21世纪高等教育计算机规划教材

COMPUTER

汇编语言程序设计教程

A Guide for Assembly Language Programming

王庆生 主编

刘维平 刘政怡 徐怡 陈洁 张燕平 副主编

从指令执行看计算机工作原理

用程序实例训练和提高编程能力

以注重基础结合实验为编排指导

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等教育计算机规划教材

COMPUTER

汇编语言程序设计教程

A Guide for Assembly Language Programming

王庆生 主编

刘维平 刘政怡 徐怡 陈洁 张燕平 副主编

人民邮电出版社

北京

图书在版编目（CIP）数据

汇编语言程序设计教程 / 王庆生主编. -- 北京 : 人民邮电出版社, 2013.1
21世纪高等教育计算机规划教材
ISBN 978-7-115-29840-9

Ⅰ. ①汇… Ⅱ. ①王… 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(2012)第294021号

内 容 提 要

本书主要介绍基于80x86的汇编语言程序设计方法和技术，共分为3个部分。第1章～第8章为主体部分，包括计算机基本组成结构和指令系统，循环、分支、子程序和宏汇编技术的程序设计；第9章～第10章为中断与输入输出的一些典型应用；第3部分为上机实验，其中有9组实验题供选用和参考。每章都有丰富的程序实例和习题，并提供实验题的指导和习题参考答案。

本书是国家质量工程特色专业“TS11483”的配套教材，内容的编排和实例的讲解力求思路清晰、通俗易懂、深入浅出。

本书可作为高等院校本科以及大专院校计算机和电子信息类专业的教材，也可供从事汇编语言编程的读者自学参考。

21世纪高等教育计算机规划教材

汇编语言程序设计教程

◆ 主　　编　王庆生
副 主 编　刘维平　刘政怡　徐　怡　陈　洁　张燕平
责任编辑　董　楠

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
大厂聚鑫印刷有限责任公司印刷

◆ 开本：787×1092　1/16
印张：17　　2013年1月第1版
字数：446千字　　2013年1月河北第1次印刷

ISBN 978-7-115-29840-9

定价：34.00元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

汇编语言程序能直接而精确地控制计算机硬件的操作，在一些需要直接控制硬件的应用场合，汇编语言更能发挥其优势。

汇编语言与计算机硬件（特别是核心部件 CPU）具有高度相关性，能清楚细致地体现计算机硬件和软件互为支撑、紧密合作的“骨肉”联系。汇编语言程序的运行能使我们更深入理解计算机的工作原理和特点，它从程序角度帮助我们认知计算机，单纯地介绍计算机的硬件知识或一门高级语言的程序设计是不可能做到这点的。

作为计算机学科中处于基础地位的“汇编语言”，不仅具有兼顾计算机硬件和软件这一特点，而且对后续的“计算机组成原理”、“微机原理与接口技术”、“嵌入式系统”、“单片机技术”等相关课程的学习会有所帮助。因此，“汇编语言程序设计”依然作为专业基础核心课程，并不因 C++、Java 等编程语言的出现而淘汰。

本书是国家质量工程特色专业“TS11483”的配套教材，在编写本书和教学的过程中，我们并不仅仅是为了讲授一门编程语言，而是希望读者能够通过指令理解硬件，通过编程提高程序设计能力。在基本程序设计部分，程序实例都由思路和流程图开始，引导读者养成正确的编程习惯。和其他编程语言一样，上机实践非常重要，所以本书的第 2 章就以两个简单实例开始介绍程序的编写、汇编、连接和调试等全过程，而不必等到学完指令系统才介入上机。

本书内容的编排和实例的讲解力求思路清晰，循序渐进。例题的选择不主张难度，而注重体现基本内容和有键盘输入及显示输出的典型应用，既便于上机验证，又不致使初学者望而生畏。

作者希望本书能体现知识内容有梯度，要点分析有深度，既保证内容的系统性，也保留选择性，以兼顾各层次的学生，既可作为本科教材，也能适用专科层次，同时也可供技术人员自学参考。

本书介绍基于 80x86 的汇编语言程序设计的方法和技术。全书分为 3 部分共 10 章，第 1 章～第 8 章为基本内容，其中第 4 章的 32 位新增指令为可选部分，第 9 章～第 10 章主要介绍中断与输入输出应用，也作为可选部分。

为配合上机实验，本书实验部分提供了 9 组实验题供选用和参考。附录部分收录了指令和伪指令、DOS 和 BIOS 功能调用，便于查阅。

本书配套课件及书中的习题和实验题的参考答案，可登录人民邮电出版社教学服务与资源网（www.ptpedu.com.cn）免费下载或通过 wqs2001@163.com 邮箱索取。

尽管本书编写组的各位教师多年教授此课，但一定也会有不足和不当之处，欢迎广大读者批评指正。

编　者

2012 年 10 月

目 录

第 1 章 汇编语言基础知识 ……1
1.1 汇编语言简介 ……1
1.1.1 机器语言与汇编语言 ……1
1.1.2 为什么要学习汇编语言 ……2
1.2 计算机中数据的表示 ……2
1.2.1 不同进位计数制及其相互转换 ……2
1.2.2 二进制数和十六进制数的运算 ……5
1.2.3 带符号数的补码表示 ……5
1.2.4 补码的加法和减法 ……6
1.2.5 无符号数的表示 ……7
1.2.6 字符的表示 ……7
1.2.7 基本逻辑运算 ……8
1.3 计算机组织 ……9
1.3.1 计算机系统组成 ……9
1.3.2 中央处理器（CPU）中的寄存器 ……10
1.3.3 存储器 ……13
1.3.4 CPU 对存储器的读写操作 ……15
1.3.5 外部设备和接口 ……17
1.3.6 32 位 80x86CPU 的工作模式 ……18
习题 ……18
第 2 章 汇编语言程序实例及上机操作 ……20
2.1 汇编语言的工作环境 ……20
2.1.1 汇编语言的系统工作文件 ……20
2.1.2 进入 DOS 命令行方式 ……21
2.1.3 常用的 DOS 命令 ……21
2.2 汇编语言程序实例 ……23
2.2.1 单个字符的键盘输入与显示输出 ……23
2.2.2 显示字符串 ……24
2.3 程序实例的上机步骤 ……25
2.3.1 编辑——建立 ASM 源程序文件 ……25
2.3.2 汇编——产生 OBJ 二进制目标文件 ……25
2.3.3 连接——产生 EXE 可执行文件 ……26
2.3.4 LST 列表文件 ……26
2.3.5 程序的运行和调试 ……28
2.4 几个常用的 DOS 系统功能调用（INT 21H） ……30
2.5 DEBUG 主要命令 ……32
习题 ……35
第 3 章 操作数的寻址方式 ……37
3.1 立即寻址方式（immediate addressing） ……37
3.2 寄存器寻址方式（register addressing） ……38
3.3 直接寻址方式（direct addressing） ……38
3.4 寄存器间接寻址方式（register indirect addressing） ……39
3.5 寄存器相对寻址方式（register relative addressing） ……40
3.6 基址变址寻址方式（based indexed addressing） ……41
3.7 相对基址变址寻址方式（relative based indexed addressing） ……41
习题 ……42
第 4 章 指令系统 ……44
4.1 数据传送指令 ……45
4.1.1 通用数据传送指令 ……45
4.1.2 累加器专用传送指令 ……47
4.1.3 地址传送指令 ……49
4.1.4 标志寄存器传送指令 ……49
4.2 算术运算指令 ……50
4.2.1 类型扩展指令 ……50
4.2.2 加法指令 ……50
4.2.3 减法指令 ……52
4.2.4 乘法指令 ……54
4.2.5 除法指令 ……55
4.2.6 BCD 码的十进制调整指令 ……57

4.2.7 非压缩 BCD 码的十进制调整指令……58
4.3 逻辑与移位指令……59
4.3.1 逻辑指令……59
4.3.2 移位指令……61
4.4 串操作指令……62
4.4.1 MOVS 串传送指令……62
4.4.2 CMPS 串比较指令……65
4.4.3 SCAS 串扫描指令……66
4.4.4 STOS 串存入指令……68
4.4.5 LODS 从串取数指令……68
4.5 程序转移指令……69
4.5.1 无条件转移指令与程序的可重新定位……69
4.5.2 条件转移指令……72
4.5.3 循环指令……74
4.5.4 子程序调用……75
4.5.5 中断调用指令……76
4.6 处理器控制指令……77
4.6.1 标志处理指令……77
4.6.2 其他处理机控制指令……77
4.7 80386 后继机型的新增指令和寻址方式（*）……78
4.7.1 数据传送指令……78
4.7.2 位操作指令……79
4.7.3 串操作指令……80
4.7.4 算术指令和其他指令……81
4.7.5 条件测试并设置指令……82
4.7.6 增加的寻址方式……82
习题……84

第 5 章 伪指令与源程序格式……88

5.1 伪指令……88
5.1.1 处理机选择伪指令……88
5.1.2 段定义伪指令……89
5.1.3 程序开始和结束伪指令……91
5.1.4 数据定义与存储器单元分配伪指令……91
5.1.5 类型属性操作符……92
5.1.6 THIS 操作符和 LABEL 伪操作……93
5.1.7 表达式赋值伪指令“EQU”和“=”……94
5.1.8 汇编地址计数器$与定位伪指令……94
5.1.9 基数控制伪指令……95
5.1.10 过程定义伪指令……96
5.2 语句格式……96
5.2.1 名字项和操作项……97
5.2.2 表达式和操作符……97
5.3 EXE 文件与 COM 文件……100
5.3.1 程序段前缀 PSP……100
5.3.2 用 RET 指令结束主程序……101
5.3.3 COM 文件……102
习题……103

第 6 章 分支与循环程序设计……107

6.1 分支程序设计……107
6.1.1 分支程序结构……107
6.1.2 单分支程序……108
6.1.3 复合分支程序……108
6.1.4 多分支程序……110
6.2 循环程序设计……111
6.2.1 循环程序结构……111
6.2.2 计数循环程序……112
6.2.3 条件循环程序……115
6.2.4 条件计数循环程序……118
6.2.5 多重循环程序……120
习题……123

第 7 章 子程序设计……125

7.1 子程序结构……125
7.1.1 过程定义与过程结构……125
7.1.2 保存和恢复现场寄存器……127
7.2 子程序的参数传递……127
7.2.1 用寄存器传递参数……128
7.2.2 用变量传递参数……131
7.2.3 用地址表传递参数的通用子程序……132
7.2.4 用堆栈传递参数的通用子程序……133
7.2.5 用结构变量传递参数的通用子程序……135
7.3 多模块程序设计……139
7.3.1 多模块之间的参数传递……139

7.3.2 显示十进制数的通用模块……141
7.3.3 C语言程序调用汇编语言子程序……144
7.3.4 段的完整定义……145
7.3.5 连接程序的作用及对程序设计的要求……146
7.3.6 简化的段定义……147
习题……150
第8章 宏汇编及其他高级伪操作……152
8.1 宏汇编……152
8.1.1 宏定义、宏调用和宏展开……152
8.1.2 宏定义的嵌套……154
8.1.3 宏定义中使用宏调用……155
8.1.4 带间隔符的实参……156
8.1.5 连接操作符&……156
8.1.6 宏替换操作符%……157
8.1.7 LOCAL伪操作……157
8.1.8 使用宏库文件……159
8.2 其他高级伪操作……162
8.2.1 PURGE伪操作……162
8.2.2 列表伪操作……162
8.2.3 重复汇编……163
8.2.4 条件汇编……166
习题……168
第9章 输入输出和中断……170
9.1 外部设备与输入/输出……170
9.1.1 I/O端口……170
9.1.2 I/O指令……171
9.1.3 I/O的数据传送控制方式……172
9.2 中断……175
9.2.1 中断的概念……175
9.2.2 中断向量表……177
9.2.3 中断过程……180
9.3 中断处理程序设计……180
9.3.1 中断处理程序的基本功能……180
9.3.2 中断处理程序设计举例……181
习题……183
第10章 输入输出应用……185
10.1 可编程定时器……185
10.1.1 可编程定时器工作原理……185
10.1.2 定时器驱动扬声器发声……187
10.1.3 通用发声程序……188
10.1.4 乐曲程序……191
10.2 键盘调用……193
10.2.1 字符码与扫描码……193
10.2.2 键盘中断调用……194
10.2.3 键盘缓冲区……195
10.3 显示器的文本方式显示……196
10.3.1 显示方式……196
10.3.2 显示存储器与直接写屏……198
10.3.3 BIOS调用……199
10.4 显示器的图形方式显示……203
10.4.1 图形存储器……203
10.4.2 直接视频显示……205
10.4.3 BIOS功能视频显示……206
10.5 磁盘文件存取……208
10.5.1 文件代号方式存取……208
习题……214
上机实验……215
实验1 上机过程及程序调试……215
实验2 算术及位串处理程序……216
实验3 分支程序设计……216
实验4 循环程序设计……217
实验5 子程序设计……217
实验6 模块化程序设计……218
实验7 宏汇编程序设计……218
实验8 中断程序设计……219
实验9 输入输出程序设计……219
附录1 80x86指令系统一览……220
附录2 伪操作与操作符……234
附录3 中断向量地址一览……247
附录4 DOS系统功能调用(INT 21H)……250
附录5 BIOS功能调用……258
附录6 windows 104键键盘扫描码……264
参考文献……266

第 1 章 汇编语言基础知识

1.1 汇编语言简介

汇编语言是基于具体硬件的编程语言，需要了解编程中涉及的硬件知识，例如对于我们普遍使用的个人计算机来说，需要了解 80x86 计算机系统的核心部件中央处理器（CPU）的寄存器、存放数据和程序的存储器以及外部设备和接口。通过本章的学习，认识汇编语言的意义，重点熟悉计算机中数据和字符的常用表示方法、补码的运算、CPU 中的寄存器以及主存储器的分段，为下一步学习汇编语言程序设计打下基础。这些基础知识对深入了解计算机也很有帮助。

1.1.1 机器语言与汇编语言

计算机程序是由各种程序设计语言根据编程规则实现的，计算机程序设计语言经历了从低级到高级的发展，通常分为三类：机器语言（Machine Language）、汇编语言（Assembly Language，也有叫组合语言）、高级语言（High Level Language）。

机器语言：计算机硬件直接识别的程序设计语言。构成这种程序的是机器指令，机器指令是用二进制编码的指令，即编码中只含二进制 0 或 1，如 1110011000110011 就是一条机器指令。由于计算机主要由数字电路构成，所以机器指令由计算机直接记忆、传输、识别和加工。

机器语言被称为第一代语言，不仅复杂难记，而且还依赖于具体的机型。程序编写难度极大，调试修改困难，无法在不同的机型间移植。早已没有人用机器语言写程序了。

汇编语言：一种面向机器的用符号表示的程序设计语言，所以也叫符号语言。和机器语言不同的是，汇编语言用直观、便于记忆和理解的英文单词或缩写符号来表示指令和数据变量，例如，MOV　AX，VAL 是一条传送指令，其中 MOV 是指令操作码，AX 是 CPU 中的寄存器，VAL 是一个变量的符号表示，指令表示将变量 VAL 的值传给 AX。所以汇编指令也叫符号指令，这些符号称为助记符。汇编指令集和伪指令集及其使用规则的统称就是汇编语言。汇编语言被称为第二代语言。

对于 MOV　AX，VAL 这样的符号指令，比较简洁易读，但是计算机并不识别助记符，只能识别二进制编码的机器指令，因此需要通过一种翻译程序把汇编语言源程序翻译成机器语言程序，才能提交计算机执行。这种翻译程序叫汇编程序，这种对汇编语言源程序的翻译过程简称汇编。汇编语言的出现，大大改善了编程条件，使更多的人可以进行程序设计了。

尽管用汇编语言编写的程序要比机器代码更容易理解，但每条汇编语言指令均对应一条机器

指令，因而与机器语言并没有本质区别，因此汇编语言仍然属于面向机器的低级语言。

为了克服低级语言程序的不好理解、编程调试困难、不易移植的弊端，人们迫切希望有一种近乎自然语言或数学表达形式的程序设计语言，使程序设计工作能避开与机器硬件相关，而着重于解决问题的算法本身，于是便产生了高级语言，例如 BASIC、C、Java 等高级语言。高级语言被称为第三代语言。

用高级语言编写的源程序也必须经过编译和连接，将其转换为机器语言程序提交给计算机执行，或将其转换为一种中间代码，通过解释程序解释运行。

无论你是用什么语言编程，最终在计算机硬件中执行的程序都是由机器语言指令组成的，因此汇编语言是最接近计算机硬件的。

1.1.2 为什么要学习汇编语言

高级语言易学好用，那为什么还要学习汇编语言呢?

（1）学习汇编语言对于从事计算机应用开发有重要作用。汇编语言程序是由符号指令写成的，本质上还是机器语言，与具体机型的硬件结构密切相关，可直接、有效地控制计算机硬件，运行速度快，程序短小精悍，占用内存容量少。在某些特定应用场合更能发挥作用，如实时控制系统，需要对硬件设备直接进行数据的输入输出和控制，如在嵌入式系统和智能化仪器的开发中，需要更好地利用有限的硬软件资源，发挥硬件的功能。

（2）学习汇编语言是从根本上认识和理解计算机工作过程的最好方法，通过汇编语言指令，可以清楚地看到程序在计算机中如何一步步执行，有利于更深入理解计算机的工作原理和特点，单纯地介绍计算机的硬件知识或一门高级语言的程序设计是不可能做到这点的。汇编语言把软件和硬件紧密地结合在一起，起到连接硬件和软件的桥梁作用，掌握汇编语言对今后学习其他计算机相关课程非常有利。

1.2 计算机中数据的表示

1.2.1 不同进位计数制及其相互转换

1. 二进制数

进位计数制是一种计数方法，我们最熟悉的是十进制数，如 423.5 可表示为

$423.5=4\times10^2+2\times10^1+3\times10^0+5\times10^{-1}$

这里每位数字只能取 0 到 9 共 10 个数字符号，因此基数为 10。逢 10 进 1。不同位置上的数字代表的“权”是不同的，如百位 4，该位的权值为 10^2，第 K 位的权值为 10^K。

计算机为便于数字电路对数据存储及计算的物理实现，采用二进制数。二进制数只有 0 和 1 两个数码，基数为 2，逢 2 进 1。不同位置上的数码代表的“权”不同，即各位权值为 2^k。例如二进制数 $101101B=1\times2^5+1\times2^3+1\times2^2+1\times2^0=45D$。

可以看出，上面的二进制数按权展开就得到对应的十进制数。为便于明确计数制而不致误解，通常在二进制数后面加（B），在十进制数后面加（D）（也可用下标 2 和下标 10）。即：

$101101B=45D$，或 $(101101)_2=(45)_{10}$

n 位二进制数可以表示 2^n 个数，例如 4 位二进制数，可以表示 16 个数，如表 1.1 所示。

表 1.1　　二进制与十进制

二进制数	0000	0001	0010	0011	0100	0101	0110	0111
十进制数	0	1	2	3	4	5	6	7
二进制数	1000	1001	1010	1011	1100	1101	1110	1111
十进制数	8	9	10	11	12	13	14	15

2. 十六进制数

用二进制数表示一个较大的数总是不太方便，为便于程序员表示数据，通常还采用十六进制。

十六进制数有 16 个数码，基数为 16，逢 16 进 1。第 K 位置上的数码代表的权值为 16^k。每位的数码作如下规定：0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F，共 16 个数码。其中 A，B，C，D，E，F 分别表示十进制的 10，11，12，13，14，15。十六进制数后面要加上 H 以示区别。

例如十六进制数：

$5FH=5\times16^1+15\times16^0=80+15=95D$

显然十六进制表示比二进制表示来得简洁，该数用二进制表示则为

5FH=01011111B

一位十六进制数用四位二进制数表示即可，反之亦然。可见二进制数与十六进制数有着简单直接的互相转换关系，这是因为十六进制数的基数是 2 的幂。不仅如此，二进制数与 2^K 进制数都可以简单直接地互相转换。

实际生活中存在多种计数制，如计时采用时分秒就是 60 进制。但道理都是一样的。

3. 二进制数、十六进制数转换为十进制数

如前所述，各位二进制数码乘以对应的权之和即得到十进制数。如：

例 1.1　$N=101101.1B=1\times2^5+1\times2^3+1\times2^2+1\times2^0+1\times2^{-1}=45.5D$

各位十六进制数码乘以对应的权之和即得到十进制数。如：

例 1.2　$N=5FH=5\times16^1+15\times16^0=80+15=95D$

4. 十进制数转换为二进制数

这里介绍常用的两种方法。

（1）降幂法（适用于数值不大的数），降幂法就是先写出小于此数的各位二进制权值，然后再求和。

例 1.3　求 N=13.5D 的二进制数。小于此数的各位二进制权值为

8　　4　　2　　1　　0.5

显然应选 8，再选 4，而不能选 2（因为 8+4+2=14），再选 1，最后选 0.5，所以：

13.5D=8+4+1+0.5 =1101.1B

```
  1000
  0100
  0001
+    0.1
------
  1101.1
```

（2）除法（又叫除 2 取余法，仅适用于整数部分），除 2 取余法就是不断除以 2，记下余数，直到商为 0 为止。

例 1.4 求 N=13D 的二进制数。

13/2=6　　余 1　(b_0) ↑
6/2=3　　余 0　(b_1) |
3/2=1　　余 1　(b_2) |
1/2=0　　余 1　(b_3) |

$13D=b_3b_2b_1b_0=1101B$

对于二进制数的小数部分除了降幂法也可采用乘法，即不断乘 2，并记下整数，而小数部分再乘 2，直到结果的小数部分为 0 为止。注意：并非所有的十进制小数都能用二进制完全表示，如小数 0.3，这时按实际需要取一定精度表示即可。

例 1.5 求 N=0.625D 的二进制数。

$0.625\times2=1.25$　　($b_{-1}=1$)
$0.25\times2=0.5$　　($b_{-2}=0$)
$0.5\times2=1.0$　　($b_{-3}=1$) ↓

$N=0.625D=b_{-1}b_{-2}b_{-3}=0.101B$

5. 十进制数转换为十六进制数

和十进制数转换为二进制数类似，也有降幂法和除法。

（1）降幂法（适用于数值不大的数），降幂法就是先写出小于此数的各位十六进制数权值，然后再求和。

例 1.6 求 N=95D 的十六进制数。小于此数的各位十六进制权值为

16　　1

显然应选 16×5，再选 1×F，所以

$N=95D=80+15=16\times5+1\times F=5FH$

（2）除法（又叫除 16 取余法，仅适用于整数部分），除 16 取余法就是不断除以 16，记下余数，直到商为 0 为止。

例 1.7 求 N=95D 的十六进制数。

95/16=5　　余 15 (h_0) ↑
5/16=0　　余 5　(h_1) |

$N=95D=h_1h_0=5FH$

对于十进制数的小数部分除了降幂法也可采用乘法，即不断乘 16，并记下整数，而小数部分再乘 16，直到结果的小数部分为 0 为止。此处不再举例。

6. 二进制数和十六进制数的相互转换

由于十六进制数的基数 $16=2^4$，故一位十六进制数由四位二进制数组成，相互转换极为简单。

例 1.8 N=1011111.11(B)=01011111.1100(B)=5F.C(H)

注意到从二进制数转换到十六进制数时，二进制数的整数部分从最低位开始每 4 位一组，不足 4 位的，高位补 0 补足 4 位。小数部分从最高位开始每 4 位一组，不足 4 位的，低位补 0 补足 4 位。

1.2.2　二进制数和十六进制数的运算

1. 二进制数的运算规则

加法规则：0 + 0=0，0 + 1=1，1 + 1=0（进位 1）

乘法规则：0 × 0=0，0 × 1=0，1 × 1=1

2. 十六进制数的运算

十六进制数的加减运算只要遵循逢 16 进 1 规则即可，当然也可先把十六进制数转换为二进制数，运算后的结果再转换为十六进制数。

例 1.9

```
    43A5
+   5A34
--------
    9DD9
```

例 1.10

```
    5A34
-   43A5
--------
    168F
```

十六进制数的乘、除法运算可以先把十六进制数转换为十进制数，运算后的结果再转换为十六进制数。十六进制数的乘法也可以用十进制数的乘法规则计算，但结果用十六进制数表示。

例 1.11

```
    2A34
×   0025
--------
    D304
+ 5468
--------
  61984(H)
```

1.2.3　带符号数的补码表示

数有正数和负数，计算机中的数是用二进制来表示的，数的符号也用二进制来表示，所谓带符号数就是最高位是符号位，一般规定正数的符号位为 0，负数的符号位为 1。把一个数连同其符号在内数值化表示叫机器数，机器数的表示可以用不同的码制，常用的有原码、补码、反码。这里只介绍最常用的补码。

补码表示法中的正数用符号位+绝对值表示，即数的最高位为 0，其余部分为该数的绝对值。例如，用 8 位二进制来表示：

$[+1]_{补}$=00000001，$[+127]_{补}$=01111111，$[+0]_{补}$=00000000

负数用补码表示时，方法是对其正数各位取反，然后最低位加 1。我们把这种对二进制数取反加 1 的运算叫做求补运算。负数用补码表示时，其符号位必定为 1。

例 1.12　用 8 位二进制来表示，求$[-3]_{补}$。

先写出+3：　　0000 0011

各位取反为：　1111 1100

最低位加 1 为：1111 1101

$[-3]_{补}$=1111 1101，或用十六进制表示，$[-3]_{补}$=FDH。

读者也许会问，当用 16 位二进制来表示[-3]$_{补}$呢？其实只要在刚求出的[-3]$_{补}$的前面加上 8 个 1 就可以了，即[-3]$_{补}$=1111 1111 1111 1101，或[-3]$_{补}$=FFFDH。这叫符号扩展。对负数的符号扩展只需在前面补 1，对正数的符号扩展只需在前面补 0，符号扩展并没有改变数的大小，只是改变了位数。读者可自行验证。

我们已经知道对负数用补码表示时，方法是对其正数取反加 1，即作求补运算。其实这个方法是根据补码定义得出的。下面给出证明。

补码定义：

$$（X \geqslant 0 时）\quad [X]_{补}=符号+|X| \tag{1}$$

$$（X < 0 时）\quad [X]_{补}=2^n-|X| \tag{2}$$

现在我们把（2）式进一步改写：

$$[X]_{补}=2^n-|X| =(2^n-1-|X|)+1 \tag{3}$$

请注意（3）式右边括号中的$(2^n-1-|X|)$，就是对|X|取反码。再+1 就得到[X]$_{补}$。由此可见，求负数的补码，方法是对其正数取反加 1。

现在我们把（3）式进一步改写：

$$|X|=2^n-[X]_{补}=(2^n-1-[X]_{补})+1 \tag{4}$$

注意到上式右边的$(2^n-1-[X]_{补})+1$，就是对[X]$_{补}$再求补码，就得到|X|。由（3）式和（4）式说明了以下结论：

$$[X]_{补} \xleftrightarrow{\text{求补}} [-X]_{补}$$

例 1.13 依据补码定义写出以下各数的补码，以 8 位二进制表示。

[-1]$_{补}$=2^8-1=1 0000 0000-1= 1111 1111，直接由（2）式得到。

[-127]$_{补}$=2^8-127=（2^8-1-127）+1

=（1111 1111-0111 1111）+1

= 1000 0000+1

= 1000 0001

依据补码定义求一个数的补码表示，有些烦琐，用取反加 1 的规则更为简便。

例 1.14 识别以下各数的十进制值。

[a]$_{补}$=1111 1111，求补后为 0000 0001=[1]$_{补}$，所以，a=-1

[b]$_{补}$=1000 0000，求补后为 1000 0000=[128]$_{补}$，所以，b=-128

[c]$_{补}$=1000 0001，求补后为 0111 1111=[127]$_{补}$，所以，C=-127

前面我们都是以 8 位二进制数讨论，8 位二进制数可以表示 2^8=256 个数，当它们是补码表示的数时，所能表示的范围是$-128 \leqslant N \leqslant +127$。

1.2.4 补码的加法和减法

计算机中主要是用补码表示数据，因此我们应关注补码的加法和减法。

补码的加法规则是：

$$[X+Y]_{补}=[X]_{补}+[Y]_{补} \tag{5}$$

补码的减法规则是：

$$[X-Y]_{补}=[X]_{补}+[-Y]_{补} \quad (6)$$

这个规则说明了用 2 个数的补码相加就可以完成 2 个数的加减法,得到的还是补码。

下面给出证明:

需明确 X > 0，Y > 0，先看（5）式，由补码定义可知，$[X+Y]_{补}= X+Y =[X]_{补}+[Y]_{补}$。

（5）式得证。再看（6）式:

如果$[X-Y]\geq 0$，由补码定义可知，（6）式左边应有$[X-Y]_{补}=X-Y$，而（6）式右边$=X+(2^n-Y)=X-Y+2^n=X-Y$。

如果$[X-Y]<0$（或者说 Y > X），应有$[X-Y]_{补}=2^n-|X-Y|=2^n-(Y-X)=2^n-Y+X=[-Y]_{补}+[X]_{补}$。

（6）式得证。

下面给出例子说明补码的加法运算。

例 1.15　8 位补码的加法运算。

十进制	二进制
25	0001 1001
+（-32）	+ 1110 0000
-7	1111 1001
32	0010 0000
+（-25）	+ 1110 0111
7	0000 0111
	1↙

从例中可以看出补码的加法很简便，不必考虑数的正负，符号位参于运算即可，计算结果都是正确的。从最高有效位向高位的进位由于机器字长的限制而自动丢弃，但结果依然正确。同时这个进位被保存到机器中的标志寄存器中，其作用以后再说明。

1.2.5　无符号数的表示

如果要处理的数全是正数，保留符号位就没有必要，我们可以把最高有效位也作为数值，这样的数就叫无符号数。8 位二进制来表示的无符号数的表数范围是 0≤N≤255，16 位二进制来表示的无符号数的表数范围是 0≤N≤65535。在计算机中最常见的无符号数是表示内存单元的地址。例如：1100 0010B=C2H=194D，而不再表示一个负数。

1.2.6　字符的表示

除了数值以外，人们有时还需要用计算机处理字符或字符串。例如，从键盘输入或打印输出信息都是以字符方式进行的。字符包括以下内容。

字母：A，B，C，D……;

数字：0，1，2，3……;

专门符号：+，-，×，/，SP（space 空格）……;

非打印字符：CR（carriage return 回车），LF（line feed 换行）……。

这些字符必须采用二进制的编码方式，目前采用最常用的美国信息交换标准代码 ASCII（American Standard Code for Information Interchange）来表示。

这种代码用一个字节（8 位二进制码）来表示一个字符，其中低 7 位为字符的 ASCII 值，故

能表示 128 个符号和代码，最高位一般用作检测校验位。表 1.2 为部分常用字符的 7 位 ASCII 码表（十六进制表示）。

为了能表示更多的符号，将 7 位 ASCII 码扩充到 8 位，可以表示 256 个符号和代码，称为扩充的 ASCII 码。

表 1.2　　部分常用字符的 7 位 ASCII 码表（十六进制表示）

字符	ASCII	字符	ASCII	字符	ASCII
NUL	00	A	41	a	61
BEL	07	B	42	b	62
LF	0A	C	43	c	63
FF	0C	D	44	d	64
CR	0D	E	45	e	65
SP	20	F	46	f	66
#	23	G	47	g	67
$	24	H	48	h	68
%	25	I	49	i	69
0	30	J	4A	j	6A
1	31	K	4B	k	6B
2	32	L	4C	l	6C
3	33	M	4D	m	6D
4	34	N	4E	n	6E
5	35	O	4F	o	6F
6	36	P	50	p	70
7	37	Q	51	q	71
8	38	R	52	r	72
9	39	S	53	s	73
:	3A	T	54	t	74
;	3B	U	55	u	75
<	3C	V	56	v	76
=	3D	W	57	w	77
>	3E	X	58	x	78
?	3F	Y	59	y	79
@	40	Z	5A	z	7A

1.2.7　基本逻辑运算

（1）“与”运算（AND），又叫逻辑乘，可用符号·或∧来表示，只有当逻辑变量 A、B 都为 1 时，“与”运算的结果才为 1。

（2）“或”运算（OR），又叫逻辑加，可用符号+或∨来表示，只要变量 A、B 其中有一个为 1 时，“或”运算的结果就为 1。

（3）“异或”运算（XOR），可用符号∀来表示，只有当变量 A、B 中仅一个为 1 时，“异或”运算的结果才为 1。

（4）“非”运算（NOT），对变量 A 取反，即如果 A=1，则 $\overline{A}$ =0，反之亦然。

具体参见表 1.3。

逻辑运算都是按位操作，例如，X=0011，Y=1011，均为二进制数，则有：

X（AND）Y=0011，X（OR）Y=1011，X（XOR）Y=1000，（NOT）X=1100

表 1.3　　四种基本逻辑运算

A	B	AND	OR	XOR	NOT A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

1.3　计算机组织

1.3.1　计算机系统组成

汇编语言是面向机器的用符号表示的程序设计语言，所以使用汇编语言进行程序设计时，除了需要考虑求解问题的过程或者算法，安排数据在计算机内的存储格式，同时还要根据程序和算法的需要使用计算机内的资源。因此，作为一名汇编语言程序员，必须了解计算机的基本逻辑结构，了解有哪些可供使用的资源，以及如何使用，但无须了解其电子线路组成和电气特性。本节主要结合 80x86 系列微型计算机来介绍程序员需要掌握的计算机逻辑结构。

典型的计算机结构如图 1.1 所示。主要由微处理器芯片构成的中央处理机（CPU）、存储器（memory）和输入输出（I/O）子系统三大部分组成，用系统总线（bus）连接在一起。

（1）中央处理机（CPU，Central Process Unit）或叫微处理器（MPU，Micro Process Unit），主要包括运算器和控制器。运算器执行指令，控制器负责计算机的控制，负责从主存储器取指令，对指令进行译码，发出访问主存储器或 I/O 设备接口的控制信号，完成程序的要求。显然，CPU 是最核心的部件，指令都是在这里执行的。

（2）存储器，是计算机记忆部件，以二进制形式存放程序和数据。这里是指主存储器，简称主存，或叫内存储器，简称内存，称为 RAM。而把硬盘、光盘这些大容量存储器称为外部存储器，简称外存。

（3）输入输出（I/O）子系统包括大容量存储器（如硬盘）和其他外设，如显示器、键盘、打印机、鼠标等。

（4）系统总线连接 CPU、主存储器和 I/O 子系统三大部分，用以完成各部分的数据交换。系统总线包括数据总线、地址总线和控制总线。数据总线负责传送数据，地址总线负责指示主存地址或 I/O 接口地址，控制总线负责总线的动作，如时间、方向、状态。16 位微处理器数据总线的位数为 16 位（bit），表示一次可以并行传输和处理 16 位二进制数据。32 位微处理器数据总线的位数为 32 位（bit）。地址总线宽度的多少决定了可访问的内存容量的大小。部分机型的总线位数如表 1.4 所示。

表 1.4　　部分机型的总线位数

CPU	处理器字长	数据总线位数	地址总线位数	最大寻址空间
8086	16	16	20	1MB
80386/86486	32	32	32	4GB
Pentium2/p3/p4	32	64	36	64GB

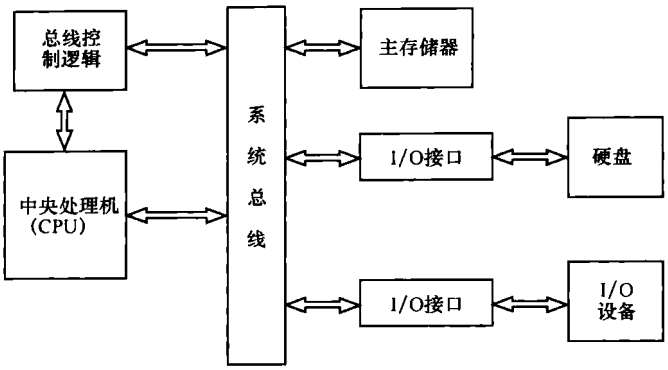

图 1.1　计算机结构

1.3.2　中央处理器（CPU）中的寄存器

现代微型计算机把运算器、控制器、寄存器和高速缓冲存储器集成在一块集成电路芯片中。寄存器相当于运算器中的高速存储单元，放置当前参与运算的操作数地址、数据、中间结果、处理机状态等。

作为汇编语言程序员，实际上是通过对寄存器的操作来实现对 CPU 的操作。也就是说这些寄存器是可编程的，因此我们只对寄存器感兴趣。

80x86 CPU 中的寄存器可以分为通用寄存器、段寄存器和专用寄存器三类。

在 80286 以前的 CPU 中，一条指令可运算的最大数据长度为 16 位二进制数（16 bit），因此我们也称这样的 CPU 是字长 16 位的 CPU。其中的寄存器的标准长度为 16 位，有些寄存器可以分拆作为 2 个 8 位寄存器使用。32 位的 80x86 微处理器的寄存器基本长度是 32 位，有些寄存器可以分拆作为 8 位、16 位寄存器使用。

这里需要指出的是，因为本书重点介绍 16 位机的指令系统和程序设计，而 32 位机对 16 位机实现了软硬件的完全兼容，在 16 位机上编制运行的程序在 32 位机上照样可以运行。所以我们重点介绍 16 位机的寄存器。80x86 的寄存器组如图 1.2 所示。

1. 数据寄存器

16 位的 80x86 处理器有 4 个 16 位的通用数据寄存器。它们的主要作用是存放数据，有时候也可以存放地址。

① AX：累加器，运算时较多使用这个寄存器，有些指令规定必须使用它。

② BX：基址寄存器，除了存放数据，它经常用来存放一段内存的起始偏移地址。

③ CX：计数寄存器，除了存放数据，它经常用来存放重复操作的次数。

④ DX：数据寄存器，除了存放数据，它有时存放 32 位数据的高 16 位。

上面的寄存器都可以拆分为两个 8 位寄存器使用，分别命名为 AH，AL，BH，BL，CH，CL，DH，DL。

32 位 80x86 处理器的 4 个数据寄存器扩展为 32 位，更名为 EAX，EBX，ECX，EDX，但仍然可以使用原有的 16 位和 8 位寄存器，且名称不变，如 AX，BX，CX，DX，AH，AL 等，对程序员来说，似乎是增加了 4 个 32 位的数据寄存器。

2. 地址寄存器

16 位的 80x86 处理器有 4 个 16 位的通用地址寄存器。它们的主要作用是存放数据的所在偏