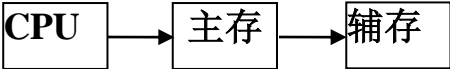
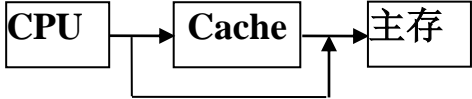


# 第 5 章 Pentium 微处理器保护模式存储管理

## 5.1 虚拟存储器及其工作原理

- **虚拟存储器：**由主存储器、辅助存储器、辅助硬件和操作系统管理软件组成的一种存储体系。
- **虚拟存储系统目标：**增加存储器的存储容量。它的速度接近于主存，单位造价接近于辅存，因此性能价格比很高。
- 虚拟存储器概念是1961年由英国曼彻斯特大学的Kilburn等人提出的，并于20世纪70年代广泛应用于大中型计算机之中，现在的微型计算机也都采用了这种技术。

# 表 5.1.1 虚拟存储器和 Cache 存储器的比较

存储体系	虚拟存储器	Cache存储器
存储层次	主存-辅存	Cache-主存
主要功能	主存速度，辅存容量	CPU速度，主存容量
信息传送单位	信息块（比如段、页），有多种划分，长度较大。	信息块（比如块、行），长度较小，并且固定。
结构差别	 <pre> graph LR     CPU[CPU] --&gt; MM[主存]     MM --&gt; AM[辅存]             </pre> <p>主存不命中，进行辅存调度，而CPU程序换道。</p>	 <pre> graph LR     CPU[CPU] --&gt; Cache[Cache]     Cache --&gt; MM[主存]     CPU --&gt; MM             </pre> <p>在CPU与主存之间具有直接访问通路。</p>
操作过程	由部分硬件和操作系统存储管理软件实现，对应用程序员透明，对存储管理软件程序员不透明。	全部用硬件实现，对各类程序员透明。

# 5.1.1 地址空间及地址

- 在虚拟存储器中有3种地址空间及对应的3种地址。
- **虚拟地址空间**：又称为虚存地址空间，是应用程序员用来编写程序的地址空间，与此相对应的地址称为虚地址或逻辑地址。
- **主存地址空间**：又称为实存地址空间，是存储、运行程序的空间，其相应的地址称为主存物理地址或实地址。
- **辅存地址空间**：也就是磁盘存储器的地址空间，是用来存放程序的空间，相应的地址称为辅存地址或磁盘地址。

## 5.1.2 虚拟存储器工作原理

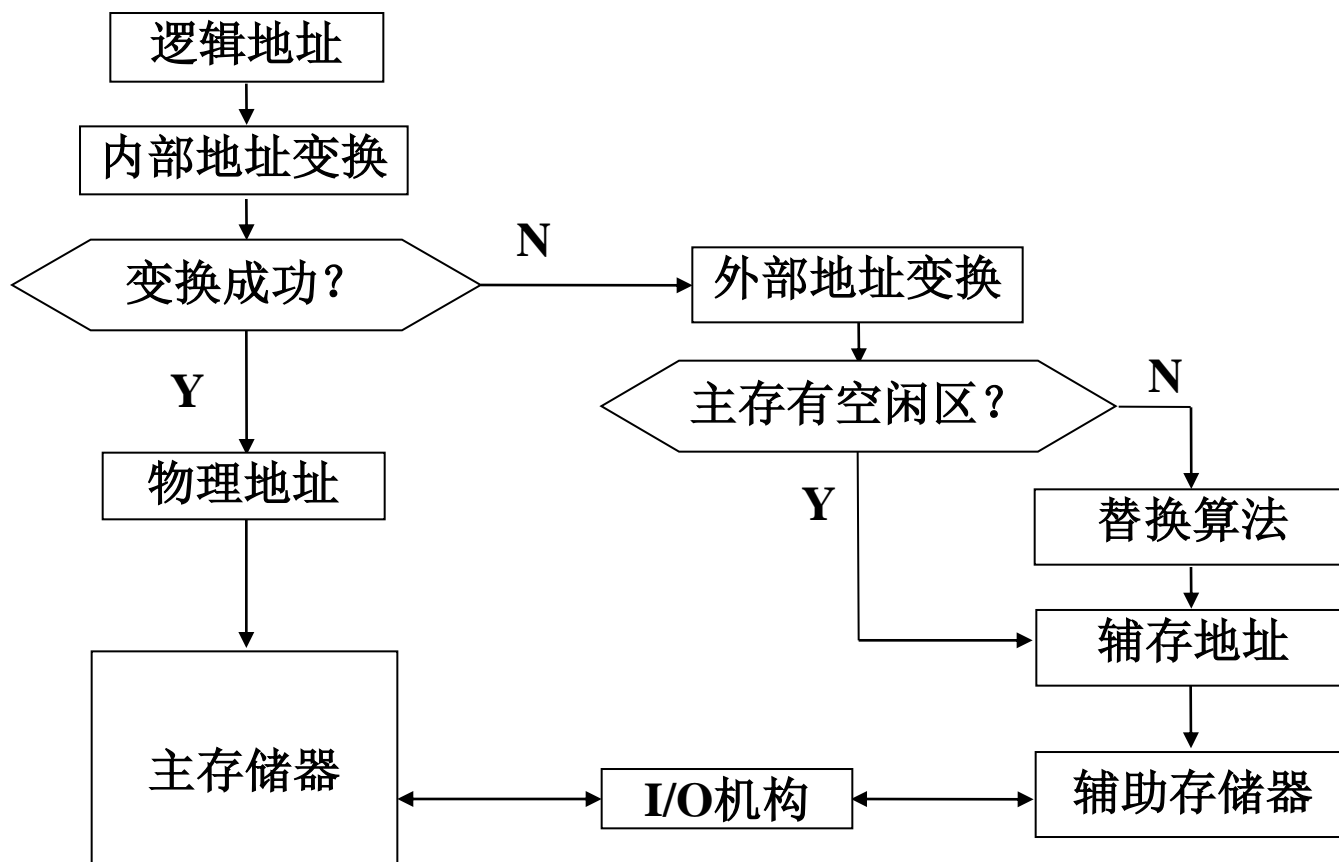


图5.1.1 虚拟存储器工作过程示意图

# 存贮管理方式

- 由于采用的存贮映象算法不同，就形成了不同的存贮管理方式，其中主要有3种：段式、页式、段页式
- Pentium支持分段存储管理、分页存储管理和段页式存储管理。
- Pentium微处理机的存储管理部件：由分段部件和分页部件组成。
- 分段部件功能：将逻辑地址转换成一个连续的不分段的地址空间，这个地址空间的地址叫做线性地址。
- 分页部件功能：将线性地址转换成物理地址。

## 5.2 分段存储管理

- 5.2.1 分段存储管理的基本思想

- 一个程序由多个模块组成。
- 每一个模块都是一个特定功能的独立的程序段。
- **段式管理**：把主存按段分配的存储管理方式。
- 程序模块→段→段描述符→段描述符表→段描述符表寄存器

# 1. 分段存储管理工作过程

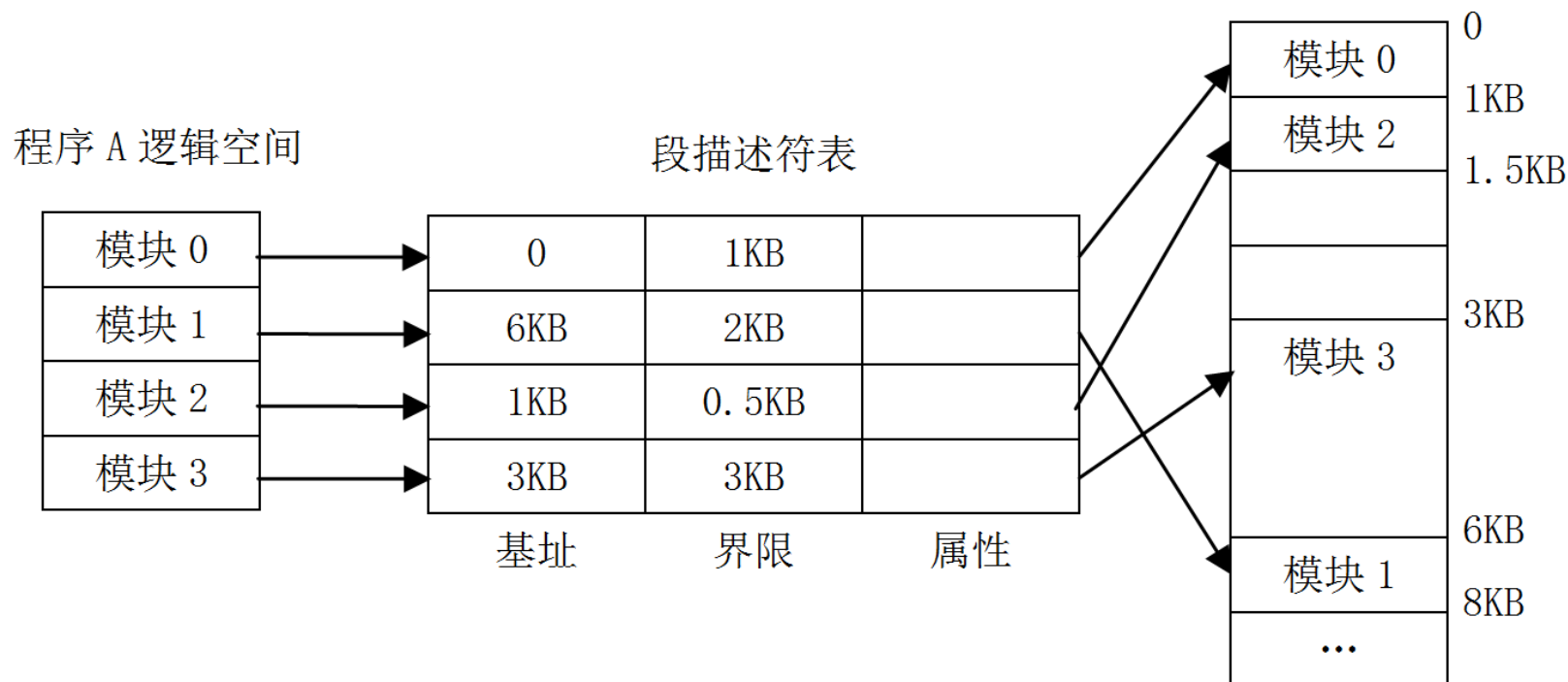


图5.2.1 分段存储管理的示意图

## 2. 虚拟地址和虚拟地址空间

- Pentium 微处理机在保护模式下的存储器管理单元使用48位的存储器指针。

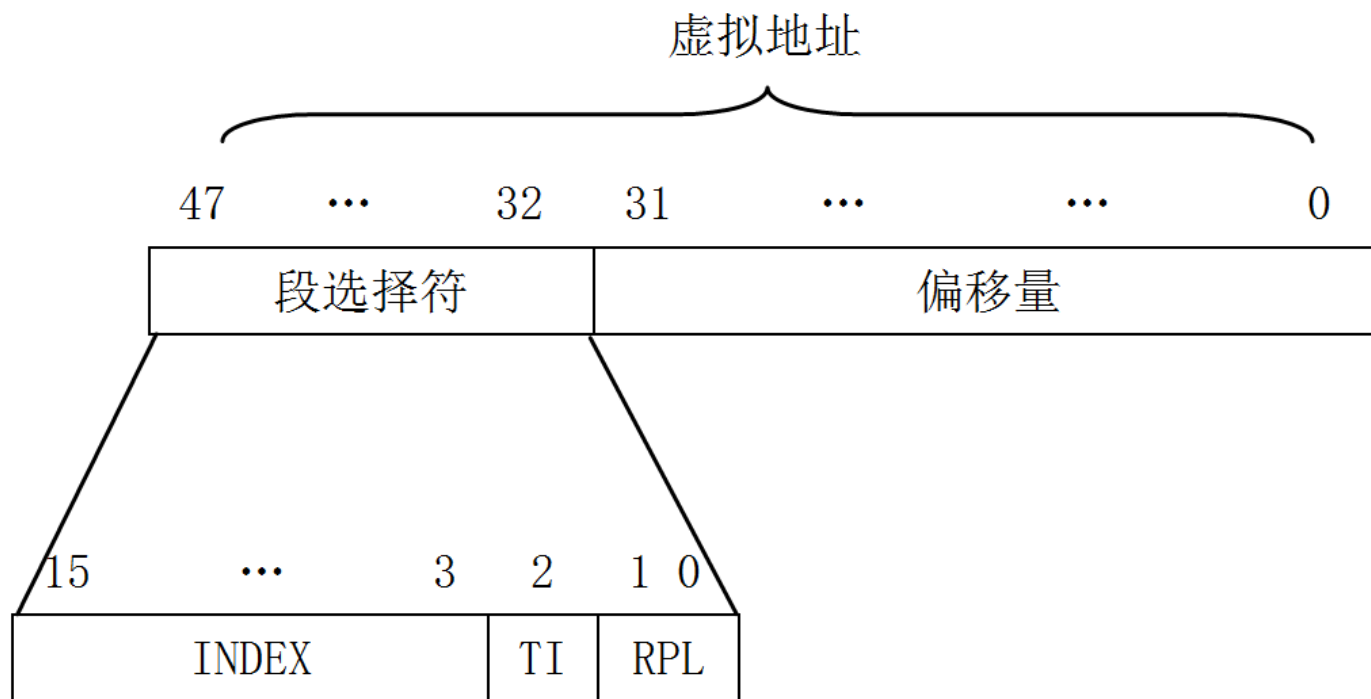


图 5.2.2 保护模式下的存储器指针及段选择符格式



### 3. 虚实地址转换

- 段选择符 → 段描述符表 → 段描述符 → 段基址 → 偏移量 → 物理地址。

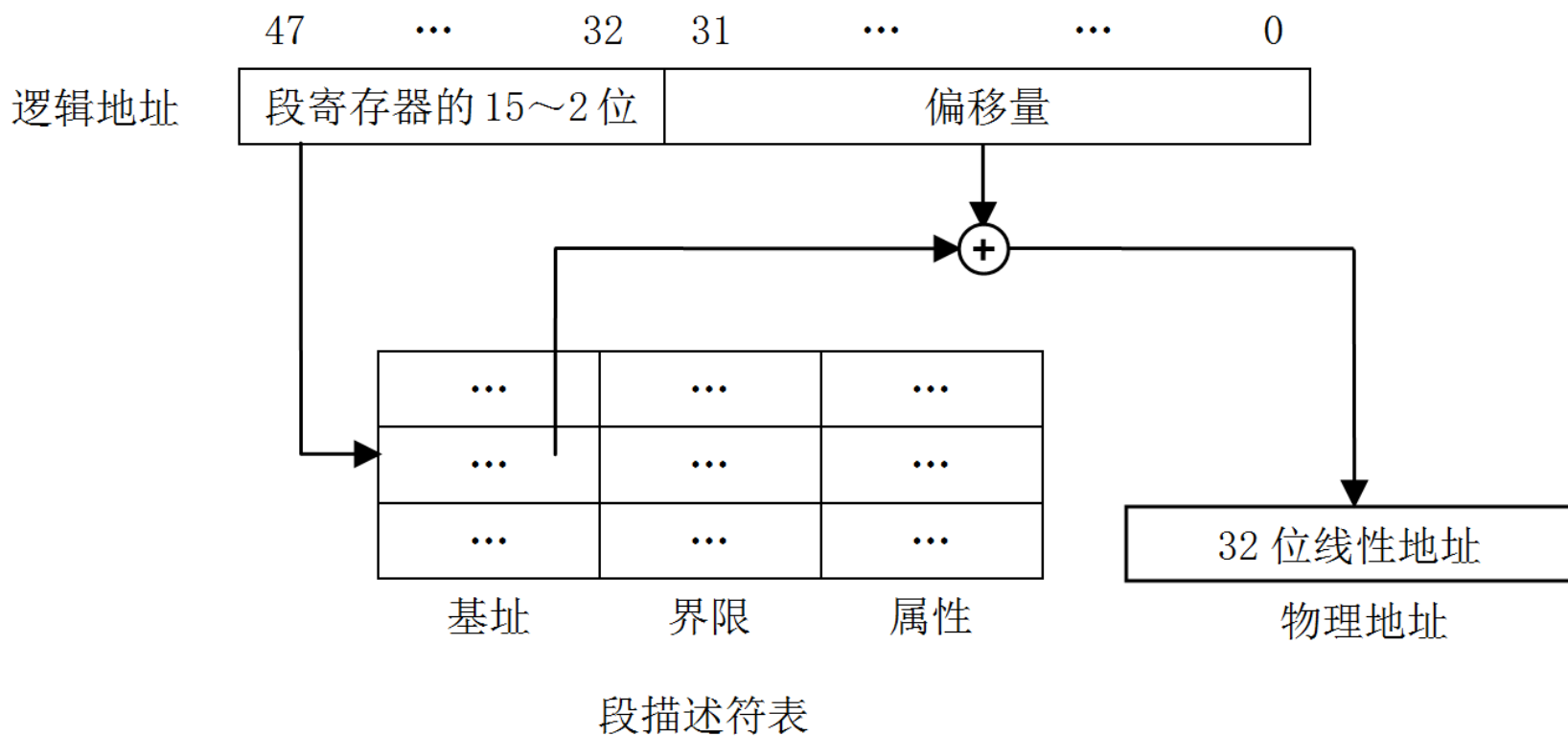


图 5.2.3 虚实地址转换示意图

## 5.2.2 段描述符

- **段描述符**：用于描述段的基本信息。
- 由8个字节组成。
- 段描述符保存段的属性、段的大小、段在存储器中的位置以及控制和状态信息。
- 一般说来，各段描述符都是由各种编译程序、各种连接程序、各种装入程序或者是操作系统产生的，而不是由各种应用程序生成的。

# 段描述符的分类

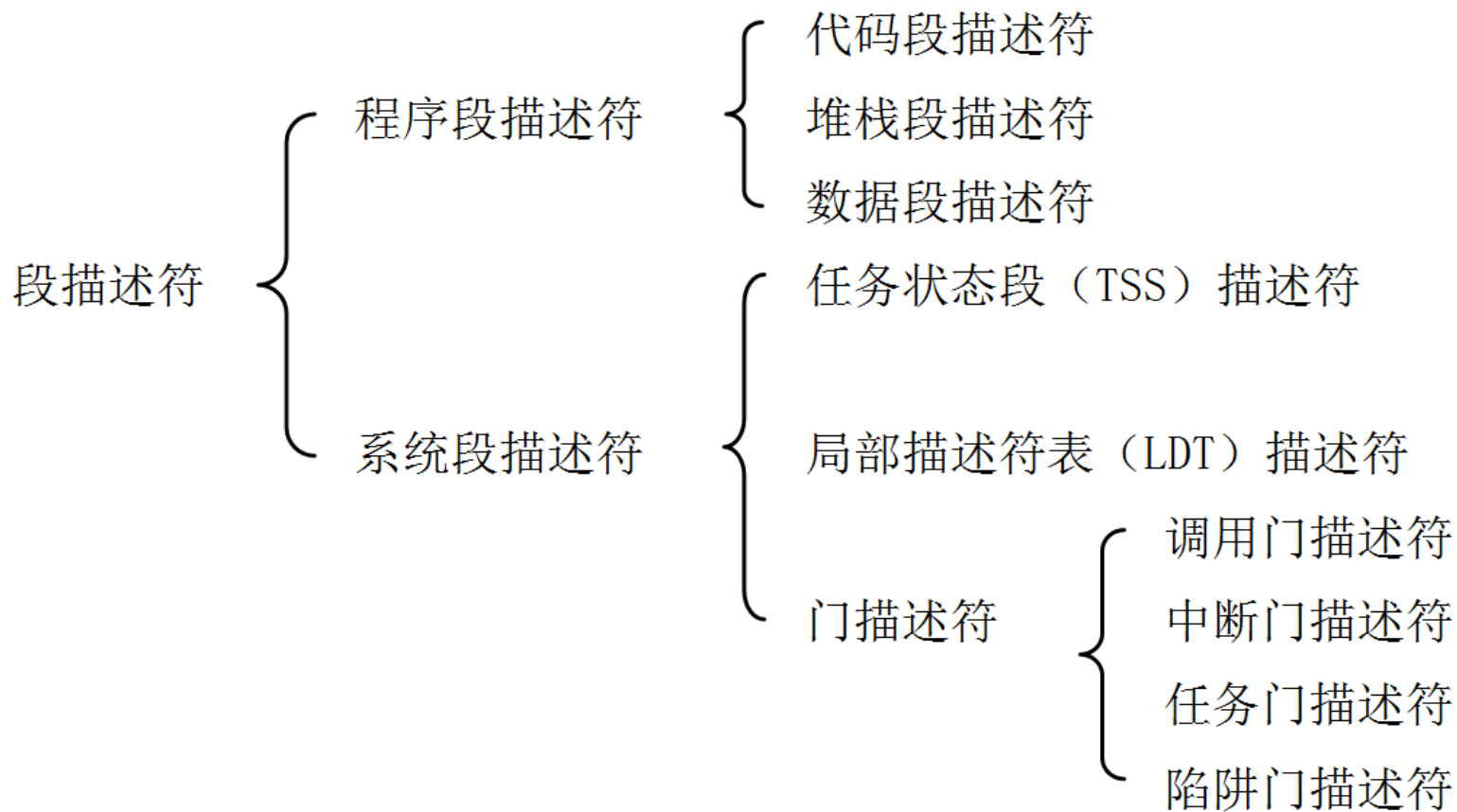


图5.2.4 段描述符的分类

# 1. 程序段描述符

表 5.2.1 程序段描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
段界限 7~0								0
段界限 15~8								1
段基址 7~0								2
段基址 15~8								3
段基址 23~16								4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
G 粒度	D/B	0	AVL	段界限 19~16				6
段基址 31~24								7

**基地址：** 32位，规定一个段在4GB物理地址空间中的起始位置。

# 1. 程序段描述符

表 5.2.1 程序段描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
段界限 7~0								0
段界限 15~8								1
段基址 7~0								2
段基址 15~8								3
段基址 23~16								4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
G 粒度	D/B	0	AVL	段界限 19~16				6
段基址 31~24								7

(1) **段界限**：20位，决定了段的长度，该字段的值的单位由“G”位决定。

(2) “G”位称作粒度位，用来确定段界限所使用的长度单位

# 1. 程序段描述符

表 5.2.1 程序段描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
段界限 7~0								0
段界限 15~8								1
段基址 7~0								2
段基址 15~8								3
段基址 23~16								4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
G 粒度	D/B	0	AVL	段界限 19~16				6
段基址 31~24								7

- (1) **粒度位G**：确定段界限所使用的长度单位。
- (2) G=0时，段的长度以一个字节为单位。
- (3) G=1时，段的长度以4K字节为单位。

# 1. 程序段描述符

表 5.2.1 程序段描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
段界限 7~0								0
段界限 15~8								1
段基址 7~0								2
段基址 15~8								3
段基址 23~16								4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
G 粒度	D/B	0	AVL	段界限 19~16				6
段基址 31~24								7

- (1) **分类S**: 区分是系统段描述符还是非系统段描述符。
- (2) 当S=0时, 是系统段描述符。
- (3) 当S=1时, 是非系统段描述符。

# 1. 程序段描述符

表 5.2.1 程序段描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
段界限 7~0								0
段界限 15~8								1
段基址 7~0								2
段基址 15~8								3
段基址 23~16								4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
G 粒度	D/B	0	AVL	段界限 19~16				6
段基址 31~24								7

- (1) **段存在位P**：该段是否在内存中。
- (2) 当P=1时，表示该段在内存中。
- (3) 当P=0时，表示该段不在内存中。



# 1. 程序段描述符

表 5.2.1 程序段描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
段界限 7~0								0
段界限 15~8								1
段基址 7~0								2
段基址 15~8								3
段基址 23~16								4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
G 粒度	D/B	0	AVL	段界限 19~16				6
段基址 31~24								7

(1) **系统可用位AVL**：表示系统软件是否可用本段。

(2) 当AVL=1时，表示系统软件可用本段。

(3) 当AVL=0时，表示系统软件不能用本段。

# 1. 程序段描述符

表 5.2.1 程序段描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
段界限 7~0								0
段界限 15~8								1
段基址 7~0								2
段基址 15~8								3
段基址 23~16								4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
G 粒度	D/B	0	AVL	段界限 19~16				6
段基址 31~24								7

(1) **特权级DPL**: 定义段的特权级。

(2) 2位, 有4个特权级: 00、01、10、11, 称作0级、1级、2级、3级。0级的特权最高, 1级次之, 3级的特权最低。

(3) 用这个字段定义的特权级去控制对这个段的访问。

# 1. 程序段描述符

表 5.2.1 程序段描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
段界限 7~0								0
段界限 15~8								1
段基址 7~0								2
段基址 15~8								3
段基址 23~16								4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
G 粒度	D/B	0	AVL	段界限 19~16				6
段基址 31~24								7

- (1) **D位/B位**: 32/16大小选择。 D/B=1, 选32位; D/B=0, 选16位。
- (2) 在代码段描述符中, 指示操作数长度和有效地址长度, D位。
- (3) 在堆栈段描述符中, 指示ESP或SP, B位。
- (4) 在数据段描述符中, 指示操作数长度。 B位

# 1. 程序段描述符

表 5.2.1 程序段描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
段界限 7~0								0
段界限 15~8								1
段基址 7~0								2
段基址 15~8								3
段基址 23~16								4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
G 粒度	D/B	0	AVL	段界限 19~16				6
段基址 31~24								7

**兼容位：**第6字节的D5位必须是0，以便与将来的处理器兼容。

# 1. 程序段描述符

表 5.2.1 程序段描述符的格式


D7	D6	D5	D4	D3	D2	D1	D0	字节
段界限 7~0								0
段界限 15~8								1
段基址 7~0								2
段基址 15~8								3
段基址 23~16								4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
G 粒度	D/B	0	AVL	段界限 19~16				6
段基址 31~24								7

**类型TYPE:** 在不同的段描述符中有不同的格式。

# 数据段或堆栈段描述符中的类型TYPE字段

表 5.2.2 数据段或堆栈段描述符中的 TYPE 字段的格式

D7	D6	D5	D4	D3	D2	D1	D0
P	DPL		S=1	E=0	ED	W	A



**E可执行位：**当E=0时，是数据段或堆栈段。

**ED扩展方向位：**当ED=0时，向上扩展（地址增加方向），通常用于数据段。  
当ED=1时，向下扩展（地址减小方向），通常用于堆栈段。

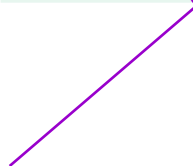
**W可写位：**当W=0时，不允许写入。当W=1时，允许写入。

**A访问位：**当A=0时，该段尚未被访问。当A=1时，该段已被访问

# 代码段描述符中的类型TYPE字段

表5.2.3 代码段描述符中的类型TYPE字段的格式

D7	D6	D5	D4	D3	D2	D1	D0
P	DPL		S=1	E=1	C	R	A



**E可执行位：**当E=1时，是代码段。

**C一致性位：**一致性检查就是采用特权级进行控制。C=0，表示非一致性代码段，此时忽视段描述符的特权值。C=1，表示一致性代码段，需要进行特权级检查。

**R可读位：**当R=0时，不允许读。当R=1时，允许读。

**A访问位：**当A=0时，该段尚未被访问；当A=1时，该段已被访问

## 2. 系统段描述符

表 5.2.4 系统段描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
段界限 7~0								0
段界限 15~8								1
段基址 7~0								2
段基址 15~8								3
段基址 23~16								4
P 存在	DPL 特权级		S 分类		TYPE 类型			5
G 粒度	0	0	0	段界限 19~16				6
段基址 31~24								7

分类S=0，表示为系统段描述符。此时TYPE类型有另的意义。



# 系统段描述符中的类型TYPE字段的格式

表 5.2.5 系统段描述符中的 TYPE 字段的格式

TYPE	段的类型（用途）	TYPE	段的类型（用途）
0000	未定义（无效）	1000	未定义（无效）
0001	286 TSS 描述符，非忙	1001	TSS 描述符，非忙
0010	LDT 描述符	1010	未定义（保留）
0011	286 TSS 描述符，忙	1011	TSS 描述符，忙
0100	286 调用门描述符	1100	<u>调用门描述符</u>
0101	<u>任务门描述符</u>	1101	未定义（保留）
0110	286 中断门描述符	1110	<u>中断门描述符</u>
0111	286 陷阱门描述符	1111	<u>陷阱门描述符</u>

### 3. 门描述符

- **门**：一种关卡，用来控制从一段程序到另一段程序或从一个任务到另一个任务的转移。
- **门描述符**：用于控制转入目标代码段的入口点。
- **门描述符包括**：调用门、任务门、中断门和陷阱门。
- 调用门用于改变特权级别。
- 任务门用于任务切换。
- 中断门和陷阱门用于确定中断服务程序。

# 门描述符的格式

表 5.2.6 门描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
偏移地址 7~0								0
偏移地址 15~8								1
			段选择符 7~0					2
			段选择符 15~8					3
0	0	0	字计数					4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
偏移地址 23~16								6
偏移地址 31~24								7

- (1) **段选择符**: 16位, 指出段描述符位置, 索引值。
- (2) 任务门送TR, 其他门则送CS。

# 门描述符的格式

表 5.2.6 门描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
			偏移地址 7~0					0
			偏移地址 15~8					1
			段选择符 7~0					2
			段选择符 15~8					3
0	0	0	字计数					4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
			偏移地址 23~16					6
			偏移地址 31~24					7

**偏移地址：**32位，指出目标程序的入口偏移量。

# 门描述符的格式

表 5.2.6 门描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
偏移地址 7~0								0
偏移地址 15~8								1
段选择符 7~0								2
段选择符 15~8								3
0	0	0	字计数					4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
偏移地址 23~16								6
偏移地址 31~24								7

(1) **分类S**: S=0是系统段描述符。S=1是非系统段描述符。

(2) 此处应该S=0，但到底是系统段，还是门，要看类型TYPE。

# 门描述符的格式

表 5.2.6 门描述符的格式

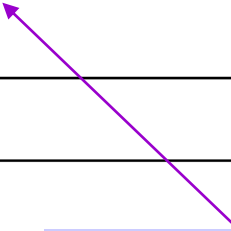
D7	D6	D5	D4	D3	D2	D1	D0	字节
偏移地址 7~0								0
偏移地址 15~8								1
段选择符 7~0								2
段选择符 15~8								3
0	0	0	字计数					4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
偏移地址 23~16								6
偏移地址 31~24								7

- (1) **类型TYPE**: 确定门的分类: 调用门、任务门、中断门和陷阱门。
- (2) 类型**TYPE**字段的规则与系统段描述符中的类型**TYPE**字段的格式完全相同。

# 门描述符的格式

表 5.2.6 门描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
偏移地址 7~0								0
偏移地址 15~8								1
段选择符 7~0								2
段选择符 15~8								3
0	0	0	字计数					4
P 存在	DPL 特权级		S 分类	TYPE 类型				5
偏移地址 23~16								6
偏移地址 31~24								7

- 
- (1) **P字段**: 表示描述符内容是否有效。
  - (2) 当P=0时, 表示描述符内容无效。
  - (3) 当P=1时, 表示描述符内容有效。

# 门描述符的格式

表 5.2.6 门描述符的格式

D7	D6	D5	D4	D3	D2	D1	D0	字节
偏移地址 7~0								0
偏移地址 15~8								1
段选择符 7~0								2
段选择符 15~8								3
0	0	0	字计数				4	
P 存在	DPL 特权级		S 分类	TYPE 类型				5
偏移地址 23~16								6
偏移地址 31~24								7

(1) **字计数**: 指示已有多少字参数要从调用者的堆栈复制到被调用的子程序堆栈 (字计数值)。

(2) 字计数只用于特权级有变化的调用门, 别的门都不用字计数。



## 5.2.3 全局描述符表及寄存器

- **全局描述符表GDT**：保存系统使用、各任务共享的段描述符，只有一个。
- **全局描述符表寄存器GDTR**：指定了GDT的起始地址

48位 = 32位基地址 + 16位界限

- **GDT保存的描述符类型**：除中断门、陷阱门外的各类描述符。
- **GDT寻址可归纳为**：
  1. 段选择符 \* 8 + GDTR基址 = 段描述符地址
  2. 段描述符内容（包括基址）→ 相应段cache
  3. cache中段基址 + 虚地址偏移量 = 物理地址

# 由GDTR确定GDT存储位置和界限

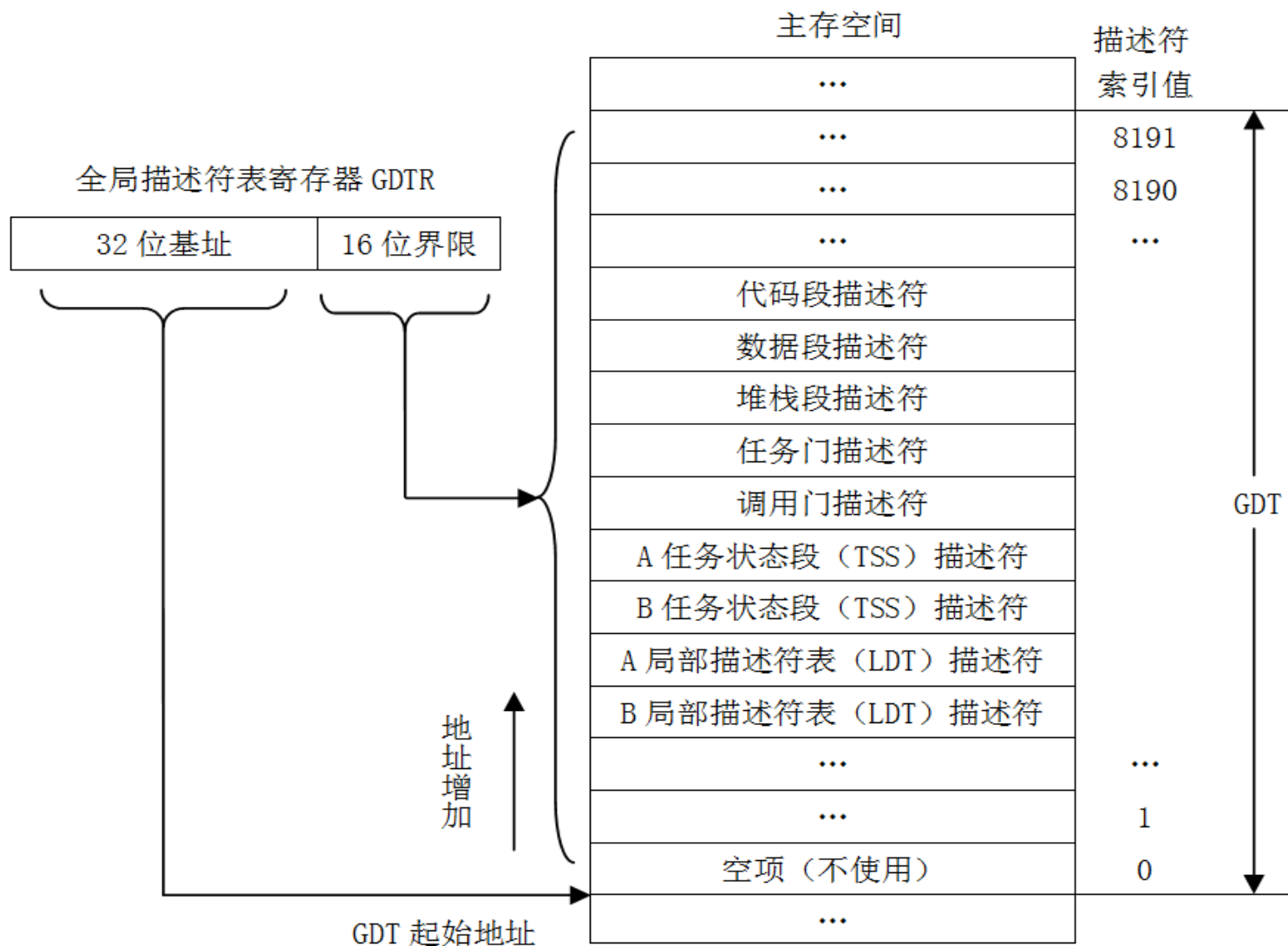


图5.2.5 由GDTR确定GDT存储位置和界限

## 5.2.4 局部描述符表及寄存器

- **局部描述符表LDT**: 保存某任务使用的段描述符, 每个任务一个。
- **LDTR**: 指出LDT的基址

16位选择符 + (32位基址 + 20位界限 + 12位属性)

- LDTR的段选择符确定LDT描述符在GDT中的位置。
- 如果LDTR中装入了段选择符, 处理器自动地将相应的LDT描述符从全局描述符表GDT中读出来, 并装入LDTR中的cache部分, 其中包括LDT基址, 从而为当前任务创建一个LDT。

# 由LDTR确定LDT存储位置和界限

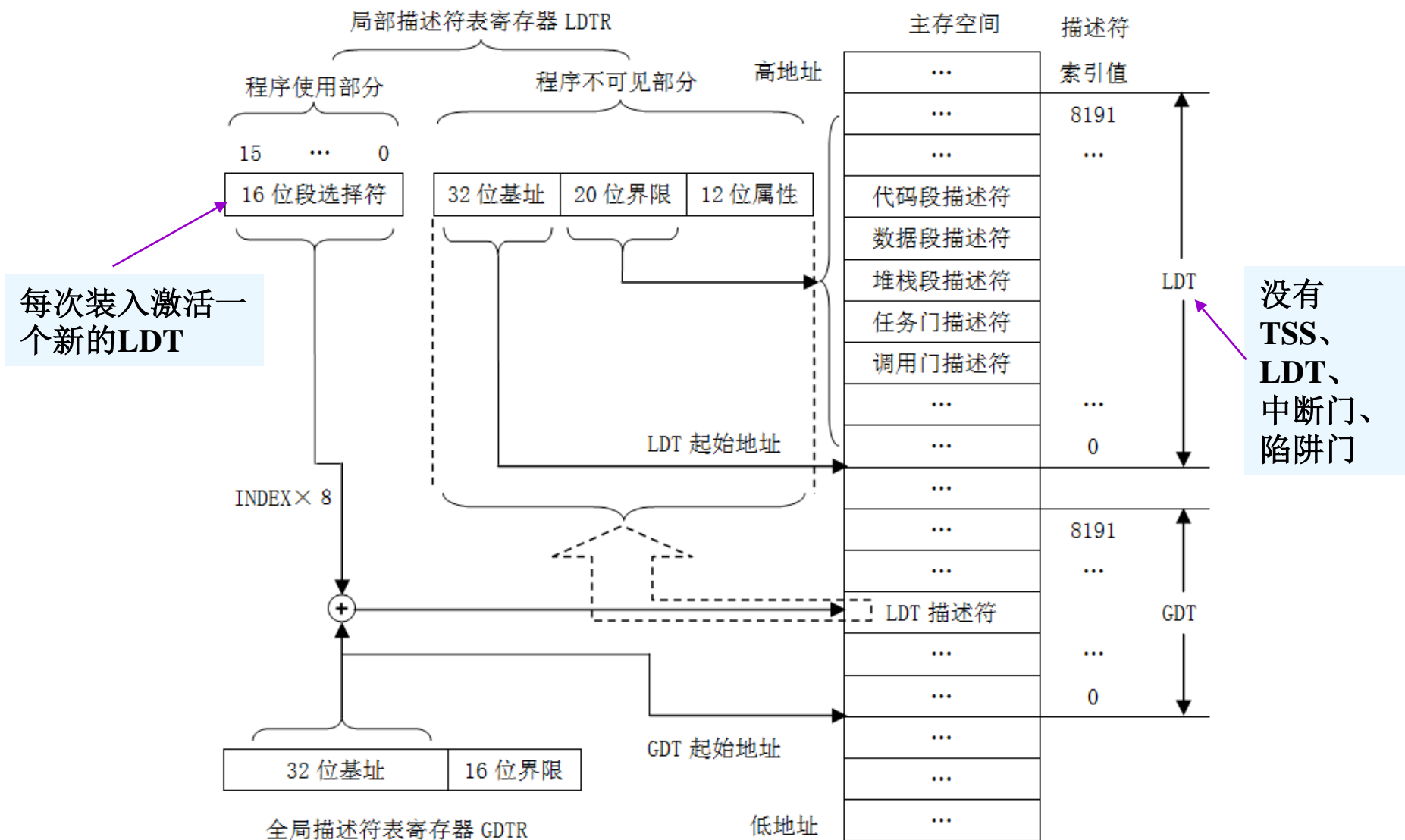


图 5.2.6 由 LDTR 确定 LDT 存储位置和界限

## 5.2.5 中断描述符表及寄存器

- **中断描述符表IDT**：保存门描述符，整个系统一个，包括中断门、陷阱门、任务门（通常没有调用门）。
- 门提供了一种将程序控制转移到中断服务程序入口的手段。每个门8个字节，包含服务程序的属性和起始地址。

- **中断描述符表寄存器IDTR**：

48位 = 32位基地址 + 16位界限

IDT按字节计算大小，IDT最大可达64KB（但是Pentium微处理机只能支持256个中断和异常，最多占用2KB）。

- 中断描述符表IDT中存放的描述符类型均是门描述符。

# 由IDTR确定IDT存储位置和界限

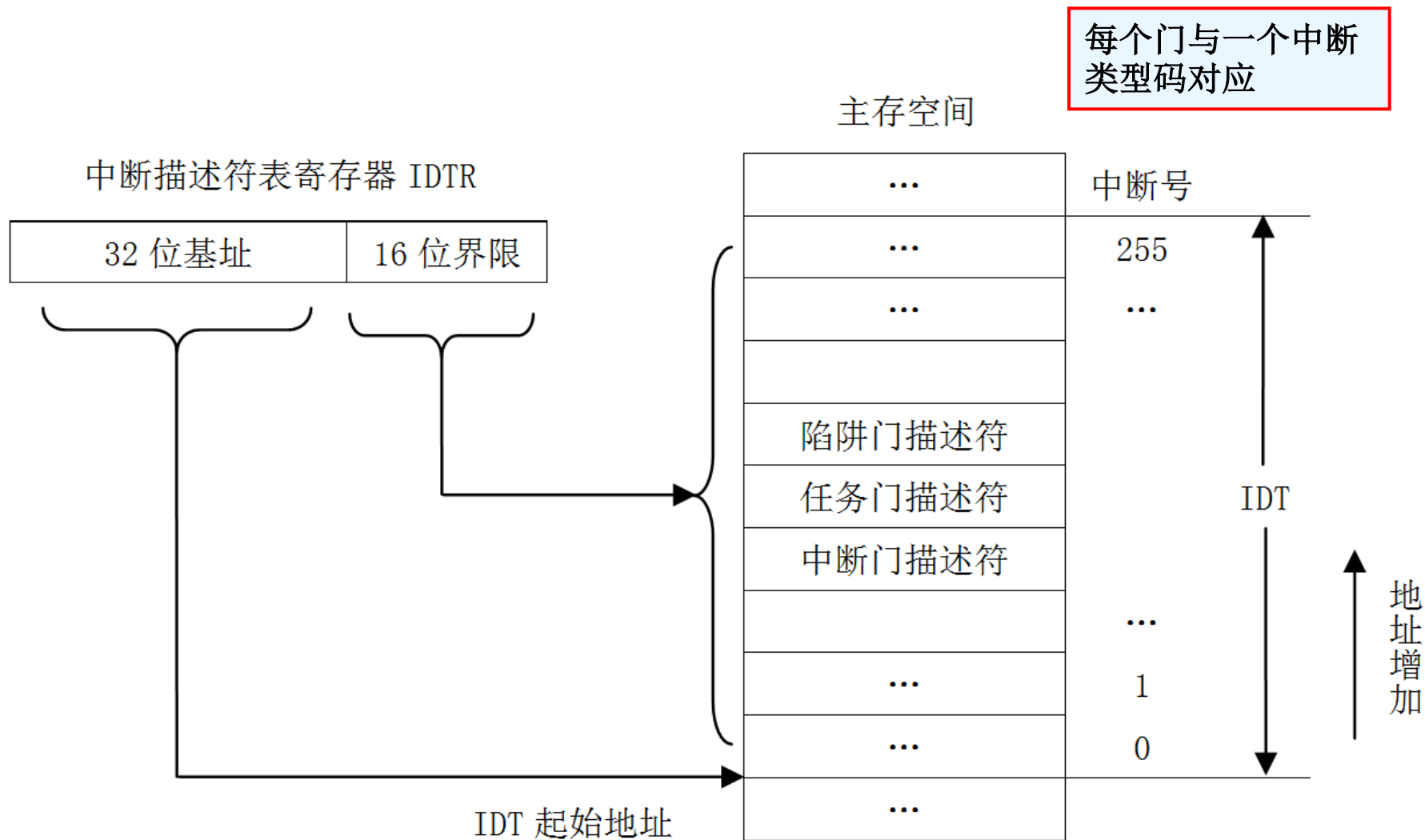


图 5.2.7 由 IDTR 确定 IDT 存储位置和界限

## 5.2.6 任务状态段及寄存器

- **任务状态段TSS:** 保存现有任务的机器状态（指处理器的工作环境，比如各个寄存器的状态）及其任务间的关联信息。
- 没有数据和代码，每个任务一个。



图 5.2.8 TSS 的格式

# TSS的格式

(1) **返回链BACK\_LINK**: 是一个段选择符, 把前一个任务的TSS描述符的段选择符(即原来TR中的16位可见部分)转入新任务TSS中, 供任务返回时使用。

(2) 即由返回指令IRET将其装入TR寄存器, 从而回到前一个TSS。

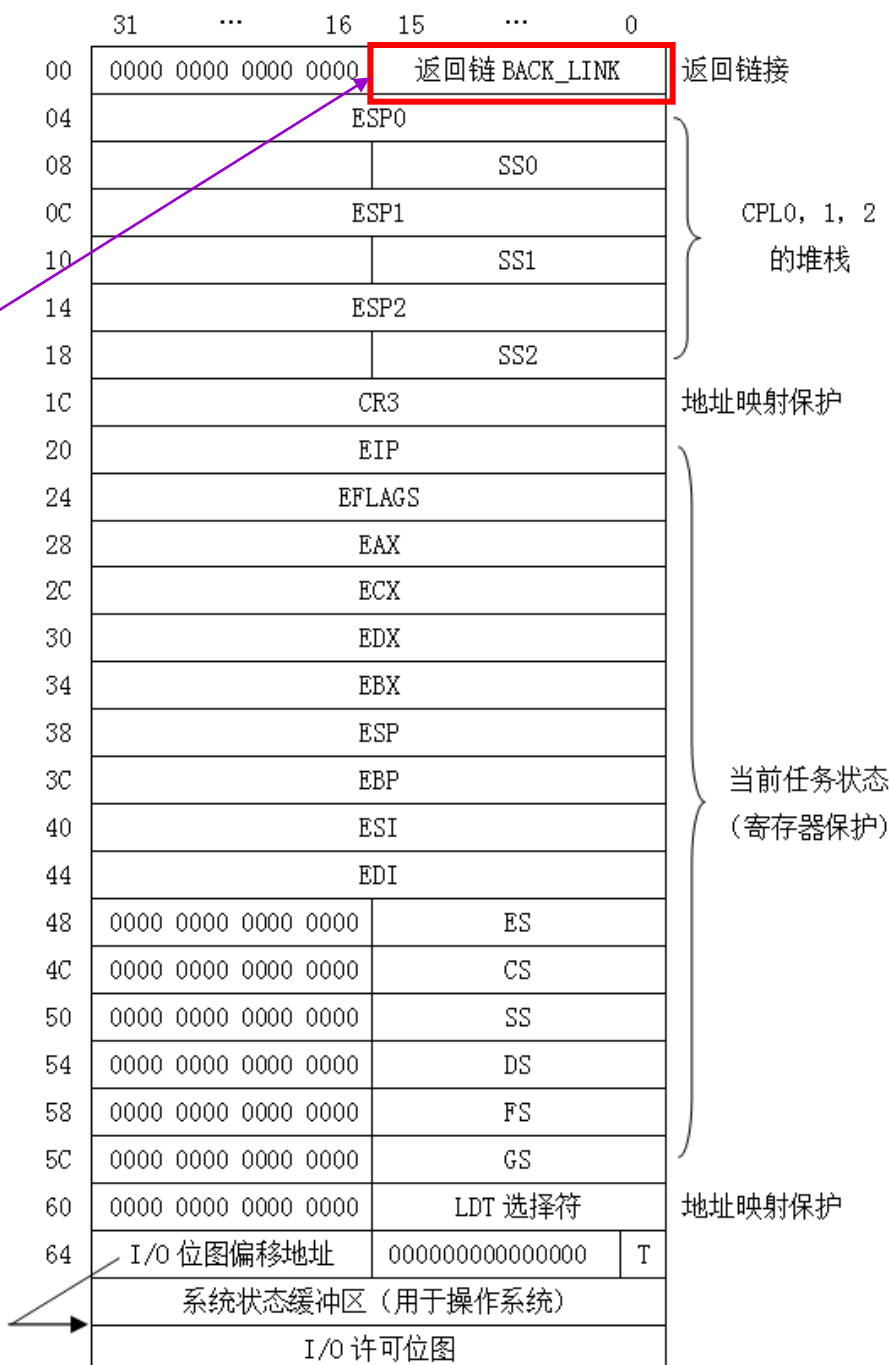


图 5.2.8 TSS 的格式



# TSS的格式

- (1) 偏移量4H到偏移量18H双字中包含特权级0~2的ESP和SS值。
- (2) 当前任务被中断时要用这些值来对特权级0~2的堆栈进行寻址。
- (3) 为了有效地实现保护，同一个任务在不同的特权级下，使用不同的堆栈。
- (4) 当从某一个特权级A变换到另一个特权级B时，任务使用的堆栈也同时从A级变换到B级。
- (5) 没有指向3级堆栈的指针，因为3级是最低特权级，任何一个向高特权级的转移都不可能转移到3级。



图 5.2.8 TSS 的格式

# TSS的格式

(1) 偏移量1CH双字包含CR3的内容。

(2) CR3中保存前一个状态的页目录寄存器的地址。

(3) 如果分页有效，则必须保存这项信息。



图 5.2.8 TSS 的格式

# TSS的格式

(1) 偏移量20H到偏移量5CH双字的值被装入指定的寄存器。

(2) 这部分是寄存器保存区域，用于保存通用寄存器、段寄存器、指令指针和标志寄存器。

(3) 每当任务切换时，处理器当前所有寄存器的内容都被保存在TSS的这些单元中，然后将新任务TSS所对应的单元内容装入所有的寄存器。

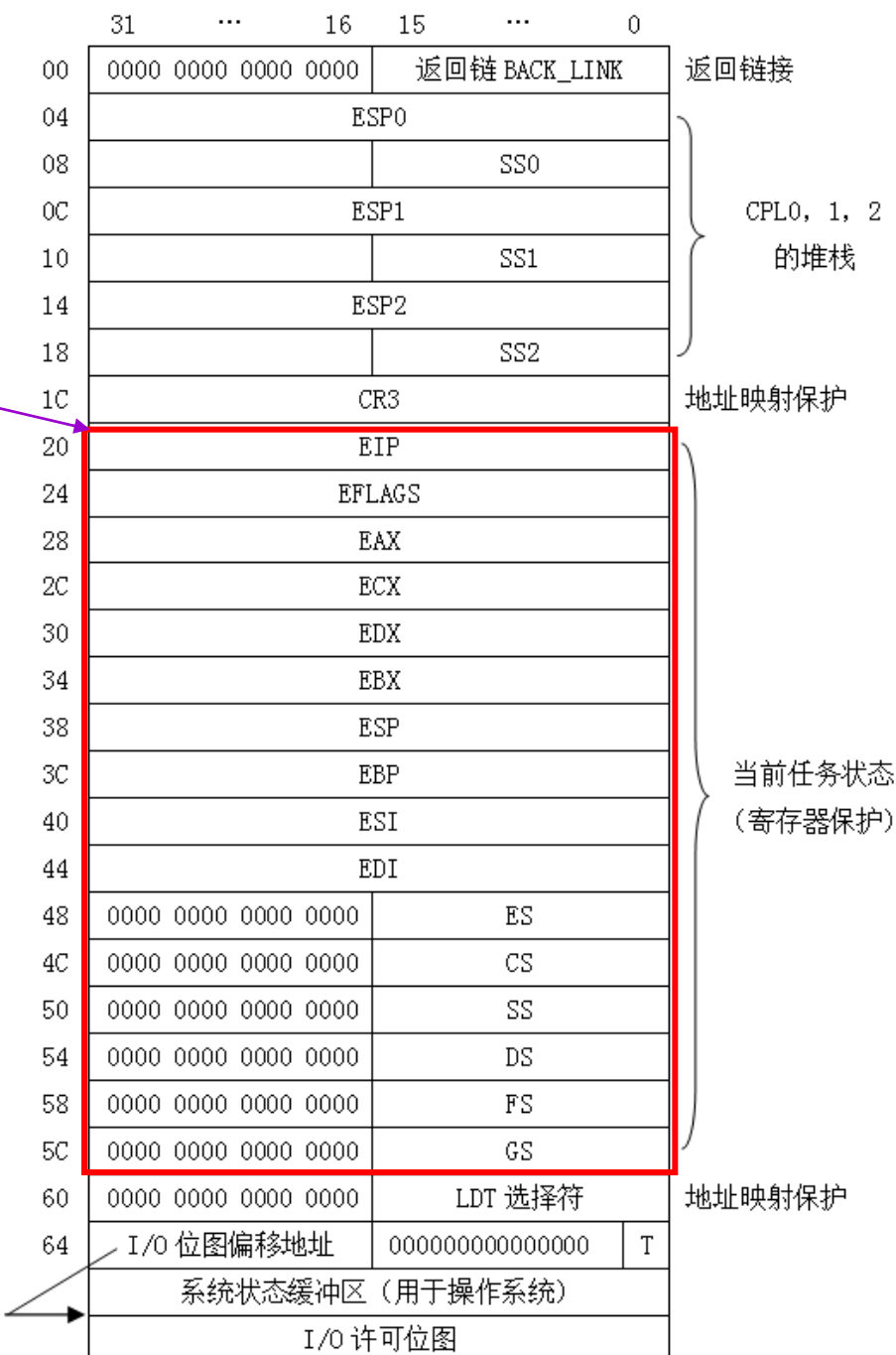


图 5.2.8 TSS 的格式

# TSS的格式

(1) 偏移量60H字标记为LDT描述符的选择符。

(2) 每一个任务都有自己的LDT，这里的“LDT描述符的选择符”就是指该TSS所对应的任务的LDT描述符的选择符。

(3) 在任务切换时，要选择新任务的LDT，需要对原来的LDTR内容进行修改，此时，该域作为LDTR选择符的修改值使用。

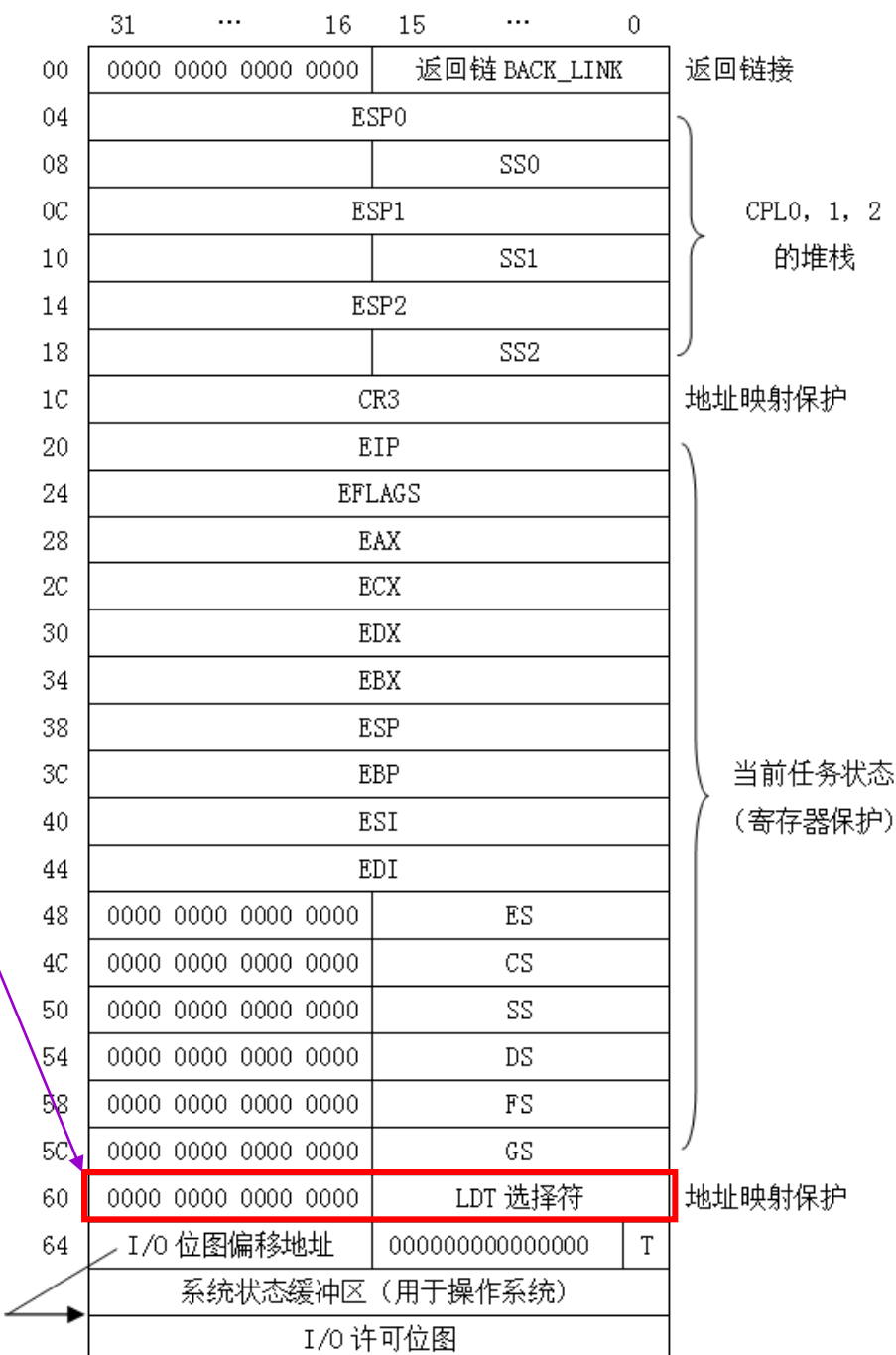


图 5.2.8 TSS 的格式

# TSS的格式

(1) 偏移量64H字的最低位T与调试状态寄存器DR6中的BT位相关联，该位用于调试。

(2) 若T=1，则进入该任务会发生调试异常。

(3) 若T=0，则进入该任务不会发生调试异常。

	31	...	16	15	...	0	
00	0000 0000 0000 0000				返回链 BACK_LINK		返回链接
04	ESP0						CPL0, 1, 2 的堆栈
08					SS0		
0C	ESP1						
10					SS1		
14	ESP2						
18					SS2		
1C	CR3						地址映射保护
20	EIP						当前任务状态 (寄存器保护)
24	EFLAGS						
28	EAX						
2C	ECX						
30	EDX						
34	EBX						
38	ESP						
3C	EBP						
40	ESI						
44	EDI						
48	0000 0000 0000 0000				ES		地址映射保护
4C	0000 0000 0000 0000				CS		
50	0000 0000 0000 0000				SS		
54	0000 0000 0000 0000				DS		
58	0000 0000 0000 0000				FS		
5C	0000 0000 0000 0000				GS		
60	0000 0000 0000 0000				LDT 选择符		
64	I/O 位图偏移地址				0000000000000000		T
系统状态缓冲区（用于操作系统）							
I/O 许可位图							

图 5.2.8 TSS 的格式

# TSS的格式

偏移量66H中包含I/O允许位图的偏移量：存放由TSS的起始地址到I/O位映像首字节的偏移量。

(1) 为了实现输入/输出保护，要使用I/O允许位图。

(2) I/O允许位图中的每一个位对应一个I/O端口，I/O允许位图的首字节为I/O端口0000H~0007H的许可位，最右端的一位为端口0000H的许可位，最左一位为端口0007H的许可位。一直到图最后一个字节的最左一位与最后一个端口0FFFFH对应。

(3) I/O允许位图中某位为逻辑0则对应I/O端口地址开放，为逻辑1则对应I/O端口地址被封锁。

	31	...	16	15	...	0	
00	0000 0000 0000 0000				返回链 BACK_LINK		返回链接
04	ESP0						CPL0, 1, 2 的堆栈
08					SS0		
0C	ESP1						
10					SS1		
14	ESP2						
18					SS2		地址映射保护
1C	CR3						
20	EIP						当前任务状态 (寄存器保护)
24	EFLAGS						
28	EAX						
2C	ECX						
30	EDX						
34	EBX						
38	ESP						
3C	EBP						
40	ESI						
44	EDI						
48	0000 0000 0000 0000				ES		地址映射保护
4C	0000 0000 0000 0000				CS		
50	0000 0000 0000 0000				SS		
54	0000 0000 0000 0000				DS		
58	0000 0000 0000 0000				FS		
5C	0000 0000 0000 0000				GS		
60	0000 0000 0000 0000				LDT 选择符		
64	I/O 位图偏移地址				0000000000000000		T
系统状态缓冲区 (用于操作系统)							
I/O 许可位图							

# 由TR确定TSS存储位置和界限

(1) TR的内容由装任务寄存器指令LTR和存任务寄存器指令STR进行装入和保存。

(2) 也可由保护模式下运行远转移JMP或远调用CALL指令来改变。

(3) 也可来自任务门。

TSS描述符由任务寄存器TR寻址。

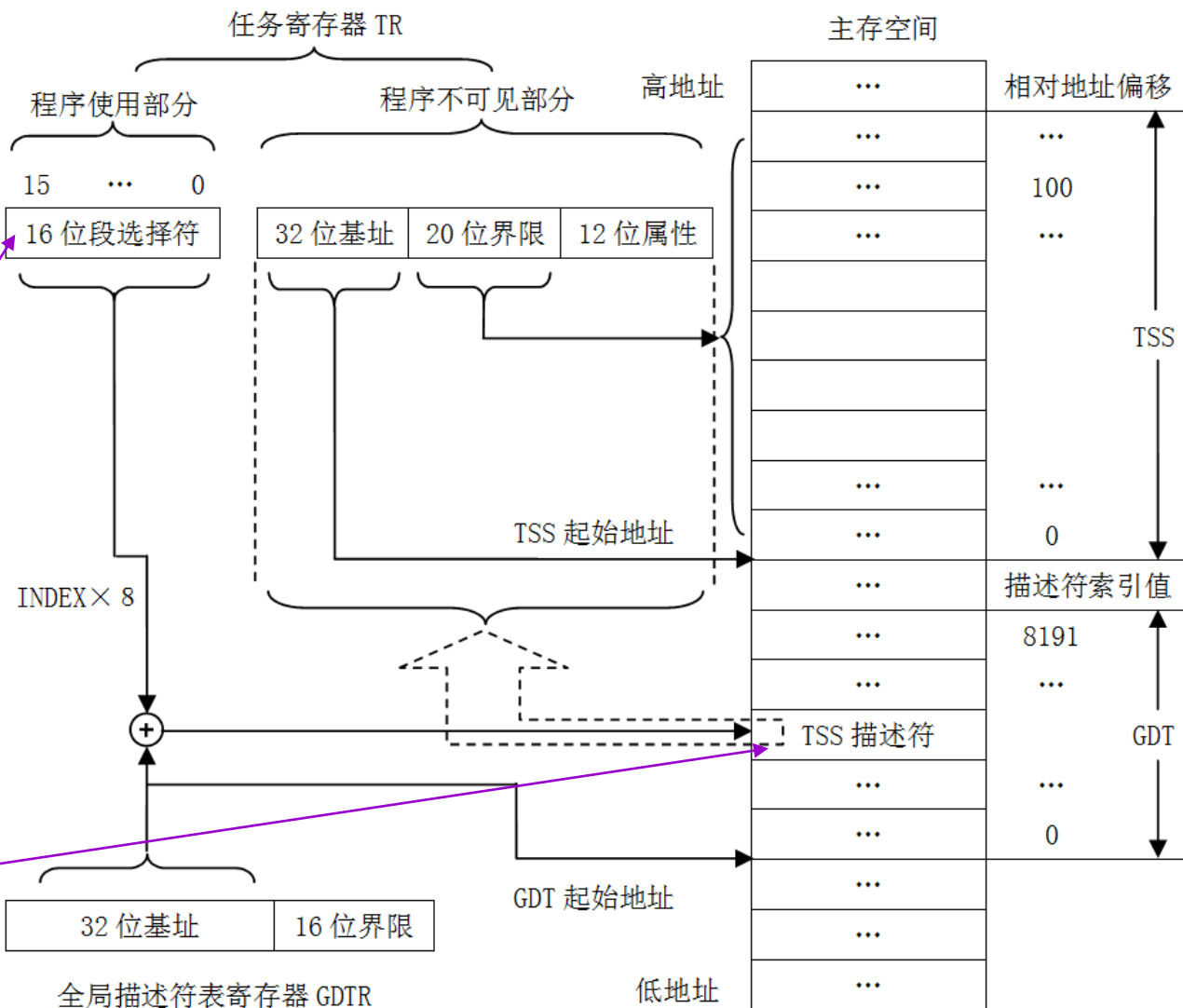


图5.2.9 由TR确定TSS存储位置和界限

## 5.2.7 段选择符及寄存器

- 在实模式下，段寄存器的16位值是基地址。
- 在保护模式下，每一个段寄存器由两部分组成：  
可见部分 + 不可见部分（高速缓存）

16位

64位

(1) 可用传送指令MOV装入，装入的是16位的段选择符。

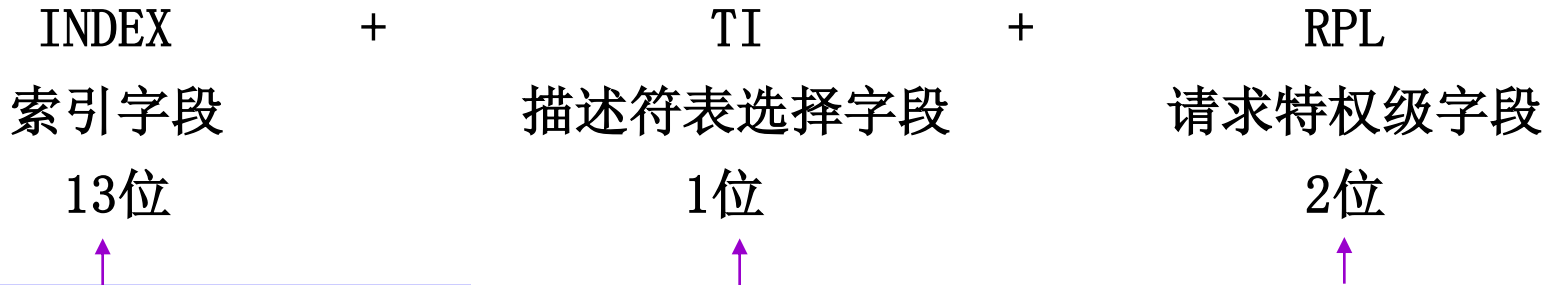
(2) 段选择符用于识别（选择）在全局描述符表GDT或局部描述符表LDT内登记的段描述符。

只能由处理机装入，用户不能干预



# 1. 段选择符

- 选择符分为3个字段：



索引值乘以8就是相对于GDT或LDT首址的偏移量，这个偏移量再加上描述符表的基地址（来自全局描述符表寄存器GDTR，或者局部描述符表寄存器LDTR）就是段描述符在描述符表中的地址。

(1) 当TI=0时，选择的是全局描述符表GDT。  
(2) 当TI=1时，选择的是局部描述符表LDT。

(1) 有4个特权级，00、01、10、11，称作0级、1级、2级、3级。  
(2) 0级的特权最高，1级次之，3级的特权最低。

## 2. 段选择符装入段寄存器的操作

- 装入段寄存器的指令有2类：
  - （1）直接的装段寄存器指令：可使用传送指令MOV、弹出堆栈指令POP、加载段寄存器指令LDS、LSS、LGS、LFS。这些指令都是显式地访问段寄存器。
  - （2）隐含的装段寄存器指令：可使用调用一个过程指令CALL、远转移指令JMP。这种指令更改代码段寄存器CS的内容。

## 5.3 保护模式下的访问操作与保护机制

- 5.3.1 保护机制的分类
- 1. 任务间存储空间的保护
- 任务间的保护是通过每一个任务所专用的LDT描述符实现的。根据LDT描述符，每个任务都有它特定的虚拟空间，因而避免各任务之间的干扰，起到隔离、保护的作用。
- 2. 段属性和界限的保护
- 当段寄存器进行加载时，需要进行段存在性检查以及段限检查。
- 在段描述符中给出了20位的段界限值，每当产生一个逻辑地址时，都要比较偏移量和段限值。一旦偏移地址大于段限值，CPU就终止执行命令，并发出越限异常。由此限制每个程序段只在自己的程序、数据段内运行，不相互干扰。
- 最后，还要对该段的读写权限进行检查。
- 3. 特权级与特权级保护
- 特权级与特权保护是为了支持多用户多任务操作系统，使系统程序和用户的任务程序之间、各任务程序之间互不干扰而采取的保护措施。

# 3种形式的特权管理

- 3种形式的特权管理：

- (1) 当前特权级CPL

- CPL是当前正在执行的代码段所具有的访问特权级。
    - 每一项任务都是在其代码段描述符所确定的特权级中运行。当前特权级就是任务执行时所处的特权级。例如，一个正在运行的任务的CPL，就是其描述符中访问权限字节的DPL。当前特权级的值一般就是代码段描述符中的DPL。

- (2) 描述符特权级DPL

- DPL是段被访问的特权级，保存在该段的段描述符的特权级DPL位。

- (3) 请求特权级RPL

- RPL是新装入段寄存器的段选择符的特权级，存放在段选择符的最低两位
    - **特权管理规定：** 特权级为P的段中存储的数据，只能由特权级高于或等于P的段中运行的程序使用；特权级为P的代码段/过程，只能由在低于或等于P的特权级下执行的任务调用。
    - 为记忆方便，**特权级规则可以不严格地归纳为：**
      - 高特权级可以访问低（等于）特权级的数据；
      - 低特权级可以调用高（等于）特权级的程序。

## 5.3.2 数据段访问及其特权级检查

图5.3.2 数据段访问过程  
及其寻址过程

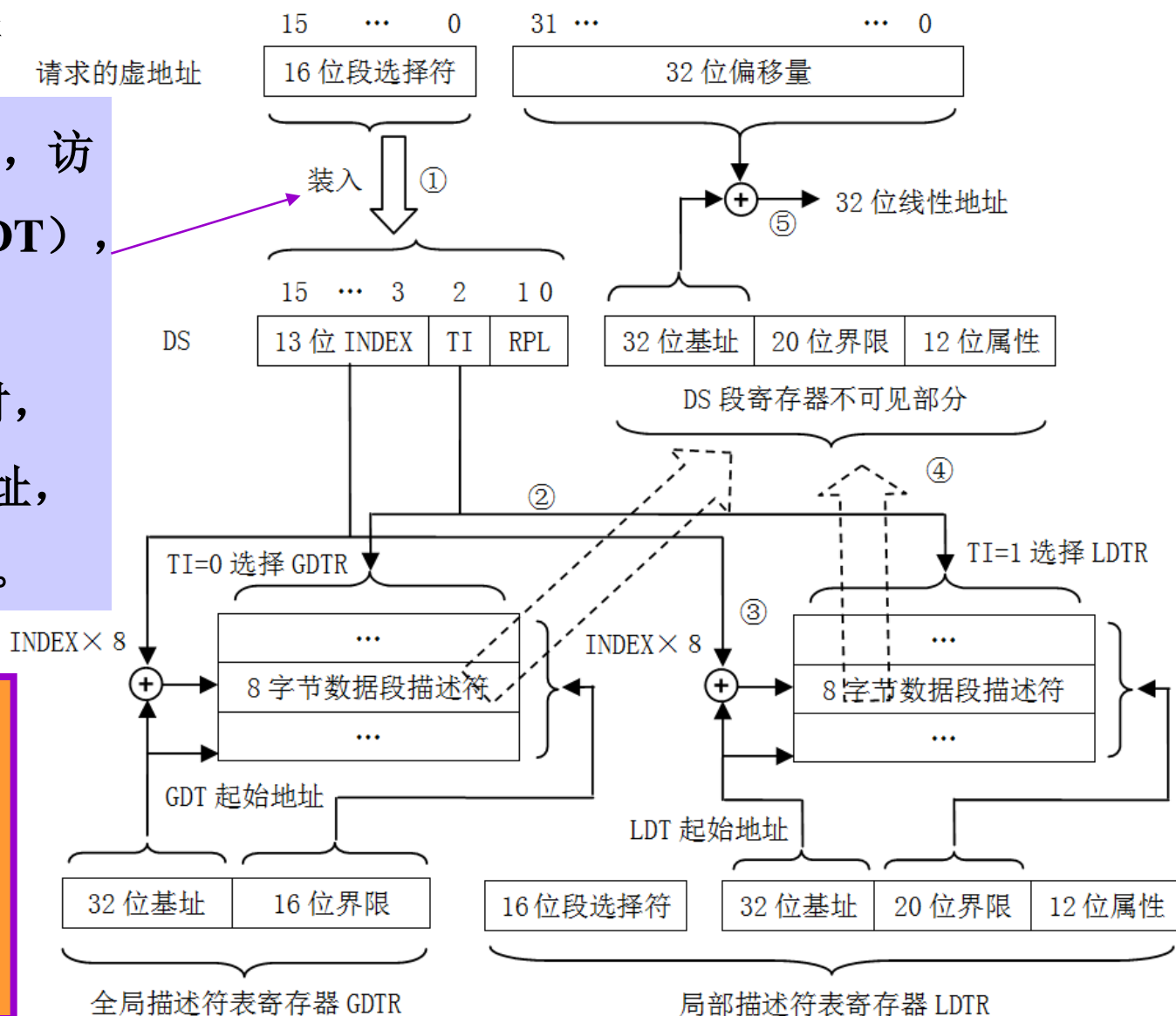
(1) 第1次装入DS, 访问内存 (GDT或LDT), 装入DS的Cache。

(2) 以后再访问时, 从DS的Cache取基址, 直接形成线性地址。

要求:

1. 能够解释①~  
⑤ 步骤。

2. 能够画出该图



# 数据段访问的特权级检查

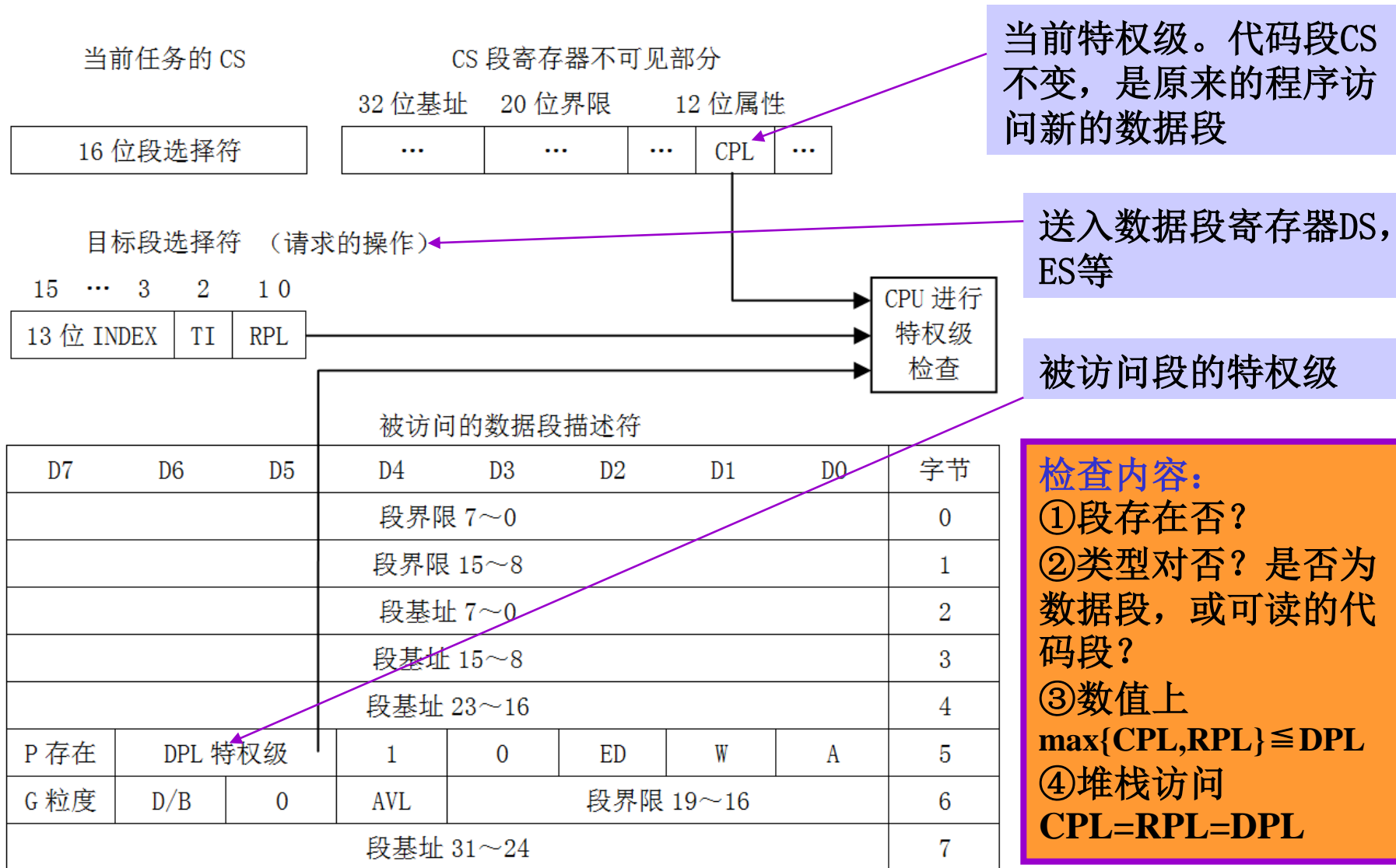


图 5.3.3 数据段访问的特权级检查

## 5.3.3 任务内的段间转移及其特权级检查



# 任务内段间控制转移的描述符访问规则

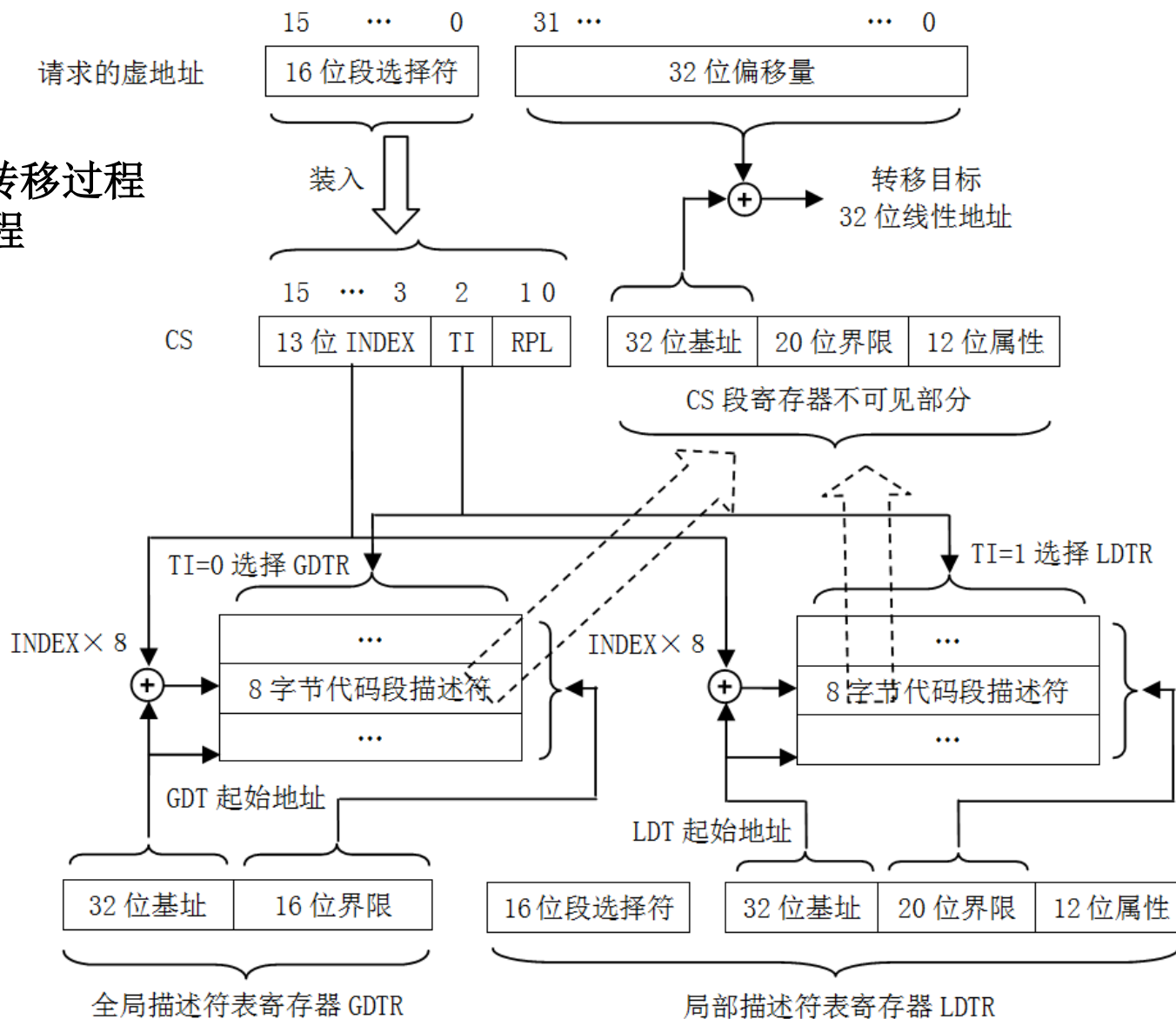
表5.3.1 任务内段间控制转移的描述符访问规则

控制转移类型	操作类型	引用的描述符	涉及的描述符表
同一个特权级	JMP、CALL、 RET、IRET*	代码段	GDT/LDT
同一个特权级， 或转移到 更高特权级	CALL	调用门	GDT/LDT
	中断指令、异常、 外部中断	陷阱门、 中断门	IDT
转移到较低特权级	RET、IRET*	代码段	GDT/LDT
注：使用IRET实现控制转移时，需要嵌套任务位NT=0			



# 1. 段间直接转移的操作过程

图 5.3.4 段间直接转移过程及其寻址过程



过程类似  
访问数据段

## 2. 段间直接转移的特权级检查

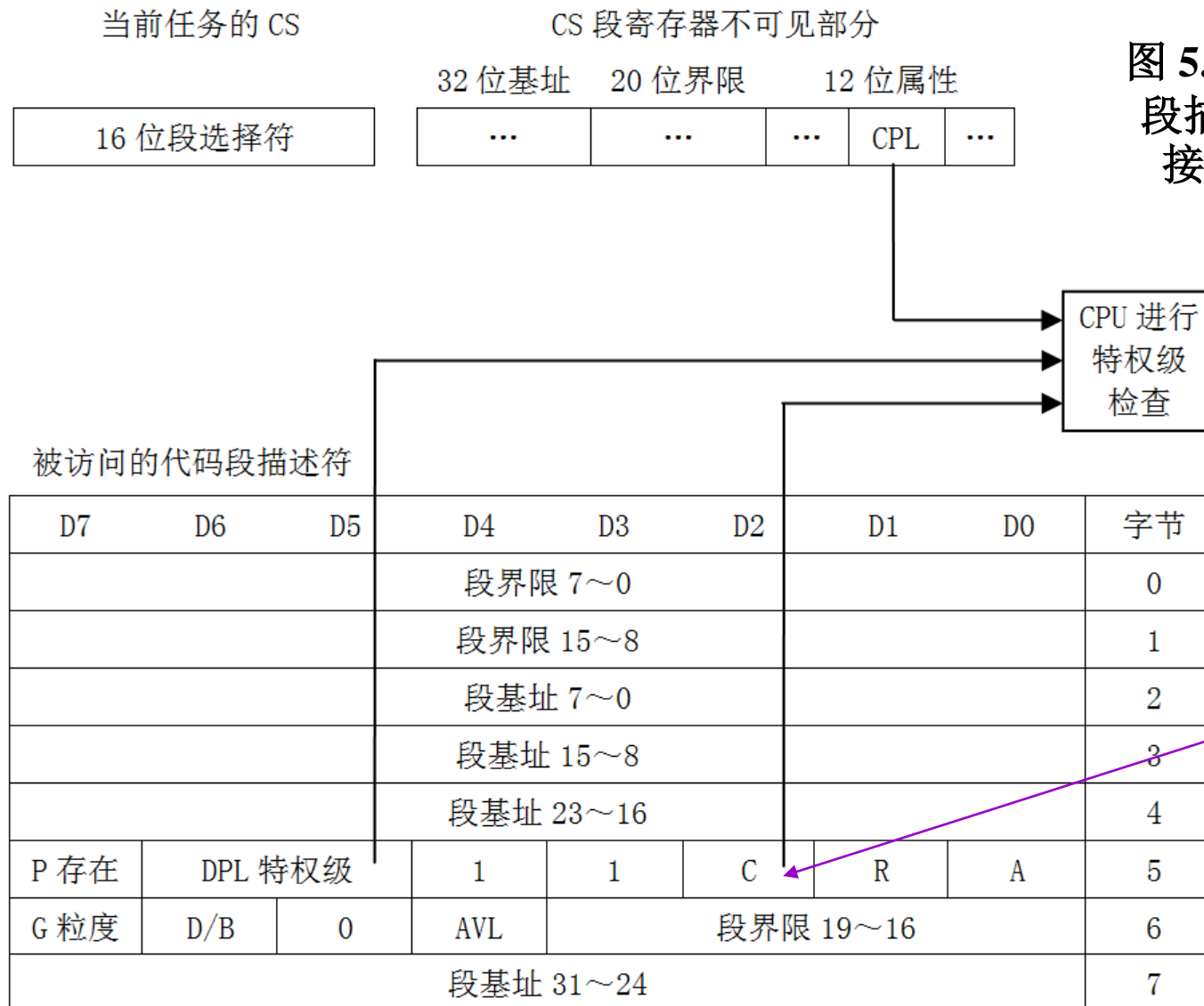


图 5.3.5 通过代码段的段描述符实现段间直接转移的特权检查

一致性检查,  
C=1有效

### 3. 段间间接转移的操作过程：使用调用门

段间，任务内间接调用（改变特权级）

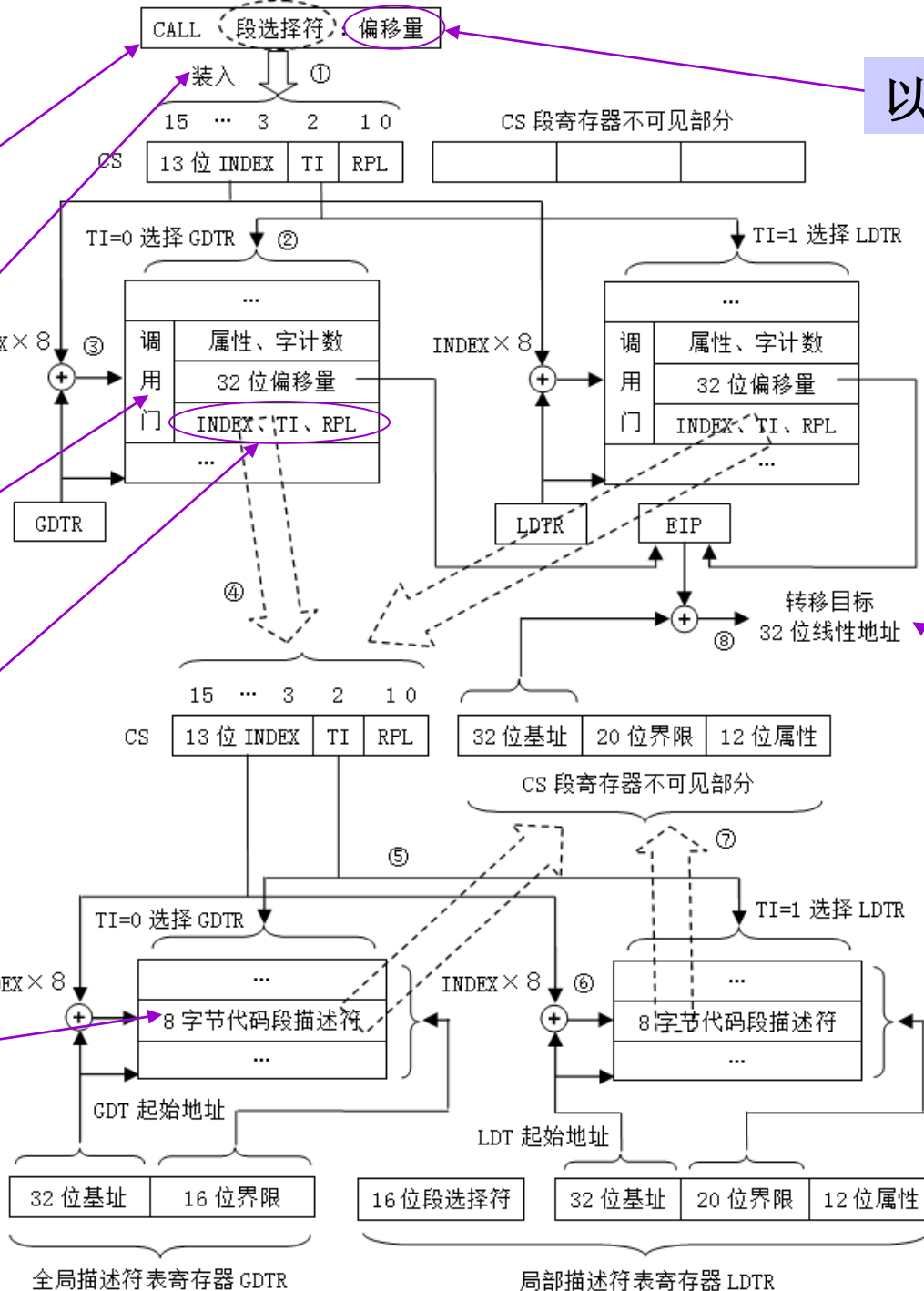
由指令装入

调用门按数据段保护

目标段的段选择符，装入CS

目标段按代码段保护

图5.3.6 用调用门实现段间间接转移的过程



以后不用

调用的  
目的地址

## 4. 段间间接转移的特权级检查

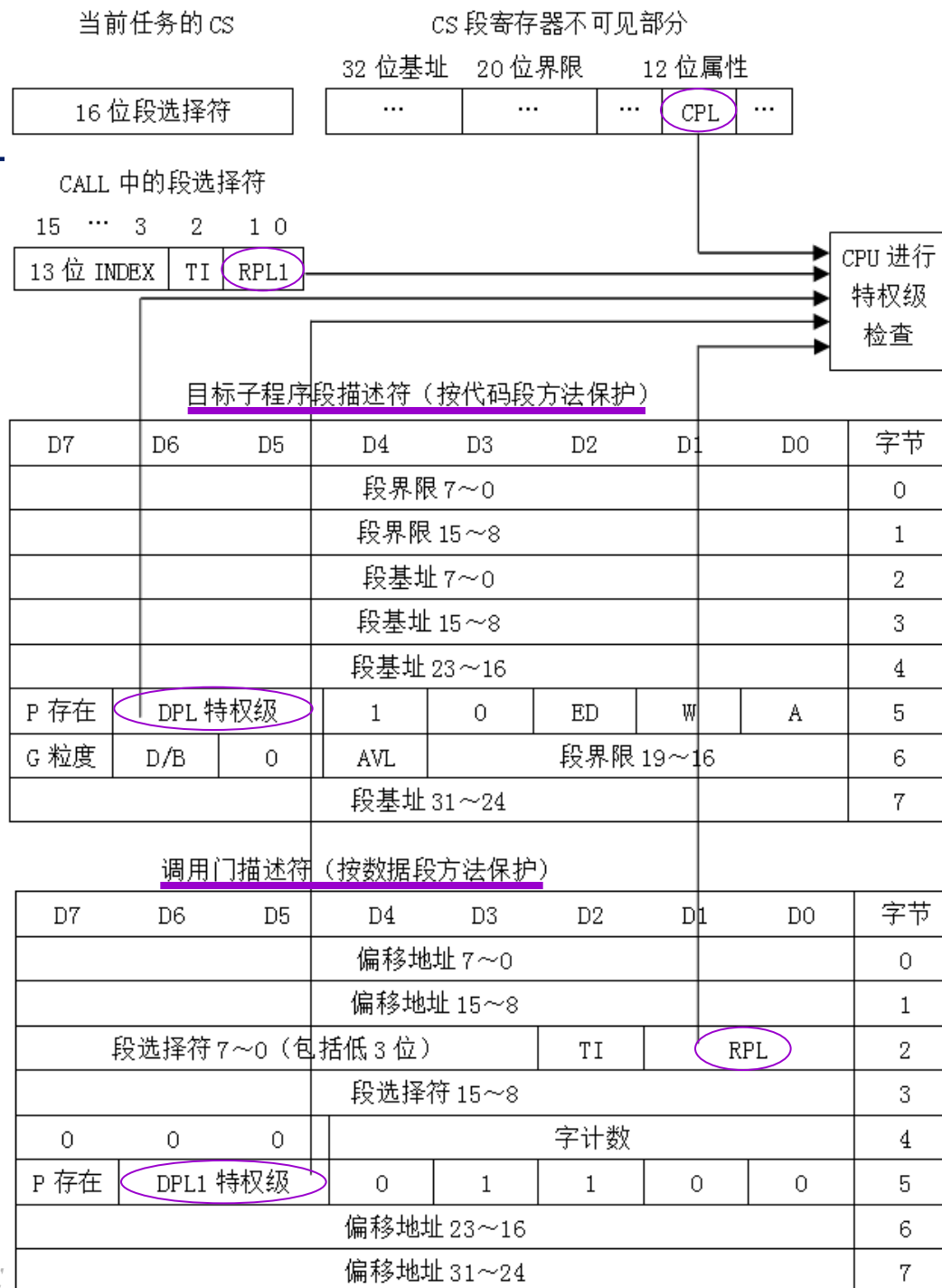


图 5.3.7 使用调用门进行段间间接转移时的特权级检查

## 5.3.4 任务切换及其特权级检查

- **任务切换**：指从执行某一个任务转换到执行另外一个任务的过程。
- **任务切换的过程**：保存机器的整个状态（比如所有的寄存器、地址空间、到原来任务的链接等），装入新的执行状态，进行保护检查，开始新任务的执行，执行完毕后回到原来的任务继续执行。
- **1. 任务的设定**
  - 在执行某任务以前，必须在存储器中定义GDT、IDT、LDT和TSS，在GDT中登记（写入）所需要的段描述符、门描述符、LDT描述符、TSS描述符，在IDT中登记（写入）所需要的中断门、陷阱门、任务门等，并且还必须对GDTR、IDTR、LDTR、TR设置适当的数值。
  - TR给出TSS段的基址。
- **2. TSS描述符和任务门**
  - 在任务切换中，通常用到任务状态段TSS和任务门。
  - 每一个任务必须有一个任务状态段TSS与其关联。
  - TSS描述符属于系统描述符类（属性字节中S=0），该描述符包含了TSS在内存中的基址和界限。TSS描述符位于GDT中，所以指向TSS描述符的段选择符的TI位应该为0。

### 3. 任务切换的方法

- 在进行任务切换时，要把新任务的TSS描述符的选择符传送到TR的选择符字段。
- 对TR的选择符字段有**两种修改方法**：
  - (1) **直接任务切换**：直接访问新任务的TSS描述符，从而得到新任务的TSS。在直接任务切换中，段间JMP/CALL指令的操作数的段选择符就是新任务的TSS描述符的选择符，它被直接加载到TR的选择符字段，对于执行IRET指令的情况（必须NT=1），则是把曾经压入到当前执行任务的TSS中的返回链（返回链就是前一个任务的TSS描述符的段选择符，即原来TR中的16位可见部分内容），作为TR选择符字段的修改值。
  - (2) **间接任务切换**：新任务的TSS描述符的选择符由任务门加载。通过任务门间接访问新任务的TSS描述符，从而得到新任务的TSS。在间接任务切换中，段间JMP/CALL指令的操作数的段选择符是任务门的选择符，而任务门的内容包含新任务的TSS描述符的选择符，所以，新任务的TSS描述符的选择符将由任务门间接加载到TR的选择符字段，

# 任务切换的方法

- 这样，对任务的切换，可以采用以下方法：
  - (1) **段间JMP/CALL指令**：进行直接任务切换或间接任务切换。
  - (2) **INT指令**（包括异常中断和外部中断）：只能进行间接任务切换。  
访问IDT中的任务门，新任务的TSS描述符的选择符由任务门加载。当中断/异常发生时，如果IDT的目标项是中断门或陷阱门，则执行中断处理程序。如果目标项是任务门，则进行任务切换。
  - (3) **IRET指令（当NT=1时）**：只能进行直接任务切换。EFLAGS寄存器的NT位必须为1，表明是处于任务嵌套。NT为0时，执行IRET指令与正常中断处理程序最后执行IRET指令的结果相同，即只完成正常的中断返回，不进行任务切换。



# 4. 任务切换的过程

开始任务切换

(1) 一个CALL指令，到底执行什么操作，调用还是任务切换，需要看2件事：CALL产生一个选择符，于是对应一个段描述符，描述符中看① S位、② TYPE类型。

(2) S=1程序段，直接处理。

(3) S=0为系统段，再看TYPE，若为TSS则直接任务切换，若为任务门则间接任务切换，若为调用门则间接程序调用/转移，其他均按类型操作。

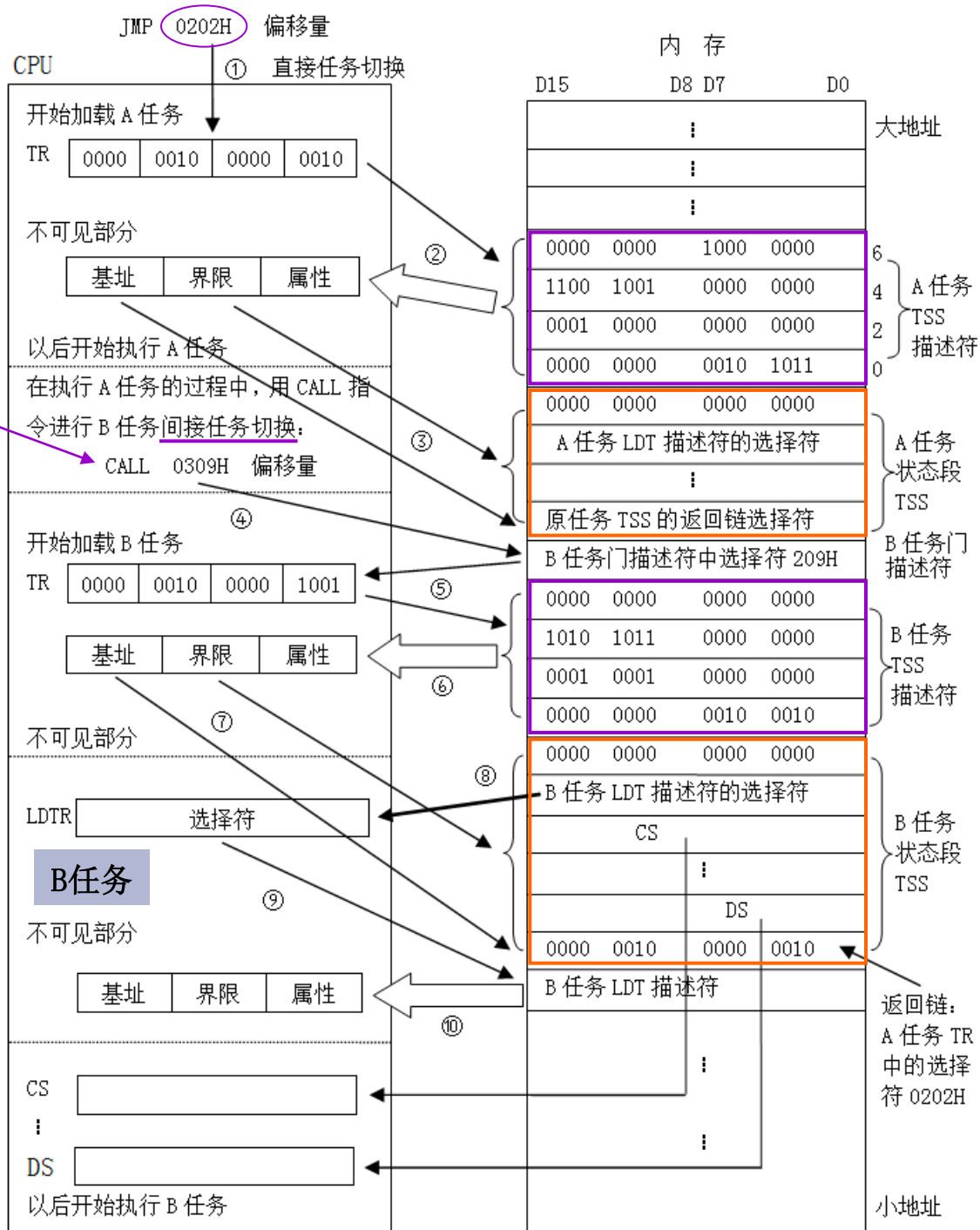


图5.3.8 任务间的切换过程示意图



## 5.3.5 保护模式下的中断转移操作

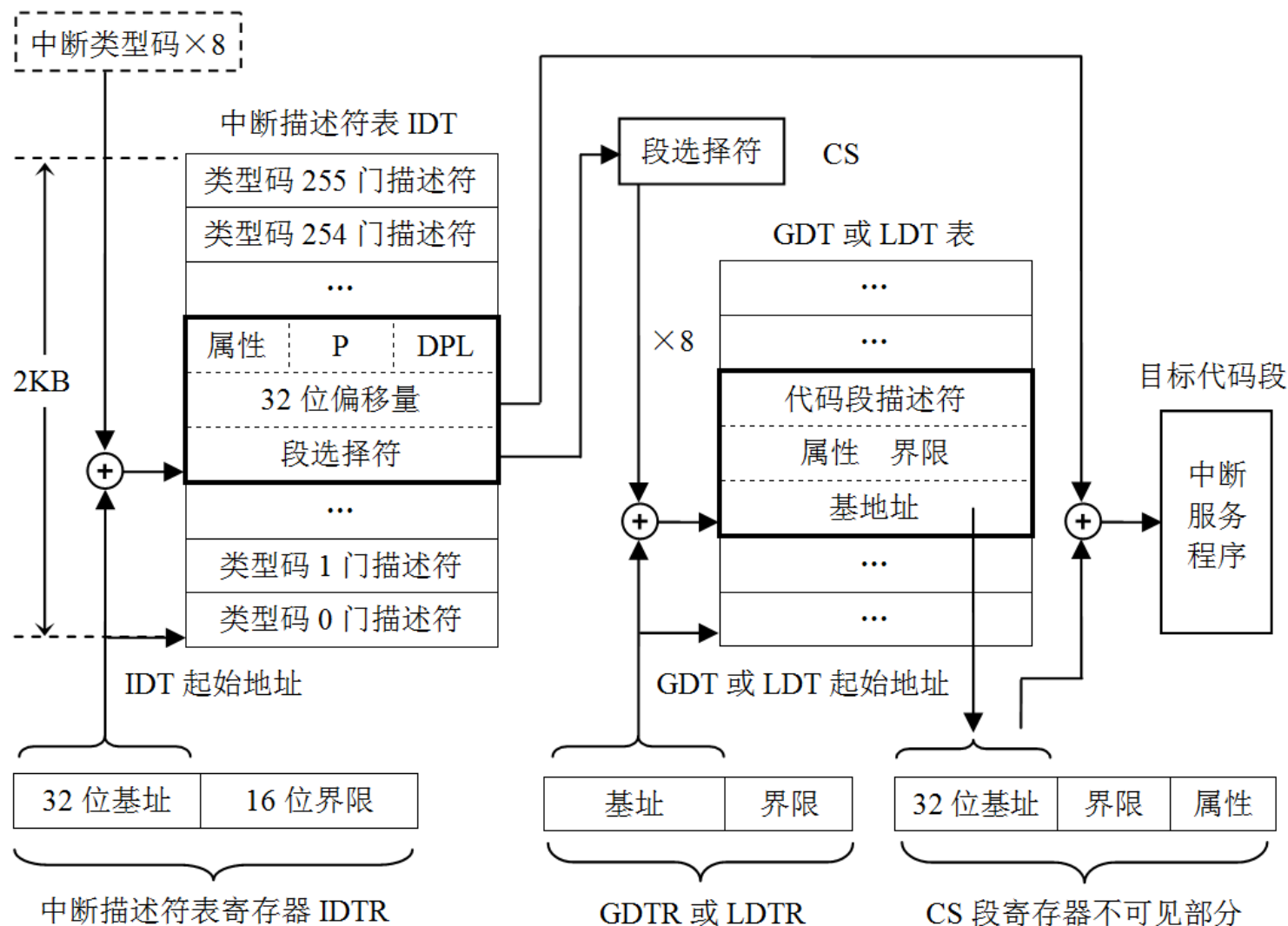


图 5.3.9 Pentium 微处理器在保护模式下中断转移的过程示意图

## 5.4 向保护模式的转换

- 从实地址方式到保护模式的切换步骤:

- (1) 初始化中断描述符表IDT，使其包含至少前32种中断类型有效的中断门描述符。
- (2) 初始化全局描述符表GDT，使其第0项为一个空描述符，并且使其至少包含一个数据段描述符、一个代码段描述符、一个堆栈段描述符。
- (3) 通过指令MOV CR0, R/M 使CR0寄存器中的PE位置1。使Pentium置为保护模式。
- (4) 进入保护模式后，执行一条段间远JMP指令清除内部指令队列并把TSS描述符基址装入到TR中。
- (5) 将初始数据段选择符的值装入到所有的数据段寄存器中。
- (6) 现在Pentium已运行在保护模式下，正在使用GDT和IDT中定义的段描述符。

# 向保护模式的转换

- 另一种利用任务切换使Pentium进入到保护模式的步骤：
  - (1) 初始化中断描述符表IDT。
  - (2) 初始化全局描述符表GDT，使其最少有两个任务状态段TSS描述符和初始任务所需要的原始代码段及数据段描述符。
  - (3) 初始化任务寄存器TR，使它指向一个TSS，当初始任务发生切换并访问新的TSS时，当前寄存器值将保存在这个原始的TSS中。
  - (4) 进入保护模式后，执行一条段间远JMP指令清除内部指令队列，切换到保护模式下。将当前的TSS选择符装入到TR寄存器中。
  - (5) 用一条远转移指令装载TR寄存器，以便访问新的TSS并保存当前状态。
  - (6) 现在Pentium已运行在保护模式下。

## 5.5 分页存储管理

- 分页是虚拟存储器多任务操作系统另一种存储器管理方法。段的长度是可变的，而页的长度是固定的，比如每页4KB。
- **分页：**将程序分成若干个大小相同的页，各页与程序的逻辑结构没有直接的关系。
- 分页存储器的这种固定大小页有一个缺点，就是存储管理程序每次分配最少是一个页（即使它们并不全用）。
- **碎片：**一页中未用的存储器区域称为碎片，碎片导致存储器使用效率降低，但是分页大大简化了存储管理程序的实现。
- Pentium微处理器采用**二级页表方法**对页面进行管理，**第1级页表**称作**页目录**，页目录中的页目录项指明**第2级页表**中各页表的基址。

## 5.5.1 页目录与页表

- **页目录基地址寄存器CR3:** 保存页目录的基地址，该基地址起始于任意4KB的边界。指令MOV CR3, reg用来对CR3寄存器进行初始化。
- **页故障线性地址寄存器CR2:** 保存检测到的最后引起故障的32位线性地址。
- **页目录:** 由页目录项组成，页目录项包含下一级页表的基址和有关页表的信息。Pentium微处理器中，页目录最多包含1024个页目录项，每个页目录项为4个字节，所以，页目录自身占用一个4KB的页面（存储页）。
- **页表:** 由页表项组成，页表项包含页面（存储页）的基址和有关页面的信息。Pentium微处理器中，页表最多包含1024个页表项，每个页表项为4个字节，所以，页表自身也占用一个4KB的页面（存储页）。

# 页目录项格式

D31	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
页表基址	AV	AI	L	0	0	0	A	PCD	PWT	U/S	R/W	P	
页表 起始地址	系统保留位 系统可 任意使用						访问	页面 Cache 禁止	页面 写直 达	用户/ 系统	读/写	存在	

20位

图 5.5.1 页目录项的格式

页目录项是页表描述符，  
4个字节

0=只读

1=存在

# 页表项格式

D31	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
页面基址	AV	AI	L	0	0	D	A	PCD	PWT	U/S	R/W	P	
物理页面 起始地址	系统保留位 系统可 任意使用						修改	访问	页面 Cache 禁止	页面 写直达	用户/ 系统	读/写	存在

20位

如该页多长时间  
未被访问

图 5.5.2 页表项的格式

页表项是页描述符，  
4个字节

## 5.5.2 分页转换机制

- 在分页转换机制中，当要访问一个操作单元时，32位线性地址转换为32位物理地址是通过两级查表来实现的。
- 分页机制的工作过程如下：
  - (1) 4KB长的页目录存储在由CR3寄存器所指定的物理地址。此地址常称为根地址。
  - (2) 用线性地址中的最高10位（A31-A22）进行页目录索引。
  - (3) 用线性地址中的A21-A12这10位进行页表索引。
  - (4) 以物理页的起始地址为基址，再加上线性地址的最低12位（A11-A0）页内偏移量，就确定了所寻址的物理单元。



# 1. 分页转换的工作过程

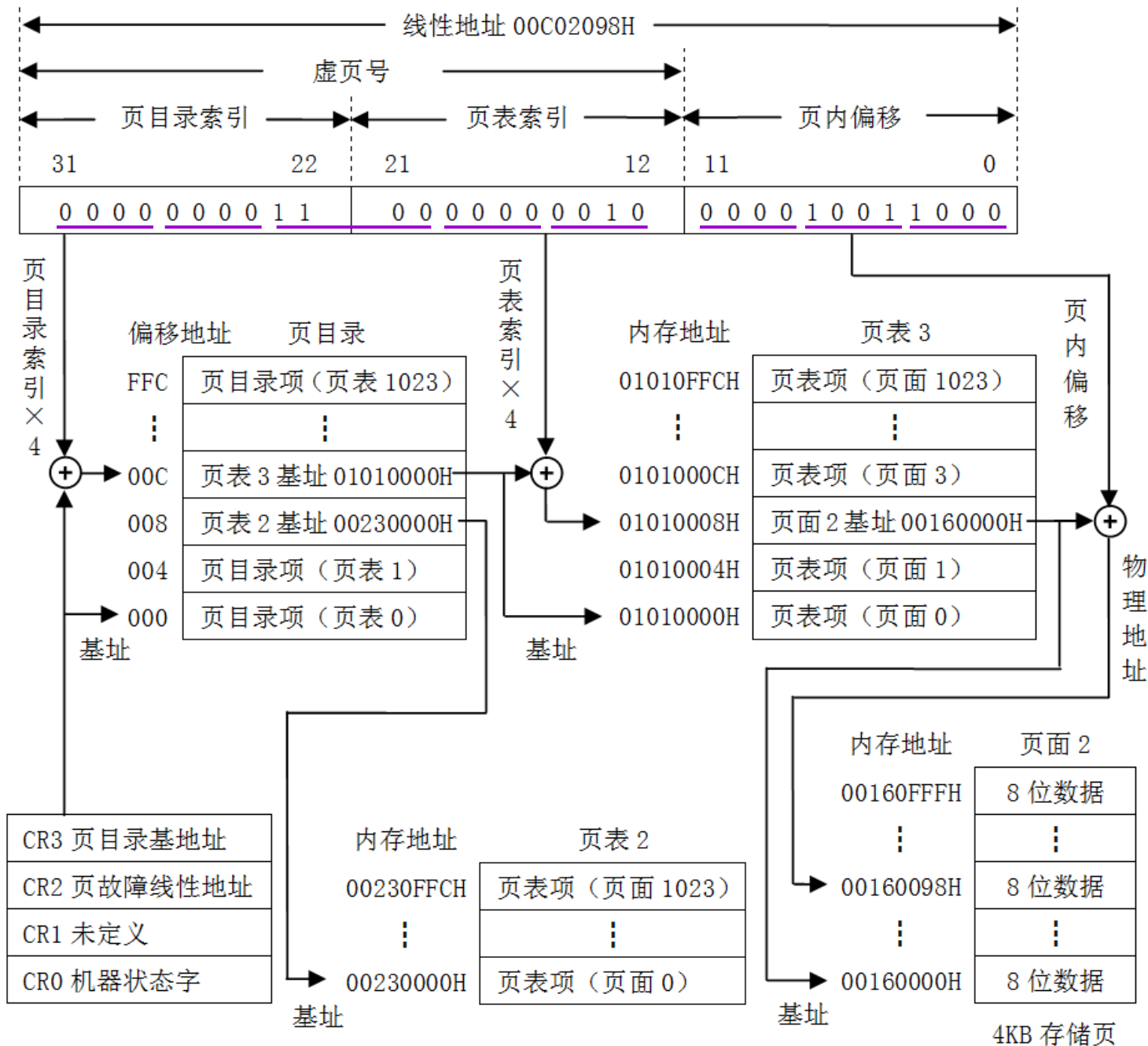


图 5.5.3 分页转换机制示意图

## 2. 4MB 页的管理机制

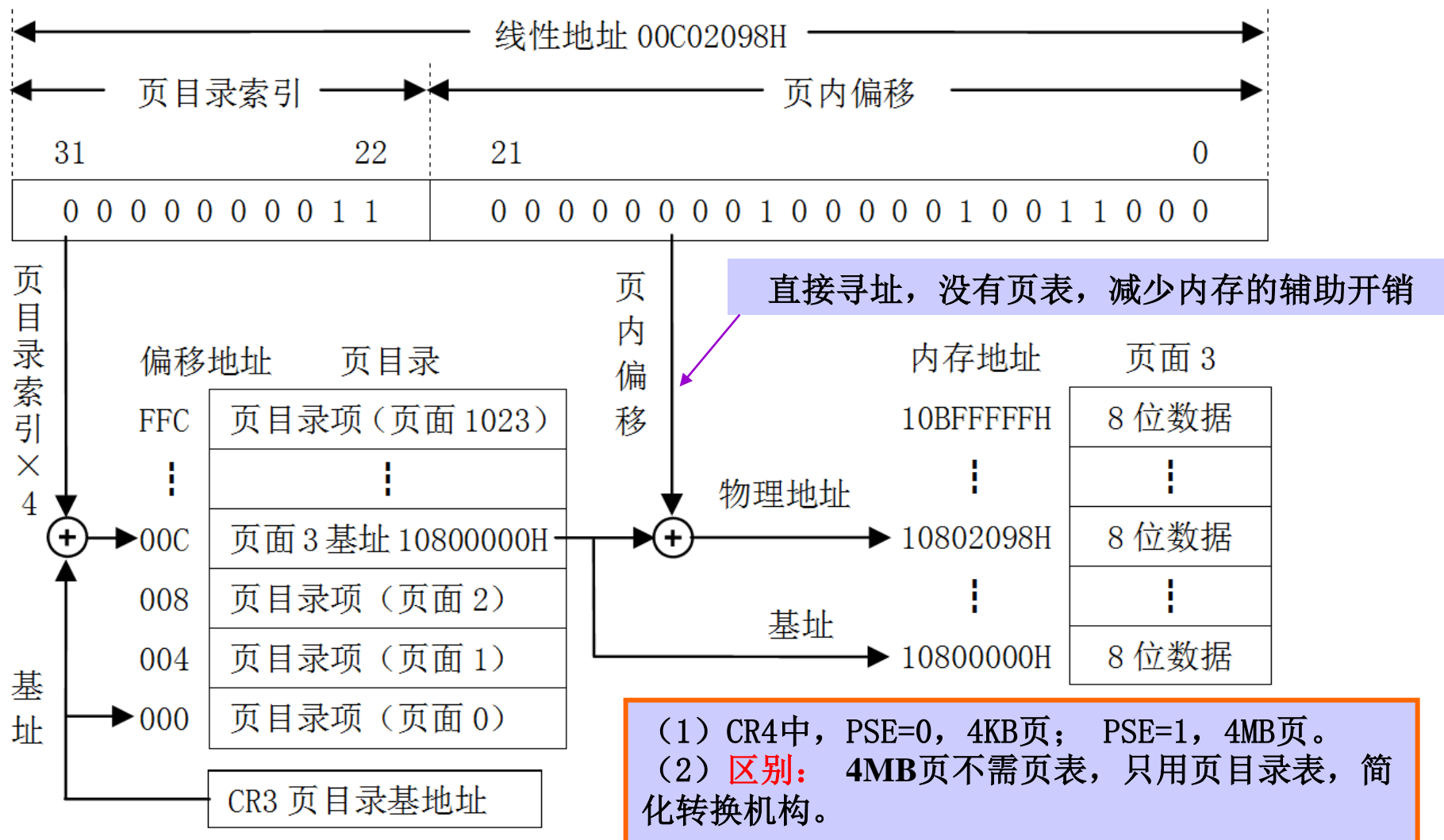


图 5.5.4 4MB 分页转换机制示意图

## 5.5.3 转换旁视缓冲存储器

- **两次访问内存：**页目录和页表都存放在主存中，当进行地址变换时，处理器要对主存访问两次，这样将极大地降低微机的性能。
- 为了提高由线性地址向物理地址的转换速度，Pentium微处理器设有一个高速转换旁视缓冲存储器TLB。
- **高速转换旁视缓冲存储器TLB组成：**4组高速缓冲寄存器，每组8个寄存器，每个寄存器可存放一个线性地址（高20位，即A31-A12）和与之对应的页表项。
- TLB按照最近频繁使用的原则可存放32项。当32项存满后而又有新的页表项产生时，按照最近最少使用的原则置换其中最少使用的项。
- TLB技术采用高速硬件进行地址变换，地址变换非常快，所以又称TLB为快表，相对而言，存于主存中的页表称作慢表。

# 通过TLB进行地址转换示意图

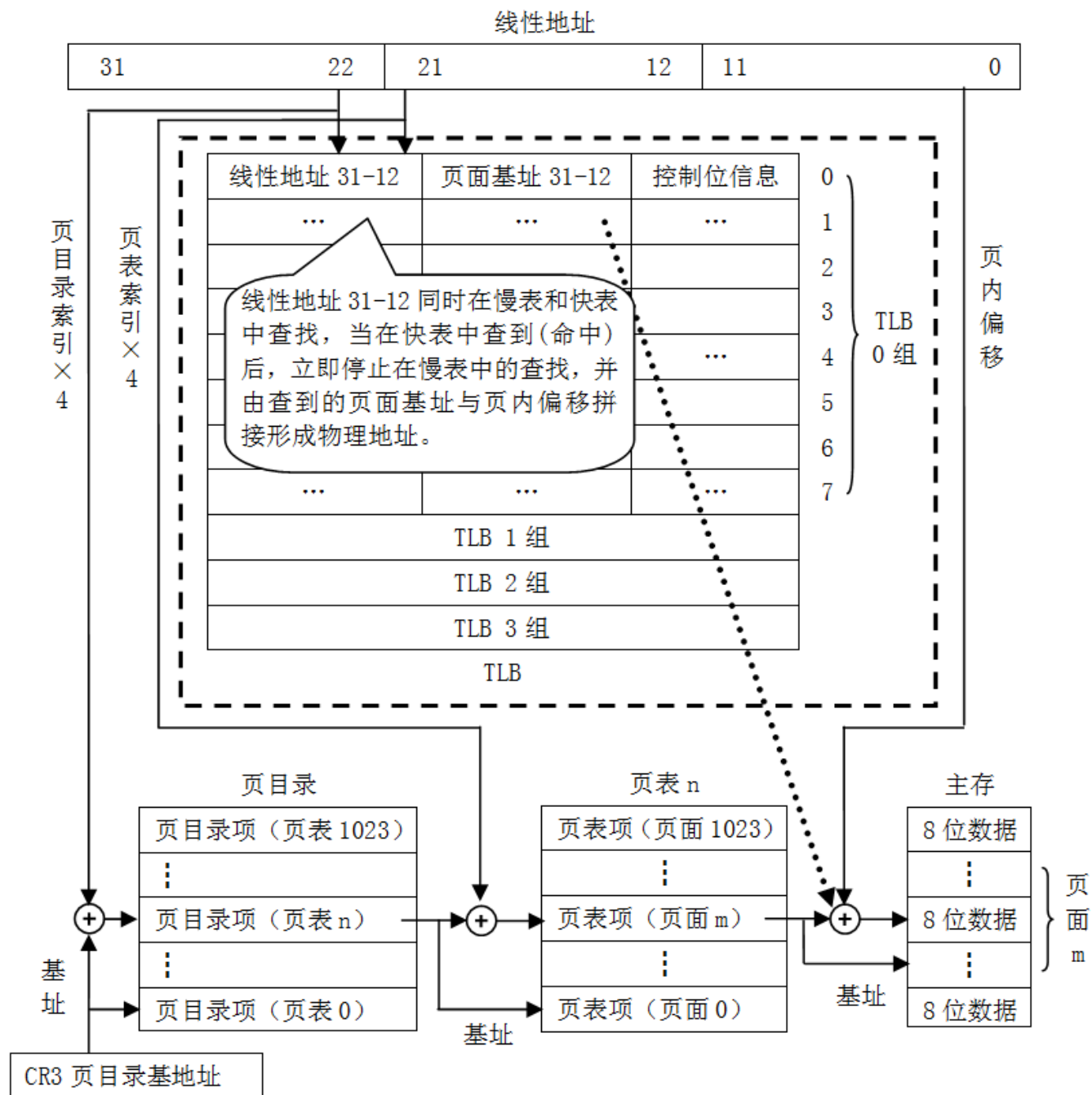


图 5.5.5 通过 TLB 进行地址转换的示意图

## 5.6 段页式存储管理的寻址过程

- 在段页式存储管理的寻址过程中，首先将虚地址通过段式存储管理部件转换为线性地址，然后将线性地址通过页式存储管理部件转换为物理地址。
- 在保护模式下，存储器的管理具有分段管理模式、分页管理模式、段页式管理模式等3种，**3种模式的特点：**

**(1) 分段不分页。**此时，一个任务拥有的最大空间是64T，由分段管理部件将二维虚地址（段选择符，偏移量）转换成一维的32位线性地址，这个线性地址就是物理地址。不分页的好处是：不用访问页目录和页表，地址转换速度快。缺点是：大容量的段调入调出，比较耗时，不够灵活。

**(2) 分段分页。**由分段管理部件和分页管理部件共同管理。

**(3) 不分段分页。**此时分段管理部件不工作，分页管理部件工作。程序不提供段选择符，只用32位寄存器地址（作为线性地址）。

# 段页式存储管理的寻址过程

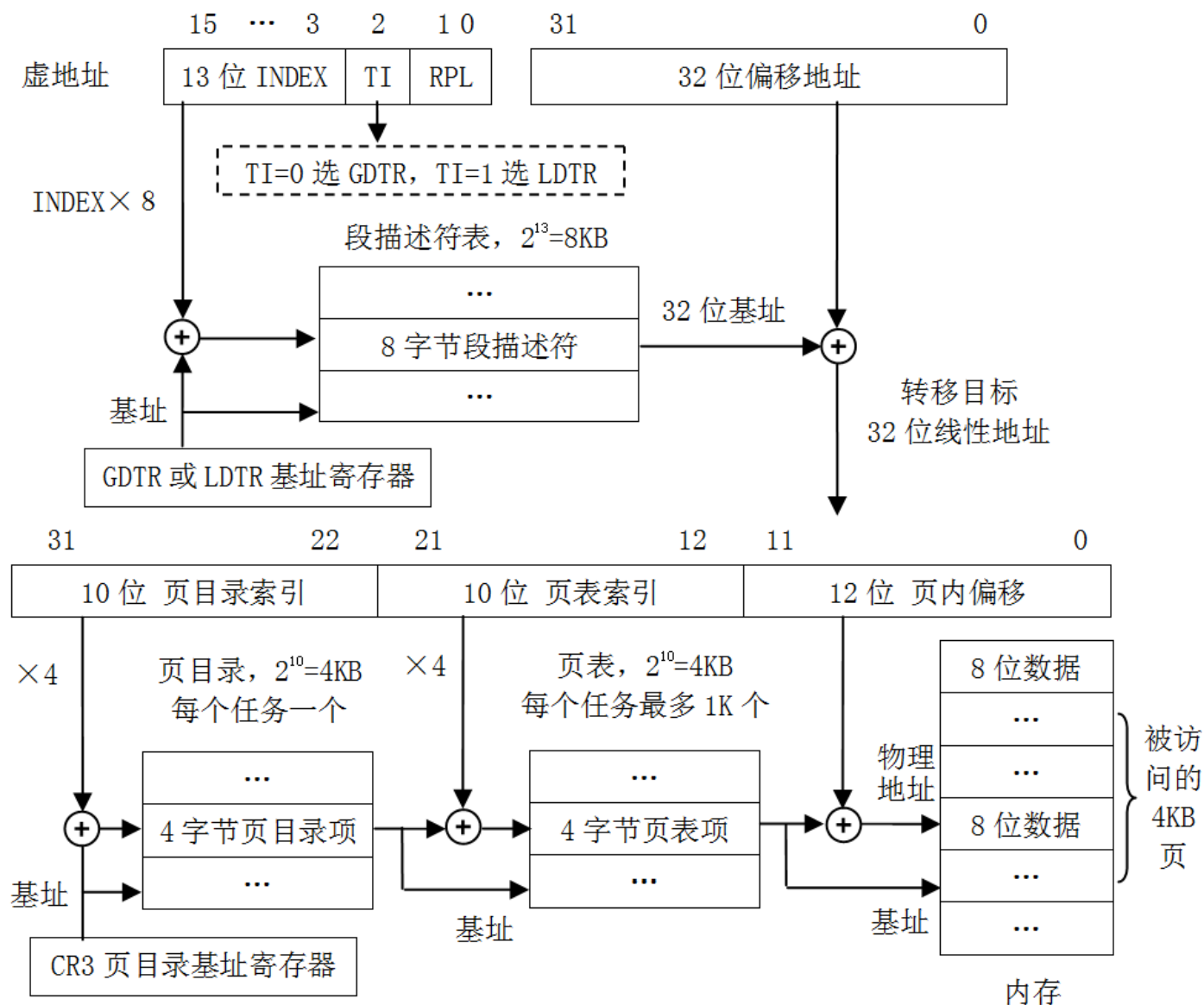


图 5.6.1 段页式存储管理的寻址过程

## 5.7 虚拟8086模式

- **虚拟8086模式：**特殊运行模式。这种特殊运行模式的设计，使得多个8086实模式的应用软件可以同时运行。
- **Pentium保护模式和虚拟8086模式之间的主要区别：**微处理器对段寄存器的解释方式不同。在虚拟8086模式下，段寄存器与在实模式下的使用方式相同，能寻址从00000H到0FFFFFFH的1MB存储空间。程序访问的是1MB以内的存储器，而微处理器可以访问存储系统中4GB范围内的任意物理存储单元。
- **启动虚拟8086模式有两种方式：**
  - (1) 通过任务切换给标志寄存器赋值。
  - (2) 通过中断返回。在这种情况下标志寄存器的内容从堆栈中重新装入。

---

结 束