在消息通过交换网络之后，模块号的左端的位就不再需要了。它们可以有很好的用途，可以用来记录入线编号，这样，应答消息可以找到返回路径。对于路径a，入线编号分别是0（向上输入到1D）、1（低输入到2D）和1（低输入到3D）。使用011作为应答路由，只要从右向左读出每位即可。

在上述这一切进行的同时，CPU 001需要往存储器001里写入一个字。这里发生的情况与上面的类似，消息分别通过上、上、下端输出路由，由字母b标出。当消息到达时，从Module域读出001，代表了对应的路径。由于这两个请求不使用任何相同的开关、连线或存储器模块，所以它们可以并行工作。

现在考虑如果CPU 000同时也请求访问存储器模块000会发生什么情况。这个请求会与CPU 001的请求在开关3A处发生冲突。它们中的一个就必须等待。和交叉开关不同，Omega网络是一种阻塞网络，并不是每组请求都可被同时处理。冲突可在一条连线或一个开关中发生，也可在对存储器的请求和来自存储器的应答中产生。

显然，很有必要在多个模块间均匀地分散对存储器的引用。一种常用的技术是把低位作为模块号。例如，考虑一台经常访问32位字的计算机中面向字节的地址空间，低位通常是00，但接下来的3位会均匀地分布。将这3位作为模块号，连续的字会放在连续的模块中。而连续字被放在不同模块里的存储器系统被称作交叉（interleaved）存储器系统。交叉存储器将并行运行的效率最大化了，这是因为多数对存储器的引用是连续编址的。设计非阻塞的交换网络也是有可能的，在这种网络中，提供了多条从每个CPU到每个存储器的路径，从而可以更好地分散流量。

4. NUMA多处理机

单总线UMA多处理机通常不超过几十个CPU，而交叉开关或交换网络多处理机需要许多（昂贵）的硬件，所以规模也不是那么大。要想超过100个CPU还必须做些让步。通常，一种让步就是所有的存储器模块都具有相同的访问时间。这种让步导致了前面所说的NUMA 多处理机的出现。像UMA一样，这种机器为所有的CPU提供了一个统一的地址空间，但与UMA 机器不同的是，访问本地存储器模块快于访问远程存储器模块。因此，在NUMA 机器上运行的所有UMA程序无须做任何改变，但在相同的时钟速率下其性能不如UMA 机器上的性能。

所有NUMA 机器都具有以下三种关键特性，它们使得NUMA机器与其他多处理机相区别：

1) 具有对所有CPU都可见的单个地址空间。

2) 通过LOAD和STORE指令访问远程存储器。

3) 访问远程存储器慢于访问本地存储器。

在对远程存储器的访问时间不被隐藏时（因为没有高速缓存），系统被称为NC- NUMA（No Cache NUMA，无高速缓存NUMA）。在有一致性高速缓存时，系统被称为CC- NUMA（Cache-Coherent NUMA，高速缓存一致NUMA）。

目前构造大型CC-NUMA多处理机最常见的方法是基于目录的多处理机（directory-based multiprocessor）。其基本思想是，维护一个数据库来记录高速缓存行的位置及其状态。当一个高速缓存行被引用时，就查询数据库找出高速缓存行的位置以及它是“干净”的还是“脏”（被修改过）的。由于每条访问存储器的指令都必须查询这个数据库，所以它必须配有极高速的专用硬件，从而可以在一个总线周期的几分之一内作出响应。

要使基于目录的多处理机的想法更具体，让我们考虑一个简单的（假想）例子，一个256个节点的系统，每个节点包括一个CPU和通过局部总线连接到CPU上的16MB的RAM。整个存储器有2^{32}字节，被划分成2^{26}个64字节大小的高速缓存行。存储器被静态地在节点间分配，节点0 是0~16M，节点1是16~32M，以此类推。节点通过互连网络连接，参见图8-6a。每个节点还有用于构成其2^{24}字节存储器的2^{18}个64字节高速缓存行的目录项。此刻，我们假定一行最多被一个高速缓存使用。

为了了解目录是如何工作的，让我们跟踪引用了一个高速缓存行的发自CPU 20的LOAD指令。首先，发出该指令的CPU把它交给自己的MMU，被翻译成物理地址，比如说，0x24000108。MMU将这个地址拆分为三个部分，如图8-6b所示。这三个部分按十进制是节点36、第4行和偏移量8。MMU看到引用的存储器字来自节点36，而不是节点20，所以它把请求消息通过互连网络发送到该高速缓存行的的主节点（home node）36上，询问行4是否被高速缓存，如果是，高速缓存在何处。