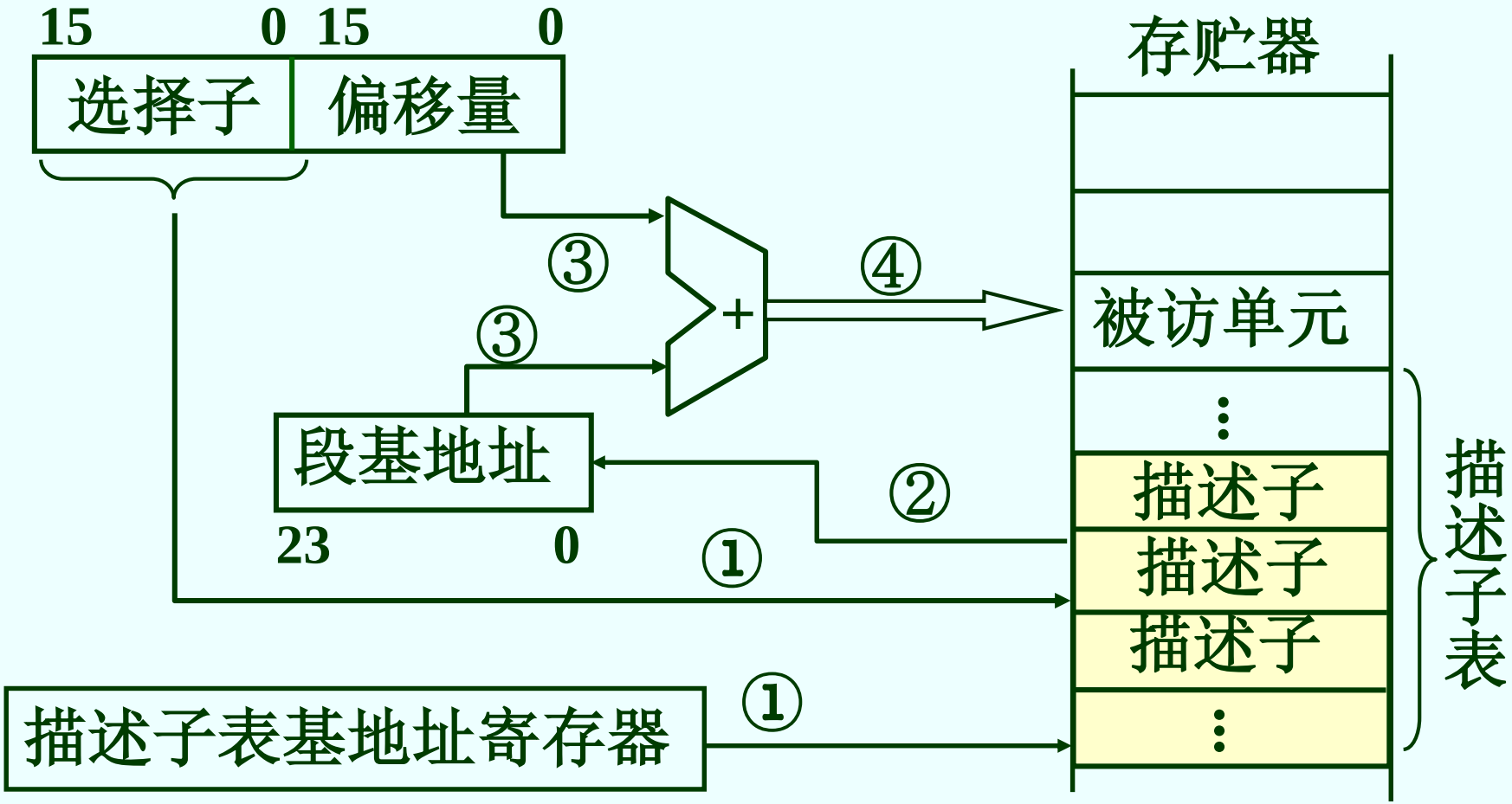
保护模式下寻址过程示意图:

虚地址(选择子, 偏移量)



3、描述子结构和选择子

(1) 数据/代码段描述子

描述子是一个数据结构，用于描述所对应的(或所描述的)那个存储段的访问属性。

访问属性主要包括:

1) 一个存储段可以被哪一特权级的任务访问
2) 该段的大小
3) 该段的读写/可执行权限
4) 该段的基地址

数据/代码段描述子的结构

- **$BASE_{23\sim16}$ $BASE_{15\sim0}$：**
  描述子所描述的那个段的段基地址(即:$A_{23\sim16}A_{15\sim0}$)
- **Limit (段限):**
  该段最后一个字节的偏移量, 表明了该段的大小。

- A: 所描述的段是否被访问过

  该段已被访问过, 则 A←1
  该段未被访问过, 则 A←0
  } 该位与时钟相结合, 可进行段淘汰

- S: 描述子类型

  1 数据代码段描述子
  0 系统描述子(如门描述子/任务状态段描述子)

- **DPL:** 规定可以访问该描述子所描述的那个段的任务的最低特权级。
- **P:** { 0 该描述子所描述的段不在物理空间
  1 该描述子所描述的段在物理空间
- **TYPE:** 由三位构成，即：数据段 (E, ED, W) 或 代码段 (E, C, R)

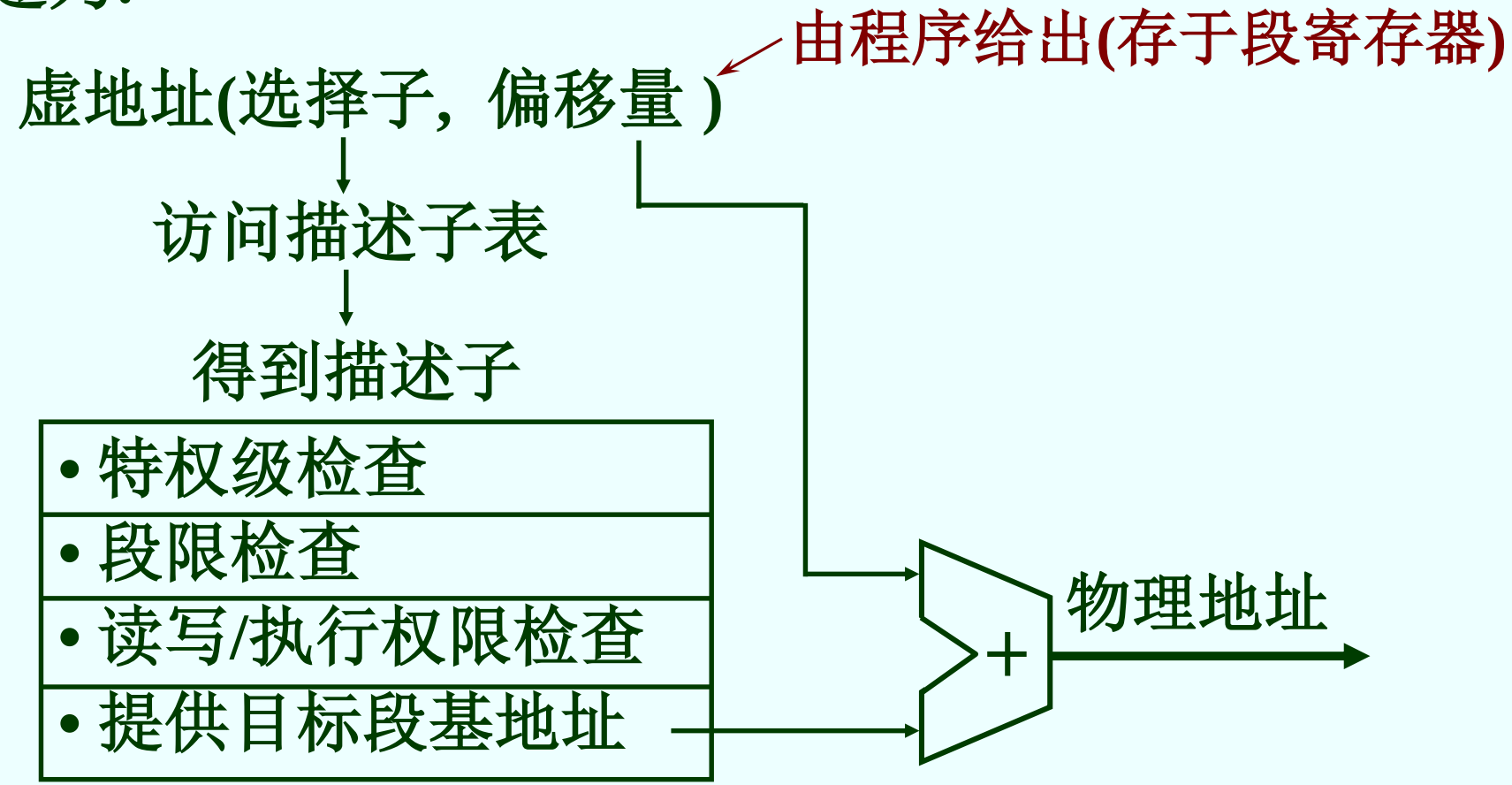
- 若该段为数据段，则 E=0

E=0
- ED
  - 0: 段向上生长，则要求偏移量 ≤ 段限
  - 1: 段向下生长，则要偏移量 ≥ 段限
- W
  - 0: 数据段只能读，不能写
  - 1: 数据段可读、可写

- 若该段为代码段，则 E=1

E=1
- C
  - 0 非一致性代码段
    访问和被访问代码段特权级相同
  - 1 一致性代码段
    访问和被访问代码段特权级可以不同
- R
  - 0 代码段只能执行，不能读
  - 1 代码段可以执行，也可以读

根据描述子的内容和定义，将保护模式下的寻址过程描述为:



程序如何访问描述子? (描述子的地址在哪里?)

程序如何访问描述子? (描述子的地址在哪里?)

系统提供三个寄存器存放描述子表的基地址，称为描述子表基地址寄存器，分别为:

局部描述子表基地址寄存器

全局描述子表基地址寄存器

中断描述子表基地址寄存器

- 选择子
  - 指明使用该选择子的任务的特权级
  - 指明所要访问的描述子在描述子表中的偏移量
  - 指明访问全局描述子表还是访问局部描述子

- **RPL**：称为请求特权级，标明使用该选择子的任务的特权级

当前运行的任务的特权级称为当前特权级CPL。

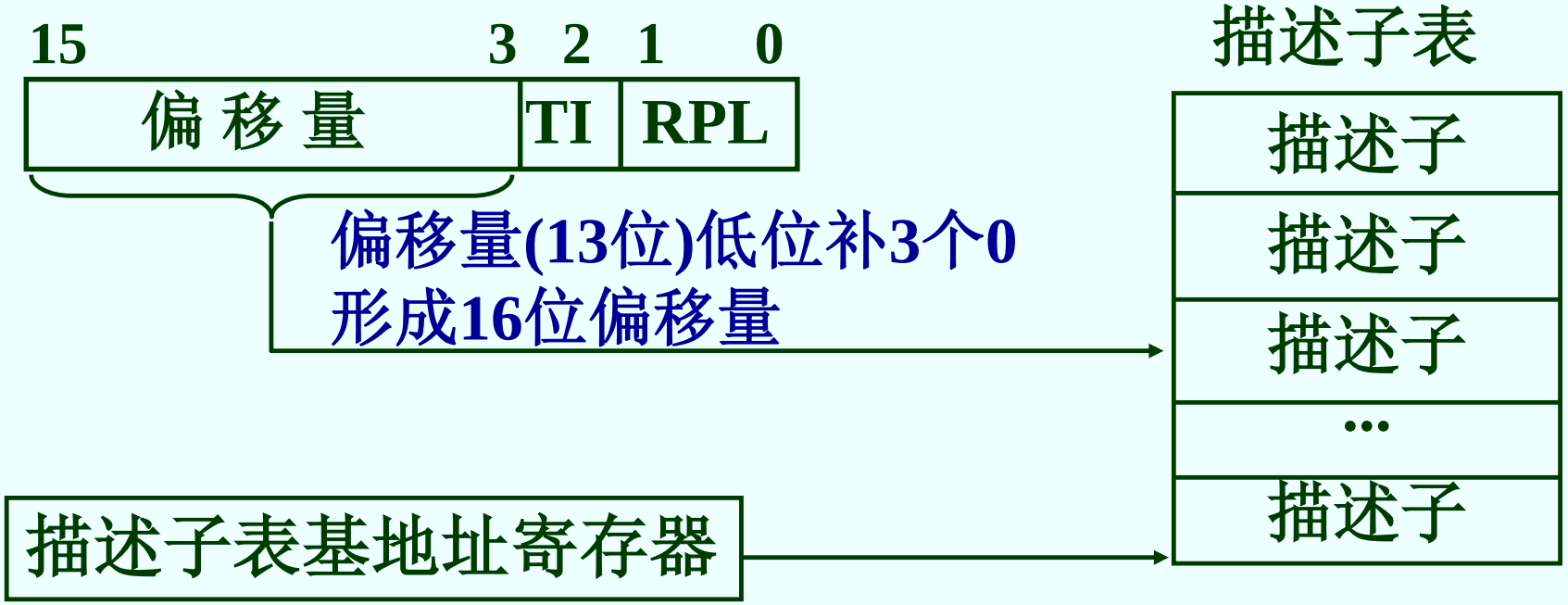
一般有：RPL= CPL

- **TI: 区分访问全局描述子还是局部描述子:**

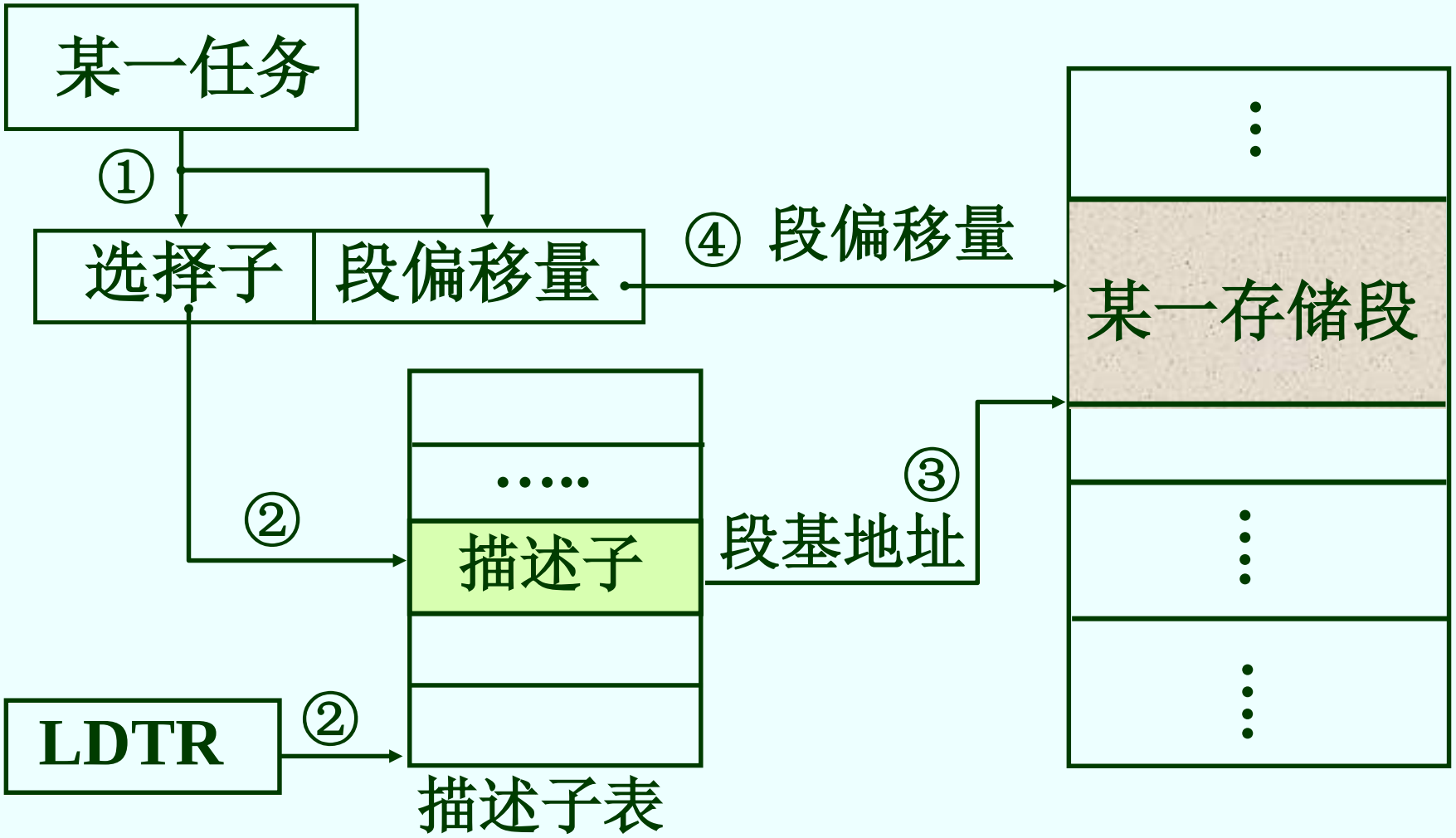
$$TI=\begin{cases}0 & \text{访问全局描述子}\\ 1 & \text{访问局部描述子}\end{cases}$$

- **偏移量D_{15}～D_3:**

  所要访问的描述子在描述子表中的偏移量(以描述子表基地址寄存器的内容为基地址)。



描述子和选择子的引出，可将保护模式下存储段的访问过程描述如下：

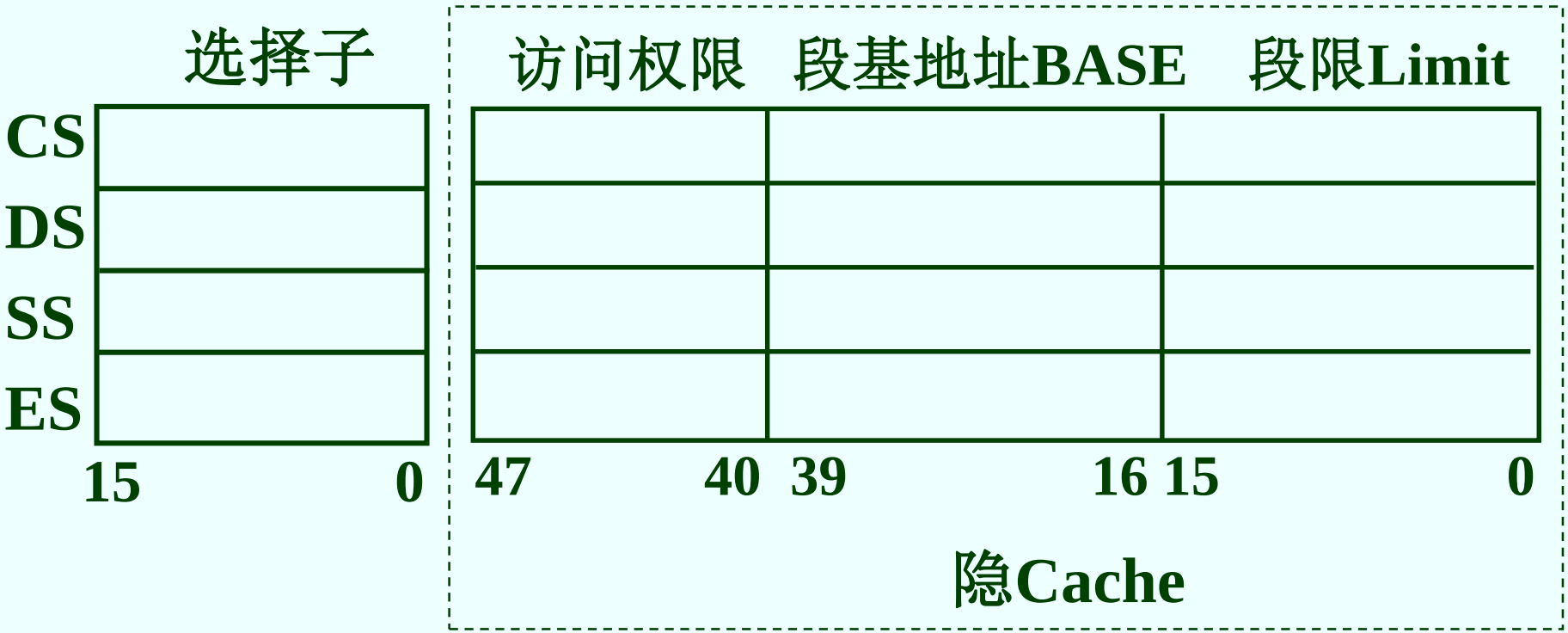


- 虚存空间的计算
  - 可以访问的描述子的数量为2^{13}=8K (个描述子)
  - TI位区分访问全局描述子还是局部描述子, 因此可以访问的描述子的总数为: 2×8K=16K (=2^{14}个)
  - 一个描述子对应一个存储段, 段的最大空间64K, 因此可访问的最大存储空间(虚地址空间)为:

$$16K \times 64K = 1000M$$

选择子的高13位, 作为访问描述子表的偏移量

- 加快访问速度



隐Cache的内容随着段寄存器的修改而被重新装入，该装入操作对程序员透明。

- 关于“数据/代码段描述子”寻址过程例

假设一个32位的虚地址：

选择子	偏移量
005E	0100

选择子 005E=0000000001011 1 10

低位补3个0，为0058H，作为访问LDT的偏移量

RPL=2

TI=1，访问局部描述子

为什么低位补3个0?

每个描述子为8个字节，意味着选择子中的偏移量每增减一个单位，应指向另一个描述子(偏移8个字节)，因此偏移量的$D_2D_1D_0$保持为0，增减一个单位均在D_3上进行，以保证偏移8字节。

假设 LDTR=100000H

第一步: 将描述子表基地址LDTR+选择子偏移量
=100000H+0058H=100058H

第二步: 物理地址100058H访问并得到相应的描述子，检查对该描述子访问的合法性(比较CPL和DPL)，假设DPL=3, 则CPL=RPL=2≤DPL(数值上), 访问合法

第三步: 由描述子中的访问权字段(TYPE)检查本次访问的访问权限, 假设通过检查, 将虚地址中的偏移量(即0100H)与描述子中的段限Limit进行比较，以确定访问是否越界，假设描述子中给出的段基地址位046000H，Limit=2000H，有偏移量0100≤段限2000H，未越界。

第四步: 形成物理地址046000+0100=046100H，以此访问存储单元的物理地址，得到所需要的数据。

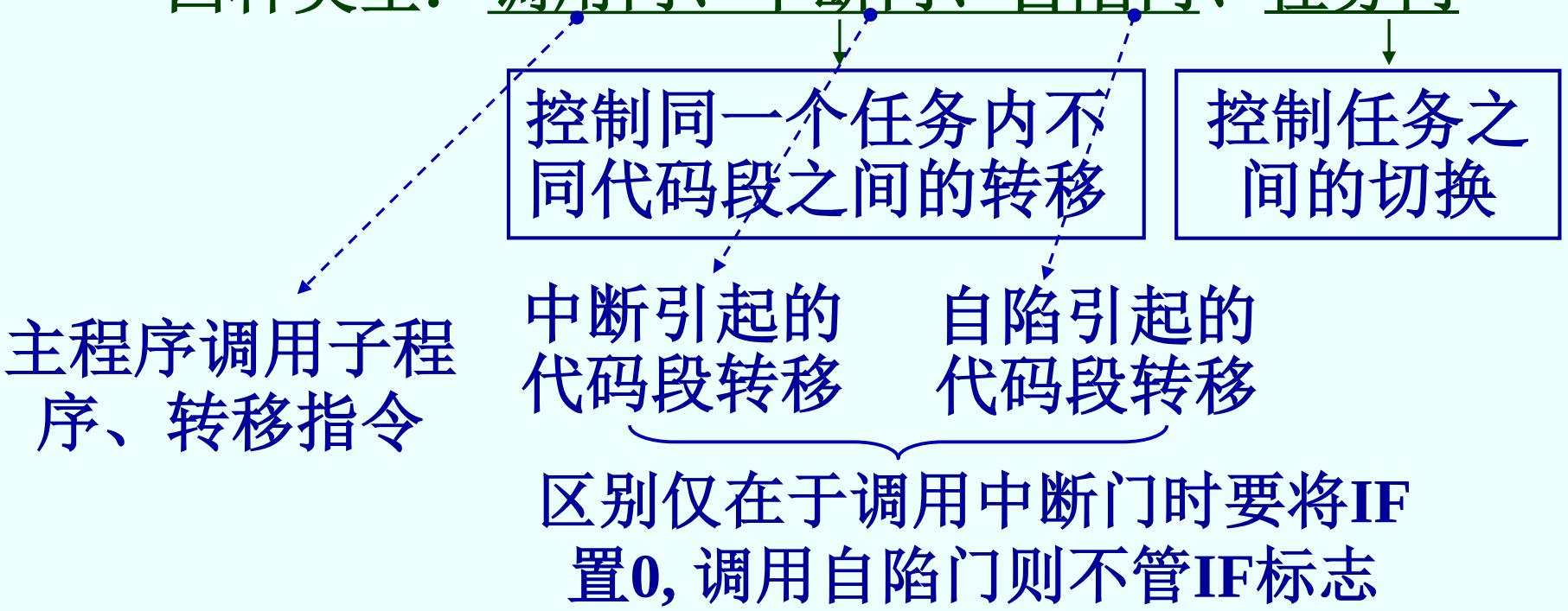
(2) 系统描述子之一: 门描述子

- 用途

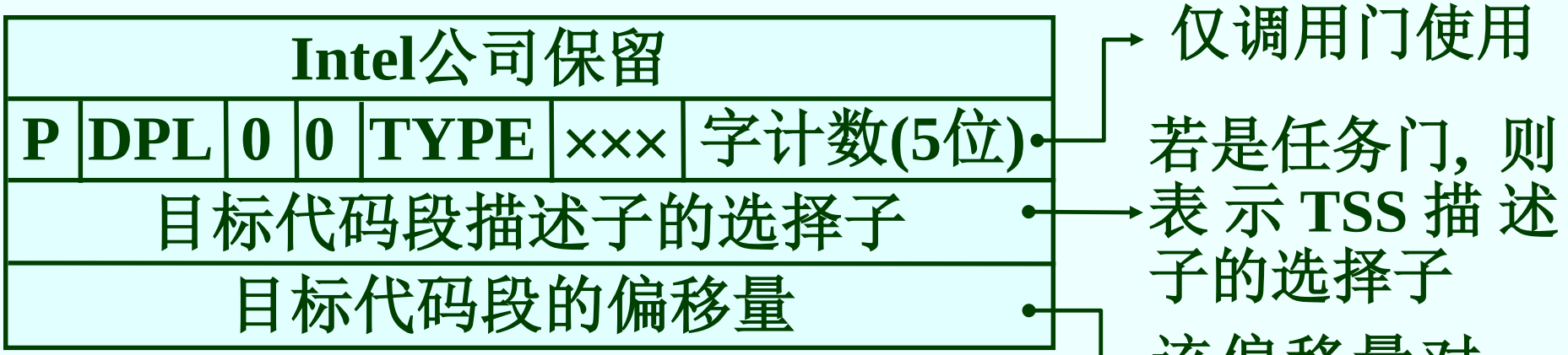
  用于代码段之间的转移控制和保护，以及任务之间的切换: (不用于描述某个存储段的属性)

- 类别及格式

  四种类型: 调用门、中断门、自陷门、任务门



门描述子的格式



- P
  - 0 该描述子内容无效
  - 1 该描述子内容有效
- DPL: 与数据代码段中的DPL相同
- TYPE
  - 4: 调用门
  - 5: 任务门
  - 6: 中断门
  - 7: 自陷门

对任务门

Intel公司保留						
P	DPL	0	0	TYPE	×××	×××××
TSS描述子的选择子						
无效						

任务门与任务状态段描述子
协同控制任务之间的切换

实现代码段转移调用的过程:

- 通过中断门/自陷门实现代码段转移调用的过程：

中断类型码×8(作为访问IDT表的选择子)，结合IDTR → 访问中断描述子表IDT得到中断门/自陷门 → 由中断门或自陷门 → 得到服务程序代码段描述子的选择子以及代码段的偏移量 → 由该选择子 → 访问服务程序代码段描述子

由代码段描述子中提供的基地址和中断/自陷门中的代码段偏移量 → 得到 → 服务程序入口地址

Intel公司保留						
P	DPL	0	0	TYPE	×××	字计数
目标代码段描述子的选择子						
目标代码段的偏移量						

Intel 公司保留					
P	DPL	S	TYPE	A	$BASE_{23\sim16}$
$BASE_{15\sim0}$					
Limit (段限)					

- 通过调用门实现代码段调用的过程

程序中给出调用门描述子的选择子结合LDTR或GDTR → 访问调用门描述子 →

→ 得到目标代码段描述子的选择子以及代码段的偏移量 → 由该选择子访问得到目标代码段描述子 →

→ 由代码段描述子中提供的基地址和调用门中的代码段偏移量 → 转移到目标代码段

说明:

- 说明:

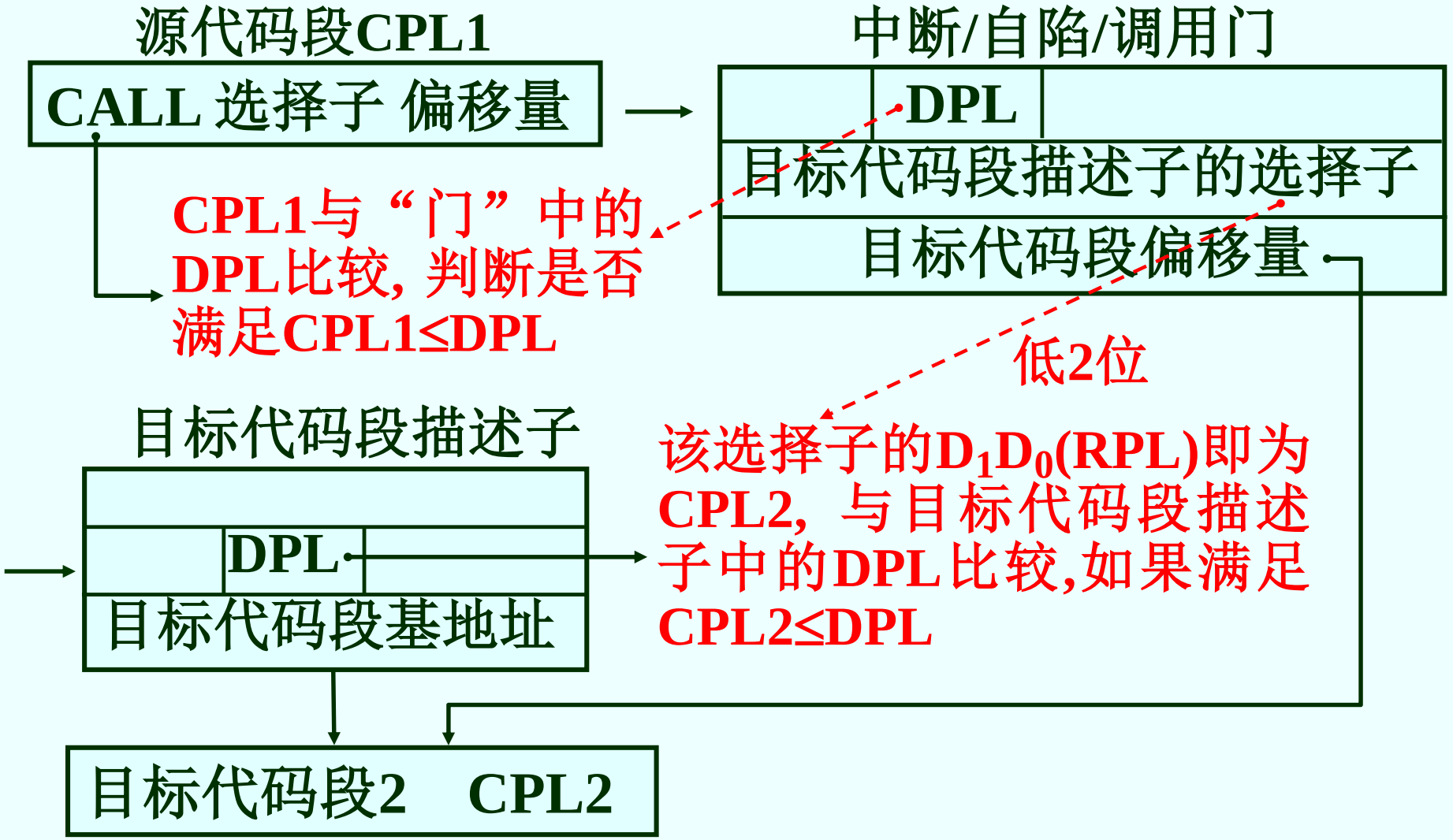
[1] 流程中的地址转换过程, 只要访问描述子, 都需要作的保护性检查，比如仅当CPL≤DPL时转换过程(调用过程)才能继续往下进行。

[2] 为什么需要两次访问描述子(第一次是访问门描述子, 第二次是访问目标代码段描述子)

- 保护模式下，同一任务的不同代码段，也可有不同特权级，意味着主调代码段与被调代码段的特权级可能不同，因此需要指明目标代码段特权级，并由此实现这种同一任务特权级的改变。

  为实现特权级的改变，通过 “门”这样一个描述子中提供目标代码段描述子的选择子，该选择子的低2位(RPL)指明目标代码段的特权级。

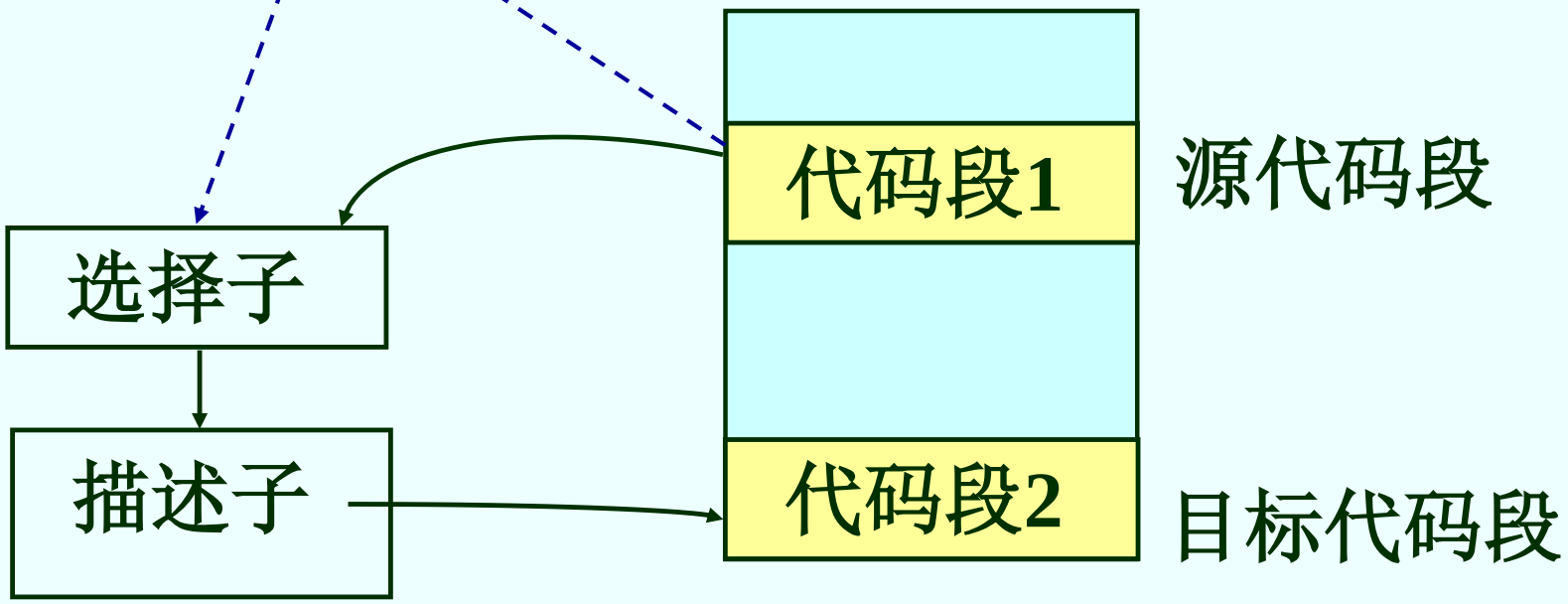
假设：源代码段的特权级为CPL1;目标代码段的特权级为CPL2;转换过程如下图所示:



[3] 中断/自陷/调用门的使用场合

- 相同特权级之间转移

  可以使用也可以不使用“门”。不使用“门”意味着在指令中直接引用目标代码段描述子的选择子来访问目标代码段描述子。即:

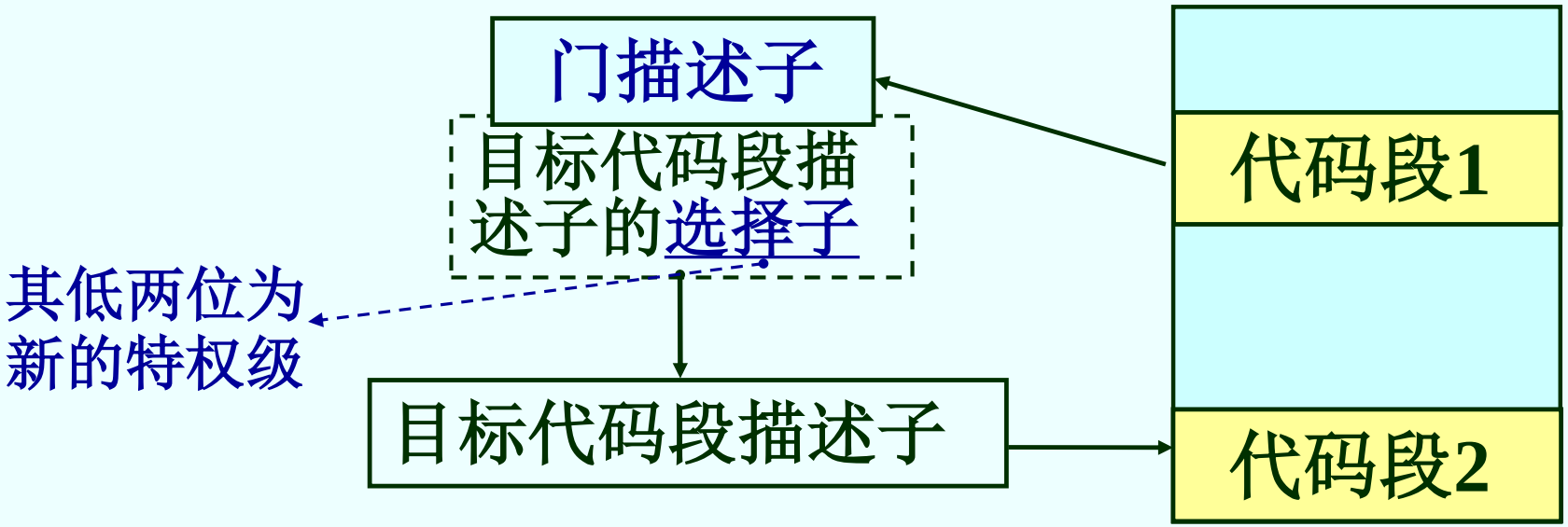


- 向更低特权级转移

  可以使用也可以不使用调用门, 但不管使用与否, 都只能发生在**RET**或**IRET**两种情况。可以直接引用目标代码段描述子的选择子, 此时,该选择子中的**RPL**将成为新的**CPL2**。

- 向更高级转移

  向更高特权级转移, 必须“门”来实现。即:



[4] 任务门的存放位置

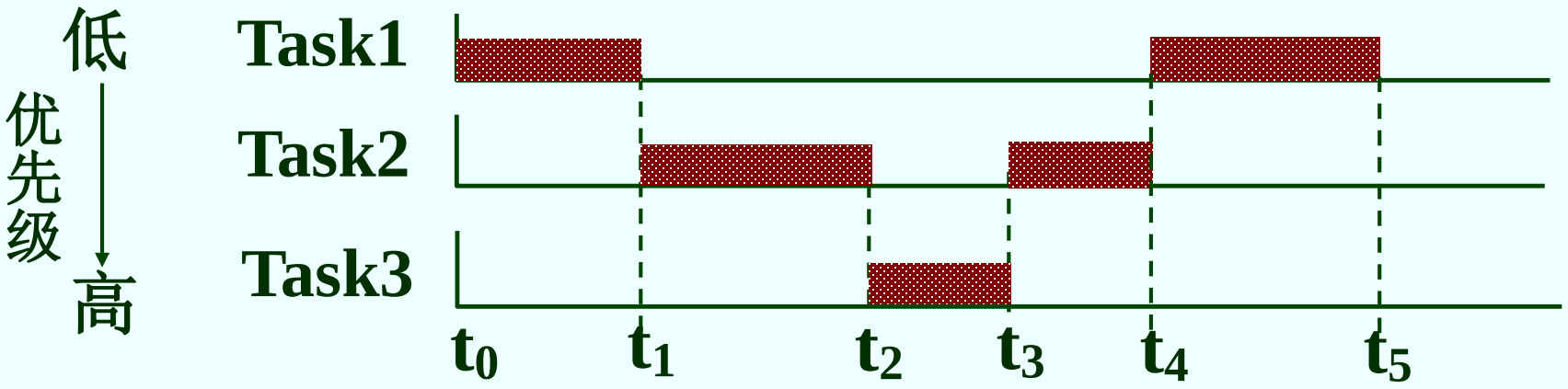
任务门可在LDT、GDT和IDT任何一个表中。

如果将任务门放在IDT表中，即可以通过访问IDT表来访问任务门，则可以达到由于中断而发生任务切换的目的。

任务之间的切换过程，与同一任务不同代码段之间的转移过程是不同的，任务切换还涉及其它描述子的支持，即任务状态段描述子

(3) 系统描述子之二: 任务状态段描述子

- 任务状态段
  - 多个任务运行过程例(按优先级剥夺方式)



在每一个时间点(t_0、t_1、t_2、t_3、t_4、t_5)都存在:

(设置新任务状态) 或者

(保护前一任务状态, 切换到新任务状态) 或者

(恢复前一任务状态)

由此可知, 系统应为每一个任务提供一种保存任务状态的场所, 在Intel80×86系统中称为任务状态段。

任务状态段(Task State Segment –TSS):

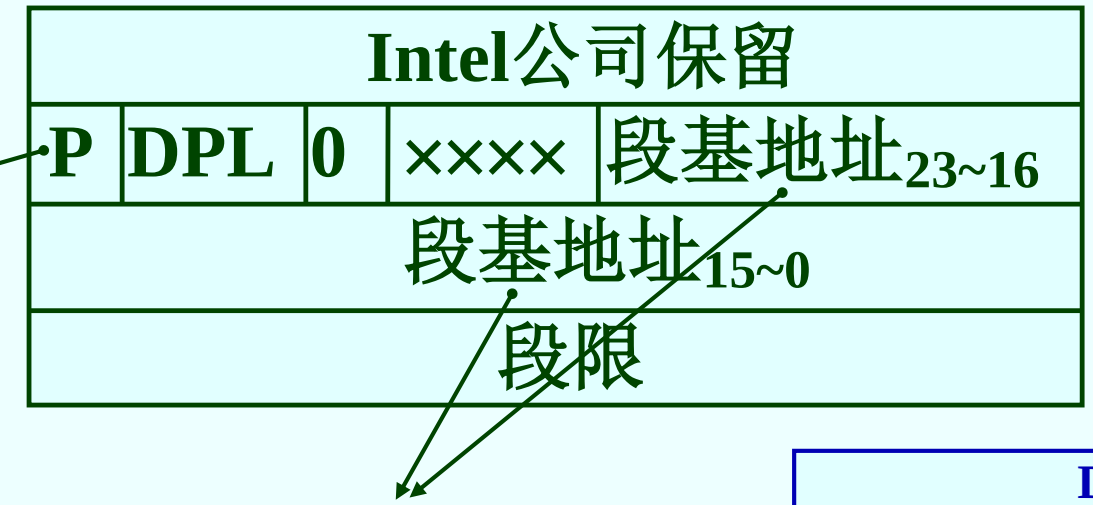
是一个存储段, 用于存放一个任务被切换时的处理器现场, 每个任务都有一个TSS。

显然, TSS中的内容随任务执行的推进不断发生变化。

TSS的内容主要包括:


DS
SS
CS
ES
DI
SI
BP
SP
AX
BX
CX
DX
FR
在各特权层的堆栈指针等
任务状态段
TSS
任务之间的切换必须通过访问TSS来进行，以实现CPU状态(现场)之间的切换
任务状态段，作为一个存储段，需要描述子来刻划其属性

- 任务状态段描述子



段基地址即为TSS的基地址

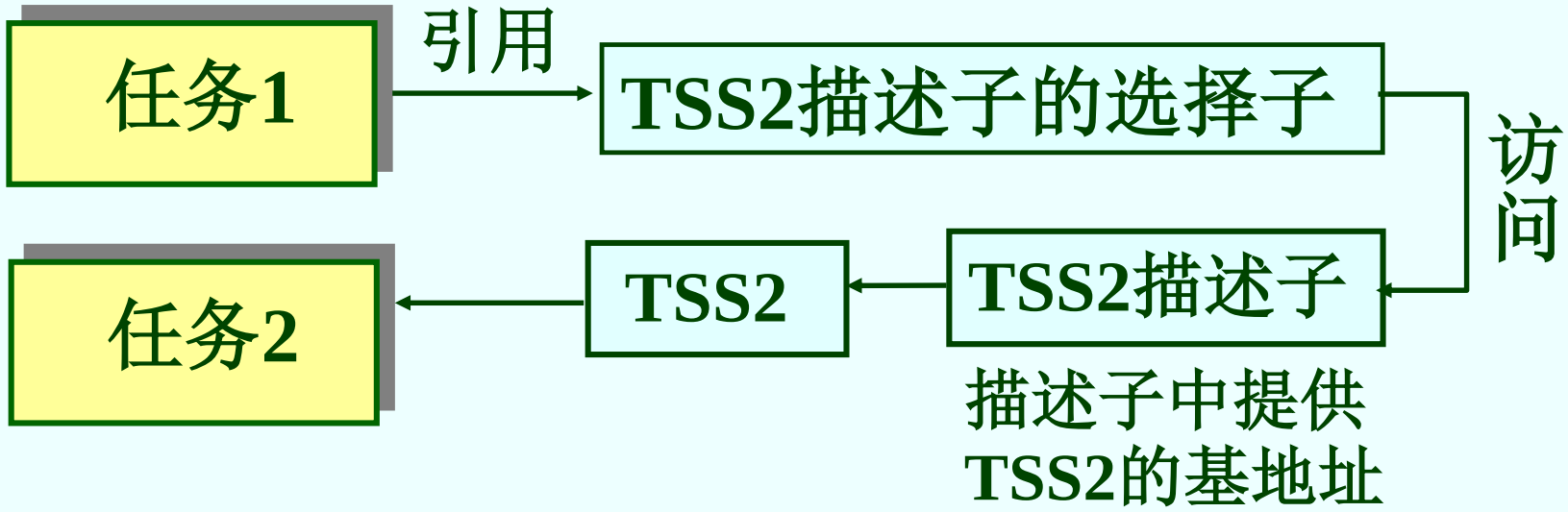
为什么任务状态段描述子不提供任务状态段的偏移量?

- 任务切换的引起
  - 任务切换可由JMP、CALL指令或中断(INT)指令, 异常或外部中断引起。
  - JMP、CALL指令可以直接引用一个任务状态段, 也可以先引用一个GDT或LDT中的任务门, 再由任务门的选择子引用任务状态段描述子, 进而访问任务状态段而实现任务转换。
  - 中断类的指令则必须先从IDT中引用任务门,再由任务门中TSS的选择子引用任务状态段描述子而实现任务切换。IRET指令通过引用IDT中的任务门而返回到原任务中去。

- 任务切换方式

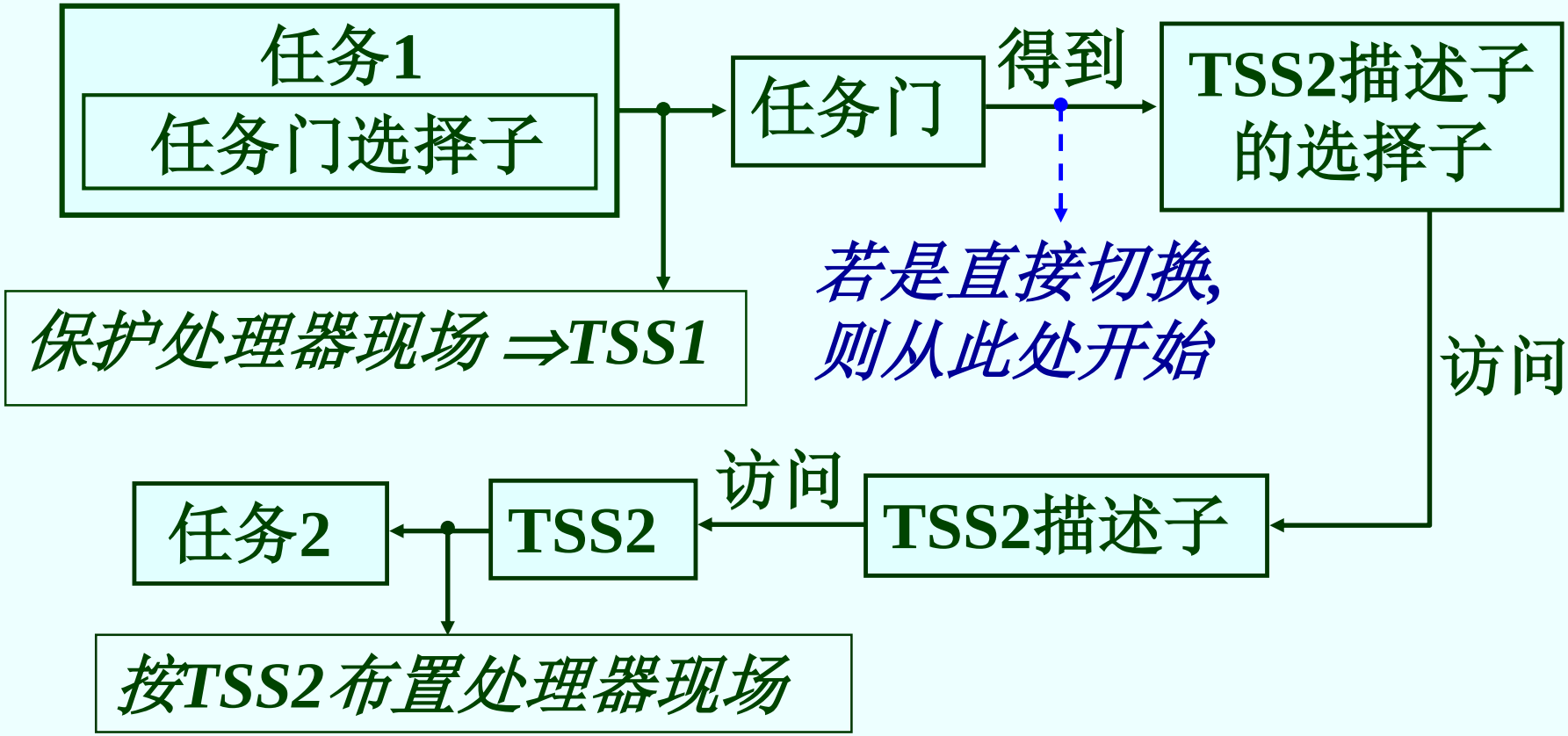
(1) 直接切换

同特权级之间或向更低的特权级切换, 可采用直接切换。直接切换不使用任务门, 而是直接引用任务状态段描述子的选择子来访问TSS, 以实现任务切换。如下图所示:

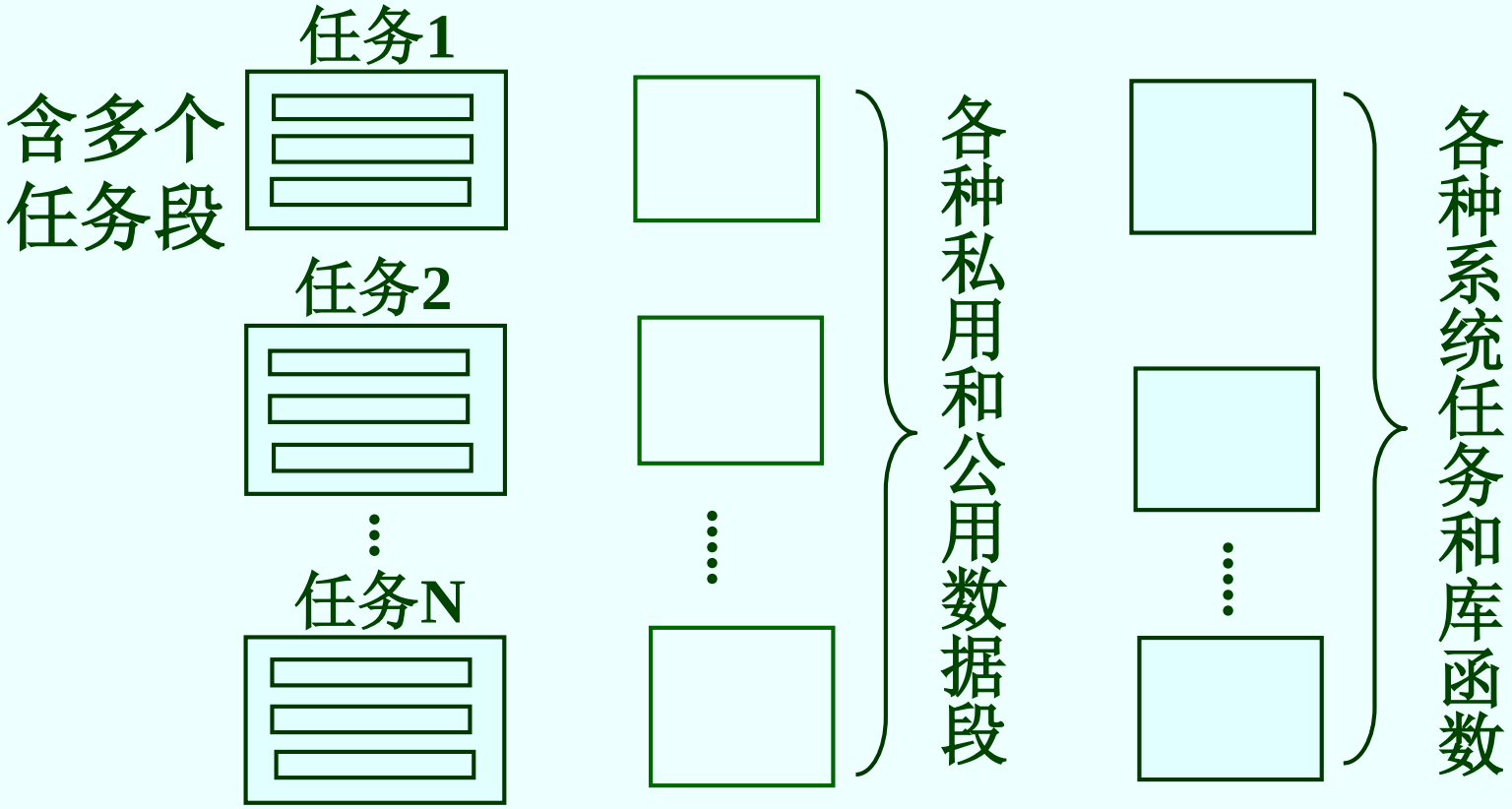


(2) 间接切换

间接切换可以向任何特权级切换。从引用任务门开始，由任务门提供目标任务状态段描述子的选择子。切换过程如下图所示：



小结 虚地址保护：保证复杂多任务系统的可靠的运行



实现：① 系统任务与应用任务之间隔离与保护

② 应用任务与应用任务之间隔离与保护

③ 任务与数据之间的隔离与保护