80386技术、性能、特点及总体论述

北京科海高技术集团公司（培）

一九八八年十二月

编辑：科 海 培 训
发行：科 海 培 训
地址：北京2725信箱
资料组
（北京海淀路3
印刷：河 北 省 蔚

前　言

本书向软件工程师、硬件工程师和经理人员综合性介绍Intel 80386微处理器。

经理人员应认真阅读第一章，在这一章中概括了80386的特点和优点。其后两章的对象分别为应用软件工程师和系统软件工程师。第二章叙述面向汇编语言程序员或编译程序设计者的80386体系结构；第三章讨论的则是支持操作系统的体系结构的有关内容。对在80386上运行8086和（或）80286软件感兴趣的读者应阅读第四章。最后一章是为硬件工程师准备的，本章主要描述了80386的外部接口。

由于本书所述内容不尽详细，为方便起见，不妨对你所不熟悉的内容进行略读。

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第一章 概 述

80386是一高性能32位微处理器，在其设计过程中充分考虑了现在及将来基于计算机应用的需要。目前成功地使用80386的典型应用有：CAE/CAD（计算机辅助教育／计算机辅助设计）工作站、高分辨率图形、排版及办公室和工厂自动化等。将来的应用更被系统设计者的想像力所限制，而不是80386所具有的能力和适应性。

80386为系统设计者提供了很多崭新而强有力的能力，它们是：每秒钟执行3-4百万条指令的空前性能、全32位的体系结构、4GB（2^{32}字节）物理地址空间及分页式虚拟存储器的片载支持等。在利用了最新微处理器技术的同时，80386还在目标代码级保持同8086和80286软件的兼容性。尤其是80386的虚拟机（VM）能力，使80386能对运行在不同操作系统（如Unix和DOS）环境下的程序间进行切换。这样可使OEMS（初始设备制造厂家）把标准16位应用软件直接容入崭新的32位应用软件的设计中。

结合了超级小型机（高速度，高性能）和微型机（造价低、适应性强）二者的优点，80386可以开辟基于微处理器系统的新的市场。在过去的许多应用中使用微处理器速度比较慢，而使用超级小型机造价又太高；现在80386为这些应用提供了最好的选择。由于80386的出现，使目前涌现出具有很大实验性的机器视觉、语音识别、高级机器人及专家系统等应用可以形成商品投放市场。

未来的应用，需要比32位更宽的寄存器、指令和总线。而在80386的设计过程中考虑了这些因素。下面几节概述了80386的32位体系结构及其富有创新性的特色：

- 高性能实现
- 虚拟存贮器支持
- 可构造性保护机制
- 扩展的调试支持
- 目标码兼容性

1.1 32位体系结构

80386的32位体系结构为支持“大型”应用提供程序设计资源。所谓“大型”应用，即由大型整数、大型数据结构、大型程序（或大量的程序）等所刻化的应用。80386的物理地址空间为2^{32}字节或4GB；其逻辑地址空间是2^{46}字节或64TB。80386的8个32位通用寄存器既可以用作指令操作数也可以用作寻址方式变量。其数据类型有8位、16位及32位整数和序数（ordinal）、压缩（packed）及非压缩十进制数、指针及位串、字节串、字串和双字串。80386具有一个完整的指令系统处理这些类型的数据，并控制机器的运行。80386还为访问如下标准数据结构中的元素提供了相应的寻址方式支持：数组、记录、记录数组和其中包含数组的记录。

1.2 高性能实现

32位体系结构并不意味着具有高性能。要发挥这一结构的潜能还需要先进的半导体技术、精心的功能分划及对脱片操作，尤其是对处理器和存储器间交互作用的管理。把所有这些因素协调起来，可以说80386发挥出了现行可用微处理器的最高性能。

80386是用Intel公司的CMOSⅢ工艺制成的，这是一种综合了HMOS的高频特性和CMOS的一般电源要求的新型半导体工艺。利用1.5微米几何形状和双金属层结构，将275,000个晶体管压缩到一块芯片上。80386的运行频率有两个，分别为12兆赫和16兆赫；在运行时不需等待状态，当80386以16兆赫的频率运行时，每秒钟可执行3-4百万条指令。

80386由六大部件组成。这些部件自动并行工作，需要时还可以进行同步。连接这些部件的所有内部总线都是32位宽。由于用流水线方式组织这些部件，80386可以并行执行同一条指令的不同阶段，并能够同时处理多条指令。这样可达到如下的并行执行效果：当一条指令正在执行时，另一条指令被译码，同时第三条指令正被从存储器中往外取。

80386除了以流水线方式执行所有指令外，还使用专用硬件执行一些重要操作。依赖于有效数位的个数，80386的乘/除部件能够在9-14个时钟内完成两个32位数的乘法运算；对两个无符号或有符号操作数的除法运算分别在38个时钟和43个时钟内完成。

很多涉及32位机的应用，如可改编程序的多用户计算机，需要存储器管理部件(MMU)进行逻辑地址到物理地址的转换并提供保护。另外一些应用，如嵌入式实时控制系统却不需要这些。鉴于这两种要求，大多数32位微处理器在一个选件芯片上来实现存储器管理部件。而80386的MMU（由两个功能部件组成）以流水线方式同其它80386部件连接共同在微处理器芯片上实现。由操作系统控制MMU的操作，如允许一个实时系统放弃页转换。片载实现存储器管理显然提高了那些需要MMU的应用的性能，同时不会损害那些不需要MMU的应用的性能。之所以能实现这一目标，是由于片载实现存储器管理，可以缩短信号传播延迟、利用半时钟周期和进行并行操作。

80386提供的另一设施是“数字离弄”，尤其是对单、双精度浮点数的运算。这对某些应用是至关重要的。浮点操作数是大操作数，对它们的运算是非常复杂的；几千个晶体管用来实现标准的一组浮点运算，例如由IEEE 754 标准定义的运算。因而，80386 通过一个分离的数字协处理器芯片，为数字(运算)提供硬件支持。有两个数字协处理器可供选择，即80287和80387，其中任何一个都可与80386相连。数字协处理器对应用软件而言是透明的；它们使用与IEEE 754 标准兼容的寄存器、数据类型及指令，从而有效地扩展了80386体系结构。80386若同80387相连，则每秒钟可以执行一百八十万次数字运算。[1]

一个以16兆赫的频率运行的32位处理器，其速度可超过几乎所有存储器的访问速率，从而使存储器访问时间成为一个潜在的瓶颈口。为了获得较好的性能价格比，80386的存储器既使用了高速静态RAM，也使用了造价较低的动态RAM，并进行了相应的总线设计。当访问快速存储器（如超高速缓冲器）时，80386提供的总线周期是两个时钟。(注：80386的超高速缓冲器的大小可为4K字节到整个物理地址空间范围内的任何大小）。当访问较慢的存储器（或I/O设备）时，可以利用80386的地址流水线设施，将总线周期扩展为三个时钟；而处理器仍维持两个时钟的吞吐量。由于地址转换同指令执行的内在并行性，80386通常在当前总线周期内计算地址及下一总线周期定义。地址流水线将这一超前信息告知存储器子系统，从而达到如下并行效果：一个存储单元对下一总线周期进行译码，同时另一存储单元对当前周期进行响应。

1.3 虚拟存储器支持

在一个虚拟存储器系统中，程序访问的存储空间由RAM存储器扩展为整个磁盘空间（注：RAM存储器比磁盘贵400多倍）。这一灵活性使制造商、程序员及最终用户都能受益。制造商可以提供多个性能级产品，这些产品只是存储器构成有所不同；程序员可以把存储管理留给操作系统，而无需使用写覆盖；最终用户可以运行更多更大的应用而不用考虑存储器是否够用。

虚拟存储由操作系统及其硬件支持来实现。80386支持基于段或页的虚拟存储系统。通常基于段的虚拟存储适宜于较小的十六位微机系统，其段最大长度是64K字节。而80386所支持的虚拟存储系统中，段的最大长度为4GB，所以，大多数使用80386的大规模系统把虚拟存储系统建立在80386的请求分页设施之上。80386为每一页设置了存在位（Present）、污损位（Dirty）及访问位（Accessed），以实现请求页式虚拟存储。当一条指令引用一个不存在的页时，80386自动转入操作系统；一旦操作系统从磁盘中调进所请求的页，80386即继续执行该指令。为提高虚拟存储系统的性能，80386为分页信息提供了一个联想式片载高速缓存。在该缓存中（该缓存被称为转换备用缓冲器或TLB）包含有32个最近使用页的映射信息。每一个80386页长为4K字节，这样使用TLB可以立即映射128K（4K×32）字节的存储区，从而使80386能片载转换大多数地址而无需查阅一个基于存储器的页表。在典型的系统中，98—99%的地址引用都落在TLB所映射的存储区中。

1.4 可构造性保护机制

由于每秒钟可执行3-4百万条指令，80386有能力支持极其复杂的应用，这些应用由几百甚至几千个程序模块组成。在这些应用中，问题不在于是否会出错，而是如何尽快地发现并纠正错误，以及如何严格限制由这些错误所导致的危害。如果处理器对每一条指令同保护准则的一致性都进行了验证，则可以较快地对这些应用进行排错，从而使之更健壮。但是保护机制同特定的应用有关，不同的应用采用不同的保护机制，有些应用如简单的嵌入式实时应用，不用保护设施照样工作得很好。通过有选择地使用一组保护设施，来满足一定的保护需要。为此80386提供了如下保护设施：

- 任务地址空间的分离
- 0～4共五个特权级
- 特权指令（例如，Halt指令）
- 类型段（如，代码段，数据段）
- 段和页的访问权限（如，只读，只执行）
- 段上限检查

所有80386的保护性检查都是通过片载流水线方式进行，从而达到最优性能。

1.5 扩展的调试支持

80386使用四个片载调试寄存器，从而大大减少了程序调试时间。这些寄存器的工作独立于保护系统，因而包括那些运行时不需保护的应用在内的所有应用都可以使用这些寄存器。值得一提的是，除了指令断点（这是大家都熟悉的）外，它们还具有设置数据断点

的能力。80386同时管理四个当前断点地址，但它执行速度并不因此而减慢。

一旦设置了指令断点，当程序执行到断点处时一般转到调试程序中去执行。大多数处理器提供这种能力，通常由调试者向所感兴趣的指令位置上写一条特殊指令来实现。而80386是在寄存器中说明指令断点地址，这样可以避免由于向被保护或共享代码中写入断点指令带来的麻烦。设置数据断点（一般微处理器没有这种能力）是一个特别有用的调试手段。例如，通过设置数据断点，程序员能够迅速定位使用了错误数据结构的指令。除断点寄存器外，80386也提供传统的断点指令和单步调试设施。

1.6 目标码兼容性

80386是在86系列中继8086和80286之后的第三代微处理器，并在二进制数（目标码）级保持同它们的兼容性。这一兼容性保护了软件投资，保证了快速投放市场并使它可使用为86系列机所编写的丰富软件库。

当然，80386可以运行8086程序，并能并发执行80386和80286程序。但80386最富创新性的兼容性特色是虚拟86（V86）能力，即在80386多任务框架中建立一个被保护的8086环境。80386分页机制可在80386提供的物理地址空间中的任何区域为每一个虚拟86任务分配一个IMB地址空间。另外，如果80386操作系统支持虚拟存储的话，虚拟86任务能够象其它任务一样被调进而不需要专门处理。一句话，80386虚拟86设施可使86系列中的三代软件同时运行。

1.7 小结

80386为实现那些高目标微处理器系统提供了必要的原始性能。80386具有灵活的体系结构，因而系统设计者可以选择必要的选件以满足特定应用的需要。健全的存储器管理设施，包括分段、分页及虚拟存储设施都是片载可用的。多达四级的保护设施能够在各软件组件间建造“防火墙”，当然根据需要也可以不用保护。已经为商业系统和其它86系列机开发的标准软件，借助于虚拟86任务可以应用于32位系统中。

通过使用其它Intel芯片与80386互连，可以增强80386的能力和适应性。这些Intel芯片有：局部网控制器、高级DMA控制器、磁盘控制器及图形处理器等。

Intel公司还提供了一些开发工具和插件板，用以缩短设计时间并减少花费。所提供的开发工具有：编译程序、连接程序和加载程序、操作系统及在线（in-circuit）仿真器（ICE™386）等。几百个工业标准MULTIBUS®I板可用来完成标准功能而不会引出额外的设计和测试花费；而且高性能MULTIBUS II 板阵列不断涌现出来。最后需要指出的是，在世界范围内，Intel公司经验丰富的应用工程师和专家随时为用户提供设计帮助。

注：原文为Whetstonb，这里是根据上下文而译出。

第二章　应用体系结构

80386为使用汇编语言的应用程序员或编译程序设计者提供很多32位资源。这些资源包括：1）寄存器　2）存储器及逻辑寻址　3）数据类型和指令。本章分三节对它们分别进行描述。

2.1　寄存器

包括80386在内的所有计算机都为程序员提供一组寄存器，作为快速本地存储器。当对驻存在寄存器中的数据进行访问时不需要使用总线周期，从而提高指令执行速度并为其它处理器（如直接存储器存取控制器）腾出较多的总线带宽。80386拥有八个通用寄存器；由80287或80387数字协处理器提供另外八个寄存器；80386还提供两个寄存器，即标志寄存器和指令指针。它们面向处理器控制和状态，而不是用于存储数据。这两个寄存器对程序员来说也是很重要的。

2.1.1　通用寄存器

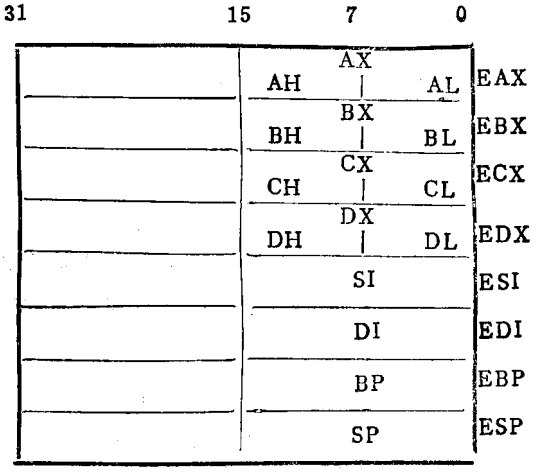


图 2-1　通用寄存器

如图2-1所示，80386的通用寄存器都为32位宽；处理器内部的数据通路、数据总线及地址总线全为32位宽。这样，80386是一个字长为32位的处理器，但按惯例，仍规定80386的字长是16位，而称一个32位的量为一个双字（dword）。

任何一个通用寄存器都可以被当作16位或32位寄存器加以使用，而且其中的四个寄存器还可以当成八个8位的寄存器使用。通用寄存器以操作数的形式可以出现在几乎所有的指令中，例如，两个寄存器可以进行乘法运算。任一寄存器在地址运算时又可用作基址或索引寄存器（本章后面进一步讨论）。每个实用程序都需要一个栈，而栈顶都由通用寄存器ESP隐含地给出。

2.1.2 标志和指令指针

图2-2给出了80386标志寄存器各位的含义。这些标志共分为三类：状态标志、控制标志及系统标志。在许多指令执行之后，处理器都要设置相应的标志位来反映操作结果。例如，当两个操作数进行比较，结果相等，则处理器设置零标志位。又如条件跳步指令，首先测试一个标志位，然后根据不同的值进行不同的动作。这样程序员可通过设置控制标志来修改某些指令的语义。例如搜索串（Scan string）指令就是根据方向标志的值来决定

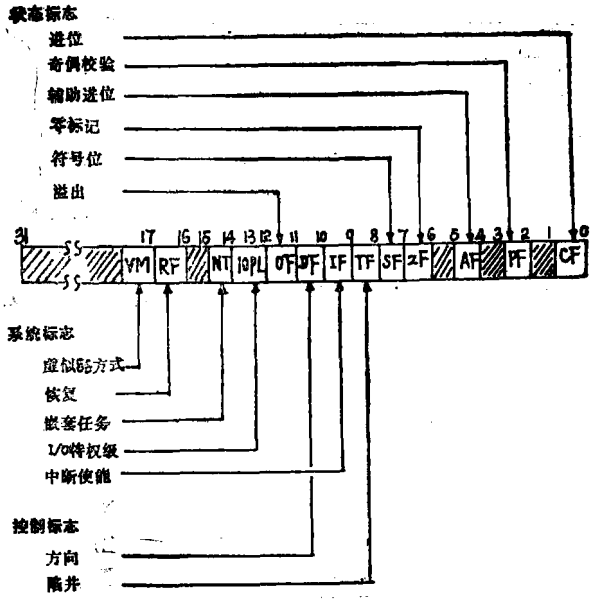


图 2-2 标志寄存器

是向高地址方向搜索，还是向低地址方向搜索的。系统标志是为操作系统设置的，应用程序员可以忽略它们（有关内容在第三章讨论）。其实，80386的保护系统可以防止应用程序有意无意地修改系统标记。

80386的指令指针（EIP）是一32位宽寄存器，它控制指令的取出（包括预取）。处理器执行完一条指令后自动增加指针值。中断、意外及控制转移指令，如跳步指令、子程序调用指令等，都会导致指针值的改变。

2.1.3 数字协处理器寄存器

图2-3给出了数字协处理寄存器格式，当把80287或80387同80386互连时，即可把这些寄存器容入80387系统中，从而提高数字应用的性能。与数字协处理器识别各种长度的整数、压缩十进制数和浮点格式的同时，在内部它把相应的值存储在一个8×80位的浮点寄存器栈中。数字指令一般隐含地引用栈顶元素，或明确指出所引用的寄存器。状态寄存器中有栈顶指针、识别意外（如溢出）的标志、反映上一条指令执行结果的条件码等。控制寄存器中包含选择和屏蔽位，程序员通过设置相应的标志位可选择循环算法、无穷量的模拟方式及决定是由处理器还是由软件处理意外。

此为试读，需要完整PDF请访问：www.ertongbook.com

寄存器栈

79 78 64 63

±	指数	尾数

1 0

控制寄存器
状态寄存器

图 2-3 数字协处理器寄存器

2.2 存储器与逻辑寻址

在80386应用程序中，并没有直接给出操作数在4GB物理地址空间中的位置，而是使用的逻辑地址。处理器将自动把这些逻辑地址转换成物理地址再把它发送至系统总线上。正如将在第三章中详细讨论的那样，80386操作系统能够决定逻辑地址空间呈现在应用程序员面前的面貌。例如，操作系统可把逻辑地址空间定义为一个简单的2^{32}字节阵列（许多体系结构就是这样定义的）。作为选择之一，80386操作系统把逻辑地址空间化分为一组可变长的段。操作系统可以定义很多段，也可以只定义少数几个段；80386并没有限定段的使用方法，这要根据具体应用需要而定。当阅读以下几节时，应牢记：应用程序对段的使用不能超越由操作系统所建立的框架。

2.2.1 段

上面已提到，操作系统把80386逻辑地址空间定义为一个或多个段。注意到程序结构长度是可变的，而段作为一个逻辑部件，能和程序结构建立良好的映射关系。例如，一个长为1516字节的过程，适合使用一个长为1516字节的段；同样一个8M字节阵列（如一个1028×1028×8的显示缓冲区）适合于使用一个同样大小的段。通过提供结构支持（如段可分别进行保护并有选择性地在任务间共享），80386提高了以段作为构造机制的系统的性能。（在第三章中讨论的另一逻辑部件是页，其大小固定，不能很好地映射程序结构，但对操作系统却特别适用，如用于交换功能）。

80386的段长不限，小可为1个字节，大可为4GB字节。操作系统为每个段都设置一个结构定义的描述信息，用于说明段的属性。段的属性包括一个32位基地址及上限（长度）和用于防止错误使用段而设置的保护信息等。由于描述信息由操作系统管理，详细介绍请见第三章。应用程序只能间接处理描述信息，并通过逻辑地址来对段进行访问。

2.2.2 逻辑地址

由于一个程序可能会对多个段进行访问，所以，在80386逻辑地址中必须指明所要访问的段。这样80386逻辑地址由两部分组成：一个16位长段选择器和一个相对所选择段的32位偏移量，见图2-4。处理器把选择器作为一个描述信表（该表由操作系统维护）的索

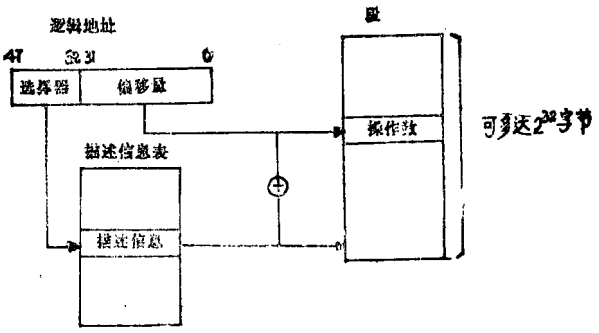
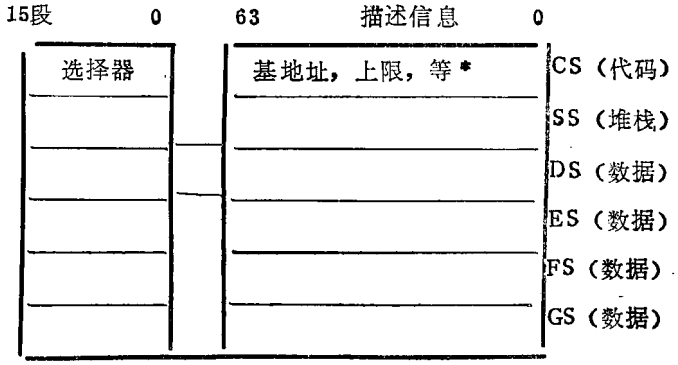


图 2-4 逻辑地址转换

引来获得段的起始地址，然后把偏移量加到该起始地址上，结果就是操作数的有效地址。

2.2.3 段和描述信息寄存器

为了提高逻辑寻址的效率，80386提供了六个段和描述信息寄存器（见图2-5）。实际上，这些寄存器的作用如同一个由程序员控制的超高速缓存，使得在大多数指令中不用特别指明操作数逻辑地址中的选择器部分，从而可以片载进行逻辑地址的转换，而不需要查



* 其它信息域在第三章中描述。

图 2-5 段和描述信息寄存器

阅描述信息表。

大多数程序的地址引用都集中于一个极小地址范围内（这就是使虚存成为可能的“引用局部性”原理）。例如，某个段中的一个过程，在把控制转交给另外一个段中的某个过程前，绝大多数指令都从当前段中取得。根据引用局部性原理，在程序控制下，80386把最近使用的选择器及描述信息存放在片载寄存器中。这样使用片载信息进行逻辑地址转换而不需要存贮器访问时间。

在任何时候，六个段是可寻址的，即代码段、堆栈段和四个数据段。这些段的选择器分别被CS，SS，DS，ES，FS和GS段寄存器给出，相应的描述信息寄存器给出与之匹配的描述信息。需要时，程序只要给一个段寄存器加载一个新的选择器值那可建立一个新的可寻址段。处理器自动维护描述信息寄存器，每当程序改变一个段选择寄存器值时，即相应地加载正确的描述信息（其实，描述信息寄存器只能由处理器加载，而程序不能访问它们）。注意到，指令指针中的值是当前指令在当前代码段（由 CS 寄存器定义）上的偏移量，而 ESP中的值是栈顶在当前堆栈段（由SS寄存器定义）上的偏移量。

为了提高指令译码效率，在大多数指令中没有指明段寄存器，而由80386自动为指令选择一个适当的段寄存器。例如，跳步（Jump）指令选择CS寄存器，而压入堆栈(Push)指令选择SS寄存器。必要时，程序员可在指令前面加一个单字节段超越前缀（Segment override prefix）指明所要访问的段。以后处理器就用这个段翻译指令中的地址。

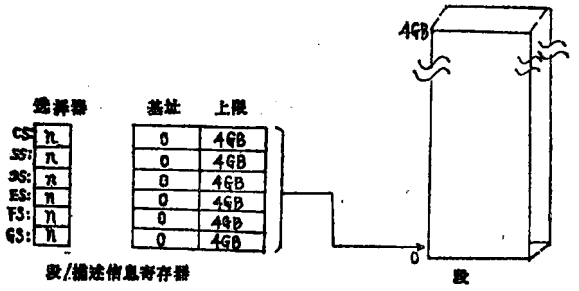


图 2-6 一个4GB逻辑地址空间

一个基地址为0而长度为4GB的段可定义一个4GB的逻辑地址空间。既然处理器自动选择段寄存器，那么仅使用一个32位偏移量，一条指令即可在这一4GB空间中指明其操作数。如图2-6所示，如果所有描述信息寄存器都用0基址和4GB长度加载的话，分段的意义就不存在了。那样，逻辑空间中的每个字节，不管是一条指令、一个变量、还是栈中一的顶都可以用一个32位偏移量进行寻址。这样，段寄器即给出了六个长度为4GB的瞬时可寻址的逻辑地址空间。当这些段一致时，对程序而言该4GB逻辑地址空间与由一个不灵活的32位体系结构给出的地址空间没什么两样。

2.2.4 寻址方式

80386有寄存器和立即寻址方式，分别用于访问寄存器和指令中的操作数。更重要的是80386为访问基于存贮器的数据结构中的元素提供了相应的寻址方式，这些数据结构有：数组、记录（结构）、记录数组及包含数组的记录等。程序使用某种存贮器寻址方式指明一个逻辑地址的偏移量部分。

偏移量：= 基地址 + （索引 * 比例）十位移量

基址、索引和位移量用来计算一个偏移量。基地址和索引值由通用寄存器给出，而位移量值则包含在某一指令中。任一通用寄存器都可以用作一个基址或索引寄存器。索引寄存器中的值与一个比例因子（该因子的值为1,2,4或8）相乘，用于访问多字节数组或记录中的元素。位移量值被处理器翻译成一个8位或32位的有符号的补码值。

基地址、索引及位移量的组合形成如下所列的80386存贮器寻址方式：

- 直接方式：只有位移量
- 寄存器间接方式：只使用基址
- 基址方式：基地址 + 位移量
- 索引方式：索引（比例）
- 具有位移量的索引方式

  索引（比例）+位移量
- 基址索引方式：基地址 + 索引（比例）
- 具有位移量的基址索引方式：

基址＋索引（比例）＋位移量

2.3 数据类型和指令

本节描述应用程序最经常使用的指令。既然多数指令涉及到对一定类型的数据进行操作，那我们就将数据类型和指令在一起描述。将权指令，包括进行I／O和处理中断的特权

表 2-1 基本数据类型和指令

类型	大小	指令
整数， 序数，	8,16,32位	MoVe（传送），Exchange（交换），Translate（转换），Test（测试），Compare(比较)，ConVert（变换），Shift（移位），Double Shift（双移），Rotate（环移），Not（非），Negate（反），And（与），Or（或），ExclusiVe Or（异或），Add（加），Suletract（减），Multinly（乘），DiVide（除），Increment（增量），Decrement（减量），ConVert(MoVe with sign/zero extension)(连同符号或0扩展一起的传送）。
非压缩 十进制数	1个数位(digit)	十进制调整的：Add（加），Subtract（减），Mutinlg（乘），DiVide（除），
压缩十 进制数	2个数位	十进制调整的：Add（加），Subtract（减），
（字节，字， 双字）串	0—4G个字节、 字、双字	MoVe（传送），Load（取），Store（存），Compare（比较），Scan（扫描），Repeat（重复），
位串	0—4G位	Teat（测试），Testandset（测试并置位），Test and Reset（测试并复位），TestandComnlement（测试并求补），Scan(扫描），Inserr（插入），Extract(分离)。
近指针[1]	32位	(同序数)
远指针	48位	Load(加载)

1. 一个近指针是一个相对于由段/描述信息，寄存器对定义的段的32位偏移量。一个远指针是一个全逻辑地址，即一个选择器和一个偏移量。

指令在下面一章中描述。

2.3.1 主要数据类型

表2-1列出了80386支持的部分数据类型和指令。这些指令都是经常使用的，而其中有些指令的变种，如循环移位指令中的循环左移，循环右移及带进位的循环移位指令，表中也没有列出。

图2-7给出了基本数据类型在存储器中的存储方式。多字节项可以定位在任何字节地址上，但根据总线设计，对一个定位地址不是其大小（以字节为单位）整数倍的操作数进行访问时，也许需要额外的总线周期。因而，为防止存取效率受总线设计的影响，大多数程序都把字操作数定位在字边界上、双字操作数定位在双字边界上，依次类推。

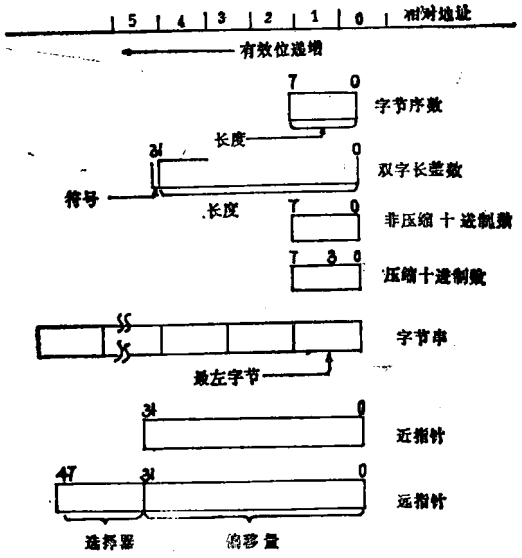


图 2-7 数据类型存储

2.3.2 数字协处理器数据类型

80287或80387数字协处理器为80386提供了表2-2所列的数据类型和指令。大多数数字应用的输入值和输出结果都是整型、实型或压缩十进制数；而临时实型(temporary real)

表 2-2 基本数字协处理器数据类型和指令

类型	大小	指令
整数	16,32,64位	Load（取），Store（存），Compare（比较），Add（加），Substract（减），Multinly（乘），DiVide（除）
压缩十进制数	18个数位	Load（取），Store（存），
实数	32,64位	（同整数）
临时实数	80位	Add,Sulstract,Multiply,Diride Suqare Root(平方根)，Scale Remainder（换算余数），Inter Part（整数部），Change（改变），Sign(符号)，Aksolute Value（绝对值），Extract Exnonent-and Signikicand(分离整数和指数)，Test，Exch-anye，Tangent（正切），Actangent（余切）2^X-1，$Y*\log_2(X+1)$，$Y*\log_2(X)$，Load constane (0.C,pi,etc)（取常数0.C,π等）（80387增加Sine（正弦），Consine（余弦），Sine and Cosine，Unordered Comnare）

主要用于存放中间结果，由于它具有特别高的精度，所以它可以极大地减少在一些复杂计算中常常出现的循环（roundig）、下溢和上溢问题。数字协处理器的主要工作是对寄存

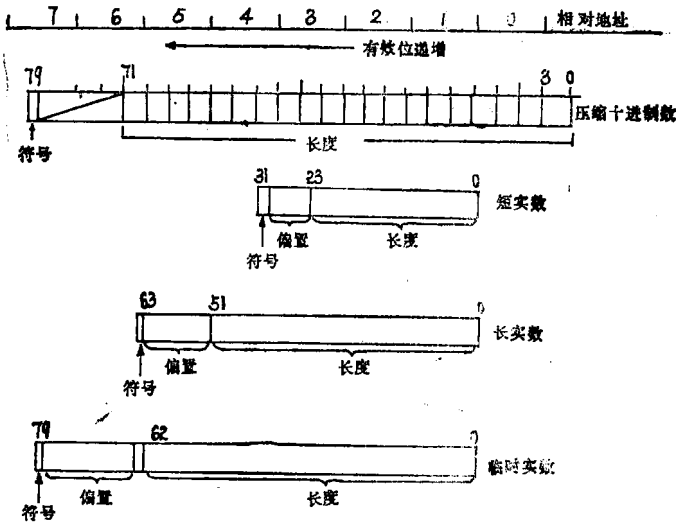


图 2-8　数字协处理器数据类型存储举例

器中临时实数值进行运算。任何类型的数据被取到寄存器栈中时都自动转换为临时实型数；寄存器中的临时实型数通过存储指令（Store）再转换成相应类型的数用于输出。

2.3.3　其它指令

并非所有80386指令都和特定数据类型有关，下面几节就分别描述这些与数据类型无关的指令。

·堆栈指令

80386的一个堆栈是双字长（32位）栈，其基址和栈顶分别由SS和ESP寄存器给出。Push指令把一个双字压入堆栈，而Pop指令从栈中弹出一个双字并送入寄存器或存储器中。Push All指令把所有通用寄存器的内容压入栈中，而Pon All指令执行相反的操作。

Enter和Leave指令是给块结构高级语言提供的。Enter指令建立堆栈结构（stack prame）和显象（disalay），以便编译程序用于连接过程调用；而Leave指令在返回到调用过程前执行相反操作。

·控制转移指令

Jump指令能够改变指令指针值，从而把控制转移给另外一条指令，接受控制的目标指令既可以与Jump指令处于同一代码段中，也可以在另外一个段中。从而可把跳步指令分为段内跳步指令和段间跳步指令，在段内跳步指令中的操作数是一个近指针，既目标指令在当前代码段中的偏移量；在段间跳步指令中的操作数是一个远指针，其中的选择器部分赋予CS寄存器，而把偏移量值赋给EIP寄存器。条件跳步指令与跳步指令大体一样，只是它的分支转移要根据状态标志值而定。

可以用CAll指令来唤醒过程和函数（子程序），在被调用的程序中使用Return指令返回到调用者。同跳步指令一样，段内调用指令中使用的是近指针操作数，只是给指令指针赋予一个新值；而段间调用指令中使用的是远指针操作数，除EIP外，还给CS寄存器赋

此为试读，需要完整PDF请访问：www.ertongbook.com