Hello World 汇编语言说明文档

2254269 吴昊泽

一、传统编译方式

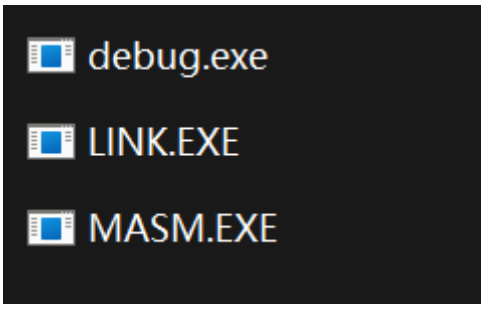
1、环境配置

- 安装`DOSBOX`，并创建文件夹用于存放汇编语言需要用到的配件，我使用的文件名为`AsPro`
- 在`DOSBOX`安装目录中，打开`DOSBox 0.74 Options.bat`，末尾添加如下内容设置编译虚拟环境

```
[autoexec]
# Lines in this section will be run at startup.
# You can put your MOUNT lines here.
mount d d:\AsPro  # 此处文件夹路径为上一步创建好的文件夹
d:
```

  这一步可做可不做，但是不做的话需要每次启动`DOSBOX`都输入一遍该`mount`指令

- 在该文件夹下添加`LINK.EXE`，`debug.exe`，`MASM.EXE`（均可在学院服务器获得）



2、创建asm文件

- 在`AsPro`文件下创建`hello.asm`，并添加简单的`hello world`程序（学院服务器也提供了源代码）

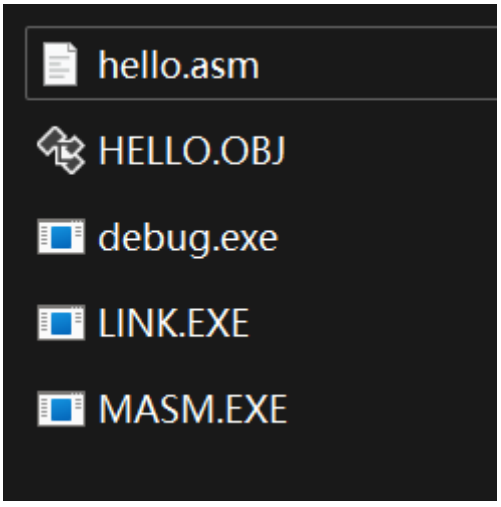
```
.model small
.data
Hello     DB 'Hello world!',0dh,0ah,'$'
.code
START:
    MOV AX,@DATA
    MOV DS,AX
    MOV DX,offset Hello
    MOV AH,9
    INT 21H

    MOV AX,4C00H
    INT 21h
END   START
```

3、编译及运行hello world程序

- 在`DOSBOX`中输入命令`masm hello.asm`，调用`MASM.EXE`编译程序，一路回车即可

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

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

  51670 + 464874 Bytes symbol space free

       0 Warning Errors
       0 Severe  Errors
```

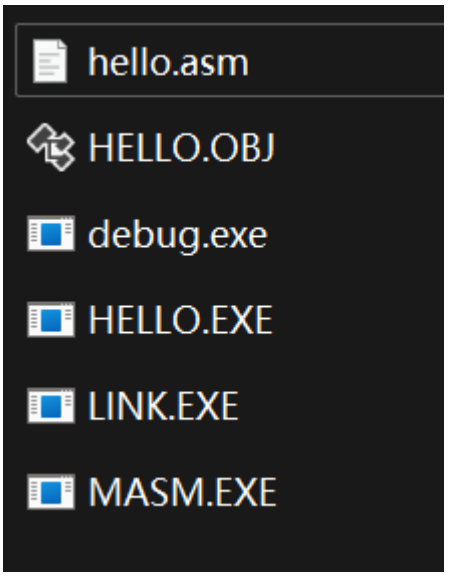
得到`hello.obj`



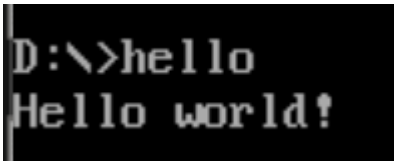
- 在`DOSBOX`中输入命令`link hello.obj`或者直接`link hello`，调用`LINK.EXE`链接程序，同样一路回车即可

```
D:\>link hello.obj

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

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

得到可执行文件`HELLO.EXE`



- 输入命令`hello`，执行`HELLO.EXE`



- 使用`debug`指令可以对EXE文件执行反汇编，在`DOSBOX`中输入`debug hello.exe`，使用`-u`指令查看反汇编结果

```
D:\>debug hello.exe
-u
076A:0000 B86B07        MOV     AX,076B
076A:0003 8ED8          MOV     DS,AX
076A:0005 BA0200        MOV     DX,0002
076A:0008 B409          MOV     AH,09
076A:000A CD21          INT     21
076A:000C B8004C        MOV     AX,4C00
076A:000F CD21          INT     21
076A:0011 004865        ADD     [BX+SI+65],CL
076A:0014 6C            DB      6C
076A:0015 6C            DB      6C
076A:0016 6F            DB      6F
076A:0017 20776F        AND     [BX+6F],DH
076A:001A 726C          JB      0088
076A:001C 64            DB      64
076A:001D 210D          AND     [DI],CX
076A:001F 0A24          OR      AH,[SI]
```

二、内存写入数据方式

1、使用debug

- 学习`hello剖析.pdf`文件中对hello的另类执行方式（学院服务器下）
- 输入`debug hello.exe`，`-r`命令查看寄存器

```
D:\>debug hello.exe
-r
AX=FFFF  BX=0000  CX=0021  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076A  IP=0000   NV UP EI PL NZ NA PO NC
076A:0000 B86B07        MOV     AX,076B
```

2、写数据到内存

- 使用`-e 076a: 0`命令，将“Hello$"对应的ASCII码`48 65 6c 6c 6f 24`写入内存

```
-e 076a: 0
076A:0000  00.48   00.65   00.6c   00.6c   00.6f   00.24
```

3、写代码的机器码到内存

- 通过`-u`查看自己代码的机器码内容：

```
-u
076F:0000 B86E07        MOV     AX,076E
076F:0003 8ED8          MOV     DS,AX
076F:0005 B409          MOV     AH,09
076F:0007 BA0000        MOV     DX,0000
076F:000A CD21          INT     21
076F:000C B8004C        MOV     AX,4C00
076F:000F CD21          INT     21
076F:0011 0000          ADD     [BX+SI],AL
076F:0013 0000          ADD     [BX+SI],AL
076F:0015 0000          ADD     [BX+SI],AL
076F:0017 0000          ADD     [BX+SI],AL
076F:0019 0000          ADD     [BX+SI],AL
076F:001B 0000          ADD     [BX+SI],AL
076F:001D 0000          ADD     [BX+SI],AL
076F:001F 0000          ADD     [BX+SI],AL
```

- 使用`-e`命令，将代码的机器码`b8 6a 07 8e d8 b4 09 ba 00 00 cd 21 b8 00 4c cd 21`（17个字节）写入内存

  其中：`B8 6A 07`代表将`076A`中的数据存入`AX`，`076A`来源于编写者在写数据到内存中时的选择

- 用`-e 076b: 0`回车 逐个写入（空格自动分开）

```
-e 076b: 0
076B:0000  00.b8   00.6a   00.07   00.8e   00.d8   00.b4   00.09   00.ba
076B:0008  00.00   00.00   00.cd   00.21   00.b8   00.00   00.4c   00.cd
076B:0010  00.21
```

4、修改寄存器及执行

- 通过`-r`查看及修改对应的寄存器

  其中`DS`为数据段，`CS`为代码段，其余暂不重要

```
-r
AX=FFFF  BX=0000  CX=0061  DX=0000  SP=0040  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=076A  CS=076F  IP=0000   NV UP EI PL NZ NA PO NC
076F:0000 B86E07        MOV     AX,076E
-r cs
CS 076F
:076b
-r ds
DS 075A
:076a
-r ip
IP 0000
:0
-r
AX=FFFF  BX=0000  CX=0061  DX=0000  SP=0040  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=076A  CS=076B  IP=0000   NV UP EI PL NZ NA PO NC
076B:0000 0000          ADD     [BX+SI],AL                          DS:0000=00
```

- 输入`-g`运行：

```
-r
AX=FFFF  BX=0000  CX=0061  DX=0000  SP=0040  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=076A  CS=076B  IP=0000   NV UP EI PL NZ NA PO NC
076B:0000 B86A07        MOV     AX,076A
-g
Hello
```