·16位微型机丛书·

Z8000程序设计

白晔泰 乐山 译
云梯 达理
李三立 校

中国电子学会
湖南电子学会 《微处理机与微系统》编辑部

Z 8000 程 序 设 计

编　辑　《微处理机与微系统》编辑部
出　版　湖 南 省 电 子 学 会
印　刷　长 沙 潇 湘 印 刷 厂
发　行　湖 南 省 电 子 研 究 所

责 任 编 辑　谭桥庆　贺光辉

2.60

说　　明

本书译自美国SYBEX出版公司1980年出版的《Programming the Z8000》一书。书中既介绍了一般程序设计知识，又说明了Z8000的具体编程技术，不但是开发应用16位微型机（尤其是Z8000）的科技人员必备的工具书，也可作为微型机程序设计参考书和入门书，供大专院校有关专业的师生和从事实际微机应用工作的广大工程技术人员阅读参考。

为避免重复，省去了附录B、C、D及E。附录B为ASCII 字符集，同图1.15。附录C是各指令段的编码，其中使用方式／寄存器的寻址方式的编码，同图4.16；条件码，同图4.12；指令中寄存器字段的解释，同图5.6；指令中使用的地址格式，同图5.2。附录D是控制寄存器格式，其中标志控制字（FCW）中位的分配，同图2.6；REFRESH寄存器中字段的配置，同图2.10；PSAP和NSP的格式，参见图2.3，注意PSAP 偏移量的低8位永远为0。附录E是程序状态区的分配，同图2.9。

出　版　消　息

处在信息时代　　应知局部网络

《计算机局部网络技术》即将出版发行

该书译自美国Academic Press出版公司1981年出版的《Local Computer Network Technologies》一书(作者Carl Tropper)。这是系统介绍计算机局部网络技术的一本很好的参考书。它好就好在：**既全面综述了计算机局部网络技术的发展演变和基本定义，又重点介绍了环形网和总线网这两种最主要的局网结构的技术特性。**

全书共分三章。第一章说明了计算机局部网络技术的发展起因及演变过程，论述了它的初步定义、分类及基本组成。第二章和第三章分别详细介绍环形网和总线网。这两章均从协议讲起，叙述了各种协议的性能模型，比较了各种不同协议的性能，最后对各种模型给予了适当评价。在这两章中还相应介绍了一些典型的局部网络实例。

国外计算机局部网络技术的发展十分迅速。当前，国内对局部网络的研究也越来越重视，正在大力开发和推广应用计算机局部网络。**该书对从事和即将从事这一工作的科技人员以及大学中专的师生均有参考价值，也适合作训练班参考教材。**我们处在信息时代，面临着一场新的技术革命。这场新的技术革命最突出的特点和影响之一，就是实现事务管理自动化，极大提高办公室管理人员的工作效率，就象前一次技术革命大大提高工厂工人的生产效率一样。而这种办公自动化只不过是计算机局部网络所具有的内在能力的一个方面，它还将进一步推动信息网络化、过程自动化、家庭信息化……。**处在信息时代，应知局部网络。该书将系统地为你提供计算机局部网络的技术知识。**我们欢迎各行各业的科技人员、管理干部、办公人员、工人以及业余爱好者踊跃订购此书，为迎接新的技术革命的挑战作好充分准备。

该书约16万字，定价1.50元（包括邮费0.10元在内），预计8月底出版。**凡本刊《微处理机与微系统》的订户，我们将通过第三期（7月初发行）发送该书订单。**非本刊订户，请先来函索取订单，再按订单要求汇寄订款。

《微处理机与微系统》编译部启

地　址：湖南省长沙市朝阳二村

目　录

序　言

这是一本完整的论述Z8000微处理器编程的书。

1.它详细叙述了Z8000的结构及功能，并说明了与它的支持器件系列间的相互关系。

2.以Z8000为具体例子介绍了机器语言的编程。

3.它给出了很多Z8000的程序实例，这些程序都是围绕一些中心问题而组织的，例如与终端操作员的交互作用，存贮器管理，可共享的程序与分时问题。给出这些程序主要是为了说明程序设计的技巧与原理。

4.它说明怎样用机器语言编程才能获得简单、清晰和模块化的程序以及怎样实现软件工程上的其它重要原则。

本书的内容适用于任何乐于使用Z8000的读者，不同的读者阅读本书时，侧重点可以不同。希望学习机器语言程序设计的读者可以首先学习第一章到第五章、第六章的前半部分及第七章的一小部分的内容；第二、第四、第五、第七各章介绍Z8000及其配套器件，对编程不太感兴趣的设计工程师，可以阅读这些章节，以获得有关机器的说明以及评述机器优缺点方面的内容；希望继续深入学习的读者，将在本书的其余部分看到很多例子，但是有些例子对初学者而言可能理解起来有些困难；有经验的编程员（尤其是熟悉PDP-11计算机一类的CPU系列的编程员）可以跳过第一章和第三章，但是他们应该详读第六章的后半部分、第八章和第九章。

本书首先讨论了基本的编程方法以及与编程有关的硬件说明，然后列举了编程的例子。第一章是编程入门，内容有算法、流程图以及在计算机存贮器中表示信息的各种方法。

第二章概述Z8000的体系结构。由于Z8000是一个先进而复杂的CPU，故本章不得不包含很多较深的概念，其中有些内容对于以前从未学过计算机的人而言是有些困难的。如果对这些概念的细节或意义感到难以理解的话，最好是先浏览一下这些内容，然后在读完后述各章之后再回头阅读那些较难理解的内容。

第三章介绍了Z8000程序的整个开发过程，从最初的算法到主程序及其两个子程序的最终编码。作为例子给出的算法是文本加密算法，办法是用文本字符与密钥字符进行“异或”操作。以此为纲阐明了某些概念、原理和技巧。

第四章介绍了Z8000指令系统。为了便于理解，把指令分为八大类，这样就可逐类加以说明。重要的是，要理解和记住指令，而不是指令类。本章还用图解来说明指令执行与数据通路的情况，而图解本身只是一种重要工具而已。第四章末是指令的基本参考资料。Z8000的105个助记符分为45组，分组的原则是把操作及指令格式基本相同的指令作为一组，把它们作为一个单元处理起来比分开处理更为清楚。第四章最后的45种指令的说明，以两个索引开头：其一是按字母顺序排列的105个基本助记符表，每个助记符都标有相应指令说明的索引；另一个则是按数值排列的64个“操作码”表，每个操作码都标有相应助记符及其指令

说明的索引。因为指令说明及其索引包括在参考手册内，故第四章以相当长的篇幅叙述了它们的使用方法，其中包括用手工汇编和反汇编Z8000程序的例子。

第五章通过介绍Z8000的多种寻址方式，补充说明了CPU的操作和指令。本章篇幅不大，介绍了主要的寻址方式（寄存器寻址、寄存器间接寻址、立即寻址、直接寻址和变址寻址），两种只用于装入指令的寻址方式（基址寻址和基址变址寻址）以及隐含相对寻址方式。此外还举例说明了指令中立即自变量和地址的格式，并对寄存器字段的编码作了解释。

第六章介绍输入／输出技术。讨论了I／O寻址，还介绍了电平和脉冲发生技术、等待循环和同步位串行数据传送。本章以电传机的子程序为例说明了位串行传送的原理，同时讨论了半双工和全双工终端操作以及并行异步数据的传送。文中以七段显示为例详细叙述了它的工作情况。最后讨论了查询I／O调度方式和中断方式，同时给出了一个终端设备的I／O中断程序的例子。这些例子涉及很多编程技巧，可能在初次阅读时感到比较难以理解。

第七章论及Z8000的外围部件，例如存贮器管理单元以及并行和串行I／O电路。作者编写本书时，使用的是他所能获得的最新资料，不过在本书将来的版本中会介绍得更详细些。本章还讨论了Z8000的某些设计之外的性能，例如以REFRESH（刷新）寄存器为基础构成时钟等。

第八章包括了很多例子，这些例子说明了Z8000如何编程以及构成有用的工具是多么简单。在本章中详细介绍的这种工具之一是终端处理程序，它通过与终端（CRT或TTY）操作员的对话而便利了编程工作。本章使用了第六章中介绍的中断程序，并给出了提出问题、译解答案、光标控制及屏幕显示的基本程序。在本章中还介绍了很多其它的实用程序，包括环形及行缓冲器处理程序、相关搜索翻译程序以及位表的查阅和扫描程序。

第九章通过介绍一个简单的分时系统来说明Z8000体系结构的能力，该分时系统的用户可以使用具有独立堆栈的共用程序。此外还讨论了堆栈管理技术，并详细介绍了陷阱处理及系统初始化的方法。

最后，在第十章叙述了程序开发的环境，讨论了编辑程序、汇编程序、装入程序和调试程序的各种版本和设计问题，还介绍了两种开发硬件，即开发系统与单板计算机。

在第十章后面有几个附录。附录A给出了书中的习题答案，附录B、C、D和E复制了本书其它章节中的图表和图，因此可作为参考。附录F概述了扩展结构（EPA），它允许其外接器件（如浮点处理器）受在线指令控制。这一特性在本书第一次印刷后才由制造厂公布。该附录解释了如何用标志和控制字的第13位、操作码0E和0F（16进制）以及设定为A、B及E（16进制）的状态线（ST_3-ST_0）来支持这一特性。

本书包含了很多实用的语句与程序示例。作者尽力保证内容正确，但是错误在所难免，欢迎批评指正，以便在再版时修正。

第一章　基本概念

本章讨论算法、流程图和信息表示法。了解这些内容的读者只要浏览一下即可。

§1·1　什么是程序设计

程序设计是一种需要技能的技巧——象做木工活或练书法一样，必须进行基本功的训练并经过多种多样的实践。我们有加工用的原材料和待完成的任务，这种“原材料”就是计算机系统（Z8000）的指令、存贮器和其它附属设备，至于任务在以后会逐渐被我们深入了解，目前暂且先介绍一下解题的过程。

§1·2　算　法

每个学习计算机的人首先必须懂得，计算机只能按人的意志行事。因此，人们一定要清清楚楚地告诉计算机应该做什么，否则就可能陷入困境，人的意志无法让计算机理解。要使计算机能理解人们的意志，先得从算法讲起。所谓算法就是按人们给定的顺序执行的一系列编上号的指令（称作步骤）。为了使算法实用，需要完成一些步骤，例如，“如果满足某种条件就向前（或向后）跳到某一步”。这就使事情复杂化了。我们必须保证有一种转出的办法，以免无休止地循环某一步骤。

图1.1说明了两个数相乘的常用方法。图1.2是其对应的算法，由此得到一个关于我们如何去告诉计算机进行工作的概念。每当我们有一新的问题需要解决，就得开发某种算法，并将它告诉计算机。

让我们以图1.1为例看一看图1.2算法的实际运算情况。第一步，把112写在12573的下面，并使112对准573，然后在这两数的下面划一条线。

```
  1 2 5 7 3   ……第1步
    1 1 2
  ---------
  2 5 1 4 6   ……第4步，第一次（n=1）
  1 2 5 7 3   ……第4步，第二次（n=2）
1 2 5 7 3     ……第4步，第三次（n=3）
  ---------
1 4 0 8 1 7 6 ……第7步
```

图1·1　两个数的相乘

第二步称为计算器的初始化，在图中未标明。这一步只是记住正在处理的112这个数中3个数字的一种手段。具体乘法从右到左进行，因此n＝1对应处理2这个数；n＝2对应于中间的1；n＝3对应于第一位的1。

1. 写出两个数，一个数写在另一数的下面，并使它们末位数字对齐，在这两数的下面画一条线。
2. 将计数器置1。
3. 如果计数器的n值超过乘数的位数，则跳到第7步。
4. 第一个数和第二个数的从右边算起的第n个数字相乘，答案写在横线下面，其最右边的数字与所乘的数字对齐。
5. 计算器加1（即用n＋1代替n）。
6. 返回到第3步。
7. 将各次第4步得到的数加在一起，这就是所要求的结果。

图1·2　乘法算法

第3步称为条件测试，把计数器内容n（刚预置成1）与3（112的位数）进行比较，显然1并不大于3，因此不跳到第7步去。但是通过第5及第6步，我们将使n增加到2，然后增加到3，再增加到4，并且每次都返回去将其值与3进行相比。当n增加到4时，其值超过3，这时就进入第7步。第2、第3、第5、第6步合在一起相当于说：对下面的数的每位数字执行一次步骤4，即对112而言是3次。

第2到第6步构成所谓“循环”。第4步是循环内反复执行的操作，而第3、第5、第6步是机械似的处理过程，用以使第4步的循环次数恰好等于所要求的次数。

第4步是循环的核心部分，它指明12573应与112的某位数字相乘，并记下答案，答案的末位数字与相乘的那位数字对齐。用这种方法使末位对齐是因为112数中两个“1”分别表示了100和10，使答案对位就可省去这些数字后面所写的零。实际上，图1.1横线下面所得的和可写成：

```
  25146
 125730
1257300
1408176
```

如果想用112和12573相乘，不必给出七步命令，只需询问“12573乘以112是多少？”如果读者学过（并记得）这个算法，这个简化了的命令就可得出所需的结果。

与此类似，一旦我们把算法输入计算机，我们只要用一简单命令去告诉计算机处理这种情况就行了。这就是所谓的子程序。免去重复乃是子程序成为程序员重要工具的原因之一。

练习1： 在图1.2的乘法算法中，并非所有的步骤都重复同样的次数。在使用算法解12573×112的题目时，第3步究竟要执行几次？哪些步骤仅仅只执行一次？如果112在上面，12573在下面，两数相乘的情况又怎样呢？

上述算法是从下边的数的右边开始并向左进行的。你能设计一个从左边开始并向右进行的算法吗？你如何使数相应地对齐？用你设计的方法试解9536×121，假设你只希望得一近似答案，从左边开始有何优点？

§1.3 流程图

另一表示算法的方法是流程图。流程图是算法的一种图形表示法。算法中的每一步画在框中，引线及箭头表示各步的次序。将“如果满足这样或那样的条件就跳到(或返回)某一步”这种形式的步骤画成菱形，这样在流程图中就可从菱形框中引出不止一根引线。乘法算法的流程图示于图1.3,流程图中的框框及引线都用与它们对应的步骤数作标记。究竟用流程图的方法还是用叙述的方法去表示算法这都无关紧要，关键是在写程序前要用一种或另一种方法来描述算法。

练习 2：画出“打开你的前门”的流程图。假设门锁着，用一种办法开门；如果没有锁门，就用另一种办法开门。

§1.4 信息表示法

计算机有三个基本组成部分：执行各种操作的中央处理器(CPU)；被存贮的程序，它只是一种以CPU能够执行的操作和判定过程所表示的算法；存放CPU操作数据的存贮器。

1. 十六进制表示法

在讨论 CPU 和存贮程序之前，我们要研究一下在存贮器中数据的各种表示方法。计算机的存贮器都由数位即二进制位建立。二进制位的两个状态总是用 0 和 1 来表示。象一个小灯泡那样，可以规定关是0，开是 1 。在Z8000中，主要位组是十六位的字,字又分为 8 位的字节。每个字节由两个四位数字组成。反过来，有时把两个字组合成32位的长字。Z8000CPU偶而还能识别64位字长的结构形式。这些位组里各位（灯）的实际状态称作位的组合格式或字节和字的值（见图1.4)。

我们用具有不同意义的各种位组合格式来表示存贮器里的数据。有时，同一个位组合格式在不同的时刻表示不同的含义。它可表示为一个数，一个文本字符，或别的什么符号。实际上，机器语言程序（以及许多“高级”语言程序）出错的一个根源就在于，把存贮器中一种类型的数据当作另一类数据来处理。

由于任何事情都取决于对位组合格式的解释，所以我们需要用某些方法去表示它们。常用的方法是十六进制（以16为基数）表示法。图 1.5 给出了四位二进制数的十六种可能组合的命名，从中可见它们是按一定次序排列而不是任意指定的。位组合格式按二进制数进行编排和解释。

十六进制和二进制是两种数制，在常用的十进制数制中，任意一个数都可精确地写成如下形式：

$$a_0 + a_1 \times 10 + a_2 \times 100 + a_3 \times 1000 + \cdots$$

即

$$a_0 + a_1 \times 10 + a_2 \times 10^2 + a_3 \times 10^3 + \cdots$$

式中a_0，a_1，a_2…是0到 9 的整数。例如，1980是下式的缩写：

$$0 + 8 \times 10 + 9 \times 100 + 1 \times 1000$$

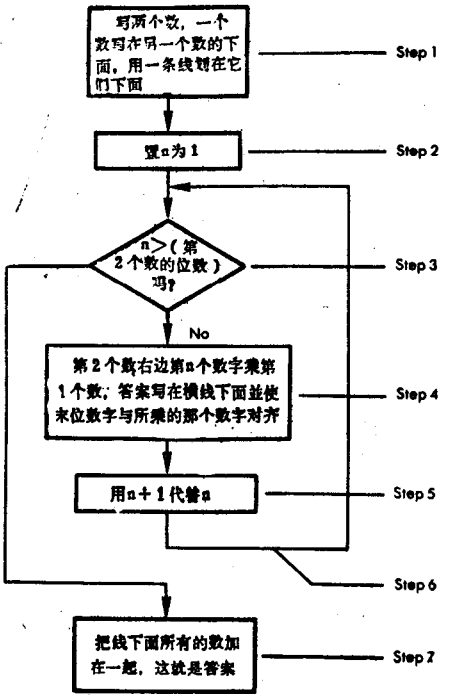

图1·3　乘法算法的流程图

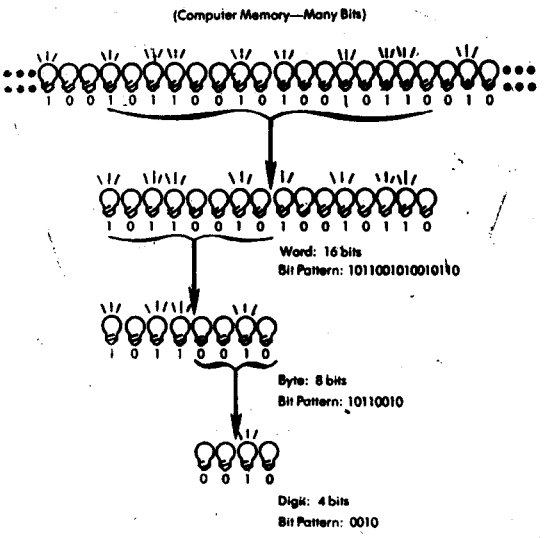

图1·4　位的分组及位组合格式

格　式	名　称	格　式	名　称
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

图1·5　十六进制位组合格式的名称

同样，如果b是大于1的整数，则可以证明任意整数可以精确地写成下列形式：

$$C_0 + C_1 \times b + C_2 \times b^2 + C_3 \times b^3 + \cdots$$

式中C_0，C_1，…是0到b-1的整数。例如：

$$196 = 4 + 0 \times 8 + 3 \times 8^2$$

对于b＝2和b＝16，相应的数制就是二进制及十六进制。当b＝2，系数C_0，C_1，…所有的值只能为0或1；所以任何位组合格式都可看作二进制数。

当b＝16，系数C_0，C_1……的取值可为0到15。为了便于书写起见，我们用下列缩写形式：A＝10，B＝11，C＝12，D＝13，E＝14，F＝15，例如：　$196 = 4 + C \times 16$

由十进制表示法类推，则可把基数为b的表达式

$$C_0 + C_1 \times b + C_2 \times b^2 + C_3 \times b^3 + \cdots$$

写成　　$\cdots C_3 C_2 C_1 C_0$

例如，用十六进制表示则有

$$196 = C4$$

由于有些十六进制数看起来象十进制数，为了不使两者混淆，可用基数作为下标。例如：

$$196_{10} = 304_8 = C4_{16}$$

练习 3：用二进制数表示 7 和196。用二进制和十六进制表示196时，二者的关系是什么？

现在，图1.5中位组合格式的命名就清楚了：每四位格式的名称就是十六 进 制数，它与由它表示的四位二进制数相等。例如：

$$1101 = 1 + 0\times 2 + 1\times 2^2 + 1\times 2^3 = 13 = D_{16}$$

而且，这个十六进制数的命名法可以扩展到8位、16位或者更长的位数，只要将位组合格式分成四位一组的形式即可（如果遇到位组合格式中的位数不是 4 的倍数，则从右边算起），并且按每 4 位为一组给以十六进制的命名，这样就可得出一个数的十六进制的叫法。这个十六进制数就表示与原来位组合格式对应的二进制数相同的数。图1.6所示即为其例子。注意，在该图的右边是如何把二进制的数16，32，64及128这些项归在一起求值，然后分解出16这个因子。显然，这种方式能扩展到更长的位数，从而证明上面所说的情况是正确的。

练习 4：用图1.6的方法说明 $11000100_2 = 304_8$

如果我们研究一下基数为b、位数为n的数，即

$$C_0 + C_1\times b + C_2\times b^2 + \cdots + C_{n-1}\times b^{n-1}$$

那末我们就能表示 0 到 $b^n - 1$ 间的任意数。例如，如果我们限于三位十进制数，那末我们就可以表示从0到999的任意数（因为 $999 = 10^3 - 1$）。因为区分一组中某项的方法是对每一项给以一个数，而从0到 b^{n-1} 有 b^n 个数，所以另一种表示方法就是：基数b的 n 位数能区分 b_n 个不同的项。具体地说可以得出下列重要的法则：

法则：（a）n 位二进制数能区分 2^n 个不同的项。

（b）n 位十六进制数能区分 16^n 个不同的项。

例如，两位的二进制数有 $2^2 = 4$ 种位组合格式，即00，01，10，11。我们可以通过指定每种格式对应于一项，从而用该数能区分四项内容（见图1.7）。

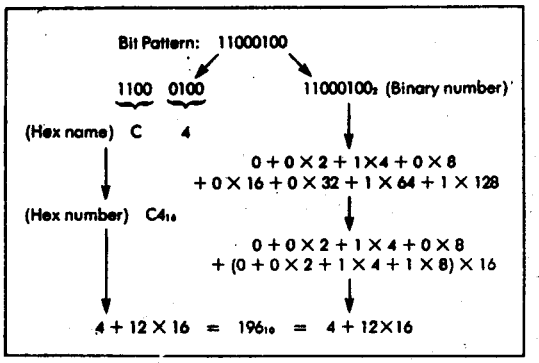

图1·6　二进制与十六进制间的关系

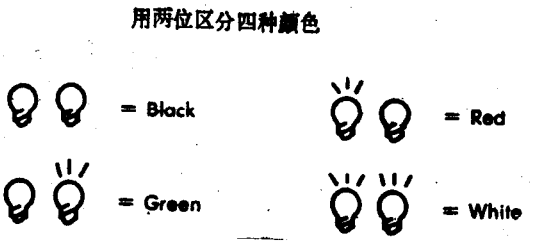

图1·7　用n位二进制数区分 2^n 项内容

练习 5：用 8 位二进制数能区分多少不同的项？需要用几位来区分27个不同的项？如果要设计一微处理器，它需要有12种不同的状态来与外界通信，那么需要用多少条状态线（位）来规定它们每种状态的不同的位组合格式？用 4 位十六进制能表示多少个不同的数？用 8 位十六进制数呢？

由于在计算机内部，数是用十六进制来表示的，而在日常生活中，数通常是用十进制表示的，所以我们应该知道十六进制和十进制间相互转换的方法。从十六进制到十进制的转换用十六进制表示法的定义就能很容易地完成。例如：

$$C4_{16} = 12 \times 16 + 4 = 192 + 4 = 196$$

图1.8表示了从十进制转换到十六进制的算法，它只用了十进制的算术运算。此算法看上去复杂，实际上相当简单。图1.9表示了如何用此算法来转换十进制的数299。在算法的每一步都除以16，余数成为下一个16进制的数（从右到左），除得的商作为后面一步的起始点。最后一个式子说明这样做的原因。十六进制的数$h_3h_2h_1h_0$可以写成：

$$(((0 \times 16 + h_3) \times 16 + h_2) \times 16 + h_1) \times 16 + h_0$$

假定十进制数是$d_nd_{n-1}\cdots\cdots d_0$。我们希望找出系数$h_m$，$\cdots h_0$，使

$$(h_m\cdots h_0)_{16} = (d_nd_{n-1}\cdots d_0)_{10}$$

1. 将计数器K的值置为0，令$N = d_nd_{n-1}\cdots d_0$。
2. N值除以16_{10}，设h_k为余数而N的新值为商
3. 如果$N = 0$，则停止；$(h_k\cdots h_0)_{16}$是所求的表示式。否则，K值增加1并进入第2步。

图1·8　十进制转换到十六进制的算法

用16除，得余数h_0和商　$(((0 \times 16 + h_3) \times 16 + h_2) \times 16 + h_1$
再用16除，得余数h_1和商　$(0 \times 16 + h_3) \times 16 + h_2$
再接着用16除，得余数h_2和商　$0 \times 16 + h_3$
最后一次除法得余数h_3。同样的方法可以扩展到任意位数的十六进制表示法。

练习6：试用图1.8的算法来转换十进制数196

如果我们要对两个十六进制的数进行加、减、乘、除，可以先把它们转换成十进数进行运算，然后再反转换为十六进制数。显然，更直接的方法是用十六进制去完成算术运算。图1.10为一位十六进制数的加法和乘法表。多位数的十六进制加法、乘法完全与十进制运算相似，即如果两个一位数的和或积为二位数，则高位数就进位到左边相邻那一位去。对于十六进制的减法和除法也是从加法及乘法推导出来的，与十进制运算完全一样。

例如，假设我们希望用26_{16}乘以$7A_{16}$。图1.11说明了这个乘法及对应的十进制乘法。开始，我们用6乘7A。先用6乘A，在乘法表里找到A行与6列相交的点3C，记下C及进位3，然后找到$7 \times 6 = 2A$。我们加上进位3而得出2D。写下得出的结果为2DC。按类似步骤求得$7A \times 2 = F4$；我们把4写在26中的2下面。最后对横线下面的数进行相加，F4实际上为F40；$C + 0 = C$；$D + 4 = 1$，进位为1；$2 + F +$进位$1 = 12$。这样得出结果为121C。

练习7：$B + B$，$1A + B1$和A^2各为多少？求$2B5\sqrt{1B93F}$的值

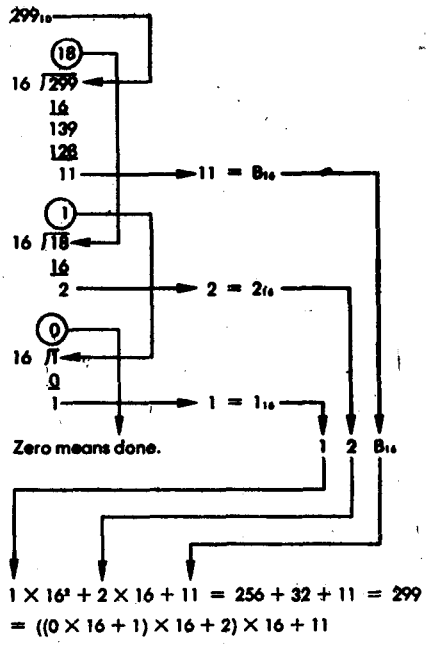

图1·9：299_{10}到$12B_{16}$的转换

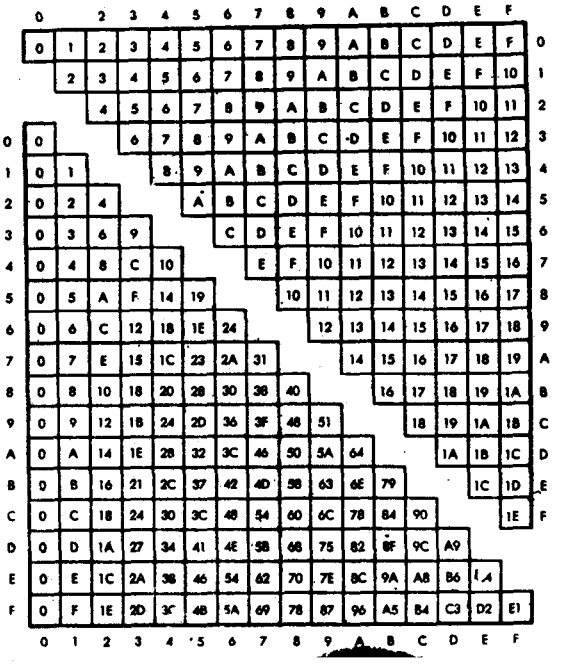

图1·10　十六进制[illegible]的加法[illegible]和乘法表

2.：二进制补码表示法

到目前为止我们已经谈到了可用16位的字来表示0到65535（因为2^{16}=65[illegible]）的数。但是我们还希望表示负数，这需要用巧妙的办法来实现。

处理负数最简单的方法是用一位表示符号。例如，我们用16位字长的最左面一位作为符号位，而其余15位表示从0到32767的数值。这称为符号数值表示法。这在Z8000计算机中是不用的（在大多数其它计算机中也不用），因为在电路上实现它比实际用的其它方法要困难。另外，它有两种表示零的方法：－0及＋0，这也是它的缺点。

十六进制	十进制
7A	119
26	34
2DC	476
F4	357
121C	4046

图1·11　十六进制的乘法

实际所用的方案称为负数的二进制补码表示法。二进制补码“借用”额外的一位，然后再抛弃这个位。因此在16位的二进制补码表示法中，－X表示为$2^{16}-X$，大于$2^{15}-1$的数被看作负数。此方案的最大优点是可以把用二进制补码表示的数都看成是正数那样来进行相加，并可得出正确的答案。由于

$$X+(2^{16}-X)=2^{16}=10000_{16}$$

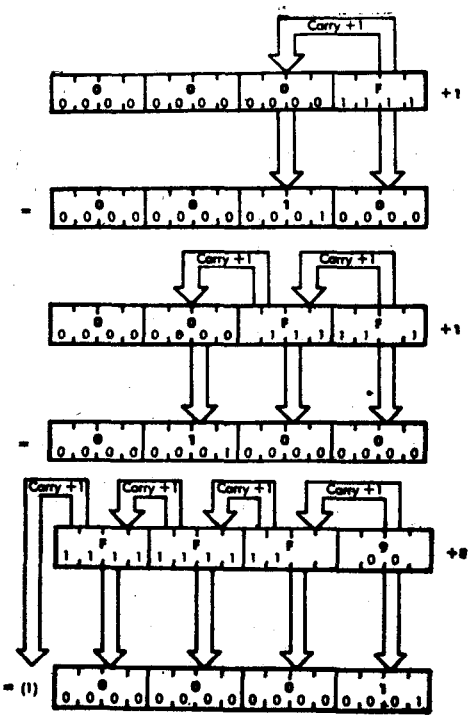

图1·12 16位字作为十六进制的里程表

根据需要如果我们只看 10000_{16} 的末16位，则为 0000_{16}，所以

$$X+(-X)=0$$

是所需的结果。这就正象汽车的五位数字里程表那样：99999 英里＋1英里＝0 英里。即在此里程表上有：

$$-1=100000-1=99999$$

16位的字就象二进制的里程表（或十六进制的里程表，参见图1.12）。

以下是Z8000 中使用的16位的二进制补码表示法的一些例子：

－1＝FFFF；－20＝FFE0；－8000＝8000

在第二章里我们要讨论寄存器与条件位。当 Z8000作加法或减法运算时，就会设置条件位使编程员知道发生了什么情况。条件位之一是C 位，它是借来进行二进制补码运算时的第17位。两个16位的数相加时，第17位称为进位位。Z8000 中的最后16位为加法的结果，但是，Z8000 也将第17位保存在C位中，供程序员需要判断时用。

练习8：用二进制补码如何表示－5？如何表示－7FFF？用－1A1的二进制补码形式去解FF7减1A1，再求FF7加1A1。

－5加－1等于多少？－7FFE加－6又是多少？

然而，在作减法运算时，C 位正好以相反的方法进行处理。如果减法时需要借位，则C位置数。如果不需要借位，则C位清零。例如，对于（4－5），C置数；而对（5－4），C不置数。因此，加法的情况与减法的情况相反，即（4－5）和4＋(－5)时C位的值不同。

练习 9：求下列各次计算的结果及C位的值：6－4；BB－BC；FFFF＋3；FFFF＋FFFF。

此练习说明了用二进制补码进行算术运算会出现的问题：算术溢出。例如，

$$7FFE+6=8004=-7FFC$$

用另一种方法（十进制方法），则

$$32766+6=-32764$$

通常，当两个正数的和是负值或两个负数的和是正值时产生算术溢出。在Z8000中发生溢出时，V位置数，否则V位清零。这也可以从前面介绍的5 位数的里程表中看出。从50000到99999中所读到的所有的数表示负数。因此，当我们把正数5 加到正数49998时得到50003，