计算机大专教材系列

16位微型计算机原理与接口

李 信 编著

南开大学出版社

16 位微型计算机原理与接口

李 信 编著

南开大学出版社

内 容 简 介

本书以 INTEL 8086 微处理器为核心，全面介绍了 16 位微型计算机的工作原理、系统组成及开发应用。

首先从一个模型机入手，概述了微型计算机的基本结构和工作流程；然后分别介绍了 8086CPU 的结构、性能、工作原理及指令系统。

为了使读者更好地对微机进行开发和应用，本书详细讨论了微机与外设的接口、数据通信以及模拟通道的实用技术。

本书可作为理工科大专学生学习微机原理与接口技术的教科书，也可供从事计算机工作的工程技术人员参考。

16 位微型计算机原理与接口

李 信 编著

南开大学出版社出版

（天津八里台南开大学校内）

邮编 300071 电话 3508542

新华书店天津发行所发行

天津宝坻二丁印刷厂印刷

1995 年 12 月第 1 版　　1995 年 12 月第 1 次印刷

开本：787×1092　1/16　　印张：14.875

字数：371 千　　印数：1-8000

ISBN 7-310-00891-X

TP·49　定价：15.30 元

“计算机大专教材系列”编委会

出版说明

随着计算机应用的日益深入、普及，目前在我国正在兴起学习计算机专业知识的高潮，各种有关计算机的书籍如雨后春笋般涌现出来，使广大读者大有应接不暇之势。但是，已经出版的这些书籍中，有的偏深偏专，取材偏多偏全，适合有一定基础的计算机专业人员阅读参考；有的则是普及性读物，只适合急于入门的计算机爱好者使用；在为数不多的教材中，大都是为计算机专业本科生使用而编写的，不适合成人教育和大专类学生的需要。鉴于这种形势，我们决定编写一套适合于计算机类各专业大专学生和成人教育使用的教材。这套教材共有十种，虽然它还不能完全覆盖上述办学层次教学计划中的所有课程，但是它包括了培养一个计算机类专科生的主要教学内容。其中入门的教材有《计算机应用基础》和《C 语言程序设计》；属于专业基础的教材有《16 位微型计算机原理与接口》，《汇编语言程序设计》，《数据结构》和《操作系统》；应用性较强的《单片机及其应用》，《数据库系统教程》，《计算机网络基础》和《软件工程引论》。

这套教材贯彻了理论联系实际、学以致用的原则。在取材方面，不追求包罗万象、面面俱到，而着力保证把最基本、最实用的部分包含进来。在叙述方面，力求做到深入浅出，尽量用实例来说明基本概念和基本方法。我们希望这套教材不仅能适合课堂讲授的需要，也便于广大读者自学。这套教材由南开大学计算机与系统科学系的教师们编写而成，他们之中既有教学经验十分丰富的教授、副教授，也有活跃在计算机应用最前沿的青年教师。这些教师不仅具有教本科生、研究生的教学经验，也具有教大专生和成人教育的教学经验，这就使这套教材的质量有了基本的保证。但是由于我们初次编写这类教材，尚未经过实践的检验，缺点和不足之处在所难免，敬希同行专家和广大使用者批评指正。

目　录

第 1 章　计算机原理概述

第 2 章　8086 CPU 的结构

第 3 章　8086 指令系统

第 4 章　存储器

第 5 章　中断系统

第 6 章 数据的并行传送

第 7 章 可编程并行接口芯片

第 8 章　串行数据通信

第 9 章　模拟通道接口

第 章

计算机原理概述

随着科学技术的高度发展，导致了计算机的诞生。计算机的普及和应用，又对人类社会的发展产生了极其深刻的影响。计算机已成为人类和自然的斗争以及从事各项社会活动的一种强有力的工具。

随着大规模集成电路技术的飞速发展，计算机的质量逐年提高，而成本却逐年下降。这就为计算机在各个方面的应用提供了物质保证。计算机不再是少数专家独享的神秘宠物，它已经成为人们参于政治、经济、科学活动乃至家庭娱乐的得力助手。

计算机具有计算、模拟、分析问题、操纵其它机器、处理各种事物的能力，所以被看作是人脑的延伸；而组成计算机的硬设备大部分都为以大规模集成电路为核心的电子器件，因此计算机被人们称为"电脑"。

微型计算机最核心的部件是一个称为 CPU(Central Processing Unit)的中央处理单元，它一般由 1 片大规模集成电路构成。这个 CPU 的特性就代表了计算机的基本性能。有时它还成为机型的代名词，如一般常说的这台微机是 386 或 486，就指它的 CPU 是 80386 或 80486。

CPU 里面一般包含运算器和控制器，随着半导体技术的发展，CPU 内部还可以配上一定容量的存储器，配上可以和外部设备相连的可编程的接口电路。

与 CPU 同步发展的，是存储器芯片制造技术。已经可以生产容量相当大的存储器芯片。如 128K×8 位的读写静态存储器 RAM(Random Access Memory)以及 4M×4 位的动态 RAM，以及 128K×8 位的只读存储器 ROM(Read Only Memory)。

这样，由 CPU 配置一定容量的 RAM、ROM，以及接口电路和必要的外部设备(如，CRT 终端，键盘，打印机，磁盘驱动器等)就构成了一台微型计算机。

本书以一个典型的 CPU8086 作为研究学习对象。它是一个标准的 16 位的 CPU。它与 32 位 CPU 的代表 80386 是一脉相承的，也是学习高档微机 386 和 486 的无法逾越的台阶。

本书着重分析 8086CPU 的结构与时序，存储器组织，并行、串行数据通信以及模拟通道接口等。

1.1 计算机中数据的表示

1.1.1 计算机中的数制

计算机是以电子器件为核心，以电子器件的状态表示数的。电子器件以两种不同的状态最为稳定可靠，它输出或高电平或低电平，用这个高、低电平表示 1 位二进制数。因此在计算中，数全部是用二进制表示的。

1. 二进制数

一个二进制数具有两个基本特征：

(1)具有两个不同的数字符号，即 0 和 1；

(2)逢二进位。

二进制数由排列起来的 0 和 1 组成，各位代表的数值不同，从位序号为 0 向左数，依次代表的数值为 1、2、4、8、16……等等。例如：

D_3	D_2	D_1	D_0
1	1	0	1

D_0 位的 1 代表 1，D_1 位的 1 代表 2，D_2 位的 1 代表 4，D_3 位的 1 代表 8。每一位有一个基值与之相对应，这个基值称为位权。整数部分位权为 2 的正次幂，若为小数，则小数点后面的数位的权为 2 的负次幂。例如

D_{-1}	D_{-2}	D_{-3}	D_{-4}
0.1	1	0	1

D_{-1}位代表 $2^{-1}=0.5$，D_{-2}位代表 $2^{-2}=0.25$，D_{-4}位代表 $2^{-4}=0.0625$。一个二进制数所表示的实际值可用如下的公式计算：

$$\sum_{i=-m}^{n-1} D_i \times 2^i$$

例如：计算 1101.1101 的实际值

$$(1101.1101)_2 = 1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}+0\times2^{-3}+1\times2^{-4}$$
$$= (13.8125)_{10}$$

2. 十六进制数

在计算机中，最常用十六进制数。一个十六进制数的基本特点是：

(1)具有 16 个数字符号，采用 0－9 和 A－F；

(2)逢 16 进位。

十六进制数是由排列起来的 0－F 组成，每一个数位有一个权与之对应，小数点左边各数位的权为 16 的正次幂，小数点右边各数据位的权是 16 的负次幂。

一个十六进制数表示的实际值可用下列公式计算：

$$\sum_{i=-m}^{n-1} D_i \times 16^i$$

例如：$(FF0E)_{16}=15\times16^3+15\times16^2+0\times16^1+14\times16^0=(65294)_{10}$

$$(A8.6C)_{16}=10\times16^1+8\times16^0+6\times16^{-1}+12\times16^{-2}$$
$$=(168\frac{27}{64})_{10}$$

但是，在机器中，数仍以二进制表示，这是由物理器件本身决定的。之所以用十六进制数表示，是因为 4 位二进制数可用 1 位十六进制数表示，这样人们看起来更方便、易懂。表 1－1 给出了二进制数与十六进制数、十进制数的对应关系。

为了区分数的进制，常用 B 表示二进制数，用 H 表示十六进制数，D 表示十进制数。

例如：$(1011)_2$ 可表示为 1011B

$(1011)_{16}$可表示为 1011H

$(1011)_{10}$可表示为 1011D

表 1—1　二进制、十六进制、十进制对照表

二进制数	十六进制数	十进制数
0 0 0 0	0	0
0 0 0 1	1	1
0 0 1 0	2	2
0 0 1 1	3	3
0 1 0 0	4	4
0 1 0 1	5	5
0 1 1 0	6	6
0 1 1 1	7	7
1 0 0 0	8	8
1 0 0 1	9	9
1 0 1 0	A	10
1 0 1 1	B	11
1 1 0 0	C	12
1 1 0 1	D	13
1 1 1 0	E	14
1 1 1 1	F	15

3. 数制的转换

(1) 二进制与十六进制的转换

将二进制数转成十六进制相当方便。整数部分从小数点向左，每 4 位一份，组成 1 位十六进制数，不足 4 位的前面补 0；小数部分由小数点向右，每 4 位一份，不足 4 位的后面补 0，每 4 位用相应的十六进制数代替，即转换成十六进制数。例如：

$(1\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0.1\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1)_2$ 转换为

0011　0101　1110　.　1101　0101　1100

3　5　E　.　D　5　C

转换结果为：$(35E.D5C)_{16}$

若将十六进制数转换成二进制数，则只要把每 1 位十六进制数用相应的 4 位二进制数代替即可。　例如：

$(8BC.7E)_{16}$　转换为

$(1000\ 1011\ 1100.0111\ 1110)_2$

(2) 二进制与十进制转换

将二进制转换成十进制的方法是借用公式

$$\sum_{i=-m}^{n-1} D_i \times 2^i$$

即，将二进制数按权求和。这在前面已有实例，这里不再举例说明。

下面介绍将十进制转换成二进制的方法。对于整数部分和小数部分要分别对待，整数部分的转换方法是除 2 取余法，小数部分的转换方法是乘 2 取整法。

例如，将十进制数 206 转换成二进制数。

		余数	
2	206	…… 0	低位
2	103	…… 1	
2	51	…… 1	
2	25	…… 1	
2	12	…… 0	
2	6	…… 0	
2	3	…… 1	
2	1	…… 1	高位

转换结果为$(1100\ 1110)_2$

采取的方法是用2连续除以十进制数，直到商为0时结束，最先得到的余数为二进制数的最低位，最后得到的余数为二进制数的最高位。

小数部分的转换是用2去乘它，取乘积的整数部分为转换后的二进制小数的最高位，再用2去乘上一步乘积的小数部分，再取整数部分为二进制小数低一位的数字。重复乘2直到积为零或已达到二进制小数位数的要求，即转换过程完成。

例如，将(0.385)转换成二进制小数。

	整数	0.385	×2
高位	0.	77	×2
	1.	54	×2
	1.	08	×2
	0.	16	×2
	0.	32	×2
	0.	64	×2
低位	1.	28	

转换结果为$(0.0110001)_2$

最后的积仍不为零，但只要达到要求的位数也就可以了；因为大部分小数是永远也不会满足乘积为零的。

若是既有整数又有小数的十进制数，则先分别进行转换，再把结果合起来就得到最后结果。

例如，将十进制数206.385转成二进制数。前面已经做过：

$(206)_{10}=(11001110)_2$，$(0.385)_{10}=(0.0110001)_2$

那么，$(206.385)_{10}=(11001110.0110001)_2$

4. 二进制数的运算

(1)加法运算

二进制加法运算规则为：

0+0=0

0+1=1

1+0=1

1+1=0　(并向高一位产生进位，结果为10。)

例如：

$$\begin{array}{r} 1011 \\ +1110 \\ \hline 11001 \end{array}$$

(2)减法运算

二进制减法运算规则为：

0－0＝0

0－1＝1　(向高位借位)

1－0＝1

1－1＝0

例如：

$$\begin{array}{r} 1101 \\ -1011 \\ \hline 0010 \end{array}$$

(3)乘法运算

二进制乘法运算规则：

0×0＝0

0×1＝0

1×0＝0

1×1＝0

例如：

$$\begin{array}{r} 1101 \\ \times\ 1011 \\ \hline 1101 \\ 1101 \\ 0000 \\ +\ 1101 \\ \hline 10001111 \end{array}$$

(4)除法运算

二进制除法与十进制除法类似，由上商、减法逐步完成。

例如：

$$\begin{array}{r|l} & 1101 \\ \hline 1011 & 10001111 \\ & -1011 \\ \hline & 1101 \\ & -1011 \\ \hline & 1011 \\ & -1011 \\ \hline & 0 \end{array}$$

1.1.2　计算机中数据的表示方法

在计算机中能直接表示和使用的，有数值数据和符号数据两大类。数值数据用来表示数量

的大小，并且还带有表示数值正负的符号位。符号数据又称非数值数据，用于表示一些符号标记，包括英文大小写字，数字符号 0—9 等，汉字和图形信息也属于符号数据。由于计算机中任何数据都采用二进制编码形式，因此讨论数据的表示方法，就是讨论它们在计算机中的组成格式和编码规则。

1. 带符号数的表示方法

在计算机中，数值有大小，但也会有正、负，用什么方法表示数值的符号呢？通常用一个数的最高位定为符号位。若字长为 8 位，则 D_7 位为符号位，D_6—D_0 为数字位。符号位用 0 表示正，用 1 表示负。如

$X=(01011011)_2=+91$

$X=(11011011)_2=-91$

这样连同一个符号位在一起作为一个数，就称为机器数，而它的数值称为机器数的真值。

为了运算方便，在机器中带符号数有三种表示方法——原码、反码和补码。

(1)原码

按上所述，正数的符号位用 0 表示，负数的符号位用 1 表示，这种表示方法称为原码。

例如，$X=+100$　　$[X]_{原}=01100100$

$Y=-100$　　$[Y]_{原}=11100100$

其中最高位为符号位，后面 7 位为数值，在原码表示中，+100 和 −100 数值位相同，符号位不同。

原码表示简单易懂，但若两个异号数相加就要做减法，为了把减法运算转换为加法运算，就又引进了反码和补码。

(2)反码

正数的反码表示与原码相同，最高位为符号位，用 0 表示正，其余位为数值位。

例如，$[+66]_{反}=0\ 1000010$

$[+6]_{反}=0\ 0000110$

负数的反码表示为它的正数“按位取反”(连同符号位)。

例如，$[-6]_{反}=1\ 1111001$

$[+127]_{反}=0\ 1111111$

$[-127]_{反}=1\ 0000000$

$[+0]_{反}=0\ 0000000$

$[-0]_{反}=1\ 1111111$

负数的反码表示与原码有很大区别。最高位仍为符号位，负数仍用 1 表示，但数值位不同。

以 8 位二进制反码表示的数有以下特点：

① 0 有两种表示方法；

② 能表示的数值范围为 +127～−127；

③ 一个带符号数由反码表示时，其最高位 D_7 为符号位，0 表示正数，1 表示负数，后 7 位数为数值；对于负数，一定把它“按位取反”才得到它的二进制值。

例如，$[10101100]_{反}$

它的最高位为 1，所以为负数，其值的大小，将数值位“按位取反”；这个用反码表示的数为：

$-1\ 0\ 1\ 0\ 0\ 1\ 1=(-83)_{10}$

(3)补码

正数的补码表示与原码相同，而负数的补码表示为它的正数“按位取反”(包括符号位)，并且在最低位加1而形成。

例如，$[+6]_{补}$ $=0\ 0000110$

$[-6]_{补}$ $=1\ 1111001+1$

$=1\ 1111010$

$[+127]_{补}=01111111$

$[-127]_{补}=10000001$

$[+0]_{补}$ $=00000000$

$[-0]_{补}$ $=00000000$

8位带符号数的补码表示，有如下特点：

① $[+0]_{补}=[-0]_{补}=00000000$；

② 8位二进制补码所能表示的数值范围为+127～-128；

③ 一个用补码表示的二进制数，其最高位为符号位；当符号位为0时，表示为正数，其余7位为此数的二进制值；但当符号位为1时表示为负数，其余几位不是此数的二进制值，应把它“按位取反”，且在最低位加1，才是它的二进制值。

例如：$[10101100]_{补}$

其最高位为1，说明是负数；若求其值，则按位取反为01010011再加1为01010100。

其值为 $-01010100=-(84)_{10}$。

表1—2给出各种码制下的数的表示。

表1—2 数的表示法

二进制数码	无符号数	原码	补码	反码
00000000	0	+0	+0	+0
00000001	1	+1	+1	+1
⋮	⋮	⋮	⋮	⋮
01111110	+126	+126	+126	+126
01111111	+127	+127	+127	+127
10000000	+128	−0	−128	−127
10000001	+129	−1	−127	−126
⋮	⋮	⋮	⋮	⋮
11111110	+254	−126	−2	−1
11111111	+255	−127	−1	−0

当负数采用补码表示时，可以把减法转换成加法。

例如：$X=64-14=64+(-14)$

$[X]_{补}=[64]_{补}+[-14]_{补}$

$[64]_{补}=01000000$

$[-14]_{补}=11110010$

$[X]_{补}$为将二者相加

$$\begin{array}{r} 0100\ 0000 \\ +\ 1111\ 0010 \\ \hline \boxed{1}\ 0011\ 0010 \end{array}$$

↓

自然丢失

由于字长只有 8 位，故当有向更高位进位时会自然丢失。

$[X]_{补}=00110010=(50)_{10}$

上述实例说明了引入补码概念之后，数的减法运算可以用加法运算代替。这是很有实用价值的。在计算机中，凡带符号的数一律用补码表示。无论是参与运算的数还是运算结果，都是采用补码形式。

为什么本来是做的减法却可以用加法代替呢？从运算式子中可以看到，若没有自然丢失的进位则决不会得到正确的结论。

设想一个环形跑道上，均匀地放好 256 把椅子，椅子编号从 0 开始到 255 号。现在有人处于 64 号椅子上，如果要到 50 号椅子上去，他可以向回走过 14 把椅子。也可以向前走，经过 65 号直到 255 号，接着又从 0 号椅子开始走一直会走到 50 号椅子的。这样他走过多少椅子呢？显然是 256－14＝242。可以看出，－14 和＋242 效果相同。因此，256 称为模，它是这个系统里所能表示的最大的数，而(－14)和(242)则互为补数。(－14)的补码可以从 2^8-14 得到。

这个例子就相当于，对一个 8 位二进制数，当用补码表示时，它的模为 $2^8=256$；而当从某数中减去一个小于模的数时，总可以用加上该数的负数与其模数之和来代替。

也就是：$64-14=64+(256-14)$

$=64+242$

$=50+256$

$=50$ ←自然丢失

由于 8 位的字长表示带符号数的范围为：

＋127～－128

若运算结果超出这个范围，则结果就不正确，一般称之为溢出。这时可扩大字长，如用 16 位字长，则它表示的数的范围为：

＋32767～－32768

2. 十进制数的表示方法

计算机内部是以二进制表示数值的，而人们习惯使用十进制数。目前功能较强的计算机都能直接处理十进制表示的数。如后几章讲的 8086CPU 就有直接处理十进制数的指令。

(1)二进制编码的十进制数

1 位十进制数可以用 4 位二进制数表示，可以有很多方法，较常用的是 8421 码。其编码表如表 1—3。

8421 码有 10 个不同的数字符号，且逢十进位，所以为十进制数；但每一位十进制数是由 4 位二进制数表示的，因此称为二进制编码的十进制数，也称为 BCD(Binary Coded Decimal)码。

①压缩的 BCD 码

压缩的 BCD 码是用一个字节即 8 位二进制数表示两位十进制数。高 4 位可以表示十进制数的 10 位数，低 4 位可以表示十进制数的个位数。

例如：$[00100110]_{BCD}$

高 4 位代表十进制数的 2，低 4 位代表十进制数的 6，所以其值应为$(26)_{10}$。