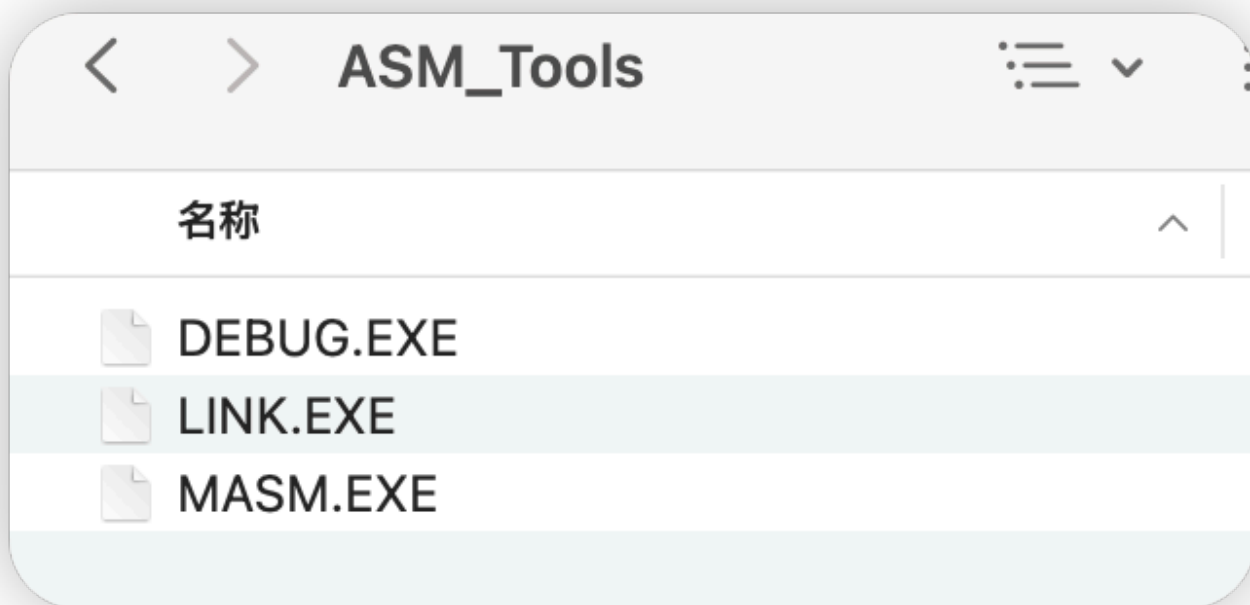


Assignment 1 Helloworld

传统编译方式

环境配置

1. 安装 [DOSBox](#)
2. 在 ~\Assembly_Language\ASM_Tools 文件夹中添加 DEBUG.EXE、LINK.EXE 和 MASM.EXE。



创建汇编文件

在 ASM_Tools 文件夹中创建 HELLO.ASM，汇编文件内容如下：

```
.MODEL SMALL                ; 定义内存模型为small模式
.STACK 100h                 ; 定义堆栈大小为256字节

.DATA
    Hello DB 'Hello world!', 0dh, 0ah, '$' ; 定义要输出的字符串 'Hello world!'
                                           ; 0Dh = 回车符 (CR)
                                           ; 0Ah = 换行符 (LF)
                                           ; '$' 表示字符串的结束符，用于DOS的INT 21h功能9

.CODE
START:
    ; 初始化数据段寄存器
    MOV AX, @DATA           ; 将数据段的基地址存入AX
    MOV DS, AX              ; 将AX中的数据段地址加载到DS寄存器

    ; 调用DOS中断输出字符串
```

MOV DX, offset Hello	; 将Hello字符串的地址存入DX
MOV AH, 9	; DOS功能号9: 输出字符串, 字符串必须以'\$'结束
INT 21H	; 触发中断21h, 执行字符串输出
; 正常结束程序	
MOV AX, 4C00H	; 设置返回代码为0的结束程序指令
INT 21h	; 触发中断21h, 返回到操作系统
END START	; 标记程序结束, START是入口点

编译并运行汇编程序

1. 启动 DOSBox 0.74。
2. 执行命令 `masm hello.asm`, Object filename 选项回车, 表示接受默认文件名, Source listing 选项回车, 表示不生成源代码列表文件, Cross-reference 回车, 表示不生成交叉引用表。这一步的作用是使用编译器将汇编语言的 .asm 源文件编译成一个 .obj 文件, 用于后续的连接步骤。

```
Z:\>mount d ~/ASM_Tools
Directory /Users/jack/ASM_Tools doesn't exist.

Z:\>mount d ~/work/Assembly_Language/ASM_Tools
Drive D is mounted as local directory /Users/jack/work/Assembly_Language/ASM_Tools/

Z:\>d:

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

D:\>
```

3. 执行命令 `link hello.obj`, Run File 选项回车, 表示接受默认文件名, List File 选项回车, 表示不生成列表文件, Libraries 选项回车, 表示不链接任何额外的库文件, 采用默认设置。这一步通过链接器将编译生成的目标文件 hello.obj 链接为可执行文件。

```
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]:

D:\>
```

4. 执行命令 `hello`，执行可执行程序。

```
D:\>hello
Hello world!
```

5. 执行命令 `debug hello.exe`，再使用 `-u` 命令，可以反汇编可执行文件的机器代码，逐条显示汇编指令。

```
D:\>debug hello.exe
-u
076A:0000 BB6B07      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 BB004C      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]
```

反汇编结果分析

1. `076A:0000 BB6B07 MOV AX,076B`
将立即数 076Bh 传送到 AX 寄存器中。
2. `076A:0003 8EDB MOV DS,AX`
将 AX 寄存器的内容（即 076Bh）传送给数据段寄存器 DS，用于设置当前的数据段基址。
3. `076A:0005 BA0209 MOV DX,0902`
将地址 0902h 加载到 DX 寄存器，通常用于指向要操作的字符串或数据。
4. `076A:0008 B409 MOV AH,09`
将功能号 09h 写入 AH 寄存器，为调用 DOS 中断 21h 的“显示字符串”功能做准备。
5. `076A:000A CD21 INT 21`
调用 DOS 中断 21h，执行 AH=09h 对应的功能，即在屏幕上输出以 '\$' 结尾的字符串。
6. `076A:000C BB004C MOV AX,4C00`
将值 4C00h 传送到 AX 寄存器，其中 AH=4Ch 表示调用 DOS 程序终止功能，AL=00h 表示正常退出状态。

7. 076A:000F CD21 INT 21

再次调用 DOS 中断 21h，执行 AH=4Ch 功能，结束当前程序并返回操作系统。

内存写入数据方式

1. 启动 DOSBox 0.74。

2. 执行命令 `debug hello.exe`，再使用 `-r` 命令，查看并显示 CPU 寄存器的当前状态。

```
D:\>debug hello.asm
-r
AX=0000 BX=0000 CX=00C2 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100  NU UP EI PL NZ NA PO NC
073F:0100 2E          CS:
073F:0101 6D          DB      6D
```

3. 执行命令 `-e 076a: 0`，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
```

4. 执行命令 `-e 076b: 0`，将代码的机器码 `b8 6b 07 be d8 ba 02 00 b4 09 cd 21 b8 00 4c cd 21` (17 个字节) 写入内存。

```
-e 076b: 0
076B:0000  00.b8  00.6b  00.07  00.be  00.d8  00.ba  00.02  00.00
076B:0008  00.b4  00.09  00.cd  00.21  00.b8  00.00  00.4c  00.cd
076B:0010  00.21
```

5. 执行命令 `-r` 查看并修改相应寄存器的值。

```
-r
AX=0000 BX=0000 CX=00C2 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100  NU UP EI PL NZ NA PO NC
073F:0100 2E          CS:
073F:0101 6D          DB      6D
-r cs
CS 073F
:076B
-r ds
DS 073F
:076A
-r
AX=0000 BX=0000 CX=00C2 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=076A ES=073F SS=073F CS=076B IP=0100  NU UP EI PL NZ NA PO NC
076B:0100 0000          ADD     [BX+SI],AL  DS:0000=48
```

6. 执行命令 `-g`，执行程序。

```
-g
Hello
```