

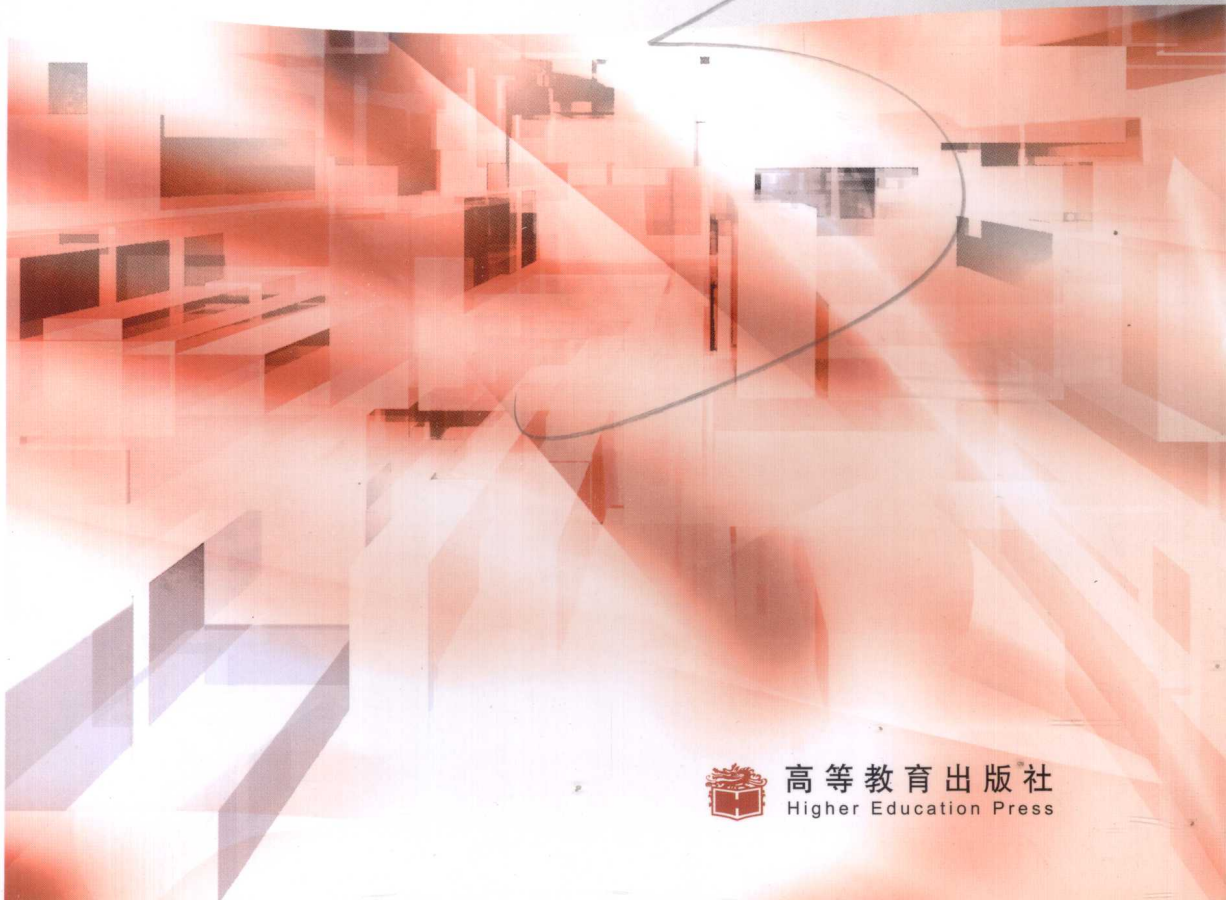


普通高等教育“十一五”国家级规划教材

汇编语言程序设计 (第2版)

Assembly Language Programming
(2nd Edition)

· 吴向军 罗源明 刘广旋 编著



高等教育出版社
Higher Education Press

普通高等教育“十一五”国家级规划教材

汇编语言程序设计

(第2版)

吴向军 罗源明 刘广旋 编著

高等教育出版社

内容提要

本书以 16 位/32 位 Intel CPU 为介绍对象,以并行推进的方式介绍这两类 CPU 的指令系统,以达到相互参考、对照的效果。另外,本书还介绍了常用的编程环境和调试工具的使用方法。

在内容组织上,把学习高级语言的习惯和汇编语言的特点相结合,按“硬件资源—变量定义—指令系统—编程”的顺序安排相应的内容。结合 MASM V6.1x 的编程环境,使汇编语言源程序具有一定的高级语言特点和结构。书中用一章来介绍浮点数的表示、浮点运算指令和编程方法,使读者对高级语言中的各种数据类型与低层实现之间的联系有一个更清晰的认识。还用一章介绍了用汇编语言编写 Windows 应用程序的基本方法,为拓宽汇编语言的应用领域做一点尝试。

书中附有一些基本的编程辅助信息,这些信息包括各种按键的码值和显示颜色信息等。在 Windows 应用程序的编写部分,附录中给出了编程中所用到的一些常量列表和函数功能的说明。这些辅助信息可为编程带来一定的方便,也可作为常用手册以待查阅。

本书可作为计算机学科的本科生、大专生及相近专业学生的教材,也可作为相关工程技术人员的学习资料或参考书。

图书在版编目(CIP)数据

汇编语言程序设计/吴向军,罗源明,刘广旋编著.
—2 版. —北京:高等教育出版社,2009.11
ISBN 978-7-04-027949-8

I. 汇… II. ①吴…②罗…③刘… III. 汇编语言—
程序设计—高等学校—教材 IV. TP313

中国版本图书馆 CIP 数据核字(2009)第 181504 号

策划编辑 刘 艳 责任编辑 张海波 封面设计 王 洋 责任绘图 郝 林
版式设计 范晓红 责任校对 杨凤玲 责任印制 朱学忠

出版发行	高等教育出版社	购书热线	010-58581118
社 址	北京市西城区德外大街 4 号	咨询电话	400-810-0598
邮政编码	100120	网 址	http:// www.hep.edu.cn
总 机	010-58581000		http:// www.hep.com.cn
经 销	蓝色畅想图书发行有限公司	网上订购	http:// www.landaco.com
印 刷	北京新丰印刷厂		http:// www.landaco.com.cn
		畅想教育	http:// www.widedu.com
开 本	787×1092 1/16	版 次	2002 年 6 月第 1 版
印 张	24.75		2009 年 11 月第 2 版
字 数	560 000	印 次	2009 年 11 月第 1 次印刷
		定 价	31.60 元

本书如有缺页、倒页、脱页等质量问题,请到所购图书销售部门联系调换。

版权所有 侵权必究
物料号 27949-00

前 言

汇编语言程序设计是计算机和电子信息类相关专业的一门重要的基础课程，是培养学生直接使用计算机硬件资源能力的一门课程。它不仅能使进一步理解计算机组成原理课程中的各种概念，而且还为其他课程，如操作系统、编译原理、接口与通信技术和计算机控制技术等课程，提供必要的预备知识。

本书以 Intel 80x86 CPU 的指令系统为介绍对象，以并行推进的方式来介绍其 16 位和 32 位 CPU 中的指令功能。选用这种方式，不仅希望利用 16 位指令系统简单好学的特点来加深对 32 位指令的理解，而且也希望读者在学习过程中能达到相互参考、相互对照的目的。

本书在内容的组织上做了大胆的尝试，把学习高级语言的习惯和汇编语言的特点结合起来，按“硬件资源—变量定义—指令系统—编程”的顺序来安排相应的内容。另外，还结合宏汇编 MASM 6.1x 编程环境介绍了一些新的伪指令，运用这些伪指令可使汇编语言源程序具有良好的结构化特征。

本书在介绍完汇编语言编程的基本方法和基本技巧后，在第十一章介绍浮点数的定义格式、浮点指令的功能和编程方法。该部分的内容不仅使计算机组成原理课程中对浮点数格式的描述更加具体化，而且也使读者对高级语言中的各种数据类型与低层实现之间的联系有一个清晰的认识。

16 位 CPU 的汇编语言程序设计是学习微机汇编语言的重要基础。在此基础上，在第十三章介绍了用汇编语言编写 Windows 应用程序的基本方法。目前很多高等学校都在进行一定程度的教学改革，在授课总时数缩减的大环境下，汇编语言课程的周学时已减少为 3，甚至更少。所以，第十一章和第十三章的内容可为选择性教学内容。编者相信读者在掌握本书基本内容的前提下，自学 Windows 应用程序的编程技术也不是一件困难的事。

在本书的编写安排上，罗源明副教授编写了前三章，吴向军副教授编写了其余内容和附录，刘广旋工程师对每章的配套练习做了相应的整理和编排工作，李磊和梁志勇等老师对配套练习做了相应的编写和测试工作。本书的统筹安排和最后定稿由吴向军副教授负责。

在编写本书的初期，姜丽帆副教授对编写计划给予了积极的肯定，也提出了一些建设性的意见。初稿完成后，李宏新高级工程师审阅了全部内容，并在内容的取舍和一些重要内容的组织安排上给出了修改意见。此外，北京航空航天大学计算机学院熊桂喜教授也对本书进行了审

稿，提出了宝贵的意见。在此，向他们表示衷心的感谢。

由于编者能力和知识更新速度等因素，书中不足和谬误之处在所难免，恳请广大读者批评指正。（E-mail: issxjwu@mail.sysu.edu.cn）

编 者

2009 年 7 月

目 录

第一章 预备知识	1	3.6 基址加变址寻址方式	27
1.1 汇编语言的由来及其特点	1	3.7 相对基址加变址寻址方式	28
1.1.1 机器语言	1	3.8 32 位地址的寻址方式	30
1.1.2 汇编语言	1	习题	31
1.1.3 汇编程序	2	第四章 标识符和表达式	33
1.1.4 汇编语言程序的主要特点	2	4.1 标识符	33
1.1.5 汇编语言的应用领域	3	4.2 简单内存变量的定义	33
1.2 数据的表示和类型	4	4.2.1 内存变量定义的一般形式	34
1.2.1 数值数据的表示	4	4.2.2 字节变量	34
1.2.2 非数值数据的表示	6	4.2.3 字变量	35
1.2.3 基本数据类型	8	4.2.4 双字变量	35
习题	9	4.2.5 六字节变量	36
第二章 CPU 资源和存储器	10	4.2.6 八字节变量	36
2.1 寄存器组	10	4.2.7 十字节变量	36
2.1.1 寄存器组	10	4.2.8 浮点类型变量	37
2.1.2 通用寄存器	11	4.3 调整偏移量伪指令	37
2.1.3 段寄存器	13	4.3.1 偶对齐伪指令 EVEN	37
2.1.4 专用寄存器	14	4.3.2 对齐伪指令 ALIGN	38
2.2 存储器的管理模式	17	4.3.3 调整偏移量伪指令 ORG	38
2.2.1 16 位微机的内存管理模式	17	4.3.4 偏移量计数器的值	39
2.2.2 32 位微机的内存管理模式	20	4.4 复合内存变量的定义	39
习题	21	4.4.1 重复说明符 DUP	40
第三章 操作数的寻址方式	23	4.4.2 结构类型的定义	40
3.1 立即寻址方式	23	4.4.3 联合类型的定义	42
3.2 寄存器寻址方式	24	4.4.4 记录类型的定义	44
3.3 直接寻址方式	24	4.4.5 数据类型的自定义	46
3.4 寄存器间接寻址方式	26	4.5 标号	46
3.5 寄存器相对寻址方式	27	4.6 内存变量和标号的属性	47

4.6.1 段属性操作符	47	第六章 程序的基本结构	101
4.6.2 偏移量属性操作符	47	6.1 源程序的基本组成	101
4.6.3 类型属性操作符	48	6.1.1 段的定义	101
4.6.4 长度属性操作符	48	6.1.2 段寄存器的说明语句	102
4.6.5 容量属性操作符	49	6.1.3 堆栈段的说明	104
4.6.6 强制属性操作符	49	6.1.4 源程序的结构	105
4.6.7 存储单元别名操作符	51	6.2 程序的基本结构	105
4.7 表达式	51	6.2.1 顺序结构	106
4.7.1 进制伪指令 RADIX	52	6.2.2 分支结构	108
4.7.2 数值表达式	52	6.2.3 循环结构	115
4.7.3 地址表达式	53	6.3 段的基本属性	121
4.8 符号定义语句	54	6.3.1 对齐类型 (ALIGN)	122
4.8.1 等价语句	54	6.3.2 组合类型 (COMBINE)	122
4.8.2 等号语句	56	6.3.3 类别 (CLASS)	123
4.8.3 符号名定义语句	56	6.3.4 段组 (GROUP)	123
习题	56	6.4 简化的段定义	126
第五章 微机 CPU 的指令系统	60	6.4.1 存储模式说明伪指令	126
5.1 汇编语言指令格式	60	6.4.2 简化段定义伪指令	128
5.1.1 指令格式	60	6.4.3 简化段段名的引用	129
5.1.2 指令的几个属性	60	6.5 源程序的辅助说明伪指令	130
5.2 指令系统	61	习题	131
5.2.1 数据传送指令	61	第七章 子程序和库	133
5.2.2 标志位操作指令	66	7.1 子程序的定义	133
5.2.3 算术运算指令	67	7.2 子程序的调用和返回指令	134
5.2.4 逻辑运算指令	72	7.2.1 调用指令	134
5.2.5 移位操作指令	74	7.2.2 返回指令	135
5.2.6 位操作指令	77	7.3 子程序的参数传递	137
5.2.7 比较运算指令	79	7.3.1 寄存器传递参数	137
5.2.8 循环指令	80	7.3.2 约定存储单元传递参数	139
5.2.9 转移指令	83	7.3.3 堆栈传递参数	141
5.2.10 条件设置字节指令	87	7.4 寄存器的保护与恢复	142
5.2.11 字符串操作指令	88	7.5 子程序的完全定义	143
5.2.12 ASCII-BCD 码运算调整指令	93	7.5.1 子程序完全定义格式	144
5.2.13 处理器指令	96	7.5.2 子程序的位距	144
习题	97	7.5.3 子程序的语言类型	144

7.5.4 子程序的可见性	145	9.1.1 宏的定义	195
7.5.5 子程序的“起始”和 “结束”操作	145	9.1.2 宏的引用	196
7.5.6 寄存器的保护和恢复	146	9.1.3 宏的参数传递方式	197
7.5.7 子程序的参数传递	147	9.1.4 宏的嵌套定义	198
7.5.8 子程序的原型说明	147	9.1.5 宏与子程序的区别	201
7.5.9 子程序的调用伪指令	147	9.2 宏参数的特殊运算符	201
7.5.10 局部变量的定义	148	9.2.1 连接运算符	201
7.6 子程序库	149	9.2.2 字符串整体传递运算符	202
7.6.1 建立库文件命令	149	9.2.3 字符转义运算符	203
7.6.2 建立库文件举例	150	9.2.4 计算表达式运算符	203
7.6.3 库文件的应用	151	9.3 与宏有关的伪指令	203
7.6.4 库文件的好处	154	9.4 重复汇编伪指令	206
习题	154	9.4.1 伪指令 REPT	207
第八章 输入输出和中断	156	9.4.2 伪指令 IRP	209
8.1 输入输出的基本概念	156	9.4.3 伪指令 IRPC	209
8.1.1 I/O 端口地址	156	9.5 条件汇编伪指令	210
8.1.2 I/O 指令	157	9.5.1 条件汇编伪指令的功能	210
8.2 中断	158	9.5.2 条件汇编伪指令的举例	211
8.2.1 中断的基本概念	158	9.6 宏的扩充	212
8.2.2 中断指令	159	9.6.1 宏定义形式	212
8.2.3 中断返回指令	160	9.6.2 重复伪指令 REPEAT	213
8.2.4 中断和子程序调用	160	9.6.3 循环伪指令 WHILE	213
8.3 中断功能的分类	161	9.6.4 循环伪指令 FOR	214
8.3.1 键盘输入的中断功能	162	9.6.5 循环伪指令 FORC	215
8.3.2 屏幕显示的中断功能	164	9.6.6 转移伪指令 GOTO	215
8.3.3 打印输出的中断功能	174	9.6.7 宏扩充的举例	216
8.3.4 串行通信口的中断功能	177	9.6.8 系统定义的宏	216
8.3.5 鼠标的中断功能	179	习题	218
8.3.6 目录和文件的中断功能	185	第十章 应用程序设计	220
8.3.7 内存管理的中断功能	188	10.1 字符串的处理程序	220
8.3.8 读取/设置中断向量	188	10.2 数据的分类统计程序	223
习题	192	10.3 数据转换程序	226
第九章 宏	195	10.4 文件操作程序	233
9.1 宏的定义和引用	195	10.5 动态数据的编程	240
		10.6 COM 文件的编程	243

10.7 驻留程序	244	13.2.2 调试语句	296
10.8 程序段前缀及其应用	248	13.3 Windows 程序的结构	297
10.8.1 程序段前缀的字段含义	248	13.3.1 Windows 程序的基本结构	297
10.8.2 程序段前缀的应用	249	13.3.2 程序的初始化阶段	303
习题	254	13.3.3 窗口的准备阶段	304
第十一章 数值运算协处理器	256	13.3.4 消息循环阶段	309
11.1 协处理器的数据格式	256	13.3.5 消息处理阶段	310
11.1.1 有符号整数	256	13.4 Windows 应用程序	311
11.1.2 BCD 码数据	257	13.4.1 按钮控件的应用	311
11.1.3 浮点数	257	13.4.2 对话框的定义	317
11.2 协处理器的结构	259	13.4.3 对话框和控件的应用	319
11.2.1 协处理器的内部结构	259	13.4.4 绘画和图像的应用	325
11.2.2 状态寄存器	260	13.4.5 命令行参数的处理	331
11.2.3 控制寄存器	263	习题	334
11.2.4 标记寄存器	264	第十四章 汇编语言编程环境和调试工具	335
11.3 协处理器的指令系统	264	14.1 汇编语言编程环境	335
11.3.1 指令操作符的命名规则	264	14.1.1 MASM 6.1x	335
11.3.2 数据传送指令	265	14.1.2 Turbo Assembler	343
11.3.3 数学运算指令	267	14.1.3 MASM32	344
11.3.4 比较运算指令	269	14.2 调试工具	345
11.3.5 超越函数运算指令	270	14.2.1 Debug	345
11.3.6 常数操作指令	270	14.2.2 CodeView	348
11.3.7 协处理器控制指令	270	14.2.3 Turbo Debugger	349
11.4 协处理器的编程举例	271	14.2.4 VKdebug	350
习题	280	附录	354
第十二章 汇编语言和 C 语言的混合编程	282	附录一 键盘按键的各种编码对照表	354
12.1 汇编指令的嵌入	282	附录二 显示地址及其显示属性	358
12.2 C 语言源程序的汇编输出	283	附录三 Windows 中各类预定义常量	360
习题	285	附录四 Windows 应用程序中部分 API 函数	371
第十三章 Windows 应用程序的编程	287	参考文献	387
13.1 消息框	287		
13.2 编程伪指令	293		
13.2.1 多路分支语句	293		

第一章 预备知识

本章介绍汇编语言的一些基本概念，给出一些用汇编语言编程所需要的基本知识。

1.1 汇编语言的由来及其特点

1.1.1 机器语言

机器指令是 CPU 能直接识别并执行的指令，它以二进制编码的形式来表示。机器指令通常由操作码和操作数两部分组成，操作码指出该指令所要完成的操作，即指令的功能。操作数指出参与运算的对象以及运算结果所存放的位置等。

由于机器指令与 CPU 紧密相关，所以不同种类的 CPU 所对应的机器指令也就不同，而且它们的指令系统往往相差很大。对同一系列的 CPU 而言，为了使各型号之间具有良好的兼容性，新一代 CPU 的指令系统必须包括先前同系列 CPU 的指令系统。只有这样，先前开发出来的各类程序才能在新一代 CPU 上正常运行。

机器语言是用来直接描述机器指令和使用机器指令规则等的语言。它是 CPU 能直接识别的唯一一种语言，即 CPU 能直接执行用机器语言描述的程序。

用机器语言编写程序是早期经过严格训练的专业技术人员的工作，普通的程序员一般难以胜任，而且用机器语言编写的程序不易读、出错率高、难以维护，也不能直观地反映用计算机解决问题的基本思路。

鉴于用机器语言编写程序有以上诸多的不便，现在已很少有程序员这样编写程序了。

1.1.2 汇编语言

虽然用机器语言编写程序对程序员有很高的要求并存在着许多不便，但编写出来的程序执行效率高，CPU 严格按照程序员的要求去做，没有多余的额外操作。所以，在保留“程序执行效率高”的前提下，人们开始着手研究一种能大大改善程序可读性的编程语言。

为了改善机器指令的可读性，应选用一些能反映机器指令功能的单词或词组来代表该机器指令，而不再关心机器指令的具体二进制编码。与此同时，也把 CPU 内部的各种资源符号化，在编写程序时，使用该符号名就相当于引用某一具体的物理资源。

如此一来，令人难懂的二进制机器指令就可用通俗易懂的、具有一定含义的符号指令来表

示了，这就是汇编语言的雏形。这些具有一定含义的符号称为助忆符，用指令助忆符、符号地址等组成的符号指令称为汇编格式指令（或汇编指令）。

汇编语言是汇编指令集、伪指令集和使用它们的规则的统称。伪指令是在程序设计时所需要的一些辅助性说明指令，它不对应具体的机器指令，有关内容在以后的章节中将有详细叙述，在此不展开介绍。

用汇编语言编写的程序称为汇编语言程序或汇编语言源程序，在本书中简称为源程序。汇编语言源程序要比用机器指令编写的程序容易理解和维护。

1.1.3 汇编程序

用汇编语言编写的程序大大提高了程序的可读性，但失去了 CPU 能直接识别的特性。例如用汇编语言书写的指令“MOV AX, BX”，CPU 不会知道这几个字符所表达的功能，但程序员一看就知道：要求 CPU 把寄存器 BX 的值传送给寄存器 AX。

把机器指令符号化增加了程序的可读性，但带来了如何让 CPU 知道程序员的用意并按照其要求完成相应操作的问题。解决该问题就需要一个翻译程序，它能把汇编语言编写的源程序翻译成 CPU 能识别的机器指令序列。这里，称该翻译程序为汇编程序。图 1.1 是一次翻译过程的示意图。

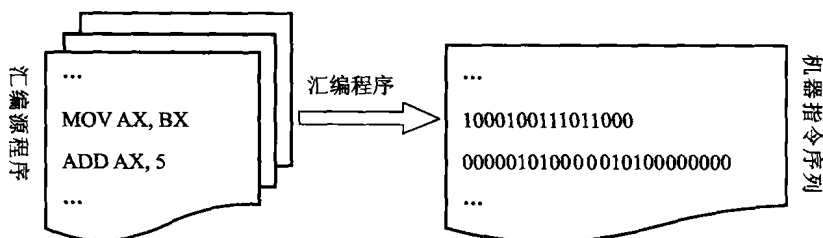


图 1.1 汇编语言指令到机器指令的示意图

从图 1.1 中不难看出，汇编程序能把左边的汇编语言源程序翻译成右边的机器指令序列。其中，把汇编语言指令“MOV AX, BX”和“ADD AX, 5”分别转换成二进制机器指令“1000100111011000”和“000001010000010100000000”，而后者是 CPU 能直接识别的。

目前，常用的汇编程序有：MASM、TASM 和 Debug 等。

1.1.4 汇编语言程序的主要特点

汇编语言指令是用一些具有相应含义的助忆符来表达的，所以，它比机器语言容易掌握和运用，但汇编语言直接使用 CPU 的资源，相对高级程序设计语言而言，它又显得较难掌握。

汇编语言归纳起来大概有以下几个主要特性。

1. 与机器相关性

汇编语言指令是机器指令的一种符号化表示，不同类型的 CPU 有不同的指令系统，其相应

的汇编语言也就有所不同，所以，汇编语言与机器有着密切的关系。

由于汇编语言与机器的相关性，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外，其他不同类型（如小型机和微机等）CPU 之间的汇编语言程序是无法移植的，也就是说，汇编语言程序的通用性和可移植性要比高级语言程序差。

2. 执行的高效率

正因为汇编语言具有“与机器相关”的特性，程序员用汇编语言编写程序时可充分发挥自己的聪明才智，对机器内部的各种资源进行合理的安排，让它们始终处于最佳的使用状态，这样做的最终效果就是：程序的执行代码短，执行速度快。

现在高级语言的编译程序在进行寄存器分配和目标代码生成时，也都有一定程度的优化，但由于所使用的“优化策略”要适应各种不同的情况，所以，这些优化策略只能在宏观上，而不可能在微观上、细节上进行优化。而用汇编语言编写程序几乎是程序员直接在写可执行代码，程序员可以在程序的每个具体细节上进行优化，这也是汇编语言程序执行效率高的原因之一。

3. 编写程序的复杂性

汇编语言是一种面向机器的语言，其汇编指令与机器指令基本上一一对应，所以，汇编指令也同机器指令一样具有功能单一、具体的特点。要想完成某项工作（如计算 $A+B+C$ 等），就必须安排 CPU 的每一步工作（如先计算 $A+B$ ，再把 C 加到前者的结果上）。另外，在编写汇编语言程序时，还要考虑机器资源的限制、汇编指令的细节和限制等。

由于汇编语言程序要安排运算的每一个细节，这就使得编写汇编语言程序比较烦琐、复杂。一个简单的计算公式或计算方法也要用一系列汇编指令一步一步来实现。

4. 调试程序的复杂性

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，其主要原因有 4 点。

① 汇编语言指令涉及机器资源的细节，调试时要清楚机器资源的变化情况。

② 程序员在编写汇编语言程序时，为了提高资源的利用率，可以使用各种实现技巧，而这些技巧完全有可能破坏程序的可读性。这样，在调试过程中，除了要知道每条指令的执行功能，还要清楚它在整个执行过程中的作用。

③ 高级语言程序几乎不显式地使用“转移语句”（goto 语句），但汇编语言程序要用到大量的转移指令，这些转移指令大大增加了调试程序的难度。如果在汇编语言程序中也强调不使用转移指令，那么，汇编语言程序就会变成功能单调的顺序程序。

④ 汇编语言的调试工具落后。高级语言程序可以在源程序级进行符号跟踪，而汇编语言程序只能跟踪机器指令。由于 CV（Code View）、TD（Turbo Debug）等软件也具有源程序级的符号跟踪功能，所以，汇编语言程序的调试方式有了极大的改善。

1.1.5 汇编语言的应用领域

综上所述，汇编语言的特点明显，其诱人的优点直接导致其严重的缺点，其“与机器相关”和“执行的高效率”导致其可移植性差和调试困难。所以，在选用汇编语言时要根据实际的应

用环境，尽可能避免其缺点对整个应用系统的影响。

下面简单列举几个领域以示说明，但不要把它们绝对化。

1. 适用的领域

- 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、实时系统等。
- ② 系统性能的瓶颈或频繁被使用的子程序或程序段。
- ③ 与硬件资源密切相关的软件开发，如设备驱动程序等。
- ④ 受存储容量限制的应用领域，如家用电器的计算机控制功能等。
- ⑤ 没有适当的高级语言开发环境。

2. 不宜使用的领域

- ① 大型软件的整体开发。
- ② 没有特殊要求的一般应用系统的开发等。

1.2 数据的表示和类型

用汇编语言进行程序设计时，程序员可以直接访问内存单元，因此对数据在存储器内的表示形式要有清晰的认识。下面只介绍最基本的数据表示知识，为本课程的学习作必要的知识准备。

1.2.1 数值数据的表示

1. 二进制

在计算机内，数值是用二进制来表示的。在书写二进制数时，在数据后面紧跟一个字母 B。表 1.1 列举出各数值进制及其字符表示。

表 1.1 数值进制及其字符表示

进 制	进制表示符	例 子	进 制	进制表示符	例 子
二进制	B/Y ^①	1010B, 1011B	十进制	D/T	1234D, 512D
八进制	Q/O	1234Q, 311Q	十六进制	H	1234H, 1011H

二进制数的一般形式为 $b_{n-1} \cdots b_1 b_0 B$ ，其代表的十进制数值为 $b_{n-1}2^{n-1} + \cdots + b_1 2^1 + b_0 2^0$ 。

数据的二进制表示形式简单、明了，但它书写起来比较长，所以，通常情况下，在程序中不直接用二进制来书写具体的数值，而改用八进制、十进制或十六进制表示。

2. 八进制

八进制是一种二进制的变形，3 位二进制数可变为 1 位八进制数，反之亦然。八进制的表示元素为 0, 1, ..., 7。书写时，在数据后面紧跟一个字母 Q。如 1234Q、7654Q、54Q 等都是

① 字符 Y、O 和 T 是宏汇编 MASM 6.1x 编程环境所增加的数值进制表示符。

八进制数。

八进制数在程序中的使用频率不高。

3. 十进制

十进制是最常见的一种数据表示形式，它的基本元素为 0, 1, ..., 9。在书写时，在数据后面紧跟一个字母 D。在程序中经常用十进制来表示数据。

4. 十六进制

十六进制是另一种二进制的变形，4 位二进制数可变为 1 位十六进制数，反之亦然。十六进制的基本元素为 0, 1, ..., 9, A, B, ..., F（字母小写也可以），其中，字母 A, B, ..., F 依次代表十进制数的 10, 11, ..., 15。

在书写时，在数据后面紧跟一个字母 H。若十六进制数的第一个字符是字母，则在其之前必须添加一个“0”。如 100H、56EFH、0FFH、0ABCDH 等都是十六进制数。

十六进制数在程序中的使用频率较高。

5. 数的补码表示法

在计算机中，为了表示正负数，并便于进行各种算术运算，对有符号数采用二进制的补码表示形式。补码的最高位用来表示正负数：0 表示正数，1 表示负数。正数的补码是其自身的二进制形式，负数的补码是把其正数的二进制编码变“反”，再加 1 而得到。

6. 二进制数的符号扩展

在汇编语言中，经常要对字节、字和双字的数据进行操作。当把“字节”转换成“字”，或把“字”转换成“双字”时，就需要进行符号扩展。符号扩展的具体操作就是把已知信息的最高位扩展到所有更高位。

例 1.1 把 8 位补码 01011010、10101100 分别扩展成 16 位补码。

解：根据符号扩展的含义，“字节→字”的具体扩展结果如图 1.2 所示。

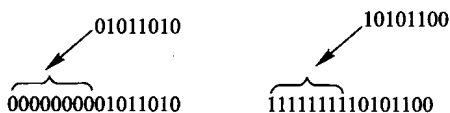


图 1.2 8 位数值扩展为 16 位数值的示意图

例 1.2 把 16 位补码 0101101111001010、101011101011011 别扩展成 32 位补码。

解：根据符号扩展的含义，“字→双字”的具体扩展结果如图 1.3 所示。

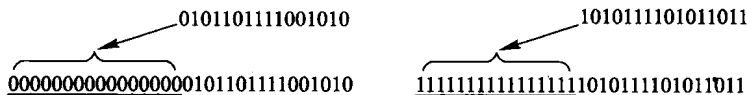


图 1.3 16 位数值扩展为 32 位数值的示意图

7. n 位二进制的表示范围

n 位二进制所能表示的无符号整数的范围： $0 \leq x \leq 2^n - 1$ 。

n 位二进制所能表示的有符号整数（补码表示）的范围： $-2^{n-1} \leq x \leq 2^{n-1} - 1$ 。

在汇编语言中，常用到 n 为 8 和 16 时的数值范围。

① $n = 8$ 时，无符号整数的范围：0~255，有符号整数的范围：-128~127。

② $n = 16$ 时，无符号整数的范围：0~65 535，有符号整数的范围：-32 768~32 767。

8. BCD 码

通常习惯用十进制来表示数据，但计算机是用二进制来表示数据的，这就需要进行数值进制之间的转换。把每位十进制数转换成二进制的编码称为 BCD（Binary Coded Decimal）码。

BCD 码是用 4 位二进制编码来表示 1 位十进制数。这种编码方法有多种，但常用的编码是 8421BCD 编码，如表 1.2 所示。这种 BCD 编码实际上就是 0~9 的“等值”二进制数。

表 1.2 8421BCD 编码列表

十进制数字	8421BCD 码	十进制数字	8421BCD 码
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

用 BCD 码进行进制的转换时，要求在两种进制的表现形式上进行快速转换，而不要求在“数值相等”的含义下进行快速转换。

例 1.3 求十进制数 2000 的 BCD 编码和其二进制数。

解：2000 的 BCD 编码是把每位上的数 2, 0, 0, 0 分别转换为其对应的 BCD 编码 0010, 0000, 0000 和 0000，把它们合在一起就是 2000 的 BCD 编码 0010 0000 0000 0000。

十进制数 2000 的二进制数是 11111010000，它们在数值上是相等的。

9. 压缩型 BCD 码

压缩型 BCD 码是在一个字节中存放两个 BCD 码，即：低四位和高四位都是一个 BCD 码。如 1000 0110，它是由 1000 和 0110 组成的压缩 BCD 码。第 5.2.12 节中将介绍一些与压缩型 BCD 码相关的汇编指令。

1.2.2 非数值数据的表示

计算机除了具有进行数值计算的能力外，还具有进行非数值计算的能力。现在，后者的应用领域已远远超过了前者，如文字处理、图形图像处理、信息检索、日常的办公管理等。所以，对非数值信息的编码就显得更加重要。

1. ASCII 码

ASCII (American Standard Code for Information Interchange) 码是目前应用极其广泛的一种信息编码, 许多计算机系统都是采用它为字符进行编码。它是一种 7 位二进制编码, 表 1.3 是 ASCII 码的具体编码方案。

表 1.3 ASCII 码的编码方案

高位 低位	000	001	010	011	100	101	110	111
0000	NUL	DEL	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

在表 1.3 中, 对学习本课程有用的主要信息有:

- ① 字符'0'~'9'是连续编码的, 其 ASCII 码的低 4 位就是该字符在十进制中对应的数值。
- ② 小写字母的 ASCII 码比大写字母的 ASCII 码大, 对应字母的编码之间相差 20H。

当然从 ASCII 码表中还可看出其他有用信息, 还有扩展的 ASCII 码等知识, 但这些内容对学习本课程的帮助不明显, 故不再叙述。

2. 汉字编码

ASCII 码是针对英文的字母、数字和其他特殊字符进行编码的, 它不能用于对汉字的编码。要想用计算机来处理汉字, 也必须先对汉字进行适当的编码。我国在 1981 年 5 月对 6 000 多个常用的汉字制定了交换码的国家标准, 即 GB2312—80。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 7 445 个, 其中汉字有 6 763 个。该标准给定了每个字符的二进制编码, 即国标码。

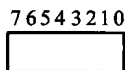
有关汉字编码的详细信息，请参阅其他有关书籍，在此不再介绍。

1.2.3 基本数据类型

汇编语言所用到的基本数据类型有字节、字和双字等，这些数据类型在以后的章节中都有相应的类型说明符。下面对它们进行最基本的描述。

1. 字节

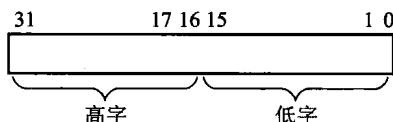
一个字节由 8 位二进制数组成，其最高位是第 7 位，最低位是第 0 位，如图 1.4 (a) 所示。在表示有符号数时，最高位就是符号位。



(a) 字节



(b) 字



(c) 双字

图 1.4 各类基本数据类型示意图

通常情况下，存储器按字节编址，读写存储器的最小信息单位就是一个字节。

2. 字

由 2 个字节组成一个字，其最高位是第 15 位，最低位是第 0 位。高 8 位称为高字节，低 8 位称为低字节，如图 1.4 (b) 所示。

字节和字是汇编语言程序中最常用的两种数据类型，也是最容易出错的数据类型。

3. 双字

由 2 个字（4 个字节）组成一个双字，其高 16 位称为高字，低 16 位称为低字，如图 1.4 (c) 所示。双字有较大的数据表示范围，它通常是为了满足数据的表示范围而选用的数据类型，也可用于存储远指针。

4. 四字

由 4 个字（8 个字节）组成一个四字类型，它总共有 64 个二进制位，当然也就有更大的数据表示范围，但在汇编语言中很少使用该数据类型。

5. 十字节

由 10 个字节组成一个十字节类型，它总共有 80 个二进制位。在汇编语言中很少使用该数据类型，但在表示浮点数时会有所涉及。

6. 字符串

字符串是由若干个字节组成的，字节数不定，通常每个字节存储一个字符。该数据形式是汇编语言程序中经常使用的一种数据形式，它没有 C 语言中的规定：以 ASCII 码 0 为字符串的结束符。