

## 汇编运行过程

### 准备

DOSBox, Link, Masm介绍

Link和masm

DOSBox

### 开始写程序

此次用到的文件

利用masm.exe生成.OBJ文件

利用link.exe运行.OBJ文件, 生成.exe文件

运行.exe文件

本次的代码

### 大作业

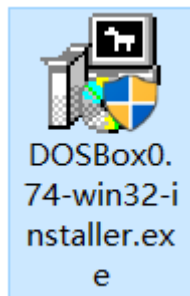
USE16的作用

# 汇编运行过程

## 准备

由于大多数win10系统都是64位系统, 所以如果直接用link.exe和masm.exe来运行asm文件是不可以的。

所以需要安装DOSBox



## DOSBox, Link, Masm介绍

### Link和masm

masm32包即可以支持开发在DOS操作系统下运行的16位应用程序, 也可以开发在Windows操作系统下运行的32位应用程序。

masm.exe——编译器。我们编写的源代码文件就是通过它来汇编生成中间代码文件, 即通常扩展名为.obj的文件。

link16.exe——连接器。由masm.exe汇编生成的.obj文件还不能直接上机运行, 必须通过连接器link16.exe将其连接制作成扩展名为.exe (或者.com)的可执行文件才能上机。

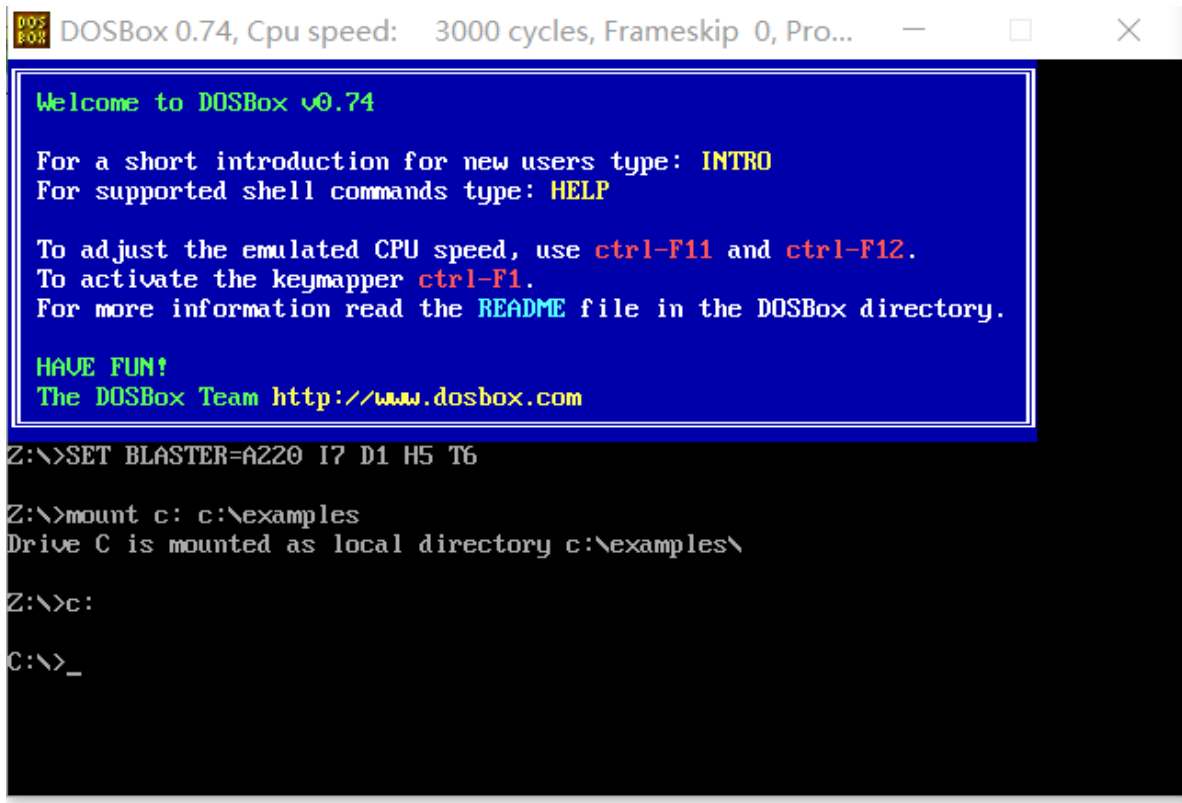
但是我们主流的系统基本都是64位了, 所以需要用DOSBox来给我们提供一个16位的DOS系统环境

### DOSBox

在现在的系统下模拟DOS环境的一个工具! 是为了运行以前纯DOS环境下的软件而开发的! 当然主要用途是运行原来的那些只能在DOS下玩的老游戏!

使用方法：

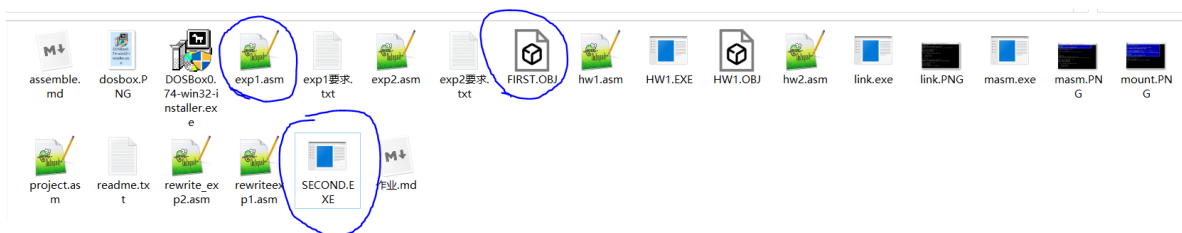
1. 将存放masm、link和源程序文件的文件夹拷贝到某一目录下，比如c:\examples
2. 然后将这个目录挂为DOSBox的一个盘符下，挂载命令为  
`Mount c c:\examples`；可以理解为把当前工作目录切换到c:\examples这里，即c:\examples就是你以后的c:。
3. 再切换到挂载的c盘：  
`c:`；切换到工作目录



## 开始写程序

### 此次用到的文件

仔细对应DOSBox图片里面的代码和这个图片里面的文件

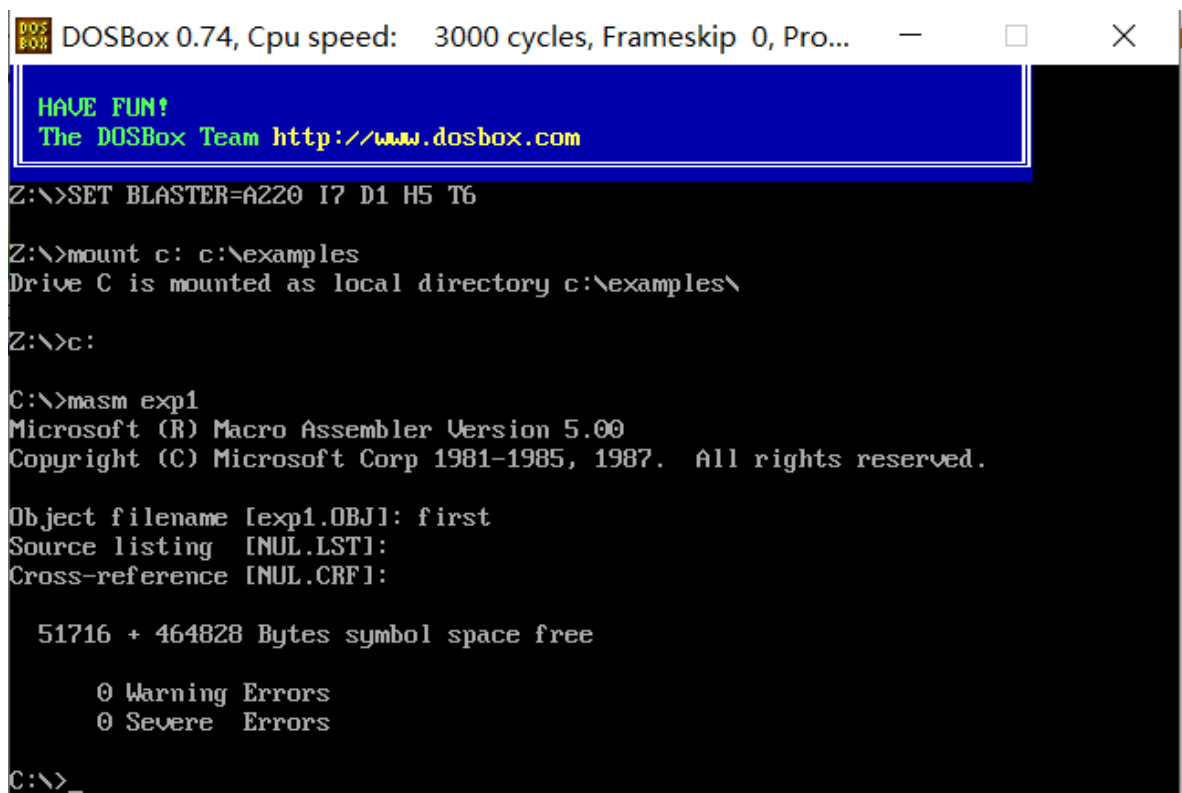


### 利用masm.exe生成.OBJ文件

在后缀名为.asm的文件中，编写完代码之后

在DOSBox中利用masm.exe运行.asm文件

`masm yourfile.asm`；注意，可以不加.asm后缀.不加的时候要名字不重复



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...
HAVE FUN!
The DOSBox Team http://www.dosbox.com

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

Z:\>mount c: c:\examples
Drive C is mounted as local directory c:\examples\

Z:\>c:

C:\>masm exp1
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Object filename [exp1.OBJ]: first
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

51716 + 464828 Bytes symbol space free

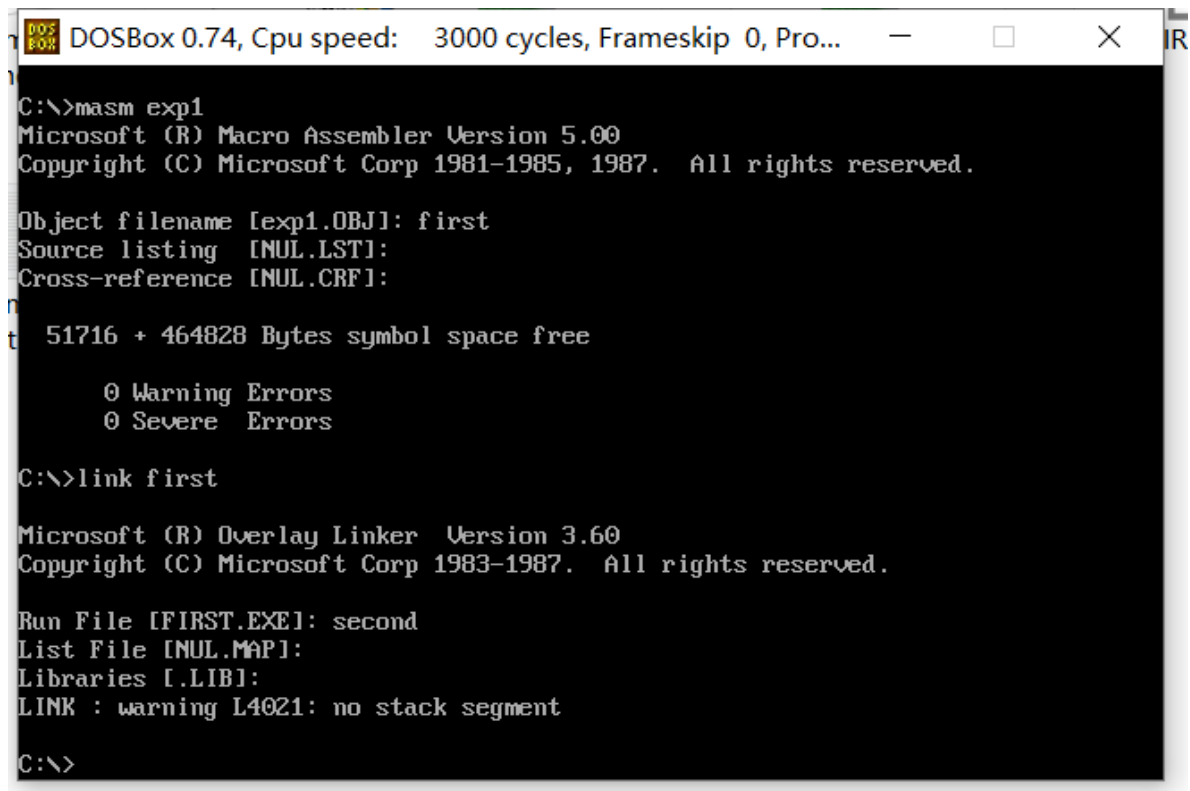
0 Warning Errors
0 Severe Errors

C:\>_
```

## 利用link.exe运行.OBJ文件，生成.exe文件

在DOSBox 中利用 `link.exe` 运行 `.OBJ` 文件,然后生成 `.exe` 文件

`link yourfile.OBJ` ; 注意，可以不加`.OBJ`后缀.不加的时候要保持名字不重复



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...
C:\>masm exp1
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Object filename [exp1.OBJ]: first
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

51716 + 464828 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>link first

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [FIRST.EXE]: second
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

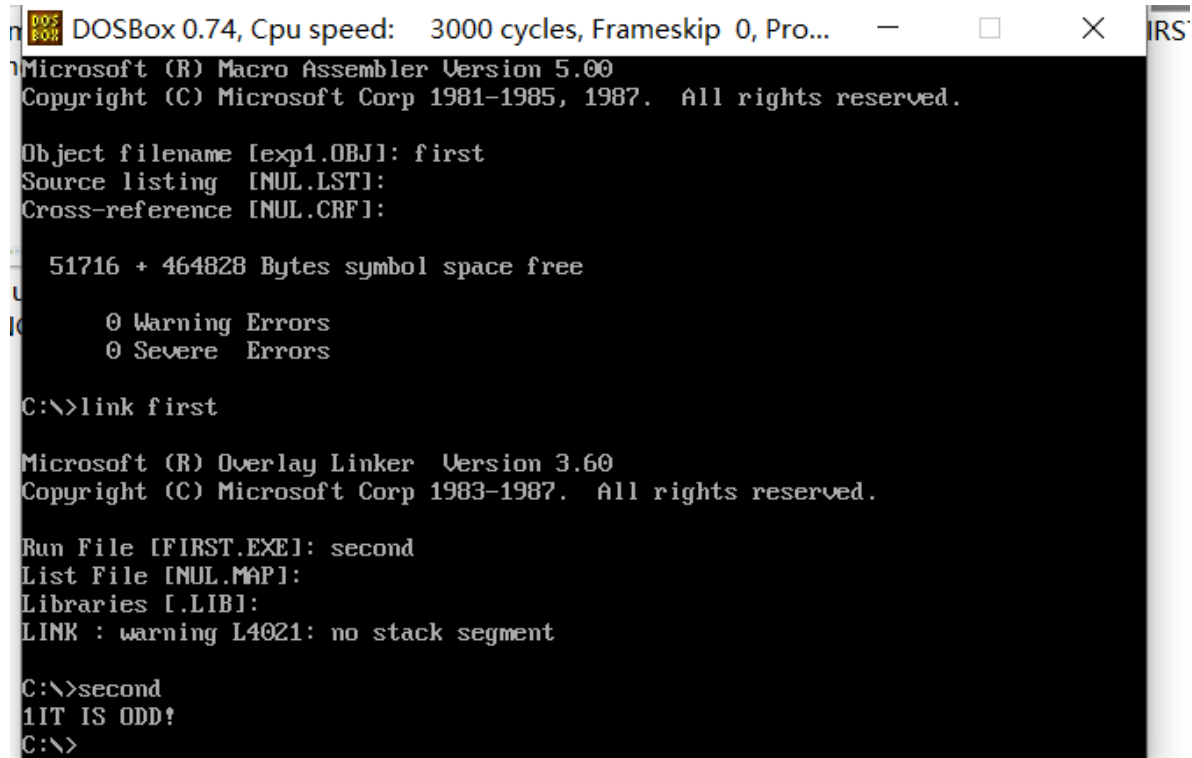
C:\>
```

这里有warning L4021是因为我在写代码的时候没有定义堆栈区，但是没有关系

系统自动会帮我们建立堆栈区

## 运行.exe文件

在DOSBox中直接运行 .exe 文件 .OBJ 文件,然后生成 .exe 文件



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Object filename [exp1.OBJ]: first
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

51716 + 464828 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>link first

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [FIRST.EXE]: second
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\>second
1IT IS ODD!
C:\>
```

## 本次的代码

```
DATA SEGMENT                                ;定义数据段
    STRING1 DB 'IT IS ODD!','$'
    STRING2 DB 'IT IS EVEN!','$'
DATA ENDS

CODE SEGMENT                                ;定义代码段
    ASSUME CS:CODE,DS:DATA                  ;ASSUME伪指令,说明段与段寄存器之间的对应关系
START:
    MOV AX,DATA                             ;实现段与段寄存器之间的对应关系,代码段系统会默认
    MOV DS,AX

    MOV AH,01H                             ;调用1号DOS功能,从键盘上读入一个字符,并将该字符回
    显在屏幕上
    INT 21H
    CLC
    RCR AL,1
    JNC EVN
    MOV DX,OFFSET STRING1                   ;将字符串String1的偏移地址赋给DX
    CALL DISPMESS
EVN: MOV DX,OFFSET STRING2
    CALL DISPMESS

DISPMESS PROC                               ;定义一个过程,用于输出字符串
    MOV AH,9                               ;调用9号子功能
    INT 21H                                ;返回DOS
    MOV AH,4CH                             ;结束程序
    INT 21H
```

```
DISPMESS ENDP
```

```
CODE ENDS
```

```
END START
```

## 大作业

### USE16的作用

这句指的是，数据段采取16位来存取数据段中的数据。如果use32则是指用32位来存取数据段中的数据。具体的16还是32，根据你编程面向的cpu类型是16位的还是32位的。

另外再补充一点点吧。

cpu将内部存储单元分成许多逻辑段（Segment），这样，一个存储单元除具有一个唯一的物理地址外，还具有多个逻辑地址。

data segment 指的就是数据段。

数据段存放运行程序所用的数据。数据段寄存器DS存放数据段的段地址。各种主存寻址方式（有效地址EA）得到存储器中操作数的偏移地址。处理器利用DS:EA存取数据段中的数据。

```
DATA          SEGMENT
    MSG1      DB      'USERNAME:',0DH,0AH,'$'      ;指定用户名存放的数据区
    MSG2      DB      0DH,0AH,'PASSWORD:',0DH,0AH,'$'      ;指定密码的数据区
    MSG3      DB      0DH,0AH,0DH,0AH,'Password Right!',0DH,0AH,'$'      ;密码正确提示
    MSG4      DB      0DH,0AH,0DH,0AH,'USERNAME Error!',0DH,0AH,'$';      ;用户名报错
    MSG5      DB      0DH,0AH,0DH,0AH,'Password Error!',0DH,0AH,'$';      ;密码报错
```

```
    BUF1      DB      '2018633007'
    USERLEN    EQU      $-BUF1      ;统计用户名长度
    BUF2      DB      '5201314'
    PWLEN      EQU      $-BUF2      ;统计密码长度
    BUF3      DB      15            ;用户输入用户名的数据区
                DB      ?
                DB      15          DUP(?)
    BUF4      DB      15          DUP(?)      ;用户输入密码的数据区
```

```
DATA          ENDS
```

```
CODE          SEGMENT
```

```
    ASSUME     CS:CODE,DS:DATA
```

```
START:
```

```
    MOV        AX,DATA
```

```
    MOV        DS,AX
```

```
    MOV        AH,9
```

```
    MOV        DX,OFFSET MSG1      ;显示用户名输入界面
```

```
    INT        21H
```

```
    MOV        AH,0AH
```

```
    MOV        DX,OFFSET BUF3      ;键入用户名字符串
```

```
    INT        21H
```

```
    MOV        BX,OFFSET BUF1      ;获得储存正确账户名的偏移地址
```

```
    MOV        SI,OFFSET BUF3+2    ;获得储存输入账户名的偏移地址
```

```
    MOV        CX,USERLEN
```

**CMPUSER:**

```
    ; 进行用户名对比
    MOV     AL,[BX]
    CMP     [SI],AL                ;将输入字符串的对应位置与正确账户名进行比较
    JNZ     ERROR1                ;当前位置和原本的用户名不一样就报错
    INC     SI
    INC     BX
    LOOP    CMPUSER                ;重复进行直到字符串结束
    MOV     AH,9
    MOV     DX,OFFSET    MSG2      ;显示密码输入界面
    INT     21H
    MOV     CX,PWLEN                ; 将正确的密码长度读入
    MOV     SI,OFFSET    BUF4      ;获得储存输入密码的偏移地址
```

**GETPW:**

```
    MOV     AH,7                    ;逐个字符输入密码，遇到回车结束输入
    INT     21H
    CMP     AL,0DH
    JZ      NEXT
    CMP     AL,08H
    JE      DELETE
    MOV     [SI],AL
    MOV     AH,2                    ;在屏幕上显示*
    MOV     DX,'*'
    INT     21H
    INC     SI
    INC     BX                        ; 统计输入密码的长度
    LOOP    GETPW
```

**DELETE:**

```
    CMP     BX, 00H                ; 如果删除到第0个位置，再删除就直接退出
    JZ      EXIT
    DEC     BX

    CALL    DELCODE
    MOV     AH,2                    ;调用2号中断，显示空格，消除星号
    MOV     DL,' '
    INT     21H
    MOV     BYTE PTR [SI],00H
    DEC     SI
    CALL    DELCODE

    JMP     GETPW
```

**NEXT:**

```
    MOV     CX,PWLEN
    MOV     BX,OFFSET    BUF2
    MOV     SI,OFFSET    BUF4
```

**NEXT3:**

```
    MOV     AL,[BX]
    CMP     [SI],AL
    JNZ     ERROR2
    INC     SI
    INC     BX
    LOOP    NEXT3                ;逐个字符比较BUF2和BUF4
    MOV     AH,9
    MOV     DX,OFFSET    MSG3
```

```

        INT     21H
        JMP     EXIT

DELCODE    PROC
        ; 删除一个字符，即把光标向左移动一格
        MOV     AH,3        ; 获取光标位置
        INT     10H
        MOV     AH,2        ; 2号中断：置光标位置
        SUB     DL,1        ; 列数减1，实现光标左移
        INT     10H
        RET
DELCODE    ENDP

ERROR1:
        ; 用户名错误时报错
        MOV     DX,OFFSET    MSG4
        CALL    DISPMESS

ERROR2:
        ; 密码错误时报错
        MOV     DX,OFFSET    MSG5
        CALL    DISPMESS

DISPMESS    PROC
        MOV     AH,9
        INT     21H
        JMP     EXIT
DISPMESS    ENDP

EXIT:
        MOV     AH,4CH
        INT     21H

CODE       ENDS
END START

```