

80X86

汇编语言程序设计教程

杨季文 等编著
钱培德 审



清华大学出版社

<http://www.tup.tsinghua.edu.cn>

80x86 汇编语言程序设计教程

杨季文等 编著

钱培德 审

清华大学出版社

(京)新登字 158 号

内 容 提 要

本书分为三部分。第一部分是基础部分,以 8086/8088 为背景,以 DOS 和 PC 兼容机为软硬件平台,以 MASM 和 TASM 为汇编器,介绍汇编语言的有关概念,讲解汇编语言程序设计技术。第二部分是提高部分,以 80386 为背景,以新一代微处理器 Pentium 为目标,细致和通俗地介绍了保护方式下的有关概念,系统和详细地讲解了保护方式下的编程技术,真实和生动地展示了保护方式下的编程细节。第三部分是上机实验指导。

本书的第一部分适合初学者,可作为学习汇编语言程序设计的教材。本书的第二部分适合已基本掌握 8086/8088 汇编语言的程序员,可作为学习保护方式编程技术的教材或参考书,也可作为其他人员了解高档微处理器和保护方式编程技术的参考书,还可作为程序员透彻地了解 Windows 程序设计技术的参考书。

版权所有,翻印必究。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

图书在版编目(CIP)数据

80x86 汇编语言程序设计教程/杨季文等编著. —北京:清华大学出版社,1998.4

ISBN 7-302-02901-6

I. 80… II. 杨… III. 汇编语言-程序设计-教材 IV. TP312

中国版本图书馆 CIP 数据核字(98)第 07169 号

出版:清华大学出版社(北京清华大学校内,邮编 100084)

因特网地址:www.tup.tsinghua.edu.cn

印刷:丰华印刷厂

发行:新华书店总店北京科技发行所

开本:787×1092 1/16 印张:38 字数:902 千字

版次:1998 年 6 月第 1 版 1998 年 6 月第 1 次印刷

书号:ISBN 7-302-02901-6/TP·1535

印数:0001~6000

定价:39.00 元

前 言

汇编语言面向机器,只有它能够为程序员提供最直接操纵机器硬件系统的途径,利用它可以编写出在“时间”和“空间”两个方面最具效率的程序。

“汇编语言程序设计”是计算机各专业的一门重要基础课程,是必修的核心课程之一,是“操作系统”和“微机原理与接口技术”等其他核心课程必要的先修课。该课程对于训练学生掌握程序设计技术,熟悉上机操作和程序调试技术都有重要作用。此外,“汇编语言程序设计”也是其他相关专业的必修或选修课。

目前,国内最广泛使用的 PC 系列机(包括兼容机),都以 Intel 的 80x86 系列微处理器或者兼容的微处理器为 CPU。在 Intel 的 80x86 家族中,16 位的 8086/8088 是基础,实现了以分段方式管理存储器;32 位的 80386 是高档微处理器的里程碑,实现了支持多任务的保护工作方式;基于 MMX 技术的 Pentium 是新一代的微处理器,实现了对多媒体处理的支持。本书以 8086/8088 为基础,以 80386 为重点,面向 Pentium 等新一代微处理器,讲解汇编语言程序设计的一般概念、基本技术和常用技巧,介绍宏和模块化程序设计的技术方法,讲解保护方式编程的相关概念、编程技术及实现细节。

本书分三个部分,共 12 章。第一部分是基础部分。第 1 章介绍汇编语言的特点和其他基本概念。第 2、第 3 章以 8086/8088 为背景,简要介绍 8086/8088 寻址方式、指令系统和汇编语言的常用伪指令语句后,讲解如何利用汇编语言实现程序的基本结构。第 4 章详细讲解了子程序的设计和如何调用 DOS 提供的子程序。第 5 章以 PC 及其兼容机为硬件平台,介绍输入/输出和中断等概念,讲解如何利用汇编语言编写 BIOS 程序和调用 BIOS 程序。第 6 章以 DOS 为软件平台,讲解如何利用汇编语言编写小型应用程序。第 7 章以 MASM 和 TASM 为汇编器,介绍宏和条件汇编等汇编语言的高级技术。第 8 章介绍模块化程序设计技术以及与高级语言的混合编程。第二部分是提高部分。第 9 章介绍实方式下的 80386 及其编程。第 10 章讲解保护方式下的 80386 及其编程,该章内容十分丰富。第 11 章介绍 80486 和 Pentium 程序设计基础。第三部分是上机实验指导,安排为第 12 章,应在上机实验前先阅读了解该章内容。

本书的第一部分适合初学者,可作为学习汇编语言程序设计的教材。本书的第二部分适合已基本掌握 8086/8088 汇编语言的程序员,可作为学习保护方式编程技术的教材或参考书,也可作为其他人员了解高档微处理器和保护方式编程技术细节的参考书,还可作为程序员透彻地了解 Windows 程序设计技术的参考书。

杨季文编写第 1 章、第 3 章、第 4 章至第 7 章、第 9 章至第 11 章,朱巧明编写第 2 章,吕强编写第 8 章,曹培培编写第 12 章。由杨季文最后统一定稿。

本书从初稿到定稿的全过程都始终得到了指导老师钱培德教授的热情关心和大力支持,承蒙他审阅了全书,特在此表示衷心感谢。本书在编著过程中得到了同事鲁征山、陈时

飏、李培峰和李廷彦等同志的帮助,还得到了赵雷和朱楠灏等同志的帮助,在此表示感谢。
刘文杰和许晨等同志在讲课时使用过本书的初稿,并提出了宝贵意见,在此表示感谢。
书中不妥和谬误之处难免,恳请读者批评指正。

编者
1998年2月

目 录

第一部分 基础部分

第 1 章 绪论	1
1.1 汇编语言概述	1
1.1.1 汇编语言	1
1.1.2 汇编语言的特点	2
1.1.3 恰当地使用汇编语言	3
1.2 数据的表示和类型	4
1.2.1 数值数据的表示	4
1.2.2 非数值数据的表示	6
1.2.3 基本数据类型	7
1.3 Intel 系列 CPU 简介	8
1.3.1 8 位微处理器	8
1.3.2 16 位微处理器	9
1.3.3 32 位微处理器	11
1.3.4 Pentium 和 Pentium Pro	13
1.4 习题	14
第 2 章 8086/8088 寻址方式和指令系统	15
2.1 8086/8088 寄存器组	15
2.1.1 8086/8088 CPU 寄存器组	15
2.1.2 标志寄存器	17
2.2 存储器分段和地址的形成	19
2.2.1 存储单元的地址和内容	19
2.2.2 存储器的分段	20
2.2.3 物理地址的形成	20
2.2.4 段寄存器的引用	21
2.3 8086/8088 的寻址方式	22
2.3.1 立即寻址方式	23
2.3.2 寄存器寻址方式	23
2.3.3 直接寻址方式	23
2.3.4 寄存器间接寻址方式	24
2.3.5 寄存器相对寻址方式	25
2.3.6 基址加变址寻址方式	26
2.3.7 相对基址加变址寻址方式	27

2.4	8086/8088 指令系统	28
2.4.1	指令集说明	28
2.4.2	数据传送指令	29
2.4.3	堆栈操作指令	32
2.4.4	标志操作指令	34
2.4.5	加减运算指令	36
2.4.6	乘除运算指令	41
2.4.7	逻辑运算和移位指令	44
2.4.8	转移指令	51
2.5	习题	58
第 3 章	汇编语言及其程序设计初步	63
3.1	汇编语言的语句	63
3.1.1	语句的种类和格式	63
3.1.2	数值表达式	64
3.1.3	地址表达式	67
3.2	变量和标号	67
3.2.1	数据定义语句	67
3.2.2	变量和标号	70
3.3	常用伪指令语句和源程序组织	73
3.3.1	符号定义语句	74
3.3.2	段定义语句	75
3.3.3	汇编语言源程序的组织	79
3.4	顺序程序设计	81
3.4.1	顺序程序举例	81
3.4.2	简单查表法代码转换	83
3.4.3	查表法求函数值	85
3.5	分支程序设计	86
3.5.1	分支程序举例	86
3.5.2	利用地址表实现多向分支	91
3.6	循环程序设计	94
3.6.1	循环程序举例	94
3.6.2	多重循环程序举例	103
3.7	习题	106
第 4 章	子程序设计和 DOS 功能调用	110
4.1	子程序设计	110
4.1.1	过程调用和返回指令	110
4.1.2	过程定义语句	115
4.1.3	子程序举例	116

4.1.4	子程序说明信息	118
4.1.5	寄存器的保护与恢复	119
4.2	主程序与子程序间的参数传递	121
4.2.1	利用寄存器传递参数	121
4.2.2	利用约定存储单元传递参数	123
4.2.3	利用堆栈传递参数	125
4.2.4	利用 CALL 后续区传递参数	127
4.3	DOS 功能调用及应用	129
4.3.1	DOS 功能调用概述	129
4.3.2	基本 I/O 功能调用	130
4.3.3	应用举例	132
4.4	磁盘文件管理及应用	141
4.4.1	DOS 磁盘文件管理功能调用	141
4.4.2	应用举例	143
4.5	子程序的递归和重入	150
4.5.1	递归子程序	150
4.5.2	可重入子程序	151
4.6	习题	152
第 5 章	输入输出与中断	155
5.1	输入和输出的基本概念	155
5.1.1	I/O 端口地址和 I/O 指令	155
5.1.2	数据传送方式	156
5.1.3	存取 RT/CMOS RAM	157
5.2	查询方式传送数据	160
5.2.1	查询传送方式	160
5.2.2	读实时钟	161
5.2.3	查询方式打印输出	162
5.3	中断	164
5.3.1	中断和中断传送方式	164
5.3.2	中断向量表	165
5.3.3	中断响应过程	168
5.3.4	外部中断	168
5.3.5	内部中断	170
5.3.6	中断优先级和中断嵌套	172
5.3.7	中断处理程序的设计	173
5.4	基本输入输出系统 BIOS	174
5.4.1	基本输入输出系统 BIOS 概述	174
5.4.2	键盘输入	175

5.4.3	显示输出	178
5.4.4	打印输出	188
5.5	软中断处理程序举例	191
5.5.1	打印 I/O 程序	191
5.5.2	时钟显示程序	194
5.6	习题	197
第 6 章	简单应用程序的设计	200
6.1	字符串处理	200
6.1.1	字符串操作指令	200
6.1.2	重复前缀	205
6.1.3	字符串操作举例	208
6.2	十进制数算术运算调整指令及应用	215
6.2.1	组合 BCD 码的算术运算调整指令	215
6.2.2	未组合 BCD 码的算术运算调整指令	216
6.2.3	应用举例	218
6.3	DOS 程序段前缀和特殊情况处理程序	224
6.3.1	DOS 程序段前缀 PSP	224
6.3.2	对 Ctrl+C 键和 Ctrl+Break 键的处理	228
6.4	TSR 程序设计举例	234
6.4.1	驻留的时钟显示程序	234
6.4.2	热键激活的 TSR 程序	236
6.5	习题	238
第 7 章	高级汇编语言技术	241
7.1	结构和记录	241
7.1.1	结构	241
7.1.2	记录	246
7.2	宏	249
7.2.1	宏指令的定义和使用	250
7.2.2	宏指令的用途	251
7.2.3	宏指令中参数的使用	253
7.2.4	特殊的宏运算符	254
7.2.5	宏与子程序的区别	256
7.2.6	与宏有关的伪指令	256
7.2.7	宏定义的嵌套	258
7.3	重复汇编	260
7.3.1	伪指令 REPT	260
7.3.2	伪指令 IRP	261
7.3.3	伪指令 IRPC	262

7.4	条件汇编	262
7.4.1	条件汇编伪指令	263
7.4.2	条件汇编与宏结合	265
7.5	源程序的结合	268
7.5.1	源程序的结合	268
7.5.2	宏库的使用	271
7.6	习题	273
第8章	模块化程序设计技术	275
8.1	段的完整定义	275
8.1.1	完整的段定义	275
8.1.2	关于堆栈段的说明	280
8.1.3	段组的说明和使用	281
8.2	段的简化定义	285
8.2.1	存储模型说明伪指令	285
8.2.2	简化的段定义伪指令	285
8.2.3	存储模型说明伪指令的隐含动作	288
8.3	模块间的通信	289
8.3.1	伪指令 PUBLIC 和伪指令 EXTRN	289
8.3.2	模块间的转移	291
8.3.3	模块间的信息传递	293
8.4	子程序库	298
8.4.1	子程序库	298
8.4.2	建立子程序库	298
8.4.3	使用举例	301
8.5	编写供 Turbo C 调用的函数	303
8.5.1	汇编格式的编译结果	303
8.5.2	汇编模块应该遵守的约定	306
8.5.3	参数传递和寄存器保护	307
8.5.4	举例	309
8.6	习题	313

第二部分 提高部分

第9章	80386 程序设计基础	314
9.1	80386 寄存器	314
9.1.1	通用寄存器	315
9.1.2	段寄存器	315
9.1.3	指令指针和标志寄存器	316
9.2	80386 存储器寻址	316

9.2.1	存储器寻址基本概念	317
9.2.2	灵活的存储器寻址方式	318
9.2.3	支持各种数据结构	320
9.3	80386 指令集	320
9.3.1	数据传送指令	321
9.3.2	算术运算指令	326
9.3.3	逻辑运算和移位指令	327
9.3.4	控制转移指令	330
9.3.5	串操作指令	334
9.3.6	高级语言支持指令	337
9.3.7	条件字节设置指令	340
9.3.8	位操作指令	342
9.3.9	处理器控制指令	345
9.4	实方式下的程序设计	346
9.4.1	说明	346
9.4.2	实例	348
9.5	习题	358
第 10 章	保护方式下的 80386 及其编程	361
10.1	保护方式简述	361
10.1.1	存储管理机制	361
10.1.2	保护机制	363
10.2	分段管理机制	364
10.2.1	段定义和虚拟地址到线性地址转换	364
10.2.2	存储段描述符	366
10.2.3	全局和局部描述符表	369
10.2.4	段选择子	370
10.2.5	段描述符高速缓冲寄存器	371
10.3	80386 控制寄存器和系统地址寄存器	372
10.3.1	控制寄存器	372
10.3.2	系统地址寄存器	374
10.4	实方式与保护方式切换实例	375
10.4.1	演示实方式和保护方式切换的实例(实例一)	376
10.4.2	演示 32 位代码段和 16 位代码段切换的实例(实例二)	382
10.5	任务状态段和控制门	389
10.5.1	系统段描述符	389
10.5.2	门描述符	390
10.5.3	任务状态段	392
10.6	控制转移	395

10.6.1	任务内无特权级变换的转移	395
10.6.2	演示任务内无特权级变换转移的实例(实例三)	397
10.6.3	任务内不同特权级的变换	408
10.6.4	演示任务内特权级变换的实例(实例四)	410
10.6.5	任务切换	420
10.6.6	演示任务切换的实例(实例五)	422
10.7	80386 的中断和异常	431
10.7.1	80386 的中断和异常	431
10.7.2	异常类型	433
10.7.3	中断和异常的转移方法	437
10.7.4	演示中断处理的实例(实例六)	442
10.7.5	演示异常处理的实例(实例七)	450
10.7.6	各种转移途径小结	465
10.8	操作系统类指令	466
10.8.1	实方式和任何特权级下可执行的指令	467
10.8.2	实方式及特权级 0 下可执行的指令	468
10.8.3	只能在保护方式下执行的指令	470
10.8.4	显示关键寄存器内容的实例(实例八)	473
10.8.5	特权指令	477
10.9	输入/输出保护	477
10.9.1	输入/输出保护	477
10.9.2	重要标志保护	481
10.9.3	演示输入/输出保护的实例(实例九)	481
10.10	分页管理机制	492
10.10.1	存储器分页管理机制	492
10.10.2	线性地址到物理地址的转换	493
10.10.3	页级保护和虚拟存储器支持	496
10.10.4	页异常	498
10.10.5	演示分页机制的实例(实例十)	499
10.11	虚拟 8086 方式	506
10.11.1	V86 方式	506
10.11.2	进入和离开 V86 方式	506
10.11.3	演示进入和离开 V86 方式的实例(实例十一)	510
10.11.4	V86 方式下的敏感指令	522
10.12	习题	523
第 11 章	80486 及 Pentium 程序设计基础	525
11.1	80486 程序设计基础	525
11.1.1	寄存器	525

11.1.2	指令系统	527
11.1.3	片上超高速缓存	530
11.2	80486 对调试的支持	535
11.2.1	调试寄存器	535
11.2.2	演示调试故障/陷阱的实例	538
11.3	Pentium 程序设计基础	543
11.3.1	寄存器	543
11.3.2	指令系统	545
11.3.3	处理器的识别	548
11.3.4	片上超高速缓存	553
11.4	基于 Pentium 的程序优化技术	557
11.4.1	流水线优化技术	557
11.4.2	分支优化技术	564
11.4.3	超高速缓存优化技术	567
11.5	习题	569

第三部分 上机实验指导

第 12 章	实验指导	572
12.1	实验的一般步骤	572
12.2	汇编器和连接器的使用	574
12.2.1	MASM 的使用	574
12.2.2	LINK 的使用	575
12.2.3	TASM 的使用	577
12.2.4	TLINK 的使用	578
12.3	调试器 DEBUG 的使用	578
12.3.1	启动和退出 DEBUG	579
12.3.2	命令一览	580
12.3.3	利用 DEBUG 调试程序	582
12.4	Turbo Debugger 的使用	587
12.4.1	启动和退出 TD	587
12.4.2	利用 TD 调试汇编程序	588
参考文献		592
附录	Pentium 指令与标志参考表	593

第一部分 基础部分

第 1 章 绪 论

本章先介绍汇编语言的一些基本概念,然后介绍数据的表示和类型,最后简单介绍了 Intel 的 x86 家族历代微处理。

1.1 汇编语言概述

尽管在使用汇编语言进行程序设计之前,完全理解汇编语言的特点有困难,但了解汇编语言的特点对学习汇编语言程序设计是有益的。本节先说明汇编语言的内容,再介绍汇编语言的特点和使用汇编语言的场合。

1.1.1 汇编语言

1. 机器语言

CPU 能直接识别并遵照执行的指令称为机器指令。机器指令在形式上表现为二进制编码。机器指令一般由操作码和操作数两部分构成,操作码在前,操作数在后。操作码指出要进行的操作或运算,如加、减、传送等。操作数指出参与操作或运算的对象,也指出操作或运算结果存放的位置,如 CPU 的寄存器、存储单元和数据等。

机器指令与 CPU 有着密切的关系。通常,CPU 的种类不同,对应的机器指令也就不同。不同型号 CPU 的指令集往往有较大的差异。但同一个系列 CPU 的指令集常常具有良好的向上兼容性,也即下一代 CPU 的指令集是上一代 CPU 指令集的超集。例如,Intel 80386 指令集包含了 8086 指令集。

机器语言是用二进制编码的机器指令的集合及一组使用机器指令的规则。它是 CPU 能直接识别的唯一语言。只有用机器语言描述的程序,CPU 才能直接执行。用机器语言描述的程序称为目的程序或目标程序。

为了阅读和书写方便,常用十六进制形式或八进制形式表示二进制编码。例如,我们用 Intel 8086 指令写一个两数相加的程序片段。具体要求是把偏移 2200H 存储单元中的数与偏移 2201H 存储单元中的数相加,将它们的和送入偏移 2202H 存储单元。完成这一工作的程序片段包含三条机器指令,用十六进制形式表示如下:

```
A0 00 20  
02 06 01 20
```

几乎没有人能直接看出该程序片段的功能,原因是程序员难以掌握机器语言。因此,程序员难以用机器语言写程序,更难写出健壮的程序;用机器语言编制出的程序也不易为人们理解、记忆和交流。所以,只是在早期或不得已时才用机器语言写程序,现在几乎没有人用机器语言写程序了。机器语言有如下缺点:机器语言不能用人们熟悉的形式来描述计算机要执行的任务;用机器语言编写程序十分繁难,极易出错;一旦有错,也很难发现,也即调试困难。

2. 汇编语言

为了克服机器语言的上述缺点,人们采用便于记忆、并能描述指令功能的符号来表示指令的操作码。这些符号被称为指令助记符。助记符一般是说明指令功能的英语词汇或者词汇的缩写。同时也用符号表示操作数,如 CPU 的寄存器、存储单元地址等。

用指令助记符、地址符号等符号表示的指令称为汇编格式指令。

汇编语言是汇编格式指令、伪指令的集合及其表示、使用这些指令的一组规则。伪指令的概念留待以后介绍。用汇编语言书写的程序称为汇编语言程序,或称为汇编语言源程序,或简称为源程序。

利用汇编语言,上述两数相加的程序片段可表示如下:

```
MOV    AL,VAR1
ADD    AL,VAR2
MOV    VAR3,AL
```

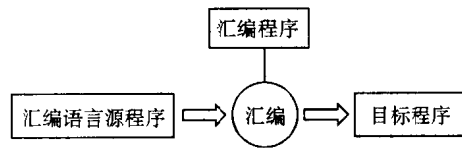


图 1.1 汇编过程示意图

显然,汇编格式指令比二进制编码的机器指令更容易掌握得多,用汇编语言编写的程序要比用机器语言编写的程序容易理解、调试和维护

3. 汇编程序

由于 CPU 能直接识别的唯一语言是机器语言,所以用汇编语言编写的源程序必须被翻译成用机器语言表示的目标程序后才能由 CPU 执行。把汇编语言源程序翻译成目标程序的过程称为汇编。完成汇编任务的程序叫做汇编程序。汇编过程如图 1.1 所示。

1.1.2 汇编语言的特点

由于汇编语言使用指令助记符和符号地址,所以它要比机器语言容易掌握得多。与高级语言相比较,汇编语言有如下特点。

1. 汇编语言与机器关系密切

因为汇编格式指令是机器指令的符号表示,所以汇编格式指令与机器有着密切的关系,因此汇编语言也与机器有着密切的关系,确切地说汇编语言与机器所带的 CPU 有着十分密切的关系。对于各种不同类型的 CPU,要使用各种不同的汇编语言。于是,对于各种不同类型的 CPU,也就有各种不同的汇编程序。

由于汇编语言与机器关系十分密切,所以汇编语言源程序与高级语言源程序相比,它

的通用性和可移植性要差得多。但通过汇编语言可最直接和最有效地控制机器,这常常是大多数高级语言难以做到的。

2. 汇编语言程序效率高

用汇编语言编写的源程序在汇编后所得的目标程序效率高。这种目标程序的高效率反映在时间和空间两个方面:其一是运行速度快;其二是目标程序短。在采用相同算法的前提下,任何高级语言程序在这两方面的效率都不如汇编语言程序,许多情况下更是远远不及。

汇编语言程序能获得“时空”高效率的主要原因是:构成汇编语言主体的汇编格式指令是机器指令的符号表示,每一条汇编格式指令都是所对应的某条机器指令的“化身”;另一个重要原因是汇编语言程序能直接并充分利用机器硬件系统的许多特性。高级语言程序在上述两点上要逊色得多。

3. 编写汇编语言源程序繁琐

编写汇编语言源程序要比编写高级语言源程序繁琐得多。汇编语言是面向机器的语言,高级语言是面向过程或面向目标、对象的语言。如下两点突出表现了汇编语言的这一特性:

作为机器指令符号化的每一条汇编格式指令所能完成的操作极为有限。例如,Z80 指令集中没有乘法指令,8086 指令集中没有能够同时完成两次算术运算的指令。

程序员在利用汇编语言编写程序时,必须考虑包括寄存器、存储单元和寻址方式在内的几乎所有细节问题。例如:指令执行对标志的影响,堆栈设置的位置等。在使用高级语言编写程序时,程序员不会遇到这些琐碎却重要的问题。

4. 汇编语言程序调试困难

调试汇编语言程序往往要比调试高级语言程序困难。汇编格式指令的功能有限和程序员要注意太多的细节问题是造成这种困难的两个客观原因;汇编语言提供给了程序员最大的“舞台”,而程序员往往为了追求“时空”上的高效率而不顾程序的结构,这是造成调试困难的主观原因。

1.1.3 恰当地使用汇编语言

1. 汇编语言的优缺点

为了恰当地使用汇编语言,我们先明确一下它的优缺点。

汇编语言的主要优点是利用它可能编写出在“时空”两个方面最有效率的程序。另外,通过它可最直接和最有效地操纵机器硬件系统。

汇编语言的主要缺点是它面向机器,与机器关系密切,它要求程序员比较熟悉机器硬件系统,要考虑许多细节问题,最终导致程序员编写程序繁琐;调试程序困难;维护、交流和移植程序更困难。

正是由于汇编语言与机器关系密切,才使汇编语言具有其他高级语言所不具备的上述优点和缺点。为了利用汇编语言的优点,必须付出相应的代价。但汇编语言的每一个优点常常闪耀出诱人的光芒,使人们勇敢地面对它的缺点。

2. 使用汇编语言的场合

根据汇编语言的优缺点,我们要恰当地使用汇编语言,即尽可能地“扬长避短”。是否利用汇编语言编写程序,要看具体的应用场合,要充分考虑到软件的开发时间和软件的质量等诸多方面的因素。我们认为下列应用场合,可考虑使用汇编语言编写程序。

(1) 对软件的执行时间或存储容量有较高要求的场合。例如:系统程序的关键核心,智能化仪器仪表的控制系统,实时控制系统等。

(2) 需要提高大型软件性能的场合。通常把大型软件中执行频率高的子程序(过程)用汇编语言编写,然后把它们与其他程序一起连接。

(3) 软件与硬件关系密切,软件要有直接和有效控制硬件的场合。如设备驱动程序等。

(4) 没有合适的高级语言的场合。

3. 适度地追求“时空”效率

在用汇编语言编写程序时,追求“时空”效率要适度。在编写汇编语言程序时,我们要尽量利用最恰当的指令,以便节约一个字节或节省几个机器周期。但时至今日,计算机硬件系统的整体性能已极大地提高,所以,除非不得已,不要为节约少量字节或机器周期而影响程序的结构性、健壮性和可读性等。要在确保汇编语言程序上述性能良好的前提下追求“时空”性能。

1.2 数据的表示和类型

熟悉数据在计算机内的表示形式是掌握汇编语言程序设计的关键之一。本节简单介绍数据的表示形式和类型。

计算机中存储信息的最小单位称为位,在绝大多数系统中它只能表示两种状态。这两种状态可分别代表0和1。计算机系统内部采用二进制表示数值数据,也采用二进制编码表示非数值数据和指令,其主要原因就在于此。

1.2.1 数值数据的表示

所谓数值数据就是数。这里仅介绍定点整数的有关内容。

1. 数的二进制表示

尽管日常生活中大多采用十进制计数,但在计算机内,数却大多采用二进制表示。某个二进制数 $b_n b_{n-1} \cdots b_2 b_1 b_0$ 所表示的数值用十进制数来衡量时,可利用如下按权相加的方法计算得到:

$$b_n 2^n + b_{n-1} 2^{n-1} + \cdots + b_2 2^2 + b_1 2^1 + b_0 2^0$$

在书写时,为了与十进制数相区别,通常在二进制数后加一个字母B。

2. 有符号数的补码表示

为了方便地表示负数和容易地实现减法操作,有符号数采用补码形式表示。所以,有符号数二进制表示的最高位是符号位,0表示正数,1表示负数。正数数值的补码形式用二进制表示。为得到一个负数数值的补码形式,方法可以是先得出该负数所对应正数的二进