IBM—PC

汇编语言程序设计

例题习题集

温冬婵 沈美明 编著

清华大学出版社

IBM PC 汇编语言程序设计

例 题 习 题 集

温冬婵　沈美明　编

清华大学出版社

内 容 简 介

本书为《IBM PC [0520]汇编语言程序设计》、《IBM PC 汇编语言程序设计》二本书的教学参考书，共收集、编写了约 300 道复习练习题。按照由指令到程序设计的系统分为八章，每章包括复习提要、例题分析及习题三个部分，并给出部分答案。书后还附有二份自测题及其答案。本书在练习的形式和内容上，突出了基础知识的复习巩固，也注意了程序设计能力的培养和提高，既适用于大、中学生学习汇编语言，也适用于教师和工程技术人员作参考。

（京） 新登字 158 号

IBM PC 汇编语言程序设计例题习题集

温冬婵 沈美明 编
责任编辑 贾仲良
☆
清华大学出版社出版
北京 清华园
国防工业出版社印刷厂印刷
新华书店总店科技发行所发行
☆
开本：787×1092 1/16 印张：7.75 字数：198 千字
1991 年 6 月第 1 版 1992 年 11 月第 3 次印刷
印数：27001 — 42000
ISBN 7-302-00756-X/TP・254
定价：3.60 元

前　言

近年来，由于微型计算机的迅速发展和广泛应用，越来越多的技术人员在系统设计或技术开发中需要分析汇编语言程序或编制汇编语言程序。在国内外的中、高等院校中，汇编语言程序设计也是计算机专业学生必修的专业基础课程之一.为了帮助在校学生和自学汇编语言程序设计的工程技术人员掌握本课程的主要内容和学习重点，我们应广大读者的要求，配合清华大学出版社出版的《IBM PC [0520]汇编语言程序设计》一书，编写了这本习题集，供学习时使用和参考。

根据初学者的特点，本书按照由浅入深，由指令到程序设计的系统编写。每章包括复习提要、例题分析和习题三个部分。书后还附有自测题和部分习题的参考答案。书中有很多例题和习题可编写成完整的程序，在 IBM PC XT/AT 或 PC 兼容机上运行。如果练习编写的程序经过上机调试，则更有助于学习和掌握汇编语言程序设计的基本方法和技巧。

由于我们的水平有限，进行 IBM PC 汇编语言程序设计教学和实践的时间也不长，可能会出现错误和不妥之处，恳请广大读者批评指正。

编　者

1989 年于清华大学

目　　录

第一章　数和字符的表示法

复　习　提　要

1. 计算机只能识别以二进制形式存在的机器语言。计算机内部数的存储及运算也都是采用二进制。

2. 一个二进制数的值由 1 所在位置的权来确定。如$(1101)_2=2^3+2^2+2^0=(13)_{10}$

3. 二进制的负数用补码来表示。对一个二进制数求补(按位求反，末位再加 1)，即得到这个数相反数的补码表示。

4. 补码的加法和减法的规则是：

$$[X+Y]_{补}=[X]_{补}+[Y]_{补},\quad [X-Y]_{补}=[X]_{补}+[-Y]_{补}$$

5. 16 进制是一种很重要的短格式计数法，它把二进制数每 4 位分成一组，分别用 0～9 和 A～F 来表示 0000～1111。反之，16 进制数的每一位用四位二进制表示，就是相应的二进制数。

6. 十进制转换为二进制数的方法主要有降幂法和除法。计算机十化二程序中采用下面的算法：

$$N_mN_{m-1}\cdots N_1N_0=(\cdots((N_m+0)\times 10+N_{m-1})\times 10+\cdots+N_1)\times 10+N_0$$

7. 标志位 OF＝1 表示带符号数的运算结果无效。CF＝1 表示无符号数的运算结果无效。

8. 计算机中的字符数据用 ASCII 码表示，一个字符在存储器中占用一个字节(8 位二进制码)。

9. BCD 码是一种用二进制编码的十进制数，又称二-十进制数或 8421 码，它用 4 位二进制数表示一个十进制数码。BCD 码有压缩和非压缩两种格式。压缩的 BCD 码用 4 位二进制数表示一个十进制数位，如 95_{10}表示为 1001,0101。非压缩的 BCD 码用 8 位二进制数表示一个十进制数位。如 95_{10}表示为 00001001 00000101。

例　题　分　析

例 1.1　完成下列各式补码数的运算，并根据结果设置标志位 SF、ZF、CF 和 OF，指出运算结果有效否。

(1) 0100,1001b+1001,1101b

(2) 0100,0001b−1010,1011b

(3) 0A95Bh+8CA2h

(4) 6531h−42DAh

解：

(1)　0100,1001　　SF ZF CF OF

+1001,1101　　1　0　0　0　　运算结果有效

1110,0110

这两个二进制补码数相加，和为一个非零的负数(SF＝1，ZF＝0)；最高有效位无进位(CF＝0)；正负两数相加，结果不会溢出(OF＝0)。

(2) 0100,0001　　SF ZF CF OF

＋0101,0101　　1 0 1 1　结果无效

1001,0110

两个补码数相减，先对减数求补，(1010,1011 求补后得 0101,0101)，再把减法转化为加法进行运算。运算结果为非零负数(SF＝1，ZF＝0)；最高位无进位(CF＝1)；结果符号与减数相同(均为负)，则 OF＝1，结果是错误的。

(3) A95B　　SF ZF CF OF

＋8CA2　　0 0 1 1　结果无效

1 35FD

这两个十进制数相加，所得结果是一个非零的正数(SF＝0，ZF＝0)；最高位有进位(CF＝1)；两负数相加，结果为正数，故 OF＝1，结果无效。

(4) 6531　　SF ZF CF OF

＋BD26　　0 0 0 0　结果有效

1 2257

先对减数求补(42DA 求补后得 0BD26)，然后用加法进行运算。运算结果为非零正数(SF＝0，ZF＝0)；最高位有进位(CF＝0)；结果符号与减数不同(OF＝0)，结果正确。

例 1.2　把字符串“PART1:Memory”存放在 1100 开始的存储区中，请写出字符串的存储情况。

字　符：	P	A	R	T	1	:	M	e	m	o	r	y
ASCII 码：	50	41	52	54	31	3A	4D	65	6D	6F	72	79
地　址：	1100	1101	1102	1103	1104	1105	1106	1107	1108	1109	110A	110B

IBM PC 的存储器按字节编址，一个 ASCII 码字符占用一个字节。

例 1.3　写出十进制数 3590 的非压缩 BCD 码和压缩的 BCD 码，并分别把它们存入数据区 UNPAK 和 PAKED。

UNPAK＋0	0 0
＋1	0 9
＋2	0 5
＋3	0 3

PAKED＋0	9 0
＋1	3 5

3590 的非压缩 BCD 码为 0000 0011，0000 0101，0000 1001，0000 0000；压缩的 BCD 码为 0011 0101，1001，0000。它们以相反的顺序存储在数据区中。

习　题

1.1　用降幂法和除法将下列十进制数转换为二进制数和 16 进制数。

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

1.2 将下列二进制数转换为16进制数和十进制数。

(1) 101101 (2) 10000000 (3) 1111111111111111 (4) 11111111

1.3 将下列16进制数转换为二进制数和十进制数。

(1) FA (2) 5B (3) FFFE (4) 12D4

1.4 完成下列二进制数的加法：

(1) 00010101 (2) 00111110 (3) 11111111 (4) 00111001
+00001101 +00101001 +00000001 +01000111

1.5 完成下列16进制数的加法：

(1) 23A6 (2) 51FD (3) 7779 (4) EABE
+0076 +0036 +0887 +26C4

1.6 完成下列八位二进制数的减法：

(1) 00011111 (2) 00011001 (3) 10101010 (4) 00111000
−00000101 −00010000 −00010101 −11111111

1.7 完成下列十六进制数的减法：

(1) FFFF (2) 12AA
−AAAA −02AB

1.8 写出下列二进制数的补码表示：

(1) −00010011 (2) −00111100 (3) −00111001 (4) −01000000

1.9 写出下列补码数的相反数：

(1) 11001000 (2) 10111101 (3) 10000000 (4) 00000000

1.10 下列各数均为十进制数，请用8位二进制补码计算下列各题，并用16进制数表示其运算结果，同时说明进位值，并判断是否溢出。

(1) (−85)+76 (2) 85+(−76) (3) 85−76 (4) 85−(−76)

(5) (−85)−76 (6) (−85)−(−76)

1.11 把下列数和字符用16进制表示出来：

(1) 字母Q (2) ASCII码7 (3) 二进制数10111101

(4) 十进制数100 (5) 空格(space) (6) ？符

1.12 下列各数均为用16进制表示的8位二进制数，请说明当它们分别看作补码表示的数及字符的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

1.13 请写出下列字符串的ASCII码。

For example,

This is a number 3692.

1.14 分别用8位二进制和二位16进制数写出下列十进制数的补码表示：

(1) 16 (2) 100 (3) 127 (4) 0

(5) −16 (6) −100 (7) −128 (8) −1

1.15 16位的二进制补码数所能表示的十进制最大数和最小数分别是什么？

1.16　16位二进制数所能表示的无符号数的范围是多大?

1.17　假设两个二进制数A=01101010,B=10001100,试比较它们的大小。

(1) A、B两数均为带符号的补码数。

(2) A、B两数均为无符号数。

1.18　有一个16位的数值0101,0000,0100,0011,

(1) 如果它是一个二进制数,和它等值的十进制数是多少?

(2) 如果它们是ASCII码字符,则是些什么字符?

(3) 如果是压缩的BCD码,它表示的数是什么?

1.19　求出以下各16进制数与62A0H的和,并根据结果设置标志位SF、ZF、CF和OF:

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

1.20　求出以下各16进制数减去4AE0H的差值,并根据结果设置标志位SF、ZF、CF和OF:

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

1.21　从键盘输入一个十进制数4096存入缓冲区BUFFER,再将该数所对应的非压缩BCD码和压缩的BCD码分别存入UNPAK和PAKED数据项,请写出BUFFER、UNPAK和PAKED中各字节的内容。

1.22　变量INCOME用伪操作DW定义了一个5位的十进制数,请将此数以压缩和非压缩BCD码的格式,用伪操作DB定义在变量INCOME1和INCOME2中。

```
INCOME  DW  32767
```

第二章　IBM PC 计算机组织

复　习　提　要

1. PC XT/AT 的核心是 8088/80286 的微处理器，它能完成存取字或字节并对其进行处理等功能，也能对存储器进行管理和保护。

2. 两种类型的内部存储器是 ROM(只读存储器)和 RAM(随机存储器)。存储器按字节编址，存储器地址一般用 16 进制的无符号数表示。

3. 字数据在存储器中存放的顺序为高地址字节存放高 8 位，低地址字节存放低 8 位。

4. AX、BX、CX 和 DX 是通用寄存器，每个通用寄存器可作两个 8 位寄存器使用(如 AH 和 AL)。

5. 一个 20 位的物理地址可表示成段地址:偏移地址。计算存储器单元的物理地址，可将段地址乘以 10H，再加上偏移地址。

物理地址 = (段地址 × 10H) + 偏移地址

6. 一个程序可包括四个段:代码段包含可执行的指令；堆栈段包含一个后进先出的数据区，它可保存程序的返回地址，数据以及寄存器的内容；数据段是程序的数据区；附加段也是一个数据区，它通常和数据段定义在同一存储区。

7. 段寄存器 CS、SS、DS 和 ES 分别寄存代码段、堆栈段、数据段和附加段的段地址。

8. 指针寄存器 SP 和 BP，如无特殊说明，它们指示堆栈段内存储单元的地址。

9. 堆栈的最高地址叫做栈底，堆栈指示器 SP 总是指向栈顶。

10. 堆栈存取必须以字为单位。一个字存入堆栈时，SP 减 2，再把字数据存入 SP 所指示的字单元中。从堆栈中取出一个字时，将 SP 所指示的字单元中的数据取出，然后 SP 加 2。

11. 变址寄存器 SI 和 DI 一般指示数据段内单元的地址，有时也可作为数据寄存器用。

12. 16 位的标志寄存器包括 6 个状态标志(SF、ZF、PF、CF、AF、OF)和 3 个控制标志(DF、IF、TF)。CF、AF、SF、ZF 和 OF 反映了算术运算以及移位、循环、逻辑等操作的结果状态。

例　题　分　析

例 2.1　一个有 16 个字的数据区，它的起始地址为 70A0:DDF6，请写出这个数据区首末字单元的物理地址。

解：首地址为：

(70A0×10H)+0DDF6=7E7F6H

末地址为：

7E7F6H+(10H−1)×2=7E814H

物理地址是一个 20 位的数值，分别由 16 位的段地址和 16 位的偏移地址来表示。数据区的最后一个字的地址为：首地址+(字数−1)×2

例 2.2 假设堆栈段寄存器 SS 的内容为 2250，堆栈指示器 SP 的内容为 0140，如果在堆栈中存入 5 个数据，SS 和 SP 的内容各是什么？如果又从堆栈中取出 2 个数据，SS 和 SP 的内容又各是什么？

答：在堆栈中存入 5 个数据后，SP 的内容变为 0136(0140－2×5＝0136H)。如又取出 2 个数据，SP 的内容又成为 013A，(0136＋4＝013A)。SS 的内容不变。

往堆栈中存入一个数(只能以字为单位)时，(SP)←(SP)－2，然后把数存入 SP 指示的地址。从堆栈中取数据时，先将 SP 所指示的地址的内容取出，然后(SP)←(SP)＋2

习　题

2.1　一台微型计算机的字长为 16 位，如果采用字节编址，那么它可以访问的最大存储空间是多少字节？试用 16 进制数表示该机的地址范围。

2.2　IBM PC 机的 I/O 端口号通常是由 DX 寄存器来提供的，但有时也可以在指令中用一个字节来表示端口号。试问可以直接由指令指定的 I/O 端口数是多少？

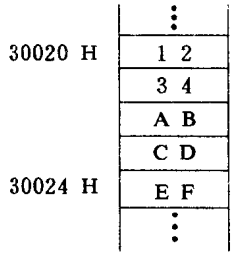
2.3　IBM PC 机有哪两种主要的存储器？它们所起的主要作用是什么？

2.4　有两个 16 位字 1EF5 和 2A3C 分别存放在 PC 机存储器的 000B0H 和 000B3H 单元中，请用图表示出它们在存储器里的存放情况。

2.5　将下列字符的 ASCII 码依次存入 00100 开始的字节单元中。(用图标出各单元的地址及其内容)。

IBM PC PERSONAL COMPUTER

2.6　在 PC 机的存储器中存放的信息如下图所示，试读出 30022，30024 字节单元的内容以及 30021、30022 字单元的内容。



2.7　写出下列存储器地址的段地址、偏移地址和物理地址。

(1) 2314：0035　　(2) 1FD0：000A

2.8　如果在一个程序段开始执行之前，(CS)＝0A7F0H，(IP)＝2B40H，试问该程序段的第一个字的物理地址是什么？

2.9　存储器中的每一段最多可含有 64K 个字节(1K＝1024)，假设用 DEBUG 命令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，以及状态标志 OF、SF、ZF、CF 的值。

```
A> debug
-r
AX=0000  BX=0000  CX=0080  DX=0000  SP=0000
```

BP=0000 SI=0000 DI=0000 DS=10E4 ES=10F4

SS=21F0 CS=31FF IP=0000 NV UP DI PL NZ NA PO NC

2.10 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) 表示运算结果的特征; (6) 将要执行的指令地址; (7) 将要从堆栈中取出数据的地址。

2.11 IBM PC 有哪些寄存器可用来指示存储器的地址?

2.12 如果一个堆栈从地址 1250 : 0000 开始,它的最后一个字的偏移地址为 0100H, SP 的内容为 0052H,

问:(1) 栈顶地址是什么?

(2) 栈底地址是什么?

(3) 在 SS 中的段地址是什么?

(4) 存入数据 3445H 后,SP 的内容是多少?

2.13 请将左边的词汇和右边的说明联系起来,括号内填入所选的 A,B,C…。

(1) CPU ( )
(2) 存储器 ( )
(3) EU ( )
(4) BIU ( )
(5) 堆栈 ( )
(6) IP ( )
(7) SP ( )
(8) 状态标志 ( )
(9) 控制标志 ( )
(10) 段寄存器 ( )
(11) 物理地址 ( )
(12) 汇编语言 ( )
(13) 机器语言 ( )
(14) 汇编程序 ( )
(15) 连接程序 ( )
(16) 目标码 ( )
(17) 指令 ( )
(18) 伪指令 ( )

A. 保存当前栈顶地址的寄存器。
B. 指示下一条要执行的指令的地址。
C. 总线接口部件,实现执行部件所需要的所有总线操作。
D. 分析并控制指令执行的部件。
E. 存储程序、数据等信息的记忆装置,PC 机有 RAM 和 ROM 两种。
F. 以后进先出方式工作的存储器空间。
G. 把汇编语言程序翻译成机器语言程序的系统程序。
H. 唯一代表存储器空间中的每个字节单元的地址。
I. 能被计算机直接识别的语言。
J. 用指令的助记符、符号地址、标号等符号书写程序的语言。
K. 把若干个模块连接起来成为可执行文件的系统程序。
L. 保存各逻辑段的起始地址的寄存器。PC 机有四个寄存器 CS、DS、SS、ES。
M. 控制操作的标志,PC 机有三位:DF、IF、TF。
N. 记录指令操作结果的标志,共六位:OF、SF、ZF、AF、PF、CF
O. 执行部件,由算逻单元(ALU)和寄存器组等组成。
P. 由汇编程序在汇编过程中执行的指令。
Q. 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。
R. 机器语言代码。

第三章　IBM PC 指令系统与寻址方式

复　习　提　要

1. 寻址方式小结

寻址方式	操作数地址(PA)	指令格式举例
立即寻址	操作数由指令给出	MOV DX,100H;(DX)←100
寄存器寻址	操作数在寄存器中	ADD AX,BX;(AX)←(AX)+(BX)

直接寻址	操作数的有效地址由指令直接给出	MOV AX,[100]
		MOV AX, VAR
		;(AX)←(100)或(VAR)

寄存器间接寻址	$PA=(DS)\times16+\begin{pmatrix}(BX)\\(DI)\end{pmatrix}$ $PA=(SS)\times16+(BP)$	MOV AX,[BX] ;(AX)←((DS)×16+(BX))
寄存器相对寻址	$PA=(DS)\times16+\begin{pmatrix}(BX)\\(DI)\end{pmatrix}$ +位移量 $PA=(SS)\times16+(BP)$+位移量	MOV AL,MESS[SI] ;(AL)←$\begin{pmatrix}(DS)\times16+(SI)\\+OFFSET\ MESS\end{pmatrix}$

基址变址寻址	$PA=(DS)\times16+(BX)+\begin{pmatrix}(SI)\\(DI)\end{pmatrix}$
	或$=(SS)\times16+(BP)+\begin{pmatrix}(SI)\\(DI)\end{pmatrix}$
		MOV AX,[BX][DI]
		MOV AX,[BX+DI]
		;(AX)←((DS)×16+(BX)+(DI))

相对基址变址寻址	$PA=(DS)\times16+(BX)+\begin{pmatrix}(SI)\\(DI)\end{pmatrix}$+位移量 或$=(SS)\times16+(BP)+\begin{pmatrix}(SI)\\(DI)\end{pmatrix}$+位移量	MOV AX,BUFF[BX][DI] ;(AX)←$\begin{pmatrix}(DS)\times16+(BX)+(DI)\\+OFFSET\ BUFF\end{pmatrix}$

2. 编写指令时,应注意的几个问题

(1) 注意区别立即寻址方式和直接寻址方式。

如：MOV AX,126;将数据 126 送入 AX 寄存器

MOV AX,[126];将数据段中的 126 单元的内容送 AX.

(2) 使用寄存器间接寻址时应注意和寄存器寻址方式的区别。

如：MOV AX,BX ;BX 中的内容传送到 AX

MOV AX,[BX];BX 所指示的地址中的内容送 AX

(3) 在双操作数指令中,源操作数和目的操作数的地址不能同时为存储器地址。

如：M1 和 M2 为两个存储器变量,

则 ADD M1,M2 是错误指令。

(4) 段跨越前缀可修改操作数所在的段。

如：MOV DL,MESS1[SI]；源操作数地址为：

；(DS)×16+(SI)+OFFSET MESS1

MOV DL，ES：MESS2[SI]；源操作数地址为：

；(ES)×16+(SI)+OFFSET MESS2

应注意：段跨越前缀不能使用 CS。

(5) 代码段寄存器 CS 不能用作指令的目的寄存器。

3. 正确使用指令系统，关键要清楚每条指令的功能以及它们规定或限制使用的寄存器。下面是初学者易混淆的几个问题：

(1) 指令对地址还是对地址中的内容进行操作，这要严格加以区分。

```
如：LEA BX,MESS              ；(BX)←MESS 的偏移地址
    MOV BX,OFFSET MESS       ；(BX)←MESS 的偏移地址
    MOV BX,MESS              ；(BX)←字变量 MESS 中的内容
```

(2) 使用指令时，要清楚指令隐含的操作寄存器。

如在乘法和除法指令中，只指出源操作数地址，但要清楚目的操作数必须存放在(AX)或(AL)中(乘法)，或(AX)、(DX：AX)中(除法)。又如串指令(MOVS、STOS、LODS、CMPS、SCAS)，它们的寻址方式也是隐含的，指令规定操作是在数据段中 SI 所指示的地址和附加段中 DI 所指示的地址之间进行串处理的；在存取串时，AL 是隐含的存取寄存器。

十进制调整指令(DAA、DAS、AAA、AAS、AAM、AAD)也隐含地使用了 AL 寄存器。

类似这些在指令语句中不反映出隐含操作数的指令还有换码指令 XLAT、循环指令 LOOP、LOOPE、LOOPNE 等，它们都要求预先在规定的寄存器内设置好操作数地址或计数值。

(3) 对带符号数和无符号数的操作应正确选择相应的条件转移指令。

(4) 用移位指令来倍增或倍减一个值是很方便的，但要注意对带符号数和无符号数所使用的指令应是不同的。

如：(AX)=8520H，当(AX)为无符号数时，(AX)/2 可用指令 SHR AX,1，结果是(AX)=4290H。

当(AX)为带符号数时，(AX)/2 应用指令 SAR AX,1，结果为(AX)=0C290H

(5) 标号是程序中指令的符号地址，要注意和变量(数据符号)的区别.

如定义 VAR 是一个变量，LAB 是程序中的一个标号，则 JMP LAB 指令的转移地址为 LAB，而 JMP VAR 是一条非法指令。

例 题 分 析

例 3.1 程序在数据段中定义的数组如下：

```
NAMES   DB      'TOM..'
        DB      20
        DB      'ROSE.'
        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]

解：(1) 两条指令都是合法指令。第一条指令取得 NAMES 的偏移地址，第二条 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，第三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，而源操作数据寄存器为 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。在汇编过程中，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。第四条指令中，AX 不能作为基址寄存器用。

(6) 均为合法指令。第三条指令中的 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。操作结果 (AL)=30。

例 3.2 两个数据段 DSEG1 和 DSEG2 的定义如下：

```
;-------------------------------------
DSEG1   SEGMENT  'DATA'     ;define data segment 1
L1      EQU      100         ;Length of STR1
STR1    DB       L1 DUP(?)
ASTR2   DB       5 DUP(?)    ;For storing STR2
DSEG1   ENDS
```

```
;-------------------------------------
DSEG2   SEGMENT  'DATA'     ;define data segment 2
L2      EQU      5          ;Length of STR2
STR2    DB       L2 DUP(?)
COUNT   EQU      L1-L2+1
DSEG2   ENDS
;-------------------------------------
```

图 3.01 （例 3.2）

请编写一段程序，在串 STR1 中查找子串 STR2(子串 STR2 的长度小于 STR1)，如果 STR1 中包含有子串 STR2，则程序结束；如果 STR1 中不包含子串 STR2，则将 STR2 加在 STR1 之后。

```
;-------------------------------------
CSEG    SEGMENT
MAIN    PROC     FAR
        ASSUME CS : CSEG, DS : DSEG2, ES : DSEG1
START:
        PUSH     DS                ;for return to DOS
        SUB      AX,AX
        PUSH     AX
        MOV      AX,DSEG2          ;put dseg2 in DS
        MOV      DS,AX
        MOV      AX,DSEG1          ;put dseg2 in ES
        MOV      ES,AX

        CLD                        ;set direction flag to forward
        MOV      CX,COUNT          ;put count in CX
        MOV      BX,OFFSET STR1
NEXT:   MOV      DI,BX             ;dest addr in DI
        MOV      SI,OFFSET STR2    ;source string addr
        PUSH     CX
        MOV      CX,L2
        REP      CMPSB             ;compare STR2 in STR1
        POP      CX
        JE       MATCH             ;STR1 contain STR2
        INC      BX                ;no match,point next addr
        LOOP     NEXT              ;compare again
NO_MATCH:
        LEA      DI,ASTR2          ;STR1 not contain STR2
        LEA      SI,STR2
        MOV      CX,L2
        REP      MOVSB             ;move STR2 at after STR1
MATCH:  RET                        ;return to DOS
MAIN    ENDP
```