	13
打印机	378-37F	15
单色显示	380-3BF	-
彩色显示	3DO-3DF	-
软盘	3FO-3F7	14
主RS-232	3F8-3FF	12

PC上的I/O控制器及其对应的I/O地址和中断向量

Memory-mapped I/O

I/O设备占据了一部分内存空间

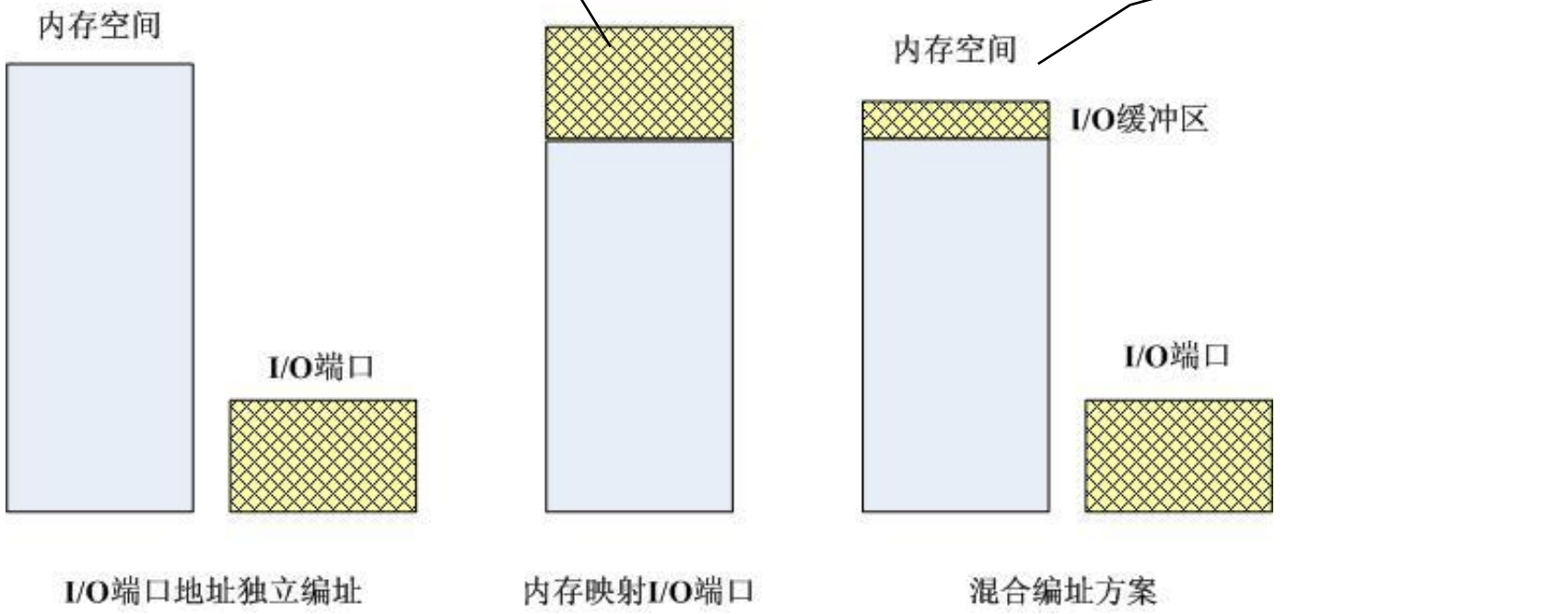
- Advantages
  - Treat I/O address as a part of memory address, an global address space is generated
  - The difference between I/O and memory is hidden, programmer can design complex I/O program
  - I/O address can be protected efficiently
- Disadvantage
  - It is difficult for hardware to distinguish the address of memory and I/O devices
  - The cost of management is higher, and it is more complex under double-bus or multi-bus architecture

Memory-mapped I/O

这种IO地址映射进入内存的设计，可以使IO设备与内存共享寻址机制与数据传输介质因此使用更为广泛

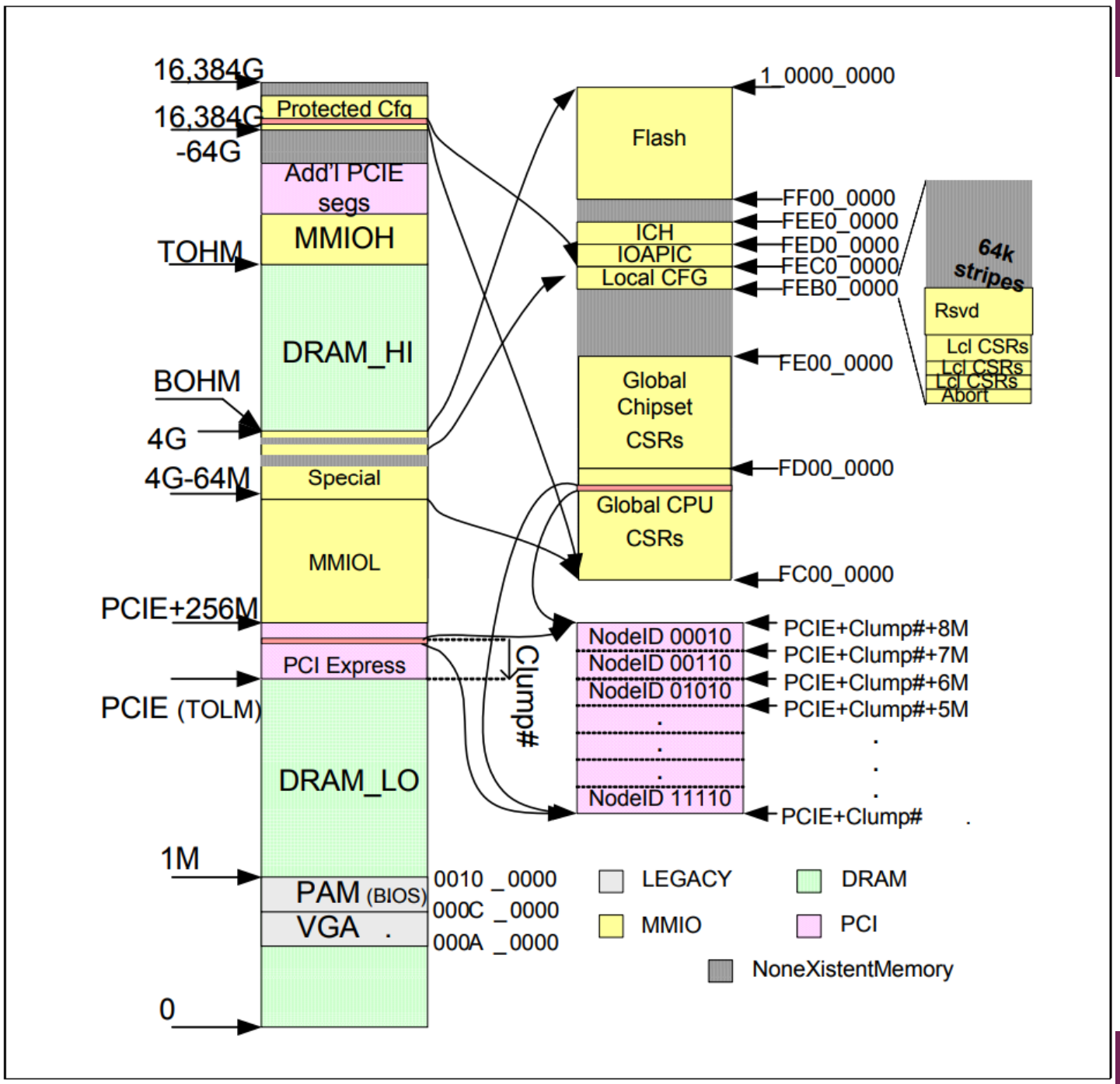
一条总线

Data buffer is mapped into memory space ;
Registers are identified by unique I/O port.
Real Sample: Pentium



- 现实中处

Figure 4-1. Intel® Xeon® Processor 7500 Series Address Map



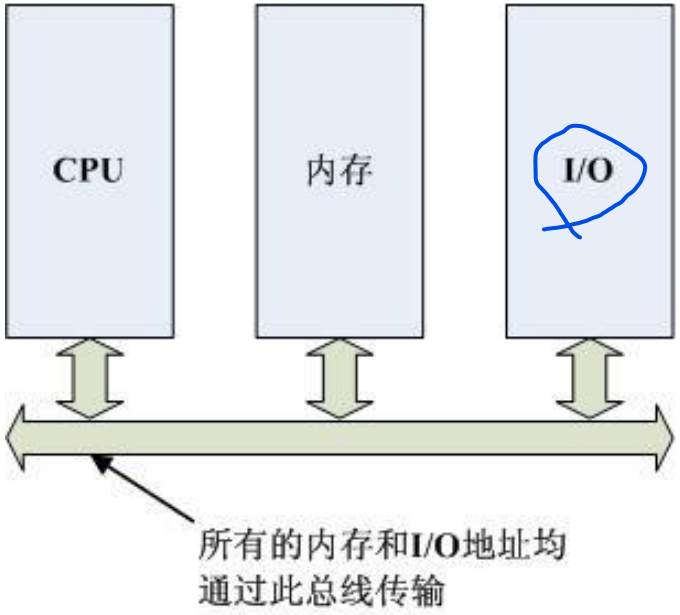
设备管理的问题

CPU从外设取指令：**用**I/O指令或者用内存映射的地址上的专门load **或者**store**指令**

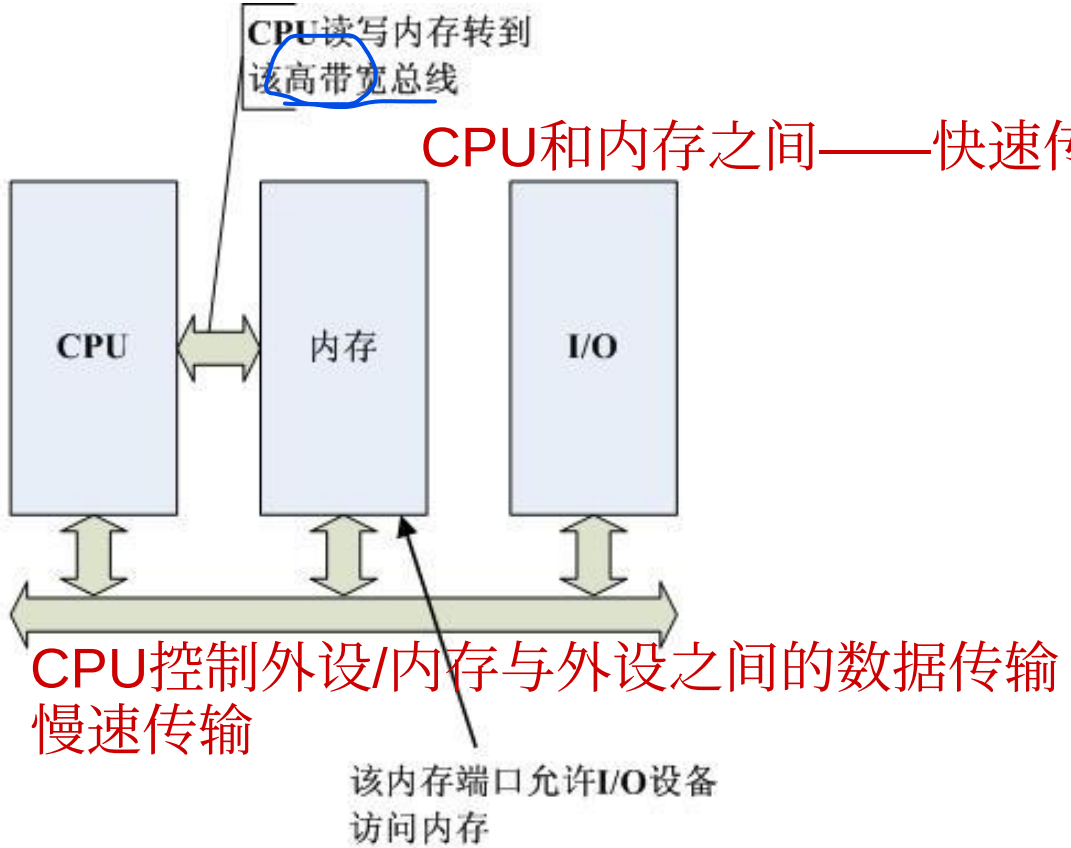
- 专用的IO指令以访问专用的IO端口区域读写
  - 普通指令还是特权指令?
  - IO指令是早于特权模型出现的
- 设备管理接口映射到地址空间的特定区域
  - 普通区域还是特权区域?
  - 物理内存地址还是虚拟内存地址?
- 设备管理程序是操作系统的一部分，或者与操作系统一并运转在特权模式
- 现代高性能系统中，为了提高效率，直接映射到用户空间

How to access devices---BUS

共享的总线方便了设计但也限制速度

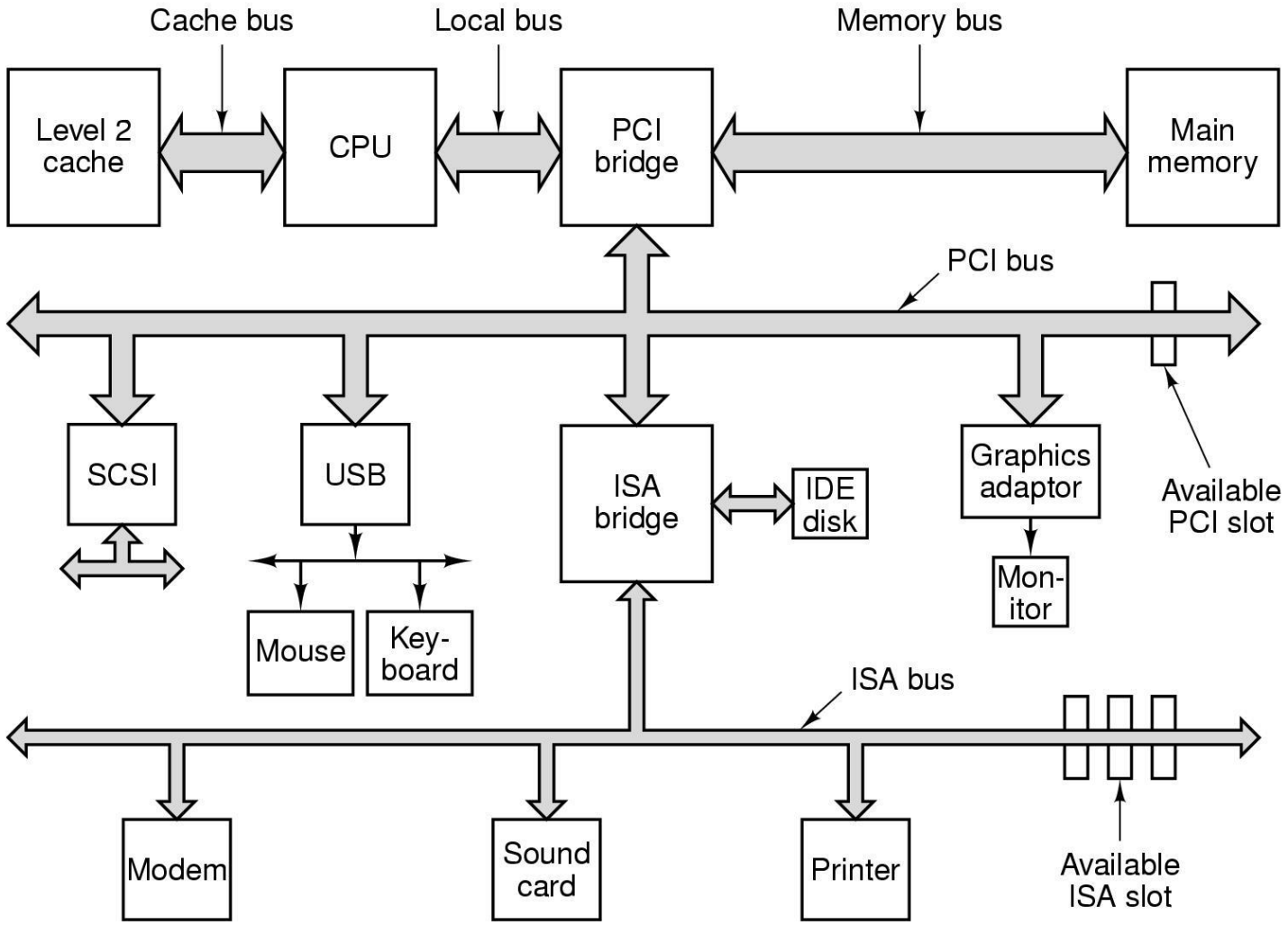


单总线体系结构



双总线体系结构

How to access devices---BUS



如何及时发现数据并传输

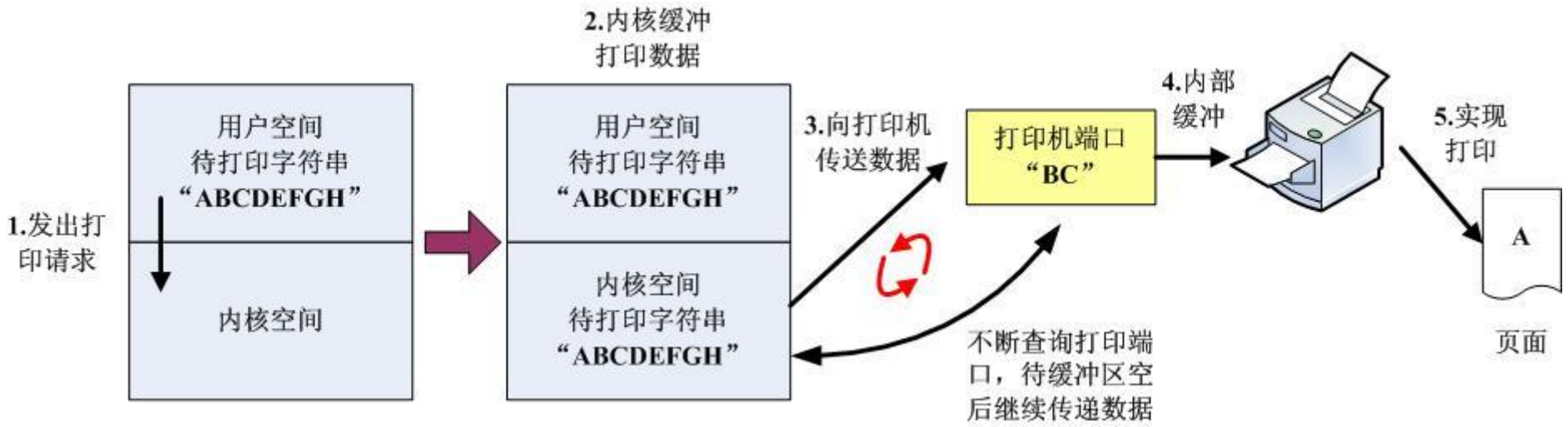
什么时候从外设取数据？

- 发现
  - 等待并轮循
  - 中断
- 传输
  - 由CPU发起，逐字节传输
  - 由DMA设备代为发起，以块为单位传输

Working mode of devices: busy waiting

CPU一直等待
简单

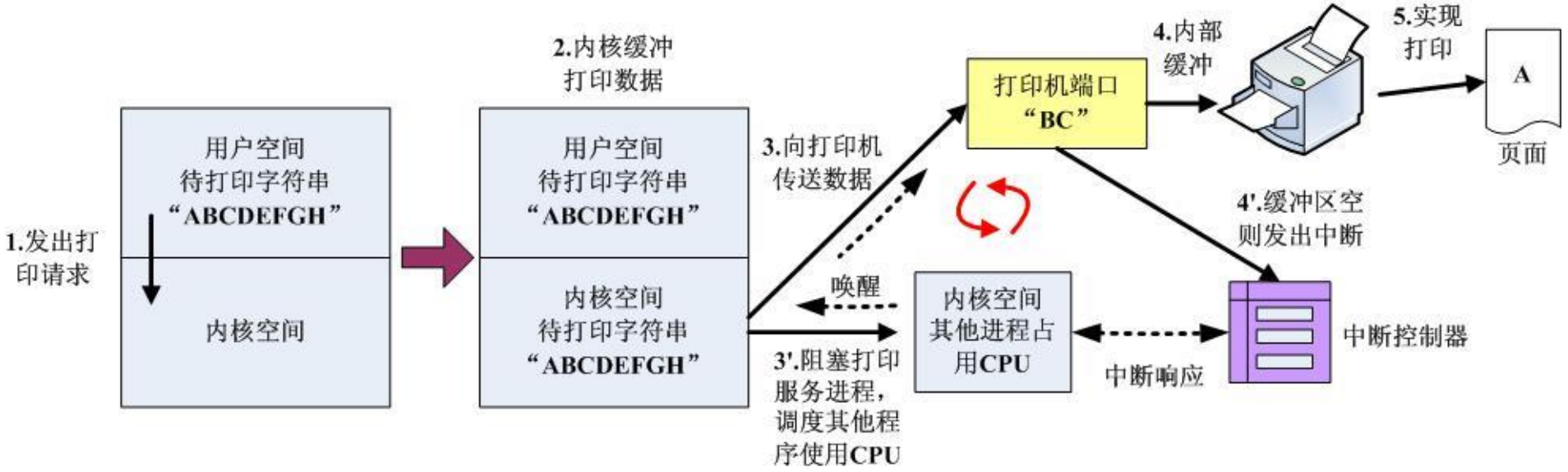
- Special kernel process sends the data to device port;
- The process checks the port repeatedly until the port is available and sends rest data;
- The user process continues run after the kernel process is finished;
- Disadvantage: CPU is wasted too much



Working mode of devices: interrupt

使用硬件感知中断
中断时CPU过来

- Special kernel process sends the data to device port;
- The process goes to sleep and CPU will run other processes;
- The device send interrupt to CPU after the data buffer is empty;
- The kernel process is waked up and send the rest data
- Disadvantage: frequent interrupts are time-consuming

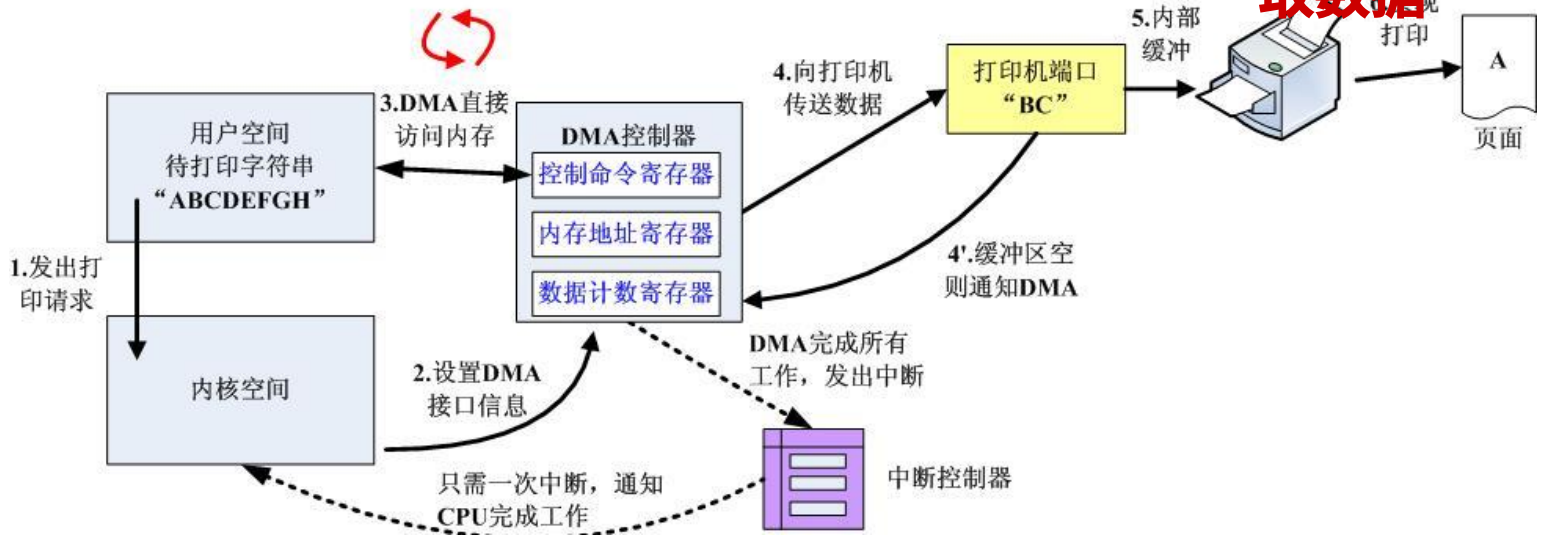


Working mode of devices: DMA

- User process causes a CPU trap, the special kernel process sets the registers in device and exits;
- Device read data from memory directly;
- The device send interrupt to CPU after the job is done;
- The user process is waked up and continues run

设置电路：
1.响应中断
2.产生数据的读写指令

CPU**不用**/用另一块内存时，DMA才能真正实现到内存中读**取数据**



Thanks for your time!
Questions & Answers