吕晓庆　编著

80386/486 系统编程实践

浙江大学出版社

80386/486 系统编程实践

吕 晓 庆 编著

浙江大学出版社

(浙)新登字 10 号

80386/486 系统编程实践

吕 晓 庆 编著

责任编辑 俞妙送

浙江大学出版社出版

浙江大学出版社计算机中心电脑排版

浙江省煤田地质局印刷厂印刷

浙江省新华书店发行

开本:787×1092 1/16 印张:18.125 字数:487 千字

1993 年 8 月第 1 版 1993 年 8 月第 1 次印刷

印数:0001—5000

ISBN 7-308-01162-3/TP·080 定价:12.80 元

前 言

自从 Intel 公司推出 32 位微处理器 80386/486 以来，微型计算机的 CPU 已经逐步从 16 位的 8086/80286 微处理器升级到 32 位微处理器 80386/486。在 90 年代，微机市场将是 32 位机的天下，由于 Intel 公司计划未来的微处理器 80586 和 80686 仍以 32 位的 80386 微处理器作为其基本单元，因此，对于微机系统程序员来说，学习和掌握 80386 微处理器的体系结构与编程方法尤为重要。

要学习和掌握一种新的微处理器，光看书不编程实验，肯定收效甚微，最多只能了解一些皮毛。要想精通某种处理器，必须通过反复的编程实验，方能透视其奥妙之处。对于 8086/8088 微处理器来说，其编程实验环境已经很成熟了，只要一台 PC 机、DOS 操作系统、以及一套汇编语言软件即可构成良好的编程环境。然而，对于 80386/486 微处理器来说，其编程实验环境就并非理想，一台 80386/486 PC 机以及一套支持 80386 指令集的汇编语言都很容易找到，但以何种操作系统为基础呢？MS DOS 操作系统是基于 8086 的，不能很好地支持 80386/486 的新特点；基于 80386/486 的 OS/2 或 UNIX、XENIX 操作系统应能满足要求吧，但令人遗憾的是，这些操作系统不会为用户提供 80386/486 特权级（0 级、1 级、2 级）上的编程环境。这样，用户只能在用户级（3 级）上编程，不能对其结构上的新特点进行实验。虽然，目前已经有不少关于 80386/486 微处理器的书籍，但还没有一个良好的实验环境提供给读者，学习起来很费劲，很难真正地掌握细节。

本书的目的，就是为初学者提供一个学习 80386/486 的良好的实验环境，并以此为基础，通过大量的实验，展示 80386/486 微处理器的魅力，同时，初学者能在此环境下实验自己编写的 80386/486 程序，从而迅速地理解和掌握 80386/486 微处理器。

本书是面向那些具有一定的计算机基本理论基础及 8086/8088 汇编语言编程经验的人的，如果读者缺乏这方面的知识，可先阅读一些有关这方面的书籍，然后，再阅读本书。

本书特别适合那些想全面了解 80386/486 的体系结构，并想在其上做系统开发的系统程序员，包括在校计算机专业的本科生与硕士研究生、计算机行业的工程师，以及广大的微电脑爱好者。

本书共由十一章组成：

第一章：认识 80386/486 讲述 80386/486 系列微处理器的各种型号和特点，以及其寄存器、寻址方式、指令格式等。

第二章：实模式讨论 80386/486 微处理器在实模式下的工作情况，以及其新增的指令，如 BT、BTC、BTS、BSF、MOVZX、MOVSX、SHRD、SHLD、SET 等。对于 80386/486 的系统指令，如 LGDT、LIDT、LTR、MOV EAX，CR0 等，将在后面章节中讨论。

第三章：建立新的实验环境将构造一个编程实验环境，用于学习 80386/486 微处理器的体系结构，这个实验环境包括一台以 80386 或 80486 为 CPU 的 PC 机、一套支持 80386 指令的汇编语言软件（如 TURBO ASSEMBLER V1.0 或 MS MASM V5.0 等等）、以及作者本人设计的 V86DOS 系统。本章还描述如何在新的实验环境下进行编程、调试和观察分析。

第四章：保护模式下的内存管理介绍 80386/486 在保护模式下的内存管理技术，主要包括段内存管理、分页内存管理，以及相应的 80386/486 系统指令。为了加深理解，本章安排了多个编程实验。

第五章：保护模式下的异常与中断将通过实验学习 80386/486 在保护模式下的中断和异常处理，重点介绍 80386/486 所产生的各种中断和异常的特点，中断描述符表以及怎样设计系统的中断和异常处理程序。

第六章：多任务讲述 80386/486 在保护模式下的任务管理，主要包括任务状态段（TSS）、任务门以及相应的系统指令和系统寄存器；另外，还研究了多任务系统的设计方法等等。

第七章：保护机制描述了 80386/486 微处理器检测错误的手段。

第八章：虚拟 86 模式介绍 80386/486 的另一执行模式：虚拟 86 模式。这章是本书的重点之一，我们将全面研究虚拟 86 模式下的内存管理、中断和异常处理、虚拟 86 任务管理、I/O 管理以及虚拟 86 模式下的保护机制；同时还详细地介绍了 V86DOS 操作系统的结构和流程。

第九章：调试功能将讨论 80386/486 微处理器所提供的调试功能，包括调试寄存器、相应的系统指令以及调试异常处理程序的设计。

第十章：其它分析 80386/486 微处理器的三种执行模式（实模式、保护模式和虚拟 86 模式）之间的联系；比较 80386 与 80486 微处理器之间的差异，并对 80486 的高速缓存器与协处理器作简要介绍等等。

第十一章：系统设计实例将通过三个基于 80386/486 微处理器的系统编程设计的例子，对前面几章的内容进行综合复习。

附录 A：V86DOS 系统源程序清单给出 V86DOS 系统的所有源程序，供读者参考与修改。

附录 B：80386/486 指令操作数映象图给出 80386/486 微处理器的所有指令译码。

尽管本书提供了良好的实验环境，但是，要想真正掌握 80386/486，还必须通过反复的编程实验来证实自己的想法。相信读者能始终保持浓厚的兴趣，通过艰苦的努力，最终成为 80386 的专家！下列参考书对阅读本书会有所帮助。

[1] *80386 Programmer'S Reference Manual*, Intel Corporation.

[2] *80386 Hardware Reference Manual*, Intel Corporation.

[3]John H. Crawford & Patrick P. Gelsing. *Programming the* 80386.

[4]*80386 Data Sheet*, Intel Corporation, Order No. 231630

[5]*80387 Data Sheet*, Intel Corporation, Order No. 231920

[6]*80486 Programmer's Reference Manual*, Intel Corporation.

[7]*80486 Hardware Reference Manual*, Intel Corporation.

[8]*TURBO Assembler User manual.*

浙江大学吕维雪教授给予我热情的关怀和帮助，并详细地审阅了全书，借此机会表达我由衷的敬意和感谢。感谢京粤汉字电脑技术研究开发中心的领导和同事们对我的支持和帮助，并为我提供了良好的学习和工作环境。感谢廖伟民高级工程师对我的培养，此书的完成也得益于从他那学到的一套工作方法。特别感谢浙江大学出版社的同志们以及我的妻子罗劲洪，他(她)们为本书的录入及编排付出了艰苦的劳动。

谨以此书献给我的父母。

目　录

1 认识80386/486

1.1 Intel 86 系列微处理器简介

近 10 年来个人计算机的发展速度之快真是令人瞠目结舌，这里有很大一部分应归功于几家研制和生产微处理器的厂家和公司的不懈努力，Intel 公司就是其中的一家。自 1978 年推出 8086 微处理器后，该公司又相继推出了 8088、80186、80286、80386DX、80386SX、80386SL、80486、80486SX 等微处理器产品，这些微处理器先后被 IBM 等计算机厂商所采用。今天以 Intel 86 系列微处理器为 CPU 的个人计算机已经成为个人计算机市场的主流产品。而且，自 Intel 公司推出 80486 后，越来越多的计算机厂家开始用它来作为小型机的微处理器。可见 80486 这类通用微处理器已开始向小型机领域渗透了。下面我们将简要介绍一下 Intel 86 系列微处理器的功能。

8086 是 Intel 公司于 1978 年推出的一种 16 位的微处理器，大约含有 29000 个晶体管。它采用 16 位数据线，20 位地址线，所以，其存储器寻址能力为 1MB。

8088 可以说是 8086 的“衍生物”，其指令系统与 8086 完全兼容，其差别只是在硬件上，8088 采用 8 位外部数据线，20 位地址线，其存储器寻址能力为 1MB，可能是因为其性能价格比比较好而被 IBM 采用，推出了著名的 IBM PC 个人计算机。

80186 也是 8086 的改进型，80186 在芯片上增加了时钟发生、中断控制、计时电路、DMA 控制器等电路，另外，还对指令系统有所扩充，例如，增加了 PUSHA、POPA、ENTER 和 LEAVE 等指令。

在 1982 年，Intel 公司推出了 80286 微处理器。80286 也是 16 位的微处理器，但是它在存储器管理与保护方面比 8086 大大增加，它采用 24 位地址，其存储器寻址能力已达到 16MB。在硬件上支持多任务，这是 80286 的一大突破，其指令系统是 8086、80186 的超集，且支持实模式与保护模式两种程序执行模式。

到了 1985 年，80386 微处理器终于问世。它的功能完善，远远超出了 80286 和 8086 微处理器。80386 共使用约 275000 个晶体管，80386 是 32 位的微处理器，采用 32 位地址线，其存储器寻址能力达到了 4GB。在存储器管理方面，除了保留段内存管理外，还增加了分页内存管理；另外，还支持实模式、保护模式和虚拟 86 模式三种程序执行模式，具有模拟 8086、80286 任务的能力，其指令系统是 80286 的超集。另外，在硬件上还增加了硬件调试功能。

80386 SX 其实是 80386 的“衍生物”，就像 8086 和 8088 的关系一样。80386 SX 采用 16 位的数据线，其性能价格比较好，许多个人计算机厂家用它作为 CPU，生产低档的 386 PC 与 286 PC 竞争。

80186 微处理器于 1989 年推出。它由三个部件组成：一个主处理器、一个数字协处理器和一个带有 8KB 数据 CACHE 的 CACHE 控制器。其中主处理器采用 80386 的体系结构，数字协处理器与 80387 DX 芯片兼容。8KB 数据 CACHE 的目的是减少处理器的等待状态，当微处理器访问内存时，

它首先在8KB的数据Cache中寻找，其静态Cache RAM的响应时间小于25ns。80486的芯片内含有约120万晶体管，是80386的4倍。80486也是32位数据线、32位地址线，其指令系统保持与80386兼容，实际上下列等式应该成立：

80486＝80386＋80387＋80385 Cache controller

80486 SX实际上也是80486的“衍生物”。它通过去掉数字协处理器来获得较好的性能价格比。

由于80486的主处理器采用了80386的体系结构，因此，80486与80386在软件上的差异是微乎其微的。主要表现在：

- 486增加了新指令去控制内部Cache的初始化。
- 在保护模式下，数字协处理器的某些标志不同。

本书主要注重于80386/486主处理器的系统程序设计，至于数字协处理器的结构与程序设计，若读者感兴趣可阅读有关参考书籍。鉴于此：在本书中为了简单起见，除了涉及到80486的新指令和新功能部分外，将只用80386的字样代替80386/486。

1.2 寄存器

80386寄存器大致分为五个部分：通用寄存器、状态寄存器、段寄存器、控制寄存器和调试寄存器，如图1.1所示。前三类寄存器可供系统和应用程序员使用，我们在下面着重介绍；而后两类寄存器仅供系统程序员使用，控制寄存器用于支持内存管理，将在第四章介绍；调试寄存器用于程序调试，将在第九章叙述。

为了与86系列微处理器兼容，80386的寄存器被设计成86系列早期微处理器的超集，其32位寄存器名只是在原有16位寄存器名前加上前缀“E”。

1.2.1 通用寄存器

80386共有8个32位通用寄存器，它们是EAX、EBX、ECX、EDX、ESP、EBP、ESI和EDI，其中这些32位寄存器又包含有8个16位寄存器(AX、BX、CX、DX、SP、BP、SI和DI)，以及8个8位寄存器(AL、AH、BL、BH、CL、CH、DL和DH)。通用寄存器主要用于算术运算、逻辑运算，以及在内存寻址时用作基址和索引寄存器。使用32位寄存器可大大地方便编程，这在以后的编程中读者将会体验到。

1.2.2 状态寄存器

状态寄存器由指令指针寄存器和标志寄存器组成。指令指针寄存器用于标识当前处理器所执行程序指令的位置。在80386中，32位的指令指针寄存器EIP可以指向内存的任意位置；另外，考虑到兼容性，EIP寄存器中又包含了一个16位的IP寄存器，IP寄存器可用于标识的16位指令代码的位置。

这里提到了两个非常重要的概念：16位指令和32位指令。在86系列微处理器中，对内存的引用(memory reference)由段和偏移量两部分构成。16位的指令代码段使用16位偏移量，用IP寄存器指向，其最大段尺寸为64KB；而32位的指令代码段使用32位偏移量，使用EIP寄存器指向，其最大段尺寸为4GB。

笔者在刚开始学习80386时，因为没有重视这两个概念的区别，走了不少弯路，故此特别提醒读者，时刻牢记到了80386时代，已经有了16位指令和32位指令代码的区别！

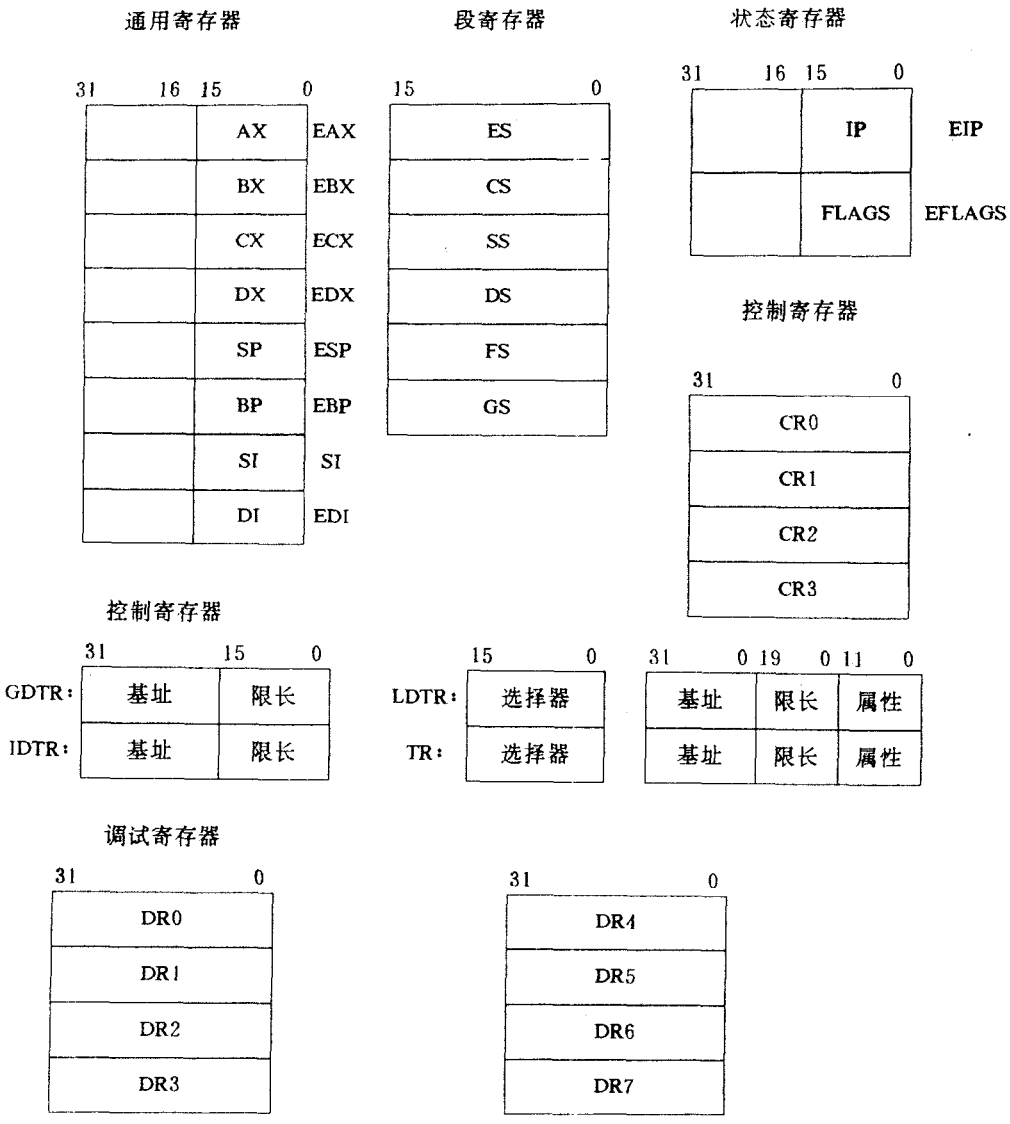

图1.1 80386寄存器结构

那么，如何区分一段指令代码是16位指令还是32位指令呢？在实模式和虚拟86模式下执行的程序代码肯定是16位指令代码；而在保护模式下执行的程序代码既可以是16位指令代码也可以为32位指令代码，我们将在1.3节和第四章中论述。

标志寄存器用于标识当前微处理器的自身状态。32位的EFLAGS寄存器主要由状态标志位和控制标志位这两类标志位组成。图1.2列出了EFLAGS寄存器的标志位分布。

状态标志位是由微处理器自身设置；而控制标志位则是由程序来设置，用于控制微处理器的某一功能操作。下面我们分别描述各标志位：

1. 状态标志位

CF、PF、AF、ZF、SF和OF位是由大多数算术和逻辑指令所设置。

· **CF位**： 进位标志(Carry Flag)。当某一算术操作需要进位或置位时，处理器将CF位置1，否则，CF被清零。

· **PF位**： 奇偶标志(Parity Flag)。假如某一操作结果的低八位中含有偶数个1，则该位置1，

80386的EFLAGS：

31	18	17	16	15	14	13	12										0
00000000000000		VM	RF	0	NT	IOPL		OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

80486的EFLAGS：

31	18	17	16	15	14	13	12										0
0000000000000	AC	VM	RF	0	NT	IOPL		OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

图1.2 EFLAGS寄存器

否则，PF 位被清零。

· AF 位： 辅助进位标志(Auxiliary Carry Flag)。在进行加减法运算时，若第 3 位有进位或借位，则 AF 置 1，这个标志只供 BCD 算术运算使用。

· ZF 位： 零标志(Zero Flag)。若处理器运算结果为零，则 ZF 位置 1，否则，ZF 位被清零。

· SF 位： 符号标志。此标志位用于标识处理器运算结果是负还是正，若结果为负，则 SF 位置 1，否则，SF 位被清零。

· OF 位： 溢出标志(Overflow Flag)。假如处理器在运算过程中，发生溢出，则 OF 位置 1。例如，加法时，若有一个进位进位到最高位时，OF 标志位被置 1。

2. 控制标志位

标志寄存器中的 TF、IF、DF、IOPL、NT、RF 和 VM 位都是控制标志位，它们都可以由程序来设置，下面先作简要的介绍，至于它们的细节将在以后各章中详细的说明。

· DF 位： 方向标志(Direction Flag)。用于串处理指令，假如该标志位清零，则串处理从串最低地址的第一个元素开始；否则从最高地址开始，向低地址进行。

· IF 位： 中断允许标志(Interrupt Enable Flag)。假如该标志置 1，则处理器可以响应某种类型的中断(可屏蔽中断)，若 IF 位为零，则处理器将不理会可屏蔽中断。

· TF 位： 陷阱标志(Trap Enable Flag)。假如该标志置 1，则在每条指令执行后产生一次调试中断，此标志位主要用于程序的调试。

· IOPL 域： I/O 优先级域，由两位组成。IOPL 域规定了请求执行 I/O 操作指令的优先级，对 IOPL 域的详细描述见第七章和第八章。

· NT 位： 嵌套任务位(Nest Task)。用于控制 IRET 指令的操作。如果 NT 位为零，则执行中断的正常返回；如果 NT＝1，则进行任务切换，对 NT 位的详细介绍见第六章。

· VM 位： 虚拟 86 模式位(Virtual 8086 Mode)。若置位，则微处理器将工作在虚拟 86 模式下；否则处理器既可能工作在实模式下，也可能工作在保护模式下，详见第八章。

· RF 位： 重启动标志(Reset Flag)。用于处理器的程序调试。详见第九章。

· AC 位： 对齐检查(Alignment Check)，是 80486 新增加的位。设置 AC 位和 CR0 寄存器的 AM 位后，将激活访问内存的对齐检查。

上面的标志寄存器的控制标志位中，IOPL、NT、VM 和 RF 位是 80386 处理器新增的，8086 微处理器所没有的。因此，对于熟悉 8086 微处理器的人，在 80386 上编程时，应特别注意对这些标志位的处理，如果对它们的设置出现错误，将导致系统的崩溃。另外，微处理器的控制标志位的设置与程序所处的优先级有关。例如，VM 位与 IOPL 域仅能由执行在优先级 0 级上的程序来改变，而 IF 的

设置却与 IOPL 域的值及优先级有关。

1.2.3 段寄存器

正像我们大家所知，86 系列微处理器均使用一种分段的内存管理技术。它将内存空间分成多个段，要寻址某一内存空间位置，需要两部分，即段：偏移量。段寄存器就是用来保存段值的。到了 80386 时代，除了段寄存器 CS、SS、DS、ES 外，新增加了 FS 和 GS 段寄存器，共支持 6 个段寄存器。段寄存器值的定义又与程序在哪一种模式下执行有关，在实模式和虚拟 86 模式下，段寄存器的值为 16 位段地址，与 8086 微处理器兼容；但在保护模式下，段寄存器的内容有了新的定义，引入了选择器(Selector)的概念，详见第四章。

下面对各段寄存器的作用作简要的描述：

CS 是代码段(Code Segment)寄存器，常常用于对指令代码的寻址。CS：EIP 构成了所执行的下一条指令的地址。

SS 是堆栈段(Stack Segment)寄存器，常用于堆栈寻址，SS：ESP 构成了程序的堆栈指针，SS 也用于对数据的寻址。

DS、ES、FS 和 GS 段寄存器用于对程序的数据进行寻址。其中，DS 为主要的数据段寄存器，ES、FS 和 GS 为辅助段寄存器。一般地，如果某程序的数据被经常引用，则应用 DS 作为其段寄存器。

1.3 寻址方式

我们知道，86 系列微处理器的内存寻址需要有段和偏移量，我们这里所说的寻址方式主要针对偏移量而言。

与 86 系列的早期微处理器相比，80386 提供了更加灵活的寻址方式。在 80386 中，偏移量一般由三个部分构成，它们是基址寄存器(Base Register)，索引寄存器(Index Register)乘以某一比例因子(1、2、4 或 8)，以及一个常数的位移量(Displacement)。

在 80386 中，基址寄存器可以是任何一个 32 位的通用寄存器；而在 8086 微处理器中，只有 BX 和 BP 寄存器才充当此角色。索引寄存器可以是除了 ESP 外的任何一个 32 位通用寄存器；而在 8086 中只有 SI 和 DI 才担当此角色。下面列出了各种可用的寻址方式。

偏移量 = 基址 + (索引 × 比例因子) + 位移量

基址：EAX、EBX、ECX、EDX、EBP、ESP、EDI、ESI。

索引：EAX、EBX、ECX、EDX、EBP、ESI、EDI。

比例因子：1、2、4、8。

位移量：8 位立即数、32 位立即数。

80386 的这种“基址+(索引×比例因子)+位移量”的寻址方式，是非常灵活有效的，它可以满足对各种不同的数据结构的寻址。

1.4 指令格式与译码

除非要设计汇编器(Assemblers)或调试器(Debugger)，我们没有必要详细了解每一条指令对应的机器码以及其译码方法，有了汇编语言(Assembly Language)，就足以进行程序设计了。然而，学习并掌握一点 80386 指令的格式，对编程是会有不少好处的。本节首先简要地介绍一下指令的结构，

然后，我们将具体分析 80386 指令格式的特点。

图 1.3 显示了 80386 指令的通用格式，每一条指令最多可由 5 个域组成。

·前缀字节：

一条指令最多可以规定 4 个前缀字节。指令前缀位于指令的头几个字节，它仅仅改变此指令的翻译，而不会影响到其它的指令。

·操作码字节(Opcode Bytes)：

指令的操作码字节紧跟着前缀字节。在每一条指令中，至少应有一个操作码字节，某些指令需要 2 个操作码字节，两个字节的操作码第一字节为 0FH。

· MODRM 操作数标识符(MODRM Operand Specifier)：

MODRM 紧跟着指令操作码之后，它包含下面的信息：

(1). 指令中所用的变址类型或寄存器数；

(2). 使用寄存器或选择指令的详细信息；

(3). 基址、变址、比例因子信息。

·地址位移量(Address Displacement)：

它位于 MODRM 域之后，MODRM 域的子域 mod 决定指令有无此位移量以及它的长度，地址位移量只出现于有内存操作数的指令。

·立即数(Immediate Constant)：

它位于指令的末端，操作码决定了此域是否出现以及其长度。

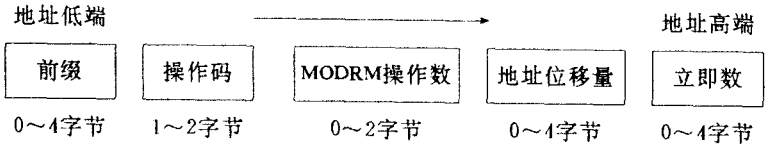

图1.3 指令的一般形式

在 1.2.2 节中，我们向大家介绍了 16 位指令与 32 位指令，那么，它们对应的指令译码格式有何关系呢？先看下面的例子，在 80386 中，16 位指令 POP AX 和 32 位指令 POP EAX 都对应同一个机器码 58H，这说明到了 80386 时代，指令译码与指令已不再是简单的一一对应关系了，不同指令的译码可能是相同的。因此，在使用调试工具对某一程序进行反汇编时，必须明确所反汇编的程序是 16 位指令代码还是 32 位指令代码，否则，会产生混乱。下面的一段程序的机器码就可以有两种翻译：

机器码	16 位指令助记符	32 位指令助记符
50	PUSH AX	PUSH EAX
53	PUSH BX	PUSH EBX
51	PUSH CX	PUSH ECX
52	PUSH DX	PUSH EDX
89 D8	MOV AX,BX	MOV EAX,EBX
5A	POP DX	POP EDX
59	POP CX	POP ECX
5B	POP BX	POP EBX
58	POP AX	POP EAX
CF	IRET	IRETD