《计算机系统》理论课—程序机器级表示：数据

课堂要点

本次课堂教学主要内容为：**数组的存储及访问、结构体的存储及访问、联合体的存储及访问**。本节课内容着重讲解以上三种特殊数据结构在机器中的存储特点及元素访问方法，以及其汇编代码的实现。要求学生能够理解机器如何进行上述数据结构的读与写操作

1.**数组**：数组是某一类型数据的集合，在 C 语言中声明时要给出数据类型，数组名，及元素个数。数组在内存中一般被分配一段连续的空间，且空间大小为元素个数×数据类型字节数。

对一维数组元素的访问，元素地址 = 数组首地址+元素下标×类型字节数

注意：第一个元素为 A[0]

对二维数组的访问，

元素地址 = 所在行首地址+元素列标×类型字节数

其中：**所在行首地址 = 数组首地址+元素行标×数组列数×类型字节数**

具体公式参见 PPT。

对嵌套数组元素的访问，**首先要取得元素所在“行向量”的头指针**（首地址），然后**再依据一维数组的访问方式计算元素的地址**（因此有两次内存访问）

2. 结构体：结构（struct）是将不同类型数据打包集中存放和访问的一种方式。结构体 r 的访问在 C 语言中通过 r.i 或者 r->的形式实现。在机器中，将 r 看做一个一维不定长数组（元素长度各异），用首地址加字节偏移量的方式访问。如果是结构体数组（每一个数组元素都是一个结构体），则先要定位到元素所属结构体在数组中的位置，再对结构体进行“一维”数组访问定位具体元素的地址。

结构体的存储要遵循三大对齐规则：

1. 结构的首地址必须是最大元素字节数的整数倍；

2. 结构中每个数据的地址必须是其字节数的整数倍；

3. 结构的总体长度必须是最大元素字节数的整数倍。

需要注意 IA32 Linux 和 Windows（及 X86-64）对齐规则的区别！

考虑到对齐规则的影响，在代码中声明结构体时，为了节省存储空间，推荐的做法是将尺寸较大的数据（例如 double 类型）放在前面，较小的（例如 short 和 char）放在后面。

3.**联合体：**联合与结构类似的地方在于都是将不同类型的数据打包，但是联合体只有一段符合最大数据尺寸的内存字节，其它声明的类型都从这一段字节中按照声明时的位置来取对应的字节作为自己的数值。这**类似于给定一个二进制向量，然后用不同的解释方法来解释它（或者它的一部分）。**

在访问联合体时，要注意从中取部分字节时的字节顺序。如同课件上举的例子一样（大端法，小端法，32 位系统，64 位系统）。