图9-6的左侧显示了该数据在一种假想的体系结构中的一种可能的内存安排：在一个单字节内存中包含了字符“a”，其后的16比特的字包含了整数值259，以高阶字节在先的顺序存储。在另一台计算机中的内存安排显示在图9-6的右侧。字符“a”后面是16比特的整数值，以低阶字节在先的顺序存储，它们以16比特的字为边界对齐。当然，如果某人在这两台机器的内存之间执行逐字复制，使用相同的结构定义来存取存储的值，他将在这两台计算机上看到极为不同的结果！

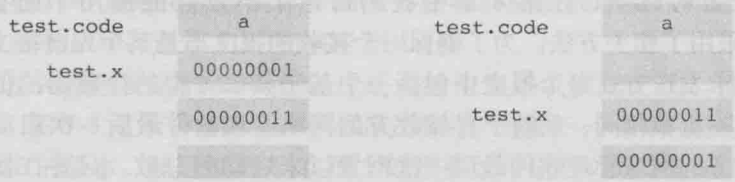

图9-6 两种不同体系结构上的两种不同的数据安排

不同的体系结构具有不同的内部数据格式，这是一个真实而普遍的问题。以不同的形式存储整数的特定问题非常常见，并且有了一个专有名称。用于存储整数的“大端法”（big-endian）次序要求最高阶字节先存储（在最低的存储地址），“小端法”（little-endian）次序要求先存储最低阶字节。Sun SPARC和Motorola处理器采用“大端法”，而Intel和DEC/Compaq Alpha处理器采用“小端法”。作为一个小插曲提一下，术语“big-endian”和“little-endian”来自Jonathan Swift的书《格列佛游记》，书中的两群人固执己见地坚持以两种不同方式做一件简单的事情（希望对计算机体系结构方面的这个类比有助于理解）。来自小人国的一群人坚持在较大一端打开鸡蛋（“大端法”），而其他人坚持在较小一端打开鸡蛋。这个分歧导致了大规模民事冲突和叛乱。

既然存在不同方式的计算机存储和数据表示，网络协议该怎样处理这些问题呢？例如，如果一个SNMP代理打算发送一个响应报文，其中包含接收到的UDP数据报数量的整数计数器的值，它怎样表示将向管理实体发送的该整数值呢？是以大端法还是以小端法次序呢？代理的一种选择是以在管理实体中存储的相同顺序来发送这些字节。另一种选择是代理以自己的存储顺序发送，并让接收实体根据需要重新排序。任何一种选择都将要求发送方或接收方知道其他人的整数表示格式。

第三种选择是用与机器无关、OS无关、语言无关的方式描述整数和其他数据类型（即一种数据定义语言）和规则，该规则定义了每种数据类型跨越网络传输的方式。当接收到给定类型的数据时，它以一种已知格式来接收，并能以任意机器特定的格式进行存储。我们在9.3节中学习的SMI和ASN.1都采用了第三种选择。用ISO术语来说，这两个标准描述了一种**表示服务**（presentation service），该服务用于将信息从一种机器特定格式传输和转换到另一种机器特定格式。图9-7阐述了一个真实世界中的表示问题；两个接收方都不理解交流的基本思想，如谈话者喜欢的东西。如图9-8所示，一个表示服务能够解决该问题的过程是，通过将思想转换成共同理解的（通过表示服务）、与个人无关的语言，向接收方发送该信息，然后转换成接收方能理解的语言。

表9-5显示了一些ASN.1定义的数据类型。前面我们在学习SMI时已遇到了INTEGER、OCTET STRING和OBJECT IDENTIFIER数据类型。这里我们的目标不是提供对ASN.1的完整概述，因此我们向读者推荐描述ASN.1类型和结构的标准或已出版的和在线的书籍