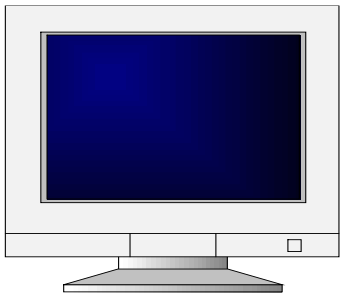
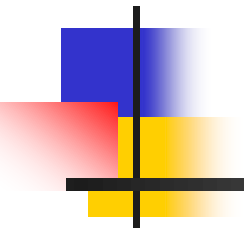
第七章

输入输出及中断技术



主要内容

- 基本概念
  - 输入输出系统
  - I/O接口和端口
  - 端口的编址方式
- 简单接口芯片及其应用
- 基本输入输出方法
- 中断的基本概念及工作过程



§6.1 输入输出系统

了解和掌握：

- 接口的基本功能
- 端口的概念
- 端口的编址方式
- I/O地址译码

一、为什么要引入I/O接口

- 微机和I/O设备的信息类型和格式可能不一样。
- 微机和I/O设备信号传输处理的速度可能不匹配。
- 不用接口， I/O直接接CPU，随着外设增加，会大大降低CPU的效率。
- I/O直接接CPU，会使外设硬件结构过于依赖CPU，对外设本身发展不利。

接口的功能

- 数据的缓冲与暂存
- 信号电平与类型的转换
- 增加信号的驱动能力
- 对外设进行监测、控制与管理，中断处理

接口的功能

数据缓冲功能：通过寄存器或锁存器实现。

存放数据的寄存器或锁存器称之为**数据口**。

接受和执行CPU命令功能：

存放CPU命令代码的寄存器称之为**命令口**，

存放执行状态信息的寄存器称之为**状态口**。

设备选择功能：CPU通过地址译码选择不同外设。

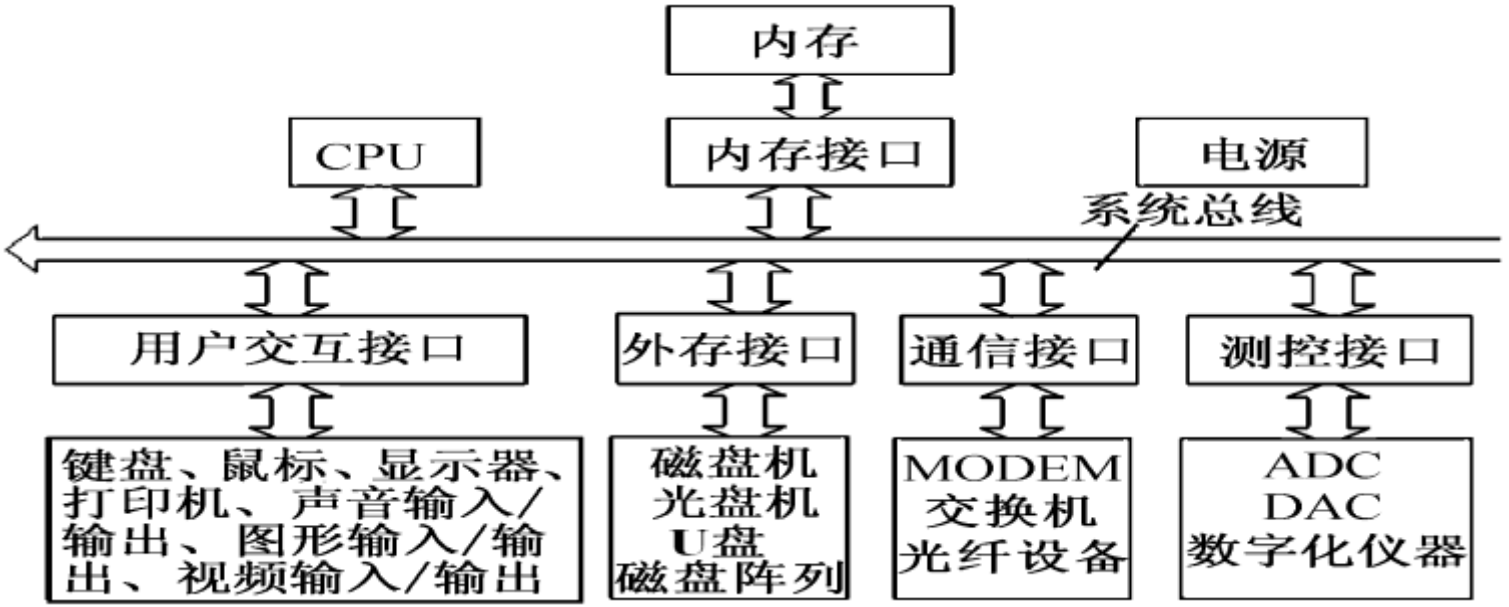
即CPU通过地址译码选择不同I/O接口和I/O接口中连接的不同的设备。

信号转换功能：协调总线信号与I/O设备信号。

转换包括信号的逻辑关系、时序配合和电平转换。

可编程功能：增加接口的灵活性和智能性。

接口的概念

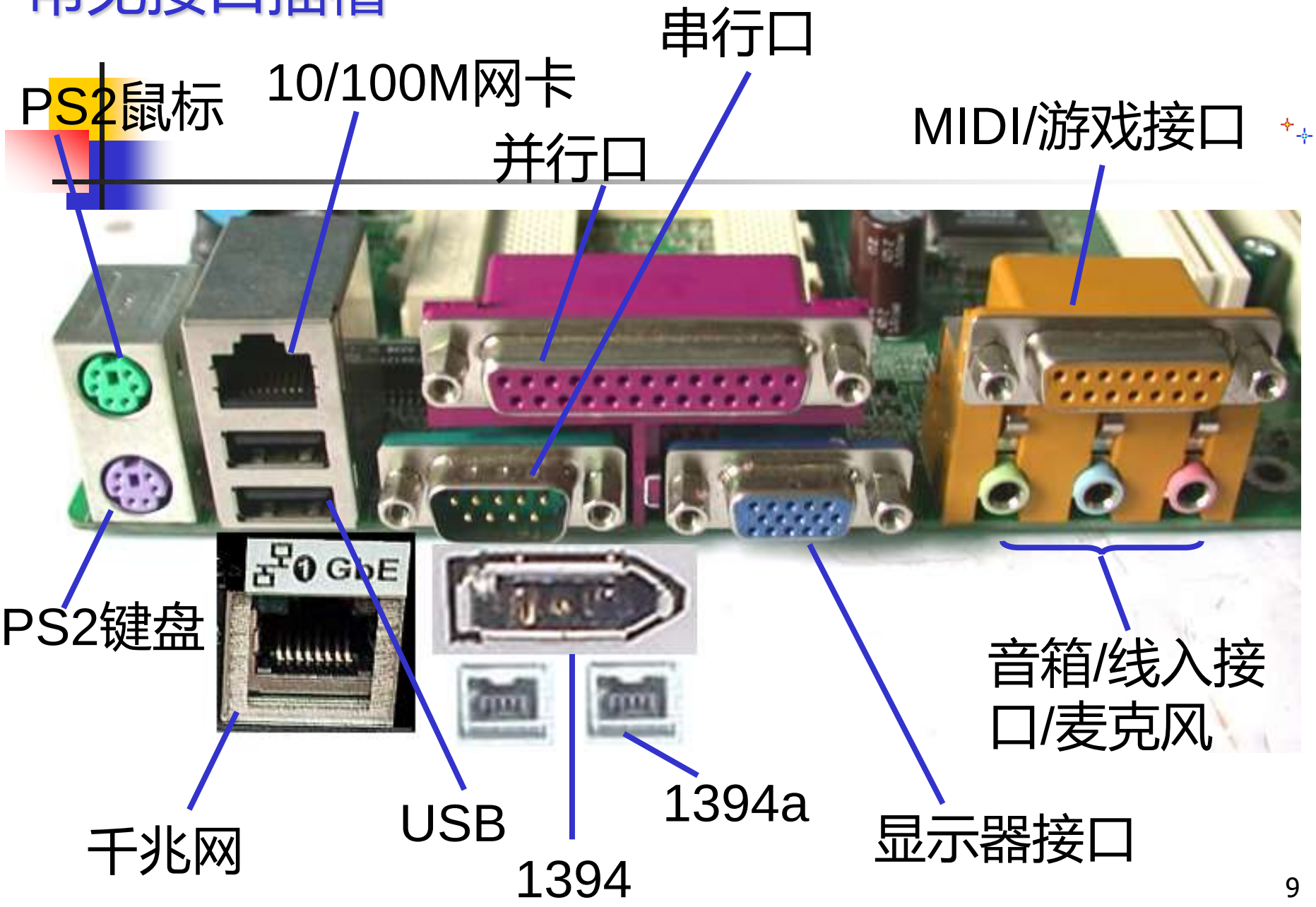


定义：接口是CPU与“外部世界”的连接电路，负责“中转”各种信息。

分类：存储器接口和I/O接口。

位置：介于系统总线与外部设备之间。

常见接口插槽


串行口
10/100M网卡
PS2鼠标
并行口
MIDI/游戏接口
PS2键盘
音箱/线入接口/麦克风
千兆网
USB
1394
1394a
显示器接口

二、I/O端口

端口
- 数据端口
- 状态端口
- 控制端口

- 电脑“端口”是英文port的义译，可以认为是计算机与外界通讯交流的出口。其中硬件领域的端口又称接口，如：USB端口、串行端口等。软件领域的端口一般指网络中面向连接服务和无连接服务的通信协议端口，是一种抽象的软件结构，包括一些数据结构和I/O（基本输入输出）缓冲区。

数据信息

- 数据信息是最基本的一种信息。它又包括3种：
- 数字量：通常为8位二进制数或ASCII代码。
- 模拟量：连续变化的物理量（如温度、压力、流量、位移、速度等
- 开关量：是一些“0”或“1”两个状态的量，用一位“0”或“1”二进制数表示。一台字长为8位的微机一次输入或输出可控制8个这类物理量。

状态信息

- 状态信息是反映外设当前所处工作状态的信息，以作为CPU与外设间可靠交换数据的条件。
- 状态信息在输入与输出时的作用：
  - 当输入时，它告知CPU：有关输入设备的数据是否准备好（Ready=1?）；
  - 输出时，它告知CPU：输出设备是否空闲（Busy=0?）。CPU是通过接口电路来掌握输入输出设备的状态，以决定可否输入或输出数据。

控制信息

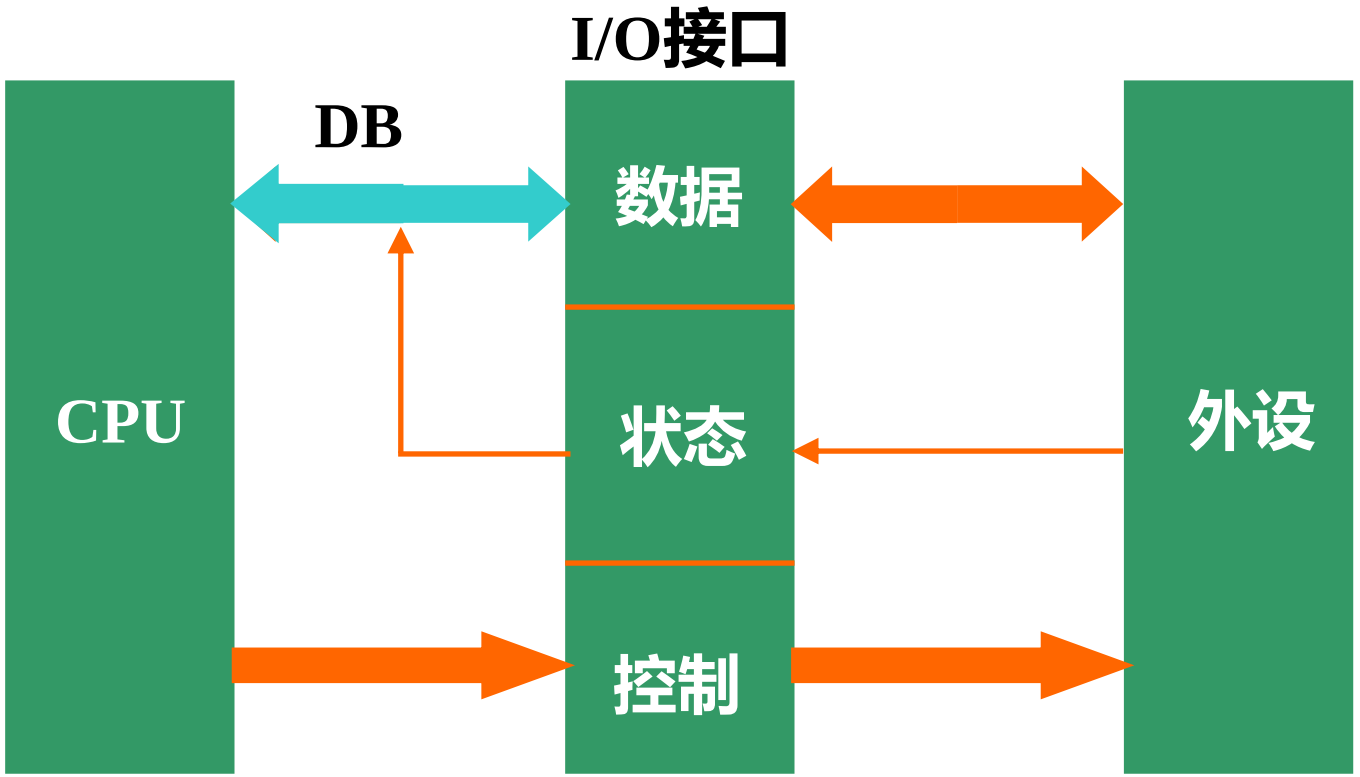
- 控制信息用于控制外设的启动或停止。
- CPU与外设传送的信息存放在哪里？

存储器或寄存器中，引入端口的概念

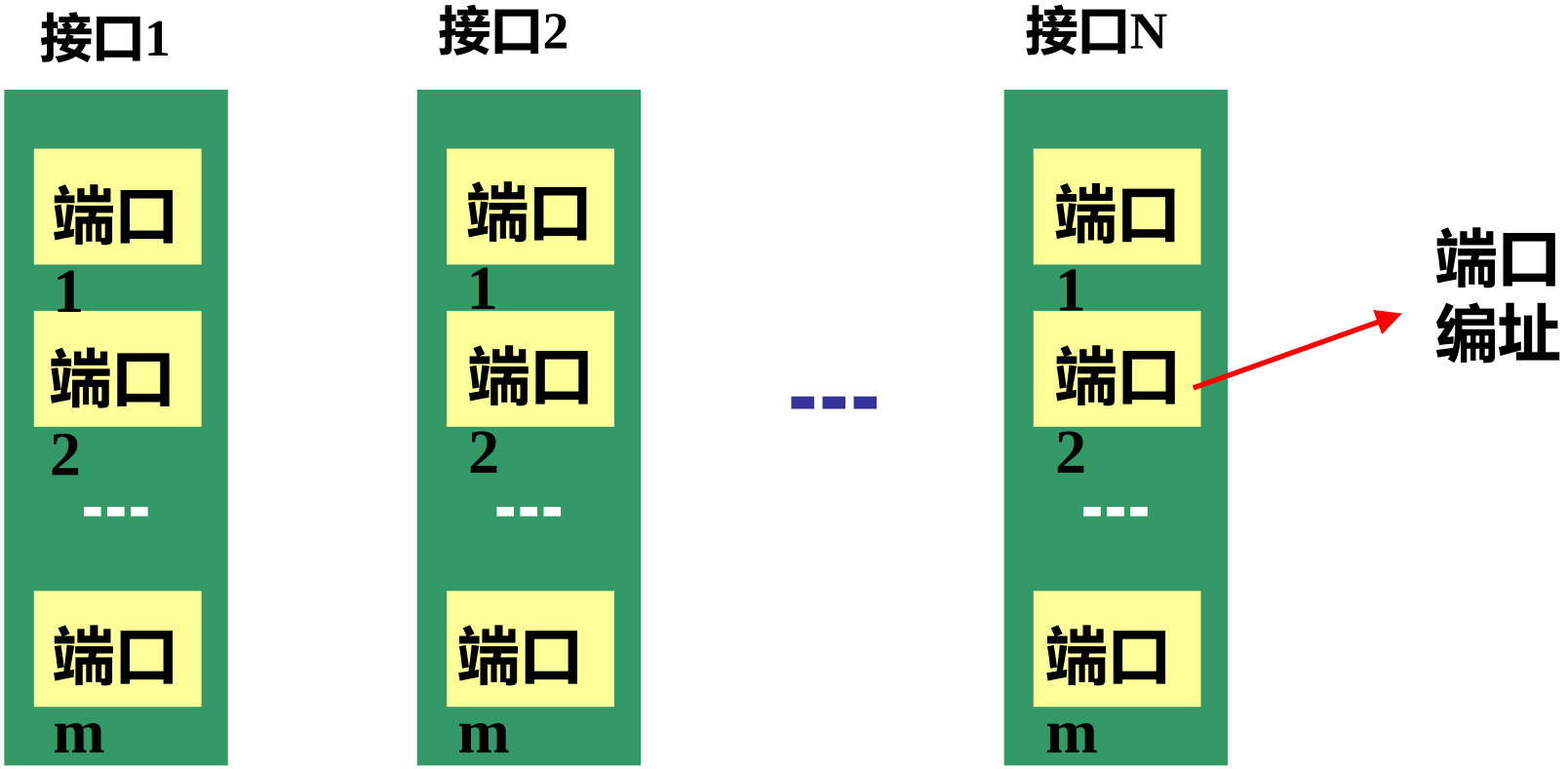
- CPU与外设传送的控制信息与状态信息是不同的性质的信息，如何将它们区分开？

使用不同的端口，所以接口中有若干的端口，每个端口给予不同的地址

I/O端口



接口和端口



端口地址=芯片地址（高位地址）+片内地址

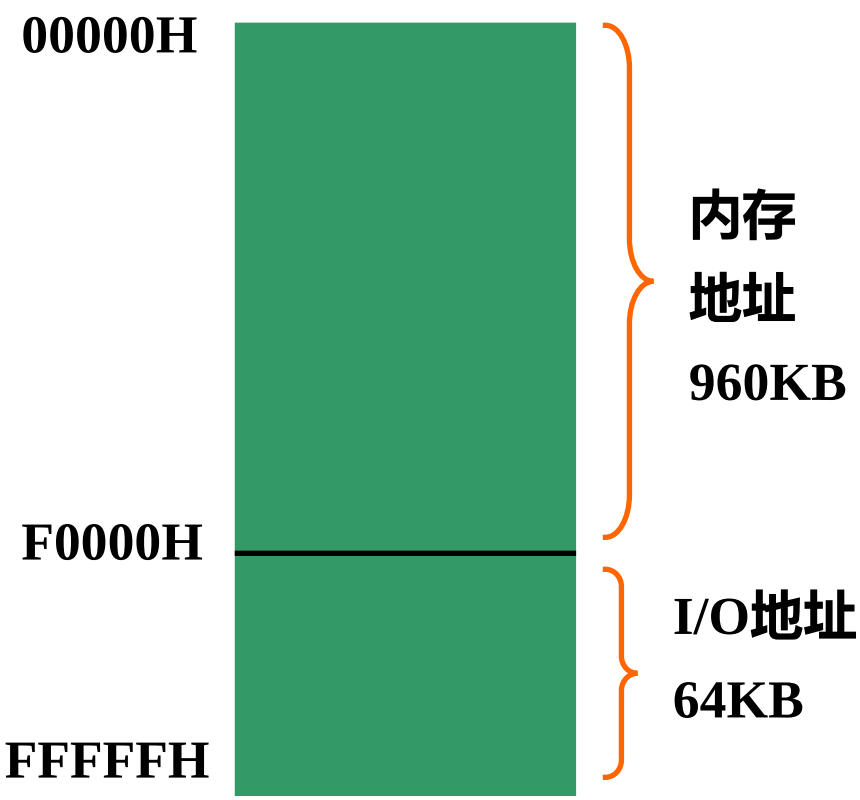
三、I/O端口的编址方式

- 8086/8088的寻址能力：
  - 内存：
    - 1MB
  - 端口：
    - 64K个
- 编址方式：
  - 与内存统一编址
  - 独立编址

端口与内存的统一编址

特点：

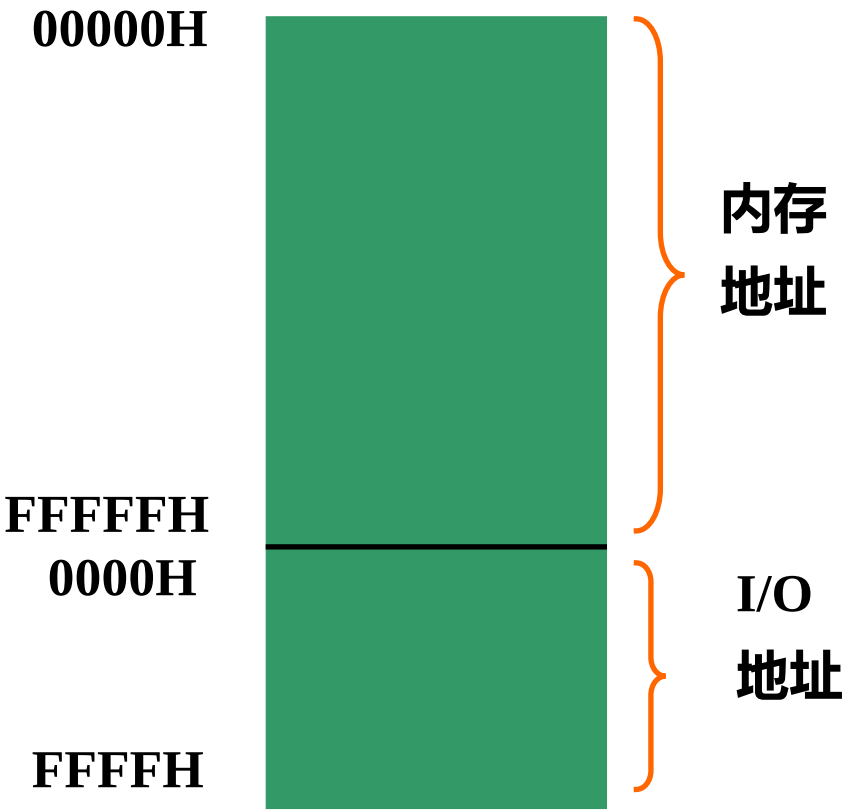
- 指令及控制信号统一；
- 内存地址资源减少



端口的独立编址

特点：

- 内存地址资源充分利用
- 能够应用于端口的指令较少



端口的寻址

- **8088/8086寻址端口数：**
  - **64K个**
- **寻址端口的信号：**
  - $\overline{\text{IOR}}$、$\overline{\text{IOW}}$
  - **A15 ～ A0**

直接寻址

在输入输出指令中直接写上8位I/O端口地址号。

例：IN　AL，50H ；50H端口的字节读入AL

例：OUT　80H，AX

；AX中的内容输出到80H、81H两个端口

间接寻址

16/8位I/O端口地址号存入DX，在输入输出指令中写DX，表示从其指示的I/O端口中输入输出数据。

例：MOV DX,0050H
IN AX, DX ；从DX和DX+1所指的2个端口输入1个字

例：MOV DX,FF02H
OUT DX, AL ；AL中的字节输出到DX所指端口中

注意：16位I/O端口地址必须使用DX寄存器间接寻址

输入输出指令

指令格式： IN　AL/AX/EAX, imm8/DX

OUT　imm8/DX, AL/AX/EAX

例：输入字数据

```
in al,21h ; 直接寻址，字节数据输入
mov ah,al
in al,20h
```

```
in ax,20h ; 直接寻址，字数据输入
```

```
mov dx,20h ; 间接寻址，字数据输入
in ax,dx
```

输入指令执行过程

- 三段功能相同；
- 字数据传送实际上实现了连续的两个端口地址的字节数据传送。

8088/8086的I/O端口编址

- 采用I/O独立编址方式(地址线与存储器共用)
- 地址线上的地址信号用IO/$\overline{M}$来区分
- I/O操作只使用20根地址线中的16根：A_{15}～A_0
- 可寻址的I/O端口数为64K(65536)个
- I/O地址范围为0～FFFFH
- IBM PC只使用了1024个I/O地址(0～3FFH)

四、I/O地址的译码

目的：

- 确定端口的地址

参加译码的信号：

- $\overline{IOR}$，$\overline{IOW}$，高位地址信号
- OUT指令将使总线的$\overline{IOW}$信号有效
- IN指令将使总线的$\overline{IOR}$信号有效

I/O译码的地址信号

- 当接口只有一个端口时，16位地址线一般应全部参与译码，译码输出直接选择该端口；
- 当接口具有多个端口时，则16位地址线的高位参与译码（决定接口的基地址），而低位则用于确定要访问哪一个端口。

I/O地址译码例

- 某外设接口有4个端口，地址为2F0H——2F3H，由A_{15}～A_2译码得到，而A_1、A_0用来区分接口中的4个端口。试画该接口与系统的连接图。

I/O地址译码例

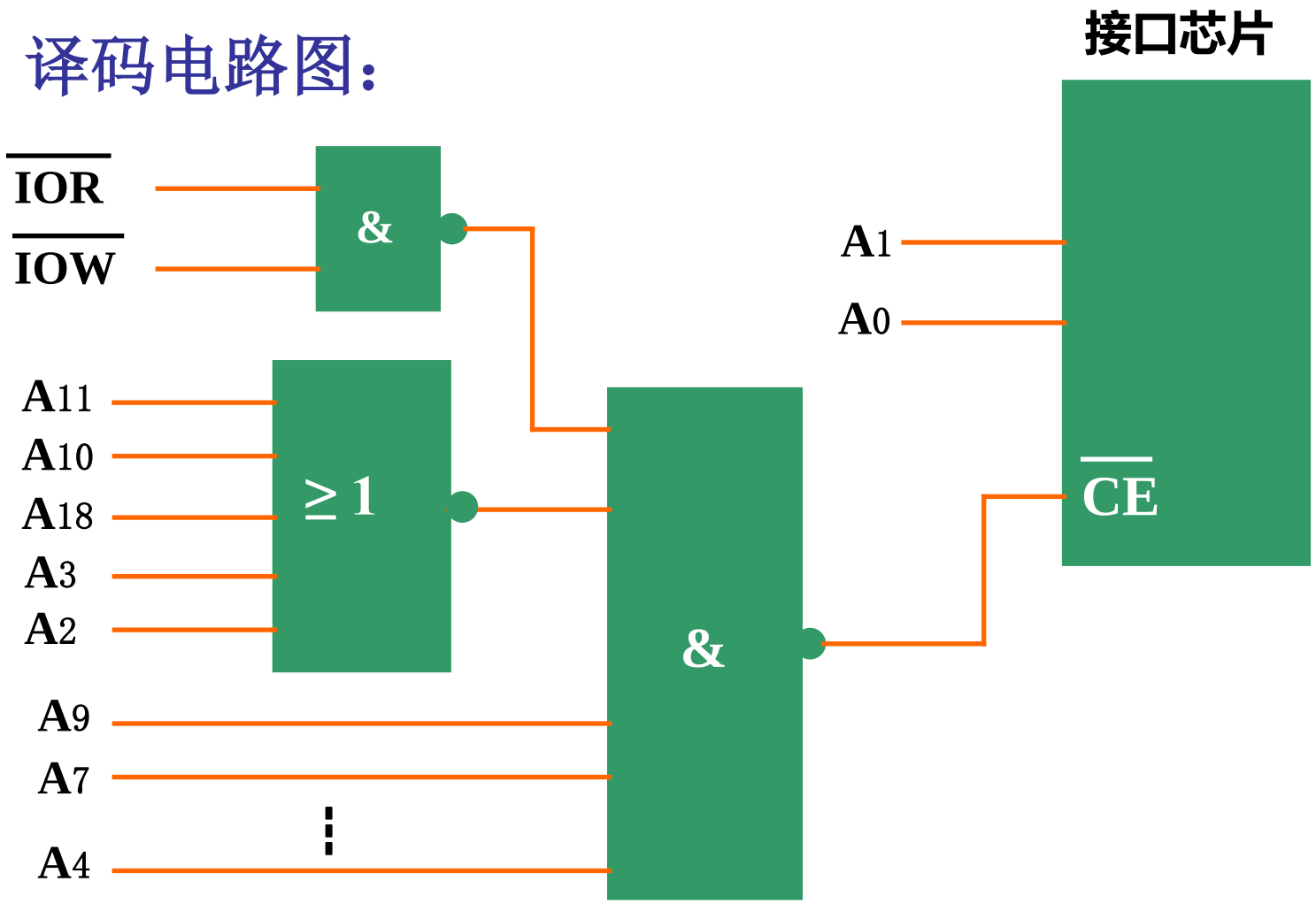
- 地址范围：

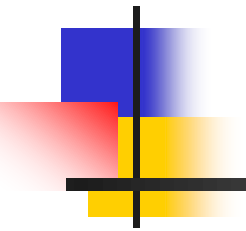
A11

- × × × × 0 0 1 0 1 1 1 1 0 0 0 0
- × × × × 0 0 1 0 1 1 1 1 0 0 1 1

I/O地址译码例

- 译码电路图：



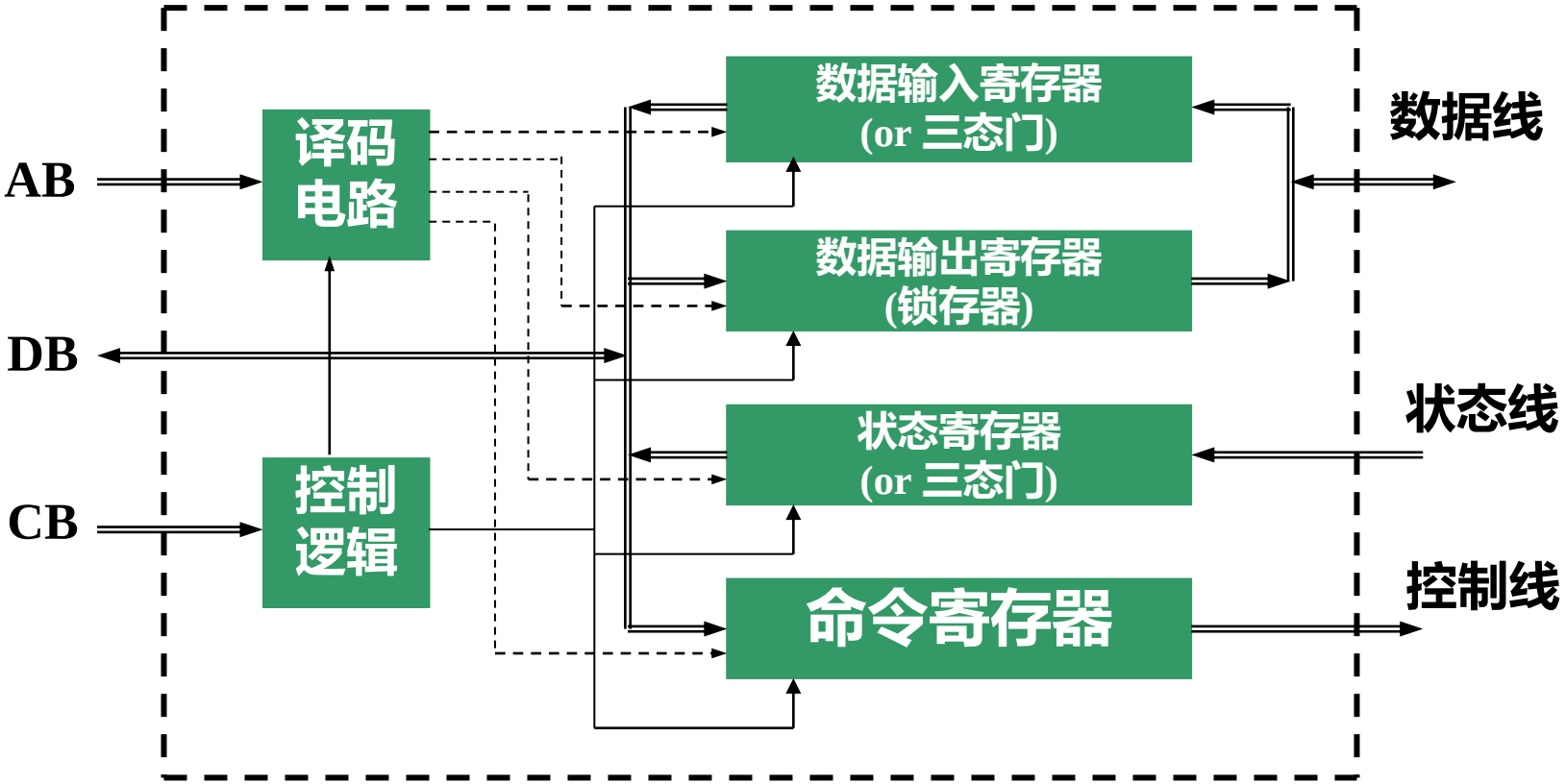


§6.2 简单接口电路

掌握：

- 接口电路的分类及特点；
- 两类简单接口芯片的应用

一、接口的基本构成


AB
DB
CB
译码
电路
控制
逻辑
数据输入寄存器
(or 三态门)
数据输出寄存器
(锁存器)
状态寄存器
(or 三态门)
命令寄存器
数据线
状态线
控制线

接口的基本构成

- 数据输入/输出寄存器

  暂存输入/输出的数据，又称数据输入端口、数据输出端口，合成数据端口。暂存输入/输出的数据。输入端口具有三态输出能力；输出常用锁存器实现；

- 命令寄存器

  又称命令端口或控制端口。存放CPU发出的控制命令(信息)，控制接口的工作方式及功能和控制外设按命令要求动作；

- 状态寄存器

  又称状态端口，存放外设或接口的状态信息，供CPU读；

- 译码电路

  CPU执行I/O指令时，从地址总线发送16位外部设备端口地址，其中高位地址线通过译码选中某接口芯片，低位地址线用来选择该接口芯片内不同端口；

- 控制逻辑

  产生接口内部控制信号和对外控制信号，以协调处理器和外设间的输入/输出操作 。

二、接口的类型及特点

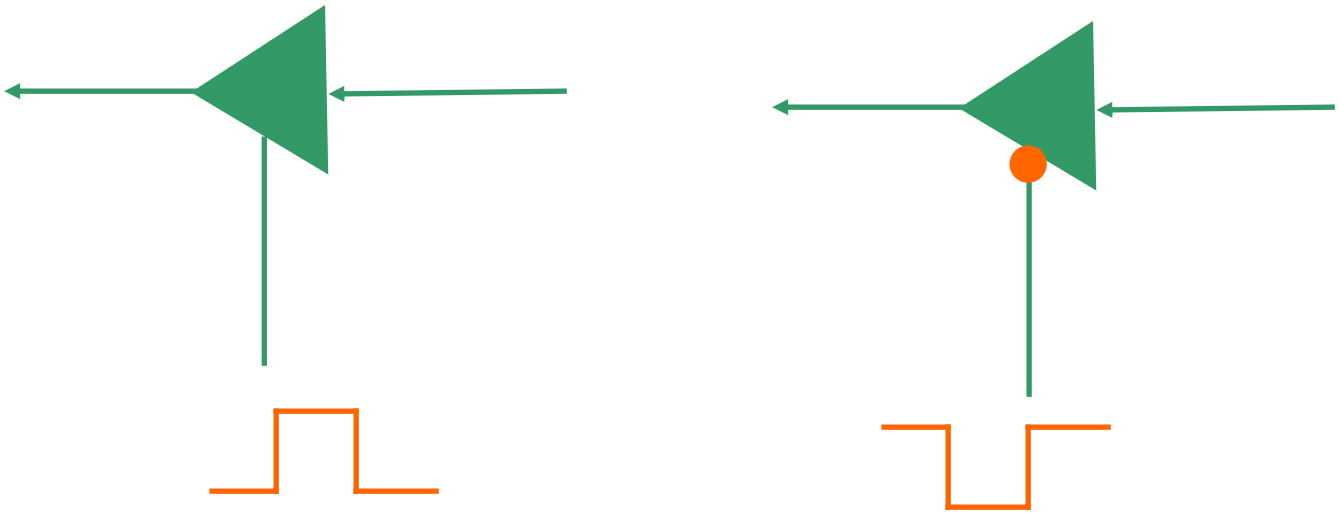
- 按传输信息的方向分类：
  - 输入接口
  - 输出接口
- 按传输信息的类型分类：
  - 数字接口
  - 模拟接口
- 按传输信息的方式分类：
  - 并行接口
  - 串行接口

接口特点

- 输入接口：
  - 要求对数据具有控制能力（缓冲）
  - 常用三态门实现
- 输出接口：
  - 要求对数据具有锁存能力
  - 常用锁存器实现

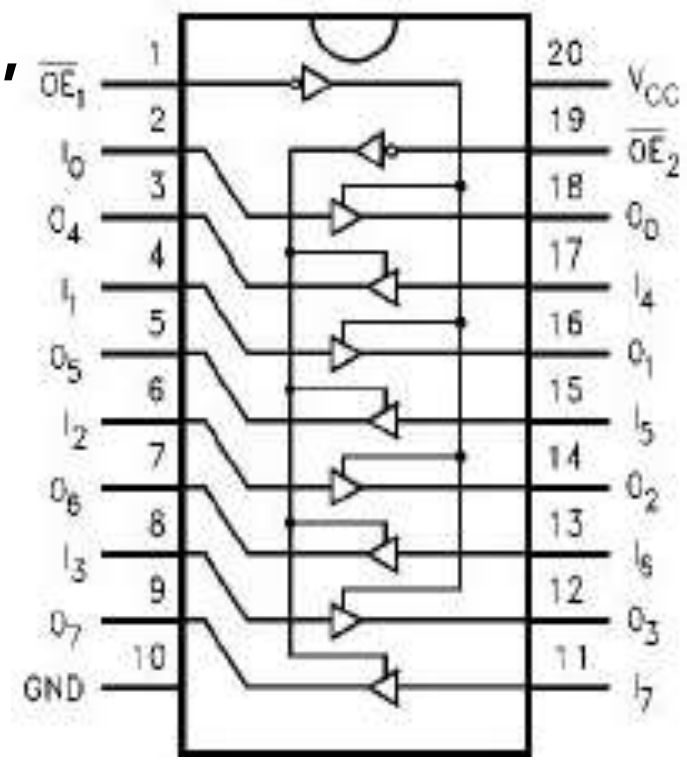
三、三态门接口

- 高电平、低电平、高阻态



74LS244

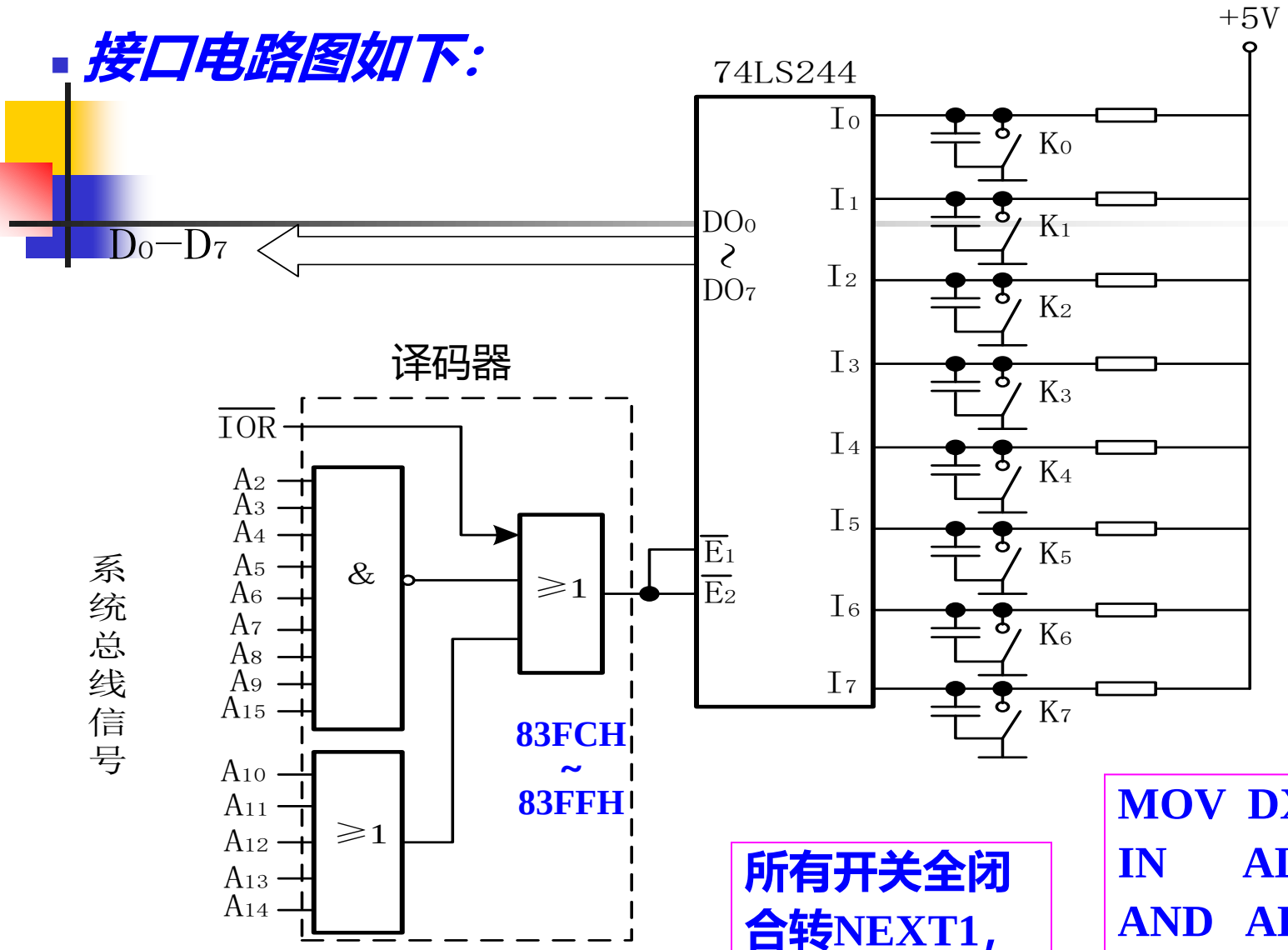
典型芯片74LS244，没有锁存功能的3态8位缓冲器（**含8个三态门**），3态是：**高电平、低电平、高阻态。当控制端有效时，三态门导通，输出等于输入；否则输出高阻态。**



➢ *典型应用：开关接口*

例：编写程序判断下图中的开关状态。如果所有开关都闭合，则程序转向标号NEXT1执行，否则转向标号为NEXT2执行。

- **接口电路图如下：**



所有开关全闭合转NEXT1，否则转NEXT2

。

```
MOV  DX,83FCH
IN   AL,DX
AND  AL,0FFH
JZ   NEXT1
JMP  NEXT2
```

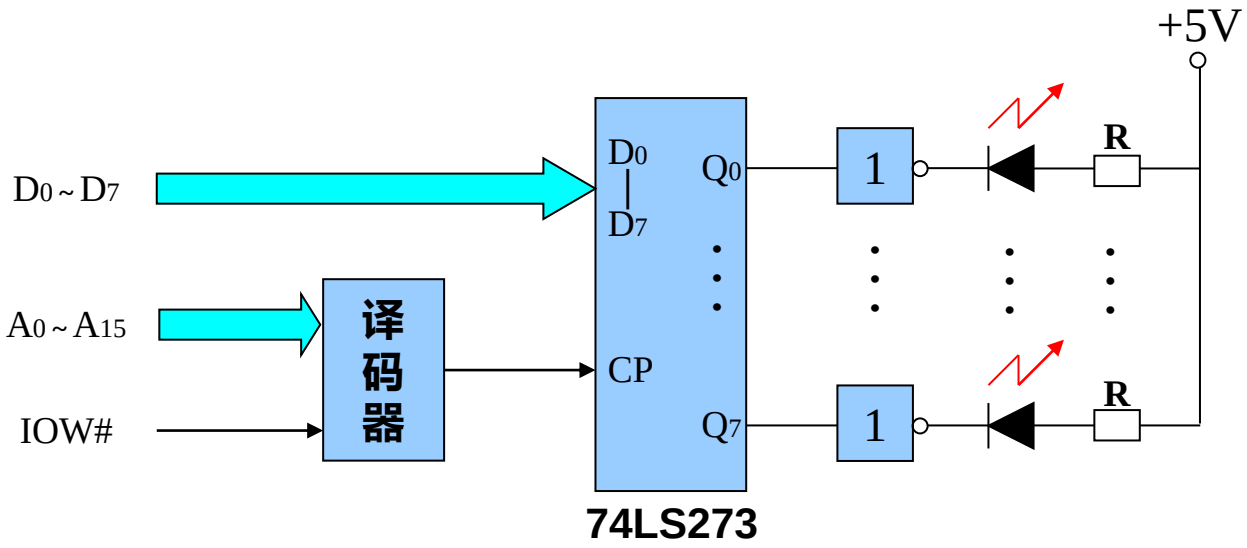
四、锁存器接口

- 通常由D触发器构成；
- 特点：
  - 具有对数据的锁存能力；
  - 不具备对数据的控制能力

- 74LS273(二态锁存）：是一种带清除功能的8D触发器， D_0～D_7为数据输入端，Q_0～Q_7为数据输出端，CP正脉冲（上升沿）触发将D_i端数据锁存到Q_i端，S低电平清除Q_i端。常用作8位地址锁存器或输出接口。

- ***74LS374(三态锁存器)：可作输入/输出缓冲器接口。74LS374带了三态门，它比74LS273多了一个输出允许端OE。只有当 OE=0时，74LS374的输出三态门才导通， OE=1时呈高阻态。***

锁存器芯片74LS273

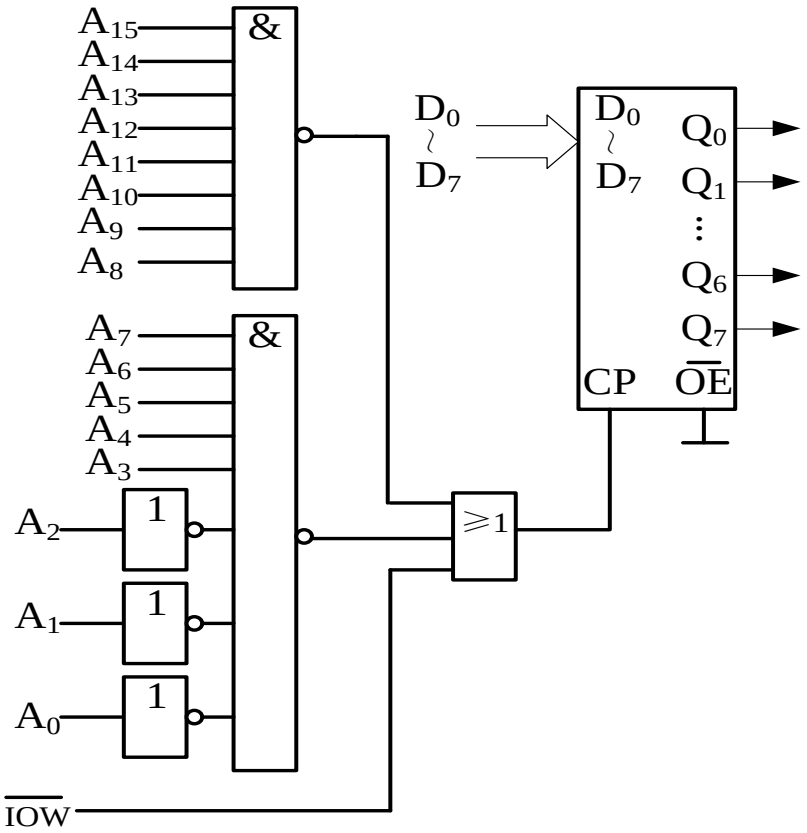


设该输出接口地址是0FFFFH，则下列程序段的功能?

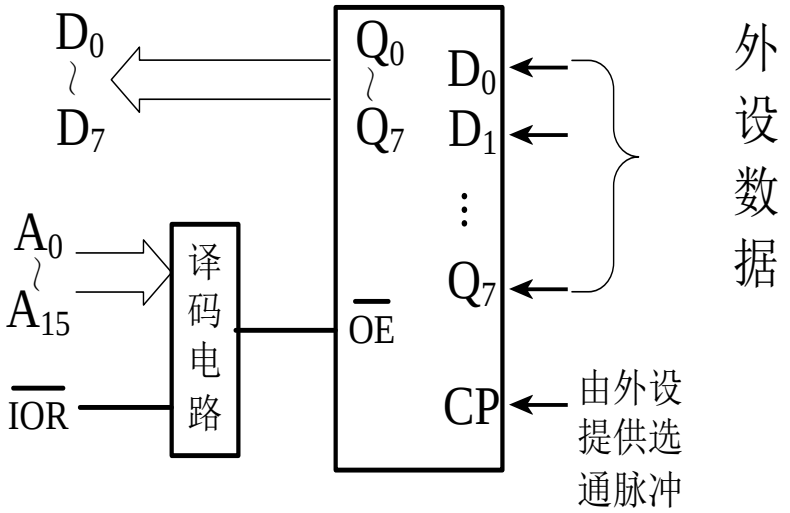
```
NOV   DX, 0FFFFH
MOV   AL, 01000001B
OUT   DX, AL
```

锁存器芯片74LS374

❖做输出端口:三态门一直处于导通状态。



❖做输入端口:CP正脉冲锁存输入数据，译码选中时$\overline{OE}$=0，CPU读取数据。

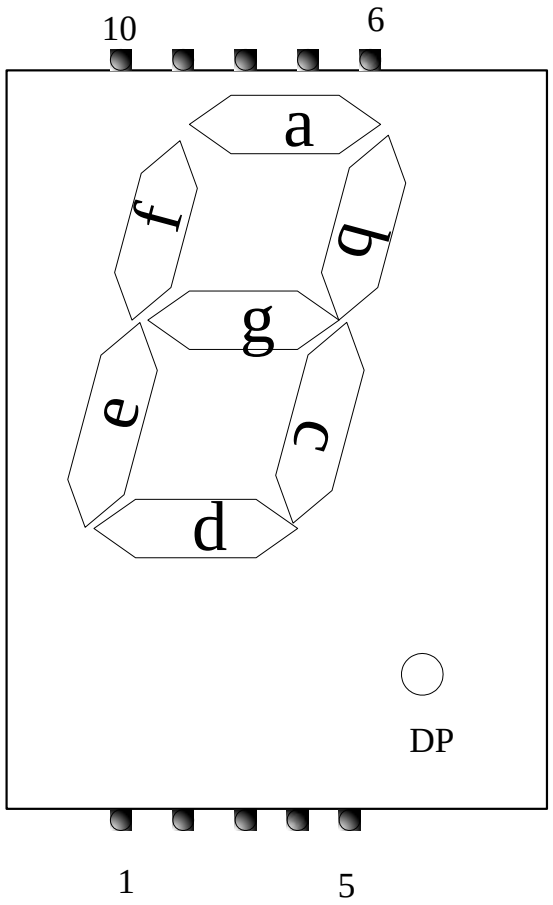


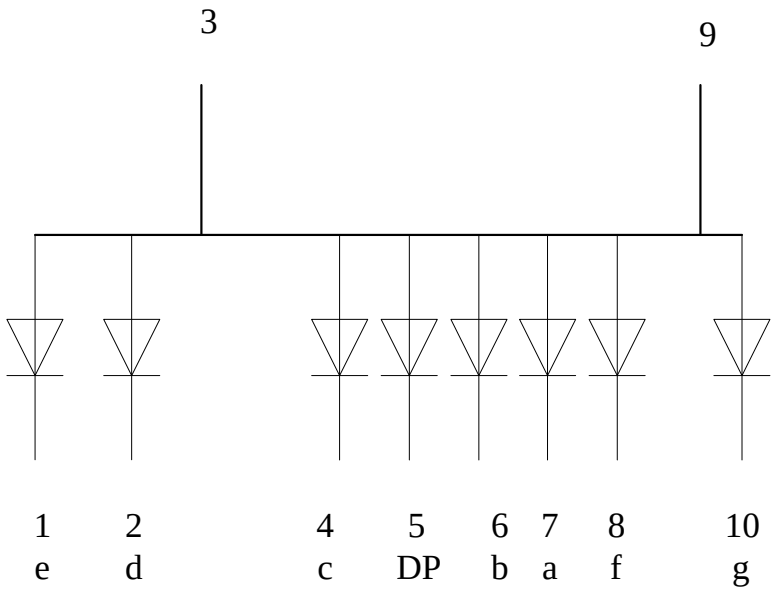
I/O接口综合应用例

利用74LS244和74LS273作为输入和输出接口，编写程序，**根据开关状态**在LED数码管显示十六进制数码。

- 根据开关状态在7段数码管上显示数字或符号
- 设输出接口的地址为F0H
- 设输入接口地址为F1H
- 当开关的状态分别为0000～1111时，在7段数码管上对应显示’0’～’F’

共阳极7段LED数码管结构(7段码表见下页)

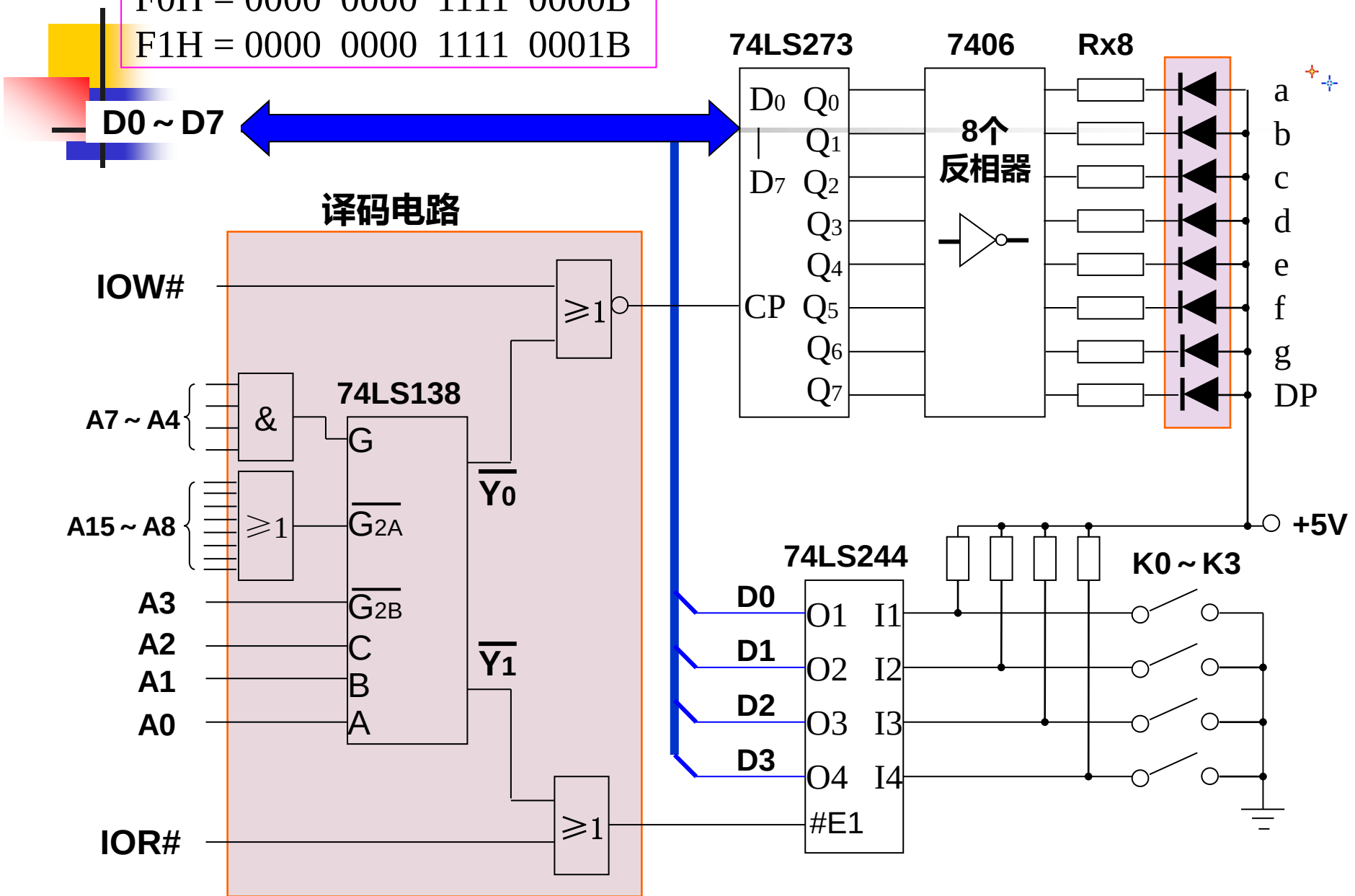




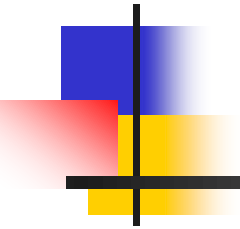
符号	形状	7段码 .gfedcba	符号	形状	7段码 .gfedcba
'0'		00111111	'8'		01111111
'1'		00000110	'9'		01100111
'2'		01011011	'A'		01110111
'3'		01001111	'B'		01111100
'4'		01100110	'C'		00111001
'5'		01101101	'D'		01011110
'6'		01111101	'E'		01111001
'7'		00000111	'F'		01110001

F0H = 0000 0000 1111 0000B

F1H = 0000 0000 1111 0001B



```
        ……
Seg7    DB      3FH,06H,5BH,4FH,66H,6DH,7DH,07H
        DB      7FH,67H,77H,7CH,39H,5EH,79H,71H
        ……
        LEA     BX, Seg7        ；取7段码表基地址
        MOV     AH, 0
GO:     MOV     DX, 0F1H        ；开关接口的地址为F1H
        IN      AL, DX          ；读入开关状态
        AND     AL, 0FH         ；保留低4位
        MOV     SI, AX          ；作为7段码表的表内位移量
        MOV     AL, [BX+SI]；取7段码
        MOV     DX, 0F0H        ；7段数码管接口的地址为F0H
        OUT     DX, AL
        JMP     GO
```



§6.3 基本输入/输出方法

基本输入/输出方法

- 无条件传送
- 查询式传送

  } **程序控制方式**

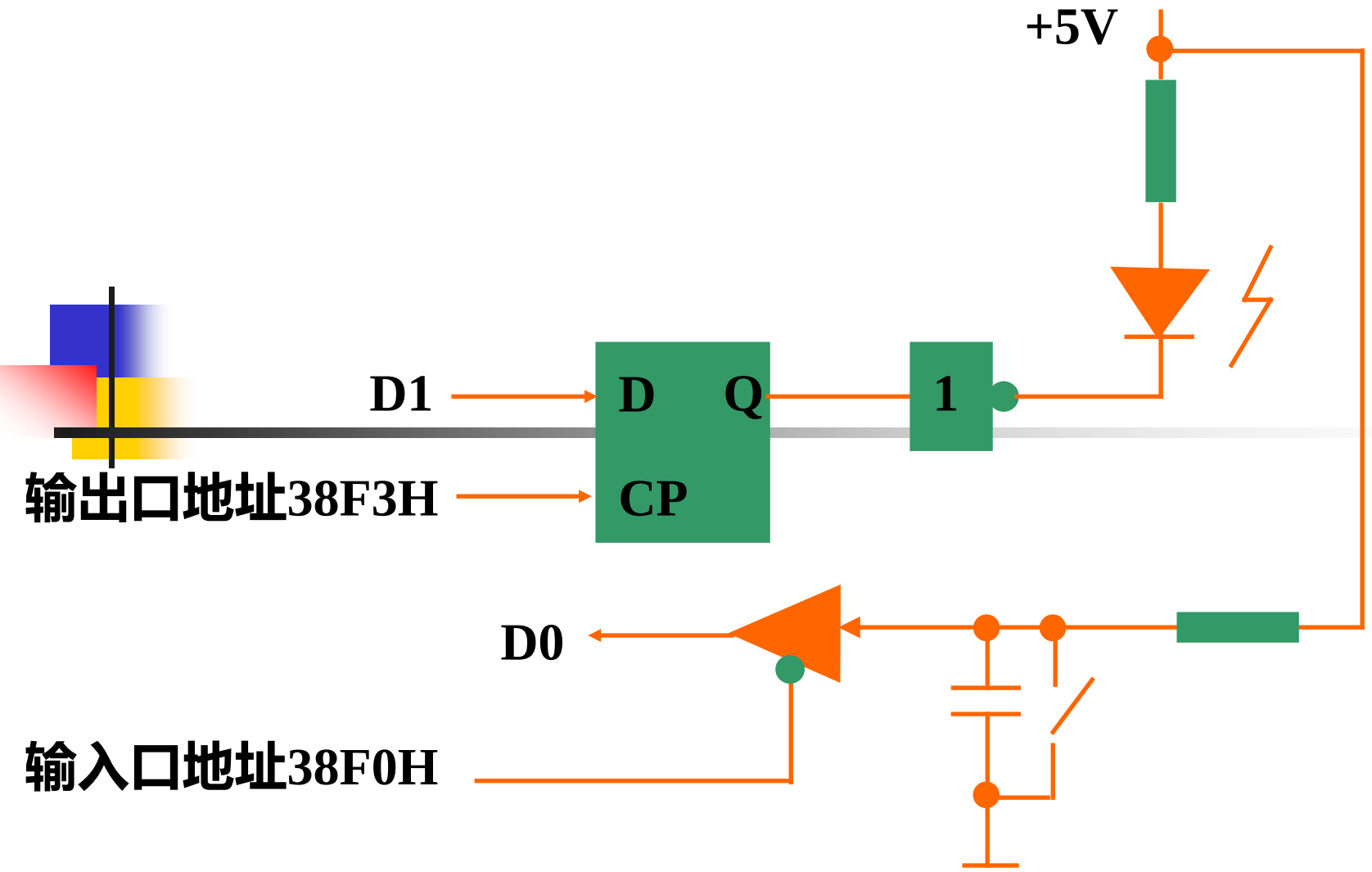
- 中断方式传送
- 直接存储器存取(DMA)

一、无条件传送

- 适用于总是处于准备好状态的外设，输入输出指令的执行就是数据传送过程；
- 以下外设可采用无条件传送方式：
  - *开关；*
  - *发光器件(发光二极管、7段数码管、灯泡等)；*
  - *继电器；*
  - *步进电机。*
- 优点：软件及接口硬件简单；
- 缺点：只适用于简单外设，适应范围较窄。

无条件传送例

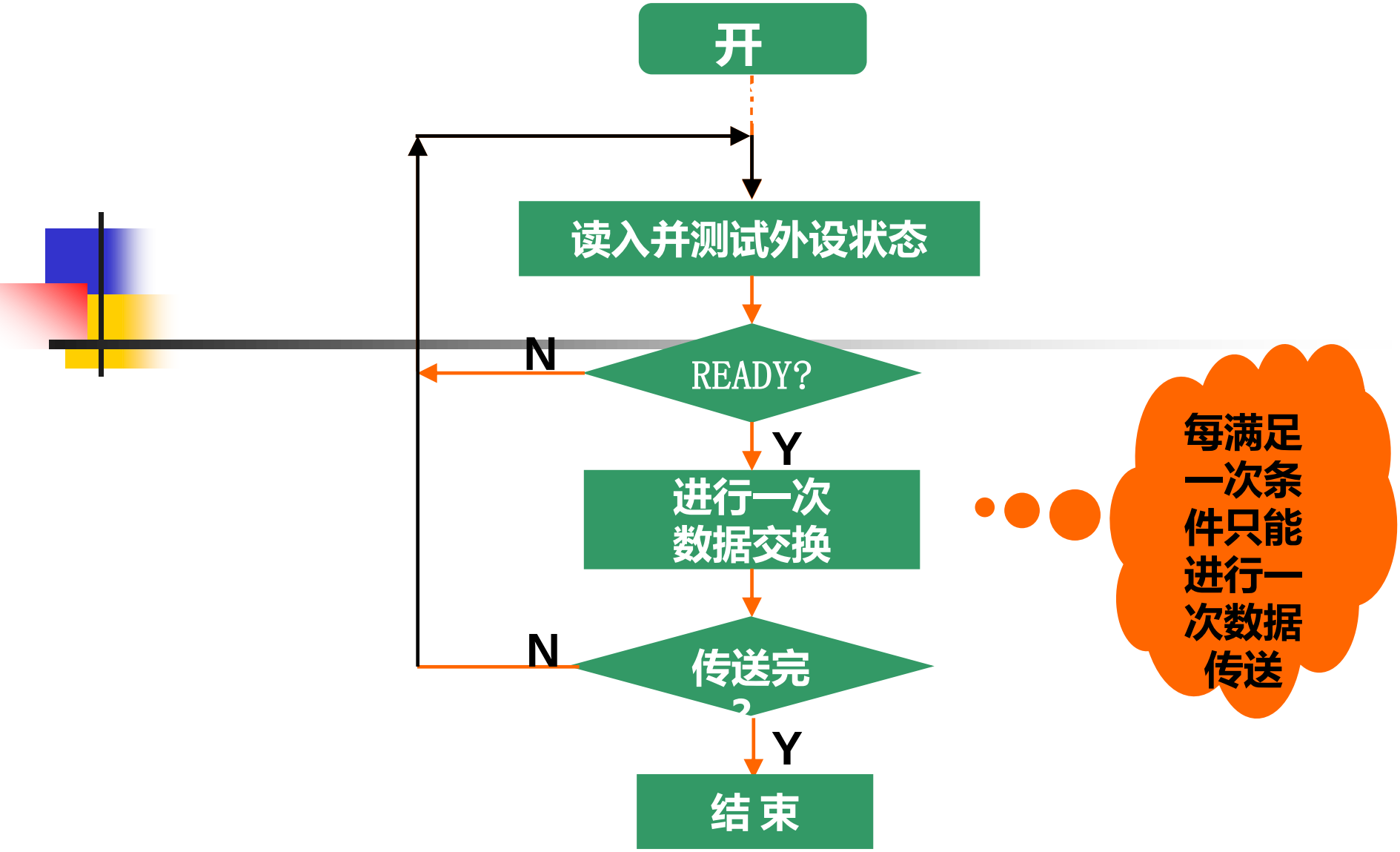
- 读取开关的状态；
- 当开关闭合时，输出编码使发光二极管亮。

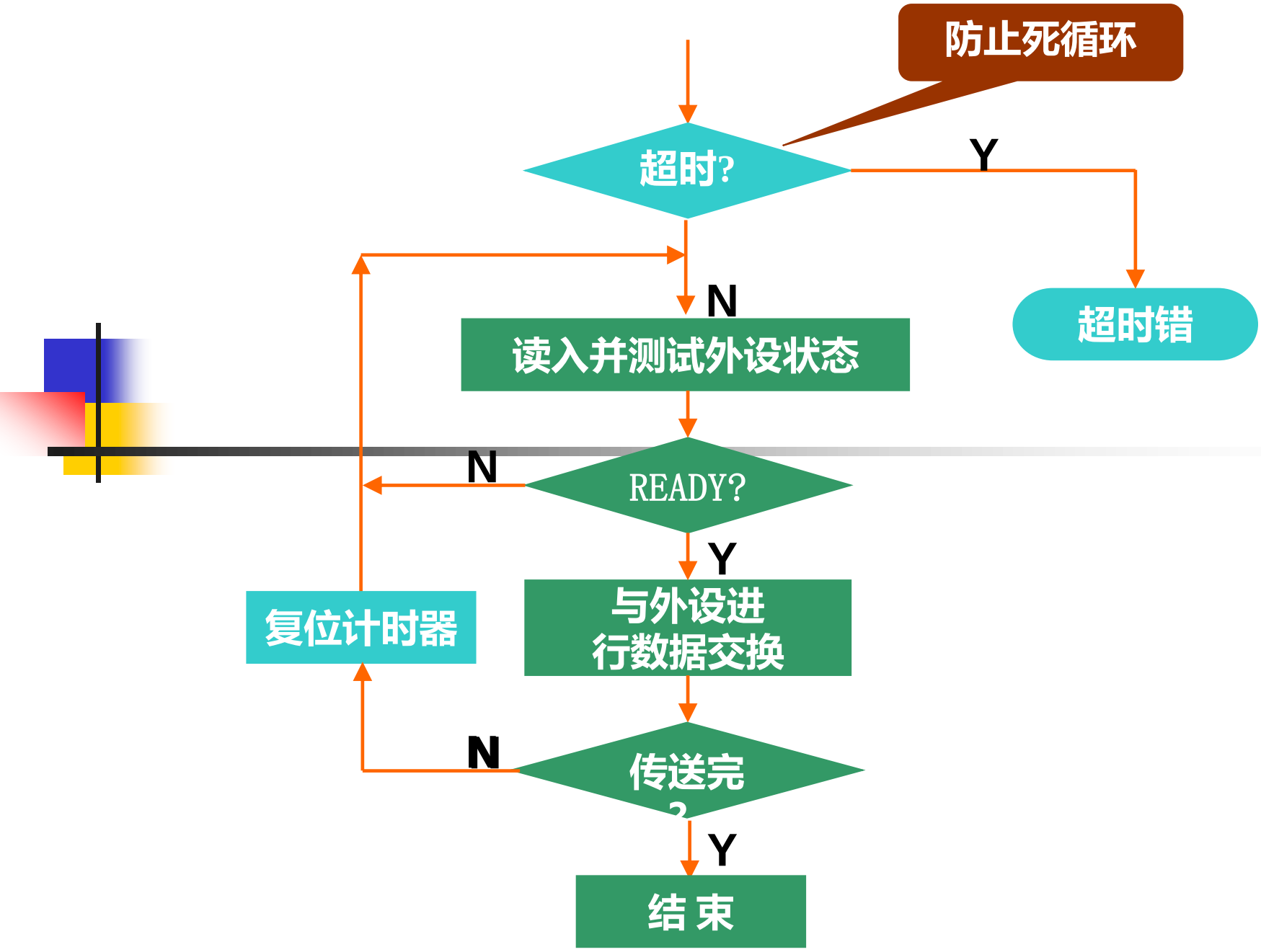

+5V
D1
D
Q
1
输出口地址38F3H
CP
D0
输入口地址38F0H

二、查询工作方式

- 仅当条件满足时才能进行数据传送；
- 每满足一次条件只能进行一次数据传送。
- 适用场合：
  - 外设并不总是准备好
  - 对传送速率和效率要求不高
- 工作条件：
  - 外设应提供设备状态信息
  - 接口应具备状态端口

查询工作方式流程图



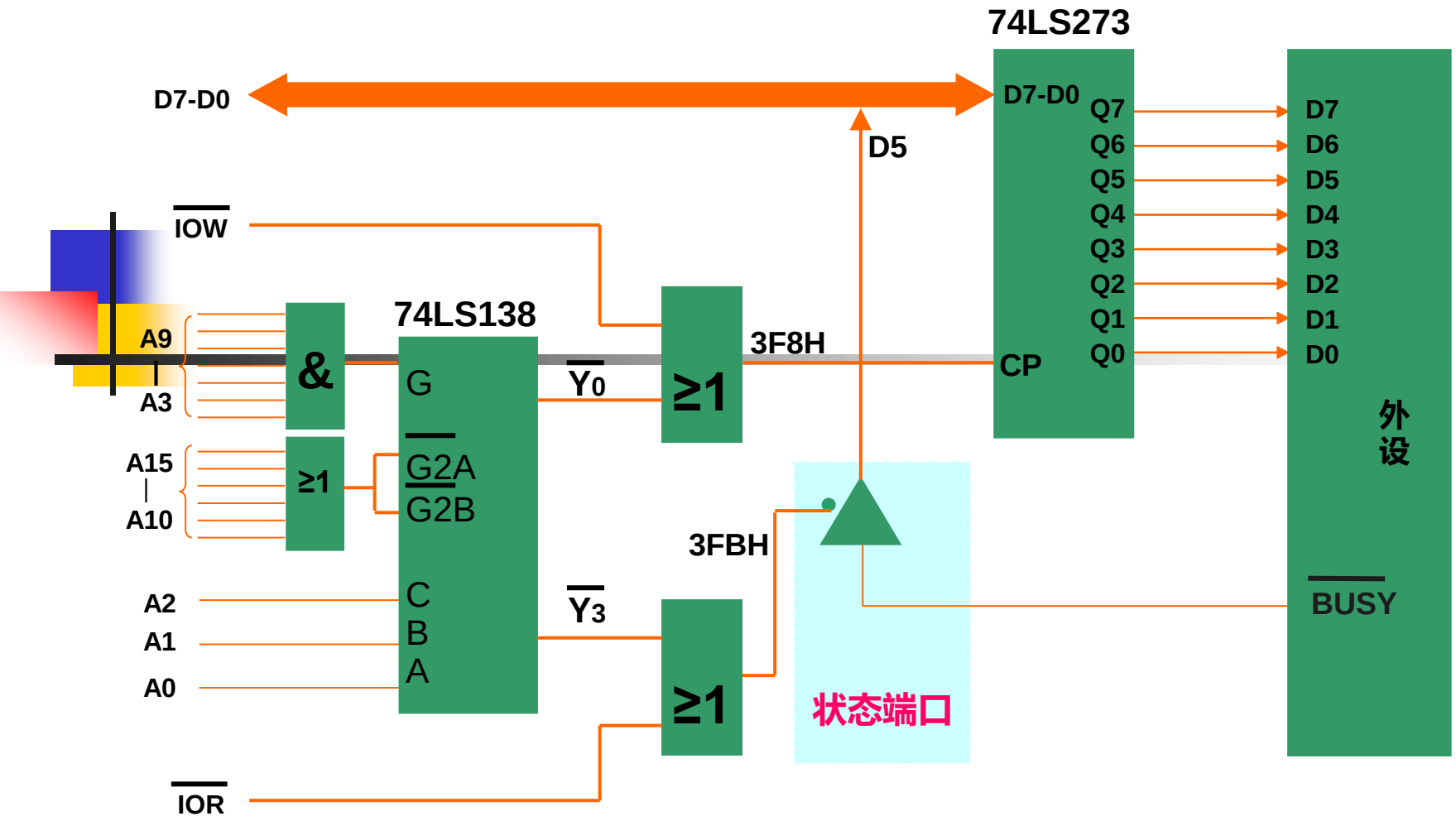

防止死循环
超时?
Y
超时错
N
读入并测试外设状态
N
READY?
Y
复位计时器
与外设进
行数据交换
N
传送完
Y
结 束

查询工作方式例

- 外设状态端口地址为03FBH，第5位(bit5)为状态标志（=1忙，=0准备好）
- 外设数据端口地址为03F8H，写入数据会使状态标志置1；外设把数据读走后又把它置0。
- 试画出其电路图，并将DATA下100B数据输出。

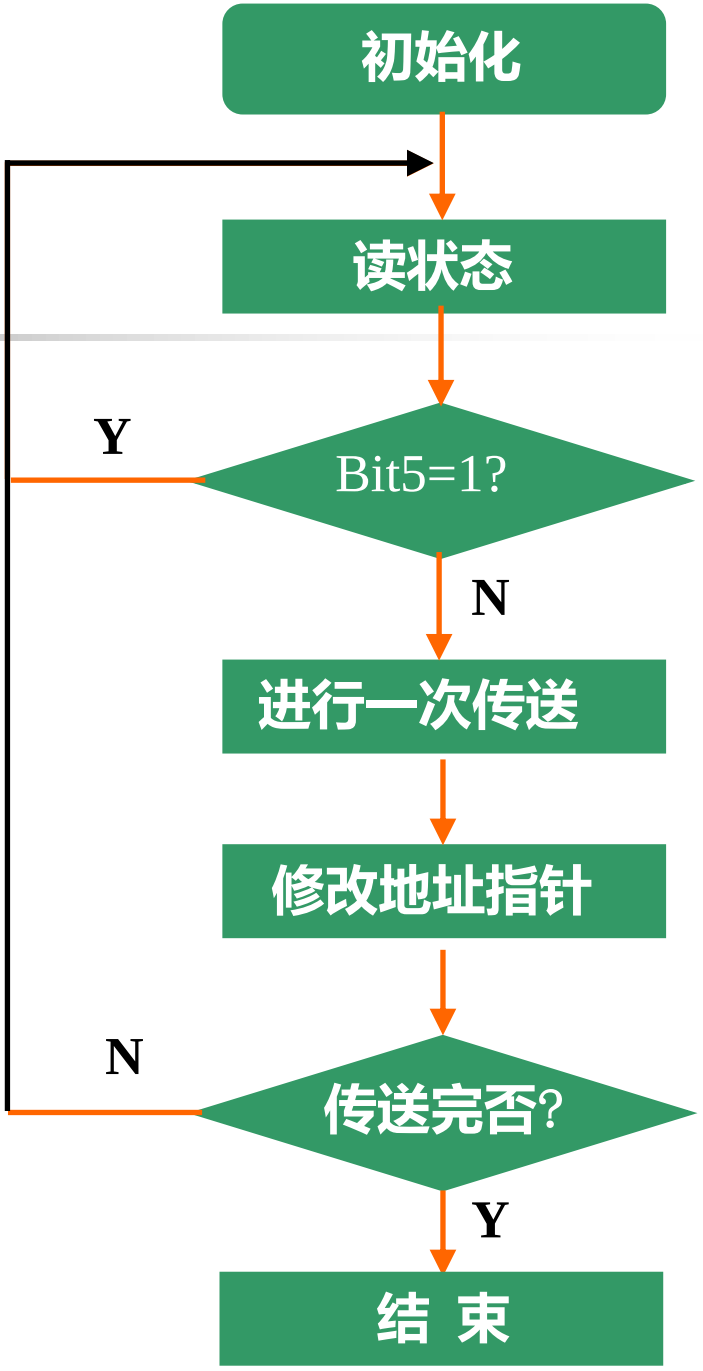
状态端口地址：0000 0011 1111 1011

数据端口地址：0000 0011 1111 1000



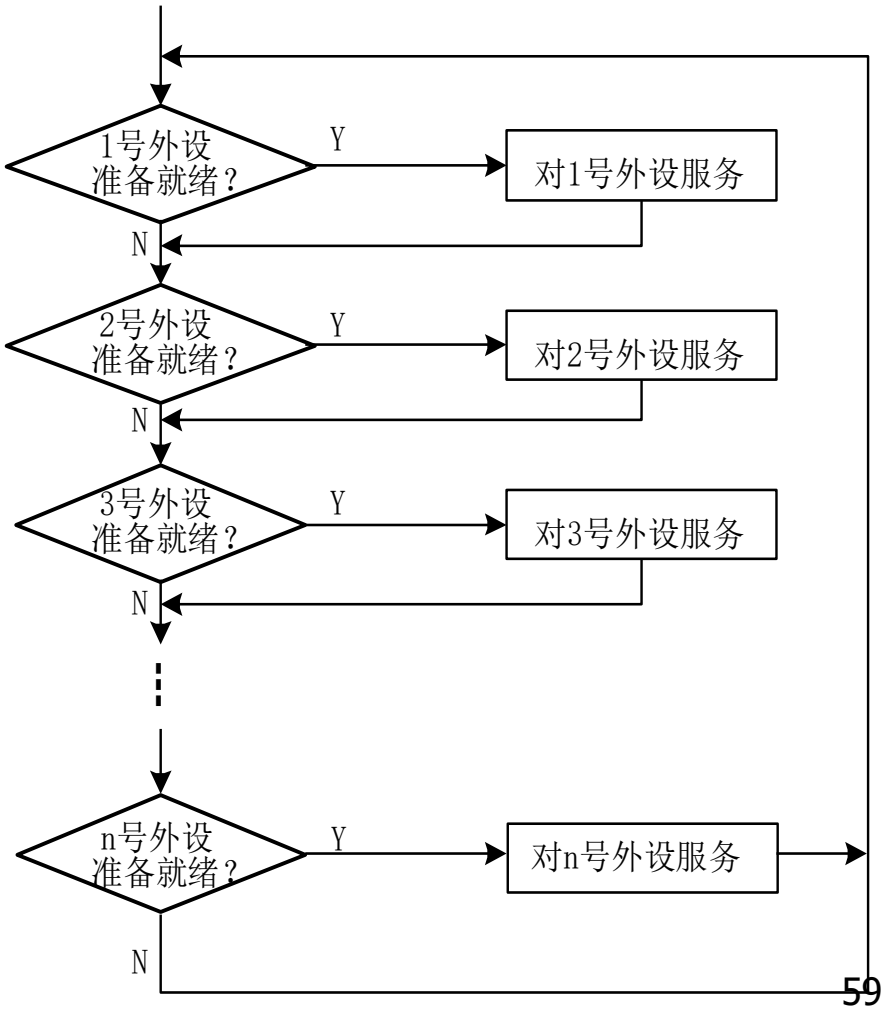
控制程序

```
        LEA SI,DATA
        MOV CX,100
AGAIN : MOV DX,03FBH
WAITT：  IN AL,DX
        TEST AL,20H
        JNZ   WAITT
        MOV DX,03F8H
        MOV AL，[SI]
        OUT DX，AL
        INC SI
        LOOP  AGAIN
        HLT
```



查询工作方式

- 优点：
  - 软硬件比较简单
- 缺点：
  - CPU效率低，数据传送的实时性差，速度较慢



三、中断控制方式

- 特点：
  - 外设在需要时向CPU提出请求，CPU再去为它服务。服务结束后或在外设不需要时，CPU可执行自己的程序。
- 优点：
  - CPU效率高，实时性好，速度快。
- 缺点：
  - 程序编制相对较为复杂。

以上三种I/O方式的共性

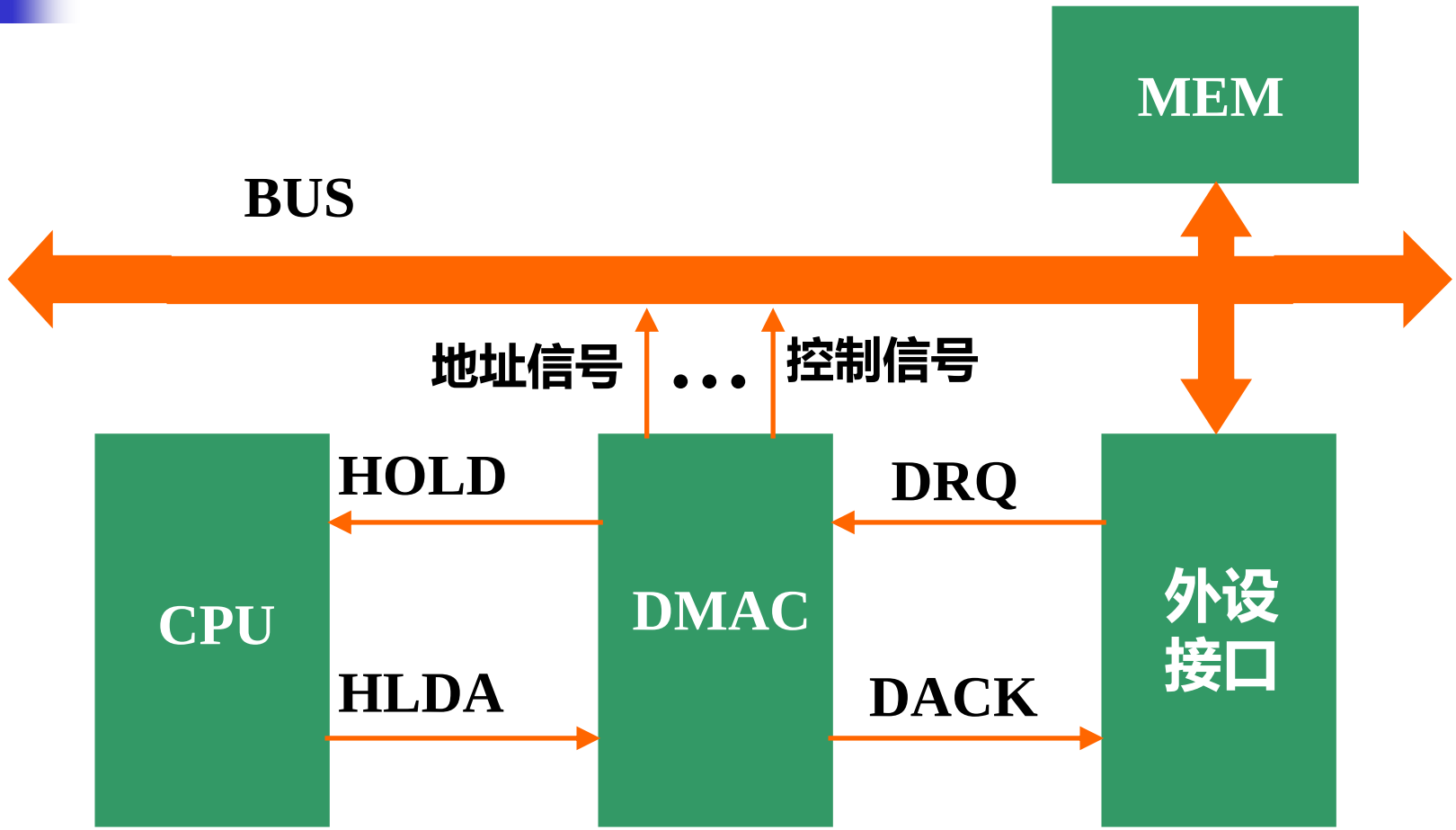
- 信息的传送均需通过CPU
- 软件：
  - 外设与内存之间的数据传送是通过CPU执行程序来完成的（PIO方式）；
- 硬件：
  - I/O接口和存储器的读写控制信号、地址信号都是由CPU发出的。
- 缺点：
  - 程序的执行速度限定了传送的最大速度

四、DMA控制方式

- 特点：
  - 外设直接与存储器进行数据交换，CPU不再担当数据传输的中介者；
  - 总线由DMA控制器（DMAC）进行控制（CPU要放弃总线控制权），内存/外设的地址和读写控制信号均由DMAC提供。

这种专门的硬件电路就是DMA控制器，简称为DMAC。该集成电路产品有Zilog 公司的Z80-DMA，Intel公司的8257，8237A和Motorola的MC6844等。

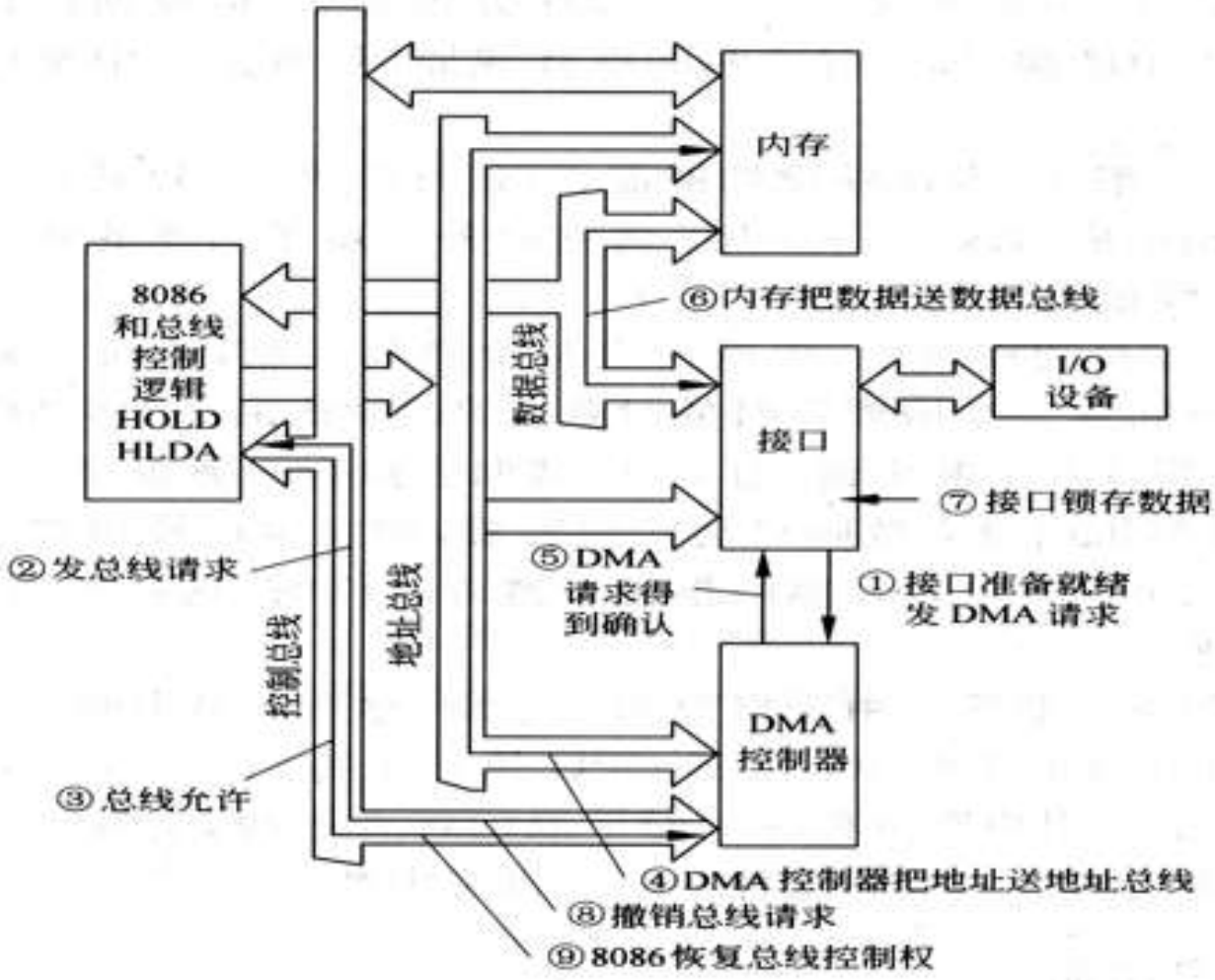
DMA控制方式



DMA控制方式的工作过程

- 外设向DMA控制器发出“DMA传送请求”信号DRQ；
- DMA控制器收到请求后，向CPU发出“总线请求”信号HOLD；
- CPU在完成当前总线周期后会立即发出HLDA信号，对HOLD信号进行响应；
- DMA控制器收到HLDA信号后，就开始控制总线，并向外设发出DMA响应信号DACK。

8086用DMA方式传送单个数据



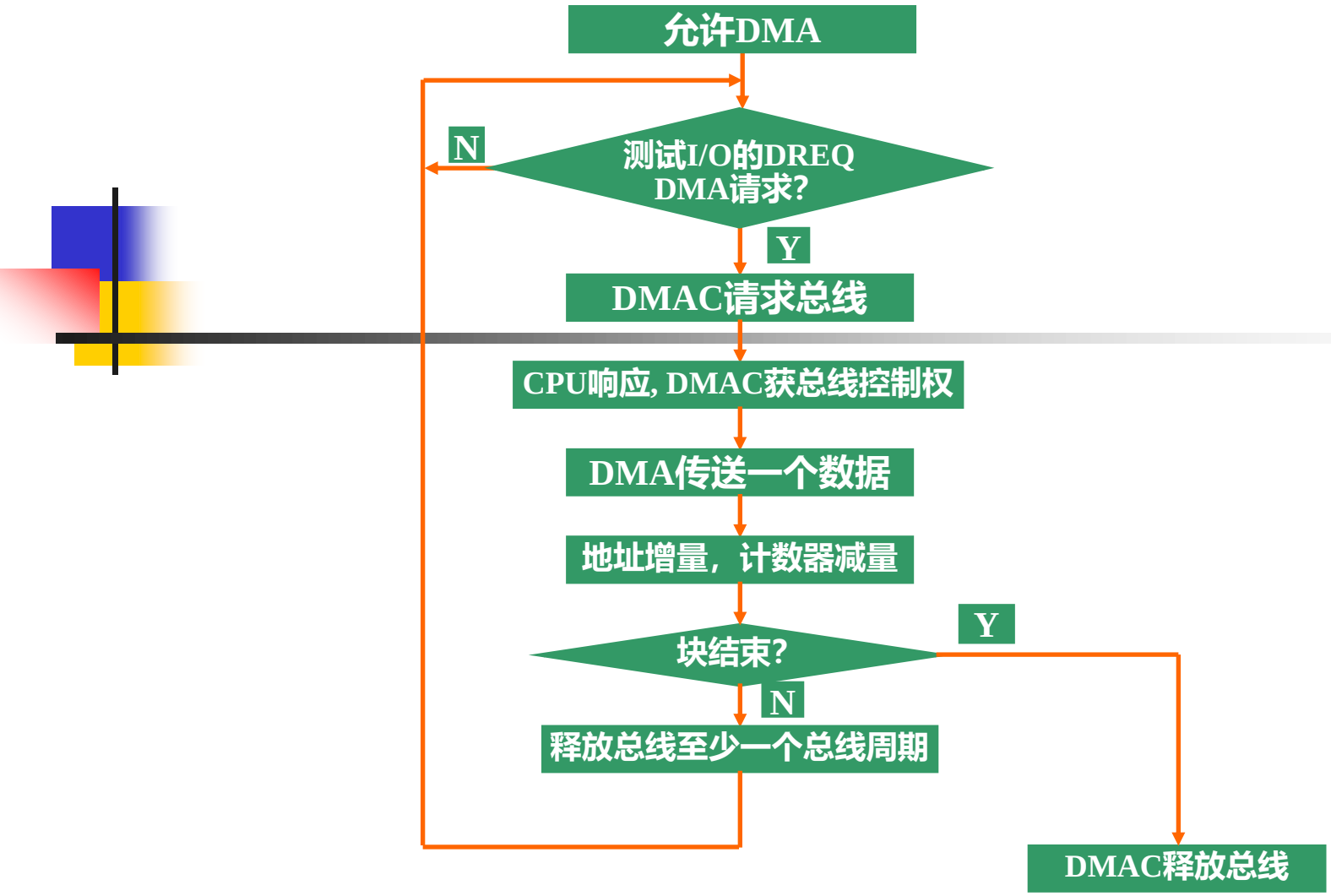
DMA控制方式工作过程例

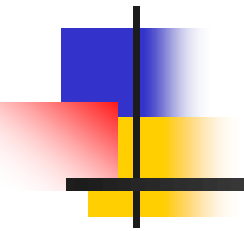
- 例：从外设向内存传送若干字节数据
  - DMAC向I/O接口发出读信号；
  - 向地址总线上发出存储器的地址；
  - 发出存储器写信号和AEN信号；
  - 传送数据并自动修改地址和字节计数器
  - 判断是否需要重复传送操作；
  - 若数据传送完，DMA控制器撤销发往CPU的HOLD信号；
  - CPU检测到HOLD失效后，则撤销HLDA信号，并在下一时钟周期重新开始控制总线。

DMA工作方式

- 周期窃取：
  - 每个DMA周期只传送一个字节或一个字就立即释放总线。
- 数据块传送：
  - DMAC在申请到总线后，将一块数据传送完后才释放总线，而不管中间DREQ是否有效。
- 直接存取方式：
  - DMA的数据传送请求直接发到主存储器，在得到响应后，整个工作过程在DMA控制器中由硬件完成。

周期窃取的DMA方式：


允许DMA
测试I/O的DREQ
DMA请求?
N
Y
DMAC请求总线
CPU响应, DMAC获总线控制权
DMA传送一个数据
地址增量，计数器减量
块结束?
Y
N
释放总线至少一个总线周期
DMAC释放总线



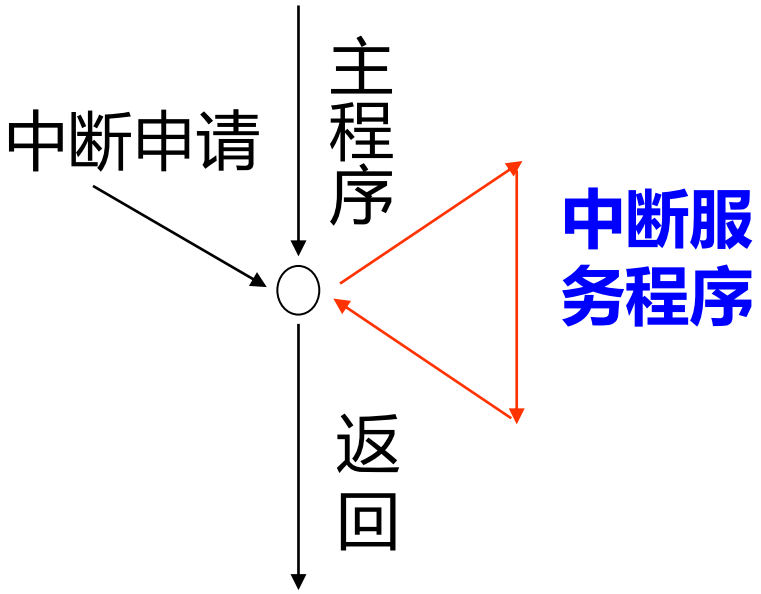
§6.4 中断技术

掌握：

- 中断的基本概念
- 中断响应的一般过程
- 中断向量表及其初始化
- **8088/8086**中断系统

一、中断的基本概念

所谓中断，是指这样一个过程：当CPU正在执行程序过程中，由于某一**突然**事件的发生，**暂时中止**正在执行的程序，转去处理突然发生的事件（执行中断服务程序），处理完毕，**再返回**原来被中止的程序继续运行。



中断示意图

日常生活中类似“中断” 的例子：

例1：门铃响时，放下手中正在做的事情，跑去开门，招待客人；客人走后，接着做原来的事情。

例2：教师讲课过程中，同学遇有疑问，不同理解之处，随时向老师提问。

中断与子程序的异同

*在中断的整个处理过程中，外部事件发出的**中断请求**和CPU对**中断的响应**与当前正在执行的指令之间没有任何关系。——**随机性***

*主程序调用子程序的过程则是程序员事先通过编程，利用CALL指令来实现的，当CPU执行到该指令时，就会转去执行被调用的子程序。——**确定性***

引入中断的原因

- 提高数据传输率；
- 避免了CPU不断检测外设状态的过程，提高了CPU的利用率。
- 实现对特殊事件的实时响应。

二、外部中断响应的一般过程

- 中断请求
- 中断源识别及中断判优
- 中断响应
- 中断处理（服务）
- 中断返回

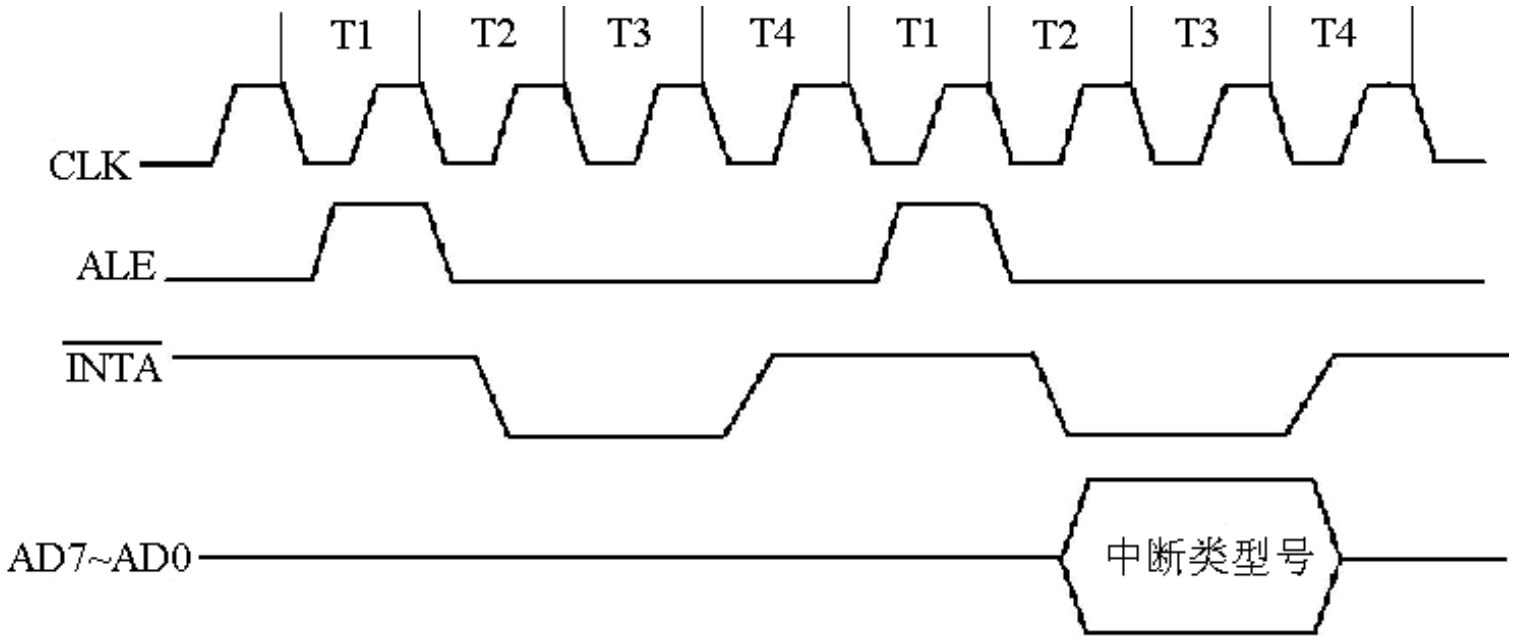
中断请求

- **NMI**
- **INTR**

- 中断请求信号应保持到中断被处理为止；
- CPU响应中断后，中断请求信号应及时撤销。

中断源的识别

- 中断源识别及判优由硬件系统完成
- CPU响应INTR中断时，会产生两个中断响应总线周期，要求PIC（可编程中断控制器）在第2个中断响应总线周期把中断类型码放到数据总线上，供CPU读入。



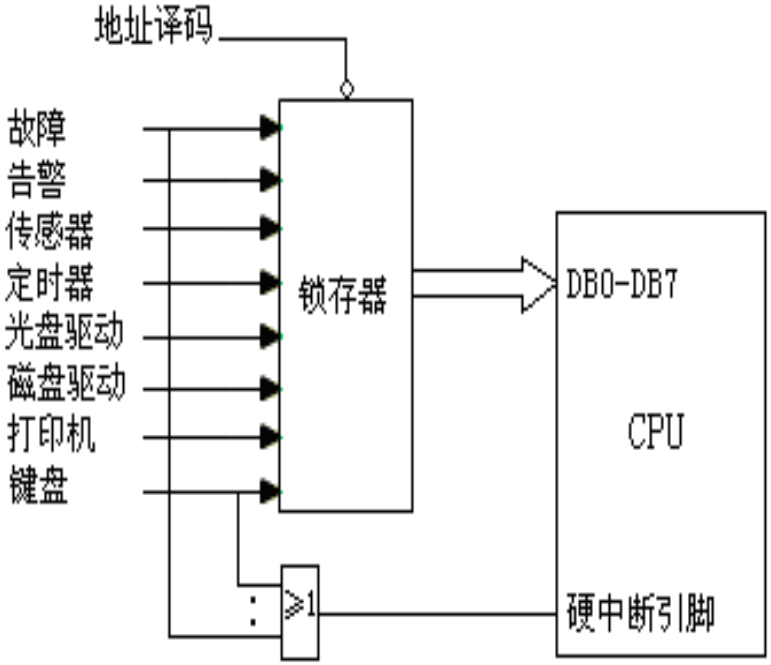
中断判优

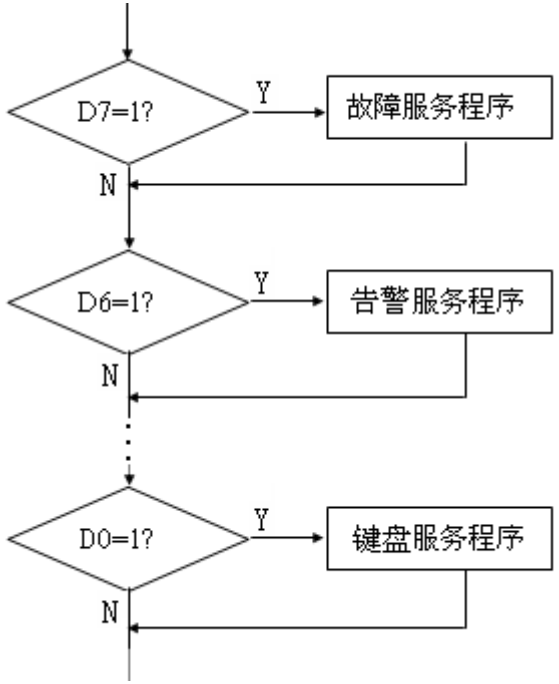
- 当有多个中断源同时提出请求时，需要确定首先响应哪一个中断源。
- 优先级法则
  - 低优先级的中断程序允许被高优先级的中断源所中断
- 排队法则
  - 先来先响应

中断嵌套

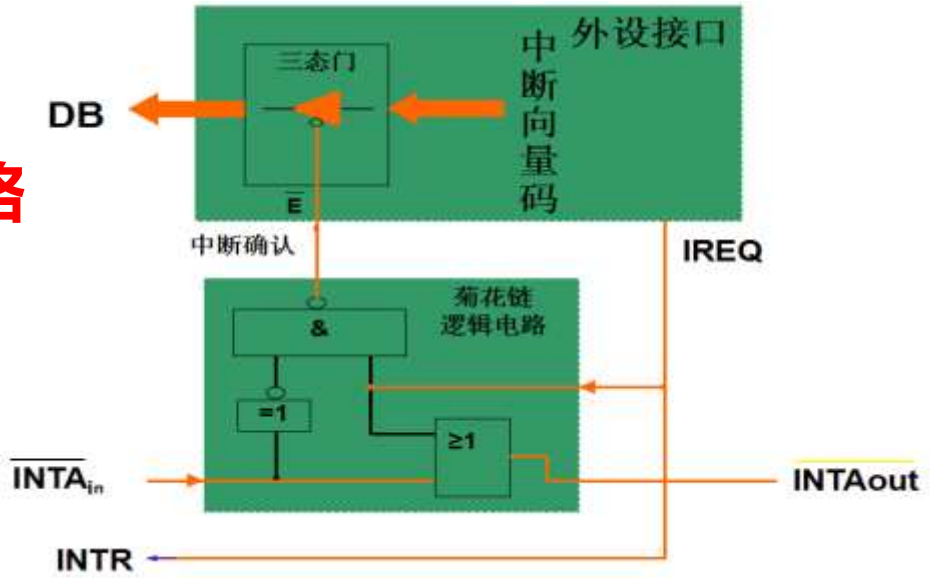
```
        IN AL, PORT_R  ; PORT_R是锁存器地址
        CLC
        RCR AL, 1
        JC INTS01          ; 转向故障处理
A1:     RCR     AL, 1
        JC INTS02          ; 转向告警处理
A2:     ……
        ……
INTS01: ……
        JMP     A1
INTS02: ……
        JMP     A2
        ……
```

- 软件判优
  - 顺序查询中断请求，先查询的先服务
    - 即先查询的优先级别高



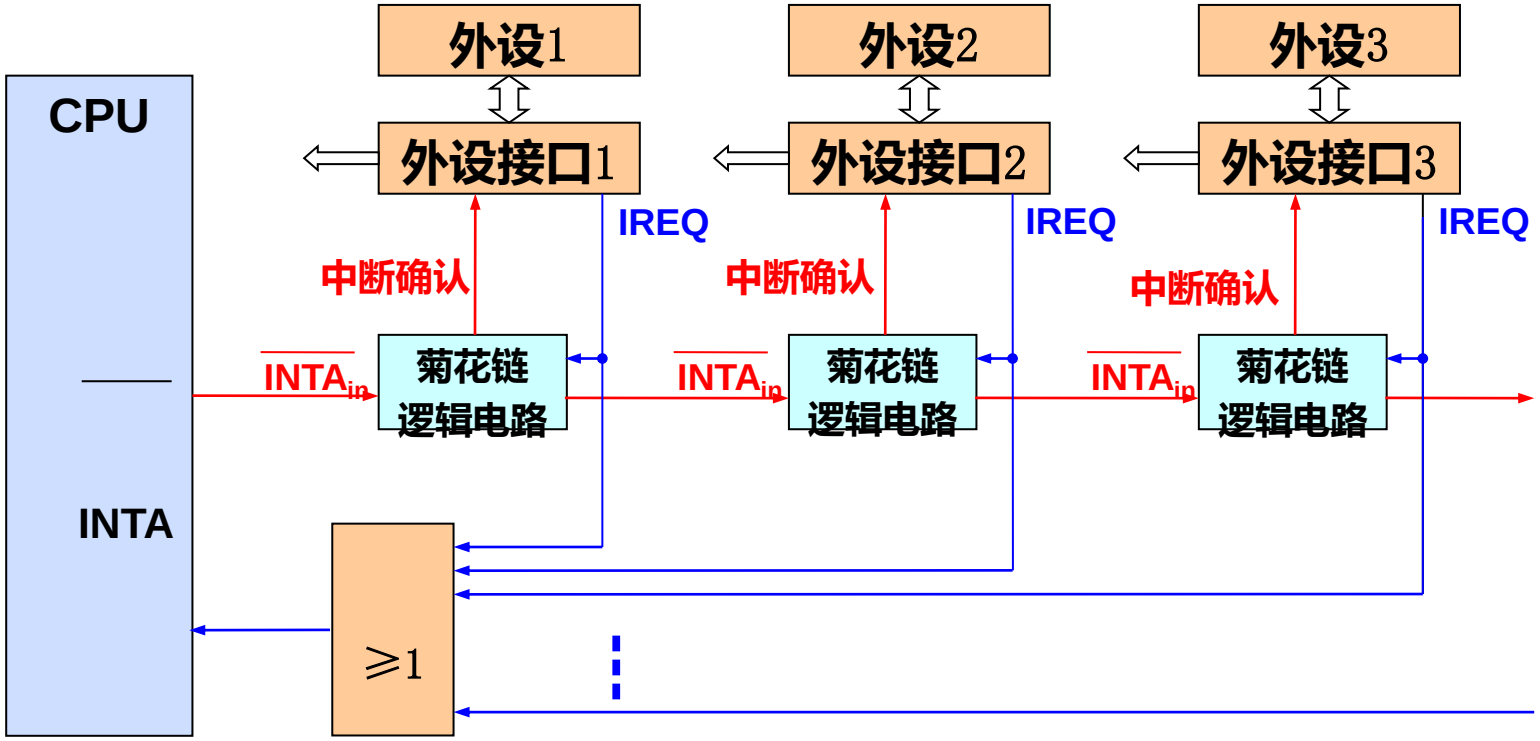


菊花链逻辑电路



- 硬件判优
  - 硬件控制器（8259）
  - 链式判优、并行判优（中断向量法）

链式判优电路



中断响应

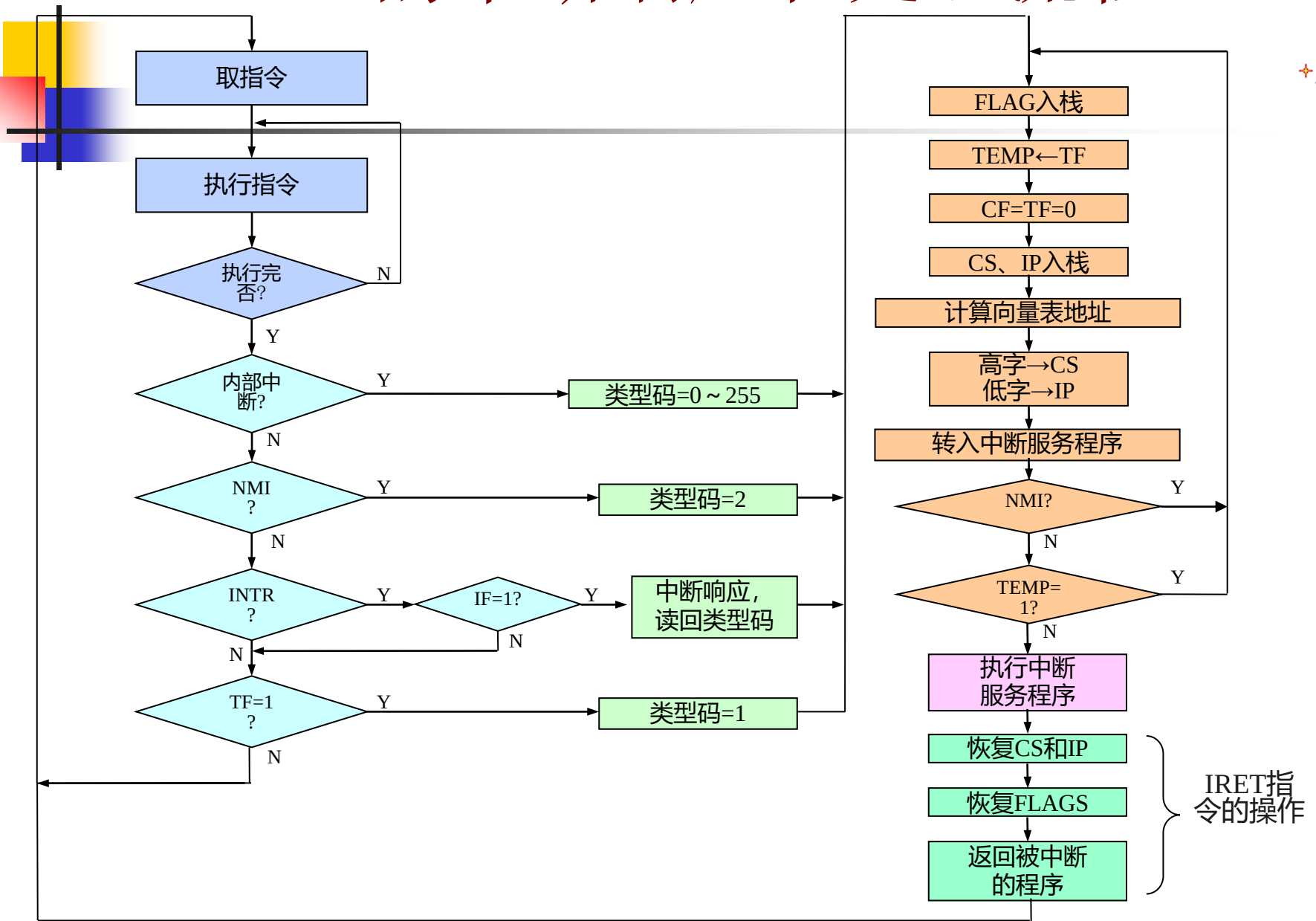
- 向中断源发出$\overline{\text{INTA}}$中断响应信号；
- 关中断（进入中断服务程序之前，系统自动进行）
- 保护硬件现场
  - 将FLAGS压入堆栈
- 保护断点
  - 将CS、IP压入堆栈
- 获得中断服务程序入口地址

由硬件系统完成

中断服务子程序（响应、处理、返回）

- 关中断，保护现场，保护断点，找入口地址
- 保护软件现场（参数）
- 开中断（**STI**）
- 中断处理
- 关中断（**CLI**）
- 恢复现场
- 中断返回 → 执行IRET指令，使IP、CS和FLAGS从堆栈弹出

8088/8086的中断响应和处理流程



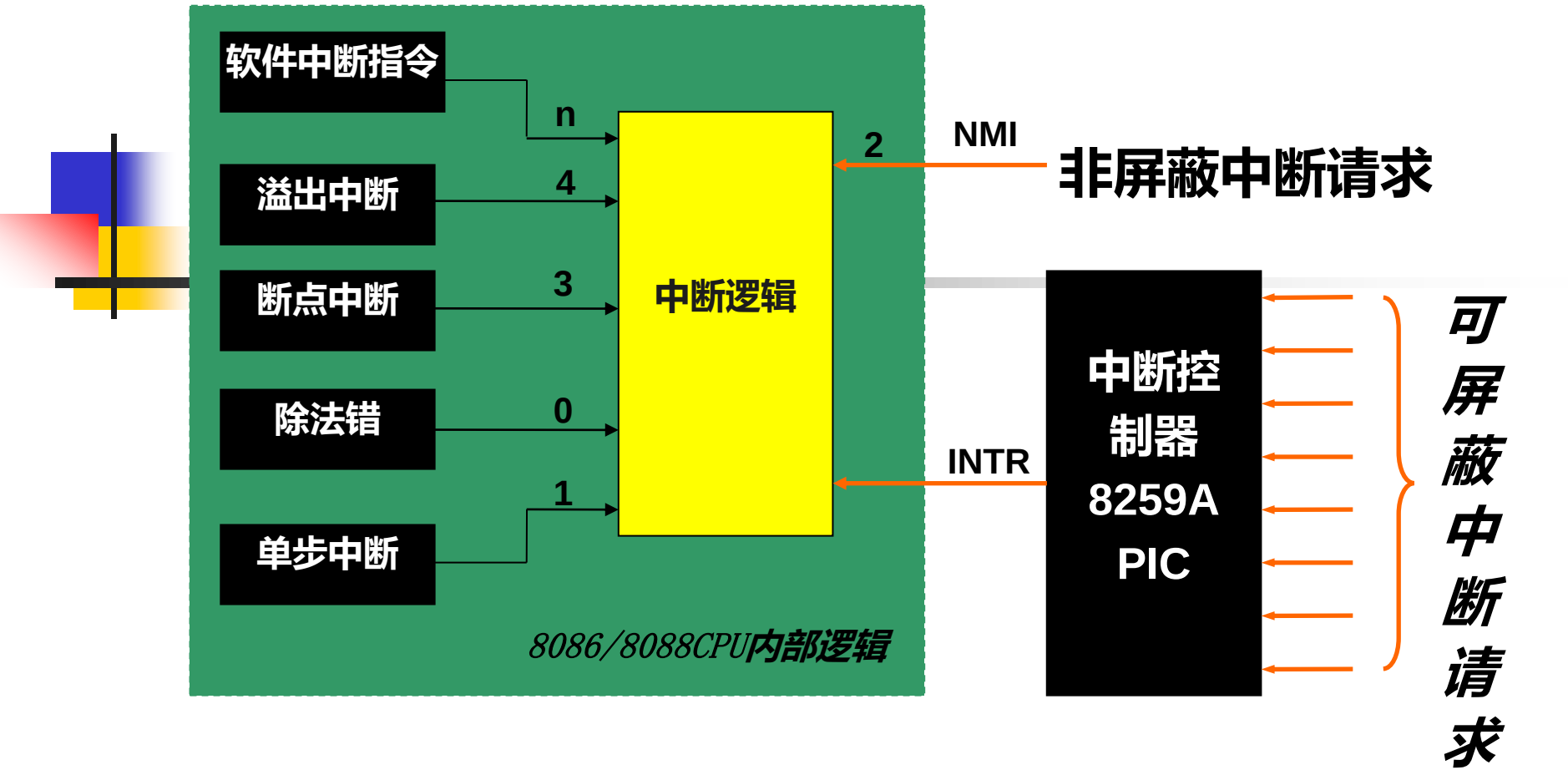
三、8086/8088的中断系统

- 与中断有关的控制线为：NMI、INTR、INTA
- 8086系统的中断源（256个）
  - 内部中断
    - 软件中断：即INT n指令，类型号n(0-255)。
      - *除法出错：类型号0，商大于目的操作数所能表达的范围时产生。*
      - *单步中断：类型号1，TF=1时产生（当前指令需执行完）*
      - *断点中断：类型号3，这是一个软件中断，即INT 3指令。*
      - *溢出中断：类型号4，这是一个软件中断，即INTO指令。*
  - 外部中断
    - 非屏蔽中断NMI：类型号2，不可用软件屏蔽，CPU必须响应它。
    - 可屏蔽中断INTR：类型号由PIC提供。IF=1时CPU才能响应。

- **中断源：引起CPU中断的事件，发出中断请求的来源**

中断源	优先级
除法出错中断 INT 0	最高 ↓ 最低
单步中断 INT 1	
非屏蔽中断 NMI 2号	
溢出中断 INT 4	
软件中断 INT n	
可屏蔽中断 INTR	

8086/8088中断源类型：



除法出错中断

- 在执行除法指令时，若除数为0或商超过了寄存器所能表达的范围，则产生一个向量号为0的内部中断，称为除法出错中断

```
MOV BL,0
IDIV BL      ；除数BL = 0，产生除法溢出中断
```

```
MOV AX,200H
MOV BL,1
DIV BL       ；商 = 200H，不能用AL表达
             ；产生除法溢出中断
```

溢出中断

- 在执行溢出中断指令INTO时，若溢出标志OF为1，则产生一个向量号为4的内部中断，被称为溢出中断

```
MOV AX,2000H
ADD AX, 7000H
        ；2000H＋7000H＝9000H，溢出：OF＝1
INTO  ；因为OF＝1，所以产生溢出中断
```

单步中断

- 若单步中断TF为1，则在每条指令执行结束后产生一个向量号为1的内部中断，称为单步中断

例如：DEBUG.EXE调试程序的单步命令T就利用单步中断实现对程序的单步调试

指令软中断（用户自定义的软件中断）

- 在执行中断调用指令INT n时产生的一个向量号为n（0～255）的内部中断，称为指令中断
- 其中向量号为3的指令中断比较特别（生成一个字节的指令代码：11001100），常用于程序调试，被称为断点中断

例如：DEBUG.EXE调试程序的运行命令G设置的断点，就是利用INT 3指令实现的

不可屏蔽中断

- 通过不可屏蔽中断请求信号向微处理器提出的中断请求，微处理器无法禁止，将在当前指令执行结束予以响应，这个中断被称为不可屏蔽中断
- **8086**的不可屏蔽中断的向量号为**2**，不可屏蔽中断请求信号为**NMI**
- 不可屏蔽中断主要用于处理系统的意外或故障。例如：
  - 电源掉电前的数据保护
  - 存储器读写错误的处理

可屏蔽中断

- 外部通过可屏蔽中断请求信号向微处理器提出的中断，微处理器在允许可屏蔽中断的条件下，在当前指令执行结束予以响应，同时输出可屏蔽中断响应信号，这个中断就是可屏蔽中断
- **8086的可屏蔽中断请求和响应信号分别是INTR和INTA*；由IF标志控制可屏蔽中断是否允许响应（CLI或STI)；向量号来自外部中断控制器**
- **8086通常需要配合中断控制器8259A共同处理可屏蔽中断**
- 可屏蔽中断主要用于主机与外设交换数据

中断向量表

中断 ⟺ 中断向量码代号 ⟺ 中断服务程序

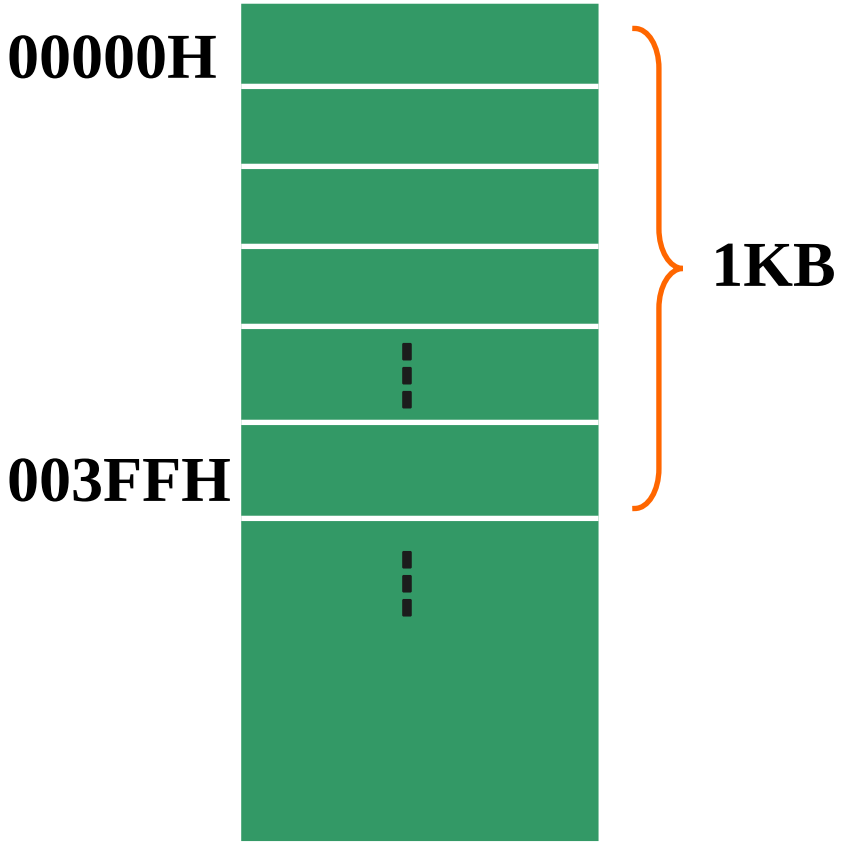
（0～255）

中断向量：是中断处理子程序的入口地址，每个中断类型对应一个中断矢量

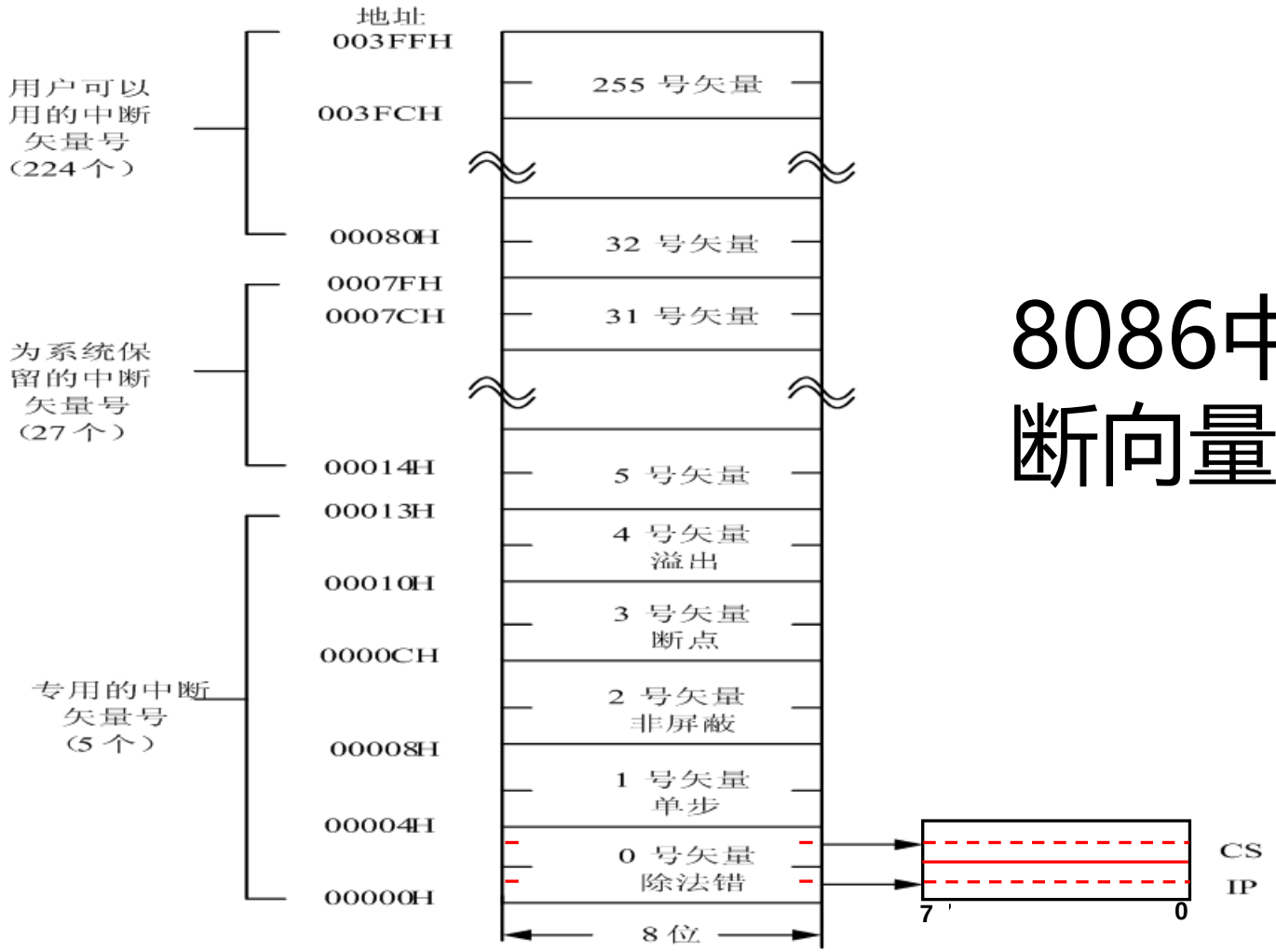
中断向量表：在内存中，开辟一个区域存放中断服务程序的入口地址的表。对于8086，这个区域的首地址00000H（固定）

地址（H）	矢量号	中断名称	地址（H）	矢量号	中断名称
0—3	0	除以零	60—63	18	常驻 BASIC 入口
4—7	1	单步	64—67	19	引导程序入口
8—B	2	不可屏蔽	68—6B	1A	时间调用
C—F	3	断点	6C—6F	1B	键盘 CTRL-BREAK 控制
10—13	4	溢出	70—73	1C	定时器报时
14—17	5	打印屏幕	74—77	1D	显示器参数表
18—1B	6	保留	78—7B	1E	软盘参数表
1D—1F	7	保留	7C—7F	1F	字符点阵结构参数表
20—23	8	定时器	80—83	20	程序结束，返回 DOS
24—27	9	键盘	84—87	21	DOS 系统功能调用
28—2B	A	保留	88—8B	22	结束地址
2C—2F	B	串口 2	8C—8F	23	CTRL-BREAK 退出地址
30—33	C	串口 1	90—93	24	标准错误出口地址
34—37	D	硬盘	94—97	25	绝对磁盘读
38—3B	E	软盘	98—9B	26	绝对磁盘写
3C—3F	F	打印机	9C—9F	27	程序结束，驻留内存
40—43	10	视频显示 I/O 调用	A0—FF	28—3F	为 DOS 保留
44—47	11	设备配置检查调用	100—17F	40—5F	保留
48—4B	12	存储器容量检查调用	180—19F	60—67	为用户软中断保留

- 每个入口占用4 Bytes，低字为段内偏移，高字为段基址；
- 表的地址位于内存的00000H～003FFH，大小为1KB，共256个入口。



8086中断向量表


地址
003FFH
003FCH
00080H
0007FH
0007CH
00014H
00013H
00010H
0000CH
00008H
00004H
00000H
用户可以用的中断矢量号（224个）
为系统保留的中断矢量号（27个）
专用的中断矢量号（5个）
255 号矢量
32 号矢量
31 号矢量
5 号矢量
4 号矢量 溢出
3 号矢量 断点
2 号矢量 非屏蔽
1 号矢量 单步
0 号矢量 除法错
8 位
CS
IP
7
0

8086中断矢量表的初始化

方法一：采用DOS系统功能调用，即INT 21H的25H号(设置中断向量)功能调用

已知：某一中断服务程序的入口地址为INT（逻辑地址）

格式：

```
MOV   DX, OFFSET INT   ；中断服务程序偏移量→DX
MOV   AX, SEG INT      }；中断服务程序段地址→AX
MOV   DS, AX           }
MOV   AL, n            ；中断矢量号n→AL
MOV   AH, 25H          ；功能号25H→AH
INT   21H
```

方法二：直接将中断服务程序的入口地址装入中断矢量表中

已知：某一中断矢量号为n，中断服务程序的入口地址为INT（逻辑地址）

```
MOV    AX, 0
MOV    ES, AX
MOV    DI, n                    ；中断矢量号n→ DI
SHL    DI, 1                    ；n× 2
SHL    DI, 1                    ；n× 4
MOV    AX, OFFSET  INT          ；中断服务程序的偏移量→AX
MOV    BX, SEG     INT          ；中断服务程序的段基址→BX
MOV    ES: [DI],   AX           ；中断服务程序的偏移量→矢量表
MOV    ES: [DI+2], BX ；中断服务程序的段基址→矢量表
```

常用的BIOS中断调用p. 167：

特点： BIOS中断调用要比DOS中断调用复杂些，但运行速度快，功能更强些BIOS中断调用颇具特色的是显示中断子程序，其矢量号是10H

（1）设置显示方式

矢量号：　10H
功能号：　AH＝0
入口参数：AL＝常用显示方式号（0～6）

显示方式号	显示方式
0	40列×25行黑白文本方式
1	40列×25行彩色文本方式
2	80列×25行黑白文本方式
3	80列×25行彩色文本方式
4	320列×200行黑白图形方式
5	320列×200行彩色图形方式
6	640列×200行黑白图形方式

注意：对于高分辨率，只支持黑白显示

例：屏幕设置成80×25彩色文本方式

程序：

```
MOV     AH，0       ；设功能号
MOV     AL，3       ；设显示方式
INT     10H
```

（2）设置光标位置

矢量号：　10H

功能号：　AH＝2

入口参数：　BH＝页号，通常取0页

DH＝行号，取值0～24

DL＝列号，对于40列文本，取值0～39

对于80列文本，取值0～79

例：将光标置在第10行30列

程序：

```
MOV    BH, 0
MOV    DH, 10
MOV    DL, 30
MOV    AH, 2
INT    10H
```

（3）在当前光标处写字符和属性

矢量号：　10H

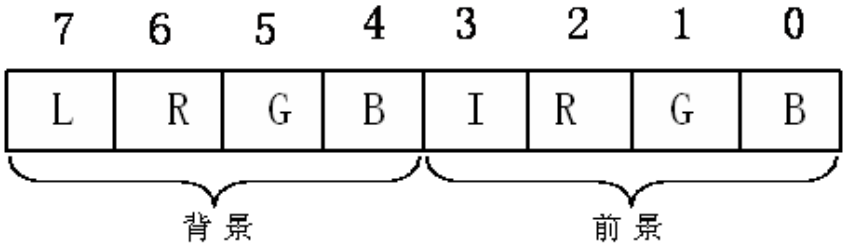
功能号：　AH＝9

入口参数：　BH＝页号

AL＝显示字符的ASCII码

BL＝属性

CX＝重复显示的次数



L位为1时表示背景闪烁，否则不闪

I位为1时表示前景为高亮度，否则为一般亮度

文本方式16种颜色的组合

颜色 IRGB	颜色 IRGB	颜色 IRGB	颜色 IRGB
黑 0000	红 0100	灰 1000	浅红 1100
兰 0001	品红 0101	浅兰 1001	浅品红 1101
绿 0010	棕 0110	浅绿 1010	黄 1110
青 0011	灰白 0111	浅青 1011	白 1111

例：用兰色清屏，然后在第10行30列显示20个红底白字 ‘A’

程序：

```
MOV     AL, 0       ；AL=0时，清屏参数
MOV     BL, 10H     ；蓝色背景，无闪烁，属性→BL
MOV     AH, 6       ；清屏功能号6
MOV     CX, 0       ；从0行到0列
```

```
MOV     DH, 24          ；到24行
MOV     DL, 79          ；79列
INT     10H             ；清屏幕
MOV     AH, 2
MOV     BH, 0           ；页号
MOV     DH, 10
MOV     DL, 30
INT     10H             ；光标设置在第10行30列
MOV     AL, ’A’
MOV     CX, 20                  ；重复显示20个字符‘A’
MOV     BH, 0                   ；页号
MOV     BL, 4FH                 ；属性→BL，红底，白字
MOV     AH, 9
INT     10H             ；显示20个字母‘A’
```

（4）设置图形方式显示的背景和彩色组

矢量号：　10H

功能号：　AH＝0BH

入口参数：　当BH＝0时，BL＝背景颜色，范围0～15

当BH＝1时，BL＝颜色组， 范围0～1：

0表示绿/红/黄，1表示青/品红/白

（5）串行通信口初始化

矢量号：　14H

功能号：　AH＝0H

入口参数：　AL＝初始化参数，DX＝0表示对COM1初始化；DX=1表示对COM2初始化

初始化参数格式：

7	6	5	4	3	2	1	0
·	·	·	·	·	·	·	·

位7、6、5表示波特率：

0	0	0	110	波特
0	0	1	150	波特
0	1	0	300	波特
0	1	1	600	波特
1	0	0	1200	波特
1	0	1	2400	波特
1	1	0	4800	波特
1	1	1	9600	波特

位4、3表示奇偶校验设定：

0	0	无奇偶校验
0	1	奇校验
1	1	偶校验

位2表示停止位数设定

0	1位停止位
1	2位停止位

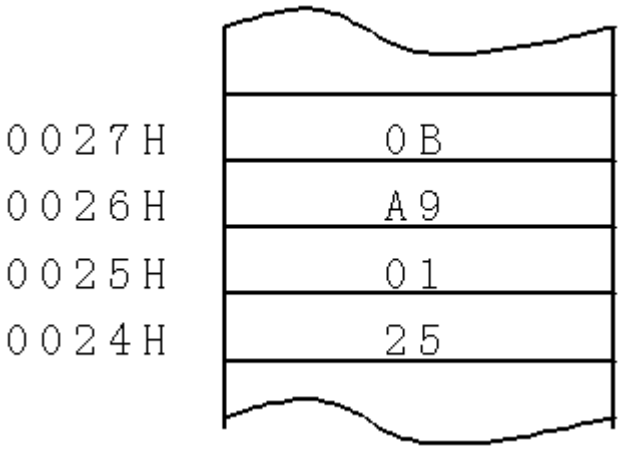
位1、0表示通信数据位数设定

1	0	7位数据通信
1	1	8位数据通信

例： 键盘中断的矢量号为09H，它的中断服务程序入口地址为0BA9H:0125H，求它在中断矢量表中的地址及相应单元内容

解： 键盘中断对应的中断矢量表位于0000:0024H (09H×4=24H)开始的4单元

这4个单元的内容见表:



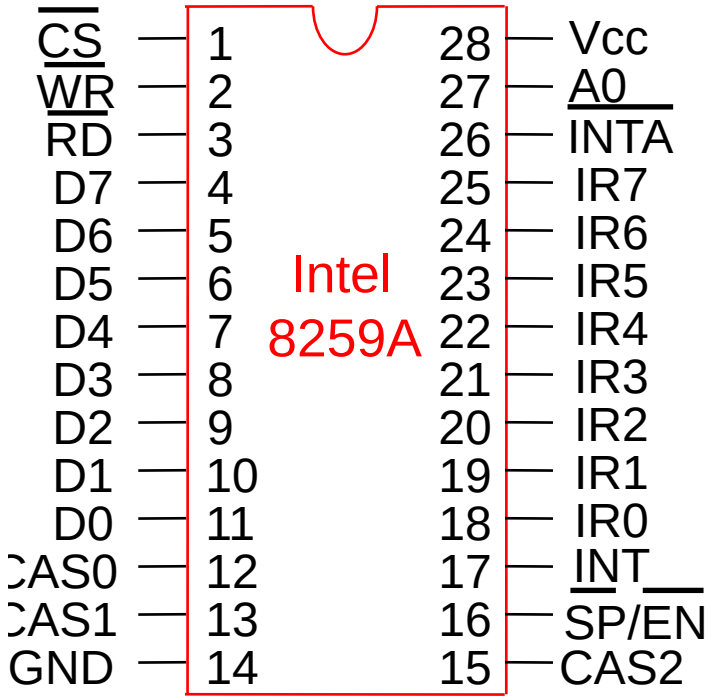
四、可编程中断控制器8259A

- **Intel8259A是一种可编程的中断控制器**PIC（Programmable Interrupt Controller)
- **可为CPU管理和处理8个中断源电路的中断，并对它们进行优先级管理。**
- **可与其它8个8259A芯片组成主从式中断系统，实现64级中断源控制。**
- **优先级方式在执行程序时可动态改变，有多种中断管理方式。**
- **每一中断源中断都可以屏蔽或允许，可通过编程选择。**
- **能自动送出中断类型号，使CPU方便的找到中断服务程序的入口地址。**

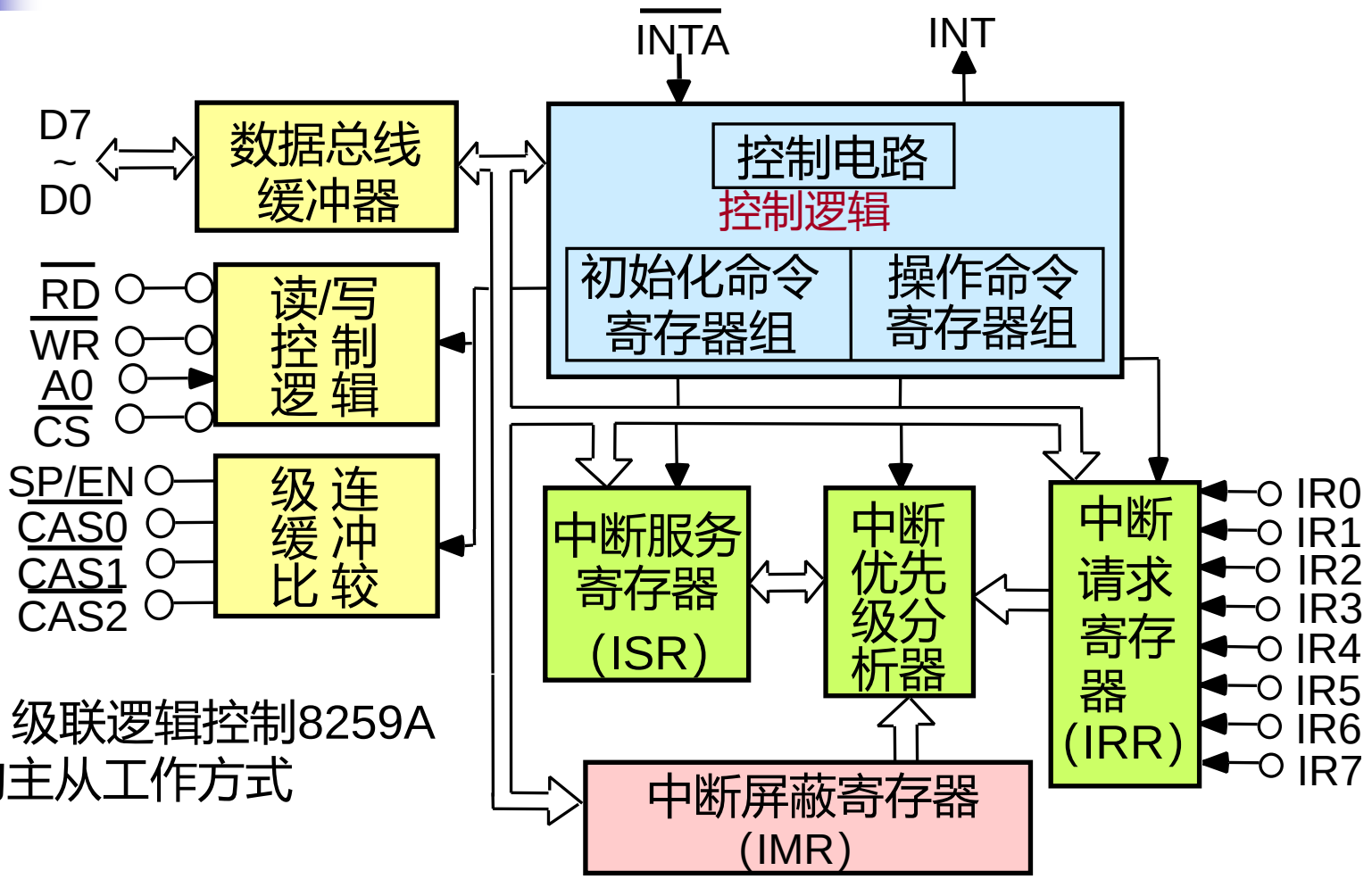
8259A的结构功能

※ 内部结构说明

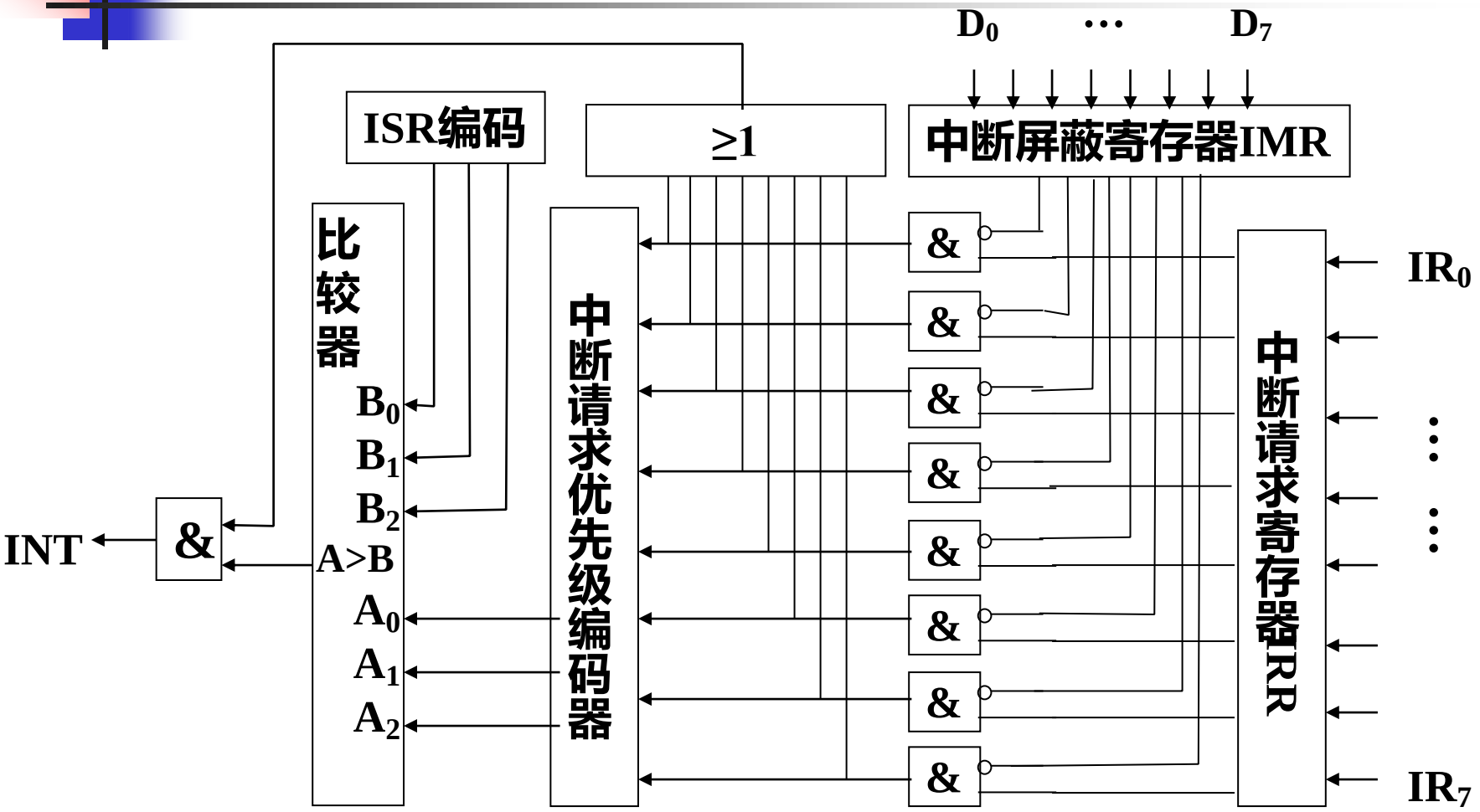
- D7~D0 双向三态数据线，传送数据
- /RD 读信号，低电平有效
- /WR 写信号，低电平有效
- /CS 片选信号，低有效，给8259分配地址
- A0 用于选择8259内部不同的寄存器。
- CAS2~CAS0 级联信号线，当8259为主片时为输出，从片时为输入
- SP/EN 在非缓冲工作方式时，用作输入，决定8259A为主片还是从片；在缓冲工作方式时，用作输出，由它打开总线缓冲器



8259A引脚图



级联逻辑控制8259A的主从工作方式


D0 ··· D7
ISR编码
≥1
中断屏蔽寄存器IMR
比较器
B0
B1
B2
A>B
A0
A1
A2
中断请求优先级编码器
&
INT
中断请求寄存器IRR
IR0
IR7

8259内部结构说明

- **IRR** — 中断请求寄存器，保存 8个输入端IR0~IR7的中断申请状态，输入可高电平有效，也可上升沿有效。当某IRi有效时，IRR相应位置1。
- **ISR** — 中断服务寄存器，保存CPU正在处理的中断请求；
- **IMR** — 中断屏蔽寄存器，对中断申请进行屏蔽控制；
- 中断优先级分析器决定中断申请的优先级；
- 控制逻辑
  - 初始化命令寄存器组：ICW1~ICW4
  - 操作命令寄存器组： OCW1~OCW3
  - INT 中断申请输出，高电平有效，可连接8086 INTR引脚
  - $\overline{\text{INTA}}$ 中断响应输入，接收8086发出的中断响应信号，有效时8259应输出中断类型码

8259A的工作过程

对8086CPU而言，响应中断申请后，向8259A回送两个负脉冲$\overline{\text{INTA}}$

第一个$\overline{\text{INTA}}$到达时，8259A做3件事：

① IRR锁存功能失效，不接受任何申请

② 使ISR相应位置位，提供判优依据（置ISRi=1）

③ 使IRR清零（置IRRi=0）

第二个$\overline{\text{INTA}}$到达时，8259A做3件事：

① 恢复IRR功能

② 将中断类型号送上数据总线的$D_7 \sim D_0$，传给CPU

③ 如果方式控制字（ICW_4），自动中断结束位为1，那么在第二个脉冲结束时，8259A会将第一个INTA脉冲到来时设置的ISR清0

8259A的工作方式

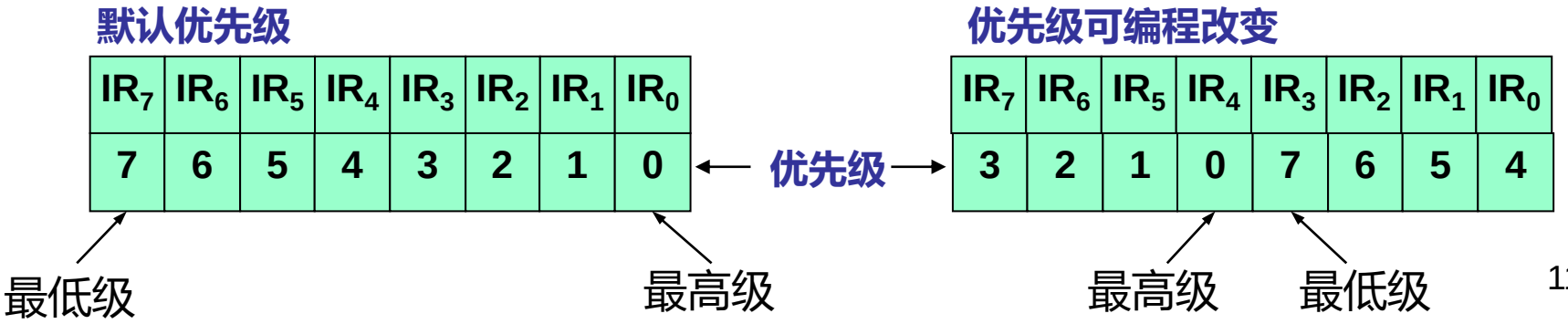
- **8259A的工作方式有如下几类：**
  - 中断优先方式与中断嵌套
  - 中断结束处理方式
  - 屏蔽中断源的方式
  - 中断触发方式
  - 级联工作方式

中断优先方式与中断嵌套

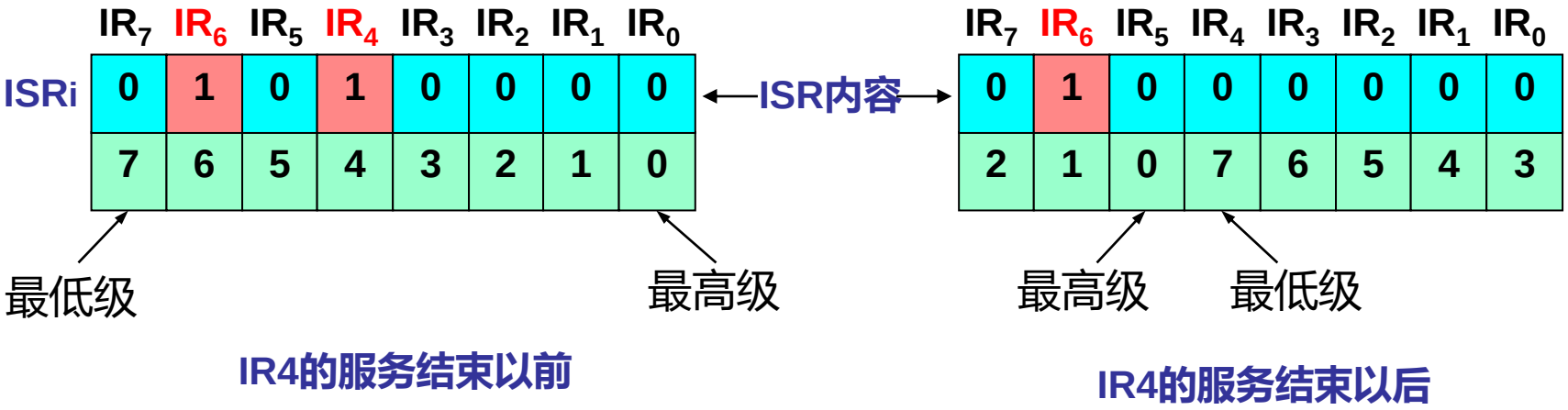
（1）中断优先方式

两类优先级控制方式：固定优先级和循环优先级

- 固定优先级方式
  - 所有中断请求IR_i的中断优先级固定不变
  - 优先级排列顺序可编程改变
  - 加电后8259A的默认方式，默认优先级顺序从高到低为IR_0～IR_7



- 循环优先级方式
  - 中断源轮流处于最高优先级，即自动中断优先级循环
  - 初始优先级顺序可用编程改变
  - 某中断请求IR_i被处理后，其优先级别自动降为最低，原来比它低一级的中断上升为最高级



IR4的服务结束以前

IR4的服务结束以后

（2）中断嵌套方式

- 在中断处理过程中允许被更高优先级的事件所中断称为中断嵌套。**8259A**有两种中断嵌套方式：

  1）普通（一般）全嵌套方式（默认方式，简称为“全嵌套方式”）

  中断正被处理时，只有更高优先级的事件可以打断当前的中断处理过程而被服务。优先级默认为：**0~7**（依次降低）。所以该模式也称为全嵌套优选权模式

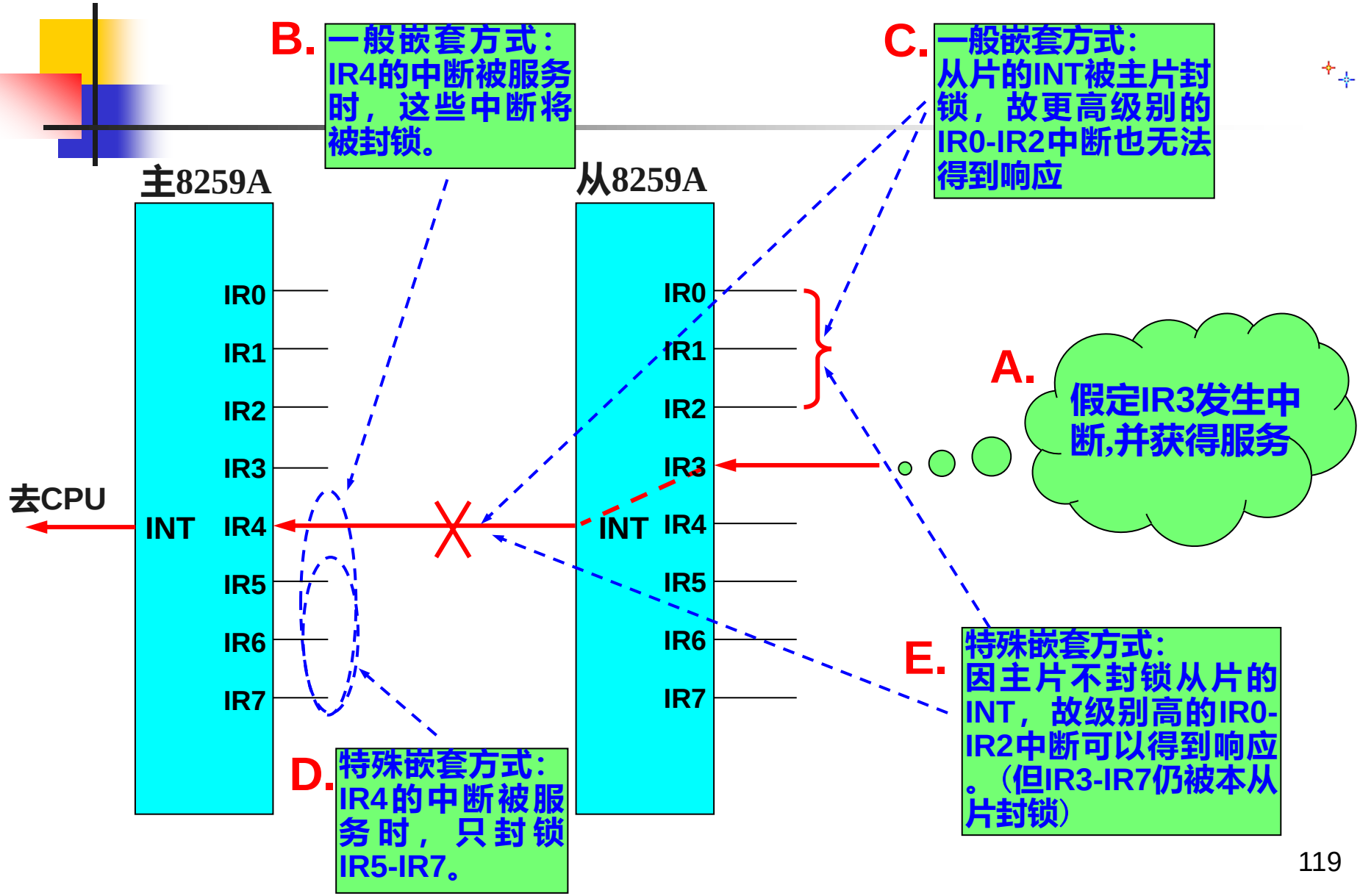
  此方式是**8259A**在初始化时默认选择的方式。其特点是：**IR0**优先级最高，**IR7**优先级最低。在**CPU**中断服务期间，若有新的中断请求到来，只允许比当前服务的优先级更高的中断请求进入，对于“同级”或“低级”的中断请求则禁止响应。

2）特殊全嵌套方式

其特点是：IR7～IR0 的优先级顺序与普通嵌套方式相同；不同之处是在CPU中断服务期间，除了允许高级别中断请求进入外，还允许同级中断请求进入，从而实现了对同级中断请求的特殊嵌套。

在多片8259A级联的情况下，主片设置为特殊完全嵌套方式，从片设置为普通嵌套方式。当主片响应某一个从片的中断请求时，从片中的IR7～IR0 的请求都是通过主片中的某个IRi请求引入的。因此从片的IR7～IR0 对于主片IRi来说，它们属于同级，只有主片工作于特殊完全嵌套方式时，从片才能实现完全嵌套。

一般全嵌套方式与特殊全嵌套方式的区别

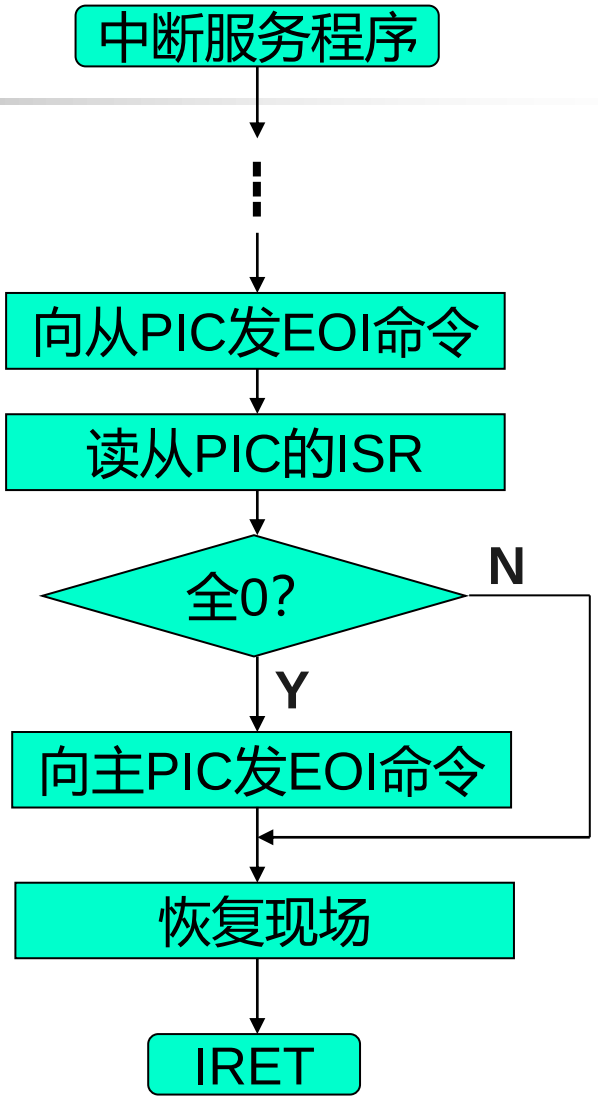


中断结束处理方式

- 当某一IR_i中断被服务时，ISR中的相应位ISR_i=1。当服务结束后，则必须清零该ISR_i位。使ISR_i=0是通过向8259A发出中断结束命令（EOI命令）实现的。
- 三种EOI命令
  - 自动EOI（AEOI）——（自动EOI方式）
  - 非指定EOI（NSEOI）——（正常EOI方式）
  - 指定EOI（SEOI）——（特殊EOI方式）

- AEOI：在第2个INTA#结束时，由8259A使ISRi自动复位（硬件自动进行）；
  - 因不保留当前正在服务的中断的状态，故AEOI不能用于中断嵌套方式
- SEOI：由CPU发出一条SEOI命令，该EOI命令中指出了所要复位的ISR的位号。
  - 用于全嵌套方式（普通全嵌套和特殊全嵌套）和非全嵌套方式（中断优先）
- NSEOI：由CPU发出正常EOI命令，该EOI命令使ISRi=1的位中优先级最高的那一位复位。
  - 用于普通全嵌套方式（配合中断优先权方式）

不管是正常中断结束方式（NSEOI），还是特殊中断结束方式（SEOI），在一个中断服务程序结束时，对于级联使用的8259a都必须发**两次中断结束命令**，一次是发给主片的，另一次是发给从片的。



屏蔽中断源的方式

- IMR屏蔽字决定了允许或禁止某位IR_i所对应的中断：IM_i=1 禁止， IM_i=0 允许。
- 普通屏蔽方式：在普通屏蔽方式中，如果将IMR某位置1，则它对应的IR_i将被屏蔽，从而使整个中断请求不能从8259a送到CPU，反之则可以。
- 特殊屏蔽方式：
  - 提供了允许较低优先级的中断能够得到响应的特殊手段。
  - 原理：使IMR的某位置1，就会同时使ISR的对应位自动复位。这样就可以使更低优先级的中断请求进入，当然未被屏蔽的更高优先级的中断请求也可以进入。特殊屏蔽方式。可以通过编程写入操作命令字OCW3来设置或取消。
  - 特殊屏蔽方式中只能用SEOI命令结束中断。

中断触发方式

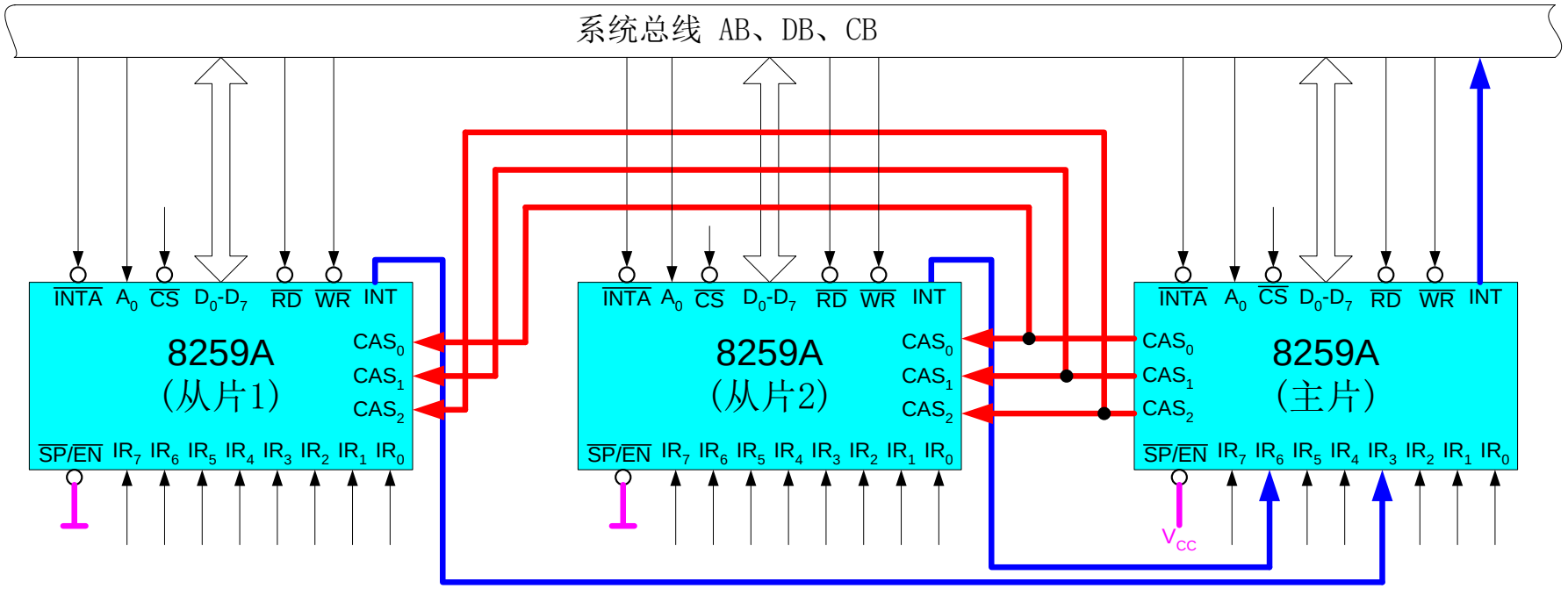
- 边沿触发
  - IR_i出现上升沿表示有中断请求
- 电平触发
  - IR_i出现高电平表示有中断请求
- 在第1个$\overline{INTA}$结束前，IR_i必须保持高电平

级联工作方式

- 单片8259A可支持8个中断源；
- 采用多片8259A级连，可最多支持64个中断源。n片8259A可支持7n+1个中断源；

  8*(n-1)+8-(n-1)=7*n+1
- 级连时只能有一片8259A为主片，其余的均为从属片；
- 涉及到的8259A引脚包括：
  - CAS_0-CAS_2
  - $\overline{SP}/\overline{EN}$
  - IRi
  - INT

级连电路连接方法


系统总线 AB、DB、CB
INTA A0 CS D0-D7 RD WR INT
8259A
(从片1)
CAS0
CAS1
CAS2
SP/EN IR7 IR6 IR5 IR4 IR3 IR2 IR1 IR0
INTA A0 CS D0-D7 RD WR INT
8259A
(从片2)
CAS0
CAS1
CAS2
SP/EN IR7 IR6 IR5 IR4 IR3 IR2 IR1 IR0
INTA A0 CS D0-D7 RD WR INT
CAS0
CAS1
CAS2
8259A
(主片)
SP/EN IR7 IR6 IR5 IR4 IR3 IR2 IR1 IR0
Vcc

1#从片挂在主片的IR3
2#从片挂在主片的IR6

- 1#从片
- IR5有申请

  IRR: 0010 0000

  ISR: 0010 0000
- 向主片发INT信号
- 从片地址与主片地址同
- 1#从片与识别码一致,1#从片送出中断类型码

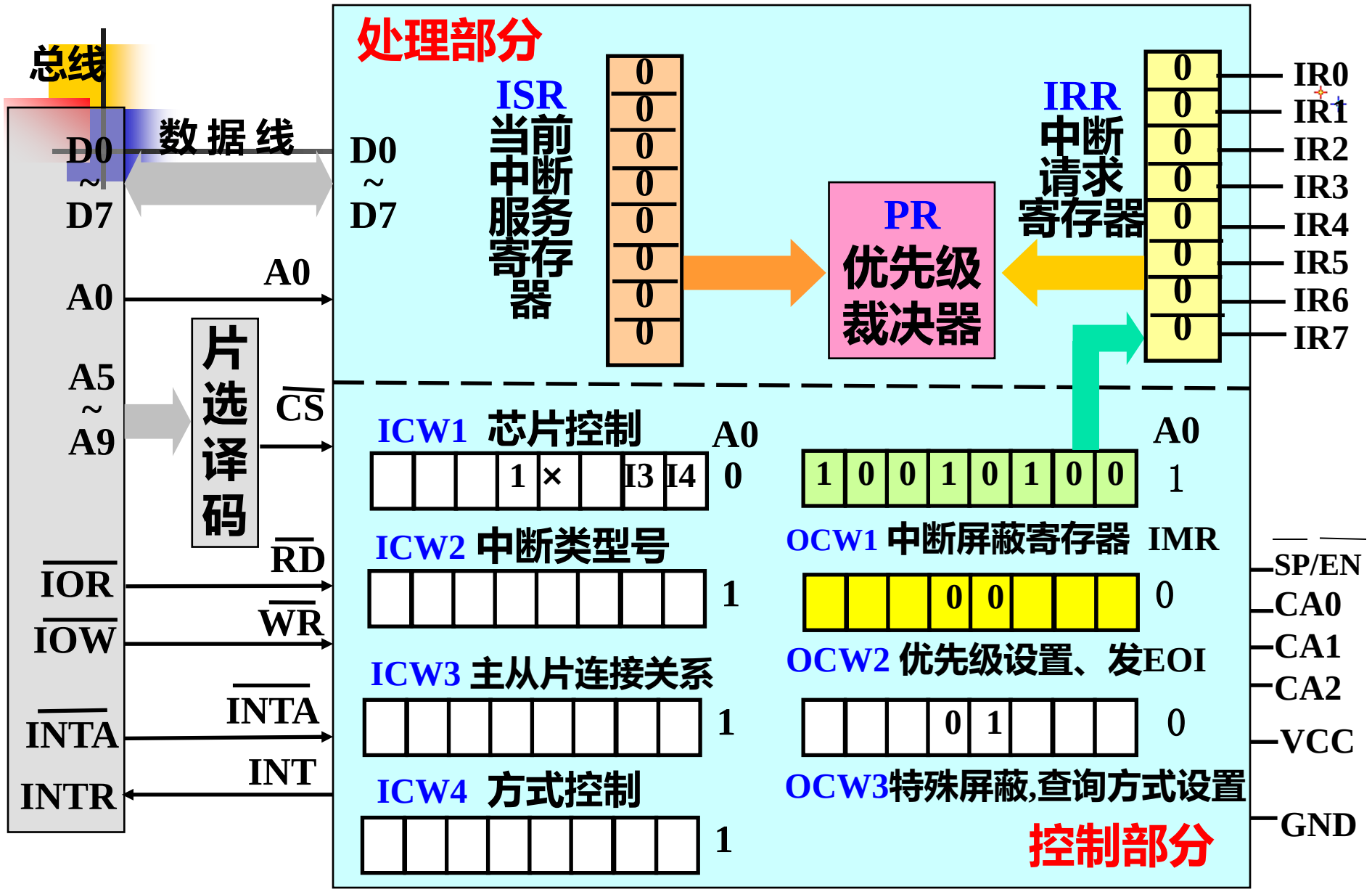
- 主片： IR3有申请

  IRR:0000 1000

  ISR:0000 1000
- 主片向CPU发INTR信号
- CPU发回答信号INTA
- 2个从片都收到INTA
- 主片向从片送识别码:

  $CAS_0 \sim CAS_2 = 011$

8259A的编程结构


处理部分
总线
数据线
D0
~
D7
A0
A5
~
A9
片选译码
RD
WR
INTA
INT
IOR
IOW
INTA
INTR
ISR
当前中断服务寄存器
PR
优先级裁决器
IRR
中断请求寄存器
IR0
IR1
IR2
IR3
IR4
IR5
IR6
IR7
ICW1 芯片控制
ICW2 中断类型号
ICW3 主从片连接关系
ICW4 方式控制
OCW1 中断屏蔽寄存器 IMR
OCW2 优先级设置、发EOI
OCW3特殊屏蔽,查询方式设置
SP/EN
CA0
CA1
CA2
VCC
GND
控制部分

8259A的编程使用

- **8259A的控制命令分为**
  - 初始化命令字**ICW**
    - **ICW1～ICW4**
    - 向**8259A**写入**ICW**的过程称为初始化编程
  - 操作命令字**OCW**
    - **OCW1～OCW3**
    - 向**8259A**写入**OCW**的过程称为操作方式编程

8259A内部寄存器的寻址方法

- 需要$\overline{CS}$、A_0、$\overline{RD}$、$\overline{WR}$和D_4、D_3的配合
- 8259A有两个I/O端口地址，寄存器的读写均通过这两个端口实现,根据写入数据的特征位和写入的先后顺序区分是对哪个寄存器进行操作

CS#	RD#	WR#	A0	D4	D3	读写操作
0	1	0	0	0	0	写OCW2
			0	0	1	写OCW3
			0	1	x	写ICW1
			1	x	x	写ICW2,ICW3,ICW4,OCW1(顺序写入)
0	0	1	0	x	x	读出IRR、ISR
			1			读出IMR

8259A的初始化顺序

- **8259的初始化流程如图**
  - 注意次序不可颠倒

初始化8259A必须从ICW1开始，
写ICW1意味着重新初始化8259A，
写入ICW1后，8259A的状态如下：

清除ISR和IMR(全0)；

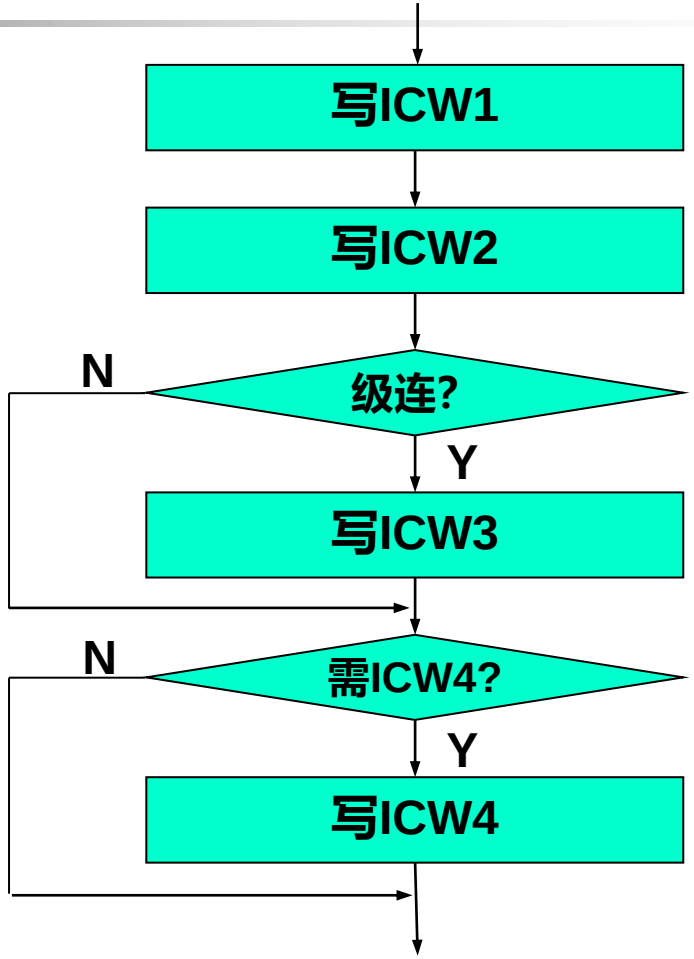
将中断优先级设成初始状态：

IR_0最高，IR_7最低；

设定为一般屏蔽方式；

采用非自动中断结束方式；

状态读出逻辑预置为读RR。



8259A的初始化命令字：ICWi

ICW1：芯片控制初始化命令字

ICW2：设置中断类型码的

ICW3：主片/从片的初始化命令字

ICW4：方式控制初始化命令字

ICW1——初始化字

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	x	x	x	1	LTIM	x	SNGL	IC4

- **LTIM: 触发方式**
  - **=1 高电平触发**
  - **=0 上升沿触发**
- **SNGL: 级连控制**
  - **=1 单片**
  - **=0 级连**
- **IC4: ICW4控制**
  - **=1 要写ICW4(对于8086系统都置1)**
  - **=0 不写ICW4（默认ICW4为全0）**

ICW2——中断向量码

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	T_7	T_6	T_5	T_4	T_3	x	x	x

- T_7～T_3: 中断向量码的高5位
- T_2～T_0: 最低3位为中断源的序号IR_n
  - 000～111分别对应IR_0～IR_7
  - 由8259A根据中断源的序号自动填入

例如：若ICW2命令字为48H，则IR_0的中断向量码为48H，IR_7的中断向量码为4FH，等等。

ICW3——级连控制字

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

- 主片的级联控制字
  - S_i=1 对应IR_i线上连接了从片

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	ID_2	ID_1	ID_0

- **从片的级联控制字**

⑩ *ID2 ~ID0 标识码，说明本从片连接到主片的哪个IR引脚上。 000 ~111分别对应IR_0 ~IR_7。*

- ICW3必须与主从片的连接关系一致：

  例如，主片的IR_4与从片的INT线连接，则主片的ICW3=10H，从片的ICW3=04H。

- 中断响应时，主片通过级连线CAS_2-CAS_0送出被允许中断的从片标识码，各从片用自己的ICW3与CAS_2-CAS_0比较，二者一致的从片才可发送中断向量码。

ICW4——中断结束方式字

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	SFNM	BUF	M/S	AEOI	1

- **SFNM: 特殊全嵌套**
  - 1 特殊全嵌套方式
  - 0 一般全嵌套方式
- **AEOI: 自动EOI**
  - 1 自动EOI方式
  - 0 非自动EOI方式
- BUF: 缓冲方式

  M/S: 主/从缓冲选择

$\overline{BUF}$	$\overline{M/S}$	
1	1	缓冲方式/主PIC
1	0	缓冲方式/从PIC
0	x	非缓冲方式/正常

两种方式（总线连接方式）：缓冲方式和非缓冲方式

① 缓冲方式

多片8259A级连的大系统中，8259A通过总线驱动器和数据总线相连，即为缓冲方式。该方式下，将8259A的SP/EN端和总线驱动器的允许端相连。此时主从片的区分不能依靠SP/EN脚，而是要通过ICW4的D2位决定。

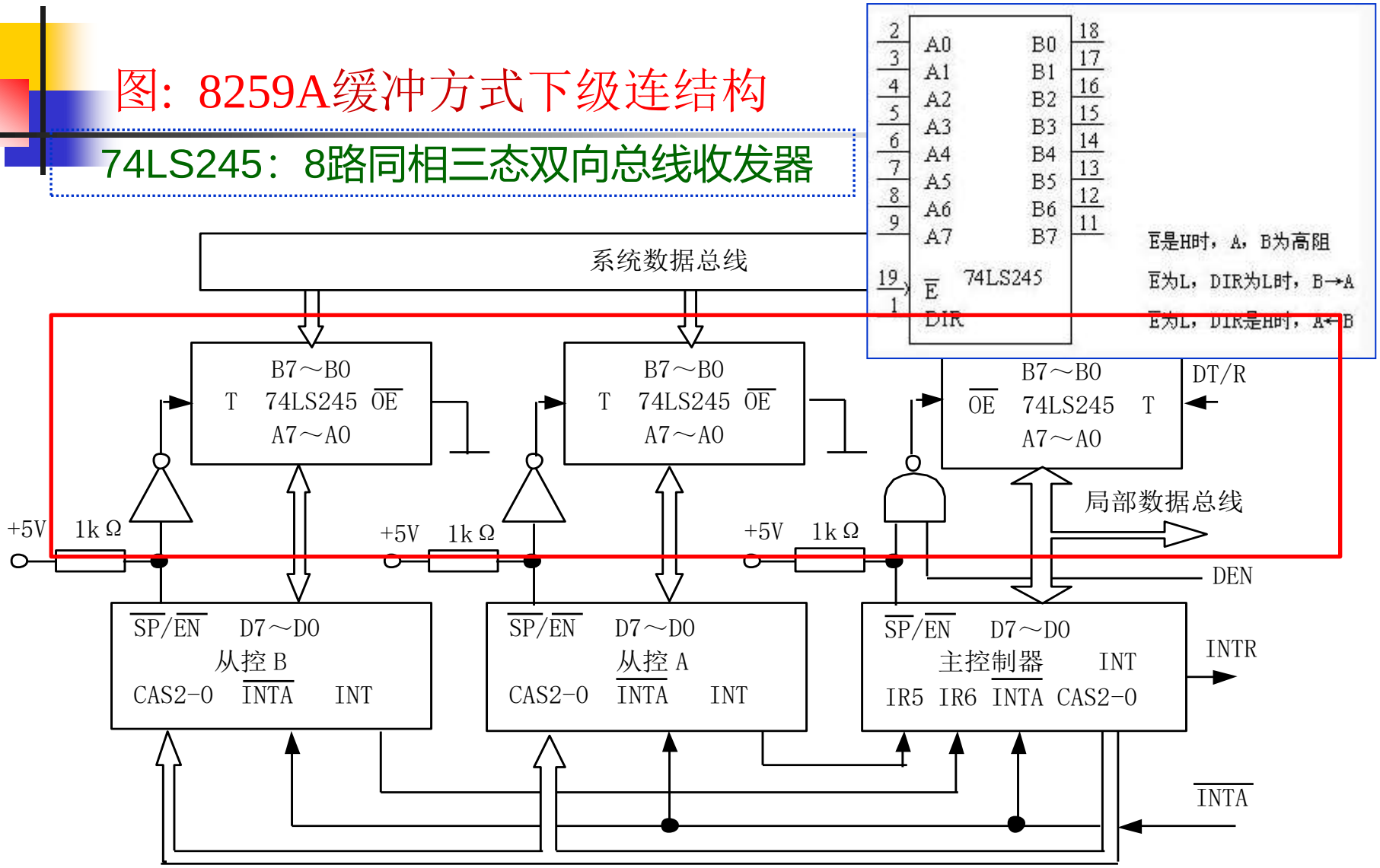
② 非缓冲方式

当系统中只有单片8259A时，将它直接与数据总线相连；在某些不太大的系统中，即使有多片8259A工作在级连方式，只要片数不多，也可将8259A直接与数据总线相连。在非缓冲方式下，8259A的 SP/EN端作为输入端。当系统中只有单片8259A时，此端必须接高电平；主从片由硬件引脚确定：主片的SP/EN端接高电平，从片的 SP/EN端接低电平。

PC机初始化为非缓冲方式

图：8259A缓冲方式下级连结构

74LS245：8路同相三态双向总线收发器



- 例. PC/XT机，CPU为8088，8259A与系统总线间采用缓冲器连接，非自动结束方式，只用1片8259A，正常完全嵌套。

  ICW4=00001101B=0DH

  MOV AL，0DH

  OUT 21H，AL

- 例. TP86单板机，CPU为8086，采用非自动结束方式，使用2片8259A，非缓冲方式，采用特殊全嵌套方式。

  ICW4＝00010001B＝11H

  MOV DX，FFDEH

  MOV AL，11H

  OUT DX，AL

8259A的操作命令字OCW

- **OCW用于设置8259的工作状态**
- **在初始化后写入**
- **OCW的写入顺序可任意**
- **写入地址要求：**
  - **OCW1必须写入奇地址端口(A_0=1)**
  - **OCW2，OCW3必须写入偶地址端口(A_0=0)**

OCW1——中断屏蔽字

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0

- **M_i=1 中断请求线IR_i被屏蔽(不允许中断)**

  =0 允许该IR_i中断

- **OCW1将写入IMR寄存器。**
- **A0=1时读OCW1可读出设置的IMR内容。**

OCW2——中断结束和优先级循环

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	R	SL	EOI	0	0	L_2	L_1	L_0

D4D3：=00，特征位

D7　：=1，中断优先级按循环方式设置；R=0，非循环方式设置。

D6　：=1，OCW2中的L2-L0设置有效，SL=0无效

D5　：=1，非自动结束标志（EOI）

EOI:中断结束命令位。EOI＝1中断结束时发送中断结束命令，使当前ISR寄存器的相应位清0。EOI=0时，不需要发送中断结束命令，即为中断自动结束方式。

L2-L0：两个作用：在D7=1的情况下，3位表达循环开始的最低优先级；中断结束方式中，指明要清除ISR寄存器的哪一位。

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	R	SL	EOI	0	0	L_2	L_1	L_0

- **01100110**
  - ——结束IR6的中断
- **11000110**
  - ——IR6为最低级开始循环方式

OCW3——屏蔽方式和读出控制字

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	ESMM	SMM	0	1	P	RR	RIS

- **ESMM:** 允许使能特殊屏蔽方式
- **SMM:** 特殊屏蔽方式

ESMM	SMM	
1	1	特殊屏蔽方式置位
1	0	特殊屏蔽方式复位
0	x	非特殊屏蔽方式

P(Polling): =1 查询方式
=0 非查询方式

- RR: 读寄存器
- RIS: ISR/IRR选择

RR	RIS	
1	1	读ISR
1	0	读IRR
0	x	无效

- 特殊屏蔽例：

```
；IR4中断处理程序
CLI
MOV   AL，68H          ；OCW3：0 1 1 0 1 0 0 0    ？
OUT   0C0H，AL         ；设置特殊屏蔽方式
IN    AL，0C2H
OR    AL，10H          ；屏蔽IR4
OUT   0C2H，AL
STI
；IR7请求，响应，返回
CLI                    ；为设命令字
IN    AL，0C2H         ；读出屏蔽字
AND   AL，0EFH         ；清除IMR4
OUT   0C2H，AL
MOV   AL，48H          ；OCW3：0 1 0 0 1 0 0 0
OUT   0C0H，AL         ；取消特殊屏蔽
STI
；继续IR4中断服务
 MOV   AL，20H         ；OCW2：0 0 1 0 0 0 0 0 (EOI)
 OUT   0C0，AL
IRET
```

- 查询方式允许8259A不工作于中断方式，而是以查询方式工作。
  - CPU先写一个D_2=1的OCW3，再对同一地址读入，即可得到如下状态字节：

    I x x x x R_2 R_1 R_0

    I=1表示有中断请求，中断请求号为R_2-R_0

    此查询步骤可反复执行，以响应多个同时发生的中断。

- 中断查询方式中，8259A不通过INT信号向CPU发中断请求信号，因为CPU内部的中断允许寄存器复位，所以禁止了8259A对CPU的中断请求。CPU要使用软件查询来确定中断源，才能实现对外设的中断服务。因此，中断查询方式既有中断的特点，又有查询特点。
- 中断查询方式一般用在多于64级中断的场合，或者用在一个中断服务程序中几个模块分别为几个中断设备服务的情况。

- CPU所执行的查询程序应包含如下过程：

  ①系统关中断。

  ②用OUT指令使CPU向8259A端口（偶端口）送OCW3命令字。

  ③若外设已发出过中断请求，8259A在当前中断服务寄存器中使对应位置"1"，且立即组成查询字。

  ④CPU用IN指令从端口（偶端口）读取8259A的查询字。

例1

- 若该8259A为主片，请用查询方式，查询哪个从片有中断请求。

```
MOV  AL, 00001100B ;发OCW3,欲查询是否有中断请求
OUT  20H,AL
IN  AL, 20H              ;读查询字，并判断最高位是否为1
TEST  AL, 80H
JZ  DONE
AND AL,07H               ;判断中断源的编码
                          (低三位为最高优先级请求的序号 )
……
DONE:  HLT
```

说明：查询字的结构

D_7：=1表示有中断

$D_2 \sim D_1$：最高优先级序号

8259A编程举例

- 按以下要求初始化8259A：
  - 接口地址为20H和21H；
  - 中断为上升沿触发；单片8259A；不写ICW4；
  - 与IR_0-IR_3对应的中断向量码为08H-0BH；
  - IR_4-IR_7不使用。

ICW1

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	x	x	x	1	LTIM	x	SNGL	IC4

ICW2

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	T_7	T_6	T_5	T_4	T_3	x	x	x

OCW1

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0

根据要求，各初始化参数及工作参数如下：

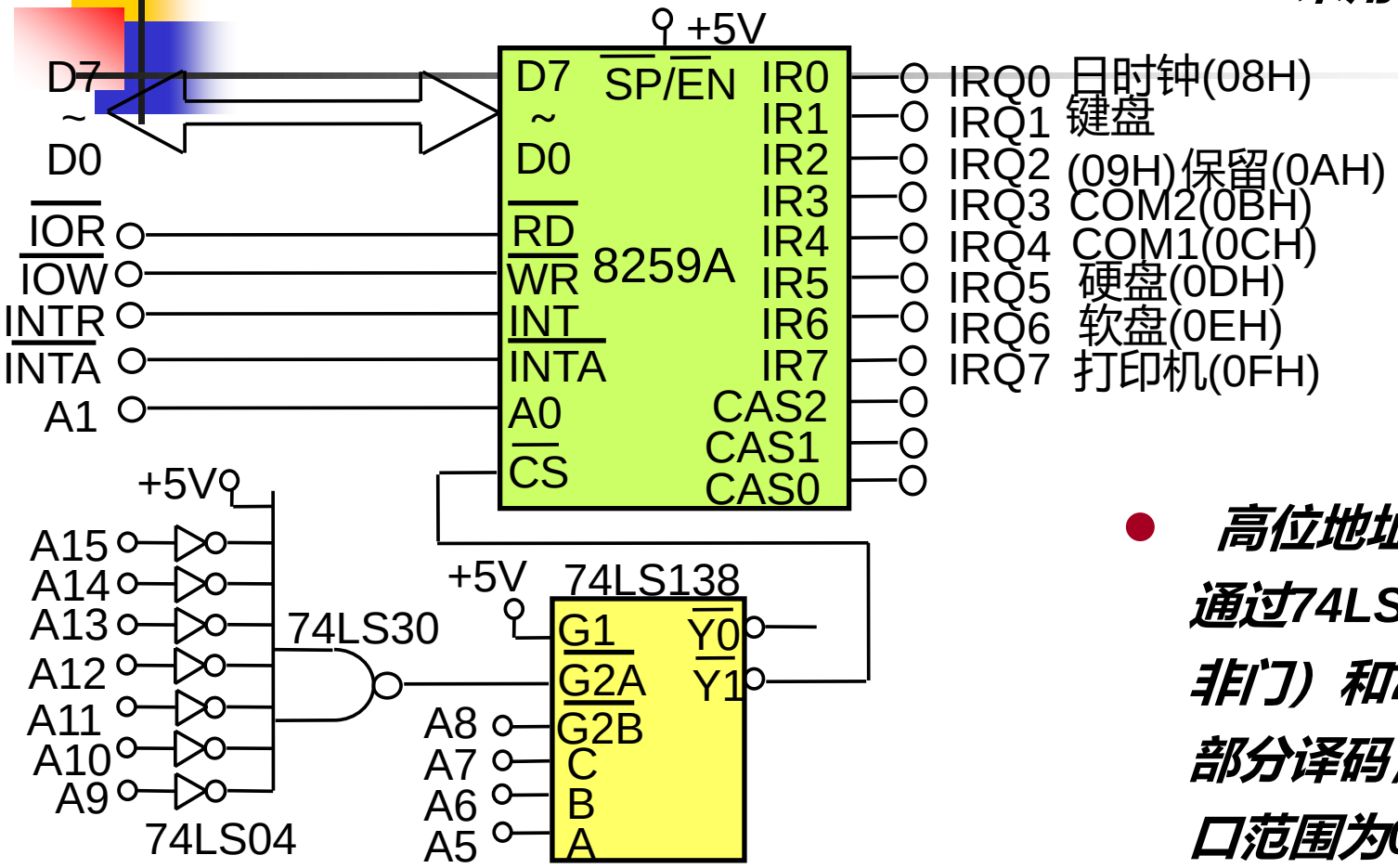
ICW1 = 00010010 = 12H

ICW2 = 08H ***中断向量码***

OCW1 = 11110000 = 0F0H ***中断屏蔽字***

8259A与计算机的连接 8259与计算机的连接

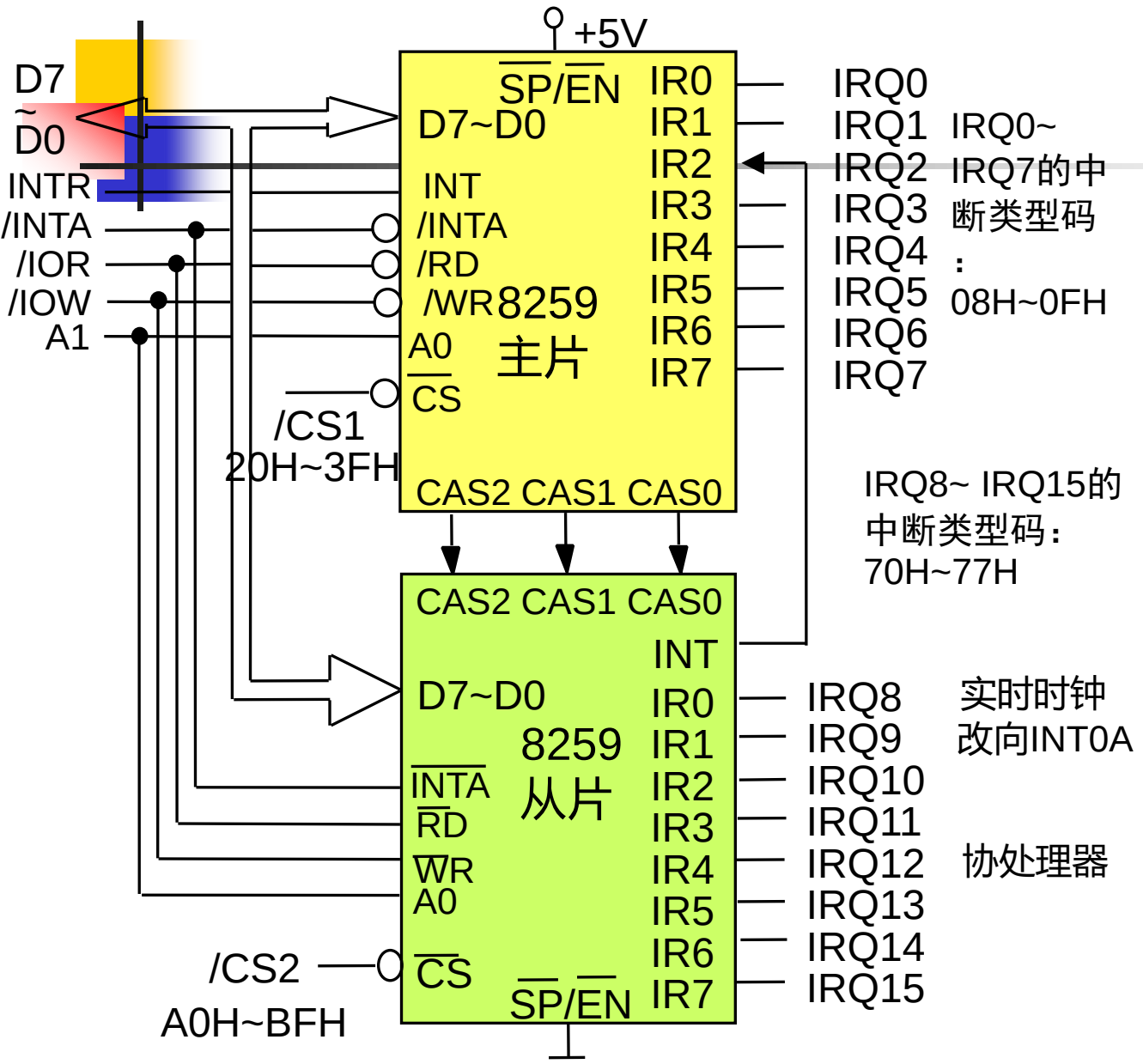
8259A单片工作方式的接口电路



- ***8259采用D7~D0传送数据***
- 低位地址线A1接8259的A0引脚，用于选择8259内部寄存器
- ***高位地址线A15~A5，通过74LS30（8输入与非门）和74LS138进行部分译码，控制CS，端口范围为020H~03FH，常用20H和22H两个偶端口***

A15 A14 A13 A12 A11 A10 A9 A8	A7 A6 A5	A4 A3 A2 A1 A0	范　围
0 0 0 0 0 0 0 0	0 0 1	X X X X X	0020H～003FH

两片8259级联管理15级中断



- 收到第一个$\overline{\text{INTA}}$时，主片通过CAS2~CAS0通知有中断申请的从片在第二个$\overline{\text{INTA}}$时送出中断类型码
- 从片8259A的中断申请端INT接主片的IR2
- 主片的$\overline{\text{SP/EN}}$接+5V,从片的SP/EN接地，主从片的CAS2、CAS1、CAS0对应连接

8259与计算机的连接

※ 说明

从片8259A的中断申请端INT接主片的IRQ2

主片的SP/EN接+5V,从片的SP/EN接地，主从片的CAS2、CAS1、CAS0对应连接。

主片的端口地址INTA1、CS为020H~03FH, 从片的端口地址

INTA2、CS为0A0~0BFH.

A0接在8086的那根地址线上？

8086系统主从芯片都用偶地址传递信息。

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	范围
主芯片	0	0	0	0	0	0	0	0	0	0	1	X	X	X	X	X	0020H～003FH
从芯片	0	0	0	0	0	0	0	0	1	0	1	X	X	X	X	X	00A0H～00BFH

例1

某系统分配给8259A的端口地址为20H和21H，中断类型号为40H~47H，初始化要求：高电平触发方式、缓冲器方式，中断结束采用EOI方式，采用全嵌套方式。

◆主程序初始化编程：

```
......
MOV   AL，1BH
OUT   20H，AL
MOV   AL，40H
OUT   21H，AL
MOV   AL，09H
OUT   21H，AL
......
```

◆中断服务程序：

```
......
MOV   AL，20H
OUT   20H，AL
IRET
```

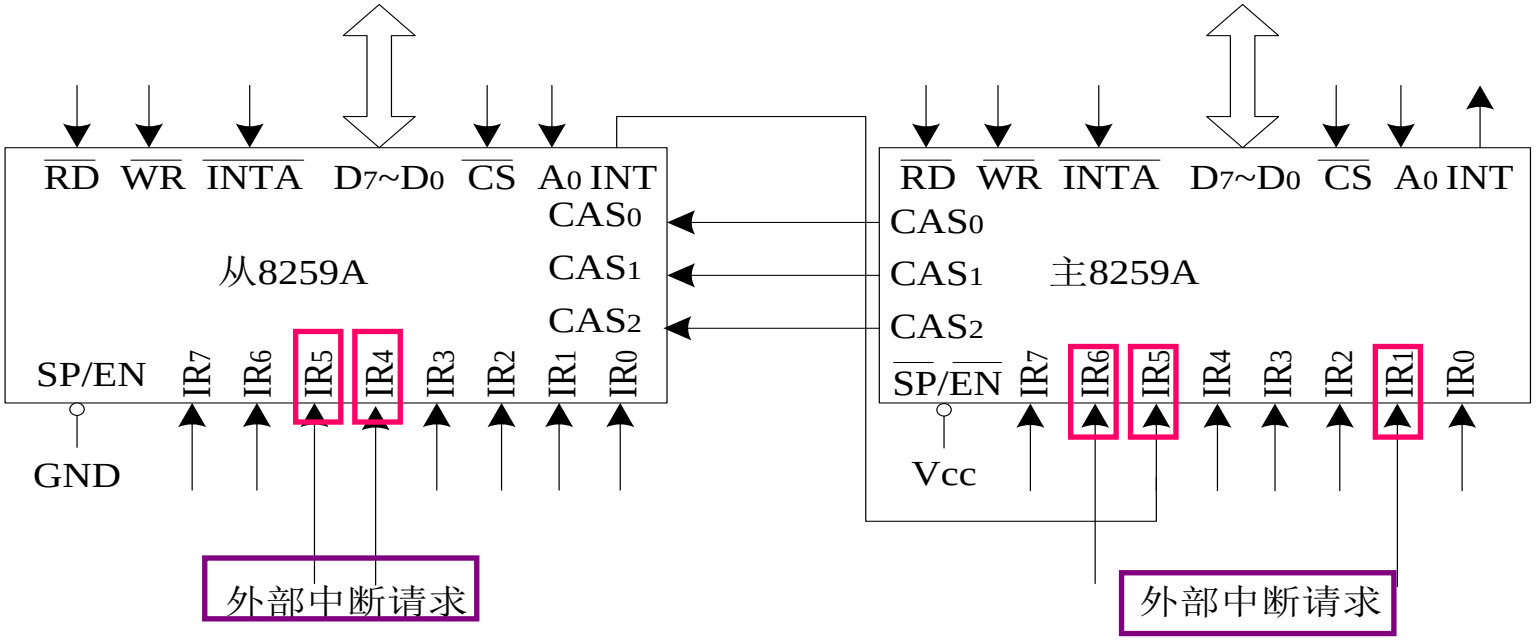
例2

读8259A相关寄存器的内容。设8259A的端口地址为20H、21H，请读入IRR、ISR、IMR寄存器的内容，并相继保存在数据段2000H开始的内存单元中。

```
MOV AL, 00001010B       ;发OCW3,读取IRR命令
OUT  20H, AL
IN  AL, 20H             ;读入并保存IRR的内容
  MOV [2000H], AL
  MOV AL,00001011B      ;发OCW3,读取ISR命令
  OUT  20H, AL
  IN  AL, 20H           ;读入并保存ISR的内容
  MOV [2001H], AL
  IN  AL, 21H           ;直接读入并保存IMR的内容
  MOV [2002H], AL
```

例3

- 【例】某级联系统，从片INT引脚接到主片IR_5引脚。
  - 主片：IR_1、IR_6引入中断请求，中断类型号31H和36H。中断向量2000H:1000H和2000H:6000H。
  - 从片：IR_4、IR_5引入中断请求，中断类型号44H和45H。向量3000H:3500H和3000H:4500H。



- 主片初始化

```
MOV  AL，19H   ；ICW1：级联连接，高电平触发
OUT  80H，AL
MOV  AL，30H   ；ICW2：中断类型号为30H
OUT  81H，AL
MOV  AL，20H   ；ICW3：从片连接在IR5引脚
OUT  81H，AL
MOV  AL，11H   ；ICW4：特殊全嵌套方式，非缓冲方式
OUT  81H，AL
MOV  AL，9DH   ；OCW1：允许IR1、IR5和IR6中断
OUT  81H，AL
```

- 从片初始化

```
MOV  AL，11H   ；ICW1：级联连接，边沿触发
OUT  90H，AL
MOV  AL，40H   ；ICW2：中断类型号为40H
OUT  91H，AL
MOV  AL，05H   ；ICW3：从片连接在主片的IR5引脚
OUT  91H，AL
MOV  AL，01H   ；ICW4：全嵌套方式，非缓冲方式
OUT  91H，AL
MOV  AL，0CFH  ；OCW1：允许IR4、IR5中断
OUT  91H，AL
```

例4

- 2片8259级联，从片的INT接主片的IR2。
- 端口地址：主片20H、21H，从片A0H、A1H。
- 主片和从片均采用边沿触发。
- 主片采用特殊全嵌套方式，从片是一般全嵌套方式。
- 采用非缓冲方式，主片接+5V,从片接地。
- 主片的中断类型号为08H~0FH，从片的中断类型号为70H~77H。
- 写出主8259A和从8259A的初始化程序段。

```
ICW1A EQU 20H          ；主片端口地址
ICW2A EQU ICW1A+1
ICW3A EQU ICW2A
ICW4A EQU ICW2A
ICW1B EQU 0A0H         ；从片端口地址
ICW2B EQU ICW1B+1
ICW3B EQU ICW2B
ICW4B EQU ICW2B
```

- **主片初始化**

```
MOV AL，11H　；ICW1，边沿触发，多片，需ICW4
OUT ICW1A，AL
MOV AL，08H　；ICW2，中断类型码
OUT ICW2A，AL
MOV AL，04H　；ICW3，IR2接从片
OUT ICW3A，AL
MOV AL，11H　；ICW4，非缓冲，特殊全嵌套，非自动结
    束
OUT ICW4A，AL
OUT ICW4B，AL
```

- **从片初始化**

```
MOV AL，11H        ；ICW1，边沿触发，多片，需
    ICW4
OUT ICW1B，AL
MOV AL，70H        ；ICW2，中断类型码
OUT ICW2B，AL
MOV AL，02H        ；ICW3，INT接主片的IR2
OUT ICW3B，AL
MOV AL，01H        ；ICW4，非缓冲，全嵌套，非自
```

PC机中断程序设计

1）主程序

保存原中断向量

设置自己的中断向量

初始化堆栈指针

设置8259A的中断屏蔽字（其他的已由操作系统设置）

STI

2）中断服务程序

保存所用到的寄存器内容

中断服务程序主体

恢复进入时保存的寄存器内容

发EOI命令（必要时可提前）

STI（必要时可提前）

IRET

例1

- PC机中主8259A的I/O地址为20H和21H，中断向量码为08H；
- 中断请求从IR_6引入，故中断类型号为0EH；
- 主程序如下：

```
;保存原来的中断向量
mov     ah，35h
mov     al，0eh          ；中断类型码
int     21h              ；取原中断向量（在es:bx中）
mov     save_ip，bx      ；保存原来的中断向量
mov     save_cs，es
```

```
;设置自己的中断向量
push    ds
mov     dx，offset  my_int
mov     ax，seg my_int
mov     ds，ax              ；DS:DX的内容为my_int的首地址
mov     ah，25h
mov     al，0eh
int     21h                 ；设置自己的中断向量
pop     ds
;初始化堆栈指针
mov     sp，my_stack_top
;设置8259A的中断屏蔽字
in      al，21h
and     al，10111111b       ；开放IR6中断
out     21h，al
sti
……
<其他代码放在这里>
……
```

子程序如下：

```
my_int      proc    far
            pusha
            …
            <中断服务程序主体>
            …
            popa
            mov     al，20h；EOI命令，00100000B
            out     20h，al；写OCW2
            sti
            iret
my_int      endp
```

例2

8259A的IRQ0(中断类型码为08H)的中断请求来自定时器8253，每隔55ms产生一次。DOS系统利用它来实现时钟计时功能。本程序用新的08H中断服务程序暂时替代计时程序，使得每次中断显示一串信息，即显示“A 8259A Interrupt !”，显示20次后，恢复中断服务程序，返回DOS。

DOS系统提供中断信息处理功能 int 21H

获得中断向量——利用DOS功能调用35H号子功能

入口参数 ***AH=35H，AL=中断类型码***

返回参数 ***ES:BX 中断服务程序入口地址***

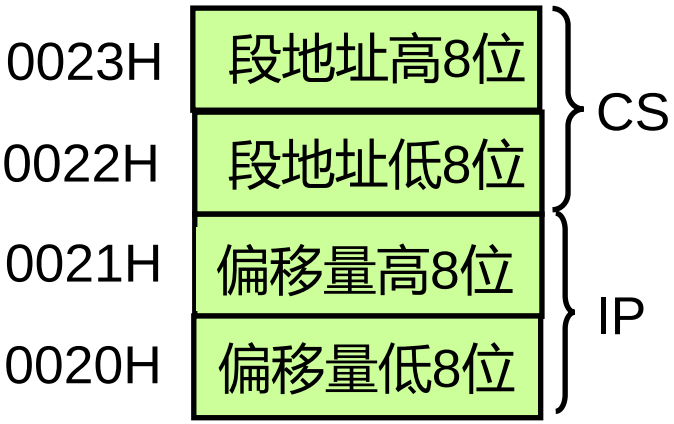
中断向量安装——利用DOS功能调用25H号子功能

入口参数 ***AH=25H，AL=中断类型码***

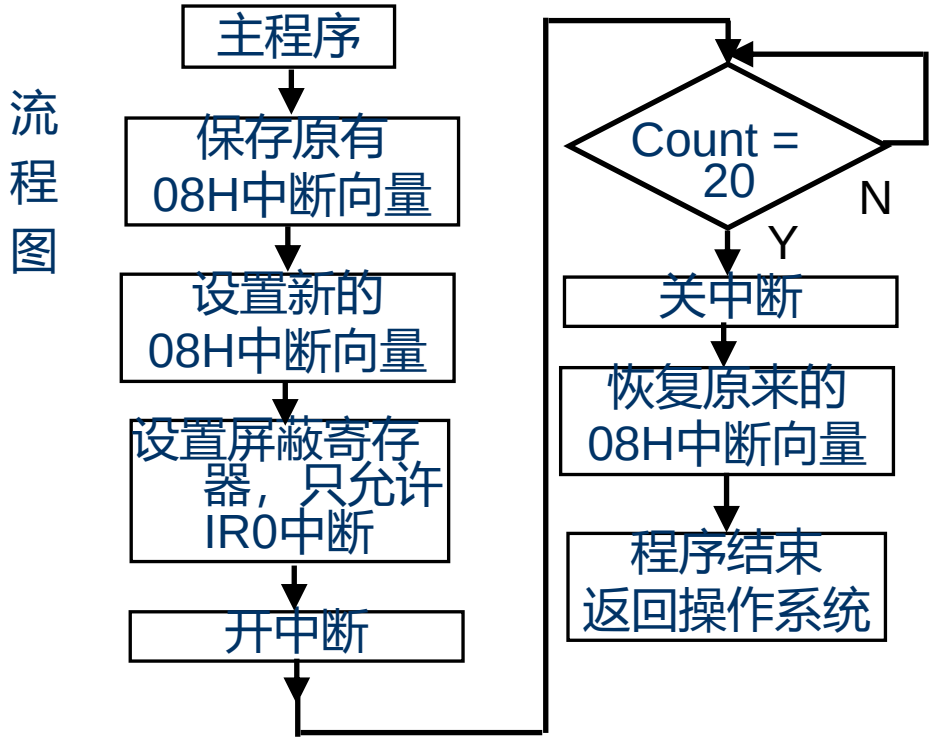
DS:DX 中断服务程序入口地址

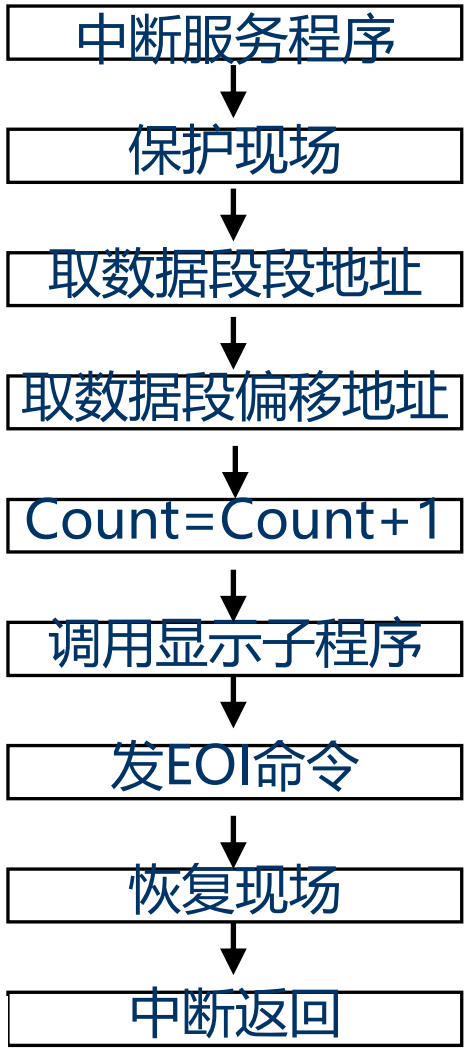
返回参数：无

中断向量(类型码08H)



流程图





保存原有08H

利用DOS功能调用35H号子功能

AH=35H，AL=中断类型码

ES:BX 中断服务程序入口地址

```
mov ax,3508h;获取原中断向量
int 21h
push es
push bx
```

设置新的08H

利用DOS功能调用25H号子功能

AH=25H，AL=中断类型码

DS:DX 中断服务程序入口地址

```
mov ax,seg int08h;
mov ds,ax;设置新中断向量的段地址
mov dx,offset int08h;设新中断向量偏移地址
mov ax,2508h
int 21h
```

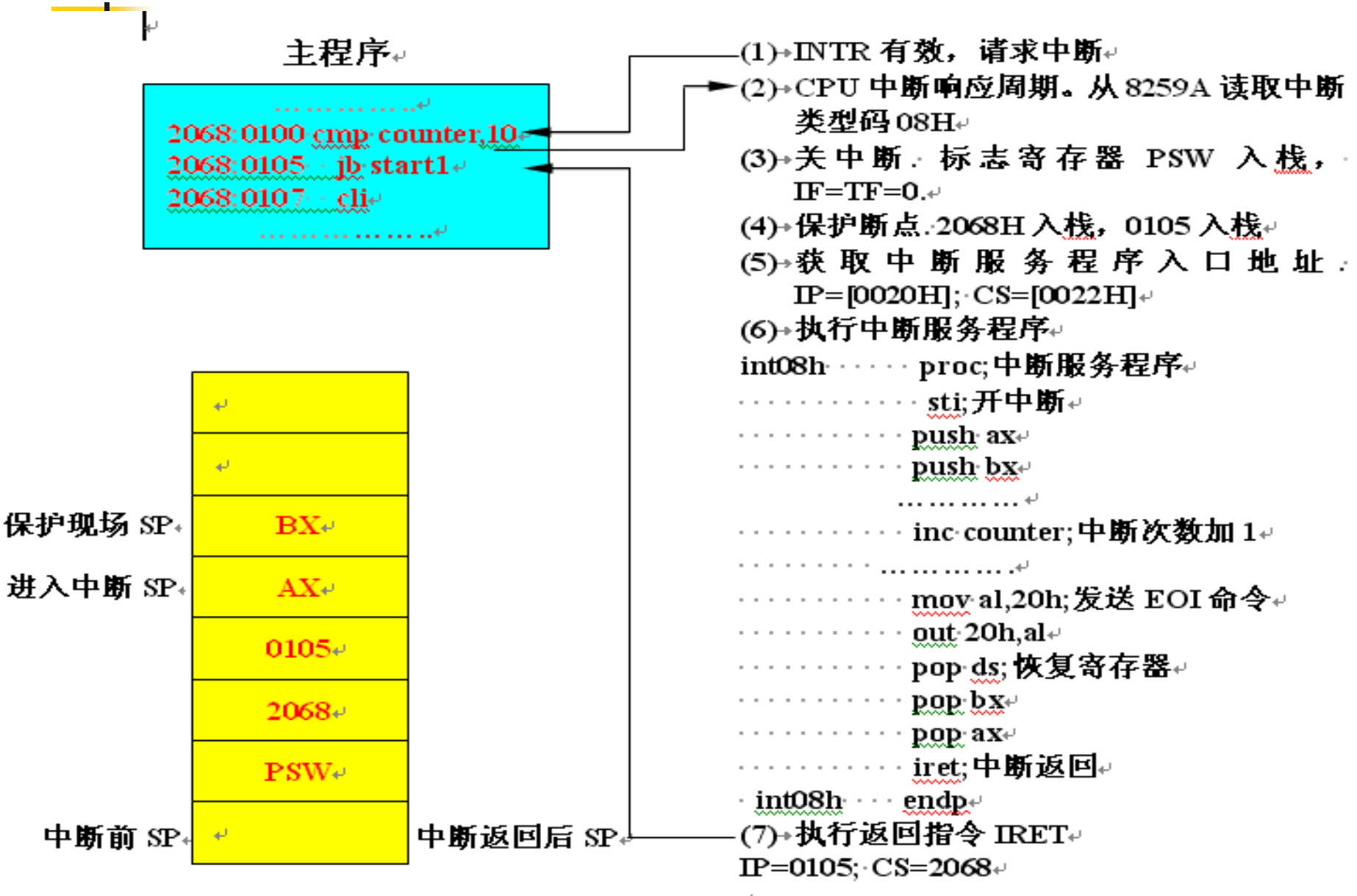
```
data        segment  para
intmsg      db 'A 8259A Interrupt !', 0dh,0ah,0
counter     db 0
data        ends
st_seg      segment stack
            db 256 dup (?)
st_seg      ends
code        segment
            assume cs:code,ds:data,ss:st_seg
main    proc far
            mov ax,data
            mov ds,ax
            mov ax,3508h;获取原中断向量
            int 21h
            push es
            push bx
            cli
            push ds；设置新中断向量
            mov ax,seg int08h
            mov ds,ax；设置新中断向量的段地址
            mov dx,offset int08h；设新中断向量
                                ；偏移地址
            mov ax,2508h
            int 21h
            pop ds
            in al, 21h；读出IMR
            push ax；保存IMR内容
            and al,0feh；允许IRO中断，其他屏蔽
                    out 21h,al;
        mov counter,0；设置中
                    ；断次数初值
        sti
start1: cmp counter,20；主程序
                    ；仅循环等待中断
            jb start1;中断20次退出
            cli
            pop ax
            out 21h,al
            pop dx；恢复原中断向量
            pop ds
            mov ax,2508h
            int 21h
            mov ax,4c00h；返回DOs
            int 21h；主程序结束
main    endp
```

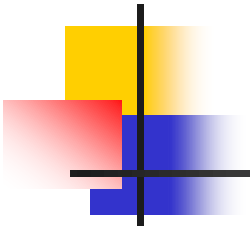
```
int08h  proc；中断服务程序
        sti；开中断
        push ax
        push bx
        push ds
        mov ax,data；取数据段的段地
                  ；址
        mov ds,ax
        inc counter；中断次数加1
        mov si,offset intmsg；显示
                  ；信息
        call dpstri；进行显示
        mov al,20h ；发送EOI命令
        out 20h,al
        pop ds     ；恢复寄存器
        pop bx
        pop ax
        iret       ；中断返回
int08h  endp
```

```
dpstri  proc
        push ax;
        push bx;
dps1: lodsb;  AL ← DS : [SI],
            ;  SI←SI±1
        cmp al,0
        jz dps2
        mov bx,0;  字符颜色
        mov ah,0eh;  调用int 10显
            ;  示al的字符
        int 10h
        jmp dps1
dps2: pop bx
        pop ax
        ret
dpstri  endp

code  ends
  end main
```

从主程序标号 start1 的语句来看（假设其逻辑地址为 2068：0100），counter 单元似乎没有变化，像是一个“死循环”。实际上，由于系统每隔 55ms 会请求一次 IRQ0 中断，现在随时可以响应，这样执行这里的 int08h 中断服务程序就使 counter 增量（每执行一次中断服务程序就加 1），主程序就在这个不断比较、转移中经过 20 次中断后退出。

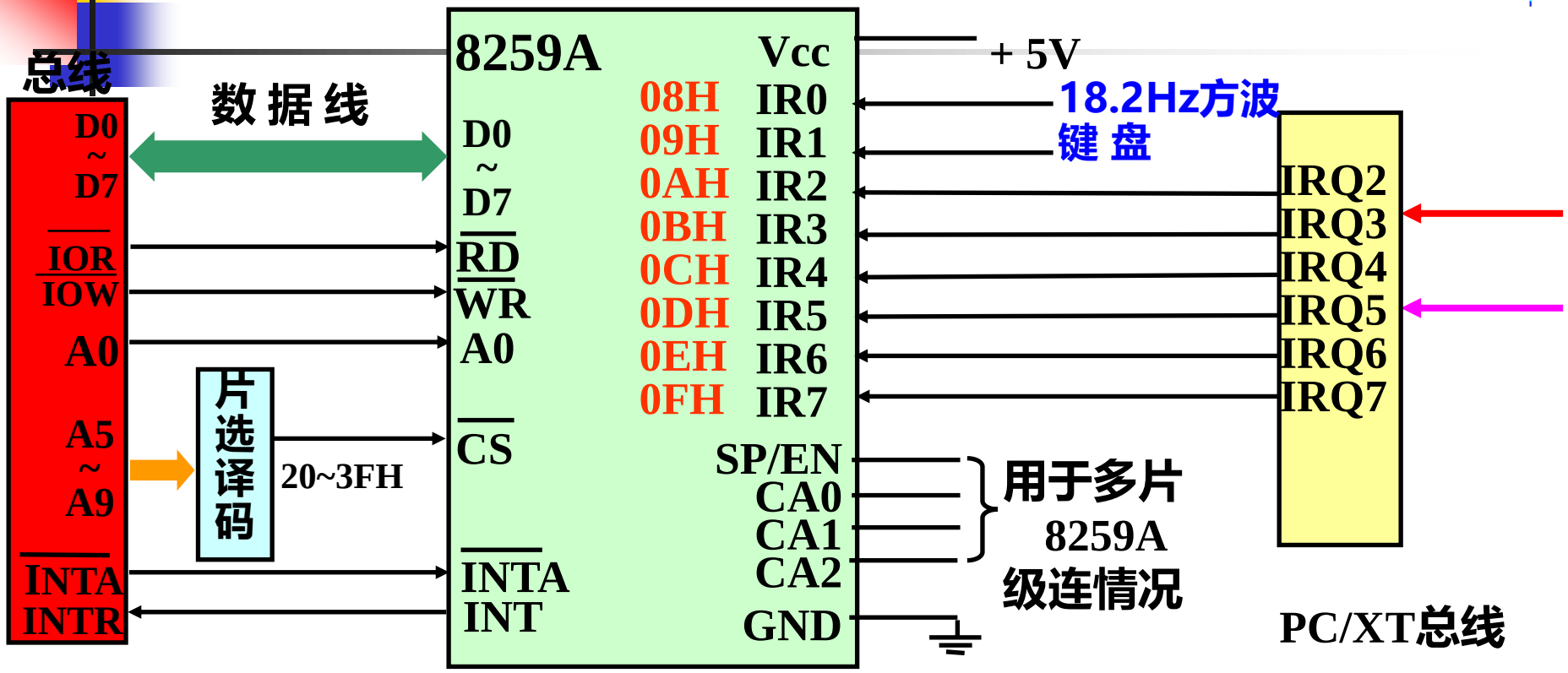




可屏蔽硬中断程序设计

- 了解IBM PC/XT系统可屏蔽硬中断的响应过程,
- 据连线确定外设中断申请对应的中断类型号;主程序做好外设发出中断申请，CPU能够响应的准备工作;
- 编写可屏蔽硬中断子程，完成中断源请求的任务.

- *了解IBM PC/XT系统可屏蔽硬中断的响应过程,*
- *根据连线确定外设中断申请对应的中断类型号。*



例 接口的中断申请连 IRQ3 类型号0BH

接口的中断申请连 IRQ5 类型号0DH

☑ 主程中做好外设发出中断申请，CPU能够响应的准备工作

- 在主程序中应做的工作：
  - ① CPU关中断　　（ 用CLI置 IF=0 ）
  - ② 保存原中断向量 (用35H系统功能)
  - ③ 设置中断向量　(用25H系统功能)
  - ④ 设置8259A的中断屏蔽字，使21H端口对应位为0，允许中断申请进入优先级裁决器
  - ⑤ CPU开中断　　(用STI置IF=1)
- 准备工作做好后，此后若该级有中断申请，则CPU响应中断，执行相应类型的中断子程序。
- 主程在返回DOS前，应恢复原中断向量。(用25H系统功能)

编写硬中断子程序，完成中断源请求的任务

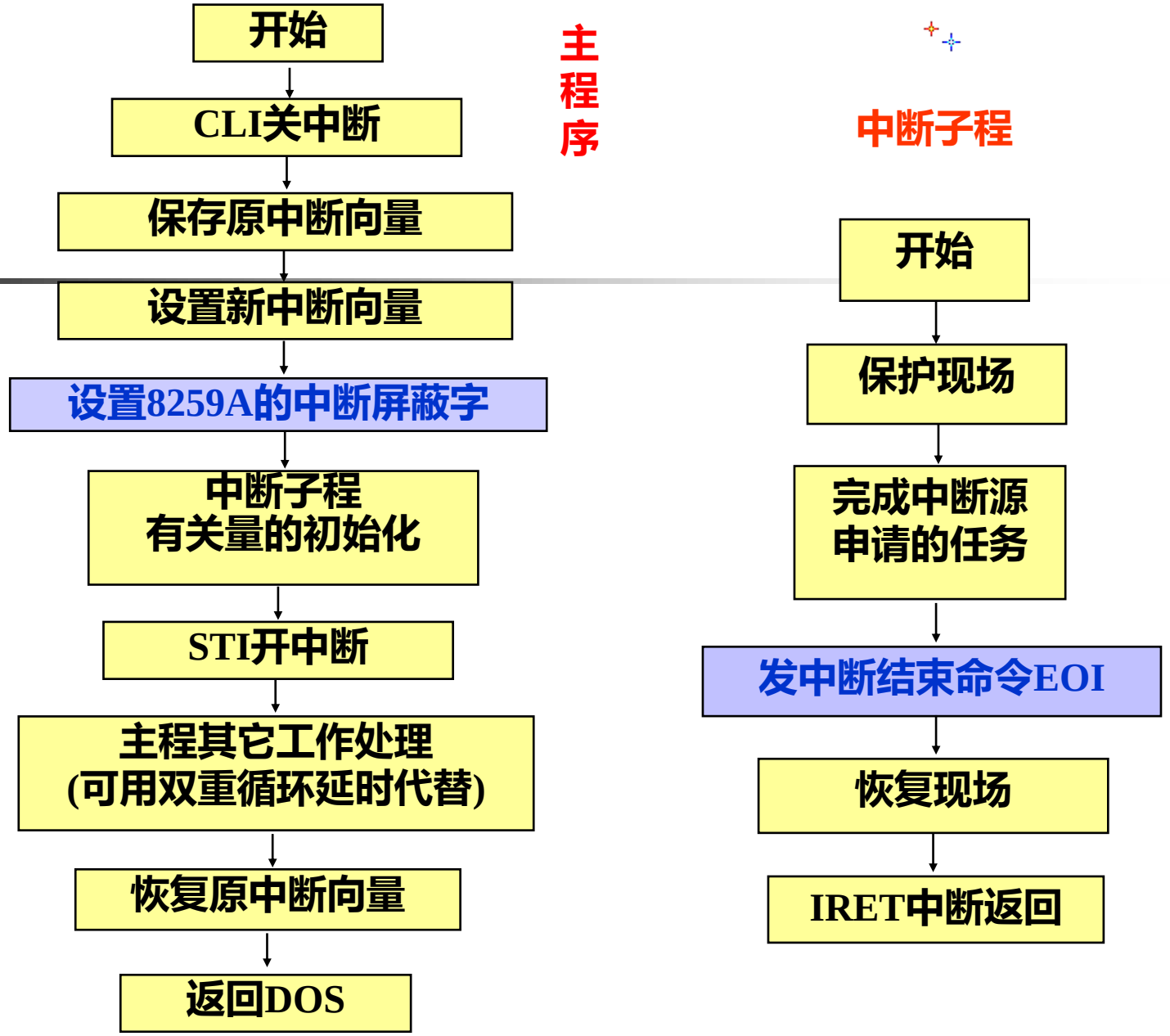
① 处理中断源申请任务部分的编写与普通子程类似

② 在中断子程结束前，

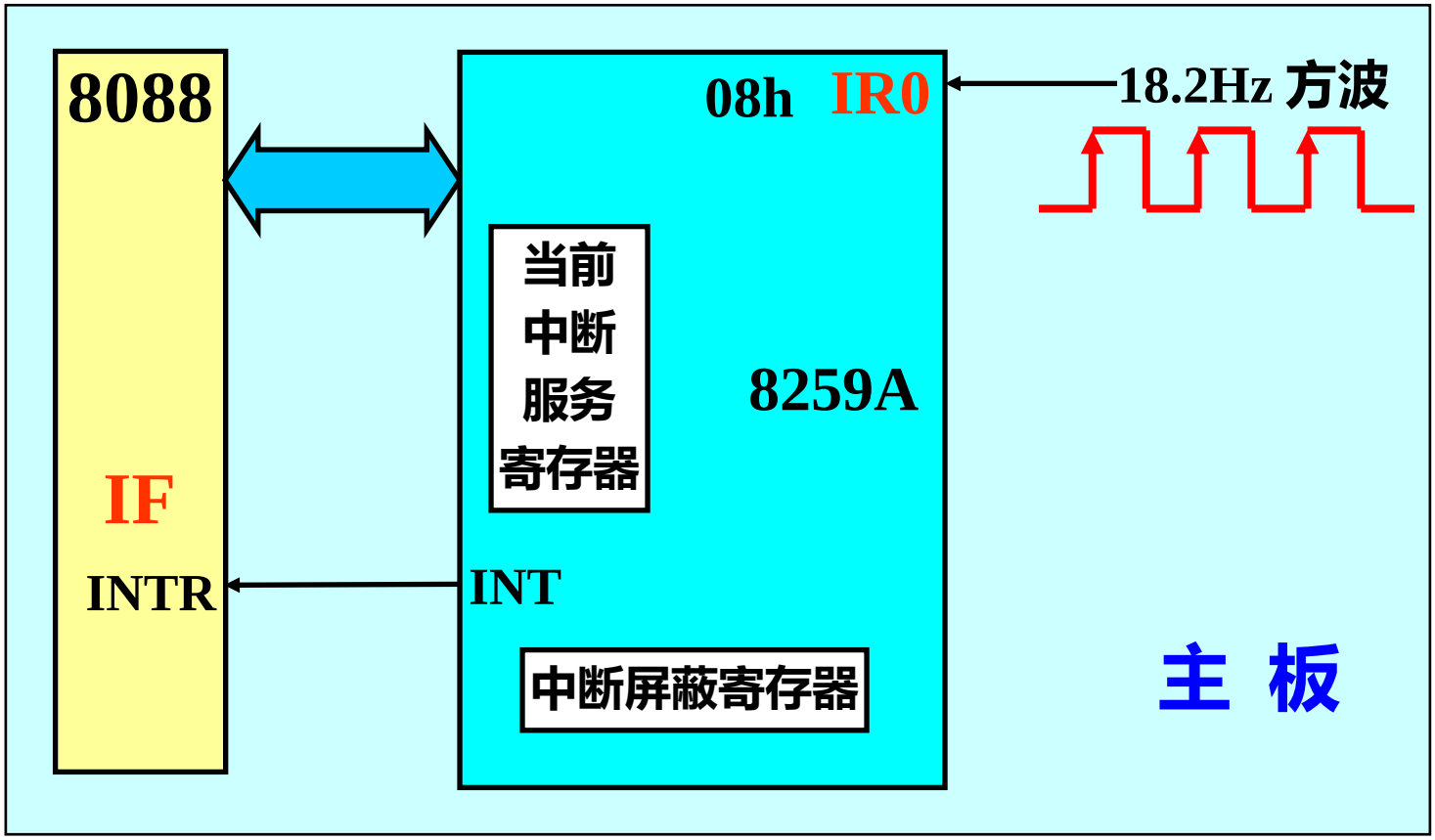
- 发中断结束命令清除 8259A 中ISR的记录；
- 否则，响应一次中断后,同级中断和低级中断将被优先级裁决器屏蔽

③ 用IRET中断返回指令返回主程被中断处

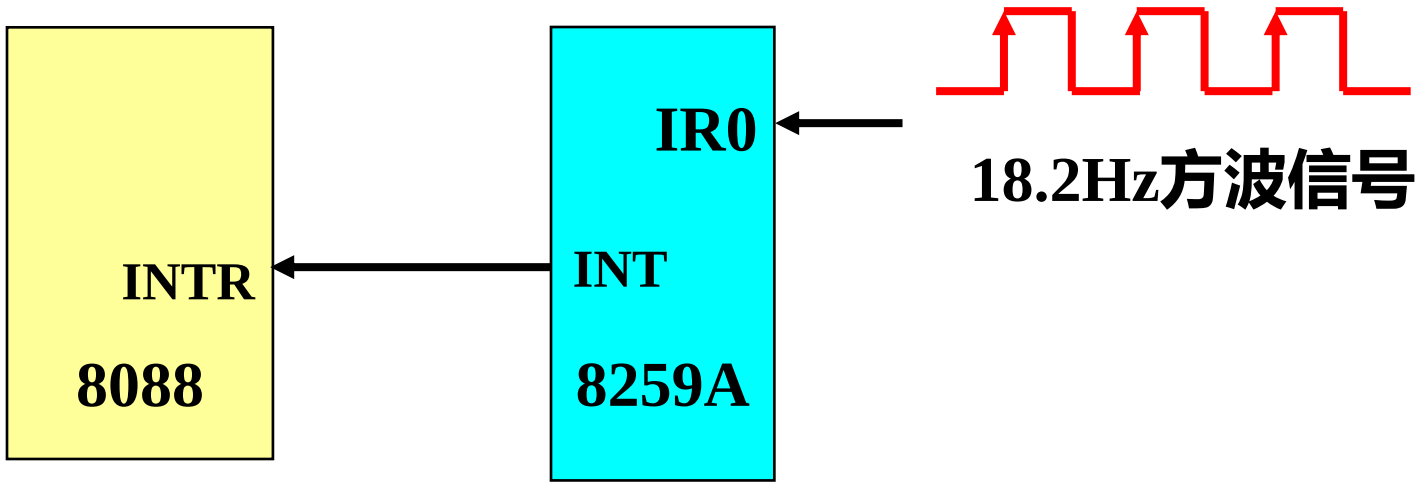
可屏蔽硬中断程序流程图



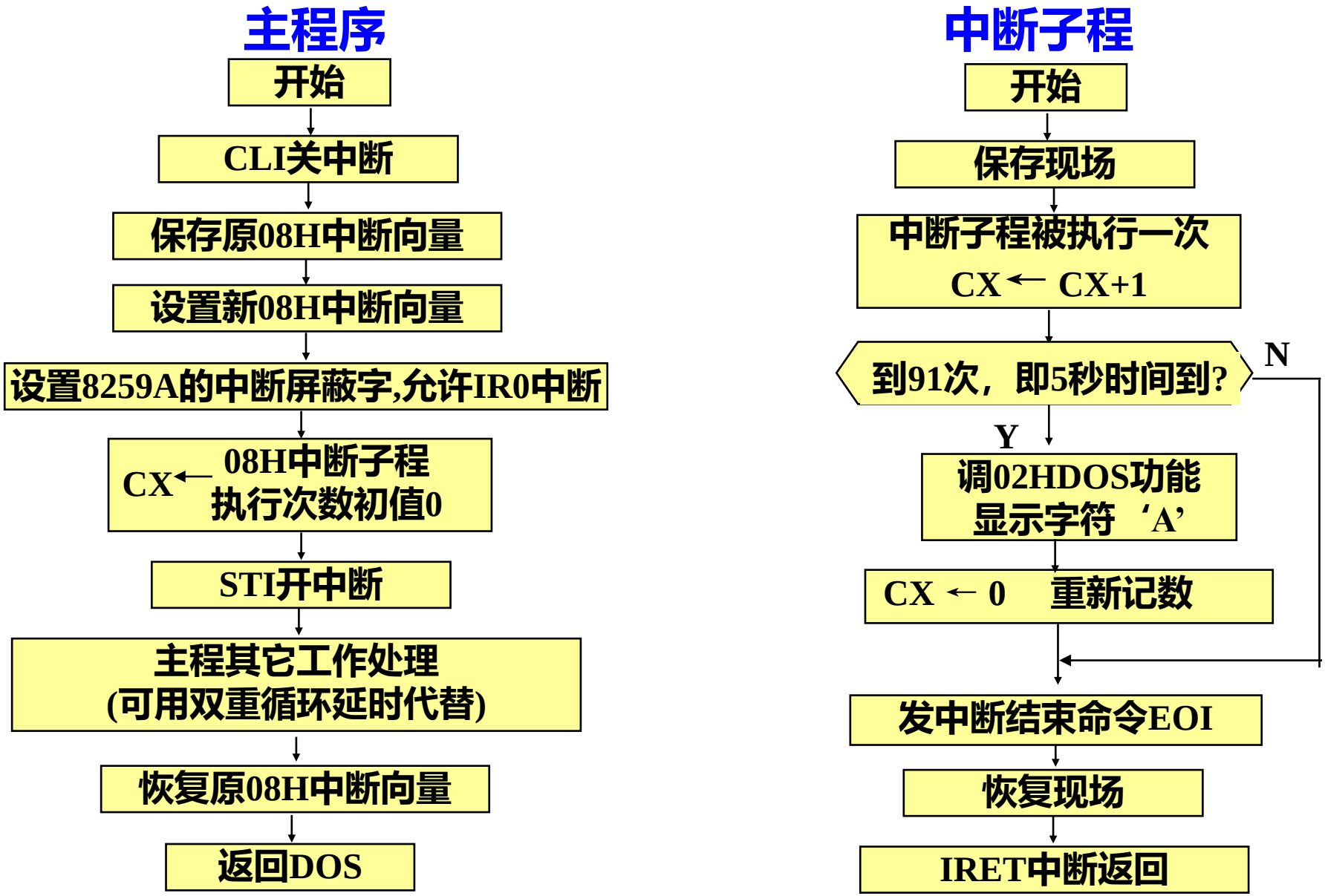
例　利用连接在8259A IR0上的时钟信号，
编写具有定时功能程序，
要求每隔5秒在屏幕上显示一个字符 ' A'。



- 分析
  - 时钟信号连接在IR0上，申请的中断类型号为08H。
  - 时钟信号是一个频率为18.2HZ的方波信号，
    - 即每秒向8259A发出18.2次的中断申请。
    - 如果CPU响应该中断申请，
    - 则以每秒18.2次的频率执行08H类型的中断子程。



- 中断申请的任务是每5秒显示一个字符‘A’
  - 1秒执行18.2次，则5秒执行18.2×5=91次，
  - 只在执行到第91次时显示‘A’，其它不显示。
- 在中断子程中用一个量来计数中断子程被执行的次数，
  - 即CPU响应IR0中断申请的次数。
  - 当达到91次，则显示‘A’，并将计数值清0，重新计数；
  - 其它情况只计数，不显示‘A’。
- 中断子程的最后应发中断结束命令EOI。


主程序
开始
CLI关中断
保存原08H中断向量
设置新08H中断向量
设置8259A的中断屏蔽字,允许IR0中断
CX← 08H中断子程
执行次数初值0
STI开中断
主程其它工作处理
(可用双重循环延时代替)
恢复原08H中断向量
返回DOS
中断子程
开始
保存现场
中断子程被执行一次
CX← CX+1
到91次，即5秒时间到?
N
Y
调02HDOS功能
显示字符 ‘A’
CX ← 0 重新记数
发中断结束命令EOI
恢复现场
IRET中断返回

```
code    SEGMENT
        ASSUME   CS:code
start:  CLI                                  ;关中断
        MOV   AL, 08H
        MOV   AH, 35H
        INT     21H                          ;取系统08H类型中断向量
        PUSH    ES                           ;用堆栈保存
        PUSH    BX
        PUSH    DS
        MOV   DX, OFFSET  display   ;设置08H 类型中断向量
        MOV   AX, SEG     display
        MOV   DS, AX
        MOV   AL, 08H
        MOV   AH, 25H
        INT     21H
        POP     DS
```

```
        IN      AL,   21H              ;设置8259A中断屏蔽字,
        AND     AL,   1111 1110B       ;允许IR0中断
        OUT     21H,  AL
        MOV     CX,   0                ;置中断子程计数器初值为0
        STI                            ;CPU开中断
        MOV     SI,   8000H            ;用双重循环延时,
w2:     MOV     DI,   8000H            ;模拟CPU对其他任务的处理
w1:     DEC     DI                     ;DI,SI的值可修改,控制主程执行时间
        JNZ     w1
        DEC     SI
        JNZ     w2
        POP     DX                     ;恢复系统08H类型中断向量
        POP     DS
        MOV     AL,  08H
        MOV     AH,  25H
        INT     21H
        MOV     AH,  4CH               ;返回DOS
        INT     21H
```

```
display   PROC                ;自编的08H类型中断子程
          PUSH  AX            ;保护现场
          INC   CX            ;执行中断子程一次，计数值加1
          CMP   CX, 91
          JNZ   exit          ;到91次了吗？未到跳转至exit
          MOV   DL, 'A'       ;到，显示 'A'
          MOV   AH, 02
          INT   21H
          MOV   CX, 0         ;清计数值为0，重新计时
exit:     MOV   AL, 20H       ;发中断结束命令EOI
          OUT   20H, AL
          POP   AX            ;恢复现场
          IRET                ;中断返回
display   ENDP
code      ENDS
          END  start
```

DOS系统提供的08H类型中断子程:

```
sys08   PROC
        …
        MOV    AX, 0040H
        MOV    DS,  AX
        …
        INT    1CH
        …
        MOV   AL,  20H
        OUT   20H,  AL
        IRET
sys08   ENDP
```

DOS系统提供的1CH类型中断子程:

```
sys1c   PROC
        IRET
sys1c   ENDP
```

系统08H类型中断子程

要点:

- *进入子程后，修改了DS段寄存器的值;*
- *程序中调用了1CH类型中断子程，每执行一次08H中断子程，执行一次1CH中断子程;*
- *程序最后发中断结束命令EOI*

```
sys08    PROC
         …
         MOV    AX, 0040H
         MOV    DS,  AX
         …
         INT    1CH
         …
         MOV   AL,  20H
         OUT    20H,  AL
         IRET
sys08 ENDP
```

- 1CH类型是系统为用户保留的定时中断子程。系统1CH中断子程只有一条中断返回语句，确保08H中断子程调用后能返回。
- 用户可通过编写1CH中断子程实现具有定时功能的任务

```
sys1c    PROC
         IRET
sys1c    ENDP
```

1CH类型是软中断，按编写软中断子程方法编写：

① 系统08H中断子程序改变DS值，故：

1CH中断子程用到定义的变量，须在子程序中给DS赋值。

② 1CH类型是软中断，不需要发中断结束命令。


谢谢大家！