



x86 汇编语言

从实模式到保护模式

李忠 王晓波 余洁◎著

- 你想知道汇编语言到底是什么吗？
- 你想探究x86处理器的内部原理吗？
- 你想从侧面理解操作系统的工作原理，并尝试打造一个自己的系统吗？
- 这本书将会带你学习和掌握汇编语言程序设计的基本方法，了解汇编语言和处理器、计算机系统以及操作系统之间的关系。



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

x86 汇编语言

从实模式到保护模式

李 忠 王晓波 余 洁 著

电子工业出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书采用开源的 NASM 汇编语言编译器和 VirtualBox 虚拟机软件,以个人计算机广泛采用的 Intel 处理器为基础,详细讲解了 Intel 处理器的指令系统和工作模式,以大量的代码演示了 16 / 32 / 64 位软件的开发方法,集中介绍处理器的 16 位实模式和 32 位保护模式,以及基本的指令系统。

这是一本有趣的书,它没有把篇幅花在计算一些枯燥的数学题上。相反,它教你如何直接控制硬件,在不借助于 BIOS、DOS、Windows、Linux 或任何其他软件支持的情况下来显示字符、读取硬盘数据、控制其他硬件等。本书可作为大专院校相关专业学生和计算机编程爱好者的教程。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

x86 汇编语言:从实模式到保护模式 / 李忠, 王晓波, 余洁著. —北京: 电子工业出版社, 2013.1
ISBN 978-7-121-18799-5

I. ①x… II. ①李… ②王… ③余… III. ①汇编语言—程序设计 IV. ①TP313

中国版本图书馆 CIP 数据核字(2012)第 253290 号

责任编辑:董亚峰 特约编辑:王 纲

印 刷:涿州市京南印刷厂

装 订:涿州市京南印刷厂

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本:787×1 092 1/16 印张:24.25 字数:620 千字

印 次:2013 年 1 月第 1 次印刷

定 价:56.00 元



凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010) 88258888。

前言

尽管汇编语言也是一种计算机语言，但却是与众不同的，与它的同类们格格不入。处理器的工作是执行指令并获得结果，而为了驾驭处理器，汇编语言为每一种指令提供了简单好记、易于书写的符号化表示形式。

一直以来，人们对于汇编语言的认识和评价可以分为两种，一种是觉得它非常简单，另一种是觉得它学习起来非常困难。

你认为我会赞同哪一种？说汇编语言难学，这没有道理。学习任何一门计算机语言，都需要一些数制和数制转换的知识，也需要大体上懂得计算机是怎么运作的。在这个前提下，汇编语言是最贴近硬件实体的，也是最自然和最朴素的。最朴素的东西反而最难掌握，这实在说不过去。因此，原因很可能出在我们的教科书上，那些一上来就搞一大堆寻址方式的书，往往以最快的速度打败了本来激情高昂的初学者。

但是，说汇编语言好学，也同样有些荒谬。据我的观察，很多人掌握了若干计算机指令，会编写一个从键盘输入数据，然后进行加减乘除或者归类排序的程序后，就认为自己掌握了汇编语言。还有，直到现在，我还经常在网上看到学生们使用 DOS 中断编写程序，他们讨论的也大多是实模式，而非 32 位或者 64 位保护模式。他们知道如何编译源程序，也知道在命令行输入文件名，程序就能运行了；又或者使用一个中断，就能显示字符。至于这期间发生了什么，程序是如何加载到内存中的，又是怎么重定位的，似乎从来不关汇编语言的事。这样做的结果，就是让人以为汇编语言不过如此，而且非常枯燥。

很难说我已经掌握了汇编语言的要义。但至少我知道，尽管汇编语言不适合用来编写大型程序，但它对于理解计算机原理很有帮助，特别是处理器的工作原理和运行机制。就算是为了这个目的，也应该让汇编语言回归它的本位，那就是访问和控制硬件（包括处理器），而不仅仅是编写程序，输入几个数字，找出正数有几个、负数有几个，大于 30 的有几个。

事实上，汇编语言对学习和理解高级语言，比如 C 语言，也有极大的帮助。老教授琢磨了好几天，终于想到一个好的比喻来帮助学生理解什么是指针，实际上，这对于懂得汇编语言的学生来说，根本就不算个事儿，并因此能够使老教授省下时间来喝茶。

在这本书之前，我也写过《穿越计算机的迷雾》一书。它们是一个系列，没有基础的读者可以先看那本书，打一点计算机原理的基础再来学习汇编语言。

在计划写这本书的时候，我就给自己画了几条线。首先不能走老路，一上来就讲指令、寻址方式，而是采用任务驱动的方式来写，每一章都要做点事情，最好是比较有趣，有吸引力。在解决问题的过程中，不断地引入新指令，并进行讲解。一句话，我希望是润物细

无声式的；其次，汇编语言和硬件并举，完全抛弃 BIOS 中断和 DOS 中断，直接访问硬件，发挥汇编语言的长处，因为这才是我们学习汇编语言的目的。也只有这样，读者才能深刻体会到汇编语言的妙处。

王晓波和湖北经济学院的余洁共同参与了本书的创作。

本书主要讲述 INTEL x86 处理器的 16 位实模式、32 位保护模式，至于虚拟 8086 模式，则是为了兼容传统的 8086 程序，现在看来已经完全过时，不再进行讲述。本书的特色之一是提供了大量典型的源代码，这些代码以及相配套的工具程序可以到书中指定的网站，或者电子工业出版社华信教育资源网搜索下载。

很多读者在读书的时候会遇到这种情况：一开始读得很快，一口气读了好几章。随着内容的深入，学习越来越吃力，不得不频繁回到前面重新学习已经讲过的内容，这就是因为前面的知识没有完全掌握。为此，本书每一章都设有检测点，读者应当在通过检测点之后再继续往后阅读。

本书原来有 18 章，后来考虑到实模式的内容过多，而去掉了一章。这一章的标题是《聆听数字的声音》，讲述如何通过直接访问和控制 Sound Blaster 16 声卡来播放声音，感兴趣的朋友可以从下载的配书文件包中找到这部分内容。

在本书的写作和出版过程中，长春电视台台长王志强，副台长周武军和技术部主任刘贵先后对本书给予了关心和支持，在此表示衷心的感谢。

好友王南洋、桑国伟、刘维钊、蒋胜友、邱海龙、万利、李文心等负责了本书的一部分校对工作；好友周卫平帮我验证配书代码是否能在他的机器上正常工作，在这里向他们表示感谢，同时也谢谢所有关心和支持本书的朋友们。

感谢我的母亲、我的妻子和我的女儿，她们是我的精神支柱，是我努力创作这本书的动力来源。

在阅读本书的过程中，如果有任何问题，可以往电子邮件地址 leechung@126.com 给我写信；要了解其他更多的情况，请访问我的博客：<http://blog.163.com/leechung@126>。



二〇一二年十一月

目 录

第1部分 预备知识

第 1 章 十六进制计数法	3
1.1 二进制计数法回顾.....	3
1.1.1 关于二进制计数法.....	3
1.1.2 二进制到十进制的转换.....	4
1.1.3 十进制到二进制的转换.....	4
1.2 十六进制计数法.....	5
1.2.1 十六进制计数法的原理.....	5
1.2.2 十六进制到十进制的转换.....	6
1.2.3 十进制到十六进制的转换.....	6
1.2.4 为什么需要十六进制.....	6
1.3 使用 Windows 计算器方便你的学习过程.....	8
本章习题.....	9
第 2 章 处理器、内存和指令	10
2.1 最早的处理器.....	10
2.2 寄存器和算术逻辑部件.....	10
2.3 内存存储器.....	12
2.4 指令和指令集.....	14
2.5 古老的 Intel 8086 处理器.....	16
2.5.1 8086 的通用寄存器.....	16
2.5.2 程序的重定位难题.....	16
2.5.3 内存分段机制.....	19
2.5.4 8086 的内存分段机制.....	21
本章习题.....	24
第 3 章 汇编语言和汇编软件	25
3.1 汇编语言简介.....	25
3.2 NASM 编译器.....	27
3.2.1 NASM 的下载和安装.....	27

3.2.2	代码的书写和编译过程	27
3.2.3	用 HexView 观察编译后的机器代码	30
	本章习题	31
第 4 章	虚拟机的安装和使用	32
4.1	计算机的启动过程	32
4.1.1	如何将编译好的程序提交给处理器	32
4.1.2	计算机的加电和复位	33
4.1.3	基本输入输出系统	33
4.1.4	硬盘及其工作原理	34
4.1.5	一切从主引导扇区开始	36
4.2	创建和使用虚拟机	37
4.2.1	别害怕, 虚拟机是软件	37
4.2.2	下载和安装 Oracle VM VirtualBox	37
4.2.3	虚拟硬盘简介	39
4.2.4	练习使用 FixVhdWr 工具向虚拟硬盘写数据	40

第 2 部分 实模式

第 5 章	编写主引导扇区代码	45
5.1	本章代码清单	45
5.2	欢迎来到主引导扇区	45
5.3	注释	46
5.4	在屏幕上显示文字	46
5.4.1	显卡和显存	46
5.4.2	初始化段寄存器	49
5.4.3	显存的访问和 ASCII 代码	49
5.4.4	显示字符	51
5.4.5	MOV 指令的格式	52
5.5	显示标号的汇编地址	54
5.5.1	标号	54
5.5.2	如何显示十进制数字	58
5.5.3	在程序中声明并初始化数据	58
5.5.4	分解数的各个数位	59
5.5.5	显示分解出来的各个数位	63
5.6	使程序进入无限循环状态	64
5.7	完成并编译主引导扇区代码	66
5.7.1	主引导扇区有效标志	66
5.7.2	代码的保存和编译	67
5.8	加载和运行主引导扇区代码	67
5.8.1	把编译后的指令写入主引导扇区	67
5.8.2	启动虚拟机观察运行结果	68
5.9	程序的调试技术	68
5.9.1	开源的 Bochs 虚拟机软件	68

5.9.2	Bochs 下的程序调试入门	69
	本章习题	75
第 6 章	相同的功能，不同的代码	76
6.1	代码清单 6-1	76
6.2	跳过非指令的数据区	76
6.3	在数据声明中使用字面值	77
6.4	段地址的初始化	77
6.5	段之间的批量数据传送	78
6.6	使用循环分解数位	80
6.7	计算机中的负数	81
6.7.1	无符号数和有符号数	81
6.7.2	处理器视角中的数据类型	85
6.8	数位的显示	87
6.9	其他标志位和条件转移指令	88
6.9.1	奇偶标志位 PF	88
6.9.2	进位标志 CF	89
6.9.3	溢出标志 OF	89
6.9.4	现有指令对标志位的影响	90
6.9.5	条件转移指令	90
6.10	NASM 编译器的\$和\$\$标记	92
6.11	观察运行结果	93
6.12	本章程序的调试	93
6.12.1	调试命令“n”的使用	93
6.12.2	调试命令“u”的使用	94
6.12.3	用调试命令“info”察看标志位	96
	本章习题	97
第 7 章	比高斯更快的计算	98
7.1	从 1 加到 100 的故事	98
7.2	代码清单 7-1	98
7.3	显示字符串	98
7.4	计算 1 到 100 的累加和	99
7.5	累加和各个数位的分解与显示	99
7.5.1	栈和栈段的初始化	99
7.5.2	分解各个数位并压栈	101
7.5.3	出栈并显示各个数位	103
7.5.4	进一步认识栈	104
7.6	程序的编译和运行	105
7.6.1	观察程序的运行结果	105
7.6.2	在调试过程中察看栈中内容	106
7.7	8086 处理器的寻址方式	107
7.7.1	寄存器寻址	107

7.7.2	立即寻址	107
7.7.3	内存寻址	108
	本章习题	112
第 8 章	硬盘和显卡的访问与控制	113
8.1	本章代码清单	114
8.2	用户程序的结构	114
8.2.1	分段、段的汇编地址和段内汇编地址	114
8.2.2	用户程序头部	117
8.3	加载程序(器)的工作流程	120
8.3.1	初始化和决定加载位置	120
8.3.2	准备加载用户程序	121
8.3.3	外围设备及其接口	122
8.3.4	I/O 端口和端口访问	123
8.3.5	通过硬盘控制器端口读扇区数据	125
8.3.6	过程调用	127
8.3.7	加载用户程序	133
8.3.8	用户程序重定位	134
8.3.9	将控制权交给用户程序	137
8.3.10	8086 处理器的无条件转移指令	138
8.4	用户程序的工作流程	140
8.4.1	初始化段寄存器和栈切换	140
8.4.2	调用字符串显示例程	141
8.4.3	过程的嵌套	142
8.4.4	屏幕光标控制	142
8.4.5	取当前光标位置	143
8.4.6	处理回车和换行字符	144
8.4.7	显示可打印字符	145
8.4.8	滚动屏幕内容	145
8.4.9	重置光标	146
8.4.10	切换到另一个代码段中执行	146
8.4.11	访问另一个数据段	147
8.5	编译和运行程序并观察结果	147
	本章习题	148
第 9 章	中断和动态时钟显示	149
9.1	外部硬件中断	149
9.1.1	非屏蔽中断	150
9.1.2	可屏蔽中断	150
9.1.3	实模式下的中断向量表	152
9.1.4	实时时钟、CMOS RAM 和 BCD 编码	154
9.1.5	代码清单 9-1	157
9.1.6	初始化 8259、RTC 和中断向量表	157

9.1.7	使处理器进入低功耗状态	159
9.1.8	实时时钟中断的处理过程	160
9.1.9	代码清单 9-1 的编译和运行	162
9.2	内部中断	163
9.3	软中断	163
9.3.1	BIOS 中断	163
9.3.2	代码清单 9-2	165
9.3.3	从键盘读字符并显示	165
9.3.4	代码清单 9-2 的编译和运行	165
	本章习题	166

第 3 部分 32 位保护模式

第 10 章	32 位 x86 处理器编程架构	169
10.1	IA-32 架构的基本执行环境	169
10.1.1	寄存器的扩展	169
10.1.2	基本的工作模式	172
10.1.3	线性地址	173
10.2	现代处理器的结构和特点	174
10.2.1	流水线	174
10.2.2	高速缓存	175
10.2.3	乱序执行	175
10.2.4	寄存器重命名	176
10.2.5	分支目标预测	177
10.3	32 位模式的指令系统	178
10.3.1	32 位处理器的寻址方式	178
10.3.2	操作数大小的指令前缀	179
10.3.3	一般指令的扩展	181
	本章习题	184
第 11 章	进入保护模式	185
11.1	代码清单 11-1	185
11.2	全局描述符表	186
11.3	存储器的段描述符	187
11.4	安装存储器的段描述符并加载 GDTR	191
11.5	关于第 21 条地址线 A20 的问题	193
11.6	保护模式下的内存访问	195
11.7	清空流水线并串行化处理器	199
11.8	保护模式下的栈	200
11.8.1	关于栈段描述符中的界限值	200
11.8.2	检验 32 位下的栈操作	201
11.9	程序的运行和调试	202
11.9.1	运行程序并观察结果	202
11.9.2	处理器刚加电时的段寄存器状态	203

11.9.3	设置 PE 位后的段寄存器状态	205
11.9.4	JMP 指令执行后的段寄存器状态	205
11.9.5	察看全局描述符表 GDT	206
11.9.6	察看控制寄存器的内容	207
	本章习题	207
第 12 章	存储器的保护	208
12.1	代码清单 12-1	208
12.2	进入 32 位保护模式	208
12.2.1	话说 mov ds,ax 和 mov ds,eax	208
12.2.2	创建 GDT 并安装段描述符	209
12.3	修改段寄存器时的保护	211
12.4	地址变换时的保护	213
12.4.1	代码段执行时的保护	213
12.4.2	栈操作时的保护	214
12.4.3	数据访问时的保护	216
12.5	使用别名访问代码段对字符排序	217
12.6	程序的编译和运行	219
	本章习题	220
第 13 章	程序的动态加载和执行	221
13.1	本章代码清单	222
13.2	内核的结构、功能和加载	222
13.2.1	内核的结构	222
13.2.2	内核的加载	223
13.2.3	安装内核的段描述符	225
13.3	在内核中执行	229
13.4	用户程序的加载和重定位	230
13.4.1	用户程序的结构	230
13.4.2	计算用户程序占用的扇区数	232
13.4.3	简单的动态内存分配	233
13.4.4	段的重定位和描述符的创建	234
13.4.5	重定位用户程序内的符号地址	238
13.5	执行用户程序	242
13.6	代码的编译、运行和调试	243
	本章习题	244
第 14 章	任务和特权级保护	245
14.1	任务的隔离和特权级保护	246
14.1.1	任务、任务的 LDT 和 TSS	246
14.1.2	全局空间和局部空间	248
14.1.3	特权级保护概述	250
14.2	代码清单 14-1	257
14.3	内核程序的初始化	257

14.3.1	调用门	258
14.3.2	调用门的安装和测试	261
14.4	加载用户程序并创建任务	264
14.4.1	任务控制块和 TCB 链	264
14.4.2	使用栈传递过程参数	266
14.4.3	加载用户程序	268
14.4.4	创建局部描述符表	269
14.4.5	重定位 U-SALT 表	270
14.4.6	创建 0、1 和 2 特权级的栈	271
14.4.7	安装 LDT 描述符到 GDT 中	271
14.4.8	任务状态段 TSS 的格式	272
14.4.9	创建任务状态段 TSS	276
14.4.10	安装 TSS 描述符到 GDT 中	276
14.4.11	带参数的过程返回指令	277
14.5	用户程序的执行	278
14.5.1	通过调用门转移控制的完整过程	278
14.5.2	进入 3 特权级的用户程序的执行	281
14.5.3	检查调用者的请求特权级 RPL	284
14.5.4	在 Bochs 中调试程序的新方法	286
	本章习题	286
第 15 章	任务切换	287
15.1	本章代码清单	287
15.2	任务切换前的设置	287
15.3	任务切换的方法	289
15.4	用 call/jmp/iret 指令发起任务切换的实例	292
15.5	处理器在实施任务切换时的操作	296
15.6	程序的编译和运行	298
	本章习题	299
第 16 章	分页机制和动态页面分配	300
16.1	分页机制概述	301
16.1.1	简单的分页模型	301
16.1.2	页目录、页表和页	305
16.1.3	地址变换的具体过程	307
16.2	本章代码清单	308
16.3	使内核在分页机制下工作	309
16.3.1	创建内核的页目录表和页表	309
16.3.2	任务全局空间和局部空间的页面映射	314
16.4	创建内核任务	319
16.4.1	内核的虚拟内存分配	319
16.4.2	页面位映射串和空闲页的查找	320
16.4.3	创建页表并登记分配的页	323

16.4.4	创建内核任务的 TSS	324
16.5	用户任务的创建和切换	325
16.5.1	多段模型和段页式内存管理	325
16.5.2	平坦模型和用户程序的结构	327
16.5.3	用户任务的虚拟地址空间分配	328
16.5.4	用户程序的加载	329
16.5.5	段描述符的创建(平坦模型)	332
16.5.6	重定位 U-SALT 并复制页目录表	333
16.5.7	切换到用户任务执行	334
16.6	程序的编译、执行和调试	336
16.6.1	本程序的编译和运行方法	336
16.6.2	察看 CR3 寄存器的内容	337
16.6.3	察看线性地址对应的物理页信息	337
16.6.4	察看当前任务的页表信息	338
16.6.5	使用线性(虚拟)地址调试程序	339
本章习题		339
第 17 章	中断和异常的处理与抢占式多任务	340
17.1	中断和异常	340
17.1.1	中断和异常概述	340
17.1.2	中断描述符表、中断门和陷阱门	343
17.1.3	中断和异常处理程序的保护	345
17.1.4	中断任务	347
17.1.5	错误代码	348
17.2	本章代码清单	349
17.3	内核的加载和初始化	349
17.3.1	彻底终结多段模型	349
17.3.2	创建中断描述符表	352
17.3.3	用定时中断实施任务切换	354
17.3.4	8259A 芯片的初始化	359
17.3.5	平坦模型下的字符串显示例程	362
17.4	内核任务的创建	362
17.4.1	创建内核任务的 TCB	362
17.4.2	宏汇编技术	364
17.5	用户任务的创建	366
17.5.1	准备加载用户程序	366
17.5.2	转换后援缓冲器的刷新	367
17.5.3	用户任务的创建和初始化	368
17.6	程序的编译和执行	370
本章习题		371
附录 I	本书用到的 x86 指令及其页码	372
附录 II	本书用到的重要图表及其页码	374

第 1 部分

预备知识

- ◇ 了解数制的基本知识和数制转换的方法。
- ◇ 了解 8086 处理器的结构和工作方式，初步认识所谓的针对处理器编程，是针对处理器的哪些部件和哪些方面进行的，理解分段的原理。
- ◇ 了解什么是汇编语言，以及如何书写、编译汇编语言源程序，掌握在虚拟机上运行程序的方法。

第1章 十六进制计数法

电子计算机，顾名思义，就是计算的机器。因此，学习汇编语言，就不可避免地要和数字打交道。在这个过程中，我们要用到三种数制：十进制（这是我们再熟悉不过的）、二进制和十六进制。本章的目标是：

1. 熟悉后两种数制，了解这两种数制的计数特点。
2. 能够在这三种数制之间熟练地进行转换，特别是看到一个二进制数时，能够口算出它对应的十六进制数，反之亦然。
3. 对于0~15之间的任何一个十进制数，能够立即说出它对应的二进制数和十六进制数。

1.1 二进制计数法回顾

1.1.1 关于二进制计数法

在《穿越计算机的迷雾》那本书里我们已经知道，计算机也是一台机器，唯一不同的地方在于它能计算数学题，且具有逻辑判断能力。

与此同时，我们也已经在那本书里学到，机器在做数学题的时候，也面临着一个如何表示数字的问题，比如你采用什么办法来将加数和被加数送到机器里。

同样是在那本书里，我们揭晓了答案，那就是用高、低两种电平的组合来表示数字。如图 1-1 所示，参与计算的数字通过电线送往计算机，高电平被认为是“1”，低电平被认为是“0”，这样就形成了一个序列“11111010”，这就是一个二进制数，在数值上等于我们所熟知的二百五，换句话说，等于十进制数 250。

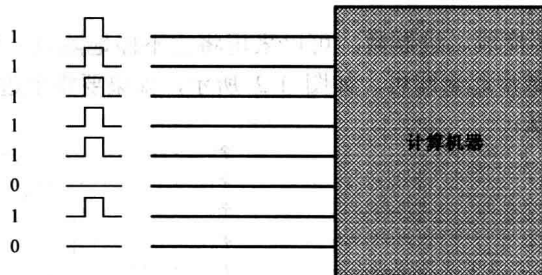


图 1-1 在计算机里，二进制数字对应着高低电平的组合

从数学的角度来看，二进制计数法是现代主流计算机的基础。一方面，它简化了硬件设计，因为它只有两个符号“0”和“1”，要得到它们，可以用最少的电路元件来接通或者关断电路就行了；另一方面，二进制数与我们熟悉的十进制数之间有着一对一的关系，任何一个十进制数都对应着一个二进制数，不管它有多大。比如，十进制数 5，它所对应的二进制数是 101，而十进制数 5785478965147 则对应着一长串“0”和“1”的组合，即 1010100001100001001011010110010011110011011。

组成二进制数的每一个数位，称为一个比特（bit），而一个二进制数也可以看成是一个比特串。很明显，它的数值越大，这个比特串就越长，这是二进制计数法不好的一面。

1.1.2 二进制到十进制的转换

每一种计数法都有自己的符号（数符）。比如，十进制有 0、1、2、3、4、5、6、7、8、9 这十个符号；二进制呢，则只有 0、1 这两个符号。这些数字符号的个数称为基数。也就是说，十进制有 10 个基数，而二进制只有两个。

二进制和十进制都是进位计数法。进位计数法的一个特点是，符号的值和它在这个数中所处的位置有关。比如十进制数 356，数字 6 处在个位上，所以是“6 个”；5 处在十位上，所以是“50”；3 处在百位上，所以是“300”。即：

$$\text{百位 } 3、\text{十位 } 5、\text{个位 } 6 = 3 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 = 356$$

这就是说，由于所处的位置不同，每个数位都有一个不同的放大倍数，这称为“权”。每个数位的权是这样计算的（这里仅讨论整数）：从右往左开始，以基数为底，指数从 0 开始递增的幂。正如上面的公式所清楚表明的那样，“6”在最右边，所以它的权是以 10 为底，指数为 0 的幂 10^0 ；而 3 呢，它的权则是以 10 为底，指数为 2 的幂 10^2 。

上面的算式是把十进制数“翻译”成十进制数。从十进制数又算回到十进制数，这看起来有些可笑，注意这个公式是可以推广的，可以用它来将二进制数转换成十进制数。

比如一个二进制数 10110001，它的基数是 2，所以要这样来计算与它等值的十进制数：

$$10110001\text{B} = 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 177\text{D}$$

在上面的公式里，10110001B 里的“B”表示这是一个二进制数，“D”则表示 177 是个十进制数。“B”和“D”分别是英语单词 Binary 和 Decimal 的头一个字母，这两个单词分别表示二进制和十进位的意思。

◆ 检测点 1.1

将下列二进制数转换成十进制数：

1101、1111、1001110、11111111、10000000、1101101100011011

1.1.3 十进制到二进制的转换

为了将一个十进制数转换成二进制数，可以采用将它不停地除以二进制的基数 2，直到商为 0，然后将每一步得到的余数串起来即可。如图 1-2 所示，如果要将十进制数 26 转换成二进制数 11010，那么可采用如下方法：

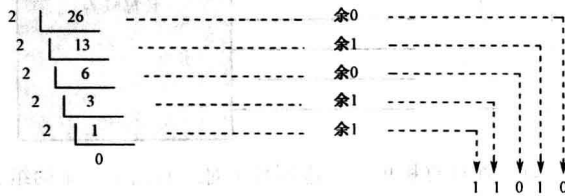


图 1-2 将十进制数 26 转换成二进制数

第 1 步，将 26 除以 2，商为 13，余数为 0；

第 2 步，用 13 除以 2，商为 6，余数为 1；

第 3 步，用 6 除以 2，商为 3，余数为 0；

第 4 步，用 3 除以 2，商为 1，余数为 1；

第 5 步，用 1 除以 2，商为 0，余数为 1，结束。

然后，从下往上，将每一步得到的余数串起来，从左往右书写，就是我们所要转换的二进制数。