目录

[0929\_综合研究5研究报告 1](#_Toc9725)

[a.c 1](#_Toc12555)

[b.c 10](#_Toc18226)

[c.c 12](#_Toc30650)

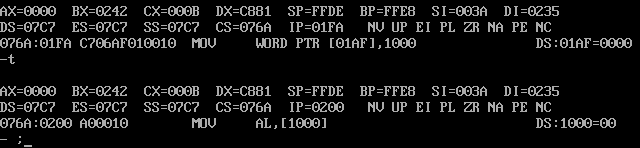
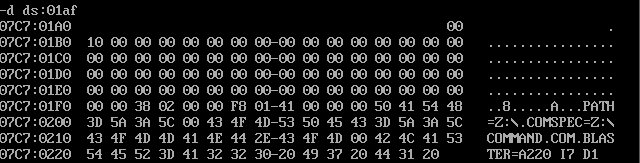
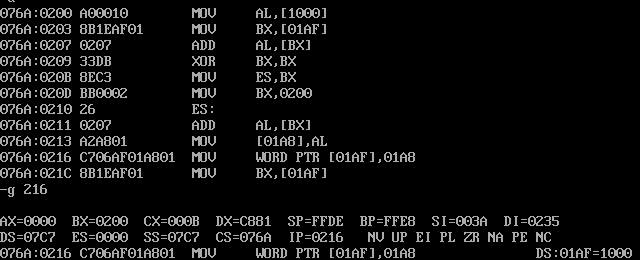
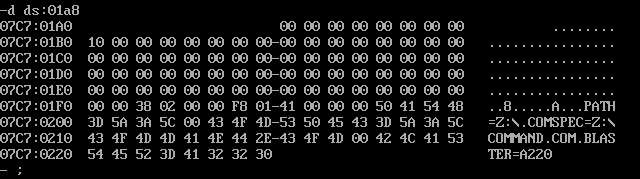
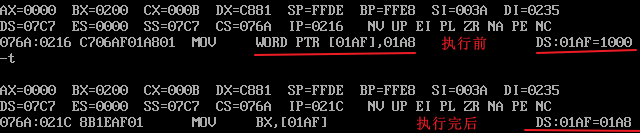
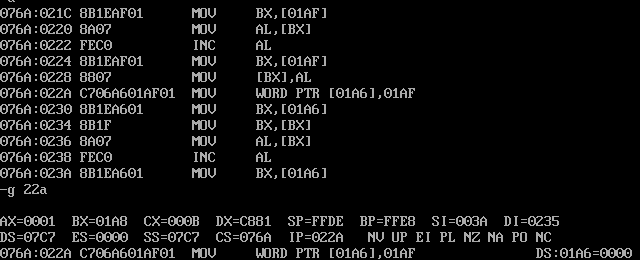
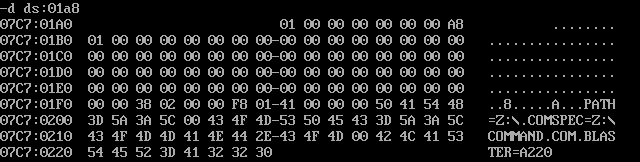
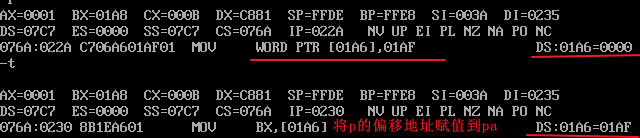
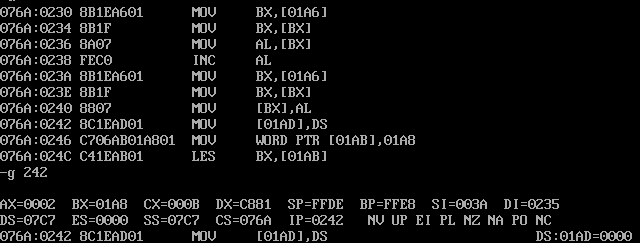
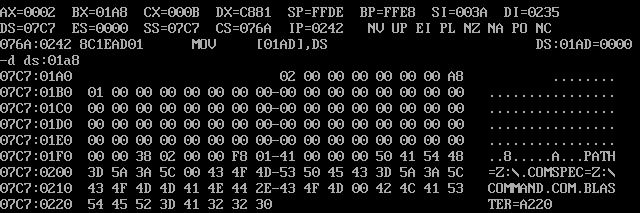
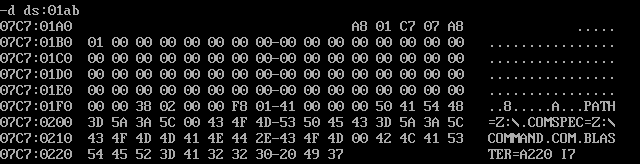
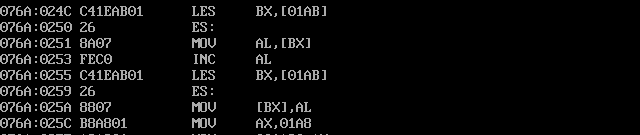
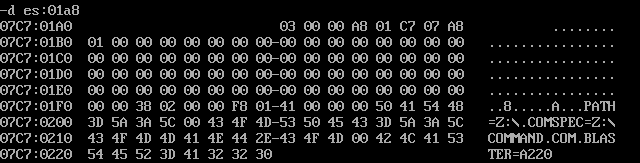
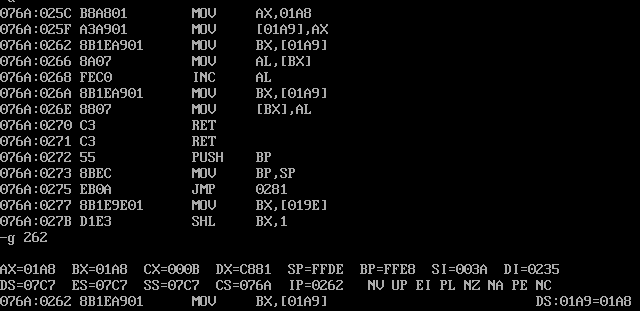
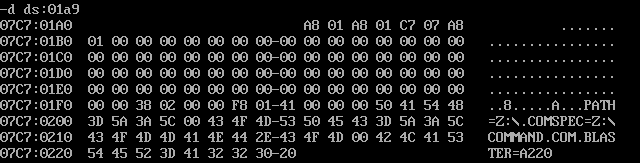
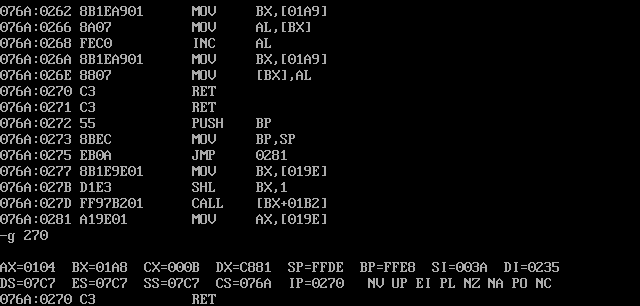
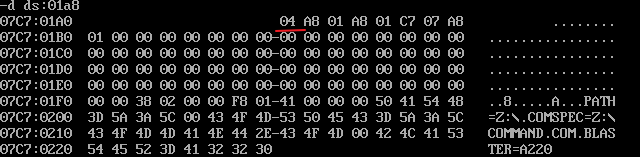
# 0929\_综合研究5研究报告

用debug 对一下程序进行分析，记录每一条c语句运行后，相关内存单元的值

## a.c

注意理解指针的机制，“\*\*” 和 “&” 运算的意义

char ch;  
char \*p;  
char \*\*pa;  
char far \*pf;  
int n;  
  
main() {  
 p = (unsigned char \*)0x1000;  
 ch = \*(unsigned char \*)0x1000 + \*p + \*(unsigned char far \*)0x200;  
  
 p = &ch;  
  
 \*p = \*p + 1;  
  
 pa = &p;  
 \*\*pa = \*\*pa + 1;  
  
 pf = (char far \*)&ch;  
 \*pf = \*pf + 1;  
  
 n = (int)&ch;  
 \*(char \*)n = \*(char \*)n + 1;  
}

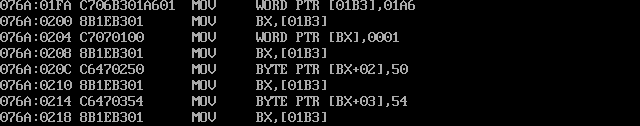
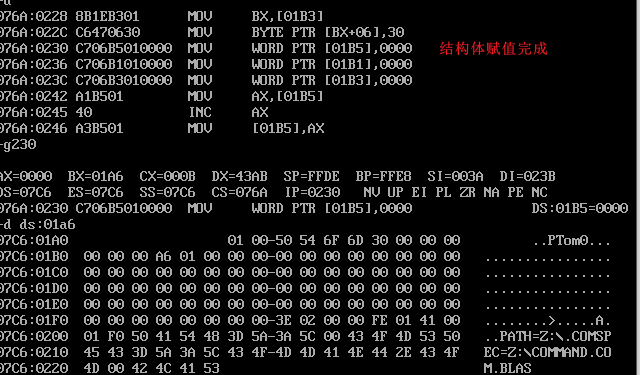
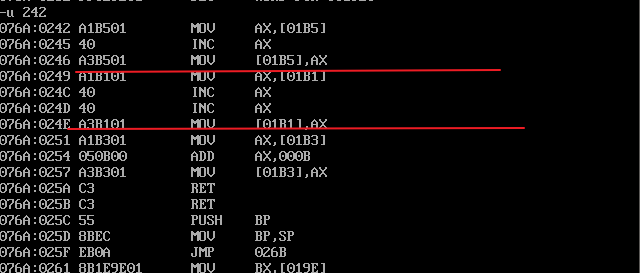
* 第一句 p = (unsigned char \*)0x1000;p在数据段中偏移地址为01af然后看内存中的值
* /\*mov word ptr DGROUP:\_p,4096\*/  
   p = (unsigned char \*)0x1000;
* 
* 
* 第二句 ch = \*(unsigned char \*)0x1000 + \*p + \*(unsigned char far \*)0x200;
* /\*  
   mov al,byte ptr [4096]  
   mov bx,word ptr DGROUP:\_p  
   add al,byte ptr [bx]  
   xor bx,bx  
   mov es,bx  
   mov bx,512  
   add al,byte ptr es:[bx]  
   mov byte ptr DGROUP:\_ch,al  
   \*/  
   ch = \*(unsigned char \*)0x1000 + \*p + \*(unsigned char far \*)0x200;
* 
* 
* 第三句 p = &ch;
* /\* [01af] 01a8\*/  
   /\*mov word ptr DGROUP:\_p,offset DGROUP:\_ch\*/  
   p = &ch;
* 
* 第四句 \*p = \*p + 1;
* /\*  
   mov bx,word ptr DGROUP:\_p  
   mov al,byte ptr [bx]  
   inc al  
   mov bx,word ptr DGROUP:\_p  
   mov byte ptr [bx],al  
   \*/  
   \*p = \*p + 1;
* 
* 可以看到p指向的内存中的值增加一
* 
* 第五句 pa = &p;
* /\* ds:[01a6] 01af  
   mov word ptr DGROUP:\_pa,offset DGROUP:\_p  
   \*/  
   pa = &p;
* 
* 第六句 \*\*pa = \*\*pa + 1;
* /\*  
   mov bx,word ptr DGROUP:\_pa  
   mov bx,word ptr [bx]  
   mov al,byte ptr [bx]  
   inc al  
   mov bx,word ptr DGROUP:\_pa  
   mov bx,word ptr [bx] bx=01a6  
   mov byte ptr [bx],al  
   \*/  
   \*\*pa = \*\*pa + 1;
* 
* 可以看到p指向的内存中的值增加一
* 
* 第七句 pf = (\*\*char\*\* far \*)&ch;
* /\* 01ad  
   mov word ptr DGROUP:\_pf+2,ds  
   [01a8] 01a8  
   mov word ptr DGROUP:\_pf,offset DGROUP:\_ch  
   \*/  
   pf = (char far \*)&ch;
* 
* 可以看到pf存在的是ch的地址
* 
* 第八句 \*pf = \*pf + 1;
* /\*  
   les bx,dword ptr DGROUP:\_pf  
   mov al,byte ptr es:[bx]  
   inc al  
   les bx,dword ptr DGROUP:\_pf  
   mov byte ptr es:[bx],al  
   \*/  
   \*pf = \*pf + 1;
* 
* 可以看到p指向的内存中的值增加一
* 
* 第九句 n = (\*\*int\*\*)&ch;
* /\* 01a8  
   mov ax,offset DGROUP:\_ch  
   01a9  
   mov word ptr DGROUP:\_n,ax  
   \*/  
   n = (int)&ch;
* 
* 
* 第十句 \*(\*\*char\*\* \*)n = \*(\*\*char\*\* \*)n + 1;
* /\*  
   mov bx,word ptr DGROUP:\_n  
   mov al,byte ptr [bx]  
   inc al  
   mov bx,word ptr DGROUP:\_n  
   mov byte ptr [bx],al  
   \*/  
   \*(char \*)n = \*(char \*)n + 1;
* 
* 

综上可以看出\*p的功能是取出以p中数据作为偏移地址的内存中的值，&p的功能就是取出p的偏移地址

## b.c

注意理解 struct指针的用法，指针“＋”运算的意义。

typedef struct {  
 int number;  
 char c;  
 char name[8];  
} stu;  
  
stu a;  
  
char \*pchar;  
int \*pint;  
stu \*pstu;  
  
main() {  
 pstu = &a;  
  
 pstu->number = 1;  
 (\*pstu).c = 80;  
 pstu->name[0] = 'T';  
 pstu->name[1] = 'o';  
 (\*pstu).name[2] = 'm';  
 (\*pstu).name[3] = '0';  
  
 pchar = 0;  
 pint = 0;  
 pstu = 0;  
  
 pchar = pchar + 1;  
 pint = pint + 1;  
 pstu = pstu + 1;  
}

* 通过汇编代码可以得出不论是指针的-> 运算和. 运算最后翻译成的汇编都是把stu 的首地址传给bx然后通过bx加上偏移来访问结构体变量的真正内存地址
* 
* 
* 字符型指针加一就把指针内存中的数据增加一，整形指针加一就把指针内存中的数据增加二，结构体指针加一就是把指针内存中的数据增加结构体中各个变量长度总和。
* 

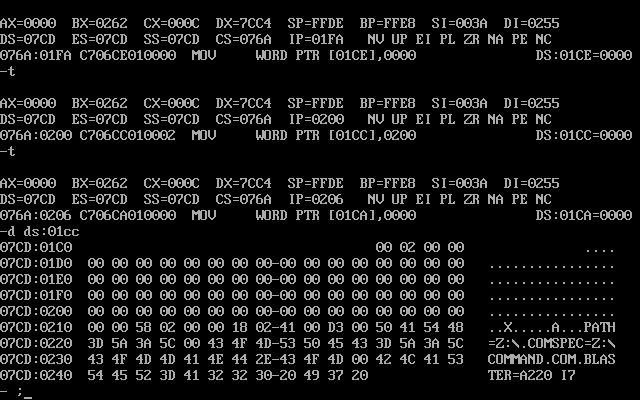
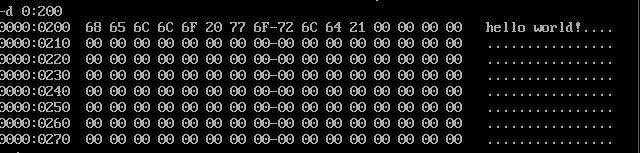
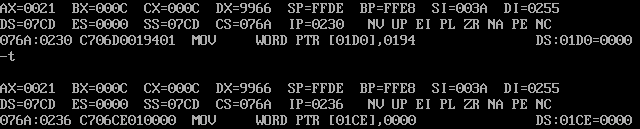
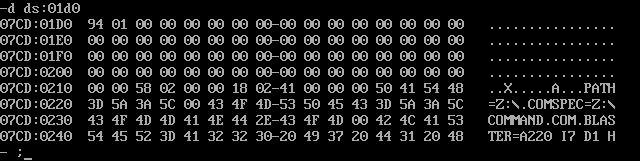
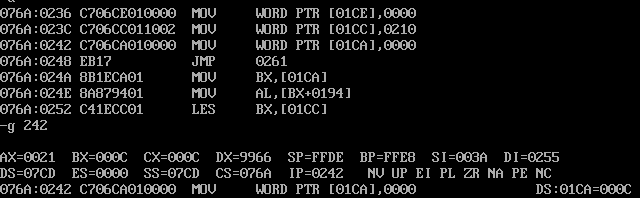
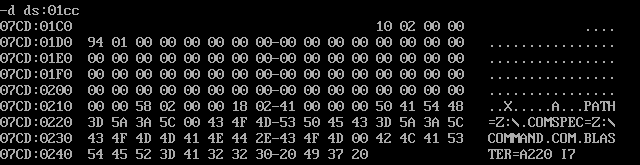
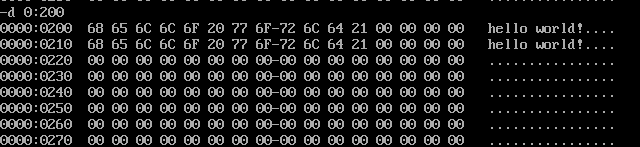
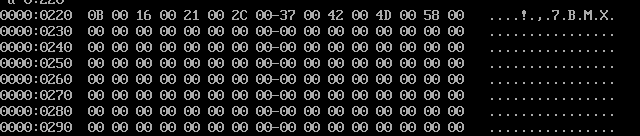
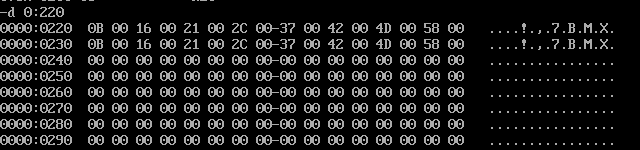
## c.c

将字符串“hello world！”分别拷贝到从0:200、:210起始的内存中:将数组a分别拷贝到0:220、0:230起始的内存中。

注意理解“［］”运算的意义及数组名与指针的关系。

假设p是一个指针，p［n］的意义等同于＊（p＋n）

char \*p;  
char far \*pf;  
char str[20] = "hello world!";  
int a[8] = {11, 22, 33, 44, 55, 66, 77, 88};  
int n;  
  
main() {  
 pf = (char far \*)0x200;  
 for (n = 0; str[n]; n++)  
 \*(pf + n) = str[n];  
  
 p = str;  
 pf = (char far \*)0x210;  
 for (n = 0; p[n]; n++)  
 pf[n] = \*(str + n);  
  
 for (n = 0; n < 8; n++)  
 ((int far \*)0x220)[n] = \*(a + n);  
 for (n = 0; n < 8; n++)  
 \*(int far \*)(0x230 + n \* 2) = \*(&a[0] + n);  
}

* 1
* /\*  
   mov word ptr DGROUP:\_pf+2,0  
   mov word ptr DGROUP:\_pf,512  
  \*/  
  pf = (char far \*)0x200;
* 
* 2
* /\*  
  ; ?debug L 9 [01ca]  
   mov word ptr DGROUP:\_n,0  
   jmp short @5  
  @4:  
  ; ?debug L 10  
   mov bx,word ptr DGROUP:\_n ;[01ca]  
   mov al,byte ptr DGROUP:\_str[bx] ;[bx+0194]  
   les bx,dword ptr DGROUP:\_pf  
   add bx,word ptr DGROUP:\_n  
   mov byte ptr es:[bx],al  
  @3:  
   inc word ptr DGROUP:\_n  
  @5: [01ca]  
   mov bx,word ptr DGROUP:\_n  
   cmp byte ptr DGROUP:\_str[bx],0  
   jne @4  
  \*/  
  for (n = 0; str[n]; n++)  
   \*(pf + n) = str[n];
* 初始化后
* 
* 3
* /\* 01d0 0194  
   mov word ptr DGROUP:\_p,offset DGROUP:\_str  
  \*/  
  p = str;
* 
* 
* 4
* /\*  
   mov word ptr DGROUP:\_pf+2,0  
   mov word ptr DGROUP:\_pf,528  
  \*/  
    
  pf = (char far \*)0x210;
* 
* 
* 5
* /\*  
  ; ?debug L 14  
   mov word ptr DGROUP:\_n,0  
   jmp short @9  
  @8:  
  ; ?debug L 15  
   mov bx,word ptr DGROUP:\_n  
   mov al,byte ptr DGROUP:\_str[bx]  
   les bx,dword ptr DGROUP:\_pf  
   add bx,word ptr DGROUP:\_n  
   mov byte ptr es:[bx],al  
  @7:  
   inc word ptr DGROUP:\_n  
  @9:  
   mov bx,word ptr DGROUP:\_p  
   add bx,word ptr DGROUP:\_n  
   cmp byte ptr [bx],0  
   jne @8  
  \*/  
  for (n = 0; p[n]; n++)  
   pf[n] = \*(str + n);
* 拷贝hello world从0:200 -> 0:210
* 
* 6
* /\*  
  @6:  
  ; ?debug L 17  
   mov word ptr DGROUP:\_n,0  
   jmp short @13  
  @12:  
  ; ?debug L 18  
   mov bx,word ptr DGROUP:\_n  
   shl bx,1  
   mov ax,word ptr DGROUP:\_a[bx]  
   mov dx,word ptr DGROUP:\_n  
   shl dx,1  
   xor bx,bx  
   mov es,bx  
   mov bx,544  
   add bx,dx  
   mov word ptr es:[bx],ax  
  @11:  
   inc word ptr DGROUP:\_n  
  @13:  
   cmp word ptr DGROUP:\_n,8  
   jl @12  
  \*/  
  for (n = 0; n < 8; n++)  
   ((int far \*)0x220)[n] = \*(a + n);
* 初始化
* 
* 7
* /\*  
  @10:  
  ; ?debug L 19  
   mov word ptr DGROUP:\_n,0  
   jmp short @17  
  @16:  
  ; ?debug L 20  
   mov bx,word ptr DGROUP:\_n  
   shl bx,1  
   mov ax,word ptr DGROUP:\_a[bx]  
   push ax  
   mov ax,word ptr DGROUP:\_n  
   shl ax,1  
   add ax,560  
   cwd   
   mov bx,ax  
   mov es,dx  
   pop ax  
   mov word ptr es:[bx],ax  
  @15:  
   inc word ptr DGROUP:\_n  
  @17:  
   cmp word ptr DGROUP:\_n,8  
   jl @16  
  \*/  
  for (n = 0; n < 8; n++)  
   \*(int far \*)(0x230 + n \* 2) = \*(&a[0] + n);
* 拷贝a从0:220->0:230
* 
* 综上p[n]的意思是访问以p为基地址n为偏移地址中的数据，数组名p和指针\*p存储的都是数据的起始地址