微机原理与接口技术 楼顺天 周佳社编著 科学出版社

找了好久，终于在期末从老师那儿弄了一份，希望对大家有用！

微机原理与接口技术

楼顺天　周佳社 编著

科学出版社 2006 年

习题解答

因时间紧，习题解答由部分老师提供，还没有经过仔细校对，肯定有不少错误，请各位指正。另外，习题有多种解法，答案不唯一，仅供参考。

第 1 章 数制与码制

1. 将下列十进制数转换成二进制数：

（1）58；（2）67.625； （3）5721；

解：（1） 58D = 0011 1010B

（2） 67.625D = 0100 0011.1010B

（3） 5721D = 0001 0110 0101 1001B

2. 将二进制数变换成十六进制数：

（1）1001 0101B； （2）11 0100 1011B； （3）1111 1111 1111 1101B；

（4）0100 0000 10101B；（5）0111 1111B； （6）0100 0000 0001B

解： （1）1001 0101B = 95H

（2）11 0100 1011B = 34BH

（3）1111 1111 1111 1101B = FFFDH

（4）0 1000 0001 0101B = 815H

（5）0111 1111B = 7FH

（6）0100 0000 0001B = 401H

3. 将十六进制数变换成二进制数和十进制数：

（1）78H； （2）0A6H； （3）1000H； （4）0FFFFH

解：（1）78H = 120D = 0111 1000B

（2）0A6H = 166D = 1010 0110B

（3）1000H = 4096D = 0001 0000 0000 0000H

（4）0FFFFH = 65535D = 1111 1111 1111 1111B

4. 将下列十进制数转换成十六进制数：

（1）39； （2）299.34375； （3）54.5625

解：（1）39D = 27H

（2）299.34375D = 12B.58H

（3）54.5625D = 36.9H

5. 将下列二进制数转换成十进制数：

（1）10110.101B； （2）10010010.001B； （3）11010.1101B

解：（1）10110.101B = 22.625D

（2）10010010.001B = 146.125D

（3）11010.1101B = 26.8125D

6. 计算（按原进制运算）：

（1）10001101B＋11010B；（2）10111B＋11100101B；（3）1011110B－1110B；

（4）124AH＋78FH；（5）5673H＋123H；（6）1000H－F5CH；

解：（1）10100111B

（2） 11111100B

（3）1010000B

（4）19D9H

（5）5796H

（6）A4H

7. 已知 a=1011B，b=11001B，c=100110B，按二进制完成下列运算，并用十进制运算检查计算结果：

（1）a+b；（2）c-a-b；（3）a×b；（4）c÷b

解：a=1011B=11D b=11001B=25D c=100110B =38D

（1）100100B = 36D

（2）10B = 2D

（3）1 0001 0011B=275D

（4）1B 余 1101B= 13D

8. 已知 a=00111000B，b=11000111B，计算下列逻辑运算：

（1）a AND b；（2）a OR b；（3）a XOR b；（4）NOT a

解：（1）00000000B

（2）11111111B

（3）11111111B

（4）11000111B

9. 设机器字长为 8 位，写出下列各数的原码和补码：

（1）+1010101B；（2）-1010101B；（3）+1111111B；

（4）-1111111B；（5）+1000000B；（6）-1000000B

解：（1）原 01010101B 补 01010101B

（2）原 11010101B 补 10101011B

（3） 原 01111111B　补 01111111B

（4） 原 11111111B　补 10000001B

（5） 原 01000000B　补 01000000B

（6） 原 11000000B　补 11000000B

10. 写出下列十进制数的二进制补码表示（设机器字长为 8 位）：

（1）15；（2）－1；（3）117；（4）0；

（4）－15；（5）127；（6）－128；（7）80

解：（1） （00001111B）补

（2） （11111111B）补

（3） （01110101B）补

（4） （00000000B）补

（5） （11110001B）补

（6） （01111111B）补

（7） （10000000B）补

（8） （01010000B）补

11. 设机器字长为 8 位，先将下列各数表示成二进制补码，然后按补码进行运算，并用十进制数运算进行检验：

（1）87－73；（2）87＋（－73）；（3）87－（－73）；

（4）（－87）＋73；（5）（－87）－73；（6）（－87）－（－73）；

解：（1）1110B=14D

（2）00001110B 进位舍弃

（3）10100000B=-96D　溢出

（4）11110010B=-14D

（5）01100000B=96D 溢出

（6）11110010B=-14D

12. 已知 a, b, c, d 为二进制补码：a=00110010B, b=01001010B, c=11101001B, d=10111010B, 计算：

（1）a+b；（2）a+c；（3）c+b；（4）c+d；

（5）a-b；（6）c-a；（7）d-c；（8）a+d-c

解：（1）01111100B

（2）00011011B

（3）00110011B

（4）10100011B

（5）11101000B

（6）10110111B

（7）11010001B

（8）11B

13. 设下列四组为 8 位二进制补码表示的十六进制数，计算 a+b 和 a-b，并判断其结果是否溢出：

（1）a=37H, b=57H;　（2）a=0B7H, b=0D7H;

（3）a=0F7H, b=0D7H;（4）a=37H, b=0C7H

解：（1）a+b=8EH 溢出,a-b=E0H 未溢出

（2）8EH 未溢出,E0H 未溢出

（3）CEH 未溢出,20H 未溢出

（4）FEH 未溢出 70H 未溢出

14. 求下列组合 BCD 数的二进制和十六进制表示形式：

（1）3251（2）12907（3）2006

解：（1）0011 0010 0101 0001B = 3251H

（2）0001 0010 1001 0111B = 12907H

（3）0010 0000 0000 0110B = 2006H

15. 将下列算式中的十进制数表示成组合 BCD 码进行运算，并用加 6/减 6 修正其结果：

（1）38＋42；（2）56＋77；（3）99＋88；（4）34＋69；

（5）38－42；（6）77－56；（7）15－76；（8）89－23

解：（1） 0011 1000B + 0100 0010B = 0111 1010B 低 BCD 码位需要加 6 修正

0111 1010B + 0000 0110B = 1000 0000B = 80BCD

（2） 0101 0110B + 0111 0111B = 1100 1101B 高、低 BCD 码位都需要加 6 修正

1100 1101B + 0110 0110B = 0001 0011 0011B=133BCD

（3） 1001 1001B+1000 1000B = 0001 0010 0001B 高、低 BCD 码位都需要加 6 修正

0001 0010 0001B +0110 0110B = 0001 1000 0111B=187BCD

（4） 0011 0100B + 0110 1001B = 1001 1101B 低 BCD 码位需要加 6 修正

1001 1101B + 0000 0110B = 1010 0011B 修正结果使高 BCD 码位需要加 6 修正

1010 0011B +0110 0000B = 0001 0000 0011B = 103BCD

（5） 00111000B-01000010B =（-1）1111 0110B 高 BCD 码位需要减 6 修正

（-1）1111 0110B -0110 0000B =（-1）1001 0110B=-100+96=-4BCD

（6）01110111B-01010110B = 0010 0001B = 21BCD

（7） 00011001B-01110110B =（-1）1001 1111B 高、低 BCD 码位都需要减 6 修正

（-1）1001 1111B -01100110B =（-1）0011 1001B = -100+39 = -61BCD

（8）10001001B-00100011B = 0110 0110B = 66BCD

16. 将下列字符串表示成相应的 ASCII 码（用十六进制数表示）：

（1）Example 1;（2）XiDian University;（3）-108.652;

（4）How are you?;（5）Computer（6）Internet Web

解：（1）45H，78H,61H,6DH,70H,6CH,65H，20H，31H

（2） 58H,69H,44H,69H,61H,6EH,20H,55H,6EH,69H,76H,65H，72H,73H,69H,74H,79H

（3）2DH,31H,30H,38H,2EH,36H,35H,32H

（4）48H,6FH,77H,20H,61H72H,65H,20H79H,6FH,75H

（5）43H,6FH,6DH,70H,75H,74H,65H,72H

（6）49H,6EH,74H,65H72H,6EH,65H,74H,20H,57H,65H,62H

17. 将下列字符串表示成相应的 ASCII 码（用十六进制数表示）：

（1）Hello（2）123<CR>456;（注：<CR>表示回车）（3）ASCII;

（4）The number is 2315

解：（1）48H,65H,6CH,6CH,6FH （2）31H,32H,33H,0DH,34H,35H,36H

（3）41H,53H,43H,49H,49H （4）54H,68H,65H,20H,6EH,75H,6DH,62H,65H,72H

第 2 章 8086 CPU 结构与功能

1. 微处理器内部结构由哪几部分组成？阐述各部分的主要功能。

解：微处理器内部结构由四部分组成：

（1）算术逻辑运算单元 ALU：完成所有的运算操作；

（2）工作寄存器：暂存寻址信息和计算过程中的中间结果；

（3）控制器：完成指令的读入、寄存和译码，并产生控制信号序列使 ALU 完成指定操作；

（4）I/O 控制逻辑：处理 I/O 操作。

2. 微处理器级总线有哪几类？各类总线有什么作用？

解：微处理器级总线有三类：

（1）数据总线：传送信息；

（2）地址总线：传送地址码；

（3）控制总线 传送控制信号。

3. 为什么地址总线是单向的，而数据总线是双向的？

解：地址码只能由 CPU 生成。而数据需要在 CPU 和存储器之间传输。

4. 8086/8088 微处理器内部有哪些寄存器？其主要作用是什么？

解：8086CPU 内部有 14 个 16 位寄存器，其中 8 个通用寄存器（4 数据寄存器 AX、BX、CX、DX，4 地址指针/变址寄存器 SI、DI、SP、BP），4 个段寄存器（CS、DS、ES、SS），2 个控制寄存器（指令指针 IP，微处理器状态字 PSW）。

应该注意的是：可以在指令中用作为地址指针的寄存器有：SI、DI、BP 和 BX；在微处理器状态字 PSW 中，一共设定了 9 个标志位，其中 6 个标志位用于反映 ALU 前一次操作的结果状态（CF，PF，AF，ZF，SF，OF），另 3 个标志位用于控制 CPU 操作（DF，IF，TF）。

5. 如果某微处理器有 20 条地址总线和 16 条数据总线：

（1）假定存储器地址空间与 I/O 地址空间是分开的，则存储器地址空间有多大？

（2）数据总线上传送的有符号整数的范围有多大？

解：（1）存储器地址空间为：$2^{20}=1MB$

（2）有符号数范围为： $-2^{15}\sim 2^{15}-1$， 即 −32768～32767

6. 将十六进制数 62A0H 与下列各数相加，求出其结果及标志位 CF、AF、SF、ZF、OF

和 PF 的值：

（1） 1234H；（2） 4321H；（3） CFA0H；（4） 9D60H

解：（1） 74D4H CF=0 AF=0 SF=0 ZF=0 OF=0 PF=1

（2） A5C1H CF=0 AF=0 SF=1 ZF=0 OF=1 PF=0

（3） 3240H CF=1 AF=0 SF=0 ZF=0 OF=0 PF=0

（4） 0000H CF=1 AF=0 SF=0 ZF=1 OF=0 PF=1

7. 从下列各数中减去 4AE0H，求出其结果及标志位 CF、AF、SF、ZF、OF 和 PF 的值：

（1）1234H；（2）5D90H；（3）9090H；（4）EA04H

解：（1） C754H CF=1 AF=0 SF=1 ZF=0 OF=0 PF=0

（2） 12B0H CF=0 AF=0 SF=0 ZF=0 OF=0 PF=0

（3） 45B0H CF=0 AF=0 SF=0 ZF=0 OF=1 PF=0

（4） 9F24H CF=0 AF=0 SF=1 ZF=0 OF=0 PF=1

9. 写出下列存储器地址的段地址、偏移地址和物理地址：

（1）2134：10A0；（2）1FA0：0A1F；（3）267A：B876

解：物理地址＝段地址*10H+偏移地址

（1）段地址：2134H，偏移地址：10A0H，物理地址：223E0H

（2）段地址：1FA0H，偏移地址：0A1FH，物理地址：2041FH

（3）段地址：267AH，偏移地址：B876H，物理地址：32016H

10. 给定一个数据的有效地址为 2359H，并且（DS）＝490BH，求该数据的物理地址。

解：物理地址＝段地址*10H+偏移地址

物理地址＝490BH ＋2359H ＝ 4B409H

11. 如果在一个程序段开始执行之前，（CS）＝0A7F0H，（IP）＝2B40H，求该程序段的第一个字的物理地址。

解：物理地址＝段地址*10H+偏移地址

物理地址＝CS*10H+IP＝AAA40H

12. IBM PC 有哪些寄存器可用来指示存储器的地址？

解：变址寄存器 SI，DI，堆栈指针 SP，BP，另外还有 BX。

第 3 章 8086CPU 指令系统

1. 写出完成下列要求的变量定义语句：

（1）在变量 var1 中保存 6 个字变量：4512H，4512，－1，100/3，10H，65530；

（2）在变量 var2 中保存字符串：’ BYTE’，’ word’，’ WORD’；

（3）在缓冲区 buf1 中留出 100 个字节的存储空间；

（4）在缓冲区 buf2 中，保存 5 个字节的 55H，再保存 10 个字节的 240，并将这一过程重复 7 次；

（5）在变量 var3 中保存缓冲区 buf1 的长度；

（6）在变量 pointer 中保存变量 var1 和缓冲区 buf1 的偏移地址。

解：var1 DW 4512H,4512,-1,100/3,10H,65530

var2 DB ’BYTE’,’word’,’WORD’

buf1 DB 100 DUP（?）

buf2 DB 7 DUP（5 DUP（55H）,10 DUP（240））

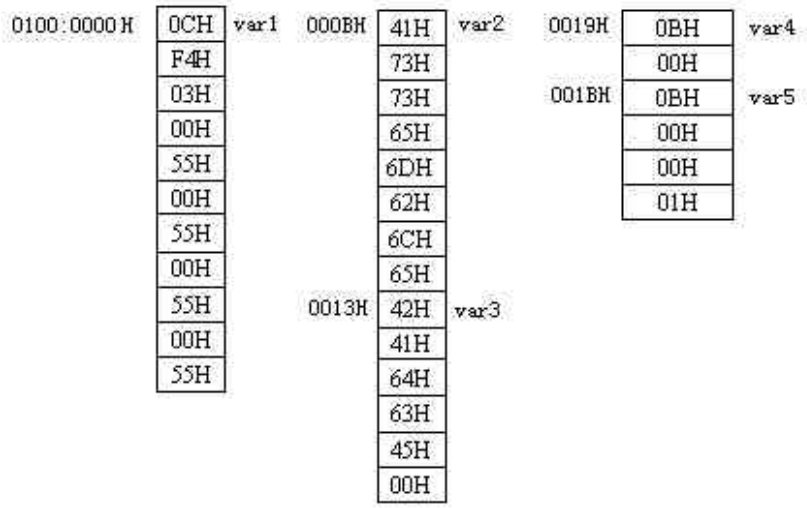
var3 DB LENGTH buf1

pointer DW var1,buf1 （或者 pointerDW OFFSET var1，OFFSET buf1）

2. 设变量 var1 的逻辑地址为 0100：0000，画出下列语句定义的变量的存储分配图：

```
var1  DB  12，－12，20/6，4 DUP（0，55H）
var2  DB  ‘Assemble’
var3  DW  ‘AB’，‘cd’，‘E’
var4  DW  var2
var5  DD  var2
```

解：



3. 指令正误判断，对正确指令写出源和目的操作数的寻址方式，对错误指令指出原因（设 VAR1, VAR2 为字变量, L1 为标号）：

（1）MOV SI，100　　（2）MOV BX，VAR1[SI]

（3）MOV AX, [BX]　　（4）MOV AL, [DX]

（5）MOV BP, AL　　（6）MOV VAR1, VAR2

（7）MOV CS, AX　　（8）MOV DS, 0100H

（9）MOV [BX][SI], 1　　（10）MOV AX, VAR1+VAR2

（11）ADD AX, LENGTH VAR1　　（12）OR BL, TYPE VAR2

（13）SUB [DI], 78H　　（14）MOVS VAR1, VAR2

（15）PUSH 100H　　（16）POP CS

（17）XCHG AX, ES　　（18）MOV DS, CS

（19）JMP L1+5　　（20）DIV AX, 10

（21）SHL BL, 2　　（22）MOV AL, 15+23

（23）MUL CX　　（24）XCHG CL, [SI]

（25）ADC CS:[0100], AH　　（26）SBB VAR1－5,154

解：（1）MOV SI,100　　正确。源：立即数寻址， 目的：寄存器寻址

（2）MOV BX,VAR1[SI]　　正确。源：寄存器相对寻址， 目的：寄存器寻址

（3）MOV AX,[BX]　　正确。源：寄存器间接寻址，目的：寄存器寻址

（4）MOV AL,[DX]　　错误。寄存器间接寻址时，DX, AX, CX 不能作地址寄存器

（5）MOV BP,AL　　错误。操作数类型不一致

（6）MOV VAR1,VAR2　　错误。两存储单元之间不能用 MOV 指令传送数据

（7）MOV CS,AX　　错误。CS 不能为目的操作数

（8）MOV DS,0100H　　错误。目的操作数为段寄存器时，源操作数不能为立即数

（9）MOV [BX][SI]，1　　错误。指令类型不定。

（10）MOV AX,VAR1+VAR2　　错误。MOV 指令中不能完成加法运算

（11）ADD AX,LENGTH VAR1　正确。源：立即数寻址。目的：寄存器寻址

（12）OR　BL,TYPE VAR2　　正确。源：立即数寻址。目的：寄存器寻址

（13）SUB [DI],78H　　错误。指令类型不定

（14）MOVS VAR1,VAR2　正确。目的、源均为隐含寻址。操作数仅指出操作数类型

（15）PUSH　100H　　错误。将常数压入堆栈，要通过寄存器来实现

（16）POP　　CS　　错误。目的操作数不能为 CS

（17）XCHG　AX，ES　　错误。XCHG 指令的操作数不能是段寄存器

（18）MOV　DS，CS　　错误。MOV 指令不能从段寄存器到段寄存器

（19）JMP　L1+5　　正确。段内直接转移

（20）DIV　　AX，10　　错误。指令格式错误。

（21）SHL　　BL，2　　错误。移位指令的移位数为 1 或者 CL

（22）MOV　AL，15+23　正确。源:立即数寻址，目的:寄存器。编译时就处理为 38

（23）MUL　CX　　正确。源：寄存器寻址，目的：寄存器寻址

（24）XCHG　CL，[SI]　正确。源：寄存器间接寻址，目的：寄存器寻址

（25）ADC　CS:[0100],AH　正确。源:寄存器寻址,目的: 直接寻址（数据在代码段中）

（26）SBB　　VAR1-5,154　正确。源：立即数寻址，目的：直接寻址。

4.　说明下列指令对的区别：

（1）MOV AX，VAR1　　与 MOV AX，OFFSET VAR1

（2）MOV AX，VAR2　　与 LEA　AX，VAR2

（3）MOV AL，LENGTH　VAR1　与 MOV AL，SIZE　VAR1

（4）MOV AL，ES:［DI］　　CMP　AL,［SI］与　CMPSB

（5）SHR AL，1　　与 SAR AL，1

（6）SHR AL，1　　与 ROR AL，1

（7）ROL BX，1　　与 RCL BX，1

解：（1）MOV　　AX，VAR1　　把变量 VAR1 对应地址单元中的一个字送入 AX

MOV　　AX，OFFSET VAR1　　把 VAR1 的有效地址的偏移地址送入 AX

（2）MOV　　AX，VAR2　　把变量 VAR2 对应地址单元中的一个字送入 AX

LEA　AX，VAR2　　把 VAR2 的有效地址的偏移地址送入 AX

（3）MOV AL，LENGTH VAR1　　把变量 VAR1 的长度送入 AL

MOV AL，SIZE VAR1　　把变量 VAR1 的大小送入 AL

（4）MOV　　AL，ES：[DI]

CMP　AL，[SI]　　把以 ES 为段地址，DI 为偏移地址的一个字节送入 AL，并与以 SI 内容为偏移地址的一个字节作比较，改变标志寄存器内容。（相当于作 ES：（DI）与（DS: （SI）内容比较）

CMPSB　　对字符串中的一字节比较。寻址方式隐含。源串的地址由 DS:SI 指定，目的串的地址由 ES:DI 指定。（相当于作 DS: （SI）与 ES：（DI）内容比较）

（5）SHR AL，1　AL 逻辑右移 1 位，最高位移入 0, 最低位移入 CF。

SAR AL，1　AL 算术右移 1 位，以最高位内容移入，最低位移入 CF, 其余各位右移一位。

（6）SHR AL，1　AL 逻辑右移 1 位，最高位移入 0, 最低位移入 CF。

ROR AL，1　AL 的各位构成环形移位，右移一位，最低位内容同时移入到 CF 和最高位。

（7）ROL BX，1　BX 各位构成环形移位，左移一位，最高位内容同时移入到 CF 和最低位。

RCL BX，1　BX 和 CF 构成环形移位，左移一位，CF 内容移入到最低位，最高位移入 CF。

5. 写出下列转移指令的寻址方式（设 L1 为标号，VAR1 为字型变量，DVAR1 为双字型变量）：

（1）JMP　L1　　（2）JMP NEAR L1

（3）JNZ　L1　　（4）JMP　BX

（5）JG　L1　　（6）JMP　VAR1[SI]

（7）JMP　FAR PTR L1　　（8）JMP　DVAR1

解：（1）JMP L1　段内直接寻址　　（2）JMP NEAR PTR L1　段内直接寻址

（3）JNZ　　L1　段内直接寻址　　（4）JMP BX　　段内间接寻址

（5）JG　L1　段内直接寻址　　（6）JMP VAR1[SI]　　段内间接寻址

（7）JMP FAR PTR L1　段间直接寻址　　（8）JMP DVAR1　段间间接寻址

6. 设（DS）＝2000H，（BX）＝0100H，（SI）＝0002H，（20100）＝3412H，（20102）＝7856H，（21200）＝4C2AH，（21202）＝65B7H，求下列指令执行后 AX 寄存器的内容：

（1）MOV AX，1200H；　　（2）MOV AX，BX；　　（3）MOV AX，[1200H]；

（4）MOV AX，[BX]；　　（5）MOV AX,1100[BX]；（6）MOV AX，[BX][SI]；

（7）MOV AX,1100[BX][SI]

解：（1）1200H　（2） 0100H　（3）4C2AH　（4）3412H　（5）4C2AH

（6）7856H　（7）65B7H

7. 执行下列指令后，DX 寄存器中的内容是多少？

```
TABLE    DW  25，36，－1，－16，10000，13
PYL      DW  7
……
MOV  BX，OFFSET TABLE
ADD  BX，PYL
MOV  DX，[BX]
```

解：DX＝10FFH　由-16（FFF0H）的高 8 位和 10000（2710H）的低 8 位构成

8. 如果堆栈的起始地址为 2200：0000，栈底为 0100H，（SP）＝00A8H，求

（1）栈顶地址；

（2）SS 的内容；

（3）再存入数据 5678H，3AF2H 后，SP 的内容。

解：栈顶地址 00A8H， SS＝2200H， 再存入 2 个字后，SP＝00A4H

9. 设已用伪指令 EQU 定义了 4 个标识符：

```
N1  EQU  2100
N2  EQU  10
N3  EQU  20000
N4  EQU  25000
```

下列指令是否正确？并说明原因。

（1）ADD AL，N1－N2；　　（2）MOV AX，N3＋N4；

（3）SUB BX，N4－N3；　　（4）SUB AH，N4－N3－N1；

（5）ADD AL，N2；　　（6）MOV AH，N2*N2

解：（1）错误。N1-N2=2090>255　　（2）正确　　（3）正确

（4）错误。N4-N3-N1=2900>255　　（5）正确　　（6）正确

10. 按下列要求写出指令：

（1）将 AX 寄存器的低 4 位清零，其余位不变；

（2）将 BX 寄存器的低 4 位置 1，其余位不变；

（3）将 AL 寄存器的低 4 位保持不变，高 4 位取反；

（4）测试 BX 中的位 1 和位 2，当这两位同时为 0 时将 AL 置 0FFH，否则 AL 清零；

（5）测试 BX 中的位 1 和位 2，当这两位有一位为 0 时将 AL 置 0FFH，否则 AL 清零；

（6）将 AL 中保存的字母 ASCII 码变换成相应的大写字母的 ASCII 码；

（7）将 AL 中保存的字母 ASCII 码变换成相应的小写字母的 ASCII 码；

（8）将 AX 中的各位取反；

（9）将 DX 中的低 7 位取反，高 9 位不变；

（10）将 CX 中的低 8 位与高 8 位互换。

解：（1）AND　AX，0FFF0H

（2）OR　BX，000FH

（3）XOR　　AL，0F0H

（4）

```
        TEST    BX，06H
        JZ      ZERO
        MOV     AL，00H
        JMP     OVER
ZERO:   MOV     AL，0FFH
OVER:
```

（5）

```
        MOV     AX，BX
        AND     AX，06H
        XOR     AX，06H
        JZ      OVER
        MOV     AL，0FFH
OVER:
```

（6）AND AL，5FH

或者：

```
    CMP     AL，61H
    JL      OVER        （无需变换或不是字母）
    CMP     AL，7AH
    JG      OVER        （不是字母）
    AND AL，5FH     或  SUB  AL，20H
OVER:
```

（7）OR AL，20H

或者：

CMP　　AL，41H

JL　　OVER　　（不是字母）

CMP　　AL，5AH

JG　　OVER　　（无需变换或不是字母）

OR　AL，20H　　或　ADD　AL，20H

OVER：

（8）XOR　AX，0FFFFH　　或者　　NOT　　AX

（9）XOR　DX，007FH

（10）XCHG　CH，CL

11. 写出完成下述功能的程序段：

（1）传送 40H 到 AL 寄存器；

（2）将 AL 的内容乘以 2；

（3）传送 16H 到 AH 寄存器；

（4）AL 的内容加上 AH 的内容。

计算最后结果（AL）＝？

解：（1）MOV　AL,40H

（2）SHL　AL,1

（3）MOV　AH,16H

（4）ADD　AL，AH

AL=96H

12. 写出完成下述功能的程序段：

（1）从缓冲区 BUF 的 0004 偏移地址处传送一个字到 AX 寄存器；

（2）将 AX 寄存器的内容右移 2 位；

（3）将 AX 内容与 BUF 的 0006 偏移地址处的一个字相乘；

（4）相乘结果存入 BUF 的 0020H 偏移地址处（低位在前）。

解：　（1）　LEA　SI, BUF

MOV AX, [SI+4]

(2) SHR AX,1

SHR AX,1

(3) MUL WORD PTR 6[SI]

(4) MOV 20H[SI],AX

MOV 22H[SI],DX

13. 设（BX）＝11001011B，变量 VAR 的内容为 00110010B，求下列指令单独执行后 BX 的内容：

（1）XOR BX，VAR；　　（2）AND BX，VAR；

（3）OR BX，VAR；　　（4）XOR BX，11110000B；

（5）AND BX，00001111B；　　（6）TEST BX，1

解：（1）00F9H

（2）0002H

（3）00FBH

（4）003BH

（5）000BH

（6）00CBH

14. 设（DX）＝10111011B，（CL）＝3，（CF）＝1，求下列指令单独执行后 DX 的内容：

（1）SHR DX，1；（2）SAR DX，CL；　　（3）SHL DX，CL；

（4）SHL DX，1；（5）ROR DX，CL；　　（6）ROL DL，CL；

（7）SAL DH，1；（8）RCL DX，CL；　　（9）RCR DL，1

解：DX= 0000 0000 1011 1011B　CF=1　CL=3

（1）SHR　DX，1　　DX 逻辑右移 1　0000 0000 0101 1101B ＝　005DH

（2）SAR DX，CL　　DX 算术右移 3　0000 0000 0001 0111B ＝　0017H

（3）SHL DX，CL　　DX 逻辑左移 3　0000 0101 1101 1000B ＝　05D8H

（4）SHL DX，1　　DX 逻辑左移 1　0000 0001 0111 0110B ＝　0176H

（5）ROR　DX，CL　　DX 循环右移 3　0110 0000 0001 0111B ＝　6017H

（6）ROL　DL，CL　　DL 循环左移 3　0000 0000 1101 1101B ＝　00DDH

（7）SAL DH，1　　DH 算术左移 1　0000 0000 1011 1011B ＝　00BBH

（8）RCL DX，CL　　DX 带进位循环左移 3　　0000 0101 1101 1100B ＝　05DCH

（9）RCR　　DL，1　DL 带进位循环右移 1　　0000 0000 1101 1101B ＝　00DDH

15. 选择题（各小题只有一个正确答案）

（1）执行下列三条指令后：

```
MOV SP，1000H
PUSH AX
CALL BX
```

a.　（SP）＝1000H；　　b.　（SP）＝0FFEH；

c.　（SP）＝1004H；　　d.　（SP）＝0FFCH；

（2）要检查寄存器 AL 中的内容是否与 AH 相同，应使用的指令为：

a.　AND AL, AH　　b.　OR AL, AH

c.　XOR AL, AH　　d.　SBB AL, AH

（3）指令 JMP NEAR PTR L1 与 CALL L1（L1 为标号）的区别在于：

a. 寻址方式不同；　　b.　是否保存 IP 的内容；

c. 目的地址不同；　　d.　对标志位的影响不同。

解：（1）D　PUSHU AX 则 AX 入栈，SP=0FFEH；CALL BX 则 IP 入栈，SP=0FFCH

（2）C　异或，若相同，则 AL=0，ZF＝1。

（3）B

16. 寄存器 DX：AX 组成 32 位数，DX 为高位，编写程序段实现：

（1）DX：AX 右移 3 位，并将移出的低 3 位保存在 CL 中；

（2）DX：AX 左移 3 位，并将移出的高 3 位保存在 CL 中；

解：（1）移出的 3 位应该按时序移入 CL 中。

```
    XOR CL,CL
    MOV BL,3
L1: SHR  DX，1
    RCR  AX，1
    RCL  CL，1
    DEC  BL
    JNZ  L1
```

（2）移出的 3 位应该按时序移入 CL 中。

```
    XOR CL,CL
```

```
        MOV BL,3
L1:     SHL   AX，1
        RCR   DX，1
        RCR   CL，1
        DEC   BL
        JNZ   L1
```

17. 编写程序段实现将 BL 中的每一位重复 4 次，构成 32 位的双字 DX：AX，例如当 BL＝01011101B 时，则得到的（DX）＝0F0FH，（AX）＝0FF0FH。

解：算术右移时，移入的值就是最高位本身，这样可以使位内容重复，利用这一点可以实现题目的要求。

```
        XOR     DX,DX
        XOR     AX,AX
        MOV     CX,4
L1:     SHR     BL,1
        RCR     AX,1
        SAR     AX,1
        SAR     AX,1
        SAR     AX,1
        LOOP    L1
        MOV     CX,4
L2:     SHR     BL,1
        RCR     DX,1
        SAR     DX,1
        SAR     DX,1
        SAR     DX,1
        LOOP    L2
```

18. 字变量 VAR1 中保存有小于 38250 的 16 位无符号数，编写程序段实现 VAR1÷150，并进行四舍五入操作，将商保存在字节变量 VAR2 中。

解：根据题意，38250÷150＝255，因此商不会超过 255，可以用一个字节表示。

a÷b 的四舍五入操作可以通过判断除后余数实现：余数大于等于除数的一半，则商加 1；

否则不用加 1。但这种方法用汇编语言编程实现时比较复杂，这里介绍另外一种方法：设 a ÷b 的四舍五入后的结果为 c，用『』表示取整数操作，则

$$c=\left[\frac{a}{b}+0.5\right]=\left[\frac{a+b/2}{b}\right]$$

这种方法是在除法操作之前，在被除数上加上除数的一半，这样除法操作后得到的值就是考虑了四舍五入的商。

```
VAR1    DW   12345
VAR2    DB   ?
DATAA   DB   150

MOV   AX,VAR1
XOR   BX,BX
MOV   BL,DATAA
SHR   BX,1
ADD   AX,BX
DIV   DATAA
MOV   VAR2,AL
```

19. 有一组无符号的 16 位数据保存在 BUFFER 中，前两个字节存放数据的个数，编程实现按下式进行滤波处理：

$$y(k)=\frac{1}{3}\left(x(k)+x(k-1)+x(k-2)\right) \qquad k\geq 2$$
$$y(k)=x(k) \qquad k<2$$

解：滤波结果保存在 FILT 中。

```
BUFFER DW 0CH
       DW 33H, 18H, 1BH, 06H, 33H, 08H
       DW 3H, 6H, 0FH, 51H, 05H, 0CH
FILT DW 100H DUP（?）

    LEA SI,BUFFER
    LEA DI,FILT
```

```
        MOV CX,[SI]
        MOV [DI],CX
        ADD SI,2
        ADD DI,2
        XOR DX,DX
        MOV AX,[SI]
        MOV [DI],AX
        MOV BX,2[SI]
        MOV 2[DI],BX
        ADD SI,4
        ADD DI,4
        DEC CX
        DEC CX
        ADD AX,BX
        ADC DX,0
        MOV BX,3
L1:
        ADD AX,[SI]
        ADC DX,0
        PUSH DX
        PUSH AX
        DIV BX
        MOV [DI],AX
        POP AX
        POP DX
        SUB AX, [SI-4]
        SUBB DX,0
        ADD DI,2
        ADD SI,2
        LOOP L1
```

20. 在由字符串构成的缓冲区 BUFFER 中，前 2 个字节存放字符个数，后续每个字节存放一个字符的 ASCII 码。编写程序实现将字符串‘2004’替换成‘2006’。

解：在数据段中定义：

```
BUFFER    DW 74
          DB ‘This year is 2004. In 2004, we have a plan for reducing annual expensive 10%’
DEST      DB ‘2004’
```

在代码段中编写程序段：

```
          CLD
          LEA     SI, BUFFER
          MOV     CX,[SI]
          ADD     SI,2
          LEA     DI,DEST
L1:       PUSH    SI
          PUSH    DI
          PUSH    CX
          MOV     CX,4
          REPZ    SCASB
          JNZ     L2
          MOV     BYTE PTR [SI-1],’6’
L2:       POP     CX
          POP     DI
          POP     SI
          INC     SI
          INC     DI
          LOOP    L1
```

21. 定义有下列宏指令：

```
WAGS    MACRO    S1,S2,S3
            SUB AX,AX
```

```
        MOV DX,AX
        ADD AX,S1
        ADD AX,S2
        ADC DX,0
        ADD AX,S3
        ADC DX,0
        ENDM
```

当采用宏调用指令“WAGS 60000，25000，3000”时，执行后 DX＝____ AX＝_____。

解：宏指令 WAGS 完成的功能为 S1+S2+S3，结果放在 DX:AX 中。所以，调用“WAGS 60000,25000,3000”时，其结果为 DX=0001H ，AX=57C0H

22. 对上题定义的宏指令，如果采用宏调用指令“WAGS BX,CX,SI”时，写出宏展开形式。

解：调用“WAGS BX,CX,SI”时，宏展开形式：

```
SUB AX,AX
MOV DX,AX
ADD AX,BX
ADD AX,CX
ADC DX,0
ADD AX,SI
ADC DX,0
```

23. 写出宏指令 SUMMING，实现将字节缓冲区 array 中的内容求校验和（保留低 8 位），并保存在 VALUE 中。

解：设 array 前两个字节保存缓冲区字节数，在宏指令 SUMMING，将 array 和 VALUE 作为形式参数。

```
SUMMING MACRO array,VALUE
        LEA     SI,array
        MOV     CX,[SI]
        ADD     SI,2
        XOR     AL,AL
L1:     ADD     AL,[SI]
        INC     SI
```

```
LOOP    L1
MOV     VALUE,AL
ENDM
```

第 4 章 汇编语言程序设计

1. 已知在 BUF 的起始处保存有 N 个字符的 ASCII 码，编写汇编语言程序实现，将这组字符串传送到缓冲区 BUFR 中，并且使字符串的顺序与原来的顺序相反。

解：
```
BUF     DB "BONJOUR_BELLE"
BUFR    DB 100 DUP(?)

        MOV CX, N
        LEA SI, BUF
        LEA DI, BUFR
        ADD DI,CX
        DEC DI
L1:
        MOV AL,[SI]
        MOV [DI],AL
        INC SI
        DEC DI
        LOOP L1
```

2. 利用移位、传送和相加指令实现 AX 的内容扩大 10 倍。

解：将扩大后的结果放在 DX：AX 中，注意到 10×AX＝8×AX＋2×AX。

```
XOR     DX,DX
SHL     AX, 1
RCL     DX, 1
MOV     BX,AX
MOV     CX,DX
SHL     AX, 1
RCL     DX, 1
SHL     AX, 1
RCL     DX, 1
```

```
ADD     AX, BX
ADC     DX, CX
```

3. 在缓冲区 VAR 中连续存放着 3 个 16 位的无符号数，编写程序实现将其按递增关系排列；如果 VAR 中保存的为有符号数，则再编写程序实现将其按递减关系排列。

解：

```
VAR DW 1236，-432，3900
    XOR   SI,,SI
    MOV   AX,VAR[SI]
    CMP   AX, VAR[SI+2]
    JAE   L1
    XCHG   AX, VAR[SI+2]
L1:
    CMP   AX, VAR[SI+4]
    JAE   L2
    XCHG   AX, VAR[SI+4]
L2:
    MOV   VAR[SI], AX
    MOV   AX,VAR[SI+2]
    CMP   AX, VAR[SI+4]
    JAE   L3
    XCHG   AX, VAR[SI+4]
L3:
    MOV   VAR[SI+2], AX
```

4. 编写程序段实现将 AL 和 BL 中的每一位依次交叉，得到的 16 位字保存在 DX 中，例如（AL）＝01100101B，（BL）＝11011010B，则得到的（DX）＝10110110 10011001B。

解：利用移位指令完成。

```
    XOR DX,DX
    MOV CX,8
L1:
    SHR   AL,1
    RCR   DX,1
```

```
        SHR    BL,1
        RCR    DX,1
        LOOP   L1
```

5. 在变量 VAR1 和 VAR2 中分别保存有两个字节型的正整数，编写完整的汇编语言程序实现：

（1）当两数中有一个奇数时，将奇数存入 VAR1，偶数存入 VAR2；

（2）当两数均为奇数时，两个变量的内容不变；

（3）当两数均为偶数时，两数缩小一倍后存入原处。

解：当 VAR1 为奇数时，不论 VAR2 的奇偶性，这两个单元的内容均不变；只有当 VAR1 为偶数时，如果 VAR2 为奇数，则 VAR1 与 VAR2 内容交换；如果 VAR2 为偶数，则两数缩小一倍后存入原处。

```
DATA    SEGMENT
        VAR1    DB 28
        VAR2    DB 36
DATA    ENDS
CODE SEGMENT
            ASSUME CS:CODE, DS:DATA, ES:DATA
        START:
            MOV    AX, DATA
            MOV    DS,AX
            MOV    ES, AX
            MOV    AL,VAR1
            MOV    BL,VAR2
            TEST   AL,1
            JZ     EVEN1
            JMP    OVER
  EVEN1:
            TEST   BL,1
            JZ     EVEN2
            MOV    VAR1,BL
```

```
        MOV     VAR2,AL
        JMP     OVER
EVEN2:
        SHR     AL,1
        MOV     VAR1,AL
        SHR     BL,1
        MOV     VAR2,BL
OVER:
        MOV     AH,4CH
        MOV     AL,0
        INT     21H
CODE    ENDS
        END   START
```

6. 已知在字变量 VAR1、VAR2 和 VAR3 中保存有 3 个相同的代码，但有一个错码，编写程序段找出这个错码，并将它送到 AX，其地址送 SI；如果 3 个代码都相同，则在 AX 中置－1 标志。

解：在数据段中定义：

```
VAR1 DW 5A34H
VAR2 DW 5A35H
VAR3 DW 3A34H
```

在代码段中编写程序段：

```
MOV   AX,-1
MOV   BX,VAR1
CMP   BX,VAR2
JZ    L2
CMP   BX,VAR3
JZ    L1
MOV   AX,BX
LEA   SI,VAR1
JMP   OVER
```

```
L1:
        MOV    AX,VAR2
        LEA    SI,VAR2
        JMP    OVER
L2:
        CMP    BX,VAR3
        JZ     OVER
        MOV    AX,VAR3
        LEA    SI,VAR3
OVER:
```

7. 分析下列程序段的功能：

```
MOV CL，04
SHL DX，CL
MOV BL，AH
SHL AX，CL
SHR BL，CL
OR DL，BL
```

解：程序段完成 DX:AX 组成的 32 位无符号数左移 4 位，低位补零（也即除以 16）。

8. 下列程序段执行后，求 BX 寄存器的内容：

```
MOV CL，3
MOV BX，0B7H
ROL BX，1
ROR BX，CL
```

解：实际上完成 BX 内容循环右移 2 位，因此，BX 寄存器的内容为 C02DH。

9. 下列程序段执行后，求 BX 寄存器的内容：

```
MOV CL，5
MOV BX，7D5CH
SHR BX，CL
```

解：完成 BX 内容逻辑右移 5 位，因此，BX 寄存器的内容为 03EAH。

10. 将 BUFFERS 中 N 个字按相反顺序传递到 BUFFERT 中。

解：

```
        LEA    SI,BUFFERS
        LEA    DI,BUFFERT
        MOV    CX,N
        ADD    DI,N
        ADD    DI,N
        SUB    DI,2
    L1:
        MOV    AX,[SI]
        MOV    [DI],AX
        ADD    SI,2
        SUB    DI,2
        LOOP L1
```

11. 数组 ARRAY 中存放有一组字型数据，前两个字节存放数据长度（5 的倍数）。为给这个数组中的数据进行加密保护，每 5 个数据取出一个数据进行加密处理：奇数位进行取反，偶数位不变，例如对数据 0110 1100 1011 0001B 加密后变成 1100 0110 0001 1011B，编写加密程序 encrpytion 和解密程序 unencrpytion 。

解：约定从第一个数据开始，每 5 个数据为一组，每组中的第一个数据采取加密/解密处理。由于加密算法采用的是取反操作，解密算法也采用取反操作，因此解密和解密算法是同一个程序。

```
    ENCRPYTION    PROC NEAR
            LEA    SI,ARRAY
            XOR    DX,DX
            MOV    AX,[SI]
            MOV    BX,5
            DIV    BX
            MOV    CX, AX
            ADD SI, 2
        L1:
```

```
        MOV    AX, [SI]
        XOR    AX,0AAAAH
      MOV    [SI], AX
      ADD SI,10
      LOOP    L1
      RET
ENCRPYTION ENDP
```

13. 设 BUF 中存放有 N 个无符号数（或有符号数），编程实现求它们的最小值（存入 AX）和最大值（存入 DX）。

解：BUF 存放有 N 个无符号数的程序如下：

```
    MOV    CX,N
    LEA    SI,BUF
    MOV    AX,[SI]
    MOV    DX,AX
    ADD SI,2
L1:
    CMP AX,[SI]
    JBE    NOCHG1
    XCHG AX,[SI]
NOCHG1:
    CMP DX,[SI]
    JAE    NOCHG2
    XCHG DX,[SI]
NOCHG2:
    ADD SI,2
    LOOP    L1
```

如果 BUF 中存放的是有符号数，则只需要将程序中的两行内容修改：

```
    JBE    NOCHG1        改成：JLE    NOCHG1
    JAE    NOCHG2        改成：JGE    NOCHG2
```

14. 设 BUFFER 中存放有 N 个无符号（第 1 个字节存放缓冲区的长度），编程实现将其中的 0 元素抹去，并更新其长度。

解：设 BUFFER 中存放的是字节型数据。采用双指针方法：SI 为读指针，DI 为写指针，从低地址开始，内存中读出一个字节，如果不为 0，则写入内存；如果为 0，则不进行写操作。

```
        LEA    SI, BUFFER
        XOR    CX,CX
        MOV    CL, [SI]
        INC    SI
        MOV    DI, SI
        XOR    BH,BH
        XOR    AL,AL
L1:
        CMP    [SI],AL
        JZ    L2
        MOV    BL,[SI]
        MOV    [DI],BL
        INC    DI
        INC    BH
L2:
        INC    SI
        LOOP    L1
        MOV    BUFFER,BH
```

16. 编写一个子程序实现统计 AL 中 1 的个数，然后检测出字节型缓冲区 BUF 中 0 和 1 个数相等的元素个数。

解：统计 AL 中 1 的个数，只需将 AL 右移，移出的一位内容进行累加，子程序为：

```
COUNTBYTE    PROC NEAR
                PUSH AX
                PUSH CX
```

```
        MOV CX,8
        XOR BL,BL
    COU1:
        SHR AL,1
        ADC BL,0
        LOOP COU1
        POP CX
        POP AX
        RET
COUNTBYTE   ENDP
```

在此基础上，可以检测出字节型缓冲区 BUF 中 0 和 1 个数相等的元素个数，即一个字节中有 4 个 1。设 BUF 中有 N 个字节型数据，结果保持在 BH 中。

```
    MOV   CX,N
    LEA   SI, BUF
    XOR   BH,BH
L1: MOV   AL,[SI]
    CALL  COUNTBYTE
    CMP   BL,4
    JNZ   L2
    INC   BH
L2: INC   SI
    LOOP  L1
```

19. 在缓冲区 BUFFER 中，第 1 个字节存放数组的长度（<256），从第 2 个字节开始存放字符的 ASCII 码，编写子程序完成在最高位给字符加上偶校验。

解：

```
STACK   SEGMENT STACK 'STACK'
        DW 100H DUP（?）
        TOP LABEL BYTE
    STACK ENDS
```

```
DATA SEGMENT
BUFFER    DB 3              ;首字节为字符串长度
          DB 'ABC'          ;字符串
DATA ENDS
CODE SEGMENT
     ASSUME CS:CODE,DS:DATA,ES:DATA,SS:STACK
START:
          MOV AX,STACK
          MOV SS,AX
          LEA SP,TOP
          MOV AX,DATA
          MOV DS,AX
          MOV ES,AX
          LEA DI,BUFFER
          XOR CX,CX
          MOV CL,[DI]
          INC DI
          CALL SETEVEN
          MOV AH,4CH        ;返回 DOS
          MOV AL,0
          INT 21H
SETEVEN   PROC    NEAR      ;加偶校验子程序
          PUSH AX
          PUSH BX
          PUSH CX
          PUSH DI
 SETEVEN1:
          MOV AL,[DI]
          CALL COUNTBYTE
```

```
                AND BL,01H
                JZ SETEVEN2
                OR AL,80H
                MOV [DI],AL
 SETEVEN2:
                INC DI
                LOOP SETEVEN1
                POP DI
                POP CX
                POP BX
                POP AX
                RET
SETEVEN         ENDP
COUNTBYTE       PROC NEAR
                PUSH AX
                PUSH CX
                MOV CX,8
                XOR BL,BL
        COU1:
                SHR AL,1
                ADC BL,0
                LOOP COU1
                POP CX
                POP AX
                RET
COUNTBYTE    ENDP
        CODE    ENDS
                END START
```

20. 编写程序完成求多位数（N 个字）的绝对值。

21. 已知斐波那契数列的定义为：$F_1=1,F_2=1,F_i=F_{i-1}+F_{i-2}\quad(i\geq 3)$，编写求该数列前 n 项的子程序。

解：设奖斐波那契数列存放在字变量 RESULT 中。在数据段中定义

```
RESULT   DW 100H DUP（?）
```

在代码段中编写子程序

```
FIBONACCI       PROC   NEAR
        XOR    DI,DI
        MOV    RESULT[DI],1        ；前两个数为 1
        MOV    RESULT[DI+2],1
        ADD    DI,4
        MOV CX,N
    L1:
        MOV    AX, RESULT[DI-4]
        ADD    AX, RESULT[DI-2]
        MOV    RESULT[DI],AX
        ADD    DI,2
        LOOP   L1
        RET
FIBONACCI   ENDP
```

22. 编写程序实现循环显示 10 条信息，保存每条信息的变量分别为 INFOM1～INFORM10。

解：在数据段中定义变量：

```
TABLE   DW    INFORM1, INFORM2, INFORM3, INFORM4, INFORM5
        DW    INFORM6, INFORM7, INFORM8, INFORM9, INFORM10
```

在代码段中编写程序段：

```
        MOV     CX,10
        XOR     SI,SI
L1:     MOV     DX,TABLE[SI]
        MOV     AH,9
```

```
        INT     21H
        ADD     SI,2
        CALL    WAIT
        LOOP    L1
```

这里，WAIT 为延时子程序，用于在显示信息之间的停顿。

23. 编写程序实现将包含 20 个数据的数组 ARRAY 分成两个数组：正数数组 ARRAYP 和负数数组 ARRAYN，并分别将这两个数组中数据的个数显示出来。

解：先编写一个子程序 DISPALD，完成以 3 位十进制数形式显示出 AL 的内容。

```
DISPALD PROC NEAR
        PUSH AX
        PUSH CX
        PUSH DX
        XOR  AH,AH
        MOV  CL，100
        DIV  CL
        PUSH  AX
        MOV  DL,30H
        ADD  DL,AL
        MOV  AH,2
        INT  21H
        POP AX
        MOV AL,AH
        XOR AH,AH
        MOV CL,10
        DIV  CL
        PUSH  AX
        MOV  DL,30H
        ADD  DL,AL
        MOV  AH,2
```

```
        INT  21H
        POP AX
        MOV  DL,30H
        ADD  DL,AH
        MOV  AH,2
        INT  21H
        POP DX
        POP CX
        POP AX
        RET
DISPALD ENDP
```

在此基础上，根据题目要求，需要用到 3 个指针：SI 指向源数组 ARRAY，DI 指向正数数组 ARRAYP，BX 指向负数数组 ARRAYN。

```
        MOV     CX,20
        XOR     DX,DX
        LEA     SI,ARRAY
        LEA     DI,ARRAYP
        LEA     BX,ARRAYN
L1:     MOV     AL,[SI]
        AND     AL,AL
        JS      L2
        MOV     [DI],AL
        INC     DI
        INC     DL
        JMP     L3
L2:     MOV     [BX],AL
        INC     BX
        INC     DH
L3:     INC     SI
```

```
        LOOP    L1
        MOV     AL,DL
        CALL    DISPALD
        MOV     AL,DH
        CALL    DISPALD
```

24. 编写程序实现求缓冲区 BUFFER 的 100 个字中的最小偶数（存入 AX）。

解：设 BUFFER 中存放的是有符号数。

```
        MOV   CX,100
        LEA   SI,BUFFER
        MOV   AX, 7FFFH
L1:     AND WORD PTR [SI],1
        JNZ   L2
        CMP   [SI],AX
        JGE   L2
        MOV   AX,[SI]
L2:     ADD   SI,2
        LOOP  L1
```

25. 编写程序实现求级数$1^2+2^2+\cdots+n^2+\cdots$的前 n 项和刚大于 2000 的项数 n。

解：BL 用于存放项数。

```
STACK   SEGMENT STACK 'STACK'
        DW 100H DUP（?）
TOP     LABEL WORD
STACK   ENDS
DATA    SEGMENT
        DB 100H DUP（?）
DATA    ENDS
CODE    SEGMENT
        ASSUME CS:CODE, DS:DATA, ES:DATA, SS:STACK
START:
```

```
        MOV    AX,DATA
        MOV    DS,AX
        MOV    ES,AX
        MOV    AX,STACK
        MOV    SS,AX
        LEA    SP,TOP
        MOV    BL,1
        MOV    CX,0
COUNT:
        MOV    AL, BL
        MUL    BL
        ADD    CX,AX
        CMP    CX,2000
        JA     EXIT
        INC    BL
        JMP    COUNT
    EXIT:
```

26. 定义一条宏指令，实现将指定数据段的段地址传送到段寄存器 ES 或 DS 的功能。

解：

27. 定义一条宏指令，实现从键盘中输入一个字符串（利用 INT 21H 的 09 号功能）。

28. 定义一条宏指令，实现在屏幕上输出回车、换行。

29. 利用其它指令完成与下列指令一样的功能：

（1）REP MOVSB；　　（2）REP LODSB；

（3）REP STOSB；　　（4）REP SCASB。

解：设 DF=0

```
（1）    L1:  MOV    AL,[SI]
              MOV    ES:[DI],AL
              INC      SI
              INC      DI
```

```
        LOOP    L1
```

（2）
```
    L1: MOV   AL,[SI]
        INC     SI
        LOOP    L1
```

（3）
```
    L1: MOV   ES:[DI],AL
        INC     DI
        LOOP    L1
```

（4）
```
    L1: MOV   AL,[SI]
        CMP   AL,ES:[DI]
        INC     SI
        INC     DI
        LOOP    L1
```

30. 设在数据段中定义了：

```
        STR1   DB  'ASSEMBLE  LANGUAGE'
    STR2   DB   20 DUP(?)
```

利用字符串指令编写程序段实现：

（1）从左到右将 STR1 中的字符串传送到 STR2；

（2）从右到左将 STR1 中的字符串传送到 STR2；

（3）将 STR1 中的第 6 个和第 7 个字节装入 DX；

（4）扫描 STR1 字符串中有无空格，如有则将第一个空格符的地址传送到 SI。

解： STR1 中有 17 个字符（含一个空格），设 DS 和 ES 均指向 STR1 和 STR2 所在的段。

（1）
```
    CLD
    MOV     CX,17
    LEA     SI,STR1
    LEA     DI,STR2
    REP     MOVSB
```

（2）
```
    MOV     CX,17
    LEA     SI,STR1
    LEA     DI,STR2
```

```
        ADD     SI,CX
        DEC     SI
L1:     MOV     AL,[SI]
        MOV     [DI],AL
        DEC     SI
        INC     DI
        LOOP    L1
（3）   LEA     SI,STR1
        MOV     DX,[SI+6]
（4）   MOV     CX,17
        LEA     SI,STR1
        MOV     AL,20H
L1:     CMP      [SI], AL
        JZ      L2
        INC     SI
        LOOP    L1
L2:
```

31. 设在数据段中定义了：

```
STRING  DB  ‘Today is Sunday & July 16, 2000’
```

编写程序实现将 STRING 中的’&’用’/’代替。

解：STRING 中保存了 30 个字符。

```
        MOV     CX,30
        LEA     SI,STRING
        MOV     AL,’&’
L1:     CMP     [SI],AL
        JNZ     L2
        MOV     BYTE PTR [SI],’/’
L2:     INC     SI
        LOOP    L1
```

32. 分析下列程序段完成的功能：

```
MOV CX，100
LEA SI，FIRST
LEA DI，SECOND
REP MOVSB
```

解：将缓冲区 FIRST 中 100 个字节传送到 SECOND 中。

33. 分析下列程序段：

```
LEA DI，STRING
MOV CX，200
CLD
MOV AL，20H
REPZ    SCASB
JNZ    FOUND
JMP    NOT_FOUND
```

问：转移到 FOUND 的条件。

解：在缓冲区 STRING 中搜索非空格字符，如果有非空格则转到 FOUND，如果 200 个单元中都是空格，则转到 NOT_FOUND。

34. 设在数据段的变量 OLDS 和 NEWS 中保存有 5 个字节的字符串，如果 OLDS 字符串不同于 NEWS 字符串，则执行 NEW_LESS，否则顺序执行程序。

解：设 DS 和 ES 均指向字符串 OLDS 和 NEWS 所在的段。

```
CLD
MOV     CX,5
LEA     SI,OLDS
LEA     DI,NEWS
REPZ    CMPSB
JNZ     NEW_LESS
```

35. 编程实现将 STRING 字符串中的小写字母变换成大写字母。

解：设 STRING 中的字符个数为 N。

```
        MOV     CX,N
        LEA     SI,STRING
        MOV     AL,5FH
L1:     AND     [SI],AL
        INC     SI
        LOOP    L1
```

36. 设在数据段中定义了：

STUDENT_NAME　DB　30 DUP（？）

STUDENT_ADDR　DB　9 DUP（？）

STUDENT_PRINT　DB　50 DUP（？）

编写程序实现：

用空格符清除缓冲区 STUDENT_PRINT；

在 STUDENT_ADDR 中查找第一个’_’字符；

在 STUDENT_ADDR 中查找最后一个’_’字符；

如果 STUDENT_NAME 中全为空格符，则 STUDENT_PRINT 全存入’*’；

将 STUDENT_NAME 传送到 STUDENT_PRINT 的前 30 个字节中，将 STUDENT_ADDR 传送到 STUDENT_PRINT 的后 9 个字节中。

37. (上机题)编写程序实现，将缓冲区 BUFFER 中的 100 个字按递增排序，并按下列格式顺序显示：

数据 1　<原序号>

数据 2　<原序号>

……

38. (上机题)按同余法产生一组随机数 N（1<N<=50），并按 N＋50 赋给 45 名同学的 5 门课程的成绩，要求编程实现计算每个同学的平均成绩，并根据平均成绩统计全班的成绩各等级的人数（A：90～100，B：80～89，C：70～79，D：66～69，E：60～65，F：60 分以下），按下列格式显示：

Total　<总人数>

A:　<人数 1>

B:　<人数 2>

C:　<人数 3>

D:　　<人数 4>

E:　　<人数 5>

F:　　<人数 6>

39．(上机题)编写程序实现下列 5 项功能，通过从键盘输入 1～5 进行菜单式选择：

（1）按数字键“1”，完成将字符串中的小写字母变换成大写字母。用户输入由英文大小写字母或数字 0～9 组成的字符串（以回车结束），变换后按下列格式在屏幕上显示：

<原字符串>例如：abcdgyt0092

<新字符串>　　ABCDGYT0092

按任一键重做；按 Esc 键返回主菜单。

（2）按数字键“2”，完成在字符串中找最大值。用户输入由英文大小写字母或数字 0～9 组成的字符串（以回车结束），找出最大值后按下列格式在屏幕上显示：

<原字符串>　　The maximum is <最大值>.

按任一键重做；按 Esc 键返回主菜单。

（3）按数字键“3”，完成输入数据组的排序。用户输入一组十进制数值（小于 255），然后变换成十六进制数，并按递增方式进行排序，按下列格式在屏幕上显示：

<原数值串>

<新数值串>

按任一键重做；按 Esc 键返回主菜单。

（4）按数字键“4”，完成时间的显示。首先提示用户对时，即改变系统的定时器 HH：MM：SS（以冒号间隔，回车结束），然后在屏幕的右上角实时显示出时间：HH：MM：SS。

按任一键重新对时；按 Esc 键返回主菜单。

（5）按数字键“5”，结束程序的运行，返回操作系统。

解：

※主程序的编程思路：

此程序共 5 个功能，可采用跳转表法来实现多路分支结构程序设计。现将这 5 个程序段，各程序段的首地址分别标号为 G1，G2，G3，G4，G5。将 5 个程序段的入口地址做成表 TABLE 放入数据段，程序根据给定的参数计算出欲转入的程序段的首地址在 TABLE 中的位置后，取出该地址，跳转至该程序段。

首先，通过调用子程序 MENU，设置显示器，并输出提示文档。接着，读取‘1’-‘5’之间的 ASCII 表示数。然后，通过跳转表 TABLE 实现由输入参数转入相应的程序段。由于表中

按“字”存放数据，则每个数据的位移量是：0、2、4、6、8。对于输入参数 N，计算位移量的公式是 N=(N-1)*2。

当输入‘1’时，跳转到标号 G1。调用子程序 CHGLTR，完成将输入字符串中的小写字母变换成大写字母。用户按键，若为 ESC，则转到主程序段首调用 MENU，否则，转到标号 G1；

当输入‘2’时，跳转到标号 G2。调用子程序 MAXLTR，完成在输入字符串中找最大值。用户按键，若为 ESC，则转到主程序段首调用 MENU，否则，转到标号 G2；

当输入‘3’时，跳转到标号 G3。调用子程序 SORTNUM，完成输入数据组的排序。用户按键，若为 ESC，则转到主程序段首调用 MENU，否则，转到标号 G3。

当输入‘4’时，跳转到标号 G4。调用子程序 TIMCHK，完成时间的显示。用户按键，若为 ESC，则转到主程序段首调用 MENU，否则，转到标号 G4。

当输入‘5’时，跳转到标号 G5。结束程序的运行，返回操作系统。

其流程框图见图 3-1。

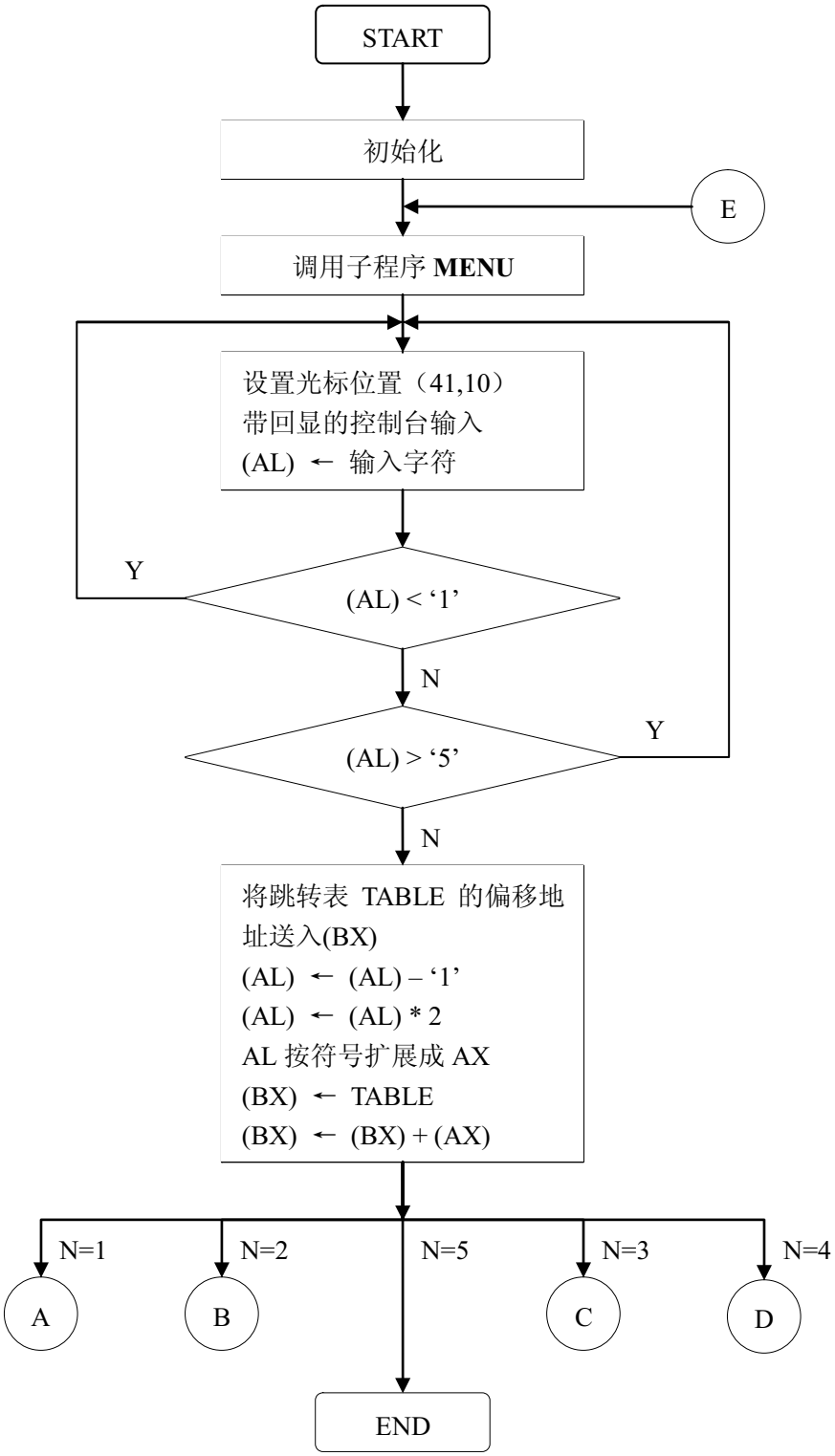

START
初始化
E
调用子程序 MENU
设置光标位置（41,10）
带回显的控制台输入
(AL) ← 输入字符
Y
(AL) < '1'
N
Y
(AL) > '5'
N
将跳转表 TABLE 的偏移地址送入(BX)
(AL) ← (AL) – '1'
(AL) ← (AL) * 2
AL 按符号扩展成 AX
(BX) ← TABLE
(BX) ← (BX) + (AX)
N=1
N=2
N=5
N=3
N=4
A
B
C
D
END

图 3-1 主程序流程框图

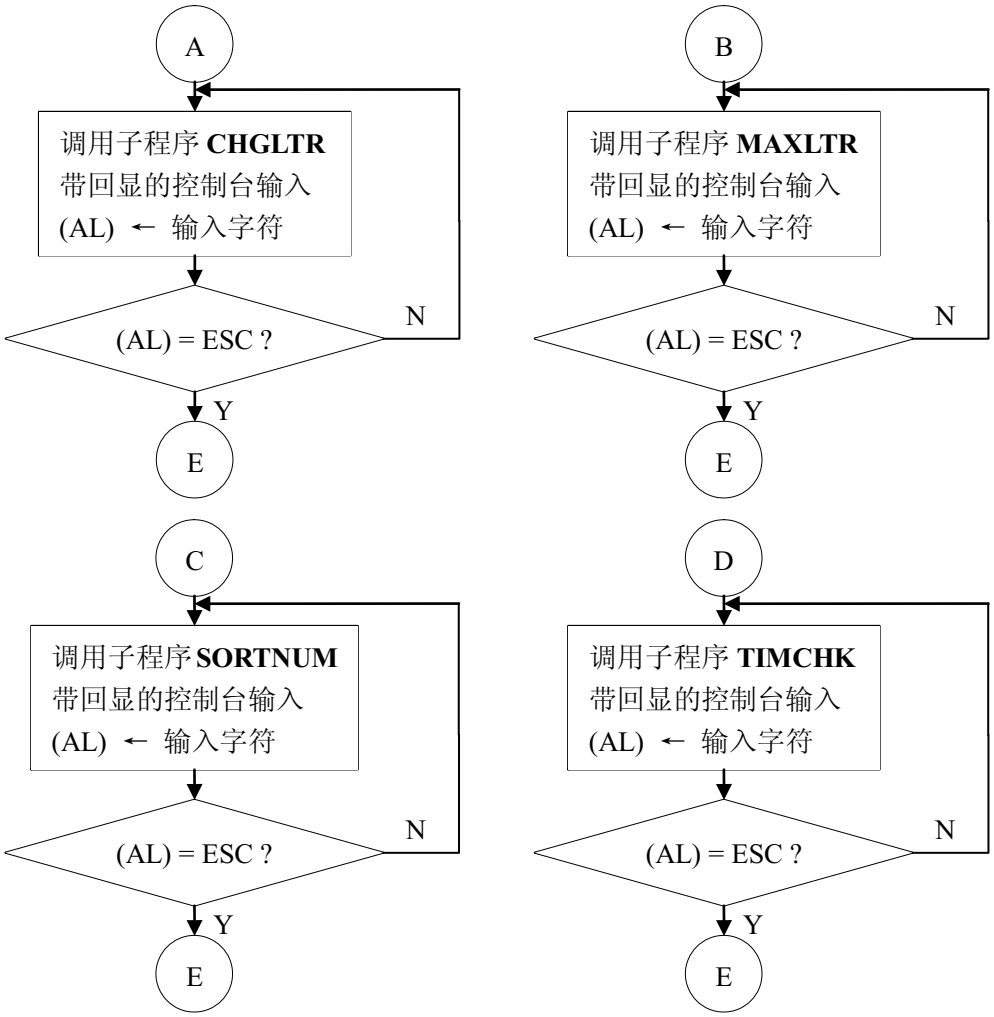


图 3-1（续） 主程序流程框图

※子程序 MENU 的编程思路：

设置显示器显示方式为 80*25 彩色文本方式，清屏。逐行设置光标位置，使提示文档左对齐整体居中。输出 1-5 的提示文档，再输出输入 N 的提示。其流程框图见图 3-2。

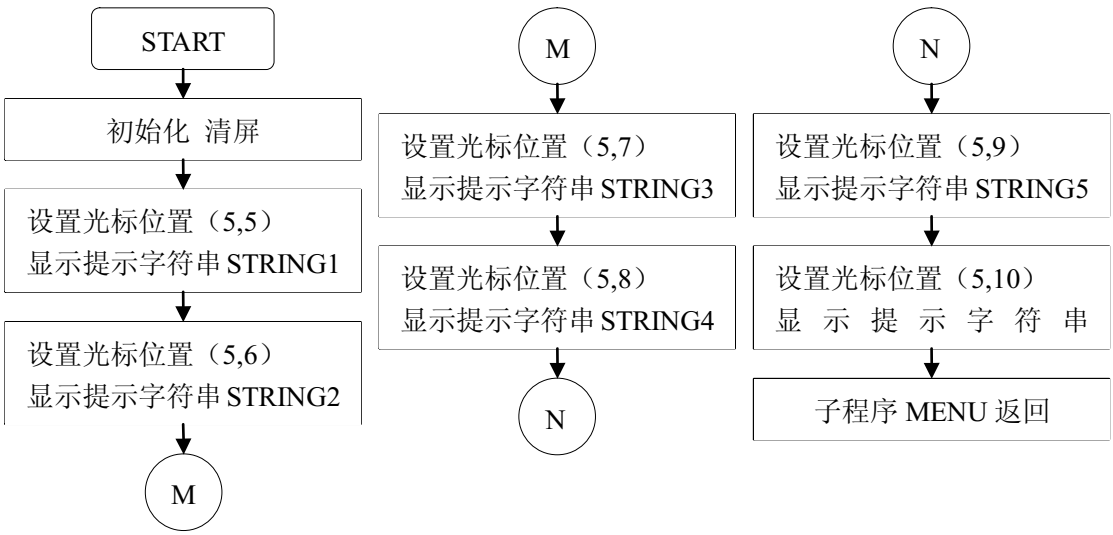


图 3-2 子程序 MENU 流程框图

※子程序 CHGLTR 的编程思路：

设置显示器显示方式为 80*25 彩色文本方式，清屏。设置光标位置，使提示文档左对齐整体居中。输出输入字符串提示文档，读取输入字符串并将其放入 KEYBUF。在输入字符串尾加结束标志$，输出输入字符串。从前往后，依次取字符串中的每个字符，若其为小写字母，则将其 ASCII 码减去 20H。输出变换后的字符串。最后输出说明文档。

其流程框图见图 3-3。

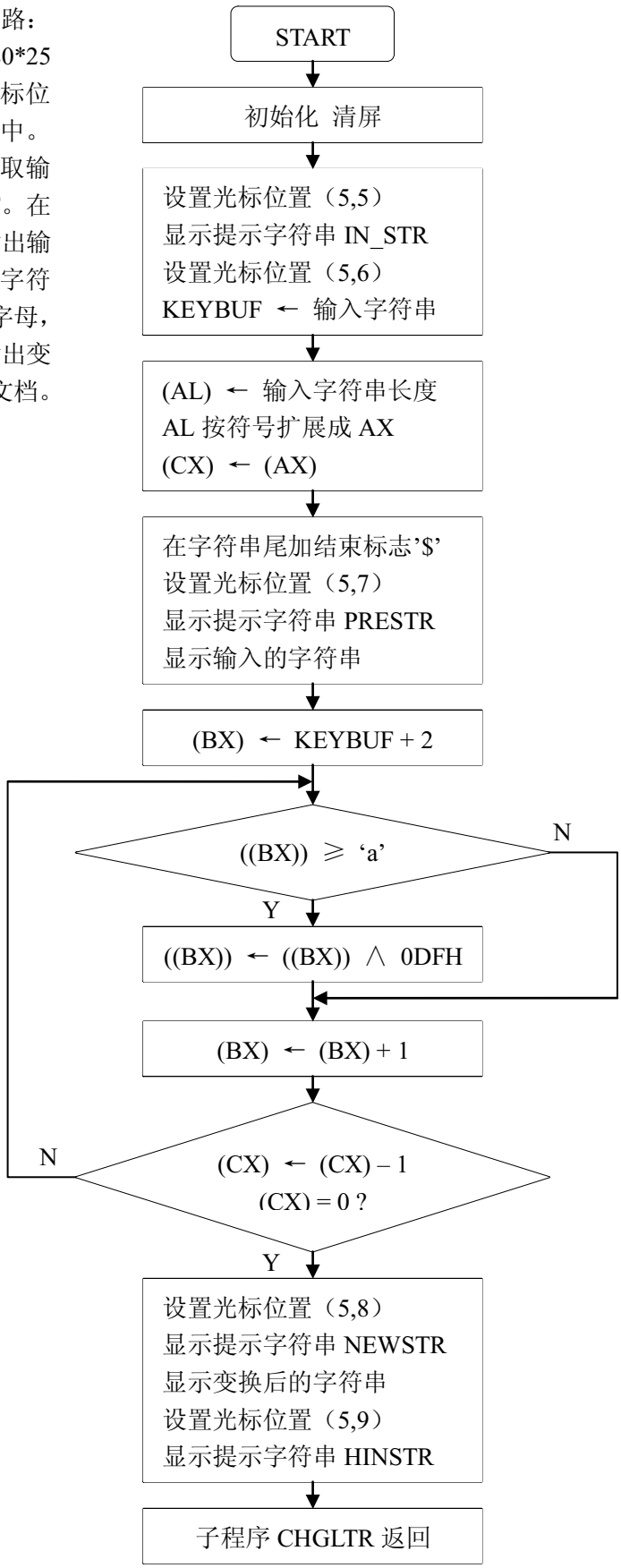


图 3-3 子程序 CHGLTR 流程框图

※子程序 MAXLTR 的编程思路：

设置显示器显示方式为 80*25 彩色文本方式，清屏。设置光标位置，使提示文档左对齐整体居中。输出输入字符串提示文档，读取输入字符串并将其放入 KEYBUF。在输入字符串尾加结束标志$，输出输入字符串。预设字符串中最大值为 0。从前往后，依次取字符串中的每个字符，若其大于当前最大值，则进行替换，即可得到字符串中的最大值，并输出。最后输出说明文档。

其流程框图见图 3-4。

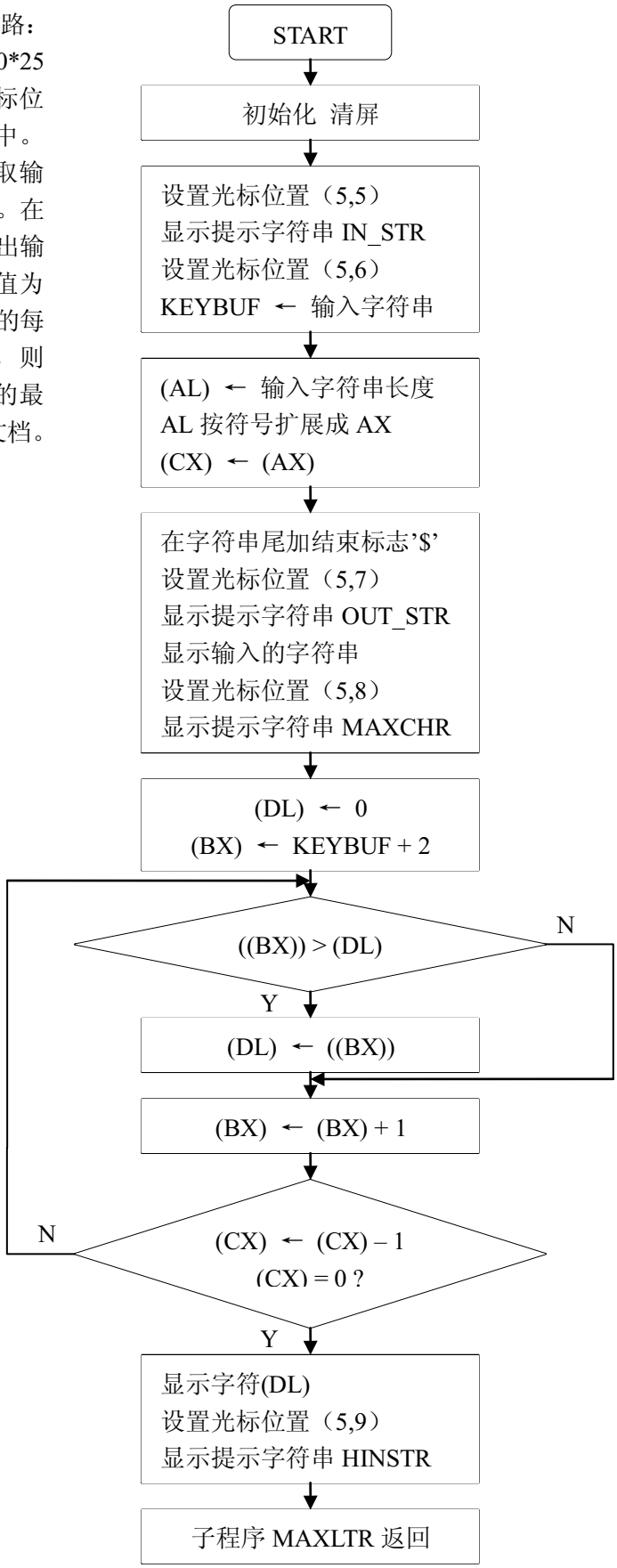


图 3-4 子程序 MAXLTR 流程框图

※子程序 SORTNUM 的编程思路：

设置显示器显示方式为 80*25 彩色文本方式，清屏。设置光标位置，使提示文档左对齐整体居中。输出输入数据组提示文档，读取输入数据组字符串并将其放入 KEYBUF。调用子程序 CIN_INT，将字符串转换成数据串。判断数据串是否有错误或者为空，若是，则重新输入数据组。调用子程序 MPSORT，采用冒泡法对数据串进行排序。再调用子程序 INT_OUT，输出排序后的数据组。最后输出说明文档。

其流程框图见图 3-5。

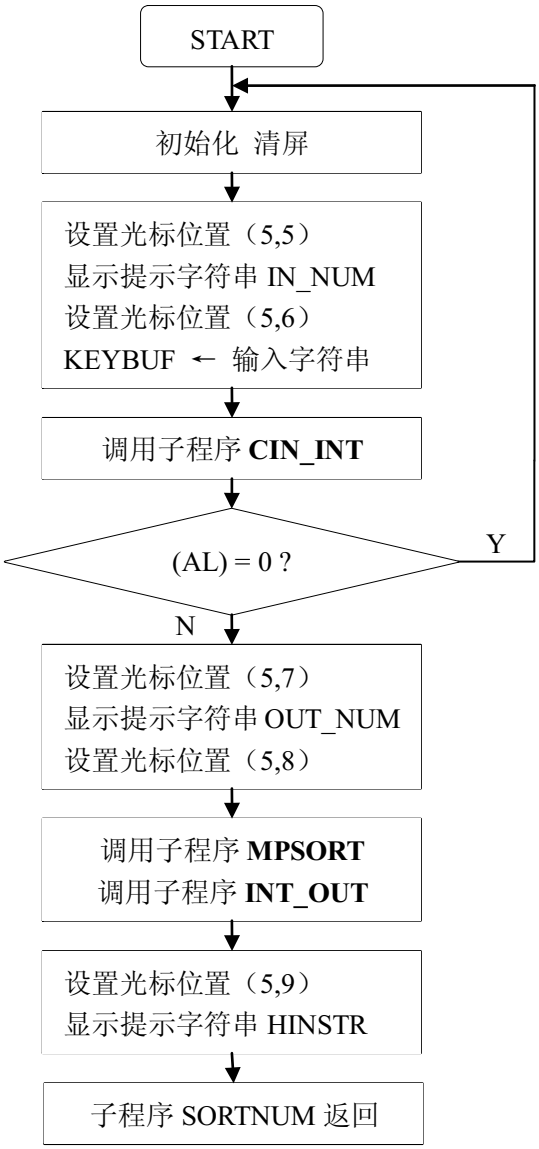


图 3-5 子程序 SORTNUM 流程框图

※子程序 CIN_INT 的编程思路：

入口参数为：无；出口参数为：AL（有无错误标志，0 为有，1 为无）。从前往后，依次取字符串中的每个字符进行判断。CH 表示数据组数据个数，AL 表示当前数据 x，DL 作为有无数据标志。若当前字符为空格，则转到 ADDNUM，判断 DL 是否为 1，若为 1，则 CH 增 1，调用子程序 ADDNEW，增加新数 x，然后 DL、AL 清零；否则判断当前字符 c 是否在‘0’-‘9’之间，若不是，则判错，将 AL 置 0，子程序 CIN_INT 返回；否则，DL 置 1，x=x*10+c-‘0’，判断 x 是否超过 255，若是，则判错，将 AL 置 0，子程序 CIN_INT 返回；否则，对下一个字符进行操作。字符串判断结束后，若 DL 为 1，则有新数 x 未加至数据组，调用子程序 ADDNEW，增加新数 x。将数据组个数 CH 放入 NUMBUF，将 AL 置 1。

其流程框图见图 3-6。

※子程序 ADDNEW 的编程思路：

入口参数为：CH（数据组数据个数）、AL（当前数据 x）；出口参数为：无。取出数据组 NUMBUF 的首地址，加上数据组数据个数，即为当前数据 x 的地址，将 x 放入该地址。

其流程框图见图 3-7。

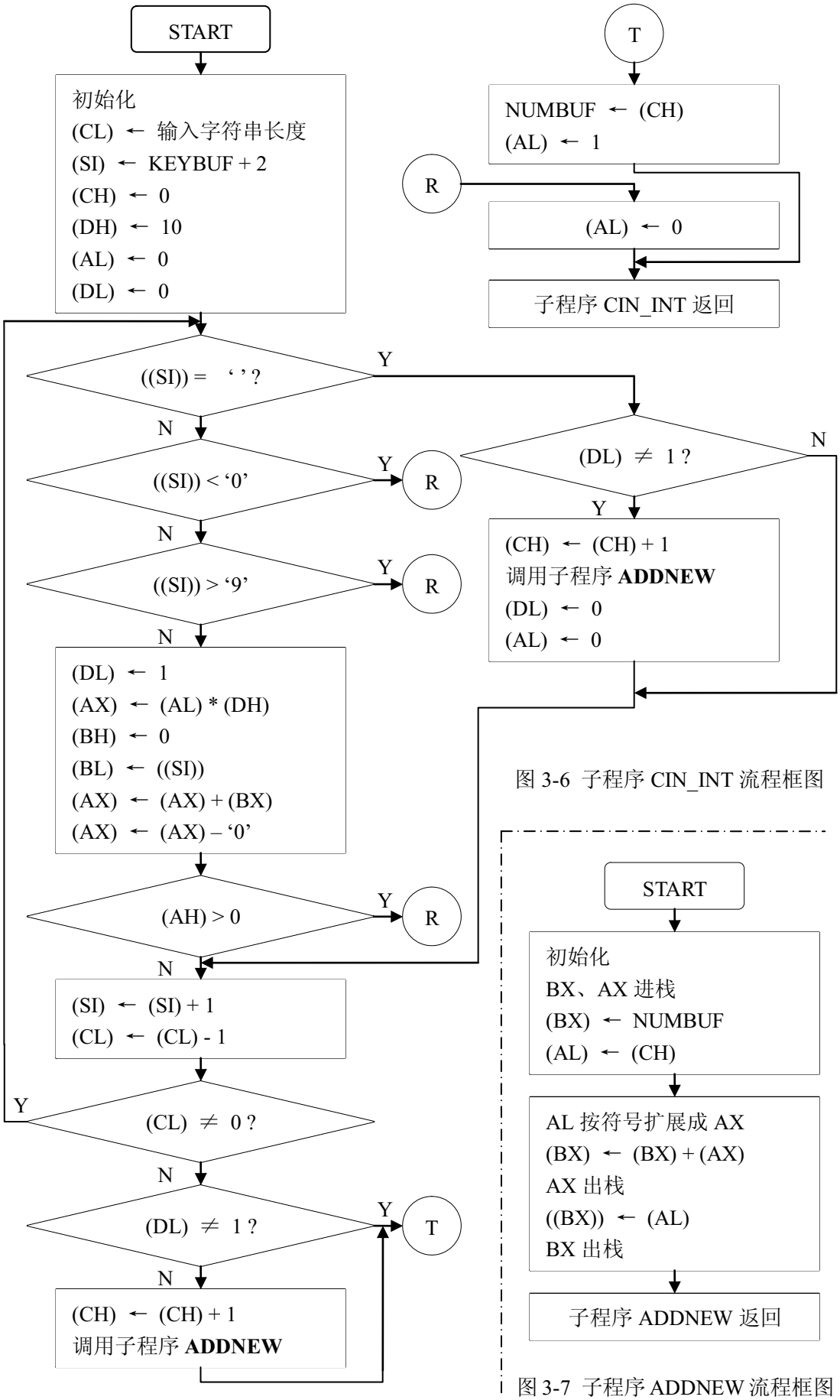


图 3-6 子程序 CIN_INT 流程框图

图 3-7 子程序 ADDNEW 流程框图

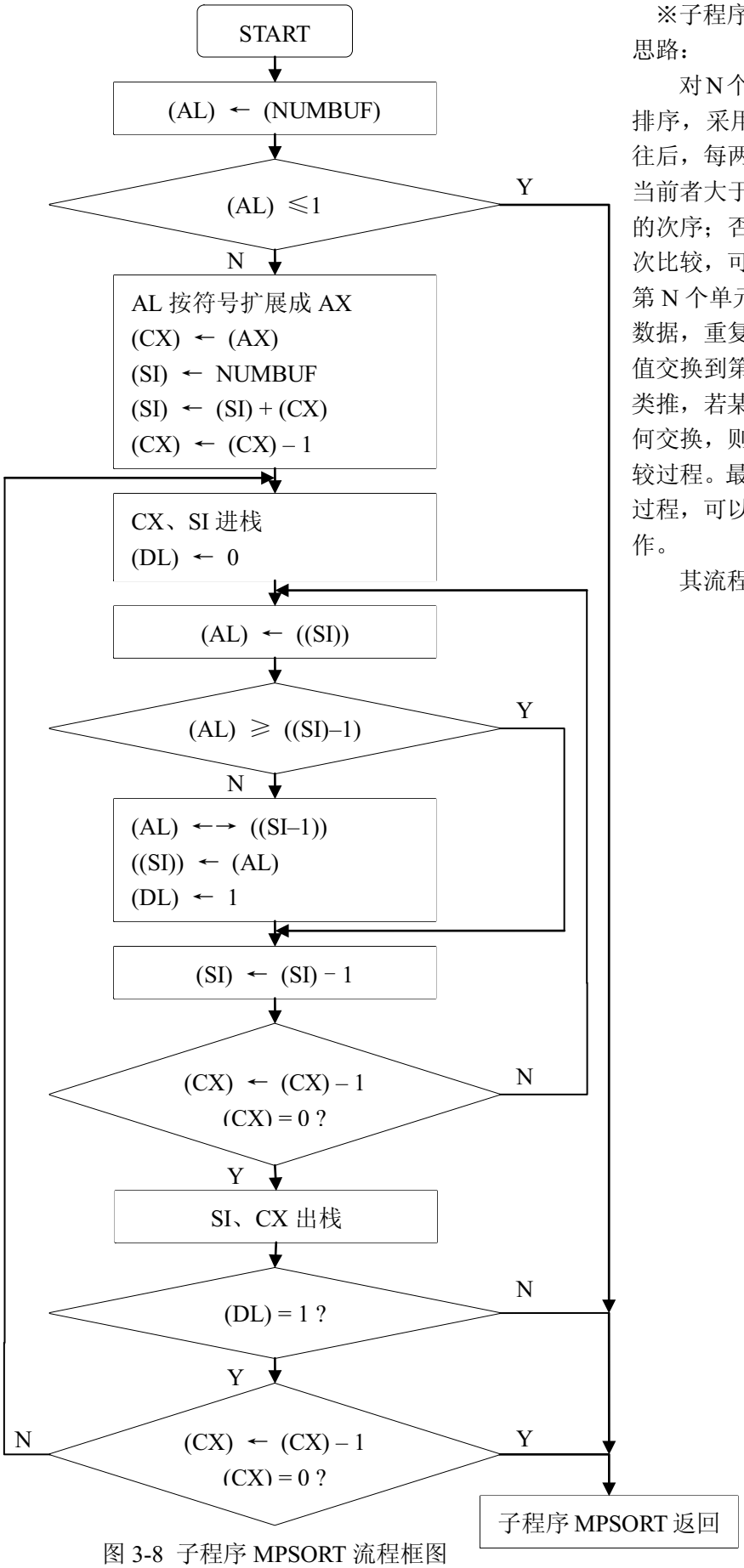


图 3-8 子程序 MPSORT 流程框图

※子程序 MPSORT 的编程思路：

对N个数据进行从小到大排序，采用“冒泡法”：从前往后，每两个数据进行比较，当前者大于后者时，交换两者的次序；否则不变。经过 N-1 次比较，可以将最大值交换到第 N 个单元。接着对前 N-1 个数据，重复上述过程，使次大值交换到第 N-1 个单元；依此类推，若某次比较过程，无任何交换，则终止其后的所有比较过程。最多进行 N-1 次比较过程，可以完成数据的排序操作。

其流程图见图 3-8。

※子程序 IN_OUT 的编程思路：

从数据组中依次取出每个数据，调用 OUTNUM，将十进制数据转成十六进制进行输出，数据之间输出一个空格作为分隔符。

其流程图见图 3-9。

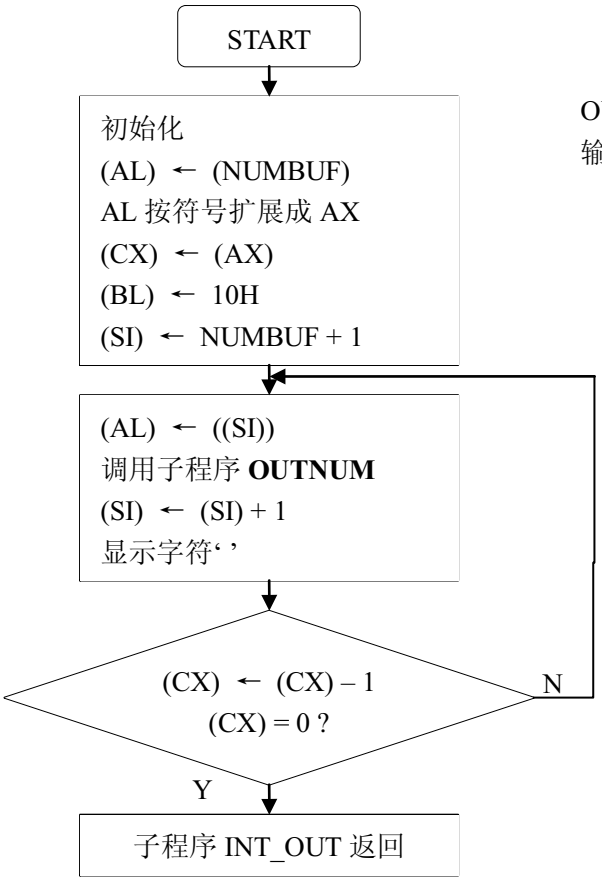


图 3-9 子程序 INT_OUT 流程框图

※子程序 OUTNUM 的编程思路：

入口参数：AL（待转换的数据），BL（转换进制数）；出口参数：无。待转换数据 x 除以转换进制数，商为新的 x，余数 y 为转换后的低位。保存 y，调用子程序 OUTNUM 本身，对新的 x 进行进制转换并输入。取出 y，对其进行输入，若低于 10，则直接输出，否则转换成字母输出。

其流程图见图 3-10。

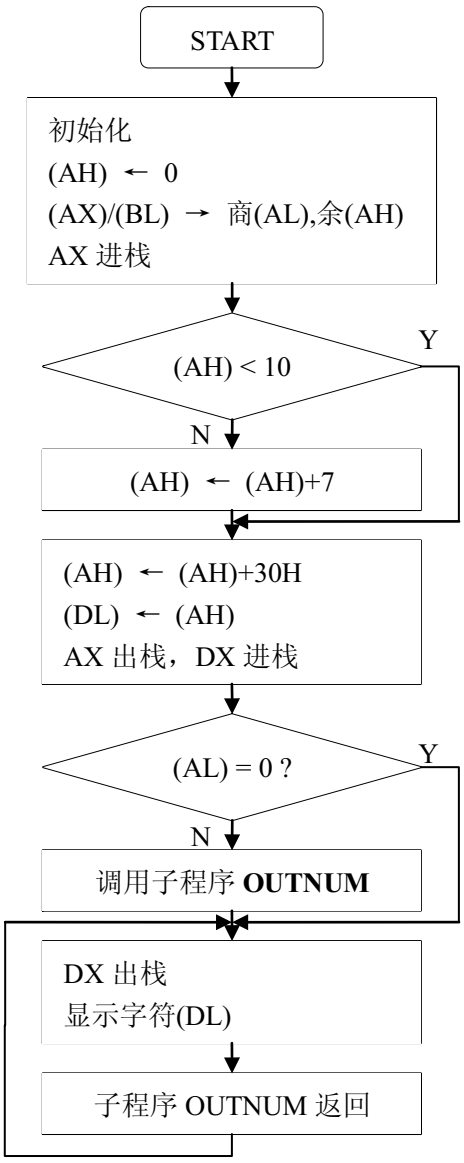


图 3-10 子程序 OUTNUM 流程框图

※子程序 TIMCHK 的编程思路：

设置显示器显示方式为 80*25 彩色文本方式，清屏。设置光标位置，输出设置时间提示文档。读取输入时间字符串并将其放入 KEYBUF。分别判断时、分、秒是否在有效数字范围，若有效，则设置新的系统时间。调用子程序 TIME，在屏幕的右上角实时显示时间。

其流程框图见图 3-11。

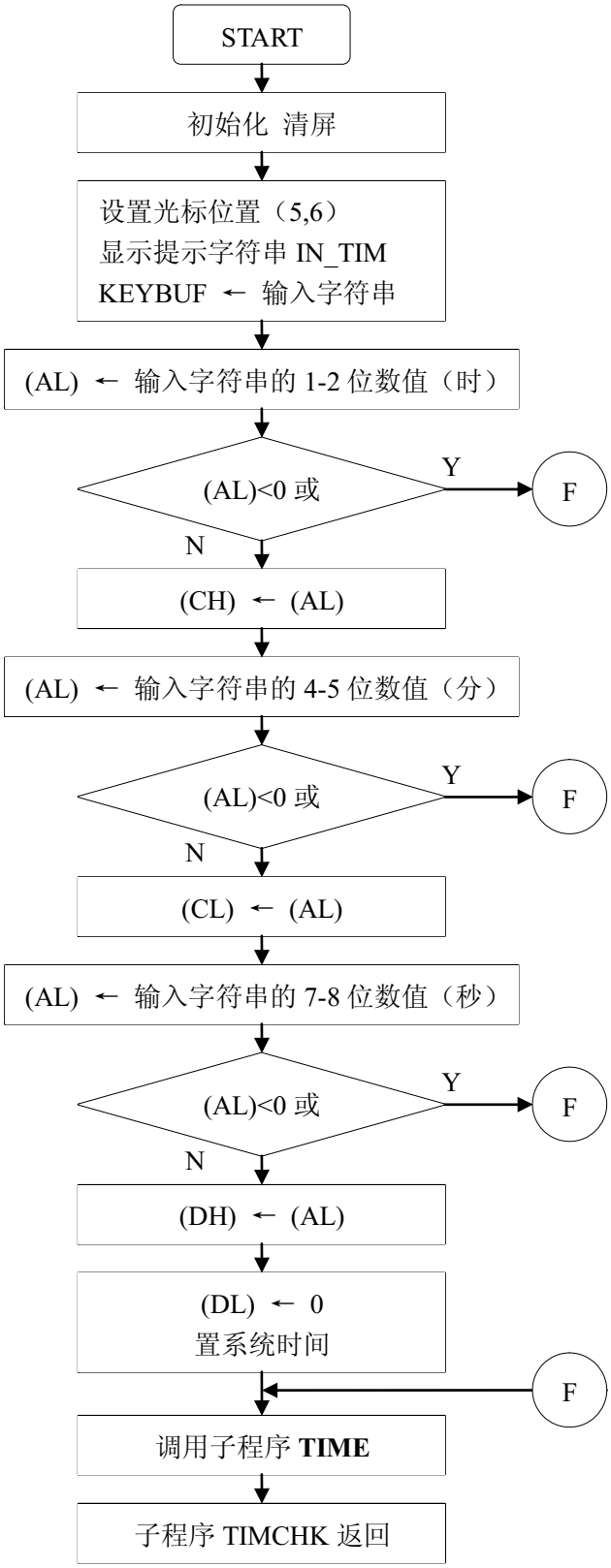


图 3-11 子程序 TIMCHK 流程框图

※子程序 TIME 的编程思路：

设置显示器显示方式为 80*25 彩色文本方式，清屏。设置光标位置，输出说明文档。实时显示时间：设置光标至屏幕右上角；读取系统时间，调用子程序 SHOWNUM 分别对时、分、秒进行显示，并用':'作分隔符；设置光标等待位置，并进行延时；检查键盘状态，若无键盘输入，则重复上述步骤显示时间，否则子程序 TIME 返回。

其流程框图见图 3-12。

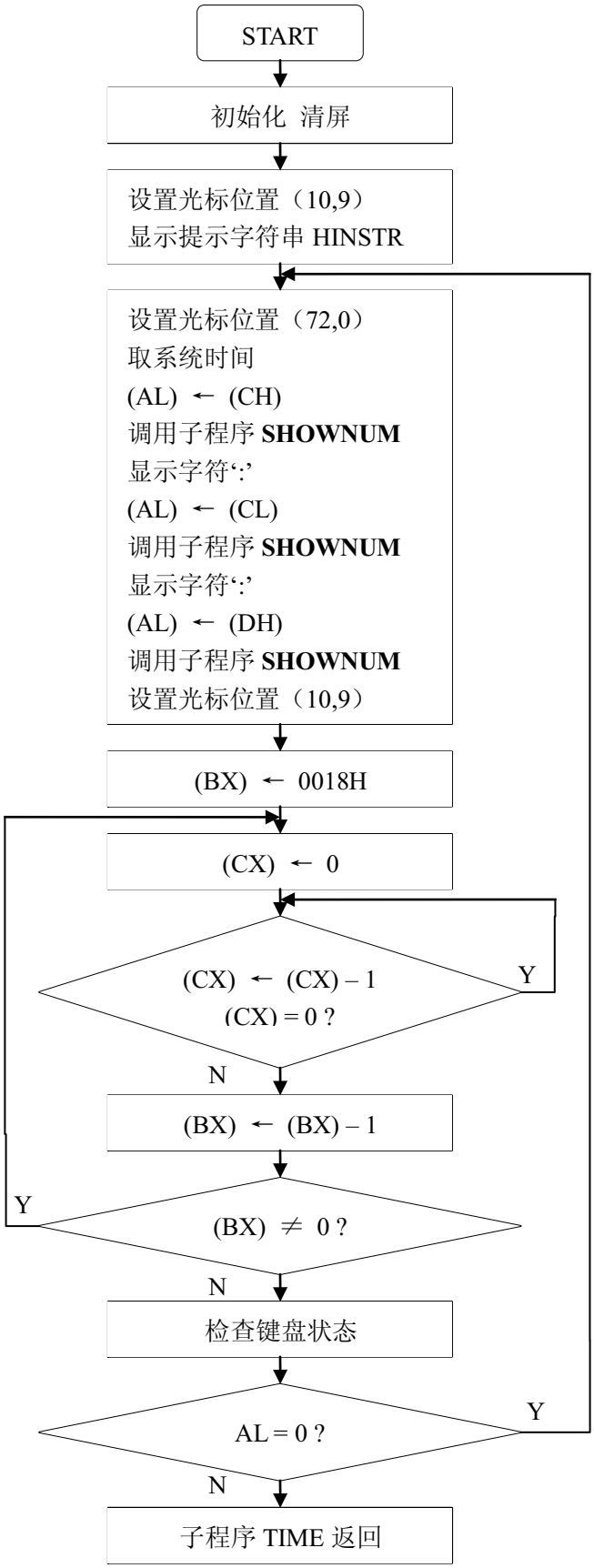


图 3-12 子程序 TIME 流程框图

※子程序 SHOWNUM 的编程思路：

入口参数：AL（待显示的数据）；出口参数：无。待显示数据 x 除以 10，商为 y，余数为 z。将数字 y、z 分别转换成字符数字，然后输出。

其流程框图见图 3-13。

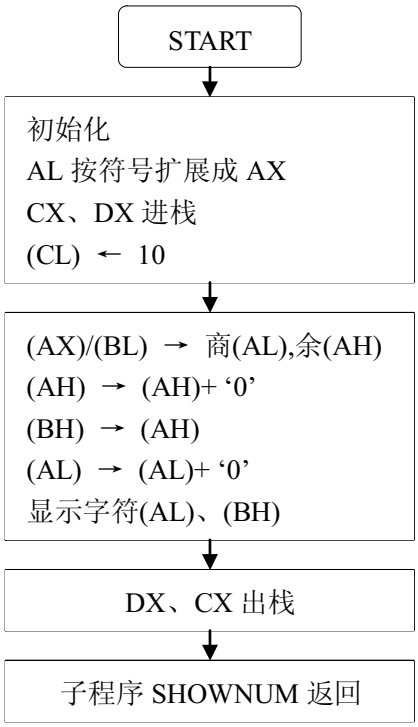


图 3-13 子程序 SHOWNUM 流程框图

四、程序代码

```
STACK   SEGMENT STACK
        DB 256 DUP(?)
TOP     LABEL WORD
STACK   ENDS

DATA        SEGMENT
TABLE       DW G1, G2, G3, G4, G5
STRING1     DB '1. Change small letters into capital letters of string;', 0DH, 0AH, '$'
STRING2     DB '2. Find the maximum of string;', 0DH, 0AH, '$'
STRING3     DB '3. Sort for datas;', 0DH, 0AH, '$'
STRING4     DB '4. Show Time;', 0DH, 0AH, '$'
STRING5     DB '5. Exit.', 0DH, 0AH, '$'
STRINGN     DB 'Input the number you select (1-5) : $'
IN_STR  DB 'Input the string (including letters & numbers, less than 60 letters) :', 0DH, 0AH, '$'
PRESTR      DB 'Original string : $'
NEWSTR      DB 'New string      : $'
OUT_STR     DB 'The string is $'
MAXCHR      DB 'The maximum is $'
IN_NUM      DB 'Input the numbers (0 - 255, no more than 20 numbers) : ', 0DH, 0AH, '$'
OUT_NUM     DB 'Sorted numbers : ', 0DH, 0AH, '$'
IN_TIM      DB 'Correct the time (HH:MM:SS) : $'
HINTSTR     DB 'Press ESC, go back to the menu; or press any key to play again!$'
KEYBUF      DB 61
            DB ?
            DB 61 DUP (?)
NUMBUF      DB ?
            DB 20 DUP (?)
DATA        ENDS

CODE    SEGMENT
        ASSUME CS:CODE, DS:DATA, SS:STACK
START:
        MOV AX, DATA
        MOV DS, AX
        MOV AX, STACK
        MOV SS, AX
        MOV SP, OFFSET TOP
MAIN:   CALL FAR PTR MENU       ; 设置显示器
AGAIN:
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 41              ; 列号
        MOV DH, 10              ; 行号
        INT 10H                 ; 光标位置设置
```

```
        MOV AH, 1
        INT 21H
        CMP AL, '1'
        JB AGAIN
        CMP AL, '5'
        JA AGAIN
        SUB AL, '1'             ; N-1
        SHL AL, 1               ; (N-1)*2
        CBW                     ; AL->AX
        LEA BX, TABLE
        ADD BX, AX
        JMP WORD PTR [BX]
G1:
        CALL FAR PTR CHGLTR
        MOV AH, 8
        INT 21H
        CMP AL, 1BH
        JZ MAIN
        JMP G1
G2:
        CALL FAR PTR MAXLTR
        MOV AH, 8
        INT 21H
        CMP AL, 1BH
        JZ MAIN
        JMP G2
G3:
        CALL FAR PTR SORTNUM
        MOV AH, 8
        INT 21H
        CMP AL, 1BH
        JZ MAIN
        JMP G3
G4:
        CALL FAR PTR TIMCHK
        MOV AH, 8
        INT 21H
        CMP AL, 1BH
        JZ MAIN
        JMP G4
G5:
        MOV AH, 4CH
        INT 21H
```

```
MENU    PROC FAR                ; 显示主界面
        ;设置显示器方式
        MOV AH, 0
        MOV AL, 3;
        MOV BL, 0;
        INT 10H                 ; 清屏
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 5               ; 列号
        MOV DH, 5               ; 行号
        INT 10H                 ; 光标位置设置
        MOV AH, 9
        LEA DX, STRING1
        INT 21H
        MOV AH, 2
        MOV DL, 5               ; 列号
        MOV DH, 6               ; 行号
        INT 10H                 ; 光标位置设置
        MOV AH, 9
        LEA DX, STRING2
        INT 21H
        MOV AH, 2
        MOV DL, 5               ; 列号
        MOV DH, 7               ; 行号
        INT 10H                 ; 光标位置设置
        MOV AH, 9
        LEA DX, STRING3
        INT 21H
        MOV AH, 2
        MOV DL, 5               ; 列号
        MOV DH, 8               ; 行号
        INT 10H                 ; 光标位置设置
        MOV AH, 9
        LEA DX, STRING4
        INT 21H
        MOV AH, 2
        MOV DL, 5               ; 列号
        MOV DH, 9               ; 行号
        INT 10H                 ; 光标位置设置
        MOV AH, 9
        LEA DX, STRING5
        INT 21H
        MOV AH, 2
        MOV DL, 5               ; 列号
```

```
        MOV DH, 10              ; 行号
        INT 10H                 ; 光标位置设置
        MOV AH, 9
        LEA DX, STRINGN
        INT 21H
        RET
MENU    ENDP

CHGLTR      PROC FAR            ; 将输入字符串中小写字母便换成大写字母
RECHG:
        ;设置显示器方式
        MOV AH, 0
        MOV AL, 3
        MOV BL, 0
        INT 10H                 ; 清屏
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 5               ; 列号
        MOV DH, 5               ; 行号
        INT 10H                 ; 输入提示光标位置设置
        MOV AH, 9
        LEA DX, IN_STR
        INT 21H                 ; 输入字符串提示
        MOV AH, 2
        MOV DL, 5               ; 列号
        MOV DH, 6               ; 行号
        INT 10H                 ; 输入字符串光标位置设置
        MOV AH, 0AH
        LEA DX, KEYBUF
        INT 21H                 ; 输入字符串
        CMP KEYBUF + 1, 0
        JZ RECHG                ; 判断输入字符串是否为空串
        LEA BX, KEYBUF + 2
        MOV AL, KEYBUF + 1
        CBW
        MOV CX, AX
        ADD BX, AX
        MOV BYTE PTR [BX], '$'  ; 在输入字符串尾加结束标志$
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 5               ; 列号
        MOV DH, 7               ; 行号
        INT 10H                 ; 源字符串提示光标位置设置
        MOV AH, 9
        LEA DX, PRESTR
```

```
        INT 21H                 ; 输出源字符串提示
        MOV AH, 9
        LEA DX, KEYBUF + 2
        INT 21H                 ; 输出源字符串
        LEA BX, KEYBUF + 2
LCHG:
        CMP BYTE PTR [BX], 61H
        JB NOCHG
        AND BYTE PTR [BX], 0DFH
NOCHG:
        INC BX
        LOOP LCHG               ; 将字符串中小写字母转换成大写字母
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 5               ; 列号
        MOV DH, 8               ; 行号
        INT 10H                 ; 新字符串提示光标位置设置
        MOV AH, 9
        LEA DX, NEWSTR
        INT 21H                 ; 输出新字符串提示
        MOV AH, 9
        LEA DX, KEYBUF + 2
        INT 21H                 ; 输出新字符串
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 5               ; 列号
        MOV DH, 9               ; 行号
        INT 10H                 ; 提示信息光标位置设置
        MOV AH, 9
        LEA DX, HINTSTR
        INT 21H                 ; 输出提示信息
        RET
CHGLTR  ENDP

MAXLTR  PROC FAR                ; 在输入字符串中找出最大值
REMAX:
        ;设置显示器方式
        MOV AH, 0
        MOV AL, 3
        MOV BL, 0
        INT 10H                 ; 清屏
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 5               ; 列号
        MOV DH, 5               ; 行号
```

```
        INT 10H                 ; 输入提示光标位置设置
        MOV AH, 9
        LEA DX, IN_STR
        INT 21H                 ; 输入字符串提示
        MOV AH, 2
        MOV DL, 5               ; 列号
        MOV DH, 6               ; 行号
        INT 10H                 ; 输入字符串光标位置设置
        MOV AH, 0AH
        LEA DX, KEYBUF
        INT 21H                 ; 输入字符串
        CMP KEYBUF + 1, 0
        JZ REMAX                ; 判断输入字符串是否为空串
        LEA BX, KEYBUF + 2
        MOV AL, KEYBUF + 1
        CBW
        MOV CX, AX
        ADD BX, AX
        MOV BYTE PTR [BX], '$'  ; 在输入字符串位加结束标志$
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 5               ; 列号
        MOV DH, 7               ; 行号
        INT 10H                 ; 源字符串提示光标位置设置
        MOV AH, 9
        LEA DX, OUT_STR
        INT 21H                 ; 输出字符串提示
        MOV AH, 9
        LEA DX, KEYBUF + 2
        INT 21H                 ; 输出字符串
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 5               ; 列号
        MOV DH, 8               ; 行号
        INT 10H                 ; 新字符串提示光标位置设置
        MOV AH, 9
        LEA DX, MAXCHR
        INT 21H                 ; 输出字符串中最大值提示
        MOV DL, 0
        LEA BX, KEYBUF + 2
LCMP:
        CMP [BX], DL
        JB NOLCHG
        MOV DL, [BX]
```

```
NOLCHG:
        INC BX
        LOOP LCMP               ; 找出字符串中最大字符，放入 DL
        MOV AH, 2
        INT 21H                 ; 输出字符串中最大字符
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 5               ; 列号
        MOV DH, 9               ; 行号
        INT 10H                 ; 提示信息光标位置设置
        MOV AH, 9
        LEA DX, HINTSTR
        INT 21H                 ; 输出提示信息
        RET
MAXLTR  ENDP

SORTNUM PROC FAR                ; 对输入数据组排序
RESORT:
        ;设置显示器方式
        MOV AH, 0
        MOV AL, 3
        MOV BL, 0
        INT 10H                 ; 清屏
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 5               ; 列号
        MOV DH, 5               ; 行号
        INT 10H                 ; 输入提示光标位置设置
        MOV AH, 9
        LEA DX, IN_NUM
        INT 21H
        MOV AH, 2
        MOV DL, 5               ; 列号
        MOV DH, 6               ; 行号
        INT 10H                 ; 输入数据组光标位置设置
        MOV AH, 0AH
        LEA DX, KEYBUF
        INT 21H                 ; 输入数据组字符串
        CALL CIN_INT            ; 字符串转换成数据串
        CMP AL, 0
        JZ RESORT               ; 判断数据串是否有错
        CMP NUMBUF, 0
        JZ RESORT               ; 判断数据串是否为空
```

```
        MOV AH, 2
        MOV BH, 0                   ; 页号
        MOV DL, 5                   ; 列号
        MOV DH, 7                   ; 行号
        INT 10H                     ; 输出提示光标位置设置
        MOV AH, 9
        LEA DX, OUT_NUM
        INT 21H                     ; 输出数据串提示
        MOV AH, 2
        MOV BH, 0                   ; 页号
        MOV DL, 5                   ; 列号
        MOV DH, 8                   ; 行号
        INT 10H                     ; 输出数据组光标位置设置
        CALL FAR PTR MPSORT         ; 数据组排序
        CALL FAR PTR INT_OUT        ; 数据组的输出
        MOV AH, 2
        MOV BH, 0                   ; 页号
        MOV DL, 5                   ; 列号
        MOV DH, 9                   ; 行号
        INT 10H                     ; 提示信息光标位置设置
        MOV AH, 9
        LEA DX, HINTSTR
        INT 21H                     ; 输出提示信息
        RET
SORTNUM    ENDP

CIN_INT PROC NEAR                   ; 读入整型数
; 入口参数：无
; 出口参数为：AL（有无错误标志，0 为有，1 为无）
        MOV CL, KEYBUF + 1
        LEA SI, KEYBUF + 2
        MOV CH, 0                   ; 数据组数据个数置 0
        MOV DH, 10
        MOV AL, 0                   ; 当前数据 x=0
        MOV DL, 0                   ; 有无数据标志置 0，即无数据
FNDNUM:
        CMP BYTE PTR [SI], ' '
        JZ ADDNUM                   ; 判断当前字符是否为空格
        CMP BYTE PTR [SI], '0'
        JB ERRNUM
        CMP BYTE PTR [SI], '9'
        JA ERRNUM                   ; 判断当前字符是否在'0'-'9'之间
        MOV DL, 1                   ; 有无数据标志置 1，即有数据
        MUL DH
```

```
        XOR BH, BH
        MOV BL, [SI]
        ADD AX, BX
        SUB AX, '0'             ; 计算出当前数据 x
        CMP AH, 0
        JA ERRNUM               ; 判断 x 是否越界
        JMP NEXT
ADDNUM:
        CMP DL, 1
        JNZ NEXT                ; 判断是否有数据
        INC CH                  ; 数据组数据个数加 1
        CALL ADDNEW
        MOV DL, 0
        MOV AL, 0               ; 清零
NEXT:
        INC SI
        DEC CL
        CMP CL, 0
        JNZ FNDNUM              ; 依次检查各字符
        CMP DL, 1
        JNZ TOTAL               ; 判断是否有未加入的数据
        INC CH
        CALL ADDNEW
TOTAL:
        MOV NUMBUF, CH          ; 置数据组数据个数
        MOV AL, 1               ; 输入数据无错误
        JMP CRTNUM
ERRNUM:
        MOV AL, 0               ; 输入数据有错误
CRTNUM:
        RET
CIN_INT ENDP

ADDNEW    PROC NEAR             ; 增加新数
; 入口参数：CH（数据组数据个数）、AL（当前数据 x）
; 出口参数：无
        PUSH AX
        LEA BX, NUMBUF
        MOV AL, CH
        CBW
        ADD BX, AX
        POP AX
        MOV [BX], AL
        RET
ADDNEW          ENDP
```

```
MPSORT      PROC FAR            ; 数据组排序
            MOV AL, NUMBUF
            CMP AL, 1
            JBE NOSORT          ; 若只有一个元素，停止排序
            CBW
            MOV CX, AX
            LEA SI, NUMBUF      ; SI 指向数据组首地址
            ADD SI, CX          ; SI 指向数据组末地址
            DEC CX              ; 外循环次数
LP1:                            ; 外循环开始
            PUSH CX
            PUSH SI
            MOV DL, 0           ; 交换标志置 0
LP2:                            ; 内循环开始
            MOV AL, [SI]
            CMP AL, [SI - 1]
            JAE NOXCHG
            XCHG AL, [SI - 1]   ; 交换操作
            MOV [SI], AL
            MOV DL, 1           ; 交换标志置 1
NOXCHG:
            DEC SI
            LOOP LP2
            POP SI
            POP CX
            CMP DL, 1
            JNZ NOSORT          ; 判断交换标志
            LOOP LP1
NOSORT:RET
MPSORTENDP

INT_OUT     PROC FAR            ; 输出数据组
            MOV AL, NUMBUF
            CBW
            MOV CX, AX
            MOV BL, 10H
            LEA SI, NUMBUF + 1
PRINT:
            MOV AL, [SI]
            CALL OUTNUM
            INC SI
            MOV AH, 2
            MOV DL, ' '
            INT 21H
            LOOP PRINT
```

```
        RET
INT_OUT ENDP

OUTNUM    PROC NEAR            ; 将十进制数以十六进制输出
; 入口参数：AL（待转换的数据），BL（转换进制数 16）
; 出口参数：无
        MOV AH, 0
        DIV BL
        PUSH AX
        CMP AH, 10
        JB PNUM
        ADD AH, 7
PNUM:   ADD AH, 30H
        MOV DL, AH
        POP AX
        PUSH DX
        CMP AL, 0
        JZ OUTN
        CALL OUTNUM
OUTN:
        POP DX
        MOV AH, 2
        INT 21H
        RET
OUTNUM    ENDP

TIMCHK    PROC FAR                ; 设定并显示时间
        ;设置显示器方式
        MOV AH, 0
        MOV AL, 3;
        MOV BL, 0;
        INT 10H                   ; 清屏
        MOV AH, 2
        MOV BH, 0                 ; 页号
        MOV DL, 5                 ; 列号
        MOV DH, 6                 ; 行号
        INT 10H                   ; 设置提示光标位置设置
        MOV AH, 9
        LEA DX, IN_TIM
        INT 21H                   ; 时间串提示
        MOV AH, 0AH
        LEA DX, KEYBUF
        INT 21H                   ; 输入时间串
```

```
        MOV BL, 10
        MOV AL, KEYBUF + 2
        SUB AL, '0'
        MUL BL
        ADD AL, KEYBUF + 3
        SUB AL, '0'
        CMP AL, 0
        JB INVALID
        CMP AL, 24
        JAE INVALID             ; 判断 时 有效性
        MOV CH, AL
        MOV AL, KEYBUF + 5
        SUB AL, '0'
        MUL BL
        ADD AL, KEYBUF + 6
        SUB AL, '0'
        CMP AL, 0
        JB INVALID
        CMP AL, 60
        JAE INVALID             ; 判断 分 有效性
        MOV CL, AL
        MOV AL, KEYBUF + 8
        SUB AL, '0'
        MUL BL
        ADD AL, KEYBUF + 9
        SUB AL, '0'
        CMP AL, 0
        JB INVALID
        CMP AL, 60
        JAE INVALID             ; 判断 秒 有效性
        MOV DH, AL
        MOV DL, 0
        MOV AH, 2DH
        INT 21H                 ; 置系统时间
INVALID:
        CALL TIME
        RET
TIMCHK  ENDP

TIME    PROC                    ; 显示时间子程序
        ;设置显示器方式
        MOV AH, 0
        MOV AL, 3;
        MOV BL, 0;
        INT 10H                 ; 清屏
```

```
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 10              ; 列号
        MOV DH, 9               ; 行号
        INT 10H                 ; 提示信息光标位置设置
        MOV AH, 9
        LEA DX, HINTSTR
        INT 21H                 ; 输出提示信息
DISP1:
        MOV AH, 2
        MOV BH, 0               ; 页号
        MOV DL, 72              ; 列号
        MOV DH, 0               ; 行号
        INT 10H                 ; 提示光标位置设置
        MOV AH, 2CH             ; 取系统时间,CH,CL,DH 分别存放时/分/秒
        INT 21H
        MOV AL, CH              ; 显示 时
        CALL SHOWNUM
        MOV AH, 2
        MOV DL, ':'
        INT 21H
        MOV AL, CL              ; 显示 分
        CALL SHOWNUM
        MOV AH, 2
        MOV DL, ':'
        INT 21H
        MOV AL, DH              ; 显示 : 秒
        CALL SHOWNUM
        MOV AH,02H              ; 设置光标位置
        MOV DX,090AH
        MOV BH,0
        INT 10H
        MOV BX,0018H
RE:     MOV CX,0FFFFH           ; 延时
REA:    LOOP REA
        DEC BX
        JNZ RE
        MOV AH, 0BH             ; 或     MOV AH, 01H
        INT 21H                 ;        INT 16H
        CMP AL, 0               ;        JE DISP1
        JZ DISP1                ; 检查键盘状态
        RET
TIME    ENDP
```

```
SHOWNUM PROC                    ; 把 AL 中的数字以十进制输出
; 入口参数：AL（待显示的数据）
; 出口参数：无
        CBW
        PUSH CX
        PUSH DX
        MOV CL, 10
        DIV CL
        ADD AH, '0'
        MOV BH, AH
        ADD AL, '0'
        MOV AH, 2
        MOV DL, AL
        INT 21H
        MOV DL, BH
        INT 21H
        POP DX
        POP CX
        RET
SHOWNUM ENDP

CODE    ENDS
        END     START
```

五、实验结果

运行程序。

主菜单界面，如图 5-1。

图 5-1 主菜单界面

在主菜单界面（图 5-1）输入 1，进入功能 1 界面，实现将字符串中的小写字母变换成大写字母。功能 1 界面，如图 5-2。输入字符串，如：HuangCui02061488，输出结果如图 5-3。按 ESC 键，返回主菜单界面（图 5-1）；按其他任意键，返回功能 1 界面（图 5-2）。

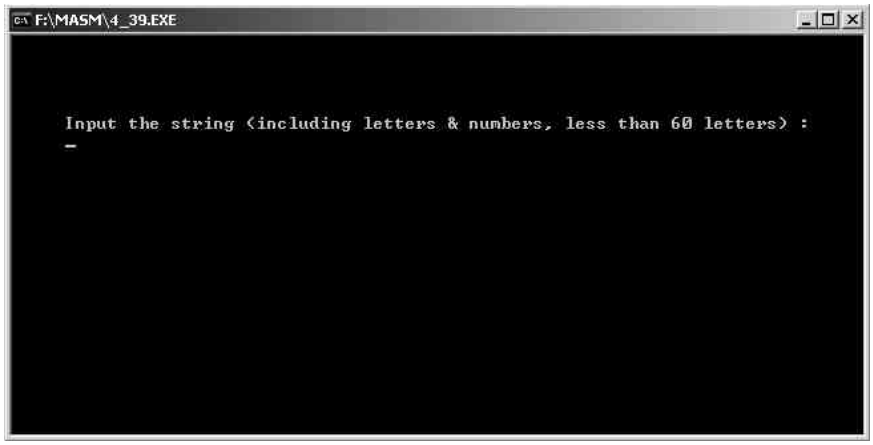


图 5-2 功能 1 界面

图 5-3 功能 1 运行结果

在主菜单界面（图 5-1）输入 2，进入功能 2 界面，实现在字符串中找最大值。功能 2 界面，如图 5-4。输入字符串，如：HuangCui02061488，输出结果如图 5-5。按 ESC 键，返回主菜单界面（图 5-1）；按其他任意键，返回功能 2 界面（图 5-4）。

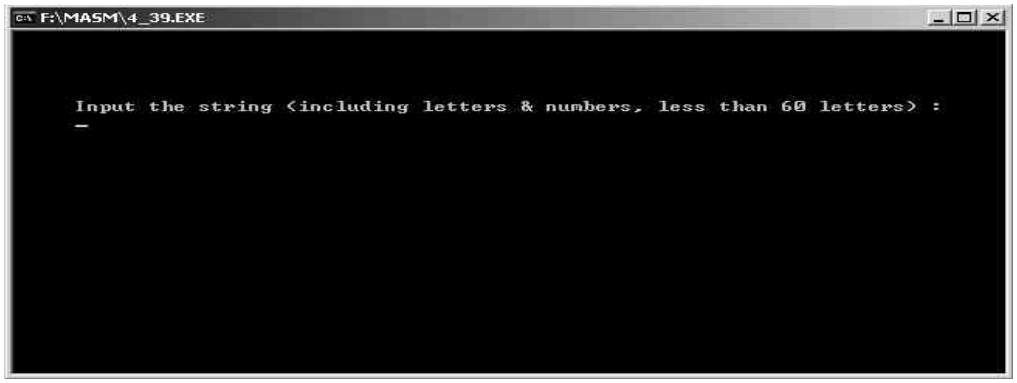


图 5-4 功能 2 界面

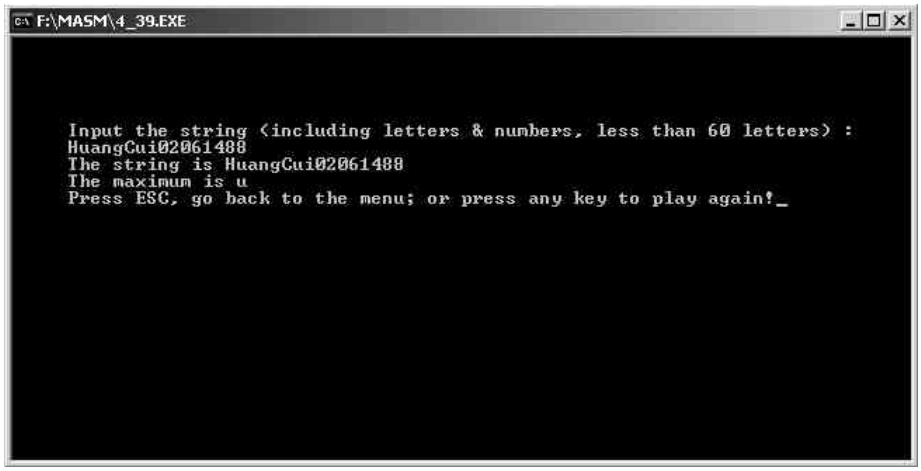


图 5-5 功能 2 运行结果

在主菜单界面（图 5-1）输入 3，进入功能 3 界面，实现输入数据组的排序，按递增方式进行排序并以十六进制数输出。功能 3 界面，如图 5-6。输入数据组，如：255 15 0 1 16，输出结果如图 5-7；若输入数据越界或有错误，则需重新输入，如图 5-6。按 ESC 键，返回主菜单界面（图 5-1）；按其他任意键，返回功能 3 界面（图 5-6）。

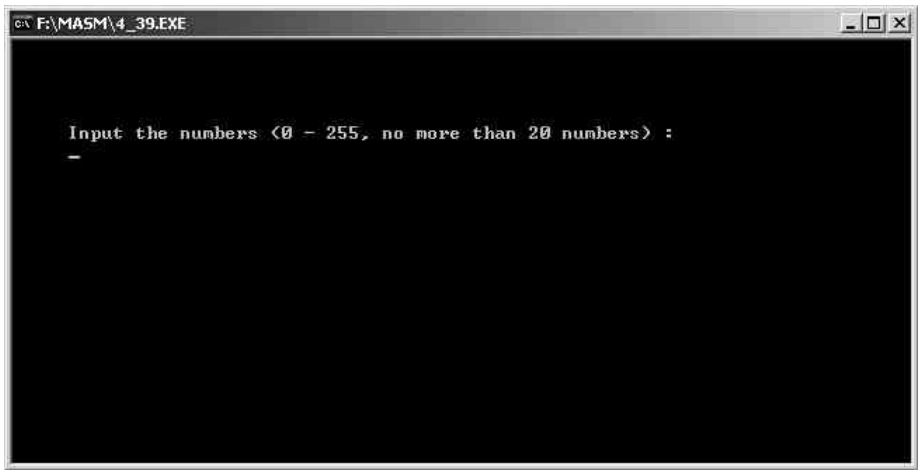


图 5-6 功能 3 界面

图 5-7 功能 3 运行结果

在主菜单界面（图 5-1）输入 4，进入功能 4 界面，实现时间的显示。功能 4 界面，如图 5-8。输入用户设置的时间；若输入时间越界或有错误，则不修改系统时间，直接进行实时显示。如：输入 12:70:00，输出结果如图 5-9（显示时间由系统当时获取时间决定，本实验时，系统时间为 18:27:36）；输入 12:00:00，输出结果如图 5-10。接着，界面对时间进行实时显示，此处仅作文字说明，不附图。按 ESC 键，返回主菜单界面（图 5-1）；按其他任意键，返回功能 4 界面（图 5-8）。

图 5-8 功能 4 界面

图 5-9 功能 4 运行结果 1

图 5-10 功能 4 运行结果 2

在主菜单界面（图 5-1）输入 5，结束程序的运行，返回操作系统。

5 章习题

1. 微处理器的外部结构表现为 数量有限的输入输出引脚 ，它们构成了微处理器级总线。

2. 微处理器级总线经过形成电路之后形成了 系统级总线 。

3. 简述总线的定义及在计算机系统中采用标准化总线的优点。

答：总线是计算机系统中模块（或子系统）之间传输数据、地址和控制信号的公共通道，它是一组公用导线，是计算机系统的重要组成部分。

采用标准化总线的优点是：

1） 简化软、硬件设计。

2） 简化系统结构。

3） 易于系统扩展。

4） 便于系统更新。

5） 便于调试和维修。

4. 在微型计算机应用系统中，按功能层次可以把总线分成哪几类。

答：在微型计算机应用系统中，按功能层次可以把总线分成：片内总线、元件级总线、系统总线和通信总线。

5. 简述 RESET 信号的有效形式和系统复位后的启动地址。

答：RESET 为系统复位信号，高电平有效，其有效信号至少要保持四个时钟周期，且复位信号上升沿要与 CLK 下降沿同步。

系统复位后的启动地址为 0FFFF0H。即：（CS）=0FFFFH，（IP）=0000H。

6. 8086 CPU 的 $M/\overline{IO}$ 信号在访问存储器时为 高 电平，访问 I/O 端口时为 低 电平。

7. 在 8086 系统总线结构中，为什么要有地址锁存器？

答：8086CPU 有 20 条地址线和 16 条数据线，为了减少引脚，采用了分时复用，共占了 20 条引脚。这 20 条引脚在总线周期的 T1 状态输出地址。为了使地址信息在总线周期的其他 T 状态仍保持有效，总线控制逻辑必须有一个地址锁存器，把 T1 状态输出的 20 位地址信息进行锁存。

8. 根据传送信息的种类不同，系统总线分为 数据总线 、 地址总线 和 控制总线 。

9. 三态逻辑电路输出信号的三个状态是 高电平 、 低电平 和 高阻态 。

10. 在 8086 的基本读总线周期中，在 T_1 状态开始输出有效的 ALE 信号；在 T_2 状态开始输出

低电平的$\overline{RD}$信号，相应的$\overline{DEN}$为__低__电平，$DT/\overline{R}$为__低__电平；引脚AD_{15}~AD_0上在T_1状态期间给出地址信息，在T_4状态完成数据的读入。

11. 利用常用芯片 74LS373 构成 8086 系统的地址总线， 74LS245 作为总线收发器构成数据总线，画出 8086 最小方式系统总线形成电路。

答：8086 最小方式系统总线形成电路如图 5.1 所示。

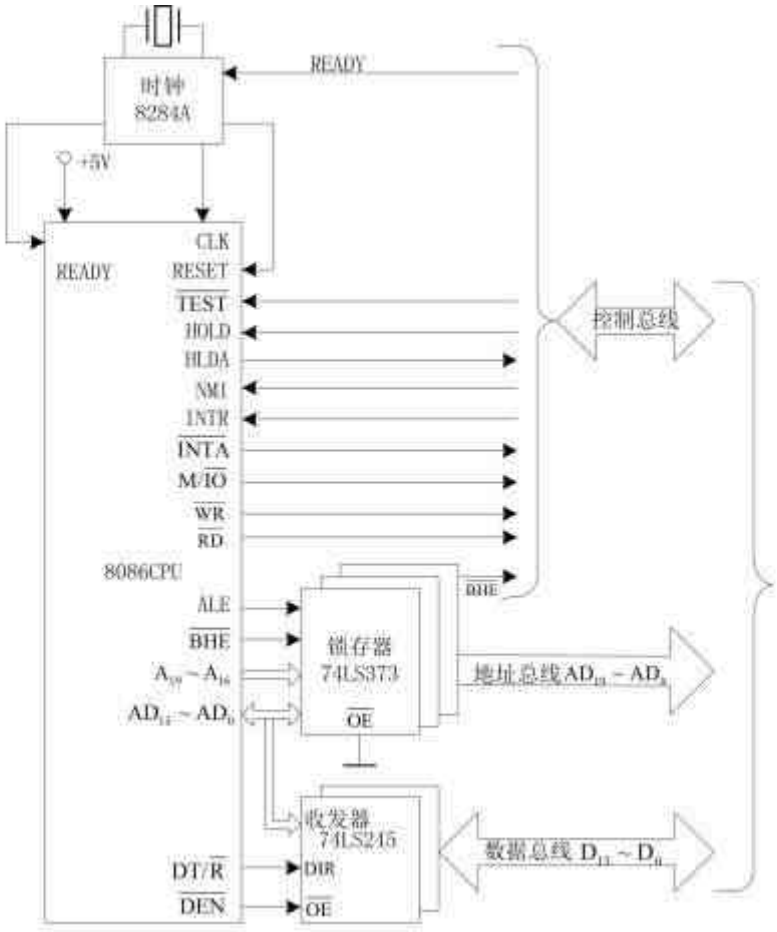


图 5.1 8086 最小方式系统总线形成电路

12. 微机中的控制总线提供__H__。

A. 数据信号流；

B. 存储器和 I/O 设备的地址码；

C. 所有存储器和 I/O 设备的时序信号；

D. 所有存储器和 I/O 设备的控制信号；

E. 来自存储器和 I/O 设备的响应信号；

F. 上述各项；

G. 上述 C，D 两项；

H. 上述 C，D 和 E 三项。

13. 微机中读写控制信号的作用是 E 。

A. 决定数据总线上数据流的方向；

B. 控制存储器操作读/写的类型；

C. 控制流入、流出存储器信息的方向；

D. 控制流入、流出 I/O 端口信息的方向；

E. 以上所有。

14. 8086 CPU 工作在最大方式，引脚 $MN/\overline{MX}$ 应接 地 。

15. RESET 信号在至少保持 4 个时钟周期的 高 电平时才有效，该信号结束后，CPU 内部的 CS 为 0FFFFH ，IP 为 0000H ，程序从 0FFFF0H 地址开始执行。

16. 在构成 8086 最小系统总线时，地址锁存器 74LS373 的选通信号 G 应接 CPU 的 ALE 信号，输出允许端 $\overline{OE}$ 应接 地 ；数据收发器 74LS245 的方向控制端 DIR 应接 $DI/\overline{R}$ 信号，输出允许端 $\overline{E}$ 应接 $\overline{DEN}$ 信号。

17. 8086 CPU 在读写一个字节时，只需要使用 16 条数据线中的 8 条，在 一 个总线周期内完成；在读写一个字时，自然要用到 16 条数据线，当字的存储对准时，可在 一 个总线周期内完成；当字的存储为未对准时，则要在 两 个总线周期内完成。

18. CPU 在 T_3 状态开始检查 READY 信号， 高 电平时有效，说明存储器或 I/O 端口准备就绪，下一个时钟周期可进行数据的读写；否则，CPU 可自动插入一个或几个 等待周期（T_W ），以延长总线周期，从而保证快速的 CPU 与慢速的存储器或 I/O 端口之间协调地进行数据传送。

19. 8086 最大系统的系统总线结构较最小系统的系统总线结构多一个芯片 8288 总线控制器 。

20. 微机在执行指令 MOV [DI]，AL 时，将送出的有效信号有 B C 。

A．RESET　　B.高电平的 $M/\overline{IO}$ 信号　C. $\overline{WR}$　　D. $\overline{RD}$

21. 设指令 MOV AX，DATA 已被取到 CPU 的指令队列中准备执行，并假定 DATA 为偶地址，试画出下列情况该指令执行的总线时序图:

（1）没有等待的 8086 最小方式;

（2）有一个等待周期的 8086 最小方式。

答：（1）没有等待的 8086 最小方式时序如图 5.2 所示。

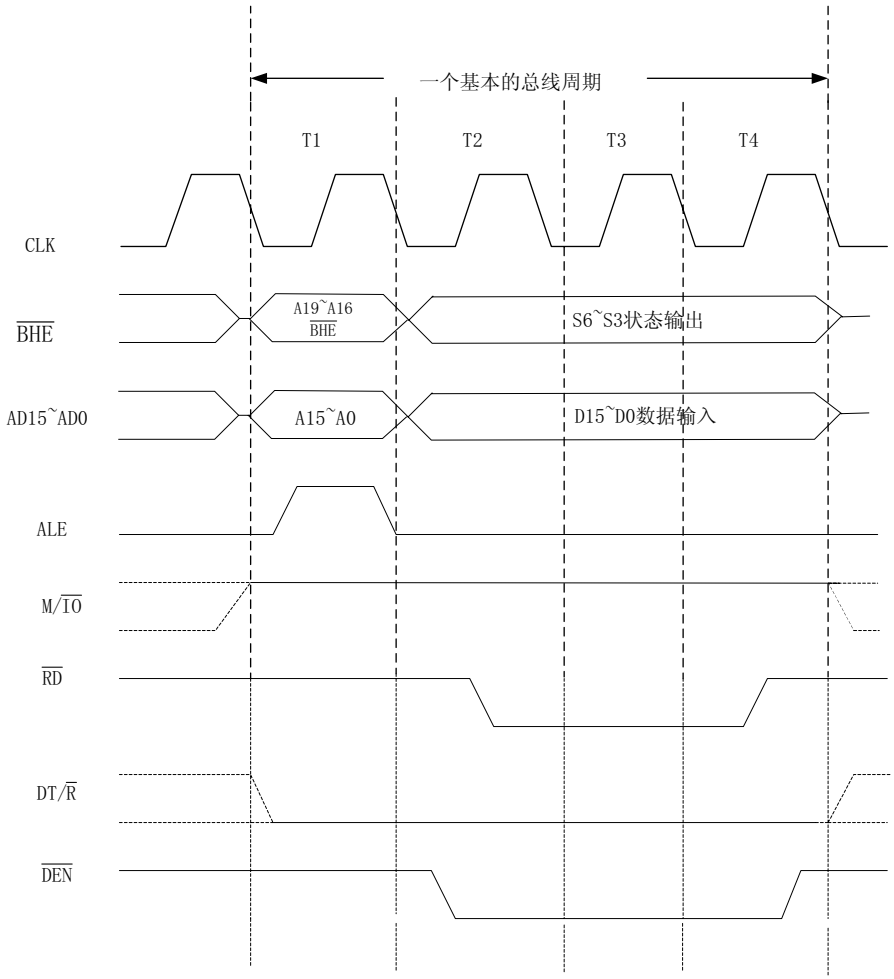


图 5.2 没有等待的 8086 最小方式时序

（2）有一个等待周期的 8086 最小方式时序图如图 5.3 所示。

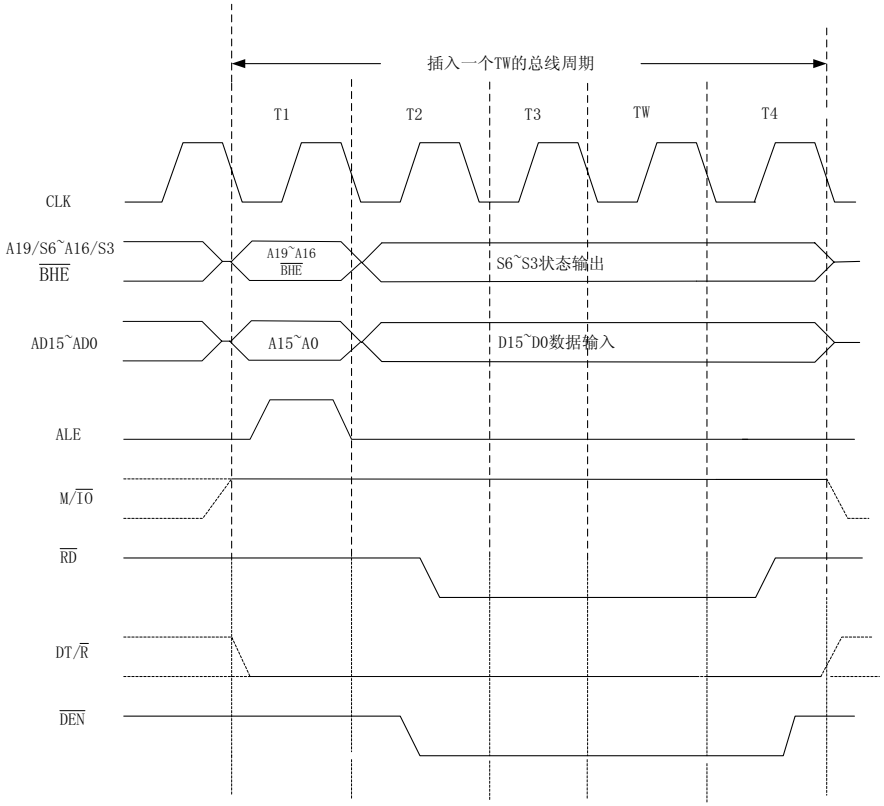


图 5.3 有一个等待周期的 8086 最小方式时序图

22. 上题中如果指令分别为：

（1） MOV DATA+1，AX

（2） MOV DATA+1，AL

（3） OUT DX，AX （DX 的内容为偶数 ）

（4） IN AL，0F5H

重做上题（1）。

答：（1）因为 DATA 为偶地址，则 DATA+1 为奇地址。故要完成本条指令，需要两个总线周期。时序图如图 5.4 所示。

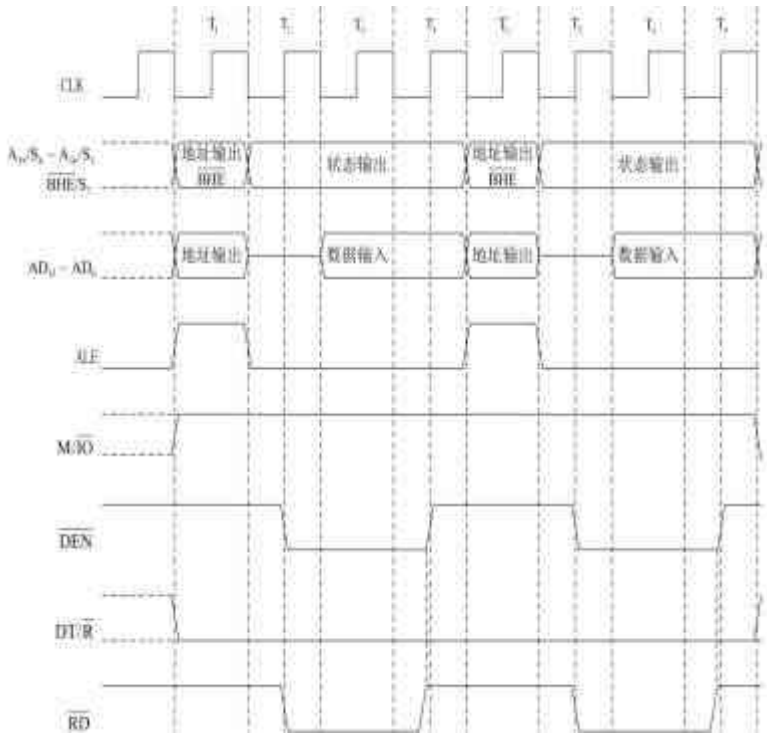


图 5.4　执行 MOV　DATA+1，AX 指令的时序参考图

（2）　DATA+1 虽然为奇地址，但是 AL 为八位存储器，故本条指令需用一个总线周期，时序图如图 5.5 所示。

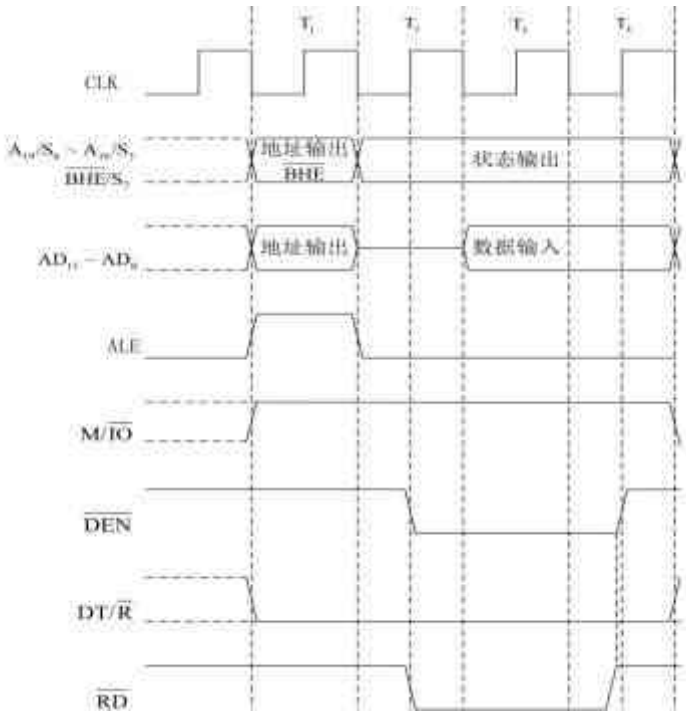


图 5.5 执行 MOV　DATA+1，AL 指令的时序参考图

（3）　执行 OUT　DX，AX（DX 的内容为偶数 ）指令的时序图如图 5.6 所示。

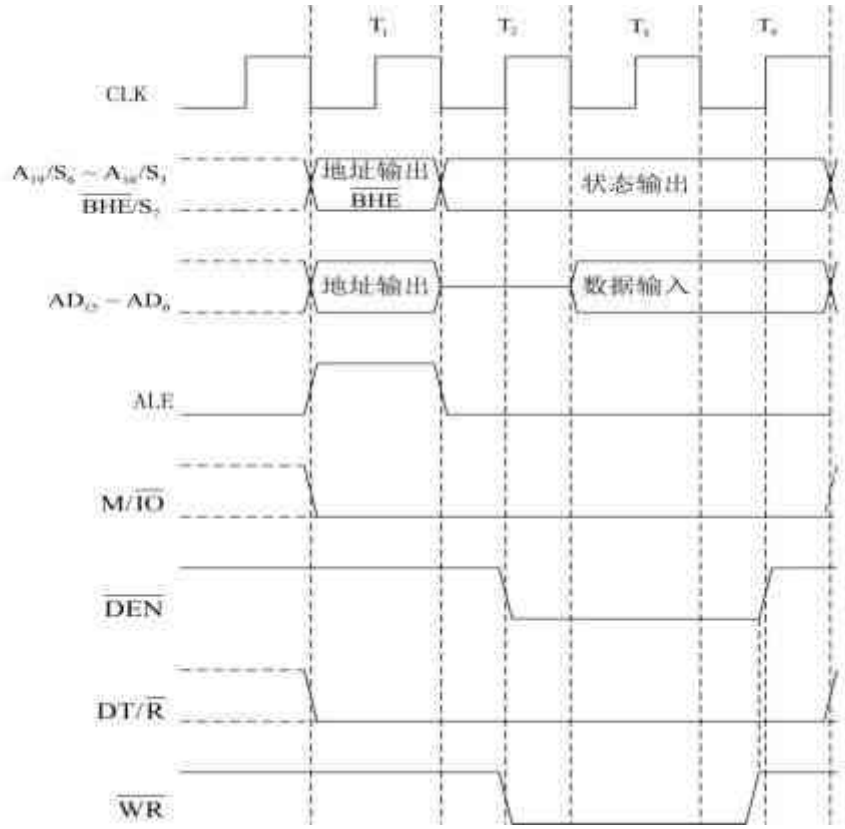


图 5.6 执行 OUT　DX，AX 指令的时序参考图

（4）　执行 IN AL，0F5H 指令的时序图如图 5.7 所示。

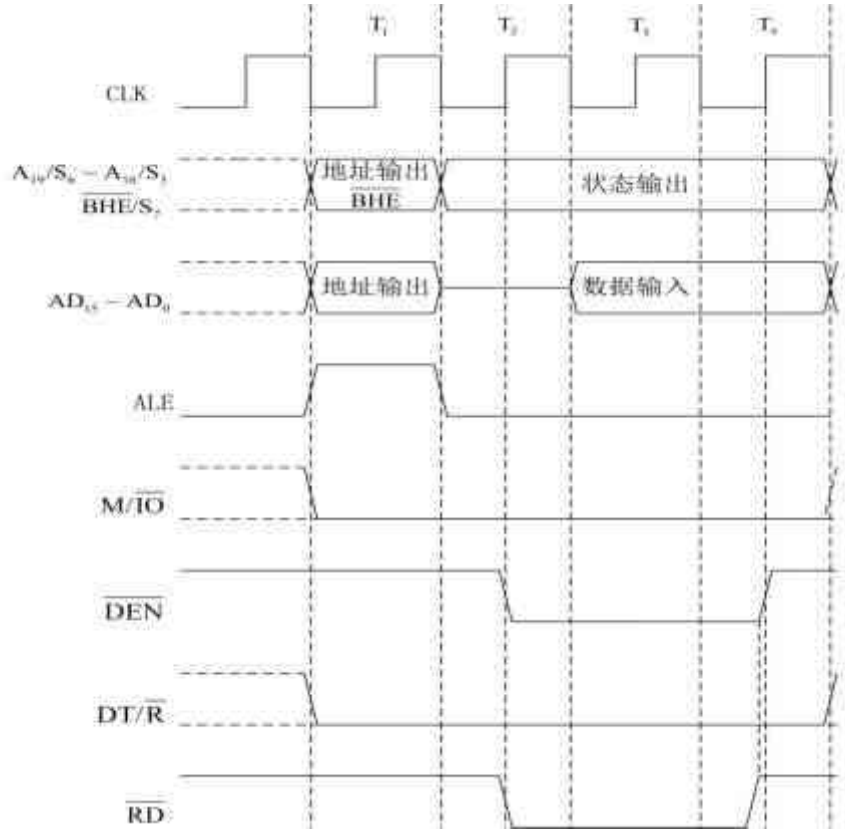


图 5.7 执行 IN AL，0F5H 指令的时序参考图

23．8086 最小方式下，读总线周期和写总线周期相同之处是：在 T_1 状态开始使 ALE 信号变为有效 高 电平，并输出 $M/\overline{IO}$ 信号来确定是访问存储器还是访问 I/O 端口，同时送出 20 位有效地址，在 T_1 状态的后部，ALE 信号变为 低 电平，利用其下降沿将 20 位地址和 $\overline{BHE}$ 的状态锁存在地址锁存器中；相异之处从 T_2 状态开始的数据传送阶段。

6 章习题

1. 简述内存储器的分类及每种存储器的用途？

解：内存储器按其工作方式的不同，可以分为随机存取存储器（简称随机存储器或 RAM）和只读存储器（简称 ROM）。

随机存储器。随机存储器允许随机的按任意指定地址向内存单元存入或从该单元取出信息，对任一地址的存取时间都是相同的。由于信息是通过电信号写入存储器的，所以断电时 RAM 中的信息就会消失。计算机工作时使用的程序和数据等都存储在 RAM 中，如果对程序或数据进行了修改之后，应该将它存储到外存储器中，否则关机后信息将丢失。通常所说的内存大小就是指 RAM 的大小，一般以 KB 或 MB 为单位。

只读存储器。只读存储器是只能读出而不能随意写入信息的存储器。ROM 中的内容是由厂家制造时用特殊方法写入的，或者要利用特殊的写入器才能写入。当计算机断电后，ROM 中的信息不会丢失。当计算机重新被加电后，其中的信息保持原来的不变，仍可被读出。ROM 适宜存放计算机启动的引导程序、启动后的检测程序、系统最基本的输入输出程序、时钟控制程序以及计算机的系统配置和磁盘参数等重要信息。

2.简述存储器的主要技术指标有哪些？

解：存储器的主要技术指标有：存储容量、读写速度、非易失性、可靠性等。

3.在实际工程应用中，存储器芯片的速度怎样估算？

解：在选择存储器芯片时应注意是否与微处理器的总线周期时序匹配。作为一种保守的估计，在存储器芯片的手册中可以查得最小读出周期 t_{cyc}(R)(Read Cycle Time)和最小写周期 t_{cyc}(W)(Write Cycle Time)。如果根据计算，微处理器对存储器的读写周期都比存储器芯片手册中的最小读写周期大，那么我们认为该存储器芯片是符合要求的，否则要另选速度更高的存储器芯片。

8086CPU 对存储器的读写周期需要 4 个时钟周期(一个基本的总线周期)。因此，作为一种保守的工程估计，存储器芯片的最小读出时间应满足如下表达式：

$$t_{cyc}(R) < 4T - t_{da} - t_D - T$$

其中：T 为 8086 微处理器的时钟周期；t_{da} 为 8086 微处理器的地址总线延时时间；t_D 为各种因素引起的总线附加延时。这里的 t_D 应该认为是总线长度、附加逻辑电路、总线驱动器等引起的延时时间总和。

同理，存储器芯片的最小写入时间应满足如下表达式：

$$t_{cyc}(W) < 4T - t_{da} - t_D - T$$

4.用下列 RAM 芯片构成 32kB 存储器模块，各需多少芯片？16 位地址总线中有多少位参与片

内寻址？多少位可用作片选控制信号？

(1)1k×1　(2)1k×4　(3)4k×8　(4)16k×4

解：(1)1k×1

$\frac{32K\times8}{1K\times1}=256$ 片，

片内寻址：A_0□　，共 10 位；　片选控制信号：A_{10}□　，共 6 位。

(2)1k×4

$\frac{32K\times8}{1K\times4}=64$ 片，

片内寻址：A_0□　，共 10 位；　片选控制信号：A_{10}□　，共 6 位。

(3)4k×8

$\frac{32K\times8}{4K\times8}=8$ 片，

片内寻址：A_0□　，共 12 位；　片选控制信号：A_{12}□　，共 4 位。

(4)16k×4

$\frac{32K\times8}{16K\times4}=4$ 片，

片内寻址：A_0□　，共 14 位；　片选控制信号：$A_{14}A_{15}$，共 2 位。

5.若存储器模块的存储容量为 256kB，则利用上题中给出的 RAM 芯片，求出构成 256kB 存储模块各需多少块芯片？20 位地址总线中有多少位参与片内寻址？多少位可用作片选控制信号？

解：(1)1k×1

$\frac{256K\times8}{1K\times1}=2048$ 片，

片内寻址：A_0□　，共 10 位；　片选控制信号：A_{10}□　，共 10 位。

(2)1k×4

$\frac{256K\times8}{1K\times4}=512$ 片，

片内寻址：A_0□　，共 10 位；　片选控制信号：A_{10}□　，共 10 位。

(3)4k×8

$\frac{256K\times8}{4K\times8}=64$ 片，

片内寻址：A_0□ ，共 12 位；　　片选控制信号：A_{12}□ ，共 8 位。

(4)16k×4

$\frac{256K\times8}{16K\times4}=32$ 片，

片内寻址：A_0□ ，共 14 位；　　片选控制信号：A_{14}□ ，共 6 位。

6.一台 8 位微机系统的地址总线为 16 位，其存储器中 RAM 的容量为 32kB，首地址为 4000H，且地址是连接的。问可用的最高地址是多少？

解：32K=2^{15}=8000H，所以，最高地址为：

4000H+8000H-1=BFFFH

则，可用的最高地址为 0BFFFH.

7.某微机系统中内存的首地址为 4000H，末地址为 7FFFH，求其内存容量。

解：7FFFH-4000H+1=4000H=2^{14}=16KB

内存容量为 16KB。

8. 利用全地址译码将 6264 芯片接在 8088 的系统总线上，其所占地址范围为 00000H~03FFFH，试画连接图。写入某数据并读出与之比较，若有错，则在 DL 中写入 01H；若每个单元均对，则在 DL 写入 EEH，试编写此检测程序。

解：因为 6264 的片容量为 8KB。

RAM 存储区域的总容量为 03FFFH-00000H+1=4000H=16KB，故需要 2 片 6264 芯片。

连接图如图 6.1 所示。

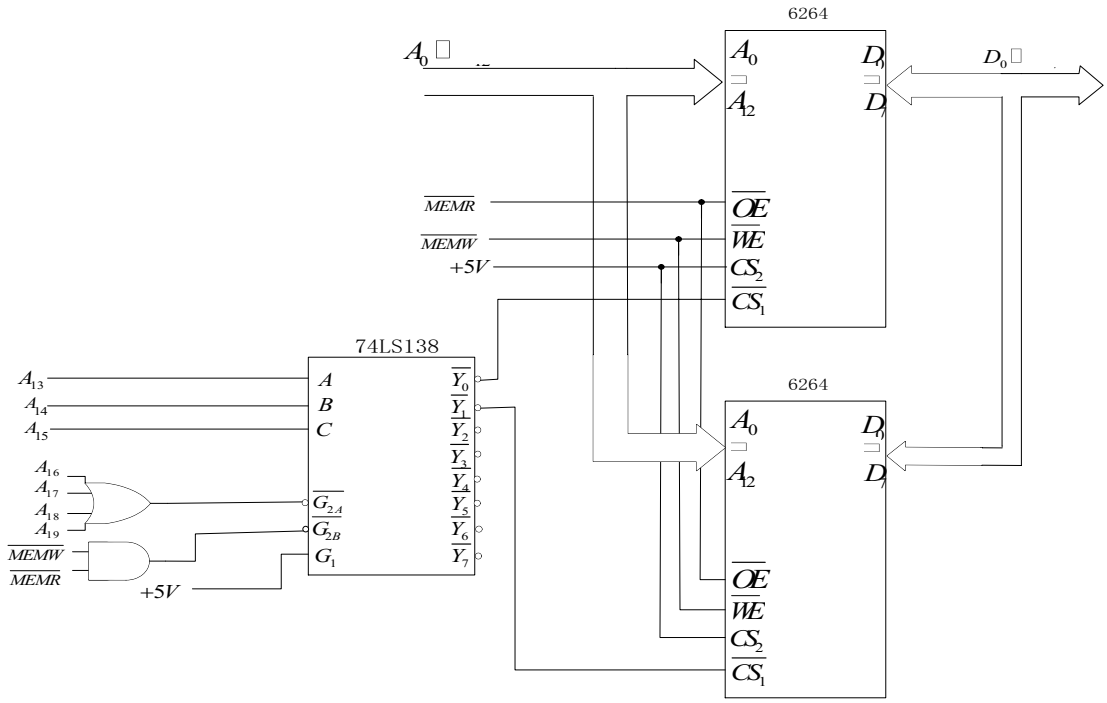


图 6.1 与 8088 系统总线的连接图

检测程序段：

```
        MOV AX,0000H
        MOV DS,AX
        MOV SI,0
        MOV CX,16*1024
        MOV AL,55H
CMPL:   MOV [SI],AL
        MOV BL,[SI]
        CMP BL,AL
        JNE ERROR
        INC SI
        LOOP CMPL
        MOV DL,0EEH
        JMP NEXT
ERROR: MOV DL,01H
NEXT:
      …
```

9.简述 EPROM 的编程过程，并说明 EEPROM 的编程过程。

解：EPROM 芯片的编程有两种方式：标准编程和快速编程。

在标准编程方式下，每给出一个编程负脉冲就写入一个字节的数据。Vpp 上加编程电压，地址线、数据线上给出要编程单元的地址及其数据，并使$\overline{CE}$=0, $\overline{OE}$=1。上述信号稳定后，在$\overline{\text{PGM}}$ 端加上宽度为 50ms±5ms 的负脉冲，就可将数据逐一写入。写入一个单元后将$\overline{OE}$变低，可以对刚写入的数据读出进行检验。

快速编程使用 100μs 的编程脉冲依次写完所有要编程的单元，然后从头开始检验每个写入的字节。若写的不正确，则重写此单元。写完再检验，不正确可重写。

EEPROM 编程时不需要加高电压，也不需要专门的擦除过程。并口线 EEPROM 操作与 SRAM 相似，写入时间约 5ms。串行 EEPROM 写操作按时序进行，分为字节写方式和页写方式。

10. 若要将 4 块 6264 芯片连接到 8088 最大方式系统 A0000H~A7FFFH 的地址空间中，现限定要采用 74LS138 作为地址译码器，试画出包括板内数据总线驱动的连接电路图。

解：8088 最大方式系统与存储器读写操作有关的信号线有：地址总线A_0 ，数据总线：D_0 ，控制信号：$\overline{MEMR},\overline{MEMW}$ 。

根据题目已知条件和 74LS138 译码器的功能，设计的板内数据总线驱动电路如图 6.2(a)所示，板内存储器电路的连接电路图如图 6.2 （b）所示。

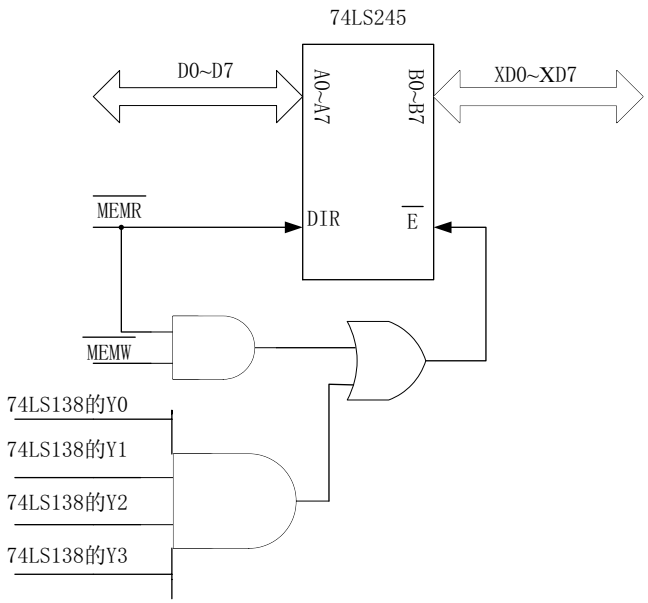


图 6.2（a）板内数据总线驱动电路

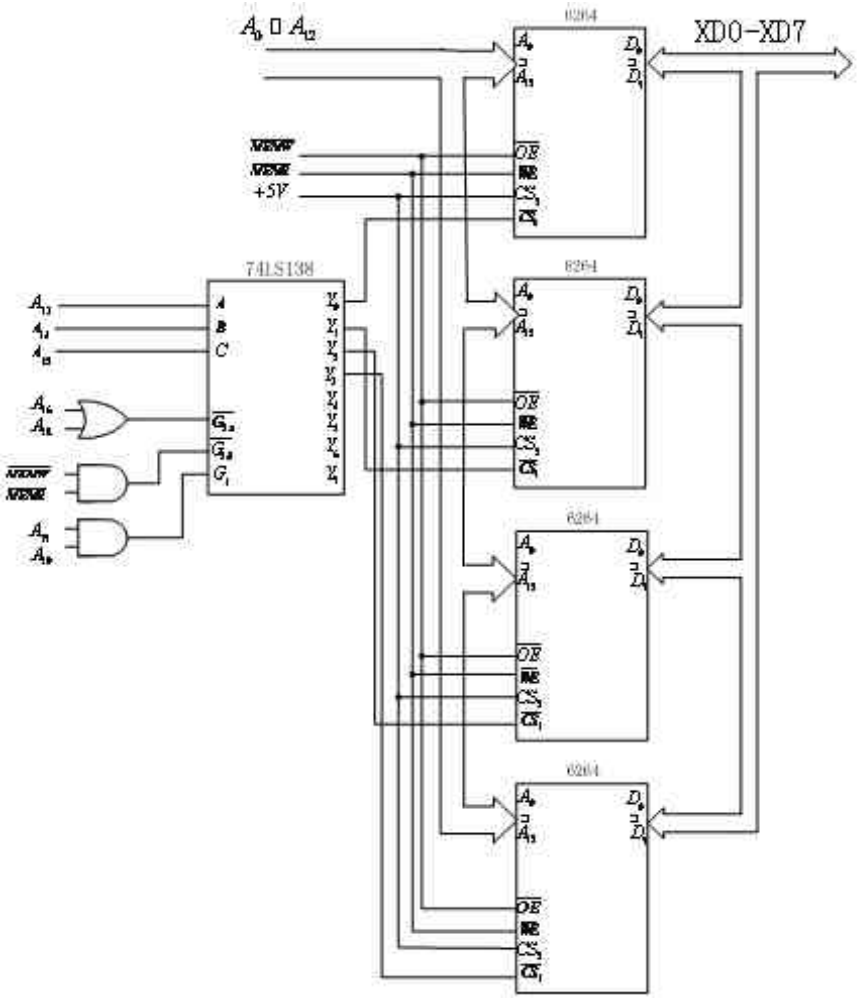


图 6.2 （b）板内存储器电路的连接图

11.若在某 8088 微型计算机系统中，要将一块 2764 芯片连接到 E0000H~E7FFFH 的空间中去，利用局部译码方式使它占有整个 32kB 的空间，试画出地址译码电路及 2764 芯片与总线的连接图。

解：Intel 2764 的片容量为 8KB，而题目给出的地址共 32KB，说明有 4 个地址区重叠，即采用部分地址译码时，有 2 条高位地址线不参加译码（即 A_{13}，A_{14} 不参加译码）。

地址译码电路及 2764 与总线的连接如图 6.3 所示。

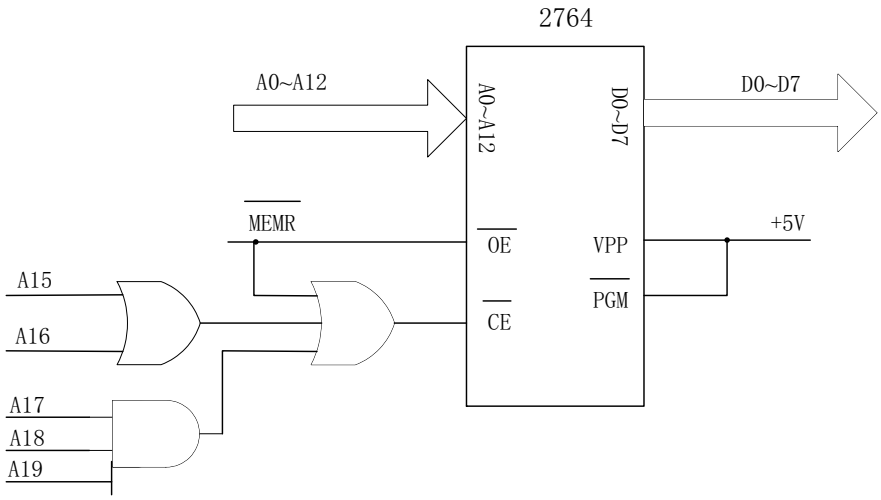


图 6.3 地址译码电路及 2764 与总线的连接

12．在 8086 CPU 工作在最小方式组成的微机系统中，扩充设计 16kB 的 SRAM 电路，SRAM 芯片选用 Intel 6264,内存地址范围为 70000H~73FFFH，试画出此 SRAM 电路与 8086 系统总线的连接图。

解：73FFFH-70000H+1=4000H=16K

Intel 6264 的片容量为 8KB，RAM 存储区总容量为 16KB，故需要 2 片 6264.

8086 最小方式系统与存储器读写操作有关的信号线有：地址总线 A_0□ ，数据总线：

D_0□ ，控制信号：$M/\overline{IO},\overline{RD},\overline{WR},\overline{BHE}$ 。

此 SRAM 电路与 8086 系统总线的连接图如图 6.4 所示。

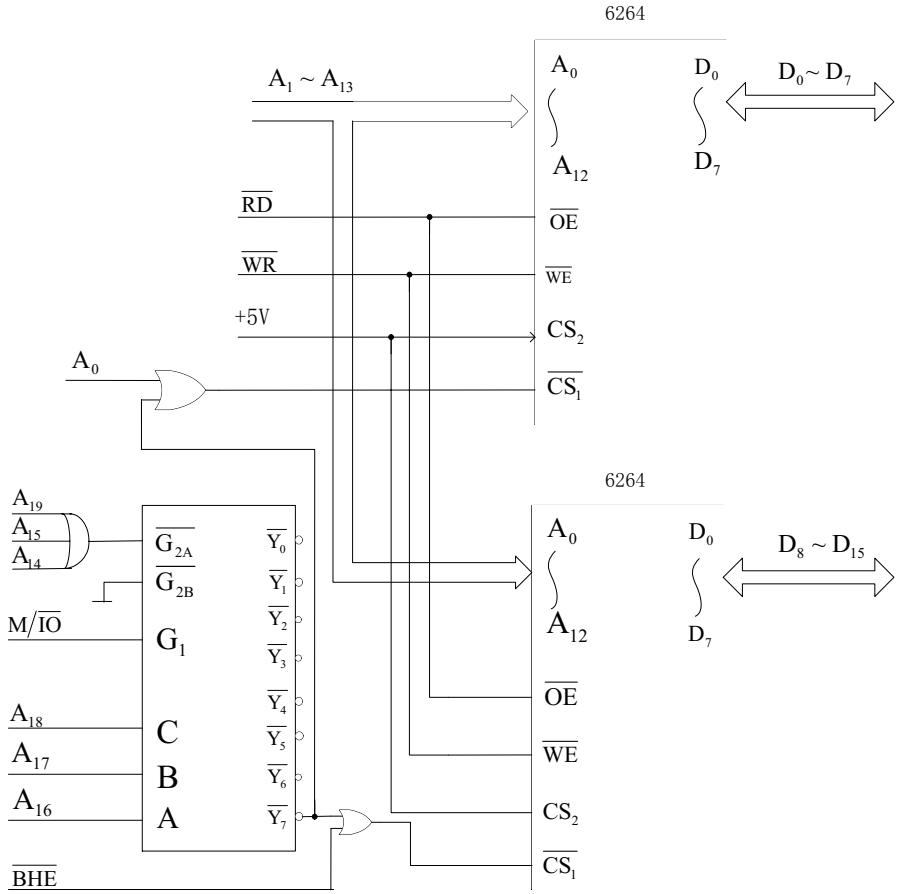


图 6.4 SRAM 电路与 8086 系统总线的连接图

13.E^2PROM 28C16 芯片各引脚的功能是什么？如果要将一片 28C16 与 8088 系统总线相连接，并能随时改写 28C16 中各单元的内容，试画出 28C16 和 8088 系统总线的连接图(地址空间为 40000H~407FFH)。

解：28C16 的引脚功能：

- VCC，GND：电源和地
- $A_{10} \sim A_0$：11 位地址线，可寻址 2KB 地址空间
- $D_7 \sim D_0$：8 位数据线
- $\overline{WE}$：写允许，低电平有效。
- $\overline{OE}$：输出允许，低电平有效。
- $\overline{CE}$：片选信号，低电平有效。

根据所学知识，28C16 与 8088 系统的连接图如图 6.5 所示。

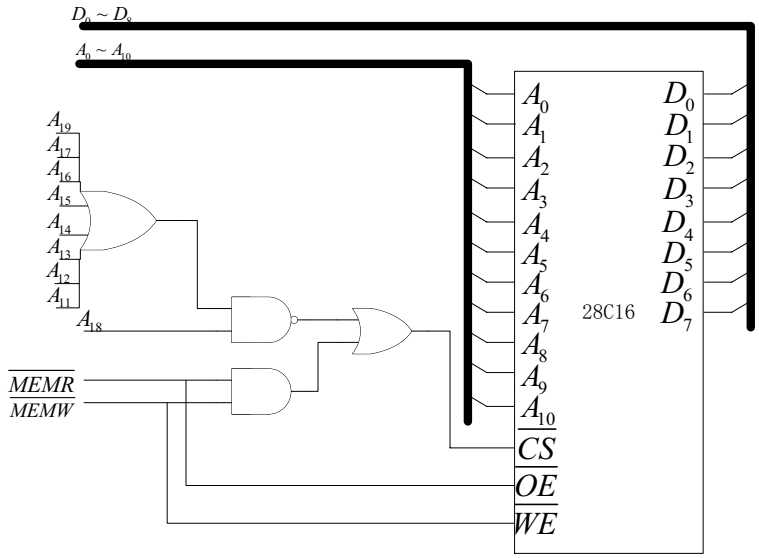


图 6.5 28C16 与 8088 系统的连接图

7 章习题

1. 简述 I/O 接口的基本功能。

答：（1）地址选择 （2）控制功能 （3）状态指示 （4）速度匹配

（5）转换信息格式 （6）电平转换 （7）可编程性

2. 简述 I/O 接口与 I/O 端口的区别。

答：I/O 接口是指 I/O 设备与系统总线之间的连接部件。

I/O 端口是指 I/O 接口内部可由 CPU 进行读写操作的各种寄存器，根据存放信息的不同，这些寄存器分别称为数据端口、控制端口和状态端口。

3. 简述 I/O 端口的编址方式及优缺点。

答：I/O 端口编址的方式可以分为独立编址和统一编址两种方式。

独立编址方式是指 I/O 端口与存储器有相互独立的地址空间。

统一编址方式是指 I/O 端口与存储器共享一个地址空间，所有的存储单元只占用其中的一部分地址，而 I/O 端口则占用另外一部分地址。

优缺点：独立编址方式的优点之一是存储器的容量可以达到与地址总线所决定的地址空间相同；优点之二是访问 I/O 端口时的地址位数可以较少，提高总线的利用率。但是缺点是必须设置专门的 I/O 指令，增加了指令系统和有关硬件的复杂性。

与独立编址方式相比，统一编址方式的优点是无需专门的 I/O 指令，从而使编程较灵活，但是 I/O 端口占用了存储器的一部分地址空间，因而影响到系统中的存储器的容量，并且访问存储器和访问 I/O 端口必须使用相同位数的地址，使指令地址码加长，总线中传送信息量增加。

4. 简述程序查询、中断和 DMA 三种方式的优缺点。

答：程序查询方式的优点在于可以防止数据的丢失，实现起来较为简单；缺点是它占用了微处理器的大量时间，实时性较差。

中断方式具有较好的实时性；但在一定程度上增加成本和复杂性。

DMA 方式的突出优点是传送过程无须处理器的控制，数据也无须经过微处理器，而是直接在 I/O 设备与主存储器间进行，因此既节约了微处理器的时间，也使传送速率大大提高；缺点是输入/输出操作占用微处理器时间，而且很难达到较高的数据传输率。

5. 8086 CPU 有 20 条地址总线,可形成 1MB 的存储器地址空间,可寻址范围为 00000H--FFFFFH;地址总线中的 16 条线可用于 I/O 寻址,形成 64KB 的输入输出地址空间,地址范围为 0000H--FFFFH ;PC 机中用了 10 条地址线进行 I/O 操作,其地址空间为 1KB ,可寻址范围为 000H—3FFH 。

6. 对于微机而言，任何新增的外部设备，最终总是要通过 I/O 接口 与主机相接。

7. 在主机板外开发一些新的外设接口逻辑，这些接口逻辑的一侧应与 I/O 设备 相接，另一侧与 系统总线 相接。

8. 需要靠在程序中排入 I/O 指令完成的数据输入输出方式有 B C 。

（A）DMA　　（B）程序查询方式　　（C）中断方式

9. 8086CPU 用 IN 指令从端口读入数据，用 OUT 指令向端口写入数据。

10. 在 8088 CPU 组成的计算机系统中有一接口模块，片内占用 16 个端口地址 300~30FH，设计产生片选信号的译码电路。

解：由于片内有 16 个端口，非别占用 300~30FH 地址。因此，该接口模块的片选信号的译码电路设计时，A3~A0 不参加译码。其译码电路如图 7.1 所示。

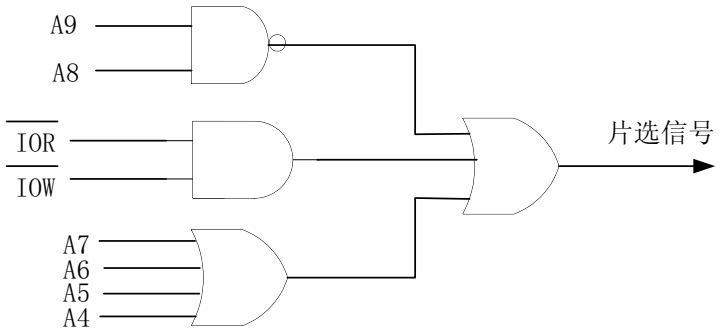


图 7.1

11. 在 IBM PC 系统中，如果 AEN 信号未参加 I/O 端口地址译码，会出现什么问题？在没有 DMA 机构的其它微机系统中，是否存在同样的问题？

答：在 IBM PC 系统中，如果 AEN 信号未参加 I/O 端口地址译码，则会出现 DMA 机构与 I/O 端口竞争总线的问题。在没有 DMA 机构的其他微机系统中，不会存在同样的问题。

12. 在 8088 CPU 工作在最大方式组成的微机系统中，利用 74LS244 设计一个输入端口，分配给该端口的地址为 04E5H，试画出连接图。

解：连接图如图 7.2 所示。

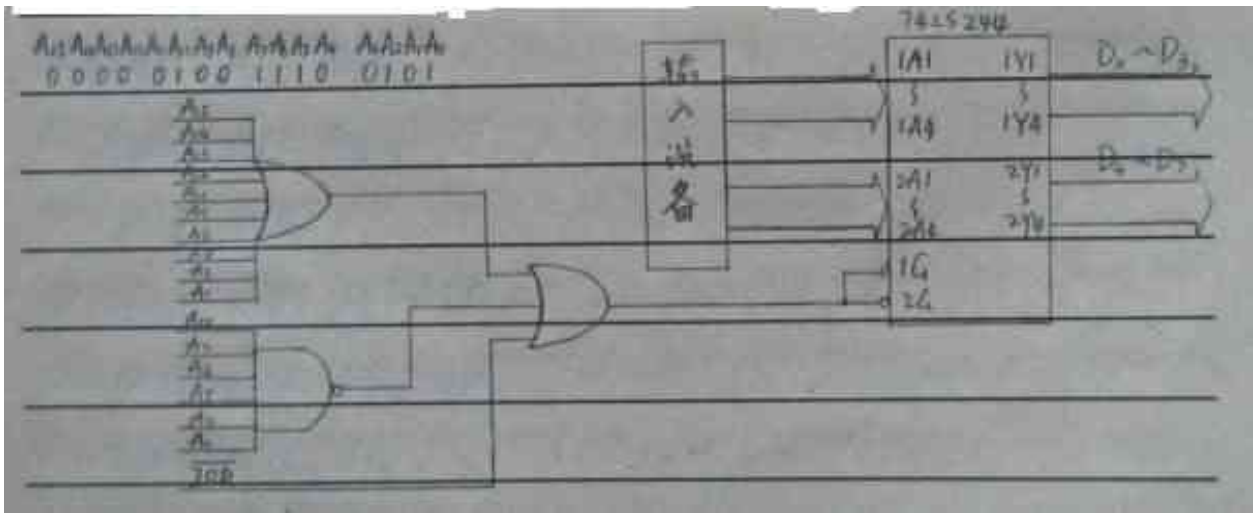


图 7.2

13. 在上题的基础上，利用 74LS374 设计一个输出端口，分配给该端口的地址为 E504H，试画出连接图。若上题中输入端口的 bit3、bit4 和 bit7 同时为 1，将内存 BUFFER 开始的连续 10 个字节单元的数据由 E504H 端口输出；若不满足条件，则等待。试编写程序。

解：连接图如图 7.3 所示。

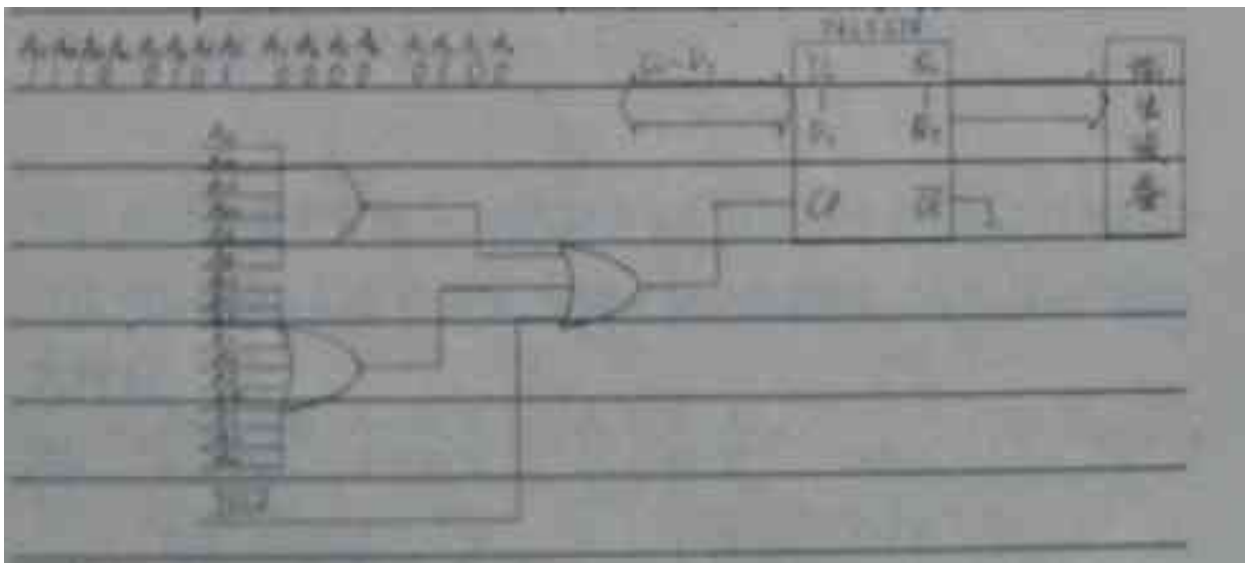


图 7.3

程序如下：

```
        MOV   CX ,10
        LEA   SI,BUFFER
        MOV   DX,04E5H
WAIT1：IN    AL,DX
        AND   AL,98H
        CMP   AL,98H
```

```
    JNZ    WAIT1
    MOV DX,0E504H
L1：MOV    AL,[SI]
    OUT    DX,AL
    INC     SI
    LOOP   L1
    HLT
```

14．在 8086 最大系统中，分别利用 2 片 74LS244 和 74LS273 设计 16 位输入和输出接口，其起始端口地址为 504H、506H，画出硬件连接图

解：硬件连接图如图 7.4 所示。

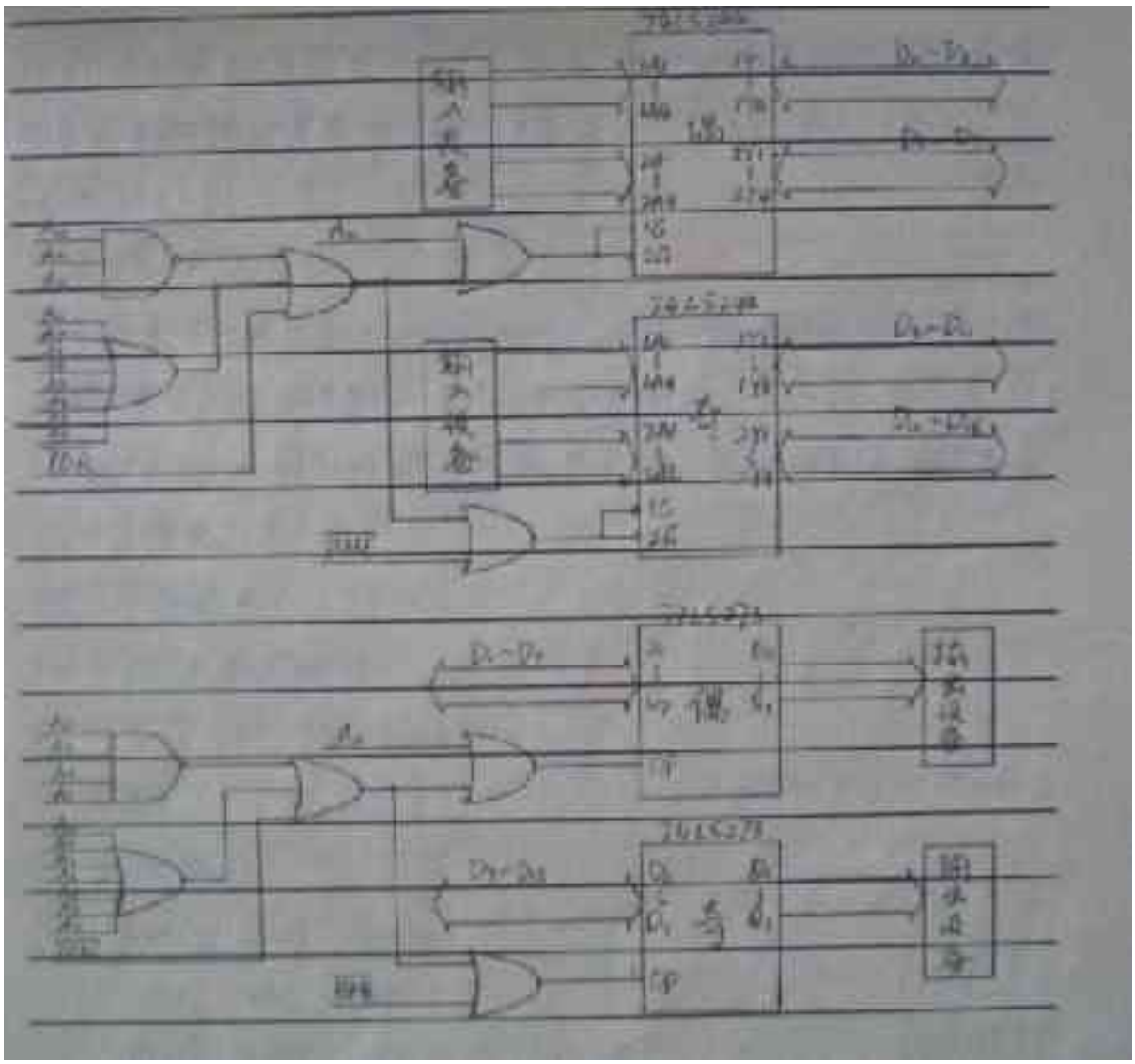


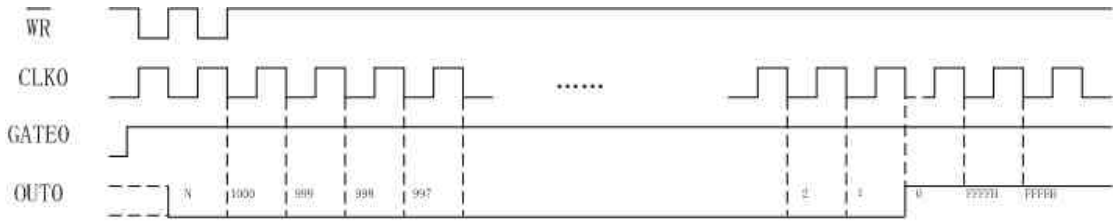
图 7.4 硬件连接图

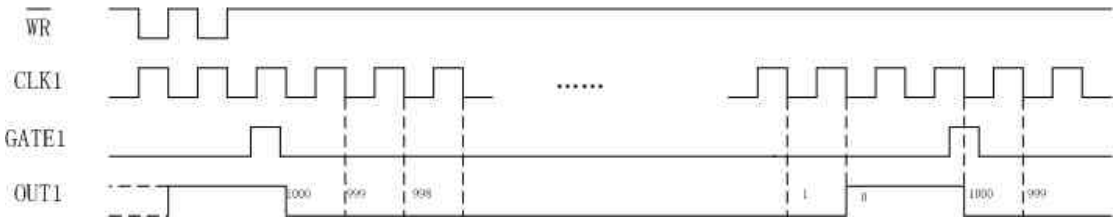
9 章习题

1. 下列地址哪些能够分配给 8253/8254 的计数器 0？为什么？（23H、54H、97H、51H、FCH、59H）

   解：因为已经约定采用 A2,A1 作为 8253 的内部地址线，而且计数器 0 的地址为 00，所以在题中所给的地址中只有 51H,59H 的 A2 和 A1 同时为 0，即：A2A1=00.

2. 如果计数器 0 设定为方式 0，GATE0＝1，CLK0＝1MHz，时常数为 N＝1000，请画出 OUT0 的波形。如果计数器 1 设定为方式 1，其它参数与计数器 0 相同，画出 OUT1 的波形。





3. 编程实现：将 8253 计数器 0 设置成方式 4，并置时常数 10000，然后处于等待状态，直到 CE 的内容≤1000 后再向下执行。

   解：

```
        MOV DX,COUNTD       ;写入计数器 0 的方式控制字
        MOV AL,00111000B
        OUT DX,AL
        MOV DX,COUNTA       ;设置计数器 0 的常数
        MOV AX,10000
        OUT DX,AL
        XCHG AL,AH
        OUT DX,AL
L1:     MOV DX,COUNTD       ;写入计数器 0 的方式控制字
```

```
MOV AL,0H
OUT DX,AL
MOV DX,COUNTA   ；读入 CE
IN AL,DX
MOV AH,AL
IN AL,DX
XCHG AL,AH
CMP AX,1000     ；判别 CE 当前大小
JA L1
```

4. 利用 8253 可以实现确定时间的延迟，编程实现延时 10 秒的程序段（设可以使用的基准时钟为 1MHz）。

解：本题使用计数器 0 和计数器 1，并且计数器 0 的输出 OUT0 作为计数器 1 的时钟输入 CLK1.

程序如下：

```
MOV   DX,COUNTD     ;写计数器 0 方式控制字
MOV   AL,00110100B
OUT   DX,AL
MOV   DX,COUNTA
MOV   AX,10000      ;写计数器 0 时常数，分频得到 100Hz 时钟频率
OUT   DX,AL
XCHG  AL,AH
OUT   DX,AL
MOV   DX,COUNTD     ；写计数器 1 方式控制字
MOV   AL,01110000B
OUT   DX,AL
MOV   DX，COUNTB
MOV   AX,999        ；分频得到 0.1Hz 时钟频率。（在方式 0 下，时常数为 N 时，
                    ；OUT 输出的低电平宽度为 N+1).
OUT   DX,AL
```

```
XCHG    AL,AH
OUT     DX,AL
L1:                         ；延时
MOV     DX,COUNTD       ；当前 CE 的内容锁存到 OL
MOV     AL,01000000B
OUT     DX,AL
MOV     DX,COUNTB
IN      AL,DX
MOV     AH,AL
IN      AL,DX
XCHG    AL,AH
CMP     AX,999
JNA     L1                  ；延时结束，则继续执行，否则，跳到 L1,继续延时
….
```

5. 比较 8254 方式 0 与方式 4、方式 1 与方式 5 的区别？

   方式 0 与方式 4

   方式 0 OUT 端计数过程中为低,计数值减为 0 时,输出变高

   方式 4 OUT 端计数过程中为高，计数值减为 0 时输出宽度为 1 个 CLK 的负脉冲

   方式 1 与方式 5

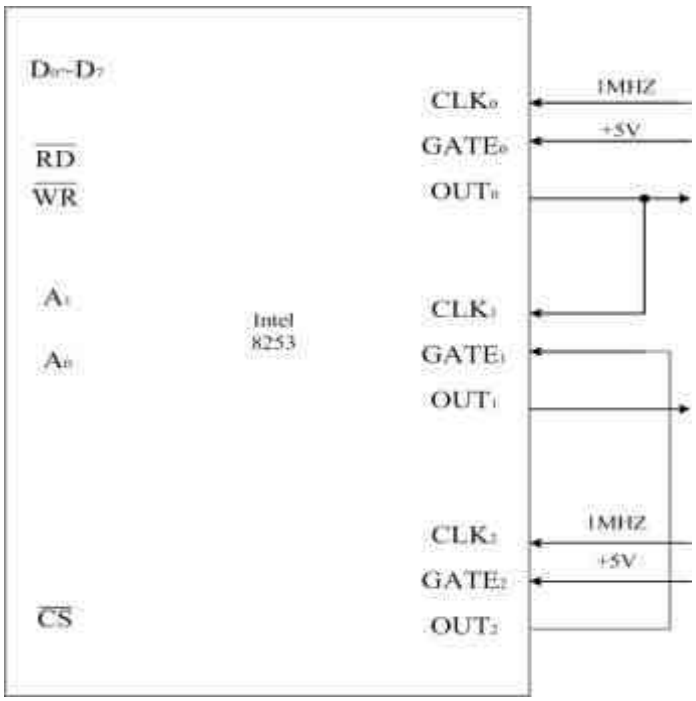
   方式 1 OUT 端输出宽度为 n 个 CLK 的低电平，计数值减为 0 时，输出为高

   方式 5 OUT 端计数过程中为高，计数值减为 0 时输出宽度为 1 个 CLK 的负脉冲

6. 在 8088 最小系统中，8253 的端口地址为 284H～287H。系统提供的时钟为 1MHz，要求在 OUT0 输出周期为 20 微秒的方波，在 OUT1 输出周期为 200 微秒，其中每周期为负的时间是 180 微秒的信号。请编写 8253 的初始化程序。

   解：

   OUT0 输出为 20 微妙方波，可用方式三直接产生，OUT1 输出波形与书中例 9.2 类似，可用其中思想产生此信号。如果允许增加外部器件，也可在例 9.1 的基础上，将 OUT 端信号通过与非门取反，这样即可产生题目要求信号。本例利用例 9.1 思想解答




```
MOV DX,287H             ;写计数器 0 控制方式字
MOV AL,00010110B
OUT DX,AL
MOV DX,284H             ；写计数器 0 时常数
MOV AL,20
OUR DX,AL
MOV DX,287              ；写计数器 2 控制方式字
MOV AL,10010110B
OUT DX,AL
MOV DX,286H             ；写计数器 2 时常数
MOV AL,200
OUT DX,AL
MOV DX,287H
MOV AL,01010010B    ；写计数器 1 控制方式字
OUT DX,AL
MOV DX,285H
MOV AL,9                ；写计数器 1 时常数
OUT DX,AL
```

7. 通过 8253 计数器 0 的方式 0 产生中断请求信号，现需要延迟产生中断的时刻，可采用：

A）在 OUT0 变高之前重置初值；

B）在 OUT0 变高之前在 GATE0 端加一负脉冲信号；

C）降低加在 CLK0 端的信号频率；

D）以上全是。

解：D

A:方式 0 下，在 OUT0 变高之前重置初值，将在下一个 CLK 的下降沿使时常数从 CR 读入 CE 并重新计数。

B:在 OUT0 变高之前在 GATE0 端加一负脉冲信号可以延时一个时钟周期，达到延时的目的。

C:降低加在 CLK0 端的信号频率，可以增大时钟周期，达到延长 OUT0 端低电平的时间。（注：A 中，如果重置的初值为 1，则不会达到延时的效果）

8. 已知 8254 计数器 0 的端口地址为 40H，控制字寄存器的端口地址为 43H，计数时钟频率为 2MHz，利用这一通道设计当计数到 0 时发出中断请求信号，其程序段如下，则中断请求信号的周期是 32.7675 ms。

```
MOV  AL,00110010B
OUT  43H,  AL
MOV  AL,  0FFH
OUT  40H,  AL
OUT  40H,  AL
```

9. 若 8254 芯片可使用的 8086 端口地址为 D0D0H～D0DFH，试画出系统设计连接图。设加到 8254 上的时钟信号为 2MHz，

（1）利用计数器 0～2 分别产生下列三种信号：

① 周期为 10us 的对称方波

② 每 1s 产生一个负脉冲

③ 10s 后产生一个负脉冲

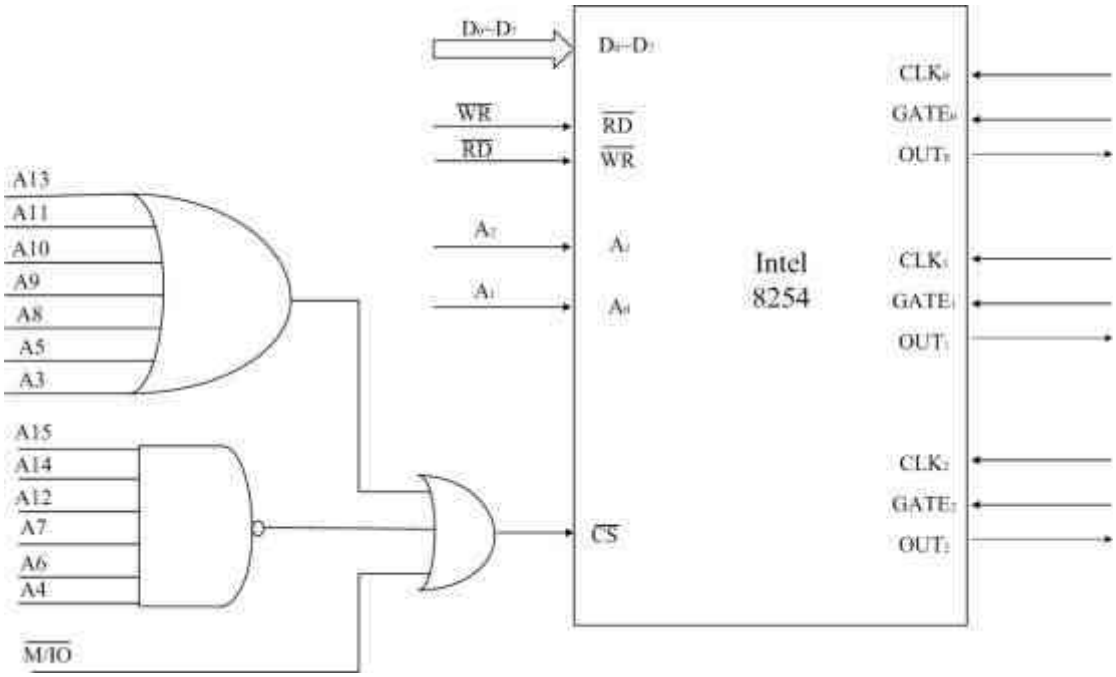
每种情况下，说明 8254 如何连接并编写包括初始化在内的程序段。

（2） 希望利用 8086 通过一专用接口控制 8253 的 GATE 端，当 CPU 使 GATE 有效开始，

20us 后在计数器 0 的 OUT 端产生一个正脉冲，试设计完成此要求的硬件和软件。

解：

(1) 选用 D0D0H~D0DFH 中的偶地址 D0D0,D0D2,D0D4,D0D6 为基本地址作为 8254 的端口地址，设 8086 工作在最小方式下。8254 端口地址译码电路如下图：



① 计数器 0 输入端加 2MHz 的时钟信号，$GATE_0$ 加+5V 电压，输出 OUT_0 信号为周期为 10μs 的对称方波。

初始化代码：

```
MOV DX,0D0D6H          ;写计数器 0 工作方式
MOV AL,00010110B
OUT DX,AL
MOV DX,0D0D0H          ；写计数器 0 时常数
MOV AL,20
OUT DX,AL
```

②CLK_0 加 2MHz 的始终信号，$GATE_0$,$GATE_1$ 加+5V 电压，OUT_0 输出加到 CLK1 做时钟信号，OUT1 输出为每 1s 产生一个负脉冲。

初始代码：

```
MOV DX,0D0D6H          ；写计数器 0 的工作方式
MOV AL,00010110B
```

```
OUT DX,AL
MOV DX,0D0D0H          ；写计数器 0 的时常数
MOV AL,100
OUT DX,AL
MOV DX,0D0D6H          ；写计数器 1 的工作方式
MOV AL,01110100B
OUT DX,AL
MOV DX,0D0D2H          ；写计数器 1 的时常数
MOV AX,20000
OUT DX,AL
XCHG AL,AH
OUT DX,AL
```

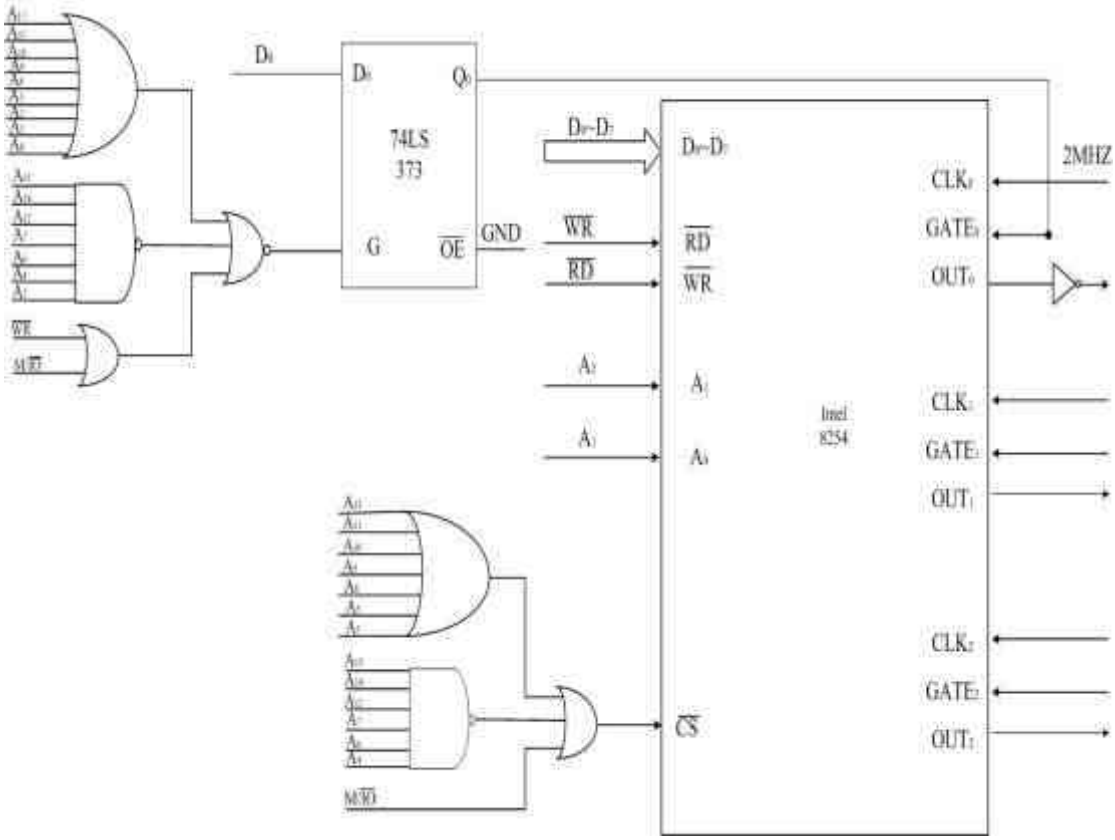
③CLK_0 加 2MHz 的始终信号，$GATE_0$,$GATE_1$ 加+5V 电压，OUT_0 输出加到 CLK2 做时钟信号，OUT2 输出为 10s 后产生一个负脉冲。

初始代码：

```
MOV DX,0D0D6H          ；写计数器 0 的工作方式
MOV AL,00110110B
OUT DX,AL
MOV DX,0D0D0H          ；写计数器 0 的时常数
MOV AX,1000
OUT DX,AL
XCHG AL,AH
OUT DX,AL
MOV DX,0D0D6H          ；写计数器 2 的工作方式
MOV AL,10111000B
OUT DX,AL
MOV DX,0D0D4H          ；写计数器 2 的时常数
MOV AX,20000
OUT DX,AL
XCHG AL,AH
```

OUT DX,AL

1) 选用地址 D0D0,DOD2,DOD4,DOD6 为 8253 的端口地址，D0D8 为 GATE 端口地址，该端口采用 74LS373，8253 用方式 4，在 OUT 输出端加非门实现脉冲功能。接口电路如图：



初始代码为：

```
MOV DX,0D0D8H          ;GATE 初始化
MOV AL,0
OUT DX,AL
MOV DX,0D0D6H          ;写计数器 0 工作方式
MOV AL,00011000B
OUT DX,AL
MOV DX,0D0D0H          ;写计数器 0 时常数
MOV AL,40
OUT DX,AL
MOV DX,0D0D8H
MOV AL,1
OUT DX,AL              ;使 GATE 变高有效
```

10. 若加到 8254 上的时钟频率为 0.5MHz，则一个计数器的最长定时时间是多少?若要求 10 分钟产生一次定时中断，试提出解决方案。

解：一个计数器的最长定时时间应该是置入时常数 0 时，此时定时时间为：

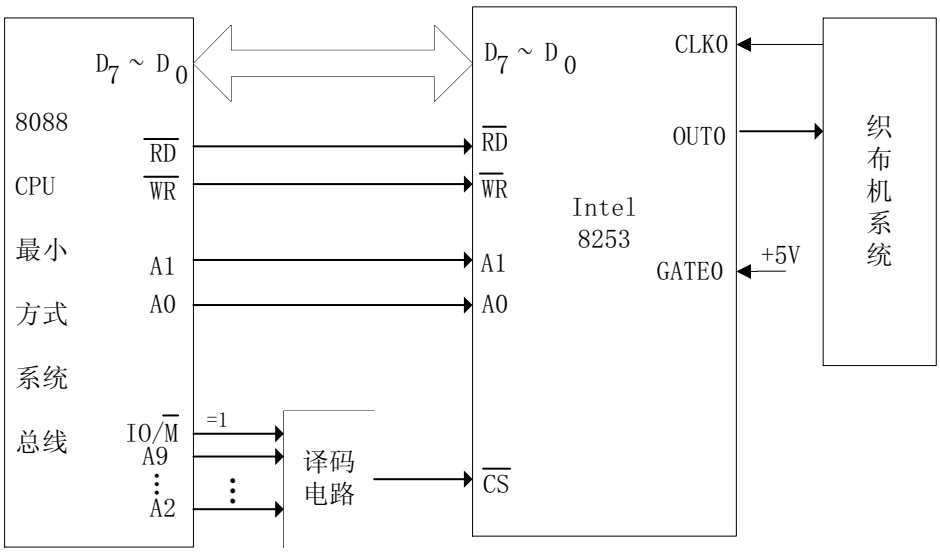
65536/0.5*10^6s=131ms

采用方式 0 即：计数达到终值时中断来 10 分钟产生一次定时中断，此时时常数 CR 为：10*60*0.5*10^6=3*10^9.

由于一个计数器最多分频 65536，所以至少得使用 2 个计数器。我们采用计数器 0 和计数器 1.计数器 0 的时常数 CR0 为 60000，计数器 1 的时常数 CR1 为 50000.

连接方式为：把 0.5MHz 的时钟频率接到计数器 0 的 CLK0，然后把计数器 0 的 OUT0 接到计数器 1 的 CLK1。这样计数器 1 的 OUT1 端输出的就是 10 分钟产生一次的定时中断。

11. 织布机控制系统如图 9.26 所示，已知织布机每织 1 米发出一个正脉冲，每织 100 米要求接收到一脉冲，去触发剪裁设备把布剪开。（1）设 8253 的端口地址为 80H～83H，编写对 8253 初始化程序。（2）假定系统提供的信号频率为 1MHz，希望利用 8253 的其余通道产生 0.1 秒的周期信号，编写初始化程序。



解：（1）

```
MOV     DX,83H
MOV     AL,00010100B
```

```
OUT     DX,AL
MOV     DX,80H
MOV     AL,100
OUT     DX,AL
```

（2）

将计数器 1 的输出 OUT1 信号作为计数器 2 的时钟输入 CLK2，计数器 1 的时钟输入为系统提供 1MHZ 的信号

```
MOV     DX,83H
MOV     AL,01110100B
OUT     DX,AL
MOV     DX,81H
MOV     AX,1000
OUT     DX,AL
XCHG    AL,AH
MOV     DX,AL
MOV     DX,83H
MOV     AL,10010110B
OUT     DX,AL
MOV     DX,82H
MOV     AL,100
OUT     DX,AL
```

图 9.26　织布机控制系统

12. 在 IBM PC 系统中根据下列不同条件设计接口逻辑，利用 8253 完成对外部脉冲信号重复频率的测量。

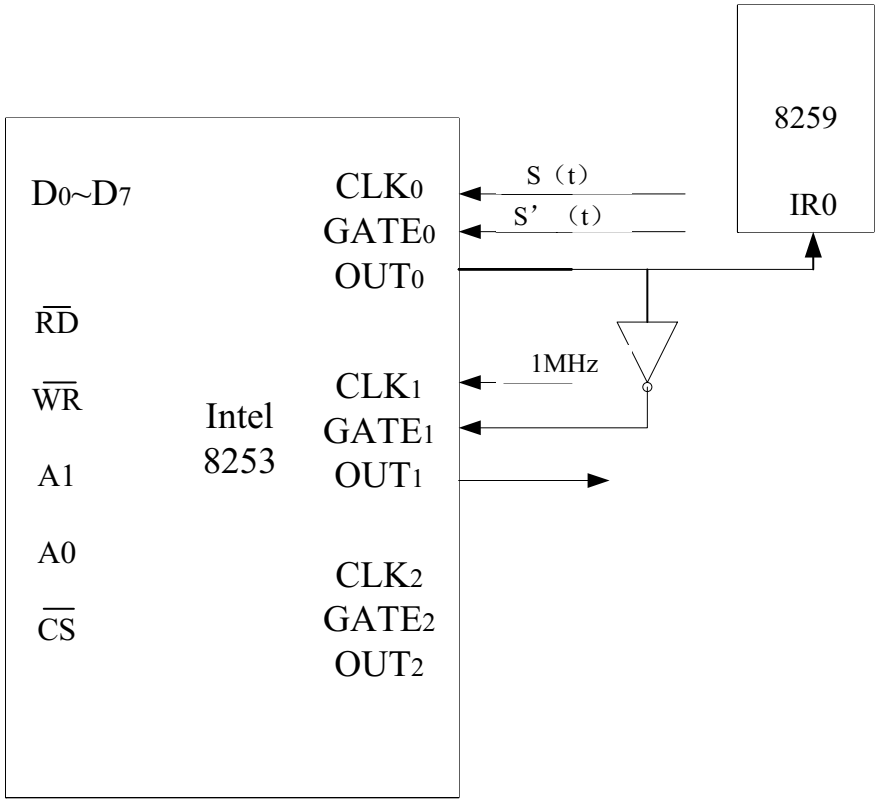
（1） 被测脉冲信号的重复频率在 10～1000Hz 范围内。

（2） 被测脉冲信号的重复频率在 0.5～1.5Hz 范围内。

（3） 被测脉冲信号重复频率在 10～100Hz 范围内。

（4） 被测是间歇脉冲信号,每次有信号时有 100 个脉冲,重复频率为 0.8～1.2MHz，间歇频率大约每秒 15 次,要求测有信号时的脉冲重复频率。

解：用两个计数器，计数器 0 的 CLK 接待测信号，GATE 接半周期为 10s 的高电平信号，OUT 接 8259，同时取反接计数器 1 的 GATE 端。计数器 1 的 CLK 接系统时钟，半周期为 T0。在这样的逻辑电路下，计数器 0 的功能是记录待测信号的脉冲数 N0，计数器 1 的功能是记录在相同时间里系统时钟信号的脉冲数 N1。根据 T=N1*T0/N0 可计算出待测信号的周期。S（t）是待测信号，S’（t）为给定的周期大于 10s 的高电平信号。



端口声明：COUNTA 为计数器 0 的地址，COUNTB 为计数器 2 的地址，COUNTD 为控制器地址，COUNT 为 373 地址

程序如下：

```
MOV DX,COUNTD          ；计数器 1 初始化
MOV AL,01110000B
OUT DX,AL
MOV DX,COUNTB
OUT DX,AL
MOV DX,COUNTB
MOV AL,0
OUT DX,AL
MOV DX,COUNTD          ；计数器 0 初始化
MOV AL,00010000B
OUT DX,AL
MOV DX,COUNTA
MOV AL,0
```

```
OUT DX,AL
OUT DX,AL
STI
```

读两计数器的计数，并进行计算的中断服务子程序：

```
PUSH AX
PUSH BX
PUSH CX
PUSH DX
MOV DX,COUNTD
MOV AL,00000000B
OUT DX,AL
MOV DX,COUNTA
IN AL,DX
XCHG AL,AH
IN AL,DX
XCHG AL,AH
NEG AX
INC AX
MOV BX,AX
MOV DX,COUNTD
MOV AL,00010000B
OUT DX,AL
MOV DX,COUNTB
IN AL,DX
XCHG AL,AH
IN AL,DX
XCHG AL,AH
NEG AX
INC AX
MOV CX,T0
```

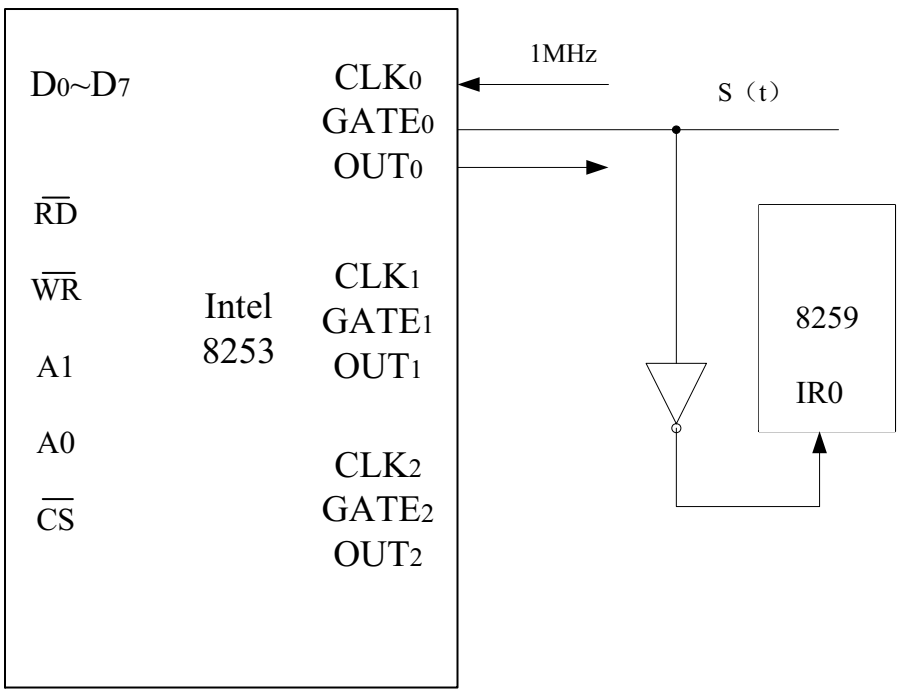
```
MUL CX
DIV BX
MOV SFR,AX
POP DX
POP CX
POP BX
POP AX
IRET
```

SFR 中保存结果即为待测信号的周期。

对于（1）题，10*10 不小于 100，10*1000 不大于 65535，可以用计数法。

同理（3）也可用此方法。

对于（2）题，可用周期法。逻辑电路图如下：



程序如下：

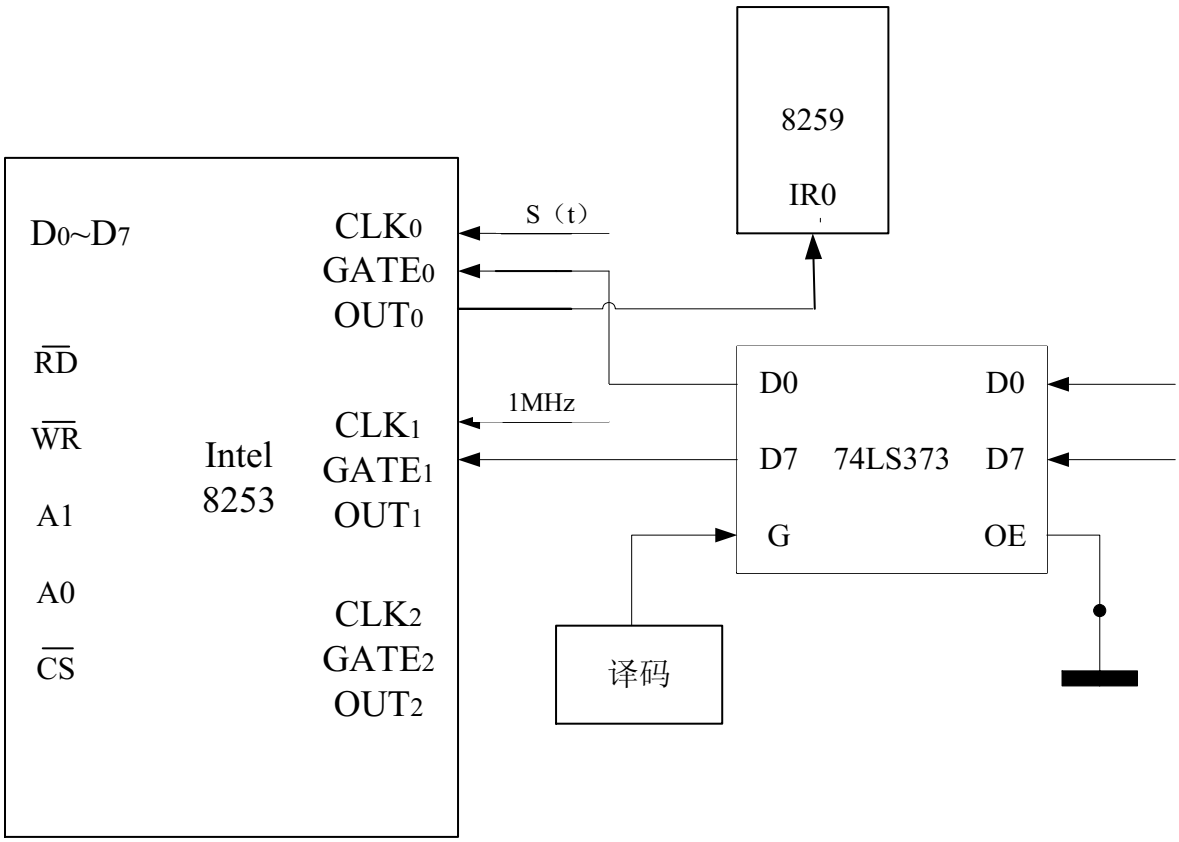
```
MOV DX,COUNTD
MOV AL,0011 0100B
OUT DX,AL
MOV DX,COUNTA
```

```
MOV AL,0
OUT DX,AL
OUT DX,AL
STI

PUSH AX
PUSH BX
PUSH DX
MOV DX,COUNTD
MOV AL,0000 0000B
OUT DX,AL
MOV DX,COUNTA
IN AL,DX
XCHG AL,AH
IN AL,DX
XCHG AL,AH
NEG AX
INC AX
MOV BX,AX
MOV DX,000FH
MOV AX,4240H
DIV BX
MOV SFR,AX
POP DX
POP BX
POP AX
IRET
```

(4) 如图设计接口，计数器 1 用来记录在 50 个脉冲所用时间，50 个信号脉冲最多用 1/0.8*50(约为 63us)由于计数器 1 用 1MHz CLK，故其计数个数 N 即为 N us，

所以当 N>63 时，则有低电平间隔计入，须重新计数。当 N>63 时，则计算得待测频率。



程序如下：

```
MOV DX,COUNTD
MOV AL,00H
OUT DX,AL
MOV DX,COUNTD
MOV AL,0001 0000B
OUT DX,AL
MOV DX,COUNTA
MOV AL,50
OUT DX,AL
MOV DX,COUNTD
MOV AL,0111 0000B
OUT DX,AL
MOV COUNTB
```

```
MOV AL,0
OUT DX,AL
OUT DX,AL
L2:  MOV DX,COUNT       ；给GATE0和GATE1高电平，开始计数
MOV AL,81H
OUT DX,AL
L1:  NOP
MOV DX,COUNTD
MOV AL,00000000B
OUT DX,AL
MOV DX,COUNTA
IN AL,DX
MOV DX,COUNTA
IN AL,DX
AND AL,AL               ；判断是否计完50个脉冲，若未计完继续等待
JNZ L1
MOV DX,COUNT
MOV AL,00H              ；若计完则暂停计数
OUT DX,AL
MOV DX,COUNTD           ；读计数器1结果
MOV AL,01000000B
OUT DX,AL
MOV DX,COUNTB
IN AL,DX
XCHG AL,AH
IN AL,DX
XCHG AL,AH
NEG AX
INC AX
CMP AX,70H          ；当AL大于70，则有间歇计入，重新测试
```

```
JA L2
MOV BL,AL
MOV AL,50          ；计算频率
DIV BL
MOV FREC,AL
```

10 章习题

1. 试分析 8255A 方式 0、方式 1 和方式 2 的主要区别，并分别说明它们适合于什么应用场合。

   答：方式 0 是基本的输入/输出，端口 A、B、C 都可以作为输入输出端口。适用于 CPU 与非智能 I/O 设备的数据传输；

   方式 1 是有联络信号的输入/输出，端口 A、B 都可以设定成该方式，此时三个端口的信号线分成了 A、B 两组，$PC_7 \sim PC_4$ 用作 A 组的联络信号，$PC_3 \sim PC_0$ 用作 B 组的联络信号。适用于高速 CPU 与低速 I/O 设备的数据传输；

   方式 2 是双向传输，只有 A 组端口可以设定成该方式，$PC_6 \sim PC_7$ 用作输出的联络信号，$PC_4 \sim PC_5$ 用作输入的联络信号，PC_3 用作中断请求信号。适用于双机之间数据的并行传送。

2. 8255A 的 A 组设置成方式 1 输入,与 CPU 之间采用中断方式联络,则产生中断请求信号 INTRA 的条件是 STBA=__1__,IBFA=__1__,INTEA=__1__。

3. 如果 8255A 的端口地址为 300H～303H，A 组和 B 组均为方式 0，端口 A 为输出，端口 B 为输入，PC3～PC0 为输入，PC7～PC4 为输出，写出 8255A 的初始化程序段；编程实现将从端口 C 低 4 位读入的值从高 4 位送出。

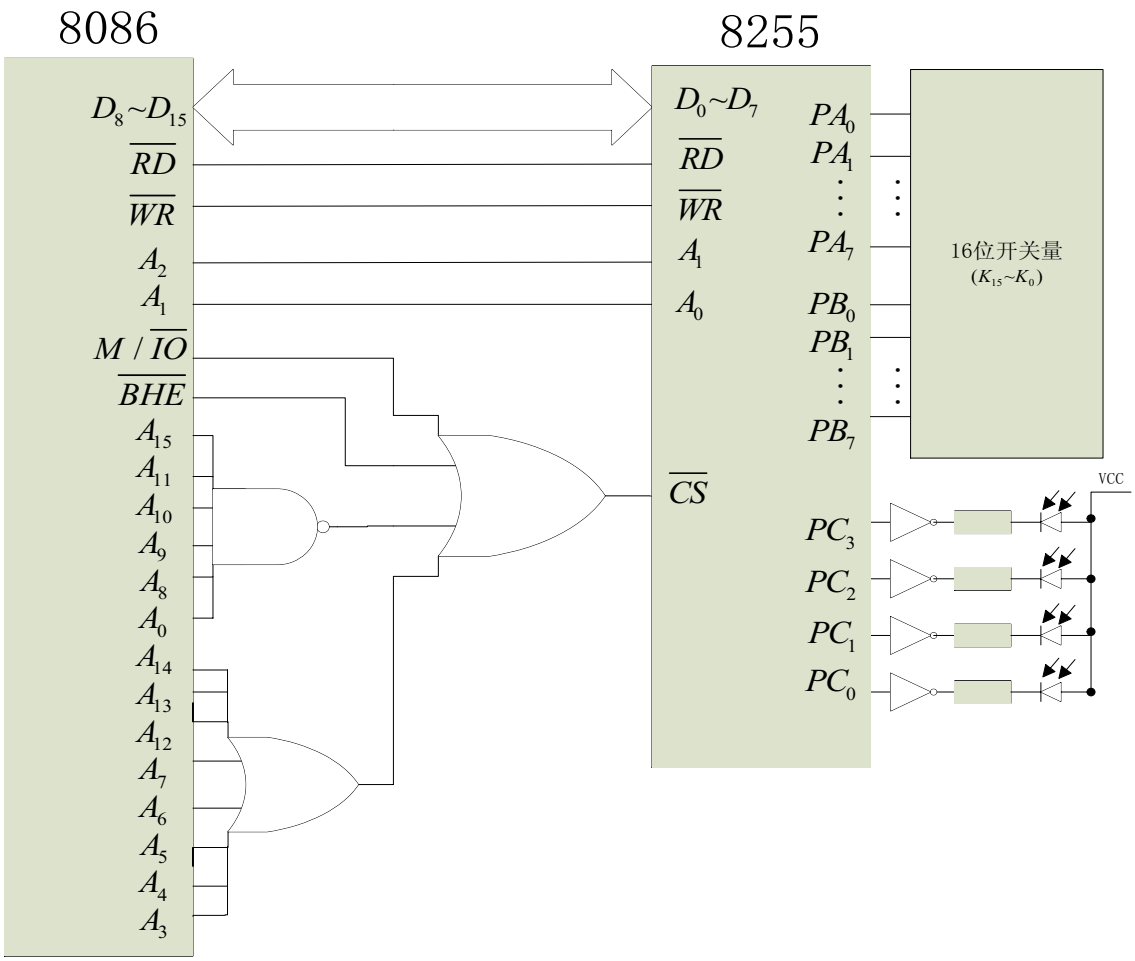
   解：
   ```
   MOV DX,303H
   MOV AL,10000011B
   OUT DX,AL
   MOV DX,302H
   IN AL,DX
   MOV CL,4
   ```

SHL AL,CL

OUT DX,AL

4. 在实际应用中经常需要检测设备的状态，并进行指示。在 8086 最小方式系统下，有一片 8255A，其分配的端口地址为 8F00H～8F07H 中的奇地址，外部设备产生的状态有 16 个（K15～K0），要求采用 4 个发光二极管来指示开关量中“1”的个数。（1）画出 8255A 的连接图；（2）编写程序段实现连续检测并显示。

解：（1）



(2) MOV DX,8F07H

MOV AL,10010010B ;端口 A、B 方式 0 输入，端口 C 方式 0 输出

OUT DX,AL

NEXT:

```
        MOV DX,8F03H                ;从端口 B 读取高 8 位开关量
        IN AL,DX
        XCHG AL,AH
        MOV DX,8F01H                ;从端口 A 读取低 8 位开关量
        IN AL,DX
        MOV BX,AX
        XOR AL,AL
        MOV CX,16
        CLC
    L2:
        SHL BX,1
        JNC L1
        INC AL
    L1:
        LOOP L2
        MOV DX,8F05H                ;从端口 C 送出
        OUT DX,AL
        JMP NEXT                    ;进行下一次检测
```

5. 利用 IBM PC 系统机的总线槽，开发由一片 8255A 构成的子系统，8255A 端口地址为 260H～263H，编程实现产生如图 10.20 所示的 8 个信号(各个信号的节拍不必严格相等)。

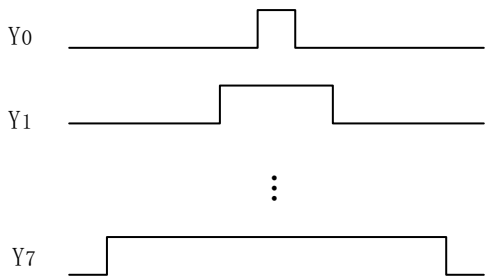
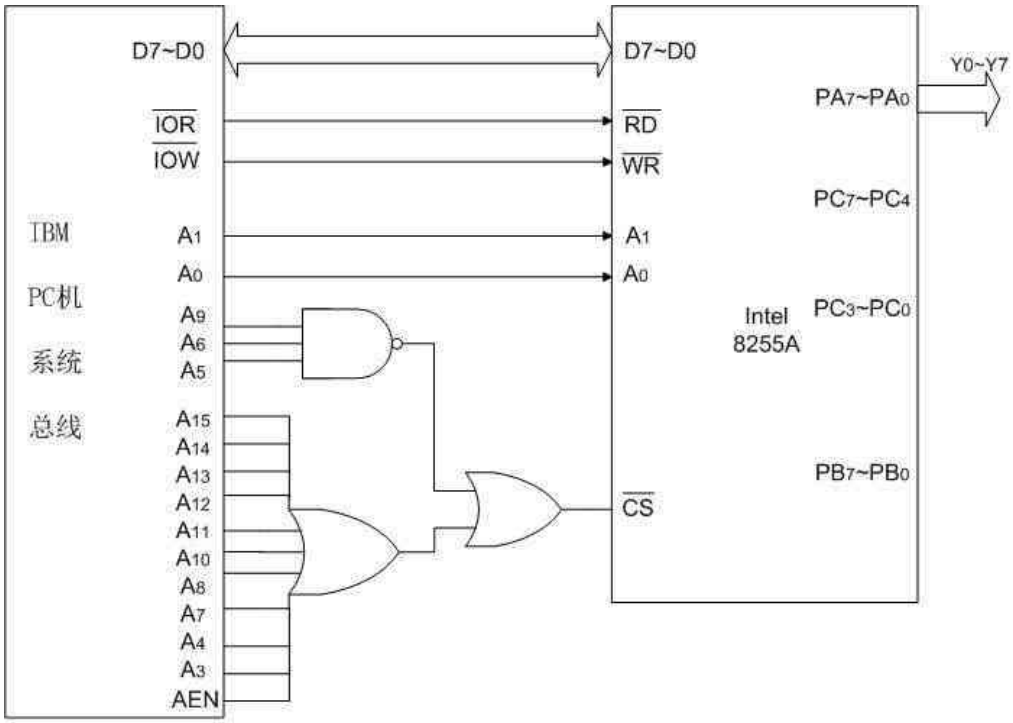


图 10.20　要求产生的信号波形

解：8355A 与 IBM PC 机总线的连接框图如下：



可将 8255A 的端口 A 作为要产生的信号的输出端口，设定为方式 0 输出，端口 B 和端口 C 不做使用，均设定为方式 0 输出。程序段如下：

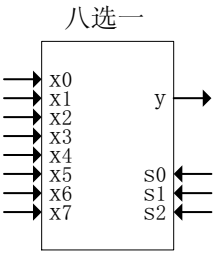
```
        MOV     DX,263H             ;设定 8255A 的工作方式
        MOV     AL,10000000B
        OUT     DX,AL
        MOV     DX,260H             ;产生指定信号
        XOR     AL,AL
        OUT     DX,AL
REP:
        MOV     AL,80H
        MOV     CX,7

REP1:
        OUT     DX,AL
        SAR     AL,1
        LOOP    REP1
        MOV     CX,8
REP2:
        SHL     AL,1
        OUT     DX,AL
        LOOP    REP2
        JMP     REP
```

6. 在实际应用中，经常会遇到要求输入多个数据量，这时需要用到多路开关，如图 10.21

表示八选一的逻辑框图及其真值表。

现有 8 组 16 位开关量数据（无符号数），要求通过一片 8255A（端口地址为 260H～263H）分时输入到 CPU（8088 最小方式系统）中，并找出它们中的最大值，并通过 4 个发光二极管指示其序号（灯亮表示“1”）。画出 8255A 的连接图，并编程实现。

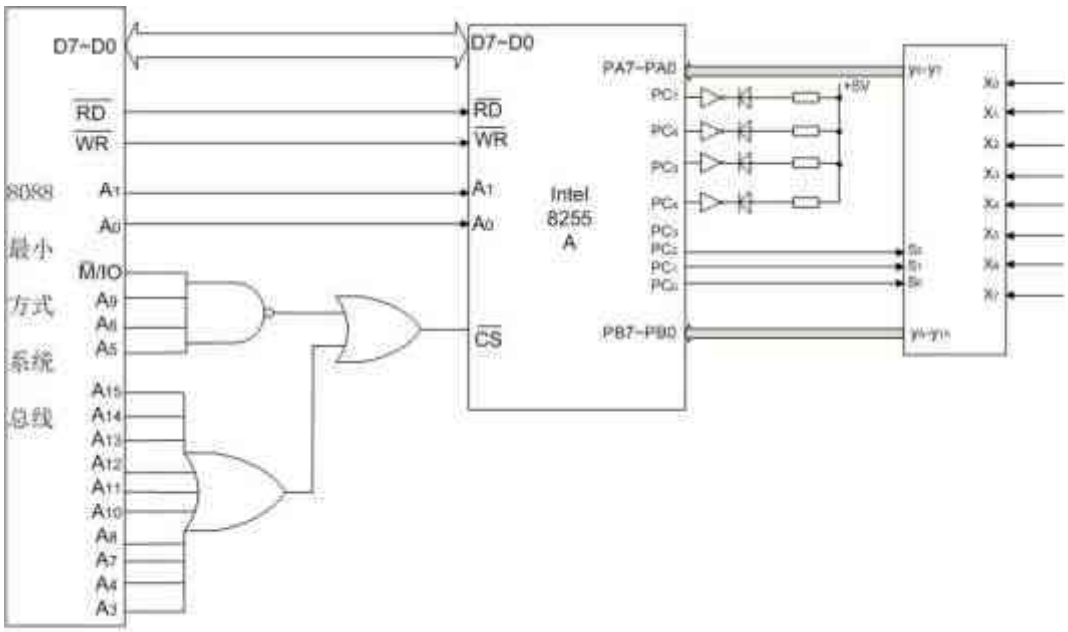


S2	S1	S0	y
0	0	0	x0
0	0	1	x1
0	1	0	x2
0	1	1	x3
1	0	0	x4
1	0	1	x5
1	1	0	x6
1	1	1	x7

图10.21　八选一逻辑电路

解：由于开关量是 16 为数据，故可以将 8255A 的 PA 端与 PB 端设定为方式 0，分别读取开关量的低八位和高八位，以 PC 低三位端口的控制八选一电路的输出依次从 X_0 到 X_7，使用 PC 端口的高八位输出最大开关量的序号（该序号为 1~8），控制发光二极管的亮灭以码指示序号。

连接图如下：



程序段如下：

```
        MOV   DX,263H
        MOV   AL,10011010B   ;设定工作方式，PA,PB 均工作于方式 0，PA、PB 为输入，PC 为输出
        OUT    DX,AL
        MOV    CX,8
        XOR    BX,BX
        MOV    SI,0                    ;SI 表示输入开关量的序号
ST1:
```

```
        MOV     DX,262H
        OUT     DX,SI
        MOV     DX,260H          ;将开关量数据的低八位写入 AL
        IN      AL,DX
        MOV     DX,261H          ;将开关量数据的高八位写入 AH
        IN      AH,DX
        CMP     BX,AX
        JA      NEXT
        MOV     BX,AX            ;将当前最大值保存在 BX 中
        INC     SI
        PUSH    SI               ;将当前最大值的序号压栈
NEXT:
        INC     SI
        LOOP    ST1
        POP     SI               ;最大值的序号出栈
        XOR     AX,AX
        MOV     AX,SI
        MOV     CL,4
        ROL     AL,CL            ;将最大值的序号（4 位）移至 AL 的高四位
        MOV     DX,262H
        OUT     DX,AL            ;PC 的高四位输出最大值序号
```