**把寻找操作数的方式叫做（操作数）寻址方式**

**立即数寻址方式**

**寄存器寻址方式**

**存储器寻址方式**

**1、直接寻址方式 2、寄存器间接寻址方式**

**3、寄存器相对寻址方式 4、基址变址寻址方式 5、相对基址变址寻址方式**

**微处理器的定义**

**微处理器**即中央处理单元，采用大规模集成电路技术制成的半导体芯片，内部集成了计算机的主要部件：控制器、运算器、寄存器组。微处理器通过执行指令序列完成指定的操作，处理器能够执行全部指令的集合就是该处理器的指令系统。

**微机的总线结构的好处，使用特点。包括总线定义，分类。**

总线定义：指传递信息的一组公用导线，

总线结构的好处：总线结构使得微机系统组态灵活，扩展方便。

使用特点：在某个时刻只有一个总线主控设备控制系统总线。

某一时刻只能有一个设备向总线发送信号，但可以有多个设备同时从总线上获取信号。

总线按传输信号可以分为

数据总线（用于CPU与其他部件之间传递信息，具有三态功能，且是双向的）、

地址总线（用于传递CPU要访问的存储单元或I/O接口的地址信号）、

控制总线（连接CPU的控制部件和内存、I/O设备等，用来控制内存和I/O设备的全部工作）

**冯⋅诺依曼存储程序工作原理**

1. 将采取二进制形式表示数据和指令。指令由操作码和地址码组成
2. 将程序和数据存放在存储器中，计算机在工作时从存储器取出指令加以执行，自动完成计算任务。
3. 指令的执行是顺序的，即一般按照指令在存储器中存放的顺序执行，程序分支由转移指令实现。
4. 计算机由存储器、运算器、控制器、输入设备和输出设备五大基本部件组成，并规定了各部件的基本功能。

**8086微处理器的构成、每一个T状态的主要工作。基本的存储读、写总线周期构成。常用的控制信号。**

**总线周期**是指CPU通过总线与外部（存储器或I/O端口）进行一次数据交换的过程，即完成一次总线操作的时间

**指令周期**是指一条指令经取指、译码、操作数读写直到指令完成所需要的时间。

**存储器读总线周期：**T1状态——输出存储器的地址

T2状态——输出控制信号-RD，选通存储器；DEN信号，选通数据收发器

T3和Tw状态——检测数据传送是否能够完成

T4状态——前沿读取数据，完成数据传送

**存储器写总线周期：**T1状态——输出20位存储器地址A19～ A0

T2状态—— -WR信号有效，-DEN信号有效以输出数据D7～D0

T3和Tw状态—— -WR、-DEN等控制信号持续有效，T3时钟下降沿检测READY信号，决定是否插入Tw；Tw期间，各信号延续状态。

T4状态——完成数据传送，并准备过渡到下一操作。-WR、-DEN转为无效。

**常用的控制信号：**

**ALE**地址锁存允许，输出、三态、高电平有效

**IO/-M：**I/O或存储器访问，输出、三态

**-WR：**写控制，输出、三态、低电平有效

**-RD：**读控制，输出、三态、低电平有效

INTR、-INTA等

**存储器地址的译码问题**

**全译码：**

所有的系统地址线均参与对存储单元的译码寻址

**特点：**采用全译码，每个存储单元的地址都是唯一的，不存在地址重复

译码电路可能比较复杂、连线也较多

**部分译码：**

只有部分（高位）地址线参与对存储芯片的译码

**特点：**每个存储单元将对应多个地址（地址重复），需要选取一个可用地址

可简化译码电路的设计、但系统的部分地址空间将被浪费

**存储芯片为什么要设置片选信号？它与系统地址总线有哪些连接方式？采用何种连接方式可避免地址重复？采用哪些连接方式可节省用于译码的硬件？**

解答：

片选信号说明该存储器芯片是否被选中正常工作，设置它可以比较方便地实现多个存储器芯片组成大容量的存储空间

存储器片选信号通常与CPU地址总线的高位地址线相关联，可以采用“全译码”、“部分译码”、“线选译码”方式

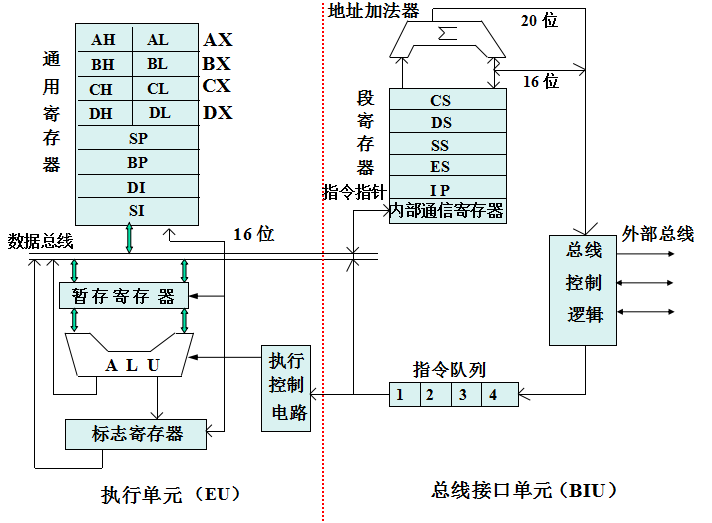
采用全译码方式可以避免地址重复

采用部分或线选译码可以节省译码硬件

**8086微处理器的内部结构，EU、BIU的定义和作用，流水线。**

BIU（总线接口单元）：由指令队列、指令指针、段寄存器、地址加法器和总线控制逻辑构成。该单元管理着8088与系统总线的接口、负责CPU对存储器和外设进行访问。

EU（执行单元）：由ALU、通用寄存器、标志寄存器和指令译码逻辑等构成，它负责指令的译码、执行和数据的运算。

两个单元相互独立，分别完成各自操作。两个单元可以并行执行，实现指令取指和执行的流水线操作

**8086的寻址方式。物理地址和逻辑地址的定义。两者之间转换。8086系统存储器采用分段管理方式。（要求很熟练）**

对于每个存储器单元都有一个唯一的20位地址，称为物理地址。

在8088的总线内部和用户编程时，所采用的“段地址：偏移地址”形式，称为逻辑地址。

一个存储器单元可以拥有多个逻辑地址，但可能拥有唯一的物理地址。

转换过程：先将段寄存器提供的16位段地址左移四位，低位补0，恢复为20位地址，然后与由各种寻址方式提供的16位偏移地址相加，即得到20位的物理地址。

**8086微处理器的内部构成。8086的寄存器结构，标志寄存器中每一个标志位的含义及应用。8086复位时各寄存器的初始状态。**

内部构成：算术逻辑单元（运算器）、寄存器组、指令处理单元（控制器）

寄存器结构：8086/8088共有8个的通用寄存器，1个标志寄存器，4个段寄存器和1个指令指针寄存器。

进位标志 CF：计算结果的最高有效位有进位，则CF=1，否则CF=0

溢出标志 OF：若算术运算的结果有溢出，则OF=1；否则 OF＝0

全零标志位 ZF：若运算结果为全0，则ZF=1，否则ZF=0

符号标志 CF：运算结果最高位为1，则SF=1；否则SF=0

奇偶标志位 PF：当运算结果最低字节中1的个数为零或偶数时，PF=1；否则PF=0（奇校验）

辅助进位标志 AF：运算时D3位（低半字节）有进位或借位时，AF=1；否则AF=0

方向标志 DF：存储地址自动增加，DF=1，否则DF=0

中断允许标志： IF＝1，则允许中断，IF＝0，则禁止中断

陷阱标志TF： TF＝0，处理器正常工作；TF＝1，处理器每执行一条指令就中断一次

**中断向量表。掌握中断向量表的构成，计算中断类型号，中断服务程序入口地址。中断的工作过程。**

**中断向量表**是一种表数据结构，是中断向量号与其对应的中断服务程序入口之间的链接表。该地址包括：偏移地址IP 、段地址CS （共32位）

每个中断向量的低字是偏移地址、高字是段地址，需占用4个字节（低对低，高对高）。

8088 微处理器从物理地址000H开始到3FFH（1KB），依次安排各个中断向量，向量号从0到255。256个中断向量所占用的1KB区域，称中断向量表

中断向量的存放首址=N\*4

**中断类型：**

**8088 CPU具有哪些中断类型？各种中断如何产生，如何得到中断向量号？**

**除法错中断**：在执行除法指令时，除数为0或商超过了寄存器所能表达的范围产生

**指令中断**：在执行中断指令INT n时产生

**溢出中断**：在执行溢出中断指令INTO时，溢出标志OF为1产生

**单步中断**：单步标志TF为1，在每条指令执行结束后产生

**非屏蔽中断**：外部通过NMI请求信号向微处理器提出请求时产生

**可屏蔽中断**：外部通过INTR请求信号向微处理器提出请求，并在允许可屏蔽中断的条件下产生

（2）除法错中断、溢出中断、单步中断、非屏蔽中断的向量号是8086微处理器内部已经确定

指令中断的操作数n就是向量号

可屏蔽中断的向量号在响应中断时通过数据总线从外部获得

**什么是8259A的中断结束字（EOI）？**

（1）IRi被响应时，ISR中对应的Di位被置1；

（2）中断处理完毕，相应的Di位应置0。

（3）向8259A送中断结束指令，使ISR的某位清0，指令的内容叫作中断结束字。

**8259A中IRR、IMR和ISR三个寄存器的含义**

中断请求寄存器IRR：

保存8条外界中断请求信号IR0～IR7的请求状态

Di位为1表示IRi引脚有中断请求；为0表示无请求

中断屏蔽寄存器IMR：

保存对中断请求信号IRi的屏蔽状态

Di位为1表示IRi中断被屏蔽（禁止）；为0表示允许

中断服务寄存器ISR：

保存正在被8259A处理的中断的状态

Di位为1表示IRi中断正在处理中；为0表示没有被处理

**中断的概念。****中断向量表的含义。深入理解8259的工作方式，优先权设置、中断结束处理、中断源屏蔽、中断触发等等。8259的初始化编程。中断服务程序编写。**

中断：指当出现需要时，CPU暂时停止当前程序的执行转而执行处理新情况的程序和执行过程。

优先权设置：在ISR的 Di 位置位期间，禁止再发生同级和低级优先权的中断，但允许高级优先权中断的嵌套

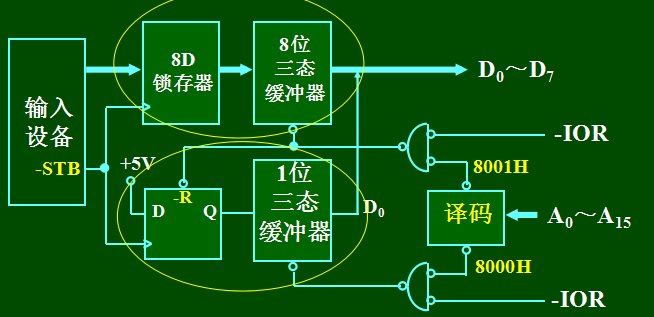
中断结束处理：自动中断结束方式、普通中断结束方式、特殊中断结束方式；

中断向量表的含义：256个中断向量所占用的1KB区域，称中断向量表

**计算机主机和I/O设备之间进行数据传送的方法。重点掌握查询方式完成数据传送的流程，要会编程。**

查询传送的特点是：工作可靠，适用面宽，但传送效率低

查询输入接口**（考电路）**：



代码如下：

mov dx,8000h ;dx指向状态端口

Status: in al, dx ；读状态端口

test al,01h ；测试标志位D0

jz status ；D0=0，未就绪，继续查询

inc dx ；D0=1，就绪，DX指向数据端口

in al,dx ；从数据端口输入数据

**中断传送：**

中断传送是一种效率更高的程序传送方式；

中断过程的完成由中断系统（硬件，如8259和CPU）和中断服务程序（软件）共同控制完成

**中断工作过程：**

1. 中断请求（外设） 2、中断响应（CPU）

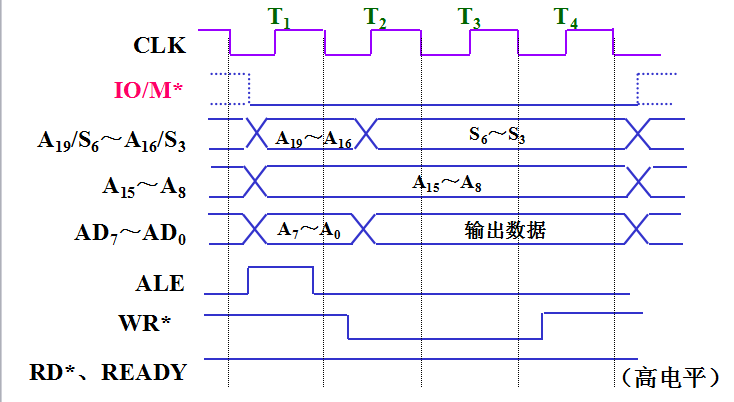
3、关中断（CPU） 4、断点保护（CPU）

5、中断识别（软件/硬件） 6、现场保护（用户）

7、中断服务（用户） 8、恢复现场（用户）

9、开中断（用户/CPU） 10、中断返回（IRET/用户）

最小组态的写总线周期时序：



**如何限制只能输入小写字母（ 61h~7Ah，*课本P22*），否则要求重新输入。**

.data

msg db 'not a lowercase,input again',

db 0dh, 0ah, '$'

input: mov ah,1 ;输入小写字母

int 21h

cmp al, 61h

jl once-again

cmp al, 7Ah

ja once-again

jmp convert

once-again: mov dx,offset msg

mov ah,9

int 21h

jmp input

convert: sub al,20h ;转换为大写字母

……

**采用查表法，实现一位16进制数转换为ASCII码显示**

04h——34h（4） 0bh——42h（B）

ASCII db 30h,31h,32h,33h,34h,35h,36h,37h,38h,39h

；对应0 ~ 9的ASCII码

db 41h,42h,43h,44h,45h,46h

；对应A ~ F的ASCII码

hex db 04h,0bh

；假设两个16进制数

;代码段

mov bx,offset ASCII

；BX指向ASCII码表

mov al, hex

；AL取得一位16进制数；恰好就是ASCII码表中的位移

and al, 0fh

；只有低4位是有效的，高4位清0

xlat ；换码：AL←DS:[BX＋AL]

mov dl,al ；欲显示的ASCII码送DL

mov ah,2 ；2号DOS功能调用

int 21h ；显示一个ASCII码字符

mov al,hex+1 ；转换并显示下一个数据

and al,0fh

xlat

mov dl,al

mov ah,2

int 21h

**编写一个源程序，在键盘上按一个键，将从AL返回的ASCII码值显示出来，如果按下ESC键则程序退出。(可调用书中的HTOASC子程序)**

HTOASC proc

and al,0fh ；al低四位保存待转 ；换的16进制数

cmp al,9

jbe htoasc1

add al,37h ;是A ~ F，加37H

ret ;子程序返回

htoasc1: add al,30h ;0 ~ 9，加30H

ret ;子程序返回

HTOASC endp

**push ax**

mov cl, 4

ror al, cl

call HTOASC

call disp\_a\_char

pop ax

call HTOASC

call disp\_a\_char

**把从键盘输入的一个小写字母用大写字母显示出来(大小写字母转换)。**

mov ah,1 ；输入小写字母

int 21h

sub al,20h ；转换为大写字母

mov dl,al

mov ah,2

int 21h ；显示

**写一个子程序，根据入口参数AL＝0、1、2，依次实现对大写字母转换成小写、小写转换成大写或大小写字母互换。欲转换的字符串在string中，用0表示结束。**

lucase proc

push bx

mov bx,offset string

cmp al,0

je case0 cmp al,1 jz case1 cmp al,2 jz case2 jmp done

case0: cmp byte ptr [bx],0 je done

cmp byte ptr [bx],’A’ jb next0

cmp byte ptr [bx],’Z’ ja next0

add byte ptr [bx],20h

next0: inc bx

jmp case0

case1: cmp byte ptr [bx],0

je done

cmp byte ptr [bx],’a’

jb next1

cmp byte ptr [bx],’z’

ja next1

sub byte ptr [bx],20h

next1: inc bx

jmp case1

case2: cmp byte ptr [bx],0 je done

cmp byte ptr [bx],’A’ jb next2

cmp byte ptr [bx],’Z’ ja next20

add byte ptr [bx],20h

jmp next2

next20: cmp byte ptr [bx],’a’

jb next2

cmp byte ptr [bx],’z’

ja next2

sub byte ptr [bx],20h

next2: inc bx

jmp case2

done: pop bx

ret

lucase endp

**循环累加（调用子程序）**

array db 12h,25h,0f0h,0a3h,3,68h,71h,0cah,0ffh,90h

count equ $-array ；数组元素个数

result db ? ；校验和；代码段（主程序）

mov bx,offset array ；BX←数组的偏移地址

mov cx,count ；CX←数组的元素个数

call checksum ；调用求和过程

mov result,al ；处理出口参数

checksum proc

xor al,al ；累加器清0

sum: add al,[bx] ；求和

inc bx ；指向下一个字节

loop sum Ret

Checksum endp

**计算AX中有符号数的绝对值**

cmp ax,0

jge nonneg ；条件满足（AX≥0），转移

neg ax ；条件不满足，求补（即绝对值P43）

nonneg: mov result,ax ；保存结果；不恰当的分支

cmp ax,0

jl yesneg ；条件满足（AX＜0），转移

jmp nonneg

yesneg: neg ax ；条件满足，求补

nonneg: mov result,ax ；保存结果

**设置两个变量maxay和minay存放最大和最小值**

array dw 10

dw -3,0,20,900,587,-632,777,234,-34,-56

maxay dw ? ；存放最大值

minay dw ? ；存放最小值

lea si,array

mov cx,[si] ；取得元素个数

dec cx ；减1后是循环次数

add si,2 mov ax,[si] mov bx,ax

maxck: add si,2

cmp [si],ax ；与下一个数据比较

jle minck

mov ax,[si] ；AX取得更大的数据

jmp next

minck: cmp [si],bx

jge next

mov bx,[si] ；BX取得更小的数据

next: loop maxck ；计数循环

mov maxay,ax ；保存最大值

mov minay,bx ；保存最小值

**挑出数组中正数（不含0）和负数，分别形成正、负数组**

DATAS SEGMENT

count equ 10

array dw 23h,9801h… ayplus dw count dup(0) ayminus dw count dup(0)

DATAS ENDS

STACKS SEGMENT

STACKS ENDS

CODES SEGMENT

ASSUME CS:CODES,DS:DATAS,SS:STACKS

START:

MOV AX,DATAS

MOV DS,AX

mov si,offset array

mov di,offset ayplus

mov bx,offset ayminus

mov ax,ds

mov es,ax

mov cx,count

cld

again: lodsw

cmp ax,0

jl minus

jz next

minus: xchg bx,di

stosw

xchg bx,di

next: loop again

MOV AH,4CH

INT 21H

CODES ENDS

END START

**8259A 的初始化程序段（7.8）**

ｍｏｖ　ａｌ，１３Ｈ ；写入ICW1：设定边沿触发方式，单片方式

　　　　ｍｏｖ　ｄｘ，０ＦＦＤＣＨ

　　　　ｏｕｔ　ｄｘ，ａｌ

　　　　ｊｍｐ　ｉｎｔｒ１

ｉｎｔｒ１：ｍｏｖ　ａｌ，９０Ｈ ；写入ICW2：设定IR0的中断向量号为90h

　　　　　　ｍｏｖ　ｄｘ，０ＦＦＤＥＨ

　　　　　　ｏｕｔ　ｄｘ，ａｌ

　　　　　　ｊｍｐ　ｉｎｔｒ２

ｉｎｔｒ２：ｍｏｖ　ａｌ，１ ；写入ICW4：设定普通嵌套方式，普通中断方式

　　　　　　ｍｏｖ　ｄｘ，０ＦＦＤＥＨ

　　　　　　ｏｕｔ　ｄｘ，ａｌ

**8259的初始化编程：**

**主片：**

mov al,11h ;写入ICW1

out 20h,al

intr1: mov al,08h ;写入ICW2

out 21h,al

intr2: mov al,04h ;写入ICW3

out 21h,al

intr3: mov al,05h ;写入ICW4

out 21h,al

**从片：**

mov al,11h ;写入ICW1

out 0a0h,al

intr5: mov al,70h ;写入ICW2

out 0a1h,al

intr6: mov al,02h ;写入ICW3

out 0a1h,al

intr7: mov al, 01h ;写入ICW4

out 0a1h,al

**试按如下要求分别编写8253的初始化程序，已知8253的计数器0～2和控制字I/O地址依次为204H～207H。⑶ 使计数器2工作在方式2，计数值为02F0H。**

mov al,0b4h

mov dx,207h

out dx,al

mov ax,02f0h

mov dx,206h

out dx,al ;先写入低字节

mov al,ah

out dx,al ;后写入高字节

**8255的初始化编程**

**要求：A口：方式1输入、CH口：输出、CL口：输入、B口：方式0输出**

**方式控制字：1 0110 001 B或B1H**

初始化的程序段：

mov dx,0fffeh ;假设控制端口为FFFEH

mov al,0b1h ;方式控制字

out dx,al ;送到控制端口

某系统中8253芯片中计数器的0-2和控制字端口地址分别是FFF0H~FFF3H。

编写8253两个计数器**初始化程序及两个计数器的连通图。**

**计数器0的计数值：**5M/1K＝5000＝1388H

方式控制字：00100101＝25H、2DH、35H、3DH （十进制计数）

00100100＝24H、2CH、34H、3CH （二进制计数）

**计数器1的计数值：**1000

方式控制字：01101001＝69H、79H

（十进制计数）

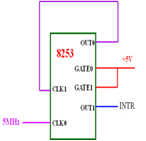
01101000＝68H、78H

（二进制计数）

MOV DX,0FFF3H

MOV AL,25H ;通道0，只写高字节，方式2，十进制

OUT DX,AL

MOV DX, 0FFF0H

MOV AL,50H ；计数初值5000

OUT DX,AL

MOV DX,0FFF3H

MOV AL,69H ；通道1，方式4

OUT DX,AL

MOV DX, 0FFF1H

MOV AL,10H ；计数初值1000

OUT DX,AL

**3、某字符输出设备（如打印机），其数据口和状态口的地址均为80H。在读取状态时，当标志位D7＝0时，表明该设备闲，可以接收一个字符，请编写利用查询方式进行数据传送的程序段。要求将存放于符号地址addr处的一串字符（以$为结束标志）输出给该设备。**

mov bx, offset addr ;利用offset操作符，可在汇编过程中得到addr的偏移地址

again: mov ah, [bx]

cmp ah, ’$’

jz done

status: in al, 80h ;查询一次

test al, 80h ;两个80h不一样

jnz status ;D7=1,表示设备忙，继续查询

mov al, ah

out 80h, al ;输出一个字节

inc bx

jmp again ;循环

done: ……

**如图10.23为用一片8255A控制8个8段共阴极LED数码管的电路。现要求按下某个开关，其代表的数字（K1为1，K2为2，…K8为8）在数码管从左到右循环显示（已有一个延时子程序delay可以调用），直到按下另一个开关。假定8255A的数据端口A、B、C及控制端口的地址依次为FFF8H～FFFBH。编写完成上述功能的程序，应包括8255A的初始化、控制程序和数码管的显示代码表**

；显示代码表

table db 0c0h ；对应0（任意）

db 0f9h,0a4h,0b0h,99h，92h,82h,0f8h,80h ；对应1～8

；8255A初始化

mov dx,0fffbh

mov al,10001001b ；＝89h

out dx,al ；控制程序

again0: mov dx,0fffah ；输入开关状态

in al,dx

mov cx,8 ；确定哪个开关闭合

mov ah,01h ；mov ah,08h

again1: shr al,1 ；shl al,1

jnc disp0

inc ah ；dec ah

loop again1

jmp disp1 ；显示字段

disp0: mov bx,offset table

mov al,ah

xlat

mov dx,0fff8h

out dx,al ；输出段码

disp1: mov cx,8 ；循环显示8位

mov al,01h

mov dx,0fff9h

disp2: out dx,al ；输出位码

call delay

shl al,1

loop disp2

jmp again0 **；写入方式字**

mov al,100×00×1b ；＝81h

mov dx,控制口地址 ；0fffeh

out dx,al ；加入下一段更好，使L0～L3全亮

mov al,0fh

mov dx,端口C地址 ；0fffch

out dx,al ；控制程序段

mov dx,端口C地址 ；0fffch

in al,dx ；读入PC0～PC3

mov cl,4 shl al,cl ；左移4位

out dx,al ；控制PC4～PC7

**以8255的B口作输入口、连接逻辑电平开关；以8255的A口作输出口、连接发光二极管电路。要求：当输入不是全0时，输出与输入保持一致（即高电平时点亮）；当输入是全0时，令发光二极管循环点亮（或闪烁告警）。**

begin: mov dx,portc ; 8255 初始化

mov al,82h ； A组方式0，A口出，

； B组方式0，B口入

out dx,al

mov ah,0ffh

mov bl,0

lp: mov dx,portb ; B 口读入开关状态

in al,dx

test al,ah ；与ffh比较，看是否8个开关全闭合（0信号）

jz shift

mov dx,porta ; A 口输出开关状态

not al

out dx,al

jmp lp

shift: mov al,bl ；bl初值为0

mov dx,porta

not al

out dx,al

call delay ; 延时

shl bl,1 ; 移位

test bl,ah ；ah初值为ffh

jnz lp

mov bl,1

jmp lp

**2、如果有一个输入设备，其数据口地址为FFE0H，状态口地址为FFE2H，当状态标志D0＝1时，表明一个字节的输入数据就绪，请编写利用查询方式进行数据传送的程序段。要求从该设备读取100个字节并写到从2000H:2000H开始的内存中。**

通过读取状态寄存器的标志位来检查外设是否就绪。若不就绪就继续查询，即程序不断循环；直至就绪。然后，进行下一步的传送工作。

mov bx, 2000h

mov ds, bx

mov cx, 100

again: mov dx, 0ffe2h

status: in al, dx ;读入状态标志D0

test al, 01h ;al & 01h

jz status

mov dx, 0ffe0h

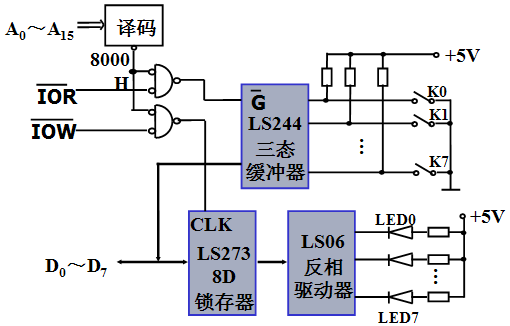
in al, dx ;输入一个字节

mov [bx], al ;al→ds:[bx], 寄存器间接寻址

inc bx

**查询方式完成数据传送。**

**1、编程实现以下功能：当K0键单独按下时，发光二极管L0～L7将流水点亮（Ln→Ln+1），每个维持200ms；当K1键单独按下时，发光二极管L0～L7将反向流水点亮（Ln←Ln+1），每个也维持200ms；在其他情况下各发光二极管均不点亮。假定有延时200ms的子程序DELAY可直接调用。**



again: mov dx, 8000h ;8000h是I/O端口地址

in al, dx ;读入端口数据

not al ;开关闭合，读入数据是0；反相，保证如图二极管点亮；

cmp al, 1 ；K7~K0＝11111110B ？或al=00000001B?

je l1 ;单独按下K0，转移到l1

cmp al, 2

je l2 ;单独按下K1，转移到l2

jmp again ;其它情况，都不亮

l1: mov cx, 8 ;计数器设为8

mov al, 1

l11: out dx, al ;点亮一个LED

call delay ;调用延时子程序，点亮状态保持200ms

rol al, 1 ;循环左移（shl行吗？）

loop l11 ;循环直到cx减到0为止

jmp again ;转到起点，继续查询端口变动

l2: mov cx, 8 ;计数器设为8

mov al, 1

l21: out dx, al ;点亮一个LED

call delay ;调用延时子程序，点亮状态保持200ms

ror al, 1

loop l21 ;

jmp again

delay proc

mov cx,0bfffh ；延时200ms。0bffh=3071（D），

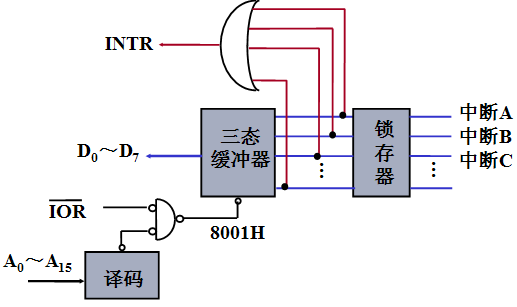
loop $ ;延时时间=49151\*33 (时钟周期)/8000000(CPU工作频率)≈200ms

delay endp

loop again ;循环，输入100个字节

**4、按照图6-14所示的中断查询接口与相应的流程图，请编写用于中断服务的程序段。具体要求是，当程序查到中断设备A有中断请求（对应数据线D0），它将调用名为PROC0的子程序；如此，依次去查中断设备B～中断设备D，并分别调用名为PROC1～PROC3的子程序。**

sti ;开中断

 push ax

push dx

……

mov dx,8001h ;接口地址是8001h

status: in al, dx

test al, 01h

jnz service0

test al, 02h

jnz service1

test al, 04h

jnz service2

test al, 08h

jnz service3

……

service0: call proc0

jmp done

service1: call proc1

jmp done

service2: call proc2

jmp done

service3: call proc3

jmp done

……

done: pop dx

pop ax

iret ;中断返回