- 综合研究2

```
1    1)    输出的段地址和在debug中输出的段地址之不一样    @已解决
2    **2）通过函数名直接打印出段地址和偏移地址（不使用_CS）。 @已解决**
```

- 综合研究3

```
1    1）程序1 ①中的问题（除全局变量存储空间分配和释放）要根据具体的代码，分析得出答案。
```

@ 已解决

```
1      2)**写程序验证全局变量存储空间是加载时分配，还是c0s分配。   @未解决**
2
3      3）程序5首先要找到每条C语句对应的汇编代码，然后单步跟踪，跟踪时要将栈中的情况都画出来（这部分自己纸上画就可以，不必体现在研究报告中，但是必须通过这个过程去完全理解程序）。跟踪一遍后，回答教材中的两个问题。   @未解决
```

0928_综合研究1-3补充研究报告

综合研究2

> 通过函数名直接打印出段地址和偏移地址（不使用_CS）

通过查阅相关资料可以尝试把函数名前强制转换成长整型

```
1   int a;
2   void f1(void) { a = 1; }
3   void f2(void) { a = 2; }
4   void f3(void) { a = 3; }
5   main() {
6       char *string = "--------------------";
7       printf("\nCS: %x\n", _CS);
8       printf("%s", string);
9       printf("\nf1: %lx\n", (long)f1);
10      printf("\nf2: %lx\n", (long)f2);
11      printf("\nf3: %lx\n", (long)f3);
12  }
```

```
C:\>\src\TWO\A.EXE

CS: 1a2
--------------------
f1: 1a201fa

f2: 1a20201

f3: 1a20208

C:\>DEBUG.EXE \SRC\TWO\A.EXE
-g

CS: 76a
--------------------
f1: 76a01fa

f2: 76a0201

f3: 76a0208

Program terminated normally
```

通过 `cs` 输出的数字可以检验出 强制转换可以正确输出函数对应的段地址和偏移地址。

综合研究3

> 写程序验证全局变量存储空间是加载时分配，还是c0s分配。 @未解决

- 编写测试程序

```
int b = 9;
main() {
    static int a = 8;
    a++;
    b++;
}
```

- 编译链接生成可执行程序通过debug查看变量在数据段的偏移地址

```
076A:0203 C3            RET
076A:0204 55            PUSH    BP
076A:0205 8BEC          MOV     BP,SP
076A:0207 EB0A          JMP     0213
076A:0209 8B1EA201      MOV     BX,[01A2]
076A:020D D1E3          SHL     BX,1
076A:020F FF97AA01      CALL    [BX+01AA]
076A:0213 A1A201        MOV     AX,[01A2]
076A:0216 FF0EA201      DEC     WORD PTR [01A2]
-g 0202

AX=0000  BX=023A  CX=0007  DX=27EA  SP=FFDE  BP=FFE8  SI=003A  DI=022D
DS=07C0  ES=07C0  SS=07C0  CS=076A  IP=0202   NV UP EI PL NZ NA PE NC
076A:0202 C3            RET
-d ds:0194
07C0:0190              0A 00 09 00-03 02 03 02 03 02 00 00       ............
07C0:01A0  00 10 00 00 D2 01 D2 01-D9 01 00 00 00 00 00 00   ................
```

- 可以看见存储全局变量的段地址和偏移地址，然后在重新debug后直接跳转到相应的位置查看

```
C:\>DEBUG.EXE \SRC\THREE\A.EXE
-d 07c0:0194
07C0:0190                09 00 08 00-03 02 03 02 03 02 00 00      ............
07C0:01A0  00 10 00 00 D2 01 D2 01-D9 01 00 00 00 00 00 00   ................
07C0:01B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
07C0:01C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
07C0:01D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
07C0:01E0  00 00 00 00 00 00 00 00-00 00 30 02 00 00 F0 01   ..........0.....
07C0:01F0  41 00 C6 47 50 41 54 48-3D 5A 3A 5C 00 43 4F 4D   A..GPATH=Z:\.COM
07C0:0200  53 50 45 43 3D 5A 3A 5C-43 4F 4D 4D 41 4E 44 2E   SPEC=Z:\COMMAND.
07C0:0210  43 4F 4D 00                                       COM.
-_;
```

- 可以推断全局变量是在程序加载时给全局变量分配空间

> 程序5首先要找到每条C语句对应的汇编代码，然后单步跟踪，跟踪时要将栈中的情况都画出来（这部分自己纸上画就可以，不必体现在研究报告中，但是必须通过这个过程去完全理解程序）。跟踪一遍后，回答教材中的两个问题。 @未解决

- LEA 取有效地址指令

```
-g 201

AX=0000  BX=04C0  CX=0007  DX=C405  SP=FFCE  BP=FFD6  SI=003A  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=0201   NV UP EI NG NZ NA PO NC
076A:0201 8D5EFA        LEA     BX,[BP-06]                      SS:FFD0=0010
-t

AX=0000  BX=FFD0  CX=0007  DX=C405  SP=FFCE  BP=FFD6  SI=003A  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=0204   NV UP EI NG NZ NA PO NC
076A:0204 16            PUSH    SS
```

- 程序的汇编代码

```
-u
076A:01FA 55            PUSH    BP
076A:01FB 8BEC          MOV     BP,SP
076A:01FD 83EC06        SUB     SP,+06
076A:0200 56            PUSH    SI
076A:0201 8D5EFA        LEA     BX,[BP-06]
076A:0204 16            PUSH    SS
076A:0205 53            PUSH    BX
076A:0206 E85D00        CALL    0266
076A:0209 1E            PUSH    DS
076A:020A 50            PUSH    AX
076A:020B B90600        MOV     CX,0006
076A:020E 9AEA136A07    CALL    076A:13EA
076A:0213 8D5EFA        LEA     BX,[BP-06]
076A:0216 8CD2          MOV     DX,SS
076A:0218 8BC3          MOV     AX,BX
```

```
076A:021A B90600        MOV     CX,0006
076A:021D 9A06146A07    CALL    076A:1406
076A:0222 E83100        CALL    0256
076A:0225 83C406        ADD     SP,+06
076A:0228 8BF0          MOV     SI,AX
076A:022A 56            PUSH    SI
076A:022B B89401        MOV     AX,0194
076A:022E 50            PUSH    AX
076A:022F E81809        CALL    0B4A
076A:0232 59            POP     CX
076A:0233 59            POP     CX
076A:0234 E82F00        CALL    0266
076A:0237 8CDA          MOV     DX,DS
076A:0239 B90600        MOV     CX,0006
```

```
076A:023C 9A06146A07    CALL    076A:1406
076A:0241 E81200        CALL    0256
076A:0244 83C406        ADD     SP,+06
076A:0247 50            PUSH    AX
076A:0248 B89801        MOV     AX,0198
076A:024B 50            PUSH    AX
076A:024C E8FB08        CALL    0B4A
076A:024F 59            POP     CX
076A:0250 59            POP     CX
076A:0251 5E            POP     SI
076A:0252 8BE5          MOV     SP,BP
076A:0254 5D            POP     BP
076A:0255 C3            RET
076A:0256 55            PUSH    BP
076A:0257 8BEC          MOV     BP,SP
076A:0259 8B4604        MOV     AX,[BP+04]
```

- 执行 `call 266` 可以看到把结构体的数据的首地址偏移地址和要返回结果的段地址的偏移地址存储到栈中然后调用 `call 076a:13ea`

```
076A:0266 55            PUSH    BP
076A:0267 8BEC          MOV     BP,SP
076A:0269 83EC06        SUB     SP,+06
076A:026C C746FA0100    MOV     WORD PTR [BP-06],0001
076A:0271 C746FC0200    MOV     WORD PTR [BP-04],0002
076A:0276 C746FE0300    MOV     WORD PTR [BP-02],0003
076A:027B BB2A04        MOV     BX,042A
076A:027E 1E            PUSH    DS          目的地址
076A:027F 53            PUSH    BX
076A:0280 8D5EFA        LEA     BX,[BP-06]
076A:0283 16            PUSH    SS          源地址
076A:0284 53            PUSH    BX
076A:0285 B90600        MOV     CX,0006
```

```
076A:0288 9AEA136A07    CALL    076A:13EA
076A:028D B82A04        MOV     AX,042A
076A:0290 EB00          JMP     0292
076A:0292 8BE5          MOV     SP,BP
076A:0294 5D            POP     BP
076A:0295 C3            RET
076A:0296 55            PUSH    BP
076A:0297 8BEC          MOV     BP,SP
076A:0299 56            PUSH    SI
076A:029A 8B7604        MOV     SI,[BP+04]
076A:029D 0BF6          OR      SI,SI
076A:029F 7C14          JL      02B5
076A:02A1 83FE58        CMP     SI,+58
076A:02A4 7603          JBE     02A9
076A:02A6 BE5700        MOV     SI,0057
```

  - `call 076a:13ea`

> 执行 REP MOVS 之前，应先做好：
>
> (1) 源串首地址（末地址）→ SI
>
> (2) 目的串首地址（末地址）→ DI
>
> (3) 串长度 → CX
>
> (4) 建立方向标志
>
> （ CLD 使 DF=0，STD 使 DF=1 ）

```
076A:13EA 55            PUSH    BP
076A:13EB 8BEC          MOV     BP,SP
076A:13ED 56            PUSH    SI
076A:13EE 57            PUSH    DI
076A:13EF 1E            PUSH    DS
076A:13F0 C57606        LDS     SI,[BP+06]
076A:13F3 C47E0A        LES     DI,[BP+0A]
076A:13F6 FC            CLD
076A:13F7 D1E9          SHR     CX,1
076A:13F9 F3            REPZ
076A:13FA A5            MOVSW
076A:13FB 13C9          ADC     CX,CX
076A:13FD F3            REPZ
076A:13FE A4            MOVSB
076A:13FF 1F            POP     DS
076A:1400 5F            POP     DI
076A:1401 5E            POP     SI
076A:1402 5D            POP     BP
076A:1403 CA0800        RETF    0008
076A:1406 5B            POP     BX
076A:1407 07            POP     ES
076A:1408 2BE1          SUB     SP,CX
```

- 执行到 `076a:13f0 c57606` 可以看到传入栈中的结构体数据的源地址和要返回结果的目的地址

```
AX=0000  BX=FFC0  CX=0006  DX=4CE7  SP=FFAC  BP=FFB2  SI=003A  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=13F0   NV UP EI NG NZ NA PE NC
076A:13F0 C57606        LDS     SI,[BP+06]          源地址          SS:FFB8=FFC0
-t

AX=0000  BX=FFC0  CX=0006  DX=4CE7  SP=FFAC  BP=FFB2  SI=FFC0  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=13F3   NV UP EI NG NZ NA PE NC
076A:13F3 C47E0A        LES     DI,[BP+0A]          目的地址        SS:FFBC=042A
```

- 执行到 `adc cx,cx` 前后查看目的地址中的数据可以发现数据已经由源地址传送到目的地址

```
AX=0000  BX=FFC0  CX=0006  DX=4CE7  SP=FFAC  BP=FFB2  SI=FFC0  DI=042A
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=13F7   NV UP EI NG NZ NA PE NC
076A:13F7 D1E9          SHR     CX,1
-t

AX=0000  BX=FFC0  CX=0003  DX=4CE7  SP=FFAC  BP=FFB2  SI=FFC0  DI=042A
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=13F9   NV UP EI PL NZ AC PE NC
076A:13F9 F3            REPZ
076A:13FA A5            MOVSW
-p

AX=0000  BX=FFC0  CX=0000  DX=4CE7  SP=FFAC  BP=FFB2  SI=FFC6  DI=0430
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=13FB   NV UP EI PL NZ AC PE NC
076A:13FB 13C9          ADC     CX,CX
-d ds:042a
08AD:0420                                     01 00 02 00 03 00            ......
08AD:0430  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
08AD:0440  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
08AD:0450  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
08AD:0460  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
08AD:0470  B6 04 00 00 76 04 41 00-9A DB 50 41 54 48 3D 5A   ....v.A...PATH=Z
08AD:0480  3A 5C 00 43 4F 4D 53 50-45 43 3D 5A 3A 5C 43 4F   :\.COMSPEC=Z:\CO
08AD:0490  4D 4D 41 4E 44 2E 43 4F-4D 00 42 4C 41 53 54 45   MMAND.COM.BLASTE
08AD:04A0  52 3D 41 32 32 30 20 49-37 20                     R=A220 I7
- ;_
```

- 从函数返回的结构体数据类型存储在数据段中，在函数中先把返回结果的段地址和偏移地址压入栈中然后把结构体的段地址和偏移地址压入栈中，接着调用子程序（参数就是两个偏移地址）把栈中的数据移动到目标地址的内存中，子程序完成返回目标地址的偏移地址。
- 执行到 `call 076a:1406`
    - 可以看到 `dx:ax` 为 `076a:1406` 的参数

```
-t

AX=042A  BX=FFD0  CX=0000  DX=DBD7  SP=FFCE  BP=FFD6  SI=003A  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=0216   NV UP EI PL ZR NA PE NC
076A:0216 8CD2          MOV     DX,SS      段地址
-t

AX=042A  BX=FFD0  CX=0000  DX=08AD  SP=FFCE  BP=FFD6  SI=003A  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=0218   NV UP EI PL ZR NA PE NC
076A:0218 8BC3          MOV     AX,BX     偏移地址
-t

AX=FFD0  BX=FFD0  CX=0000  DX=08AD  SP=FFCE  BP=FFD6  SI=003A  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=021A   NV UP EI PL ZR NA PE NC
076A:021A B90600        MOV     CX,0006
-t

AX=FFD0  BX=FFD0  CX=0006  DX=08AD  SP=FFCE  BP=FFD6  SI=003A  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=021D   NV UP EI PL ZR NA PE NC
076A:021D 9A06146A07    CALL    076A:1406
-d 08ad:ffd0    dx:ax
08AD:FFD0  01 00 02 00 03 00 E2 FF-1D 01 01 00 E0 FF BA 04   ................
08AD:FFE0  E4 FF 00 00 43 3A 5C 53-52 43 5C 54 48 52 45 45   ....C:\SRC\THREE
08AD:FFF0  5C 54 48 52 45 45 35 2E-45 58 45 00 F8 00 FB 00   \THREE5.EXE.....
- ;
```

- 子程序 `076a:1460`

```
076A:1406 5B            POP     BX
076A:1407 07            POP     ES
076A:1408 2BE1          SUB     SP,CX
076A:140A 55            PUSH    BP
076A:140B 8BEC          MOV     BP,SP
076A:140D 56            PUSH    SI
076A:140E 57            PUSH    DI
076A:140F 1E            PUSH    DS
076A:1410 8EDA          MOV     DS,DX   源地址
076A:1412 8BF0          MOV     SI,AX
076A:1414 8CC2          MOV     DX,ES          目标地址
076A:1416 8D7E02        LEA     DI,[BP+02]
076A:1419 8CD0          MOV     AX,SS
076A:141B 8EC0          MOV     ES,AX
076A:141D FC            CLD
076A:141E D1E9          SHR     CX,1
076A:1420 F3            REPZ
076A:1421 A5            MOVSW
076A:1422 13C9          ADC     CX,CX
076A:1424 F3            REPZ
076A:1425 A4            MOVSB
```

```
076A:1426 1F            POP     DS
076A:1427 5F            POP     DI
076A:1428 5E            POP     SI
076A:1429 5D            POP     BP
076A:142A 52            PUSH    DX
076A:142B 53            PUSH    BX
076A:142C CB            RETF
```

  - 可以看到此程序的功能是把结构体数据复制到目的地址

```
AX=08AD  BX=0222  CX=0006  DX=076A  SP=FFC0  BP=FFC6  SI=FFD0  DI=FFC8
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=141E   NV UP EI NG NZ NA PO NC
076A:141E D1E9          SHR     CX,1
-t

AX=08AD  BX=0222  CX=0003  DX=076A  SP=FFC0  BP=FFC6  SI=FFD0  DI=FFC8
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=1420   NV UP EI PL NZ AC PE NC
076A:1420 F3            REPZ
076A:1421 A5            MOVSW
-p

AX=08AD  BX=0222  CX=0000  DX=076A  SP=FFC0  BP=FFC6  SI=FFD6  DI=FFCE
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=1422   NV UP EI PL NZ AC PE NC
076A:1422 13C9          ADC     CX,CX
-d ds:ffc8
08AD:FFC0                               01 00 02 00 03 00 3A 00          ......:.
08AD:FFD0  01 00 02 00 03 00 E2 FF-1D 01 01 00 E0 FF BA 04   ................
08AD:FFE0  E4 FF 00 00 43 3A 5C 53-52 43 5C 54 48 52 45 45   ....C:\SRC\THREE
08AD:FFF0  5C 54 48 52 45 45 35 2E-45 58 45 00 F8 00 FB 00   \THREE5.EXE.....
```

- 执行 `call 0256`

```
076A:0256 55            PUSH    BP
076A:0257 8BEC          MOV     BP,SP
076A:0259 8B4604        MOV     AX,[BP+04]
076A:025C 034606        ADD     AX,[BP+06]
076A:025F F76608        MUL     WORD PTR [BP+08]
076A:0262 EB00          JMP     0264
076A:0264 5D            POP     BP
076A:0265 C3            RET
```

可以看到子程序通过栈来使用传过来的结构体数据

```
AX=08AD  BX=0222  CX=0000  DX=076A  SP=FFC4  BP=FFD6  SI=003A  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=0257   NV UP EI PL ZR NA PE NC
076A:0257 8BEC          MOV     BP,SP
-t

AX=08AD  BX=0222  CX=0000  DX=076A  SP=FFC4  BP=FFC4  SI=003A  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=0259   NV UP EI PL ZR NA PE NC
076A:0259 8B4604        MOV     AX,[BP+04]                      SS:FFC8=0001
-t

AX=0001  BX=0222  CX=0000  DX=076A  SP=FFC4  BP=FFC4  SI=003A  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=025C   NV UP EI PL ZR NA PE NC
076A:025C 034606        ADD     AX,[BP+06]                      SS:FFCA=0002
-t

AX=0003  BX=0222  CX=0000  DX=076A  SP=FFC4  BP=FFC4  SI=003A  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=025F   NV UP EI PL NZ NA PE NC
076A:025F F76608        MUL     WORD PTR [BP+08]                SS:FFCC=0003
-t

AX=0009  BX=0222  CX=0000  DX=0000  SP=FFC4  BP=FFC4  SI=003A  DI=04B3
DS=08AD  ES=08AD  SS=08AD  CS=076A  IP=0262   NV UP EI PL NZ NA PE NC
076A:0262 EB00          JMP     0264
-_;
```

- 向函数传递结构体数据
    - 通过调用子程序来把结构体数据复制到栈中
    - 然后函数在通过栈来使用结构体数据