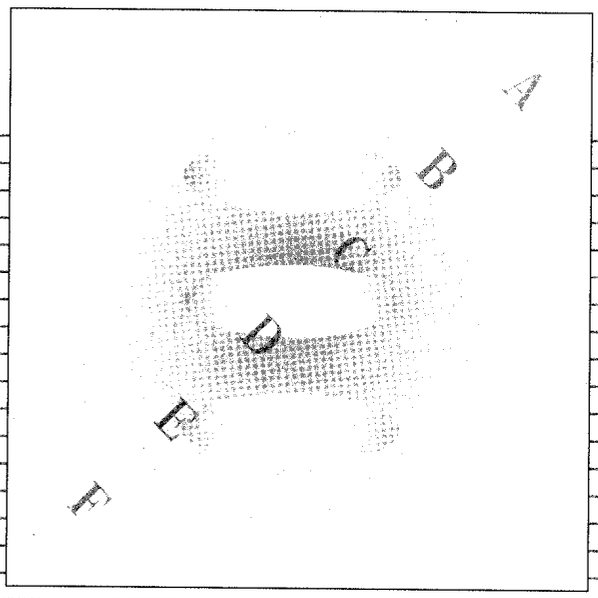
北京市高等教育学历文凭考试计算机专业教材

IBM-PC汇编语言程序设计

熊桂喜 编著



科学出版社

内 容 简 介

本书是根据国务院发布的《高等教育自学考试暂行条例》，以及北京市高等教育自学考试委员会关于《高等教育学历文凭考试课程大纲》编写的，其内容的深度和广度符合大纲要求。

本书较系统地介绍了基于 Intel 8086/8088 CPU 的 IBM 宏汇编语言程序设计的基本知识、基本原理和程序设计技术。

全书共分 12 章。第一、二章介绍了 IBM-PC 的基础知识；第三章介绍了寻址方式及指令系统，第四章介绍了组织汇编程序所需的伪指令及操作符、程序结构；第五、六章以大量实例介绍了用汇编语言编程的基本技术；第七章介绍了宏和结构；第八～十一章介绍了用汇编语言进行 I/O 程序设计的技术；第十二章简要介绍了多模块程序设计技术。各章内容结构清晰，重点突出。每章后还附有一定数量的习题。

本书由北京市高等教育自学考试委员会推荐使用，不仅可作为高等教育自学考试计算机专业文凭考试课程的理想教材，还可以作为各类高等专科学校、职工大学、职业大学、夜大学，以及函授大学等大专类“汇编语言”课程的教材与教学参考书。

图书在版编目(CIP) 数据

IBM-PC 汇编语言程序设计/熊桂喜编著．－北京：科学出版社，1998.8
北京市高等教育学历文凭考试计算机专业教材
ISBN 7-03-006814-9

Ⅰ.IBM… Ⅱ.熊… Ⅲ.汇编语言-程序设计-高等教育-教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（98）第 16506 号

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717

北京双青印刷厂 印刷

新华书店北京发行所发行 各地新华书店经售

*

1998 年 8 月第 一 版 开本：787×1092 1/16
1998 年 8 月第一次印刷 印张：16
印数：1－4 000 字数：356 000

定 价：21.00 元

（如有印装质量问题，我社负责调换〈环伟〉）

编委会名单

序

高等教育学历文凭考试，是我国高等教育事业发展过程中出现的一个新生事物。它是社会力量办学与国家考试相结合，以宽进严出、教考分离、全日制教学为特点的高等教育形式。这种形式从它产生之日起，就受到社会各界的重视和赞誉，认为它为我国社会主义经济建设对人才的大量需求又提供了一种培养手段，同时它也得到了国家有关领导同志的称赞，认为这是“穷国办大教育”的一条好的途径。北京在全国是最早进行这项试点工作的，目前有 24 所民办高校参加，开设的专业有 16 个，在校生 3 万余人，年均招生规模都在万人左右，其中计算机应用专业 1997 年成为招生规模最大的专业。经过五年的实践和总结，特别是结合国家为民办高等教育培养目标“应用性、职业性”定位的理解，我们感到，我市试点工作在发展过程中，基本建设做得还不够，这其中一个表现就是抓教材建设做得不够，特别是目前市场上还缺乏与高等职业技术教育相匹配的有关教材，导致目前参加文凭考试的民办学校在教学上基本上是借用普通高校的教材，因而教材与培养要求上的矛盾就尤显突出。

我们非常感谢的是，北京航空航天大学计算机系非常积极和认真地提出了要组织编写一套适合这种考试的教材的建议，并做了很具体的安排。北京航空航天大学的许多专家、教授克服了学校教学、科研任务繁忙的困难，按时、保质地完成了撰稿工作。同时此项工作也得到了科学出版社有关同志的大力配合，没有他们耐心、细致地做组稿和出版等工作，这套教材能这样快地面世是不可想象的。

经过各方的通力协作，今天这套教材终于可以奉献给大家了。我们觉得这套教材基本上体现了高等教育学历文凭考试计算机应用专业培养方向上的要求，在内容上也是科学、严谨的，我们同意把这套教材作为推荐使用的教材。同时，我们也希望社会各界对这套教材有什么意见和建议，能及时反馈给我们，以便使它不断完善。

谢谢大家。

北京市高等教育自学考试委员会办公室

1998 年 6 月 12 日

前　言

《IBM-PC 汇编语言程序设计》几乎是所有计算机专业的一门专业必修课。学习汇编语言的要求有：

（1）加深对于微机结构及其内部运行过程的理解。

（2）掌握 Intel 8086/8088 的宏汇编语言，了解其语言成分、语法结构，能够读懂和编写汇编语言程序。

（3）了解 IBM-PC 的 BIOS，MS-DOS 的支持基础，掌握 BIOS，MS-DOS 的功能调用方法，并能运用主要的功能调用。

（4）了解 IBM-PC 主要外部设备的工作原理及简单编程方法。

（5）掌握汇编语言的编程、调试技术，能够以汇编语言为工具，解决一些与数据结构、外部设备、系统硬件有关的实际问题。

按照上述要求，并结合大专教学特点，我们编写了本教材。全书共分 12 章，内容如下：

- 第一、二章介绍了 IBM-PC 的基础知识。
- 第三章介绍了指令系统及寻址方式。
- 第四章介绍了伪指令、操作符和汇编语言程序格式，以及上机调试的过程和方法。
- 第五、六章结合汇编语言程序结构知识，介绍了基本编程知识。
- 第七章简要介绍了宏及结构。
- 第八章介绍了 I/O 程序设计的基本技术，重点介绍了用汇编语言设计中断处理程序的内容。
- 第九章介绍了 DOS，BIOS 功能调用的方法及应用。
- 第十章简要介绍了两种显示卡的图形程序设计。
- 第十一章简要介绍了 DOS 句柄式文件访问技术。
- 第十二章简要介绍了多模块程序设计技术。

汇编语言是一门学习及讲授难度比较大的课程，它内容枯燥，需要记忆的东西较多，要了解的机器知识及编程技巧也较多。本教材在许多内容上专为大专课程要求作了调整。例如减少了指令的介绍，对复杂的数据结构及技巧也不作要求。但一些基本指令、基本结构、解决基本问题的方法仍是必不可少的。对学生的动手编程、上机调试能力也有要求。但本书内容在讲授和学习时，可以有所取舍。

限于作者水平，本书难免有错误及疏漏之处，欢迎读者指正。

编　者

1998 年 7 月

目　录

第一章　基础知识

在讲授汇编语言的指令、伪指令等内容之前，先介绍二进制、十进制、十六进制及ASCII码、BCD码以及数制与代码之间相互转换的知识。正确地理解数制之间的转换关系、数值与代码之间的关系，是学习汇编语言的基础。

对于具有一定计算机基础知识的读者，可将本章内容作为一个总结或复习。

1.1　数制及数制间的转换

目前计算机内部都采用二进制(binary)数进行操作和运算，因为这种方式比较符合机器的特点。在人们的日常生活中，表达及思维时，更习惯于使用十进制(decimal)数，它是一般算术和数学的基础。在表达和描述计算机中的内容时，还常用到十六进制(hexadecimal)数。二进制、十进制、十六进制，是我们与计算机打交道时经常用到的三种计数制。在一些高级语言中，还用到了八进制数，但在本书讲解的过程中，主要用到的是前三种数制。

1.1.1　二进制数、十进制数及十六进制数

逢十进位是十进制数的基本特点之一。实际上，这种进位计数是一种通用的方法，它同样适用于其他计数制，只不过它可能是“逢2进1”(二进制)或“逢16进1”(十六进制)。

以十进制为例，一个任意十进制数可表示为：

$$a_n\,a_{n-1}\cdots\,a_0.\;b_1\,b_2\cdots\,b_m$$

其含意是：

$$a_n\times10^n+a_{n-1}\times10^{n-1}+\cdots+a_0\times10^0+b_1\times10^{-1}+b_2\times10^{-2}+\cdots+b_m\times10^{-m}$$

其中，$a_i(i=1,2,\cdots,n)$及$b_j(j=1,2,\cdots,m)$是0,1,2,3,4,5,6,7,8,9十个数码中的任意一个。

上式中对应于每位数字的10^k称为该位数字的权(即常说的个位、十位、百位……)。一个数值即是每位数字乘以它的权值相加的结果。例如：

$$12345.67=1\times10^4+2\times10^3+3\times10^2+4\times10^1+5\times10^0+6\times10^{-1}+7\times10^{-2}$$

前面的公式同样适用于其他计数制。基数为r的r进制数的值可以表示为：

$$a_n\times r^n+a_{n-1}\times r^{n-1}+\cdots+a_0\times r^0+b_1\times r^{-1}+b_2\times r^{-2}+\cdots+b_m r^{-m}$$

其中a_i,b_j可以是$0,1,\cdots,r-1$中的任一个数码，r^k则为各位数相应的权。

对于二进制数，上述公式中的r值为2；对于十六进制数，上述公式中的r值为16。

由于二进制数的基数为2，因而每位数字只能为0或1，且遵循逢二进一的规则。各位数的权值为2^k。因此，前面的公式可以写为：

$$a_n\times 2^n+a_{n-1}\times 2^{n-1}+\cdots+a_0\times 2^0+b_1\times 2^{-1}+b_2\times 2^{-2}+\cdots+b_m\times 2^{-m}$$

其中，a_i，b_j 只能为 0，1 两个数码中的一个。

按照上述公式，一个 101010 的二进制数值为：

$$1\times 2^5+0\times 2^4+1\times 2^3+0\times 2^2+1\times 2^1+0\times 2^0=42_{10}$$

最后结果 42_{10}是一个十进制数，其下标表示的是此数的基数。转换成这样的结果，人们更易于理解，但实际上，101010_2 与 42_{10}是相同的数值。

n 位二进制数可以表示 2^n 个数(对应的 n 位十进制数可以表示 10^n 个数)。例如，4 位二进制数可以表示 2^4 个数，即 0～15 共 16 个数(在谈到"真正"的值、个数时，仍用十进制作为参考值)，它们是：

二进制数	0000	0001	0010	0011	0100	0101	0110	0111
十进制数	0	1	2	3	4	5	6	7
二进制数	1000	1001	1010	1011	1100	1101	1110	1111
十进制数	8	9	10	11	12	13	14	15

上述表中的 16 个二进制数以后会经常用到，应该记住。

如果理解了二进制数，则学习十六进制数就比较容易。十六进制的基数为 16，且遵循逢 16 进 1 的规则。每位数的数值为 0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F；其中，A，B，C，D，E，F 分别对应于十进制数的 10，11，12，13，14，15。十六进制数第 k 位数的权值为 16^k。因此，表示一个十六进制数的公式为：

$$a_n\times 16^n+a_{n-1}\times 16^{n-1}+\cdots+a_0\times 16^0+b_1\times 16^{-1}+b_2\times 16^{-2}+\cdots b_m\times 16^{-m}$$

其中，a_i，b_j 为 0，1，…，9，A，B，…，F 中的任意一个数。

例如，一个 $213B_{16}$的十六进制数，其值为：

$$213B_{16}=2\times 16^3+1\times 16^2+3\times 16^1+11\times 16^0=8507_{10}$$

在本书及其他书籍中，表示一个数的计数制时，常用如下符号：

D ——十进制数；B ——二进制数；H ——十六进制数

例如，2136H，8502D，101010B；如果省略最后一个字母，则表示它是一个十进制数。

1.1.2 二进制数与十进制数之间的转换

数值在机器内部进行存储和处理时，必须使用二进制数，而在输入输出过程中时，常常采用十进制数，这就需要在两种数制之间进行转换。

1. 二进制数转换成十进制数

将每位二进制数码乘以与该数码对应的权值，再将乘积相加，即得到对应的十进制数。例如：

$11111100B=2^7+2^6+2^5+2^4+2^3+2^2=252D$

2. 十进制数转换成二进制数

将十进制数转换成二进制数的办法很多，但最常见的，且最符合于常规思维习惯的是除法。具体做法是：将要转换的十进制数（整数）不断地除以 2，并记下余数，直到商为 0 时为止。例如：

252D ——252/2=126 余 0
126/2=63 余 0
63/2=31 余 1
31/2=15 余 1
15/2=7 余 1
7/2=3 余 1
3/2=1 余 1

将余数从后往前排列，得到 1111100B，这就是转换的结果。

1.1.3 十六进制数与二进制数、十进制数之间的转换

虽然在计算机内部数的存储和运算本质上都是二进制数，但如果要说明给人看，则既不易阅读、书写，也太啰嗦。例如 252D=1111100B，这一长串数字是很难一眼看出它的值的。十进制数虽易理解，但它与二进制数之间的对应关系却不直观。一种更常用的数便是十六进制数，因为它的 1 位可以对应于二进制数的 4 位，非常简炼。252D=1111100B=FCH，FCH 就很简炼。更重要的是，这种表示方法与计算机中的运算器及存储器的对应关系也很直接。

计算机中存储信息的最小单位是“位”或“比特(bit)”，这与二进制的 1 位数相对应。但计算机中的字符一般是由 8 位二进制数组成的一个“字节(byte)”来表示，计算机存储器的基本单位是字节。计算机的字长一般都为字节的倍数，如 8 位、16 位、32 位、64 位等。无论是字节还是字，都刚好是 16 进制数的整数倍，如一个字节可以用两个十六进制位来表示。例如，字节 11101110B 的十六进制数为 EEH。

1. 二进制数转换成十六进制数

二进制数转换成十六进制数，只需从低位往高位，每 4 位一组，然后找出对应的十六进制数即可。例如：

1001111B=01001111B=4FH

2. 十六进制数转换成二进制数

将十六进制数转换成二进制数时，采用上面转换的逆过程，即一个十六进制数位对应于 4 个二进制数位。例如：

66FH=011001101111B=11001101111B

上例中的 6,6,F 分别对应的 4 位二进制数为 0110,0110,1111,组织到一起为 011001101111B。也可去掉高位 0,得 11001101111B。

实际运算中更常见的是十六进制数与十进制数的相互转换。

3. 十六进制数转换成十进制数

将各位十六进制数与对应的权值相乘,其结果相加,即为对应的十进制数。例如:

$$\begin{aligned}FFFFH &= 15\times16^3+15\times16^2+15\times16^1+15\times16^0\\ &= 15\times4096+15\times256+15\times16+15\times1\\ &= 65535D\end{aligned}$$

4. 十进制数转换成十六进制数

将十进制数转换成十六进制的方法主要有除法和降幂法。

· **降幂法**。采用降幂法来将十进制数转换成十六进制数的方法是:先根据要转换的十进制数值,找出小于或等于该数的最接近的十六进制权值,然后找出该数中包含此数值的倍数,用原数减去此倍数与权值的乘积,记下倍数。差值作为新的十进制数重复上述运算,直到差值为零。例如将 65534D 转换成十六进制数结果如下:

$65534D - 15\times16^3 = 65534 - 15\times4096 = 4094$ ——倍数为 15
$4094 - 15\times16^2 = 4094 - 15\times256 = 254$ ——倍数为 15
$254 - 15\times16^1 = 254 - 15\times16 = 14$ ——倍数为 15
$14 - 14\times16^0 = 14 - 14 = 0$ ——倍数为 14

将上式中的各阶段倍数 15,15,15,14 按顺序组成对应的十六进制数码,即为 FFFEH。

降幂法的困难在于难于找出第一个最接近的十六进制数位的权值。

· **除法**。把要转换的十进制数(整数)不断除以 16,并记下余数,直到商为 0 时为止。最后将所得的余数倒序排列,即为所求的十六进制数。例如将 65534D 转换成十六进制数:

65534D —— 65534/16 = 4095 余 14
4095/16 = 225 余 15
255/16 = 15 余 15
15/16 = 0 余 15

将余数从后往前排列,即得 FFFEH。

采用除法转换时必须先缓存余数。

实际编程运算时,降幂法和除法都常被采用。

1.2　二进制数与十六进制数的运算

1.2.1　二进制数和十六进制数的算术运算

二进制数算术运算规则如下：

加法规则：	乘法规则：
0+0=0	0×0=0
0+1=1	0×1=1
1+0=1	1×0=0
1+1=0(进位 1)	1×1=1

十六进制数的运算与十进制数的运算相似，只不过要记住权值为 16 的 n 次方，应“逢 16 进 1”，而不是“逢 10 进 1”。例如：

9999H + 9999H = 13332H

$$\begin{array}{r} 9999\text{H} \\ +\ 9999\text{H} \\ \hline 13332\text{H} \end{array}$$

如果不是“逢 16 进 1”，按照十进制的“逢 10 进 1”，就会变成“79998”。

又如：1333H − 999H = 99AH

$$\begin{array}{r} 1333\text{H} \\ -\ 999\text{H} \\ \hline 99A\text{H} \end{array}$$

减法时，如果某位数不够减，会向前借位，借来的位是“以 1 当 16”，而不是“以 1 当 10”。因此，上面的结果不应是 334。

前面介绍过，计算机内部的数都是二进制的。带符号的二进制数在表示时，常采用“补码”，二进制数的减法运算，实质上是被减数与减数的补码相加。例如：

−117D=−0000 0000 0111 0101B

它的补码为：

1111 1111 1000 1011

于是：118D−117D=118D+(−117D)

用补码运算时，过程如下：

$$\begin{array}{r} 0000\ 0000\ 0111\ 0110 \\ +\ 1111\ 1111\ 1000\ 1011 \\ \hline 1\ 0000\ 0000\ 0000\ 0001 \end{array}$$

去掉最高位的进位，结果为 0000000000000001B，即 1D。

二进制负数补码的基本规则是“各位变反，末位再加 1”。

这里只简单地介绍了补码的基本规则，详细的补码运算，包括二进制数、十六进制数的小数运算、补码运算等，限于篇幅，这里不再介绍。

1.2.2 二进制数和十六进制数的逻辑运算

逻辑运算是二进制数的"按位"运算，无进位关系。由于1个十六进制数位对应于4个二进制数位，所以十六进制数的逻辑运算，本质上是二进制数的逻辑运算。常见的逻辑运算有"与"、"或"、"非"、"异或"四种。

1."与(AND)"运算

"与"运算也称为逻辑乘，可用符号 · 或∧表示。两位 A 和 B 相与时，只有当 A 和 B 都为1时，结果才为1，否则，结果为0。即：

$0 \wedge 0 = 0$　　$0 \wedge 1 = 0$

$1 \wedge 0 = 0$　　$1 \wedge 1 = 1$

2."或(OR)"运算

"或"运算也称逻辑加，可用符号+或∨来表示。两位 A 和 B 相或时，只要 A 和 B 中的任一位为1，结果便为1；A 和 B 都为0时，结果才为0。即：

$0 \vee 0 = 0$　　$0 \vee 1 = 1$

$1 \vee 0 = 1$　　$1 \vee 1 = 1$

3."异或(XOR)"运算

"异或"运算可用符号$\veebar$表示。两位 A 和 B 相异或时，如果 A 和 B 分别为1，0或0，1，则结果为1；如果 A 和 B 都为0或都为1，则结果为0。即：

$0 \veebar 1 = 1$　　$1 \veebar 0 = 1$

$0 \veebar 0 = 0$　　$1 \veebar 1 = 0$

4."非(NOT)"运算

如变量为 A，则它的"非"用 $\overline{A}$ 表示。"非"运算的结果是该位变反。即：

$\overline{0} = 1$　　$\overline{1} = 0$

由于逻辑运算是按位操作的，所以多位操作时，每位分别运算。例如：X=00FFH，Y=5555H，这两个变量都是一个16位的二进制数，则它们的各种运算结果如下：

X = 00FFH = 0000 0000 1111 1111B

Y = 5555H = 0101 0101 0101 0101B

与：$X \wedge Y$ = 0000 0000 0101 0101B = 0055H

或：$X \vee Y$ = 0101 0101 1111 1111B = 55FFH

异或：$X \veebar Y$ = 0101 0101 1010 1010B = 55AAH

非：$\overline{Y}$ = 1010 1010 1010 1010B = AAAAH

1.3 ASCII 码和 BCD 码

二进制数或十六进制数对计算机的操作和运算十分方便，但二进制数或十六进制数的值与人们的日常生活中习惯的十进制数值之间要进行转换，很不直观。为了便于计算机直接对十进制数进行存储和运算，便产生了用二进制编码的十进制数，这就是 BCD 码(Binary Coded Decimal)。每 1 位十进制数用 4 位二进制数表示。这 4 位二进制数通常用 8421 编码方式，即用 4 位二进制数 0000B～1001B 表示十进制数 0～9，其余的 4 位二进制数 1010B～1111B(即十六进制数的 A～F)不用。

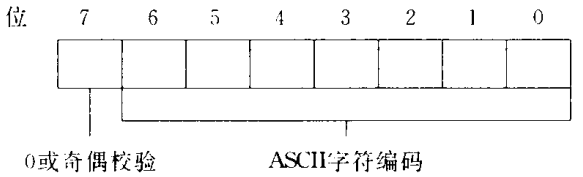
在实践中，用一个字节来表示 BCD 码时，有两种方式。第一种方式，用一个字节的高 4 位及低 4 位各表示 1 个十进制数码，称为压缩的 BCD 码；第二种方式，用一个字节的低 4 位表示 1 个十进制数，高 4 位为零，这种方式称为非压缩 BCD 码或直接称为 BCD 码。例如：

压缩 BCD 码：	0101 0101B	表示十进制数 55
非压缩 BCD 码：	00000101B	表示十进制数 5，不是 05

当一个字节中的值是 BCD 码或压缩 BCD 码时，它代表的值与对应的十六进制值不同。例如 1 字节的压缩 BCD 码 55，存放于 8 位中，为 01010101B，如果理解成十六进制数值 55H 就不对了，因为 55H＝85D。这里的根本区别是压缩 BCD 码高 4 位和低 4 位各自独立，没有权值的含义在其中。

ASCII 码

计算机在输入、存储和输出字符(也包括数字字符)时，常采用某种编码。在这种编码方式下，每一个字符都用一个字节中的某个值来代表(也有不用 1 字节的其他编码)，每个字符都用一个唯一的值来表示。其中，IBM-PC 机和现代大多数计算机，都采用 ASCII 码。ASCII 码是“美国信息交换标准码(American Standard Code for Information Interchange)”的英文首字母缩写。它用 1 个字节中的低 7 位表示字符编码，最高位(第 7 位)为 0 或用于奇偶校验位(parity bit)。



标准的 ASCII 码共有 128 个字符，分为可打印的 ASCII 码字符及非打印的 ASCII 码字符两大类。表 1.1 列出了主要的 ASCII 码。

可打印的 ASCII 码字符共有 95 个，主要的有：

26 个大写字母 A～Z	41H～5AH
26 个小写字母 a～z	61H～7A