UL: 0UL 表示无符号[长整型](https://so.csdn.net/so/search?q=%E9%95%BF%E6%95%B4%E5%9E%8B&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_44553164/article/details/_blank) 0

如果不写UL后缀，系统默认为：int, 即有符号整数。

.equ: 在8051汇编语言中，EQU用于定义一个符号常量，其效果不会带来存储空间的占用（这和C语言中定义变量有区别）。它仅仅是将一个常量（立即数）用一个符号来表示，即实现常量符号化的效果。汇编编译器在编译时会将这个符号替换为对应的常量。所以，EQU的作用和C语言中的宏常量十分类似。

.align: 地址对齐的伪指令，用来指定符号的对齐方式。.align的作用范围只限于紧跟它的那条指令或者数据，而接下来的指令或者数据的地址由上一条指令的地址和其长度决定。

不同平台的align 指令 有两种对齐方式

第一种：

a29k,hppa, m68k, m88k, w65, sparc, and Hitachi SH, and i386 的elf 格式的目标文件，直接按字节数对齐。如下：

.align 4 //按 4 个字节的倍数对齐下一个符号，空隙默认用0 来填充

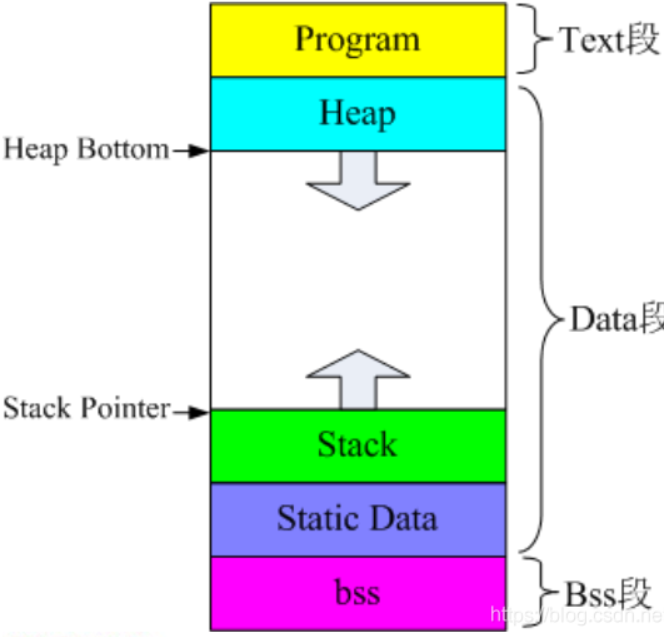
.align 4,0x90 //以4的倍数对齐地址，空隙用0x90 也就是nop 指令来填充

第二种：

For other systems, including the i386 using a.out format, and the arm and strongarm, 是按2的指数bit来对齐

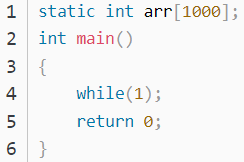
.align 5 //就是2的5次方对齐，也就是4字节对齐

.BSS:



在程序执行前，这块区域的大小就已经固定。值得注意的是，这块区域可以读写。

BSS段通常是指用来存放程序中未初始化的或者初始化为0的全局变量和静态变量。也就是说，只要初始值为0的类型，都会先放在这里，等到再次赋值时才会被取出。节省磁盘空间。



在什么都不做的情况下，你猜程序运行时 数组arr 的内存占用是多少？

int 是四个字节， 共有1000个元素sizeof(arr) = 4\*1000 = 4000B ≈ 4kB

对于数组 arr , bss段仅仅记录了变量名和数组大小，原来4kB大小的空间，现在只需记录两个变量即可，可能没有太多变化，但是，比如说游戏启动，这却是极大的提升，毕竟不是所有资源都需要立刻申请并赋值的，这就成为了提升的关键。

现在赋值 arr[999] = 1，操作系统做了这些事：

1. 从bss段中找到 arr 变量；
2. 到符号表中查找 arr 的类型大小，是4个字节；（符号表的可以用 map 实现，查找时间可忽略）
3. bss段将记录的 arr 的大小改为：4000-4.