2.1 SYSTEM ARCHITECTURE OVERVIEW x86 系统架构概览

This chapter provides a description of each part of this architecture. It also describes the system registers that are used to set up and control the processor at the system level and gives a brief overview of the processor's system-level (operating system) instructions.

All Intel 64 and IA-32 processors enter real-address mode following a power-up or reset (see Chapter 10, "Processor Management and Initialization"). Software then initiates the switch from real-address mode to protected mode. If IA-32e mode operation is desired, software also initiates a switch from protected mode to IA-32e mode.

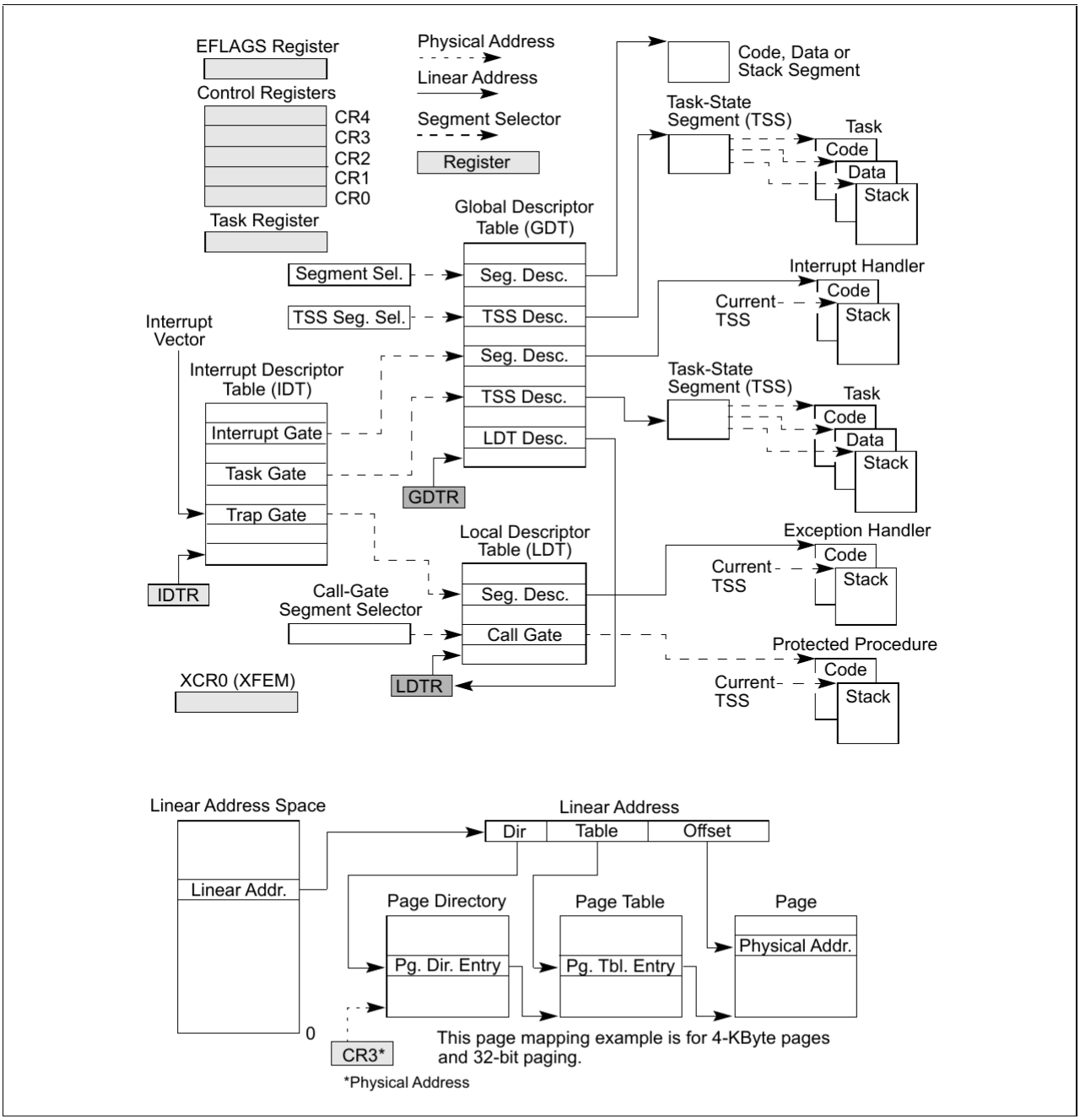


Figure 2-1. IA-32 System-Level Registers and Data Structures

关于 GDT: 【构建操作系统】全局描述符表GDT - 知乎 (zhihu.com)

> **全局描述符表 GDT（Global Descriptor Table）** 在整个系统中，全局描述符表 GDT 只有一张 (一个处理器对应一个 GDT)，GDT 可以被放在内存的任何位置，但 CPU 必须知道 GDT 的入口，也就是基地址放在哪里，Intel 的设计者门提供了一个**寄存器 GDTR**用来存放 GDT 的入口地址，程序员将 GDT 设定在内存中某个位置之后，可以通过**LGDT 指令**将 GDT 的入口地址装入此寄存器，从此以后，CPU 就根据此寄存器中的内容作为 GDT 的入口来访问 GDT 了。GDTR 中存放的是 GDT 在内存中的基地址和其表长界限。

也就是说，GDT是全局的，存放在内存中的某个位置，而这个位置是由你来指定给CPU的，换句话说，你来钦定！

2.1.1 GDT 和 LDT

当在保护模式下，访问内存时必须通过全局描述符表 `GDT` 或*可选的*的本地描述符表 `LDT`.

这两个表的表项是段描述符。段描述符提供段的基址、访问权限、类型和用法信息。

每个段描述符都有一个关联的段选择符。段选择符为使用它的软件提供了一个索引到 GDT 或 LDT 中的偏移量（其关联的段描述符的偏移量），一个全局/本地标志（确定选择符指向 GDT 还是 LDT）和访问权限信息。

要访问段中的一个字节，必须提供一个段选择符和偏移量。段选择符提供对段的段描述符（在GDT或LDT中）的访问。从段描述符中，处理器将在线性地址空间中获得段的基地址。偏移量提供了相对于基地址的字节位置。只要该段从处理器正在运行的当前特权级（CPL）可以访问，就可以使用此机制访问任何有效的代码、数据或堆栈段。CPL定义为当前执行代码段的保护级别。

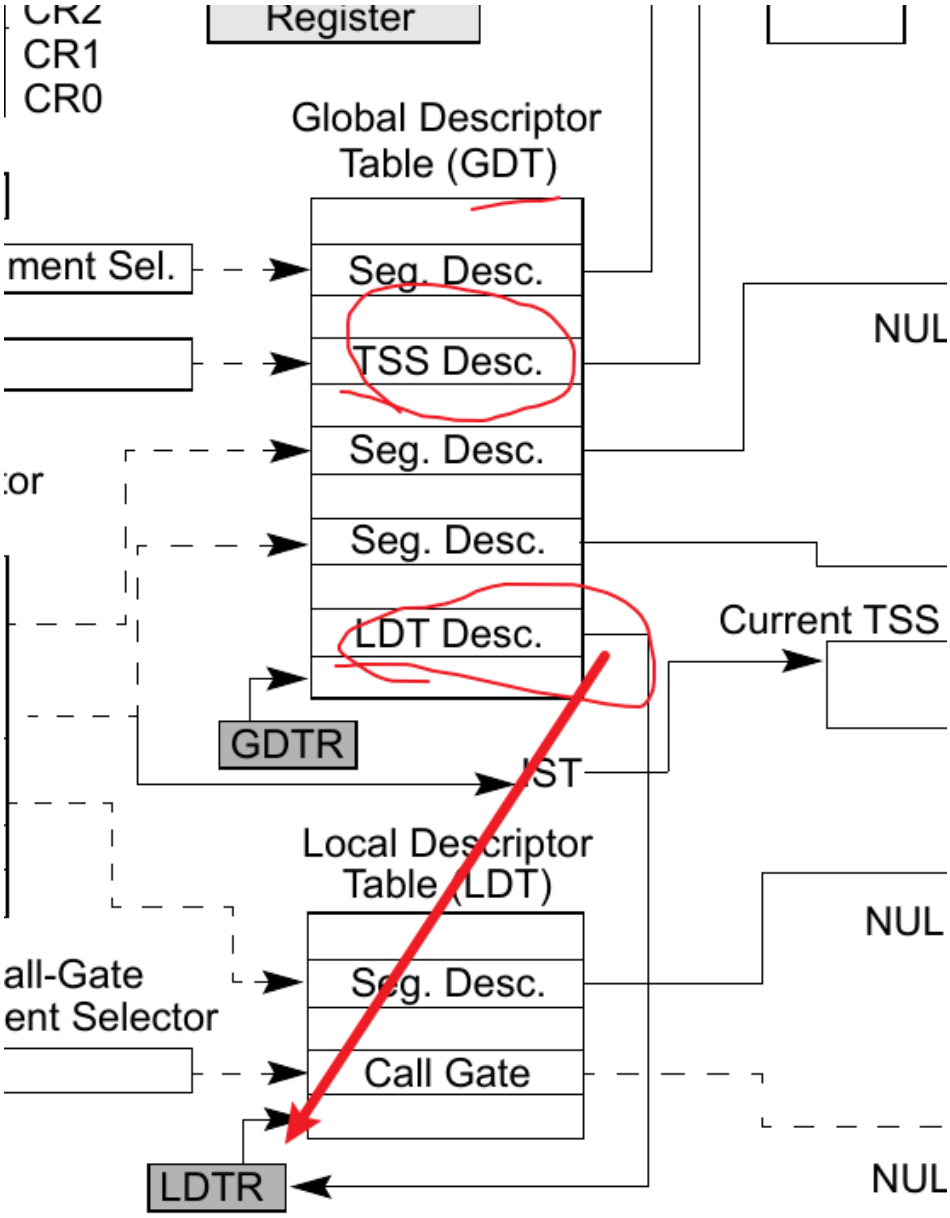
请参见图2-1。图中实线箭头表示线性地址，虚线表示段选择符，点线箭头表示物理地址。为简洁起见，许多段选择符被显示为直接指向一个段。然而，从段选择符到其关联的段的实际路径始终是通过GDT或LDT。

GDT的基地址的线性地址包含在GDT寄存器（GDTR）中；LDT的基地址的线性地址包含在LDT寄存器（LDTR）中。

GDTR 和 LDTR 寄存器在 IA-32e 子模式（64 位模式和兼容模式）中都扩展为 64 位宽。 全局和局部描述符表在 64 位模式下扩展以支持 64 位基址（16 字节 LDT 描述符包含 64 位基址和各种属性）。在兼容模式下，描述符不会扩展。

2.1.2 System Segments, Segment Descriptors, and Gates

除了程序或过程的执行环境所组成的代码、数据和堆栈段之外，架构还定义了两个系统段：任务状态段（TSS）和 LDT。 GDT不被认为是段，因为它不能通过段选择器和段描述符进行访问。 TSS和LDT具有为它们定义的段描述符。



架构还定义了一组称为*门 (gates)*（调用门、中断门、陷阱门和任务门）(call gates, interrupt gates, trap gates, and task gates) 的*特殊描述符*。它们为系统过程和处理程序提供了受保护的入口，这些过程和处理程序可能在不同的特权级别上运行，高于应用程序和大多数过程的特权级别。例如，对调用门的调用可以提供对位于与当前代码段相同或 `a numerically lower privilege level` （更高特权级别）的代码段中的过程的访问。要通过调用门访问过程，调用过程应提供调用门的选择器。然后，处理器对调用门执行访问权限检查，比较当前特权级别（CPL）与调用门和调用门指向的目标代码段的特权级别。

> Gates 与通常用于内存分段的段描述符类似，但具有不同的功能和使用方式。与段描述符一样，Gates 也包含一些控制和管理代码执行的元数据，但它们不用于内存分段，而是用于提供受保护的访问通道到特定的系统过程和处理程序。
>
> - 调用门 (Call gates)：提供对运行在不同特权级别的代码段中过程的访问，*并可以在 64 位模式和兼容模式之间进行转换*。
> - 中断门(Interrupt gates)：用于*中断处理程序*，为其提供受保护的入口，并可防止非授权访问。
> - 陷阱门(Trap gates)：用于*陷阱处理程序*，为其提供受保护的入口，并可防止非授权访问。
> - 任务门(Task gates)：*提供访问TSS(Task-State Segment)的受保护方式*，TSS包含了当前正在运行的任务的状态信息。然而，在*IA-32e模式下，任务门不再被支持*。

(Chatgpt 的回答)

如果允许访问目标代码段，则处理器获取目标代码段的段选择器和来自调用门的该代码段中的偏移量。如果调用需要特权级别的更改，则处理器还会切换到针对目标特权级别的堆栈。新堆栈的*段选择器*从*当前运行任务的TSS中获取*。gates还方便16位和32位代码段之间的转换，反之亦然。

2.1.3 TSS (Task-State Segments and Task Gates)

TSS定义了*一个任务的执行环境状态*。它包括通用寄存器状态、段寄存器状态、EFLAGS 寄存器状态、EIP 寄存器状态、以及带有*三个堆栈段*的段选择器的栈指针状态（每个特权级别一个堆栈）。TSS 还包括与任务相关的 LDT 的段选择器以及分页结构层次的基地址。

在*保护模式*下，所有程序执行都发生在任务的上下文中（称为当前任务）。当前任务的 TSS 的段选择器存储在 `the task register` 中。切换到新任务的最简单方法是调用或跳转到新任务。在这里，新任务的 TSS 的段选择器在 CALL 或 JMP 指令中给出。切换任务时，处理器执行以下操作：

1. 将当前任务的状态存储在当前 TSS 中。
2. 用新任务TSS 的段选择器装载任务寄存器。
3. 通过 GDT 中的段描述符访问新 TSS。
4. 将新任务的状态从新 TSS 装载到通用寄存器、段寄存器、LDTR、控制寄存器 CR3（分页结构层次的基地址）、EFLAGS 寄存器和 EIP 寄存器中。
5. 开始运行新任务。

任务也可以通过任务门访问。任务门与调用门类似，但它提供对 TSS 而不是代码段的访问（通过一个段选择器）。

IA-32e 模式下的任务状态段在 IA-32e 模式下不支持硬件任务切换。但是，TSS 仍然存在。一个 TSS 的基地址由其描述符指定。 64 位 TSS 包含以下对 64 位操作很重要的信息：

- 每个特权级别的堆栈指针地址。
- 中断堆栈表的指针地址。
- IO 权限位图的偏移地址（从 TSS 基地址开始）。在 IA-32e 模式下，任务寄存器扩展为包含 64 位基地址。另请参阅：第 8.7 节，“64 位模式下的任务管理”。

2.1.4 Interrupt and Exception Handling

外部中断、软中断和异常通过中断描述符表（IDT）处理。`IDT` 存储了一系列的门描述符，提供对中断和异常处理程序的访问。与 GDT 一样，IDT 不是一个段。IDT 的基地址的线性地址包含在 `IDTR` 寄存器中。

IDT 中的门描述符可以是*中断门、陷阱门或任务门描述符*。为了访问中断或异常处理程序，处理器首先从内部硬件、外部中断控制器或通过 *INT n*、*INTO*、*INT3*、*INT1* 或 `BOUND` 指令从软件中接收中断向量。`中断向量` 提供了一个索引到 `IDT`。

如果选择的门描述符是一个*中断门*或*陷阱门*，则以类似于通过*调用门*访问过程的方式访问关联的处理程序。

如果描述符是*任务门*，则通过*任务切换*访问处理程序。

In IA-32e mode, interrupt gate descriptors are expanded to 16 bytes to support 64-bit base addresses. This is true for 64-bit mode and compatibility mode.

The IDTR register is expanded to hold a 64-bit base address. Task gates are not supported.

2.1.5 Memory Management

系统架构支持直接物理地址寻址或虚拟内存（通过分页）。

当使用物理寻址时，线性地址被视为物理地址。

当使用分页时：所有代码、数据、堆栈和系统段（包括 GDT 和 IDT）都可以进行分页，只有最近访问的页面保留在物理内存中。

页面（有时称为页面帧）在物理内存中的位置包含在分页结构中。这些结构位于物理内存中（对于 32 位分页的情况，请参见图 2-1）。分页结构层次结构的基本物理地址包含在控制寄存器**CR3**中。分页结构中的条目确定页面帧基地址、访问权限和内存管理信息。要使用此分页机制，线性地址被分为几个部分。这些部分提供了对分页结构和页面帧的单独

偏移量。系统可以有一个分页结构层次结构，也可以有多个分页结构层次结构。例如，每个任务都可以有自己的分页结构层次结构。

Memory Management in IA-32e Mode

在 IA-32e 模式下，物理内存页面由一组系统数据结构管理。在兼容模式和 64 位模式下，使用四或五级系统数据结构（参见第 4 章“分页”）。其中包括以下内容：

- 第 5 级页映射（PML5）- PML5 表中的一个条目包含一个 PML4 的*物理基地址*、访问权限和内存管理信息的物理地址。*PML5 表的基地址*存储在 *CR3* 中。PML5 表仅与 5 级分页一起使用。
- 第 4 级页映射（PML4）- PML4 表中的一个条目包含页面目录指针表的*物理基地址*、访问权限和内存管理信息。在 4 级分页中，只有一个 PML4 表，其基物理地址存储在 *CR3* 中。
- 一组页面目录指针表- 页面目录指针表中的一个条目包含页面目录表的*物理基地址*、访问权限和内存管理信息。
- 一组页面目录- 页面目录表中的一个条目包含页表的*物理基地址*、访问权限和内存管理信息。
- 一组页表- 页表中的一个条目包含*页面帧的物理地址*、访问权限和内存管理信息。

2.1.6 System Registers

为了帮助初始化处理器并控制系统操作，系统架构在 EFLAGS 寄存器和几个系统寄存器中提供了系统标志（system flags）：

- EFLAGS 寄存器中的系统标志和 IOPL 字段控制任务和模式切换、中断处理、指令跟踪和访问权限。参见：第 2.3 节“EFLAGS 寄存器中的系统标志和字段”。
- 控制寄存器（CR0、CR2、CR3 和 CR4）包含各种标志和数据字段，用于控制系统级操作。这些寄存器中的其他标志用于指示操作系统或应用中特定处理器功能的支持。参见：第 2 章“控制寄存器”和第 2.6 节“扩展控制寄存器（包括 XCR0）”。
- 调试寄存器（图 2-1 中未显示）允许设置断点以用于调试程序和系统软件。参见：第 18 章“调试、分支分析、TSC 和英特尔®资源管理器技术（Intel® RDT）特性。”
- GDTR、LDTR 和 IDTR 寄存器包含它们各自表的线性地址和大小（限制）。参见：第 2.4 节“内存管理寄存器”。
- 任务寄存器包含当前任务的 TSS 的线性地址和大小。参见：第 2.4 节“内存管理寄存器”。
- 特定于模型的寄存器（图 2-1 中未显示）。

> Model-specific registers（MSR）是一组寄存器，主要供操作系统或执行程序使用（即在特权级别 0 运行的代码）。这些寄存器控制诸如调试扩展、性能监控计数器、机器检查体系结构和内存类型范围（MTRRs）之类的项目。这些寄存器的数量和功能因 Intel 64 和 IA-32 处理器系列的不同成员而异。另请参阅：《Intel® 64 和 IA-32 架构软件开发人员手册》卷 4 中的第 10.4 节“模型特定寄存器（MSR）”和第 2 章“模型特定寄存器（MSR）”。

大多数系统限制应用程序对系统寄存器（除了EFLAGS寄存器）的访问。但是，系统可以设计为所有程序和过程在最高特权级别（特权级别0）运行。在这种情况下，应用程序将被允许修改系统寄存器。

System Registers in IA-32e Mode

在 IA-32e 模式下，四个系统描述符表寄存器（GDTR，IDTR，LDTR 和 TR）被硬件扩展为 64 位基地址。EFLAGS 变成了 64 位的 RFLAGS 寄存器。CR0-CR4 被扩展到 64 位。CR8 也变得可用。CR8 提供读写访问任务优先级寄存器（TPR），以便操作系统可以控制外部中断的优先级类别。在 64 位模式下，调试寄存器 DR0-DR7 是 64 位的。

在兼容模式下，DR0-DR3 中的地址匹配也以 64 位进行。在支持 IA-32e 模式的系统上，扩展特征启用寄存器（IA32_EFER）可用。

这个特定于模型的寄存器控制 IA-32e 模式和其他 IA-32e 模式操作的激活。此外，还有几个特定于模型的寄存器来管理 IA-32e 模式指令：

- IA32_KERNEL_GS_BASE - 由 SWAPGS 指令使用。
- IA32_LSTAR - 由 SYSCALL 指令使用。
- IA32_FMASK - 由 SYSCALL 指令使用。

• IA32_STAR - 由 SYSCALL 和 SYSRET 指令使用

2.2 MODES OF OPERATION

IA-32 架构支持三种操作模式和一种准操作模式：

• **保护模式** - 这是处理器的本机操作模式。它提供了丰富的体系结构特性、灵活性、高性能和向后兼容性，以支持现有软件基础。

• **实地址模式** - 这种操作模式提供了Intel 8086处理器的编程环境，带有一些扩展（例如能够切换到保护或系统管理模式）。

• **系统管理模式（SMM）** - SMM是所有IA-32处理器的标准架构特性，从Intel386 SL处理器开始。该模式为操作系统或执行体提供了一个透明机制，用于实现电源管理和原始设备制造商差异化功能。SMM通过激活外部系统中断引脚（SMI#）而进入，该引脚生成系统管理中断（SMI）。在SMM中，处理器切换到单独的地址空间，同时保存当前运行程序或任务的上下文。然后可以透明地执行SMM特定代码。从SMM返回时，处理器被放回到SMI之前的状态。

• 虚拟8086模式 - 在保护模式下，处理器支持称为虚拟8086模式的*准操作模式*。此模式允许处理器在受保护的、多任务环境中执行8086软件。

Intel 64架构支持IA-32架构和IA-32e模式的所有操作模式：

• IA-32e 模式 - 在 IA-32e 模式下，处理器支持两个子模式：兼容模式和 64 位模式。64 位模式提供 64 位线性寻址和支持物理地址空间大于 64 GB。兼容模式允许大多数传统受保护模式应用程序不变地运行。

模式之间的切换：
在上电或重置后，处理器处于实地址模式。控制寄存器 CR0 中的 PE 标志控制处理器是在实地址模式还是保护模式下运行。

EFLAGS 寄存器中的 VM 标志确定处理器是在保护模式还是虚拟 8086 模式下运行。在任务切换或从中断或异常处理程序返回时，通常会在保护模式和虚拟 8086 模式之间进行转换。

LMA 位（IA32_EFER. LMA [第 10 位]）确定处理器是在 IA-32e 模式下运行。在 IA-32e 模式下运行时，64 位或兼容子模式操作由代码段的 CS. L 位确定。通过启用分页并设置 LME 位（IA32_EFER. LME [第 8 位]），处理器从保护模式进入 IA-32e 模式。

当处理器处于实地址、保护、虚拟 8086 或 IA-32e 模式时，只要接收到 SMI 就会切换到 SMM。执行 RSM 指令时，处理器总是返回到 SMI 发生时的模式。

2.3 SYSTEM FLAGS AND FIELDS IN THE EFLAGS REGISTER

EFLAGS 寄存器的系统标志和 IOPL 字段控制 I/O、可屏蔽硬件中断、调试、任务切换和虚拟 8086 模式。只有特权代码（通常是操作系统或执行代码）允许修改这些位。

系统标志和 IOPL 包括：

- Trap (bit 8)陷阱（第 8 位）--设置以启用单步调试模式；清除以禁用单步调试模式。在单步调试模式下，处理器在每个指令后生成一个调试异常。这允许在每个指令后检查程序的执行状态。如果应用程序使用 POPF、POPFD 或 IRET 指令设置 TF 标志，则在 POPF、POPFD 或 IRET 之后的指令后会生成调试异常。
- Interrupt enable (bit 9)中断使能（第 9 位）--控制处理器对可屏蔽硬件中断请求的响应（另请参见：第 6.3.2 节“可屏蔽硬件中断”）。该标志被设置以响应可屏蔽硬件中断；清除以禁止可屏蔽硬件中断。IF 标志不影响异常或非可屏蔽中断（NMI 中断）的生成。CPL、IOPL 和控制寄存器 CR4 中 VME 标志的状态确定 IF 标志是否可以由 CLI、STI、POPF、POPFD 和 IRET 修改。
- I/O privilege level field (bits 12 and 13) I/O 特权级字段（第 12 位和第 13 位）--指示当前运行的程序或任务的 I/O 特权级（IOPL）。
- Nested task (bit 14) 嵌套任务（第 14 位）--控制中断和调用任务的链接。处理器在使用 CALL 指令、中断或异常启动任务时设置此标志。在使用 IRET 指令返回任务时，它会检查和修改此标志。
- Resume 恢复（第 16 位）--控制处理器对指令断点条件的响应。

- Alignment check or access control (bit 18) 对齐检查或访问控制（位 18）-如果 CR0 寄存器中的 AM 位被设置，则仅当此标志为 1 时，启用用户模式数据访问的对齐检查。当引用未对齐的操作数时（例如奇数字节地址处的字或地址不是四的整数倍的双字），将生成对齐检查异常。对齐检查异常仅在用户模式（特权级 3）下生成。默认为特权级 0 的内存引用，例如段描述符加载，即使是由在用户模式下执行的指令引起的，也不会生成此异常。对齐检查异常可用于检查数据的对齐方式。这在与需要所有数据对齐的处理器交换数据时非常有用。

> 在 IA-32e 模式下，处理器不允许设置 VM 位，因为不支持虚拟 8086 模式（尝试设置该位会被忽略）。此外，处理器不会设置 NT 位。但是，处理器允许软件设置 NT 位（请注意，在 IA-32e 模式下，如果设置了 NT 位，则 IRET 会导致通用保护故障）。在 IA-32e 模式下，SYSCALL/SYSRET 指令具有可编程方法来指定在 RFLAGS/EFLAGS 中清除哪些位。这些指令保存/恢复 EFLAGS/RFLAGS。

2.4 MEMORY-MANAGEMENT REGISTERS

处理器提供四个内存管理寄存器（GDTR、LDTR、IDTR 和 TR），它们指定控制分段内存管理的数据结构的位置。提供了特殊指令来加载和存储这些寄存器。

GDTR

GDTR 寄存器包含 GDT 的基地址（在保护模式下为 32 位，在 IA-32e 模式下为 64 位）和 16 位表限制。基地址指定 GDT 的字节 0 的线性地址；表限制指定表中的字节数。

LGDT 和 SGDT 指令分别加载和存储 GDTR 寄存器。在处理器上电或重置时，基地址被设置为默认值 0，限制被设置为 0FFFFH。在保护模式操作的处理器初始化过程中，必须加载新的基地址到 GDTR 中。

LDTR

LDTR 寄存器包含 LDT 的 16 位段选择器、基地址（在保护模式下为 32 位，在 IA-32e 模式下为 64 位）、段限制和描述符属性。基地址指定 LDT 段的字节 0 的线性地址；段限制指定段中的字节数。

LLDT和SLDT指令分别加载和存储LDTR寄存器的段选择器部分。包含LDT的段必须在GDT中有一个段描述符。当`LLDT`指令在LDTR中加载段选择器时：从LDT描述符中自动加载基地址、限制和描述符属性到LDTR中。

当任务切换发生时，LDTR会自动加载新任务的LDT的段选择器和描述符。在写入新的LDT信息到寄存器之前，LDTR的内容不会自动保存。

在处理器上电或重置时，段选择器和基地址被设置为默认值 0，限制被设置为 0FFFFH。

IRTR (Interrupt Descriptor Table Register)

IDTR 寄存器保存 IDT 的基地址（在保护模式下为 32 位，在 ia-32e 模式下为 64 位）和 16 位表限制。基地址指定 idt 字节 0 的线性地址；表限制指定表中的字节数。`LIDT` 和 `SIDT` 指令分别加载和存储 IDTR寄存器。在处理器上电或复位时，基地址被设置为默认值 0，限制被设置为 0xffffh。然后可以作为处理器初始化过程的一部分来更改寄存器中的基地址和限制。

Task Register (TR)

TR 任务寄存器保存当前任务的 TSS 的 16 位段选择器、基地址（在保护模式下为 32 位，在 IA-32e 模式下为 64 位）、段限制和描述符属性。*16 位选择器引用 GDT 中的 TSS 描述符*。基地址指定 TSS 的字节 0 的线性地址；段限制指定 TSS 中的字节数。

`LTR`和`STR`指令分别加载和存储任务寄存器的段选择器部分。当`LTR`指令在任务寄存器中加载段选择器时，TSS描述符中的基地址、限制和描述符属性会自动加载到任务寄存器中。在处理器上电或重置时，基地址被设置为默认值0，限制被设置为0FFFFH。

当任务切换发生时，任务寄存器会自动加载新任务的 TSS 的段选择器和描述符。

2.5 CONTROL REGISTERS

控制寄存器（CR0、CR1、CR2、CR3 和 CR4）确定处理器的操作模式和当前执行任务的特性。在所有 32 位模式和兼容模式下，这些寄存器都是 32 位的。

在64位模式下，控制寄存器扩展到64位。MOV CRn指令用于操作寄存器位。这些指令的操作数大小前缀被忽略。以下也是正确的：

• 控制寄存器可以使用移动到或从控制寄存器的形式的MOV指令读取和加载（或修改）。在保护模式下，MOV指令允许读取或加载控制寄存器（仅在特权级0）。这个限制意味着应用程序或操作系统过程（在特权级1、2或3运行）被阻止读取或加载控制寄存器。

• CR0和CR4中的一些位是保留位，必须写入零。尝试设置CR0[31:0]中的任何保留位会被忽略。尝试设置CR0[63:32]中的任何保留位会导致通用保护异常#GP(0)。尝试设置CR4中的任何保留位都会导致通用保护异常#GP(0)。

• 所有64位的CR2都可以由软件编写。

• CR3[63:MAXPHYADDR]中的保留位必须为零。尝试设置它们中的任何一个会导致#GP(0)。

• MOV CR2指令不检查写入CR2的地址是否规范。

• 64位处理器在退出IA-32e模式时将保留每个控制寄存器的上32位。

• 在64位模式下，对CR的MOV执行会将控制寄存器的上32位清零。

• 控制寄存器CR8仅在64位模式下可用。

下面总结了控制寄存器，这些控制寄存器中的每个体系结构定义的控制字段都有单独的描述。

• CR0 – 包含系统控制标志，控制处理器的操作模式和状态。

• CR1 – 保留。

• CR2 – 包含页故障线性地址（引起页故障的线性地址）。

• CR3 – 包含整个多级页表结构的*基地址*和两个标志（PCD 和 PWT）。只指定基地址的最高有效位（不包括低 12 位）；地址的低 12 位假定为 0。因此，第一个分页结构必须对齐到页面（4-KByte）边界。PCD 和 PWT 标志控制处理器内部数据缓存中该分页结构的缓存（它们不控制 TLB 缓存页目录信息）。

使用物理地址扩展时，CR3寄存器包含页目录指针表的基地址。使用4级分页和5级分页时，*CR3寄存器分别包含PML4表和PML5表的基地址*。如果启用了PCID，则CR3具有与图2-7中所示不同的格式。请参阅第4.5节，“4级分页和5级分页”。

• CR4 – 包含一组标志，启用几个体系结构扩展，并指示操作系统或执行支持特定处理器功能的支持。CR4[63:32]位仅用于进入64位模式后启用的IA-32e模式特性。CR4[63:32]位在IA-32e模式之外没有任何影响。

• CR8 – 仅在64位模式下提供对任务优先级寄存器（TPR）的读写访问。它指定操作系统用于控制允许中断处理器的外部中断的优先级类别的优先级阈值值。但是，在兼容模式下仍然应用中断过滤。

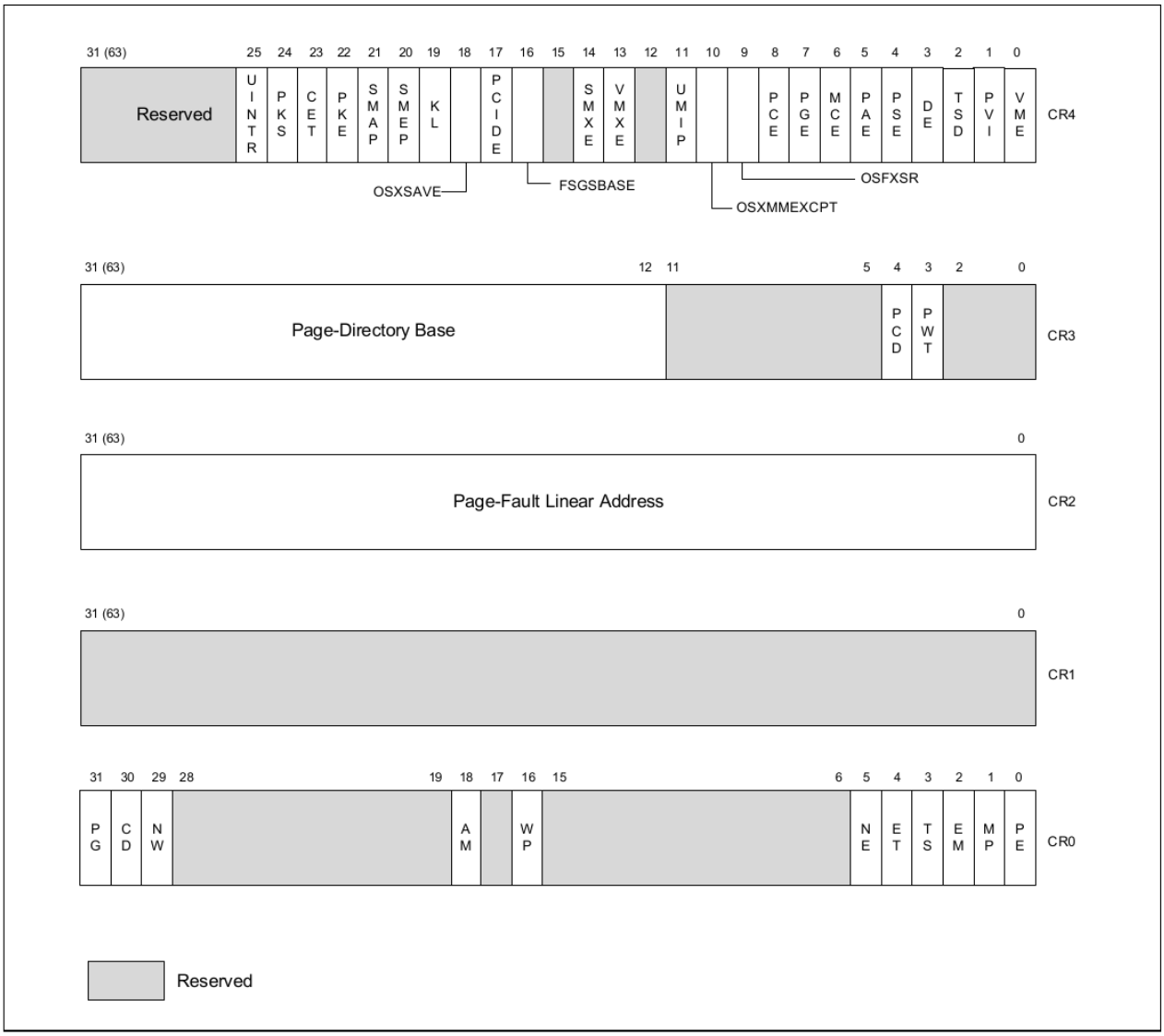


Figure 2-7. Control Registers

加载和存储系统寄存器的系统级指令：

加载和存储系统寄存器

GDTR、LDTR、IDTR和TR寄存器每个都有一个加载和存储指令，用于将数据加载到寄存器中并从寄存器中存储数据：

- LGDT（加载 GDTR 寄存器）-从内存中加载 GDT 基址和限制到 GDTR 寄存器中。
- SGDT（存储 GDTR 寄存器）-将 GDTR 寄存器中的 GDT 基址和限制存储到内存中。
- LIDT（加载 IDTR 寄存器）-从内存中加载 IDT 基址和限制到 IDTR 寄存器中。
- SIDT（存储 IDTR 寄存器）-将 IDTR 寄存器中的 IDT 基址和限制存储到内存中。
- LLDT（加载 LDTR 寄存器）-从内存中加载 LDT 段选择器和段描述符到 LDTR 中。（段选择器操作数也可以位于通用寄存器中。）
- SLDT（存储 LDTR 寄存器）-将 LDTR 寄存器中的 LDT 段选择器存储到内存或通用寄存器中。
- LTR（加载任务寄存器）-从内存中加载 TSS 的段选择器和段描述符到任务寄存器中。（段选择器操作数也可以位于通用寄存器中。）
- STR（存储任务寄存器）-将当前任务TSS的段选择器从任务寄存器中存储到内存或通用寄存器中。