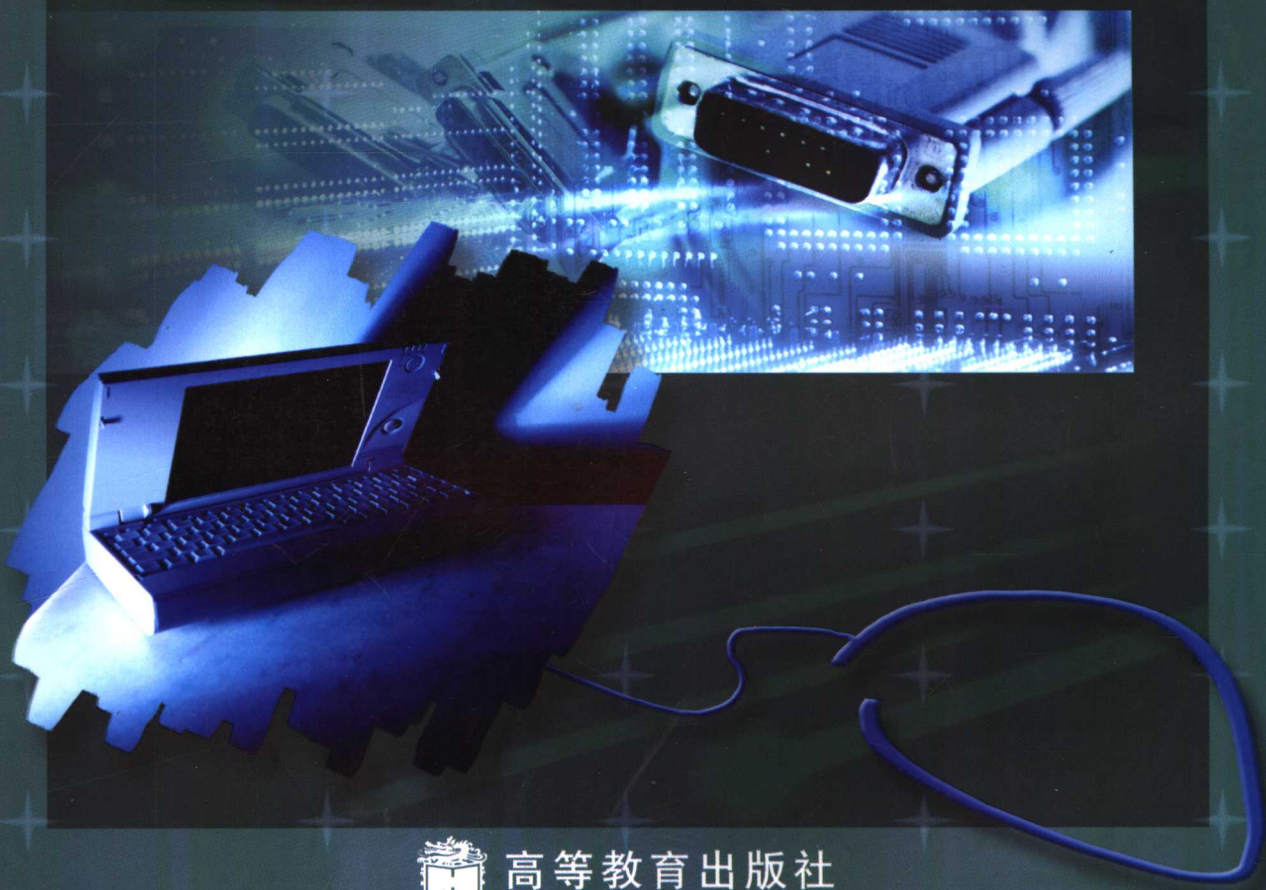


高等学校教材

32位 微型计算机原理 与接口技术

谢瑞和 翁虹 张士军 杨明 编著



高等教育出版社

高等学校教材

32 位微型计算机原理 与接口技术

谢瑞和 翁 虹 张士军 杨 明 编著

高等教育出版社

内容提要

本书以 Pentium 系列微处理器为主体,全面地介绍了 32 位微型计算机的原理与接口技术。根据由浅入深的原则,精心组织内容,共分为微型计算机基础、接口与提高三个层次,不仅能明显提高学生的学习效率,而且显著压缩了全书的篇幅。基础篇主要讲述 Pentium 系列微处理器的基本结构与中断技术等基本操作原理、指令系统和汇编语言程序设计;接口篇以崭新的面貌区别于传统的同类教材,首先系统介绍微处理器的总线操作与时序、存储器与 I/O 接口技术,然后分章详细讨论了当代 PC 的串行接口、并行接口、USB 接口与 PCI 扩展总线;提高篇首先全面剖析了 Pentium 系列的操作模式,尤其对保护模式进行了系统而精辟的描述,然后详细介绍了 Pentium 系列的 MMX、超标量、动态执行、分支预测、条件传送及 CPU 识别等一系列增强技术以及 Pentium 系列主板与芯片组。

本书集作者 20 多年从事微型计算机技术应用的教學、科研与写作经验,内容丰富、层次分明、语言精炼流畅,概念清晰,便于读者自学。

本书是高等院校理工专业微型计算机原理与接口技术类课程的通用教材,对于广大的工程应用与科技人员也是一本难得的参考书。

图书在版编目(CIP)数据

32 位微型计算机原理与接口技术/谢瑞和等编著.

北京:高等教育出版社,2004.8

ISBN 7-04-015501-X

I. 3... II. 谢... III. ①微型计算机-理论-教材
②微型计算机-接口-教材 IV. TP36

中国版本图书馆 CIP 数据核字(2004)第 066965 号

策划编辑 陈红英 责任编辑 陈红英 市场策划 刘 茜
封面设计 李卫青 责任印制 陈伟光

出版发行 高等教育出版社
社 址 北京市西城区德外大街 4 号
邮政编码 100011
总 机 010-82028899

购书热线 010-64054588
免费咨询 800-810-0598
网 址 <http://www.hep.edu.cn>
<http://www.hep.com.cn>

经 销 新华书店北京发行所
印 刷 北京民族印刷厂

开 本 787×1092 1/16
印 张 20
字 数 480 000

版 次 2004 年 7 月第 1 版
印 次 2004 年 7 月第 1 次印刷
定 价 25.00 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

前 言

每当我们主讲微型计算机原理与接口技术课程时,不时有学生发问:“我们现在使用的计算机都是 Pentium III 或 Pentium 4,为什么学的还是 20 世纪 70 年代的 8086、8088?”。面对这类问题,作为教师确实没有任何充足的理由能说服他们。

接口技术也是一样,学的是 ISA 扩展总线与 825x 系列早期接口芯片,但使用的都是 PCI 扩展总线和功能强大的超大规模芯片组,或者各类可编程逻辑器件,时差也有约 20 年。

在和许多兄弟院校的学习交流中,大家都一致认为应该彻底改变这种现象,但难于找到一本真正适合本科教学的 32 位微型计算机原理与接口技术教材。

本书试图在这方面作一些尝试,全书以 Pentium 系列微处理器为主线,介绍 32 位微型计算机原理与接口技术。在内容的选材与安排上颇具特色,彻底抛弃了同类教材中近 20 年来基本维持不变的传统内容,实际上这些内容早已被当代微型计算机所淘汰。根据由浅入深的原则,精心组织内容,将其划分为基础、接口与提高三个层次,不仅能明显提高学生学习效率,而且显著压缩了全书的篇幅。基础篇主要讲述 Pentium 系列微处理器的基本结构与中断技术等基本操作原理、指令系统和汇编语言程序设计。接口篇以崭新的面貌区别于传统的同类教材,首先系统介绍了 Pentium 微处理器的总线操作与时序、存储器与 I/O 接口技术,然后分章详细讨论了当代 PC 的串行接口、并行接口、USB 接口与 PCI 扩展总线。提高篇首先全面剖析了 Pentium 系列的操作模式,尤其对保护模式进行了系统而精辟的描述,然后详细介绍了 Pentium 系列的 MMX、超标量、动态执行、分支预测、条件传送及 CPU 识别等一系列增强技术以及 Pentium 系列主板与芯片组。

本书集编著者 20 多年从事微型计算机技术应用的教学、科研与写作经验,在内容层次、语言表达以及概念描述等诸方面都力争便于读者自学。本书是高等院校理工专业微型计算机原理与接口技术类课程的通用教材。同时我们正在着手编著同本书配套的实验课教材及开发相应实验装置,不久即可问世。

参加本书编著的有谢瑞和、翁虹、张士军、杨明、马爱梅、曾延安等多年主讲微型计算机原理与接口技术的教师。全书由谢瑞和主编、统一策划。第 1~4 章由谢瑞和、翁虹、张士军、杨明等集体编著,第 5~15 章由谢瑞和编著。周丽军、张金与何博等研究生为本书的例题程序和接口技术实例做了大量开发工作。高等教育出版社领导与编辑以非凡的决策与胆略全力支持本书出版,在此一并致谢。

由于我们组织 32 位微型计算机原理与接口技术教学的时间还不长,积累的经验不多,编著者个人学识很有限,书中难免会有不少错误或不妥之处,诚心期待同行与读者批评赐教。

编 者

2004 年 4 月于华中科技大学

目 录

第一篇 微型计算机基础

第1章 计算机运算基础	(3)	3.2.1 MOV 指令	(36)
1.1 无符号数	(3)	3.2.2 堆栈操作指令	(37)
1.1.1 二进制与十六进制	(3)	3.2.3 地址与标志传送指令	(38)
1.1.2 数制转换	(4)	3.3 算术运算指令	(39)
1.2 常用的编码	(6)	3.3.1 加法指令	(39)
1.2.1 BCD 码	(6)	3.3.2 减法指令	(40)
1.2.2 ASCII 码与奇偶校验	(7)	3.3.3 乘法指令	(41)
1.3 带符号数	(8)	3.3.4 除法指令	(42)
1.3.1 原码、反码与补码	(8)	3.4 逻辑、移位和位操作指令	(43)
1.3.2 补码的换算	(9)	3.4.1 逻辑运算指令	(43)
思考与练习	(11)	3.4.2 移位指令	(44)
第2章 计算机硬件基础	(12)	3.4.3 位测试指令	(46)
2.1 计算机发展简史	(12)	3.5 I/O 与串操作指令	(48)
2.2 微型计算机系统概述	(13)	3.5.1 I/O 操作指令	(48)
2.2.1 微型计算机硬件系统的基本结构	(13)	3.5.2 串操作指令	(49)
2.2.2 微型计算机的工作原理	(15)	3.6 分支转移指令	(50)
2.2.3 简单程序执行过程	(16)	3.6.1 CALL 和 RET 指令	(50)
2.3 8086、8088、80286 16 位微处理器	(18)	3.6.2 INT 指令	(51)
2.3.1 8086、8088 CPU 的内部结构	(18)	3.6.3 循环指令	(53)
2.3.2 8086、8088 CPU 中的寄存器结构	(19)	3.6.4 无条件转移指令	(53)
2.3.3 存储器的分段与物理地址形成	(22)	3.6.5 条件转移指令	(54)
2.3.4 80286 微处理器	(23)	3.6.6 JCXZ/JECXZ 指令	(55)
2.4 80386、80486 32 位微处理器	(24)	3.7 其他常用指令	(55)
2.4.1 80386 微处理器	(24)	3.7.1 转换指令	(55)
2.4.2 80486 微处理器	(26)	3.7.2 其他典型应用指令	(57)
2.5 Pentium 系列微处理器	(26)	思考与练习	(57)
2.5.1 Pentium 的结构	(27)	第4章 汇编语言程序设计	(59)
2.5.2 Pentium 的寄存器	(28)	4.1 汇编伪指令	(59)
思考与练习	(31)	4.1.1 标号	(59)
第3章 汇编语言指令基础	(33)	4.1.2 子程序定义	(61)
3.1 寻址方式	(33)	4.1.3 宏定义	(62)
3.2 传送指令	(36)	4.1.4 段定义	(63)
		4.1.5 简化段定义	(64)
		4.1.6 指令集版本	(65)
		4.1.7 定义存储方式	(65)

4.1.8 其他汇编伪指令	(66)	5.1.1 中断的基本概念	(82)
4.2 程序设计举例	(69)	5.1.2 中断向量表	(84)
4.3 汇编与 C/C++ 接口	(76)	5.2 软件中断	(86)
4.3.1 C/C++ 程序中内嵌汇编的 方法	(76)	5.2.1 异常中断	(86)
4.3.2 独立汇编模块的汇编与 C/C++ 混合编程	(76)	5.2.2 INTN 指令中断	(89)
思考与练习	(81)	5.3 硬件中断	(90)
第 5 章 中断	(82)	5.3.1 Pentium 硬件中断综述	(90)
5.1 中断系统	(82)	5.3.2 可屏蔽中断 INTR	(92)
		5.3.3 外部中断应用编程	(94)
		思考与练习	(96)

第二篇 接口部分

第 6 章 存储器与 I/O 端口	(99)	7.1.3 端口模式简介	(137)
6.1 存储器基本知识	(99)	7.1.4 端口 I/O 地址	(138)
6.1.1 概述	(99)	7.2 SPP 端口模式	(138)
6.1.2 ROM	(100)	7.2.1 简单的输入/输出	(138)
6.1.3 RAM	(102)	7.2.2 兼容模式	(141)
6.1.4 内存条	(102)	7.2.3 4 位组模式	(142)
6.1.5 地址译码	(104)	7.3 EPP 端口模式	(143)
6.2 高速缓冲存储器	(106)	7.4 ECP 端口模式	(146)
6.2.1 Cache 的作用	(106)	7.5 ADC 接口设计	(147)
6.2.2 Cache 的读/写策略	(106)	7.5.1 SPP 模式下的 ADC 接口	(147)
6.2.3 Cache 的实施原理简介	(109)	7.5.2 EPP 模式下的 ADC 接口	(149)
6.3 双重独立总线结构	(110)	7.6 DAC 接口设计	(153)
6.4 存储器访问	(111)	思考与练习	(156)
6.4.1 信号通道	(111)	第 8 章 串行接口	(158)
6.4.2 周期状态	(114)	8.1 RS-232 标准	(158)
6.4.3 单次传送周期	(114)	8.1.1 接口及引脚定义	(159)
6.4.4 等待状态	(116)	8.1.2 电气特性	(160)
6.4.5 4 字边界对准	(116)	8.2 接收器与发送器	(160)
6.4.6 突发周期	(118)	8.3 COM 端口及数据通信	(162)
6.4.7 流水线访问	(120)	8.3.1 调制解调器简介	(162)
6.5 DMA 方式	(122)	8.3.2 端口寄存器	(163)
6.5.1 DMA 传送原理	(122)	8.3.3 通信编程	(166)
6.5.2 8237-5 DMAC 简介	(123)	8.4 应用举例	(168)
6.6 I/O 端口访问	(124)	思考与练习	(170)
6.7 ISA 扩展总线	(129)	第 9 章 USB 接口	(171)
思考与练习	(131)	9.1 USB 概述	(171)
第 7 章 并行接口	(133)	9.1.1 背景	(171)
7.1 概述	(133)	9.1.1 物理接口	(172)
7.1.1 物理接口	(133)	9.1.2 系统组成	(175)
7.1.2 IEEE 1284 标准	(135)	9.2 USB 传输协议	(177)

9.2.1 端点与管道	(177)	10.4 PCI 总线协议	(193)
9.2.2 信息包的结构与种类	(177)	10.4.1 传输控制	(193)
9.2.3 4 种传输方式	(180)	10.4.2 地址空间	(194)
9.3 USB 系统开发简介	(183)	10.4.3 总线的驱动	(195)
思考与练习	(184)	10.5 数据传输过程	(195)
第 10 章 PCI 总线	(186)	10.5.1 读操作	(195)
10.1 扩展总线	(186)	10.5.2 写操作	(196)
10.2 PCI 总线	(187)	10.5.3 总线的仲裁	(197)
10.2.1 PCI 系统框图	(187)	10.6 配置空间	(198)
10.2.2 PCI 总线信号	(188)	10.6.1 配置空间的结构	(198)
10.3 PCI 信号的定义	(190)	10.6.2 设备的识别寄存器	(199)
10.3.1 系统信号	(190)	10.6.3 设备的控制寄存器	(200)
10.3.2 地址和数据信号	(190)	10.6.4 设备的状态寄存器	(201)
10.3.3 接口控制信号	(191)	10.6.5 其他寄存器	(201)
10.3.4 其他常用的控制或状态 信号	(192)	10.7 PCI BIOS 功能调用	(203)
		思考与练习	(206)

第三篇 提高部分

第 11 章 操作模式	(209)	11.5 I/O 地址空间的保护	(234)
11.1 实模式综述	(209)	11.5.1 I/O 特权级的保护	(234)
11.2 保护模式下的存储器分段管理	(211)	11.5.2 I/O 许可位图的保护	(235)
11.2.1 控制寄存器与存储器管理 寄存器	(212)	11.6 进出保护模式	(236)
11.2.2 虚拟存储器	(213)	11.7 分页管理机制	(242)
11.2.3 描述符表的概念	(214)	11.7.1 概述	(242)
11.2.4 特权级及其保护	(215)	11.7.2 4 KB 分页	(243)
11.2.5 段选择器与段描述符的 结构	(216)	11.7.3 线性地址转换全过程	(245)
11.2.6 描述符表的定位	(219)	11.7.4 4 MB 分页	(246)
11.2.7 存储器寻址过程	(220)	思考与练习	(247)
11.2.8 应用段	(222)	第 12 章 MMX 技术	(248)
11.2.9 系统段	(223)	12.1 MMX 技术	(248)
11.2.10 控制转移与调用门	(225)	12.2 MMX 指令解说	(250)
11.2.11 保护机制要点	(226)	12.2.1 MMX 指令规则概述	(250)
11.3 保护模式中断	(227)	12.2.2 MMX 指令集表	(253)
11.3.1 中断描述符表与中断门	(227)	12.3 单指令多数据处理	(256)
11.3.2 保护模式下的异常中断	(228)	12.4 SSE 技术简介	(261)
11.3.3 保护模式下的 DPMI 控制 功能	(229)	思考与练习	(262)
11.4 多任务系统	(229)	第 13 章 其他增强技术	(263)
11.4.1 任务状态段及任务门	(230)	13.1 Pentium 的超标量流水线	(263)
11.4.2 任务切换	(233)	13.1.1 整型流水线	(263)
		13.1.2 指令配对	(265)
		13.1.3 浮点流水线	(267)
		13.2 动态执行技术	(269)

13.2.1 Pentium II 的流水线结构	(269)	14.1 北桥南桥芯片组	(286)
13.2.2 动态执行过程	(270)	14.2 Intel 810/815 芯片组	(288)
13.3 分支预测	(272)	14.3 Pentium 4 芯片组	(290)
13.4 条件传送指令	(273)	14.4 主板结构	(292)
13.5 CPU 识别指令	(274)	14.5 系统 BIOS ROM 与 CMOS RAM	(294)
13.5.1 CPU 识别	(275)	14.5.1 BIOS ROM	(294)
13.5.2 CPUID 指令的特殊功能	(278)	14.5.2 CMOS RAM	(295)
13.5.3 UD2 指令	(279)	思考与练习	(297)
13.6 特殊方式寄存器	(279)	第 15 章 IA 指令集	(298)
13.6.1 读/写特殊方式寄存器	(279)	15.1 整型基本指令	(298)
13.6.2 读性能监视计数器	(280)	15.2 浮点部件指令	(304)
13.6.3 读时间标志计数器	(280)	15.3 MMX 指令	(305)
13.7 Pentium 4 的增强点	(281)	思考与练习	(307)
13.8 Pentium 系列微处理器总览	(283)	参考文献	(308)
思考与练习	(284)		
第 14 章 芯片组与主板	(286)		

第一篇 微型计算机基础

本篇系统地介绍了微型计算机的运算基础、硬件基础及其软件基础,重点是学习 Intel 系列的指令系统、汇编语言程序设计及中断。

第 1 章系统地介绍了二进制数及其同十进制数之间的转换与计算机中常用的几种编码。二进制数制是计算机操作原理的基础,因而本章内容是学习微型计算机技术最起码的入门常识。

第 2 章概述了微型计算机发展的过程,简单描述了计算机操作的基本过程,扼要介绍了 Intel 体系各代微处理器的结构与特点,重点描述了 Pentium 系列微处理器的内部寄存器。这些内容都是为第 3、4 章的指令解说与汇编语言和第 5 章的中断系统奠定基础。至于更细节的微型计算机系统硬件原理描述将放在后面的第二、三篇中。

第 3 章详细介绍 Intel 体系微处理器的指令系统,这是全书的重点。由于指令系统的内容繁杂,试图逐条地解说将使本章内容繁琐而庞大,本章只能针对最常用的典型指令进行解说,并总结出一些规则,起到举一反三的作用,从而使读者全面理解与掌握整个指令集并不困难。

第 4 章专门讨论汇编语言程序设计方法以及汇编语言与高级语言如何接口的问题,全书内容是本书的重点,也可以视为重中之重。学习汇编语言主要是解决微型计算机系统底层用户接口的问题,用汇编语言来编制复杂的数学运算程序似乎没有必要,因为使用高级语言来解决此类问题非常简便。因此掌握汇编语言同高级语言之间的接口方法极其重要。

第 5 章描述了微型计算机系统的中断概念,包括中断系统的原理与操作过程,对内部软中断与外部硬中断、中断向量表等基本模块或概念进行了全面的解释。但这里只涉及实模式下的中断,保护模式下的中断将在第三篇中讨论。

本篇没有涉及操作模式这个基本概念,因为保护模式包含的内容太多,而且相对于传统的 16 位微型计算机教学而言都是全新的知识,学习起来比较费力,本书安排在第三篇中学习保护模式。

第1章 计算机运算基础

计算机是用来进行各种信息处理的工具,尽管这些被处理的信息千差万别,但它们都是以数据的形式由计算机来操作的,而且通常都是以二进制为基础的。本章简要地概述了计算机中使用的数制系统及其几种常用的编码,可能多数读者对这些内容并不陌生,但作为计算机的运算基础来温习一遍,有助于更好地理解计算机内部操作的机理,提高学习后续内容的效率。

1.1 无符号数

无论计算机如何发展,它的内部操作总是基于由“0”与“1”组成的二进制数制,换言之,计算机实质上只能识别出二进制的“0”和“1”,只是由不同的二进制位串可以编制出由它执行的各种不同的复杂操作而已。

1.1.1 二进制与十六进制

1. 二进制数

二进制数的基数为2,逢二进一,它的多项式表示为:

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

二进制整数部分的位权从小到大依次为 $2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7 \cdots$, 亦即为十进制的 1, 2, 4, 8, 16, 32, 64, 128, 256…。二进制小数部分的位权从大至小依次为 $2^{-1}, 2^{-2}, 2^{-3}, 2^{-4} \cdots$, 即为十进制的 1/2, 1/4, 1/8, 1/16…。它的系数只有“0”与“1”两个数字,例如,

$$(1101.11)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$$

可见它等于十进制的 13.75。

为区别起见,规定二进制数以英文字母“B”(Binary)为后缀标记,如 1101.11B。

为方便起见,在二进制中经常使用 K(Kilo)、M(Mega)、G(Giga)等作为计量单位,应注意,它们与十进制表示的数有所差异。

$$1 \text{ K} = 2^{10} = 1\,024$$

$$1 \text{ M} = 2^{20} = 1\,048\,576$$

$$1 \text{ G} = 2^{30} = 1\,073\,741\,824$$

2. 十六进制数

十六进制数的基数是十六,逢十六进一,它的多项式表示法为

$$h_n \times 16^n + h_{n-1} \times 16^{n-1} + \cdots + h_1 \times 16^1 + h_0 \times 16^0 + h_{-1} \times 16^{-1} + h_{-2} \times 16^{-2} + \cdots$$

十六进制整数部分的位权从小到大依次为 $16^0, 16^1, 16^2, 16^3, 16^4 \cdots$, 亦即十进制的 1, 16, 256, 4096, 65536…。十六进制小数部分的位权从大到小依次为 $16^{-1}, 16^{-2}, 16^{-3}, 16^{-4}$

...,亦即十进制的 $1/16, 1/256, 1/4096, 1/65536 \dots$ 。它的系数有 16 个数字,前 10 个为十进制数字 0~9,后 6 个为英文字母 A,B,C,D,E,F,分别为十进制数的 10,11,12,13,14,15。例如:

$$(89AB.4)_{16} = 8 \times 16^3 + 9 \times 16^2 + 10 \times 16^1 + 11 \times 16^0 + 4 \times 16^{-1}$$

可见它等于十进制的 35243.25。

为区别起见,规定十六进制数以英文字母“H”(Hexadecimal)为后缀标记,如 89AB.4H。必须注意的是,为了与非数字含义的字符串相区别,当十六进制数的最高位为 A~F 中任何一个数字时,应在最前面加一位数字“0”,例如,ABCD3456H 是非法的,而应改写为 0ABCD3456H;当最高位为数字 1~9 时,则默认为是数据而非字符串,例如,9FEDCH 是合法的。

表 1.1 列举了部分二进制数、十进制数与十六进制数的对照。这里将所有的位都用来表示数值,所以是无正负符号位的数。

表 1.1 无符号二进制数、十进制数和十六进制数的对照

二进制	十进制	十六进制	二进制	十进制	十六进制	二进制	十进制	十六进制
00000000B	0	00H	00001000B	8	08H	00010000B	16	10H
00000001B	1	01H	00001001B	9	09H	00010001B	17	11H
00000010B	2	02H	00001010B	10	0AH	00010010B	18	12H
00000011B	3	03H	00001011B	11	0BH	00010011B	19	13H
00000100B	4	04H	00001100B	12	0CH	00010100B	20	14H
00000101B	5	05H	00001101B	13	0DH	00010101B	21	15H
00000110B	6	06H	00001110B	14	0EH	00010110B	22	16H
00000111B	7	07H	00001111B	15	0FH	00010111B	23	17H

在计算机中,将 8 位二进制数称为一个字节(Byte),在数据的存储与处理中往往以字节为基本单元。

- 对于 8 位无符号的二进制数,所能表示的十进制数范围是 0~255。
- 对于 16 位无符号的二进制数,所能表示的十进制数范围是 0~65535。
- 对于 32 位无符号的二进制数,所能表示的十进制数范围是 0~4294967295。

1.1.2 数制转换

上述二进制数与十六进制数的介绍中就已经描述了将它们转换成十进制数的方法,这种方法也适用于其他任何非十进制数,即只需将非十进制数以多项式的形式展开表示,然后计算它们的十进制之和即可将其转换为十进制数。

下面讨论十进制数转换成非十进制数,以及二进制数与十六进制数之间的相互转换。

1. 十进制数转换为非十进制数

将十进制数整数部分转换成其他进制数,只需将整数除以该进制的基数,取其余数作为转换结果。将十进制数小数部分转换成其他进制数,则应将该进制的基数乘以十进制数小数部分,然后取乘积的整数作为转换结果。

① 十进制数转换为二进制数,采用除 2 取余法。即将十进制整数除以 2,得到一个商数和余数,再将商数除以 2 又得到一个商数和余数,如此继续除下去直至除尽为止。以最后所得的商数“1”作为最高位,将各次所得的余数按“后得先排”的顺序写下来,即得二进制转换结果。

【例 1.1】 100 = (?) B

转换过程如图 1.1 所示。

除数	被除数/商数	余数	
2	100		最低位 ↑ 次高位
2	50	0	
2	25	0	
2	12	1	
2	6	0	
2	3	0	
	1	1	

图 1.1 除 2 取余法

结果得:100 = 1100100B。

② 同样的方法可将十进制数转换成十六进制数,采用除 16 取余法。即将十进制整数除以 16,得到一个商数和余数,再将商数除以 16 又得到一个商数和余数,如此继续除下去直至除尽为止。以最后所得的商数作为最高位,将各次所得的余数按“后得先排”的顺序写下来,即得十六进制转换结果。

【例 1.2】 35243 = (?) H

转换过程如图 1.2 所示。

除数	被除数/商数	余数	
16	35243		最低位 ↑ 次高位
16	2202	11	
16	137	10	
	8	9	

图 1.2 除 16 取余法

结果得:35243 = 89ABH。

③ 将十进数小数部分转换成二进制小数,采用乘 2 取整法。即将十进制纯小数乘以 2,摘除乘积中的整数后保留小数部分再乘 2,如此继续下去直至乘积小数部分为零或者得到要求的精度为止。将各次摘取的整数依先后顺序写出来即为转换的二进制纯小数结果。

【例 1.3】 0.1875 = (?) B

转换过程如图 1.3 所示。

	积的整数部分	被乘数/积的小数部分	乘数
高位 ↓ 低位		0.1875	2
	0	0.375	2
	0	0.75	2
	1	0.5	2
	1	0.0	

图 1.3 乘 2 取整法

结果得: $0.1875 = 0.0011\text{B}$ 。

④ 将十进数小数部分转换为十六进制小数,采用乘 16 取整法。即将十进制纯小数乘以 16,摘除乘积中的整数后保留小数部分再乘 16,如此继续下去直至乘积小数部分为零或者得到要求的精度为止。将各次摘取的整数依先后顺序写出来即为转换的二进制纯小数结果。

【例 1.4】 $0.78125 = (?)\text{H}$ 。

转换过程如图 1.4 所示。

	积的整数部分	被乘数/积的小数部分	乘数
高位 ↓ 低位		0.78125	16
	12	0.5	16
	8	0.0	

图 1.4 乘 16 取整法

结果得: $0.78125 = 0.\text{C8H}$ 。

2. 十六进制数与二进制数之间的转换

由于十六进制数实际上是 4 位二进制数,所以两者之间的转换是轻而易举的。将二进制数转换成十六进制数时,以小数点为界,整数部分自右至左每 4 位用对应的 1 位十六进制数表示,最高位部分不足 4 位时在左边补 0;而小数部分则自左至右每 4 位用对应的 1 位十六进制数表示,最低位部分不足 4 位时在右边补 0。例如:

$$\begin{array}{ccccccc}
 1101 & 1010 & 0011 & . & 0110 & 11 & \text{B} \\
 = \text{D} & \text{A} & 3 & . & 6 & \text{C} & \text{H}
 \end{array}$$

注意,不要将十六进制数小数点最后一位“C”误写为“3”。

由此可见,引入十六进制数的主要目的是为了免除书写与阅读一长串二进制数码的麻烦,克服容易出错的弊病,计算机本身并不需要转换运算。

如果要将十六进制数转换成二进制数,只需将十六进制数的每 1 位用对应的 4 位二进制数表示,并按原顺序排列即可。例如:

$$\begin{array}{ccccccc}
 5 & \text{F} & 8 & 4 & . & \text{E} & 4 & \text{H} \\
 = 0101 & 1111 & 1000 & 0100 & . & 1110 & 0100 & \text{B}
 \end{array}$$

1.2 常用的编码

计算机总是以量化的数据形式来进行操作的,但它处理的对象不一定是“数”,例如,我们现在阅读的就是“字符”。因此采用一些标准的二进制编码来表示部分常用的对象或信息就会方便得多,这里介绍最常用的 BCD 码与 ASCII 码。

1.2.1 BCD 码

人们已经习惯了十进制数,而且计算机的原始数据多数也是十进制的,但十进制数不能

直接在计算机中进行处理,必须用二进制为它编码,这样就产生了二进制编码的十进制数,简称 BCD(Binary Coded Decimal)码。

BCD 码用 4 位二进制数表示一位十进制数,但这 4 位二进制数中可表示的 16 个数码中有 6 个数码是多余的,应该抛弃,可以使用不同的方法来处理这些数码,因而产生了各种不同的 BCD 码,但最通用的是 8421 BCD 码,它是将十六进制数的 A~F 放弃不用。例如:

89 的 BCD 码为 1000 1001;

105 的 BCD 码为 0001 0000 0101;

2004 的 BCD 码为 0010 0000 0000 0100。

可见,BCD 码是很容易编制的,而且用它来表示十进制数也很直观,但是一定要区别于二进制数,两者表征的数值完全不同,例如:

$(0010000000000001.1001)_{\text{BCD}} = 2001.9$

$0010000000000001.1001\text{B} = 8193.5625$

以上用 4 位二进制表示 1 位十进制的编码称为紧(压)缩型 BCD 码(Packed BCD),此时用 8 位二进制就能表示 2 位十进制。将一个字节分成高半字节与低半字节,各表示一位十进制数,CPU 通过对一个专门针对半字节的辅助进位的调整就可对 BCD 数直接进行运算。

如果用 8 位二进制来表示 1 位十进制,则称为非紧(压)缩型 BCD 码(Unpacked BCD),此时它的高 4 位全为 0。

例如,86 的紧缩型 BCD 码是 1000 0110B,它的非压缩型 BCD 码是 0000 1000 0000 0110B。

BCD 码的不足之处是抛弃了二进制中 6/16 的信息位不使用,非压缩的 BCD 码浪费更大,在相同的二进制位数条件下 BCD 能表示的数值范围变窄。换言之,如果信息量相同的话,那么使用 BCD 数据占用的内存空间比使用纯二进制数据要大得多。

1.2.2 ASCII 码与奇偶校验

计算机不仅仅只对数据进行运算,而且还要处理其他各种事务,因而它应能识别字母与各种字符,目前普遍使用 ASCII 码来表示字母与字符。

ASCII(American Standard Code for Information Interchange)码,即美国标准信息交换码,这是一种 7 位的二进制编码,可表示 128 个字符,包括英文大小写字母与数字 0~9,表 1.2 列出了全部 ASCII 码及其表示方法。

表 1.2 ASCII 码表

$b_6b_5b_4$ $b_3b_2b_1b_0$		0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	\$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u

(续表)

$b_6b_5b_4$		0	1	2	3	4	5	6	7
$b_3b_2b_1b_0$		000	001	010	011	100	101	110	111
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O	_	o	DEL

由于存储器的基本单元是 8 位一个字节,故通常还是用一个字节来表示 ASCII 码,并认为最高位 b_7 恒为零,于是 0~9 的 ASCII 码为 30H~39H,大写英文字母 A~Z 的 ASCII 码为 41H~5AH,小写英文字母 a~z 的 ASCII 码为 61H~7AH 等。

必须指出的是,在许多实际应用中,最高位 b_7 又常常用来作为 ASCII 码的奇/偶校验位。奇校验时该位的取值应使得 8 位 ASCII 码中 1 的个数为奇数;反之,偶校验时该位的取值应使得 8 位 ASCII 码中 1 的个数为偶数。例如:

- “8”的奇校验 ASCII 码为 00111000B,偶校验 ASCII 码为 10111000B;
- “B”的奇校验 ASCII 码为 11000010B,偶校验 ASCII 码为 01000010B。

奇偶校验的主要目的是用于数据传输中检测接收方的数据是否正确。收发双方先预约为何种校验,接收方收到数据后检验 1 的个数,判断是否与预约的校验相符,倘若不符,则说明传输出错,可请求重新发送。

1.3 带符号数

如同用“+”、“-”符号表示正数与负数一样,计算机用最高位作为符号位,该位为 0 表示正数,该位为 1 表示负数。这样在一个字节的 8 位二进制中就只有 7 位表示数值了,这种数就称为带符号的数。在此之前介绍的数均未涉及符号这一概念,它们是不带符号的数,两者在表达形式上看不出任何差别,但它们所表示的数值是完全不同的。

计算机中的数有原码、反码、补码三种表示方法,而且一般都是采用补码表示法,由于三者有不可分割的联系,下面将逐一介绍。

1.3.1 原码、反码与补码

1. 原码

原码是在原来二进制数值位中用最高位作正负符号位,其他位均为数值位,其值不变,

最高位为“0”表示正数,最高位为“1”表示负数。例如:

- +1010101B 的原码为 01010101B;
- -1010101B 的原码为 11010101B。

2. 反码

对于负数而言,最高位始终为“1”,数值位则与原码相反,即“0”变成“1”,“1”变成“0”。

例如:

-1010101B 的原码表示法为 11010101B,反码表示法则是 10101010B。

3. 补码

在反码的最低位加“1”,就成为补码表示法,例如:

原码 11010101B 的反码是 10101010B,补码是 10101011B。

值得指出的是,对于正数来说,其最高位保持为“0”,不引入反码与补码之说,换言之,正数只采用原码表示法。

由于计算机的数字逻辑电路对于“取反”、“加1”、“进位”等操作轻而易举,所以,引入补码为计算机的运算操作提供了极大方便,提高了机器的运算效率。例如采用补码就可以将减法当加法来操作。

【例 1.5】 $127 - 16 = ?$

因为 $127 - 16 = 127 + (-16)$,所以只需将 127 的原码与 -16 的补码相加:

$$\begin{array}{r} 01111111 \\ + 11110000 \\ \hline = 10110111 \end{array}$$

忽略进位即为运算结果。验算

$$01101111\text{B} = 6\text{FH} = 6 \times 16 + 15 = 111$$

1.3.2 补码的换算

对正数求补可得到它对应的负数,如果再对该负数求补又可以得到正数,也就是得到了绝对值。因此补码的换算并不复杂。

当要将一个十进制负数转换为补码表示时,可按以下步骤进行:

- ① 将该负数对应的正数转换成二进制数;
- ② 将该二进制数按位取反后末位加1。

【例 1.6】 $-100 = (?)\text{B}$

因为 $+100 = 64\text{H} = 01100100\text{B}$,所以

$$\begin{array}{r} 01100100 \\ 10011011 \text{ ;取反} \\ + \quad \quad \quad 1 \text{ ;加1} \\ \hline = 10011100 \end{array}$$

结果得: $-100 = 10011100\text{B} = 9\text{CH}$

反过来,已知二进制的补码,将它转换为十进制符号数的方法是:

- ① 判断最高位,即符号位是0还是1。如果符号位是0,说明是正数,此时不需做任何