

计算机基础教材

孙俊杰 任天平 编著
白国长 门茂琛

微型计算机 原理及应用

中国计算机教育

教育部 教育部
教育部 教育部

微型计算机

原理及应用

中国计算机教育

计算机基础教材

孙俊杰 任天平 编著
白国长 门茂琛

微型计算机 原理及应用



郑州大学出版社

图书在版编目(CIP)数据

微型计算机原理及应用/孙俊杰,任天平,白国长等编著. —郑州:
郑州大学出版社,2005.9
ISBN 7-81106-115-5

I. 微… II. ①孙…②任…③白… III. 微型计算机-高等学校-教材 IV. TP36

中国版本图书馆 CIP 数据核字 (2005) 第 079330 号

郑州大学出版社出版发行

郑州市大学路 40 号

出版人:邓世平

全国新华书店经销

新乡市凤泉印务有限公司印制

开本:787 mm × 1 092 mm

印张:18.25

字数:458 千字

版次:2005 年 9 月第 1 版

邮政编码:450052

发行部电话:0371-66966070

1/16

印数:1~3 100

印次:2005 年 9 月第 1 次印刷

书号:ISBN 7-81106-115-5/T·14

定价:28.00 元

本书如有印装质量问题,请向本社调换

内容提要

本书以主流机 IBM PC 系列及兼容机为主要对象,系统地叙述了微型计算机的组成及各组成部分的工作原理;叙述了汇编语言程序设计的思路、方法和技术;阐述了微型计算机的接口技术及应用。全书共分 8 章,内容包括计算机基础,微处理器,存储器,8086/8088 的结构,指令系统和汇编语言语法,分支、循环、子程序的设计,DOS 系统功能调用,ROM BIOS 中断调用,磁盘文件管理等。涵盖了几乎所有常用典型接口技术,包括存储器接口、并行接口、人-机接口、串行通信接口、D/A 和 A/D 接口、硬磁盘机接口,并对接口问题的一些共性技术,如 I/O 接口地址译码、总线、中断、DMA 和定时/计数技术等集中讨论。每章开始列出该章内容提要和学习目标,结尾列有本章内容小结、练习和思考题。

本书既涉及微型计算机的共性技术,也涉及计算机系统中各类常用外部设备的接口技术,内容丰富,层次分明,实例丰富,便于教学、自学和应用。本书既可供高等学校工科计算机和非计算机类有关专业作为本科生、研究生或高层次专业技术培训教材,也可供从事计算机应用与开发的科研及工程技术人员自学参考。

前 言

本书是作者在总结多年教学和科研实践的基础上,吸收国内外先进的理论、方法和技术,经过多次使用和反复修改而成的。

本书在内容组织上既注重全面性和实用性,又强调系统性与新颖性。全书由浅入深、全面系统地介绍了微型计算机的组成、工作原理、接口电路和典型应用等,使读者建立微型计算机系统的整体概念,掌握微型计算机系统软硬件开发的初步方法,了解简单计算机应用系统的工作原理和设计方法。每章中都有大量的例题和综合应用实例。

全书以 8086 CPU 为背景,内容分为三部分:一是计算机硬件的组成和工作原理;二是 8086 CPU 的指令系统及汇编语言程序设计;三是常用 I/O 电路、A/D 和 D/A 电路的工作原理以及扩展的方法。从非计算机专业的特点出发,以信息在微型计算机的流动过程为主线,以集成芯片的外部特性为重点来讲述。

全书由孙俊杰、任天平、白国长、门茂琛编写,由孙俊杰副教授任主编。其中,白国长编写了第一、二、三章,孙俊杰编写了第四、五章,任天平编写了第六、八章,门茂琛编写了第七章。

目 录

绪论	1
第一章 微型计算机基础	2
第一节 计算机中数的表示方法及运算	2
一、计算机中数的表示方法	2
二、数的运算方法	5
三、计算机中常用编码	9
第二节 微型计算机概述	11
一、微型计算机系统的组成	11
二、微型计算机的工作原理	14
三、衡量计算机性能的主要技术指标	14
第二章 微处理器(CPU)	17
第一节 8086 CPU	17
一、8086 CPU 结构	17
二、8086 的工作方式	24
三、8086 的时序	30
第二节 各种常见的 CPU 特点简介	35
一、80286 微处理器	35
二、80386 微处理器	37
三、80486 微处理器	38
四、Pentium 微处理器	39
第三章 存储器	40
第一节 存储器的分类	40
一、分类	40
二、存储器的技术指标	41
第二节 存储器的工作原理	42
一、随机读写存储器(RAM)	43
二、只读存储器(ROM)	46
第三节 8086 的存储器结构	51
一、8086 的存储器结构	51
二、IBM PC/XT 的存储器	53
第四节 CPU 与存储器的连接	55

一、ROM 系统	55
二、RAM 系统	56
第四章 8086 指令系统	59
第一节 概述	59
一、指令格式	59
二、操作数类型	59
第二节 8086 的寻址方式	60
一、立即寻址	60
二、寄存器寻址	60
三、直接寻址	60
四、寄存器间接寻址	61
五、变址寻址	62
六、基址寻址	63
七、基址变址寻址	63
八、串寻址	64
九、端口寻址	64
十、隐含寻址	65
第三节 8086 指令系统	65
一、数据传送指令	65
二、算术运算指令	71
三、逻辑运算和移位指令	77
四、控制转移指令	81
五、串操作指令	88
六、处理器控制指令	92
七、输入输出指令	93
第五章 汇编语言程序设计	97
第一节 汇编语言基本语法	97
一、语句的结构	97
二、语句中的数据项	98
第二节 常用伪指令	101
一、段定义伪指令	101
二、符号定义伪指令	102
三、数据定义伪指令	103
四、过程定义伪指令	104
五、模块定义与连接伪指令	104
六、列表控制伪指令	105
七、宏处理伪指令	105
八、条件汇编伪指令	107
第三节 汇编语言程序结构	108
一、汇编语言程序结构	108

二、汇编语言程序格式	110
三、汇编语言的调试和运行	112
第四节 DOS 和 BIOS 服务程序调用	113
一、DOS 功能调用	113
二、BIOS 调用	117
第五节 汇编语言程序设计	118
一、程序设计步骤	119
二、顺序程序设计	119
三、分支程序设计	121
四、循环程序设计	126
五、子程序设计	131
第六章 输入输出接口电路	141
第一节 I/O 接口概述	141
一、接口技术发展简述	141
二、接口的通用模型	142
三、CPU 与接口之间传送信息的方式	143
四、设计与分析接口电路的基本方法与注意事项	144
五、常用外围接口芯片	145
第二节 I/O 接口电路地址译码技术	145
一、I/O 接口电路的寻址方式	145
二、系统对 I/O 接口地址的分配	146
三、I/O 接口地址译码方法	147
第三节 总线技术	150
一、概述	150
二、微型计算机总线(内总线 I-B)	154
三、外总线(E-B)	160
第四节 并行接口	161
一、并行接口的概念	161
二、硬线连接并行接口	162
三、可编程并行接口 8255A	165
第五节 串行通信接口	181
一、串行通信的基本概念	181
二、串行接口标准	186
三、串行通信接口	192
第七章 CPU 与接口间信息传送及定时/计数器	204
第一节 中断技术	204
一、中断的基本概念	204
二、8086 的中断系统	208
三、8259A 可编程中断控制器	210
四、8259A 在微机系统中的应用	225

第二节 DMA 技术	227
一、DMA 控制器	227
二、DMA 控制器在系统中的使用	235
三、DMA 控制器的应用举例	239
第三节 定时/计数器 8253/8254	242
一、定时/计数器概述	242
二、可编程定时/计数器 8253(8254)	243
三、8253 在系统中的应用	252
第八章 D/A 和 A/D 转换器接口	256
第一节 D/A 转换器接口	256
一、D/A 转换器及其连接特性	256
二、D/A 转换器与微处理器的接口方法	257
三、D/A 转换器接口电路举例	258
四、D/A 转换器的应用	261
第二节 A/D 转换器接口	263
一、A/D 转换器及其连接特性	263
二、A/D 转换器与微处理器的接口方法	264
三、A/D 转换器接口电路举例	266
第三节 微型计算机系统的 A/D、D/A 通道	272
一、多通道模拟开关	272
二、采样保持器	272
三、A/D 通道的结构形式	273
四、D/A 通道的结构形式	274
五、数据采集与数据系统的 A/D、D/A 通道设计	275
第四节 高速微机数据采集系统	279
一、采用 DMA 方式的 A/D 转换器接口电路	279
二、初始化编程	280
参考文献	282

绪 论

从1946年第一台电子计算机问世到现在不过50多年的历史,但它已渗透到国防尖端、工业、农业、企业管理、交通运输、日常生活的各个领域,其作用和成就日益卓越,成为现代工业水平的标志之一。

50多年来,电子计算机的发展,按照组成的逻辑元件的不同已经经历了四代。

1946~1957年为第一代电子管数字计算机。其主要特点是:逻辑元件采用电子管,主存储器采用磁芯、磁鼓,外存储器已开始使用磁带。运算速度为每秒几千次到几万次。主要用途是科学计算。编写程序主要采用机器语言,后期逐渐使用汇编语言。

1958~1964年为第二代晶体管数字计算机。其主要特点是:逻辑元件采用晶体管,主存储器仍采用磁芯,外存储器已开始使用磁盘。运算速度提高到每秒几万次到几十万次。计算机软件有了很大发展,高级语言和编译语言已很普遍。其应用也已扩展到各种数据处理、工业控制。

1964~1970年为第三代数字计算机。其主要特点是:逻辑元件采用中、小规模集成电路,主存储器依然采用磁芯。体积进一步缩小,性能进一步提高。运算速度已达每秒几十万次到几百万次。

1971年以后,数字计算机进入第四代,其标志为逻辑元件全面采用大规模、超大规模集成电路。体积更小,性能更高。

计算机虽然已经历了四代,功能有了极大的扩展,速度、性能有了极大的提高,但计算机的基本工作原理仍然没有改变。

微型计算机(microcomputer)的特征是它的中央处理器(central process unit, CPU)采用大规模集成技术把运算器和控制器集成在一块芯片上,被称为微处理器。

从1971年Intel公司的4004作为微处理器诞生的标志到1985年Intel公司发布80386,短短的15年中,微处理器已经历了4位机、8位机、16位机、32位机,芯片上的集成度已经由4 000个发展到20万个。Intel公司继80386后,又相继发布了80486、Pentium、Pentium II、Pentium III和Pentium IV等微处理器,从而极大地促进了微型计算机的发展。

1981年,IBM公司推出的IBM PC,是计算机发展史上的重要里程碑。从此,计算机的应用从科研、军事领域,迅速推广到工农业、科学文化、社会生活和人们的日常生活等各种领域,极大地推动了经济的发展。

目前,计算机的分类界限越来越模糊,通常以功能和应用面来区分,可分为:

PC机(personal computer)——以Intel 80X86的各类芯片为标志的微型计算机,通常作为网络的节点,广泛用于各种信息处理、文字处理和办公自动化。

工作站(work station)——主要由各种精简指令系列计算机(RISC)芯片构成,用于各种计算机辅助设计、辅助测试等领域,也可作为网络工作站。

网络服务器(server)——目前的网络,主要采用客户/服务器(client/server)结构,一定数量的客户机有一个功能强大的服务器进行服务,以提高效率并降低成本。

大型机、巨型机(main frame)——一般具有大量的终端,广泛用于银行等大型业务处理领域。

第一章 微型计算机基础

【内容提要】

1. 计算机运算基础:数的表示方法——机器数、真值、无符号数、带符号数、原码、反码、补码等;数的编码方法——BCD 码、ASCII 码、汉字码等;数的运算方法——算术运算、逻辑运算。

2. 微型计算机的软硬件系统。

3. 微型计算机工作过程简述。

4. 微型计算机系统主要技术指标。

【学习目标】

1. 掌握计算机中数的表示方式、运算方法及编码系统。

2. 掌握软硬件的概念及其相互关系。

3. 掌握计算机常用术语及主要技术指标。

4. 理解微型计算机的工作过程。

第一节 计算机中数的表示方法及运算

一、计算机中数的表示方法

在数字计算机的设计与使用上,常使用的数制是二进制(B)、十进制(D)、八进制(O)和十六进制(H)。十进制是人们习惯的数制,但在计算机内部几乎没有例外地都采用二进制数。八进制数和十六进制数只是为了将二进制数表示得更简洁时才用到。

在数字计算机中,一位二进制数称为位(bit),它是计算机中信息存储的最小单位。八位二进制数称为一个字节(byte),它是存储器容量的基本单位。字(word)是计算机内部进行数据传递处理的基本单位,通常它与计算机内部的寄存器、数据总线的宽度相一致;一个字所包含的二进制位数称为字长。

这一节将着重讨论带符号数在计算机中的表示方法和运算。

(一) 真值与机器数

对于带符号的二进制数,通常规定一个数的最高位为符号位,符号位用“0”表示正,用“1”表示负。例如: +1000001B 在计算机中表示为 01000001B, -1000001B 在计算机中表示为 11000001B。也就是说,数的符号在机器中数字化了。

一个数在机器中的表示形式,称为机器数。机器数所表示的数值称为机器数的真值,机器数的真值通常用十进制数表示。

(二) 数的小数点表示

计算机中没有专门的器件表示小数点,而是用数的两种不同表示法来表示它的位置,即定点表示法和浮点表示法。所谓定点与浮点是指小数点在数中的位置是固定的还是浮动的。

1. 定点表示法

在定点表示法中,小数点的位置固定。计算机中处理的定点数通常分为纯整数和纯小数两类,纯整数表示为:



纯小数表示为:



如果计算机采用定点整数表示,参与运算的数必须是整数;若参与运算的数是小数,就需要在运算前乘以一个比例因子,将小数放大为整数。如果计算机采用定点小数表示,参与运算的数必须是小数;若参与运算的数是整数,则需要在运算前除以一个比例因子,将整数缩小为小数。

定点数的这两种表示方法,在具体的计算机中均有应用,究竟采用哪一种,事先要约定好。

注意:此处的小数点是假设的。

2. 浮点表示法

任意一个二进制数 N 可表示为:

$$N = S \times 2^J$$

其中, S 称为数 N 的尾数,表示数 N 的全部有效数字,并决定数 N 的精度。 $S = s_f s_1 s_2 \cdots s_n$, s_f 表示尾数的符号,称为尾符; $s_1 s_2 \cdots s_n$ 表示尾数的数值。 J 称为数 N 的阶码,指明小数点的位置,决定数 N 的大小范围。 $J = j_f j_1 j_2 \cdots j_m$, j_f 是阶码的符号位, $j_1 j_2 \cdots j_m$ 表示阶码的数值。

在浮点表示法中,小数点的位置是浮动的,即阶码 J 可取不同的值。如二进制数 110.011 可表示为:

$$N = 110.011 = 1.10011 \times 2 = 11\ 001.1 = \cdots$$

一般地,一个浮点数可表示为阶码和尾数两部分,尾数是纯小数。其形式为:

$$\underbrace{j_f \ j_1 j_2 \cdots j_m}_{\text{阶符 阶码}}; \quad \underbrace{s_f \ s_1 s_2 \cdots s_n}_{\text{尾符 尾数}}$$

如数 $N = 10.10 = 0.1010 \times 2^{10}$ 的浮点表示为:

$$\underbrace{0 \quad 10}_{\text{阶符 阶码}}; \quad \underbrace{0 \quad 1010}_{\text{尾符 尾数}}$$

浮点数所表示的数值的范围远远超过定点数,但是浮点数的算术运算也要复杂得多。微型计算机中这两种表示法都可以用。

(三) 带符号数的表示

带符号数在机器中有 3 种表示方法:原码、反码和补码表示法。

1. 原码表示法

X 的原码定义为

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

由定义可知,原码的性质如下:

(1) 正数的符号位为“0”,其余位是数值位,且表示这个数的真值。

(2) 负数的符号位为“1”，其余位是数值位，且表示这个数的真值。

(3) 当 $X=0$ 时，有 $[+0]_{\text{原}}$ 和 $[-0]_{\text{原}}$ 两种情况：

$$[+0]_{\text{原}} = \underbrace{0000 \cdots 0}_{n \text{ 个 } 0} \quad [-0]_{\text{原}} = \underbrace{1000 \cdots 0}_{n-1 \text{ 个 } 0}$$

例如：若 $n=8, X_1 = 67$ ，则有 $[X_1]_{\text{原}} = 01000011$ ；

$X_2 = -67$ ，则有 $[X_2]_{\text{原}} = 11000011$ 。

原码表示法简单易懂，而且与真值的转换方便。但原码表示的数不便于计算机运算，因为在两原码数运算时，首先要判断它们的符号，然后再决定用加法还是用减法。比起下面要讲的补码表示法烦琐。

2. 反码表示法

X 的反码定义为

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

由定义可知，反码的性质如下：

(1) 正数的反码和原码同形。

(2) 负数的反码符号位为“1”，其余位为对应原码各位取反。

(3) 当 $X=0$ 时，有 $[+0]_{\text{反}}$ 和 $[-0]_{\text{反}}$ 两种情况：

$$[+0]_{\text{反}} = \underbrace{0000 \cdots 0}_{n \text{ 个 } 0} \quad [-0]_{\text{反}} = \underbrace{1111 \cdots 1}_{n \text{ 个 } 1}$$

例如：若 $n=8, X_1 = 67$ ，则有 $[X_1]_{\text{反}} = 01000011$ ；

$X_2 = -67$ ，则有 $[X_2]_{\text{反}} = 10111100$ 。

同样，反码表示的数也不便于计算机运算，所以计算机中也很少用反码表示法表示带符号数。

3. 补码表示法

如果有两个整数 a 和 b ，当除以一正整数 M 时，所得余数相等，则称 a 和 b 对模 M 是同余的。即 a 和 b 在模为 M 时，互为补码。

当 a, b 对模 M 同余时，就称 a, b 在以 M 为模时是相等的，记为

$$a = b \pmod{M}$$

例如

$$4 = -6 \pmod{10}$$

$$1 = 13 \pmod{12}$$

X 的补码定义为

$$[X]_{\text{补}} = 2^n + X \quad -2^{n-1} \leq X < 2^{n-1} \pmod{2^n}$$

由定义可知，补码的性质如下：

(1) 正数的补码和原码同形，即 $[X]_{\text{补}} = [X]_{\text{原}}$ 。

(2) 负数的补码符号位为“1”，其余位为原码各位取反，再在最低位加 1，即

$$[X]_{\text{补}} = [X]_{\text{反}} + 1$$

(3) 当 $X=0$ 时， $[+0]_{\text{补}} = [-0]_{\text{补}} = 000 \cdots 0$ 。

例如：若 $n=8, X_1 = 67$ ，则有 $[X_1]_{\text{补}} = 01000011$ ；

$X_2 = -67$ ，则有 $[X_2]_{\text{补}} = 10111101$ 。

补码再次求补以后就可求得其真值。对于正数当然没有必要这样作,对负数的补码经过再次求补并加上负号后,就为其真值。例如,对于 $[X]_{\text{补}} = 10110$, 则

$$X = -[[X]_{\text{补}}]_{\text{补}} = -01010$$

对于 8 位二进制数来说,用补码所表示的数的范围为 $-128 \sim +127$ 。

由于字长为 n 位的机器只能表示 n 位二进制数,而 2^n 是一个 $n+1$ 位数,因此,数 2^n 在机器中仅能以 n 个 0 来表示,而该数最左边的数字就自动丢失了。由此可知,这时 2^n 和 0 在机器中的表示形式是完全一样的,即在字长为 n 的机器中, 2^n 为其模。

这样(设 $A \geq 0, B \geq 0$):

$$A - B = A + (0 - B) = A + (2^n - B) = A + [(2^n - 1) + (-B)] + 1$$

由反码的定义可知

$$[-B]_{\text{反}} = [(2^n - 1) + (-B)]$$

由补码的定义可知

$$[-B]_{\text{补}} = [-B]_{\text{反}} + 1$$

因此上式

$$A - B = A + [-B]_{\text{反}} + 1 = A + [-B]_{\text{补}}$$

也就是说,利用补码表示法可把减法运算转化为加法运算,可使计算机实现减法运算非常方便。因此,在微型计算机中带符号数一般都采用补码表示法。

二、数的运算方法

(一) 补码运算

在微型计算机中,带符号数一般都以补码的形式在机器中存放和进行运算。这是因为补码的加减运算比原码简单,它是符号位与数值部分一起参加运算,并且能自动获得结果(包括符号和数值都在内)。

一般而言,两个数的补码运算可按以下步骤进行:

- (1) 把参与运算的两数连同其前面的正负号,变为补码;
- (2) 进行补码加法,得到两数运算结果的补码,若最高位上有进位则舍弃不要;
- (3) 若要求运算结果的真值,则按补码数求真值的办法进行。

【例 1-1】用补码运算求 $64 - 10$, 设 $n = 8$ 。

解:1) $[64]_{\text{补}} = 01000000$

$$[-10]_{\text{补}} = 11110110$$

2) 做加法: 01000000

$$\begin{array}{r} +) \quad 11110110 \\ \hline 00110110 \end{array}$$

3) 求真值。由于是正数,可直接求:

$$(00110110)_2 = 54$$

【例 1-2】用补码运算求 $64 - 65$, 设 $n = 8$ 。

解:1) $[64]_{\text{补}} = 01000000$

$$[-65]_{\text{补}} = 10111111$$

2) 做加法:

$$\begin{array}{r} 01000000 \\ +) 10111111 \\ \hline 11111111 \end{array}$$

3) 求真值。由于是负数,要对结果求反加 1 后再加上负号:

$$(11111111)_{\text{补}} = -(00000001) = -1$$

若参加运算的两个补码数运算的结果,超出了机器所允许表示的范围,将使符号位出现错误,如两个正数相加符号位为 1,两个负数相加符号位为 0,即得出了不正确的结果,这种情况称为溢出。

【例 1-3】用补码运算求 $64 + 65$, 设 $n = 8$ 。

解: 1)

$$[64]_{\text{补}} = 01000000$$

$$[65]_{\text{补}} = 01000001$$

2) 做加法:

$$\begin{array}{r} 01000000 \\ +) 01000001 \\ \hline 10000001 \end{array}$$

此时两个正数相加,其结果的符号位为 1,表明符号位出现了错误,即产生了溢出。也就是说, $64 + 65$ 不可能通过单字节表示的补码数的运算来得出正确结果。为了得到正确结果,可采用双字节表示(16 位)的补码数来运算:

$$[64]_{\text{补}} = 0000000001000000$$

$$[65]_{\text{补}} = 0000000001000001$$

$$[64 + 65]_{\text{补}} = 0000000001000001$$

此时,和的 16 位补码数的最高位仍为 0,因此没有出现溢出,结果正确。

(二) 原码的乘、除运算

实现原码的乘、除运算时,要分别确定积(或商)的符号及数值部分。

两个用原码表示的数相乘(或相除)时,乘积(或商)的符号位按同号乘(或除)结果为正,异号乘(或除)结果为负。这正好可以用两数的符号位通过异或运算来取得。

积(或商)的数值部分可按二进制数的乘、除法则来获得。由于符号位和数值部分是分别处理,因此,在求积(或商)的数值部分时,可把两个数都当作两个无符号数的乘(或除)来计算。

1. 无符号数相乘运算

例如有两个无符号数 1001 和 0101 相乘,按一般习惯,两者相乘可按以下方式进行:

$$\begin{array}{r} 1001 \\ \times) 0101 \\ \hline 1001 \\ 0000 \\ 1001 \\ 0000 \\ \hline 0101101 \end{array}$$

但是,在计算机中,一般不用这种方法来求积,因为它要一次求出 n 组二进制数的和,这对硬件的要求太高,即相当于要求有一个 n 位并行加法器,一般微型计算机中不配备这样的硬件。在计算机中实现乘、除运算有许多算法,这里介绍一种部分积右移的乘法算法。

设两数相乘: $a \times b = (a_n a_{n-1} \cdots a_1) \times (b_n b_{n-1} \cdots b_1)$, 部分积右移的乘法算法可描述如下:

- (1) 设部分积 $P = 0, i = 1$;
- (2) 若 $b_i = 1$, 则令 $P = P + a$, 否则不加;
- (3) 部分积 P 右移一位;
- (4) $i = i + 1$, 若 $i \leq n$, 则回到(2), 否则顺序执行;
- (5) 结束, P 即为所求的积。

上面的乘法示例按这种算法可实现如下:

	0000		部分积初值 $P = 0$
+) 1001			$b_0 = 1, P = P + a$
	1001		
	0100	1	P 右移一位
	0010	01	$b_1 = 0, P$ 只右移一位
+) 1001			$b_2 = 1, P = P + a$
	1011	01	
	0101	101	P 右移一位
	0010	1101	$b_3 = 0, P$ 只右移一位

结果亦为 00101101。在这里, 两个 n 位的无符号数的积一定是 $2n$ 位。

2. 无符号数相除运算

无符号数相除的过程也可以按一般十进制数相除的过程来进行。同样亦是為了适应计算机的操作, 提出了多种计算机除法算法。在除法运算过程中, 须存储和处理 4 个数据: 被除数、除数、商及余数。为了节省存储单元和便于操作, 可将被除数和商合用一个存储单元, 即在运算的开始阶段, 先存入被除数, 随着除法的进行, 被除数的高位逐渐失去作用, 正好可以空出位置来存放所得的商。设除数、被除数、商及余数都为 n 位。除法算法如下:

- (1) 设余数为 0, $i = 1$ 。
- (2) 被除数和余数分别左移一位, 并使被除数的最高位移入余数的最低位。
- (3) 求(余数 - 除数)之差。
- (4) 若差为正, 则令差代替余数, 并让商为 1, 亦即使被除数加 1; 若差为负, 则不作任何操作。
- (5) $i = i + 1$, 若 $i \leq n$, 则回到(2), 否则顺序执行。
- (6) 结束。余数寄存器中为余数, 被除数寄存器中为商。

【例 1-4】设被除数为无符号数 1101, 除数为无符号数 0010, 求商及余数。

解: