第1章 基础知识

1.1 内容提要

1.1.1 进位计数制

按进位的原则进行计数，称为进位计数制。常用的进位计数制有十进制、二进制、八进制和十六进制等。进位计数制的基本特点如下：

(1)逢 R 进一。R 是计数制中所用到的数码的个数。

(2)采用位权表示法。在一个进位计数制表示的数中，处在不同数位的数码，代表着不同的数值，某一个数位的数值是由这一位数码的值乘上处在这位的一个固定常数。不同数位上的固定常数称为该位的位权或权。不同数位上有不同的位权值。

一般来说，一个 R 进制的数 N 有两种表示方式：

(1)并列表示方式：

$$(N)_R=(K_{n-1}K_{n-2}\cdots K_1K_0.K_{-1}K_{-2}\cdots K_{-m})_R$$

其中 n 为整数的位数，m 为小数的位数，R 表示基数，K_i 为不同数位上的数值。

(2)多项式表示法，也称按权展开式：

$$(N)_R=K_{n-1}R^{n-1}+K_{n-2}R^{n-2}+\cdots+K_1R^1+K_0R^0+K_{-1}R^{-1}+\cdots+K_{-m}R^{-m}$$

或：

$$(N)_R = (\sum_{i=-m}^{n-1} K_i R^i)_R$$

其中 R 代表进位制的基数；m、n 为正的常整数，分别代表 N 的整数部分的位数和小数部分的位数，K_i 是 R 进制中 R 个数字符号中的任何一个：

$$0 \leqslant K_i \leqslant R-1$$

1.1.2 不同进制数据间的转换

在转换时，其整数部分与小数部分应分别进行转换，将转换后的结果合并，整数部分与小数部分之间用小数点隔开，就得到相应的转换结果。

(1)二进制数转换为十进制数

转换规则是“按权值相加”。也就是说，只要把二进制数中数位是“1”的那些位的权值相加，其和就是等效的十进制数。

(2)十进制数转换为二进制数

对十进制整数和小数部分分别进行转换。转换结束后将整数转换结果写在左边，小数转换结果写在右边，中间点上小数点。

整数部分的转换规则：将十进制整数用基数 2 连续去除，直到商为 0 为止，将每次除得的余数反向排列，就可得到十进制数整数部分的转换结果。反向排列是指最后得到的余数排在前面，作为结果的最高位，最先得到的余数排在后面，作为结果的最低位。

小数部分的转换规则：将十进制数的小数部分用基数 2 连续去乘，直到小数部分为 0 或达到精度为止，将每次所得的乘积的整数部分正向排列，就可得到十进制小数的转换结果。正向排列是指最先得到的整数为结果的最高位，最后得到的整数为结果的最低位。

(3)二进制数转换为(八进制)十六进制数

将二进制数以小数点为界，向左、向右分别按(3 位)4 位一组

划分，不足(3位)4位的部分用“0”补足，将每一组数写成对应的(八进制)十六进制数，就可得到转换结果。

1.1.3 常用各进制数据的运算

B——表示二进制；D——表示十进制；Q——表示八进制；H——表示十六进制。

1. 二进制数的运算规则

二进制数之间可以执行算术与逻辑运算，其规则简单，容易实现。

①加法规则：0+0=0

0+1=1

1+0=1

1+1=0(产生向高位的一次进位，即结果为10)

②减法规则：0−0=0

0−1=1(要向高位借位一次)

1−0=1

1−1=0

③乘法规则：0×0=0

0×1=0

1×0=0

1×1=1

④除法规则：二进制除法的计算方法，与十进制除法类似，也由减法、上商等操作逐步完成。

2. 十六进制数的运算规则

①加法规则：“逢十六进一”。

②减法规则：“借一当十六”。

③乘法规则：

(i)与十进制乘法一样，逐位相乘，结果错位相加。

(ii)两位相乘时，化为十进制数相乘。

(iii)两位相乘的结果除以 16，商做进位，余数留本位。

④除法规则：除法的计算方法，与十进制除法类似，也由减法、上商等操作逐步完成，但应是“逢十六进一”。

3. 八进制数的运算规则

八进制数的运算规则与十六进制数的运算规则相类似。

1.1.4 计算机中的数据表示

1. 原码

(1)原码表示法

将数的真值形式中的正(负)号，用代码 0(1)来表示，数值部分用二进制来表示。

(2)原码的特点

①“0”的原码有两种表示法：

$$[+0]_{原}=00000000B$$

$$[+0]_{原}=10000000B$$

②n 位二进制原码所能表示的数值范围为：

$$-(2^{n-1}-1)\sim(2^{n-1}-1)$$

③原码表示一个数时，最高位为符号位。

符号位为 0 时，其后面的 n－1 位为数值部分，这个数为正数。

符号位为 1 时，其后面的 n－1 位为数值部分，这个数为负数。

2. 补码

(1)补码表示法

正数的补码表示与正数的原码相同，负数的补码表示为它的原码表示除符号位以外，其余位按位取反且在最低位加 1 形成。

(2)补码的特点

①“0”的补码表示是唯一的。

$[+0]_{补}=[-0]_{补}=00000000B$（8位二进制补码表示）

②n位二进制补码所能表示的数值范围为：

$$-2^{n-1}\sim 2^{n-1}-1$$

③补码表示一个数时，最高位为符号位。

④补码运算时符号位无需单独处理。

⑤采用补码运算时，减法可用加法来实现。

(3)补码加法和减法的规则

$$[x+y]_{补}=[x]_{补}+[y]_{补}$$

$$[x-y]_{补}=[x]_{补}+[-y]_{补}$$

1.1.5 二进制编码

1. BCD码

BCD码即8421码，它用4位二进制数来表示十进制数，4个二进制位的权从高向低分别为8、4、2和1，使用的是基2码头0000，0001，…，1001这10种组合，分别表示0～9这10个值。BCD码包括非压缩的BCD码和压缩的BCD码两种。

在计算机内实现BCD数之间的算术运算时，在某些情况下，需要对运算的结果进行“加6修正”。

2. ASCII码

ASCII编码即American Standard Code for Information interchange，是一种常用的字符表示形式。它对128个字符进行了编码，其中包括大写字母、小写字母、字数0～9及回车、换行和响铃等非控制字母。

有些机器的编码采用8位的扩展ASCII码编码，可表示256个字符。有些时候第8位用于奇偶校验。

3. 国标码

GB2312－80是我国在计算机中表示汉字的标准编码。该标准

规定了汉字交换用的基本汉字字符和一些图形字符，它们共计 7445 个，其中汉字有 6763 个。汉字编码表共 94 行(区)和 94 列(位)。行号称为区号，列号称为位号。国标码是直接把第一字节和第二字节编码拼起来得到的，通常用十六进制表示。在一个汉字的区码和位码上分别加十六进制数 20H，即构成该汉字的国标码。

1.1.6 几种基本的逻辑运算

表 1-1 列出了几种基本的逻辑运算。

表 1-1

逻辑运算	含　义	逻辑变量	符号表示	运算规则
“与”(AND)	逻辑乘	A,B	·或∧	只有当 A,B 两变量取值均为 1 时，它们的运算结果才是 1
“或”(OR)	逻辑加	A,B	＋或∨	A,B 两变量只要有一个变量取值均为 1 时，则它们的运算结果就是 1
“非”(OR)	逻辑非	A	$\bar{A}$	若 A＝0，则 $\bar{A}$＝1 若 A＝1，则 $\bar{A}$＝0
“异或”(XOR)	逻辑异或	A,B	∀	A,B 两变量的取值不同(相异)时，它们的运算结果就是 1

1.1.7 考　点

1. 常用的各种进制数的表示、转换规则和运算；
2. 带符号数的原码、补码表示方法及其补码运算。

1.2 例题精解

例 1　完成下列数制转换：

(1)将十进制数 20.75 转换为二进制数。

(2)将二进制数 1101.11 转换为八进制和十六进制数。

【分析与解答】

(1)20.75

整数部分转换(将十进制整数用基数 2 连续去除,直到商为 0 为止,将每次所得的余数反向排列。)

首先 20 除以 2→ 2 | 20

其次 10 除以 2→ 2 | 10 ←第 1 次得商"10",余数为"0"

再次 5 除以 2→ 2 | 5 ←第 2 次得商"5",余数为"0" ↑低位

2 | 2 ←第 3 次得商"2",余数为"1"

2 | 1 ←第 4 次得商"1",余数为"0" 高位

0 ←第 5 次得商"0",余数为"1"

整数转换的结果为:10100B。

所以 20D=10100B。

小数部分转换(将十进制数的小数部分用基数 2 连续去乘,直到小数部分为 0 或达到精度为止,将每次所得的乘积的整数部分正向排列。)

$$\begin{array}{r} 0.75 \\ \times\quad 2 \\ \hline 1.50 \end{array}$$

第 1 次得整数 1 ←首先 0.75 乘 2

$$\begin{array}{r} 2 \\ \hline 1.00 \end{array}$$

第 2 次得整数 1 ←最后小数部分为 0

小数转换结果为:0.11B。

所以 0.75D=0.11B。

最后将整数部分和小数部分的转换结果合并起来,中间点上小数点,就得到 20.75 的转换结果为:20.75D=10100.11B。

(2)1101.11

①将二进制数转换为八进制数

根据转换规则，将 1101.11B 从小数点开始向左、向右按 3 位分组，不足 3 位的用 0 补足，分组如下：

001，101.110

将每 3 位二进制数写成对应的 1 位八进制数。转换结果为：

1101.11B＝15.6Q

②将二进制数转换为十六进制数

根据转换规则，将 1101.11B 从小数点开始向左、向右按 4 位分组，不足 4 位的用 0 补足，分组如下：

1101.1100

将每 4 位二进制数写成对应的 1 位十六进制数。转换结果为：

1101.11B＝D.CH

例 2 已知：X＝－11011B，Y＝＋11101B，求$[X+Y]_{补}$＝？，X＋Y＝？$[X-Y]_{补}$＝？，X－Y＝？（设机器为 8 位字长）。

【分析与解答】 此题的运算步骤如下：

①求出$[X]_{原}$，$[Y]_{原}$

$[X]_{原}$＝10011011B，$[Y]_{原}$＝00011101B

②求出$[X]_{补}$，$[Y]_{补}$

$[X]_{补}$＝11100101B，$[Y]_{补}$＝00011101B

③求出$[X+Y]_{补}$

$[X+Y]_{补}=[X]_{补}+[Y]_{补}$＝11100101＋00011101＝00000010B

④求出 X＋Y

根据$[X+Y]_{补}$求出 X＋Y。$[X+Y]_{补}$＝00000010B，补码符号位为 0，表示结果为正数，其余 7 位二进制为 X＋Y 的值。所以：

X＋Y＝2

⑤求出$[X-Y]_{补}$

$[X-Y]_{补}=[X]_{补}-[Y]_{补}$＝11100101－00011101＝11001000B

⑥求出 X－Y

由于$[X-Y]_{补}$＝11001000B，符号位为 1，故为负数，其余 7 位二进

制数按位取反且末位加1即可求得X－Y的值。所以：

$$X-Y=-56$$

1.3 课后习题解答

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

【解答】

(1)369

①降幂法

转换为二进制：

369－256＝113　　$a_8=1$

113－128＜0　　$a_7=0$

113－64＝49　　$a_6=1$

49－32＝17　　$a_5=1$

17－16＝1　　$a_4=1$

1－8＜0　　$a_3=0$

1－4＜0　　$a_2=0$

1－2＜0　　$a_1=0$

1－1＝0　　$a_0=1$

所以二进制为：101110001B。

转换为十六进制：

369－256×1＝133　　$a_2=1$

113－7×16＝1　　$a_1=7$

1－1＝0　　$a_0=1$

所以十六进制为：171H。

②除法

转换为二进制：

369/2＝184　　a_0＝1

184/2＝92　　a_1＝0

92/2＝46　　a_2＝0

46/2＝23　　a_3＝0

23/2＝11　　a_4＝1

11/2＝5　　a_5＝1

5/2＝2　　a_6＝1

2/2＝1　　a_7＝0

1/2＝0　　a_8＝1

所以二进制为：101110001B。

转换为十六进制：

369/16＝23　　a_0＝1

23/16＝1　　a_1＝7

1/16＝0　　a_2＝1

所以十六进制为：171H。

(2)10000

①降幂法

转换为二进制：

10000－8192＝1808　　a_{13}＝1

1808－1024＝784　　a_{10}＝1

784－512＝272　　a_9＝1

272－256＝16　　a_6＝1

16－16＝0　　a_4＝1

所以二进制为：10011100010000B。

转换为十六进制：

10000－4096×2＝1808　　a_3＝2

1080－256×7＝16　　a_2＝7

16－16×1＝0　　$a_1=1$

所以十六进制为：2710H。

②除法

转换为二进制：

10000/2＝5000　　$a_0=0$

5000/2＝2500　　$a_1=0$

2500/2＝1250　　$a_2=0$

1250/2＝625　　$a_3=0$

625/2＝312　　$a_4=1$

312/2＝156　　$a_5=0$

156/2＝78　　$a_6=0$

78/2＝39　　$a_7=0$

39/2＝19　　$a_8=1$

19/2＝9　　$a_9=1$

9/2＝4　　$a_{10}=1$

4/2＝2　　$a_{11}=0$

2/2＝1　　$a_{12}=0$

1/2＝0　　$a_{13}=1$

所以二进制为：10011100010000B。

转换为十六进制：

10000/16＝625　　$a_0=0$

625/16＝39　　$a_1=1$

39/16＝2　　$a_2=7$

2/16＝0　　$a_3=2$

所以十六进制为：2710H。

(3)4095

①降幂法

转换为二进制：

4095－2048＝2047　　a_{11}＝1

2047－1024＝1023　　a_{10}＝1

1023－512＝511　　a_9＝1

511－256＝255　　a_8＝1

255－128＝127　　a_7＝1

127－64＝63　　a_6＝1

63－32＝31　　a_5＝1

31－16＝15　　a_4＝1

15－8＝7　　a_3＝1

7－4＝3　　a_2＝1

3－2＝1　　a_1＝1

1－1＝0　　a_0＝1

所以二进制为：111111111111B。

转换为十六进制：

4095－256×15＝255　　a_2＝F

255－16×15＝15　　a_1＝F

15－1×15＝0　　a_0＝F

所以十六进制为：0FFFH。

②除法

转换为二进制：

4095/2＝2047　　a_0＝1

2047/2＝1023　　a_1＝1

1023/2＝511　　a_2＝1

511/2＝255　　a_3＝1

255/2＝127　　a_4＝1

127/2＝63　　a_5＝1

63/2＝31　　a_6＝1

31/2＝15　　a_7＝1

15/2=7　　$a_8=1$

7/2=3　　$a_8=1$

3/2=1　　$a_{10}=1$

1/2=0　　$a_{11}=1$

所以二进制为:111111111111B。

转换为十六进制:

4095/16=225　　$a_0=F$

255/16=15　　$a_1=F$

15/16=0　　$a_2=F$

所以十六进制为:0FFFH。

(4)32767

①降幂法

转换为二进制:

32767−16384=16383　　$a_{14}=1$

16383−8192=8191　　$a_{13}=1$

8191−4096=4095　　$a_{12}=1$

4095−2048=2047　　$a_{11}=1$

2047−1024=1023　　$a_{10}=1$

1023−512=511　　$a_9=1$

511−256=255　　$a_8=1$

255−128=127　　$a_7=1$

127−64=63　　$a_6=1$

63−32=31　　$a_5=1$

31−16=15　　$a_4=1$

15−8=7　　$a_3=1$

7−4=3　　$a_2=1$

3−2=1　　$a_1=1$

1−1=0　　$a_0=1$

所以二进制为：1111111111111B。

转换为十六进制：

32767－4096×7＝4095　a_3＝7

4095－256×15＝255　a_2＝F

255－16×15＝15　a_1＝F

15－1×15＝0　a_0＝F

所以十六进制为：7FFFH。

②除法

转换为二进制：

32767/2＝16383　a_0＝1

16383/2＝8191　a_1＝1

8191/2＝4095　a_2＝1

4095/2＝2047　a_3＝1

2047/2＝1023　a_4＝1

1023/2＝511　a_5＝1

511/2＝255　a_6＝1

255/2＝127　a_7＝1

127/2＝63　a_8＝1

63/2＝31　a_9＝1

31/2＝15　a_{10}＝1

15/2＝7　a_{11}＝1

7/2＝3　a_{12}＝1

3/2＝1　a_{13}＝1

1/2＝0　a_{14}＝1

所以二进制为：1111111111111B。

转换为十六进制：

32767/16＝2047　a_0＝F

2047/16＝127　a_1＝F

127/16=7　　a_2=F

7/16=0　　a_3=7

所以十六进制为:7FFFH。

1.2 将下列二进制数转换为十六进制数和十进制数。

(1)101101　　(2)10000000

(3)1111111111111111　　(4)11111111

【解答】 本题答案如表 1-2 所示。

表 1-2

二进制	十六进制	十进制
1011 01	2D	$1\times2^0+0\times2^1+1\times2^2+1\times2^3+0\times2^4+1\times2^5=45$
1000 0000	80	$1\times2^7=127$
1111 1111 1111 1111	FFFF	$1\times2^{16}-1=65535$
1111 1111	FF	$1\times2^8-1=255$

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

【解答】 本题答案如表 1-3 所示

表 1-3

十六进制	二进制	十进制
FA	11111010	$15\times16^1+10\times16^0=250$
5B	1011011	$5\times16^1+11\times16^0=91$
FFFE	1111111111111110	$15\times16^3+15\times16^2+15\times16^1+14=655$
1234	1001000110100	$1\times16^3+2\times16^2+3\times16^1+4=4660$

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验。

(1)3A+B7　　(2)1234+AF

(3)ABCD−FE　　(4)7AB×6F

【解答】

(1)
$$\begin{array}{r} 3\,A \\ +B\,7 \\ \hline F\,1 \end{array}$$

3AH＝58D

B7H＝183D

F1＝241D

十进制验证
$$\begin{array}{r} 5\,8 \\ +1\,8\,3 \\ \hline 2\,4\,1 \end{array}$$

(2)
$$\begin{array}{r} 1\,2\,3\,4 \\ +\quad A\,F \\ \hline 1\,2\,E\,3 \end{array}$$

1234 H＝4660 D

AFH＝175 D

12E3H＝4835 D

十进制验证
$$\begin{array}{r} 4\,6\,6\,0 \\ +\quad 1\,7\,5 \\ \hline 4\,8\,3\,5 \end{array}$$

(3)
$$\begin{array}{r} A\,B\,C\,D \\ -\quad\;\; F\,E \\ \hline A\,A\,C\,F \end{array}$$

ABCDH＝43981D

FEH＝254D

AACFH＝43727D

十进制验证
$$\begin{array}{r} 4\,3\,9\,8\,1 \\ -\quad 2\,5\,4 \\ \hline 4\,3\,7\,2\,7 \end{array}$$

(4)
$$\begin{array}{r} 7\,A\,B \\ \times\quad 6\,F \\ \hline 7\,3\,0\,5 \\ 2\,E\,0\,2 \\ \hline 3\,5\,3\,2\,5 \end{array}$$

7ABH=1963D

6FH=111D

35325H=217893D

十进制验证 1963×111=217893

1.5 下列各数均为十六进制数，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)

【解答】 计算$[85]_{补}$、$[-85]_{补}$、$[76]_{补}$和$[-76]_{补}$如下：

$[85]_{补}$=01010101B

$[-85]_{补}$=10101011B

$[76]_{补}$=01001100B

$[-76]_{补}$=10110100B

(1)(−85)+76

$$\begin{array}{r} 1\,0\,1\,0\,1\,0\,1\,1 \\ +0\,1\,0\,0\,1\,1\,0\,0 \\ \hline 1\,1\,1\,1\,0\,1\,1\,1 \end{array}$$

十六进制：0F7 H

(2)85+(−76)

$$\begin{array}{r} 0\,1\,0\,1\,0\,1\,0\,1 \\ +1\,0\,1\,1\,0\,1\,0\,0 \\ \hline 0\,0\,0\,0\,1\,0\,0\,1 \end{array}$$

十六进制：09H

(3)85−76=85+(−76)

运算同(2)。

(4)85−(−76)=85+76

$$\begin{array}{r} 0\,1\,0\,1\,0\,1\,0\,1 \\ +0\,1\,0\,0\,1\,1\,0\,0 \\ \hline 1\,0\,1\,0\,0\,0\,0\,1 \end{array}$$

十六进制：0A1H

(5)－85－76＝－85＋(－76)

```
  10101011
+10110100
  01011111
```

十六进制：5FH

(6)－85－(－76)＝－85＋76

```
  10101011
+01001100
  11110111
```

十六进制：0F7H

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

【解答】

(1)1101 1000

带符号数为：－40

不带符号数为：216

(2)1111 1111

带符号数为：－1

不带符号数为：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

【解答】 本题答案如表 1-4 所示。

表 1-4

二进制数	十进制数	字　符
4F	79	O(大写)
2B	43	＋
73	115	s(小写)
59	89	Y

1.8 请写出下列字符串的 ASCII 码值。

For example,

This is a number 3692.

【解答】 此符号串的 ASCII 码值为：

46	6F	72	20	65	78	61	6D
70	6C	65	2C	0A	0D	54	68
69	73	20	69	73	20	61	20
6E	75	6D	62	65	72	20	33
36	39	32	2E				

1.4 强化训练

1. 给出 n 位二进制整数的表示范围。

2. 请在表 1-5 中的空白处填入各数的真值、原码、反码、补码(采用 8 位二进制，最左 1 位为符号位)。

表 1-5

真值 x(十进制)	真值 x(二进制)	$[x]_{原}$	$[x]_{反}$	$[x]_{补}$
−128				
−127				
−1				
−0				
+0				
+1				
127				

3. 已知 x、y，求$[x-y]_{补}$=？ 并求 x−y=？

(1)x=+24D　　y=−64D

(2)x=−1010111 B　　y=+101 0101B

4. 写出下列十六进制数，在被看作是有符号数和无符号数的十进制值分别是多少？

(1)FE　　(2)85

(3)2C　　(4)57

答　案

1. n 位二进制整数的表示范围如下：

(1)无符号整数的范围

8 位二进制数所能表示的无符号整数的范围是 0～255

16 位二进制数所能表示的无符号整数的范围是 0～65 535

n 位二进制数 N 能够表示的无符号数的范围是 $0 \leqslant N \leqslant 2^{n}-1$

(2)采用补码表示的带符号整数的表示范围

8 位二进制数所能表示的带符号整数的范围是－128～127

16 位二进制数所能表示的带符号整数的范围是－32 768～＋3 2767

n 位二进制数 N 能够表示的带符号数的范围是 $-2^{(n-1)} \leqslant N \leqslant 2^{n-1}-1$

2. 各数的真值、原码、反码、补码如表 1-6 所示：

表 1-6

真值 x(十进制)	真值 x(二进制)	$[x]_{原}$	$[x]_{反}$	$[x]_{补}$
－128	－10000000	不存在	不存在	10000000
－127	－01111111	11111111	10000000	10000001
－1	－00000001	10000001	11111110	11111111
－0	－00000000	10000000	11111111	00000000
＋0	00000000	00000000	00000000	00000000
＋1	＋00000001	00000001	00000001	00000001
＋127	＋01111111	01111111	01111111	01111111

3. (1)$[x-y]_{补}=6AH$　　$x-y=106$

   (2)$[x-y]_{补}=0FEH$　　$x-y=-2$

4. (1)254，−2

   (2)85，−42

   (3)45，45

   (4)57，57

第2章　80x86计算机组织

2.1　内容提要

2.1.1　计算机系统组成

一台完整的计算机系统由硬件系统和软件系统两大部分组成。

1.硬件系统结构

硬件系统由运算器、控制器、存储器、输入设备和输出设备五大部分组成，这五个部分之间既有分工又有合作，共同完成处理任务。计算机所处理的数据都以二进制形式进行存储和处理，描述反映数据存储位置的地址也是二进制数。计算机以运算器和控制器（合称为中央处理单元）为中心，在控制器的作用下从存储器中依次读出指令和数据，在运算器中执行相应操作，最终结果也在控制器的作用下写回存储器中。这一过程是反复取指令、指令译码和执行指令的过程。

学习汇编语言应了解的硬件系统术语有：

(1)运算器(ALU)

运算器又称为算术逻辑单元，用来进行算术或逻辑运算以及移位循环等操作。

(2)控制器(CU)

控制器负责统一指挥计算机各部分协调工作，能根据事先安排好的指令发出各种控制信号，以控制计算机各个部分的工作。控制器包

括指令寄存器、指令译码器和可编程逻辑阵列共3部分。

(3)存储器(Memory)

存储器是计算机的记忆部件,负责存储处理器要运行的程序和数据。

(4)输入、输出设备(I/O)

输入、输出设备一般包括一些I/O设备和存储器两类。

2.软件系统结构

软件是为了运行、管理和维护计算机而编制的各种程序的总和。

计算机软件主要分为应用软件和系统软件两大类。

应用软件是用户为解决科学计算、办公自动化、检测与实时控制等不同领域的任务所编制的应用程序。

系统软件是指不需用户干预的能生成、准备和执行其他程序所需的一组程序。常见的操作系统就属于系统软件。

2.1.2 8086/8088CPU的寄存器组

由于8086与8088的指令相同,CPU的编程结构相同(如图2.1所示)。因此从汇编语言的角度来看,8086和8088微处理器的组成是相同的。

①数据寄存器包括AX,BX,CX,DX。它们用来临时存放计算过程中所用到的操作数、结果或其他信息。它们以字或字节形式访问。

②指针及变址寄存器包括SP,BP,SI,DI。它们可以存放运算过程中的数据,但只能以字的形式访问。它们的常用用途是存放操作数在段内的偏移地址。

③标志寄存器PSW:标志寄存器PSW存有16位,其中7位未用。

标志寄存器分为有条件标志和控制标志。

条件标志的值取决于一个操作完成后,运算逻辑部件ALU所处的状态。条件标志有OF、SF、ZF、CF、AF和PF共6个。

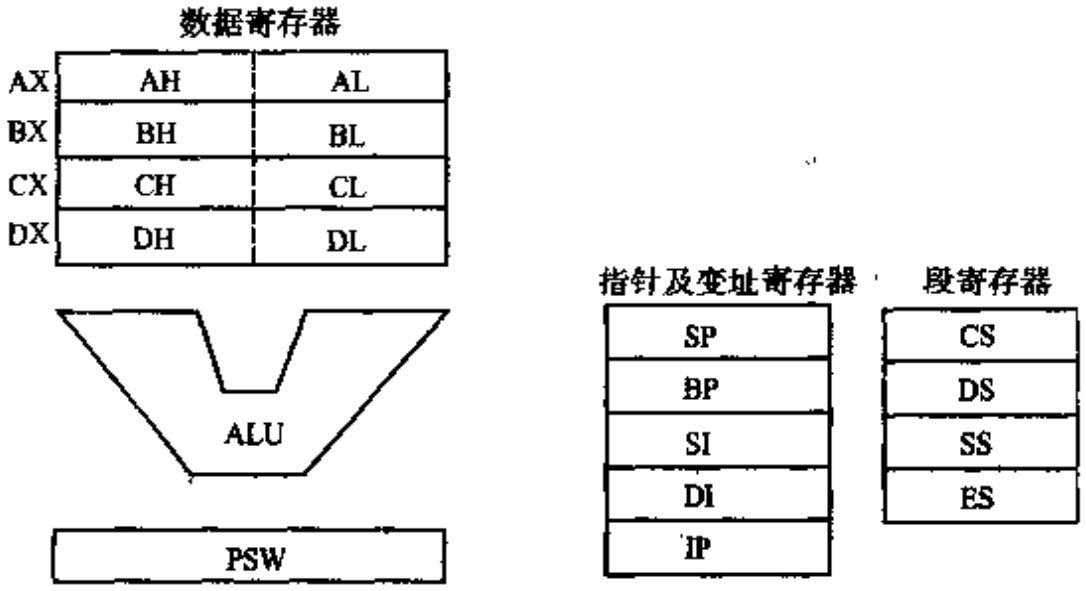


图 2.1 CPU 的组成

OF 溢出标志:所谓溢出是指字节运算时结果超过了范围－128～＋127,字运算时结果超过了范围－32 768～＋32 767。溢出时标志 OF 为 1,否则 OF 为 0。

SF 符号标志:SF 的值与运算结果的最高位相同。

ZF 零标志:运算结果为零时,ZF 为 1,否则,ZF 为 0。

CF 进位标志:在运算中,若最高有效位产生进位或借位,则 CF 为 1,否则,CF 为 0。

AF 辅助进位标志:记录运算中,低半字节向高半字节的进位情况。若有进位,AF=1,否则,AF=0。

PF 奇偶标志:若运算结果的低 8 位中,"1"的个数为偶数,则 PF 为 1,否则,PF 为 0。

控制标志的值是通过指令人为设置的,以控制程序的执行。控制标志有 DF、IF、TF 三个。

DF 方向标志:用于控制串操作。DF 为 1 时,串操作后使变址寄存器 SI、DI 减量;DF 为 0 时,串操作后使 SI,DI 增量。

IF 中断标志:IF 为 1 时,允许中断;IF 为 0 时,禁止中断。

TF 陷阱标志：TF 为 1 时，CPU 处于单步运行方式；TF 为 0 时，CPU 处于正常工作方式。

④16 位指令指针寄存器 IP：IP 为指令指针寄存器，它用来存放代码段中的偏移地址。

⑤段寄存器：每个段寄存器用于确定一个段的起始地址，这些段各有各的用途。

CS——16 位代码段寄存器。

DS——16 位数据段寄存器。

ES——16 位附加数据段寄存器。

SS——16 位堆栈段寄存器。

各段寄存器使用的一些基本约定如表 2-1 所示。

表 2-1

访问存储器类型	默认段寄存器	可指定段寄存器	段内偏移地址来源
指令寻址	CS	无	IP
堆栈寻址	SS	无	SP
串操作源地址	DS	CS、ES、SS	SI、DI、BX 或 16 位数
串操作目的地址	ES	无	DI
BP 用作基址寄存器	SS	CS、DS、ES	以寻址方式求得有效地址
一般数据存取	DS	CS、ES、SS	以寻址方式求得有效地址

2.1.3 80x86 系统微处理器的寄存器结构

80x86 微处理器的内部寄存器结构如图 2.2 所示。

①数据寄存器

数据寄存器包括 EAX、EBX、ECX 和 EDX 四个，它们除了可以作为一般数据寄存器用来存放和传递操作数信息或结果外，还各有一些专门的用途。

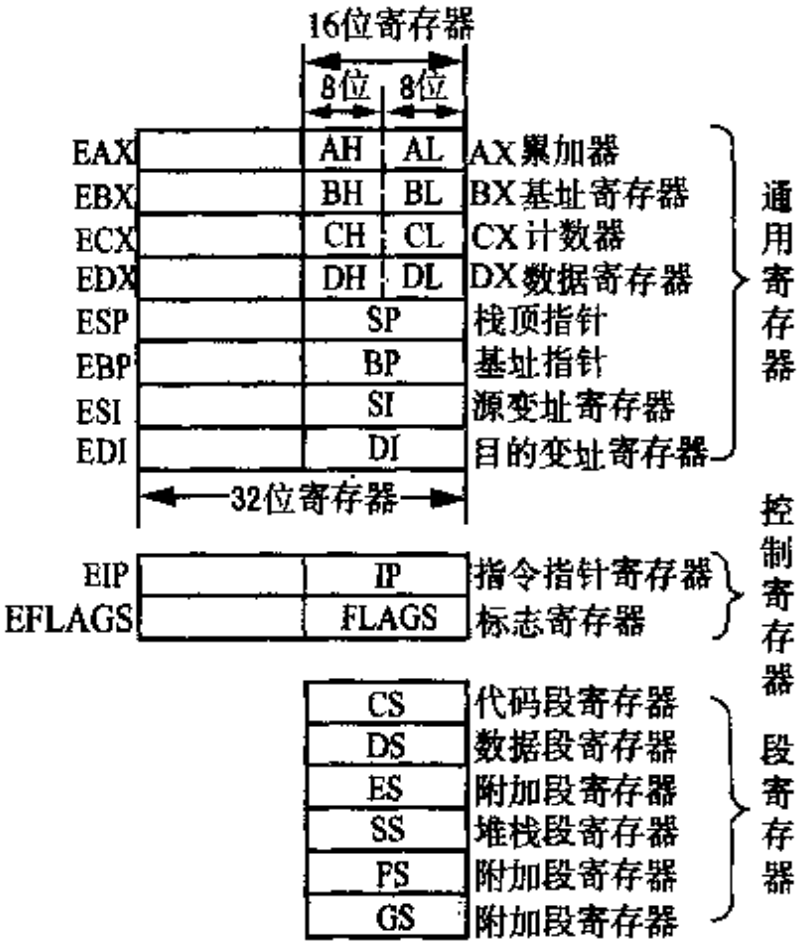


图 2.2　80x86 微处理器的内部寄存器结构

②指针寄存器

栈顶指针寄存器 ESP 和基址指针寄存器 EBP 合称为指针寄存器。ESP 是指向堆栈区的栈顶指针，用来存放栈顶的偏移地址。EBP 是指针寄存器，用于存放堆栈段中某一存储单元的偏移地址指向堆栈段的基地址。

③变址寄存器

ESI 和 EDI 均为变址寄存器。

④指令指针寄存器和状态标志寄存器

指令指针寄存器 EIP 指向程序的下一条指令，并由转移和调用指令进行修改。

状态标志寄存器用来反映一些算术或逻辑运算结果标志和处理器的状态。

⑤段寄存器

80286之前的CPU只有四个段：CS、DS、SS、ES，80286以后增加了FS和GS两个段寄存器。

2.1.4 IBM－PC的存储器

1.存储单元的地址和内容

存储器由许多存储单元组成，存储单元以字节为单位进行组织的。为了方便存取每一个储存单元，每个存储单元规定一个编号，这就是存储单元地址。

在计算机中，存储单元地址以二进制形式表示。常用十六进制数来表示存储单元的地址。

一个存储单元中存放的信息称为该存储单元的内容。

2.实模式下物理地址的形成

在实模式方式下，物理地址的计算采用段加偏移的方法，计算公式为：

物理地址＝段地址×16d＋储移地址

即将16的地址左移4位再加上16的储移地址，从而得到20位的物理地址。

2.1.5 考 点

1.计算机系统的组成；

2.80x86 CPU的寄存器组，标志寄存器各位标志的意义及应用；

3.存储器地址的分段表示及物理地址的计算。

2.2 例题精解

例 1 已知两个数 m＝00111011B，n＝01001010B，完成下列运算，并给出运算后 SF、ZF、PF、CP、AF、OF 标志位的状态。

(1)m＋n　　(2)m－n　　(3)n－m

【分析与解答】

(1)m＋n＝10000101B

00111011						
＋01001010	SF	ZF	PF	CF	AF	OF
10000101	1	0	0	0	1	1

(2)m－n＝11110000B

00111011						
－01001010	SF	ZF	PF	CF	AF	OF
11110000	1	0	0	1	0	0

(3)n－m＝00001111B

01001010						
－00111011	SF	ZF	PF	CF	AF	OF
00001111	0	0	1	0	1	0

例 2 计算在实模式下，如下 CS:IP 组合寻址的物理地址。

(1)CS＝2000H　　IP＝0200H

(2)CS＝40FAH　　IP＝2A00H

【分析与解答】

(1)物理地址＝20000H＋0200H＝20200H

(2)物理地址＝40FA0H＋2A00H＝31080H

例 3 假设用以下寄存器组合来访问存储单元，试求出它们所访问单元的物理地址。

(1)DS＝1000H 和 DI＝2000H

(2)DS＝2000H 和 SI＝1002H

(3)SS=2300H 和 BP=3200H

(4)DS=A000H 和 BX=1000H

(5)SS=2900H 和 SP=3A00H

【分析与解答】

(1)物理地址=10000H+2000H=12000H

(2)物理地址=20000H+1002H=21002H

(3)物理地址=23000H+3200H=26200H

(4)物理地址=A0000H+1000H=A1000H

(5)物理地址=29000H+3A00H=2CA00H

2.3 课后习题解答

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。

【解答】 可直接由指令指定的I/O端口数为$2^8=256$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。

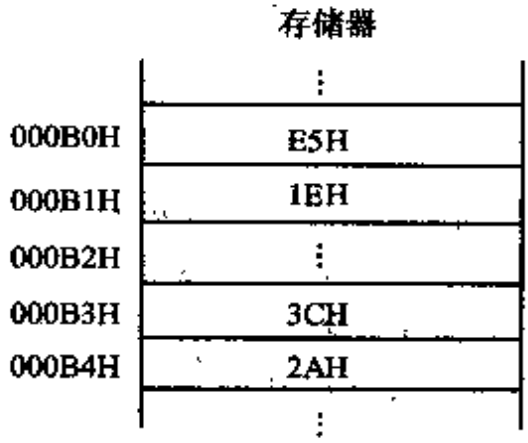


图 2.3

【解答】 两个 16 位字在存储器里的存放情况如图 2.3 所示。

2.3 80x86 微机的存储器中存放信息如图 2.4 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。

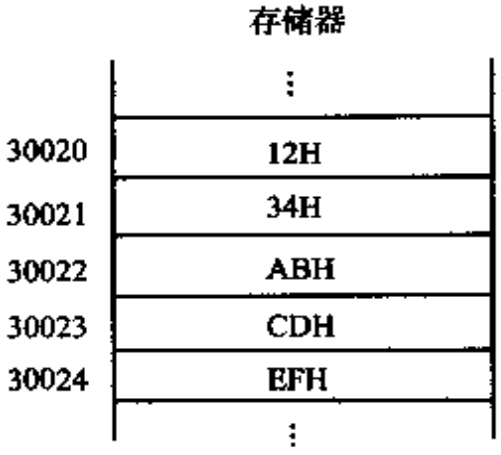


图 2.4 2.3 题的信息存放情况

【解答】

$(30022H)_b$＝0ABH

$(30024H)_b$＝0EFH

$(30021H)_w$＝0AB34H

$(30022H)_w$＝0CDABH

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？

【解答】 (1)3017：000A 的物理地址是：3017AH；

(2)3015：002A 的物理地址是：3017AH；

(3)3010：007A 的物理地址是：3017AH。

2.5 如果在一个程序开始执行以前(CS)＝0A7F0H(如十六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？

【解答】 物理地址＝(CS) * 10H＋(IP)＝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
```

【解答】 存储器的分段示意图如图2.5所示。

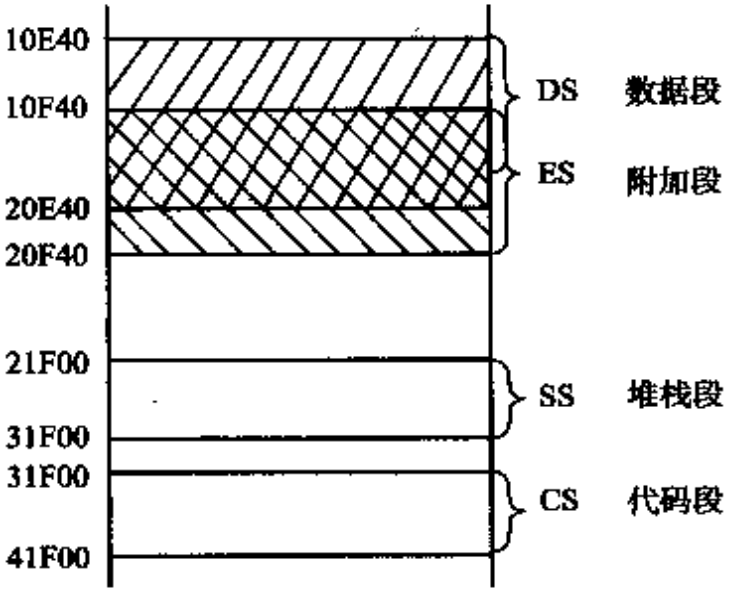


图2.5 存储器的分段示意图

OF、SF、ZF、CF都为0。

2.7 下列操作可使用哪些寄存器？

(1)加法和减法。

(2)循环计数。

(3)乘法和除法。

(4)保存段地址。

(5)表示运算结果为 0。

(6)将要执行的指令地址。

(7)将要从堆栈取出数据的地址。

【解答】

(1)所有寄存器

(2)CX

(3)AX、DX

(4)SI、DI、SP、BP

(5)FLAG

(6)CS、IP

(7)SS、BP、SP

2.8 哪些寄存器可以用来指示存储器地址?

【解答】 DS、ES、SS、CS、BX、SI、DI、SP、BP、IP 均可以用来指示存储器地址。

2.9 请将下列左边的项和右边的解释联系起来(把所选的字母写在括号内):

(1)CPU	( )	A. 保存当前栈顶地址的寄存器
(2)存储器	( )	B. 指示下一条要执行的指令的地址
(3)堆栈	( )	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	( )	D. 以后进先出方式工作的存储空间
(5)SP	( )	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	( )	F. 唯一代表存储空间中每个字节单元的地址
(7)控制标志	( )	G. 能被计算机直接识别的语言
(8)段寄存器	( )	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 ( ) 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 ( ) J. 保存各逻辑段的起始地址的寄存器，8086/8088 机有 4 个：CS、DS、SS、ES

(11)机器语言 ( ) K. 控制操作的标志，如 DF 位

(12)汇编程序 ( ) L. 记录指令操作结果的标志，共 6 位：OF，SF，ZF，AF，PF 和 CF

(13)连接程序 ( ) M. 分析、控制并执行指令的部件，由算术逻辑部件 ALU 和寄存器组等组成

(14)指令 ( ) N. 由汇编程序在汇编过程中执行的指令

(15)伪指令 ( ) 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

【解答】 (1)－M (2)－C (3)－D (4)－B (5)－A (6)－L (7)－K (8)－J (9)－F (10)－H (11)－G (12)－E (13)－I (14)－O (15)－N

2.4 强化训练

1. 将十六进制数 2AE0H 与下列各数相加，给出结果及各条件标志的状态(字长为 16 位)。

(1)F9B8H (2)58CFH

2. 计算实模式下，如下寄存器组合访问单元的物理地址。

(1)DS＝3200H，ESI＝00000200H

(2)SS＝2000H，ESP＝00005F00H

(3)DS＝5600H，EDX＝00002A00H

(4)DS＝45A0H，EBX＝0000A3F8H

3. 有两个字 1234H 和 5678H 分别存放在 80x86 微机存储器的 000B0H 和 000A3H 单元中，请用图表示出它们在存储器里的存放情况。

答　案

1. 运算结果及各标志位的状态如表 2-2 所示。

表 2-2

运　算	结果	SF	OF	AF	ZF	PF	CF
2AE0H＋0F9B8H	2498H	0	0	0	0	0	1
2AE0H＋58CFH	83AFH	1	1	0	0	1	0

2. (1)32200H
   (2)25F00H
   (3)58A00H
   (4)4FDF0H

3. 存储情况如图 2.6 所示。

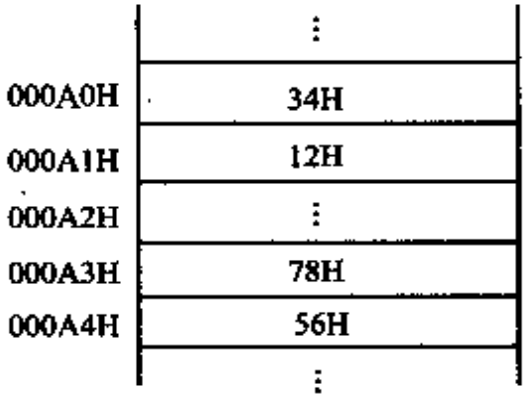


图 2.6

第3章 80x86的指令系统和寻址方式

3.1 内容提要

3.1.1 计算机的一般指令格式

计算机所能执行的所有指令的集合就是指令系统。

指令是告诉计算机如何处理数据的命令。一条指令应该包括以下内容：指令执行的操作、操作数的来源、操作结果的去向。

计算机常用的指令格式分为以下几种：

(1)零地址指令

格式：指令操作码

零地址指令是指指令中操作数地址的个数为0个，即指令中不提供操作数的地址。

一般有如下两种情况：

①指令无需任何操作数。例如：空操作指令NOP，停机指令HLT。

②所需的操作数是隐含约定的。例如：子程序返回指令RET，串传送指令MOVSW。

(2)一地址指令

格式：

指令操作码	操作数

一地址指令是指指令中操作数地址的个数为 1 个，即指令中只提供一个操作数的地址。

①只有目的操作数的单操作数指令。例如：DEC　BX 等指令。

②隐含约定目的地的双操作数指令。例如：PUSH　AX 等指令。

(3)两地址指令

格式：

指令操作码	目的操作数	源操作数

两地址指令是指指令中操作数地址的个数为 2 个。

指令中指出参加操作的 2 个操作数的地址，操作后的结果存放在目的操作数地址中(目的操作数不需要保留的情况)。例如：

ADD　CX,AX　　这条指令完成 CX＋AX→CX

SUB　DX,10H　　这条指令完成 DX－10H→DX

以上介绍的 3 种指令格式中，两地址指令是微机中最常见的指令格式。

3.1.2　80x86 寻址方式

1. 与数据有关的寻址方式(见表 3-1)

表 3-1

寻址方式	定义	有效地址的计算方法	举例
立即寻址方式	操作数紧跟在指令操作码之后，它作为操作码一起存放在存储器的代码段中		MOV AX,2008H 2008H→AX 源操作数 2008H 为立即寻址方式

（续表）

寻址方式	定义	有效地址的计算方法	举例
寄存器寻址方式	操作数在寄存器中指令指定寄存器号		MOV AX,CX CX→AX 源操作数 CX 和目的操作数 AX 均为寄存器寻址方式
直接寻址方式	操作数存放在内存单元中，指令中给出操作数在存储器数据段中存放的偏移地址	PA＝DS×16＋位移量 PA＝ES×16＋位移量	MOV AX,[2006H] [DS:2006H]→AX 源操作数[2006H]为直接寻址方式
寄存器间接寻址方式	操作数存放在存储器的内存单元中，操作数的偏移地址由指令指定的寄存器 BX，BP，SI 或 DI 给出	PA＝DS×16＋{BX / SI / DI} PA＝SS×16＋BP	MOV AX,[BP] (SS×16＋BP)→CX (DS×16＋DI)→CX 源操作数[BP]、[DI]为寄存器间接寻址方式

寄存器相对寻址方式	操作数存放在存储器的内存单元中，操作数的偏移地址由指令指定的寄存器 BX，BP，SI，DI 和指令中给定的位移量相加	PA＝DS×16＋{BX / SI / DI}＋位移量
		PA＝SS×16＋BP＋位移量
			MOV AX,AAA[BX]
			MOV BX,BBB[BP]
			源操作数 AAA[BX]和 BBB[BP]为寄存器相对寻址方式

（续表）

寻址方式	定义	有效地址的计算方法	举例

基址变址寻址方式	操作数的偏移地址由指令指定的基址寄存器（BX，BP）和变址寄存器（SI，DI）内容相加	PA＝DS×16＋BX ＋{SI / DI}
		PA＝SS×16＋BP ＋{SI / DI}
			MOV CX,[BP][DI]
			MOV AX,[BX][SI]
			源操作数[BP][DI]和[BX][SI]为基址变址寻址方式。以上两条指令也可写成：
			MOV CX,[BP+DI]
			MOV AX,[BX+SI]

相对基址变址寻址方式	操作数存放在存储器的内存单元中，操作数的偏移地址由指令指定的基址寄存器（BX，BP）和变址寄存器（SI，DI）以及指令中给定的位移量相加	PA＝DS×16＋BX ＋{SI / DI}＋位移量 PA＝SS×16＋BP ＋{SI / DI}＋位移量	MOV AX，AAA[BP][DI] 源操作数AAA[BP][DI]为相对基址加变址寻址方式。以上语句也可写成： MOV AX，[BP＋DI＋AAA]
比例变址寻址方式	操作数存放在存储器的内存单元中，操作数的偏移地址为变址寄存器的内容乘指令中指定的比例因子以及指令中给定的偏移量相加	与寄存器相对寻址相比，只是增加了比例因子	MOV EAX，AAA[ESI＊8] 源操作数AAA[ESI＊8]为比例变址寻址方式

（续表）

寻址方式	定义	有效地址的计算方法	举例
基址比例变址寻址方式	操作数存放在存储器的内存单元中，操作数的偏移地址为变址寄存器的内容乘比例因子，加上基址寄存器的内容，再加上指令中给定的位移量	与基址变址寻址方式相比，只是增加了比例因子	MOV EAX,[EBX][ECX * 4] 源操作数[EBX][ECX * 4]为比例变址寻址方式
相对基址比例变址寻址方式	操作数存放在存储器的内存单元中，操作数的偏移地址为变址寄存器的内容乘以比例因子，加上基址寄存器的内容，再加上位移量之和	与相对基址变址寻址方式相比，只是增加了比例因子	MOV EBX, BBB[EBP][ESI * 8] 源操作数 BBB[EBP][ESI * 8]为相对基址比例变址寻址方式

2. 与转移地址有关的寻址方式

这种寻址方式用来确定转移指令及 CALL 指令的转向地址。

(1)段内直接寻址

转向的有效地址是当前 IP 寄存器的内容和指令的 8 位或 16 位位移量之和。

指令的汇编语言格式表示为：

JMP NEAR PTR PROGIA

JMP SHORT QUEST

(2)段内间接寻址

转向的有效地址是一个寄存器或是一个存储单元的内容。这个寄存器或存储单元的内容可以用数据寻址方式中除立即数以外的任何一种寻址方式取得，所得到的转向的有效地址用来取代 IP 寄存器的内容。

指令的汇编格式可以表示为：

```
JMP   BX
JMP   WORD PTR[BP+TABLE]
```

(3)段间直接寻址

在指令中直接提供了转向段地址和偏移地址，所以只要用指令的偏移地址取代 IP 寄存器的内容，用指令中指定的段地址取代 CS 寄存器的内容就完成了从一个段到另一个段的转移操作。

指令的汇编语言格式可表示为：

```
JMP   FAR PTR NEXTROUTINT
```

(4)段间间接寻址

用存储器中的两个相继字的内容来取代 IP 和 CS 寄存器中的原始内容，以达到段间转移的目的。这里，存储单元的地址是指令指定除立即数方式和寄存器方式以外的任何一种数据寻址方式取得。

指令的汇编语言格式可表示为：

```
JMP   DWORD PTR [INTERS+BX]
```

3.1.3 80x86 微处理器的指令系统

1. 数据传送指令

(1)通用数据传送指令

MOV	传送
MOVSX	带符号扩展传送
MOVZX	带零扩展传送

PUSH	进栈
POP	出栈
PUSHA/PUSHAD	所有寄存器进栈
POPA/POPAD	所有寄存器出栈
XCHG	交换

①MOV 传送指令

格式为:MOV　DST,SRC

执行操作:(DST)←(SRC)

其中 DST 表示目的操作数,SRC 表示源操作数。

MOV 指令可以分以下 4 种情况:

(ⅰ)寄存器与寄存器之间的数据传送指令

代码段寄存器 CS 及指令指针 IP 不参与数的传送,其中 CS 可以作为源操作数参与传送,但不能作为目的操作数参与传送。

(ⅱ)立即数到通用寄存器的数据传送指令

立即数只能作为源操作数使用,不能作为目的操作数使用。

(ⅲ)寄存器与存储器之间的数据传送指令。

(ⅳ)立即数到存储器的数据传送。

使用 MOV 指令传送数据,必须注意以下几点:

(ⅰ)立即数只能作为源操作数,不允许作目的操作数,也不能送至段寄存器。

(ⅱ)通用寄存器可以与段寄存器、存储器互相传送数据,寄存器之间也可以互相传送。但是 CS 不能作为目的操作数,IP 不能参加数据传送。

(ⅲ)存储器与存储器之间不能进行数据直接传送。若要实现存储单元之间的数据传送,可以借助于通用寄存器作为中介来进行。

②MOVSX 带符号扩展传送指令

格式为:MOVSX　DST,SRC

执行操作：(DST)←符号扩展(SRC)

可以有两种格式：

MOVSX　reg1,reg2

MOVSX　reg,mem

MOVSX 不影响标志位。

③MOVZX 带零扩展传送指令

格式为：MOVZX　DST,SRC

执行操作：(DST)←零扩展(SRC)

本指令可以有两种格式：

MOVZX　reg1,reg2

MOVZX　reg,mem

④PUSH 进栈指令

格式为：PUSH　　SRC

执行操作：

16 位指令：(SP)←(SP)－2

((SP)＋1,(SP))←(SRC)

32 位指令：(ESP)←(ESP)－4

((ESP)＋3,(ESP)＋2,(ESP)＋1,(ESP))←(SRC)

PUSH 指令允许的格式有：

PUSH　　reg

PUSH　　mem

PUSH　　data

PUSH　　segreg

⑤POP 出栈指令

格式为：POP　　DST

执行操作：

16 位指令：(DST)←((SP)＋1,(SP))

(SP)←(SP)＋2

32 位指令：(DST)←((ESP)+3,(ESP)+2,(ESP)+1,(ESP))
(ESP)←(ESP)+4

POP 指令允许的格式有：

POP　　reg

POP　　mem

POP　　segreg

PUSH 和 POP 指令只能作字或双字操作。

PUSH 和 POP 指令均不影响标志位。

⑥PUSHA/PUSHAD　所有寄存器进栈指令

格式为：PUSHA

PUSHAD

执行操作：

PUSHA：16 位通用寄存器依次进栈，进栈次序为：AX，CX，DX，BX，指令执行前的 SP，BP，SI，DI。指令执行后(SP)←(SP)－16 仍指向栈顶。

PUSHAD：32 位通用寄存器依次进栈，进栈次序为：EAX，ECX，EDX，EBX，指令执行前的 ESP，EBP，ESI，EDI。指令执行后(SP)←(SP)－32。

⑦POPA/POPAD 所有寄存器出栈指令

格式为：POPA

POPAD

执行操作：

POPA：16 位通用寄存器依次进栈，进栈次序为：DI，SI，BP，SP，BX，DX，CX，AX，指令执行后(SP)←(SP)＋16 仍指向栈顶。

POPAD：32 位通用寄存器依次进栈，进栈次序为：EDI，ESI，EBP，ESP，EBX，EDX，ECX，EAX。指令执行后(ESP)←(ESP)＋32 仍指向栈顶。

上述两条堆栈指令均不影响标志位。

⑧XCHG 交换指令

格式为：XCHG　DPR1，OPR2

执行操作：(OPR1)↔(OPR2)

其中 OPR 表示操作数。该指令的两个操作数中必须有一个在寄存器中，因此它可以在寄存器之间或者在寄存器与存储器之间交换信息，但不允许使用段寄存器。

(2)累加器专用传送指令

IN	输入
OUT	输出

输入、输出指令不影响标志位。

在 80x86 里，所有 I/O 端口与 CPU 之间的通信都由 IN 和 OUT 指令来完成。其中 IN 完成从 I/O 到 CPU 的信息传送，而 OUT 则完成从 CPU 到 I/O 端口的信息传送。CPU 只能用累加器(AL 或 AX 或 EAX)接收或发送信息。

(3)地址传送指令

LEA	有效地址送寄存器
LDS	指针送寄存器和 DS
LES	指针送寄存器和 ES
LFS	指针送寄存器和 FS
LGS	指针送寄存器和 GS
LSS	指针送寄存器和 SS

(4)标志寄存器传送指令

LAHF	标志送 AH
SAHF	AH 送标志寄存器
PUSHF/PUSHFD	标志进栈
POPF/POPFD	标志出栈

(5)类型转换指令

CBW	字节转换为字

CWD/CWDE	字转换为双字
CDQ	双字转换为 4 字
BSWAP	字节交换

2. 算术指令

(1)加法指令

ADD	加法
ADC	带进位加法
INC	加 1
XADD	交换并相加

执行加法指令时,CF 位是根据最高有效位是否有向高位的进位设置的。有进位时 CF=1,无进位时 CF=0。OF 位则根据操作数的符号及其变化情况来设置:若两个操作数的符号相同,而结果的符号与之相反时 OF=1,否则 OF=0。

(2)减法指令

SUB	减法
SBB	带借位减法
DEC	减 1
NEG	求补
CMP	比较
CMPXCHG	比较并交换
CMPXCHG8B	比较并交换 8 字节

当作无符号数运算时,若减数>被减数,此时有借位,则 CF=1,否则 CF=0。

减法的 OF 位的设置方法为:若两个数的符号相反,而结果的符号与减数相同,则 OF=1;除上述情况外 OF=0。

(3)乘法指令

MUL	无符号数乘法
IMUL	带符号数乘法

在乘法指令里，目的操作数必须是累加器，字运算为 AX，字节运算为 AL。两个 8 位数相乘得到的是 16 位乘积存放在 AX 中，两个 16 位数相乘得到的是 32 位乘积，存放在 DX，AX 中。其中 DX 存放高位字，AX 存放低位字。

(4)除法指令

DIV	无符号数除法
IDIV	带符号数除法

除法指令的寻址方式和乘法指令相同。其目的操作数必须存放在 AX 或 DX 或 EDX，EAX 中；而其源操作数可以用除立即数以外的任一种寻址方式。

除法指令对所有条件码位均无定义。

3. 逻辑指令

(1)逻辑运算指令

AND	逻辑与
OR	逻辑或
NOT	逻辑非
XOR	异或
TEST	测试

(2)位测试并修改指令

BT	位测试
BTS	位测试并置 1
BTR	位测试并置 0
BTC	位测试并变反

(3)位扫描指令

386 及其后继机型增加了本组指令。

BSF	正向位扫描
BSR	反向位扫描

(4)移位指令

SHL	逻辑左移
SAL	算术左移
SHR	逻辑右移
SAR	算术右移
ROL	循环左移
ROR	循环右移
RCL	带进位循环左移
RCR	带进位循环右移
SHLD	双精度左移
SHRD	双精度右移

4. 串处理指令

MOVS	串传送
CMPS	串比较
SCAS	串扫描
LODS	从串取
STOS	存入串
INS	串输入
OUTS	串输出

与上述基本指令配合使用的前缀有：

REP	重复
REPE/REPZ	相等/为零则重复
REPNE/REPNZ	不相等/不为零则重复

在执行串处理指令前，应该先做好以下准备工作：

(ⅰ)把存放在数据段中的源串首地址(如反向传送则应是末地址)放入源变址寄存器中；

(ⅱ)把将要存放数据串的附加段中的目的串首地址(或反向传送时的末地址)放入目的变址寄存器中；

(ⅲ)把数据串长度放入计数寄存器；

(ⅳ)建立方向标志。

其中,建立方向标志,有两条指令:

(ⅰ)CLD 该指令使 DF=0,在执行串处理指令时可使地址自动增量;

(ⅱ)STD 该指令使 DF=1,在执行串处理指令时可使地址自动减量。

5. 控制转移指令

(1)无条件转移指令

JMP 跳转指令。

无条件地转移到指令指定的地址去执行从该地址开始的指令。

(2)条件转移指令

满足测试条件则转移到由指令指定的转向地址去执行那里的程序。

JZ(或 JE)结果为零(或相等)则转移。

JNZ(或 JNE)结果不为零(或不相等)则转移。

JS 结果为负则转移。

JNS 结果为正则转移。

JO 溢出则转移。

JNO 不溢出则转移。

JP(或 JPE)奇偶位为 1 则转移。

JNP(或 JPO)奇偶位为 0 则转移。

JB(或 JNAE,或 JC)低于,或者不高于或等于,或进位为 1 则转移。

JNB(或 JAE,或 JNC)不低于,或者高于或等于,或进位为零则转移。

(3)比较两个无符号数,并根据比较的结果转移

JB(或 JNAE,或 JC)低于,或者不高于或等于,或进位位为 1

则转移。

JNB(或 JAE,或 JNC)不低于,或者高于或等于,或进位位为 0 则转移。

JBE(或 JNA)低于或等于,或不高于则转移。

JNBE(或 JA)不低于或等于,或高于则转移。

(4)比较两个带符号数,并根据比较结果转移

JL(或 JNGE)小于,或者不大于或等于则转移。

JNL(或 JGE)不小于,或者大于或等于则转移。

JLE(或 JNG)小于或等于,或者不大于则转移。

JNLE(或 JG)不大于或等于,或者大于则转移。

6. 循环指令

LOOP 循环。

LOOPZ/LOOPE 当为零或相等时循环。

LOOPNZ/LOOPNE 当不为零或不相等时循环。

3.1.4 考 点

1. 80x86 CPU 的寻址方式;

2. 80x86 CPU 指令系统及其常用指令的功能。

3.2 例题精解

例 1 判断下列指令的正误,如果错误请说明原因。

(1)MOV AL,CX

(2)MOV ES,DS

(3)MOV [BX],[DI]

(4)MOV AL,[BX],[SI]

(5)MOV DS,DX

(6)MOV CS,AX

(7)MOV　　BX,CS
(8)MOV　　DS,2008H
(9)MOVSX　DS,AL
(10)XCHG　BX,6
(11)PUSH　CS
(12)MOV　　[SP],BX
(13)MOV　　AX,BX+3
(14)XCHG　ES,AX
(15)IMUL　AX,30H
(16)SHL　　AX,2
(17)CMP　　BX,2008H

【分析与解答】

(1)错误,源操作数和目的操作数长度不匹配。
(2)错误,源操作数和目的操作数不能同时为段寄存器。
(3)错误,MOV 指令不允许从存储单元直接送到存储单元。
(4)正确。
(5)正确。
(6)错误,CS 不能作为目的操作数。
(7)正确。
(8)正确。
(9)错误,MOVSX 指令中不应出现段寄存器。
(10)错误,XCHG 指令不允许出现立即数。
(11)错误,通常不应将 CS 压栈。
(12)正确。
(13)正确。
(14)错误,XCHG 指令使用通用寄存器。
(15)错误,不支持立即数乘法。
(16)错误,移位个数送 CL。

(17)正确。

例 2 指出下列指令中操作数的寻址方式。

(1)MOV COUNT[BX],AL

(2)INC SI

(3)MOV AX,[2008H]

(4)MOV DX,[SI]

(5)MOV AX,2008H

(6)MOV [BP][SI],AX

(7)MOV [BX+100][SI],AX

(8)MOV AX,[BP+50][DI]

(9)MOV [SI],AX

(10)MOV [2008H],AL

【分析与解答】

依据寻址方式的定义确定指令的寻址方式。

(1)目的操作数为寄存器相对寻址,源操作数为寄存器寻址。

(2)单一操作数为寄存器寻址。

(3)目的操作数为寄存器寻址,源操作数为直接寻址。

(4)目的操作数为寄存器寻址,源操作数为寄存器间接寻址。

(5)目的操作数为寄存器寻址,源操作数为立即寻址。

(6)目的操作数为基址变址寻址,源操作数为寄存器寻址。

(7)目的操作数为相对基址变址寻址,源操作数为寄存器寻址。

(8)目的操作数为寄存器寻址,源操作数为相对基址变址寻址。

(9)目的操作数为寄存器间接寻址,源操作数为寄存器寻址。

(10)目的操作数为直接寻址,源操作数为寄存器寻址。

例 3 按下列要求编写指令序列。

(1)清除 DH 中的最低三位而不改变其他位,结果存入BH 中。

(2)把 DI 中的最高 5 位置 1 而不改变其地位。

(3)把 AX 中的 0～3 位置 1,7～9 位取反,13～15 位置 0。

(4)检查 BX 中的第 2、5 和 9 位中是否有一位为 1。

(5)检查 CX 中的第 1、6 和 11 位中是否同时为 1。

(6)检查 DX 中的第 1、4、11 和 14 位中是否同时为 0。

(7)右移 DI 三位,并把零移入最高位。

(8)把 AL 左移一位,使 0 移入最低一位。

(9)AL 循环左移三位。

(10)EDX 带进位位循环右移四位。

(11)把 AX 中的最右 4 位置 1,AX 中的最左 3 位清 0,并且把 AX 中的 7、8、9 位取反。

(12)将 AX 的内容,减去 2008H,和上次运算的借位。

(13)将变量名 TABL 的段地址送 AX。

【分析与解答】

```
(1)AND     DH,0F8H
(2)OR      DI,0FFF8H
(3)AND     AX,0FFF8H
   OR      AX,0E000H
   XOR     AX,0380H
(4)MOV     DX,BX
   TEST    DX,0004H
   TEST    DX,0020H
   TEST    DX,0200H
   JNZ     YES
(5)XOR     CX,0842H
   JZ      YES
(6)XOR     DX,4812H
   JZ      YES
(7)SHR     DI,3
(8)SHL     AL,1
```

```
(9)ROL      AL,3
(10)ROR     EDX,4
(11)OR      AX,000FH
    AND     AX,1FFFH
    XOR     AX,0380H
(12)SBB     AX,2008H
(13)MOV     AX,SEG  TABL
```

3.3 课后习题解答

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 的间接寻址。

(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。

【解答】

(1)无有效地址。

(2)D＝7237H,即为有效地址 EA。

(3)无有效地址。

(4)EA＝6370H。

(5)EA＝(BX)＋D＝6370H＋7237H＝0D5B4H。

(6)EA＝(BX)＋(SI)＝6370H＋2A9BH＝8E18H。

(7)EA ＝(BX)＋(SI)＋D＝6370H＋2A9BH＋7237H

＝1004F。

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，并把结果送回该存储单元中。

(5)把数 0B5H 与(AL)相加，并把结果送回 AL 中。

【解答】

```
(1)ADD      DX,BX
(2)ADD      AL,[BX][SI]
(3)ADD      [BX+0B2H],CX
(4)ADD      [0524H],2A59H
(5)ADD      AL,0B5H
```

3.3 写出把首地址为 BLOCK 的字数组的第 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。

【解答】

```
(1)LEA      BX,BLOCK[10]
   MOV      DX,[BX]
(2)LEA      BX,BLOCK
   MOV      DX,[BX+10]
(3)LEA      BX,BLOCK
```

```
MOV     SI,10
MOV     DX,[SI][BX]
```

3.4 现有(DS)＝2000H,(BX)＝0100H,(SI)＝0002H,(20100)＝12H,(20101)＝34H,(20102)＝56H,(20103)＝78H,(21200)＝2AH,(21201)＝4CH,(21202)＝B7H,(21203)＝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,1100H[BX][SI]
```

【解答】

(1)(AX)＝1200H

(2)(AX)＝(BX)＝0100H

(3)(AX)＝(DS＊10H＋1200H)＝$(21200H)_w$
＝4C2AH

(4)(AX)＝(DS＊10H＋BX)＝$(20100H)_w$
＝3412H

(5)(AX)＝(DS＊10H＋1100＋BX)＝$(21200H)_w$
＝402AH

(6)(AX)＝(DS＊10H＋BX＋SI)＝$(20102H)_w$
＝7856H

(7)(AX)＝(DS＊10H＋BX＋SI＋1100)
＝$(21202H)_w$＝65B7H

3.5 给定：

(IP)＝2BC0H,(CS)＝0200H,位移量 D＝5119H,(BX)＝

1200H,(DS)＝212AH,(224A0)＝0600H,(275B9)＝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。

【解答】

(1)EA＝5119H

(IP)＝5119H

(2)EA＝(BX)＝1200H

(IP)＝(224A0)＝0600H

(3)EA＝(BX＋D)＝6619H

(IP)＝(275B9)＝098AH

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。

【解答】

```
MOV  DX,[2000H]
MOV  DS,[2002H]
MOV  AX,[DX]
```

画图表示如图 3.1 所示。

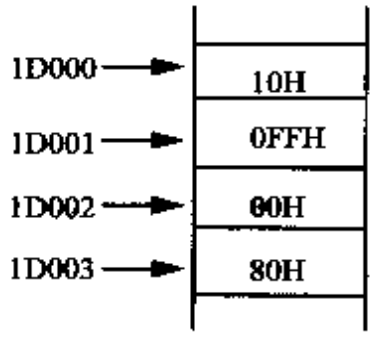


图 3.1

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

【解答】

(1)OBJ＝0625H＋27H＝064CH

(2)OBJ＝0625H＋06BH＝0690H

(3)OBJ＝0625H－03AH＝05EBH

3.8 假定(DS)＝2000H，(ES)＝2100H，(SS)＝1500H，(SI)＝00A0H，(BX)＝ 0100H，(BP)＝0010H，数据段中变量名 VAL 的偏移地址值为 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]

【解答】 各指令中源操作数字段的寻址方式及其物理地址值如表 3-2 所示。

表 3-2

寻址方式(源)	物理地址(A)
(1)立即数寻址	无
(2)寄存器寻址	无
(3)直接寻址	A＝DS＊10H＋100H＝20100H
(4)直接寻址	A＝DS＊10H＋50H＝20050H
(5)寄存器间接寻址	A＝DS＊10H＋(BX)＝20100H
(6)寄存器间接寻址	A＝ES＊10H＋(BX)＝21100H
(7)寄存器间接寻址	A＝SS＊10H＋(BP)＝15010H
(8)寄存器间接寻址	A＝DS＊10H＋(SI)＝200A0H
(9)寄存器相对寻址	A＝DS＊10H＋(BX)＋10＝2010AH
(10)寄存器相对寻址	A＝DS＊10H＋(BX)＋VAL＝20150H
(11)基址变址寻址	A＝DS＊10H＋(BX)＋(SI)＝201A0H
(12)相对基址变址寻址	A＝DS＊10H＋(BX)＋(SI)＋VAL＝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。

【解答】

```
(1) MOV     AX,[BX+6]
    MOV     [BX+7],AX
(2) LEA     DX,ARRAY[BX+1]
    MOV     AX,ARRAY[BX]
```

```
MOV     [DX],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
```

【解答】

(1) MOV AX,TABLE

将 TABLE 单元内容送入 AX，(AX)=1234H。

(2) LEA AX,TABLE

将 TABLE 符号本身的值送入 AX，(AX)=0032H。

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
ENTRY   DW      3
          ……
        MOV     BX,OFFSET TABLE
        ADD     BX,ENTRY
        MOV     AX,[BX]
```

【解答】 (AX)=1E00H

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。

【解答】

```
LEA     BX,CSTRING
MOV     DL,[BX]
MOV     DH,[BX+6]
```

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）。

【解答】 堆栈区和 SP 的内容变化过程如图 3.2 所示。

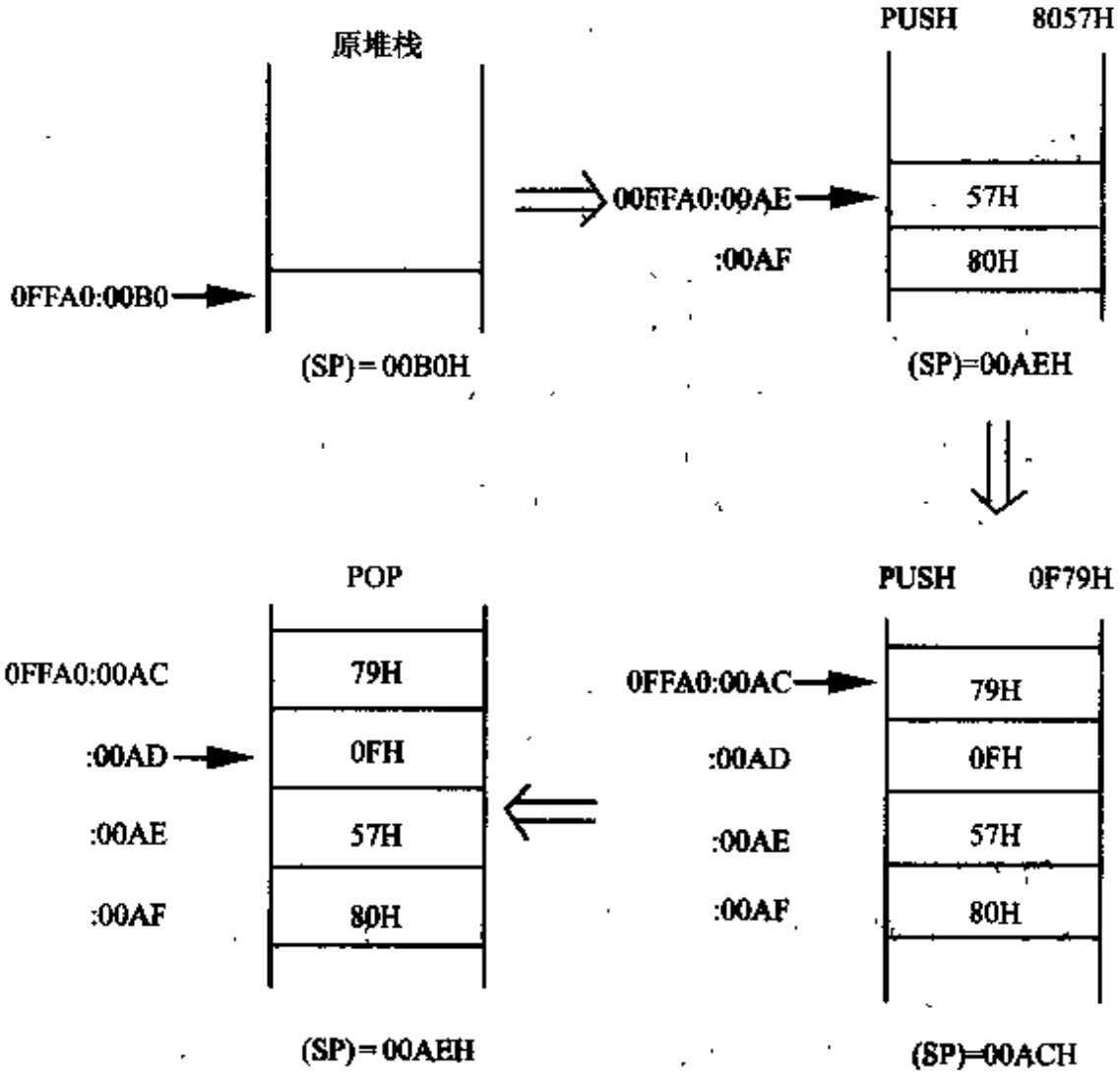


图 3.2

3.14 设(DS)＝1B00H，(ES)＝2B00H，有关存储单元的内容如图 3.3 所示。请写出两条指令把字变量 X 装入 AX 寄存器。

【解答】

```
LES     BX,[2000]
```

MOV AX,ES:[BX]

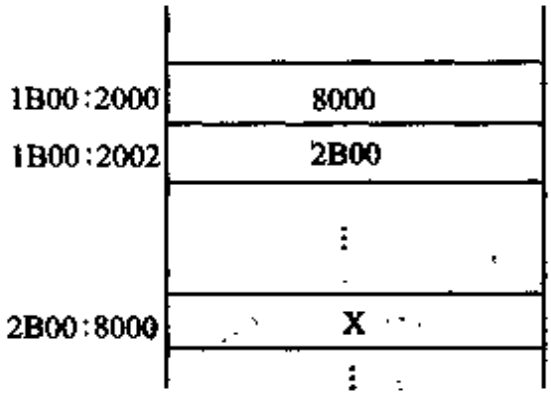


图 3.3 3.14 题的存储区情况

3.15 求出以下各十六进制数与十六进制数 62A0 之和，并根据结果设置标志位 SF,ZF,CF 和 OF 的值。

(1)1234 (2)4321 (3)CFA0 (4)9D60

【解答】

(1)62A0H+1234H=74D4H

SF=0 ZF=0 CF=0 OF=0

(2)62A0H+4321H=A5C1H

SF=1 ZF=0 CF=0 OF=1

(3)62A0H+CFA0H=3240H

SF=0 ZF=0 CF=1 OF=0

(4)62A0H+9D60H=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 4AE0 的差值，并根据结果设置标志位 SF,ZF,CF 和 OF 的值。

(1)1234 (2)5D90 (3)9090 (4)EA04

【解答】

(1)1234H−4AE0H=C744H

SF＝1　ZF＝0　CF＝1　OF＝0

(2)5D90H－4AE0H＝12A0H

SF＝0　ZF＝0　CF＝0　OF＝0

(3)9090H－4AE0H＝45A0H

SF＝1　ZF＝0　CF＝1　OF＝0

(4)EA04H－4AE0H＝AE14H

SF＝1　ZF＝0　CF＝0　OF＝1

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

(1)Z←W＋(Z－X)

(2)Z←W－(X＋6)－(R＋9)

(3)Z←(W＊K)/(Y＋6)，R←余数

(4)Z←(W－X)/(5＊Y)＊2

【解答】

```
(1)MOV     AX,[Z]
   SUB     AX,[X]
   ADD     AX,[W]
   MOV     [Z],AX
(2)ADD     [X],6
   ADD     [R],9
   MOV     AX,[W]
   SUB     AX,[R]
   MOV     [Z],AX
(3)MOV     AX,W
   IMUL    [X]
```

```
    ADD     [Y],6
    IDIV    [Y]
    MOV     [Z],AX
    MOV     [R],DX
(4) IMUL    CX,[Y],2
    XOR     DX,DX
    MOV     AX,[W]
    SUB     AX,[X]
    DIV     CX
    IMUL    2
    MOV     [Z],AX
```

3.18 已知程序段如下：

```
    MOV     AX,1234H
    MOV     CL,4
    ROL     AX,CL
    DEC     AX
    MOV     CX,4
    MUL     CX
    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？

【解答】 本题答案如表 3-3 所示。

表 3-3

	(AX)	CF	ZF	SF
MOV AX,1234H	1234H	0	0	0
MOV CL,4				
ROL AX,CL	2341H	1	0	0
DEC AX	2340H	1	0	0
MOV CX,4				
MUL CX	8D00H	0	0	1
INT 20H	(AX)=8D00H (DX)=0000H			

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

```
MOV     AX,0
DEC     AX
ADD     AX,7FFFH
ADD     AX,2
NOT     AX
SUB     AX,0FFFFH
ADD     AX,8000H
SUB     AX,1
AND     AX,58D1H
SAL     AX,1
SAR     AX,1
NEG     AK
ROR     AX,1
```

【解答】 本题答案如表 3-4 所示。

表 3-4

	(AX)	CF	SF	ZF	OF
MOV AX,0	0	0	0	1	0
DEC AX	0FFFFH	0	1	0	0
ADD AX,7FFFH	7FFEH	1	0	0	0
ADD AX,2	8000H	0	1	0	1
NOT AX	7FFFH	0	1	0	1
SUB AX,0FFFFH	8000H	1	1	0	1
ADD AX,8000H	000H	1	0	1	1
SUB AX,1	0FFFFH	1	1	0	0
AND AX,58D1H	58D1H	0	0	0	0
SAL AX,1	0B1A2H	0	1	0	1
SAR AX,1	0D8D1H	0	1	0	0
NEG AX	272FH	1	0	0	0
ROR AX,1	9397H	1	1	0	1

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，其和存放在 DATAY 中。

(2) DATAX 和 DATAY 两个双字数据相加，其和存放在从 DATAY 开始的字单元中。

(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)。

【解答】

```
(1)MOV      AX,[DATAX]
   ADD      [DATAY],AX
(2)MOV      AX,[DATAX]
   ADD      [DATAY],AX
   MOV      AX,[DATAX+2]
   ADC      [DATAY+2],AX
```

(3)$(BX)=[DATAX]_w+[DATAY]_w+1$

即 DATAX 与 DATAY 单元的字的和与 1 相加后送到 BX。

```
(4)MOV      AX,DATAX
   MUL      DATAY
(5)DATA0    DW      0
   DATA1    DW      0
   DATA2    DW      0
   DATA3    DW      0
            ......
   MOV      AX,DATAX
   MUL      DATAY
   MOV      DATA0,AX
   MOV      DATA1,DX
```

```
        MOV     AX,DATAX
        MUL     [DATAY+2]
        ADD     DATA1,AX
        ADC     DATA2,DX
        MOV     AX,[DATAX+2]
        MUL     DATAY
        ADD     DATA1,AX
        ADC     DATA2,DX
        MOV     AX,[DATAX+2]
        MUL     [DATAY+2]
        ADD     DATA2,AX
        ADC     DATA3,DX
(6)MOV  BL,23
        MOV     AX,DATAX
        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。

【解答】

```
NEG     DX
NEG     AX
ADD     AX,1
ADC     DX,0
```

3.22 试编写一个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。

【解答】 主要程序代码如下：

```
    MOV     EAX,[A]
    CMP     EAX,0
    JL      L1
    MOV     [B],EAX
    RET
L1: NEG     EAX
    MOV     [B],EAX
    RET
```

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,VALUL
(2)AND      BX,VALUE
(3)OR       BX,VALUE
(4)XOR      BX,0FFH
(5)AND      BX,0
(6)TEST     BX,01H
```

【解答】

(1)(BX)＝009AH

(2)(BX)＝0061H

(3)(BX)＝00FBH

(4)(BX)＝001CH

(5)(BX)＝0000H

(6)(BX)＝00E3H

3.24 试写出执行以下指令序列后 BX 寄存器的内容。执行前(BX)＝6D16H。

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

【解答】

(BX)＝00DAH

3.25 试用移位指令把十进制数＋53 和－49 分别乘 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？

【解答】

①乘 2 用算术左移 SHL 或逻辑左移(SAL)；

除以 2 用算术右移(SAR)。

②＋53 乘 2 为 106，除以 2 为 26；

－49 乘 2 为－98，除以 2 为－24。

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
```

【解答】 实现将 DX：AX 算术或逻辑左移 4 位。

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
(2)SAR      DX,CL
(3)SHL      DX,CL
(4)SHL      DL,1
(5)ROR      DX,CL
(6)ROL      DL,CL
(7)SAL      DH,1
(8)RCL      DX,CL
(9)RCR      DL,1
```

【解答】

(1)(DX)＝05CH

(2)(DX)＝017H

(3)(DX)＝05C8H

(4)(DX)＝0072H

(5)(DX)＝2017H

(6)(DX)＝00CDH

(7)(DX)＝00H

(8)(DX)＝05CCH

(9)(DX)＝0DCH

3.28 下列程序段执行完后，BX 寄存器中的内容是什么？

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

【解答】

(BX)＝0C02DH

3.29 假设数据定义如下：

```
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 CONMAE 中的字符串传送到 PRLINE。

(2)从右到左把 CONMAE 中的字符串传送到 PRLINE。

(3) 把 CONAME 中的第 3 和第 4 个字符装入 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 字符串中有无空格字符，如有则把它传送给 BH 寄存器。

【解答】

```
(1)LDS      SI,CONAME
```

```
    LES     DI,PRLINE
    MOV     CX,DI
    SUB     CX,SI
    CLD
    REP     MOVSB
(2) LDS     SI,CONAME
    LES     DI,PRLENE
    MOV     CX,DI
    SUB     CX,SI
    DEC     CX
    ADD     SI,CX
    ADD     DI,CX
    INC     CX
    STD
    REP     MOVSB
(3) LDS     SI,CONAME
    CLD
    ADD     SI,2
    LODSW   SI
(4) LES     DI,PRLINE
    ADD     DI,5
    CLD
    STOSW   D
(5) MOV     AL,20H
    LDS     SI,CONAME
    LES     DI,PRLINE
    MOV     CX,DI
    SUB     CX,SI
```

```
    CLD
    REPNE   SCAS  SI
    CMP     CX,0
    JZ      L1
    MOV     BH,20H
L1:……
    ……
```

3.30 编写程序段，把 STRING 中字符串的'&'字符用空格符代替。

```
    STRING    DB    'The date is FEB&03'
```

【解答】

```
    DSEG      SEGMENT
    STRING    DB        'the date is FEB&03'
    LEN       DB        ?
    DSEG      ENDS
              ……
    MOV       CX,LEN_STRING
    LEA       SI,STRING
L0: CMP       [SI],'&'
    JZ        L1
    JMP       L2
L1: MOV       [SI],' '
L2: INC       SI
    LOOP      L0
```

3.31 假设程序中数据定义如下：

```
    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 FRINT_LINE 的前 30 个字节中，把 STUDENT_ADDR 移到 PRINT_LINE 的后 9 个字节中。

【解答】

```
(1)CLD
   MOV     CX,132
   MOV     AL,' '
   LEA     DI,PRINT_LINE
   REP     STOSB
(2)MOV     CX,9
   LEA     DI,STUDENT_ADDR
   MOV     AL,'_'
   CLD
   REPNE   SCASB
(3)MOV     CX,9
   LEA     DI,STUDENT_ADDR
   MOV     AL,'_'
   STD
   REPNE   SCASB
(4)CLD
   MOV     CX,30
   MOV     AL,' '
   LEA     DI,STUDENT_NAME
```

```
    REPE    SCASB
    JZ      L1
    RET
L1: MOV     CX,30
    MOV     AL,'*'
    LEA     DI,STUDENT_NAME
    REP     STOSB
(5) MOV     CX,30
    LEA     SI,STUDENT_NAME
    LEA     DI,PRINT_LINE
    REP     MOVSB
    STD
    MOV     CX,9
    LEA     SI,STUDENT_ADDR+8
    LEA     DI,PRINT_LINE+49
    REP     MOVSB
    CLD
```

3.32 编写一程序段，比较两个 5 字节的字符串 OLDS 和 NEWS，如果 DLDS 字符串不同于 NEWS 字符串则执行 NEW_LESS，否则顺序执行程序。

【解答】

```
    LEA     SI,DLDS
    LEA     DI,NEWS
    CLD
    MOV     CX,5
    REPE    CMPSB
    JNZ     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 的内容等于零，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。

【解答】

```
(1) CMP     CX,DX
    JB      EXCEED
(2) CMP     AX,BX
    JL      EXCEED
(3) CMP     CX,0
    JZ      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) 147B　　80DC

(2) B568　　54B7

(3) 42C8　　608D

(4) D023　　9FD0

(5) 94B7　　B568

问该程序执行完后，程序转向哪里？

【解答】 AX、BX 取不同值时的结果如表 3-5 所示。

表 3-5

AX	BX	ADD	SUB	OF	CF	转向
147B	80DC	9557		0	0	L1
B56B	54B7	10A1F		0	0	L1
42C8	608D	0A355		1	0	L2
D023	9FD0	16FF3	3053	1	0	L3
94B7	B56B	14A1F	DF4F	0	1	L4

3.35 指令 CM PAX，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 和 NLE 中的任一个。如果 AX 和 BX 的内容给定如下：

　　AX　　BX

(1) 1F52　　1F52

(2) 88C9　　88C9

(3) FF82　　007E

(4) 58BA　　020E

(5) FFC5　　FF8B

(6) 09A0　　1E97

(7) 8AEA　　FC29

(8) D367　　32A6

问以上 8 条转移指令中的哪几条将引起转移到 L1?

【解答】

(1)NB	BE	NL	LE
(2)NB	BE	NL	LE
(3)NB	NBE	L	LE
(4)NB	NBE	NL	NLE
(5)NB	NBE	L	LE
(6)B	BE	L	LE
(7)B	BE	NL	NLE
(8)NB	NBE	L	LE

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

【解答】

查看 IP 是否大于 q，若 $\begin{cases} IP>q, & 则(AX)=1。 \\ IP\leqslant q, & 则(AX)=2。 \end{cases}$

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 ROUTIN_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。

【解答】 流程图略，详见以下程序段中的注释。

```
MOV   AX,STATUS ;将要测试的数据送入 AX 中
TEST  AL,2AH    ;测试 STATUS 中的低 8 位
JZ    ROUTINE_4 ;若全为 0,则转 ROUTINE_4
AND   AL,2AH    ;屏蔽其他位
JP    ROUTINE_2 ;含偶数个 1,则转 ROUTINE_2
CMP   AL,2AH
JZ    ROUTINE_1 ;三位全为 1,则转 ROUTINE_1
JMP   ROUTINE_3 ;否则,转 ROUTINE_3
```

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 和 DX 四个寄存器的内容分别是什么？

```
TITLE   EXLOOP.COM
CODESG SEGMENT
        ASSUME  CS:CODESG,DS:COOESG,SS:
                CODESG
```

```
        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
```

【解答】

(1)执行 LOOP　L20 后

(AX)=05H　　(BX)=10H

(CX)=00H　　(DX)=00H

②执行 LOOPE　L20 后

(AX)=02H　　(BX)=04H

(CX)=03H　　(DX)=01H

③LOOPNE　L20 后

(AX)=03H　　(BX)=07H

(CX)=02H　　(DX)=00H

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，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(返回的段地址是 1000，偏移地址是 0600)。

请画出每次调用及返回时的堆栈状态。

【解答】 每次调用及返回时堆栈的状态如图 3.4 所示。

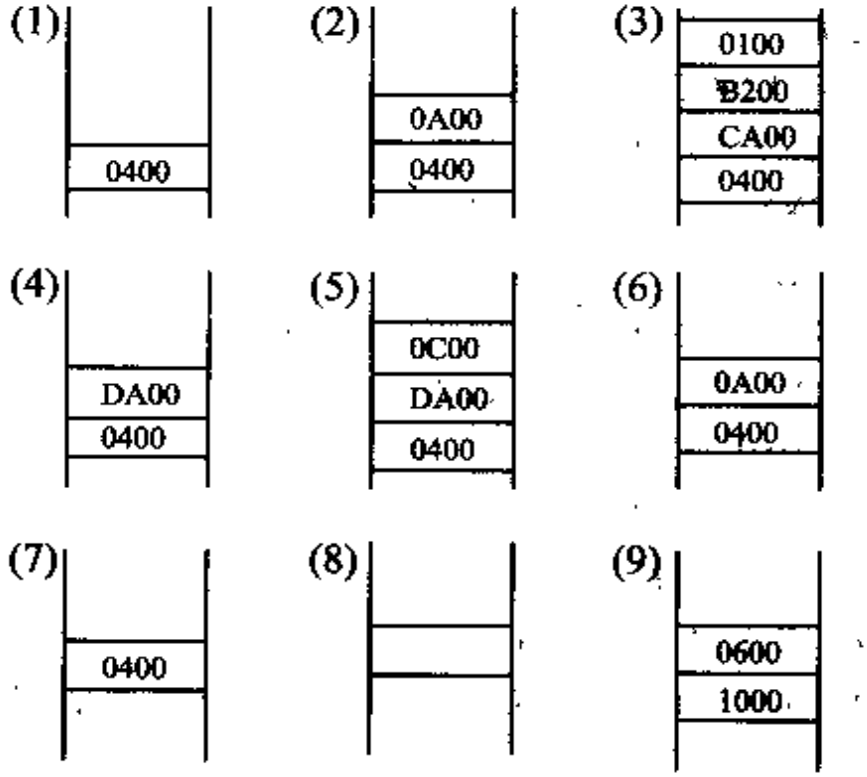


图 3.4

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]

【解答】

(1)物理地址=3100H

(2)物理地址=5100H

(3)物理地址=7100H

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]

【解答】

(1)从DS段偏移地址2000H开始的4个字节单元的内容为:56H,F4H,23H,98H。

(2)从DS段偏移地址为348CH开始的4个字节单元的内容为:91H,34H,F2H,01H。

(3)(EBX)=4CA26D92H。

3.42 说明下列指令的操作:

(1)PUSH AX

(2)POP ESI

(3)PUSH [BX]

(4)PUSHAD

(5)POP DS

(6)PUSH 4

【解答】

(1)PUSH AX

(SP)←(SP)－2

((SP+1,(SP))←(AX)

(2)POP ESI

(ESI)←((ESP)＋3,(ESP)＋2,(ESP)＋1,(ESP))←

(ESP)＋4

(3)PUSH [BX]

(SP)←(SP)－2

((SP)＋1,(SP))←[BX]

(4)PUSHAD

进栈次序 EAX,ECX,EDX,EBX,指令执行前的 ESP,EBP,ESI,EDI。

(SP)←(SP)－32

(5)POP DS

(DS)←((SP)＋1,(SP))

(SP)←(SP)＋2

(6)PUSH 0004H

(SP)←(SP)－2

((SP)＋1,(SP))←0004H

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,34FFFFH

(2)MOV EBX,40000000

SUB EBX,1500000

(3)MOV EAX,39393834H

```
    AND     EAX,0F0F0F0FH
(4)MOV      EDX,9FE35DH
    XOR     EDX,0F0F0F0H
```

【解答】

(1)(EAX)=2CEFF93H

(2)(EBX)=38500000D

(3)(EAX)=09090804H

(4)(EDX)=9FEF5FDH

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
    MOVSX   EBX,BX
(2)MOV      CL,-8
    MOVSX   EDX,CL
(3)MOV      AH,7
    MOVZX   ECX,AH
(4)MOV      AX,99H
    MOVZX   EBX,AX
```

【解答】

(1)(EBX)=0FFFFFF4H

(2)(EDX)=0FFFFFF8H

(3)(ECX)=00000007H

(4)(EBX)=00000099H

3.45 请给出下列指令序列执行完后 EAX 和 EBX 的内容。

```
    MOV     ECX,307F455H
    BSF     EAX,ECX
    BSR     EBX,ECX
```

【解答】

(EAX)=00DH

(EBX)＝25DH

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
BSR     DX,BX
```

【解答】

(AX)＝03D

(DX)＝07D

3.47 试编写一程序段，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)。

【解答】

```
ADD     ECX,EDX
ADD     ECX,ESI
MOV     EDI,ECX
```

3.48 请说明"IMUL　　BX,DX,100H"指令的操作。

【解答】

(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 2，最后的结果存入 DX 寄存器中。

【解答】

```
MOVSX   AX,BL
IDIV    CL
SHL     AL
MOVSX   DX,AL
```

3.50 请说明"JMP　DI 和 JMP　[DI]"指令的区别。

【解答】

JMP　DI　　跳转到偏移地址为(DI)的地址，属寄存器寻址

JMP [DI] 跳转到 DI 寄存器中的地址所指的地址，属存储器寻址

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。

【解答】

```
DSEG    SEGMENT
ARRAY   DB      100DUP(?)
UP      DB      ?
DOWN    DB      ?
DSEG    ENDS
CSEG    SEGMENT
          ……
        MOV     CX,100H
        LEA     SI,ARRAY
L3:     CMP     [SI],42H
        JA      L1
        JB      L2
        INC     SI
        LOOP    L3
        JMP     L4
L1:     INC     SI
        LOOP    L3
        JMP     L3
L2:     INC     AX
        INC     SI
        LOOP    L3
L4:     MOV     [UP],BX
```

```
        MOV     [DOWN],AX
        RET
        ……
```

3.52 请用图表示“ENTER 16,0”所生成的堆栈帧的情况。

【解答】 “ENTER 16,0”所生成的堆栈帧的情况如图 3.5 所示。

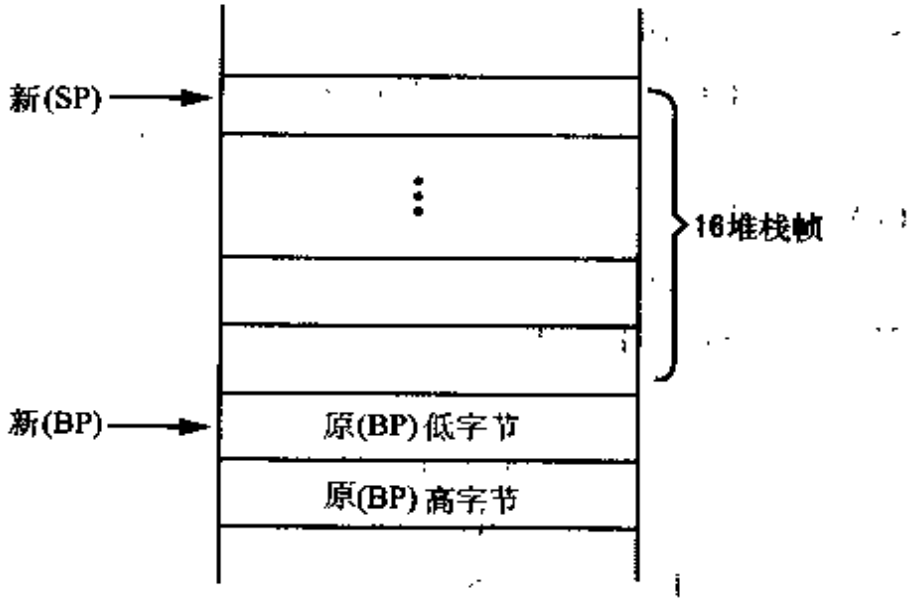


图 3.5

3.4 强化训练

1. 程序数据段中定义的数组如下：

```
NAMES   DB      'TOM.'
        DB      20
        DB      'ROST.'
        DB      30
        DB      'KATE.'
        DB      25
```

请指出下列指令是否正确？为什么？

```
(1)MOV    BX,OFFSET  NAMES
  MOV     AL,[BX+5]
(2)MOV    AX,NAMES
(3)MOV    AX,WORD  PTR  NAMES+1
(4)MOV    BX,6
  MOV     SI,5
  MOV     AX,NAMES[BX][SI]
(5)MOV    BX,6*2
  MOV     SI,5
  MOV     AX,OFFSET  NAMES[BX][SI]
  INC     [AX]
(6)MOV    BX,6
  MOV     SI,5
  LEA     DI,NAMES[BX][SI]
  MOV     AL,[DI]
```

2. 求指令段的执行结果。

执行下列指令，AX 寄存器中的内容是什么？

某数据段定义如下：

```
TABA    DW    1,2,3,4,5
TABB    DB    0AH,0BH,0CH,0DH,0EH
TABC    DW    10,20,30,40,50
```

代码段定义如下：

```
(1)MOV    BX,OFFSET  TABA
(2)LEA    SI,TABB
(3)MOV    DI,4
(4)MOV    AX,0
(5)MOV    AL,[SI+2]
(6)XCHG   AX,SI
```

(7)MOV　　AX,[BX][SI]

(8)MOV　　AX,[BX][DI]

3. 编写程序段，完成下面计算公式，并把所得的商和余数分别存入 X 和 Y 中(其中：A,B,C,X 和 Y 都是有符号的字变量)。

$$(C-120+A)*B/C$$

答　案

1.(1)两条指令都是合法指令。第 1 条指令取得 NAMES 的偏移地址，第 2 条 MOV 指令使用间接寻址方式，将地址为(DS)*10H+(BX)+5 字节中的数据传送给 AL，结果(AL)＝20。

(2)这条指令不正确，因为 NAMES 的属性为字节，而目的寄存器是 AX，所以类型不匹配。

(3)为合法指令。指令中将已定义的字节变量用伪操作 PTR 改变为字类型，所以避免了类型不匹配的错误。操作结果(AX)＝4D4FH(即 M 和 O 的 ASCII 码)。

(4)前两条指令使用的是立即数方式，第 3 条指令的源操作数字段使用的是相对基址变址方式，但形成的数据段地址中的数据属性为字节，而源操作数据存器为 AX，故出现"类型不匹配"的错误。如 AX 改为 AL，这条指令就是合法指令。

(5)前两条指令是正确的，后两条指令有错误。在汇编语言中，OFFSET 操作将得到变量的偏移值，但对相对基址变址寻址方式形成的值在汇编指令时还是未知的。同样 MOV　BX,OFFSET　NAMES [SI]也是错误指令。第 4 条指令中，AX 不能作为基址寄存器用。

(6)均为合法指令。第 3 条指信令中的 DI 取得一个字节地址：(BX)+(SI)+OFFSET　NAMES。然后再按 DI 中的偏移地址，在数据段中将一字节内容传送给 AL 寄存器，结果(AL)＝30。

2. 数据段定义的数据存放在连续的存储单元中。每个存储

单元存放一个字节数据(如图 3.6 所示)。

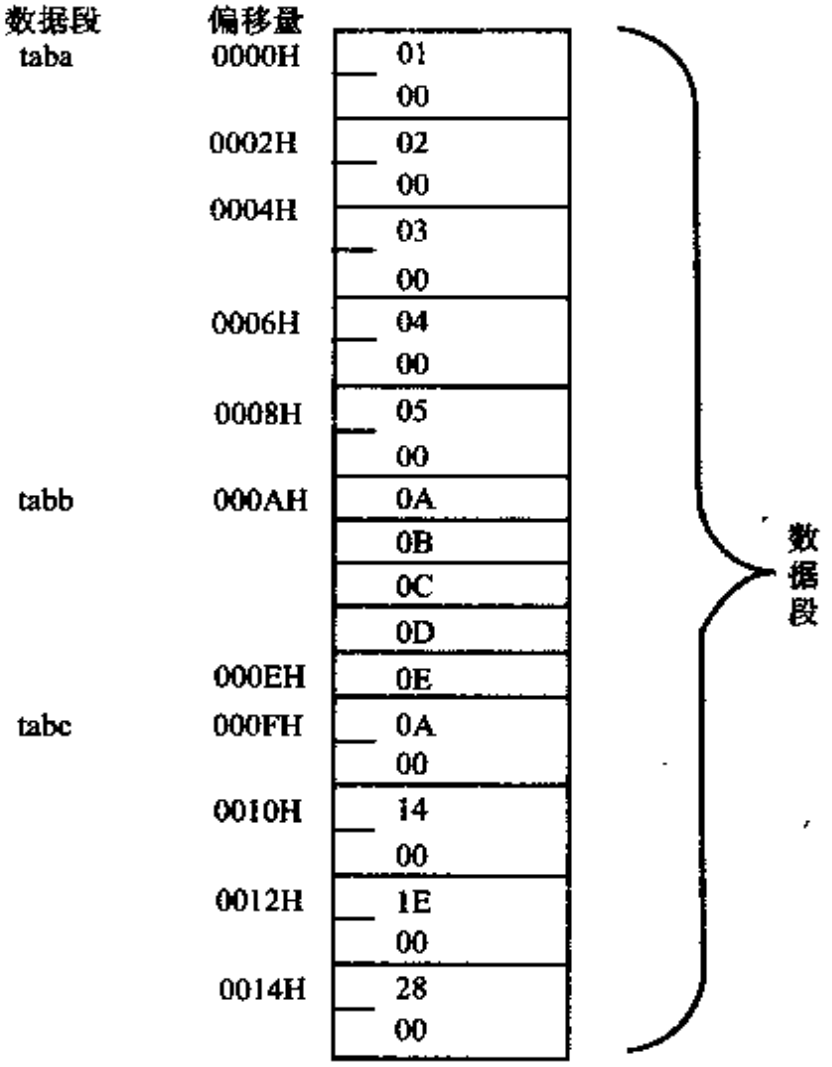


图 3.6 数据段中数据的存储情况

分析指令段的执行结果。

(1)(BX)=TABA 的偏移地址=000H

(2)(SI)=TABB 的偏移地址=000AH

(3)(DI)=4

(4)(AX)=0

(5)(AL)=(DS * 16+SI+2)=(DS * 16+000CH)=0CH

(6)交换 AX 和 SI 寄存器的内容,AX=000AH,SI=000CH

(7)(AX)=(DS * 16+BX+SI)=(DX * 16+0000H+

000CH)＝0D0CH

(8)(AX)＝(DS＊16＋BX＋DI)＝(DS＊16＋0000H＋0004H)＝0003H

上述指令段执行后(AX)＝0003H。

3. 程序代码如下：

```
MOV   AX,C
SUB   AX,120
MOV   BX,AX
MOV   AX,A
MOV   SI,B
IMUL  SI
MOV   BX,AX
MOV   CX,DX
MOV   AX,BX
CWD
ADD   AX,BX
ADC   DX,CX
MOV   CX,C
IDIV  CX
MOV   X,AX
MOV   Y,DX
```

第4章 汇编语言程序格式

4.1 内容提要

4.1.1 汇编语言简介

1. 一般概念

(1)机器语言

机器语言是用来直接描述机器指令,使用机器指令的规则及用二进制代码来表示机器指令的一种程序设计语言,是CPU能直接识别的唯一语言。机器语言的执行效率高,但却难以理解,难读懂。另外,机器语言难于移植,通用性不高。

(2)汇编语言

汇编语言是用符号化的指令助记符来表示二进制机器指令,助记符一般是能够说明指令功能的词汇的缩写。用指令助记符、符号地址等组成的符号指令称为汇编格式指令。

2. 汇编程序

汇编程序是用汇编语言编写的程序,它大大提高了程序的可读性,但失去了CPU能直接识别的特性。执行汇编语言程序时,需要用汇编程序将源程序翻译成CPU能识别的机器指令序列。

常用的汇编程序开发工具有:Microsoft公司的MASM、Borland公司的TASM和DEBUG等。

3. 汇编语言的特点

(1)与机器相关性

汇编语言程序指令是机器指令的一种符号表示,不同类型CPU有不同的机器指令系统,因此有不同的汇编语言。

(2)执行的高效率

汇编语言的执行速度快、执行效率高。

(3)编写程序的复杂性

汇编语言是一种面向机器的语言,编程时要安排运算的每一个细节,因此程序的编写过程比较繁琐、复杂。

(4)调试的复杂性

在通常情况下,调试汇编语言程序要比调试高级语言程序困难。

4.1.2 汇编语言语句的种类

1. 指令语句

微处理器指令系统中提供的指令语句,汇编时能产生目标代码。指令语句有对应的机器语言指令,如 ADD AX,BX 指令对应的机器语言指令为 01 D8;MOV AX,2008H 指令对应的机器语言指令为 B8 08 20。机器指令是微处理器的设计者规定的机器所能执行的基本操作。指令系统的特点如下:

①指令系统通常包括如下几大类指令:数据传送类、数据运算类、程序控制转移类、CPU 控制类;

②指令助记符用英文缩写符号;

③指令的操作数通常为寄存器、存储单元、常数及表达式。

2. 伪指令语句

只是告诉汇编程序如何汇编,没有与其对应的机器语言指令。伪指令语句的功能是告诉汇编程序对汇编语言源程序汇编时应该

怎样处理源程序中定义的数据、如何分配存储空间、如何区分程序定义的各个段等等。

伪指令通常包括数据定义及存储器分配、程序中符号的定义、程序中各个段的定义、过程的定义、程序结束等。

3. 宏指令语句

用户自定义指令。用户利用宏定义可以把一个程序段定义为一条宏指令。一条宏指令包含若干条指令语句。当一条宏指令作为语句出现时，该语句就称为宏指令语句。

汇编程序在对源程序进行汇编时遇到宏指令就对其进行宏展开。宏展开是用宏指令体中的指令来替代这条宏指令。

4.1.3 伪指令

1. 表达式赋值伪指令

EQU 伪指令的功能是给各种形式的表达式赋予一个名字。用表达式的名字代替表达式又使程序简单，也便于修改和调试。

格式如下：

表达式名　　EQU　　表达式

2. 数据定义伪指令

数据定义伪指令的格式如下：

变量名　　DB　　表达式

表达式可为如下几种情况：

常数表达式

问号(?)

地址表达式(适用于 DW 和 DD)

字符、字符串(适用于 DB)

重复子句 DUP(表达式)

以及用逗号分开的上述各项。

3. LABEL 伪指令

格式如下：

[变量或标号]LABEL[类型]

为当前存储单元定义一个指定类型的变量或标号。

4. 段定义伪指令

源程序的分段是由段定义伪指令完成的。

格式如下：

```
段名  SEGMENT  [定位类型][组合类型][类别]
          ⋮   过程体语句
段名  ENDS
```

5. 过程定义伪指令

(子程序)用伪指令 PROC 和 ENDP 来定义。

指令格式如下：

```
过程名  PROC [类型]
          ⋮     过程体语句
          RET
过程名  ENDP
```

6. 地址计数器与常用的伪操作

(1)地址计数器 $

地址计数器的值可用 $ 来表示，汇编语言允许用户直接用 $ 来引用地址计数器的值。

(2)ORG 伪操作

ORG 伪操作用来设置当前地址计数器的值，其格式为：

ORG 常量表达式

(3)EVEN 伪操作

EVEN 伪操作使下一个变量或指令开始于偶数字节地址。

(4)ALIGN 伪操作

ALIGN 伪操作为保证双字数组边界从 4 的倍数开始。

(5)表达式中常用的操作符：

(ⅰ)算术操作符

+、-、*、/、MOD

(ⅱ)逻辑操作符

AND、OR、XOR、NOT

(ⅲ)关系操作符

EQ、NE、LT、GT、LE、GE

(ⅳ)数值回送操作符

TYPE、LENGTH、SIZE、OFFSET、SEG

(ⅴ)属性操作符

PTR、SHORT、THIS、HIGH、LOW

4.1.4 汇编语言的上机过程

1. 汇编程序的开发过程包括编辑、编译、连接和调试四个阶段，如图 4.1 所示。

2. 汇编语言的上机步骤。

(1)建立汇编语言的工作环境。

为运行汇编语言程序至少要在磁盘上建立以下文件：

①EDIT. COM

DOS 的文本编程工具，用于编辑输入汇编语言源程序。

②MASM. EXE

宏汇编器。

③LINK. EXE

连接器。

④DEBUG. COM

DOS 的动态调试器。

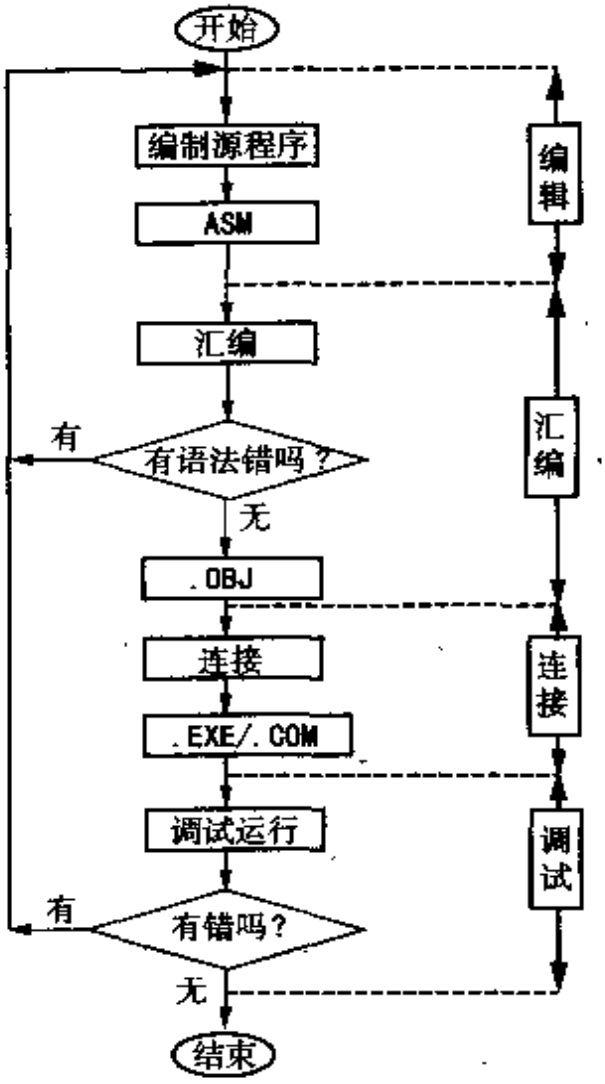


图 4.1　汇编语言程序开发流程图

(2)输入 .ASM 文件。

用编辑程序(如 DOS 的 EDIT)将汇编语言源程序输入计算机,经修改认为无误后,存入文件系统。

(3)用汇编程序产生 .OBJ 文件。

如果汇编正确,则生成 .OBJ 文件;若汇编不正确,则无法得到 .OBJ 文件。

如果在汇编过程中发现源程序有语法错误,则系统会输出"出错信息",列出第几行有什么样的错误。程序员应重新调用编辑程

序，对源程序修改后再进行汇编，直到汇编通过为止。

(4)用连接程序产生 .EXE 文件。

只有得到正确的 .OBJ 文件，才能进行连接操作。

将目标程序和库函数或其他目标程序连接成了可执行的目标程序。调用 LINK 实现连接。LINK 程序把目标模块 .OBJ 与相关的目标模块连接，产生一个运行模块文件 .EXE。

(5)利用动态调试软件 DEBUG 来验证程序的正确性。

若执行程序发现结果有错误、程序运行的中间结果或最终结果不显示，可使用 DEBUG 调试。常用的 DEBUG 命令如表 4-1 所示。

表 4-1

DEBUG 命令	命令格式	作　用
汇编命令 A	A[地址]	把汇编程序语言的语句直接汇编入内存
显示命令 D	D[范围]或 D[地址]	显示存储器中的部分内容
运行命令 G	G[地址]	执行当前正在调试的程序
退出命令 Q	Q	结束 DEBUG 程序
寄存器命令 R	R 寄存器名	显示并修改寄存器的内容
跟踪命令 T	T[地址][值]	从 CS:IP 或指定的地址开始执行一条或多条指令
反汇编命令 U	U[地址]或 U[范围]	从指定的地址开始执行一条或多条指令

4.1.5　汇编语言的源程序结构

汇编语言源程序是由若干语句序列组成的，用来实现一个应用程序所编写的若干程序段都是由语句组成的。

语句序列应包括：数据、处理数据的实体、承上启下的记录。

微机系统内存是分段管理的，为了与之相对应，汇编语言源程

序也分若干个段来构成。

80x86 微处理器系统的存储结构是分段式访问结构，这种结构是程序运行的基础，因此，80x86 汇编语言程序必须具备：数据段（定义加工处理对象）、代码段（处理数据的对象）、堆栈段。不论程序在某个时刻最多能访问多少个段，在编程序时，程序员都可以定义比该段数更多的段。

在通常情况下，一个段的长度不能超过 64KB，在 80386 及其以后系统的保护方式下，段基地址是 32 位，段的最大长度可达 4GB。数据段用来在内存建立一个适当的工作区，以存放常数、变量以及作算术运算区和用来作为 I/O 接口传送数据的工作区。如果程序需要可以定义附加段作为数据的辅助存储区。堆栈段用来在内存中建立一个堆栈区，以便在中断和子程序调用时使用，堆栈还起承上启下的作用，用于模块间参数的传送。代码段存放的是程序执行的代码，也是程序不可缺少的部分。

程序的分段结构如图 4.2 所示。

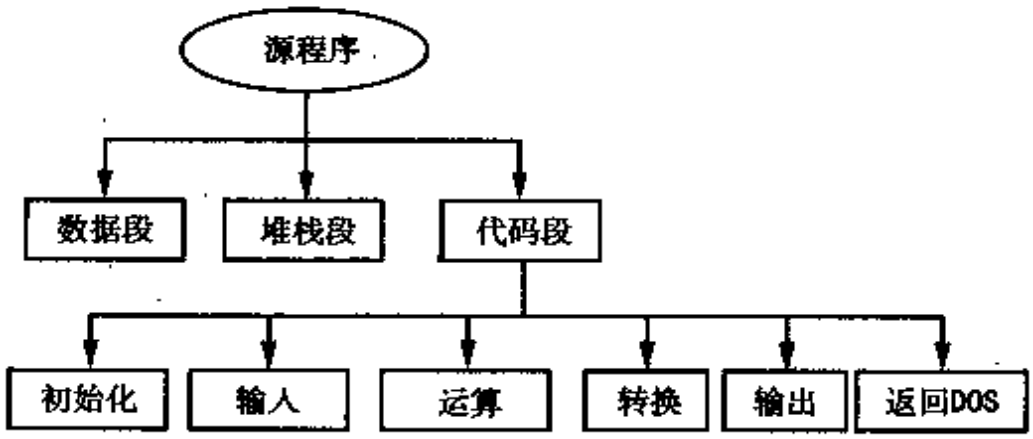


图 4.2　汇编源程序的分段结构图

在进行汇编语言程序设计时，必须考虑分段结构，根据实际问题的需要，使用段定义伪指令和段寻址伪指令来构造一个由若干段组成的程序。通常把数据放在代码的前面。

4.1.6 考 点

1. 汇编语言语句的种类、格式；
2. 常用伪指令的定义、功能及其应用；
3. DEBUG 命令的使用。

4.2 例题精解

例 1 设数据段数据定义如下：

```
NUMBER1    DW    2008H
NUMBER2    DW    15  DUP(2006H)
STRING1    DB    'How do you do? $'
```

下面几条 MOV 指令单独执行后，目的寄存器 CX 中的内容是什么？

```
(1)MOV     CX,NUMBER1
   MOV     CX,NUMBER2
(2)MOV     CX,TYPE NUMBER1
   MOV     CX,TYPE STRING1
(3)MOV     CX,LENGTH NUMBER2
   MOV     CX,LENGTH STRING1
(4)MOV     CX,SIZE STRING2
```

【分析与解答】

```
(1)MOV     CX,NUMBER1
```

这条指令是直接寻址方式、字操作，其功能是取出 DS:NUMBER1 单元中的内容 2008H→CX 中，即本条指令执行后 (CX)=2008H。

```
   MOV     CX,NUMBER2
```

同理，本条指令的执行结果(CX)=2006H。

(2)MOV　　CX,TYPE NUMBER1

TYPE 是数值回送操作符,汇编程序将回送分配给该变量(NUMBER1)以字节数表示的类型:DB 伪操作的类型属性值为1,DW 为 2,DD 为 4。

这条指令要获取 NUMBER1 的类型值,所以执行结果(CX)=0002H。

MOV　　CX,TYPE STRING1

同理,本条指令的执行结果(CX)=0001H。

(3)MOV　　CX,LENGTH NUMBER2

LENGTH 是数值回送操作符,对于变量中使用 DUP 的情况,汇编程序将回送分配给该变量(NUMBER2)的单元数。对于其他情况则回送 1。本条指令变量 NUMBER2 是用 DUP 定义的,所以执行结果(CX)=000FH。

MOV　　CX,LENGTH STRING1

同理,本条指令的执行结果(CX)=0001H。

(4)MOV　　CX,SIZE STRING2

SIZE 是数值回送操作符,汇编程序将回送分配给该变量的字节数。

SIZE=LENGTH * TYPE

所以,本条指令的执行结果(CX)=15 * 2=30=001EH。

例 2　按下面要求写出相应的数据定义语句。

(1)定义一个字节区域,第一个字节的值为 20,其后跟 20 个初值为 0 的连续字节。

(2)定义一个以 0 为结束符的字符串,其初值为:The course is easy。

(3)定义 2008 个字,其初值为 0。

(4)从一个偶地址开始定义一个字变量 word。

【分析与解答】

```
(1)VAR      DB    20,20 DUP(0)
(2)STRING   DB    'the course is easy','0'
(3)VAR      DW    2008 DUP(0)
(4)EVEN
   WORD     DW    ?
```

例3 对于如下数据定义，指出下列指令的错误。

数据定义如下：

```
DESG    SEGMENT
N1      DB      ?
N2      DB      10,20
N3      EQU     100
N4      DB      13,10
N5      DW      ?
N6      EQU     168H
DESG    ENDS
```

指令如下：

```
(1)MOV    N3,SI
(2)MOV    BX,N1
(3)MOV    N2,AX
(4)CMP    N5,N2
(5)N4     EQU     1024
(6)MOV    AL,N6
```

【分析与解答】

(1)非法直接寻址。

(2)、(3)、(4)均为操作数类型不匹配。

(5)重复定义。

(6)常数超出范围。

例 4 假设在数据段 DSEG、附加段 ESEG 和堆栈段 SSEG 中分别定义了字变量 X、Y 和 Z,试编制一完整的程序计算 X+Y+Z,并将结果送 X 。

【分析与解答】 程序代码如下:

```
DSEG    SEGMENT
BUFF    DB      0,1,2,3,4,5,6,7,8,9
ASC     DB      10 DUP(?)
COUNT   EQU     $-BUFF
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS:CSEG,DS:DSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        LEA     SI,BUFF
        LEA     DI,ASC
        MOV     CX,COUNT
L0:     MOV     AL,[ST]
        MOV     ES,ESEG
        MOV     AX,SSEG
        MOV     SS,SSEG
        MOV     AX,X
        ADD     AX,Y
```

```
        ADC     AX,Z
        MOV     X,AX
        RET     ;返回 DOS,退出程序
CESG    ENDS
        END     START
```

4.3 课后习题解答

4.1 指出下列指令的错误。

(1)MOV AH,BX
(2)MOV [BX],[SI]
(3)MOV AX,[SI][DI]
(4)MOV MYDAT[BX][SI],ES:AX
(5)MOV BYTE PTR[BX],1000
(6)MOV BX,OFFSET MYDAT[SI]
(7)MOV CS,AX
(8)MOV ECX,AX

【解答】

(1)操作数类型不匹配。
(2)存储器与存储器不能赋值。
(3)源操作数寻址方式有误。
(4)源操作数寻址方式有误。
(5)操作数类型不匹配。
(6)OFFSET 后应为地址值,而不是数据。
(7)CS 不能作为目的操作数。
(8)操作数类型不匹配。

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

(2)CMP　OP1,25

(3)CMP　OP1,OP2

(4)CMP　AX,OP1

【解答】

(1)错误。目的操作数不能是立即数。

(2)正确。

(3)错误。两操作数不能都为存储器寻址方式。

(4)错误。操作数类型不匹配。

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，请指出下列指令中哪些是非法的？它们的错误是什么？

(1)MOV　BP,AL

(2)MOV　WORD_OP[BX+4*3][DI],SP

(3)MOV　WORD_OP1,WORD_OP2

(4)MOV　AX,WORD_OP1[DX]

(5)MOV　SAVE_WORD,DS

(6)MOV　SP,SS:DATA_WORD[BX][SI]

(7)MOV　[BX][SI],2

(8)MOV　AX,WORD_OP1+WORD_OP2

(9)MOV　AX,WORD_OP1-WORD_OP2+100

(10)MOV　WORD_OP1,WORD_OP2

【解答】

(1)错误。操作数类型不匹配。

(2)正确。

(3)错误。两操作数不能都为存储器寻址方式。

(4)错误。源操作数寻址方式错误。

(5)正确。

(6)正确。

(7)正确。

(8)正确。

(9)正确。

(10)错误。两个操作数不能都为存储器寻址方式。

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

(2)SUB AL,VAR1

(3)JMP LAB[SI]

(4)JNZ VAR1

(5)JMP NEAR LAB

【解答】

(1)错误。两操作数不能都为存储器寻址方式。

(2)错误。操作数类型不匹配。

(3)错误。转向地址应为标号,不能为变量。

(4)错误。转向地址应为标号,不能为变量。

(5)错误。无 PTR。

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
```

【解答】

上述两条语句所分配的存储空间及初始化的数据值分别如图

4.3(a)、(b)所示。

42H	B
59H	Y
54H	T
45H	E
0CH	12D
F4H	-12H
00H	0
	?
01H	1
02H	2
01H	1
02H	2
	?
00H	0
	?
01H	1
02H	2
01H	1
02H	2
	?
00H	0
	?
01H	1
02H	2
01H	1
02H	2
	?

(a)

00H	0
00H	
01H	1
00H	
02H	2
00H	
00H	0
00H	
01H	1
00H	
02H	2
00H	
00H	0
00H	
01H	1
00H	
02H	2
00H	
	?
	?
FBH	
FFH	-5
59H	Y
42H	B
45H	E
54H	T
56H	
02H	

(b)

图 4.3

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(例如：DW 5150H)。

【解答】

```
    DW      5150H
    DB      50H,51H
    DB      'P','Q'
```

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 为十进制数字节变量：32；

(3)FLD3B 为十六进制数字节变量：20；

(4)FLD4B 为二进制数字节变量：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)及基数量(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；

(9)FLD2W 为二进制数字变量：01011001；

(10)FLD3W 为(7)中零件表的地址变量；

(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。

【解答】

```
    DATASG    SEGMENT
(1)FLD1B      DB    'personal  computer'
(2)FLD2B      DB    32
```

```
(3)FLD3B      DB    20H
(4)FLD4B      DB    01011001B
(5)FLD5B      DB    '32654'
(6)FLD6B      DB    10DUP(0)
(7)FLD7B      DB    'P','A','R','T','1',20,0AH,0DH
                    'P','A','R','T','2',50,0AH,0DH
                    'P','A','R','T','3',14,0AH,0DH
(8)FLD1W      DW    0FFF0H
(9)FLD2W      DW    01011001B
(10)FLD3W     DW    FLD7B
(11)FLD4W     DW    5,6,7,8,9
(12)FLD5W     DW    5DUP(0)
(13)FLD6W     EQU   $-FLD1B
```

4.8 假设程序中的数据定义如下：

```
PARTNO    DW    ?
FNAME     DB    16    DUP (?)
COUNT     DD    ?
FLENTH    EQU   $—PARTNO
```

问 PLENTH 的值为多少？它表示什么意义？

【解答】

PLENTH=22，表示 PARTNO 到 COUNT 的字节数。

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？

【解答】

1，2，3,'123'的总长度为 6，故 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 的头两个字节的内容放入 SI。

(3) 写一条伪操作使 CODE_LENGHT 的值等于 CODE_LIST 域的实际长度。

【解答】

```
(1) MOV     AX, OFFSET  LNAME
(2) MOV     SI, WORD  PTR  CODE_LIST
(3) CODE_LENGHT        EQU  5
```

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。

【解答】 程序流程图如图 4.4 所示。程序代码如下：

```
DATA_SEG    SEGMENT
COUNT       DB          5
DATA_LIST   DW          −1, 0, 2, 5, 4, 5DUP (?)
MAX         DW          ?
MIN         DW          ?
DATA_SEG    ENDS
CODE_SEG    SEGMENT
```

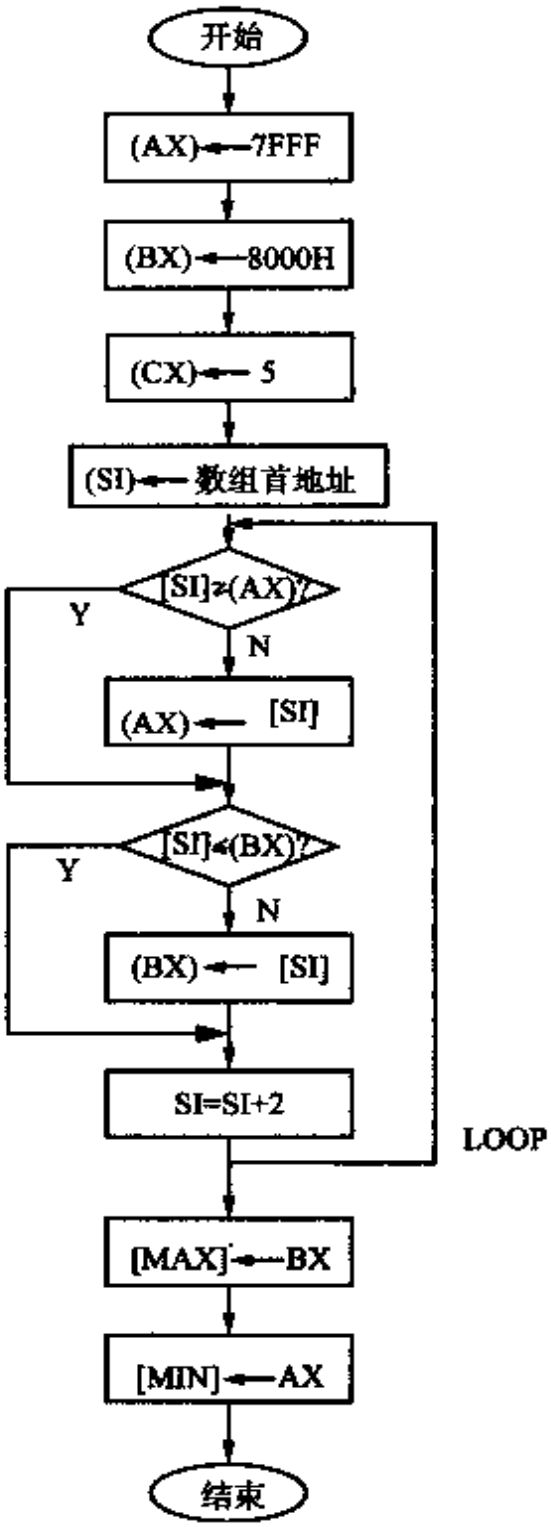

开始
(AX)←7FFF
(BX)←8000H
(CX)← 5
(SI)←数组首地址
[SI]≠(AX)?
Y
N
(AX)← [SI]
[SI]≤(BX)?
Y
N
(BX)← [SI]
SI=SI+2
LOOP
[MAX]←BX
[MIN]←AX
结束

图 4.4

```
MAIN    PROC    FAR
        ASSUME  CS: CODE_SEG, DS: DATA_SEG
START:  PUSH    DS
        XOR     AX, AX
        PUSH    AX
        MOV     AX, DATA_SEG
        MOV     DS, AX
        MOV     AX, 7FFFH      ; AX 中放最小值
        MOV     BX, 8000H      ; BX 中入最大值
        MOV     CX, 5
        LEA     SI, DATA_LIST
L3:     CMP     [SI], AX
        JGE     L1
        MOV     AX, [SI]
        JMP     L2
L1:     INC     SI
        INC     SI
        LOOP    L3
        MOV     [MAX], BX
        MOV     [MIN], AX
        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

(2) ALPHA MOD GAMMA+BETA

(3) (ALPHA+2) * BETA-2

(4) (BETA/3) MOD 5

(5) (ALPHA+3) * (BETA MOD GAMMA)

(6) ALPHA GE GAMMA

(7) BETA AND 7

(8) GAMMA OR 3

【解答】

(1) 10025 (2) 25 (3) 2548 (4) 3

(5) 103 (6) 0FFFFH (7) 1 (8) 3

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
MOV       BL, LENGTH    TABLEB
MOV       CL, LENGTH    TABLEC
```

【解答】

```
MOV    AX, 10
MOV    BL, 10
MOV    CL, 1
```

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
(2) MOV     AX, TYPE    TABLEA
(3) MOV     CX, LENGTH TABLEA
(4) MOV     DX, SIZE    TABLEA
(5) MOV     CX, LENGTH TABLEB
```

【解答】

(1) (AX) =1　(2) (AX) =2　(3) (CX) =20

(4) (DX) =40　(5) (CX) =1

4.15 指出下列伪操作表达方式的错误，并改正之。

```
(1) DATA_SEG        SEG
(2) SEGMENT         'CODE'
(3) MYDATA          SEGMENT/DATA
      ......
                    ENDS
(4) MAIN_PROC       PROC  FAR
      ......
                    END  MAIN_PROC
    MAIN_PROC       ENDP
```

【解答】

(1) 改为：DATA_SEG　SEGMENT

(2) 缺少段名。

改为：SEGMENT　SEGMENT NAME　'CODE'

(3) 结尾无段名。

改为：MYDATA　SEGMENT　'DATA'

(4) 两个结束符弄反了。

改为：MAIN_PROC　PROC　FAR

……

```
MAIN _ PROC     ENDP
END     MAIN _ PROC
```

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。

【解答】 程序代码框架如下：

```
· MODEL     SMALL
· STACK     100H
· DATA
  ORG       0E000H
  ARRAY     DB     100DUP (?)
· CODE
  ORG       1000H
  PUSH      DS
  XOR       AX, AX
  PUSH      AX
  MOV       AX, @DATA
  MOV       DS, AX
    ……
```

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 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。

【解答】 程序代码如下：

```
D_SEG    SEGMENT
AUGW     LABEL    WORD
AUGEND   DD       99251
SUMEDW   2        DUP (?)
D_SEG    SEGMENT
E_SEG    SEGMENT
ADDW     LABEL    WORD
ADDEND   DD       -15962
E_SEG    ENDS
C_SEG    SEGMENT
         ASSUME   CS: C_SEG, DS: D_SEG,
         ES: E_SEG
MAIN     PROC     FAR
START:   PUSH     DS
         MOV      AX, 0
         PUSH     AX
         MOV      AX, D_SEG
         MOV      DS, AX
         MOV      AX, E_SEG
         MOV      ES, AX
         MOV      AX, AUGW
         MOV      BX, AUGW+2
         ADD      AX, ES: ADDW
         ADC      BX, ES: ADDW+2
         MOV      SUM, AX
         MOV      SUM+2, BX
         RET
```

```
MAIN    ENDP
CSEG    ENDS
END     START
```

4.18 请说明表示程序结束的伪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？

【解答】 程序结束伪操作：END [START]。

汇编程序将在遇到 END 时结束汇编，结束程序执行语句为 RET，返回调用该程序。

4.19 试说明下述指令中哪些需要加上 PTR 伪操作。

```
BVAL    DB    10H，20H
WVAL    DW    1000H
(1) MOV    AL，BVAL
(2) MOV    DL，[BX]
(3) SUB    [BX]，2
(4) MOV    CL，WVAL
(5) ADD    AL，BVAL+1
```

【解答】

第（4）条，指令中两操作数长度不匹配，则须加上 PTR。

4.4 强化训练

1. 分析以下程序的运行情况，填空，并写出最终结果。

```
(1)    DSEG    SEGMENT
       ORG     2
       VAL1    DB    30H，36H
       ORG     7
       VAL2    DB    35H，38H
```

```
        LEN1    EQU   $-VAL2
        VAL3    DB    39H, 32H
        DSEG    ENDS
        CSEG    SEGMENT
        ASSUME  CS: CSEG, DS: DSEG
START:  MOV     AX, DSEG
        MOV     DS, AX
        MOV     DL, [VAL1+1]      ; (DL) =____
        MOV     AH, 2
        INT     21H
        MOV     DI, OFFSET  VAL3
        MOV     DL, [DI+1]        ; (DL) =____
        MOV     AH, 2
        INT     21H
        MOV     DL, DS: [LEN1]    ; LEN1=____
                                  (DL) =____
        MOV     AH, 2
        INT     21H
        LEA     AX, VAL2          ; (AL) =___
        OR      AL, 30H           ; (AL) =___
        MOV     DL, AL
        MOV     AH, 2
        INT     21H
        RET                       ; 结束程序，返回 DDS
        CSEG    ENDS
```

```
            END     START
(2)         DSEG    SEGMENT
            ORG     100H
            NUM1=8
            NUM2=NUM1+10H
            DA1     DB      'IBM PC'
            DB      0AH, 0DH
            COUNT   EQU     $-DA1
            DA2     DW      'IB','M','PC', 0A0DH
            ORG     134H
            NUM3=20H
            DA3     DW      10H, $+20H, 30H, $+40H
            DA4     DW      DA1+NUM3+14H
            DSEG    ENDS
            CSEG    SEGMENT
            ASSUME  CS:CSEG, DS:DSEG
START:      MOV     AX, DSEG
            MOV     DS, AX
            MOV     AX, OFFSET (DA1+1)
                                        ; (AX) =_____
            MOV     BL, LOW OFFSET DA2
                                        ; (BL) =_____
            MOV     CX, CUNT            ; (CX) =_____
            MOV     DX, DA2+5           ; (DX) =_____
            MOV     AX, DA3 [2]         ; (AX) =_____
```

```
        MOV     BX, DA3 [6]        ; (BX) =_____
        MOV     SI, DA4
        MOV     CX, [SI]           ; (CX) =_____
        RET                        ; 结束程序，返回
                                   DOS

CSEG    ENDS
        END     START
```

2. 改正下列语句的语法错误。

```
(1) DATA1   EQU  10H
      ……
    MOV     DATA1, BL
(2) DATA2   DW   'ABCDEFG'
(3) DATA3   DB   2008H
(4) DSEG1   SEGMENT
    VAR1    DW   2008H
    DSEG1   ENDS
    DSEG2   SEGMENT
    VAR2    DW   2008H
    DSEG2   ENDS
    CSEG    SEGMENT
    ASSUME CS: CSEG, DS: DSEG2
    START:
      ……
    MOV     AX, VAR1
      ……
```

```
CSEG    ENDS
END     START
```

答 案

1. (1) (DL) =36H
   (DL) =10H
   (LEN1) =02H
   (DL) =1EH
   (AL) =35H
   (AL) =35H

最终显示结果为：6
0

(2) (AX) =0101H
(BL) =0008H
(CX) =0008H
(DX) =0D50H
(AX) =0156H
(BX) =017AH
(CX) =0010H

没有显示结果

2. (1) MOV BL, DATA1
   (2) DATA2 DB 'ABCDEFG'
   (3) DATA3 DW 2008H
   (4) ASSUME CS: CSEG, DS: DSEG1, ES: DSEG2

第5章 循环与分支程序设计

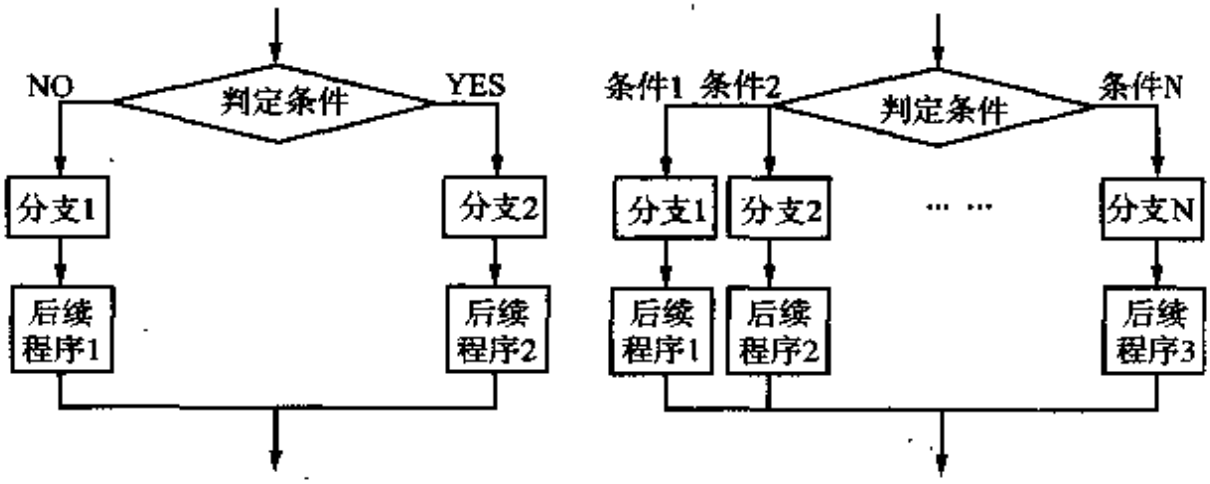
5.1 内容提要

5.1.1 分支结构程序设计

分支程序是一种重要的程序结构，在汇编语言中一般要根据比较或测试的结果，再利用条件转移语句实现分支结构。

1. 分支程序的结构形式

分支程序结构有两种形式，如图 5.1 所示。它们分别相当于高级语言中的 IF_ELSE 语句和 CASE 语句，适用于需要根据不同条件做不同处理的情况。其中 IF_ELSE 语句可以引出两个分支，



(a) IF_ELSE 结构　　(b) CASE 结构

图 5.1 分支程序的结构形式

CASE 语句则可以引出多个分支，不论哪一种形式，它们的共同特点是：执行方向是向前的，在某一种确定条件下，只能执行多个分支中的一个分支。

2. 分支程序的设计方法

(1)分支程序的设计方法。分支程序设计用我们之前介绍的转移指令来实现。分支程序的流程结构为 IF_ELSE 形式。

(2)多支程序的设计方法。多分支程序结构设计的实现过程相当于 CASE 语句形式。最直接的方法就是使用简单的分支结构进行组合，使用条件转移指令结合无条件转移指令来实现，如图 5.2 所示。

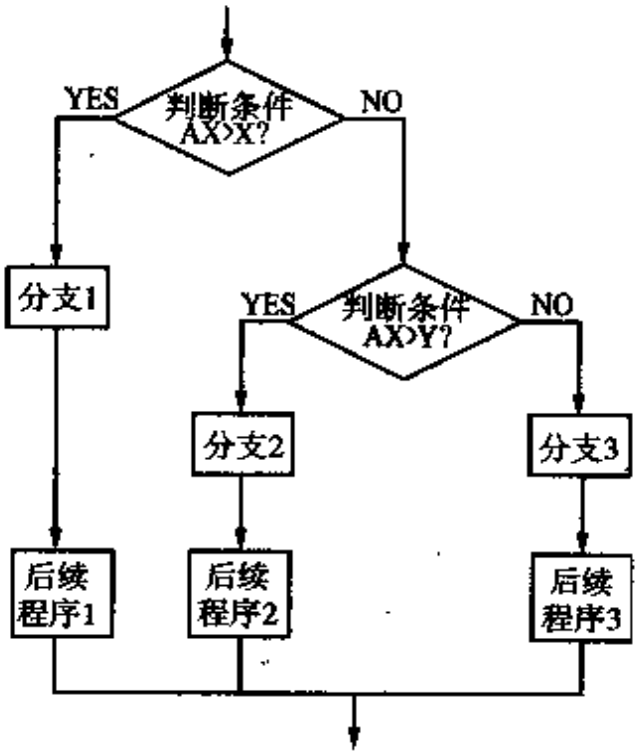


图 5.2　三分支结构的汇编程序实际流程

5.1.2　循环结构程序设计

循环程序是强制 CPU 重复执行某一指令系列(程序段)的一

种程序结构形式，凡是要重复执行的程序段都采用循环结构设计。循环结构程序设计方法简化了程序清单书写形式，而且减少了内存空间占用。但循环程序并不简化程序执行过程，相反，它增加了一些循环控制等环节，总的程序执行语句和时间会有所增加。

1. 循环程序的结构形式

循环程序可以有两种结构形式，如图 5.3 所示：一种是 DO_WHILE 结构；另一种是 DO_UNTIL 结构。

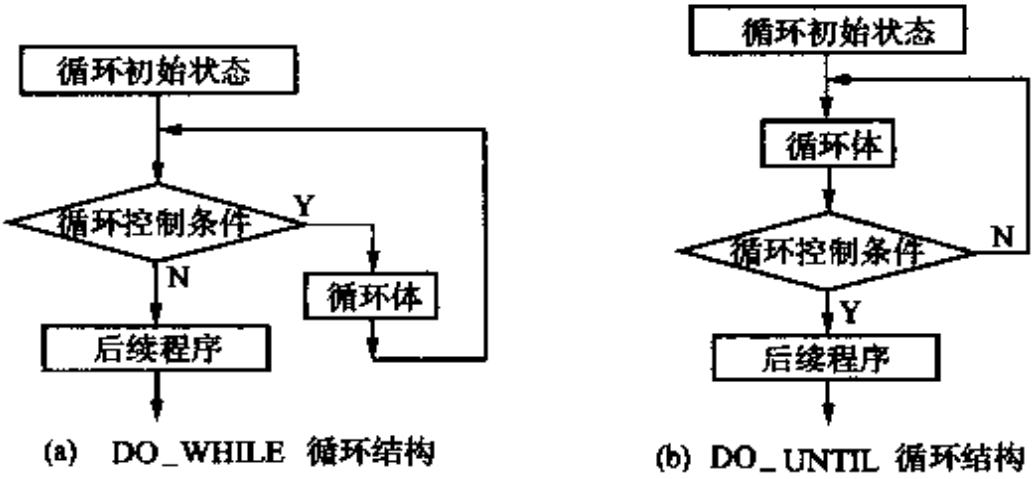


图 5.3 循环程序的结构形式

DO_WHILE 结构的特点是把对循环控制条件的判断放在循环的入口，先判断条件，满足条件就执行循环体，否则退出循环。DO_UNTIL 结构的特点是先执行循环体，然后再判断控制条件，不满足条件则继续执行操作，一旦满足条件则退出循环。

无论是哪一种结构，循环程序都包括四部分，即初始化部分、循环工作部分、控制部分和结束处理部分。

(1)初始化部分：为循环做准备工作。如设置地址指针、计数器及其他变量的初值等。

(2)循环工作部分：它是循环程序的主体。用来完成循环的基本操作，修改循环参数。

(3)控制部分：根据循环条件来判断、控制循环的继续和终止。循环控制方式有多种，如计数控制、条件控制、状态控制等。不管

哪一种循环控制方式，最终都是要达到控制循环的目的。

(4)结束处理：它主要用来分析和存放程序的结果。

一般说来，如果有循环次数等于 0 的可能，则应选择 DO_WHILE 结构，即“先判断后处理”，否则使用 DO_UNTIL 结构，即“先处理后判断”。

2. 循环程序设计方法

按循环的嵌套层次，循环程序又可分为单循环程序和多重循环程序两种。单循环程序的循环体内只是一些简单的分支程序，多重循环程序的循环体内还包含一个循环程序。

(1)单循环程序设计

(2)多重循环程序设计

单重循环的结构常常难以解决实际应用问题，所以人们引入了多重循环。这些循环是一层套一层的，因此又称为循环的嵌套。多重循环和单重循环的设计方法是一致的，应分别考虑各重循环控制条件及其程序的实现，相互之间不要混淆。应特别注意的几点是：

①内循环必须完整地包含在外循环内，内外循环不能相互交叉；

②内循环在外循环中的位置可根据需要任意设置，但应避免出现混乱；

③当通过外循环再次进入内循环时，内循环中的初始条件必须重新设置；

④多个内循环可以拥有一个外循环，这些内循环间的关系可以是嵌套的，也可以是并列的。

5.1.3 基本 DOS 功能子程序调用

DOS 系统为用户准备了一些具有特定功能的子程序供用户调用，DOS 功能于程序的调用方法如下：

(ⅰ)送入口参数到指定的寄存器；

(ⅱ)AH←功能号；

(ⅲ)INT 21H。

下面举例说明几个常用的输入输出功能子程序的调用：

(1)1 号调用(带回显的键盘输入)。

该功能调用是从键盘输入一个字符，并在屏幕显示该字符。

(2)2 号调用(在屏幕上显示一个字符)。

该功能调用是在屏幕上显示一个字符，入口参数为 DL，用来存放要显示字符的 ASCII 码。

(3)8 号调用(不带回显的键盘字符输入)。

该功能与 1 号调用相似，执行 8 号调用后，程序会等待键盘输入，但不在屏幕显示输入字符，出口参数在 AL 中，存放输入字符的 ASCII 码。

(4)9 号调用(显示字符串)。

该功能调用是在屏幕显示一个字符串。执行 9 号调用前，需要置入口参数 DX。DX 存放需要显示字符串的首地址，字符串必须先定义在数据区 DS:DX 中，且必须以 $ 作为结束字符。

(5)5 号调用(送字符到打印机打印字符)。

5 号功能调用是将要打印的字符 ASCII 码送 DL，向打印机输出一个字符。

5.1.4 考 点

顺序、分支、循环程序的设计。

5.2 例题精解

例 1 求$[X]_{补}$的绝对值，并送回原处。

【分析与解答】

补码的定义

$$[X]_{补}=\begin{cases}X & X\geqslant 0\\ 2^n+X=-|X| & X<0\end{cases}$$

显然本程序的结构为分支结构，当 X≥0 时，$|X|=[X]_{补}$；当 X<0 时，要对$[X]_{补}$进行求补运算，要用到 NEG 指令，其功能是对操作数求补(连同符号位一起，按位求返，且在最低位加 1)，再把结果送到原地址中去。

程序的流程图如图 5.4 所示。

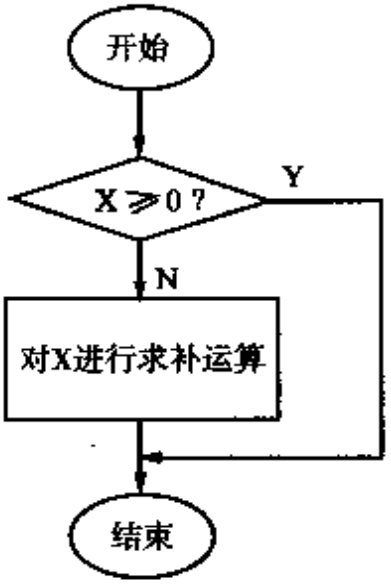


图 5.4

程序代码如下：

```
DSEG    SEGMENT
MUM     DW      2008
DSEG    ENDS
CSEG    SEGMENT
        ASSUME CS:CSEG,DS:DSEG
MAIN    PROC    FAR
```

```
        PUSH    DS
        XOR     AX,AX
        PUSH    DS,DSEG
        MOV     DS,AX
        MOV     AX,NUM
        OR      AX,AX       ;设置条件标志
        JNS     DONE        ;若 X≥0,转 DONE
        NEG     AX          ;若 X<0,求补
        MOV     NUM,AX
DONE:   RET
MAIN    ENDP
CSEG    ENDS
        END     MAIN
```

例 2 求字符串的长度。从 STRN 地址开始有一个字符串，以'$'作为结束标志，长度不超过 100 个字节，要求统计该字符串长度并存于 LENG 单元。

【分析与解答】

设 DX 存放统计的串长度，为了防止程序死循环，故可根据字符串长度不会超过 100 作为循环结束的附加条件。如果程序运行过程中找不到'$'，由于 CX 的初始值为 100，故不会使程序陷入死循环。

程序代码如下：

```
DSEG   SEGMENT
```

```
STRN    DB    'ABCDEFG…','$'
LENG    DB    0
DSEG    ENDS
CSEG    SEGMENT
        ASSUME CS:CSEG,DS:DSEG
START:  MOV   AX,DSEG
        MOV   DS,AX
        MOV   AX,0
        MOV   DX,AX   ;DX 清零
        LEA   DI,STRN
        MOV   CX,100  ;初始值为 100
        MOV   AL,'$'
LOP:    CMP   AL,[DI]
        JE    DONE    ;若为结束标志转移
        INC   DX      ;若不是继续统计
        INC   DI
        LOOP  LOP
DONE:   MOV   LENG,DL ;存字符串长度
CSEG    ENDS
        END   START
```

例3 求学生的平均成绩。

【分析与解答】

设有5门不同的课程，5个学生对应这5门课程的成绩定义为一张成绩表。因为成绩采用百分制，所以成绩的数据类型为字节型。算出来的每门课程的平均成绩存入字节型变量中。程序应采用双重循环。外层控制课程门数，内层用于计算每门课程的累加分。程序的流程图如图5.5所示。

```
DSEG    SEGMENT
GRADE   DB      98,78,100,65,78
        DB      56,65,79,97,100
        DB      20,0,0,0,8
        DB      90,80,70,60,50
        DB      91,92,93,94,95
        AVEBUFDB    5DUP(?)
DSEG    ENDS
        CSEG    SEGMENT
        ASSUMECS:CSEG,DS:DSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        MOV     BX,OFFSET  GRADE
        MOV     DI,OFFSET  AVEBUF
        MOV     SI,BX
        MOV     CX,5;外循环计数值,即课程门数
A1:     MOV     BX,SI
        SUB     BX,5
        PUSH    CX
        MOV     AL,0
```

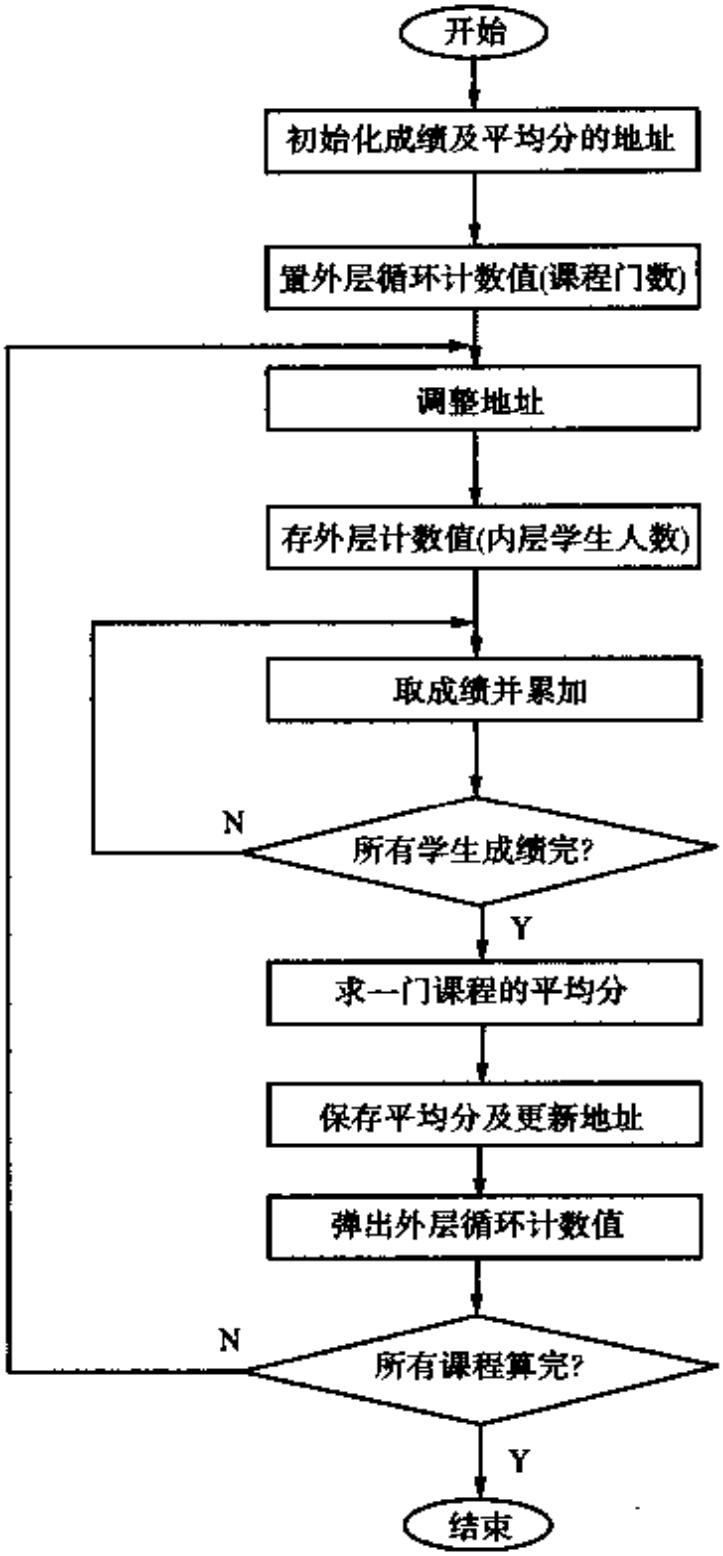

开始
初始化成绩及平均分的地址
置外层循环计数值(课程门数)
调整地址
存外层计数值(内层学生人数)
取成绩并累加
N
所有学生成绩完?
Y
求一门课程的平均分
保存平均分及更新地址
弹出外层循环计数值
N
所有课程算完?
Y
结束

图 5.5

```
        MOV     CX,5;内层循环计数值,即学生人数
A2:     ADD     BX,5;同一门课程地址更新
        ADD     AL,[BX]
        LOOP    A2
        MOV     AH,0
        MOV     DL,5;人数赋给 DL
        DIV     DL;求一门课程的平均成绩
        MOV     [DI],AL;存平均分
        INC     SI
        INC     DI
        POP     CX;弹出外层循环计数值
        LOOP    A1
        RET
CSEG    ENDS
        END  START
```

5.3 课后习题解答

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。

【解答】 程序的流程图如图 5.6 所示。

主要程序代码如下:

```
START:MOV       AH,1
      INT       21H
      CMP       AL,'a'
      JB        RET1
      CMP       AL,'z'
```

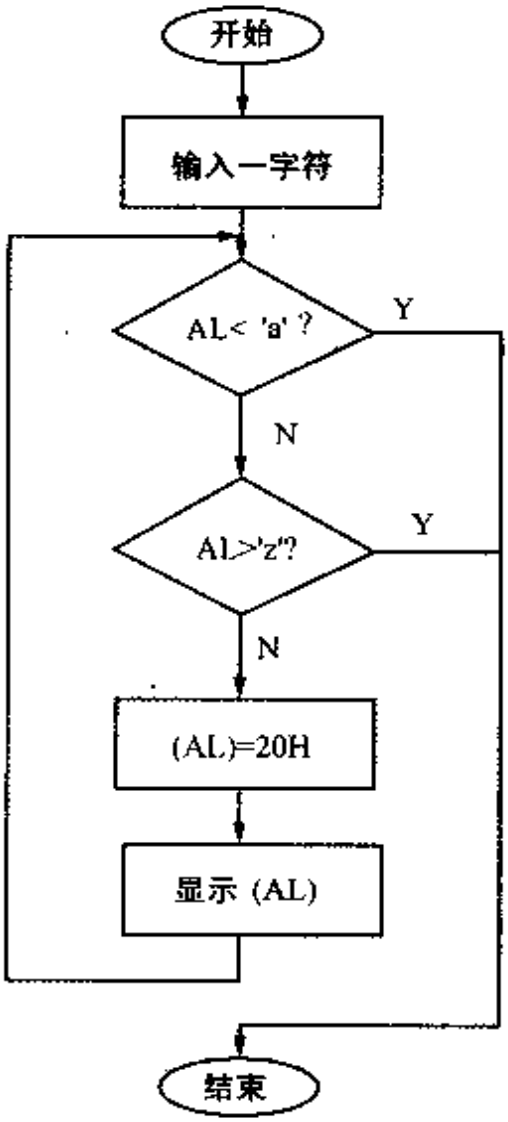


图 5.6

```
        JA      RET1
        SUB     AL,20H
        MOV     DL,AL
        MOV     AH,2
        INT     21H
        JMP     START
RET1:   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。

【解答】 程序的流程图如图 5.7 所示。

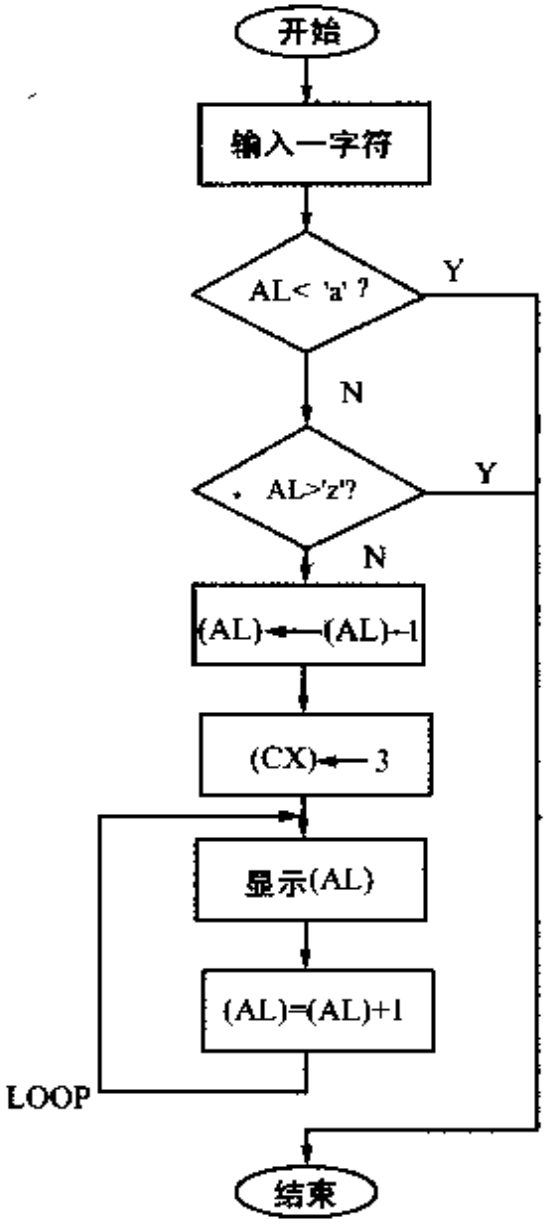


图 5.7

主要程序代码如下：

```
START:MOV     AH,1
      INT     21H
      CMP     AL,'a'
      JB      RET1
```

```
        CMP     AL,'z'
        JA      RET1
        DEC     AL
        MOV     DL,AL
        MOV     CX,3
DISP:   MOV     AH,2
        INT     21H
        INC     DL
        LOOP    DISP
RET1:   RET
```

5.3 将 AX 寄存器中的 16 位数分成 4 组，每组 4 位，然后把这四个组数分别放在 AL、BL、CL 和 DL 中。

【解答】 程序的流程图如图 5.8 所示。

程序代码如下：

```
DSEG    SEGMENT
TEMP    DB  4DUP(?)
DSEG    ENDS
        ......
START:  MOV  CL,4
        MOV  CH,4
        LEA  SI,TEMP
L1:     MOV  DX,AX
        AND  DX,0FH
        MOV  BYTE  PTR[SI],DL
        INC  SI
        SHR  AX,CL
        DEC  CH
```

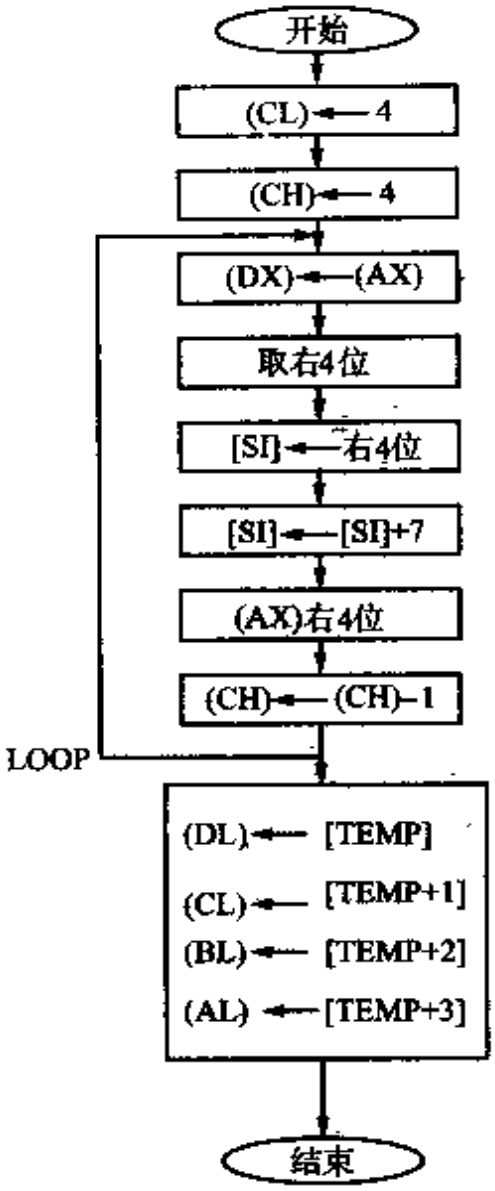


图 5.8

```
JNZ   L1
MOV   DL,TEMP
MOV   CL,TEMP+1
MOV   BL,TEMP+2
MOV   AL,TEMP+3
RET
```

5.4 试编写一程序，要求比较两个字符串 STRING1 和

STRING2 所含字符是否相同，若相同则显示'MATCH'，若不相同则显示'NO MATCH'。

【解答】 程序的流程图如图 5.9 所示。

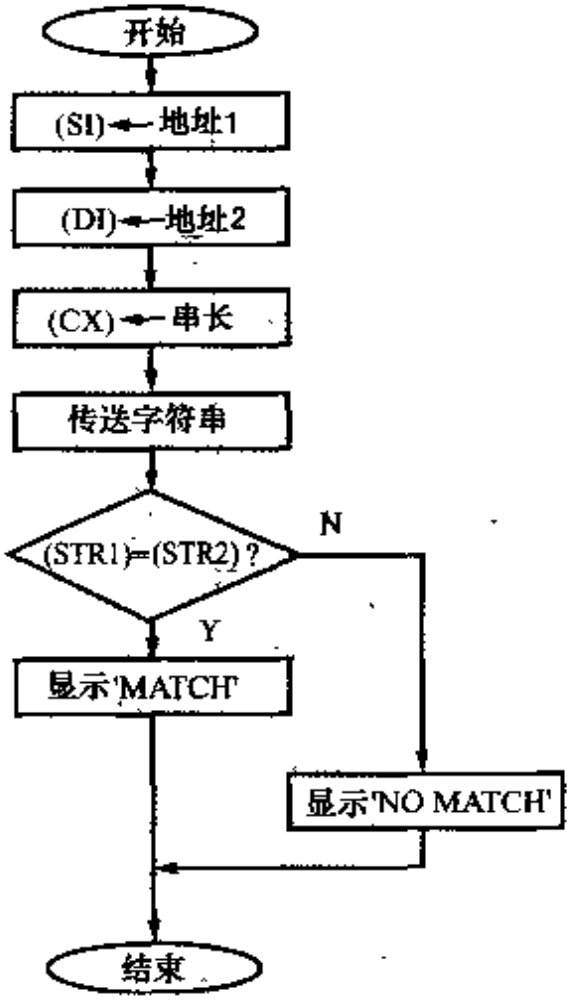


图 5.9

程序代码如下：

```
DSEG    SEGMENT
        STR1    DB  'abc'
        STR2    DB  'abd'
        M       DB  'MATCH',0DH,0AH,'$'
        NM      DB  'NO  MATCH',0DH,0AH,'$'
DSEG      ENDS
```

```
CSEG    SEGMENT
MAIN    PROC  FAR
        ASSUME CS:CSEG,DS:DSEG,ES:ESEG
START:  PUSH   DS
        SUB    AX,AX
        PUSH   AX
        MOV    AX,DSEG
        MOV    DS,AX
        MOV    ES,AX
        LEA    SI,STR1
        LEA    DI,STR2
        MOV    CX,STR2-STR1
        REPE   CMPSB
        JNE    DISPNM
        MOV    AH,09H
        LEA    DX,M
        INT    21H
        RET
DISPNM: MOV    AH,09H
        LEA    DX,NM
        INT    21H
        RET
MAIN    ENDP
CSEG    ENDS
        END    START
```

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

【解答】 主要程序代码如下：

```
        MOV     AH,1
        INT     21H
        AND     AL,0FH
RING:   MOV     DL,07H
        MOV     AH,02H
        INT     21H
        DEC     AL
        JNZ     RING
        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。

【解答】 程序的流程图如图 5.10 所示。

程序代码如下：

```
DSEG    SEGMENT   PARA   PUBLIC'DSEG'
M       DB        X1,X2,…,X20
P       DB        20 DUP(?)
N       DB        20 DUP(?)
DSEG    ENDS
        ……
        LEA       SI , P
        LEA       DI , N
        LEA       BX , M
        XOR       AX , AX
        XOR       DX , DX
        MOV       CX ,20
L1:     MOV       AL ,[BX]
        TEST      AL ,80H
```

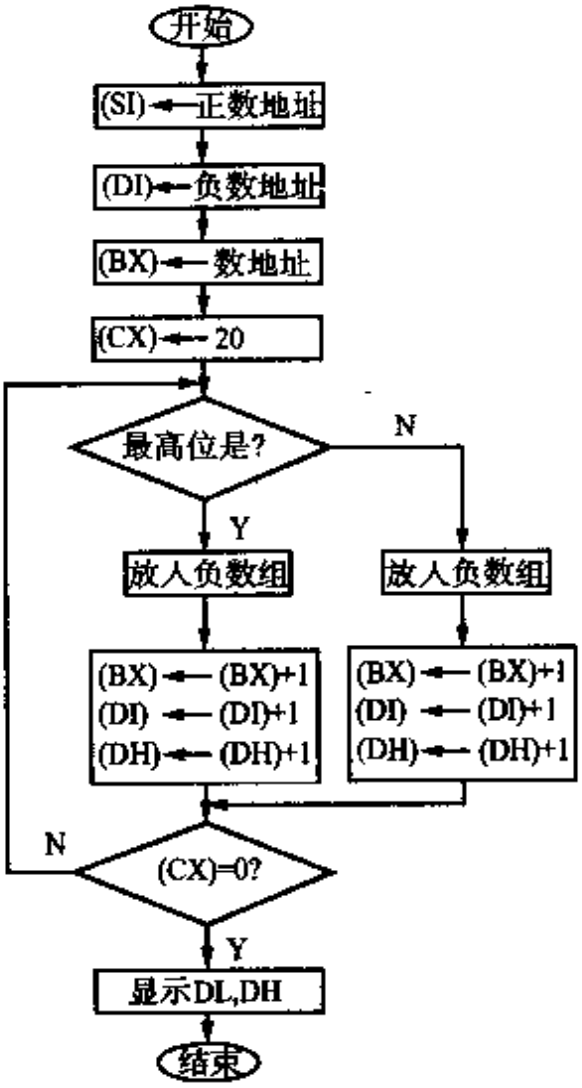


图 5.10

```
        JZ      L2
        MOV     [DI],AL
        INC     BX
        INC     DI
        INC     DH
        JMP     L3
L2 :    MOV     [SI],AL
        INC     BX
        INC     SI
```

```
        INC     DL
L3:     LOOP    L1
        MOV     CX ,1
L5:     CMP     DL ,10
        JB      L4
        ADD     DL ,7
L4:     ADD     DL ,30H
        MOV     AH ,2
        INT     21H
        MOV     DL ,DH
        LOOP    L5
```

5.7 试编制一个汇编语言程序，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。

【解答】 程序的流程图如图 5.11 所示。

主要程序代码如下：

```
DSEG    SEGMENT
DATA    DW          X1,…,X100
MIN     DW          0FFFH
          ……
        LEA         BX,DATA
        MOV         CX,100
L1:     MOV         AX,[BX]
        TEST        AX,01H
        JNZ         L2
        CMP         AX,[MIN]
        JNB         L2
        MOV         MIN,AX
L2:     ADD         BX,2
```

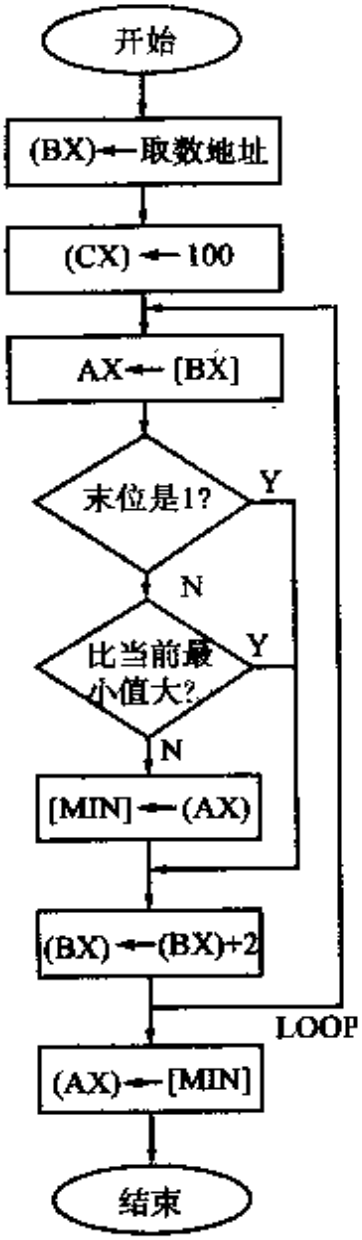


图 5.11

```
LOOP        L1
MOV         AX,[MN]
.....
```

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”。试编写一程序，要求数一下值为 3(即 11B)的四分之一字节数，并将该数在终端上显示出来。

【解答】 程序的流程图如图 5.12 所示。

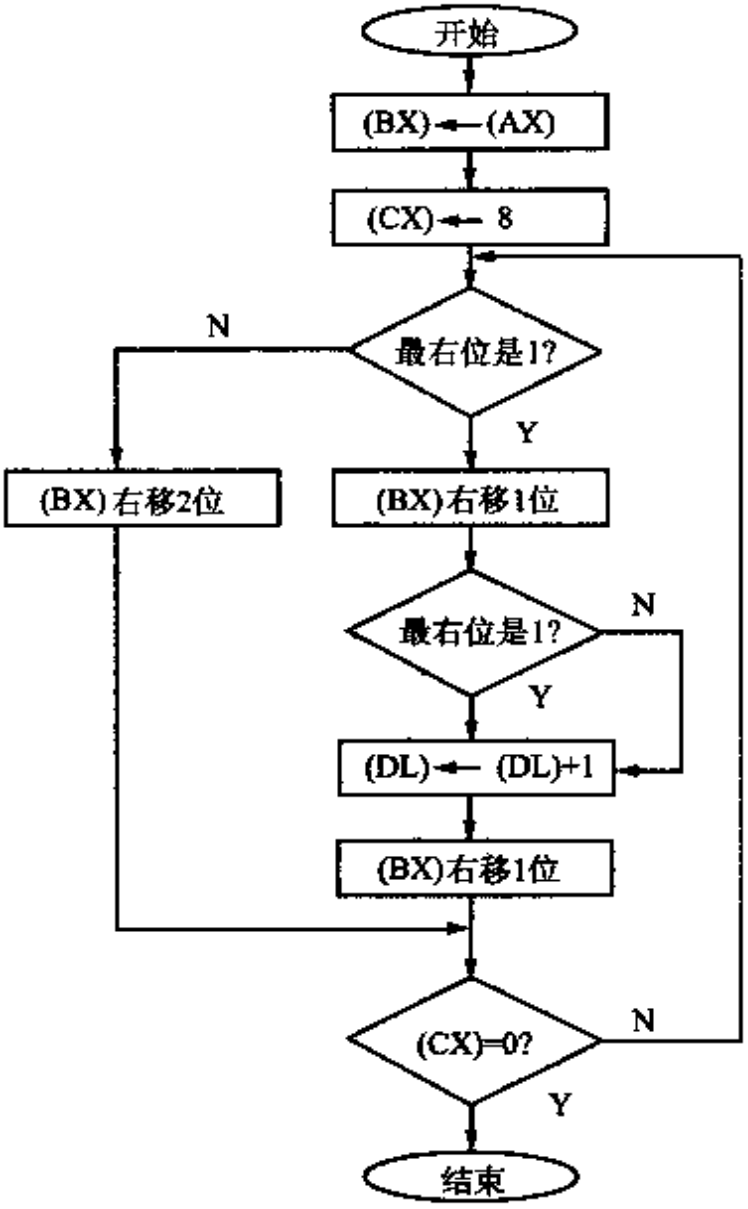


图 5.12

主要程序代码如下：

```
        MOV     DL,0
        MOV     BX,AX
        MOV     CX,8
L0:     TEST    BX,1
        JZ      L2
```

```
        SHR     BX,1
        TEST    BX,1
        JZ      L1
        INC     DL
L1:     SHR     BX,1
        LOOP    L0
        RET
L2:     SHR     BX,1
        SHR     BX,1
        LOOP    L0
        RET
```

5.9 试编写一汇编语言程序，要求从键盘接收一个四位的十六进制数，并在终端上显示与它等值的二进制数。

【解答】 主要程序代码如下：

```
        MOV     BX,0
        MOV     CL,4
        MOV     CH,4
L0:     SHL     BX,CL
        MOV     AH,1
        INT     21H
        CMP     AL,39H
        JA      L1
        AND     AL,0FH
        JMP     L2
L1:     AND     AL,0FH
        ADD     AL,9
L2:     OR      BL,AL
        DEC     CH
```

```
        JNE     L0
        MOV     CX,16
L3:     MOV     DL,0
        ROL     BX,1
        RCL     DL,1
        OR      DL,30H
        MOV     AH,2
        INT     21H
        LOOP    L3
        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，查出单词 SUN 在该文中的出现次数，并以格式“SUN××××”显示出次数。

【解答】 程序的流程图如图 5.13 所示。

程序代码如下：

```
DSEG    SEGMENT
ENG     DB      'SUN IS SUN $'
DISP    DB      'SUN'
COUNT   DB      4DUP(11),'$'
FIND    DB      'SUN'
DSEG    ENDS
          ……
        PUSH    DS
        XOR     AX,AX
        PUSH    AX
        MOV     AX,DSEG
        MOV     DS,AX
        MOV     ES,AX
```

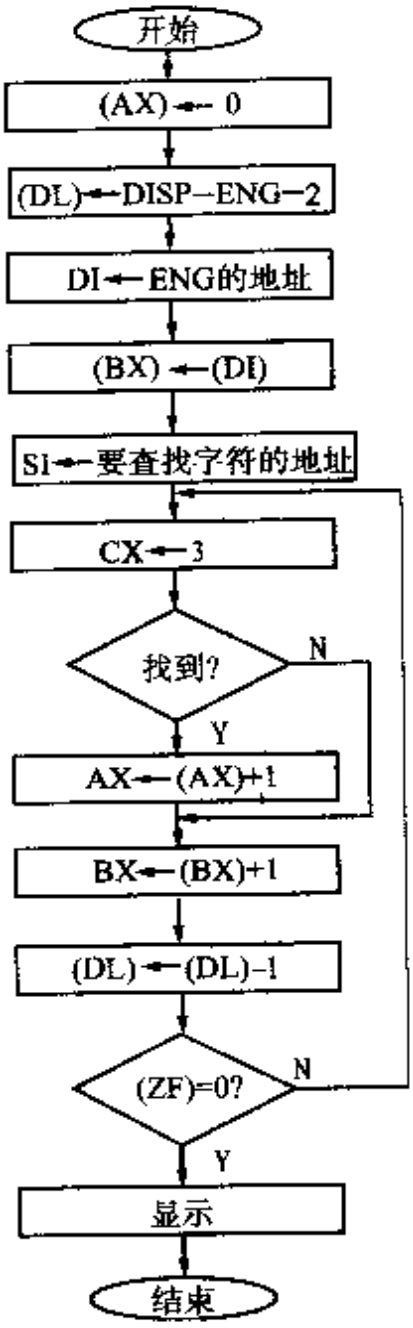


图 5.13

```
        MOV     AX,0
        MOV     DL,DISP-ENG-2
        LEA     DI,ENG
        LEA     SI,FIND
L1:     MOV     CX,3
```

```
        REPE    CMPSB
        JNZ     L2
        INC     AX
L2:     INC     DI
        DEC     DL
        JNZ     L1
L3:     MOV     CH,4
        MOV     CL,4
        LEA     BX,COUNT
L4:     ROL     AX,CL
        MOV     DX,AX
        AND     DX,0FH
        ADD     DL,30H
        CMP     DL,39H
        JLE     L5
        ADD     DL,07H
L5:     MOV     [BX],DL
        INC     BX
        DEC     CH
        JNZ     L4
L6:     LEA     DX,DISP
        MOV     AH,09H
        INT     21H
        RET
```

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。

【解答】 首先用 SI 记录 BUFF 首地址，用 BX 记录非数字字符的个数，将输入字符存入 BUFF 开始的内存中，当遇到'$'结束

输入，否则循环输入字符；输入结束后用SI记录BUFF首地址，从BUFF处取已存储的字符判断是不是'$'，若不是，则判断是不是非数字字符，并用BX记录非数字字符个数；若是'$'，则显示BX内容，并结束程序。

程序的流程图如图5.14所示。

主要程序代码如下：

```
        LEA     SI,BUFF
        MOV     BX,0
INPUT:  MOV     AH,01
        INT     21H
        MOV     [SI],AL
        INC     SI
        CMP     AL,'$'
        JNZ     INPUT
        LEA     SI,BUFF
L1:     MOV     CL,[SI]
        INC     SI
        CMP     CL,'$'
        JZ      L3
        CMP     CL,39H
        JA      L2
        CMP     CL,30H
        JAE     L1
L2:     INC,BX
        JMP     L1
L3:     MOV     COUNT,BX
        MOV     CH,4
L4:     MOV     CL,4
```

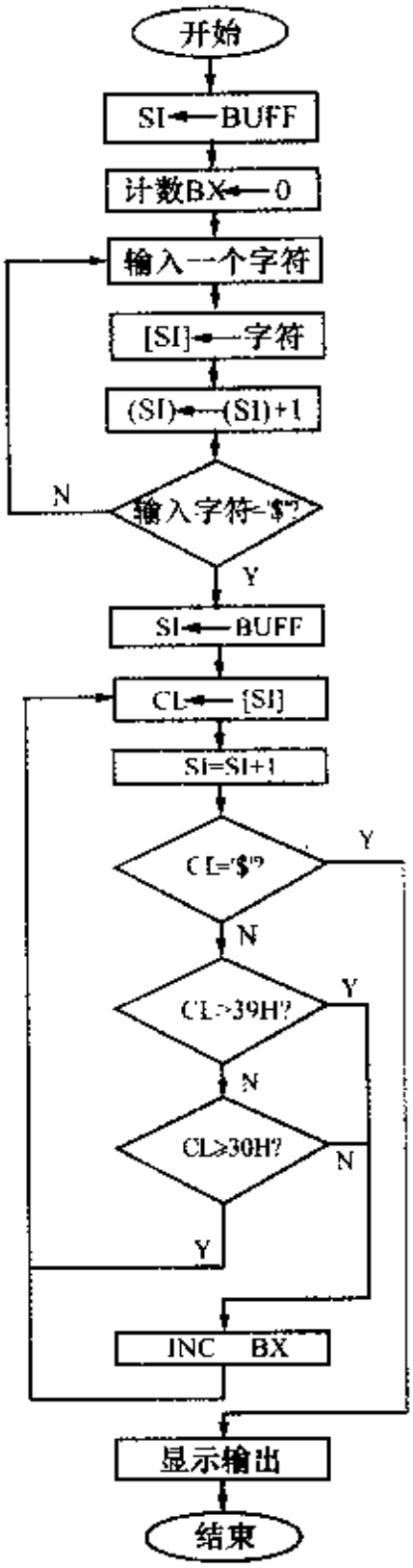

开始
SI←BUFF
计数BX←0
输入一个字符
[SI]←字符
(SI)←(SI)+1
输入字符='$'?
N
Y
SI←BUFF
CL←[SI]
SI=SI+1
CL='$'?
Y
N
CL>39H?
Y
N
CL>30H?
N
Y
INC BX
显示输出
结束

图 5.14

```
        ROR     BX,CL
        MOV     DX,BX
        AND     DX,0FH
        ADD     DL,30H
        CMP     DL,39H
        JLE     L5
        ADD     DL,07H
L5:     MOV     AH,02H
        INT     21H
        DEC     CH
        JNZ     L4
```

5.12 有一个首地址为 MEM 的 100D 字数组，试编制程序删除数组中所有为零的项，并将后续项向前压缩，最后将数组的剩余部分补上零。

【解答】 首先将要存储的字符的首地址存入 SI 中，输入一个字符判断是不是回车，若不是则继续输入字符；若是，则将记数用的寄存器 BX、CX 清零，判断字符的类型，用 BL 记录数字字符个数，用 BH 记录大写字母个数，用 CL 记录小写字母个数。直到遇到回车符为止，分别显示 BL，BH，CL 中的内容并结束程序。

程序的流程图如图 5.15 所示。

主要程序代码如下：

```
        MOV     SI,(100-L)*2
        MOV     DI,-2
        MOV     CX,100
L1:     ADD     DI,2
        CMP     MEM[DI],0
        JZ      L2
        LOOP    L1
```

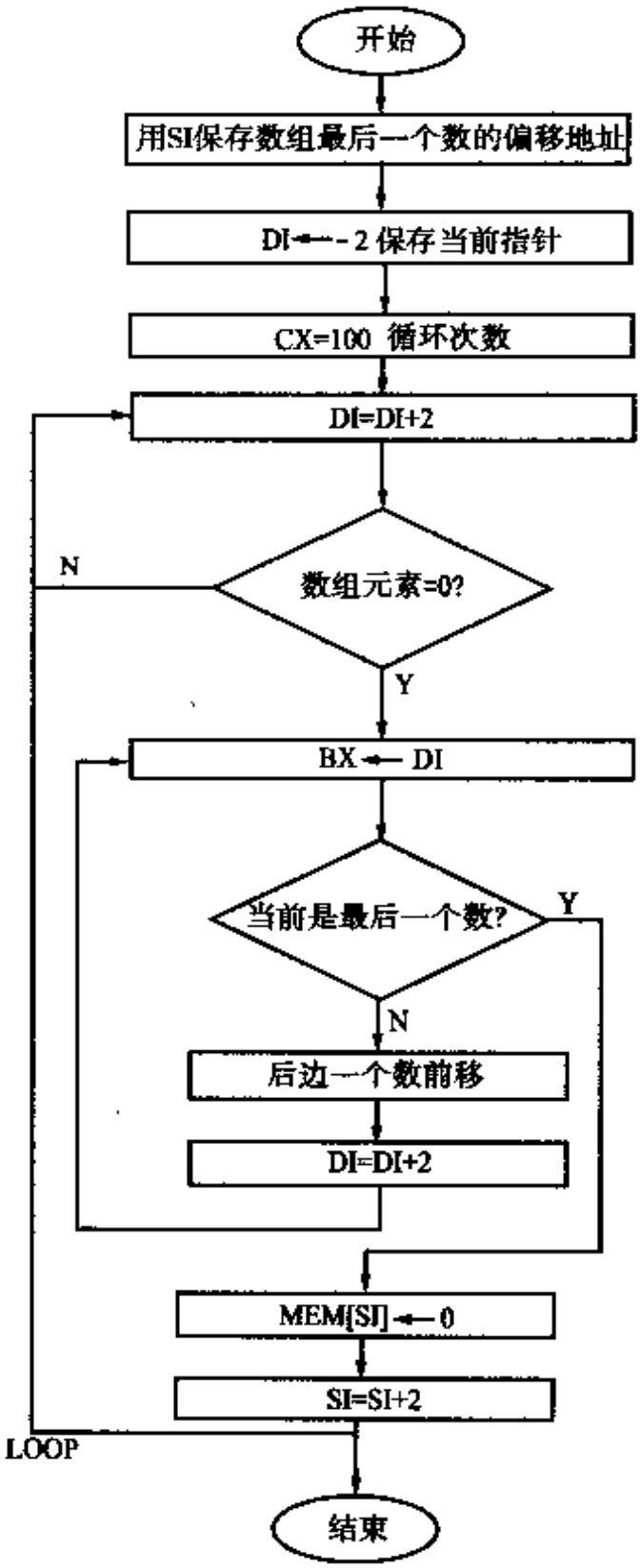

开始
用SI保存数组最后一个数的偏移地址
DI←-2 保存当前指针
CX=100 循环次数
DI=DI+2
N
数组元素=0?
Y
BX←DI
当前是最后一个数?
Y
N
后边一个数前移
DI=DI+2
MEM[SI]←0
SI=SI+2
LOOP
结束

图 5.15

```
        JMP     RET1
L2:     MOV     BX,DI
        CMP     BX,SI
        JAE     L3
        MOV     AX,MEM[BX+2]
        MOV     MEM[BX],AX
        ADD     DI,2
        JMP     L2
L3:     MOV     WORD PTR MEM[SI]
        ADD     SI,2
        LOOP    L1
RET1:   RET
```

5.13 在 STRING 到 STRING+99 单元中存放着一个字符串，试编制一程序测试：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。

【解答】 程序的流程图如图 5.16 所示。

主要程序代码如下：

```
        MN      CX,100
        LEA     SI,STRING
L0:     MOV     AL,[SI]
        CMP     AL,30H
        JB      L1
        CMP     AL,39H
        JA      L1
        OR      DL,20H
        JMP     RET1
L1:     INC     SI
        LOOP    L0
```

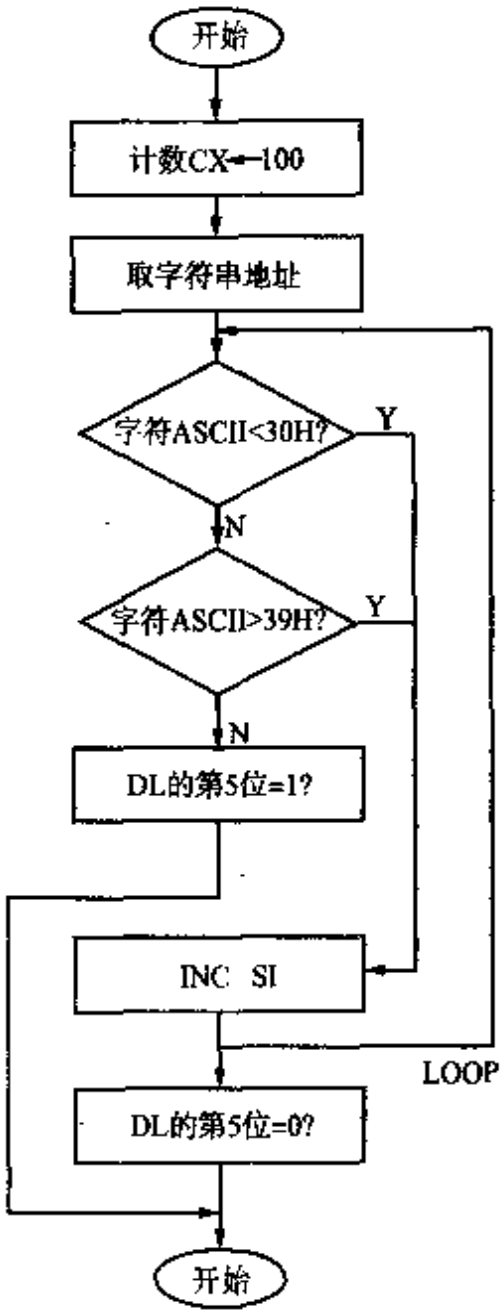


图 5.16

```
        AND     DL,0DFH
RET1:   RET
```

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。

【解答】 程序的流程图如图 5.17 所示。

程序代码如下：

```
        .MODEL SMALL
        .STACK 100H
        .DATA
TABLE   DW  100H  DUP(?)
DATA    DW  ?
COUNT   DW  0
        .CODE
MAIN    PROC  FAR
        PUSH  DS
        XOR   AX,AX
        PUSH  AX
        MOV   AX,@DATA
        MOV   DS,AX
        MOV   ES,AX
        LEA   DI,TABLE
        MOV   AX,0
        MOV   CX,100H
L0:     MOV   SI,DI
        ADD   DI,2
        MOV   DX,[SI]
L1:     CUP   DX,[DI]
        JNZ   L2
        INC   AX
        ADD   DI,2
        DEC   CX
        JNZ   L1
```

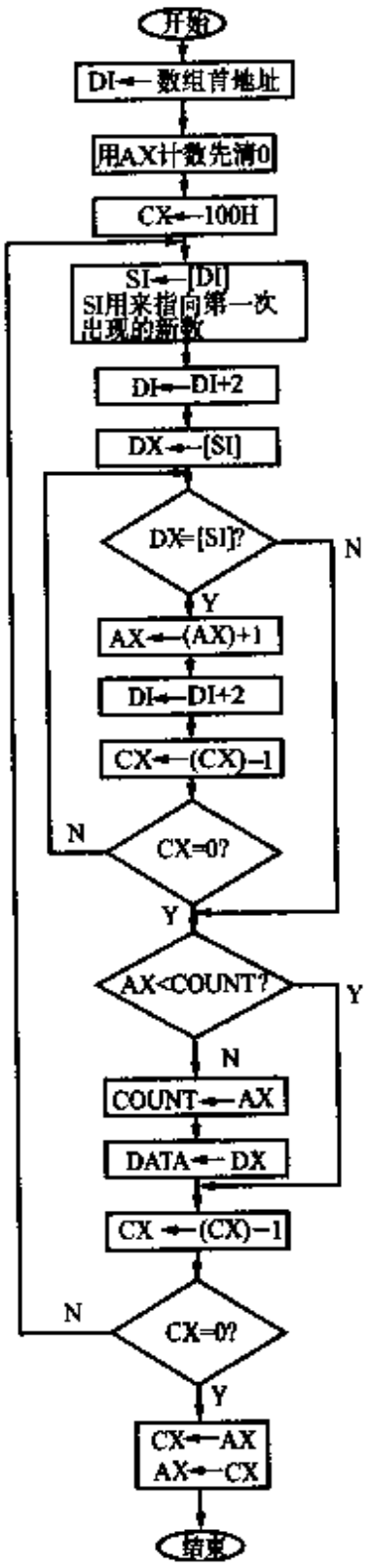

开始
DI←数组首地址
用AX计数先清0
CX←100H
SI←[DI]
SI用来指向第一次
出现的新数
DI←DI+2
DX←[SI]
DX=[SI]?
N
Y
AX←(AX)+1
DI←DI+2
CX←(CX)-1
N
CX=0?
Y
AX<COUNT?
Y
N
COUNT←AX
DATA←DX
CX←(CX)-1
N
CX=0?
Y
CX←AX
AX←CX
结束

图 5.17

```
L2:     CMP     AX,COUNT
        JBE     L3
        MOV     COUNT,AX
        MOV     DATA,DX
L3:     DEC     CX
        JNZ     L0
        MOV     CX,AX
        MOV     AX,DX
        RET
MATN    ENDP
        END     MAIN
```

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。

【解答】 程序的流程图如图 5.18 所示。

程序代码如下：

```
        ·MODEL SMALL
        ·STACK 100H
        ·DATA
M       DW  X1,X2,…,Xn
MAX     DW  0
ADR     DW  ?
        ·CODE
MAIN    PROC  FAR
PUSH    DS
XOR     AX,AX
PUSH    AX
MOV     AX,@DATA
```

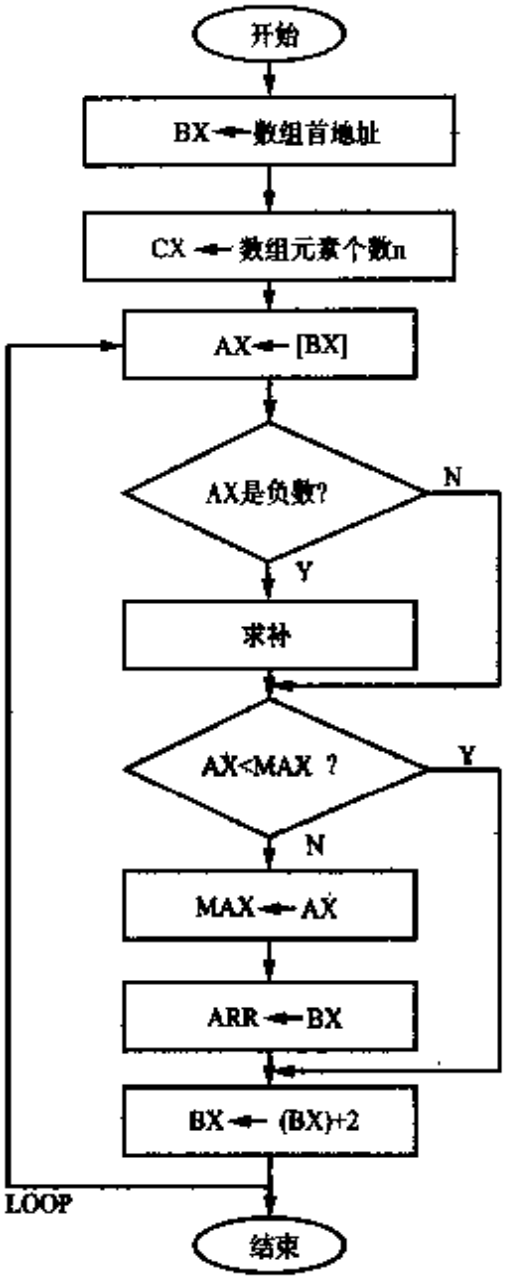


图 5.18

```
        MOV     DS,AX
        LEA     BX,M
        MOV     CX,N
L1:     MOV     AX,[BX]
        TEST    AX,8000
```

```
        JZ      L3
        NEG     AX
L3:     CMP     AX,MAX
        JB      L2
        MOV     MAX,AX
        MOV     ADR  BX
L2:     ADD     BX,2
        LOOP    L1
        RET
MAIN    ENDP
        END     MAIN
```

5.16 在首地址为 DATA 的字数组中，存放了 100H 个 16 位补码数，试编写一程序，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。

【解答】 程序的流程图如图 5.19 所示。

主要程序代码如下：

```
        MOV     CX,100H-1
        MOV     DX,0
        LEA     SI,DATA
        MOV     AX,[SI]
        CWD
L1:     ADD     SI,2
        ADD     AX,[SI]
        ADC     DX,0
        JO      RET1
        LOOP    L1
        MOV     CX,100H
```

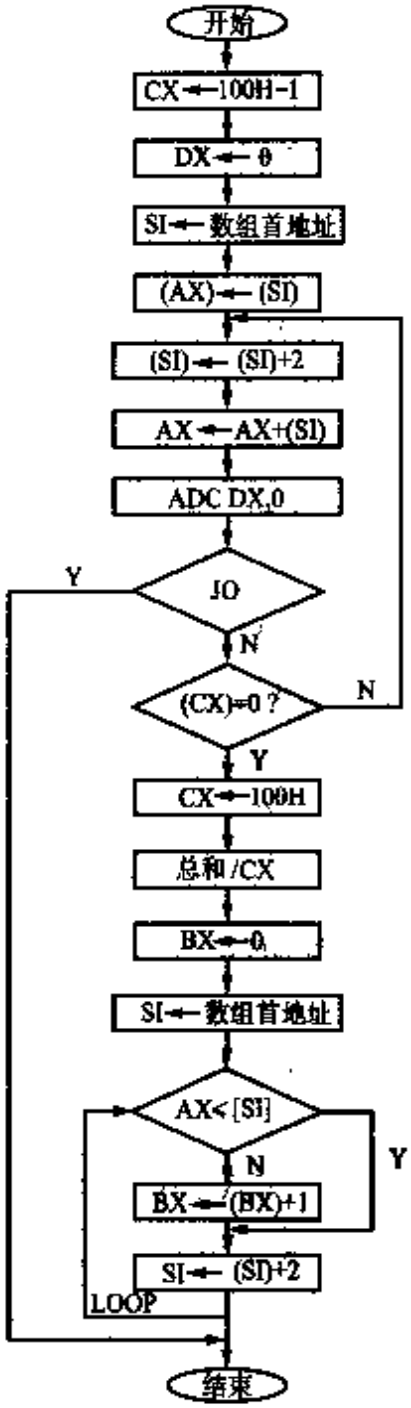


图 5.19

```
IDIV    CX
MOV     BX,0
LEA     SI,DATA
```

```
L2:     CMP     AX,[SI]
        JLE     L3
        INC     BX
L3:     ADD     SI,2
        LOOP    L2
RET1:   RET
```

5.17 试编制一个程序，把 AX 中的十六进制数转换为 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 和 32H。

【解答】 主要程序代码如下：

```
        LEA     SI,MEM
        MOV     CH,4
L1:     MOV     CL,4
        MOV     DL,0FH
        AND     DL,AL
        CMP     DL,10
        JB      L2
        ADD     DL,7
L2:     ADD     DL,30H
        MOV     [SI],DL
        ROL     AX,CL
        INC     SI
        DEC     CH
        JNZ     L1
```

5.18 把 0～100D 之间的 30 个数存入以 GRADE 为首地址的 30 个字数组中，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）。

【解答】 程序的流程图如图 5.20 所示。

程序代码如下：

```
        .MODEL SMALL
        .STACK 100H
        .DATA
GRADE   DW  30DUP(?)
RANK    DW  30DUP(?)
        .CODE
MAIN    PROC  FAR
        PUSH  DS
        XOR   AX,AX
        PUSH  AX
        MOV   AX,@DATA
        MOV   DS,AX
        MOV   ES,AX
        MOV   DI,0
        MOV   BX,30
L1:     MOV   CX,30
        MOV   SI,0
        MOV   AX,GRADE[DI]
        MOV   DX,0
L2:     CMP   GRADE[SI],AX
        JBE   L3
        INC   DX
L3:     ADD   SI,2
```

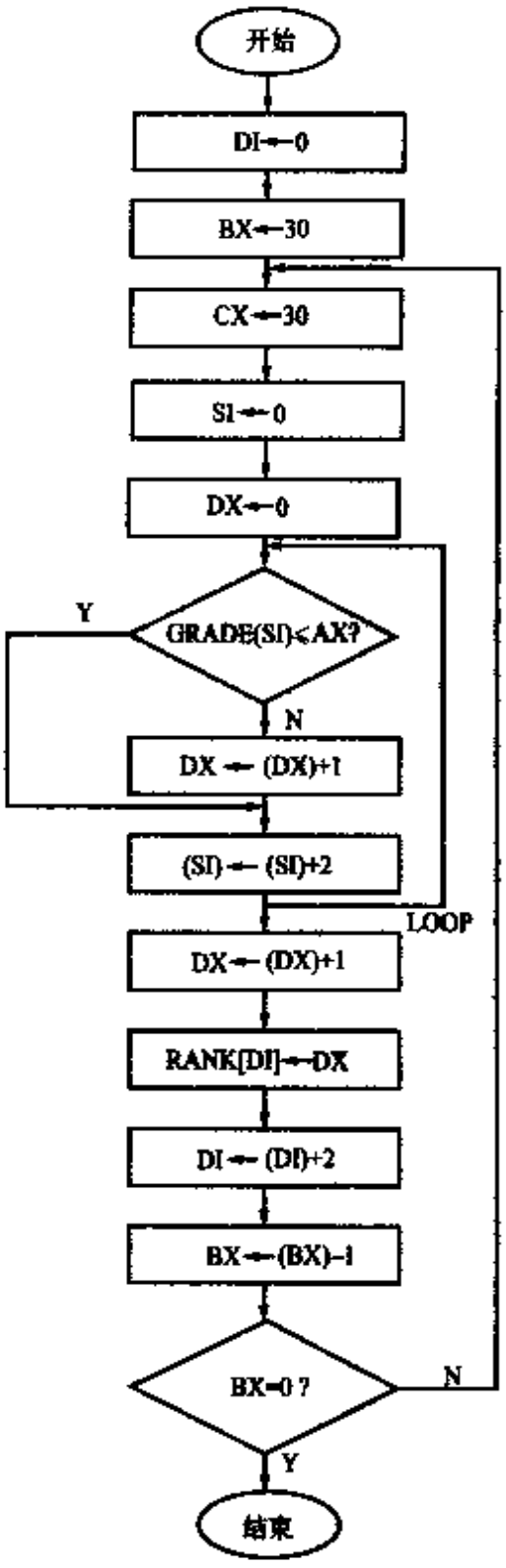

开始
DI←0
BX←30
CX←30
SI←0
DX←0
GRADE(SI)<AX?
Y
N
DX ← (DX)+1
(SI) ← (SI)+2
LOOP
DX←(DX)+1
RANK[DI]←DX
DI ← (DI)+2
BX←(BX)-1
BX=0 ?
N
Y
结束

图 5.20

```
        LOOP    L2
        INC     DX
        MOV     RANK[DI],DX
        ADD     D1,2
        DEC     BX
        JNZ     L1
        RET
MAIN    ENDP
        END     MAIN
```

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。

【解答】 程序的流程图如图 5.21 所示。

程序代码如下：

```
        .MODEL SMALL
        .STACK 100H
        .DATA
A       DW  15DUP(?)
B       DW  20DUP(?)
C       DW  15DUP(?)
        .CODE
MAIN    PROC FAR
PUSH    DS
XOR     AX,AX
PUSH    AX
MOV     AX,@DATA
MOV     DS,AX
MOV     ES,AX
```

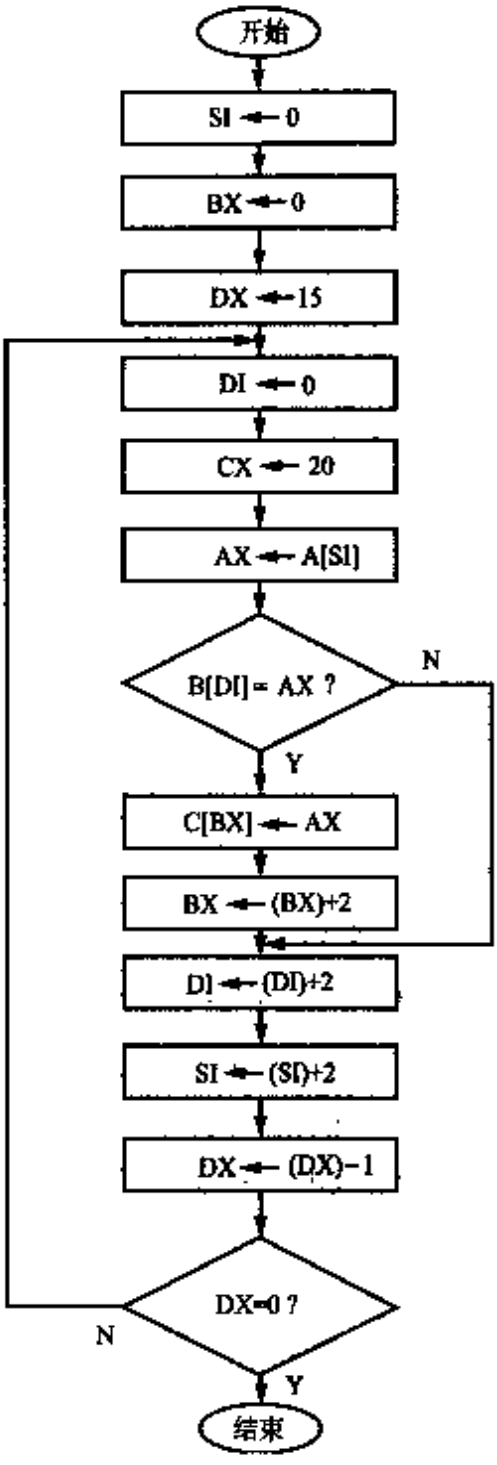


图 5.21

```
MOV     SI,0
MOV     BX,0
```

```
        MOV     DX,15
L1:     MOV     DI,0
        MOV     CX,20
        MOV     AX,A[SI]
L2:     CMP     B[DI],AX
        JNZ     L3
        MOV     C[BX],AX
        ADD     BX,2
L3:     ADD     DI,2
        LOOP    L2
        ADD     SI,2
        DEC     DX
        JNZ     L1
        RET
MAIN    ENDP
        END     MAIN
```

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,则求出三数之和并存放于 D 单元中;若其中有一个数为 0,则把其他两个单元也清零。请编写此程序。

【解答】 程序的流程图如图 5.22 所示。

程序代码如下:

```
        .MODEL SMALL
        .STACK 100H
        .DATA
A       DB      ?
B       DB      ?
C       DB      ?
D       DB      ?
```

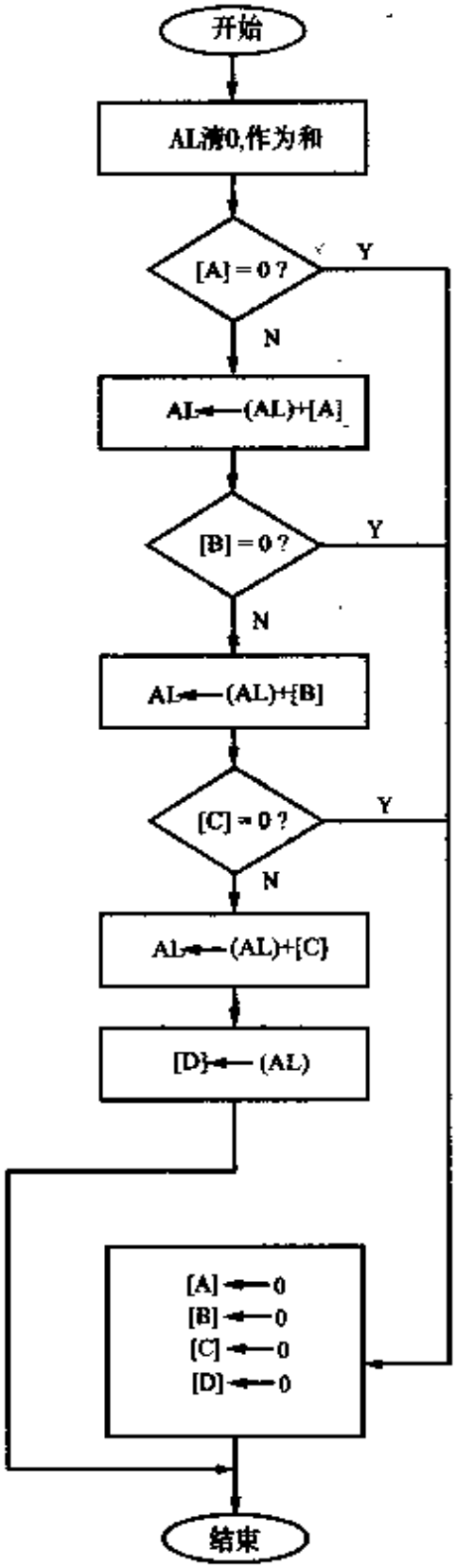


图 5.22

```
        .CODE
MAIN    PROC  FAR
        PUSH  DS
        XOR   AX    AX
        PUSH  AX
        MOV   AX,@DATA
        MOV   DS,AX
        MOV   AL,0
        CMP   [A],0
        JE    L0
        ADD   AL,[A]
        CMP   [B],0
        JZ    L0
        ADD   AL,[B]
        CMP   [C],0
        JZ    L0
        ADD   AL,[C]
        MOV   [D],AL
        RET
L0:     MOV   [A],0
        MOV   [B],0
        MOV   [C],0
        MOV   [D],0
        RET
MAIN    ENDP
        END   MAIN
```

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)如果三个数有两个相等则显示 1;

(3)如果三个数都相等则显示 2。

【解答】 程序的流程图如图 5.23 所示。

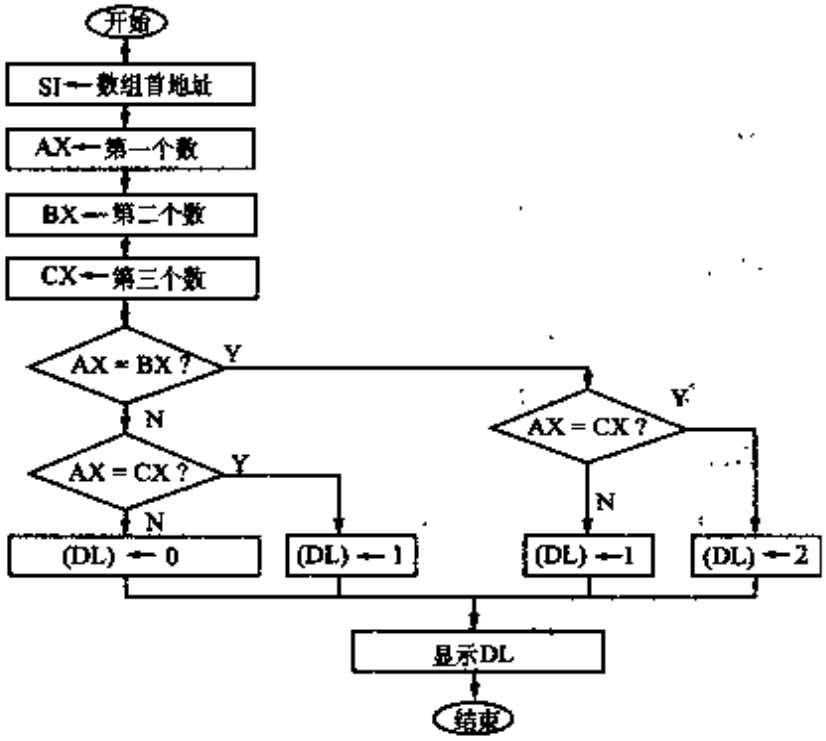


图 5.23

程序代码如下:

```
        .MODEL SMALL
        .STACK 100H
        .DATA
ARRAY   DW  3DUP(?)
        .CODE
MAIN    PROC  FAR
PUSH    DS
MOV     AX,@DATA
MOV     DS,AX
```

```
        LEA     SI,ARRAY
        MOV     AX,[SI]
        MOV     BX,[SI+2]
        MOV     CX,[SI+4]
        CMP     AX,BX
        JZ      L2
        CMP     AX,CX
        JZ      L1
        MOV     DL,0
        JMP     L4
L1:     MOV     DL,1
        JMP     L4
L2:     CMP     AX,CX
        JZ      L3
        MOV     DL,1
        JMP     L4
L3:     MOV     DL,2
        L4:ADD  DL,3
        MOV     AH,2
        INT     21H
        RET
MAIN    ENDP
        END     MAIN
```

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

【解答】 程序的流程图如图 5.24 所示。

程序代码如下:

```
        .MODEL  SMALL
```

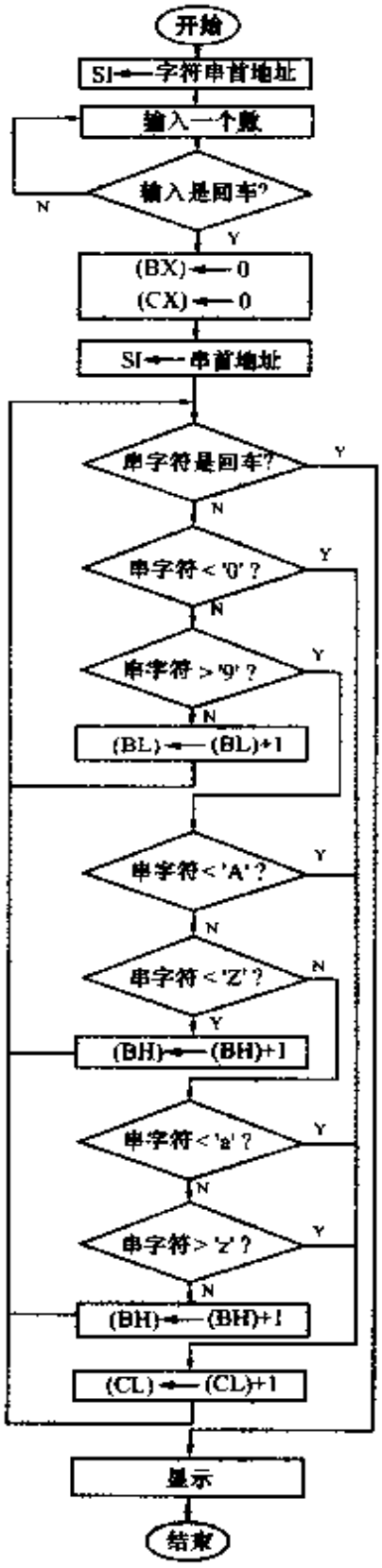

开始
SI←字符串首地址
输入一个数
输入是回车?
N
Y
(BX)←0
(CX)←0
SI←串首地址
串字符是回车?
Y
N
串字符<'0'?
Y
N
串字符>'9'?
Y
N
(BL)←(BL)+1
串字符<'A'?
Y
N
串字符<'Z'?
N
Y
(BH)←(BH)+1
串字符<'a'?
Y
N
串字符>'z'?
Y
N
(BH)←(BH)+1
(CL)←(CL)+1
显示
结束

图 5.24

```
        .STACK    100H
        .DATA
        .COUNT    DB  3DUP(?)
        .STR      DB  100DUP(?)
        .CODE
        MAIN      PROC  FAR
        PUSH      DS
        XOR       AX,AX
        PUSH      AX
        MOV       AX,@DATA
        MOV       DS,AX
        LEA       SI,STK
L2:     MOV       AH,1
        INT       21H
        CMP       AL,0DH
        JZ        L1
        JMP       L2
L1:     XOR       BX,BX
        XOR       CX,CX
        LEA       SI,STK
        DEC       SI
L4:     INC       SI
        MOV       AL,[SI]
        CMP       AL,0DH
        JE        L6
        CMP       AL,'0'
        JL        OTHER
        CMP       AL,'9'
```

```
        JG      L3
        INC     BL
        JMP     L4
L3:     CMP     AL,'A'
        JL      OTHER
        CMP     AL,'z'
        JG      L5
        INC     BH
        JMP     L4
L5:     CMP     AL,'a'
        JL      OTHER
        CMP     AL,'z'
        JG      OTHER
        INC     BH
        JMP     L4
OTHER:  INC     CL
        JMP     L4
L6:     LEA     DI,COUNT
        MOV     [DI],BL
        MOV     [DI+1],BH
        MOV     [DI+2],CL
        MOV     CH,3
L7:     MOV     DL,[DI]
        CMP     DL,'9'
        JL      L8
        ADD     DL,7
L8:     ADD     DL,30H
        MOV     AH,02H
```

```
        INT     DI
        OEC     CH
        JNZ     L7
        RET
MAIN    ENDP
        END     MAIN
```

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)若两个数均为奇数，则将两数均加 1 后存回原变量；

(3)若两个数均为偶数，则两个变量均不改变。

【解答】 程序的流程图如图 5.25 所示。

程序代码如下：

```
        .MODAL  SMALL
        .STACK  100H
        .DATA
A   DW      ?
B   DW      ?
        .CODE
MAIN    PROC    FAR
        PUSH    DS
        XOR     AX,AX
        PUSH    AX
        MOV     AX,@DATA
        MOV     DS,AX
        MOV     AX,[A]
        MOV     BX,[B]
```

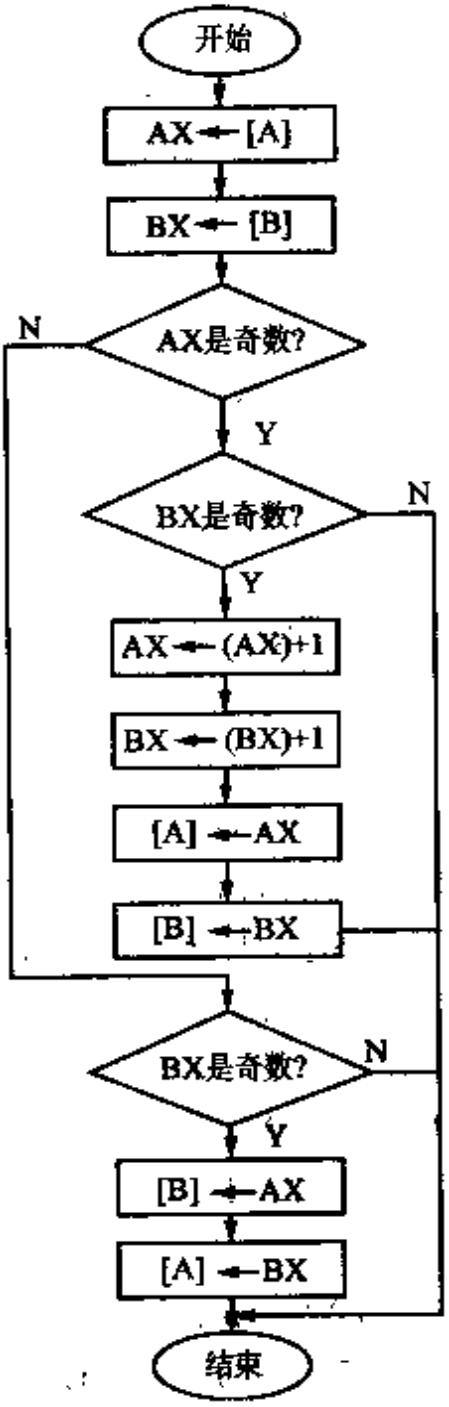


图 5.25

```
TEST    AX,01H
JNZ     L1
TEST    BX,01H
```

```
        JZ      L2
        MOV     [B],AX
        MOV     [A],BX
L2:     JMP     RET1
L1:     TEST    BX,01H
        JZ      L3
        INC     AX
        INC     BX
        MOV     [A],AX
        MOV     [B],BX
L3:     JMP     RET1
RET1:   RET
MAIN    ENDP
        END     MAIN
```

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 5 个歌曲程序中的某一个。

【解答】 程序代码如下：

```
        .MODEL  SMALL
        .STACK  100H
        .DATA
SINGLIST DW  ARR1,ARR2,ARR3,ARR4,ARR5
SONGNAME1 DB 'THE  FIRST  $'
SONGNAME2 DB 'THE  SECOND $'
SONGNAME3 DB 'THE  THIRD  $'
SONGNAME4 DB 'THE  FORTH  $'
SONGNAME5 DB 'THE  FIFTH  $'
        .CODE
```

```
        MAIN    PROC  FAR
        PUSH    DS
        XOR     AX,AX
        PUSH    AX
        MOV     AX,@DATA
        MOV     DS,AX
        MOV     AH,07H
        INT     21H
        MOV     AH,0
        SUB     AL,30H
        SHL     AL,1
        MOV     DI,AX
        JMP     SINGLIST[DI]
ADDR1:  LEA     DX,SONGNAME1
        MOV     AH,09H
        INT     21H
        RET
ADDR2:  LEA     DX,SONGNAME2
        MOV     AH,09H
        INT     21H
        RET
ADDR3:  LEA     DX,SONGNAME3
        MOV     AH,09H
        INT     21H
        RET
ADDR4:  LEA     DX,SONGNAME4
        MOV     AH,09H
        INT     21H
```

```
        RET
ADDR5:  LEA     DX,SONGNAME5
        MOV     AH,09H
        2NT     21H
        RET
MAIN    ENDP
        END     MAIN
```

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。

【解答】：

(1)
```
        .MODEL  SMALL
        .286
        .STACK  100H
        .DATA
A       DW      0
B       DW      0
C       DW      0
DATAX   DW      ?
DATAY   DW      ?,?
        .CODE
MAIN    PROC    FAR
        MOV     AX,@DATA
        MOV     DS,AX
        MOV     AX,DATAX
        MUL     DATAY
        MOV     A,AX
        MOV     B,DX
```

```
        MOV     AX,DATAX
        MUL     DATAY
        ADD     B,AX
        ADC     C,DX
        RET
MAIN    ENDP
        END     MAIN
```

(2)

```
        .MODEL  SMALL
        .386
        .STACK  100H
        .DATA
DATAX   DW  ?
DATAY   DW  ?
        .CODE
MAIN    PROC  FAR
        MOV     AX,@DATA
        MOV     DS,AX
        MOV     EAX,DATAY
        MUL     DATAX
        RET
MAIN    ENDP
        END     MAIN
```

由以上两个程序段可知：

386 机器将字长从 16 位扩展到 32 位，无论从空间和时间方面都有利于效率的提高。

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 movsb 指令来实现。

试编写这一程序。

【解答】 程序代码如下：

```
        .MODEL      SMALL
        .STACK      100H
        .DATA
MESS1   DB   35DUP(?)
MESS2   DB   35DUP(?)
        .CODE
        PUSH        DS
        XOR         AX,AX
        PUSH        AX
        MOV         AX,@DATA
        MOV         DS,AX
        MOV         ES,AX
        MOV         CX,35
        CLD
        LEA         SI,MESS1
        LEA         DI,MESS2
        MOVSB
        RET
MAIN    ENDP
        END         MAIN
```

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。

【解答】 程序代码如下：

```
        .MODEL      SMALL
        .386
        .STACK      100H
```

```
        .DATA
        DATA1       DQ    ?
        DATA2       DQ    ?
        DATA3       DQ    ?
        .CODE
MAIN    FROC  FAR
        PUSH        DX
        XOR         AX,AX
        PUSH        AX
        MOV         AX,@DATA
        MOV         DS,AX
        MOV         SI,0
        MOV         DI,0
        MOV         BX,0
        MOV         CX,2
L0:     MOV         EAX,DATA1[SI*4]
        ADC         EAX,DATA2[DI*4]
        MOV         DATA3[BX*4],EAX
        INC         DI
        INC         SI
        INC         BX
        LOOP        L0
        RET
MAIN    ENDP
        END         MAIN
```

5.4 强化训练

1. 编写程序段，完成所要求的功能：

(1)将标志寄存器中的高 8 位和低 8 位互换。

(2)按八进制数显示十六进制数 2008H 的内容。

(3)将一首地址为 ARRAY 的字数组(元素个数为 100)每个数据都减 10。

2. 变量 X 的符号函数可用下式表示：

$$Y=\begin{cases}1, & \text{当 } X>0\\ 0, & \text{当 } X=0\\ -1, & \text{当 } X<0\end{cases} \qquad (-128\leqslant X\leqslant +127)$$

设任意给定的 X 存放在 DATAX 单元，计算出函数 Y 值要求存放在 DATAY 单元中。

3. 编制程序，在存储区中构造一个九九乘法表。

答 案

```
1.(1) PUSHF
      POP     AX
      XCHG    AH,AL
      PUSH    AX
      POPF
(2)   MOV     AX,2008H    ;装入测试数据
      MOV     BX,8
      PUSH    BX
L1:   MOV     BX,0
      DIV     BX
      PUSH    DX
```

```
        CMP     AX,0
        JNZ     L1
L2:     POP     DX          ;取余数
        CMP     DX,BX
        JZ      L3          ;如果余数为 8
        ADD     DL,30H      ;转换为 ASCII 码
        MOV     AH,6
        INT     21H         ;显示
        JMP     L2
L3:     RET
(3)     MOV     AX,DATA
        MOV     DS,AX
        LEA     SI,ARRAY
        MOV     CX,100
L0:     SUB     WORD PTR[SI],10
        INC     SI
        INC     SI
        L00P    L0
        RET
```

2. 程序代码如下：

```
DSEG    SEGMENT
DATAX   DB    03H
DATAY   DB    ?
DSEG    ENDS
CSEG    SEGMENT
ASSUME  CS:CSEG,DS:DSEG
MAIN    PROC        FAR
MOV     AX,DSEG
```

```
        MOV     DS,AX
        MOV     AL,DATAX
        CMP     AL,0
        JGE     L1              ;X≥0 时转移
        MOV     AL,0FFH         ;X<0,0FFH=－1→
                                AL
        JMP     L2              ;转向出口
L1:     JE      L2              ;X=0 转向出口(此时
                                 AL=0)
        MOV     AL,1;X>0,1→AL
L2:     MOV     DATAY ,AL       ;把结果送到 DATAY 单
                                元中
        RET
        START   ENDP
        CSEG    ENDS
        END     MAIN
```

3. 程序代码如下：

```
        DSEG    SEGMENT
        TABLE   DB  9*9DUP(0)
        DSEG    ENDS
        STACK   SEGMENT  PARA  STACK
                DW  20H  DUP(0)
        STACK   ENDS
        CSEG    SEGMENT
                ASSUME  CS:CSEG,DS:DSEG
        BEING:  MOV             AX,DSEG
        MOV  DS,AX
        MOV  DI,OFFSET  TABLE,取表首地址
```

```
        XOR     BH,BH           ;置行号初值
L1:     PUSH    CX              ;暂存处循环次数
        INC     BH              ;行号记数
        XOR     BL,BL           ;置列号初值
        MOV     CX,9            ;置内循环次数初值
L2:     INC     BL              ;列号记数
        MOV     AL,BH           ;行号传入 AL
        MUL     BL              ;行号 * 列号传入 AL
        MOV     [DI],AL         ;存九九表数据
        INC     DI              ;修改指针
        LOOP    L2              ;控制内循环
        POP     CX              ;恢复外循环次数
        LOOP    L1              ;控制外循环
        RET
        CSEG    ENDS
        END     BEING
```

第6章 子程序结构

6.1 内容提要

6.1.1 子程序的结构

在程序设计的过程中，某些具有特定功能的程序段可能在程序中多次用到，或被其他程序调用。为了节省存储空间，减少重复编制程序的麻烦，往往将这样的程序段独立出来，加上一些语句，形成一个可以被直接调用的独立功能段。这样的功能程序称为子程序。

1. 子程序的定义是由过程定义为指令来实现的，一个过程是一段程序，以 PROC 伪指令语句开始，以 ENDP 伪指令语句结束。格式如下：

```
过程名  PROC  [NEAR 或 FAR]
          ⋮
          ⋮   过程体语句
        RET
过程名  ENDP
```

过程必须定义在代码段内，即使有多个过程，也都要在代码段定义它们。过程名是一种标识符，它是子程序的入口的符号地址，具有地址标号的属性。类型属性是指，它可以为 NEAR 或 FAR。NEAR 属性为段内调用；FAR 属性为段间调用。用户对子程序属

性的确定原则是：调用程序和子程序在同一个代码段中则使用NEAR属性；调用程序和子程序不在同一代码段中则使用FAR属性。

2. 子程序的返回

子程序(过程)调用指令CALL的格式为：

CALL　OPRD

子程序(过程)返回指令RET的格式为：

RET　[N]

3. 现场保护与恢复

在标准子程序中，它所使用的工作寄存器一般要存入堆栈保存，在返回调用程序之前，再恢复它们的内容。

6.1.2 子程序的参数传递

由于某些变量的赋值不同，子程序就可以对不同的数据进行相同的处理。所以，在调用子程序以前主程序应将入口参数传递给子程序，而子程序在返回主程序时也应将结果返回给主程序。

子程序参数传递的方法通常有以下几种：

(1)寄存器传送参数；

(2)过程和调用程序在同一源文件中，则过程直接访问模块中的变量；

(3)过程和调用程序不在同一源文件中，用伪操作及其参数解决多模块间的参数传递；

(4)地址表传送参数地址；

(5)堆栈传送参数或参数地址。

6.1.3 子程序的嵌套和递归调用

(1)子程序嵌套

子程序的嵌套调用就是一个子程序可以作为调用程序调用另

一个子程序。

要注意以下几点：

①正确使用 CALL 和 RET；

②寄存器的保存与恢复；

③堆栈的平衡使用。

(2)子程序递归

在子程序嵌套的情况下，如果一个子程序调用的子程序就是它本身，这就称为子程序的递归调用。

要注意以下几点：

①保证每次调用都不破坏以前调用时所用的参数；

②堆栈的正确使用；

③基数的设置；

④条件转移指令实现递归退出。

6.1.4 PUBLIC 和 EXTRN 指令

在进行程序设计时，如果程序比较大，可以将程序分成不同的独立模块，由不同的人来共同完成。与子程序设计不同的是，多模块程序设计的多个模块之间是有联系的，并且每个模块可以独立汇编成目标(.OBJ)文件，然后由连接程序将这些目标文件连接成一个可执行文件。

在多模块程序设计中，一个模块通常要引用到其他模块的变量或子程序，因此需要有指示符来指明这些变量或子程序在其他模块是可用的。PUBLIC 和 EXTRN 伪指令就是用来说明在多模块程序设计中的全局符号和外部符号的。

(1)PUBLIC 伪指令

格式：PUBLIC 符号[,符号]

功能：说明其后的符号是公共(全局)符号。公共符号能被其他模块引用。PUBLIC 伪指令可声明多个这样的符号，符号间用

逗号隔开。

(2)EXTRN 伪指令

格式:EXTRN 符号:类型[,符号:类型]

功能:说明在本模块中需要引用的由其他模块定义的符号,即外部符号。其中的类型可以是 NEAR、FAR、BYTE、WORD 或 DWORD 等符号类型。一个 EXTRN 语句可以声明多个外部符号,它们之间用逗号分隔。

6.1.5 考 点

1. 子程序的定义和参数传递的方法;
2. 嵌套和递归程序的分析和设计;
3. 用多模块法编制程序。

6.2 例题精解

例 1 在字节型变量 BCDBUF 中有一个组合 BCD 码,试将其转换为二进制数后存入 BINBUF 单元中。

【分析与解答】 将组合 BCD 码分离出相应于十进制数的十位和个位,再进行十位数乘 10 加上个位数的运算即可得到对应的二进制码。

```
DSEG    SEGMENT
BCDBUF  DB    65H          ;定义组合的 BCD 码
BINBUF  DB    ?            ;放转换结果的单元
DSEG    ENDS
CSEG    SEGMENT
        ASSUME CS:CSEG,DS:DSEG
START:  MOV   AX,DSEG
        MOV   DS,AX
```

```
        MOV   AL,BCDBUF   ;将要传递的参数放在寄
                          存器 AL 中
        CALL  TRAN
        MOV   BINBUF,AL   ;返回结果也在 AL 中
        RET
TRAN    PROC              ;转换子程序
        PUSHF
        PUSH  BX
        PUSH  CX
        MOV   AH,AL
        AND   AH,0FH      ;分离出个位数
        MOV   BL,AH
        AND   AL,0F0H     ;分离出十位数
        MOV   CL,04H
        ROR   AL,CL       ;将十位数移至低 4 位
        MOV   BH,0AH
        MUL   BH          ;十位数乘 10
        ADD   AL,BL       ;乘积与个位数相加,结
                          果存放在 AL 中
        POP   CX
        POP   BX
        POPF
        RET
TRAN    ENDP
CSEG    ENDS
        END   BEGIN
```

例 2 编写一个子程序,分类统计出一个字符串中数字字符、字母和其他字符的个数。该字符串的首地址用 DS：DX 来指定

(以$为字符串结束),各类字符个数分别存放BX,CX和DI中。

【分析与解答】 题目要求分类统计出字符串中数字字符、字母和其他字符的个数,用DS:DX指向被统计的字符串,当字符在'0'~'9'范围时,数字字符个数BX加1,在'A'~'Z'或'a'~'z'范围时,字母个数CX加1,否则DI加1,最后BX、CX、DI分别保存了三种字符的个数。

```
DSEG    SEGMENT
STR     DB    '2008 BEIJING;,! $'
DSEG    ENDS
CSEG    SEGMENT
        ASSUME CS:CSEG,DS:DSEG
MAIN    PROC  FAR
        MOV   AX,DSEG
        MOV   DS,AX
        MOV   DX,OFFSET STR
        CALL  COUNT
        RET
MAIN    ENDP
COUNT   PROC
        PUSH  AX
        PUSH  SI
        XOR   BX,BX
        XOR   CX,CX
        XOR   DI,DI       ;使各类字符计数清零
        MOV   SI,DX
AGAIN:  MOV   AL,[SI]
        INC   SI
        CMP   AL,'$'
```

```
        JE      OVER
        CMP     AL,'0'
        JL      OTHER
        CMP     AL,'9'
        JG      UPCHR
        INC     BX          ;数字字符个数加 1
        JMP     AGAIN
UPCHR:  CMP     AL,'A'
        JL      OTHER
        CMP     AL,'Z'
        JG      DNCHR
        INC     CX          ;字母个数加 1
        JMP     AGAIN
DNCHR:  CMP     AL,'a'
        JL      OTHER
        CMP     AL,'z'
        JG      OTHER
        INC     CX          ;字母个数加 1
        JMP     AGAIN
OTHER:  INC     DI          ;其他字符个数加 1
        JMP     AGAIN
OVER:   POP     SI
        POP     AX
        RET
COUNT   ENDP
CSEG    ENDS
        END     MAIN
```

6.3 课后习题解答

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
```

【解答】 "CRAY PROC"改为 CRAY PROC FAR/NEAR；
"ENDP CRAY"改为 CRAY ENDP。

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。

【解答】 堆栈及 SP 内容的变化过程如图 6.1 所示。

6.3 分析下面程序，画出堆栈最满时各单元的地址及内容。程序如下：

```
;* * * * * * * * * * * * * * * * * * * * * * * * * * *
s_seg   segment  at 1000h        ;define stack segment
        dw       200 dup(?)
  tos   label    word
s_seg   ends
;* * * * * * * * * * * * * * * * * * * * * * * * * * *
c_seg   segment                  ; define code segment
        assume   cs:c_seg,ss:s_seg
        mov      ax,s_seg
        mov      ss,ax
```

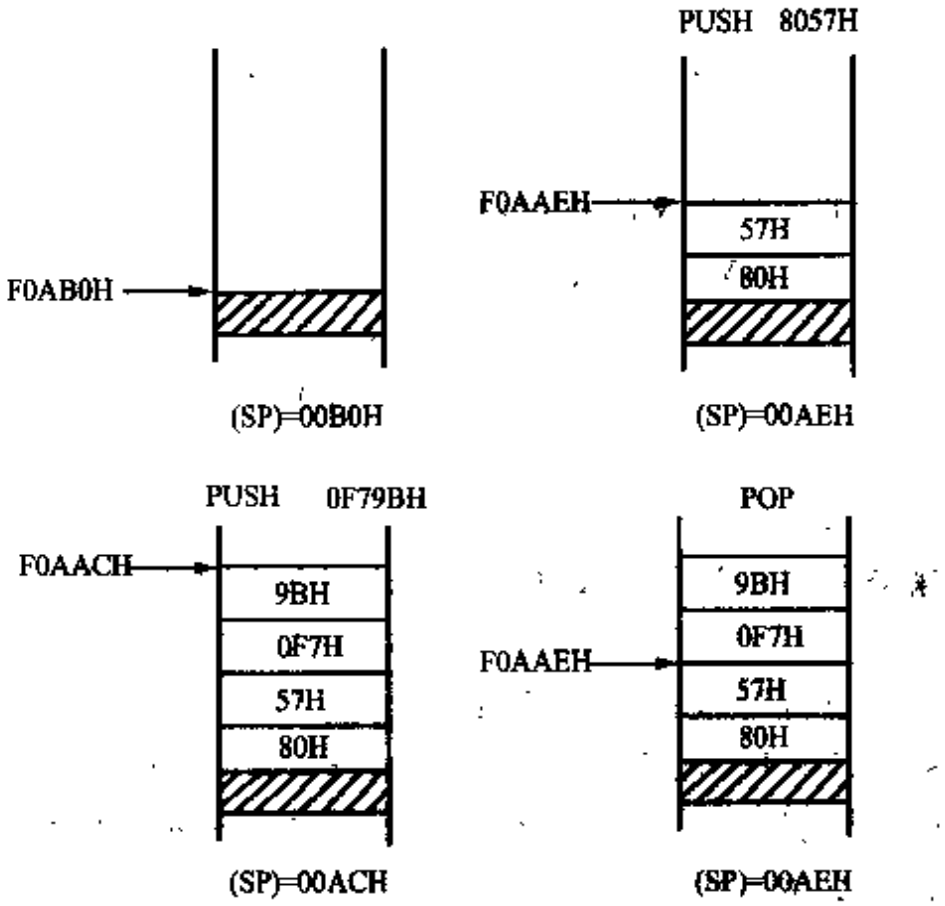


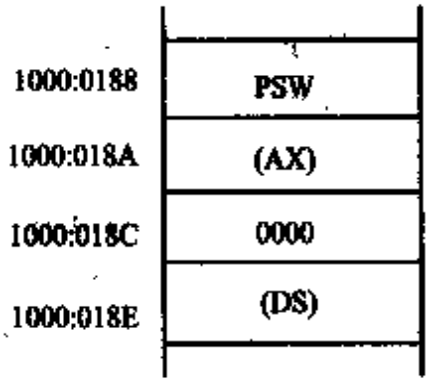
图 6.1

```
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                    ;end of code segment
;*********************************
        end     c_seg           ;end of assembly
```

【解答】 堆栈最满时各单元的地址及内容如图 6.2 所示。



(SP) = 0188H

图 6.2

6.4 分析下面程序的功能，写出堆栈最满时各单元的地址及内容。

程序如下：

```
;*********************************
stack   segment at  500h
        dw          128 dup(?)
  tos   lahel       word
stack   ends
;*********************************
code    segment                 ;define code segment
;*********************************
main    proc    far             ;main part of program
        assume cs:code,ss:stack
```

```
start:                          ;starting execution address
        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                     ;return to DOS
main    endp                    ;end of main part of program
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * *

htoa    proc    near            ;define subprocedure htoa
        cmp     ax,15
        jle     bl
        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
bl:     add     al,30h
```

```
        cmp     al,3ah
        jl      printir
        add     al,7h
printit: mov    dl,al
        mov     ah,2
        int     21h
        ret
htoa    endp                    ;end of subprocedure
;*********************************
code    end                     ;end of code segment
;*********************************
        end     start           ;end of assembly
```

【解答】 堆栈最满时各单元的地址及内容如图 6.3 所示。

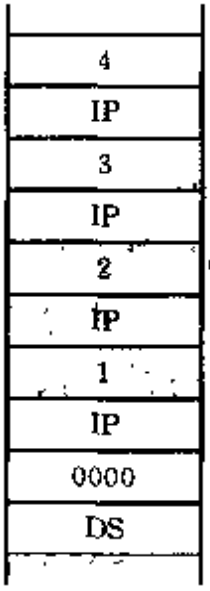


图 6.3

功能为逐个显示 4、3、2、1。

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

【解答】 子程序代码如下：

```
SKIPLINES    PROC
             PUSH   CX
             PUSH   AX
             MOV    CX,AX
             INC    CX
L1:          MOV    DL,0AH
             MOV    AH,02H
             INT    21H
             MOV    DL,0DH
             MOV    AH,0IH
             INT    21H
             LOOP   L1
             POP    AX
             POP    CX
             RET
SKIPLLNES    ENDP
```

6.6 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分的人数并分别存放到S6,S7,S8,S9和S10单元中。

【解答】 程序代码如下：

```
             .MODEL     SMALL
             .STACK     100H
             .DATA
RECORD       DW   76,69,84,90
             DW   73,88,99,63
             DW   100,80
S6           DW   0
```

```
        S7          DW  0
        S8          DW  0
        S9          DW  0
        S10         DW  0
        .CODE
MAIN    FROC        FAR
        FUSH        DS
        XOR         AX,AX
        PUSH        AX
        MOV         AX,@DATA
        MOV         DS,AX
        MOV         CX,10
        CALL        COUNT
        RET
MAIN    ENDP
COUNT   PROC        NEAR
        MOV         SI,0
L1:     MOV         AX,RECORD[SI]
        MOV         BL,10
        DIV         BL
        MOV         BL,AL
        CWB
        SUB         BX,6
        SAL         BX,1
        INC         WORD PTR  S6[BX]
        ADD         SI,2
        LOOP        L1
        RET
```

```
COUNT ENDP
      END       MAIN
```

6.7 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，并记录该字符的出现次数。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，假设出现次数小于等于 15，可以用十六进制形式把它显示出来）

【解答】 程序代码如下：

```
      .MODEL    SMALL
      .STACK    100H
      .DATA
TABLE DB        100 DUP(?)
N     DB        ?
CHAR  DB        ?
      .CODE
MAIN  PROC      FAR
      PUSH      DS
      XOR       AX,AX
      PUSH      AX
      MOV       AX,@DATA
      MOV       DS,AX
      LEA       SI,TABLE
L0:   MOV       AH,07H
      INT       21H
      MOV       [SI],AL
      CMP       AL,0DH
```

```
        JZ      L1
        JMP     L0
L1:     LEA     SI,TABLE
        MOV     CX,N
L2:     MOV     DL,[SI]
        MOV     AH,02H
        INT     21H
        MOV     DL,20H
        MOV     AH,02H
        MOV     CHAR,DL
        CALL    FIND
        CMP     DL,9
        JNA     L3
        INC     DL,07H
L3:     INC     DL,30H
        MOV     AH,02H
        INT     21H
        INC     SI
        LOOP    L2
        RET
MAIN    ENDP
FIND    PROC    NEAR
        PUSH    CX
        PUSH    SI
        PUSH    AX
        MOV     DL,0
        MOV     CX,N
        MOV     SI,0
```

```
        MOV     AX,CHAR
L0:     CMP     AX,TABLE[SI]
        JNZ     L1
        INC     DL
L1:     INC     SI
        LOOP    LD
        POP     A
        POP     S
        POP     CX
        RET
FIND    ENDP
        END     MAIN
```

6.8 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：

(1)显示提示符“INPUT NAME:”；

(2)调用子程序 INPUT_NAME，输入姓名；

(3)显示提示符“INPUT A TELEPHONE NUMBER:”；

(4)调用子程序 INPHONE，输入电话号码；

(5)调用子程序 PRINTLINE，显示姓名及电话号码。

子程序 INPUT_NAME：

(1)调用键盘输入子程序 GETCHAR，把输入的姓名存放在 INBUF 缓冲区中；

(2)把 INBUF 中的姓名移入输出行 OUTNAME。

子程序 INPHONE：

(1)调用键盘输入子程序 GETCHAR，把输入的 8 位电话号码存放在 INBUP 缓冲区中；

(2)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.
        ×××         ×××
```

【解答】 程序代码如下：

```
        .MODEL      SMALL
        .STACK      100H
        .DATA
        STR1        DB  'INPUT_NAME:$'
        STR2        DB  'INPUT  A  TELEPHONE
                            NUMBER:$'
        STR3        DB  'NAME  ',",",",",",",'TEL,$'
        INBUF       DB  10DUP(?)
        OUTNAME     DB  10DUP(?)
        OUTNAME     DB  10DUP(?)
        OUTPHONE    DB  8DUP(?)
        .CODE
MAIN    PROC        FAR
        PUSH        DS
        XOR         AX,AX
        PUSH        AX
        MOV         AX,@DATA
        MOV         DS,AX
        MOV         ES,AX
        LEA         DX,STR1
        MOV         AH,09
        INT         21H
        CALL        INPUT_NAME
```

```
        LEA     DX,STR2
        MOV     AH,09
        INT     21H
        CALL    INPHONE
        CALL    PRINTLINE
MAIN    ENDP
INPUT_NAME      PROC  NEAR
        PUSH    CX
        MOV     CX,0
        MOV     SI,0
L1:     CALL    GETCHAR
        INC     CX
        CMP     AL,0DH
        JZ      L2
        MOV     INBUF[SI],AL
        INC     SI
        JMP     L1
L2:     LEA     SI,INBUF
        LEA     DI,OUTPHONE
        CLD
        MOVSB
        RET
INPHONE         ENDP
PRINTLINE       PROC  NEAR
        PUSH    DX
        LEA     DX,STR3
        MOV     AH,09
        INT     21H
```

```
        MOV     DL,0AH
        MOV     AH,02
        INT     21H
        MOV     DL,0DH
        MOV     AH,02
        INT     21H
        LEA     DX,OUTNAME
        MOV     AH,09
        INT     21H
        MOV     DL,20H
        MOV     AH  02
        INT     21H
        LEA     DX,OUTPHONE
        MOV     AH,09
        INT     21H
        RET
PRINT_LINE      ENDP
GETCHAR         PROC  NEAR
        MOV     AH,7
        INT     21H
        RET
GETCHAR         ENDP
```

6.9 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：把整数字变量 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的子程序。

【解答】 程序代码如下：

```
          .MODEL    SMALL
          .STACK    100H
          .DATA
            VAL1    DW    ?
            VAL2    DW    ?
          .CODE
BANDO     PROC      FAR
          PUSH      DS
          XOR       AX,AX
          PUSH      AX
          MOV       AX,@DATA
          MOV       DS,AX
          PUSH      VAL1
          CALL      PAIRS
          RET
BANDO     ENDP
PAIRS     PROC      FAR
          PUSH      BP
          MOV       BP,BP
          PUSH      BX
          MOV       BX,[BP+4]
          CALL      OUTBIN
          MOV       CX,8
SPACE     MOV       DL,' '
          MOV       AH,2
```

```
        INT     21H
        LOOP    SPACE
        CALL    OUTOCT
        CALL    CRLF
        POP     BX
        POP     BP
        RET
PAIRS   ENDP
OUTBIN  PROC    NEAR
        PUSH    BX
        PUSH    DX
        MOV     CX,16
L1:     ROL     BX,1
        MOV     DX,BX
        AND     DX,1
        OR      DX,30H
        MOV     AH,2
        INT     21H
        LOOP    L1
        POP     DX
        POP     BX
        RET
OUTBIN  ENDP
OUTOCT  PROC    NEAR
        PUSH    BX
        PUSH    DX
        PUSH    CX
        MOV     CX,5
```

```
        ROL     BX,1
        MOV     DX,BX
        AND     DX,01H
        OR      DX,30H
        MOV     AH,2
        INT     21H
L2:     MOV     CL,3
        ROL     BX,CL
        MOV     DX,BX
        AND     DX,7
        OR      DX,3OH
        MOV     AH,2
        INT     21H
        POP     CX
        LOOP    L2
        RET
OUTOCT  ENDP
CRLF    PROC    NEAR
        MOV     DL,0AH
        MOV     AH,2
        INT     21H
        MOV     DL,0DH
        MOV     AH,2
        INT     21H
        RET
CRLF    ENDP
```

6.10 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?

【解答】 (1)EXTRN SUBPRO

(2)在程序中加上"PUBLIC SUBPRO"语句。

6.11 假定程序 MAINPRO 和 SUBPRO 不在同一模块中,MAINPRO 中定义字节变量 QTY、字变量 VALUE、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中定义的?

【解答】 (1)PUBLIC QTY,VALUE,PRICE

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

6.12 假设:

(1)在模块 1 中定义了双字变量 VAR1,首地址为 VAR2 的字节数组和 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。

【解答】

```
(1)EXTRN   VAR3:WORD
   PUBLIC  VAR1,VAR2,LAB1
(2)EXTRN   VAR1:DWORD,VAR2:BYTE,
           LAB1:NEAR,LAB3:FAR
```

```
        PUBLIC  VAR3,LAB2
(3)EXTRN    VAR1:DWORD,VAR2:BYTE,
            LAB1:NEAR,LAB
        PUBLIC  LAB3
```

6.13 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 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。

【解答】 程序代码如下：

```
        .MODEL      SMALL
        .STACK      100H
        .DATA
        QTY         DB      ?
        PRICE       DB      ?
        .CODE
MAIN    PROC        FAR
        PUSH        DS
        XOR         AX,AX
        PUSH        AX
        MOV         AX,@DATA
        MOV         DS,AX
        MOV         AX,PRICE
        MOV         BX,QTY
        CALL        SUBMUL
        RET
MAIN    ENDP
```

```
SUBMUL  PROC        MEAR
        MUL         BX
        RET
SUBMUL  ENDP
        END         MAIN
```

6.14 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？

【解答】

```
(1)D_SEG    SEGMENT
   X        DW      20DUP(?)
   Y        DW      20DUP(?)
   D_SEG    ENDS
   E_SEG    SEGMENT
   R        DW      20DUP(?)
   E_SEG    SEGMENT
   C_SEG    SEGMENT
   MAIN     PROC  FAR
            ASSUME  CS:C_SEG,DS:D_SEG,ES:E_SEG
```

```
START:   PUSH    DS
         XOR     AX,AX
         PUSH    AX
         MOV     AX,D_SEG
         MOV     DS,AX
         MOV     AX,ESEG
         MOV     ES,AX
         CALL    COMPUTE
         RET
MAIN     ENDP
COMPUTE  PROC    NEAR
         PUSH    AX
         MOV     AX,[X]
         ADD     AX,[Y]
         SUB     AX,3
         MOV     ES:[R],AX
         POP     AX
         RET
COMPUTE  ENDP
         END     MAIN
```

(2)段间调用，子程序属性应改为 FAR。

(3)由于主程序和子程序已经不在同一程序模块，所以子过程定义和调用都应该是 FAR 类型的，而不应该用 NEAR 型，同时在程序与子程序之间的变量传送也要改变，可用两种方法：

(Ⅰ)使用外部符号：PUBLIC、EXTRN；

(Ⅱ)动态改变 ES 寄存器内容。

6.4 强化训练

1. 求出前20个斐波那契数，存入数据段FN开始的区域中。

2. 编写完成计算3!+4!+5!，并将结果存入寄存器BX中。

3. 编写程序求某数据区中无符号字数据最大值和最小值的差，结果送RESU字单元。要求：最大值和最小值分别用子程序计算，主程序和子程序之间通过寄存器传递参数。

答 案

1. 程序代码如下：

```
DSEG    SEGMENT
F0      DW      0
F1      DW      1
FN      DW      18DUP(?)
DSEG    ENDS
CSEG    SEGMENT
        ASSUME CS:CSEG,DS:DSEG
MAIN    PROC    FAR
        MOV     AX,DSEG
        MOV     DS,AX
        MOV     SI,OFFSET  FN
        MOV     CX,9
        MOV     AX,F0
        MOV     BX,F1
        CALL    FB
        RET
START   ENDP
```

```
FB      PROC
NEXT:   ADD     AX,BX
        ADD     BX,AX
        MOV     [SI],AX
        MOV     [SI+2],BX
        ADD     SI,4
        LOOP    NEXT
        RET
FB      ENDP
CSEG    ENDS
        END     MAIN
```

2. 程序代码如下：

```
CSEG    SEGMENT
        ASSUME  CS:CSEG
MAIN    PROC    FAR
        PUSH    DS
        MOV     AX,0
        PUSH    AX
        MOV     BX,0
        MOV     AX,3
        CALL    JCH
        ADD     BX,AX
        MOV     AX,4
        CALL    JCH
        ADD     BX,AX
        MOV     AX,5
        CALL    JCH
        ADD     BX,AX
```

```
        RET         ;求(AX)! 的子程序,结果存在 AX 中
JCH     PROC
        PUSH    DX
        MOV     DX,AX
        CMP     AX,0
        JZ      DONE
        DEC     AX
        CALL    JCH
        MUL     DX
        POP     DX
        RET
DONE:   MOV     AX,1
        POP     DX
        RET
JCH     ENDP
CSEG    ENDS
        END     MAIN
```

3. 程序代码如下：

```
DSEG    SEGMENT
BUF     DW      11,3,100,10,2000,2006,25,365
COUNT   EQU     ($-BUF)/2
RESU    DW?
DSEG    ENDS
CSEG    SEGMENT
MAIN    PROC    FAR
        ASSUME  CS:CSEG,DS:DSEG
        PUSH    DS
        SUB     AX,AX
```

```
        PUSH    AX
        MOV     AX,DSEG
        MOV     DS,AX
        MOV     SI,OFFSET BUF;SI 用于将 BUF 首
                              地址传递给子程序
        MOV     CX,COUNT ;CX 用于将循环次数传
                          递给子程序
        CALL    MAX
        MOV     SI,OFFSET BUF;SI 用于将 BUF 首
                              地址传递给于程序
        MOV     CX,COUNT ;CX 用于将循环次数传
                          递给子程序
        CALL    MIN
        SUB     BX,DX
        MOV     RESU,BX
        RET
MAIN    ENDP
MAX     PROC
        MOV     BX,[SI];取第一个数据至 BX 中(BX
                        用于放最大值),并将最大
                        值传回主程序
        DEC     CX      ;字数据个数减 1
        ADD     SI,2    ;指向第二个数据
MAX1:   MOV     AX,[SI];取一个数据至 AX 中
        CMP     AX,BX
        JNA     NEXT1;AX 不高于 BX,与下一个比较
        XCHG    AX,BX;AX 高于 BX,则将较大字数据
                      送 BX
```

```
NEXT1:  ADD     SI,2
        LOOP    MAX1
        RET
MAX     ENDP
MIN     PROC
        MOV     DX,[SI];取第一个数据至 DX 中(BX
                        用于放最小值),并将最小值
                        传回主程序
        DEC     CX;字数据个数减 1
        ADD     SI,2;指向第二个字数据
MIN1:   MOV     AX,[SI];取一个字数据至 AX 中
        CMP     AX,DX
        JNB     NEXT2;AX 不低于 DX,与下一个比较
        XCHG    AX,DX;AX 低于 DX,则将较小字数
                        据送 DX
NEXT2:  ADD     SI,2
        LOOP    MIN1
        RET
MIN     ENDP
CSEG    ENDS
        END     MAIN
```

第7章　高级汇编语言技术

7.1　内容提要

7.1.1　宏汇编

在汇编语言程序中，有的程序段要多次使用，为了使这些程序段在源程序中不被重复书写，可以定义一个指令来代替该程序段，由宏汇编程序在汇编时产生所需的代码。

1. 宏的定义

宏定义格式如下：

宏名　MACRO[形参列表]

(宏体)

其中，MACRO 和 ENDM 是一组伪指令，一个宏定义由三部分组成：宏名、宏伪指令(MACRO/ENDM)、宏体。

宏名是唯一的，不能和其他标号和变量名重名。

2. 宏调用

经过宏定义后，宏指令就可以在源程序中引用宏，即宏调用。

宏调用的格式是：

宏名　[实参 1,实参 2,实参 3,…]

如果定义的宏有形参，则宏调用时要用实参来替换。

实参的属性必须与形参相对应。如果实参不止一个，则排列顺序要与形参中的顺序相一致。

3. 宏操作符

(1)操作符 &

操作符 & 在宏定义中使用时可以在形参之前，也可以出现在形参之后，宏展开时将 & 前后两个符号合并成一个符号，该符号可以是操作码、操作数，也可以是字符串。

(2)操作符%

格式：%表达式

汇编程序将操作符%后的表达式的值而不是表达式文本本身作为当前值，宏展开时，用该值取代形参，%操作符不允许出现在形参前面。

(3)操作符〈〉

汇编程序将操作符〈〉括起的内容作为一个字符串处理。在宏调用中，实参如果含有空格、逗号等间隔符，就用操作符〈〉将实参括起，作为一个单一完整的实参。若将一个特殊字符作为实参，也可以用该操作符处理，如〈&〉表示一个字符"&"，而不是操作符。

(4)操作符!

格式：! 字符

汇编程序将操作符! 后的字符只作字符含义进行处理。如"!%"表示%只作百分号使用。在宏调用时，为使实参中的一些特殊字符做一般字符处理，就要在其前加上该操作符。

4. 局部符号伪指令 LOCAL

宏定义体中可以使用标号或变量。

格式：LOCAL　标号表

如标号表中的标号多于一个时，标号间用逗号隔开。需要说明的是，LOCAL 伪指令只用于宏定义体内，且必须是宏定义伪指令 MACRO 之后的第一条语句，在 MACRO 和 LOCAL 伪指令之间不允许有注释和分号标志。

5. 宏嵌套

在宏的定义中可以使用宏调用，称为宏的嵌套。

7.1.2 重复汇编

在程序中，有时需要连续重复地完成一组完全或几乎相同的一组代码，在这种情况下，可以用重复汇编简化源程序代码。重复汇编指源程序中用重复伪指令定义需要重复的语句序列，汇编程序对该语句序列进行多次汇编。

(1)伪指令 REPT

格式：REPT　重复次数

　　(需要重复的语句组)

　　ENDM

重复次数由一数值表达式给定。

(2)伪指令 IRP

格式：IRP　形式参数，[实参 1，实参 2，…，实参 N]

　　(需要重复的语句组)

　　ENDM

(3)伪指令 IRPC

格式：IRPC 形式参数，字符串

　　(需要重复的语句组)

　　ENDM

与 IRP 相似，但实际参数列表是一个字符串。字符串的长度规定了重复次数。

7.1.3 条件汇编

条件汇编的作用是使汇编程序能根据条件有选择地对程序段进行汇编。源程序中依需要设定条件，汇编时汇编程序根据条件是否满足，对该程序进行汇编或不进行汇编，以得到所需的目标代码。

条件汇编语句是由伪指令构成的说明性语句，不是可执行语句，其功能是经过汇编程序汇编后实现的。

格式：IF×××× 条件表达式

语句组 1

[ELSE

语句组 2]

ENDIF

如果条件伪指令要求的条件满足，那么汇编语句组 1，否则不汇编语句组 1；在含有 ELSE 伪指令的情况下，如果条件不满足，则汇编语句组 2。

7.2 例题精解

例 1 定义下列宏并给出宏调用过程：

(1)从堆栈中弹出所有十六位数据寄存器内容；

(2)从键盘输入字符；

(3)两个字操作数相加，得到一个十六位的和；

(4)实现输入一串字符或显示一串提示字符的功能。

【分析与解答】 (1)宏定义：

```
POPREG   MACRO
         POP   DX
         POP   CX
         POP   BX
         POP   AX
         ENDM
```

宏调用：POPREG

(2)宏定义：

```
INPUTKEY   MACRO
```

```
        MOV     AH,1
        INT     21H;DOS系统功能调用,从键盘
                    输入一个字符
        ENDM
```

宏调用:INPUTKEY

(3)宏定义:

```
AOD12   MACRO   A1,A2,A3
        PUSH    DX
        PUSH    AX
        MOV     AX,A1;被加数赋给 A
        ADD     AX,A2;完成加法运算
        MOV     A3,AX;将和放入 A3 所指定的寄存器
                    或内存单元
        POP     AX
        POP     DX
        ENDM
```

宏调用 : `ADD12  CX,DATA,X[SI]`

`ADD12  100,BX,RESULT`

(4)以下用两个宏分别完成回车换行和输入输出字符的功能。

宏定义:

```
ENTER   MACRO               ;定义回车换行宏指令
        MOV     DL,0DH
        MOV     AH,2
        INT     21H
        MOV     DL,0AH
        INT     21H
        ENDM
INOUT   MACRO   X,Y         ;定义输入/输出宏指令
```

```
        MOV     AH,X        ;功能号赋给 AH
        LEA     DX,Y        ;要显示的字符串地址或输
                            入字符的缓冲区地址送 DX
        INT     21H
        ENDM
```

例 2 使用重复汇编结构定义将九九乘法表的数值放入连续的 81 个字节单元中。

【分析与解答】 本题应用重复汇编结构的嵌套模式，内外两层的重复次数都是 9 次。汇编后的代码等价于：

```
        DB  1,2,3,…,9
        DB  2,4,6,…,8
            ……
        DB  9,18,27,…,81
```

所以重复汇编的结构为：

```
NUM1=0
REPT  9
      NUM1=NUM1+1
      NUM2=0
      REPT  9
          NUM2=NUM2+1
          DB  NUM2 * NUM1
      ENDM
ENDM
```

7.3 课后习题解答

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。

【解答】 宏定义：

```
CLRB    MACRO   LEN,POINT
        PUSH    CX
        PUSH    DI
        MOV     CX,LEN
        MOV     AL,' '
        LEA     DI,POINT
        REP     STOSB
        POP     DI
        POP     CX
        ENDM
```

7.2 某工厂计算周工资的方法是每小时的工资率 RATE 乘工作时间 HOUR，另外每工作满十小时加奖金 3 元，工资总数存放在 WAG 中。请将月工资的计算编写成一条宏指令 WAGES，并展开宏调用：

```
WAGES   R1,42
```

【解答】 宏定义：

```
WAGES   MACRO   RATE,HOUR
        LOCAL   L1
        PUSH    AX
        PUSH    CX
        MOV     AX,RATE
        MUL     HOUR
L1:     MOV     CX,HOUR/10
        ADD     AX,3
        LOOP    L1
        MOV     [WAG],AX
        POP     CX
```

```
        POP     AX
        ENDM
```

宏展开：

```
        PUSH    AX
        PUSH    CX
        MOV     AX,R1
        MUL     42
L1:     MOV     CX,42/10
        ADD     AX,3
        LOOP    L1
        MOV     [WAG],AX
        POP     CX
        POP     AX
```

7.3 给定宏定义如下：

```
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)ABSDIP　AX,AX,AX

【解答】

```
(1)         PUSH    AX
            MOV     AX,P1
            SUB     AX,P2
            CMP     AX,0
            JGE     CONT
            NEG     AX
CONT:       MOV     DISTANCE,AX
            POP     AX
```

(2)无效调用
(3)无效调用

```
(4)         PUSH    AX
            MOV     AX,AX
            SUB     AX,AX
            CMP     AX,0
            JGE     CONT
            NEG     AX
CONT:       MOV     AX,AX
            POP     AX
```

7.4 试编制宏定义，要求把存储器中的一个用 EOT 字符结尾的字符串传送到另一个存储区中去。

【解答】 宏定义：

```
TRANS   MACRO   MESS1,MESS2
```

```
        LOCAL   L1,L2
        PUSH    SI
        PUSH    DI
        PUSH    BX
        LEA     SI,MESS1
        LEA     DI,MESS2
L2:     CMP     [SI],EOT
        JZ      L1
        MOV     BX,[SI]
        MOV     [DI],BX
        INC     SI
        INC     DI
        JMP     L2
L1:     MOV     BX,[BI]
        MOV     [DI],BX
        POP     BX
        POP     DI
        POP     SI
        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。

【解答】 宏定义：

```
SUB     MACRO   RESULT,A,B,COUNT
        LOCAL   NEXT
        PUSH    AX
```

```
        PUSH    SI
        PUSH    CX
        CLC
        MOV     CX,COUNT
        MOV     AX,A
        LEA     SI,B
NEXT:   SBB     AX,[SI]
        INC     SI
        LOOP    NEXT
        MOV     RESULT,AX
        POP     CX
        POP     SI
        POP     AX
        ENDM
```

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

【解答】 宏定义如下：

```
DISP    MACRO   X,NAME
        PUSH    DS
        PUSH    DX
        PUSH    AX
        LDS     DX,GOOD
        MOV     AH,9
        INT     21H
        POP     AX
        POP     DX
        POP     DS
```

```
        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
```

【解答】 (1) DATA0 DW ?

(2)DATA1 DW ?

7.8 定义宏指令并展开宏调用。宏指令 JOE 把一串信息'MESSAGENO. 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
```

【解答】宏定义：

```
SINGLE  MACRO   X,K
        X&K     DB  'MESSAGE  NO. &K'
        ENDM
JOE     MACRO   NO,J
```

```
        SINGLE  NO,%J
        J=J+1
        ENDM
```

宏展开：

```
I=0
TEXT0   DB  'MESSAGE  NO.0'
TEXT1   DB  'MESSAGE  NO.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
        ENDIF
        ENDM
```

试展开下列调用：

```
        I=0
        STORE  TAB,7
```

【解答】

```
MOV[TAB],0
MOV[TAB+1],1
MOV[TAB+2],2
MOV[TAB+3],3
MOV[TAB+4],4
MOV[TAB+5],5
MOV[TAB+6],6
```

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

【解答】 宏定义：

```
SINGLE  MACRO  X
        MOV  [TAB+X],X
        ENDM
STORE   MACRO  N
        REPT  N
        SINGLE  %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
```

【解答】 程序代码如下：

```
DSEG    SEGMENT
X       DB      'IS  IT  RIGHT'
N       DB      ?
DSEG    ENDS
IF      N-X     GT  5
        REPT    10
        ADD     AX,AX
        ENDM
```

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←X+2 * Y，否则执行 SUM←2 * X+Y。

【解答】 宏定义：

```
        FINSUM  MACRO  X,Y,SUM
        LOCAL   L1,L2
        PUSH    AX
```

```
        PUSH    BX
        MOV     AX,X
        MOV     BX,Y
        CMP     AX,BX
        JG      L1
        SHL     AX,1
        JMP     L2
L1:     SHL     BX,1
L2:     ADD     AX,BX
        MOV     SUM,AX
        POP     BX
        POP     AX
        ENDM
```

7.13 试编写一宏定义完成以下功能：如变元 X='VT55'，则汇编 MOV TERMINAL,0，否则汇编 MOV TERMINAL,1。

【解答】 宏定义：

```
TEST    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,OFFSETBUFF 指令，并展开以下宏调用：

```
        DOS21   01
        DOS21   0AH,IPFIELD
```

【解答】 宏定义：

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

宏调用：

①DOS2101：

```
        MOV     AH,01H
        INT     21H
```

②DOS210AH,IPFIELD：

```
        MOV     AH,0AH
        MOV     DX,OFFSET  IPFIELD
        INT     21H
```

7.15 编写一段程序，使汇编程序根据 SIGN 的值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。

【解答】 程序代码如下：

```
        MOV     AL,DIVD
```

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

7.16 试编写宏定义 SUMMING，要求求出双字节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。

【解答】 宏定义：

```
SUMMING  MACRO      MRRAY,COUNT,RESULT
         PUSH       CX
         PUSH       EAX
         PUSH       SI
         MOV        EAX,0
         MOV        SI,0
         MOV        CX,COUNT
L0:      ADD        EAX,ARRAY[SI*4]
         INC        SI
         LOOP       L0
         MOV        RESULT,EAX
         POP        SI
         POP        EAX
         POP        CX
SUMMING  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  ?
```

【解答】 由 7.16 题的宏定义：

```
SUMMING  DATA,4,SUM
```

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

【解答】 在程序前加上：

```
include  macro.mac
```

7.4 强化训练

1. 指出下面宏定义 PICT 完成的功能。

```
PICT  MACRO  NUM,LETTERSD,EXIT
      IFE    NUM  -2
      MOV    AH,2
      MOV    DL,LETTERSD
      IFE    NUM  -9
      MOV    AH,9
      LEA    DX,LETTERSD
      ELSE
      JMP    EXIT
      ENDIF
      ENDM
```

2. 编写宏指令实现以下功能：

(1)显示数据段中的一个字符串；

(2)从键盘输入字符串到数据段缓冲区；

(3)将 3 个数中最小的数放入 AL；

(4)连续显示指定数目的字符，字符串首地址和显示个数为参数。

3. 有一首地址为 NUM 的 N 字数组，编写程序使该数组中的数按照从小到大的次序排序。

答　案

1. 根据 DOS 功能调用不同的功能码(2 或 9)分别产生显示字符或字符串的程序段，否则退出。

2. (1)
```
DISP    MACRO   STRSN
        PUSH    DX
        PUSH    AX
        LEA     DX,STRSN
        MOV     AH,09H
        INT     21H
        POP     AX
        POP     DX
        ENDM
```
(2)
```
INCHARS MACRO   BUFFER
        PUSH    DX
        PUSH    AX
        LEA     DX,BUFFER
        MOV     AH,0AH
        INT     21H
        POP     AX
        POP     DX
        ENDM
```
(3)
```
MIN     MACRO   X,Y,Z
        LOCAL   NEXT,EXIT
```

```
        MOV     AL,X
        CMP     B,AL
        JNLE    NEXT
        MOV     AL,B
NEXT:   CMP     C,AL
        JNLE    EXIT
        MOV     AL,C
EXIT:   ENDM
(4)DISP MACRO   LETTERS,NUM
        LOCAL   L1
        MOV     CX,NUM
        MOV     BX,0
L1:     MOV     DL,LETTERS[BX]
        MOV     AH,02H
        INT     21H
        INC     BX
        LOOP    L1
        ENDM
3. SORT MACRO
LOCAL   L1,L2,CNT
L1:     MOV     DI,CX
        MOV     BX,0
L2:     MOV     AX,NUM[BX]
        CMP     AX,NUM[BX+2]
        JL      CNT
        XCHG    AX,NUM[BX+2]
        MOV     NMU[BX],AX
CNT:    ADD     BX,2
```

```
        LOOP    L2
        MOV     CX,DI
        LOOP    L1
        ENDM
DSEG    SEGMENT
        N       EQU10
        NUM     DW  01H,02H,03H,04H,
                    05H,06H,07H,08H,
                    09H,0AH
DSEG    ENDS
CSEG    SEGMENT
        ASSUME CS:CSEG,DS:DSEG
MAIN    PROC    FAR
        PUSH    DS
        MOV     AX,0
        PUSH    AX
        MOV     AX,DSEG
        MOV     DS,AX
        MOV     CX,K
        DEC     CX
        SORT
        RET
START   ENDP
CSEG    ENDS
        END     MAIN
```

第8章 输入输出程序设计

8.1 内容提要

8.1.1 I/O设备的数据传送方式

CPU与外设之间的数据传送方式有：程序直接控制I/O方式，中断传送方式，直接存储器传送(DMA)方式等。

1. 程序直接控制I/O方式

这种方式使用I/O指令(IN和OUT)直接在端口级上进行信息的输入/输出。CPU与各设备之间以串行方式工作。CPU要通过测试I/O接口的状态来控制传送，若I/O设备没有准备好，CPU就循环测试，直到设备准备好，CPU就执行一次传送。

2. 中断传送方式

(1)这种I/O方式实质上是一种特殊情况下的程序转移方式。所谓特殊情况一般是指：

①计算机出现异常事件，如电源掉电，内存或I/O总线奇偶错等。出现这样的事件，CPU应立即中断现行程序的运行，转去执行处理故障的子程序。

②程序中预先安排的中断指令(INT)或其他内部原因(如除法错等)，使现行程序暂时中断，转去执行相应的处理子程序。

③外部设备一切准备就绪时，向CPU发出中断现行程序的请求，以处理外设的输入输出。

以上三类情况是引起中断产生的原因，称为中断源。第①类情况一般安排为非屏蔽中断。第②类情况称为内中断，第③类情况为外中断，这是一些可屏蔽的中断类型。

(2)中断过程

当中断发生时，由中断机构自动完成下列动作：

①取中断类型号 N；

②标志寄存器(FLAGS)内容入栈；

③当前代码段寄存器(CS)内容入栈；

④当前指令计数器(IP)内容入栈；

⑤禁止硬件中断和单步中断(IF＝0，TF＝0)；

⑥从中断向量表中取 4×N 的字内容送 IP，取 4×N＋2 中的字内容送 CS；

⑦转中断处理程序。

在中断响应过程中将标志寄存器 PSW，CS 和 IP 三个字压入堆栈，中断处理程序在返回时应该从堆栈中弹出返回地址(CS，IP)和原状态值。这个任务由中断返回指令完成。

中断返回指令的格式如下：

IRET

该指令的功能是从中断返回。具体操作如下所示：

IP←[SP]

SP←SP＋2

CS←[SP]

SP←SP＋2

FLAGS←[SP]

(3)中断程序的设计方法

下面是主程序为响应中断所做的准备工作以及硬件(包括 CPU 和外设接口)自动完成的动作：

主程序
- 设置中断向量
- 设置设备的中断屏蔽位
- 设置 CPU 的中断允许位 IF(开中断)

硬件
- 外设接口送中断请求给 CPU
- 当前指令执行完后,CPU 送响应信号给外设接口
- CPU 接收中断类型号
- 当前的 FLAGS,CS 和 IP 保存入栈
- 清除 IF 和 TF
- 中断向量送 IP 和 CS

中断处理程序的编写方法和标准子程序很类似。下面是编写中断处理子程序的步骤,但它与子程序编写也有一些不同之处:

①保存寄存器内容;

②如允许中断嵌套,则开中断(STI);

③处理中断;

④关中断(CLI);

⑤送中断结束命令(EOI)给中断命令寄存器;

⑥恢复寄存器内容;

⑦返回被中断的程序(IRET)。

3. 直接存储器传送方式

对于高速 I/O 设备(如磁盘机)准备数据的时间短,数据传送的速度快,这样的设备与 CPU 交换数据采用直接存储器传送方式。

系统完成 DMA 传送的步骤如下:

①DMA 控制器向 CPU,发出 HOLD 信号,请求使用总线。

②CPU 发出响应信号 HOLD 给 DMA 控制器,并将总线让出,这时 CPU 放弃了对总线的控制,而 DMA 控制器获得了总线控制权。

③传输数据的存储器地址(在地址寄存器中)通过地址总线发

出。

④传输的数据字节通过数据总线进行传送。

⑤地址寄存器增 1，以指向下一个要传送的字节。

⑥字节计数器减 1。

⑦如字节计数器非 0，转向③。

⑧否则，DMA 控制器撤销总线请求信号 HOLD，传送结束。

8.1.2 BIOS/DOS 中断调用

1. BIOS/DOS 简介

固化在 ROM 中的基本输入输出系统 BIOS 包含了主要 I/O 设备的处理程序和许多常用例行程序，它们一般以中断处理程序的形式存在。

磁盘操作系统 DOS 建立在 BIOS 的基础上，通过 BIOS 操纵控制硬件。DOS 调用 BIOS 显示输出程序完成显示输出，调用 BIOS 打印输出程序完成打印输出，调用 BIOS 键盘输入程序完成键盘输入。DOS、BIOS 和硬件接口都为应用程序提供完成输入输出的功能，而且随着层次的加深访问外部设备的能力越强；从应用程序的编写角度出发，随着层次的加深，应用程序的编写难度和复杂度大大增加。

通常 I/O 程序应该由 DOS 提供的系统功能，完成输入输出，这样实现容易，而且对硬件的依赖性最少。如果 DOS 不提供某种服务或者不能使用 DOS 的场合可考虑 BIOS 调用。应用程序可以直接操纵外设接口来控制外设，从而获得速度上的最高效率。

2. 常用的 DOS 功能调用

(1)DOS 键盘中断

表 8-1 列出了与 DOS 键盘操作有关的“DOS 21H”功能调用。

表 8-1 DOS 键盘操作

功能号	调用参数	返回参数	功 能
01		AL=输入字符	键盘输入并回显
06	DL=FF	若有字符可取，AL=字符，ZF=0； 若无字符可取，AL=0，ZF=1	读键盘字符
07		AL=输入字符	键盘输入（无回显）
08		AL=输入字符	键盘输入（无回显）检测 Ctrl_Break
0A	DS:DX=缓冲区首地址 (DS:DX)=缓冲区最大字符数	(DS:DX+1)=实际输入的字符数	键盘输入到缓冲区
0B		AL=00 有输入 AL=FF 无输入	检验键盘状态
0C	AL=输入功能号(1,6,7,8,A)		清除输入缓冲区并请求指定的输入功能

(2)DOS 显示功能调用

表 8-2 列出了与 DOS 显示操作有关的“DOS 21H”功能调用。

表 8-2 DOS 显示操作

功能号	调用参数	功 能
02	DL＝输出字符	显示输出一个字符(检验 Ctrl_break)
06	DL＝输出字符	显示输出一个字符(不检验 Ctrl_break)
09	DS：DX＝串地址'$'结束字符串	显示字符串

(3)打印机 I/O 中断

表 8-3 列出了与打印机 I/O 中断有关的功能调用。

表 8-3 DOS/BIOS 打印操作

INT	功能号	调用参数	返回参数	功 能
21H	5	DL＝字符		打印一个字符
17H	0	AL＝字符 DX＝打印机号	AH＝状态字节	打印一个字符，并返回状态字节
17H	1	DX＝打印机号	AH＝状态字节	初始化打印机，回送状态字节
17H	2	DX＝打印机号	AH＝状态字节	回送状态字节

8.1.3 考 点

1. 输入输出指令；

2. 查询方式和中断传送方式传送数据程序设计；

3. 键盘、显示器 DOS 中断程序设计方法。

8.2 例题精解

例 1 设某输入设备有两个 8 位端口：数据口 DIN 和状态口 STATUS。状态位中的 D7 表示输入准备是否就绪。编写查询方式输入程序。

【分析与解答】 查询方式输入数据的流程如图 8.1 所示。

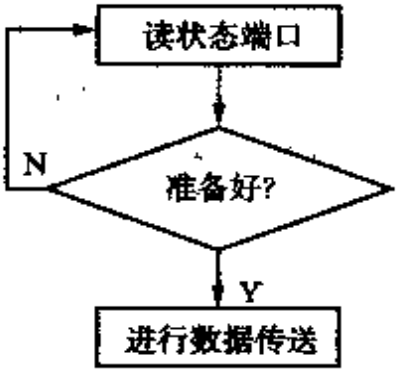


图 8.1 查询方式输入数据

```
        ……
TEST:   IN      AL,STATUS
        TEST    AL,80H
        JZ      TEST
        IN      AL,DIN
```

例 2 三个输入设备 A,B,C 的 I/O 程序起始地址分别为 PRO1,PRO2,PRO3,它们共用一个状态口 STATUS,D2,D1,D0 作为它们的设备就绪状态位，当状态位为 0 时，表示设备未准备好，当状态位为 1 时，表示设备就绪。编写查询方式输入程序。

【分析与解答】 查询方式输入数据的流程同于例 1，但要考虑顺序查询 A,B,C 三个设备。主要程序代码如下：

```
START:IN        AL,STATUS
      TEST      AL,04H
      JNZ       PRO1
      TEST      AL,02H
      JNZ       PRO2
      TEST      AL,01H
```

```
        JNZ     PRO3
        JMP     START
PRO1:   ……
        JMP     START
PRO2:   ……
        JMP     START
PRO3:   ……
        JMP     START
```

例 3 分别利用 DOS 和 BIOS 键盘中断功能调用编程。要求：检测功能键 F1，如有 F1 键按下，则转 HELP 执行。

【分析与解答】 由于功能键没有 ASCII，在采用 DOS INT 21H 键盘功能调用键盘输入时，如果有功能键输入，那么返回的字符码都为 00H。因此采用 DOS INT 21 键盘功能调用读功能键输入时，必须进行两次 DOS 功能调用。第一次回送 00，第二次回送扫描码。

主要程序代码如下：

```
X0:     MOV     AH,07
        INT     21H         ;等待键盘输入
        CMP     AL,0        ;是否为功能键
        JNE     X0          ;不是，继续等待
FKEY:   MOV     AH,07H
        INT     21H
        CMP     AL,3BH      ;是不是 F1
        JNE     X0          ;不是，继续等待
HELP:   ……
```

例 4 编写一程序，要求打印从键盘接收的数字，如输入非数字键，则退出程序。

【分析与解答】 程序代码如下：

```
DSEG    SEGMENT
        CHAR    DB 50 DUP(?)
        DSEG    ENDS
CSEG    SEGMENT
PRTCHAR PROC    FAR
        ASSUME  CS:CSEG,DS:DSEG
START:  PUSH    DS
        XOR     AX,AX
        PUSH    AX
        MOV     AX,DSEG
        MOV     DS,AX
        STI
        CLD
        MOV     AH,0    ;初始化打印机
        MOV     DX,0
        INT     17H
GETCH:  MOV     AH,01   ;读键盘
        INT     21H
CHKCH:  CMP     AL,30H  ;检查是否为数字键盘
        JB      EXIT
        CMP     AL,39H
        JA      EXIT    ;不是数字键,退出
PRINT:  MOV     DL,AL   ;是数字键,打印
        MOV     AH,05
```

```
        INT     21H
        JMP     GETCH ;继续接收字符
EXIT:   RET
PRTCHAR ENDP
CSEG    ENDS
        END     START
```

8.3 课后习题解答

8.1 写出分配给下列中断类型号在中断向量表中的物理地址。

(1) INT 12H

(2) INT 8

【解答】 (1)12H * 4=48H

(2)8H * 4=20H

8.2 用CALL指令来模拟实现INT 21H显示字符T的功能。

【解答】
```
MOV   DL,'T'
MOV   AH,02H
CALL  0086H:0084H
```

8.3 写出指令将一个字节数据输出到端口25H。

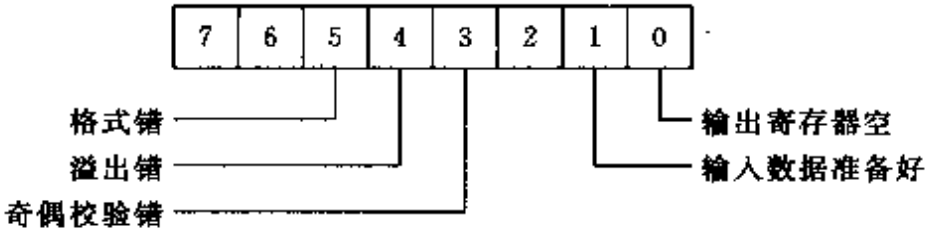
【解答】 OUT 25H,AL

8.4 写出指令将一个字数据从端口1000H输入。

【解答】
```
MOV   DX,1000H
IN    AX,DX
```

8.5 假定串行通信口的输入数据寄存器的端口地址为50H,

状态寄存器的端口地址为51H，状态寄存器各位为1时含义如下：



请编写一程序：输入一串字符并存入缓冲区BUFF，同时检验输入的正确性，如有错，则转出错处理程序ERR_ROUT。

【解答】 主要程序代码如下：

```
        LEA     SI,BUFF
        MOV     CX,0FFH
L1:     IN      AL,51H
        TEST    AL,02H
        JZ      L1
        IN      AL,50H
        MOV     [SIJ],AL
        INC     SI
        IN      AL,51H
        TEST    AL,38H
        JNZ     ERROR
        LOOP    L1
        JMP     EXIT
ERROR:CALL      ERR_OUT
EXIT:   ……
```

8.6 试编写程序，它轮流测试两个设备的状态寄存器，只要一个状态寄存器的第0位为1，则与其相应的设备就输入一个字

符；如果其中任一状态寄存器的第 3 位为 1，则整个输入过程结束。两个状态寄存器的端口地址分别是 0024 和 0036，与其相应的数据输入寄存器的端口则为 0026 和 0038，输入字符分别存入首地址为 BUFF1 和 BUFF2 的存储区中。

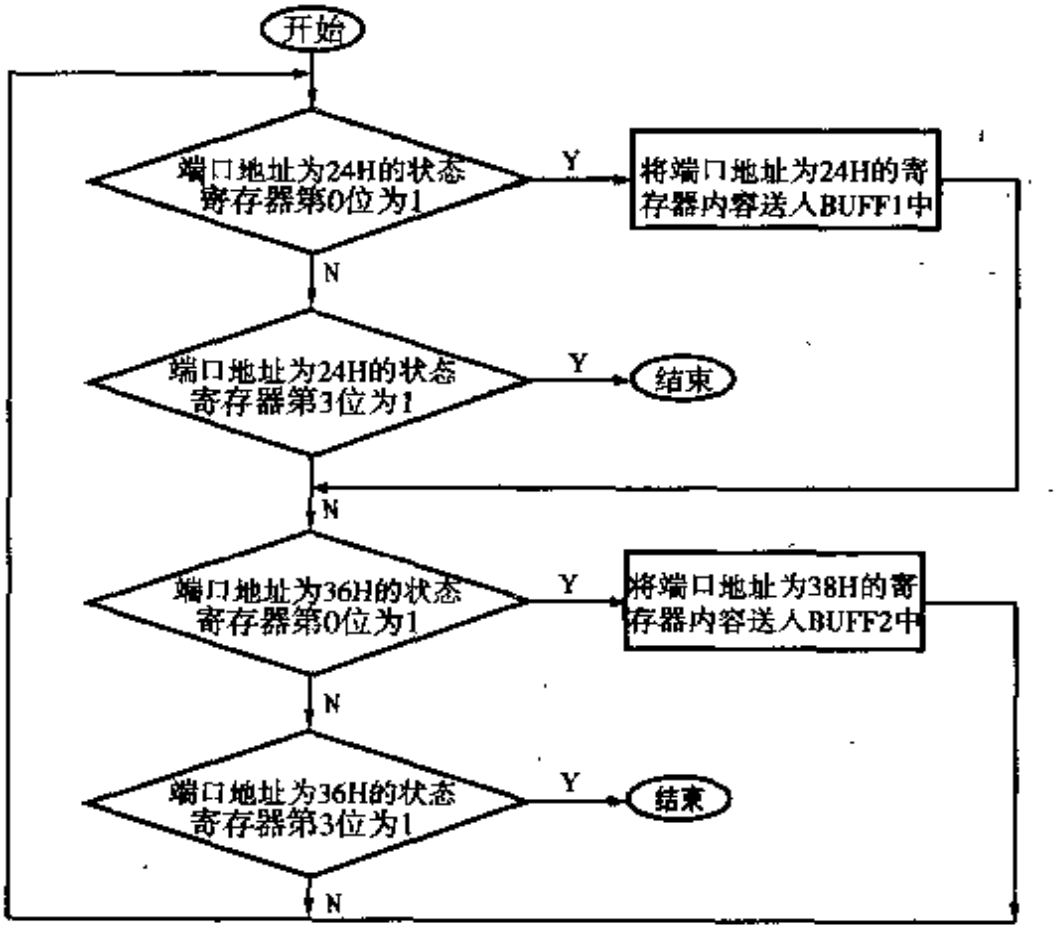


图 8.2

【解答】 程序流程图如图 8.2 所示。

主要程序代码如下：

```
L4:     IN      AL,24H
        TEST    AL,01H
        JNZ     L1
```

```
        TEST    AL,08H
        JNZ     L2
L5:     IN      AL,36H
        TEST    AL,01H
        JNZ     L3
        TEST    AL,08H
        JNZ     L2
        JMP     L4
L1:     IN      AL,26H
        MOV     BUFF1,AL
        JMP     L5
L3:     IN      AL,38H
        MOV     BUFF2,AL
        JMP     L4
L2:     RET
```

8.7 假设外部设备有一台硬币兑换器，其状态寄存器的端口地址为0006，数据输入寄存器的端口地址为0005，数据输出寄存器的端口地址为0007。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，当状态寄存器的第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，壹角纸币的代码为01，贰角纸币代码为02，伍角纸币代码则为03，然后程序在等待状态寄存器的第3位变为1后，把应兑换的伍分硬币数(用十六进制)从数据输出寄存器输出。

【解答】 程序流程图如图8.3所示。

主要程序代码如下：

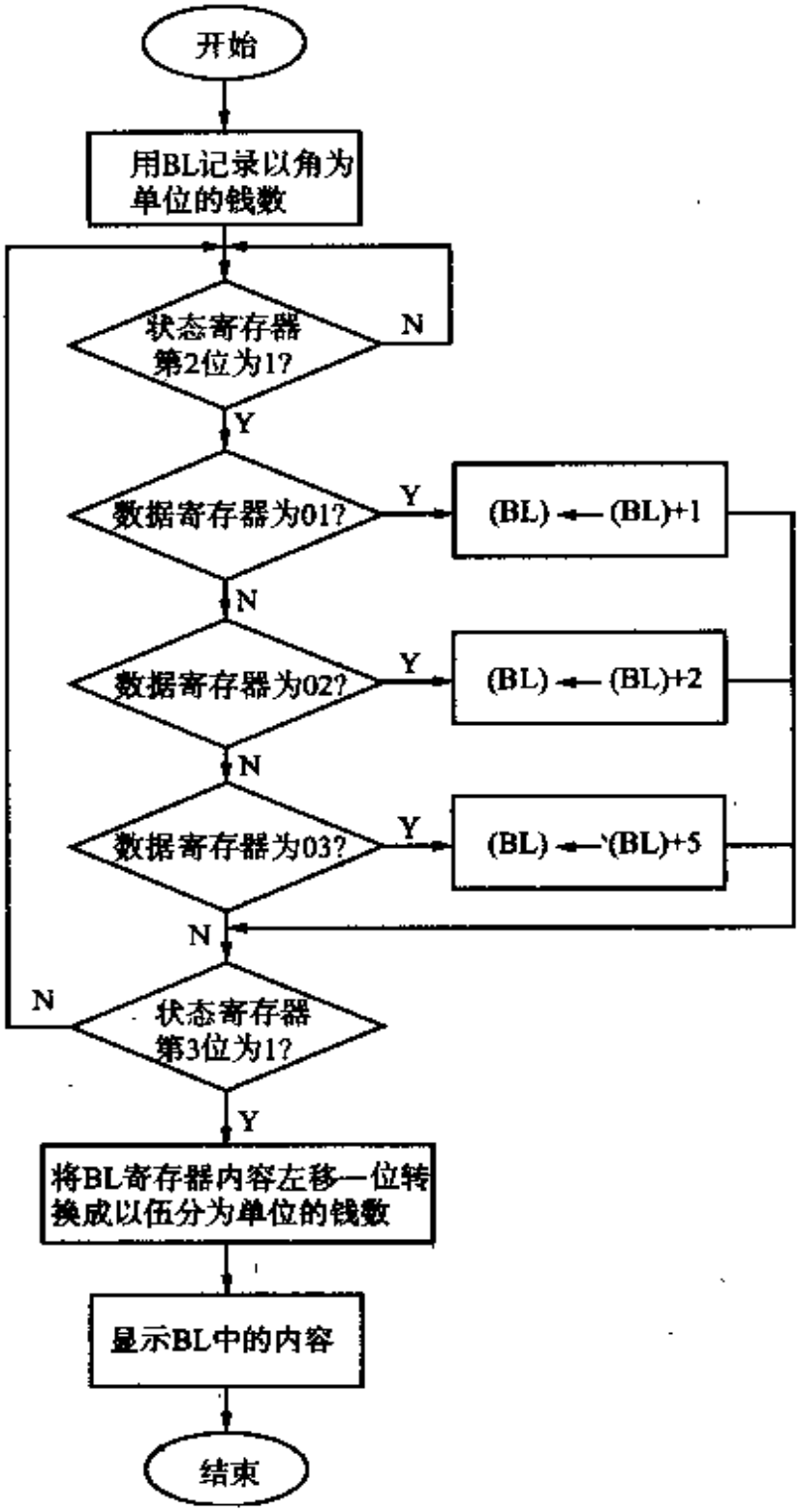

开始
用BL记录以角为单位的钱数
状态寄存器第2位为1?
N
Y
数据寄存器为01?
Y
(BL)←(BL)+1
N
数据寄存器为02?
Y
(BL)←(BL)+2
N
数据寄存器为03?
Y
(BL)←(BL)+5
N
状态寄存器第3位为1?
N
Y
将BL寄存器内容左移一位转换成以伍分为单位的钱数
显示BL中的内容
结束

图 8.3

```
        MOV     BL,0
L1:     IN      AL,06H
        TEST    AL,04H
        JZ      L1
        IN      AL,05H
        CMP     AL,01
        JNZ     L2
        ADD     BL,01
        JMP     L4
L2:     CMP     AL,02
        JNZ     L3
        ADD     BL,02
        JMP     L4
L3:     JMP     AL,03
        JNZ     L4
        ADD     BL,05
L4:     IN      AL,06H
        TEST    AL,08H
        JZ      L1
        SHL     BL,1
        MOV     AL,BL
        OUT     07H,AL
```

8.8 给定(SP):0100,(SS)=0300,(FLAGS)=0240,以下存储单元的内容为(00020)=0040,(00022)=0100,在段地址为

0900 及偏移地址为 00A0 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？

【解答】 栈顶三个字如图 8.4 所示。

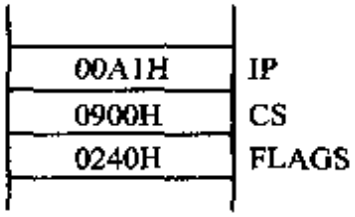


图 8.4

(SP)＝0100H－6＝00FAH

(SS)＝0300H

(IP)＝0040H

(FLAGS)＝0240H

8.9 中断类型的中断向量在存储器的哪些单元里？

【解答】 存放在 0：50H、0：51H、0：52H 和 0：53H 四个字节地址单元中。

8.10 假设中断类型 9 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。

【解答】
```
PUSH    DS
LDS     DX,IN_ROUT
MOV     AL,09
MOV     AH,25H
INT     21H
POP     DS
```

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。

【解答】
```
PUSH    DS
LDS     DX,SHOW_CLOCK
```

```
MOV     AL,1CH
MOV     AH,25H
INT     21H
POP     DS
```

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 3 和 4 同时发出中断请求;

(2)在设备 3 的中断处理程序完成之前,设备 2 发出中断请求;

(3)在设备 4 的中断处理程序未发出中断结束命令(EOI)之前,设备 5 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 1,3,5 同时发出中断请求。

【解答】 D3,D2,D3,D4,D5,D1,D3,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?

【解答】 D3,D2,D4,D5,D1,D3,D5

8.14 试编制一程序,要求测出任意程序的运行时间,并把结果打印出来。

【解答】 如果被测程序执行前,设备时钟计数值为 0,开始以计数频率 18.2Hz/s 进行计数。同时执行被测程序,结束后再取当前时钟计数值,便得到程序执行时间。

1 小时对应计数值为 65 520 次,1 分钟对应计数值为 1092 次,1 秒对应计数值为 18.2 次(取 18 次)。

在时、分、秒值不超过 24 和 60 的情况下,为将它们的个位与

十位值分开，可采用乘法十进制调整指令 AAM 进行，对 AL 除以 10，商送 AH，余数保留在 AL 中。

程序代码如下：

```
DSEG    SEGMENT
T       DB      0       ;时计数值
M       DB      0       ;分计数值
S       DB      0       ;秒计数值
TN      EQU     65520   ;求小时数的常数
MN      EQU     1092    ;求分钟数的常数
SN      EQU     18      ;求秒数的常数
TASCH   DB      0       ;小时高位的 ASCII 码
TASCL   DB      0       ;小时低位的 ASCII 码
MASCH   DB      0       ;分钟高位的 ASCII 码
MASCL   DB      0       ;分钟低位的 ASCII 码
SASCH   DB      0       ;秒高位的 ASCII 码
SASCL   DB      0       ;秒低位的 ASCII 码
DSEG    ENDS
        CSEG    SEGMENT
        ASSUME CS:CSEG,DS:DSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        MOV     CX,0
        MOV     DX,0
        MOV     AH,1
        INT     1AH         ;设置时间值为 0
        CALL    TESTPRO     ;调用被测程序
        MOV     AH,0
        INT     1AH         ;读取时间值在 CX 和
```

```
                                DX 中
XCHG    AX,DX       ;低位时间交换至 AX 中
MOV     DX,CX       ;高位时间值至 DX 中
MOV     CX,TN
DIV     CX
MOV     T,AL        ;时计数值送 T
MOV     AX,DX       ;余数小于 65520
MOV     CX,MN
XOR     DX,DX
DIV     CX
MOV     M,AL        ;分计数值送 M
MOV     AX,DX       ;余数小于 1092
MOV     CL,18
DIV     CL
MOV     S,AL
XOR     AH,AH
AAM                 ;高位 BCD 秒值在 AH
                    中,低位 BCD 秒值在
                    AL 中
OR      AX, 3030H   ;将秒计数值转为
                      ASCII 码
MOV     SASCH,AH
MOV     SASCL,AL
XOR     AX,AX
MOV     AL,M
AAM
OR      AX, 3030H   ;将分钟计数值转为
                      ASCII 码
```

```
        MOV     MASCH,AH
        MOV     MASCL,AL
        XOR     AX,AX
        MOV     AL,T
        AAM
        OR      AX,3030H ;将小时计数值转为
                         ASCII码
        MOV     TASCH,AH
        MOV     TASCL,AL
        MOV     SI,OFFSET TASCH
        MOV     DL,[SI]
        MOV     AH,2
        INT     21H         ;显示时
        INC     SI
        MOV     DL,[SI]
        INT     21H         ;显示分
        INC     SI
        MOV     DL,[SI]
        INT     21H
        MOV     DL,':'
        INT     21H
        INC     SI
        MOV     DL,[SI]
        INT     21H         ;显示秒
        MOV     DL,[SI]
        INT     21H
        RET
TEST    PROC    NEAR        ;显示A～E字符的子程序
```

```
        PUSH   AX
        PUSH   DX
        PUSH   CX
        MOV    CX,5
        MOV    DL,41H    ;送'A'的 ASCII 码
AGAIN:  MOV    AH,2
        INT    21H
        INC    DL
        LOOP   AGAIN
        POP    CX
        POP    DX
        POP    AX
        RET
TEST    ENDP
CSEG    ENDS
        END    START
```

8.4 强化训练

1. 完善下面程序，并写出此程序的功能。

```
MAXNO   EQU     41
SSEG    SEGMENT
DW      100     DUP(?)
SSEG    ENDS
DSEG    SEGMENT
MESG1   DB      'INPUT A STRING:$'
MESG2   DB      'IT IS REVERSE IS:$'
BUFFER  DB      MAXNO,?,MAXNO DUP(?)
```

```
DSEG        ENS
CSEG        SEGMENT
            ASSUME    CS:CSEG,DS:DSEG
START:      (1)
            MOV       DS,AX
            MOV       DX,OFFSET  MESG1
            MOV       AH,9
            INT       21H
            MOV       DX,OFFSET  BUFFER
            (2)       ;设置子功能号
            INT       21H
            XOR       AX,AX
            (3)       ;取实际读入字符个数
            LEA       DI,BUFFER+2
            MOV       SI,DI
            ADD       SI,AX
            MOV       BYTE  PTR[SI],'S'
            DEC       SI
COUNT:      CMP       DI,SI
            (4)
            MOV       AL,[SI]
            XCHG      AL,[DI]
            DEC       SI
            (5)       ;调整 DI
            JMP       COUNT
PINISH:     LEN       DX,BUFFER+2
            MOV       AH,9
            INT       21H
```

```
        RET
CSEG    ENDS
        END     START
```

2. 下面是一个与显示有关的子程序，读完后回答后面的问题。

```
PROG1   PROC    FAR
        PUSH    ES
        PUSH    DI
        MOV     AX,3
        INT     10H             ;①
        MOV     DI,0
        SHL     CX,1            ;②
        MOV     AL,50H
        MUL     CL
        ADD     DI,AX
        ADD     DI,DX
        ADD     DI,DX
        MOV     AX,0730H
        MOV     BX,0BB00H
        MOV     ES,BX
        OR      ES:[DI],AX      ;③
        POP     DI
        POP     ES
        RET
PROG1   ENDP
```

(1)子程序完成了什么功能？

(2)①处执行后的物理意义是什么？

(3)②处 CX 为什么要移位？

(4)③处 AH 中值的物理意义是什么?

3. 编程实现以下功能:

从终端输入字符保存在一个 64 字节的数组 BUFFER 中,当输入一个回车符时,输入结束。如果输入的前 63 个字符没有发现回车符,则从终端输出信息"BUFFER OVERFLOW"。输入字节的第 7 位为偶校验位,如果发生偶校验错,则转向出错处理程序 ERROR,如无校验错,则将字节的校验位清 0 后送 BUFFER。

假设终端接口的数据输入寄存器的端口地址是 52H,数据输出寄存器是 53H,状态寄存器是 54H,其中第 1 位为 1 表示输入寄存器数据准备好,第 0 位为 1 表示输出寄存器是空闲的。

4. 请使用 BIOS INT 17H 完成下列程序。

(1)设置换页方式。

(2)执行换行并打印地址。

答 案

1. (1)MOV　　AX,DSEG
   (2)MOV　　AH,10
   (3)MOV　　AL,BUF+1
   (4)JAE　　FINISH
   (5)INC　　DI

功能:从键盘读入一个字符串,将该串反转后,输出显示。

2. (1)在指定位置处显示出黑底白字的字符。
   (2)设置显示模式。
   (3)一个字符占两个字节。
   (4)字符的属性。

3. 程序代码如下:

```
MESSAGE  DB      'BUFFER OVERFLOW $'
         DB      0DH,0AH
```

```
BUFFER    DB      64 DUP(?)
          ……
          MOV     DI,OFFSET BUFFER
          MOV     CX,63H
NEXT:     IN      AL,54H
          TEST    AL,02
          JZ      NEXT
          IN      AL,52H
          OR      AL,0
          JPE     NOERR
          JMP     ERROR
NOERR:    AND     AL,7FH
          MOV     [DI],AL
          INC     DI
          CMP     AL,0DH
          JZ      EXIT
          LOOP    NEXT
OVER:     MOV     SI,OFFSET MESSAGE
          MOV     CX,16
OUT:      IN      AL,54H
          TEST    AL,01
          JZ      OUT
          MOV     AL,[SI]
          INC     SI
          OUT     53H,AL
          LOOP    OUT
          ……
ERROR:    ……
```

```
EXIT:       ……
4.(1)MOV    AH,0
     MOV    AL,0CH
     MOV    DX,0
     INT    17H
(2)ADDRESS  DB     AH,'ADDRESS NO:'
   LEN      EQU    $－ADDRESS
            ……
            LEA    SI,ADDRESS
            MOV    CX,LEN
   NEXT:    MOV    AH,0
            MOV    AL,[SI]
            MOV    DX,0
            INT    17H
            INC    SI
            LOOP   NEXT
```

第9章 课程考试复习指导

由于汇编语言本身的特性，此课程存在一定难度。为了帮助同学更加系统地掌握本课程的知识及考试重点，故设置本章节来进行考前集中训练。

- 总复习提纲
  - 基础知识
    - 计算机系统、汇编语言简介
    - 计算机中的数据表示方法
    - 8086/8088 微处理器及其寻址方式
    - 汇编语言程序结构
  - 程序设计
    - 数据传送程序
    - 算术运算程序
    - 逻辑运算程序
    - 分支程序
    - 字符串操作程序
    - 循环程序
    - 过程
    - 宏汇编、重复汇编、条件汇编
    - 中断及输入输出

考试主要题循：单选据，简答据，程序分析题，完善程序题以及编程题。下面将分题型进行训练，以下题目均出自名校试题。

单选题

1. 完整的计算机系统应包括(　　)。

A. 运算器、存储器、控制器　B. 主机和外围

C. 主机和应用程序　D. 配套的硬件设备和软件系统

2. 一个字符的基本 ASCII 值占用(　　)位(二进制)。

A. 6　B. 7

C. 8　D. 9

3. 下列表达式中正确的运算结果为(　　)。

A. $(10101)_2 \times (2)_{10} = (20202)_2$

B. $(10101)_2 \times (2)_{10} = (20202)_2$

C. $(10101)_2 \times (3)_{10} = (30303)_2$

D. $(101010)_2 - (20202)_2 = (11011)_2$

4. 将高级语言的程序翻译成机器语言，代码程序的实用程序是(　　)。

A. 编绎程序　B. 汇编程序

C. 解释程序　D. 目标程序

5. 十进制数－100 的 8 位二进制数的补码为(　　)。

A. 11100100　B. 01100100

C. 10011100　D. 11001110

6. 若$[X]_{补} = 11010011$，则 X 的十进制数真值是(　　)。

A. 71　B. 48

C. 65　D. 63

7. 十六进制数 88H，可表示成下面几种形式，请找出错误的表示(　　)。

A. 无符号十进制数 136

B. 有符号十进制数－120

C. 压缩型 BCD 码十进制数 88

D. 8 位二进制数－8 的补码

8. 以下四个数中，数值最小的一个是(　　)。

A. 二进制数 111011　B. 八进制数 77

C. 十进制数 57　　D. 十六进制数 3C

9. 在 8086/8088 系统中，存储器是分段组织的，每段最大字节的长度为(　　)。

A. 8KB　　B. 64KB

C. 1MB　　D. 不确定

10. 设 SP 的初值为 1000H，执行指令 PUSH BX 后，SP 值为(　　)。

A. 0FFAH　　B. 0FFEH

C. 1002H　　D. 0FFFH

11. 能保存各逻辑段的起始地址的寄存器称为(　　)。

A. 地址寄存器　　B. 段寄存器

C. 数据寄存器　　D. 计数寄存器

12. 在 CPU 中，暂存指令的寄存器是(　　)。

A. 数据寄存器　　B. 程序计数器

C. 状态条件寄存器　　D. 指令寄存器

13. 在汇编语言程序中，反映指令操作结果的标志是(　　)。

A. 状态标志　　B. 控制标志

C. SF　　D. OF

14. 在 8086/8088 中，寄存器 BP 的功能是(　　)。

A. 存放段的地址

B. 用于变址操作

C. 存放堆栈段的偏移地址

D. 存放代码段的偏移地址

15. 在串操作指令中，源串操作数的段地址一定在(　　)寄存器中。

A. CS　　B. SS

C. DS　　D. ES

16. 可用作基址变址寻址或寄存器间接寻址的寄存器是

(　　)。

A. AX,BX,CX,DX　　B. DS,ES,CS,SS

C. SP,BP,IP,BX　　D. SI,DI,BP,BX

17. 设(DS)＝8225H,(DI)＝3942H,指令 MOV AX,[DI]中源操作数的物理地址是(　　)。

A. 85B92H　　B. 86192H

C. BB692H　　D. 12169H

18. 设段地址和偏移地址是 2F4BH：00C7H,那么它所对应存储单元中的物理地址是(　　)。

A. 3F4B7H　　B. 3F598H

C. 3F577H　　D. 3EB76H

19. 在计算机中,以先进后出方式工作的存储空间是(　　)。

A. 存储器　　B. RAM

C. ROM　　D. 堆栈

20. 设栈 S 的初始状态为空,现有 5 个元素组成的序列{1,2,3,4,5},对该序列在栈 S 上依次进行 PUSH、PUSH、POP、PUSH、POP、PUSH、PUSH 操作,出栈的元素序列是(　　)。

A. 5,4,3,2,1　　B. 2,1

C. 3,4　　D. 2,3

21. 完成将累加器 AL 清单,并使进位标志 CF 清零,下面错误的指令是(　　)。

A. MOV　AL,00H　　B. AND　AL,00H

C. XOR　AL,AL　　D. SUB　AL,AL

22. 在需要从端口 365H 读入数据时,下列指令中(　　)是正确的。

A. IN　　AL,365H

B. OUT　365H,AL

C. MOV　AL,365H

D. MOV DX,365H
IN AL,DX

23. 下列指令中错误的是(　　)。

A. ADD SS:[BX+DI],2008H

B. MOV DX,2008H

C. MOV WORD PTR[BX],2008H

D. MOV DS,2008H

24. 对两个无符号数 A、B 进行比较时,采用 JBE/JNA 指令,转移的条件是(　　)。

A. CF=0 或 ZF=1　　B. CF=0 或 SF=1

C. CF=1 或 ZF=1　　D. ZF=0 或 SF=0

25. 下列指令中,影响条件标志位 CF 的指令是(　　)。

A. JC LOP　　B. SHL BX,1

C. INC AX　　D. JMP LOP

26. 在程序状态字寄存器中,当方向标志 DF 位为 0 时,每次操作后使变址寄存器 SI 和 DI(　　)。

A. 减量　　B. 增量

C. 保持不变　　D. 地址减 1

27. 汇编语言源程序中,每个语句由 4 项组成,如果语句要完成一定功能,那么该语句中不可省略的项是(　　)。

A. 名字项　　B. 操作项

C. 操作数项　　D. 注释项

28. 设 BL = 40H, SI = 0600H, DS = 3000H, (30600H) = 0C0H, CF=1。执行"SBB BL,[SI]"后,正确的结果是(　　)。

A. BL=5FH,SF=1,CF=1

B. BL=5FH,SF=0,CF=1

C. BL=60H,SF=1,CF=0

D. BL=60H,SF=0,CF=0

29. 在串操作指令前使用重复前缀指令 REPE，终止串的重复操作条件是(　　)。

A. CX＝0 且 ZF＝0　　B. CX＝0 且 ZF＝1

C. CX＝0 或 ZF＝0　　D. CX＝0 或 ZF＝1

30. 当一个带符号数大于 0FBH 时程序转移，需选用的条件转移指令是(　　)。

A. JLE　　B. JNL

C. JNLE　　D. JL

31.
```
MOV    AL,85H
SUB    AL,0ABH
```

上述指令执行后，标志位 CF 和 OF 的值是(　　)。

A. CF＝0，OF＝1　　B. CF＝1，OF＝1

C. CF＝0，OF＝0　　D. CF＝1，OF＝0

32. 下列有关输入、输出指令中，错误的指令是(　　)。

A. IN　AL，DX　　B. OUT　DX，AL

C. IN　AX，DX　　D. OUT　AL，DX

33. 在汇编语言程序中，对 END 语句的叙述正确的是(　　)。

A. END 语句是一可执行语句

B. END 语句是表示程序执行到此结束

C. END 语句是表示源程序到此结束

D. END 语句是在汇编后要产生机器码

34. 在段定义时，如果定义类型用户未选择，就表示是隐含类型，其隐含类型是(　　)。

A. WORD　　B. PAGE

C. BYTE　　D. PARA

35. 现“MOV BX，OFFSET VAR”指令完全等效的指令是(　　)。

A. MOV BX,VAR　　B. LDS BX,VAR

C. LES BX,VAR　　D. LEA BX,VAR

36. 8086 的存储器是分段的，定义一个段的伪指令是(　　)。

A. PROC 和 END P　　B. NAME 和 END

C. SEGMENT 和 ENDS　　D. SEGMENT 和 ASSUME

37. 若 AX=－15，要得到 AX=15 应执行的指令是(　　)。

A. NEG AX　　B. NOT AX

C. INC AX　　D. DEC AX

38. 有表达式 MOV BX,((VAL LT 5)AND 20)OR((VAL GT 5)AND 30)，若 VAL<5，则结果是(　　)。

A. MOV BX,20　　B. MOV BX,30

C. MOV BX,1　　D. MOV BX,0

39. 下列指令中错误的是(　　)。

A. PUSH WORD PTR 20[BX+SI－2]

B. LEA BX,4[BX]

C. MOV BYTE PTR[BX],32

D. MOV BYTE PTR[BX],[SI]

40. 完成同指令“XCHG AX,BX”相同功能的指令或指令序列是(　　)。

A. MOV AX,BX

B. MOV BX,AX

C. PUSH AX
　POP BX

D. MOV CX,AX
　MOV AX,BX
　MOV BX,CX

41. “LDS SI,ES;[2000H]”指令的全部功能是(　　)。

A. 把地址 2000H 送 SI

B. 把地址 2000H 字单元的内容送 SI

C. 把地址 2000H 字单元内容送 SI，把 2002H 字单元内容送 DS

D. 把地址 2000H 字单元内容送 DS，把 2002H 字单元内容送 SI

42. 在执行 POP[BX]指令，寻找目的操作数时，段地址和偏移地址分别是在(　　)。

A. 无段地址和偏移地址　　B. 在 DS 和 BX 中

C. 在 ES 和 BX 中　　D. 在 SS 和 SP 中

43. 在汇编语言程序设计中可使有“LEA BX，VAR”和“MOV BX，OFFSET VAR”，这两条指令取得变量 VAR 的偏移地址，关于这两条指令的执行速度，正确的结论是(　　)。

A. LEN　BX，VAR 指令快

B. MOV　BX，OFFSET VAR 指令快

C. 两条指令的执行速度相同

D. 由变量 VAR 的类型决定这两条指令的执行速度

44. 下列指令中错误的是(　　)。

A. ADD　BX，OFFSET C

B. MUL　20

C. IN　　AL，DX

D. SUB　AX，0FH

45. 执行 INC 指令除对 SF、ZF 有影响外，还要影响的标志位是(　　)。

A. OF，AF，PF　　B. OF，AF，CF

C. OF，PF，CF　　D. AF，PF，CF

46. 在 AL 和 VAR 字节单元中分别存放一个带符号数，执行“CMP AL，VAR”时，如 AL＞VAR，那么溢出位和符号位的关系是(　　)。

A. OF=SF　　B. OF≠SF

C. OF=SF=1　　D. OF=SF=0

47. 设 AL=0B4H,BL=11H,指令"MUL BL"执行后 OF,CF 的值为(　　)。

A. OF=1,CF=0　　B. OF=1,CF=1

B. OF=0,CF=0　　D. OF=0,CF=1

48. 指令"JMP FAR PTR LOP"属于(　　)。

A. 段内直接寻址　　B. 段内间接寻址

C. 段间直接寻址　　D. 段间间接寻址

49. 在下列指令的表示中,不正确的是(　　)。

A. MOV　AL,[BX+SI]　　B. JMP　SHORT DON1

C. DEC　[BX]　　D. MUL　CL

50.
```
LOP:  MOV   DL,[SI]
      MOV   [DI],DL
      INC   SI
      INC   DI
      LOOP  LOP
```

与上述程序段完成同样功能的一条指令是(　　)。

A. REP　LODSB　　B. REP　MOVSB

C. REP　STOSB　　D. REP　SCASB

51. 为使(CX)=-1 时转至 MINUS,编制了下面的指令序列,其中错误的序列是(　　)。

A. INC　CX　JZ　MINUS

B. SUB　CB,OFFFFH　JZ　MINUS

C. AND　CX,OFFFFH　JZ　MINUS

D. XOR　CX,OFFFFH　JZ　MINUS

52. 下列指令执行后总是使 CF=0,OF=0 的是(　　)。

A. AND　　B. NEG

C. NOT　　D. INC

53. 完成将累加器 AL 清零，并使进位标志 CF 清零，下面错误的指令是(　　)。

A. MOV　AL,00H　　B. AND　AL,00H

C. XOR　AL,AL　　D. SUB　AL,AL

54. 实现将(DX：AX)中存放的 32 位数扩大 4 倍，正确的程序段是(　　)。

A. SHL　AX,2
　ROL　DX,2

B. RCL　AX,2
　SHL　DX,2

C. MOV　CX,2
　LOP：SHL　AX,1
　　RCL　DX,1
　　LOOP　LOP

D. SHL　AX,1
　SHL　AX,1
　RCL　DX,1
　RCL　DX,1

55. 下列指令中，执行后，不改变标志位 ZF 的是(　　)。

A. CMP　AL,BL　　B. AND　AL,AL

C. TEST　AL,0FFH　　D. ROR　AL,CL

56. 在循环右移指令 ROR 操作中，移位完成后，操作数的最高位是(　　)。

A. 随机　　B. 1

C. 0　　D. 最低位数

57. 若(AL)＝10101101B，为了使其内容变为 01010010B，下列(　　)指令可完成此操作

A. NOT AL　　B. OR AL,AL

C. XOR AL,AL　　D. AND AL,AL

58. 在执行串操作指令时,使地址按递增方式处理,应使用的指令是(　　)。

A. STD　　B. CLD

C. STI　　D. CLI

59. 下面各组伪指令中,使用错误的是(　　)。

A. STACK_SEG SEGMENT 'STACK'

B. SEGMENT 'CODE'

C. MAIN_PROC PROC FAR

……

MAIN_PROC NEDP

END MAIN_PROC

D. MYDATA SEGMENT 'DATA'

……

ENDS

60. 从键盘输入数据 1,则在存储单元中存放的形式为(　　)。

A. 31H　　B. 01H

C. '1'　　D. "1"

61. 下面指令语句中,语法正确的是(　　)。

A. INC [BX]　　B. CMP [BX],60

C. JMP FAR OPR　　D. MOV WORD PTR[BX],5

62. 用 CMP 指令对两个无符号数进行 A－B 的比较后,如 A=B 或 A>B,则分别产生转移,这种情况应选择的条件转移指令是(　　)。

A. 先用 JE 指令,再用 JNC 指令

B. 先用 JNC 指令,再用 JE 指令

C. 上述两条条件转移指令(JE 和 JNC)无先后次序

D. 用上述两条条件转移指令不能完成上述功能要求

63. 下列叙述中(　　)情况属于子程序的递归调用。

A. 主程序调用子程序　　B. 子程序 1 调用于程序 2

C. 程序 2 调用于程序　　D. 子程序 2 调用于程序 2

64. 子程序是通过(　　)来定义的。

A. CALL～RET　　B. PROC～ENDP

C. SUB～PROC　　D. PROC～RET

65. 在汇编语言程序设计中,若调用不在本模块中的过程,则对该过程必须用伪操作命令(　　)说明。

A. ASSUME　　B. EXTRN

C. COMMON　　D. PUBLIC

66. 在汇编语言程序的开发过程中使用宏功能的顺序是(　　)。

A. 宏定义,宏调用　　B. 宏定义,宏展开

C. 宏定义,宏调用,宏展开　　D. 宏定义,宏展开,宏调用

67. 宏指令名及其参数分别出现在操作符域和操作数域,称为(　　)。

A. 宏定义　　B. 宏调用

C. 宏展开　　D. 宏嵌套

68. 中断过程分为四个步骤,这四个步骤是(　　)。

A. 中断响应、中断请求、中断处理、中断返回

B. 中断请求、中断响应、中断处理、中断返回

C. 中断处理、中断请求、中断响应、中断返回

D. 中断处理、中断请求、中断返回、中断响应

69. 硬中断服务程序结束通回断点时,程序末尾要安排一条指令 IRET,它的作用是(　　)。

A. 构成中断结束命令

B. 返回到断点处

C. 恢复断点信息并返回

D. 转移到 IRET 的下一条指令

70. 在下列指令中，能使 80x86CUP 对 I/O 端口进行读写访问的是(　　)。

A. 串操作指令　　B. 中断指令

C. 输入/输出指令　　D. 数据传送指令

答　案

1～5	D B B A C	6～10	C D C B B
11～15	B D A B C	16～20	D A C D D
21～25	A D D C B	26～30	B B B C C
31～35	D D C D D	36～40	A D B D D
41～45	C B B B A	46～50	D B D C B
51～55	C A A C D	56～60	D A B C A
61～65	D A D B B	66～70	C B C C C

简答题

1. 解释如下汇编语言的基本概念：

汇编语言、汇编语言源程序、汇编、汇编程序。

【答】

汇编语言——用指令助记符、符号地址、标号等符号书写程序的语言。

汇编语言源程序——用汇编语言编写的程序称为汇编语言源程序，简称源程序。

汇编——把源程序翻译成机器语言的程序(目标程序)的过程叫做汇编。用汇编语言编写的源程序不能直接在微机上运行。它

必须转换成等效的机器语言程序才能运行。这种转换是通过汇编程序自动完成的。

汇编程序——完成汇编任务的程序称为汇编程序。汇编程序的主要功能是将汇编语言源程序翻译成机器程序，此外还具有一些其他功能，如：根据用户指定自动分配存储区域；自动的将各种进制数转换成二进制数，把字符转换成 ASCII 码，计算表达式的值等；自动对照源程序进行检查，给出错误信息。

2. 试述汇编语言的特点。在什么情况下和场合下要用到或使用汇编语言编程？

【答】 汇编语言的特点如下：

(1)与机器相关性。汇编语言指令是机器指令的一种符号表示，不同类型的 CUP 有不同的机器指令系统，故有不同的汇编语言。

(2)执行的高效率。汇编语言的执行速度快，执行效率高。

(3)编写程序的复杂性。汇编语言是一种面向机器的语言，编程时要安排运算的每一个细节，面此程序的编写过程比较繁琐、复杂。

(4)调试的复杂性。在通常情况下，调试汇编语言程序要比调试高级语言程序困难。

以下情况可以考虑使用汇编语言：

(1)要求执行效率高、反应快的领域，如：操作系统内核、工业控制、实时控制等。

(2)系统性能的瓶颈，或在大程序中频繁被使用子程序或程序段。

(3)软件与硬件资源密切，软件要直接和有效控制硬件的场合。

(4)对执行时间和存储容量要求较高的场合。

(5)没有合适的高级语言的场合。

3. 请给出标志寄存器中标志位 OF、IF、SF、ZF、PF 和 CF 的说明 。

【答】

OF:溢出标志位。运算溢出时自动置1,当它为1时可用溢出中断指令产生中断。

IP:中断允许标志位。若IF=1,开中断,响应可屏蔽中断;IF=0,关中断。

SF:符号标志位。运算结果首位为1时置1,用作标志结果数的符号。

ZF:零标志位。当运算结果为0时置1。

PF:奇偶标志位。结果中有偶数个1时置1,否则为0,用以检验传送结果是否有误。

CF:进位/借位标志位。在进行字/字节运算产生进位/借位时置1。

4. 什么是逻辑地址?什么是物理地址?逻辑地址由哪些部分组成?如何由逻辑地址得到物理地址?

【答】 逻辑地址是编程时使用的地址,物理地址是存储器单元的地址编号。

逻辑地址是由段地址(包括该单元的段的首地址)和偏移地址(段首址到该单元的距离)组成。

段地址与偏移地址通过地址加法器相加,就可以得到20位的物理地址。具体为:物理地址=段地址*16+偏移地址。

5. 循环指令LOOP、LOOPZ、LOOPNZ的测试条件是什么?循环指令执行的步骤是怎样的?

【答】

LOOP指令的测试条件是CX≠0。

LOOPZ指令的测试条件是ZF=1且CX≠0。

LOOPNZ指令的测试条件是ZF=0且CX≠0。

这三条指令的执行步骤如下:

(1)CX−1→CX。

(2)检查是否满足测试条件,如果满足,则继续循环。

(3)如果不满足测试条件,则退出循环,程序继续顺序执行。

6. 采用子程序结构进行程序设计的优点在哪里?

【答】 采用子程序结构进行程序设计的优点如下:

(1)简化了程序设计的过程,节省程序设计的时间;

(2)缩短了程序的长度,节省了程序的存储空间;

(3)增加了程序的可读性,便于对程序的修改、调试;

(4)方便了程序的模块化、结构化和自顶向下的程序设计。

7. 简述调用指令 CALL 和转移指令 JMP 之间的主要区别。

【答】 CALL 指令是调用指令,JMP 指令是跳转指令,都可以实现指令的转移,不同之处在于 CALL 指令需要保护断点,以便正确返回,而 JMP 指令无须断点保护。

8. 子程序之间参数传递的三种方法是什么? 它们有什么特点和区别? 分别用于什么情况?

【答】 子程序中参数传递的三种方法是:寄存器传递、存储单元传递法和堆栈传递法。各自的特点分别是:

寄存器传递法简单常用、直观。但由于寄存器个数的限制,通常用于要传递参数较少的情况。

存储单元传递法不受参数的多少限制,但编制起来比较麻烦,尤其当参数与调用程序不在同一数据段的时候。

堆栈法比较方便,运用于嵌套或递归程序中。

9. 在指令系统中,段内、段间返回均为 RET 指令,试回答:

(1)执行段内返回 RET 指令时,执行的操作是什么?

(2)执行段间返回 RET 指令时,执行的操作是什么?

【答】

(1)执行段内返回 RET 指令时,执行的操作是:

(SP+1,SP)→IP,SP+2→SP(弹出断点的偏移地址)

(2)执行段间返回 RET 指令时,执行的操作是:

(SP+1,SP)→IP,SP+2→SP(弹出断点的偏移地址)

(SP+1,SP)→CS,SP+2→SP(弹出断点的段地址)

10. 宏指令和子程序有什么异同?

【答】 宏指令和子程序都是功能独立的一段程序,目的是简化源程序。但宏指令只是为了简化源程序的书写,在汇编时,汇编程序处理宏指令,把宏定义体插入到宏调用处。所以,宏指令并没有简化目标程序。有多少次宏调用,在目标程序中仍需要有同样多次的目标代码插入。因此,宏指令没有能节省目标程序所占的内存单元。

子程序在执行时是由 CPU 处理的。若在一个主程序中多次调用同一个子程序,在目标程序的代码中,主程序中仍然只有调用指令的目标代码,子程序的代码只有一份。所以子程序能节省目标程序所占的内存单元。

11. 程序控制 I/O 方式的特点是什么?

【答】 程序控制 I/O 方式的特点是:输入/输出操作完全在用户程序的控制之下进行,程序利用 IN、OUT 指令,根据设备的特点直接访问 I/O 端口。

12. 输入/输出控制方式有哪几种?

【答】 输入/输出控制方式有三种,分别是:程序控制的 I/O 方式、中断控制方式和直接内存访问方式。

13. 什么是中断向量、向量地址和中断向量表?

【答】 中断服务程序的入口地址称为中断向量。

存放中断向量的地址称为向量地址。

中断向量表是用来按中断类型号存放多个中断源对应的中断处理程序首地址的一段内存区。

14. 简要写出 INT n 指令的执行过程。

【答】 INT n 指令的执行过程为:

(1)SP=SP-2

(2)SS:[SP]← PSW

(3)SP＝SP－2

(4)SS:[SP]← INT n 下一条指令的 CS

(5)SP＝SP－2

(6)SS:[SP]← INT n 下一条指令的 IP

(7)IP ← [0000：N＊4]

(8)CS ← [0000：N＊4＋2]

15. 逐条写出以下指令序列在执行中 SP 的值，SP 所指单元的值。

```
MOV     SP,211EH
MOV     AX,0201H
MOV     BX,0A71H
MOV     DI,3202H
PUSH    BX
PUSH    DI
POP     BP
ADD     BP,AX
PUSH    AX
POP     BX
PUSH    BP
```

【解答】

```
MOV     SP,211EH;(SP)＝211EH,栈顶为原值
MOV     AX,0201H;(SP)、栈顶无变化
MOV     BX,0A710H;(SP)、栈顶无变化
MOV     DI,3202H;(SP)、栈顶无变化
PUSH    BX;(SP)＝211CH,栈顶值为 0A710H
PUSH    DI;(SP)＝211AH,栈顶值为 3202H
POP     BP;(SP)＝211CH,栈顶值为 0A710H
ADD     BP,AX;(SP)、栈顶无变化
```

```
PUSH    AX;(SP)=211AH,栈顶值为 0201H
POP     BX;(SP)=211CH,栈顶值为 0A710H
PUSH    BP;(SP)=211AH,栈顶值为 3403H
```

16. 给定(BX)=137DH,(SI)=2A9BH,位移量 D=6239H,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 的间接寻址;

(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。

【解答】

(1)无有效地址,操作数为指令中的立即数;

(2)6239H;

(3)无有效地址,操作数为 637DH;

(4)137DH;

(5)137DH+6239H=75B6H;

(6)137DH+2A9BH=3E18H;

(7)137DH+2A9BH+6239H=A051H

分析程序题

1. 假设数据区有:

```
DA1     DB      56H
DA2     DB      34H
DA3     DB      ?
DA4     DB      ?
```

(1)下列程序段完成的功能是什么?

(2)程序段执行后 DA3、DA4 字节单元的内容是什么?

```
        MOV     AL,DA1
        ADD     AL,DA2
        JO      NEXT
        MOV     DA3,AL
        MOV     DA4,0
        JMP     EXIT
NEXT:   MOV     DA4,1
EXTT:   HLT
```

【解答】 (1)两个数相加,判断是否溢出。

(2)(DA3)＝8AH,(DA4)＝0

2. 下列程序段的功能是什么?

```
        MOV     AX,X
        CMP     AX,Y
        JGE     L
        XCHG    AX,Y
L:      MOV     X,AX
```

【解答】 判断 AX 中内容的正负。若为正,(AX)＝1,若为负,(AX)＝－1。

3. 下列程序段执行后完成什么功能? 程序段执行后 AX 寄存器的内容是多少?

```
        MOV     AX,2
        MOV     DX,AX
        SAL     AX,1
        SAL     AX,1
        ADD     AX,DX
        SAR     AX,1
```

【解答】 上述程序段完成的功能：(AX) * 5/2。

执行后(AX)=0005H。

```
4.      MOV     AX,4BD5H
        MOV     BL,0
        MOV     DL,0
        MOV     CX,16
L1:     SHL     AX,1
        JC      L2
        INC     BL
        JMP     L3
L2:     INC     DL
L3:     LOOP    L1
        HLT
```

【解答】 上述程序段完成的功能：统计 AX 中 1 和 0 的个数，BL 中存放 0 的个数，DL 中存放 1 的个数。

程序段执行后：(BL)=7，(DL)=0，(AX)=0。

5. 下列程序段执行后完成的功能是什么？

```
        DA1     DB  500 DUP(?)
        DA2     DB  100 DUP(?)
        ……
        MOV     CX,100
        MOV     BX,400
        MOV     SI,0
        MOV     DI,0
LOP:    MOV     AL,DA1[BX][SI]
        MOV     DA2[DI],AL
        INC     SI
        INC     DI
```

```
        LOOP    LOP
```

【解答】 将 DA1 中的后 200 个数据送入 DA2 中。

6. 下列程序段执行后，完成什么功能？DX 中的值表示的意义是什么？

```
        BUFF    DB    'BEIJING2008$,GOOD%LUCK$'
        COUNT   EQU   $-BUFF
        ……
        CLD
        LEA     DI,BUFF
        MOV     CX,COUNT
        MOV     AL,'$'
        XOR     DX,DX
NEXT:   REPNZ   SCASB
        CMP     CX,0
        JZ      L1
        INC     DX
        JMP     NET
L1:     ……
```

【解答】 DX 中的值表示在字符串 BUFF 中字符'$'的个数。

7. 下列程序段执行后完成什么功能？程序段执行后 AL 中的内容是什么？

```
        MOV     AL,11H
        XOR     AL,3
```

【解答】 上述程序段执行后将 AL 中内容的第 0 位和第 1 位取反，程序段执行后 AL 中的内容为 12H。

8. 已知 STRING 开始的存储区中，存有一字符串，字符串以回车符(ASCII 0DH)为结束标志，程序段执行后，变量 L 的含义是什么？

```
DSEG    SEGMENT
        STRING  DB  'BEIJING 2008…',0DH
        L       DW  ?
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS:CSEG,DS:DSEG
START:MOV       AX,DSEG
MOV     DS,AX
        MOV     BX,OFFSET STRING
        MOV     CX,0
        MOV     AL,0DH
LOP:    CMP     AL,[BX]
        JE      DONE
        INE     BX
        INC     CX
        JMP     LP
DONE:   MOV     L,CX
        RE
CSEG    ENDS
        ENDS    START
```

【解答】 LOP 的功能是比较 AL 中的内容与 BX 所指单元的内容，若相等，转向 DONE，否则 CX 中的内容加 1，BX 指向下一个字节，继续比较。可见该循环体的作用就是扫描字符串，直到找到回车符。DONE 所执行的功能是把 CX 中的内容存入 L。程序执行后，L 中的数值即为字符串的长度。

9. 阅读以下程序：

```
DSEG1       SEGMENT
STRING1     DB  'COMPUTER SCIENCE'
```

```
STRING2    DB   20 DUP(?)
DSEG1  ENDS
       ……
CSEG   SEGMENT
       ASSUME  CS;CSEG,DS：DSEG,ES：ESEG
START:
       ……
       CLD
       MOV     CX,15
       LEA     SI,STRING1
       LEA     DI,STRING2
       REP     MOVSB
       ……
CSEG   ENDS
       END     START
```

上述程序段执行后，完成什么功能？

【解答】 将 STRING1 中的 15 个字符移动到 STRING2 中。

10. 阅读以下程序段：

```
       XOR     AL,AL
       CALL    SUBROUT
       MOV     BL,AL
       CALL    SUBROUT
       RCR     AL,1
       HLT
SUBROUT  PROC NEAR
       NOT     AL
       JS      NEXT
       STC
```

```
NEXT: RET
SUBROUT  ENDP
```

上述程序执行后，AL、BL 中的内容分别是什么？

【解答】 (AL)＝80H，(BL)＝0FFH。

完善程序题

1. 已知内存变量 NUM 中存有 16 位的二进制数，下面的程序将该二进制数的每一位转换为相应的 ASCII，并存入串变量 STRING 中，请完善该程序。

```
DSEG   SEGMENT
NUM    DW  4F78H
STRING DB  16 DUP(?)
DSEG   ENDS
STACK  SEGMENT  PARA  STACK  'STACK'
       DB  10 DUP(?)
STACK  ENDS
CSEG   SEGMENT
       ASSUME  (1)
START: MOV     AX,DATA
       MOV     DS,AX
       MOV     ES,AX
       MOV     DI, (2) STRING
       MOV     CX,LENGTH  STRING
       PUSH    DI
       MOV     (3) ,30H
       REP     STOSB
       POP     (4)
```

```
        POP     (5)
        MOV     AL,31H
        MOV     BX,NUM
AGAIN:  RCL     BX,1
        (6)
        MOV     [DI],AL
NEXT:   INC     (7)
        LOOP    AGAIN
        MOV     AH, (8)
        INT     21H
CSEG    ENDS
        END     (9)
```

【解答】

(1)CS：CSEG,DS：DSEG,ES：DSEG,SS：STACK

(2)OFFSET

(3)AL

(4)CX

(5)DI

(6)JNC NEXT

(7)DI

(8)4CH

(9)START

2. 下列程序分别统计 ARRAY 数组中奇、偶的个数，请完善该程序。

```
DSEG    SEGMENT
ARRAY DW 2008H,2009H,2010H,2008H,2009H,2010H
COUNT EQU ($-ADR)/2
DA1     DB    ?
```

```
DA1     DB      ?
DSEG    ENDS
        ……
        LEA     SI,ARRAY
        MOV     CX,COUNT
L1:     MOV     AX,[SI]
        (1)
        JZ      L2
        INC     BL
        JMP     L3
L2:     INC     BH
L3:     (2)
        DEC     CX
        JNZ     L1
        MOV     DA1,BL
        MOV     DA2,BH
```

【解答】(1)TEST AX,0001H

(2)ADD SI,2

3. 以下程序是将 ADR1 为起始地址的字节存储若干个字符，统计非数字字符的个数，并将结果回送到 ADR2 单元中。请完善该程序。

```
        XOR     BX,BX
        LEA     SI,ADR1
        MOV     CX,NUM
        XOR     AX,AX
L1:     MOV     AL,[SI]
        CMP     AL,30H
        (1)
```

```
        CMP     AL,39H
        (2)
        JMP     L3
L2:     INC     BL
L3:     (3)
        (4)
        MOV     ADR2,BL
        HLT
```

【解答】

(1)JZ　　L2

(2)JMP　　L4

(3)INC　　SI

(4)LOOP　　L1

4. 以下程序段计算：2＋4＋…＋20 共 10 个偶数的累加和。请完善该程序。

```
        DSEG    SEGMENT
        SUM     DW?
        ……
        XOR     AX,AX
        (1)
        MOV     BX,2
L1:     ADD     AX,BX
        INC     BX
        INC     BX
        (2)
        JNZ     L1
        MOV     SUM,AX
```

【解答】

(1)MOV CX,10

(2)DEC CX

5. 以 DATA 为起始地址的数组中存放 N 个有符号数据，下面程序完成了找出 N 个数中的最大和最小值分别送入 AH 和 AL 中。请完善该程序。

```
        DATA    DB  1,-2,3,-4,…,2008;N个数
        COUNT   EQU  $-DATA
        ……
        MOV     SI,ADR
        MOV     CX,COUNT
        MOV     BH,[SI]
        MOV     BL,BH
L1:     ____(1)____
        CMP     AL,BH
        ____(2)____
        MOV     BH,AL
        ____(3)____
L2:     CMP     AL,BL
        JGE     L3
        MOV     BL,AL
L3:     DEC     CX
        ____(4)____
        MOV     AX,BX
```

【解答】

(1)LODSB

(2)JLE L2

(3)JMP L3

(4)JNZ L1

编程题

1. 编制程序将 16 位二进制数转换成 ASCII 码表示的 5 位十进制。

【解答】 设要转换的 16 位二进制存放在数据区 BINA 的字单元中，转换后的 5 位十进制 ASCII 码串存放在 ADEC 为首地址的 5 个连续字节单元中。

程序代码如下：

```
DSEG    SEGMENT
        BINA    DB   051CH
        ADEC    DB   5 DUP(?)
        P10TAB  DW   10000,1000,100,10,1
DSEG    ENDS
DSEG    SEGMENT
        ASSUME  CS：CSEG,DS：DSEG
MAIN    PROC    FAR
START:  MOV     AX,DSEG
        MOV     DS,AX
        LEA     DI,ADEC
        MOV     AX,BINA
L0：    XOR     CL,CL
        MOV     BX,[SI]
L1：    SUB     AX,BX
        JB      NEXT
        INC     CL
        JMP     L1
NEXT：  ADD     AX,BX
```

```
        ADD     CL,30H
        MOV     [DI],CL
        INC     SI
        INC     SI
        INC     DI
        CMP     BX,1
        JNZ     L0
        RET
MAIN    ENDP
CSEG    ENDS
        END     START
```

2. 从键盘读入一字符串(长度小于40),将该串反转后,输出显示。

【解答】 程序代码如下:

```
MAXNO   EQU  51
SSEG    SEGMENT   SATACK
        DW 100 DUP(?)
SSEG    ENDS
DSEG    SEGMENT
MESG1   DB 'INPUT A STRING: $'
MESG2   DB 'IT'S REVERSE IS: $'
BUF     DB MAXNO,?,MAXNO DUP(?)
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS:CSEG,DS:DSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        MOV     DX,OFFSET MESG1
        MOV     AH,9
```

```
        INT     21H
        MOV     DX,OFFSET BUF
        MOV     AH,10
        INT     21H
        XOR     AX,AX
        MOV     AX,BUF+1    ;取实际读入字符个数
        LEA     DI,BUF+2
        MOV     SI,DI
        ADD     SI,AX
        MOV     BYTE PTR[SI],'$'
        DEC     SI
COUNT:  CMP     DI,SI
        JAE     FINISH
        MOV     AL,[SI]
        XCHG    AL,[DI]
        DEC     SI
        INC     DI          ;调整 DI
        JMP     COUNT
FINISH: LEA     DX,MSEG2
        MOV     AH,9
        INT     21H
        LEA     DX,BUF+2
        MOV     AH,9
        INT     21H
        RET
CSEG    ENDS
        END     START
```

3. 编制程序将字符串按从大到小的顺序输出(利用冒泡

排序法)。

【解答】 程序代码如下:

```
DSEG    SEGMENT
STR     DB 0DH,0AH,'CAPTIAL $'
COUNT EQU       $ -STR
STR1    DB 0DH,0AH,'BEFORE SORT $'
STR2    DB 0DH,0AH,'AFTER $'
DSEG    ENDS
CSEG    SEGMENT
        ASSUME CS : CSEG,DS : DSEG
START:MOV       AX,DSEG
        MOV     DS,AX
        MOV     DX,OFFSET STR1
        MOVE    AH,9
        INT     21H
        MOV     DX,OFFSET STR
        CALL    OUPUT
        MOV     CX,COUNT-4
L1:     MOV     SI,OFFSET STR+2
        MOV     BX,CX
L2:     MOV     AL,[SI]
        CMP     AL,[SI+1]
        JLE     NEXT
        XCHG    AL,[SI+1]
        MOV     [SI],AL
NEXT:   INC     SI
        DEC     BX
        JNZ     L2
```

```
        LOOP    L1
        MOV     DX,OFFSET STR 2
        CALL    OUTPUT
        MOV     DX,OFFSET STR
        CALL    OUTPUT
        RET
OUTPUT  PROC
        MOV     AH,9
        INT     21H
OUTPUT  ENDP
CSEG    ENDS
        END     START
```

4. 设有两个长度相等的字符串分别存放在以 STR1 和 STR2 为首地址的数据区中，试编写一程序完成检查这两个字符串是否相同，若相同时标志单元 FLAG 置 0，否则置－1。

【解答】 程序代码如下：

```
DSEG    SEGMENT
STR1    DB      'THIS IS A BOOK'
STR2    DB      'THIS IS A BOOK'
COUNT   EQU     $－STR2
FLAG    DB      ?
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS：CSEG,DS：DSEG,ES：DSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        MOV     EX,AX
        LEA     SI,STR1
```

```
        LEA     DI,STR2
        MOV     CX,COUNT
        CLD
        REPZ    CMPSB ;字符串比较,若没比完且对应
                       字符相等时,继续比较
        JNZ     UNEQU
        MOV     AL,0
        JMP     EXIT
UNEQU:  MOV     AL,0FFH
        RET
CSEG    ENDS
        END START
```

5. 设 BX 寄存器包含两个非压缩 BCD 数,请将非压缩 BCD 数据换成对应的压缩 BCD 数,存入 AL 寄存器。

【解答】 程序代码如下:

```
DSEG    SEGMENT
MIN     DB  17H,36H,0A4H
DSEG    ENDS
SSEG    SEGMENT STACK
        DB  200 DUP(?)
SSEG    ENDS
CSEG    SEGMENT
        ASSUME CS:SCEG,DS:DSEG,SS:SSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        AND     MIN,0FH
        MOV     CX,4
LOP:    SHR     MIN+1,1   ;第二个字符逻辑右移1位
```

```
        RCR     MIN+2,1   ;移入第三个字符高位
        LOOP    LOP   ;循环 4 次
        MOV     CX,4
        ROR     MIN+2,CL   ;将第三个字符高 4 位与
                           低 4 位交换
EXIT:   RET
CSEG    ENDS
        END     START
```

6. 使 AL 高位置 1,判断低 4 位是否大于 9,如大于 9,则使低 4 位变反,否则将低 4 位置成 9,试编程实现。

【解答】 程序代码如下:

```
DSEG    SEGMENT
A       DB      0AH
B       DB      ?
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS:CSEG,DS:DSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        MOV     AL,A
        OR      AL,0F0H   ;高 4 位置 1
        CMP     AL,0F9H   ;判低 4 位是否大于 9
        JA      A9   ;是,转 A9
        MOV     AL,0F9H   ;不是,将低 4 位置成 9
        JMP     NEXT
A9:     XOR     AL,0FH   ;使低 4 位变反
NEXT:   MOV     B,AL   ;结果存 B 单元
EXIT:   RET
```

```
CSEG    ENDS
        END     START
```

7. 从键盘输入一系列字符，以字符'$'为结束符，然后对其中的非数字字符计数，并显示出计数结果。

【解答】 源程序如下：

```
DSEG    SEGMENT
BUFF    DB      50 DUP (' ')
COUNT   DW      0
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS:CSEG,DS:DSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        LEA     BX,BUFF
        MOV     COUNT,0
INPUT:  MOV     AH,1
        INT     21H
        MOV     [BX],AL
        INC     BX
        CMP     AL,'$'
        JZ      DISP
        CMP     CL,30H
        JB      COUNT
        CMP     CL,39H
        JBE     NEXT
CONT:   INC     COUNT
        JMP     NEXT
DISP:   ……
```

8. 在字节数组中找出第一个非零元素，并显示输出第一个非零元素的下标。

【解答】 程序代码如下：

```
DSEG    SEGMENT
        ARRAY   DB  0,0,2,0,0,8,0,0
COUNT   EQU     $-OFFSET ARRAY
DSEG    ENDS
CSEG    SEGMENT
        ASSUME DS:DSEG,CS:CSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        MOV     CX,COUNT
        MOV     DI,0FFFFH       ;-1→DI
NEXT:   INC     DI
        CMP     ARRAY[DI],0
        LOOPZ   NEXT
        JNE     OKENTRY
        MOV     DL,'0'
        JMP     DISPLAY   ;显示输出一个零
OKENTRY: MOV    DX,DI     ;显示非零元素的下标(DI)
        OR      DL,30H
DISPLAY: MOV    AH,02H
        INT     21H
        RET
CSEG    ENDS
        END     START
```

9. 设在内存某一数据区以 STRING 地址开始存放了一字符串，其最后一个字符为“$”，ASCII 为 24H。要求检查该字符串中

所有字符的奇偶性，规定每个字符对应的一个字节数中必须有奇数个"1"则为正确。若奇偶性全部正确，则结果单元置为0，否则把结果单元置为FFH，程序结束。

【解答】 程序代码如下：

```
DSEG    SEGMENT
STRING DB       'SEND TO YOU $'
RESULTDB        ?
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS：CSEG,DS：DSEG
START：MOV      AX,DSEG
        MOV     DS,AX
        MOV     BX,OFFSET STRING
        MOV     DI,OFFSET RESULT
L1：    MOV     AL,[BX]  ;取一个字符
        INC     BX  ;修改字符地址
        CMP     AL,24H  ;比较是否为$结束标志
        JZ      DONE  ;若是转DONE
        OR      AL,AL  ;判断是否为奇校验
        JP      ERR
        JMP     L1
DONE：  MOV     AH,0  ;奇校验正确时置AH为0
        JMP     NEXT
ERR：   MOV     AH,0FFH  ;奇校验出错时置AH=
                         0FFH
NEXT：  MOV     [DI],AH  ;奇偶校验标志送RESULT
                         单元
        RET
```

```
CSEG    ENDS
        END     START
```

10. 计算矩阵：

$$\begin{bmatrix} A11 & A12 & A13 & A14 \\ A21 & A22 & A23 & A24 \\ A31 & A32 & A33 & A34 \\ A41 & A42 & A43 & A44 \end{bmatrix} \begin{bmatrix} B1 \\ B2 \\ B3 \\ B4 \end{bmatrix} = \begin{bmatrix} C1 \\ C2 \\ C3 \\ C4 \end{bmatrix}$$

【解答】 程序代码如下：

```
DSEG    SEGMENT
A       DB      A11,A12,…,A43,A44
B       DB      B1,B2,B3,B4
C       DW      4 DUP(?)
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS:CSEG,DS:DSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        MOV     SI,0
        MOV     BX,0
        MOV     CX,4
L0:     PUSH    CX
        MOV     DI,0
        MOV     WORD PRT C[BX],0
        MOV     CX,4
L1:     MOV     AH,0
        MOV     AL,A[SI]
        MOV     B[DI]
        ADD     C[BX],AX
```

```
        INC     SI
        INC     DI
        LOOP    L1
        ADD     BX,2
        POP     CX
        LOOP    L0
        RET
CSEG    ENDS
        END     START
```

11. 编写程序将某字节存储区中的10个非压缩的BCD数以相反的顺序送到另一个字节存储区中，并将这两个存储区中的数字串分两行显示出来。

【解答】 程序代码如下：

```
DSEG    SEGMENT
STR1    DB      01H,00H,09H,…,06H
COUNT   EQU     $-STR1
STR2    DB      COUNT DUP(?)
OBF1    DB      COUNT DUP(?),0DH,0AH
OBF2    DB      COUNT DUP(?),'$'
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS:CSEG,DS:DSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        MOV     CX,COUNT
        MOV     SI,0        ;指向已知存储区首
        MOV     DI,COUNT-1;指向已知存储区尾
AGAIN:  MOV     AL,STR1[SI]
```

```
        MOV     STR2[DI],AL
        ADD     AL,30H;将 AL 中的非压缩 BCD 数
                        转变为 ASCII
        MOV     OBF1[SI],AL     ;正向送 OBF1
        MOV     OBF2[DI],AL     ;反向送 OBF2
        INC     SI
        DEC     DI
        LOOP    AGAIN
        MOV     DX,OFFSET OBF1  ;输出 2 行字符
        MOV     AH,9
        INT     21H
        RET
CSEG    ENDS
        END     START
```

12. 试定义宏指令，要求把存储区中的一个用“$”字符结束的字符串，传送到另一个存储区中。

【解答】 程序代码如下：

```
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,24H
        JZ      EXIT
        INC     SI
        JMP     NEXT
```

```
EXIT:
        ENDM
```

13. 定义宏指令 PRINTB:利用 DOS 调用完成打印机连续打印一串字符的功能。如果字符串中出现制表符 TAB(ASCII 码为 09H),则打印 8 个空格符(ASCII 码为 20H)来代替它,字符串首地址及长度为变元。

【解答】 宏定义:

```
PRINTBK  MACRO  MESS,COUNT
LOCAL  NEXT,LOOP1
       MOV      CX,COUNT
       MOV      BX,0
NEXT:  MOV      SI,1
       MOV      DL,MESS[BX]
       CMP      DL,09H
       JNE      L1
       MOV      DL,20H
       MOV      SI,8
L1:    MOV      AH,5
       INT      21H
       DEC      SI
       JNE      L1
       INC      BX
       LOOP     NEXT
       ENDM
```

14. 编写宏定义 SHIFT,实现对 32 位数左移一位的功能,并写出利用此宏指令实现一个 32 位数乘 12 的宏定义。

【解答】 宏定义如下:

```
SHIFT   MACRO     NUM1,NUM2
```

```
        SAL     NUM2,1
        RCL     NUM1,1
        ENDM
MULT12  MACRO MR1,MR2,REG1,REG2
        PUSHF
        PUSH    REG1
        PUSH    REG2
        SHIFT   MR1,MR2
        SHIFT   MR1,MR2
        MOV     REG1,MR1
        MOV     REG2,MR2
        SHIFT   MR1,MR2
        ADD     MR2,REG2
        ADC     MR1,REG1
        POP     REG2
        POP     REG1
        POPF
        ENDM
```

若某数据段已定义变量：

```
DATA    DW  200H,080H
```

在程序中可以编写宏指令语句；

```
MULT12  DATA+2,DATA,AX,BX
```

15. 使用重复汇编 REPT 向数据段中 NUM 开始单元顺序存入 1,2,3,…,100,并显示前 5 个数据。

【解答】 程序代码如下：

```
DSEG    SEGMENT
        N=1
        NUM LABEL  BYTE
```

```
        REPT    100
        DB      N
        N=N+1
        ENDM
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS:CSEG,DS:DSEG
START   PROC    FAR
        MOV     AX,DSEG
        MOV     DS,AX
        MOV     SI,OFFSET NUM
        MOV     CX,5
NEXT:   MOV     DL,[SI]
        INC     SI
        OR      DL,30H
        MOV     AH,2
        INT     21H
        LOOP    NEXT
        RET
START   ENDP
CSEG    ENDS
        END     START
```

附录　课程考试模拟试题及答案

模拟试题 一

一、已知(AX)＝2000H,(BX)＝1200H,(SI)＝0002H,(DI)＝0003H,(DS)＝3000H,(SS)＝3000H,(SP)＝0000H,(31200H)＝50H,(31201H)＝02H,(31202H)＝0F7H,(31203H)＝90H。请写出下列各条指令独立执行完后有关寄存器及存储单元的内容,若该指令影响条件码,则请给出条件码 SF,ZF,OF,CF 的值。

```
1. ADD    AX,1200H
2. SUB    AX,BX
3. MOV    [BX],AX
4. PUSH   AX
5. DEC    BYTE    PTR  [1200H]
6. NEG    WORD    PTR  [1200H]
7. SAR    BYTE    PTR  1200[SI],1
8. ROL    BYTE    PTR  [BX+SI+1],1
9. MUL    WORD    PTR  [BX][SI]
10. DIV   BYTE    PTR  1200[DI]
```

二、指出下列指令的错误。

```
1. MOV  AH,BX
2. MOV  AX,[SI][DI]
3. MOV  BYTE  PTR[BX],1000
4. MOV  CS,AX
5. JNL  FAR  PTR
6. MOV  [BX],[SI]
```

7. MOV DISP[SI],ES：AX

8. MOV DS,SEG ARA

9. MOV DISP[DX+25][SI],AX

10. STOS

三、分析程序。

1. 阅读下列程序段，指出该程序段所完成的工作。

```
        DATX1   DB    300DUP(?)
        DATX2   DB    100DUP(?)
                ……
        MOV     CX,100
        MOV     BX,200
        MOV     SI,0
        MOV     DI,0
NEXT:   MOV     AL,DATX1[BX][SI]
        MOV     DATX2[DI],AL
        INC     SI
        INC     DI
        LOOP    NEXT
```

2. 已知数组 A 包含 15 个互不相等的整数，数组 B 组包含 20 个互不相等的整数，则下列程序实现的功能是什么？

```
DSEG    SEGMENT
A       DW  15 DUP(?)
B       DW  20 DUP(?)
C       DW  15 DUP(?)
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS:CSEG,DS:DSEG,ES:DSEG
START:  MOV     AX,DSEG
```

```
        MOV     DS,AX
        MOV     ES,AX
        MOV     SI,0
        MOV     BX,0
        MOV     CX,15
L1:     PUSH    CX
        MOV     DI,0
        MOV     CX,20
        MOV     AX,A[SI]
L2:     CMP     B[DI],AX
        JNE     NO
        MOV     C[BX],AX
NO:     ADD     BX,2
        ADD     DI,2
        LOOP    L2
        ADD     SI,2
        POP     CX
        LOOP    L1
        RET
CSEG    ENDS
        END     START
```

四、完善程序。

1. 下列程序段实现 1＋2＋3＋…＋100 的累加和。试在空白处填上适当的指令。

```
        (1)
        MOV     AX,1
        MOV     BX,2
LOP:    ADD     AX,BX
```

INC BX

(2)

2. 下列程序实现把 20 个字符"S"的字符串从原缓冲区传送到目的缓冲区的功能。试将程序中的空白处填上适当的指令。

```
DSEG        SEGMENT
SOUSTRING   DB  20 DUP('S')
DSEG        ENDS
ESEG        SEGMENT
DESTRING    DB 20 DUP(?)
ESEG        ENDS
CSEG        SEGMENT
            ASSUME CS: CSEG,DS:DSEG,ES:ESEG
START:      MOV AX,DSEG
            MOV DS,AX
            MOV AX,ESEG
            MOV ES,AX
              (1)
            LEA DI,DESTRING
            CLD
            MOV CX,20
              (2)
            RET
CSEG        ENDS
            END START
```

五、编程题。

1. 有一个 50 个数字的存储区，统计其为偶数和奇数的数字各为多少，分别存入 A 和 B 单元中。存储区的数据如下定义：

DATA DB −7,4,5,6,8,−4,1,0,9,2,−7,4,5,6,8,−4,1,

```
        0,9,2
    DB  -7,4,5,6,8,-4,1,0,9,2,-7,4,5,6,8,-4,1,
        0,9,2
    DB  -7,4,5,6,8
```

2. 设在 DATA 单元中存放一个－9～＋9 的字节数据，在 SRQTAB 数据区中存放 0～9 的平方值，请编写一个子程序，在 SRQTAB 中查出 DATA 单元中数据对应的平方值送 SRQ 单元，并写出主程序的调用方式。

3. 编写一条宏指令 CLRB，完成用空格符将一字符区中的字符清除的工作。字符区首地址及其长度为变元。

答　案

一、1. (AX)＝3200，SF＝0，ZF＝0，OF＝0，CF＝0

2. (AX)＝0E00H，SF＝0，ZF＝0，OF＝0，CF＝0

3. (31200H)＝2000H，不影响条件码

4. (3FFFEH)＝2000H，(SP)＝0FFFEH，不影响条件码

5. (31200H)＝4FH，SF＝0，ZF＝0，OF＝0

6. (31200H)＝0FDB0H，SF＝1，ZF＝0，OF＝0，CF＝1

7. (31202H)＝0FBH，SF＝1，ZF＝0，OF＝0，CF＝1

8. (31203H)＝21H，OF＝1，CF＝1

9. (DX)＝121EH，(AX)＝0E000H，OF＝1，CF＝1

10. (AL)＝38H，(AH)＝80H，条件码无定义。

二、1. 寄存器类型不匹配。

2. SI、DI 不能一起使用。

3. 1000 超出一个字节的范围。

4. CS 不能用作目的寄存器。

5. JNL 为条件转移指令。条件转移都为短转移。

6. 双操作数不能同时为存储器操作数。

7. 寄存器 AX 不能使用段前缀 ES。

8. SEG ARA 汇编后为立即数，不能直接传送给 DS。

9. 目的操作数不能使用 DX 作为基址寄存器。

10. 因为使用 STOS 指令时必须在操作数中表示是对字节串进行操作，还是对字串进行操作，而本指令中目的只写了[DI]未指明对字节还是对字操作。所以本指令有错。若对字节操作，可写成“STOS BYTE PTR [DI]”；若对字操作，可写成“STOS WORD PTR [DI]”。

三、1. 将 DATX1 的最后 100 个字复制到 DATX2。

2. 该程序的功能是寻找数组 A 与数组 B 中相同的数据，并将它们存在 C 开始的存储区中。

四、1. (1)MOV CX,63H

(2)LOOP LOP

2. (1)LEA SI,SOUSTRING

(2)REP MOVSB

五、

```
1. DSEG    SEGMENT
   DATA    DB      -7,4,5,6,8,-4,1,0,9,2,-7,4,
                   5,6,8,-4,1,0,9,2
           DB      -7,4,5,6,8,-4,1,0,9,2,-7,4,
                   5,6,8,-4,1,0,9,2
           DB      -7,4,5,6,8,-4,1,0,9,2
   N       EQU     50
   A       DB      0
   B       DB      0
   DSEG    ENDS
   SSEG    SEGMENT STACK
           DB  200DUP(?)
```

```
SSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS:CSEG,DS:DSEG,SS:SSEG
START:  MOV AX,DSEG
        MOV     DS,AX
        MOV     CX,N
        LEA     SI,DATA
L1:     MOV     AL,[SI] ;取数
        SHR     AL,1
        JC      ODD     ;是奇数,转 ODD
        INC     A       ;是偶数,A 单元值加 1
        JMP     NEXT
ODD:    INC     B       ;是奇数,B 单元值加 1
NEXT:   INC     SI      ;指向下一个数
        LOOP    L1      ;未统计完继续
EXIT:   RET
CSEG    ENDS
        END     START
```

2.

```
DSEG    SEGMENT
DATA    DB      6
SRQTAB  DB      0,1,4,9,16,25,35,49,64,81
SRQ     DB      ?
DSEG    ENDS
CSEG    SEGMENT
        ASSUME  CS:CSEG,DS:DSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        ……
```

```
        CALL    SUB
        ......
        RET
SUB     PROC
        PUSH    AX
        PUSH    BX
        LEA     BX,SRQTAB
        MOV     AL,DATA
        TEST    AL,80H
        JNS     NEXT
        NEG     AL
NEXT:   XTAL    SRQTAB
        MOV     SRQ,AL
        POP     BX
        POP     AX
        RET
SUBQ    ENDP
CSEG    ENDS
        END     START
```

3. CLRB

```
        MACRO   N,BUFF
        MOV     CX,N
        MOV     AL,' '
        LEA     DI,BUFF
        REP     STOSB
        ENDM
```

模拟试题二

一、指出下列指令的寻址方式：

```
1. MOV     CX,2008
2. MOV     AX,25[SI]
3. MOV     [DI+BX],AX
4. ADD     AX,ADDR
5. MUL     BL
6. INC     WORD  PTR[BX+2008]
7. SUB     AX,[BP+2008]
8. JMP     BX
9. IN      AL,20H
10. STI
```

二、写出执行以下指令序列后，标志寄存器各位的变化。

```
1. MOV     BX,93AFH
   ADD     BX,0E782H
2. MOV     AX,73A6H
   ADD     AX,8B2CH
```

三、分析程序题。

下列程序段的功能各是什么？

```
1.          MOV     CX,1AH
            MOV     DL,'z'
   L1:      PUSH    DX
            MOV     AH,02H
            INT     21H
            POP     DX
            DEC     DX
```

```
        LOOP    L1
```

2.
```
        XOR     DX,DX
L1:     MOV     AH,1
        INT     21H
        CMP     AL,'$'
        JZ      BACK
        CMP     AL,30H
        JB      L1
        CMP     AL,39H
        JA      L1
        AND     AL,0FH
        ADD     DL,AL
        ADC     DH,0
        JMP     L1
BACK:   HLT
```

四、完善程序题。

1. 以 DA 为起始地址的数组中存放的 N 个有符号数据，下面程序完成了找出 N 个数中的最大值和最小值分别送入 AH 和 AL 中。

```
        DA      DB      12,34,-24,6A,…98;N 个数据
        COUNT   EQU     $-DATA
        ……
        MOV     SI,ADR
        MOV     CX,COUNT
        MOV     BH,[SI]
        MOV     BL,BH
L1:     (1)
```

```
        CMP     AL,BH
        (2)
        MOV     BH,AL
        (3)
L2:     CMP     AL,BL
        JGE     L3
        MOV     BL,AL
L3:     DEC     CX
        (4)
        MOV     AX,BX
```

2. 下列程序段完成：2＋4＋2…＋20 共 10 个偶数的累加和。

```
DATA    SEGMENT
SUM     DW?
……
        XOR     AX,AX
        (1)
        MOV     BX,2
L1:     ADD     AX,BX
        INC     BX
        INC     BX
        (2)
        JNZ     LOP1
        MOV     SUM,AX
```

五、编程题。

1. 实现把键盘输入的小写字母用大写字母显示出来。

2. 假设在以 ARRAY 为首地址的存储区中存放 10 个带符号的字节数。试编制程序，统计出其中不大于 10 的元素的个数。

3. 编写子程序从键盘输入一个长度为 N 的字符串。

答　案

一、1. 立即数寻址(源)。

2. 变址寻址(源)。

3. 基址变址寻址(目的)。

4. AD DR 为符号常量时,是立即数寻址;为变量时,是直接寻址(源)。

5. 寄存器寻址。

6. 变址寻址。

7. 变址寻址。

8. 寄存器寻址。

9. I/O 端口寻址。

10. 固定寻址。

二、1. CF=1,ZF=0,OF=1,SF=0,PF=0,AF=1

2. CF=0,ZF=0,OF=0,SF=1,PF=1,AF=1

三、1. 执行后屏幕打印 z y x…c b a 共 26 个字母。

2. 循环输入 0～9 的数字,遇"$"字符结束,并对它们进行求和存入 DX。

四、1. (1)LODSB

(2)JLE　L2

(3)JMP　L3

(4)JNZ　L1

2. (1)MOV　CX,10

(2)DEC　CX

五、1.

```
START:  MOV     AH,1
        INT     21H
        CMP     AL,'a'
```

```
        JB      STOP
        CMP     AL,'z'
        JA      EXIT
        SUB     AL,02H
        MOV     DL,AL
        MOV     AH,2
        INT     21H
        JMP     START
EXIT:   RET
```

2.

```
DSEG    SEGMENT
ARRAY   DB      1,-1,2,-2,3,-3,4,-4,5,-5
NUM     DB      ?
ARRAY   DB      1,-2,2,-2,3,-3,4,-4,5,-5
NUM     DB      ?
DSEG    ENDS
CSEG    SEGMENT
        ASSUME CS:CSEG,DS:DSEG
START:  MOV     AX,DSEG
        MOV     DS,AX
        LEA     SI,ARRAY
        XOR     AL,AL
        MOV     CX,10
L1:     MOV     AL,[SI]
        CMP     AL,10
        JGE     L2
        INC     BL
L2:     INC     SI
```

```
        LOOP    L1
        MOV     NUM,BL
        RET
CSEG    ENDS
        END     START
```

3. 子程序如下：

```
GETSTRING  PROC  NEAR
L1:        MOV   AH,01H
           INT   21H
           MOV   [SI],AL
           INC   SI
           LOOP  L1
           RET
GETSTRING  ENDP
```