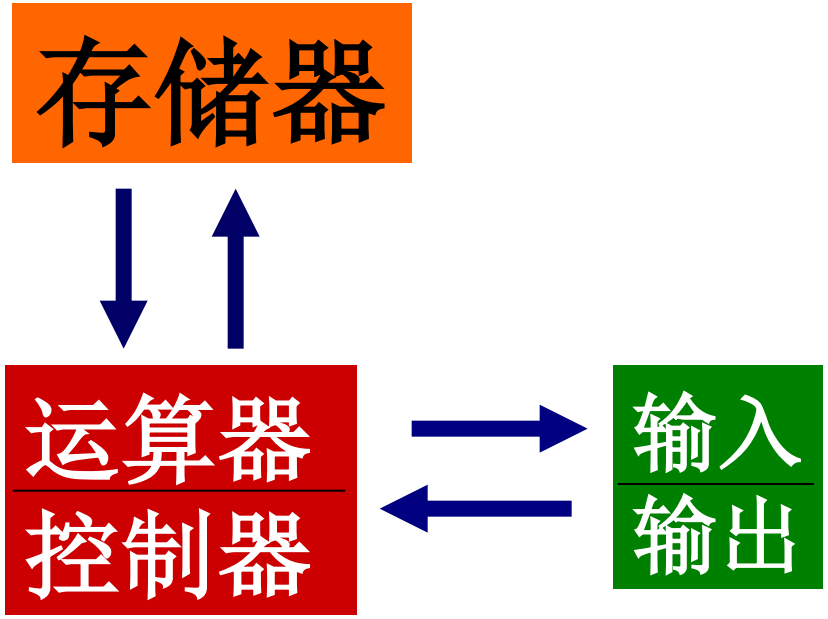
第7章

输入/输出系统

现代计算机体系结构——冯·诺依曼体系结构



7.1 输入/输出系统概述

一. 概 述

输入/输出系统是微型计算机系统中的主机与外部设备进行数据交互的系统。主机通过输入/输出接口与外部设备连接，在*接口电路*和*驱动程序*控制下进行信息交换。

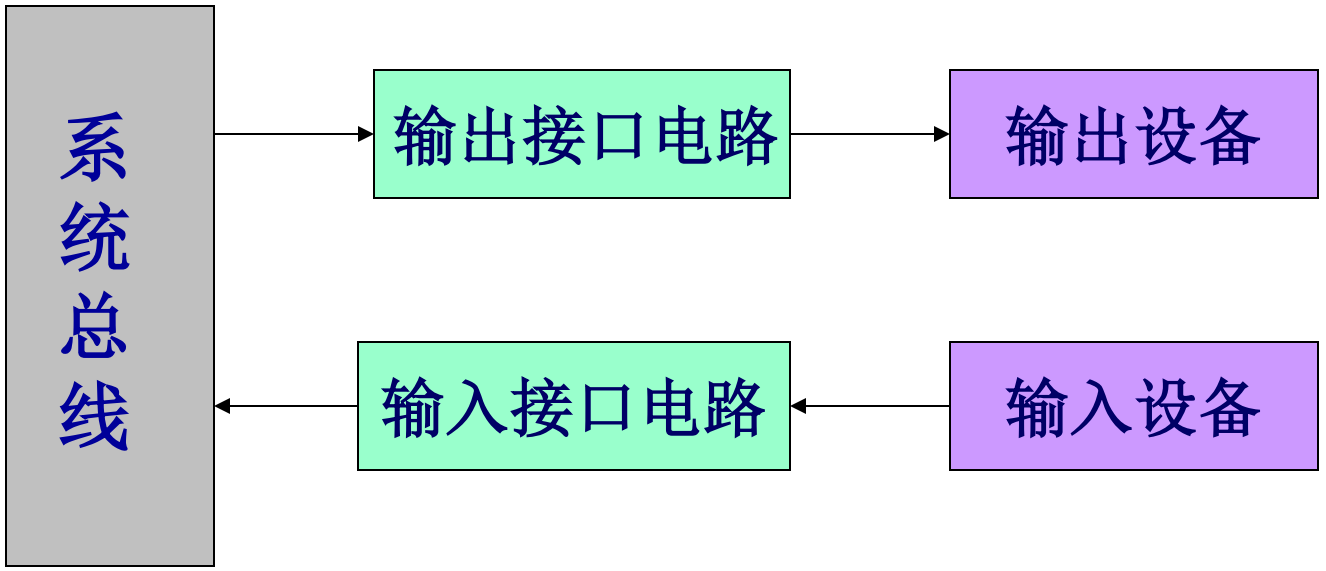
1. 接口电路的作用

CPU数据 → 输出接口电路 → 输出设备

CPU ← 输入接口电路 ← 输入设备数据

接口：是CPU与外部设备交换信息的中转站

I/O设备与总线之间的连接



I/O接口电路所处位置

2. 接口电路的功能

- 应具有数据暂存/缓冲功能
- 与外设之间有联络功能，提供外设状态
- 应有端口地址译码器，具有寻址功能

  (便于使用IN, OUT指令读写数据)
- 有数据转换功能(并 → 串, 串 → 并)
- 有中断管理能力

3. 关于端口的概念

“端口”是接口电路中，能与CPU交换信息(使用IN，OUT) 的寄存器。

端口分类:

数据口: 存放CPU向外设输出或外设输入的数据。

控制口: 存放控制信息--控制接口电路、外设的工作。

状态口: 存放状态信息 — 反映外设的状态。

注意：接口电路必须具有数据口。

注：每个端口，系统都为它编了一个地址，系统只要给出某个地址，通过译码电路，就能找到相应的I/O接口电路中的端口寄存器。

问　题：系统给出的地址是内存单元地址还是I/O端口寄存器的地址？

解决方案：合理安排I/O端口寄存器的编址方式。

4. 端口的编址方式

- 存储器映像方式

  把端口和存储单元等同看待，统一编址。

  特点: 凡访问存储单元的指令都可访问

  I/O端口, 端口地址占用存储空间。

- I/O端口独立编址

  特点: I/O 端口不占用存储空间，

  CPU要有专用的 I/O 指令。

5. PC系列机的端口编址

- PC系列机采用端口独立编址；
- 从8088～奔腾微处理器，设计时用A15～A0
  低16位地址寻址 I/O 端口；
  所以, CPU的端口寻址能力为2^{16}=65536个；
- 基于微处理器的PC系列，实际使用CPU中A9～A0
  做I/O地址线；
  所以，PC系列机 I/O 端口地址最多为
  2^{10}=1024个。
- 这1024个口地址，系统本身(主板上, 以及常规 I/O接口)已经占用了一部分。
- 端口地址(I/O 空间)没有分段的概念。

二. 最常用的I/O指令

1.直接寻址 I/O 指令

设N为8位端口地址

```
IN      AL, N       ;口地址为N的端口中取数→AL
OUT     N, AL       ;AL内容→口地址为N的端口寄存器
IN      AX, N       ;[N]→AL, [N+1]→AH
OUT     N, AX       ;AL→N口, AH→N+1口
```

如：
```
IN      AL, 61H
OUT     61H, AL
```

2. DX间址的I/O指令

当口地址 N > 8位二进制数时，用DX间址

```
IN      AL, DX   ; [DX]的端口内容 → AL
OUT     DX, AL   ; AL →[DX]的端口寄存器
IN      AX, DX   ; [DX] → AL, [DX+1] → AH
OUT     DX, AX   ; AL → [DX], AH → [DX+1]
                   的端口寄存器
```

如：
```
MOV   DX, 3F8H
IN    AL, DX   ;从3F8H端口取数 → AL
```

注意：I/O 指令只能在端口和AL, AX, EAX之间交换信息，用DX间址，但不能使用方括号，即不能写成：IN AL, [DX]。

单选题 1分

设置

下列指令中，格式合法的是(　　)。

A　OUT 3F8H, AL

B　IN AL, 60H

C　IN AL, [DX]

D　IN BL, DX

提交

填空题：

1. 端口是 I/O接口电路中能与CPU交换信息的寄存器 。

2. 按端口存放信息的物理意义来分，端口可以分为 数据 端口、 状态 端口和 控制 端口。

3. 按传输信号的性质来分，微机系统中的总线可以分为三类，分别是 数据 总线、 控制 总线和 地址 总线。

7.2 微机系统与外设交换信息的方式

微机系统与 I/O 端口的信息交换有四种方式:

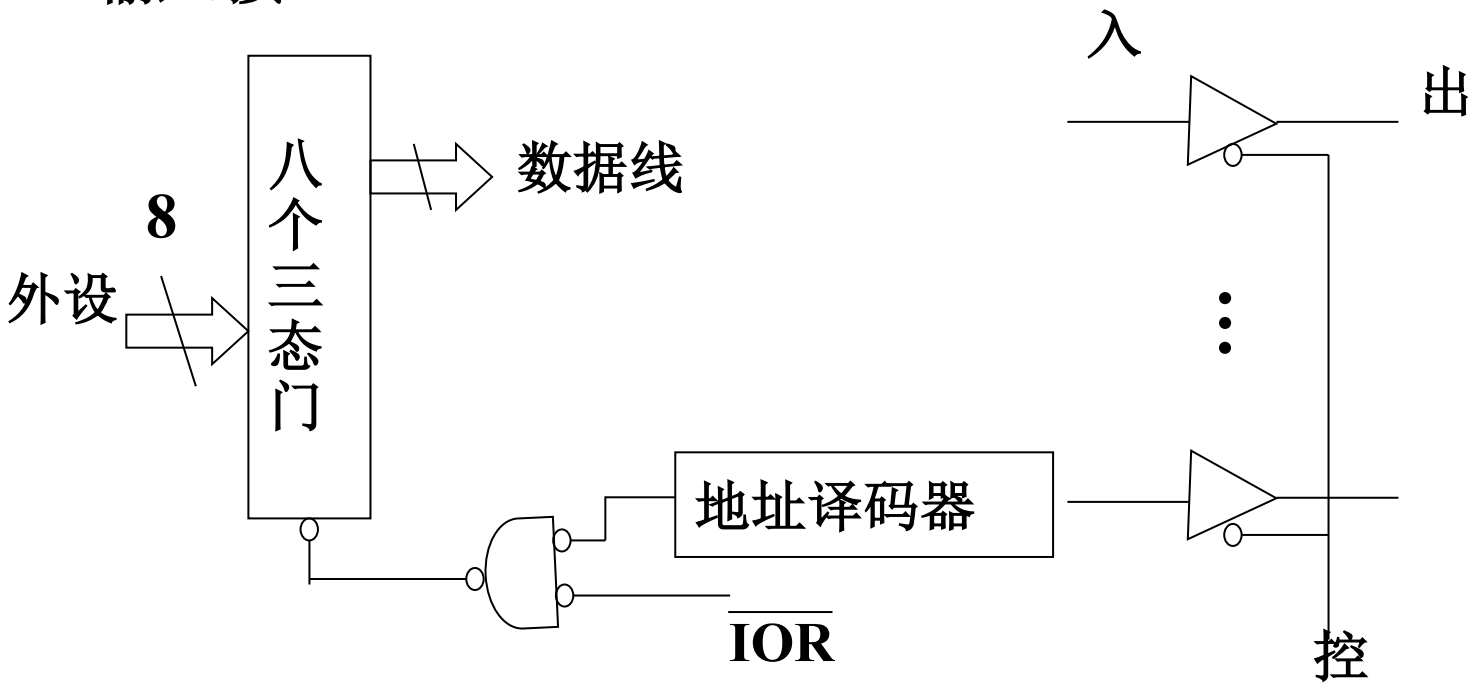
无条件传送　　查询方式

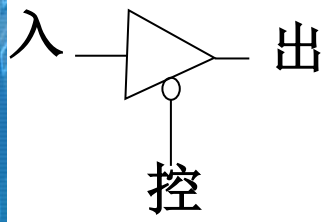
中断方式　　DMA方式

采用何种方式与接口的硬件电路有直接关系

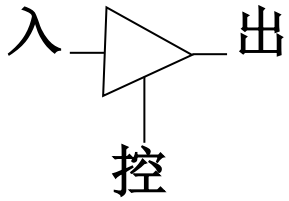
1. 无条件传送

- 输入接口：





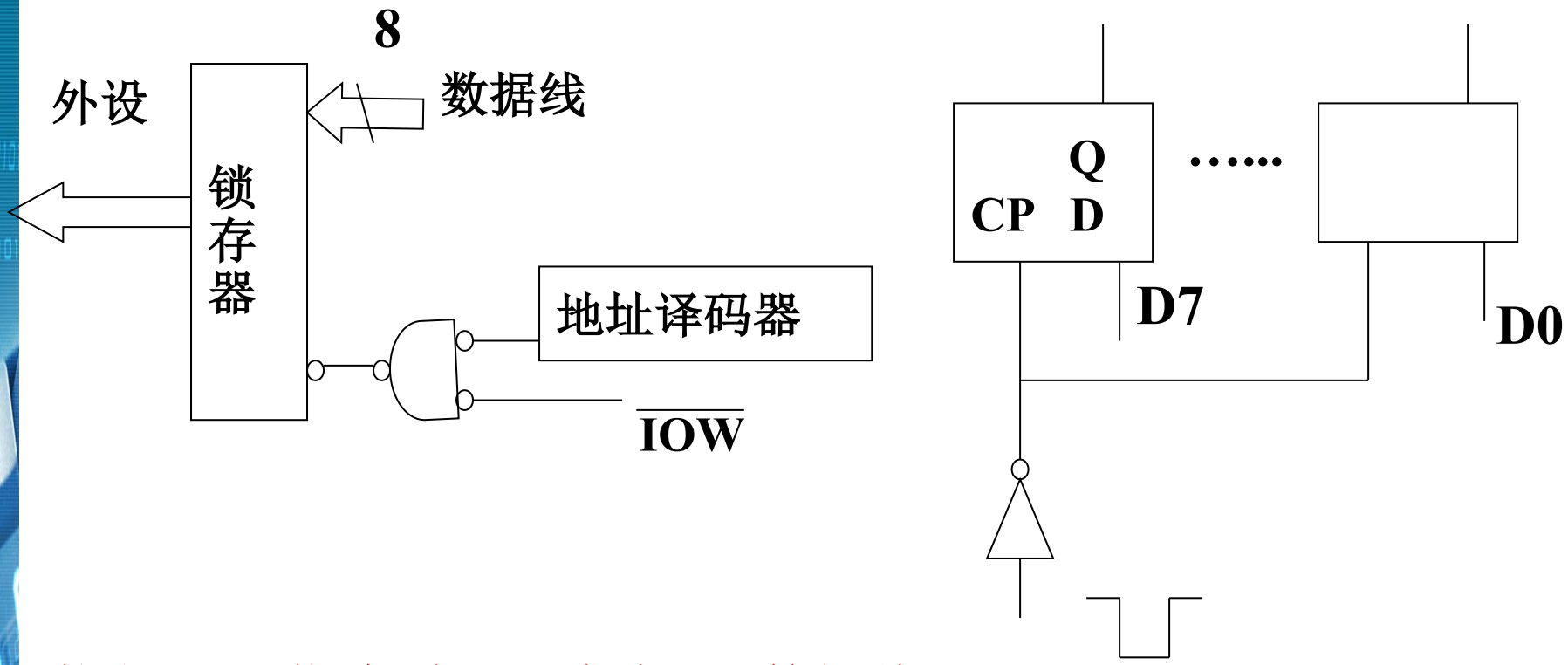
控=0，出=入
控=1，出为高阻



控=1，出=入
控=0，出为高阻

执行IN指令之前，外设数据已经准备好。

输出接口



执行OUT指令时: AL内容 → 数据线，
口地址 → 地址线上，
$\overline{IOW}$=低，把数据锁存到锁存器中。

无条件传送运用方法例子

假设201H是某个外设接口电路的数据口

无条件传送方式进行数据读取时，可以使用指令：

```
MOV DX, 201H
IN AL, DX
```

完成将201H端口中的8位数据读取到AL寄存器中。

无条件传送方式进行数据输出时，使用指令：

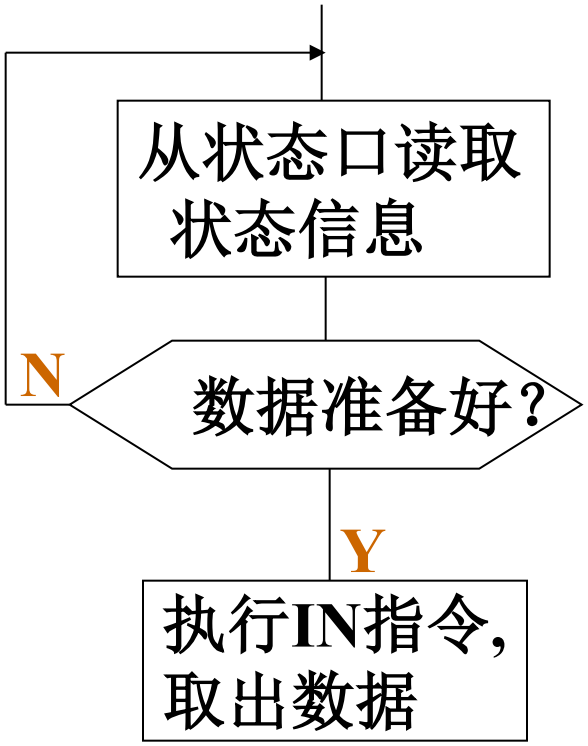
```
MOV DX, 201H
MOV AL, 7FH
OUT DX, AL
```

完成将AL寄存器中8位数据输出/写入到将201H端口中。

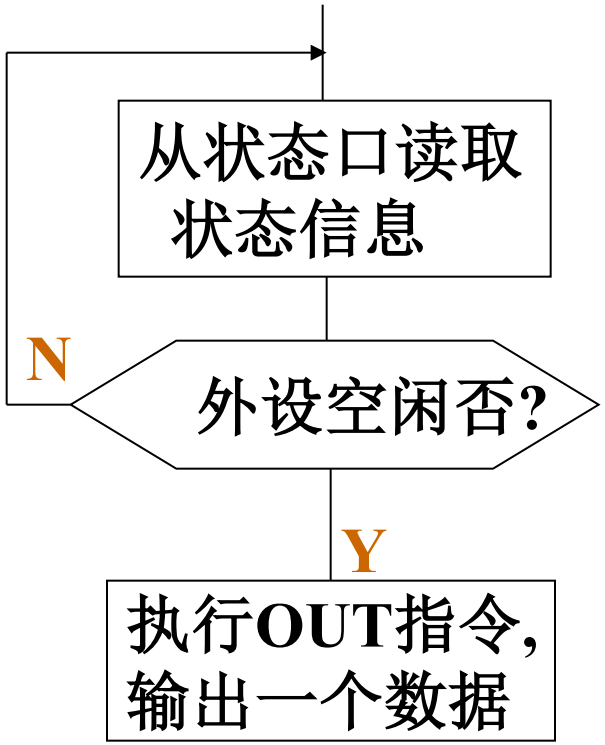
2. 查询方式(*重点！*)

CPU用查询方式与外设交换信息，必先了解外设的状态。

查询方式输入流程



查询方式输出流程



- 查询方式输入

设状态口地址=200H，使用状态端口的最高位作为标志位(为1表示外设数据准备好，可以读取数据)，数据口地址=201H

查询式外设数据输入核心程序如下：

```
RSCAN : MOV    DX , 200H
        IN     AL , DX
        TEST   AL , 80H
        JZ     RSCAN
        MOV    DX , 201H
        IN     AL , DX
```

- 查询式输出

设状态口地址=200H=数据口地址，使用状态端口的最低位作为标志位(为0表示外设空闲，可以写入数据)

查询式数据输出核心程序如下：

```
TSCAN:  MOV    DX,  200H
        IN     AL,  DX
        TEST   AL,  1
        JNZ    TSCAN
        MOV    DX,  200H
        MOV    AL,  某数
        OUT    DX,  AL
```

例：

设状态口地址为200H，数据口地址为201H，使用状态端口的最低位作为标志位(为0表示外设数据准备好)，请补全下面查询式输入程序。

```
RSCAN: MOV DX, 200H
       IN AL, DX
       TEST AL, 1
       JNZ RSCAN
       MOV DX, 201H
       IN AL, DX
```

3. 中断控制方式

- 在有多个外设的系统中，多个外设要求CPU为它服务是随机的
- 若采用查询方式工作，就不能保证系统实时地对外设的请求作出响应
- 为了提高CPU的效率，使系统有更好的实时性能，导致了中断处理技术的产生

特点：

- 在外设没有作好数据传送准备时，CPU可执行与传送数据无关的其它指令
- 当外设作好传送准备后，主动向CPU请求中断
- 若CPU响应这一请求，则暂停正在运行的程序，转入中断服务程序，完成数据传送
- 待服务完毕后，自动返回原来运行的程序

4. 直接存储器存取(DMA)方式

为什么要有DMA？

中断传送是由CPU通过程序来实现的，执行中断服务程序需要保护断点，中断结束还要恢复现场等，对于高速外设，中断的效率也不高。

什么是DMA？

用硬件实现在外设与内存间直接进行数据交换，而不通过CPU

特点：

- 数据传送速度的上限就取决于存储器的工作速度
- 速度快

7.3 DMA控制器

1. DMA (Direct Memory Access) : 直接存储器存取, 习惯上称DMA传送。

DMA传送: 利用硬件完成高速外设与系统RAM之间的信息交换。

2. DMAC: DMA 控制器。

它是实现DMA传送的核心芯片。

3. 专用术语:

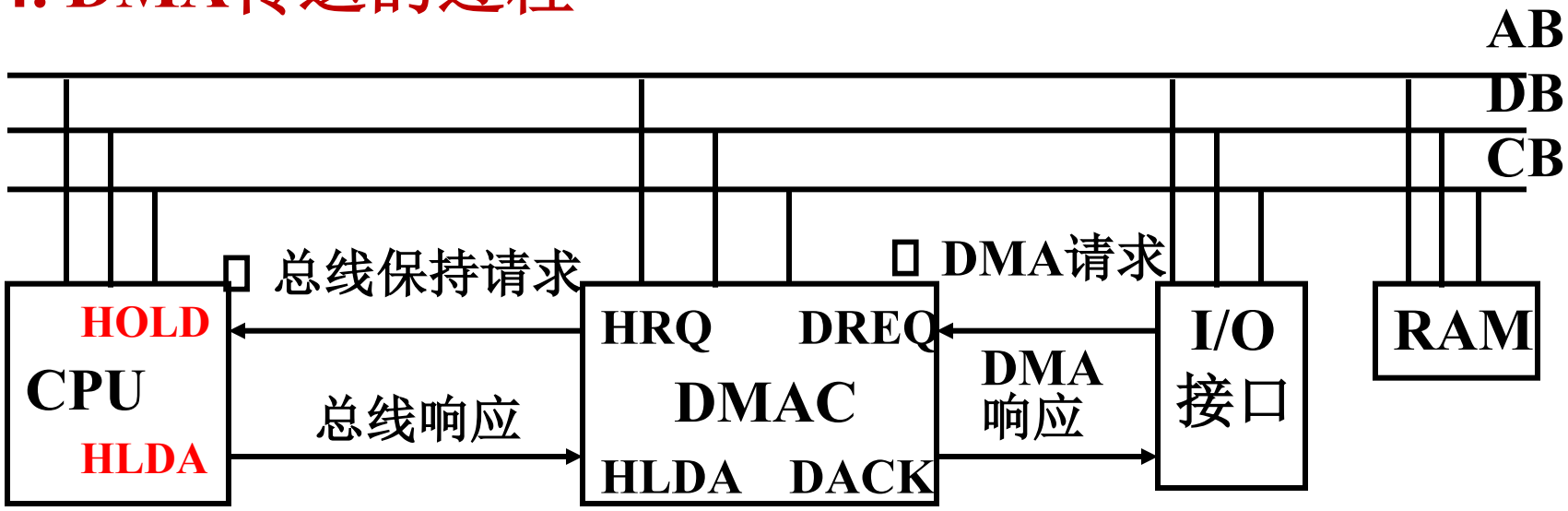
- DMA读传送: 在DMAC控制下,

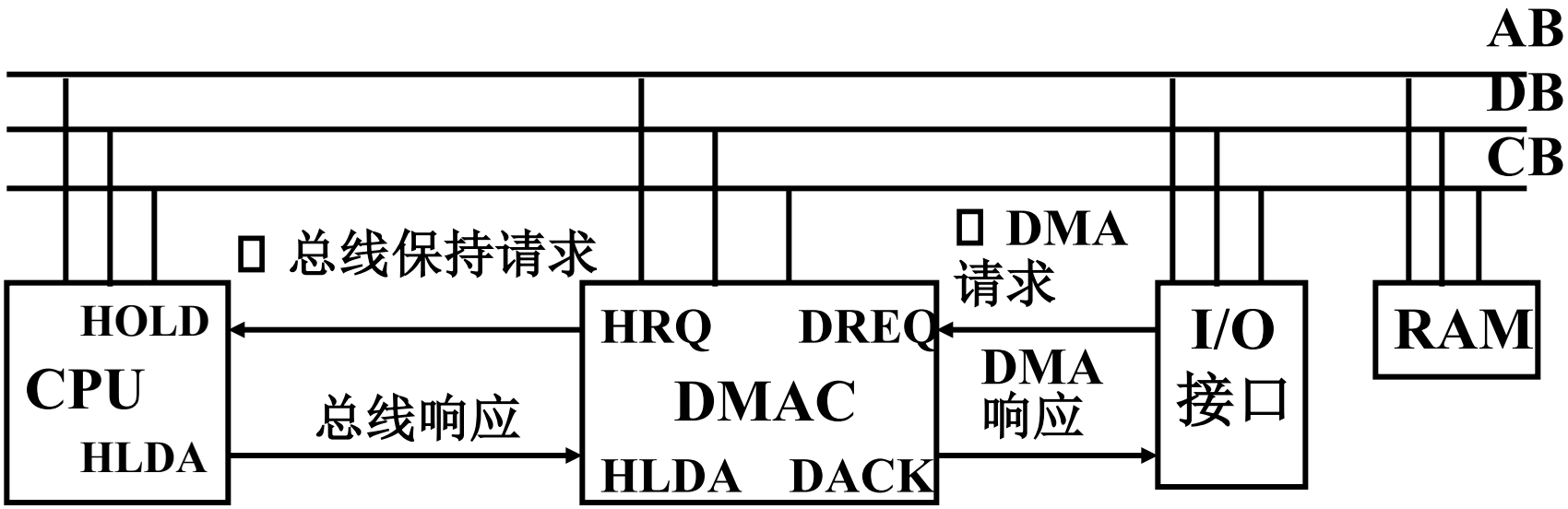
  读取RAM的内容 → I/O端口。

- DMA写传送: I/O端口信息 → 系统RAM某单元。
- 存储单元读 / 写传送: 在DMAC控制下, 实现系统 RAM ↔ RAM。

注意:在PC系列机中禁止RAM ↔ RAM传送。

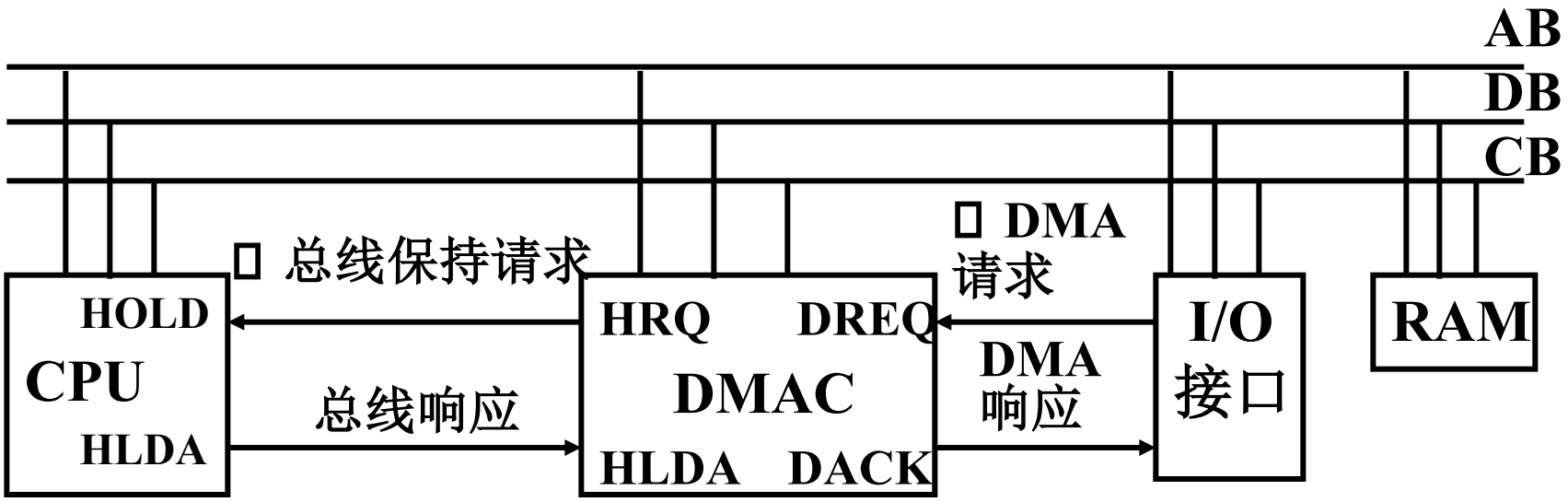
4. DMA传送的过程





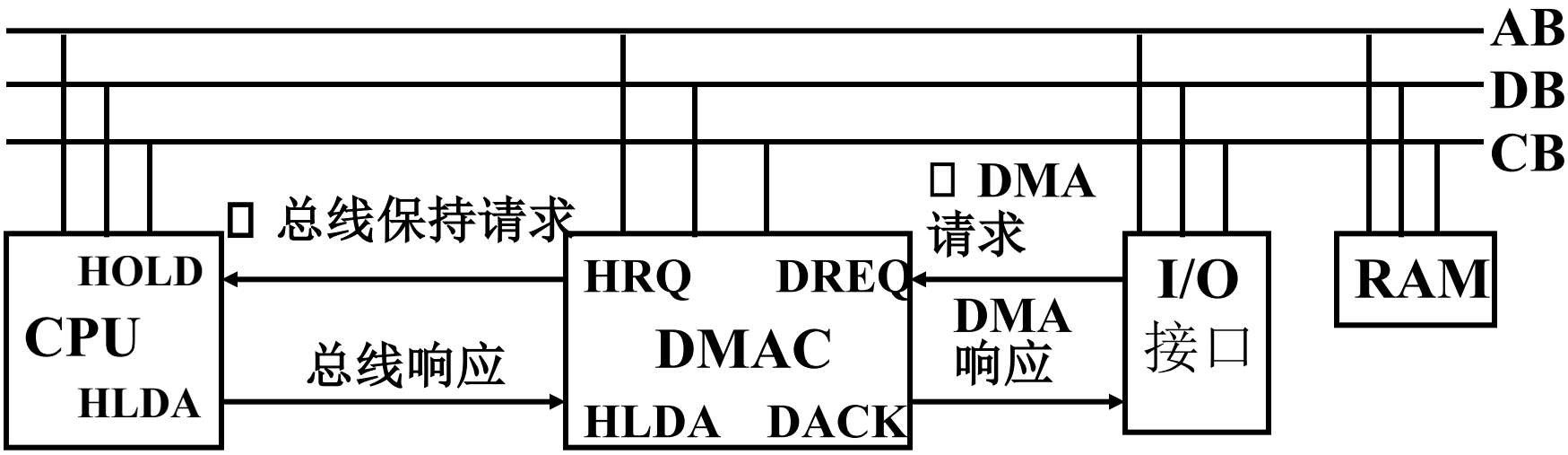
系统的三总线分别受到CPU和DMAC的控制。

CPU可以向地址总线、数据总线、控制总线上发送信息, DMAC也可以向地址总线、数据总线、控制总线上发送信息, 但同一时间, 三总线只能受一个器件的控制, 所以, 两者之间必须有联络信号:

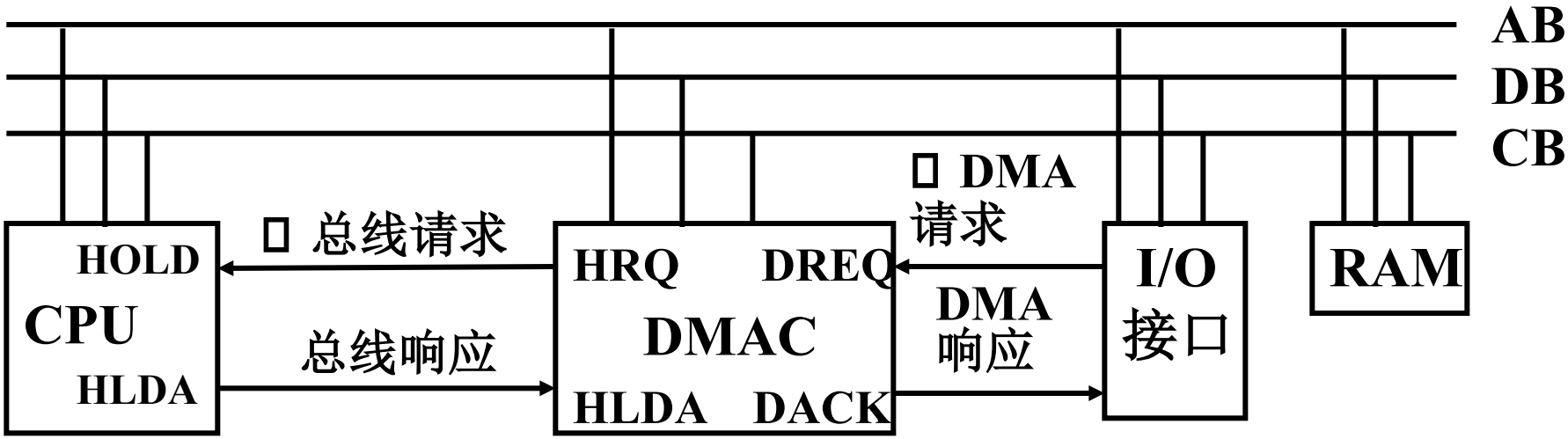


- 高速外设， 通过其接口电路向DMAC发出“DMA请求”信号 (请求DMAC为其传送数据)。

- DMAC检测到有DMA请求之后, 即向CPU提出总线保持请求 (请求CPU脱离总线)。



- CPU执行完当前指令的当前总线周期之后脱离系统总线, 并向DMAC发出“总线保持响应”信号。
- DMAC收到“总线响应”信号之后, 接管系统总线的控制权, 并向I/O接口发出DMA响应信号。



- 在这之后, 由DAMC控制系统总线, 进行DMA传送。

a. 若进行DMA读传送:

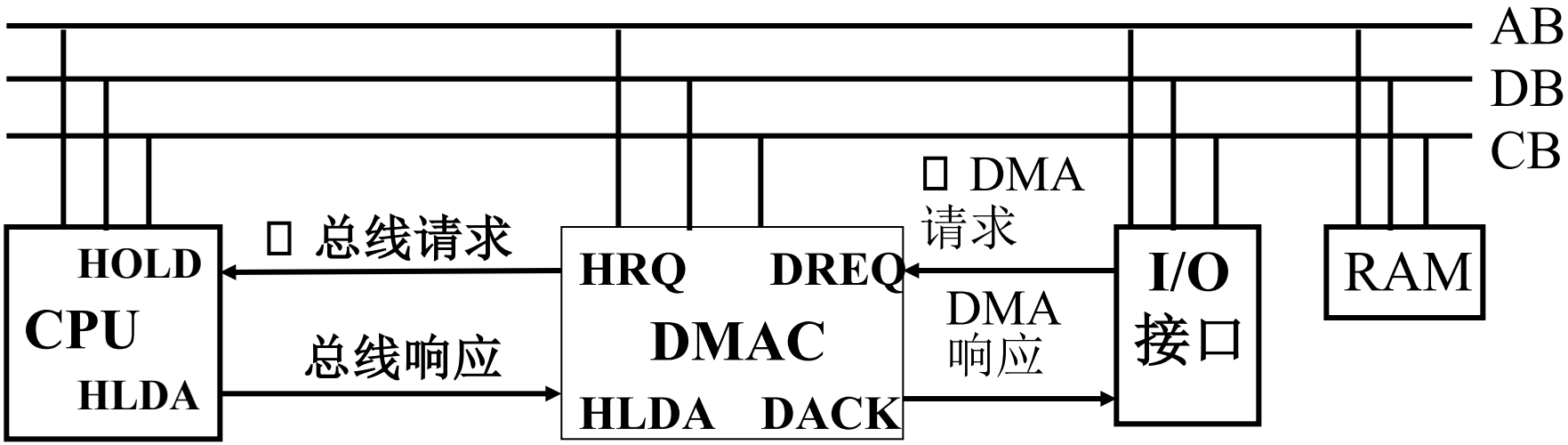
DMAC把RAM地址 → 地址总线上

DMAC发出存储器读命令和I/O写命令

b. 若进行DMA写传送

DMAC把RAM地址 → 地址总线上

DMAC发出I/O读命令和存储器写命令



- 预定的字节数全部传送完毕, DMAC脱离系统总线, CPU再次控制系统总线, 完成被中断指令的后继总线周期。

5. DMA传送与中断方式的比较

- 响应时间: CPU接到“中断请求”后要等到当前指令执行完毕才响应，而CPU接到DMAC的“总线请求”后，只要当前指令的当前总线周期执行完毕就响应!

- 数据传送速度: DMAC传送比中断传送要快!

  以一个字节数据传送为例

  中断服务程序:

  保护现场

  (I/O 端口) → CPU

  CPU → RAM

  送中断结束命令

  恢复现场

  IRET

由此看出, 中断传送是由软件完成的, 执行一次中断服务程序, 就完成一字节的 I/O 写传送。

而DMA传送是由硬件完成的, 每传送一个字节只占用CPU的一个总线周期。

- 中断请求分为内部中断和外部中断。

  DMA请求的方式也有两种:

  硬件DMA请求和软件DMA请求。