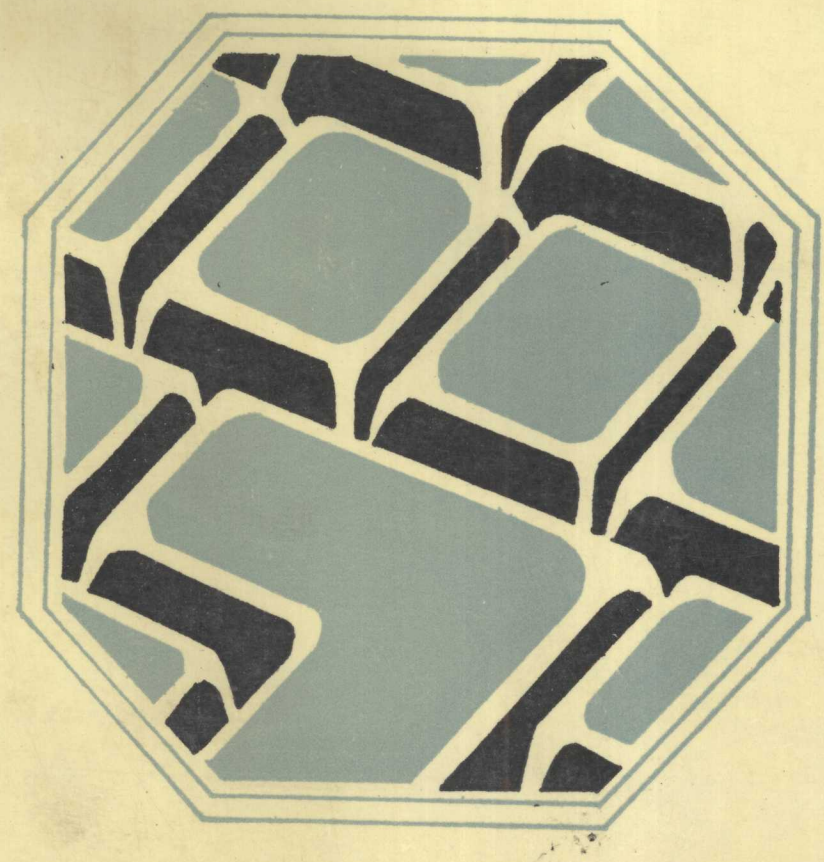
《8 0 3 8 6》

程序设计

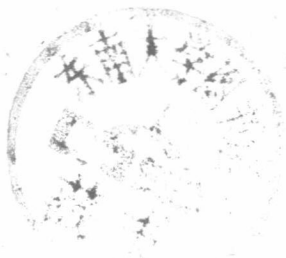
李旭东　李思东等



中国科学院计算所公司

80386程序设计

李旭东　李思东　等译



中国科学院计算所公司

一九八八、九

前　言

本书旨在为用汇编语言编写程序的人员完整详细地介绍80386，80386是得到广泛应用的Intel 86微型计算机系列的最新型号。全书重点介绍了该芯片的32位特性，该芯片完全与8086和80286兼容，我们在第九章总结了这些特性。除了完整详细地介绍80386外，我们还介绍了80387——80386的数字协处理器；介绍的方式是与80386结合起来讲，而不是象其它书一样为80387单独辟一章或放在一个附录中。

这种芯片本身的开发已经花费了多年时间，我们很高兴能在本书中介绍芯片设计者是如何编写程序和使用80386的，我们力求达到只有芯片设计者所能做到的，使全书表达准确和具有权威性。

需要回答的一个重要问题是：为什么你应该立即阅读本书？为什么你对80386的了解和用它来进行程序设计，将对你的程序设计生涯今后的十年或许今后二十年大有裨益？答案就是在Intel 86系列中已投入的不断积累起来巨大的投资。为了设计和使用计算机，必须连续不断地投资。这些投资是通过设计计算机（IBM PC，PC/AT，PSR）操作系统（unix，MS—DOS）、程序设计语言（C，FORTRAN）、应用程序（Lotus 1—2—3，Multi Mate，dBASEⅢ）和附加硬件（图形、附加磁盘、网络连接、添加存储器）实现的。这种投资也包括你为自己编写的程序，当然，还有你学习所花费的时间。因此，计算机系列，象Intel 86，不断发展，而且一代计算机与其下一代之间具有兼容性；这种兼容性使得可以继续使用已经在一个计算机系列中开发的所有上述资源并考虑这些资源的影响。

我们假定你已经具备了计算机操作的基本理论，同时还假定你在阅读书本之前，已经接触过其它的汇编语言，因此，我们有意避免这些介绍性的标题，并建议在这些方面缺乏经验的读者先去读一本介绍性的书。本书大致分为三部分第1章至第4章介绍应用程序员所需了解的80386；应用程序员不必阅读完全书，而仅需读完这几章。第5章至第7章介绍了操作系统序程员所了解的80386，这几章相对前4章而言，指导更少，而且安排的有关详细使用操作系统的参考资料。第5章至第7章是操作系统程序员需要阅读的章节。第8章和第9章介绍了调试和80386与8086及80286的兼容性。下面详细地介绍一下各章。

第一章简单地介绍了一下8086系列处理器和存贮器组织、数字表示等。重点介绍了80386和80387支持的数据类型。

第二章首先介绍了80386的内部机器状态、通用寄存器、控制寄存器和段寄存器，接下来讲述存贮器的寻址方式、指令编码和I/O空间寻址，最后介绍了80387的内部机器状态、通用寄存器和控制寄存器。

第三章是本书最长的一章，介绍了80386和80387的每一条指令。指令介绍分为四部分：整数指令、多段指令、用于编写操作系统的指令和浮点运算指令。

第四章介绍了几个应用程序员运用指令和机器状态的例子，为应用程序员给80386和80387作了一个总结。

第五章介绍存贮器管理、保护和80386的多任务功能。将多任务功能所涉及用到的几个

寄存器和系统段在这里作了一个介绍，而没有放在第二章中和其它寄存器一起讲述。本章描述了所有分段的精确语义性、存贮器存取、控制转换和任务开关(task—switching)操作。

第六章介绍80386中断和异常，并介绍了它们的处理方式，这包括中断优先权、怎样屏蔽它们和在中断处理过程中控制转换的细节。如同第五章，本章对中断和异常处理的细节作了权威性的描述。最后介绍了80387的异常原因和处理方法。

第七章提供了几个80386的操作系统功能的例子。这些例子演示了第五、六章中讨论过的分段，分页和异常手段，以及第三章中的操作系统指令和多段指令。

第八章讲述了80386中特地为支持调试（debugging）所提供的手段。

第九章回顾讨论了在80386上执行16位代码，这包括真8086（real—8086）、虚拟8086（Uirtual—8086）和16位保护操作方式。

下面的参考书提供了有关80386和80387的其它一些资料。因为本书没有介绍80386的硬件方面的知识，若需要了解，建议读者阅读参考书(2，3)。

1. 80386 Programmer's Reference Manual, Intel Corporation, Order No 230985.
2. 80386 Hardware Reference Manual, Intel Corporation, Order No. 231732.
3. 80386 Data Sheet, Intel Corporation, Order No. 231630
4. 80386 Assembly Language Reference Manual, Intel Corporation Order No122332.
5. 80387 Data Sheet, Intel Corporation, Order No. 231920

目　　录

第一章 基本概念

本章以简要介绍 Intal 公司研制的86系列历史背景开始，80386是86系列处理器最后的功能最强的成员，介绍历史之后，我们将描述 80386 所支持的数据类型。本章描述80387所支持的浮点数据类型。

第一节 Intel 微处理器的历史

第一个微处理器4004是 1971 年由 Intel 研制的。4004很快就增强到8008。这些以今天的标准来说非常普遍的器件，在那时都为新的产品，但几乎不能用以制造任何有价值的计算机。一九七四年，Intel 的第二代微处理器8080的被采用，这是第一个通用微处理器，它对微处理器工艺至关重要。一九七八年，第三代微处理8086被采用，这标志着微处理器做为真正的计算机的开始，这就是86系列的开始。

8086是 16 位处理器，8080是 8 位处理器，4004是 4 位处理器。8088——8086的小兄弟被采用于 IBM 的个人计算机（在一九八一年被采用的），这导致了个人计算机的革命。随着8086和8088在个人计算机和其他许多设计上的使用，8086体系结构那时和现在都体现重要的微处理器体系结构。

但是，86系列不能在8086上止步不前。一九八二年，Intel 采用了80186。这个元件在体系结构上不同于8086，但使用了其他几个共同的系统器件，接着在一九八二年，80286 被采用，80286是8086体系结构的超集。这意思是它既能严格地象8086一样操作，又能做更多的工作。特别是，它增加的支持多任务，意即一次能执行多个应用程序或任务的能力。多任务要求 80286 对每一个任务和每一任务的。存贮区域进行保护。80186 和 80286 都是 16 位元件。

最后，80386 于一九八五年被采用。80386比之80286和8086提供了两个主要的和许多次要的增加功能。这些增强功能总结在附录A中。两个主要的增强功能是 32 位操作和数据类型，以及除了具有所有86系列成员所具有的分段内存管理技术之外，还拥有分页内存管理技术。分页和分段技术将在第 5 、 6 、 7 章作详细讨论。另外，80386扩充了 80286 多任务能力，80386允许8086，80286和80386的任务和操作系统的同时运行。

最后来讨论数字协处理器。数字协处理器与86系列的每一代主要成员相关。主要的86系列元件是8086，80286和80386，而与他们相关的数字协处理器分别是8087，80287和80387。这些协处理器与处理器紧密地结合以使计算机体系结构支持浮点运算和数据类型。这些浮点元件间的微小区别列在附录B。

一、与8086、80286的兼容性

86系列的每一代都保持与先前每一代成员相兼容。这样，80386能够执行在8086和80286

上运行的任何程序。第9章将讲解如何在80386上运行8086和80286的程序。但是，除开向后兼容性的问题不谈，这本书将集中讨论可用 386 的全部32位装置，以及讲解如何将386作为32位机来进行程序设计。这本书不讨论如何设计8086或 80286 的程序。（如果你要设计8086或80286的程序的话，你应阅读别的书！）

然而，既然许多读者确实有设计8086和80286程序的经验，我们则适当地指出 80386 与86系列这些成员间的区别。附录A总结了80386与8086，80286间的区别。

第二节 数 据 格 式

计算机的主要目的是存贮，检查和操作数据。这样，理解由计算机所支持的数据类型是学习设计新型计算机程序的很好的起点。数据类型是带符号整数，BCD（压缩的和非压缩的二——十进制数）、串、位和浮点数。大多数的数据类型能在大多数计算机中见到。因而，我们强调80386数据类型与其他大多数计算机，更重要的是与80386或 80286 间的区别。

表 1·1　　2 的 幂 值 的 简 写

缩写	2的幂	十进制值
1K	2^{10}	1024
4K	2^{12}	4096
16K	2^{14}	16,384
32K	2^{15}	32,768
64K	2^{16}	65,536
2G	2^{31}	2,147,483,648
4G	2^{32}	4,294,697,296

一、内存

在详细讨论数据类型之前，你需要了解内存的组织，同所有传统的计算机一样，内存是有信息的基本源点和目的点。内存可以简单地看作一字节有唯一地址的连续的字节阵列。地址一般从 0 开始并且向上增加到计算机所支持的最大值。80386是32位机器，总共具有2^{32}字节的物理地址范围即 4G 字节的物理存贮空间。请注意所强调的物理地址范围。在第 2、5 和 7 章，你将学到关于分段和分页的虚拟存贮。你将发现最大的虚拟存贮地址远远超出2^{32}。

如果需要多于 8 位来表示一个数据类型的数值，多个连续的字节则被使用。一个字是两个连续的字节，能够存贮2^{16}个不同的数值。一个双字（dword，或double-word）是四个连续的字节，能够存贮 2^{32} 个不同数值。

关于多字节数据的一个简单而又重要的问题是低位数字节是放在最低地址（即数字上较小的地址），还是放在最高地址。图 1·1 解释了存贮一个字数据到两个字节内存的两种可能的途径。

高位结止方法（在左图）把高 8 位放在该字的最低地址字节，而把低 8 位放在最高地址字节，如果这显得与你所熟悉的东西相反，那是因为 80386 使用了低位结止方法（显示在右

图）。在80386中，一个字的低8位是放在最低地址（m）的字节中，这最低地址也是该字的地址；该字的高8位则存贮在最高地址（m+1）中。

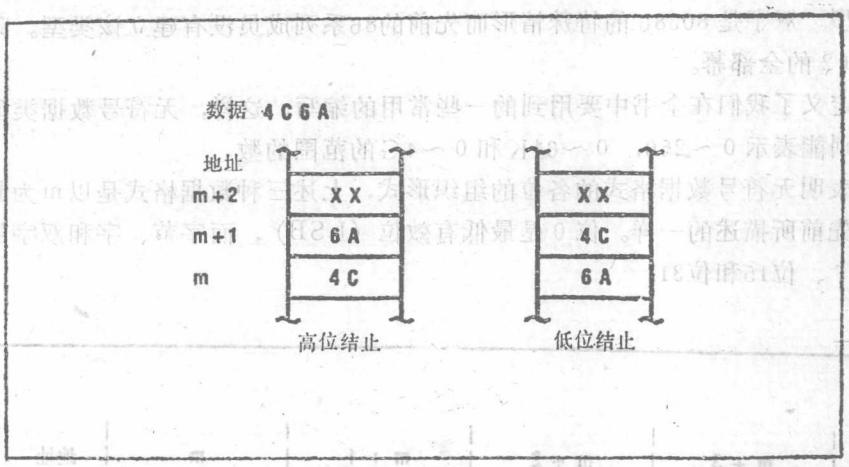


图 1·1 高位结止和低位结止

我们不想解决这个时有的宗教式的争论，并且一直讨厌高位和低位结止步的讨论，但你必须揣磨我们的观点。图1·2表明作为双字的各字节是怎样以“高位结止”和“低位结止”次序存贮在内的。另外，80386是“低位结止”计算机。

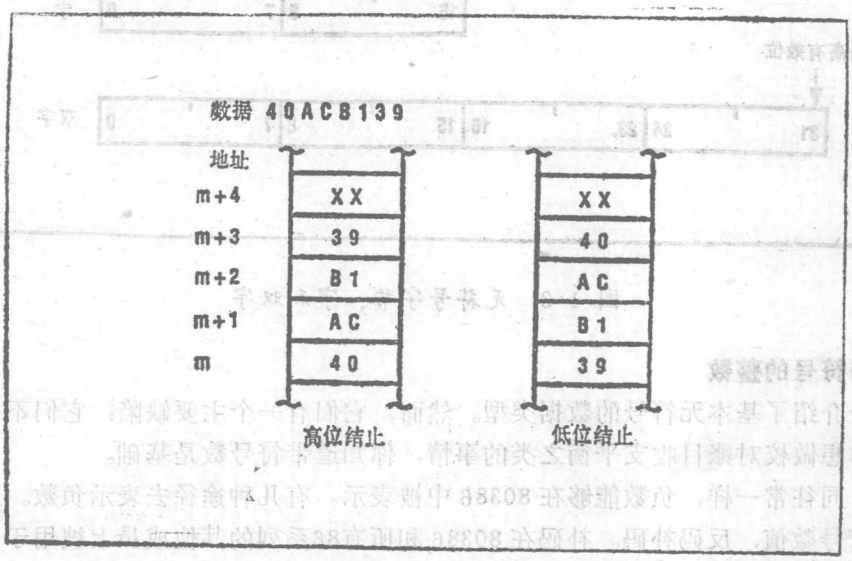


图 1·2 双字的高位结止和低位结止

二、符号

贯穿全书，我们将介绍那些允许准确和非二义说明的符号的方便之处。方便之一是数的说明。后面跟着h的数表明是十六进制数。后面跟着b的数表明是2进制数。如果一个数既无h也无b为后缀，则该数被认作是十进制数。这样，100是十进制数，100b是二进制数（其十进制值为4），而100h是十六进制数（其十进制数为256）。附录C给出了二进制到十六

进制和十进制转换的完整的清单。

三、无符号数

与常规计算机的情形一样，80386 支持字节，字和双字（8、16和32位）的基本的无符号数数据类型。双字是 80386 的特殊情形而先前的86系列成员没有建立该类型。附录D列出了直到2^{32}的 2 的全部幂。

表 1·1 定义了我们在全书中要用到的一些常用的缩写。这样，无符号数据类型对字节、字和双字分别能表示 0 ～256，0 ～64K 和 0 ～4G 的范围的数。

图 1·3 表明无符号数据格式的各位的组织形式，上述三种数据格式是以 m 为起址的内存地址，正如先前所描述的一样。位 0 是最低有效位（LSB），而字节、字和双字的最高有效位分别是位 7、位15和位31。

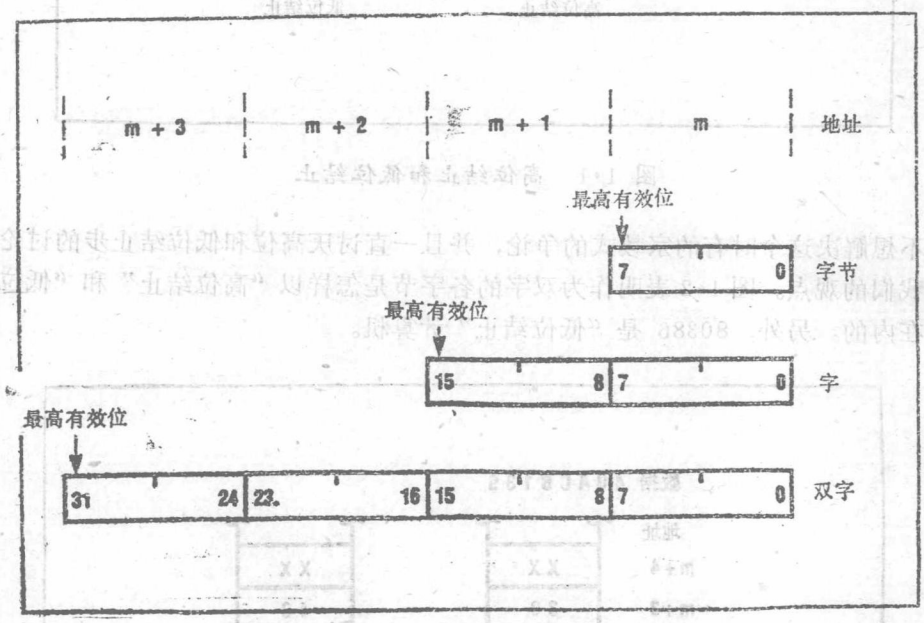


图 1·3 无符号字节、字和双字

四、带符号的整数

前部分介绍了基本无符号的数据类型。然而，它们有一个主要缺陷：它们不能表示负数。如果你想做校对账目收支平衡之类的事情，你知道带符号数是基础。

这样，同往常一样，负数能够在 80386 中被表示。有几种途径去表示负数。这些包括增码数值，符号数值，反码补码。补码在 80386 和所有86系列的其他成员上被用于表示带符号的整数。简单再描述补码表示法，你将看到一些其他的表示法，因此值得在这里对它们作些描述。

表 1·2 列出了上述每一形式的几个数。这虽然不是全部的清单，但它包含了你在此书中用到的所有格式。这表假定它是 8 位数据。

1. 增码数

增码数被用于表示浮点数的指数，因为它们使数据比较（例如大于使小于）简单容易。增码数由初始的正负数值加上偏差值而计算得到。偏差值一般地是既能使所能表示的最大负

表 1·2　　负　数　格　式

十行制数	补码	反码	增码（偏差值=127）	符号数
128	NR	NR	11111111 b	NR
127	01111111 b	01111111 b	11111110 b	01111111 b
126	01111110 b	01111110 b	11111101 b	01111110 b
⋮	⋮	⋮	⋮	⋮
2	00000010 b	00000010 b	00000010 b	00000010 b
1	00000001 b	00000001 b	00000001 b	00000001 b
0	00000000 b	00000000 b	00000000 b	00000000 b
−0	NR	11111111 b	11111111 b	10000000 b
−1	11111111 b	11111110 b	11111110 b	10000001 b
−2	11111110 b	11111101 b	11111101 b	10000010 b
⋮	⋮	⋮	⋮	⋮
−126	10000010 b	10000001 b	10000001 b	11111110 b
−127	10000001 b	10000000 b	10000000 b	11111111 b
−128	10000000 b	NR	NR	NR

注意：你指的是不能以这种格式表示

数变成 0，又能使最大正数变成所能表示的最大值的数值。表1.2表明－127是最大负数，而偏差值127，则使－127的偏差表示值为 0。

2．符号数值

符号数值有一位表示符号（0 为正而 1 为负），其余位表示无符号的数值，既绝对值。浮点数的有效值用带有符号位（此符号位给出该浮点数的正负性）的符号数值。

3．反码

在反码中，MSB 表示数值的符号（0 为正，1 为负）。一个负的反码数可由简单地取反正数的每一位（包括 MSB）计算得到。反码表示法在早期的计算机中是常见的，因为它易于计算（每一位简单取反）。然而，它在现今并不常用。

4．补码

补码表示法叙述如下。由于补码表示法有一个非常好的特性，那就是用于无符号数值的简单的二进制加法器可以不作任何变换地以补码形式相加二个数值，因而它一般被选来表示带符号整数。在象 80386 这样的计算机中，既能支持无符号数据又能支持补码数据这点是重要的。补码是把数值的反码加 1 而计算得到的。与反码的情形一样，MSB 是符号位。MSB＝0表明是正数，而MSB＝1则表明是负数。图1·4显示了80386的 2 的补码形式

下图1·4是 2 的补码字节、字和双字

80386 能够进行补码的各种算术运算。可执行的严格的操作将在第三章中讨论。另外，如用惯常情形，算术运算（例如加法）引起溢出则表明结果有错。这就是 2 个大的正数相加而得到的结果为小正数的情形。第二章讨论象这样的错误如何被记录下来。这些情况类似于在86系列其他成员中所遇到的那样。

80386能够有 8 、16和32位补码数据类型。这些对字节、字和双字来说分别能表示－128

到127，－32K到32K－1，－2G到2G－1范围的数值。

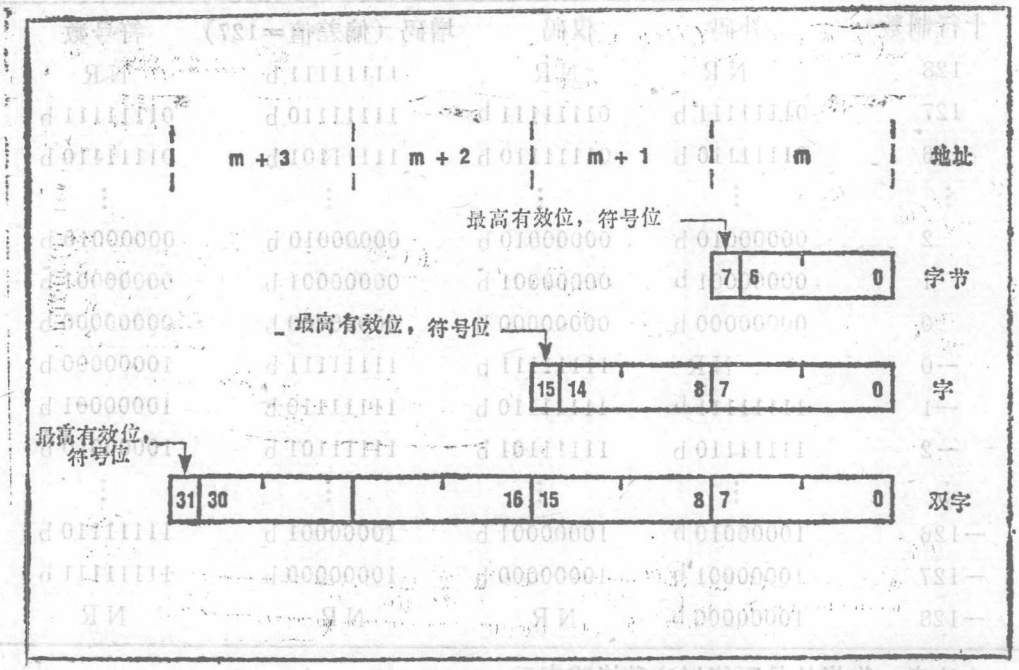


图 1·4　补码数字节、字和双字

五、串

如同先前的系列成员一样，80386 支持串数操作。一个串数是一个连续的字节、字或双字。双字串的支持是80386新增加的。串的长度是从 1 到2^{32}（4G）个元素。图1·5表示了三个类型的串。

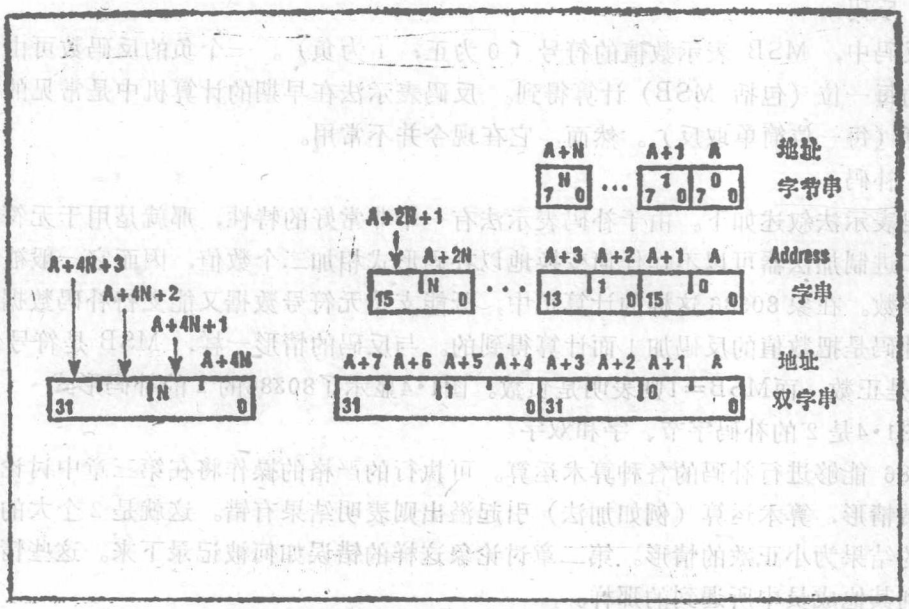


图 1·5　字节、字和双字串

80386具有移动串从内存某一区域到另一区域，比较2个串，用固定元素填充串，从I/O端口读/写串和在串中搜索规定数据的各种指令。

1. ASCII

最常见的串的形式之一是ASCII串。ASCII数据在80386系统中是最常见的数据，因为大多数来自于终端的数据是以ASCII 形式。这样，同86系列其他成员一样，对80386来说支持ASCII是重要的。附录E给出了完整的 ASCII 码表。该表包括数字、字段、特殊字符和控制字符。

80386支持ASCII 数的算术运算，例如加法和除法。这些操作将在第3章中做更详细的叙述。

六、位

我们至今已经讨论的内容表明80386操作的数据至少为8位，而常常是更长。对于位串的操作是80386新增加的，其他的86系列成员不支持该操作。

位支持是重要的，因为数据经常可能被表示为一单个的位。一个常见的例子是位平面显示。在位平面显示中，每一象素既显示平面上的单个点与内容的某一位相对应，内存的这一位或者是1（对应点是亮的）或者是0（对应点是暗的），另一例子是信号灯，单个位表示信号灯是空闲（0）还是忙（1）。当然也可以使用整个象素或信号灯，但势必浪费大量存贮空间，事实上，对应每一数据元素，你将浪费7位，即87..5% 的内存。由于这个原因，80386支持位数据的操作。

80386能够支持长达2^{32}位的位串，其长度由一带符号双字给出。实际操作将在第3章中给出。

图1·6给出了位串在内存中的表示。注意串的下标是一个带符号数值。这样，一个位串不必把位0做为最低有效位来引用。在一个字节内，低位是位0而高位是位7。这个位数值与80386作为低位结止机是相一致的。

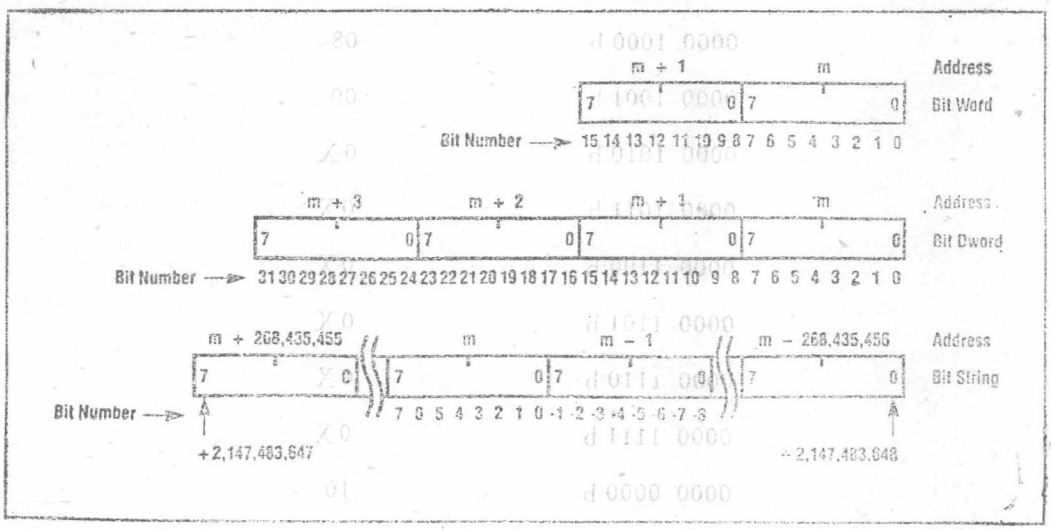


图 1·6 位数据类型

带符号的32位整数用于编址位串中的特定位，这个带符号的 32 位整数称为位偏移量，

其值为－2G～2G-1。位偏称量被分为一个字节地址和一个余量。字节地址等于位偏移量除以8，所需的位就在这个字节内，这个字节内所感兴趣的位是通过偏移数量模8得到的。图1·7给出了2个位例子：在地址为N的一个位串内的位偏移量分别是23和－18。第4章的第四节、第七节和第八节给出了更进一步对位数据进行操作的例子。

七、BCD码

和其它86系列成员一样，80386支持对BCD（二——十进制）数据类型的操作。80386中有对BCD数据进行加减运算的指令。表1·3总结了BCD编码。

80386只直接处理单字节BCD数。在第二章，我们会给出多字节BCD数据是怎么处理的。一个字节可放2个数字，低位数字在第0～第3位，高位数字在第4～第7位，非压缩的BCD数则每个字节放在一个BCD数据，存放在第0～第3位。

表 1·3　　二进制码的十进制表示

二进制	十进制
0000 0000 b	00
0000 0001 b	01
0000 0010 b	02
0000 0011 b	03
0000 0100 b	04
0000 0101 b	05
0000 0110 b	06
0000 0111 b	07
0000 1000 b	08
0000 1001 b	09
0000 1010 b	0X
0000 1011 b	0X
0000 1100 b	0X
0000 1101 b	0X
0000 1110 b	0X
0000 1111 b	0X
0000 0000 b	10
0001 0001 b	11

注：X表示BCD表示的非法值