Chapter 6

输入输出系统

外设发展与分类

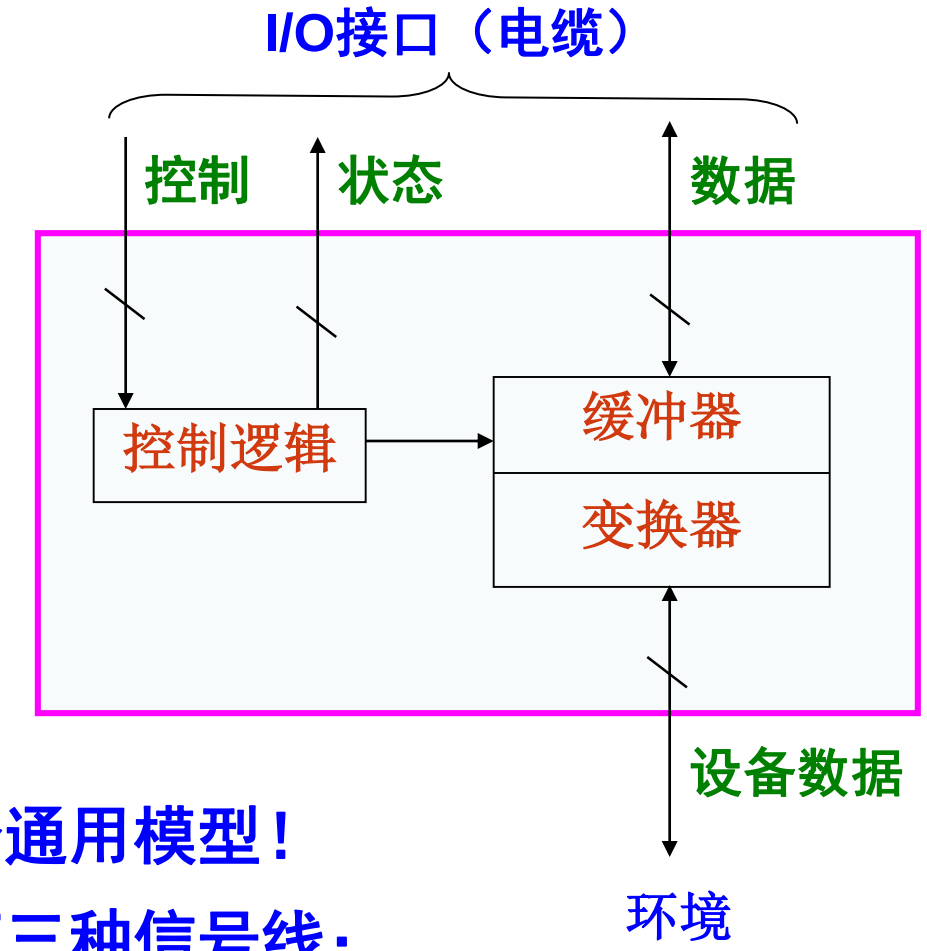
- 从交互方式上来分，外设分为：
  - 人-机交互设备，输入/输出的信息是人可读的
    - 如：键盘、鼠标、扫描仪、打印机、显示器等
  - 机器可读设备，输入/输出的信息是机器可读的，人无法读取
    - 如：网络、Modem、D/A、A/D、磁盘、声音输入设备等
- 从功能行为来分，外设分为：
  - 输入/输出设备（大部分为字符型设备），用于信息的输入/输出
    - 输入设备：键盘、鼠标、扫描仪等
    - 输出设备：打印机、显示器等
  - 外部存储设备（大部分为成块传送设备），用于信息的存储（其输入/出的信息是机器可读的）
    - 如：磁盘、磁带、光盘等

I/O Device Examples

Device	Behavior	Partner	Data Rate (KB/sec)
Keyboard	Input	Human	0.01
Mouse	Input	Human	0.02
Line Printer	Output	Human	1.00
Laser Printer	Output	Human	100.00
Graphics Display	Output	Human	30,000.00
Network-LAN	Input or Output	Machine	200.00
Floppy disk	Storage	Machine	50.00
Optical Disk	Storage	Machine	500.00
Magnetic Disk	Storage	Machine	2,000.00

外部设备的通用模型

- 通过电缆与计算机内部I/O接口进行数据、状态和控制信息的传送
- 控制逻辑根据控制信息控制设备的操作，并检测设备状态
- 缓冲器用于保存交换的数据信息
- 变换器用于在电信号形式（内部数据）和其他形式的设备数据之间进行转换



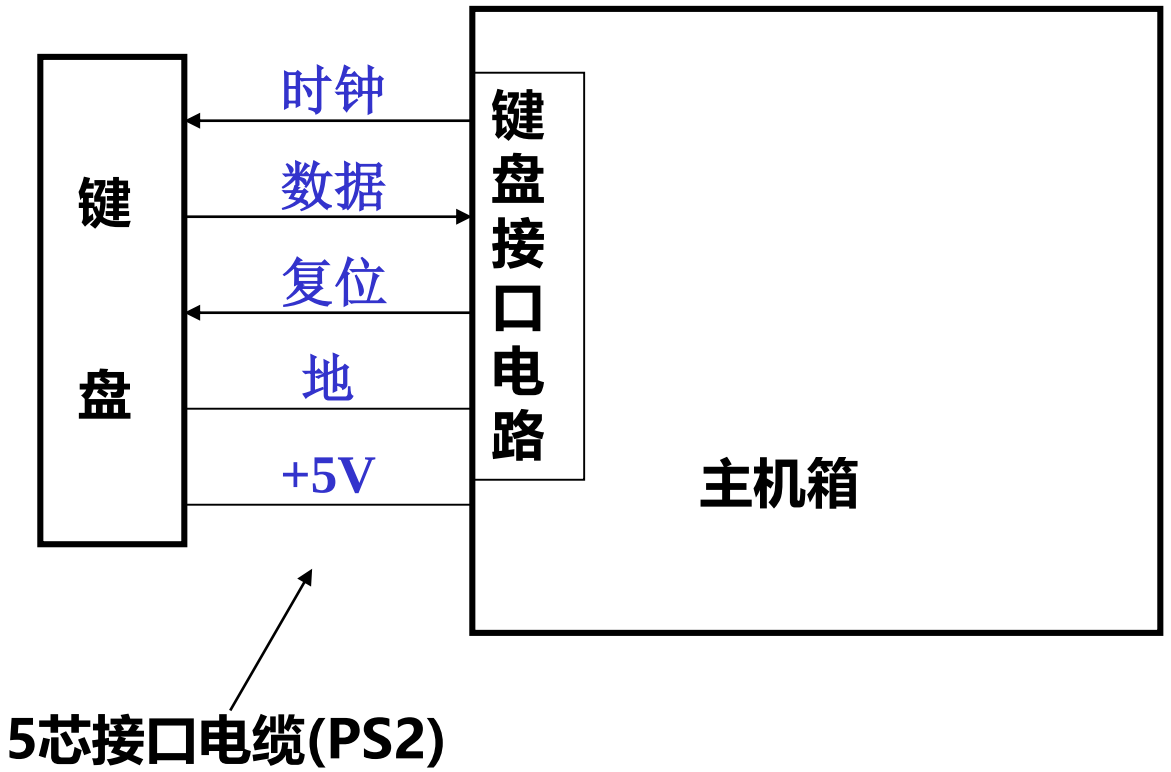
所有设备都可以抽象成这个通用模型！

设备所用的电缆线中有以下三种信号线：

控制信号、状态信号、数据信号

键盘(Keyboard)

键盘和主机的连接



目前，USB接口也很多！

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 (American Standard Code for Information Interchange)
  - 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。

表 2.1　ASCII 码表

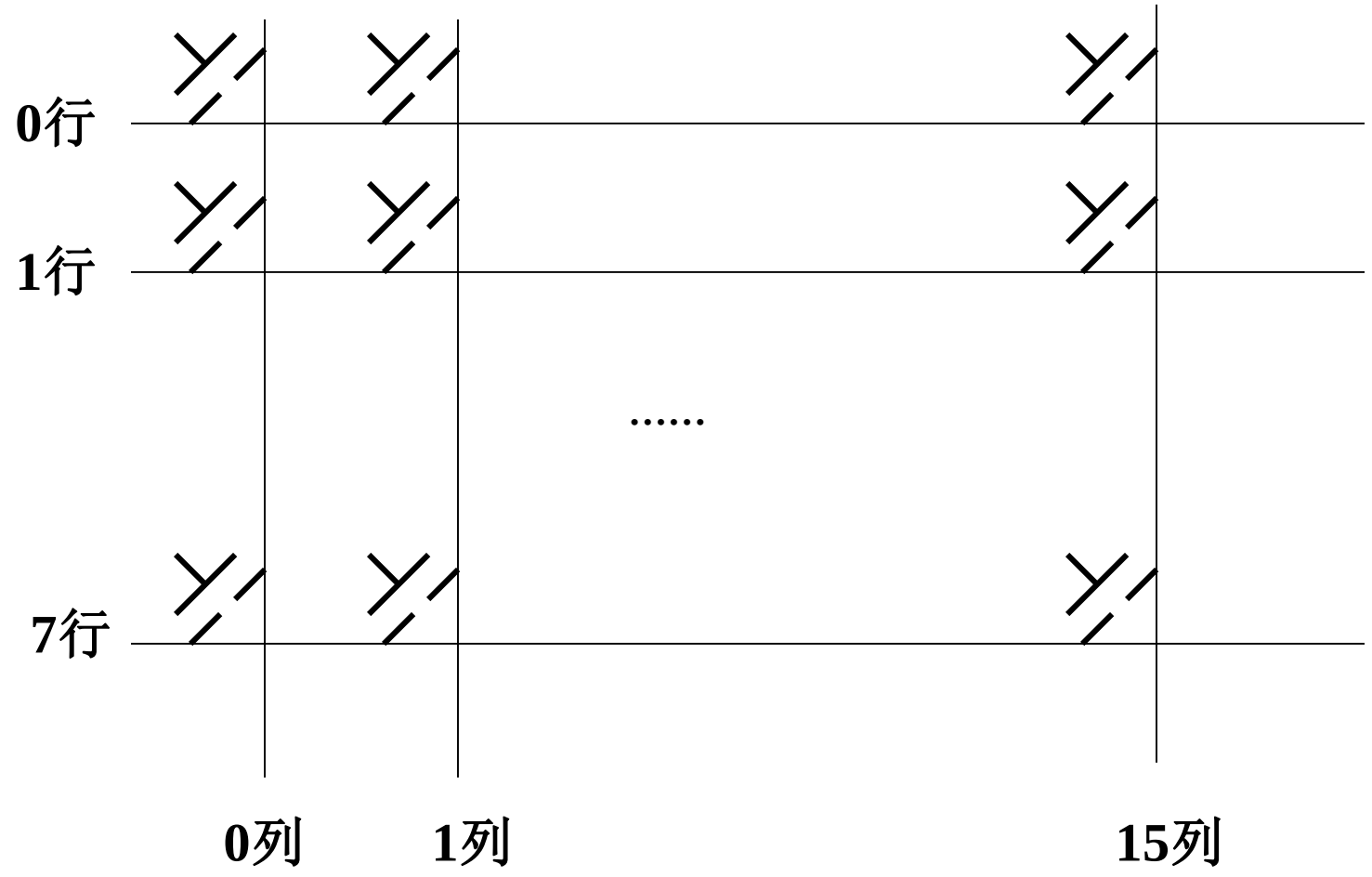
	$b_7b_6b_5$ =000	$b_7b_6b_5$ =001	$b_7b_6b_5$ =010	$b_7b_6b_5$ =011	$b_7b_6b_5$ =100	$b_7b_6b_5$ =101	$b_7b_6b_5$ =110	$b_7b_6b_5$ =111
$b_3b_2b_1b_0$=0000	NUL	DLE	SP	0	@	P	`	p
$b_3b_2b_1b_0$=0001	SOH	DC1	!	1	A	Q	a	q
$b_3b_2b_1b_0$=0010	STX	DC2	“	2	B	R	b	r
$b_3b_2b_1b_0$=0011	ETX	DC3	#	3	C	S	c	s
$b_3b_2b_1b_0$=0100	EOT	DC4	$	4	D	T	d	t
$b_3b_2b_1b_0$=0101	ENQ	NAK	%	5	E	U	e	u
$b_3b_2b_1b_0$=0110	ACK	SYN	&	6	F	V	f	v
$b_3b_2b_1b_0$=0111	BEL	ETB	‘	7	G	W	g	w
$b_3b_2b_1b_0$=1000	BS	CAN	(	8	H	X	h	x
$b_3b_2b_1b_0$=1001	HT	EM	)	9	I	Y	i	y
$b_3b_2b_1b_0$=1010	LF	SUB	*	:	J	Z	j	z
$b_3b_2b_1b_0$=1011	VT	ESC	+	;	K	[	k	{
$b_3b_2b_1b_0$=1100	FF	FS	,	<	L	\	l	|
$b_3b_2b_1b_0$=1101	CR	GS	-	=	M	]	m	}
$b_3b_2b_1b_0$=1110	SO	RS	.	>	N	^	n	~
$b_3b_2b_1b_0$=1111	SI	US	/	?	O	_	o	DEL

键盘

- 键盘输入信息过程
  - 按下一个键
  - 查出按下的是哪个键

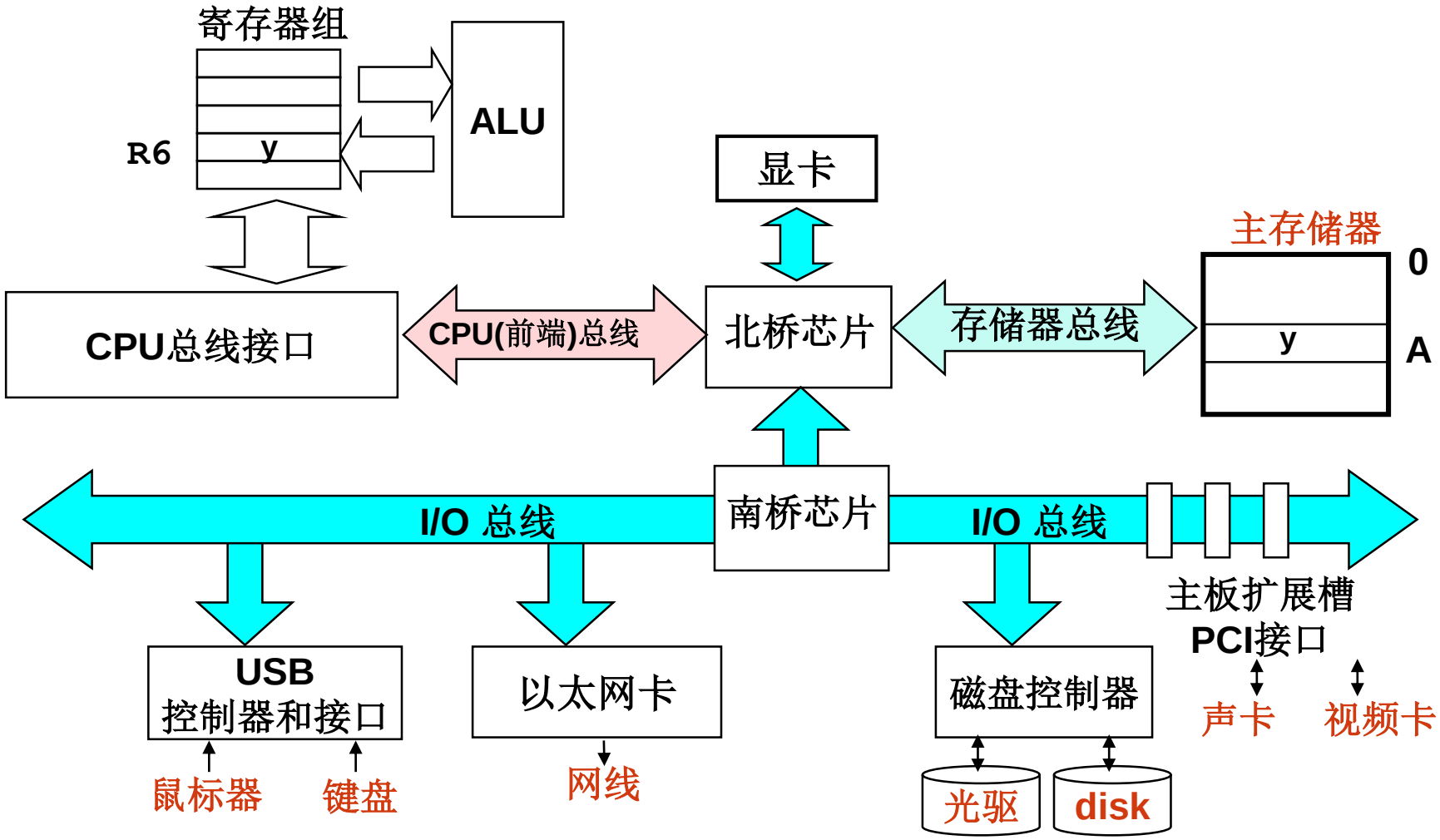
    ①行扫描法(微机系统中应用广)　　②线反转法
  - 将按键位置信息转换为对应的ASCII码，保存到计算机中。
- 按功能可分为以下两类：
  - 编码键盘：检测被按键，并提供相应的ASCII码送CPU。
  - 非编码键盘：只简单提供键盘的行列矩阵，识别按键位置，提供相应的位置码送CPU。

按键位置识别法之一：行扫描法



基本做法：按行扫描、接地检查

I/O总线、I/O控制器与I/O设备的关系



系统总线的组成

- 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。
- 现在总线设计的趋势是：点对点、异步、串行

系统总线的组成

- 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。

总线的性能指标

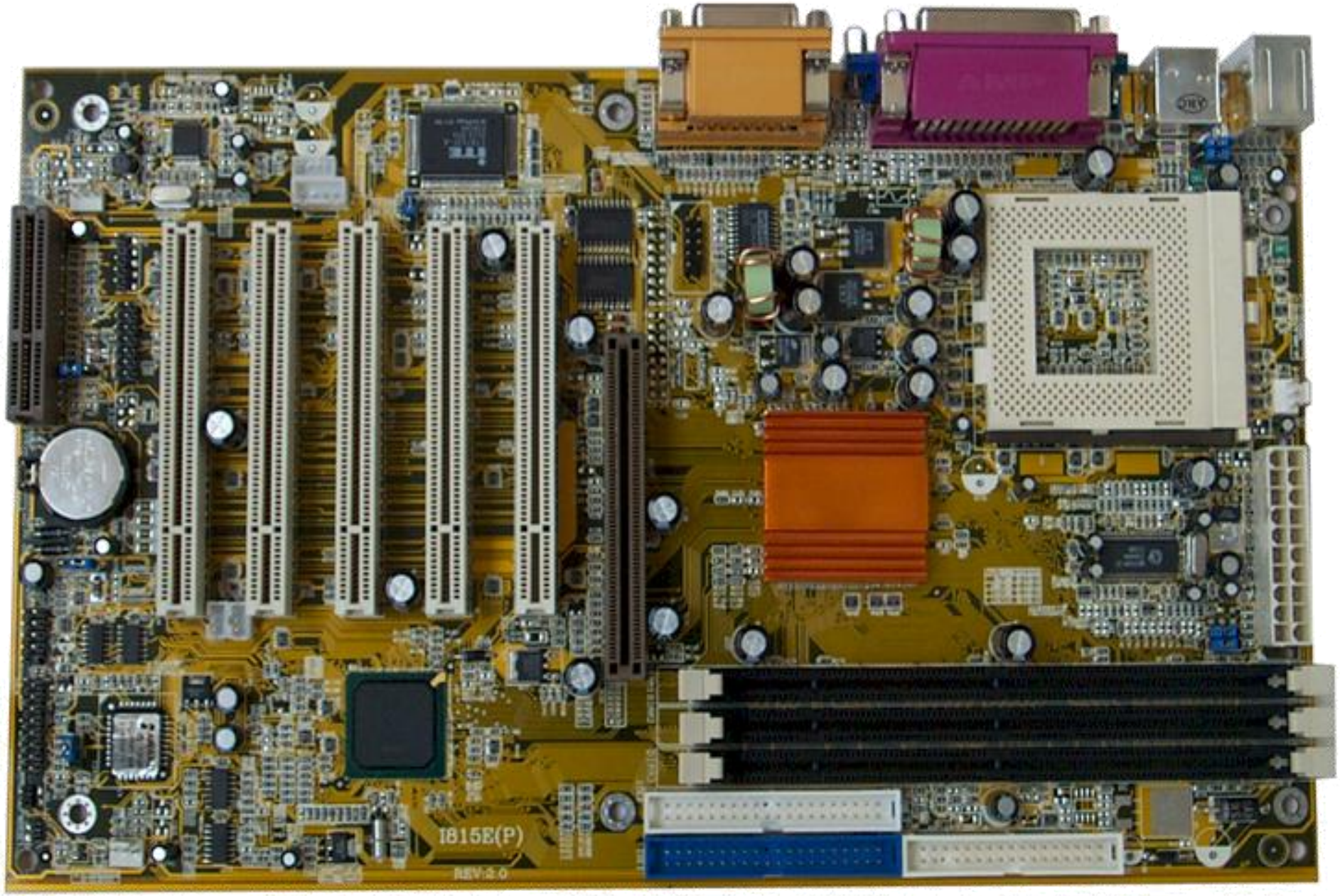
- 总线带宽
  - 总线的最大数据传输率
  - 对于同步总线，总线带宽计算公式：

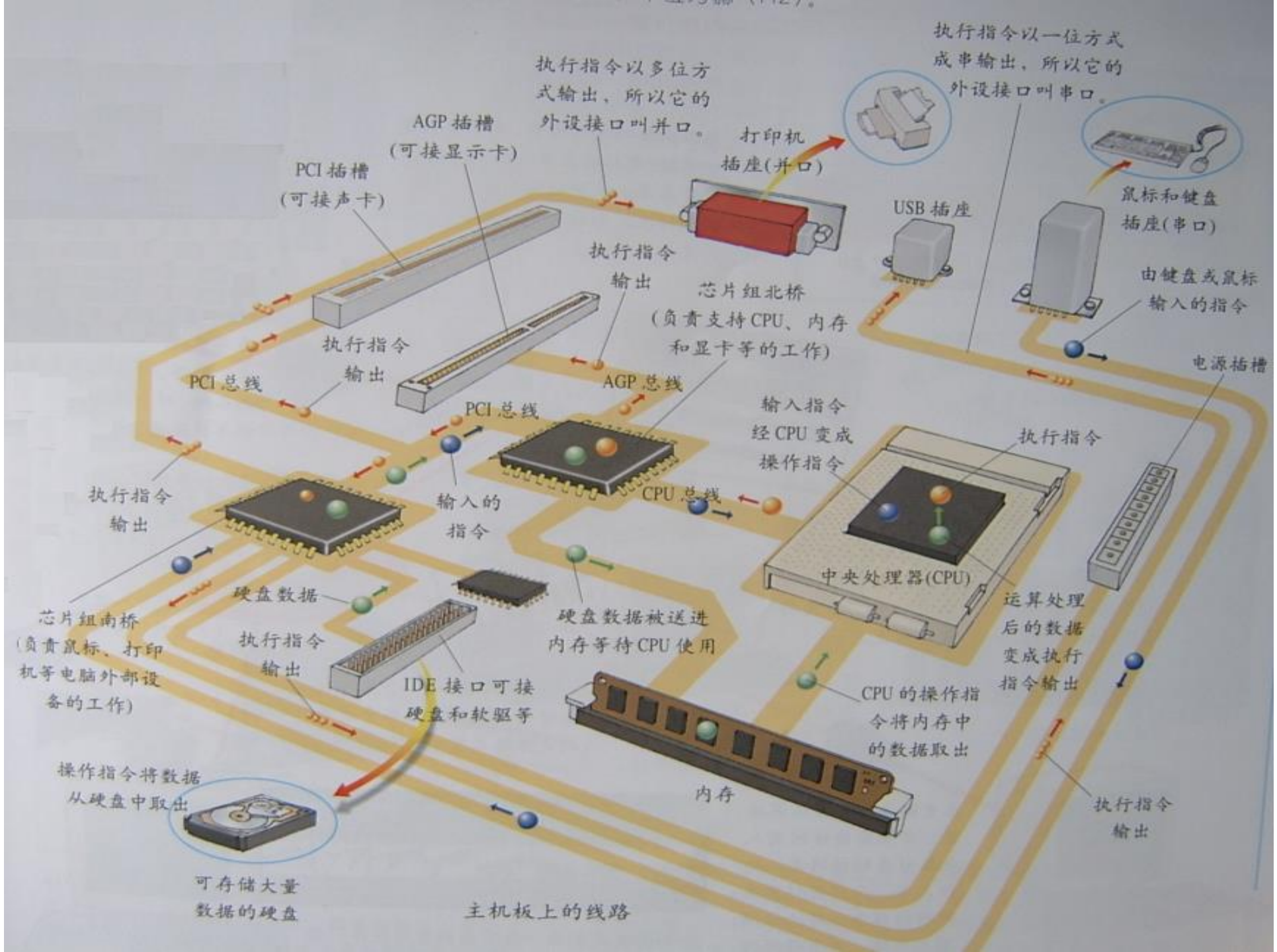
    B=W×F/N

    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

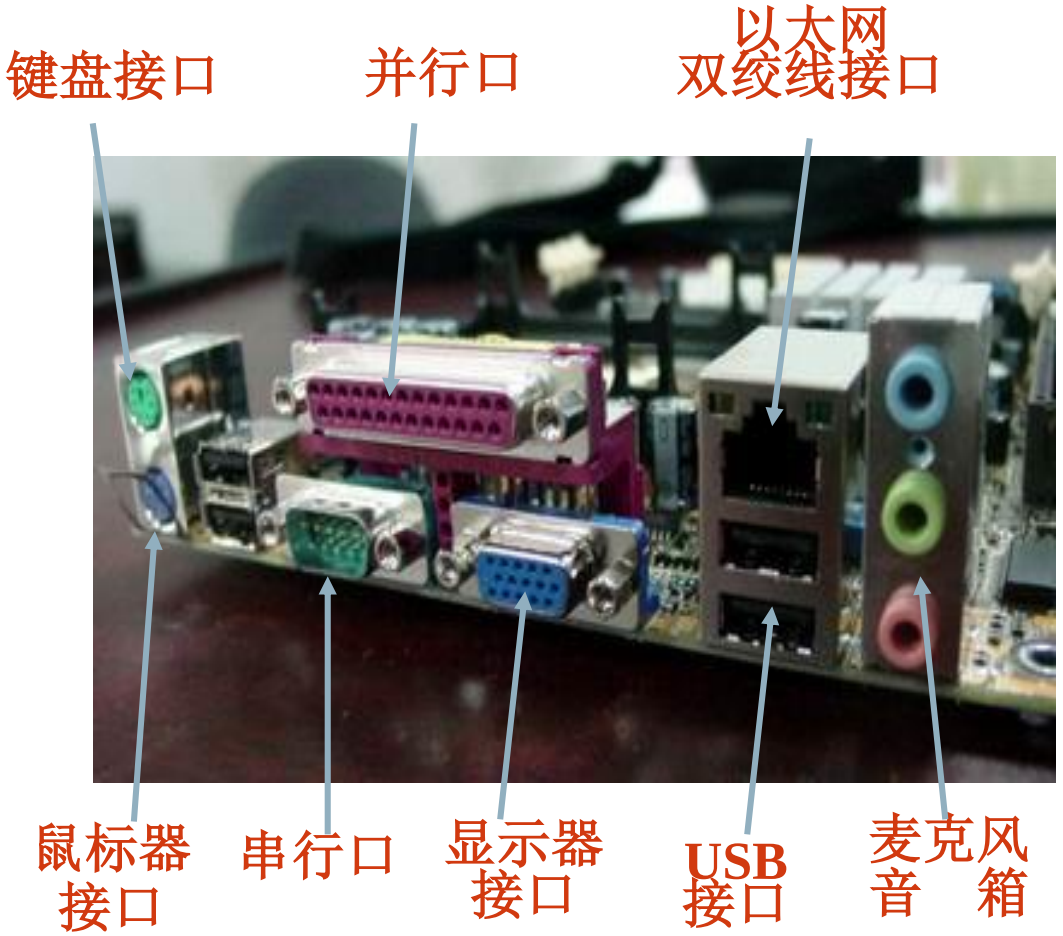
    F/N实际上就是总线工作频率

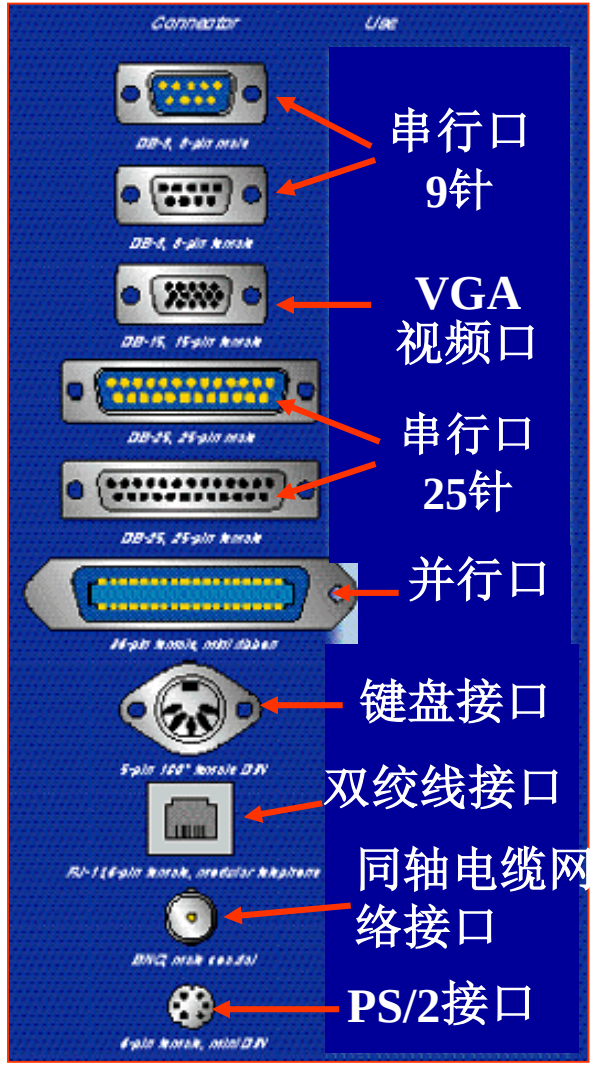
主板上的扩充插槽都是一种总线插座




执行指令以一位方式成串输出，所以它的外设接口叫串口。
执行指令以多位方式输出，所以它的外设接口叫并口。
AGP 插槽
(可接显示卡)
PCI 插槽
(可接声卡)
打印机
插座(并口)
USB 插座
鼠标和键盘
插座(串口)
执行指令
输出
芯片组北桥
(负责支持CPU、内存和显卡等的工作)
由键盘或鼠标输入的指令
执行指令
输出
PCI 总线
AGP 总线
电源插槽
PCI 总线
输入指令
经CPU 变成
操作指令
执行指令
执行指令
输出
输入的
指令
CPU 总线
中央处理器(CPU)
运算处理后的数据变成执行指令输出
硬盘数据
芯片组南桥
(负责鼠标、打印机等电脑外部设备的工作)
硬盘数据被送进内存等待CPU 使用
执行指令
输出
IDE 接口可接硬盘和软驱等
CPU 的操作指令将内存中的数据取出
内存
操作指令将数据从硬盘中取出
执行指令
输出
可存储大量数据的硬盘
主机板上的线路

I/O设备接口插座（连接器）



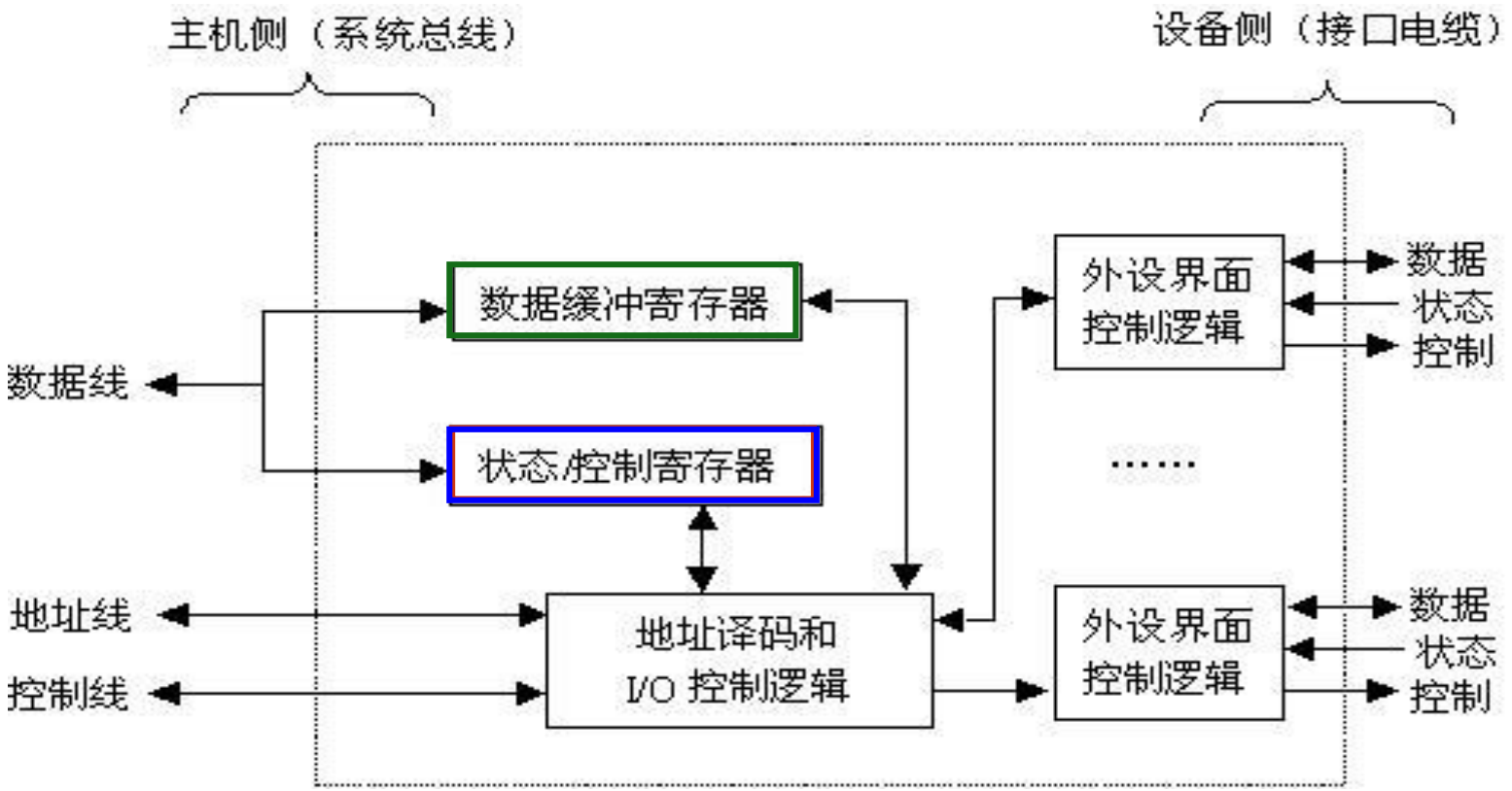


(安装在主板上的I/O设备接口插座)

I/O接口（I/O控制器）的功能

- 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（I/O控制器）的结构



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

不同I/O模块在复杂性和控制外设的数量上相差很大

I/O设备的寻址方式

- 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
- 统一编址方式
  - 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
- 独立编址方式
  - 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

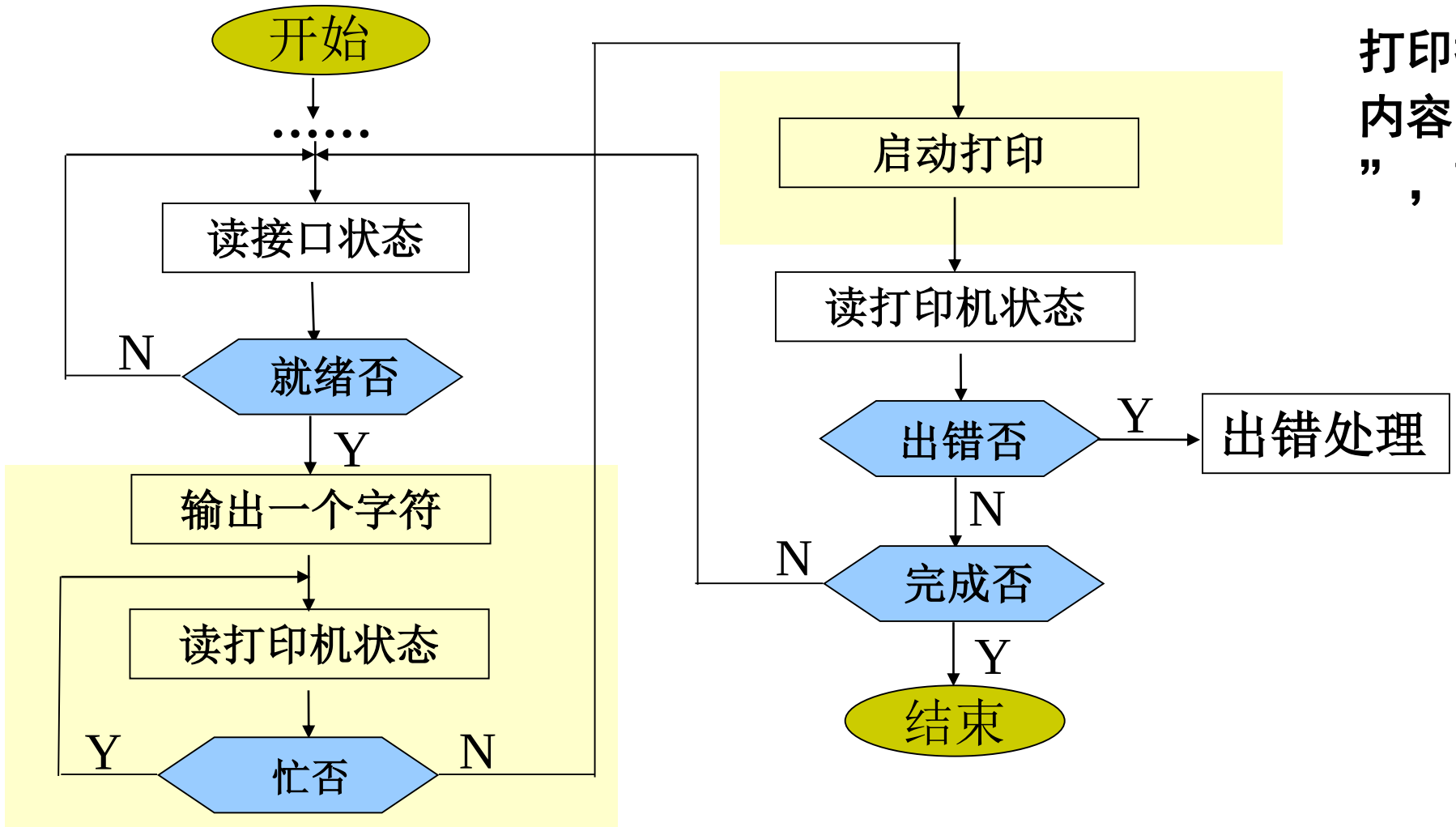
I/O设备与主机进行数据交换的三种基本方式

- 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

I/O设备与主机进行数据交换的三种基本方式

- Direct Memory Access (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

举例：用程序直接控制方式制打印输出

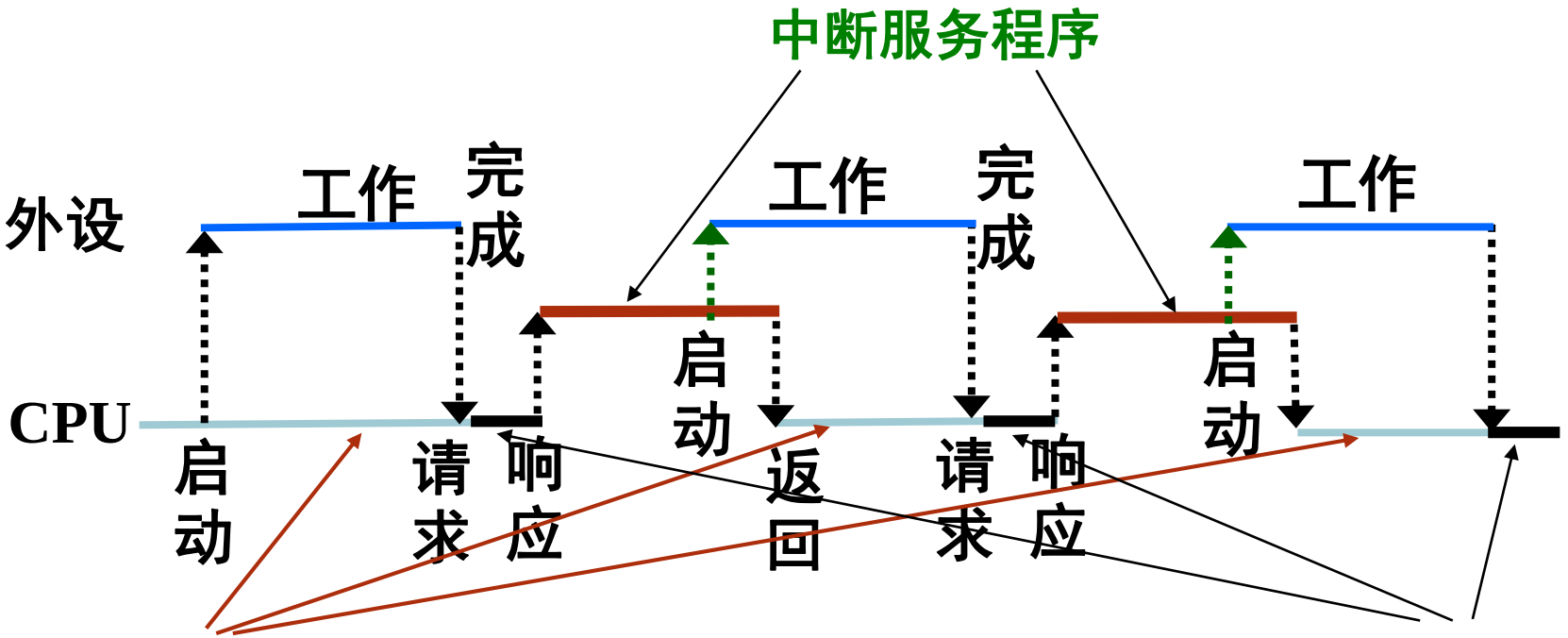


“就绪”的含义是什么？

打印控制器的数据缓冲中内容已被取走，现为“空”，可接受新的打印字符

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。



上述时段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”

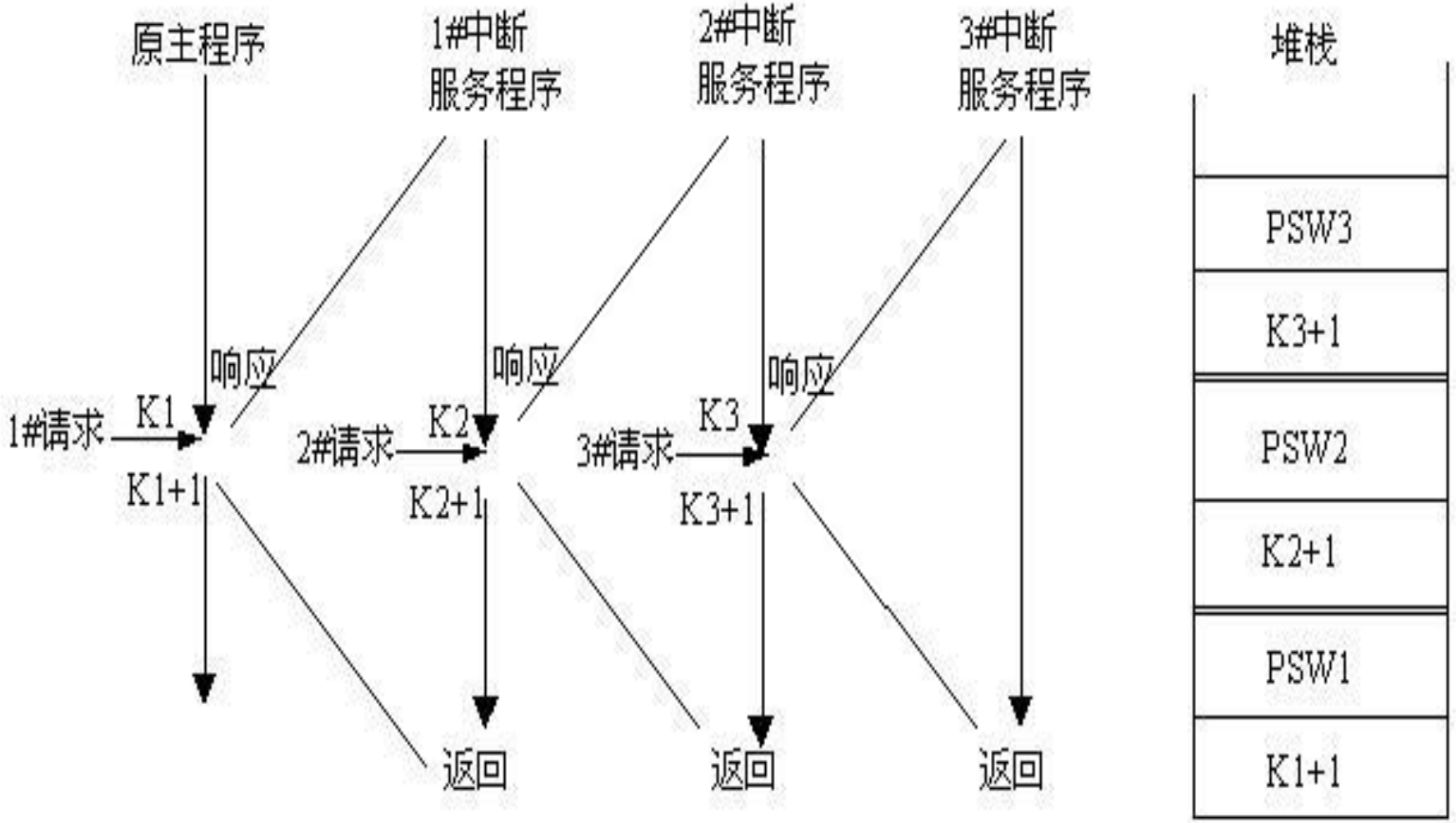
处理器中的异常（中断）处理机制

- 异常(中断)发生时，处理器做以下基本处理（执行中断隐指令以响应中断）
  - 首先，关中断（禁止中断）！即：将中断允许（触发器）标志清0。
  - 其次，保护断点和程序状态：将返回原程序执行的断点和程序状态保存到堆栈或特殊寄存器中：PC=>堆栈 或 EPC
  - 然后，识别异常事件并转到具体的异常处理程序执行，有两种不同方式
    - 软件识别（MIPS采用）：设置一个异常状态寄存器（MIPS中为Cause寄存器），用于记录异常原因。操作系统使用一个统一的异常处理程序（MIPS的入口为0x8000 0180），该程序按优先级顺序查询异常状态寄存器，识别出异常事件。
    - 硬件识别（向量中断）（80x86采用）：用专门的硬件查询电路按优先级顺序识别异常，得到一个“中断类型号”，根据此号，到中断向量表中读取对应的中断服务程序的入口地址。

多重中断的概念

- 多重中断：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
  - 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

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）

DMA输入/出方式

- 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

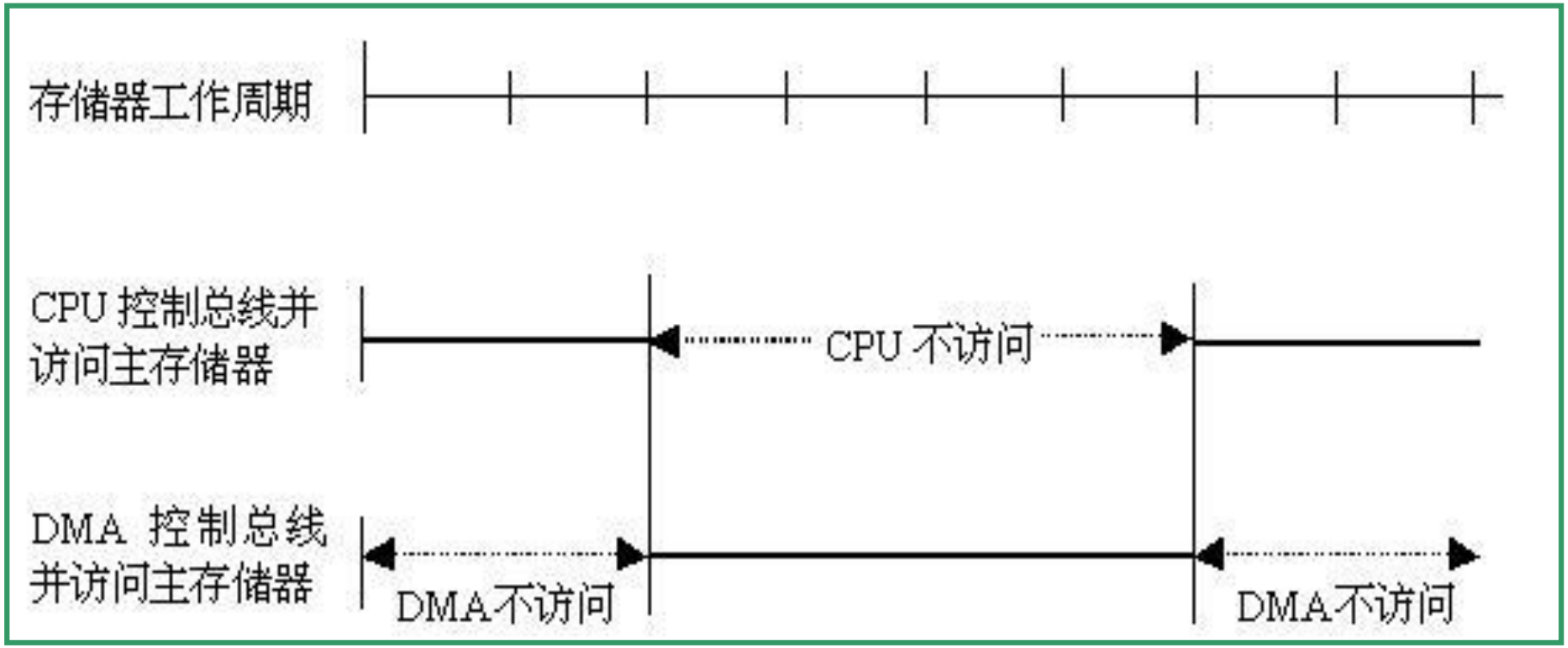
DMA方式的基本要点

- 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，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
- 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。
  - 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。
  - 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。
  - 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。

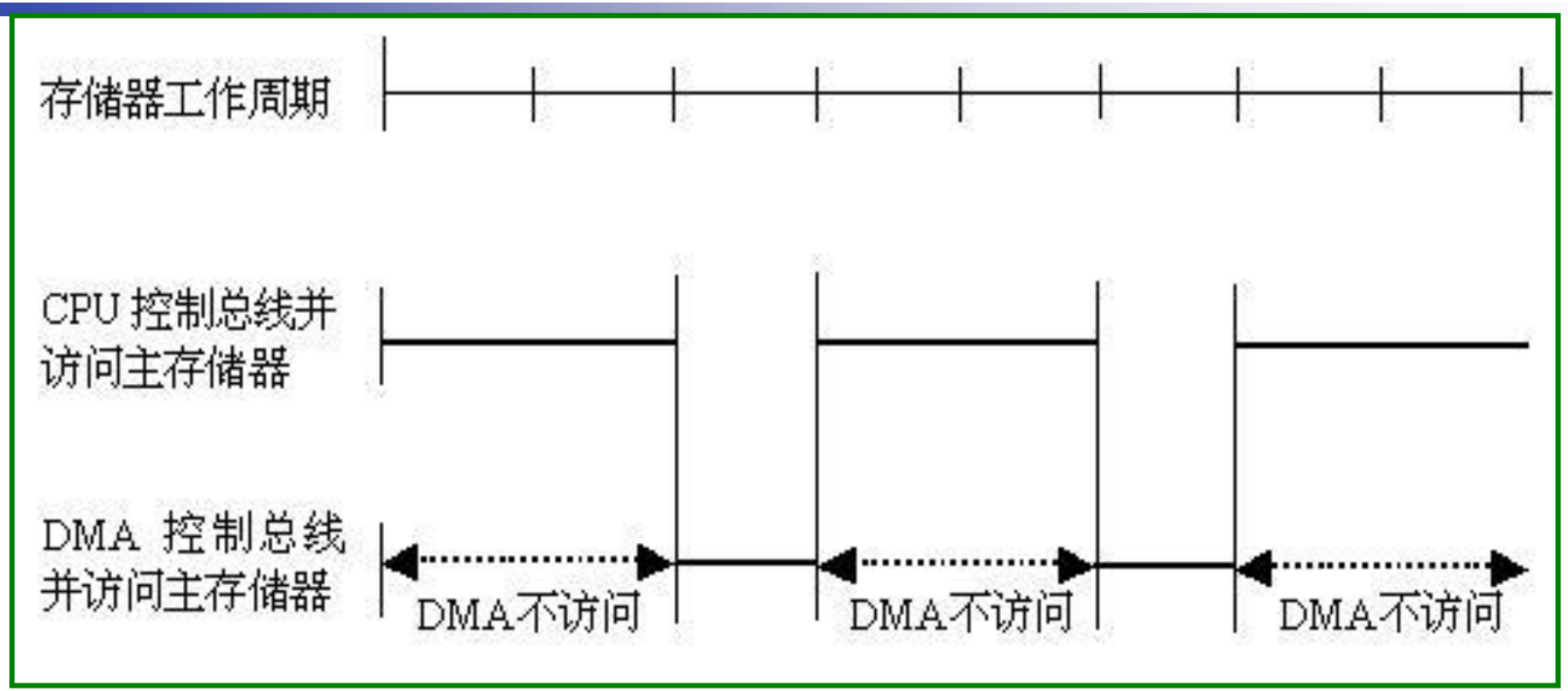
CPU停止法



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。可采用在DMA接口中引入缓冲器进行弥补。

周期挪用法

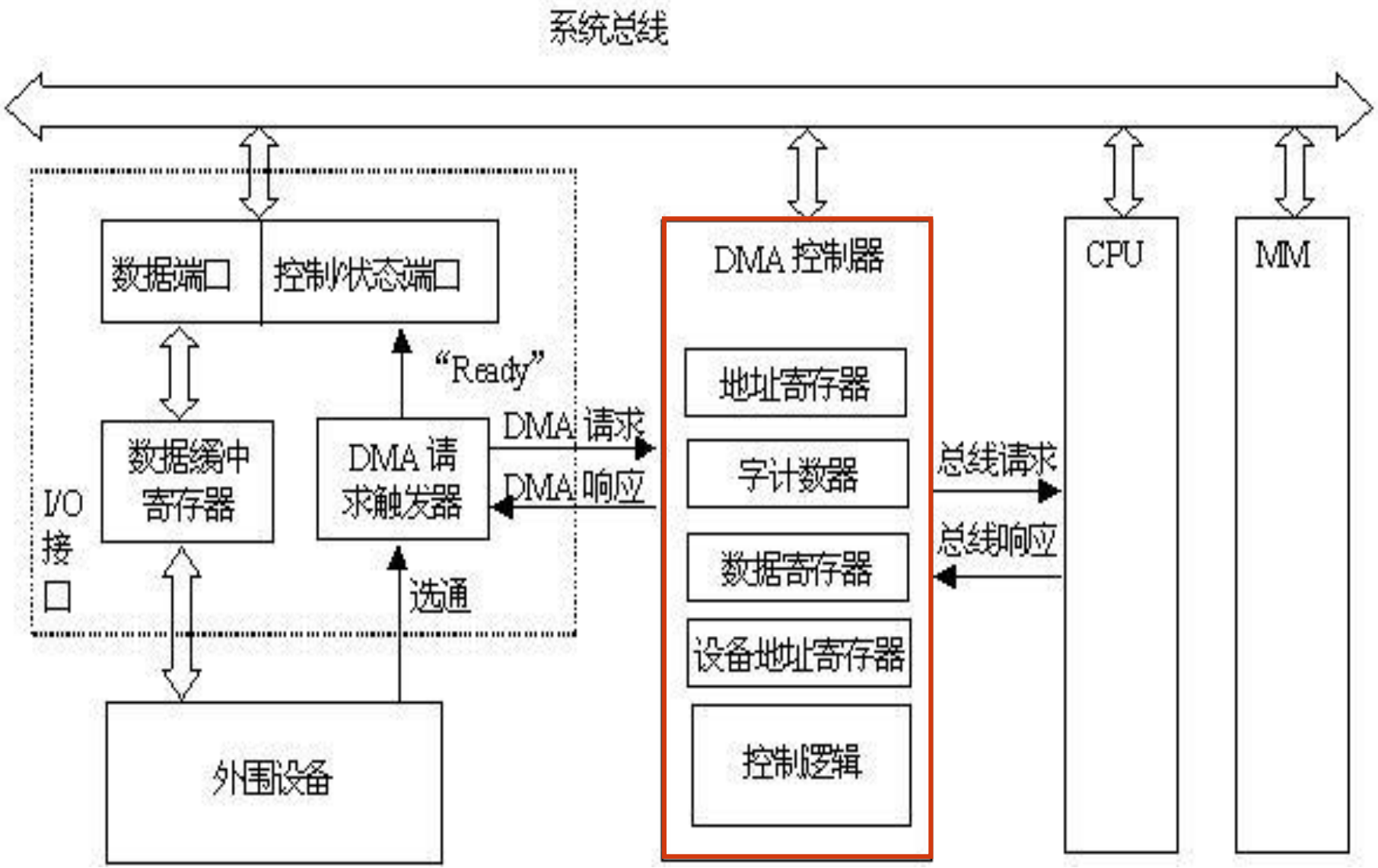


优点：既能及时响应I/O请求，又能较好地发挥CPU和主存的效率。

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。

DMA方式的系统逻辑结构



例：中断、DMA方式下CPU的开销

- 假设处理器频率500MHz，硬盘控制器中有一个16B的数据缓存器，磁盘传输速率为4MB/Sec，在磁盘传输数据过程中，要求没有任何数据被错过，并假定CPU访存和DMA访存没有冲突
  - 若用中断驱动I/O，每次传送的开销（包括用于中断响应和处理的时间）是500个时钟周期。如果硬盘仅用5%的时间进行传送，那么处理器用在硬盘I/O操作上所花的时间百分比（主机占用率）为多少？
  - 若用DMA方式，处理器花1000个时钟进行DMA传送的初始化设置，并且在DMA完成后的中断处理需要500个时钟。如果每次DMA传送8000B的数据块，那么当硬盘进行传送的时间占100%（即：硬盘一直进行读写，并传输数据）时，处理器用在硬盘I/O操作上的时间百分比（主机占用率）为多少？

例：中断、DMA方式下CPU的开销

- 中断传送：
  - 硬盘每次中断，以16字节为单位进行传送，为保证没有任何数据被错过（一旦磁盘被启动传送，就以4MB/s的速度进行！），应达到每秒4MB/16B=250k次中断的速度；
  - 每秒钟用于中断的时钟周期数为250k x 500=125x10^6；
  - 在一次数据传输中，处理器花费在I/O上的时间的百分比为：125x10^6/(500x10^6)=25%；
  - 假定硬盘仅用其中5%的时间来传送数据，则处理器花费在I/O方面的百分比为25% x 5%=1.25% 。

例：中断、DMA方式下CPU的开销

- DMA传送：
  - 每次DMA传送将花费8000B/(4MB/Sec)≈$2x10^{-3}$秒；
  - 一秒钟内有$1/(2x10^{-3})$=500次DMA传送；
  - 如果硬盘一直在传送数据的话，处理器必须每秒钟花(1000+500)x500=$750x10^3$个时钟周期来为硬盘I/O操作服务；
  - 在硬盘I/O操作上处理器花费的时间占：

    $750x10^3/(500x10^6)=1.5x10^{-3}$=0.15%

例：中断、DMA方式下CPU的开销

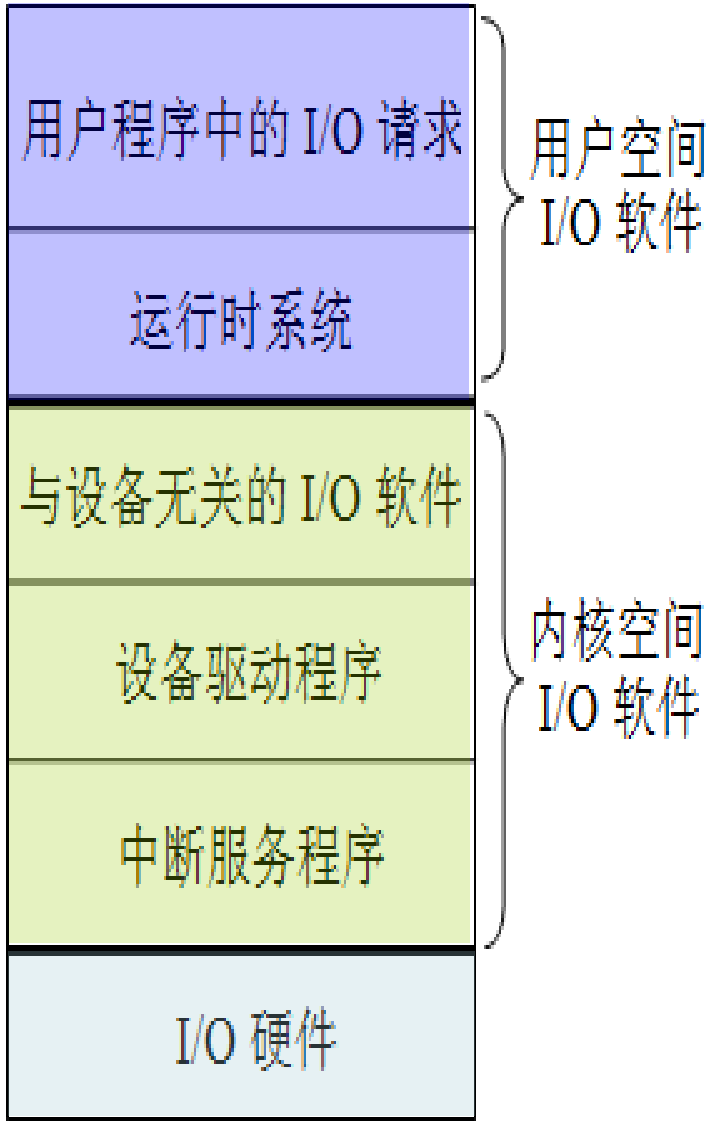
- 想象一下：假定大仓库门口有一个箱子，可放16个零件。要将大仓库中的一批零件运到小仓库中，可以有几种方法?
  - 中断方式：每装满一个箱子就喊车床上的技工来运到车间，再从车间运到小仓库
  - DMA方式：车床技工停下来告诉搬运工说，一次要8000个零件放到小仓库固定的地方，然后回到车床工作；搬运工开始分两组工作，一组从大仓库搬货到箱子中，另一组将箱子直接运到小仓库指定地方，8000个运完后，技工再停下来检查；然后继续下一次8000个零件的搬运，……

- 上述两种方式中，哪种方式的生产效率更高呢?

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）

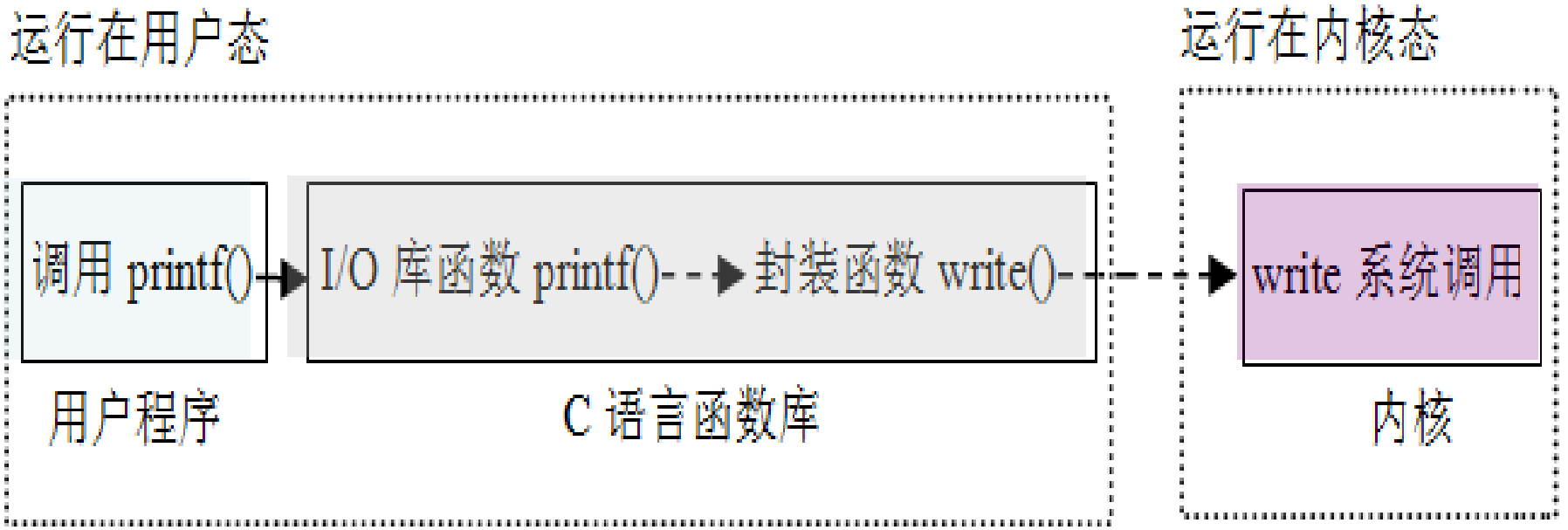


操作系统在I/O中扮演的角色

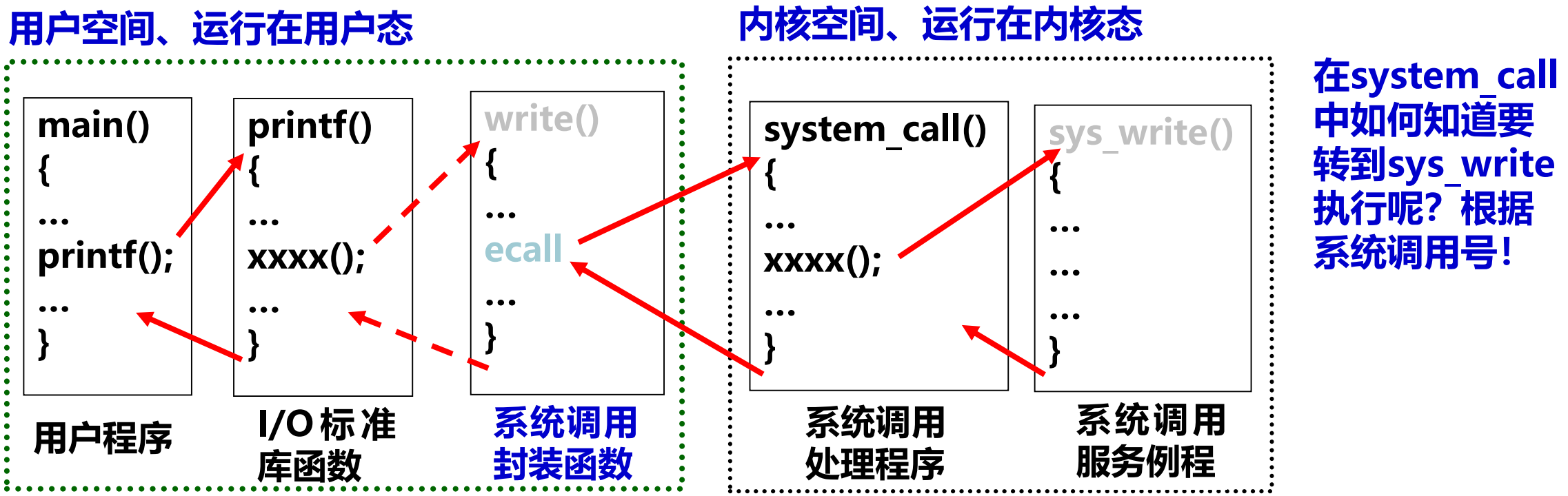
- OS的职责由I/O系统的三个特性决定
  - 共享性：I/O系统被处理器执行的多个程序共享，由OS统一调度管理
  - 复杂性：I/O设备控制的细节比较复杂，不能由上层用户程序来实现，需OS提供专门的驱动程序
  - 采用中断I/O方式：I/O系统通常使用外部中断请求来要求处理器执行专门的输入/出程序。中断导致向内核态转移，故必须由OS来处理
- OS在I/O中的职能
  - 保证用户程序只能访问自己有权访问的那部分I/O设备
  - 为用户程序提供设备驱动程序以屏蔽设备控制的细节
  - 处理外部I/O中断，提供中断服务程序
  - 对共享的I/O资源提供合理的调度管理，使系统的吞吐率达到最佳

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。

设备无关I/O软件层

- 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

设备无关I/O软件层

- 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。

设备无关I/O软件层

- 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

设备驱动程序

- 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 。
  - 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