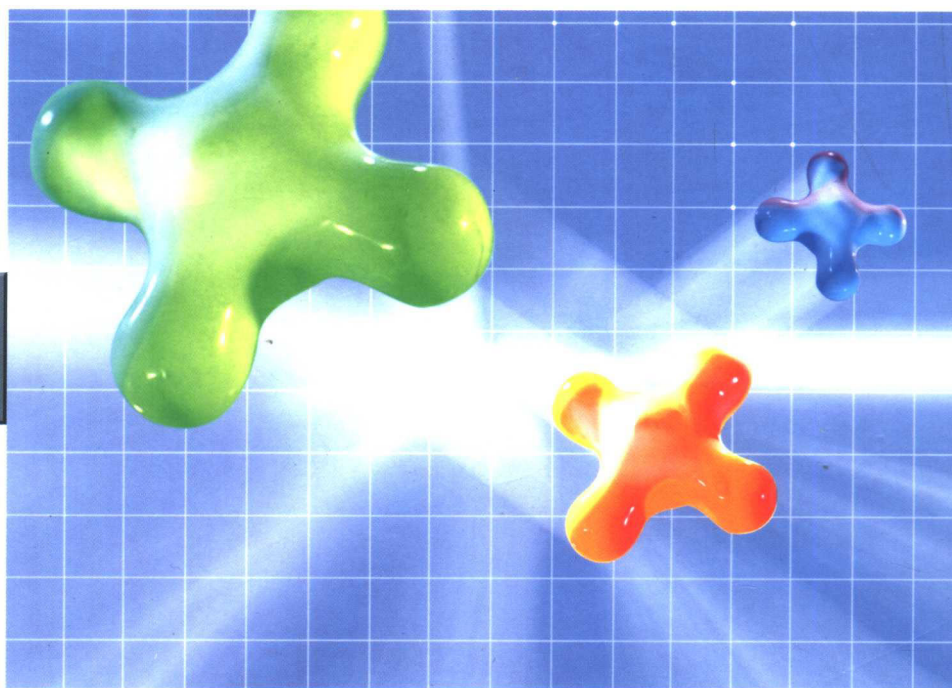


21世纪高等院校应用型规划教材

汇编语言 与接口技术



提供电子教案



叶继华 主编

甘登文 主审



机械工业出版社
CHINA MACHINE PRESS



21 世纪高等院校应用型规划教材

汇编语言与接口技术

叶继华 主编

罗贤海 周琪云 杨志文

高明华 王懿华 王知源

编著

甘登文 主审



机械工业出版社

本书以 Intel 8086 为主, 兼顾 Intel 80x86 系列, 系统介绍了 Intel 微处理器的结构和工作原理, 阐述了汇编语言程序设计的原理和方法、计算机接口技术的基本知识和编程技术。主要内容包括: Intel 80x86 系列微处理器的结构与原理、寻址方式和指令系统、基于 MASM5.1 和 MASM6.1 的 80x86 汇编语言程序设计、接口技术的基本知识、可编程接口技术。

本书结构合理、层次分明、逻辑严密、内容丰富、深入浅出, 涵盖了汇编语言程序设计的主要知识和接口技术的应用。本书可作为计算机及相关专业的本科、专科生教材, 也可作为工程技术人员的参考书。

图书在版编目 (CIP) 数据

汇编语言与接口技术 / 叶继华主编. —北京: 机械工业出版社, 2005.6
(21 世纪高等院校应用型规划教材)

ISBN 7-111-16805-4

I. 汇... II. 叶... III. ①汇编语言—程序设计—高等学校—教材②微处理器—接口—高等学校—教材 IV. TP3

中国版本图书馆 CIP 数据核字 (2005) 第 068629 号

机械工业出版社 (北京市百万庄大街 22 号 邮政编码 100037)

策 划: 胡毓坚

责任编辑: 蔡 岩

责任印制: 石 冉

三河市宏达印刷有限公司印刷 · 新华书店北京发行所发行

2005 年 8 月第 1 版 · 第 1 次印刷

787mm × 1092mm 1/16 · 19 印张 · 471 千字

0001—5000 册

定价: 27.00 元

凡购本图书, 如有缺页、倒页、脱页, 由本社发行部调换
本社购书热线电话 (010) 68326294
封面无防伪标均为盗版

出版说明

进入信息时代,我国高等教育面临的情况发生了巨大变化。信息技术日新月异,使得与其相关的课程知识结构更新迅速。由于社会对应用型人才的需求日趋强烈,高校也越来越注重对学生实践能力的培养。大多数高校的上机环境、教师的业务水平和工作条件都得到了明显改善,为教学模式、方法与手段的改革提供了必备的条件。多媒体教室的建设、学生上机时数的增加,实验室建设这一系列措施对教材的建设提出了新的要求。

为了切实体现教育思想和教育观念的转变,依据高等院校教学内容、教学方法和教学手段的现状,机械工业出版社推出了这套“21世纪高等院校应用型规划教材”。

本教材系列以建设“一体化设计、多种媒体有机结合的立体化教材”为宗旨,其目标是:建设一批符合应用型人才培养目标的、适合应用型人才培养模式的系列精品教材。本系列教材的编写者均为相关课程的一线主讲教师,教材内容注重理论与实际应用相结合,其中大力补充新知识、新技术、新工艺、新成果,非常适合各类高等院校、高等职业学校的教学。

为方便老师授课,本套教材为主干课程配备了电子教案、实验指导、习题解答等相关辅助内容。

机械工业出版社

前 言

汇编语言相对于高级语言来说,要求程序设计人员更深入地了解硬件结构,编程与调试过程也较繁琐,但它是计算机能提供给用户的最快而又最有效的语言,也是能够利用计算机所有硬件特性并能直接控制硬件的一种语言。它可以充分发挥机器系统的特性,达到最佳的时间和空间运行效率。汇编语言尤其适用于软件与硬件关系密切、软件需要直接和有效控制硬件的场合,如设备控制驱动程序、接口技术等,是高校计算机专业必修的核心课程之一。计算机接口技术是计算机专业的专业课程,也是自动控制、通信等专业的必修课程,作为计算机应用的重要方面,接口技术大量采用可编程接口芯片,其所用的编程语言一般是汇编语言。

在教学过程中,如果能够将“汇编语言”和“接口技术”这两门课程有效地结合起来,不仅能使学生加深对汇编语言的理解和掌握,而且能更好地掌握接口电路的应用编程,强化学生的实践能力。实践证明,通过“汇编语言和接口技术”的集成教学,能够达到这个目的。

汇编语言课程集硬件、软件技术于一体,随着硬件和软件技术的发展,以及计算机应用领域广度和深度的拓展,相关的理论与硬件都处于更新状况,并已有了集成的编程调试环境等,因此必须重视有关教材中汇编语言内容的更新,本书基于 MASM5.0 和 MASM6.11 对汇编语言的编程进行了介绍。

考虑到国内广泛使用的微型计算机都使用 Intel 的 80x86/Pentium 系列微处理器或者与其兼容的微处理器,所以本书以 80x86/Pentium 系列微处理器为基础。虽然主流微处理器已经从 8086、80286、80386、80486 发展到了 Pentium,但在 80x86 家族中 8086/8088 的指令集是最基本的,其他指令都是对 8086/8088 的指令集的扩充,因此仍然应把 8086/8088 汇编语言作为本课程教学的基本内容,在国外著名教材中也是如此处理的,但由于 Windows 已经成为主流的操作系统平台,只有掌握 80386、80486、Pentium 的结构及其保护模式的原理,才可能对基于 Windows 的系统作汇编语言级的分析、维护和开发,并有利于操作系统等后续课程的学习,所以本书中加入了这方面的内容。

全书共分 10 章,第 1 章介绍了微型计算机的有关特点和发展情况,以及计算机所使用的语言,着重阐述了计算机内的数据表示形式和运算。第 2 章主要介绍 Intel 8086 微处理器的内部结构、最大/最小工作模式及其存储器地址的形成,在此基础上,进一步阐述 Intel 80x86 微处理器系列概况、80386 微处理器的内部结构以及 Intel 80386CPU 的实地址、虚地址以及虚拟 8086 三种方式如何对存储器进行访问,为后面章节的学习进行必要准备。第 3 章着重介绍 Intel 8086/8088 的寻址方式和指令系统。第 4 章介绍了 MASM 宏汇编程序所支持的各种汇编语言知识,主要叙述了 MASM 汇编程序所支持的多种伪指令,以及组成指令语句、伪指令语句等的格式和规则要求,结合 MASM5.1 编程环境介绍了汇编语言程序的编写和调试过程。最后还介绍了在程序的设计中有时会用到的 DOS 和 BIOS 功能调用。第 5 章主要是综合运用前面的知识,结合程序实例,介绍汇编语言程序设计的方法和技术。第 6 章前面部分介绍了结构和联合、记录等数据类型,宏和重复块等操作,以及汇编语言与高级语言的连接,后面部分着重介绍了 80x86 的编程技术。第 7 章主要介绍了微机接口的功能、组成、分类,以及输入/输

出设备的数据传送方式。第 8 章主要介绍中断的基本概念、管理、实现、应用以及 DMA 的原理、DMA 控制器及其的编程。第 9 章主要介绍可编程并行接口 8255A 并行接口芯片和 8251A 串行接口芯片的概念、功能、内部结构和工作方式，以及可编程原理。第 10 章主要介绍了定时/计数器的相关概念，结合 8253 定时/计数器，阐述了可编程定时/计数器的内部结构、功能、工作方式和编程的方法。

本书各章均附有习题供读者练习，以帮助读者掌握和理解有关内容。本节的电子教案可从机工网（www.cmpbook.com）上下载。

本书由甘登文教授审阅，参与编写的有罗贤海教授、周琪云教授、杨志文副教授、高明华副教授、王懿华副教授、王知源副教授，参与本书工作的还有邱晓红教授、唐南副教授、胡全连副教授、余晓莹老师、刘晓东老师、马明磊老师、陶玲等同志。

由于水平有限，书中难免有不妥和错误之处，敬请读者批评指正。

编 者

目 录

出版说明

前言

第 1 章 基础知识	1
1.1 微型计算机概述	1
1.1.1 微型计算机的特点	1
1.1.2 微型计算机系统的层次	1
1.1.3 微处理器技术发展概况	3
1.2 汇编语言及特点	3
1.2.1 机器语言	3
1.2.2 汇编语言	4
1.2.3 高级语言	4
1.2.4 汇编语言的特点	4
1.3 数据表示	4
1.3.1 数值数据	4
1.3.2 机器数的运算	7
1.3.3 字符数据	10
1.4 习题	12
第 2 章 80x86 微处理器	13
2.1 Intel 8086 微处理器	13
2.1.1 Intel 8086 CPU 内部结构	13
2.1.2 Intel 8086 内部寄存器	15
2.1.3 Intel 8086 微处理器引脚说明	17
2.2 存储器物理地址的形成	21
2.2.1 存储器结构	21
2.2.2 物理地址的形成	22
2.2.3 存储器单元的地址和内容	24
2.3 Intel 80x86 微处理器系列	25
2.3.1 Intel 80x86 微处理器系列概况	25
2.3.2 Intel 80x86 微处理器	26
2.3.3 Intel 80x86 存储器管理	31
2.4 习题	36
第 3 章 8086 指令系统	38
3.1 8086 的寻址方式	38
3.1.1 数据的寻址方式	38
3.1.2 程序转移地址的寻址方式	43

3.1.3	对端口的寻址方式	45
3.2	8086 的指令系统	45
3.2.1	数据传送类指令	45
3.2.2	算术运算类指令	49
3.2.3	位操作类指令	57
3.2.4	串操作类指令	60
3.2.5	控制转移类指令	63
3.2.6	处理机控制类指令	72
3.3	习题	73
第 4 章	MASM 汇编语言知识	77
4.1	MASM 汇编语言格式	77
4.1.1	指令语句格式	77
4.1.2	伪指令语句格式	77
4.2	汇编语句表达式	78
4.2.1	常量	78
4.2.2	变量和表达式	79
4.2.3	标号	81
4.2.4	表达式中的运算符	81
4.2.5	运算符的优先级	85
4.3	伪指令	86
4.3.1	数据定义伪指令	86
4.3.2	符号定义伪指令	86
4.3.3	段定义伪指令	87
4.3.4	子程序(过程)定义伪指令	91
4.3.5	其他伪指令	91
4.3.6	程序正常结束方式	92
4.3.7	MASM 汇编语言源程序结构	93
4.4	DOS 功能调用和 BIOS 功能调用	96
4.4.1	常用的 DOS 功能调用	97
4.4.2	DOS 功能调用	99
4.4.3	BIOS 功能调用	105
4.5	MASM 汇编语言程序的上机过程	112
4.5.1	MASM 汇编程序的有关概念	112
4.5.2	MASM 汇编语言程序的上机过程	113
4.6	习题	117
第 5 章	汇编语言程序设计	120
5.1	程序设计概述	120
5.1.1	汇编语言程序设计的一般步骤	121
5.1.2	流程图	122

5.2	顺序程序设计	122
5.3	分支程序设计	127
5.3.1	用条件转移指令实现程序分支	128
5.3.2	用跳转表实现多路分支	131
5.4	循环程序设计	135
5.4.1	循环程序的结构	135
5.4.2	循环控制的方法	136
5.4.3	单重循环程序设计	136
5.4.4	多重循环程序设计	142
5.5	子程序设计	146
5.5.1	子程序的概念	146
5.5.2	子程序的定义	146
5.5.3	子程序设计方法	147
5.5.4	子程序应用举例	148
5.5.5	子程序的嵌套与递归调用	152
5.6	模块化程序设计	155
5.7	习题	156
第6章	高级汇编语言程序设计	162
6.1	高级汇编技术	162
6.1.1	结构和联合	162
6.1.2	记录	164
6.1.3	宏	166
6.1.4	重复汇编	178
6.1.5	条件汇编	180
6.2	MASM 汇编语言与高级语言的连接	181
6.3	80x86 的寻址方式和扩充的指令	184
6.3.1	80x86 数据的寻址方式	184
6.3.2	80x86 程序转移地址的寻址方式	185
6.3.3	80x86 扩充的指令	186
6.3.4	80x86 扩充的伪指令	191
6.3.5	保护方式专用指令	192
6.4	80x86 汇编语言编程	195
6.4.1	实地址方式汇编语言程序设计	195
6.4.2	保护方式的进入和退出	199
6.4.3	保护方式汇编语言程序设计	200
6.5	习题	206
第7章	微机接口基本知识	208
7.1	微机接口	208
7.1.1	接口的概念	208

7.1.2	接口的功能	208
7.1.3	接口的组成	209
7.1.4	接口的分类	210
7.2	I/O 设备数据传送方式	210
7.2.1	端口寻址方式	210
7.2.2	CPU 与外设之间的信息传送方式	212
7.3	习题	215
第 8 章	中断系统和 DMA	216
8.1	中断系统概述	216
8.1.1	中断的基本概念	216
8.1.2	中断的处理过程	217
8.1.3	中断的优先级	218
8.1.4	8086/8088 的中断系统	221
8.2	可编程中断控制器 8259A	226
8.2.1	8259A 引脚及内部结构	226
8.2.2	8259A 的工作方式	228
8.2.3	8259A 的编程	231
8.2.4	8259A 的应用	236
8.3	DMA 概述	237
8.3.1	DMA 简介	237
8.3.2	DMA 传送方式	238
8.4	DMA 控制器 8237A	240
8.4.1	8237A 的引脚	240
8.4.2	8237A 的内部结构	242
8.4.3	8237A 的工作时序	249
8.4.4	8237A 的应用举例	250
8.5	习题	252
第 9 章	可编程并行接口芯片和串行接口芯片	253
9.1	8255 并行接口	253
9.1.1	并行接口的概念	253
9.1.2	Intel 8255A 可编程并行接口	254
9.1.3	Intel 8255A 的控制字	256
9.1.4	Intel 8255A 工作方式	257
9.1.5	Intel 8255A 编程	261
9.2	8251 串行接口	263
9.2.1	串行接口的概念	263
9.2.2	Intel 8251A 可编程串行接口	264
9.2.3	Intel 8251A 编程	269
9.3	习题	273

第 10 章 可编程定时/计数器芯片	274
10.1 定时/计数器的概念	274
10.2 Intel 8253 可编程定时/计数器	274
10.2.1 Intel 8253 的内部结构	274
10.2.2 Intel 8253 的功能	276
10.3 Intel 8253 控制字和工作方式	278
10.3.1 Intel 8253 的控制字	278
10.3.2 Intel 8253 的工作方式	279
10.4 Intel 8253 编程	284
10.5 习题	284
附录	286
附录 A DEBUG 的使用	286
附录 B 汇编程序出错信息	290
参考文献	294

第1章 基础知识

计算机技术发展到现在,使得计算机尤其是微型计算机已经成为人们生产和生活中不可缺少的工具。本章介绍微型计算机的特点和发展情况,以及计算机所使用的语言,着重阐述了计算机内的数据表示形式和运算。

1.1 微型计算机概述

1.1.1 微型计算机的特点

电子计算机通常可分为巨型机、大型机、中型机、小型机、微型机和单片机等六类。微型计算机(简称微型机或微机)是伴随着大规模集成电路的发展而诞生的,它有以下几个特点:

- (1) 采用大规模和超大规模集成电路,主流的微处理器采用单片 VLSI 形式。
- (2) 标准的工业化装配结构,体积小、重量轻、稳定性好,便于系统升级。
- (3) 开放的标准体系结构,多元化大规模工业化生产,性能价格比高。
- (4) 微型计算机的应用范围十分广泛,在信息化社会中无所不在。

1.1.2 微型计算机系统的层次

1. 微处理器 (Microprocessor, μP)

微型计算机的产生与发展主要表现在其核心部件——微处理器的发展上,每当一种新的微处理器出现后,都能带动微型计算机其他部件的发展。

微处理器不包含微型计算机硬件的全部功能,但它是微型计算机控制、处理的核心。目前主流的微处理器采用单片 VLSI 电路,其体系结构技术、工作频率已达空前高的水平。

主流微处理器具有通用性,不仅用于微型机也用于工作站及超级计算机。

微处理器一般由算术逻辑部件、寄存器、控制部件及内部总线组成,如图 1-1 所示。

2. 微型计算机 (Microcomputer, μC)

微型计算机是指以微处理器为核心,配以存储器、输入输出接口和相应的辅助电路所构成的裸机。把微型计算机集成在一个芯片上就构成了单片微型计算机(单片机)。

微处理器是执行指令的核心,它的性能决定了整个微型计算机的性能。

存储器用于指令代码、操作数和运行结果的存储。

输入输出接口电路用于微处理器与外围设备的连接,主要包括:并口、串口、外存接口、显示器接口、网络接口、声音接口等。

系统总线将上述模块连接起来,作为各种信息的通路,按信息类别分为数据、地址、控制三类总线。图 1-2 所示为微型计算机的基本结构。

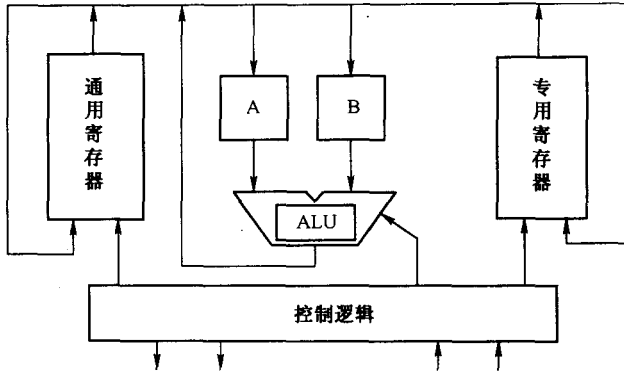


图 1-1 微处理器结构框图

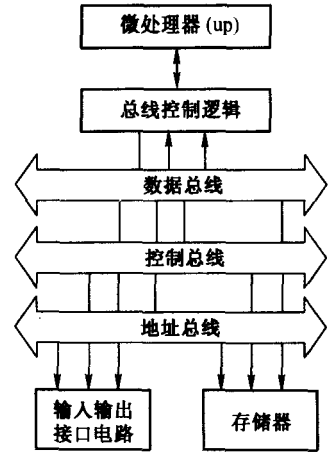


图 1-2 微型计算机基本结构

3. 微型计算机系统 (Microcomputer system)

微型计算机系统是指以微型计算机为主体，配以相应的外围设备及其他专用电路、电源、面板、机箱以及软件系统所构成的系统。图 1-3 所示为微处理器、微型计算机、微型计算机系统三者的关系。

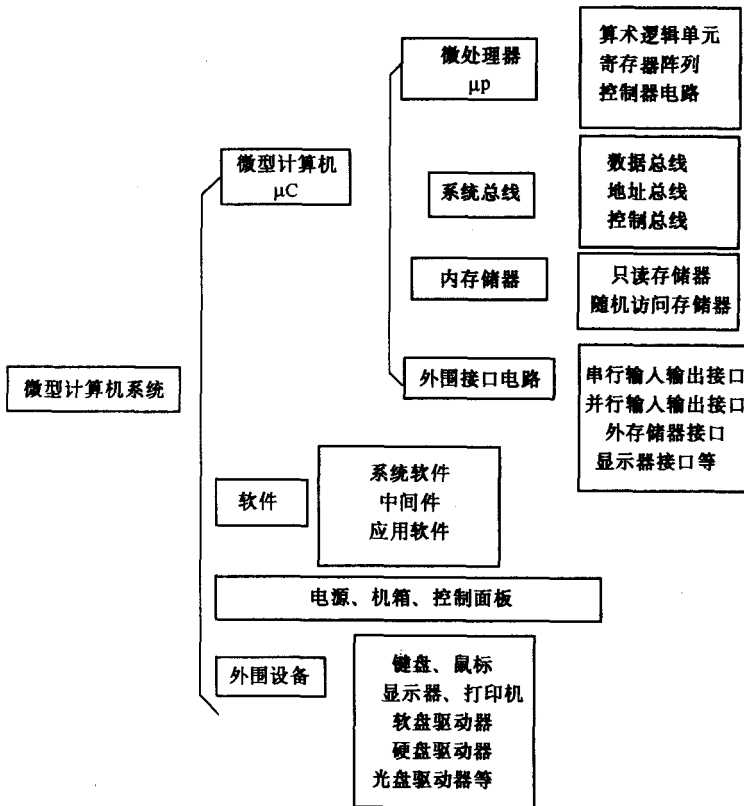


图 1-3 微处理器，微型计算机，微型计算机系统三者的关系

软件系统主要包括系统软件、中间软件、应用软件。
外围设备主要有软驱、硬驱、光驱、键盘、鼠标、显示器。

1.1.3 微处理器技术发展概况

当今微电子技术水平迅速提高,平均每 18 个月电子电路的集成度提高一倍,使得微处理器结构设计者可在片内实现各种先进体系结构,微处理器及外围支援器件的性价比达到前所未有的水平,微计算机系统的性价比迅速提高,已达到前期的工作站与小型机的水平。

Intel 微处理器技术发展概况:

20 世纪 80 年代 IBM 公司以 Intel 8086/8088 作为核心处理器研制出个人计算机——IBM PC, Intel 微处理器成为市场主流。

1985 年 Intel 推出 80386 微处理器,完成了 16 位结构向 32 位结构的转换。

1989 年 Intel 推出 80486 微处理器,片内集成了 Cache 和浮点部件,基本指令用硬线逻辑实现,指令执行效率大大提高,比 386 快 2~3 倍。

1993 年 3 月 Intel 推出 Pentium(奔腾)微处理器,片内 L1 Cache 分为 I Cache 和 D Cache,设有两条流水线,提高了指令执行的并行性,是一种超标量结构(Super scalar)。

1994 年又推出 Pentium/MMX(多媒体扩充技术)微处理器,Pentium/MMX 为第一代“奔腾”微处理器,简称 P5。

1995 年 2 月 Intel 推出第 2 代 Pentium 微处理器,Pentium Pro(高能奔腾),简称 P6, P6 的创新点是:

(1) L2 Cache 集成到封装内,微处理器与 L2 Cache 数据交换频宽大大提高。

(2) 采用“无序执行”技术(Out of order execution),使处理器内部保持很高的指令执行并行度。

1997 年 5 月 Intel 推出名为 Pentium II 的微处理器,是具有 MMX 技术的 Pentium Pro 微处理器,Intel 微体系结构从 P5 内核转向 P6 内核。

1999 年 3 月 Intel 推出名为 Pentium III 的微处理器。它基于 P6 微内核,具有 MMX 技术,提供数据流 SIMD 扩展(SSE)指令(70 条),进一步支持多媒体信息处理。由于采用先进的半导体工艺技术,其工作频率可达 1000MHz 以上,前沿总线频率为 100MHz 或 133MHz。

2000 年 11 月 Intel 推出名为 Pentium 4 的微处理器。采用全新的,称之为 Net Burst 的微结构(IA-32),为第四代奔腾微处理器。其特点是:

(1) 20 级的超长流水线,便于提高片内主频,最低 1.4GHz。

(2) 更先进的动态执行技术,更高的分支预测准确性。

(3) 双倍速 ALU 部件,SSE2 指令集,400MHz 的 FSB。

1.2 汇编语言及特点

1.2.1 机器语言

计算机能够直接识别的数据是由二进制数 0 和 1 组成的代码。机器指令就是用二进制代

码组成的指令，一条机器指令控制计算机完成一个基本操作。

用机器语言编写的程序是计算机唯一能够直接识别并执行的程序，而用其他语言编写的程序必须经过翻译才能转换成机器语言程序，所以，机器语言程序被称为目标程序。

1.2.2 汇编语言

为了克服机器语言的缺点，人们采用助记符表示机器指令的操作码，用变量代替操作数的存放地址等，这样就形成了汇编语言。所以汇编语言是一种用符号书写的、基本操作与机器指令相对应的（一一对应）、并遵循一定语法规则的计算机语言。

用汇编语言编写的程序称为汇编源程序，扩展名为 ASM。

汇编语言是一种符号语言，比机器语言容易理解和掌握，也容易调试和维护。但是，汇编语言源程序要翻译成机器语言程序才可以由计算机执行，这个翻译的过程称为“汇编”。这种把汇编源程序翻译成目标程序的语言加工程序称为汇编程序。汇编程序的主要功能有：

- (1) 检查源程序；
- (2) 测出源程序中的语法错误，并给出出错信息；
- (3) 产生源程序的目标程序，并可给出列表文件（LST 文件）；
- (4) 展开宏指令。

1.2.3 高级语言

汇编语言虽然较机器语言直观，但仍然烦琐难懂。于是人们研制出了高级程序设计语言。高级程序设计语言接近于人类自然语言的语法习惯，与计算机硬件无关，易被用户掌握和使用。

目前广泛应用的高级语言有多种，如 BASIC、FORTRAN、PASCAL、C、C++等。

1.2.4 汇编语言的特点

1. 汇编语言的特点

- (1) 汇编语言与处理器密切相关。
- (2) 汇编语言程序效率高。
- (3) 编写汇编语言源程序比编写高级语言源程序烦琐。
- (4) 调试汇编语言程序比调试高级语言程序困难。

2. 汇编语言的主要应用场合：

- (1) 程序执行占用较短的时间，或者占用较小存储容量的场合。
- (2) 程序与计算机硬件密切相关，程序直接控制硬件的场合。
- (3) 需提高大型软件性能的场合。
- (4) 没有合适的高级语言的场合。

1.3 数据表示

1.3.1 数值数据

人们在日常生活中习惯使用十进制数，而二进制数由于简单、容易实现，是数字系统中，

特别是计算机中广泛采用的一种数制。但使用二进制数不方便，比如表示一个十进制数时，需用四位二进制数才能表示一位十进制数，所用的位数太多，读写很不方便，所以在实际工作中又常采用八进制或十六进制，因此计算机中的数据表示常用二进制、八进制、十六进制和十进制：

二进制：由一串 0、1 组成，其后跟字母 B；

十进制：由 0~9 的数字组成，其后跟字母 D，可缺省；

八进制：由数字 0~7 组成，其后跟字母 O 或 Q；

十六进制：由 0~9 及 A~F 组成，其后跟字母 H，如果数的第一个字符是 A~F，则应在其前加数字 0。由于二进制数的基数太小，书写和阅读都不方便，而十六进制的基数 $16=2^4$ ，这样二进制数与十六进制之间能方便地转换。因此，习惯把二进制数改写成十六进制数，在汇编语言程序设计时尤其如此。

数值数据分为有符号数和无符号数。无符号数最高位表示数值，而有符号数最高位表示符号。有符号数由两部分组成：一部分是表示数的符号，另一部分是表示数的数值。由于数的符号是一个具有正、负两种值的离散信息，所以它可以用一位二进制数来表示。通常是以 0 表示正数，以 1 表示负数。对于一个 n 位二进制数，如果数的第一位为符号位，那么余下的 $n-1$ 位就表示数的数值部分。我们把直接用正号“+”和负号“-”来表示符号的二进制数称为符号数的真值。数的真值形式是一种原始形式，无法直接用在数字计算机中。但是，当将符号数值化之后，便可以在计算机中使用它了。因此在计算机中使用的符号数便称为机器数。一般机器数有三种表示形式，即原码、反码和补码，常用的是补码。

1. 原码

原码又被称为“符号—数值表示”。当用原码形式表示正数和负数时，最高位是符号位。对于正数，符号位表示的是 0，对于负数，符号位表示的是 1，其余各位表示数值位，这样就得到了有符号数的原码表示。

原码表示简单易懂，但若是两个异号数相加（或两个同号数相减），就要做减法。为了把减法运算转换为加法运算就引进了反码和补码。

【例 1-1】假如两个带符号的二进制数分别为 S_1 和 S_2 ，其真值形式为：

$$S_1 = +11001 \quad S_2 = -01011$$

则 S_1 和 S_2 的原码表示形式为：

$$[S_1]_{\text{原}} = 011001 \quad [S_2]_{\text{原}} = 101011$$

根据上述原码形成规则，一个 n 位的整数 S （包括一位符号位）的原码一般表达式为：

$$[S]_{\text{原}} = \begin{cases} S & 0 \leq S < 2^{n-1} \\ 2^{n-1} - S & -2^{n-1} < S \leq 0 \end{cases}$$

对于定点小数而言，一般将小数点定在最高位的左边，此时，数值小于 1。定点小数原码一般表达式为：

$$[S]_{\text{原}} = \begin{cases} S & 0 \leq S < 1 \\ 1 - S & -1 < S \leq 0 \end{cases}$$

由原码的一般表达式可以得出：

(1) 当 S 为正数时, $[S]_{\text{原}}$ 和 S 的区别只是增加一位用 0 表示的符号位。由于在数的左边增加一位 0 对该数的数值并无影响, 所以 $[S]_{\text{原}}$ 就是 S 本身。

(2) 当 S 为负数时, $[S]_{\text{原}}$ 和 S 的区别是增加了一位用 1 表示的符号位。

(3) 在原码表示中, 有两种不同形式的 0, 即:

$$[+0]_{\text{原}} = 0.00\dots 0$$

$$[-0]_{\text{原}} = 1.00\dots 0$$

2. 反码

反码又称为“对 1 的补数”。当用反码表示时, 最高位为符号位, 符号位为 0 代表正数, 符号位为 1 代表负数。对于正数, 反码和原码相同, 即符号位用 0 表示, 数值位值不变。而对于负数, 反码的数值是将原码数值按位求反, 若原码的某位为 1, 则反码的相应位便为 0, 或者原码的某位为 0, 反码的相应位便为 1, 即负数的反码符号位用 1 表示, 数值位为原码数值位按位取反形成, 即 0 变 1、1 变 0。所以, 反码数值的形成与它的符号位有关。

【例 1-2】假如两个带符号的二进制数分别为 S_1 和 S_2 , 其真值形式为:

$$S_1 = +11001 \quad S_2 = -01011$$

则 S_1 和 S_2 的反码表示形式为:

$$[S_1]_{\text{反}} = 011001 \quad [S_2]_{\text{反}} = 110100$$

根据上述的反码形成规则, 一个 n 位的整数 S (包括一位符号位) 的反码一般表达式为:

$$[S]_{\text{反}} = \begin{cases} S & 0 \leq S < 2^{n-1} \\ (2^n - 1) + S & -2^{n-1} < S \leq 0 \end{cases}$$

同样, 对于定点小数, 如果小数部分的位数为 m , 则它的反码一般表达式为:

$$[S]_{\text{反}} = \begin{cases} S & 0 \leq S < 1 \\ (2 - 2^m) + S & -1 < S \leq 0 \end{cases}$$

从反码的一般表达式可以看出:

- (1) 正数 S 的反码 $[S]_{\text{反}}$ 与原码 $[S]_{\text{原}}$ 相同。
- (2) 对于负数 S , 其反码 $[S]_{\text{反}}$ 的符号位为 1, 数值部分是将原码数值按位求反。
- (3) 在反码表达式中, 0 的表示有两种不同的形式, 即:

$$[+0]_{\text{反}} = 0.00\dots 0$$

$$[-0]_{\text{反}} = 1.11\dots 1$$

3. 补码

补码又称为“对 2 的补数”。在补码表示方法中, 正数的表示同原码和反码的表示是一样的, 即符号位用 0 表示, 数值位值不变; 而负数的表示却不相同, 对于负数, 将原码转变成补码的规则是: 符号位不变, 仍为 1, 数值部分变反加 1, 即逐位变反, 在最低位加 1, 也就是反码加 1 形成。

对一个二进制数按位求反后在末位加 1 的运算称为求补运算。

对一个正数的补码将其符号位和数值位都按位求反后在末位加 1, 可以得到与此正数相对的负数的补码; 对负数的补码同样操作可得到与此负数相对的正数的补码, 即:

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