\_05\_[BX]和loop指令

**1.[BX]**

**（1）基址寄存器BX**

BX（Base Register）：基址寄存器（常用于存放存储单元的偏移地址）。

8086CPU的BX是16位的基址寄存器，它在各种**内存访问操作**中扮演着重要的角色。**BX主要用于存储内存地址的偏移量**，与段寄存器（DS、SS、ES）配合使用，共同表示 20位的物理地址。

**（2）[BX]**

8086CPU中，[…]均表示内存空间。（内存空间[XXX]对应的段地址是DS）

[BX]表示一个内存空间。[BX]是当前DS段中，BX的值作为偏移地址所指向的内存空间。[BX]的逻辑地址为DS:BX。

寄存器间接寻址：即[BX]，通过寄存器中存放的偏移地址，间接的访问内存空间。

**注意：4个通用寄存器中，只有BX可以用来间接寻址：[BX]**

[AX]、[CX]、[DX]：均是错误写法，会报错。

（3）段前缀

概念：段前缀，即“cs:”、“ds:”、“ss:”、“es:”。段前缀可以在访问内存单元[XXX]时，显式的指明该内存单元的段地址。

注意：①所有[XXX]内存单元的段地址，默认都是ds数据段寄存器中的值。

②**拓展段寄存器es：常用于存放程序中附加段的段地址。当我们想要描述多段内存时，一个ds不够用，那么可以用es存放数据段地址，配合段前缀es:[XXX]即可描述内存空间。**

e.g. mov ax, [bx] 等价于 mov ax, ds:[bx]

也可以通过段前缀，将内存中cs、ss、es段的内容送到ax中：

mov ax, cs:[bx]

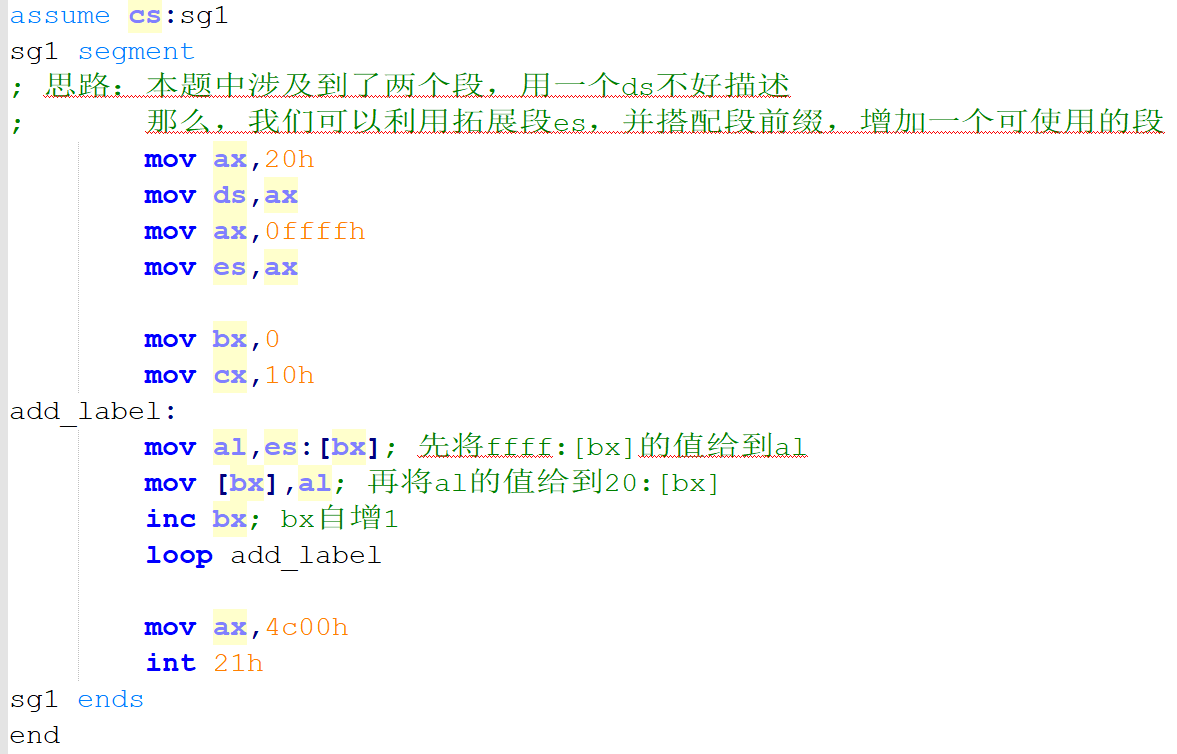
mov ax, ss:[bx]

**mov ax, es:[bx] （常用）**

案例：将ffff:0~ffff:f段空间的值拷贝到20:0~20:f中

案例分析：**本题中用到两个段，我们可以用ds、es配合段前缀来同时操纵两段内存。**

**令ds=20h，es=0ffffh；则ds:[bx]、es:[bx]就可以表示两个不同的段中偏移地址相同的内存空间。**



**2.loop指令和CX**

**（1）计数器寄存器CX**

CX（Counter Register）：计数器寄存器（常用来保存计数值，一般作为移位、循环、串操作等指令中的隐含计数器）。

8086CPU的CX是16位的计数器寄存器，**它常被用作隐含计数器，控制指令重复循环的次数。**如在loop循环指令中，CX的值用于控制循环次数；在shl、shr位移指令中，CX的值作为移动的位数；在movsb、movsw串操作指令中，CX的值用于指定要复制的字节数/字数。

**（2）loop指令**

语法：loop 标号

功能：①先让CX=CX-1

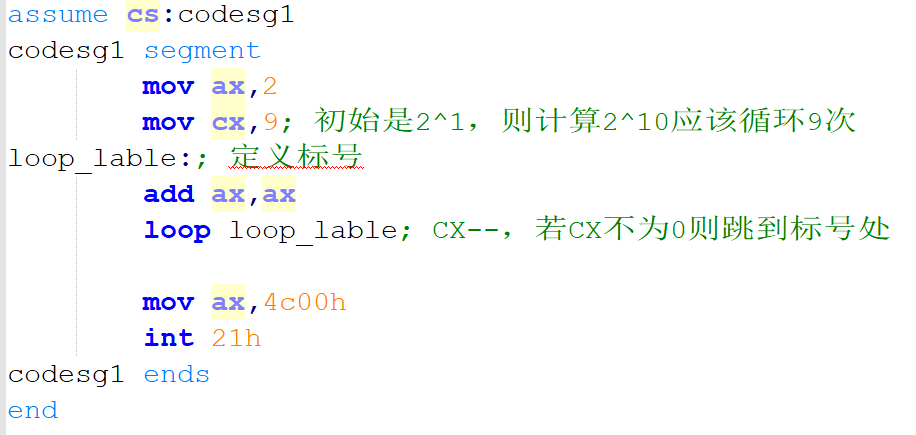
②判断CX中的值：若不为0则跳到标号处执行程序，若为0则向下执行。

注意事项：

①**CX是loop指令的隐含计数器，CX中的值决定了loop指令循环的次数**，因此在进入loop语句前要先设定好CX的值。

②**loop指令的标号记录了循环的开始位置**，因此该标号应该设置在loop指令的前几条语句处，这样才能保证循环的成立。

如下图，是利用loop循环计算2^10



（3）loop和[BX]的联合应用

有些时候，我们在处理内存数据时，也会使用到loop循环。

案例：计算ffff:0~ffff:b内存单元中的字节数据和，并将结果存放到dx中。

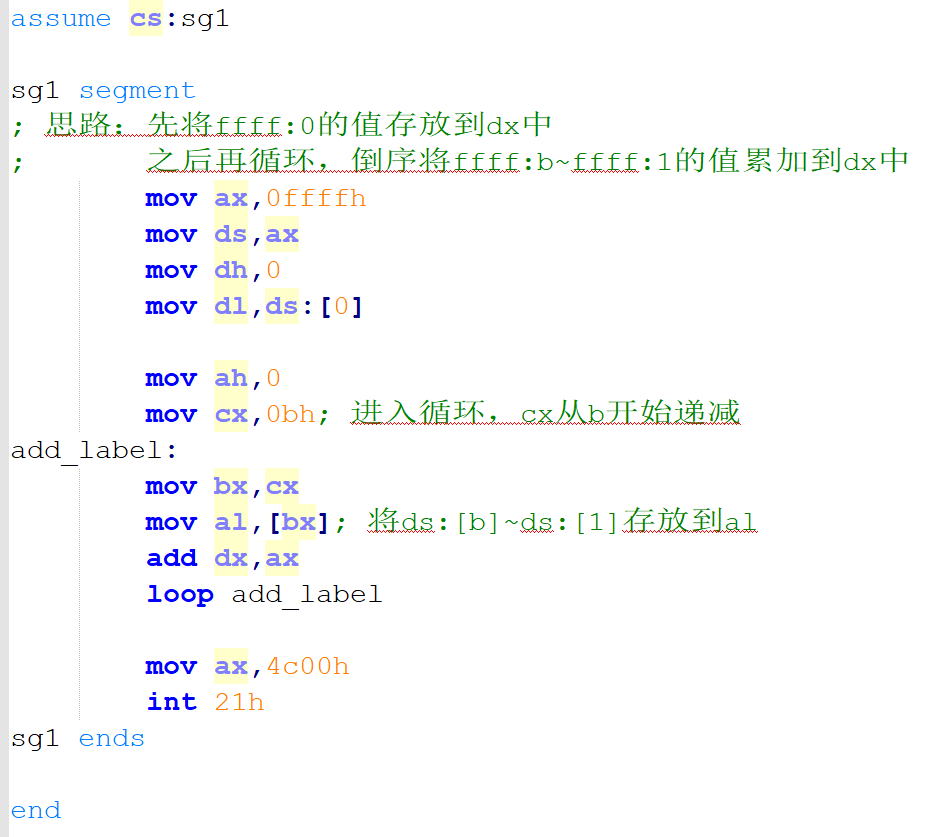
案例分析：

①分析存储范围：12个字节的和，总长度<=16，放入dx不会溢出。

②将16位的ax作为临时变量，存储内存中8位的字节：

高8位ah置为0；低8位al存储指定字节的值。

③利用loop，将连续的内存单元通过临时变量ax，累加到dx中



**3.Debug和.asm文件中指令格式的不同点**

①debug中的所有数字默认是16进制，但是.asm文件严格区分数字的进制。

**.asm文件中，数字默认是10进制，如123；若想表示十六进制数，需要在数字后面加h，如123h；若想表示二进制数，需要在数字后面加b，如11001001b**

注意：.adm文件中，为了区分数字和变量名，数字必须以0~9开头。如16进制数ffffh，必须写成0ffffh，否则报错。

②存储器直接寻址写法的区别

Debug中，[123]表示ds段中，偏移地址为123的内存空间。

.asm文件中，[123]被视为立即数123。若想在.asm文件中用存储器直接寻址的方式表示内存单元，必须为其添加段前缀“ds:”，写成ds:[123]的形式。

**4.一段安全的空间**

在8086模式中，随意向一段内存空间中写入内容是危险的，因为这段空间可能存放着重要的系统数据或者代码。

**绝对安全的空间：0:200~0:2FF**

这段空间是绝对安全的，其中存放的数据默认是0。当我们想向内存中写入数据时，就可以写入0:200~0:2FF这段空间。