第5.3　中断技术

- 主要内容：

一、基本概念

1.　中断的功能

2.　中断的分类

3.　中断的响应过程

二、可编程中断控制器8259A

【课前思考】

1. 什么中断?
2. 为什么要在微机系统中引入中断技术?
3. 微机系统是如何利用中断技术的?
4. 什么是中断向量和中断向量表?
5. 什么是中断服务程序?
6. 什么是中断响应？它包括哪些过程?

【学习目标】

掌握中断相关的概念及其好处；了解实模式与保护模式中断技术的区别与联系；了解 中断与异常的区别与联系；了解中断控制器8259的组成结构、寄存器模型、级联方式；掌握中断应用程序的编写方法

【学习指南】　学习中断，首先要理解其相关的概念，如中断向量、中断向量表、中断优先级、中断服务程序等，然后了解中断控制器与CPU之间的关系，以及如何初始化中断控制器。最后通过实例程序了解中断执行的过程，体会中断技术的优势。

【难点提示】　中断相关概念的理解；中断的处理过程；中断控制器与CPU的关系；中断控制器组成与初始化。

【内容提要】　中断是一种数据传输方式，本章首先介绍了中断的相关概念，然后介绍其工作原理，接着以中断控制器8259为例，介绍了其组成结构、工作方式以及与CPU之间的联接。最后以实例程序介绍了中断程序的编写方法。

5.3.1 中断的概念与类型

5.3.1.1 中断的概念

CPU在正常运行程序时由于内部/外部事件，或由程序事先安排的事件发生，而被中途打断，且转到为事件服务的程序中去，服务完毕，再返回原程序中。

中断系统应具有的功能：

中断接受功能 ：

接受来自外部的中断请求，启动中断，由中断控制器实现。其中断触发方式可以用上升沿触发，也可以用电平触发。

中断响应功能:

对来自外部的中断请求予以回答，由微处理器经过外部支持电路给出2个中断回答（确认）信号来实现，并以此回答信号来读取中断源的中断号。

中断屏蔽功能 ：

两层: 允许/禁止中断请求，由中断控制器实现。

开中断/关中断，由微处理器完成

中断系统应具有的功能：

中断排队功能 ：

外部硬件中断排队，由中断控制器实现。系统中断排队，由系统安排。

中断嵌套功能:

高级别的中断源可以打断低级别的中断服务程序去为高级中断服务，由中断控制器完成。

中断返回功能 ：

服务完毕返回上一级程序，由微处理器完成。

中断共享功能:

采用路由器与中断控制器共同完成。

中断的引出

中断最初是作为处理器与外部设备交换信息的一种控制方式提出的。由此，最初的中断全部是对外部设备而言的，称为**外部中断或硬件中断**。

随着计算机技术的发展，中断的范围也随之扩大，出现了**内部软件中断**的概念，它是为解决机器内部运行时出现的**异常**以及为编程方便而提出的。

5.3.1.2 中断的类型

Intel 80x86CPU和Pentium系列CPU最多支持256种中断和异常

- 中断
  - 外部中断（中断）
    - 不可屏蔽中断 NMI
    - 可屏蔽中断 INTR
  - 内部中断（异常）
    - 软件中断
      - 单步、断点、溢出、零除
      - 指令INT n
    - 异常
      - 故障Fault
      - 陷阱Trap
      - 中止Abort

5.3.2 中断号

- 微处理器为每个不同类型的中断与异常分配一个中断号，以便识别和处理。
- 32位微处理器支持256个中断号，并且对中断与异常统一编号为0号～255号。
- 微处理器以提交的中断号为向导到中断向量表或中断描述符表IDT找到相应的中断/异常处理程序：

在实模式下：中断号×4 得一个指针，指向中断向量表，在中断向量表中可以找到中断服务程序的入口；

保护模式下：中断号×8 得一个指针，指向中断描述表IDT，在IDT中找到中断/异常处理程序的中断门/陷阱门描述符，然后通过门描述符获得中断/异常处理程序的入口。

32位微处理器的中断号分配表

中断号	中断/异常名称	中断/异常类型	出错代码	引起中断的引脚/引起异常的指令
0	除法错误	故障	无	DIV,IDIV
1	调试	故障/陷阱	无	任何指令
2	不可屏蔽中断			由引脚 NMI 引入的外部中断
3	断点	陷阱	无	INT 3
4	溢出	陷阱	无	INTO
5	边界检查	故障	无	BOUND
6	非法操作码	故障	无	非法指令编码或操作数
7	协处理器不可用	故障	无	ESC 或 WAIT
8	双重故障	中止	有	引起异常 0、10、11、12、13 的任何指令
9	协处理器段越界	中止	有	访问存储器的浮点指令
10	无效 TSS	故障	有	JMP、CALL、IRET 或 INT 指令
11	段不存在	故障	有	装载段寄存器的指令
12	堆栈错误	故障	有	堆栈操作指令
13	一般保护性错误	故障	有	任何特权指令、任何访问存储器的指令
14	页异常	故障	有	任何取指操作或访问存储器的指令
15	保留			
16	数字运算错误	故障	无	浮点指令或 WAIT
17	对齐检查	故障	有	用户代码的任何访存指令
18—31	保留			
32—255	软件中断	陷阱	无	INT n
	可屏蔽中断			由引脚 INTR 引入的外部硬件中断

16位微处理器的中断号分配表

中断号	名　　称	中断号	名　　称
0	除零数	25H	磁盘扇区读
1	单步	26H	磁盘扇区写
2	NMI	27H	程序终止驻留
3	断点	28H	等待状态处理
4	溢出	29H	字符输出处理
5	屏幕打印	2AH	保留
6	保留	2BH	保留
7	保留	2CH	保留
8	日时钟中断	2DH	保留
9	键盘中断	2EH	命令执行处理
0AH	保留／从片中断	2FH	多路复用处理
0BH	串行口2中断	30H	内部使用
0CH	串行口1中断	31H	内部使用
0DH	硬盘／并行口2中断	32H	保留
0EH	软盘中断	:	:
0FH	打印机/并行口1中断	:	:
10H	视频显示I/O	67H	用户保留
11H	设备配置检测	68H	保留
12H	内存容量检测	:	:
13H	磁盘I/O	6FH	保留
14H	串行通信I/O	70H	保留／实时钟中断
15H	盒带／多功能实用	71H	保留／改向INT0AH
16H	键盘I/O	72H	保留／保留
17H	打印机I/O	73H	保留／保留
18H	ROM-BASIC	74H	保留／保留
19H	磁盘自举	75H	保留／协处理器中断
1AH	日时钟/实时钟I/O	76H	保留／硬盘中断
1BH	Ctrl-Break 中断	77H	保留／保留
1CH	定时器报时	78H	未使用区
1DH	视频显示方式参数	:	:
1EH	软盘基数表	7FH	
1FH	图形显示扩展字符	80H	BASIC 使用区
20H	程序终止退出	:	:
21H	系统功能调用	EFH	
22H	程序结束地址	F0H	内部使用区
23H	Ctrl-c出口地址	:	:
24H	严重错误出口地址	FFH	

5.3 .3中断优先级

系统优先级

内部中断和异常

软件中断

外部非屏蔽中断

外部可屏蔽中断

排队方式：

1. 按优先级排队 ——根据任务的轻重缓急

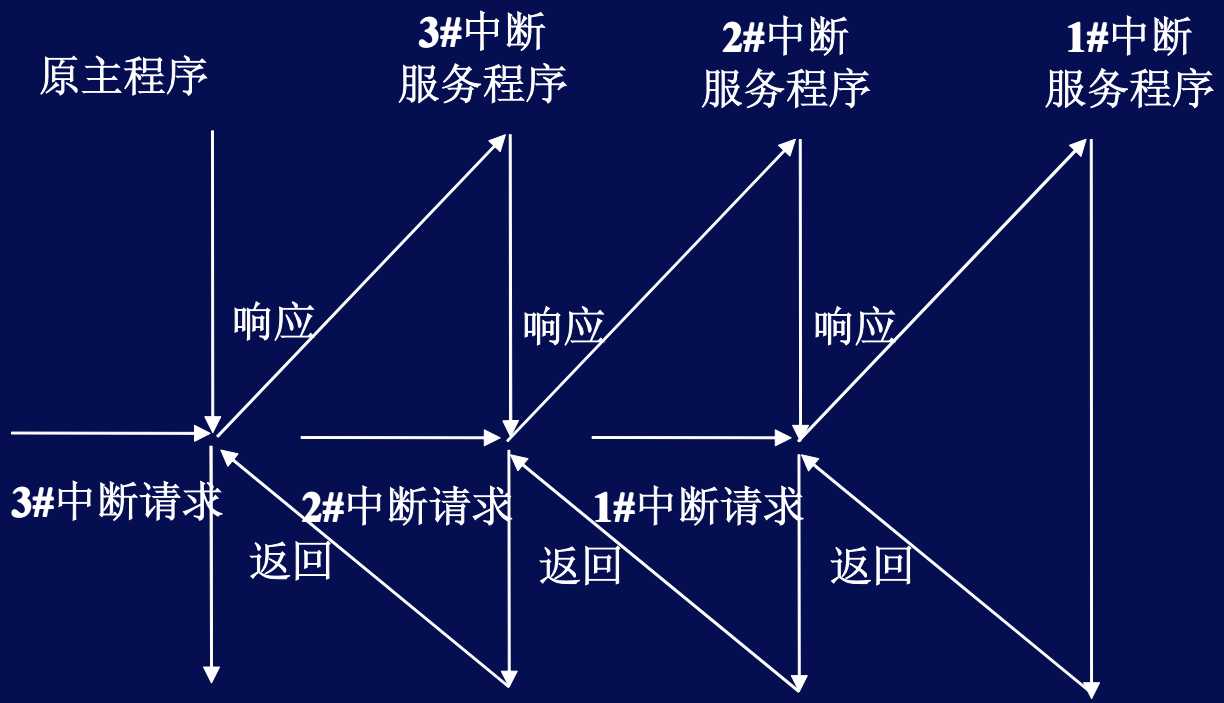
2. 循环轮流排队——CPU轮流响应各个中断源的中断请求

中断嵌套

当CPU正在响应某一中断源的请求，执行为其服务的中断服务程序时，如果有优先级更高的中断源发出请求，CPU将中止正在执行的中断服务程序而转入为新的中断源服务，等新的中断服务程序执行完后，再返回到被中止的中断服务程序，这一过程称为**中断嵌套**。

中断嵌套可以有多级，具体级数原则上不限，只取决于堆栈深度。

☞中断的优先级与中断嵌套



中断优先级1#>2#>2#

5.3.4 中断向量与中断向量表

- 每个中断类型对应一段中断服务程序，如何进入中断服务程序？**获得中断服务程序的入口地址是关键。**
- 在**实模式**引入了中断向量及中断向量表，通过中断向量表中的中断向量获取入口地址，
- 在**保护模式**引入了中断门描述符及中断门描述符表IDT，通过IDT中的中断门描述符获取入口地址。

5.3.4 中断向量与中断向量表

实模式下：

中断向量——中断服务程序的入口地址

即中断服务程序的**段基址CS，偏址IP**

共4个字节

中断向量表——所有的中断向量集中存放到存储器的某一区域，该表称为**中断向量表**。

中断类型号与中断向量指针

中断向量表包含256个中断向量。

每个中断向量包含两个字（4个字节），

高地址字为中断服务程序所在代码段的段基址，

低地址字为中断服务程序第一条指令的偏移量。

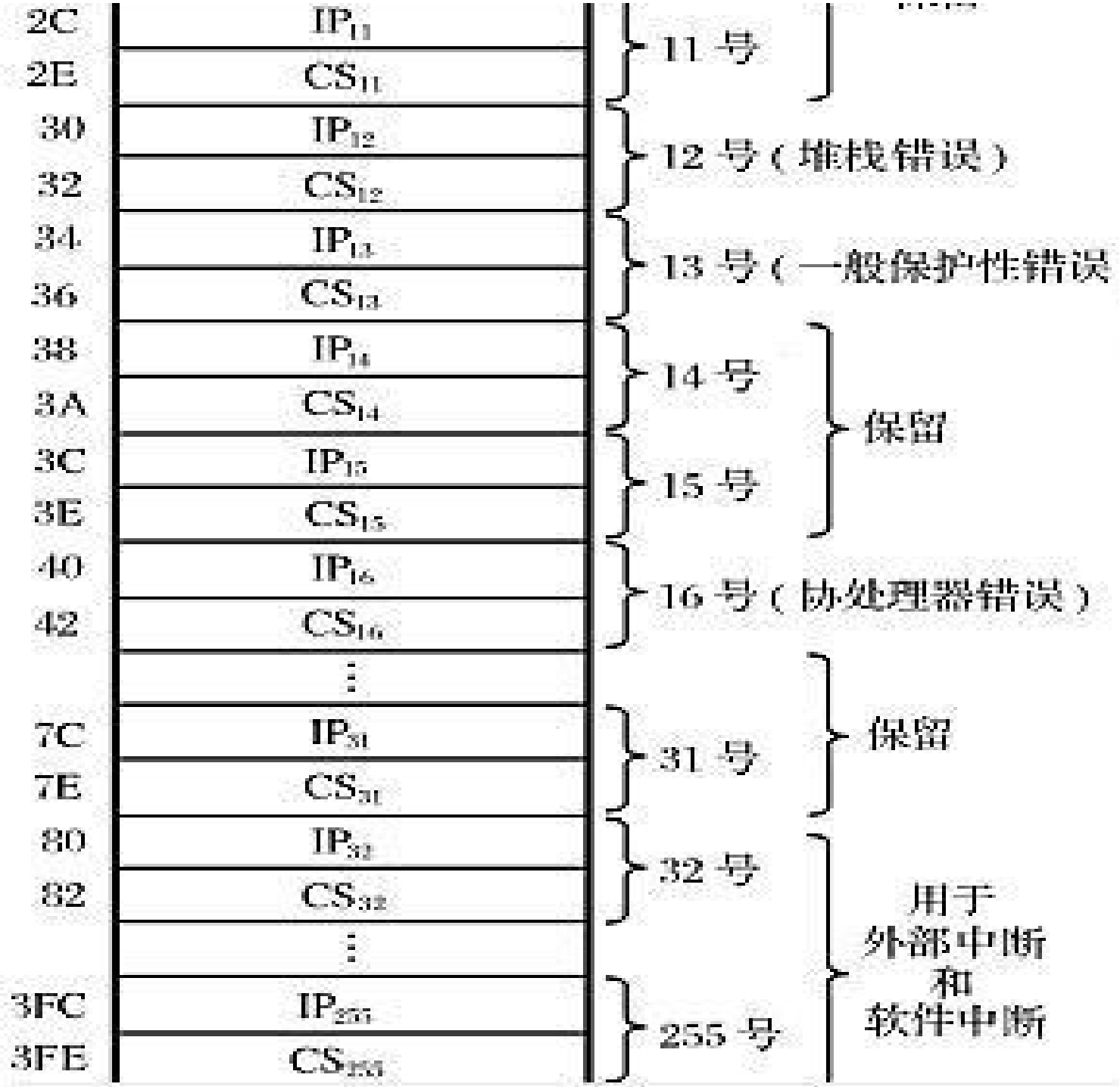
实模式下，中断向量表存放在内存最低端的1K单元之中，物理地址00000H～003FFH

中断向量指针:

指出中断向量存放在中断向量表的位置(或地址)

32位微处理器实模式下的中断向量

32位微处理器实模式下的中断向量

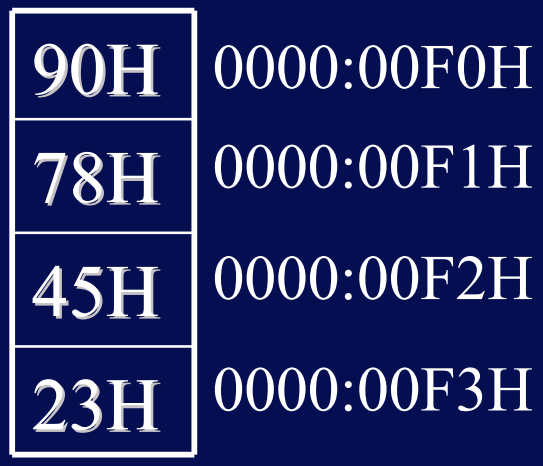


中断类型码和中断向量所在位置之间的对应关系

向量地址 = 中断向量最低字节的指针 = 中断类型号 × 4

例1. 60号中断的中断向量CS_{60}：IP_{60}存放在存储器的什么位置？(假如CS_{60}=2345H；IP_{60}=7890H)

地址=0000+60×4 = 240=0F0H



再如：硬盘　1NT　13H

向量地址=0000：13H×4 = 0000：004CH

004CH开始连续4个单元中用来存放“INT 13H”的中断向量

5.3.4.4 中断向量表的填写

用于系统未配置系统软件

例1. 假设中断类型号为60H，中断服务程序的段地址为SEG-$INTR_{60}$，偏移地址是OFFSET-$TNTR_{60}$，编写装入程序：

```
CLI                          ；关中断
CLD                          ；串操作时地址增量
MOV  AX，0
MOV  ES，AX                  ；中断向量表在0段
MOV  DI，4×60H               ；中断向量指针 ⟶ DI
MOV  AX，OFFSET-INTR；中断服务程序偏移值 ⟶AX
STOSW                        ；AX ⟶[DI]和[DI+1]中，DI=DI+2
MOV  AX，SEG-INTR            ；段地址 ⟶AX
STOSW                        ；AX⟶ [DI]和[DI+1]中，DI=DI+2
STI                          ；开中断
```

5.3.4.5 中断向量表的修改

用于有系统资源的情况

修改中断向量并非修改中断号，而是修改同一中断号下的中断服务程序入口地址。

在实模式下，用**DOS**功能调用**INT21H**的**35H**号功能和**25H**号功能

35H号功能
- 功能：从向量表中读取中断向量
- 入口参数：**AH=35H**，**AL=**中断号
- 出口参数：**ES**：**BX=**读取的中断向量段基址：偏移量

25H号功能
- 功能：向向量表中写入中断向量
- 入口参数：**AH=25H**，**AL=**中断号
  DS：**DX=**要写入的中断向量的段基址：偏移量
- 出口参数：无


IP60
CS60
90H
78H
45H
23H
0000:00F0H
0000:00F1H
0000:00F2H
0000:00F3H


IP60
CS60
12H
34H
00H
FFH
0000:00F0H
0000:00F1H
0000:00F2H
0000:00F3H

5.3.4.5 中断向量表的修改——步骤

用于有系统资源的情况

1) 取中断向量——用35H功能，获取原中断向量，并保存在字变量中（假设中断类型号为n）：

格式：MOV AH，35H

MOV AL，nH ；n为中断类型号

INT 21H

出口参数：BX放原中断程序的偏移地址

ES放原中断程序的段地址

即原中断向量取出放到ES:BX中保存

2) 设置新中断向量——用25H功能，设置新中断向量，取代原中断向量，以便当中断发生后转移到新中断服务程序中。

入口参数：**DX** 放新中断服务程序入口地址的偏移地址

DS 放新中断服务程序入口地址的段地址

```
MOV     AH，25H
MOV     AL，  nH      ；n为中断类型号
INT     21H
```

把**DS:DX**指向的中断向量放到中断向量表类型号为**n**的中断向量处

3) 恢复中断向量——新中断服务程序完毕后，用25H功能恢复原中断向量

格式：

入口参数：**DX** 放原中断服务程序入口地址的偏移地址

DS 放原中断服务程序入口地址的段地址

AH=25H，AL=中断号

```
MOV     DX,OLD-SEG
MOV     DS,DX                   ;原中断服务程序入口地址的段地址
MOV     DX,OLD-OFF              ;原中断服务程序入口地址的偏移地址
MOV     AH,25H
MOV     AL, nH                  ;  n为原中断类型号
INT     21H
```

例：假如原中断服务程序的中断号为N，新中断程序的入口地址的段基址为SEG_INTRnew，偏移地址为OFFSET_INTRnew，中断向量修改程序：

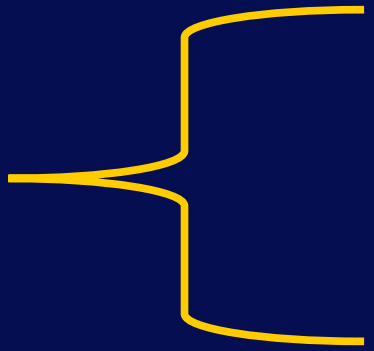
```
DATA        SEGMENT
        OLD_SEG    DW ?
        OLD_OFF    DW ?
                   …
DATA        ENDS
            ….
            MOV   AH, 35H
                  MOV   AL, N
                  INT     21H
                  MOV   OLD_SEG, ES
                  MOV   OLD_OFF, BX
                  MOV   AL, N
                  MOV   AH, 25H
```

```
MOV     DX, SEG_INTRnew
MOV      DS, DX
MOV      DX, OFFSET_INTRnew
INT      21H
....

MOV     AH, 25H
MOV     AL, N
MOV     DX, SEG OLD_SEG
MOV     DS, DX
MOV     DX, OFFSET OLD_OFF
INT     21H
```

中断服务程序

```
INTRnew  PROC FAR
.....
IRET
INTRnew  ENDP
```



5.5 中断描述符与中断描述符表

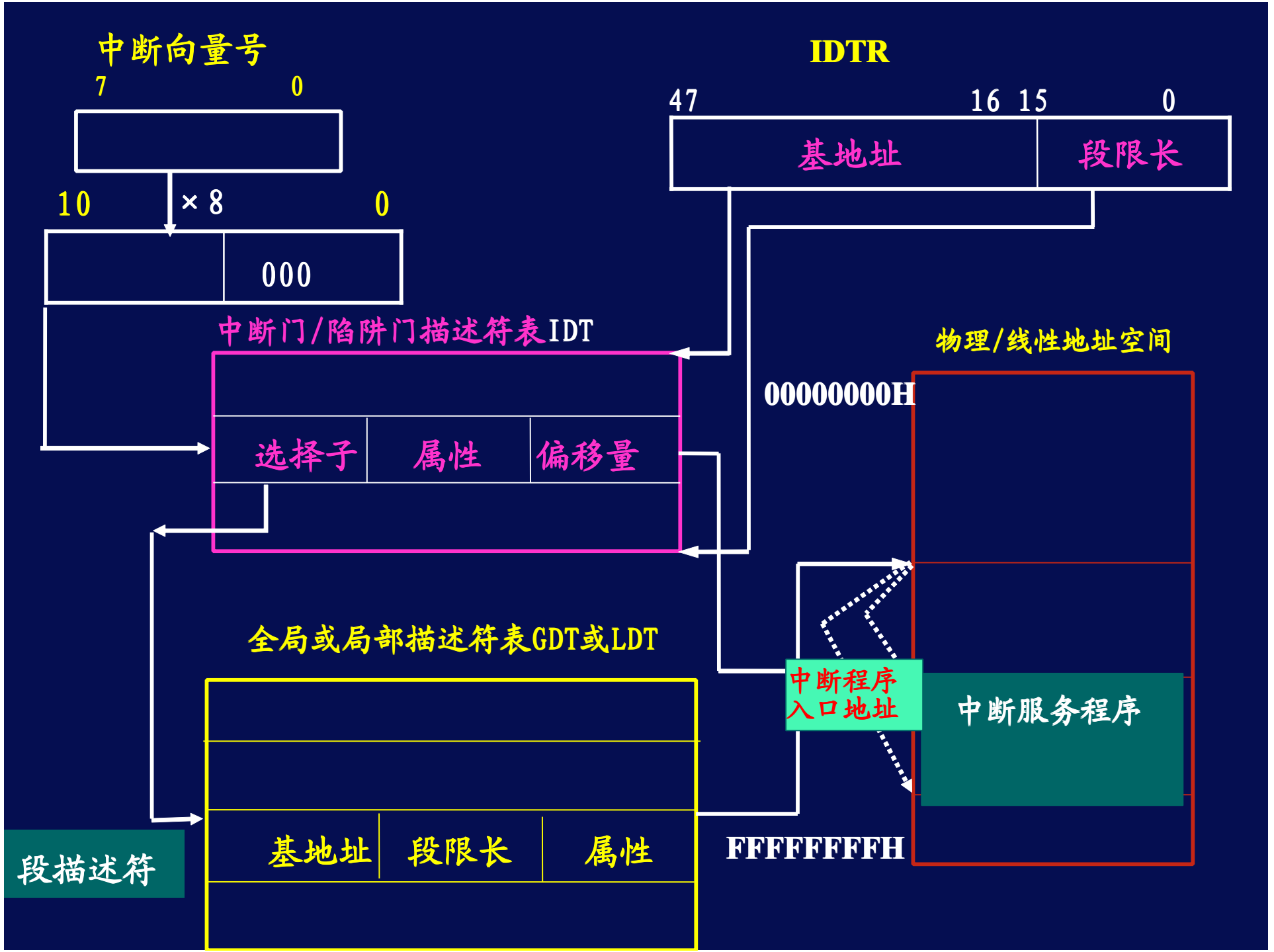
在保护模式下，由于存放中断服务程序的程序存储区地址是虚拟地址，用段描述符加偏移量表示。因此，中断服务程序的入口地址，由段描述符和偏移量两部分组成：

① 服务程序的段描述符（8个字节）

② 服务程序的偏移量（4个字节）

5.5.1~5.5.2 中断描述符、描述符表IDT

- 保护模式下采用中断描述符表IDT管理各级中断
- IDT中最多可以有256个描述符，对应于256个中断源
- IDT表中的描述符包括了中断服务程序的入口地址信息
- IDT可置于内存的任意区域，其起始地址由中断描述符表寄存器(IDTR)设置


中断向量号
7
0
10
× 8
0
000
IDTR
47
16 15
0
基地址
段限长
中断门/陷阱门描述符表IDT
选择子
属性
偏移量
物理/线性地址空间
00000000H
全局或局部描述符表GDT或LDT
基地址
段限长
属性
段描述符
中断程序
入口地址
中断服务程序
FFFFFFFFH

5.3.6 中断/异常处理

5.3.6.1 实模式下的中断/异常处理

可屏蔽中断、不可屏蔽中断、软件中断的处理

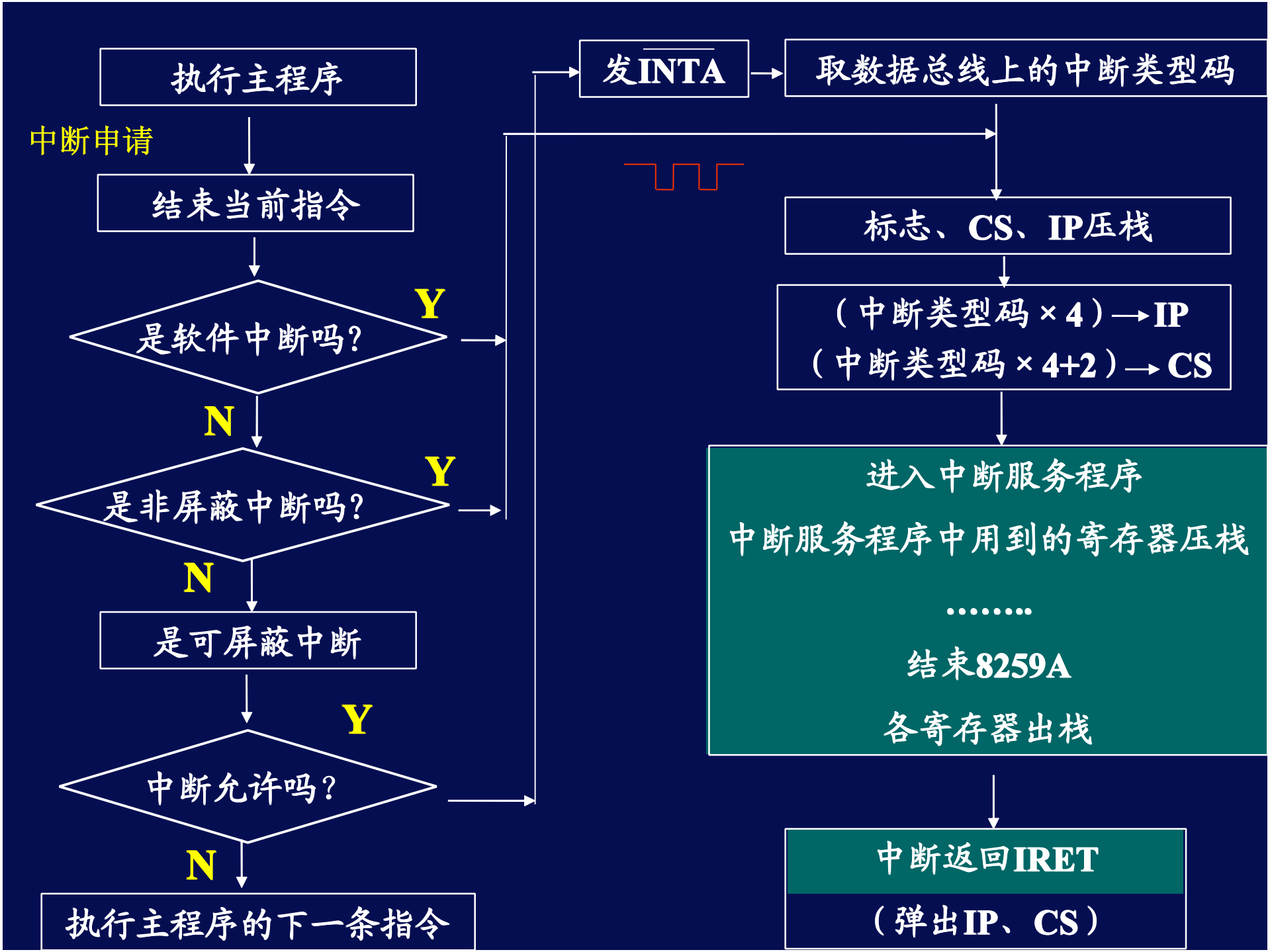
全过程包括以下4个阶段：

（1）中断申请与响应握手

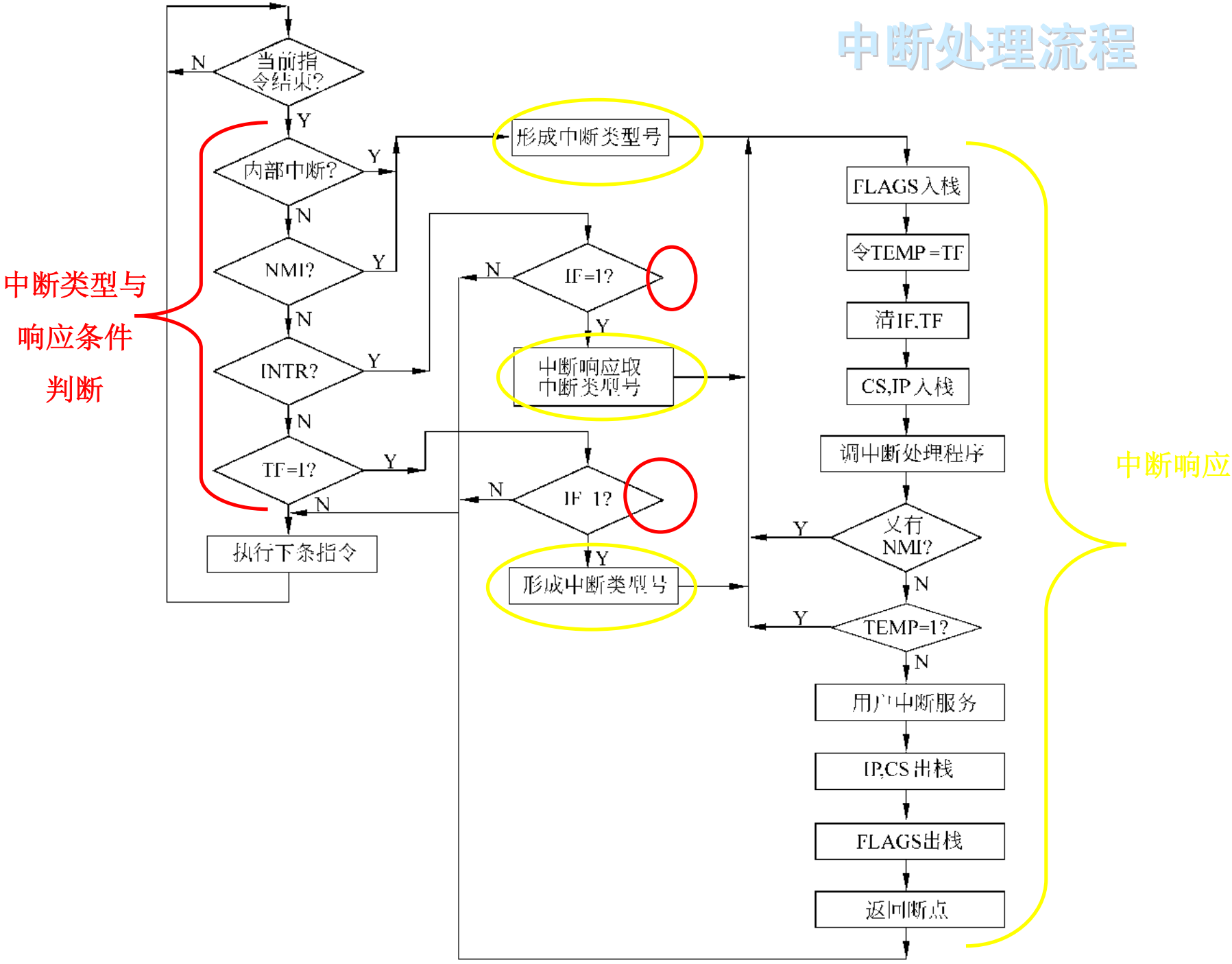
（2）标志位的处理与断点保存

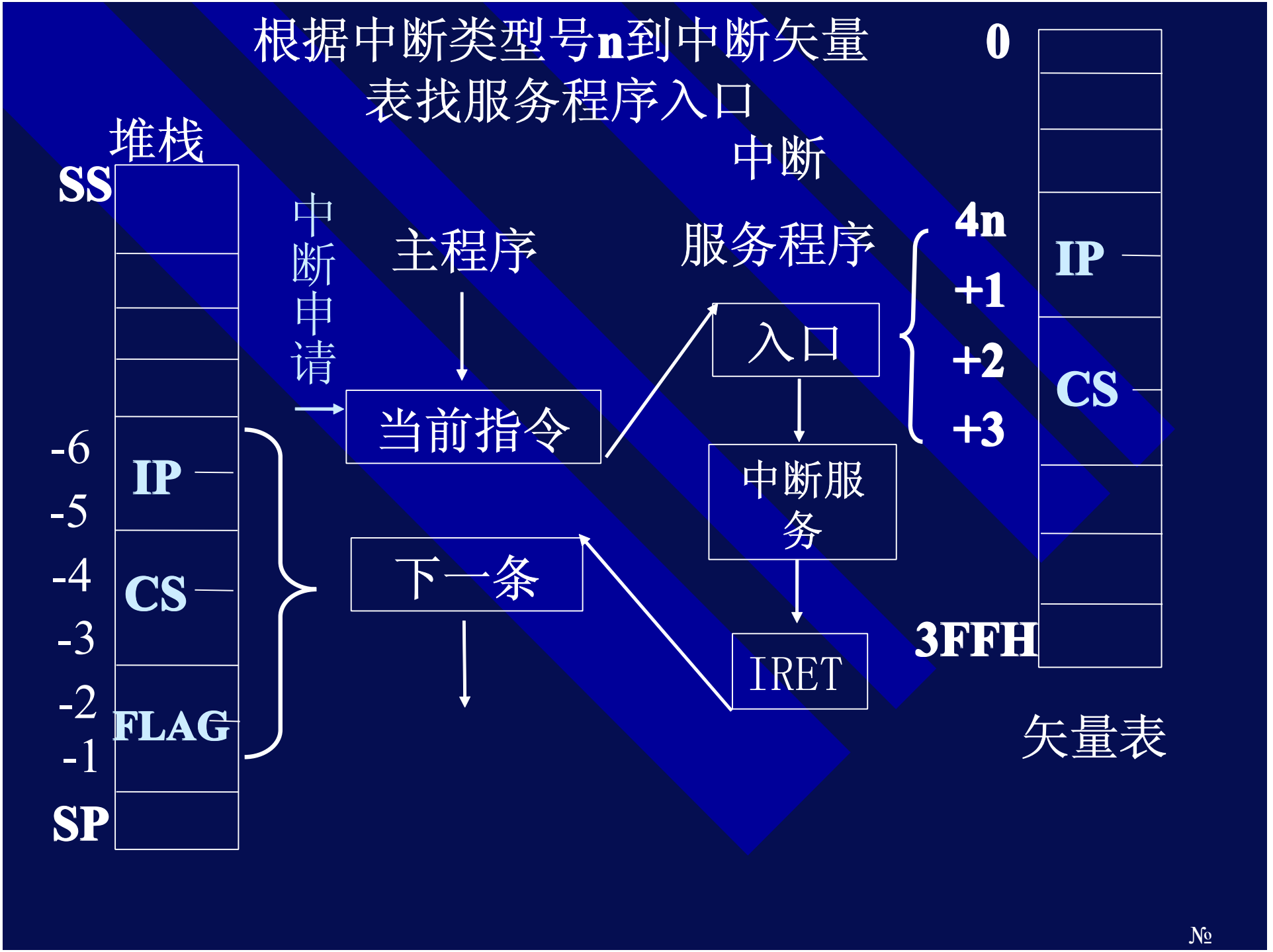
（3）向中断服务程序转移并执行中断服务程序

（4）返回断点


执行主程序
中断申请
结束当前指令
是软件中断吗?
Y
N
是非屏蔽中断吗?
Y
N
是可屏蔽中断
Y
中断允许吗?
N
执行主程序的下一条指令
发INTA
取数据总线上的中断类型码
标志、CS、IP压栈
(中断类型码×4)→IP
(中断类型码×4+2)→CS
进入中断服务程序
中断服务程序中用到的寄存器压栈
……
结束8259A
各寄存器出栈
中断返回IRET
(弹出IP、CS)

中断处理流程


当前指令结束?
N
Y
内部中断?
Y
N
NMI?
Y
N
INTR?
Y
N
TF=1?
Y
N
执行下条指令
形成中断类型号
IF=1?
N
Y
中断响应取中断类型号
IF=1?
N
Y
形成中断类型号
FLAGS入栈
令TEMP=TF
清IF,TF
CS,IP入栈
调中断处理程序
义有NMI?
Y
N
TEMP=1?
Y
N
用户中断服务
IP,CS出栈
FLAGS出栈
返回断点
中断类型与响应条件判断
中断响应


根据中断类型号n到中断矢量表找服务程序入口
堆栈
SS
SP
-6
-5
-4
-3
-2
-1
IP
CS
FLAG
中断申请
主程序
当前指令
下一条
中断服务程序
入口
中断服务
IRET
0
4n
+1
+2
+3
IP
CS
3FFH
矢量表
№

可屏蔽中断的响应周期

当：1. CPU收到INT中断请求

2. 前一条指令执行完，且中断标志位IF=1

进入中断响应周期；完成两个工作：

1）第一个INTA脉冲时，CPU产生LOCK信号，使总线处于封锁状态，防止DMA占用总线。

2）第二个INTA时，LOCK撤除，总线解封。

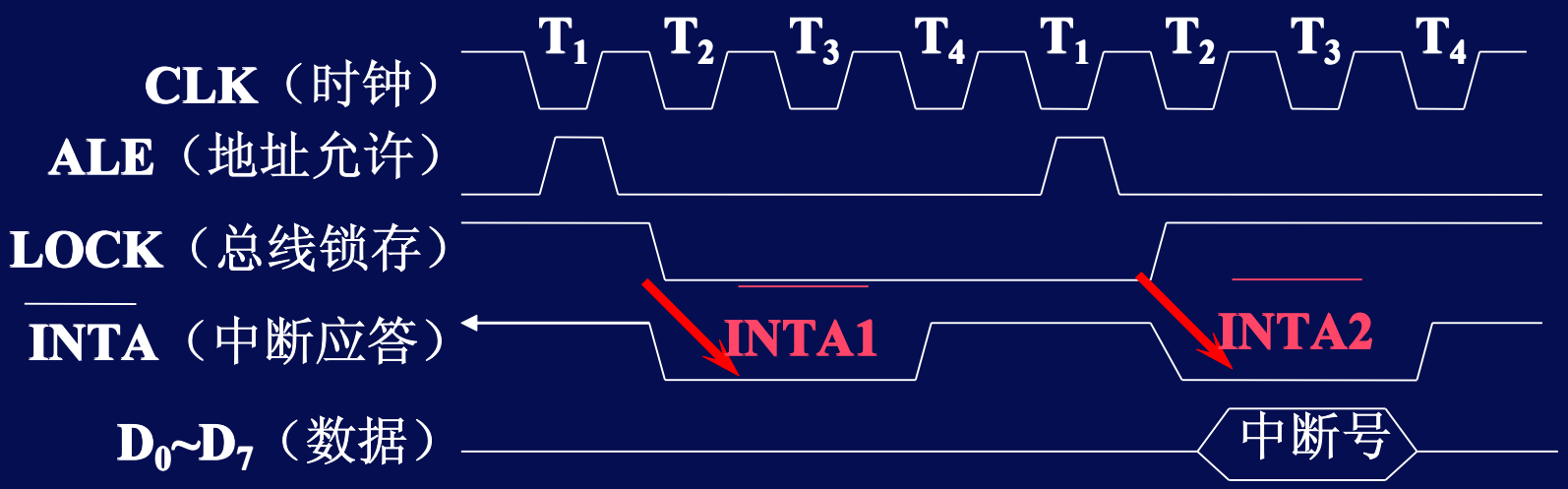


图8.2 可屏蔽中断响应总线周期

4. 异常的处理（不可屏蔽）

●异常是由于指令执行结果有错而使指令不能执行而产生的故障，其故障号由系统安排。

●因此一旦发生异常，就自动按所分配的故障号通过中断向量表进入异常处理程序，而不需要从外部获取中断号。

●异常具有软件中断的特征，不产生中断响应总线周期。异常处理程序不需发中断结束命令**EOI**。

5.3.6.2 对实模式与保护模式下中断处理过程的分析

两种模式下对中断/异常的处理过程存在很大的差别，根本原因是微处理器的两种模式下对存储器的管理方式不同和是否引入保护机制所致。

- 实模式下使用中断向量和中断向
- 保护模式下使用中断门描述符和

保护模式下使用描述符来描述虚拟地址空间，而实模式下使用段来表示实际地址空间的缘故。

- ➢保护模式下必须通过门(中断门、陷阱门、任务门)描述符来实现向服务程序转移，
- ➢实模式直接转移到服务程序

保护模式下引入了保护机制，而实模式下没有保护机制。

◆保护模式下可以转移到一个以独立任务方式出现的中断/异常服务程序，而实模式下不可以

实模式下不支持多任务，只有保护模式下才有多任务的功能

◆保护模式下，执行中断程序控制转移和返回断点的过程中，都要进行一系列的特权级与条件保护性检测，而实模式对此不予考虑。

5.3.7 可编程中断控制器8259A

5.3.7.1 82C59A外部特性和内部寄存器

可编程中断控制器**PIC(Programable Interrupt Controller)82C59A**作为中断系统的核心器件，协助**CPU**管理外部中断，是一个十分重要的芯片。

主要功能：

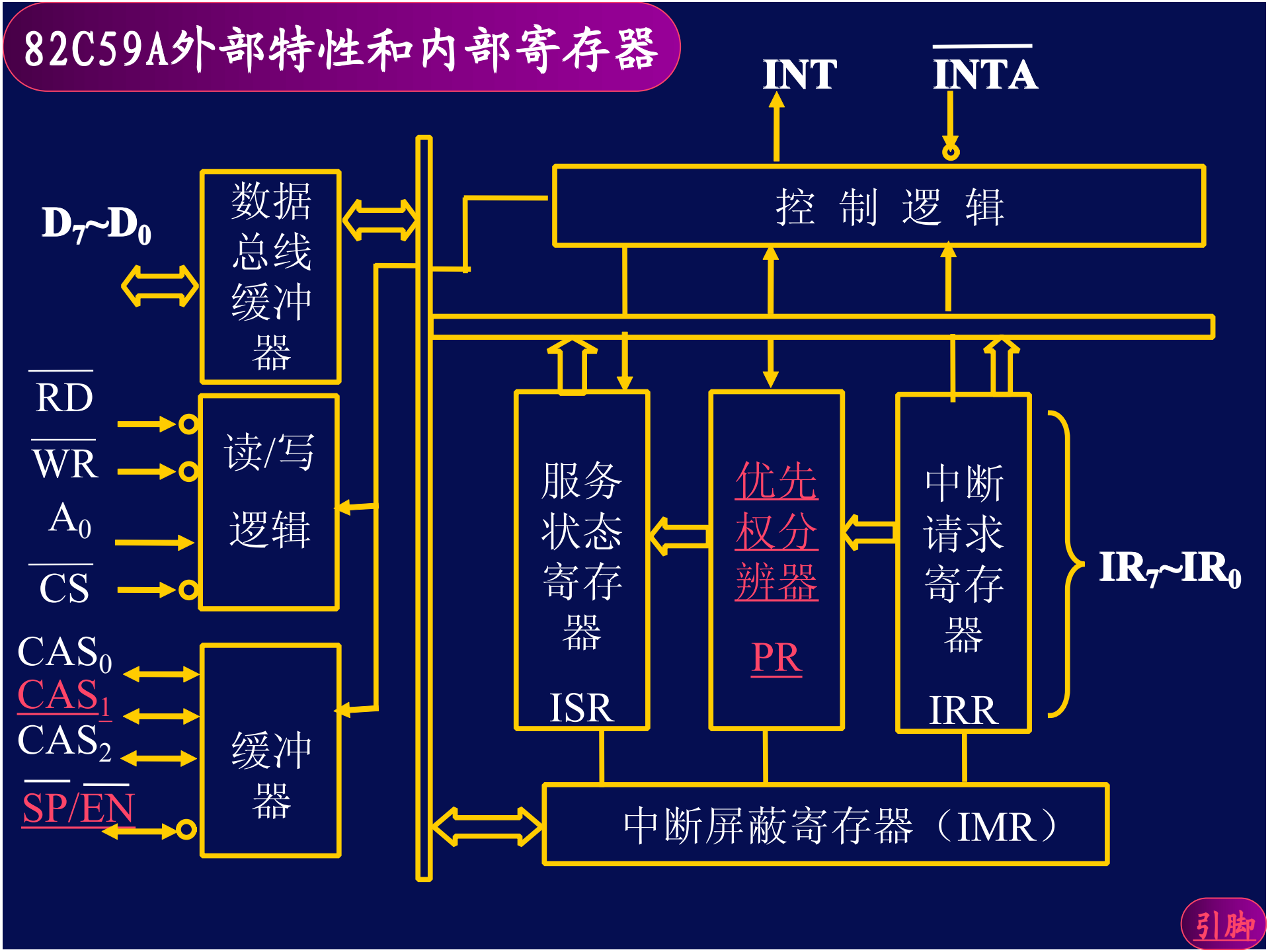
- 针对多个中断请求，对其进行屏蔽、优先级等管理
- 向CPU转达中断请求，并视CPU的响应送出中断类型号

5.3.7 可编程中断控制器8259A

要点：

- 中断请求是如何识别的？
- 片内是如何寻址的？
- 类型号是如何确定及何时向CPU送出的？
- 中断请求的处理是如何完结的？

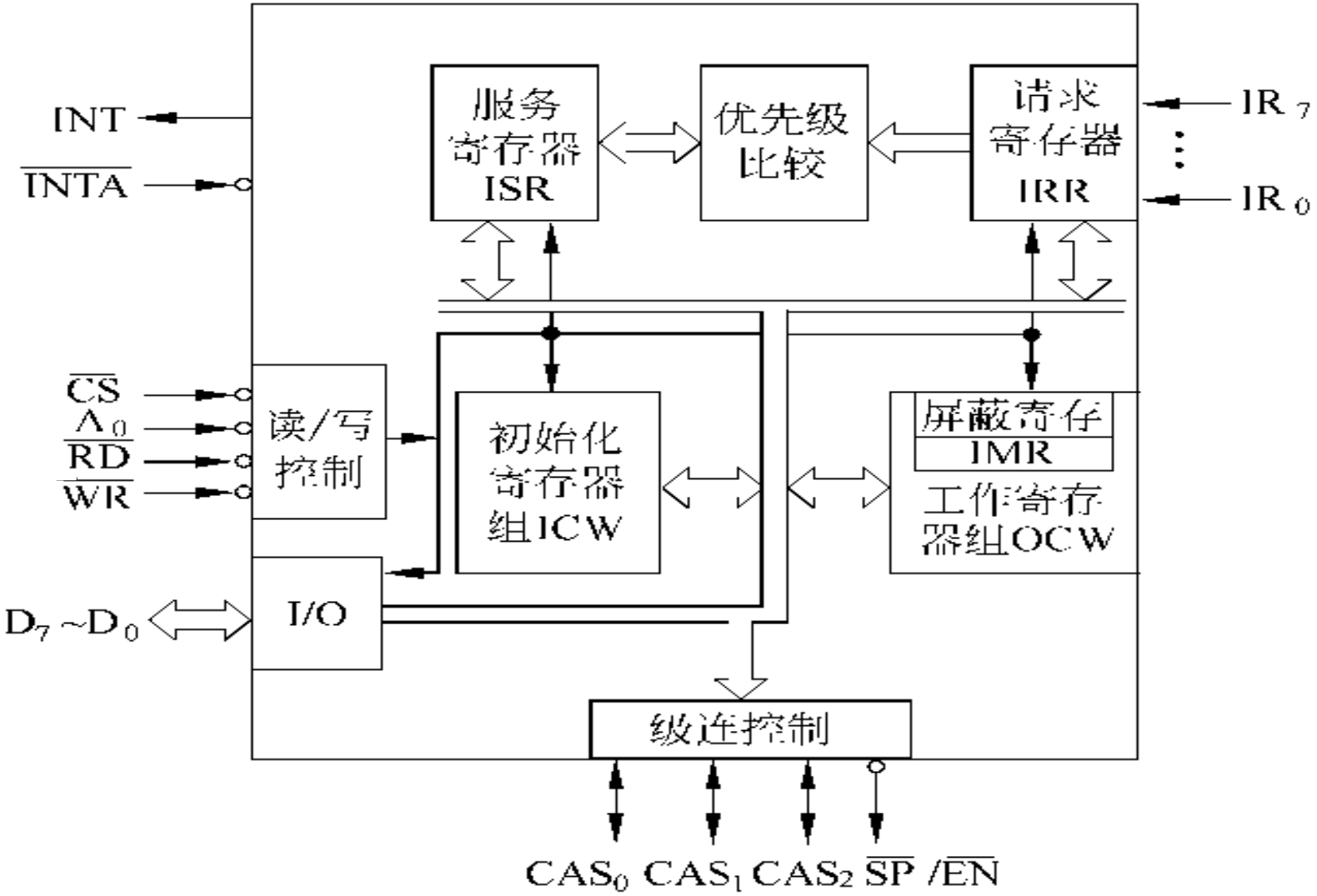
82C59A外部特性和内部寄存器

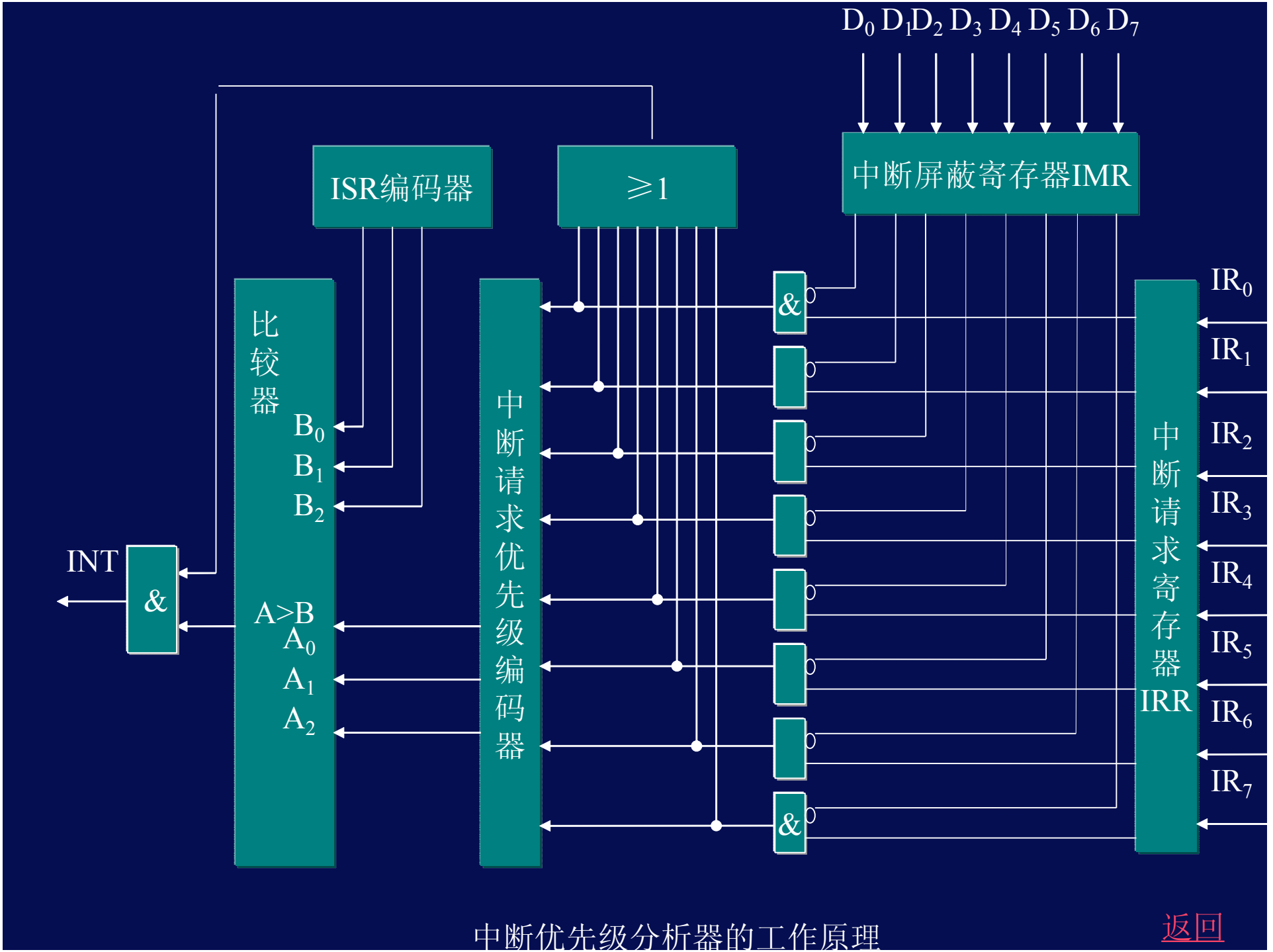


引脚

每片可管理8个外部中断源

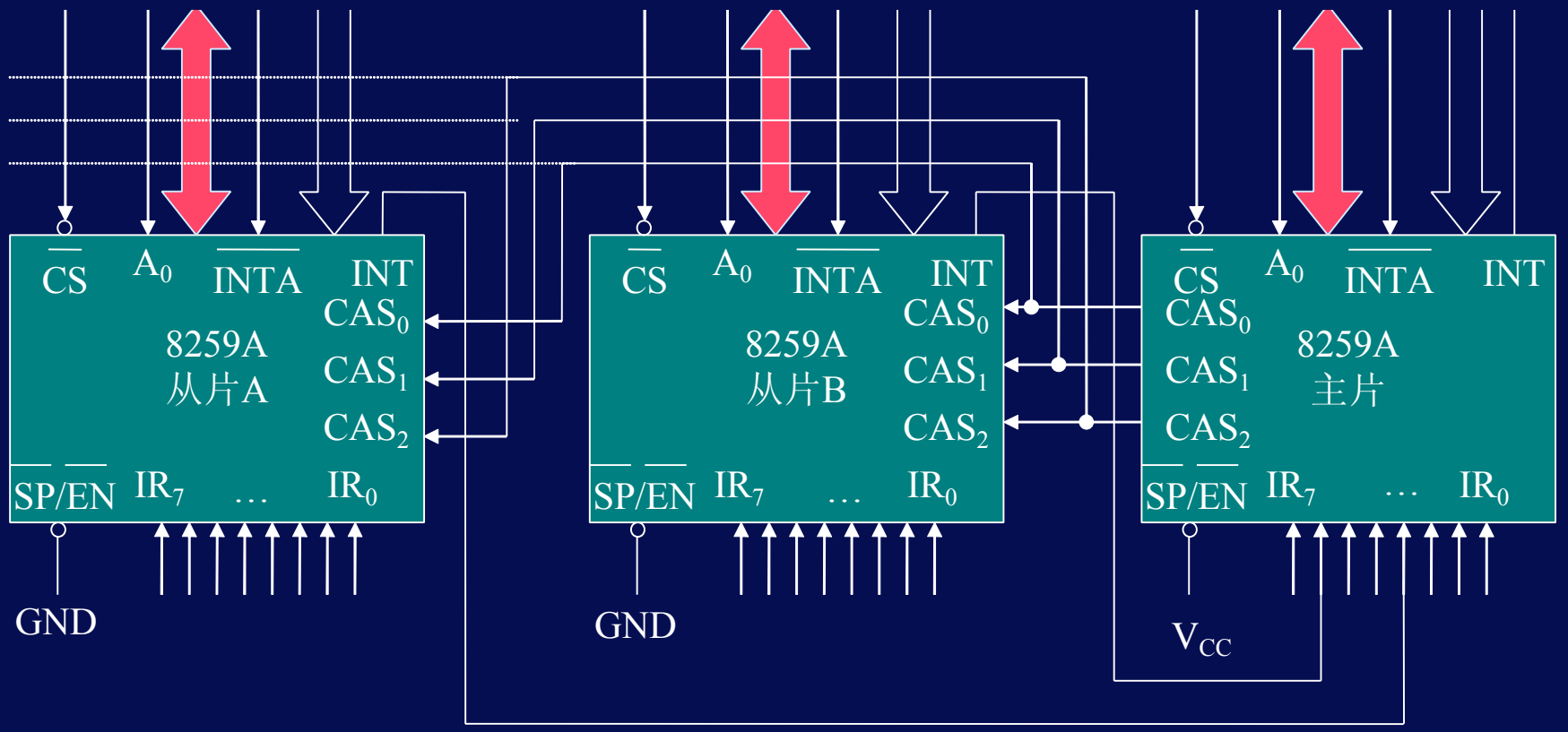
8529A的构成



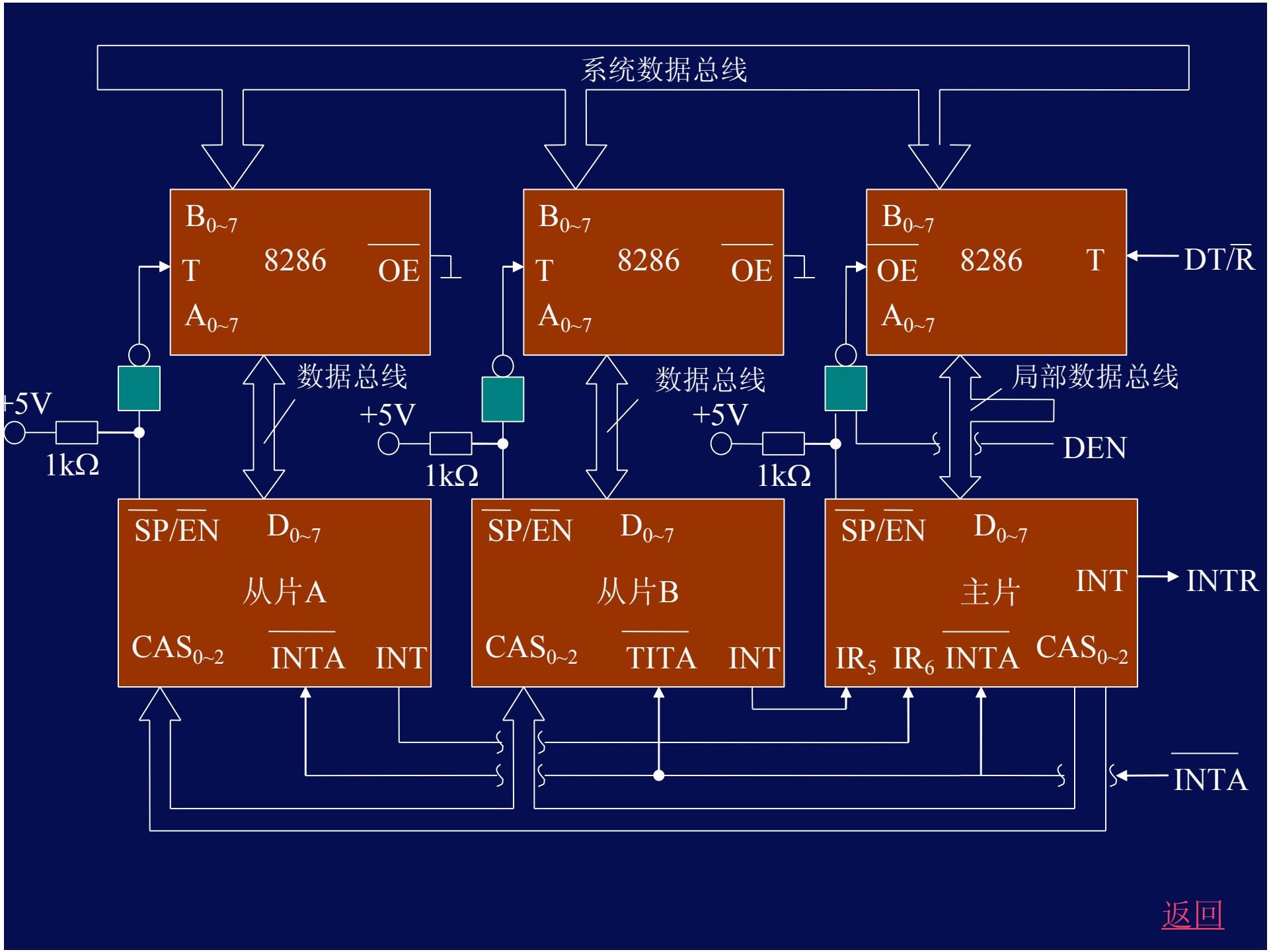


中断优先级分析器的工作原理

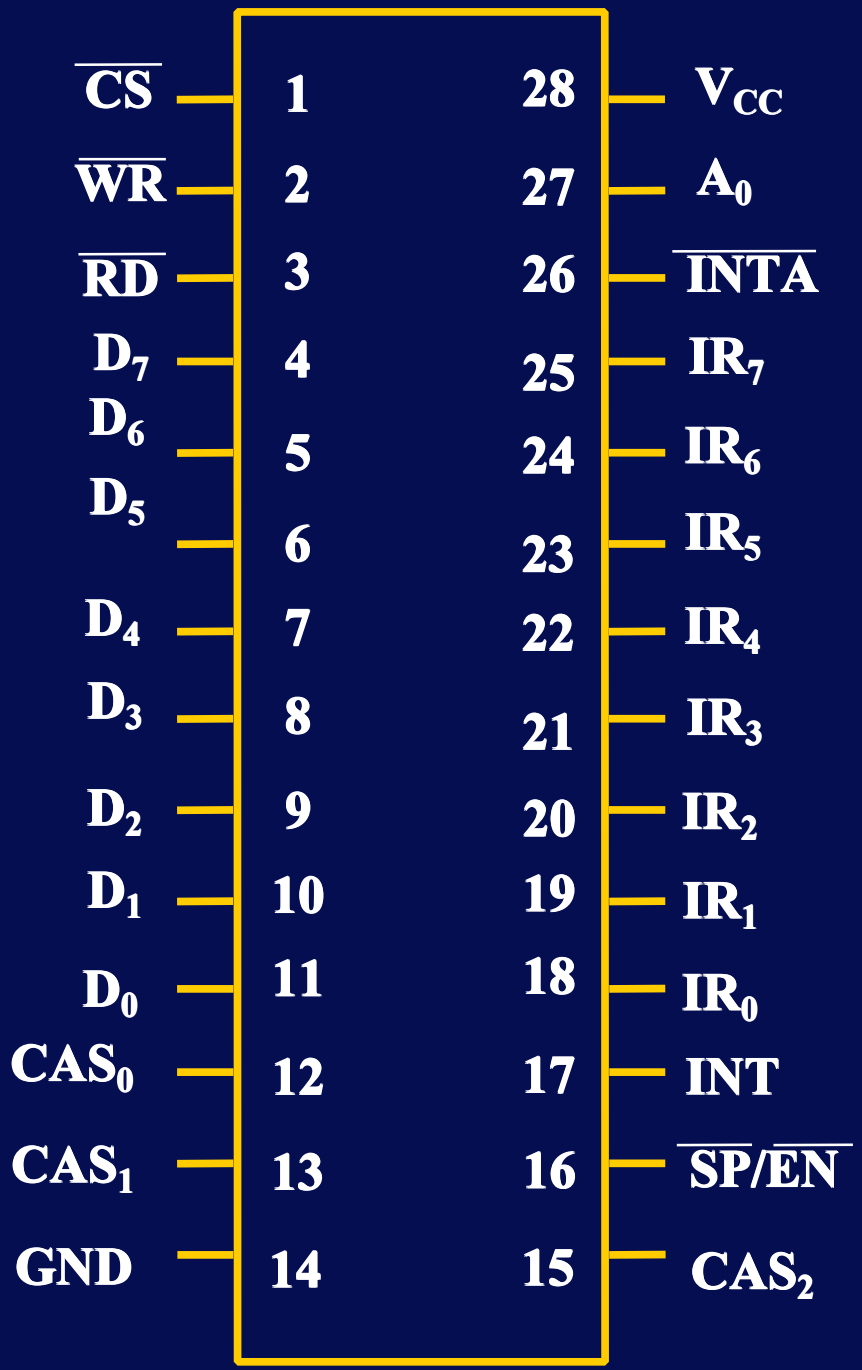
返回


CS
A0
INTA
INT
CAS0
CAS1
CAS2
8259A
从片A
SP/EN
IR7
…
IR0
GND
8259A
从片B
8259A
主片
VCC

返回


系统数据总线
B0~7
T
A0~7
8286
OE
B0~7
T
A0~7
8286
OE
B0~7
OE
A0~7
8286
T
DT/R
数据总线
数据总线
局部数据总线
+5V
1kΩ
+5V
1kΩ
+5V
1kΩ
DEN
SP/EN
D0~7
从片A
CAS0~2
INTA
INT
SP/EN
D0~7
从片B
CAS0~2
TITA
INT
SP/EN
D0~7
主片
INT
INTR
IR5
IR6
INTA
CAS0~2
INTA
返回

82C59A的引脚


$\overline{CS}$ 1
$\overline{WR}$ 2
$\overline{RD}$ 3
D_7 4
D_6 5
D_5 6
D_4 7
D_3 8
D_2 9
D_1 10
D_0 11
CAS_0 12
CAS_1 13
GND 14
28 V_{CC}
27 A_0
26 $\overline{INTA}$
25 IR_7
24 IR_6
23 IR_5
22 IR_4
21 IR_3
20 IR_2
19 IR_1
18 IR_0
17 INT
16 $\overline{SP}/\overline{EN}$
15 CAS_2

D_7—D_0：数据总线，双向，三态。用于与CPU之间传送命令、状态、中断类型码。

$\overline{RD}$ ：读信号，输入。用来通知8259把某个内部寄存器的值送数据线D_7~D_0 。

$\overline{WR}$：写信号，输入。用来通知8259把数据线D_7~D_0上的值写入内部某个寄存器。

$\overline{CS}$：片选信号，输入。通过地址译码逻辑电路与地址总线相连。

A_0：地址线，输入。用来指出当前8259的哪个 端口被访问，选择内部寄存器的端口地址。

在标准AT机中，使用两片8259构成主从式中断系统:

主8259的端口地址： 20H，21H

从8259的端口地址： A0H，A1H

INT：中断请求，输出。把IR_7～IR_0上的最高优先级请求传送到CPU的INTR引脚，向CPU发中断请求。

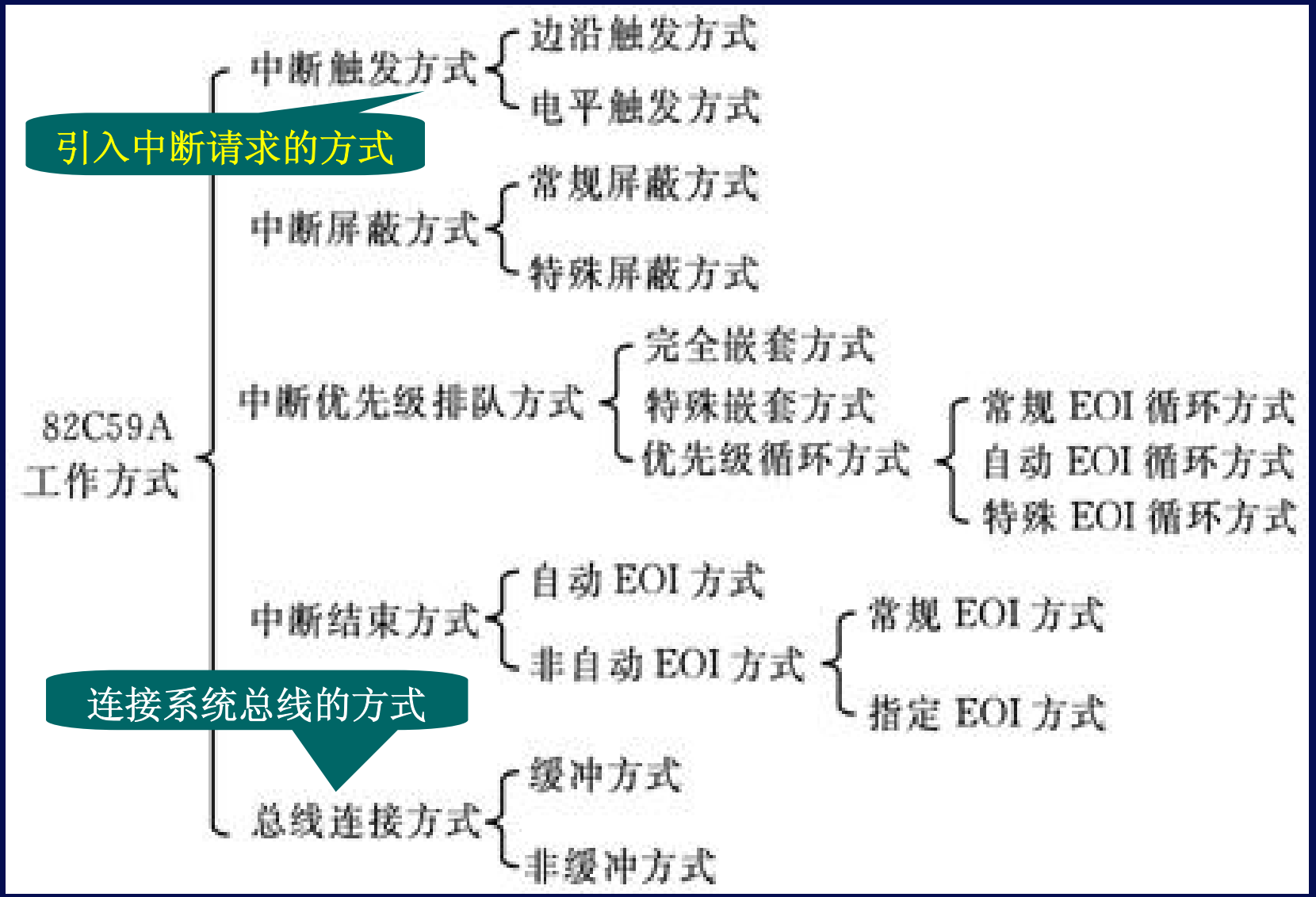
$\overline{\text{INTA}}$：中断响应，接收CPU的中断应答信号。CPU发出的中断响应信号为两个负脉冲。第一个负脉冲作为中断应答信号，第二个负脉冲到来时，8259从数据线D_7~D_0上发出中断类型码。

$IR_7 \sim IR_0$：外设中断请求输入。在含有多片8259的复杂系统中，主片的IR_7~IR_0分别与从片的INT端相连，用来接收来自从片的中断请求。

$CAS_2 \sim CAS_0$：级联线，用来指出具体从片。

$\overline{SP}/\overline{EN}$：从设备编程/缓冲器允许，双向，输入时用来决定8259是主片还是从片（1-主片；0-从片，非缓冲方式）；输出时，使数据总线驱动器启动（缓冲方式）。

5.3.7.2 82C59A的工作方式



1. 引入中断请求的方式

（1）边沿触发方式

（2）电平触发方式

（3）中断查询方式

（1）边沿触发方式

正跳沿接入**IRi**，向**8259A**请求中断。上跳沿后可一直维持高电平，不会产生中断。

（2）电平触发方式

高电平申请中断。但在响应中断后必须及时清除高电平，以防止引起第二次误中断。

（3）中断查询方式

用软件确定中断请求位的方式。

特点：a. 外设仍通过**8259A**的**IRi**申请中断，但**8259A**却不使用**INT**信号向**CPU**申请中断。

b. **CPU**内部关中断（**IF=0**）

c. **CPU**用软件查询确定中断源

用**OUT**指令向**8259A**的偶地址发一个查询命令字**OCW3**。

再用**IN**指令从**8259A**的偶地址读这个查询字，以确定中断源。

2. 屏蔽中断源的方式

- 常规屏蔽方式
- 特殊屏蔽方式

常规屏蔽方式：用**OCW1**使屏蔽寄存器**IMR**中的一位或几位置**1**来屏蔽一个或几个中断源的中断请求。

特殊屏蔽方式：用**OCW3**的D_6D_5=11 屏蔽自己

3. 中断优先级排队方式

- 完全嵌套方式
- 特殊全嵌套方式
- 优先级循环方式
  - 一般是自动和特殊两种
  - 自动**EIO**循环
  - 特殊**EIO**循环

（1）完全嵌套方式：

特点：

a. 优先级别**IR0**最高，**IR7**最低，且级别固定不变。

b. 只允许响应高级中断。

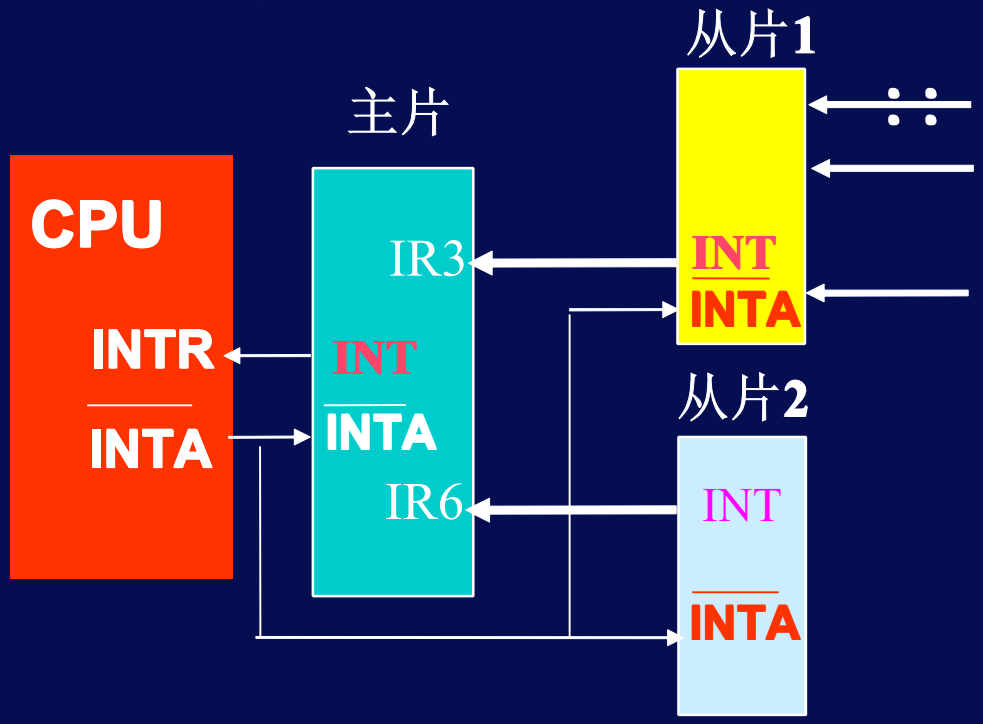
c. 中断嵌套的深度取决于整个中断系统所具有的中断级数。

（2）特殊全嵌套方式：

具有与完全嵌套方式基本相同的功能。

另有：可以响应同级的中断请求。

例：



（3） 优先级循环方式

a.自动循环

当任何一级中断被处理完，优先级别变为最低。

初始优先级：低 IR_7 IR_6 IR_5 IR_4 IR_3 IR_2 IR_1 IR_0 高

处理完IR_1：低 IR_1 IR_0 IR_7 IR_6 IR_5 IR_4 IR_3 IR_2 高

b. 优先级特殊循环方式

用一条优先权置位指令（写**OCW2**）把最低优先级赋给某一中断源（**IRi)**，于是最高优先级便为IR_{i+1}，其它循环。

初始优先级: 低 $IR_7\ IR_6\ IR_5\ IR_4\ IR_3\ IR_2\ IR_1\ IR_0$ 高

正在服务时发一条命令让IR_4为最低: 低 $IR_4\ IR_3\ IR_2\ IR_1\ IR_0\ IR_7\ IR_6\ IR_5$ 高

5. 中断结束方式（**EOI**）

- 自动中断结束方式
- 非自动中断结束方式
  - 常规中断结束方式
  - 指定中断结束方式

自动中断结束方式：用$\overline{\mathbf{INTA}}$第二个负脉冲。

常规中断结束方式：不指定要结束的中断是哪一个

非自动中断结束方式：（**OCW2=00100xxx)**

指定中断结束方式：指定要结束的中断是哪一个

（**OCW2=00100$L_2L_1L_0$)**

一般情况下：

全嵌套方式用普通和自动中断结束方式

特殊全嵌套方式用特殊中断结束方式

优先级自动循环方式用普通中断结束方式

优先级特殊循环方式用特殊中断结束方式

5. 连接系统总线的方式

- 数据缓冲方式
- 非缓冲方式

5.3.7.3. 82C59A的编程命令

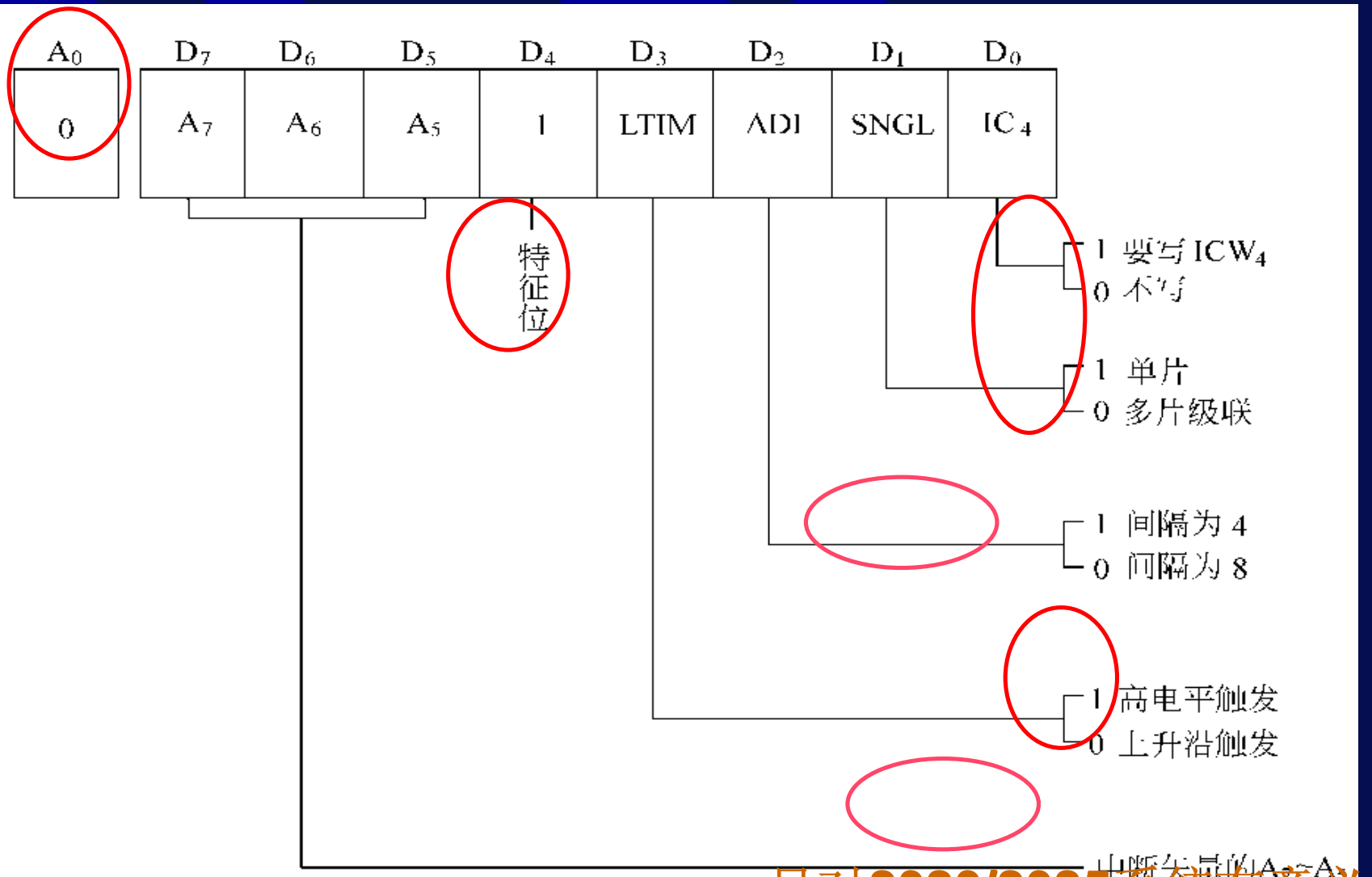
- 编程命令有两种
  - 设置工作方式：初始化命令字 ICW_1~ICW_4
  - 控制操作：操作命令字：OCW_1~OCW_3
- 每片8259A有2个片内地址
  - A_0=0 偶地址端口
  - A_0=1 奇地址端口

所有的命令都是通过这两个端口按一定的规则写入8259A。

1. 8259的初始化命令字：
（预置命令字ICW1—ICW4）

- **ICW_1~ICW_4**在初始化程序中设定，且在整个工作过程中保持不变。
- **ICW1~ICW4**必须按顺序设定。
- **ICW1**写入**8259**偶地址中。
- **ICW2~ICW4**写入**8259**奇地址中。

ICW1格式



只对**8080/8085**系统有意义

ICW_1 设置工作方式

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	X	X	X	1	LTIM	X	SGNL	IC_4

对A_0=0的端口写入一个D_4=1的数据，表示初始化编程开始。

- D_3——LTIM：中断信号的触发方式
  - 0：边沿
  - 1：高电平
- D_1——SGNL：是否单片方式
  - 0：多片级联
  - 1：单片
- D_0——IC_4：是否有ICW_4
  - 0：无
  - 1：有

ICW_2 设置中断类型码

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	T7	T6	T5	T4	T3	0	0	0

- 在写ICW_1之后，对A_0=1的端口第一次写入的数据是ICW_2。
- 设置D_7~D_3

  D_2~D_0为000（由8259A根据IR_0~IR_7自动填充为000~111）

ICW_3——设置级联：

只有当系统中有多片**8259A**级联时（ICW_1中**SNGL=0**），才需在ICW_2之后写ICW_3，且主片和从片的格式不同。

对于主片：置**1**的位表示对应的引脚**IR**有从片级联。

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0

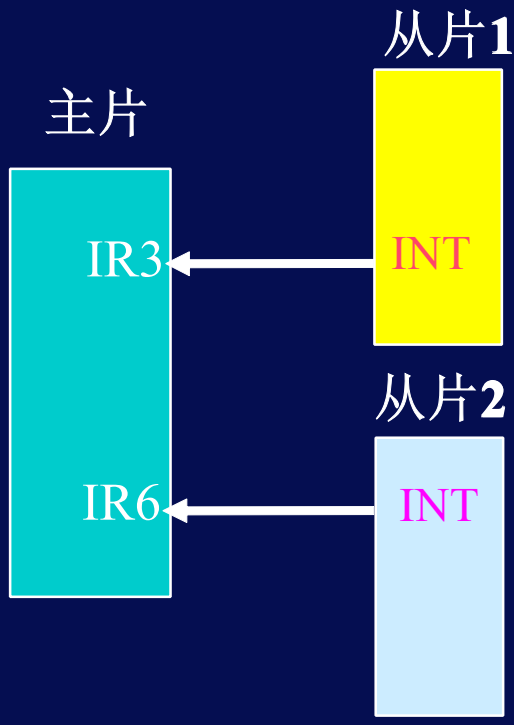
IRi=
- **0**：表示**IRi** 引脚上未接**8259A**从片
- **1**：表示**IRi** 引脚上接有**8259A**从片

■对于从片：D_2~D_0表示该从片接在主片的哪个IR引脚上

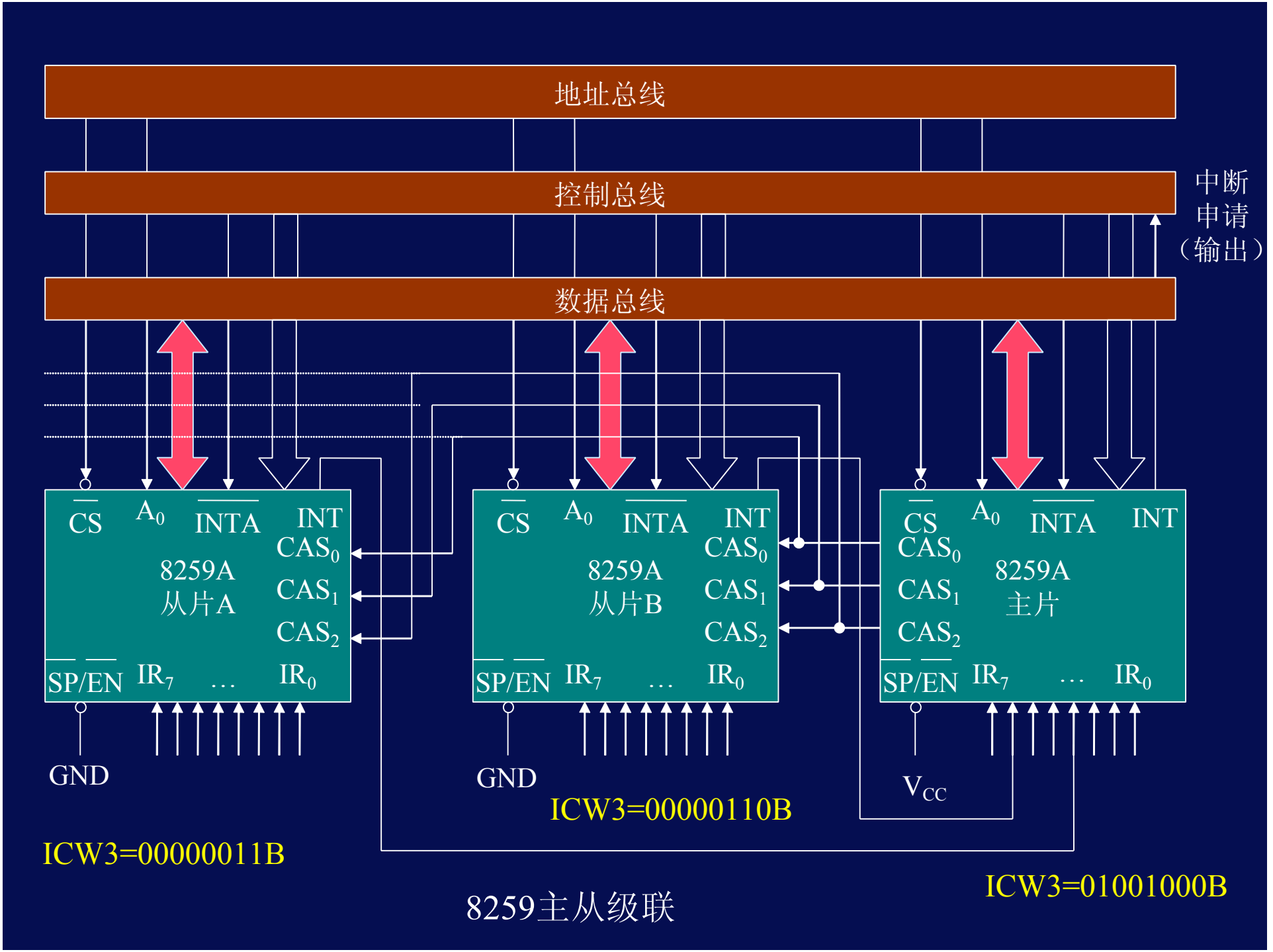
A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	X	X	X	X	X	ID2	ID1	ID0

IR0	0	0	0
IR1：	0	0	1
…	…	…	…
IR7：	1	1	1

例：



ICW3：
- 主 片：01001000B
- 从片1：00000011B
- 从片2：00000110B


地址总线
控制总线
数据总线
中断
申请
（输出）
CS
A0
INTA
INT
CAS0
CAS1
CAS2
8259A
从片A
SP/EN
IR7
…
IR0
GND
ICW3=00000011B
8259A
从片B
ICW3=00000110B
8259A
主片
VCC
ICW3=01001000B

8259主从级联

ICW_4 设置模式

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	SFNM	BUF	M/$\overline{S}$	AEOI	μ PM

当ICW_1中的IC_4=1时，有ICW_4。

- **D_4——SFNM** 中断的嵌套方式
  - **0**：一般嵌套
  - **1**：特殊全嵌套

- **D_1——AEOI** 自动结束中断方式
  - **0**：不自动清除ISR
  - **1**：CPU响应中断后，自动清除ISR

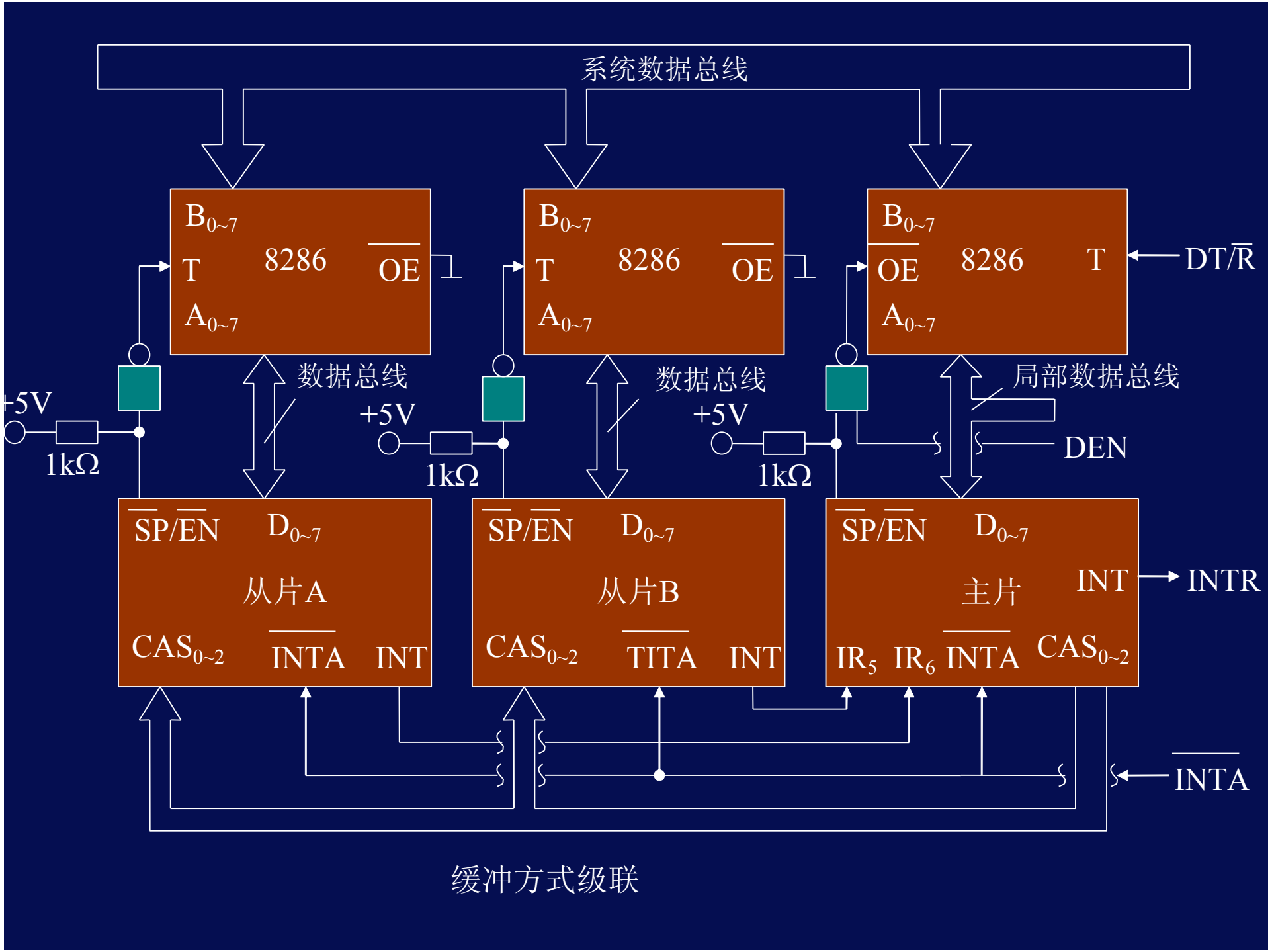
- **D_0——μPM** 微处理器类型
  - **0**：8080/8085/Z80
  - **1**：8086/8088

- **D_3：BUF 缓冲**

  1：8259通过数据缓冲器和总线相连，$\overline{\text{SP}}/\overline{\text{EN}}$引脚输出， 缓冲器选通端。

  0：无缓冲，$\overline{\text{SP}}/\overline{\text{EN}}$引脚输入，用作主片、从片选择端。

- **D_2：M/$\overline{\text{S}}$** 主片/从片选择（**BUF=1**时，有效）{ **0：**从片 **1：**主片


系统数据总线
B0~7
T
A0~7
8286
OE
DT/R
数据总线
局部数据总线
+5V
1kΩ
DEN
SP/EN
D0~7
从片A
从片B
主片
INT
INTR
CAS0~2
INTA
TITA
IR5
IR6
缓冲方式级联

2．8259的操作命令字OCW

- 系统初始化完成以后，可以在应用程序中进行操作编程。
- **8259A有3条操作命令字：OCW1，OCW2，OCW3**

1）OCW_1 ——中断屏蔽操作命令字

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0

Mi=1 屏蔽中断源 **IRi**

Mi=0 允许**IRi** 端请求中断

例：
```
MOV  AL, 0F7H      ;开放IR3中断
OUT  21H, AL
```

2）OCW_2——设置优先级轮换方式和中断结束方式

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	R	SL	EOI	0	0	L_2	L_1	L_0

对A_0=0端口写入D_4D_3=00的数据，表示是OCW_2

R=1 表示循环；

SL＝ 1 表示需要由L_2~L_1指定：
- 指定优先级轮换
- 指定中断结束位

EOI=1 表示需要发中断结束命令位。

在PC机中常用的EOI命令是：

```
MOV AL, 20H
OUT 20H, AL
```

3）OCW_3

- 设置和撤销特殊屏蔽方式
- 设置中断查询方式
- 设置对**8259A**内部寄存器的读出。

A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	ESMM	SMM	0	1	P	RR	RIS

ESMM SMM
- **0X**：无效
- **10**：特殊屏蔽方式复位
- **11**：特殊屏蔽方式置位

P
- **1**：查询**8259A**中断状态
- **0**：不查询

RR RIS
- **0X**：无效
- **10**：下次读有效，读**IRR**
- **11**：下次读有效，读**ISR**

查询字格式：

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
IR	X	X	X	X	W_2	W_1	W_0

1表示有中断请求　　当前中断请求的最高优先级

读 **IRR**：先向偶端口写**0AH**，再读偶端口

读 **ISR**：先向偶端口写**0BH**，再读偶端口

读优先级最高的中断请求 **IR**（查询**8259A**中断状态）：

先向偶端口写**0CH**，再读偶端口

读 **IMR**：初始化后随时可向奇端口读

5.3.7.4 82C59A在微机系统中对中断管理的作用

82C59A与微处理器组成微机的中断系统，它协助CPU实现一些中断事务的管理功能。

- 接收和记录各级中断源的中断请求。
- 优先级排对管理：判优，确定是否响应和响应哪一级的中断请求。
- 当CPU响应中断时，为CPU提供中断类型码。
- 屏蔽和开放中断请求:一片Intel 82C59可管理8个中断请求,允许9片级联，构成64级中断系统。
- 执行中断结束命令: 可屏蔽中断的中断服务程序，在中断返回之前，要求发中断结束命令。

5.3.7.5 82C59A在32位微机中的应用

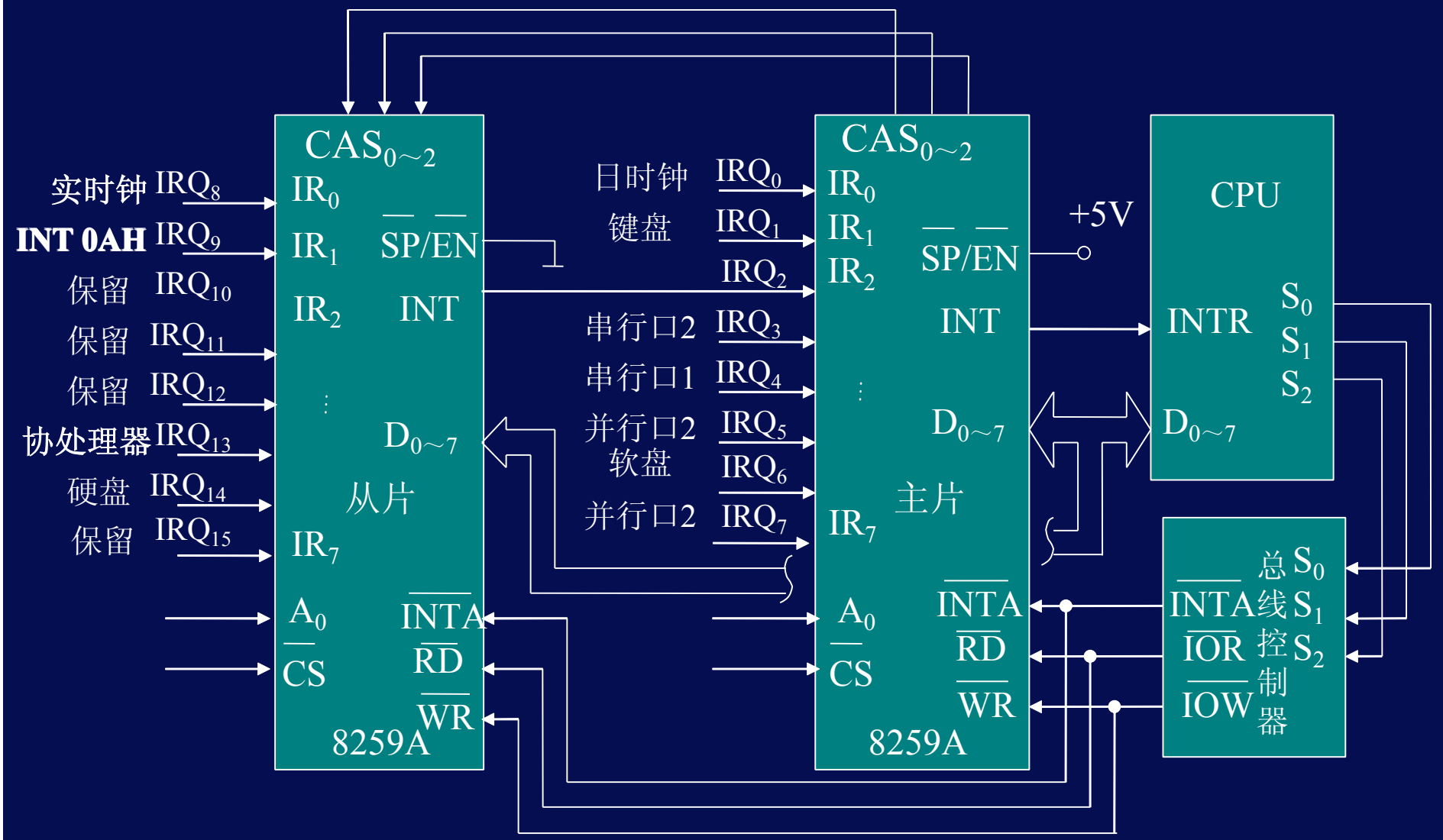
32位微机的微处理器已不使用单个的82C59A作为中断处理的支持芯片，而采用芯片组。

如在815EP芯片组的82801BA模块中，集成了两个82C59A可编程中断控制器的功能，作为ISA兼容中断提供可屏蔽中断服务。

1.82C59A的级联 (在32位微机中，实模式下82C59A的应用)

82801BA模块中有两片82C59A，进行级联可支持15级可屏蔽中断处理.

5.3.7.5 82C59A在32位微机中的应用



(与16位微机的中断系统在逻辑功能上兼容)

2. 82C59A的初始化

① 中断触发方式采用边沿触发，上跳变有效。

② 中断屏蔽方式采用常规屏蔽方式，即使用**OCW1**向**IMR**写屏蔽码。

③ 中断优先级排队方式采用固定优先级的完全嵌套方式

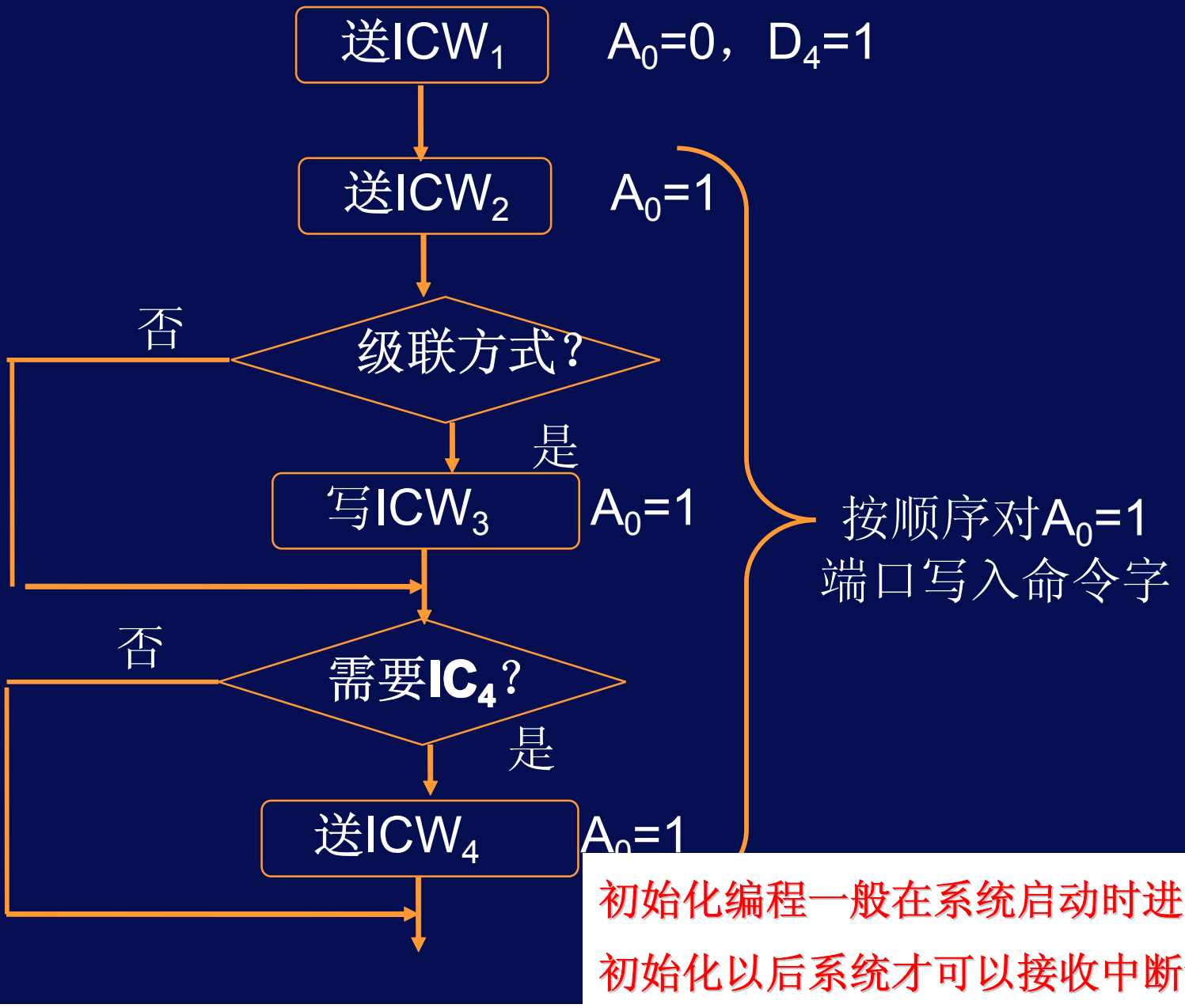
④ 中断结束方式采用非自动结束方式的两种命令格式，即在中断服务程序服务完毕，中断返回之前，发结束命令代码**20H**或**6XH**均可（**X**为**0**～**7**）。

⑤ 级联方式采用两片主/从连接方式，并且，规定把从片的中断申请输出引脚**INT**连到主片的中断请求输入引脚**IR2**上。两片级联处理**15**级中断。

⑥ **15**级中断号的分配为:中断号**08H**～**0FH**对应**IRQ0**～**IRQ7**；中断号**70H**～**77H**对应**IRQ8**～**IRQ15**。

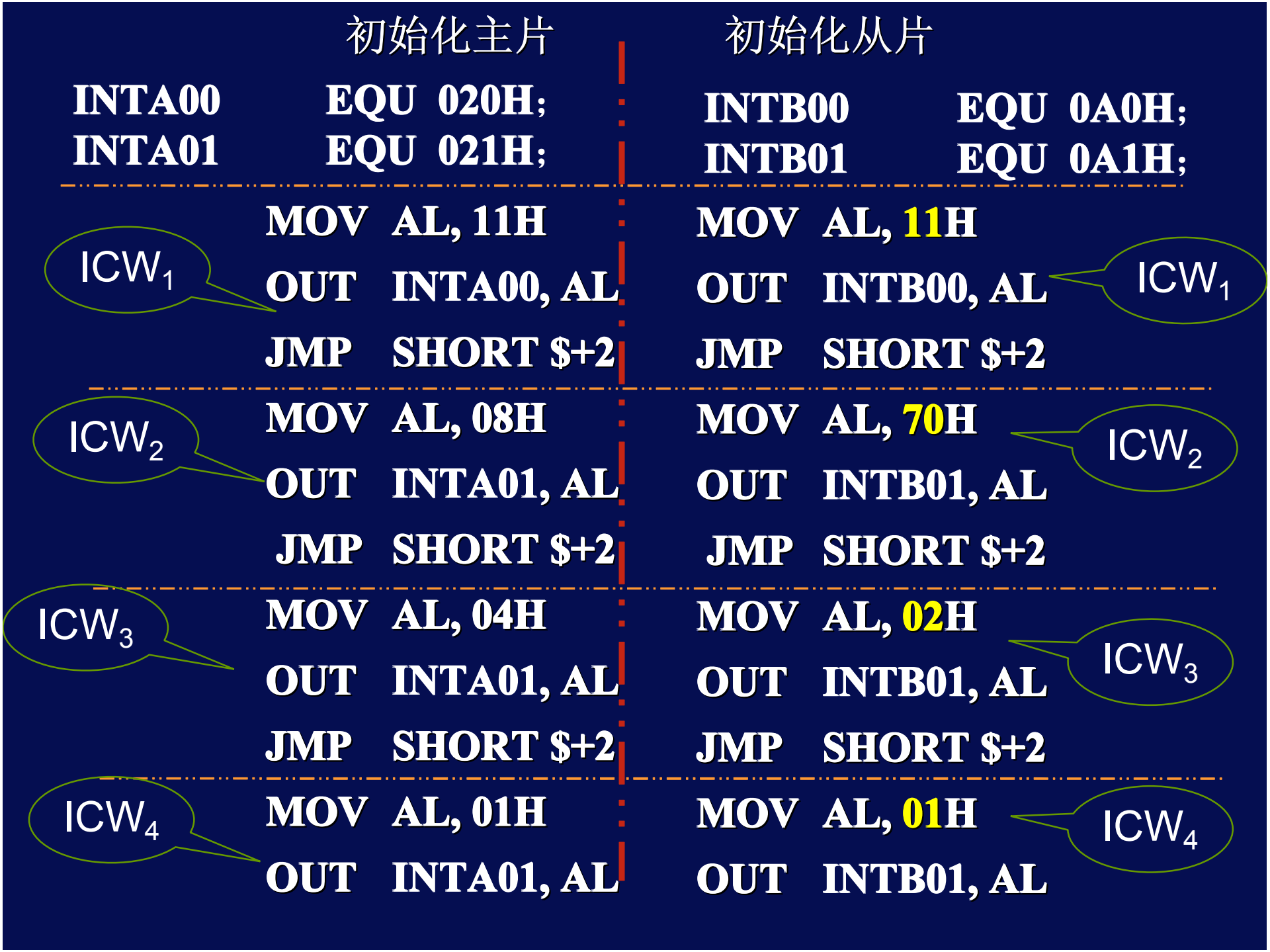
⑦ 两片**82C59A**的端口地址分配为：主片**82C59A**的两个端口是**20H**（**A0=0**）和**21H**（**A0=1**）；从片**82C59A**的两个端口是**0A0H**和**0A1H**。

82C59A芯片的初始化流程



初始化编程一般在系统启动时进行，

初始化以后系统才可以接收中断请求信号


初始化主片
初始化从片
INTA00 EQU 020H;
INTA01 EQU 021H;
INTB00 EQU 0A0H;
INTB01 EQU 0A1H;
MOV AL, 11H
OUT INTA00, AL
JMP SHORT $+2
ICW1
MOV AL, 11H
OUT INTB00, AL
JMP SHORT $+2
ICW1
MOV AL, 08H
OUT INTA01, AL
JMP SHORT $+2
ICW2
MOV AL, 70H
OUT INTB01, AL
JMP SHORT $+2
ICW2
MOV AL, 04H
OUT INTA01, AL
JMP SHORT $+2
ICW3
MOV AL, 02H
OUT INTB01, AL
JMP SHORT $+2
ICW3
MOV AL, 01H
OUT INTA01, AL
ICW4
MOV AL, 01H
OUT INTB01, AL
ICW4

作业：将书中没有答案的习题写在作业纸上