实验一

(1/2) DEBUG 调试功能及其操作

一. 基础性实验

1. 实验目的

- 1. 熟悉 DEBUG 的功能,运行并掌握其常用命令.
- 2. 掌握用 DEBUG 调试程序的方法.

2. 实验内容

DEBUG 的加载及其常用命令(DEFRGATUNLWQ)的使用情况.

3. 实验仪器设备

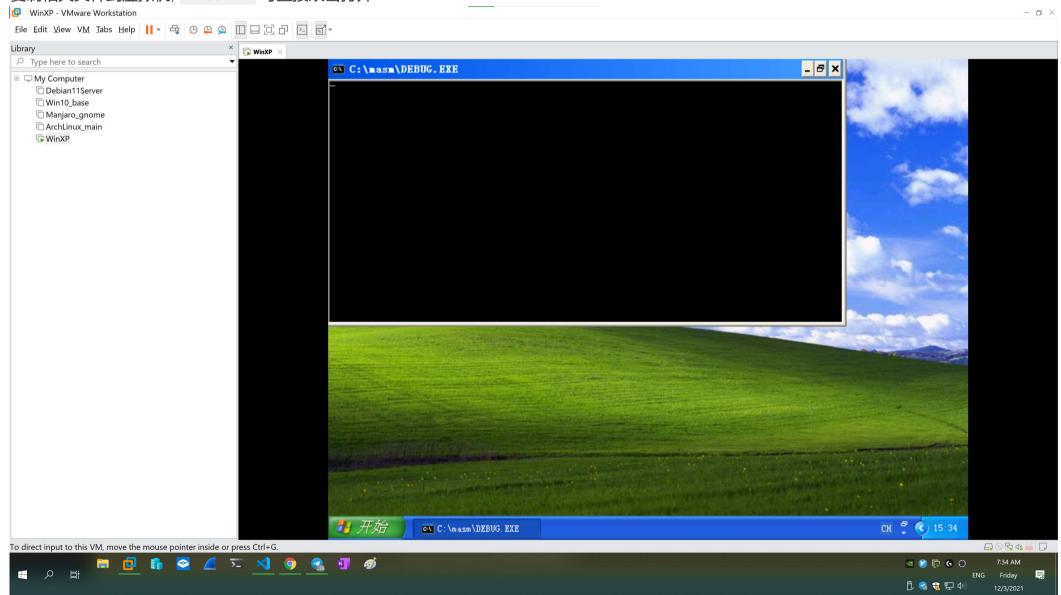
- 硬件环境: AMD Ryzen 9.
- 软件环境: Windows XP Pro SP3 (in VMware Workstation 16 Pro).
- 调试程序: DEBUG.EXE.

4. 实验步骤

(1) 进入Debug程序环境

在 VMware Workstation 16 Pro 中使用 Windows XP Pro SP3 镜像(SHA256 39430C2B8DD5C21BBD5AF9116573F8C574AE896CE31D47280914EF268F01E33F) 安装虚拟机.

复制相关文件到虚拟机, DEBUG.EXE 可直接双击打开.



也可使用 Dosbox ,相关配置如下.

```
[sd1]
fullscreen=false
fulldouble=false
fullresolution=3840x1920
windowresolution=1920x1080
output=overlay
autolock=true
sensitivity=100
waitonerror=true
priority=higher,normal
mapperfile=mapper-0.74.map
usescancodes=true
```

输入程序命令:

```
-a
mov ax,1234
mov bx,1111
add ax,bx
```

(2) 运行程序命令

```
-t
-t
-t
-t
-t=100
-t=100,3
```



-g=100,108

```
_ 🗆 ×
                C:\masm\DEBUG.EXE
实验&工具
               ØADC:0108 DØF6
                                                     ???
                                                                 DH,1
                 t = 100
               AX=1234 BX=1111
DS=0ADC ES=0ADC
0ADC:0103 BB1111
-t=100,3
                                                         DX=0000 SP=1
CS=0ADC IP=0
J BX,1111
                                           CX=0000
SS=0ADC
                                                                       SP=FFEE
IP=0103
                                                                                      BP=0000 SI=0000 DI=0000
                                                                                       NU UP EI PL NZ NA PO NC
                                                     MOU
 2
Dev-C++
                                                                        SP=FFEE
IP=0103
                                                                                      BP=0000 SI=0000 DI=0000
                                                                                       NU UP EI PL NZ NA PO NC
               AX=1234 BX=1111 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0106 NV UP EI PL NZ NA PO NC
0ADC:0106 01D8 ADD AX,BX
 7
  measure
               AX=2345 BX=1111 CX=0000 DX=0000 SP=FFEE
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0108
0ADC:0108 D0F6 ??? DH,1
                                                                                     BP=0000 SI=0000 DI=0000
                                                                                       NU UP EI PL NZ NA PO NC
                -g=100,108
               AX=2345 BX=1111
DS=0ADC ES=0ADC
0ADC:0108 D0F6
                                           CX=0000 DX=0000 SP=FFEE
SS=0ADC CS=0ADC IP=0108
??? DH,1
                                                                                      BP=0000 SI=0000 DI=0000
                                                                                       NU UP EI PL NZ NA PO NC
```

```
-n qwq

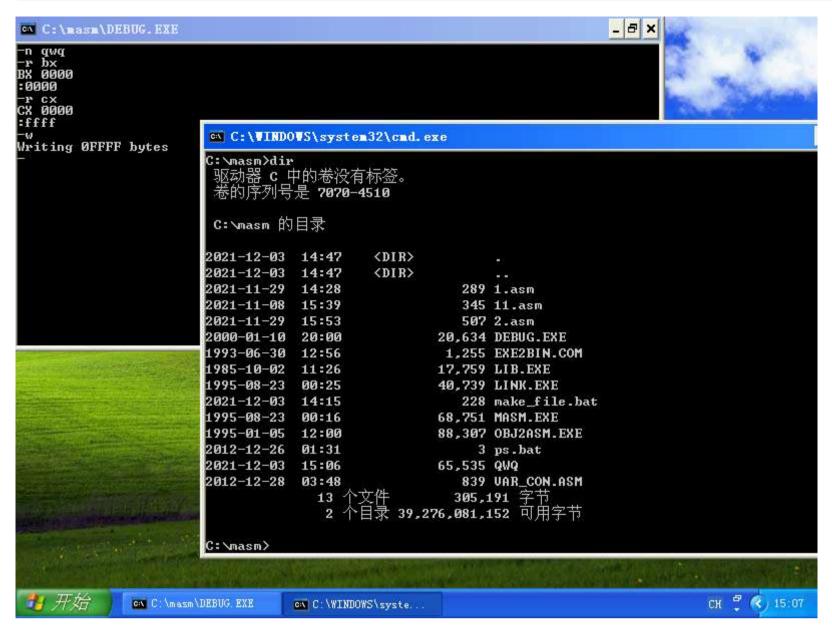
-r bx

:0000

-r cx

:ffff

-w
```



(3) 装入文件命令

```
-n qwq
-l
```

(4) 反汇编

-u

(5) 显示存贮单元命令

-d 100 140

```
CX C:\masm\DEBUG.EXE
                                                                                                                                                                                                                                   _ 🗆 ×
文件利
           -u
60ADC:0100 7303
0ADC:0102 E94EFF
0ADC:0105 2BC8
0ADC:0107 74D0
0ADC:0109 F606159980
0ADC:010E 7439
0ADC:0110 F606159920
0ADC:0115 750A
0ADC:0117 803ED29900
0ADC:011C 3400
0ADC:011E CB
                                                                                                         0105
0053
CX,AX
00D9
其它位
                                                                                  JMP
SUB
                                                                                   JZ
TEST
                                                                                                         BYTE PTR [9915],80
0149
BYTE PTR [9915],20
0121
       퐢
                                                                                   JZ
TEST
JNZ
       퐗
                                                                                                          BYTE PTR [99D2],00
                                                                                   XOR
                                                                                                          AL,00
            ### RETF

@ADC:@11F @AB8BA@D OR BH,[BX+SI+@DBA]

-d 100 140

@ADC:@110 73 @3 E9 4E FF 2B C8 74-D0 F6 @6 15

@ADC:@110 F6 @6 15 99 20 75 @A 80-3E D2 99 @0

@ADC:@120 B8 BA @D 8C EB 23 33 D2-87 D1 B8 @1

@ADC:@130 DF 99 89 16 E1 99 80 3E-C5 96 @0 74

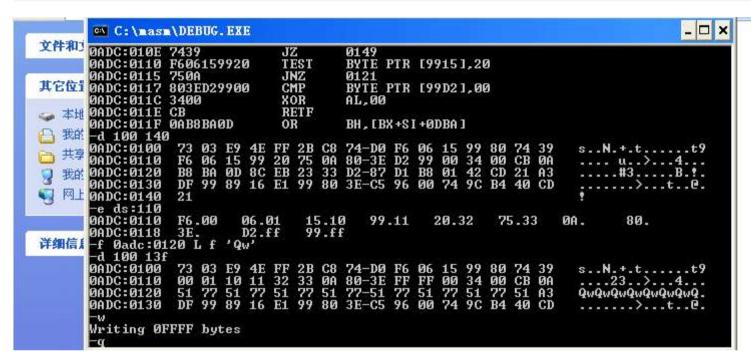
@ADC:@140 21
                                                                                                          BH, [BX+SI+ØDBA]
```

使用 hexdump 查看文件 QWQ:

```
Offset: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 00000000: 73 03 E9 4E FF 2B C8 74 D0 F6 06 15 99 80 74 39 s.in.+Htpv...t9 00000010: F6 06 15 99 20 75 0A 80 3E D2 99 00 34 00 CB 0A v...u..>R..4.K. 00000020: B8 BA 0D 8C EB 23 33 D2 87 D1 B8 01 42 CD 21 A3 8:..k#3R.Q8.BM!# 00000030: DF 99 89 16 E1 99 80 3E C5 96 00 74 9C B4 40 CD __...a..>E..t.4@M 00000040: 21 72 5F C6 06 E3 99 1A C3 E8 FF 0F FE 06 D2 96 !r_F.c..Ch..~.R.
```

(6) 其它命令

```
-e ds:110
00 01 10 11 32 33 ff ff
-f 0adc:0120 L f 'Qw'
-d 100 13f
-w
```



再次使用 hexdump 查看文件 QwQ:

```
      ■ QWQ1
      ■ QWQ1.hexdump
      ×
      >
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
      □
```

5. 思考题

(1) DEBUG 中 T=100 与单独的 T 指令有何不同? 100 的含义是什么? T=100,3 指令中的 3 有何含义? 应如何应用?

T=100 满足 T[=address] 格式,将会从 address (即 CS:0100)单元中的指令开始单步执行; address 缺省值为 CS:IP ,即单独使用 T 将会从 CS:IP 单元中的指令开始单步执行.

100 是指令指针位置.

T=100,3 指令中的 3 表示执行三步.

(2) DEBUG 中 G=100,108 命令的 100,108 的含义是什么? 说明 T 与 G 命令的异同.

在 CS:0108 处设置断点并从 CS:0100 开始执行程序.

T 为单步执行, G 为连续执行.

(3) 说明 D 与 U 命令的区别.

D 直接以十六进制显示文件内容, U 显示反汇编所得指令.

二. 加强型实验

1. 实验目的

- 1. 利用 DEBUG 掌握有关的命令.
- 2. 利用 DEBUG 运行简单的程序段.

2. 实验内容

用命令 A 汇编下列程序段,并将程序写到硬盘上,然后调试程序段,观察程序段能否完成指定功能.若不能,说明原因,并修改程序.

(1) 内存操作数及各种寻址方式使用

```
MOV AX,1000
MOV [1200], AX
MOV BX,1202
MOV BYTE PTR[BX], 20
MOV DL, 39
INC BX
MOV [BX], DL
DEC DL
MOV SI, 3
MOV [BX+SI], DL
MOV [BX+SI], DL
MOV [BX+SI+1], DL
MOV WORD PTR[BX+SI+2], 1234
```

(2) 多字节加法程序

```
MOV DI,1234
MOV CX,8
MOV SI,2234
CLC
Again: MOV AL,[SI]
ADC [DI],AL
INC SI
INC DI
LOOP Again
```

(3) 移位操作

```
MOV AL,06
SHL AL,1
MOV BL,AL
MOV CL,2
SHL AL,CL
ADD AL,BL
```

(4) 数据串移动

用 A 命令 DB 伪指令在 1000H 键入下列字符串: This's a string...

用 a 命令键入下列程序:

```
MOV SI,1000

MOV DI,2000

MOV CX,0FH

LOP:MOV AL,[SI]

MOV [DI],AL

INC SI

INC DI

LOOP LOP

INT 20
```

3. 实验仪器设备

- 硬件环境: AMD Ryzen 9.
- 软件环境: Windows XP Pro SP3 (in VMware Workstation 16 Pro).
- 编辑程序: Visual Studio Code 1.62.3.
- 调试程序: DEBUG.EXE.

4. 实验步骤

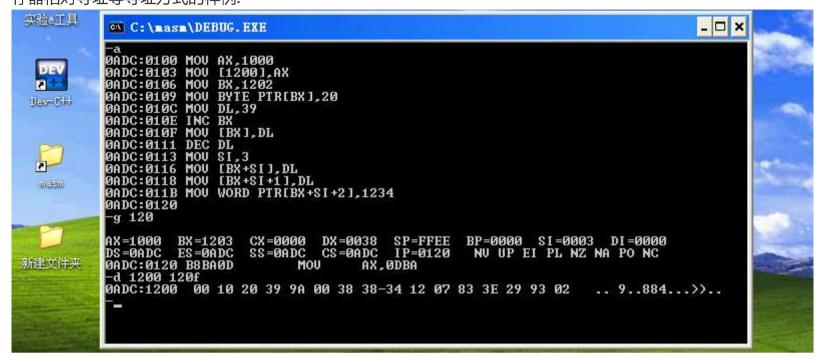
(1) 内存操作数及各种寻址方式使用

原程序需要将中文标点修改为英文标点.

运行和查看结果:

```
-a
# 程序段
-g 120
-d 1200 120f
```

立即寻址、寄存器寻址、直接寻址、寄存器间接寻址、变址寻址、寄存器相对寻址等寻址方式的样例.



(2) 多字节加法程序

修改后程序:

```
MOV DI,1234
MOV CX,8
MOV SI,2234
CLC
MOV AL,[SI]
ADC [DI],AL
INC SI
INC DI
LOOP 010A
```

运行和查看结果:

```
-a
# 程序段
-d 1230 123f
-d 2230 223f
-g 112
-d 1230 123f
-d 2230 223f
```

将 DS:1234 处(小端模式)存储的 64-bit 整数与 DS:2234 处(小端模式)存储的 64-bit 整数相加并存回 DS:1234 处.

```
实验&工具
                                                                                                                                                                                                                                             _ & ×
                              C:\masm\DEBUG.EXE
                          -a
0ADC:0100 MOU DI,1234
0ADC:0103 MOU CX,8
0ADC:0106 MOU SI,2234
0ADC:0109 CLC
0ADC:010A MOU AL,[SI]
0ADC:010C ADC [DI],AL
0ADC:010E INC SI
0ADC:010F INC DI
0ADC:0110 LOOP 010A
0ADC:0112
-d 1230 123f
0ADC:1230 01 BA 78 8A A3 78 8A 0E-07 E9 12 D9 53 E8 08 00
-d 2230 223f
0ADC:2230 25 32 2E 35 30 30 13 49-6E 76 61 6C 69 64 20 64
-g 112
     DIEV
    7
  Bev-C++
    7
     masm
                                                                                                                                                                                                        ..x..x.....S....
                                                                                                                                                                                                       22.500.Invalid d
                            AX=006C BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=22
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0112 NU UP EI PL N
0ADC:0112 159920 ADC AX,2099
-d 1230 123f
0ADC:1230 01 BA 78 8A D3 A8 9D 57-75 5F 74 45 53 E8 08 00
-d 2230 223f
0ADC:2230 25 32 2E 35 30 30 13 49-6E 76 61 6C 69 64 20 64
                                                                              CX=0000 DX=0000 SP=FFEE BP=0000 SI=223C DI=123C SS=0ADC CS=0ADC IP=0112 NU UP EI PL NZ NA PE CY ADC AX,2099
新建文件夹
                                                                                                                                                                                                        ..x....Wu_tES...
                                                                                                                                                                                                       22.500.Invalid d
```

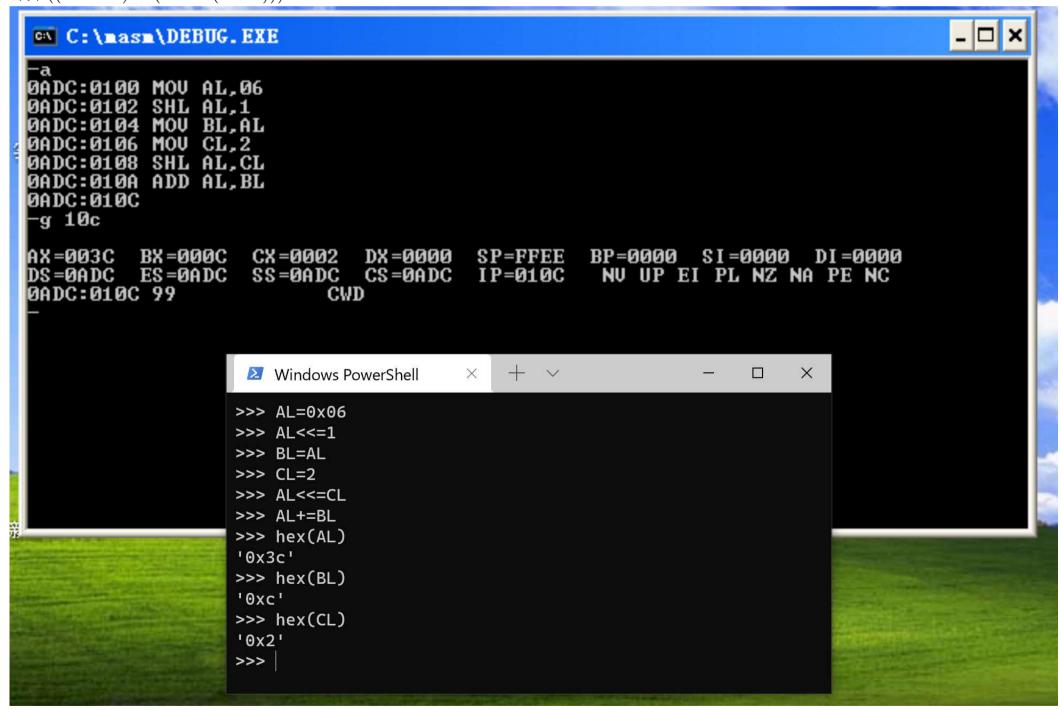
```
>>> a=0xA3788A0E07E912D9
>>> b=0x303013496E76616C
>>> aa=int.from_bytes(a.to_bytes(8,'big'),'little')
>>> bb=int.from_bytes(b.to_bytes(8,'big'),'little')
>>> cc=aa+bb
>>> c=int.from_bytes(cc.to_bytes(9,'little'),'big')
>>> hex(c).upper()
'0XD3A89D57755F744501'
```

(3) 移位操作

原程序不需要修改. 运行和查看结果:

```
-a
# 程序段
-g 10C
```

计算 ((6 << 1) + (6 << (1+2))).



(4) 数据串移动

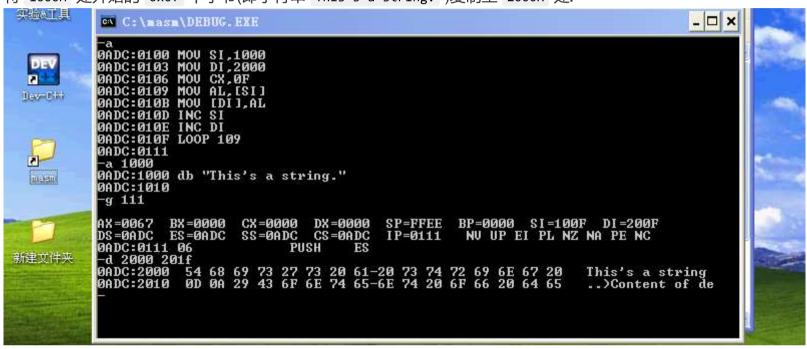
修改后程序:

```
MOV SI,1000
MOV DI,2000
MOV CX,0F
MOV AL,[SI]
MOV [DI],AL
INC SI
INC DI
LOOP 109
```

运行和查看结果:

```
-a
# 程序段
-a 1000
db "This's a string."
-g 111
-d 2000 201f
```

将 1000H 处开始的 0x0f 个字节(即字符串 This's a string.)复制至 2000H 处.



5. 思考题

(1) 假设某内存单元的内容为 ссн , 试问该内容如看作一条指令/无符号数/带符号数,那将分别表示什么?不查看教材中的指令系统代码,如何在 DEBUG 下知道 ссн 表示一条什么指令?

指令: INT 3; 无符号数: 204; 带符号数(8-bit): -52.

```
-e 100
cc
-u 100 100
```



(2) 在指令 MOV [BX],AX 中,操作数 [BX] 的寻址方式是什么? 在 DEBUG 下执行完该指令后,如何查看执行结果?

寄存器间接寻址.

使用 D 命令查看相应地址数据.

(3) 可否随意在 DEBUG 提示符 - 后不带参数发出命令 G ? 什么情况下使用命令 G 时,可不用 = 给出执行的首地址?

否.

CS:IP 正确时.

(2/2)汇编语言上机环境及使用

一. 基础性实验

1. 实验目的

- 1. 掌握汇编语言程序上机过程.
- 2. 掌握汇编语言结构.
- 3. 学习汇编语言的伪操作.
- 4. 了解汇编程序、连接程序、DOS 系统装入和返回功能.
- 5. 掌握用 DEBUG 调试汇编语言程序的方法.

2. 实验内容

编写完整的汇编语言程序: 从变量 DD1 开始存放 3 个无符号字节数据,编制一个找出其中最大者存入 DD2 单元的源程序.

3. 实验仪器设备

- 硬件环境: AMD Ryzen 9.
- 软件环境: Windows XP Pro SP3 (in VMware Workstation 16 Pro).
- 编辑程序: Visual Studio Code 1.62.3.
- 调试程序: DEBUG.EXE.
- 编译程序: MASM.EXE.
- 连接程序: LINK.EXE.

4. 实验步骤

- (1) 建立汇编语言程序源文件
- (2) 汇编文件,形成目标模块

若文件有语法错误,根据错误提示,返回(1),修改源文件;若无错,则形成 .OBJ 目标文件和 .LST 列表文件.

(3)连接目标模块,形成可执行文件

若有错,根据错误提示,返回(1),修改错误; 若无错,则形成 .exe 可执行文件和 .map 映像文件.

(4)执行

DOS下直接键入可执行文件名,即可执行该程序.

若运行结果不对,需调试程序,查找原因:

- 1. 返回步骤(1)查看源文件,查找错误,修改文件;
- 2. 若无法通过查看源文件找出错误,可在 **DEBUG** 下调试 .EXE 文件,找出错误之处,再返回步骤(1)修改源文件.

可使用如下脚本快速编译连接运行:

```
@REM code by hyper
@REM edit by userElaina
set /p file="Enter file name: "
set CC=masm.exe
set LD=link.exe
echo _compile_
%CC% %file%.asm
echo _link_
%LD% %file%.obj
echo _execute_
%file%.exe
pause
```

5. 思考题

(1) 汇编语言程序中语句 END 后的标号作用是什么?

程序结束.

(2) 一个段的大小为多少? 一定是 64KB 吗? 如何在 DEBUG 下查看程序在内存的存放情况?

一个段的大小通常为 64KB.

不一定.

使用 D 命令查看.

二. 加强型实验

1. 实验目的

- 1. 进一步熟悉 DEBUG 的使用方法.
- 2. 掌握汇编语言程序的编辑/汇编/连接以及调试过程.

2. 实验内容

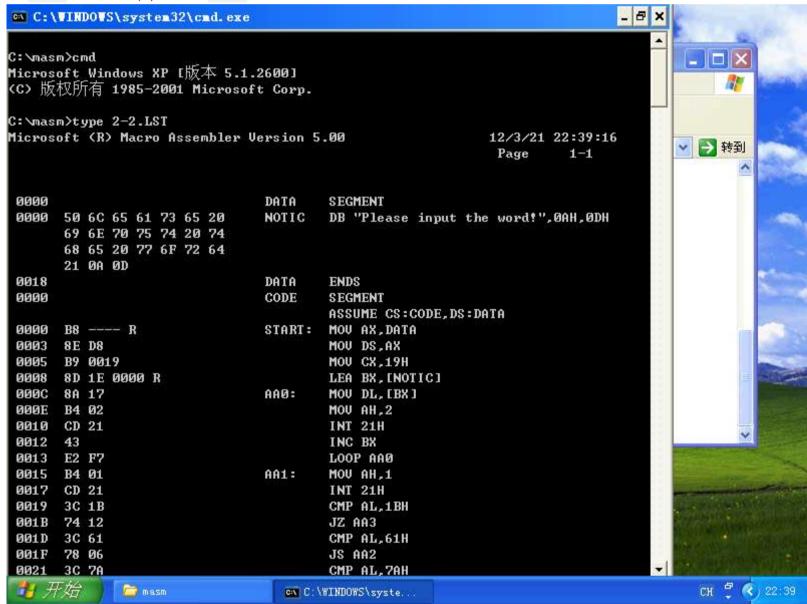
试编写一个汇编语言程序,要求对键盘输入的小写字母用大写字母显示出来.

3. 实验仪器设备

- 硬件环境: AMD Ryzen 9.
- 软件环境: Windows XP Pro SP3 (in VMware Workstation 16 Pro).
- 编辑程序: Visual Studio Code 1.62.3.
- 调试程序: DEBUG.EXE.
- 编译程序: MASM.EXE.
- 连接程序: LINK.EXE.

4. 实验步骤

- 1. 编译和连接.
- 2. 用 TYPE 命令显示(1)产生的 .LST 文件.

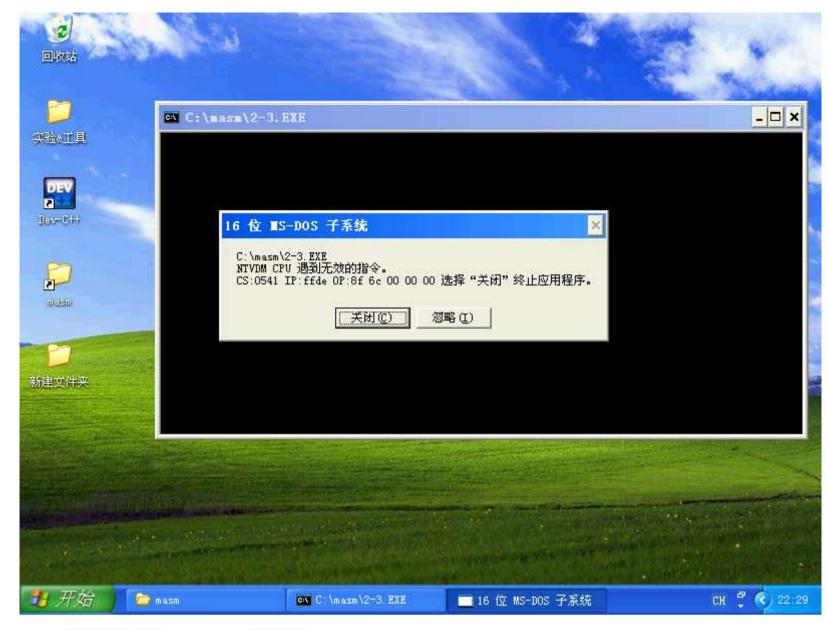


- 3. 运行.
- 4. 键入大量字符.
- 5. 按 ESC 退出.

5. 思考题

(1) 若在源程序中把 INT 21H 的 H 省去,会产生什么现象?

正常编译连接,运行后无反应,一段时间后弹出错误.



(2) 把 INT 21H 4CH 号功能改为 INT 20H, 行不行?

不行.

正常编译连接,运行后显示功能正常,但按 ESC 无法回到DOS,同时光标在界面内快速跳跃闪动,CPU消耗大幅度增加. 若直接双击运行,则按 ESC 时直接退出.