普通高等教育“十三五”规划教材

汇编语言程序设计

周　明 ◎ 编著

科学出版社

普通高等教育“十三五”规划教材

汇编语言程序设计

周 明 编著

科学出版社
北 京

内 容 简 介

本书面向应用型人才培养，以突破传统的组织结构为创新点，以全程伴随上机训练为特色，以通俗易懂的语言讲解了汇编语言程序设计的相关知识。

本书内容包括汇编语言基本概念；8086 CPU 的逻辑结构和 CPU 对存储器的读写过程；8086 的寻址方式和指令系统；汇编语言编程技巧，包括堆栈、端口、中断及子程序；DOS 系统功能调用和 BIOS 中断调用；32 位汇编语言的相关基础知识和编程技巧。

本书可作为普通高等院校计算机及相关专业汇编语言课程的教材，也可作为非计算机专业本科生的通识教材。

图书在版编目(CIP)数据

汇编语言程序设计/周明编著. —北京：科学出版社，2016

（普通高等教育“十三五”规划教材）

ISBN 978-7-03-050705-1

Ⅰ.①汇… Ⅱ.①周… Ⅲ.①汇编语言-程序设计-高等学校-教材 Ⅳ.①TP313

中国版本图书馆 CIP 数据核字（2016）第 278346 号

责任编辑：戴 薇 王 惠 / 责任校对：刘玉靖

责任印制：吕春珉 / 封面设计：东方人华平面设计部

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

三河市良远印务有限公司印刷

科学出版社发行 各地新华书店经销

*

2016 年 12 月第 一 版 开本：787×1092 1/16

2016 年 12 月第一次印刷 印张：14 3/4

字数：335 000

定价：33.00 元

（如有印装质量问题，我社负责调换〈良远印务〉）

销售部电话 010-62136230 编辑部电话 010-62135397-2052

前 言

随着计算机科学技术的日新月异，64 位计算机对读者来说已不再陌生。那么，如何跟上信息时代的脚步，学好、用好汇编语言？这是人们一直在思考和讨论的问题。

汇编语言与其他高级语言不同，由于它的操作指令直接面向硬件，是最接近计算机运行机制的编程语言。使用汇编语言编程时，编程者能够深刻感知计算机的运行过程和原理，对计算机硬件和应用程序之间的联系产生清晰的认识，从而在头脑中形成一个软硬兼备的知识体系。汇编语言编程能够锻炼编程者充分利用硬件资源的能力，这是任何高级语言无法替代的。汇编语言指令集合简约，指令执行效率高，对于计算机科技领域的人才培养来说，它的重要性毋庸置疑。

本书内容共分三大部分：第一部分（第 1～4 章）主要讲解 16 位汇编语言的基础知识，包括汇编语言的基本概念，8086 CPU 编程结构、存储系统、寻址方式、指令系统；第二部分（第 5～7 章）主要讲解汇编语言的程序设计、子程序设计、DOS 系统功能调用和 BIOS 中断调用；第三部分（第 8、9 章）主要讲解 32 位汇编语言的基础知识和编程技巧。书中列举的所有实例都在编译器下测试运行通过，每章均附有习题和上机训练内容，以便读者及时巩固所学知识。在本书的编写过程中，充分考虑到读者在学习过程中经常遇到的难点问题，紧密结合计算机硬件工作原理来组织内容，让读者对抽象的汇编语言不再感到迷惑不解。

本书面向应用型人才的培养，内容安排由浅入深。读者可借助各章后的上机训练，实现理论和实践的互补学习，有效提高汇编语言的编程能力。学好汇编语言，能使编程者更加深入地理解、运用 C 语言，同时也为学习单片机、ARM 嵌入式等应用技术打下坚实的基础。

吉林师范大学计算机学院王海燕、侯锟两位老师对第 3～6 章内容进行了材料搜集和加工整理，在此表示衷心的谢意。全书统稿工作由吉林师范大学计算机学院周明完成。

在本书的编写过程中，编者参考了许多汇编语言书籍、文献和网站的相关内容，在此对这些资料的作者表示衷心的感谢。

由于本书内容涉及的知识点多、范围广，加之编写时间仓促，难免存在不妥之处，敬请广大读者批评指正，并提出宝贵的改进意见和建议。

编 者

2016 年 10 月

目 录

第 1 章 汇编语言基础 …… 1

1.1 位、字节、字及字长的概念 …… 1
1.2 机器语言 …… 1
1.3 汇编语言的产生、发展及特点 …… 2
1.4 Intel 系列 CPU 简介 …… 4
习题 1 …… 5
上机训练 1 调试工具 Debug 的常用命令 …… 6

第 2 章 8086 CPU 和存储器 …… 10

2.1 冯·诺依曼型计算机结构 …… 10
2.1.1 冯·诺依曼型计算机的基本结构 …… 10
2.1.2 三总线结构 …… 11
2.2 8086 CPU 的逻辑结构 …… 12
2.2.1 引脚及芯片 …… 12
2.2.2 逻辑结构 …… 14
2.3 8086 CPU 内部的寄存器 …… 15
2.3.1 通用寄存器 …… 15
2.3.2 段寄存器 …… 17
2.3.3 控制寄存器 …… 17
2.3.4 寄存器的常规使用方法简介 …… 19
2.4 存储器 …… 20
2.4.1 存储器的分类 …… 20
2.4.2 存储单元 …… 20
2.4.3 存储单元的内容与地址 …… 21
2.4.4 8086 微机存储系统的地址空间分配 …… 22
2.5 8086 CPU 物理地址的形成 …… 23
2.5.1 段的概念 …… 23
2.5.2 与地址相关的基本概念 …… 24

2.5.3 8086 CPU 物理地址形成机制 ····· 26
2.6 8086 CPU 对存储器的读/写操作 ····· 26
2.7 8086 CPU 如何完成内存字的读/写 ····· 28
习题 2 ····· 29
上机训练 2 用 Debug 实现简单程序段的调试 ····· 30
第 3 章 8086 系统的寻址方式 ····· 32
3.1 寻址方式的概念 ····· 32
3.2 寻址方式的分类 ····· 32
3.2.1 立即寻址 ····· 33
3.2.2 寄存器寻址 ····· 34
3.2.3 直接寻址 ····· 35
3.2.4 寄存器间接寻址 ····· 36
3.2.5 寄存器相对寻址 ····· 37
3.2.6 基址加变址寻址 ····· 38
3.2.7 相对基址加变址寻址 ····· 39
3.3 寻找转移地址的寻址方式 ····· 40
3.3.1 段内直接寻址 ····· 40
3.3.2 段内间接寻址 ····· 41
3.3.3 段间直接寻址 ····· 42
3.3.4 段间间接寻址 ····· 43
习题 3 ····· 43
上机训练 3 掌握 Debug 下各种寻址方式的使用方法 ····· 45
第 4 章 8086 指令系统 ····· 46
4.1 数据传送指令 ····· 46
4.1.1 通用数据传送指令 ····· 46
4.1.2 地址传送指令 ····· 51
4.1.3 标志传送指令 ····· 52
4.1.4 查表指令 ····· 53
4.2 算术运算指令 ····· 53
4.2.1 加法指令 ····· 54
4.2.2 减法指令 ····· 55
4.2.3 乘法指令 ····· 56
4.2.4 除法指令 ····· 57

4.3 十进制调整指令 …… 58
4.4 逻辑运算指令 …… 60
4.5 移位指令 …… 63
4.5.1 逻辑移位指令 …… 63
4.5.2 算术移位指令 …… 64
4.5.3 循环移位指令 …… 65
4.6 标志位操作指令 …… 66
4.7 字符串操作指令 …… 67
4.8 控制转移指令 …… 70
4.8.1 无条件转移指令 …… 70
4.8.2 条件转移指令 …… 72
4.9 常用伪指令 …… 75
习题 4 …… 77
上机训练 4 在 Debug 下运行程序段 …… 79
第 5 章 汇编语言程序设计 …… 80
5.1 汇编语言源程序的基本框架 …… 80
5.1.1 段的定义 …… 80
5.1.2 ASSUME 伪指令 …… 81
5.1.3 段寄存器的装入 …… 82
5.2 汇编语言中的基本数据 …… 84
5.2.1 标识符 …… 84
5.2.2 常量、变量和标号 …… 84
5.2.3 运算符与表达式 …… 85
5.3 基本结构程序设计 …… 87
5.3.1 顺序结构程序设计 …… 88
5.3.2 分支结构程序设计 …… 92
5.3.3 循环结构程序设计 …… 102
5.4 数据块的传送 …… 111
5.5 段超越前缀 …… 114
5.6 堆栈操作程序 …… 116
5.6.1 堆栈的基本概念 …… 116
5.6.2 堆栈操作程序举例 …… 120
5.7 端口操作程序 …… 124
5.7.1 端口的概念 …… 124
5.7.2 输入/输出指令 …… 126

5.7.3 端口操作编程 …… 127
5.8 用户中断服务程序 …… 129
5.8.1 关于中断的相关概念 …… 129
5.8.2 中断处理过程 …… 132
5.8.3 用户中断服务程序的编写 …… 134
5.9 可执行文件与 PSP …… 137
5.9.1 .exe 可执行程序与 PSP …… 137
5.9.2 .com 可执行程序与 PSP …… 140
习题 5 …… 141
上机训练 5 对源程序进行汇编、连接与调试 …… 144
第 6 章 子程序设计 …… 145
6.1 子程序的定义与应用条件 …… 145
6.1.1 子程序的定义 …… 145
6.1.2 子程序的应用条件 …… 145
6.2 子程序的调用和返回指令 …… 146
6.2.1 子程序的调用指令 …… 146
6.2.2 子程序的返回指令 …… 147
6.3 子程序的结构 …… 147
6.4 子程序的参数传递 …… 150
6.4.1 寄存器传递参数 …… 151
6.4.2 存储器传递参数 …… 153
6.4.3 堆栈传递参数 …… 154
6.5 子程序的嵌套与递归调用 …… 161
6.5.1 子程序的嵌套调用 …… 161
6.5.2 子程序的递归调用 …… 163
6.6 模块化程序设计 …… 164
习题 6 …… 166
上机训练 6 子程序的编写、编译及调试 …… 167
第 7 章 DOS 系统功能调用和 BIOS 中断调用 …… 168
7.1 DOS 系统功能调用说明 …… 168
7.2 DOS 系统功能调用方法 …… 169
7.3 BIOS 中断调用说明 …… 173
7.4 BIOS 中断调用举例 …… 174
7.4.1 INT 10H 中断调用举例 …… 174

7.4.2　BIOS 其他类型中断调用举例 …… 177
习题 7 …… 179
上机训练 7　使用 BIOS 中断调用实现屏幕控制输出 …… 179

第 8 章　80386 汇编语言程序设计基础 …… 180

8.1　80386 CPU 的逻辑结构及引脚 …… 180
8.2　80386 CPU 中的寄存器 …… 182
8.3　80386 系统的寻址方式 …… 187
8.3.1　寻址方式 …… 188
8.3.2　实模式下编程 …… 189
8.4　80386 新增指令 …… 190
8.5　保护模式概述 …… 193
8.6　80386 保护模式下物理地址形成机制 …… 194
8.6.1　选择子与描述符 …… 195
8.6.2　线性地址的形成 …… 197
8.6.3　物理地址的形成 …… 198
8.7　中断和异常处理 …… 200
习题 8 …… 203
上机训练 8　建立 Windows 环境下 32 位汇编语言的集成开发环境 …… 206

第 9 章　80386 保护模式下的程序设计 …… 209

9.1　一个简单的编程实例 …… 209
9.2　Win32 API 概述 …… 210
9.3　常用简化段定义伪指令 …… 211
9.4　Win32 汇编语言程序结构 …… 213
9.5　结果输出程序举例 …… 214
9.6　控制台输出 …… 216
9.7　控制台输入 …… 219
习题 9 …… 222
上机训练 9　利用 MASM32 集成开发工具编写 32 位汇编语言程序 …… 223

参考文献 …… 224

第 1 章　汇编语言基础

汇编语言程序的执行直接面向计算机硬件，通过学习、使用汇编语言编程，能够深刻感知指令执行机制和程序运行的具体过程，从而对计算机硬件和程序之间的交互作用形成一个清晰的认识。

本章主要阐述汇编语言的产生、发展历程，使读者能够体会到学习汇编语言程序设计的重要性。

1.1　位、字节、字及字长的概念

位（bit）是计算机存储和处理信息的最基本单位。位值用二进制数码 1 或 0 表示，它是以电路元器件所记忆电平的高（1）低（0）来区分的。

字节（byte）是 CPU 读写内存储器或外设的基本单位，由连续的 8 个二进制位组成。

在计算机中，一串与寄存器宽度一致的数码常作为一个整体来进行传送或处理，这串数码称为计算机的一个字，简称字（word）。字通常由若干个字节组成，例如，16 位 8086 计算机的一个字含有 2 个字节。在计算机寄存器、运算器和控制器之间，通常以字为单位进行信息传送。

字所包含的二进制位数称为字长。16 位、32 位和 64 位计算机的字长分别为 16 位、32 位和 64 位。

1.2　机 器 语 言

学习汇编语言，首先要了解机器语言。机器语言是特定 CPU 所能执行机器指令的集合，而机器指令是指计算机能够直接识别、执行的二进制命令。

机器指令由“1”和“0”两种数码组成，能够被计算机 CPU 从存储器中读取并加以分析、执行。其中，数字“1”“0”分别是物理高、低电平的逻辑表征。

来看两条 Intel 8086 CPU 的机器指令：

```
10110000 00000100
10111000 00000011 00000010
```

这两条指令都是由“1”和“0”组成，但指令的字节长度却不一致，第一条指令是 2 字节指令，第二条指令是 3 字节指令。这种形式的指令，读、写与记忆都很不方便。

机器语言是唯一能被计算机 CPU 直接识别与执行的语言，与其他高级语言相比，

机器语言具有直接执行、执行效率高等优点。

用机器语言编写程序，编程人员首先要熟记所用计算机的二进制指令代码，并要求处理每条指令和每个数据的存储分配和输入/输出，还要记忆所用工作单元所处的状态，这是一件十分烦琐的工作，出错在所难免。

1.3 汇编语言的产生、发展及特点

用机器语言编写程序的过程过于复杂，所编程序的读写也很困难，于是汇编语言应运而生。最早的汇编语言应用可追溯到 20 世纪 50 年代初。

汇编语言是一种介于高级语言与机器语言之间，直接以处理器指令系统为编程基础的低级语言。汇编语言采用比较容易识别、记忆的英文助记符来表示指令的操作码部分，采用标识符来表示指令的操作数部分。

1.2 节中的两条机器指令所对应的汇编语言指令分别为

```
MOV AL,04H
MOV AX,0203H
```

对于 MOV AL,04H 指令来说，MOV 是指令助记符，代表指令的操作码部分，作用是传送数据；AL 是目的操作数，它是一个寄存器的名称，代表数据传送的目的地；04H 是源操作数，是用十六进制表示的立即数，代表传送数据的来源。

很显然，机器指令转变成汇编语言指令就很容易读写与记忆了。

采用汇编指令编写的汇编语言源程序，必须先经过汇编器（汇编程序）生成目标代码，再经过连接器（连接程序）生成可执行代码后，方可在操作系统下运行。图 1.1 描述了采用汇编语言编程的具体过程。

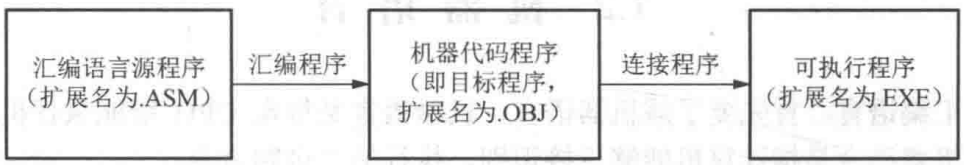

图 1.1 汇编语言编程过程

汇编程序：把汇编语言源程序翻译成机器代码的程序。在 DOS 操作系统平台下，汇编程序可采用汇编 ASM.EXE、宏汇编 MASM.EXE 及 TASM 等。一般使用宏汇编 MASM.EXE 来汇编源程序，因为它比 ASM.EXE 功能更强大，能够实现宏的汇编。TASM 适用于采用 Intel 8086 至 Pentium 系列指令系统所编写源程序的汇编，也是比较先进的汇编工具。

连接程序：将机器代码程序连接成可执行文件的程序。对应汇编程序，连接程序有 LINK.EXE 和 TLINK 等。

1. 汇编语言的特点

汇编语言的一个重要特点，就是它所操作的对象不是具体的数据，而是寄存器或存储器单元。也就是说，它直接与寄存器、存储器打交道。正因为汇编语言具备“机器相关性”，程序员用它编写程序时，可充分利用机器内部的各种资源，并使其始终处于最佳的工作状态。采用汇编语言编写的程序执行代码短、执行速度快。但是，不同类型的CPU有各自的机器指令系统，体现为不同形式的汇编语言。除了同系列、不同型号CPU之间的汇编语言程序有一定程度的可移植性之外，不同类型（如单片机和PC）CPU之间的汇编语言程序是无法移植的。所以，汇编语言程序的通用性和可移植性要比高级语言低。

归纳起来，汇编语言的总体特点有以下几个方面。

（1）机器相关性

汇编语言是面向机器（CPU）的低级语言，通常是为特定的计算机或系列计算机专门设计的。不同的机器有不同的汇编语言指令集，使用汇编语言编程能够充分发挥机器特性。

（2）高速度和高效率

汇编语言保持了机器语言的优点，具有直接和简洁的特点，可高效地访问、控制计算机的各种硬件设备，如磁盘、存储器、CPU、I/O 端口等，且占用内存少，执行速度快。因而，汇编语言是一种高效的程序设计语言。

（3）编程、调试的复杂性

由于汇编语言直接控制硬件，简单的任务实现会需要很多条汇编语言指令来完成，需要编程人员考虑到一切可能的软、硬件资源使用问题。若较为复杂的汇编应用程序出现问题，则不易调试。

2. 为什么要学习汇编语言

首先，通过学习、使用汇编语言，能够感知计算机程序的运行过程和运行原理，从而对计算机硬件与应用程序之间的联系和交互形成一个清晰的认识，锻炼自身的编程逻辑思维能力，掌握解决复杂问题的手段和技巧。只有对软、硬件知识融会贯通之后，才可能编写出质量较好的计算机加密、解密、病毒分析与防治、程序调试等程序。

其次，汇编语言是学习高级语言的基础，它能让人更好地理解高级语言。以C语言为例，最令人头疼的就是指针，指针指的是内存的地址，而汇编语言恰恰以讲解地址的形成和使用方法为重点。另外，对于C语言中的数据类型、形参、实参、函数调用、全局变量、局部变量等概念及操作，都与汇编语言中的操作直接关联。因此，只有学好汇编语言，才能更深入地理解和运用高级语言。请大家记住一句话：不懂汇编语言，永远不能成为一流程序员。

目前，ARM、PIC、DSP等智能芯片的应用越来越广泛，它们都是采用汇编语言来

编写底层设备驱动程序和实时性好的应用程序。8086 汇编语言非常具有代表性，学好之后，就掌握了软件编程的灵魂，也会对各种智能芯片的汇编语言不再陌生。

3. 如何学好汇编语言

要学好汇编语言，应从 16 位计算机的汇编语言入手，循序渐进，逐步过渡到 16 位、32 位汇编语言的混合编程。

由于汇编语言是面向机器的语言，在学习汇编语言的过程中，必须要了解与掌握 CPU 逻辑结构、寄存器常规使用方法、寻址方式、常用指令、数据的存储方式及常规编程方法。

汇编语言的实践性非常强，只有边学习边动手实践才能掌握其知识要领，切忌死记硬背。

1.4 Intel 系列 CPU 简介

1971 年，美国 Intel 公司推出了世界上第一款微处理器芯片，型号为 4004。该芯片内部集成了 2300 个晶体管，主频为 108 kHz，4 位数据操作。

1978 年，Intel 公司又首次推出 16 位的微处理器，命名为 i8086，同时推出与之相配合的数字协处理器 i8087。这两种芯片使用相互兼容的指令集，人们将此指令集统一称为 x86 指令集，该指令集属于复杂指令集（CISC）。i8086 具有 16 位数据通道，内存寻址能力达 1 MB，最高工作频率为 8 MHz。

1979 年，8088 芯片推出，它是第一块成功用于个人计算机的 CPU 芯片。8088 仍旧属于 16 位微处理器，内含 29 000 个晶体管，时钟频率为 4.77 MHz，地址总线为 20 位，内存寻址范围为 1 MB。8088 的内部数据总线为 16 位，外部数据总线为 8 位，这样做的目的是方便兼容以前的 8 位计算机主板。

1982 年，80286 芯片推出，它与 8086 和 8088 相比有了飞跃性的发展，虽然它仍旧是 16 位结构，但其内部集成了 13.4 万个晶体管，时钟频率由最初的 6 MHz 逐步提高到 20 MHz。80286 的内部和外部数据总线皆为 16 位，地址总线 24 位，可寻址达 16 MB 内存，是应用比较广泛的一款 CPU。IBM 公司则采用 80286 芯片推出 AT 机，在当时引起了世界轰动。

1985 年，80386 芯片推出，它是 x86 系列中第一款 32 位微处理器，而且制造工艺也有了很大的进步。80386 内含 27.5 万个晶体管，时钟频率从 12.5 MHz 提高到 33 MHz。80386 的内部和外部数据总线都是 32 位，地址总线也是 32 位，可寻址高达 4 GB 内存，可以使用 Windows 操作系统。

1989 年，80486 芯片推出，它的特殊意义在于，首次突破了 100 万个晶体管的集成界限，内含 120 万个晶体管。80486 将 80386、数字协处理器 80387 及一个 8 KB 的高速

缓存集成在一个 CPU 芯片内，并且在 x86 架构中首次采用了精简指令集（RISC）技术，可以在一个时钟周期内执行一条指令。80486 采用了突发总线（burst）方式，大大提高了 CPU 与内存数据交换的速度，主频高达 100 MHz。

1993 年，Intel 推出 Pentium CPU，它是 x86 架构的第五代微处理器。Pentium 本应命名为 80586 或 i586，最终命名为 Pentium，是因为阿拉伯数字无法用于注册商标。Pentium CPU 内部集成晶体管数目高达 310 万个，时钟频率由最初推出的 60 MHz 和 66 MHz，后提高到 200 MHz。100 MHz 的 Pentium 处理器比 33 MHz 的 80486DX 要快 6～8 倍。

后来，Pentium 处理器经历了奔腾、奔腾 II、奔腾III、奔腾 4 等 32 位时代，并走向 64 位时代。Intel 公司推出的安腾 64 位微处理器成为计算机领域的又一里程碑。基于 IA-64（IA 即 Intel Architecture 的简写）架构的处理器具有 64 位运算能力、64 位寻址空间、64 位数据通路及 3 GHz 的主频，突破了传统 IA-32 架构的许多限制，在数据的处理能力，系统的稳定性、安全性、可用性等方面获得了突破性的提高。

注意：80386 CPU 的工作模式有 3 种，分别为实模式、保护模式和虚拟 8086 模式。

在系统复位（reset）或上电（power on）时，CPU 以实模式方式启动。在实模式下，80386 以上 CPU 的内存寻址方式与 8086 相同，由 16 位段寄存器的内容乘以 16（10H）作为段基地址，再加上 16 位偏移地址，形成 20 位的物理地址，最大内存寻址空间 1 MB，最多内存分段数为 64 KB。此时，32 位的 x86 CPU 用作高速的 8086。

在 Windows 操作系统控制下，80386 CPU 工作于保护模式。保护模式的内存寻址采用 32 位段地址和偏移量，最大寻址空间为 4 GB，最大内存分段数 4 GB。

在保护模式下，CPU 可以进入虚拟 8086 模式。虚拟 8086 模式是在保护模式下实模式应用程序的运行环境，它的设置目的是保持与 8086 的兼容性，使 16 位的 DOS 程序可以在 80386 的保护模式下工作。虚拟 8086 模式实际上就是保护模式下的一项进程，并且受监控程序的监控。

习　题　1

填空题

1．一个字节是相邻的_____个二进制位。

2．能被计算机直接识别的语言是_____。

3．一般地，物理高电平代表二进制数值_____，而低电平代表数值_____。

4．$(100011000)_2$=_____$_{16}$

5．$(AE)_{16}$=_____$_2$

6．为了解决实际问题，用汇编语言所编写的程序被称为_____。

7．8086 CPU 中内部数据线宽度是 16 位，那么，它的字长是_____位。

8．汇编语言源程序必须先经过_____生成目标代码，再经过_____生成可执行代码后，方可在操作系统下运行。

9．与 8086 CPU 配合使用的数字协处理器芯片是_____。

10．计算机开机时以_____模式启动。

上机训练 1　调试工具 Debug 的常用命令

一、训练目的

熟悉常用的 Debug 命令及使用方法，为程序调试打好基础。

二、训练内容

1．进入 Debug。

在 Windows 7 下，进入 Debug 调试工具的方法有两种。一种方法是，直接在“开始”菜单搜索框中输入“debug”命令后按【Enter】键；另一种方法是，先在搜索框中输入“cmd”按【Enter】键，再在命令提示符窗口的光标处输入“debug”命令。Debug 界面如图 1.2 所示，Debug 的命令提示符为“-”。

图 1.2　Debug 调试工具界面

此时，就可以在闪动光标处输入 Debug 命令了。

在光标处输入具有显示和修改寄存器内容功能的“R”命令，就可以看到 CPU 内部寄存器的存储内容，如图 1.3 所示。

图 1.3　Debug 的 R 命令

各寄存器的名称及使用方法将在下一章讲述。下面先来介绍寄存器中的内容，这些内容都是用十六进制数表示的。在 Debug 中，不论是寄存器的内容还是存储器的内容，都是以十六进制形式表示的。

观察各寄存器的值，要注意以下两方面内容。

① 各段寄存器初始值相等，CS=DS=ES=SS，说明 Debug 不带程序调试时，代码段、数据段、堆栈段和附加段为同一个内存中可用的段。

② 在不同计算机上，由于运行环境存在差异，这个相等的值并不相同。

2. Debug 常用命令及使用方法。

Debug 命令都是以一个不区分大小写的字母来表示的，后面可跟一个或多个参数，也可以使用默认参数。输入每条命令后要按【Enter】键才被执行。

下面列出常用的 Debug 命令及其格式，读者要边上机操作边熟悉各命令的使用方法。

（1）查看或修改寄存器内容命令 R。

命令格式：R[寄存器名]

命令功能：不带参数，显示寄存器内容；带参数，则修改指定寄存器内容。例如，R AX 命令的执行结果如图 1.4 所示。

图 1.4 R AX 命令的执行结果

（2）显示内存单元内容命令 D。

命令格式：D [地址] 或 D [范围]

命令功能：不带参数，显示当前段中从偏移地址 0100H 开始的 128 个字节单元内容。D 命令的执行结果如图 1.5 所示。

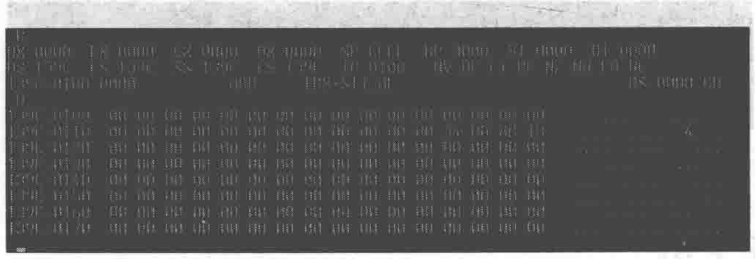

图 1.5 D 命令的执行结果

D 命令可带参数。例如：

D DS:200 命令是显示指定段中从偏移地址开始的 128 个字节单元内容。

D DS:200 20F 命令是显示指定段中偏移地址范围内的单元内容。该命令的执行结果

如图 1.6 所示。

图 1.6　D DS:200 20F 命令的执行结果

（3）修改内存单元内容命令 E。

命令格式：E 地址 [内容表]

命令功能：显示并修改指定段中由偏移地址开始的一个或多个单元内容，直到按【Enter】键为止。例如，E DS:200 命令的执行结果如图 1.7 所示。

图 1.7　E DS:200 命令的执行结果

E DS:200 11 22 33 44 命令是将内容表中的内容替换指定段中从偏移地址开始的对应单元内容。该命令的执行结果如图 1.8 所示。

图 1.8　E DS:200 11 22 33 44 命令的执行结果

（4）汇编命令 A。

命令格式：A [地址]

命令功能：从代码段中指定偏移地址处输入汇编语言指令，并将指令汇编成的指令代码从指定地址开始存放。如果省略参数，则第一次输入指令的机器码从代码段偏移地址 0100H 处开始存放。按【Enter】键能够结束汇编状态。A 100 命令的执行结果如图 1.9 所示。

图 1.9　A 100 命令的执行结果

（5）反汇编命令 U。

命令格式：U [地址] 或 U [范围]

命令功能：从指定地址开始，将主存 32 个字节范围内的机器指令反汇编成汇编语言指令，或将指定地址范围内的机器指令进行反汇编。如果该命令不带参数，则接着上一个 U 命令进行反汇编。若是第一次使用该命令，则从 CS:IP 处开始反汇编。U 命令的执行结果如图 1.10 所示。