

实验报告一

一、实验目的：

掌握常用的 DOS 目录操作类命令、文件操作类命令以及其它命令的格式及使用方法。

二、实验任务：

1. 练习目录操作类命令：MD、CD、RD、DIR、PATH、TREE；
2. 练习文件操作类命令：COPY、TYPE、REN、DEL；
3. 其它命令：CLS、VER、DATA、TIME。

二、实验过程：

1. 目录操作类命令：

(1) MD——建立子目录：

- ①在 C 盘的根目录创建名为 FOX 的子目录：

```
C:\>MD FOX
```

- ②在 FOX 子目录下再创建 USER 子目录：

```
C:\>MD FOX\USER
```

(2) CD——改变当前目录：

- ①进入到 USER 子目录：

```
C:\>CD FOX\USER
```

- ②从 USER 子目录退回到子目录：

```
C:\FOX\USER>CD..
```

- ③返回到根目录：

```
C:\FOX>CD\
```

(3) RD——删除子目录：

把 C 盘 FOX 子目录下的 USER 子目录删除，操作如下：

- ①先将 USER 子目录下的文件删空：

```
C:\>DEL C:\FOX\USER\*.*  
C:\FOX\USER\*.*. 是否确认(Y/N)? Y
```

- ②删除 USER 子目录：

```
C:\>RD C:\FOX\USER
```

(4) DIR——显示磁盘目录：

- ①显示 C:\WINDOWS 目录的内容：

```
C:\ 命令提示符
C:\WINDOWS>DIR
驱动器 C 中的卷没有标签。
卷的序列号是 3CAB-01BF

C:\WINDOWS 的目录

2018-10-27 17:08 <DIR>      .
2018-10-27 17:08 <DIR>      ..
2018-10-27 17:08          0 0.log
2018-10-02 00:42 <DIR>      addins
2018-10-26 17:06 <DIR>      AppPatch
2018-10-26 21:11          0 BcdLog.txt
2008-04-14 20:00      1,272 Blue Lace 16.bmp
2008-04-14 20:00     82,944 clock.avi
2018-10-01 16:50      200 cmsetacl.log
2008-04-14 20:00     17,062 Coffee Bean.bmp
2018-10-27 17:04     290,568 comsetup.log
2018-10-02 00:42 <DIR>      Config
2018-10-02 00:42 <DIR>      Connection Wizard
2018-10-01 16:55          0 control.ini
2018-10-01 16:52 <DIR>      Cursors
2018-10-01 16:46 <DIR>      Debug
2008-04-14 20:00          2 desktop.ini
2018-10-02 00:42 <DIR>      Driver Cache
搜狗 半:
```

②使用/P 参数分面显示目录内容:

```
命令提示符 - DIR /P
C:\WINDOWS>DIR /P
驱动器 C 中的卷没有标签。
卷的序列号是 3CAB-01BF

C:\WINDOWS 的目录
2018-10-27 17:08 <DIR> .
2018-10-27 17:08 <DIR> ..
2018-10-27 17:08 0 0.log
2018-10-02 00:42 <DIR> addins
2018-10-26 17:06 <DIR> AppPatch
2018-10-26 21:11 0 BcdLog.txt
2008-04-14 20:00 1,272 Blue Lace 16.bmp
2008-04-14 20:00 82,944 clock.avi
2018-10-01 16:50 200 cmsetacl.log
2008-04-14 20:00 17,062 Coffee Bean.bmp
2018-10-27 17:04 290,568 comsetup.log
2018-10-02 00:42 <DIR> Config
2018-10-02 00:42 <DIR> Connection Wizard
2018-10-01 16:55 0 control.ini
2018-10-01 16:52 <DIR> Cursors
2018-10-01 16:46 <DIR> Debug
2008-04-14 20:00 2 desktop.ini
2018-10-02 00:42 <DIR> Driver Cache
2018-10-01 16:52 130 DtcInstall.log
请按任意键继续. . .
2018-10-02 00:44 <DIR> ehome
2008-04-14 20:00 978,432 explorer.exe
2008-04-14 20:00 80 explorer.scf
2018-10-27 17:04 840,055 FaxSetup.log
2008-04-14 20:00 16,730 FeatherTexture.bmp
2008-04-14 20:00 17,336 Gone Fishing.bmp
2008-04-14 20:00 26,582 Greenstone.bmp
2018-10-26 17:06 <DIR> Help
2008-04-14 20:00 10,752 hh.exe
2018-10-26 17:01 92,807 ie8.log
2018-10-26 17:01 <DIR> ie8updates
2018-10-26 17:01 55,450 ie8_main.log
2018-10-27 17:04 957,971 iis6.log
2018-10-01 16:47 <DIR> ime
2018-10-27 17:03 1,393 insins.BAK
2018-10-27 17:04 1,393 insins.log
2018-10-02 00:42 <DIR> java
2018-10-26 16:57 49,208 KB2115168.log
2018-10-26 16:56 38,413 KB2229593.log
2018-10-26 16:57 37,995 KB2296011.log
2018-10-27 17:04 17,458 KB2345886.log
2018-10-26 16:55 51,626 KB2347290.log
2018-10-26 14:35 13,328 KB2378111.log
2018-10-26 16:58 42,536 KB2387149.log
请按任意键继续. . .
搜狗 半:
```

③使用/W 参数只显示文件名:

```
命令提示符
C:\WINDOWS>DIR /W
驱动器 C 中的卷没有标签。
卷的序列号是 3CAB-01BF

C:\WINDOWS 的目录

[.]                [..]                0.log
[addins]           [AppPatch]          BcdLog.txt
Blue Lace 16.bmp   clock.avi             cmsetac.log
Coffee Bean.bmp    comsetup.log          [Config]
[Connection Wizard] control.ini           [Cursors]
[Debug]            desktop.ini          [Driver Cache]
DtcInstall.log     [ehome]             explorer.exe
explorer.scf        FaxSetup.log          FeatherTexture.bmp
Gone Fishing.bmp   Greenstone.bmp        [Help]
hh.exe             ie8.log              [ie8updates]
ie8_main.log       iis6.log             [ime]
imsins.BAK         imsins.log            [java]
KB2115168.log      KB2229593.log         KB2296011.log
KB2345886.log      KB2347290.log         KB2378111.log
KB2387149.log      KB2393802.log         KB2419632.log
KB2423089.log      KB2443105.log         KB2478960.log
KB2478971.log      KB2479943.log         KB2481109.log
KB2483185.log      KB2485663.log         KB2506212.log
KB2507938.log      KB2508429.log         KB2509553.log
KB2510531-IE8.log  KB2510581.log         KB2535512.log
KB2536276-v2.log   KB2544893-v2.log      KB2564958.log
KB2566454.log      KB2570947.log         KB2584146.log
KB2585542.log      KB2592799.log         KB2598479.log
KB2603381.log      KB2619339.log         KB2620712.log
KB2631813.log      KB2653956.log         KB2655992.log
KB2659262.log      KB2661637.log         KB2676562.log
KB2686509.log      KB2691442.log         KB2698365.log
KB2705219-v2.log   KB2712808.log         KB2719985.log
KB2723135-v2.log   KB2727528.log         KB2749655.log
KB2757638.log      KB2770660.log         KB2780091.log
KB2802968.log      KB2803821-v2.log      KB2807986.log
KB2813345.log      KB2820917.log         KB2834886.log
KB2847311.log      KB2850869.log         KB2859537.log
KB2862152.log      KB2862330.log         KB2862335.log
KB2864063.log      KB2868626.log         KB2876217.log
KB2876331.log      KB2879017.log         KB2892075.log
KB2893294.log      KB2898715.log         KB2900986.log
KB2904266.log      KB2909212.log         KB2909921-IE8.log
KB2914368.log      KB2916036.log         KB2922229.log
KB2929961.log      KB2930275.log         KB898461.log
KB923561.log       KB946648.log          KB950762.log
KB950974.log       KB951376-v2.log       KB951978.log
KB952004.log       KB952069.log          KB952287.log
KB952954.log       KB954155.log          KB955759.log
KB956572.log       KB956844.log          KB959426.log
KB960803.log       KB960859.log          KB968389.log
KB969059.log       KB970430.log          KB971029.log
KB971657.log       KB972270.log          KB973507.log
KB973540.log       KB973815.log          KB973869.log
KB973904.log       KB974112.log          KB974318.log
KB974392.log       KB974571.log          KB975025.log
KB975467.log       KB975558.log          KB975560.log
KB975713.log       KB977816.log          KB977914.log
KB978338.log       KB978542.log          KB978695.log
KB978706.log       KB979309.log          KB979482.log
KB979687.log       KB981997.log          KB982132.log
KB982665.log       [LLSchemas]          MedCtrOC.log
[Media]            [Minidump]            [msgagent]
[msapps]           msdfmap.ini           msgsocm.log
msmqinst.log       [mui]                 netfxocm.log
[Network Diagnostic] NOTEPAD.EXE            ntbtlog.txt
ntdtcsetup.log     ocgen.log             ocmn.log
ODBCINST.INI       OEWABlog.txt          [Offline Web Pages]
ooobeact.log        [pchealth]            [PeerNet]
Prairie Wind.bmp    [Prefetch]            [Provisioning]
regedit.exe         [Registration]         REGLOCS.OLD
regopt.log          [repair]               [Resources]
Rhododendron.bmp   River Sumida.bmp      Rule.dat
Santa Fe Stucco.bmp SchedLgU.Txt           [security]
sessmgr.setup.log   SET3.tmp               SET4.tmp
SET8.tmp            setupact.log           setupapi.log
setuperr.log        setuplog.txt           Soap Bubbles.bmp
[SoftwareDistribution] spupdsvc.log           [srchasst]
Sti_Trace.log       [system]               system.ini
[system32]          tabletoc.log           TASKMAN.EXE
[Temp]              tsoc.log               twain.dll
[twain_32]          twain_32.dll           twunk_16.exe
twunk_32.exe        updspapi.log           vb.ini
vbaddin.ini         vmnreg32.dll           [WBEM]
[Web]               wiadefug.log           wiaserve.log
win.ini             WindowsUpdate.log      winhelp.exe
winhlp32.exe        [WinSxS]               wmpfrchs.prx
wmsetup.log         WMSysPr9.prx           Zapotec.bmp
_default.pif

211 个文件      29,284,114 字节
39 个目录      541,204,705,280 可用字节

搜狗 半:
```

(5) PATH——设置可执行文件的搜索路径:

显示目前所设的路径:

```
C:\FOX>PATH
PATH==D:\Program Files\Java\JDK8\bin;D:\Program Files\Java\JDK8\jre\bin;D:\kotli
nc\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;D:\MASM;
```

(6) TREE——显示磁盘目录结构:

①显示文件夹的目录结构:

```
D:\ytb>TREE
文件夹 PATH 列表
卷序列号为 F15F-33E0
D:.
├──3
└──4
```

②使用/F 参数, 显示目录和目录下的文件:

```
D:\ytb>TREE /F
文件夹 PATH 列表
卷序列号为 F15F-33E0
D:.
├──DEBUG.EXE
├──EDIT.COM
├──LINK.EXE
├──MASM.EXE
├──3
│   ├──31.asm
│   └──32.asm
└──4
    ├──41.asm
    ├──41.EXE
    ├──41.OBJ
    ├──42.asm
    ├──42.EXE
    ├──42.OBJ
    ├──43.asm
    ├──43.EXE
    └──43.OBJ
```

2. 文件操作类命令:

(1) COPY——文件复制:

①复制 D:\MASM 中的所有文件到自己的文件夹:

```
D:\MASM>COPY *.* D:\ytb
DEBUG.EXE
EDIT.COM
LINK.EXE
MASM.EXE
已复制          4 个文件。
```

②从键盘上输入数据建立文件:

```
D:\ytb>COPY CON file.txt
dos
^Z
已复制          1 个文件。
```

(2) TYPE——显示文件内容:

①显示 file.txt 的内容:

```
D:\ytb>TYPE file.txt
dos
```

②分屏显示内容:

```
C:\WINDOWS\system32>TYPE eula.txt!MORE
Microsoft 软件最终用户许可协议

MICROSOFT WINDOWS XP PROFESSIONAL EDITION
SERVICE PACK 3

重要须知 - 请仔细阅读: 本最终用户许可协议 (《
协议》) 是您 (个人或单一实体) 与 Microsoft
Corporation ('Microsoft') 或其附属实体之一之间
就本《协议》随附的 Microsoft 软件达成的法律协议
, 其中包括计算机软件, 并可能包括相关介质、印刷资
料、'联机' 或电子文档和基于因特网的服务 ('软件')
。本《协议》的一份修正条款或补充条款可能随 '软件'
一起提供。

自 Windows XP Service Pack 2 首次发布以来, 其
中部分条款已经变更。变更内容包括:

* 关于该软件验证功能的更多信息, 用于确定该软件
是盗版的、未获得适当许可的, 还是非正版 Windows
产品

* 有关基于 Internet 的服务的更多隐私披露信息

本列表重点说明了其中一些变更。使用该软件须遵守
下列条款。

您一旦安装、复制或以其他方式使用 '软件', 即表示
您同意接受本《协议》各项条款的约束。如果您不同意
本《协议》中的条款, 请不要安装、复制或使用 '软件'
; 您可在适用的情况下将其退回原购买处, 并获得全
额退款。

-- More --
搜狗 半:
```

(3) REN——文件改名:

将 file.txt 改名为 test.text:

```
D:\yqh>REN file.txt test.text
```

(4) DEL——删除文件:

①删除当前目录下的所有 exe 文件:

```
D:\yqh>DEL *.exe
```

②使用/P 参数删除前询问:

```
D:\yqh>DEL /P test.text
D:\yqh\test.text, 要删除(Y/N)吗? Y
```

3. 其它命令:

(1) CLS——清屏幕:

(2) VER——查看系统版本号:

```
D:\yqh>VER
Microsoft Windows XP [版本 5.1.2600]
```

(3) DATE——日期设置:

①设置系统日期为 2018 年 2 月 1 日:

```
D:\ytb>DATE 18-2-1
```

②显示系统时间并设置新日期:

```
D:\ytb>DATE
当前日期: 2018-02-01 星期四
输入新日期: <年月日> 2018-11-9
```

三、实验心得:

通过本次实验,我强化了对 DOS 命令的理解,学习了一些常用命令(如: COPY 命令从键盘输入数据建立文件)不曾使用过的用法,学习这些 DOS 命令,为进一步学习书写批处理程序让计算机自动解决一些问题做了铺垫,也为后续在 Windows 下利用 DEBUG 调试汇编程序奠定了基础。

实验报告二

一、实验目的：

了解什么是 Debug，学习使用 Debug 的基本功能，熟练使用 Debug 调试汇编程序。

二、实验任务：

1. 使用 Debug，将下面的程序段写入内存，逐条执行，观察每条指令执行后 CPU 中相关寄存器的内容变化。

机器码	汇编指令	
b8 20 4e	mov ax, 4e20h	; (ax)=4e20h
05 16 14	add ax, 1416h	; (ax)=(ax)+1416h
Bb 00 20	mov bx, 2000h	; (bx)=2000h
01 d8	add ax, bx	; (ax)=(ax)+(bx)
89 c3	mov bx, ax	; (bx)=(ax)
01 d8	add ax, bx	; (ax)=(ax)+(bx)
b8 1a 00	mov ax, 001ah	; (ax)=001ah
bb 26 00	mov bx, 0026h	; (bx)=0026
00 d8	add al, bl	; (al)=(al)+(bl)
00 dc	add ah, bl	; (ah)=(ah)+(bl)
00 c7	add bh, al	; (bh)=(bh)+(al)
b4 00	mov ah, 0	; (ah)=0
00 d8	add al, bl	; (al)=(al)+(bl)
04 9c	add al, 9ch	; (al)=(al)+9ch

提示，可用 E 命令和 A 命令以两种方式将指令写入内存。注意用 T 命令执行时，CS:IP 的指向。

2. 将下面 3 条指令写入从 2000:0 开始的内存单元中，利用 3 条指令计算 2 的 8 次方。

```
mov ax, 1
add ax, ax
jmp 2000: 0003
```

3. 查看内存中的内容：

PC 机主板上的 ROM 中写有一个生产日期，在内存 FFF00H~FFFFFH 的某几个单元中，请找到这个生产日期并试图改变它。

提示，如果读者对实验的结果感到疑惑，请仔细阅读第 1 章中的 1.15 节。

4. 向内存从 B8100h 开始的单元中填入数据，如：

```
-e B810:0000 01 01 02 02 03 03 04 04
```

请读者先填写不同的数据，观察产生的现象；再改变填写的地址，观察产生的现象。

提示，如果读者对实验的结果感到疑惑，请仔细阅读第 1 章、中的 1.15 节。

三、实验过程：

1. 向内存中输入相应的指令，用 t 命令单步执行，记录每次 CPU 寄存器中的内容：


```

C:\ 命令提示符 - debug
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00B7  NU UP EI PL NZ NA PO NC
0B05:00B7 B8204E      MOV     AX,4E20
-t
AX=4E20 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00BA  NU UP EI PL NZ NA PO NC
0B05:00BA 051614      ADD     AX,1416
-t
AX=6236 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00BD  NU UP EI PL NZ NA PE NC
0B05:00BD BB0020      MOV     BX,2000
-t
AX=6236 BX=2000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00C0  NU UP EI PL NZ NA PE NC
0B05:00C0 01D8      ADD     AX,BX
-t
AX=8236 BX=2000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00C2  OU UP EI NG NZ NA PE NC
0B05:00C2 89C3      MOV     BX,AX

```

```

C:\ 命令提示符 - debug
AX=8236 BX=8236 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00C4  OU UP EI NG NZ NA PE NC
0B05:00C4 01D8      ADD     AX,BX
-t
AX=046C BX=8236 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00C6  OU UP EI PL NZ NA PE CY
0B05:00C6 B81A00      MOV     AX,001A
-t
AX=001A BX=8236 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00C9  OU UP EI PL NZ NA PE CY
0B05:00C9 BB2600      MOV     BX,0026
-t
AX=001A BX=0026 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00CC  OU UP EI PL NZ NA PE CY
0B05:00CC 00D8      ADD     AL,BL
-t
AX=0040 BX=0026 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00CE  NU UP EI PL NZ AC PO NC
0B05:00CE 00DC      ADD     AH,BL

```

```

C:\ 命令提示符 - debug
AX=2640 BX=0026 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00D0  NU UP EI PL NZ NA PO NC
0B05:00D0 00C7      ADD     BH,AL
-t
AX=2640 BX=4026 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00D2  NU UP EI PL NZ NA PO NC
0B05:00D2 B400      MOV     AH,00
-t
AX=0040 BX=4026 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00D4  NU UP EI PL NZ NA PO NC
0B05:00D4 00D8      ADD     AL,BL
-t
AX=0066 BX=4026 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00D6  NU UP EI PL NZ NA PE NC
0B05:00D6 049C      ADD     AL,9C
-t
AX=0002 BX=4026 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B05 ES=0B05 SS=0B05 CS=0B05 IP=00D8  NU UP EI PL NZ AC PO CY
0B05:00D8 0000      ADD     [BX+SI],AL      DS:4026=65

```

2. 通过多次执行命令，每次寄存器 ax 的值翻倍，可以得到 2 的 8 次方为 100h:

```

C:\ 命令提示符 - debug
AX=0040 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B06 ES=0B06 SS=0B06 CS=2000 IP=0005  NU UP EI PL NZ NA PO NC
2000:0005 EBFC JMP 0003
-t
AX=0040 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B06 ES=0B06 SS=0B06 CS=2000 IP=0003  NU UP EI PL NZ NA PO NC
2000:0003 01C0 ADD AX,AX
-t
AX=0080 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B06 ES=0B06 SS=0B06 CS=2000 IP=0005  NU UP EI PL NZ NA PO NC
2000:0005 EBFC JMP 0003
-t
AX=0080 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B06 ES=0B06 SS=0B06 CS=2000 IP=0003  NU UP EI PL NZ NA PO NC
2000:0003 01C0 ADD AX,AX
-t
AX=0100 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B06 ES=0B06 SS=0B06 CS=2000 IP=0005  NU UP EI PL NZ NA PE NC
2000:0005 EBFC JMP 0003

```

3. 通过调用 d 命令可以看到生产日期为：2017 年 5 月 19 日

```

-d fff0:f0 ff
FFF0:00F0 EA 5B E0 00 F0 30 35 2F-31 39 2F 31 37 00 FC 6B .[...05/19/17..k

```

通过 e 命令改变地址为 FFF0~FFFF 的内容，d 命令查看，正常情况其值不会被改变，但由于此次是在虚拟机中运行，其值有变化：

```

-d fff0:f0 ff
FFF0:00F0 01 02 03 04 F0 30 35 2F-31 39 2F 31 37 00 FC 6B .....05/19/17..k
-e fff0:f0
FFF0:00F0 01.0 02.1 03.2 04.3 F0.4 30.5 35.6 2F.7
-d fff0:f0 ff
FFF0:00F0 00 01 02 03 04 05 06 07-31 39 2F 31 37 00 FC 6B .....19/17..k

```

4. 修改 b800:0000 处的内容，发现在控制台的左上角会发现彩色的字符：

```

C:\ 命令提示符 - debug
hello worldff
FFF0:00F0 EA 5B E0 00 F0 30 35 2F-31 39 2F 31 37 00 FC 6B .[...05/19/17..k
-e fff0:f0
FFF0:00F0 EA.1 5B.2 E0.3 00.4
-d fff0:f0
FFF0:00F0 01 02 03 04 F0 30 35 2F-31 39 2F 31 37 00 FC 6B .....05/19/17..k
FFF0:0100 34 12 00 00 00 00 00 00-00 00 00 00 00 00 00 4.....
FFF0:0110 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
FFF0:0120 70 00 2E 8E 06 30 00 BF-7F 01 B9 02 00 AB 47 47 p.....GG
FFF0:0130 E2 FB CB 56 50 51 52 57-55 1E 06 53 8B EC 8B 76 ...UPQRMU..S..v
FFF0:0140 12 2E 8E 1E 30 00 8B 44-02 A2 22 00 88 26 08 01 ...0..D..".&..
FFF0:0150 8B 34 C4 1E 18 00 26 8A-47 01 26 8A 67 0D 26 8B .4....&.G.&.g.&.
FFF0:0160 4F 12 26 8B 57 14 97 26-8A 47 02 2E 3A 04 73 2C 0.&.W...&.G...s.
-d fff0:f0 ff
FFF0:00F0 01 02 03 04 F0 30 35 2F-31 39 2F 31 37 00 FC 6B .....05/19/17..k
-e fff0:f0
FFF0:00F0 01.0 02.1 03.2 04.3 F0.4 30.5 35.6 2F.7
-d fff0:f0 ff
FFF0:00F0 00 01 02 03 04 05 06 07-31 39 2F 31 37 00 FC 6B .....19/17..k
-e b810:0000
B810:0000 30.
-e b800:0000 'h e l l o w o r l d '

```

四、实验总结：

在 Debug 中：

- R 命令：查看，修改 CPU 中寄存器的内容
- D 命令：查看内存中的内容
- E 命令：修改内存中的内容

U 命令：将内存中的内容解释为机器指令和对应的汇编指令

T 命令：执行 **CS:IP** 指向的内存单元处的命令

A 命令：以汇编指令的形式向内存中写入指令

通过熟练掌握这些命令，为我们后续编程打下基础。

实验报告三

一、实验目的：

通过实验了解 DOS 常用命令，汇编就是把用汇编语言编写的源程序翻译（汇编）成机器语言的目标程序。汇编程序可以使用小汇编程序（ASM）也可以用宏汇编程序（MASM），用于宏汇编程序不但可以代替 ASM，而且可以汇编具有宏定义的汇编程序，因此我们在汇编程序时使用宏汇编程序（MASM）。开发汇编语言源程序的主要步骤有哪些？首先用 EDIT 等编辑程序产生汇编语言的源程序，源程序是用汇编语言的语句编写的且不能为机器所识别的程序，所以要经过汇编程序加以翻译，因此汇编程序的作用就是把源文件转换成用二进制代码表示的目标文件（称为 OBJ 文件）。在转换的过程中，如果源程序中有语法错误，则汇编结束后，汇编程序将指出源程序中的错误信息，如非法格式，未定义的助记符、标号，漏掉操作数等。用户还可以用编辑程序来修改源程序中的错误，最后得到无语法错误的目标文件。目标文件虽然已经是二进制文件，但它还不能直接上机运行，必须经过连接程序（LINK）把目标文件与库文件或其他目标文件连接在一起形成可执行文件（EXE 文件），才可以在机器上运行。

二、实验任务：

1. dos 命令建立目录拷贝文件（edit.com、masm.exe、link.exe、debug.exe）执行宏汇编程序。
2. 通过调用程序 DEBUG 的主要命令，熟悉各种命令的用法。

三、实验过程：

1. 将下面的程序保存为 1.asm，将其生成可执行文件 1.exe。

```
assume cs: codesg
codesg SEGMENT
    mov ax, 2000h
    mov ss, ax
    mov sp, 10h
    add sp, 4
    push ax
    push bx
    pop ax
    pop bx
    mov ax, 4c00h
    int 21h
codesg ends
end
```

- (1) 编辑 1.asm:

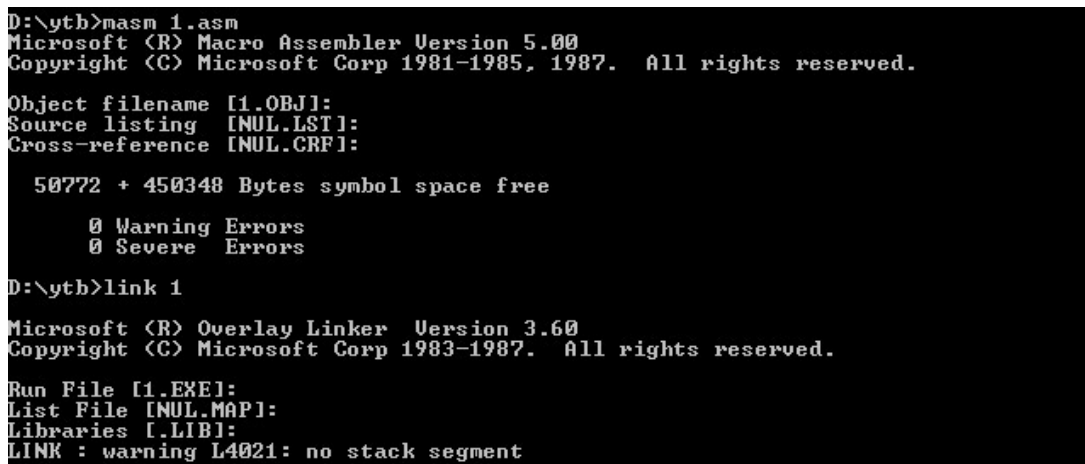


```
命令提示符 - edit 1.asm
File Edit Search View Options Help
D:\ytb\1.asm

assume cs: codesg
codesg SEGMENT
    mov ax, 2000h
    mov ss, ax
    mov sp, 10h
    add sp, 4
    push ax
    push bx
    pop ax
    pop bx
    mov ax, 4c00h
    int 21h
codesg ends
end
```

F1=Help | Line:15 Col:1

(2) 使用 MASM 编译生成目标文件 1.obj, 使用 LINK 将目标程序连接定位, 生成可执行文件 1.exe:



```
D:\ytb>masm 1.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Object filename [1.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

    50772 + 450348 Bytes symbol space free

    0 Warning Errors
    0 Severe Errors

D:\ytb>link 1

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

Run File [1.EXE]:
List File [NUL.MAP]:
Libraries [LIB]:
LINK : warning L4021: no stack segment
```

2. 使用 DEBUG 跟踪 1.exe, 写出每一步执行后, 相关寄存器的内容。

(1) 使用 DEBUG 跟踪 1.exe:

```

D:\y\y\y>debug 1.exe
-u
0067:0000 B80020      MOV     AX,2000
0067:0003 8ED0          MOV     SS,AX
0067:0005 BC1000      MOV     SP,0010
0067:0008 83C404      ADD     SP,+04
0067:000B 50          PUSH    AX
0067:000C 53          PUSH    BX
0067:000D 58          POP     AX
0067:000E 5B          POP     BX
0067:000F B8004C      MOV     AX,4C00
0067:0012 CD21          INT     21
0067:0014 5E          POP     SI
0067:0015 8BE5      MOV     SP,BP
0067:0017 5D          POP     BP
0067:0018 C3          RET
0067:0019 90          NOP
0067:001A 55          PUSH    BP
0067:001B 8BEC      MOV     BP,SP
0067:001D 81EC9000    SUB     SP,0090
-r
AX=0000 BX=0000 CX=0014 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B57 ES=0B57 SS=0B67 CS=0B67 IP=0000  NU UP EI PL NZ NA PO NC
0067:0000 B80020      MOV     AX,2000
-

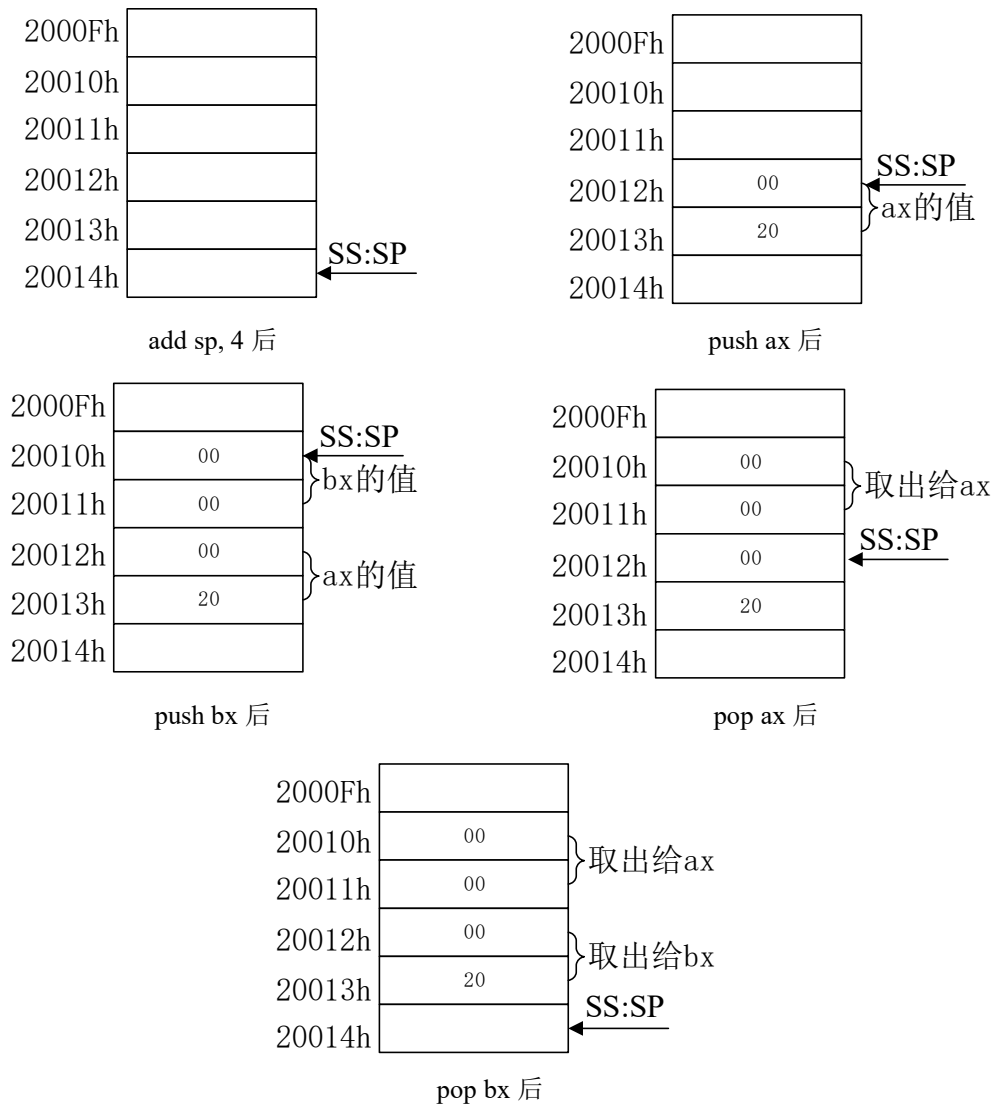
```

(2) 记录每一步执行后相关寄存器的内容

行号	机器码	字节数	汇编指令	汇编指令执行后寄存器的内容			
				ax	bx	ip	sp
1	B80020	3	MOV AX, 2000	2000	0000	0000	0000
2	8ED0	2	MOV SS, AX				
3	BC1000	3	MOV SP, 0010	2000	0000	0008	0010
4	83C404	3	ADD SP, +04	2000	0000	000B	0014
5	50	1	PUSH AX	2000	0000	000C	0012
6	53	1	PUSH BX	2000	0000	000D	0010
7	58	1	POP AX	0000	0000	000E	0012
8	5B	1	POP BX	0000	2000	000F	0014
9	B8004C	3	MOV AX, 4C00	4C00	2000	0012	0014
10	CD21	2	INT 21				

在 Debug 中用 T 命令执行第二行语句 mov ss, ax 后，它的下一条指令 mov sp, 10h 并没有显示，这是因为 Debug 的 T 命令在执行修改寄存器 SS 的指令时，下一条指令也紧接着被执行。

下面以压栈图的形式展示栈操作的过程：



上述过程中, **push ax** 后, $SP=SP-2$, 将 **ax** 的值送入 **SS:SP** 指向的内存单元; **push bx** 后, $SP=SP-2$, 将 **bx** 的值送入 **SS:SP** 指向的内存单元; **pop ax** 后, 先取出 **SS:SP** 指向内存单元处的一个字放入 **ax**, 然后 $SP=SP+2$; **pop bx** 后, 先取出 **SS:SP** 指向内存单元处的一个字放入 **bx**, 然后 $SP=SP+2$ 。

经过以上程序, 交换了寄存器 **ax** 和 **bx** 中的值, **IP** 每次增加的值是执行汇编指令对应的机器码的字节数。

实验报告四

一、实验目的：

学习计算机中的数制，记住常用字符的 ASCII 码。

二、实验任务：

1. 下面的程序的功能是将 `mov ax, 4c00h` 之前的指令复制到内存 0:200 处，上机调试，跟踪运行结果。

```
assume cs: code
code segment
    mov ax, cs
    mov ds, ax      ;将代码段作为数据段
    mov ax, 0020h
    mov es, ax      ;(es)=0020h
    mov bx, 0
    mov cx, offset a
s:  mov al, [bx]
    mov es: [bx], al ;将 ds: [bx]复制到 es: [bx]
    inc bx
    loop s
a:  mov ax, 4c00h
    int 21h
code ends
end
```

使用 `u` 命令进行反汇编，`d` 命令查看 0:200 处内容并记录。

2. 编程：向内存 0:200-0:023f 依次传送 0-63 (3fh)。程序只能使用 9 条指令，9 条指令中包括 `mov ax, 4c00h`, `int 21h` 不包括伪指令。

3. 编写一个汇编程序，要求从键盘输入一个小写字母，将其转换成大写字母在屏幕上显示出来。

二、实验过程：

1.

(1) 开始时使用 `d` 命令查看内存 0020:0 处全为 0:

```
命令提示符 - debug 41.EXE
D:\yhb\4>debug 41.EXE
-d 20:0
0020:0000  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0010  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0020  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0030  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0040  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0050  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0060  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0070  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-
```

(2) 使用 u 命令进行反汇编，记录机器码：

```
命令提示符 - debug 41.EXE
0020:0010  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0020  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0030  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0040  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0050  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0060  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0020:0070  00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-u
0B6B:0000  8CC8      MOV     AX,CS
0B6B:0002  8ED8      MOV     DS,AX
0B6B:0004  B82000    MOV     AX,0020
0B6B:0007  8EC0      MOV     ES,AX
0B6B:0009  BB0000    MOV     BX,0000
0B6B:000C  B91700    MOV     CX,0017
0B6B:000F  8A07      MOV     AL,[BX]
0B6B:0011  26        ES:
0B6B:0012  8807      MOV     [BX],AL
0B6B:0014  43        INC     BX
0B6B:0015  E2F8      LOOP    000F
0B6B:0017  B8004C    MOV     AX,4C00
0B6B:001A  CD21      INT     21
0B6B:001C  8B7F04    MOV     DI,[BX+04]
0B6B:001F  2E        CS:
0B6B:0020  893E4E91 MOV     [914E],DI
-
```

(3) 连续使用 t 命令单步执行，执行到 LOOP 指令使用 p 命令跳过整个循环：

```
命令提示符 - debug 41.EXE
AX=0020 BX=0000 CX=0017 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B6B ES=0020 SS=0B6B CS=0B6B IP=000F  NU UP EI PL NZ NA PO NC
0B6B:000F  8A07      MOV     AL,[BX]          DS:0000=8C
-t
AX=008C BX=0000 CX=0017 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B6B ES=0020 SS=0B6B CS=0B6B IP=0011  NU UP EI PL NZ NA PO NC
0B6B:0011  26        ES:
0B6B:0012  8807      MOV     [BX],AL          ES:0000=00
-t
AX=008C BX=0000 CX=0017 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B6B ES=0020 SS=0B6B CS=0B6B IP=0014  NU UP EI PL NZ NA PO NC
0B6B:0014  43        INC     BX
-t
AX=008C BX=0001 CX=0017 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B6B ES=0020 SS=0B6B CS=0B6B IP=0015  NU UP EI PL NZ NA PO NC
0B6B:0015  E2F8      LOOP    000F
-p
AX=00F8 BX=0017 CX=0000 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B6B ES=0020 SS=0B6B CS=0B6B IP=0017  NU UP EI PL NZ NA PE NC
0B6B:0017  B8004C    MOV     AX,4C00
-
```

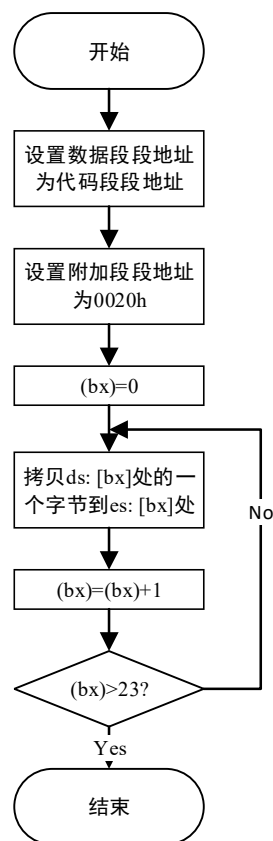
(4) 再次查看内存，可以看到，和第（2）步中记录的机器码相同：

```

C:\ 命令提示符 - debug 41.EXE
-t
AX=008C BX=0000 CX=0017 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B6B ES=0020 SS=0B6B CS=0B6B IP=0014  NU UP EI PL NZ NA PO NC
0B6B:0014 43          INC     BX
-t
AX=008C BX=0001 CX=0017 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B6B ES=0020 SS=0B6B CS=0B6B IP=0015  NU UP EI PL NZ NA PO NC
0B6B:0015 E2F8      LOOP    000F
-p
AX=00F8 BX=0017 CX=0000 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B6B ES=0020 SS=0B6B CS=0B6B IP=0017  NU UP EI PL NZ NA PE NC
0B6B:0017 B8004C     MOV     AX,4C00
-d 0020:0
0020:0000 8C C8 8E D8 B8 20 00 8E-C0 BB 00 00 B9 17 00 8A .....
0020:0010 07 26 88 07 43 E2 F8 00-00 00 00 00 00 00 00 00 .&...C.....
0020:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0020:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0020:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0020:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0020:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0020:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....

```

(4) 该代码的流程图如下：



2.

(1) 代码如下：

```

assume cs: code
code segment
    mov ax, 0020h
    mov ds, ax    ;设置数据段地址为 0020h

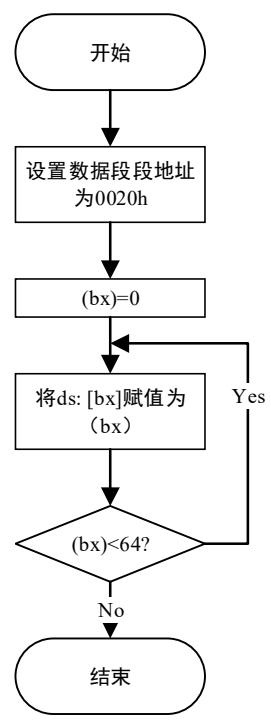
```

```
mov bx, 0
mov cx, 0040h ;循环 64 次
s: mov [bx], bx
inc bx
loop s
mov ax, 4c00h
int 21h
code ends
end
```

(2) 程序结束前使用 d 命令查看相应位置的内存：

```
-d 0020:0 3f
0020:0000  00 01 02 03 04 05 06 07-08 09 0A 0B 0C 0D 0E 0F  .....
0020:0010  10 11 12 13 14 15 16 17-18 19 1A 1B 1C 1D 1E 1F  .....
0020:0020  20 21 22 23 24 25 26 27-28 29 2A 2B 2C 2D 2E 2F  !"#$%&'(<)*+,-./
0020:0030  30 31 32 33 34 35 36 37-38 39 3A 3B 3C 3D 3E 3F  0123456789:;<=>?
```

(3) 该代码的流程图如下：



3.

(1) 进制和 ASCII 码的一些相关知识：

表 1-1 十进制数、二进制数及十六进制数对照表

十进制	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
二进制	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
十六进制	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

表 1-4 常用字符的 ASCII 码

字符	ASCII 码 (H)	字符	ASCII 码 (H)
0~9	30~39	\$	24
A~Z	41~5A	换行 LF	0A
a~z	61~7A	回车 CR	0D
Blank(space)	20		

(2) 代码如下:

```
STACK SEGMENT
    DB 200 DUP(0)
STACK ENDS
CODE SEGMENT
ASSUME CS: CODE, SS: STACK
BEGIN:
    MOV AH, 1
    INT 21H          ;从键盘输入一个小写字母放入 AL
    SUB AL, 20H      ;小写字母转大写字母
    MOV DL, AL
    MOV AH, 2
    INT 21H          ;在屏幕上显示 DL 的内容
    MOV AH, 4CH
    INT 21H          ;程序返回
CODE ENDS
END BEGIN
```

(3) 测试成功输入小写字母 a 输出大写字母 A:



(4) 该程序的流程图如下:



实验报告五

一、实验目的：

学习了解从键盘上输入字符、在显示器上输出字符的方法。

二、实验任务：

1. 设计程序，要求从键盘上逐一输入字符，并在显示器上输出，当输入到“\$”时，则停止操作。

2. 编程：在已知 BUF 为首地址的字节存储区中，存放着一个以“\$”作结束标志的字符串。编程在显示器上显示该字符，并要求将小写字母以大写字母的形式显示出来。

三、实验代码：

1.

```
CODE SEGMENT
    ASSUME CS: CODE
G1:
    MOV AH, 1
    INT 21H      ;从键盘读入字符送入 AL
    CMP AL, '$'
    JZ EXIT      ;若字符为$结束输入
    MOV DL, AL
    MOV AH, 2    ;输出字符
    INT 21H
    JMP G1       ;跳转 G1 继续输入
EXIT:
    MOV AH, 4CH
    INT 21H
CODE ENDS
    END G1
```

2.

```
DATA SEGMENT
    BUF DB 'Hello', 0Dh, 0Ah, 'END$'
DATA ENDS
STACK SEGMENT
    DB 100 DUP (0)
STACK ENDS
CODE SEGMENT
    ASSUME CS: CODE, DS: DATA, SS: STACK
BEGIN:
    MOV AX, DATA
    MOV DS, AX
    LEA BX, BUF
```



```

LA:
    MOV DL, [BX]    ;(DL)=([BX])
    CMP DL, '$'
    JZ EXIT        ;若 DL 中的字符是$则结束
    CMP DL, 'a'
    JB K
    CMP DL, 'z'
    JA K          ;若 DL 中的字符不是小写字母, 转 K
    SUB DL, 20H    ;小写转大写
K:
    MOV AH, 2
    INT 21H        ;显示字符
    INC BX
    JMP LA
EXIT:
    MOV AH, 4CH
    INT 21H
CODE ENDS
    END BEGIN

```

四、实验记录:

1.

```

C:\ 命令提示符 - debug 31.EXE
0B6B:0002 CD21          INT     21
-p
$
AX=0124 BX=0000 CX=0014 DX=0061 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B5B ES=0B5B SS=0B6B CS=0B6B IP=0004  NU UP EI PL NZ AC PO NC
0B6B:0004 3C24          CMP     AL,24
-t
AX=0124 BX=0000 CX=0014 DX=0061 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B5B ES=0B5B SS=0B6B CS=0B6B IP=0006  NU UP EI PL ZR NA PE NC
0B6B:0006 7408          JZ      0010
-t
AX=0124 BX=0000 CX=0014 DX=0061 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B5B ES=0B5B SS=0B6B CS=0B6B IP=0010  NU UP EI PL ZR NA PE NC
0B6B:0010 B44C          MOV     AH,4C
-t
AX=4C24 BX=0000 CX=0014 DX=0061 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B5B ES=0B5B SS=0B6B CS=0B6B IP=0012  NU UP EI PL ZR NA PE NC
0B6B:0012 CD21          INT     21
-p
Program terminated normally

```

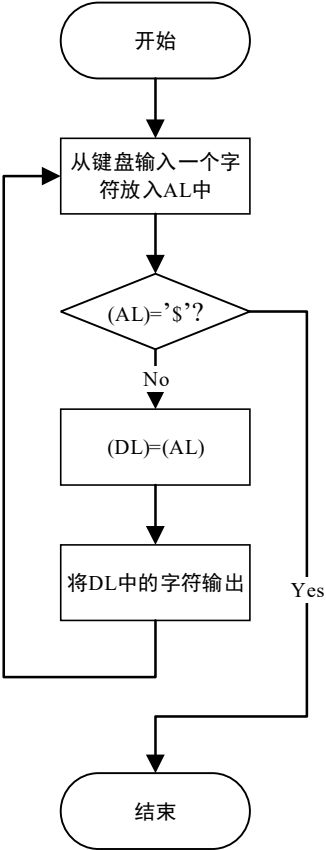
2. 显示缓冲区中的信息:

```

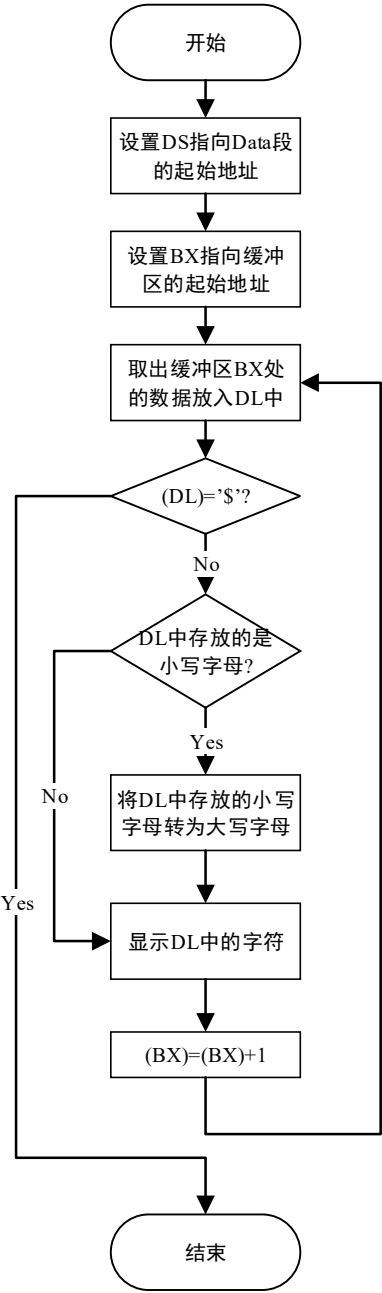
-d ds:0
0B6B:0000 48 65 6C 6C 6F 0D 0A 45-4E 44 24 00 00 00 00 00 Hello..END$.
0B6B:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B6B:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B6B:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B6B:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B6B:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B6B:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0B6B:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....

```

五、程序框图：



任务 1 流程图



任务 2 流程图