输入输出系统

一、输入输出系统

1、输入输出系统是控制外设与主存、外设和 CPU 之间进行数据交换的软硬件系统。输入输出系统包含了 I/O 设备（外设）、I/O 接口、参与 I/O 任务的专用软件（CPU 通过执行 I/O 指令和 I/O 控制管理程序，对 I/O 硬件的行为进行控制）以及在一定程度上可以认为总线也是其中的一部分。

2、I/O 系统的特点：

- 异步性：外围设备相对于 CPU 通常是异步工作的，即两者时钟不同
- 实时性：当外设和 CPU 进行交互时，由于设备的类型不同，它们的工作步调是不同的，CPU 必须按照不同设备要求的传送方式和传输速率不失时机地位外设提供服务，这就要求实时性控制
- 与设备无关性：各种外设必须根据其特点和要求选择一种标准接口和 CPU 进行连接，它们之间的差别必须由设备本身地控制器通过硬件和软件来填补；这样，CPU 无须了解外设的具体细节，可以采用统一的硬件和软件对其管理

3、数据输入/输出的大致过程：

对于输入：

1）CPU 通过地址总线选定 I/O 接口

2）CPU 等待输入设备输入的数据（实则是等待 I/O 接口的状态寄存器）有效

3）CPU 通过数据总线读入数据

对于输出：

1）CPU 通过地址总线选定 I/O 接口

2）CPU 通过数据总线将数据放入到 I/O 接口的输出缓冲寄存器当中，

3）外设认为数据有效，取走寄存器中的数据，并将完成的消息通知 CPU 或等待 CPU 来检查

注意：上面 CPU 检查或者通知 CPU 完成工作这件事就需要考虑到 I/O 控制方式，即：该输入输出系统是如何具体实现输入输出功能的

4、I/O 系统地性能评价

- 系统响应时间：完成一次 I/O 事务所需要的时间
- 吞吐率：单位时间内完成地输入/输出操作次数，通常用 IOP 表示
- I/O 带宽：单位时间内从系统输入/输出地数据量

二、I/O 接口

在现代计算机系统中，由于输入输出设备（外设）千差万别，使得输入输出设备不能直接与 CPU 进行相连，因此我们需要在输入输出设备与 CPU 之间设置一个中转站，即 I/O 接口

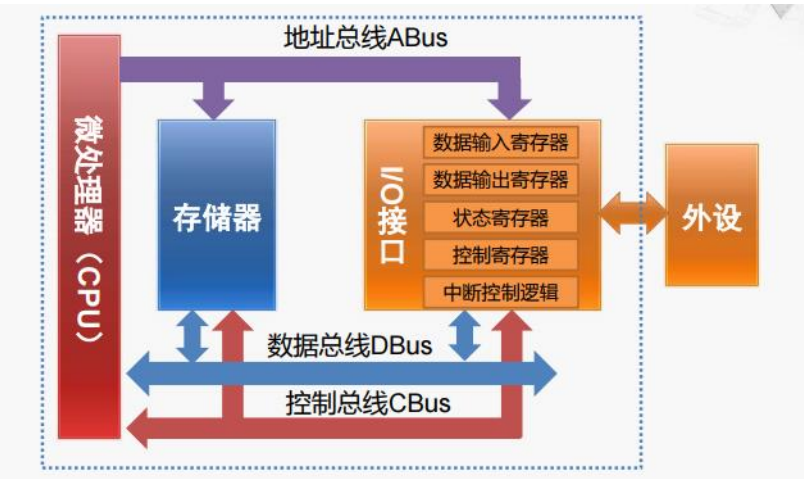
1、I/O 接口的基本功能如下：

1. 数据缓冲
   - 解决CPU和外设之间的速度差距
2. 提供联络信息
   - 协调与同步数据交换过程
3. 信号与信息格式的转换
   - 模/数、数/模转换，串/并、并/串转换，电平转换
4. 设备选择
5. 中断管理
6. 可编程功能

2、I/O 接口与 I/O 设备（外设）在现代计算机中的基本结构：

在现代计算机系统中，往往把 I/O 相关的设备分解成 I/O 接口（由芯片或者插口组成，其内部有寄存器，CPU 通过系统总线访问 I/O 接口中的寄存器，而 I/O 接口会有一些管脚与外部的设备相连）与外设（一般认为的 I/O 设备）。这与冯诺依曼系统中定义的输入输出设备有着一定的出入，我们这里描绘的是现代计算机的一种具体实现。

I/O 接口在一端通过系统总线与 CPU 相连，而在另一端通过串行或者并行方式与外设相连。

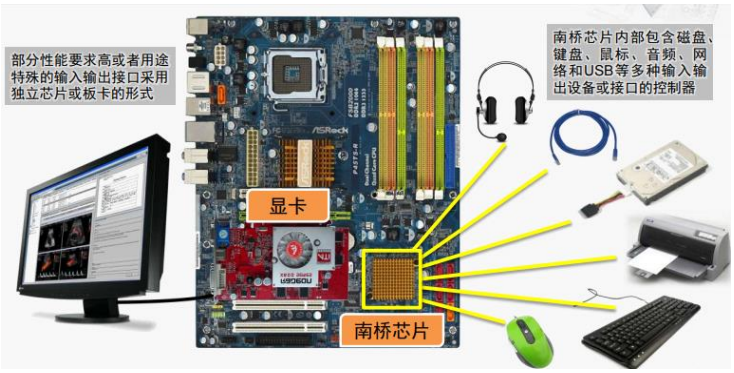


3、I/O 接口示例：

并行接口电路是 I/O 接口的一种，一般有两种常见的物理实现方式：1）独立的芯片 2）包含在多功能的芯片中

下面就是两种比较常见的连接形式：

在现代计算机中，上述的并行接口会与其他 I/O 接口集成在南桥芯片上，而少数对性能要求比较高的 I/O 接口会以独立芯片或者板卡的方式存在，而在一些紧凑型的设备当中，比如嵌入式设备，CPU 与 I/O 接口一般会集成在一个芯片当中，即 SoC



三、I/O 接口的编址方式

I/O 端口：

- I/O端口
  - I/O接口内部包含一组称为I/O端口的寄存器
  - 每个I/O端口都需有自己的端口地址（或称端口号），以便CPU访问

1、I/O 端口的编址方式：

- I/O端口和存储器分开编址
  - I/O映像的I/O方式，I/O Mapped I/O
  - x86体系结构采用该方式
- I/O端口和存储器统一编址
  - 存储器映像的I/O方式，Memory Mapped I/O
  - ARM、MIPS、PowerPC等体系结构采用该方式

2、独立编址：

- 对 I/O 端口单独编号，使之成为独立的 I/O 地址空间，存储单元地址和外设地址毫无关系
- 需要专门的 I/O 指令对其进行操作

X86 中采用 IN 和 OUT 指令达到对 I/O 接口输入输出数据的目的

IN指令（输入）
- 格式：IN AC, PORT
- 操作：把外设端口的内容输入到AL或AX

OUT指令（输出）
- 格式：OUT PORT, AC
- 操作：把AL或AX的内容输出到外设端口

对于 IN/OUT 指令的寻址方式有直接寻址和间接寻址两种方式，之所以要设定两种寻址方式，主要是考虑到了指令的长度。对于直接寻址方式的指令长度为 2 字节，能够寻址 256 个 I/O 端口，若要寻址范围超过 256 个，显然必须要延长指令的长度。为了缩短指令的长度，x86 设计者宁可增加一个指令完成相应的功能，这也是 CISC 的特点。对于间接寻址 I/O 端口的 IN 指令，它默认采用 DX 寄存器保存需要输入的 I/O 端口地址。特别的，需要一个 M/IO 的信号去区分当前要访问是 I/O 接口还是主存

- 端口地址为0~255
  - 直接寻址：用一个字节立即数指定端口地址
  - 间接寻址：用DX的内容指定端口地址
- 端口地址大于255
  - 间接寻址：用DX的内容指定端口地址

	76543210	76543210
IN：直接寻址	1110010w	port
IN：间接寻址	1110110w	
OUT：直接寻址	1110011w	port
OUT：间接寻址	1110111w	

```
IN   AL, 80H
IN   AX, 80H
OUT  80H, AL
OUT  80H, AX
```

```
MOV  DX, 288
IN   AL, DX
IN   AX, DX
OUT  DX, AL
OUT  DX, AX
```

3、统一编址：

- 将 I/O 端口映射到主存空间的某个区域，与主存统一编址，将主存空间分出一部分地址给 I/O 端口进行编号
- 将外设接口中可访问的寄存器和存储器的存储单元同等对待，可用访存指令去访问 I/O 接口中的寄存器

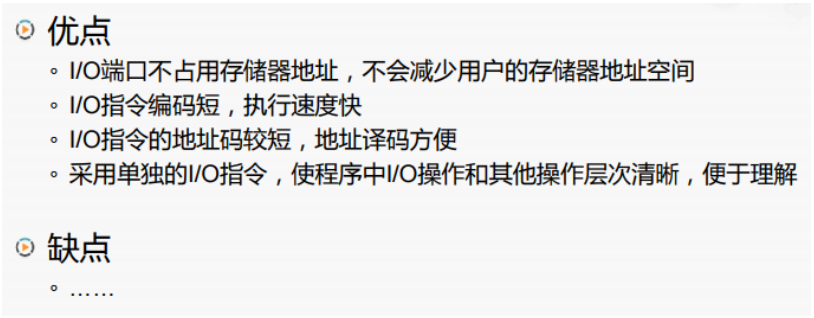
两种编址方式的优缺点：

统一编址方式的优缺点：

- 优点
  - 可以用访问存储器的指令来访问I/O端口，访问存储器的指令功能比较齐全，可以实现直接对I/O端口内的数据进行处理
  - 可以将CPU中的I/O操作与访问存储器操作统一设计为一套控制逻辑，简化内部结构，同时减少CPU的引脚数目
- 缺点
  - 由于I/O端口占用了一部分存储器地址空间，因而使存储地址空间减小
  - 由于利用访问存储器的指令来进行I/O操作，指令的长度通常比单独I/O指令要长，因而指令的执行时间也较长

但是随着机器字长的增加，统一编址使得存储地址空间减少的问题显然不在成问题。同时由于 RISC 指令都是固定长度的，所以即使对 I/O 端口采用独立编址方式设置单独的 I/O 指令也不会比普通的访存指令更短一些，而 x86 是 CISC 采用变长指令，所以可以设计专门用于 I/O 操作的更短的指令，从而提高指令密度。这也是 RISC 普遍采用统一编址方式的原因之一。

独立编址的特点：



四、I/O 控制方式

I/O 控制方式的含义：CPU 如何控制外设的数据传送，在一些教科书也叫做输入输出方式，即在实际当中如何实现输入输出。

I/O 控制方式的分类：

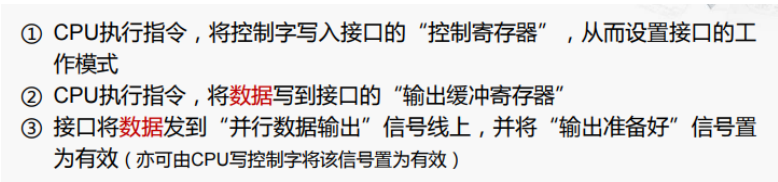
- 程序控制方式
- 中断控制方式
- 直接存储器访问方式 DMA

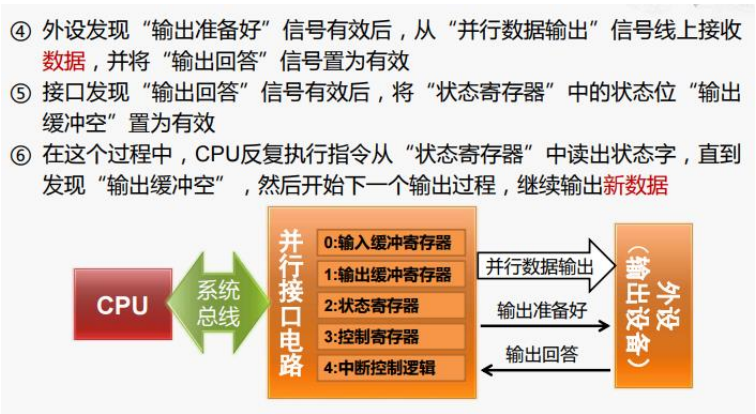
1、程序控制方式

• 无条件传送方式：在程序适当位置直接安排 I/O 指令，当程序执行到这些 I/O 指令的时候，CPU 默认外设始终是准备就绪的（I/O 接口总是准备好接受 CPU 的输出数据，或总是准备好向 CPU 输入数据），无须检查 I/O 接口的状态寄存器，就直接进行数据传输。这种方式适合于硬件接口电路和软件控制程序都比较简单的设备

• 程序查询传送方式（轮询）：数据在 CPU 和外设之间传送全靠程序控制，CPU 执行 I/O 指令时，先获取外设状态（访问 I/O 接口的状态寄存器），并根据外设的状态决定下一步操作

程序查询方式的数据输出过程：





程序查询方式的数据输入过程：

① 系统初始化时，CPU执行指令，将控制字写入接口的“控制寄存器”，设置接口的工作模式
② 外设将数据发到“并行数据输入”信号线上，并将“输入准备好”信号置为有效
③ 接口发现“输入准备好”信号有效后，从“并行数据输入”信号线上接收数据，放入“输入缓冲寄存器”，并将“输入回答”信号置为有效，阻止外设输入新数据

④ 接口将“状态寄存器”中的状态位“输入缓冲满”置为有效
⑤ 在上述过程中，CPU反复执行指令从“状态寄存器”中读出状态字，直到发现“输入缓冲满”，然后执行指令从“输入缓冲寄存器”中读出数据
⑥ 接口将“输入回答”信号置为无效，等待外设输入新数据

上面两种传送方式的优缺点：

- 无条件传送方式
  - 优点：控制程序简单
  - 缺点：只适用于简单外设的操作
- 程序查询传送方式
  - 优点：比无条件传送方式准确和可靠
  - 缺点：查询外设状态占用了大量的时间
- 共同的优缺点
  - 优点：对外设的要求低，操作流程清晰
  - 缺点：由CPU进行数据传送操作，占用了宝贵的运算资源

2、中断控制方式

当外设已经完成某项操作或者需要引起注意的时候，主动通知 CPU，CPU 接收到由外设引发的中断后暂停现行的工作，转入中断服务程序，等中断服务程序处理完毕后，再返回到被中断的原程序中继续以前被暂停的工作

中断控制方式数据输入过程：

① 系统初始化时，CPU执行指令，将控制字写入接口的“控制寄存器”，设置接口的工作模式
② 外设将数据发到“并行数据输入”信号，并将“输入准备好”信号置为有效
③ 接口发现“输入准备好”信号有效后，从“并行数据输入”信号接收数据，放入“输入缓冲寄存器”，并将“输入回答”信号置为有效，阻止外设输入新数据

④ 接口通过“中断控制逻辑”向CPU发出中断请求信号，并将“状态寄存器”中的状态位“输入缓冲满”置为有效
⑤ CPU收到中断请求后，进入中断服务程序，执行指令从“状态寄存器”中读出状态字，发现“输入缓冲满”，因此执行指令，从“输入缓冲寄存器”中读出数据
⑥ 接口将“输入回答”信号置为无效，等待外设输入新数据

需要注意：采用中断控制方式的 I/O 系统，I/O 接口不仅需要产生 I/O 中断，还需要给出像确认哪个设备产生中断这类更进一步的信息，以及中断设备所具有的优先级。为了向处理器传递诸如中断的设备标识这类信息，系统采用向量中断或者 Cause 寄存器。

中断控制方式数据输出过程：

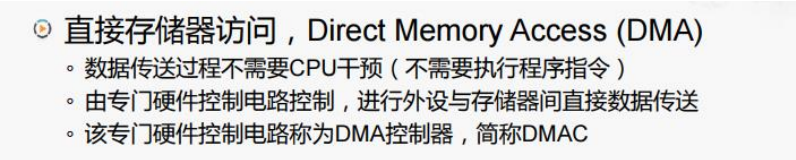
① CPU执行指令，将控制字写入接口的“控制寄存器”，从而设置接口的工作模式
② CPU执行指令，将数据写到接口的“输出缓冲寄存器”
③ 接口将数据发到“并行数据输出”信号，并将“输出准备好”信号置为有效（亦可由CPU写控制字将该信号置为有效）

④ 外设发现“输出准备好”信号有效后，从“并行数据输出”信号接收数据，并将“输出回答”信号置为有效
⑤ 接口发现“输出回答”信号有效后，通过“中断控制逻辑”向CPU发出中断请求信号，并将“状态寄存器”中的状态位“输出缓冲空”置为有效
⑥ CPU收到中断请求后，进入中断服务程序，执行指令从“状态寄存器”中读出状态字，发现“输出缓冲空”，因此开始下一个输出过程，继续输出新数据

中断控制方式的特点：

- 优点
  - CPU可以和外设并行工作，提高了工作效率
  - 外围设备具有申请服务的主动权
  - 一定程度上满足了I/O处理的实时性要求
- 缺点
  - 外设和存储器之间的数据交换仍由CPU承担
    - 使用数据传送指令，占用了宝贵的CPU运算资源
    - 数据要经过CPU中的通用寄存器中转，过程冗长
    - （注：程序查询方式同样有这些缺点）
  - 进入和退出中断服务程序，需要额外的指令

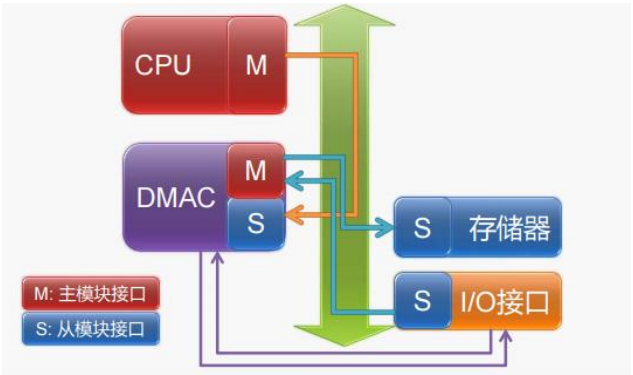
3、直接存储器访问（DMA）方式



需要注意的是 DMA 控制器本身就是一个 I/O 接口

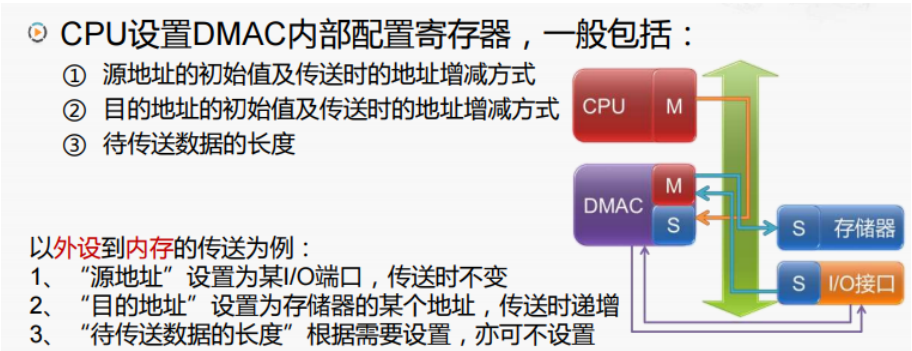
一次 DMA 传输有以下几个步骤：
1）CPU 设置 DMA 控制器内部配置寄存器，通过使用 OUT 指令完成
2）DMA 控制器处于空闲等待状态，此时 CPU 可以执行别的工作
3）I/O 接口向 DMAC 发出 DMA 传送请求
4）DMAC 响应 I/O 接口的请求
5）DMAC 向 I/O 接口发起总线读传输
6）DMAC 向存储器或者是其他目标地址发起总线写传输
7）重复上面 5~6 直到本次 DMA 传送完成
8）返回 2，等待下一次 DMA 传送请求
9）当 DMA 传输完成后，DMAC 会通过中断的方式通知 CPU 传输完成，然后 CPU 通过询问 DMA 设备或检查内存决定整个操作是否成功完成

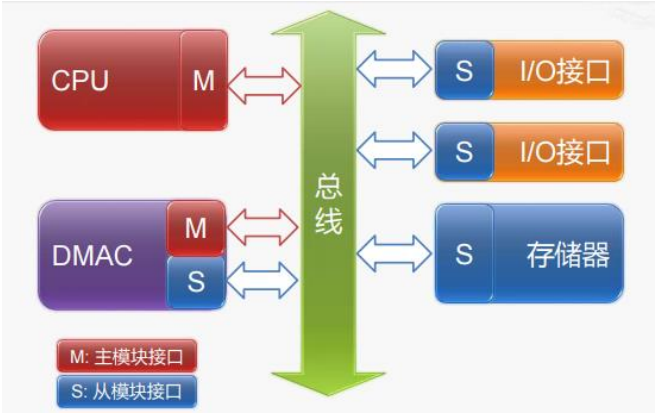
综合上面的过程大致由 3 各个抽象的过程描述：
CPU 配置 DMAC 内部配置寄存器——>DMAC 单独完成数据传输工作——>DMAC 完成工作后通过中断通知 CPU



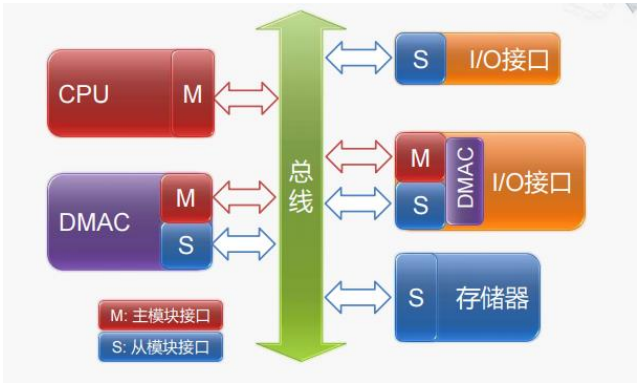
注意：Master 指的是该设备可以在系统总线上主动发起传输；Slave 指的是在系统总线上只能被动地接受传输

在上面的 DMA 传送任务的一开始，需要 CPU 配置 DMAC 中的内部配置寄存器，处理器主要提供参数：设备标识、设备要执行的操作、内存中传输数据的原或目的地址、传输的字节数





在最先的DMA控制方式中，在计算机上会以独立的DMA芯片的形式出现，但随着计算机的发展，有一些I/O接口的速度越来越快，对DMA的传输要求也越来越高，那么多个I/O接口共享一个独立的DMAC的方式无法再满足部分I/O接口的需求。这时就出现了自带DMAC的I/O接口。这样的I/O接口内部有一个专属的DMAC，只为这个I/O接口服务，比如显卡、网卡。



其他的注意点：

• 一般的DMA不仅提供I/O接口与内存之间的传输任务，还提供内存到内存的数据传送功能

• 并不是每一种I/O接口都需要DMA控制方式来实现输入输出数据的工作。通常情况下，DMA控制方式比较适合高速、数据量大的输入输出设备，而对于慢速设备，比如键盘适合使用中断或轮询的方式来完成控制工作

由DMA引发的cache一致性问题

在输入输出方式不采用DMA控制方式时，所有的对内存的访问都需通过地址转换或者cache的路径来访问内存，在以前的学习过程中都提出了一些方式来解决相关的cache一致性问题的方法。但是采用了DMA控制方式来控制输入输出工作，使得有了另外一条路径——I/O接口直接到内存的路径去访问内存。这种控制方式就会引发cache一致性问题。

因为在虚拟内存系统中，使用DMA控制方式不仅需要解决内存中的页有虚地址和物理地址的问题（一种方式就是在DMA部件当中保存少量的虚地址到实地址的映射，另一种方式就是由操作系统来管理），同时由于cache的存在使得内存中的一个数据项可能在cache中具有副本，使得当使用DMA方式改写内存中的相应数据项的时候并没有更新cache中的副本，或者通过DMA控制方式从内存中读出的一个数据项准备输出到外设时，却发现这个数据项并不是最新的值，

因为最新的值有可能还处在 cache 当中却没有回写到内存当中

DMA 控制方式引发的 cache 一致性问题通常有如下三种方式来解决：

1）将 cache 与 I/O 操作相关联，确保读操作（读内存）能看到最新值而写操作（写内存）会更新 cache 中的数据项

2）让操作系统选择性地将 cache 中与 I/O 读操作（对应写内存）相关地数据项设置为无效；对于写操作（对应读内存）强迫 cache 写回内存

3）提供一种硬件机制，选择性地刷新 cache 中的一些块（或使之无效）

总线

一、总线的概念

1、总线的定义：连接两个以上部件或设备的信息通路

数据在总线上通常以串行或者并行的方式传输。总线长度和总线连接的设备数量时影响总线速度的主要受限因素

2、总线的设计要素：

①总线带宽：单位时间内在总线上传输的最大数据量
②信号线类型：专用/复用
③仲裁方法：集中式/分布式
④定时方式：同步通信/异步通信
⑤事务类型：总线所支持的各种数据传输类型和其他总线操作类型，比如存储器读、存储器写、I/O读、I/O写、读指令、中断响应等

3、总线性能参数：

◆ 总线频率：反映总线工作的速率（f），通常单位是MHz；
◆ 总线宽度：数据总线的位数（w），单位是b（位），是微型计算机的一个重要指标，通常与处理器的字长相一致；
◆ 总线传输速率：总线上可传输的数据总量（BW），单位是MB/s：

总线传输速率=（总线宽度÷8位）×总线频率

BW =（w÷8）× f

4、总线的分类：

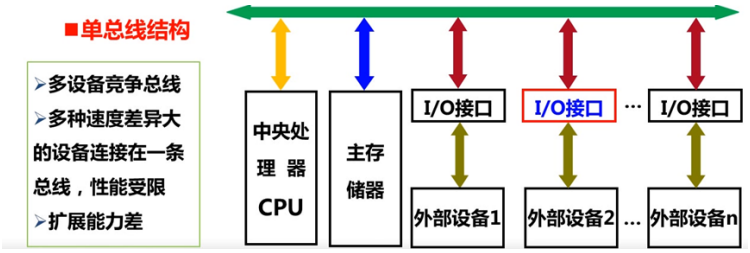
1）*芯片内总线*：在芯片内部各部件之间提供连接

2）*系统总线*：在计算机系统主要部件（处理器、存储器和各种 I/O 设备控制器）间提供连接

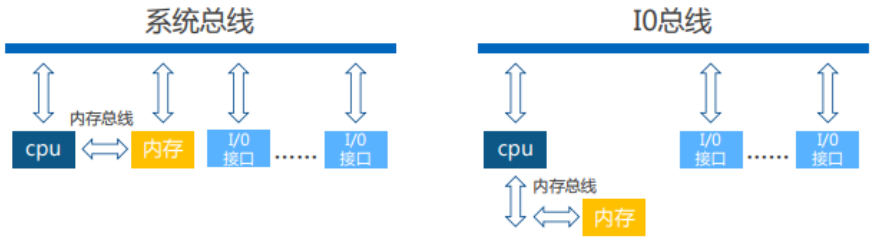
3）*通信总线*：在主机和 I/O 设备之间或计算机系统之间提供连接

5、系统总线连接方式：

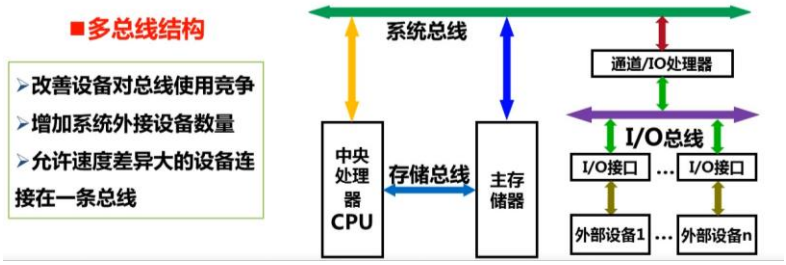
•单总线结构：将 CPU、主存和各种 I/O 适配卡通过底板总线连接，底板总线为标准总线



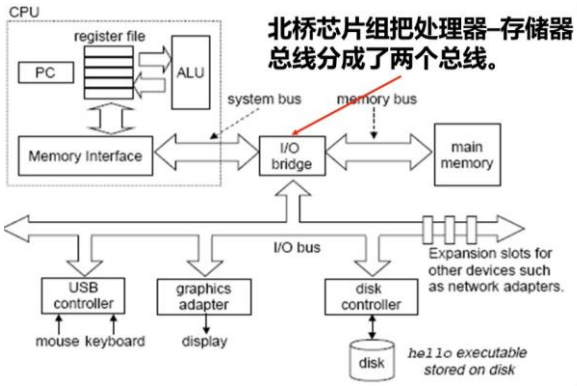
•双总线结构：在单总线结构的基础上，使用处理器-存储器总线将处理器和内存相连，减轻系统总线的负担，提高并行性，不过需要额外增加 I/O 指令（这也是单总线结构和其他总线结构一个很大的区别，也是对指令系统的一个影响），但总体而言仍然保持了单总线结构简单的特性也有不少的缺点

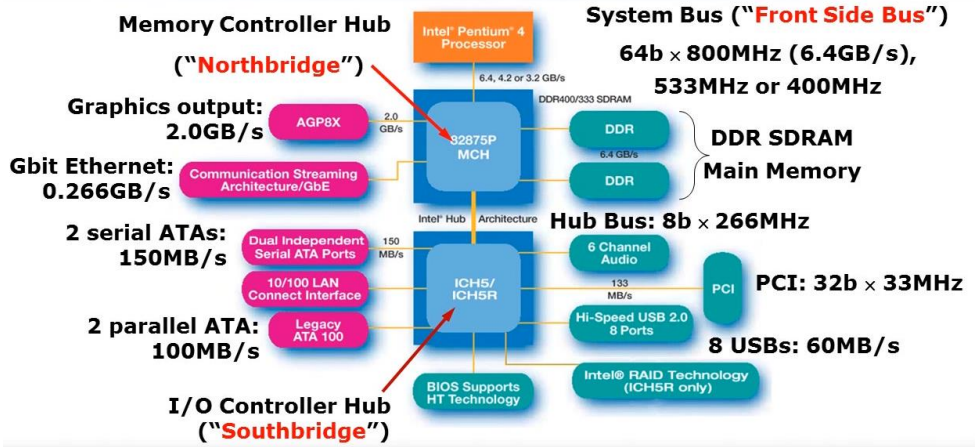


• 多总线结构：将 CPU、cache、主存和各种 I/O 适配卡通过局部总线、处理器-存储器总线、高速 I/O 总线、拓展 I/O 总线等互联。值得注意的是 I/O 接口连接的是 I/O 总线，而 I/O 总线通过一个 I/O 通道与系统总线相连，这个 I/O 通道就是总线桥（连接不同速率之间的器件，起到信号速度缓冲、电平转换、控制协议转换等作用）



在实际中，以 Intel 早期奔腾的结构，北桥芯片组将处理器-存储器总线分成两个总线，即前端总线（Front Side Bus，Intel 定义其为系统总线）和存储总线（Memory Bus）





在上面这张图中，CPU 通过前端总线与北桥芯片组（内存控制器集线器，实质上就是一个 DMAC）相连，而北桥芯片组连接到显卡、内存和其他高速设备，并通过 Hub 总线与南

桥芯片组（I/O 控制器集线器）相连，从南桥拓展出一些低速的 I/O 总线。随着时代的发展北桥已经集成到 CPU 内部，整块主板上只留下了单桥 PCH 连接一些 I/O 设备，甚至在一些嵌入式领域，南北桥都被继承到了 CPU 内部即 SoC

6、系统总线的构成

系统总线由数据总线、地址总线和控制总线构成

• 控制总线：控制对数据线和地址线的访问和使用，用来传输时序信号和控制命令信号。典型的控制信号包含如下：

- **控制线：一些典型的控制信号**
  - **时钟：用于总线同步**
  - **复位：初始化所有设备**
  - **总线请求：表明发出该请求信号的设备要使用总线**
  - **总线允许：表明接收到该允许信号的设备可以使用总线**
  - **中断请求：表明某个中断正在请求**
  - **中断回答：表明某个中断请求已被接受**
  - **存储器读：从指定主存单元中读数据到数据总线上**
  - **存储器写：数据总线上的数据写到指定主存单元中**
  - **I/O读：从指定的I/O端口中读数据到数据总线上**
  - **I/O写：将数据总线上的数据写到指定的I/O端口中**

• 数据总线：承载在源和目的部件之间传输的信息。数据线的宽度反映了一次能传送的数据的位数

• 地址总线：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，地址线的宽度反映了最大的寻址空间

注意：有一些总线可能没有单独的地址线，地址信息通过数据总线来传送，称之为数据/地址复用，而控制总线决定当前数据总线传送的是数据还是地址。比如分时复用

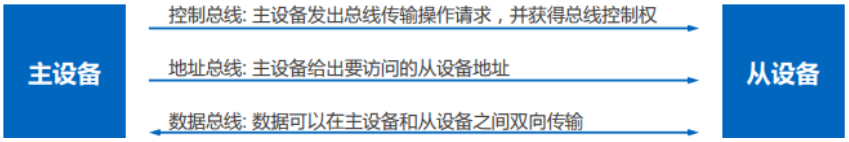
二、总线设备和总线控制器

1、总线设备的分类：

按照设备对总线的使用权可以分成

• 总线主设备：能够申请并获得总线使用权的设备，具有控制总线的能力，发起总线事务，比如 CPU 和 DMA 控制器

• 总线从设备：不具有申请总线使用权的设备，是被总线事务激活的模块或设备



按照设备的访问方式可以分成

• 存储器设备：使用访问存储器的方法（访存型总线指令）访问的设备

• I/O 设备：使用访问 I/O 的方法（I/O 型总线指令）访问的设备

按照设备传送数据的方向可以分为总线源设备和总线目标设备

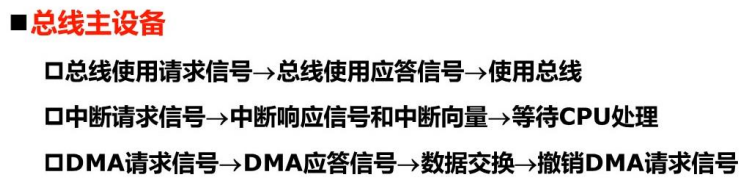
什么是总线事务？

从请求总线到完成总线使用的操作序列，它是在一个总线周期中发生过的一系列活动。典型的总线事务包括：请求操作、裁决操作、地址传输、数据传输和总线释放

2、总线设备接口：

总线设备接口是连接设备与总线的桥梁，完成设备信号到总线信号的协调和转换

对于总线主设备，其为总线提供的工作：



• 若使用的是请求应答方式工作：

在使用总线之气那会先通过总线设备接口发出总线请求信号——>在接收到应答信号后——>即可开始使用总线

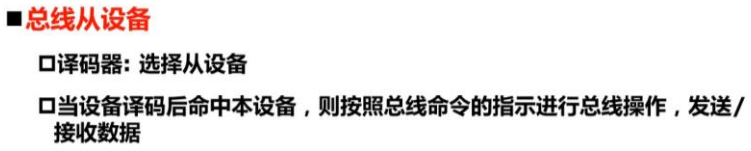
• 若使用中断方式工作：

在最开始会先发出中断请求信号，在接收到中断响应信号和中断向量之后——>等待 CPU 处理

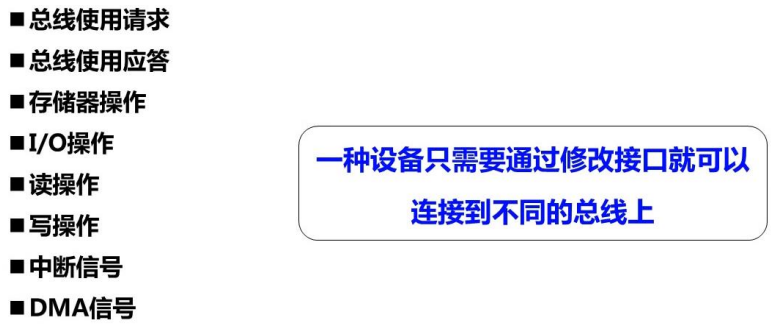
• 若使用 DMA 方式工作：

会先发出 DMA 请求信号，在接收到应答信号后，即可进行数据交换——>当完成处理后，会撤销 DMA 请求信号

对与总线从设备，其为总线提供的工作：



总线设备接口对设备具有的处理能力：



3、总线设备控制器：

总线设备控制器是总线系统的核心，其任务是管理总线的使用，实现总线协议。其具体功能如下：

■总线系统资源的管理

□对存储空间、设备端口空间、通道、中断等进行分配、启动等操作

■总线系统的定时：产生总线时序和总线命令

■总线的仲裁：确定哪个主设备获总线使用权

■总线的连接

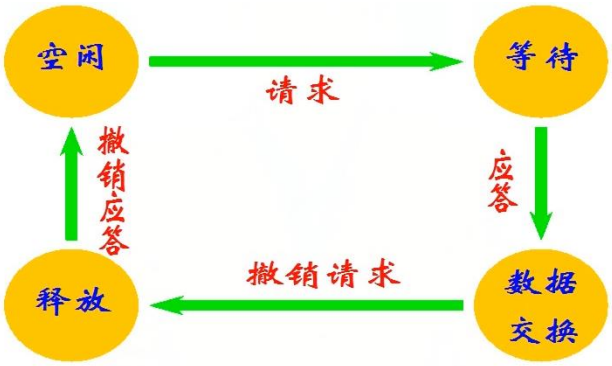
□不同总线协议之间的转换

□完成总线之间的连接

总线传输操作过程：

在主设备控制下通过总线进行信息传送操作，一般会经历如下 4 个阶段：申请与仲裁阶段——>寻址阶段——>数据传输阶段——>结束阶段

或者说总线设备具有 4 个状态：空闲状态、等待状态、数据交换、释放；那么一次对总线的使用过程就是在这 4 个状态之间进行转换，而总线控制器即是控制总线设备如何在这 4 个状态之间转换



常见的总线操作包括：

• 写操作

• 读操作

•读修改写操作：即从总线设备上读入数据并在主设备进行修改后然后写入到总线从设备中

•写后读操作：即总线主设备将数据写入到从设备后读取从设备中的数据（这上面两种操作可以省去在一次读或者写之后申请总线使用权的步骤）

•块操作：上面的所有总线操作都是读/写一个总线位宽（很多时候就是一个机器字长）的数据，而块操作可以连续读/写多个字的数据

三、总线仲裁

1、总线仲裁的定义：

由于总线被连接在其上的所有设备共享。如果没有任何控制，那么当多个设备需要进行通信的时候，每一个设备都试图为各自的传输将信号发送到中线上，这样就会产生总线混乱。所以需要进行总线仲裁选择一个设备有权使用总线

2、如何避免总线混乱？

1）在总线中引入一个/多个总线主设备，使得只有总线主设备能够控制总线

• 主设备：能够发起总线控制请求并控制总线的设备。如 DMA
• 从设备：只能响应从主设备发来的总线命令的设备。如内存

2）使用总线仲裁决定哪一个总线主设备将获得下次的总线使用权

3、总线仲裁的分类：

• 集中式：将控制逻辑设置在一个专门的总线控制器或仲裁器中，通过将所有的总线请求集中在一个特定的仲裁算法进行仲裁

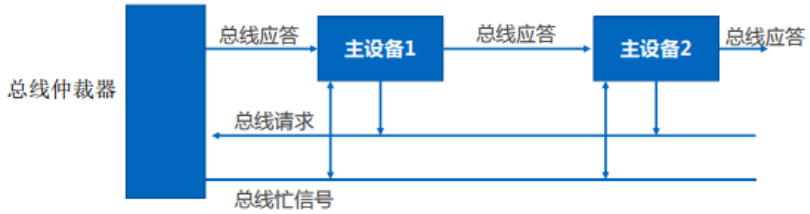
•分布式：没有专门的总线控制器，其控制逻辑分散在各个部件或设备中（特指总线设备接口）

在进行总线仲裁时，必须考虑到以下两个因素并进行平衡：

• 等级性——具有高优先级的设备应该被先服务
• 公平性——即使具有最低优先级的设备也不能永远得不到总线使用权

4、集中式总线仲裁方式：

1）菊花链式查询：

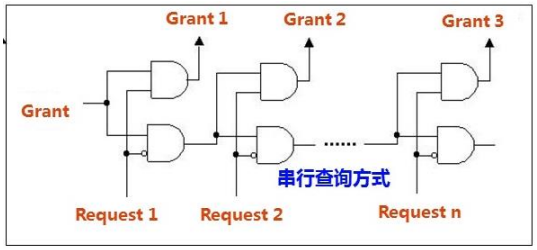


a）当总线上的设备需要使用总线时，经过总线请求线 BR 发送总线请求信号，若总线空闲时，则立即发送总线响应（授权）信号 BG

b）总线授权信号 BG 从高优先级的设备一次向低优先级的设备传递，若到达设备有总线请求，则总线授权信号 BG 不在往下传递

c）该设备主动建立总线忙 Busy 信号，防止其他总线设备占用总线

其内部串行查询电路很简单，如图所示：



菊花链式查询方式的优缺点：

□ 优点

① 简单，只需几根线就能按一定优先次序实现总线裁决
② 易扩充设备(flexible)

□ 缺点

① 不能保证公平性
② 对电路故障很敏感
③ 菊花链仲裁限制了总线性能

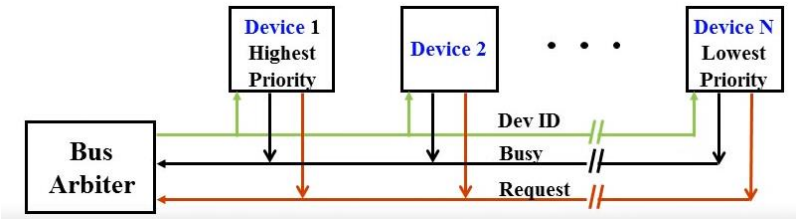
2）计数器定时查询：

a）当总线上的设备需要使用总线时，经过总线请求线 BR 发送总线请求信号，若总

线空闲时，则计数器开始计数，计数值通过设备地址线 DevID 发送给各个设备

b）当设备地址线 DevID 上的计数值与请求总线设备的地址一致时，该设备获得总线控制权，同时中止计数器的计数和查询

c）主设备主动建立总线忙 Busy 信号，防止其他设备占用总线



计数器定时查询方式的优点：

• 灵活。设备优先级可以通过设置不同的计数初始值来改变。若每一个初始值都是0，则是固定优先级方式，这样就跟菊花链式查询法没有什么区别；若每一次初值总是刚获得总线使用权的设备的 ID 号（即从上一次的中止点开始），那么是平等的循环优先级方式

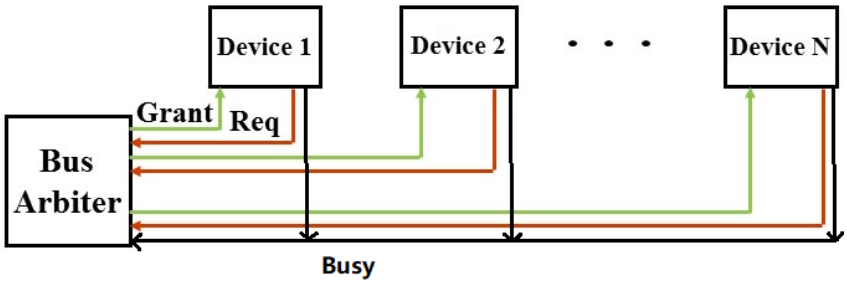
• 对电路故障不如菊花链式查询那样敏感

缺点：

• 需要增加一组设备线

• 总线设备控制逻辑复杂（需对设备号进行译码比较等）

3）独立请求：



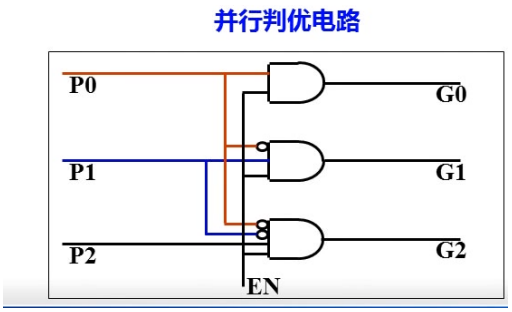
使用独立请求方式的总线仲裁控制逻辑设计时，各设备都有一对总线请求线 BRi 和总线允许线 BGi

a）当总线上的设备需要使用总线时，经过各自的总线请求线 BRi 发送总线请求信号

b）总线仲裁器中有一个判优电路，可根据各设备优先级确定选择哪个设备获得总线使用权。仲裁器可采用固定的优先级，也可编程设置优先级

c）获得总线控制权的设备主动建立总线忙 Busy 信号，防止其他设备占用总线

其中总线仲裁器内部的判优电路也很简单，如图所示：



5、分布式总线仲裁方式：

在分布式总线仲裁方式中，每一个设备都有自己的仲裁电路（分布在总线设备的接口上面），这个仲裁电路一方面要发出总线请求信号，同时还要监听其他设备的总线请求信号，然后将自己的设备的优先级与其他总线设备的优先级进行比较仲裁，在仲裁之后自己决定是否应该控制总线的使用权