汇编语言程序设计

# IBM PC-8088

# 汇编语言程序设计

雍正正　黄云森
庄　华　宁
汤弘寿

科海总

# 编译者序

书根据美国Richard E. Haskell所著的“IBM PC-8088 Assembly Language
mming”编译而成。原书是近两年来北美高等院校计算机专业本科所选用的教材之
的特点是采用了一个计算机辅助教学软件：“TUTOR监控程序”。此软件将计算
的操作过程直观地显示在屏幕上。此外，本书将8088汇编语言的教学与IBM PC机
介绍有机地融合在一起。因此，此书不仅对具有一定计算机基础的人员来说是一本
材和参考书，而且还特别适合于非计算机专业人员作为自学用书。

书由深圳大学电脑中心雍正正、黄云森、庄华、宁文编译。具体分工如下：第一章
由黄云森同志执笔，第六章至第八章由庄华同志执笔，第九至第十二章由宁文同
第十三章至第十四章由雍正正同志执笔。清华大学计算机科学与技术系汤弘寿同
全书的审校工作。

译者的水平及时间所限，本书一定存在不少错误与不足之处，敬请读者不吝赐

一九八八年四月一日

# 前 言

IBM PC是目前在商业和专业应用中用得最普遍的微型计算机。为了从计算机中得最高性能，用汇编语言书写程序或某些子程序是十分必要的。这本书将教你如何为IBM PC编写汇编语言程序。你也可以学到许多关于计算机的组织和IBM PC内部如何工作方面的知识。这本书可以作为计算机组织和汇编语言程序设计的入门课程的教材，也可以用作自学用书。

这本书描述了一个称为“导师”（TUTOR）的一种强有力的软件监控程序的使用。该监控程序，通过让你很容易地看到计算机内部如何工作，而帮助你理解8088微处理器。TUTOR监控程序，显示所有寄存器的内容和你所希望了解的存贮器任意区域的内容，它允许你单步执行程序，设置断点，及汇编任何程序代码，装入和检查任何磁盘文件等等。它可以被用作一个功能很强的调试工具，来对用IBM宏汇编语言建立的。EXE（可执行）文件进行调试。这样的。EXE文件，可以直接被装入至TUTOR中，进行检查，调整和执行。经这样调试所得的程序，可以不做任何改动而在DOS下运行。

经过第一章的一些介绍，在第二章你将开始使用TUTOR监控程序。你将学习十六进二进制与十进制数之间的关系。在第三章，你将学习如何用段地址和偏移地址形成的地址空间。你将看到如何检验和改变任意存贮器单元的内容。你也将看到ASCII码被用来表示计算机中的字符。

8088寄存器在第四章加以描述。与通用寄存器有关的指令也在这里说明。二进制和十进制算术运算在第五章叙述。状态标志在分支指令中的使用，在第六章描述。

第七章，你将学习关于堆栈和子例行程序。在这一章，你也将学习如何使用IBM PC的喇叭和键盘。在第七章的末尾，你可以开始使用IBM宏汇编程序（它的操作在第十四章叙述）。寻址方式这一重要课题在第八章涉及。8088的所有寻址方式都将加以描述，并用简单的例子来讲解它们的使用。

IBM PC在屏幕上显示字符的方法在第九章叙述。图形在第十章加以讨论。在那里开发一个画直线的子例行程序。第十一章涉及多字节运算，那里包括了完成多字节加法，减法，乘法和除法的一整套子例行程序。

第十二章描述IBM PC中所用的中断，包括硬件中断和软件中断，并介绍一个在IBM PC中显示实时时钟的程序。IBM PC的输入输出系统在第十三章中涉及。并且还讨论游戏杆接口，打印机接口、串行I/O接口以及如何完成磁盘I/O。

# 目　录


前　言

第一章　开篇 …… ( 1 )

第二章　8088微处理器 …… ( 2 )

2.1　数据总线 …… ( 2 )

2.2　二进制和十六进制计算 …… ( 3 )

2.3　计数系统 …… ( 7 )

2.4　内部寄存器 …… ( 8 )

2.5　十进制与十六进制之转换 …… ( 12 )

第三章　计算机存贮器 …… ( 14 )

3.1　8088地址空间 …… ( 14 )

3.2　TUTOR的存贮器显示 …… ( 15 )

3.2.1　滚动查阅存贮器 …… ( 16 )

3.2.2　转向（>）任意存贮器单元 …… ( 16 )

3.2.3　改变存贮器内容 …… ( 21 )

3.2.4　ASCII　数据 …… ( 23 )

第四章　8088的寄存器 …… ( 28 )

4.1　通用寄存器 …… ( 28 )

4.2　指令指针IP …… ( 30 )

4.3　状态寄存器 …… ( 32 )

4.3.1　进位（C） …… ( 32 )

4.3.2　零标志（Z） …… ( 33 )

4.3.3　符号标志（S） …… (

4.3.4　溢出标志（O） …… (

4.3.5　中断允许标志（I） …… (

4.3.6　辅助进位（A） …… (

4.3.7　奇偶标志（P） …… (

4.3.8　方向标志（D） …… (

4.3.9　陷井标志（T） …… (

4.4　一些通用寄存器指令 …… (

4.5　移位和循环指令 …… (

4.5.1　左移SHL或SAL …… (

4.5.2　右移SHR …… (

4.5.3　带进位循环左移RCL …… (

4.5.4　带进位循环右移RCR …… (

4.5.5 循环左移和循环右移ROL，ROR ……（38）
4.5.6 算术右移SAR ……（39）
4.5.7 带计数的移位和循环 ……（39）
4.6 段寄存器CS，DS，SS，ES ……（40）
4.7 索引寄存器SI和DI ……（40）
4.8 传送和交换指令 ……（41）
4.9 指针寄存器SP和BP ……（43）
**第五章 8088的运算** ……（44）
5.1 二进制加法 ……（44）
5.1.1 负数 ……（45）
5.1.2 进位和溢出 ……（48）
5.2 二进制减法 ……（50）
5.3 16位加法和减法 ……（51）
5.4 二进制乘法 ……（53）
5.4.1 无符号乘法 ……（53）
5.4.2 带符号乘法 ……（56）
5.5 二进制除法 ……（57）
5.6 压缩的BCD运算 ……（58）
5.6.1 BCD加法 ……（58）
5.6.2 BCD减法 ……（59）
5.7 非压缩的BCD运算 ……（59）
5.7.1 ASCII加法 ……（60）
5.7.2 ASCII减法 ……（60）
5.7.3 BCD乘法 ……（61）
5.7.4 BCD除法 ……（61）
**第六章 分支指令** ……（63）
8088条件转移指令 ……（63）
8088无条件转移指令 ……（64）
计算跳转偏移量 ……（65）
分支举例 ……（66）
6.4.1 零标志位Z转移 ……（66）
6.4.2 符号标志位S转移 ……（66）
6.4.3 进位标志位C转移 ……（67）
其他条件转移指令 ……（68）
循环（Loop）指令 ……（69）
**堆栈和子程序** ……（71）
堆栈 ……（71）
7.1.1 PUSH和POP指令 ……（71）
子程序 ……（74）

7.3 延时 …… (78)
7.3.1 软件延时 …… (78)
7.4 IBM PC的发声 …… (79)
7.4.1 设置断点 …… (79)
7.4.2 执行程序 …… (80)
7.5 IBM PC键盘 …… (81)
7.5.1 将程序存入磁盘 …… (85)
**第八章 寻址方式** …… (87)
8.1 汇编语言编程 …… (87)
8.2 立即寻址方式 …… (87)
8.3 8088寻址方式字节 …… (90)
8.4 寄存器寻址 …… (91)
8.5 直接寻址方式 …… (93)
8.6 寄存器间接寻址方式 …… (98)
8.6.1 段替换前缀 …… (99)
8.6.2 串指令 …… (99)
8.7 索引寻址 …… (102)
8.8 基址寻址 …… (103)
8.9 索引加基址寻址方式 …… (105)
8.10 堆栈寻址方式 …… (110)
**第九章 屏幕上的字符显示** …… (111)
9.1 光栅扫描显示 …… (111)
9.2 IBM PC的显示缓冲区 …… (113)
9.2.1 彩色/图形监视器的适配器 …… (113)
9.2.2 单色显示器 …… (116)
9.2.3 属性码字节 …… (117)
9.3 屏幕上的字符显示 …… (117)
9.3.1 一个字符的输出子程序 …… (122)
9.4 显示信息 …… (124)
9.5 清屏 …… (128)
9.6 键盘控制 …… (130)
9.6.1 屏幕上键入字符 …… (131)
9.6.2 转移表的使用 …… (132)
**第十章 IBM PC的图形显示** …… (136)
10.1 IBM PC的图形方式 …… (136)
10.2 画点 …… (138)
10.3 画线 …… (140)
**第十一章 多字节运算** …… (149)
11.1 二进制运算 …… (149)

11.1.1 二进制加法 …… (149)
11.1.2 二进制减法 …… (150)
11.2 非压缩型BCD码的运算 …… (151)
11.2.1 十进制加法 …… (151)
11.2.2 通过堆栈传递参数 …… (152)
11.2.3 十进制减法 …… (155)
11.3 压缩型BCD码的运算 …… (156)
11.3.1 BCD码的加法 …… (163)
11.3.2 BCD码的减法 …… (165)
11.3.3 BCD码的乘法 …… (166)
11.3.4 BCD码的除法 …… (167)
**第十二章 中断** …… **(199)**
12.1 8088中断 …… (169)
12.1.1 复位 …… (169)
12.1.2 硬件中断 …… (169)
12.1.3 软件中断 …… (172)
12.1.4 单步操作 …… (173)
12.2 IBM PC机的中断 …… (173)
12.2.1 DOS中断 …… (175)
12.3 实时时钟 …… (177)
**第十三章 IBM PC输入/输出** …… **(185)**
13.1 8088输入/输出 …… (185)
13.1.1 IBM PC I/O地址空间 …… (186)
13.2 IBM PC喇叭接口 …… (187)
13.3 打印机接口 …… (189)
13.4 串行I/O …… (194)
13.4.1 IBM RS-232通讯接口 …… (195)
13.4.2 IBM PC终端 …… (200)
13.5 磁盘输入/输出 …… (202)
13.5.1 BIOS磁盘I/O程序 …… (203)
13.5.2 DOS磁盘I/O程序 …… (206)
13.5.2.1 顺序读/写 …… (207)
13.5.2.2 随机读/写 …… (213)
13.5.2.3 随机块读/写 …… (216)
13.5.2.4 DOS 2.0磁盘I/O程序 …… (217)
**第十四章 IBM PC宏汇编** …… **(223)**
14.1 8088汇编语言程序的一般结构 …… (223)
14.2 使用汇编程序的步骤 …… (225)
*14.2.1 使用行编辑器EDLIN* …… (226)

14.2.2 使用宏汇编…… (228)
14.2.3 使用链接程序LINK…… (230)
14.2.4 运行程序…… (231)
14.3 用TUTOR调试.EXE程序 …… (231)
14.4 链接多个模块 …… (234)

# 第一章 开 篇

IBM个人计算机（PC—Personal Computer）已经成为一个小型和台式计算机的工业标准。它使用Intel 8088微处理器。为了从IBM PC获得最好的性能，以及了解它内部如何工作，用8088汇编语言为计算机编程序是很需要的。

8088微处理器是什么，它是如何工作的？这正是这本书要解决的。这本书独特地利用了一个在IBM PC中运行的称为“导师”（TUTOR）的特殊监控程序，来帮助你学习微处理器。TUTOR监控程序向你展示计算机内部任何瞬间所发生的事情。经过特殊设计，它很容易使用，并使你较容易地学习微处理器。

当编写长的汇编语言程序时，你需要使用一个汇编程序来将指令助记符自动转换为机器语言代码。在第十四章，我们简短地描述IBM PC宏汇编的操作。然而，如果你对于一些短程序用手工进行初始的转换，你将对于微处理器的操作得到相当透彻的了解。TUTOR监控程序使你很容易地将你的程序输入到计算机中，运行它们，并且调试它们。TUTOR监控程序也可以用作一种强有力的工具，来对你用汇编语言写的程序进行调试。

一个TUTOR监控程序的摘要和关于如何运行它的叙述，在附录B中给出。你现在就应该很快地看一下这个附录。它对你将是一个很有用的指南，直至你对它的使用熟悉为止。大多数的命令都将在这本书中介绍，需要时还将举例加以说明。你将发现，TUTOR监控程序是很容易使用的。

虽然高级语言到处都有，你仍应该对学习汇编语言感兴趣。这是为什么呢？有几个理由。首先，汇编语言程序运行速度快。对于一个汇编语言程序，它的运行速度，比用BASIC语言写的相应程序快几个数量级，这是很普通的事。如果你想在瞬间用信息填满屏幕，或者控制某些对时间要求很苛刻的过程，那么这就显得十分重要了。

其次，汇编语言程序设计使你接近了最低层的计算机硬件。这对于为计算机与外部设备之间做接口，或者试图从计算机硬件中获得最高性能，都是极为重要的。第三，大部分微处理器的应用都是在专用系统中，在那里总在运行单一程序。这种程序通常都是以机器码存贮于一个只读存贮器（ROM）或可编程序只读存贮器（PROM）中的。这样的程序大多数是用汇编语言写的。

最后，学习汇编语言，将使你对于计算机如何工作有一个更好的理解。围绕着计算机，通常总有相当程度的神秘感。这种神秘感，不可避免地导致怀疑和畏惧。大的计算机设备，被一批精通的计算机人材包围着，不能够提供给一般人对计算机是什么和计算机是如何工作的有较多的理解。小的个人计算机，例如IBM PC，使得这成为可能：你将成为你自己的计算机的主人。这本书将打开IBM PC，向你展示是什么使它运转的。你将可以使它做任何你想做的事情。为了完全地理解和掌握它们，你必须亲身去体验那些东西。因此，让我们开始吧。

# 第二章　8088微处理器

8088微处理器是一个40脚的集成电路片子（见图2.1），它价格不到15美元，可以用于很广的应用范围。它可以用来做的其中一件事，是构成一个象IBM PC这样的通用计算机。在这本书中，你将学习这件事是如何完成的，以及如何使用8088或类似的微处理器来做许多不同的事情。这个芯片是IBM PC的“大脑”，见图2.2。板子上余下的芯片是主存贮器（见第三章），其它的芯片则与各种输入输出（I/O）操作有关。

## 2.1　数据总线

图2.3是8088芯片示意图。这个芯片是由Intel公司和Mostek公司生产的。

图 2.1　8088微处理器

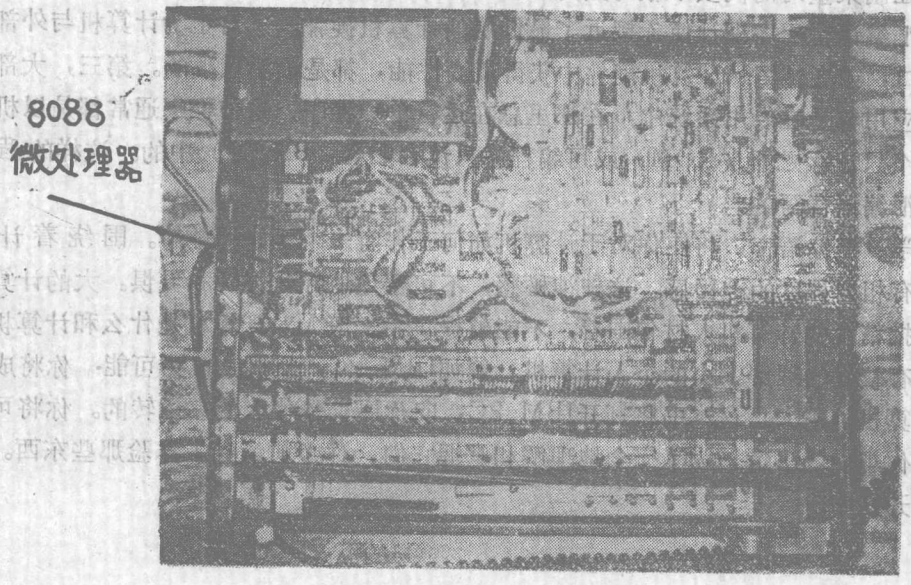


图 2.2　8088微处理器是IBM PC机的“大脑”

引腿9～16标着 AD0～AD7，形成了一个8位（bit）的双向数据总线。同是这8条线，又形成地址总线的低8位。但地址和数据出现在这些线上的时间是不同的(称为多路复用)。所有在8088微处理器和存贮器芯片间传送的数据都以8位为一组（称为字节—Byte）的方式通过这条数据总线。一个高电位被作为二进制数位（bit）的“1”，而一个低电位被作

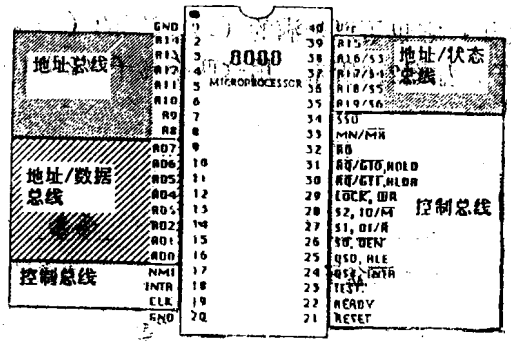


图 2.3 8088微处理器引腿说明

为二进制位的“0”。这样，在某些时刻，数据总线可以含有8位(即一个字节)01101010。最右边的位是D0，是最小的有效位（LSB）。最左边的位是D7，它是最大的有效位（MSB），参见图2.4。

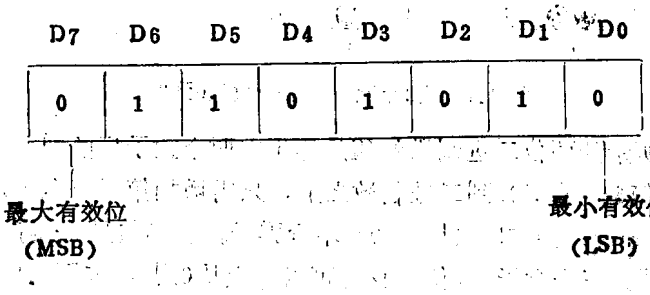


图 2.4 在数据总线上的一个字节

当使用8088微处理器时，你将是用8位的字节进行处理的。因此，了解如何用二进制计数是很重要的。用一种称为十六进制的简化符号来表示二进制数，是很方便的。

## 2.2 二进制和十六进制记数法

考虑一个盒子里装一个弹子的情况。如果弹子在盒子里，我们就说盒子是满的，并且把盒子看作“1”；如果我们将弹子从盒子中取走，盒子就是空的，我们将把盒子看作为“0”。这两个二进制数0和1被叫做比特（bit），用一个位表示，我们可以计数从0（盒子空）到1（盒子满），如图2.5所示。

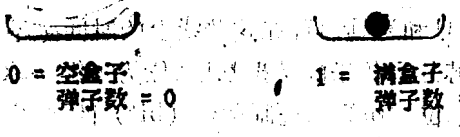


图 2.5 用一位可以计数0到1

现在考虑有第二个盒子，它也可以是满的（1）或空的（0）。然而，当这个盒子为满的时候，它将含有2个弹子，如图2.6所示。用这两个盒子（2位），我们可以计数从0到3，如图2.7所示。


图 2.6 这个盒子可以含有2个弹子（满的）或没有弹子

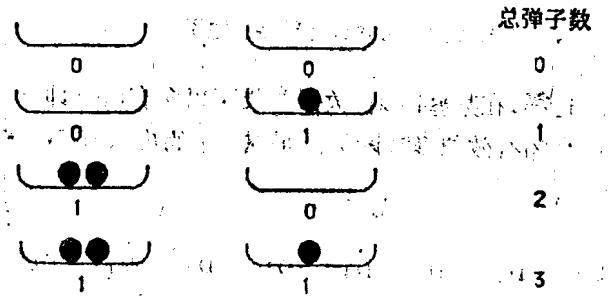


图 2.7 用2个比特可以计数从0到3

请注意：图2.7所示的两位二进制数的值是等于在两个盒子里的弹子的总数。

我们可以增加第3位（bit）到二进制数之上。只需增加第三个盒子，这个盒子满时（bit=1）含有4个弹子，而空时（bit=0）不含弹子。它必须只是满（bit=1）和空（bit=0）之一。用这三个盒子（3位）我们可以计数从0到7，如图2.8所示。

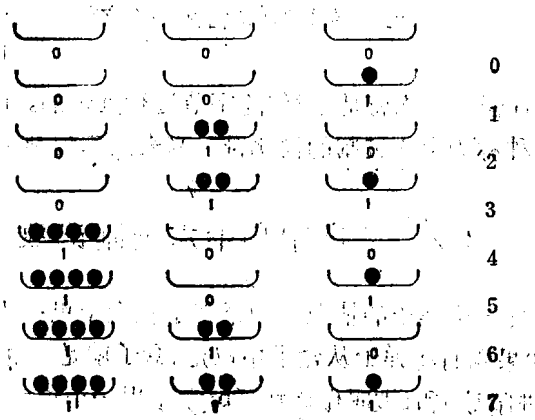


图 2.8 用3个比特可以计数从0到7

如果你想计数超出7，你必须增加另一个盒子。当这第四个盒子为满（bit=1）时，应该含有几个弹子呢？很显然，这个盒子应该含有8个弹子。8写成二进制数是1000。记住，在二进制中，“1”表示相应盒子中的弹子是满的；而“满”的盒子，弹子的数量从右开始按1、2、4、8变化。这就意味着，用4位，我们可以计数从0到15，示于图2.9。

图2.9还显示出，用四位的二进制数来表示四个盒子中弹子总数的方法，如用一位数来表示则更加方便。我们称之为十六进制数。16个十六进制数字在图2.9的最右边一列给出。十六进制的0～9与十进制的0～9是相同的。然而，十进制的10～15在十六进制中则表示为A～F。例如，十六进制数D等于十进制数13。

| 每个满盒子中的弹子数（bit=1） | | | | 弹子总数 | 十六进数字 |
|---|---|---|---|---|---|
| 8 | 4 | 2 | 1 | | |
| 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 | 10 | A |
| 1 | 0 | 1 | 1 | 11 | B |
| 1 | 1 | 0 | 0 | 12 | C |
| 1 | 1 | 0 | 1 | 13 | D |
| 1 | 1 | 1 | 0 | 14 | E |
| 1 | 1 | 1 | 1 | 15 | F |

图 2.9 用4比特可以计数从0到15

为了用二进制来表示超出15的数，你必须增加盒子。每一个满盒子里的弹子数，必须是前一个满盒子里弹子数的两倍。用8位，你可以计数从0到255。图2.10是一些例子。与给定的二进制数相对应的十进制数，等于所有盒子里面弹子的总数。为了找到这个数，只需将满的盒子里的弹子数加到一起就可以了（那些满的盒子是以二进制数“1”表示的）。

| 每个满盒子中的弹子数 (bit = 1) | | | | | | | | 弹子总数 |
|---|---|---|---|---|---|---|---|---|
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 52 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 163 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 255 |

图 2.10 用8比特可以计数从0到255

当二进制数长度增加时，使用起来变得很麻烦。因此，我们使用对应的十六进制数，作为表示二进制数的一种缩写方法。这种方法做起来很容易。你只须将二进制数从右边开始按4位一组进行分组，然后将每个4位的组，用图2.9给出的对应的十六进制数字来表示。例如，二进制数

$$\underbrace{1001}_{9}\ \underbrace{1010}_{A}$$

等于十六进制数9AH。我们将经常使用字母H跟在数字后边，以指出这是个十六进制数（Hexadecimal）。你可以验证用这个二进制数表示的弹子的总数是154。然而，你也可以通过计算十六进制盒子中的弹子数以代替计算二进制盒子中的弹子数。在十六进制盒子中，第一个盒子含有A×1=10个弹子。而第二个盒子含有9×16=144个弹子。因此，弹子的总数为144+10=154。

第三个十六进制盒子将含有$16^2=256$的倍数的弹子，而第四个十六进制盒子将含有$16^3=4,096$个弹子。作为一个例子，如下一个16位的二进制数

$$\underbrace{1000}_{8}\ \underbrace{0111}_{7}\ \underbrace{1100}_{C}\ \underbrace{1001}_{9}$$

等于十进制数34,761个弹子。这可以通过如下方法展开十六进制数便可看出：

$$\begin{aligned}
8\times16^3 &= 8\times4,096 = 32,768\\
7\times16^2 &= 7\times256 = \ \ 1,792\\
C\times16^1 &= 12\times16 = \ \ \ \ 192\\
9\times16^0 &= 9\times\ \ 1 = \ \ \ \ \ 9\\
&\qquad\qquad\qquad 34,761
\end{aligned}$$

由此你可以看到，使用十六进制数可以使位数减少至原先的四分之一。

表2.1令你能方便地将高达4位的十六进制数转换为它的十进制数。例如，你可注意一下，在87C9H这个数的转换中，它的4个项是如何从表中直接读得的。

表 2.1 十六进制与十进制间的转换

| 15 | BYTE | 8 | 7 | BYTE | 0 |
|---|---|---|---|---|---|
| 15 CHAR | 12 | 11 CHAR | 8 | 7 CHAR | 4 | 3 CHAR | 0 |
| HEX | DEC | HEX | DEC | HEX | DEC | HEX | DEC |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 4,096 | 1 | 256 | 1 | 16 | 1 | 1 |
| 2 | 8,192 | 2 | 512 | 2 | 32 | 2 | 2 |
| 3 | 12,288 | 3 | 768 | 3 | 48 | 3 | 3 |
| 4 | 16,384 | 4 | 1,024 | 4 | 64 | 4 | 4 |
| 5 | 20,480 | 5 | 1,280 | 5 | 80 | 5 | 5 |
| 6 | 24,576 | 6 | 1,536 | 6 | 96 | 6 | 6 |
| 7 | 28,672 | 7 | 1,792 | 7 | 112 | 7 | 7 |
| 8 | 32,768 | 8 | 2,048 | 8 | 128 | 8 | 8 |
| 9 | 36,864 | 9 | 2,304 | 9 | 144 | 9 | 9 |
| A | 40,960 | A | 2,560 | A | 160 | A | 10 |
| B | 45,056 | B | 2,816 | B | 176 | B | 11 |
| C | 49,152 | C | 3,072 | C | 192 | C | 12 |
| D | 53,248 | D | 3,328 | D | 208 | D | 13 |
| E | 57,344 | E | 3,584 | E | 224 | E | 14 |
| F | 61,440 | F | 3,840 | F | 240 | F | 15 |

## 2.3 计数系统

二进制数是以2为基的数，十六进制数是以16为基的数。一个数N可以以任何基b写成如下的位置表示方式：

$$N = P_4P_3P_2P_1P_0b = P_4b^4 + P_3b^3 + P_2b^2 + P_1b^1 + P_0b^0$$

在这里，数总是从右边的最小有效位开始的。

例如，十进制数584是一个以10为基的数，它可以表示为

$$584_{10} = 5\times10^2 + 8\times10^1 + 4\times10^0$$
$$= 500 + 80 + 4$$
$$= 584_{10}$$

以b为基的数，必须有b个不同的数字。于是，十进制数（以10为基）就使用10个数字0～9。

二进制数是以2为基的数，因此就仅使用“0”和“1”两个数字。例如，二进制数110100是以2为基的数

$$110100_2 = 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0$$
$$= 32 + 16 + 0 + 4 + 0 + 0$$
$$= 52_{10}$$

这与图2.10的第一个例子是相同的，在那里弹子的总数是52（32+16+4）。

十六进制数是一个以16为基的数，因此需要十六个不同的数字来加以表示。我们使用0～9这十个数字加上A～F这六个字母，见图2.9。例如，十六进制数3AF可以写成以16为基的数

$$3AF_{16} = 3\times16^2 + A\times16^1 + F\times16^0$$
$$= 3\times256 + 10\times16 + 15\times1$$
$$= 768 + 160 + 15$$
$$= 943_{10}$$

微计算机中传送数据，通常是以8位的二进制字节组进行的。因此，在计算机中就很自然地以二进制（以2为基的）数来作描述。正如我们已经看到的，使用十六进制数则更简单些，十六进制的每一位表示4个二进制位。有些较大的计算机，用3位为一组来表示数，而不用4位为一组。得到的数是一个八进制的（以8为基的）数。八进制数仅使用0～7共8个数字。例如，八进制数457可以写为以8为基的数

$$457_8 = 4\times8^2 + 5\times8^1 + 7\times8^0$$
$$= 256 + 40 + 7$$
$$= 303_{10}$$

在IBM PC中，我们将仅用二进制和十六进制来描述数据。这一节中的例子，告诉你如何将一个以任意数为基的数，转换成一个十进制数。在这一章的后面一点。我们将看到是如何将一个十进制数转换为十六进制数的。

## 2.4 内部寄存器

8088微处理器有几个可以存贮二进制数据的内部寄存器。这些寄存器示于图2.11。所有这些寄存器都是16位（比特）的寄存器。通用寄存器AX，BX，CX和DX，每个都可以进一步分成高8位字节和低8位字节。例如，AX是由高字节AH和低字节AL组成的。我们将在第四章更详细地讨论这些寄存器。

如果你运行监控程序，你将得到如图2.12*所示的屏幕显示。所有内部寄存器的当前内容都显示在屏幕的上半部。所有寄存器的值都以十六进制方式显示，只有状态标志（STATUS FLAGS）例外，状态标志是以十六进制和二进制两种方式显示的。

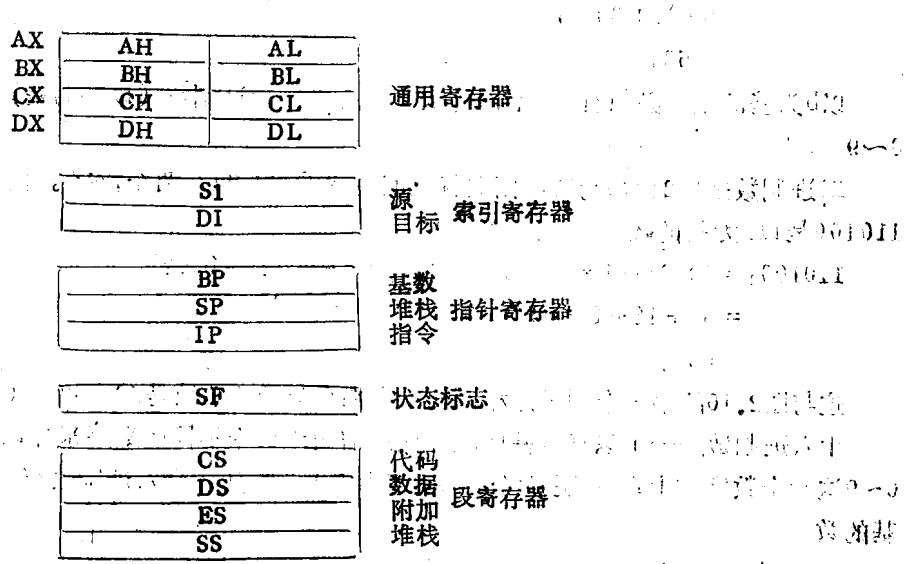


图 2.11 8088微处理器的内部寄存器