80386 扩展内存编程

王建文　程　军　贺乐天

扩展 80386 内存

西安电子科技大学出版社

80386扩展内存编程

王建文　程　军　贺乐天

西安电子科技大学出版社
1994

(陕)新登字 010 号

内 容 简 介

80386 提供了 16M～4G 的可寻址空间，给用户提供了极好的运行环境。本书主要介绍如何利用 80386 所提供的内存空间进行应用程序设计。

本书共分八章。第一章介绍 80386 内存管理机制及其相关技术。第二章介绍 80386 硬件调试功能及其应用。第三章至第六章介绍扩充内存编程技术。第七章通过剖析一个 DOS 扩展器来介绍如何编写扩展内存驱动程序。第八章介绍了几个系统/应用软件使用扩充内存的情形。

本书适用于软件开发人员和广大计算机用户，也可作为大专院校计算机专业的教学参考书。

80386 扩展内存编程

王建文　程　军　贺乐天　编著

责任编辑　汪海洋

西安电子科技大学出版社出版发行

陕西省大荔县印刷厂印刷

新华书店经销

开本 787×1092　1/16　印张 24　字数 572 千字

1994 年 4 月第 1 版　1994 年 4 月第 1 次印刷　印数：1—5 000

ISBN 7-5606-0310-6/TP・0112　　定价：19.50 元

目　录

引言 …… 1
第一章　386 保护模式及存贮管理机制 …… 5
1.1　概要 …… 5
1.2　80386 实地址模式 …… 5
1.3　保护模式及其地址变换机制 …… 7
1.4　描述符 …… 19
1.5　任务切换及保护机制 …… 24
1.6　V86 模式 …… 34
第二章　80386 硬件调试及应用 …… 37
2.1　硬件调试 …… 37
2.2　BREAK 386 …… 38
2.3　编程要点 …… 62
2.4　高级 C 中断程序 …… 68
第三章　扩页内存规范(EMS) …… 73
3.1　EMS 的工作原理 …… 73
3.2　EMS 的检测 …… 74
3.3　常用 EMS 命令 …… 75
3.4　兼容性考虑 …… 88
3.5　CEMS 库函数 …… 88
3.6　CEMS 应用举例:DUP …… 88
3.7　在 EMS 中运行程序 …… 103
第四章　扩展内存的使用 …… 109
4.1　BIOS 调用 …… 109
4.2　扩展内存分配 …… 110
4.3　CEXT 库 …… 111
4.4　应用举例 …… 116
第五章　实地址模式访问 4G 内存 …… 119
5.1　原理 …… 119
5.2　关于汇编语言子过程 …… 131
5.3　使用 SEG4G 库函数 …… 131
5.4　应用举例 …… 132
第六章　扩展内存管理规范(XMS) …… 136
6.1　XMS 功能调用 …… 136
6.2　XMS 调用库 …… 142
第七章　DOS 扩展器(extender) …… 150

7.1 PROT 简介 …… 150
7.2 PROT 的使用 …… 151
7.3 运行文件的生成 …… 155
7.4 动态链接模式 …… 156
7.5 调试 …… 160
7.6 错误浅析 …… 161
7.7 多任务 …… 162
7.8 存在问题 …… 163
7.9 解决办法 …… 165
7.10 硬件中断 …… 166
7.11 32 位环境下的 16 位工具 …… 166
7.12 示范程序 …… 166
7.13 编写 DOS extender 产品 …… 167
第八章 扩展内存与扩页内存及其应用 …… 295
8.1 扩页内存 …… 295
8.2 磁盘高速缓存(Disk Cashing) …… 297
8.3 虚拟磁盘(VDISK) …… 298
8.4 AST 高速打印缓冲(SPERSPOOL) …… 300
8.5 Windows 与 DESQview 的内存管理 …… 300
附录 I 基本指令集 (与 8086/8088 指令集兼容) …… 306
附录 II 80286/80386 扩展指令集 …… 353
附录 III 80386 专用指令集 …… 359
附录 IV 保护模式系统控制指令集 …… 370

引　言

一、DOS 内存管理现状

MS－DOS 操作系统是目前市场上 PC 机的主流操作系统，PC/XT 机所使用的 8088/8086 芯片最大寻址空间为 1M 字节；更高档的微机所使用的芯片 80286 最大寻址空间为 16M 字节；80386 芯片的寻址空间更大。然而，基于这些芯片的操作系统 MS－DOS 所能管理的最大 RAM 空间却仅为 640K，也就是说，在 MS－DOS 下执行的程序受限于 640K 这个狭小的空间。这个数字与以上所列 80286/80386 芯片最大寻址空间相比可以看出，80286/80386 芯片的最大寻 址能力还远未得到发挥。

从操作系统角度看，常常需要牺牲时间来换取空间，或者以牺牲空间来换取时间，因此从根本意义上来讲，内存大小直接影响到计算机的运行速度和应用范围大小。内存越大，相应来说，程序的执行速度也越快，程序的应用范围也越大。因而如何尽可能地发挥 80286/80386 芯片的最大寻址能力便是本书所要讲述的主题。

IBM PC/XT 及其兼容机可直接寻址 1M 的内存空间，这一寻址空间又可分成若干 个区域留作专用。例如 0C0000H～0A0000H 用于视频显示 RAM，0F000H～(1M－1)H 用于系统 ROM 等。640K 以上的所有寻址空间都留作专用，但是其中有些区域还未明确规定其用途，如：0D0000H～0E0000H 的区域未被使用。0～640K 的寻址空间为用户程序 RAM。PC、XT、AT 机内存分配情况如图 1 所示。

二、什么是扩展内存/扩页内存?

一般地，我们将 80286/80386 微机的内存分为 3 种类型：基本内存(BASE MEMORY)，扩展内存(EXTENDED MEMORY)和扩页内存(EXPANDED MEMORY)。基本内存是指以 MS－DOS所能使用的自由 RAM 空间，从 0 到 640K 范围的线性地址空间；扩展内存是指 MS－DOS 应用软件通常所不能访问的 1M 字节以上的线性地址空间；扩页内存是指依据 Lotus，Intel 和 Microsoft 公司联合提出的 EMS 标准实现的页式内存，这种内存是非线性空间。

扩展内存与 80286/80386 芯片实际的寻址能力相对应，而扩页内存与芯片的实际寻址能力并无对应关系。实际上，基于 8088/8086 芯片的微机也可以使用扩页内存，使得应用程序可以使用超出 8088/8086 寻址能力(1M 字节)以上的内存空间。相应地，扩充内存的 RAM 卡必须附加额外的硬件机构。符合 EMS 标准的扩充内存卡我们称之为 EMS 卡。

以下说明符合 EMS 标准的扩页内存示例。

EMS 标准是软件和硬件设计标准的混合，它们规定了扩展内存卡如何与软件一起工作，根据硬件的特点怎样编写或修改软件。在 EMS 卡上的 RAM 不是永久的占据某一 个地址空间，而是由软件命令为 16K 逻辑页(EMS 卡上的存贮体)分配一个特定的 16K PC 地址空间。该地址空间也称为一个寻址窗口。通过使用软件开关，使扩展内存卡提供大量的内存逻辑页，使得多个不同的逻辑页对应于同一个寻址窗口，从而可以将一个大的存贮体空

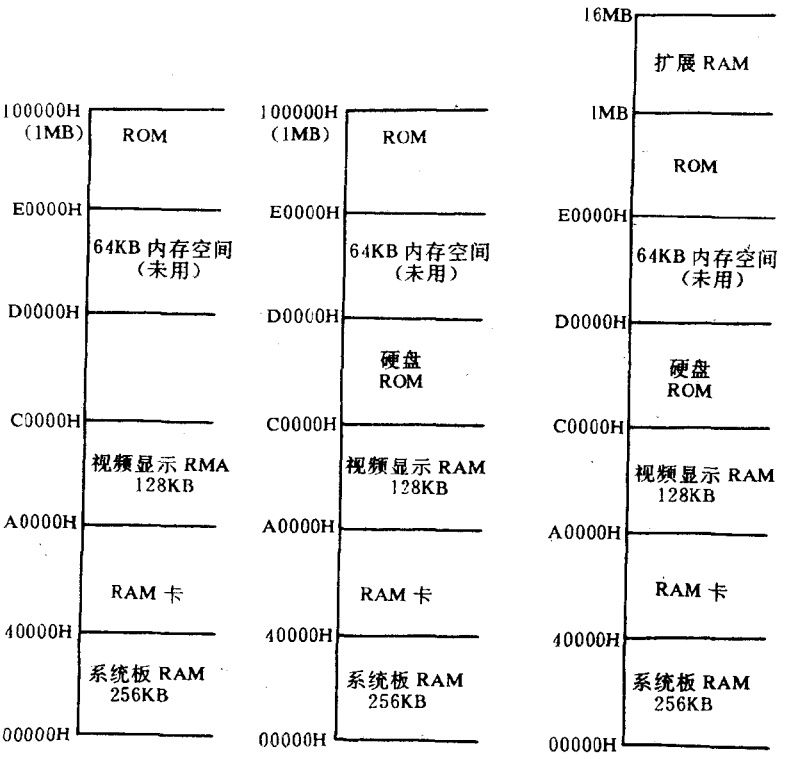


图 1　PC XT 和 AT 机的内存空间分配

间(EMS 卡 RAM) 映射到一个小的 PC 地址空间，从而达到扩充内存的目的。实际上，软体开关是 EMS 卡上的一些内部寄存器。

一旦某个逻辑页由软件命令分配了某个寻址窗口，程序对其存取就像对常规内存的操作一样。图 2 说明了 EMS 扩展卡上多个逻辑页中的一个出现在一个地址窗口上，其它的逻辑页等待的情形，EMS 要求在实际内存 1M 寻址空间上有连续的四个 16K 窗口，从而形成一个 64K 寻址空间主体，该 64K 寻址体的起始地址必须为 64K 整数倍。图 2 示例中，我们选取 0D0000H～0E0000H 未用区域作为 64K 寻址体空间。该体的四个窗口分别编号

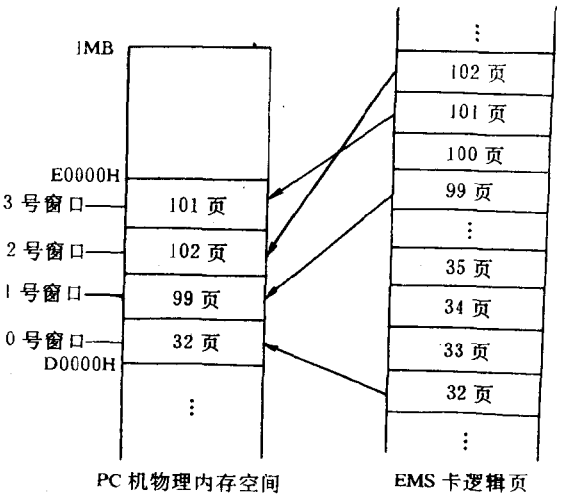


图 2　当前物理窗口与 EMS 逻辑页的映射关系

为 0 至 3 号。在 EMS 扩充卡上 的任一逻辑页都是可以出现在该寻址体的任一个窗口上的。

实现 EMS 标准的管理程序称为 EMM，EMM 实现了以上提到的扩充卡上逻辑页到寻址窗口的控制映射，而且 EMM 将 EMS 标准以 INT 67H 中断调用的方式提供给应用程序。通过加载 EMM 程序，应用程序可借助于软中断 INT 67H 直接与 EMM 建立通信，从而存取高达 16M 字节的扩充内存。

扩展存储器是地址高于 FFFFFH 或 1MB 的任何存储单元，只能由 80286 和 80386 处理器直接访问。80286 处理器有 24 位地址，可寻址到 16MB 内存，80386 处理器有 32 位地址，可寻址到 4GB，见图 3 和图 4。

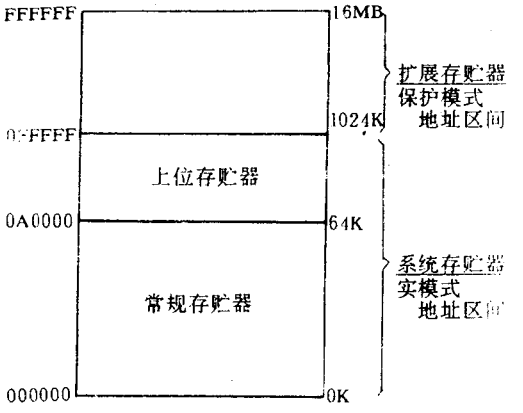


图 3　80286 PC 的扩展存储器

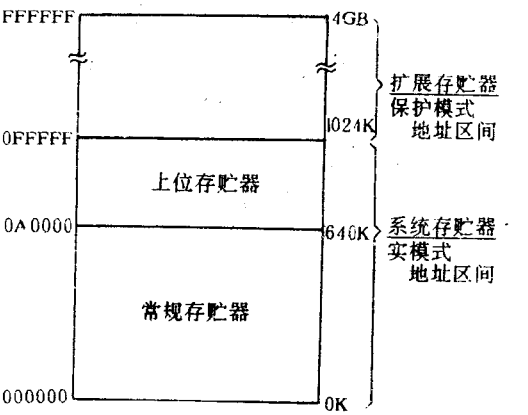


图 4　80286 PC 的扩展存储器

扩展存储器经常也被称为 XMS 存储器。Lotus，Intel ，Microsoft 和 AST 公司共同制定了 Extended Memory Specification (XMS) 2.0 (3.0 已问世)，定义了内存中三个特定区域的分配，即：高端存储区(HMA - High Memory Area)，上端存储块(UMB - Upper Memory Blocks)和扩展存储块(EMB - Extended Memeory Blocks)，见图 5。

HMA 是扩展存储器中紧挨 DOS 1MB 边界的第一个 64KB 区段(0FFFFFH 到 10FFFFH)，只能在 80286 和 80386 机器上使用．通过启动第 21 条地址线(A20)，可以在实模式下访问到。然而，HMA 必须做为一个单独的存储块处理，即不能分割共享，只能调入一个单独的程序。所以，HMA 一般用来存放尽量接近 64KB 的程序。

UMB 是上位存储器中的一些存储地址。虽然上位存储器(384K)技术上是保留的，但仍有一些未用部分可以通过专门的硬件和内存管理程序找到，构成 UMB，并在实模式下访问。而 EMB 是从 10FFFFH 以上的扩展存储器中分配的，只能在保护模式下访问。

在 80286 或 80386 PC 上使用扩展存储器，首先必须增加物理内存。为使性能更好，应把扩展内存与系统内存物理上一起放在 PC 母板上。如果母板上没有额外的存储器位置，可加一块内存适配板，并将其一部或全部配置成扩展存储器。然后，安装一个独立的扩展内存驱动程序，如 HIMEM .SYS 或 QEMM .SYS 等，用来管理系统或应用程序对扩展内存的访问。各种扩展内存驱动程序之间可能互不兼容。

在保护模式下 80286/80386 芯片引入了许多全新的概念：访问权限检查(读写保护)，地址限界检查，多任务切换等。80286/80386 芯片是为支持多任务操作系统而引入这些概

念和管理机制的。目前，在微机上实现运行的多任务操作系统有 OS/2，XEMX，或 UNIX System V（如 Release 4.0 版）等。在这些操作系统下应用程序实际上可以直接运行在扩展内存上。即使是在 MS－DOS 任务操作系统下，我们也可以通过指令来切换到 80286/80386 保护模式下，使用保护模式下的程序设计方法和工具来自由地使用扩展内存。这些程序设计方法和工具正是本书所要讲述的核心。我们将在书中给出全部工具软件的源程序代码。目前，在 DOS 下运行的许多实用程序，如高速磁盘缓冲存储，虚拟磁盘和打印缓冲等实用程序均是使用本书所提到的一些原理和类似方法来实现对扩展内存的访问。

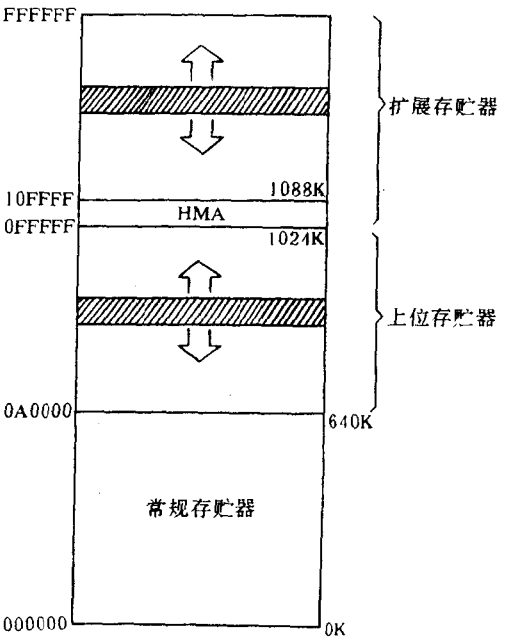


图 5　XMS 分配区域

另外，值得一提的是，如果读者已具备了并发、分时、多任务等操作系统概念，本书所介绍的内容将有助于您在 80286/80386 微机上编制出一个小型的多任务操作系统，如果您还具有某些非凡的天才（如组织管理能力），这个操作系统也许会发展成为商用性质的，从而得到推广。

三、本书内容组织

全书共分八章，第一章主要从原理方面介绍 80386 内存管理机制及其相关技术，比较系统地讲述了 80386 芯片所实现的保护模式及其地址变换机制，任务切换及保护机制，其中详细探讨了保护模式下最重要的数据结构——描述符(descript)，最后简要介绍了 V86 模式。

第二章介绍了 80386 硬件调试及其应用，并分别给出了用汇编语言和 C 语言编写的调试工具程序范例。80386 硬件调试功能可以使用在保护模式下。

第三章全面介绍了扩页内存规范 EMS，讲述了 EMS 原理、EMS 检测方法、常用 EMS 命令等内容，最后给出了一个在 EMS 环境下的程序设计实例——DUP

第四章至第七章逐次深入地讲述了扩展内存的使用方法，从单纯访问扩展内存时使用的 BIOS 调用方法到 DOS 扩展器(extender)源代码举例，系统地为读者介绍使用扩展内存编程的方法和工具，这部分内容也可以称为全书的主干。

最后一章简要介绍了几个扩页/扩展内存实用软件。

附录 I 至 IV 系统，分类列举了全部的 80386 指令集。

第一章　386保护模式及存贮管理机制

1.1　概　　要

第一章是以后各章的理论基础。在这一章里，我们着重介绍80386不同于8086/8088的内存访问方式及386保护模式，为大家以后阅读程序之便，还将80386扩展指令集做为附录列于书后。

386共有三种不同的内存管理模式：实地址模式(real address mode)，保护模式(protected mode)和虚拟8086模式(virtual 86 mode)。本章首先对读者比较熟悉的实地址模式做简单回顾，最后对虚拟8086模式一带而过，重点放在对保护模式及其寻址机制的详细描述上，使读者能够对386保护机制有一个清晰完整的概念。

本章内容如下：

(1) 80386实地址模式；

(2) 保护模式及其地址变换机制；

(3) 描述符；

(4) 任务切换及保护机制；

(5) V86模式。

1.2　80386实地址模式

当80386开机启动时，便自动进入实地址方式．用户可以一直运行在实模式，也可以通过软件指令切换到保护模式。80386实模式和8086/8088的寻址方式是一样的，只不过运行速度更快，内存配置更大些，可以说，在实模式下，80386是增强型的8086/8088。

在实模式下，386可以执行8086/8088基本指令集。这就是说，用户在8086/8088上运行的程序可以不做任何修改直接在386上运行。因此，80386在目标代码一级与8086/8088完全兼容。386另外增加了一些指令，目的在于增强性能，方便操作。例如，386增加了PUSH ALL (PUSHA)，POP ALL (POPA)，堆栈操作指令ENTER和LEAVE，输入/输出串指令INS、OUTS等。有了这些指令，用户可以将常用的大段代码以及大量循环操作用一两条指令代替。

80386的这部分指令称为80386扩展指令集，需强调一点，由于386使用了扩展指令集，使得用户在386上编写的程序无法在8086/8088上运行。因此大家在使用386扩展指令时需注意一点，即单纯为了编写方便是否值得牺牲可移植性？本章后面所说的程序除非特别说明，一般指的是汇编语言程序。

与8086/8088相比，80386还增加(增强)了一些寄存器，如图1.2.1所示，其中，寄存器EAX、EBX、ECX、EDX、EBP、ESP、ESI、EDI、EFLAGS分别由原来的16位扩展为32位，但原来的16位寄存器AX、BX等仍可单独使用，另外386还增加了两个数据段寄存器

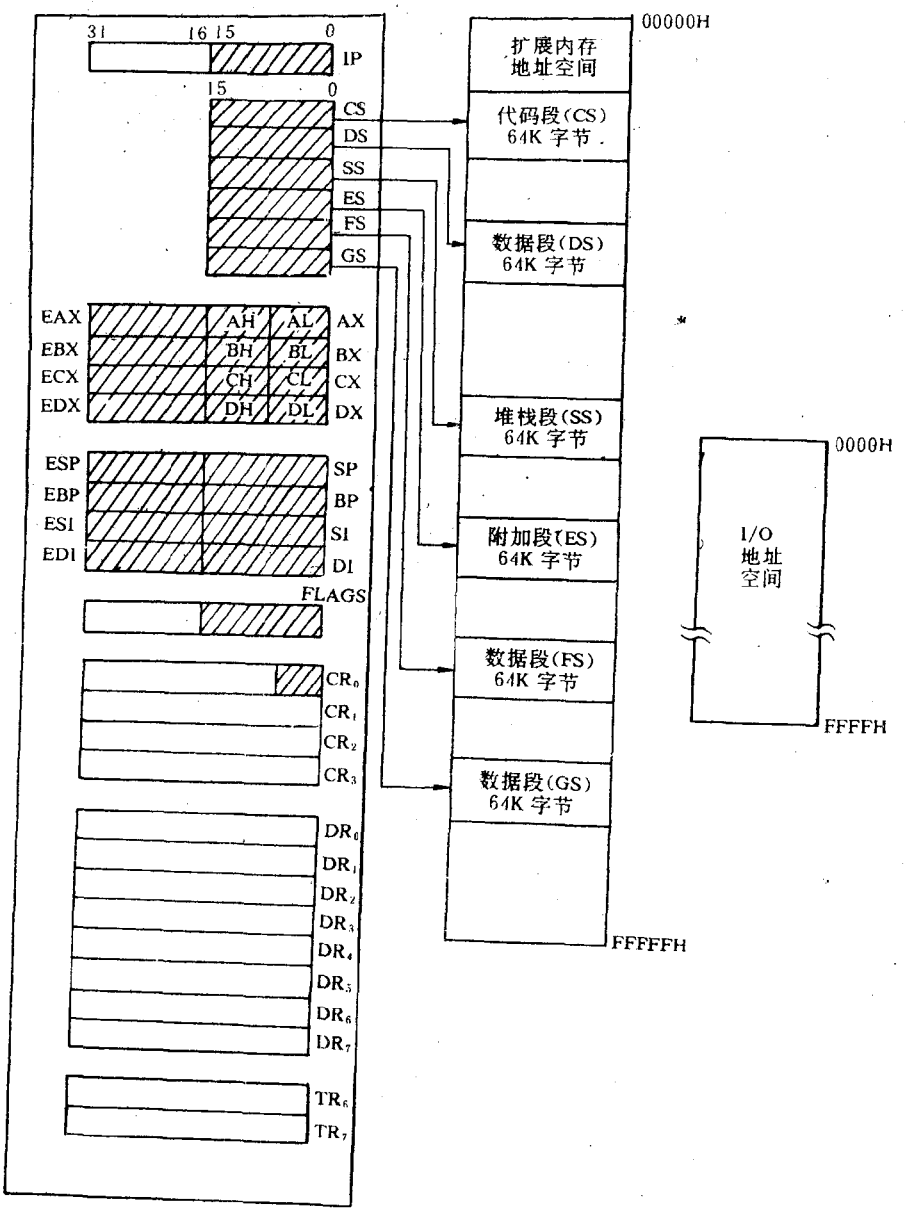


图 1.2.1　实模式下 80386 地址映射

FS 和 GS，这就是说，同时有 6 个段处在活动方式。在实模式下，386 仍使用 CS：IP 指示指令地址。从图 上可以看出，386 每段仍为 64K，寻址空间仍为 1M 字节。这是由于 386 通过段寄存器(内含基地址)和偏移量生成物理地址的方式与 8086/8088 完全相同，如 1.2.2 所示，段寄存器值左移 4 位与段内偏移量相加，形成 20 位物理地址，$2^{20}=1M$，故寻址空间为 00000H—0FFFFFH，即 1 兆字节。

80386 与 8086/8088 有相同的存贮空间和 I/O 空间，386 的 I/O 空间为 64K 字节，0000H—FFFFH，如图 1.2.3 所示。在图 1.2.3 (*a*) 中，我们可以看出，最初的 1K 字节(000H—3FFH)仍然用做中断向量表。在图 1.2.3 (*b*) 中，前 256 字节 I/O 地址空间(00～

FFH)被称做第 0 页。这些端口地址可以直接被 I/O 指令所访问。

最后，我们介绍怎样从实地址模式切换到保护模式。386 增加了四个控制寄存器，分别称为 CR_0、CR_1、CR_2、CR_3，这些控制寄存器均为 32 位，其中 CR_0 的第 0 位称为 PE 位(保护激活位)，在实模式下，PE 位置 0；当 PE 位置 1 时，系统将由实模式进入保护模式。PE 位可

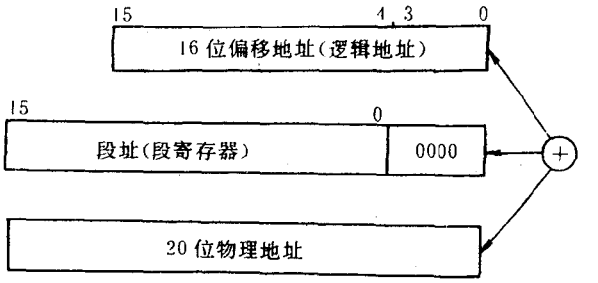


图 1.2.2　实模式下 80386 物理地址生成

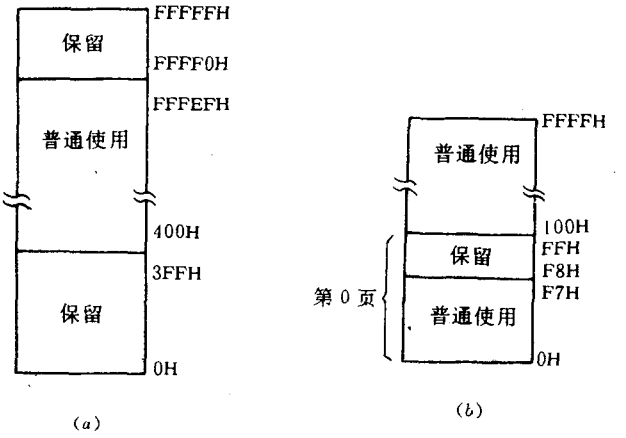


图　1.2.3

(a) 实模式下内存指定使用情况；　(b) I/O 地址空间

以通过 386 指令来改变。

1.3　保护模式及其地址变换机制

通过使控制寄存器 CR_0 的 PE 位置位，我们可以从实地址模式进入保护模式。保护模式是 386 不同于 8086/8088 内存管理模式的高级模式，在保护模式下，386 支持虚地址机制，页面机制，保护机制，以及多任务机制。在这一节里，我们将讲述保护模式下 80386 寄存器及描述符表，虚拟地址空间以及地址变换方式。

1.3.1　保护模式有关概念及 80386 寄存器

在保护模式下出现了许多不同于实地址方式的全新的概念，在此先做简要介绍：

(1) 任务(TASK)：是一个子程序的集合，这些子程序配合起来完成一个特定的功能。

(2) 描述符表(Descriptor Tables)：由多个描述符连续排列所形成的列表，其中每个描述符称为一项，描述符表长度称为限长，描述符表首项所在内存地址称为该描述符表的基地址。

(3) 描述符(Descriptor)：每个描述符在内存中占连续 8 个字节，用来描述一段内存地址及访问权限。

(5) 选择符(Selector)：作用类似于指针，用来在某个描述符表中选取某个描述符。

(6) 描述符表之描述符：一个描述符，该描述符用以指定一个描述符表。

(7) 描述符表寄存器：寄存器中的内容用来指定一个描述符表。

保护模式下所使用的寄存器模式如图 1.3.1 所示。比较图 1.2.1，我们发现保护模式下的寄存器集合是实地址模式下寄存器集合的超集。保护模式下出现 4 个新的寄存器：全局描述符表寄存器(GDTR)，中断描述符表寄存器(IDTR)，局部描述符表寄存器(LDTR) 和任务寄存器(TR)，除了这 4 个新出现的寄存器，另外有几个寄存器进行了扩展，如指令指针寄存器 IP 现在称为 EIP，由原来的 16 位增加为 32 位，标志寄存器 EFLAG 更多的位有了实际定义，控制寄存器 CR_0 到 CR_3 定义了实际的功能。接下去我们讨论这些寄存器扩展的目的及这些寄存器在保护模式下所起的作用。

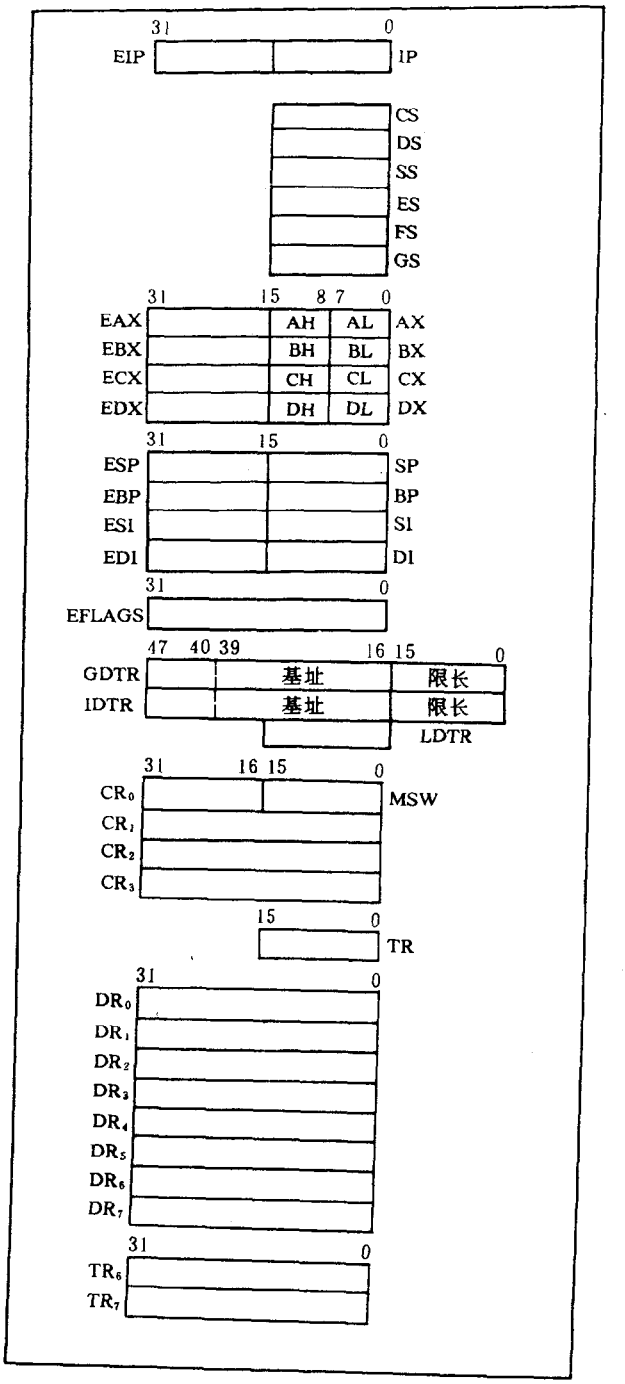


图 1.3.1　保护模式下的寄存器

(1) 全局描述符表寄存器

如图 1.3.2 所示，全局描述符表寄存器由两部分组成：基地址(32 位)及限长(16 位)，因而全局描述符表是一个 48 位的寄存器，通过基地址和限长，全局描述符表寄存器定义了一个全局描述符表(GDT)。

全局描述符表寄存器低 16 位称为限长，由这个限长所决定的全局描述符表 GDT 最大长度为 2^{16} 字

节，由于 GTD 中每个描述符占 8 个字节，故一个 GDT 最大可以包含 $2^{16}/2^3=8\,192$ 个描述符。限长不一定非取最大值，它是一个上限，只要小于等于这个上限且是 8 的整数倍即可，全局描述符表高 32 位称为基地址，它可以取 80386 4 GB 线性地址空间中的任一个地址。例如若 GDTR 值为 0700 0FFFH，则其基地址为 0700H，限长为 0FFFH，因此 GDTR 表长为 0FFFH +1＝1000H 字节，包含 1000H/8 ＝0200H 即 320 个描述符。关于线性地址、逻辑地址等概念详见下一小节。

一般地 80386 保护模式下只需有一个全局描述符表。在由实地址模式切换到保护模式之前，必须将基地址和限长装入全局描述符表寄存器，这可以通过软件指令来完成。指令 LGDT 读出 GDTR 内容，指令 SGDT 写入 GDTR 内容，有关指令详细情况参见附录。一旦装入，DGTR 的值一般不再变动。

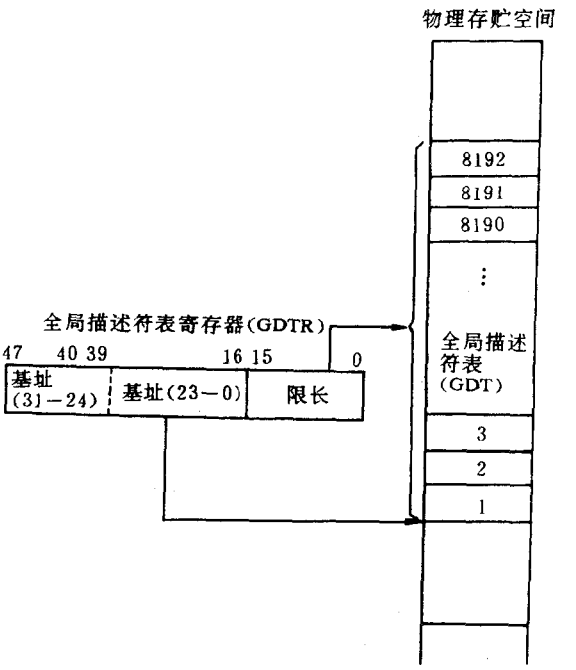


图 1.3.2　全局描述符表

(2) 中断描述符表寄存器

和全局描述符表寄存器类似，中断描述符表寄存器在内存中定义了一个中断描述符表(IDT)，所不同的是，中断描述符表中的描述符称为中断描述符，而不同于通常的段描述符。通过 IDTR 和 IDT，386 提供了一种机制使得微处理器控制权可以转交给中断服务程序或异常服务程序。

中断描述符表寄存器是 48 位寄存器，由两部分组成，即基地址(32 位)和限长(16 位)，如图 1.3.3 所示，基地址可以是 80386 4GB 线性地址空间中的任一个地址，限长

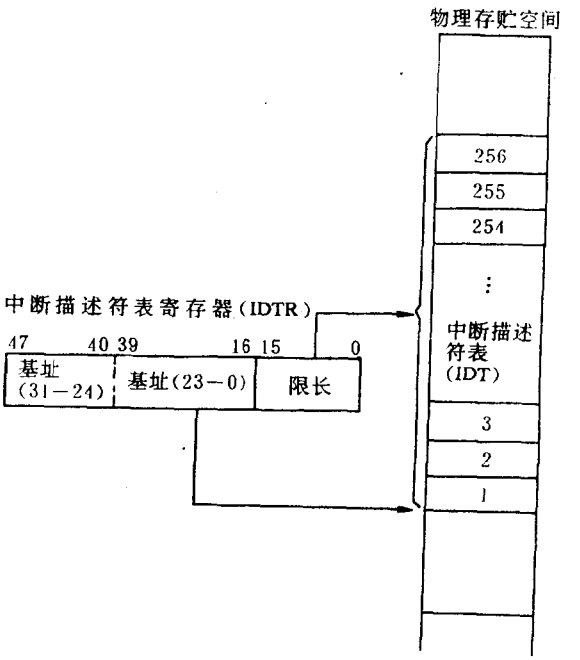


图 1.3.3　中断描述符表

最大可以是 65 535 字节($2^{16}-1$)，但由于 386 只有 256 个中断，所以只需 256 个中断描述符，即 8K 字节。

事实上，IDT 中的中断描述符称为中断门，我们将在下一节详细讨论。通过中断门，我们可调用中断服务程序。每个中断门也是 8 个字节，包含属性和中断服务程序的入口地址。

和全局描述符表一样，在切换到保护模式之前，必须将基地址和限长装入 IDTR 寄存器，一旦装入，IDTR 的值不再改变，指令 LIDT 和 SIDT 分别用来完成读出和写入 IDTR 的功能。

(3) 局部描述符表寄存器

局部描述符表寄存器和其它寄存器一起完成 80386 保护模式下的内存管理机制。局部描述符表寄存器必须和全局描述符表一起使用。如图 1.3.4 (*a*) 所示。每个任务拥有它自己的局部存贮器空间，这些存贮器空间由局部描述符表来定义，每个任务对应一个局部描述符表。在 386 保护模式下，多个任务间可以进行切换，因而内存中保留有多个局部描述表，图 1.3.4 (*a*) 中有几个描述符表。

一个描述符表之描述符包含这个描述符表所在的基地址和限长，因而描述符表之描述符指向一个描述符表。全局描述符表中，包含所有的局部描述符表之描述符，顾名思义，每一个局部描述符表描述符都指示一个局部描述符表，如图 1.3.4 (*b*) 所示。

局部描述符表寄存器是一个 16 位寄存器，里面的内容是一个选择符，通过选择符可以在全局描述符表中找到对应的局部描述符表描述符。每一个任务执行时都将其自身的选择符装入局部描述符表寄存器中，从而指示出自己所使用的局部内存地址空间。80386 微处理器在某一时刻只能执行一个任务，LDTR 内容即是与这个任务对应的选择符。

当一个选择符装入局部描述符表寄存器的同时，386 内部硬件逻辑自动将 GDT 中相应的局部描述符表描述符装入一个高速缓存器(cache)，这个高速缓存器称为局部描述符表高速缓存器，LDTR 与这个高速缓存器相互配合，从而指示出一个局部描述符表。图 1.3.4 (*b*) 虚线示出了这个过程。

(4) 任务寄存器

在 80386 保护模式中，任务间切换是一个重要的概念，而这一切最终都是通过任务寄存器内容的改变来实现的。任务寄存器是一个 16 位的寄存器。与 LDTR 相似，任务寄存器的内容也是一个选择符，它间接地指示出一个任务的任务状态段 TSS。每个任务都有一个任务状态段 TSS，任务状态段中保留有与这个任务相关的外部环境和内部数据，可以说在 386 保护模式中，TSS 是一个任务的唯一标识。

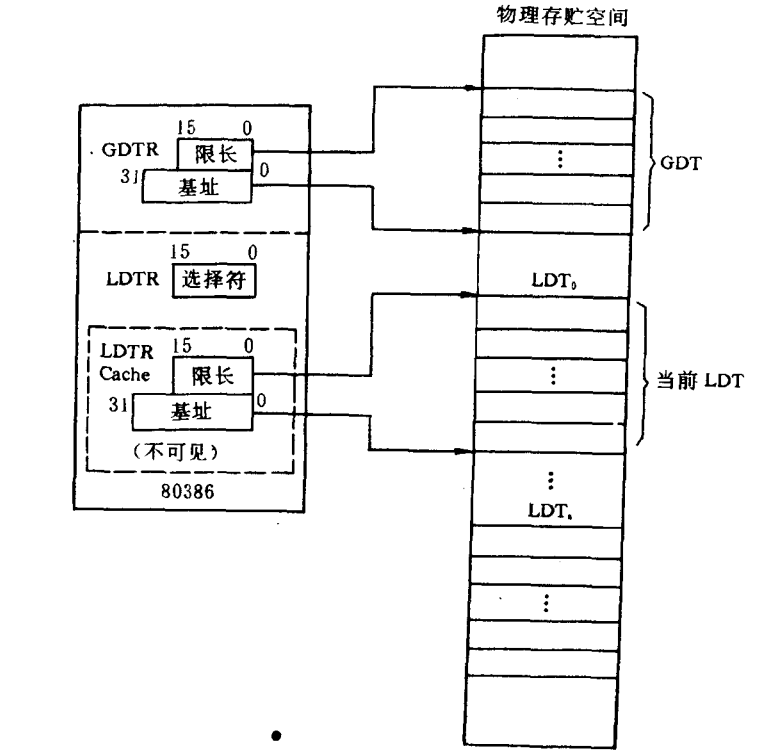
一个 TSS 描述符包含有 TSS 的基址和限长，在全局描述符表中保留有每个任务的 TSS 描述符，通过选择符可以选择某一个 TSS 描述符。

任务寄存器中装有当前任务的选择符。任务切换时，任务寄存器装入新的任务选择符。在任务寄存器内容改变的同时，80386 内部逻辑自动地将这个选择符所指示的 GDT 中的 TSS 描述符装入一个高速缓存器，这个高速缓存器与任务寄存器配合使用，称为任务描述符高速缓存器。

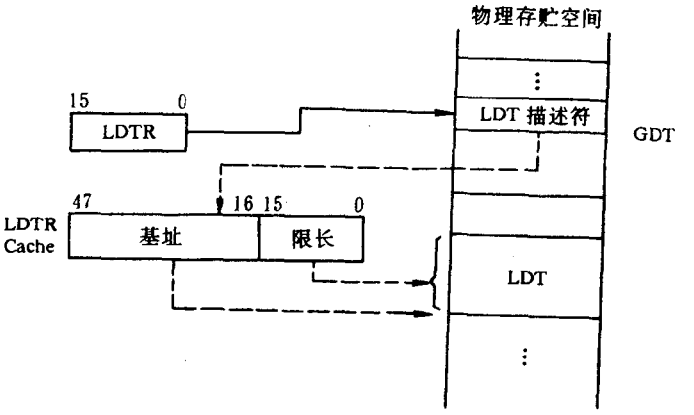
与图 1.3.4 (*b*) 类似，图 1.3.5 描述了任务寄存器间接指示一个任务状态段的全过程。

(5) 控制寄存器

前面我们已经介绍过，386 保护模式有 4 个系统控制寄存器，分别为 CR_0，CR_1，CR_2，



(a)



(b)

图　1.3.4

(a) 全局描述符表与当前任务的局部描述符表；　(b) LDTR 对局部描述符表的寻址机制

CR_3，如图 1.3.6 所示。每个控制寄存器均是 32 位寄存器，CR_0 的低 5 位是系统控制标志，被称为机器状态字(MSW)，CR_0 的最高位 PG 与 CR_2，CR_3 一起用于页面寻址机制。

让我们首先来看看 CR_0。MSW 共有 5 位，从低到高表示为 PE、MP、EM、TS、ET，其中 PE、MP，EM，ET 是保护模式系统设置信号，TS 是任务标志位。所有各位都可以通过语句

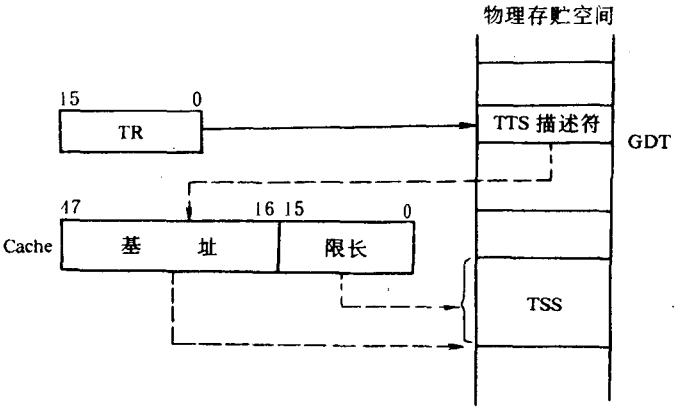


图 1.3.5　任务寄存器与任务切换机制

来改变。

保护模式激活位 PE 定义了 80386 是处于实地址模式还是保护模式。386 启动时，PE 位清零，此时系统处于实地址模式。通过指令设置 PE 位等于 1 时，系统进入保护模式。一旦系统进入保护模式，80386 就不能再通过软件指令切换到实地址方式。此时由保护模式进入到实地址模式的唯一办法便是硬件重新启动。

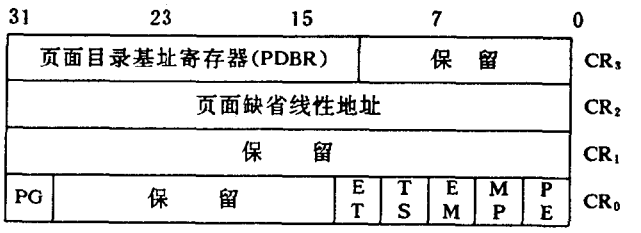


图 1.3.6　控制寄存器

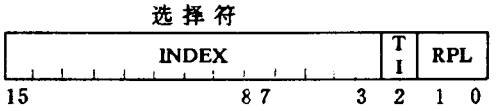
协处理器标志位 MP 置 1 表示数学协处理器存在，另一方面，如果系统通过软件模拟协处理器，则模拟协处理器位 EM 置 1。这两位，一次只能设置 1 位。协处理器扩展类型位 ET 用来指示系统所使用的协处理器是 80387 还是 80287，当 ET 置 1 时，表示系统使用的是 80387 协处理器。

任务切换位 TS，当系统由一个任务切换到另一个任务时，任务切换位自动置位。

通过将 CR_0 的页面激活位 PG 置 1，386 的页面寻址机制开始起作用。与 PG 位配合使用的有 CR_2 和 CR_3 两个控制寄存器。CR_2 的 32 位用来装载页面缺省线性地址，这是由于 386 使用了虚拟地址方式，逻辑地址远远大于实际的物理地址。因而 386 允许操作系统使用换入/换出机制，即可以将暂不使用的内存内容换出到外部大容量磁盘上，在用到时再将其从磁盘换入到内存中来，所以当 80386 寻址机制发现逻辑页面不在内存中时，便将缺省的页面线性地址写入到 CR_2 中以便换入。CR_3 的高 20 位(8－31 位)称为页面目录基址寄存器(PDBR)，用来保存页面目录的基地址。低 12 位全部为 0 以便页面目录基址定位在 4K 字节的整数倍上，这是由于每个目录表是 4K 字节大小的缘故。详细内容参见下一小节页面寻址机制部分。

(6) 段寄存器

在 386 保护模式下，所有的段寄存器的作用已不同于实地址模式，现在的段寄存器称为段选择符寄存器，它们的作用相当于前面定义的选择符。通过选择符可以间接地指示一块内存地址空间。图 1.3.7 更详细地描述了一个选择符的内容。



位	名称	功　能
1—0	请求优先级(RPL)	保护方式下选择符的优先级
2	表指示(TI)	TI=0 在全局描述符表中索引 TI=1 在局部描述符表中索引
15—3	索引	在表中选择描述符入口

图 1.3.7　选择符形式

一个选择符的低两位(第 0 位和第 1 位)称为请求优先级位 RPL，可以是 0，1，2，3，中的任一个数，数值越大优先权越低。第 2 位称为任务指示位 TI，如果 TI 的值等于零，选择符将作用在全局描述符表 GDT 中。如果 TI 的值等于 1，选择符将作用在局部描述符表 LDT 中。

选择符的高 13 位才是真正起指针作用的部分，因而一个选择符最多可选 2^{13}，即 8K 个描述符。如果将 TI 指示出两个表的作用计算在内，那么一个选择符可选 2^{14}，即 16 K 个描述符。

(7) 标志寄存器

在 386 保护模式下，标志寄存器由原来的 FLAG 16 位扩展为 EFLAG 32 位，因而增加了 80386 保护模式的许多关键标志位，如图 1.3.8 所示。

(一) 模拟 80386 模式位 VM

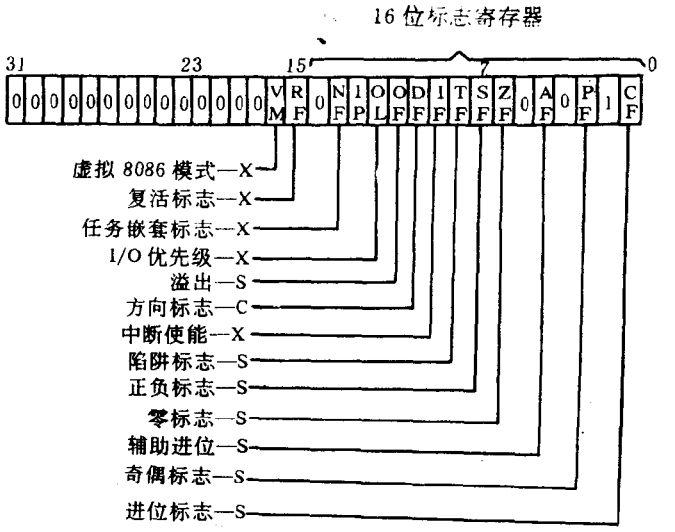
VM 位在 386 保护模式下提供模拟 8086 模式．如果在保护模式下将此位置 1，386 将切换到 V_{86}模式。在这种模式下，386 将像 8086 微处理器一样工作。VM 只能在保护模式下置位，且不能直接置 1，必须通过 IRET 指令或者通过任务切换来实现。其它的软件指令对这一位将不起作用，例如 386 指令 PUSHF 每次只能在这一位写 0，即使系统正处在 V_{86}方式。这是由于 V_{86}模式被当作 386 保护模式下的一个任务来执行。

对于 V_{86}模式更详细的情况将单独作为一节在后面论述。

(二) 恢复标志位 RF

恢复标志位 RF 用于调试寄存器断点设置或单步跟踪。每次断点执行前，RF 将在指令边界上被检查。如果 RF 置位，它将在下一条语句执行时忽略任何跟踪错误。当一条指令(不包含 IRET)正确执行完成时，RF 自动恢复为 0。

(三) 嵌套任务位 NT



注：　S＝状态标志　C＝控制标志　X＝系统标志

图中 0 和 1 是 INTEL 保留位，无定义

图 1.3.8　保护模式标志寄存器

当一个任务嵌套调用另一个任务时，将产生 NT 置位，更详细的情况参见第五节任务切换和保护机制。

（四）I/O 优先级位 IOPL

IOPL 位用于为输入/输出设置一个最大优先级。举例来说，如果 IOPL＝00，则输入/输出只能在 386 处于最高优先级 0 时执行。如果 IOPL＝3，则输入/输出可以在任一优先级上实现。

其它几个标志位，如溢出位 OF，方向标志位 DF，中断使能标志位 IF 等与 8086/8088 定义完全一样，在此不再介绍。

1.3.2　保护模式内存管理及地址变换机制

在上一小节，我们已经比较详细地介绍了 80386 保护模式下和各种描述符表相关的寄存器，涉及了如何由选择符间接指示一个描述符表，但是还没有叙述如何得到一个具体的物理地址，本小节将详细叙述 386 保护模式地址变换机制，包括页面寻址机制，使读者对 386 如何完成虚拟地址存贮管理有一个比较完整的概念。本小节以几个实例图示作引导，力争深入浅出地给读者留下一个印象。

386 内存管理涉及到三种类型的地址空间，分别称为逻辑地址空间，线性地址空间和物理地址空间。逻辑地址空间是一个任务所看到的地址空间，即从用户某个任务角度来看，系统可提供的最大内存空间。一般来说，逻辑地址空间远远大于物理地址空间，但它可以映射到物理地址空间；如果没有页面寻址机制的话，线性地址空间就是物理地址空

间，二者没有什么差别。然而一旦处于页面激活状态，线性地址空间就远远不同于物理地址空间，此时的线性地址空间具有目录，页表，页面三级索引，从而建立起一个层次结构的地址空间。实际上，当页面激活时，线性地址空间把物理地址空间层次化，形成了一个以页为单位的主体化结构。

前面我们已经讲过，通过改变控制寄存器 CR_0 的 PG 位，我们可以激活页面寻址机制。图 1.3.9 说明了 386 保护模式下地址变换的一般过程。

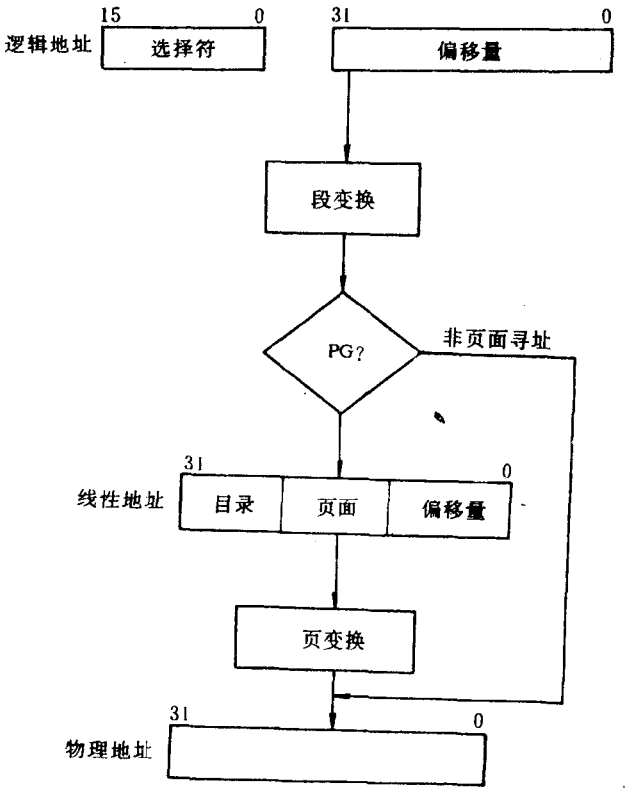


图 1.3.9　逻辑地址向物理地址的变换

当 PG＝0 时，页面没有被激活，线性地址即是物理地址。

当 PG＝1 时，页面激活，线性地址通过目录、页表等索引变换生成物理地址。

下面我们结合前面的内容，以 CS：EIP 寻址为例，分别说明以上两种情况。

首先来说明 PG＝0 时的情况，我们假设代入实际数据，如图 1.3.10 所示，现在来分析一下地址变换的过程。

如图所示，首先假设正在执行任务 W，CS 值为 2005H，EIP 值为 000100FFH，全局描述符表寄存器 GDTR 基址为 00900000H，限长为 0FFFFH，LDTR 值为 1000H，则 CS：EIP 指示当前指令地址变换过程如下：

(1) GDTR 通过基址和限长指示出 GDT(64K)

(2) LDTR 内容为 1000H，即 0001 0000 0000 0000B 根据前面所学选择符内容，我们知道 TI 位为 0，说明选择符将作用在全局描述符表 GDT 中，最后两位是 RPL 位，即请求优先权为最高优先权 00；去掉后三位，我们得到 0 0010 0000 0000B，即(1000÷8)H，说明选择

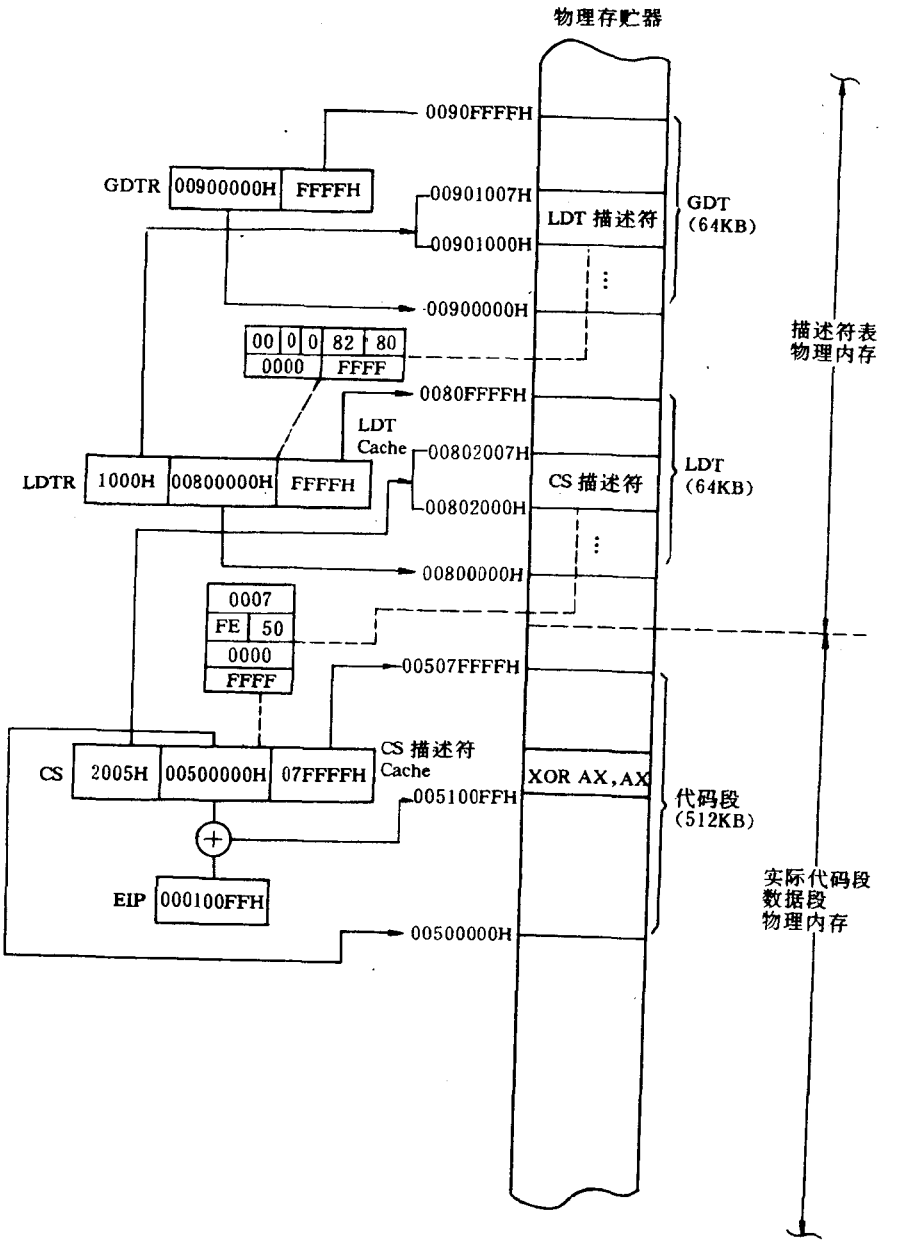


图 1.3.10　保护模式下代码段寻址实例

符指定的是 GDT 中第(1000÷8)H 个描述符，由于每个描述符占 8 个字节，所以描述符是从

[(1000÷8)×8+0090 0000]H=00901000H

处开始的连续 8 个字节。我们从 0090 1000H 处取出 LDT 描述符，将其基地址和限长装入局部描述符表高速缓存器，内容如图上所示，基址为 0080 0000H，限长为 FFFFH，因而我们得到任务 W 所对应的局部描述符表(64K)。

以上涉及到描述符具体内容，详见下一节“描述符”。

(3) CS 值为 2005H，即 0010 0000 0000 0101B，TI 位为 1，因此段选择符 CS 将作用在任务 W 的局部描述符表 LDT 中。类似于过程②，我们在 LDT 中得到描述符如图所示，其基址和限长装入 CS 对应的高速缓存中，基址为 0050 0000H，限长为 07FFFFH，因此我们得到 CS 所指示的任务 W 的代码段内存空间(512KB)。

(4) 由于 PG＝0，EIP 内容加上代码段基址，我们最后得到当前正在执行指令在内存中的实际物理址：

000100FFH＋00500000H＝005100FFH ，

由图上我们可以看出，这条指令的实际内容为

XOR AX，AX ；

以上我们示例了 PG＝0 时代码段地址变换过程。事实上，数据段的寻址过程与此完全相似，只不过相应地需把 CS 改为 DS，把 EIP 值改为相应偏移量即可。读者可以对照上图自己分析 386 数据段寻址过程。

现在来说明 PG＝1 时的地址变换过程。当页面激活时，从图 1.3.9 我们可以看到，与 PG＝0 时的唯一不同在于多了一个从线性地址到物理地址的变换过程，以下我们来分析这种情况。

如图 1.3.11 所示，线性地址被分为三个部分，分别称为偏移、页表和目录。偏移指出在一个页面内的偏移量，共 12 位，因此我们知道一个页面的大小为 2^{12}＝4K 字节；页表占 10 位，因此一个页表包含 1K 个页面，从图上可以看出，每一个页表项的内容是一个相应

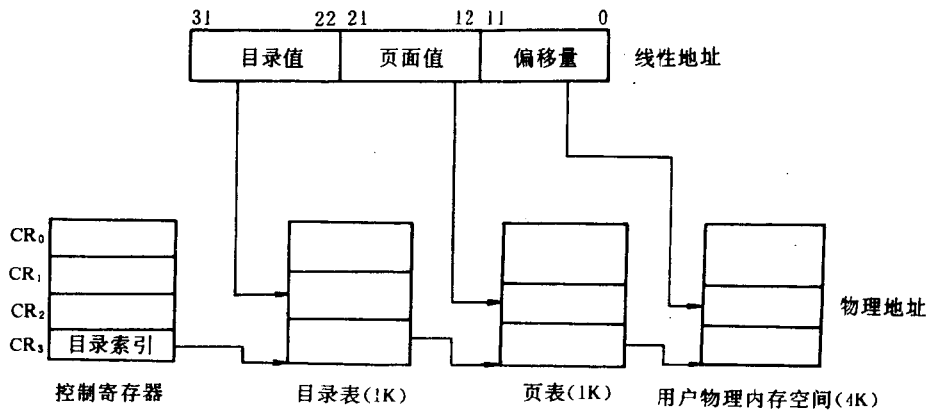


图 1.3.11　页面变换机制

页面的基址；线性地址的目录部分也占 10 位，因而一个目录表包含 1K个目录项，其中每个目录项内容指向一个页表的基地址。每个任务执行时一般只有一个目录表，其在内存中的物理基址由控制寄存器 CR_3 来指示。

之所以把线性地址层次化，在于产生出一个适当大小(4K)的页面做为内存管理的一个单位，使得在换入/换出等操作系统时能够节省时间和空间。

参照图 1.3.11，线性地址变换为物理地址的过程读者想必已经很清楚了，在此不再多述。

我们再从宏观上看一下，在页面激活情况下，怎样由逻辑地址，比如说 CS：EIP 对应出页面寻址机制。段选择符 CS 间接地指示出代码段的基地址，一般来说这个基地址只有高位部分，低位为 0，也即相应于一个任务目录表中的一个目录项，我们可以称之为基本目录项。我们将 EIP 分为三部分，相应于目录，页表和偏移。我们假设 EIP 值有所改变，变为 EIP′，相应地，将 EIP′也分为三部分。假如 EIP 改变的是偏移部分，则 EIP′与 EIP 在同一页面。如果 EIP 改变的是页表部分，EIP 与 EIP′在同一页表内但不在同一页。如果 EIP 改变了目录部分，则 EIP 所在目录为基本目录加上 EIP 目录部分的值。因而我们知道，如果将 CS 仍看做段寄存器指示某一段的话(此时段大小为 2^{32}字节)，那么一段可以对应于一个目录，此时逻辑偏移地址相应目录部分不变；也可以对应于多个目录，这时相应目录部分改变。当然，在 386 保护模式下已没有了段的概念，我们不必再将保护模式下目录、页的概念同实模式下段的概念做对比，最好的办法是在保护模式下忘掉段这个词。

需要补充一点，和前面类似，在页面寻址时，386 内部也使用了高速缓存器。

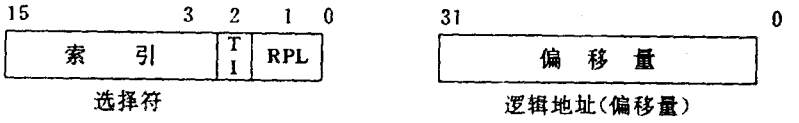


图 1.3.12　逻辑地址组成

最后，我们研究 386 保护模式下逻辑地址空间的大小。386 有 32 根地址线，因此物理上可寻址 2^{32}即 4G 字节；如图 1.3.12 所示，逻辑地址由段选择符和逻辑偏移量组成。段选择符有 16 位，但最后两位为 RPL 位，在实际寻址时不起作用。TI 位指示 GDT 还是 LDT，故能够将逻辑地址空间分为两部分，属于有效寻址位，因此选择符有 14 位有效。每个逻辑偏移量由 32 位组成，每个段选择符都对应于 32 位的逻辑地址空间，故每个任务所拥有的逻辑地址空间为 $2^{32}\cdot 2^{14}=2^{46}$，即 64T 字节．逻辑地址空间又可被 TI 位分为两大部分，即全局逻辑地址空间和局部逻辑地址空间，每一部分占 32T 字节。386 保护模式下，每个任务在执行时都拥有 64T 的逻辑地址空间。由于 386 可支持多任务，因而对多任务来讲，386 的逻辑地址空间是无上限的，这可以用图 1.3.13 来说明。

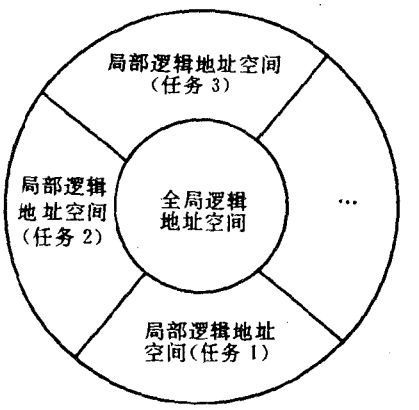
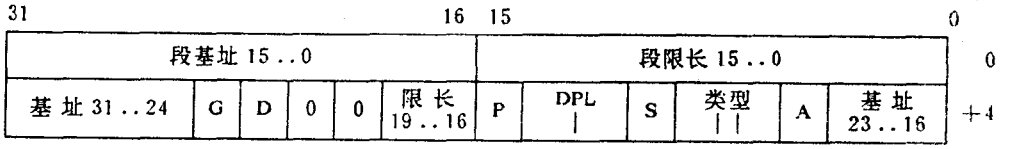


图 1.3.13　逻辑空间

1.4 描 述 符

前面几节已经多次提到描述符的概念。在 386 保护模式下，描述符是一个核心的概念。前面我们已经提到，段选择符通过在描述符表中选择某个描述符从而间接定义一段内存空间。描述符种类很多，我们前面用到的概念其实只是其中的一种，称为段描述符。下面我们分类介绍各描述符的含义和作用。

每个描述符在内存中占 8 个字节，其中包含一段线性存贮空间的信息。这些信息包括 32 位的线性空间基地址，20 位的限长，还包括保护权限，读写或执行优先权，操作码缺省大小(是 16 位还是 32 位地址操作)以及段类型等．所有这些信息都包含在一个描述符的 64 位中。图 1.4.1 说明了描述符的一般格式。



注：基址　段的基地址
限长　段的长度
P　存在位　P=1 表示存在　P=0 表示不存在
DPL　描述符优先级 0—3
S　系统描述符标志　S=0 系统描述符　S=1 代码或数据段描述符
类型　段的类型
A　访问位
G　分片位　G=1 每段长以页为单位(每页 4 K 字字)
　　G=0 每段长以字节为单位
D　缺省操作码大小　D=1 表示 32 位段
　　D=0 表示 16 位段

图 1.4.1　段描述符

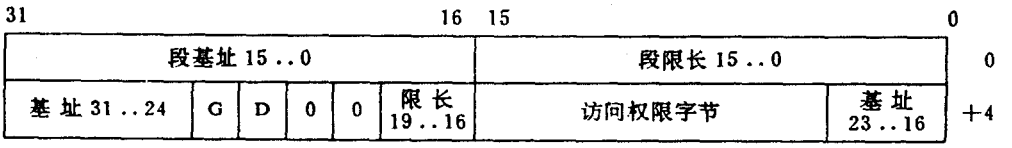
386 所有的描述符都有三个相同的特征信息位，分别称为存在位 P，描述符优先权位 DPL 和系统位 S。当 P=1 时，这个描述符所指示的段已经装入了内存。当 P=0 时，表示这个描述符指示的段不在内存中，此时任何对这个段的访问都将引起一个异常中断 11。描述符优先权位 DPL 占两位，它定义了所指示段的保护优先级 0—3。386 保护模式将段分为两大类：系统段和非系统段。系统位 S 定义了系统类型。当 S=0 时表示描述符所指示的是系统段；当 S 为 1 时，表示该描述符指示的是非系统段，即代码段或数据段。

(1) 386 代码数据段描述符(S=1)

图 1.4.2 表示了代码段、数据段描述符的一般格式。表 1.4.1 列出了访问权限字节各位的定义。

代码段和数据段描述符有几个相同的位。A 位代表访问位，当 CPU 访问过这个描述符时，A 位置位；反之 A 位置 0，表示这个描述符从未被访问过。在操作系统中，A 位被用作访问记录，与 P 位相互配合。比如我们发现 A=0，则认为这个描述符所指示的段目前使用

不频繁；为节省实际内存，我们可以将其从内存中移出到外部存贮器，并将 P 位置 0。G 位代表分片位，定义了一个段是按字节分还是按页来分。如果 G＝0，则 386 的一段将是 1 兆字节长；如果 G＝1，则 386 的一段将是 1 兆个页面，因为每个页面占 4K 字节，故 386 一段可达 4000M 字节。分片位与页面地址变换机制无关，386 系统可以是由按字节划分的段组成，也可以由按页划分的段组成，而不管页面变换机制是否激活。E 代表可执行位，用来区分一个段是代码段还是数据段。当 E＝1，代表代码段；E＝0 代表数据段。



注：D　缺省操作码大小
　　G　分片位
　　0　系统限制 386 芯片必需为此值，以此和将来芯片兼容

图 1.4.2　段描述符

表 1.4.1　代码段和数据段描述符的访问权限字节定义

位	名　称	功　　能
7	存在位(P)	P＝1　表示此段已经映射到物理内存中 P＝0　表示此段不在物理内存中
6—5	描述符优先级(DPL)	在优先级检查中段的优先级特征值
4	系统段描述符	S＝1　代码段或数据段(包括堆栈段)描述符 S＝0　系统段描述符或门描述符
3	可执行位(E)	E＝0　数据段描述符
2	扩展方向(ED)	ED＝0　向上扩展，偏移量小于等于限长值 ED＝1　向下扩展，偏移量大于限长值
1	可写位(W)	W＝0　数据段不可写 W＝1　数据段可写 （如果是数据段 (S＝1，E＝0)）
3	可执行位(E)	E＝1　代码段描述符
2	相容位(C)	C＝1　代码段只能执行在 CPL≥DPL，并且 CPL 保持不变
1	可读位(R)	R＝0　代码段不可读 R＝1　代码段可读
0	访问位(A)	A＝0　此段尚未被访问过 A＝1　段选择符已经装入段寄存器 或执行选择符测试指令

当 S＝1 且 E＝1 时，说明此段为代码段，此时，如果可读位 R＝1，表示此代码段可读且可执行。如果 R＝0，则表示该代码段可执行但不可读。所有的代码段均不可写，因而我

们从一个侧面看出386保护模式对破坏内存的病毒起到了有效的防范作用。

以下补充说明几点：

代码段可以通过别名(aliases)来修改。别名是可写的数据段，它与代码段有相同的线性地址空间，因而我们知道，对别名进行修改，也就是对相应代码段进行了修改。别名提供了一个有效的修改代码段的途径。

D位定义了缺省的操作码作用于有效地址的大小。D=1时，操作码针对于32位地址操作。当D=0时，操作码作用于16位地址，因而如果我们将D置为0，则所有现存的286代码段均可在386环境下执行。

代码段的另一个特征位是相容位C。如果C=1，代码段可由不同优先级的各段代码组成并在不同的优先级上执行。

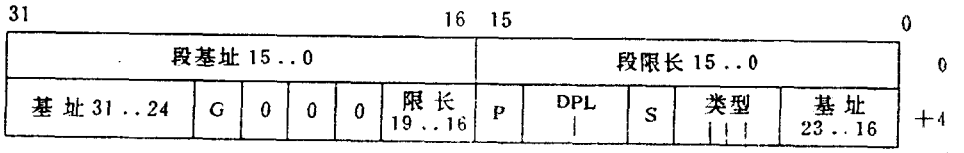
当S=1且E=0时，表示此段为数据段。数据段又可分为两类：堆栈段和数据段。这两类数据段有不同的操作方向。堆栈段由高地址向低地址方向操作，而数据段由低地址向高地址方向操作。沿伸方向位ED定义了一个段是向下(堆栈段)还是向上(数据段)。如果一个段是堆栈段，所有的偏移量都必须大于限长，因为其限长是指下限，其基址从高地址处开始。反之，如果一个段是数据段，所有的偏移量都必须小于等于限长，因为其限长是上限，基地址从低地址处开始。

可写位W定义了对一个段进行修改的权力。对于数据段，如果W=0，则说明该数据段只读。如果W=1，则可对此数据段进行读/写操作。对于堆栈段，W值必须置1。

B位控制栈指针寄存器的大小，如果B=1，所有有关堆栈的操作均由32位栈指针寄存器ESP执行。如果B=0，则所有操作由16位栈指针寄存器SP执行。

(2) 系统描述符一般格式(S=0)

系统描述符包括操作系统表、任务和门。图1.4.3表示了系统段描述符的一般形式，图的下方定义了不同类型的系统段描述符。386的系统描述符包括32位的线性基地址和20位的限长(286只有24位的基地址和16位的限长，286描述符最高两字节16位全部为0)。



注：　类型定义

0	Intel保留	8	Intel保留
1	可用80286 TSS	9	可用80386 TSS
2	LDT	A	保留
3	忙80286 TSS	B	忙80286 TSS
4	调用门(80286)	C	调用门(80386)
5	任务门	D	Intel保留
6	中断门(80286)	E	中断门(80386)
7	陷井门(80286)	F	陷井门(80386)

图1.4.3　系统段描述符

(3) 局部描述符表描述符(S=0，类型 Type=2)

局部描述符表描述符包含了一个局部描述符表的有关信息。局部描述符表的每一项内容由某一特定任务的所有段描述符构成。由于有关 LDTR 的指令只能在保护优先权最高级 0 级下执行，所以局部描述符表描述符中的 DPL 项将不起作用。另外，局部描述符表描述符只允许出现在全局描述符表 GDT 中。

(4) 任务状态段描述符(S=0，Type=1，3，9，B)

一个任务状态段描述符包含了这个任务状态段 TSS 的位置、大小和保护权限等信息，而任务状态段包含了一个任务本身的所有信息以及与这个任务嵌套的所有链指针。类型域用来指出一个任务是否正处于忙状态(也就是说与一个活动任务有链接)，这个任务状态段描述符是否可用，类型域还指出了这个段是否包含了 286 TSS 或 386 TSS。任务寄存器 TR 包含了指向当前任务状态段的选择符。Type 域内容如表 1.4.2 所示。

表 1.4.2　类型域内容(简)

类型值	功　　能
0001	可用 286 TSS
0011	忙 286 TSS
1001	可用 386 TSS
1011	忙 386 TSS

(5) 门描述符(S=0，Type=4—7，C，F)

门描述符经常称为门，用于控制对目标代码段入口点的访问。门描述符有四种类型，称为调用门(call gates)，任务门(task gates)，中断门(interrupt gates)和陷阱门(trap gates)。门用于间接地在不同保护优先级的源代码和目标代码之间进行控制权移交。在对门的访问过程中处理器自动检查保护权限。门还允许系统设计者对操作系统入口点进行控制。

调用门用于改变优先权，任务门用于任务切换，中断门和陷阱门用于特定的中断服务程序。

图 1.4.4 表示了这四种门的形式。调用门用于低优先权代码对高优先权代码的调用。调用门由三部分组成：访问字节、一个包含选择符和偏移量的长指针、一个字计数，其中长指针指向要调用例程的开始处，字计数用于计数需从调用代码堆栈拷贝多少个参数到被调用例程的堆栈段，字计数域只有调用门才使用，在其它门中字计数域闲置。

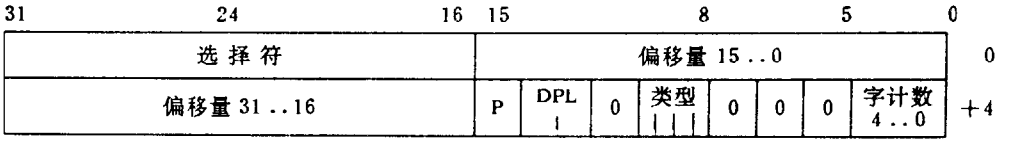
和调用门类似，中断门和陷阱门也使用一个长指针指向中断服务例程，这个长指针包含目标码的选择符和偏移量。中断门和陷阱门的不同在于中断门不允许中断(将 IF 位复位)，而陷阱门允许中断。

任务门用于任务切换，由于任务门只需指向任务状态段即可，故任务门中只有选择符起作用，而偏移量域处于闲置状态。

访问字节格式对于所有类型的门都一样，P=1 代表门的内容有效，P=0 代表门的内容无效并引起异常 11。描述符保护优先级 DPL 在这个描述符被使用时由一个任务所定义，系统访问位 S 必须为 0，类型域由图 1.4.4 所定义。

(6) 286 描述符与 386 描述符的比较。

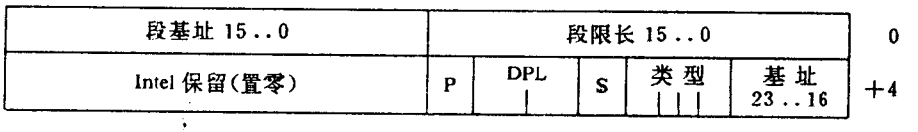
为了使得 286 与 386 的操作系统能够兼容，386 支持所有的 286 段描述符。图 1.4.5 给出了 286 系统段描述符的一般格式。286 与 386 的不同在于类型域、基地址域和限长值的不同。386 相比 286 来说有所扩展。286 系统段描符包含 24 位基地址和 16 位限长，而 386 系统段描述符具有 32 位基地址和 20 位限长，还有一个分片位。



门描述符域

类型	4	286 调用门
	5	任务门
	6	286 中断门
	7	286 陷井门
	C	386 调用门
	E	386 中断门
	F	386 陷井门
P	0	描述符内容无效
	1	描述符内容有效
DPL	任务访问门描述符的最低优先权	
字计数	原过程向调用过程的堆栈拷贝参数个数(最多 32 个)	
选择符	目标代码段的选择符	
	或指向 TSS 的选择符	
偏移量	16 位为 286，32 位为 386	

图 1.4.4　门描述符格式



基址	段的基准地址
限长	段的长度
P	存在位
DPL	描述符优先级 0—3
S	系统段描述符
类型	段的类型

图 1.4.5　80286 代码段和数据段描述符

为了支持 286 系统段，80386 可以在 386 操作系统下运行 286 应用程序，因为 386 CPU 能够自动分辨哪些是 286 描述符，哪些是 386 描述符，只要描述符的高两字节为 0，则这个描述符一定是 286 类型描述符。

286 描述符与 386 描述符的另一个区别在于字计数域和 B 位，对于 286 调用门来说，字计数是以双字节为单位(16 位)计数，而对于 386 调用门来说，是以四个字节为单位(32

位)计数，B 位控制调用门压入参数的大小。如果 B=0，压人 16 位，如果 B=1，则压入 32 位的参数。

1.5　任务切换及保护机制

386 支持多任务。这就是说，386 从硬件芯片上实现支持多个任务的存在并且支持多个任务分时执行。多任务分时操作指的是在一个固定时间片上程序控制权从一个任务传递给另一个任务。例如，我们可以将多个可执行任务排列在一张首尾相接的表上，按时间片轮转执行多个任务，多任务分时操作由操作系统来实现。

前面我们已经定义过，任务是一个子程序的集合，这些子程序配合起来完成一个特定的功能。一个正在执行的任务称为一个进程，任务间的切换称为进程切换。386 提供了一种实现任务切换的机制。386 任务间切换速度相当快，例如在 16 MHZ 的 386 上进行任务切换，只需 19 μs。每个任务执行时间的大小可由用户或操作系统提供。

在上一节我们已经讲过，一个任务的逻辑地址空间可分为全局地址空间和局部地址空间，各个任务的局部逻辑地址空间均不相同，虽然它们有可能占用相同的物理地址空间，但换入换出技术避免了这种情况可能造成的混乱。全局地址空间可被每个任务共享，因此，一个任务可访问全局地址空间中的每一段。

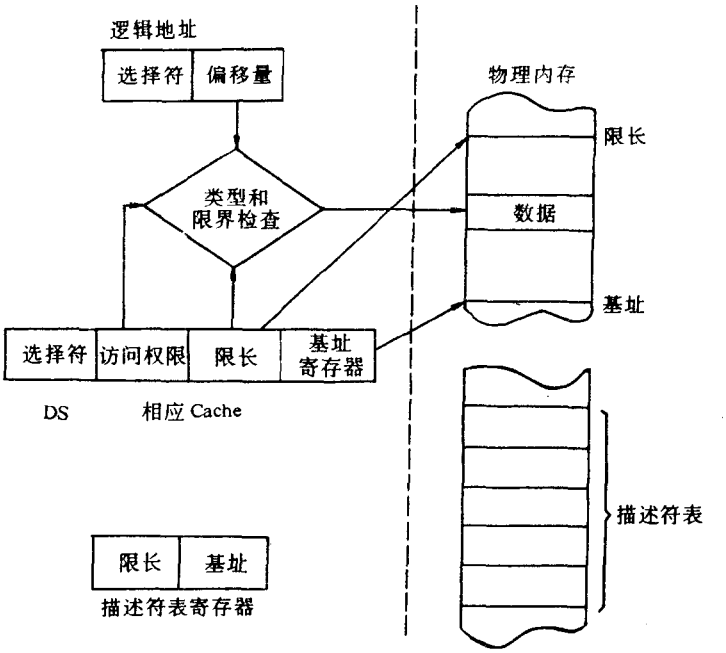
1.5.1　保护模式下的安全保护

在保护模式下，安全机制可以防止非法用户或错误的内存访问。80386 的硬件芯片实现了安全保护机制，这种机制限制一个任务对局部和系统资源的访问，并且在多任务环境中将各个任务独立区分开来。

段、页和描述符是 80386 保护机制的主要元素，我们已经知道，当使用一个段内存模式时，段是具有独立保护特权的虚似内存地址空间的最小元素，这些保护特权是由段描述符中的访问权限字节和地址限界构成。图 1.5.1 表示了 80386 硬件芯片提供的几种保护检测机制，这些保护机制在内存访问时将被激活。这些保护机制检测类型包括：类型检测、限界检测、地址领域限制、进程接口限制和指令限制。例如，当一个数据要在内存中存贮时，386 保护机制将检测段的访问权限字节中的类型域，以便确保它的保护特权与将要装入的高速缓存寄存器相一致，限界检测确保它的访问是在段的界限之内。

我们来看看保护权限的具体内容。保护权限反映在段描述符中，包括段访问信息。图 1.5.2 具体说明了段描述符和保护权限的内容。存在位 P 定义了一个段当前是否正在物理内存之中。类型域的第 4 位代表这个段是代码段还是数据段，参见表 1.4.1。如果这一位是 0，代表此段为数据段；如果为 1，代表此段为代码段。段特权位，如可读、可写、相容、上下扩展和访问位由类型域的其它位表示，如图上注释所示。DPL 域代表优先权的大小，有关 DPL 的重要作用将在本节后面部分详叙。

前面我们已经说明，当一个段被访问时，80386 自动将其基地址和限长装入高速缓存寄存器。在图 1.5.1 中，我们看出，访问权限信息也同时装入了高速缓存寄存器中。在装入之前，80386 的内存管理部件自动检查被访问的当前物理内存段是否和段选择符寄存器相一致，其中，CS 代表代码段，DS、ES、FS、GS 和 SS 代表数据段，最后核查段访问区域是



(a)

类型检查
限界检查
寻址区域限制
过程进入点限制
指令集限制

(b)

图　1.5.1

(a) 检查描述符访问权限；　(b) 保护检查和限制

否越界，一但出现非法访问等情况，系统内部将自动标志出错信息，一般出错有以下几种情况。

(1) 代码段和数据段相混淆，例如 CS 寄存器指向数据段描述符。

(2) 非法读写，例如用户程序试图读取一段标志为不可读的代码段。

(3) 越界访问，例如用户程序访问偏移量超出段界限等。

80386 为每个任务提供四种优先级，分别称为 0，1，2，3，其中优先级 0 为最高优先级，3 为最低优先级，值越大，优先级越小。如图 1.5.3 所示，系统软件和应用软件典型地分层表示出来，系统内核优先级为 0，它是面向微处理器硬件芯片的第一层，提供 I/O 控制、任务切换(进程管理)、内存管理等 OS 核心功能。系统服务为第二层，优先级为 1，提供诸如文件管理等功能。用户扩展例程为第三层，优先级为 2，由用户特定用途的例程组成，为用户提供最近、最优服务。应用层为第四层，优先级为 3，用户程序在这一层上运行。以上前三层为系统层，第四层为应用层。一个任务可以运行系统层提供的系统服务，因而一个任务运行期间可以具有不同的优先级，但是任务在运行期间不能更改这些系统例程，这里隐含地说明了一点，即低优先级代码可以调用高优先级的代码，这将在后面详细