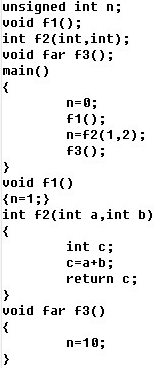
**程序1**

****

**1.程序运行时 n,a,b,c的段地址在哪个寄存器中？**

经验证 n为全局变量，段地址在DS寄存器。a,b是形参，其通过堆栈得到实参的值，段地址在ss寄存器。c为局部变量，当局部变量的定义少于等于两个则存放在si,di寄存器中，多的话会放在堆栈中，还有一种情况就是如果一个局部变量被多次使用也会放在si,di中的。所以c的段地址在ss寄存器中或者没有。

**2.函数返回值放在什么地方？**



JMP指令对应return

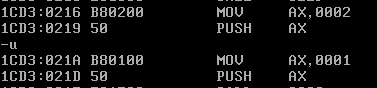


char型的返回值放在AL中，int型 的返回值放在AX中。Long int型返回值在AX,DX中。

**3.局部变量什么时候分配?什么时候释放？**

如果局部变量在自己函数里使用，则开始就去分配空间，函数结束前释放空间，如果没有使用则不分配内存空间。

**4.参数的存储空间什么时候分配，什么时候释放？**



调用有参数传递的函数前会去分配空间

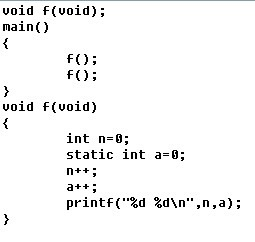


函数返回后释放空间。

**5.函数f3在调用与返回方式上与f1,f2有何不同？**

调用f3用 call 段地址+偏移地址 的方式，说明f3的调用支持不同段间的调用，距离可超过一个段的大小，而f1，f2的调用默认为本段，所以如果f1，f2与main函数不再一个段则会出现错误。对于返回，肯定是和你开始调用时相对应了。调用f3会进行 push cs,push ip, 则返回原地址则必须进行pop ip，pop cs。这就是为什么 call far[地址] 要和retf 相搭配使用的原因。

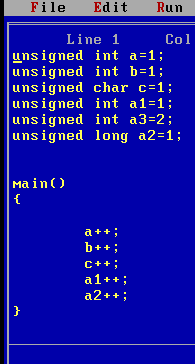
**程序2**

****

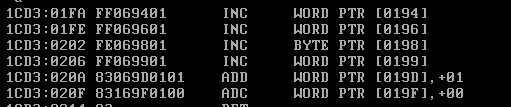
**6.静态变量的存储空间分配方式**

静态变量和全局变量一样都存放在全局变量区域。它的生命周期是从程序开始到程序结束。不过静态变量的作用域与此变量所在的位置有关，在函数内则只能在此函数能使用。

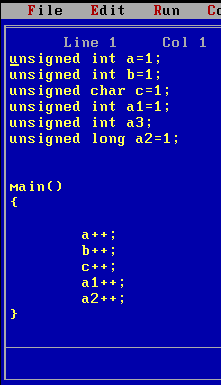
**程序3**

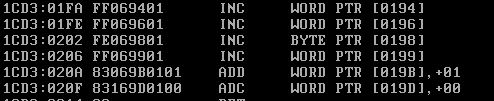
****

**7.程序中所有变量的存储空间相邻吗？各类型所占空间多大以及对运算方式的影响？**



从图上可以看出 所有变量的存储空间是相邻的。

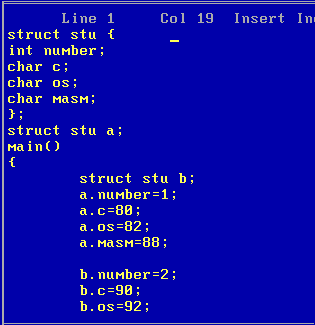


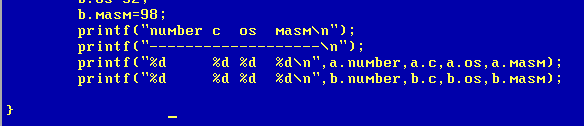


从图上可以看出，定义的全局变量定义了但是没有分配空间。所以没有使用的全局变量是不分配空间的。 和局部变量类似，没有赋值是不会分配空间的，虽然你定义了它们。

int型占2个字节，char型占1个字节，long 型占4个字节。对于int型 和char型一条基本的加减乘除指令就可以解决，而long型需要组合一次，比如加法，先add，在adc。这些是由8086基本指令集决定的。

**程序4**

****

****

**8.结构体变量和它们各个数据项的存储空间是如何分配的？**

结构体 a变量在外部声明，所以其分配在全局变量区，结构体b变量在函数内部声明，所以其分配在堆栈区。经查看汇编代码确实是这样。不过发现本来分配给结构体5个字节的内存空间，系统分配了6个字节空间。当我在结构体中在加一个char型的变量，发现系统分配还是6个字节。经查资料得在存储过程中，为了提高CPU的存储速度，编译器会对变量的起始地址做“对齐”处理。VC规定结构体的各变量存放的起始地址相对于结构体的起始地址的偏移量必须是该变量的类型所占字节数的倍数，并且整个结构体的字节数必须是该结构体中占用空间最大的类型的字节数的整数倍。验证如下：

struct stu{

char a1;

char a2;

char a3;

char a4;

long a5;

} 得到应该为8字节



经调试和猜想一样，继续验证

struct stu{

char a1;

long a5;

char a2;

char a3;

char a4;

}得到应该为4+4+3+1=12字节



经调试和猜想不一样，难道结构体的分配地址大小与编译环境的有关系？

其上的猜想是基于VC的编译环境规定。所以我继续大胆猜想tc2.0 为结构体分配地址大小是最大类型数据的整数倍。

验证如下：

struct stu{

char a1;

int a2;

char a3;

long a4;

};得到应该是1+2+1+4=8；刚好是整数倍，所以分配大小为8个字节。



结果与猜想一致

再验证：

struct stu{

char a1;

int a2;

char a3;

long a4;

char a5;

};得到应该是1+2+1+4+1=9；不是是整数倍，所以分配大小为12个字节.



结果与猜想不一致。

难道tc2.0 为结构体分配地址大小是int数据的整数倍?

struct stu{

char a1;

char a2;

long a4;

char a3;

};分配大小应为8个字节



结果与猜想一致

继续验证：

Struct stu{

char a1;

long a4;

};分配大小应为6个字节。



结果与猜想一致。

Struct stu{

char a1;

};分配大小应为2个字节。



结果与猜想一致

所以得出结论：tc2.0 为结构体分配地址大小是int数据的整数倍。

**程序5**

**8.向函数传递结构体数据是如何实现的？**

对函数进行了研究

main函数：

push bp

mov bp,sp

sub sp,+06 //为结构体分配内存空间

push si //为局部变量分配空间，用si 代表变量b

LEA bx,[bp-06] //取[bp-06]单元处的偏移地址

push ss

push bx

call 0266 //返回值为042a，即042a出存放结构体变量数据

push ds

push ax //保存，以供进行拷贝

mov cx,0006

call 1cd3:13ea //此函数将042a处的数据拷贝到堆栈中，即把值赋给我在主函数中定义的局部结构体变量

lea bx,[bp-06]

mov dx,ss

mov ax,bx //参数 ax放我们结构体变量的首地址，dx放段地址

mov cx,0006

call 1cd3:1406 //将要传递的结构体参数分配空间

call 0256 //将传递过去的值进行运算结果放到AX中返回

add sp,06

mov si,ax //将运算结果赋给变量b

0266处函数：

push bp

mov bp,sp

sub sp,+06 //为结构体分配空间

mov word ptr[bp-06],0001 //为结构体每个变量赋值

mov word ptr[bp-04],0002

mov word ptr[bp-02],0003

mov bx,042a //这是返回结构体的数据存放地址

push ds

push bx

lea bx,[bp-06] //取偏移地址

push ss

push bx

mov cx,0006

call 1cd3:13ea

mov ax,042a //地址放在AX中 返回

jmp 0292

mov sp,bp

pop bp

ret

1cd3:13ea处的函数

push bp

mov bp,sp

push si

push di

push ds

lds si,[bp+06] //取得[bp+06]偏移地址给si,[bp+07]处段地址给ds

les di,[bp+0a] //取得[bp+0a]偏移地址给di,[bp+0b]处段地址给es

cld // 按地址+的方向依次传送

shr cx,1 //

repz movsw

adc cx,cx

repz movsb

pop ds

pop di

pop si

pop bp

retf 0008

1cd3:1406处的函数：

pop bx

pop es

sub sp,cx //为传递的结构体分配6个空间

push bp

mov bp,sp

push si

push di

push ds //保护现场

mov ds,dx

mov si,ax // ds:si 指向实参结构体的首地址

mov dx,es

lea di,[bp+02]

mov ax,ss // ss:di指向形参的空间段地址：偏移地址 然后下来进行传送

mov es,ax

cld

shr cx,1

repz movsw

adc cx,cx

repz movsb

pop ds

pop di

pop si

pop bp

push dx

push bx

retf

0256处的函数

push bp

mov bp,sp

mov ax,[bp+04]

add ax,[bp+06]

mul word ptr [bp+08]

jmp 0264

pop bp

ret

结论：向函数传递结构体型数据 调用call 1cd3:1406

两个参数 DX 放我们结构体变量坐在的段地址

AX 放我们定义结构体的偏移地址

函数返回的结构体数据存储在042a处。