

附录:《IBM—PC 汇编语言程序设计》习题参考答案

[第一章](#) * [第二章](#) * [第三章](#) * [第四章](#) * [第五章](#) * [第六章](#) * [第七章](#) * [第八章](#) * [第九章](#) * [第十章](#) * [第十一章](#)

第一章. 习 题

1.1 用降幂法和除法将下列十进制数转换为二进制数和十六进制数:

(1) 369 (2) 10000 (3) 4095 (4) 32767

答: (1) 369=1 0111 0001B=171H
(2) 10000=10 0111 0001 0000B=2710H
(3) 4095=1111 1111 1111B=FFFH
(4) 32767=111 1111 1111 1111B=7FFFH

1.2 将下列二进制数转换为十六进制数和十进制数:

(1) 10 1101 (2) 1000 0000 (3) 1111 1111 1111 1111 (4) 1111 1111

答: (1) 10 1101B=2DH=45
(2) 1000 0000B=80H=128
(3) 1111 1111 1111 1111B=FFFFH=65535
(4) 1111 1111B=FFH=255

1.3 将下列十六进制数转换为二进制数和十进制数:

(1) FA (2) 5B (3) FFFE (4) 1234

答: (1) FAH=1111 1010B=250
(2) 5BH=101 1011B=91
(3) FFEH=1111 1111 1111 1110B=65534
(4) 1234H=1 0010 0011 0100B=4660

1.4 完成下列十六进制数的运算, 并转换为十进制数进行校核:

(1) 3A+B7 (2) 1234+AF (3) ABCD-FE (4) 7AB×6F

答: (1) 3A+B7H=F1H=241
(2) 1234+AFH=12E3H=4835
(3) ABCD-FEH=AACFH=43727
(4) 7AB×6FH=35325H=217893

1.5 下列各数均为十进制数, 请用 8 位二进制补码计算下列各题, 并用十六进制数表示其运算结果。

(1) (-85)+76 (2) 85+(-76) (3) 85-76 (4) 85-(-76) (5) (-85)-76 (6) -85-(-76)

答: (1) (-85)+76=1010 1011B+0100 1100B=1111 0111B=0F7H; CF=0; OF=0
(2) 85+(-76)=0101 0101B+1011 0100B=0000 1001B=09H; CF=1; OF=0
(3) 85-76=0101 0101B-0100 1100B=0101 0101B+1011 0100B=0000 1001B=09H; CF=0; OF=0
(4) 85-(-76)=0101 0101B-1011 0100B=0101 0101B+0100 1100B=10100001B=0A1H; CF=0; OF=1
(5) (-85)-76=1010 1011B-0100 1100B=1010 1011B+1011 0100B=0101 1111B=5FH; CF=0; OF=1
(6) -85-(-76)=1010 1011B-1011 0100B=1010 1011B+0100 1100B=11110111B=0F7H; CF=0; OF=0

1.6 下列各数为十六进制表示的 8 位二进制数, 请说明当它们分别被看作是用补码表示的带符号数或无符号数时, 它们所表示的十进制数是什么?

(1) D8 (2) FF

答: (1) D8H 表示的带符号数为 -40, D8H 表示的无符号数为 216;
(2) FFH 表示的带符号数为 -1, FFH 表示的无符号数为 255。

1.7 下列各数均为用十六进制表示的 8 位二进制数, 请说明当它们分别被看作是用补码表示的数或字符的 ASCII 码时, 它们所表示的十进制数及字符是什么?

(1) 4F (2) 2B (3) 73 (4) 59

答: (1) 4FH 表示的十进制数为 79, 4FH 表示的字符为 O;
(2) 2BH 表示的十进制数为 43, 2BH 表示的字符为 +;
(3) 73H 表示的十进制数为 115, 73H 表示的字符为 s;
(4) 59H 表示的十进制数为 89, 59H 表示的字符为 Y。

1.8 请写出下列字符串的 ASCII 码值。

For example,

This is a number 3692.

答: 46H 6FH 72H 20H 65H 78H 61H 6DH 70H 6CH 65H 2CH 0AH 0DH
 54H 68H 69H 73H 20H 69H 73H 20H 61H 20H 6EH 75H 6DH 62H 65H
 72H 20H 33H 36H 39H 32H 2EH 0AH 0DH

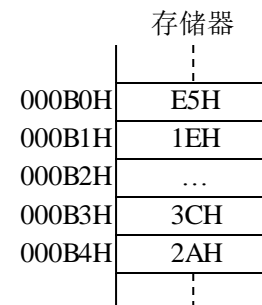
第二章. 习 题

- 2.1 在 80x86 微机的输入/输出指令中, I/O 端口号通常是由 DX 寄存器提供的, 但有时也可以在指令中直接指定 00~FFH 的端口号。试问可直接由指令指定的 I/O 端口数。

答: 可直接由指令指定的 I/O 端口数为 256 个。

- 2.2 有两个 16 位字 1EE5H 和 2A3CH 分别存放在 80x86 微机的存储器的 000B0H 和 000B3H 单元中, 请用图表示出它们在存储器里的存放情况。

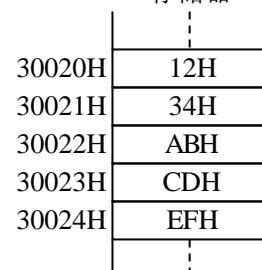
答: 存储器里的存放情况如右上图所示。



2.2 题的信息存放情况

- 2.3 在 IBM PC 机的存储器中存放信息如右下图所示。试读出 30022H 和 30024H 字节单元的内容, 以及 30021H 和 30022H 字单元的内容。

答: 30022H 字节单元的内容为 ABH; 30024H 字节单元的内容为 EFH。
 30021H 字单元的内容为 AB34H; 30022H 字单元的内容为 CDABH。



2.3 题的信息存放情况

- 2.4 在实模式下, 段地址和偏移地址为 3017:000A 的存储单元的物理地址是什么? 如果段地址和偏移地址是 3015:002A 和 3010:007A 呢?

答: 3017:000A、3015:002A 和 3010:007A 的存储单元的物理地址都是 3017AH。

- 2.5 如果在一个程序开始执行以前(CS)=0A7F0H, (如 16 进制数的最高位为字母, 则应在其前加一个 0) (IP)=2B40H, 试问该程序的第一个字的物理地址是多少?

答: 该程序的第一个字的物理地址是 0AAA40H。

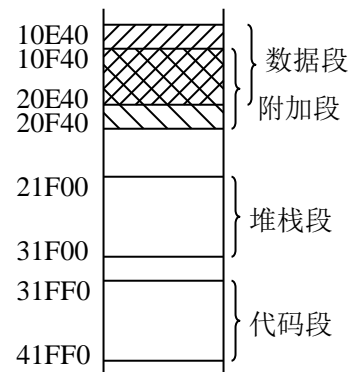
- 2.6 在实模式下, 存储器中每一段最多可有 10000H 个字节。如果用调试程序 DEBUG 的 r 命令在终端上显示出当前各寄存器的内容如下, 请画出此时存储器分段的示意图, 以及条件标志 OF、SF、ZF、CF 的值。

C>debug

-r

AX=0000 BX=0000 CX=0079 DX=0000 SP=FFEE BP=0000
 SI=0000 DI=0000 DS=10E4 ES=10F4 SS=21F0 CS=31FF
 IP=0100 NV UP DI PL NZ NA PO NC

答: 此时存储器分段的示意图如右图所示。OF、SF、ZF、CF 的值都为 0。



2.6 题的存储器分段示意图

- 2.7 下列操作可使用那些寄存器?

- | | |
|-----------------|------------------------|
| (1) 加法和减法 | 数据寄存器等 |
| (2) 循环计数 | CX |
| (3) 乘法和除法 | AX、DX, 乘数和除数用其他寄存器或存储器 |
| (4) 保存段地址 | 段寄存器 |
| (5) 表示运算结果为 0 | ZF=1 |
| (6) 将要执行的指令地址 | CS:IP |
| (7) 将从堆栈取出数据的地址 | SS:SP |

答: 答案见题目的右边。

- 2.8 那些寄存器可以用来指示存储器地址?

答: BX、BP、SI、DI、堆栈操作时的 SP、对应的段地址、386 及其后继机型的 Exx。

- 2.9 请将下列左边的项和右边的解释联系起来(把所选字母放在括号中):

- | | | |
|---------|-----|---------------------------------------|
| (1) CPU | (M) | A. 保存当前栈顶地址的寄存器。 |
| (2) 存储器 | (C) | B. 指示下一条要执行的指令的地址。 |
| (3) 堆栈 | (D) | C. 存储程序、数据等信息的记忆装置, 微机有 RAM 和 ROM 两种。 |
| (4) IP | (B) | D. 以后进先出方式工作的存储空间。 |

- | | | |
|-----------|-----|-------------------------------------------------|
| (5) SP | (A) | E.把汇编语言程序翻译成机器语言程序的系统程序。 |
| (6) 状态标志 | (L) | F.唯一代表存储空间中每个字节单元的地址。 |
| (7) 控制标志 | (K) | G.能被计算机直接识别的语言。 |
| (8) 段寄存器 | (J) | H.用指令的助记符、符号地址、标号等符号书写程序的语言。 |
| (9) 物理地址 | (F) | I.把若干个模块连接起来成为可执行文件的系统程序。 |
| (10) 汇编语言 | (H) | J.保存各逻辑段的起始地址的寄存器, 8086/8088 机有四个: CS、DS、SS、ES。 |
| (11) 机器语言 | (G) | K.控制操作的标志, 如 DF 位。 |
| (12) 汇编程序 | (E) | L.记录指令操作结果的标志, 共 6 位: OF、SF、ZF、AF、PF、CF。 |
| (13) 连接程序 | (I) | M.分析、控制并执行指令的部件, 由算术逻辑部件 ALU 和寄存器等组成。 |
| (14) 指令 | (O) | N.由汇编程序在汇编过程中执行的指令。 |
| (15) 伪指令 | (N) | O.告诉 CPU 要执行的操作(一般还要指出操作数地址), 在程序运行时执行。 |

答: 答案见题目的括号中。

第三章. 习 题

3.1 给定(BX)=637DH, (SI)=2A9BH, 位移量 D=7237H, 试确定在以下各种寻址方式下的有效地址是什么?

- (1) 立即寻址
- (2) 直接寻址
- (3) 使用 BX 的寄存器寻址
- (4) 使用 BX 的简接寻址
- (5) 使用 BX 的寄存器相对寻址
- (6) 基址变址寻址
- (7) 相对基址变址寻址

答: (1) 操作数在指令中, 即立即数;

(2) EA=D=7237H;

(3) 无 EA, 操作数为(BX)=637DH;

(4) EA=(BX)=637DH;

(5) EA=(BX)+D=0D5B4H;

(6) EA=(BX)+(SI)=8E18H;

(7) EA=(BX)+(SI)+D=1004FH; 超过了段的边界, 最高进位位丢失, 因此 EA=004FH。

3.2 试根据以下要求写出相应的汇编语言指令

- (1) 把 BX 寄存器和 DX 寄存器的内容相加, 结果存入 DX 寄存器中。
- (2) 用寄存器 BX 和 SI 的基址变址寻址方式把存储器中的一个字节与 AL 寄存器的内容相加, 并把结果送到 AL 寄存器中。
- (3) 用寄存器 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)相加, 并把结果送回存储器中。
- (4) 用位移量为 0524H 的直接寻址方式把存储器中的一个字与数 2A59H 相加, 并把结果送回存储单元中。
- (5) 把数 0B5H 与(AL)相加, 并把结果送回 AL 中。

答: (1) ADD DX, BX

(2) ADD AL, [BX][SI]

(3) ADD [BX+0B2H], CX

(4) ADD WORD PTR [0524H], 2A59H

(5) ADD AL, 0B5H

3.3 写出把首地址为 BLOCK 的数组的第 6 个字送到 DX 寄存器的指令。要求使用以下几种寻址方式:

- (1) 寄存器间接寻址
- (2) 寄存器相对寻址

(3) 基址变址寻址

答: (1) MOV BX, OFFSET BLOCK
ADD BX, (6 - 1)*2

MOV DX, [BX]

(2) MOV BX, OFFSET BLOCK
MOV DX, [BX+(6 - 1)*2]

改为: MOV BX, (6-1)*2

也可 MOV DX, BLOCK[BX]

(3) MOV BX, OFFSET BLOCK
MOV SI, (6 - 1)*2
MOV DX, [BX][SI]

3.4 现有(DS)=2000H, (BX)=0100H, (SI)=0002H, (20100H)=12H, (20101H)=34H, (20102H)=56H, (20103H)=78H, (21200H)=2AH, (21201H)=4CH, (21202H)=B7H, (21203H)=65H, 试说明下列各条指令执行完后 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]

答: (1) (AX)=1200H

(2) (AX)=0100H

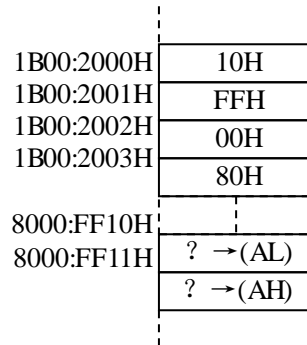
(3) (AX)=4C2AH

(4) (AX)=3412H

(5) (AX)=4C2AH

(6) (AX)=7856H

(7) (AX)=65B7H



3.6 题的作图表示

3.5 给定(IP)=2BC0H, (CS)=0200H, 位移量 D=5119H, (BX)=1200H, (DS)=212AH, (224A0H)=0600H, (275B9H)=098AH, 试为以下的转移指令找出转移的偏移地址。

(1) 段内直接寻址

(2) 使用 BX 及寄存器间接寻址方式的段内间接寻址

(3) 使用 BX 及寄存器相对寻址方式的段内间接寻址

答: (1) JMP NEAR PTR 5119H ; (IP)=5119H+((IP)+03H)=7CDCH, 物理地址 PA=09CDCH
(IP)+03H 是 JMP NEAR PTR 5119H 指令的下一条指令的首地址。

(2) JMP WORD PTR [BX] ; (IP)=((DS)*10H+(BX))=0600H, PA=02600H

(3) JMP D[BX] ; (IP)=((DS)*10H+(BX)+D)=098AH, PA=0298AH

3.6 设当前数据段寄存器的内容为 1B00H, 在数据段的偏移地址 2000H 单元内, 含有一个内容为 0FF10H 和 8000H 的指针, 它们是一个 16 位变量的偏移地址和段地址, 试写出把该变量装入 AX 的指令序列, 并画图表示出来。

答: MOV BX, [2000H] ; 图示如上所示。

MOV AX, [2000H+2]

MOV BX, 2000H

MOV ES, AX

LES BX, [BX]

MOV AX, ES:[BX]

MOV AX, ES:[BX]

3.7 在 0624H 单元内有一条二字节 JMP SHORT OBJ 指令, 如其中位移量为(1) 27H, (2) 6BH, (3) 0C6H, 试问转向地址 OBJ 的值是多少?

答: (1) OBJ=0624H+02H+27H=064DH

(2) OBJ=0624H+02H+6BH=0691H

(3) OBJ=0624H+02H+0C6H=05ECH ; C6H 对应的负数为-3AH (向上转移, 负位移量)

3.8 假定(DS)=2000H, (ES)=2100H, (SS)=1500H, (SI)=00A0H, (BX)=0100H, (BP)=0010H, 数据段中变量名 VAL 的偏移地址为 0050H, 试指出下列源操作数字段的寻址方式是什么? 其物理地址值是多少?

(1) MOV AX, 0ABH

(2) MOV AX, BX

(3) MOV AX, [100H]

(4) MOV AX, VAL

(5) MOV AX, [BX]

(6) MOV AX, ES:[BX]

- (7) MOV AX, [BP] (8) MOV AX, [SI]
 (9) MOV AX, [BX+10] (10) MOV AX, VAL[BX]
 (11) MOV AX, [BX][SI] (12) MOV AX, VAL[BX][SI]
- 答: (1) 立即方式; 操作数在本条指令中
 (2) 寄存器寻址方式; 操作数为 (BX)=0100H
 (3) 直接寻址方式; PA=20100H
 (4) 直接寻址方式; PA=20050H
 (5) BX 寄存器间接寻址方式; PA=20100H
 (6) 附加段 BX 寄存器间接寻址方式; PA=21100H
 (7) BP 寄存器间接寻址方式; PA=15010H
 (8) SI 寄存器间接寻址方式; PA=200A0H
 (9) BX 寄存器相对寻址方式; PA=20110H
 (10) BX 寄存器相对寻址方式; PA=20150H
 (11) BX 和 SI 寄存器基址变址寻址方式; PA=201A0H
 (12) BX 和 SI 寄存器相对基址变址寻址方式; PA=201F0H

3.9 在 ARRAY 数组中依次存储了七个字数据, 紧接着是名为 ZERO 的字单元, 表示如下:

ARRAY DW 23, 36, 2, 100, 32000, 54, 0

ZERO DW ?

- (1) 如果 BX 包含数组 ARRAY 的初始地址, 请编写指令将数据 0 传送给 ZERO 单元。
 (2) 如果 BX 包含数据 0 在数组中的位移量, 请编写指令将数据 0 传送给 ZERO 单元。

答: (1) MOV AX, [BX+(7-1)*2]
 MOV [BX+(7)*2], AX
 (2) MOV AX, ARRAY [BX]
 MOV ARRAY [BX+2], AX

3.10 如 TABLE 为数据段中 0032 单元的符号名, 其中存放的内容为 1234H, 试问以下两条指令有什么区别? 指令执行完后 AX 寄存器的内容是什么?

MOV AX, TABLE

LEA AX, TABLE

答: MOV AX, TABLE 是将 TABLE 单元的内容送到 AX, (AX)=1234H
 LEA AX, TABLE 是将 TABLE 单元的有效地址送到 AX, (AX)=0032H

TABLE	0AH
	00H
	14H
TABLE+3	00H
	1EH
	00H
	28H
	00H
	32H
	00H

3.11 执行下列指令后 AX 寄存器中的内容是什么?

TABLE DW 10, 20, 30, 40, 50 ; 000AH, 0014H, 001EH, 0028H, 0032H

ENTRY DW 3

⋮

MOV BX, OFFSET TABLE

ADD BX, ENTRY

MOV AX, [BX]

答: (AX)=1E00H (TABLE 的存储方式如右图所示)

3.11 题的 TABLE 存储方式

3.12 下列 ASCII 码串(包括空格符)依次存储在起始地址为 CSTRING 的字节单元中:

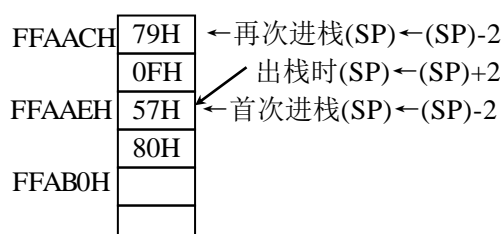
CSTRING DB 'BASED ADDRESSING'

请编写指令将字符串中的第 1 个和第 7 个字符传送给 DX 寄存器。

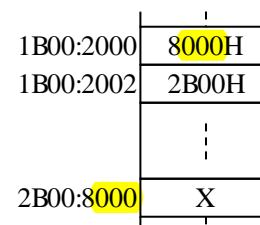
答: MOV DH, CSTRING
 MOV DL, CSTRING+7-1

3.13 已知堆栈段寄存器 SS 的内容是 0FFA0H, 堆栈指针寄存器 SP 的内容是 00B0H, 先执行两条把 8057H 和 0F79H 分别进栈的 PUSH 指令, 再执行一条 POP 指令。试画出堆栈区和 SP 的内容变化过程示意图(标出存储单元的物理地址)。

答: 堆栈区和 SP 的内容变化过程示意图如下左图所示。



3.13 题的堆栈区和 SP 的内容变化过程示意图



3.14 题的存储区情况

3.14 设(DS)=1B00H, (ES)=2B00H, 有关存储单元的内容如上右图所示。请写出两条指令把字变量 X 装入 AX 寄存器。

答: MOV BX, [2000H]
MOV AX, ES:[BX]

3.15 求出以下各十六进制数与十六进制数 62A0H 之和, 并根据结果设置标志位 SF、ZF、CF 和 OF 的值。

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

答: (1) 和为 74D4H; SF=0, ZF=0, CF=0, OF=0

(2) 和为 A5C1H; SF=1, ZF=0, CF=0, OF=1

(3) 和为 3240H; SF=0, ZF=0, CF=1, OF=0

(4) 和为 0000H; SF=0, ZF=1, CF=1, OF=0

3.16 求出以下各十六进制数与十六进制数 4AE0H 的差值, 并根据结果设置标志位 SF、ZF、CF 和 OF 的值。

(1) 1234H (2) 5D90H (3) 9090H (4) EA04H

答: (1) 差为 C754H; SF=1, ZF=0, CF=1, OF=0

(2) 差为 12B0H; SF=0, ZF=0, CF=0, OF=0

(3) 差为 45B0H; SF=0, ZF=0, CF=0, OF=1

(4) 差为 9F24H; SF=1, ZF=0, CF=0, OF=0

3.17 写出执行以下计算的指令序列, 其中 X、Y、Z、R、W 均为存放 16 位带符号数单元的地址。

(1) $Z \leftarrow W + (Z - X)$ (2) $Z \leftarrow W - (X + 6) - (R + 9)$

(3) $Z \leftarrow (W * X) / (Y + 6)$, R ← 余数 (4) $Z \leftarrow ((W - X) / 5 * Y) * 2$

答: (1) MOV AX, Z ; 以下程序都未考虑带符号数的溢出

SUB AX, X

ADD AX, W

MOV Z, AX

(2) MOV BX, X

ADD BX, 6

MOV CX, R

ADD CX, 9

MOV AX, W

SUB AX, BX

SUB AX, CX

MOV Z, AX

(3) ADD Y, 6

MOV AX, W

IMUL X

IDIV Y

MOV Z, AX

MOV R, DX

(4) MOV AX, W

SUB AX, X

CWD

MOV BX, 5

IDIV BX

IMUL Y

SHL AX, 1 ; ((DX),(AX))*2

RCL DX, 1

3.18 已知程序段如下:

MOV AX, 1234H ; (AX)=1234H, 标志位不变

MOV CL, 4 ; (AX)和标志位都不变

ROL AX, CL ; (AX)=2341H, CF=1, SF 和 ZF 不变


```

DEC    AX           ; (AX)=2340H, CF=1 不变, SF=0, ZF=0
MOV    CX, 4        ; (AX)和标志位都不变
MUL    CX           ; (AX)=8D00H, CF=OF=0, 其它标志无定义
INT    20H

```

试问:

- (1) 每条指令执行完后, AX 寄存器的内容是什么?
- (2) 每条指令执行完后, 进位、符号和零标志的值是什么?
- (3) 程序结束时, AX 和 DX 的内容是什么?

答: (1) 见注释;

(2) 见注释;

(3) (AX)=8D00H, (DX)=0

3.19 下列程序段中的每条指令执行完后, AX 寄存器及 CF、SF、ZF 和 OF 的内容是什么?

```

MOV    AX, 0        ; (AX)=0, 标志位不变
DEC    AX           ; (AX)=0FFFFH, CF 不变, SF=1, ZF=0, OF=0
ADD    AX, 7FFFH     ; (AX)=7FFEh, CF=1, SF=0, ZF=0, OF=0
ADD    AX, 2         ; (AX)=8000H, CF=0, SF=1, ZF=0, OF=1
NOT    AX           ; (AX)=7FFFH, 标志位不变
SUB    AX, 0FFFFH    ; (AX)=8000H, CF=1, SF=1, ZF=0, OF=1
ADD    AX, 8000H     ; (AX)=0, CF=1, SF=0, ZF=1, OF=1
SUB    AX, 1         ; (AX)=0FFFFH, CF=1, SF=1, ZF=0, OF=0
AND    AX, 58D1H     ; (AX)=58D1H, CF=0, SF=0, ZF=0, OF=0
SAL    AX, 1         ; (AX)=0B1A2H, CF=0, SF=1, ZF=0, OF=1
SAR    AX, 1         ; (AX)=0D8D1H, CF=0, SF=1, ZF=0, OF=0
NEG    AX           ; (AX)= 272FH, CF=1, SF=0, ZF=0, OF=0
ROR    AX, 1         ; (AX)= 9397H, CF=1, SF 和 ZF 不变, OF=1

```

答: 见注释。

3.20 变量 DATAX 和变量 DATAY 的定义如下:

```

DATAX  DW  0148H
        DW  2316H
DATAY  DW  0237H
        DW  4052H

```

请按下列要求写出指令序列:

- (1) DATAX 和 DATAY 两个字数据相加, 和存放在 DATAY 中。
- (2) DATAX 和 DATAY 两个双字数据相加, 和存放在从 DATAY 开始的双字单元中。
- (3) 解释下列指令的作用:

```

STC
MOV    BX, DATAX
ADC    BX, DATAY

```

- (4) DATAX 和 DATAY 两个字数据相乘(用 MUL)。
- (5) DATAX 和 DATAY 两个双字数据相乘(用 MUL)。
- (6) DATAX 除以 23(用 DIV)。
- (7) DATAX 双字除以字 DATAY (用 DIV)。

答: (1) MOV AX, DATAX

ADD DATAY, AX

MOV AX, DATAX+2

ADD DATAY+2, AX

(2) MOV AX, DATAX

ADD DATAY, AX

MOV AX, DATAX+2

ADC DATAY+2, AX

MOV DATAY+4, 0

; 用于存放进位位

ADC DATAY+4, 0

(3) DATAX 和 DATAY 两个字数据之和加 1, 结果存入 BX 寄存器。

(4) RESULT1 DW 0

```

                DW 0
RESULT2 DW 0
                DW 0
                ⋮
MOV  AX, DATAX
MUL  DATAY
MOV  RESULT1 , AX
MOV  RESULT1+2, DX
MOV  AX, DATAX+2
MUL  DATAY+2
MOV  RESULT2 , AX
MOV  RESULT2+2, DX
(5) AA  DW 0
    BB  DW 0
    CC  DW 0
    DD  DW 0
        ⋮
MOV  AX, DATAX
MUL  DATAY
MOV  AA , AX
MOV  BB, DX
MOV  AX, DATAX
MUL  DATAY+2
ADD  BB, AX
ADC  CC, DX
MOV  AX, DATAX+2
MUL  DATAY
ADD  BB, AX
ADC  CC, DX
ADC  DD, 0
MOV  AX, DATAX+2
MUL  DATAY+2
ADD  CC, AX
ADC  DD, DX
(6) MOV  AX, DATAX
    MOV  BL, 23
    DIV  BL
(7) MOV  DX, DATAX+2
    MOV  AX, DATAX
    DIV  DATAY

```

3.21 写出对存放在 DX 和 AX 中的双字长数求补的指令序列。

答: NEG	DX	也可为:	NOT	DX
NEG	AX		NOT	AX
SBB	DX, 0		ADD	AX, 1
			ADC	DX, 0

3.22 试编写一程序求出双字长数的绝对值。双字长数在 A 和 A+2 单元中，结果存放在 B 和 B+2 单元中。

答：程序段如下：

```

                MOV  AX, A
                MOV  DX, A+2
                CMP  DX, 0
                JNS  ZHENSHU    ; 不是负数则转走
                NEG  DX
                NEG  AX
                SBB  DX, 0
ZHENSHU: MOV  B, AX
                MOV  B+2, DX
                INT  20H

```


3.23 假设(BX)=0E3H, 变量 VALUE 中存放的内容为 79H, 确定下列各条指令单独执行后的结果。

- (1) XOR BX, VALUE ; (BX)=9AH, CF、OF 都为 0, AF 无定义, SF=1, ZF=0, PF=1
 (2) AND BX, VALUE ; (BX)=61H, CF、OF 都为 0, AF 无定义, SF=0, ZF=0, PF=0
 (3) OR BX, VALUE ; (BX)=0FBH, CF、OF 都为 0, AF 无定义, SF=1, ZF=0, PF=0
 (4) XOR BX, 0FFH ; (BX)=1CH, CF、OF 都为 0, AF 无定义, SF=0, ZF=0, PF=0
 (5) AND BX, 0 ; (BX)=00H, CF、OF 都为 0, AF 无定义, SF=0, ZF=1, PF=1
 (6) TEST BX, 01H ; (BX)=0E3H, CF、OF 都为 0, AF 无定义, SF=1, ZF=0, PF=0
 答: 见注释。

3.24 试写出执行下列指令序列后 BX 寄存器的内容。执行前(BX)=6D16H。

```
MOV CL, 7
SHR BX, CL
```

答: (BX)=00DAH。

3.25 试用移位指令把十进制数+53 和-49 分别乘以 2。它们应该用什么指令? 得到的结果是什么? 如果要除以 2 呢?

```
答: MOV AL, 53
      SAL AL, 1 ; (AL)=(+53*2)=6AH
      MOV AL, -49
      SAL AL, 1 ; (AL)=(-49*2)=9EH
      MOV AL, 53
      SAR AL, 1 ; (AL)=(53/2)= 1AH
      MOV AL, -49
      SAR AL, 1 ; (AL)=(-49/2)=0E7H
```

3.26 试分析下面的程序段完成什么功能?

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

答: 本程序段将 ((DX),(AX)) 的双字同时左移 4 位, 即将此双字乘以 10H(16)。

3.27 假定(DX)=0B9H, (CL)=3, (CF)=1, 确定下列各条指令单独执行后 DX 中的值。

- (1) SHR DX, 1 ; (DX)=05CH
 (2) SAR DX, CL ; (DX)=17H
 (3) SHL DX, CL ; (DX)=5C8H
 (4) SHL DL, 1 ; (DX)=72H
 (5) ROR DX, CL ; (DX)=2017H
 (6) ROL DL, CL ; (DX)=0CDH
 (7) SAL DH, 1 ; (DX)=0B9H
 (8) RCL DX, CL ; (DX)=2CCH
 (4) RCR DL, 1 ; (DX)=0DCH

答: 见注释。

3.28 下列程序段执行完后, BX 寄存器的内容是什么?

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

答: (BX)=0C02DH。

3.29 假设数据段定义如下:

```
CONAME DB 'SPACE EXPLORERS INC.'
PRLINE DB 20 DUP('')
```

用串指令编写程序段分别完成以下功能:

- (1) 从左到右把 CONAME 中的字符串传送到 PRLINE。

- (2) 从右到左把 CONAME 中的字符串传送到 PRLINE。
- (3) 把 CONAME 中的第 3 和第 4 个字节装入 AX。
- (4) 把 AX 寄存器的内容存入从 PRLINE+5 开始的字节中。
- (5) 检查 CONAME 字符串中是否有空格字符, 如有则把第一个空格字符的地址传送给 BX 寄存器。

答: (1) MOV CX, 20

```

    CLD
    MOV SI, SEG CONAME
    MOV DS, SI
    MOV ES, SI
    LEA SI, CONAME
    LEA DI, PRLINE
    REP MOVSB
(2) MOV CX, 20
    STD
    MOV SI, SEG CONAME
    MOV DS, SI
    MOV ES, SI
    LEA SI, CONAME
    ADD SI, 20-1
    LEA DI, PRLINE
    ADD DI, 20-1
    REP MOVSB
(3) MOV AX, WORD PTR CONAME+3-1
(4) MOV WORD PTR PRLINE +5, AX
(5) MOV AL, ' ' ; 空格的 ASCII 码送 AL 寄存器
    CLD
    MOV DI, SEG CONAME
    MOV ES, DI
    LEA DI, CONAME
    REPNE SCASB
    JNE NEXT
    DEC DI
    MOV BX, DI
NEXT:  ;

```

3.30 编写程序段, 把字符串 STRING 中的‘&’字符用空格符代替。

STRING DB ‘The date is FEB&03’

答: 程序段如下:

```

    MOV CX, 18
    MOV AL, '&'
    CLD
    MOV DI, SEG STRING
    MOV ES, DI
    LEA DI, STRING
    REPNE SCASB
    JNE NEXT
    DEC DI
    MOV ES: BYTE PTR [DI], ' ' ; 送空格符
NEXT:  ;

```

3.31 假设数据段中数据定义如下:

```

STUDENT_NAME DB 30 DUP (?)
STUDENT_ADDR DB 9 DUP (?)
PRINT_LINE DB 132 DUP (?)

```

分别编写下列程序段:

- (1) 用空格符清除 PRINT_LINE 域。
- (2) 在 STUDENT_ADDR 中查找第一个‘-’。
- (3) 在 STUDENT_ADDR 中查找最后一个‘-’。
- (4) 如果 STUDENT_NAME 域中全是空格符时, 填入‘*’。

- (5) 把 STUDENT_NAME 移到 PRINT_LINE 的前 30 个字节中，把 STUDENT_ADDR 移到 PRINT_LINE 的后 9 个字节中。

答：公共的程序段如下：

```

    MOV     DI, DS
    MOV     ES, DI
(1) MOV     CX, 132
    MOV     AL, ' '           ; 空格的 ASCII 码送 AL 寄存器
    CLD
    LEA     DI, PRINT_LINE
    REP     STOSB
(2) MOV     CX, 9
    MOV     AL, '-'
    CLD
    LEA     DI, STUDENT_ADDR
    REPNE   SCASB
    JNE     NO_DASH
    DEC     DI
NO_DASH:   ⋮
(3) MOV     CX, 9
    MOV     AL, '-'
    STD
    LEA     DI, STUDENT_ADDR
    ADD     DI, 9-1
    REPNE   SCASB
    JNE     NO_DASH
    INC     DI
NO_DASH:   ⋮
(4) MOV     CX, 30
    MOV     AL, ' '           ; 空格的 ASCII 码送 AL 寄存器
    CLD
    LEA     DI, STUDENT_NAME
    REPE    SCASB
    JNE     NEXT
    MOV     CX, 30
    MOV     AL, '*'           ; "*" 的 ASCII 码送 AL 寄存器
    LEA     DI, STUDENT_NAME
    REP     STOSB
NEXT:      ⋮
(5) MOV     CX, 30
    CLD
    LEA     SI, STUDENT_NAME
    LEA     DI, PRINT_LINE
    REP     MOVSB
    MOV     CX, 9
    STD
    LEA     SI, STUDENT_ADDR+9-1
    LEA     DI, PRINT_LINE+132-1
    REP     MOVSB

```

- 3.32 编写一程序段：比较两个 5 字节的字符串 OLDS 和 NEWS，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。

答：程序段如下：

```

    MOV     CX, 5
    CLD
    MOV     DI, SEG  OLDS
    MOV     DS, DI
    MOV     ES, DI
    LEA     SI, OLDS
    LEA     DI, NEWS

```

```

    REPE    CMPSB
    JNE     NEW_LESS
    .
    .
    .
NEW_LESS:
    .
    .
    .

```

3.33 假定 AX 和 BX 中的内容为带符号数, CX 和 DX 中的内容为无符号数, 请用比较指令和条件转移指令实现以下判断:

- (1) 若 DX 的内容超过 CX 的内容, 则转去执行 EXCEED。
- (2) 若 BX 的内容大于 AX 的内容, 则转去执行 EXCEED。
- (3) 若 CX 的内容等于 0, 则转去执行 ZERO。
- (4) BX 与 AX 的内容相比较是否产生溢出? 若溢出则转 OVERFLOW。
- (5) 若 BX 的内容小于等于 AX 的内容, 则转 EQ_SMA。
- (6) 若 DX 的内容低于等于 CX 的内容, 则转 EQ_SMA。

答: (1) CMP DX, CX
 JA EXCEED
 (2) CMP BX, AX
 JG EXCEED
 (3) JCXZ ZERO
 (4) CMP BX, AX
 JO OVERFLOW
 (5) CMP BX, AX
 JLE EQ_SMA
 (6) CMP DX, CX
 JBE EQ_SMA

3.34 试分析下列程序段:

```

ADD     AX, BX
JNO     L1
JNC     L2
SUB     AX, BX
JNC     L3
JNO     L4
JMP     SHORT L5

```

如果 AX 和 BX 的内容给定如下:

	AX	BX
(1)	147BH	80DCH
(2)	B568H	42C8H
(3)	42C8H	608DH
(4)	D023H	9FD0H
(5)	94B7H	B568H

问该程序分别在上面 5 种情况下执行后, 程序转向哪里?

答: (1) 转向 L1
 (2) 转向 L1
 (3) 转向 L2
 (4) 转向 L5 ; 因为加法指令后 AX 中已经是 6FF3H
 (5) 转向 L5 ; 因为加法指令后 AX 中已经是 4A14H

3.35 指令 CMP AX, BX 后面跟着一条格式为 J... L1 的条件转移指令, 其中...可以是 B、NB、BE、NBE、L、NL、LE、NLE 中的任意一个。如果 AX 和 BX 的内容给定如下:

	AX	BX
(1)	1F52H	1F52H
(2)	88C9H	88C9H
(3)	FF82H	007EH
(4)	58BAH	020EH
(5)	FFC5H	FF8BH
(6)	09A0H	1E97H
(7)	8AEAH	FC29H
(8)	D367H	32A6H

问以上 8 条转移指令中的哪几条将引起转移到 L1?

- 答：(1) JNB、JBE、JNL、JLE
 (2) JNB、JBE、JNL、JLE
 (3) JNB、JNBE、JL、JLE
 (4) JNB、JNBE、JNL、JNLE
 (5) JNB、JNBE、JL、JLE
 (6) JB、JBE、JL、JLE
 (7) JB、JBE、JNL、JNLE
 (8) JNB、JNBE、JL、JLE

3.36 假设 X 和 X+2 单元的内容为双精度数 p，Y 和 Y+2 单元的内容为双精度数 q，(X 和 Y 为低位字) 试说明下列程序段做什么工作？

```

MOV  DX, X+2
MOV  AX, X
ADD  AX, X
ADC  DX, X+2
CMP  DX, Y+2
JL   L2
JG   L1
CMP  AX, Y
JBE  L2
L1:  MOV  AX, 1
      JMP  SHORT  EXIT
L2:  MOV  AX, 2
EXIT: INT  20H

```

答：此程序段判断 $p*2 > q$ ，则使 (AX)=1 后退出； $p*2 \leq q$ ，则使 (AX)=2 后退出。

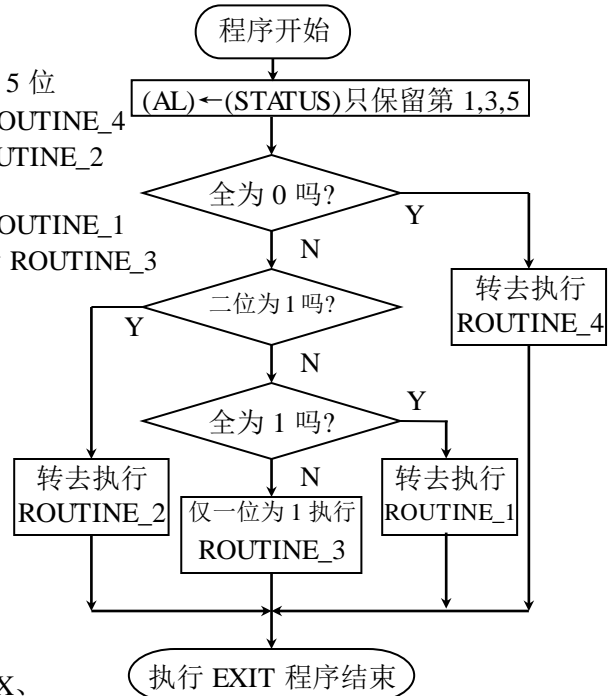
3.37 要求测试在 STATUS 中的一个字节，如果第 1、3、5 位均为 1 则转移到 ROUTINE_1；如果此三位中有两位为 1 则转移到 ROUTINE_2；如果此三位中只有一位为 1 则转移到 ROUTINE_3；如果此三位全为 0 则转移到 ROUTINE_4。试画出流程图，并编制相应的程序段。

答：程序段如下：

```

MOV  AL, STATUS
AND  AL, 00010101B ; 只保留第 1、3、5 位
JZ   ROUTINE_4     ; 3 位全为 0 转 ROUTINE_4
JPE  ROUTINE_2     ; 两位为 1 转 ROUTINE_2
CMP  AL, 00010101B
JZ   ROUTINE_1     ; 3 位全为 1 转 ROUTINE_1
ROUTINE_3:  ; 仅一位为 1 执行 ROUTINE_3
      JMP  EXIT
ROUTINE_1:  ;
      JMP  EXIT
ROUTINE_2:  ;
      JMP  EXIT
ROUTINE_4:  ;
EXIT: INT  20H

```



3.44 题的程序流程图

3.38 在下列程序的括号中分别填入如下指令：

- (1) LOOP L20
 (2) LOOPE L20
 (3) LOOPNE L20

试说明在三种情况下，当程序执行完后，AX、BX、CX、DX 四个寄存器的内容分别是什么？

```

TITLE      EXLOOP.COM
CODESEG    SEGMENT
      ASSUME CS:CODESEG, DS:CODSEG, SS:CODSEG
      ORG 100H
BEGIN: MOV  AX, 01
      MOV  BX, 02
      MOV  DX, 03

```

```

        MOV     CX, 04
L20:
        INC     AX
        ADD     BX, AX
        SHR     DX, 1
        (      )
        RET
CODESG  ENDS
        END     BEGIN

```

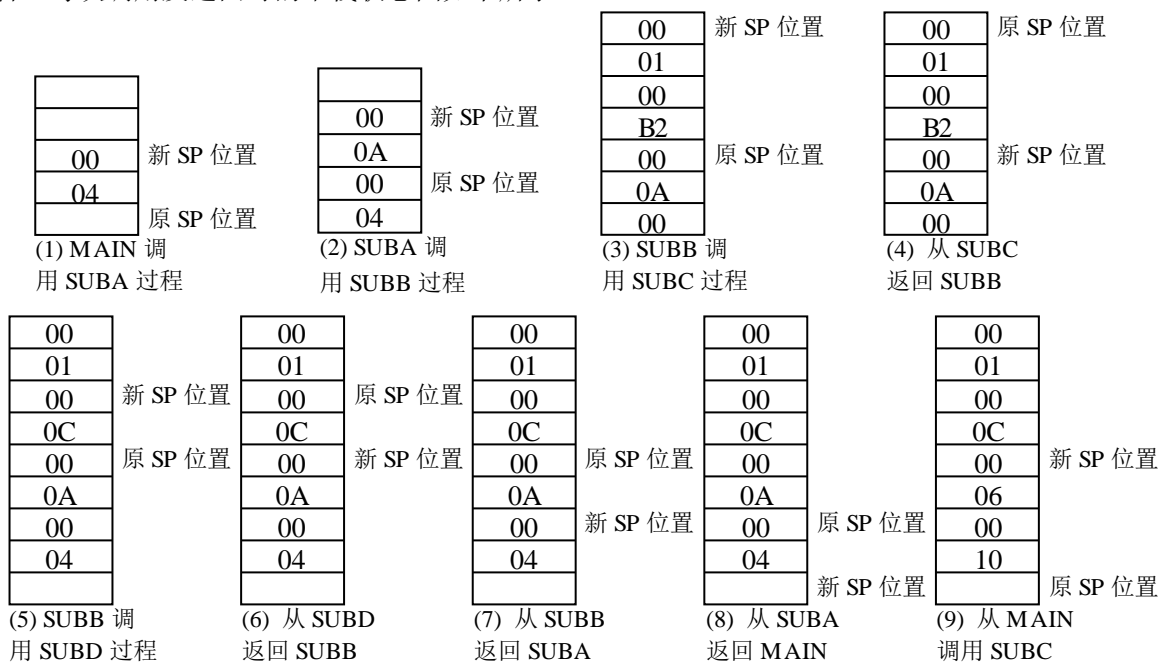
答: (1) (AX)=5H, (BX)=10H, (CX)=0H, (DX)=0H
 (2) (AX)=2H, (BX)=4H, (CX)=3H, (DX)=1H
 (3) (AX)=3H, (BX)=7H, (CX)=2H, (DX)=0H

3.39 考虑以下的调用序列:

- (1) MAIN 调用 NEAR 的 SUBA 过程(返回的偏移地址为 0400);
- (2) SUBA 调用 NEAR 的 SUBB 过程(返回的偏移地址为 0A00);
- (3) SUBB 调用 FAR 的 SUBC 过程(返回的段地址为 B200, 返回的偏移地址为 0100);
- (4) 从 SUBC 返回 SUBB;
- (5) SUBB 调用 NEAR 的 SUBD 过程(返回的偏移地址为 0C00);
- (6) 从 SUBD 返回 SUBB;
- (7) 从 SUBB 返回 SUBA;
- (8) 从 SUBA 返回 MAIN;
- (9) 从 MAIN 调用 SUBC(返回的段地址为 1000, 返回的偏移地址为 0600);

请画出每次调用及返回时的堆栈状态。

答: 每次调用及返回时的堆栈状态图如下所示:



3.40 假设(EAX)=00001000H, (EBX)=00002000H, (DS)=0010H, 试问下列指令访问内存的物理地址是什么?

- (1) MOV ECX, [EAX+EBX]
- (2) MOV [EAX+2*EBX], CL
- (3) MOV DH, [EBX+4*EAX+1000H]

答: (1) PA=(DS)*10H+EA=00100H+00001000H+00002000H=00003100H
 (2) PA=(DS)*10H+EA=00100H+00001000H+2*00002000H=00005100H
 (3) PA=(DS)*10H+EA=00100H+00002000H+4*00001000H+1000H=00007100H

3.41 假设(EAX)=9823F456H, (ECX)=1F23491H, (BX)=348CH, (SI)=2000H, (DI)=4044H。在 DS 段中从偏移地址 4044H 单元开始的 4 个字节单元中, 依次存放的内容为 92H, 6DH, 0A2H 和 4CH,

试问下列各条指令执行完后的目的地址及其中的内容是什么？

- ```
(1) MOV [SI], EAX
(2) MOV [BX], ECX
(3) MOV EBX, [DI]
```

答: (1) 目的地址为 DS:2000H, 内容依次为: 56H, 0F4H, 23H 和 98H

(2) 目的地址为 DS:348CH, 内容依次为: 91H, 34H, 0F2H 和 01H

(3) 目的操作数为 EBX 寄存器, (EBX)=4CA26D92H

### 3.42 说明下列指令的操作

- |     |        |      |                        |
|-----|--------|------|------------------------|
| (1) | PUSH   | AX   | ； 将(AX)压入堆栈            |
| (2) | POP    | ESI  | ； 将堆栈中的双字弹出到 ESI 寄存器中  |
| (3) | PUSH   | [BX] | ； 将((BX))对应存储单元中的字压入堆栈 |
| (4) | PUSHAD |      | ； 32 位通用寄存器依次进栈        |
| (5) | POP    | DS   | ； 将堆栈中的字弹出到 DS 寄存器中    |
| (6) | PUSH   | 4    | ； 将立即数 4 以字的方式压入堆栈     |

答：见注释。

3.43 请给出下列各指令序列执行完后目的寄存器的内容。

- ```
(1)  MOV    EAX, 299FF94H
      ADD    EAX, 34FFFFFH      ; (EAX)= 2CEFF93H
(2)  MOV    EBX, 40000000
      SUB    EBX, 1500000      ; (EBX)= 3EB00000H
(3)  MOV    EAX, 39393834H
      AND    EAX, 0F0F0F0FH    ; (EAX)= 09090804H
(4)  MOV    EDX, 9FE35DH
      XOR    EDX, 0F0F0F0H     ; (EDX)= 6F13ADH
```

答：见注释。

3.44 请给出下列各指令序列执行完后目的寄存器的内容。

- | | | | |
|-----|-------|---------|----------------------|
| (1) | MOV | BX, -12 | |
| | MOVSB | EBX, BX | ; (EBX)= 0FFFF FFF4H |
| (2) | MOV | CL, -8 | |
| | MOVSB | EDX, CL | ; (EDX)= 0FFFF FFF8H |
| (3) | MOV | AH, 7 | |
| | MOVSB | ECX, AH | ; (ECX)= 0000 0007H |
| (4) | MOV | AX, 99H | |
| | MOVSB | EBX, AX | ; (EBX)= 0000 0099H |

答：见注释。

3.45 请给出下列指令序列执行完后 EAX 和 EBX 的内容。

- ```
MOV ECX, 307 F455H
BSF EAX, ECX ; (EAX)= 0D
BSR EBX, ECX ; (EBX)= 25D
```

答：见注释。

3.46 请给出下列指令序列执行完后 AX 和 DX 的内容。

- |     |         |            |
|-----|---------|------------|
| MOV | BX, 98H |            |
| BSF | AX, BX  | ; (AX)= 3D |
| BSR | DX, BX  | ; (DX)= 7D |

答：见注释。

3.47 请编写一程序段，要求把 ECX、EDX 和 ESI 的内容相加，其和存入 EDI 寄存器中(不考虑溢出)。

|        |          |         |          |
|--------|----------|---------|----------|
| 答: MOV | EDI, 0   | 也可: MOV | EDI, ECX |
| ADD    | EDI, ECX | ADD     | EDI, EDX |
| ADD    | EDI, EDX | ADD     | EDI, ESI |
| ADD    | EDI, ESI |         |          |



3.48 请说明 IMUL BX, DX, 100H 指令的操作。

答: (BX) ← (DX) \* 100H

3.49 试编写一程序段, 要求把 BL 中的数除以 CL 中的数, 并把其商乘以 2, 最后的结果存入 DX 寄存器中。

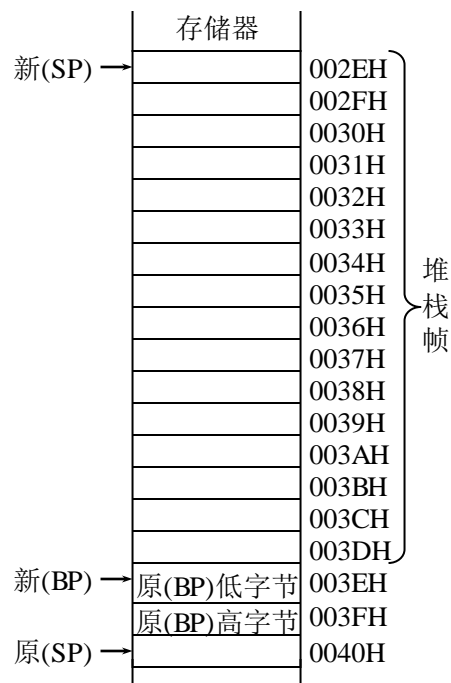
答: MOV AL, BL  
MOV AH, 0 ; 假定为无符号数, 否则用 CBW 指令即可  
DIV CL  
MOV AH, 0  
SHL AX, 1  
MOV DX, AX

3.50 请说明 JMP DI 和 JMP [DI] 指令的区别。

答: JMP DI 是转移到以(DI)内容为偏移地址的单元去执行指令; JMP [DI] 是转移到以(DI)间接寻址的内存单元内容为偏移地址的单元去执行指令。

3.51 试编写一程序段, 要求在长度为 100H 字节的数组中, 找出大于 42H 的无符号数的个数并存入字节单元 UP 中; 找出小于 42H 的无符号数的个数并存入字节单元 DOWN 中。

答: JMP BEGIN  
UP DB 0  
DOWN DB 0  
TABLE DB 100H DUP (?) ; 数组  
BEGIN:  
MOV CX, 100H  
MOV BX, -1  
MOV SI, 0  
MOV DI, 0  
L1: INC BX  
CMP TABLE[BX], 42H  
JA L2  
JB L3  
JMP L4  
L2: INC SI  
JMP L4  
L3: INC DI  
L4: LOOP L1  
MOV UP, SI  
MOV DOWN, DI



3.52 请用图表示 ENTER 16, 0 所生成的堆栈帧的情况。

答: 答案见右图。

3.52 题的答案

## 第四章. 习 题

4.1 指出下列指令的错误:

- (1) MOV AH, BX ; 寄存器类型不匹配
  - (2) MOV [BX], [SI] ; 不能都是存储器操作数
  - (3) MOV AX, [SI][DI] ; [SI]和[DI]不能一起使用
  - (4) MOV MYDAT [BX][SI], ES:AX ; AX 寄存器不能使用段超越
  - (5) MOV BYTE PTR [BX], 1000 ; 1000 超过了一个字节的范围
  - (6) MOV BX, OFFSET MYDAT [SI] ; MYDAT [SI]已经是偏移地址,不能再使用 OFFSET
  - (7) MOV CS, AX ; CS 不能用作目的寄存器
  - (8) MOV ECX, AX ; 两个操作数的数据类型不同
- 答: 见注释。

4.2 下面哪些指令是非法的? (假设 OP1, OP2 是已经用 DB 定义的变量)

- (1) CMP 15, BX ; 错, 立即数不能作为目的操作数
- (2) CMP OP1, 25
- (3) CMP OP1, OP2 ; 错, 不能都是存储器操作数
- (4) CMP AX, OP1 ; 错, 类型不匹配, 应为 CMP ax, word ptr op1

答: 见注释。

4.3 假设下列指令中的所有标识符均为类型属性为字的变量, 请指出下列哪些指令是非法的? 它们的错误是什么?

- (1) MOV BP, AL ; 错, 寄存器类型不匹配
- (2) MOV WORD\_OP [BX+4\*3][DI], SP
- (3) MOV WORD\_OP1, WORD\_OP2 ; 错, 不能都是存储器操作数
- (4) MOV AX, WORD\_OP1[DX] ; 错, DX 不能用于存储器寻址
- (5) MOV SAVE\_WORD, DS
- (6) MOV SP, SS:DATA\_WORD [BX][SI]
- (7) MOV [BX][SI], 2 ; 错, [BX][SI]未指出数据类型
- (8) MOV AX, WORD\_OP1+WORD\_OP2 ; 错, 计算表达式中不可包含变量, 只能包含常量
- (9) MOV AX, WORD\_OP1-WORD\_OP2+100 ; 错, [计算表达式中不可包含变量, 只能包含常量
- (10) MOV WORD\_OP1, WORD\_OP1-WORD\_OP2 ; 错, ①计算表达式中不可包含变量, 只能包含常量; ②mov指令不支持内存到内存的寻址

答: 见注释。

4.4 假设 VAR1 和 VAR2 为字变量, LAB 为标号, 试指出下列指令的错误之处:

- (1) ADD VAR1, VAR2 ; 不能都是存储器操作数
- (2) SUB AL, VAR1 ; 数据类型不匹配
- (3) JMP LAB [SI] ; LAB 是标号而不是变量名, 后面不能加[SI]
- (4) JNZ VAR1 ; VAR1 是变量而不是标号
- (5) JMP NEAR LAB ; 应使用 NEAR PTR

答: 见注释。

4.5 画图说明下列语句所分配的存储空间及初始化的数据值。

- (1) BYTE\_VAR DB 'BYTE', 12, -12H, 3 DUP(0, ?, 2 DUP(1, 2), ?)
- (2) WORD\_VAR DW 5 DUP(0, 1, 2), ?, -5, 'BY', 'TE', 256H

答: 答案如下图所示。

4.6 试列出各种方法, 使汇编程序把 5150H 存入一个存储器字中(如: DW 5150H)。

答: DW 5150H

DB 50H, 51H

DB 'PQ'

DW 'QP'

ORG 5150H

DW \$

| BYTE_VAR | WORD_VAR     |
|----------|--------------|
| 42H      | 00H          |
| 59H      | 00H          |
| 54H      | 01H          |
| 45H      | 00H          |
| 0DH      | 02H          |
| EEH      | 00H          |
| 00H      | 将上面内容再重复 4 次 |
| -        |              |
| 01H      |              |
| 02H      |              |
| 01H      | -            |
| 02H      | -            |
| -        | EBH          |
| 00H      | EFH          |
| -        | 00H          |
| 01H      | 59H          |
| 02H      | 42H          |
| 01H      | 45H          |
| 02H      | 54H          |
| -        | 56H          |
| -        | 02H          |

4.7 请设置一个数据段 DATASG, 其中定义以下字符变量或数据变量。

- (1) FLD1B 为字符串变量: 'personal computer';
- (2) FLD2B 为十进制数字字节变量: 32;
- (3) FLD3B 为十六进制数字字节变量: 20;
- (4) FLD4B 为二进制数字字节变量: 01011001;
- (5) FLD5B 为数字的 ASCII 字符字节变量: 32654;
- (6) FLD6B 为 10 个零的字节变量;
- (7) FLD7B 为零件名(ASCII 码)及其数量(十进制数)的表格:

PART1 20

PART2 50

PART3 14

- (8) FLD1W 为十六进制数字变量: FFF0;

4.5 题答案

- (9) FLD2W 为二进制数的字变量: 01011001;  
 (10) FLD3W 为(7)零件表的地址变量;  
 (11) FLD4W 为包括 5 个十进制数的字变量: 5, 6, 7, 8, 9;  
 (12) FLD5W 为 5 个零的字变量;  
 (13) FLD6W 为本段中字数据变量和字节数据变量之间的地址差。

答: DATASG SEGMENT

```

 FLD1B DB 'personal computer'
 FLD2B DB 32
 FLD3B DB 20H
 FLD4B DB 01011001B
 FLD5B DB '32654'
 FLD6B DB 10 DUP (0)
 FLD7B DB 'PART1', 20
 DB 'PART2', 50
 DB 'PART3', 14
 FLD1W DW 0FFF0H
 FLD2W DW 01011001B
 FLD3W DW FLD7B
 FLD4W DW 5, 6, 7, 8, 9
 FLD5W DW 5 DUP (0)
 FLD6W DW FLD1W-FLD1B

```

DATASG ENDS

4.8 假设程序中的数据定义如下:

```

PARTNO DW ?
PNAME DB 16 DUP (?)
COUNT DD ?
PLENTH EQU $-PARTNO

```

问 PLENTH 的值为多少? 它表示什么意义?

答: PLENTH=22=16H, 它表示变量 PARTNO、PNAME、COUNT 总共占用的存储单元数(字节数)。

4.9 有符号定义语句如下:

```

BUFF DB 1, 2, 3, '123'
EBUFF DB 0
L EQU EBUFF - BUFF

```

问 L 的值是多少?

答: L=6。

4.10 假设程序中的数据定义如下:

```

LNAME DB 30 DUP (?)
ADDRESS DB 30 DUP (?)
CITY DB 15 DUP (?)
CODE_LIST DB 1, 7, 8, 3, 2

```

- (1) 用一条 MOV 指令将 LNAME 的偏移地址放入 AX。
- (2) 用一条指令将 CODE\_LIST 的头两个字节的內容放入 SI。
- (3) 用一条伪操作使 CODE\_LENGTH 的值等于 CODE\_LIST 域的实际长度。

答: (1) MOV AX, OFFSET LNAME

(2) MOV SI, WORD PTR CODE\_LIST

(3) CODE\_LENGTH EQU \$ - CODE\_LIST ; 此语句必须放在 CODE\_LIST 语句之后

4.11 试写出一个完整的数据段 DATA\_SEG, 它把整数 5 赋予一个字节, 并把整数-1, 0, 2, 5 和 4 放在 10 字数组 DATA\_LIST 的头 5 个单元中。然后, 写出完整的代码段, 其功能为: 把 DATA\_LIST 中头 5 个数中的最大值和最小值分别存入 MAX 和 MIN 单元中。

答: DATA\_SEG SEGMENT

```

 NUM DB 5
 DATA_LIST DW -1, 0, 2, 5, 4, 5 DUP (?)
 MAX DW ?
 MIN DW ?
 DATA_SEG ENDS

```

```

; -----
CODE_SEG SEGMENT
MAIN PROC FAR
 ASSUME CS: CODE_SEG, DS: DATA_SEG
 START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DATA_SEG ; 给 DS 赋值
 MOV DS, AX
;
 MOV CX, 4 ; 程序段开始
 LEA BX, DATA_LIST
 MOV AX, [BX]
 MOV MAX, AX
 MOV MIN, AX
ROUT1: ADD BX, 2
 MOV AX, [BX]
 CMP AX, MAX
 JNGE ROUT2
 MOV MAX, AX
ROUT2: CMP AX, MIN
 JNLE ROUT3
 MOV MIN, AX
ROUT3: LOOP ROUT1 ; 程序段结束
 RET
MAIN ENDP
CODE_SEG ENDS
; -----
END START

```

4.12 给出等值语句如下:

```

ALPHA EQU 100
BETA EQU 25
GAMMA EQU 2

```

下列表达式的值是多少?

- (1)  $ALPHA * 100 + BETA$  ; =2729H
- (2)  $ALPHA \text{ MOD } GAMMA + BETA$  ; =19H
- (3)  $(ALPHA + 2) * BETA - 2$  ; =9F4H
- (4)  $(BETA / 3) \text{ MOD } 5$  ; =3H
- (5)  $(ALPHA + 3) * (BETA \text{ MOD } GAMMA)$  ; =67H
- (6)  $ALPHA \text{ GE } GAMMA$  ; =0FFFFH
- (7)  $BETA \text{ AND } 7$  ; =01H
- (8)  $GAMMA \text{ OR } 3$  ; =03H

答: 见注释。

4.13 对于下面的数据定义, 三条 MOV 指令分别汇编成什么? (可用立即数方式表示)

```

TABLEA DW 10 DUP (?)
TABLEB DB 10 DUP (?)
TABLEC DB '1234'
;
MOV AX, LENGTH TABLEA ; 汇编成 MOV AX, 000AH
MOV BL, LENGTH TABLEB ; 汇编成 MOV BL, 000AH
MOV CL, LENGTH TABLEC ; 汇编成 MOV CL, 0001H

```

答: 见注释。

4.14 对于下面的数据定义, 各条 MOV 指令单独执行后, 有关寄存器的内容是什么?

```

FLDB DB ?
TABLEA DW 20 DUP (?)

```

TABLEB DB 'ABCD'

- |                           |              |
|---------------------------|--------------|
| (1) MOV AX, TYPE FLDB     | ; (AX)=0001H |
| (2) MOV AX, TYPE TABLEA   | ; (AX)=0002H |
| (3) MOV CX, LENGTH TABLEA | ; (CX)=0014H |
| (4) MOV DX, SIZE TABLEA   | ; (DX)=0028H |
| (5) MOV CX, LENGTH TABLEB | ; (CX)=0001H |

答：见注释。

4.15 指出下列伪操作表达方式的错误，并改正之。

- |                         |                                   |
|-------------------------|-----------------------------------|
| (1) DATA_SEG SEG        | ; DATA_SEG SEGMENT (伪操作错)         |
| (2) SEGMENT 'CODE'      | ; SEGNAME SEGMENT 'CODE' (缺少段名字)  |
| (3) MYDATA SEGMENT/DATA | ; MYDATA SEGMENT                  |
| :                       |                                   |
| ENDS                    | ; MYDATA ENDS (缺少段名字)             |
| (4) MAIN_PROC PROC FAR  | ; 删除 END MAIN_PROC 也可以            |
| :                       |                                   |
| END MAIN_PROC           | ; MAIN_PROC ENDP ; 上下两句交换位置       |
| MAIN_PROC ENDP          | ;                   END MAIN_PROC |

答：见注释。

4.16 按下面的要求写出程序的框架

- (1) 数据段的位置从 0E000H 开始，数据段中定义一个 100 字节的数组，其类型属性既是字又是字节；
- (2) 堆栈段从小段开始，段组名为 STACK；
- (3) 代码段中指定段寄存器，指定主程序从 1000H 开始，给有关段寄存器赋值；
- (4) 程序结束。

答：程序的框架如下：

```
DATA_SEG SEGMENT AT 0E000H
 ARRAY_B LABEL BYTE
 ARRAY_W DW 50 DUP (?)
DATA_SEG ENDS ; 以上定义数据段
; -----
STACK_SEG SEGMENT PARA STACK 'STACK'
 DW 100H DUP (?)
 TOS LABEL WORD
STACK_SEG ENDS ; 以上定义堆栈段
; -----
CODE_SEG SEGMENT
 MAIN PROC FAR
 ASSUME CS: CODE_SEG, DS: DATA_SEG, SS: STACK_SEG
 ORG 1000H
 START: MOV AX, STACK_SEG
 MOV SS, AX ; 给 SS 赋值
 MOV SP, OFFSET TOS ; 给 SP 赋值
 PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DATA_SEG
 MOV DS, AX ; 给 DS 赋值
 ; ; 程序段部分
 RET
 MAIN ENDP
CODE_SEG ENDS ; 以上定义代码段
; -----
END START
```

4.17 写一个完整的程序放在代码段 C\_SEG 中，要求把数据段 D\_SEG 中的 AUGEND 和附加段 E\_SEG

中的 ADDEND 相加, 并把结果存放在 D\_SEG 段中的 SUM 中。其中 AUGEND、ADDEND 和 SUM 均为双精度数, AUGEND 赋值为 99251, ADDEND 赋值为 -15962。

答: 程序如下:

```

D_SEG SEGMENT
 AUGW LABEL WORD
 AUGEND DD 99251
 SUM DD ?
D_SEG ENDS ; 以上定义数据段
; -----
E_SEG SEGMENT
 ADDW LABEL WORD
 ADDEND DD -15962
E_SEG ENDS ; 以上定义附加段
; -----
C_SEG SEGMENT
 MAIN PROC FAR
 ASSUME CS: C_SEG, DS: D_SEG, ES: E_SEG
 START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, D_SEG
 MOV DS, AX ; 给 DS 赋值
 MOV AX, E_SEG
 MOV ES, AX ; 给 ES 赋值
 ;
 MOV AX, AUGW ; 以下 6 条指令进行加法计算
 MOV BX, AUGW+2
 ADD AX, ES: ADDW
 ADC BX, ES: ADDW+2 ; 不考虑有符号数溢出
 MOV WORD PTR SUM, AX
 MOV WORD PTR [SUM+2], BX
 RET
 MAIN ENDP
C_SEG ENDS ; 以上定义代码段
; -----
 END START

```

4.18 请说明表示程序结束的微操作和结束程序执行的语句之间的差别。它们在源程序中应如何表示?

答: 表示程序结束的微操作是指示汇编程序 MASM 结束汇编的标志, 在源程序中用 END 表示; 结束程序执行的语句是结束程序运行而返回操作系统的指令, 在源程序中有多种表示方法, 比如 INT 20H 或 MOV AX, 4C00H INT 21H 以及 RET 等。

4.19 试说明下述指令中哪些需要加上 PTR 操作符:

```

BVAL DB 10H, 20H
WVAL DW 1000H
(1) MOV AL, BVAL ; 不需要
(2) MOV DL, [BX] ; 不需要
(3) SUB [BX], 2 ; 需要, 如 SUB BYTE PTR [BX], 2
(4) MOV CL, WVAL ; 需要, 如 MOV CL, BYTE PTR WVAL
(5) ADD AL, BVAL+1 ; 不需要

```

答: 见注释。

## 第五章. 习 题

5.1 试编写一个汇编语言程序, 要求对键盘输入的小写字母用大写字母显示出来。

答: 程序段如下:

```

BEGIN: MOV AH, 1 ; 从键盘输入一个字符的 DOS 调用
 INT 21H
 CMP AL, 'a' ; 输入字符<'a'吗?
 JB STOP
 CMP AL, 'z' ; 输入字符>'z'吗?
 JA STOP
 SUB AL, 20H ; 转换为大写字母, 用 AND AL, 1101 1111B 也可
 MOV DL, AL ; 显示一个字符的 DOS 调用
 MOV AH, 2
 INT 21H
 JMP BEGIN
STOP: RET

```

**5.2** 编写程序，从键盘接收一个小写字母，然后找出它的前导字符和后续字符，再按顺序显示这三个字符。

答：程序段如下：

```

BEGIN: MOV AH, 1 ; 从键盘输入一个字符的 DOS 调用
 INT 21H
 CMP AL, 'a' ; 输入字符<'a'吗?
 JB STOP
 CMP AL, 'z' ; 输入字符>'z'吗?
 JA STOP
 DEC AL ; 得到前导字符
 MOV DL, AL ; 准备显示三个字符
 MOV CX, 3
DISPLAY: MOV AH, 2 ; 显示一个字符的 DOS 调用
 INT 21H
 INC DL
 LOOP DISPLAY
STOP: RET

```

**5.3** 将 AX 寄存器中的 16 位数分成 4 组，每组 4 位，然后把这四组数分别放在 AL、BL、CL 和 DL 中。

答：程序段如下：

```

DSEG SEGMENT
STORE DB 4 DUP (?)
DSEG ENDS
 :
BEGIN: MOV CL, 4 ; 右移四次
 MOV CH, 4 ; 循环四次
 LEA BX, STORE
A10: MOV DX, AX
 AND DX, 0FH ; 取 AX 的低四位
 MOV [BX], DL ; 低四位存入 STORE 中
 INC BX
 SHR AX, CL ; 右移四次
 DEC CH
 JNZ A10 ; 循环四次完了吗?
B10: MOV DL, STORE
 MOV CL, STORE+1
 MOV BL, STORE+2
 MOV AL, STORE+3
STOP: RET

```

**5.4** 试编写一程序，要求比较两个字符串 STRING1 和 STRING2 所含字符是否完全相同，若相同则显示'MATCH'，若不相同则显示'NO MATCH'。

答：程序如下：

```

DSEG SEGMENT
STRING1 DB 'I am a student.'

```



```

 STRING2 DB 'I am a student!'
 YES DB 'MATCH', 0DH, 0AH, '$'
 NO DB 'NO MATCH', 0DH, 0AH, '$'
DSEG
ENDS
; -----
CSEG SEGMENT
MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG, ES: DSEG
START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值
 MOV ES, AX ; 给 ES 赋值
;
BEGIN: LEA SI, STRING1 ; 设置串比较指令的初值
 LEA DI, STRING2
 CLD
 MOV CX, STRING2 - STRING1
 REPE CMPSB ; 串比较
 JNE DISPNO
 LEA DX, YES ; 显示 MATCH
 JMP DISPLAY
DISPNO: LEA DX, NO ; 显示 NO MATCH
DISPLAY: MOV AH, 9 ; 显示一个字符串的 DOS 调用
 INT 21H
 RET
MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
END START

```

5.5 试编写一程序，要求能从键盘接收一个个位数 N，然后响铃 N 次(响铃的 ASCII 码为 07)。

答：程序段如下：

```

BEGIN: MOV AH, 1 ; 从键盘输入一个字符的 DOS 调用
 INT 21H
 SUB AL, '0'
 JB STOP ; 输入字符<'0'吗?
 CMP AL, 9 ; 输入字符>'9'吗?
 JA STOP
 CBW
 MOV CX, AX ; 响铃次数 N
 JCXZ STOP
BELL: MOV DL, 07H ; 准备响铃
 MOV AH, 2 ; 显示一个字符的 DOS 调用，实际为响铃
 INT 21H
 CALL DELAY100ms ; 延时 100ms
 LOOP BELL
STOP: RET

```

5.6 编写程序，将一个包含有 20 个数据的数组 M 分成两个数组：正数数组 P 和负数数组 N，并分别把这两个数组中数据的个数显示出来。

答：程序如下：

```

DSEG SEGMENT
COUNT EQU 20
ARRAY DW 20 DUP (?) ; 存放数组
COUNT1 DB 0 ; 存放正数的个数

```

```

 ARRAY1 DW 20 DUP (?) ; 存放正数
 COUNT2 DB 0 ; 存放负数的个数
 ARRAY2 DW 20 DUP (?) ; 存放负数
 ZHEN DB 0DH, 0AH, 'The positive number is: ', '$' ; 正数的个数是:
 FU DB 0DH, 0AH, 'The negative number is: ', '$' ; 负数的个数是:
 CRLF DB 0DH, 0AH, '$'
DSEG ENDS

; -----
CSEG SEGMENT
MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG

START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

BEGIN: MOV CX, COUNT
 LEA BX, ARRAY
 LEA SI, ARRAY1
 LEA DI, ARRAY2
BEGIN1: MOV AX, [BX]
 CMP AX, 0 ; 是负数吗?
 JS FUSHU
 MOV [SI], AX ; 是正数, 存入正数数组
 INC COUNT1 ; 正数个数+1
 ADD SI, 2
 JMP SHORT NEXT
FUSHU: MOV [DI], AX ; 是负数, 存入负数数组
 INC COUNT2 ; 负数个数+1
 ADD DI, 2
NEXT: ADD BX, 2
 LOOP BEGIN1
 LEA DX, ZHEN ; 显示正数个数
 MOV AL, COUNT1
 CALL DISPLAY ; 调显示子程序
 LEA DX, FU ; 显示负数个数
 MOV AL, COUNT2
 CALL DISPLAY ; 调显示子程序
 RET
MAIN ENDP

; -----
DISPLAY PROC NEAR ; 显示子程序
 MOV AH, 9 ; 显示一个字符串的 DOS 调用
 INT 21H
 AAM ; 将(AL)中的二进制数转换为二个非压缩 BCD 码
 ADD AH, '0' ; 变为 0~9 的 ASCII 码
 MOV DL, AH
 MOV AH, 2 ; 显示一个字符的 DOS 调用
 INT 21H
 ADD AL, '0' ; 变为 0~9 的 ASCII 码
 MOV DL, AL
 MOV AH, 2 ; 显示一个字符的 DOS 调用
 INT 21H
 LEA DX, CRLF ; 显示回车换行
 MOV AH, 9 ; 显示一个字符串的 DOS 调用

```

```

 INT 21H
 RET
 DISPLAY ENDP ; 显示子程序结束
 CSEG ENDS ; 以上定义代码段
; -----
 END START

```

5.7 试编写一个汇编语言程序，求出首地址为 DATA 的 100D 字数组中的最小偶数，并把它存放在 AX 中。

答：程序段如下：

```

BEGIN: MOV BX, 0
 MOV CX, 100
COMPARE: MOV AX, DATA[BX] ; 取数组的第一个偶数
 ADD BX, 2
 TEST AX, 01H ; 是偶数吗？
 LOOPNZ COMPARE ; 不是，比较下一个数
 JNZ STOP ; 没有偶数，退出
 JCXZ STOP ; 最后一个数是偶数，即为最小偶数，退出
COMPARE1: MOV DX, DATA[BX] ; 取数组的下一个偶数
 ADD BX, 2
 TEST DX, 01H ; 是偶数吗？
 JNZ NEXT ; 不是，比较下一个数
 CMP AX, DX ; (AX)<(DX)吗？
 JLE NEXT
 MOV AX, DX ; (AX)<(DX)，则置换(AX)为最小偶数
NEXT: LOOP COMPARE1
STOP: RET

```

5.8 把 AX 中存放的 16 位二进制数 K 看作是 8 个二进制的“四分之一字节”。试编写程序要求数一下值为 3(即 11B)的四分之一字节数，并将该数(即 11B 的个数)在终端上显示出来。

答：程序段如下：

```

BEGIN: MOV DL, 0 ; 计数初始值
 MOV CX, 8
COMPARE: TEST AX, 03H ; 是数 03 吗？
 JNZ NOEQUAL ; 不是，转走
 INC DL ; 是，计数
NOEQUAL: ROR AX, 1 ; 准备判断下一个数
 ROR AX, 1
 LOOP COMPARE
 ADD DL, '0' ; 将计数值转换为 ASCII 码
 MOV AH, 2 ; 进行显示
 INT 21H
STOP: RET

```

**5.9** 试编写一个汇编语言程序，要求从键盘接收一个四位的 16 进制数，并在终端上显示与它等值的二进制数。

答：程序段如下：

```

BEGIN: MOV BX, 0 ; 用于存放四位的 16 进制数
 MOV CH, 4
 MOV CL, 4
INPUT: SHL BX, CL ; 将前面输入的数左移 4 位
 MOV AH, 1 ; 从键盘取数
 INT 21H
 CMP AL, 30H ; <0 吗？
 JB INPUT ; 不是‘0~F’的数重新输入
 CMP AL, 39H ; 是‘0~9’吗？
 JA AF ; 不是，转 ‘A~F’ 的处理

```

```

 AND AL, 0FH ; 转换为: 0000B~1001B
 JMP BINARY
AF: AND AL, 1101 1111B ; 转换为大写字母
 CMP AL, 41H ; 又<A 吗?
 JB INPUT ; 不是'A~F'的数重新输入
 CMP AL, 46H ; >F 吗?
 JA INPUT ; 不是'A~F'的数重新输入
 AND AL, 0FH ; 转换为: 1010B~1111B
 ADD AL, 9
BINARY: OR BL, AL ; 将键盘输入的数进行组合
 DEL CH
 JNZ INPUT
DISPN: MOV CX, 16 ; 将 16 位二进制数一位位地转换成 ASCII 码显示
DISP: MOV DL, 0
 ROL BX, 1
 RCL DL, 1
 OR DL, 30H
 MOV AH, 2 ; 进行显示
 INT 21H
 LOOP DISP
STOP: RET

```

5.10 设有一段英文，其字符变量名为 ENG，并以\$字符结束。试编写一程序，查对单词 SUN 在该文中的出现次数，并以格式“SUN: xxxx”显示出次数。

答：程序如下：

```

DSEG SEGMENT
ENG DB 'Here is sun, sun ,..., $'
DISP DB 'SUN: '
DAT DB '0000', 0DH, 0AH, '$'
KEYWORD DB 'sun'
DSEG ENDS
; -----
CSEG SEGMENT
MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG, ES: DSEG
START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值
 MOV ES, AX ; 给 ES 赋值

BEGIN: MOV AX, 0
 MOV DX, DISP-ENG-2 ; 计算 ENG 的长度(每次比较 sun,因此比较次数-2)
 LEA BX, ENG
COMP: MOV DI, BX
 LEA SI, KEYWORD
 MOV CX, 3
 REPE CMPSB ; 串比较
 JNZ NOMATCH
 INC AX ; 是, SUN 的个数加 1
 ADD BX, 2
NOMATCH: INC BX ; 指向 ENG 的下一个字母
 DEC DX
 JNZ COMP
DONE: MOV CH, 4 ; 将次数转换为 16 进制数的 ASCII 码
 MOV CL, 4
 LEA BX, DAT ; 转换结果存入 DAT 单元中

```

```

DONE1: ROL AX, CL
 MOV DX, AX
 AND DL, 0FH ; 取一位 16 进制数
 ADD DL, 30H
 CMP DL, 39H
 JLE STORE
 ADD DL, 07H ; 是“A~F”所以要加 7
STORE: MOV [BX], DL ; 转换结果存入 DAT 单元中
 INC BX
 DEC CH
 JNZ DONE1
DISPLAY: LEA DX, DISP ; 显示字符串程序(将 DISP 和 DAT 一起显示)
 MOV AH, 09H
 INT 21H
 RET
MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

5.11 从键盘输入一系列以\$为结束符的字符串，然后对其中的非数字字符计数，并显示出计数结果。

答：程序段如下：

```

DSEG SEGMENT
 BUFF DB 50 DUP (' ')
 COUNT DW 0
DSEG ENDS
;
BEGIN: LEA BX, BUFF
 MOV COUNT, 0
INPUT: MOV AH, 01 ; 从键盘输入一个字符的功能调用
 INT 21H
 MOV [BX], AL
 INC BX
 CMP AL, '$' ; 是$结束符吗？
 JNZ INPUT ; 不是，继续输入
 LEA BX, BUFF ; 对非数字字符进行计数
NEXT: MOV CL, [BX]
 INC BX
 CMP CL, '$' ; 是$结束符，则转去显示
 JZ DISP
 CMP CL, 30H ; 小于 0 是非数字字符
 JB NEXT
 CMP CL, 39H ; 大于 9 是非数字字符
 JA NEXT
 INC COUNT ; 个数+1
 JMP NEXT
DISP: ;
; 16 进制数显示程序段(省略)

```

5.12 有一个首地址为 MEM 的 100D 字数组，试编程序删除数组中所有为 0 的项，并将后续项向前压缩，最后将数组的剩余部分补上 0。

答：程序如下：

```

DSEG SEGMENT
 MEM DW 100 DUP (?)
DSEG ENDS
; -----
CSEG SEGMENT
 MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG

```

```

START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

BEGIN: MOV SI, (100-1)*2 ; (SI)指向 MEM 的末元素的首地址
 MOV BX, -2 ; 地址指针的初值
 MOV CX, 100
COMP: ADD BX, 2
 CMP MEM [BX], 0
 JZ CONS
 LOOP COMP
 JMP FINISH ; 比较完了, 已无 0 则结束
CONS: MOV DI, BX
CONS1: CMP DI, SI ; 到了最后单元码?
 JAE NOMOV
 MOV AX, MEM [DI+2] ; 后面的元素向前移位
 MOV MEM [DI], AX
 ADD DI, 2
 JMP CONS1
NOMOV: MOV WORD PTR [SI], 0 ; 最后单元补 0
 LOOP COMP
FINISH: RET
MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

5.13 在 STRING 到 STRING+99 单元中存放着一个字符串, 试编制一个程序测试该字符串中是否存在数字, 如有则把 CL 的第 5 位置 1, 否则将该位置 0。

答: 程序如下:

```

DSEG SEGMENT
STRING DB 100 DUP (?)
DSEG ENDS
; -----
CSEG SEGMENT
MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

BEGIN: MOV SI, 0 ; (SI)作为地址指针的变化值
 MOV CX, 100
REPEAT: MOV AL, STRING [SI]
 CMP AL, 30H
 JB GO_ON
 CMP AL, 39H
 JA GO_ON
 OR CL, 20H ; 存在数字把 CL 的第 5 位置 1
 JMP EXIT
GO_ON: INC SI
 LOOP REPEAT
 AND CL, 0DFH ; 不存在数字把 CL 的第 5 位置 0
EXIT: RET

```

```

 MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

5.14 在首地址为 TABLE 的数组中按递增次序存放着 100H 个 16 位补码数，试编写一个程序把出现次数最多的数及其出现次数分别存放于 AX 和 CX 中。

答：程序如下：

```

DSEG SEGMENT
 TABLE DW 100H DUP (?) ; 数组中的数据是按增序排列的
 DATA DW ?
 COUNT DW 0
DSEG ENDS
; -----
CSEG SEGMENT
 MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
 START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

 BEGIN: MOV CX, 100H ; 循环计数器
 MOV SI, 0
 NEXT: MOV DX, 0
 MOV AX, TABLE [SI]
 COMP: CMP TABLE [SI], AX ; 计算一个数的出现次数
 JNE ADDR
 INC DX
 ADD SI, 2
 LOOP COMP
 ADDR: CMP DX, COUNT ; 此数出现的次数最多吗?
 JLE DONE
 MOV COUNT, DX ; 目前此数出现的次数最多，记下次数
 MOV DATA, AX ; 记下此数
 DONE: LOOP NEXT ; 准备取下一个数
 MOV CX, COUNT ; 出现最多的次数存入(CX)
 MOV AX, DATA ; 出现最多的数存入(AX)
 RET
 MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

5.15 数据段中已定义了一个有 n 个字数据的数组 M，试编写一程序求出 M 中绝对值最大的数，把它放在数据段的 M+2n 单元中，并将该数的偏移地址存放在 M+2(n+1)单元中。

答：程序如下：

```

DSEG SEGMENT
 n EQU 100H ; 假设 n=100H
 M DW n DUP (?)
 DATA DW ? ; M+2n 单元
 ADDR DW ? ; M+2(n+1)单元
DSEG ENDS
; -----
CSEG SEGMENT
 MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG

```



```

START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

BEGIN: MOV CX, n ; 循环计数器
 LEA DI, M
 MOV AX, [DI] ; 取第一个数
 MOV ADDR, DI ; 记下绝对值最大的数的地址
 CMP AX, 0 ; 此数是正数吗?
 JNS ZHEN ; 是正数, 即为绝对值, 转去判断下一个数
 NEG AX ; 不是正数, 变为其绝对值

ZHEN: MOV BX, [DI]
 CMP BX, 0 ; 此数是正数吗?
 JNS COMP ; 是正数, 即为绝对值, 转去比较绝对值大小
 NEG BX ; 不是正数, 变为其绝对值

COMP: CMP AX, BX ; 判断绝对值大小
 JAE ADDRESS
 MOV AX, BX ; (AX)<(BX), 使(AX)中为绝对值最大的数
 MOV ADDR, DI ; 记下绝对值最大的数的地址

ADDRESS: ADD DI, 2
 LOOP ZHEN
 MOV DATA, AX ; 记下此数
 RET

MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

5.16 在首地址为 DATA 的字数组中存放着 100H 个 16 位补码数, 试编写一个程序求出它们的平均值放在 AX 寄存器中; 并求出数组中有多少个数小于此平均值, 将结果放在 BX 寄存器中。

答: 程序如下:

```

DSEG SEGMENT
DATA DW 100H DUP (?)
DSEG ENDS
; -----
CSEG SEGMENT
MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG

START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

BEGIN: MOV CX, 100H ; 循环计数器
 MOV SI, 0
 MOV BX, 0 ; 和((DI),(BX))的初始值
 MOV DI, 0
NEXT: MOV AX, DATA [SI]
 CWD
 ADD BX, AX ; 求和
 ADC DI, DX ; 加上进位位
 ADD SI, 2
 LOOP NEXT
 MOV DX, DI ; 将((DI),(BX))中的累加和放入((DX),(AX))中

```

```

 MOV AX, BX
 MOV CX, 100H
 IDIV CX ; 带符号数求平均值, 放入(AX)中
 MOV BX, 0
 MOV SI, 0
COMP: CMP AX, DATA [SI] ; 寻找小于平均值的数
 JLE NO
 INC BX ; 小于平均值数的个数+1
NO: ADD SI, 2
 LOOP COMP
 RET
MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

5.17 试编制一个程序把 AX 中的 16 进制数转换为 ASCII 码, 并将对应的 ASCII 码依次存放到 MEM 数组中的四个字节中。例如, 当(AX)=2A49H 时, 程序执行完后, MEM 中的 4 个字节内容为 39H, 34H, 41H, 32H。

答: 程序如下:

```

DSEG SEGMENT
MEM DB 4 DUP (?)
N DW 2A49H
DSEG ENDS
; -----
CSEG SEGMENT
MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

BEGIN: MOV CH, 4 ; 循环计数器
 MOV CL, 4
 MOV AX, N
 LEA BX, MEM
ROTATE: MOV DL, AL ; 从最低四位开始转换为 ASCII 码
 AND DL, 0FH
 ADD DL, 30H
 CMP DL, 3AH ; 是 0~9 吗?
 JL NEXT
 ADD DL, 07H ; 是 A~F
NEXT: MOV [BX], DL ; 转换的 ASCII 码送入 MEM 中
 INC BX
 ROR AX, CL ; 准备转换下一位
 DEC CH
 JNZ ROTATE
 RET
MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

5.18 把 0~100D 之间的 30 个数存入以 GRADE 为首地址的 30 字数组中, GRADE+i 表示学号为 i+1 的学生的成绩。另一个数组 RANK 为 30 个学生的名次表, 其中 RANK+i 的内容是学号为 i+1 的学生的名次。编写一程序, 根据 GRADE 中的学生成绩, 将学生名次填入 RANK 数组中。(提示: 一

个学生的名次等于成绩高于这个学生的人数加 1。)

答：程序如下：

```

DSEG SEGMENT
 GRADE DW 30 DUP (?) ; 假设已预先存好 30 名学生的成绩
 RANK DW 30 DUP (?)
DSEG ENDS
; -----
CSEG SEGMENT
 MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
 START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

 BEGIN: MOV DI, 0
 MOV CX, 30 ; 外循环计数器
 LOOP1: PUSH CX
 MOV CX, 30 ; 内循环计数器
 MOV SI, 0
 MOV AX, GRADE [DI]
 MOV DX, 1 ; 起始名次为第 1 名
 LOOP2: CMP GRADE [SI], AX ; 成绩比较
 JBE GO_ON
 INC DX ; 名次+1
 GO_ON: ADD SI, 2
 LOOP LOOP2
 POP CX
 MOV RANK [DI], DX ; 名次存入 RANK 数组
 ADD DI, 2
 LOOP LOOP1
 RET
 MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

**5.19** 已知数组 A 包含 15 个互不相等的整数，数组 B 包含 20 个互不相等的整数。试编制一程序把既在 A 中又在 B 中出现的整数存放于数组 C 中。

答：程序如下：

```

DSEG SEGMENT
 A DW 15 DUP (?)
 B DW 20 DUP (?)
 C DW 15 DUP (' ')
DSEG ENDS
; -----
CSEG SEGMENT
 MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
 START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

 BEGIN: MOV SI, 0
 MOV BX, 0

```

```

 MOV CX, 15 ; 外循环计数器
LOOP1: PUSH CX
 MOV CX, 20 ; 内循环计数器
 MOV DI, 0
 MOV AX, A[SI] ; 取 A 数组中的一个数
LOOP2: CMP B[DI], AX ; 和 B 数组中的数相等吗?
 JNE NO
 MOV C[BX], AX ; 相等存入 C 数组中
 ADD BX, 2
 ADD DI, 2
 LOOP LOOP2
 ADD SI, 2
 POP CX
 LOOP LOOP1
 RET
MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

5.20 设在 A、B 和 C 单元中分别存放着三个数。若三个数都不是 0，则求出三数之和存放在 D 单元中；若其中有一个数为 0，则把其它两单元也清 0。请编写此程序。

答：程序如下：

```

DSEG SEGMENT
A DW ?
B DW ?
C DW ?
D DW 0
DSEG ENDS
; -----
CSEG SEGMENT
MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

BEGIN: CMP A, 0
 JE NEXT
 CMP B, 0
 JE NEXT
 CMP C, 0
 JE NEXT
 MOV AX, A
 ADD AX, B
 ADD AX, C
 MOV D, AX
 JMP SHORT EXIT
NEXT: MOV A, 0
 MOV B, 0
 MOV C, 0
EXIT: RET
MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

5.21 试编写一程序，要求比较数组 ARRAY 中的三个 16 位补码数，并根据比较结果在终端上显示如下信息：

- (1) 如果三个数都不相等则显示 0；
- (2) 如果三个数有二个数相等则显示 1；
- (3) 如果三个数都相等则显示 2。

答：程序如下：

```

DSEG SEGMENT
 ARRAY DW 3 DUP (?)
DSEG ENDS
; -----
CSEG SEGMENT
 MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG

 START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

 BEGIN: LEA SI, ARRAY
 MOV DX, 0 ; (DX)用于存放所求的结果
 MOV AX, [SI]
 MOV BX, [SI+2]
 CMP AX, BX ; 比较第一和第二两个数是否相等
 JNE NEXT1
 INC DX
 NEXT1: CMP [SI+4], AX ; 比较第一和第三两个数是否相等
 JNE NEXT2
 INC DX
 NEXT2: CMP [SI+4], BX ; 比较第二和第三两个数是否相等
 JNE NUM
 INC DX
 NUM: CMP DX, 3
 JL DISP
 DEC DX
 DISP: ADD DL, 30H ; 转换为 ASCII 码
 MOV AH, 2 ; 显示一个字符
 INT 21H
 RET
 MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

5.22 从键盘输入一系列字符(以回车符结束)，并按字母、数字、及其它字符分类计数，最后显示出这三类的计数结果。

答：程序如下：

```

DSEG SEGMENT
 ALPHABET DB '输入的字母字符个数为: ', '$'
 NUMBER DB '输入的数字字符个数为: ', '$'
 OTHER DB '输入的其它字符个数为: ', '$'
 CRLF DB 0DH, 0AH, '$'
DSEG ENDS
; -----
CSEG SEGMENT
 MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG

```

```

START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

BEGIN: MOV BX, 0 ; 字母字符计数器
 MOV SI, 0 ; 数字字符计数器
 MOV DI, 0 ; 其它字符计数器
INPUT: MOV AH, 1 ; 输入一个字符
 INT 21H
 CMP AL, 0DH ; 是回车符吗?
 JE DISP
 CMP AL, 30H ; <数字 0 吗?
 JAE NEXT1
OTHER: INC DI ; 是其它字符
 JMP SHORT INPUT
NEXT1: CMP AL, 39H ; >数字 9 吗?
 JA NEXT2
 INC SI ; 是数字字符
 JMP SHORT INPUT
NEXT2: CMP AL, 41H ; <字母 A 吗?
 JAE NEXT3
 JMP SHORT OTHER ; 是其它字符
NEXT3: CMP AL, 5AH ; >字母 Z 吗?
 JA NEXT4
 INC BX ; 是字母字符 A~Z
 JMP SHORT INPUT
NEXT4: CMP AL, 61H ; <字母 a 吗?
 JAE NEXT5
 JMP SHORT OTHER ; 是其它字符
NEXT5: CMP AL, 7AH ; >字母 z 吗?
 JA SHORT OTHER ; 是其它字符
 INC BX ; 是字母字符 a~z
 JMP SHORT INPUT

DISP: LEA DX, ALPHABET
 CALL DISPLAY
 LEA DX, NUMBER
 MOV BX, SI
 CALL DISPLAY
 LEA DX, OTHER
 MOV BX, DI
 CALL DISPLAY
 RET
MAIN ENDP

; -----
DISPLAY PROC NEAR
 MOV AH, 09H ; 显示字符串功能调用
 INT 21H
 CALL BINIHEX ; 调把 BX 中二进制数转换为 16 进制显示子程序
 LEA DX, CRLF
 MOV AH, 09H ; 显示回车换行
 INT 21H
 RET
DISPLAY ENDP
; -----

```

```

BINIHEX PROC NEAR ; 将 BX 中二进制数转换为 16 进制数显示子程序
 MOV CH, 4
ROTATE: MOV CL, 4
 ROL BX, CL
 MOV DL, BL
 AND DL, 0FH
 ADD DL, 30H
 CMP DL, 3AH ; 是 A~F 吗?
 JL PRINT_IT
 ADD DL, 07H
PRINT_IT: MOV AH, 02H ; 显示一个字符
 INT 21H
 DEC CH
 JNZ ROTATE
 RET
BINIHEX ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

5.23 已定义了两个整数变量 A 和 B，试编写程序完成下列功能：

- (1) 若两个数中有一个是奇数，则将奇数存入 A 中，偶数存入 B 中；
- (2) 若两个数中均为奇数，则将两数加 1 后存回原变量；
- (3) 若两个数中均为偶数，则两个变量均不改变。

答：程序如下：

```

DSEG SEGMENT
A DW ?
B DW ?
DSEG ENDS
; -----
CSEG SEGMENT
MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

BEGIN: MOV AX, A
 MOV BX, B
 XOR AX, BX
 TEST AX, 0001H ; A 和 B 同为奇数或偶数吗?
 JZ CLASS ; A 和 B 都为奇数或偶数，转走
 TEST BX, 0001H
 JZ EXIT ; B 为偶数，转走
 XCHG BX, A ; A 为偶数，将奇数存入 A 中
 MOV B, BX ; 将偶数存入 B 中
 JMP EXIT
CLASS: TEST BX, 0001H ; A 和 B 都为奇数吗?
 JZ EXIT ; A 和 B 同为偶数，转走
 INC B
 INC A
EXIT: RET
MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----

```



END      START

5.24 假设已编制好 5 个歌曲程序, 它们的段地址和偏移地址存放在数据段的跳跃表 SINGLIST 中。试编制一程序, 根据从键盘输入的歌曲编号 1~5, 转去执行五个歌曲程序中的某一个。

答: 程序如下:

```

DSEG SEGMENT
SINGLIST DD SING1
 DD SING2
 DD SING3
 DD SING4
 DD SING5
ERRMSG DB 'Error! Invalid parameter!', 0DH, 0AH, '$'
DSEG ENDS
; -----
CSEG SEGMENT
MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

BEGIN: MOV AH, 1 ; 从键盘输入的歌曲编号 1~5
 INT 21H
 CMP AL, 0DH
 JZ EXIT ; 是回车符, 则结束
 SUB AL, '1' ; 是 1~5 吗?
 JB ERROR ; 小于 1, 错误
 CMP AL, 4
 JA ERROR ; 大于 5, 错误
 MOV BX, OFFSET SINGLIST
 MUL AX, 4 ; (AX)=(AL)*4, 每个歌曲程序的首地址占 4 个字节
 ADD BX, AX
 JMP DWORD PTR[BX] ; 转去执行歌曲程序
ERROR: MOV DX, OFFSET ERRMSG
 MOV AH, 09H
 INT 21H ; 显示错误信息
 JMP BEGIN
SING1: :
 JMP BEGIN
SING2: :
 JMP BEGIN
SING3: :
 JMP BEGIN
SING4: :
 JMP BEGIN
SING5: :
 JMP BEGIN
EXIT: RET
MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
END START

```

5.25 试用 8086 的乘法指令编制一个 32 位数和 16 位数相乘的程序;再用 80386 的乘法指令编制一个 32 位数和 16 位数相乘的程序, 并定性比较两个程序的效率。

答: 8086 的程序如下(假设为无符号数):

```

DSEG SEGMENT
 MUL1 DD ? ; 32 位被乘数
 MUL2 DW ? ; 16 位乘数
 MUL0 DW 0, 0, 0, 0 ; 乘积用 64 位单元存放
DSEG ENDS
; -----
CSEG SEGMENT
 MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
 START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

 BEGIN: MOV BX, MUL2 ; 取乘数
 MOV AX, WORD PTR MUL1 ; 取被乘数低位字
 MUL BX
 MOV MUL0, AX ; 保存部分积低位
 MOV MUL0+2, DX ; 保存部分积高位
 MOV AX, WORD PTR[MUL1+2] ; 取被乘数高位字
 MUL BX
 ADD MUL0+2, AX ; 部分积低位和原部分积高位相加
 ADC MUL0+4, DX ; 保存部分积最高位，并加上进位
 EXIT: RET
 MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

80386 的程序如下(假设为无符号数):

```

.386
DSEG SEGMENT
 MUL1 DD ? ; 32 位被乘数
 MUL2 DW ? ; 16 位乘数
 MUL0 DD 0, 0 ; 乘积用 64 位单元存放
DSEG ENDS
; -----
CSEG SEGMENT
 MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
 START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

 BEGIN: MOVZX EBX, MUL2 ; 取乘数，并 0 扩展成 32 位
 MOV EAX, MUL1 ; 取被乘数
 MUL EBX
 MOV DWORD PTR MUL0, EAX ; 保存积的低位双字
 MOV DWORD PTR[MUL0+4], EDX ; 保存积的高位双字
 EXIT: RET
 MAIN ENDP
CSEG ENDS ; 以上定义代码段
; -----
 END START

```

80386 作 32 位乘法运算用一条指令即可完成，而 8086 则需用部分积作两次完成。

- 5.26 如数据段中在首地址为 MESS1 的数据区内存放着一个长度为 35 的字符串，要求把它们传送到附加段中的缓冲区 MESS2 中去。为提高程序执行效率，希望主要采用 MOVSD 指令来实现。试编写这一程序。

答：80386 的程序如下：

```
.386
.MODEL SMALL
.STACK 100H
.DATA
MESS1 DB '123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ',? ; 长度为 35 的字符串
.FARDATA
MESS2 DB 36 DUP (?)
.CODE
START: MOV AX, @DATA
 MOV DS, AX ; 给 DS 赋值
 MOV AX, @FARDATA
 MOV ES, AX ; 给 ES 赋值
 ASSUME ES:@FARDATA

BEGIN: LEA ESI, MESS1
 LEA EDI, MESS2
 CLD
 MOV ECX, (35+1)/4 ; 取传送的次数
 REP MOVSD

; -----
 MOV AX, 4C00H ; 返回 DOS
 INT 21H
 END START
```

- 5.27 试用比例变址寻址方式编写一 386 程序，要求把两个 64 位整数相加并保存结果。

答：80386 的程序如下：

```
.386
.MODEL SMALL
.STACK 100H
.DATA
DATA1 DQ ?
DATA2 DQ ?
.CODE
START: MOV AX, @DATA
 MOV DS, AX ; 给 DS 赋值

BEGIN: MOV ESI, 0
 MOV EAX, DWORD PTR DATA2[ESI*4]
 ADD DWORD PTR DATA1[ESI*4], EAX
 INC ESI
 MOV EAX, DWORD PTR DATA2[ESI*4]
 ADC DWORD PTR DATA1[ESI*4], EAX

; -----
 MOV AX, 4C00H ; 返回 DOS
 INT 21H
 END START
```

## 第六章. 习 题

- 6.1 下面的程序段有错吗？若有，请指出错误。

```
CRAY PROC
```

```

 PUSH AX
 ADD AX, BX
 RET
ENDP CRAY

```

答: 程序有错。改正如下:

```

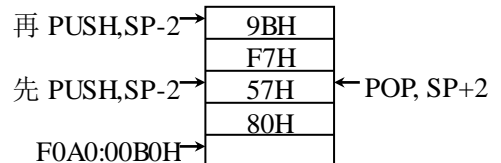
 CRAY PROC
 ADD AX, BX
 RET
 CRAY ENDP

```

; CRAY 是过程名, 应放在 ENDP 的前面

- 6.2 已知堆栈寄存器 SS 的内容是 0F0A0H, 堆栈指示器 SP 的内容是 00B0H, 先执行两条把 8057H 和 0F79BH 分别入栈的 PUSH 指令, 然后执行一条 POP 指令。试画出示意图说明堆栈及 SP 内容的变化过程。

答: 变化过程如右图所示:



6.2 题堆栈及 SP 内容的变化过程

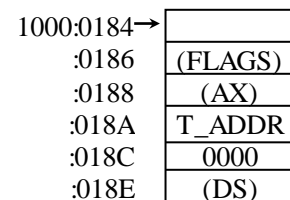
- 6.3 分析下面的程序, 画出堆栈最满时各单元的地址及内容。先 PUSH, SP-2 →

```

; *****
S_SEG SEGMENT AT 1000H ; 定义堆栈段
 DW 200 DUP (?) ; 200*2=190H
 TOS LABEL WORD
S_SEG ENDS
; *****
C_SEG SEGMENT ; 定义代码段
 ASSUME CS: C_SEG, SS: S_SEG
START: MOV AX, S_SEG
 MOV SS, AX
 MOV SP, OFFSET TOS

 PUSH DS
 MOV AX, 0
 PUSH AX
 ⋮
 PUSH T_ADDR
 PUSH AX
 PUSHF
 ⋮
 POPF
 POP AX
 POP T_ADDR
 RET
; -----
C_SEG ENDS ; 代码段结束
; *****
 END START ; 程序结束

```



SP: 0186

6.3 题堆栈最满时各单元的地址及内容

```

; -----
C_SEG ENDS ; 代码段结束
; *****
 END START ; 程序结束

```

答: 堆栈最满时各单元的地址及内容如右图所示:

- 6.4 分析下面的程序, 写出堆栈最满时各单元的地址及内容。

```

; *****
STACK SEGMENT AT 500H ; 定义堆栈段
 DW 128 DUP (?)
 TOS LABEL WORD
STACK ENDS
; *****
CODE SEGMENT ; 定义代码段
MAIN PROC FAR ; 主程序部分
 ASSUME CS: CODE, SS: STACK
START: MOV AX, STACK
 MOV SS, AX

```

```

 MOV SP, OFFSET TOS
 PUSH DS
 SUB AX, AX
 PUSH AX
; MAIN PART OF PROGRAM GOES HERE
 MOV AX, 4321H
 CALL HTOA
 RET
MAIN ENDP ; 主程序部分结束

```

```

; -----
HTOA PROC NEAR ; HTOA 子程序
 CMP AX, 15
 JLE B1
 PUSH AX
 PUSH BP
 MOV BP, SP
 MOV BX, [BP+2]
 AND BX, 000FH
 MOV [BP+2], BX
 POP BP
 MOV CL, 4
 SHR AX, CL
 CALL HTOA
 POP BP
B1: ADD AL, 30H
 CMP AL, 3AH
 JL PRINTIT
 ADD AL, 7H
PRINTIT: MOV DL, AL
 MOV AH, 2
 INT 21H
 RET
HOTA ENDP ; HOTA 子程序结束
; -----
CODE ENDS ; 代码段结束
; *****
 END START ; 程序结束

```

答：堆栈最满时各单元的地址及内容如右上图所示：

|            |              |
|------------|--------------|
| 0500:00EC→ |              |
| :00EE      | 返回 POP BP 地址 |
| :00F0      | 0003H        |
| :00F2      | 返回 POP BP 地址 |
| :00F4      | 0002H        |
| :00F6      | 返回 POP BP 地址 |
| :00F8      | 0001H        |
| :00FA      | 主程序返回地址      |
| :00FC      | 0000         |
| :00FE      | (DS)         |

SP: 00EE

6.4 题堆栈最满时各单元  
的地址及内容

6.5 下面是一个程序清单，请在下面的图中填入此程序执行过程中的堆栈变化。

```

; *****
0000 STACKSG SEGMENT
0000 20 [. DW 32 DUP (?)
 ? ? ? ?
]
0040 STACKSG ENDS
; *****
0000 CODESG SEGMENT PARA 'CODE'
; -----
0000 BEGIN PROC FAR
 ASSUME CS: CODESG, SS: STACKSG
0000 1E PUSH DS
0001 2B C0 SUB AX, AX
0003 50 PUSH AX
0004 E8 0008 R CALL B10
; -----
0007 CB RET
0008 BEGIN ENDP

```

```

; -----
0008 B10 PROC
0008 E8 000C R CALL C10
; -----
000B C3 RET
000C B10 ENDP
; -----
000C C10 PROC
; -----
000C C3 RET
000D C10 ENDP
; -----
000D CODESG ENDS
; *****
 END BEGIN

```

答：程序执行过程中的堆栈变化如下图所示。

| 偏移地址      | 堆栈    |           |         |           |          |
|-----------|-------|-----------|---------|-----------|----------|
| ( 0016H ) |       | ( 0016H ) |         | ( 0016H ) |          |
| ( 0018H ) |       | ( 0018H ) |         | ( 0018H ) |          |
| ( 001AH ) |       | ( 001AH ) |         | ( 001AH ) | 0007     |
| ( 001CH ) |       | ( 001CH ) |         | ( 001CH ) | 0000     |
| ( 001EH ) |       | ( 001EH ) | (DS)    | ( 001EH ) | (DS)     |
| ( 0020H ) |       | ( 0020H ) |         | ( 0020H ) |          |
| SP:       | 0020H |           | 001EH   |           | 001CH    |
|           | BEGIN |           | PUSH DS |           | PUSH AX  |
| ( 0016H ) |       | ( 0016H ) |         | ( 0016H ) |          |
| ( 0018H ) | 000B  | ( 0018H ) | 000B    | ( 0018H ) | 000B     |
| ( 001AH ) | 0007  | ( 001AH ) | 0007    | ( 001AH ) | 0007     |
| ( 001CH ) | 0000  | ( 001CH ) | 0000    | ( 001CH ) | 0000     |
| ( 001EH ) | (DS)  | ( 001EH ) | (DS)    | ( 001EH ) | (DS)     |
| ( 0020H ) |       | ( 0020H ) |         | ( 0020H ) |          |
| SP:       | 0018H |           | 001AH   |           | 001CH    |
|           |       |           |         |           | CALL B10 |
| ( 0016H ) |       | ( 0016H ) |         | ( 0016H ) |          |
| ( 0018H ) | 000B  | ( 0018H ) | 000B    | ( 0018H ) | 000B     |
| ( 001AH ) | 0007  | ( 001AH ) | 0007    | ( 001AH ) | 0007     |
| ( 001CH ) | 0000  | ( 001CH ) | 0000    | ( 001CH ) | 0000     |
| ( 001EH ) | (DS)  | ( 001EH ) | (DS)    | ( 001EH ) | (DS)     |
| ( 0020H ) |       | ( 0020H ) |         | ( 0020H ) |          |
| SP:       | 0018H |           | 001AH   |           | 001CH    |
|           |       |           |         |           | 0020H    |

**6.6** 写一段子程序 SKIPLINES，完成输出空行的功能。空出的行数在 AX 寄存器中。

答：程序如下：

```

CSEG SEGMENT
SKIPLINES PROC FAR
 ASSUME CS: CSEG
BEGIN: PUSH CX
 PUSH DX
 MOV CX, AX
DISP: MOV DL, 0DH ; 显示回车换行，即输出空行
 MOV AH, 2 ; 显示一个字符的 DOS 调用
 INT 21H
 MOV DL, 0AH
 MOV AH, 2 ; 显示一个字符的 DOS 调用
 INT 21H
 LOOP DISP
 POP DX
 POP CX
 RET
SKIPLINES ENDP
END

```

**6.7** 设有 10 个学生的成绩分别是 76, 69, 84, 90, 73, 88, 99, 63, 100 和 80 分。试编制一个子程序统计 60~69 分, 70~79 分, 80~89 分, 90~99 分和 100 分的人数, 分别存放到 S6, S7, S8, S9

和 S10 单元中。

答：程序如下：

```

DSEG SEGMENT
RECORD DW 76, 69, 84, 90, 73, 88, 99, 63, 100, 80
S6 DW 0
S7 DW 0
S8 DW 0
S9 DW 0
S10 DW 0
DSEG ENDS
; *****
CSEG SEGMENT
MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

BEGIN: MOV CX, 10
 CALL COUNT
 ;
 RET
MAIN ENDP
; -----
COUNT PROC NEAR ; 成绩统计子程序
 MOV SI, 0
NEXT: MOV AX, RECORD[SI]
 MOV BX, 10 ; 以下 5 句是根据成绩计算相对 S6 的地址变化量
 DIV BL ; 计算公式为: ((成绩)/10-6)*2 送(BX)
 MOV BL, AL ; 此时(BH)保持为 0 不变
 SUB BX, 6 ; 应为只统计 60 分以上成绩
 SAL BX, 1 ; (BX)*2
 INC S6[BX] ; S6 是 S6, S7, S8, S9 和 S10 单元的首地址
 ADD SI, 2
 LOOP NEXT
 RET
COUNT ENDP ; COUNT 子程序结束
; -----
CSEG ENDS ; 以上定义代码段
; *****
 END START

```

- 6.8** 编写一个有主程序和子程序结构的程序模块。子程序的参数是一个 N 字节数组的首地址 TABLE，数 N 及字符 CHAR。要求在 N 字节数组中查找字符 CHAR，并记录该字符出现的次数。主程序则要求从键盘接收一串字符以建立字节数组 TABLE，并逐个显示从键盘输入的每个字符 CHAR 以及它在 TABLE 数组中出现的次数。(为简化起见，假设出现次数≤15，可以用 16 进制形式把它显示出来。)

答：程序如下：

```

DSEG SEGMENT
TABLE DB 255 DUP (?)
N DW 255
CHAR DB ?
CHAR_N DB 0 ; 用于记录 CHAR 出现的次数
CRLF DB 0DH, 0AH, '$'
DSEG ENDS ; 以上定义数据段

```

```

; *****
STACK SEGMENT
 DW 100 DUP (?)
TOS LABEL WORD
STACK ENDS ; 以上定义堆栈段
; *****
CSEG SEGMENT
MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG, SS: STACK
START: MOV AX, STACK
 MOV SS, AX ; 给 SS 赋值
 MOV SP, OFFSET TOS ; 给 SP 赋值
 PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

BEGIN: MOV BX, 0
 MOV CX, 255 ; 最多输入 255 个字符
INPUT: MOV AH, 1 ; 从键盘接收一个字符的 DOS 功能调用
 INT 21H
 CMP AL, 0DH ; 输入回车符结束输入
 JZ IN_N
 MOV TABLE [BX], AL
 INC BX
 LOOP INPUT
IN_N: MOV N, BX ; TABLE 数组中的字符个数送 N
 CALL DISP_CRLF
IN_CHAR: MOV AH, 1 ; 从键盘接收一个字符并回显的 DOS 功能调用
 INT 21H
 CMP AL, 0DH ; 输入回车符结束
 JZ EXIT
 MOV CHAR, AL ; 输入的字符存入 CHAR 单元
 CALL SEARCH ; 调搜索字符子程序
 MOV DL, ':' ; 显示“:”，在字符 CHAR(输入时回显)的后面
 MOV AH, 2 ; 显示一个字符
 INT 21H
 MOV DL, CHAR_N ; 再显示 CHAR 出现的次数(次数≤15)
 AND DL, 0FH
 ADD DL, 30H
 CMP DL, 39H
 JBE NEXT
 ADD DL, 07H ; 是 A~F
NEXT: MOV AH, 2 ; 显示一个字符
 INT 21H
 CALL DISP_CRLF
 JMP SHORT IN_CHAR
EXIT: RET
MAIN ENDP
; -----
SEARCH PROC NEAR ; 搜索字符子程序
 MOV SI, 0
 MOV CX, N
 MOV CHAR_N, 0
 MOV AL, CHAR
ROTATE: CMP AL, TABLE [SI]

```



```

 JNZ ROTATE1
 INC CHAR_N ; 搜索到字符，则出现次数+1
ROTATE1: INC SI
 LOOP ROTATE
 RET
SEARCH ENDP ; SEARCH 子程序结束
; -----
DISP_CRLF PROC NEAR ; 显示回车换行符子程序
 LEA DX, CRLF
 MOV AH, 09H
 INT 21H
 RET
DISP_CRLF ENDP ; DISP_CRLF 子程序结束
; -----
CSEG ENDS ; 以上定义代码段
; *****
 END START

```

6.9 编写一个子程序嵌套结构的程序模块，分别从键盘输入姓名及 8 个字符的电话号码，并以一定的格式显示出来。

主程序 TELIST:

- 显示提示符 “INPUT NAME:”;
- 调用子程序 INPUT\_NAME 输入姓名;
- 显示提示符 “INPUT A TELEPHONE NUMBER:”;
- 调用子程序 INPHONE 输入电话号码;
- 调用子程序 PRINTLINE 显示姓名及电话号码。

子程序 INPUT\_NAME:

- 调用键盘输入子程序 GETCHAR，把输入的姓名存放在 INBUF 缓冲区中;
- 把 INBUF 中的姓名移入输出行 OUTNAME。

子程序 INPHONE:

- 调用键盘输入子程序 GETCHAR，把输入的 8 位电话号码存放在 INBUF 缓冲区中;
- 把 INBUF 中的号码移入输出行 OUTPHONE。

子程序 PRINTLINE:

显示姓名及电话号码，格式为:

```

NAME TEL.
X X X XXXXXXXX

```

答: 程序如下:

```

DSEG SEGMENT
INBUF DB 12 DUP (' ') ; 输入缓冲区，初始值为空格
OUTNAME DB 16 DUP (' ') ; 姓名输出行，初始值为空格
OUTPHONE DB 12 DUP (' '), 0DH, 0AH, '$' ; 号码输出行，初始值为空格
MSG1 DB 'INPUT NAME: ', '$'
MSG2 DB 'INPUT A TELEPHONE NUMBER: ', '$'
MSG3 DB 'NAME', 12 DUP (' '), 'TEL.', 0DH, 0AH, '$'
CRLF DB 0DH, 0AH, '$'
DSEG ENDS ; 以上定义数据段
; *****
STACK SEGMENT
 DW 100 DUP (?)
TOS LABEL WORD
STACK ENDS ; 以上定义堆栈段
; *****
CSEG SEGMENT
TELIST PROC FAR ; 主程序 TELIST
 ASSUME CS: CSEG, DS: DSEG, ES: DSEG, SS: STACK
START: MOV AX, STACK

```

```

 MOV SS, AX ; 给 SS 赋值
 MOV SP, OFFSET TOS ; 给 SP 赋值
 PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值
 MOV ES, AX ; 给 ES 赋值

BEGIN: LEA DX, MSG1
 MOV AH, 09H ; 显示字符串功能调用
 INT 21H
 CALL INPUT_NAME ; 输入姓名
 LEA DX, MSG2
 MOV AH, 09H ; 显示字符串功能调用
 INT 21H
 CALL INPHONE ; 输入电话号码
 CALL PRINTLINE ; 显示姓名及电话号码
 RET

TELIST ENDP

; -----
INPUT_NAME PROC NEAR ; 输入姓名子程序
 CALL GETCHAR ; 调输入字符子程序输入姓名
 LEA SI, INBUF ; 把 INBUF 中的姓名移入输出行 OUTNAME
 LEA DI, OUTNAME
 MOV CX, 12
 CLD
 REP MOVSB
 RET
INPUT_NAME ENDP ; INPUT_NAME 子程序结束

; -----
INPHONE PROC NEAR ; 输入电话号码子程序
 CALL GETCHAR ; 调输入字符子程序输入电话号码
 LEA SI, INBUF ; 把 INBUF 中的电话号码移入输出行 OUTPHONE
 LEA DI, OUTPHONE
 MOV CX, 12
 CLD
 REP MOVSB
 RET
INPHONE ENDP ; INPHONE 子程序结束

; -----
GETCHAR PROC NEAR ; 键盘输入子程序
 MOV AL, 20H ; 先将 INBUF 中填满空格字符
 MOV CX, 12
 LEA DI, INBUF
 CLD
 REP STOSB
 MOV CX, 12 ; 向 INBUF 输入字符
 MOV DI, 0
INPUT: MOV AH, 1 ; 从键盘接收一个字符并回显的 DOS 功能调用
 INT 21H
 CMP AL, 0DH ; 输入回车符返回
 JZ QUIT
 MOV INBUF[DI], AL
 INC DI
 LOOP INPUT

```

```

QUIT: CALL DISP_CRLF
 RET
GETCHAR ENDP ; GETCHAR 子程序结束
; -----
PRINTLINE PROC NEAR ; 显示姓名及电话号码子程序
 LEA DX, MSG3
 MOV AH, 09H ; 显示字符串功能调用
 INT 21H
 LEA DX, OUTNAME ; 显示姓名及电话号码
 MOV AH, 09H ; 显示字符串功能调用
 INT 21H
 RET
PRINTLINE ENDP ; PRINTLINE 子程序结束
; -----
DISP_CRLF PROC NEAR ; 显示回车换行符子程序
 LEA DX, CRLF
 MOV AH, 09H
 INT 21H
 RET
DISP_CRLF ENDP ; DISP_CRLF 子程序结束
; -----
CSEG ENDS ; 以上定义代码段
; *****
 END START

```

6.10 编写子程序嵌套结构的程序，把整数分别用二进制和八进制形式显示出来。

主程序 BANDO: 把整数变量 VAL1 存入堆栈，并调用子程序 PAIRS;

子程序 PAIRS: 从堆栈中取出 VAL1; 调用二进制显示程序 OUTBIN 显示出与其等效的二进制数; 输出 8 个空格; 调用八进制显示程序 OUTOCT 显示出与其等效的八进制数; 调用输出回车及换行符子程序。

答: 程序如下:

```

DSEG SEGMENT
VAL1 DW ?
CRLF DB 0DH, 0AH, '$'
DSEG ENDS ; 以上定义数据段
; *****
CSEG SEGMENT
BANDO PROC FAR ; 主程序 BANDO
 ASSUME CS: CSEG, DS: DSEG
START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

 PUSH VAL1
 CALL PAIRS
 RET
BANDO ENDP
; -----
PAIRS PROC NEAR ; PAIRS 子程序
 PUSH BP
 MOV BP, SP
 PUSH BX
 MOV BX, [BP+4] ; 从堆栈中取出 VAL1
 CALL OUTBIN ; 调用二进制显示子程序
 MOV CX, 8 ; 显示 8 个空格符

```

```

SPACE: MOV DL, ' '
 MOV AH, 2
 INT 21H
 LOOP SPACE
 CALL OUTOCT ; 调用八进制显示子程序
 CALL DISP_CRLF
 POP BX
 POP BP
 RET 2
PAIRS ENDP ; PAIRS 子程序结束
; -----
OUTBIN PROC NEAR ; 二进制显示子程序
 PUSH BX
 MOV CX, 16
ONEBIT: ROL BX, 1
 MOV DX, BX
 AND DX, 1
 OR DL, 30H ; 转换为 ASCII 码
 MOV AH, 2
 INT 21H
 LOOP ONEBIT
 POP BX
 RET
OUTBIN ENDP ; OUTBIN 子程序结束
; -----
OUTOCT PROC NEAR ; 八进制显示子程序
 ROL BX, 1 ; 16 位二进制数包含 6 位八进制数,最高位仅 1 位
 MOV DX, BX
 AND DX, 1
 OR DL, 30H ; 转换为 ASCII 码
 MOV AH, 2
 INT 21H
 MOV CX, 5 ; 余下还有 5 位八进制数
NEXT: PUSH CX
 MOV CL, 3 ; 1 位八进制数包含 3 位二进制数
 ROL BX, CL
 MOV DX, BX
 AND DX, 07H
 OR DL, 30H ; 转换为 ASCII 码
 MOV AH, 2
 INT 21H
 POP CX
 LOOP NEXT
 RET
OUTOCT ENDP ; OUTOCT 子程序结束
; -----
DISP_CRLF PROC NEAR ; 显示回车换行符子程序
 LEA DX, CRLF
 MOV AH, 09H
 INT 21H
 RET
DISP_CRLF ENDP ; DISP_CRLF 子程序结束
; -----
CSEG ENDS ; 以上定义代码段
; *****
 END START

```

6.11 假定一个名为 MAINPRO 的程序要调用子程序 SUBPRO，试问：

(1) MAINPRO 中的什么指令告诉汇编程序 SUBPRO 是在外部定义的?

(2) SUBPRO 怎么知道 MAINPRO 要调用它?

答: (1) EXTRN SUBPRO:FAR

(2) PUBLIC SUBPRO

6.12 假定程序 MAINPRO 和 SUBPRO 不在同一模块中, MAINPRO 中定义字节变量 QTY 和字变量 VALUE 和 PRICE。SUBPRO 程序要把 VALUE 除以 QTY, 并把商存在 PRICE 中。试问:

(1) MAINPRO 怎么告诉汇编程序外部子程序要调用这三个变量?

(2) SUBPRO 怎么告诉汇编程序这三个变量是在另一个汇编语言程序定义的?

答: (1) PUBLIC QTY, VALUE, PRICE

(2) EXTRN QTY:BYTE, VALUE:WORD, PRICE:WORD

6.13 假设:

(1) 在模块 1 中定义了双字变量 VAR1, 首地址为 VAR2 的字节数据和 NEAR 标号 LAB1, 它们将由模块 2 和模块 3 所使用;

(2) 在模块 2 中定义了字变量 VAR3 和 FAR 标号 LAB2, 而模块 1 中要用到 VAR3, 模块 3 中要用到 LAB2;

(3) 在模块 3 中定义了 FAR 标号 LAB3, 而模块 2 中要用到它。

试对每个源模块给出必要的 EXTRN 和 PUBLIC 说明。

答: 模块 1:

```
EXTRN VAR3: WORD
```

```
PUBLIC VAR1, VAR2, LAB1
```

模块 2:

```
EXTRN VAR1: DWORD, VAR2: BYTE, LAB1: NEAR, LAB3: FAR
```

```
PUBLIC VAR3, LAB2
```

模块 3:

```
EXTRN VAR1: DWORD, VAR2: BYTE, LAB1: NEAR, LAB2: FAR
```

```
PUBLIC LAB3
```

6.14 主程序 CALLMUL 定义堆栈段、数据段和代码段, 并把段寄存器初始化, 数据段中定义变量 QTY 和 PRICE; 代码段中将 PRICE 装入 AX, QTY 装入 BX, 然后调用子程序 SUBMUL。程序 SUBMUL 没有定义任何数据, 它只简单地把 AX 中的内容(PRICE)乘以 BX 中的内容(QTY), 乘积放在 DX: AX 中。请编制这两个要连接起来的程序。

答: 程序如下:

```
TITLE CALLMUL ; 主程序
EXTRN SUBMUL: FAR
; -----
STACK SEGMENT PARA STACK 'STACK'
 DW 64 DUP (?)
 TOS LABEL WORD
STACK ENDS
; -----
DATASG SEGMENT PARA 'DATA'
 QTY DW 0140H
 PRICE DW 2500H
DATASG ENDS
; -----
CODESG SEGMENT PARA 'CODE'
CALLMUL PROC FAR
 ASSUME CS: CODESG, DS: DATASG, SS: STACK
START: MOV AX, STACK
 MOV SS, AX ; 给 SS 赋值
 MOV SP, OFFSET TOS ; 给 SP 赋值
 PUSH DS
 SUB AX, AX
 POP AX
 MOV AX, DATASG
```

```

 MOV DS, AX

 MOV AX, PRICE
 MOV BX, QTY
 CALL SUBMUL
 RET
 CALLMUL ENDP
CODESG ENDS
; -----
 END CALLMUL
; *****
TITLE SUBMUL ; 子程序
PUBLIC SUBMUL
; -----
CODESG1 SEGMENT PARA 'CODE'
 ASSUME CS: CODESG1
 SUBMUL PROC FAR
 ASSUME CS: CODESG1
 MUL BX
 RET
 SUBMUL ENDP
CODESG1 ENDS
; -----
 END

```

#### 6.15 试编写一个执行以下计算的子程序 COMPUTE:

$$R \leftarrow X + Y - 3$$

其中 X, Y 及 R 均为字数组。假设 COMPUTE 与其调用程序都在同一代码段中, 数据段 D\_SEG 中包含 X 和 Y 数组, 数据段 E\_SEG 中包含 R 数组, 同时写出主程序调用 COMPUTE 过程的部分。

如果主程序和 COMPUTE 在同一程序模块中, 但不在同一代码段中, 程序应如何修改?

如果主程序和 COMPUTE 不在同一程序模块中, 程序应如何修改?

答: (1) 主程序和 COMPUTE 在同一代码段中的程序如下:

```

TITLE ADDITION ; 主程序
; -----
D_SEG SEGMENT PARA 'DATA'
 COUNT EQU 10H
 X DW COUNT DUP (?)
 Y DW COUNT DUP (?)
D_SEG ENDS
; -----
E_SEG SEGMENT PARA 'DATA'
 R DW COUNT DUP (?)
E_SEG ENDS
; -----
C_SEG SEGMENT PARA 'CODE'
 ADDITION PROC FAR
 ASSUME CS: C_SEG, DS: D_SEG, ES: E_SEG
 START: PUSH DS
 SUB AX, AX
 PUSH AX
 MOV AX, D_SEG
 MOV DS, AX
 MOV AX, E_SEG
 MOV ES, AX
 CALL COMPUTE ; 调用求和子程序
 RET
 ADDITION ENDP
; *****

```

```

 COMPUTE PROC NEAR ; 同一段的求和子程序
 MOV CX, COUNT
 MOV BX, 0
 REPEAT: MOV AX, X[BX]
 ADD AX, Y[BX]
 SUB AX, 3
 MOV ES: R[BX], AX
 RET
 COMPUTE ENDP
; -----
C_SEG ENDS
; *****
 END START
(2) 主程序和 COMPUTE 在同一程序模块中，但不在同一代码段中的程序如下：
 TITLE ADDITION ; 主程序
; -----
D_SEG SEGMENT PARA 'DATA'
 COUNT EQU 10H
 X DW COUNT DUP (?)
 Y DW COUNT DUP (?)
D_SEG ENDS
; -----
E_SEG SEGMENT PARA 'DATA'
 R DW COUNT DUP (?)
E_SEG ENDS
; -----
C_SEG SEGMENT PARA 'CODE'
 ADDITION PROC FAR
 ASSUME CS: C_SEG, DS: D_SEG, ES: E_SEG
 START: PUSH DS
 SUB AX, AX
 POP AX
 MOV AX, D_SEG
 MOV DS, AX
 MOV AX, E_SEG
 MOV ES, AX
 CALL FAR PTR COMPUTE ; 调用求和子程序
 RET
 ADDITION ENDP
C_SEG ENDS
; *****
CODESG SEGMENT PARA 'CODE'
 ASSUME CS: CODESG
 COMPUTE PROC FAR ; 不同段的求和子程序
 MOV CX, COUNT
 MOV BX, 0
 REPEAT: MOV AX, X[BX]
 ADD AX, Y[BX]
 SUB AX, 3
 MOV ES: R[BX], AX
 RET
 COMPUTE ENDP
; -----
CODESG ENDS
; *****
 END START
(3) 主程序和 COMPUTE 不在同一程序模块中的程序如下：
 TITLE ADDITION ; 主程序

```

```

EXTRN COMPUTE: FAR
PUBLIC COUNT, X, Y, R
; -----
D_SEG SEGMENT PARA 'DATA'
COUNT DW 10H
X DW 10H DUP (?)
Y DW 10H DUP (?)
D_SEG ENDS
; -----
E_SEG SEGMENT PARA 'DATA'
R DW 10H DUP (?)
E_SEG ENDS
; -----
C_SEG SEGMENT PARA 'CODE'
ADDITION PROC FAR
 ASSUME CS: C_SEG, DS: D_SEG, ES: E_SEG
START: PUSH DS
 SUB AX, AX
 POP AX
 MOV AX, D_SEG
 MOV DS, AX
 MOV AX, E_SEG
 MOV ES, AX
 CALL FAR PTR COMPUTE ; 调用求和子程序
 RET
ADDITION ENDP
C_SEG ENDS
; -----
 END START
; *****

TITLE COMPUTE ; 求和子程序
EXTRN COUNT:WORD, X:WORD, Y:WORD, R:WORD
PUBLIC COMPUTE
; -----
CODESG SEGMENT PARA 'CODE'
 ASSUME CS: CODESG
COMPUTE PROC FAR ; 不同模块的求和子程序
 MOV CX, COUNT
 MOV BX, 0
REPEAT: MOV AX, X[BX]
 ADD AX, Y[BX]
 SUB AX, 3
 MOV ES: R[BX], AX
 RET
COMPUTE ENDP
; -----
CODESG ENDS
; *****
 END

```

## 第七章. 习 题

**7.1** 编写一条宏指令 CLRB, 完成用空格符将一字符区中的字符取代的工作。字符区首地址及其长度为变元。

答: 宏定义如下:

```
CLRB MACRO N, CFIL
```



```

MOV CX, N
CLD
MOV AL, ' ' ;; 取空格符的 ASCII 码
LEA DI, CFIL
REP STOSB
ENDM

```

- 7.2 某工厂计算周工资的方法是每小时的工资率 **RATE** 乘以工作时间 **HOURL**, 另外每工作满 10 小时加奖金 3 元, 工资总数存放在 **WAG** 中。请将周工资的计算编写成一条宏指令 **WAGES**, 并展开宏调用:

**WAGES R1, 42, SUM**

答: 宏定义如下:

```

WAGES MACRO RATE, HOUR, WAG
MOV AL, HOUR ;; 计算周工资(WAG), 公式为: HOUR* RATE
MOV BL, RATE
MUL BL
MOV WAG, AX
MOV AL, HOUR ;; 计算奖金存入(AX), 公式为: HOUR/10 的商*3
MOV AH, 0
MOV BL, 10
DIV BL
MOV BL, 3
MUL BL
ADD WAG, AX ;; 计算周工资总数
ENDM

```

宏调用:

**WAGES R1, 42, SUM**

宏展开:

```

1 MOV AL, 42
1 MOV BL, R1
1 MUL BL
1 MOV SUM, AX
1 MOV AL, 42
1 MOV AH, 0
1 MOV BL, 10
1 DIV BL
1 MOV BL, 3
1 MUL BL
1 ADD SUM, AX

```

- 7.3 给定宏定义如下: (注意: 此宏指令的功能是  $V3 \leftarrow |V1 - V2|$ )

```

DIF MACRO X, Y
MOV AX, X
SUB AX, Y
ENDM

ABSDIF MACRO V1, V2, V3
LOCAL CONT
PUSH AX
DIF V1, V2
CMP AX, 0
JGE CONT
NEG AX
CONT: MOV V3, AX
POP AX
ENDM

```

试展开以下调用, 并判定调用是否有效。

- (1) **ABSDIF P1, P2, DISTANCE**
- (2) **ABSDIF [BX], [SI], X[DI], CX**
- (3) **ABSDIF [BX][SI], X[BX][SI], 240H**

## (4) ABSDIF AX, AX, AX

答：(1) 宏调用 ABSDIF P1, P2, DISTANCE 的宏展开如下：此宏调用有效。

```

1 PUSH AX
1 DIF P1, P2
1 MOV AX, P1
1 SUB AX, P2
1 CMP AX, 0
1 JGE ??0000
1 NEG AX
1 ??0000: MOV DISTANCE, AX
1 POP AX

```

(2) 宏调用 ABSDIF [BX], [SI], X[DI], CX 的宏展开如下：此宏调用有效。

```

1 PUSH AX
1 DIF [BX], [SI]
1 MOV AX, [BX]
1 SUB AX, [SI]
1 CMP AX, 0
1 JGE ??0001
1 NEG AX
1 ??0001: MOV X[DI], AX
1 POP AX

```

(3) 宏调用 ABSDIF [BX][SI], X[BX][SI], 240H 的宏展开如下：此宏调用无效。

```

1 PUSH AX
1 DIF [BX][SI], X[BX][SI]
1 MOV AX, [BX][SI]
1 SUB AX, X[BX][SI]
1 CMP AX, 0
1 JGE ??0002
1 NEG AX
1 ??0002: MOV 240H, AX
1 POP AX

```

(4) 宏调用 ABSDIF AX, AX, AX 的宏展开如下：此宏调用有效但无多大意义。

```

1 PUSH AX
1 DIF AX, AX
1 MOV AX, AX
1 SUB AX, AX
1 CMP AX, 0
1 JGE ??0003
1 NEG AX
1 ??0003: MOV AX, AX
1 POP AX

```

7.4 试编制宏定义，要求把存储器中的一个用 EOT（ASCII 码 04H）字符结尾的字符串传送到另一个存储区去。

答：宏定义如下：

```

SEND MACRO SCHARS, DCHARS
 LOCAL NEXT, EXIT
 PUSH AX
 PUSH SI
 MOV SI, 0
NEXT: MOV AL, SCHARS[SI]
 MOV DCHARS[SI], AL
 CMP AL, 04H ;; 是 EOT 字符吗?
 JZ EXIT
 INC SI
 JMP NEXT
EXIT: POP SI
 POP AX
 ENDM

```

## 7.5 宏指令 BIN\_SUB 完成多个字节数据连减的功能:

RESULT ← (A-B-C-D-...)

要相减的字节数据顺序存放在首地址为 OPERAND 的数据区中, 减数的个数存放在 COUNT 单元中, 最后结果存入 RESULT 单元。请编写此宏指令。

答: 宏定义如下:

```

BIN_SUB MACRO RESULT, A, OPERAND, COUNT
 LOCAL NEXT_SUB
 PUSH CX
 PUSH BX
 PUSH AX
 MOV CX, COUNT
 MOV AL, A
 LEA BX, OPERAND
 CLC
NEXT_SUB: SBB AL, [BX]
 INC BX
 LOOP NEXT_SUB
 MOV RESULT, AL
 POP AX
 POP BX
 POP CX
 ENDM

```

## 7.6 请用宏指令定义一个可显示字符串 GOOD: 'GOOD STUDENTS: CLASSX NAME', 其中 X 和 NAME 在宏调用时给出。

答: 宏定义如下:

```

DISP_GOOD MACRO X, NAME
GOOD DB 'GOOD STUDENTS: CLASS&X &NAME', 0DH, 0AH, '$'
 ENDM

```

## 7.7 下面的宏指令 CNT 和 INC1 完成相继字存储。

```

CNT MACRO A, B
 A&B DW ?
 ENDM
INC1 MACRO A, B
 CNT A, %B
 B=B+1
 ENDM

```

请展开下列宏调用:

C=0

```

 INC1 DATA, C
 INC1 DATA, C

```

答: 宏展开如下:

```

C=0
 INC1 DATA, C
1 DATA0 DW ?
 INC1 DATA, C
1 DATA0 DW ? (注意: C 为 0 没有变)

```

## 7.8 定义宏指令并展开宏调用。宏指令 JOE 把一串信息 'MESSAGE NO. K' 存入数据存储区 XK 中。

宏调用为:

I=0

```

 JOE TEXT, I
 ⋮
 JOE TEXT, I
 ⋮
 JOE TEXT, I
 ⋮

```

答: 宏定义如下:

```

MARY MACRO X, K
 X&K DB 'MESSAGE NO. &K'
 ENDM
JOE MACRO A, I
 MARY A, %I
 I=I+1
 ENDM

```

宏调用和宏展开：

```

I=0
 JOE TEXT, I
1 TEXT0 DB 'MESSAGE NO. 0'
 ⋮
 JOE TEXT, I
1 TEXT1 DB 'MESSAGE NO. 1'
 ⋮
 JOE TEXT, I
1 TEXT2 DB 'MESSAGE NO. 2'

```

**7.9** 宏指令 STORE 定义如下：

```

STORE MACRO X, N
 MOV X+I, I
 I=I+1
 IF I-N
 STORE X, N
 ENDF
 ENDM

```

试展开下列宏调用：

```

I=0
 STORE TAB, 7

```

答：宏展开如下：

```

I=0
 STORE TAB, 7
1 MOV TAB+0, 0
1 MOV TAB+1, 1
1 MOV TAB+2, 2
1 MOV TAB+3, 3
1 MOV TAB+4, 4
1 MOV TAB+5, 5
1 MOV TAB+6, 6

```

**7.10** 试编写非递归的宏指令，使其完成的工作与 7.9 题的 STORE 相同。

答：宏定义如下：

```

STORE MACRO K
 MOV TAB+K, K
 ENDM

```

宏调用：

```

I=0
 REPT 7
 STORE %I
 I=I+1
 ENDM

```

**7.11** 试编写一段程序完成以下功能，如给定名为 X 的字符串长度大于 5 时，下列指令将汇编 10 次。

```

 ADD AX, AX
答：程序段如下：
X DB 'ABCDEFGF'
 IF ($-X) GT 5
 REPT 10
 ADD AX, AX
 ENDM

```

ENDIF

**7.12** 定义宏指令 FINSUM: 比较两个数 X 和 Y(X、Y 为数, 而不是地址), 若  $X > Y$  则执行  $SUM \leftarrow X + 2 * Y$ ; 否则执行  $SUM \leftarrow 2 * X + Y$ 。

答: 宏定义如下:

```

CALCULATE MACRO A, B, RESULT ;; 计算 $RESULT \leftarrow 2 * A + B$
 MOV AX, A
 SHL AX, 1
 ADD AX, B
 MOV RESULT, AX
ENDM

FINSUM MACRO X, Y, SUM
 IF X GT Y
 CALCULATE Y, X, SUM
 ELSE
 CALCULATE X, Y, SUM
 ENDIF
ENDM

```

**7.13** 试编写一段程序完成以下功能: 如变元  $X = 'VT55'$ , 则汇编 `MOV TERMINAL, 0`; 否则汇编 `MOV TERMINAL, 1`。

答: 宏定义如下:

```

BRANCH MACRO X
 IFIDN <X>, <VT55>
 MOV TERMINAL, 0
 ELSE
 MOV TERMINAL, 1
 ENDIF
ENDM

```

**7.14** 对于 DOS 功能调用, 所有的功能调用都需要在 AH 寄存器中存放功能码, 而其中有一些功能需要在 DX 中放一个值。试定义宏指令 DOS21, 要求只有在程序中定义了缓冲区时, 汇编为:

```

MOV AH, DOSFUNC
MOV DX, OFFSET BUFF
INT 21H

```

否则, 无 `MOV DX, OFFSET BUFF` 指令。并展开以下宏调用:

```

DOS21 01
DOS21 0AH, IPFIELD

```

答: 宏定义如下:

```

DOS21 MACRO DOSFUNC, BUFF
 MOV AH, DOSFUNC
 IFDEF BUFF
 MOV DX, OFFSET BUFF
 ENDIF
 INT 21H
ENDM

```

宏展开:

```

DOS21 01
1 MOV AH, 01
1 INT 21H
DOS21 0AH, IPFIELD
1 MOV AH, 0AH
1 MOV DX, OFFSET IPFIELD
1 INT 21H

```

**7.15** 编写一段程序, 使汇编程序根据 SIGN 中的内容分别产生不同的指令。如果  $(SIGN) = 0$ , 则用字节变量 DIVD 中的无符号数除以字节变量 SCALE; 如果  $(SIGN) = 1$ , 则用字节变量 DIVD 中的带符号数除以字节变量 SCALE, 结果都存放在字节变量 RESULT 中。

答: 程序段如下:

```

MOV AL, DIVD

```

```

 IF SIGN
 MOV AH, 0
 DIV SCALE
 ELSE
 CBW
 IDIV SCALE
 ENDIF
 MOV RESULT, AL

```

7.16 试编写宏定义 SUMMING，要求求出双字数组中所有元素之和，并把结果保存下来。该宏定义的哑元应为数组首址 ARRAY，数组长度 COUNT 和结果存放单元 RESULT。

答：宏定义如下：

```

SUMMING MACRO ARRAY, COUNT, RESULT
LOCAL ADDITION
MOV ESI, 0
MOV ECX, COUNT
ADDITION: MOV EAX, ARRAY[ESI*4] ;; 双字为 4 字节
 ADD RESULT, EAX
 ADC RESULT+4, 0 ;; 将进位加到结果的高位双字中
 INC ESI
 LOOP ADDITION
ENDM

```

7.17 为下列数据段中的数组编制一程序，调用题 7.16 的宏定义 SUMMING，求出该数组中各元素之和。

```

DATA DD 101246, 274365, 843250, 475536
SUM DQ ?

```

答：程序如下：

```

SUMMING MACRO ARRAY, COUNT, RESULT
LOCAL ADDITION
MOV ESI, 0
MOV ECX, COUNT
ADDITION: MOV EAX, ARRAY[ESI*4] ;; 双字为 4 字节
 ADD RESULT, EAX
 ADC RESULT+4, 0 ;; 将进位加到结果的高位双字中
 INC ESI
 LOOP ADDITION
ENDM

.MODEL SMALL
.386
.DATA
DATA DD 101246, 274365, 843250, 475536
SUM DQ ?
.CODE
START: MOV AX, @DATA
 MOV DS, AX
 SUMMING DATA, 4, SUM
 MOV AX, 4C00H
 INT 21H
 END START

```

7.18 如把题 7.16 中的宏定义存放在一个宏库中，则题 7.17 的程序应如何修改？

答：程序修改如下：

```

INCLUDE MACRO.MAC ; 假设存放的宏库名为 MACRO.MAC
.MODEL SMALL
.386
.DATA
DATA DD 101246, 274365, 843250, 475536
SUM DQ ?

```



```

BEGIN: IN AL, 24H
 TEST AL, 08H ; 查询第一个设备的输入是否结束?
 JNZ EXIT
 TEST AL, 01H ; 查询第一个设备的输入是否准备好?
 JZ BEGIN1
 IN AL, 26H ; 输入数据并存入缓冲区 BUFF1
 MOV BUFF1[DI], AL
 INC DI
BEGIN1: IN AL, 36H
 TEST AL, 08H ; 查询第二个设备的输入是否结束
 JNZ EXIT
 TEST AL, 01H ; 查询第二个设备的输入是否准备好?
 JZ BEGIN
 IN AL, 38H ; 输入数据并存入缓冲区 BUFF2
 MOV BUFF2[SI], AL
 INC SI
 JMP BEGIN
EXIT: ;

```

- 8.7 假定外部设备有一台硬币兑换器，其状态寄存器的端口地址为 0006H，数据输入寄存器的端口地址为 0005H，数据输出寄存器的端口地址为 0007H。试用查询方式编制一程序，该程序作空闲循环等待纸币输入，当状态寄存器第 2 位为 1 时，表示有纸币输入，此时可从数据输入寄存器输入的代码中测出纸币的品种，一角纸币的代码为 01，二角纸币为 02，五角纸币则为 03。然后程序在等待状态寄存器的第 3 位变为 1 后，把应兑换的五分硬币数(用 16 进制表示)从数据输出寄存器输出。

答：程序段如下：

```

BEGIN: IN AL, 06H ; 查询是否有纸币输入?
 TEST AL, 04H
 JZ BEGIN
 IN AL, 05H ; 测试纸币的品种
 CMP AL, 01H ; 是一角纸币吗?
 JNE NEXT1
 MOV AH, 02 ; 是一角纸币，输出 2 个 5 分硬币
 JMP NEXT
NEXT1: CMP AL, 02H ; 是二角纸币吗?
 JNE NEXT2
 MOV AH, 04 ; 是二角纸币，输出 4 个 5 分硬币
 JMP NEXT
NEXT2: CMP AL, 03H ; 是五角纸币吗?
 JNE BEGIN
 MOV AH, 10 ; 是五角纸币，输出 10 个 5 分硬币
NEXT: IN AL, 06H ; 查询是否允许输出 5 分硬币?
 TEST AL, 08H
 JZ NEXT
 MOV AL, AH ; 输出 5 分硬币
 OUT 07H, AL
 JMP BEGIN

```

- 8.8 给定 (SP)=0100H，(SS)=0300H，(FLAGS)=0240H，以下存储单元的内容为 (00020)=0040H，(00022)=0100H，在段地址为 0900 及偏移地址为 00A0H 的单元中有一条中断指令 INT 8，试问执行 INT 8 指令后，SP，SS，IP，FLAGS 的内容是什么？栈顶的三个字是什么？

答：执行 INT 8 指令后，(SP)=00FAH，(SS)=0300H，(CS)=0100H，(IP)=0040H，(FLAGS)=0040H  
栈顶的三个字是：原(IP)=00A2H，原(CS)=0900H，原(FLAGS)=0240H

- 8.9 类型 14H 的中断向量在存储器的哪些单元里？

答：在 0000:0050H，0000:0051H，0000:0052H，0000:0053H 四个字节中。

- 8.10 假定中断类型 9H 的中断处理程序的首地址为 INT\_ROUT，试写出主程序中为建立这一中断向量



而编制的程序段。

答：程序段如下：

```

 ⋮
MOV AL, 1CH ; 取原中断向量，并保护起来
MOV AH, 35H
INT 21H
PUSH ES
PUSH BX
PUSH DS
MOV AX, SEG INT_ROUT
MOV DS, AX
MOV DX, OFFSET INT_ROUT
MOV AL, 09H
MOV AH, 25H ; 设置中断向量功能调用
INT 21H
POP DS
 ⋮
POP DX ; 还原原中断向量
POP DS
MOV AL, 1CH
MOV AH, 25H
INT 21H

```

**8.11** 编写指令序列，使类型 1CH 的中断向量指向中断处理程序 SHOW\_CLOCK。

答：程序段如下：

```

 ⋮
MOV AL, 1CH
MOV AH, 35H ; 取中断向量功能调用，取原中断向量
INT 21H
PUSH ES
PUSH BX
PUSH DS
MOV AX, SEG SHOW_CLOCK
MOV DS, AX
MOV DX, OFFSET SHOW_CLOCK
MOV AL, 1CH
MOV AH, 25H ; 设置中断向量功能调用
INT 21H
POP DS
 ⋮
POP DX
POP DS
MOV AL, 1CH
MOV AH, 25H ; 设置中断向量功能调用，还原原中断向量
INT 21H
 ⋮

```

8.12 如设备 D1, D2, D3, D4, D5 是按优先级次序排列的，设备 D1 的优先级最高。而中断请求的次序如下所示，试给出各设备的中断处理程序的运行次序。假设所有的中断处理程序开始后就有 STI 指令。

- (1) 设备 D3 和 D4 同时发出中断请求。
- (2) 在设备 D3 的中断处理程序完成之前，设备 D2 发出中断请求。
- (3) 在设备 D4 的中断处理程序未发出中断结束命令(EOI)之前，设备 D5 发出中断请求。
- (4) 以上所有中断处理程序完成并返回主程序，设备 D1, D3, D5 同时发出中断请求。

答：各设备的中断处理程序的运行次序是：INT\_D3, INT\_D2 嵌套 INT\_D3, INT\_D4, INT\_D5; INT\_D1, INT\_D3, INT\_D5。

8.13 在 8.12 题中假设所有的中断处理程序中都没有 STI 指令，而它们的 IRET 指令都可以由于 FLAGS 出栈而使 IF 置 1，则各设备的中断处理程序的运行次序应是怎样的？

答：各设备的中断处理程序的运行次序是：INT\_D3, INT\_D2, INT\_D4, INT\_D5;  
INT\_D1, INT\_D3, INT\_D5。

8.14 试编制一程序，要求测出任一程序的运行时间，并把结果打印出来。

答：程序段如下：

```

TITLE TEST_TIME.EXE ; 测试程序运行时间程序
; *****
DSEG SEGMENT ; 定义数据段
COUNT DW 0 ; 记录系统时钟(18.2 次中断/秒)的中断次数
SEC DW 0 ; 存放秒钟数
MIN DW 0 ; 存放分钟数
HOURS DW 0 ; 存放小时数
PRINTTIME DB 0DH, 0AH, 'The time of exection program is:'
CHAR_NO EQU $- PRINTTIME
DSEG ENDS ; 以上定义数据段
; *****
CSEG SEGMENT ; 定义代码段
MAIN PROC FAR
ASSUME CS: CSEG, DS: DSEG

START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

 MOV AL, 1CH ; 取原来的 1CH 中断向量
 MOV AH, 35H
 INT 21H
 PUSH ES ; 保存原来的 1CH 中断向量
 PUSH BX

 PUSH DS ; 设置新的 1CH 中断向量
 MOV AX, SEG CLINT
 MOV DS, AX
 MOV DX, OFFSET CLINT
 MOV AL, 1CH
 MOV AH, 25H
 INT 21H
 POP DS

 IN AL, 21H ; 清除时间中断屏蔽位并开中断
 AND AL, 0FEH
 OUT 21H, AL
 STI

 ; ; 要求测试时间的程序段

 POP DX ; 恢复原来的 1CH 中断向量
 POP DS
 MOV AL, 1CH
 MOV AH, 25H
 INT 21H

 CALL PRINT ; 打印输出测试时间
 RET ; 返回 DOS

```

```

MAIN ENDP
; -----
CLINT PROC NEAR ; 中断服务子程序
 PUSH DS
 PUSH BX
 MOV BX, SEG COUNT
 MOV DS, BX
 LEA BX, COUNT
 INC WORD PTR [BX] ; 记录系统时钟的中断次数单元+1
 CMP WORD PTR [BX],18 ; 有 1 秒钟吗?
 JNE TIMEOK
 CALL INCTEST ; 有 1 秒钟, 转去修改时间
ADJ: CMP HOURS, 12 ; 有 12 小时吗?
 JLE TIMEOK
 SUB HOURS, 12 ; 有 12 小时, 将小时数减去 12
TIMEOK: MOV AL, 20H ; 发中断结束命令
 OUT 20H, AL
 POP BX
 POP DS
 IRET
CLINT ENDP ; CLINT 中断服务子程序结束
; -----
INCTEST PROC NEAR ; 修改时间子程序
 MOV WORD PTR [BX], 0 ; 中断次数单元或秒单元或分单元清 0
 ADD BX, 2
 INC WORD PTR [BX] ; 秒单元或分单元或时单元+1
 CMP WORD PTR [BX],60 ; 有 60 秒或 60 分吗?
 JLE RETURN
 CALL INCTEST ; 先修改秒单元, 再修改分单元, 再修改时单元
RETURN: RET
INCTEST ENDP ; INCTEST 子程序结束
; -----
PRINT PROC NEAR ; 打印输出子程序
 LEA BX, PRINTTIME ; 打印输出 PRINTTIME 信息
ROTATE: MOV CX, CHAR_NO
 MOV DL, [BX]
 MOV AH, 05H
 INT 21H
 INC BX
 LOOP ROTATE
 MOV BX, HOURS ; 打印时间的小时数
 CALL BINIDEC ; 调二进制转换为 10 进制并打印输出子程序
 MOV DL, ':' ; 打印输出冒号 ':'
 MOV AH, 05H
 INT 21H
 MOV BX, MIN ; 打印时间的分钟数
 CALL BINIDEC
 MOV DL, ':'
 MOV AH, 05H
 INT 21H
 MOV BX, SEC ; 打印时间的秒钟数
 CALL BINIDEC
 RET
PRINT ENDP ; PRINT 子程序结束
; -----
BINIDEC PROC NEAR ; 二进制转换为 10 进制子程序

```

```

 MOV CX, 10000D
 CALL DEC_DIV ; 调除法并打印输出子程序
 MOV CX, 1000D
 CALL DEC_DIV
 MOV CX, 100D
 CALL DEC_DIV
 MOV CX, 10D
 CALL DEC_DIV
 MOV CX, 1D
 CALL DEC_DIV
 RET
BINIDEC ENDP ; BINIDEC 子程序结束
; -----
DEC_DIV PROC NEAR ; 除法并打印输出子程序
 MOV AX, BX
 MOV DX, 0
 DIV CX
 MOV BX, DX ; 余数保存在(BX)中作下一次的除法
 MOV DL, AL ; 商(在 00H~09H 范围内)送(DL)
 ADD DL, 30H ; 转换为 0~9 的 ASCII 码
 MOV AH, 05H ; 打印输出
 INT 21H
 RET
DEC_DIV ENDP ; DEC_DIV 子程序结束
; -----
CSEG ENDS ; 以上定义代码段
; *****
 END START ; 汇编语言源程序结束

```

## 第九章. 习 题

9.1 INT 21H 的键盘输入功能 1 和功能 8 有什么区别?

答: 键盘输入功能 1: 输入字符并回显(回送显示器显示) (检测 Ctrl\_Break);

键盘输入功能 8: 输入字符但不回显(也检测 Ctrl\_Break)。

9.2 编写一个程序, 接受从键盘输入的 10 个十进制数字, 输入回车符则停止输入, 然后将这些数字加密后(用 XLAT 指令变换)存入内存缓冲区 BUFFER。加密表为:

输入数字: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

密码数字: 7, 5, 9, 1, 3, 6, 8, 0, 2, 4

答: 程序段如下:

```

SCODE DB 7, 5, 9, 1, 3, 6, 8, 0, 2, 4 ; 密码数字
BUFFER DB 10 DUP (?)
;
 MOV SI, 0
 MOV CX, 10
 LEA BX, SCODE
INPUT: MOV AH, 1 ; 从键盘输入一个字符的功能调用
 INT 21H
 CMP AL, 0DH ; 输入回车符则停止输入
 JZ EXIT
 SUB AL, 30H ; 是 0~9 吗?
 JB INPUT
 CMP AL, 09H
 JA INPUT
 XLAT
 MOV BUFFER[SI], AL ; 换为密码
 ; 保存密码

```

```

 INC SI
 LOOP INPUT
EXIT: RET

```

9.3 对应黑白显示器屏幕上 40 列最下边一个像素的存储单元地址是什么？

答：对应黑白显示器屏幕上 40 列最下边一个像素的存储单元地址是：B000:0F78H

9.4 写出把光标置在第 12 行，第 8 列的指令。

答：指令如下：

```

 MOV DH, 0BH ; 0BH=12-1
 MOV DL, 07H ; 07H=8-1
 MOV BH, 0
 MOV AH, 2 ; 置光标功能调用
 INT 10H

```

9.5 编写指令把 12 行 0 列到 22 行 79 列的屏幕清除。

答：指令如下：

```

 MOV AL, 0 ; 清除屏幕
 MOV BH, 07
 MOV CH, 12 ; 左上角行号
 MOV CL, 0 ; 左上角列号
 MOV DH, 22 ; 右下角行号
 MOV DL, 79 ; 右下角列号
 MOV AH, 6 ; 屏幕上滚功能调用
 INT 10H

```

9.6 编写指令使其完成下列要求。

(1) 读当前光标位置

(2) 把光标移至屏底一行的开始

(3) 在屏幕的左上角以正常属性显示一个字母 M

答：指令序列如下：

```

(1) MOV AH, 3 ; 读当前光标位置，返回 DH/DL=光标所在的行/列
 MOV BH, 0
 INT 10H

(2) MOV DH, 24 ; 设置光标位置
 MOV DL, 0
 MOV BH, 0
 MOV AH, 2
 INT 10H

(3) MOV AH, 2 ; 设置光标位置
 MOV DX, 0
 MOV BH, 0
 INT 10H
 MOV AH, 9 ; 在当前光标位置显示一个字符
 MOV AL, 'M'
 MOV BH, 0
 MOV BL, 7
 MOV CX, 1
 INT 10H

```

9.7 写一段程序，显示如下格式的信息：

Try again, you have n starfighters left.

其中 n 为 CX 寄存器中的 1~9 之间的二进制数。

答：程序段如下：

```

MESSAGE DB 'Try again, you have '
CONT DB n
 DB ' starfighters left.$'
; ;
 ADD CL, 30H

```

```

MOV CONT, CL ; 保存 ASCII 码
LEA DX, MESSAGE
MOV AH, 9 ; 显示一个字符串的 DOS 调用
INT 21H

```

- 9.8 从键盘上输入一行字符，如果这行字符比前一次输入的一行字符长度长，则保存该行字符，然后继续输入另一行字符；如果它比前一次输入的行短，则不保存这行字符。按下‘\$’输入结束，最后将最长的一行字符显示出来。

答：程序段如下：

```

STRING DB 0 ; 存放字符的个数
 DB 80 DUP (?), 0DH, 0AH, '$' ; 存放前一次输入的字符串，兼作显示缓冲区
BUFFER DB 80 ; 输入字符串的缓冲区，最多输入 80 个字符
 DB ?
 DB 80 DUP (20H)
;
;
INPUT: LEA DX, BUFFER ; 输入字符串
 MOV AH, 0AH ; 输入字符串的 DOS 调用
 INT 21H
 LEA SI, BUFFER+1 ; 比较字符串长度
 LES DI, STRING
 MOV AL, [SI]
 CMP AL, [DI]
 JBE NEXT
 MOV CX, 80+1 ; 大于前次输入的字符串，更换前次的字符串
 CLD
 REP MOVSB
NEXT: MOV AH, 1 ; 输入结束符吗？
 INT 21H
 CMP AL, '$' ; 是结束符吗？
 JNE INPUT ; 不是则继续输入
 LEA DX, STRING+1 ; 显示字符串
 MOV AH, 9 ; 显示一个字符串的 DOS 调用
 INT 21H

```

- 9.9 编写程序，让屏幕上显示出信息 “What is the date (mm/dd/yy)?” 并响铃(响铃符为 07)，然后从键盘接收数据，并按要求的格式保存在 date 存储区中。

答：程序段如下：

```

MESSAGE DB 'What is the date (mm/dd/yy)?', 07H, '$'
DATAFLD DB 10, 0
DATE DB 10 DUP (' ')
;
;
 MOV AH, 9 ; 显示一个字符串的 DOS 调用
 LEA DX, MESSAGE ; 显示字符串
 INT 21H
 MOV AH, 0AH ; 输入字符串的 DOS 调用
 LEA DX, DATAFLD
 INT 21H

```

- 9.10 用户从键盘输入一文件并在屏幕上回显出来。每输入一行(≤80 字符)，用户检查一遍，如果用户认为无需修改，则键入回车键，此时这行字符存入 BUFFER 缓冲区保存，同时打印机把这行字符打印出来并回车换行。

答：程序段如下：

```

INAREA DB 80 ; 输入字符串的缓冲区，最多输入 80 个字符
ACTLEN DB ?
BUFFER DB 80 DUP (?)
;
;
INPUT: LEA DX, INAREA ; 输入字符串

```

```

 MOV AH, 0AH ; 输入字符串的 DOS 调用
 INT 21H
 CMP ACTLEN, 0
 JE EXIT
 MOV BX, 0
 MOV CH, 0
 MOV CL, ACTLEN
PRINT: MOV AH, 5 ; 打印输出
 MOV DL, BUFFER[BX]
 INT 21H
 INC BX
 LOOP PRINT
 MOV AH, 5 ; 打印输出回车换行
 MOV DL, 0AH
 INT 21H
 MOV DL, 0DH
 INT 21H
 JMP INPUT
EXIT: RET

```

9.11 使用 MODE 命令，设置 COM2 端口的通信数据格式为：每字 8 位，无校验，1 位终止位，波特率为 1200b/s。

答：命令格式如下：

```
MODE COM2: 12, N, 8, 1
```

## 第十章. 习 题

10.1 写出指令，选择显示方式 10H，并将背景设为绿色。

```

答： MOV AH, 00H
 MOV AL, 10H ; 选择显示方式 10H(16 色图形)
 INT 10H
 MOV AH, 10H
 MOV AL, 00H
 MOV BH, 10H ; 背景设为绿色(02H 也可以，是用 DEBUG 调试出来的)
 MOV BL, 0 ; 选择 0 号调色板
 INT 10H

```

设置背景色也可用：

```

 MOV AH, 0BH ; 设置背景色和调色板
 MOV BH, 0 ; 设置背景色功能
 MOV BL, 8 ; 绿色背景
 INT 10H

```

10.2 如何使用 INT 10H 的功能调用改变显示方式？

答：在 AH 中设置功能号 00H，在 AL 中设置显示方式值，调用 INT 10H 即可。

10.3 VGA 独有的一种显示方式是什么？

答：像素值为 640×480，可同时显示 16 种颜色，这种显示方式(12H)是 VGA 独有的。

10.4 对于 EGA 和 VGA 显示适配器，使用显示方式 13H 时(只有 VGA 有)，显示数据存在哪里？

答：显示数据存在显示存储器里。

10.5 对于 VGA 的显示方式 13H 时存放一屏信息需要多少字节的显存？

答：需要 64000 个字节。

10.6 利用 BIOS 功能编写图形程序：设置图形方式 10H，选择背景色为蓝色，然后每行(水平方向)显示

一种颜色, 每 4 行重复一次, 一直到整个屏幕都显示出彩条。

答: 程序如下:

```

TITLE GRAPHIX.COM
codeseg segment
 assume cs:codeseg, ds:codeseg, ss:codeseg
 org 100h
main proc far
 mov ah, 00h
 mov al, 10h ; 选择显示方式 10h(16 色图形)
 int 10h
 mov ah, 0bh
 mov bh, 00h
 mov bl, 01h ; 背景设为蓝色
 int 10h
 mov ah, 0bh
 mov bh, 01h
 mov bl, 00h ; 设置调色板 0#
 int 10h
 mov bx, 0 ; 显存的第 0 页
 mov cx, 0 ; 起始列号为 0 列
 mov dx, 0 ; 起始行号为 0 行
line: mov ah, 0ch ; 写像素点
 mov al, bl
 int 10h
 inc cx
 cmp cx, 640
 jne line
 mov cx, 0 ; 起始列号为 0 列
 inc bl
 and bl, 03h ; 只显示四种颜色(因此保留最低两位)
 inc dx
 cmp dx, 350
 jne line
 int 20h
main endp
codeseg ends
 end main

```

10.7 修改 10.6 题的程序, 使整个屏幕都显示出纵向的彩条。

答: 程序如下:

```

TITLE GRAPHIX.COM
codeseg segment
 assume cs:codeseg, ds:codeseg, ss:codeseg
 org 100h
main proc far
 mov ah, 00h
 mov al, 10h ; 选择显示方式 10h(16 色图形)
 int 10h
 mov ah, 0bh
 mov bh, 00h
 mov bl, 01h ; 背景设为蓝色
 int 10h
 mov ah, 0bh
 mov bh, 01h
 mov bl, 00h ; 设置调色板 0#
 int 10h
 mov bx, 0 ; 显存的第 0 页
 mov cx, 0 ; 起始列号为 0 列

```



```

 mov dx, 0 ; 起始行号为 0 行
line: mov ah, 0ch ; 写像素点
 mov al, bl
 int 10h
 inc dx
 cmp dx, 350
 jne line
 mov dx, 0 ; 起始行号为 0 行
 inc bl
 and bl, 03h ; 只显示四种颜色(因此保留最低两位)
 inc cx
 cmp cx, 640
 jne line
 int 20h
main endp
codeseg ends
end main

```

10.8 按动键盘上的光标控制键，在屏幕上下左右任一方向上绘图，每画一点之前，由数字键 0~3 指定该点的颜色值，按动 ESC 键，绘图结束，返回 DOS。

答：程序如下：

```

; DRAW—Program to draw on screen with sursor arrows
; For 640*350 color mode
up equ 48h ; 向上键的扫描值
down equ 50h ; 向下键的扫描值
left equ 4bh ; 向左键的扫描值
right equ 4dh ; 向右键的扫描值
escape equ 1bh ; “Esc” character
codeseg segment
main proc far
 assume cs:codeseg
; clear screen by scrolling it, using ROM call
start: mov ah, 06h
 mov al, 00h
 mov cx, 00h
 mov dl, 79
 mov dh, 24
 int 10h
; screen pointer will be in CX, DX registers; row number (0 to 350d) in DX
; coumn number (0 to 640d) in CX
 mov ah, 00h
 mov al, 10h ; 选择显示方式 10h(16 色图形)
 int 10h
 mov ah, 0bh
 mov bh, 00h
 mov bl, 01h ; 背景设为蓝色
 int 10h
 mov ah, 0bh
 mov bh, 01h
 mov bl, 00h ; 设置调色板 0#
 int 10h
 mov dx, 175 ; 设在屏幕中心
 mov cx, 320
; get character from keyboard
get_char: mov ah, 0 ; 键盘输入
 int 16h
 cmp al, escape

```

```

 jz exit
 cmp al, 33h ; > '3' 吗?
 jg plot
 cmp al, 30h ; < '0' 吗?
 jl plot
 mov bl, al ; 是 '0' ~ '3', 设置颜色
 and bl, 03
 jmp get_char
; figure out which way to go, and draw new line
plot: mov al, ah
 cmp al, up
 jnz not_up
 dec dx
not_up: cmp al, down
 jnz not_down
 inc dx
not_down: cmp al, right
 jnz not_right
 inc cx
not_right: cmp al, left
 jnz write
 dec cx
; use ROM routine to write dot, requires row# in DX, col in CX, color in AL
write: mov al, bl
 mov ah, 0ch
 int 10h
 jmp get_char
exit: int 20h
main endp
codeseg ends
end start

```

10.9 位屏蔽寄存器的作用是什么？在 16 色，640×480 显示方式中如何使用位屏蔽寄存器？

答：位屏蔽寄存器的作用是决定了新的像素值产生的方法。当位屏蔽寄存器的某位设为 0 时，相对应的像素值直接由锁存器写入显存；位屏蔽寄存器的某位为 1 时，所对应的像素值由锁存器中的像素值与 CPU 数据或置位/重置寄存器中相应位合并之后产生。

10.10 读映像选择寄存器的作用是什么？如果 4 个位面的内容都需要读取，读映像选择寄存器应如何设置？

答：读映像选择寄存器的作用是用于选择哪一个位面的字节读入 CPU。读映像选择寄存器的 0 和 1 位，用来指定哪个位面的锁存器内容读到 CPU。如果 4 个位面的内容都需要读取，则必须对同一地址执行 4 次读操作，在每次读之前，用指令分别设置读映像选择寄存器。

10.11 编写程序使一只“鸟”飞过屏幕。飞鸟的动作可由小写字母 v (ASCII 码 76H)变为破折号(ASCII 码 0C4H)来模仿，这个字符先后交替在两列显示。鸟的开始位置是 0 列 20 行，每个字符显示 0.5 秒，然后消失。

答：程序段如下：

```

TITLE Flier.EXE ; 飞鸟程序
; *****
DSEG SEGMENT ; 定义数据段
 BIRD DB 76H, 07 ; 小写字母 v 及属性
 DB 0C4H, 07 ; 破折号及属性
DSEG ENDS ; 以上定义数据段
; *****
CSEG SEGMENT ; 定义代码段
 MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
 START: PUSH DS ; 设置返回 DOS

```

```

SUB AX, AX
PUSH AX
MOV AX, DSEG
MOV DS, AX ; 给 DS 赋值

MOV AH, 0FH ; 取当前显示方式
INT 10H
PUSH AX ; 保存当前显示方式(AL)
MOV AH, 0 ; 设置彩色 80×25 文本方式
MOV AL, 3
INT 10H
MOV DH, 20 ; 20 行
MOV DL, 0 ; 0 列
BEGIN: MOV SI, 2 ; 字符 v 和破折号“-”交替显示
MOV CX, 1 ; 一次显示一个字符及属性
LEA DI, BIRD
DISP: CMP DL, 79 ; 飞到 79 列就退出
JAE EXIT
MOV AH, 2 ; 置光标位置
INT 10H
MOV AH, 9 ; 在光标位置显示字符及属性
MOV AL, [DI] ; 取显示字符及属性
MOV BL, [DI+1]
INT 10H
CALL DELAY ; 延时 0.5 秒
MOV AH, 9 ; 在光标位置显示字符及属性
MOV AL, ' ' ; 显示空格，擦除该位置的字符
MOV BL, 7
INT 10H
INC DL ; 飞到下一列
ADD DI, 2
DEC SI
JNZ DISP
JMP BEGIN
EXIT: POP AX ; 恢复当前显示方式(AL)
MOV AH, 0
INT 10H
RET ; 返回 DOS
MAIN ENDP

; -----
DELAY PROC NEAR ; 延时 0.5s 子程序
PUSH CX
PUSH DX
MOV DX, 50 ; 延时 0.5s
DEL1: MOV CX, 2801 ; 延时 10ms
DEL2: LOOP DEL2
DEC DX
JNZ DEL1
POP DX
POP CX
RET
DELAY ENDP ; DELAY 子程序结束
; -----
CSEG ENDS ; 以上定义代码段
; *****
END START ; 汇编语言源程序结束

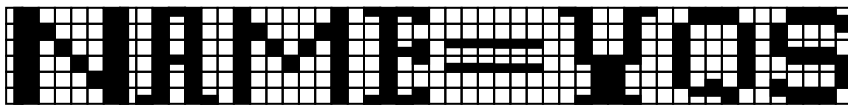
```

10.12 用图形文本的方法设计 “Name=XXX” (X 为你自己姓名的缩写)，并将其数据编码定义在一个数组中。

答：用图形文本的方法设计 “NAME=YQS” 的程序和数组如下：

显示格式如下：

Shooting



ASCII 码为 0DCH

ASCII 码为 0DFH

ASCII 码为 0DDH

ASCII 码为 0DEH

ASCII 码为 0DBH

```

TITLE NAME_YQS.EXE ; 显示 “NAME=YQS” 的程序
; *****
; Graphics block message for the words shooting NAME=YQS
; 00H→end of message, 0FFH→end of screen line
DSEG SEGMENT ; 定义数据段
NAME_YQS DB 2 ; Start row (开始行)
 DB 2 ; Start column (开始列)
 DB 1000 0011B ; Color attribute
 DB 'Shooting',0FFH,0FFH ; 显示 “Shooting”
 DB 7 DUP(0DCH),0FFH,0FFH

; Graphics encoding of the word NAME=YQS using IBM character set
 DB 0DEH, 0DBH, 4 DUP(20H), 0DBH, 0DDH, 20H, 0DBH, 0DFH, 0DBH
 DB 20H, 20H, 0DBH, 5 DUP(20H), 0DBH, 20H, 2 DUP(0DFH, 0DBH)
 DB 8 DUP(20H), 0DFH, 0DBH, 20H, 20H, 0DBH, 0DFH, 20H, 20H
 DB 3 DUP(0DBH), 3 DUP(20H), 3 DUP(0DBH), 0DCH, 0FFH

 DB 0DEH, 0DBH, 0DBH, 3 DUP(20H), 0DBH, 0DDH, 2 DUP(20H, 0DBH)
 DB 20H, 20H, 0DBH, 0DBH, 3 DUP(20H), 0DBH, 0DBH, 20H, 20H, 0DBH
 DB 11 DUP(20H), 3 DUP(0DBH, 20H, 20H), 20H, 0DBH, 20H, 0DBH
 DB 3 DUP(20H), 0DFH, 0FFH

 DB 0DEH, 0DBH, 20H, 0DBH, 20H, 20H, 0DBH, 0DDH, 2 DUP(20H, 0DBH)
 DB 20H, 4 DUP(20H, 0DBH), 20H, 20H, 0DBH, 0DCH, 0DBH, 20H
 DB 7 DUP(0DFH), 3 DUP(20H, 20H, 0DBH), 3 DUP(20H), 0DBH, 20H, 0DBH
 DB 3 DUP(0DCH), 20H, 0FFH

 DB 0DEH, 0DBH, 20H, 20H, 0DBH, 20H, 0DBH, 0DDH, 20H, 0DBH, 0DFH
 DB 0DBH, 4 DUP(20H, 20H, 0DBH), 20H, 0DFH, 20H, 7 DUP(0DCH), 20H
 DB 20H, 0DFH, 0DBH, 0DBH, 0DFH, 20H, 20H, 0DBH, 3 DUP(20H), 0DBH
 DB 20H, 20H, 3 DUP(0DFH), 0DBH, 0FFH

 DB 0DEH, 0DBH, 3 DUP(20H), 0DBH, 0DBH, 0DDH, 2 DUP(20H, 0DBH), 20H
 DB 20H, 0DBH, 5 DUP(20H), 2 DUP(0DBH, 20H, 20H), 10 DUP(20H), 0DBH
 DB 0DBH, 3 DUP(20H), 0DBH, 20H, 0DCH, 20H, 0DBH, 20H, 0DCH
 DB 3 DUP(20H), 0DBH, 0FFH

 DB 0DEH, 0DBH, 4 DUP(20H), 0DBH, 0DDH, 0DCH, 0DBH, 20H, 0DBH
 DB 0DCH, 20H, 0DBH, 5 DUP(20H), 0DBH, 20H, 2 DUP(0DCH, 0DBH)
 DB 9 DUP(20H), 0DCH, 0DBH, 0DBH, 0DCH, 3 DUP(20H), 0DFH, 0DFH
 DB 0DBH, 20H, 20H, 0DFH, 3 DUP(0DBH), 20H, 0FFH
 DB 00 ; 结束显示标志

START_COL DB ?
DSEG ENDS ; 以上定义数据段

```

```

; *****
; Text display procedures: display a message on the graphics screen
CSEG SEGMENT ; 定义代码段
MAIN PROC FAR
 ASSUME CS: CSEG, DS: DSEG
START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX

 LEA DI, NAME_YQS
 MOV DH, [DI] ; Get row into DH
 INC DI ; Bump pointer
 MOV DL, [DI] ; And column into DL
 MOV START_COL, DL ; Store start column
 MOV AH, 2 ; Set cursor position
 MOV BH, 0 ; Page 0
 INT 10H
 INC DI ; Bump pointer to attribute
 MOV BL, [DI] ; Get color code into BL
Char_write: INC DI ; Bump to message start
 MOV AL, [DI] ; Get character
 CMP AL, 0FFH ; End of line?
 JE BUMP_ROW ; Next row
 CMP AL, 0 ; Test for terminator
 JE END_TEXT ; Exit routine
 CALL SHOW_CHAR
 JMP CHAR_WRITE
END_TEXT: RET ; 返回 DOS
Bump_row: INC DH ; Row control register
 MOV DL, START_COL ; Column control to start column
 MOV AH, 2 ; Set cursor position
 MOV BH, 0 ; Page 0
 INT 10H
 JMP CHAR_WRITE
MAIN ENDP

; -----
; Display character in AL and using the color code in BL
Show_char PROC NEAR ; 显示字符子程序
 MOV AH, 9 ; BIOS service request number
 MOV BH, 0 ; Page 0
 MOV CX, 1 ; No repeat
 INT 10H

; Bump cursor
 INC DL
 MOV AH, 2 ; Set cursor position
 MOV BH, 0 ; Page 0
 INT 10H
 RET
Show_char ENDP ; SHOW_CHAR 子程序结束
; -----
CSEG ENDS ; 以上定义代码段
; *****
 END START ; 汇编语言源程序结束

```

10.13 游戏程序常常用随机数来控制其图形在屏幕上移动。请编写一程序,用随机数来控制笑脸符(ASCII 码 02H)显示的位置。笑脸符每次显示的列号总是递增 1。而行的位置可能是前次的上一行,下一行或同一行,这根据随机数是 0、1 或 2 来决定,当行号变为 0、24 或列号变为 79 时显示结束。笑脸在每个位置上显示 0.25s。(提示:INT 1AH 的 AH=0 是读当前时间的功能调用,利用该功能返回的随时都在变化的时间值作为产生随机数的基数。)

答:程序段如下:

```

TITLE Disp_Laugh.EXE ; 笑脸显示程序
; *****
CSEG SEGMENT ; 定义代码段
MAIN PROC FAR
 ASSUME CS:CSEG

START: PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX

 MOV AH, 0FH ; 取当前显示方式
 INT 10H
 PUSH AX ; 保存当前显示方式(AL)
 MOV AH, 0 ; 设置彩色 80×25 文本方式
 MOV AL, 3
 INT 10H

 MOV CX, 1 ; 一次显示一个笑脸字符及属性
 MOV DH, 12H ; 12 行, 从屏幕左边的中间开始
 MOV DL, 0 ; 0 列

BEGIN: CMP DL, 79 ; 移到 79 列就退出
 JAE EXIT
 CMP DH, 0 ; 移到第 0 行就退出
 JBE EXIT
 CMP DH, 24 ; 移到第 24 行就退出
 JAE EXIT
 MOV AH, 2 ; 置光标位置
 INT 10H
 MOV AH, 9 ; 在光标位置显示字符及属性
 MOV AL, 02H ; 取笑字符及属性
 MOV BL, 7
 INT 10H
 CALL DELAY ; 延时 0.25 秒
 MOV AH, 9 ; 在光标位置显示字符及属性
 MOV AL, ' ' ; 显示空格, 擦除该位置的字符
 MOV BL, 7
 INT 10H
 INC DL ; 移到下一列
 PUSH DX
 MOV AH, 0 ; 读当前时间, CH:CL=时:分, DH:DL=秒:1/100 秒
 ; 产生随机数基数

 INT 1AH
 MOV AX, DX
 POP DX
 AND AL, 03H ; 随机数为 1/100 秒的最低两位
 JZ DOWN ; 随机数的最低两位为 0 则下降一行
 CMP AL, 1
 JNZ LEVEL ; 随机数的最低两位为 ≥2 则水平移动
 DEC DH ; 随机数的最低位为 1 则上跳一行
 JMP BEGIN

```

```

DOWN: INC DH
LEVEL: JMP BEGIN
EXIT: POP AX ; 恢复当前显示方式(AL)
 MOV AH, 0
 INT 10H
 RET ; 返回 DOS
MAIN ENDP
; -----
DELAY PROC NEAR ; 延时 0.25s 子程序
 PUSH CX
 PUSH DX
 MOV DX, 25 ; 延时 0.25s
DEL1: MOV CX, 2801 ; 延时 10ms
DEL2: LOOP DEL2
 DEC DX
 JNZ DEL1
 POP DX
 POP CX
 RET
DELAY ENDP ; DELAY 子程序结束
; -----
CSEG ENDS ; 以上定义代码段
; *****
 END START ; 汇编语言源程序结束

```

10.14 分配给 PC 机主板上的 8253/54 定时器的端口地址是什么？

答：8253/54 定时器的 3 个独立计数器 Counter0、Counter1 和 Counter2 的端口地址分别为 40H、41H 和 42H。8253/54 内部还有一个公用的控制寄存器，端口地址为 43H。

10.15 8253/54 定时器的三个计数器，哪一个用于扬声器？它的端口地址是什么？

答：8253/54 定时器的计数器 Counter2 用于扬声器，它的端口地址为 42H。

10.16 下面的代码是利用监控端口 61H 的 PB4 来产生延迟时间的，它适用于所有的 286、386、Pentium PC 及兼容机。请指出该程序的延迟时间是多少？

```

 MOV DL, 200
BACK: MOV CX, 16572
WAIT: IN AL, 61H
 AND AL, 10H
 CMP AL, AH
 JE WAIT
 MOV AH, AL
 LOOP WAIT
 DEC DL
 JNZ BACK

```

答：该程序的延迟时间是  $200 \times 16572 \times 15.08\mu s = 49981152\mu s \approx 50s$ 。

10.17 在 PC 机上编写乐曲程序“Happy Birthday”，乐曲的音符及音频如下：

| 歌词    | 音符 | 音频  | 节拍  | 歌词    | 音符 | 音频  | 节拍  | 歌词    | 音符 | 音频  | 节拍  |
|-------|----|-----|-----|-------|----|-----|-----|-------|----|-----|-----|
| hap   | C  | 262 | 1/2 | day   | C  | 262 | 1   | so    | D  | 294 | 3   |
| py    | C  | 262 | 1/2 | to    | G  | 392 | 1   | hap   | Bb | 466 | 1/2 |
| birth | D  | 294 | 1   | you   | F  | 349 | 2   | py    | Bb | 466 | 1/2 |
| day   | C  | 262 | 1   | hap   | C  | 262 | 1/2 | birth | A  | 440 | 1   |
| to    | F  | 349 | 1   | py    | C  | 262 | 1/2 | day   | C  | 262 | 1   |
| you   | E  | 330 | 2   | birth | D  | 294 | 1   | to    | G  | 392 | 1   |
| hap   | C  | 262 | 1/2 | day   | A  | 440 | 1   | you   | F  | 349 | 2   |
| py    | C  | 262 | 1/2 | dear  | F  | 349 | 1   |       |    |     |     |
| birth | D  | 294 | 1   | so    | E  | 330 | 1   |       |    |     |     |

答：程序如下：

```

TITLE MUSIC — A music of ‘Happy Birthday’ ; 连接时需加上 GENSOUND 程序
EXTRN SOUND: FAR ; SOUND 是外部过程——通用发声程序
; *****
STACK SEGMENT PARA STACK ‘STACK’ ; 定义堆栈段
 DB 64 DUP (‘STACK...’)
STACK ENDS ; 以上定义堆栈段
; *****
DSEG SEGMENT PARA ‘DATA’ ; 定义数据段
MUS_FREQ DW 262, 262, 294, 262, 349, 330, 262, 262, 294, 262, 392, 349, 262, 262
 DW 294, 440, 349, 330, 294, 466, 466, 440, 262, 392, 349, -1
MUS_TIME DW 25, 25, 50, 50, 50, 100
 DW 25, 25, 50, 50, 50, 100
 DW 25, 25, 50, 50, 50, 150
 DW 25, 25, 50, 50, 50, 100
DSEG ENDS ; 以上定义数据段
; *****
CSEG SEGMENT PARA ‘CODE’ ; 定义代码段
 ASSUME CS: CSEG, DS: DSEG, SS: STACK
MUSIC PROC FAR
 PUSH DS ; 设置返回 DOS
 SUB AX, AX
 PUSH AX
 MOV AX, DSEG
 MOV DS, AX ; 给 DS 赋值

 LEA SI, MUS_FREQ ; 取发声的频率(音阶)表首地址
 LEA BP, MUS_TIME ; 取发声的节拍(时间)表首地址
FREQ: MOV DI, [SI] ; 读取频率值
 CMP DI, -1 ; 歌曲结束了吗?
 JE END_MUS
 MOV BX, DS:[BP] ; 读取节拍
 CALL SOUND ; 调通用发声子程序
 ADD SI, 2
 ADD BP, 2
 JMP FREQ
END_MUS: RET ; 返回 DOS
MUSIC ENDP
CSEG ENDS ; 以上定义代码段
; *****
 END MUSIC ; 汇编语言源程序结束

```

以下是 SOUND — 外部的通用发声子程序（教材 392 页）

```

TITLE SOUND — 通用发声子程序
; *****
PUBLIC SOUND ; 定义为公共过程
; *****
CSEG1 SEGMENT PARA ‘CODE’ ; 定义代码段
 ASSUME CS: CSEG1
SOUND PROC FAR
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 PUSH DI
 MOV AL, 0B6H ; 写定时器 8253 的工作方式
 OUT 43H, AL

```



```

 MOV DX, 12H ; 根据频率求 8253 的计数值, 即 533H*896/freq
 MOV AX, 533H*896 ; (DX),(AX)=123280H=533H*896
 DIV DI ; (DI) = freq
 OUT 42H, AL ; 向 8253 送计数值
 MOV AL, AH
 OUT 42H, AL
 IN AL, 61H ; 取 8255 的 PB 口当前内容, 并保护
 MOV AH, AL
 OR AL, 3 ; 开始发声, PB1=1, PB0=1
 OUT 61H, AL
WAIT1: MOV CX, 663 ; 延时(BX)×10ms
 CALL WAITF
 MOV AL, AH
 AND AL, 0FCH ; 停止发声, PB1=0, PB0=0
 OUT 61H, AL
 POP DI
 POP DX
 POP CX
 POP BX
 POP AX
 RET
SOUNDF ENDP
; *****
WAITF PROC NEAR
 PUSH AX
WAITF1: IN AL, 61H
 AND AL, 10H
 CMP AL, AH
 JE WAITF1
 MOV AH, AL
 LOOP WAITF1
 POP AX
 RET
WAITF ENDP
CSEG1 ENDS ; 以上定义代码段
; *****
 END

```

10.18 编写用键盘选择计算机演奏歌曲的程序。首先在屏幕上显示出歌曲名单如下:

```

A MUSIC 1
B MUSIC 2
C MUSIC 3

```

当从键盘上输入歌曲序号 A, B 或 C 时, 计算机则演奏所选择的歌曲, 当在键盘上按下 0 键时, 演奏结束。

答: 程序段如下:

```

MUS_LST DB 'A MUSIC 1', 0DH, 0AH
 DB 'B MUSIC 2', 0DH, 0AH
 DB 'C MUSIC 3', 0DH, 0AH
 DB '0 END', 0DH, 0AH, '$'
 ;
 MOV AH, 09 ; 显示字符串的 DOS 功能调用
 LEA DX, MUS_LIST
 INT 21H
INPUT: MOV AH, 1 ; 键盘输入一个字符的 DOS 功能调用
 INT 21H
 CMP AL, '0' ; 结束演奏吗?
 JE EXIT
 OR AL, 0010 0000B ; 变为小写字母

```

```

 CMP AL, 'a' ; 演奏歌曲 a 吗?
 JNZ B0
 CALL MUSIC1 ; 去演奏歌曲 A
 JMP INPUT
B0: CMP AL, 'b' ; 演奏歌曲 b 吗?
 JNZ C0
 CALL MUSIC2 ; 去演奏歌曲 B
 JMP INPUT
C0: CMP AL, 'c' ; 演奏歌曲 c 吗?
 JNZ INPUT
 CALL MUSIC3 ; 去演奏歌曲 C
 JMP INPUT
EXIT: RET ; 返回

```

## 第十一章. 习 题

11.1 写出文件代号式磁盘存取操作的错误代码:

- (1) 非法文件代号      (2) 路径未发现      (3) 写保护磁盘

答: 错误代码为:

- (1) 06                      (2) 03                      (4) 19

11.2 使用 3CH 功能建立一文件, 而该文件已经存在, 这时会发生什么情况?

答: 此操作将文件长度置为 0, 写新文件, 原文件内容被清除。

11.3 从缓冲区写信息到一个文件, 如果没有关闭文件, 可能会出现什么问题?

答: 文件结尾的部分信息就没有被写入磁盘, 从而造成写入的文件不完整。

11.4 下面的 ASCIZ 串有什么错误?

PATH\_NAME DB 'C:\PROGRAMS\TEST.DAT'

答: 此 ASCIZ 串的最后少了一个全 0 字节, 应改为:

PATH\_NAME DB 'C:\PROGRAMS\TEST.DAT', 0

11.5 下面为保存文件代号定义的变量有什么错误?

FILE\_HNDL DB ?

答: 文件代号是字类型, 因此应改为:

FILE\_HNDL DW ?

11.6 在 ASCPATH 字节变量中为驱动器 D 的文件 PATIENT.LST, 请定义 ASCIZ 串。

答: ASCPATH DB 'D:\PATIENT.LST', 0

11.7 对 11.6 题中的文件, 它的每个记录包含:

病例号(patient number):      5 字符,      姓名(name):      20 字符,

城市(city):      20 字符,      街道(street address):      20 字符,

出生年月(mmddyy):      6 字符,      性别(M/Fcode):      1 字符,

病房号(room number):      2 字符,      床号(bed number):      2 字符,

(1) 定义病人记录的各个域      (2) 定义保存文件代号的变量 FHANDLE

(3) 建文件      (4) 把 PATNTOUT 中的记录写入

(5) 关文件      (6) 以上文件操作包括测试错误

答: (1) PATNTOUT EQU THIS BYTE

patient DB 5 DUP (?)

name DB 20 DUP (?)

city DB 20 DUP (?)

street DB 20 DUP (?)

mmddyy DB 6 DUP (?)

M\_Fcode DB ?

room DB 2 DUP (?)

bed DB 2 DUP (?), 0AH, 0DH

```

COUNT = $-PATNTOUT ; 记录长度
(2) FHANDLE DW ?
(3) MOV AH, 3CH ; 建文件功能
 MOV CX, 00 ; 普通文件属性
 LEA DX, ASCPATH
 INT 21H
 JC ERROR
 MOV FHANDLE, AX ; 保存文件代号
(4) MOV AH, 40H ; 写文件功能
 MOV BX, FHANDLE ; 取文件代号
 MOV CX, COUNT ; 记录长度
 LEA DX, PATNTOUT ; 记录的首地址
 INT 21H
 JC ERROR
 CMP AX, COUNT ; 所有的字节都写入了吗?
 JNE ERROR1
(5) MOV AH, 3EH ; 关闭文件功能
 MOV BX, FHANDLE ; 取文件代号
 INT 21H
 JC ERROR
(6) 文件操作的测试错误已包括在(3)、(4)、(5)的操作中。

```

11.8 对 11.7 题的文件，用文件代号式编写一个完整的读文件程序，读出的每个记录存入 PATNTIN 并在屏幕上显示。

答：程序如下：

```

TITLE READDISP.EXE ; 利用文件代号式顺序读文件程序
; Read disk records created by hancreat
; -----
.model small
.stack 100h
.data
endcde db 0 ; 结束处理指示
fhandle dw ?
patntin db 80 DUP(' ') ; DTA
ascpath db 'd:\patient.lst', 0
openmsg db '***open error***', 0dh, 0ah
readmsg db '***read error***', 0dh, 0ah
row db 0
; -----
.code
begin proc far
 mov ax, @data
 mov ds, ax
 mov es, ax

 mov ax, 0600h
 call screen ; 清屏
 call curs ; 设置光标
 call openh ; 打开文件，设置 DTA
 cmp endcde, 0 ; 打开错误吗?
 jnz a0 ; 错误，转结束
contin: call readh ; 读磁盘记录
 cmp endcde, 0 ; 读错误吗?
 jnz a0 ; 错误，转结束
 call disph ; 没错，显示记录
 jmp contin

```

```

a0: mov ax, 4c00h ; 退出程序, 返回 DOS
 int 21h
begin endp
; -----
; 打开文件
openh proc near
 mov ah, 3dh
 mov al, 0
 lea dx, ascpath
 int 21h
 jc b1 ; 打开错误吗?
 mov fhandle, ax ; 没有错, 保存文件代号
 ret
b1: mov endcde, 01 ; 打开错误, 指示结束处理
 lea dx, openmsg
 call errm ; 显示出错信息
 ret
openh endp
; -----
; 读磁盘记录
readh proc near
 mov ah, 3fh
 mov bx, fhandle
 mov cx, 80
 lea dx, patntin
 int 21h
 jc c1 ; 读错误吗?
 cmp ax, 0 ; 文件已读完吗?
 je c2 ; 读完, 退出
 ret
c1: lea dx, openmsg ; 读错误
 call errm ; 显示出错信息
c2: mov endcde, 01 ; 读错误或文件读完, 指示结束处理
 ret
readh endp
; -----
; 显示记录
disph proc near
 mov ah, 40h ; 向标准输出设备(文件代号=01)写文件
 mov bx, 01 ; 标准输出设备的文件代号=01
 mov cx, 80
 lea dx, patntin
 int 21h
 cmp row, 24 ; 已到屏幕底部吗?
 jae d1 ; 已到屏幕底部, 退出
 inc row
 ret
d1: mov ax, 0601h
 call screen ; 屏幕上卷一行
 call curs ; 设置光标
 ret
disph endp
; -----
; 屏幕上卷
screen proc near ; 入口参数为 ax
 mov bh, 1eh ; 设置颜色

```

```

 mov cx, 0 ; 屏幕左上角
 mov dx, 184fh ; 屏幕右下角
 int 10h
 ret
screen endp
; -----
; 设置光标
curs proc near
 mov ah, 2 ; 设置光标
 mov bh, 0
 mov dh, row ; 行号
 mov dl, 0 ; 列号
 int 10h
 ret
curs endp
; -----
; 显示出错信息
errm proc near
 mov ah, 40h ; 向标准输出设备(文件代号=01)写文件
 mov bx, 01 ; 标准输出设备的文件代号=01
 mov cx, 20
 int 21h
 ret
errm endp
; -----
 end begin

```

- 11.9 编写建立并写入磁盘文件的程序。允许用户从键盘键入零件号(3 字符)，零(配)件名称(12 字符)，单价(1 个字)。程序使用文件代号式建立含有这些信息的文件。注意要把单价从 ASCII 码转换为二进制数。下面是输入的例子：

| part# | Description | price | part# | Description | price |
|-------|-------------|-------|-------|-------------|-------|
| 023   | Assemblers  | 00315 | 122   | Lifters     | 10520 |
| 024   | Linkages    | 00430 | 124   | Processors  | 21335 |
| 027   | Compilers   | 00525 | 127   | Labtlers    | 00960 |
| 049   | Compressors | 00920 | 232   | Bailers     | 05635 |
| 114   | Extractors  | 11250 | 237   | Grinders    | 08250 |
| 117   | Haulers     | 00630 | 999   |             | 000   |

答：程序如下：

```

TITLE HANCREAT.EXE ;利用文件代号式建立文件程序
;-----
 .model small
 .stack 100h
 .data
prompt1 db 'Please input Part#: $' ;提示输入零件号
prompt2 db 'Please input Description: $' ;提示输入零件名称
prompt3 db 'Please input Price: $' ;提示输入单价
maxlen db 13 ;最大输入长度，输入字符串功能的缓冲区
actlen db ? ;实际输入长度
buffer db 13 DUP (' ') ;输入字符串缓冲区
crlf db 0dh, 0ah, '$'
pathname db 'filename.lst', 0
handle dw ?
dta db 19 DUP (' ') ;DTA
errcde db 0 ;错误处理指示
opnmsg db '***open error***', 0dh, 0ah
wrtmsg db '***write error***', 0dh, 0ah
;-----

```

```

 .code
begin proc far
 mov ax, @data
 mov ds, ax
 mov es, ax

 mov ax, 0600h
 call scren ;清屏
 call curs ;设置光标
 call creath ;建立文件
 cmp errcde, 0 ;建立错误吗?
 jnz a0 ;错误, 转结束
contin: call proch ;记录处理
 cmp actlen, 0 ;输入的字符串长度为 0, 结束输入吗?
 jne contin ;不结束, 继续
 call clseh ;结束输入, 关闭文件
a0: mov ax, 4c00h ;退出程序, 返回 DOS
 int 21h
begin endp
;-----
;建立文件
creath proc near
 mov ah, 3ch
 mov cx, 0 ;普通属性
 lea dx, pathname
 int 21h
 jc bbb ;建立文件错误吗?
 mov handle, ax ;没有错, 保存文件代号
 ret
bbb: lea dx, opnmsg ;建立文件错误
 call errm ;显示出错信息
 ret
creath endp
;-----
;接收输入
proch proc near
 cld
 lea di, dta ;在 di 中设置 dta 的首地址
 lea dx, prompt1 ;输入零件号
 mov bx, 3 ;零件号最多 3 个字符
 call in_proc
 jc exit ;没有输入, 结束
 lea dx, prompt2 ;输入零件名称
 mov bx, 12 ;零件名称最多 12 个字符
 call in_proc
 jc exit ;没有输入, 结束
 lea dx, prompt3 ;输入单价
 mov bx, 5 ;零件单价最多 5 个十进制字符(相当于一个二进制字)
 call in_proc
 call dec_bin ;将十进制的单价转换为二进制的单价
 mov word ptr [dta+17], 0a0dh ;在 DTA 的最后插入回车换行符
 call writh ;用文件代号法写记录
exit: ret
proch endp
;-----
;输入字符串子程序

```

```

in_proc proc near
 mov ah, 09h ;显示提示信息
 int 21h
 push di
 lea di, buffer ;在 buffer 中填入空格符
 mov cl, maxlen
 mov ch, 0
 mov al, ' '
 rep stosb
 pop di
 mov ah, 0ah ;输入字符串
 lea dx, maxlen
 int 21h
 call disp_crlf
 cmp actlen, 0 ;实际输入字符数=0, 则没有输入, 结束
 je end_in
 push di
 lea di, buffer ;在 buffer 的后面填入空格符
 mov al, actlen
 mov ah, 0
 add di, ax
 mov cl, maxlen
 mov ch, 0
 mov al, actlen
 sub cl, al
 mov al, ' '
 rep stosb
 pop di
 lea si, buffer ;将 buffer 缓冲区内容送入 dta
 mov cx, bx
 rep movsb ;将输入内容送入 dta
 cld ;有输入字符, 返回(cf)=0
 jmp in_end
end_in: stc ;没有输入字符, 返回(cf)=1
in_end: ret
in_proc endp

```

-----

;将十进制的单价转换为二进制的单价子程序

```

dec_bin proc near
 mov bx, 0
 mov si, 0
 mov cx, 5
transfer: mov al, buffer[si] ;从十进制的高位到低位取数
 cmp al, 0dh ;是回车吗?
 je dec_bin1
 cmp al, ' ' ;是空格吗?
 je dec_bin1
 and al, 0fh ;将 ascii 码转换为十进制数
 mov ah, 0
 push cx
 xchg ax, bx ;十进制数高位×10+低位 = 二进制数
 mov cx, 10
 mul cx
 xchg ax, bx
 add bx, ax ;转换的二进制数在(bx)中
 pop cx
 inc si
 loop transfer

```

```

dec_bin1: mov word ptr [dta+15], bx ;存入单价到 dta 中的单价位置
 ret
dec_bin endp
;-----
;用文件代号法写记录
writh proc near
 mov ah, 40h
 mov bx, handle
 mov cx, 19
 lea dx, dta
 int 21h
 jnc ddd ;写文件错误吗?
 lea dx, wrtmsg
 call errm ;显示出错信息
 mov actlen, 0
ddd: ret
writh endp
;-----
;用文件代号法关闭文件
clseh proc near
 mov dta, 1ah ;写文件结束符 1ah
 call writh
 mov ah, 3eh
 mov bx, handle
 int 21h
 ret
clseh endp
;-----
;屏幕上卷
scren proc near ;入口参数为 ax
 mov bh, 1eh ;设置颜色
 mov cx, 0 ;屏幕左上角
 mov dx, 184fh ;屏幕右下角
 int 10h
 ret
scren endp
;-----
;设置光标
curs proc near
 mov ah, 2 ;设置光标
 mov bh, 0
 mov dh, 0 ;行号
 mov dl, 0 ;列号
 int 10h
 ret
curs endp
;-----
;显示出错信息
errm proc near
 mov ah, 40h ;向标准输出设备(文件代号=01)写文件
 mov bx, 01 ;标准输出设备的文件代号=01
 mov cx, 20
 int 21h
 mov errcde, 01 ;错误代码置 1
 ret
errm endp
;-----

```



```

disp_crlf proc near ; 显示回车换行符子程序
 lea dx, crlf
 mov ah, 09h
 int 21h
 ret
disp_crlf endp ; disp_crlf 子程序结束
;-----
end begin ; 汇编语言源程序结束

```

11.10 编写一个程序使用文件代号式读出并显示 11.9 题建立的文件。注意, 要把二进制数表示的单价转换为 ASCII 码。

答: 用文件代号式读出并显示文件, 程序如下:

```

TITLE HANDREAD.EXE ; 利用文件代号式顺序读并显示文件程序
; Read disk records created by hancreat
;-----
 .model small
 .stack 100h
 .data
endcde db 0 ; 结束处理指示
crlf db 0dh, 0ah, '$'
pathname db 'filename.lst', 0
message db ' Part# Description Price', 0dh, 0ah, '$'
handle dw ?
tackline db ' | $'
dta db 19 DUP (' ') ; DTA
errcde db 0 ; 错误处理指示
opnmsg db '***open error***', 0dh, 0ah
readmsg db '***read error***', 0dh, 0ah
row db 0
;-----
 .code
begin proc far
 mov ax, @data
 mov ds, ax
 mov es, ax

 mov ax, 0600h
 call screen ; 清屏
 call curs ; 设置光标
 lea dx, message ; 显示标题
 mov ah, 09h
 int 21h
 inc row
 call openh ; 打开文件, 设置 DTA
 cmp endcde, 0 ; 打开错误吗?
 jnz a0 ; 错误, 转结束
contin: call readh ; 读磁盘记录
 cmp endcde, 0 ; 读错误吗?
 jnz a0 ; 错误, 转结束
 call disph ; 没错, 显示记录
 jmp contin
a0: mov ax, 4c00h ; 退出程序, 返回 DOS
 int 21h
begin endp
;-----
; 打开文件
openh proc near

```

```

 mov ah, 3dh
 mov al, 0
 lea dx, pathname
 int 21h
 jc bbb ;打开错误吗?
 mov handle, ax ;没有错, 保存文件代号
 ret
bbb: mov endcde, 01 ;打开错误, 指示结束处理
 lea dx, readmsg
 call errm ;显示出错信息
 ret
openh endp
;-----
;读磁盘记录
readh proc near
 mov ah, 3fh
 mov bx, handle
 mov cx, 19
 lea dx, dta
 int 21h
 jc c1 ;读错误吗?
 cmp ax, 0 ;文件已读完吗?
 je c2 ;读完, 退出
 cmp dta, 1ah ;文件结束符吗?
 je c2
 ret
c1: lea dx, opnmsg ;读错误
 call errm ;显示出错信息
c2: mov endcde, 01 ;读错误或文件读完, 指示结束处理
 ret
readh endp
;-----
;显示记录
disph proc near
 lea dx, tackline ;显示输出“ | ”
 mov ah, 09h
 int 21h
 mov ah, 40h ;向标准输出设备(文件代号=01)写文件
 mov bx, 01 ;标准输出设备的文件代号=01
 mov cx, 3
 lea dx, dta
 int 21h
 lea dx, tackline ;显示输出“ | ”
 mov ah, 09h
 int 21h
 mov ah, 40h ;向标准输出设备(文件代号=01)写文件
 mov bx, 01 ;标准输出设备的文件代号=01
 mov cx, 12
 lea dx, dta+3
 int 21h
 lea dx, tackline ;显示输出“ | ”
 mov ah, 09h
 int 21h
 mov si, word ptr [dta+15]
 call bin_dec ;转换为十进制数显示
 lea dx, tackline ;显示输出“ | ”
 mov ah, 09h

```

```

 int 21h
 call disp_crlf
 cmp row, 24 ;已到屏幕底部吗?
 jae ddd ;已到屏幕底部, 退出
 inc row
 ret
ddd: mov ax, 0601h
 call screen ;屏幕上卷一行
 call curs ;设置光标
 ret
disph endp
;-----
;将二进制的单价转换为十进制的单价并显示子程序
bin_dec proc near
 push cx
 mov cx, 10000d
 call dec_div ;调除法并显示输出子程序
 mov cx, 1000d
 call dec_div
 mov cx, 100d
 call dec_div
 mov cx, 10d
 call dec_div
 mov cx, 1d
 call dec_div
 pop cx
 ret
bin_dec endp
;-----
;除法并显示输出子程序
dec_div proc near
 mov ax, si
 mov dx, 0
 div cx
 mov si, dx ;余数保存在(si)中作下一次的除法
 mov dl, al ;商(在 00h~09h 范围内)送(dl)
 add dl, 30h ;转换为 0~9 的 ascii 码
 mov ah, 02h ;显示输出
 int 21h
 ret
dec_div endp
;-----
;屏幕上卷
screen proc near ;入口参数为 ax
 mov bh, 1eh ;设置颜色
 mov cx, 0 ;屏幕左上角
 mov dx, 184fh ;屏幕右下角
 int 10h
 ret
screen endp
;-----
;设置光标
curs proc near
 mov ah, 2 ;设置光标
 mov bh, 0
 mov dh, row ;行号
 mov dl, 0 ;列号

```

```

 int 10h
 ret
curs endp
;-----
;显示出错信息
errm proc near
 mov ah, 40h ;向标准输出设备(文件代号=01)写文件
 mov bx, 01 ;标准输出设备的文件代号=01
 mov cx, 20
 int 21h
 ret
errm endp
;-----
disp_crlf proc near ;显示回车换行符子程序
 lea dx, crlf
 mov ah, 09h
 int 21h
 ret
disp_crlf endp ; disp_crlf 子程序结束
;-----
 end begin

```

11.11 对 11.9 题建立的文件按下面的要求编写程序:

- (1) 把所有的记录读入内存的数据缓冲区 TABLE;
- (2) 显示字符串提示用户输入零(配)件号及其数量;
- (3) 按零件搜索 TABLE;
- (4) 如果发现所要求的零件, 用它的单价计算出总价(单价×数量);
- (5) 显示零(配)件说明及总价值。

答: 程序如下:

```

TITLE READ11.EXE ;利用文件代号式读并计算显示程序
;Read disk records created by hancreat
;-----
 .model small
 .stack 100h
 .data
endcde db 0 ;结束处理指示
pathname db 'filename.lst', 0
in_mes1 db '请输入 3 位数的零件号 Part#: ', '$'
in_mes2 db '请输入该零件的数量: ', '$'
out_mes1 db '输入的不是数字! 请重新输入数字: ', '$'
out_mes2 db '输入的零件号不存在! 请重新输入 3 位数的零件号 Part#: ', '$'
in_buffer db 6, ?, 6 dup(20h) ;输入缓冲区
message db ' Part# Description Sum_Price', 0dh, 0ah, '$'
tackline db ' | $'
sum_price dw 0, 0
decimal db 10 DUP(0), '$'
crlf db 0dh, 0ah, '$'
handle dw ?
table db 19*100 DUP(' ') ;table, 足够大
errcde db 0 ;错误处理指示
opnmsg db '***open error***', 0dh, 0ah
readmsg db '***read error***', 0dh, 0ah
;-----
 .code
begin proc far
 mov ax, @data
 mov ds, ax
 mov es, ax

```

```

 mov ax, 0600h
 call screen ;清屏
 call curs ;设置光标
 call openh ;打开文件, 设置 TABLE
 cmp endcde, 0 ;打开错误吗?
 jnz a0 ;错误, 转结束
 call readh ;读磁盘记录
 cmp endcde, 0 ;读错误吗?
 jnz a0 ;错误, 转结束
 call in_Part ;没错, 输入零件号和零件数量
a0: mov ax, 4c00h ;退出程序, 返回 DOS
 int 21h
begin endp
;-----
;打开文件
openh proc near
 mov ah, 3dh
 mov al, 0
 lea dx, pathname
 int 21h
 jc bbb ;打开错误吗?
 mov handle, ax ;没有错, 保存文件代号
 ret
bbb: mov endcde, 01 ;打开错误, 指示结束处理
 lea dx, opnmsg
 call errm ;显示出错信息
 ret
openh endp
;-----
;读磁盘记录
readh proc near
 mov ah, 3fh
 mov bx, handle
 mov cx, 19*100 ;准备读入的字节数
 lea dx, table
 int 21h
 jc c1 ;读错误吗?
 cmp ax, 0 ;文件已读完吗?
 je c2 ;读完, 退出
 cmp table, 1ah ;文件结束符吗?
 je c2
 mov bp, ax ;读成功则在 AX 中返回实际读入的字节数存入 bp
 ret
c1: lea dx, readmsg ;读错误
 call errm ;显示出错信息
c2: mov endcde, 01 ;读错误或文件读完, 指示结束处理
 ret
readh endp
;-----
;输入零件号和零件数量
in_Part proc near
 lea dx, in_mes1 ;显示提示信息, 提示输入零件号
in_Part1: call input ;输入数据
 cmp in_buffer+1, 3 ;输入的零件号个数是 3 位吗?
 lea dx, out_mes2 ;显示提示信息, 提示重新输入零件号

```

```

 jne in_Part1
 cld
 mov ax, bp ;取实际读入文件的字节数
 mov cl, 19 ;每个记录的长度为 19 个字符
 div cl ;计算实际读取的记录数在 al 中
 mov bl, al
 mov bh, 0 ;从第 0 个记录开始顺序查找
in_Part2: lea si, in_buffer+2 ;查找零件号对应的零件
 lea di, table
 mov al, 19
 mul bh
 add di, ax ;计算某个记录的首地址
 mov word ptr decimal, di ;保存首地址
 mov cx, 3
 repe cmpsb
 je in_Part3 ;找到对应的零件
 inc bh ;找下一个记录
 cmp bh, bl
 jb in_Part2
 jmp in_Part1 ;未找到对应的零件重新输入
in_Part3: lea dx, in_mes2 ;显示提示信息，提示输入零件数量
 call input ;输入数据
 call dec_bin ;将输入数据转换为二进制数，在 bx 中
 mov di, word ptr decima ;di 指向该记录的首地址
 mov ax, [di+15] ;取单价
 mul bx ;总价格在(dx),(ax)中
 mov sum_price, ax
 mov sum_price+2, dx
 call disp_rec ;显示信息
 ret
in_Part endp
;-----
;输入数据
input proc near
input1: mov ah, 09h ;显示字符串
 int 21h
 mov ah, 0ah ;输入字符串
 lea dx, in_buffer
 int 21h
 lea dx, out_mes1 ;显示提示信息
 mov cl, in_buffer+1
 cmp cl, 0 ;输入的数字个数为 0 吗?
 jz input1
 mov ch, 0
 mov bx, 2
input2: mov al, in_buffer[bx] ;输入的是数字 0~9 吗?
 cmp al, '0'
 jb input1
 cmp al, '9'
 ja input1
 inc bx
 loop input2
 ret
input endp
;-----
;将十进制数转换为二进制数子程序
dec_bin proc near

```

```

 mov bx, 0
 mov si, 2
 mov cl, in_buffer+1
 mov ch, 0
transfer: mov al, in_buffer[si] ;从十进制的高位到低位取数
 and al, 0fh ;将 ascii 码转换为十进制数
 mov ah, 0
 push cx
 xchg ax, bx ;十进制数高位×10+低位 = 二进制数
 mov cx, 10
 mul cx
 add bx, ax ;转换的二进制数在(bx)中
 pop cx
 inc si
 loop transfer
 ret
dec_bin endp
;-----
;显示记录
disp_rec proc near
 call disp_crlf
 lea dx, message ;显示标题
 mov ah, 09h
 int 21h
 lea dx, tackline ;显示输出“ | ”
 mov ah, 09h
 int 21h
 mov ah, 40h ;向标准输出设备(文件代号=01)写文件
 mov bx, 01 ;标准输出设备的文件代号=01
 mov cx, 3 ;显示 3 位数的零件号
 mov dx, word ptr decima ;dx 指向该记录的首地址
 int 21h
 lea dx, tackline ;显示输出“ | ”
 mov ah, 09h
 int 21h
 mov ah, 40h ;向标准输出设备(文件代号=01)写文件
 mov bx, 01 ;标准输出设备的文件代号=01
 mov cx, 12 ;显示 12 位的零件说明
 mov dx, word ptr decima ;dx 指向该记录的首地址
 add dx, 3
 int 21h
 lea dx, tackline ;显示输出“ | ”
 mov ah, 09h
 int 21h
 call bin_dec ;总价格转换为十进制数显示
 lea dx, tackline ;显示输出“ | ”
 mov ah, 09h
 int 21h
 call disp_crlf
 ret
disp_rec endp
;-----
;4 字节二进制数转换为 10 进制子程序
bin_dec proc near
 mov bx, 0 ;10 字节的 bcd 码单元清 0
 mov cx, 10
bin_dec1: mov decimal[bx], 0

```

```

 inc bx
 loop bin_dec1
 mov cx, 4*8 ;4 字节二进制数共 4*8=32 位
bin_dec2: mov bx, 10-1 ;计算(((a31*2+a30)*2+a29)...)*2+a0
 shl word ptr [sum_price],1 ;4 字节二进制数左移 1 位
 rcl word ptr [sum_price+2],1
 push cx
 mov cx, 10
bin_dec3: mov al, decimal[bx] ;计算(...)*2+ai, ai 由进位位带入
 adc al, al
 aaa ;非压缩 bcd 码加法调整
 mov decimal[bx], al
 dec bx
 loop bin_dec3
 pop cx
 loop bin_dec2
 call disp
 ret
bin_dec endp
;-----
disp proc near ;显示输出子程序
 mov cx, 10
 mov bx, 0
disp1: add decimal[bx], 30h ;变为 ascii 码
 inc bx
 loop disp1
 mov cx, 10 ;下面 5 条指令是为了不显示数据左边的“0”
 cld
 lea di, decimal
 mov al, 30h ;30h 为“0”的 ascii 码
 repe scasb
 dec di
 mov dx, di
 mov ah, 09h
 int 21h
 ret
disp endp ;disp 子程序结束
;-----
;屏幕上卷
screen proc near ;入口参数为 ax
 mov bh, 1eh ;设置颜色
 mov cx, 0 ;屏幕左上角
 mov dx, 184fh ;屏幕右下角
 int 10h
 ret
screen endp
;-----
;设置光标
curs proc near
 mov ah, 2 ;设置光标
 mov bh, 0
 mov dh, 0 ;行号
 mov dl, 0 ;列号
 int 10h
 ret
curs endp
;-----

```



```

;显示出错信息
errm proc near
 mov ah, 40h ;向标准输出设备(文件代号=01)写文件
 mov bx, 01 ;标准输出设备的文件代号=01
 mov cx, 20
 int 21h
 ret
errm endp

;-----
disp_crlf proc near ;显示回车换行符子程序
 lea dx, crlf
 mov ah, 09h
 int 21h
 ret
disp_crlf endp ; disp_crlf 子程序结束
;-----
 end begin

```

- 11.12 用随机处理记录的方式编写程序, 将用户需要的零(配)件记录读取到 TABLE, 并根据键入的数量, 计算出总价值, 然后显示出零(配)件说明及总价值。

答: 程序如下:

```

TITLE READ_RAN.EXE ;利用文件代号式随机读并计算显示程序
;Read disk records created by hancreat
;-----
 .model small
 .stack 100h
 .data
endcde db 0 ;结束处理指示
pathname db 'filename.lst', 0
in_mes1 db '请输入 3 位数的零件号 Part#: ', '$'
in_mes2 db '请输入该零件的数量: ', '$'
out_mes1 db '输入的不是数字! 请重新输入数字: ', '$'
out_mes2 db '输入的零件号不存在! 请重新输入 3 位数的零件号 Part#: ', '$'
in_buffer db 6, ?, 6 dup(20h) ;输入缓冲区
message db ' Part# Description Sum_Price', 0dh, 0ah, '$'
tackline db ' | $'
sum_price dw 0, 0
decimal db 10 DUP(0), '$'
crlf db 0dh, 0ah, '$'
handle dw ?
table db 19 DUP(' ') ;table
errcde db 0 ;错误处理指示
opnmsg db '***open error***', 0dh, 0ah
readmsg db '***read error***', 0dh, 0ah
movmsg db '***move error***', 0dh, 0ah
;-----
 .code
begin proc far
 mov ax, @data
 mov ds, ax
 mov es, ax

 mov ax, 0600h
 call screen ;清屏
 call curs ;设置光标
 call openh ;打开文件, 设置 TABLE
 cmp endcde, 0 ;打开错误吗?
 jnz a0 ;错误, 转结束

```

```

 call in_Part ;没错，输入零件号和零件数量
a0: mov ax, 4c00h ;退出程序，返回 DOS
 int 21h
begin endp
;-----
;打开文件
openh proc near
 mov ah, 3dh
 mov al, 0
 lea dx, pathname
 int 21h
 jc bbb ;打开错误吗？
 mov handle, ax ;没有错，保存文件代号
 ret
bbb: mov endcde, 01 ;打开错误，指示结束处理
 lea dx, opnmsg
 call errm ;显示出错信息
 ret
openh endp
;-----
;读磁盘记录
readh proc near
 mov ah, 3fh
 mov bx, handle
 mov cx, 19 ;准备读入的字节数
 lea dx, table
 int 21h
 jc c1 ;读错误吗？
 cmp ax, 0 ;文件已读完吗？
 je c2 ;读完，退出
 cmp table, 1ah ;文件结束符吗？
 je c2
 mov bp, ax ;读成功则在 AX 中返回实际读入的字节数存入 bp
 ret
c1: mov endcde, 01 ;读错误或文件读完，指示结束处理
 lea dx, readmsg
 call errm ;读错误
 call errm ;显示出错信息
 jmp c3
c2: mov endcde, 02 ;读错误或文件读完，指示结束处理
c3: ret
readh endp
;-----
;绝对移动文件读写指针
mov_pointer proc near
 mov ah, 42h
 mov al, 0
 mov bx, handle
 int 21h
 jc d1 ;错误吗？
 ret
d1: lea dx, movmsg
 call errm ;错误
 call errm ;显示出错信息
 mov endcde, 01 ;错误，指示结束处理
 ret
mov_pointer endp
;-----
;输入零件号和零件数量

```

```

in_Part proc near
 lea dx, in_mes1 ;显示提示信息, 提示输入零件号
in_Part1: call input ;输入数据
 cmp in_buffer+1, 3 ;输入的零件号个数是 3 位吗?
 lea dx, out_mes2 ;显示提示信息, 提示重新输入零件号
 jne in_Part1
 cld
 mov cx, 0 ;位移量的高位字
 mov dx, 0 ;位移量的低位字
 call mov_pointer ;绝对移动文件读写指针到文件首
in_Part2: call readh ;读磁盘记录
 cmp endcde, 2 ;读文件结束吗?
 je in_Part1 ;结束, 未找到对应的零件重新输入
 cmp endcde, 1 ;读错误吗?
 je in_Part4 ;错误, 转结束
 lea si, in_buffer+2 ;查找零件号对应的零件
 lea di, table
 mov cx, 3
 repe cmpsb
 je in_Part3 ;找到对应的零件
 jmp in_Part2 ;找下一个零件

in_Part3: lea dx, in_mes2 ;显示提示信息, 提示输入零件数量
 call input ;输入数据
 call dec_bin ;将输入数据转换为二进制数, 在 bx 中
 lea di, table ;di 指向该记录的首地址
 mov ax, [di+15] ;取单价
 mul bx ;总价格在(dx),(ax)中
 mov sum_price, ax
 mov sum_price+2, dx
 call disp_rec ;显示信息
in_Part4: ret
in_Part endp
;-----
;输入数据
input proc near
input1: mov ah, 09h ;显示字符串
 int 21h
 mov ah, 0ah ;输入字符串
 lea dx, in_buffer
 int 21h
 lea dx, out_mes1 ;显示提示信息
 mov cl, in_buffer+1
 cmp cl, 0 ;输入的数字个数为 0 吗?
 jz input1
 mov ch, 0
 mov bx, 2
input2: mov al, in_buffer[bx] ;输入的是数字 0~9 吗?
 cmp al, '0'
 jb input1
 cmp al, '9'
 ja input1
 inc bx
 loop input2
 ret
input endp

```

```

;-----
;将十进制数转换为二进制数子程序
dec_bin proc near
 mov bx, 0
 mov si, 2
 mov cl, in_buffer+1
 mov ch, 0
transfer: mov al, in_buffer[si] ;从十进制的高位到低位取数
 and al, 0fh ;将 ascii 码转换为十进制数
 mov ah, 0
 push cx
 xchg ax, bx ;十进制数高位×10+低位 = 二进制数
 mov cx, 10
 mul cx
 add bx, ax ;转换的二进制数在(bx)中
 pop cx
 inc si
 loop transfer
 ret
dec_bin endp
;-----
;显示记录
disp_rec proc near
 call disp_crlf
 lea dx, message ;显示标题
 mov ah, 09h
 int 21h
 lea dx, tackline ;显示输出“ | ”
 mov ah, 09h
 int 21h
 mov ah, 40h ;向标准输出设备(文件代号=01)写文件
 mov bx, 01 ;标准输出设备的文件代号=01
 mov cx, 3 ;显示 3 位数的零件号
 lea dx, table ;dx 指向该记录的首地址
 int 21h
 lea dx, tackline ;显示输出“ | ”
 mov ah, 09h
 int 21h
 mov ah, 40h ;向标准输出设备(文件代号=01)写文件
 mov bx, 01 ;标准输出设备的文件代号=01
 mov cx, 12 ;显示 12 位的零件说明
 lea dx, table ;dx 指向该记录的首地址
 add dx, 3
 int 21h
 lea dx, tackline ;显示输出“ | ”
 mov ah, 09h
 int 21h
 call bin_dec ;总价格转换为十进制数显示
 lea dx, tackline ;显示输出“ | ”
 mov ah, 09h
 int 21h
 call disp_crlf
 ret
disp_rec endp
;-----
;4 字节二进制数转换为 10 进制子程序
bin_dec proc near

```

```

 mov bx, 0 ;10 字节的 bcd 码单元清 0
 mov cx, 10
bin_dec1: mov decimal[bx], 0
 inc bx
 loop bin_dec1
 mov cx, 4*8 ;4 字节二进制数共 4*8=32 位
bin_dec2: mov bx, 10-1 ;计算(((a31*2+a30)*2+a29)...)*2+a0
 shl word ptr [sum_price],1 ;4 字节二进制数左移 1 位
 rcl word ptr [sum_price+2],1
 push cx
 mov cx, 10
bin_dec3: mov al, decimal[bx] ;计算(...)*2+ai, ai 由进位位带入
 adc al, al
 aaa ;非压缩 bcd 码加法调整
 mov decimal[bx], al
 dec bx
 loop bin_dec3
 pop cx
 loop bin_dec2
 call disp
 ret
bin_dec endp
;-----
disp proc near ;显示输出子程序
 mov cx, 10
 mov bx, 0
disp1: add decimal[bx], 30h ;变为 ascii 码
 inc bx
 loop disp1
 mov cx, 10 ;下面 5 条指令是为了不显示数据左边的“0”
 cld
 lea di, decimal
 mov al, 30h ;30h 为“0”的 ascii 码
 repe scasb
 dec di
 mov dx, di
 mov ah, 09h
 int 21h
 ret
disp endp ;disp 子程序结束
;-----
;屏幕上卷
screen proc near ;入口参数为 ax
 mov bh, 1eh ;设置颜色
 mov cx, 0 ;屏幕左上角
 mov dx, 184fh ;屏幕右下角
 int 10h
 ret
screen endp
;-----
;设置光标
curs proc near
 mov ah, 2 ;设置光标
 mov bh, 0
 mov dh, 0 ;行号
 mov dl, 0 ;列号
 int 10h

```

```
 ret
curs endp
;-----
;显示出错信息
errm proc near
 mov ah, 40h ;向标准输出设备(文件代号=01)写文件
 mov bx, 01 ;标准输出设备的文件代号=01
 mov cx, 20
 int 21h
 ret
errm endp
;-----
disp_crlf proc near ;显示回车换行符子程序
 lea dx, crlf
 mov ah, 09h
 int 21h
 ret
disp_crlf endp ; disp_crlf 子程序结束
;-----
 end begin
```