第1章　微型计算机系统概述

〔习题 1.2〕

什么是通用微处理器、单片机（微控制器）、DSP 芯片、嵌入式系统？

〔解答〕

通用微处理器：适合较广的应用领域的微处理器，例如装在 PC 机、笔记本电脑、工作站、服务器上的微处理器。

单片机：是指通常用于控制领域的微处理器芯片，其内部除 CPU 外还集成了计算机的其他一些主要部件，只需配上少量的外部电路和设备，就可以构成具体的应用系统。

DSP 芯片：称数字信号处理器，也是一种微控制器，其更适合处理高速的数字信号，内部集成有高速乘法器，能够进行快速乘法和加法运算。

嵌入式系统：利用微控制器、数字信号处理器或通用微处理器，结合具体应用构成的控制系统，其典型的特点是把计算机直接嵌入到应用系统之中。

〔习题 1.5〕

说明微型计算机系统的硬件组成及各部分作用。

〔解答〕

CPU：CPU 也称处理器，是微机的核心。它采用大规模集成电路芯片，芯片内集成了控制器、运算器和若干高速存储单元（即寄存器）。处理器及其支持电路构成了微机系统的控制中心，对系统的各个部件进行统一的协调和控制。

存储器：存储器是存放程序和数据的部件。

外部设备：外部设备是指可与微机进行交互的输入（Input）设备和输出（Output）设备，也称 I/O 设备。I/O 设备通过 I/O 接口与主机连接。

总线：互连各个部件的共用通道，主要含数据总线、地址总线和控制总线信号。

〔习题 1.6〕

什么是总线？微机总线通常有哪 3 组信号？各组信号的作用是什么？

〔解答〕

总线：传递信息的共用通道，物理上是一组公用导线。

3 组信号线：数据总线、地址总线和控制总线。

（1）地址总线：传输将要访问的主存单元或 I/O 端口的地址信息。

（2）数据总线：传输读写操作的数据信息。

（3）控制总线：协调系统中各部件的操作。

〔习题 1.7〕

简答如下概念：

（1）计算机字长

（2）取指－译码－执行周期

（3）ROM-BIOS

（4）中断

（5）ISA 总线

〔解答〕

（1）处理器每个单位时间可以处理的二进制数据位数称计算机字长。

（2）指令的处理过程，即指处理器从主存储器读取指令（简称取指），翻译指令代码的功能（简称译码），然后执行指令所规定的操作（简称执行）的过程。

（3）ROM-BIOS 是“基本输入输出系统”，操作系统通过对 BIOS 的调用驱动各硬件设备，用户也可以在应用程序中调用 BIOS 中的许多功能。

（4）中断是 CPU 正常执行程序的流程被某种原因打断、并暂时停止，转向执行事先安排好的一段处理程序，待该处理程序结束后仍返回被中断的指令继续执行的过程。

（5）ISA 总线是指 IBM PC/AT 机上使用的系统总线。

〔习题 1.8〕

下列十六进制数表示无符号整数，请转换为十进制形式的真值：

（1）FFH　　（2）0H　　（3）5EH　　（4）EFH

〔解答〕

（1） 255

（2） 0

（3） 94

（4） 239

〔习题 1.9〕

将下列十进制数真值转换为压缩 BCD 码：

（1）12　　（2）24　　（3）68　　（4）99

〔解答〕

（1） 12H

（2） 24H

（3） 68H

（4） 99H

〔习题 1.10〕

将下列压缩 BCD 码转换为十进制数：

（1）10010001　　（2）10001001　　（3）00110110　　（4）10010000

〔解答〕

（1） 91

（2） 89

（3） 36

（4） 90

〔习题 1.11〕

将下列十进制数用 8 位二进制补码表示：

（1）0　　（2）127　　（3）－127　　（4）－57

〔解答〕

（1） 00000000

（2） 01111111

（3） 10000001

（4） 11000111

〔习题 1.12〕

数码 0～9、大写字母 A～Z、小写字母 a～z 对应的 ASCII 码分别是多少？ASCII 码 0DH 和 0AH 分别对应什么字符？

〔解答〕

数码 0～9 对应的 ASCII 码依次是 30H～39H。

大写字母 A～Z 对应的 ASCII 码依次是：41H～5AH。

小写字母 a～z 对应的 ASCII 码依次是：61～7AH。

ASCII 码 0DH 和 0AH 分别对应的是回车和换行字符。

第 2 章 微处理器指令系统

〔习题 2.1〕

微处理器内部具有哪 3 个基本部分？8088 分为哪两大功能部件？其各自的主要功能是什么？这种结构与 8 位 CPU 相比为什么能提高其性能？

〔解答〕

算术逻辑单元 ALU、寄存器组和控制器；

总线接口单元 BIU：管理 8088 与系统总线的接口负责 cpu 对接口和外设进行访问

执行单元 EU：负责指令译码、执行和数据运算；

8 位 cpu 在指令译码前必须等待取指令操作的完成，8088 中需要译码的指令已经取到了指令队列，不需要等待取指令。而取指令是 cpu 最为频繁的操作，因此 8088 的结构和操作方式节省了大量等待时间，比 8 位 cpu 节省了时间，提高了性能。

〔习题 2.2〕

说明 8088 的 8 个 8 位和 8 个 16 位通用寄存器各是什么？

〔解答〕

8 个 8 位寄存器：AH、AL、BH、BL、CH、CL、DH、DL；

8 个 16 位寄存器：累加器 AX、基址寄存器 BX、计数器 CX、数据寄存器 DX、源地址寄存器 SI、目的地址寄存器 DI、基址指针 BP、堆栈指针 SP。

〔习题 2.3〕

标志用于反映指令执行结果或者控制指令执行形式。

状态标志用于记录程序运行结果的状态信息；控制标志用于控制指令执行的形式。

〔习题 2.4〕

举例说明 CF 和 OF 标志的差异。

〔解答〕

例：有运算：3AH+7CH=B6H

作为无符号数运算，没有进位，CF=0;

作为有符号数运算，结果超出范围，OF=1.

〔习题 2.5〕

什么是 8088 中的逻辑地址和物理地址？逻辑地址如何转换成物理地址？1MB 最多能分成多少个逻辑段？请将如下逻辑地址用物理地址表达：

⑴ FFFFH:0 ⑵ 40H:17H ⑶ 2000H:4500H ⑷ B821H:4567H

〔解答〕

物理地址：物理存储单元具有的一个唯一的 20 位编号

逻辑地址：在 8088 内部和用户编程时，所采用的“段地址：偏移地址”形式

将逻辑地址中的段地址左移二进制 4 位（对应 16 进制是一位，即乘以 16），加上偏移地址就得到 20 位物理地址

1MB 最多能分成 1MB÷16B＝$2^{20}\div 2^{4}=2^{16}$ 个逻辑段，因为每隔 16 个字节单元就可以开始一个逻辑段

⑴ FFFFH:0＝FFFF0H

⑵ 40H:17H＝00417H

⑶ 2000H:4500H＝24500H

⑷ B821H:4567H＝BC777H

〔习题 2.7〕

代码段：存放程序的指令序列；

堆栈段：确定堆栈所在的主存储区；

数据段：存放当前运行程序的数据；

附加段：附加数据段，用于数据保存。另外串操作指令将其作为目的操作数的存放区。

〔习题 2.8〕

已知 DS＝2000H、BX＝0100H、SI＝0002H，存储单元[20100H]～[20103H]依次存放 12H、34H、56H、78H，[21200H]～[21203H]依次存放 2AH、4CH、B7H、65H，说明下列每条指令执行完后 AX 寄存器的内容，以及源操作数的寻址方式？

⑴ mov ax,1200h

⑵ mov ax,bx

⑶ mov ax,[1200h]

⑷ mov ax,[bx]

⑸ mov ax,[bx+1100h]

⑹ mov ax,[bx+si]

⑺ mov ax,[bx][si+1100h]

〔解答〕

⑴ mov ax,1200h ； AX＝1200H，立即数寻址

⑵ mov ax,bx ； AX＝0100H，寄存器寻址

⑶ mov ax,[1200h]； AX＝4C2AH，直接寻址

⑷ mov ax,[bx] ； AX＝3412H，寄存器间接寻址

⑸ mov ax,[bx+1100h] ； AX＝4C2AH，寄存器相对寻址

⑹ mov ax,[bx+si] ； AX＝7856H，基址变址寻址

⑺ mov ax,[bx][si+1100h] ； AX＝65B7H，相对基址变址寻址

〔习题 2.9〕

说明下面各条指令的具体错误原因

⑴ mov cx,dl ⑵ mov ip,ax

⑶ mov es,1234h ⑷ mov es,ds

⑸ mov al,300 ⑹ mov [sp],ax

⑺ mov ax,bx+di ⑻ mov 20h,ah

〔解答〕

2.9 (1) CX 16 位 DL 8 位 操作数宽度不一样（类型不一致）

(2) IP 不能作为源操作数，也不能作为目的操作数

(3) 立即数不能直接赋给段寄存器

(4) 段寄存器间不能直接赋值

(5) 类型不一致，立即数 300 超过 8 位，不能赋给 8 位寄存器 AL

(6) 寄存器间接寻址方式不能使用 sp 寄存器

(7)+作为算术运算符，它的操作数为常量

(8)20h 为立即数不能作为目的操作数

〔习题 2.10〕

已知数字 0～9 对应的格雷码依次为：18H、34H、05H、06H、09H、0AH、0CH、11H、12H、14H，它存在于以 table 为首地址（设为 200H）的连续区域中。请为如下程序段的每条指令加上注释，说明每条指令的功能和执行结果。

```
lea bx,table
mov al,8
xlat
```

〔解答〕

```
lea bx,table          ；获取 table 的首地址，BX＝200H
mov al,8              ；传送欲转换的数字，AL＝8
xlat                  ；转换为格雷码，AL＝12H
```

〔习题 2.11〕

给出下列各条指令执行后 AL 值，以及 CF、ZF、SF、OF 和 PF 的状态：

```
mov al,89h
```

```
add al,al
add al,9dh
cmp al,0bch
sub al,al
dec al
inc al
```

〔解答〕

```
mov al,89h          ; AL＝89H      CF  ZF  SF  OF  PF
add al,al           ; AL＝12H      1   0   0   1   1
add al,9dh          ; AL＝0AFH     0   0   1   0   1
cmp al,0bch         ; AL＝0AFH     1   0   1   0   1
sub al,al           ; AL＝00H      0   1   0   0   1
dec al              ; AL＝0FFH     0   0   1   0   1
inc al              ; AL＝00H      0   1   0   0   1
```

〔习题 2.12〕

请分别用一条汇编语言指令完成如下功能：

⑴ 把 BX 寄存器和 DX 寄存器的内容相加，结果存入 DX 寄存器。

⑵ 用寄存器 BX 和 SI 的基址变址寻址方式把存储器的一个字节与 AL 寄存器的内容相加，并把结果送到 AL 中。

⑶ 用 BX 和位移量 0B2H 的寄存器相对寻址方式把存储器中的一个字和 CX 寄存器的内容相加，并把结果送回存储器中。

⑷ 用位移量为 0520H 的直接寻址方式把存储器中的一个字与数 3412H 相加，并把结果送回该存储单元中。

⑸ 把数 0A0H 与 AL 寄存器的内容相加，并把结果送回 AL 中

〔解答〕

⑴ add dx,bx

⑵ ADD AL,BYTE PTR [BX+SI] 或 ADD AL,BYTE PTR [BX][SI]

⑶ add [bx+0b2h],cx

⑷ add word ptr [0520h],3412h

⑸ add al,0a0h

〔习题 2.13〕

设有 4 个 16 位带符号数，分别装在 X、Y、Z、V 存储单元中，阅读如下程序段，得出它的运算公式，并说明运算结果存于何处。

```
mov ax,X
imul Y
mov cx,ax
mox bx,dx
mov ax,Z
```

```
cwd
add cx,ax
adc bx,dx
sub cx,540
sbb bx,0
mov ax,V
cwd
sub ax,cx
sbb dx,bx
idiv X
```

〔解答〕

（V-（X*Y+Z-540）/X；商存储在 ax，余数存储在 dx。

〔习题 2.14〕

给出下列各条指令执行后的结果，以及状态标志 CF、OF、SF、ZF、PF 的状态。

```
mov ax,1470h
and ax,ax
or ax,ax
xor ax,ax
not ax
test ax,0f0f0h
```

〔解答〕

mov ax,1470h	；AX＝1470H	CF	ZF	SF	OF	PF
and ax,ax	；AX＝1470H	0	0	0	0	0
or ax,ax	；AX＝1470H	0	0	0	0	0
xor ax,ax	；AX＝0000H	0	1	0	0	1
not ax	；AX＝FFFFH	0	1	0	0	1
test ax,0f0f0h	；AX＝FFFFH	0	0	1	0	1

〔习题 2.15〕

控制转移类指令中有哪三种寻址方式?

〔解答〕

相对寻址、直接寻址、间接寻址（还可以分成寄存器间接、存储器间接）

〔习题 2.16〕

假设 DS＝2000H、BX＝1256H、TABLE 的偏移地址是 20A1H，物理地址 232F7H 处存放 3280H，试问执行下列段内间接寻址的转移指令后，转移的有效地址是什么？

⑴ JMP BX

⑵ JMP TABLE[BX]

〔解答〕

（1）1256H

（2）3280H

〔习题 2.17〕

判断下列程序段跳转的条件

⑴ xor ax,1e1eh
je equal

⑵ test al,10000001b
jnz there

⑶ cmp cx,64h
jb there

〔解答〕

⑴ AX＝1e1eh（异或后为 0）

⑵ AL 的 D_0 或 D_7 至少有一位为 1

⑶ CX（无符号数）＜ 64h

〔习题 2.18〕

如下是一段软件延时程序，请问 NOP 指令执行了多少次？

```
        xor cx,cx
delay:  nop
        loop delay
```

〔解答〕

2^{16}=65536 次

〔习题 2.19〕

功能：将数组中每个字元素相加，结果存储于 total。

〔习题 2.20〕

按照下列要求，编写相应的程序段：

⑴ 由 string 指示起始地址的主存单元中存放有一个字符串（长度大于 6），把该字符串中的第 1 个和第 6 个字符（字节量）传送给 DX 寄存器。

⑵ 有两个 32 位数值，按“小端方式”存放在两个缓冲区 buffer1 和 buffer2 中，编写程序段完成 DX.AX←buffer1－buffer2 功能。

⑶ 编写一个程序段，在 DX 高 4 位全为 0 时，使 AX＝0；否则使 AX＝－1。

⑷ 把 DX.AX 中的双字右移 4 位

⑸ 有一个 100 个字节元素的数组，其首地址为 array，将每个元素减 1（不考虑溢出或借位）存于原处。

〔解答〕（1）

```
mov dl,string[1]
mov dh,string[6]
```

（2）

```
mov ax, word ptr buffer1
```

```
        sub ax, word ptr buffer2
        ; 先减低 16 位
        mov dx, word ptr buffer1+2
        sbb dx, word ptr buffer2+2
        ; 后减高 16 位，需减低 16 位的借位
（3）   test dx,0f000h
        jz next
        mov ax,-1
        jmp done
next:   mov ax,0
done:   …
（4）   mov cx,4
again:  shr dx,1  ; 右移一位，移出的低位进入 CF 标志
        rcr ax,1  ; 将 CF 移进 AX 高位，同时实现 AX 右移
        loop again    ; 循环 4 次，实现 4 位右移
（5）   mov cx,100
        mov bx,0
again:  sub array[bx],1
        inc bx
        loop again
```

〔习题 2.21〕

```
    Htoasc  proc
            Mov bl,al
            Mov al,ah
            Mov bh,10
            Mul bh
            And ax,00FFH
            Add al,bl
    Htoend: ret
Htoasc  endp
```

〔习题 2.22〕

①计算机系统利用中断为用户提供硬件设备驱动程序。在 IBM-PC 系列微机中，基本输入输出系统 ROM-BIOS 和 DOS 都提供了丰富的中断服务程序，称为系统功能调用。

②调用步骤（1）AH 中设置系统功能调用号

（2）在指定寄存器中设置入口参数

（3）使用中断调用指令执行功能调用

（4）根据出口参数分析调用情况

〔习题 2.23〕

```
Htoasc    proc
          And al,0FH
          Add al,90H
          Daa
          Adc al,40H
          Daa
          Mov ah,02H
          Mov dl,al
          Int 21H
          Ret
Htoasc    endp
```

〔习题 2.24〕

```
Numout    proc
          Xor ah,ah
          Aam
          Add ax,3030H
          Mov dl,ah
          Mov ah,02H
          Int 21H
          Mov dl,al
          Mov ah,02H
          Int 31H
          Ret
Numout    endp
```

〔习题 2.25〕

```
Msgkey    db”input number 0-9”,”$”
Msgwrg    db”error”,”$”
          Mov ah,09H
          Mov dx,offset msgkey
          Int 21H
Again:    mov ah,01H
          Int 21H
          Cmp al,30H
          Jb disp
          Cmp al,39H
```

```
        Ja disp
        Mov dl,al
        Mov ah,02H
        Int 21H
        Jmp done
Disp:   mov dx,offset msgwrg
        Mov ah,09H
        Int 21H
        Jmp again
Done:   mov ah,4cH
        Int 21H
```

第 3 章 汇编语言程序设计

3.1 解：汇编语言是一种以处理器指令系统为基础的低级程序设计语言，它采用助记符表达指令操作码，采用标识符号表示指令操作数，可以直接、有效地控制计算机硬件，因而容易创建代码序列短小、运行快速的可执行程序

3.2 解：（1）完整的汇编语言源程序由段组成

（2）一个汇编语言源程序可以包含若干个代码段、数据段、附加段或堆栈段，段与段之间的顺序可随意排列

（3）需独立运行的程序必须包含一个代码段，并指示程序执行的起始点，一个程序只有一个起始点

（4）所有的可执行性语句必须位于某一个代码段内，说明性语句可根据需要位于任一段内

（5）通常，程序还需要一个堆栈段

3.3 解：

存储模式	特 点
TINY	COM 类型程序，只有一个小于 64KB 的逻辑段（MASM 6.x 支持）
SMALL	小应用程序，只有一个代码段和一个数据段（含堆栈段），每段不大于 64KB
COMPACT	代码少、数据多的程序，只有一个代码段，但有多个数据段
MEDIUM	代码多、数据少的程序，可有多个代码段，只有一个数据段
LARGE	大应用程序，可有多个代码段和多个数据段（静态数据小于 64KB）
HUGE	更大应用程序，可有多个代码段和多个数据段（对静态数据没有限制）

FLAT	32 位应用程序，运行在 32 位 80x86CPU 和 Windows 9x 或 NT 环境

3.4 解：

开始位置：用标号指明

返回 DOS：利用 DOS 功能调用的 4CH 子功能来实现

汇编停止：执行到一条 END 伪指令时，停止汇编

3.5 解：

段定位、段组合和段类型。

3.6 解：

```
stack  segment  stack
        db       1024(0)
stack  ends
data   segment
string  db   'Hello,Assembly!', 0dH, 0aH,  '$'
data   ends
code   segment  'code'
         assume   cs:code,ds:data,ss:stack
start:  mov  dx,offset string
        mov  ah,9
        int   21h
code  ends
         end  start
```

3.7 解：

(1). EXE 程序

程序可以有多个代码段和多个数据段，程序长度可以超过 64KB

通常生成 EXE 结构的可执行程序

(2). COM 程序

只有一个逻辑段，程序长度不超过 64KB

需要满足一定条件才能生成 COM 结构的可执行程序（MASM 6.x 需要采用 TINY 模式）

3.8 解：

符号定义伪指令有“等价 EQU”和“等号＝”：

符号名 EQU 数值表达式

符号名 EQU〈字符串〉

符号名 ＝ 数值表达式

EQU 用于数值等价时不能重复定义符号名，但“＝”允许有重复赋值。例如：

```
X= 7        ; 等效于：X equ 7
X= X+5   ; “X EQU X+5”是错误的
```

〔习题 3.9〕

给出下列语句中，指令立即数（数值表达式）的值：

⑴ mov al,23h AND 45h OR 67h

⑵ mov ax,1234h/16+10h

⑶ mov ax,23h SHL 4

⑷ mov al,’a’ AND (NOT(’a’-’A’))

⑸ mov ax,(76543 LT 32768) XOR 7654h

〔解答〕

（1） al=67h

（2） ax=133h,dx=4h

（3） ax=0230h

（4） al=41h

（5） ax=7654h

〔习题 3.10〕

画图说明下列语句分配的存储空间及初始化的数据值：

⑴ byte_var db ’ABC’,10,10h,’EF’,3 dup(-1,?,3 dup(4))

⑵ word_var dw 10h,-5,3 dup(?)

〔解答〕

（1）从低地址到高地址，各个字节依次是：

41h 42h 43h 0ah 10h 45h 46h ffh - 04h 04h 04h ffh - 04h 04h 04h ffh - 04h 04h 04h

（2）从低地址到高地址，各个字节依次是：

10h 0 FBh FFh - - - - - -

其中“-”表示无初值，实际上汇编程序会填入 0。

41h	42h	43h	10	10h	45h	46h	-1	?	4	4	4	-1	?	4	4	4	-1	?	4	4	4

（2）

10h	00h	0fbh	0ffh	?	?	?	?	?	?

〔习题 3.11〕

请设置一个数据段，按照如下要求定义变量：

⑴ my1b 为字符串变量，表示字符串“Personal Computer”

⑵ my2b 为用十进制数表示的字节变量，这个数的大小为 20

⑶ my3b 为用十六进制数表示的字节变量，这个数的大小为 20

⑷ my4b 为用二进制数表示的字节变量，这个数的大小为 20

⑸ my5w 为 20 个未赋值的字变量

⑹ my6c 为 100 的符号常量

⑺ my7c 为字符串常量，代替字符串“Personal Computer”

〔解答〕

```
my1b    db 'Personal Computer'
my2b    db 20
my3b    db 14h
```

my4b db 00010100b

my5w dw 20 dup(?)

my6c = 100

my7c equ <Personal Computer>

3.12 解：

利用定位伪指令控制，如 org,even,align

3.13 解：

包括逻辑地址和类型两种属性。

〔习题 3.14〕

AX=114H

AX=6

AX=0DH

AX=02H

〔习题 3.15〕

假设 myword 是一个字变量，mybyte1 和 mybyte2 是两个字节变量，指出下列语句中的具体错误原因。

⑴ mov byte ptr [bx],1000

⑵ mov bx,offset myword[si]

⑶ cmp mybyte1,mybyte2

⑷ mov mybyte1,al+1

⑸ sub al,myword

⑹ jnz myword

〔解答〕

(1) 1000 超过一个字节所能表达的最大整数

(2)SI 应为偶数

(3)两个内存单元不能直接运算

(4)不能使用 al+1，应改为[al+1]

(5)源操作数与目的操作数类型不匹配

(6)条件转移指令后面应接标号，而不是变量

〔习题 3.16〕

编写一个程序，把从键盘输入的一个小写字母用大写字母显示出来。

〔解答〕

```
mov ah,1        ；只允许输入小写字母
int 21h
sub al,20h      ；转换为大写字母
mov dl,al
mov ah,2
```

int 21h ；显示

3.17 解：

```
        mov bx,offset LEDtable
        mov al,lednum
        xlat
```

3.18 解：

```
        mov ax, bufX
        cmp ax, bufY
        jae done
        mov ax, bufY
done:   mov bufZ, ax
```

3.19 解：

```
        .model small
        .stack
        .data
bufX    dw -7
signX   db ?
        .code
        .startup
        cmp bufX,0      ;test bufX,80h
        jl next           ;jnz next
        mov signX,0
        jmp done
next:   mov signX,-1
done:   .exit 0
        end
```

3.20 解：

```
        mov dl,' 2'
        mov ax,bufX
        cmp ax,bufY
        je next1
        dec dl
next1:  cmp ax,bufZ
        je next2
        dec dl
next2:  mov ah,2
        int 21h
```

3.21 解：

```
        ;代码段
        mov al,number
```

```
            mov bx,0          ;BX←记录为 1 的位数
restart:    cmp al,0          ;AL＝0 结束
            jz done
again:      shr al,1          ;最低位右移进入 CF
            jc next           ;为 1，转移
            inc bx            ;不为 1，继续
            jmp again
next:       push ax
            push bx
            shl bx,1          ;位数乘以 2（偏移地址要用 2 个字节单元）
            jmp addrs[bx]     ;间接转移：IP←[table＋BX]
            ;以下是各个处理程序段
fun0:       mov dl,'0'
            jmp disp
fun1:       mov dl,'1'
            jmp disp
fun2:       mov dl,'2'
            jmp disp
fun3:       mov dl,'3'
            jmp disp
fun4:       mov dl,'4'
            jmp disp
fun5:       mov dl,'5'
            jmp disp
fun6:       mov dl,'6'
            jmp disp
fun7:       mov dl,'7'
            jmp disp
            ;
disp:       mov ah,2          ;显示一个字符
            int 21h
            pop bx
            pop ax
            jmp restart
done:       …
```

3.22　编制程序完成 12H、45H、0F3H、6AH、20H、0FEH、90H、0C8H、57H 和 34H 等 10 个字节数据之和，并将结果存入字节变量 SUM 中（不考虑溢出和进位）。

```
; wjxt322.asm
            .model  small
            .stack
            .data
b_data      db  12h,45h,0f3h,6ah,20h,0feh,90h,0c8h,57h,34h ; 原始数据
num         equ  10           ; 数据个数
sum         db  ?             ; 预留结果单元
```

```
            .code
            .startup
            xor si, si      ; 位移量清零
            xor al, al      ; 取第一个数
            mov cx, num     ; 累加次数
again:      add al, b_data[si]; 累加
            inc si          ; 指向下一个数
            loop again      ; 如未完，继续累加
            mov sum, al     ; 完了，存结果
            .exit 0
            end
```

3.23　求主存 0040h：0 开始的一个 64KB 物理段中共有多少个空格?

```
; wjxt323.asm
            .model small
            .code
start:      mov ax,0040h    ; 送段地址
            mov ds, ax
            mov si, 0       ; 偏移地址
            mov cx, si      ; 计数（循环次数）
            xor ax, ax      ; 空格计数器清零
again:      cmp byte ptr [si], 20h ; 与空格的 ASCII 码比较
            jne next        ; 不是空格，转
            inc ax          ; 是空格，空格数加 1
next:       inc si          ; 修改地址指针
            loop again      ; cx＝cx－1，如 cx＝0 退出循环
            .exit 0
            end start
```

3.24　编写计算 100 个 16 位正整数之和的程序。如果和不超过 16 位字的范围（65535），则保存其和到 wordsum，如超过则显示‘overflow’。

答：

```
            ; 数据段
count       equ 100
parray      dw count dup(?) ; 假设有 100 个数据
wordsum     dw 0
msg         db ‘overflow’,’ $’
            ; 代码段
            mov cx,count
            mov ax,0
            mov bx,offset parray
again:      add ax,[bx]
            jnc next
            mov dx,offset msg
            mov ah,9
            int 21h         ; 显示溢出信息
```

```
            jmp done          ; 然后，跳出循环体
next:       add bx,2
            loop again
            mov wordsum,ax
done:       …
```

3.25　编程把一个 16 位无符号二进制数转换成为用 8421BCD 码表示的 5 位十进制数。转换算法可以是：用二进制数除以 10000，商为“万位”，再用余数除以 1000，得到“千位”；依次用余数除以 100、10 和 1，得到“百位”、“十位”和“个位”。

```
;wjxt325.asm
            .model small
            .stack 256
            .data
array       dw ?              ; 源字数据
dbcd        db 5 dup(?)       ; 五位 bcd 结果，高对高低对低
            .code
            .startup
            mov dx, array     ; 取源数据（余数）
            mov bx, 10000     ; 除数
            mov cx, 10        ; 除数系数
            mov si, 4         ; 目的数据高位位移量
again:      mov ax, dx        ; dx.ax 中存放被除数
            mov dx, 0
            div bx            ; 除于 bx，商 ax，余数 dx
            mov dbcd[si], al; 商＜10，存结果
            push dx           ; 暂存余数
            mov ax, bx        ; 除数除于 10
            mov dx,0
            div cx            ; dx.ax 除于 cx，商 ax、余数 0 存在 dx
            mov bx, ax        ; bx 是除数
            pop dx
            dec si            ; 目的数据位移量减 1
            jnz again
            mov dbcd, dl      ; 存个位数（ < 10 ）
            .exit 0
            end
```

3.26 解：

（1）汇编语言中，子程序要用一对过程伪指令 PROC 和 ENDP 声明，格式如下：

```
过程名  PROC [NEAR|FAR]
        ……          ; 过程体
过程名  ENDP
```

（2）保护用到的寄存器内容，以便子程序返回时进行相应的恢复。

（3）改错：

```
crazy   proc
        pish    bx
```

```
            push    cx
            xor ax,ax
            xor dx,dx
    again:add   a,[bx]
            adc dx,0
            inc bx
            inc bx
            loop    again
            pop cx
            pop bx
```

3.27 解（不需调用 HTOASC 子程序）：

```
again:      mov ah,1
            int 21h
            cmp al,1bh      ; ESC 的 ASCII 码是 1bh
            je done
            mov dl,al
            mov ah,2
            int 21h         ; 是大写字母则转换为小写字母
            jmp again
done:       …
```

3.28 解答：

```
asctob      proc
            push cx
            and dh,0fh      ; 先转换十位数
            shl dh,1        ; 十位数乘以 10（采用移位指令）
            mov ch,dh
            shl dh,1
            shl dh,1
            add dh,ch
            and dl,0fh      ; 转换个位数
            add dh,dl       ; 十位数加个位数
            mov al,dh       ; 设置出口参数
            pop cx
            ret
asctob      endp
```

3.29 解：

```
DIPASC      proc            ; 入口参数：AL＝要显示的一个 16 进制数
            push cx
            push dx
            push ax
            mov cl,4        ; 转换高位
            shr al,cl
            call HTOASC
            mov dl,al       ; 显示
```

```
            mov ah,2
            int 21h
            pop ax          ; 转换低位
            call HTOASC
            mov dl,al       ; 显示
            mov ah,2
            int 21h
            mov dl,' H'     ; 显示一个字母“H”
            mov ah,2
            int 21h
            pop dx
            pop cx
            ret
DIPASC      endp
HTOASC      proc            ;将 AL 低 4 位表达的一位 16 进制数转换为 ASCII 码
            and al,0fh
            cmp al,9
            jbe htoasc1
            add al,37h      ;是 0AH～0FH，加 37H 转换为 ASCII 码
            ret             ;子程序返回
htoasc1:    add al,30h      ;是 0～9，加 30H 转换为 ASCII 码
            ret             ;子程序返回
HTOASC      endp
```

【3.30】 解：

```
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
```

【3.31 解】

（1）用寄存器传递参数：

最简单和常用的参数传递方法是通过寄存器，只要把参数存于约定的寄存器中就可以了

由于通用寄存器个数有限，这种方法对少量数据可以直接传递数值，而对大量数据只能传递地址

采用寄存器传递参数，注意带有出口参数的寄存器不能保护和恢复，带有入口参数的寄存器可以保护、也可以不保护，但最好能够保持一致

（2）用共享变量传递参数

子程序和主程序使用同一个变量名存取数据就是利用共享变量（全局变量）进行参数传递

如果变量定义和使用不在同一个源程序中，需要利用 PUBLIC、EXTREN 声明

如果主程序还要利用原来的变量值，则需要保护和恢复

利用共享变量传递参数，子程序的通用性较差，但特别适合在多个程序段间、尤其在不同的程序模块间传递数据

（3）用堆栈传递参数

参数传递还可以通过堆栈这个临时存储区。主程序将入口参数压入堆栈，子程序从堆栈中取出参数；子程序将出口参数压入堆栈，主程序弹出堆栈取得它们

采用堆栈传递参数是程式化的，它是编译程序处理参数传递、以及汇编语言与高级语言混合编程时的常规方法

3.32 解：

方法：主程序将入口参数压入堆栈，子程序从堆栈中取出参数；子程序将出口参数压入堆栈，主程序弹出堆栈取得它们

注意：压栈与弹栈必须要一一对应。

【3.33】解：

方法 1：

```
neg32       proc            ; 入口参数：DX.AX＝32 位有符号数
            neg ax          ; 实现 0－DX.AX 功能
            neg dx
            sbb dx,0        ; 这条指令也可以用 dec dx 代替
            ret
neg32       endp            ; 出口参数：DX.AX＝32 位有符号数的补码
```

方法 2：

```
neg32       proc            ; 入口参数：DX.AX＝32 位有符号数
            not ax          ; 实现 DX.AX 求反加 1
            not dx
            add ax,1
            adc dx,0
            ret
neg32       endp            ; 出口参数：DX.AX＝32 位有符号数的补码
```

【3.34】 解：

```
            ;数据段
array       db 12h,25h,0f0h,0a3h,3,68h,71h,0cah,0ffh,90h   ;数组
count       equ $-array     ;数组元素个数
result      db ?            ;校验和
            ;代码段
            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
            mov ax,4c00h
            int 21h
            ;计算字节校验和的通用过程
            ;入口参数：DS:BX＝数组的段地址:偏移地址，CX＝元素个数
            ;出口参数：AL＝校验和
            ;说明：除 AX/BX/CX 外，不影响其他寄存器
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
              end
```

【3.35】 解：

(1)

```
              .model small
              .stack
              .data
wdata         dw 34abh
              .code
              .startup
              mov ax,wdata
              call dispa
              .exit 0
              ;
dispa         proc
              push cx
              push dx
              mov cl,4
              mov dl,ah
              shr dl,cl
              call dldisp
              mov dl,ah
              and dl,0fh
              call dldisp
              mov dl,al
              shr dl,cl
              call dldisp
              mov dl,al
              and dl,0fh
              call dldisp
              pop dx
              pop cx
              ret
dispa         endp
              ;
dldisp        proc
              push ax
              or dl,30h
              cmp dl,39h
              jbe dldisp1
              add dl,7
dldisp1:      mov ah,2
              int 21h
              pop ax
```

```
            ret
dldisp      endp
            end
(2)
            .model small
            .stack
            .data
wdata       dw 34abh
wordtemp    dw ?
            .code
            .startup
            mov ax,wdata
            mov wordtemp,ax
            call dispa
            .exit 0
            ;
dispa       proc
            push cx
            push dx
            mov cl,4
            mov dl,byte ptr wordtemp+1
            shr dl,cl
            call dldisp
            mov dl,byte ptr wordtemp+1
            and dl,0fh
            call dldisp
            mov dl,byte ptr wordtemp
            shr dl,cl
            call dldisp
            mov dl,byte ptr wordtemp
            and dl,0fh
            call dldisp
            pop dx
            pop cx
            ret
dispa       endp
            ;
dldisp      proc
            push ax
            or dl,30h
            cmp dl,39h
            jbe dldisp1
            add dl,7
dldisp1:    mov ah,2
```

```
            int 21h
            pop ax
            ret
dldisp      endp
            end
```

(3)

```
            .model small
            .stack
            .data
wdata       dw 34abh
            .code
            .startup
            push wdata
            call dispa
            pop ax          ;add sp,2
            .exit 0
            ;
dispa       proc
            push bp
            mov bp,sp
            push ax
            push cx
            push dx
            mov ax,[bp+4]
            mov cl,4
            mov dl,ah
            shr dl,cl
            call dldisp
            mov dl,ah
            and dl,0fh
            call dldisp
            mov dl,al
            shr dl,cl
            call dldisp
            mov dl,al
            and dl,0fh
            call dldisp
            pop dx
            pop cx
            pop ax
            pop bp
            ret
dispa       endp
            ;
```

```
dldisp      proc
            push ax
            or dl,30h
            cmp dl,39h
            jbe dldisp1
            add dl,7
dldisp1:    mov ah,2
            int 21h
            pop ax
            ret
dldisp      endp
            end
```

【3.36】解：

如果利用共享变量传递函数，且变量定义和使用不在同一个源程序中，需要利用PUBLIC、EXTERN 声明。

3.37 解：

（1）宏定义由一对宏汇编伪指令 MACRO 和 ENDM 来完成，格式如下：

```
宏名        MACRO [形参表]
……          ；宏定义体
            ENDM
```

宏定义之后就可以使用它，即宏调用：

```
            宏名 [实参表]
```

（2）宏调用的格式同一般指令一样：在使用宏指令的位置写下宏名，后跟实体参数；如果有多个参数，应按形参顺序填入实参，也用逗号分隔

（3）宏展开：在汇编时，宏指令被汇编程序用对应的代码序列替代，这就是宏展开

宏展开的具体过程是：当汇编程序扫描源程序遇到已有定义的宏调用时，即用相应的宏定义体完全替代源程序的宏指令，同时用位置匹配的实参对形参进行取代

3.38 解：

宏调用的参数通过形参、实参结合实现传递，简捷直观、灵活多变。宏汇编的一大特色是它的参数。宏定义时既可以无参数，也可以有一个或多个参数；宏调用时实参的形式也非常灵活，可以是常数、变量、存储单元、指令（操作码）或它们的一部分，也可以是表达式；只要宏展开后符合汇编语言的语法规则即可。

3.39 解：

宏：仅是源程序级的简化：宏调用在汇编时进行程序语句的展开，不需要返回；不减小目标程序，执行速度没有改变

通过形参、实参结合实现参数传递，简捷直观、灵活多变

子程序：还是目标程序级的简化：子程序调用在执行时由 CALL 指令转向、RET 指令返回；形成的目标代码较短，执行速度减慢

需要利用寄存器、存储单元或堆栈等传递参数

选择：宏与子程序具有各自的特点，程序员应该根据具体问题选择使用那种方法。通常，当程序段较短或要求较快执行时，应选用宏；当程序段较长或为减小目标代码时，要选用子程序

3.40 编写一个宏指令 move doprnd,soprnd，它实现任意寻址方式的字量源操作数 soprnd 送到目的操作数 doprnd，包括存储单元到存储单元的传送功能。

答：

```
move        macro doprnd,soprnd
            mov ax,soprnd
            mov doprnd,ax
            endm
```

3.41　定义一个宏 logical，用它代表 4 条逻辑运算指令：and/or/xor/test。注意需要利用 3 个形式参数，并给出一个宏调用以及对应宏展开的例子。

答：

```
logical     macro lcode,dopd,sopd
            lcode dopd,sopd
            endm
```

例如，如果使用“and ax,[bx]”指令，可以利用该宏定义，写出宏指令如下：

```
            logical and,ax,[bx]
```

3.42　解：

```
utol        macro
            local next
            cmp al,' A'      ; 小于“A”不转换
            jb next
            cmp al,' Z'      ; 大于“A”不转换
            ja next
            add al,20h       ; 是大写字母则转换为小写字母
next:
            endm
```

3.43　定义一个宏 movestr strn,dstr,sstr，它将 strn 个字符从一个字符区 sstr 传送到另一个字符区 dstr

解：（假设它们都在数据段）

```
movestr     macro strn,dstr,sstr
            mov cx,ds
            mov es,cx
            mov cx,strn
            mov di,offset dstr
            mov si,offset sstr
            cld
            rep movsb        ; ; 重复传送 ES:[DI]←DS:[SI]
            endm
```

第 4 章 微机总线

〔习题 4.1〕

微机总线的信号线包括_____、_____、_____、以及电源和地线。微机系统可以将总线划分为三层（类），它们是_____、_____和_____。

〔解答〕

数据总线 、 地址总线 、 控制总线

芯片总线、 内总线、外总线

〔习题 4.2〕

占用总线进行数据传输，一般需要经过总线请求和仲裁、_____、_____和结束 4 个阶段。

〔解答〕

寻址，数据传送

〔习题 4.6〕

RESET：复位请求，高电平有效时，CPU 回到初始状态。

HOLD：总线请求，高电平有效时，其他总线主控设备向 CPU 申请占用总线。

NMI：不可屏蔽中断请求。外界向 CPU 申请不可屏蔽中断。

INTR：可屏蔽中断请求。高电平有效时，中断请求设备向 CPU 申请可屏蔽中断。

〔习题 4.7〕

执行一条指令所需要的时间被称为_____周期，而总线周期指的是_____，8088 基本的总线周期由_____个 T 组成。如果 8088 的 CLK 引脚接 5MHz 的时钟信号，那么每个 T 状态的持续时间为_____。

〔解答〕

指令

CPU 通过总线操作与外界（存储器和 I/O 端口）进行一次数据交换的过程（时间）

4

200ns

〔习题 4.8〕

请解释 8088 以下引脚信号：CLK、A_{19}/S_6～A_{16}/S_3、A_{15}～A_8、AD_7～AD_0、IO/$\overline{M}$、$\overline{RD}$ 、$\overline{WR}$ 、ALE 的含义，并画出它们在存储器写总线周期中的波形示意。

〔解答〕

CLK：CPU 时钟信号

A19/S6～A16/S3：地址信号 A19～A16 和状态信号 S6～S3 分时复用信号

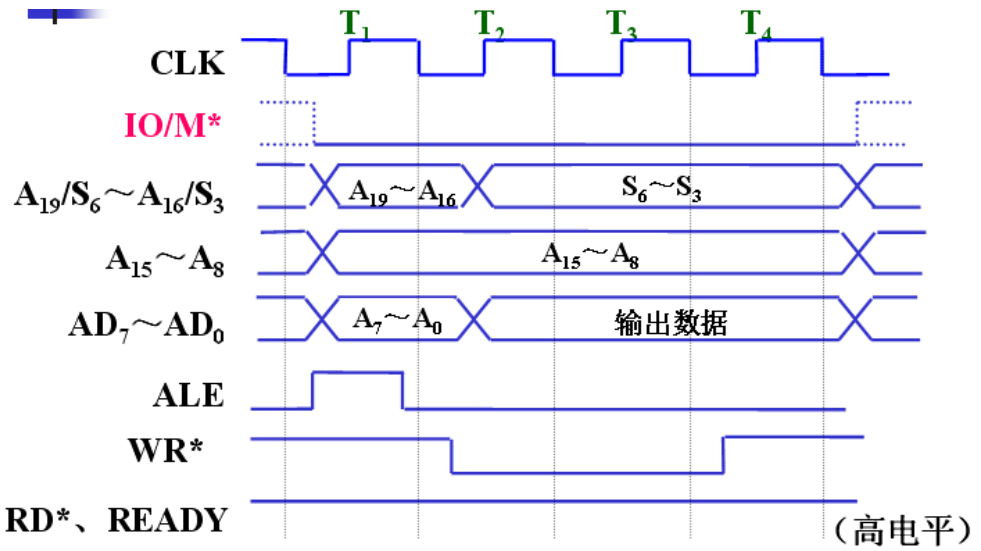
A15～A8：地址信号 A15～A8

AD7～AD0：地址信号 A7～A0 和数据信号 D7～D0 分时复用信号

IO/M*：I/O 接口和存储器操作区别信号

RD*：读控制信号，WR*：写控制信号

ALE：地址锁存信号



〔习题 4.9〕

在 8088 的工作过程中，什么情况下会产生 T_W？具体发生在什么时刻？

〔解答〕

□ 当 8088 进行读写存储器或 I/O 接口时，如果存储器或 I/O 接口无法满足 CPU 的读写时序（来不及提供或读取数据时），需要 CPU 插入等待状态 Tw。

□ 在读写总线周期的 T3 和 T4 之间插入 Tw。

〔习题 4.10〕

（1）T1 周期，CPU 进行读操作。

(2) T2-T4 期间，CPU 对数据总线输出高阻态，选通存储器或 I/O 接口，向 CPU 传送数据。

（3）T4 的下降沿，CPU 对数据总线采样。

〔习题 4.11〕

在 8088 系统中，读取指令“ADD [2000H], AX”（指令长度为 3 个字节）和执行该指令各需要几个总线周期？它们各是什么样的总线周期？

〔解答〕

□ 8088 每个总线周期只能读写一个字节数据。所以读取指令长度为 3 个字节的指令“ADD [2000H], AX”需要 3 个时钟周期，执行时需要 2 个时钟周期。

□ 读取指令是 存储器读 总线周期，执行时是 存储器写 总线周期。

第 5 章 主存储器

〔习题 5.3〕

类似处理器总线，存储器芯片也分成数据、地址和控制 3 类引脚。以存储结构为 32K×8 的 SRAM 62256 为例，该芯片应有_____个数据引脚、_____个地址引脚，3 个典型的控制引脚分别是_____、_____和_____。

〔解答〕

8

15

片选

输出允许

写入允许

〔习题 5.7〕

EEPROM 的擦写与闪存的擦写有什么不同？以 AT28C040 或 AT29C512 为例，说明常用的两种判断擦写是否完成的方法，并估算两者完成整个芯片编程的最快时间。

〔解答〕

AT28C040 最快编程时间：

（512×1024/256）×10ms＝20480ms≈20s

AT29C512 最快编程时间：

512 扇区×（10ms＋128 字节×（$150\times10^{-3}+90\times10^{-6}$）ms）≈14950.4ms≈15s

〔习题 5.8〕

SRAM 芯片的片选引脚有什么用途？假设在 8088 微处理器系统中，地址信号 A_{19}～A_{15}输出 01011 时译码电路产生一个有效的片选信号，则该片选信号将占有多少主存容量？其地址范围是什么？

〔解答〕

主存容量：2^{15}＝32KB，因低位地址信号的个数是 15 个。

地址范围：01011 000000000000000 ～ 01011 11111111111111，即 58000H～5FFFFH。

〔习题 5.11〕

什么是存储器芯片连接中的“位扩展”和“字扩展”？采用 DRAM 21256（256K×1）构成 512KB 的 RAM 存储模块，需要多少个芯片，怎样进行位扩展和字扩展？

〔解答〕

位扩充：存储器芯片数据引脚个数小于主机数据信号个数时，利用多个存储器芯片在数据“位”方向的扩充。

字扩充：当一个存储器芯片不能满足系统存储容量时，利用多个存储器芯片在“数据字”方向的扩充。

组成 512KB 存储模块，用 DRAM 21256（256K×1）需要 16 个芯片；位扩充 8 个，字扩充 2 组。

〔习题 5.13〕

给出图 5-28 中 4 个存储器芯片各自占用的地址范围。如果采用部分译码，要指出重复的地址范围。

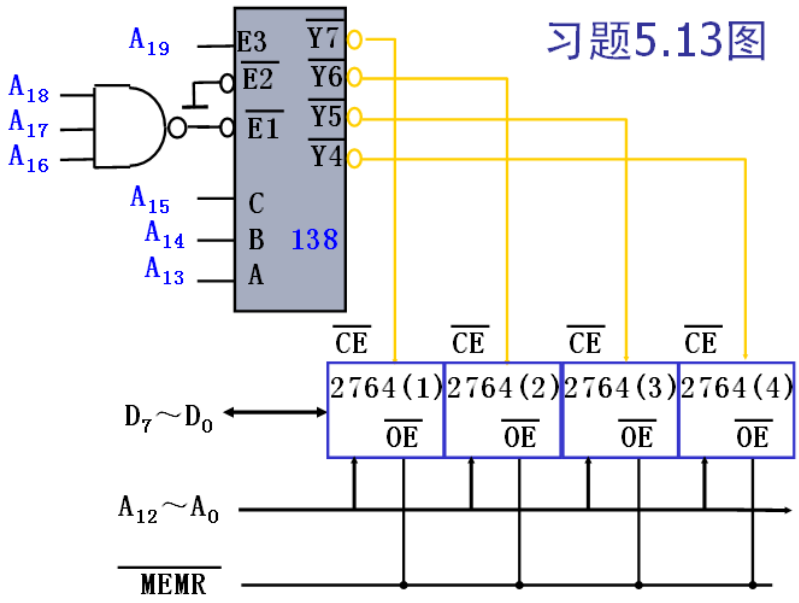
〔解答〕

4 个存储芯片各自的可用地址范围：

芯片号	A19A18	A17A16	A15～A0	地址范围
1	00	××	全 0～全 1	00000H～3FFFFH
2	01	××	全 0～全 1	40000H～7FFFFH
3	11	×0	全 0～全 1	C0000H～CFFFFH，重复地址：E0000H～EFFFFH
4	11	×1	全 0～全 1	D0000H～DFFFFH，重复地址：F0000H～FFFFFH

〔习题 5.14〕

	A_{19}～A_{16}	A_{15}～A_{13}	A_{12}～A_0	地址
1	1111	100	全0～全1	F8000H～F9FFFH
2	1111	101	全0～全1	FA000H～FBFFFH
3	1111	110	全0～全1	FC000H～FDFFFH
4	1111	111	全0～全1	FE000H～FFFFFH



第 6 章 输入输出接口

〔习题 6.1〕

典型的 I/O 接口电路通常有哪 3 类可编程寄存器？各自的作用是什么？

〔解答〕

（1）数据寄存器：保存外设给 CPU 和 CPU 发往外设的数据

（2）状态寄存器：保存外设或接口电路的状态

（3）控制寄存器：保存 CPU 给外设或接口电路的命令

〔习题 6.2〕

I/O 端口与存储器地址常有_____和_____两种编排方式，8088/8086 处理器支持后者，设计有专门的 I/O 指令。其中指令 IN 是将数据从_____传输到_____，执行该指令时 8088/8086 处理器引脚产生_____总线周期。指令“OUT DX, AL”的目的操作数是_____寻址方式，源操作数是_____寻址方式。

〔解答〕

（I/O 端口与存储器地址）统一编址

（I/O 端口与存储器地址）独立编址

外设

处理器

I/O 读

（I/O 端口的 DX）寄存器间接

寄存器

〔习题 6.4〕

基于教程 P142 图 6-7 接口电路，编程使发光二极管循环发光。具体要求是：单独按下开关 K_0，发光二极管以 L_0、L_1、L_2、……L_7 顺序依次点亮，每个维持 200ms，并不断重复，直到有其他按键操作；单独按下开关 K_1，发光二极管以 L_7、L_6、L_5、……L_0 顺序依次点亮，每个也维持 200ms，并不断重复，直到有其他按键操作；其他开关组合均不发光，单独按下开关 K_7，则退出控制程序。延时 200ms 可以直接调用子程序 DELAY 实现。

〔解答〕

```
again:   mov dx,6000h
         mov al,0ffh
         out dx,al              ;全不亮
again1:  in al,dx
         cmp al,7fh             ;D7~D0＝0111111B ？
         jz done                ;单独按下 K7，退出
         cmp al,0feh            ;D7~D0＝11111110B ？
         jz next1               ;单独按下 K0，转移到 next1
         cmp al,0fdh            ;D7~D0＝11111101B ？
```

```
        jz next2            ;单独按下 K1，转移到 next2
        jmp again           ;其它情况不点亮
next1:  mov cx,8
        mov al,1            ;从 K0 开始
next11: out dx,al           ;某个 LED 电亮
        call delay          ;延时 200ms
        shl al,1            ;rol al,1
        loop next11
        jmp again1

next2:  mov cx,8
        mov al,80h          ;从 K7 开始
next21: out dx,al           ;某个 LED 电亮
        call delay          ;延时 200ms
        shr al,1            ;ror al,1
        loop next21
        jmp again1

done:   mov al,0ffh
        out dx,al           ;全不亮
```

〔习题 6.5〕

有一个查询输入接口电路类似图 6-9，但其数据端口为 8F40H、状态端口为 8F42H。从状态端口最低位可以获知输入设备是否准备好一个字节的数据：D_0＝1 表示准备好，D_0＝0 说明没准备好。不考虑查询超时，编程从输入设备读取 100 个字节保存到 INBUF 缓冲区。

〔解答〕

```
        mov bx,offset inbuf
        mov cx,100
again:  mov dx,8f42h
status: in al,dx            ; 查询一次
        test al,01h
        jz status
        mov dx,08f40h
        in al,dx            ; 输入一个字节
        mov [bx],al
        inc bx
        loop again          ; 循环，输入 100 个字节
```

〔习题 6.6〕

有一个查询输出接口电路类似图 6-10，但其数据端口和状态端口均为 8000H，并从状态端口的 D_6 位获知输出设备是否能够接收一个字节的数据：D_6＝1 表示可以接收、D_6＝0 说明不能接收。不考虑查询超时，编程将存放于缓冲区 OUTBUF 处的字符串（以 0 为结束标志）传送给输出设备。

〔解答〕

```
        mov bx,offset outbuf
        mov dx,8000h
again:  mov ah,[bx]
        cmp ah,0
        jz done
status: in al,dx              ; 查询一次
        test al,40h
        jnz status
        mov al,ah
        out dx,al             ; 输出一个字节
        inc bx
        jmp again             ; 循环
done:   ......
```

[习题 6.7]

中断请求：外设通过硬件信号的形式、向处理器引脚发送有效请求信号。

中断响应：在满足一定条件时，处理器进入中断响应总线周期。

关中断：处理器在响应中断后会自动关闭中断。

断点保护：处理器在响应中断后将自动保护断点地址。

中断源识别：处理器识别出当前究竟是哪个中断源提出了请求，并明确与之相应的中断服务程序所在主存位置。

现场保护：对处理器执行程序有影响的工作环境（主要是寄存器）进行保护。

中断服务：处理器执行相应的中断服务程序，进行数据传送等处理工作。

恢复现场：完成中断服务后，恢复处理器原来的工作环境。

开中断：处理器允许新的可屏蔽中断。

中断返回：处理器执行中断返回指令，程序返回断点继续执行原来的程序。

第 7 章 中断控制接口

〔习题 7.1〕

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

〔习题 7.2〕

8088 中断向量表的作用是什么？

中断向量表是一种表数据结构。是中断向量号与对应中断服务程序之间的连接表。

〔习题 7.4〕

8259A 中 IRR、IMR 和 ISR 三个寄存器的作用是什么？

〔解答〕

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。

中断服务寄存器 ISR：保存正在被 8259A 服务着的中断状态，Di 位为 1 表示 IRi 中断正在服务中；为 0 表示没有被服务。

中断屏蔽寄存器 IMR：保存对中断请求信号 IR 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。

〔习题 7.5〕

PC/XT 机的 ROM-BIOS 对 8259A 的初始化程序如下：

```
mov al,13h
out 20h,al
mov al,08h
out 21h,al
mov al,09h
out 21h,al
```

请说明其设定的工作方式。

〔解答〕

```
mov al, 13h    ；13H = 0001 0011，设定工作方式：单片，边沿触发，要写入 ICW4
out 20h, al    ；写入 ICW1：主（单）片地址是 20H（参见表 8-1）

mov al, 08h    ；08H =0000 1000，设定主（单）片 IR0 的中断向量号为 08H
out 21h, al    ；写入 ICW2：地址参见表 8-1

mov al, 09h    ；09H=0000 1001，设定为 16 位 80x86CPU、非自动中断结束、
               ；该片 8259A 是从片（按教材是“从”片，个人认为应该是“主片”）、
```

；8259A 数据线采用缓冲方式、8259A 工作于普通全嵌套方式

out 21h, al ；写入 ICW4：地址参见表 8-1

〔习题 7.6〕

某时刻 8259A 的 IRR 内容是 08H，说明_____。某时刻 8259A 的 ISR 内容是 08H，说明_____。在两片 8259A 级连的中断电路中，主片的第 5 级 IR_5 作为从片的中断请求输入，则初始化主、从片时，ICW3 的控制字分别是_____和_____。

〔解答〕

IR3 引脚有中断请求

IR3 正在被中断服务，其它不被处理

20H 和 05H。

〔习题 7.8〕某一 8086CPU 系统中，采用一片 8259A 进行中断管理。设定 8259A 工作在普通全嵌套方式，发送 EOI 命令结束中断，采用边沿触发方式请求中断，IR_0 对应的中断向量号为 90H。另外，8259A 在系统中的 I/O 地址是 FFDCH（A_0=0）和 FFDEH（A_0=1）。请编写 8259A 的初始化程序段。

〔解答〕

```
MOV   DX，0FFDCH      ；地址参考教材表 8-1
MOV   AL，00010011    ；参见教材图 8-7
OUT   DX，AL          ；

MOV   DX，0FFDEH      ；地址参考教材表 8-1
MOV   AL，10001000    ；参见教材图 8-7、204/205 页例程
OUT   DX，AL          ；

MOV   AL，00000111    ；参见教材图 8-7
OUT   DX，AL          ；
```

〔习题 7.11〕

下段程序读出的是 8259A 的哪个寄存器？

```
mov al,0bh
out 20h,al
nop
in   al,20h
```

〔解答〕

读取中断服务寄存器 ISR 的内容。

因为执行输入指令（A0＝0）之前，执行的输出指令，写入了 OCW3（D4D3＝01），其中 P RR RIS（D2D1D0）＝011 指明随后读出 ISR。

不是查询字。

〔习题 7.14〕

中断服务程序的人口处为什么通常要使用开中断指令？

〔解答〕

□ 开中断，以便可以实现中断嵌套。

补充：1. 如何利用 DOS 功能调用设置中断向量？

——解答参考教程第 7.1.3 节 P157（第二版 195 页）

解答：

设置中断向量即为把新中断向量写入中断向量表内。方法如下：

```
MOV   AH,25H
MOV   AL,中断向量号
MOV   DS,新中断向量的段地址
MOV   DX,新中断向量的偏移地址
INT    21H
```

补充：2. 如何利用 DOS 功能调用获取中断向量？

——解答参考教材第 7.1.3 节 P157（第二版 195 页）

解答：

获取中断向量即为把中断向量表内的中断向量读出到 ES:BX 中。方法如下：

```
MOV AH,35H
MOV AL,中断向量号
INT   21H
```

补充：3. 如何开 CPU 的中断允许？

解答：

用 STI 指令使 IF=1。

补充：4. 如何开 8259A 的中断允许？

解答：

OCW1 中的 IMR 的第 i 位=0，即允许 IR 中的第 i 位发出中断申请。

编程方法参见教程 P169 5）。第二版 P207 页（5）。

第 8 章 定时计数控制接口

〔习题 8.2〕

8253 每个计数通道与外设接口有哪些信号线，每个信号的用途是什么？

〔解答〕

CLK 时钟输入信号——在计数过程中，此引脚上每输入一个时钟信号（下降沿），计数器的计数值减 1

GATE 门控输入信号——控制计数器工作，可分成电平控制和上升沿控制两种类型

OUT 计数器输出信号——当一次计数过程结束（计数值减为 0），OUT 引脚上将产生一个输出信号

〔习题 8.3〕

8253 每个通道有 6 种工作方式可供选择。若设定某通道为方式 0 后，其输出引脚为 低 电平；当 写入计数初值（并进入减 1 计数器） 后通道开始计数， CLK 信号端每来一个脉冲 减 1 计数器 就减 1；当 计数器减为 0 ，则输出引脚输出 高 电平，表示计数结束。8253 的 CLK0 接 1.5MHz 的时钟，欲使 OUT0 产生频率为 300KHz 的方波信号，则 8253 的计数值应为 5（＝1.5MHz÷300KHz） ，应选用的工作方式是 3

〔习题 8.4〕

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。

⑴ 使计数器 1 工作在方式 0，仅用 8 位二进制计数，计数初值为 128。

⑵ 使计数器 0 工作在方式 1，按 BCD 码计数，计数值为 3000。

⑶ 使计数器 2 工作在方式 2，计数值为 02F0H。

〔解答〕

```
（1）   mov al,50h
        mov dx,207h
        out   dx,al
        mov al,128     ；80h
        mov dx,205h
        out   dx,al
（2）   mov al,33h
        mov dx,207h
        out   dx,al
        mov ax,3000h ；不是 3000
        mov dx,204h
        out   dx,al
        mov al,ah
        out   dx,al
（3）   mov al,0b4h
        mov dx,207h
        out   dx,al
        mov al,02f0h
```

```
mov dx,206h
out   dx,al
mov al,ah
out   dx,al
```

【8.5】

- 设8253计数器0～2和控制字的I/O地址依次为F8H～FBH，说明如下程序的作用。

解答：

```
；设置计数器0采用工作方式1，先低后高写入计数值
；BCD码十进制计数
    mov al,33h
    out 0fbh,al
；计数值为5080
    mov al,80h
    out 0f8h,al
    mov al,50h
    out 0f8h,al
```

总结：计数器0采用工作方式1，计数初值为5080

www.docin.com

第 10 章 并行接口

[习题 10.1]

A 组，B 组工作于方式 0 时，24 条作为一般 I/O 线，PA7-0,PB7-0,PC7-4,PC3-0 可分别定义输入/输出；

A 组，B 组工作于方式 1 时各使用 C 口 3 条线作联络线。C 口其余可做一般 I/O 线；

A 组工作于方式 2 时，需使用 C 口 5 条线作联络线.

〔习题 10.2〕

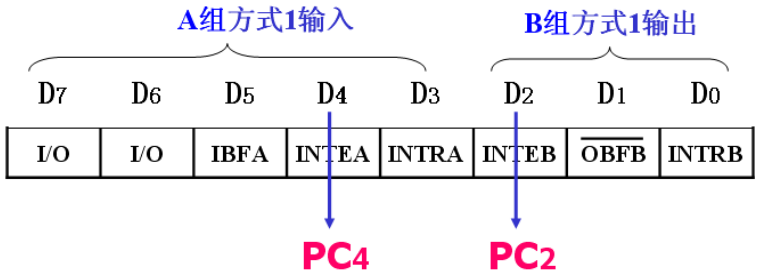
- 方式控制字为：1 01 1 X 1 1 X
- 其中任意的D3位确定PC4～PC7、D0位确定PC0～PC3是方式0输入还是输出；由于两组都为方式1输入，PC0～PC5被征用为控制信号。所以D3位仅确定PC6、PC7是方式0输入还是输出；而D0位没有作用

www.docin.com

〔习题 10.4〕

设定 8255A 的口 A 为方式 1 输入，口 B 为方式 1 输出，则读取口 C 的数据的各位是什么含义？

〔解答〕



〔习题 10.5〕

对 8255A 的控制寄存器写入 B0H，则其端口 C 的 PC_5 引脚是什么作用的信号线？

〔解答〕

□ 方式控制字为 B0H＝10110000B，说明 A 组为方式 1 输入，它将征用 PC5 引脚作为输入缓冲器满信号 IBFA

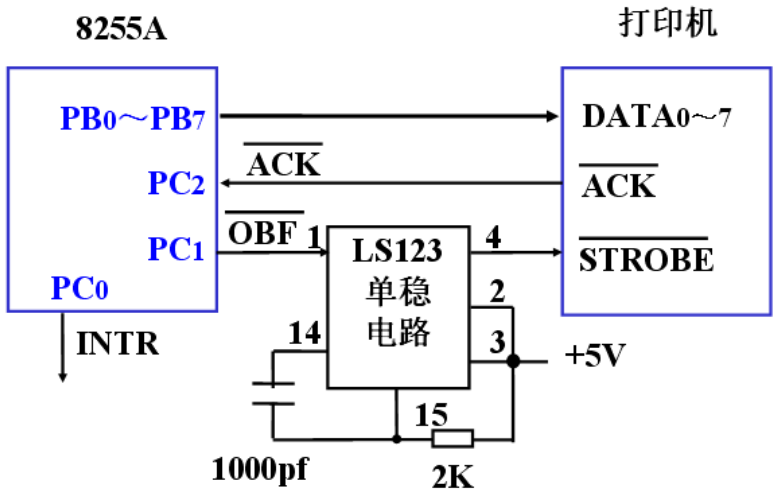
〔习题 10.6〕

- 接口电路：用端口引脚**PB0～PB7**与打印机**DATA0～7**连接，其他不变
- 程序：用端口**B**的**I/O**地址**FFFAH**替换端口**A**的**FFF8H**
- 另外，应设置端口**B**为方式**0**输出。但由于原控制字已经做到，所以无需修改

〔习题 10.7〕

第 10.2.3 节用 8255A 端口 A 方式 1 与打印机接口，如果采用端口 B，其他不变，请说明如何修改接口电路和程序。

〔解答〕




```
mov dx,0fffeh
mov al,84h
out dx,al
mov al,04h
;使 INTEB（PC2）为 0，禁止中断
out dx,al
……

mov cx,counter      ;打印字节数送 CX
mov bx,offset buffer      ;取字符串首地址
call prints;调用打印子程序
prints      proc
            push ax   ;保护寄存器
            push dx
print1:     mov al,[bx]     ;取一个数据
            mov dx,0fffah
            out dx,al  ;从端口 B 输出
mov dx,0fffch
print2:     in al,dx
            test al,02h       ;检测（PC1）为 1 否?
            jz print2
            inc bx
            loop print1
            pop dx
```

```
        pop ax
        ret
prints  endp
```

〔习题 10.8〕

设一工业控制系统，有四个控制点，分别由四个对应的输入端控制，现用 8255A 的端口 C 实现该系统的控制，如图 10-22。开关 K_0～K_3 打开则对应发光二极管 L_0～L_3 亮，表示系统该控制点运行正常；开关闭合则对应发光二极管不亮，说明该控制点出现故障。编写 8255A 的初始化程序和这段控制程序。

〔解答〕

```
；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
```

〔习题 10.16〕

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（K_1 为 1，K_2 为 2，…K_8 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。

〔解答〕

```
显示代码表
table   db 0c0h  ；对应 0（任意）
        db 0f9h,0a4h,0b0h,99h
        db 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
```

第 12 章 模拟接口

〔习题 12.1〕

说明在模拟输入输出系统中，传感器、放大器、滤波器、多路开关、采样保持器的作用。DAC 和 ADC 芯片是什么功能的器件？

〔解答〕

传感器：将各种现场的物理量测量出来并转换成电信号。

放大器：放大器把传感器输出的信号放大到 ADC 所需的量程范围。

低通滤波器：滤波器用于降低噪声、滤去高频干扰，以增加信噪比。

多路开关：对多个模拟信号分时地接通到 A/D 转换器上转换，达到共用 A/D 转换器

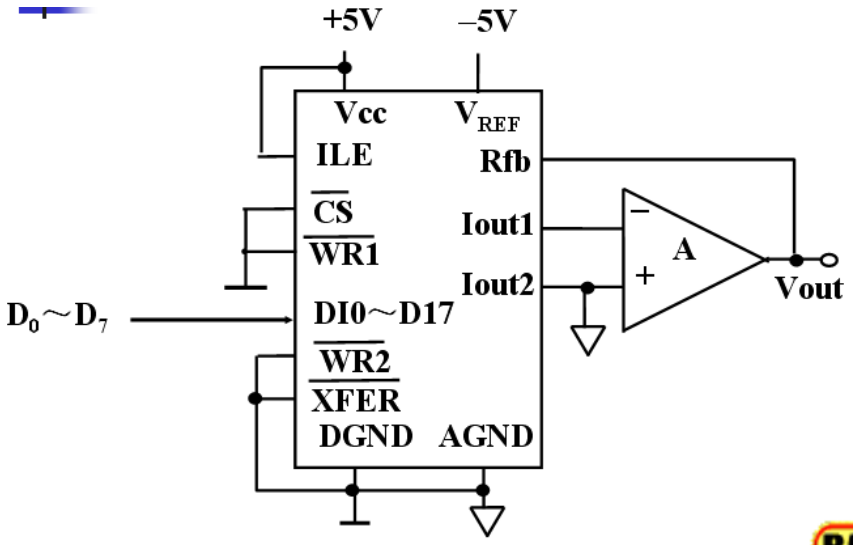
以节省硬件的目的。

采样保持器：对高速变化的信号，使用采样保持器可保证 A/D 转换期间信号不变，保证转换精度。

〔习题 12.2〕

如果将 DAC0832 接成直通工作方式，画图说明其数字接口引脚如何连接。

〔解答〕



〔习题 12.3〕

对应第 13.2.4 节的图 13-9a 电路，编写输出一个 12 位数字量的程序段。假定这 12 位数据在 BX 的低 12 位中。

〔解答〕

```
mov dx,port1l
mov al,bl
out dx,al
mov dx,port1h
mov al,bh
out dx,al
mov dx,port2
out dx,al
```

〔习题 12.5〕

ADC 的转换结束信号起什么作用，可以如何使用该信号，以便读取转换结果？

〔解答〕

当 A/D 转换结束，ADC 输出一个转换结束信号（EOC），通知主机读取结果，主机检查判断是否结束的方法有 4 种，不同处理方式对应的应用程序设计方法也不同：

（1） 查询方式：把结束信号作为状态信号经三态缓冲区送到主机系统数据总线的某一位置。ADC 开始转换后，主机不断查询这个状态位，发现结束信号有效，便读取数据。（常用）

（2） 中断方式：把结束信号作为中断请求信号接到主机的中断请求线上，ADC 转换结束，主动向 cpu 申请中断，cpu 响应中断后，在中断服务程序中读取数据

（3） 延时方式： 不使用转换信号，主机主动 A/D 转换后，延迟一段略大于 A/D 转换时间的时间，此时转换已结束即可读取数据

（4） DMA 方式：把结束信号作为 DMA 请求信号。 转换结束后， A/D 即启动 DMA 传送，通过 DMA 控制器直接将数据送入内存缓冲区 8255A

〔习题 12.6〕

某控制接口电路如图 12-16。需要控制时，8255A 的 PC_7 输出一个正脉冲信号 START 启动 A/D 转换；ADC 转换结束在提供一个低脉冲结束信号 EOC 的同时送出数字量。CPU 采集该数据，进行处理，产生控制信号。现已存在一个处理子程序 ADPRCS，其入口参数是在 AL 寄存器存入待处理的数字量，出口参数为 AL 寄存器给出处理后的数字量。假定 8255A 端口 A、B、C 及控制端口的地址依次为 FFF8H～FFFBH，要求 8255A 的端口 A 为方式 1 输入、端口 B 为方式 0 输出。编写采用查询方式读取数据，实现上述功能的程序段。

〔解答〕

```
；8255A 初始化
mov al,1011000×b
mov dx,0fffbh
out dx,al
；使 PC7＝0（START 为低）
mov al,00001110b
mov dx,0fffbh
out dx,al
；启动 A/D 转换
mov al,00001111b
mov dx,0fffbh
out dx,al ；使 PC7＝1（START 为高）
nop
mov al,00001110b
out dx,al ；使 PC7＝0（START 为低）
；查询是否转换结束
        mov dx,0fffah
again:  in dx,al
        test al,20h
    ；PC5＝0（转换未结束，继续检测）
        jz again
```

；PC5＝1（转换结束）

mov dx,0fff8h；输入数据

in al,dx

call adprcs　　；处理数据

mov dx,0fff9h

out dx,al ；输出数据

〔习题 12.7〕

假设系统扩展有一片 8255A 供用户使用，请设计一个用 8255A 与 ADC0809 接口的电路连接图，并给出启动转换、读取结果的程序段。为简化设计，可只使用 ADC0809 的一个模拟输入端，例如 IN0。

〔解答〕

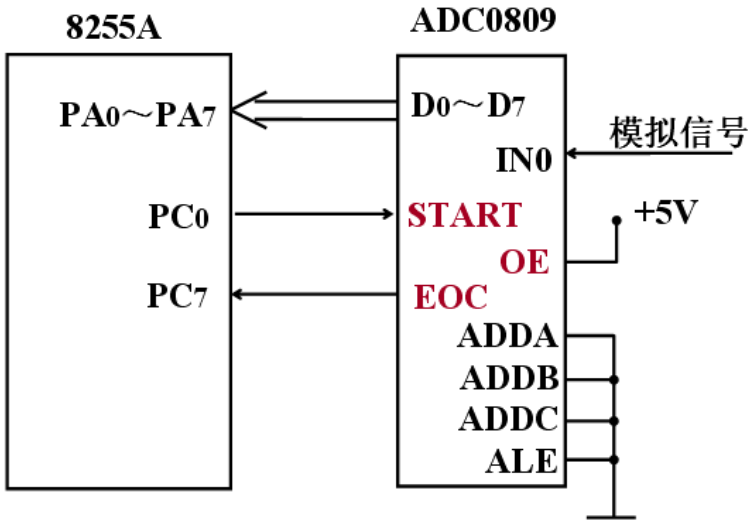
采用 8255A 的方式 0，端口 A 输入

PC0 接 ADC0809 的 START，用于启动转换

PC7 接 ADC0809 的 EOC，用于输入、判断 A/D 转换是否结束；采用查询方式

ADC0809 的 OE 接＋5V，这样可以随时读取 A/D 转换的结果

假设 8255A 的 A、B、C 和控制口地址依次为 portA～portD



；8255A 初始化

mov al,100110×0b

mov dx,portd

out dx,al

；使 PC0＝0（START 为低）

mov al,0000000b

mov dx,portd

out dx,al

；启动 A/D 转换

```
mov al,00000001b
mov dx,portd
out dx,al ；使 PC0＝1（START 为高）
nop
mov al,00000000b
out dx,al ；使 PC0＝0（START 为低）
；查询是否转换结束
          mov dx,portc
again:    in dx,al
          test al,80h
    ；PC7＝0（转换未结束，继续检测）
          jz again
    ；PC7＝1（转换结束）
；输入数据
     mov dx,porta
   in al,dx
```