普通高等教育“十一五”国家级规划教材

电子信息类精品教材·优秀畅销书

微机原理与接口技术

Microcomputer Principle and Interface Technology

（第4版）

郑初华　等编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY　http://www.phei.com.cn

普通高等教育“十一五”国家级规划教材
电子信息类精品教材

微机原理与接口技术

（第4版）

郑初华　柴明钢　周卫民　袁　坤
石永革　赵文龙　戴仕明　郭　亮　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书由汇编语言、微机原理、接口技术及附录四个部分组成，主要内容有：快速进制转换，真值与补码直接转换，微机硬件基础，8086/88CPU 指令系统以及内部结构、引脚、时序，汇编语言及编程方法，内存的存储原理及与 CPU 的连接，I/O 方式及编程，中断概念及实现，8255、8253、8251 等接口芯片硬件设计及编程驱动，A/D、D/A 转换以及工业自动化控制，键盘及接口，显示及接口，并口通信，串口通信，总线技术，微机系统应用设计，附录等。

本书可作为高等学校有关专业汇编语言、微机原理、接口技术等课程的教材。

图书在版编目(CIP)数据

微机原理与接口技术/郑初华等编著．—4 版．—北京：电子工业出版社，2014.5
电子信息类精品教材
ISBN 978-7-121-23117-9

Ⅰ．① 微… Ⅱ．① 郑… Ⅲ．① 微型计算机－理论－高等学校－教材 ② 微型计算机－接口技术－高等学校－教材 Ⅳ．① TP36

中国版本图书馆 CIP 数据核字(2014)第 087028 号

责任编辑：韩同平　　特约编辑：李佩乾
印　　刷：北京丰源印刷厂
装　　订：三河市鹏成印业有限公司
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：787×1092　1/16　印张：22　字数：600 千字
印　　次：2014 年 5 月第 1 次印刷
印　　数：3 000 册　　定价：49.80 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

本书第1、2、3版分别于2003、2006、2010年出版。本书于2004年荣获江西省首届高校优秀教材一等奖(计算机类第一名),2008年列选普通高等教育"十一五"国家级规划教材。

本书由汇编语言、微机原理、接口技术及附录四个部分组成。本书融入多位老师的教学经验,重点突出,详略有序,分类讲解,图表丰富,有一些讲法是其他同类教材未曾涉及的,如快速进制转换、真值与补码直接转换、指令的6个要点等。本书适合作为高等学校有关专业汇编语言、微机原理、接口技术以及它们的组合课程的教材。

本书共16章,主要内容有:快速进制转换,真值与补码直接转换,微机硬件基础,8088/8086 CPU指令系统以及内部结构、引脚、时序,汇编语言及编程方法,内存的存储原理及与CPU的连接,I/O方式及编程,中断概念及实现,8255、8253、8251等接口芯片硬件设计及编程驱动,A/D、D/A转换以及工业自动化控制,键盘及接口,显示及接口,并口通信,串口通信,总线技术,微机系统应用设计,书后附有附录A～E。

本书由郑初华等编著。郑初华主要编写第1～6章、第10章,并负责全书的策划和统编定稿;柴明钢主要编写第14章、第16章;周卫民主要编写第12章;袁坤主要编写第8章、第9章;石永革主要编写第7章;赵文龙主要编写第15章;戴仕明主要编写第11章;郭亮主要编写第13章。本书由郑初华和赵文龙共同审定,由于时间紧,错误在所难免,欢迎广大读者批评指正(zhengchuhua@126.com)。

在此,对曾给本书的编写提出意见及参加校稿的秦梅、胡锦春、石永革、程从从、万光逵、衷裕水、向瑛、周琪、万在红、杨谊华、彭洁、崔丽珍、肖洁、刘洪、洪连环、冀春涛、邓黎鹏、吴国辉、温靖、龚廷恺、周波、万承兴、黄华、黎明、代冀阳、彭玉玲、宋凯、王青松、曹党生、田祖伟等同志一并表示感谢!

本书参考学时建议:汇编部分40～64课时,微机原理及接口技术60～80课时。汇编部分安排8个实验:2～3个DEBUG上机实验便于熟悉第3章的指令和调试过程,5～6个汇编语言完整程序上机实验便于熟悉程序框架及程序编写方法;接口部分安排4个实验,熟悉I/O方式及或编程接口芯片的连接与驱动。另外,部分章节上课顺序可根据需要适当调整。

编著者

目　录

第一部分　汇编语言

第二部分　微机原理

第三部分 接 口 技 术

第四部分 附 录

第一部分　汇编语言

第1章　进制及码元

进制和码元换算是计算机重要基础之一，计算机内采用的是二进制数值或编码，而在各种汇编语言中习惯使用十六进制，也可使用八进制、二进制和十进制，在C语言中可使用八进制、十六进制和十进制，特别是调试程序时更要与进制和码元换算打交道。所以掌握进制和码元换算的快速方法，对学好计算机相关课程特别是汇编语言、微机原理及接口技术非常重要。

本章所介绍的进制转换方法可以完成十进制、二进制、十六进制及八进制数之间的快速转换，一般可以在10s内完成万以内的数值转换。此外，本章所介绍的真值（有符号数）与补码（或无符号数）之间的直接转换也是前人未曾涉及的，负数与补码（或无符号数）之间转换也只要10s左右。

1.1　进制转换及计算

本节主要讲解进制的快速转换方法，学会此法可在10s内实现万以内的数值转换。

1. 进制

现实生活中除了最常用的十进制外，还有秒分时之间的六十进制、月年之间的十二进制以及古代钱两斤之间的十六进制等，在计算机语言中主要采用的是二进制（后缀B，Binary）、八进制（后缀O或Q，Octal，O易与0混淆，所以一般用Q替代O）、十进制（后缀D，Decimal，或不要后缀）和十六进制（后缀H，Hex）。4种进制基本信息如表1.1所示。

表1.1　计算机语言中的基本进制

进　制	英　文	尾　缀	数据位取值	举　例	算术运算
二	Binary	B	0、1	10110101B	逢二进一，借一等于二
八	Octal	O或Q	0～7	123O或357Q	逢八进一，借一等于八
十	Decimal	D或省略	0～9	68D或259	逢十进一，借一等于十
十六	Hex	H	0～9、A～F	2FCH	逢十六进一，借一等于十六

N进制的每个数据位取值范围为0～$N-1$，其算术运算规则同十进制，只不过是逢N进一、借一等于N而已。例如，二进制只有0和1两个数字，逢2进1，借1等于2；十六进制有0～9、A～F（分别代表10～15）16个数字，逢16进1，借1等于16。

2. 进制转换的一般方法

进制转换的一般方法如图1.1和图1.2所示。

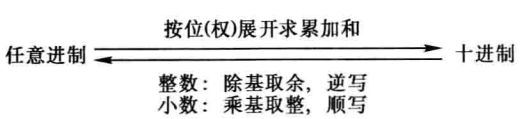

图1.1　任意进制数与十进制数之间转换关系图

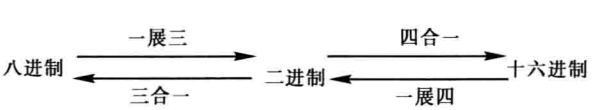

图1.2　二进制、八进制、十六进制之间转换关系图

例 1.1 $(101101)_2=101101\mathrm{B}=1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0=45$

例 1.2 $156.4\mathrm{Q}=1\times8^2+5\times8^1+6\times8^0+4\times8^{-1}=110.5$

例 1.3 $6\mathrm{C}.4\mathrm{H}=6\times16^1+12\times16^0+4\times16^{-1}=108.25$

下式中 a_i 代表 b 进制的第 i 位，任意的 b 进制转化为十进制的一般式子：

$$(a_na_{n-1}\cdots a_1a_0,a_{-1}\cdots a_{-m})_b=a_n\times b^n+a_{n-1}\times b^{n-1}+\cdots+a_0\times b^0+a_{-1}\times b^{-1}+\cdots+a_{-m}\times b^{-m}$$

$$=\sum_{i=-m}^{n}a_i\times b^i$$

例 1.4 $123.25=(1\ 111\ 011.01)_2=(173.2)_8=(7\mathrm{B}.4)_{16}$

解题步骤如图 1.3 所示。

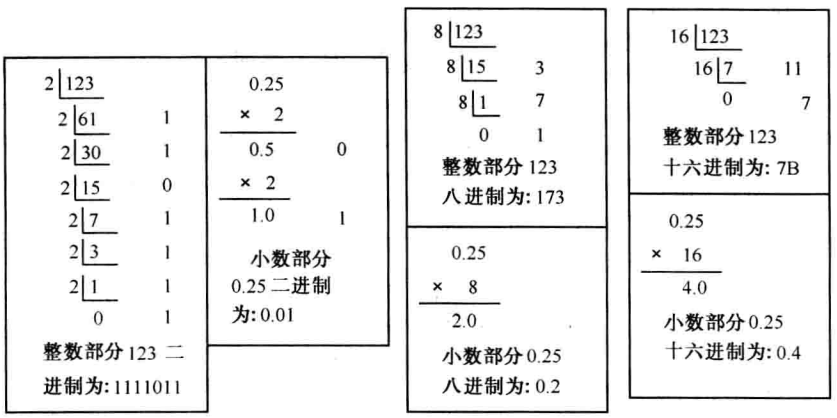

图 1.3　十进制转换为其他进制的一般方法

3. 进制快速转换方法

掌握进制快速转换方法的前提是记住 16 的倍数或 2 的 n 次方，如表 1.2 所示。

表 1.2　2 的指数及 16 的倍数表

n 的值	2^n	n 的值	$16\times n$	十六进制
−4	0.0625	1	16	10H
−3	0.125	2	32	20H
−2	0.25	3	48	30H
−1	0.5	4	64	40H
0	1	5	80	50H
1	2	6	96	60H
2	4	7	112	70H
3	8	8	128	80H
4	16	9	144	90H
5	32	10	160	A0H
6	64	11	176	B0H
7	128	12	192	C0H
8	256	13	208	D0H
9	512	14	224	E0H
10	1K(1 024)	15	240	F0H

（续表）

n 的值	2^n	n 的值	$16\times n$	十六进制
14	16K	1×16	256	100H
16	64K	2×16	512	200H
20	1M(1 024K)	3×16	768	300H
24	16M	4×16	1024(1K)	400H
30	1G(1 024M)	8×16	2 048(2K)	800H
40	1T(1 024G)	1×16×16	4 096(4K)	1 000H

记住表 1.2 的主要数据后，再结合图 1.4 及图 1.2 就可以在 10s 内完成进制转换。

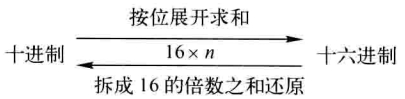

图 1.4　十进制与十六进制的快速转换思路

具体方法为：

将十进制转换为十六进制，只要把它拆成 16 的倍数之和(注：有时视情况可用 16 的倍数之差)还原成十六进制即可，再利用图 1.2 一展四转换为二进制，而后再用三合一转换为八进制。

例 1.5　280=256+16+8=118H=100 011 000B=430Q

例 1.6　2 000=2 048−48=800H−30H=7D0H=11 111 010 000B=3 720Q

例 1.7　5 000=4 096+768+128+8=1 388H=1 001 110 001 000B=11 610Q

也可先将十进制转换为二进制，只要把它拆成 2 的 n 次方之和(注：有时视情况可用 2 的 n 次方之差)，有 n 次方的二进制位写成 1，无 n 次方的二进制位写成 0 即可，再利用图 1.2 四合一转换为十六进制及用三合一转换为八进制。

例 1.8　$280=2^8+2^4+2^3=100011000B=118H=430Q$

例 1.9　$2000=2^{10}+2^9+2^8+2^7+2^6+2^4=11111010000B=7D0H=3720Q=2^{11}-2^5-2^4$

例 1.10　$5000=2^{12}+2^9+2^8+2^7+2^3=1001110001000B=1388H=11610Q$

4. 进制计算

进制计算主要有加、减、乘、除等算术运算及与、或、非等逻辑运算。其他进制加、减、乘、除等算术运算的运算方法与十进制的运算方法类似，要点是逢 N 进一、借一等于 N。与、或、非等逻辑运算一般是指变量取值为二值(0 或 1)的逻辑运算，将 1 当成真，将 0 当成假，与、或、非的真值表如图 1.5 所示。

A 与 B

B \ A	0	1
0	0	0
1	0	1

A 或 B

B \ A	0	1
0	0	1
1	1	1

非 A

A	0	1
$\overline{A}$	1	0

图 1.5　三种位逻辑运算真值表

在本书 3.3 节的汇编指令部分和 4.2 节的表达式部分将给出具体举例。

1.2　码制及其转换

本节介绍计算机主要使用的二进制编码，重点讲解真值(有符号数)与补码(或无符号数)间的快速转换方法。此方法使得 8 位或 16 位二进制补码的求解及有无符号数之间的转换变得轻而易举。

1. BCD 码

常见的 BCD 码有 8421 码、2421 码以及余 3 码等，一般使用 8421 码，它又分为压缩 BCD 码和非压缩 BCD 码。压缩 BCD 码是用 4 位二进制代码表示一位十进制，一个字节可以表示两位十进制(00～99)；而非压缩 BCD 码是用 8 位二进制代码中的低 4 位表示一位十进制、高 4 位无效，一个字节只能表示一位十进制(0～9)，高 4 位为 0 时则叫标准非压缩 BCD 码。例如，十进制数 35 的压缩 BCD 码为 35H，其标准非压缩 BCD 码为 0305H。它们的比较示意图如图 1.6 所示。

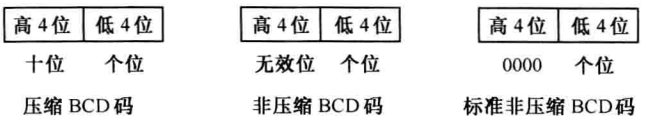

图 1.6　三种 8421 BCD 码的比较

2. ASCII 码

ASCII 码使用 8 位二进制编码，占一个字节，最高位为 0 的 ASCII 码称为基本 ASCII 码。重要的 8 个字符的 ASCII 码值如表 1.3 所示，其他字符参看附录 E。

'0'～'9' 的 ASCII 码依次加 1，'A'～'Z' 的 ASCII 码依次加 1，'a'～'z' 的 ASCII 码也是依次加 1，所以记住'0'、'A' 以及'a' 的 ASCII 码，也就记住了 62 个字符的 ASCII 码。'0'～'9' 的 ASCII 码是一种特殊的非压缩 BCD 码。例如 '35' 是十进制数 35 的非压缩 BCD 码即 3335H。

表 1.3　重要的 ASCII 字符

字　符	ASCII 码 十进制值	ASCII 码 十六进制值
NUL(空)	0	00H
LF(换行)	10	0AH
CR(回车)	13	0DH
SP(空格)	32	20H
'$'	36	24H
'0'～'9'	48～57	30H～39H
'A'～'Z'	65～90	41H～5AH
'a'～'z'	97～122	61H～7AH

3. 汉字内码

汉字在计算机及相关设备内存储、处理以及传输所用的编码称为汉字内码。我国目前主要采用的是国标内码(GB2312)，它在计算机内占用两个字节，每个字节的最高位为 1，最多可表示 $2^{14}=16\ 384$ 个可区别代码。它与国标区位码的计算关系为：国标内码＝国标码(十六进制)＋8080H＝国标区位码(十六进制)＋A0A0H。GB2312—80 中有：一级汉字 3 755 个、按拼音顺序排列，二级汉字 3 008 个、按偏旁笔画数排列，字符 682 个。中国香港地区、中国台湾地区以及新加坡等繁体汉字区主要采用大五码(BIG5)，它在计算机内也是占用两个字节，每个字节的最高位也为 1。

为了统一表示世界上各国的文字，1993 年国际标准化组织公布了“通用多八位编码字符集”的国际标准 ISO/IEC10646，简称 UCS(Universal Code Set)，其中汉字部分叫 CJK(中、日、韩)统一汉字集。UCS 用 4 字节足以表示世界上所有的文字，包括英文、中文、日文、韩文、俄文以及法文等。我国的相应标准为 GB13000。

4. 原码、反码和补码

原码、反码和补码均为有符号数的编码，正、负号也用二进制编码来表示，它们所代表的实际数值称为“真值或原值”。

原码是直接在真值的绝对值之前增加一个符号位，并取正数的符号为 0，负数的符号为 1。正数的反码、补码与原码相同，负数的反码为原码的符号位不变其他位变反而得，负数的补码为原码的符号位不变其他位变反＋1 而得。负数的三种编码之间的转换关系如图 1.7 所示。

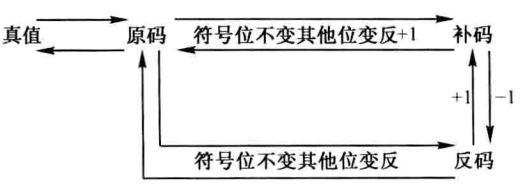

图 1.7　负数的原、反、补码之间转换关系图

补码是计算机中最基本的有符号数编码方案，最主要原因是因为采用补码后：减法可变加法如5－3＝5＋(－3)；加减时符号位如同数值位一样参加计算，具体例子请参看 3.3.2 节。

例 1.11 （8 位二进制数的原、反和补码）

－107 ＝－6BH＝－1101011B＝11101011B(原)＝10010100B(反)＝10010101B(补)

＝EBH(原)＝94H(反)＝95H(补)

107＝6BH(原)＝6BH(反)＝6BH(补)

5. 二进制数据的表示范围

二进制数据的表示范围要分有符号数还是无符号数。无符号数的所有二进制位(bit)均作为数值位；有符号数的最高位代表符号位，1 代表负、0 代表正，其余位才是数值位。n 位二进制无符号数的表示范围为 0～(2^n-1)。n 位二进制有符号数的表示范围还取决于编码方案，补码为 -2^{n-1}～$+(2^{n-1}-1)$；原码、反码的表示范围为$-(2^{n-1}-1)$～$+(2^{n-1}-1)$。计算机中内外存容量以字节(B,Byte)为单位，一个字节由 8 个二进制位构成(即 1 B＝8 b)。8 位二进制数(1 字节)的无符号数表示范围为 0～255，有符号补码表示范围为－128～＋127；16 位二进制(2 字节)的无符号数表示范围为 0～65 535，有符号补码表示范围为－32 768～＋32 767。

6. 真值与补码(无符号数)之间的直接转换

正数的真值与补码(无符号数)完全相同，负数的真值与补码(无符号数)之间的直接转换方法如图 1.8 所示(0 在用 n 位二进制补码表示时也代表 2^n，即 $0=2^n$)。

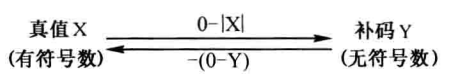

图 1.8　负数的真值与补码之间转换关系图

例 1.12　8位二进制时：

20＝14H(补)＝20(无)

－5＝0－5＝00H－05H＝FBH(补)＝251(无)＝256－5＝2^8－5

－120＝0－120＝00H－78H＝88H(补)＝136(无)＝256－120＝2^8－120

F8H(补)＝248(无)＝－(00H－F8H)＝－08H＝－(256－248)＝－8(有)

5CH(补)＝92(无)＝92(有)

16 位二进制时：

20＝0014H(补)＝20(无)

－5＝0－5＝0 000H－0 005H＝FFFBH(补)＝65 531(无)＝65536－5＝2^{16}－5

－120＝0－120＝0 000H－78H＝FF88H(补)＝65 416(无)＝65536－120＝2^{16}－120

FFC6H(补)＝0 000H－(0 000H－FFC6H)＝65 536－58＝65 478(无)

＝－(0－FFC6H)＝－3AH＝－58(有)＝－(65 536－65 478)＝－58(有)

048FH＝1 024＋128＋15＝1 167(无)＝1 167(有)

7. 定点数和浮点数

机器数的表示是受设备限制的。计算机一般是以字为单位进行数据的处理、存储和传递的。所以运算器中的加法器、累加器以及其他一些寄存器，都选择与字长相同的位数。字长一定，则计算机所能表示数的范围也就确定了。例如，使用 8 位字长的计算机，它可以表示无符号整数的表示范围为 0～255，补码的有符号数表示范围为－128～127。如果运算数值超出机器数所能表示的范围，机器就需要进行相应处理。这种现象称为溢出。

计算机中的数，既有整数，也有小数。如何确定小数点的位置呢？通常有两种约定：一种是规定小数点位置固定不变，这时的机器数称为定点数；另一种是小数点位置可以浮动，这样的机器数称为浮点数。

(1) 定点数

对于定点数，小数点位置可以固定在符号位之后，这样的机器表示的全是定点小数。例如，假定机器字长为 16 位，符号位占 1 位，数值占有 15 位，于是-2^{-15}用机器数原码表示如图 1.9 所示。

其相当于十进制数为-2^{-15}。

小数点位置固定在数的最后，则该机器表示的全是定点整数。例如，假设机器字长为 16 位，符号位占有 1 位，数值部分占 15 位，图 1.10 表示的机器数相当于十进制数为+32 767。

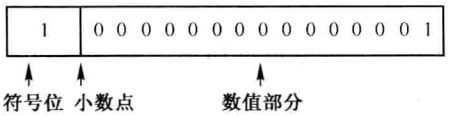

图 1.9　定点小数示意图

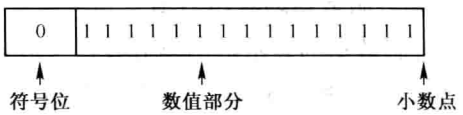

图 1.10　定点整数示意图

定点表示法表示的数值范围及精度有限，为了扩大定点数的表示范围或提高精度，可以采用多个字节来表示一个定点数，例如，采用 4 字节或 8 字节来表示。

(2) 浮点数

浮点数表示法就是小数点在数中的位置是浮动的。由于定点数表示的数的范围较窄，不能满足实际问题的需要，因此要采用浮点表示法。在同样字长情况下，浮点表示法能表示数的范围扩大了。

浮点表示法包括两部分：一部分是阶码，另一部分是尾数。浮点数在机器中的表示方法如图 1.11 所示。

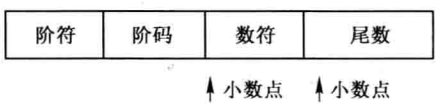

图 1.11　浮点数示意图

由尾数部分隐含的小数点位置可知，尾数的绝对值总是小于 1 的数，它给出该浮点数的有效数字，为了有更多位有效数字，一般用规范化小数表示，即尾数的绝对值大于等于 0.5、小于 1。尾数部分的数符确定该浮点数的正负。阶码总是整数，它确定小数点浮动的位数。若阶符为正，尾数的小数点向右移动；若阶符为负，则向左移动。即浮点数的值为：尾数$\times 2^{阶码}$。

当浮点数的尾数为零或者阶码为最小值时，机器通常规定，把该数看做 0，称为“机器零”。在浮点数的表示和运算中，当一个数的阶码大于机器所能表示的最大阶码时，产生“上溢”，当一个数的阶码小于机器所能表示的最小阶码时，产生“下溢”。

浮点数的取值范围如图 1.12 所示。

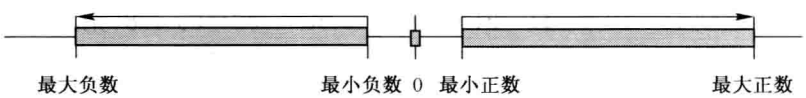

图 1.12　浮点数表示范围示意图

例 1.13　设阶码用 8 位补码表示，尾数部分用 16 位补码表示，则$-128.0625=-(2^7+2^{-4})=-(2^{-1}+2^{-12})\times 2^8=-0.100\ 000\ 000\ 001\ 000B\times 2^8$ 的尾数部分为$-0.100\ 000\ 000\ 001\ 000B$，补码为 1 011 111 111 111 000B；阶码部分为 8，即 00 001 000B，对应的十六进制数为 08BFF8H。

习题

1. 进制转换

129=_____ H=_____ B=_____ Q

298=_____ H=_____ B=_____ Q

1000=_____ H=_____ B=_____ Q

5DH=_____ B=_____ Q=_____ D

3E8H=_____ B=_____ Q=_____ D

357Q=_____ B=_____ H=_____ D

2. 进制计算

101101B+1101001B=_____ B

3FC9H−0FE6H=_____ H

一个字节的 NOT 8=_____ H=_____(有符号数)

两个字节的 NOT 8=_____ H=_____(有符号数)

5 AND 6=_____ D

5 OR 6=_____ D

3. 数据表示范围

一个字节的无符号数表示范围为_____,有符号数补码表示范围为_____。

两个字节的无符号数表示范围为_____,有符号数补码表示范围为 _____。

N 位二进制数的无符号数表示范围为_____,有符号数补码表示范围为_____。

4. 35H 代表的 ASCII 字符为_____,代表十六进制数时等价的十进制值为_____,代表压缩 8421 BCD 码等价的十进制值为_____,代表非压缩 8421 BCD 码等价的十进制值为_____。

5. FFH 代表无符号数时等价的十进制值为_____,代表补码有符号数时等价的十进制值为_____,代表反码有符号数时等价的十进制值为_____,代表原码有符号数时等价的十进制值为_____。

6. −20 的 8 位二进制补码为_____,原码为_____,反码为_____。

158 的 16 位二进制补码为_____,原码为_____,反码为_____。

7. 英文字符一般在计算机内占用_____个字节,每个字节的最高位一定为_____。全角英文字符在计算机内占用_____个字节,一个汉字在计算机内占用_____个字节,每个字节最高位为_____。

8. 设阶码用 8 位补码表示,尾数部分用 16 位补码表示,则−(1/32+1/128+1/512)的尾数部分及阶码分别为多少?

第 2 章　微机硬件基础

本章重点介绍学习汇编指令及编程前必须掌握的硬件知识。计算机硬件功能模块如图 2.1 所示，从图中可看出 CPU 可直接访问 CPU、内存和接口，CPU 在执行指令过程中就需要 CPU 去访问 CPU 中的寄存器、内存的存储单元和接口中的端口，所以本章主要介绍 8086/88 CPU 内部结构及寄存器、内在单元地址组织及存放次序、接口及端口等方面的硬件知识。

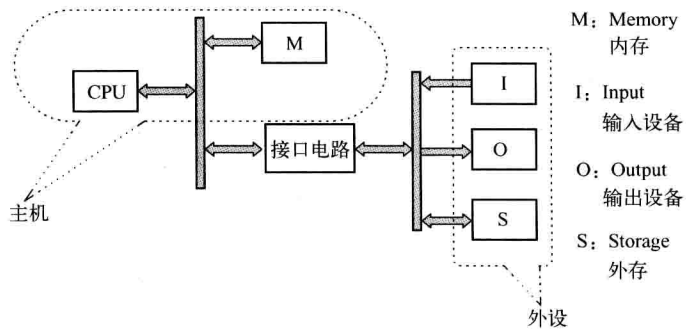

图 2.1　计算机硬件结构图

2.1　8086/88 CPU 的编程结构

8086/88 CPU 的内部结构是理解微机工作原理的重要部分，其寄存器构成及作用是编写汇编程序所必须掌握的。本节内容的掌握对后续章节的学习很重要。

1. 8086/88 CPU 的内部结构

在 8086/88 之前，微处理器执行指令的过程是串行的，即取出指令而后分析执行，再取下一条指令分析执行。为了使取指和分析、执行指令可并行处理、提高 CPU 的执行效率，8086/88 CPU 由两大模块——总线接口单元 BIU(Bus Interface Unit)和执行单元 EU(Execution Unit)组成，如图 2.2 所示。学会从所需功能推导出组成部件。

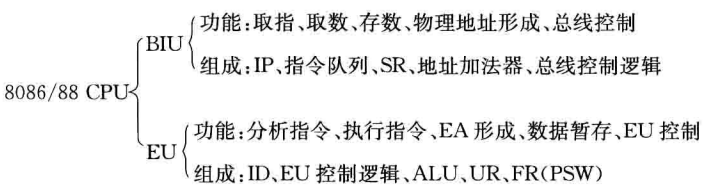

图 2.2　8086/88 CPU 的两大模块

图中英文缩写说明如下。IP：指令指针(Instruction Pointer)，SR：段寄存器(Segment Register)，ID：指令译码器(Instruction Decoder)，ALU：算术逻辑运算单元(Arithmetic Logic Unit)，UR：通用寄存器(Universal Register)，FR：标志寄存器(Flag Register)，PSW：程序状态字(Program Status Word)。

8086/88 CPU 的内部结构如图 2.3 所示。8086/88 之间的内部结构区别主要有一点：8086 指令列队有 6 字节，而 8088 指令列队只有 4 字节。

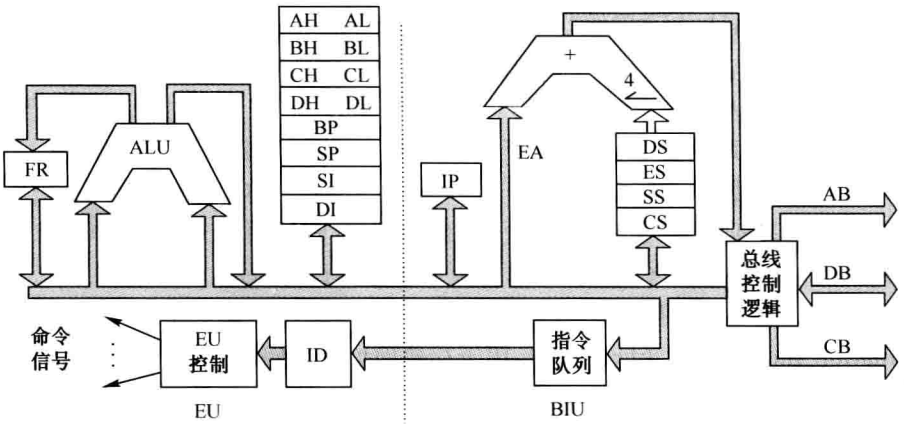

图 2.3　8086/88 CPU 的内部结构图

可以结合 ADD　AX,BX(将 AX+BX 的值送 AX)指令的取指及执行过程理解图 2.3,也可在图 2.3 的基础上更好地理解指令的取指及执行过程。授课时应该边讲解边画图,在画图的同时讲解指令的取指及执行过程,并且在图上写上相应序号。

(1) 取指:①由 CS:IP 形成取指的物理地址;②CPU 将此地址送地址总线 AB;③译码选中内存单元;④CPU 发出取指信号;⑤内存中指令送至数据总线 DB;⑥CPU 将读入的指令存入指令队列;⑦IP=IP+1,移向下一条指令。

(2) 执行指令:①指令队列中的指令送指令译码器译码;② 译码后执行单元发出相关命令信号完成指令的执行,即将 AX 送 ALU 的一端、BX 送 ALU 的另一端,并完成相加,结果送 AX,而且根据结果填充标志位。

2. 8086/88 CPU 内部的寄存器

汇编程序设计的要点之一是熟悉 CPU 的寄存器及它们的作用,本节仅对它们进行简单介绍,具体使用请参看 3.3 节。8086/88 CPU 中共有 14 个 16 位寄存器(R),其分类如图 2.4所示。这 14 个寄存器均为 16 位二进制的寄存器,其中 4 个通用数据寄存器 AX、BX、CX 和 DX 可分为 8 个 8 位寄存器 AH、BH、CH、DH 和 AL、BL、CL 和 DL 使用。通用寄存器主要用于存放一般数据。

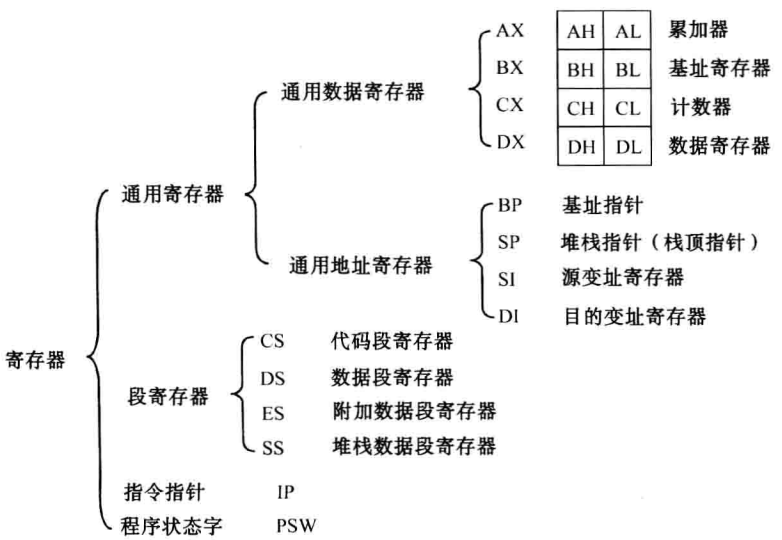

图 2.4　8086/88 内部寄存器

AX(Accumulator)(AH、AL)累加器,它是汇编编程中最常用的一个寄存器,主要用于乘除运算、BCD运算、换码、I/O操作、串操作和中断调用等。

BX(Base)(BH、BL)基址寄存器,主要用于存放地址和基址(默认相对于DS段)等。

CX(Counter)(CH、CL)计数器,主要用于循环计数、串操作计数和移位计数(CL)等。

DX(Data)(DH、DL)数据寄存器,主要用于16位乘除、间接I/O和中断调用等。

BP(Base Pointer)基址指针,主要用于存放地址和基址(默认相对于SS段)等。

SP(Stack Pointer)堆栈指针(栈顶指针),主要用于存放栈顶地址。

SI(Source Index)源变址寄存器,用于存放地址、变址和串操作源变址。

DI(Destination Index)目的变址寄存器,用于存放地址、变址和串操作目的变址。

CS(Code Segment)代码段寄存器(代码段),用于存放正在或正待执行的程序段的起始地址的高16位二进制数据,即程序段的段地址。

DS(Data Segment)数据段寄存器(数据段),用于存放正在或正待处理的一般数据段的起始地址的高16位二进制数据,即一般数据段的段地址。

ES(Extra Segment)附加数据段寄存器(附加段),用于存放正在或正待处理的附加数据段的起始地址的高16位二进制数据,即附加数据段的段地址。

SS(Stack Segment)堆栈数据段寄存器(堆栈段),用于存放正在或正待处理的堆栈数据段的起始地址的高16位二进制数据,即堆栈数据段的段地址。

IP(Instruction Pointer)指令指针,它的内容始终是下一条待执行指令的起始偏移地址,与CS一起形成下一条待执行指令的起始物理地址。CS:IP的作用是控制程序的执行流程。IP一般会自动加1(逻辑加1、实际随指令长度变化)移向下一条指令实现顺序执行;若通过转移类指令修改CS或IP的值,则可实现程序的转移执行。

PSW(Program Status Word)程序状态字,它有3个控制标志(IF、DF、TF)和6个状态标志(SF、PF、ZF、OF、CF、AF)。控制标志是用于控制CPU某方面操作的标志,状态标志是部分指令执行结果的标志(下面介绍的是状态标志的通用填充方法,特定指令特定的填充方法将在指令系统中介绍)。

IF(Interrupt enable Flag)中断允许标志,用于控制CPU能否响应可屏蔽中断请求,IF=1能够响应,IF=0不能响应。

DF(Direction Flag)方向标志,用于指示串操作时源串的源变址和目的串的目的变址的变化方向,DF=1向减的方向变化,DF=0向加的方向变化。

TF(Trap Flag)陷阱标志(单步中断标志),TF=1程序执行当前指令后暂停,TF=0程序执行当前指令后不会暂停。

SF(Sign Flag)符号标志,指令执行结果的最高二进制位是0还是1,为0,则SF=0,代表正数;为1,则SF=1,代表负数。我们一般是用十六进制数表示,则可以看十六进制的最高位是落在0～7还是落在8～F之间,若落在0～7之间则SF=0,否则SF=1。

PF(Parity check Flag)奇偶校验标志,指令执行结果的低8位中1的个数是奇数个还是偶数个,若为奇数个则PF=0,若为偶数个则PF=1。

ZF(Zero Flag)零标志,指令执行结果是不是为0,若为0则ZF=1,否则ZF=0。

OF(Overflow Flag)有符号数的溢出标志,指令执行结果是否超出有符号数的表示范围,若超过则OF=1,否则OF=0。我们可以通过是否出现以下四种情况之一来判断:正加正得负,正减负得负,负加负得正,负减正得正。若出现则OF=1,否则OF=0。

CF(Carry Flag)进位/借位标志(无符号数的溢出标志),指令执行结果的最高位是否有向更