WEIJIYUANLI

JIEKOUYUWANGLUOSHIYONGJISHU

微机原理、接口与网络实用技术

杨绍国　唐　芬　吴宗祥　唐　斌　编著

电子科技大学出版社

微机原理、接口与网络实用技术

杨绍国　唐　芬　吴宗祥　唐　斌　编著

电子科技大学出版社

内容简介

本书以PC系列微机为研究对象，系统地介绍了微机应用系统开发的软硬件接口、微机通信与组网等方面的实用技术。本书的特点是软件与硬件相结合，方法技术与应用实例相结合，以尽可能短的篇幅介绍尽可能多的实用技术，使广大读者在极短的时间内就能掌握微机应用系统设计、微机联网中的关键技术。本书附有大量应用实例，均通过了实践检验，读者可以按本书提供的思路自己设计微机应用系统，自己组建微机网络与通信系统。同时，本书提供了大量的实验，为读者提供了良好的实验环境。

全书共分六章。第一章系统地讨论了微机应用系统中汇编语言程序设计，程序调试及混合编程等实用技术；第二章讨论了微机系统软硬件接口技术；第三章详细介绍了用户接口扩展卡软硬件设计中的关键技术；第四章讨论了微机应用系统设计中的常用接口技术；第五章讨论了微机通信系统中的接口技术；第六章介绍怎样组建局域网。

本书内容充实，构思严谨，并附有大量的应用实例和实验，便于读者学习和借鉴。本书具有广泛的实用性，对于从事微型计算机应用系统开发、微机联网的广大工程技术人员来说，是一本很好的参考书；同时也可作为大专院校《微机原理》、《接口技术》、《计算机控制》、《计算机网络》、《汇编语言程序设计》等课程的教科书或参考书，是一本很好的学习指导书和实验教程。

微机原理、接口与网络实用技术

杨绍国　唐　芬　吴宗祥　唐　斌　编著

*

电子科技大学出版社出版

（成都建设北路二段四号）邮编 610054

电子科技大学出版社印刷厂印刷

新华书店经销

*

开本 787×1092　1/16　印张 14.5　字数 351.4 千字

版次　1996 年 9 月第一版　印次　1996 年 9 月第一次印刷

印数 1—4000 册

ISBN 7—81043—502—7/TP・191

定价：15.00 元

前　言

PC 系列微机是我国目前使用最为广泛的机种，已经渗透到社会各个领域并开始进入家庭。由于这种微机配置了性能优良的外围设备，如显示器、键盘、磁盘、异步串行通信口、打印机等，故可直接用于管理、科学计算、辅助设计等。但由于数据采集与处理、工业控制、网络通信以及多媒体系统等应用领域所需的各类设备接口，微机本身没有也无法提供，这就只能根据应用需要，开发出相应的微机应用系统。

汇编语言是和计算机硬件结合最密切的语言，是微机应用系统软件开发的主要语言，在第一章中详细讨论了汇编语言程序设计、程序调试和混合编程等实用技术。

PC 系列微机一般都配置了显示器、键盘、打印机、磁盘、声音、定时/计数、中断等接口。怎样根据需要来控制它们实现系统软硬件接口？关于这个问题将在第二章讨论。

虽然计算机系统为用户提供了一些常用的接口，但是微机应用领域广泛，用途各异，常常需要用户自行设计和开发满足特定需求的接口控制卡。第三章和第四章专门讨论微机用户接口扩展卡的软硬件设计问题。

随着微型计算机在各行各业的普遍应用，微机与微机之间以及微机与应用系统之间的信息交换便成为一个十分迫切而实际的问题，解决这个问题的方法在第五章中讨论。

80 年代以来，PC 技术及产品的迅速发展，推动了 PC 组网技术的飞速发展。目前，PC 组网已进入到各种应用领域。怎样组建局域网以便实现软硬件资源共享是一个需要解决的实际问题。本书的第六章专门介绍了局域网的组网技术，书中所列方法均通过了实践检验，读者可以直接移植。

本书在编写过程中，参考了许多书刊和文献资料，在此向提供帮助的各位作者表示感谢。

由于编者水平有限，错误及不当之处，在所难免，敬请广大读者批评指正。

编　者

一九九五年十二月于成都

目　录

第一章　汇编语言程序设计

第二章　系统软硬件接口技术

第三章 用户接口扩展软硬件设计

第四章 微机应用系统常用接口

第五章 微机通信接口技术

第六章 自己动手组建局域网

第一章　汇编语言程序设计

1.1　汇编语言程序的上机操作

1.1.1　汇编语言程序上机过程

汇编语言的上机过程如图 1.1 所示，现以后面讲到的例 1.1 为例分框说明如下：

一、用编辑程序建立和修改源程序(ASM 文件)

汇编源程序是文本文件，可以用任何文本文件的编辑软件来建立和修改汇编源程序。常用的有 EDLIN、WORDSTAR、WPS(N 命令)、NE 等编辑软件。例如用 EDLIN 编辑：

C>EDLIN EXAM11. ASM↵

二、用 ASM 或者 MASM 命令产生目标文件(OBJ 文件)

例如，对 EXAM11. ASM 汇编

C>MASM↵

此时，汇编程序给出如下回答：

```
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.
Source filename [.ASM]: EXAM11
Object filename [EXAM11.OBJ]:
Source listing [NUL.LST]: EXAM11
Cross-reference[NUL.CRF]: EXAM11
50628+272508 Bytes symbol space free
    0 Warning Errors
    0 Severe Errors
```

从上面的操作过程中可以见到，汇编程序的输入文件就是用户编写的汇编语言源程序，它必须以 ASM 为文件扩展名。汇编程序的输出文件有三个，第一个是目标文件，它以 OBJ 为扩展名，产生 OBJ 文件是我们进行汇编操作的主要目的，所以这个文件是一定要产生，也一定会产生的，操作时，这一步只要打入回车就行了；第二个是列表文件，它以 LST 为扩展名，列表文件同时给出源程序和机器语言程序，从而，可以使调试变得方便，列表文件是可有可无的，如果不需要，则在屏幕上出现提示信息[NUL. LST]:时，打入回车即可，如果需要，则打入文件名和回车；第三个是交叉符号表，此表给出了用户定义的所有符号，对每个符号都列出了将其定义的所在行号和引用的行号，并在定义行号上加上"#"号，同列表文件一样，交叉符号表也是为了便于调试而设置的，对于一些规模较大的程序，交叉符号表为调试工作带来很大方便，当然，交叉符号表也是可有可无的，如不需要，那么，在屏幕上出现提示信息

[NUL. CRF]:时，打入回车即可。

汇编过程结束时，会给出源程序中的警告性错误[Warning Errors]和严重错误[Severs Errors]，前者指出一般性错误，后者指出语法性错误，当存在这两类错误时，屏幕上除指出错误个数外，还给出错误信息代号，程序员可以通过查找手册弄清错误的性质。

如果汇编过程中，发现有错误，则程序员应该重新用编辑命令修改错误，再进行汇编，最终直到汇编正确通过。要指出的是汇编过程只能指出源程序中的语法错误，并不能指出算法错误和其他错误。

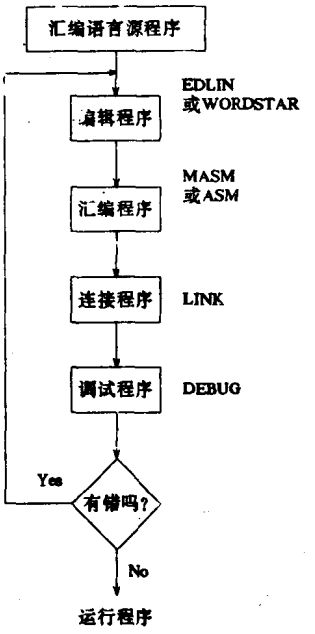

图 1.1 汇编语言过程流程图

三、用 LINK 命令产生执行文件(EXE 文件)

由汇编程序建立的目标码文件必须经过连接后，才能成为可执行文件。连接过程如下：

C>LINK ↙

此时屏幕上见到如下回答信息：

```
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.
Object Modules [.OBJ]:EXAM11
Run File [EXAM11.EXE]:
List File [NUL.MAP]:EXAM11
Libraries [.LIB]:
```

LINK 命令有一个输入文件，即 OBJ 文件，有时，用户程序用到库函数，此时，对于提示信息 Libraries[. LIB]，要输入库名。

LINK 过程产生两个输出文件，一个是扩展名为 EXE 的执行文件，产生此文件当然是 LINK 过程的主要目的；另一个是扩展名为 MAP 的列表分配文件，有人也称它为映像文件，它给出每个段在内存中的分配情况。MAP 文件是可有可无的。

四、用 DEBUG 调试程序

由 LINK 命令产生的可执行文件在 DOS 操作系统下就可以运行了。但是，绝大多数程序初次运行结果都是不对的，也就是说存在运算错误，运算错误相对于 MASM、LINK 程序发现的语法错误要难发现得多，它必须通过 DEBUG 程序来进行调试，才能发现和改正错误。

1.1.2 程序段前缀 PSP

汇编语言程序经过汇编、连接以后生成的 EXE 文件，可在 PC-DOS 支持下装入内存，并从程序中指定的地址开始运行。装入文件并设置启动地址由 PC-DOS 的 COMMAND. COM 文件的 EXEC 来完成的。COMMAND. COM 在装入 EXE 文件前，首先确定最低内存可用地址

作为被装入程序的可用内存起点；再在这个程序段内偏移地址 0000H 处构造一个 0100H 字节的程序段前缀 PSP(Program Segment Prefix)控制块。EXEC 在这个程序段的偏移量 0100H 处开始装载被调用的程序，并把控制交给 0100H 处的指令。

程序段前缀 PSP 实际上是一个程序控制块，PC-DOS 建立这个程序控制块，并且利用它来管理系统的进程。当一个程序获得控制时，PSP 的结构如表 1-1：

表 1-1 PSP 的结构

字段范围 (16 进制)	字段长度 (10 进制)	字段含义
0—1 字节	2	指令 INT20H
2—3 字节	2	可用内存空间(以 16 字节为单位)
4 字节	1	保留
5—9 字节	5	远调用指令(Call 功能调用入口)
A—D 字节	4	程序结束地址(INT22H 入口)
E—11 字节	4	Ctrl-Break 出口地址(INT23H 入口)
12—15 字节	4	标准错误出口地址(INT24H 入口)
16—2B 字节	22	保留
2C—2D 字节	2	传送环境的段地址
2E—31 字节	4	保留
32—4F 字节	30	保留给 DOS 专用
50—51 字节	2	DOS 功能调用 INT21H 指令
52 字节	1	远返回指令
53—5B 字节	9	保留
5C—6B 字节	16	格式化参数 1，为未打开的 FCB_1 使用
6C—7B 字节	16	格式化参数 2，为未打开的 FCB_2 使用
7C—7F	4	保留
80 字节	1	约定的磁盘传送地址区或非格式化参数长度
81—FF 字节	127	约定的磁盘传送地址区或格式化参数

0—1 字节存放一条 DOS 程序结束的指令，当被调入的用户程序段执行完后返回到此字段，则由 INT20H 指令将控制权送回给 DOS，这是用户程序回到 DOS 的一种方法。

2—3 字节中存放当前可用内存的总空间，以 16 字节为单位，例如 1000H 表示 64K 字节。

5—9 字节放一条长调用指令 Call。在 5 字节处存放 Call 指令码 9AH，6—9 字节存放 DOS 系统功能调用的入口(即 INT21H 的入口)。因此，在用户程序中只需发一个近调用到 PSP 的位移 5 处，并把调用的子程序号送 AH，即可进入该调用的子程序。如果 AH＝00 或 4CH，则进入“程序结束”子程序，而把控制权返回给 DOS，这是从用户程序回到 DOS 的又一种方法。

A—D,E—11,12—15等三个字段是进入用户程序段时,DOS将相应的三个中断入口地址存放在此(低字节放位移,高字节放段值),以便在用户程序中可以建立用户自己的结束程序地址、Ctrl-Break的出口以及出错处理程序。当然,这需要在用户程序中修改相应中断向量表,但不管怎样,当用户程序运行结束时,上述三个向量相应地用PSP在此三个字段所保存的值来加以恢复。

2C—2D字节包含传送环境块段地址,环境块包含许多ASCII字符串(许多以空格和零字节结尾的ASCII串)。DOS规定环境的全长要小于32K,其格式为"名字=参数"的ASCII字符串组成,且规定每个字符串的最后一个字节为00H;整个字串集也用00H结尾。环境由命令处理程序建立的。所以引入环境这个概念,也是为使用上的方便,比如在程序运行过程中,用户想随时能加入一些信息,就可以使用这种方式,通过有关DOS命令(比如SET)建立一个环境。

5C—6B,6C—7B字段存放两个未打开的文件控制块FCB_1,FCB_2(FCB的格式和功能下面将介绍),它们被解释为DOS外部命令或可执行的程序名之后的参数部分,其一般形式为:

文件名　参数1　参数2

其中,文件名的扩展名必须是.COM或.EXF。这样,当把参数1,2假定为文件名时,则PSP前缀块按FCB格式将两个文件名分别格式化成未打开的FCB,用户便可在调入的用户程序段内对FCB进行所需的操作。由于早期为和CP/M兼容,这种缺省的FCB很有用,但现在DOS必须给应用程序提供完善的路径支持,FCB型文件不支持多级目录结构,因而缺省的FCB便失去了它的作用。

80—FF字段有两个作用:一是用作系统约定的磁盘传送地址区DTA,另一是存放跟在上述文件名之后的全部参数字符,称作非格式化参数区。

所谓DTA,就是磁盘传输时系统约定的缓冲区,起始地址在80H处,其容量为128个字节。用户也可以通过功能调用1AH,在内存任何位置设定自己的DTA。在用户程序中凡使用磁盘I/O的子程序时,都应先设置好DTA,且一次只能有一个DTA是有效的。在用1AH子程序定义为另一个有效的DTA前,所有的磁盘操作都使用这个有效的DTA。

以上说明使我们对PSP的各段含义和作用有一个大致了解,但涉及到有关寄存器的值还没给出。这些寄存器的值随着本段程序的属性不同,而有不同的解释:

对使用扩展名.EXE的程序(图1.2)

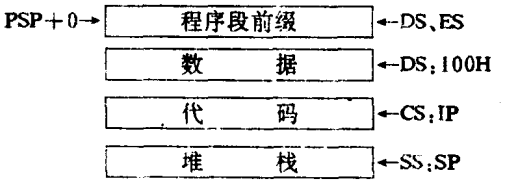

图1.2　EXE文件的执行

(1)DS、ES寄存器指向程序段前缀(用PSP+0表示)而不是指向用户程序的数据段和附加段,而数据段起始地址为DS:100H。

(2)CS、IP、SS和SP寄存器的值是由连接程序(Link)传递过来的。

对使用扩展名.COM的程序(图1.3)

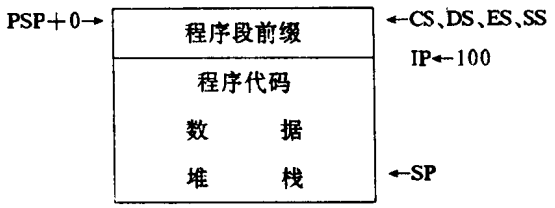

图 1.3　COM 文件的执行

(1)所有四个段寄存器都包含有程序前缀段的段地址，即 DOS 分配的程序段首地址(用 PSP+0 表示)这时内存全部用户区都分配给用户程序。

(2)指令指针 IP 被置为 100H，也就是说从程序段的第一条指令开始执行。

(3)在 PSP 的位移 6 处包含有程序段的可用字节数。

(4)SP 寄存器被置到程序段的末尾。

1.1.3　怎样用 DEBUG 调试程序

用 DEBUG 程序装入.EXE 的程序后，在程序尚未运行之前，各段寄存器的初值为：DS、ES 寄存器指向程序段前缀，因此，DS:100H 和 ES:100H 指向数据段。CS、SS 分别指向代码段和堆栈段。要想 DS 和 ES 分别指向用户数据段和附加段，用户程序必须重新设置，而 CS 和 SS 则不必。下面我们以例 1.1 的程序为例说明怎样用 DEBUG 调试程序。

一、怎样查看和修改寄存器的内容

为了了解程序运行的正确性，常常需要检查寄存器的内容，R 命令就是用来检查寄存器的内容，它有如下三种功能：

1. 显示和修改一个指定寄存器的内容，其格式为：

　　R〈寄存器〉

2. 显示所有寄存器的内容和全部标志位的状态，其格式为：

　　R

3. 显示和修改所有标志位的状态，其格式为：

　　RF

例如，

```
D>DEBUG EXAM11.EXE
-R
AX=0000  BX=0000  CX=002E  DX=0000  SP=0014  BP=0000  SI=0000  DI=0000
DS=13F2  ES=13F2  SS=1405  CS=1403  IP=0000   NV UP EI PL NZ NA PO NC
1403:0000  1E              PUSH    DS
-R  AX
AX  0000
:FFFF
-R
AX=FFFF  BX=0000  CX=002E  DX=0000  SP=0014  BP=0000  SI=0000  DI=0000
DS=13F2  ES=13F2  SS=1405  CS=1403  IP=0000   NV UP EI PL NZ NA PO NC
```

```
1403:0000  1E                PUSH    DS
-RF
NV UP EI PL NZ NA PO NC -OV DI CY
-RF
OV UP DI PL NZ NA PO CY -
-Q
```

在上例中显示了如下操作:(1)用 R 命令查看全部寄存器的内容和 CS:IP 所指的指令(PUSH DS)、指令机器码(1E)和所在存储单元的段地址和偏移地址(1403:0000);(2)用 R 命令把 AX 寄存器的内容由 0000H 改为 FFFFH,并查看改正后 AX 的值;(3)用 RF 命令修改了标志寄存器的值(NV 改为 OV,EI 改为 DI,NC 改为 CY)并查看修改后的结果。

二、怎样查看和修改代码段的内容

1. 怎样查看代码段的内容

在程序调试过程中,经常需要查看内存中装入的程序,这可用反汇编命令 U 完成,其格式为:

U〈起始地址〉L〈长度〉

若省略起始地址,则起始地址为上一次 U 命令反汇编过的最后一条指令后面的地址。若前面没有输入过 U 命令,则起始地址为 CS:IP。若段地址为 CS,则可以省略。若省略地址长度,则为 32 个字节。例如,

```
D>DEBUG EXAM11.EXE
-P=0  2
AX=0000  BX=0000  CX=002E  DX=0000  SP=0012  BP=0000  SI=0000  DI=0000
DS=13F2  ES=13F2  SS=1405  CS=1403  IP=0001   NV UP EI PL NZ NA PO NC
1403:0001  B80000            MOV     AX,0000

AX=0000  BX=0000  CX=002E  DX=0000  SP=0012  BP=0000  SI=0000  DI=0000
DS=13F2  ES=13F2  SS=1405  CS=1403  IP=0004   NV UP EI PL NZ NA PO NC
1403:0004  50                PUSH    AX
-U
1403:0004  50                PUSH    AX
1403:0005  B80214            MOV     AX,1402
1403:0008  8ED8              MOV     DS,AX
1403:000A  BB0000            MOV     BX,0000
1403:000D  8A07              MOV     AL,[BX]
1403:000F  43                INC     BX
1403:0010  0207              ADD     AL,[BX]
1403:0012  43                INC     BX
1403:0013  8A0F              MOV     CL,[BX]
1403:0015  43                INC     BX
1403:0016  020F              ADD     CL,[BX]
1403:0018  2AC1              SUB     AL,CL
1403:001A  43                INC     BX
```

```
1403:001B  8807          MOV     [BX],AL
1403:001D  CB            RETF
1403:001E  41            INC     CX
1403:001F  63            DB      63
1403:0020  7469          JZ      008B
1403:0022  7665          JBE     0089
-U  0008 L 4
1403:0008  8ED8          MOV     DS,AX
1403:000A  BB0000        MOV     BX,0000
-U
1403:000D  8A07          MOV     AL,[BX]
1403:000F  43            INC     BX
1403:0010  0207          ADD     AL,[BX]
1403:0012  43            INC     BX
1403:0013  8A0F          MOV     CL,[BX]
1403:0015  43            INC     BX
1403:0016  020F          ADD     CL,[BX]
1403:0018  2AC1          SUB     AL,CL
1403:001A  43            INC     BX
1403:001B  8807          MOV     [BX],AL
1403:001D  CB            RETF
1403:001E  41            INC     CX
1403:001F  63            DB      63
1403:0020  7469          JZ      008B
1403:0022  7665          JBE     0089
1403:0024  20636F        AND     [BP+DI+6F],AH
1403:0027  64            DB      64
1403:0028  0000          ADD     [BX+SI],AL
1403:002A  0000          ADD     [BX+SI],AL
1403:002C  0400          ADD     AL,00
```

关于上例，作如下几点说明：(1)指令 RETF 之前的内容才是本程序的指令，RETF 之后的内容是内存中其他程序的指令；(2)由 U 命令只能看到指令性语句，对于指示性语句，是不形成机器代码的，因此用 U 命令是看不到的；(3)源程序中的标号和名字，在此均换成了它们所代表的地址。

2. 怎样修改代码段的内容

在程序调试过程中，若发现某些指令有错，这时需要修改指令，可用汇编命令 A 来完成，其格式如下：

A〈地址〉

若命令中没有指定地址，则为上一个 A 命令的最后一个单元之后的地址；若前面没有用过 A 命令，则为 CS:0000H。若段地址为 CS，可以省略。例如，要把指令 SUB AL,CL 改为 ADD AL,CL，该指令存放的开始地址为 DS:0018H，则可按如下步骤完成：

```
-A  0018
```

```
1403:0018  ADD   AL,CL
1403:001A
-U0
1403:0000  1E              PUSH    DS
1403:0001  B80000          MOV     AX,0000
1403:0004  50              PUSH    AX
1403:0005  B80214          MOV     AX,1402
1403:0008  8ED8            MOV     DS,AX
1403:000A  BB0000          MOV     BX,0000
1403:000D  8A07            MOV     AL,[BX]
1403:000F  43              INC     BX
1403:0010  0207            ADD     AL,[BX]
1403:0012  43              INC     BX
1403:0013  8A0F            MOV     CL,[BX]
1403:0015  43              INC     BX
1403:0016  020F            ADD     CL,[BX]
1403:0018  00C8            ADD     AL,CL
1403:001A  43              INC     BX
1403:001B  8807            MOV     [BX],AL
1403:001D  CB              RETF
1403:001E  41              INC     CX
1403:001F  63              DB      63
```

三、怎样查看和修改数据段的内容

程序所需的数据和最终运算结果通常是保存在数据段内，故常需要查看和修改数据段的内容。

1. 怎样查看数据段的内容

查看数据段的内容，可用 D 命令来完成，其格式如下

D〈起始地址〉L〈长度〉

若起始地址省略，则为上一个 D 命令显示最后一个单元后面的地址，如果以前没有用过 D 命令，则约定地址为 DS:100H。D 命令约定段地址为 DS。

若长度省略，则显 80H 个字节。

在 DEBUG 装入.EXE 程序之后，在数据段寄存器重新定向之前，DS:100H 指向数据段，故查看数据段的内容可用如下命令：

D DS:100 L〈长度〉

或 D 100 L〈长度〉

例如：

```
D>DEBUG EXAM11.EXE
  -D DS:100 L 20
  13F2:0100  13 27 11 12 00 00 00 00-00 00 00 00 00 00 00 00    .'..............
  13F2:0110  1E B8 00 00 50 B8 02 14-8E D8 BB 00 00 8A 07 43    ....P..........C
```

```
-D 100 L 20
13F2:0100  13 27 11 12 00 00 00 00-00 00  00 00 00 00 00 00     .'..............
13F2 0110  1E B8 00 00 50 B8 02 14-8E D8 BB 00 00 8A 07 43     ....P..........C
```

若用 G 命令运行程序，则仍可用上述方法来查看存放在数据段的运行结果。例如：

```
D>DEBUG EXAM11.EXE
-G=0

Program terminated normally
-D DS:100 L 20
13F2:0100  13 27 11 12 17 00 00 00-00 00  00 00 00 00 00 00     .'..............
13F2 0110  1E B8 00 00 50 B8 02 14-8E D8 BB 00 00 8A 07 43     ....P..........C
```

从结果单元可见，其值为 17H，运行结果正确。

在程序运行之后，若 DS 重新定义，指向用户数据段，则查看数据段内容可用如下命令：

D DS:0000 L〈长度〉

或 D 0 L〈长度〉

上述命令也用于查看存放在数据段内由单步运行方式所得到的运行结果。例如：

```
-T=0 5

AX=0000  BX=0000  CX=002E  DX=0000  SP=0012  BP=0000  SI=0000  DI=0000
DS=13F2  ES=13F2  SS=1405  CS=1403  IP=0001   NV UP EI PL NZ NA PO NC
1403:0001  B80000            MOV     AX,0000

AX=0000  BX=0000  CX=002E  DX=0000  SP=0012  BP=0000  SI=0000  DI=0000
DS=13F2  ES=13F2  SS=1405  CS=1403  IP=0004   NV UP EI PL NZ NA PO NC
1403:0004  50                PUSH    AX

AX=0000  BX=0000  CX=002E  DX=0000  SP=0010  BP=0000  SI=0000  DI=0000
DS=13F2  ES=13F2  SS=1405  CS=1403  IP=0005   NV UP EI PL NZ NA PO NC
1403:0005  B80214            MOV     AX,1402

AX=1402  BX=0000  CX=002E  DX=0000  SP=0010  BP=0000  SI=0000  DI=0000
DS=13F2  ES=13F2  SS=1405  CS=1403  IP=0008   NV UP EI PL NZ NA PO NC
1403:0008  8ED8              MOV     DX,AX

AX=1402  BX=0000  CX=002F  DX=0000  SP=0010  BP=0000  SI=0000  DI=0000
DS=1402  ES=13F2  SS=1405  CS=1403  IP=000A   NV UP EI PL NZ NA PO NC
1403:000A  BB0000            MOV     BX,0000
-D  0  L  20
1402:0000  13 27 11 12 00 00 00 00-00 00  00 00 00 00 00 00     .'..............
1402 0010  1E B8 00 00 50 B8 02 14-8E D8 BB 00 00 8A 07 43     ....P..........C
-P  A
```