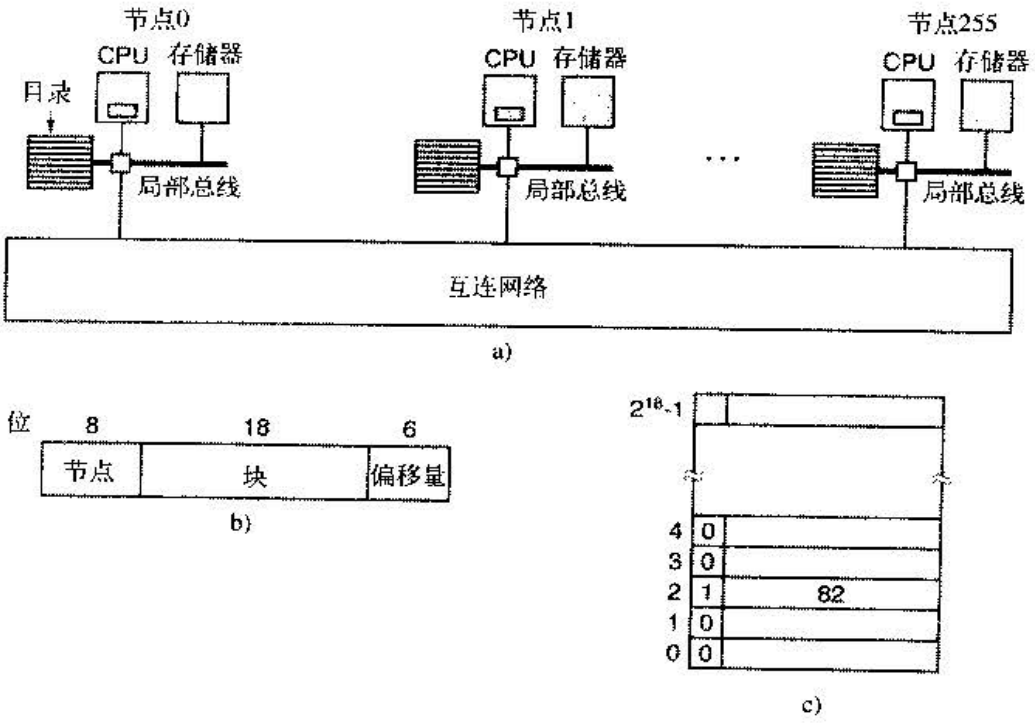

图8-6 a) 256个节点的基于目录的多处理机；b) 32位存储器地址划分的域；c) 节点36中的目录

当请求通过互连网络到达节点36时，它被路由至目录硬件。硬件检索其包含2^{18}个表项的目录表（其中的每个表项代表一个高速缓存行）并解析到项4。从图8-6c中，我们可以看到该行没有被高速缓存，所以硬件从本地RAM中取出第4行，送回给节点20，更新目录项4，指出该行目前被高速缓存在节点20处。

现在来考虑第二个请求，这次访问节点36的第2行。在图8-6c中，我们可以看到这一行在节点82处被高速缓存。此刻硬件可以更新目录项2，指出该行现在在节点20上，然后送一条消息给节点82，指示把该行传给节点20并且使其自身的高速缓存无效。注意，即使一个所谓“共享存储器多处理机”，在下层仍然有大量的消息传递。

让我们顺便计算一下有多少存储器单元被目录占用。每个节点有16 MB的RAM，并且有2^{18}个9位的目录项记录该RAM。这样目录上的开支大约是9×2^{18}位除以16 MB，即约1.76%，一般而言这是可接受的（尽管这些都是高速存储器，会增加成本）。即使对于32字节的高速缓存行，开销也只有4%。至于128字节的高速缓存行，它的开销不到1%。

该设计有一个明显的限制，即一行只能被一个节点高速缓存。要想允许一行能够在多个节点上被高速缓存，我们需要某种对所有行定位的方法，例如，在写操作时使其无效或更新。要允许同时在若干节点上进行高速缓存，有几种选择方案，不过对它们的讨论已超出了本书的范围。

5. 多核芯片

随着芯片制造技术的发展，晶体管的体积越来越小，从而有可能将越来越多的晶体管放入一个芯片中。这个基于经验的发现通常称为摩尔定律（Moore's Law），得名于首次发现该规律的Intel公司创始人之一Gordon Moore。Intel Core 2 Duo系列芯片已包含了3亿数量级的晶体管。

随之一个显而易见的问题是：“你怎么利用这些晶体管？”按照我们在第1.3.1小节的讨论，一个选择是给芯片添加数兆字节的高速缓存。这个选择是认真的，带有4兆字节片上高速缓存的芯片现在已经很常见，并且带有更多片上高速缓存的芯片也即将出现。但是到了某种程度，再增加高速缓存的大小只能将命中率从99%提高到99.5%，而这样的改进并不能显著提升应用的性能。

另一个选择是将两个或者多个完整的CPU，通常称为核（core），放到同一个芯片上（技术上来说是同一个小硅片）。双核和四核的芯片已经普及，八十核的芯片已经被制造出来，而带有上百个核的芯片也即将出现。

虽然CPU可能共享高速缓存或者不共享（如图1-8所示），但是它们都共享内存。考虑到每个内存字总是有惟一的值，这些内存是一致的。特殊的硬件电路可以确保在一个字同时出现在两个或者多个的高