




# 第6章 输入输出接口



## 第6章 输入输出接口



### 学习重点

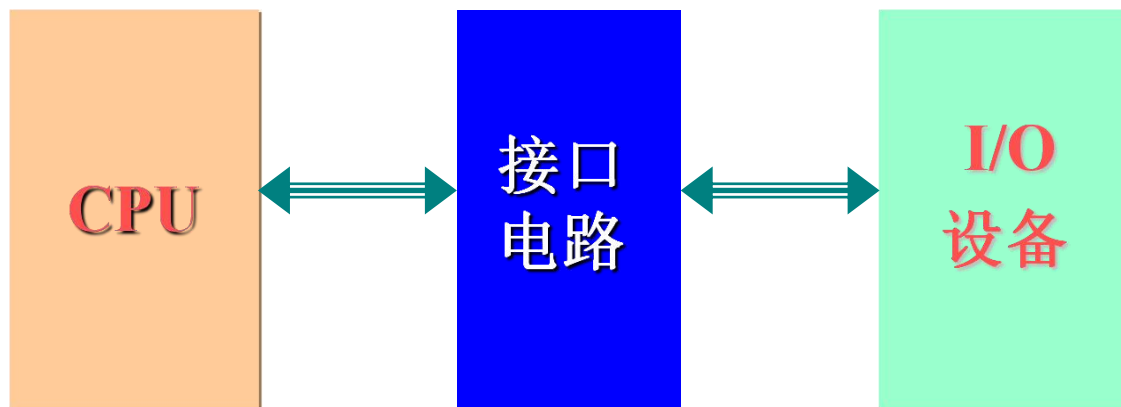
- ◇ I/O接口电路的典型结构
- ◇ 无条件传送方式
- ◇ 程序查询传送方式
- ◇ 程序中断传送方式

## 6.1 I/O接口概述

### 什么是I/O接口

- ◇ I/O接口是位于主机系统与外设之间，用来协助完成数据传送和控制任务的逻辑电路。
- ◇ PC机系统板上的可编程接口芯片、I/O总线槽中的电路板都是接口电路。

实例



## 6.1 I/O接口概述



### 为什么需要I/O接口

微机的外部设备多种多样，工作原理、驱动方式、信息格式、以及工作速度方面彼此差别很大，它们不能与**CPU**直接相连，必须经过中间电路再与系统相连，这部分电路被称为I/O接口电路。

## 6.1 I/O接口概述



### 什么是微机接口技术

微机接口技术是处理微机系统与外设间联系的技术。

- ◇ 根据应用系统的需要，构造和使用相应的接口电路，编制配套的接口程序，支持和连接有关的设备；
- ◇ 既包括硬件之间的接口，也包括软件与硬件的接口，是软硬结合的技术。

## 6.1.1 I/O接口的主要功能

(1) 对输入输出数据进行缓冲和锁存

输入接口有缓冲环节；输出接口有锁存环节

实际的电路常见：

输出锁存缓冲环节、输入锁存缓冲环节

(2) 对信号的形式和数据的格式进行变换

微机直接处理：数字量、开关量、脉冲量

(3) 对I/O端口进行寻址

(4) 与CPU和I/O设备进行联络

## 6.1.2 I/O接口的典型结构

1. 接口电路的内部结构
2. 接口电路的外部特性
3. 接口电路芯片的分类
4. 接口电路的可编程性

# 1. 接口电路的内部结构

◇ CPU与外设主要有数据、状态和控制信息需要相互交换，于是从应用角度看内部：

## (1) 数据寄存器

- ◆ 输入数据寄存器：保存外设给CPU的数据
- ◆ 输出数据寄存器：保存CPU给外设的数据

## (2) 状态寄存器

- ◆ 保存外设或接口电路的状态

## (3) 控制寄存器

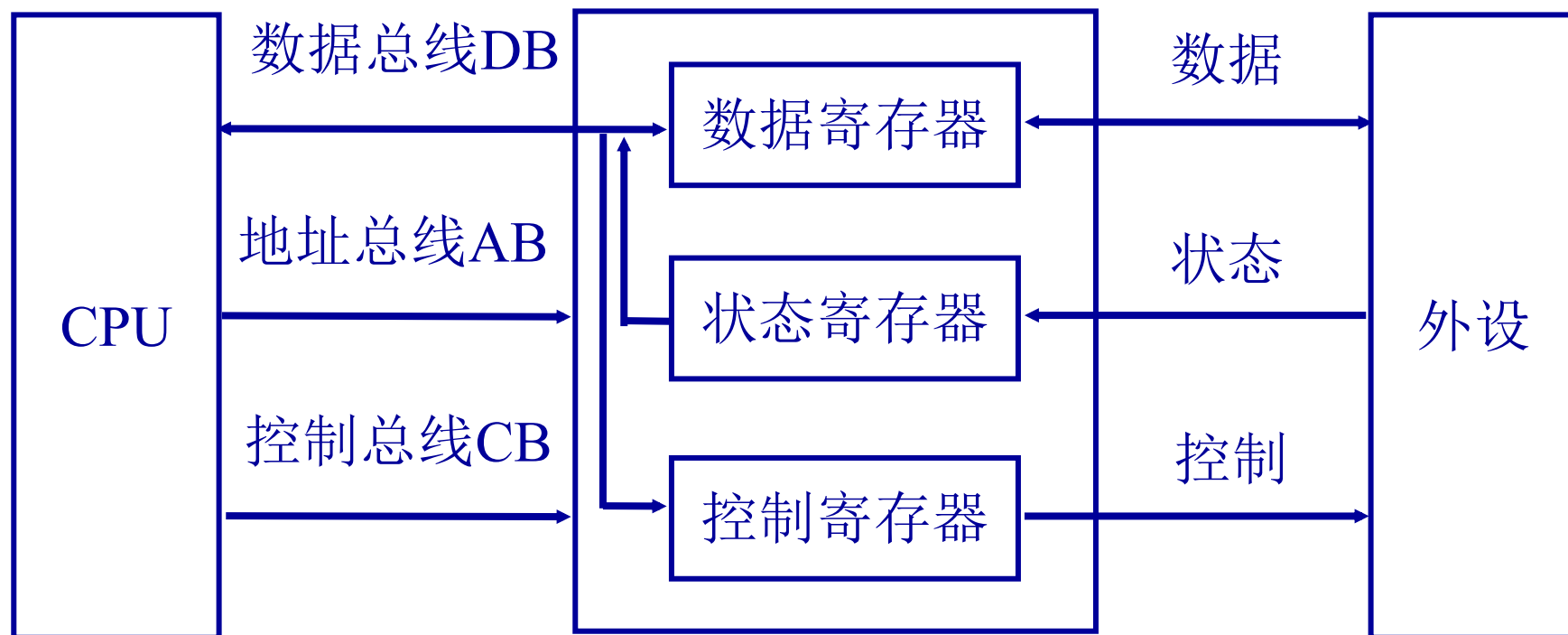
- ◆ 保存CPU给外设或接口电路的命令

结构图



## 6.1.2 I/O接口的典型结构

I/O接口电路



返回

## 2. 接口电路的外部特性

主要体现在引脚上，分成连接微机和连接外设的两类信号。

面向**CPU**一侧的信号：

- ◆ 用于与**CPU**（系统总线）连接
- ◆ 主要是数据、地址和控制信号

面向外设一侧的信号：

- ◆ 用于与外设连接
- ◆ 提供的信号五花八门
- ◆ 功能定义、时序及有效电平等差异较大

### 3. 接口电路芯片的分类

接口电路的核心部分往往是一块或数块大规模集成电路芯片，通常称为接口芯片。

- ◇ 通用接口芯片

- ◆ 支持通用的数据输入输出和控制的接口芯片

- ◇ 面向外设的专用接口芯片

- ◆ 针对某种外设设计、与该种外设接口

- ◇ 面向微机系统的专用接口芯片

- ◆ 与**CPU**和系统配套使用，以增强其总体功能

## 4. 接口电路的可编程性

- ◇ 许多接口电路具有多种功能和工作方式，可以通过编程的方法选定其中一种。
- ◇ 接口需进行物理连接，还需编写接口软件。
- ◇ 接口软件有两类：
  - ◆ 初始化程序段——设定芯片工作方式等
  - ◆ 数据交换程序段——管理、控制、驱动外设，负责外设和系统间信息交换

## 6.1.3 I/O端口的编址

接口电路占用的I/O端口地址有两类编排形式

### ◇ I/O端口单独编址

- ◆ I/O地址空间独立于存储地址空间

- ◆ 如8086/8088

### ◇ I/O端口与存储器统一编址

- ◆ 它们共享一个地址空间

- ◆ 如M6800

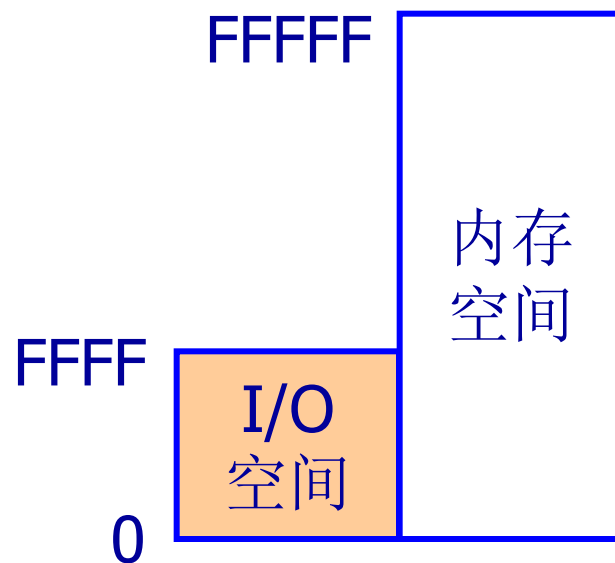
## (1) I/O端口单独编址

### ◇ 优点:

- ◆ I/O端口的地址空间独立
- ◆ 控制和地址译码电路相对简单
- ◆ 专门的I/O指令使程序清晰易读

### ◇ 缺点:

- ◆ I/O指令没有存储器指令丰富



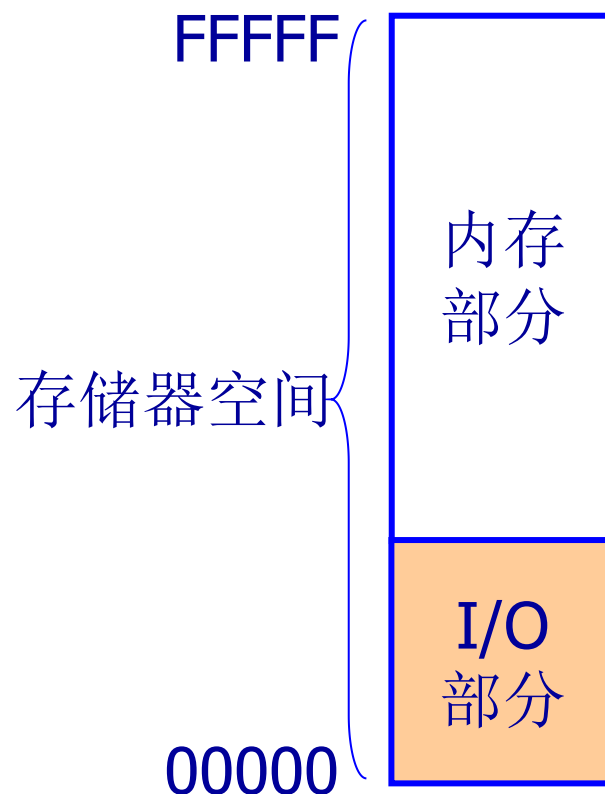
## (2) I/O端口与存储器统一编址

### ◇ 优点：

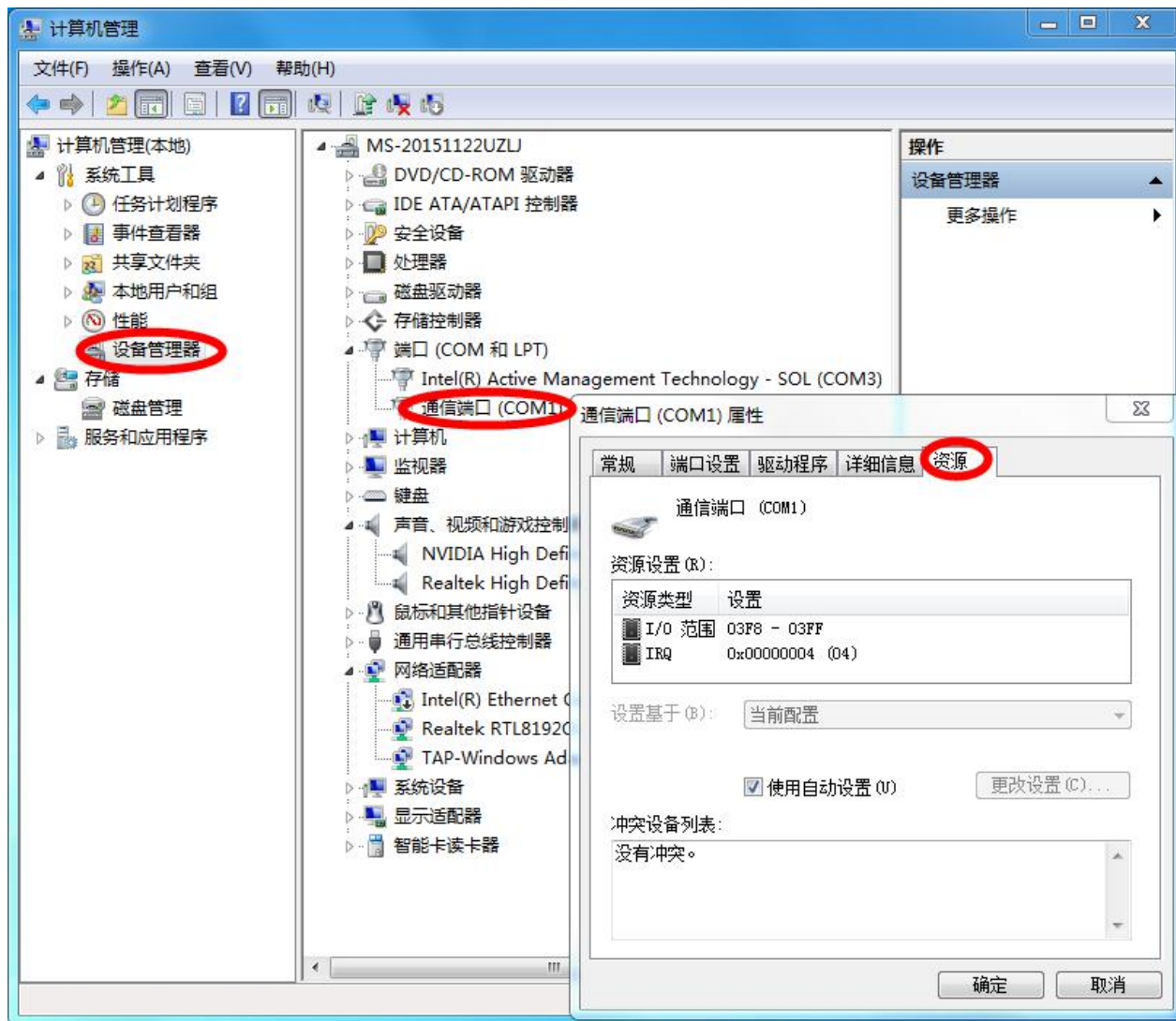
- ◆ 不需要专门的I/O指令
- ◆ I/O数据存取与存储器数据存取一样灵活

### ◇ 缺点：

- ◆ I/O端口要占去部分存储器地址空间
- ◆ 程序不易阅读（不易分清访存和访问外设）



# 端口地址是一种重要资源





# 8088/8086的I/O端口访问

- ◇ 8088可以通过输入输出指令与外设进行数据交换；呈现给程序员的外设是端口（Port）；
- ◇ 8088用于寻址外设端口的地址线为16条，8位端口数量可达 $2^{16} = 65536$ （64K）个，端口地址范围0000H ~ FFFFH；

## 6.1.4 8088/8086的输入输出指令

- ◇ 输入指令（IN：将外设数据传送给CPU内的AL/AX）

IN AL,i8 ;字节输入

IN AL,DX ;字节输入

IN AX,i8 ;字输入

IN AX,DX ;字输入

IN指令

- ◇ 输出指令（OUT：将CPU内的AL/AX数据传送给外设）

OUT i8,AL ;字节输出

OUT DX,AL ;字节输出

OUT i8,AX ;字输出

OUT DX,AX ;字输出

OUT指令



# IN指令实例



例：从**20H**端口输入一个字

方法1：直接寻址，字量输入

```
in ax, 20h
```

方法2：间接寻址，字量输入

```
mov dx, 20h
```

```
in ax, dx
```



# IN指令实例



例：从20H端口输入一个字

方法3：直接寻址，字节量输入

```
in    al,21h  
mov  ah,al  
in    al,20h
```

方法4：间接寻址，字节量输入

```
mov  dx,21h  
in    al,dx  
mov  ah,al  
dec  dx  
in    al,dx
```



# OUT指令实例

例：向300H端口输出一个字节

唯一的方法：间接寻址，字节量输出

```
mov al,bvar          ; bvar是字节变量  
mov dx,300h  
out dx,al
```

## 6.1.5 I/O地址的译码

- ◇ I/O地址的译码方法与存储器地址的译码方法一样，但有它的特点：
  - ◆ 部分译码
    - 通常是中间地址线不连接
    - 部分译码也有最低地址线不连接的情况
  - ◆ 每个接口电路通常只占用几个I/O地址，这时可以利用基本逻辑门电路进行地址译码
  - ◆ 除采用译码器、门电路进行译码外，I/O地址译码还经常采用可编程逻辑器件PLD
  - ◆ 为了给系统一定的选择余地，有些接口电路利用比较器、开关或跨接器等进行多组I/O地址的译码

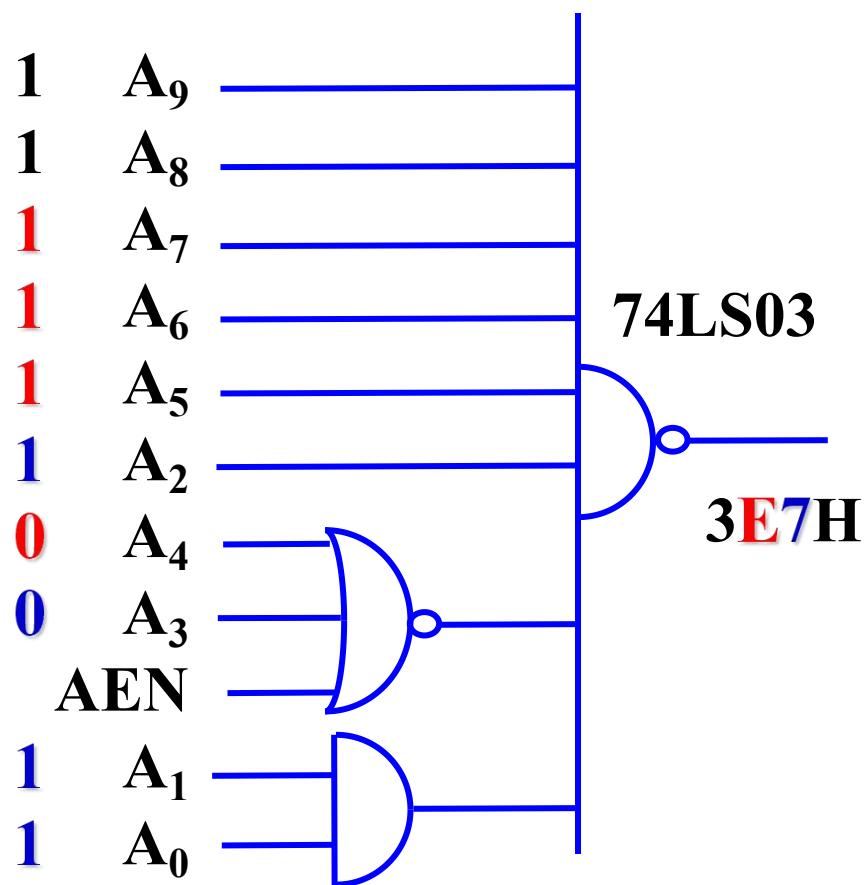
CPU

输入接口

输出接口

# 1. 利用逻辑门电路进行I/O地址译码

例：地址是**3E7H**的端口地址译码电路



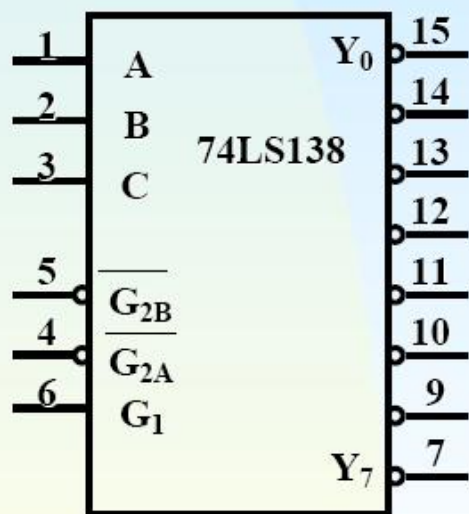
端口实例

## 2. 用译码器设计口地址译码电路

- ◇ 当接口电路需要多个端口地址的时候，采用译码器简化设计。

**例：**用138译码器设计端口地址译码电路，针对端口地址36CH~36FH产生低电平输出信号。

### 74LS138译码器：



**工作条件：**

$$G_1=1, \overline{G_{2A}}=\overline{G_{2B}}=0。$$

**工作原理：**

将复合的输入信号变为枚举的输出信号。

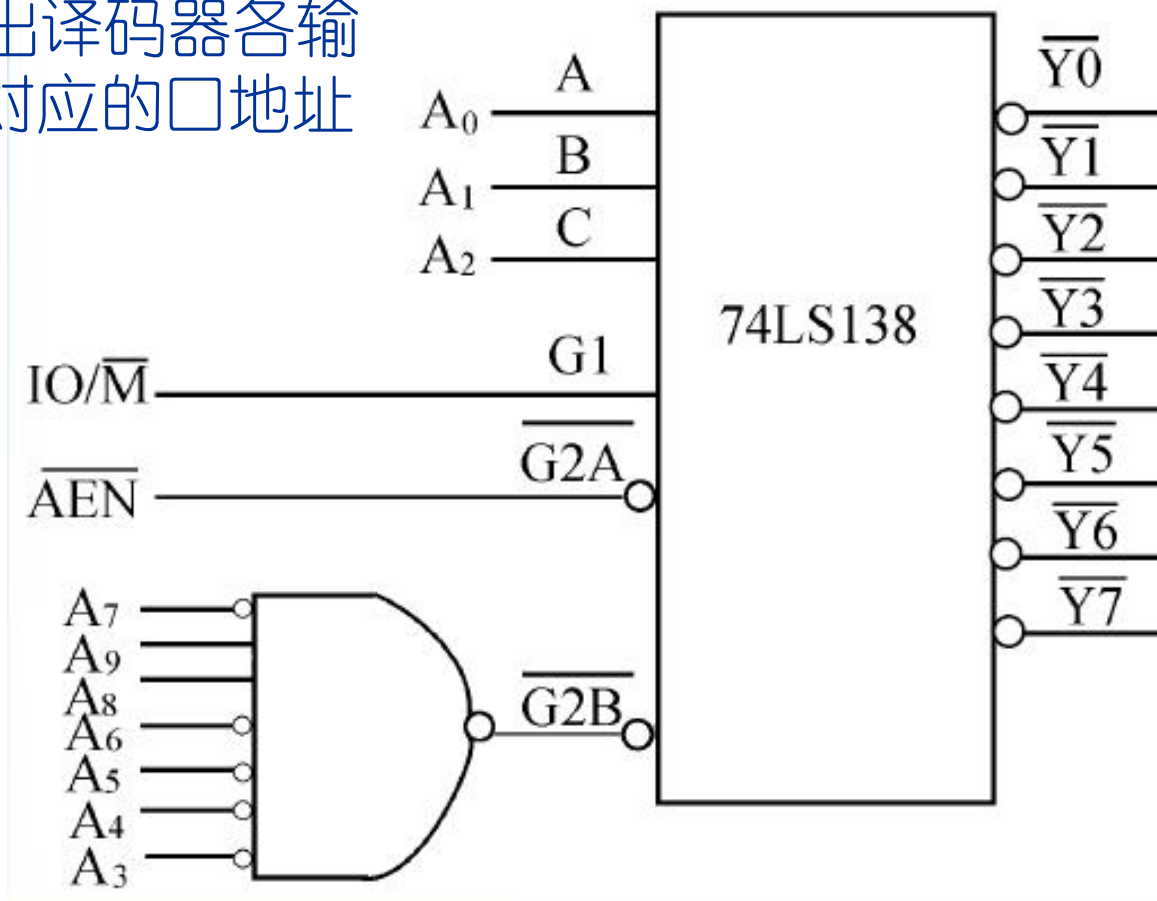


# 74LS138真值表

输 入			输 出							
$\overline{G_1}$ $\overline{G_{2A}}$ $\overline{G_{2B}}$	C	B	A	$Y_7$	$Y_6$	$Y_5$	$Y_4$	$Y_3$	$Y_2$	$Y_1$ $Y_0$
1 0 0	0	0	0	1	1	1	1	1	1	0
1 0 0	0	0	1	1	1	1	1	1	0	1
1 0 0	0	1	0	1	1	1	1	0	1	1
1 0 0	0	1	1	1	1	1	0	1	1	1
1 0 0	1	0	0	1	1	1	0	1	1	1
1 0 0	1	0	1	1	1	0	1	1	1	1
1 0 0	1	1	0	1	0	1	1	1	1	1
1 0 0	1	1	1	0	1	1	1	1	1	1
0 X X	X	X	X	1	1	1	1	1	1	1
X 1 X	X	X	X	1	1	1	1	1	1	1
X X 1	X	X	X	1	1	1	1	1	1	1

## 2. 用译码器设计口地址译码电路

例1 写出译码器各输出对应的□地址



AEN

端口实例

9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	0	0

# ISA总线信号说明

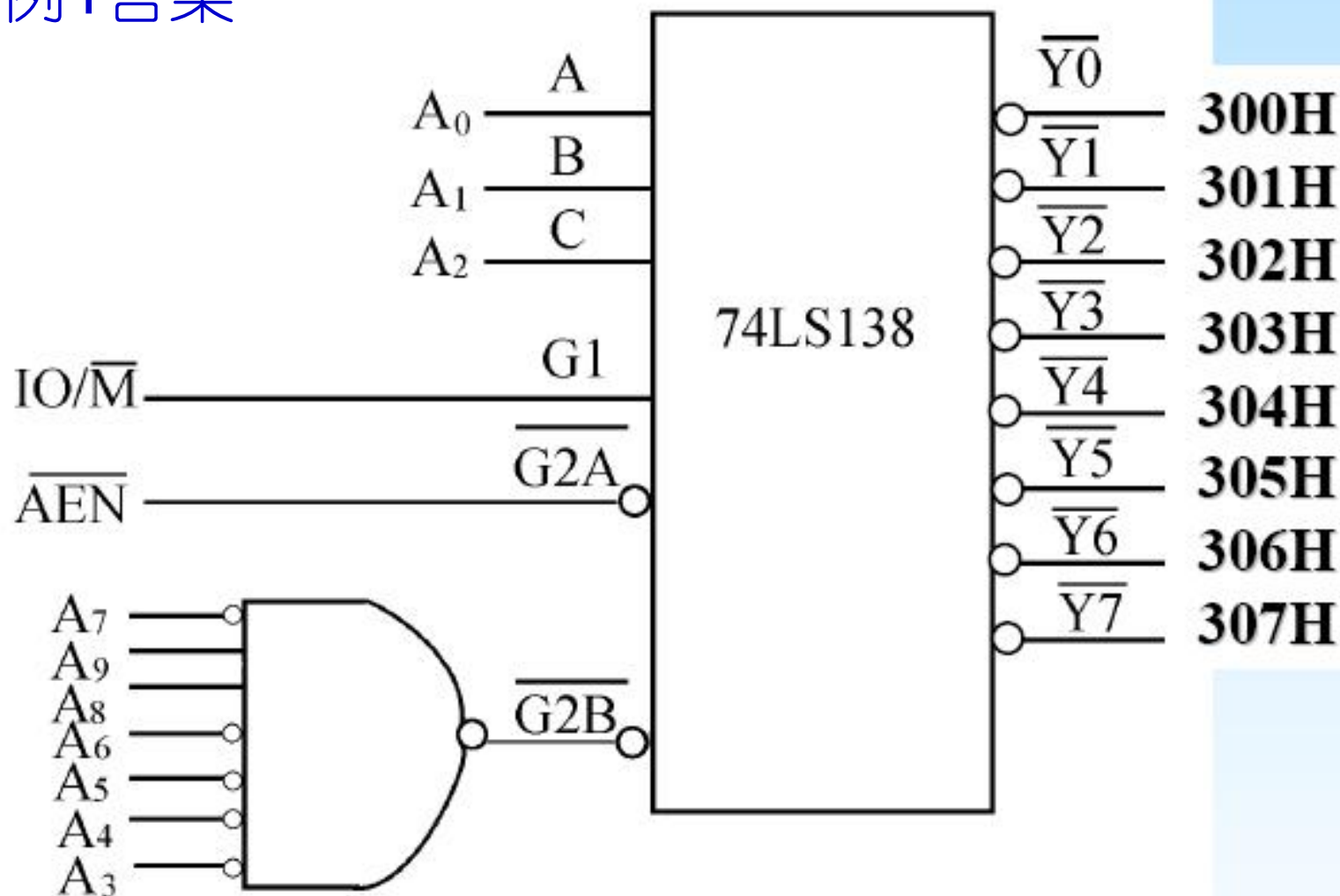
## AEN

- ◇ 地址允许，高电平表明**CPU**让出总线，**DMA**开始工作；
- ◇ 当设计非**DMA**方式的**I/O**接口时，应把**AEN**为低作为该接口工作的使能条件。以确保在总线上进行**DMA**传送时该接口不工作。



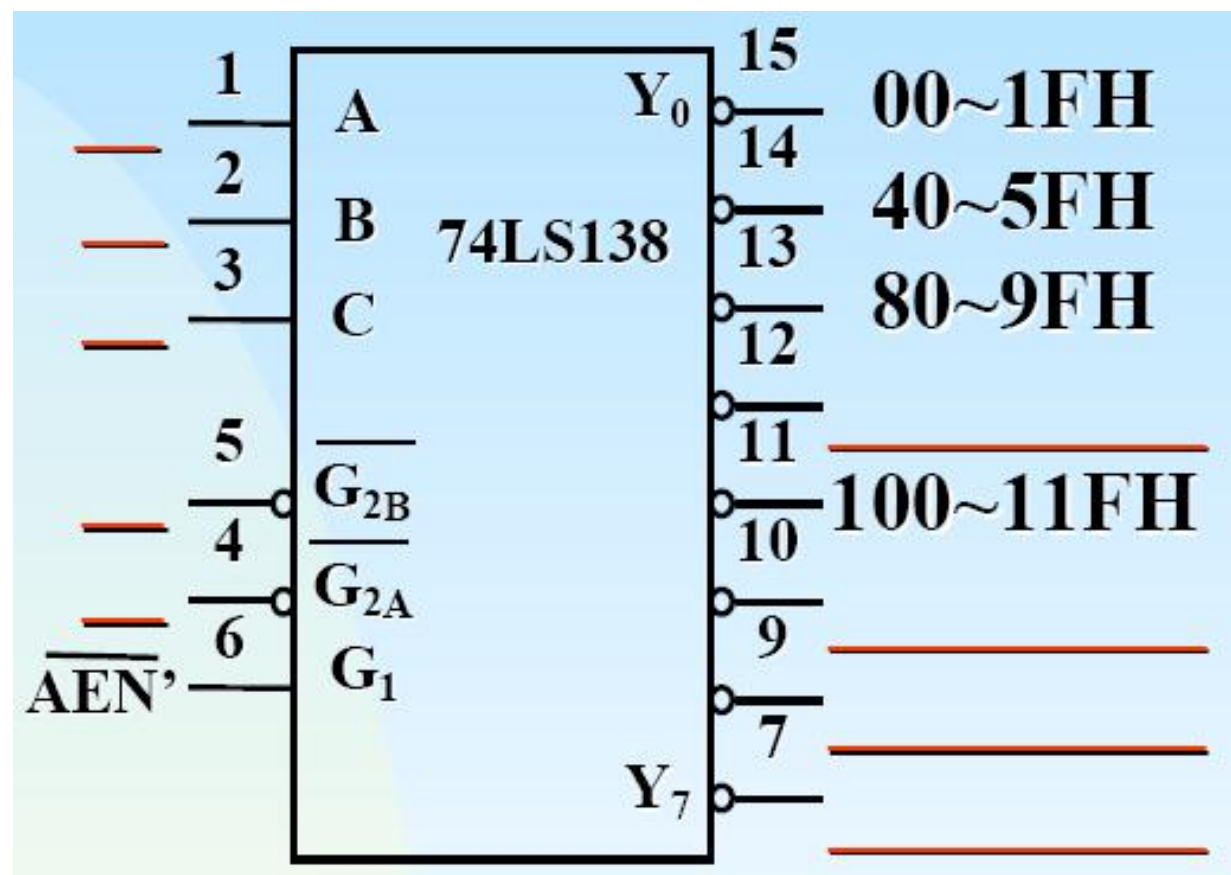
## 2. 用译码器设计口地址译码电路

### 例1答案



## 2. 用译码器设计口地址译码电路

例2 分析下图中的地址译码电路，将译码输入信号及各输出对应的地址补充完整。

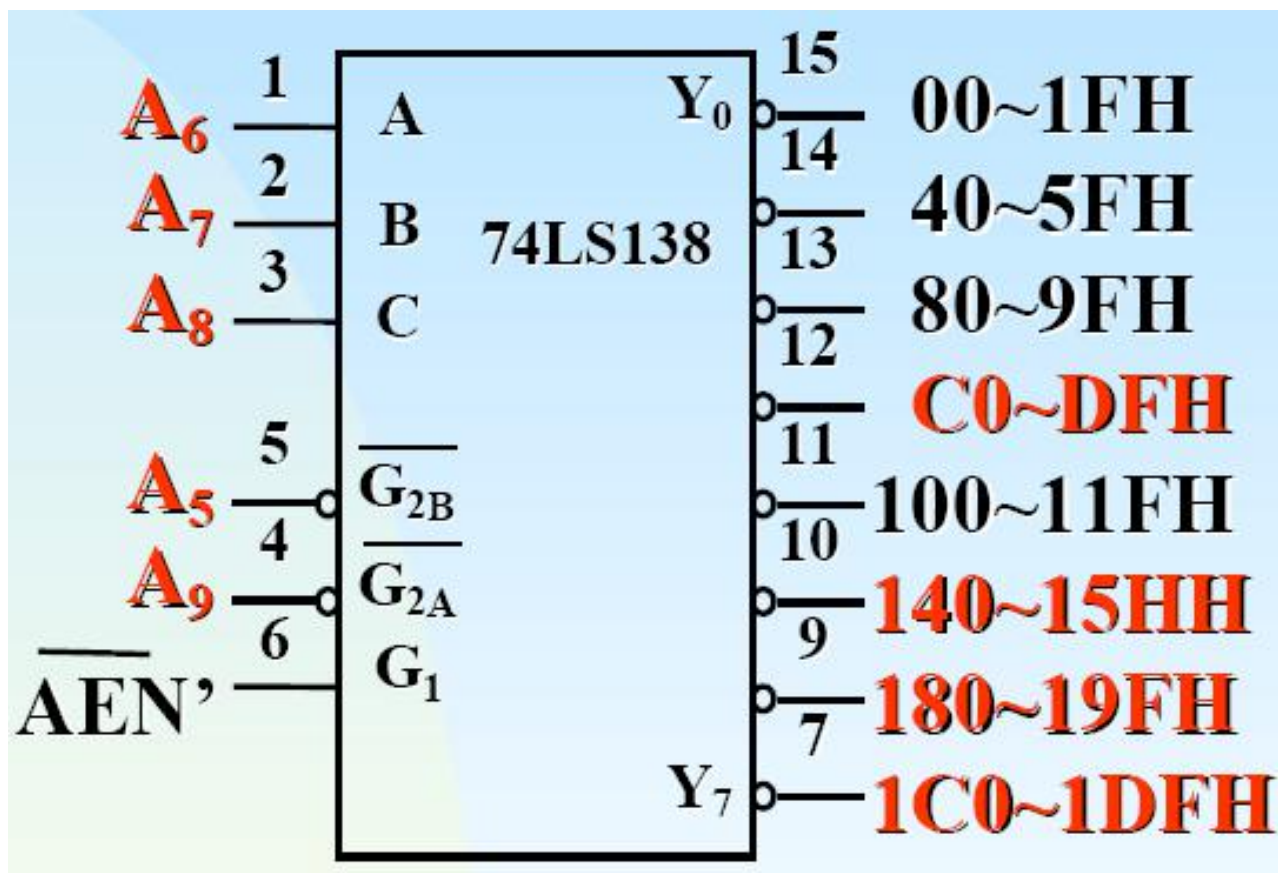




A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0				0	X	X	X	X	X
A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1
A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	1	1	1	1	1
A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	1	1	1	1	1
A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	1	1	1	1	1

## 2. 用译码器设计口地址译码电路

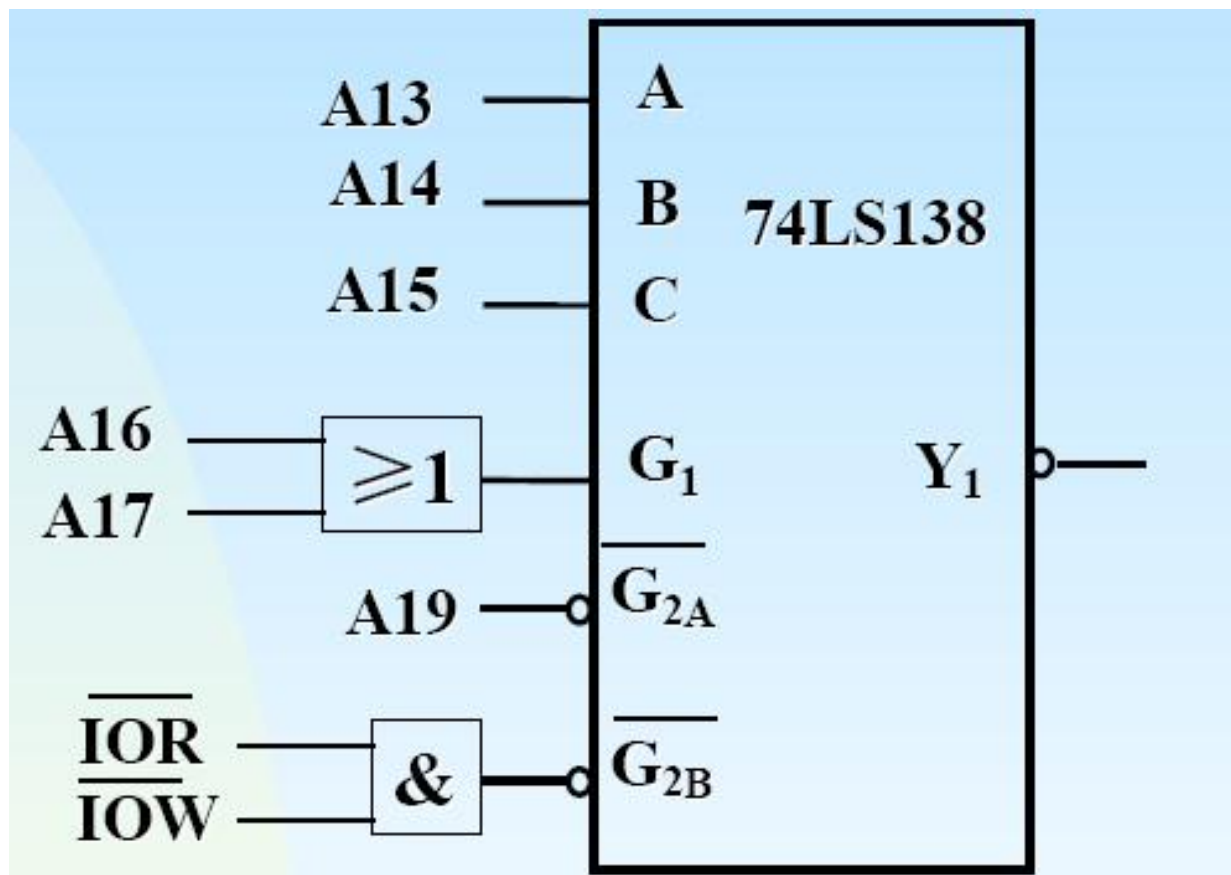
### 例2答案





## 2. 用译码器设计口地址译码电路

例3 分析下图中的译码电路，给出Y1输出对应的地址值(或地址范围)。

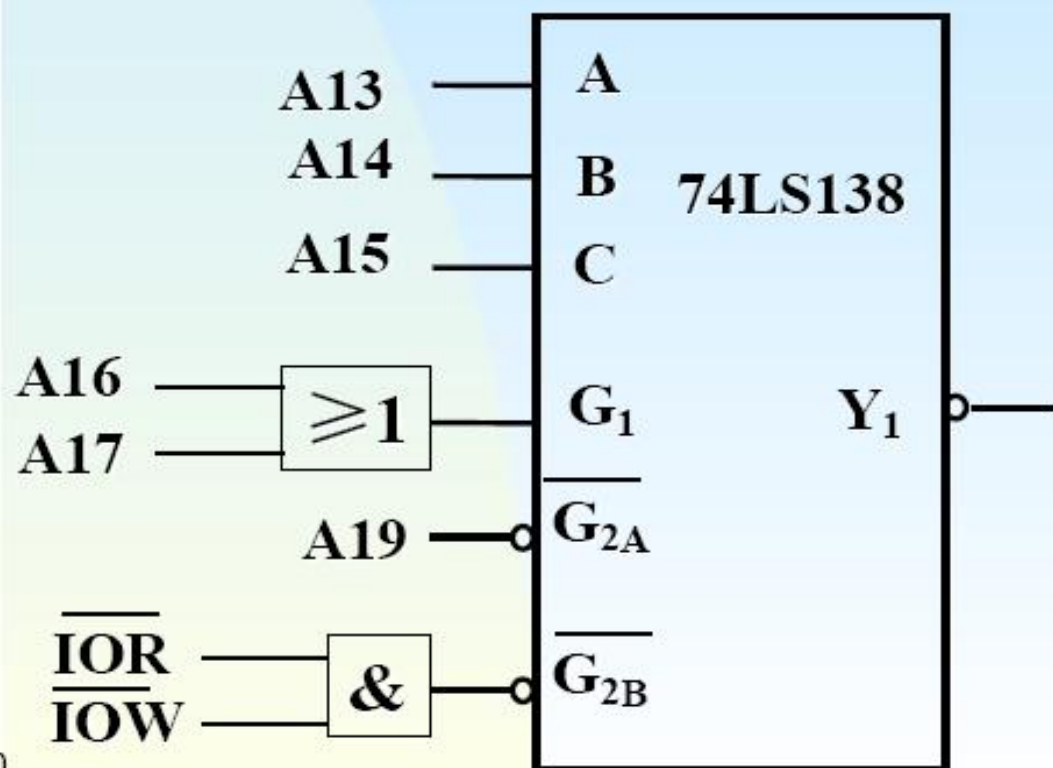




## 2. 用译码器设计口地址译码电路

$A_{19}$	$A_{18}$	$A_{17}$	$A_{16}$	$A_{15}$	$A_{14}$	$A_{13}$	$A_{12}$	$A_{11}$	$A_{10}$	$A_9$	$A_8$	$A_7$	$A_6$	$A_5$	$A_4$	$A_3$	$A_2$	$A_1$	$A_0$
0	×	0	1	0	0	1	×	×	×	×	×	×	×	×	×	×	×	×	×
		1	0																
		1	1																

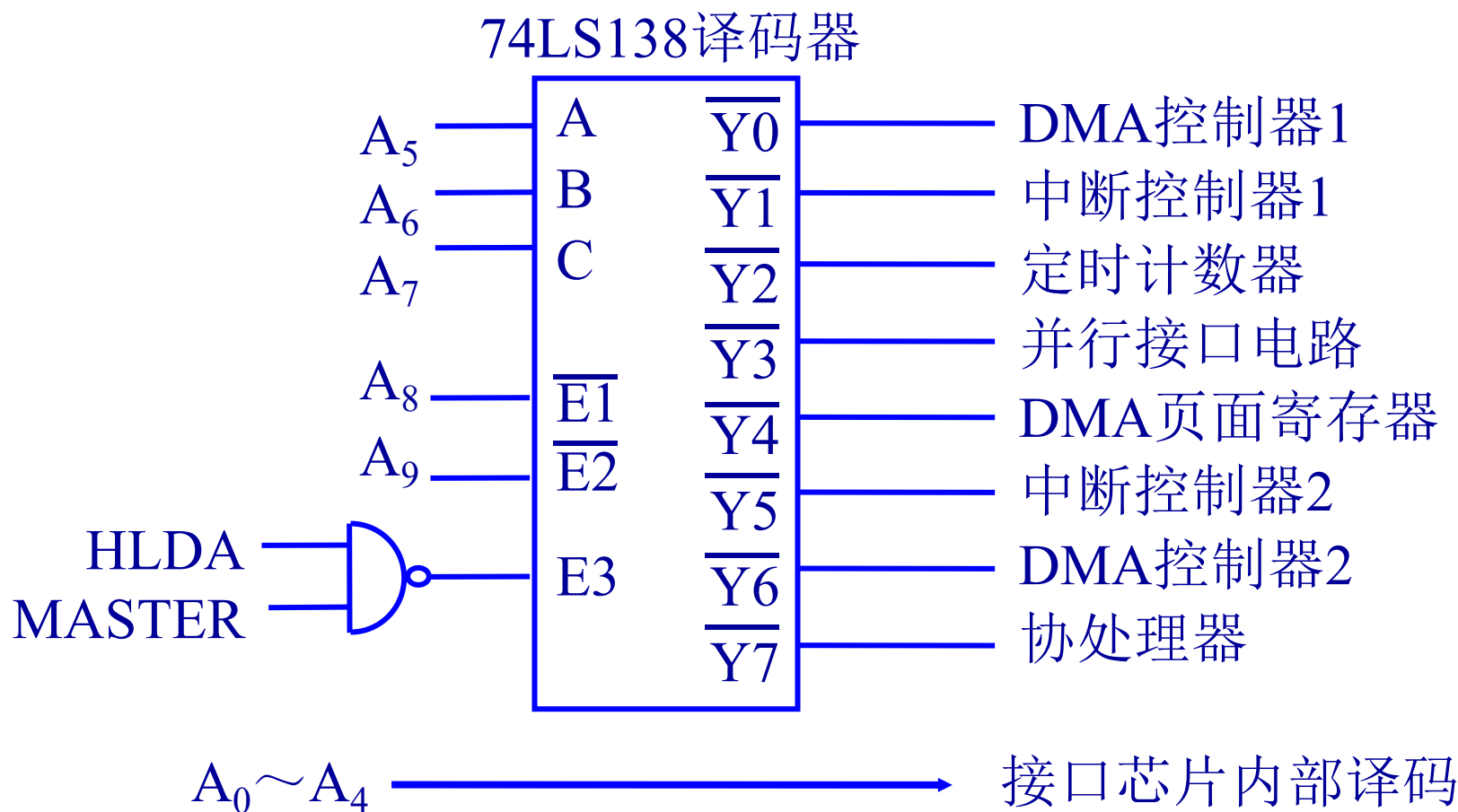
例3答案



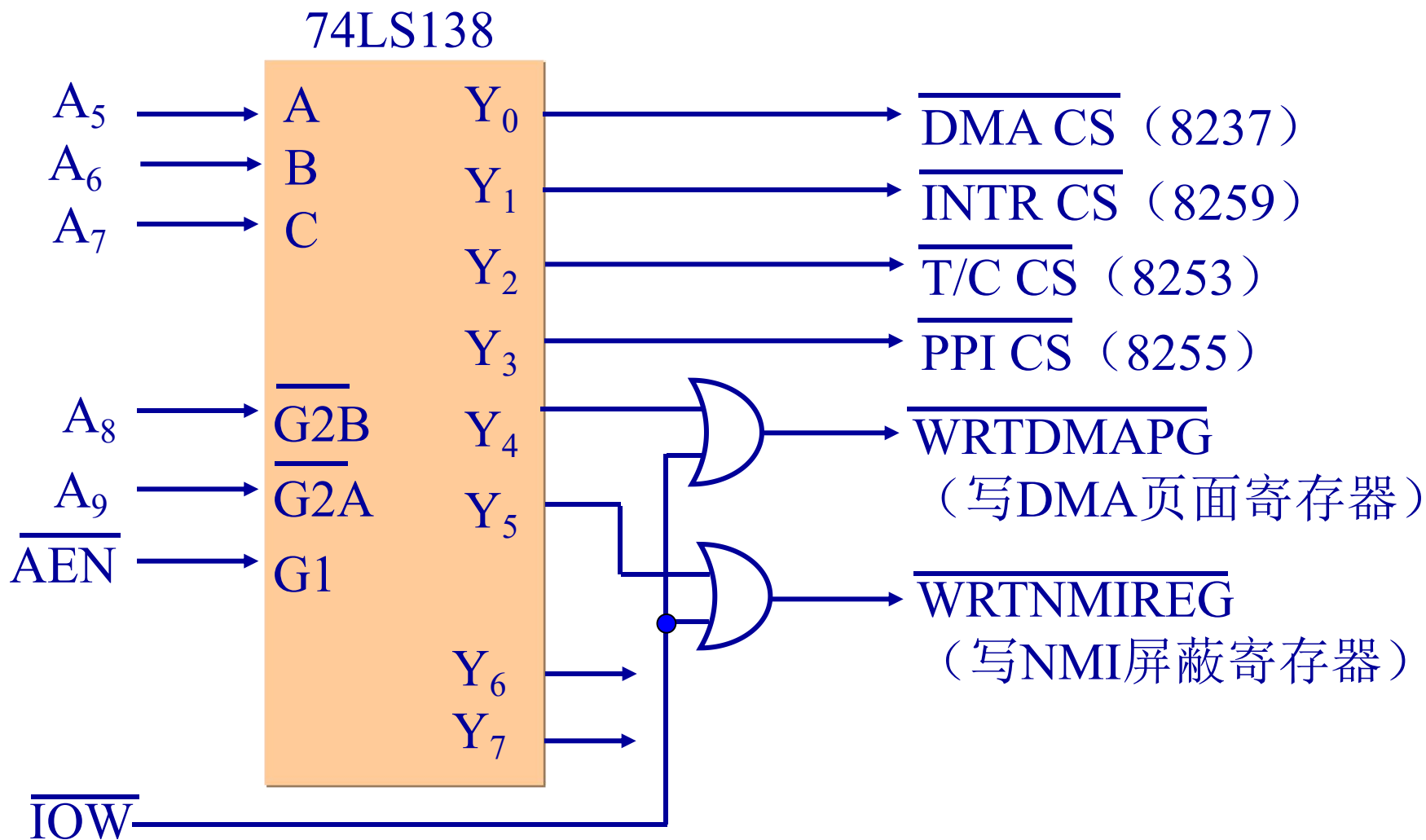
12000H~13FFFH  
 52000H~53FFFH  
 22000H~23FFFH  
 62000H~63FFFH  
 32000H~33FFFH  
 72000H~73FFFH

### 3. IBM PC/AT主机板的I/O译码电路

作业 请给出图中设计方案分配给中断控制器1的端口地址。



### 3. IBM PC/XT主机板的I/O译码电路



## 6.1.6 数据传送方式

- ◇ 程序控制下的数据传送——通过CPU执行程序中的I/O指令来完成传送，又分为：  
无条件传送、查询传送、中断传送。
- ◇ 直接存储器存取（DMA）——传送请求由外设向DMA控制器（DMAC）提出，后者向CPU申请总线，最后DMAC利用系统总线来完成外设和存储器间的数据传送。
- ◇ I/O处理机——CPU委托专门的I/O处理机来管理外设，完成传送和相应的数据处理。

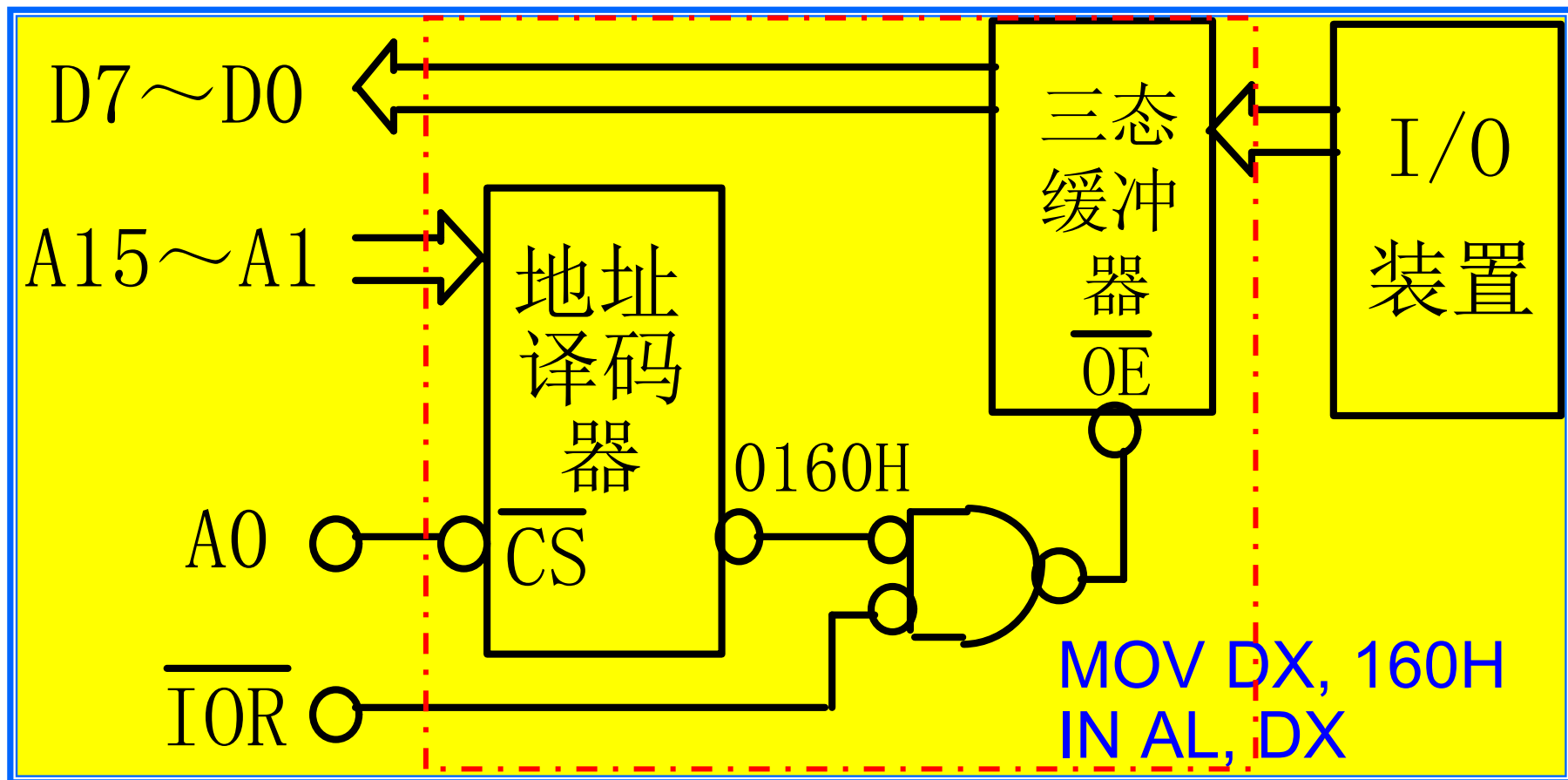
## 6.2 无条件传送方式及其接口

- ◇ 在CPU与慢速变化的设备交换数据时，可以认为它们总是处于“就绪”状态，随时可以进行数据传送，这就是无条件传送，或称立即传送、同步传送；
- ◇ 适合于简单设备，如LED数码管、按键或按钮等；
- ◇ 无条件传送的接口和操作均十分简单；
- ◇ 这种传送有前提：外设必须随时就绪。

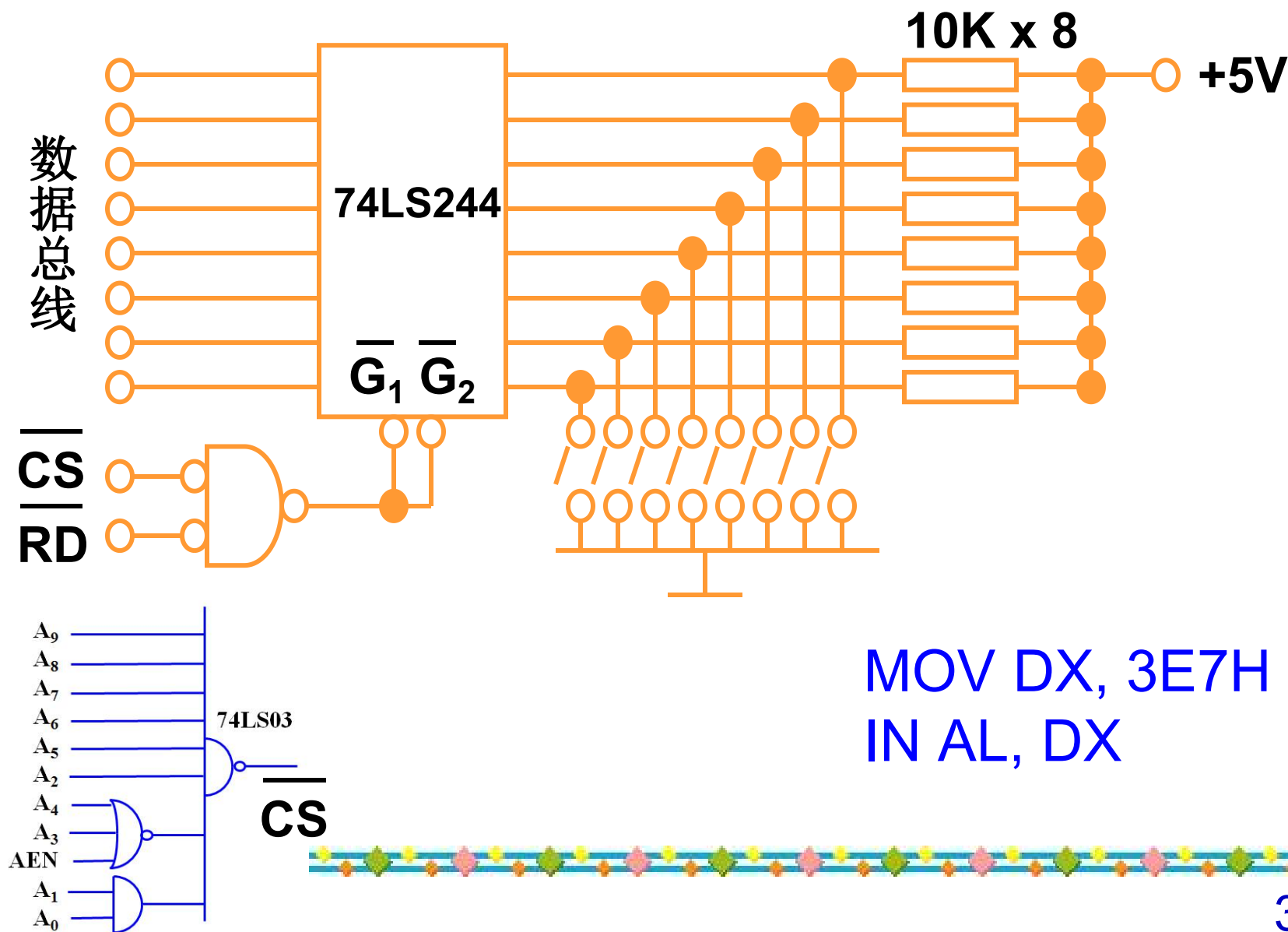


流程

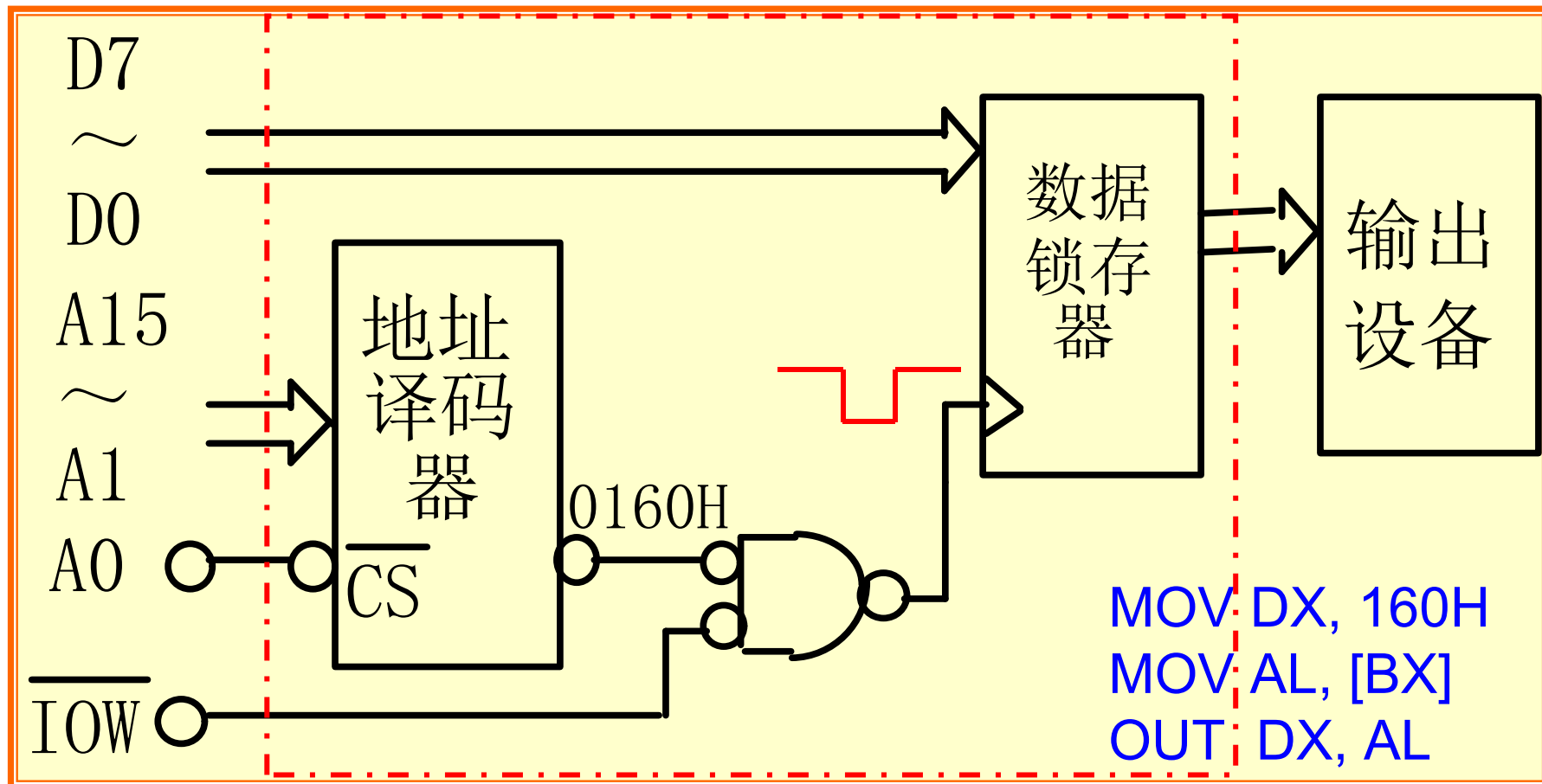
# 1. 无条件传送：输入示例



# 1. 无条件传送：输入实例

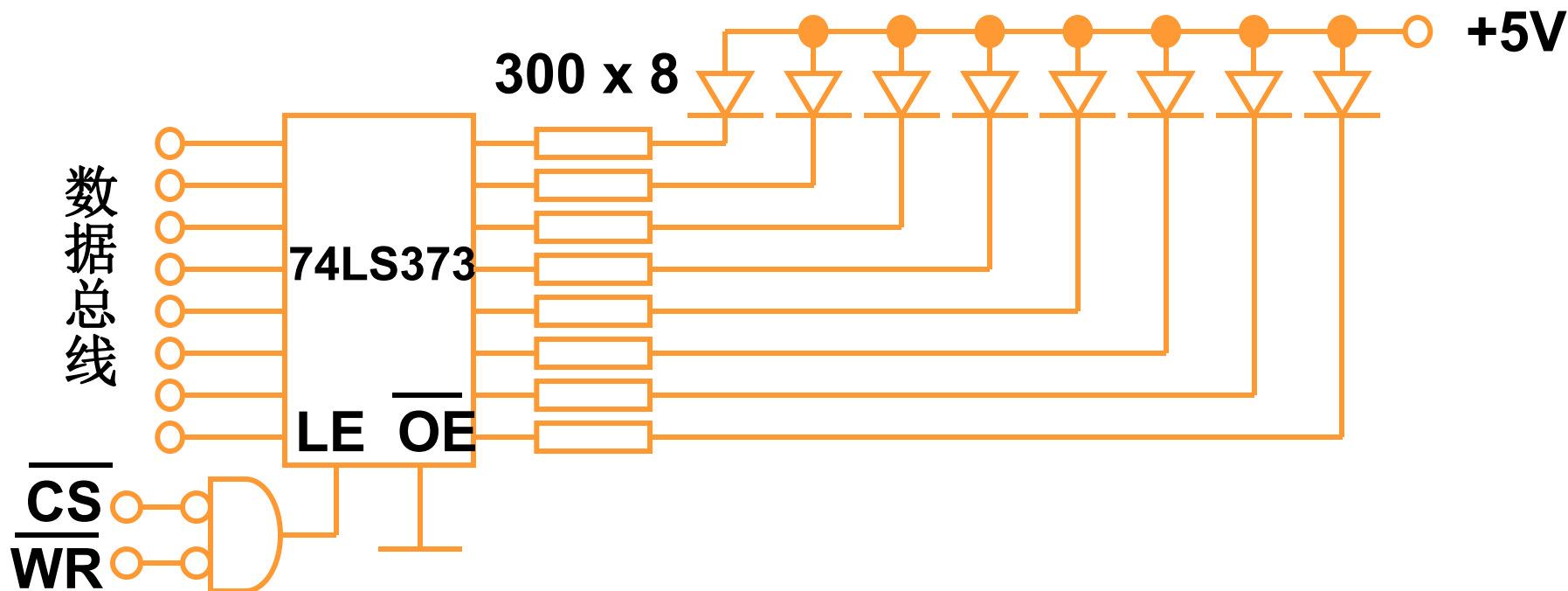


## 2. 无条件传送：输出示例



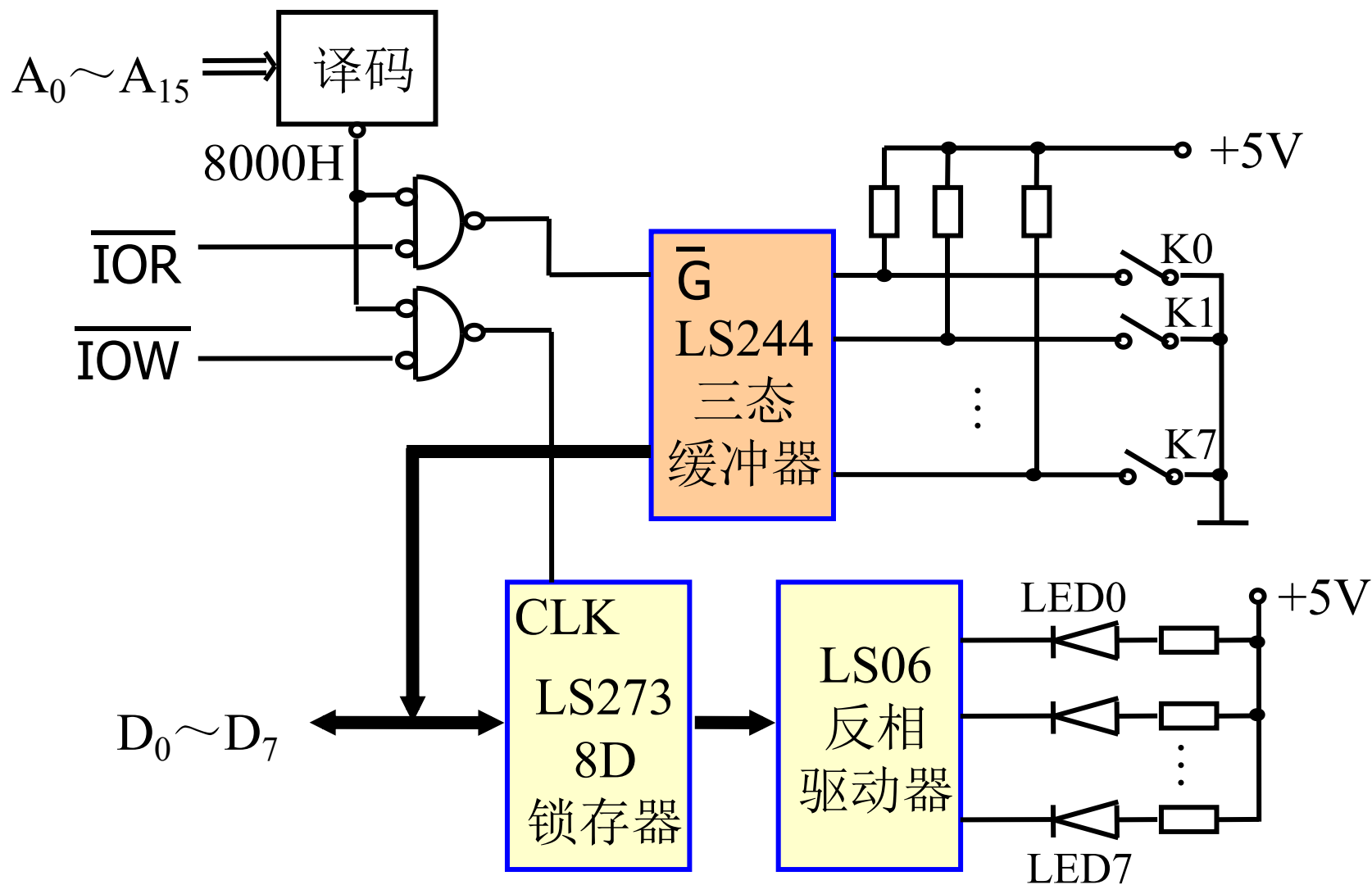


## 2. 无条件传送：输出实例



```
MOV  DX, 160H
MOV  AL, [BX]
OUT  DX, AL
```

### 3. 无条件传送：输入输出接口



### 3. 无条件传送：输入输出接口

在前面电路原理图的基础上编制程序，实现通过开关控制LED的点亮和熄灭，当某开关闭合时，其对应的LED点亮，否则熄灭。

```
next:  mov dx,8000h    ; DX指向数据端口
        in al,dx        ; 从输入端口读开关状态
        not al          ; 反相
        out dx,al       ; 送输出端口显示
        call delay    ; 调子程序延时
        jmp next        ; 重复
```

## 延时子程序delay

delay proc

push cx

mov cx,400H

Lp: loop lp

pop cx

ret

delay endp



## 6.3 查询传送方式及其接口

- ◇ **CPU**需要先查询外设的工作状态，然后在外设就绪的情况下实现数据输入或输出
- ◇ 对多个外设的情况，则**CPU**按一定顺序依次查询（轮询）。先查询的外设将优先进行数据交换
- ◇ 查询传送的特点是：工作可靠，适用面宽，但传送效率低

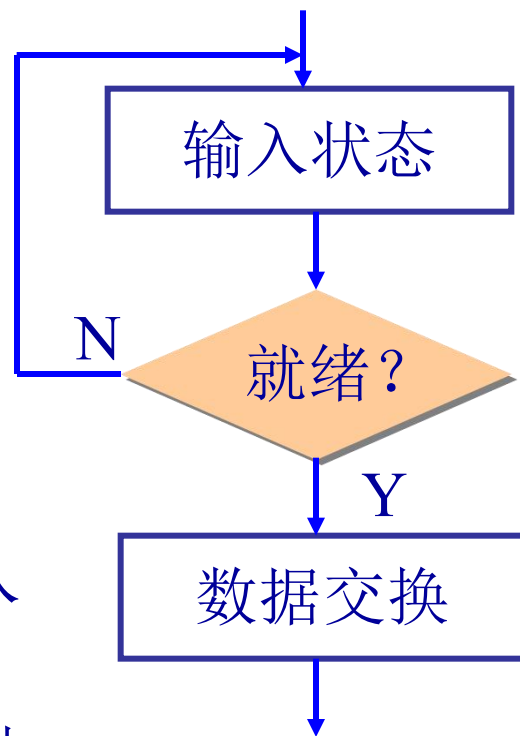
# 查询传送的两个环节

## (1) 查询环节

- ◆ 寻址状态口
- ◆ 读取状态寄存器的标志位
- ◆ 若不就绪就继续查询，直至就绪

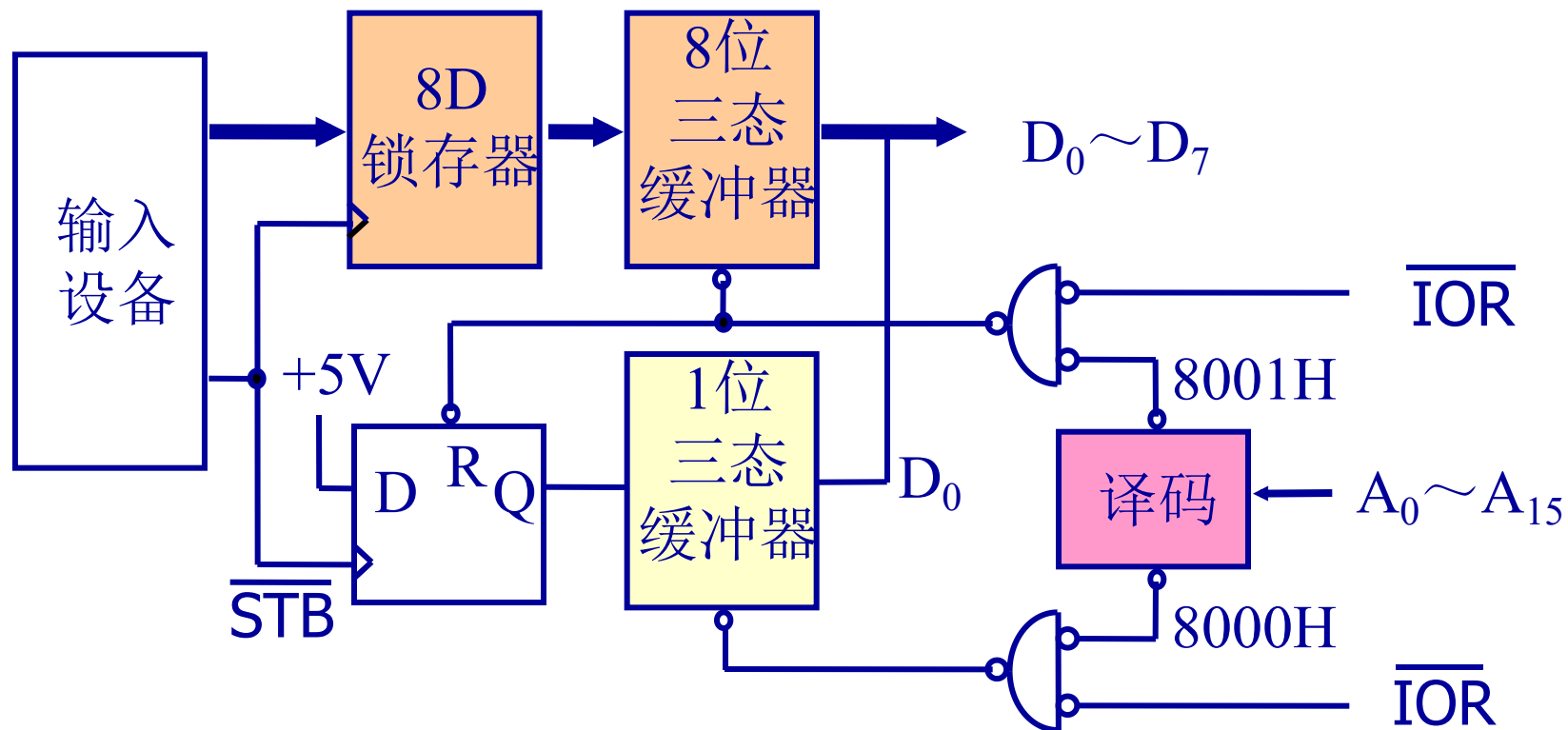
## (2) 传送环节

- ◆ 寻址数据口
- ◆ 是输入，通过输入指令从数据端口读入数据
- ◆ 是输出，通过输出指令向数据端口输出数据




流程

## 6.3.1 查询输入接口



程序

## 6.3.1 查询输入接口

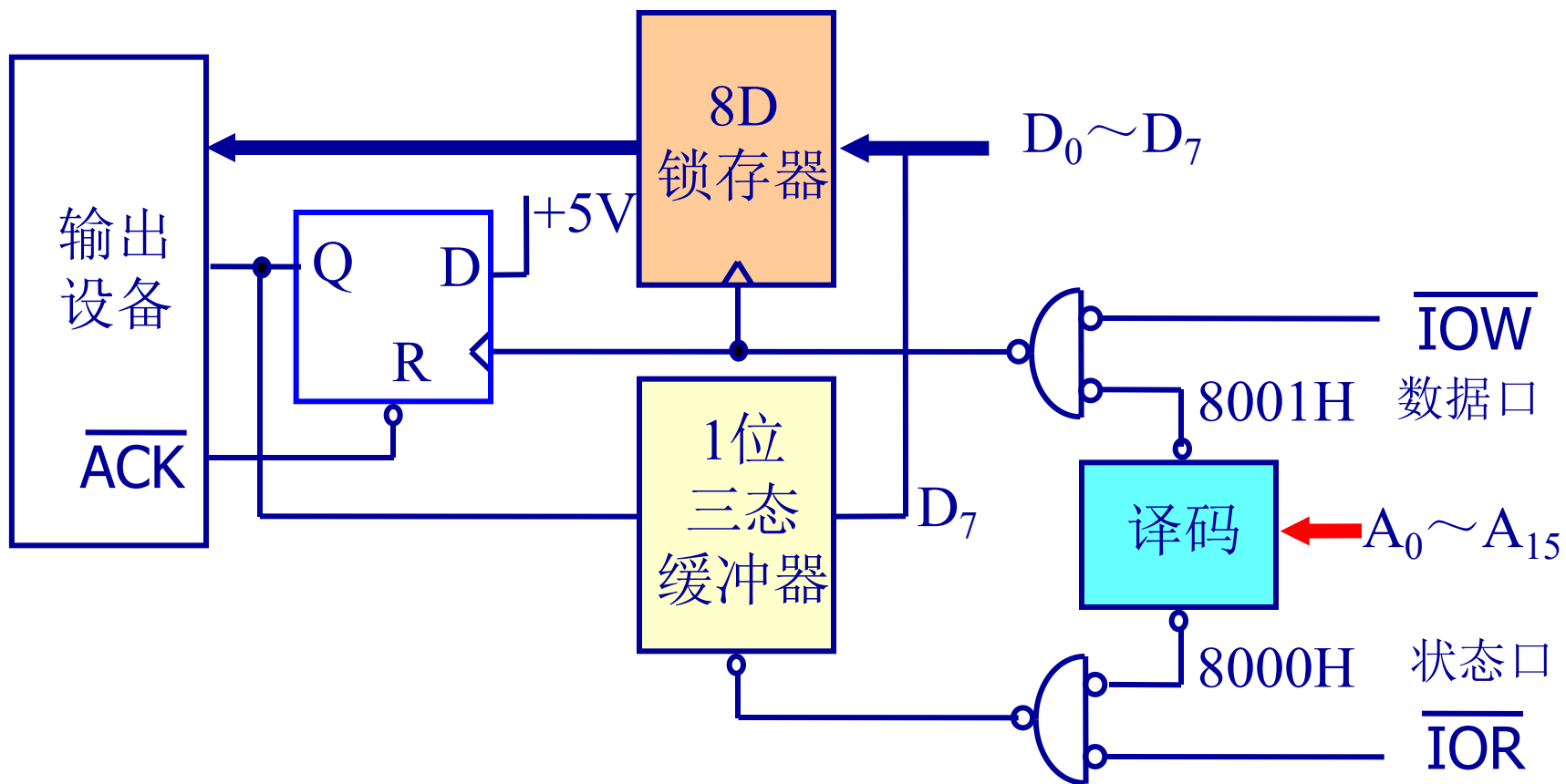


```
        mov dx,8000h    ; DX指向状态端口
wait:   in al,dx         ; 读状态端口
        test al,01h     ; 测试标志位D0
        jz wait         ; D0 = 0, 未就绪, 继续查询
        inc dx          ; D0 = 1, 就绪, DX指向数据端口
        in al,dx        ; 从数据端口输入数据
```






## 6.3.2 查询输出接口



程序

## 6.3.2 查询输出接口

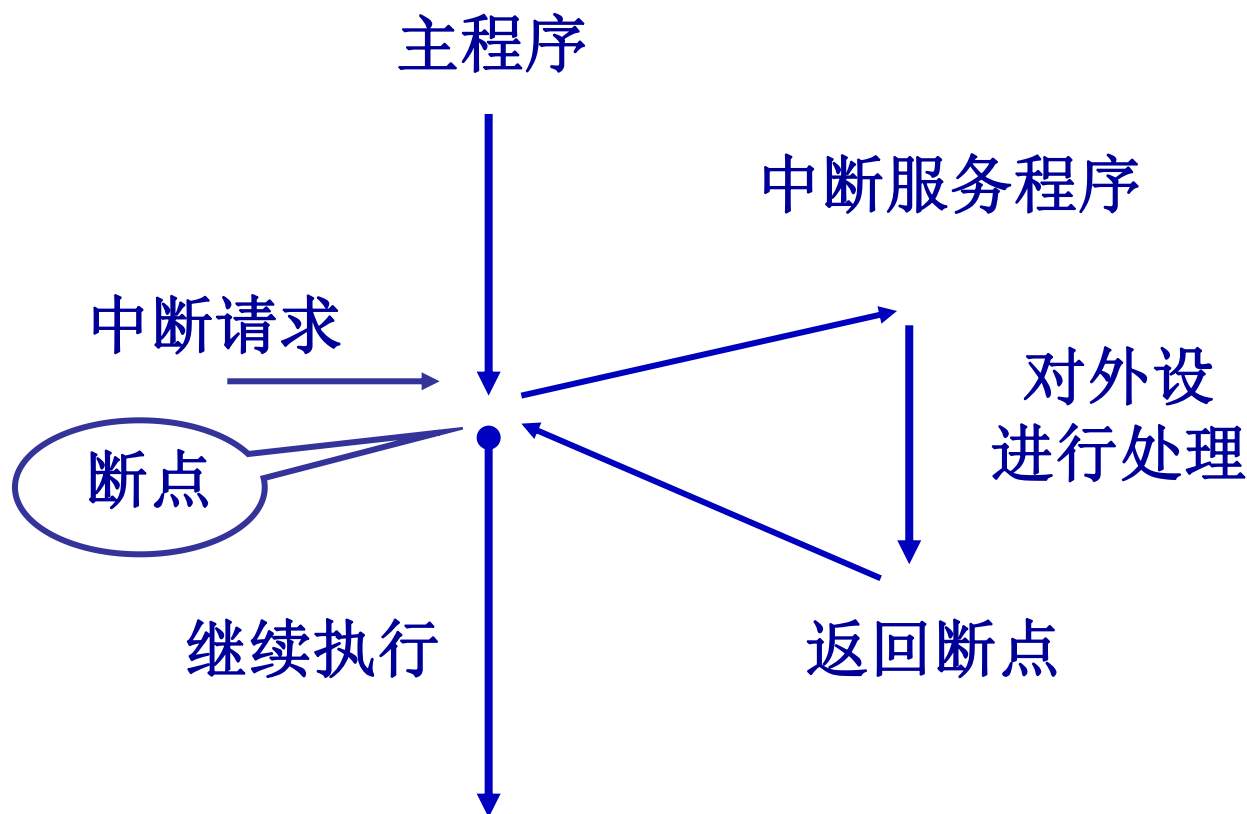


```
        mov dx,8000h    ; DX指向状态端口
status:  in al,dx        ; 读取状态端口的状态数据
        test al,80h     ; 测试标志位D7
        jnz status      ; D7 = 1, 未就绪, 继续查询
        inc dx          ; D7 = 0, 就绪, DX指向数据端口
        mov al,buf      ; 变量buf送AL
        out dx,al       ; 将数据输出给数据端口
```



## 6.4 中断传送方式

- ◇ CPU在执行程序时，被内部或外部的事件所打断，转去执行一段预先安排好的中断服务程序；服务结束后，又返回原来的断点，继续执行原来的程序。

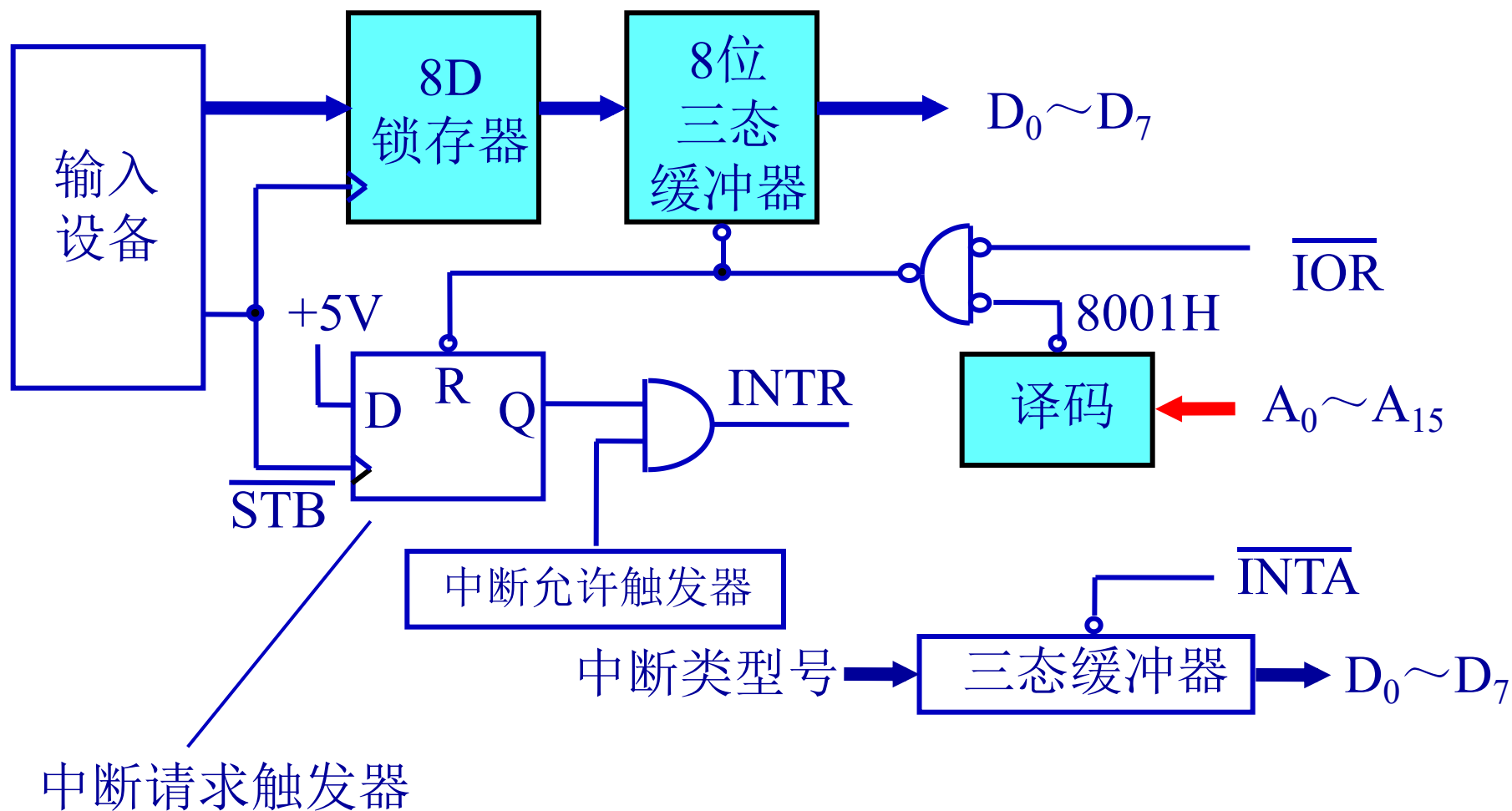


演示

## 6.4.1 中断传送与接口

- ◇ 中断传送是一种效率更高的程序传送方式
- ◇ 中断请求是外设随机向**CPU**提出的
- ◇ 进行传送的中断服务程序是预先设计好的
- ◇ **CPU**对请求的检测是有规律的：一般是在每条指令的最后一个时钟周期采样中断请求输入引脚

# 1. 中断输入接口



## 6.4.2 中断工作过程

- 中断请求
- 中断响应
- 关中断
- 断点保护
- 中断识别
- 现场保护
- 中断服务
- 恢复现场
- 开中断
- 中断返回

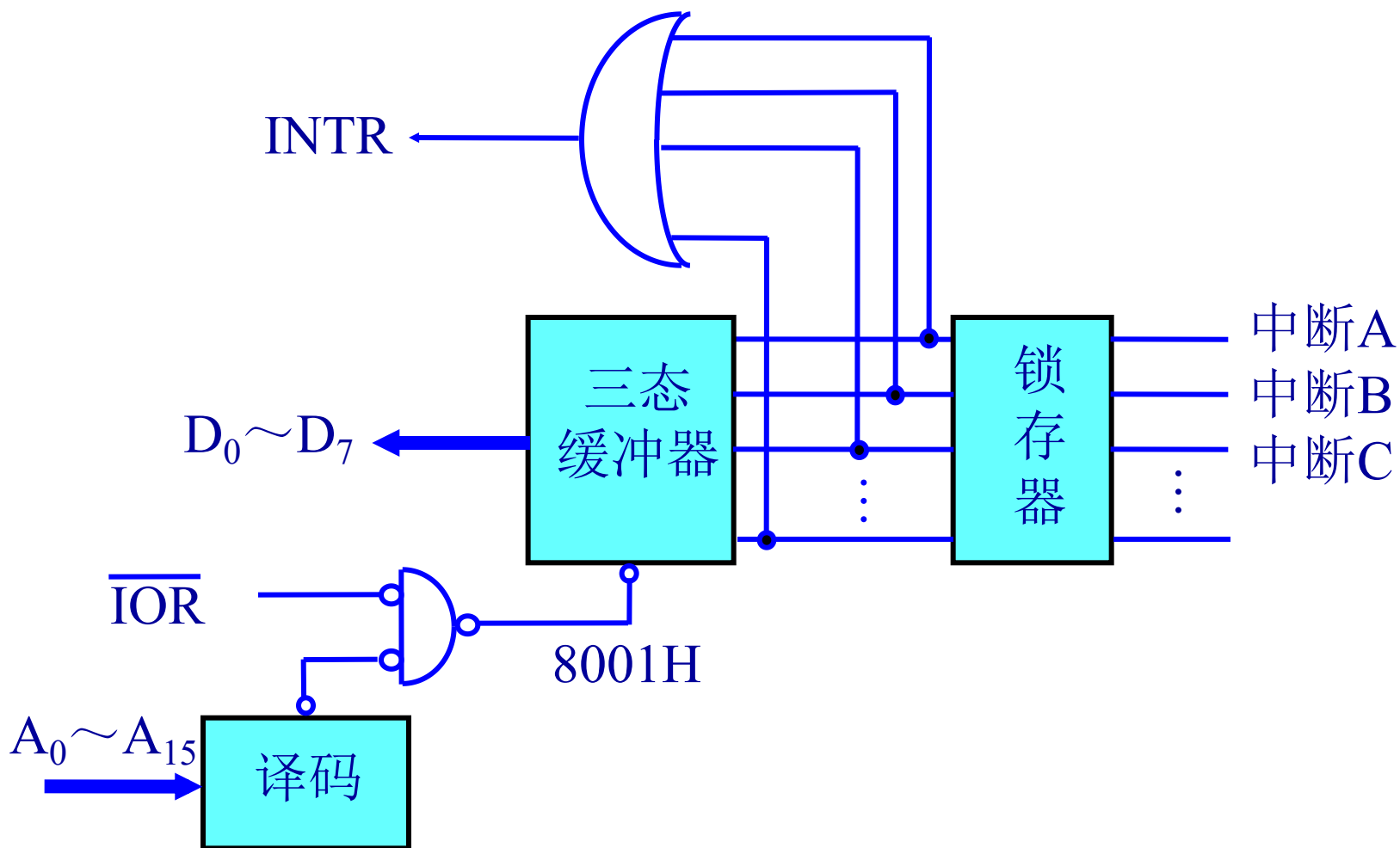
## 6.4.3 中断源识别和中断优先权管理

问题1：系统有多个中断请求，**CPU**如何识别中断源？

问题2：有多个中断同时请求，**CPU**如何应对？

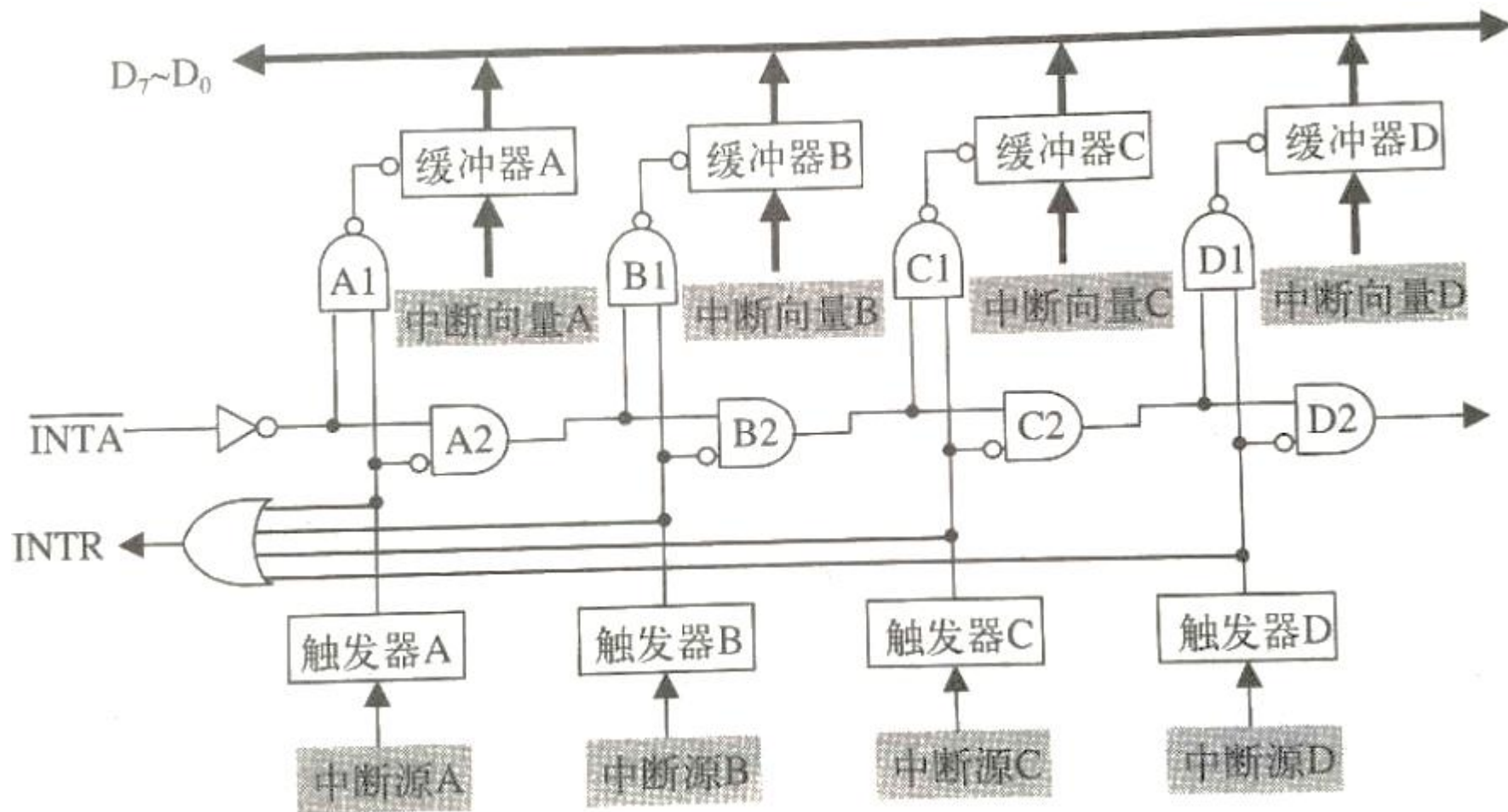
问题3：中断处理过程中，又有中断提出请求，怎么办？

# 1 中断源识别方案1:中断查询接口





## 2 中断优先权链式排队电路



### 3 中断嵌套

- ◇ 所谓中断嵌套即某一中断服务程序的执行被另一优先级别更高的中断请求中断。

## 6.5 DMA传送方式

- ◇ 希望克服程序控制传送的不足：

外设→CPU→存储器

外设←CPU←存储器

- ◇ 直接存储器存取DMA：

外设→存储器

外设←存储器

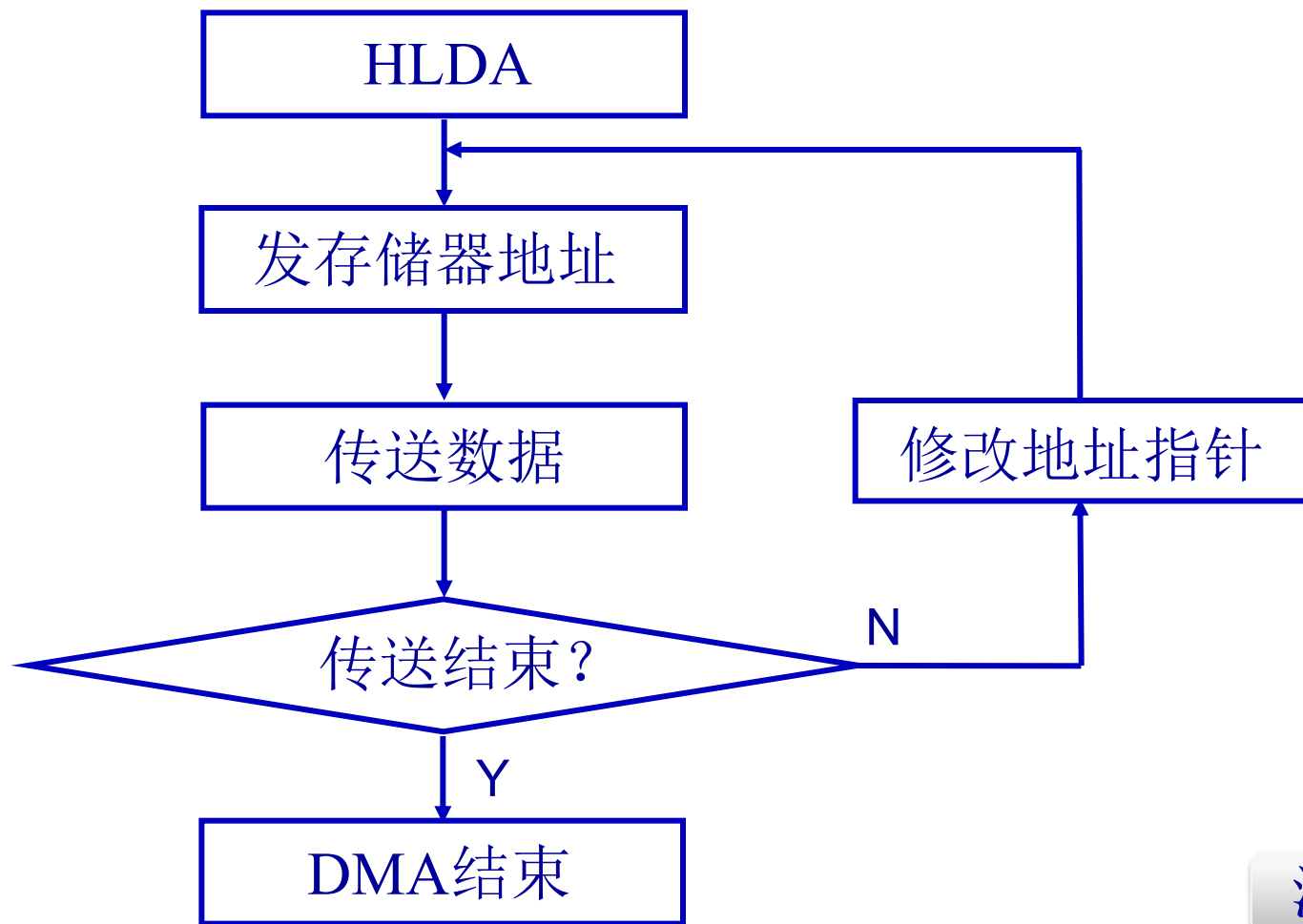
- ◇ CPU释放总线，由DMA控制器管理

# 1. DMA传送的工作过程



- (1) CPU对DMA控制器进行初始化设置
- (2) 外设、DMAC和CPU三者通过应答信号建立联系：CPU将总线交给DMAC控制
- (3) DMA传送
  - ◆ DMA读存储器：存储器  $\rightarrow$  外设
  - ◆ DMA写存储器：存储器  $\leftarrow$  外设
- (4) 自动增减地址和计数，判断传送完成否

## 2. DMA传送流程



演示

## 第6章：传送方式的比较



- ◇ 无条件传送：慢速外设需与**CPU**保持同步
- ◇ 查询传送：简单实用，效率较低
- ◇ 中断传送：外设主动，可与**CPU**并行工作，但每次传送需要大量额外时间开销
- ◇ **DMA**传送：**DMAC**控制，外设直接和存储器进行数据传送，适合大量、快速数据传送

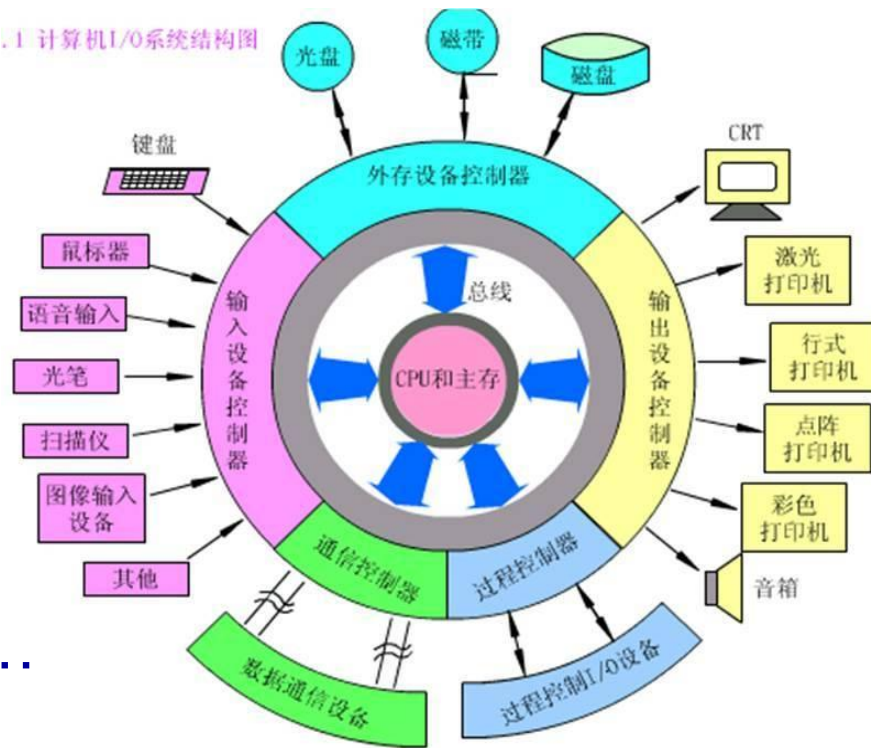


本章到此结束  
谢谢！

# 多种多样的外设

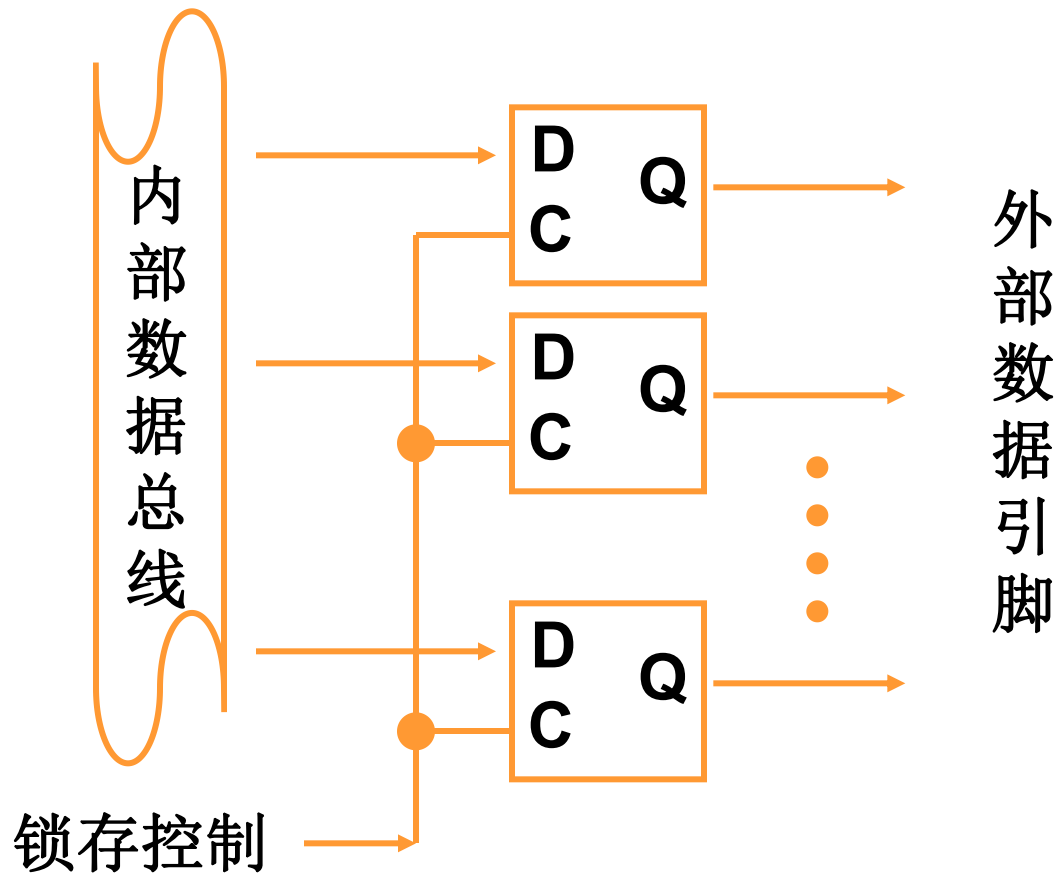
- ◇ 工作原理不同  
机械、电子、机电、电磁……
- ◇ 传送信息类型多样  
数字量、模拟量、开关量
- ◇ 传送速度差别极大
- ◇ 传送方式不尽相同  
串行、并行
- ◇ 编码方式不同  
二进制、BCD码、ASCII码……

图1 计算机I/O系统结构图

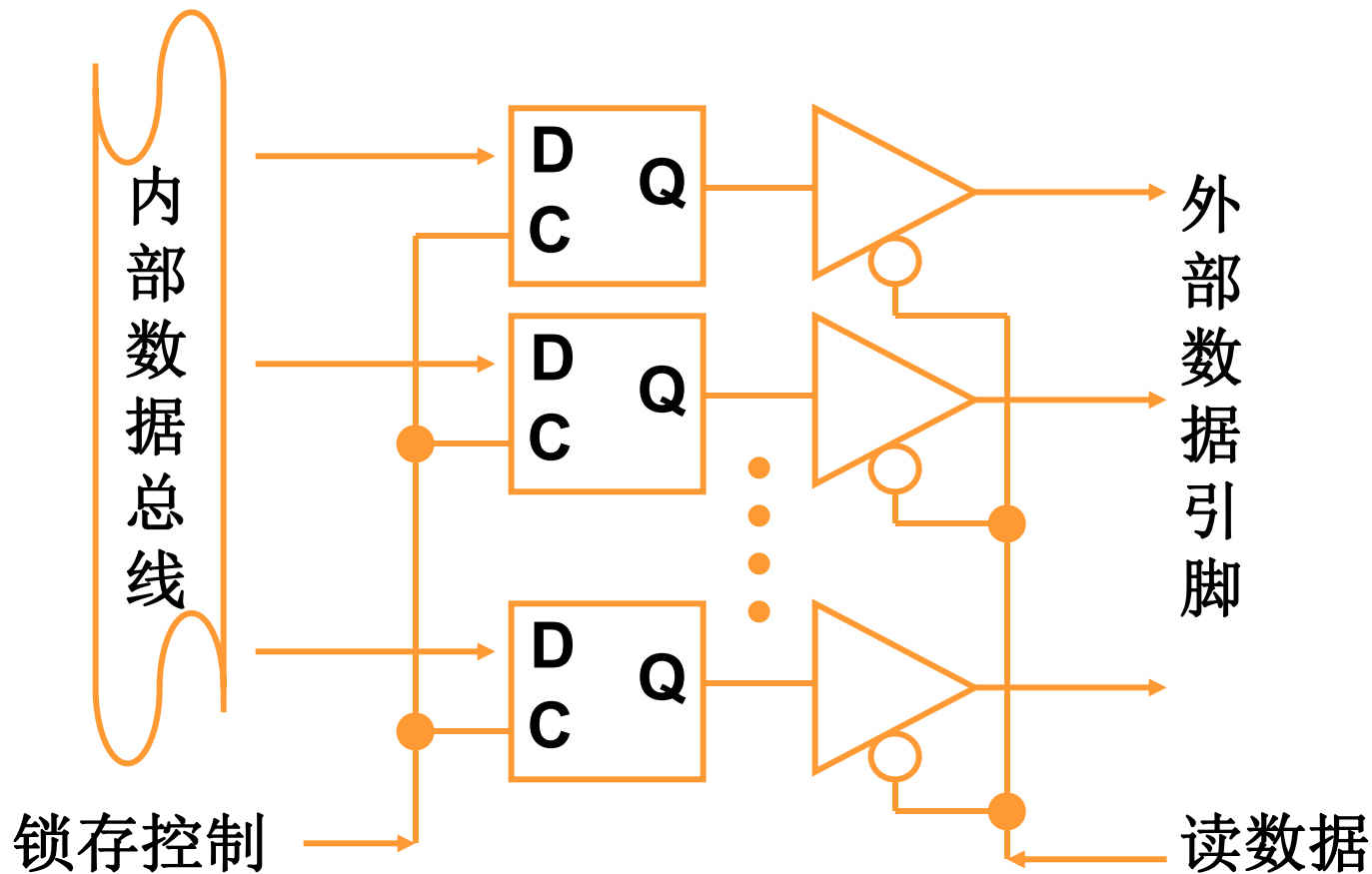




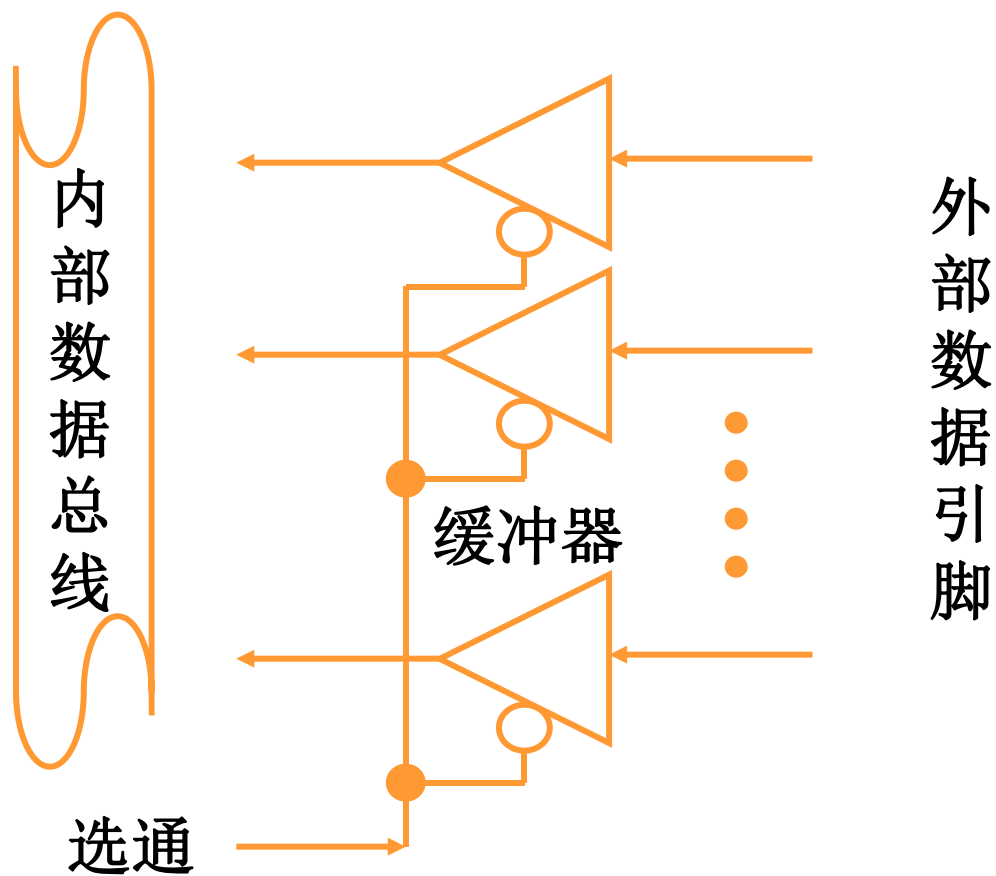
# 输出接口的锁存环节



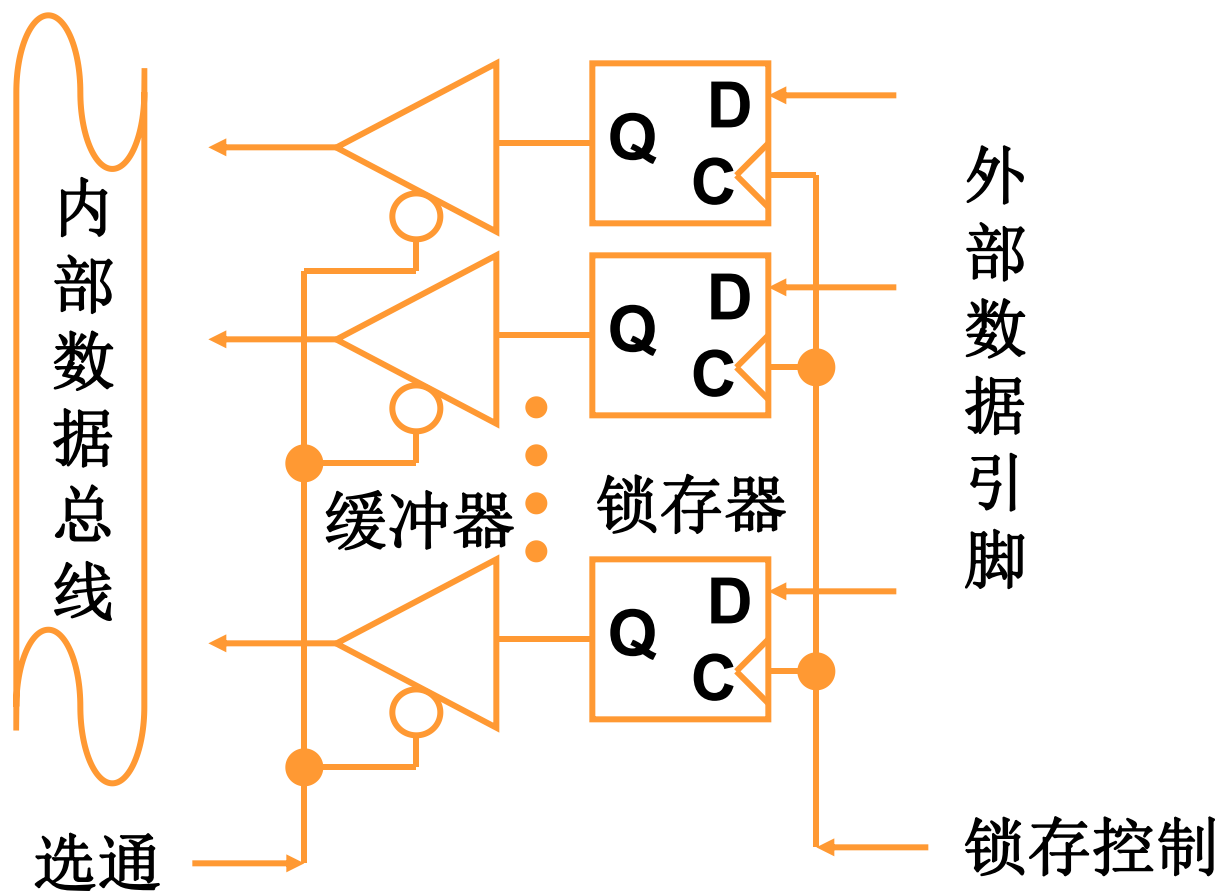
# 输出接口的锁存、缓冲环节




# 输入接口的缓冲环节



# 输入接口的锁存、缓冲环节

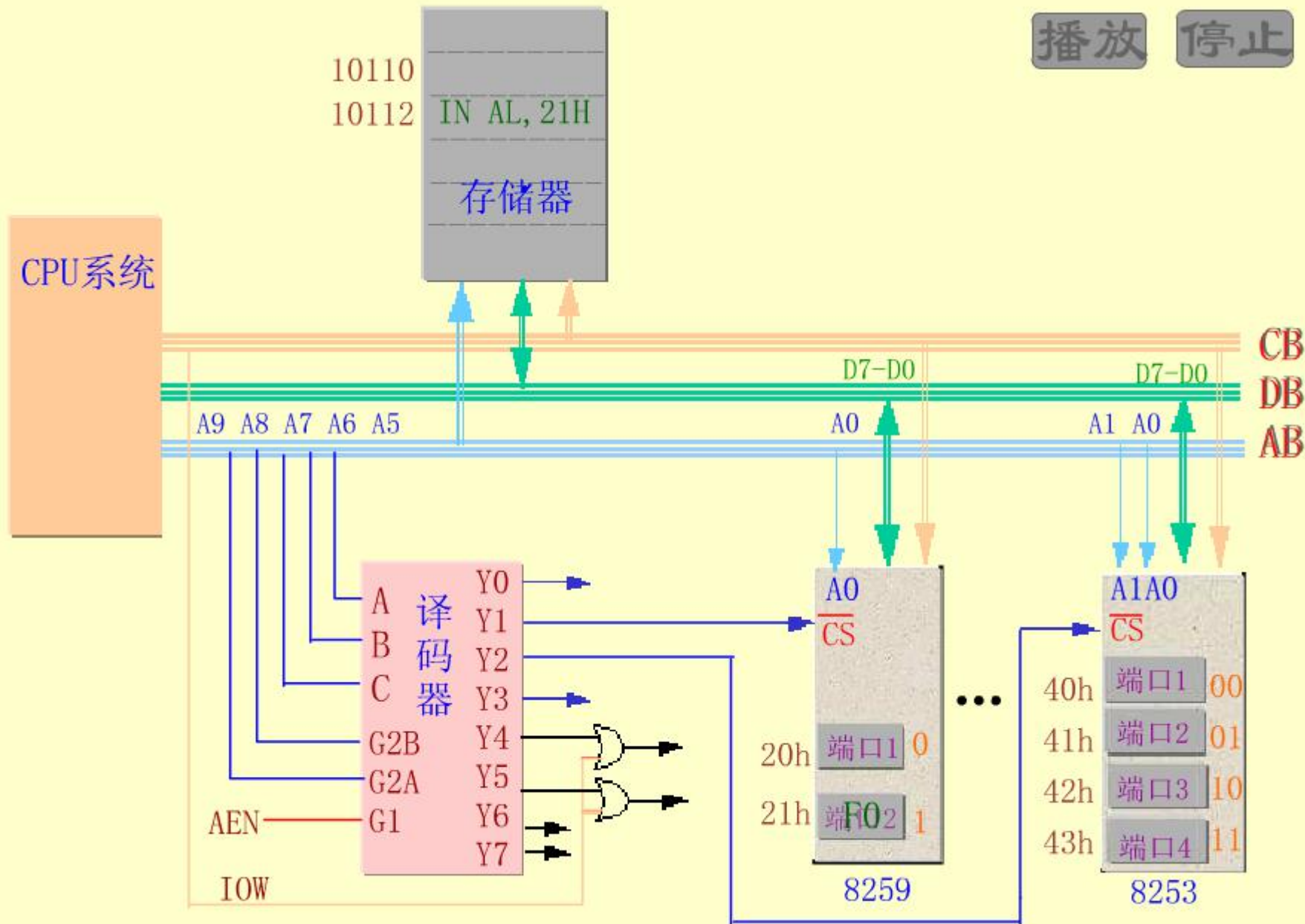


# 端口 (PORT)

- 
- ◇ 端口即接口电路中信息传输的门户，通常对应接口电路中的寄存器、三态缓冲器或此二者的组合，不同端口通过I/O地址进行区分。
  - ◇ 一个接口电路可以具有多个I/O端口，每个端口用来保存和交换不同的信息。
  - ◇ 按传输信息的类型不同，接口电路中的端口可以分为数据端口、状态端口和控制端口，分别用于传送数据、状态和控制信息。
  - ◇ 输入、输出端口可以分配同一个I/O地址。

一定要理解

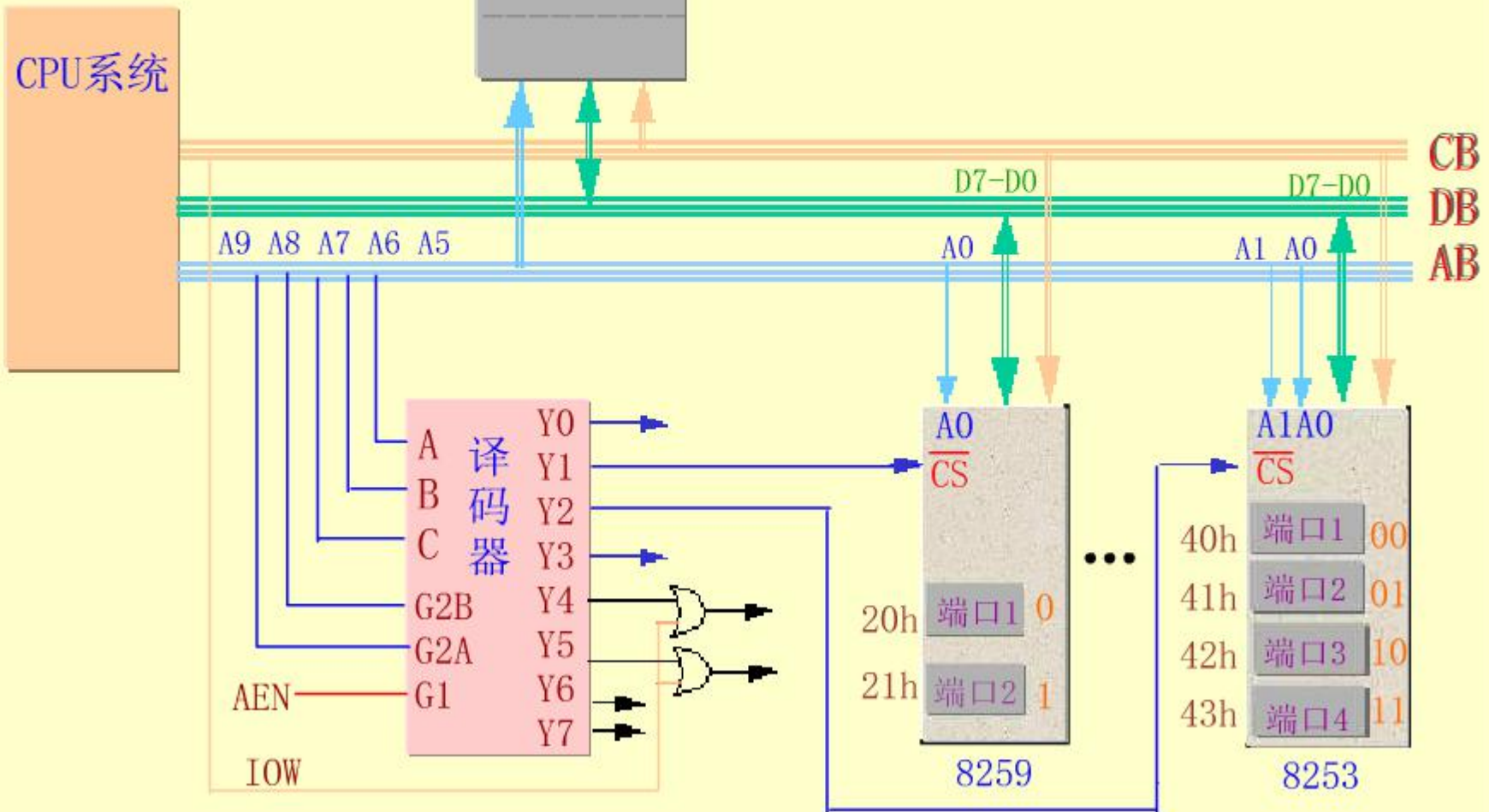




**IN AL, 21H**

1011E MOV AL, 90  
10120 OUT 43H, AL

播放 停止

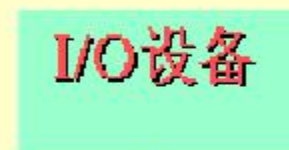
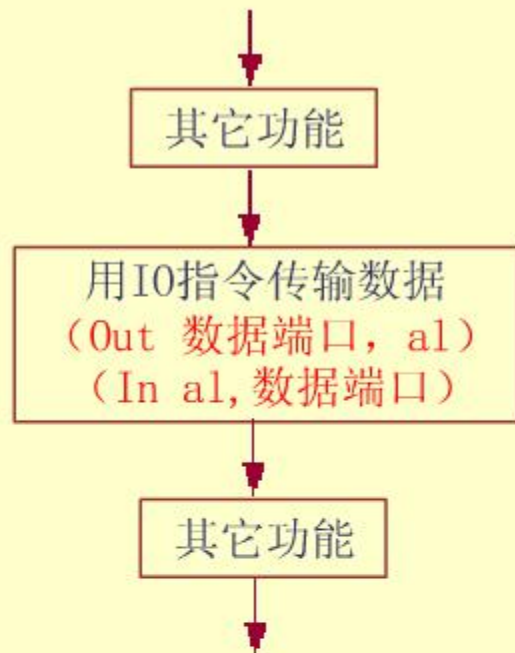


**OUT 43H, AL**

返回

播放

停止



无条件传送流程

返回



# 就绪 (Ready)



## ◇ 在输入场合

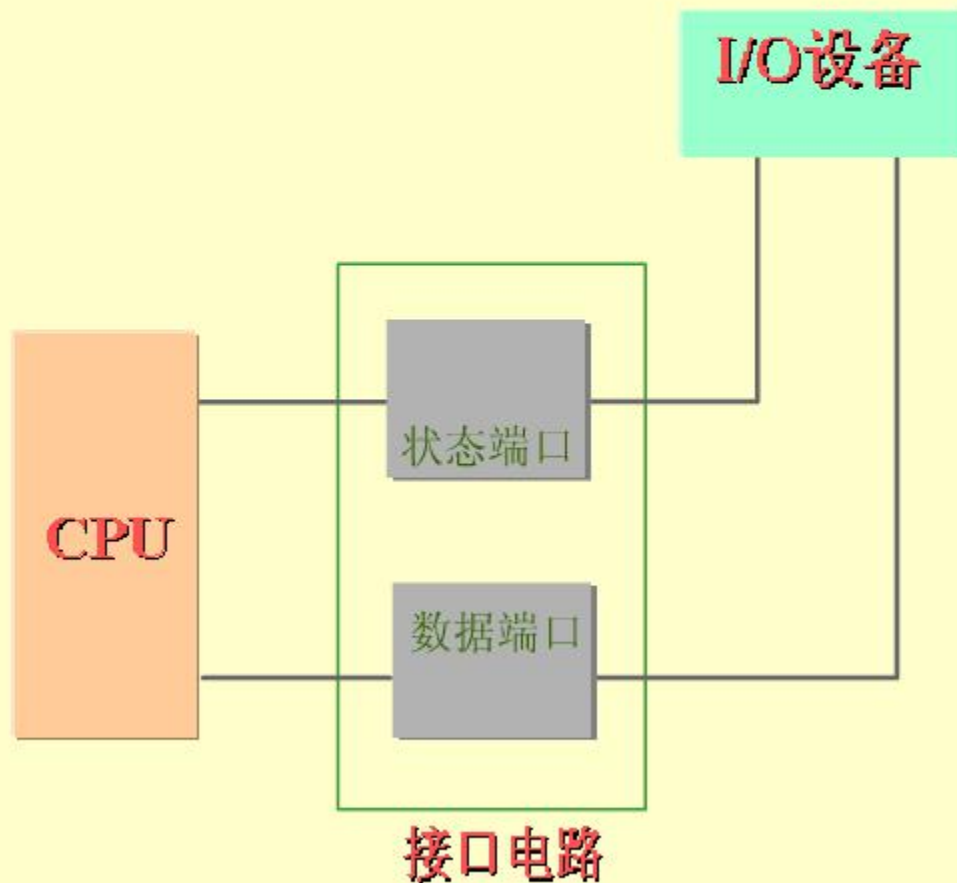
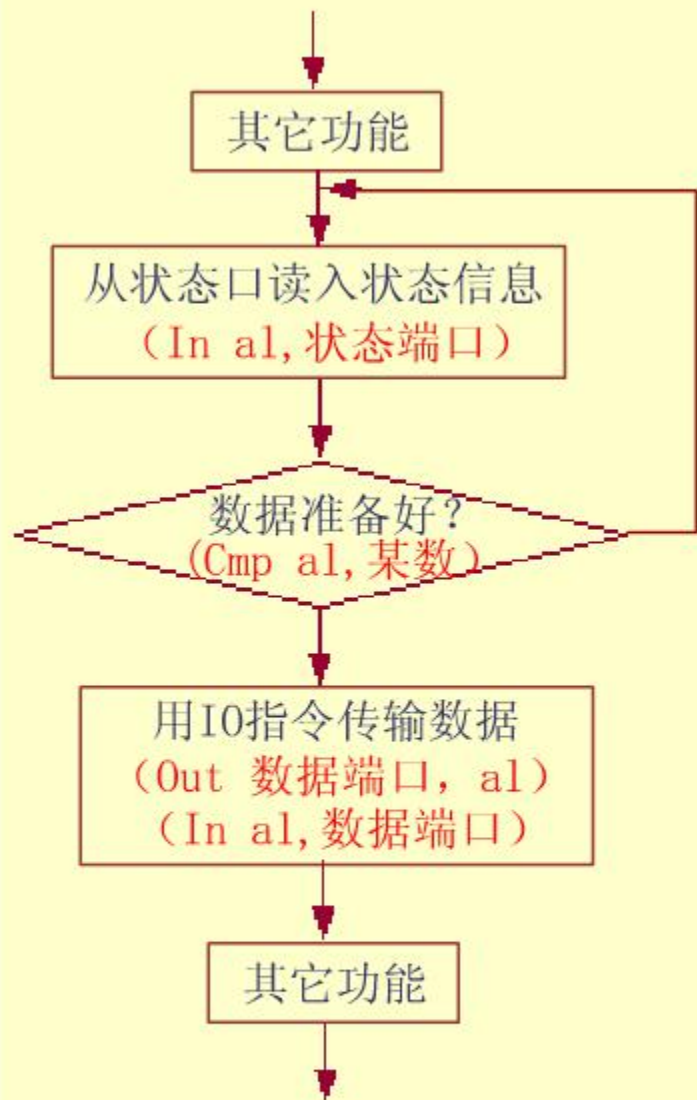
- ◆ “就绪”说明输入接口已准备好送往CPU的数据，正等着CPU来读取
- ◆ 该状态也可用接口中数据缓冲器已“满”来描述

## ◇ 在输出场合

- ◆ “就绪”说明输出接口已做好准备，等待接收CPU要输出的数据
- ◆ 该状态也可用接口数据缓冲器已“空”、或者用接口（外设）“闲”或不“忙（Busy）”来描述

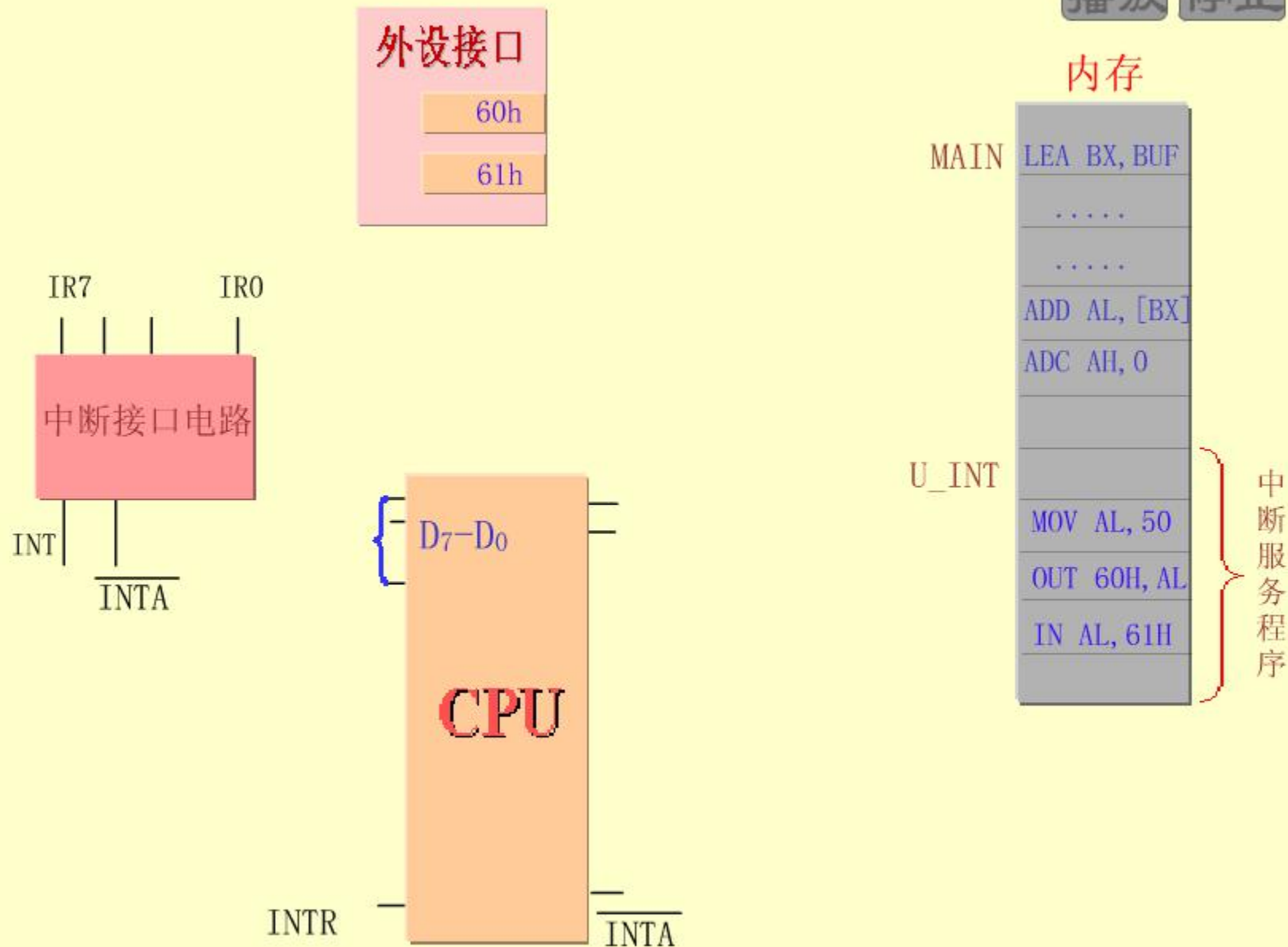


播放 停止

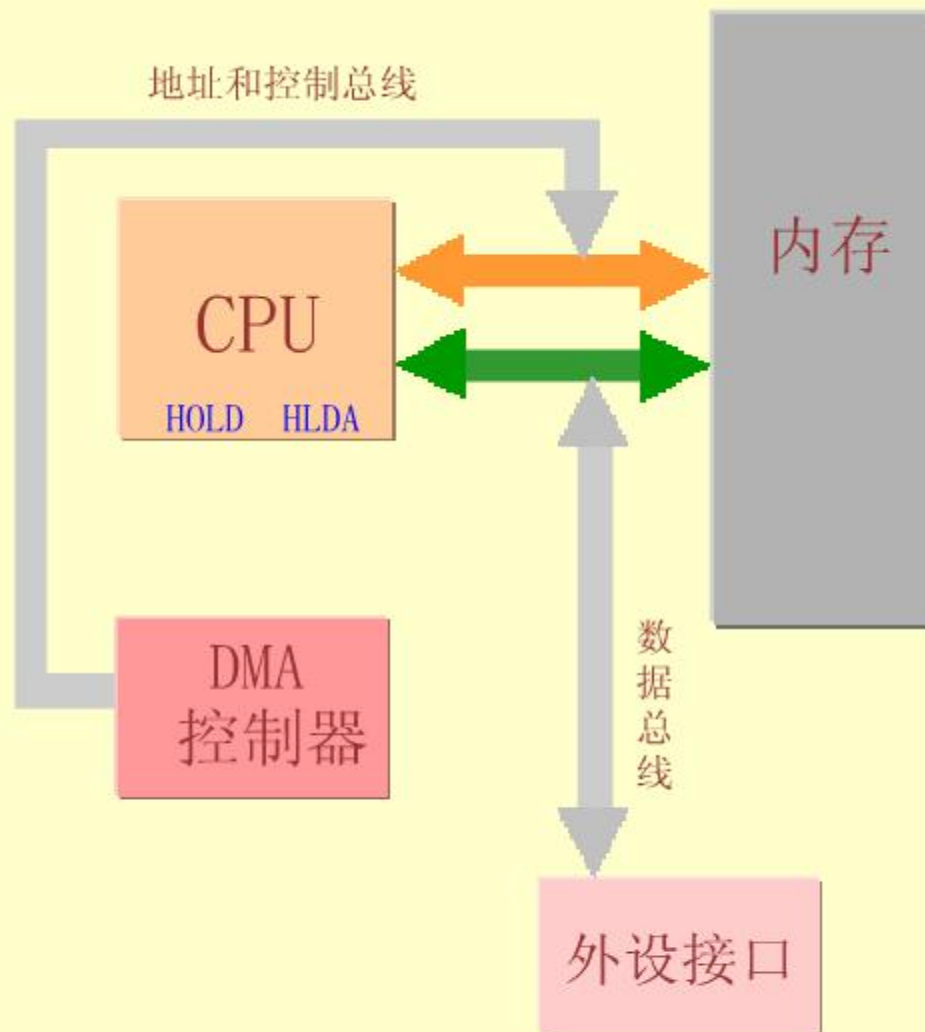


查询传送流程

返回



## 中断传送流程

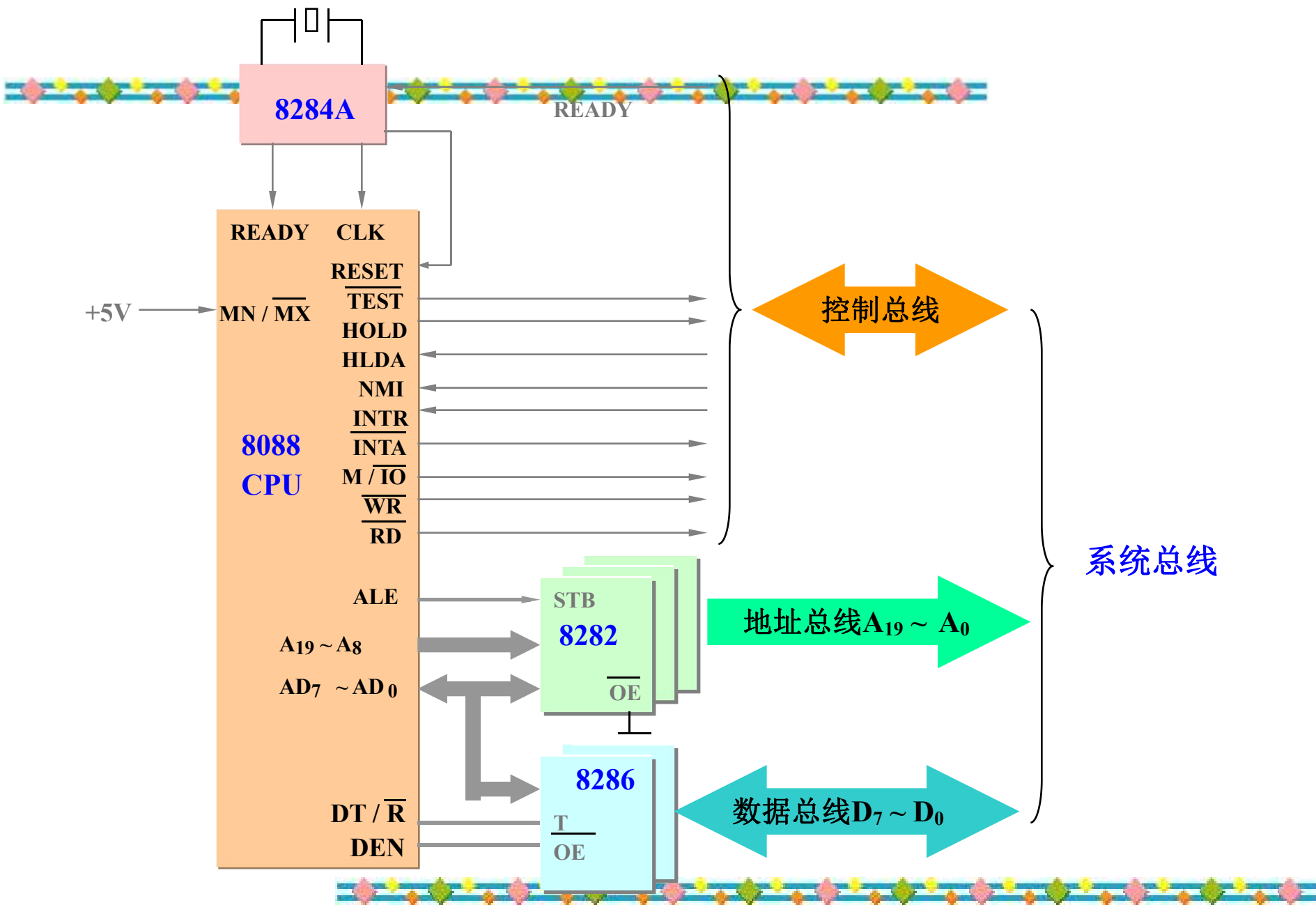


说明:

播放 停止

# DMA传送流程

返回



专用中断	00000H	类型 0 中断向量 (除法错)	偏移地址	→ IP
	00004H	类型 1 中断向量 (单步中断)	段基址	→ CS
	00008H	类型 2 中断向量 (NMI)		
	0000CH	类型 3 中断向量 (断点中断)		
	00010H	类型 4 中断向量 (溢出中断)		
系统备用中断	00014H	类型 5 中断向量		
	⋮	⋮		
	000C4H	类型 31H 中断向量		
用户中断	000C8H	类型 32H 中断向量		
	⋮	⋮		
	003FCH	类型 0FFH 中断向量		

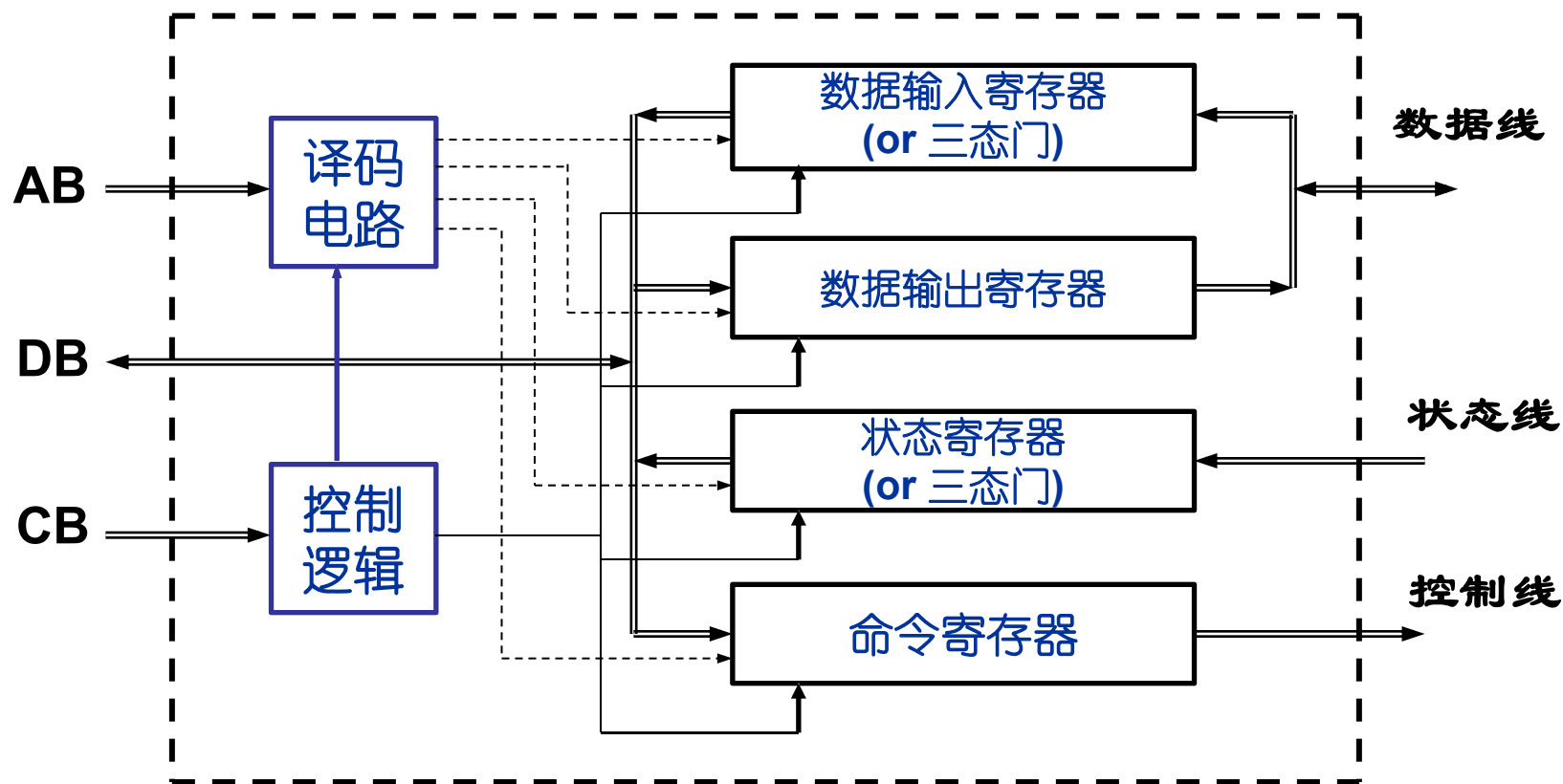
8086中断向量表

# 接口实例



返回

## 6.1.2 I/O接口的内部结构





## 第6章：I/O寻址方式

- ◇ 8088/8086的端口有64K个，无需分段，设计有两种寻址方式
  - ❖ 直接寻址：只用于寻址00H ~ FFH前256个端口，操作数i8表示端口号
  - ❖ 间接寻址：可用于寻址全部64K个端口，DX寄存器的值就是端口号
- ◇ 对端口号大于FFH的端口只能采用间接寻址方式

## 第6章：数据交换方式



- ◇ 如果输入输出一个字节，利用AL寄存器
- ◇ 如果输入输出一个字，利用AX寄存器
- ◇ 输入一个字，实际上是从连续两个端口输入两个字节，分别送AL（对应低地址端口）和AH（对应高地址端口）
- ◇ 输出一个字，实际上是将AL（对应低地址端口）和AH（对应高地址端口）两个字节的内容输出给连续两个端口

## 外设通过接口和系统的连接

