80486/80386 系统原理与接口大全(上)

——80386系统原理

艾德才 陆 明 李文彬 编著

清华大学出版社

80486/80386
系统原理与接口大全(上)
——80386 系统原理

艾德才　陆明　李文彬　编著

清华大学出版社

(京)新登字 158 号

内 容 提 要

本书由上、中、下三册组成，分别为 80386 系统原理、接口技术、80486 系统原理。

上册详细介绍了 80386 CPU、数值协同处理器 80387、高速缓冲存储器 Cache 及其控制器 82385、高性能 DMA 控制器 82380、图形协同处理器 82786 以及存储管理和虚拟存储管理等结构和原理，同时也专门讨论了与其系列(低档)机兼容的有关问题。

本系列书内容全面、实用，是 80386、80486、接口技术方面的工具书。

本书可作为计算机培训教材、高等学校教学参考用书，也可供计算机应用人员用作工具书。

图书在版编目(CIP)数据

80486/80386 系统原理与接口大全 (上)：80386 系统原理/艾德才等编著. —北京：清华大学出版社，1995
ISBN 7-302-01856-1

Ⅰ. 80… Ⅱ. 艾… Ⅲ. 微型计算机，80386-接口 Ⅳ. TP 364

中国版本图书馆 CIP 数据核字(96)第 06563 号

出 版 者：清华大学出版社(北京清华大学校内，邮编 100084)
责任编辑：焦金生
印 刷 者：清华园印刷厂
发 行 者：新华书店总店北京科技发行所
开 本：787×1092 1/16 印张：16.75 字数：396 千字
版 次：1995 年 8 月 第 1 版 1995 年 8 月 第 1 次印刷
书 号：ISBN 7-302-01856-1/TP・836
印 数：0001—8000
定 价：16.00 元

前 言

90 年代是微型计算机技术空前发展的 10 年。90 年代是 32 位微型计算机时代。32 位微型计算机是当今微型机的杰出代表。不论是速度还是性能，32 位微型计算机系统都超过了 70 年代末期、80 年代初期中小型计算机的水平。因为 32 位微型机已显露出前辈机所没有的机器视觉、人工智能和声音识别的特征，所以它在专家系统、机器人以及控制系统、工程工作站、办公室自动化、事务处理、科学计算和工程计算、人工智能、过程控制、软件开发、CAD/CAM、公共服务、教育和训练等各方面的应用潜力已初见端倪。在本世纪内，32 位微型计算机将会占领整个微型机市场。由于 80386、80486 是当代 32 位微处理机中的杰出代表之一，微型计算机的主流产品——膝上机的 CPU 将会全部采用 80386、80486 甚至是 80586。为紧紧盯住 90 年代微型计算机的潮流，紧紧盯住国际微型计算机潮流，我们不失时机地编写本书，奉献给急需了解 80386、80486 以及微型计算机接口的广大用户和科技人员。

本书共分三册。上册描述的是 80386 系统原理。围绕 80386 系统原理，主要介绍了 80386CPU、存储管理方式、数值协同处理器 80387、高性能的 DMA 控制器 82380、高速缓冲存储器 Cache 及其控制器 82385、图形协同处理器 82786 等外围芯片的体系结构和工作原理等。同时也列专题讨论了与其系列机兼容的有关问题。

中册描述的是与我们用户密切相关的微处理机外围设备及其接口。其中包括我们用户须臾离不开的键盘、显示器、打印机、磁盘等看得见摸得着的外围设备。同时也对支持显示器工作的 MDA、CGA、MCGA、EGA、VGA、8514/A 等适配器以及各种视频服务器原理和功能给以剖析解释。对 80386、80486 常采用的 Multibus Ⅰ、Multibus Ⅱ 总线以及 IBM 的最新专利技术微通道都作了详尽剖析。与高级语言接口、与 DOS 接口等也给以适当说明。

下册介绍 80486 系统。80486 是由提高了效率的 80386 微处理机、增强了性能的 80387 数值协同处理器、一个完整的 Cache 及其控制器组合而成的。根据其特征，下册着意对 80486 有别于 80386 的部分给以论述，像对 80486CPU、流水线操作原理、支持部件等给以独到论述。对 80486 特有的片内 Cache、二级 Cache 都给以详尽论述，对有别于 80386 的存储管理、总线、自测试等方面也进行了有针对性的论述。也就是说，对 80486 的论述已不再包括与 80386 相同的部分。

本书力求全面、具体、丰富实用，力求成为一本有关 80386、80486 及其接口的微型计算机系统大全和工具书。在编写过程中，得到刘晓月、陈毓弘、苗君秋、胡琳、周士松、孙丙国、刘秀云、程家莲等同志的帮助，在此深表谢意。

由于编者水平所限，时间仓促，新技术新词汇难于仔细推敲，书中不足和谬误在所难免，敬请计算机界老前辈、同仁和读者不吝指正。

编 者

1995 年 1 月于天津大学

目 录

第1章 80386基本体系结构

微处理机的不同应用需要不同的结构支持。对于某些应用(例如,在 Berkely UNIX 环境下运行的各种应用)可能需要线性地址空间,而对那些要管理大批动态数据结构的应用可能要求有由硬件实施法则,以保护其动态生成目标的可见性。80386 的体系结构为用户提供一组 32 位通用寄存器。它们在使用时不受任何限制,既可用来进行数值计算,又可用它形成存储器地址。80386 体系结构还给用户提供几种存储器管理和寻址方式,以满足不同的要求。此外,80386 还提供多种寻址方式、数据类型、指令以及某些特殊的结构,便于高级语言实施。

80386 基本体系结构包括寄存器模型、数据类型、寻址方式和指令系统,它形成了高级语言编译代码的生成基础,同时也是汇编语言应用程序的设计基础。

1.1 寄 存 器

80386 总共有 34 个寄存器,按其功能可分成以下几类:

通用寄存器(General-Purpose Register)

段寄存器(Segment Register)

状态和控制寄存器(Status and Control Register)

系统地址寄存器(System Address Register)

调试寄存器(Debug Register)

测试寄存器(Test Register)

1.1.1 通用寄存器(General-Purpose Register)

8 个通用寄存器是 8086 和 80286 寄存器的超集。它们的名字和用途分别为:

EAX 通常用作累加寄存器(Accumulator)

EBX 通常用作基址寄存器(Base)

ECX 通常用来记数(Count)

EDX 通常用来存放数据(Data)

ESP 通常用作堆栈指针(Stack Pointer)

EBP 通常用作基址指针(Base Pointer)

ESI 通常用作源变址(Source Index)

EDI 通常用作目标变址(Destination Index)

8 个通用寄存器中通常保存 32 位数据,但为进行 16 位的操作并提供与 Intel 系列 16

位机兼容，它们的低位部分被当成 8 个 16 位的寄存器。为了支持 8 位的操作还可进一步把 EAX、EBX、ECX 和 EDX 寄存器低位部分的 16 位，再分成 8 位一组的高位字节和低位字节两部分(见图 1.1)，作为 8 个 8 位寄存器。这 8 个寄存器分别被命名为 AH、BH、CH、DH 和 AL、BL、CL、DL。对 8 位或 16 位寄存器的操作只影响相应的寄存器。例如，在做 8 位加法运算时，位 7 的进位并不传送给目的寄存器的位 9，而且把在标志寄存器中的进位标志(CF)相应地置数。总之，这 8 个通用寄存器既可以支持 1 位、8 位、16 位和 32 位数据运算，也支持 16 位和 32 位存储器寻址。

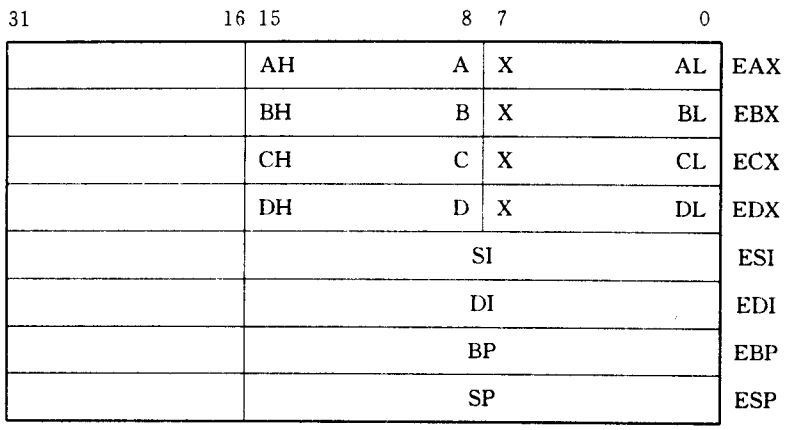

图 1.1　80386 的通用寄存器

1.1.2　段寄存器(Segment register)

80386 有 6 个 16 位段寄存器(见图 1.2)，也称选择符(selector)，它们的名字和用途如下：

CS　代码段寄存器(Code Segment)
DS　数据段寄存器(Data Segment)
SS　堆栈段寄存器(Stack Segment)
ES　附加数据段寄存器(Extra Data segment)
FS　附加数据段寄存器(Extra Data segment)
GS　附加数据段寄存器(Extra Data segment)

(程序员还可给 FS 和 GS 寄存器起恰当的缩写名)

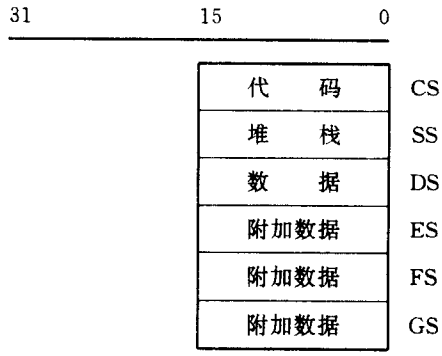

图 1.2　80386 段寄存器

6 个 16 位的段寄存器实现存储空间的分段。所谓段即是把 64T(兆兆)字节存储空间分成各自独立的逻辑地址空间。这样每个程序同时可有 6 个逻辑地址空间供交换用，其中包括 4 个独立的数据空间。

6 个 16 位的段寄存器选择各自的存储区域，其中 CS、DS 和 SS 这 3 个段寄存器给当前的代码、数据和堆栈段寻址。剩下的 3 个段寄存器给用户定义的数据区域寻址。所以在段寄存器内给当前可访问存储单元保存选择符的值。对于实地址方式，段的大小是从 1 个字节到 64K 字节。而对于保护方式寻址，可允许段的大小从 1 个字节到 4G(千兆)字节。

在实方式时，80386 段寄存器内有指定段的实际基地址(见图 1.3)。在保护方式下，段寄存器内保存着 16 位的地址。这个 16 位地址就是通常所说的选择符(selector)。由它指示描述符表中的某一项。实际的段基地址被插在描述符表中，如图 1.4 所示。

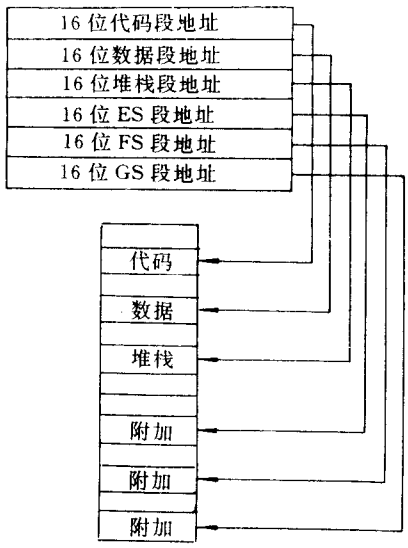

图 1.3　实方式下段寻址

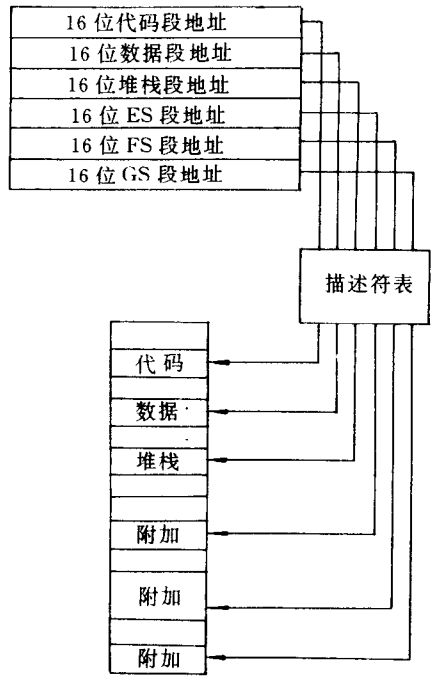

图 1.4　保护方式下段寻址

正在执行的程序代码都驻留在存储器内，而由代码段寄存器 CS 对程序代码进行寻址。现役数据段的基地址由数据段寄存器寻址。在堆栈段寄存器 SS 内保存现役堆栈段的当前基地址。因为通常都是用堆栈存放中间结果，在调用子程序时，还要给出它们自己的存储器段，所以在堆栈段寄存器 SS 内一定要保存好现役堆栈段当前基地址。当然，程序设计人员借助于附加数据段寄存器 ES，还可以访问其他段。通过 ES、FS 和 GS 这 3 个段寄存器，80386 可以同时对 3 个现役数据段寻址。

1.1.3　段描述符寄存器

对程序员来说，段描述符寄存器是不可见的，然而了解它的存在和作用却非常有益。在 80386 的内部，描述符寄存器与可见的各个段寄存器是相联的，如图 1.5 所示。每个描述符寄存器中保存 32 位的段基地址、32 位的段界限和其他属性。

当一个选择符的值装入段寄存器时，相关描述符寄存器的内容就自动地修改成正确的信息。在实地址方式中，只有基地址被直接修改(把选择符的值左移 4 位即可)，因为在实方式中最大段限和段属性是固定的。在保护方式中，基地址、段界限和属性全部按照每个选择符指引的段描述符内容进行修改。

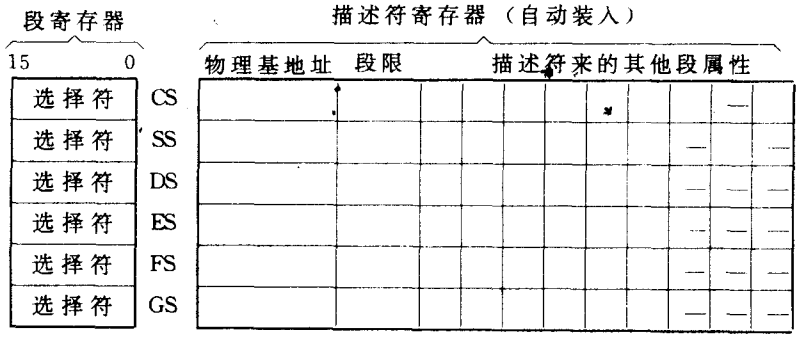

图 1.5　80386段寄存器和有关描述符寄存器

当访问存储器时，所有与其相关的段描述符寄存器自动参与该次存储器的访问操作。32 位段基地址成为线性地址计算中的一个分量，32 位界限用于界检验操作，对照要求的存储器访问类型检验属性。

1.1.4　状态和控制寄存器(status and coutrol register)

状态和控制寄存器是由标志寄存器 EFLAGS (Flag Register)、指令指针 EIP (instuction pointer)和 4 个控制寄存器 CR0～CR3 (Control Register)组成，如图 1.6 所示。

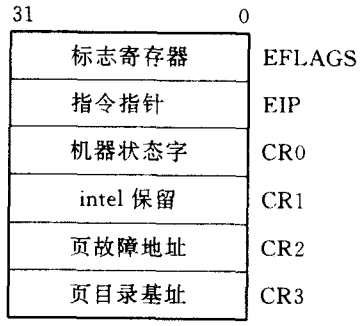

图 1.6　状态和控制寄存器

指令指针寄存器 EIP 中存放下一条要执行指令的偏移量(Offset)，这个偏移量是相对目前正在运行的代码段寄存器 CS 而言的。偏移量加上当前段的地址，形成了下一条指令的地址。EIP 中的低 16 位可以分开来进行访问，给它起名叫作指令指针 IP (Instruction Pointer)寄存器，用于 16 位寻址。

标志寄存器 EFLAGS 存放有关微处理机的信息(如图 1.7 所示)。标志寄存器中的第 1、3、5、15 位及 18～31 位都没有定义。但是，当使用 SAHF 指令或 PUSHF 指令时，或当出现中断处理时，除位 1 外，其余位全部是 0。

第 0 位 CF(Carry Flag)是进位标志位，在算术运算时如果产生进位或借位，则把 CF 位置 1，否则把 CF 位置 0。在执行移位和环移指令时也要用到 CF。CF 内保存从寄存器移出或环移出的那一位。

第 2 位 PF (Parity Flag)是奇偶校验标志位，主要是用在数据通信上。如果是奇数奇偶校验，把 PF 置 1；如果是偶数奇偶校验把 PF 置 0。

第 4 位 AF (Auxiliary Carry Flag)是辅助进位标志。在用 BCD(二—十进制)进行算

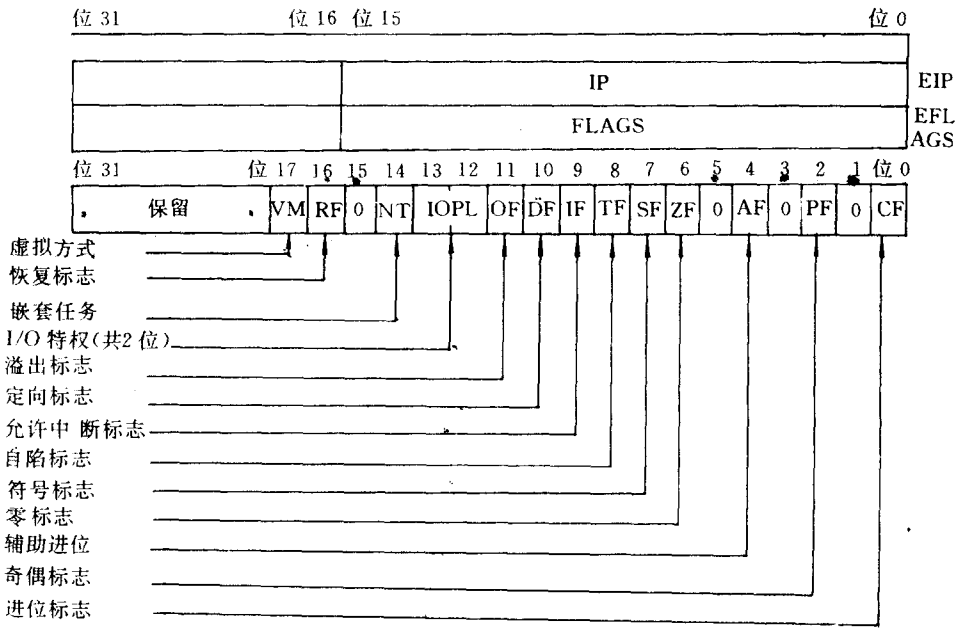

图 1.7　80386 指令指针寄存器 EIP 和标志寄存器 EFLAGS

注：0 表示没有使用。

术运算时，用第 4 位 AF 表示进位输出是否进入到 BCD 最低四位，或者是否到 BCD 最低四位借位。

第 6 位 ZF(Zero Flag)是零标志位，当结果为零时则将其置成 1，否则将其置成零。

第 7 位 SF(Sign Flag)是符号标志位。当结果为负时则将其置成 1，结果为正时则将其置成 0。

第 8 位 TF(Trap Flag)是自陷标志位，当将其置成 1 时则可以进行单步执行。当指令执行完后，就有可能生成异常事故 1 的自陷。也就是说，在程序执行过程中，每执行完一条指令，都要由异常事故 1 处理程序进行检验。当将第 8 位 TF 清 0 后，且当将断点地址装入调试寄存器 DR0—DR3 时，才会产生异常事故 1 的自陷。

第 9 位 IF(Interrupt Flag)是中断标志位，是用来表示允许或者禁止某些外部中断。若第 9 位 IF 被置成 1，则允许 CPU 认定外部中断请求信号；若将 IF 位清成 0，则表示禁止外部中断请求。只有当第 12、13 位(输入输出特权位)指出当前最高特权值 CPL，才允许将新值置入标志寄存器 EFLAGS 时再改变 IF 位的值。

第 10 位 DF(Direction Flag)是定向标志。DF 位规定了在执行串操作过程中，对源变址寄存器 ESI 或目标变址寄存器 EDI 是增值还是减值。如果 DF 值为 1，则寄存器减值；若 DF 值为 0，则寄存器值增加。

第 11 位 OF(Overflow Flag)是溢出标志位，用它表示运算时出现的进位进入了结果的高序位，可高序位却没有进位输出，或是高序位并没有接收进位输入却产生了进位输

出。对带符号的操作数进行运算时，若运算结果一旦超出了数可表示的范围，OF 位就被置成 1。否则将其清成 0。

第 12、13 位 IOPL 是输入输出特权级位，在保护方式下常要使用这两位标志。由于输入输出特权级标志共两位，它的取值范围只可能是 0、1、2 和 3 共 4 个值，恰好与输入输出特权级 0～3 级相对应。在当前任务的特权级 CPL 高于或等于输入输出特权级时，就可以执行像 IN、OUT、INS、OUTS、STI、CLI 和 LOCK 等指令而不会产生异常事故 13(中断号 13)。在当前任务特权级 CPL＝0 时，上托标志出栈指令 POPF 或中断返回指令 IRET 可以改变 IOPL 字段的值。当新的标志信息进入任务的 TSS 时，任务切换操作也可以改变 IOPL 的值。

第 14 位 NT 是嵌套任务标志位。在保护方式下常使用这个标志。当 80386 在发生中断时和执行 CALL 指令时就有可能会引起任务切换。若是由于中断或由于执行 CALL 指令而出现了任务切换，则将嵌套任务标志位 NT 置 1。若没有出现任务切换，则将 NT 位清 0。很明显，若将 NT 位置成 1，则说明欲执行的任务是嵌套在前一项任务之中的。同时还必须指出当前嵌套任务的任务状态段 TSS 和有一条返回前一项任务的 TSS 的有效途径。标志寄存器 EFLAGS 的嵌套任务标志位 NT 的值由中断返回指令 IRET 给以测定，以决定是进行同任务内的返回还是不同任务间的返回。上托标志出栈指令 POPF 或中断返回指令 IRET 会根据从堆栈弹出来的信息，在任何特权级下都能给 NT 位置值。

第 16 位 RF(Resume Flag)是恢复标志位，也是 80386 新增添的一位标志位。当每条指令成功地执行完后，都会自动地将 RF 位置成 0，但中断返回指令 IREF、上托标志出栈指令 POPF 和实现任务转换的指令除外。RF 标志与调试寄存器一起用于断点和单步操作，当 RF 置成 1 时，下一条指令的所有调试故障全被忽略。

第 17 位 VM(Virtual 8086 Mode Flag)是虚拟 8086 方式标志位，是 80386 新设置的一个标志位。表示 80386 微处理机是在虚拟的 8086 环境中运行。如果 80386 微处理机是在保护方式下，而 VM 位又被置成 1，这时 80386 就转换成虚拟 8086 操作方式，使全部段执行操作就像是在 8086 微处理机上运行一样。VM 位只能由两种方式中的一种方式给以设置，或者是在保护方式下，由最高特权级(0)级代码段的中断返回指令 IRET 设置，或者是由任务转换进行设置。

80386 还定义了由 4 个 32 位的控制寄存器 CR0—CR3 组成一个寄存器组，如图 1.8 所示。

在这几个寄存器中保存全局性和任务无关的机器状态，这些寄存器连同后面将要说明的系统地址寄存器(System Address Register)一起，保存影响系统中所有任务的机器状态。访问控制寄存器时，要用存、取这一类传送指令，例如 MOV CR0 指令。

控制寄存器 CR0 包含有 6 个预定义标志，表示和控制机器的状态。其中 0—15 位叫作机器状态字 MSW(Machine Status Word)，这使得 80386 在保护方式下能与 80286 兼容。

控制寄存器 CR0 的 0 位是允许保护位 PE (Protection Enable)，用于启动微处理机的保护方式。如果 PE 置 1，保护方式启动；如果 PE 置 0，则微处理机在实地址方式下运行。

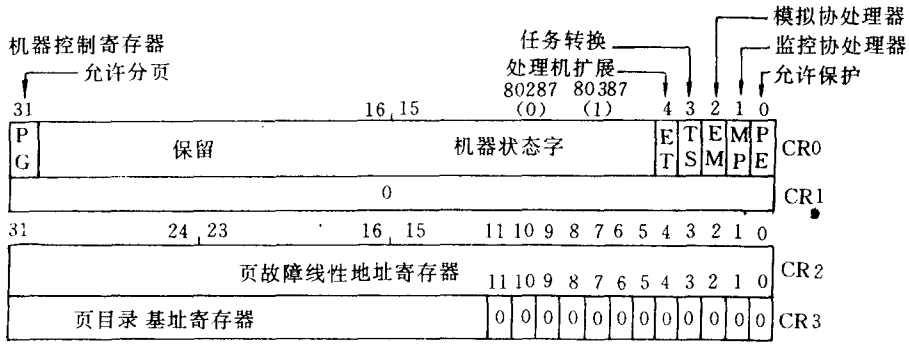

图 1.8　80386 的控制寄存器组

CR0 中的第 1 位是监控协处理位 MP(Monitor Coprocessor)。它与 TS 位一起决定：当 TS=1 时操作码 WAIT 是否产生一个“协处理器不能使用”的出错信息。

CR0 中的第 2 位是模拟协处理器位 EM(Emulate Coprocessor)。如果 EM=1，则所有协处理器的操作码都将产生一个“协处理器不能使用”的出错误信号；如果 EM=0，则所有协处理器的操作码都能在实际的 80287 或 80387 协处理器上执行。

CR0 中的第 3 位是任务转换位 TS(Task Switched)。当一个任务转换完成之后自动把它置 1。随着 TS 置 1，协处理器的操作码将会引起一个协处理器不能使用的陷阱。

CR0 中的第 4 位是微处理机的扩充类型位 ET(Processor Extension Type)，其内保存着微处理机扩充类型的信息。如果 ET=0，表示系统内用的是 80287 浮点协处理器。如果 ET=1，表示系统内用的是 80387 浮点协处理器。这样，设计人员可选择使用 80287 和 80387。

CR0 的第 31 位是允许分页位 PG(Paging Enable)。它表示芯片上的分页部件是否允许 PG 位和 PE 位配对，给 80386 提供选择操作环境的 4 种组合方式。由 PG 位和 PE 位定义的操作方式由表 1.1 说明。

表 1.1　PG、PE 位的 4 种组合

PG	PE	方　式
0	0	“实方式”，8086 操作
0	1	“保护方式”，确切的 80286 的语义加上 32 位扩充(在扩充的 80286/80386 保护方式范围内，还定义了一个支持虚拟 8086 的子环境)
1	0	未被定义。如果试图利用这一组合则将发出一个一般的保护故障
1	1	“分页保护”环境，允许保护方式下的所有设施都能分页

CR1 是未定义的控制寄存器，用作备用。

CR2 是页故障线性地址寄存器，保存最后出现页故障的全 32 位的线性地址。

CR3 是页目录基址寄存器，保存页目录表的物理基地址。因为 80386 的页目录表是按页排列的，所以低 12 位不起作用，即使写上了内容，也不被理会。

1.1.5 系统地址寄存器(System Address Register)

80386 有 4 个系统地址寄存器,如图 1.9 所示,它保存操作系统需要的保护信息和地址转换表信息。

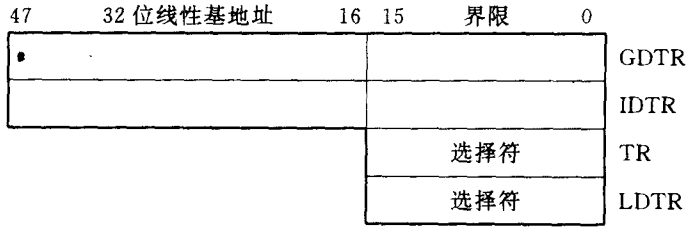

图 1.9 80386 系统地址寄存器

这 4 个专用的寄存器用于引用 80386 在保护方式下所需要的表或段。这 4 个系统地址寄存器的名字和作用如下:

全局描述符表寄存器 GDTR(Global Descriptor Table Register),是 32 位寄存器,用来保存全局描述符表(GDT)的 32 位线性基址和 16 位 GDT 的界限;

中断描述符表寄存器 IDTR(Interrupt Descriptor Table Register),是 32 位寄存器,用来保存中断描述符表(IDT)的 32 位线性地址和 IDT 的 16 位界限;

局部描述符表寄存器 LDTR(Local Descriptor Table Register),是 16 位寄存器,其内保存着 16 位的局部描述符表 LDT 段的选择符;

任务状态寄存器 TR(Task State Register)是 16 位寄存器,其内保存任务状态段 TSS 段的 16 位选择符。

用以上 4 个寄存器给目前正在执行的任务定义任务环境、地址空间和中断向量空间,这些寄存器的语义和控制与 80286 兼容。

1.1.6 调试寄存器(Debug Register)

80386 为调试提供了硬件支持。在 80386 芯片内有 8 个调试寄存器 DR0—DR7,如图 1.10 所示。

这些寄存器可以使系统程序设计人员定义 4 个断点,用它们可以规定指令执行和数据读写的任何组合。DR0—DR3 是线性断点地址寄存器,其中保存着 4 个断点地址。DR4、DR5 是两个备用的调试寄存器,目前尚无定义。DR6 是断点状态寄存器(Breakpoint Status Register),其低序位是指示符位,当允许事故调试并检测出事故而进入调试事故处理程序时,由硬件把指示符位置 1,调试处理程序在退出之前必须把这几位清

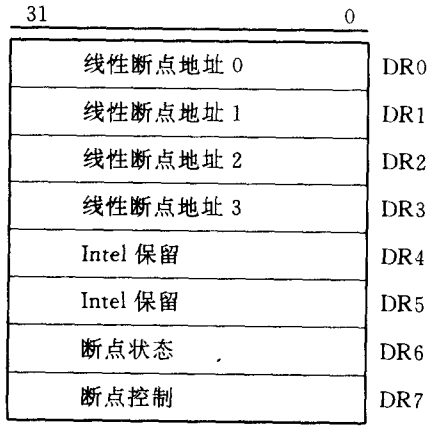

图 1.10 80386 的调试寄存器

0。DR7 是断点控制寄存器(Breakpoint Control Register),它的高序半个字又被分成 4 个字段,用来规定断点字段的长度是 1 个字节、2 个字节还是 4 个字节,和规定将引起断点的访问类型。低序半个字的位字段用于"允许"断点和"允许"所选择的调试条件。

1.1.7 测试寄存器(Test Register)

80386 有两个 32 位的测试寄存器 TR6 和 TR7,如图 1.11 所示。

TR6 和 TR7 是两个 32 位寄存器,用于在转换旁视缓冲器 TLB(Translation Lookaside Buffer)中测试随机存储器(RAM)和相联存储器(CAM)。TR6 是测试命令寄存器,其内存放测试控制命令。TR7 是数据寄存器,其内保存转换旁视缓冲器测试的数据。

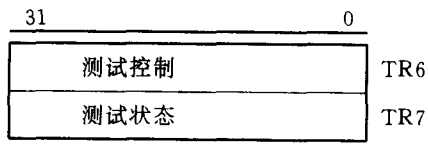

图 1.11 80386 的测试寄存器

1.1.8 寄存器的可访问性

在实方式和保护方式下,这些寄存器的可访问性有所不同,表 1.2 示出不同点。详细说明请参阅第 4 章。

表 1.2 寄存器使用方法表

寄存器	实方式		保护方式		虚拟方式	
	装入	存储	装入	存储	装入	存储
通用寄存器	可	可	可	可	可	可
段寄存器	可	可	可	可	可	可
标志寄存器	可	可	可	可	IOPL	IOPL
控制寄存器	可	可	PL=0	PL=0	不	可
GDTR	可	可	PL=0	可	不	可
IDTR	可	可	PL=0	可	不	可
LDTR	不可	不可	PL=0	可	不	不
TR	不可	不可	PL=0	可	不	不
调试控制	可	可	PL=0	PL=0	不	不
测试寄存器	可	PL=0	PL=0	PL=0	不	不

说明:1. PL=0,仅当现行特权级为 0 时,才能访问该寄存器。

2. IOPL,在虚拟 8086 方式中,PUSHF 和 POPF 指令对 I/O 特权级敏感。

1.2 寻址操作

80386 有许多寄存器,而且又提供了功能很强的指令系统,因而,80386 能够进行各种

不同数据类型的操作和寻址。80386 的寻址方式是先前 Intel 系列机寻址方式的扩充。由于增加了几个地址寄存器，所以 80386 的寻址方式又得到发展，显得更加灵活。

1.2.1 80386 的几种寻址方式

80386 为指令指定操作数提供了 11 种寻址方式，这些寻址方式可以有效地实现 PASCAL、FORTRAN、BASIC 等高级程序设计语言所需的绝大多数数据的访问。可把 80386 的寻址方式分成寄存器寻址、立即寻址与存储器寻址 3 种寻址方式。

1. 寄存器方式和立即方式

这两种寻址方式专供处理寄存器操作数和立即操作数的指令使用。

a. 寄存器操作数方式

操作数可放在 8 位、16 位或 32 位通用寄存器中。

b. 立即操作数方式

这种方式很直观，操作数作为操作码的一个组成部分含在指令中。

2. 存储器寻址方式

存储器寻址方式有 9 种。存储器寻址方式提供一种指定操作数有效地址的手段、途径。线性地址由两部分组成：段基地址和有效地址。而有效地址由 4 种地址分量任意组合而成。这 4 种地址分量如下。

a. 位移

跟在指令后面的 8 位或 32 位的立即值(在指令中加上地址前缀可使用 16 位位移)。

b. 基地址

任一通用寄存器的内容。这个基地址寄存器常常由编译程序用来指向局部变量区域的起始点。

c. 变址

除 ESP 以外的任一通用寄存器的内容。通常使用变址寄存器访问某一数组元素或某一字符串。

d. 比例常数

变址寄存器的值可以乘上一个比例常数(1、2、4 或 8)。乘上比例常数的变址方式对于访问数组或数据特别有效。

以上 4 种分量任意组合构成 9 种寻址方式。由于有效地址的计算与其他指令的执行是按流水线方式进行的，因此使用其中任何一种寻址方式组合都不会降低性能。但同时使用基地址、变址和位移分量的情况例外，因为此时要增加一个额外的时钟周期。由 4 种分量构成的 9 种寻址方式介绍如下：

① 直接方式

操作数的偏移以 8 位、16 位或 32 位位移量的形式作为指令的组成部分。

例：INC WORD PTR[500]

② 寄存器间接方式