《微机原理与接口》

第6章 中断和8259A

教师：苏曙光

华中科技大学软件学院

- 教学内容
  - 第1节 8088中断系统概念
  - 第2节 中断机制的硬件基础
  - 第3节 8259A性能和结构
  - 第4节 8259A中断响应过程
  - 第5节 8259的中断操作和命令
  - 第6节 8088中断系统应用

第1节 8088中断系统的概念

- 中断的概念
  - CPU的机制或过程
  - CPU在正常运行程序时，由于内部/外部事件事件引起CPU中断正在运行的程序，转去执行为该事件预先安排的服务程序，服务完毕后，再返回原来的程序继续执行。
- 中断的作用和应用
  - ①同步操作/并行操作
  - ②实时处理
  - ③故障或异常处理

- 中断源和其分类
  - 中断源：引起中断的事件或发出中断的外设。
  - 分类
    - 外部中断
      - 可屏蔽中断（INTR引脚）
        - 外设
        - 受IF控制， N = 8~255
      - 不可屏蔽中断（NMI引脚） **Non Maskable Interrupt**
        - 电源，内存错误等
        - 不受IF控制，**N=2**
    - 内部中断
      - CPU内部
      - 软件中断（指令中断）
      - 异常（指令执行时发生错误）

内部中断

- 内部中断（异常，Exception）： 8088内部执行程序错误
  - （1）DIV或IDIV指令
  - （2）INT指令
  - （3）INTO指令
  - （4）单步中断
    - ◆为用户提供发现、调试并解决程序执行异常的途径
      - ◘例如BIOS和DOS功能调用，DEBUG

陷阱标志位TF=1时，每条指令执行完引起中断。N=1

内部中断的部分中断的详细说明

- DIV或IDIV（无/有符号数除法）指令
  - 除数为零或商溢出
  - N = 0

```
MOV BL, 0
IDIV  BL     ;除数BL = 0，产生除法错中断
```

```
MOV AX, 200H
MOV BL, 1
DIV   BL     ;商溢出，产生中断
```

内部中断的部分中断的详细说明

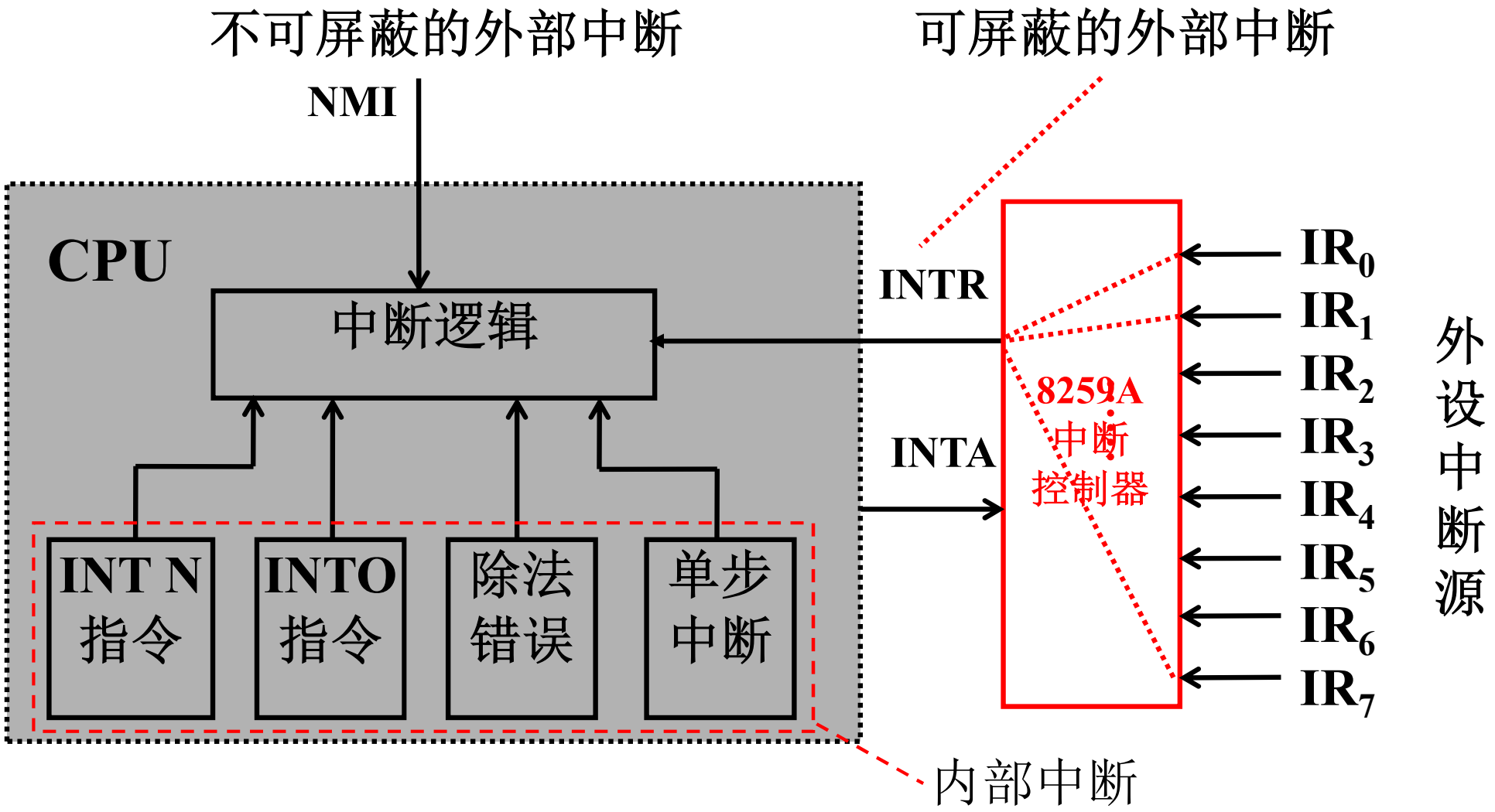
- 溢出中断
  - 若溢出标志位OF=1，则INTO指令引起中断
  - N = 4

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

- 软中断
  - INT n指令产生一个中断
  - N = n
  - 例子：n = 21H，DOS系统功能调用

- 中断源的优先权
  - 软件中断
    - 除法错中断
    - 指令中断
    - 溢出中断
  - 非屏蔽中断
  - 可屏蔽中断
  - 单步中断

8088的中断类型

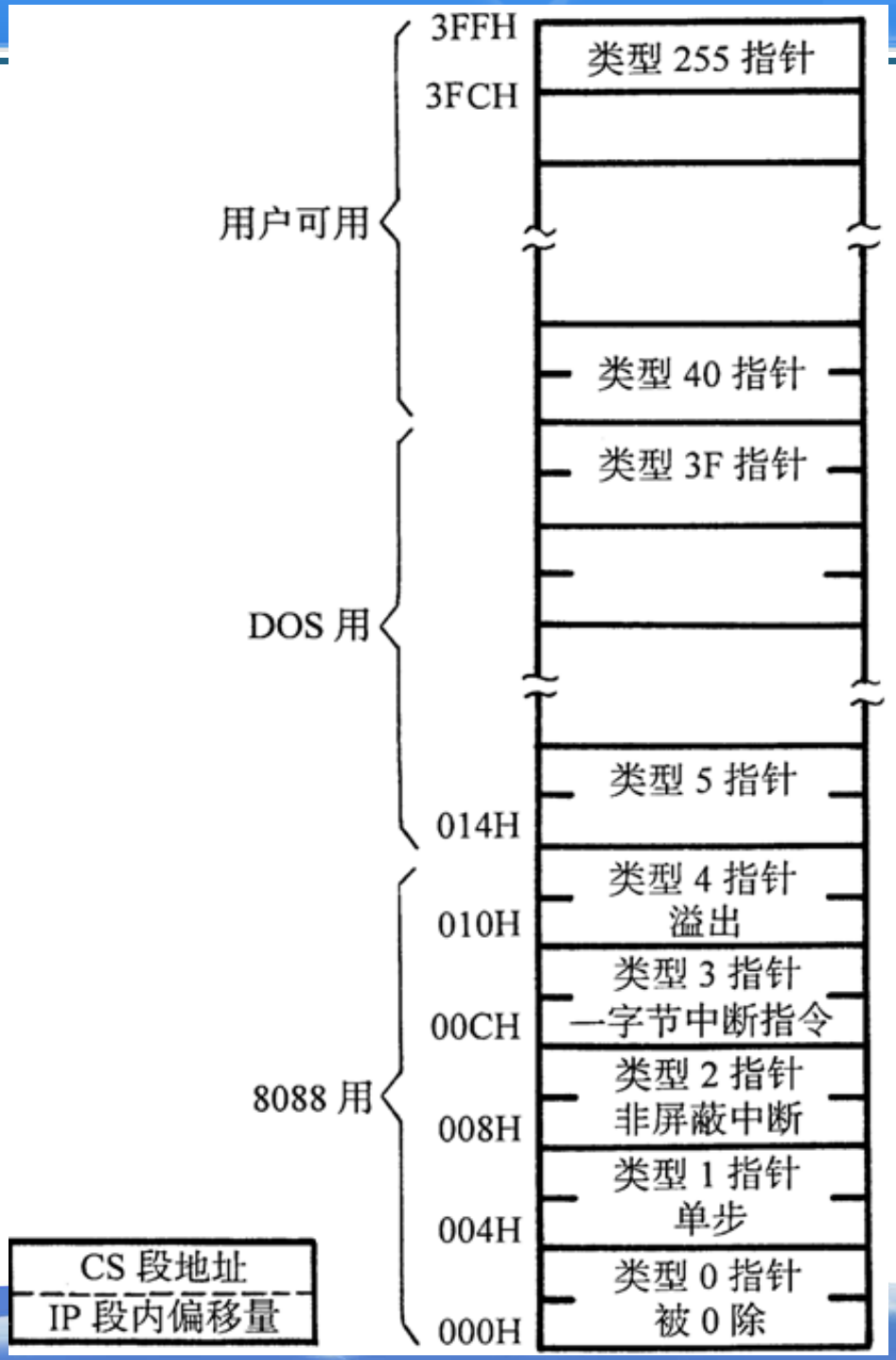

不可屏蔽的外部中断
NMI
可屏蔽的外部中断
CPU
中断逻辑
INTR
INTA
8259A
中断
控制器
IR0
IR1
IR2
IR3
IR4
IR5
IR6
IR7
外设中断源
INT N
指令
INTO
指令
除法
错误
单步
中断
内部中断

进入中断服务程序的向导：8088的中断向量表

- 中断向量
  - 中断向量是指：中断源（N）的中断服务程序的入口地址CS:IP
    - 段基址CS：偏移地址IP
    - 每个向量占4字节。
- 中断向量表
  - 中断向量表是按相应的**中断源编号N递增**存放中断向量集合。
  - 中断向量表一般存放在内存中的0地址。

8088的中断向量表

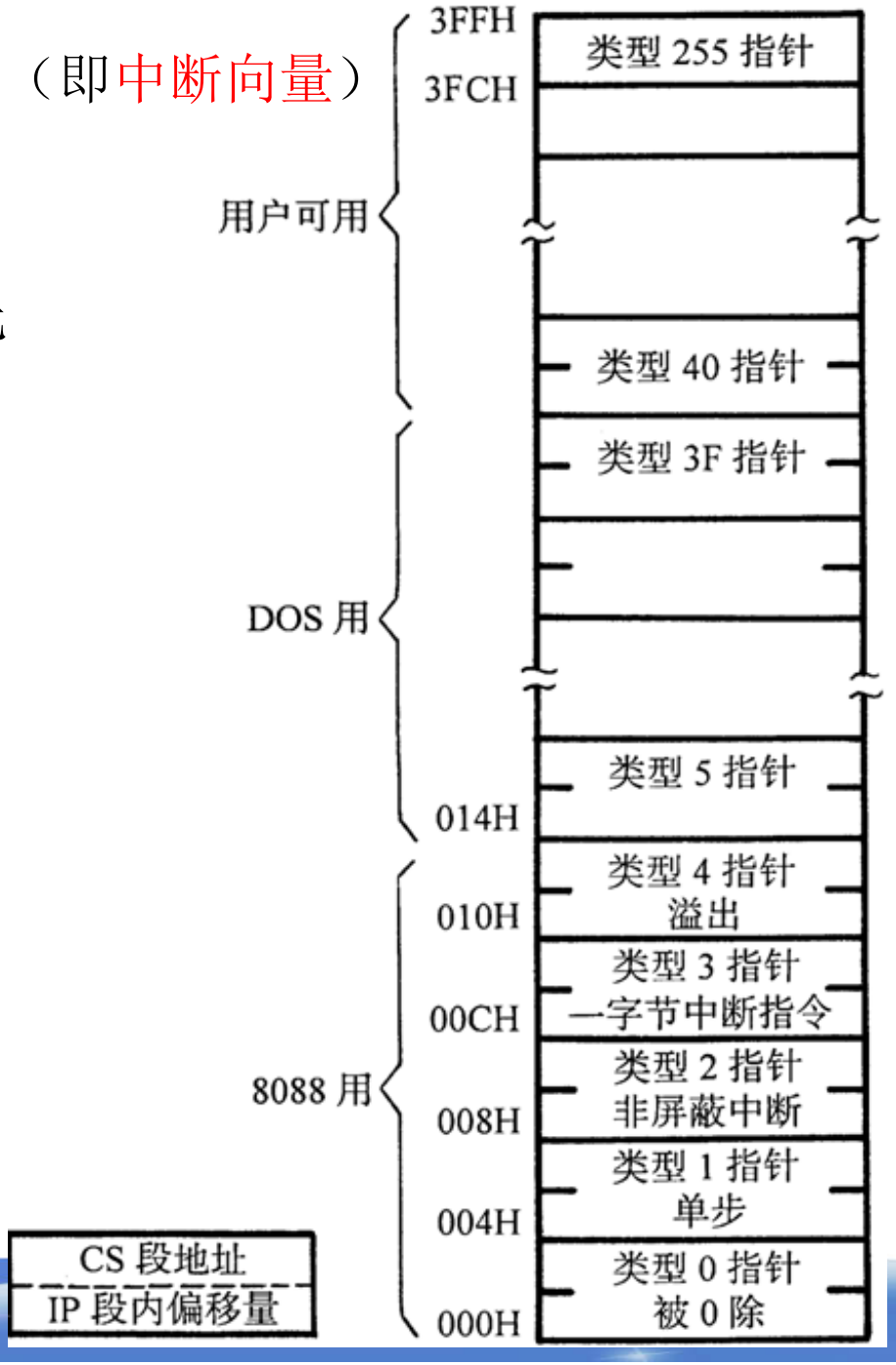
- 含有256个中断源
- 中断向量表存放位置
  - 0h ~ 3FFh
  - 共1K个字节
- 中断类型码(中断源编号)
  - 内部中断：自动形成
    - ◆被零除　　0
    - ◆单步中断　　1
    - ◆NMI　　2
    - ◆断点中断　　3
    - ◆溢出中断　　4
    - ◆INT　N
  - 外部中断：来自中断控制器



8088中断向量表

- 查找N号中断源的中断服务程序的入口地址（即中断向量）
  - （1）先找到中断向量的存放地址

    存放地址 = 4 * N
  - （2）读取（4 * N）~（4 * N +3）单元
    - ◆读出中断向量并写入CS:IP寄存器
      - ◘跳转

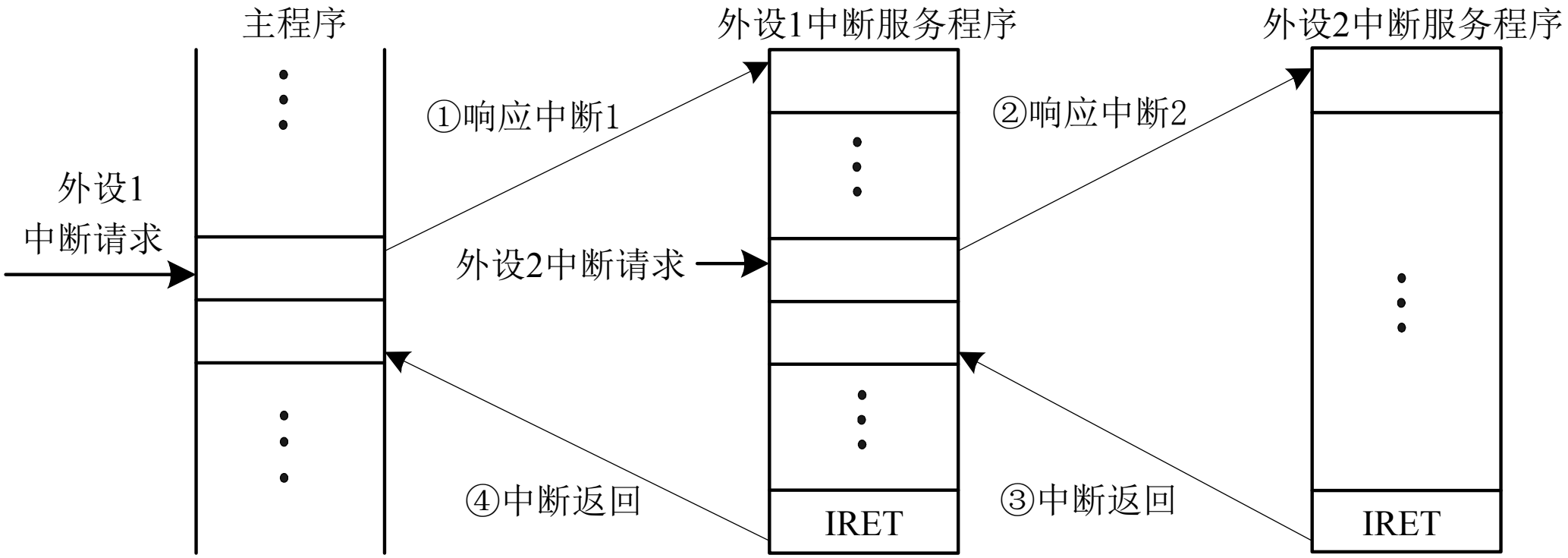


中断系统的功能

- 功能：发现和识别中断源，实现为中断源提供中断服务
- （1）能产生和识别中断
  - 外设能发出中断请求
  - 接口能屏蔽指定的中断请求
  - CPU能识别中断源
  - CPU能决定是否响应中断(响应中断的条件)
    - ◆①执行完现行指令
      - □最后1个总线周期最后1个状态(T4)，检测INTR引脚。
    - ◆②开中断状态：STI指令
- （2）能完成中断响应（提供中断服务和返回）
  - 关中断-保护断点-保护现场-中断服务-恢复现场-返回开中断
- （3）能支持优先权排队
- （4）能实现中断嵌套

中断嵌套的例子

- 外设2的优先级 > 外设1的优先级
  - 外设1进行中断服务时，被外设2的事件中断。

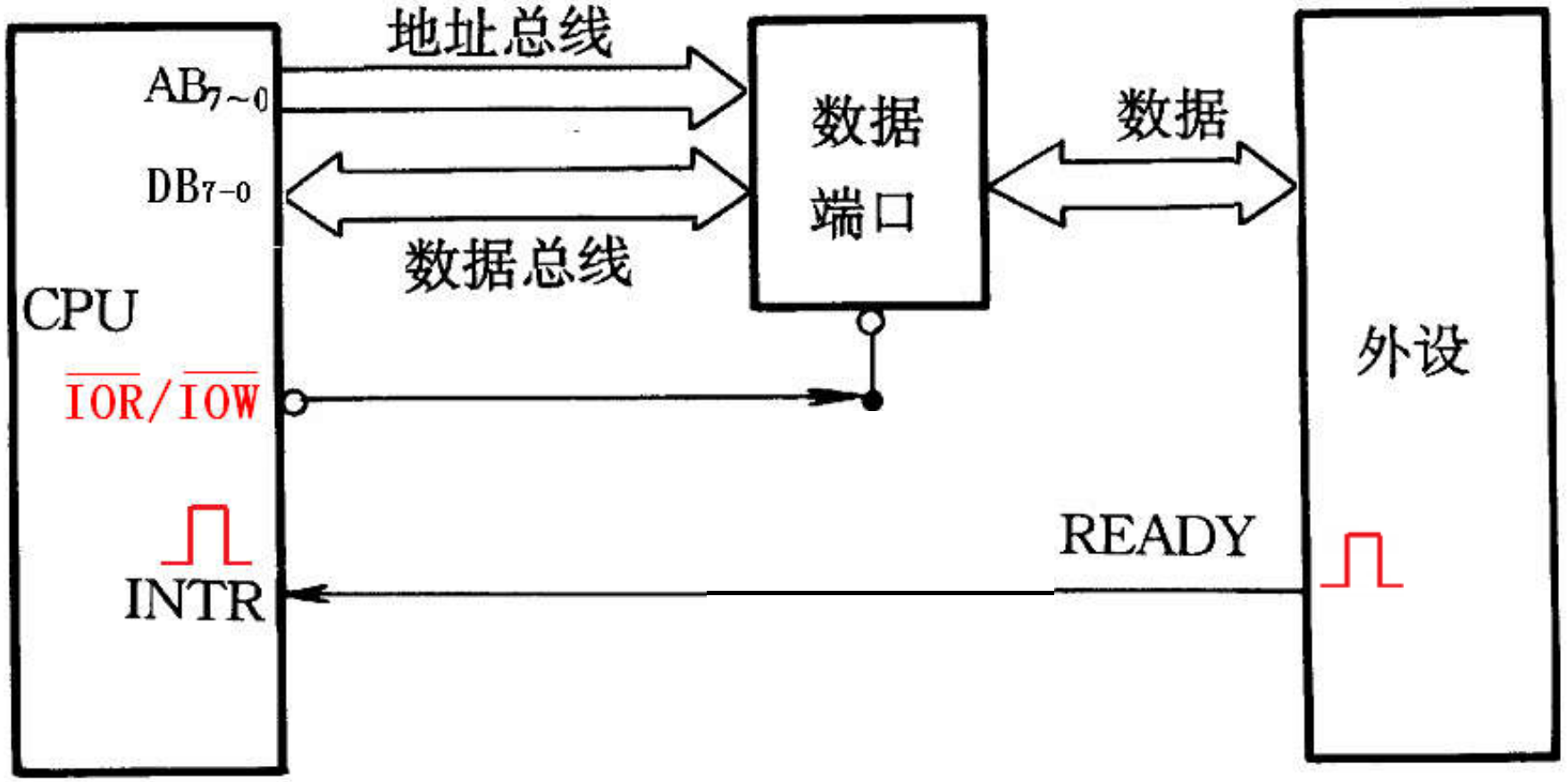


第2节 中断机制的硬件基础

中断机制的硬件基础

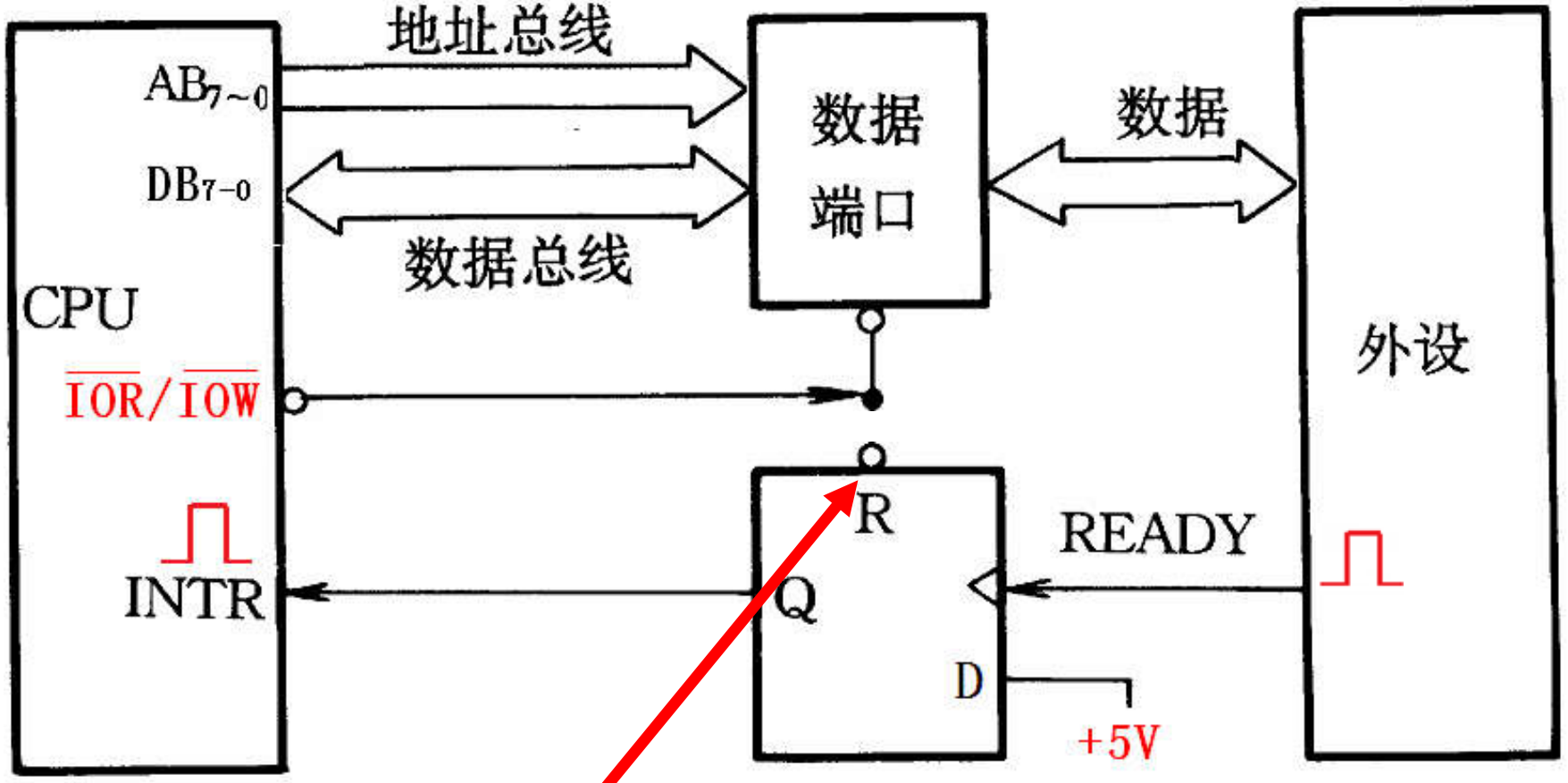
- 外设
  - 能发出且保持中断申请直到其响应为止。
- 接口
  - 能屏蔽或开放某个外设的中断请求。
- CPU
  - 能识别中断源
  - 管理中断优先级

外设：能发出且保持中断申请直到其响应为止。



外设：能发出且保持中断申请直到其响应为止。

- 措施：每个中断源配置一个中断请求触发器（1bit）。



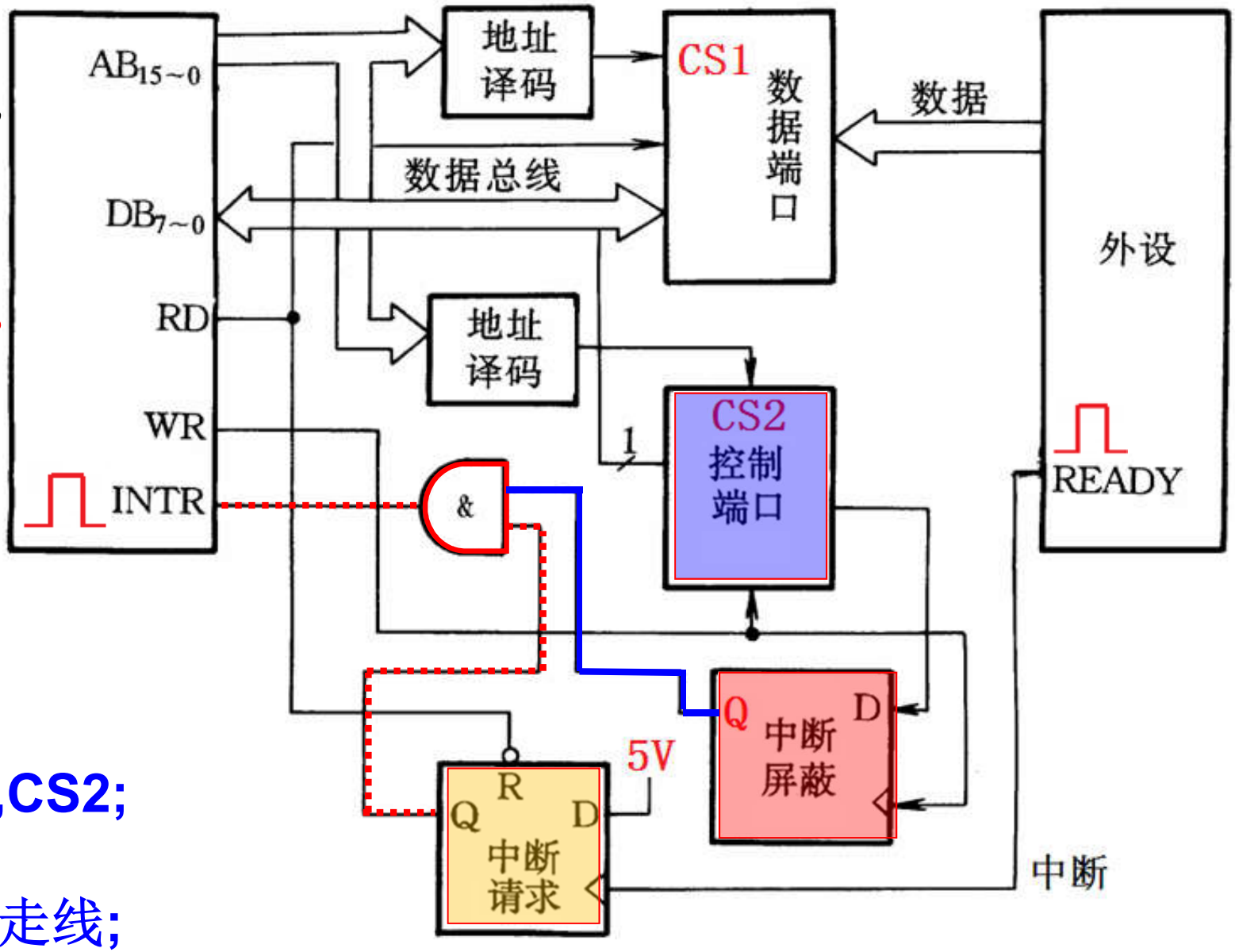
外设就绪后，INTR=1

CPU响应后，INTR=0？

中断请求
触发器

接口：能屏蔽或开放某个外设的中断请求

- 中断请求触发器的Q端不再直连INTR，而受到与门控制。
- 每个中断源配置一个中断屏蔽触发器。
  - 其Q端控制与门
- 增设一个控制端口
  - 控制中断屏蔽触发器



观察与思考：

(1)两个端口:CS1,CS2;

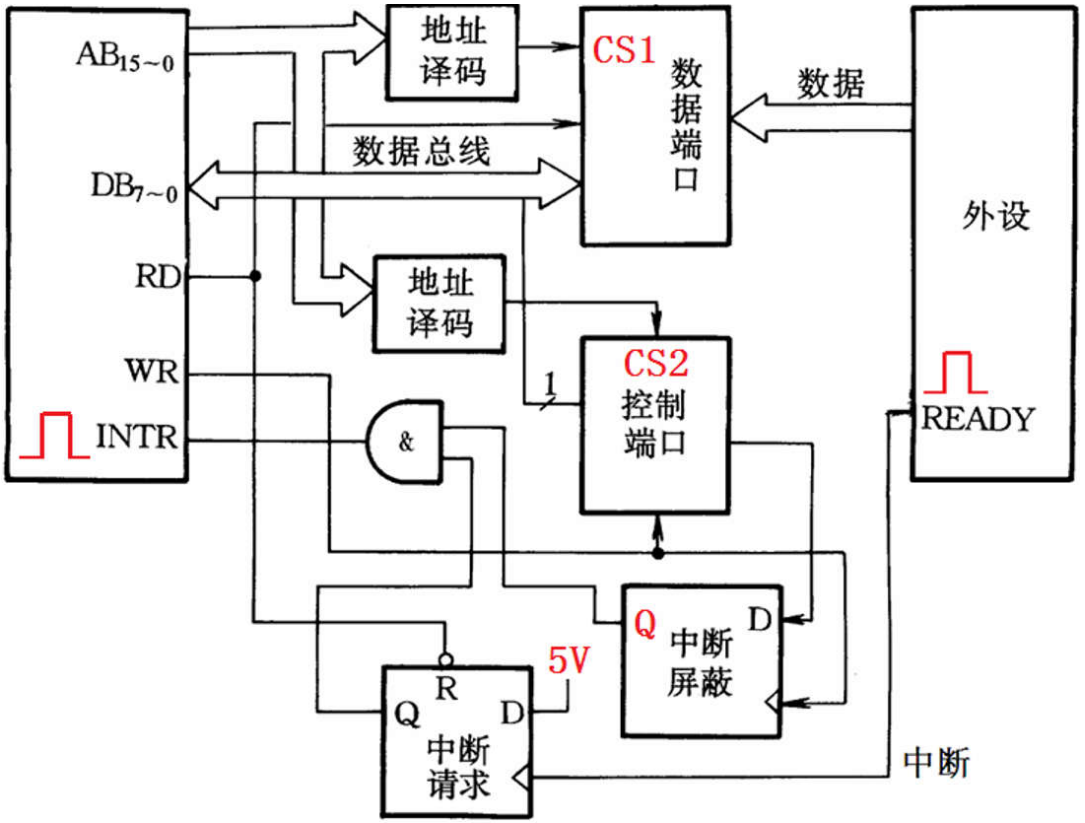
(2)两个D触发器;

(3)RD/WR信号的走线;

(4)中断请求Q怎么才能到达INTR？控制端口为0/1？

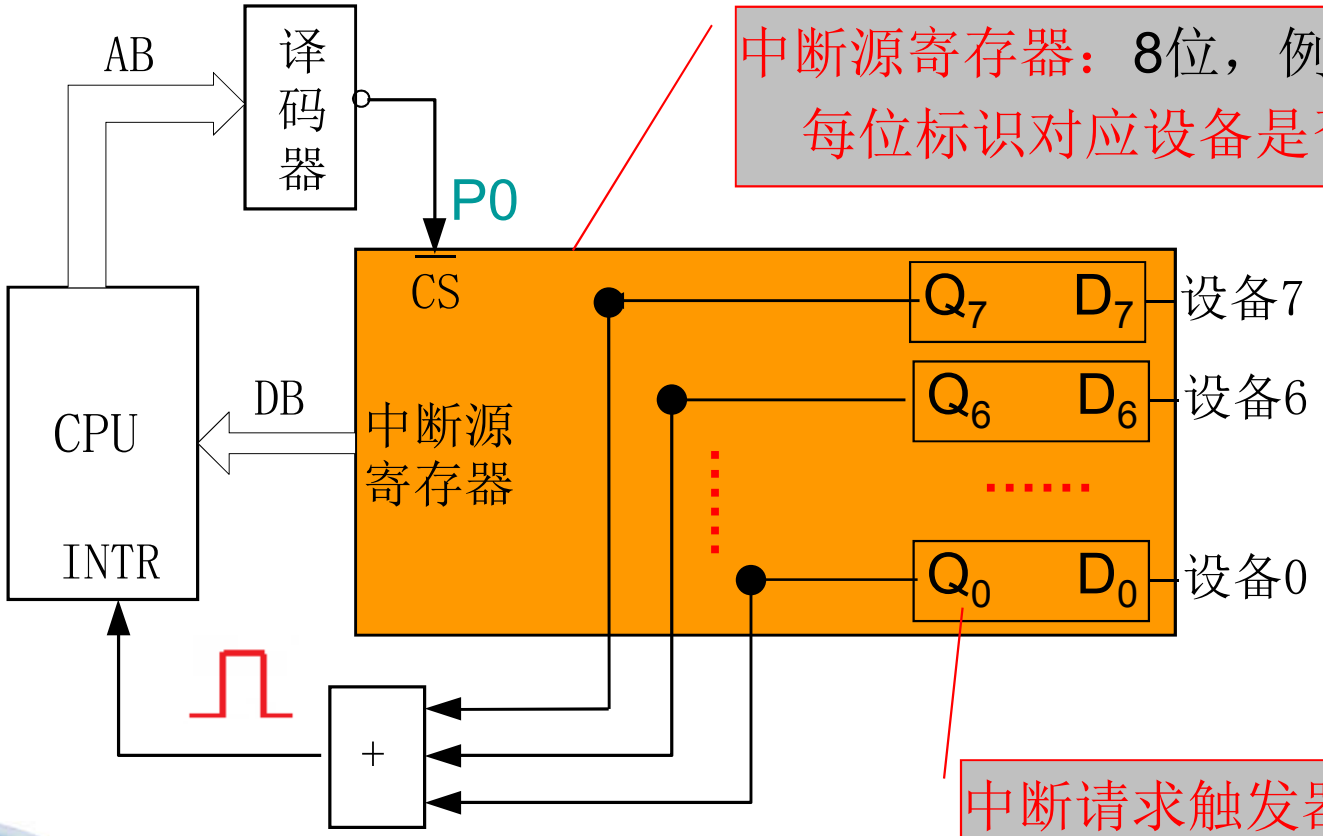
CPU：能识别中断源和管理中断优先级

- 识别中断源和中断优先级
  - 所谓优先级，是指有多个中断源同时提出中断请求时，微处理器响应中断的优先次序。
- 实现方法
  - 软件方法（查询中断）
  - 硬件方法（向量中断）

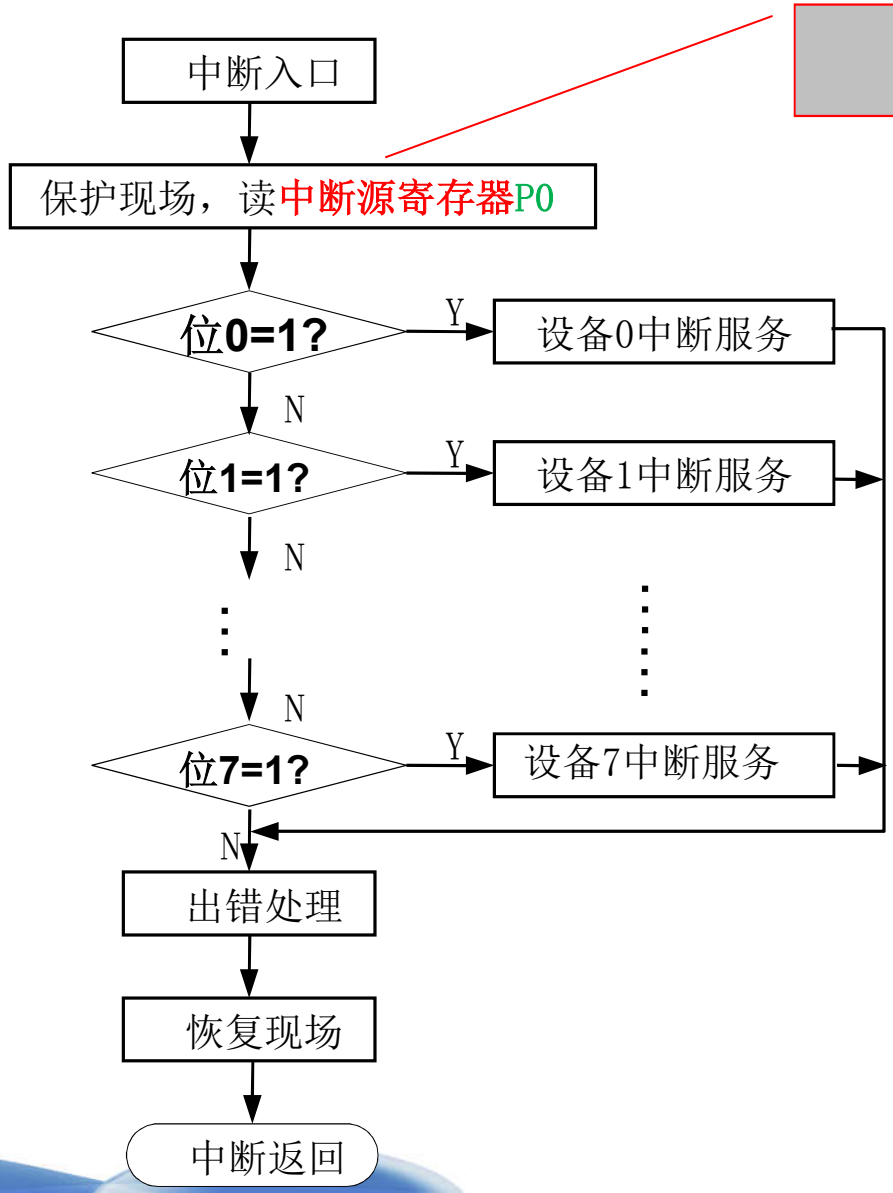


软件方法（查询中断）

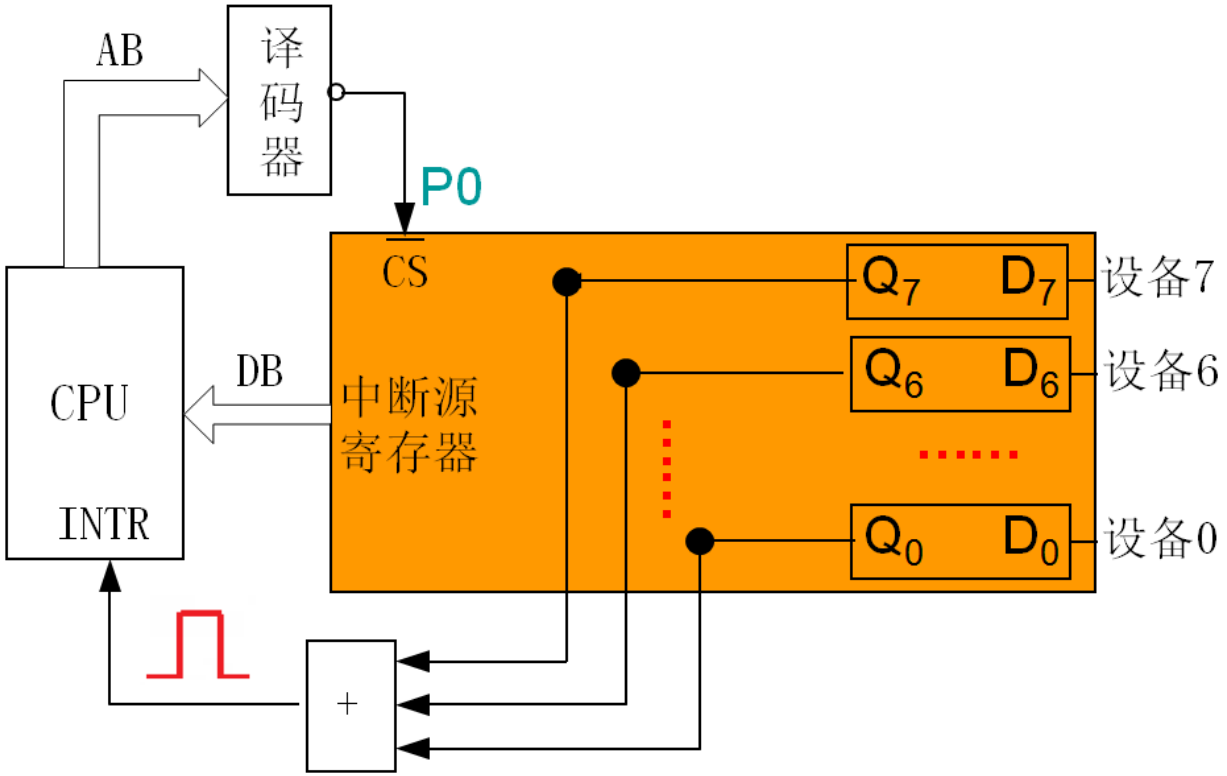
- 硬件原理
  - 将8个外设中断请求触发器组合为中断源寄存器(端口地址P0)
  - 将8个中断请求信号输入或门，其输出连接INTR。



软件流程



8位，例如01010001



- 掩码屏蔽法

```
IN      AL，[20H]  ；输入中断源寄存器的数据
TEST    AL，01H    ；检查位0（电源故障）是否有中断请求
JNE     PWF        ；有，则转至电源故障处理程序
TEST    AL，02H    ；检查位1(磁盘)是否有中断请求
JNE     DISS       ；有，转至磁盘服务程序
TEST    AL，04H    ；检查位2(键盘)是否有中断请求
JNE     KEYBRD     ；有，转至键盘服务程序
……
```

注意：JNE指令：当ZF = 0跳转 。

(位与的结果不为0，则ZF = 0，否则ZF=1)

- 移位检测法

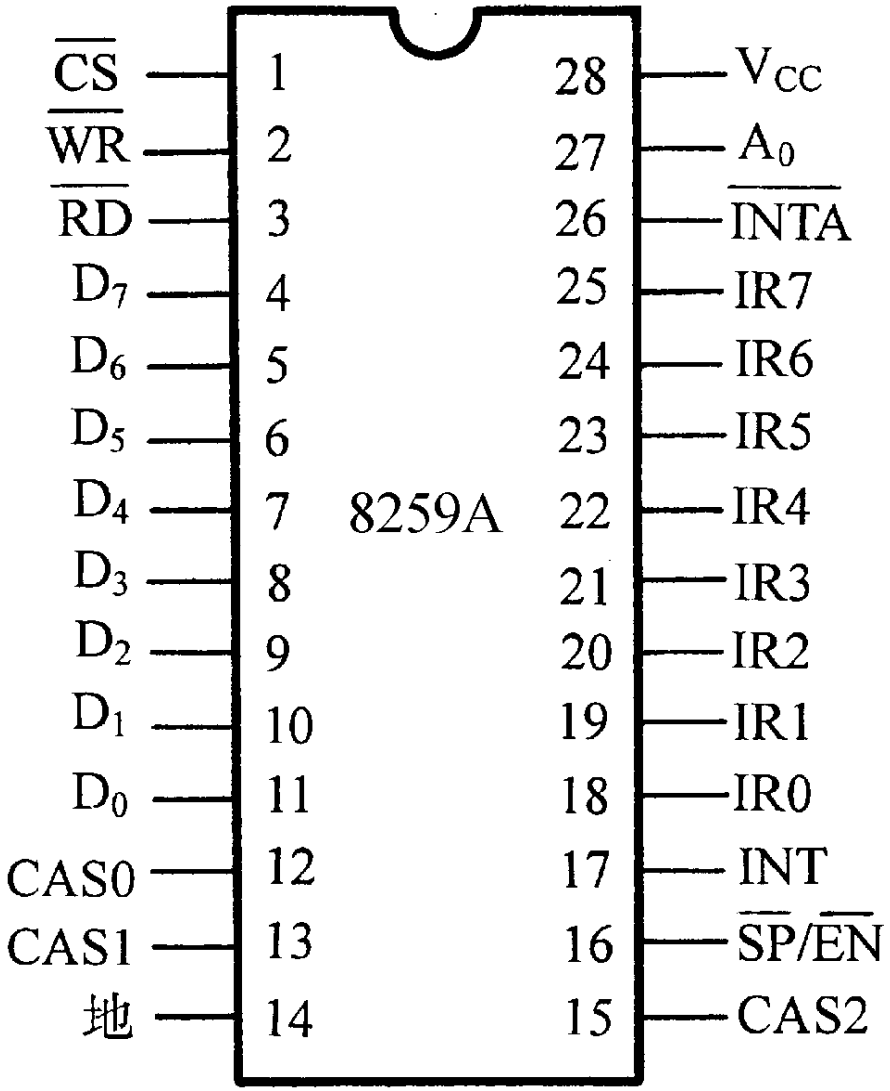
```
IN      AL, [20H]    ; 输入中断源寄存器的数据
RCL     AL, 1        ;左移1位，带进位
JC      BIT7          ;有进位跳转：位7对应中断服务程序
RCL     AL, 1        ;左移1位，带进位
JC      BIT6         ;有进位跳转：位6对应中断服务程序
……
RCL     AL, 1        ;左移1位，带进位
JC      BIT0         ;有进位跳转：位0对应中断服务程序
```

- 软件查询方法特点
  - ①外设的优先权为查询次序
    - 优先级最高的最先被查询。
  - ②硬件简单
  - ③缺点
    - 进入中断服务的时间较长。
- 软件查询方法改进
  - 用硬件实现查询和优先权管理
    - 链式电路

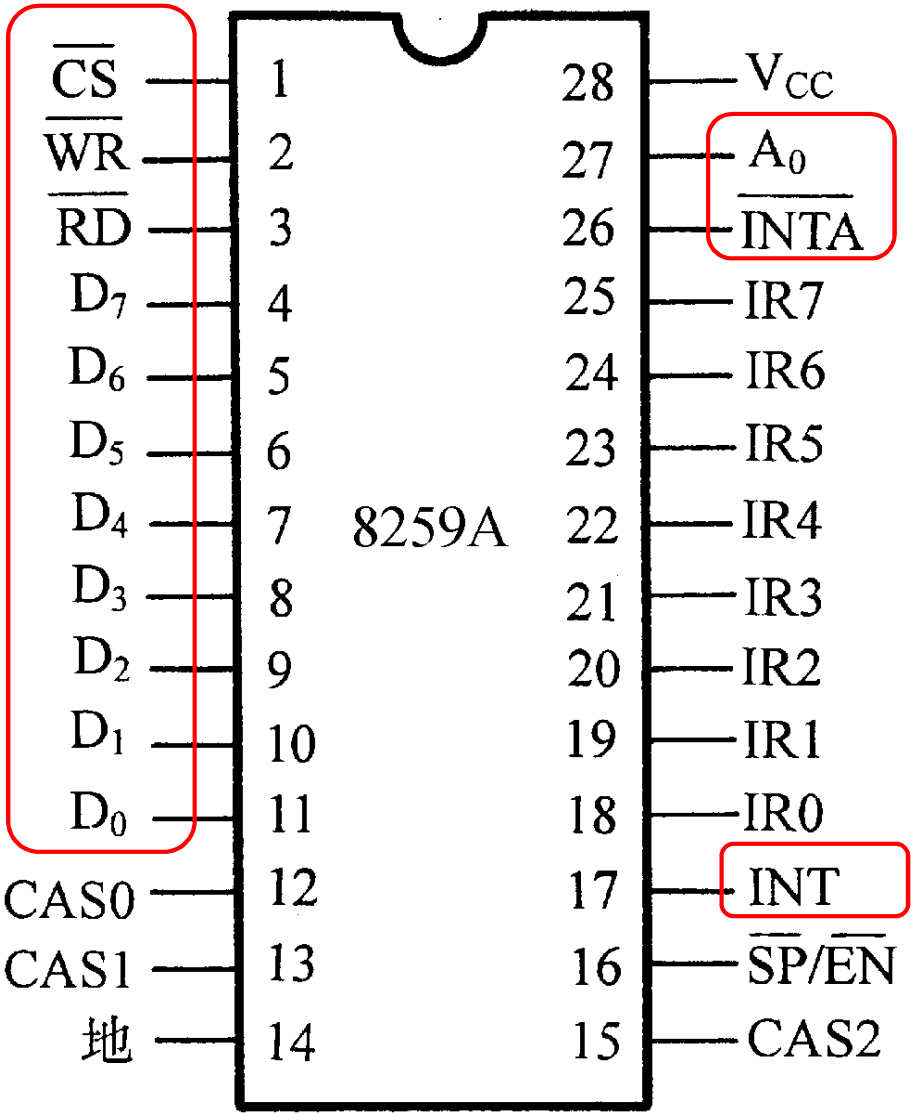
- 向量中断——用硬件方法确定中断源及优先权
  - 基本原理
    - ◆每个外设预先指定一个中断类型码（N）。当CPU识别出外设有请求中断并予以响应时，中断控制逻辑把N送入CPU，CPU据此自动查询中断向量表获取其中预存的中断服务程序的入口地址，并转入中断服务程序。
  - 特点：
    - ◆响应速度快
    - ◆硬件复杂
    - ◆典型芯片：8259A

第3节 8259A性能和结构

- 8259A的基本功能
  - 1.接收8个中断请求（可扩展64级）
  - 2.能提供中断类型号N（中断矢量）
  - 3.能屏蔽和开放外设的中断请求
  - 4.支持多种方式的优先级管理

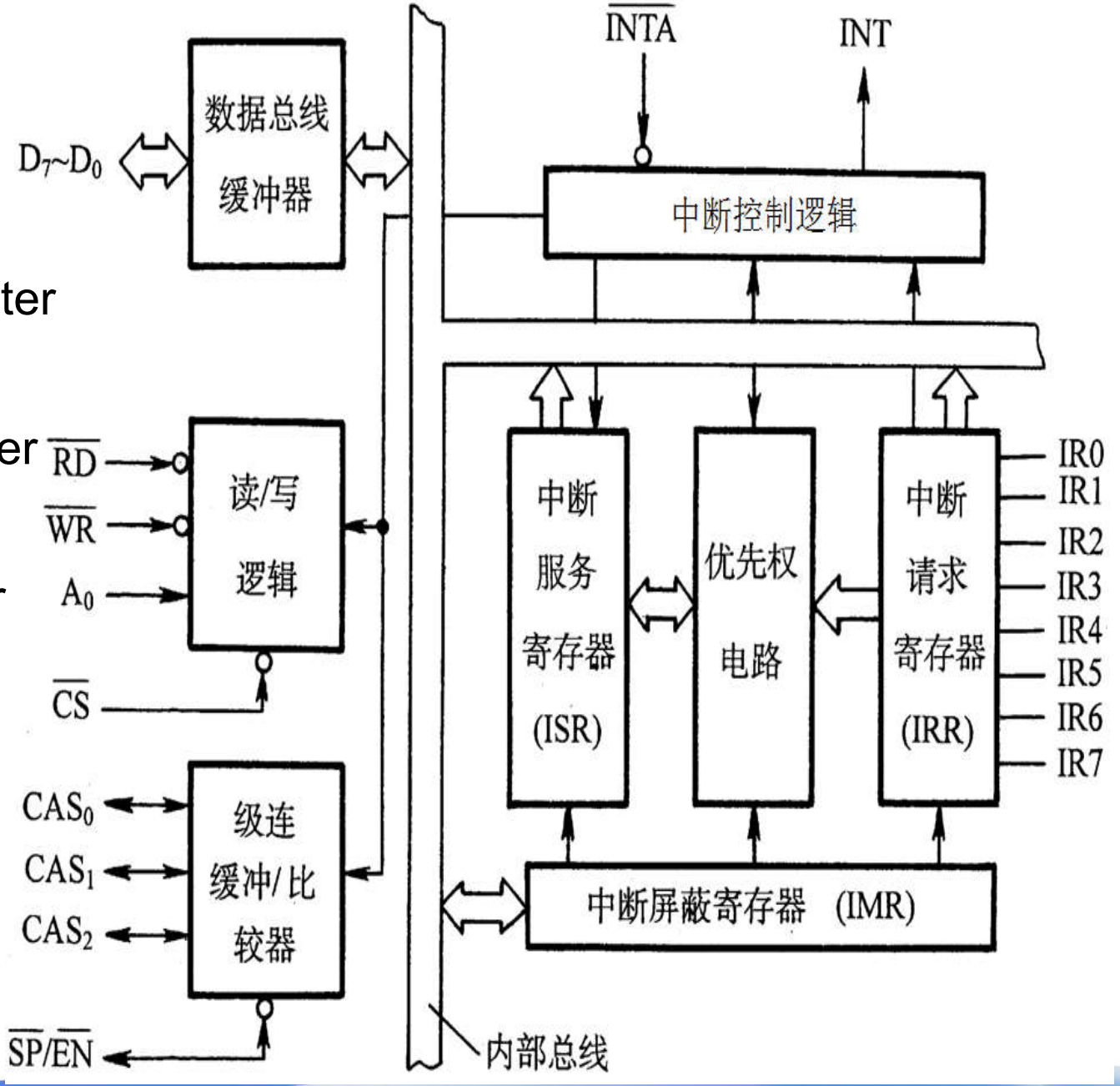


- 外部引脚
  - 面向CPU的引脚
    - ◆D7~D0，$\overline{CS}$，A0，$\overline{RD}$，$\overline{WR}$，INT，$\overline{INTA}$
  - 面向外设的引脚
    - ◆中断请求输入IR7~IR0（优先级：IR0最高）
  - 同类芯片扩展
    - ◆级联控制$\overline{SP}/\overline{EN}$，CAS2~CAS0。



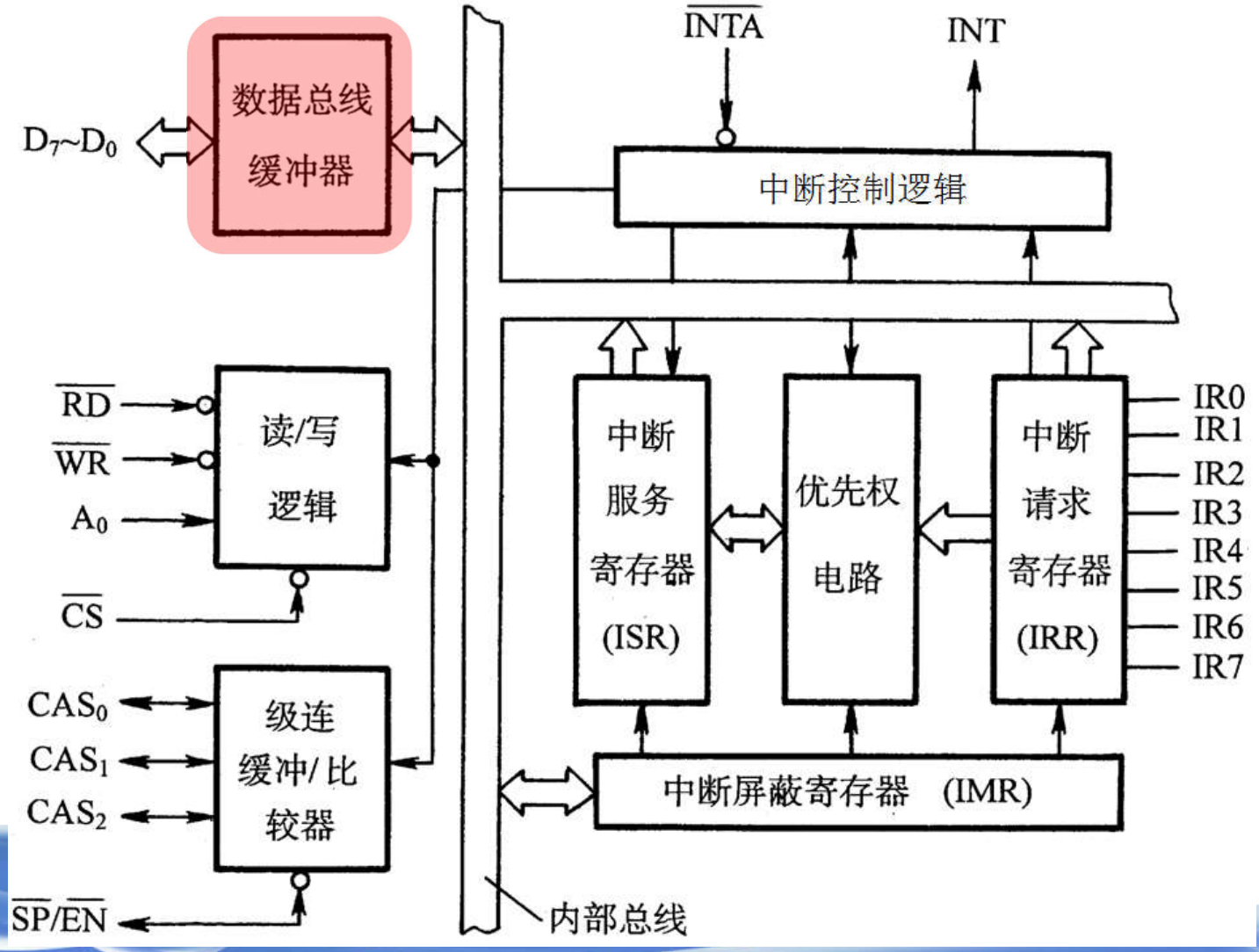
8259A内部结构

- 1.数据总线缓冲器
- 2.读/写逻辑
- 3.中断请求寄存器IRR
  - Interrupt Request Register
- 4.中断服务寄存器ISR
  - Interrupt Service Register
- 5.中断屏蔽寄存器IMR
  - Interrupt Mask Register
- 6.中断控制逻辑
- 7.级联缓冲/比较器
- 8.优先权电路



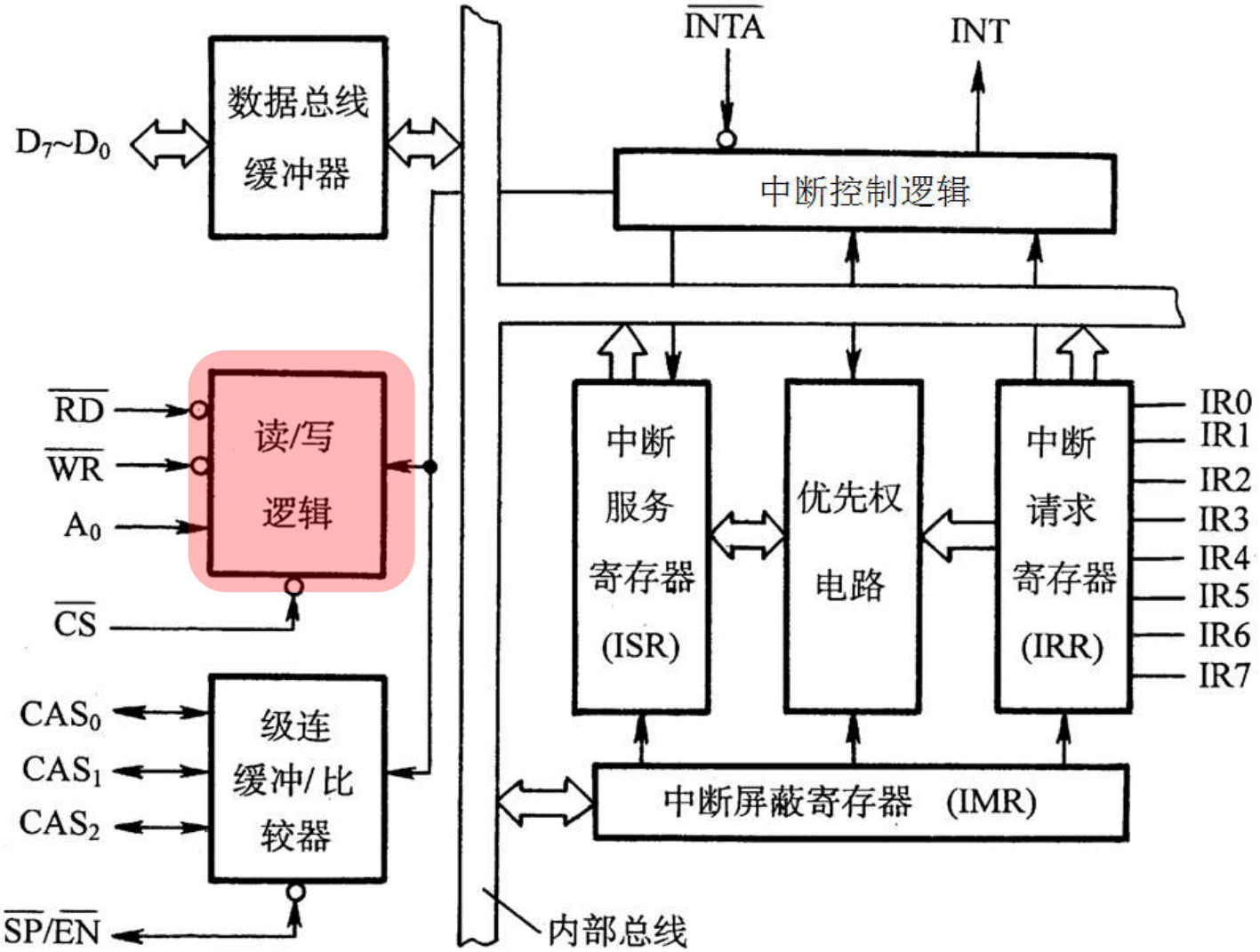
8259A结构

- ①数据总线缓冲器
  - 双向三态，与系统DB相连，传送控制字，状态，中断号等信息

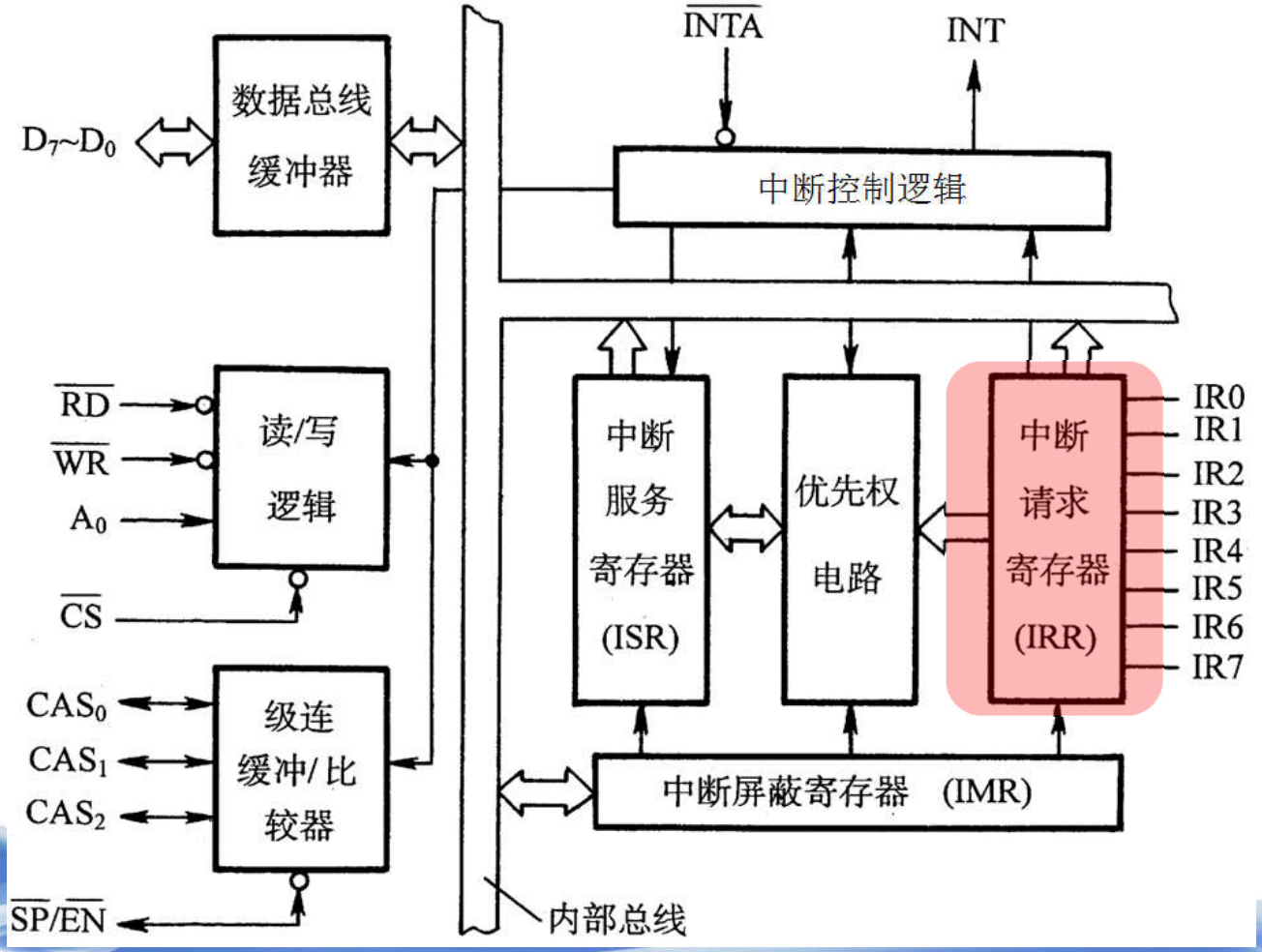


8259A结构

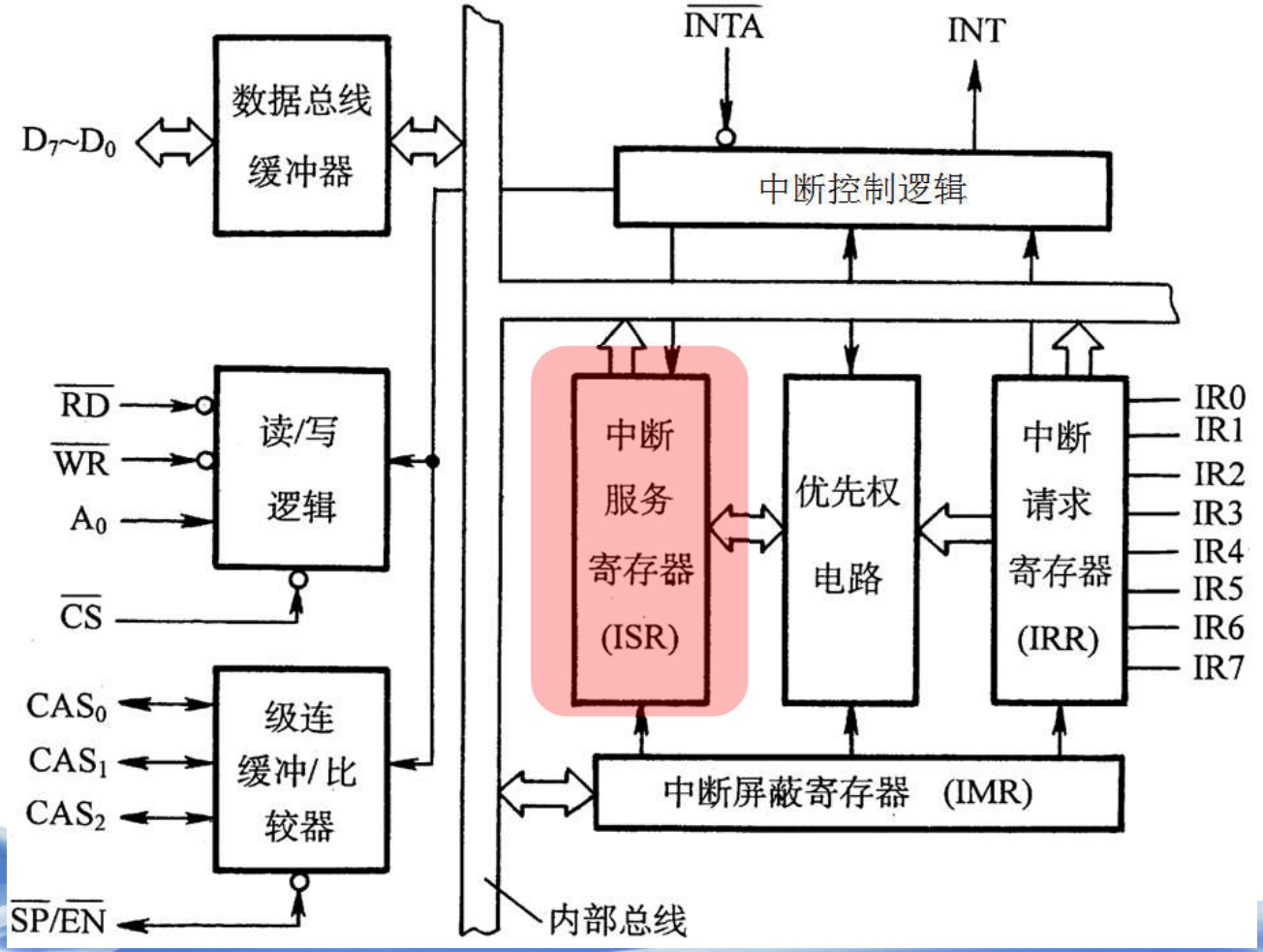
- ②读/写控制逻辑
  - 控制片选$\overline{CS}$，地址线A0,以及$\overline{WR}$/$\overline{RD}$线。执行包括写入命令（初始化和控制）或读取8259相应端口（IRR，ISR，IMR)。



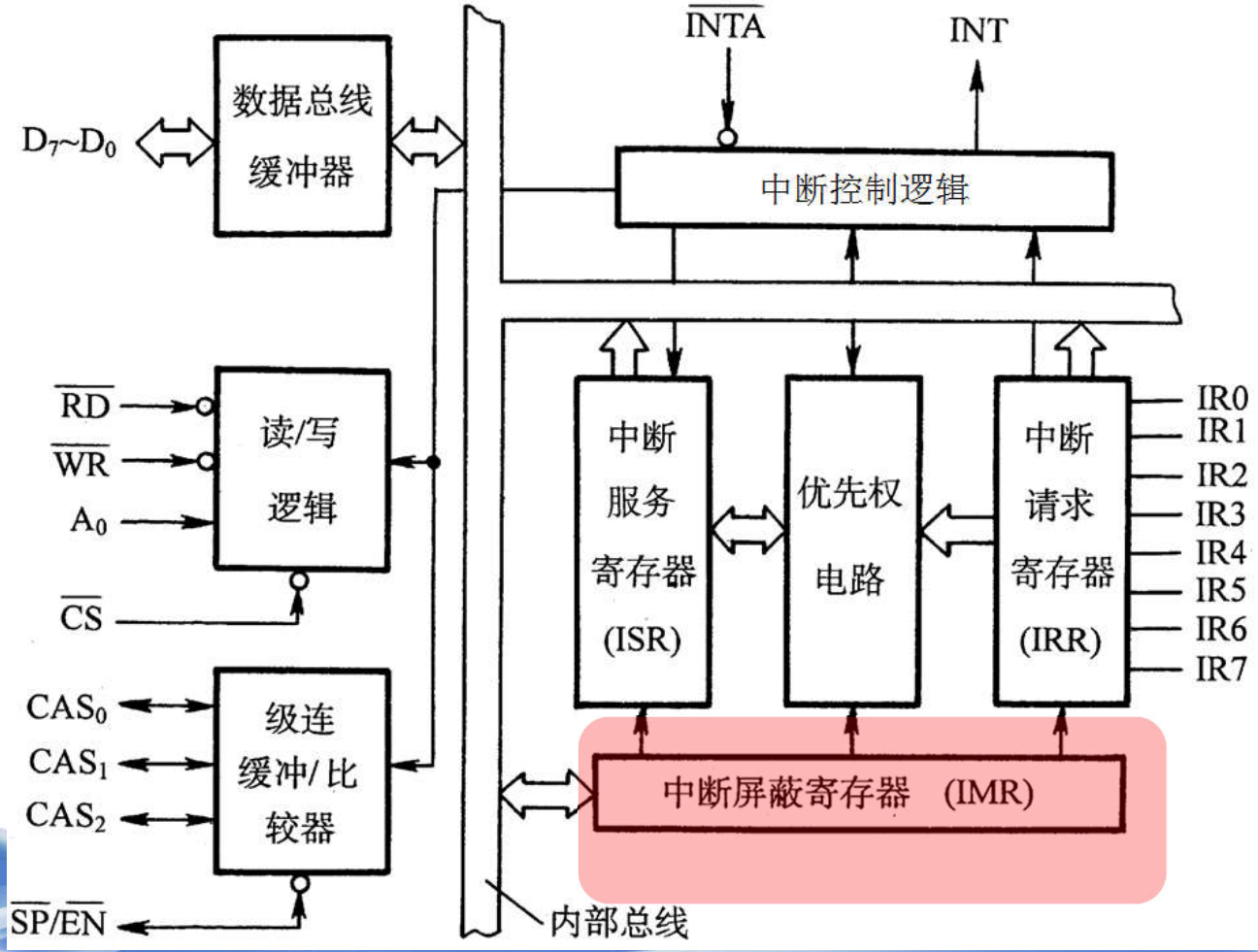
- ③中断请求寄存器IRR： Interrupt Request Register
  - 8位，寄存IR_0～IR_7是否有中断请求
  - D_i位表示IR_i脚有无中断请求：1有；0无



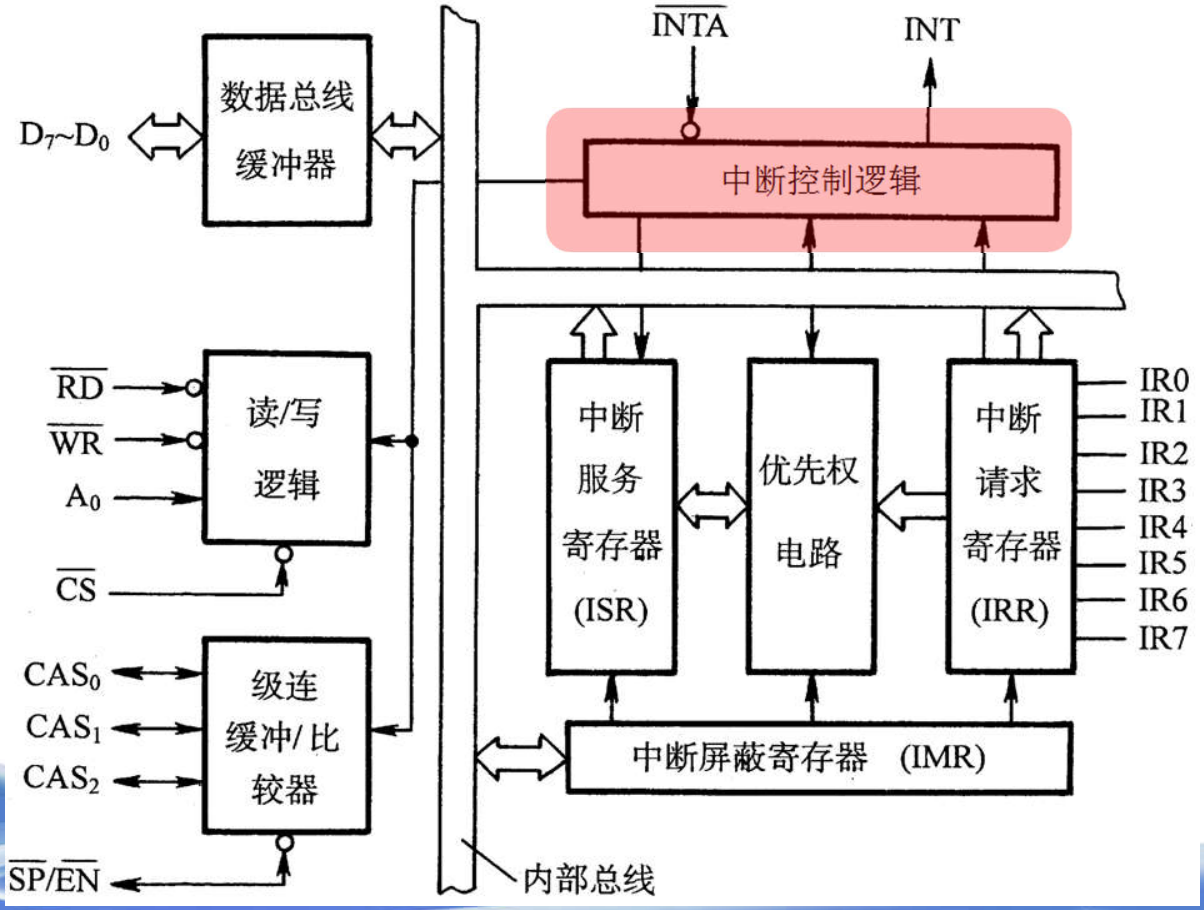
- ④中断服务寄存器ISR： Interrupt Service Register
  - 记录IR_0～IR_7中断源是否被服务中…
  - D_i位表示IR_i是否正在进行中断服务：1是；0否



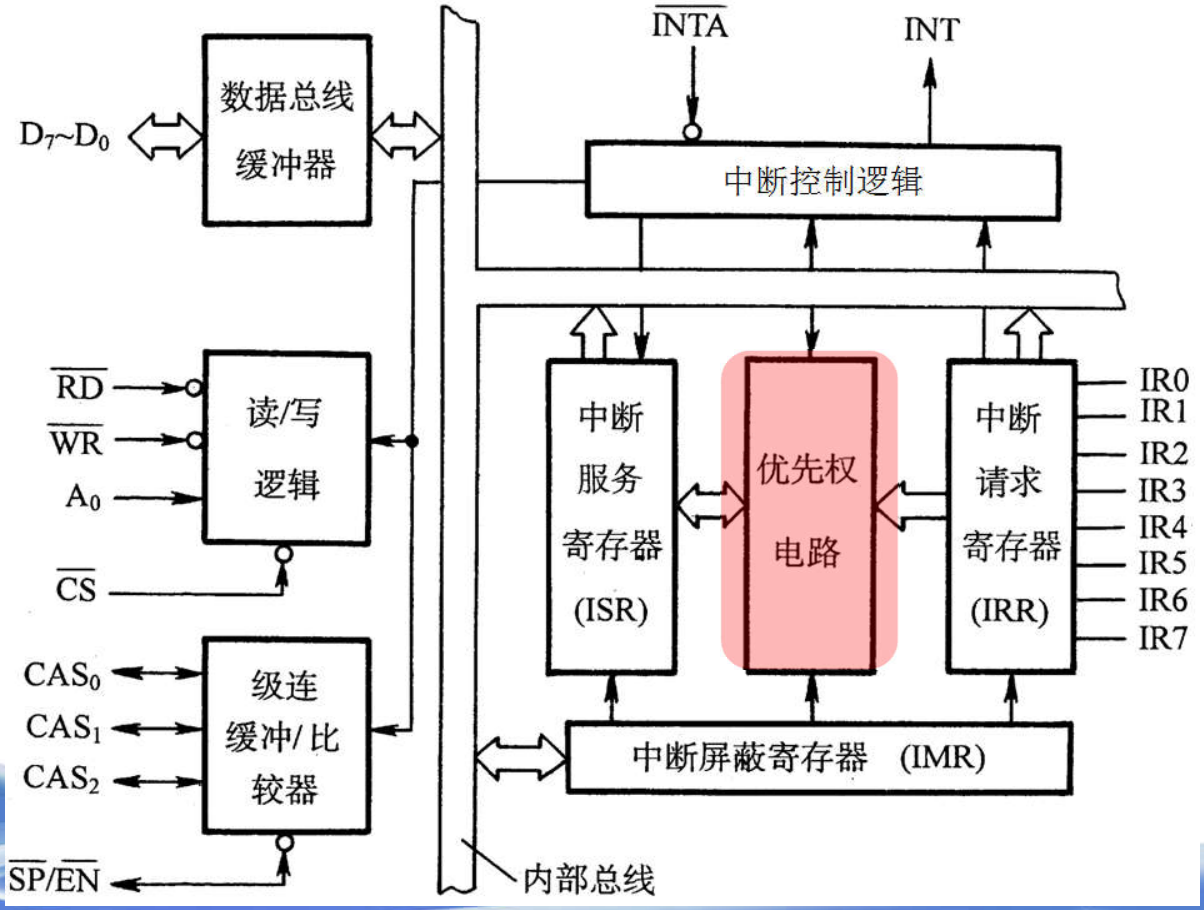
- ⑤中断屏蔽寄存器IMR： Interrupt Mask Register
  - 8位，记录是否对IR_0～IR_7的中断申请进行屏蔽（禁止）
  - D_i位表示IR_i中断申请是否进行屏蔽： 1是；0否



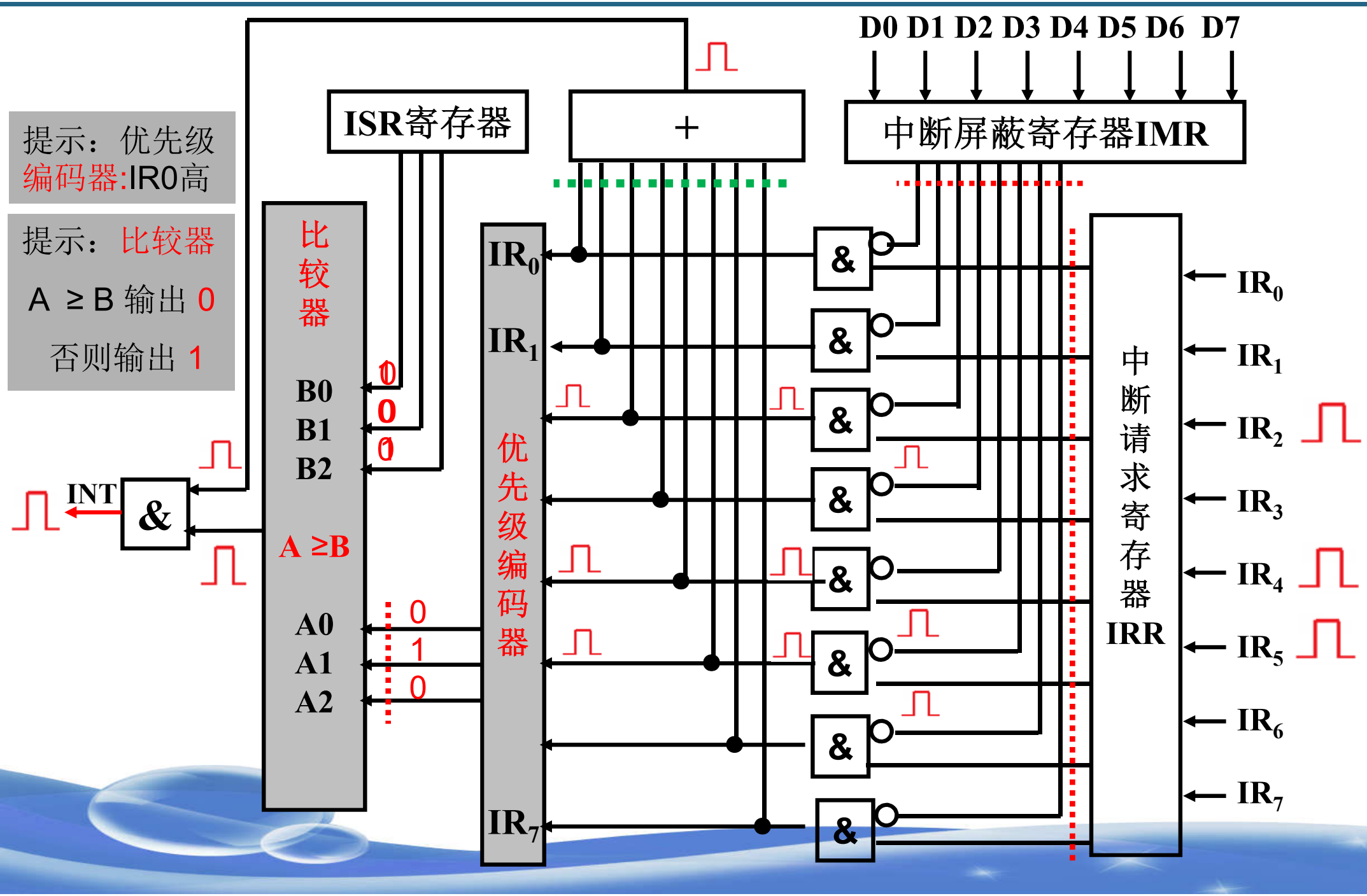
- ⑥中断控制逻辑
  - 产生向CPU输出的中断请求信号INT
  - 按收CPU送来的中断响应信号$\overline{INTA}$
  - 控制8259向DB上送出中断类型号N。



- ⑦ 优先权电路
  - 选择IRR中优先权最高的IR_i，并将其和ISR中的中断源比较，若IR_i优先级更高，则向CPU产生中断申请INT。
  - 优先权 IR_0 › IR_1…… › IR_7；禁止同级或低级中断，向高级中断开放。



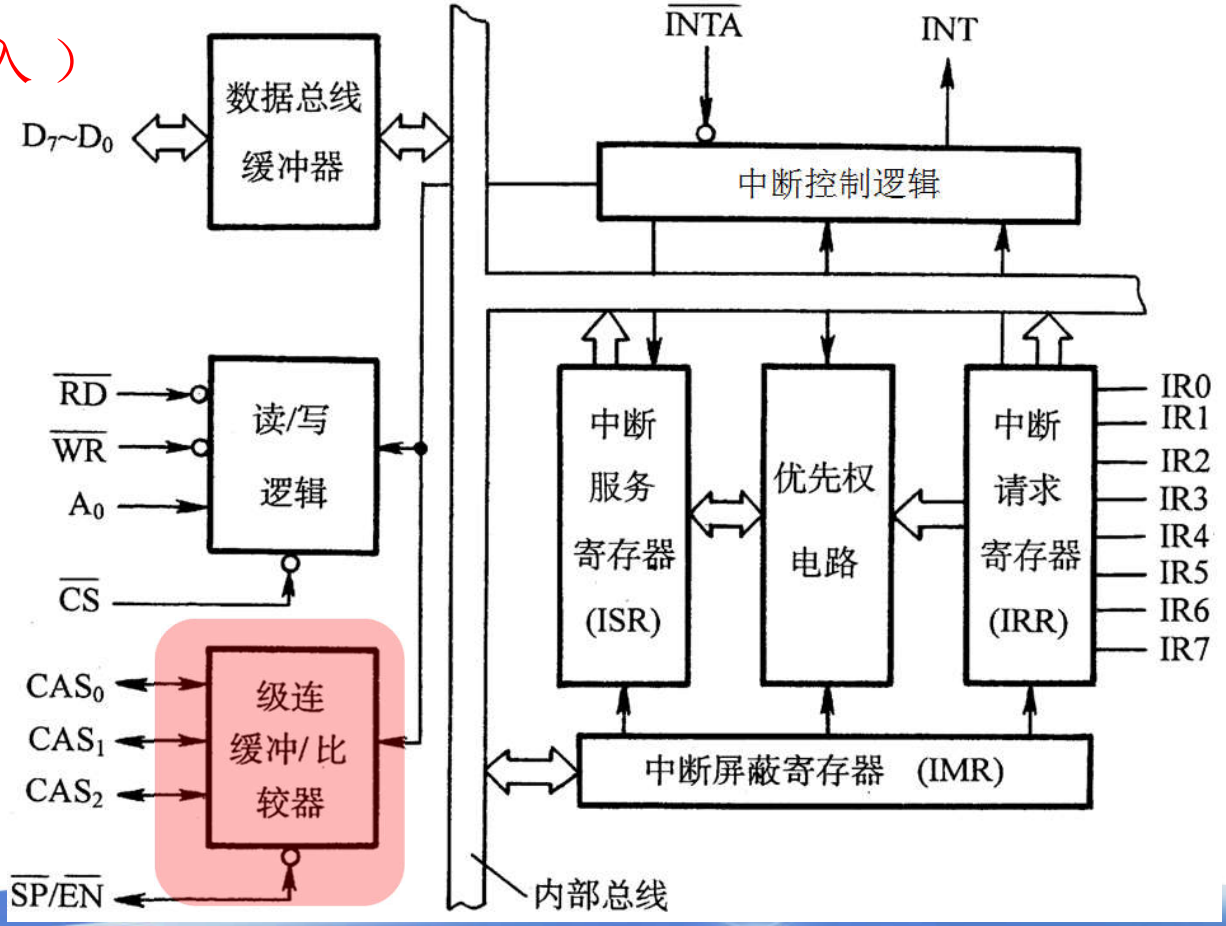
优先权电路（现有中断源如何阻止同级或低级中断源向CPU申请中断？）



优先权电路

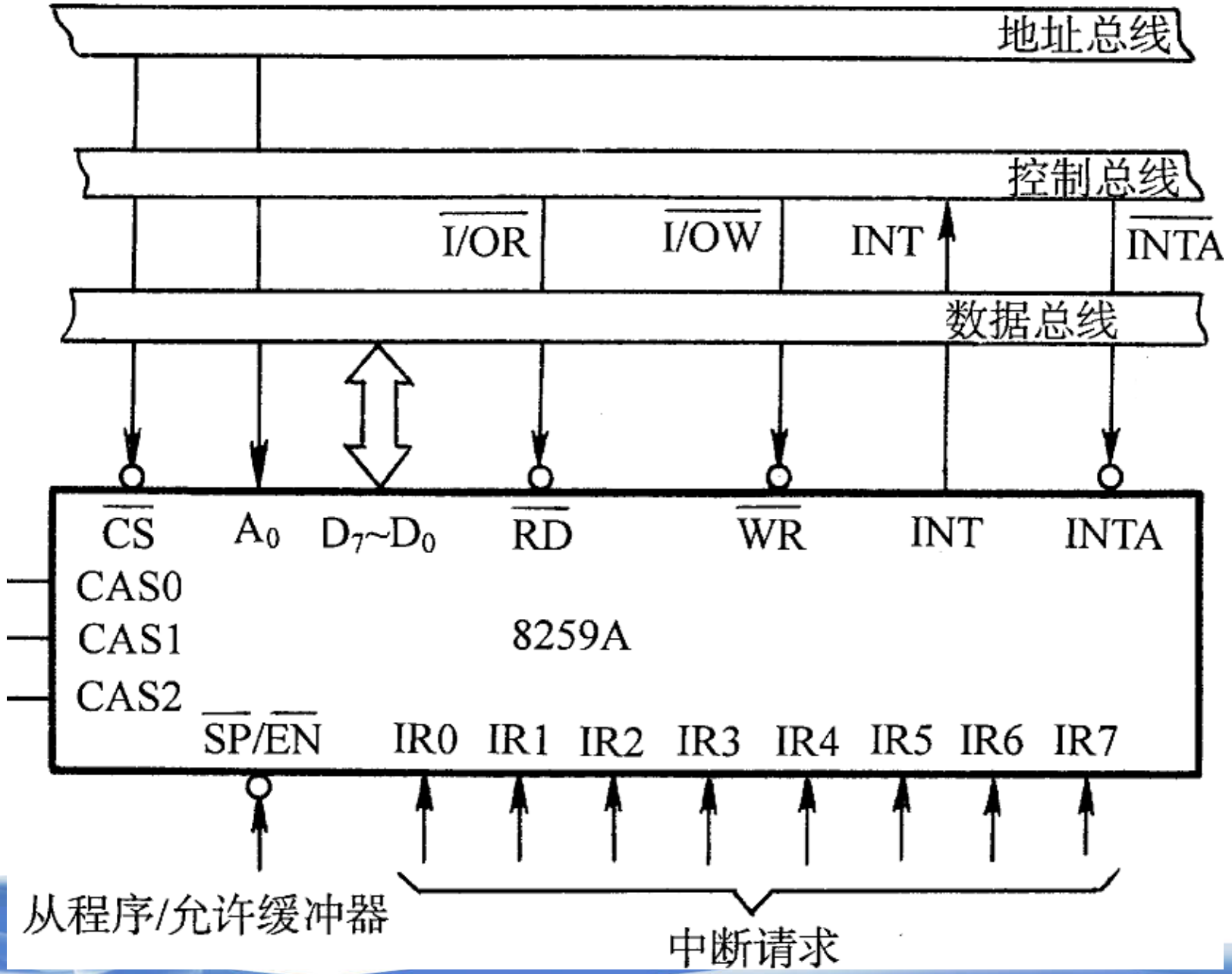
- 工作原理
  - 首先，由8个“与”门逻辑选出参加中断优先级排队的中断请求级：由8位IRR与8位IMR分别送入“与”门输入端，只有当IRR位置“1”（有中断请求）且IMR位置“0”（开放中断请求）同时成立时，相应“与”门输出才为高电平， 并送到优先级编码器的输入端参加编码。
  - 其次，优先级编码器对参加排队的中断优先级进行编码，并从中选出当前最高优先级的代码（A2，A1，A0）。
  - 最后，把ISR中当前正服务的优先级的代码（B2，B1，B0）与新来的中断请求的优先级代码（A2，A1，A0）一起送入比较器进行比较：当比较器输出有效且有中断请求时，与门将输出有效电平向CPU提出中断请求INT。
  - 结论：当一个中断源被服务时，它会禁止同级或低级中断请求的发生，但能向高一级的中断请求开放。

- ⑧级联缓冲/比较器
  - ■ $\overline{SP}/\overline{EN}$：从编程/缓冲器允许信号。
    - ◆采用缓冲方式时（输出）
      - □输出：控制数据总线缓冲器的使能；
    - ◆采用非缓冲方式（输入）
      - □区分主从片：
        - ▲主片=1,从片=0.
  - ■ $CAS_{2\sim0}$：级联信号线
    - ◆主片：输出
    - ◆从片：输入



第4节 8259A中断响应过程

8259单片工作——非缓冲方式

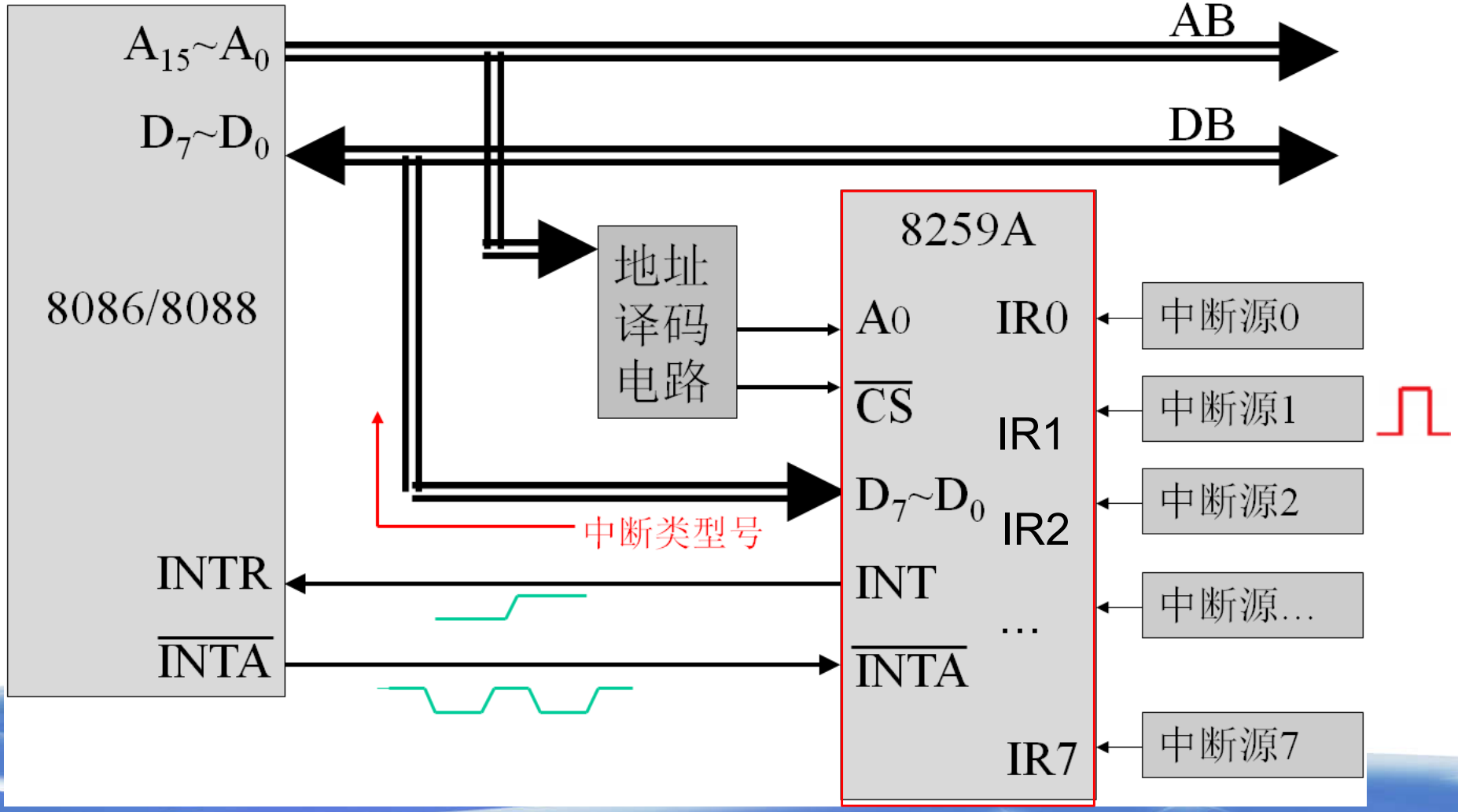


8259A中断响应过程（单片）

INTA=0←INTA=0 ←INTR=1 ← IMR$_i$=0 ← IRR$_i$=1

↓

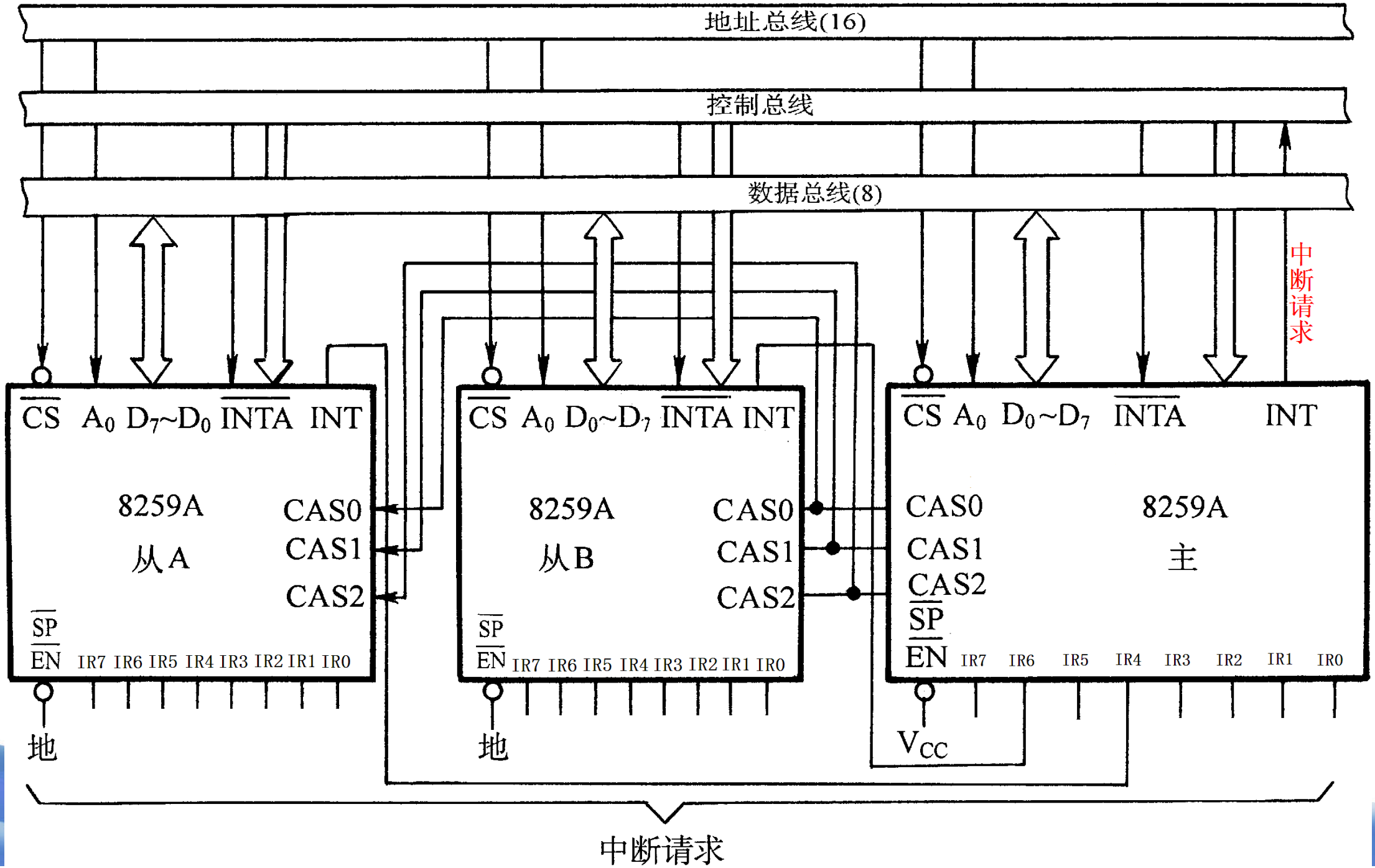
DB=N(i) → AB=N*4 → DB=CS':IP' → CS:IP



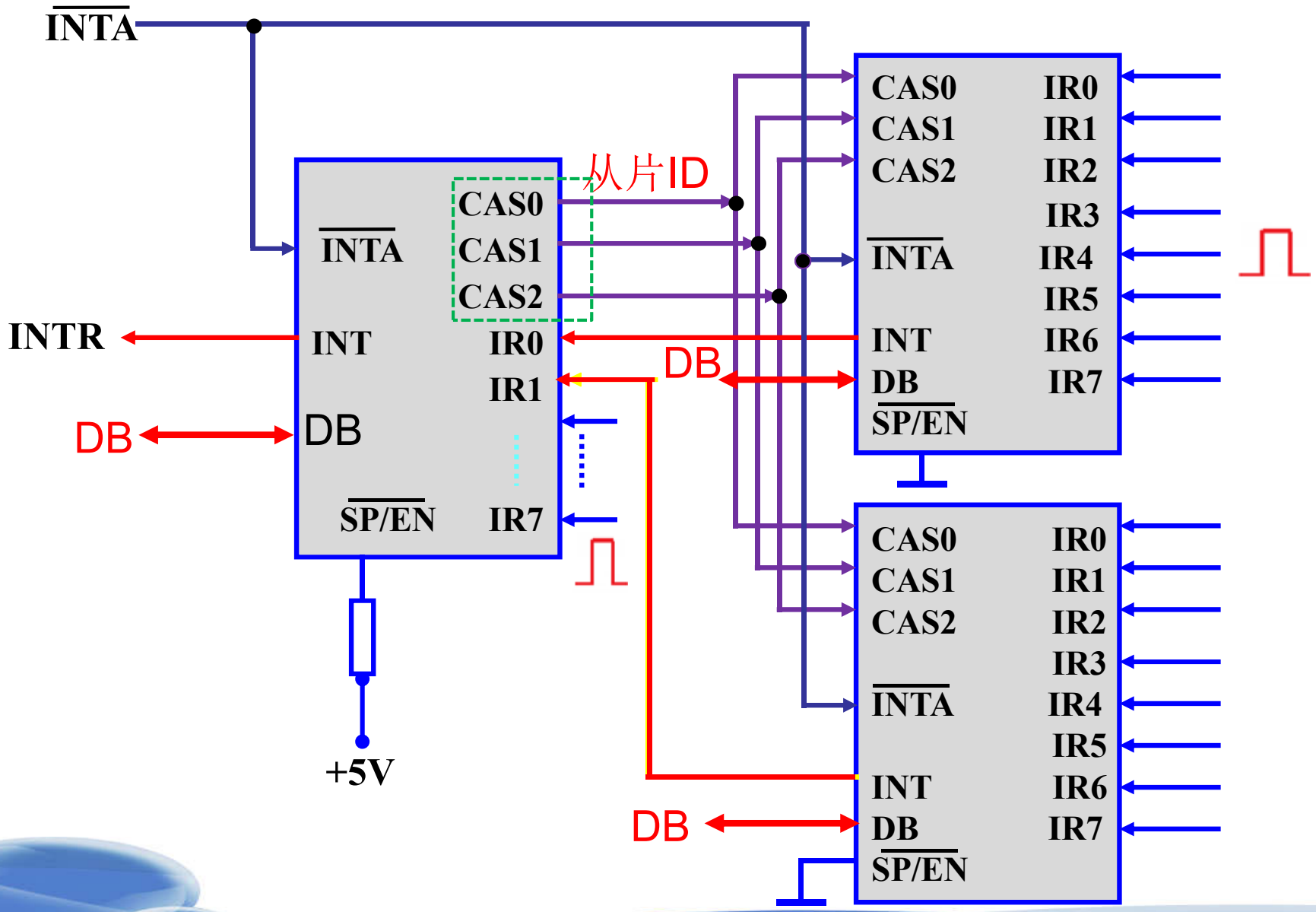
8259A中断响应过程（单片）

- 1.当IR_{7-0}有变高，中断请求寄存器IRR相应的位置1
- 2.若IMR相应位中断允许，则8259A通过INT向CPU送出中断请求
- 3.若CPU开中断，则用$\overline{INTA}$响应2个负脉冲。
- 4.当8259A收$\overline{INTA}$第1个负脉冲后
  - 使最高优先权的ISR位置位，相应IRR位复位。
    - 优先权的顺序为： IR0 › IR1…… › IR7
- 5.当8259A收第2个INTA负脉冲，8259A向DB送出中断类型号**N**。
- 6.CPU读取DB上的**N**，自动计算中断向量（CS:IP）的保存地址（N*4）,在AB上出现N*4，读取其中的中断向量（CS:IP）恢复到CS:IP中，实现进入中断服务程序。

8259多片级联——非缓冲方式



8259A中断响应过程（多片）



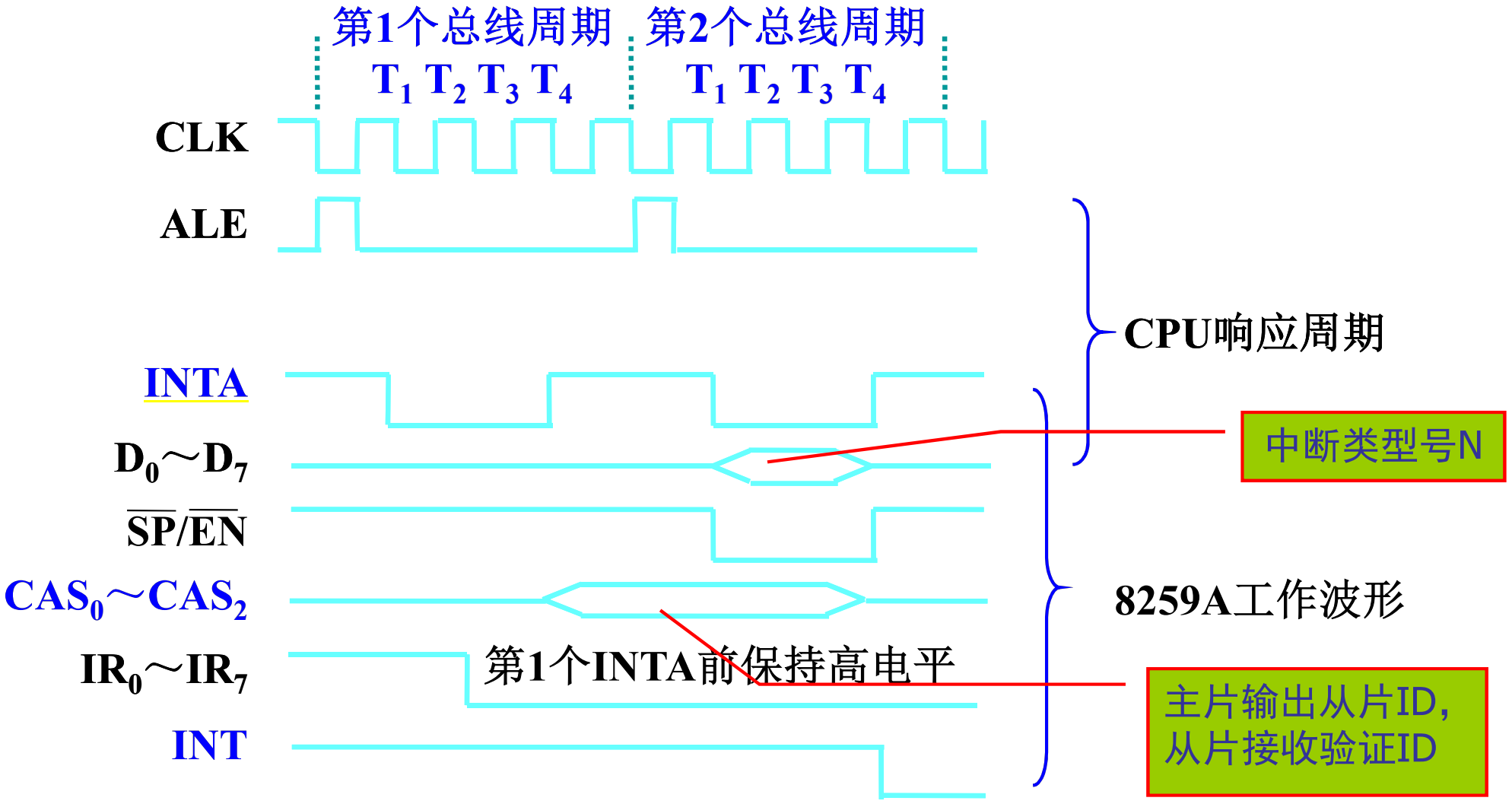
8259A中断响应过程（多片）

- 8259A可以级连，1个主片最多可以级连8个从片
  - 级连时，主片CAS_0～CAS_2连至每个从片的CAS_0～CAS_2，输出被选中的从片ID，
  - 每个从片的中断请求信号INT，连至主片8259A某个中断请求输入端IR_i；主片的INT线连至CPU的中断请求输入端INTR
  - 主片在第1个响应周期内通过$CAS_{2\sim0}$送出从片ID，相应的从片在第2个响应周期内则将中断类型码N发送到数据总线上。

8259A的工作原理/中断响应过程（多片）

- 1.当IR_{7-0}有变高，中断请求寄存器IRR相应的位置1
- 2.若IMR相应位中断允许，则8259A通过INT向CPU送出中断请求
- 3.若CPU开中断，则用$\overline{INTA}$响应2个负脉冲。
- 4.当8259A收$\overline{INTA}$第1个负脉冲后
  - 若不是主8259A的中断源产生了的中断，则主8259A通过CAS输出从片ID，通过验证的从8259A负责（下面第5步）响应CPU。
  - 使最高优先权的ISR位置位，相应IRR位复位。
    - ◆优先权的顺序为： IR0 › IR1…… › IR7
- 5.当8259A收第2个$\overline{INTA}$负脉冲，8259A向DB送出中断类型号**N**。
- 6.CPU读取DB上的**N**，自动计算中断向量（CS:IP）的保存地址（N*4）,在AB上出现N*4，读取其中的中断向量（CS:IP）恢复到CS:IP中，实现进入**中断服务程序**。

8259A的中断过程


第1个总线周期
T1 T2 T3 T4
第2个总线周期
T1 T2 T3 T4
CLK
ALE
INTA
D0～D7
SP/EN
CAS0～CAS2
IR0～IR7
INT
第1个INTA前保持高电平
CPU响应周期
中断类型号N
8259A工作波形
主片输出从片ID，
从片接收验证ID

8259A的端口和操作

- 2个端口
  - 按端口地址区分命令（偶地址A_0=0和奇地址A0=1）
  - 按顺序或特征位区分命令（同一端口地址）

$\overline{CS}$	$\overline{WR}$	$\overline{RD}$	A0	读写操作
0	0	1	0	写ICW1,OCW2,OCW3
0	0	1	1	写ICW2~ICW4,OCW1
0	1	0	0	读IRR,ISR,查询字
0	1	0	1	读IMR

第4节 8259A的工作方式

- 8259A的工作方式
  - 1.引入中断请求（中断触发）的方式
  - 2.连接系统总线的方式
  - 3.级联方式
  - 4.屏蔽中断源的方式
  - 5.优先级排队的方式
  - 6.结束中断的方式

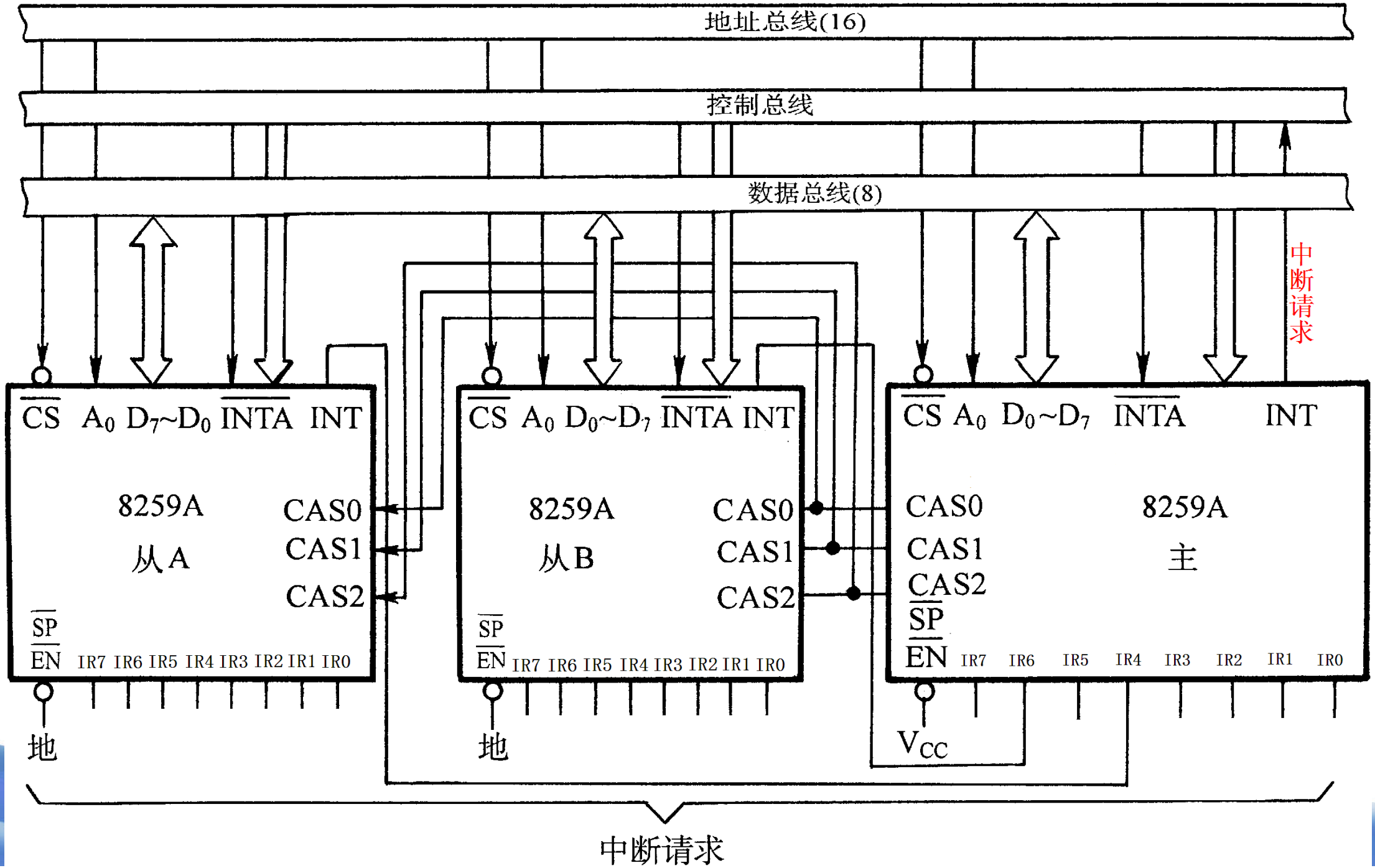
- 1.引入中断请求（中断触发）的方式
  - ①边沿触发方式。
    - ◆以正跳沿请求中断，维持高电平不会继续产生中断。

  - ②电平触发方式。
    - ◆以高电平申请中断，但响应中断后须及时清除高电平

  - ③注意：查询方式。
    - ◆CPU用软件查询确定中断源。外设通过8259申请中断，但不用INT向CPU申请中断。

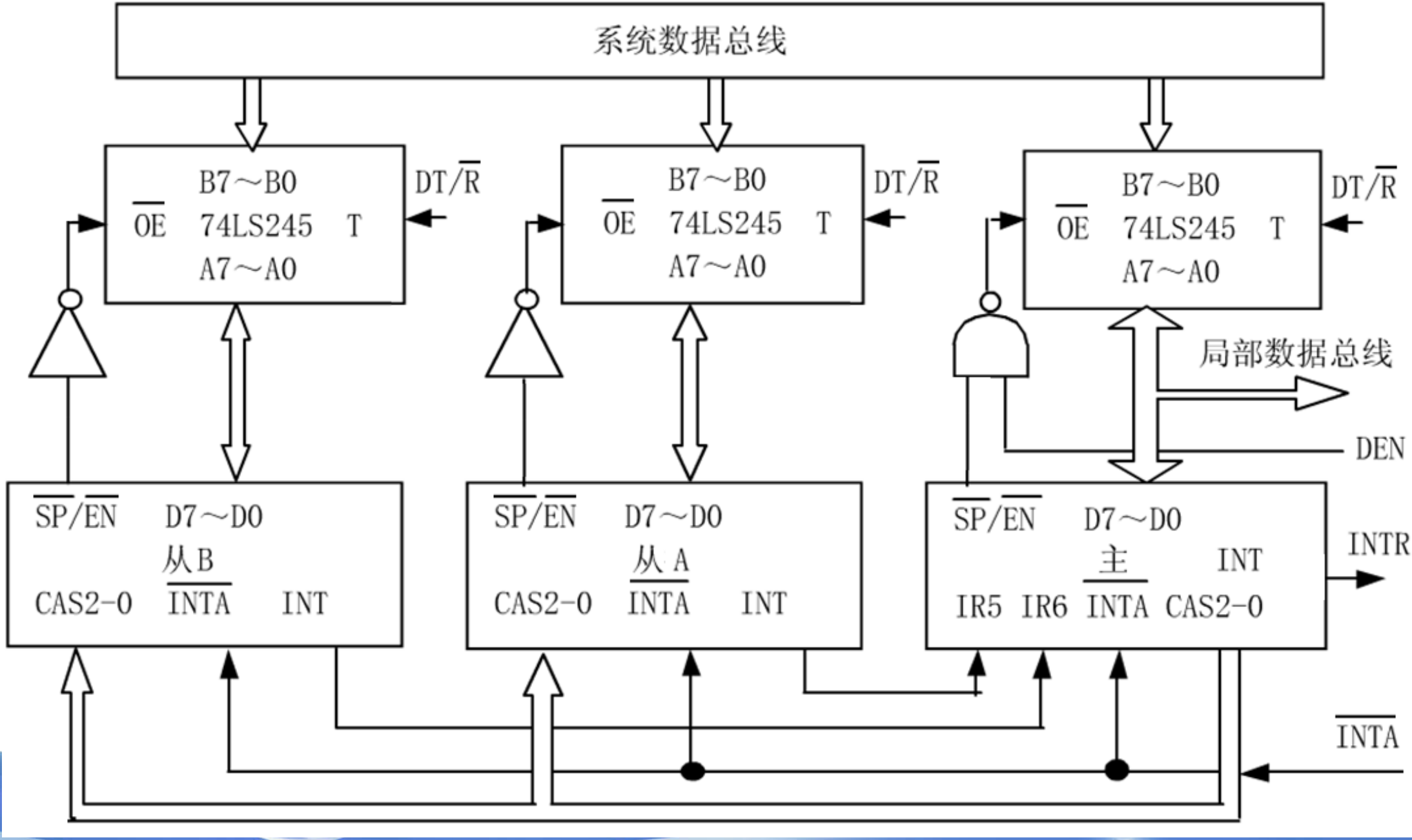
- 2. 连接系统总线的方式
  - 非缓冲方式
    - ◆数据总线直接接至CPU数据总线。
    - ◆$\overline{SP}/\overline{EN}$用于表示主/从芯片。

8259多片级联——非缓冲方式

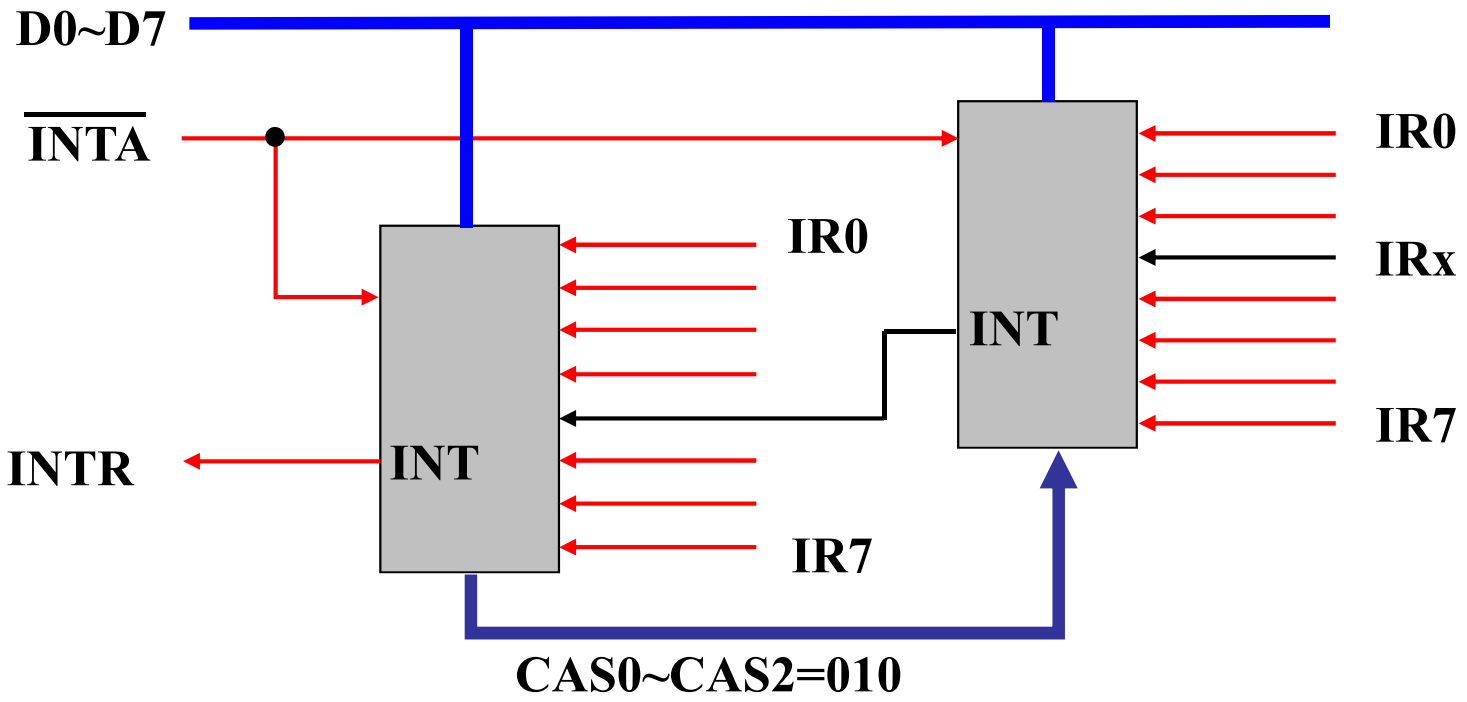


- 2. 连接系统总线的方式
  - 非缓冲方式
    - 数据总线直接接至CPU数据总线。
    - $\overline{SP}/\overline{EN}$用于表示主/从芯片。
  - 缓冲方式
    - 数据总线通过缓冲器与CPU数据总线连接。
    - $\overline{SP}/\overline{EN}$用于启动数据总线缓冲器，不表示主/从关系。

8259级联工作——多片缓冲方式

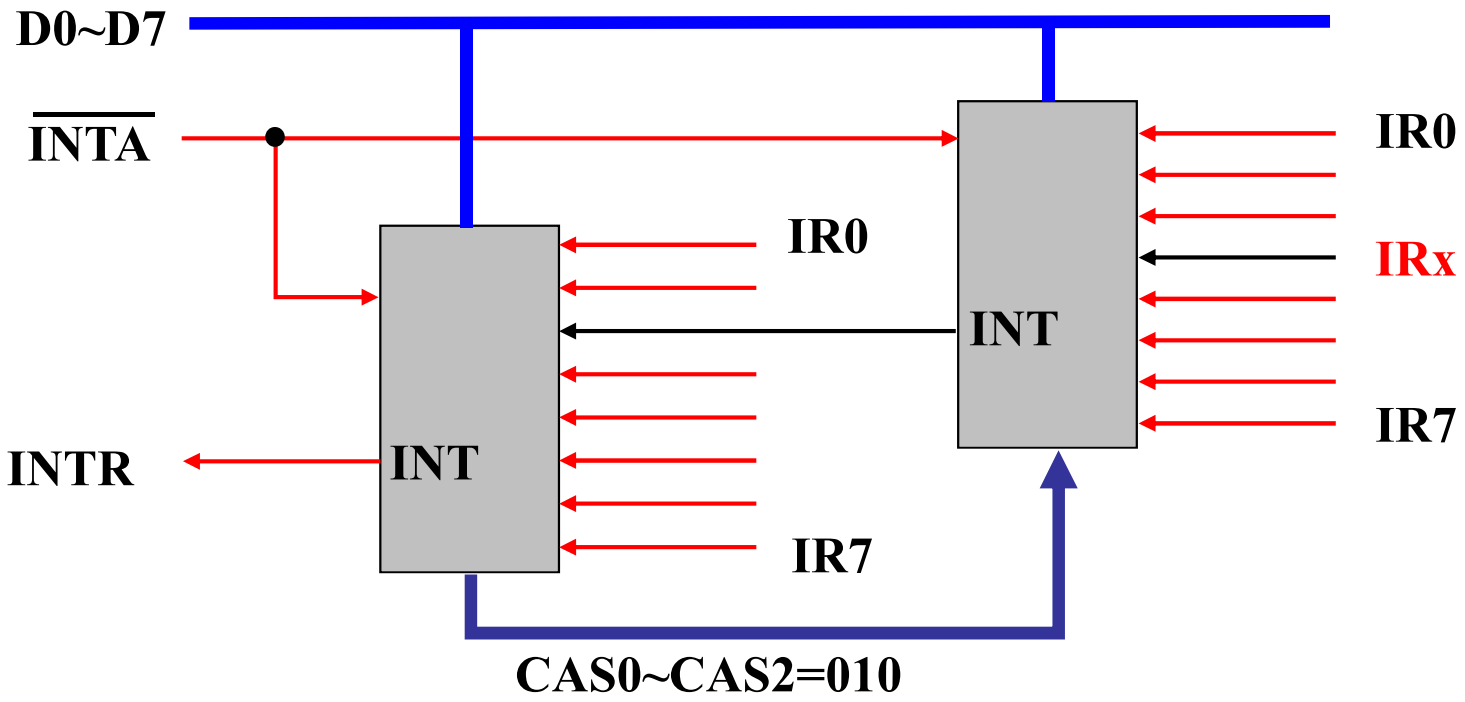


- 3.级联方式
  - 不级连——只用1片，如 PC/XT
  - 级连——使用2~9片，如 PC/AT（2片）



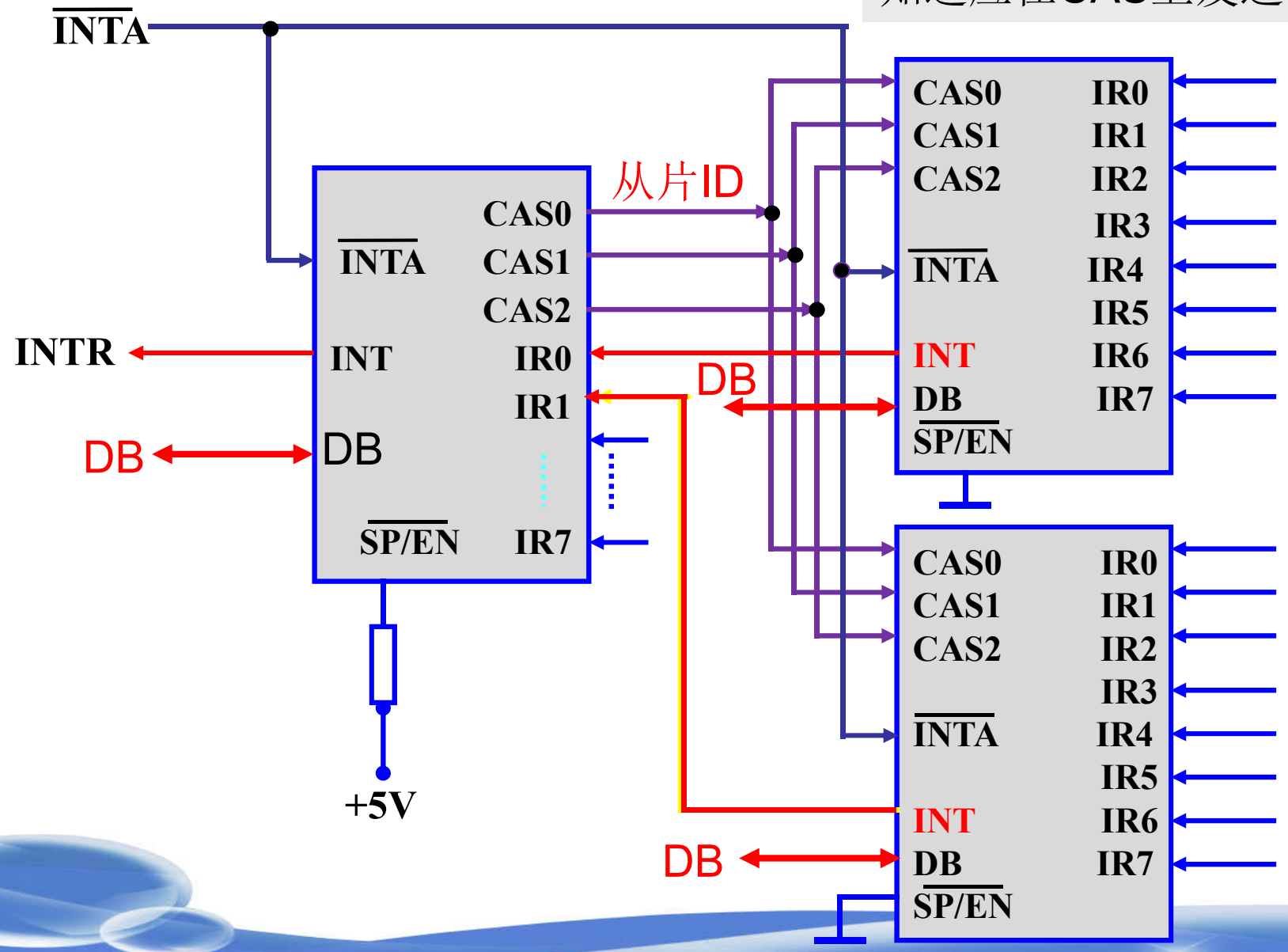
- 3.级联方式
  - 不级连——只用1片，如 PC/XT
  - 级连——使用2~9片，如 PC/AT（2片）

思考：从片的某个IRx发生中断时，主片如何知道在CAS上发送那个从片的ID？



- 3.级联方式

思考：某从片IRx中断时主片如何知道应在CAS上发送哪个从片ID？



- 4.屏蔽中断源的方式
  - ■ ①通常屏蔽方式
    - ◆利用操作命令字OCW1，设置屏蔽寄存器IMR
    - ◆例： OCW1 ：1111 0011 开放IR3，IR2两个中断源。
  - ■ ②特殊屏蔽方式
    - ◆在某些场合，当执行某一个高优先级的中断服务程序时，若要求允许另一个优先级比它低的中断请求被响应，此时可采用特殊屏蔽方式。它可通过OCW3的D6D5=11来设定。

- 5.优先级排队的方式
  - ① 全嵌套方式。常用缺省方式
    - ◆优先级按0～7顺序排队，且只允许级别高的中断源去中断级别低的中断服务程序。
  - ② 自动轮换方式
    - ◆中断服务结束后优先级降为最低（7），相邻的低优先级中断源自动升为最高，其余顺变。例：IR_2中断服务结束后:

IR0	IR1	IR2	IR3	IR4	IR5	IR6	IR7
5	6	7	0	1	2	3	4

    - ◆每个中断源都有最高优先级的资格，故称“等优先级方式”
  - ③优先级指定轮换方式
  - ④ 特殊循环方式

- 6.结束中断的处理方式
  - ①自动中断结束方式
    - ◆在第二个$\overline{\text{INTA}}$响应信号中，8259自动执行中断结束EOI指令，复位ISR中已置位的位。
    - ◆EOI：End Of Interrupt，中断结束指令
  - ②非自动中断结束方式
    - ◆在中断服务程序返回之前，必须发中断结束EOI命令才能使ISR中的当前服务位清除。

第5节 8259的编程和应用

8259的编程

- 8259的编程分为两个阶段
  - ■初始化阶段
    - ◆在系统加电或复位后进行。
    - ◆设定工作方式、工作条件、中断类型码或级联方式等。
    - ◆初始化命令（Initialize Control Word ）
      - □ICW1 ～ICW4
  - ■操作控制阶段
    - ◆对8259的状态、中断方式和工作过程的控制。
    - ◆操作命令（Operation Control Word ）
      - □OCW1～OCW3

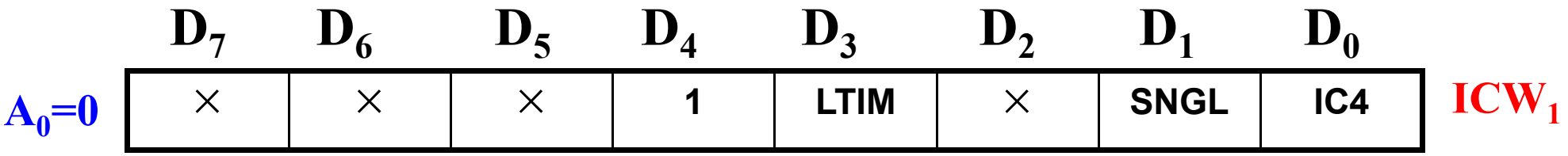
- ICW/OCW操作功能表

类型	$\overline{CS}$	$\overline{WR}$	$\overline{RD}$	A_0	功能	特征标志或流程
写命令字	0	0	1	0	数据总线→ICW_1	ICW_1的D_4为1
	0	0	1	0	数据总线→OCW_2	OCW_2的D_4 D_3为00
	0	0	1	0	数据总线→OCW_3	OCW_3的D_4 D_3为01
	0	0	1	1	数据总线→OCW_1 (屏蔽字)	无
	0	0	1	1	数据总线→ICW_2~ICW_4	ICW设置流程
读状态	0	1	0	0	IRR→数据总线	OCW_3的$D_2D_1D_0$为010
	0	1	0	0	ISR→数据总线	OCW_3的D_2D_1 $_0$为011
	0	1	0	0	中断查询字→数据总线	OCW_3的$D_2D_1D_0$为100
	0	1	0	1	IMR (屏蔽字) →数据总线	无

初始化命令（ICW: Initialize Control Word）

- ICW1
  - 设置中断请求触发方式及芯片数目，使8259A复位
- ICW2
  - 设置中断类型号N（的起码）：8位
- ICW3
  - 设置主从片的级联方式
- ICW4
  - 设置优先级嵌套方式，中断结束方式，缓冲方式，主从片

初始化编程流程



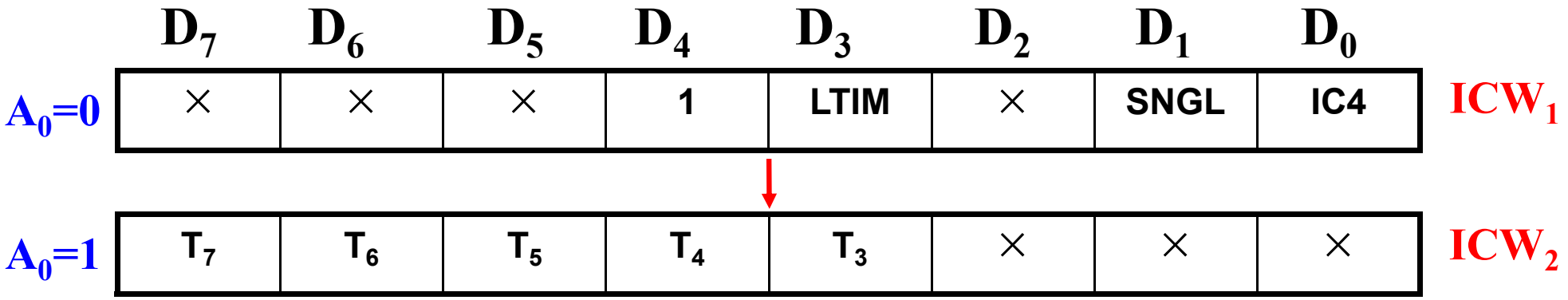
D4：1特征位

LTIM：触发中断方式。 1：电平触发； 0：边沿触发；

SNGL：是单片还是级联。1：单片；0：级联；

IC4：是否使用ICW4。 1：使用；0不使用

初始化编程流程

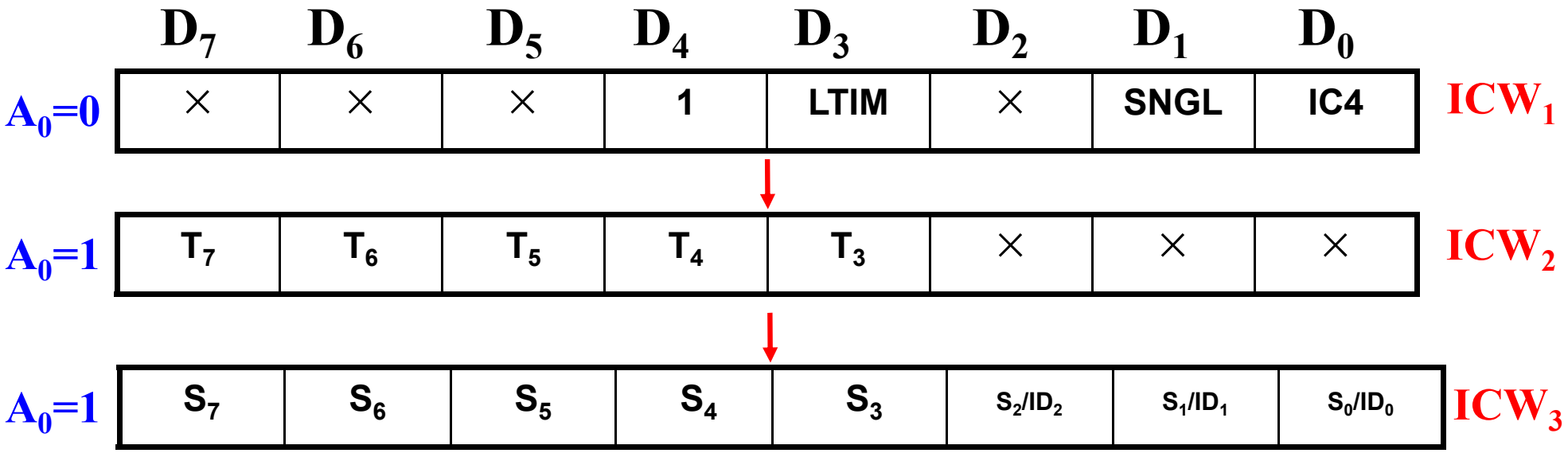


$D_{7\text{-}3}$：中断类型码高5位： T_7~T_3 ，编程写入

$D_{2\text{-}0}$：$IR_{7\sim0}$编码值，直接写0。 由8259A自动填入；

提示： 同片8259A的中断源的中断类型号N高5位相同，
N的低3位根据IR_i的引脚编号：000~111

初始化编程流程

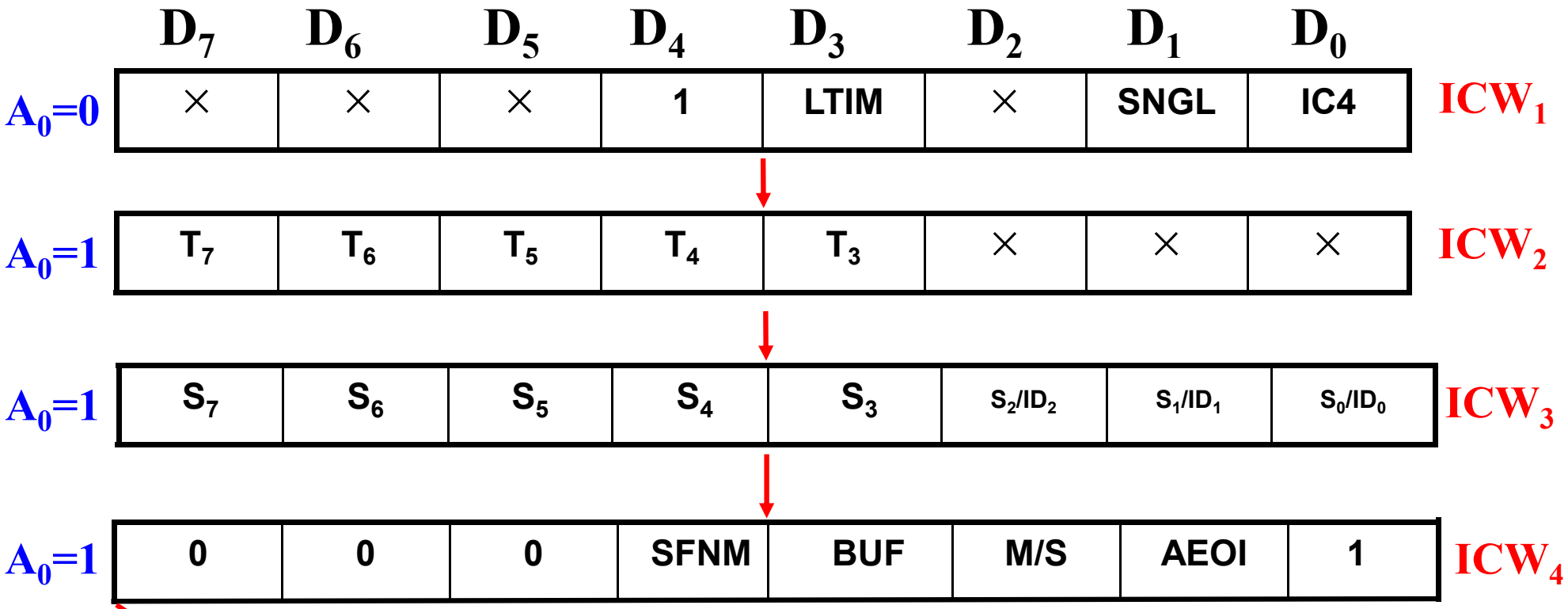


主片：S_7~S_0。IR_7~IR_0上是否连有从片；

从片：ID_2~ID_0。所连主片的IR_i引脚编号；

思考：主片如何知道该向CAS局部总线发送哪个从片的ID？从片如何知道的**ID**并将之与**CAS**上的**ID**验证？

初始化编程流程



SFNM：优先级排队。 1：特殊全嵌套方式，0：全嵌套方式

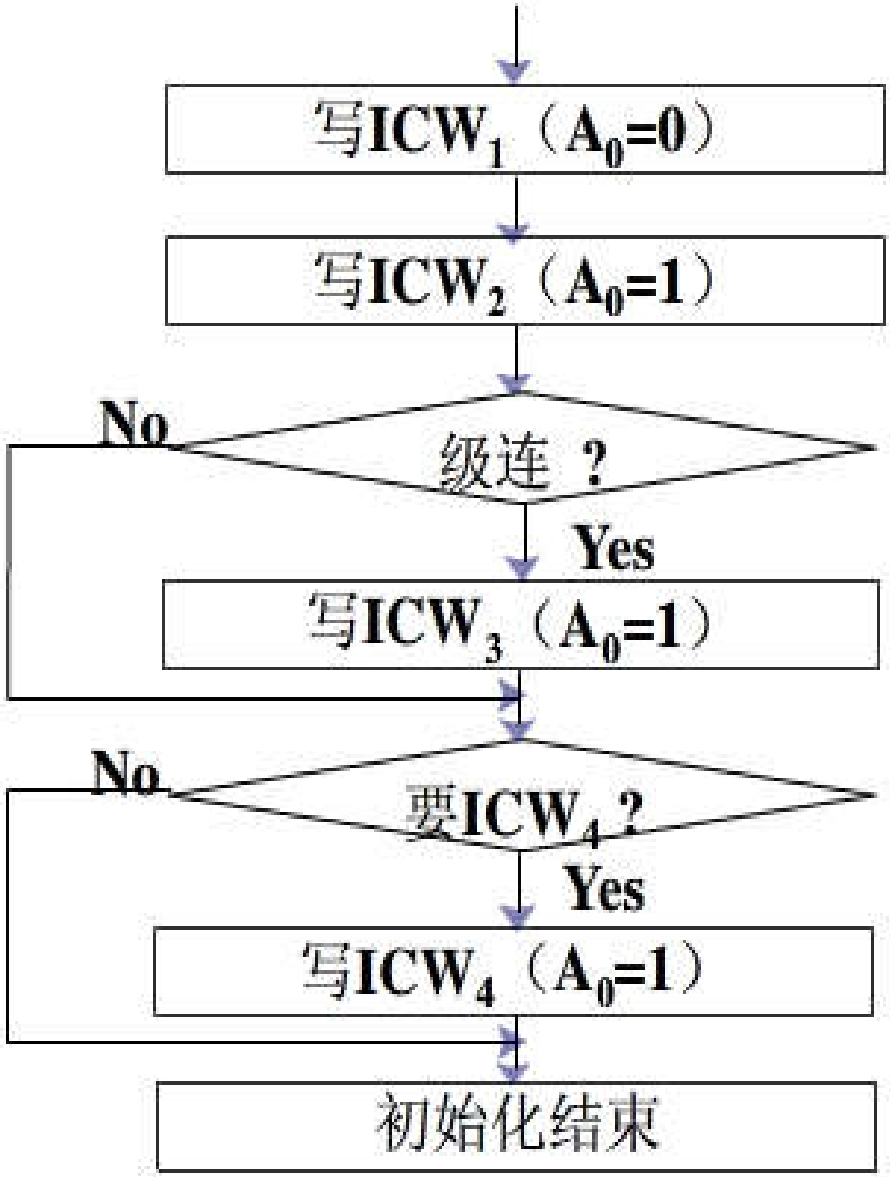
BUF：总线连接方式。1：缓冲，0：非缓冲

M/S：区分主从片。 1：主片，0：从片

AEOI：结束中断方式。1：自动，0：非自动

补充说明

- ICW的识别
  - 只要命令字D4位为“1”，地址位A0为“0”，就是ICW1。
  - 接下来1~3字节就是ICW2~ICW4。
- 在不同的初始化要求中，ICW2~ICW4并非都必须使用。
- ICW1复位芯片
  - ICW1清除IMR，缺省设置完全嵌套方式，$IRQ_{0\sim7}$优先级递降。



初始化的例子

- 例：一片8259A，边沿触发方式；中断类型码为08H~0FH；
  - 用全嵌套、缓冲、非自动结束中断方式；
  - 8259A的端口地址为20H和21H。

```
MOV   AL   ,13H  ;ICW1：边沿触发，单片，设置IC4
OUT   20H  ,AL

MOV   AL   ,8    ;ICW2：中断类型码为08~0FH
OUT   21H  ,AL

MOV   AL  ,0DH   ;ICW4：全嵌套、缓冲、非自动结束中断方式
OUT   21H  ,AL
```

初始化的例子：ICW1

- 例：一片8259A，边沿触发方式；中断类型码为08H~0FH；
  - 用全嵌套、缓冲、非自动结束中断方式；
  - 8259A的端口地址为20H和21H。

```
MOV  AL  , 13H   ; ICW1：边沿触发，单片，设置IC4
OUT  20H  , AL
```

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
A_0=0	×	×	×	1	LTIM	×	SNGL	IC4	ICW_1
	0	0	0	1	0	0	1	1	

初始化的例子：ICW2

- 例：一片8259A，边沿触发方式；中断类型码为08H~0FH；
  - 用全嵌套、缓冲、非自动结束中断方式；
  - 8259A的端口地址为20H和21H。

```
MOV  AL   ,8        ；ICW2：中断类型码为08~0FH
OUT  21H  ,AL
```

	T_7	T_6	T_5	T_4	T_3	×	×	×	
A_0=1	0	0	0	0	1	0	0	0	ICW_2

思考：如果中断类型码为 020~027H，如何设置ICW2 ？

初始化的例子：ICW4

- 例：一片8259A，边沿触发方式；中断类型码为08H~0FH；
  - 用全嵌套、缓冲、非自动结束中断方式；
  - 8259A的端口地址为20H和21H。

MOV AL,0DH ;ICW4：全嵌套、缓冲、非自动结束中断
OUT 21H,AL

A_0=1	0	0	0	SFNM	BUF	M/S	AEOI	1	ICW_4
	0	0	0	0	1	1	0	1	

- 8259的操作命令（OCW：Operation Control Word）
  - 在初始后执行，可不按顺序进行
  - OCW1：中断屏蔽操作命令
  - OCW2：优先级方式选择/结束操作命令
  - OCW:3：中断查询操作

◆操作命令字OCW_1——中断屏蔽/允许字

设置中断源IR_i的中断屏蔽/允许。

•例子：编程：屏蔽IR0,IR1,IR7三个中断源

```
a) MOV   AL, 83H      ; 屏蔽IR0，IR1，IR7，开放IR2~IR6
   OUT   21H, AL      ; 写OCW1（屏蔽字）
```

```
b) IN    AL, 21H      ; 读IMR（总是有效）
   OR    AL, 83H      ; 屏蔽IR0，IR1，IR7，保留其他的
（或 AND  AL, 83H ）  ; 开放IR2~IR6，保留其他的
   OUT   21H, AL      ; 写OCW1（屏蔽字）
```

◆操作命令字OCW_2——中断方式字

D_7：为1，循环优先级方式；为0，固定优先级方式。

D_6：为1，D_2~D_0(L_2~L_0)有效；为0，D_2~D_0(L_2~L_0)无效。

D_5：为1，EOI命令；为0，非EOI命令。

D_2~D_0：ISR中的位编码。

•例子：中断服务程序中发出EOI指令

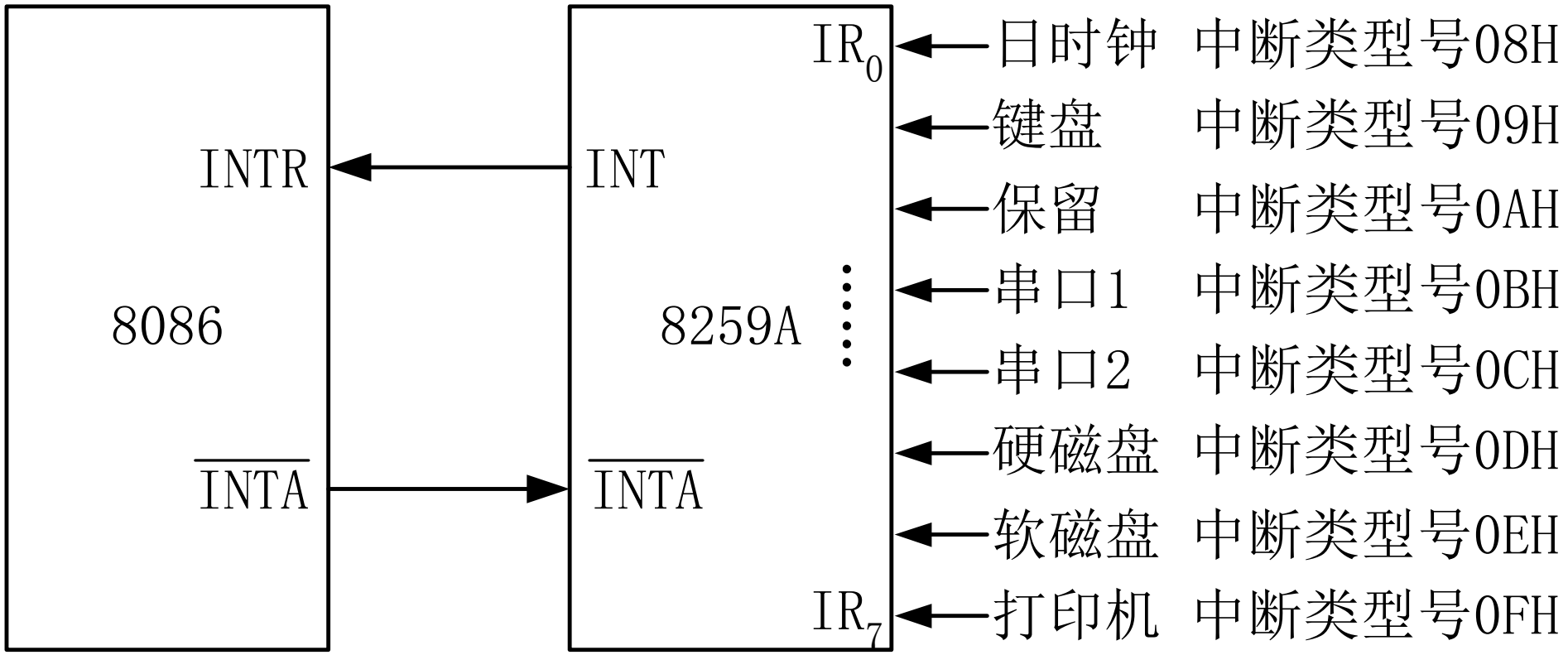
```
MOV  AL, 20H        ；设置OCW2 的EOI 命令字
OUT  20H, AL        ；写OCW2
IRET                ；必须在中断结束前设置EOI
```

- ICW/OCW操作功能表

类型	$\overline{CS}$	$\overline{WR}$	$\overline{RD}$	A_0	功能	特征标志或流程
写命令字	0	0	1	0	数据总线→ICW_1	ICW_1的D_4为1
	0	0	1	0	数据总线→OCW_2	OCW_2的D_4 D_3为00
	0	0	1	0	数据总线→OCW_3	OCW_3的D_4 D_3为01
	0	0	1	1	数据总线→OCW_1 (屏蔽字)	无
	0	0	1	1	数据总线→ICW_2~ICW_4	ICW设置流程
读状态	0	1	0	0	IRR→数据总线	OCW_3的$D_2D_1D_0$为010
	0	1	0	0	ISR→数据总线	OCW_3的D_2D_1 $_0$为011
	0	1	0	0	中断查询字→数据总线	OCW_3的$D_2D_1D_0$为100
	0	1	0	1	IMR (屏蔽字) →数据总线	无

8259A在IBM-PC/XT中的应用

- 单片，端口地址：20H和21H
- N = 0x08-0x0F
- 边沿触发，非自动中断结束方式，完全嵌套方式，缓冲方式



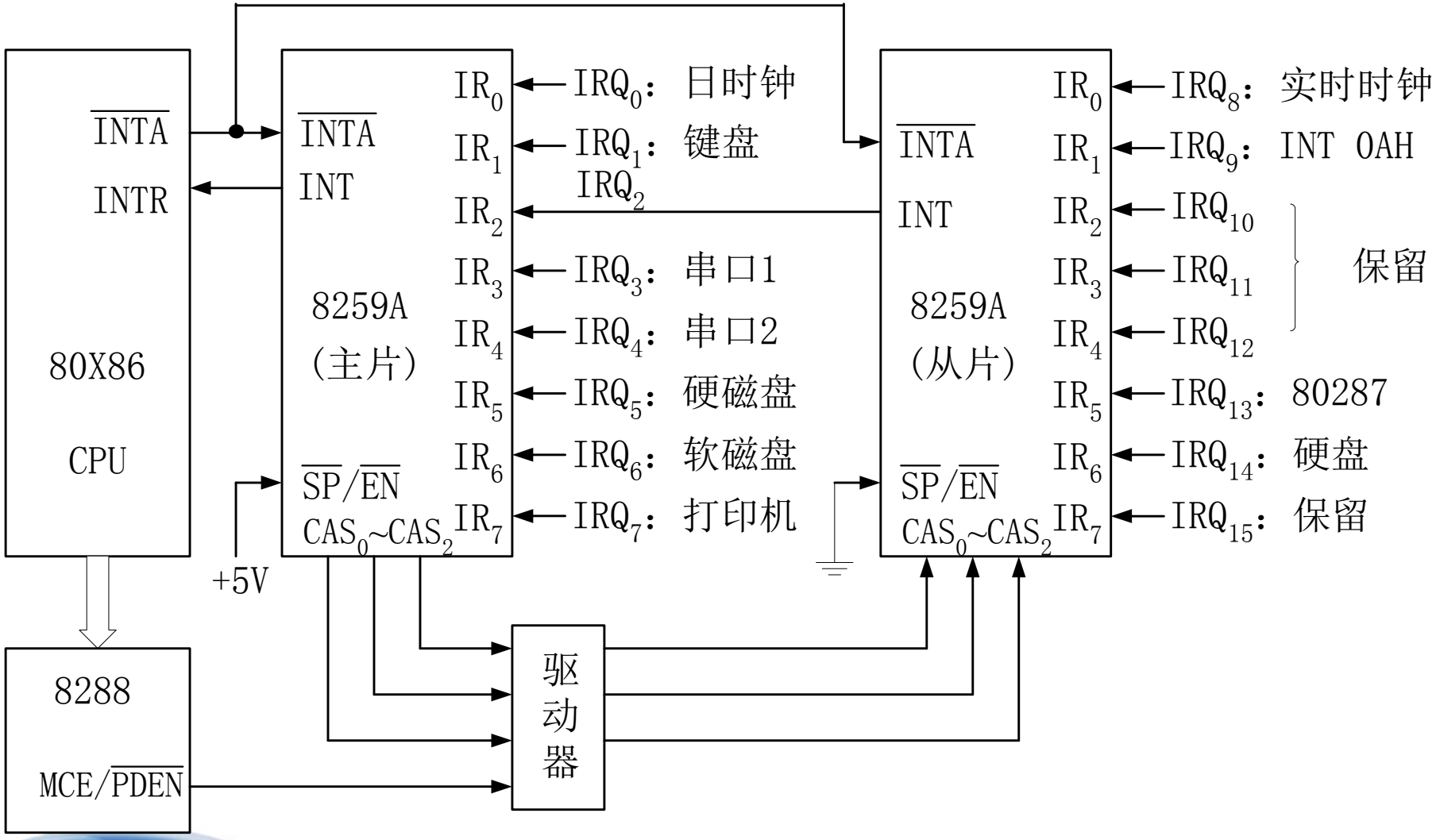
- 初始化编程

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
×	×	×	1	LTIM	×	SNGL	IC4	ICW_1
T_7	T_6	T_5	T_4	T_3	×	×	×	ICW_2
S_7	S_6	S_5	S_4	S_3	S_2/ID_2	S_1/ID_1	S_0/ID_0	ICW_3
0	0	0	SFNM	BUF	M/S	AEOI	1	ICW_4

```
;ICW1，边沿触发，单片8259A，需ICW4
MOV   AL    ,00010011B
OUT   20H   ,AL
;设置ICW2，中断类型号高5位为00001
MOV   AL    ,00001000B
OUT   21H   ,AL
;设置ICW4，非自动中断结束方式，完全嵌套方式，缓冲方式
MOV   AL   ,00001101B
OUT   21H   ,AL
```

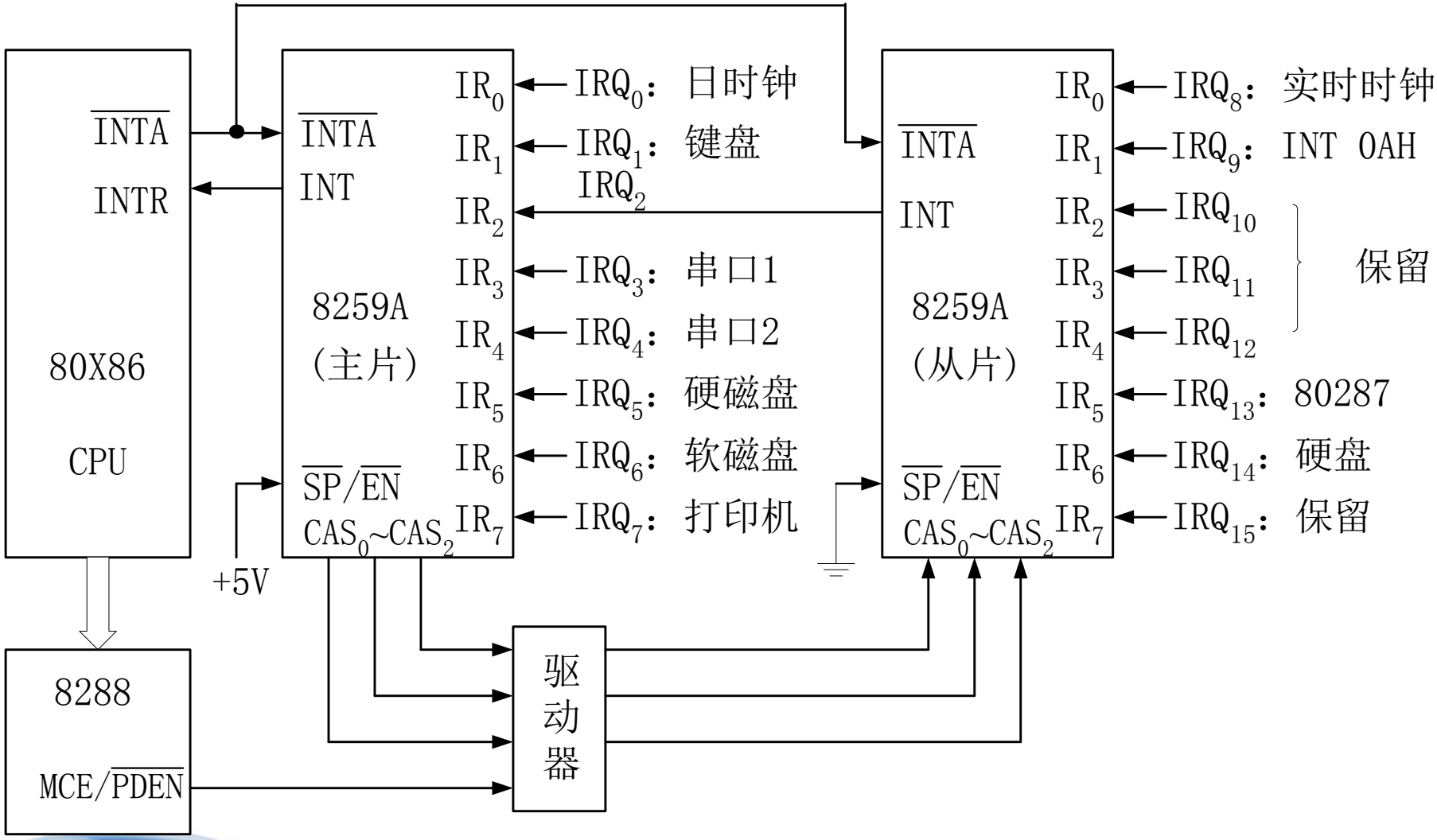
8259A在IBM-PC/AT的应用

- 两片8259A，地址：20H,21H ； A0H， A1H



例子：写出两片8259A的初始化程序

- 两片8259A，地址：20H,21H ； A0H， A1H



- 主片8259A

MOV AL,_____；设置ICW1，边沿触发，需ICW4

OUT ____ ,AL

MOV AL,____ ；设置ICW2，中断类型号的高5位为00001

OUT ____,AL

MOV AL,_____；设置ICW3，从片连到主片的IR_2上

OUT _____ ,AL

MOV AL, ___ ； 设置ICW4，非缓冲，非AEOI，特殊全嵌套方式

OUT ____,AL

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
×	×	×	1	LTIM	×	SNGL	IC4	ICW_1
T_7	T_6	T_5	T_4	T_3	×	×	×	ICW_2
S_7	S_6	S_5	S_4	S_3	S_2/ID_2	S_1/ID_1	S_0/ID_0	ICW_3
0	0	0	SFNM	BUF	M/S	AEOI	1	ICW_4

- 从片**8259A**

MOV AL,_____ ； 设置ICW1，边沿触发，需ICW4

OUT ____ ,AL

MOV AL, _____ ； 设置ICW2，中断类型号的高5位为01110

OUT _____,AL

MOV AL, _____ ； 设置ICW3，设定从片级联于主片的IR_2

OUT _____ ,AL

MOV AL, _____ ； 设置ICW4，非缓冲，非AEOI，全嵌套方式

OUT _____ ,AL

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
×	×	×	1	LTIM	×	SNGL	IC4	ICW_1
T_7	T_6	T_5	T_4	T_3	×	×	×	ICW_2
S_7	S_6	S_5	S_4	S_3	S_2/ID_2	S_1/ID_1	S_0/ID_0	ICW_3
0	0	0	SFNM	BUF	M/S	AEOI	1	ICW_4

80x86中断程序的编写过程

- 前提：1片8259芯片，N=40~47．IR5连接有某外设
- 任务：初始化8259并编写IR5的中断服务程序INTService 。
- 过程
  - 1.初始化8259A
  - 2.初始化中断向量表
  - 3.编写中断服务程序
  - 4.使能CPU的IF

1.初始化8259A

```
;ICW1，边沿触发，单片8259A，需ICW4
MOV   AL    ,00010011B
OUT   20H   ,AL
;设置ICW2，中断类型号高5位为0100  0XXX
MOV   AL    ,01000000B
OUT   21H   ,AL
;设置ICW4，非自动中断结束方式，完全嵌套方式，缓冲方式
MOV   AL   ,00001101B
OUT   21H   ,AL
```

2.初始化中断向量表

- 将用户自定义的中断服务程序入口地址放入向量表
- 例：将中断类型码N = XX H的服务程序入口地址放入向量表

2.初始化中断向量表

- 将用户自定义的中断服务程序入口地址放入向量表
- 例：将中断类型码N = 45 H的服务程序入口地址放入向量表

```
MOV AX，0000H
MOV DS，AX      ；数据段从内存0地址开始（安排向量表）
MOV SI，114H    ；  45H x 4=114H
MOV BX，OFFSET INTService   ；IP
MOV [SI]，BX
MOV BX，SEG INTService    ；CS
MOV [SI+2]，BX
```

3.编写中断服务程序

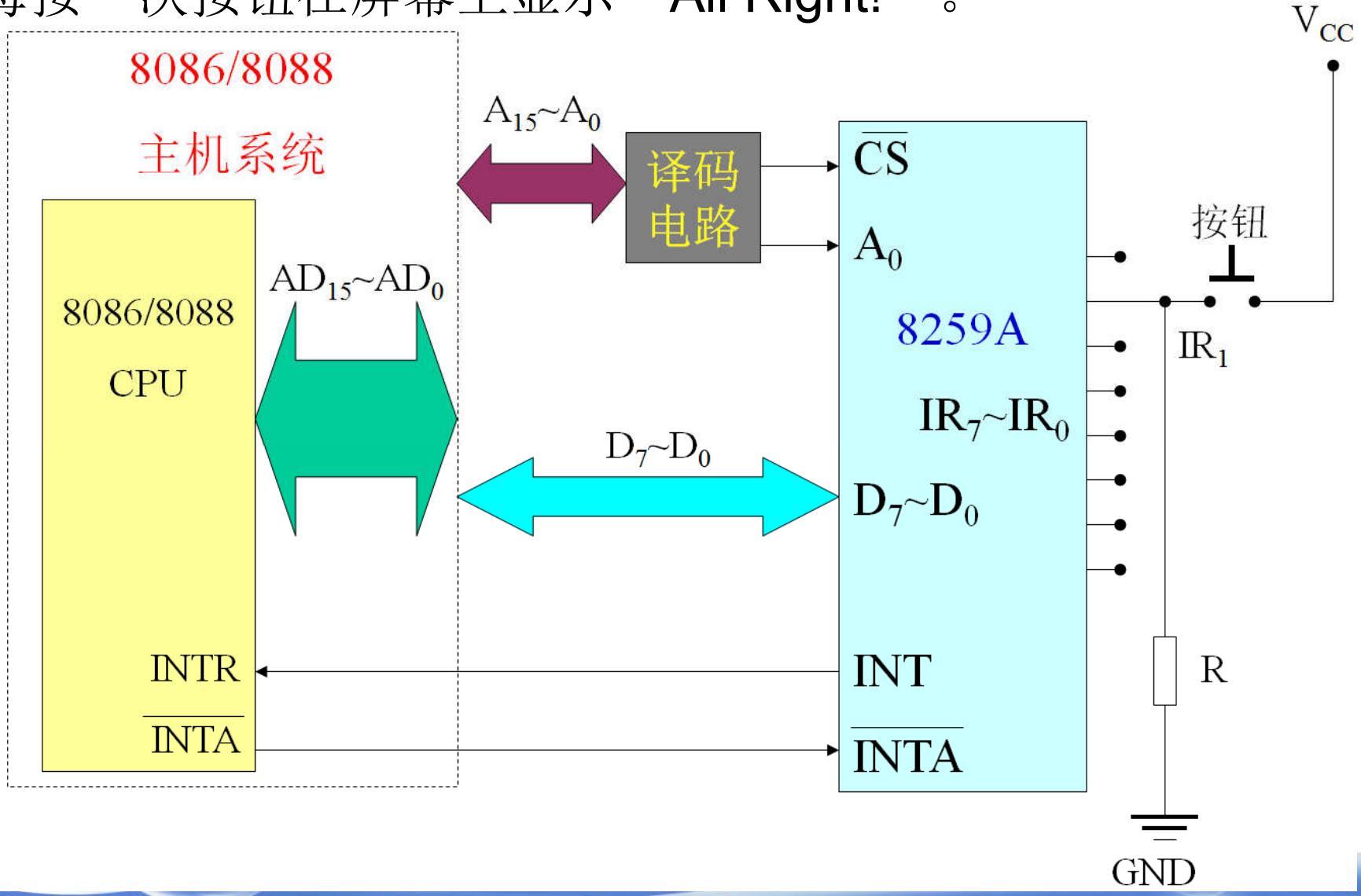
```
DATA   SEGMENT
   MESS   DB   'This is a 8259A interrupt!',0Ah,0Dh, '$'
DATA   ENDS
INTService PROC NEAR
        LEA    DX, MESS      ; 显示字符串
        MOV    AH, 09H
        INT    21H
        DEC    BL
        MOV    AL, 20H       ; 发送中断结束命令：EOI
        OUT    20H, AL
        IRET
INTService ENDP
```

4.使能CPU的IF

STI

例子：8259A应用

- 8259A地址20H和21H，IR_1接自复位开关，中断类型号是9。
- 每按一次按钮在屏幕上显示“All Right!”。



```
stack1 segment stack
        dw 20h dup(0)
stack1 ends
code segment
CharBuf db 'All Right',0Ah,0Dh,'$'      ;显示字符串
        assume cs:code,ss:stack1
start:  mov dx,seg CharInt
        mov ds,dx
        mov dx,offset CharInt
        mov al,9h                       ;中断类型号
        mov ah,25h
        int 21h                         ;装中断向量表
```

```
        mov al,0fdh
        out 21h,al              ;OCW1，置IMR＝1111,1101B
        sti                     ;开中断
next:   hlt                     ;等待外部中断
        JMP next
        mov ah,4ch              ;程序结束，返回DOS
        int 21h
CharInt: lea dx,CharBuf         ;中断服务程序入口
        mov ah,9h
        int 21h                 ;显示字符串
        mov al,20h                  ; OCW2=0010,0000B
        out 20h,al              ; OCW2，发中断结束命令
        iret
```

Code ends

end start

单片、全嵌套、普通屏蔽方式、优先权自动循环、非自动结束方式

ICW_1 ICW_2 ICW_4 $\boldsymbol{OCW_1}$ $\boldsymbol{OCW_2}$