南京大学
NANJING UNIVERSITY

系统互联及输入输出组织

殷亚凤
智能软件与工程学院
苏州校区南雍楼东区225
yafeng@nju.edu.cn，https://yafengnju.github.io/

系统互联及输入输出组织

- **外部设备的分类与特点**
- 常用输入输出设备
- 外设与CPU和主存的互连
- I/O数据传送控制方式
- 内核空间I/O软件

外设的分类

- **按信息的传输方向来分：**
  - **输入设备**：键盘、鼠标、扫描仪等
  - **输出设备**：打印机、显示器等
  - **输入/输出设备**：磁盘驱动器、光盘驱动器、CRT终端、网卡之类的通信设备等。
- **按功能来分：**
  - **人机交互设备**：用于用户和计算机之间交互通信的设备。
    (如：键盘、鼠标、扫描仪、打印机、显示器等)
  - **存储设备**：用于存储大容量数据，作为计算机的外存储器使用。
    (如：磁盘驱动器、光盘驱动器等)
  - **机-机通信设备**：用于计算机及和计算机之间的通信。
    (如：网卡、调制解调器、数/模和模/数转化设备等)

外设的特点

- **异步性**：外设与CPU之间是完全异步的工作方式，两者之间无统一的时钟。
- **实时性**：CPU必须及时按不同的传输速率和不同的传输方式接收来自多个外设的信息或向多个外设发送信息，否则高速设备可能丢失信息。
- **多样性**：外设的多样性造成了主机与外设之间连接的复杂性。为简化控制，计算机系统往往提供标准接口，以便各类外设通过自己的设备控制器与标准接口相连。

系统互联及输入输出组织

- 外部设备的分类与特点
- **常用输入输出设备**
- 外设与CPU和主存的互连
- I/O数据传送控制方式
- 内核空间I/O软件

常用输入输出设备：键盘

- 通过键盘输入字符，并在显示器上显示
- 信息交换的基本单位是字符
- **普遍使用的代码是ASCII码**

  (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)

  – 共有128个元素，其中有32个通用控制字符，10个十进制数码，52个英文字母，34个专用符号。除了32个控制字符外，其余96个全部是可打印字符。

  – 通常在7位代码后跟一位奇偶校验位，组成一个字节。

常用输入输出设备：键盘

- **键盘输入信息过程**
  - 按下一个键
  - 查出按下的是哪个键：**列扫描法**等
  - 将按键位置信息转换为对应的ASCII码，保存到计算机中。
- **按功能可分为以下两类：**
  - **编码键盘**：检测被按键，并提供相应的ASCII码送CPU。
  - **非编码键盘**：只简单提供键盘的行列矩阵，识别按键位置，提供相应的位置码送CPU。

常用输入输出设备：键盘

- **按键位置识别法之一：列扫描法**

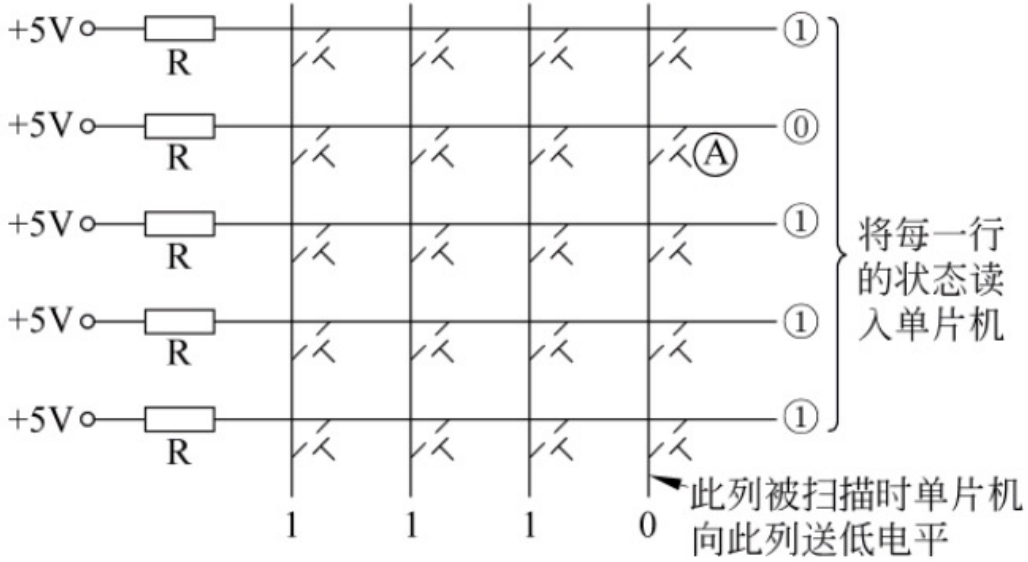


图 8.1　按键矩阵与扫描码的形成

- **基本做法：按列扫描、接地检查**

常用输入输出设备：打印机

- **击打式打印机：**最早研制成功的计算机打印设备，以机械力量击打字锤从而使字模隔着色带在纸上打印出字来。
  - **整字形打印设备**
  - **点阵打印设备（针式打印机）**：利用打印头中的多根印针经色带在纸上打印出点阵字符的印字设备。
- **非击打式打印机：**
  - **激光打印机**：由打印机控制器和打印装置两部分组成，是目前应用最广泛的一种非击打式打印机。
  - **喷墨打印机**：利用喷墨头喷射出可控的墨滴，在打印纸上形成文字或图片，是目前应用较多的一种打印输出设备。

打印机与主机的连接

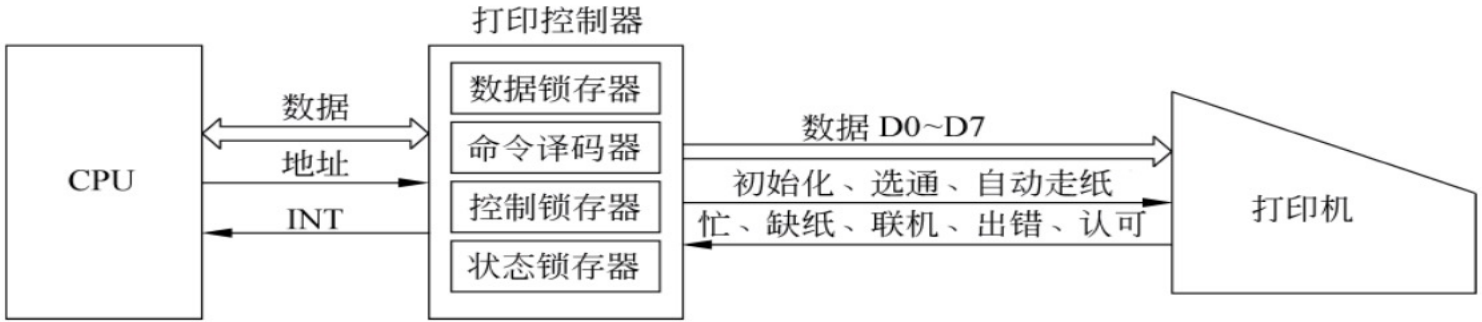


图 8.2　打印机与主机的连接

打印机通过打印控制器或打印适配器与主机连接，打印控制器由以下基本部件组成：

- **数据锁存器**：用于暂存CPU送来的打印数据。
- **命令译码器**：对CPU送来的命令进行译码，产生打印控制器内部使用的命令。
- **控制锁存器**：锁存CPU送来的控制命令。
- **状态锁存器**：保存打印机送来的状态信息。

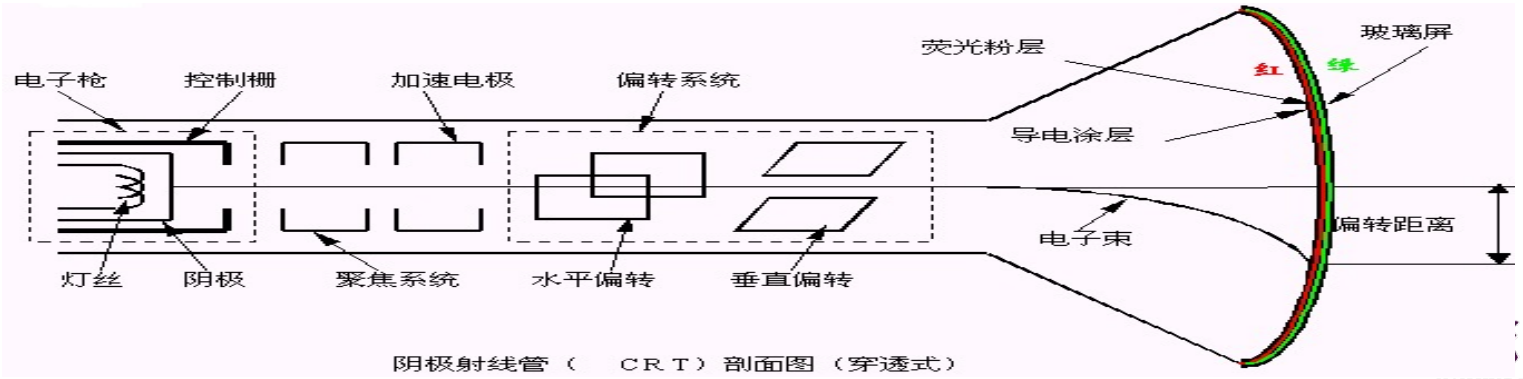
常用输入输出设备：显示器

- **显示设备分类：**
  - **按显示器件分：**
    - 阴极射线管（Cathode Ray Tube, 简称CRT ）显示器
    - 液晶显示器（Liquid Crystal Display, 简称LCD）
    - 等离子显示器 等
  - **按显示内容分：**
    - 字符显示器、图形显示器和图像显示器
  - **按功能分：**
    - 普通显示器 、终端设备显示器
  - **按扫描方式分：**
    - 光栅扫描显示器 、随机扫描显示器
  - **按分辨率高低分：**
    - 高分辨率显示器、低分辨率显示器

CRT显示器

阴极射线管（CRT）

- CRT(Cathode-Ray Tube)是图形显示器的核心。电视机中的显像管就是CRT。
- **基本工作原理**：由电子枪发出的电子束(阴极射线)，通过聚焦系统和偏转系统，射向涂覆荧光层的屏幕上的指定位置。在电子束冲击的每个位置，荧光层发出一个小的亮点，由这些小亮点构成所需的字符、图形和图像。
- 对彩色CRT而言，通常用三个电子枪发射电子束，经定色机构，分别触发红、绿、蓝三种颜色的荧光粉发光，按三元色叠加原理形成彩色图像。



阴极射线管（ CRT）剖面图（穿透式）

CRT显示器

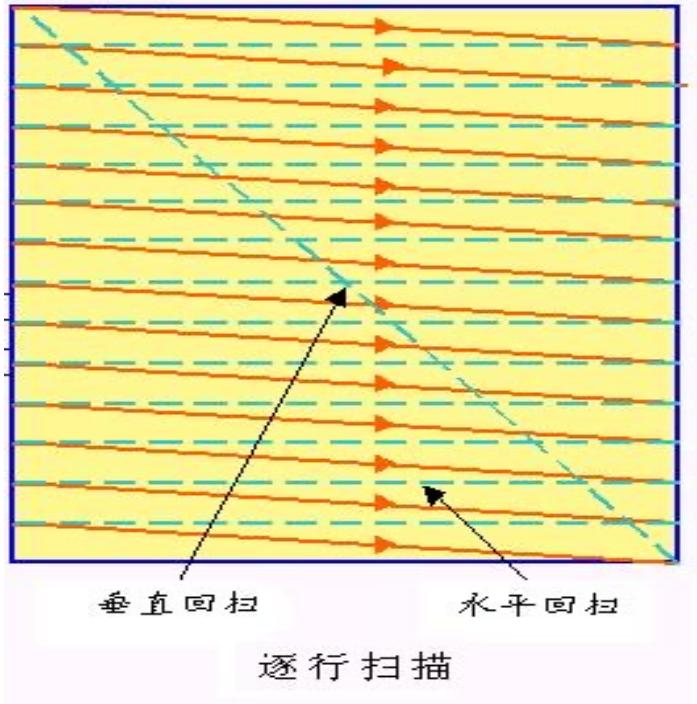
- **分辨率和灰度级**
  - CRT荧光屏在水平方向和垂直方向单位长度上能识别的最大光点数称为**分辨率**。也即：CRT的分辨率是可以无重叠显示的最多点数。
  - 通常称分辨率为水平和垂直方向的总点数，但更精确的分辨率定义是在x和y方向上每厘米可绘制的点数(水平分辨率和垂直分辨率)
  - **灰度级**是指像素点的亮度级差，在彩色显示器中表现为色彩的差别,即颜色数。真彩色显示器的颜色位数为： 24位=8位(红)+ 8位(绿)+ 8位(蓝)，所以颜色数为：2^{24} 。
- **纵横比(aspect ratio)**
  - 指在屏幕两个方向生成同等长度的线段所需垂直点数对水平点数的比值(有时，纵横比解释为水平点数对垂直点数的比值)。例如：纵横比为3/4则意味着垂直方向上画三点的长度与水平方向上画四点的长度相同。
- **物理尺寸**
  - 指屏幕对角线的长度。

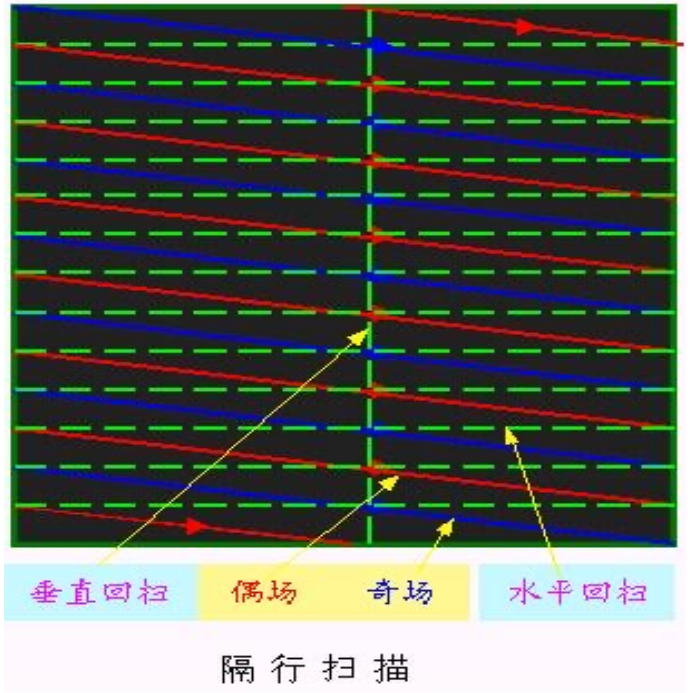
CRT显示器

- **光栅扫描**
  - 光栅扫描方式中，电子束总是不断地从左到右、从上到下反复扫描整个屏幕，
  - 电子束从左到右（横向）扫描一次称为一条扫描线，从屏幕顶部到屏幕底部的所有扫描线构成一帧图像。也就是说，一帧图像是光栅显示系统执行一次循环或屏幕刷新一次所产生的图像。
  - 为了产生无明显闪烁的图像，每秒钟至少执行30次循环。
  - 一般刷新是按每秒60到80帧的速率进行的，但有些系统设计成更高的刷新速率。每秒60帧的刷新频率为60HZ 。
  - 可分为**逐行扫描**和**隔行扫描**两种，一般都采用隔行方式。每帧显示分为先后扫描奇数场和偶数场两趟。

CRT显示器

- **光栅扫描**





LCD液晶显示器

- 具有体积小、耗电低、不闪烁等优点和良好的综合性能，广泛应用于各类计算设备中。
- **基本原理**：液晶通电时会改变其排列次序，从而影响光线的通过。
- **工作模式**：
  - **字符模式**：显示存储器中存放字符编码及其属性，字形信息存放在字符发生器中。
  - **图形模式**：字符的点阵信息直接存储在显示存储器中，可显示彩色或单色多级灰度图像，能够实现画图功能。
- **显卡**：核心是绘图处理器。早期的绘图功能由CPU在内存中完成，再将生成的图像从内存传到显存。目前显卡中的GPU专门用来绘图，减轻了CPU的负担。

系统互联及输入输出组织

- 外部设备的分类与特点
- 常用输入输出设备
- **外设与CPU和主存的互连**
- I/O数据传送控制方式
- 内核空间I/O软件

总线的基本概念

- **总线**是计算机内数据传输的公共路径，用于实现两个或两个以上部件之间的信息交换。
- **系统总线**指连接处理器芯片、存储器芯片和各种I/O模块等主要部件的总线。
- **系统总线**通常由一组控制线、一组数据线和一组地址线构成。也有些总线没有单独的地址线，地址信息通过数据线来传送，这种情况称为数据/地址复用。
  - **数据线**（Data Bus）：承载在源和目部件之间传输的信息。数据线的宽度反映一次能传送的数据的位数。
  - **地址线**（Address Bus）：给出源数据或目的数据所在的主存单元或I/O端口的地址。地址线的宽度反映最大的寻址空间。
  - **控制线**（Control Bus）：控制对数据线和地址线的访问和使用。用来传输定时信号和命令信息。典型的控制信号包括：
    - 时钟（Clock）：用于总线同步。
    - 复位（Reset）：初始化所有设备。
    - 总线请求（Bus Request）：表明发出该请求信号的设备要使用总线。
    - 总线允许（Bus Grant）：表明接收到该允许信号的设备可以使用总线。
    - 中断请求（Interrupt Request）：表明某个中断正在请求。
    - 中断回答（Interrupt Acknowledge）：表明某个中断请求已被接受。
    - 存储器读（memory read）：从指定的主存单元中读数据到数据总线上。
    - 存储器写（memory read）：将数据总线上的数据写到指定的主存单元中。
    - I/O读（I/O read）：从指定的I/O端口中读数据到数据总线上。
    - I/O写（I/O Write）：将数据总线上的数据写到指定的I/O端口中。
    - 传输确认（transmission Acknowledge）：表示数据已被接收或已送总线

总线的性能指标

- **总线宽度**
  - 总线中数据线的条数，决定了每次能同时传输的信息位数。
- **总线工作频率**
  - 早期的总线通常一个时钟周期传送一次数据，此时，工作频率等于总线时钟频率；现在有些总线一个时钟周期可以传送2次或4次数据，因此，工作频率是时钟频率的2倍或4倍。
- **总线带宽**
  - 总线的最大数据传输率
  - 对于同步总线，总线带宽计算公式： $B=W\times F/N$

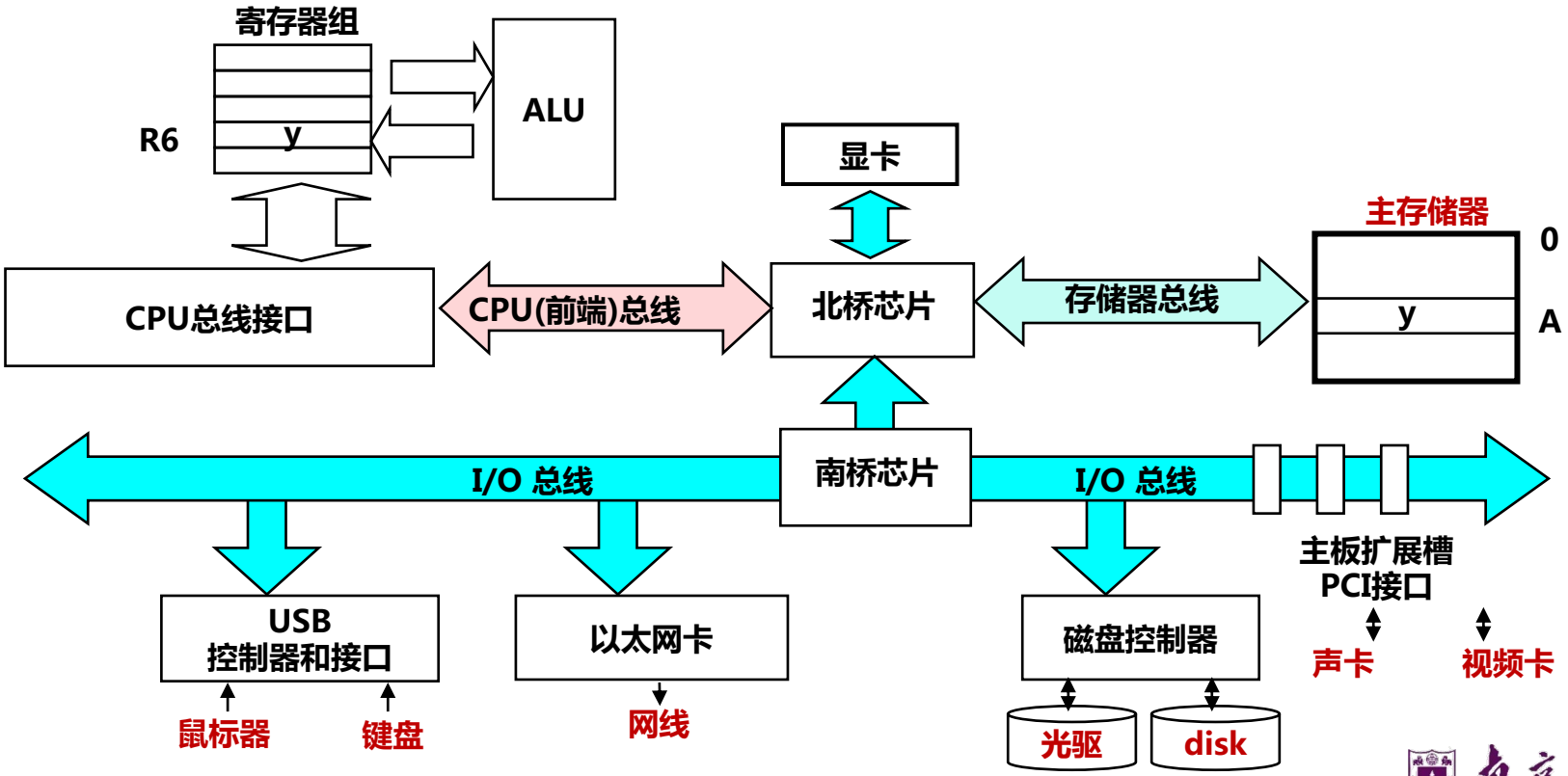
    W-总线宽度；F-总线时钟频率；N-完成一次数据传送所用时钟周期数。

    F/N实际上就是总线工作频率
- **总线寻址能力**
  - 由地址线位数所确定的可寻址地址空间的大小。

总线的性能指标

- **总线定时方式**
  - 同步通信：由时钟信号同步
  - 异步通信：前一个信号的结束就是下一个信号的开始，信息的改变是顺序的
  - 半同步通信：同步和异步两种总线定时方式的结合
- **总线传送方式**
  - 非突发传送：每个总线事务都传送地址，一个地址对应一次数据传送。
  - 突发传送：即为成块数据传送。突发传送总线事务中，先传送一个地址，后传送多次数据，后续数据的地址默认为前面地址自动增量。
- **总线负载能力**
  - 总线上所能挂接的遵循总线电气规范的总线设备的数目(一般指总线上扩展槽的个数)

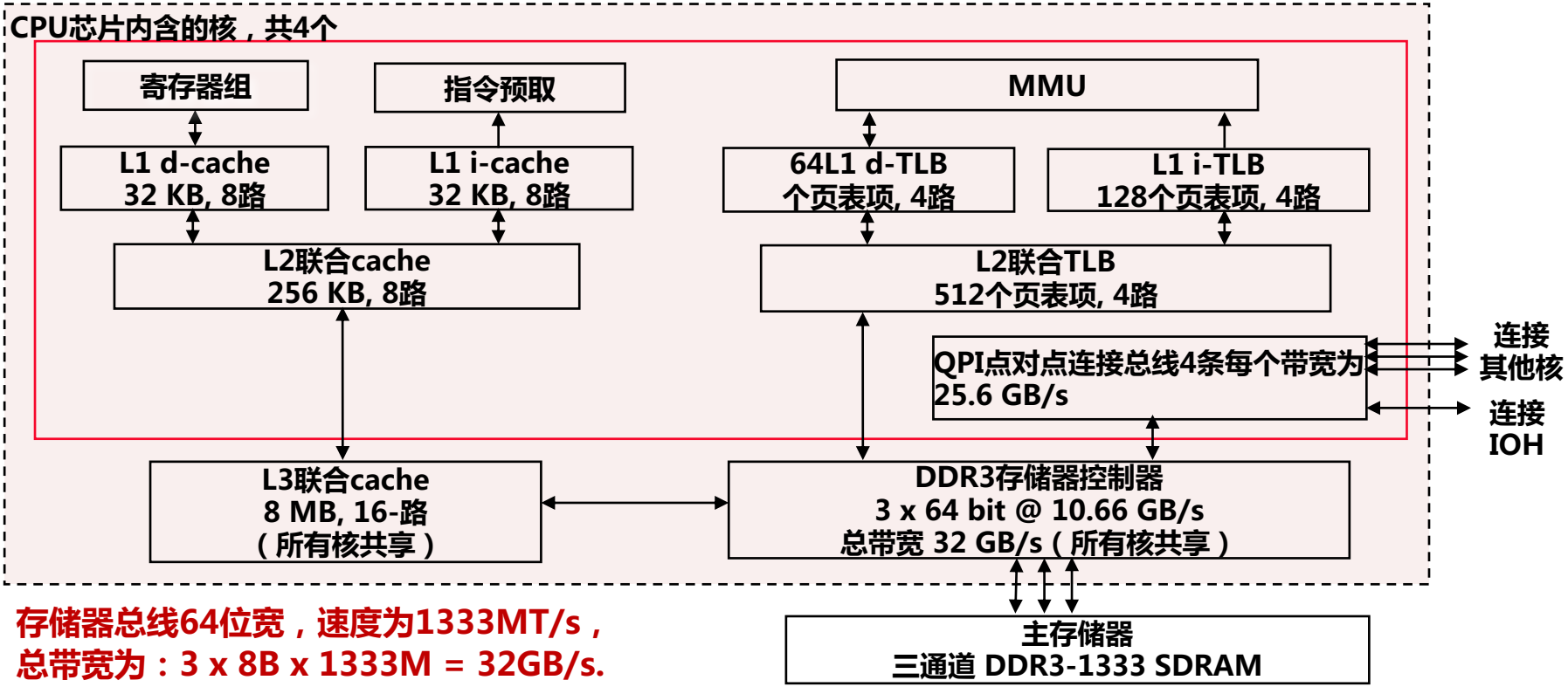
基于总线的互连结构


寄存器组
R6
y
ALU
显卡
主存储器
0
A
y
CPU总线接口
CPU(前端)总线
北桥芯片
存储器总线
I/O 总线
南桥芯片
I/O 总线
主板扩展槽
PCI接口
USB
控制器和接口
以太网卡
磁盘控制器
声卡
视频卡
鼠标器
键盘
网线
光驱
disk

处理器总线

- **前端总线（Front Side Bus，FSB）**
  - 并行传输、同步定时方式
  - 早期Intel架构使用，位于CPU芯片与北桥芯片之间互连
  - 从Pentium Pro开始，FSB采用quad pumped技术：每个总线时钟周期传送4次数据。
  - 若工作频率为1333MHz（实际单位应是MT/s，表示每秒传送1333M次数据，实际时钟频率为333MHz），总线宽度为64位，则总线带宽为1333MT/s×8B=10.5GB/s。
- **快速通道互连QPI（Quick Path Interconnect）总线**
  - 目前在Intel架构中CPU芯片内部核之间、CPU芯片之间、CPU芯片与IOH（I/O Hub）芯片之间，都通过QPI总线互连
  - QPI是基于包交换的串行、高速点对点连接：发送方和接收方各有时钟信号，双方同时传输数据（各有20条数据线），每个QPI数据包含80位，分两个时钟周期传送，每个时钟周期传两次，故每次传20位（16位数据+4位校验位），QPI总线带宽为：每秒传送次数×2B×2。
  - QPI总线的速度单位（工作频率）为GT/s，表示每秒传送多少G次。若QPI时钟频率为2.4GHz，则速度为4.8GT/s，带宽为4.8G×2B×2=19.2GB/s.

存储器总线



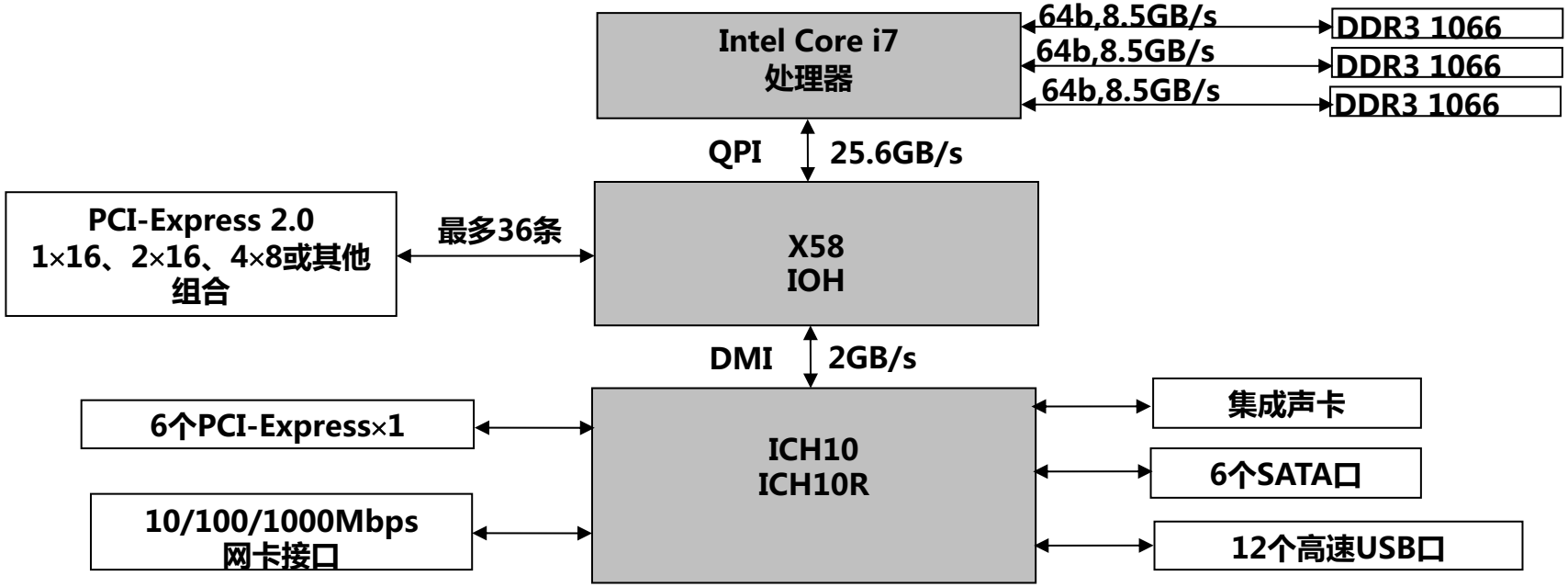
存储器总线64位宽，速度为1333MT/s，
总带宽为：3 x 8B x 1333M = 32GB/s.

从Core i7开始，北桥在CPU芯片内，CPU通过存储器总线（即内存条插槽，图中为三通道插槽）直接和内存条相连。3个存控包含在CPU芯片内。

I/O总线

- **I/O总线用于为系统中的各种I/O设备提供输入输出通道**
- **I/O总线在物理上可以是主板上的I/O扩展槽，**如：
  - 第一代：ISA/EISA总线、VESA总线，早被淘汰
  - 第二代：PCI、AGP、PCI-X，被逐渐淘汰
  - **第三代：PCI-Express（串行总线，主流总线）**
- **PCI-Express总线**
  - 两个PCI-Express设备之间以一个链路（link）相连
  - 每个链路包含多条通路（lane），可以是1,2,4,8,16或32条
  - PCI-Express×n表示一个具有n条通路的PCI-Express链路
  - 每条通路可同时发送和接受，每个数据字节被转换为10位信息被传输
  - PCI-Express1.0下，每条通路的发送和接受速率都是2.5Gb/s，故PCI-Express×n的带宽为：2.5Gb/s×2×n/10=0.5GB/s×n。
  - PCI-Express1.0下，PCI-Express×2的带宽为1GB/s，PCI-Express×4的带宽为2GB/s，PCI-Express×16的带宽为8GB/s.

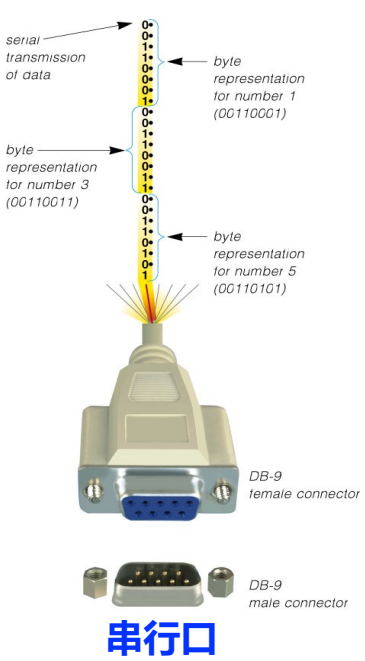
基于Core i7系列处理器的互连结构举例

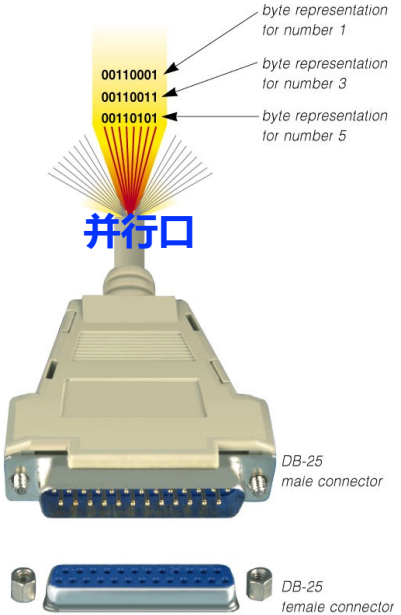


- ✓ QPI总线的带宽为：6.4GT/s×2B×2=25.6GB/s
- ✓ 每个存储器总线的带宽为：64b/8×1066 MT/s = 8.5 GB/s .

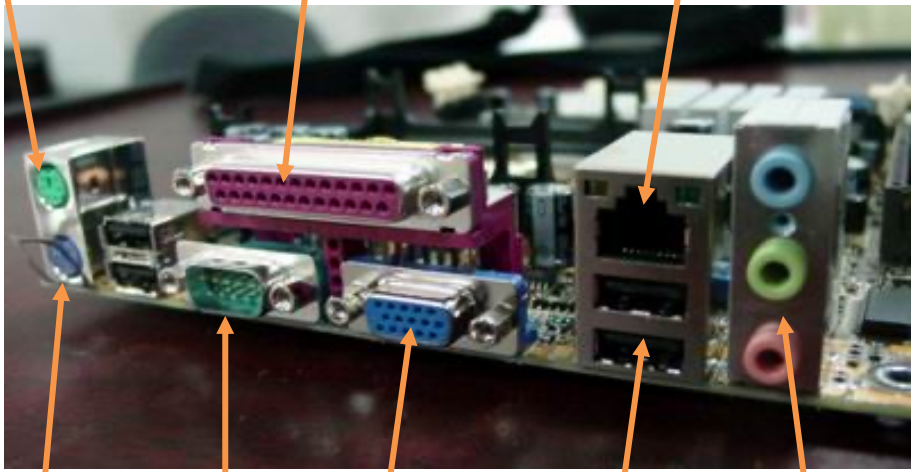
I/O接口

- **I/O接口：**I/O设备控制器（如网卡、显卡、键盘适配器、磁盘控制器）
  包括：插头 / 插座的形式、通讯规程和电器特性等
- **分类：**
  - **从数据传输方式来分**：
    - 串行（一次只传输1位）
    - 并行（多位一起进行传输）
  - **从是否能连接多个设备来分**：
    - 总线式（可连接多个设备）
    - 独占式（只能连接1个设备）
  - **从是否符合标准来分**：
    - 标准接口 （通用接口）
    - 专用接口 （专用接口）
  - **按功能选择的灵活性来分**：
    - 可编程接口
    - 不可编程接口

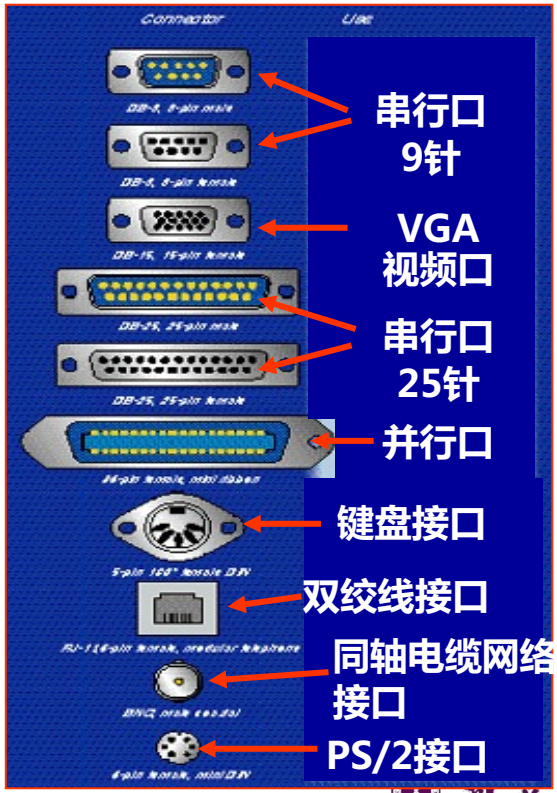




I/O设备接口插座（连接器）



(安装在主板上的I/O设备接口插座)



I/O接口的职能

- **数据缓冲**

  提供数据缓冲寄存器，以达到主机和外设工作速度的匹配。

- **错误或状态检测**

  提供状态寄存器，以保存各种错误或状态信息供CPU查用。

- **控制和定时**

  提供控制和定时逻辑，以接受从系统总线来的控制定时信号。

- **数据格式转换**

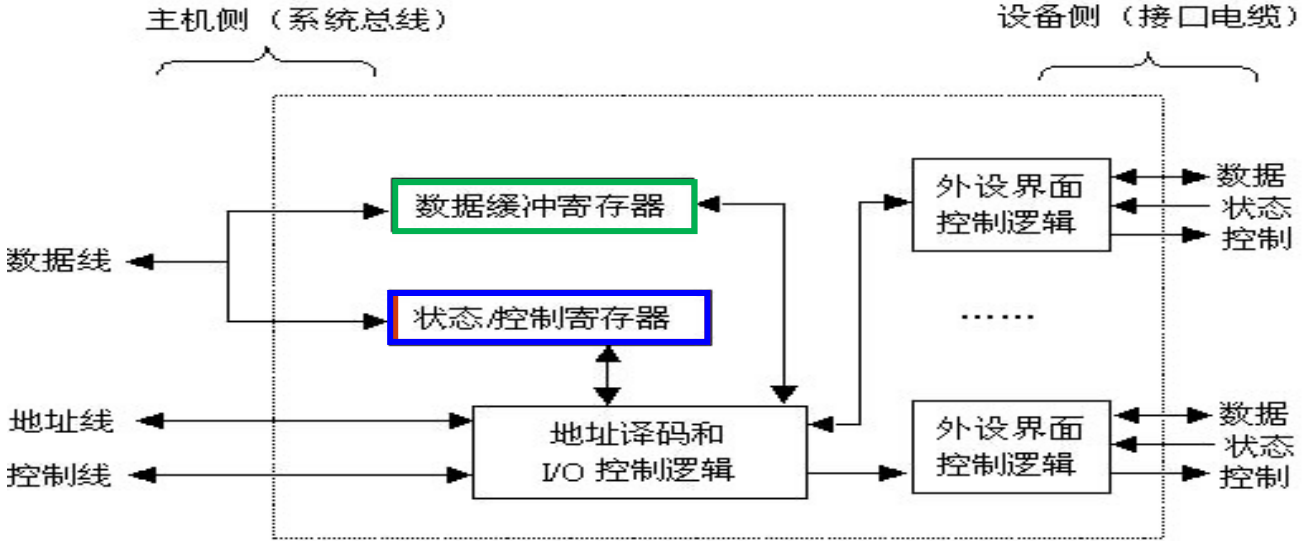
  提供数据格式转换部件使通过外部接口得到的数据转换为内部接口需要的格式，或在相反的方向进行数据格式转换。

- **与主机和设备通信**

  上述功能通过I/O接口与主机之间、I/O接口与设备之间的通信来完成。

I/O接口的通用结构

- **I/O控制器的一般结构：不同I/O模块在复杂性和控制外设的数量上相差很大**



- 通过发送命令字到I/O控制寄存器来**向设备发送命令**
- 通过从状态寄存器读取状态字来**获取外设或I/O控制器的状态信息**
- 通过向I/O控制器发送或读取数据来**和外设进行数据交换**

将I/O控制器中CPU能够访问的各类寄存器称为I/O端口，对外设的访问通过向I/O端口发命令、读状态、读/写数据来进行

I/O端口及其编址

- 对I/O端口读写，就是向I/O设备送出命令或从设备取得状态或读/写设备数据
- 一个I/O控制器可能会占有多个端口地址
- I/O端口必须编号后，CPU才能访问它
- I/O设备的寻址方式就是I/O端口的编号方式

（1）统一编址方式（内存映射方式）

➢ 与主存空间统一编址，将主存空间分出一部分地址给I/O端口进行编号。

（该方法是将I/O端口映射到某主存区域，故也称为“存储器映射方式”）

例如，RISC架构（如MIPS、RISC-V等）、Motorola公司的处理器等采用该方案

（2）独立编址方式（特殊I/O指令方式）

➢ 不和主存单元一起编号，而是单独编号，使成为一个独立的I/O地址空间

（因需专门I/O指令，故也称为“特殊I/O指令方式”）

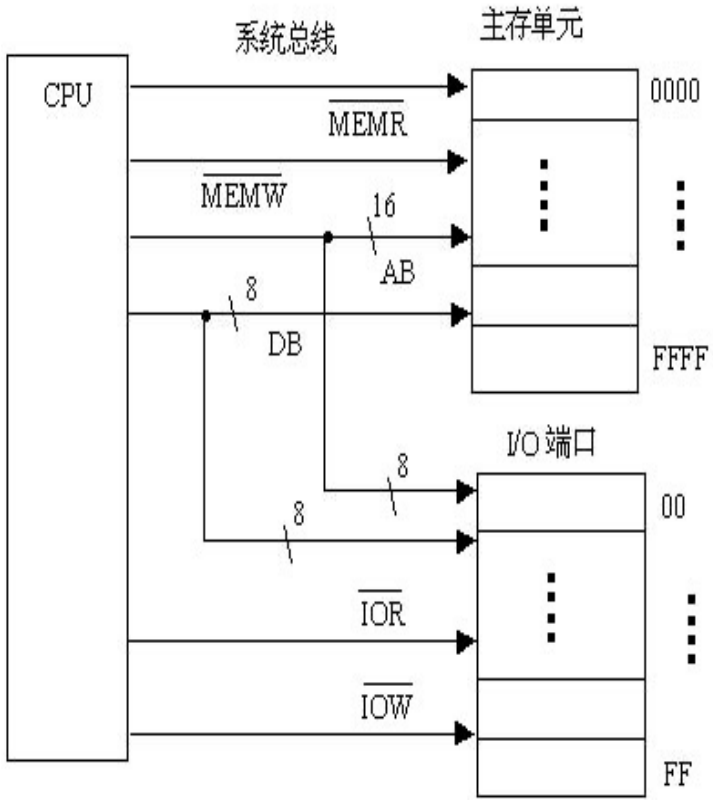
例如，Intel公司和Zilog公司的处理器就是独立编址方式

独立编址方式

- 通过不同的读写控制信号$\overline{IOR}$、 $\overline{IOW}$和$\overline{MEMR}$、 $\overline{MEMW}$来控制对I/O 端口和存储器的读写。
- 一般I/O端口比存储器单元少，所以选择I/O端口时，只需少量地址线。
- 指令系统必须设计专门的I/O指令。

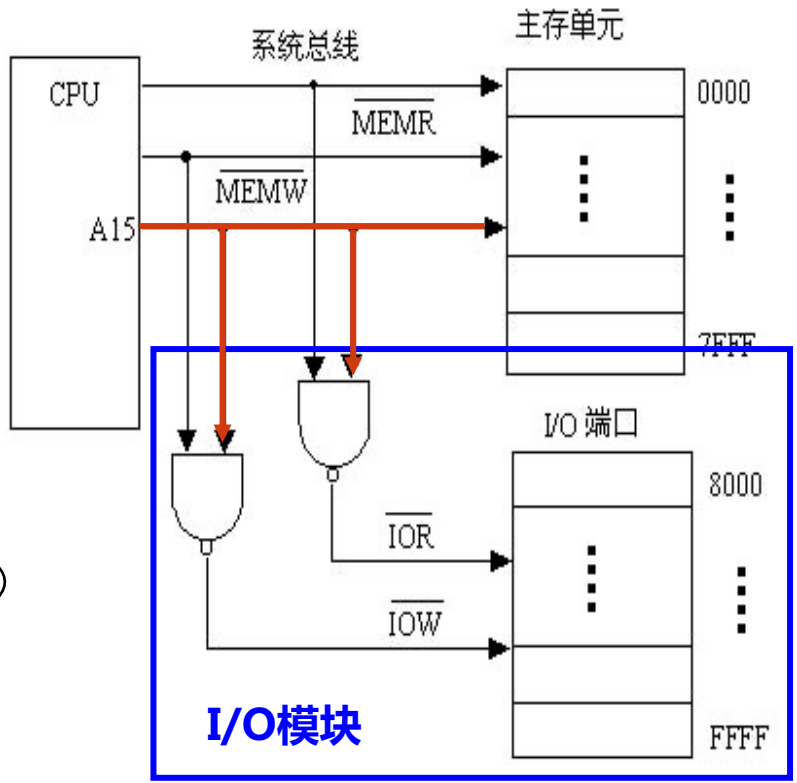
$\overline{MEMR}$或$\overline{MEMW}$命令由访存指令发出，$\overline{IOR}$和$\overline{IOW}$命令怎样呢？

由专门的I/O指令确定，指令中给的地址可能相同，但操作命令不同！



统一编址方式

- CPU不直接通过读写控制信号$\overline{IOR}$、$\overline{IOW}$对I/O端口读写，而是根据I/O端口在地址空间的位置，通过地址译码来实现。
- 地址线的高位参与片选控制逻辑。
- 无需设置专门I/O指令，只要用一般访存指令就可存取I/O端口。



$\overline{MEMR}$或$\overline{MEMW}$命令由访存指令发出，$\overline{IOR}$和$\overline{IOW}$命令怎样呢？

也由访存指令发出，只是访问的地址范围不同！

系统互联及输入输出组织

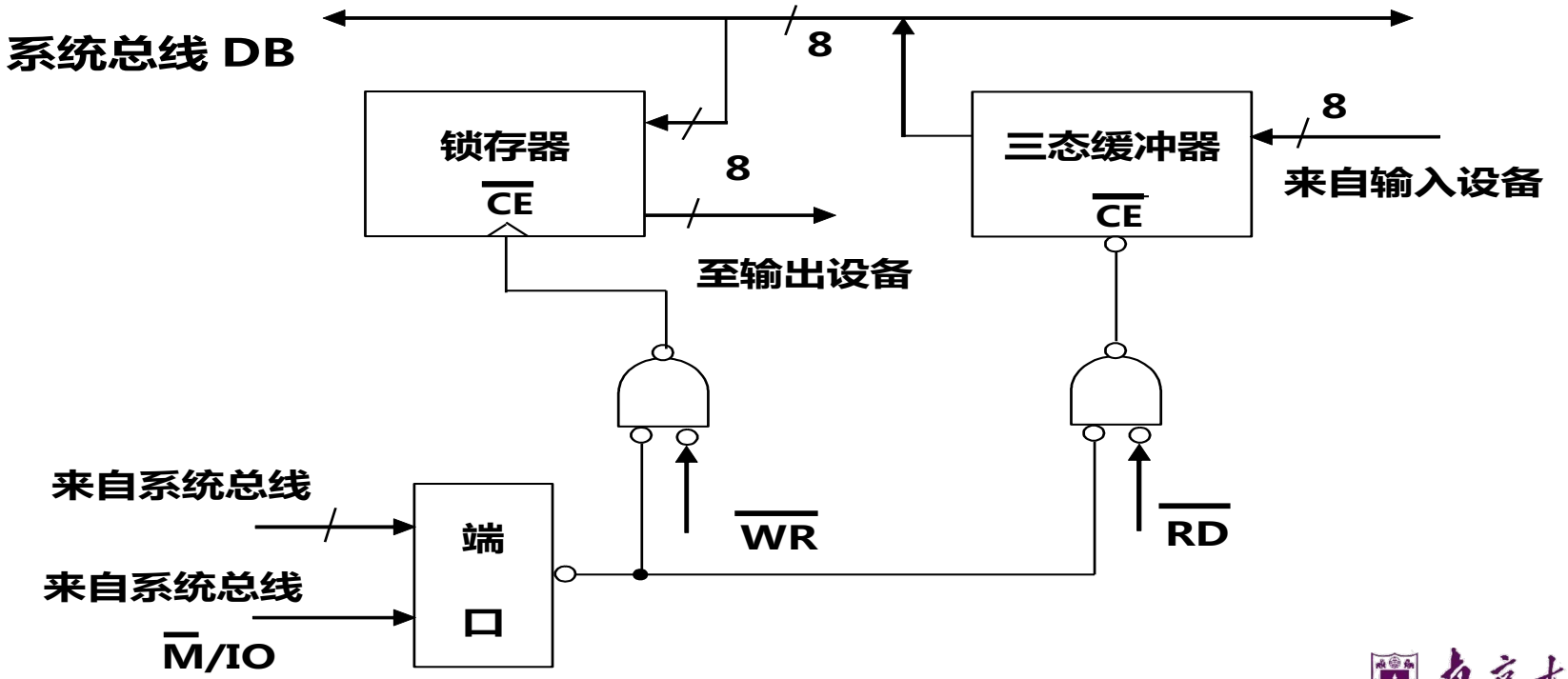
- 外部设备的分类与特点
- 常用输入输出设备
- 外设与CPU和主存的互连
- **I/O数据传送控制方式**
- 内核空间I/O软件

I/O数据传送控制方式

- **程序直接控制方式（最简单的I/O方式）**
  - **无条件传送**：对简单外设定时（同步）进行数据传送
  - **条件传送**：Polling (轮询，查询)：OS主动查询，也称为程序查询方式
    - I/O设备（包括I/O接口）将自己的状态放到一个状态寄存器中
    - OS阶段性地查询状态寄存器中的特定状态，以决定下一步动作
- **I/O Interrupt (中断I/O方式): 几乎所有系统都支持的中断I/O方式**
  - 若一个I/O设备需要CPU干预，它就通过中断请求通知CPU
  - CPU中止当前程序的执行，调出OS（中断处理程序）来执行
  - 处理结束后，再返回到被中止的程序继续执行
  - OS是被动调出的，也称为中断驱动I/O方式
- **直接存储器存取(DMA方式): 磁盘等高速外设特有的I/O方式**
  - 磁盘等高速外设成批地直接和主存进行数据交换
  - 需要专门的DMA控制器控制总线，完成数据传送
  - 当外设准备好数据后，向DMA控制器发DMA请求信号，DMA控制器再向CPU发总线请求，CPU让出总线后，由DMA控制器控制总线进行传输，无需CPU干涉

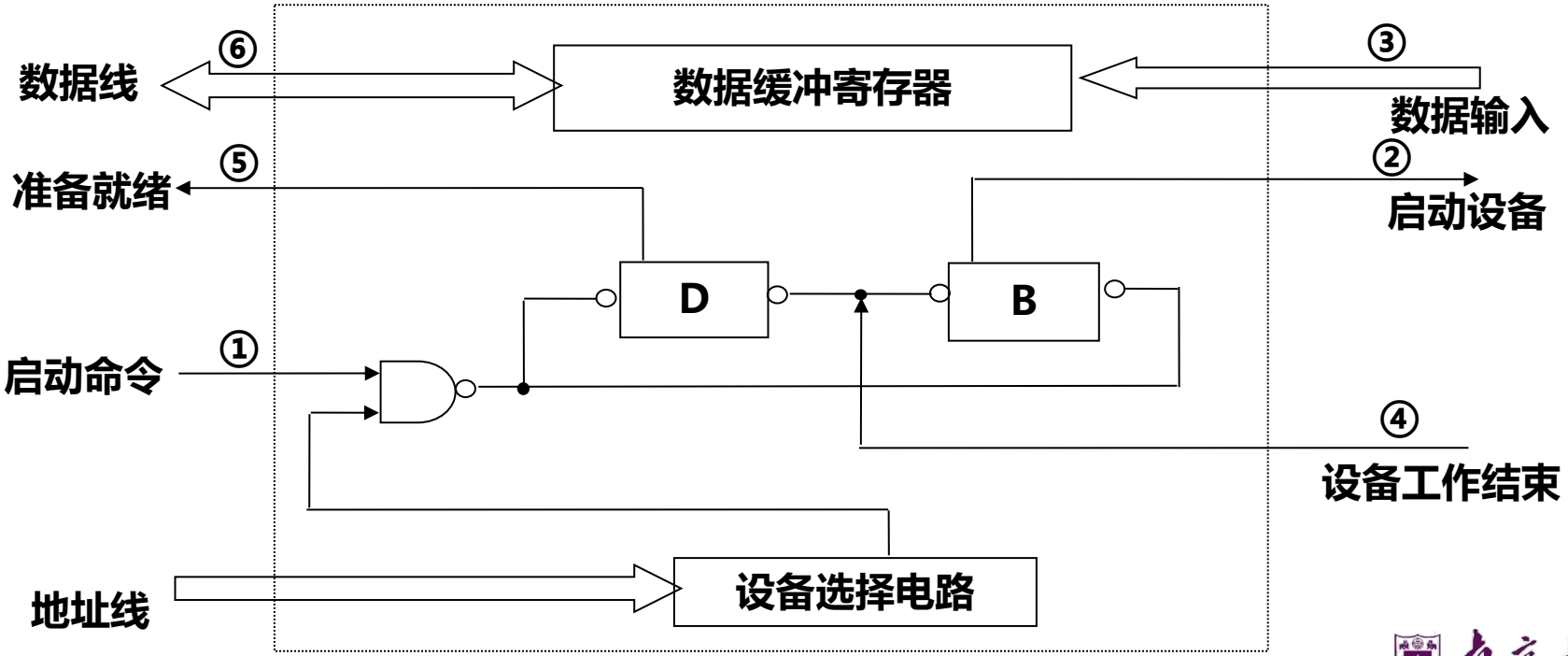
程序直接控制方式

- 无条件传送方式



程序直接控制方式

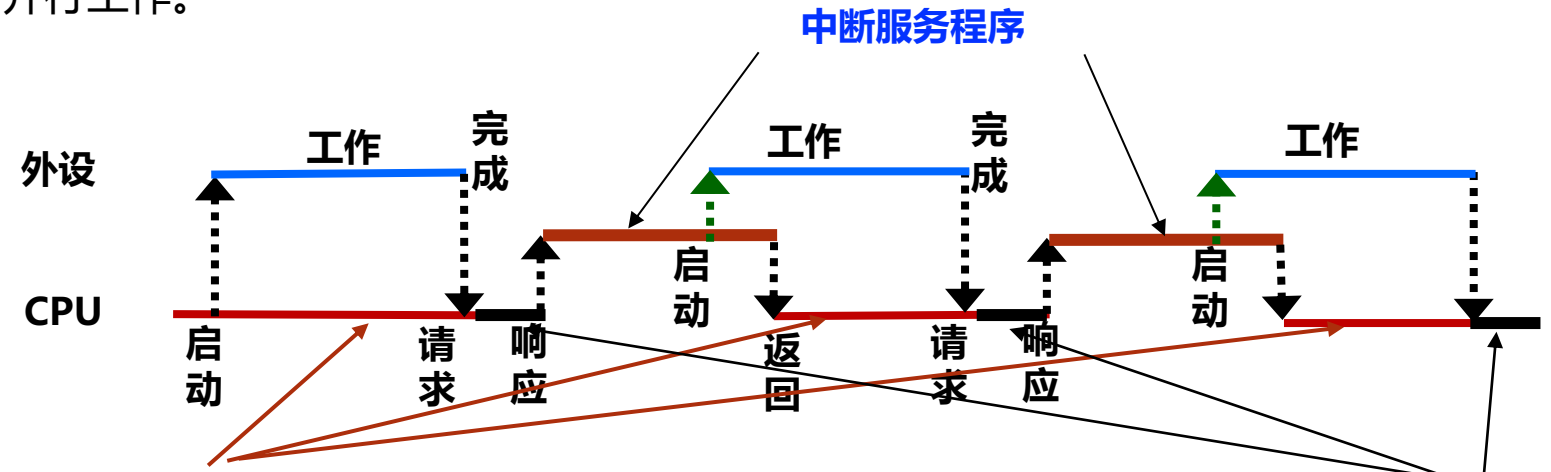
- 条件传送接口方式



中断I/O方式

- **基本思想：**

当外设准备好时，便向CPU发中断请求，CPU响应后，中止现行程序的执行，转入一个“中断服务程序”进行输入/出操作，实现主机和外设接口之间的数据传送，并启动外设工作。 “中断服务程序”执行完后，返回原被中止的程序断点处继续执行。此时，外设和CPU并行工作。



上述哪段时间CPU和外设并行工作？

程序切换（响应中断过程）由硬件完成，即执行“中断隐指令”

中断系统的基本职能

- **及时记录各种中断请求信号**：通常用一个中断请求寄存器来保存。
- **自动响应中断请求**：CPU在每条指令执行完、下条指令取出前，自动检测中断，并根据情况决定是否响应和响应哪个中断。
- **自动判优**：判断中断优先级，选择优先级最高的中断先响应。
- **保护被中断程序的断点和现场**：原程序被中止处的指令地址和原程序的程序状态和各寄存器的内容等必须被保存，以便能正确回到原被中止处继续执行。
- **中断屏蔽**：通过中断屏蔽实现多重中断的嵌套执行，中断屏蔽功能通过一个中断屏蔽字寄存器来实现。

中断控制器的基本结构

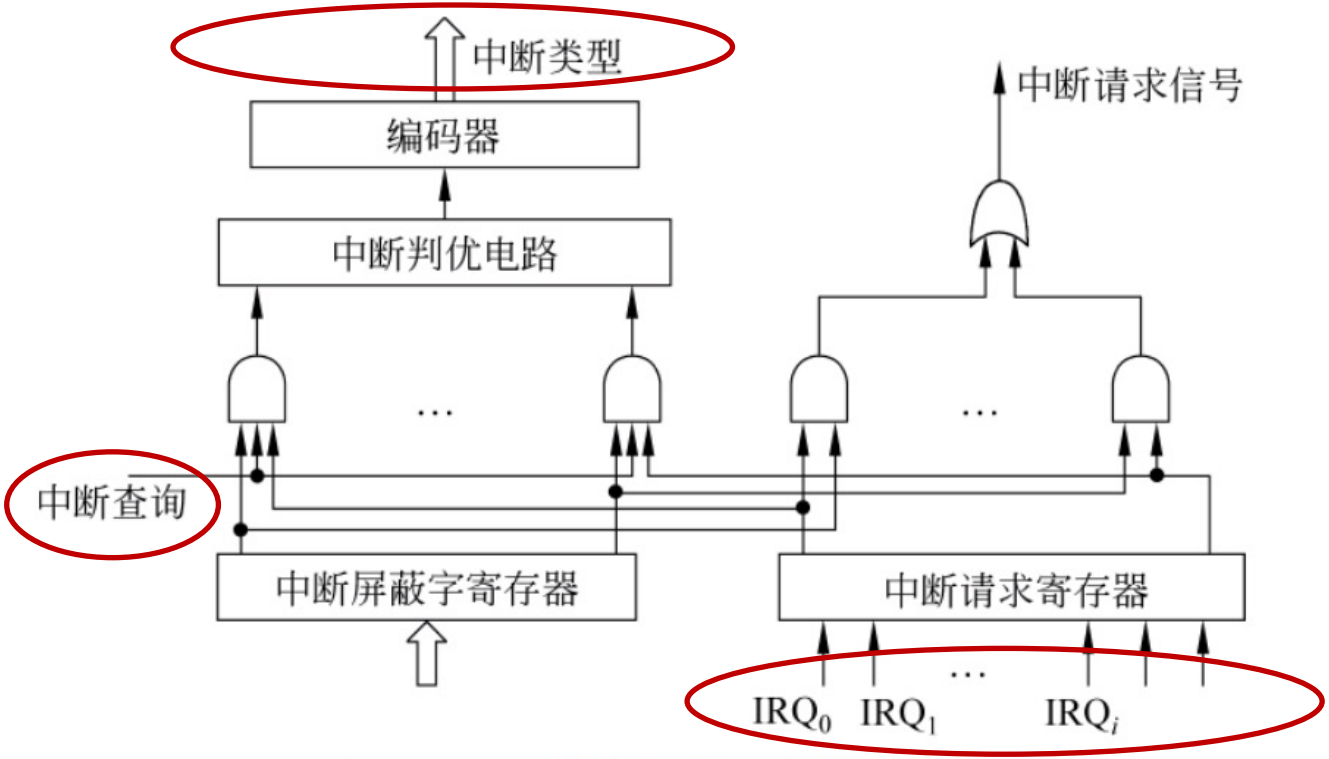


图 8.16　中断控制器的基本结构

中断处理过程

- **中断过程：中断响应 + 中断处理**
- **中断响应**的结果就是调出相应的中断服务程序（处在“禁止中断”状态）
- **中断处理**是指执行相应中断服务程序的过程
  - 不同的中断源其对应的中断服务程序不同。
  - 典型的**多重中断**处理（中断服务程序）分为三个阶段：
    - **先行段（准备阶段）**
      - 保护现场及旧屏蔽字
      - 查明原因（软件识别中断时）
      - 设置新屏蔽字
      - 开中断

      处在“**禁止中断**”状态，不允许被打断
    - **本体段（具体的中断处理阶段）**

      处在“**允许中断**”状态，可被新的处理优先级更高的中断打断
    - **结束段（恢复阶段）**
      - 关中断
      - 恢复现场及旧屏蔽字
      - 清“中断请求”
      - 开中断
      - 中断返回

      处在“**禁止中断**”状态，不允许被打断

单重中断不允许在中断处理时被新的中断打断，因而直到中断返回前才会开中断。单重中断系统无需设置中断屏蔽字。

多重中断的概念

- **多重中断和中断处理优先权的动态分配**
  - **多重中断的概念：**

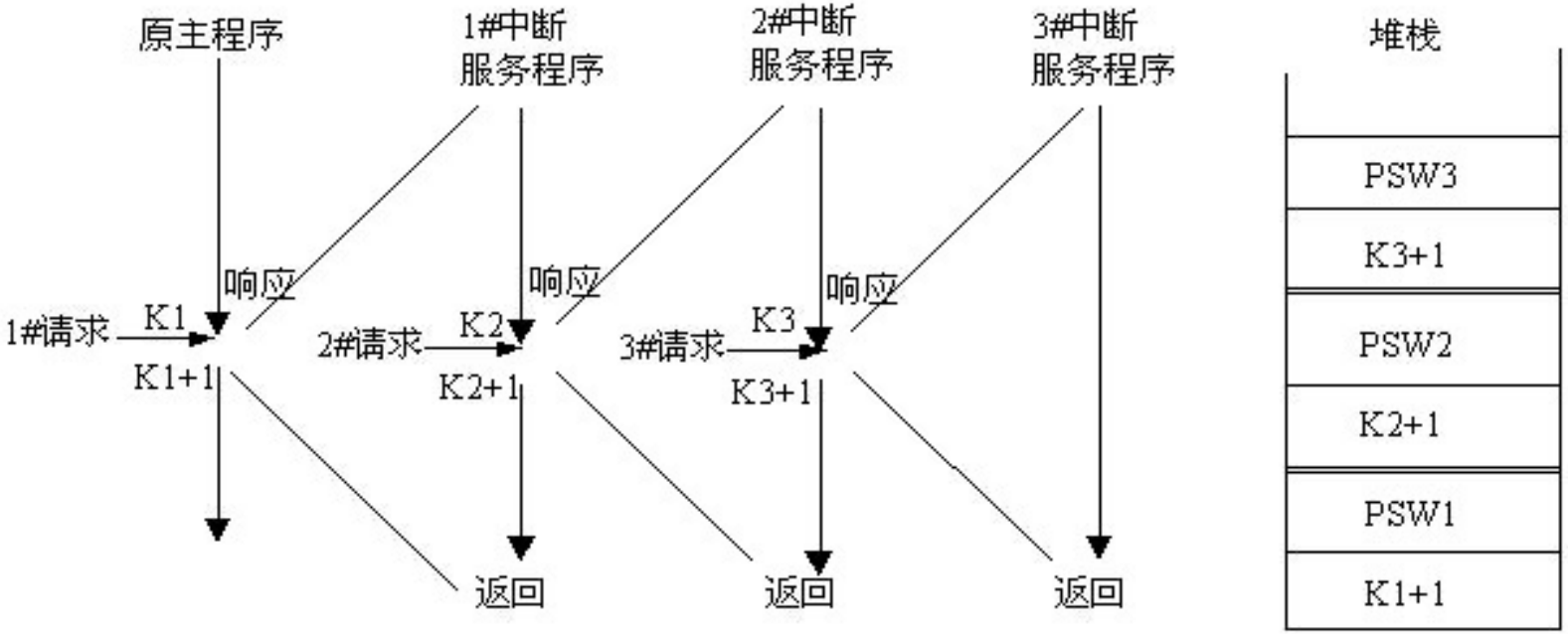
    在一个中断处理（即执行中断服务程序）过程中，若又有新的中断请求发生，而新中断优先级高于正在执行的中断，则应立即中止正在执行的中断服务程序，转去处理新的中断。这种情况为多重中断，也称中断嵌套。
  - **中断优先级的概念：**

    中断响应优先级----由查询程序或硬联排队线路决定的优先权，反映多个中断同时请求时选择哪个响应。

    中断处理优先级----由各自的中断屏蔽字来动态设定，反映本中断与其它中断间的关系。

回想一下，中断屏蔽字在何处用到的？

多重中断嵌套



中断优先级的顺序是：
3# > 2# > 1#

1#对2#开放（不屏蔽）
2#对3#开放（不屏蔽）

中断优先权的动态分配

- 举例：假定某中断系统有四个中断源，其响应优先级为1>2>3>4。假定在用户程序时同时发生1、3、和4级中断请求，执行3级中断服务程序时发生2级中断请求。分别写出处理优先级为1>2>3>4和1>4>3>2时各中断的屏蔽字及CPU完成中断处理的过程。

(1) 中断处理优先级为1>2>3>4时：

表 8.1　处理优先级为 1＃＞2＃＞3＃＞4＃时的屏蔽字

中断源	中断屏蔽字			
	1＃	2＃	3＃	4＃
1＃中断源	1	0	0	0
2＃中断源	1	1	0	0
3＃中断源	1	1	1	0
4＃中断源	1	1	1	1

（假定 0 是屏蔽，1 是开放）

中断优先权的动态分配

- 举例：假定某中断系统有四个中断源，其响应优先级为1>2>3>4。假定在用户程序时同时发生1、3、和4级中断请求，执行3级中断服务程序时发生2级中断请求。分别写出处理优先级为1>2>3>4和1>4>3>2时各中断的屏蔽字及CPU完成中断处理的过程。

(1) 中断处理优先级为1>2>3>4时：

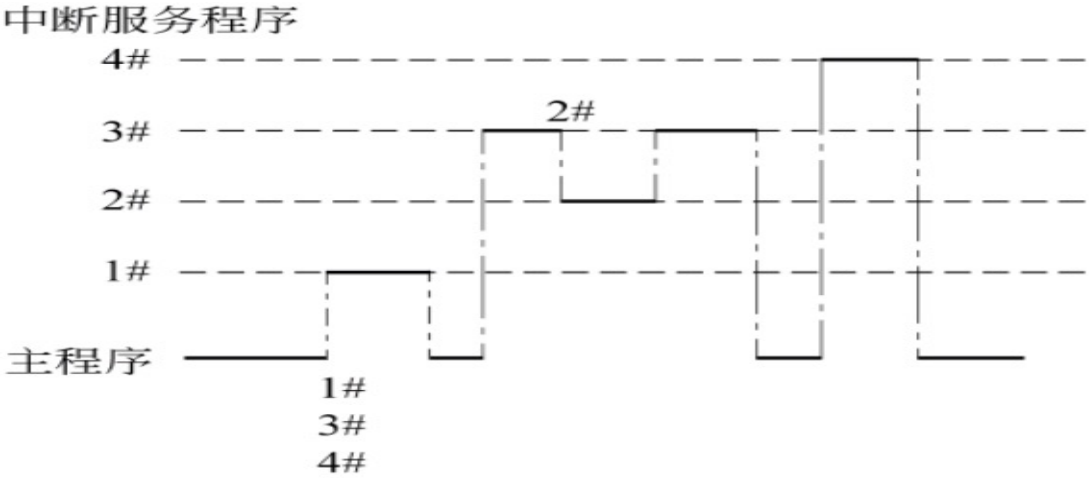


图 8.19　处理优先级为 1＃＞2＃＞3＃＞4＃时 CPU 运动轨迹

中断优先权的动态分配

- 举例：假定某中断系统有四个中断源，其响应优先级为1>2>3>4。假定在用户程序时同时发生1、3、和4级中断请求，执行3级中断服务程序时发生2级中断请求。分别写出处理优先级为1>2>3>4和1>4>3>2时各中断的屏蔽字及CPU完成中断处理的过程。

(2) 中断处理优先级为1>4>3>2时：

表 8.2　处理优先级为 1＃＞4＃＞3＃＞2＃时的屏蔽字

中断源	中断屏蔽字			
	1＃	2＃	3＃	4＃
1＃中断源	1	0	0	0
2＃中断源	1	1	1	1
3＃中断源	1	0	1	1
4＃中断源	1	0	0	1

（假定 0 是屏蔽，1 是开放）

中断优先权的动态分配

- 举例：假定某中断系统有四个中断源，其响应优先级为1>2>3>4。假定在用户程序时同时发生1、3、和4级中断请求，执行3级中断服务程序时发生2级中断请求。分别写出处理优先级为1>2>3>4和1>4>3>2时各中断的屏蔽字及CPU完成中断处理的过程。

(2) 中断处理优先级为1>4>3>2时：

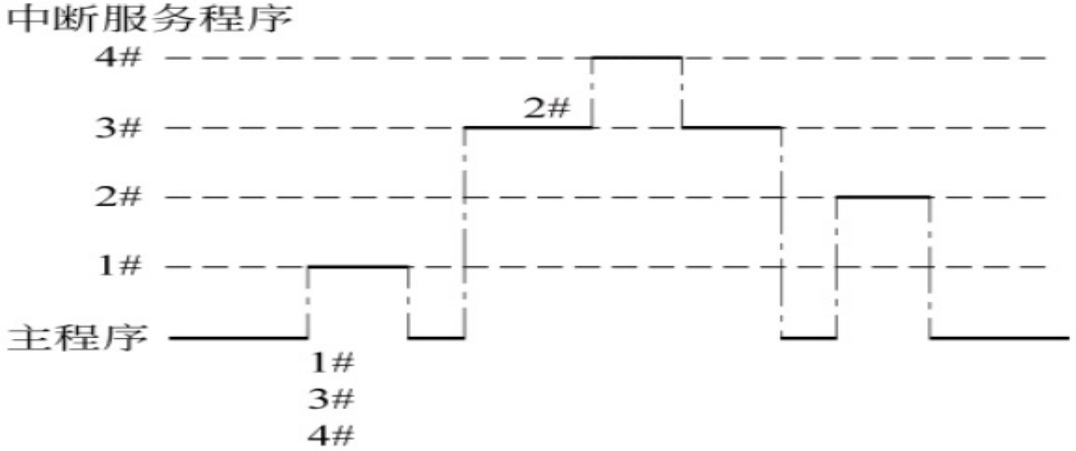
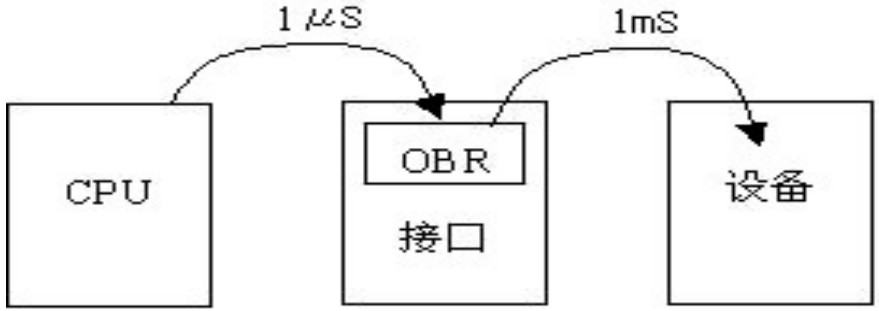


图 8.20　处理优先级为 1＃＞4＃＞3＃＞2＃时 CPU 运动轨迹

程序控制方式和中断方式的比较

- **举例：**假定某机控制一台设备输出一批数据。数据由主机输出到接口的数据缓冲器OBR，需要1μs。再由OBR输出到设备，需要1ms。设一条指令的执行时间为1μs(包括隐指令)。试计算采用程序传送方式和中断传送方式的数据传输速度和对主机的占用率。

- **问题：**CPU如何把数据送到OBR，I/O接口如何把OBR中的数据送到设备？
- CPU执行I/O指令来将数据送OBR；而I/O接口则是自动把数据送到设备。



- **对主机占用率**：在进行I/O操作过程中，处理器有多少时间花费在输入/出操作上。
- **数据传送速度（吞吐量、I/O带宽）**：单位时间内传送的数据量。

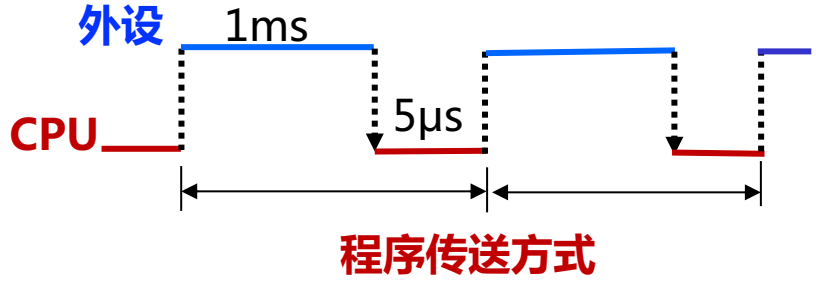
假定每个数据的传送都要重新启动！即是字符型设备

程序控制方式和中断方式的比较

（1）程序直接控制传送方式

若查询程序有10条，第5条为启动设备的指令，则：

- 数据传输率为：1/(1000+5) μs，约为每秒995个数据。
- 主机占用率=100%

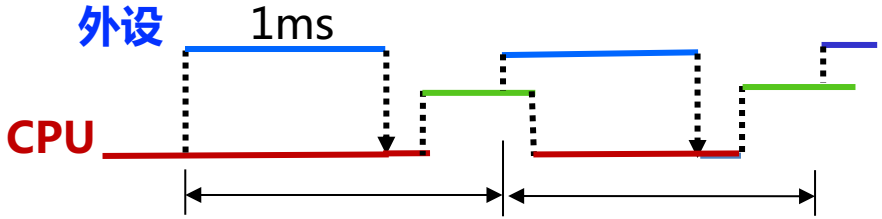


程序传送方式

（2）中断传送方式

若中断服务程序有30条，在第20条启动设备，则：

- 数据传输率为：1/(1000+1+20)μs，约为每秒979个数据。
- 主机占用率为：(1+30)/(1000+1+20)=3%



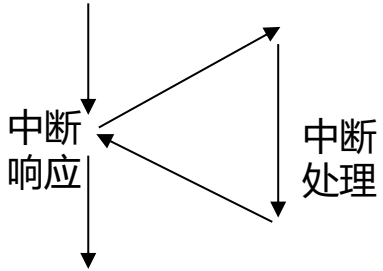
中断传送方式

- **为什么中断服务程序比查询程序长？**

因为中断服务程序有额外开销，如：保存现场、保存旧屏蔽字、开中断、（查询中断源）等

中断响应过程

- **中断过程**
  - 中断检测（硬件实现）
  - 中断响应（硬件实现）
  - 中断处理（软件实现）



- **中断响应**
  - 中断响应是指主机发现外部中断请求，中止现行程序的执行，到调出中断服务程序这一过程。

(1) 中断响应的条件

① CPU处于开中断状态

② 在一条指令执行完

③ 至少要有一个未被屏蔽的中断请求

问题：中断响应的时点与异常处理的时点是否相同？为什么？

通常在一条指令执行结束后开始查询有无中断请求，有的话立即响应，所以，一般在指令执行完时响应中断；而“异常”发生在指令执行过程中，所以，不能等到指令执行完才进行异常处理。

中断响应过程

(2) 中断响应过程

执行一条隐指令，需完成一次总线操作，从总线上取中断类型号

- **具体来说，处理器做三件事：**

① 关中断

0 => 中断允许触发器C_{IEN}

② 保护断点和程序状态

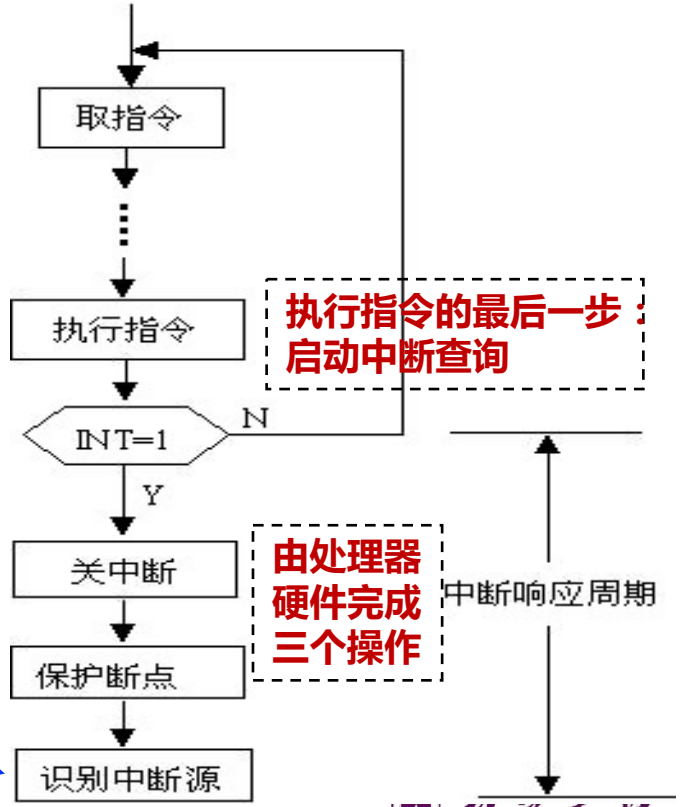
PC => 堆栈（或特殊寄存器EPC）

PSW => 堆栈

③ 识别中断源：

取得中断服务程序首地址和初始PSW分别送PC和PSWR

从总线的数据线上取得中断类型号后得到中断服务程序首址（**向量中断**）或直接转中断查询程序（**软件查询**）

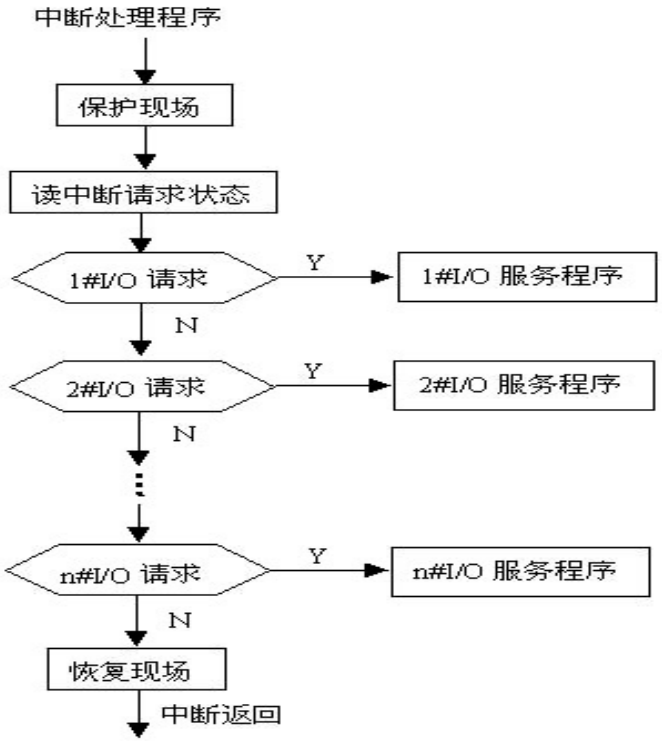


中断识别和判优方法

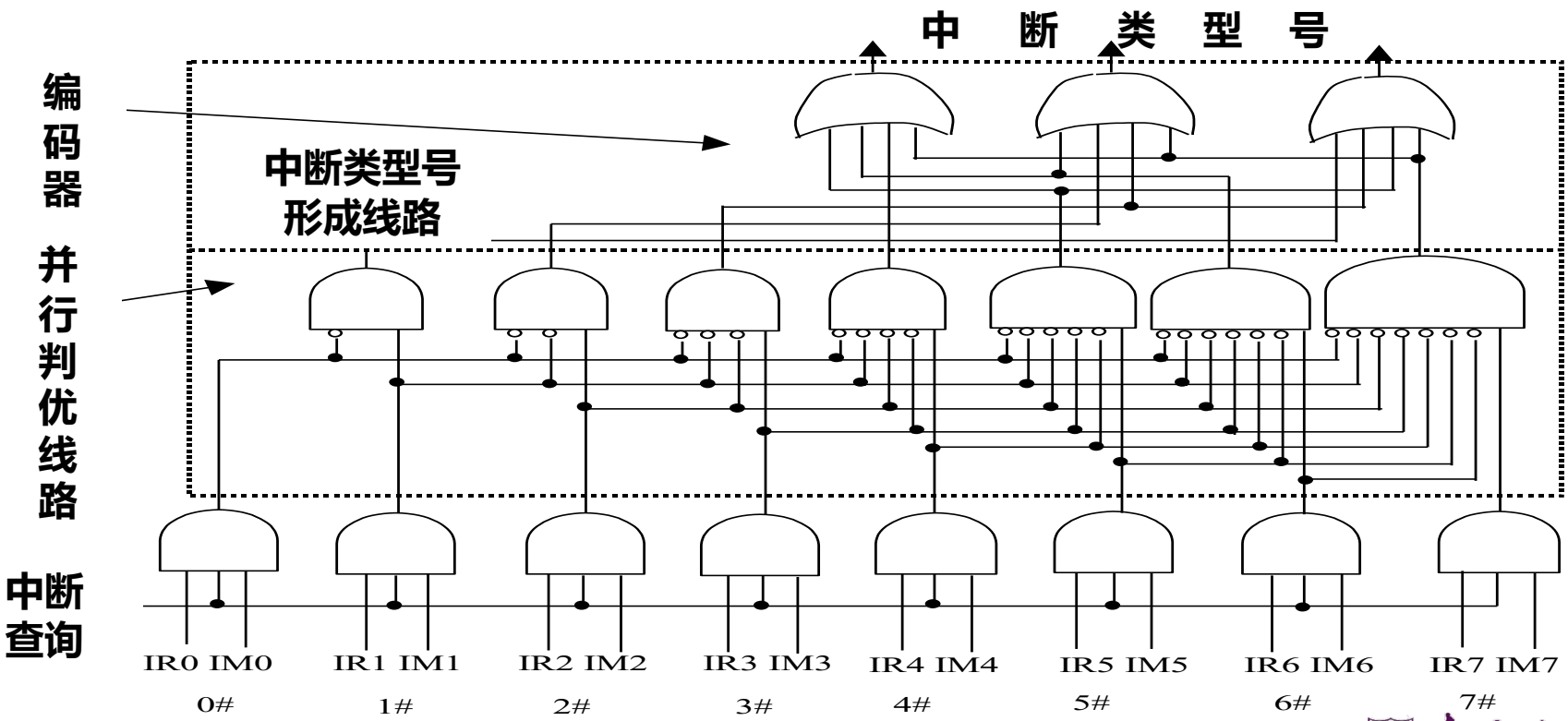
- **软件方法（轮询）**
  - 中断查询程序根据中断请求状态，按优先级顺序来识别（如：MIPS的异常/中断查询程序入口为：0x80000180）
- **硬件方法（向量中断）**
  - 将所有中断请求状态送到一个排队电路中，根据中断优先级识别出最高优先级的中断请求
    - **链式查询（菊花链）**
    - **独立请求（并行判优）**

中断控制器中的“**中断优先权编码器**” 专门用来进行中断源识别

注：内部异常和外部中断都有优先级，通常所有内部异常的优先级都比外部中断高。为什么？



中断优先权编码器



DMA方式

- **DMA的全称**
  - 直接存储器存取（Direct Memory Access）
- **为什么要引入DMA方式？**
  - 程序直接控制方式受“踏步”现象的限制，效率低下，不适合高速设备和主机间的数据传送。
  - 中断控制方式虽比程序直接控制方式有效，CPU和外设有一定的并行度，但由于下列原因也不适合高速设备和主机间的数据传送。
    - **对I/O请求响应慢**：每传送一个数据都要等待外设的中断请求，并增加许多中断响应和中断处理前、后的附加开销（保护断点、现场等），不能及时响应I/O请求。
    - **数据传送速度慢**：数据传送由软件完成（由CPU执行相应的中断服务程序来完成），速度慢 。

DMA方式的基本要点

- **DMA方式的基本思想**
  - 在高速外设和主存间直接传送数据
  - 由专门硬件（即：DMA接口）控制总线进行传输
- **DMA方式适用场合**
  - 高速设备（如：磁盘、光盘等）
  - 成批数据交换，且数据间间隔时间短，一旦启动，数据连续读写
- **采用“请求-响应”方式**
  - 每当高速设备准备好数据，就进行一次“DMA请求”，DMA控制器接受到DMA请求后，申请总线使用权
  - DMA控制器的总线使用优先级比CPU高
- **与中断控制方式结合使用**
  - DMA传送前，寻道、旋转等操作结束时，通过“中断”告知CPU
  - 在DMA控制器控制总线进行数据传送时，CPU执行其他程序
  - DMA传送结束时，要通过“DMA结束中断”告知CPU

DMA数据传送方式

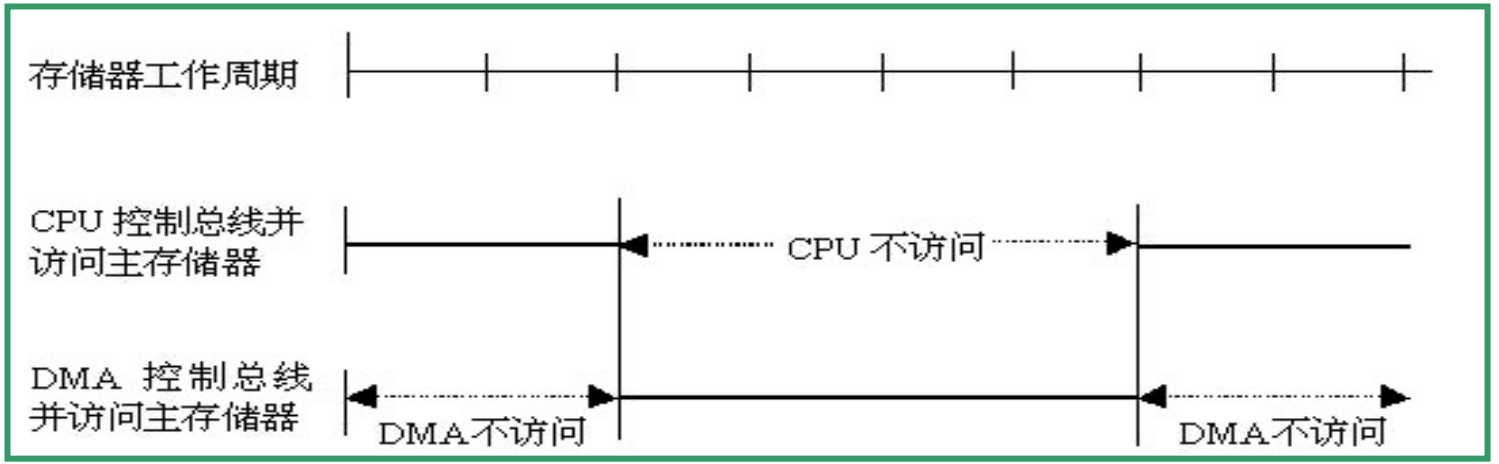
- 由于DMA接口和CPU共享主存，所以可能出现两者争用主存的现象，为使两者协调使用主存，DMA通常采用以下三种方式进行数据传送。
- **(1) CPU停止法(成组传送)**

  DMA传输时，CPU脱离总线，停止访问主存，直到DMA传送一块数据结束。
- **(2) 周期挪用(窃取)法(单字传送)**

  DMA传输时，CPU让出一个总线事务周期，由DMA控制总线来访问主存，传送完一个数据后立即释放总线。
- **(3) 交替分时访问法**

  每个存储周期分成两个时间片，一个给CPU，一个给DMA，这样在每个存储周期内，CPU和DMA都可访问存储器。

CPU停止法



- **优点：**控制简单、适用于传输率很高的外设实现成组数据传送。
- **缺点：**
  - **CPU工作受影响**：DMA访存时CPU基本上处于停止状态。
  - **主存周期没有被充分利用**：即使I/O设备高速运行，但两个数据之间的准备间隔时间也总大于一个存储周期，所以主存周期没有被充分利用。

CPU停止法

- **弥补CPU停止法缺点的做法：**
  - **在DMA接口中引入缓冲器**

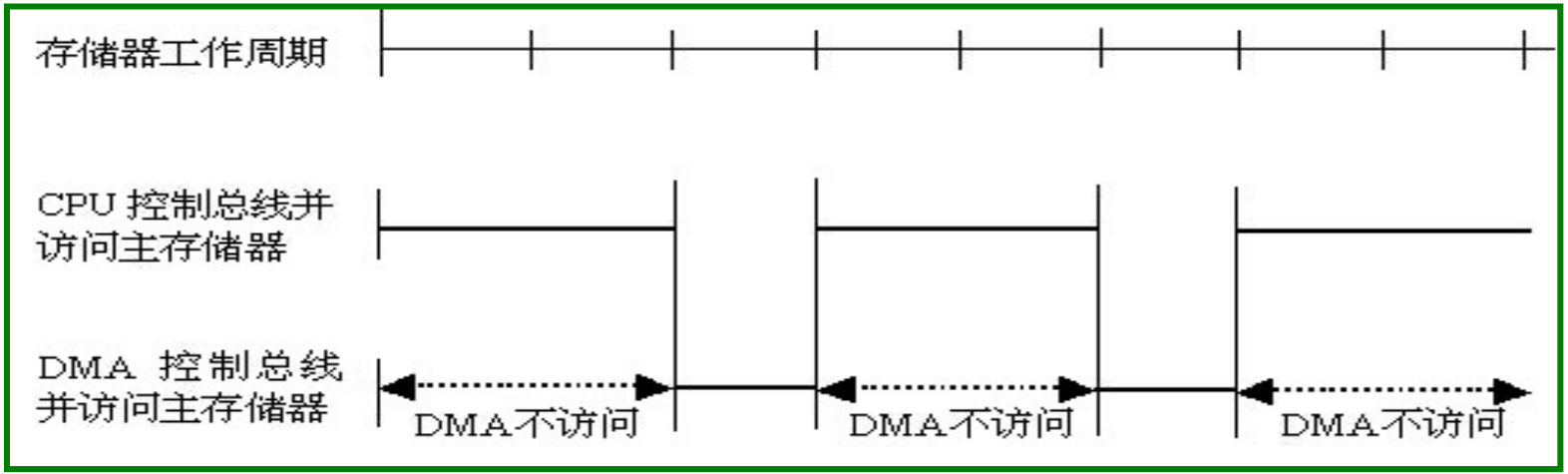
    在DMA接口中采用一个小容量的半导体存储器，使I/O设备先和这个小容量存储器交换数据，然后再使用总线由小容量存储器与主存进行数据交换。这样可减少DMA传送数据时占用总线的时间，也就减少了CPU的等待时间。

  - **采用周期挪用（窃取）法**

    挪用一个存储周期进行外设和主存的一个数据交换。

    每次DMA传送完一个数据就释放总线，使在外设准备下一数据时，CPU能插空访问主存。

周期挪用法



- **优点：**既能及时响应I/O请求，又能较好地发挥CPU和主存的效率。
  ➢这种方式下，在下一数据的准备阶段，主存周期被CPU充分利用。因此适合于I/O设备的读写周期大于主存周期的情况。
- **缺点：**每次DMA访存都要申请总线控制权、占用总线进行传送、释放总线，因此，会增加传输开销。

周期挪用法

- **I/O设备要求DMA传送时可能会遇到以下三种情况之一：**
  - **CPU不需访问主存**

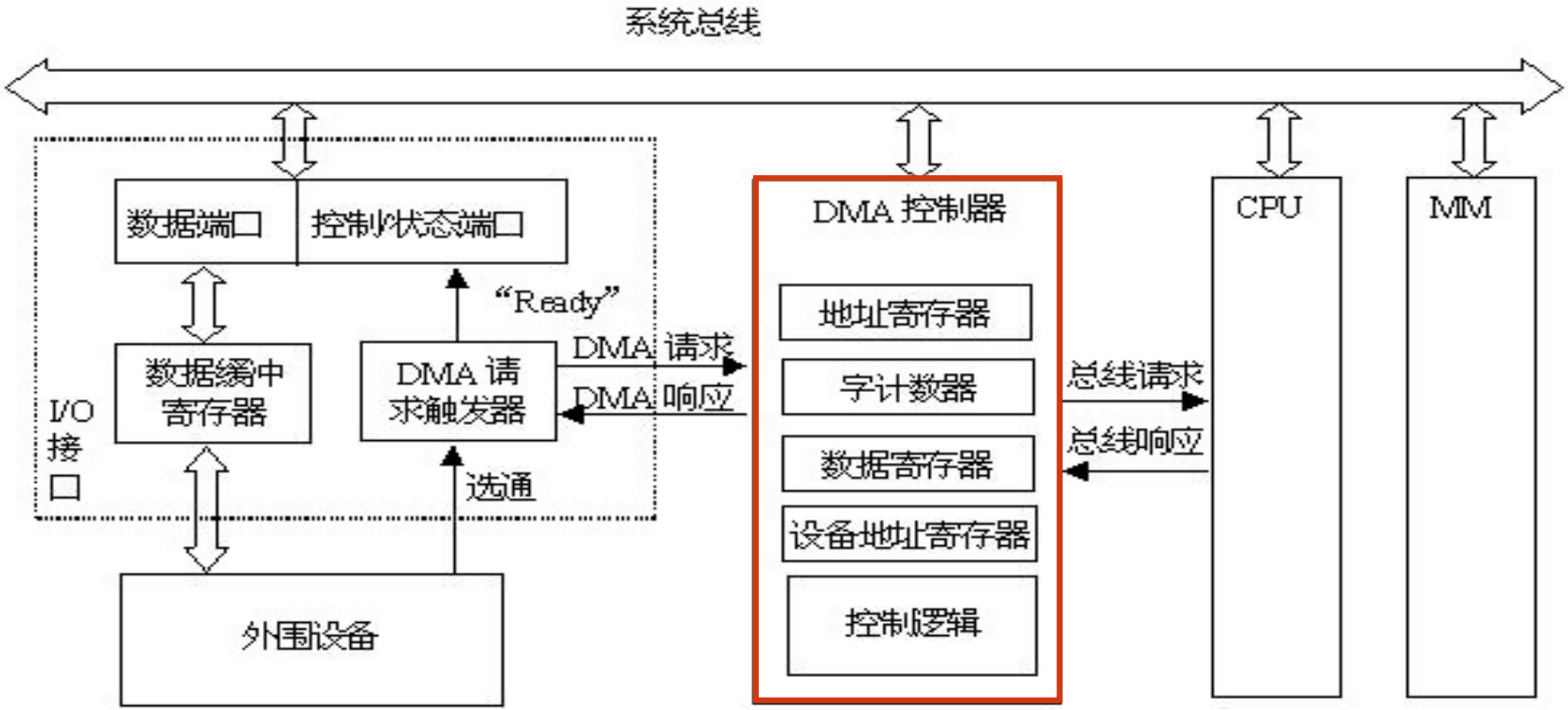
    此时，不会发生冲突，两者并行。

    如: CPU正在执行乘法指令，要花很长时间而不需马上访存。
  - **CPU正在访问主存**

    此时须等到存储周期结束，CPU让出总线，DMA才能访存。
  - **CPU也同时要访问主存**

    此时出现访存冲突。因为不马上响应DMA请求的话，高速设备可能会发生数据丢失，所以，DMA的总线优先权比CPU高。这时，先让DMA占用总线，窃取一个主存周期，完成一个数据的交换。这样，CPU便要延迟一段时间才能访存。

DMA方式下的系统逻辑结构



DMA控制器的功能

- DMA数据传送过程由DMA接口的控制逻辑完成，所以DMA 接口也称DMA控制器。其功能为：
  - **(1) 请求**：能接收外设发来的“DMA请求”信号，并能向CPU发“总线请求”信号。
  - **(2) 响应**：当CPU发出“总线响应”信号响应请求后，能接管对总线的控制。
  - **(3) 修改主存地址**：能在地址线上给出主存地址，并自动修改主存地址。
  - **(4) 识别传送方向**：能识别传送方向以在控制线上给出正确的读写控制信息。
  - **(5) 确定传送数据个数**。
  - **(6) 能发出DMA结束信号**：引起一次DMA中断，进行数据校验等一些后处理。

DMA控制器的操作步骤

- **DMA操作步骤**

(1) DMA控制器的预置(初始化)----软件实现

- 准备内存
- 设置参数
- 启动外设

(2) DMA数据传送----硬件实现

- DMA请求：选通-〉DMA请求-〉总线请求
- DMA响应：总线响应(CPU让出总线)-〉DMA响应
- DMA传送：DMA控制总线进行数据传送

(3) DMA结束处理----软件实现

根据计数值为“0”，发出DMA结束信号去接口控制产生DMA中断请求信号，转入中断服务程序，做一些数据校验等后处理工作。

DMA控制器的初始化

- **DMA控制器的预置(初始化)**
  - **准备内存区**
    * 输入：在内存设置好缓冲区
    * 输出：先在内存准备好数据
  - **设置传送参数**

    执行I/O指令，测试外设状态，对DMA控制器设置各种参数：
    * 内存首址=〉地址寄存器
    * 字计数值=〉字计数器
    * 传送方向=〉控制寄存器
    * 设备地址=〉设备地址寄存器
  - **启动外设，然后执行其他程序**

DMA传输过程

(1) 当外设准备好数据，或准备好接收数据时，发“选通”信号，使数据送数据缓冲寄存器，同时DMA请求触发器置“1”。

(2) DMA请求触发器向控制/状态端口发“Ready”信号，同时向DMA控制器发“DMA请求”信号。

(3) DMA控制器向CPU发“总线请求”信号。

(4) CPU完成现行机器周期后，响应DMA请求，发出“总线响应”信号。DMA控制器接受到该信号后，发出“DMA响应”信号，使DMA请求触发器复位。此时，CPU浮动它的总线，让出总线控制权，由DMA控制器控制总线。

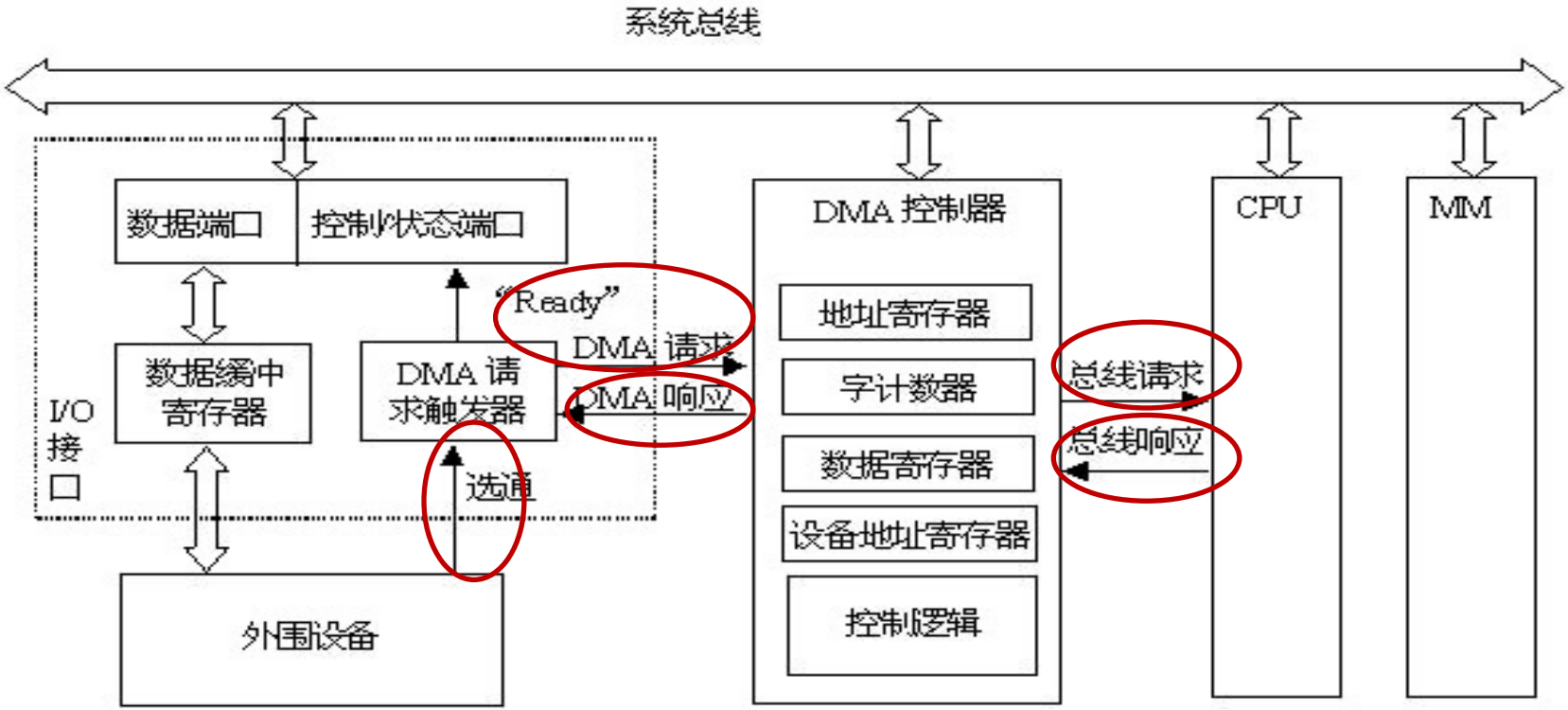
(5) DMA控制器给出内存地址，并在其读/写线上发出“读”或“写”命令，随后在数据总线上给出数据。

(6) 根据读写命令，将数据总线上的数据写入存储器中，或写入数据端口，并进行主存地址增量，计数值减1，

若采用“CPU停止法”，则循环第6步，直到计数值为“0”。

若采用“周期挪用法”，则释放总线（下次数据传送时再按过程(1)到(6)进行）。

DMA方式下的系统逻辑结构



DMA方式和中断方式的区别

(1) DMA方式下数据传送由硬件(DMA控制器)完成；中断方式下，数据传送由软件（CPU执行中断服务程序）完成。

(2) DMA请求的是对存储器访问，也即对总线控制权的请求，没有中止现行程序的必要；而中断请求要处理器转去执行中断服务程序，因此要中止现行程序，保存断点、现场等。

(3) 中断除了能完成外设和主机的数据交换，还能处理异常事件；而DMA方式下不能处理异常事件。

(4) 中断响应在一个指令周期结束后；而DMA响应是在一个总线周期后。

(5) DMA方式用于高速设备；而中断方式用于低、慢速设备。

(6) DMA方式下，外设与CPU并行度高；而中断方式下，外设与CPU并行度低。（体现在数据传送时的并行性）

系统互联及输入输出组织

- 外部设备的分类与特点
- 常用输入输出设备
- 外设与CPU和主存的互连
- I/O数据传送控制方式
- **内核空间I/O软件**

I/O子系统概述

- 所有高级语言的运行时（runtime）都提供了执行I/O功能的机制

  例如，C语言中提供了包含像printf()和scanf()等这样的标准I/O库函数，C++语言中提供了如 << （输入）和 >>（输出）这样的重载操作符。

- 从高级语言程序中通过I/O函数或I/O操作符提出I/O请求，到设备响应并完成I/O请求，涉及到多层次I/O软件和I/O硬件的协作。
- I/O子系统也采用层次结构

从用户I/O软件切换到内核I/O软件的唯一办法是“异常”机制：**系统调用（自陷）**

I/O子系统概述

- 各类用户的I/O请求需要通过某种方式传给OS：
  - 最终用户：键盘、鼠标通过操作界面传递给OS
  - 用户程序：通过函数（高级语言）转换为系统调用传递给OS
- I/O软件被组织成从高到低的四个层次，层次越低，则越接近设备而越远离用户程序。这四个层次依次为：

  (1) 用户层I/O软件（I/O函数调用系统调用）

  (2) 与设备无关的操作系统I/O软件

  (3) 设备驱动程序

  (4) I/O中断处理程序

} OS

OS在I/O系统中极其重要！

大部分I/O软件都属于操作系统内核态程序，最初的I/O请求在用户程序中提出。

操作系统在I/O中扮演的角色

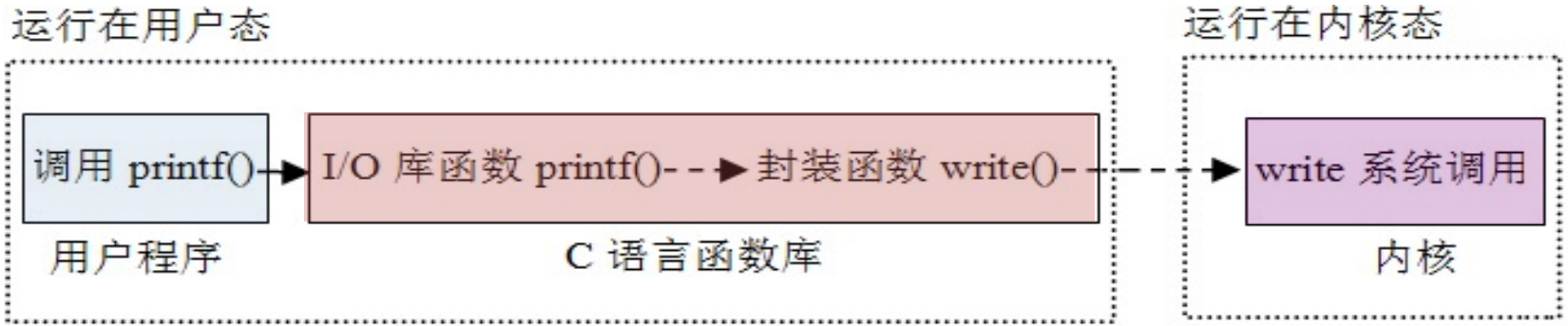
- **OS的职责由I/O系统的三个特性决定**
  - 共享性：I/O系统被处理器执行的多个程序共享，由OS统一调度管理
  - 复杂性：I/O设备控制的细节比较复杂，不能由上层用户程序来实现，需OS提供专门的驱动程序
  - 采用中断I/O方式：I/O系统通常使用外部中断请求来要求处理器执行专门的输入/出程序。中断导致向内核态转移，故必须由OS来处理
- **OS在I/O中的职能**
  - 保证用户程序只能访问自己有权访问的那部分I/O设备
  - 为用户程序提供设备驱动程序以屏蔽设备控制的细节
  - 处理外部I/O中断，提供中断服务程序
  - 对共享的I/O资源提供合理的调度管理，使系统的吞吐率达到最佳
- **主机必须和I/O设备进行以下三类信息的通信**
  - OS必须能向I/O设备提供命令，如：磁盘寻道
  - 需要知道：何时I/O设备能完成操作？何时遇到什么异常问题？
  - 数据必须能在主机（主存或CPU）和I/O设备之间进行传输

用户程序、C函数和内核

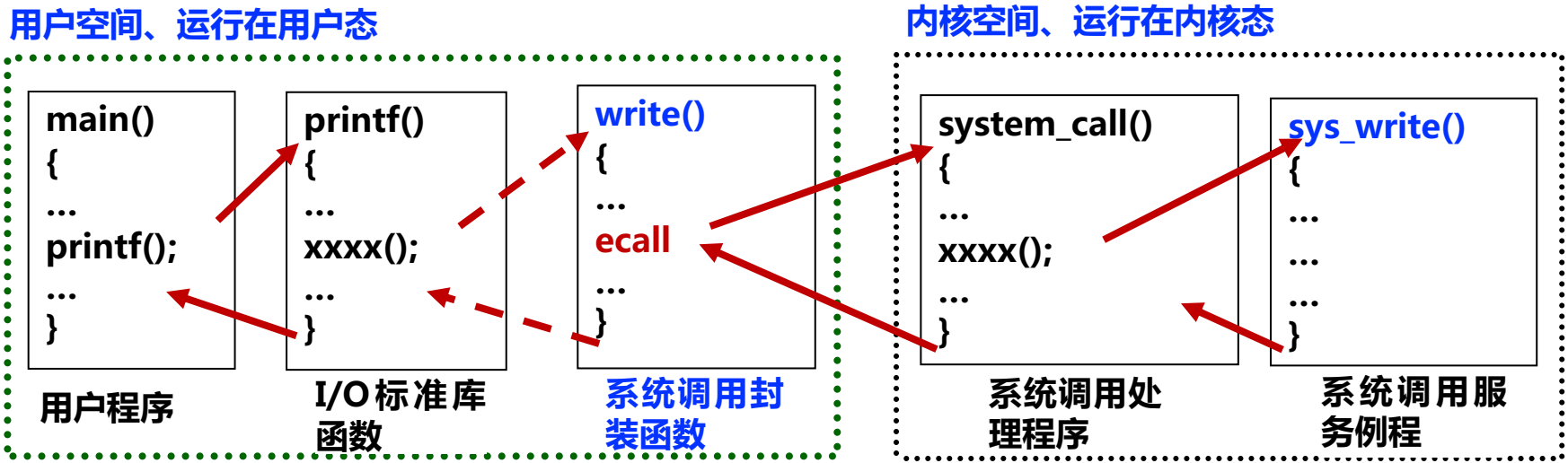
- **用户程序总是通过某种I/O函数或I/O操作符请求I/O操作。**

  例如，读一个磁盘文件记录时，可调用C标准I/O库函数**fread()**，也可直接调用系统调用封装函数**read()**来提出I/O请求。不管是C库函数、API函数还是系统调用封装函数，最终都通过操作系统内核提供的系统调用来实现I/O。

  printf()函数的调用过程如下：



Linux系统中printf()函数的执行过程



- 某函数调用了printf()，执行到调用printf()语句时，便会转到C语言I/O标准库函数printf()去执行；
- printf()通过一系列函数调用，最终会调用函数write()；
- 调用write()时，便会通过一系列步骤在内核空间中找到write对应的系统调用服务例程sys_write来执行。

在system_call中如何知道要转到sys_write执行呢？ **根据系统调用号！**

与设备无关的I/O软件

- **设备驱动程序统一接口**
  - 操作系统为所有外设的设备驱动程序规定一个统一接口，这样，新设备的驱动程序只要按统一接口规范来编制，就可在不修改操作系统的情况下，添加新设备驱动程序并使用新的外设进行I/O。
  - 所有设备都抽象成文件，设备名和文件名在形式上没有差别，设备和文件具有统一的接口，不同设备名和文件名被映射到对应设备驱动程序。
- **缓冲处理**
  - 每个设备的I/O都需使用内核缓冲区，因而缓冲区的申请和管理等处理是所有设备公共的，可包含在与设备无关的I/O软件部分
- **错误报告**
  - I/O操作在内核态执行时所发生的错误信息，都通过与设备无关的I/O软件返回给用户进程，也即：错误处理框架与设备无关。
  - 直接返回编程等错误，无需设备驱动程序处理，如，请求了不可能的I/O操作；写信息到一个输入设备或从一个输出设备读信息；指定了一个无效缓冲区地址或者参数；指定了不存在的设备等。
  - 有些错误由设备驱动程序检测出来并处理，若驱动程序无法处理，则将错误信息返回给设备无关I/O软件，再由设备无关I/O软件返回给用户进程，如写一个已被破坏的磁盘扇区；打印机缺纸；读一个已关闭的设备等。

与设备无关的I/O软件

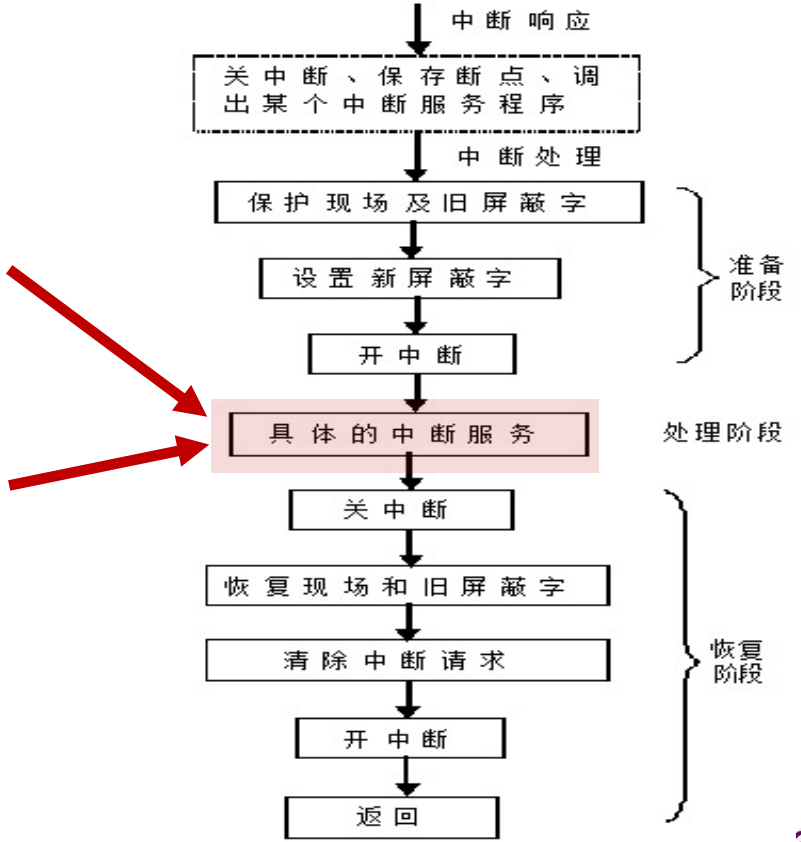
- **打开与关闭文件**
  - 对设备或文件进行打开或关闭等I/O函数所对应的系统调用，并不涉及具体的I/O操作，只要直接对主存中的一些数据结构进行修改即可，这部分工作也由设备无关软件来处理。
- **逻辑块大小处理**
  - 为了为所有的块设备和所有的字符设备分别提供一个统一的抽象视图，以隐藏不同块设备或不同字符设备之间的差异，与设备无关的I/O软件为所有块设备或所有字符设备设置统一的逻辑块大小。
  - 对于块设备，不管磁盘扇区和光盘扇区有多大，所有逻辑数据块的大小相同，这样，高层I/O软件就只需处理简化的抽象设备，从而在高层软件中简化了数据定位等处理 。

设备驱动程序

- **每个外设具体的I/O操作需通过执行设备驱动程序来完成**
- 外设种类繁多、其控制接口不一，导致不同外设的设备驱动程序千差万别，因而设备驱动程序与设备相关
- 每个外设或每类外设都有一个设备控制器，其中包含各种I/O端口。 CPU通过执行设备驱动程序中的I/O指令访问个各种I/O端口
- **设备所采用的I/O控制方式不同，驱动程序的实现方式也不同**
  - **程序直接控制**：驱动程序完成用户程序的I/O请求后才结束。这种情况下，用户进程在I/O过程中不会被阻塞，内核空间的I/O软件一直代表用户进程在内核态进行I/O处理 。（干等！）
  - **中断控制**：驱动程序启动第一次I/O操作后，将调出其他进程执行，而当前用户进程被阻塞。在CPU执行其他进程的同时，外设进行I/O操作，此时，CPU和外设并行工作。外设完成I/O时，向CPU发中断请求，然后CPU调出相应中断服务程序执行。在中断服务程序中再次启动I/O操作。
  - **DMA控制**：驱动程序对DMA控制器初始化后，便发送“启动DMA传送”命令，外设开始进行I/O操作并在外设和主存间传送数据。同时CPU执行处理器调度程序，转其他进程执行，当前用户进程被阻塞。DMA控制器完成所有I/O任务后，向CPU发送一个“DMA完成”中断请求信号。

中断服务程序

- 中断控制和DMA控制两种方式下都需进行中断处理
- **中断控制方式**：中断服务程序主要进行从数缓器取数或写数据到数缓器，然后启动外设工作
- **DMA控制方式**：中断服务程序进行数据校验等后处理工作



在内核I/O软件中用到的**I/O指令、“开中断”和“关中断”等指令都是特权指令**，只能在操作系统内核程序中使用

课程习题（作业）——截止日期：12月20日晚23:59

- **课本310-312页**：第7、8、9、10题
- 提交方式：https://selearning.nju.edu.cn/（教学支持系统）

- 命名：学号+姓名+第*章。
- 若提交遇到问题请及时发邮件或在下一次上课时反馈。

课程习题（作业）——截止日期：12月20日晚23:59

7. 某计算机 CPU 主频为 500MHz，所连接的某外设最大数据传输率为 20KB/s，该外设接口中有一个 16 位的数据缓存器，相应的中断服务程序执行时间为 500 个时钟周期，是否可以用中断方式进行该外设的输入输出？假定该外设的最大数据传输率改为 2MB/s，是否可以用中断方式进行该外设的输入输出？

8. 若某计算机有 5 个中断源 1＃、2＃、3＃、4＃、5＃，中断响应优先级为 1＃＞2＃＞3＃＞4＃＞5＃，而中断处理优先级为 1＃＞4＃＞5＃＞2＃＞3＃。要求：

（1）设计各中断源对应的中断屏蔽字（假设 0 为屏蔽，1 为开放）。

（2）若在运行主程序时，同时出现 2＃、4＃中断请求，在处理 2＃中断过程中，又同时出现 1＃、3＃、5＃中断请求，试画出 CPU 的运行过程示意图。

课程习题（作业）——截止日期：12月20日晚23:59

9. 假定某计算机字长 16 位，没有 cache，运算器一次定点加法时间等于 100ns，配置的磁盘旋转速度为每分钟 3000 转，每个磁道上记录两个数据块，每一块有 8000 字节，两个数据块之间间隙的越过时间为 2ms，主存存储周期为 500ns，存储器总线宽度为 16 位，总线带宽为 4MB/s。

（1）磁盘读写数据时的最大数据传输率是多少？

（2）当磁盘按最大数据传输率与主机交换数据时，主存存储周期空闲百分比是多少？

（3）直接寻址的“存储器-存储器”SS 型加法指令在无磁盘 I/O 操作打扰时的执行时间为多少？当磁盘 I/O 操作与一连串这种 SS 型加法指令执行同时进行时，则这种 SS 型加法指令的最快和最慢执行时间各是多少？（假定采用多周期处理器方式，CPU 时钟周期等于主存存储周期。）

10. 假定某计算机所有指令都可用两个总线周期完成，一个总线周期用来取指令，另一个总线周期用来存取数据。总线周期为 250ns，因而每条指令的执行时间为 500ns。若该计算机中配置的磁盘上每个磁道有 16 个 512 字节的扇区，磁盘旋转一圈的时间是 8.192ms，则采用周期挪用法进行 DMA 传送时，总线宽度为 8 位和 16 位的情况下该计算机指令执行速度分别降低了百分之几？

提问

Q & A

殷亚凤
智能软件与工程学院
苏州校区南雍楼东区225
yafeng@nju.edu.cn，https://yafengnju.github.io/

南京大学 NANJING UNIVERSITY