Part 1. Overview

第一章 操作系统导论

Note : These lecture materials are based on the lecture notes prepared by the authors of the book titled *Operating System Concepts.*

学习目标

- 掌握计算机系统的基本组织结构
- 掌握操作系统的主要组成部分
- 了解其他计算环境

学习内容

1. 操作系统定义
2. 计算机系统组织
3. 计算机系统体系结构
4. 操作系统结构
5. 操作系统操作
6. 操作系统管理
7. 其他计算机系统

第一节
操作系统定义

1. 定义

Windows

Unix，Linux

Mac OS

安装系统

WinCE

Android

iOS

1. 定义

操作系统是什么？
它做什么工作?

计算机用户与计算机硬件之间运行的一个程序。

1. 定义

操作系统是什么?

它做什么工作?

1. 资源管理
2. 提供服务

- 硬件资源
- 软件资源

1. 定义

1. 硬件资源
   - 中央处理器、内存、硬盘、打印机等
2. 软件资源
   - MS Office, eClipse, Visual Studio etc.
3. 服务提供
   - 方便用户

操作系统的目标?

1. 通过运行计算机程序方便用户解决问题
2. 方便用户使用计算机
3. 有效使用计算机硬件

1. 定义

操作系统是一直运行在计算机上的程序（通常称为内核Kernel），其他程序则为系统程序和应用程序。

1. 资源管理平台

2. 运行应用程序平台

3. 用户服务平台

第二节
计算机系统结构

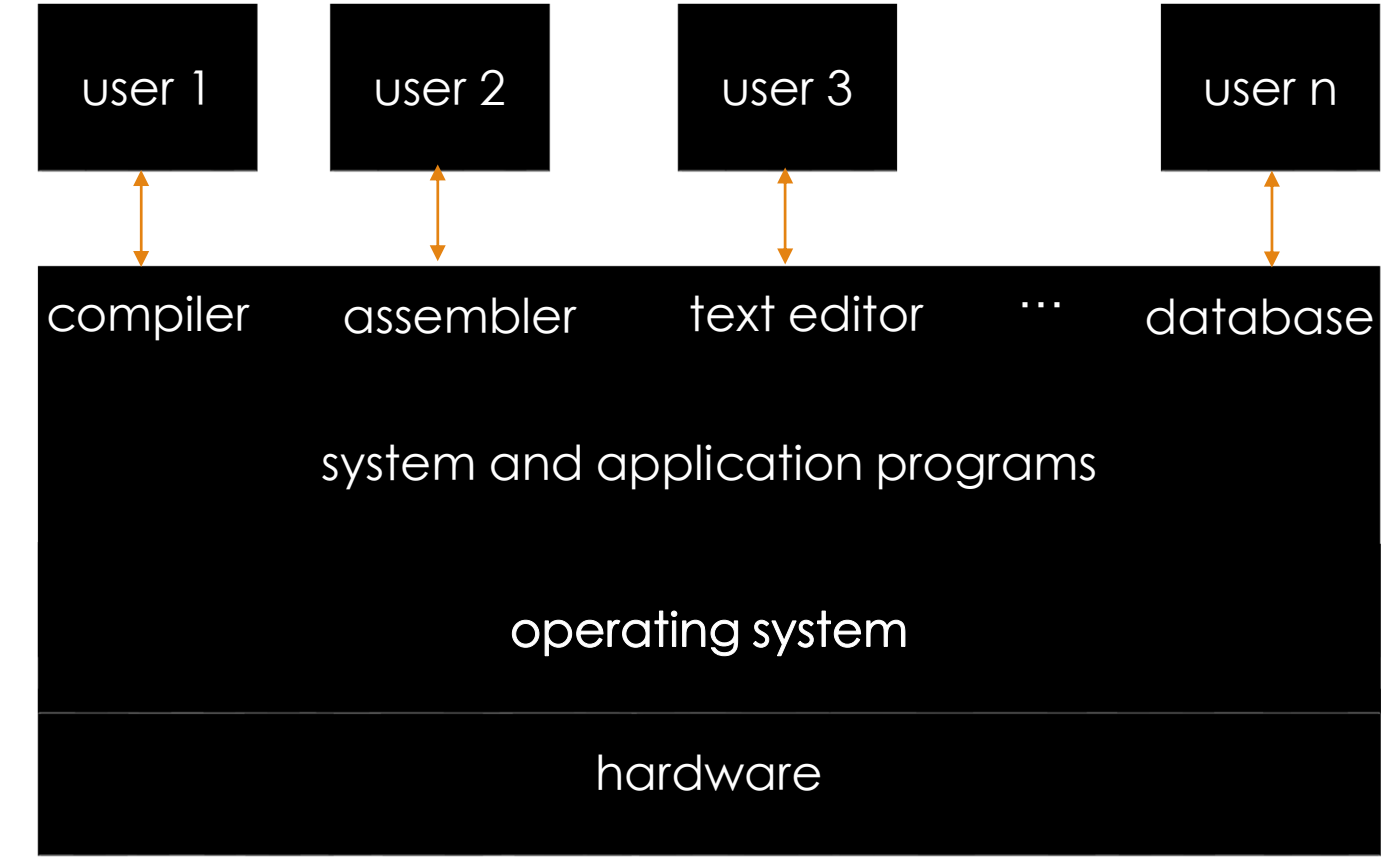
2. 计算机系统结构

计算机系统由以下四大部门组成

1. 硬件	2. 操作系统	3. 应用程序	4. 用户
• Provides basic computing resources， for example, CPU, memory, I/O devices	• Controls and coordinates use of hardware among various applications and users	• Define the ways in which the system resources are used to solve the computing problems of the users, • for example, ms office, compilers, web browsers, database, games	• People, • Machines, • Other computers

2. 计算机系统结构

层次结构



第三节
操作系统的组织

3. 计算机系统组织

3.1 启动

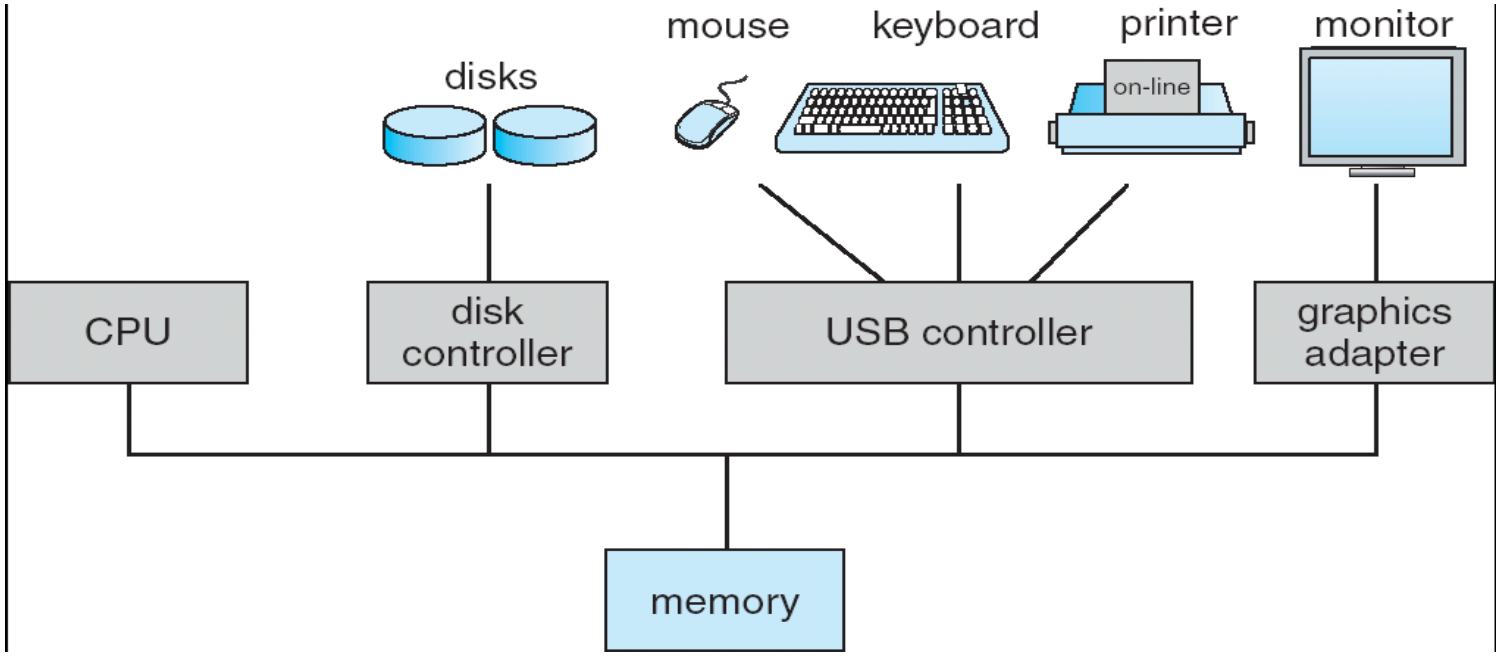
3.2 中断 （Interrupt）

3.3 I/O 结构 （I/O structure）

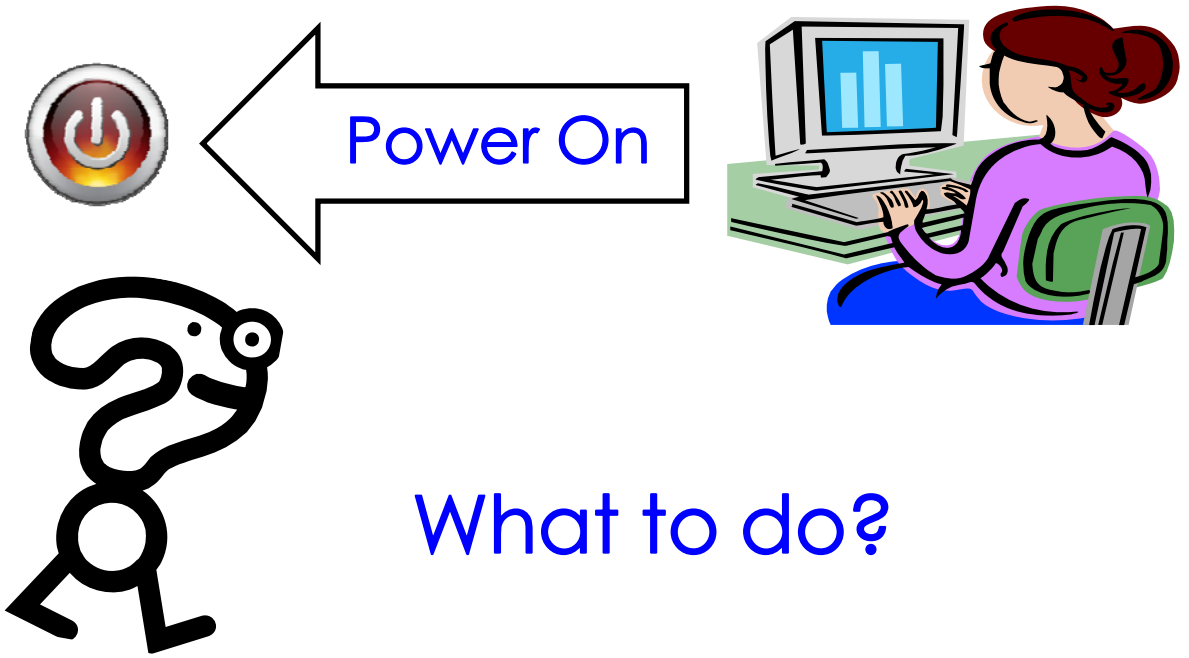
3.4 存储结构 （Storage structure ）

计算机系统操作

单个或多个中央处理器、设备控制器通过总线



3.1 启动



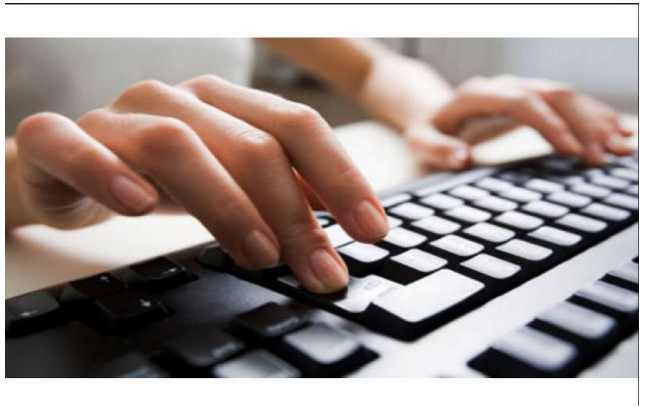
3.1 启动

- 确认每个设备是否工作正常
- 确认无误后，开启引导程序

- bootstrap 引导 程序一般位于ROM或EEROM中称为计算机硬件中的固件。
- 工作内容包括以下内容：
  1. 设备初始化
  2. 把操作系统载入到内存中
  3. 运行第一个进程 init(), 等待事件发生

3.2 中断

Watch me



Touch me

3.2 中断

一个事件的触发是通过硬件或软件的中断来实现的

在实现操作系统功能时，中断是一个非常重要的实现机制

软中断

硬中断

通过软件触发中断。如系统调用(System Call) 会触发软件中断。如，系统调用、异常

硬件通过向CPU发送信号来触发中断，一般对用户不可见，但可触发程序的运行。

3.2 中断

- 一旦发生中断，CPU会运行中断服务程序
- 每个中断都有自己的“中断服务程序”Interrupt service routine
- 系统应持有中断与中断服务程序之间的对应表，对应表称为中断向量表（Interrupt vector table）

中断举例

- I/O设备发生的事件，如按键盘、点击鼠标、磁盘的读写等；
- 异常事件、重要事件的发生、如断电、部件失灵等；
- 非法指令，如除于零、下达不存在的指令等。

3.2 中断的操作流程

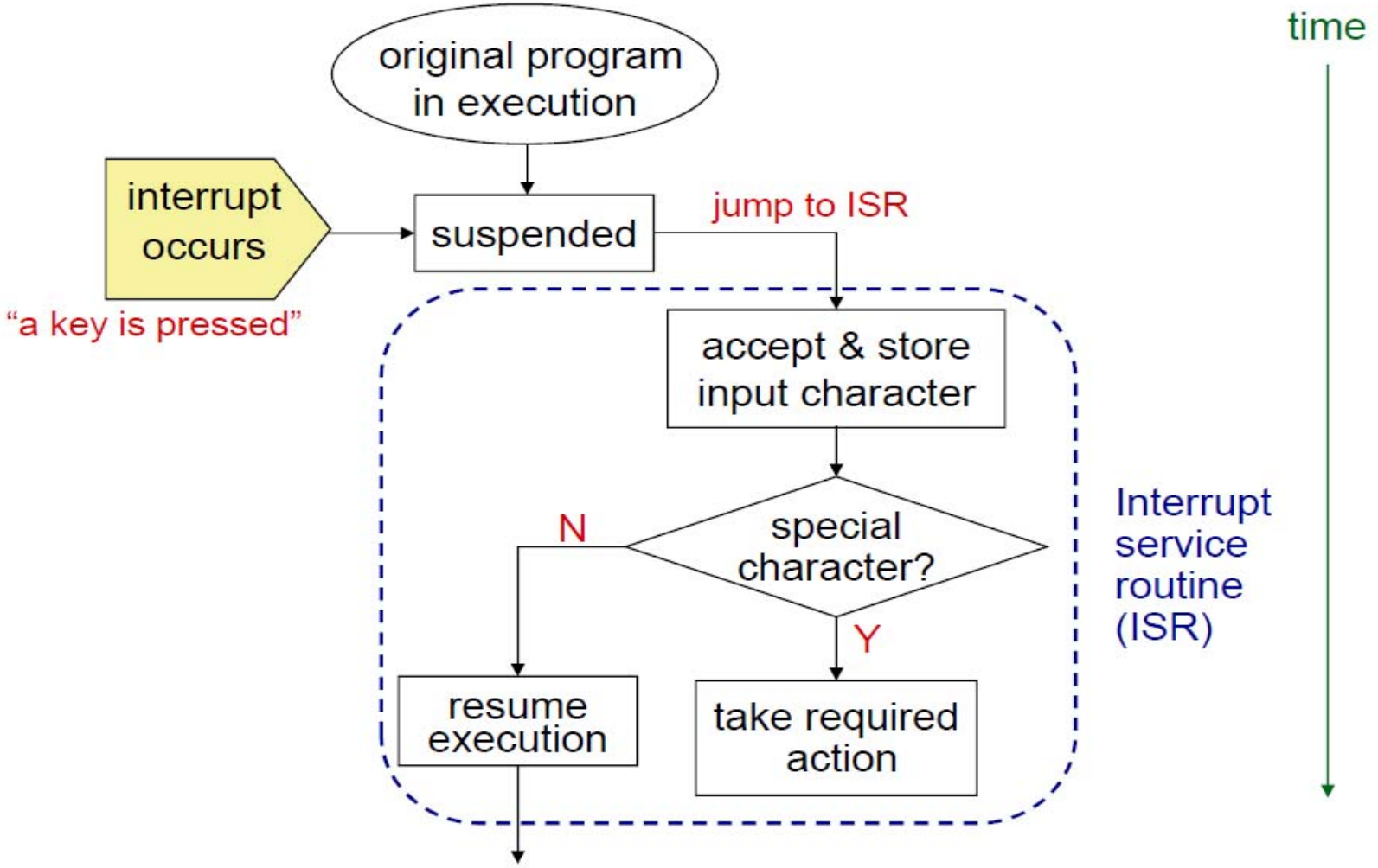
当发生中断时，

- 保存当前进程的程序计数器（program counter）
- 跳到相应的中断服务程序中

一旦中断服务程序运行结束，

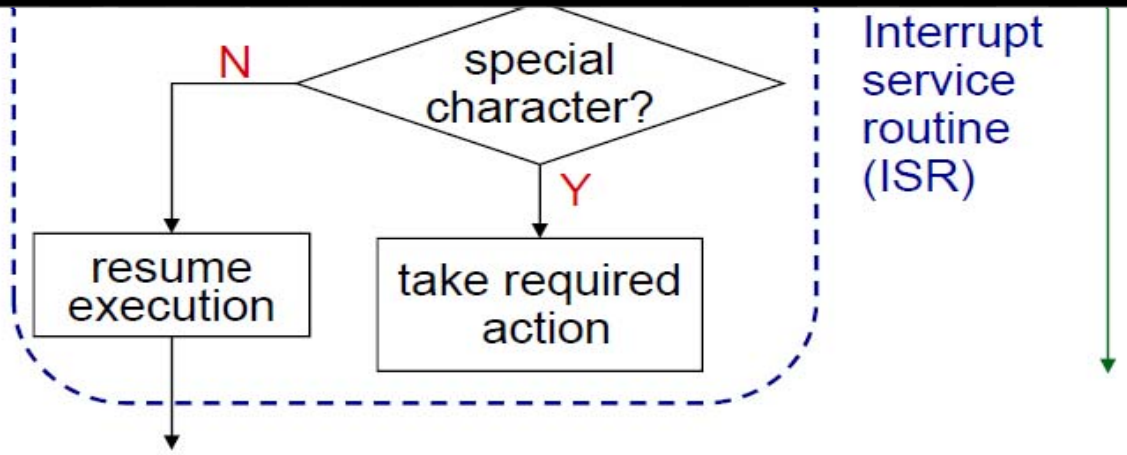
- 返回到被中断的程序，并继续往下运行
- 或者返回到OS制定的程序中

3.2 中断- 键盘为例


original program in execution
interrupt occurs
"a key is pressed"
suspended
jump to ISR
accept & store input character
special character?
N
Y
resume execution
take required action
Interrupt service routine (ISR)
time

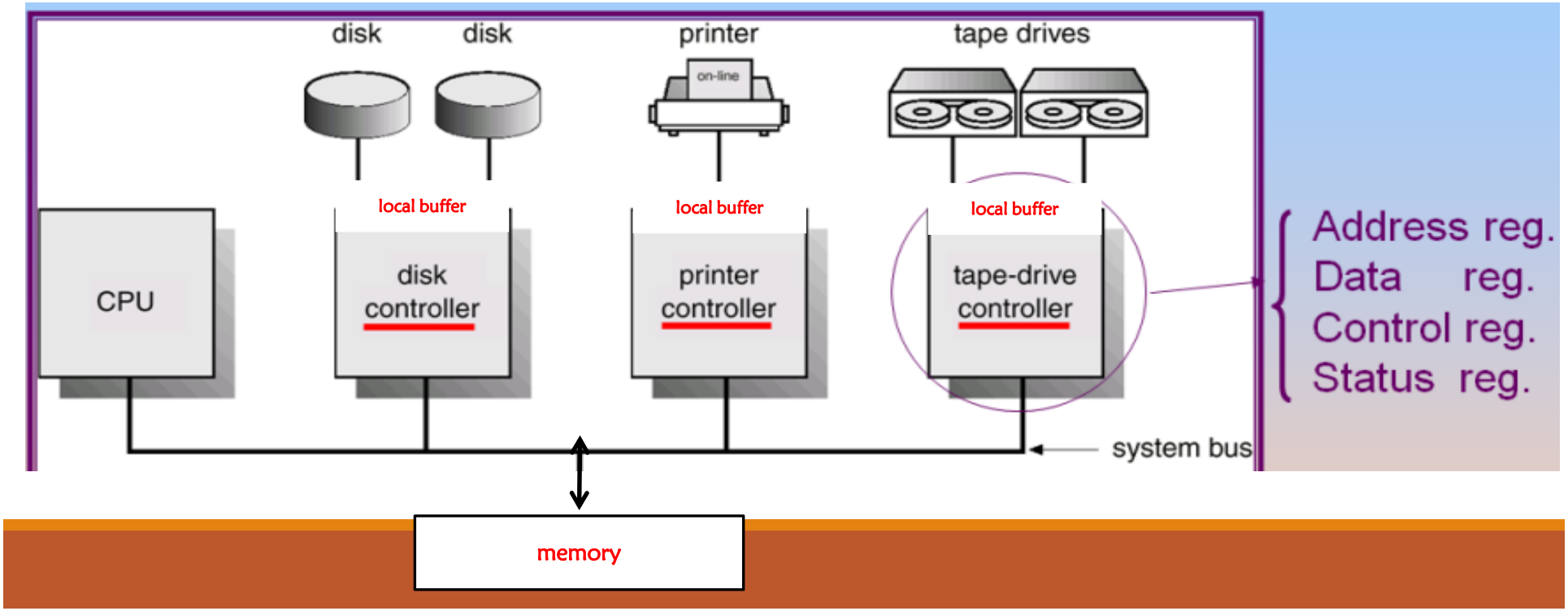
3.2 中断- 键盘为例

我们可以认为，现代操作系统是以中断驱动或事件驱动的系统



3.3 I/O 结构

1. I/O 设备与 CPU 可同时运行
2. 每个设备控制器负责相应类型的设备（如， 磁盘控制器负责磁盘）
3. 每个设备控制器拥有自己的本地缓冲器和寄存器



Tips

Stand I/O – “stdio.h” in C library

1. 标准输入
2. 标准输出
3. 标准错误

```
#define stdin    __create_file(0)
#define stdout   __create_file(1)
#define stderr   __create_file(2)
```

File descriptor (File pointer) 文件标识符

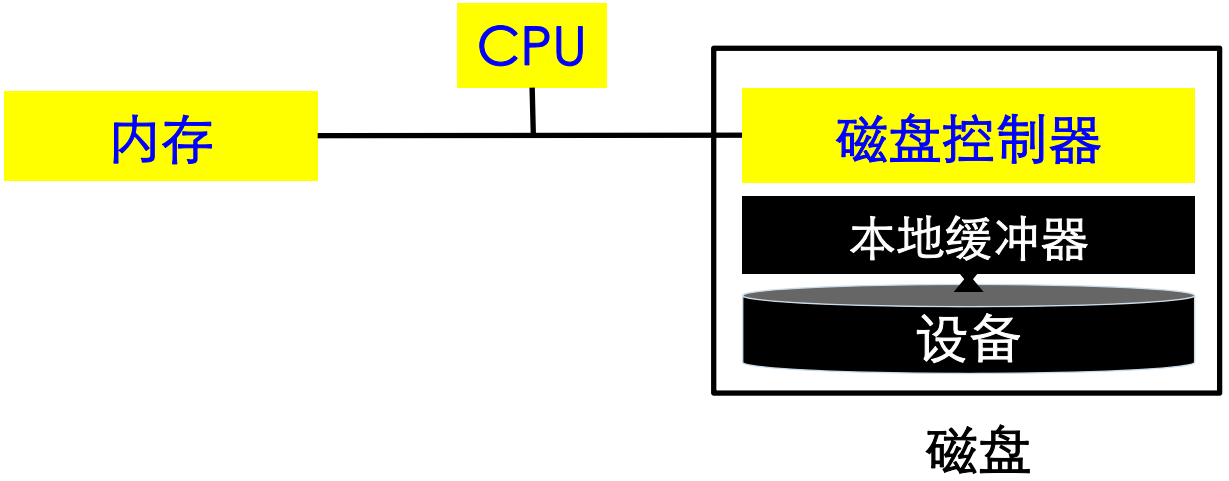
0： 表示标准输入

1： 表示标准输出

2： 表示标准错误

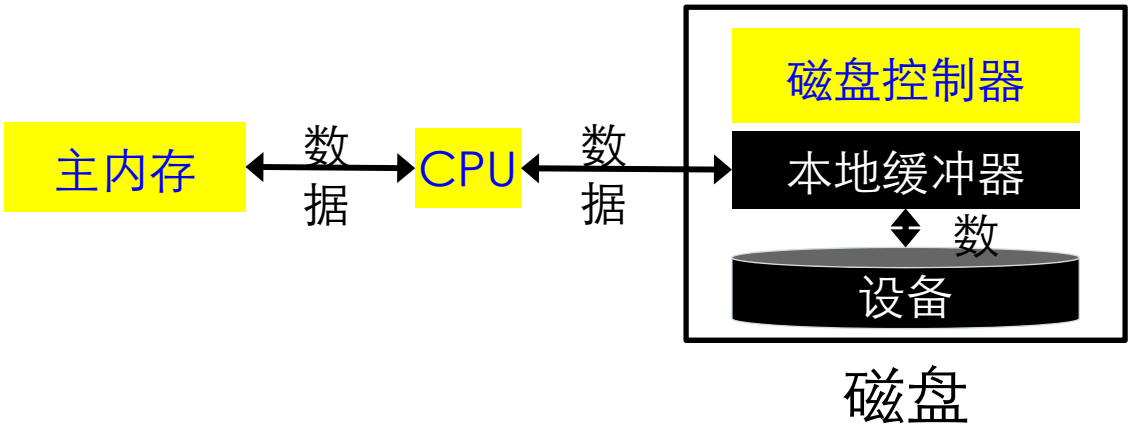
3.3 I/O 结构

1. CPU负责内存与本地缓冲器之间的数据传递
2. 设备控制器负责在其所控制的外部设备与本地缓冲存储之间进行数据传递
3. I/O操作结束后，设备控制器通过中断通知CPU，表示I/O结束



3.3 I/O 结构

举例，磁盘写的操作是怎么工作的?

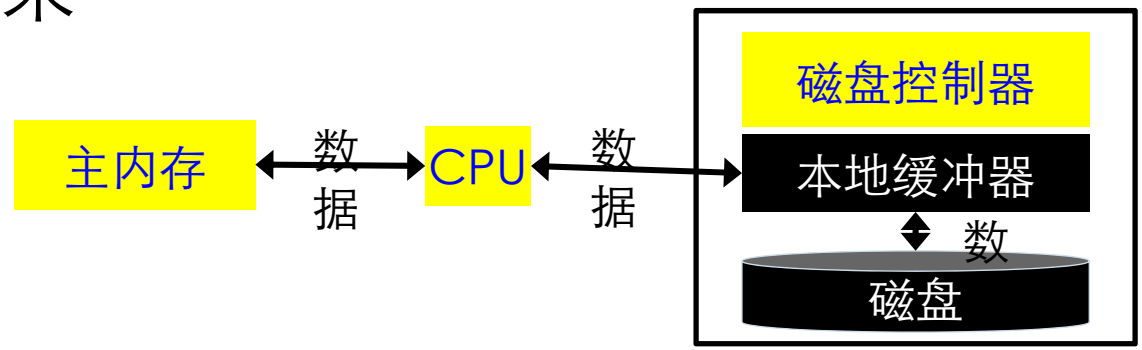


- 传输单位为字节(byte)；
- CPU 和磁盘控制器小容量的寄存器用于传输数据

有什么问题?

3.3 I/O 结构

磁盘控制器每次传输数据时应通知CPU，表示I/O结束



- Notice that, the CPU and disk controller only have small registers for delivering data.
- Delivering unit is byte

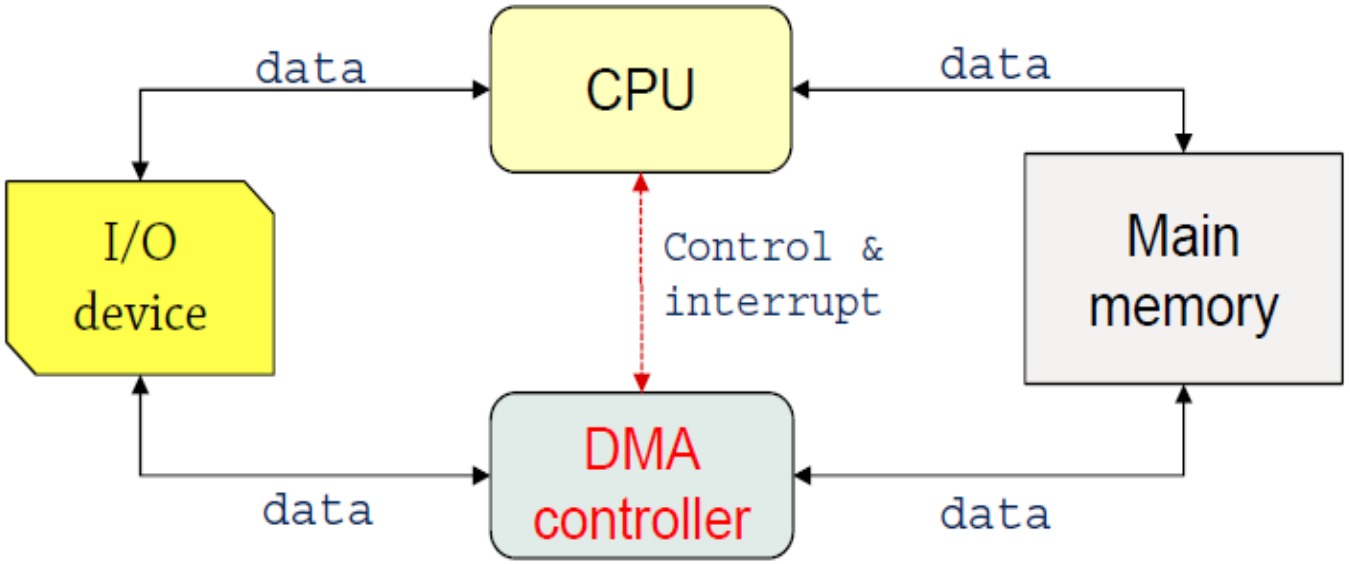
CPU说：你烦不烦?
能不能让我干点别的?

DMA直接访问内存

DMA(Direct Memory Access)是在专门的硬件控制下，实现高速外设和主存储器之间自动成批交换数据尽量减少CPU干预的输入/输出操作方式。

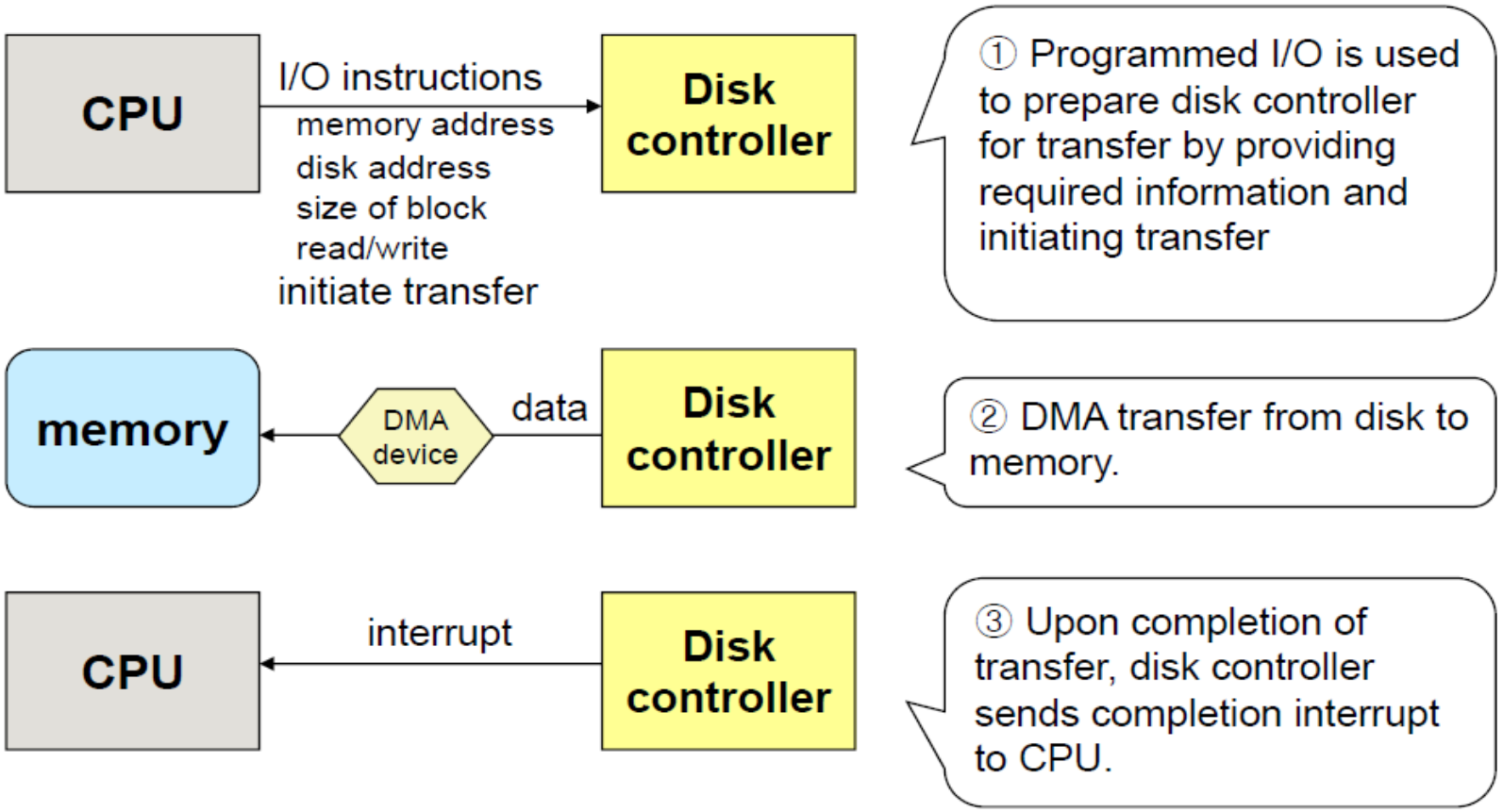
1. 有利于高速 I/O 设备传送数据，接近于内存速度
2. DMA 控制器，在没有CPU干涉的情况下，以块儿（Block）为单位，负责在设备缓冲器与主存之间的数据传送
3. 之前是每个字节传送完成后就触发中断表示完成，而DMA是每完成块儿传送后，触发中断。

直接访问内存


data
CPU
data
I/O
device
Control &
interrupt
Main
memory
DMA
controller
data
data

DMA 操作 – 举例

读操作举例



3.3 I/O 结构 – I/O Method（方法）

After I/O starts

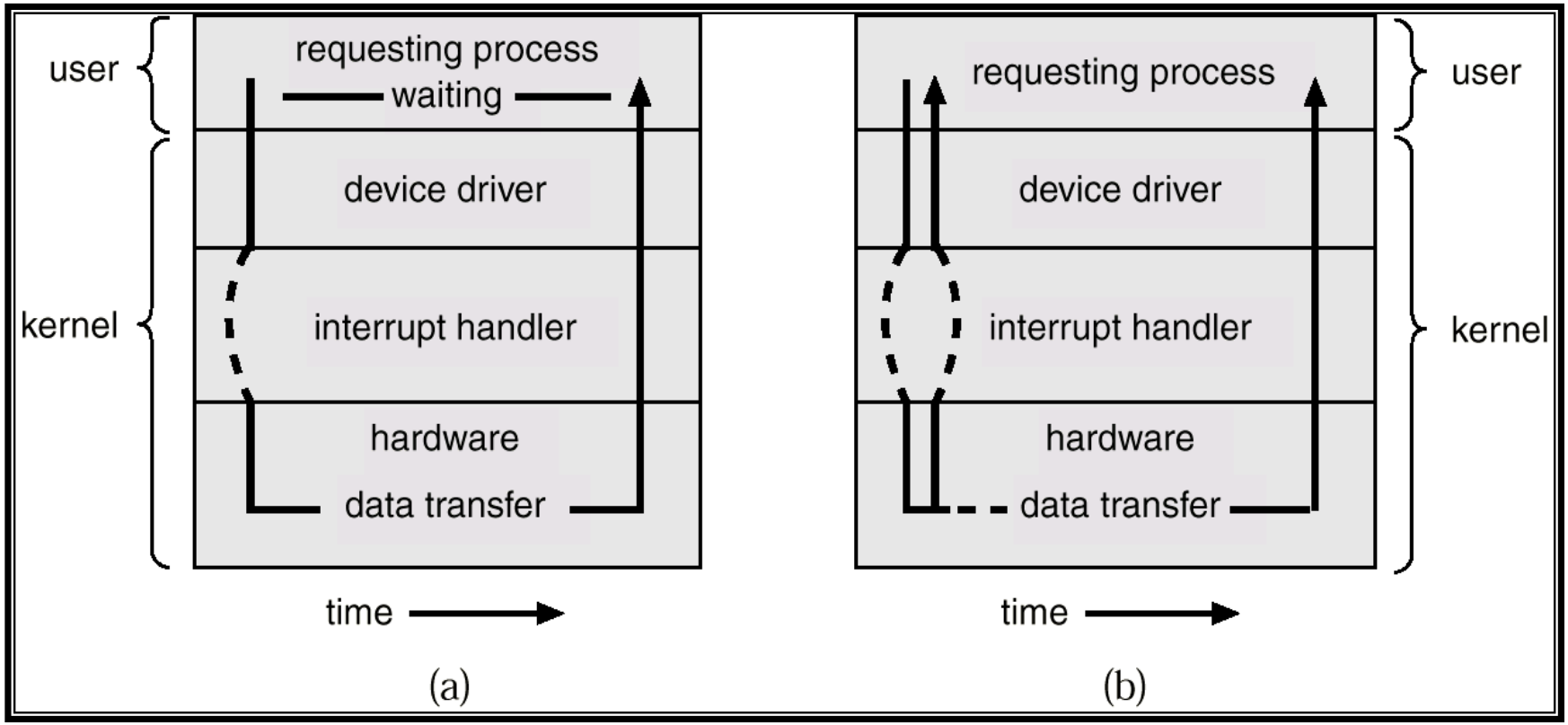
1. 同步 （ Synchronous ）
   - 只有 I/O 结束后，用户程序才能获得控制权

2. 异步 （ Asynchronous ）
   - I/O 还没有结束的情况下，用户程序可以获得控制权

3.3 I/O 结构

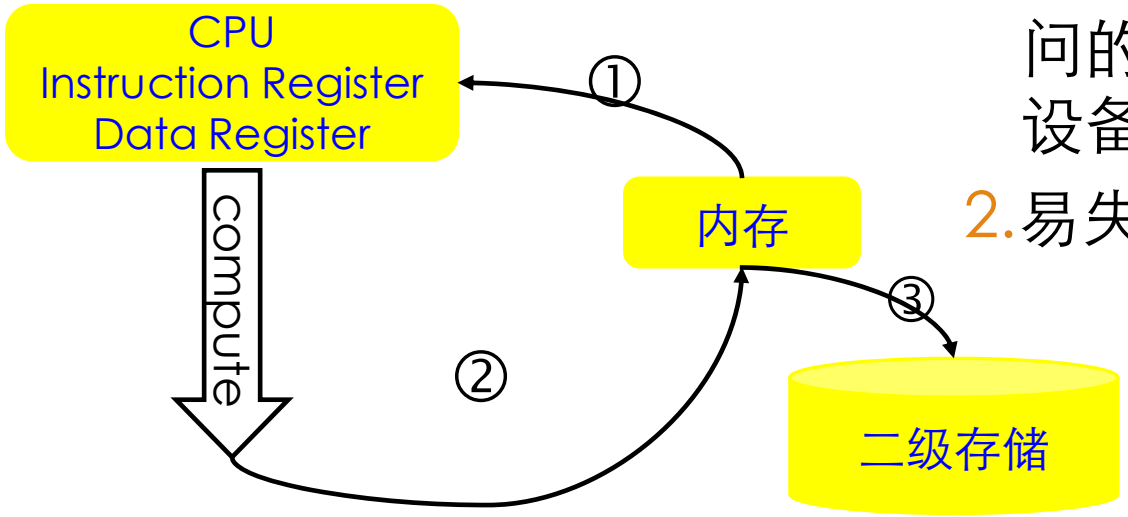
同步

异步


user
kernel
requesting process
waiting
device driver
interrupt handler
hardware
data transfer
time
(a)
requesting process
device driver
interrupt handler
hardware
data transfer
user
kernel
time
(b)

3.4 存储结构

1. 指令寄存器
2. 数据寄存器



主存（内存）

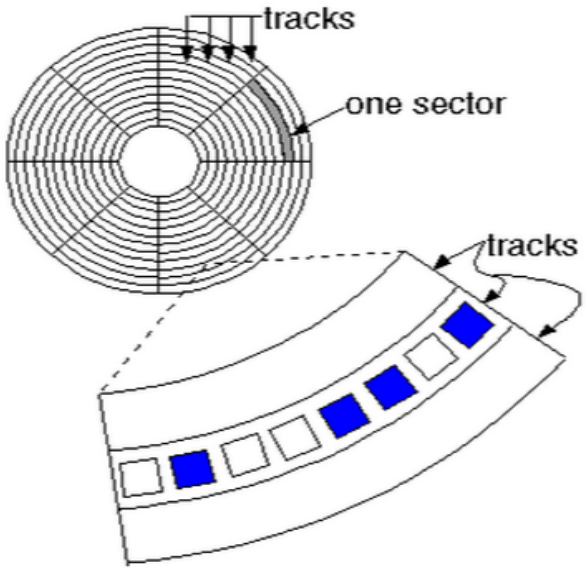
1. CPU可以直接随机访问的唯一大容量存储设备
2. 易失的设备

二级存储－不易失的存储设备，一般是磁盘类

3.4 存储设备

磁盘 （ Magnetic Disks ）

1. 磁盘的表面可以逻辑划分为多个磁道（track），而每个磁道在划分为扇区（sector）
2. 磁盘控制器负责设备与计算机之间的逻辑交互



3.4 存储设备

闪存 Flash Memory

- 芯片级别，移动存储设备
- 分为Nand Flash 和Nor Flash

固态状态硬盘，简称固态硬盘（Solide-State-disk： SSD）

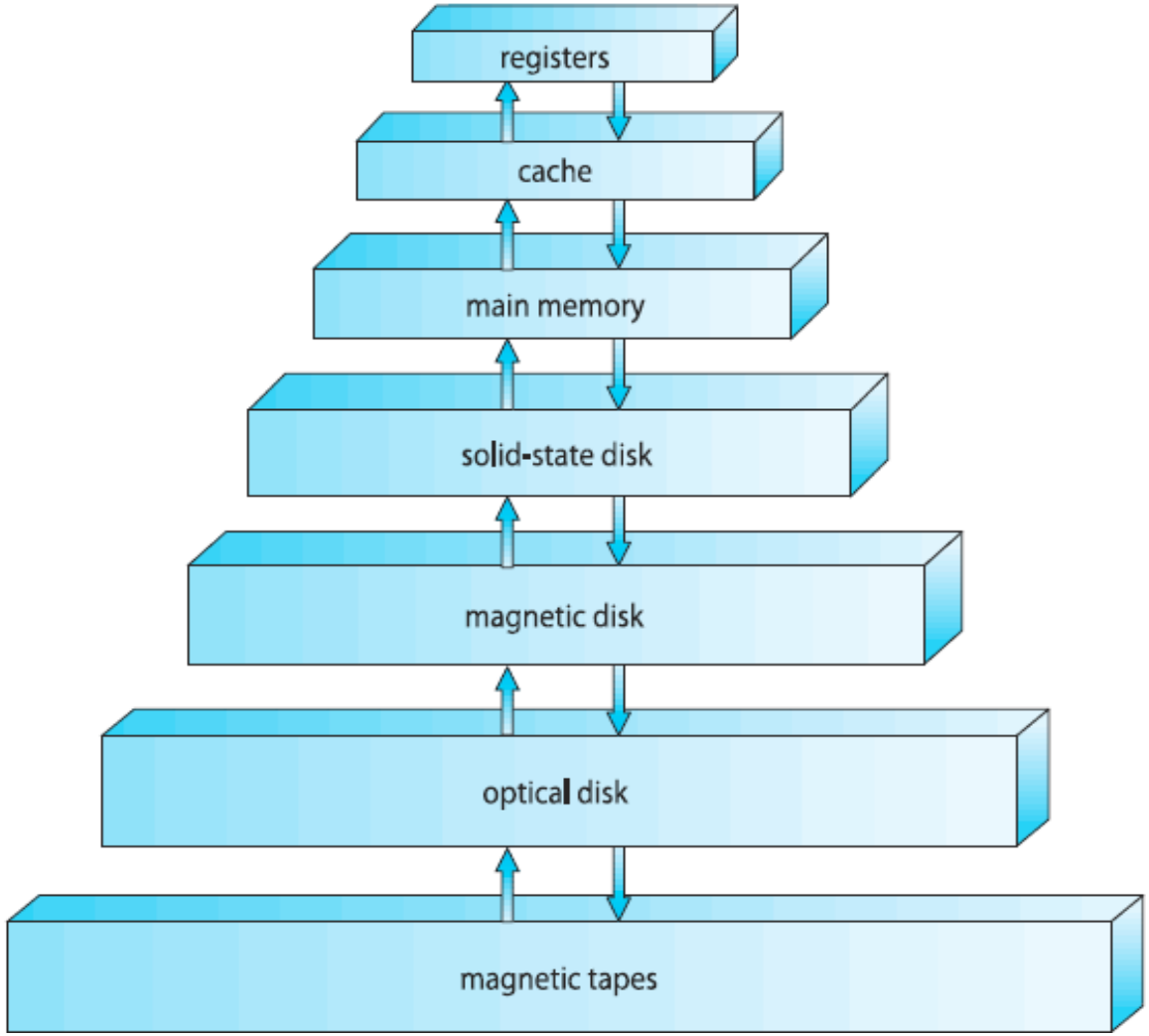
- 比磁盘速度快，普及
- 基于NAND Flash的集成品

Compact flash (CF) & secure digital (SD) cards, a Sony memory stick, and a USB memory key.

3.4 存储结构- 层次结构

1. 速度
2. 价格
3. 易失性

表现出层次结构



Units of Digital Information

Summary: Specific Units of Digital Information

IEC prefix		Representations			Customary prefix	
Name	Symbol	Base 2	Base 1024	Base 10	Name	Symbol
kibi	Ki	2^{10}	1024^{1}	$\approx 1.02 \times 10^{3}$	kilo	k, K
mebi	Mi	2^{20}	1024^{2}	$\approx 1.05 \times 10^{6}$	mega	M
gibi	Gi	2^{30}	1024^{3}	$\approx 1.07 \times 10^{9}$	giga	G
tebi	Ti	2^{40}	1024^{4}	$\approx 1.10 \times 10^{12}$	tera	T
pebi	Pi	2^{50}	1024^{5}	$\approx 1.13 \times 10^{15}$	peta	P
exbi	Ei	2^{60}	1024^{6}	$\approx 1.15 \times 10^{18}$	exa	E
zebi	Zi	2^{70}	1024^{7}	$\approx 1.18 \times 10^{21}$	zetta	Z
yobi	Yi	2^{80}	1024^{8}	$\approx 1.21 \times 10^{24}$	yotta	Y

- binary prefix = IEC prefix
- Source: http://en.wikipedia.org/wiki/Binary_prefix

第四节
计算机系统体系结构

4. 计算机系统结构

1.单处理器系统

Single processor systems

2.多处理器系统

Multi processor systems

3.集群系统

Cluster systems

4.1 单处理器系统

1. 系统中，只有一个通用处理器，用来处理来自用户进程的指令
2. 除了通用处理器，系统一般还包括其他专用处理器，如磁盘控制器（处理器）、图形控制器（处理器）等
3. 专用处理器不接受用户的指令
4. 有的专用处理器与通用处理器集成在一起，通用处理器具有专用处理器功能

4.2 多处理器系统

Multiprocessor System, 系统中，有多个处理器，又称为并联系统（parallel systems），多个处理器共享一个内存

特点

- CPU之间通过共享内存来进行通讯
- 操作系统可以运行在某一个CPU上或多个CPU上

优点

- 增加了吞吐量
- 方便扩展
- 增加了可靠性 – graceful degradation or fault tolerance（容错）

4.2 多处理器系统

1. 非对称处理器(异构多处理器)

Asymmetric multiprocessor(ASMP, AMP)

- 处理器在结构上不同
- 一个处理器负责运行操作系统，其它处理器运行其他程序， 处理器之间有主从关系

2. 对称处理器(同构多处理器)

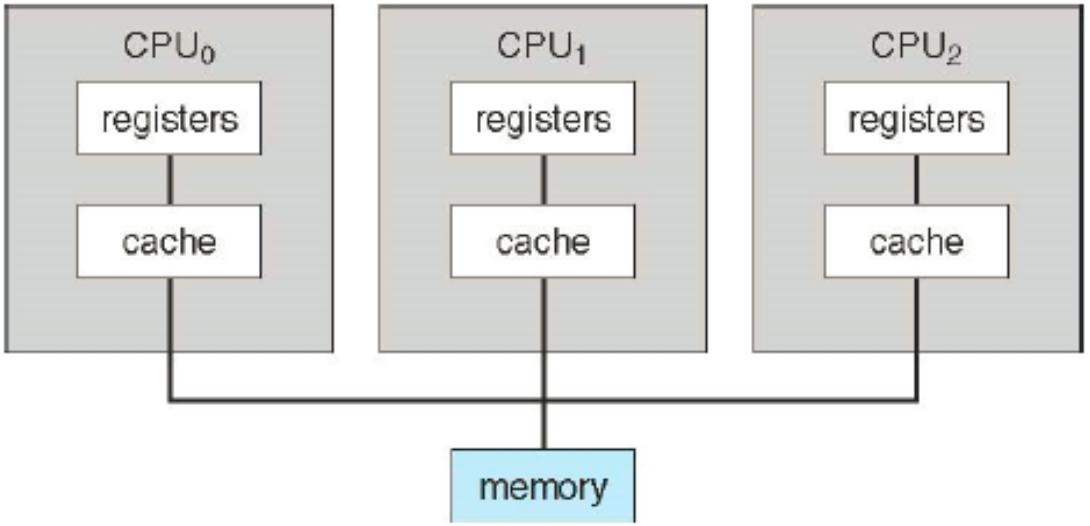
Symmetric multiprocessor(SMP)

- 各个处理器在结构上完全相同
- 操作系统可以运行在任何一个处理器上

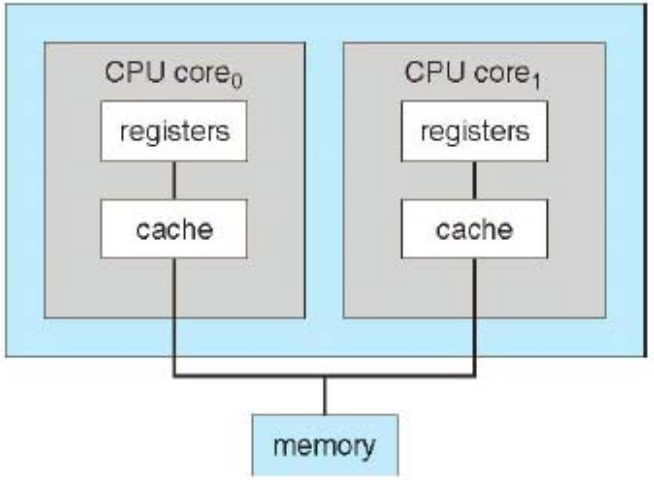
3. Multi-Core -多核

- 一个处理器上有多个CPU
- 可以是 SMP 也可以是 ASMP

4.2 多处理器系统



Symmetric multiprocessing architecture

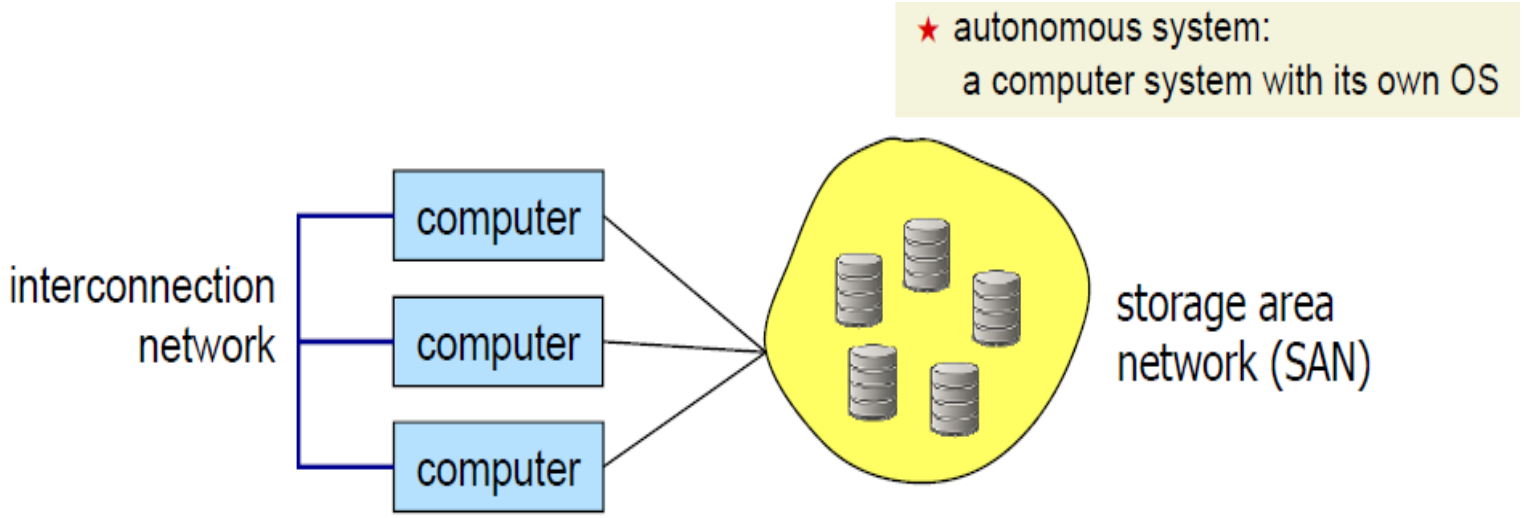


Dual-core microprocessor

注意：不管多处理还是多核、每个处理器都有自己的寄存器和高速缓存

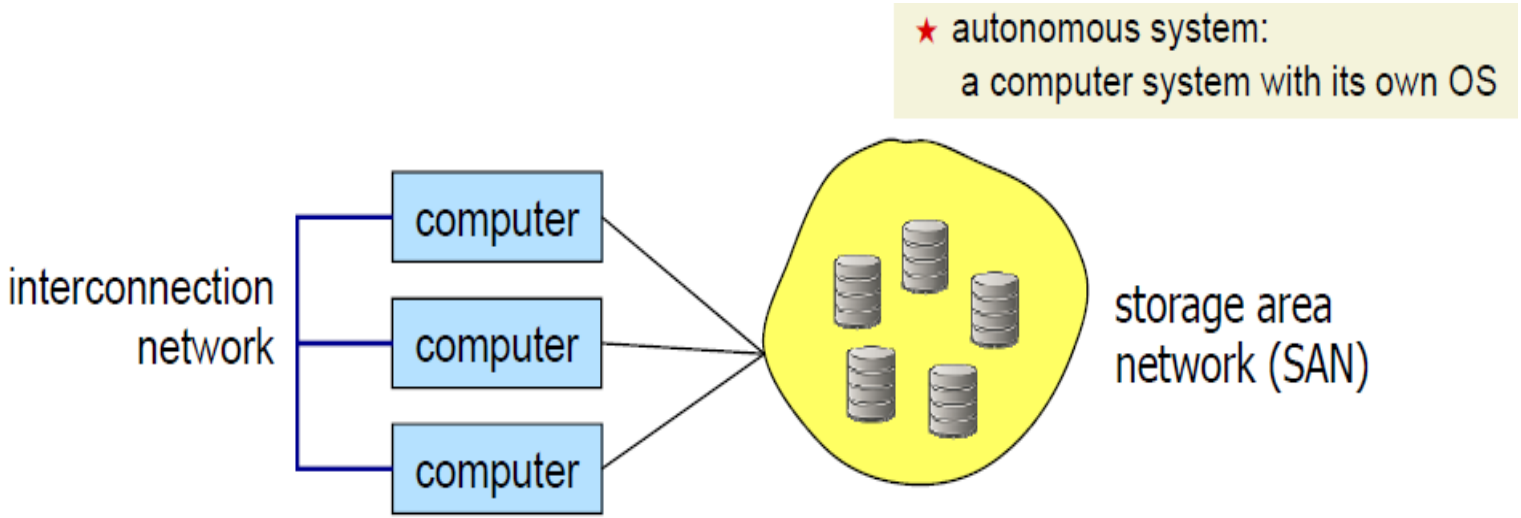
4.3 集群系统

集群系统是一种由互相连接的计算机组成的并行分布式系统（Distributed System），并共享存储，可以作为单独、统一的计算资源来使用。



4.3 集群系统

1. 多个自治系统（计算机系统）协同工作
2. 每个节点(计算机系统)都有自己的操作系统
3. 一个集群系统可以看成是单个逻辑计算单元，它由多个节点通过网络连接组成



4.3 集群系统

优点 - 高性能

计算机集群提供了更快的处理速度，更大的存储容量，更好的数据完整性，更高的可靠性和更广泛的可用性资源。

缺点 – 高费用

但是，需要更昂贵的实现和维护。相比一台计算机，这将导致更高的运行开销

4.3 集群系统

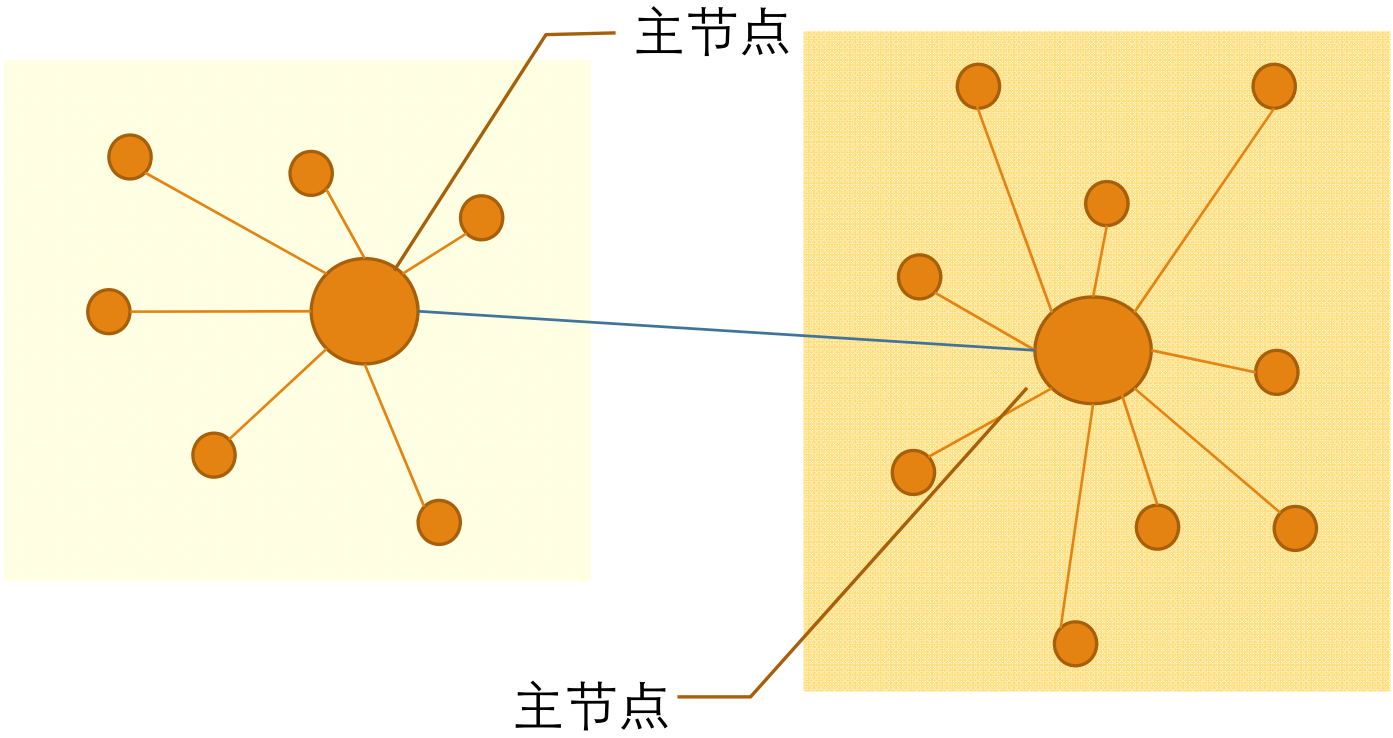
1. 非对称集群（Asymmetric clustering）：选出一个节点当主节点(header node)，即主机待机模式的节点

Host standby node is a Header node,
Header node monitors all the other nodes

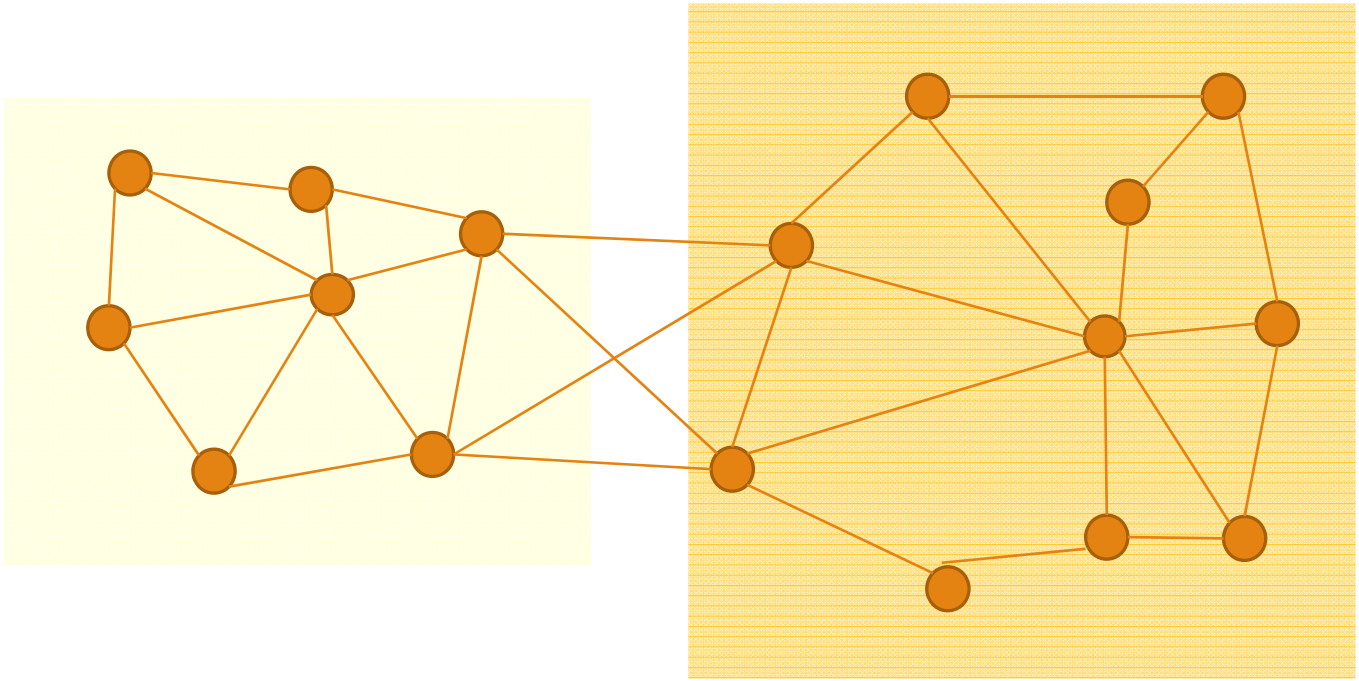
2.对称集群（Symmetric clustering）：没有主节点，全部节点地位相等，没有主从关系

No Host standby node,
All nodes are monitored each other

非对称集群



对称集群



第五节
操作系统结构及操作

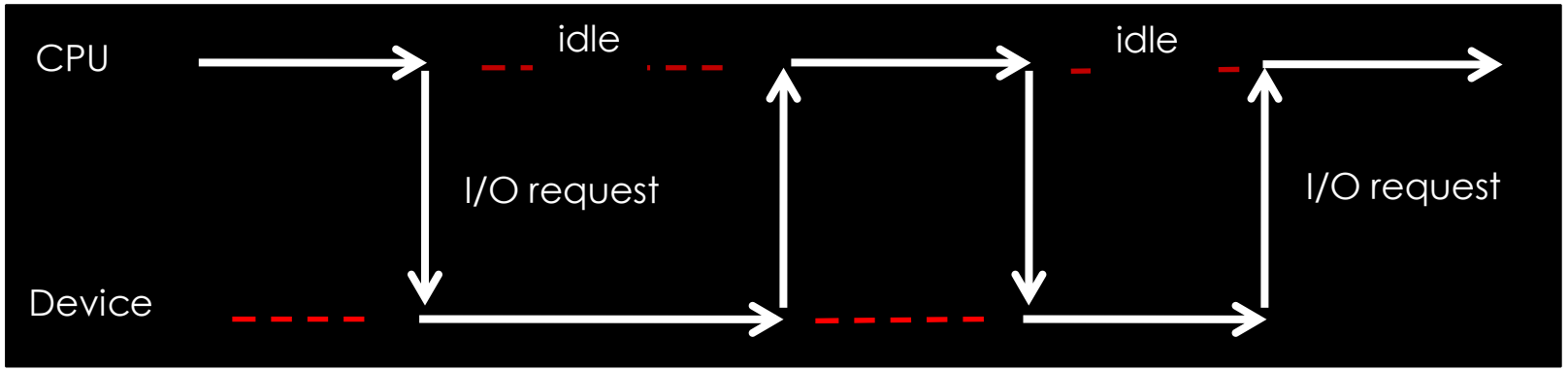
5. 操作系统结构&操作

1. 操作系统结构
   a) 批处理系统
   b) 多道程序系统
   c) 分时系统(多任务系统)

2. 操作系统操作
   双重模式操作(Dual Mode Operation)

批处理系统 Batch Processing System

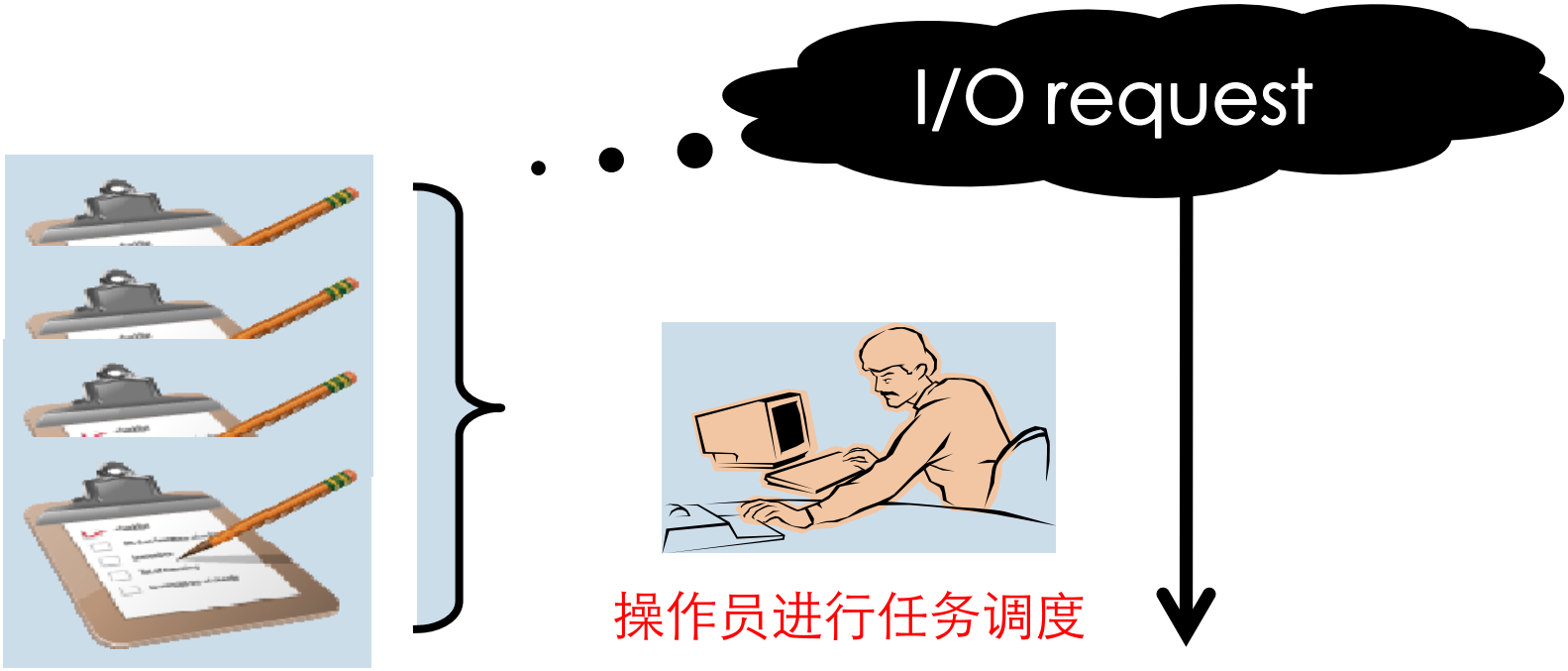
单处理器系统工作流程



当CPU发出I/O请求，设备进行I/O操作时，CPU就会空闲（idle）

问题是 I/O 操作相对于CPU操作速度很慢，导致CPU空闲，而CPU是非常昂贵的资源

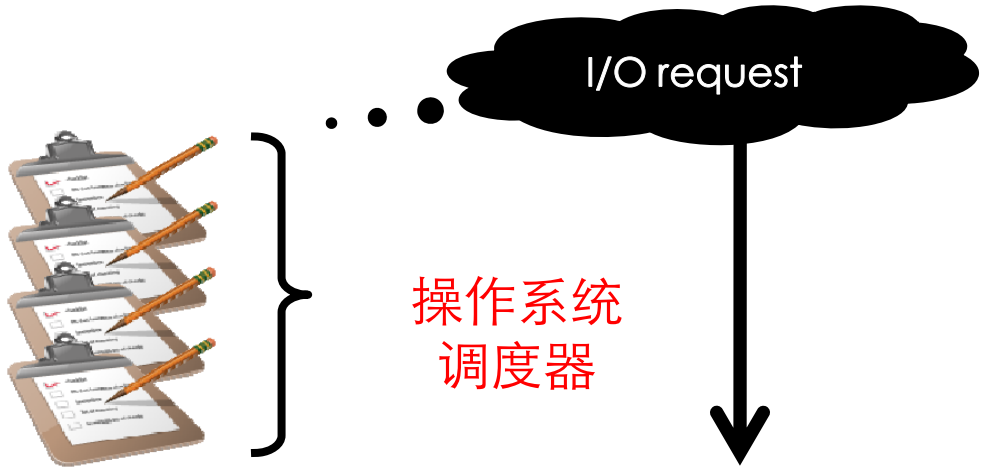
批处理系统



当发生I/O操作时，由操作员调度另一个程序运行，从而提高CPU的使用率。

多道程序系统 Multiprogramming System

1. 我们把操作员的工作内容写到操作系统里，由操作系统进行任务的调度，可不可以?
2. 当系统进行I/O操作时，调度器就会选择另一个任务并运行
3. 为保障CPU总有任务运行，多道程序系统必须把需要运行的多个任务载入到内存以备选择并运行
4. 需要调度器通过一定的机制选择任务并运行



可能出现的问题?

分时系统(多任务系统:multitasking)

多用户交互式系统，需要对每个用户的响应时间应小于1秒钟

Q: 如何去确保这个响应时间?

Solution: multitasking（多任务）

给每个用户赋予一个给定的时间片(time slot).

CPU 在多用户之间相互切换，如20 msec为单位切换另一个用户

But need Job Scheduling(CPU scheduling)

多道程序系统 vs 多任务系统

目的不同

多道程序系统：为提高CPU的使用率，让CPU忙

多任务系统：让每个任务能公平使用CPU，体现公平性

Q：那老师，现在的操作系统是批处理系统，还是多道程序系统，还是多任务系统呀?

5.2 操作系统操作

- 一个操作系统可以被多个用户、多个程序所共享
- 非法或不正确的操作会导致系统崩溃或破坏
- 一个程序的运行可能会影响另一个程序的运行，如无限循环有可能会导致系统失灵或破坏

操作系统需要有一个保护机制

Solution: Dual Mode Operation

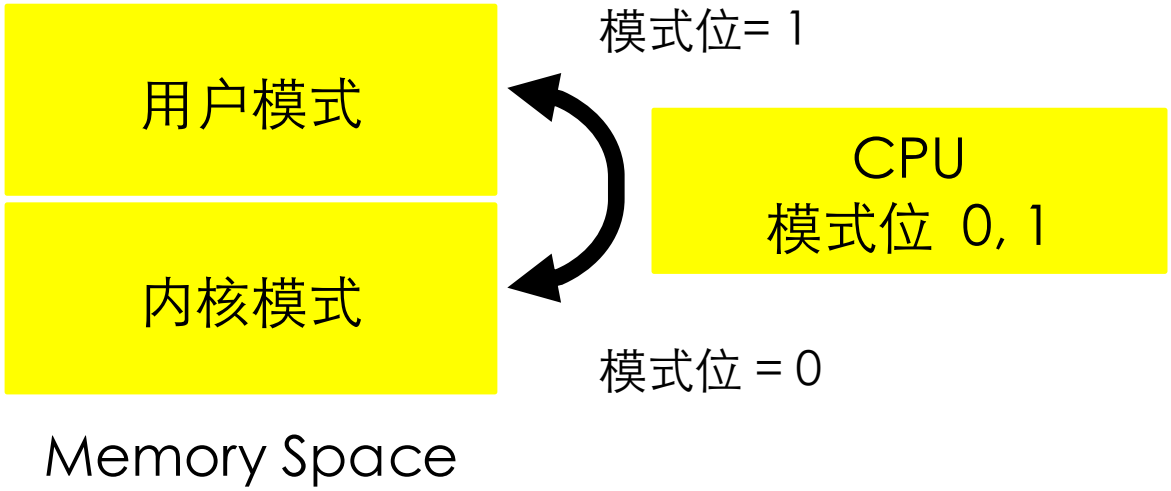
双重模式操作

5.2 操作系统操作

- 为了确保操作系统的正常执行，必须区分操作系统代码和用户系统代码的执行。
- 大部分采用的方法是提供硬件支持的双重模式操作，即用户模式(User Mode) 和内核模式(Kernel Mode)
- 硬件提供模式位（Mode bit ）来区分运行用户代码和系统代码，即用户模式和内核模式
  - 用户模式运行用户代码，表示为0
  - 内核模式运行系统代码，表示为1
- 用户模式时用户掌握计算机的控制权，内核模式时操作系统掌握计算机的控制权

5.2 操作系统操作

操作系统有双重运行模式：用户模式、内核模式，相互切换



管理模式：Supervisor mode,
系统模式：System mode
特权模式： Privileged mode
监督程序模式：monitor mode

5.2 操作系统操作

如系统调用，当用户程序调用系统调用函数的时候，操作模式从用户模式转变成系统模式

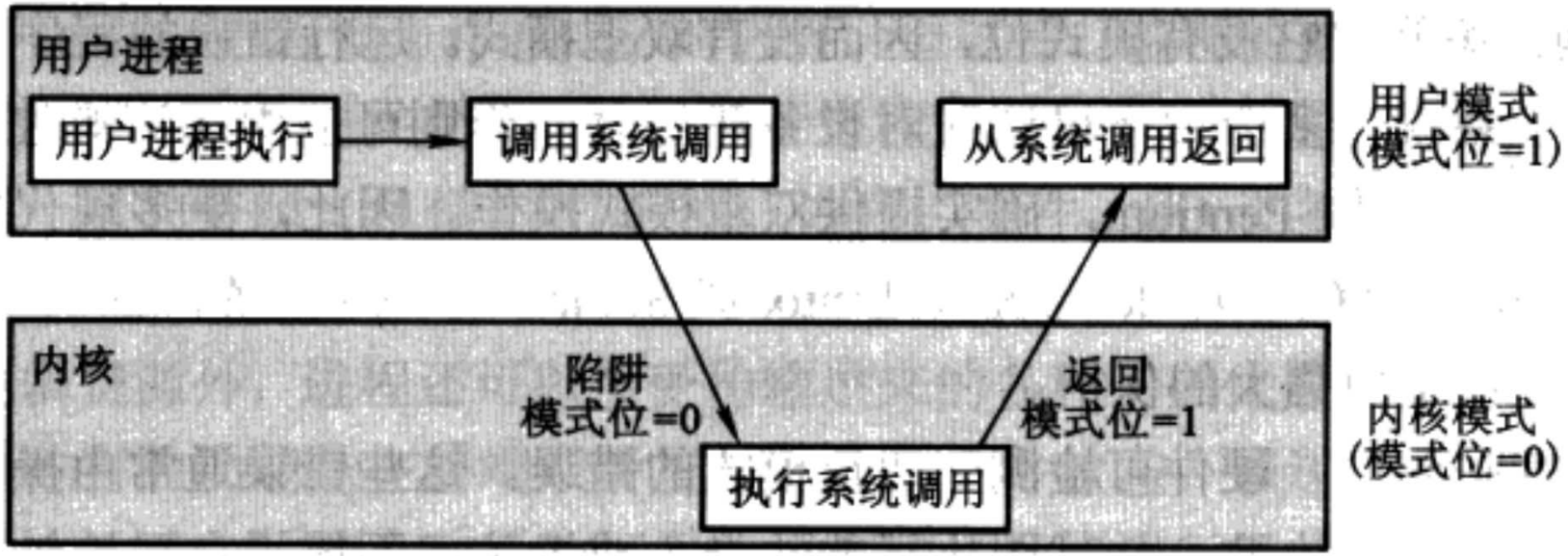


图 1.8　用户模式到内核模式的转换

第六节
操作系统管理

6. 系统管理

1. 进程管理
2. 内存管理
3. 存储管理
4. I/O管理
5. 网络管理
6. 安全管理
7. 等

6.1 进程管理

进程是运行中的程序，是系统的运行单元。

程序与进程的概念区别：程序是被动实体，而进程是活动实体

1. 在一个系统中有无数个进程在同时运行， 运行在一个或多个CPU上，进程之间通过复用CPU并发运行
2. 运行进程需要分配一定的资源，如CPU, 内存, I/O设备,文件等。当进程结束时，也应收回已分配的资源，即让进程有效使用这些资源

6.1 进程管理

进程的管理活动包括以下内容：

1. 创建和删除进程
2. 挂起和重启进程
3. 需提供进程同步机制
4. 需提供进程通讯机制
5. 需提供死锁处理机制
6. 等

6.2 内存管理

- 内存管理的主要目的就是提高内存的使用率，从而有效使用内存
- 管理内存中的数据的存储、指令的运行

内存管理活动包括以下内容：

1. 记录内存的哪些部分正在被使用及被谁使用
2. 当有内存有空闲空间时，决定哪些进程可以载入内存、载入到哪里等
3. 根据需要分配和释放内存空间，即分配的方式

6.3 存储管理：文件系统管理

为了便于使用计算机系统，操作系统提供了统一的逻辑信息存储观点，即文件系统

1. 操作系统对存储设备的物理属性进行了抽象的定义，即文件，它是存储的逻辑单元。
2. 计算机可以在多种类型的物理介质上存储信息
3. 每种介质通过一个设备来控制，如磁盘驱动器、磁带驱动器
4. 每个介质有不同的访问速度、容量、数据传输率和访问方法。
5. 文件通常组成目录以方便使用
6. 多用户访问文件时，需要控制权限问题

6.3 存储管理：文件系统管理

文件系统管理活动包括以下内容：

1. 创建和删除文件
2. 创建和删除目录来组织文件
3. 提供操作文件和目录的原语
4. 将文件映射到二级存储设备上
5. 在稳定存储介质上备份文件
6. 等

6.3 存储管理：大容量存储系统

Mass-Storage Management，一般为二级存储设备，如硬盘，它的管理活动包括以下内容：

1. 空闲空间的管理
2. 存储空间的分配
3. 硬盘调度

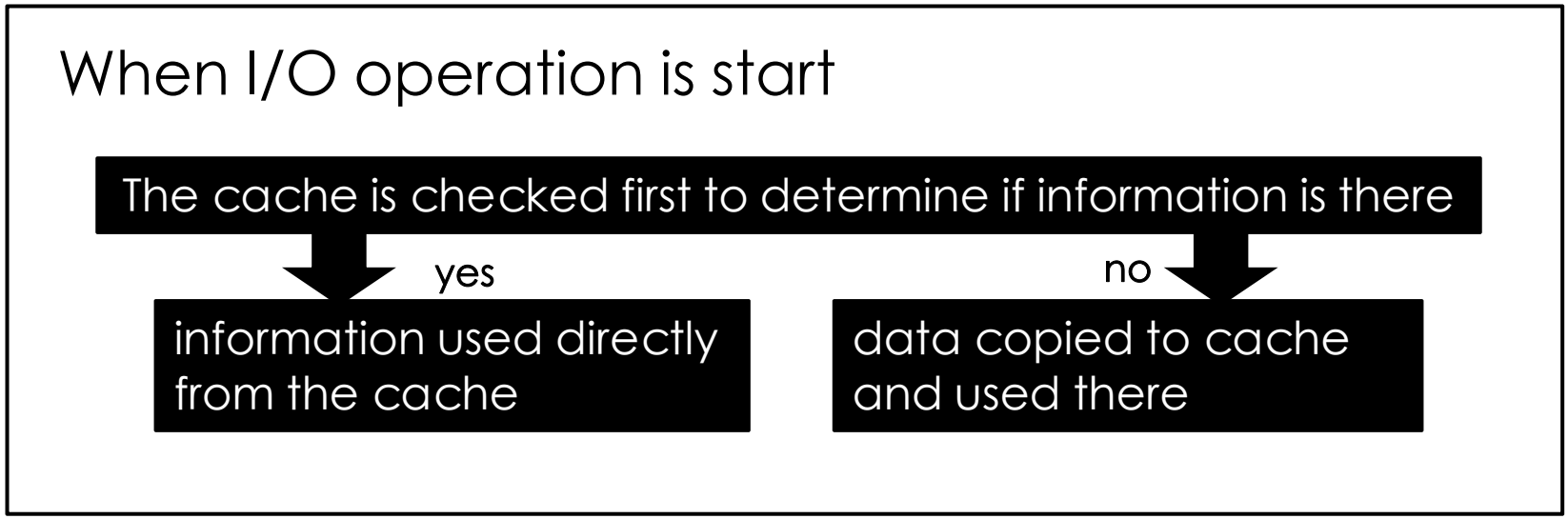
不同级别存储设备的性能比较

Storage hierarchy

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Cache（高速缓存）

1. 高速缓存是计算机系统的重要概念之一，是临时存储设备，一般设置在高速设备与低速设备之间。
2. 当高速设备从低速设备上读取数据时，会把数据临时复制到高速缓存上

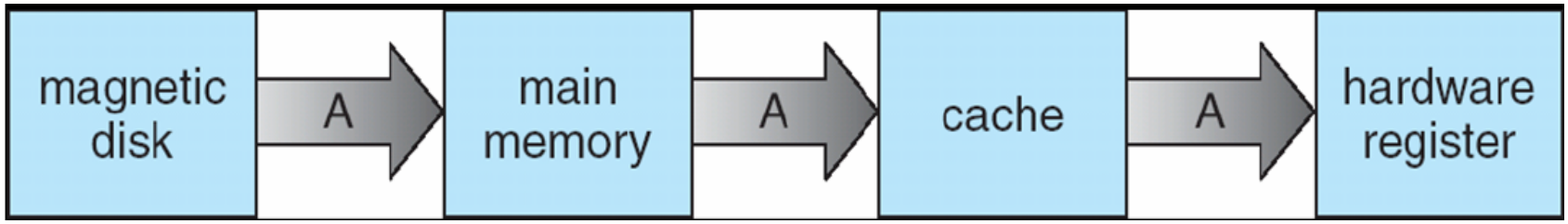


高速缓存

问：

1. 缓存能代替存储设备吗?
2. 而且缓存大小一般很小，缓存满了怎么办?
3. 一种解决办法是置换缓存里的内容，怎么置换?

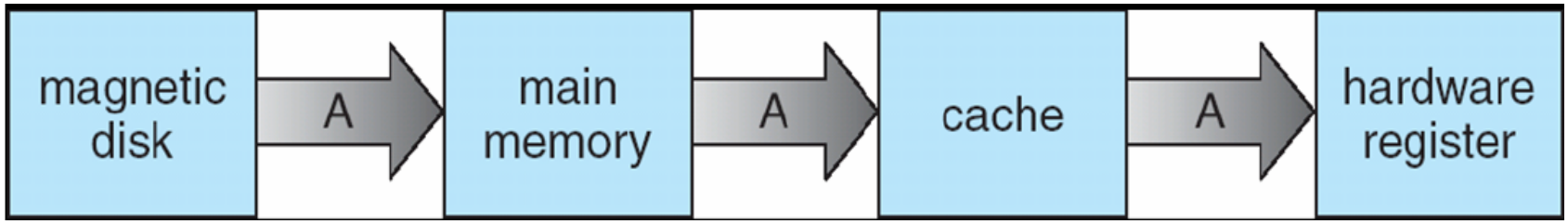
把整数A 从磁盘移动到寄存器



把磁盘上的A=1读到内存加一，并存储到硬盘的流程

1. 现从磁盘上读取A到内存上
2. 复制A到高速缓存上
3. 为了加一复制A到寄存器上
4. 加一后，写入磁盘上

把整数A 从磁盘移动到寄存器



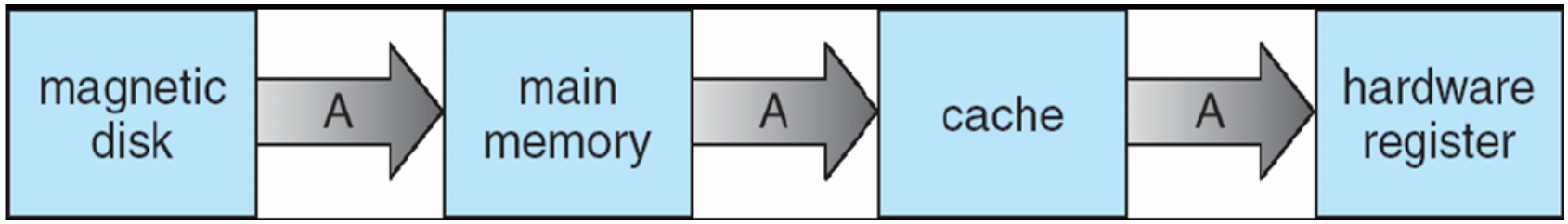
加一操作结束后，各个存储器内的A的值会不同。

磁盘：2、内存：1、缓存：1、寄存器：2

在多任务环境下，CPU切换到另一个进程读取A的时候，第一个先到高速缓存上读取，而缓存上的A的值不是最新值。

在多处理器环境下表现为更复杂，因为每个处理器都有自己的高速缓存和寄存器。

把整数A 从磁盘移动到寄存器



So, 在多任务环境下、多处理器环境下，分布式环境下必须要保证数据的一致性

Coherency 一致性问题

Solution：用硬件来解决

6.4 I/O 子系统

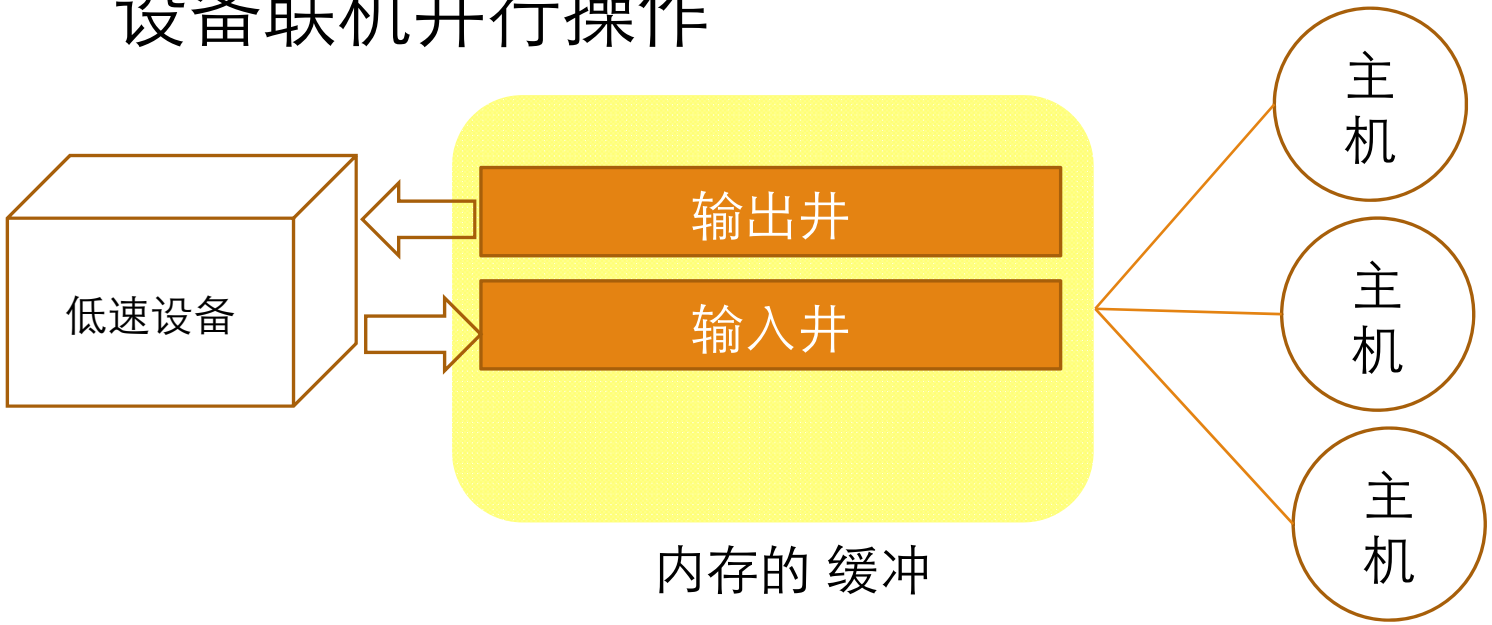
I/O子系统的目的是针对用户隐藏具体硬件设备的特性，它包括以下几个部分

1. 一个包括缓冲（buffer）、高速缓存（cache）和假脱机（spooling）的内存管理部分
2. 通用设备驱动器接口
3. 特定硬件设备的驱动程序

Notice that, In Linux/Unix, a device is accessed through the use of file management interface

Tips

1. 缓冲(Buffering)：为传输数据暂时存储数据
2. 缓存(Caching)：为性能提高暂时存储数据
3. 假脱机(Spooling)：是关于低速字符设备与计算机主机交换信息的一种技术，又称外部设备联机并行操作



Tips

假脱机技术的优点

1. 提高I/O速度
2. 设备并没有单独分配给任何一个任务
3. 实现了虚拟设备功能，从而可以共享设备

7. 其他计算机系统

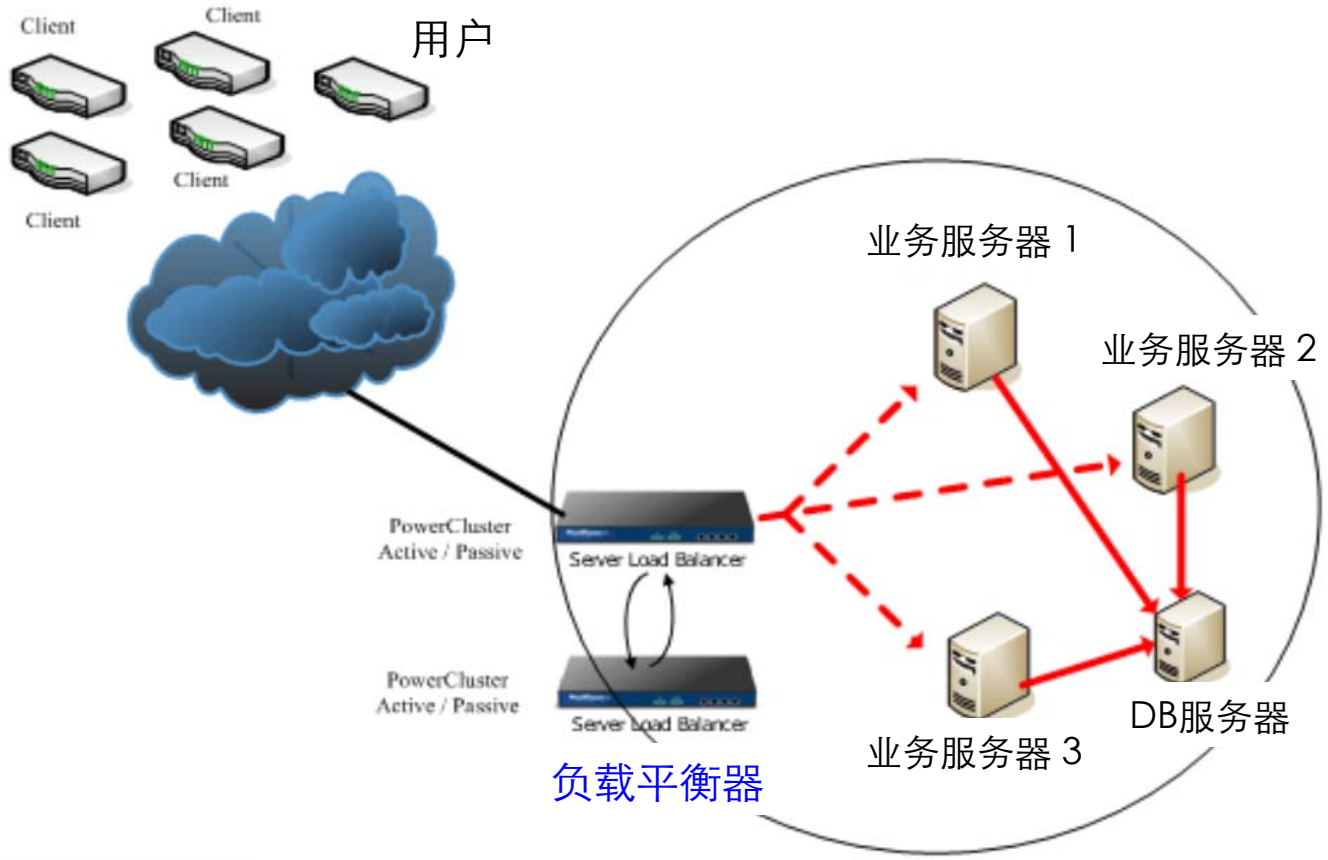
- 分布式系统 Distributed Systems – 是将物理上分开的、各种可能异构的计算机系统通过网络连接在一起，为用户提供系统所维护的各种资源的计算机集合
- 其他专用系统， 如实时系统(real-time system)，嵌入式系统(embedded system), 实时嵌入式系统, 多媒体系统, 手持系统

- 实时系统指的是系统中的任务有时间节点(截止时间)的系统，如军事上、医疗上使用的系统

7. 其他计算环境

1. 客户机/服务器计算（C/S computing）
   - 计算服务器系统
   - 文件服务器系统

   存在瓶颈问题 bottle-neck problem
2. 点对点计算（peer-to-peer computing）
   - 又是客户机、又是服务器
   - 复杂
3. 基于Web 计算（Web-based computing）
   - 负载平衡器(load balancer)


Client
Client
Client
Client
Client
用户
业务服务器 1
业务服务器 2
PowerCluster
Active / Passive
Server Load Balancer
PowerCluster
Active / Passive
Server Load Balancer
业务服务器 3
DB服务器
负载平衡器

Q&A