任务管理

4.1 任务管理概述

任务是处理器可以调度、执行和暂停的工作单元，它可用于执行程序、任务或进程、操作系统服务实用程序、中断或异常处理程序、内核或执行实用程序。

IA-32 架构提供了一种机制，用于保存任务状态、调度任务执行以及从一个任务切换到另一个任务。在受保护模式下运行时，所有处理器执行都在任务内进行，即使是简单的系统也必须定义至少一个任务。更复杂的系统可以使用处理器的任务管理功能来支持多任务应用程序。

4.1.1 任务的结构

任务由两部分组成：任务执行空间和任务状态段 (TSS)。任务执行空间由代码段、堆栈段和一个或多个数据段组成（见图 7-1）。如果操作系统或执行程序使用处理器的特权级保护机制，则任务执行空间还为每个特权级提供单独的堆栈。

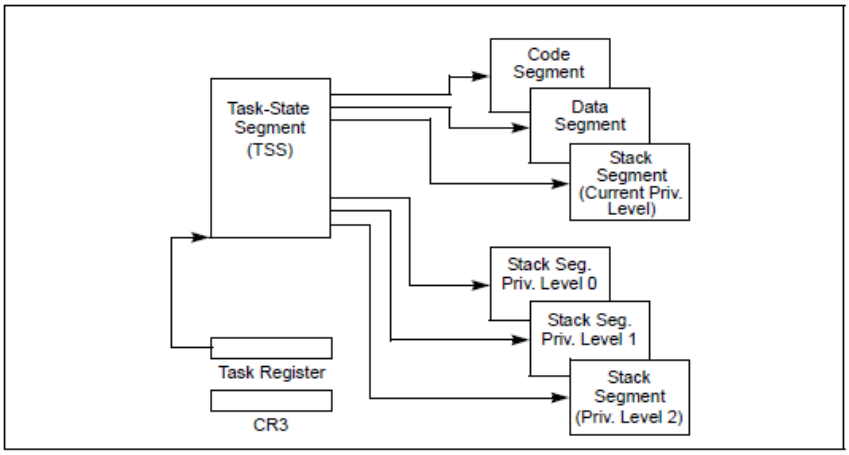

Figure 7-1. Structure of a Task

TSS 指定组成任务执行空间的段，并提供任务状态信息的存储空间。在多任务系统中，TSS 还提供链接任务的机制。

任务由其 TSS 的段选择器标识。当将任务加载到处理器中执行时，TSS 的段选择器、基址、限制和段描述符属性将加载到任务寄存器中。

如果为任务实现了分页，则任务使用的页目录的基址将加载到控制寄存器 CR3 中。

4.1.2 任务状态

以下项目定义当前正在执行的任务的状态：

- 任务的当前执行空间，由段寄存器 (CS、DS、SS、ES、FS 和 GS) 中的段选择器定义。
- 通用寄存器的状态。
- EFLAGS 寄存器的状态。
- EIP 寄存器的状态。
- 控制寄存器 CR3 的状态。
- 任务寄存器的状态。
- LDTR 寄存器的状态。
- I/O 映射基址和 I/O 映射 (包含在 TSS 中)。
- 指向特权 0、1 和 2 堆栈的堆栈指针 (包含在 TSS 中)。
- 链接到先前执行的任务 (包含在 TSS 中)。

在分派任务之前，除任务寄存器的状态外，所有这些项目都包含在任务的 TSS 中。此外，LDTR 寄存器的完整内容不包含在 TSS 中，只有 LDT 的段选择器包含在其中。

4.1.3 任务的执行

件或处理器可以通过以下方式之一调度任务以执行：

- 使用 CALL 指令显式调用任务。
- 使用 JMP 指令显式跳转到任务。
- 隐式调用（由处理器）中断处理程序任务。
- 隐式调用异常处理程序任务。
- 当 EFLAGS 寄存器中的 NT 标志设置时返回（使用 IRET 指令启动）。

所有这些调度任务的方法都使用指向任务门或任务 TSS 的段选择器来标识要调度的任务。当使用 CALL 或 JMP 指令调度任务时，指令中的选择器可以直接选择 TSS，也可以选择保存 TSS 选择器的任务门。当调度一个任务来处理中断或异常时，中断或异常的 IDT 条目必须包含一个任务门，该任务门保存中断或异常处理程序 TSS 的选择器。

当任务被调度执行时，当前正在运行的任务和调度的任务之间会发生任务切换。在任务切换期间，当前正在执行的任务的执行环境（称为任务状态或上下文）保存在其 TSS 中，并且任务的执行被暂停。调度任务的上下文随后被加载到处理器中，该任务的执行从新加载的 EIP 寄存器指向的指令开始。如果自系统上次初始化以来该任务尚未运行，则 EIP 将指向该任务代码的第一条指令；否则，它将指向该任务上次活动时执行的最后一条指令之后的下一条指令。

如果当前正在执行的任务（调用任务）调用了正在调度的任务（被调用任务），则调用任务的 TSS 段选择器存储在被调用任务的 TSS 中，以提供返回到调用任务的链接。

对于所有 IA-32 处理器，任务都不是递归的，任务不能调用或跳转到自身。

可以通过将任务切换到处理程序任务来处理中断和异常。在这里，处理器执行任务切换来处理中断或异常，并在从中断处理程序任务或异常处理程序任务返回时自动切换回中断的任务。此机制还可以处理中断任务期间发生的中断。

作为任务切换的一部分，处理器还可以切换到另一个 LDT，允许每个任务对基于 LDT 的段具有不同的逻辑到物理地址映射。页面目录基址寄存器 (CR3) 也在任务切换时重新加载，允许每个任务拥有自己的一组页表。这些保护设施有助于隔离任务并防止其相互干扰。

如果不使用保护机制，处理器将不提供任务之间的保护。即使操作系统使用多个特权级别进行保护，情况也是如此。在特权级别 3 上运行的任务使用与其他特权级别 3 任务相同的 LDT 和页表，可以访问代码和损坏的数据以及其他任务的堆栈。

使用任务管理设施来处理多任务应用程序是可选的。多任务可以通过软件处理，每个软件定义的任务都在单个 IA-32 架构任务的上下文中执行。

4.2 任务的数据结构

处理器定义了五种数据结构来处理与任务相关的活动：

- 任务状态段 (TSS)。
- 任务门描述符。
- TSS 描述符。
- 任务寄存器。
- EFLAGS 寄存器中的 NT 标志。

在保护模式下操作时，必须为至少一个任务创建 TSS 和 TSS 描述符，并且必须将 TSS 的段选择器加载到任务寄存器中（使用 LTR 指令）。

4.2.1 任务状态段（TSS）

恢复任务所需的处理器状态信息保存在称为任务状态段 (TSS) 的系统段中。图 7-2 显示了为 32 位 CPU 设计的任务的 TSS 格式。TSS 的字段分为两大类：动态字段和静态字段。

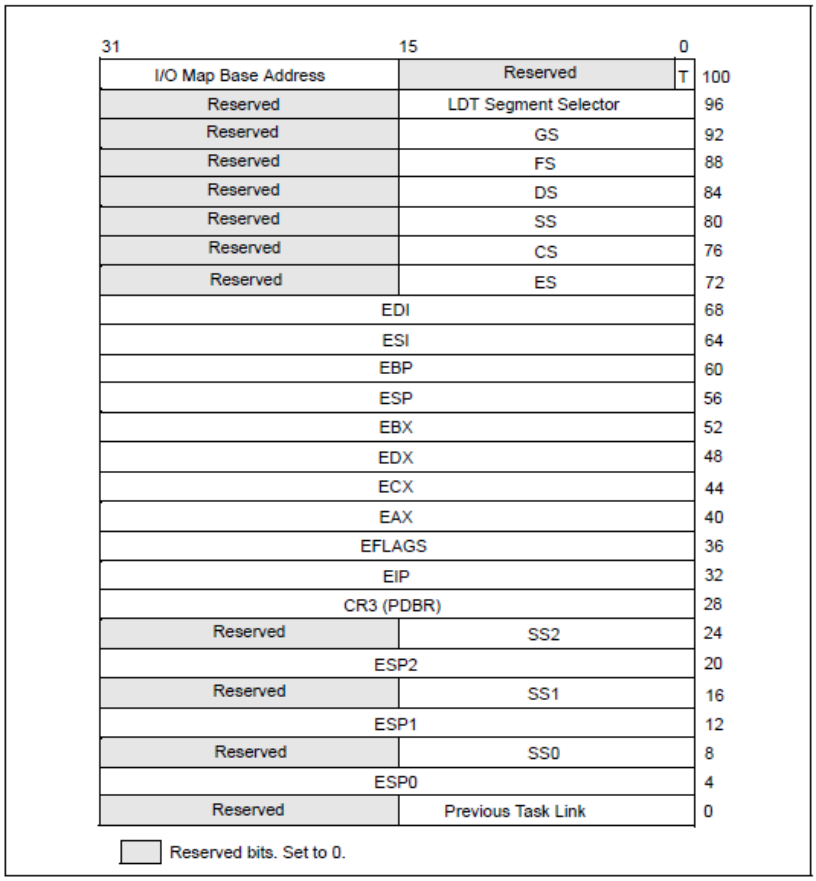

Figure 7-2. 32-Bit Task-State Segment (TSS)

4.2.2 TSS 描述符

与所有其他段一样，TSS 由段描述符定义。图 7-3 显示了 TSS 描述符的格式。TSS 描述符只能放在 GDT 中；不能放在 LDT 或 IDT 中。

如果尝试使用设置了 TI 标志（指示当前 LDT）的段选择器访问 TSS，则在 CALL 和 JMP 期间会导致生成一般保护异常 (#GP)；在 IRET 期间会导致无效 TSS 异常 (#TS)。如果尝试将 TSS 的段选择器加载到段寄存器中，也会生成一般保护异常。

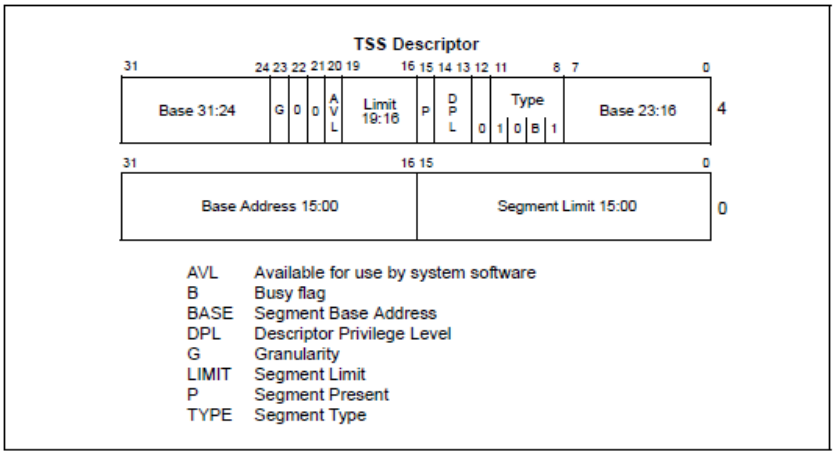

Figure 7-3. TSS Descriptor

类型字段中的忙标志 (B) 指示任务是否忙，忙任务当前正在运行或暂停。值为 1001B 的类型字段表示非活动任务；值为 1011B 表示忙任务。任务不是递归的，处理器使用忙碌标志来检测调用已被中断的任务的尝试。为了确保只有一个忙碌标志与任务相关联，每个 TSS 应该只有一个指向它的 TSS 描述符。

基数、限制和 DPL 字段以及粒度和存在标志的功能与它们在数据段描述符中的用途类似。当 32 位 TSS 的 TSS 描述符中的 G 标志为 0 时，限制字段的值必须等于或大于 67H，比 TSS 的最小大小小一个字节。尝试切换到 TSS 描述符的限制小于 67H 的任务会产生无效 TSS 异常 (#TS)。如果包含 I/O 权限位图或操作系统存储了其他数据，则需要更大的限制。处理器在任务切换时不会检查是否限制大于 67H；但是，它会在访问 I/O 权限位图或中断重定向位图时进行检查。

任何可以访问 TSS 描述符的程序或过程（即其 CPL 在数值上等于或小于 TSS 描述符的 DPL）都可以通过调用或跳转来调度任务。

在大多数系统中，TSS 描述符的 DPL 设置为小于 3 的值，因此只有特权软件才能执行任务切换。但是，在多任务应用程序中，某些 TSS 描述符的 DPL 可以设置为 3，以允许在应用程序（或用户）特权级别进行任务切换。

4.2.3 任务寄存器

任务寄存器保存当前任务的 TSS 的 16 位段选择器和整个段描述符（32 位基址（IA-32e 模式下为 64 位）、16 位段限制和描述符属性）。此信息是从当前任务的 GDT 中的 TSS 描述符复制而来的。图 7-5 显示了处理器用于访问 TSS 的路径（使用任务寄存器中的信息）。

任务寄存器具有可见部分（可由软件读取和更改）和不可见部分（由处理器维护，软件无法访问）。可见部分中的段选择器指向 GDT 中的 TSS 描述符。处理器使用任务寄存器的不可见部分来缓存 TSS 的段描述符。将这些值缓存在寄存器中可提高任务的执行效率。LTR（加载任务寄存器）和 STR（存储任务寄存器）指令加载并读取任务寄存器的可见部分：

LTR 指令将段选择器（源操作数）加载到指向 GDT 中的 TSS 描述符的任务寄存器中。然后，它将来自 TSS 描述符的信息加载到任务寄存器的不可见部分。LTR 是一条特权指令，仅当 CPL 为 0 时才可执行。它在系统初始化期间用于将初始值放入任务寄存器中。之后，当发生任务切换时，任务寄存器的内容会隐式更改。

STR（存储任务寄存器）指令将任务寄存器的可见部分存储在通用寄存器或内存中。此指令可由在任何特权级别运行的代码执行，以识别当前正在运行的任务。但是，它通常仅由操作系统软件使用。

在处理器上电或重置时，段选择器和基址设置为默认值 0；限制设置为 FFFFH。

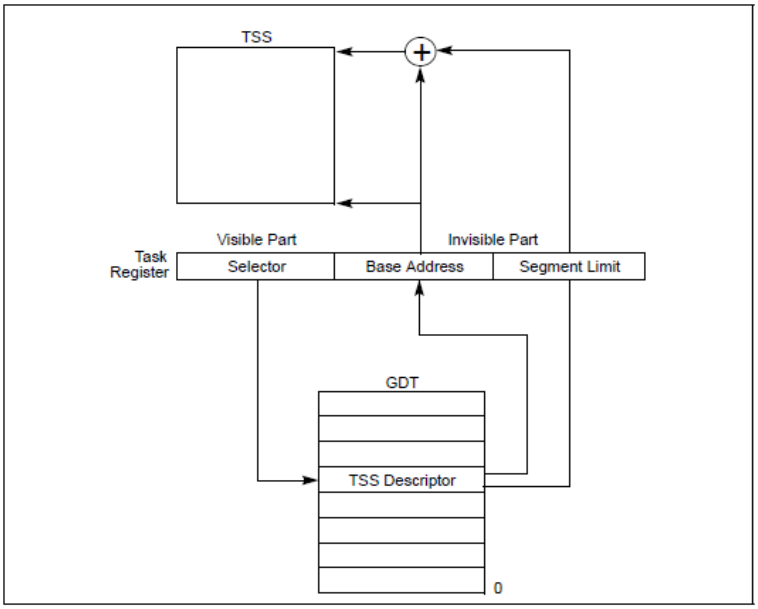

Figure 7-5. Task Register

4.2.4 任务门描述符

任务门描述符提供对任务的间接、受保护引用（见图 7-6）。它可以放在 GDT、LDT 或 IDT 中。任务门描述符中的 TSS 段选择器字段指向 GDT 中的 TSS 描述符。此段选择器中的 RPL 不被使用。

任务门描述符的 DPL 控制任务切换期间对 TSS 描述符的访问。当程序或过程通过任务门调用或跳转到任务时，指向任务门的门选择器的 CPL 和 RPL 字段必须小于或等于任务门描述符的 DPL。请注意，当使用任务门时，不使用目标 TSS 描述符的 DPL。

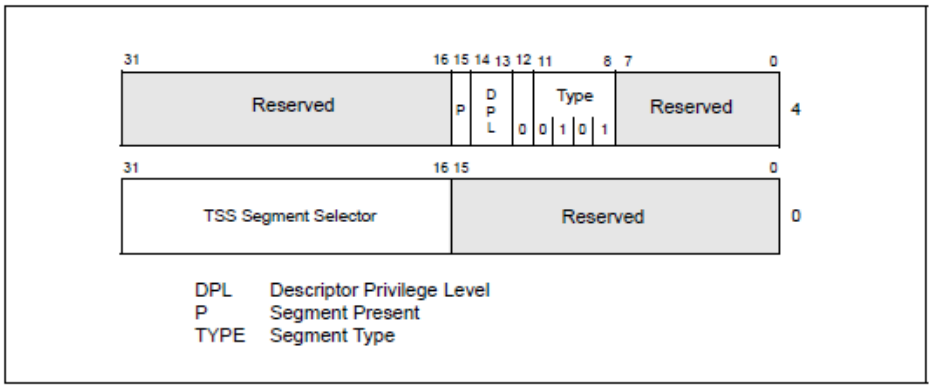

Figure 7-6. Task-Gate Descriptor

4.3 任务切换

处理器在以下四种情况之一中将执行转移到另一个任务：

- 当前程序、任务或过程执行 JMP 或 CALL 指令，指向 GDT 中的 TSS 描述符。
- 当前程序、任务或过程执行 JMP 或 CALL 指令，指向 GDT 或当前 LDT 中的任务门描述符。
- 当 EFLAGS 寄存器中的 NT 标志被设置时，当前任务执行 IRET。

JMP、CALL 和 IRET 指令以及中断和异常都是重定向程序的机制。TSS 描述符或任务门的引用（调用或跳转到任务时）或 NT 标志的状态（执行 IRET 指令时）决定是否发生任务切换。

处理器在切换到新任务时执行以下操作：

（1）从任务门或上一个任务链接字段（对于使用 IRET 指令启动的任务切换）获取新任务的 TSS 段选择器作为 JMP 或 CALL 指令的操作数。

（2）检查当前（旧）任务是否允许切换到新任务。数据访问权限规则适用于 JMP 和 CALL 指令。当前（旧）任务的 CPL 和新任务的段选择器的 RPL 必须小于或等于所引用的 TSS 描述符或任务门的 DPL。无论目标任务门或 TSS 描述符的 DPL 如何，异常、中断（INT n 指令生成的中断除外）和 IRET 指令都可以切换任务。对于由 INT n 指令生成的中断，将检查 DPL。

（3）检查新任务的 TSS 描述符是否标记为存在且具有有效限制（大于或等于 67H）。

（4）检查新任务是否可用（调用、跳转、异常或中断）或忙（IRET 返回）。

（5）检查当前（旧）TSS、新 TSS 以及任务切换中使用的所有段描述符是否已分页到系统内存中。

（6）如果任务切换是通过 JMP 或 IRET 指令启动的，则处理器会清除当前（旧）任务的 TSS 描述符中的忙 (B) 标志；如果是通过 CALL 指令、异常或中断启动的：忙 (B) 标志保持设置状态。（见表 7-2。）

（7）如果任务切换是通过 IRET 指令启动的，则处理器会清除 EFLAGS 寄存器临时保存映像中的 NT 标志；如果任务切换由 CALL 或 JMP 指令、异常或中断发起，则保存的

EFLAGS 映像中的 NT 标志保持不变。

（8）将当前（旧）任务的状态保存在当前任务的 TSS 中。处理器在任务寄存器中找到当前 TSS 的基址，然后将以下寄存器的状态复制到当前 TSS 中：所有通用寄存器、段寄存器中的段选择器、EFLAGS 寄存器的临时保存映像和指令指针寄存器 (EIP)。

（9）如果任务切换是由 CALL 指令、异常或中断发起的，则处理器将在从新任务加载的 EFLAGS 中设置 NT 标志。如果任务切换由 IRET 指令或 JMP 指令发起，则 NT 标志将反映从新任务加载的 EFLAGS 中的 NT 状态（参见表 7-2）。

（10）如果任务切换是由 CALL 指令、JMP 指令、异常或中断发起的，则处理器会在新任务的 TSS 描述符中设置忙 (B) 标志；如果是由 IRET 指令发起的，则忙 (B) 标志保持设置状态。

（11）将新任务的 TSS 的段选择器和描述符加载到任务寄存器中。

（12）TSS 状态加载到处理器中。这包括 LDTR 寄存器、PDBR（控制寄存器 CR3）、EFLAGS 寄存器、EIP 寄存器、通用寄存器和段选择器。加载此状态期间的故障可能会破坏架构状态。（如果未启用分页，则会从新任务的 TSS 读取 PDBR 值，但不会将其加载到 CR3 中。）

（13）与段选择器关联的描述符已加载并限定。与此加载和限定相关的任何错误都会在新任务的上下文中发生，并可能破坏架构状态。

（14）开始执行新任务。

Table 7-2. Effect of a Task Switch on Busy Flag, NT Flag, Previous Task Link Field, and TS Flag

Flag or Field	Effect of JMP instruction	Effect of CALL Instruction or Interrupt	Effect of IRET Instruction
Busy (B) flag of new task.	Flag is set. Must have been clear before.	Flag is set. Must have been clear before.	No change. Must have been set.
Busy flag of old task.	Flag is cleared.	No change. Flag is currently set.	Flag is cleared.
NT flag of new task.	Set to value from TSS of new task.	Flag is set.	Set to value from TSS of new task.
NT flag of old task.	No change.	No change.	Flag is cleared.
Previous task link field of new task.	No change.	Loaded with selector for old task's TSS.	No change.
Previous task link field of old task.	No change.	No change.	No change.
TS flag in control register CR0.	Flag is set.	Flag is set.	Flag is set.

成功切换任务时，始终会保存当前正在执行任务的状态。如果任务恢复，则从保存的 EIP 值指向的指令开始执行，并且寄存器将恢复为任务暂停时所持有的值。

切换任务时，新任务的特权级别不会从暂停的任务继承其特权级别。新任务开始在 CS 寄存器的 CPL 字段中指定的特权级别执行，该字段从 TSS 加载。由于任务由其单独的地址空间和 TSS 隔离，并且特权规则控制对 TSS 的访问，因此软件不需要在任务切换时执行显式特权检查。

每次发生任务切换时，控制寄存器 CR0 中的 TS（任务切换）标志都会被设置。系统软件使用 TS 标志来协调浮点单元在与处理器其余部分生成浮点异常时的操作。TS 标志表示浮点单元的上下文可能与当前任务的上下文不同。

4.4 任务链

TSS 的上一个任务链接字段（有时称为“反向链接”）和 EFLAGS 寄存器中的 NT 标志用于将执行返回到上一个任务。EFLAGS.NT = 1 表示当前执行的任务嵌套在另一个任务的执行中。

当 CALL 指令、中断或异常导致任务切换时：处理器将当前 TSS 的段选择器复制到新任务的 TSS 的上一个任务链接字段；然后设置 EFLAGS.NT = 1。

如果软件使用 IRET 指令暂停新任务，处理器将检查 EFLAGS.NT = 1；然后使用上一个任务链接字段中的值返回到上一个任务。参见图 7-8。

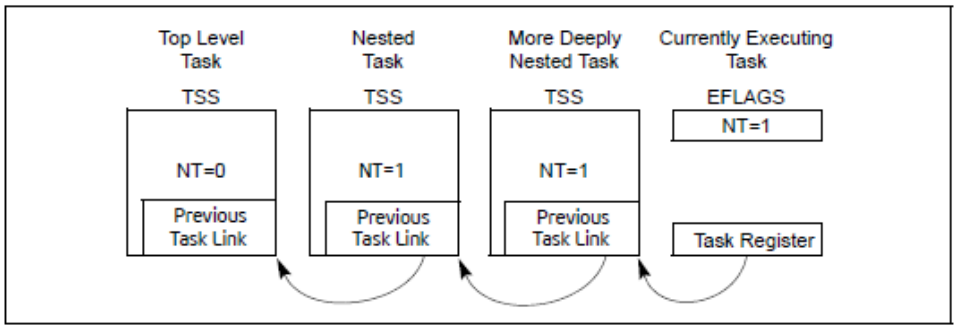

Figure 7-8. Nested Tasks

当 JMP 指令导致任务切换时，新任务不嵌套。不使用上一个任务链接字段，EFLAGS.NT = 0。当不需要嵌套时，使用 JMP 指令分派新任务。

表 7-2 显示了任务切换期间的忙标志（在 TSS 段描述符中）、NT 标志、上一个任务链接字段和 TS 标志（在控制寄存器 CR0 中）。

NT 标志可以由在任何特权级别执行的软件修改。程序可以设置 NT 标志并执行 IRET 指令。这可能会随机调用当前任务的 TSS 的上一个链接字段中指定的任务。为了防止此类虚假任务切换成功，操作系统应将其创建的每个 TSS 中的上一个任务链接字段初始化为 0。

4.4.1 使用忙碌标志来防止递归任务切换

TSS 只允许为一个任务保存一个上下文；因此，一旦调用（分派）任务，对该任务的递归（或重入）调用将导致任务的当前状态丢失。TSS 段描述符中的忙标志用于防止重入任务切换和随后的任务状态信息丢失。处理器按如下方式管理忙标志：

（1）分派任务时，处理器设置新任务的忙标志。

（2）如果在任务切换期间，当前任务被置于嵌套链中（任务切换由 CALL 指令、中断或异常生成），则当前任务的忙标志保持设置状态。

（3）切换到新任务（由 CALL 指令、中断或异常启动）时，如果新任务的忙标志已设置，处理器将生成通用保护异常 (#GP)。如果任务切换由 IRET 指令启动，则不会引发异常，因为处理器期望设置忙标志。

（4）当通过跳转到新任务（使用任务代码中的 JMP 指令启动）或通过任务代码中的 IRET 指令终止任务时，处理器将清除忙标志，使任务返回“不忙”状态。

处理器通过阻止任务切换到自身或嵌套任务链中的任何任务来防止递归任务切换。由于多次调用、中断或异常，嵌套挂起任务链可能会增长到任意长度。忙标志可防止调用此链中的任务。

忙标志可用于多处理器配置，因为处理器在设置或清除忙标志时遵循 LOCK 协议（在总线上或缓存中）。此锁定可防止两个处理器同时调用同一任务。

4.4.2 修改任务链接

在单处理器系统中，当需要从链接任务链中删除任务时，使用以下过程删除该任务：

（1）禁用中断。

（2）更改抢占任务（暂停要删除的任务的任务）的 TSS 中的上一个任务链接字段。假设抢占任务是要删除的任务链中的下一个任务（较新的任务）。将上一个任务链接字段更改为指向链中下一个最旧任务的 TSS 或指向链中更旧的任务。

（3）清除要从链中删除的任务的 TSS 段描述符中的忙 (B) 标志。如果要从链中删除多个任务，则必须清除要删除的每个任务的忙标志。

（4）启用中断。

在多处理系统中，必须向此过程添加额外的同步和序列化操作，以确保在更改前一个任务链接字段并清除忙标志时，TSS 及其段描述符都被锁定。

4.5 任务地址空间

任务的地址空间由任务可以访问的段组成。这些段包括 TSS 中引用的代码、数据、堆栈和系统段以及任务代码访问的任何其他段。这些段被映射到处理器的线性地址空间，而后者又被映射到处理器的物理地址空间（直接或通过分页）。

TSS 中的 LDT 段字段可用于为每个任务提供自己的 LDT。为任务提供自己的 LDT 允许将与任务相关的所有段的段描述符放置在任务的 LDT 中，从而将任务地址空间与其他任务隔离。多个任务也可以使用相同的 LDT。这是一种内存高效的方式，允许特定任务相互通信或控制，而不会放弃整个系统的保护屏障。由于所有任务都可以访问 GDT，因此也可以创建通过此表中的段描述符访问的共享段。

如果启用了分页，TSS 中的 CR3 寄存器 (PDBR) 字段允许每个任务拥有自己的一组页表，用于将线性地址映射到物理地址，或者多个任务可以共享同一组页表。

4.5.1 将任务映射到线性和物理地址空间

任务可以通过以下两种方式之一映射到线性地址空间和物理地址空间：

- 所有任务共享一个线性到物理地址空间映射。没有分页时，所有线性地址都映射到相同的物理地址。启用分页后，这种形式的线性到物理地址空间映射是通过对所有任务使用一个页面目录来获得的。如果支持按需分页虚拟内存，线性地址空间可能会超出可用的物理空间。
- 每个任务都有自己的线性地址空间，该空间映射到物理地址空间，这种映射形式是通过为每个任务使用不同的页面目录来实现的。由于 PDBR（控制寄存器 CR3）是在任务切换时加载的，因此每个任务可能具有不同的页面目录。

不同任务的线性地址空间可能映射到完全不同的物理地址。如果不同页面目录的条目指向不同的页表，并且页表指向物理内存的不同页面，则任务不共享物理地址。

无论使用哪种映射任务线性地址空间的方法，所有任务的 TSS 都必须位于物理空间的共享区域中，该区域可供所有任务访问。这种映射是必需的，以便处理器在任务切换期间读取和更新 TSS 时，TSS 地址的映射不会发生变化。GDT 映射的线性地址空间也应映射到物理空间的共享区域；否则，GDT 的目的就落空了。图 7-9 显示了两个任务的线性地址空间如何通过共享页表在物理空间中重叠。

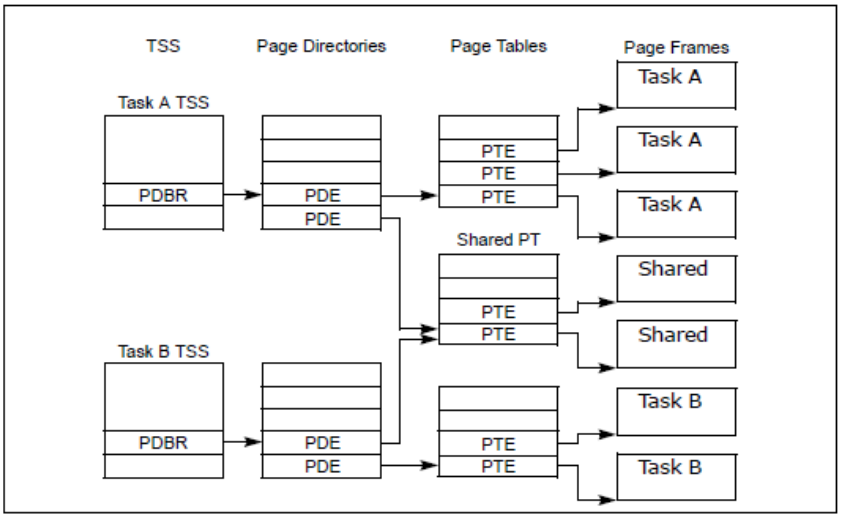

Figure 7-9. Overlapping Linear-to-Physical Mappings

4.5.2 任务逻辑地址空间

为了允许任务间共享数据，使用以下技术为数据段创建共享的逻辑到物理地址空间映射：

- 通过 GDT 中的段描述符 — 所有任务都必须能够访问 GDT 中的段描述符。如果 GDT 中的某些段描述符指向线性地址空间中的段，而这些段被映射到所有任务共用的物理地址空间区域，则所有任务都可以共享这些段中的数据和代码。
- 通过共享 LDT — 如果两个或多个任务的 TSS 中的 LDT 字段指向同一个 LDT，则它们可以使用同一个 LDT。如果共享 LDT 中的某些段描述符指向映射到物理地址空间公共区域的段，则这些段中的数据和代码可以在共享 LDT 的任务之间共享。这种共享方法比通过 GDT 共享更具选择性，因为共享可以限制在特定任务中。系统中的其他任务可能具有不同的 LDT，这些 LDT 不允许它们访问共享段。
- 通过不同 LDT 中的段描述符（这些段描述符映射到线性地址空间中的公共地址）— 如果线性地址空间的这个公共区域映射到每个任务的物理地址空间的相同区域，则这些段描述符允许任务共享段。此类段描述符通常称为别名。这种共享方法比上面列出的方法更具选择性，因为 LDT 中的其他段描述符可能指向不共享的独立线性地址。