# 任务管理

## 任务管理概述

**任务**是处理器可以分派、执行或挂起的**工作单元**；它可以用来执行一个程序，任务、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体系结构提供了一种机制，用于保存任务的状态、调度任务、切换任务；

在**保护模式**下执行时，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支持多任务应用程序的任务管理工具。

## 任务的结构

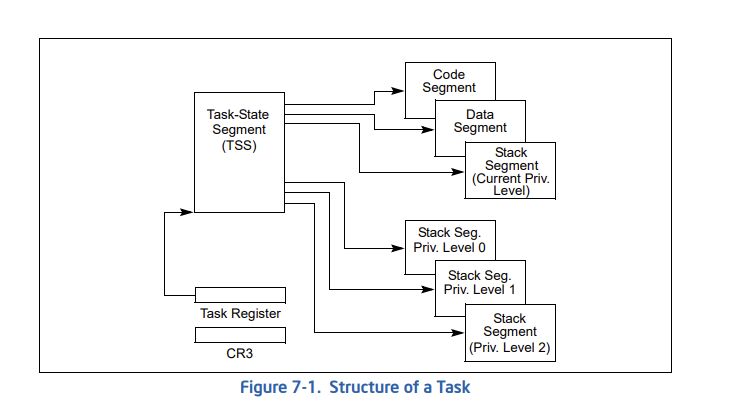
任务由两部分组成:**任务执行空间**和**任务状态段**(TSS)。

任务执行空间由**代码段、堆栈段和一个或多个数据段**组成，如图所示。

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别提供单独的堆栈。

**TSS指定组成任务执行空间的段，并为任务状态信息提供存储位置**。

在多任务系统中，TSS还提供了一种连接任务的机制。任务由其TSS的段选择器标识。当一个任务被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被加载到任务寄存器中，如果为任务实现分页，则将任务使用的页目录的基址加载到控制寄存器CR3中。



## 任务状态

以下项定义当前正在执行的任务的**状态**:

1.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。

2.通用寄存器的状态。

3.EFLAGS寄存器的状态。

4.EIP寄存器的状态。

5.控制寄存器CR3的状态。

6.任务寄存器的状态。

7.LDTR寄存器的状态。

8.I/O映射基址和I/O映射(包含在TSS中)。

9.指向特权0、1和2堆栈的堆栈指针(包含在TSS中)。

10.链接到先前执行的任务(包含在TSS中)。

在调度任务之前，除了任务寄存器的状态外，所有这些项都包含在任务的TSS中。此外，LDTR寄存器的完整内容不包含在TSS中，只包含LDT的段选择器。

## 任务的执行

软件或处理器可以通过下列方式之一**调度任务执行**:

1.用call指令对任务的显式调用。

2.使用JMP指令显式跳转到任务。

3.(由处理器)对中断处理程序任务的隐式调用。

4.对异常处理程序任务的隐式调用。

5.当设置了EFLAGS寄存器中的NT标志时返回(由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的TSS的段选择器来标识要调度的任务。

当用CALL或JMP指令调度任务时，指令中的选择器可以直接选择TSS，也可以选择一个为TSS保留选择器的任务门。当调度任务来处理中断或异常时，用于中断或异常的IDT项必须包含一个任务门，该任务门持有中断或异常处理程序TSS的选择器。

当任务被调度执行时，会在当前运行的任务和被调度的任务之间发生**任务切换**。在任务切换期间，当前执行任务的执行环境(称为任务的状态或上下文)保存在其TSS中，并暂停任务的执行。然后将已分派任务的上下文加载到处理器中，该任务的执行从新加载的EIP寄存器所指向的指令开始。如果自系统上次初始化以来**任务还没有运行过**，那么EIP将指向任务代码的第一条指令;否则，它将指向任务上次活动时执行的最后一条指令之后的下一条指令。

如果当前执行的任务(调用任务)**调用正在分派的任务**(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，以提供回调用任务的链接。

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己。中断和异常可以通过任务切换到处理程序任务来处理。

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被中断的任务。该机制还可以处理在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。页目录基寄存器(CR3)也在任务开关上重新加载，从而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它们相互干扰。

如果不使用保护机制，处理器就不会在任务之间提供保护。即使对于使用多个特权级别进行保护的操作系统也是如此。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，可以访问代码并破坏数据和其他任务的堆栈。

使用任务管理工具来处理多任务应用程序是可选的。多任务可以在软件中处理，每个软件定义的任务在单个IA-32架构任务的上下文中执行。

## 任务的数据结构

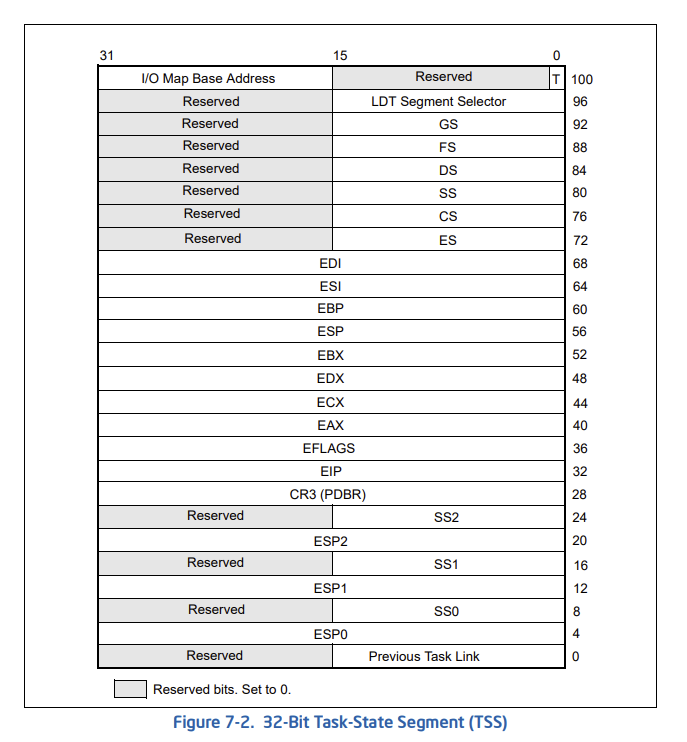
处理器定义了以下寄存器和数据结构来处理与任务相关的活动：

* 任务状态段 TSS
* TSS 描述符
* 任务寄存器 TR
* 任务门描述符
* EFLAGS 寄存器中的 NT 标志

在保护模式下运行时，必须为至少一个任务创建 TSS 和 TSS 描述符，并且必须使用 LTR 指令将 TSS 的段选择器加载到任务寄存器中。

### TSS

用于恢复一个任务执行的处理器状态信息被保存在TSS段中：

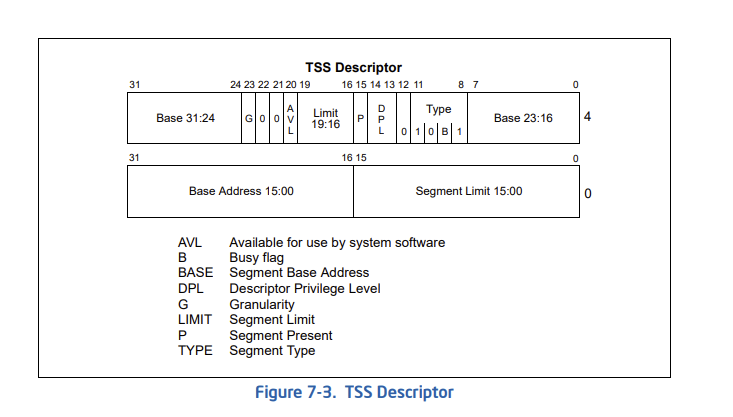


这些字段可以被分为两大类：

* **动态字段**。当任务在任务切换期间挂起时，处理器更新动态字段的内容
  + 通用寄存器字段。用于保存 EAX、ECX、EDX、EBX、ESP、EBP、ESI 和 EDI 寄存器的内容
  + 段选择器字段。保存 ES、CS、SS、DS、FS 和 GS 寄存器的内容
  + EFLAGS 寄存器域。EFLAGS 寄存器在任务切换之前的状态
  + 指令指针 EIP 字段。任务切换之前 EIP 寄存器的状态
  + 先前任务链接字段。包含前一个任务 TSS 段选择器（在调用、中断或异常启动的任务切换时更新）。该字段允许使用 IRET 指令将任务切换回先前的任务
* **静态字段**。在创建任务时设置，处理器在读取静态字段时通常不会修改它们
  + LDT 段选择器字段。包含任务的 LDT 的段选择器。
  + CR3 控制寄存器字段。包含任务要使用的页面目录的物理基地址。控制寄存器 CR3 也称为页目录基址寄存器 PDBR
  + 特权级为 0、1 和 2 的堆栈指针字段。这些堆栈指针由一个逻辑地址组成，该逻辑地址由堆栈段（SS0、SS1 和 SS2）的段选择器和堆栈中的偏移量（ESP0 、ESP1 和 ESP2）组成。请注意，这些字段中的值对于特定任务是静态的，但如果任务中发生了堆栈切换，SS 和 ESP 值将发生变化
  + 调试陷阱 T 标志字段。当设置该位时，T 标志会导致处理器在从其他任务切换到此任务时引发调试异常
  + I/O 映射基址字段。该字段包含从 TSS 基址到 I/O 许可位图和中断重定向位图的 16 位偏移量。如果存在，这些映射将存储在 TSS 中的更高地址处。 I/O 映射基地址指向 I/O 权限位图的开始和中断重定向位图的结束位置

### TSS描述符

TSS描述符只能放在GDT中，而不能放在LDT或者IDT中：



* TYPE 中的忙标志 B 指示任务是否忙：一个忙的任务是指该任务当前正在运行或者被挂起。B的值如果为 1001B ，则表示该任务是一个非活动任务；B的值 1011B 则表示任务是忙的。由于任务不是递归的，处理器使用B标志来程序检测调用已被中断执行的任务的操作。为了确保只有一个B标志与任务相关联，每个 TSS 都应该只有一个指向它的 TSS 描述符。
* base、limit 和 DPL 字段以及 granularity 和 present 标志的功能类似于它们在数据段描述符中的作用。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 0x67H，这比 TSS 的规定最小尺寸小一个字节。如果尝试切换到 TSS 描述符限制小于 0x67H 的任务，则会生成无效 TSS 异常 (invalid-TSS exception, #TS)。如果包含 I/O 权限位图或操作系统存储额外数据，那么需要更大的限制。处理器不会在任务切换时检查大于 0x67H 的限制，但它会在访问 I/O 权限位图或中断重定向位图时检查该值。
* 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分派任务。
* 在大多数系统中，TSS 描述符的 DPL 被设置为小于 3 的值。因此，只有特权级软件（privileged software）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 可能会设置为 3，以允许在应用程序（或用户）特权级别进行任务切换操作

### 任务寄存器

任务寄存器 TR 中存放着 16 位的段选择符以及当前任务 TSS 段的整个描述符。这些信息是从 GDT 中当前任务的 TSS 描述符中复制过来的。

### 任务门描述符

任务门描述符提供对任务的间接、受保护的引用，可以放在 GDT、LDT 或 ID 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。在这种段选择器中并没有使用 RPL。任务门描述符的 DPL 在任务切换期间控制对 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。值得注意的是，当使用任务门时，不会使用目标 TSS 描述符的 DPL。

## 任务切换

处理器可使用以下 4 种方式之一执行任务切换：

* 当前任务对 GDT 中的 TSS 描述符执行 JMP 或 CALL 指令
* 当前任务对 GDT 或当前 LDT 中的任务门描述符执行 JMP 或 CALL 指令
* 中断或异常向量指向 IDT 中的任务门描述符
* 当前任务在设置 EFLAGS 寄存器中的 NT 标志时执行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：

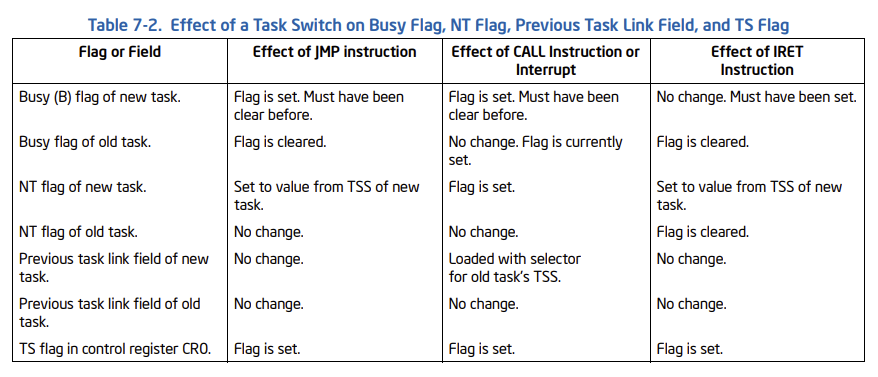
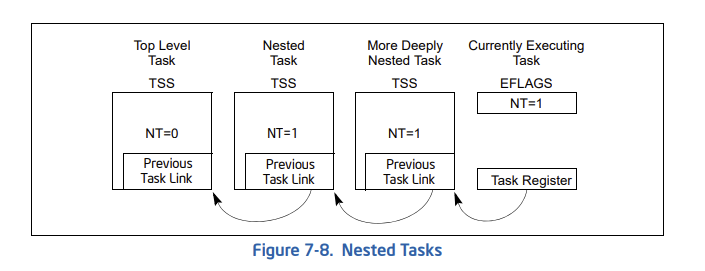
1. 从任务门或先前的任务链接字段（对于使用 IRET 指令启动的任务切换），获取新任务的 TSS 段选择器作为 JMP 或 CALL 指令的操作数。
2. 检查是否允许当前任务切换到新任务。数据访问权限规则适用于 JMP 和 CALL 指令。当前任务的 CPL 和新任务的段选择器的 RPL 必须小于或等于被引用的 TSS 描述符或任务门的 DPL。异常、中断（由 INT n 指令生成的中断除外）和 IRET 指令允许切换任务，而不管目标任务门或 TSS 描述符的 DPL。对于 INT n 指令产生的中断，需要检查 DPL。
3. 检查新任务的 TSS 描述符是否标记为存在（present）并且具有有效限制值（大于或等于 67H）。
4. 如果任务切换产生自 JMP 或 IRET 指令，处理器就会把当前任务 TSS 描述符中的忙标志 B 复位。
5. 如果任务切换自 IRET，则会把临时保存的 EFLAGS 中的 NT 复位。
6. 在当前任务的 TSS 中保存当前任务的状态。处理器在任务寄存器中找到当前 TSS 的基地址，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，来自段寄存器的段选择器，EFLAGS 寄存器的临时保存映像和指令指针寄存器 EIP。
7. 如果任务切换是通过 CALL 指令、JMP 指令、异常或中断启动的，则处理器会在新任务的 TSS 描述符中设置 B 标志；如果使用 IRET 指令启动，则 B 标志保持设置状态。
8. 如果任务切换由CALL、JMP 指令或者异常或中断产生，处理器就会设置新任务 TSS 描述符中的忙标志B。如果任务切换由 IRET 产生，则不去改动 B 标志。
9. 将新任务的 TSS 的段选择器和描述符加载到任务寄存器中。设置 CR0 寄存器的 TS 标志。
10. 加载新任务的 TSS 状态。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体系结构状态。
11. 开始执行新任务

## 任务链

TSS 的前一个任务链接字段和 EFLAGS 寄存器中的 NT 标志用于将处理器的执行返回到前一个任务。 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 的段选择符复制到新任务的 TSS 的前一个任务链接字段；然后设置 EFLAGS.NT = 1。如果软件使用 IRET 指令暂停新任务，处理器检查 EFLAGS.NT = 1；然后它使用上一个任务链接字段中的值返回到上一个任务。

当 JMP 指令导致任务切换时，新任务不会嵌套，也就是不会使用先前的任务链接字段且 EFLAGS.NT = 0。因此，当任务不需要嵌套时，使用 JMP 指令分派新任务。



## 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，而线性地址空间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其自己的 LDT。通过将与任务关联的所有段的段描述符放置在任务的 LDT 中，为任务提供自己的 LDT ，可以允许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，而不会降低整个系统的保护屏障。

因为所有任务都可以访问 GDT，所以也可以创建可以通过 GDT 中的段描述符访问的共享段。

如果启用分页，则 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。

可以通过以下两种方式之一将任务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。 未启用分页时，这是唯一的选择。在没有分页的情况下，所有线性地址都映射到相同的物理地址。当启用分页时，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。如果支持请求分页的虚拟内存，那么线性地址空间可能会超过可用的物理空间。
* 每个任务都有自己的线性地址空间。 这种形式的映射是通过为每个任务使用不同的页面目录来完成的。因为PDBR（控制寄存器CR3）加载在任务开关上，每个任务可能有不同的页目录。

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。如果不同页目录的条目指向不同的页表，而页表又指向物理内存的不同页，那么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，所有任务都可以访问该区域。此映射是必需的，以便在任务切换期间，处理器在读取和更新 TSS 时，TSS 地址的映射不会更改。 GDT 映射的线性地址空间也应该映射到物理空间的共享区域；否则，GDT 的目的就落空了。下图展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。

为了在任务之间共享数据，可使用下列方法之一来为数据段建立共享的逻辑到物理地址空间的映射：

* 通过使用 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，这些段映射到所有任务共有的物理地址空间区域，那么所有任务都可以共享这些段中的数据和代码。
* 通过共享的 LDT。如果两个或多个任务的TSS 中的 LDT 字段指向同一个LDT，那么两个或多个任务可以使用同一个 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 共享更加值得选择，因为共享可以限于特定任务。系统中的其他任务可能有不同的 LDT，而这些 LDT 不允许它们访问共享段。
* 通过映射到线性地址空间公共地址区域的不同 LDT 中的段描述符。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，这些段描述符允许任务共享段。这样的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加值得选择，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。