MULTICS有256K个独立的段，每个段最长可以有64K个36位字。Pentium处理器有16K个独立的段，每个段最多可以容纳10亿个32位字。这里虽然段的数目较少，但是相比之下Pentium较大的段大小特征比更多的段个数要重要得多，因为几乎没有程序需要1000个以上的段，但是有很多程序需要大段。

Pentium处理器中虚拟内存的核心是两张表，即LDT（Local Descriptor Table，局部描述符表）和GDT（Global Descriptor Table，全局描述符表）。每个程序都有自己的LDT，但是同一台计算机上的所有程序共享一个GDT。LDT描述局部于每个程序的段，包括其代码、数据、堆栈等；GDT描述系统段，包括操作系统本身。

为了访问一个段，一个Pentium程序必须把这个段的选择子（selector）装入机器的6个段寄存器的某一个中。在运行过程中，CS寄存器保存代码段的选择子，DS寄存器保存数据段的选择子，其他的段寄存器不太重要。每个选择子是一个16位数，如图3-39所示。

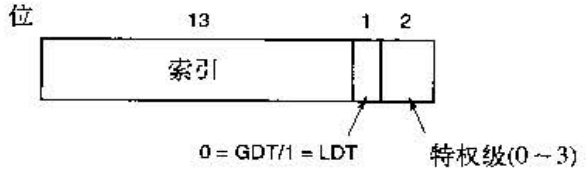


图3-39 Pentium处理器中的选择子

选择子中的一位指出这个段是局部的还是全局的（即它是在LDT中还是在GDT中），其他的13位是LDT或GDT的表项编号。因此，这些表的长度被限制在最多容纳8K个段描述符。还有两位和保护有关，我们将在后面讨论。描述符0是禁止使用的，它可以被安全地装入一个段寄存器中用来表示这个段寄存器目前不可用，如果使用会引起一次陷阱。

在选择子被装入段寄存器时，对应的描述符被从LDT或GDT中取出装入微程序寄存器中，以便快速地访问。一个描述符由8个字节构成，包括段的基地址、大小和其他信息，如图3-40所示。

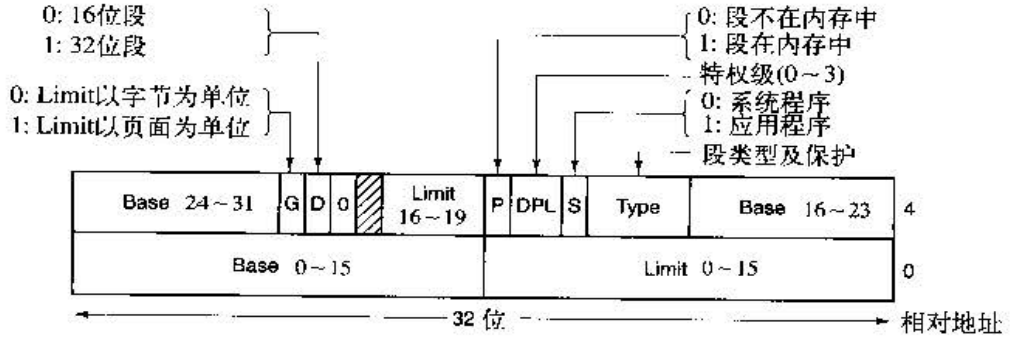


图3-40 Pentium处理器代码段描述符（数据段稍有不同）

选择子的格式经过合理设计，使得根据选择子定位描述符十分方便。首先根据第2位选择LDT或GDT；随后选择子被复制进一个内部擦除寄存器中并且它的低3位被清0；最后，LDT或GDT表的地址被加到它上面，得出一个直接指向描述符的指针。例如，选择子72指向GDT的第9个表项，它位于地址GDT+72。

现在让我们跟踪一个描述地址的（选择子，偏移量）二元组被转换为物理地址的过程。微程序知道我们具体要使用哪个段寄存器后，它就能从内部寄存器中找到对应于这个选择子的完整的描述符。如果段不存在（选择子为0）或已被换出，则会发生一次陷阱。

硬件随后根据Limit（段长度）域检查偏移量是否超出了段的结尾，如果是，也发生一次陷阱。从逻辑上来说，在描述符中应该简单地有一个32位的域给出段的大小，但实际上剩余20位可以使用，因此采用了一种不同的方案。如果Gbit（Granularity）域是0，则是精确到字节的段长度，最大1MB；如果是1，Limit域以页面替代字节作为单元给出段的大小。Pentium处理器的页面大小是固定的4KB，因此20位足以描述最大$2^{32}$字节的段。

假设段在内存中并且偏移量也在范围内，Pentium处理器接着把描述符中32位的基址和偏移量相加形成线性地址（linear address），如图3-41

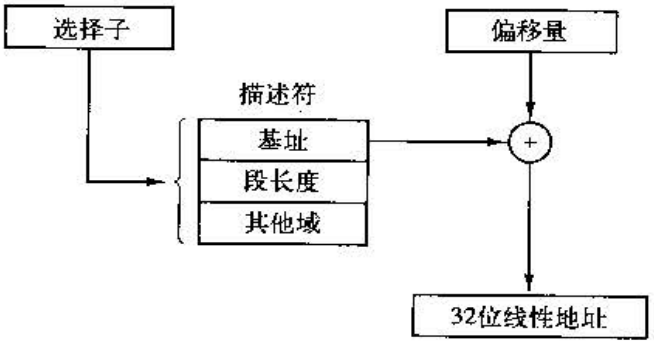


图3-41 （选择子，偏移量）对转换为线性地址