80386程序设计及其技术

H

中国科学院希望高级电脑技术公司

一九八八年九月

前 言

本书讨论微处理器80386和80387的特性并提供有关的程序设计技术。

80386是流行的Intel 86处理器系列目前最新、最强功能的成员。它在1985年被推出，比80286和8086有许多重大的改进。两个最主要的增强是32位的操作和数据类型，以及除分段技术外，还把分页作为存储管理技术。

此外，80386扩展了80286的多任务能力。80386允许同时执行8086、80286和80386任务和操作系统。80386能执行在80286或8086上执行的任何程序。换句话说，它是和8086，80286兼容的。与它相联的数值协处理器是80387。它们紧密配合提供支持浮点操作和数据类型的计算机体系结构。

目　录

第六章　中断和异常

第七章　操作系统实例

第八章　调试支持程序

第九章　执行8086和80286程序

第一章　基本概念

1.1　Intel微处理器的历史

Intel在1971年开发了第一台微处理器4004。没多久它就被改进为8008。按今天的标准这些设备是十分平凡的，它们虽然新颖但严格说来还很难得算上是有价值的计算机。1974年Intel的第二代微处理器8080诞生了。这是第一台通用的微处理器，对微处理器工业的发展起十分重要的作用。1978年诞生了第三代微处理器8086。这标志着微处理器作为“真正”计算机的开始。同时这也就是86系列的开始。

8086是16位的微处理器，8080是8位的，而4004是4位的。8088（8086的小兄弟）用在IBM PC以后（1981年推出），引起了个人计算机的重大改革，随着这个和许多使用8086和8088的其它设计的出现，从那时候起到现在，8086体系结构变成了最重要的微处理器构。

但是，这个系列并没有停滞在8086。在1982年，Intel推出80186。这种元件在结构上和8086一致，但在相同元件上包含了一些其它的一般系统设备。也是在1982年，又推出了80286。80286是8086的“结构性超级设备”，这意味着它能精确地象8086那样运算，但还具有更多功能。特别是，它增加了对多任务的支持，那是可以一次执行多于一个应用程序（或任务）的能力。多任务要求80286支持每一任务和每一任务的存储区之间的保护。80186和80286都是16位的元件。

最后，在1985年推出了80386。它比80286和8086有两个主要的和许多微小的增强。这些增强已概括在附录A中。

每一代的86系列都有它们相应的数值协处理器。主要的86系列元件是8086、89286和80386，而它们对应的数值协处理器分别是8087，80287和80387。这几代的浮点元件之间微小的差异已在附录B中列出。

每一代的86系列都保持与上一代成员的兼容。第九章描述如何在80386上运行8086和80286程序。

1.2　数据格式

计算机的主要目的是存储、检索和运算数据。因此，了解机器所支持的数据类型是学习为新计算机编写程序的好起点。各种数据类型是：带符号和无符号数、BCD（压缩和非压缩二进编码的十进数），串、位和浮点数。

存储器

在开始详细讨论数据类型前，需要了解一下存储器的组织。如所有传统的计算机那样，存储器是所有信息主要的源和目标。存储器可以简单看成是字节的顺序阵列，每一字节都有唯一的地址。地址通常从0开始，向上增加到计算机所支持的最大地址。80386是32位的机器，总的“物理”地址范围是2^{32}字节或4G字节的物理存储。注意，这里强调的是物理地址范围。在第二、五和七章你将知道有关分段和分页虚拟存储。你会发现最大的

虚拟存储地址要比 2^{32} 大得多。

如果为表示数据类型中的值需要多于 8 位，则使用多个连续的字节。“字”是两个连续字节，可以存放 2^{16} 个不同的值。“双字”是四个连续字节，可以存放 2^{32} 个不同的值。

多字节数据的一个简单但重要的问题是，低阶字节在最低地址（数值上较小的）还是在最高地址。图1.1说明存放字数据在存储的两字节中的两种可能的方法。“大尾端”方法（在左边）把高阶 8 位放在字的最低编址字节，把低阶 8 位放最高编址字节。如果这看起来象和你已熟悉的相反，那是因为80386使用的是右边示出的“小尾端”方法。在80386，字的低阶 8 位是在带有最低地址（m）的字节（m也是字的地址）。字的高阶 8 位存放在最高地址（m + 1）。

我们并不打算解决这个不时发生的有关大与小尾端的信仰之争和总是热烈地讨论。不过，你们大概能猜得到我们的意见是什么。图1.2示出对于大和小尾端的方法，字节是怎样存放在存储的双字中的。再说一遍，80386是小尾端的计算机。

记号

贯穿本书，我们将介绍允许精确而无二义说明的记号方便表示。一种方便表示是数的说明，跟有h的数表示16进（基16）数，跟有b的数表示 2 进（基 2 ）数。如果该数既没有h也没有b后缀，则假定它为十进数。于是，100是十进数，100b是二进数（十进数值 4 ），而100h是十六进数（十进数值256）。附录c给出二进制向十六进制和十进制转换的完整表。

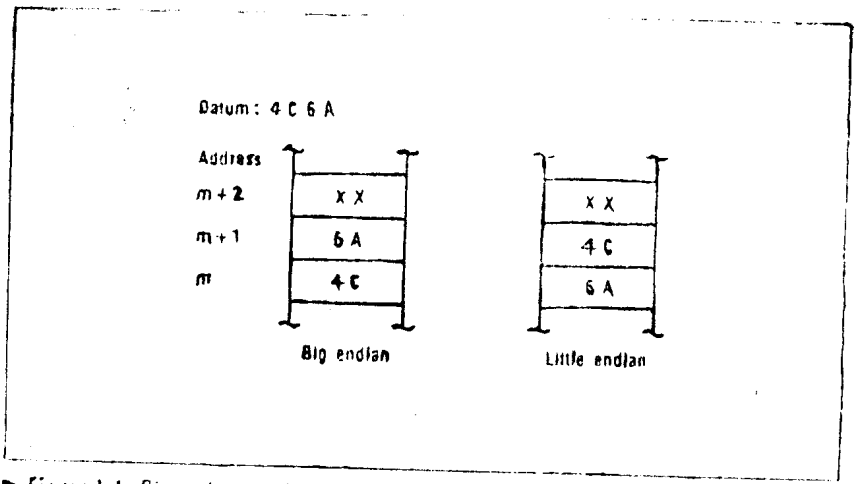
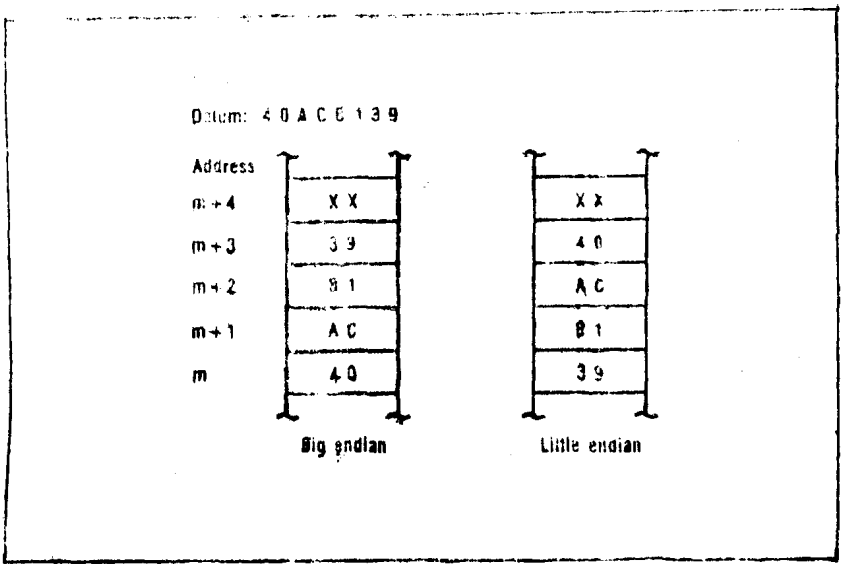


Figure 1.1: Big endian vs. little endian

图1.1 大尾端和小尾端



► Figure 1.2: Big endian vs little endian for dwords

图1.2 对双字的大尾端与小尾端

无符号数

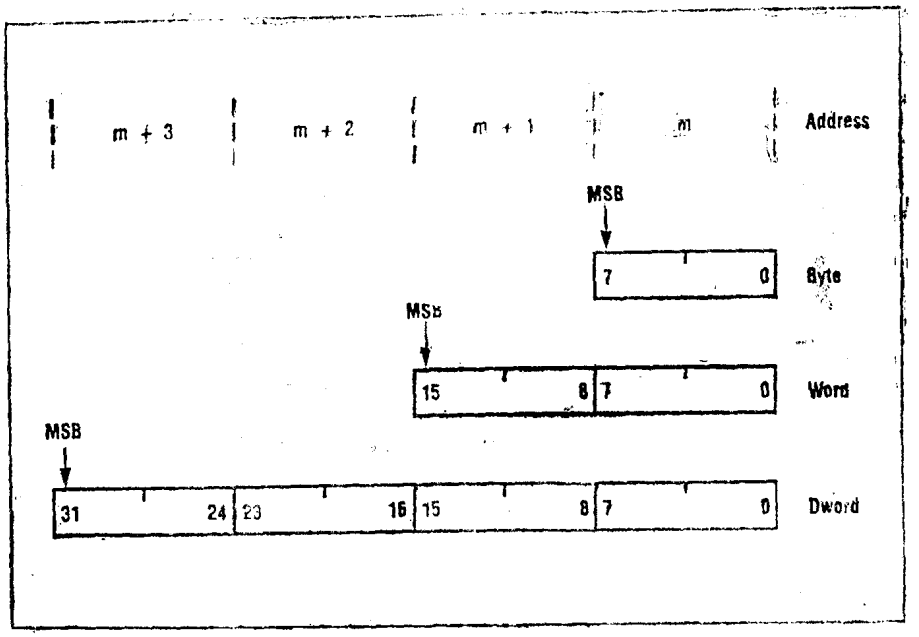
象通常的计算机那样，80386支持字节、字和双字长（8、16和32位）类型的基本无符号数。双字是添加到80386的，先前的86系列成员没有。附录D给出直到2^{32}的2的幂的完整表。

表1.1定义我们在全书使用的少量公用的缩写，于是，无符号数据类型可表示数的范围从0到256，从0到64K，和从0到4G（分别对应于字节、字和双字）。图1.3示出无符号数据格式中位的组织。在三个数据格式上面是先前提出的从地址m开始的存储地址。位0是最低有效位（LSB）。无符号数的最高有效位（MSB）是位7、15和31（分别对应于字节、字和双字表达）。

ABBREVIATION	POWER OF TWO	DECIMAL VALUE
1K	2^{10}	1024
4K	2^{12}	4096
16K	2^{14}	16,384
32K	2^{15}	32,768
54K	2^{16}	65,536
2G	2^{31}	2,147,483,648
4G	2^{32}	4,294,967,296

► Table 1.1: Abbreviations for powers of two

表1.1 二的幂的缩写



► Figure 1.3: Unsigned byte, word, and dword

图1.3 无符号字节、字和双字

带符号整数

前一节提出基本的无符号数据类型。但是，它们有一种主要的不足：它们不能表示负数。如果你想核对你的帐目结余，你将会感到带符号数是很重要的。

于是，作为一般的情况，80386可以表示负数。有几种方法表示负数。这包括偏置数、符号尾数、反码和补码。80386和所有其它的86系列成员使用补码以表示带符号整数。在简短的离题以描述表示负数的另一种方法之后，我们将描述补码表达。

表1.2给出每一这些形式的若干数的表。它采用 8 位的数据。

偏置数用于表示浮点数的“指数”，因为它们使得数字比较（象小于或大于）容易。通过对原始的正或负数添加一偏置值可算出偏置数。偏置通常使得表达中允许的最负的数变为 0，而最正的数变为表达的最大值。表1.2展示了这点。它用127作为最负的数，127的偏置使 0 成为 - 127的偏置表达。

带符号尾数有表示符号的位（如果正为 0，如果负为 1），其余的位给出表示数的尾数（或绝对值）的无符号整数。用尾数和给出整个浮点数符号的符号位表示浮点数的有效数字。

在反码数中，MSB指示数的符号（如果正为 0，如果负为 1）。负的反码数可简单地通过逆转正数的每一位（包括MSB）求得。反码表示在早期的计算机是很普遍的，因为它是如此容易计算（简单的逆转每一位）。但是，今天一般已不用它。

下面描述补码表示。通常选它来表示带符号整数，因为它有合意的特性。为无符号数使用的简单二进加法器也能加两个补码格式的数而不用附加的转换，这对于象80386这样的计算机是重要的，它支持无符号和补码数据。

通过计算数的反码并把 1 加该结果可形成补码。和反码的情况一样，MSB是符号位。

MSB = 0 指示正数，而MSB = 1 指示负数。图1.4显示80386的补码形式。

80386可以对补码数执行各种算术运算。80386可以有 8 、16和32位补码数据类型。这些可以表示范围从 - 128到127，从 - 32K到32K - 1， 以及从 - 2 G到2 G - 1 （分别对应于字节、字和双字）。

DECIMAL	TWO'S COMPLEMENT	ONE'S COMPLEMENT	BIAS (BIAS = 127)	SIGN MAGNITUDE
128	NR	NR	11111111b	NR
127	01111111b	01111111b	11111110b	01111111b
126	01111110b	01111110b	11111101b	01111110b
⋮	⋮	⋮		
2	00000010b	00000010b	10000001b	00000010b
1	00000001b	00000001b	10000000b	00000001b
0	00000000b	00000000b	01111111b	00000000b
-0	NR	11111111b	NR	10000000b
-1	11111111b	11111110b	01111110b	10000001b
-2	11111110b	11111101b	01111101b	10000010b
⋮	⋮	⋮	⋮	
-126	10000010b	10000001b	00000001b	11111110b
-127	10000001b	10000000b	00000000b	11111111b
-128	10000000b	NR	NR	NR

Note: NR indicates not representable in this format.

► Table 1.2: Negative number formats

表1.2 负数格式

串

和以前的86系列成员一样， 80386支持对数据的串运算。串是字节、字和双字的连续序列。支持双字的串是80386新有的。串的长度是从 1 到 2^{32} （ 4 G ）个元素。图1.5示出三种的串。

80386有指令把串从一个存储区向另一个移动、比较两个串、用定量的元素填写一个串、从输入／输出端口读或写串以及为指定的数据值搜索串等。

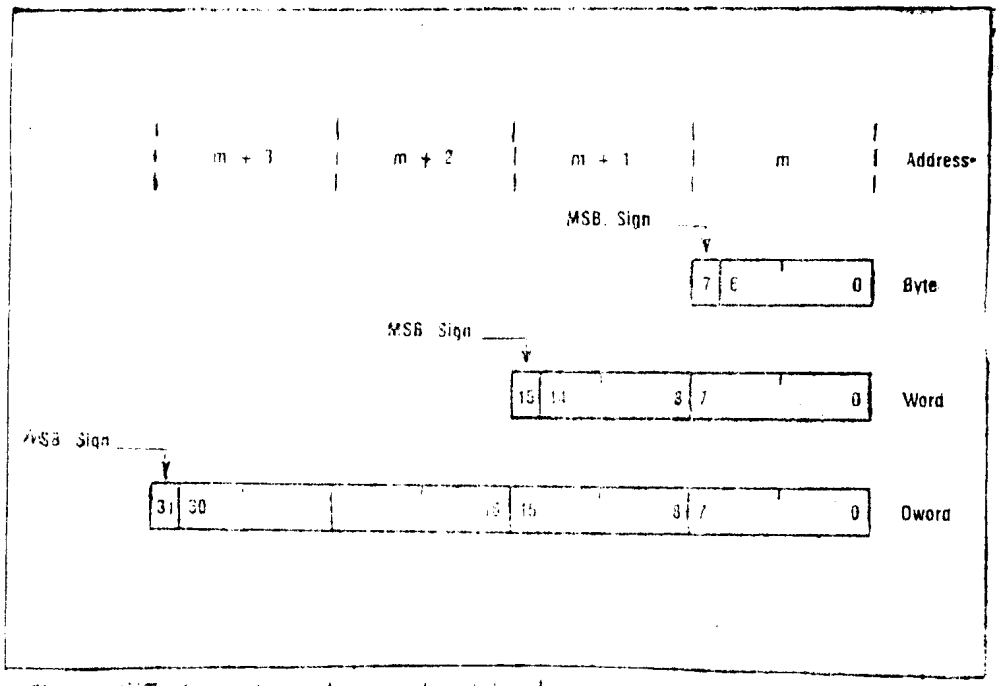


Figure 1.4: Two's complement byte, word and dword

图1.4 补码字节、字和双字

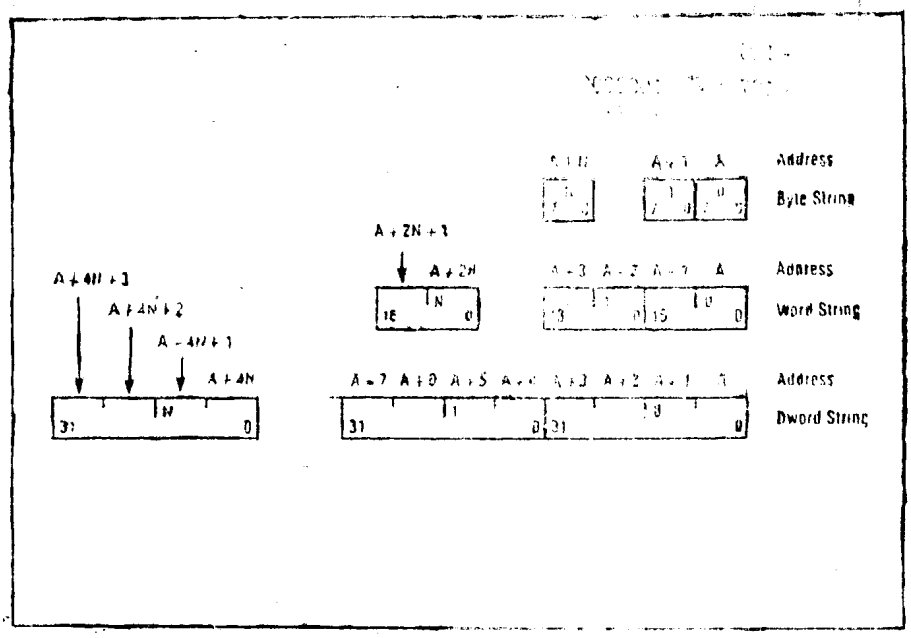


Figure 1.5: Byte, word and dword strings

图1.5 字节、字和双字的串

ASCII

字的最普遍形式之一是ASCII串。在80386系统中ASCII数据十分普遍，因为来自

终端的大多数数据是ASCII(American Standard Code for Information Interchange)。于是，和其它86系列成员一样，80386支持ASCII是重要的。附录E给出完整的ASCII表。这包括整数、字母字符、专用字符和控制字符。

80386支持对ASCII数的加法和除法等算术运算。这些运算在第三章会更详细地描述。

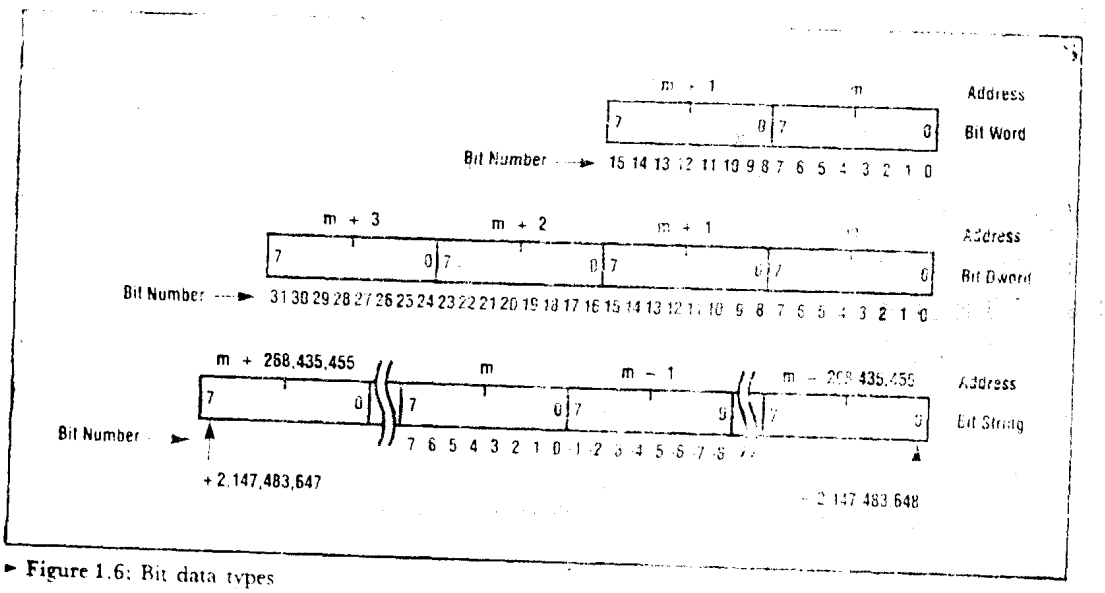
位

迄今我们已经讨论的一切，示出80386对至少8位宽（往往更长）的数据运算。对位串运算的支持是80386新有的，因为没有其它86系列成员支持它。

位支持是重要的，因为数据常常可表示为单个位。一个普通的例子是位图表示。在位图显示中，每一个“象点”（显示中的单个点）变换为存储中的单个位，它或者为1（点或萤光是亮的）或者为0（点是黑的）。其它例子是信号，其中单个点指示信号是闲(0)还是忙（1）。也有可能把整个字节奉献给每一象点或信号的，但这将浪费许多存储。事实上，你将对每一数据元素浪费7位（或87.5%的存储）。为此，80386也支持对位数据的操作。80386支持包含多至2^{32}位由带符号双字索引的位串。第三章给出实际的操作。

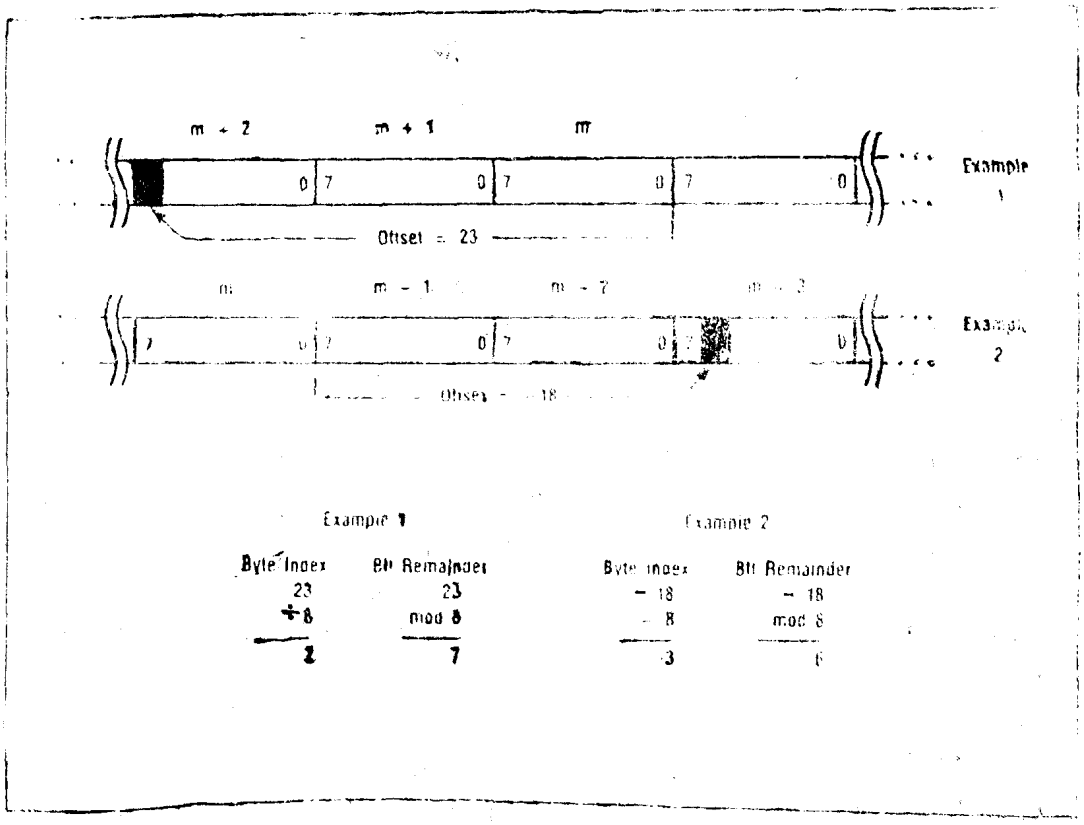
图1.6示出存储中位串的概貌。注意，位索引是带符号的数。于是，位串不需要有作为0用的最低有效位。在一个字节中，低阶位是位0而高阶位是位7。位编号与作为“小尾端”机器的80386是一致的。

使用32位带符号整数寻址位串中的特定位。这个32位带符号整数被称为“位偏移”，可以有值从－2G到2G－1。位偏移划分为字节地址和位余数。字节地址是指定的位偏移除以8，而要求的位就在这个字节中。在这个字节中感兴趣的位由位偏移模8确定。图7给出两个位的例子：在地址N的位串中的位偏移23和位偏移－18。



► Figure 1.6: Bit data types

图1.6 位数据类型



►Figure 1.7: Bit string offset examples

图1.7 位串偏移例子

BCD

如同其它的86系列成员那样，80386支持对BCD（binary-coded decimal）数据类型操作，80386包含允许加和减BCD数据的指令。表1.3概述BCD的编码。

80386只直接处理字节BCD量。在第三章，我们将示出如何处理多字节的BCD数。两个BCD数字装在一字节中，低阶数字在位0到3，而高阶数字在位4到7。非压缩BCD每字节存放一个BCD数字，在位0到3。

1.3 浮点数据类型

浮点数据类型由80386的数值协处理器（80387）支持。它们非常类似于8087和80287，而它们的差异在附录B说明。

IEEE浮点标准

在深入到80387的数据类型细节之前，我们用少量注释谈谈有关IEEE浮点标准。在1979年前，许多主机和小型机都有浮点数据类型，但有关浮点表达却没有什么标准。事实上，你可以在计算机A运行一个FORTRAN程序，而后来又在计算机B上运行它，并得到十分不同的结果。

在开发8087（8086和8088的数值协处理器）时，Intel已有兴趣开发浮点标准，特别是

8087能执行的标准。Intel发表了它建议的标准后不久，IEEE批准一个委员会开发浮点运算的标准。经过这个委员会的许多成员几年的工作，得到IEEE任务草案10.0 P754的“二进制浮点运算标准”。这个标准的几点目的如草案所陈述，要求：

1．“简化”把已有程序从形形色色的计算机向那些追随此标准的计算机的迁移。

2．提高程序员的能力。这些程序员即使不是数值方法的专家，也能良好地产生数学上高深的程序。

3．鼓励专家开发和贡献健全和有效的可移植数值程序，通过少量编辑和再编译。可移到遵从这个标准的任何计算机上。

IEEE标准在工业上正获得承认，不仅Intel芯片支持这个标准，许多其它的微处理

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

Note: X indicates an illegal value for BCD representation.

► Table 1.3: Binary coded decimal representations

表1.3 二进制编码的十进制数表达

器，小型计算机以及主机计算机也支持它。80387追随这个标准，简言之，80387提供非常广范围的数据支持，很好的立即值支持，安全的结果以及高性能。

数据格式

80387支持7种数据类型，概括在表1.4。请注意能表示多大的数。记住386支持的最大整数为 2^{32}（或$4.29*10^9$十进制）。

还应注意80387支持整型数据类型。如果计算有实和整型数据。整个计算可以用80387执行。不要求在80387和80386之间的数据传输。

DATA TYPE	BITS	SIGNIFICANT DIGITS (DECIMAL)	APPROXIMATE RANGE (DECIMAL)
Word Integer	16	4	$-32768 \leq X \leq 32767$
Short Integer	32	9	$-2*10^9 \leq X \leq +2*10^9$
Long Integer	64	18	$-9*10^{18} \leq X \leq +9*10^{18}$
Packed Decimal (BCD)	80	18	$-99..99 \leq X \leq +99..99$ (18 digits)
Short Real	32	6-7	$-3.39*10^{-38} \leq X \leq 3.39*10^{38}$
Long Real	64	15-16	$-1.80*10^{-308} \leq X \leq 1.80*10^{308}$
Temporary Real	80	19	$-1.19*10^{-4932} \leq X \leq 1.19*10^{4932}$

►Table 1.4: Data types supported by the 80387

表1.4　80387支持的数据类型

整型数据类型

下表概括80386和80387支持的整型数据类型。在此表中，NR指示这个处理器表示不了。

位	80386	80387
8	带符号字节	NR
16	带符号字	字整数
32	带符号双字	短整数
64	NR	长整数

图1.8示出整型数据类型。这三种数据类型可表示数从-32768到32767，从$-2.147*10^9$到$2.147*10^9$，和从$-9.233*10^{18}$到$9.233*10^{18}$（分别适应于用字、短和长整数格式）。

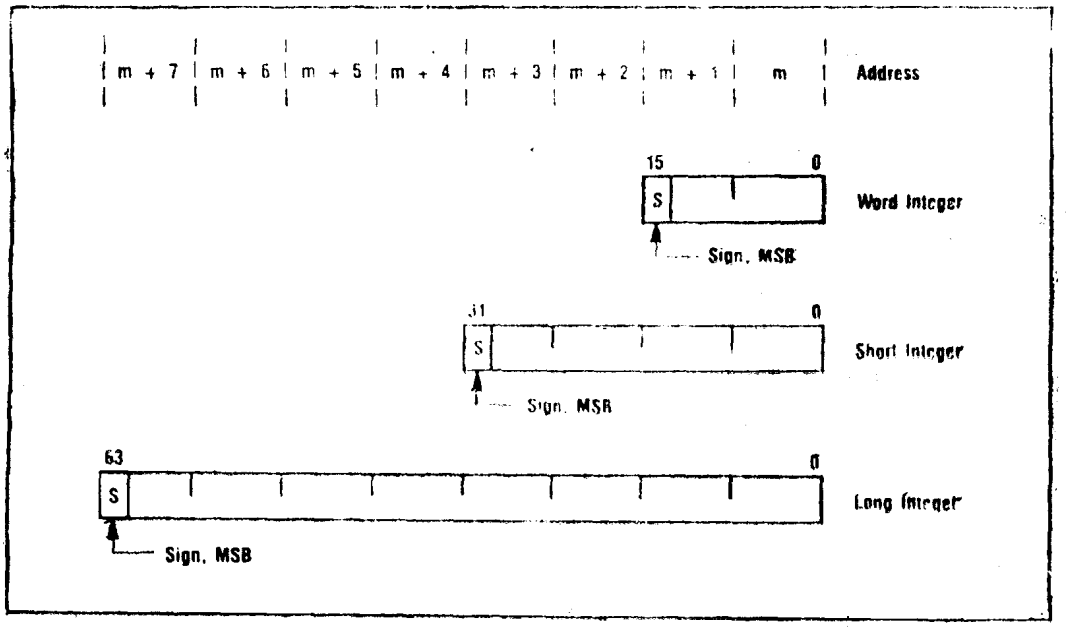


Figure 1.8: Integer (two's complement) 80387 data types

图1.8 整型（补码）80387数据类型

BCD

再说一下，80386支持BCD，为什么80387也支持？80387支持这种数据类型的理由基本上与上面对整数给出的道理相同。80387支持的BCD类型是压缩的80位十进类型，它保存18个十进数字和1个符号位。这在图1.9示出。为什么压缩BCD停在18个数字并在表达中留下7个不用的位？因为18个数字满足COBOL标准（一种使用BCD的主要语言），并且没有理由要超过18个数字而把设计复杂化。

实格式

实格式是上面提到的浮点格式。图1.10给出一般的实格式。此格式由三部分组成：有效数字、指数和符号。

下表给出用于三种实数据格式的每一部分的位。

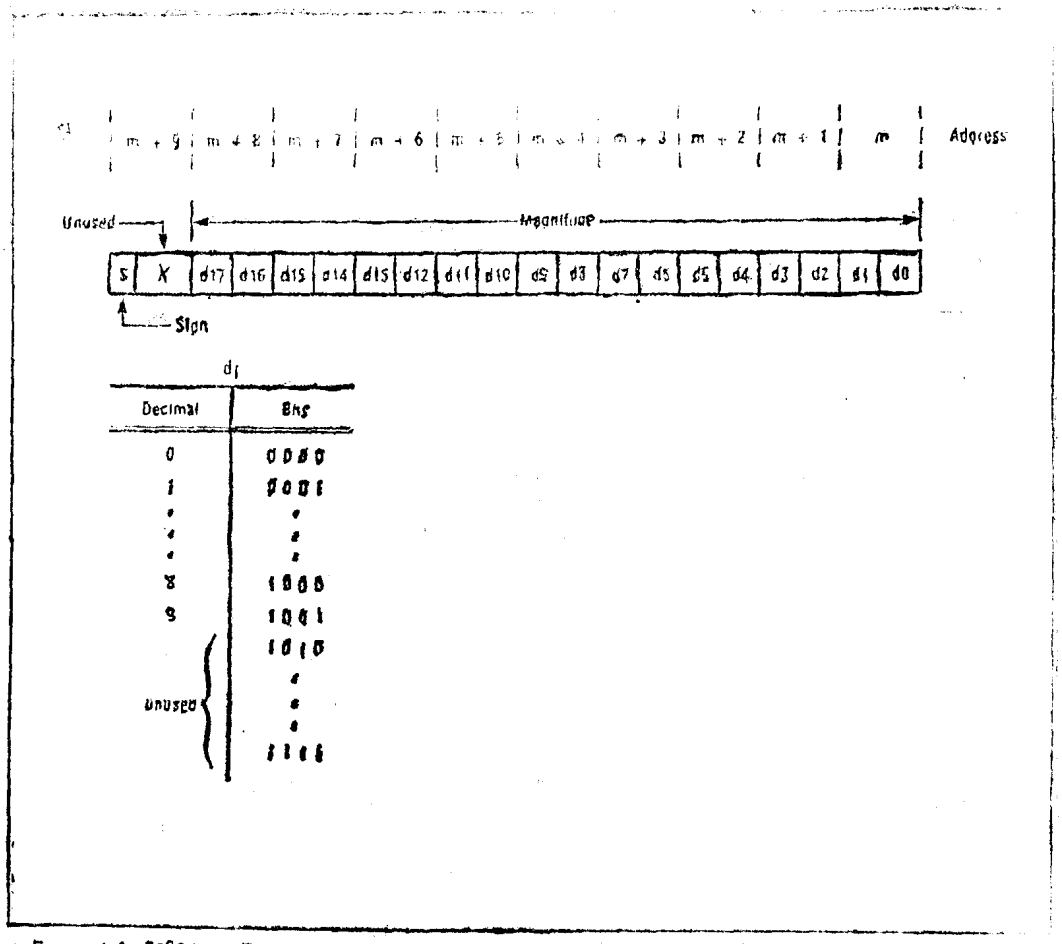


Figure 1.9: 80387 BCD data type

图1.9 80387 BCD数据类型

数据类型	总数	符号	指数	有效数字
短实数	32	1	8	23
长实数	64	1	11	52
临时实数	80	1	15	64

图1.11给出三种数据类型详细的位的位置。让我们从最容易的一种开始。“符号”是简单的一个符号位。如果这个位是1，则该数是负的，如果这个位是0，则该数是正的。“有效数字”给出数的有效数字位。在某些上下文中有效数字称为尾数。“指数”字段也含需要放大有效数字以取得最后结果的二的幂。指数以偏置的形式存放。这样做是为了简化数值比较，因为大的数值总是大数——其它表达如带符号整数情况不是这样。对短、长和临时实格式的偏置分别是127、1023和16383。于是10000000b的指数（短实数）实际上是2^1。

现在我们已经解释了浮点格式的所有三种成分，我们准备用图1.12和少量例子概述它们。

图1.12中的“范围”给出这种数据类型能表示的最大数（正的和负的）。“精度”给出这种数据类型能表示的最小的可能数。“不规格数”在下面讨论，为完整性在这里给出。所有例子都将以短实数给出，因为短实数有足够充分的位使它难于领会。

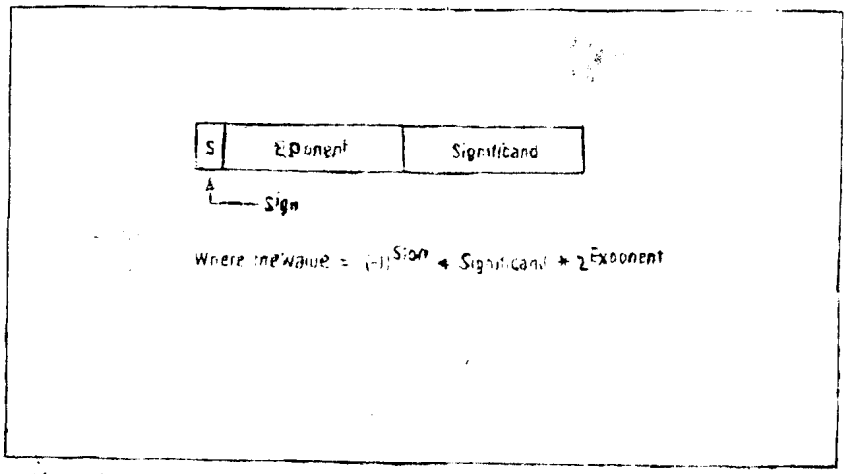


Figure 1.10: General real format

图1.10 一般实格式

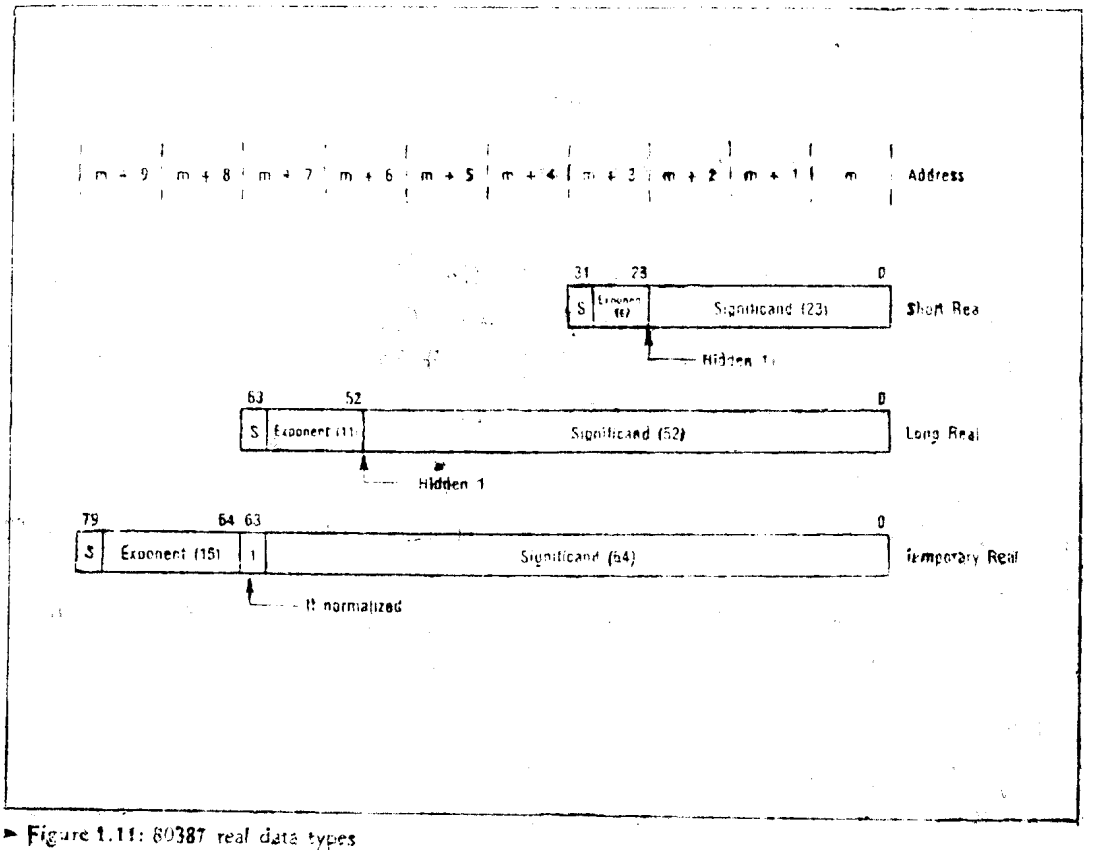


Figure 1.11: 80387 real data types

图1.11 80387实数据类型