\_06\_包含多个段的程序

**1.伪指令DW和DB**

**①DW（Define Word）：定义字**

语法：**变量名 DW 字数据1 [,字数据2] [,字数据3....]**

功能：在内存中开辟一片连续的存储空间，用于存放字数据。变量名表示这块空间的首元素的偏移地址。

变量名详解：

①**变量名：DW定义的变量名（标签），经过编译和连接过程后会被转换为“数组首元素的偏移地址”。**

②**访问DW定义的内存空间中的数据：由于变量名表示数组首元素的偏移地址，因此 “[变量名]” 表示变量的值；“[变量名+常数]” 表示数组中指定位置的元素的值。（注意：直接调用变量名(不使用括号)，是不合法的）**

**③通过offset操作符获取变量的偏移地址：“offset 变量名”可获取变量的偏移地址。（注意：虽然变量名本身就表示首元素的偏移地址，但是直接使用变量名是不合法操作）**

注意事项：

①DW定义的空间中存放字数据（2字节），因此数组的每个元素之间地址相差2。

②**DW是伪指令，本身不占内存空间。DW经过编译后会直接在当前段中（不一定是ds），DW语句所在的位置开辟内存空间。**

**②DB（Define Byte）：定义字节**

语法：**变量名 DB 字节数据1 [,字节数据2] [,字节数据3....]**

功能：在内存中开辟一片连续的存储空间，用于存放字节数据。变量名表示这块空间的首元素的偏移地址。

注意事项：DB的用法和功能与DW基本相同，二者唯一的区别在于DW定义的是字数据，而DB定义的是字节数据。

**2.将数据、栈、代码放入不同的段**

语法如下：



注意事项：

①在代码段中，我们必须首先将数据段地址datasg传入ds中，将栈段地址stacksg传入ss中。

注意：assume ds:datasg, ss:stacksg，功能是将ds与datasg、ss与stacksg相关联。assume的关联是做给编译器看的，该语句使用后编译器会把datasg认作数据段、把stacksg认作代码段。但是assume伪指令并不会改变cpu中段寄存器的值，因此我们必须在代码段中手动为段寄存器赋值。

②当一个.asm文件中定义多个段，我们必须用“end 标号”定义程序的入口地址。

注意：若不定义入口地址，程序默认从上到下，将第一个段的首行作为入口，将此处存放的数据认作机器码执行，导致程序出错。

③伪指令中的变量名、标号会被转换为偏移地址（常数），段名会被转换为段地址（常数）。因此，段名不能直接传入段寄存器，而是需要使用通用寄存器作为中转。