第二章 微处理器管理模式

张华平 副教授 博士

Email: kevinzhang@bit.edu.cn

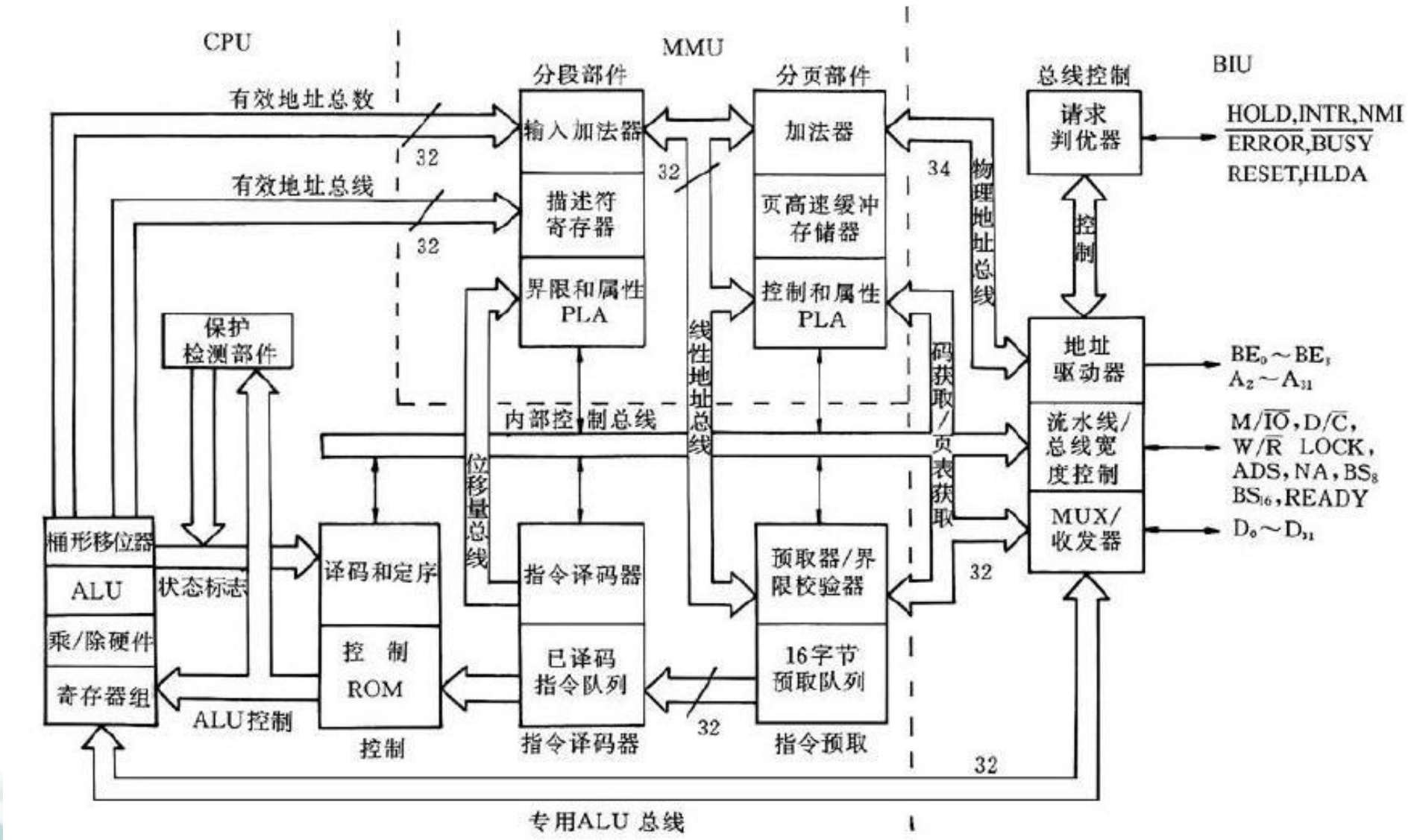
Website: http://www.nlpir.org/

@ICTCLAS张华平博士

大数据搜索挖掘实验室（wSMS@BIT）

- Intel 80x86系列，从4位计算机发展到32位计算机，微处理器内部结构虽然有些变化，但是大同小异。

80386微处理器基本结构



- 1、总线接口部件（Bus Interface Unit，BIU）：与外部环境联系，包括从存储器中预取指令、读写数据，从I/O端口读写数据，以及其他的控制功能。
- 2、中央处理部件（Central Process Unit，CPU）：

① 指令部件：完成指令预取和指令译码。

② 执行部件：执行从已译码指令队列中取出的指令。

- 3、存储管理部件（MMU）：

① 分段部件：实现段式存储管理。

② 分页部件：实现保护模式下的分页模型。

2.2 CPU工作模式

- 从80386开始，Intel的32位CPU具有三种运行模式：
  - 实模式
  - 保护模式
  - 虚拟8086模式

实模式（Real mode）

特点：

- 1、兼容8086，寻址机制、中断处理机制均和8086相同。
- 2、只使用低20位地址线，寻址空间为2^{20} = 1MB地址空间。
- 3、采用分段式内存管理，物理地址形成为16位“段首址：偏移”。段最大为2^{16} = 64KB。
- 4、32位CPU加电或者复位后处于实模式。

- 5、不支持硬件上的多任务切换。
- 6、不支持特权级。

- 实模式下，存储器保留两个专用区：
- 1、初始化程序区：FFFF0H~FFFFFH，存放进入ROM引导程序的一条跳转指令。
- 2、中断向量表区：00000H～003FFH，在这1K字节的存储空间中存放256个中断服务程序的入口地址，每个入口地址占4个字节，这与8086的情形相同。

重点

- 特点：
- 1、支持多任务和特权级 。
- 2、支持内存分页机制，提供段式和页式内存管理功能。
- 3、物理寻址空间高达2^{32} = 4GB（80386/80486）或2^{36} = 64GB（Pentium及以上CPU）。

- 4、引入虚拟存储器的概念，以扩充编程者所使用的地址空间，段内偏移地址32位，每个段最大2^{32}B=4GB，每个程序最多可以使用16K个段，理论上的虚拟地址空间为4GB×16K=64TB。
- 5、提供了一系列的保护机制，如任务地址空间的隔离，设置特权级，设置特权指令，进行访问权限(如只读、只执行)及段限检查等。

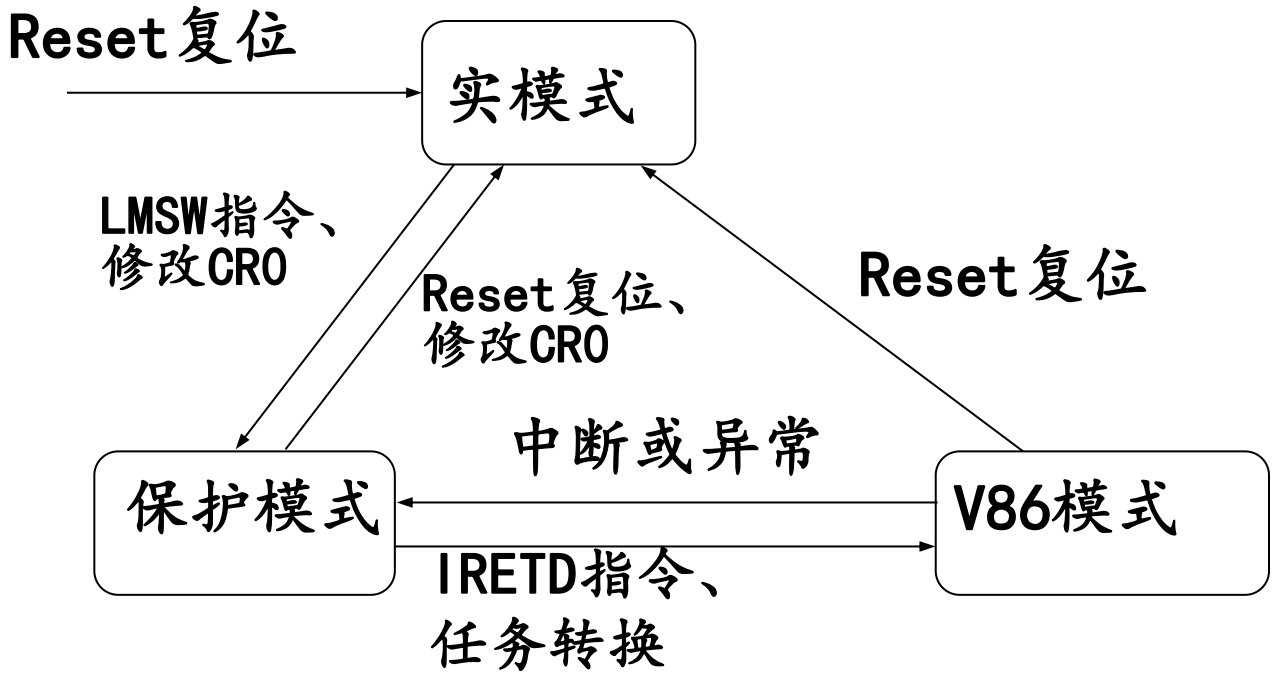
- 特点：

1、兼容8086程序，又称“V86模式”。

2、V86模式以任务的形式在保护模式上执行。操作系统中有专门的V86管理程序。

3、采用模拟的方法实现特权指令。

CPU的3种运行模式以及切换



CPU的3种运行模式及其切换

64位CPU的工作模式

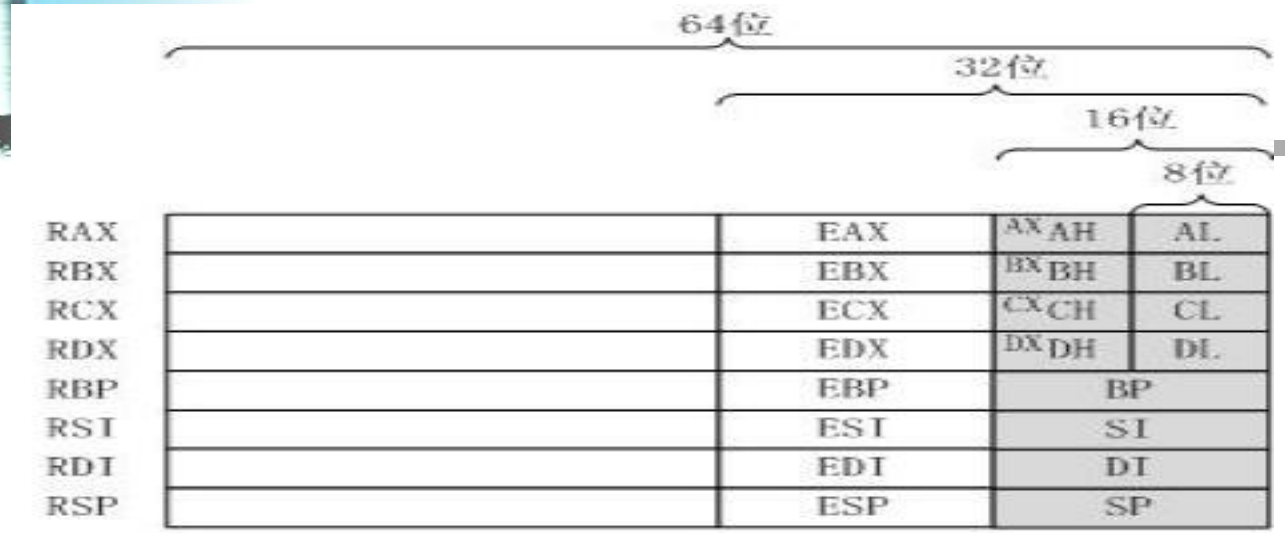
- Intel的EM64T技术中包含：
- 1、IA-32
- 传统的模式（Legacy Mode），本质上的32位x86时代的IA-32模式。
- 2、IA-32e
  - 兼容模式

    允许64位操作系统运行基于32位和16位代码的程序。
  - 64位模式

    EM64T技术中最为高效的模式。要求纯64位环境的支持，包括64位操作系统和64位应用程序。

对于编程人员来说，微处理器内部的寄存器可以分为两大类：程序可见的寄存器和程序不可见的寄存器。

本课程主要介绍Intel 8086~Core2微处理器内部寄存器的功能和用途。



↗ Intel 8086~Core2（包括64位扩展）的程序设计模型

Intel 8086~Core2的程序设计模型

1. 8086，8088和80286使用16位寄存器模型。

其中AX、BX、CX和DX包含两个8位寄存器。

2. 80386~Core2使用6个16位的寄存器（CS，DS，SS，ES，FS和GS）和其他32位的寄存器。
3. Pentium 4和Core2使用16位的段寄存器以及其他64位寄存器，并且在64位扩展允许的情况下，新增R8~R15寄存器。

1. 8位寄存器

16位寄存器AX，BX，CX和DX包含两个8位寄存器，可以通过命名访问，赋值时只影响部分值。

AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

例如：假定初始AX = 2016H，则AH=20H，AL=16H。

若执行指令MOV AL，0

执行指令后AX= 2000H

2. 16位寄存器

AX，BX，CX，DX，BP，SI，DI，SP

IP，FLAGS

CS，DS，ES，SS，FS，GS

注意：段寄存器FS和GS只用于80386以上微处理器。

3. 32位寄存器

EAX，EBX，ECX，EDX，ESI，EDI，EBP，ESP

EIP，EFLAGS

注意：EAX，EBX，ECX和EDX的低16位分别对应AX，BX，CX和DX。32位寄存器只用于80386以上微处理器。

4. 64位寄存器

RAX，RBX，RCX，RDX，SRI，RDI，RBP，RSP

RIP，RFLAGS

R8，R9，R10，R11，R12，R13，R14，R15

- 注意：R8~R15只用于64位扩展允许情况下，可以按照字节（8位）、字（16位）、双字（32位）或者四字（64位）方式寻址。

表3-2 R8~R15寄存器访问控制字

寄存器大小	控制字	访问位数	示例
8位	B	7~0	MOV R9B, R10B
16位	W	15~0	MOV R10W, AX
32位	D	31~0	MOV R14D, R15D
64位	——	63~0	MOV R13, R12

- 注意：8位部分是寄存器最右边的8位。
- 第8位¯第15位不能按照一个字节直接寻址。

表 3- 1 通用寄存器功能

寄存器	常用功能	64位	32位	16位	8位
RAX	累加器，乘法、除法运算等指令的专用寄存器。	RAX	EAX	AX	AH，AL
RBX	保存数据，可用作基址寄存器。	RBX	EBX	BX	BH，BL
RCX	保存数据，计数值，80386以上CPU也可用于访问存储器的偏移地址。	RCX	ECX	CX	CH，CL

表 3- 1 通用寄存器功能（续）

RDX	保存数据，乘法、除法运算指令的专用寄存器，80386以上CPU也可用于访问存储器的偏移地址。	RDX	EDX	DX	DH，DL
RBP	保存访问存储单元时的偏移地址。	RBP	EBP	BP	无
RDI	用于寻址串指令的目的操作数。	RDI	EDI	DI	无
RSI	用于寻址串指令的源的操作数。	RSI	ESI	SI	无

1. RIP（指令指针）

指令指针，指向程序的下一条指令。当微处理器为8086/8088，80286或者工作在实模式下时，这个寄存器取16位IP；80386以及更高型号的微处理，工作于保护模式下时这个寄存器取32位EIP。

2. RSP（堆栈指针）

堆栈指针，指向栈顶满单元。这个寄存器作为16位寄存器时使用SP；作为32位寄存器时，使用ESP。

- RFlags：用于指示微处理器的状态并控制它的操作。向上兼容，8086~80286使用16位Flags；80386以及以上32位微处理器使用EFlags；64位微处理器使用RFlags。

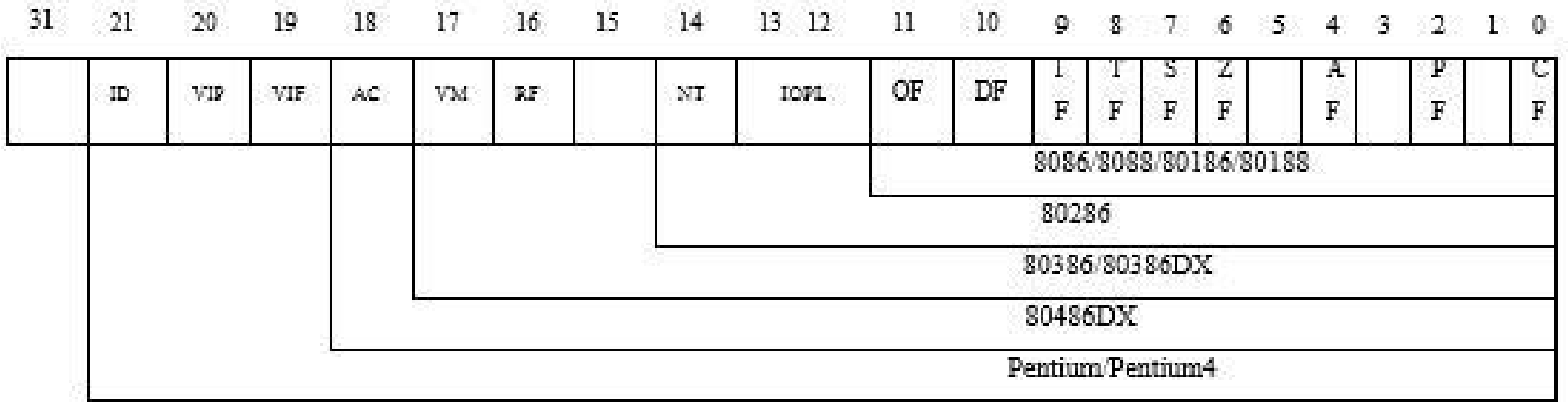


图 3-3 Intel 80X86~Pentium 全系列的微处理器 EFlag 和 Flag 寄存器

标志寄存器中有效位含义

- D0： 进位标志CF（Carry Flag）：当结果的最高位（字节操作时的第7位或字操作时的第15位）产生一个进位或借位，CF＝1，否则CF＝0。在移位或循环移位指令中，会把操作数的最高位（左移时）或最低位（右移时）移入CF中。
- D2：奇偶标志PF（Parity Flags）：Intel微处理器中采用奇校验，当执行结果的低8位中二进制形式1的个数为奇数时，PF为0；否则为1。

- D4：辅助进位标志AF（Auxiliary Carry Flag）：在字节操作时若低半字节（一个字节的低4位）向高半字节有进位或借位；在字操作时若低位字节向高位字节有进位或借位，则AF＝1，否则AF＝0。这个标志用于十进制算术运算指令中，即通过二进制数运算调整为十进制数表示的结果。
- D6：零标志（Zero Flag）：当运算结果为零时，ZF为1；否则为0。

- D7：符号标志SF（Sign Flag）：它与运算结果的最高位相同。对字节操作（8位运算）来说，是结果的第7位；对字操作（16位运算）来说，是结果的第15位。当SF＝0时，结果为正数或0；当SF＝1时，结果为负数。
- D8：单步标志TF（Trap Enable Flag）：当TF=1时，CPU进入单步方式，在每条指令执行以后产生一个内部中断（单步中断）。当TF＝0时，CPU执行指令后不产生单步中断。

- D9：中断允许标志（Interrupt Enable Flag）：当IF=1时，允许CPU接收外部中断请求，否则屏蔽外部中断请求。
- D10：方向标志DF（Direction Flag）：在字符串操作指令中，当DF=0时，串操作为自动增址；当DF=1时，串操作为自动减址。STD指令置位DF，CLD指令清除DF。
- D11：溢出标志OF（Overflow Flag）：有符号数运算时，当其运算结果超出了表达的范围时，OF=1，否则OF＝0。

※状态标志寄存器举例：

	0101	0100	0011	1001
+	0100	0101	0110	1010
	1001	1001	1010	0011

↗运算后 CF PF OF AF ZF SF

CF=0 PF=1 OF=1

AF=1 ZF=0 SF=1

80386新增的四个标志

- D13 D12：I/O特权级IOPL占2位，取值为0、1、2、3对应四个特权级。只有特权级高于IOPL的程序才能够执行I/O指令，否则会产生异常13，并将任务挂起。
- D14：NT（Nest Task）：嵌套任务位，此位只用于保护模式，如果保护模式下当前的任务嵌套在其他任务中，此位为1，否则为0。

- D15：RF（Resume Flag）：与程序调试有关的一个控制位，在程序调试时使用。
- D17：VM（Virtual 8086 Mode Flag）：V86模式位，=1时，表示当前CPU正工作在V86模式下；=0，表示当前CPU工作在实模式或保护模式下。VM位只能在保护方式中由IRET指令置位（如果当前特权级为0），以及在任意特权级上通过任务切换而置位。

Pentium新增的四个标志

- D18：AC（Alignment Check）：地址对齐检查位，当寻址一个字或者双字时，当地址不在字或者双字的边界上，此时AC=1，否则AC=0。80486以上微处理器支持这个标志。
- D19：VIF（Virtual Interrupt）：虚拟中断标志，与VIP一起使用，在虚拟方式下提供中断允许标志位IF的副本(copy)。只有Pentium~Pentium 4处理器才有。

- D20：VIP（Virtual Interrupt Pending）：虚拟中断挂起标志，为Pentium~Pentium 4处理器提供有关虚拟模式中断的信息，它用于多任务环境下，为操作系统提供虚拟中断标志和中断挂起信息。
- D21：ID（Identification）微处理器标识标志，用来指示Pentium~Pentium 4处理器，支持CPUID指令。

段寄存器

- 16位微处理器4个16位段寄存器： CS，DS，SS，ES
- 32位和64位微处理器6个16位段寄存器：CS，DS，SS，ES，FS，GS

- 段寄存器用来和微处理器中的其他寄存器组合生成存储器地址，其功能在不同模式下不同。

不同模式下的段寄存器含义

在实模式和V86模式下，兼容16位CPU，段寄存器内保存的是20位段首址的高16位，段首址的低4位为0。

在保护模式下，段寄存器中的16位内容是一个段选择符（selector），用于在段描述符表（GDT或LDT）中选择段描述符。每个段描述符含８个字节。通过这个描述符中的段基址再加上偏移地址，就组成了线性地址。

CS：Code Segment，代码段寄存器

- 代码段是存放代码（程序，包括过程）的一段存储区。
- 在实地址方式下，定义了一段64KB存储区的起始地址。
- 在保护方式下，用来检索一个描述符，该描述符用来描述一个代码段的若干特性——起始地址、段限以及访问权等。
- 最大段限在8086及80286中为2^{16}=64KB，而在80386及其以上的32位微处理器中则为2^{32}=4GB。64位模式下，代码段寄存器仍用于平展模式，但用法不同。

DS：Data Segment，数据段寄存器

- 数据段存放供程序使用的数据的一段存储区
- 数据段中的数据按其给定的偏移地址值offset（或称有效地址EA，Effective Address）来访问。
- 数据段的长度规定同代码段。
- 64位平展模式中不用。

SS：Stack Segment，堆栈段寄存器

- 堆栈段是一段用作堆栈的存储区。
- 堆栈段现行的入口地址由堆栈指针RSP（或ESP、SP）决定。
- RBP（或EBP、SP）也可寻址堆栈段中的数据。

ES：Extra Segment，附加段寄存器

- 附加段是一段附加的数据段，通常供串操作类指令用于存放目的串数据。

FS、GS

- 80386以上的微处理器（含80386）有两个新增的附加的段存储区。Windows将这些段寄存器用于内部操作没有说明其使用方法。

80386以后新增的寄存器

- 1、控制寄存器
- 2、调试和测试寄存器
- 3、全局描述符表寄存器
- 4、中断描述符表寄存器
- 5、局部描述符表寄存器
- 6、任务寄存器
- 注：其他更高级的处理器与80386微处理器本质上相同。

80386微处理器的控制寄存器结构

CRx	31	30 ~12	11~5	4	3	2	1	0
CR0	PG	00000000000000000000	0000000	ET	TS	EM	MP	PE
CR1	保留未用							
CR2	页故障线性地址							
CR3	页目录基址		000000000000					

CR0内容机器状态字的含义

- PE（Protection Mode Enable）：保护模式允许标志。=0为实模式，CPU复位（启动）时自动进入实模式；=1为保护模式。可以通过软件设置PE进入或退出保护模式。
- MP（Monitor Coprocessor Extension）：运算协处理器存在位。=1表示系统中有协处理器。

- EM（Emulate Processor Extension）：仿真位。=1时，可以利用7号中断，用软件来仿真协处理器的功能；=0时用硬件控制浮点指令。
- TS（Task Switched）：任务切换标志。=1时表明任务已经切换，在保护模式下，TR的内容改变将自动设置此位为1。
- ET（Extension Type）：协处理器选择标志；80386以后的系统中ET位被置为1表示系统中存在协处理器。
- PG（Paging Enable）：分页标志。=1时，存储器管理单元允许分页。=0时，分页功能被关闭，此时CR2和CR3寄存器无效。

- CR2：页面故障线性地址寄存器（Page Fault Linear Address Register），用于发生页异常时报告出错信息。

- CR3：页目录基址寄存器PDBR（Page Descriptor Base Register），它的高20位用于保存页目录表的起始物理地址的高20位。

调试和测试寄存器

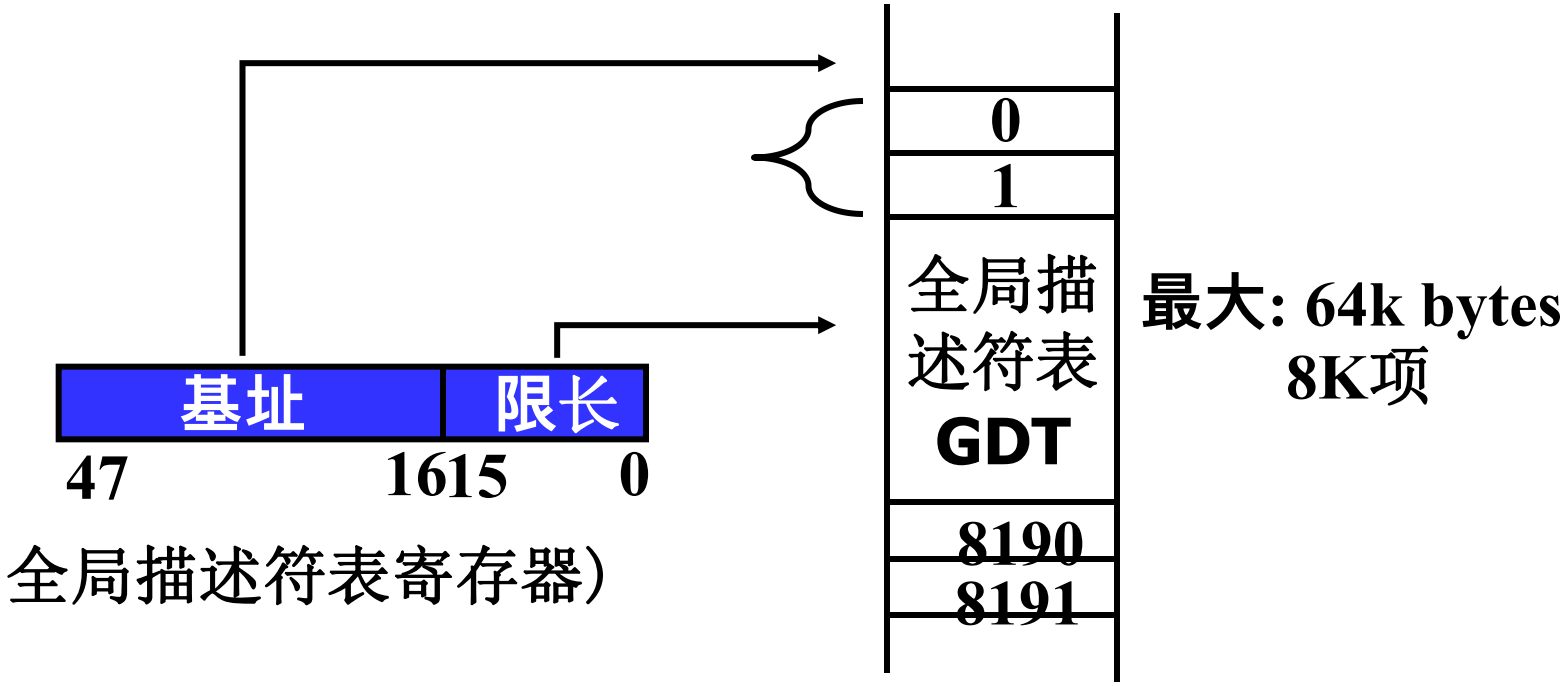
- 80386提供8个32位的调试寄存器和8个32位的测试寄存器。
- 调试寄存器在为调试软件错误提供硬件支持。
- 测试寄存器用户可见的只有TR6、TR7。它们用来测试转换后备缓冲区（Translation Look-aside Buffer, TLB）。

全局描述符表寄存器GDTR

- GDTR （Global Descriptor Table Register）为48位寄存器。

- 高32位基址指出GDT在物理存储器中存放的基地址。
- 低16位限长+1=全局描述符表的字节大小。

GDTR指出GDT在存储器中的位置



全局描述符表GDT

- 全局描述符表是一种共享系统资源，通常包含操作系统所使用的代码段、数据段和堆栈段的描述符，也包含多种特殊描述符，如LDT描述符。该存储区域可以被所有任务访问，在任务切换时，并不切换GDT。

- 每个GDT最多含有8192个描述符（8192×8＝64KB）。GDT可以在内存的任意位置，用LGDT把描述符表的起始位置装入GDTR。

- 已知GDTR=0E003F0003FFH，则全局描述符表的基址是多少？这个全局描述符表有多大，里面有多少个描述符？

解答：

GDT的地址为0E003F00H，

长度为3FFH+1=400H。

可容纳400H/8=80H个段描述符。

中断描述符表寄存器IDTR

IDTR （Interrupt Descriptor Table Register），是48位的寄存器。其最低16位是限长，给出中断描述符表IDT的字节大小（即限长+1）；其高32位是基址，指出IDT在物理存储器中存放的基地址。

中断描述符表IDT

- IDT中保存的是中断门描述符。每个门描述符包含8字节，IDT最多包含256个门描述符，因为CPU最多支持256个中断。
- 思考：IDT最大长度是多少？

保护模式下的中断描述符表的功能，类似于实模式下的中断向量表。但IDT的位置可变，由相应的描述符说明，而实模式下的中断向量表的地址是固定的，必须在物理地址00000H处。

例题3-2

- 已知IDTR=0E003F40007FFH，求IDT的地址和长度。这个IDT中有多少个中断门描述符？

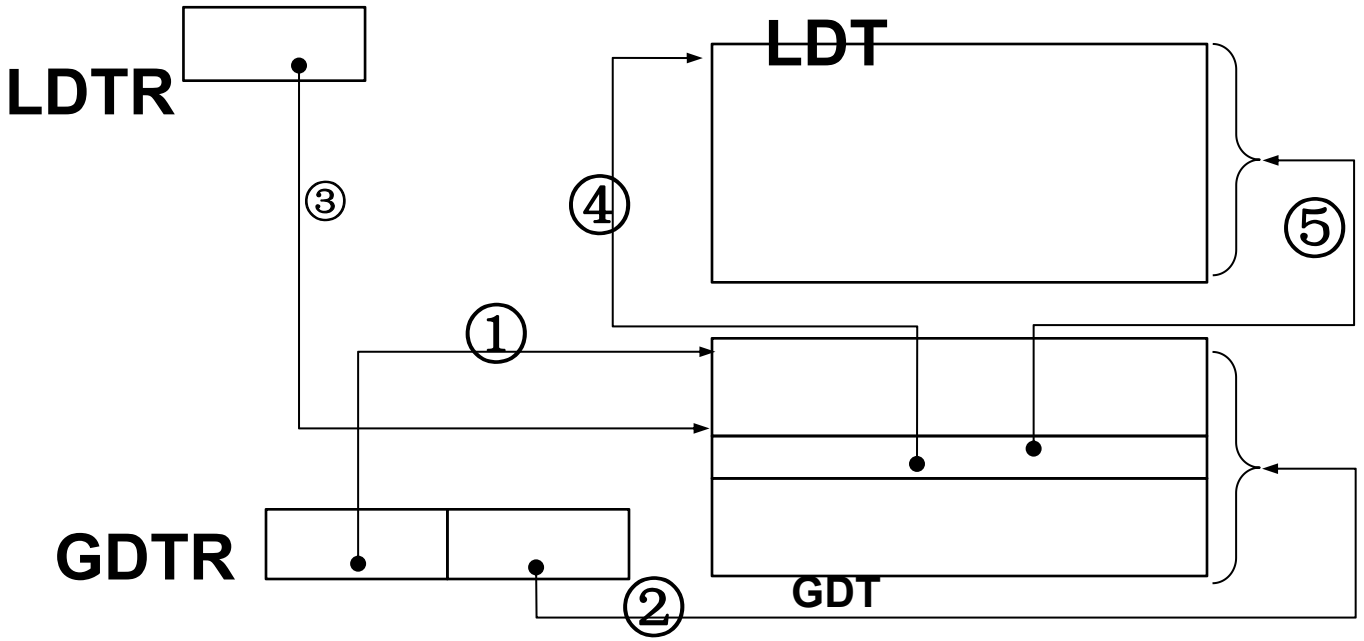
则IDT的地址为0E003F400H，
长度为7FFH+1=800H。
其中可容纳800H/8=100H个中断门描述符。

注意：GDTR和IDTR的值必须在进入保护模式之前装入。
在实模式下，可以执行LGDT和LIDT指令设置GDTR和IDTR。

局部描述符表LDT

- LDT（Local Descriptor Table Register）定义的是某项任务用到的局部存储器地址空间。
- 多任务环境下由于每项任务都有自己的LDT（且每项任务最多只能有一个LDT），因此保护模式下可以有多个LDT。
- LDT由LDTR（局部描述符表寄存器）确定。LDTR为16位的选择符。

LDTR寄存器确定LDT的位置和限长的过程



①和②步由GDTR确定了GDT表在存储器中的位置和限长。LDTR中是一个选择符，它包含了LDT描述符在GDT中的索引。③步是依据LDTR在GDT中取出LDT描述符的过程。在LDT描述符中，包含由LDT的位置和限长，即④和⑤步。

- 假定LDT的基址为00120000H，GDT基址为00100000H。如果装入CS寄存器的选择符为1007H，那么请求特权级是多少？段描述符地址是多少？描述符在GDT还是LDT里面获得？

（CS）=0001000000000111b

最右边两位为1则RPL=3

TI=1表示段描述符在LDT中。

偏移量为高13位乘以8得到相对于表基址的偏移量。

偏移量=0001000000000b × 8=512 × 8=4096=1000H

段描述符地址为00120000H+1000H=00121000H

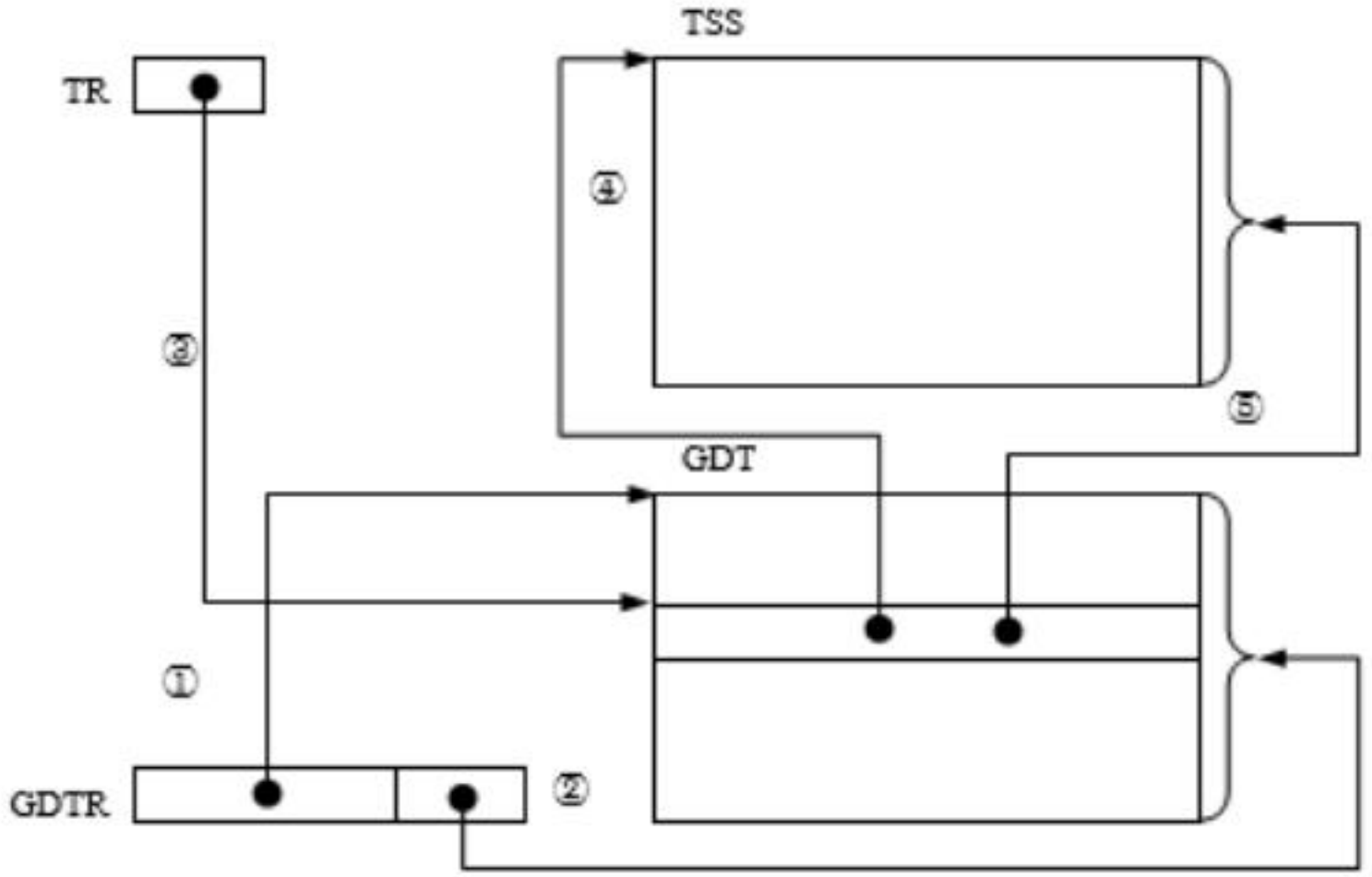
任务寄存器TR

- 任务寄存器TR （Task Register），在保护模式的任务切换机制中使用，TR中内容为16位选择符。
- TR的作用是选中TSS描述符。每一个任务都有一个任务状态段TSS，由TSS描述符描述。从GDT中检索出TSS描述符后，微处理器自动将TSS描述符装入TSS Cache中。
- TR的初值由软件装入，当执行任务切换指令时TR的内容自动修改。

例题 3- 3 假定全局描述符表的基址为00011000H，TR为2108H，问TSS描述符的起始范围是多少？

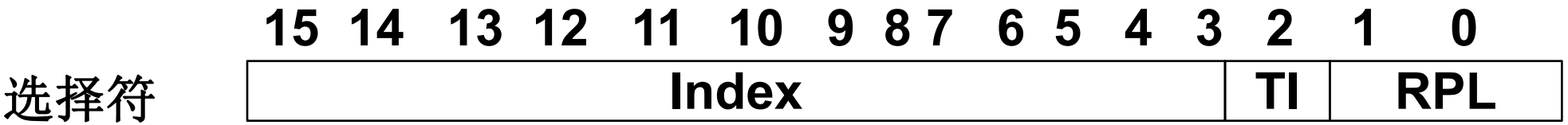
解答：TSS起始地址=00011000H+2108H=00013108H
由于描述符为8字节，故TSS终止位置为
00013108H+7H=0001310FH

由TR取得TSS的过程



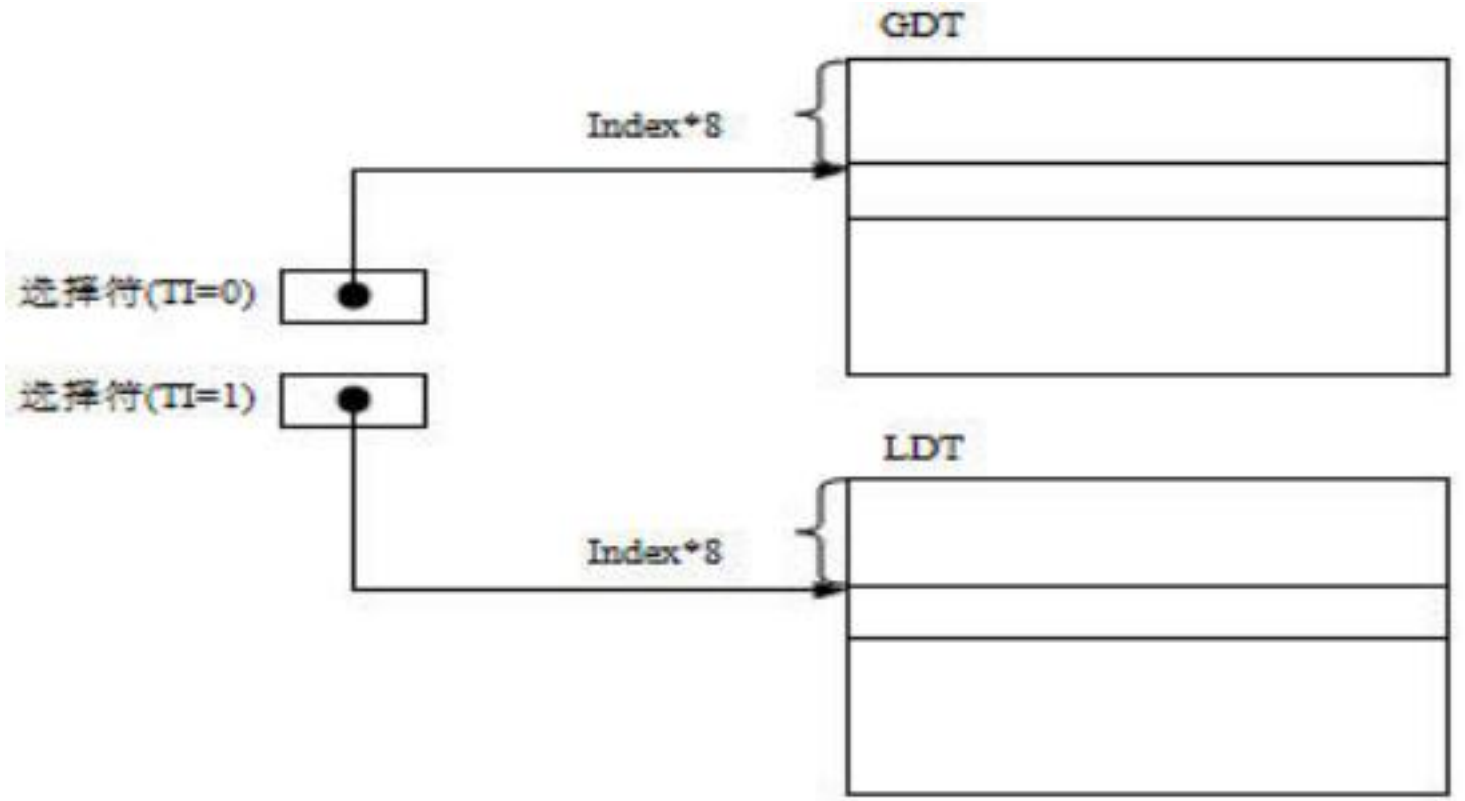
- ①和②步由GDTR确定了GDT表在存储器中的位置和限长。TR是一个选择符，这个选择符中包含了TSS描述符在GDT中的索引。③步依据TR在GDT中取出TSS描述符。在第④和⑤步中，在TSS描述符中取得TSS的基址和限长。

选择符格式



- RPL（Requestor Privilege Level）：　请求特权级，2位二进制数字，范围为0～3。00代表特权级0，01代表特权级1，10代表特权级2，11代表特权级3。
- 请求特权级是将要访问的段的特权级。
- TI（Table Indicator）：表指示符。为0时，从GDT中选择描述符；为1时，从LDT中选择描述符。
- Index：索引。指出要访问描述符在段描述符表中的顺序号，Index占13位。因此，顺序号的范围是0～8191。每个段描述符表（GDT或LDT）中最多有8192=2^{13}个描述符。

由选择符确定一个描述符



已知DS=0023H，问该数据段的请求特权级和索引值。

- 解答
- DS =0000 0000 0010 0011b，
- 故：Index=0 0000 0000 0100b=4，TI=0，RPL=11b=3。
- 因此请求特权级为3，索引值为4。

2.4 内存管理---存储器分段管理

IBM PC机的存储器采用分段管理的方法。存储器采用分段管理后，一个内存单元地址要用段基地址和偏移量两个逻辑地址来描述，表示为　　，其段基址和偏移量的限定、物理地址的形成视CPU工作模式决定。

内存管理---实模式存储器寻址[1]

3种工作模式

实模式：微处理器只可以寻址最低的1M字节。

保护模式：寻址4GB

虚拟86模式：寻址类似于8086

2.4.1 实模式分段管理

对段基址的限定：

只要工作在实模式，段基址必须定位在地址为16的整数倍上，这种段起始边界通常称做节或小段。

对段长的限定：

在实模式下段长不能超过64K。

2.4.1 实模式分段管理

存储器采用分段管理后，其物理地址的计算方法为：

10H×段基址＋偏移量

简便的计算方法：因为段基址和偏移量一般用十六进制数表示，直接在段基址的最低位补以0H，再加上偏移量。

实模式存储器寻址示例

例．某内存单元的地址用十六进制数表示为1234:5678，则其物理地址为

12340H＋5678H＝179B8H。

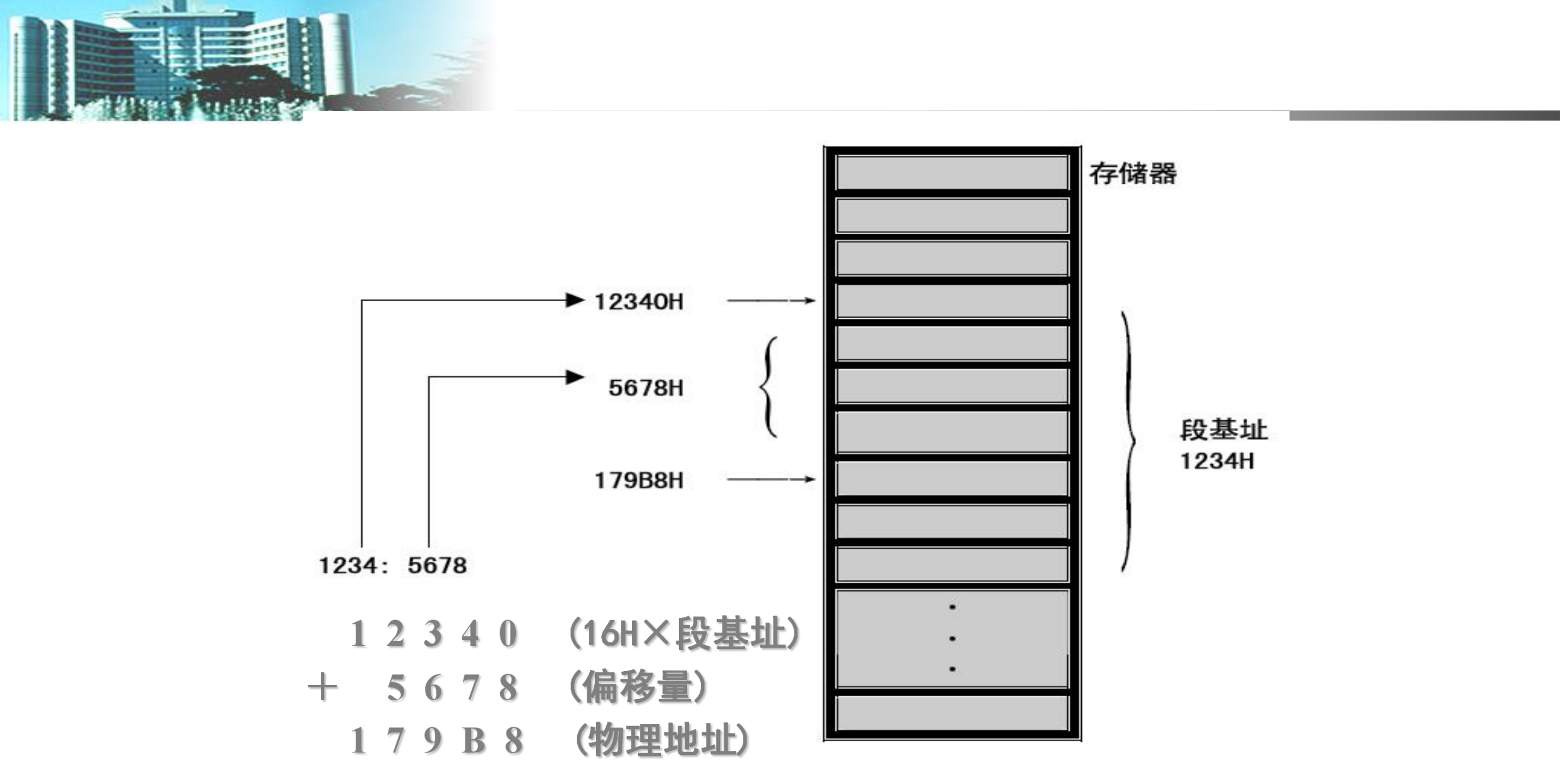

存储器
12340H
5678H
179B8H
段基址
1234H
1234：5678
1 2 3 4 0 （16H×段基址）
＋ 5 6 7 8 （偏移量）
1 7 9 B 8 （物理地址）

图2-5 物理地址的形成

实模式寻址约定规则

程序执行时，其当前段的段基址放在相应的段寄存器中，偏移量视访问内存的操作类型决定，其规律如下页表所示。

操作类型	约定段寄存器	允许指定的段寄存器	偏移量
1. 指令	CS	无	IP
2. 堆栈操作	SS	无	SP
3. 普通变量	DS	ES、SS、CS	EA
4. 字符串指令的源串地址	DS	ES、SS、CS	SI
5. 字符串指令的目标串地	ES	无	DI
6. BP用作基址寄存器	SS	DS、ES、CS	EA

2.4.2 保护模式下分段管理

在保护模式下，其内存管理既可以使用分段机制访问多达4 GB（386/486）或64 GB（Pentium）的内存空间，也可以使用分页机制访问多达16 TB的虚拟存储器。总之，保护模式打破了实模式只允许访问装在内存第一个1 MB之内的程序和数据的限制。

描述符

- 描述符是用以管理64TB虚拟存储地址空间分段的基本元素。一个描述符对应虚拟地址空间中的一个存储段。
- 描述符包括段描述符、系统段描述符、局部描述符、调用门描述符、任务状态段描述符，以及任务门描述符等。

段描述符

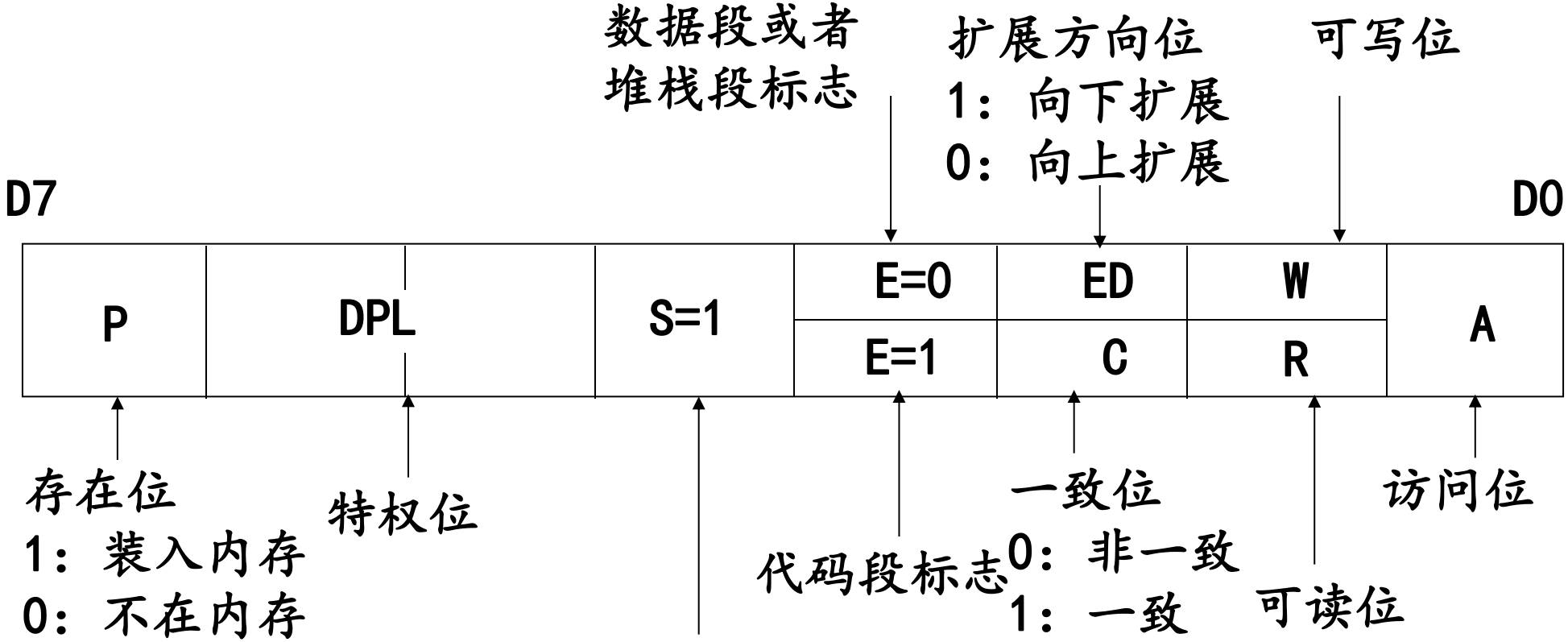
段描述符用于描述代码段、数据段和堆栈段。

段描述符位于GDT或LDT中，占8字节，由以下几个部分组成：段基址（32位）、限长（20位）、访问权限（8位）和属性（4位）。

段描述符

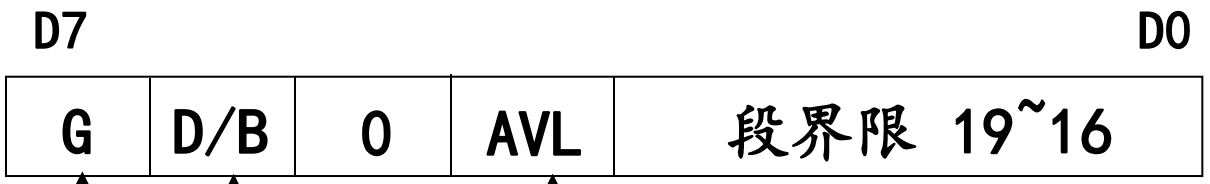
D7				D0	
段界限 7~0					0
段界限 15~8					1
基址 7~0					2
基址 15~8					3
基址 23~16					4
P	DPL		S	TYPE	5
G	D/B	0	AVL	段界限 19~16	6
基址 31~24					7

段描述符第5字节访问权限，
非系统段（代码段、数据段、堆栈段）的格式：



S=1是非系统段：数据段或者代码段
S=0是系统段描述符

段描述符中的第6字节：四位属性



用户/操作系统可用位

D/B位，系统段不用这位

粒度位 ，=1以页为单位，=0以字节为单位

D/B位
- 代码段(D位)
  - D=1 使用32位操作系统和32位寻址方式
  - D=0 使用16位操作系统和16位寻址方式
- 数据段(B位)
  - B=1 堆栈使用ESP寄存器，上限为FFFFFFFFH
  - B=0 堆栈使用SP寄存器，上限为FFFFH

已知Windows XP中各段寄存器及段描述符的值如下，分析CS对应的段描述符属性。

- CS=001B DS=0023 ES=0023 SS=0023 FS=0030 GS=0000
- GDTbase=E003F000 Limit=03FF
- E003F000 00 00 00 00 00 00 00 00-FF FF 00 00 00 9B CF 00
- E003F010 FF FF 00 00 00 93 CF 00-FF FF 00 00 00 FB CF 00
- E003F020 FF FF 00 00 00 F3 CF 00-AB 20 00 20 04 8B 00 80
- E003F030 01 00 00 F0 DF 93 C0 FF-FF 0F 00 00 00 F3 40 00
- E003F040 FF FF 00 04 00 F2 00 00-00 00 00 00 00 00 00 00

选择符	类型	段基址	结束地址	DPL		
0008	Code32	00000000	FFFFFFFF	0	P	RE
0010	Data32	00000000	FFFFFFFF	0	P	RW
001B	Code32	00000000	FFFFFFFF	3	P	RE
0023	Data32	00000000	FFFFFFFF	3	P	RW
0028	TSS32	80042000	000020AB	0	P	B
0030	Data32	FFDFF000	00001FFF	0	P	RW
003B	Data32	00000000	00000FFF	3	P	RW
0043	Data16	00000400	0000FFFF	3	P	RW
0048	Reserved	00000000	00000000	0	NP	

内容	二进制		7 0
FF	11111111B	+0	限长（位7-0）
FF	11111111B	+1	限长（位15-8）
00	00000000B	+2	段基址（位7-0）
00	00000000B	+3	段基址（位15-8）
00	00000000B	+4	段基址（位23-16）
FB	11111011B	+5	P DPL S TYPE A
CF	11001111B	+6	G D 0 AVL 限长（位19-16）
00	00000000B	+7	段基址（位31-24）

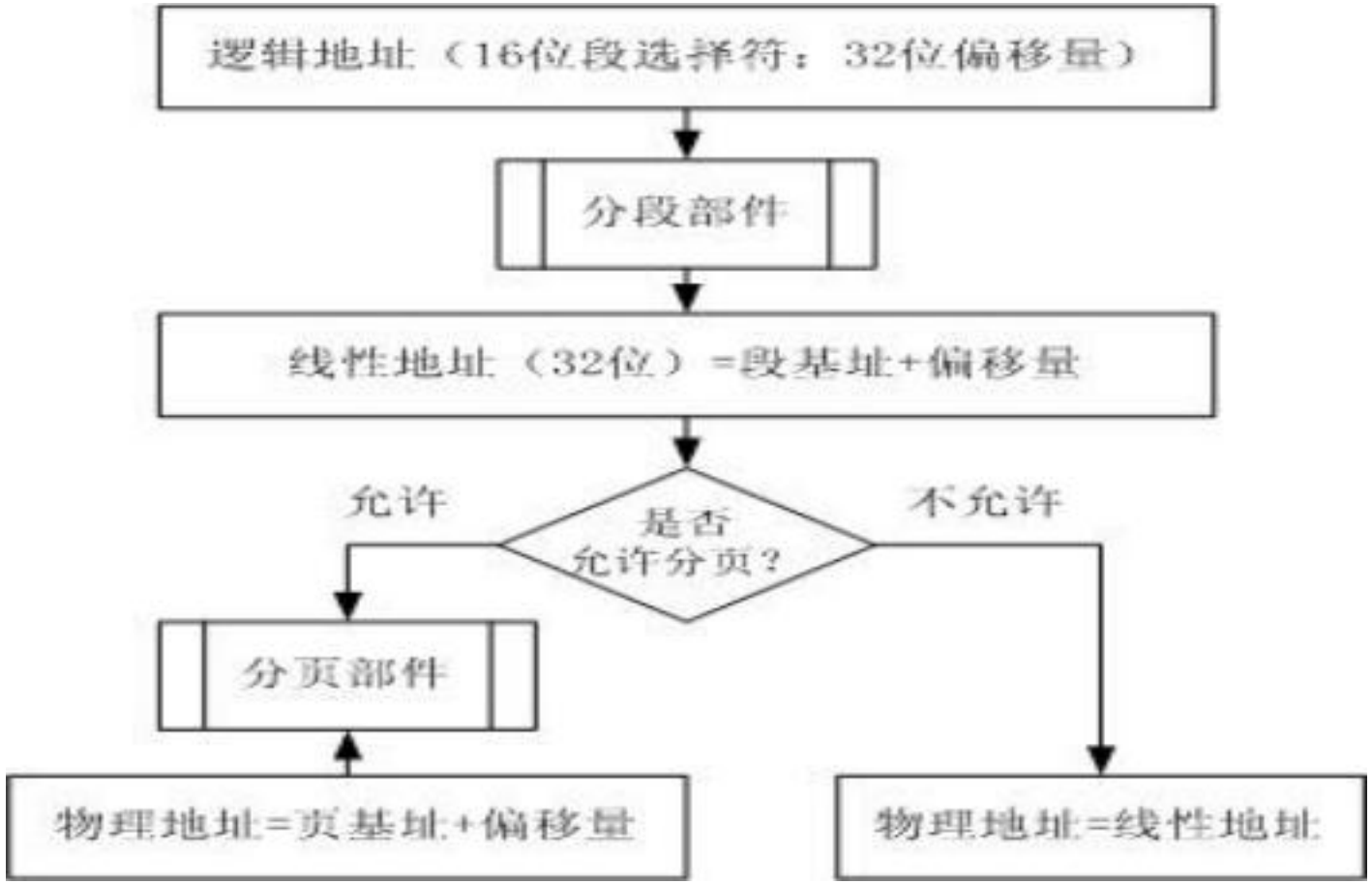
- 分析CS段描述符得：
- 段基址（位31～0）=00000000H
- 限长（位19～0）=FFFFFH
- G=1，限长是以页为单位的，段的大小为（FFFFFH+1）×2^{12}=100000H×2^{12}=2^{32}=4GB。段的基址为00000000H，长度为4GB，其线性地址范围为00000000H～FFFFFFFFH。

- 其他的各个位解释如下：
- D=1，因此这是一个32位的段。
- AVL=0。
- P=1，这个段在内存中。
- DPL=11_2=3_{16}，段的特权级为3。
- S=1，这不是一个系统段，而是一个代码、数据段或堆栈段。
- E=1，这是一个代码段（可执行）。
- C=0，这不是一个一致代码段。因为E=1，这一位是C；若E=0，这一位代表ED。
- R=1，可以对这个段进行读操作。因为E=1，这一位是R；若E=0，这一位代表W。

段描述符高速缓存

为了提高CPU的运行效率在CPU内部设置了段描述高速缓存，这些缓存不可见，自动装入段描述符。段描述符高速缓冲的内容和6个段描述符寄存器索引的描述符当前值保持一致。用户不可见。

32位CPU系统中三者的转换



2.4.3 页式内存管理

- 保护模式下的CPU支持分页机制，并且分页管理是在分段管理机制的基础上工作，它将分段管理机制得到的线性地址转换为物理地址。
- 使用分页机制的好处在于，它可以把每个活动任务当前所必需的少量页面放在内存中，而不必将整个段调入内存，从而提高了内存的使用效率。

分页

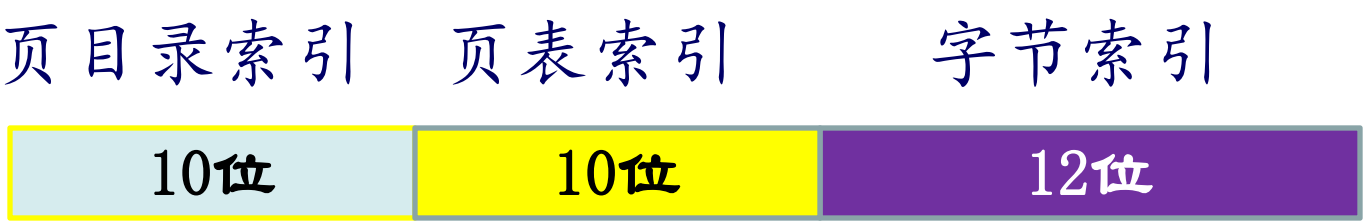
每一个任务都有它自己的线性地址空间。分页管理时所有页的长度固定，页与页之间也没有重叠。

假定页面大小为4KB，则32位CPU将4GB的线性地址空间划分成2^{20}页。固定长度的页可能会产生碎片，会浪费一些内存空间。

分页机制下线性地址转换为物理地址的过程

分页机制就是一种将线性地址的页面映射到物理地址页面的手段，也就是从线性地址到物理地址的转换过程。分页机制中用到了两个表：页表目录表和页表。

32位线性地址被划分为3个部分：



其中第1项是对页目录（Page Directory）的索引，第2项是对页表（Page Tables）的索引，第3项是线性地址在页面内的偏移。

线性地址转换为物理地址的过程

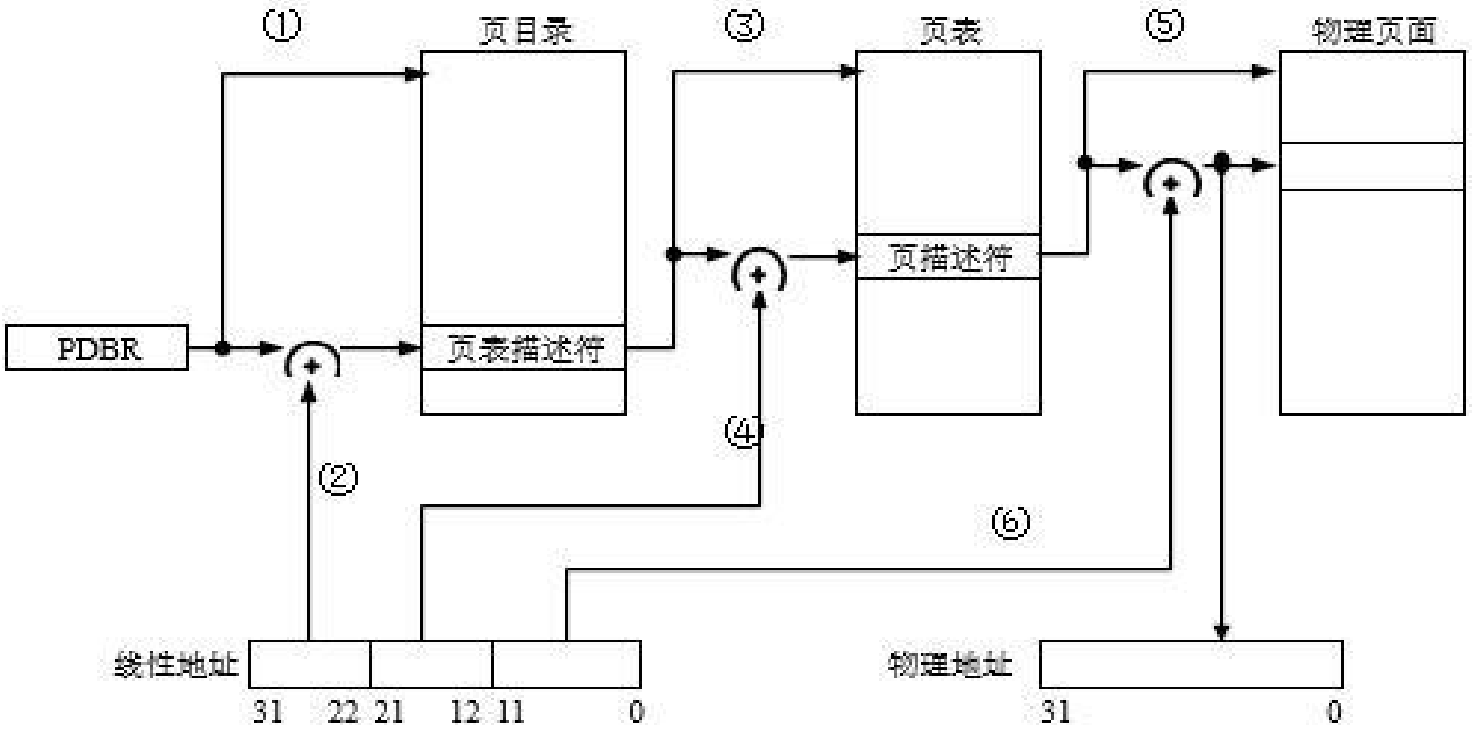
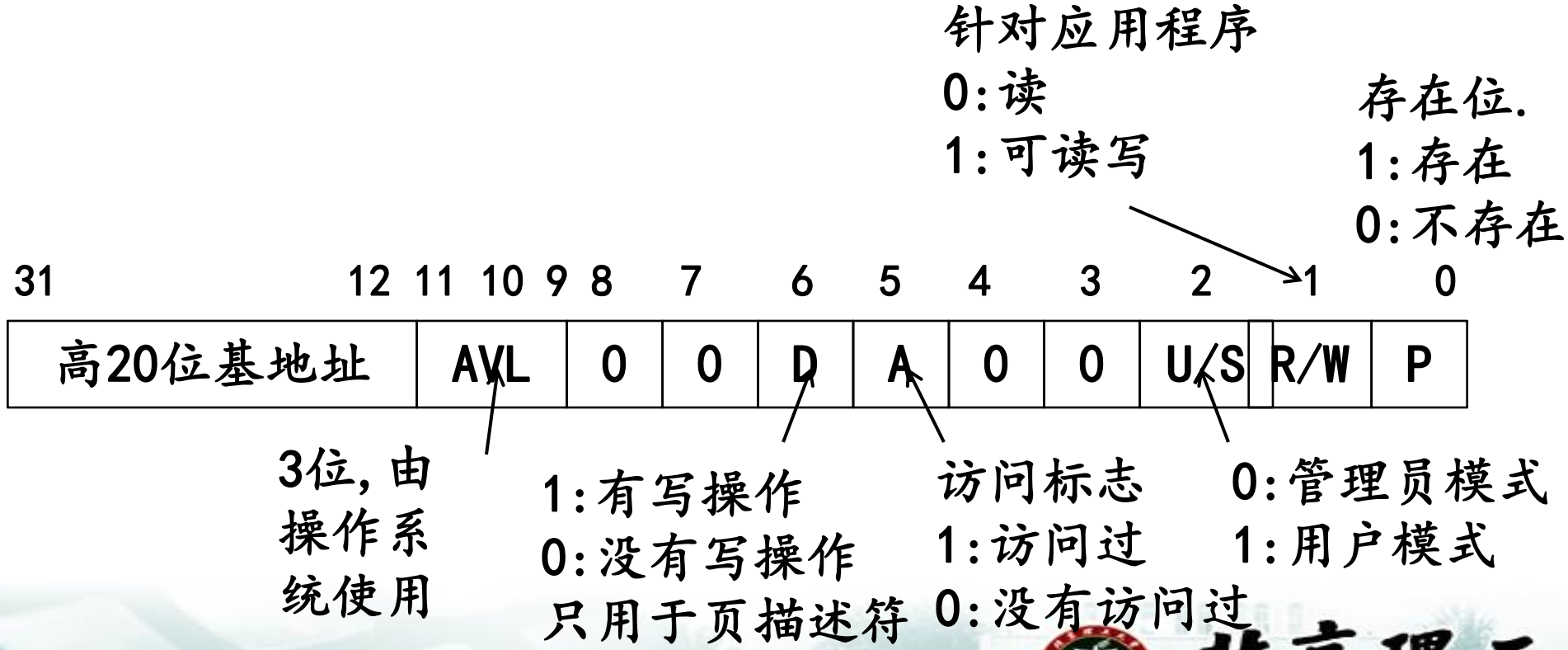


图 3-28 页目录和页表的索引过程

Linear Address从何而来？

页表项

页表项就是在分页转换时用到的页表描述符和页描述符，都是32位，页表项格式如图。

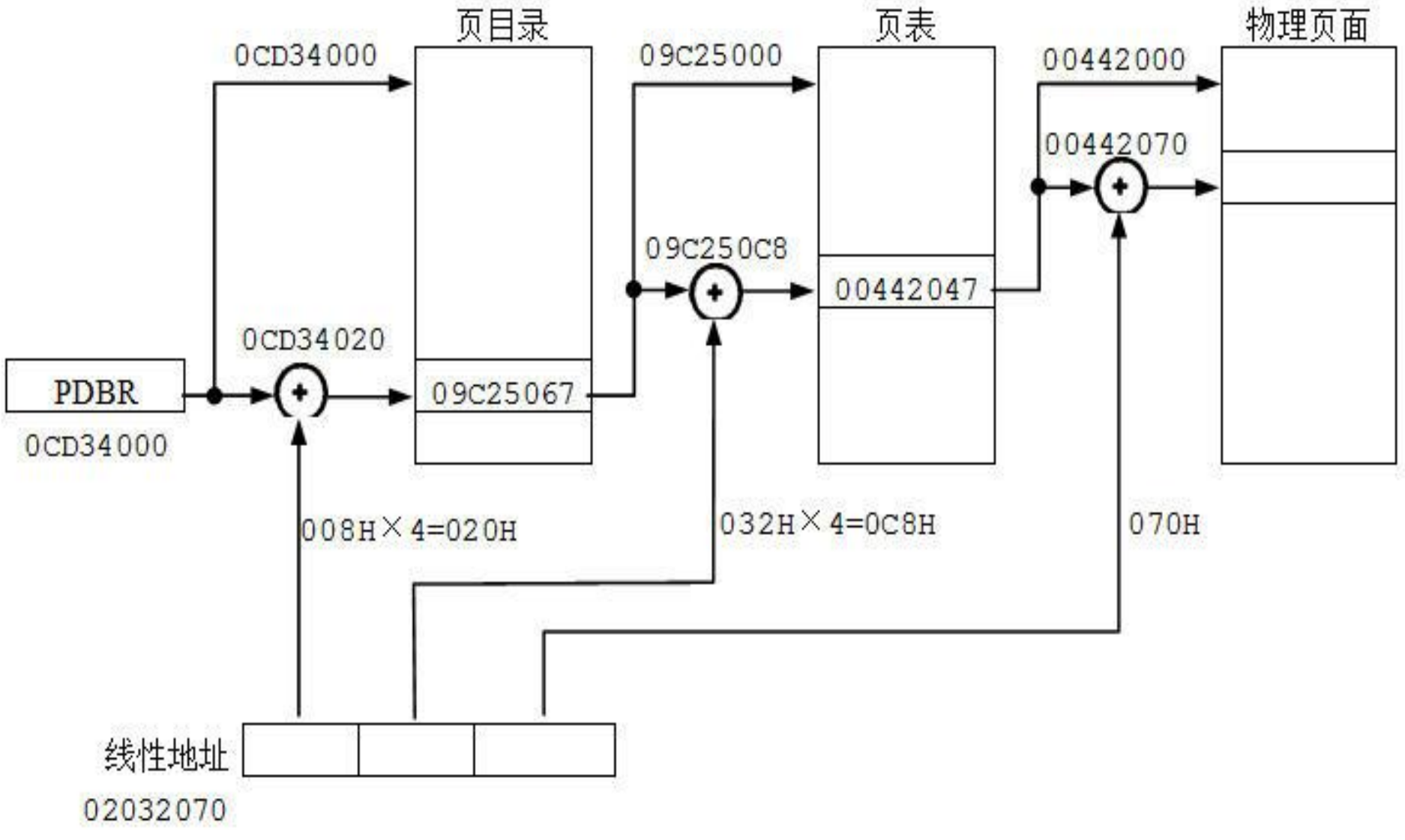


- 页目录、页表和物理页的基地址的低12位全部为0，定位在页的边界上。
- 页表项的低12位提供保护功能和统计信息。U/S位、R/W位、P位实现页保护机制，p32。
- 一个物理页存在两级保护，页表描述符和页描述符属性。当二者不一致的时候取严格属性。

片内转换检测缓冲器TLB

每次内存操作都需要将线性地址转换为物理地址，转换过程中需要访问页目录表和页表来取得页表描述符和页描述符。为了提高转换效率，CPU内部设置了片内转换检测缓冲器TLB（Translation Lookaside Buffer），其中保存了32个页描述符，它们都是最近使用过的。

线性地址转换为物理地址示例P55



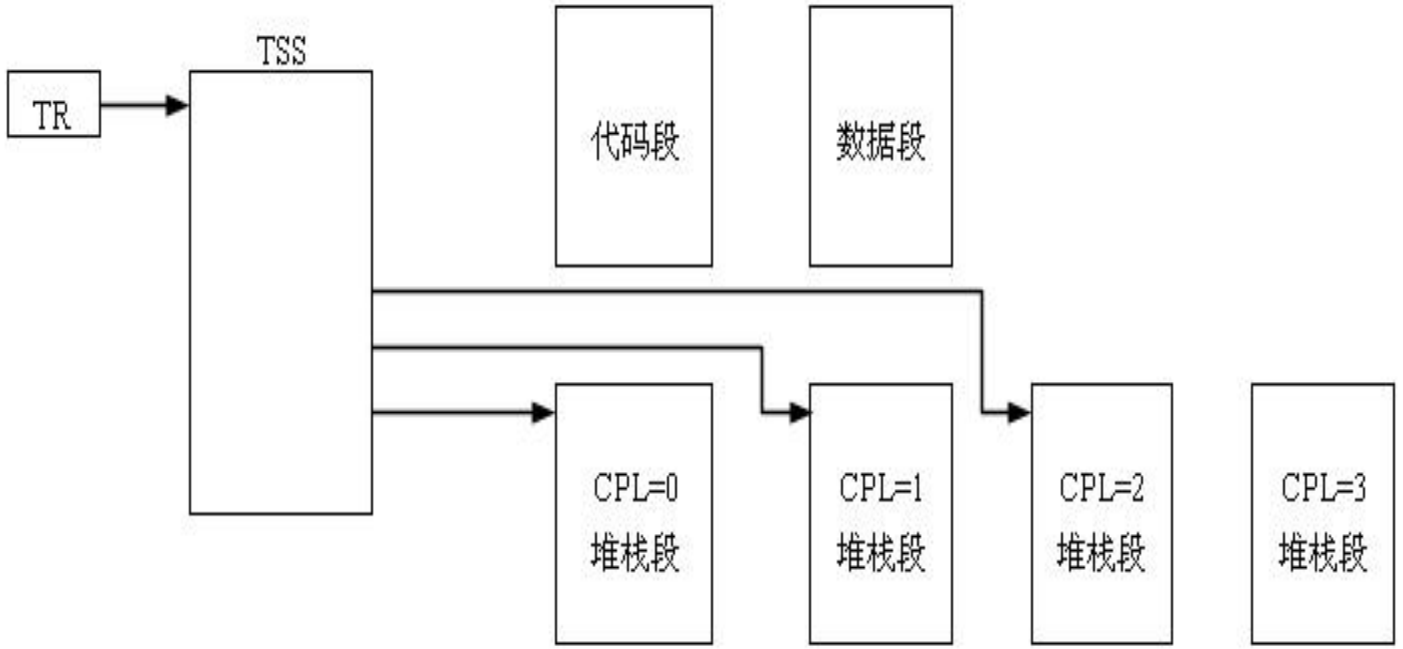
完成某项功能的多个程序的集合称为任务，系统中至少存在一个任务。

当运行一个应用程序后，操作系统就为这个程序创建一个任务。每个任务都由两个部分组成：

- 任务执行环境（Task Executation Space）
- 任务状态段TSS（Task State Segment）。

任务执行环境

任务执行环境包括一个代码段、堆栈段和数据段等，任务在每一个特权级上执行时都有一个堆栈段。



任务状态段TSS

在多任务环境下，每个任务都有属于自己的任务状态段TSS，TSS中包含启动任务所必需的信息，如用户可访问的寄存器等 。

任务状态段TSS在存储器的基地址和限长由TSS描述符指出。

TSS描述符必须放在全局描述符表GDT中，TR内容为选择符，它指出TSS描述符在GDT中的位置。TSS描述符说明各TSS的位置和限长。

TSS描述符格式

	7							0
+0	限长（位7～0）							
+1	限长（位15～8）							
+2	段基址（位7～0）							
+3	段基址（位15～8）							
+4	段基址（位23～16）							
+5	P	DPL		S=0	1	0	B	1
+6	G	0	0	AVL	限长（位19～16）			
+7	段基址（位31～24）							

B=0表示386的可用的任务状态段
B=1表示386的忙碌任务状态段

TSS基本格式

	BIT31—BIT16	BIT15—BIT1	BIT0	Offset
任务状态段基本部分的格式	0000000000000000	链接字段		0
	ESP0			4
	0000000000000000	SS0		8
	ESP1			0CH
	0000000000000000	SS1		10H
	ESP2			14H
	0000000000000000	SS2		18H
	CR3			1CH
	EIP			20H
	EFLAGS			24H
	EAX			28H
	ECX			2CH
	EDX			30H
	EBX			34H
	ESP			38H
	EBP			3CH
	ESI			40H
	EDI			44H
	0000000000000000	ES		48H
	0000000000000000	CS		4CH
	0000000000000000	SS		50H
	0000000000000000	DS		54H
	0000000000000000	FS		58H
	0000000000000000	GS		5CH
	0000000000000000	LDTR		60H
	I/O许可位图偏移	0000000000000000	T	64H

链接字段

内层堆栈指针区域

寄存器保存区域

地址映射寄存器区域

I/O许可位图

门（Gate）可以看做是一种转换机构，可以实现不同特权级别之间的控制传送。

门的类型有4种：

- 调用门：用于控制传送，来改变任务或程序的特权级别。
- 任务门：执行任务切换。
- 中断门和陷阱门：用来指出中断服务程序的入口地址。

门描述符

门描述符描述控制转移的入口点，属于系统描述符。

	7							0
+0	偏移（位 7～0）							
+1	偏移（位 15～8）							
+2	段选择符（位 7～0）							
+3	段选择符（位 15～8）							
+4	0	0	0	参数计数值（5 位）				
+5	P	DPL		S=0	TYPE（4 位）			
+6	偏移（位 23～16）							
+7	偏移（位 31～24）							

调用门

1、系统描述符类型为4或者C时表示调用门。

2、调用门描述某个子程序的入口。调用门内的选择符必须实现代码段描述符，调用门内的偏移是对应代码段内的偏移。

3、利用段间调用指令CALL，通过调用门可实现任务内从低特权级变换到高特权级。

4、如果在利用调用门调用子程序时引起特权级的转换和堆栈的改变，那么就需要将低特权级堆栈中的参数复制到高特权级堆栈。复制的参数个数由调用门描述符中的参数计数值给出。

通过调用门进行段间调用

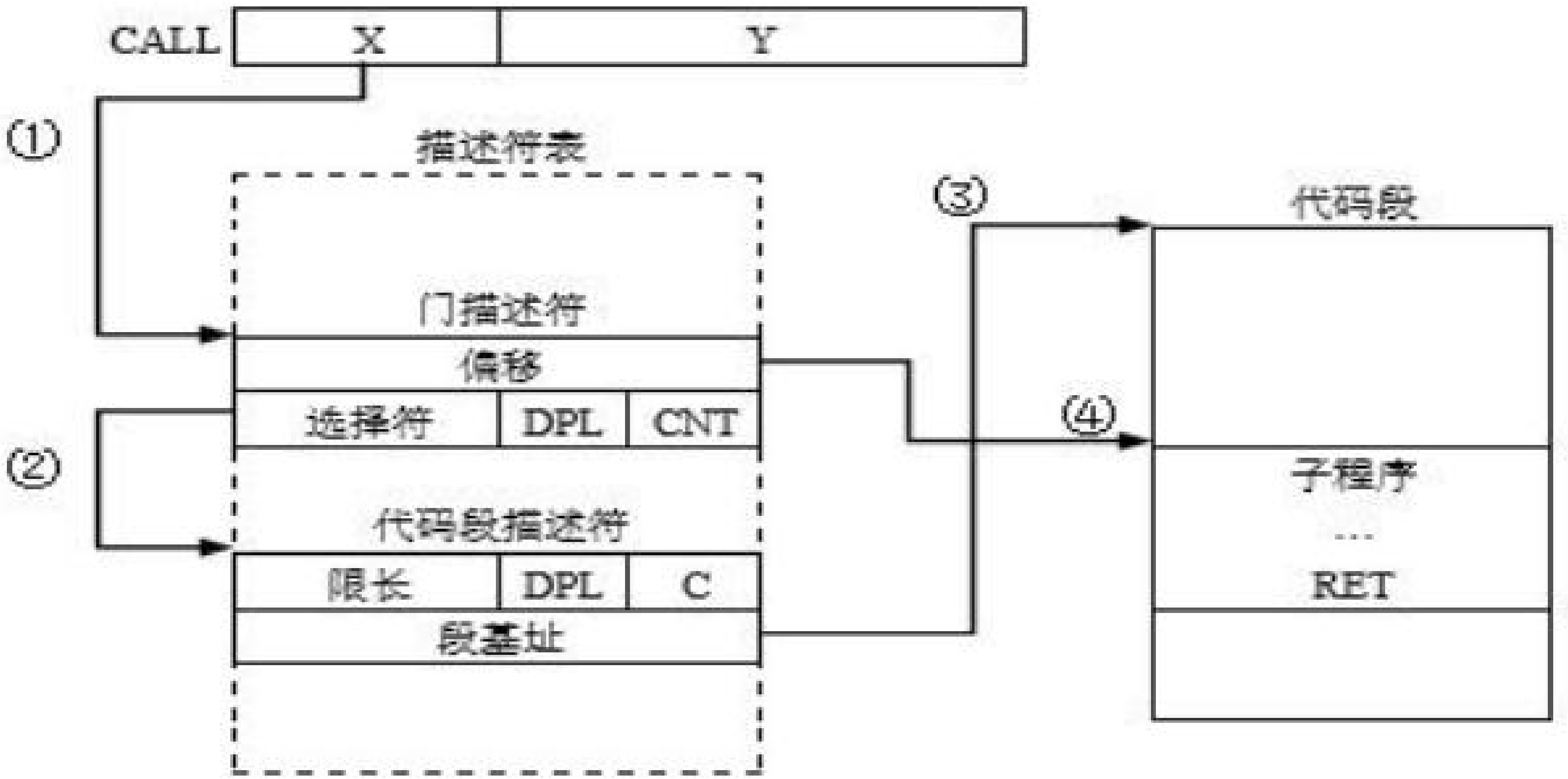


图 3-38 通过调用门转移到更高特权级

这里Y值没有用到。

任务门

- 任务门指示任务。任务门内的选择子必须指示GDT中的任务状态段TSS描述符，门中的偏移无意义。任务的入口点保存在TSS中。

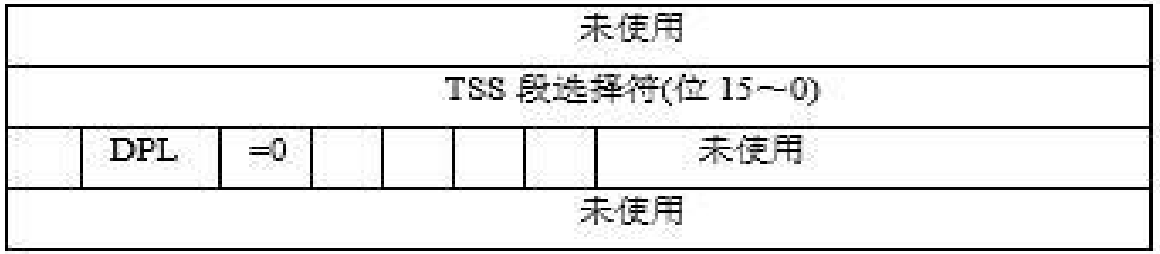


图 3-39 任务门描述符的格式

- 通过任务门可实现任务切换。

中断门和陷阱门

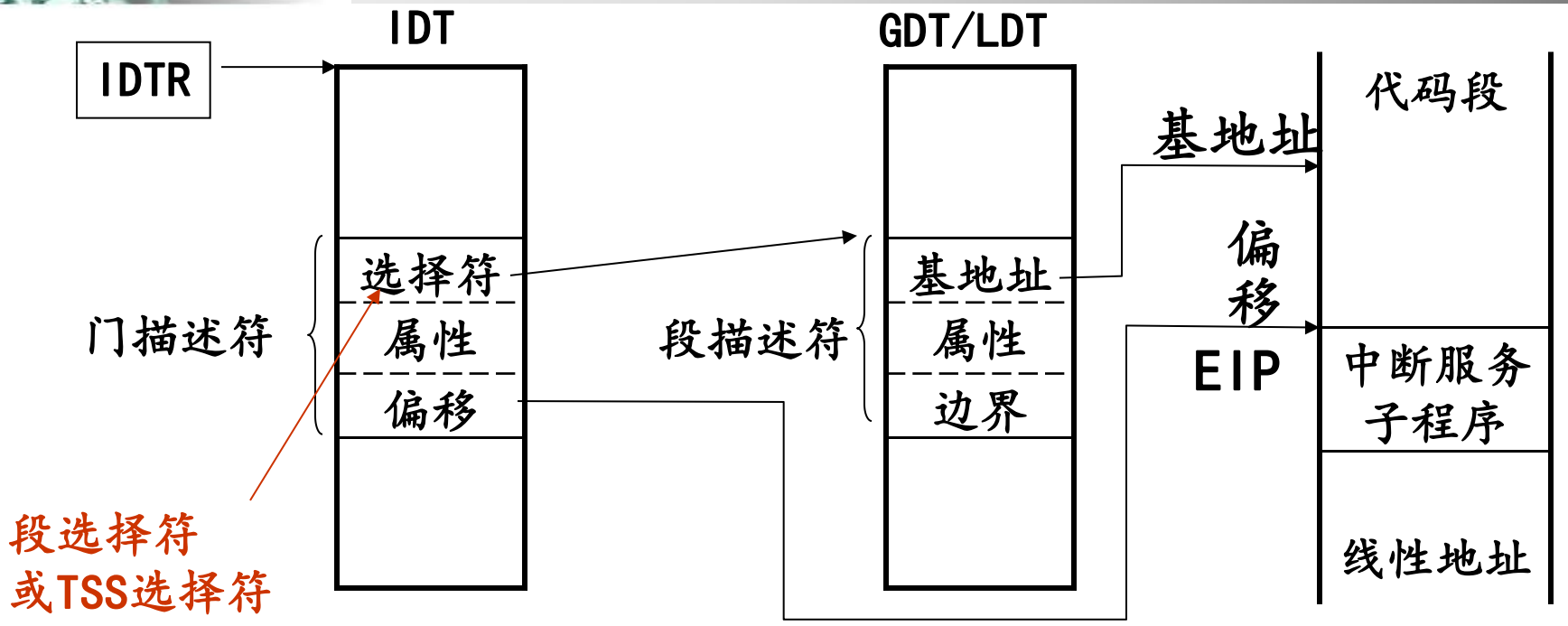
1、中断门和陷阱门描述中断/异常处理程序的入口点。

2、中断门和陷阱门内的选择符必须指向代码段描述符，门内的偏移就是对应代码段的入口点的偏移。

3、中断门和陷阱门只有在中断描述符表IDT中才有效。

中断门和陷阱门的详细内容将在第7章7.3.1中介绍。

门描述符与段选择符关系



说明： 利用段选择符的bit2确定查询GDT或LDT。
段选择符指向一个段描述符。
对任务门而言，选择符为TSS选择符。

任务的切换： 通过直接改变TR方式，
或通过任务门间接方式实现。

任务切换的四种情况

1、执行远程JMP或者CALL指令，选择了GDT中的TSS描述符。

2、执行远程JMP或者CALL指令，从GDT或者LDT中选择了任务门。

3、发生了中断或异常，中断向量选择了IDT中的任务门。

4、当FLAGs中的NT=1时，执行IRET指令，目的任务选择符在执行IRET任务的TSS链接域中。

任务切换的两种方式

- 1、直接任务切换

  在JMP/CALL X:Y指令中，X是一个段选择符，指向一个可用的TSS描述符。

- 2、间接任务切换

- 间接任务切换通过任务门来完成。任务的入口点保存在TSS中。

任务切换过程

如果转移或调用指令将任务状态选择符作为操作数，那么任务进行直接切换。间接切换使用任务门，这是一种将控制传给RPL比CPL高的任务切换方法。

1. 直接任务切换

段间跳转指令JMP X:Y或段间调用指令CALL X:Y可以用来执行任务切换。在中断/异常或者执行IRET指令时也可能发生任务切换。

直接任务切换

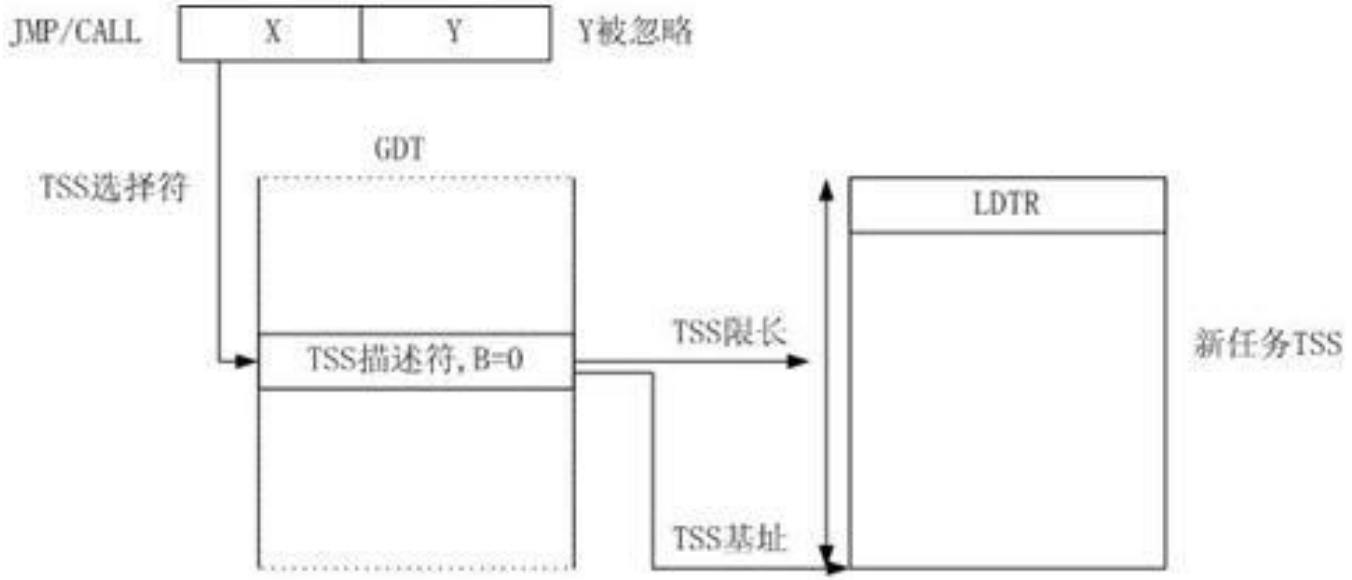


图 3-40 直接任务切换

先由指令中的X段描述符查全局描述符表，获得TSS描述符；TSS描述符提供新任务状态段的基地址和限长等参数。

访问TSS的规则

TSS段描述符的DPL（DPL_{TSS}）规定了访问该描述符的最低特权级，只有在相同级别或更高级别的程序中才可以访问它，即：$DPL_{TSS} \geq MAX$（CPL，RPL），RPL是X的最后2位。

间接任务切换

当CPL＞DPL_{TSS}时，就不能采用任务的直接切换，必须通过任务门进行任务的切换。

JMP和CALL指令中包含的是任务门选择符。选择符指向的是任务门描述符，门中的TSS选择符指向新任务的TSS描述符，激活新的TSS，启动新的任务。

间接任务切换

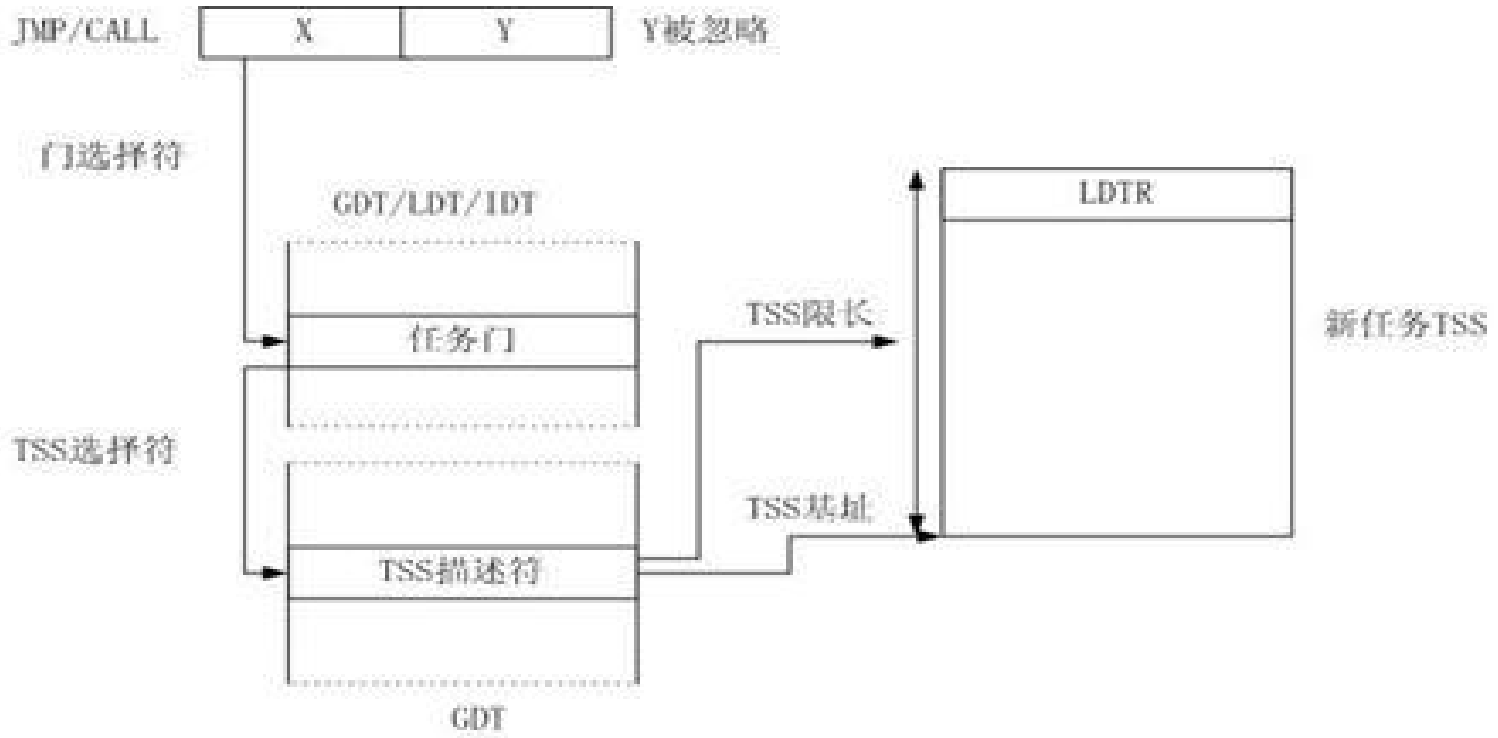


图 3-41 间接任务切换

访问任务门规则

- 任务门的DPL（DPL_{GATE}）规定了访问该任务门的最低特权级，只有在同级或更高级别的程序中才可以访问它。即$DPL_{GATE} \geq MAX$（CPL，RPL），RPL是任务门选择符X的最后2位。

任务A切换到任务B的步骤

1. 把寄存器现场保存到当前任务的TSS（任务A的TSS）。
2. 把指示目标任务（任务B）TSS的选择符装入TR寄存器中，同时把对应TSS的描述符装入TR的高速缓冲寄存器中。
3. 恢复当前任务（任务B）的寄存器现场。只装入选择符，P位为0。
4. 进行链接处理。JMP指令引起的切换不需要链接处理，CALL指令需要。

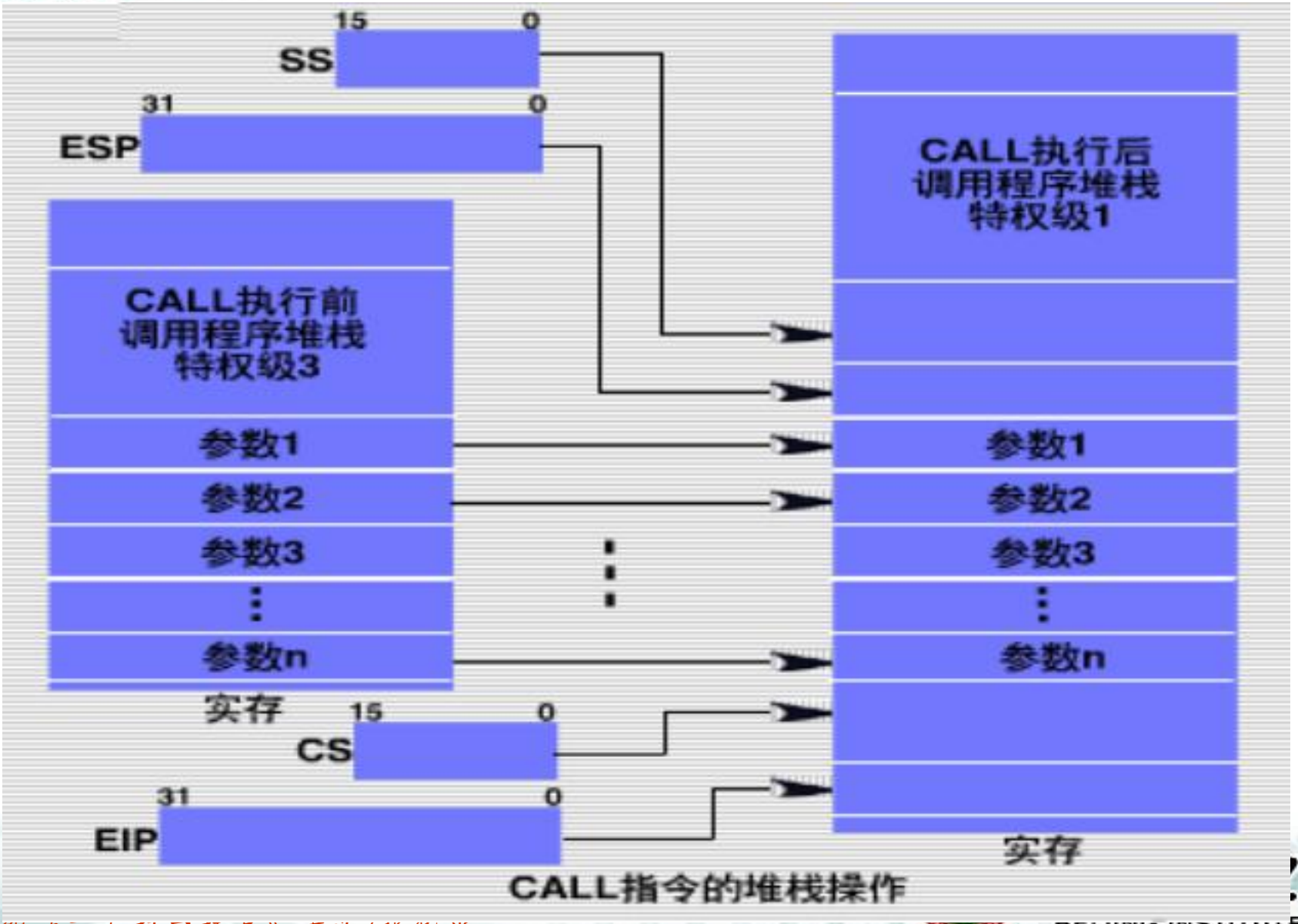
5. 把CR0中的TS标志置为1，这表示已发生过任务切换。
6. 把TSS中的CS选择符的RPL作为当前任务特权级，设置为CPL。
7. 装载LDTR寄存器。
8. 装载代码段寄存器CS、堆栈段寄存器SS和各数据段寄存器及其高速缓冲寄存器。

注意：任务切换不能递归。

TSS中堆栈指针的使用

只有特权级别发生变化时才切换堆栈。

1. TSS中包含有指向0级、1级和2级堆栈的指针。
2. 使用CALL指令通过调用门向高特权级转移。
3. 使用RET指令实现向低特权级转移。



CALL指令的堆栈操作

主要包含以下三个方面：

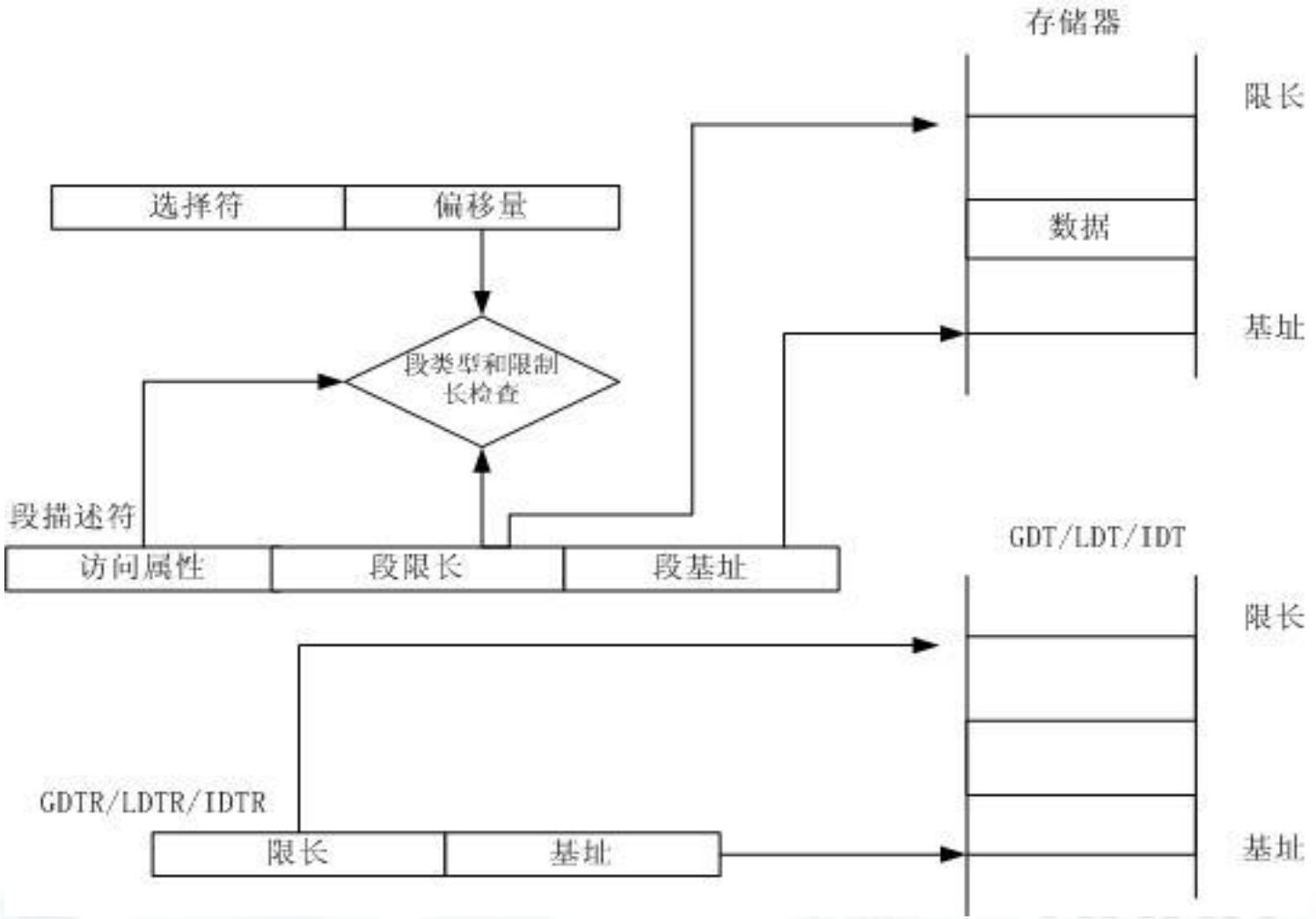
1. 数据访问的保护
2. 对程序的保护
3. 对输入输出的保护

为段寄存器赋值时的检查

保护模式下程序要访问某一个段时，都要完成类型、限长和特权级三种形式的检查。

1、类型检查：检查段描述符的段类型是否与目标一致，例如装入CS时，段描述符中的E位必须为1，标记为可执行的段；装入DS、SS等寄存器时，E位必须为0；只有可写的数据段选择符才能够被加载到SS寄存器中等。

2、限长、类型和属性检查


存储器
限长
选择符
偏移量
数据
段类型和限制
长检查
基址
段描述符
访问属性
段限长
段基址
GDT/LDT/IDT
限长
GDTR/LDTR/IDTR
限长
基址
基址

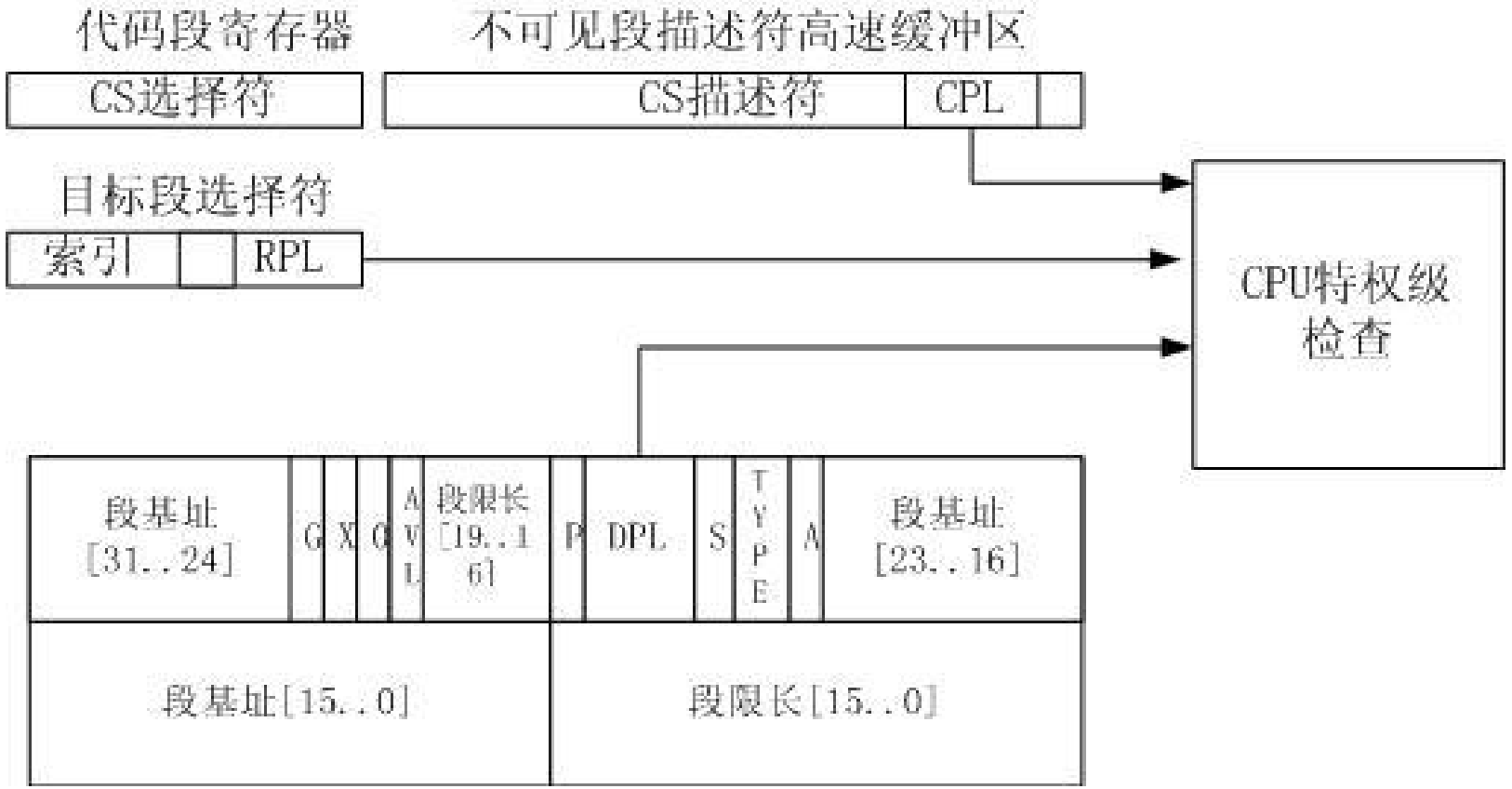
3、特权级检查

CPU在给段寄存器赋值时，要根据CPL、DPL、RPL来检查特权级是否满足要求：DPL ≥ MAX(CPL, RPL)，即程序只能访问特权级相同或者较低的数据。

其中：

- CPL是当前正在运行的程序的特权级（CS）。
- DPL是描述符特权级，它表明了什么样的特权级程序可以使用这个段。
- RPL是请求特权级。表明程序将以什么样的特权级来访问这个段。

特权级检查示意图



数据段保护示例3-11

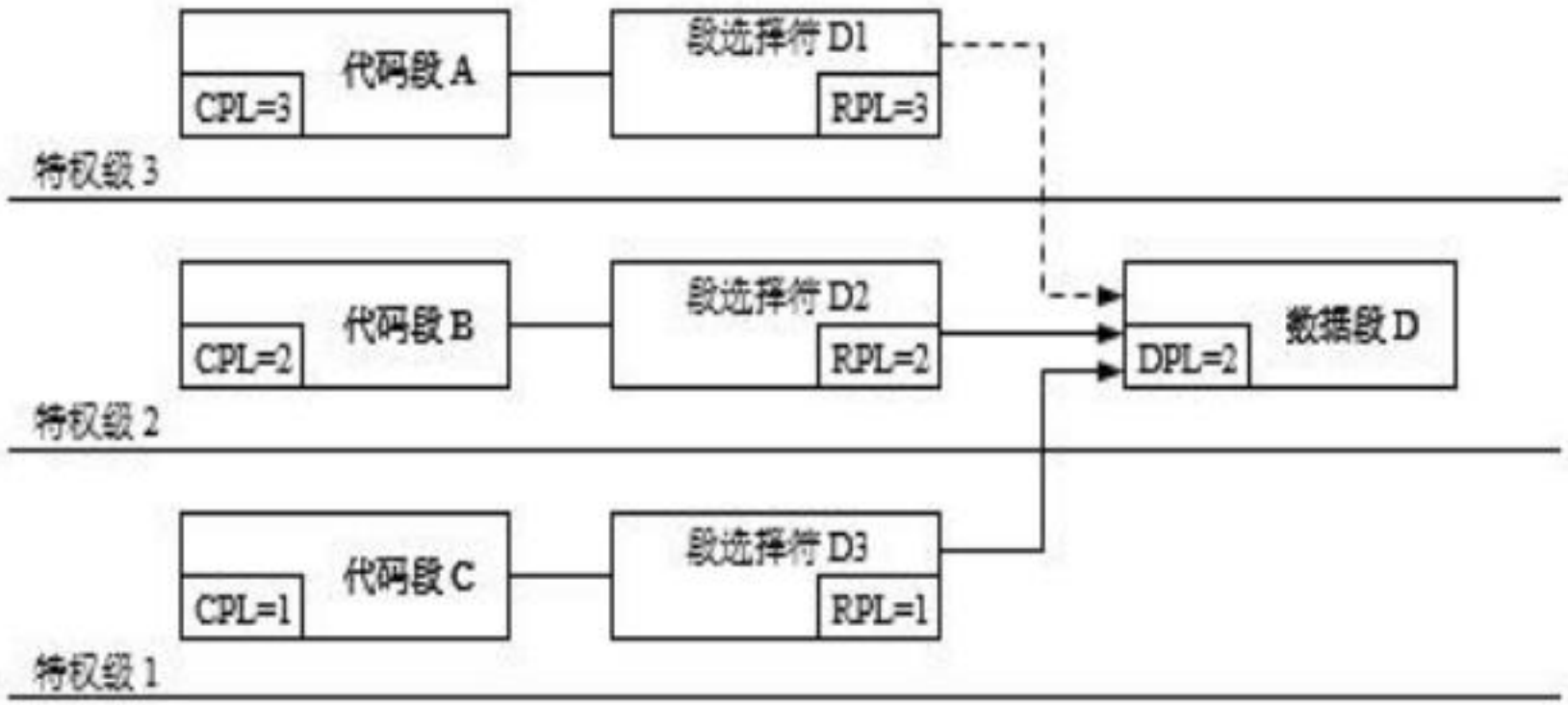


图 3-45 特权级保护数据段访问的一个例子

对程序的保护

- CPU还利用特权级实现对程序执行的控制。

- 根据控制转移方式的不同分为：

1. 直接转移的保护
2. 间接转移的保护

直接转移的保护

1）同一代码段内转移时，只需要检查限长。

2）段间调用或跳转，需要检查限长，特权级CPL和DPL。

- CPL=DPL，允许跳转和调用。
- CPL<DPL，禁止。
- CPL>DPL，此时要检查段描述符的C位。如果C位为1，表示这是一致代码段，允许跳转和调用。

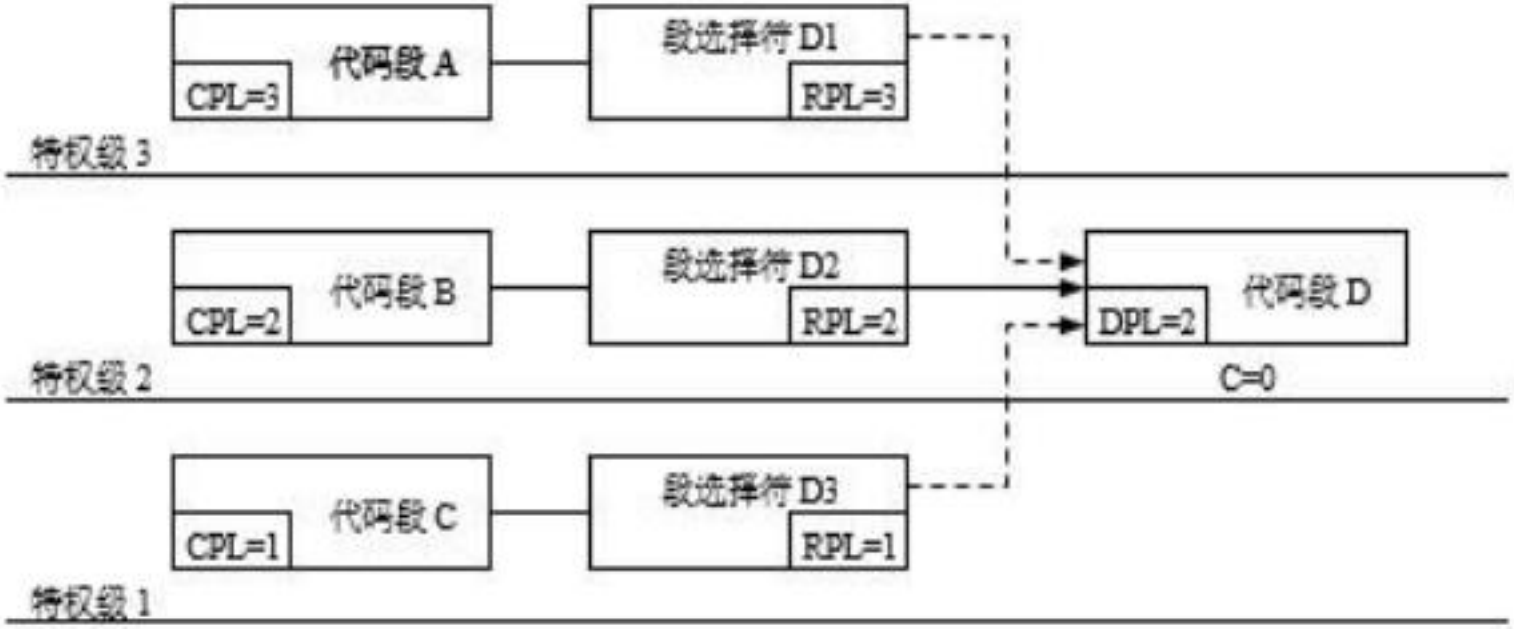


图 3-46 C=0 的代码段

代码段D不是一致代码段（C=0），只能由特权级相同的程序来调用或跳转。特权级较高的代码段C和特权级较低的代码段A都不能调用代码段D。C=0时，还要求RPL ≤ CPL。

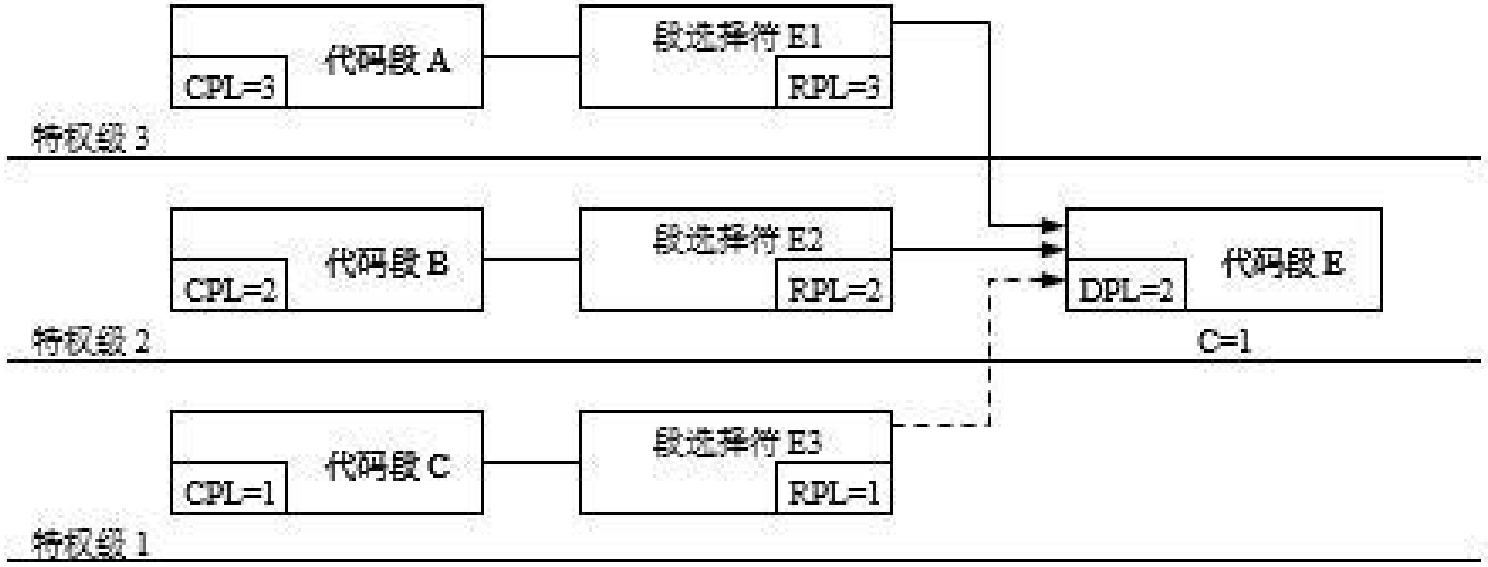


图 3-47 C=1 的代码段

代码段E是一致代码段（C=1），可以由特权级相同或更低的程序来调用或跳转。特权级较高的代码段C不能调用代码段E。而特权级相同的代码段B和特权级更低的代码段A可以转移到代码段E上，转移后特权级CPL不变。

间接转移的保护

- 特权检查时要检查以下几个要素：

1. 当前特权级CPL，即JMP或CALL指令所在的程序的特权级；
2. 请求特权级RPL，即选择符X的最低2位；
3. DPL_{GATE}，即门描述符的DPL；
4. DPL_{CODE}，即目标代码段描述符的DPL；
5. C_{CODE}，即目标代码段描述符的C位（目标代码段是否为一致代码段）。

CALL指令

满足以下两个条件时，才允许使用调用门：

（1） $DPL_{GATE} \geqslant MAX（CPL，RPL）$

（2） $DPL_{CODE} \leqslant CPL$

如果C_{CODE}为1，则当前特权级不变；

如果C_{CODE}为0，则当前特权级提升为DPL_{CODE} （$DPL_{CODE} < CPL$）
或者维持不变（$DPL_{CODE} = CPL$）。

$DPL_{CODE} \leqslant CPL$的条件是防止执行调用门后，当前特权级降低。

JMP指令

满足以下两个条件时，才允许使用调用门：

（1） $DPL_{GATE} \geqslant MAX（CPL，RPL）$

（2） C_{CODE}为1且$DPL_{CODE} \leqslant CPL$ 或 C_{CODE}为0且$DPL_{CODE} = CPL$

如果C_{CODE}为1，则当前特权级不变；

如果C_{CODE}为0，则当前特权级也不变（$DPL_{CODE} = CPL$）。

结论：JMP指令使用调用门不能提升特权级。

间接转移的保护示例

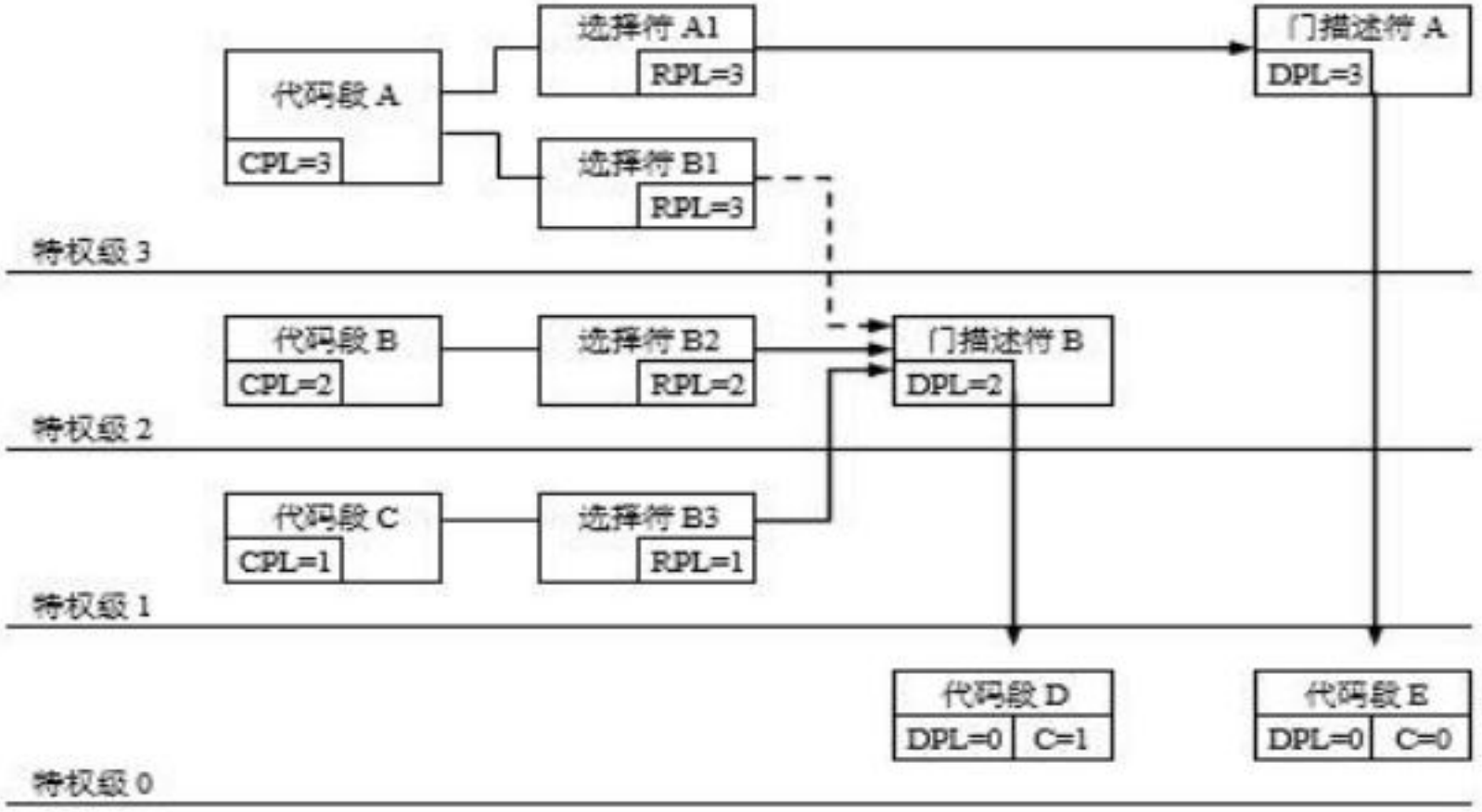


图 3-48 调用门的特权级检查

输入输出保护

CPU采用I/O特权级IOPL（I/O Privilege Level）和TSS段中I/O许可位图的方法来控制输入/输出，实现输入/输出保护。

只要满足一下条件之一就可以进行输入输出：

1. CPL≤IOPL时，可以执行I/O敏感指令。
2. CPL > IOPL时，TSS中IO位图等于0时，也可以执行I/O敏感指令。

基于IOPL的保护

在EFLAGS寄存器中，有2位是输入输出特权级IOPL。CPL≤IOPL时，可以执行I/O敏感指令。

图 3-49 EFLAGS 标志寄存器中的 IOPL

I/O敏感指令

IOPL的敏感指令在实模式下总是可以执行；保护模式下CPL=0也可以执行，即限制是针对应用程序的。

表 3-6 I/O 敏感指令

指令	功能	保护方式下的执行条件
CLI	置 IF 位=0，关中断	CPL<=IOPL
STI	置 IF 位=1，开中断	CPL<=IOPL
IN	从 I/O 地址读数据	CPL<=IOPL 或 I/O 位图许可
INS	从 I/O 地址连续读数据	CPL<=IOPL 或 I/O 位图许可
OUT	向 I/O 地址写数据	CPL<=IOPL 或 I/O 位图许可
OUTS	向 I/O 地址连续写数据	CPL<=IOPL 或 I/O 位图许可

I/O许可位图

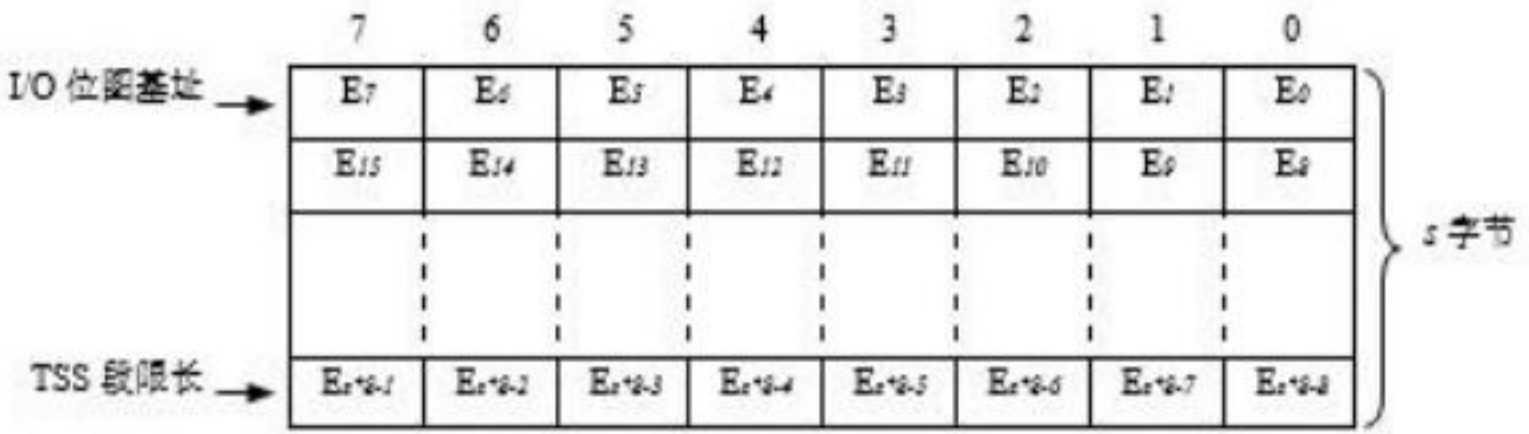


图 3-50 I/O 许可位图

I/O许可位图中每一位对应一个I/O地址。如果位串中的第*n*位取值为0，表示I/O地址*n*就可以由任何特权级的程序访问；取值为1，表示I/O地址*n*只能由在IOPL特权级或更高特权级运行的程序访问，如果低于IOPL特权级的程序访问许可位为1的I/O地址，那么将引起通用保护异常。

I/O许可位图要点

1. Intel 80x86系列计算机的I/O地址空间范围是0000H～0FFFFH，所以I/O许可位图的二进制位串最大为8KB。
2. TSS内偏移66H的字确定I/O许可位图在TSS段中的位置，叫做I/O位图的基址。I/O位图大小=TSS段长度-I/O位图基址。由于I/O许可位图最长可达8KB，所以开始偏移应小于56KB，但必须大于等于104B。
3. 未在I/O位图中说明的I/O地址许可位默认为1，禁止低于IOPL特权级的程序访问这些I/O地址。故I/O位图的大小设置为0，就表示不允许应用程序

涉及多个I/O地址的处理

例如指令：IN EAX, DX

涉及4个I/O地址，则CPU在检查许可位时，这条指令用到的所有I/O地址的许可位都必须为0，才允许访问；如果其中的任何一个许可位为1，则引起保护异常。

对IOPL位的保护

1、只有在特权级0下执行的程序才可以修改IOPL位及VM位；

2、只有IOPL级或更高的特权级的程序才可以修改IF位。

应用程序能够访问的I/O地址空间受操作系统限制。当CPL≤IOPL的条件不能满足，应用程序能够访问的I/O端口就取决于I/O位图。

感谢关注聆听！

张华平
Email: kevinzhang@bit.edu.cn
微博：@ICTCLAS张华平博士
实验室官网：
http://www.nlpir.org

大数据千人会