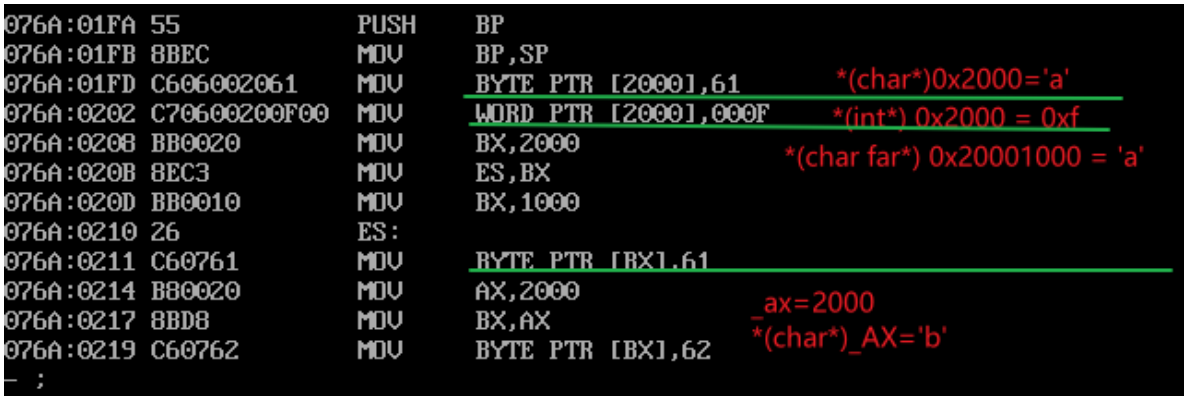
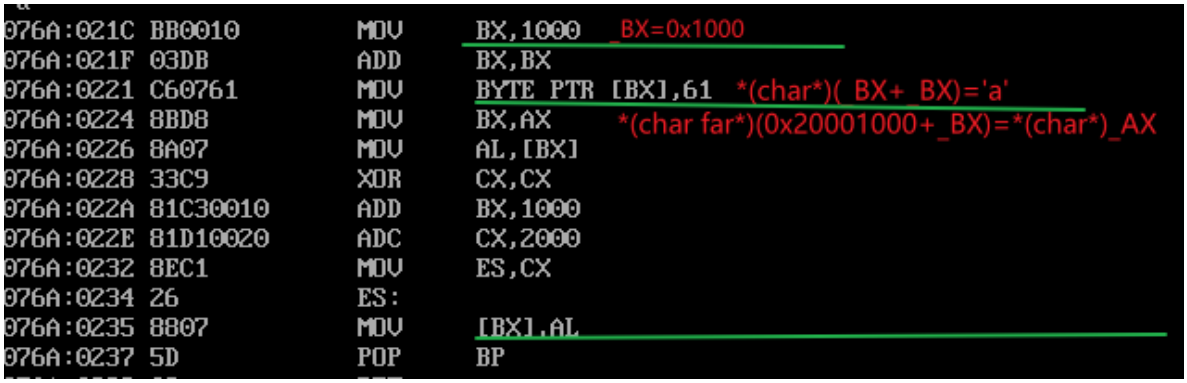
0604-研究试验3-宣讲会研究报告-尹忠恩

学习过程

（1）





（2）

```
main()
{
    *(char far *)0x0b80009B0 = 0x61;
    *(char far *)0x0b80009B1 = 0x2;
}
```

（3）

```
076A:01FA 55            PUSH    BP
076A:01FB 8BEC          MOV     BP,SP
076A:01FD 83EC06        SUB     SP,+06
076A:0200 C706A601A100  MOV     WORD PTR [01A6],00A1
076A:0206 C706A801A200  MOV     WORD PTR [01A8],00A2
076A:020C C706AA01A300  MOV     WORD PTR [01AA],00A3
076A:0212 C746FAB100    MOV     WORD PTR [BP-06],00B1
076A:0217 C746FCB200    MOV     WORD PTR [BP-04],00B2
-u
076A:021C C746FEB300    MOV     WORD PTR [BP-02],00B3
076A:0221 8BE5          MOV     SP,BP
076A:0223 5D            POP     BP
076A:0224 C3            RET
076A:0225 55            PUSH    BP
076A:0226 8BEC          MOV     BP,SP
076A:0228 83EC06        SUB     SP,+06
076A:022B C706A601A10F  MOV     WORD PTR [01A6],0FA1
076A:0231 C706A801A20F  MOV     WORD PTR [01A8],0FA2
076A:0237 C706AA01A30F  MOV     WORD PTR [01AA],0FA3
```

```
076A:023D C746FAC100    MOV     WORD PTR [BP-06],00C1
076A:0242 C746FCC200    MOV     WORD PTR [BP-04],00C2
076A:0247 C746FEC300    MOV     WORD PTR [BP-02],00C3
076A:024C 8BE5          MOV     SP,BP
076A:024E 5D            POP     BP
076A:024F C3            RET
076A:0250 C3            RET
```

（4）

书上的代码转换为的汇编

```
076A:01FA 55            PUSH    BP
076A:01FB 8BEC          MOV     BP,SP
076A:01FD 83EC02        SUB     SP,+02
076A:0200 E80700        CALL    020A
076A:0203 8946FE        MOV     [BP-02],AX
076A:0206 8BE5          MOV     SP,BP
076A:0208 5D            POP     BP
076A:0209 C3            RET
076A:020A 55            PUSH    BP
076A:020B 8BEC          MOV     BP,SP
076A:020D A1A601        MOV     AX,[01A6]
076A:0210 0306A801      ADD     AX,[01A8]
076A:0214 A3AA01        MOV     [01AA],AX
076A:0217 A1AA01        MOV     AX,[01AA]
```

```
076A:0214 A3AA01        MOV     [01AA],AX
076A:0217 A1AA01        MOV     AX,[01AA]
-u
076A:021A EB00          JMP     021C
076A:021C 5D            POP     BP
076A:021D C3            RET
076A:021E C3            RET
```

（5）

```
-u
076A:01FA 55            PUSH    BP
076A:01FB 8BEC          MOV     BP,SP
076A:01FD B81400        MOV     AX,0014
076A:0200 50            PUSH    AX
076A:0201 E8E702        CALL    04EB
076A:0204 59            POP     CX
076A:0205 33DB          XOR     BX,BX
076A:0207 8EC3          MOV     ES,BX
076A:0209 BB0002        MOV     BX,0200
076A:020C 26            ES:
076A:020D 8907          MOV     [BX],AX
076A:020F 33DB          XOR     BX,BX
076A:0211 8EC3          MOV     ES,BX
076A:0213 BB0002        MOV     BX,0200
076A:0216 26            ES:
076A:0217 8B1F          MOV     BX,[BX]
076A:0219 C6470A00      MOV     BYTE PTR [BX+0A],00
```

```
076A:0219 C6470A00      MOV     BYTE PTR [BX+0A],00
-u
076A:021D EB3C          JMP     025B
076A:021F 33DB          XOR     BX,BX
076A:0221 8EC3          MOV     ES,BX
076A:0223 BB0002        MOV     BX,0200
076A:0226 26            ES:
076A:0227 8B1F          MOV     BX,[BX]
076A:0229 8A470A        MOV     AL,[BX+0A]
076A:022C 0461          ADD     AL,61
076A:022E 33DB          XOR     BX,BX
076A:0230 8EC3          MOV     ES,BX
076A:0232 BB0002        MOV     BX,0200
076A:0235 26            ES:
076A:0236 8B1F          MOV     BX,[BX]
076A:0238 50            PUSH    AX
076A:0239 53            PUSH    BX
076A:023A 33DB          XOR     BX,BX
076A:023C 8EC3          MOV     ES,BX
```

```
076A:023C 8EC3          MOV     ES,BX
-u
076A:023E BB0002        MOV     BX,0200
076A:0241 26            ES:
076A:0242 8B1F          MOV     BX,[BX]
076A:0244 8A470A        MOV     AL,[BX+0A]
076A:0247 98            CBW
076A:0248 5B            POP     BX
076A:0249 03D8          ADD     BX,AX
076A:024B 58            POP     AX
076A:024C 8807          MOV     [BX],AL
076A:024E 33DB          XOR     BX,BX
076A:0250 8EC3          MOV     ES,BX
076A:0252 BB0002        MOV     BX,0200
076A:0255 26            ES:
076A:0256 8B1F          MOV     BX,[BX]
076A:0258 FE470A        INC     BYTE PTR [BX+0A]
076A:025B 33DB          XOR     BX,BX
076A:025D 8EC3          MOV     ES,BX
```

```
076A:025D 8EC3          MOV     ES,BX
-u
076A:025F BB0002        MOV     BX,0200
076A:0262 26            ES:
076A:0263 8B1F          MOV     BX,[BX]
076A:0265 807F0A08      CMP     BYTE PTR [BX+0A],08
076A:0269 75B4          JNZ     021F
076A:026B 33DB          XOR     BX,BX
076A:026D 8EC3          MOV     ES,BX
076A:026F BB0002        MOV     BX,0200
076A:0272 26            ES:
076A:0273 FF37          PUSH    [BX]
076A:0275 E87004        CALL    06E8
076A:0278 59            POP     CX
076A:0279 5D            POP     BP
076A:027A C3            RET
076A:027B C3            RET
076A:027C 55            PUSH    BP
076A:027D 8BEC          MOV     BP,SP
```

解决的问题

（2）

```
Welcome to DOSBox v0.74

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>MOUNT c: F:\GitHub\ASM-\minic        a
Drive C is mounted as local directory F:\GitHub\ASM-\minic\

Z:\>c:

C:\>GREENA.EXE

C:\>_
```

（3）

1. 全局变量放在内存中

1.
```
AX=0000  BX=023C  CX=000C  DX=D385  SP=FFD8  BP=FFDE  SI=003A  DI=022F
DS=07C5  ES=07C5  SS=07C5  CS=076A  IP=0200   NV UP EI NG NZ NA PE NC
076A:0200 C706A601A100  MOV     WORD PTR [01A6],00A1              DS:01A6=0000
-t

AX=0000  BX=023C  CX=000C  DX=D385  SP=FFD8  BP=FFDE  SI=003A  DI=022F
DS=07C5  ES=07C5  SS=07C5  CS=076A  IP=0206   NV UP EI NG NZ NA PE NC
076A:0206 C706A801A200  MOV     WORD PTR [01A8],00A2              DS:01A8=0000
-t

AX=0000  BX=023C  CX=000C  DX=D385  SP=FFD8  BP=FFDE  SI=003A  DI=022F
DS=07C5  ES=07C5  SS=07C5  CS=076A  IP=020C   NV UP EI NG NZ NA PE NC
076A:020C C706AA01A300  MOV     WORD PTR [01AA],00A3              DS:01AA=0000
```

2. 局部变量放在栈中

1.
```
AX=0000  BX=023C  CX=000C  DX=D385  SP=FFD8  BP=FFDE  SI=003A  DI=022F
DS=07C5  ES=07C5  SS=07C5  CS=076A  IP=0212   NV UP EI NG NZ NA PE NC
076A:0212 C746FAB100    MOV     WORD PTR [BP-06],00B1             SS:FFD8=01FD
-t

AX=0000  BX=023C  CX=000C  DX=D385  SP=FFD8  BP=FFDE  SI=003A  DI=022F
DS=07C5  ES=07C5  SS=07C5  CS=076A  IP=0217   NV UP EI NG NZ NA PE NC
076A:0217 C746FCB200    MOV     WORD PTR [BP-04],00B2             SS:FFDA=076A
-t

AX=0000  BX=023C  CX=000C  DX=D385  SP=FFD8  BP=FFDE  SI=003A  DI=022F
DS=07C5  ES=07C5  SS=07C5  CS=076A  IP=021C   NV UP EI NG NZ NA PE NC
076A:021C C746FEB300    MOV     WORD PTR [BP-02],00B3             SS:FFDC=7346
```

3. 每个函数开头的 `push bp mov bp sp` 就是为了在函数中正确的使用局部变量

1. main程序开始

```
0C  DX=F22E  SP=FFE0  BP=FFEA
C5  CS=076A  IP=01FA   NV UP EI
PUSH    BP
```

```
0C  DX=F22E  SP=FFDE  BP=FFEA
C5  CS=076A  IP=01FB   NV UP EI
MOV     BP,SP
```

```
0C  DX=F22E  SP=FFDE  BP=FFDE
C5  CS=076A  IP=01FD   NV UP EI
```

2. main程序结束

```
00C  DX=F22E  SP=FFD8  BP=FFDE
7C5  CS=076A  IP=0221   NV UP EI
MOV     SP,BP
```

```
00C  DX=F22E  SP=FFDE  BP=FFDE
7C5  CS=076A  IP=0223   NV UP EI
POP     BP
```

```
00C  DX=F22E  SP=FFE0  BP=FFEA
7C5  CS=076A  IP=0224   NV UP EI
RET
```

（4）

1. 返回值存放在 **ax** 中

```
1  //由于书上的代码返回值为0，不好对比。故将 a b 赋值为 1
2  int f(void);
3
4  int a, b, ab;
5
6  main()
7  {
```

```
    int c;
    c = f();
}

int f(void)
{
    a = b = 1;//返回值为2
    ab = a + b;
    return ab;
}
```

- ```
  AX=0000  BX=023C  CX=000D  DX=16D7  SP=FFDE  BP=FFE0  SI=003A  DI=022F
  DS=07C2  ES=07C2  SS=07C2  CS=076A  IP=0200   NV UP EI NG NZ AC PE NC
  076A:0200 E80700        CALL    020A
  -p
  AX=0002  BX=023C  CX=000D  DX=16D7  SP=FFDE  BP=FFE0  SI=003A  DI=022F
  DS=07C2  ES=07C2  SS=07C2  CS=076A  IP=0203   NV UP EI PL NZ NA PO NC
  076A:0203 8946FE        MOV     [BP-02],AX                      SS:FFDE=7346
  076A:020A 55            PUSH    BP
  076A:020B 8BEC          MOV     BP,SP
  076A:020D B80100        MOV     AX,0001
  076A:0210 A3A801        MOV     [01A8],AX
  076A:0213 A3A601        MOV     [01A6],AX
  076A:0216 A1A601        MOV     AX,[01A6]
  076A:0219 0306A801      ADD     AX,[01A8]
  076A:021D A3AA01        MOV     [01AA],AX
  076A:0220 A1AA01        MOV     AX,[01AA]
  076A:0223 EB00          JMP     0225
  076A:0225 5D            POP     BP
  076A:0226 C3            RET
  ```

（5）

`void *malloc(long NumBytes)`：该函数分配了`NumBytes`个字节，并返回了指向这块内存的指针。如果分配失败，则返回一个空指针（NULL）。

`void free(void *FirstByte)`：该函数是将之前用malloc分配的空间还给程序或者是操作系统，也就是释放了这块内存，让它重新得到自由。

研究体会

本次实验较前两实验略显复杂，需要大量编译链接分析工作，通过本次实验，认识的C语言分配内存的机制
  ```