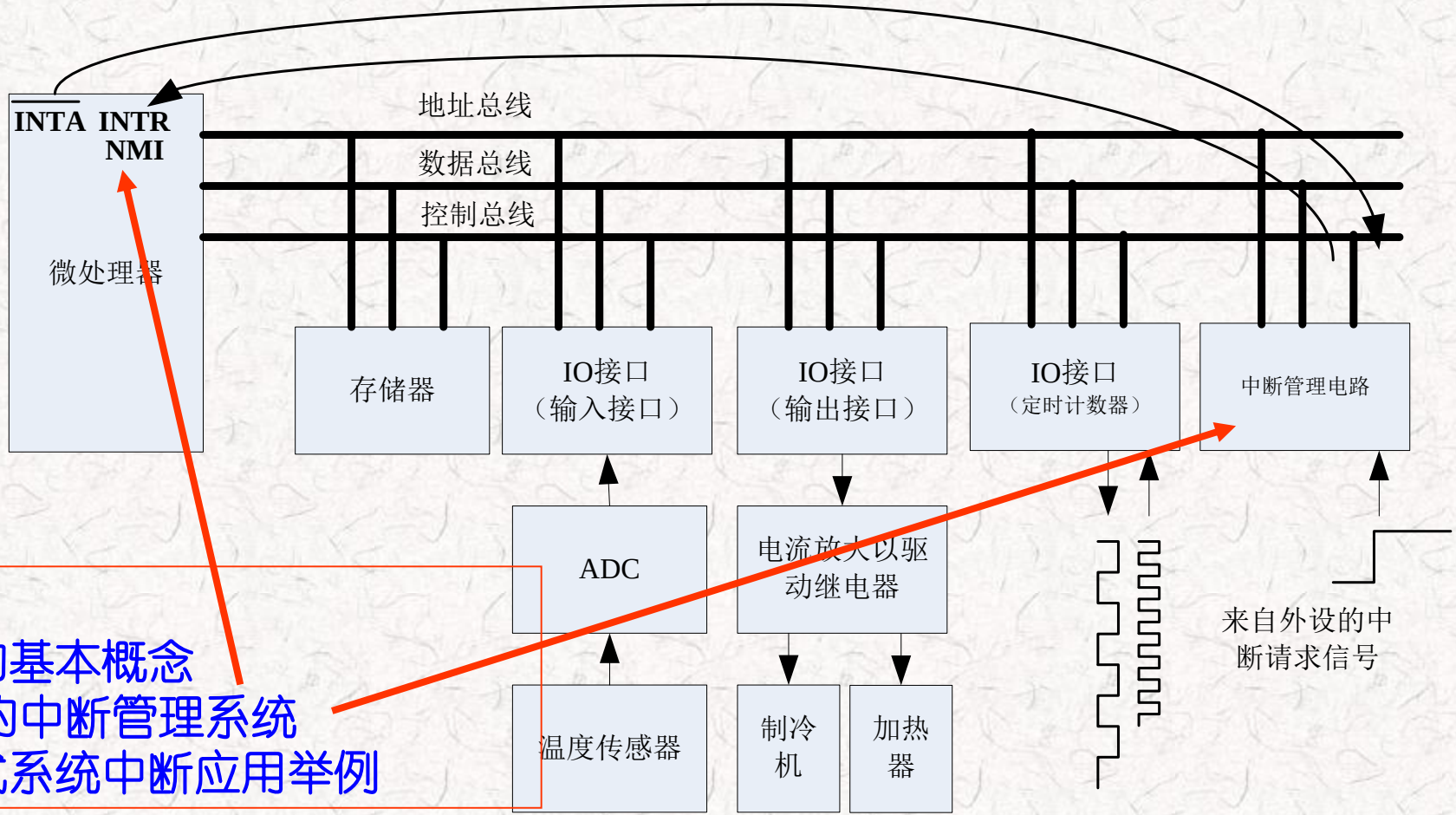
微机原理与接口技术

第八章 中断系统

课程内容之中断管理



第8章
中断的基本概念
8086的中断管理系统
嵌入式系统中断应用举例

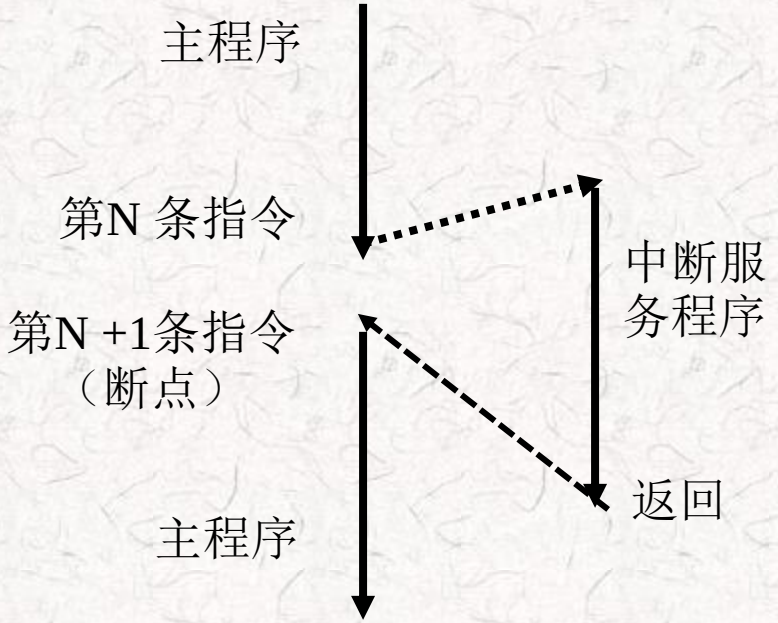
1.中断的基本概念

- 8086的中断源有哪几类？
- 外中断的种类？被8086响应的条件是什么？
- 有中断请求，中断使能，没有更高级别中断
- IF位起什么作用？程序中如何对IF位进行设置和清除？
- 8086内中断由哪几种情况引发？这些中断请求可以被屏蔽么？
- 8086中断类型号20H的中断向量存放在哪里？各字节表示什么含义？
- 中断优先级的含义是什么？同时发生两个中断，会响应哪个中断请求？在执行低优先级中断时，发生高优先级中断，会产生什么操作？

学习课本 第八章8.1.1节

什么是中断

- 正常情况CPU顺序执行指令，根据指令进行跳转、程序调用和返回，中断通常为突发操作，受某个信号或某个状态的触发，离开正常流程，跳转处理突发事件，再回到之前的位置继续向下执行



中断用在哪里

- 实现实时处理
  - 计算机用于控制、通信时，外设要求的服务是随机的，CPU可立即响应，避免经常查询
- 故障处理
  - 计算机运行中会出现各种故障，如，硬件错误、掉电、存储出错、运算溢出等异常情况，CPU可利用中断系统进行处理

中断的相关概念

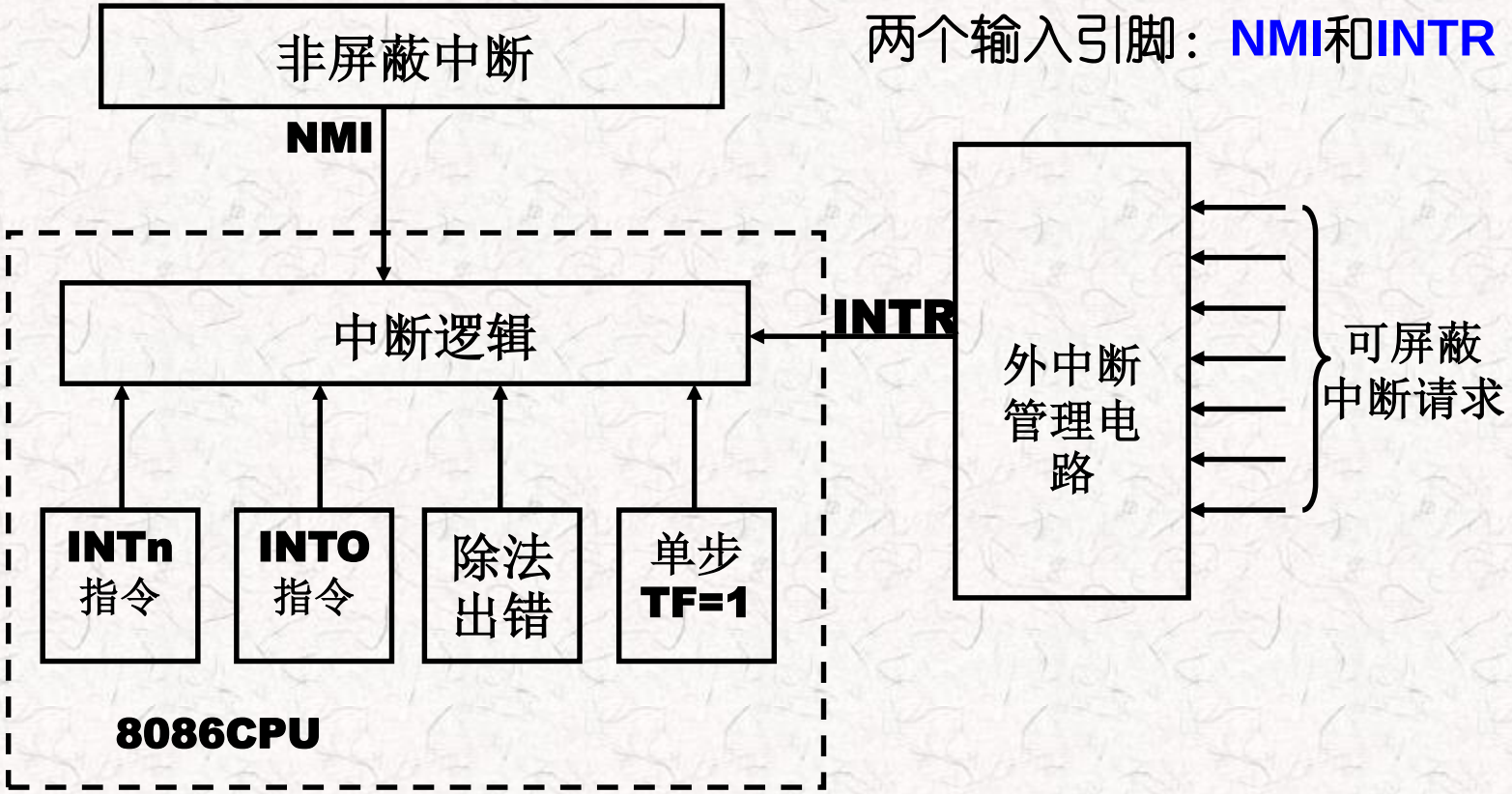
- 中断源 --- 产生中断请求的事件，包括内部事件和外部事件
- 外中断触发方式 --- 电平触发和跳变沿触发
- 中断屏蔽 --- 允许或禁止响应某一类中断
- 中断优先级 --- 多个中断同时发生，先响应谁？
- 中断挂号 --- 记录还没来得及响应的中断请求
- 中断向量 --- 中断服务程序入口地址
- 中断响应 --- 执行中断服务程序，同时清除中断挂号
- 中断嵌套 --- 中断服务程序里再响应中断
- 中断过程：
- 产生中断请求---是否允许CPU响应中断---CPU获取中断向量---进入中断服务程序---保护现场---执行相应操作---恢复现场---中断返回（现场保护与恢复包括硬件自动与程序员控制两部分）

8086中断系统的核心：如何进入中断服务程序？

- 中断系统需要解决的重要问题：发生中断请求时怎么进入相应的中断服务程序？
- 最多支持256个中断源
- 每个中断源对应一个类型号，共0~255
- 通过中断类型号到中断向量表中找到对应的中断向量，从而跳转到相应的中断服务程序执行

8086的中断源

两个输入引脚：NMI和INTR



8086中断源

内中断类型及特点1

- 除法出错（类型0）、单步中断（类型1）、断点中断（类型3）、INTO（类型4）、INT n（类型n）
- 除法错： 结果溢出自动产生
- 单步中断：每条指令后执行，受TF使能，TF=1中断
- 断点中断：调试用，插入 INT 3
- INTO： 通常安排在带符号数算术运算之后，如果OF=1则执行，否则无操作，继续向下执行

假设从80H端口读取的数据依次为
10H，10H，70H

```
      MOV CL, 3
      MOV BL,0
NEXT: IN AL, 80H
      ADD BL, AL
      INTO
      DEC CL
      JNE NEXT
```

```
MOV AL,10H
MOV BL,20H
MUL BL
INTO
MOV RESULT, AL
```

内中断类型及特点2

- **INT n：　n为类型号，0~255，例如DOS功能调用 INT 21H**

- 内中断特点
- 类型号已知　中断类型号包含在指令内或隐含规定
- 不可屏蔽　除单步中断外，任何内中断不能被禁止
- 不可嵌套　内中断服务程序不能再处理内中断

外中断的特点1

- NMI、INTR
- NMI：上升沿触发，内部锁存其请求状态
- NMI：当前指令执行完立刻响应，类型2
- NMI不可嵌套
- 注意：NMI和内中断的中断类型号均已知

外中断的特点2

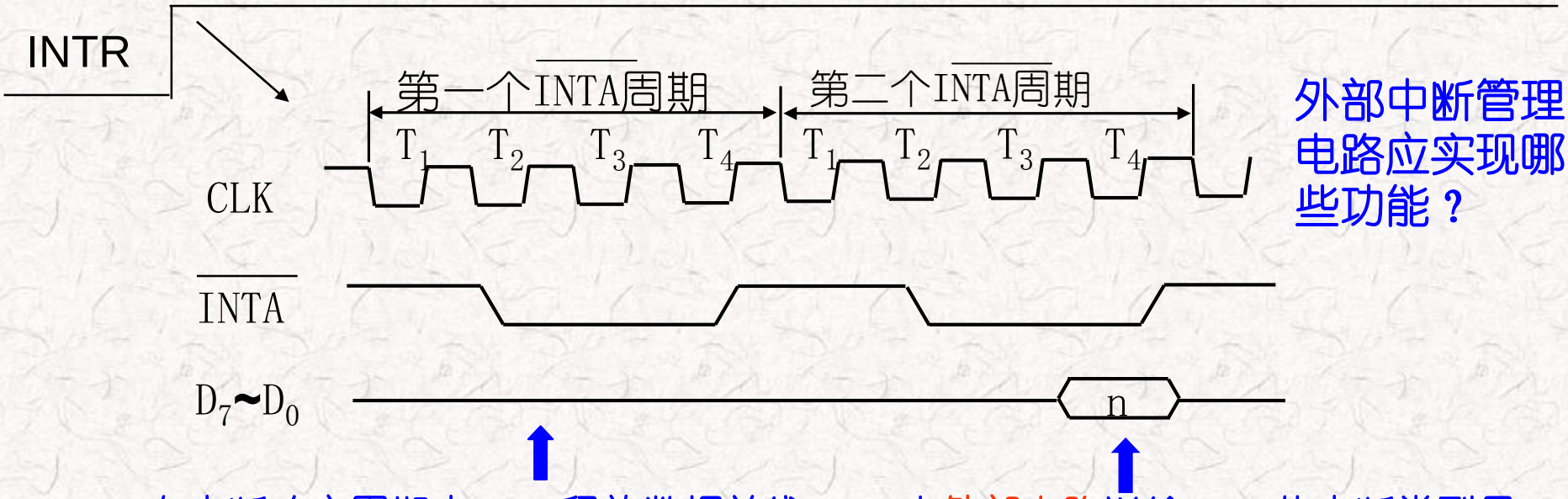
- INTR：高电平有效
- 由IF标志位决定是否响应，由 CLI 和 STI 指令禁止和开放
- 可以中断嵌套
- 响应时刻：在指令最后一个时钟周期采样INTR引脚状态
- 响应过程：
- 如IF=1，则CPU进入外中断响应周期， /INTA给出两周期低电平
- 中断类型号由外部电路在/INTA低电平期间送到数据总线上

外中断的特点2

由此可见，外部的外中断管理电路需要实现哪些功能？

- INTR：高电平有效
- 由IF标志位决定是否响应，由 CLI 和 STI 指令禁止和开放
- 可以中断嵌套
- 响应时刻：在指令最后一个时钟周期采样INTR引脚状态
- 响应过程：
- 如IF=1，则CPU进入外中断响应周期， /INTA给出两周期低电平
- 中断类型号由外部电路在/INTA低电平期间送到数据总线上
- 某中断请求被响应后，该请求信号应被清除，避免重复中断

8086 可屏蔽外中断响应周期



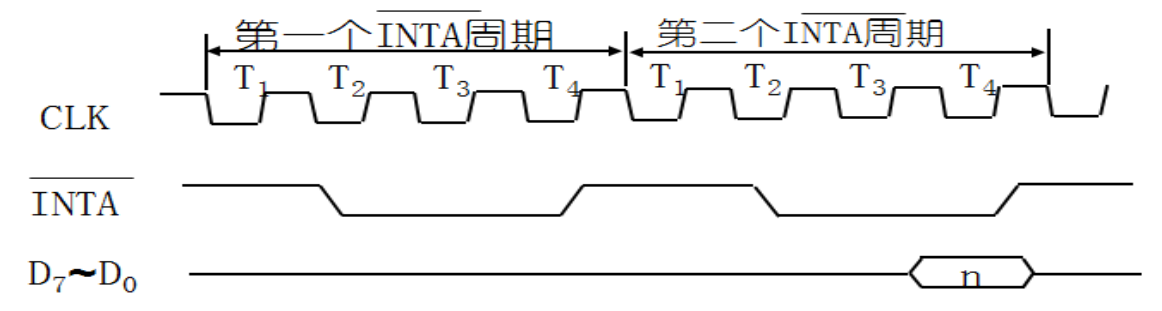
外部中断管理电路应实现哪些功能？

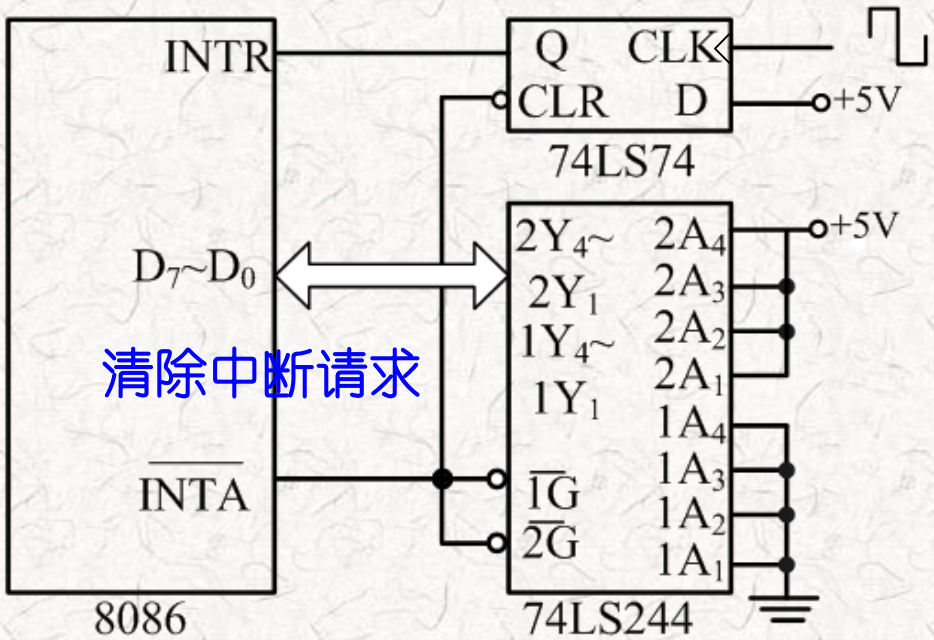
在中断响应周期中8086释放数据总线控制权，处于输入（高阻）状态

由外部电路送给8086的中断类型号

外中断管理电路至少应能够：（1）产生高电平的INTR，（2）在/INTA低电平期间发送类型号n到数据总线，（3）在/INTA低电平后清除INTR

8086如何获得外中断的类型号？





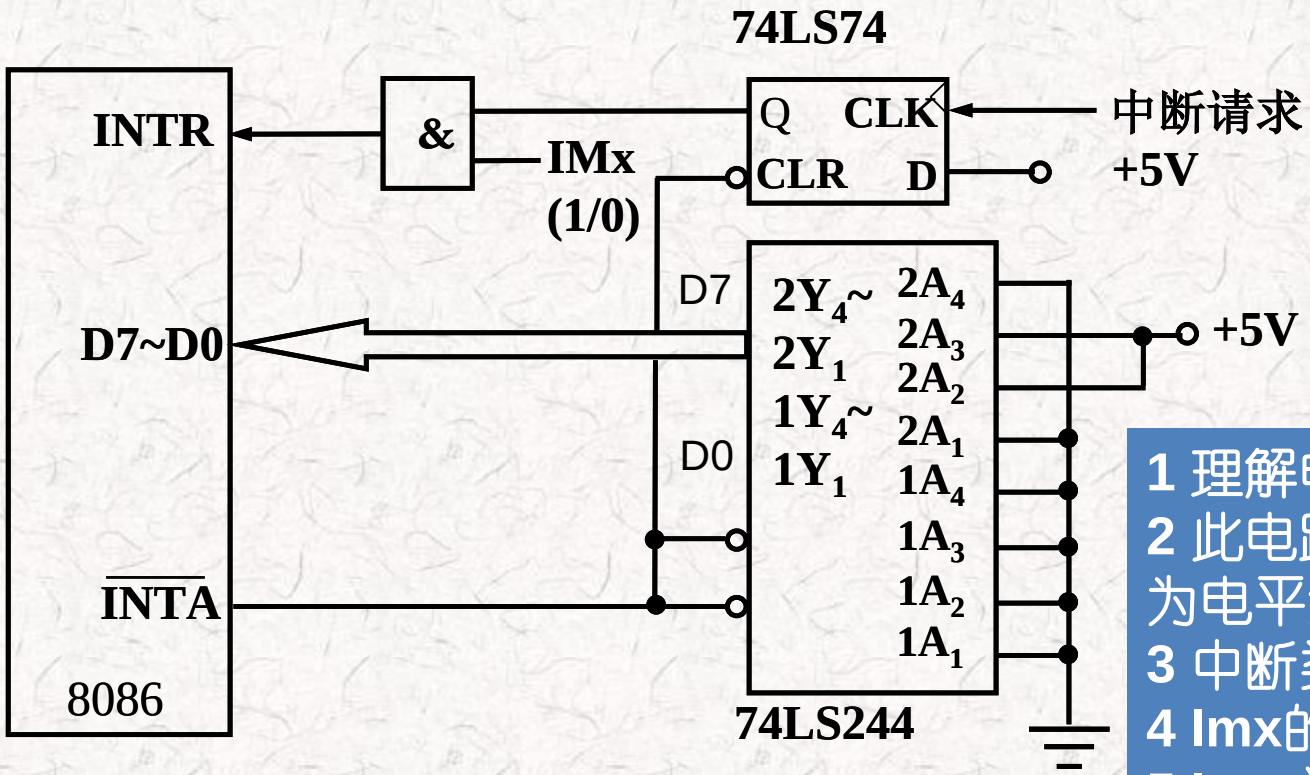
来自外设

每秒一次发出中断请求

清除中断请求

送出中断类型号

中断管理电路练习



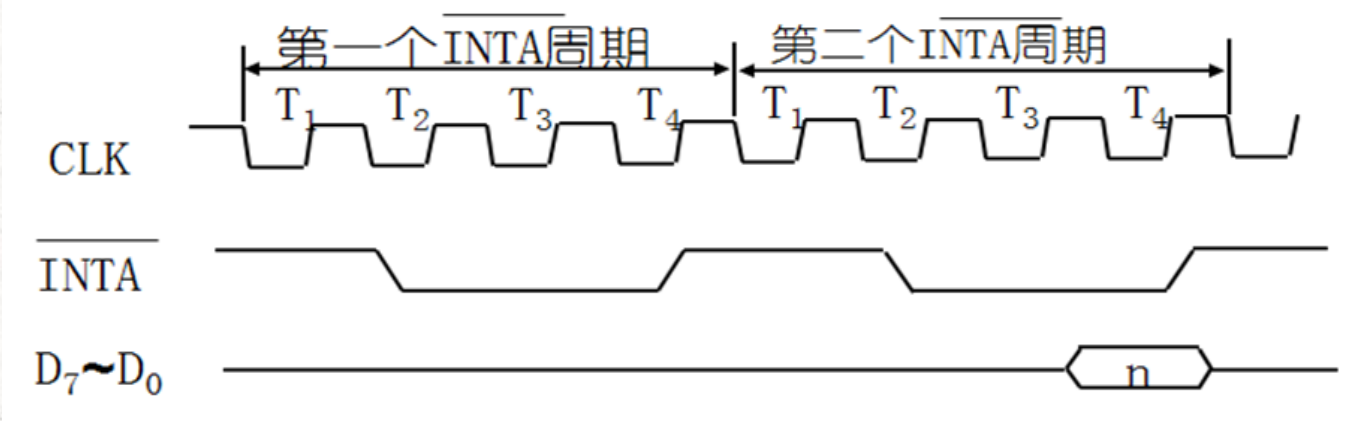
$D7\sim D0 = 2Y_4\sim 2Y_1 1Y_4\sim 1Y_1$

1 理解电路各部分功能
2 此电路的中断请求信号应为电平还是跳变沿有效？
3 中断类型号？
4 Imx的作用？
5 Imx应如何实现使其可由CPU配置？

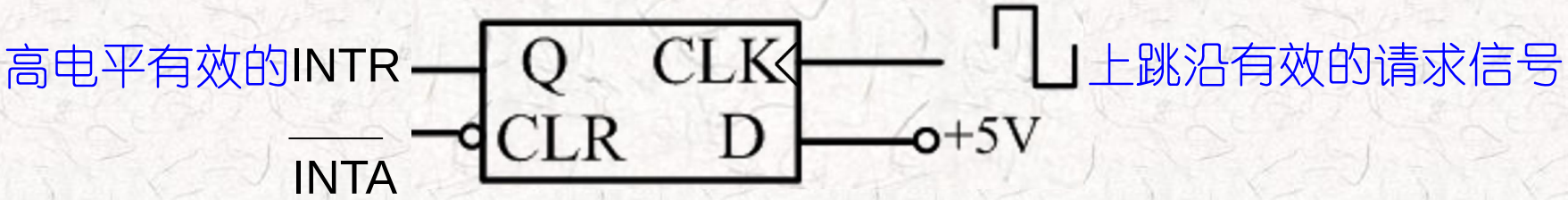
外中断管理电路功能总结

- 外中断响应周期
- 响应INTR时发出低电平的INTA信号，并要求在期间得到中断类型号
- 中断信号的产生和清除
- 将外设的请求转为符合CPU要求的中断请求信号，并且在得到响应后清除该请求
- 屏蔽某些中断源
- 根据程序员的要求屏蔽某些外设的中断请求
- 提供中断类型号
- 根据8086的要求在外中断中断响应周期将中断类型号送到数据总线低8位

外中断管理电路功能-发起与清除

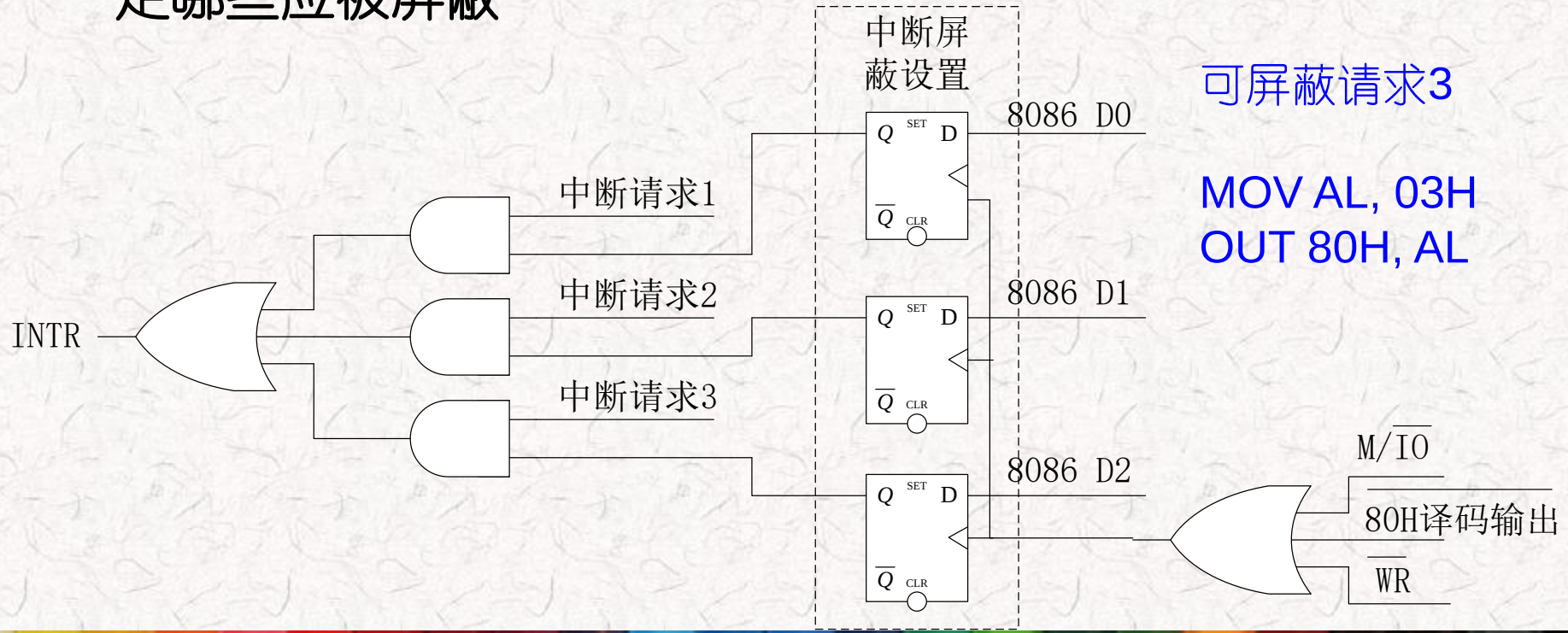


- 记录外设发起的中断请求，并在响应后清除该记录信息
- 通过带清零端的触发器实现



外中断管理电路功能-整合与屏蔽

- 1 通过与逻辑实现屏蔽
- 2 通过构造输出接口构造中断屏蔽寄存器由程序员编程决定哪些应被屏蔽

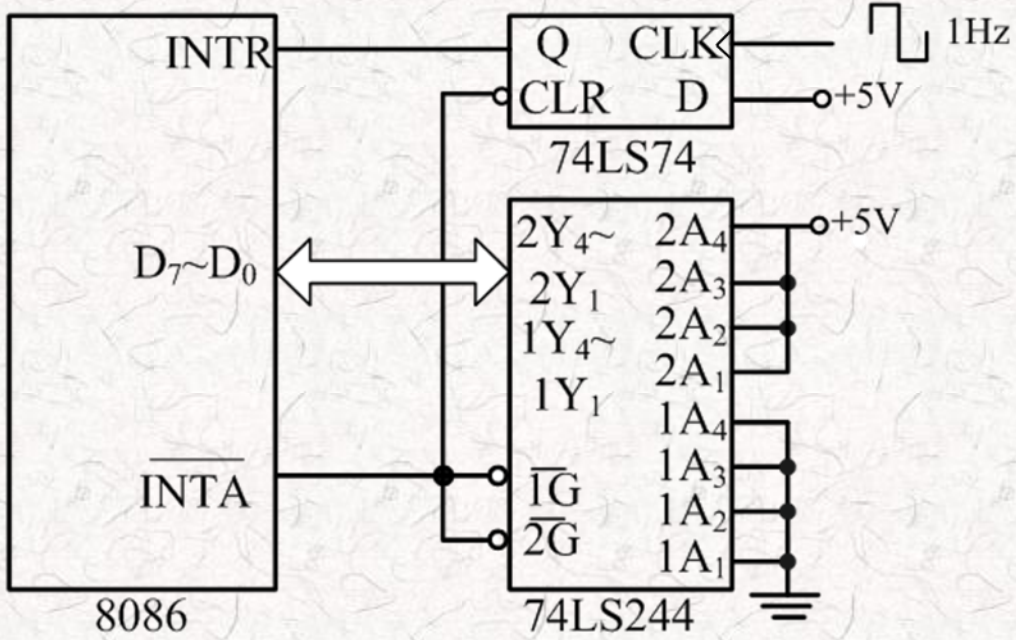


可屏蔽请求3

MOV AL, 03H
OUT 80H, AL

外中断管理电路功能-产生中断类型号

- 将中断类型号在/INTA低电平期间送到D7:0
- 通过三态驱动器实现，选通信号连接到/INTA



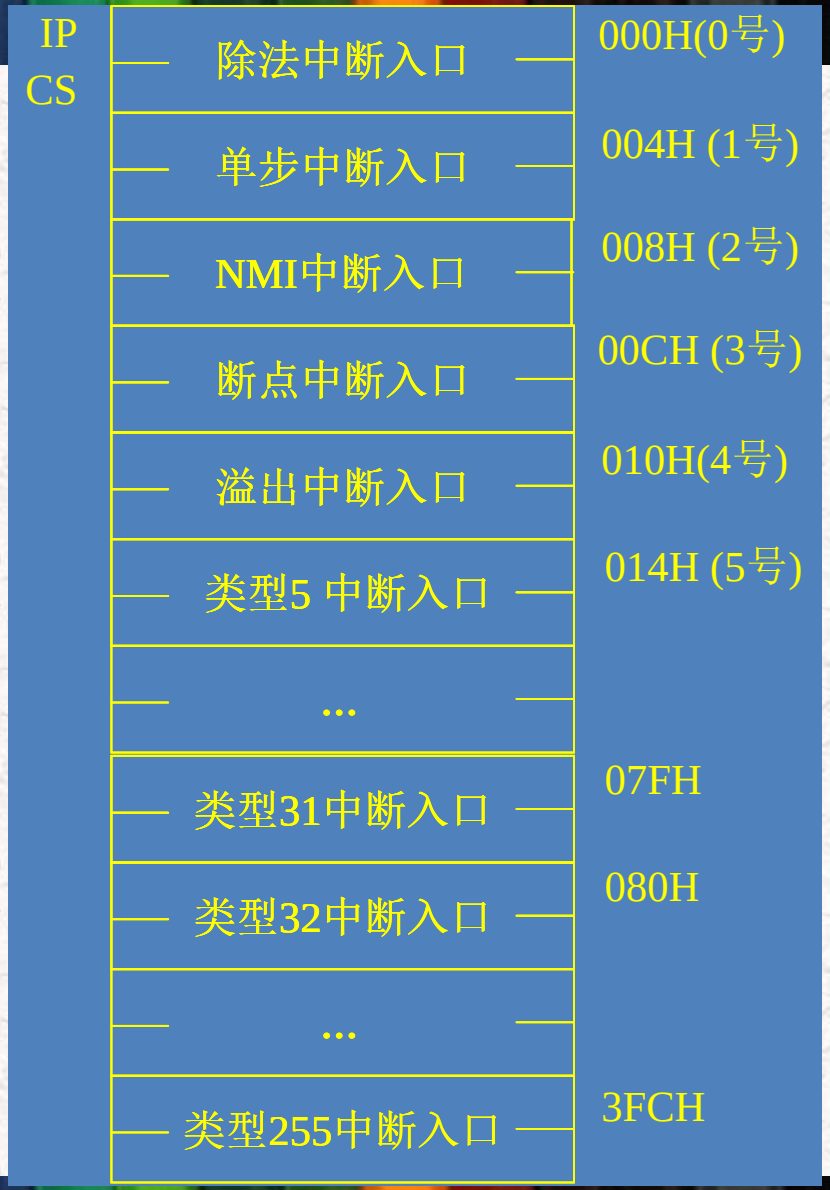
如何根据中断类型号得到中断向量？

- 内中断和NMI自带中断类型号n
- INTR从外部电路得到中断类型号n

- 8086根据中断类型号n查“中断向量表”得到中断向量

中断向量表

- 存放256个中断向量
- 每个中断向量占4字节
- 16b偏移地址，16b段地址
- 1K的中断向量表，位于8086存储空间物理地址00000H~003FFH
- 中断向量在表中地址：
- 4×n~4×n+3
- 查表是硬件自动完成的
- 当前CS、IP入栈
- 4*n,4*n+1->IP，4*n+2,4*n+3->CS



如何将中断向量写入中断向量表？

- 中断向量是程序员写入的
- 何时将中断向量写入中断向量表？
- 写入方法：
- 绝对地址定义法：使用变量定义伪指令DD
- 直接指令装入法：使用MOV指令
- 使用DOS功能调用

绝对地址定义法

在段地址为0，偏移地址为n*4的单元（即物理地址n*4）定义了INT_VCE标号所在的地址

```
INT-TBL     SEGMENT AT 0
            ORG     n*4     ；n是某个中断类型号
        DD  INT_VCE
INT-TBL     ENDS
            …
```

注意，物理地址0开始是向量表，此处并非定义数据段

```
MCODE SEGMENT                       ；代码段
ASSUME CS:MCODE, DS:DATA, SS:STACK
        …
INT_VCE     PROC    FAR     ；中断服务程序
            …
            IRET
INT_VCE     ENDP
MCODE ENDS
```

使用绝对地址定义法初始化举例1

设检测有符号加法运算，如发生溢出，则提示“overflow”，程序设计如下：

```
INT SEGMENT AT 0
     ORG 4*4H
  DD INT_O
INT ENDS

DATA SEGMENT
P DB ‘overflow$’
DATA ENDS

STACK SEGMENT
TT DW 100 DUP(0)
STACK ENDS
```

```
CODE SEGMENT
ASSUME CS:CODE, DS:DATA,
SS:STACK
START: MOV AX, STACK
          MOV SS, AX
          MOV SP, 200
          MOV AX, DATA
          MOV DS, AX

          MOV AL,-128
          ADD AL, -1
          INTO
          HLT
```

```
INT_O PROC FAR
          LEA DX,P
          MOV AH,9
          INT21H
          IRET
INT_O ENDP


CODE ENDS
END START
```

使用绝对地址定义法初始化举例2

设使用了可屏蔽外中断10H和20H，对应的中断服务程序入口地址分别为INT_10H，和INT_20H，则程序设计如下：

```
INI SEGMENT AT 0
    ORG 40H
  DD INT_10H
    ORG 80H
  DD INT_20H
INT ENDS

DATA SEGMENT
A1 DB  ?
DATA ENDS

STACK SEGMENT
TT DW 100 DUP(0)
TOP EQU $-TT
STACK ENDS
```

```
CODE SEGMENT
ASSUME CS:CODE, DS:DATA,
SS:STACK
START: MOV AX, STACK
       MOV SS, AX
       MOV SP, TOP
       MOV AX, DATA
       MOV DS, AX
       STI
       ……
```

```
INT_10H PROC FAR
        ……
        IRET
INT_10H ENDP

INT_20H PROC FAR
        ……
        IRET
INT_20H ENDP

CODE ENDS
END START
```

直接指令装入法

```
…
XOR     AX, AX
MOV     DS, AX                          ; 数据段指向0地址
MOV     AX, 2000H
MOV     [01ACH], AX                     ;  01ACH ÷ 4 = 6BH
MOV     AX, 1000H
MOV     [01ACH+2], AX
…
```

这段程序设置中断类型 **6BH** 对应的中断向量（即中断服务程序首地址）为 **1000H：2000H** 。

根据中断服务程序标号直接指令装入

```
XOR     AX, AX
MOV     DS, AX                          ; DS指向地址0
MOV     AX, OFFSET INTVCE               ; 中断程序偏移地址
MOV     [01ACH], AX                     ; 6B*4=1ACH
MOV     AX, SEG INTVCE                  ; 中断程序段基址
MOV     [01ACH+2], AX                   ; 存入段基址
重新设置DS指向数据段
…
INTVCE  PROC  FAR                       ; 中断服务程序,远过程
              …
              IRET
INTVCE  ENDP
```

使用直接指令装入法初始化举例

设使用了可屏蔽外中断11H和22H，对应的中断服务程序入口地址分别为INT_11H，和INT_22H，则程序设计如下：

```
DATA SEGMENT
A1 DB 'for example'
DATA ENDS

STACK SEGMENT
TT DW 100 DUP(0)
STACK ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA,
SS:STACK
START:  MOV AX, 0
        MOV DS, AX
      MOV AX,OFFSET INT_11H
        MOV [44H], AX
        MOV AX, SEG INT_11H
        MOV [46H], AX
MOV AX,OFFSET INT_22H
        MOV [88H], AX
        MOV AX, SEG INT_22H
        MOV [8AH], AX

        MOV AX, STACK
        MOV SS, AX
        MOV SP, 200
        MOV AX, DATA
        MOV DS, AX
        STI
        ……

INT_11H PROC FAR
        ……
        IRET
INT_11H ENDP

INT_22H PROC FAR
        ……
        IRET
INT_22H ENDP

CODE ENDS
END START
```

DOS功能调用法

- 设置中断向量
  - 把由AL指定的中断类型的中断向量DS:DX放置在中断向量表的相应位置中。
    - AH=25H 执行：INT 21H
    - AL=中断类型号
    - DS:DX=中断向量
- 取中断向量
  - 把由AL指定的中断类型的中断向量从中断向量表中取到ES:BX中
    - 预置： AH=35H 执行：INT 21H
      AL=中断类型号
    - 返回： ES:BX=中断向量

```
中断服务程序 INTVCE 类型号 n
MOV AX, SEG INTVEC
MOV DS,AX
MOV DX, OFFSET INTVEC
MOV AH, 25H
MOV AL,  n
INT   21H
```

中断基本概念练习

- 8086通过 _____把中断申请和中断服务程序关联起来
- 8086的中断向量表存放在物理地址 _____H ~ _____H 范围内
- 8086的（硬件自动 OR 软件编程）方式实现在中断向量表中查找中断向量

中断向量小练习1

- 1. INT 11H的中断向量存放在0000:0044H开始的连续4个字节
- 2. 若中断向量表中(0040H)=240BH，(0042H)=0D169H，
- 请问这里（0040H）的物理地址是多少？ 00040H
- 填写相应存储单元内容
- 对应的中断类型号是？ 10H
- 中断服务程序入口地址？

0D169H:240BH

0BH	0000:0040
24H	0000:0041
69H	0000:0042
0D1H	0000:0043

中断向量的小练习2

- 假设NMI（类型号2）中断的服务程序为INT_NMI，请填空采用绝对地址定义法完成中断向量设置

```
1  INT_VEC SEGMENT  AT  0
2     ORG  8
3     DD INT_NMI
4  INT_VEC ENDS
```

中断向量的小练习3

- 假设断点中断（类型号3）的服务程序为INT_BREAK，找出以下初始化和中断向量设置程序的错误

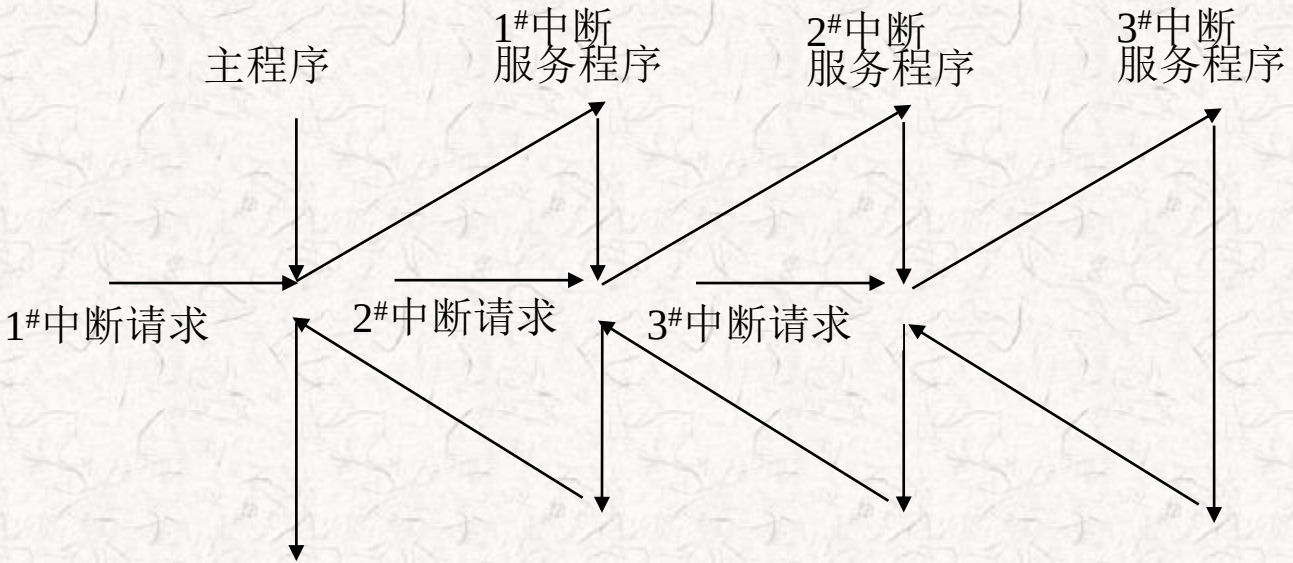
```
MOV SS, STACK          ;假设STACK为堆栈段
MOV SP, 100
MOV AX, DATA           ;假设DATA为数据段
MOV DS, AX
MOV WORD PTR [12H], OFFSET INT_BREAK
MOV WORD PTR [14H], SEG INT_BREAK
```

作业

- P271
- 1~4
- 6~7
- 20

中断嵌套的概念

- 响应低级中断的过程中发生高级中断，允许转去响应高级中断（前提是开中断）



中断优先级

- 除法错、INT n、INTO，断点中断；最高级，同一行的
- ；有同等优先级
- NMI ；次高级
- INTR ；较低级（如多个INTR，优先级
  ；由外部电路管理）
- 单步中断 ；最低级

8086系统只有可屏蔽外中断可以中断嵌套

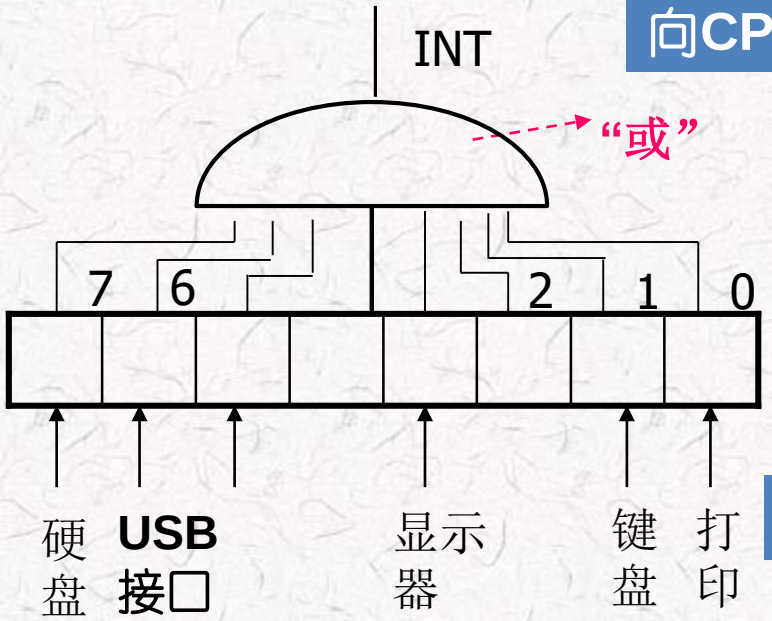
8086系统如何实现对外中断的中断优先级管理？

外中断优先级管理电路的功能和方法

- 功能
- 发起中断请求送给CPU
- 将发起中断中具有最高优先级的中断类型号发送给CPU

- 方法
- 软件查询法（需要少量硬件）
- 硬件优先级管理电路
- 可编程中断控制器（优先级管理芯片，例如8259A）

中断优先级管理方法--软件查询

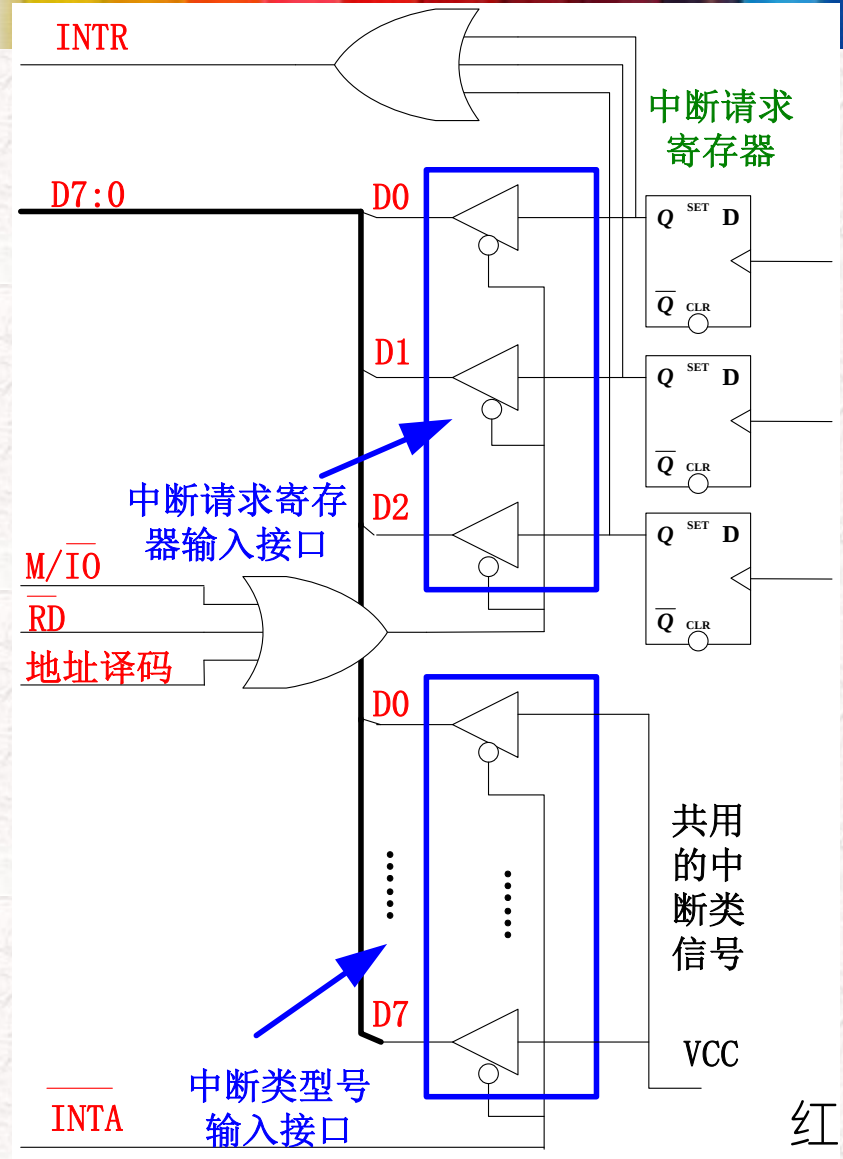


向CPU发起的中断请求

外设的中断请求

特点：所有中断共用一个中断类型号（类型号另行提供）
优点：节省硬件；
缺点：中断响应慢

查询方式的电路与编程示例

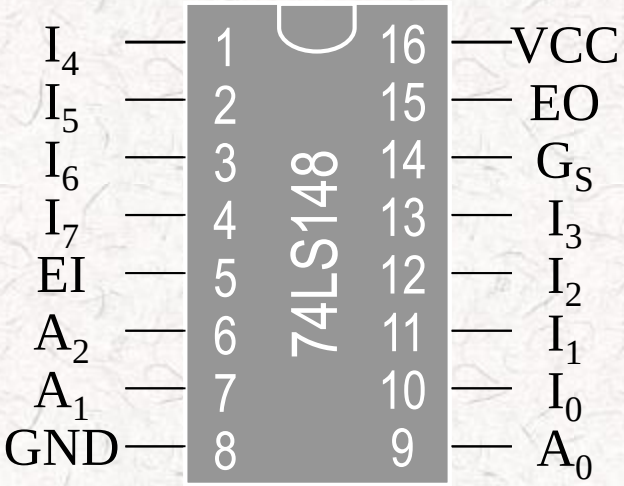


中断服务程序

```
...
IN AL,IPORT   ;取中断信息
TEST AL,80H   ;0#设备请求？
JNZ SEV0      ;是,转0 #设备服务
TEST AL,40H   ;否,1#设备请求？
JNZ SEV1      ;是,转1#设备服务
TEST AL,20H   ;否,2#设备请求？
JNZ SEV2      ;是,转2#设备服务
TEST AL,10H   ;否,3#设备请求？
JNZ SEV3      ;是,转3#设备服务
...
```

红色的信号来自8086

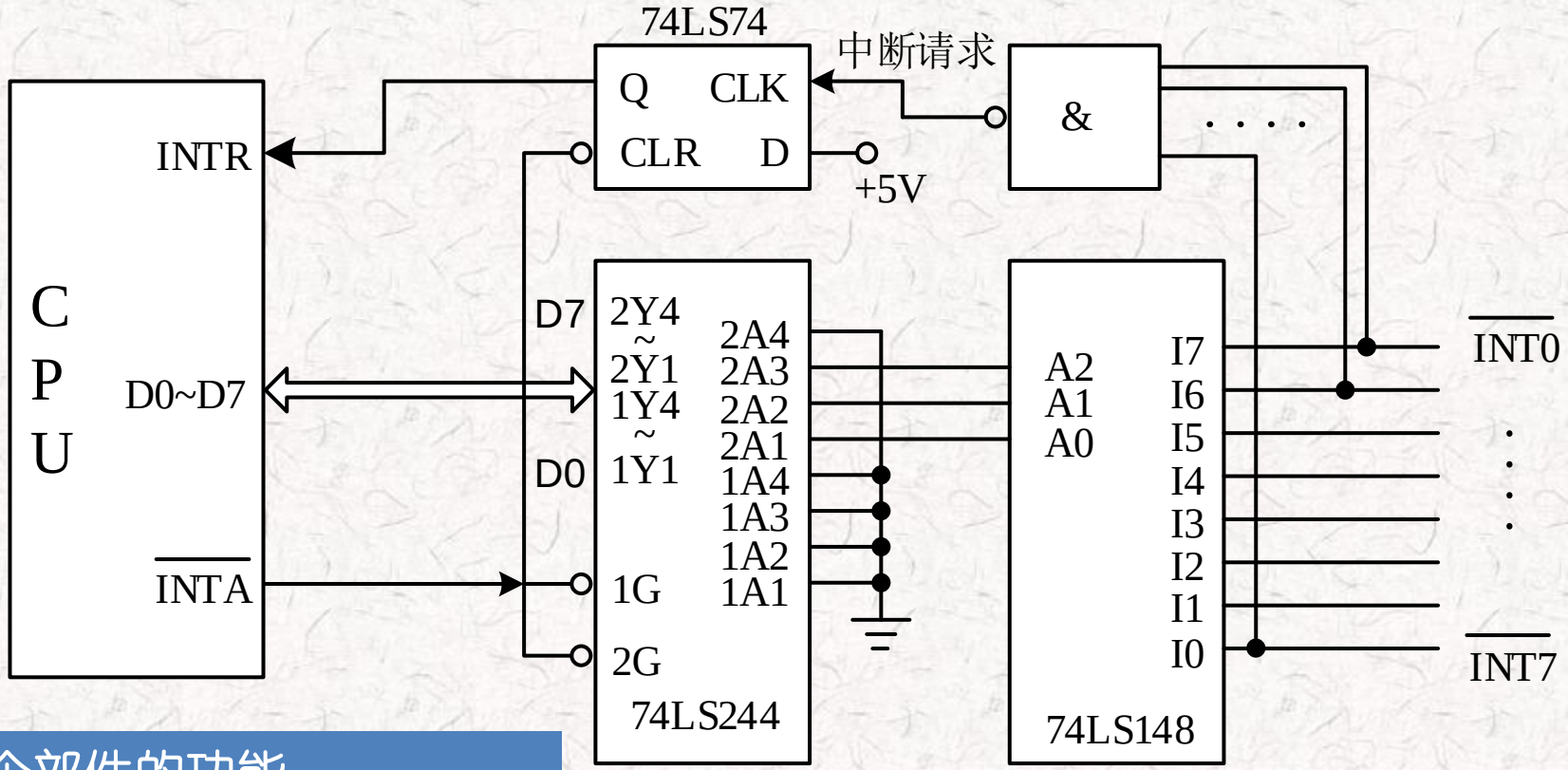
编码器管理中断优先级-74LS148



输入									输出				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	1	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	0	1	1	0	1
0	X	X	X	0	1	1	1	1	1	0	0	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

EI：输入的使能信号；**GS**:输出编码是否有效；**EO**：输出使能用于级联扩展

练习：编码器组成的优先级管理电路

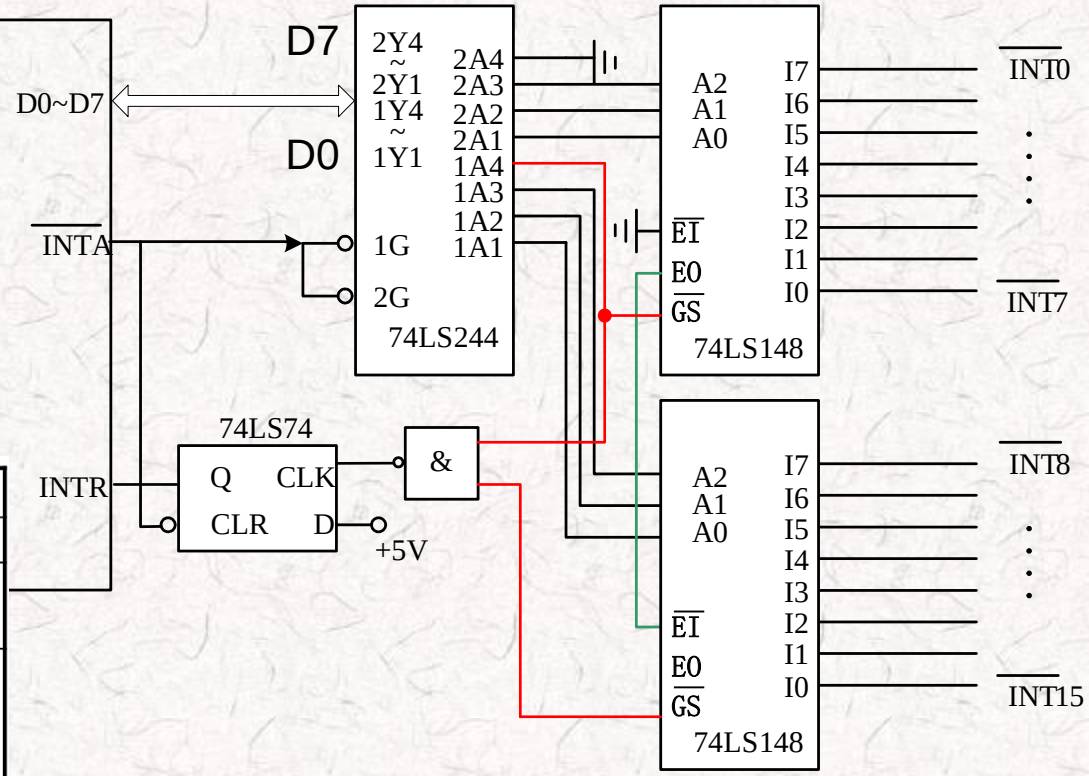


各个部件的功能

上图如同时发生多个中断请求，CPU如何响应？

如何用编码器实现更多路中断管理？

分析电路，请说明各中断优先级和中断类型号



输入								输出				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	1	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	0	1	1	0	1
0	X	X	X	0	1	1	1	1	1	0	0	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

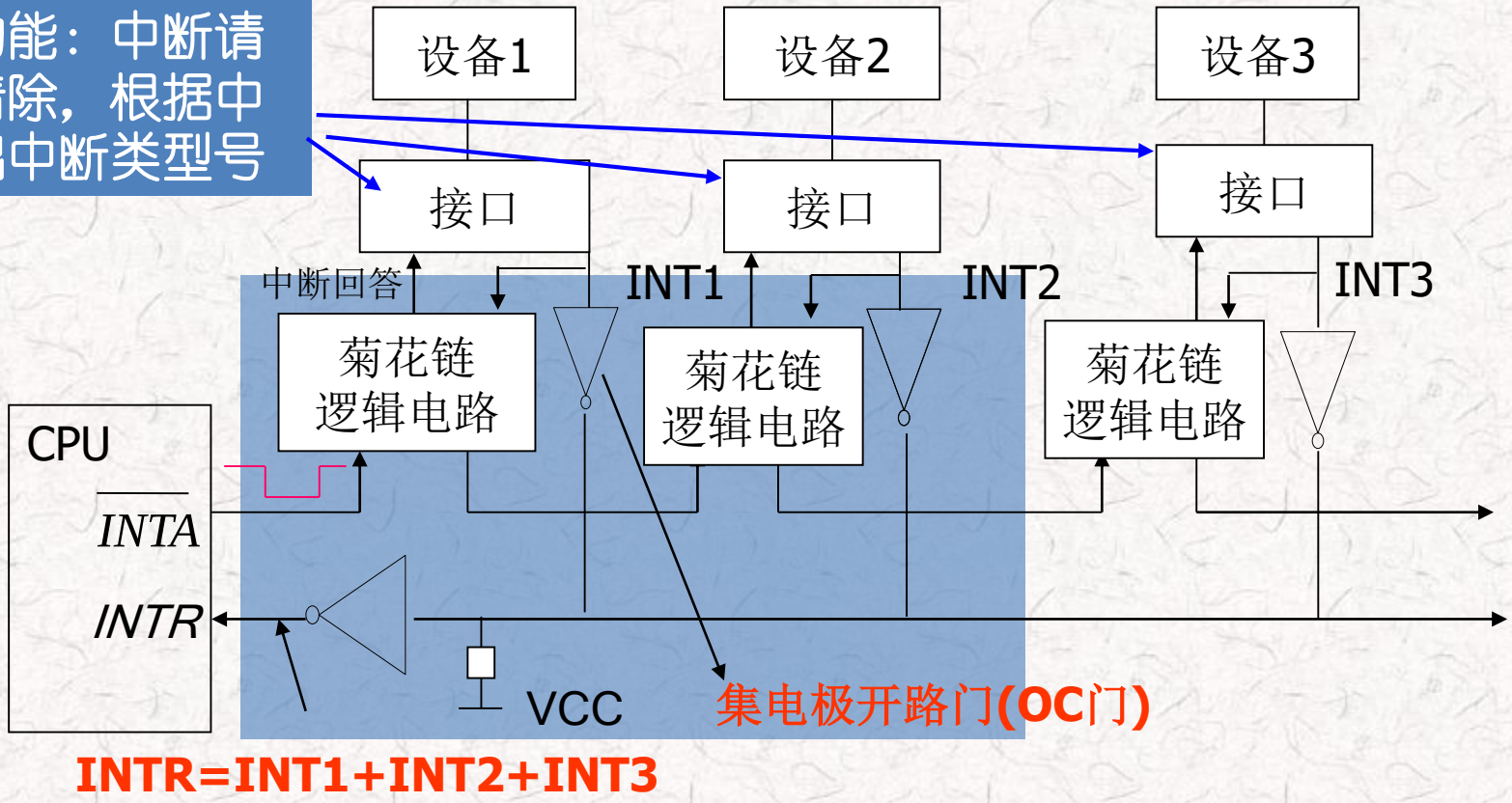
上片具有更高优先级

上片中断类型号：07H，17H~77H

下片中断类型号：78H，79H~7FH

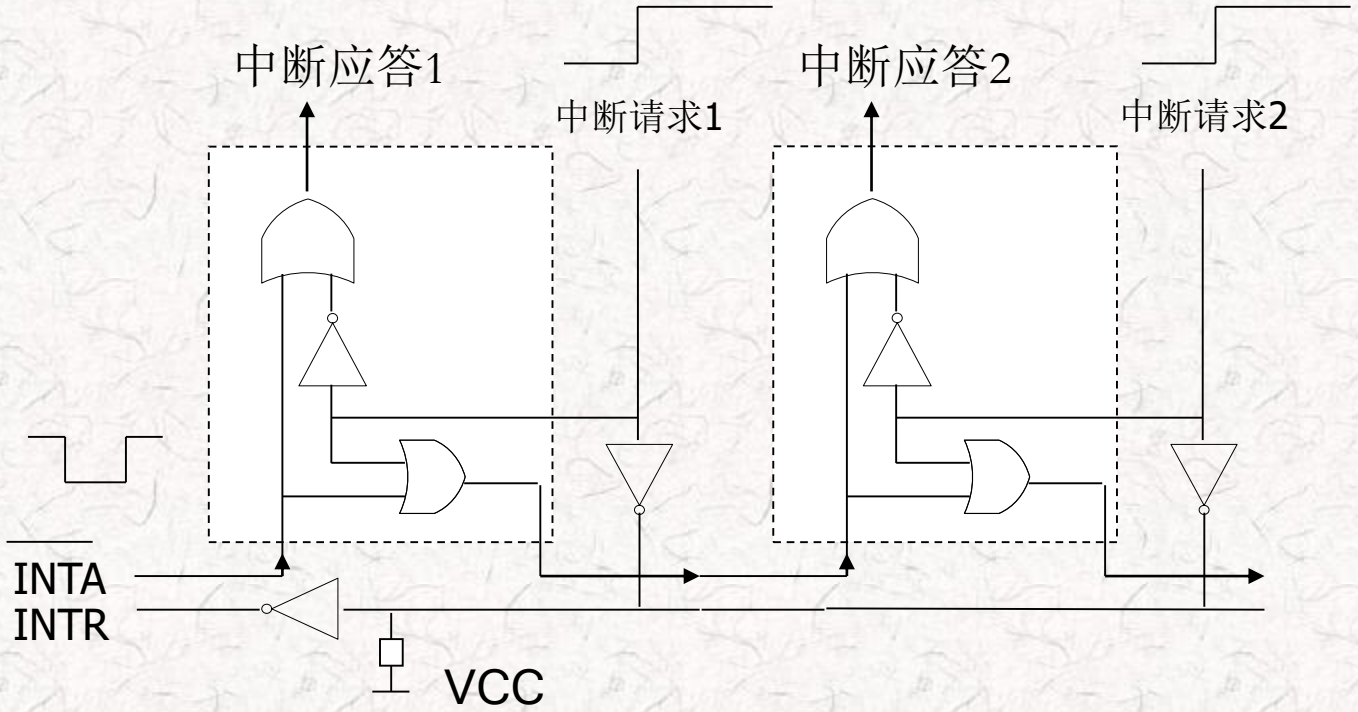
菊花链优先级管理电路--中断申请

接口电路功能：中断请求发起与清除，根据中断应答送出中断类型号



INTR=INT1+INT2+INT3

菊花链优先级管理电路--中断响应

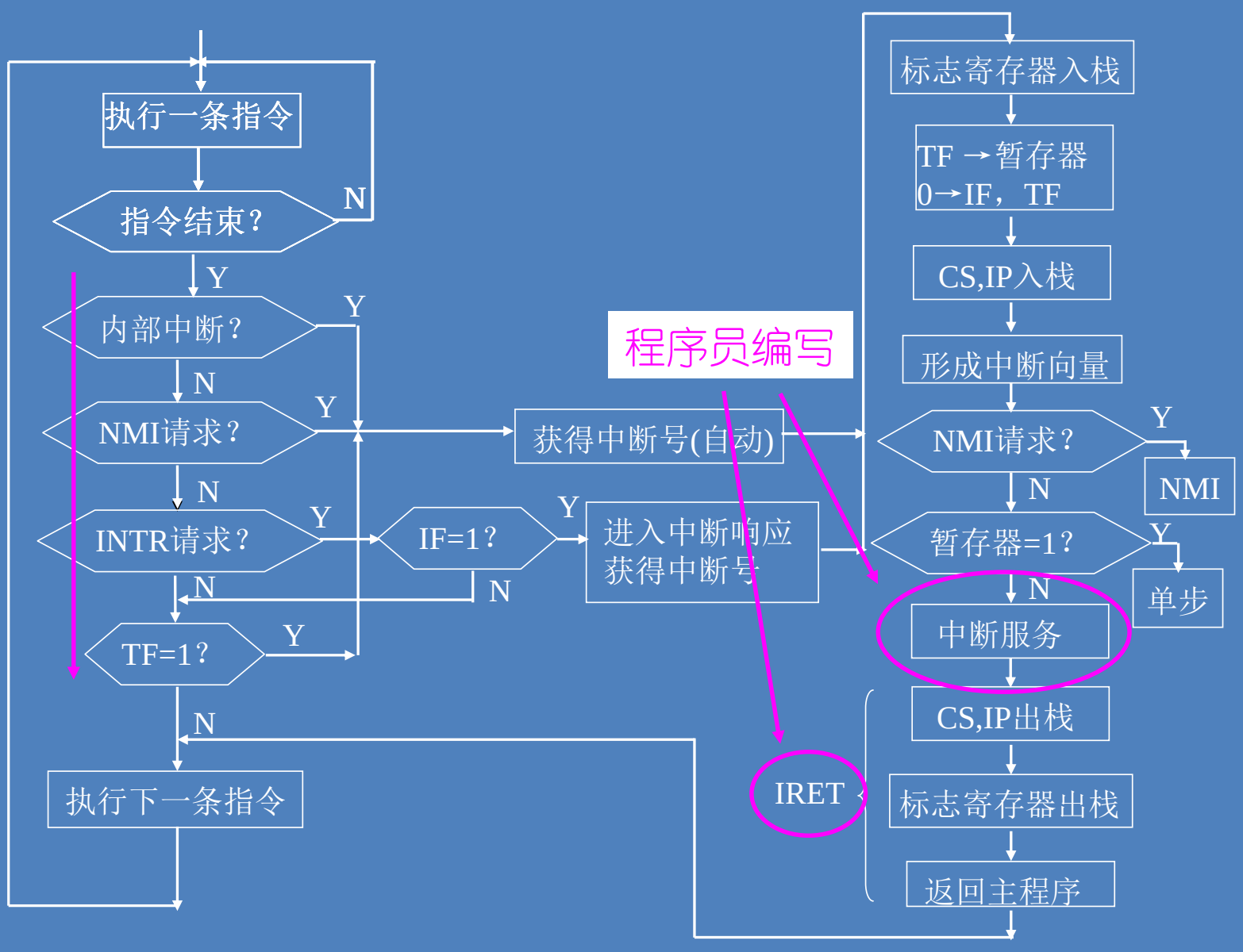


2.中断响应与处理流程

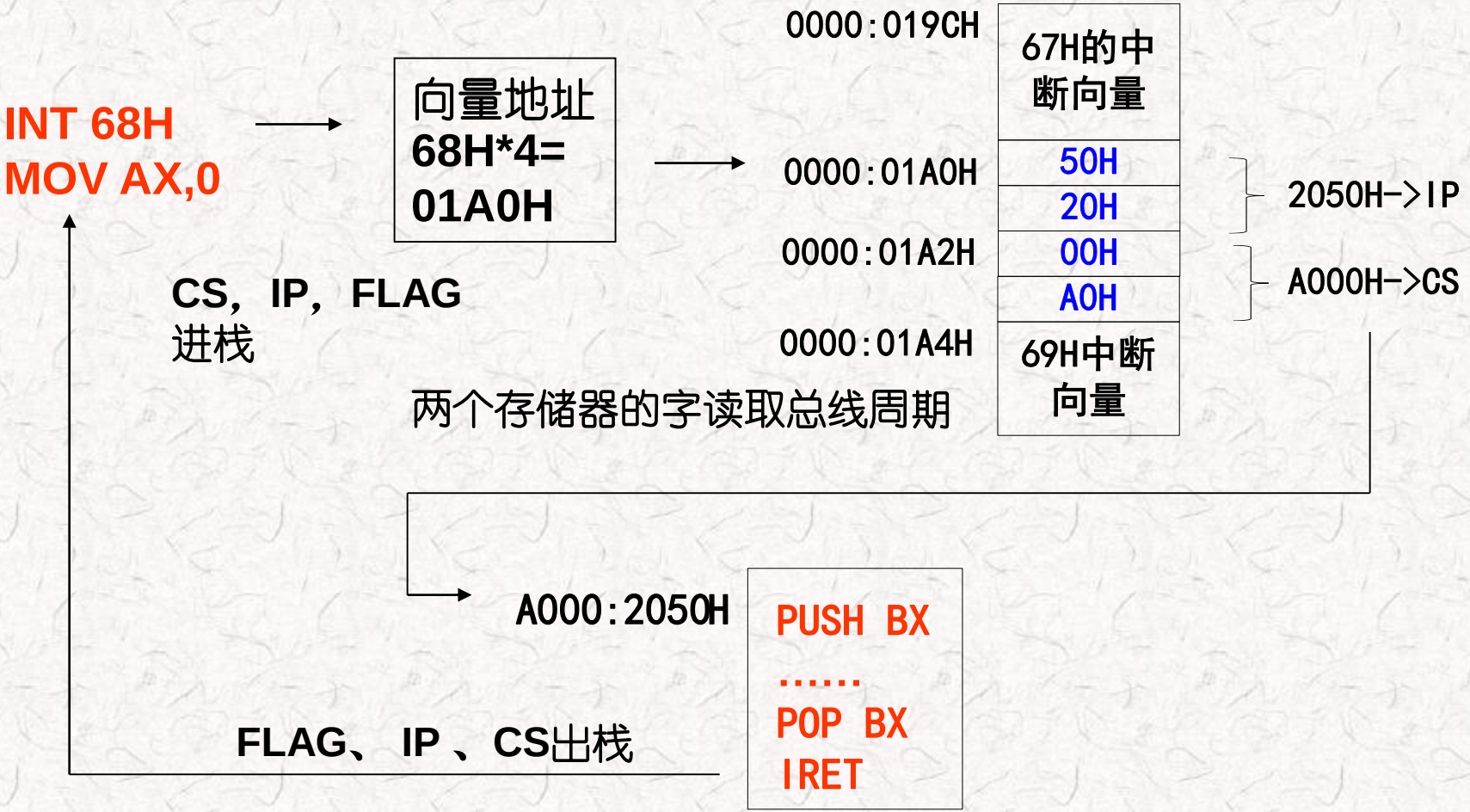
- 简述一般中断响应的过程
- 8086响应中断时自动执行什么堆栈操作？
- 8086退出中断时自动执行什么堆栈操作？
- 如何在中断服务程序中响应高级中断？
- 8086响应外中断时如何获得中断类型号？
- 中断服务程序结束前必须执行什么指令？

学习课本 第八章8.1.2节

☆中断响应过程


执行一条指令
指令结束？
N
Y
内部中断？
Y
N
NMI请求？
Y
N
INTR请求？
Y
N
TF=1？
Y
N
执行下一条指令
IF=1？
Y
N
获得中断号(自动)
进入中断响应
获得中断号
程序员编写
标志寄存器入栈
TF →暂存器
0→IF，TF
CS,IP入栈
形成中断向量
NMI请求？
Y
NMI
N
暂存器=1？
Y
单步
N
中断服务
CS,IP出栈
IRET
标志寄存器出栈
返回主程序

举例：类型号为68H的中断响应过程



主程序相关初始化操作

- 设置堆栈--SS和SP初始化
- 设置中断向量表—中断n的中断向量写入中断向量表
- 数据段和扩展段寄存器DS、ES初始化（可选）
- 开中断STI（可选，如使用可屏蔽外中断INTR）

中断服务程序

- 保护现场，将所有用到的寄存器和存储器压栈（可选）
- PUSH XX ；等 （可选）
- 开中断 STI （可选）
- 中断程序主体
- 关中断 CLI （可选）
- 恢复现场，将所有压栈的寄存器和存储器出栈（可选）
- POP XX ；等（可选，与压栈顺序相反）
- 中断返回 IRET

一个溢出中断的例子

两数相加未溢出则AL返回0，否则返回0FFH

```
DATA    SEGMENT
ADD1    DB   ?
ADD2    DB   ?
DATA    ENDS

STACK   SEGMENT
SPACE   DB   100  DUP(?)
TOP     EQU  LENGTH  SPACE
STACK   ENDS

CODE    SEGMENT
        ASSUME CS:CODE,DS:DATA,SS:STACK
START:  MOV   AX，STACK
        MOV   SS，AX
        MOV   SP，TOP
        MOV   AX, 0
        MOV   DS, AX        ;指向中断入口地址向量表
        MOV   DI, 04*4      ;溢出中断类型4
        MOV   AX,OFFSET INTOF
        MOV  [DI], AX
          ；设置中断服务程序偏移地址
        MOV   AX，SEG INTOF
        MOV [DI+2], AX
          ；设置中断服务程序段地址

        MOV   AX, DATA
        MOV   DS,AX
        MOV   BL,0 ;设置未溢出标志
        MOV   AL,ADD1
        ADD   AL,ADD2
        INTO
LOP1:   MOV   AL,BL
        HLT

INTOF   PROC FAR
        MOV   BL,0FFH
        IRET
INTOF   ENDP
CODE    ENDS
        END       START
```

以中断方式从输入端口80H读取1个字节，共读取10个字节，设中断类型号20H

```
DATA    SEGMENT
REC     DB   10 dup(?)
NUM     DW   0
DATA    ENDS

STACK   SEGMENT
SPACE   DB   100  DUP(?)
TOP     EQU  LENGTH  SPACE
STACK   ENDS

INT_TAB SEGMENT AT 0
        ORG 20H*4
   DD INT_REC
INT_TAB ENDS

CODE    SEGMENT
        ASSUME CS:CODE,DS:DATA,SS:STACK
START:  MOV   AX，STACK
        MOV   SS，AX
        MOV   SP，TOP
        MOV   AX, DATA
        MOV   DS, AX
        STI
WAIT:   ……       ;处理其它任务
        CMP  WORD PTR NUM, 10
        JE     PROC_DATA
        ……
PROC_DATA:   ……
        HLT

INT_REC PROC FAR
        PUSH AX              ;保护现场
        PUSH SI
        MOV SI, NUM
        IN AL, 80H           ;收数据
        MOV REC[SI],AL
        INC SI
        MOV NUM, SI
        POP SI               ;恢复现场
        POP AX
        IRET
INT_REC ENDP
CODE    ENDS
        END      START
```

指令中断与子程序调用的区别

- 形式不同：CALL，INT n，或 INTO
- 指明中断服务程序和子程序的入口地址方式不同
- 子程序：入口地址在CALL 指令中直接指明，如：CALL 子程序名，CALL FAR PTR 子程序名，或者存放在某个变量中，如：CALL WORD PTR [BX]；CALL DWORD PTR [BX+SI]
- 中断服务程序：通过中断类型号到中断向量表中找入口地址
- 所执行的堆栈操作不同
- 子程序调用：近调用IP进栈，远调用CS、IP进栈，子程序返回：对应的IP或IP、CS出栈
- 进入中断：CS、IP、FLAG压栈，中断返回： FLAG、IP、CS出栈
- 返回指令不同：RET和IRET
- 能否嵌套不同：子程序调用可以嵌套子程序调用，软件中断不可以

中断响应过程小练习

- 8086CPU中SS=1000H，SP=0400H，flags=0240H。则CPU响应中断开始执行中断服务程序第一条指令时SS=____H，SP=____H，flags=____H。（标志位DF、IF、TF为标志寄存器flags的D10、D9和D8位）

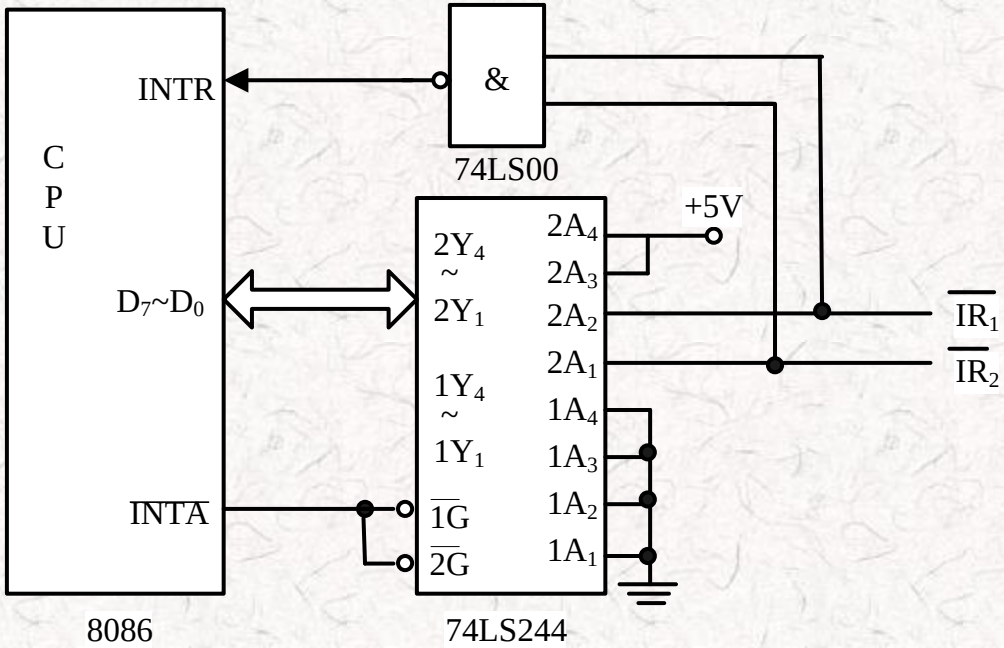
- 进入中断服务程序需要执行PUSHF，PUSH IP，PUSH CS吗？为什么？

- 8086中断服务程序退出前是否需要STI开中断？为什么？

综合练习1

（1）分析电路，系统如何获得外部中断的中断类型码？

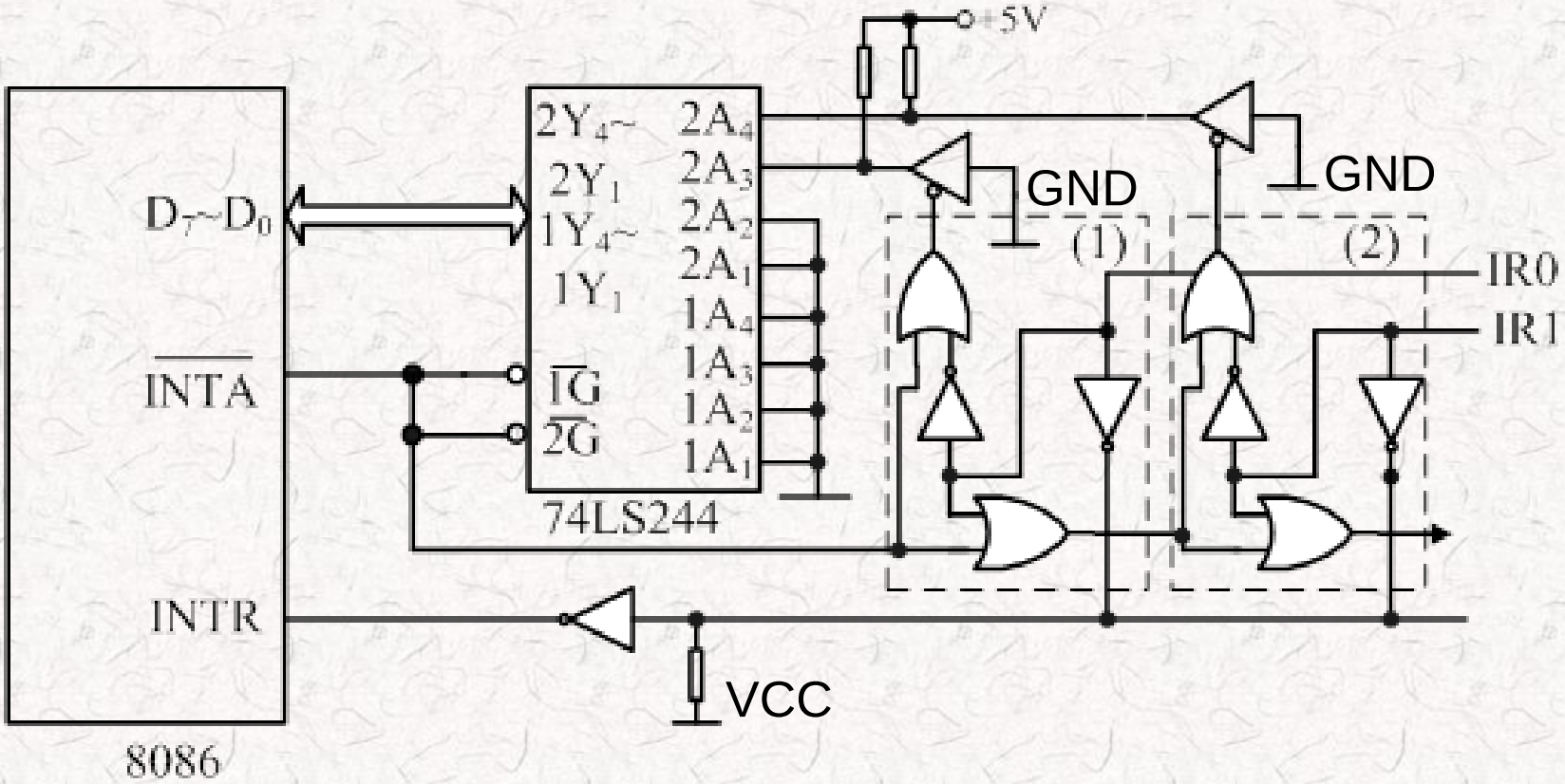
（2）写出电路中所有可屏蔽中断类型号及其中断向量在内存中的地址范围。若要求中断请求IR1具有高优先级，应如何设置中断向量



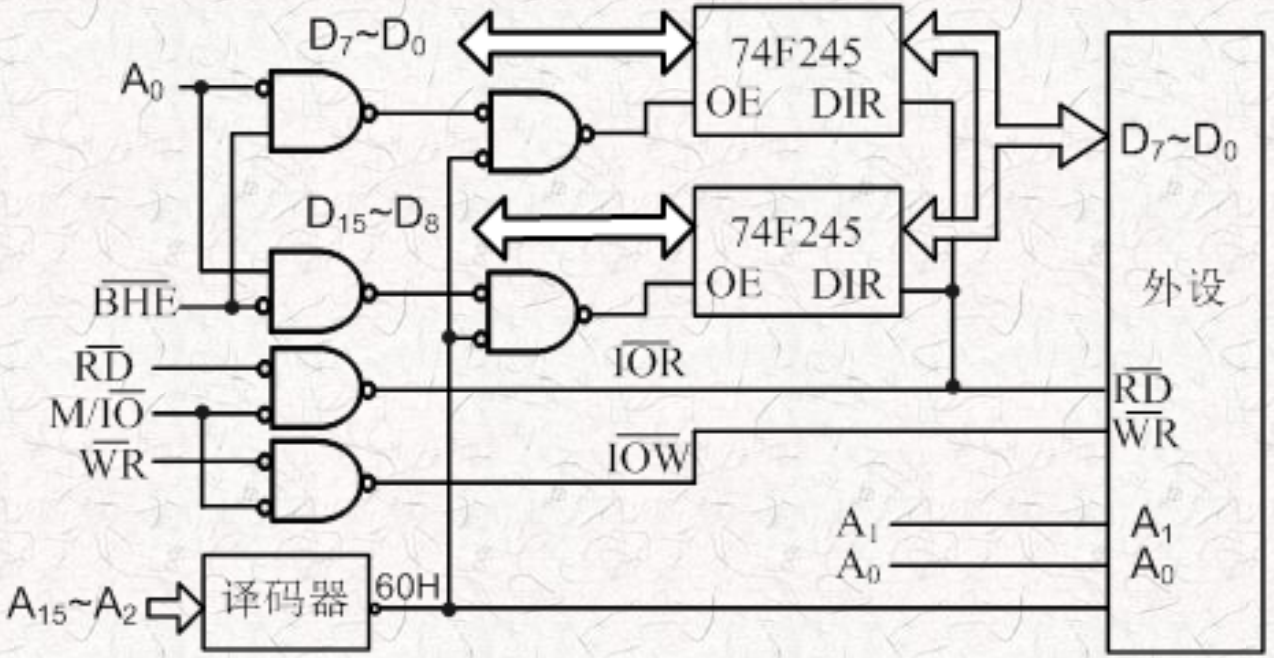
（3）外部中断源IR1的中断服务程序的首地址为ISR1，编写主程序中为该外部中断源建立中断向量的程序

练习2

- 图为8086CPU的外部可屏蔽中断扩展电路，IR0，IR1是外部设备的中断请求信号，高电平有效。
- （1）说明CPU响应中断过程，类型码各为多少，优先级如何？
- （2）为两个中断请求设置中断向量

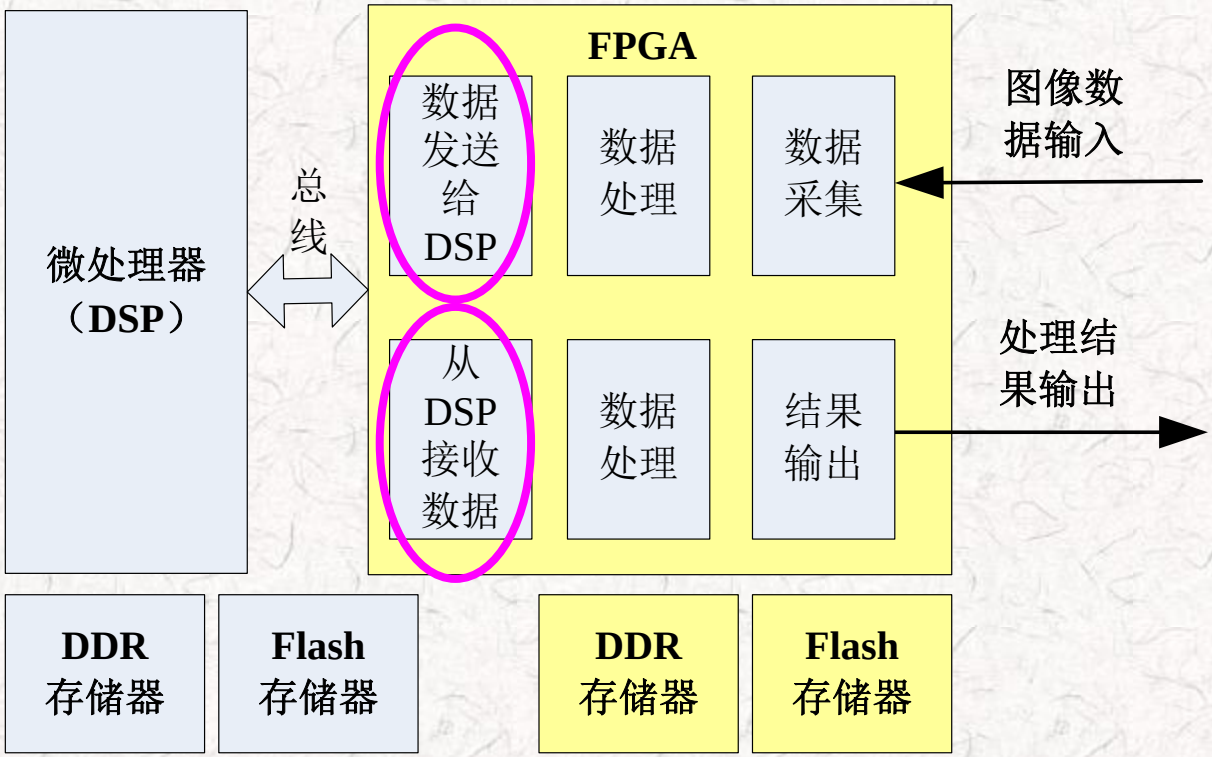


- 图中8086微处理器通过字节交换逻辑电路连接8位外设。要求：
- 1）写出字节交换逻辑电路的功能。
- 2）外设有几个端口，写出端口的地址范围，分别是输入端口还是输出端口？
- 3）编程实现分别读入各端口当前数据，存储到当前数据段中从地址PDATA开始的内存中，并将读入数据取反，分别送回各端口。



综合练习3

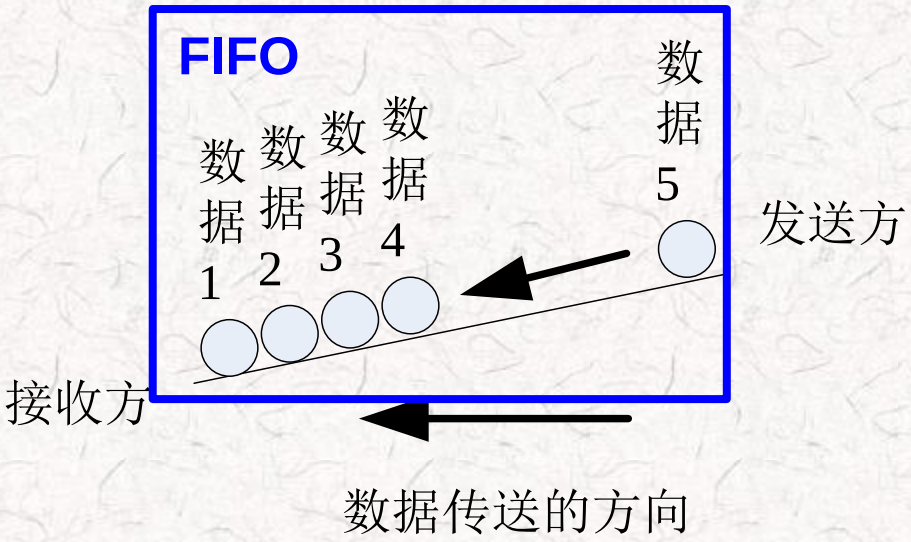
应用举例-现代图像信息处理系统典型架构

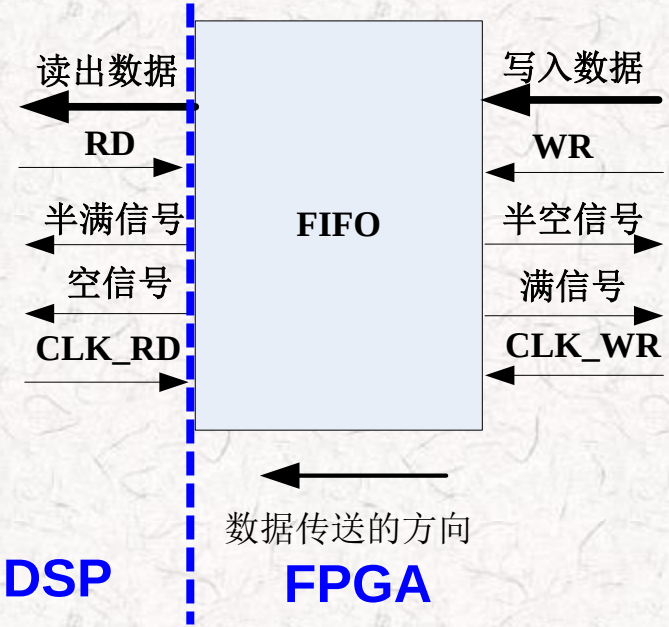


特点：DSP需要频繁读写大批量数据　　通过中断+DMA减少CPU传输开销

FPGA如何发送批量数据给DSP？

FIFO是数据缓存，双方可以不同的速率/时刻顺序写入或读出数据



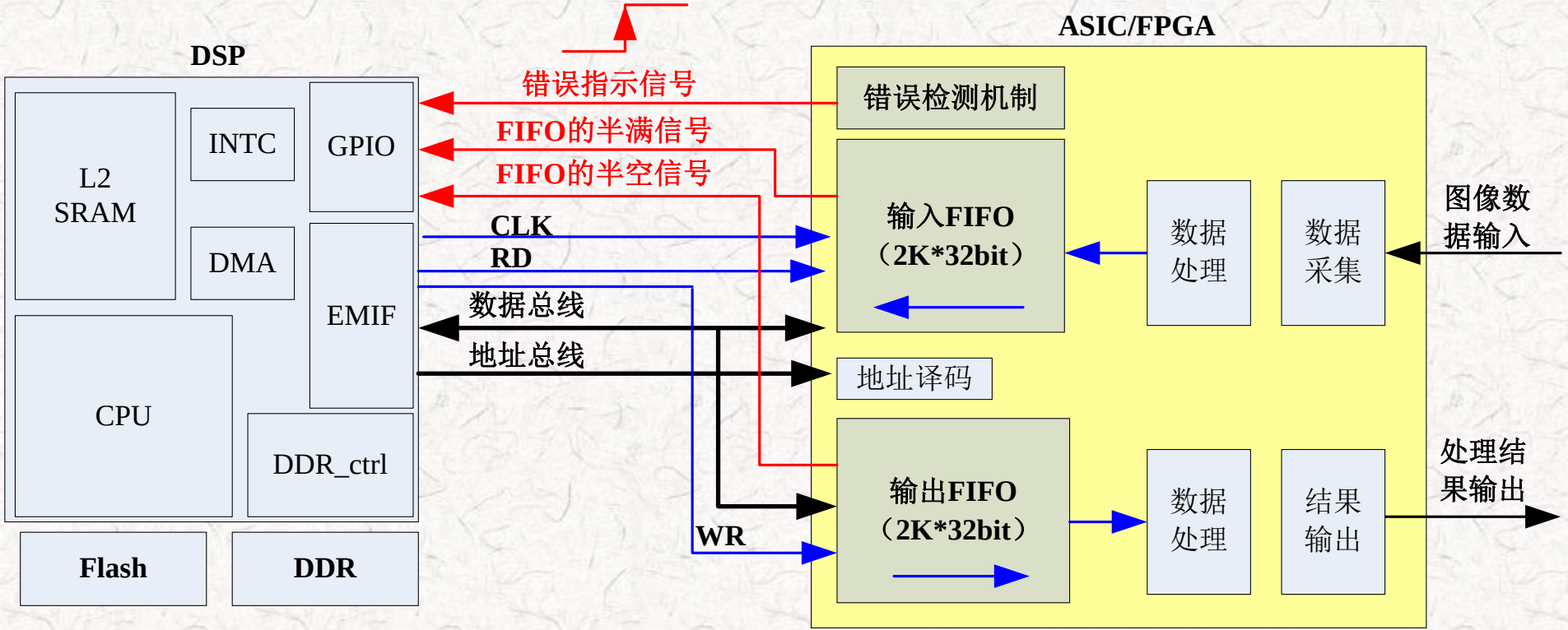


FIFO给出信号指示其中暂存数据的数量

DSP检测到半满信号后连续读走相应数量的数据

FPGA：有数据&非满则写

典型的图像信息处理系统信号流图



FPGA中输入FIFO半满->信号跳变引发DSP内部事件->关联启动DMA

DMA从固定源地址批量传输到L2或DDR ->启动DMA完成中断->数据处理

DSP图像处理软件中数据传输部分举例

- 若一帧图像为64K*32bit，输入FIFO深度8K*32bit，每次半满启动DMA传输，可以传输4K*32bit，共需传输16次，获取一帧图像后开始执行数据处理算法
- 主程序初始化：
- 1 设置DMA某通道与半满信号连接的GPIO上跳沿事件关联
- 2 为该通道设置16个DMA传输配置表：令源地址不变（数据来自FIFO端口），目的地址自动增加（数据写入存储器），令配置表每次传输后顺序更新
- 3 设置最后一次DMA传输启动DMA完成中断
- DMA完成中断服务程序：设置数据有效标志位
- 主程序：根据数据有效标志启动算法处理数据

6678（8核）中断管理系统简介

- 中断的三级映射管理
- 4个全局中断控制器CIC0~3，分别负责将80个系统事件发送给核0~3，核4~7，EDMA 0~2，其中只有核中断涉及中断服务程序编写
- CIC将80个系统事件与76个通道进行关联
- 每个核中断控制器可以管理128个中断源，其中一部分固定，另有中断源来自76个通道
- 每个核的内核中断控制器管理16个中断，其中4个固定，12个可以由用户映射为128个中断源之一

作业-Verilog编程练习

- 设计任务1：设计电路将INT_request作为8086中断请求，并将中断类型号30H送到8086；
- 设计任务2：在initial中完成中断向量的写入操作；
- 观察ok_intnumber和ok_intvector是否最后保持高电平

作业2

- P271
- 8
- 10, 11

- 全部内容学习完毕