



山东大学

信息科学与工程学院

2019 – 2020 学年第二学期

实 验 报 告

课程名称: 微处理器原理与应用

实验名称: 实验 1.2

专 业 班 级 通信工程 二班

学 生 学 号 201800121050

学 生 姓 名 孟麟芝

实 验 时 间 2020 年 2 月 15 日

实验报告

【实验目的】

1. 掌握 DEBUG 的基本命令及其功能，掌握 win10 使用 DEBUG 功能
2. 了解使用机器指令编程，掌握汇编指令编程

【实验要求】

1. 掌握一些基本汇编指令的运用
2. 掌握 DEBUG 的 R,D,E,U,T,A 等指令的运用

【实验具体内容】

1. 预备知识：

(1) 本次实验所需汇编指令

mov A,B 即将 B 的值赋给 A

add ax,bx 即将 ax, bx 中的值相加的结果赋给 ax

sub ax,bx 即将 bx 减去 ax 后的结果赋给 ax

jmp 无条件跳转，用以改变 CS, IP

(2) 何为 DEBUG

Debug 是 DOS 和 Windows 都提供的实模式（8086 方式）程序的调试工具。使用它可以查看 CPU 各种寄存器中的内存、内存情况和在机器码级别跟踪程序的运行

(3) DEBUG 中的几个指令

-r 查看、改变 CPU 寄存器的内容

-d 查看内存中的内容

-e 改写内存中的内容

-u 将内存中的机器指令翻译成汇编指令

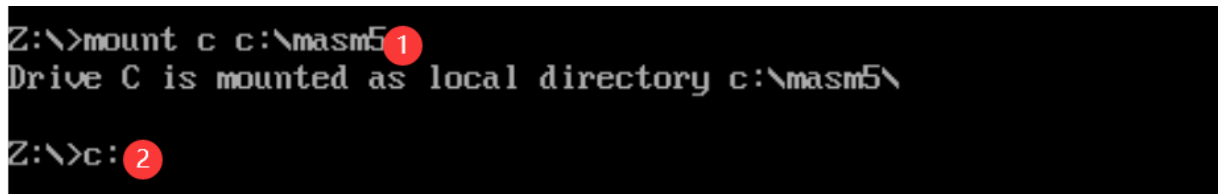
-t 执行一条机器指令

-a 以汇编形式在内存中写入一条机器指令

2.

【调试 DEBUG】

(1) 在 winxp 环境下的 CMD 提供的 DOS 环境可直接进入 DEBUG，在 win10 下需要由 DOSBox 实现



```
Z:\>mount c c:\masm5
Drive C is mounted as local directory c:\masm5\
Z:\>c:
```

① 将本地的 c:\masm5 挂载为虚拟环境中的 C 盘（可挂载多个盘，只要分别命名即

可)

② 在虚拟环境中选择 C 盘符

(2) 打开 debug 程序

```
C:\>debug
-r
AX=0000 BX=0000 CX=0000 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 0000          ADD     [BX+SI],AL          DS:0000=CD
```

可见 DEBUG 已可正常使用

【第一个实验】

实验要求：使用 DEBUG 将下面的程序写入内存，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
05 16 14	add ax,1416H
bb 00 20	mov bx,2000H
01 d8	add ax,bx
89 c3	mov bx,ax
01 d8	add ax,bx
b8 1a 00	mov ax,001AH
bb 26 00	mov bx,0026H
00 d8	add al,bl
00 dc	add ah,bl
00 c7	add bh,al
b4 00	mov ah,0
00 d8	add al,bl
04 9c	add al,9CH

(1) 使用-r 指令查看当前 CS:IP 的位置，不妨在该位置开始写入指令

```
C:\>debug
-r
AX=0000 BX=0000 CX=0000 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 0000          ADD     [BX+SI],AL          DS:0000=CD
-a 073f:0100
```

(2) 使用-a 指令写入内存

```

-a 073f:0100
073F:0100 mov ax,4e20
073F:0103 add ax,1416
073F:0106 mov bx,2000
      ^ Error
073F:0106 mov bx,2000
073F:0109 add ax,bx
073F:010B mov bx,ax
073F:010D add ax,bx
073F:010F mov ax,001a
073F:0112 mov bx,0026
073F:0115 add al,bl
073F:0117 add ah,bl
073F:0119 add bh,al
073F:011B mov ah,0
073F:011D add al,bl
073F:011F add al,9c
    
```

多出一空格

再尝试使用 e 命令写入（已为正确结果，直接按空格跳过更改）

```

-e 073f:0100
073F:0100 B8.    20.    4E.    05.    16.    14.    BB.    00.
073F:0108 20.    01.    D8.    89.    C3.    01.    D8.    B8.
073F:0110 1A.    00.    BB.    26.    00.    00.    D8.    00.
073F:0118 DC.    00.    C7.    B4.    00.    00.    D8.    04.
073F:0120 9C. _
    
```

（3）使用-u 命令查看输入的指令

```

-u 073f:0100
073F:0100 B8204E      MOV     AX,4E20
073F:0103 051614      ADD     AX,1416
073F:0106 BB0020      MOV     BX,2000
073F:0109 01D8        ADD     AX,BX
073F:010B 89C3        MOV     BX,AX
073F:010D 01D8        ADD     AX,BX
073F:010F B81A00      MOV     AX,001A
073F:0112 BB2600      MOV     BX,0026
073F:0115 00D8        ADD     AL,BL
073F:0117 00DC        ADD     AH,BL
073F:0119 00C7        ADD     BH,AL
073F:011B B400        MOV     AH,00
073F:011D 00D8        ADD     AL,BL
073F:011F 049C        ADD     AL,9C
    
```

注意，如果不指定 Max IP，则最多翻译 20H 个字节的指令

（4）使用-t 指令逐条执行

MOV AX,4E20

```

AX=0000 BX=0000 CX=0000 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 B8204E          MOV     AX,4E20
-t
AX=4E20 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0103  NU UP EI PL NZ NA PO NC
073F:0103 051614          ADD     AX,1416
    
```

ADD AX,1416

```

AX=4E20 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0103  NU UP EI PL NZ NA PO NC
073F:0103 051614          ADD     AX,1416
-t
AX=6236 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106  NU UP EI PL NZ NA PE NC
073F:0106 BB0020          MOV     BX,2000
    
```

MOV BX,2000

```

AX=6236 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106  NU UP EI PL NZ NA PE NC
073F:0106 BB0020          MOV     BX,2000
-t
AX=6236 BX=2000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0109  NU UP EI PL NZ NA PE NC
073F:0109 01D8          ADD     AX,BX
    
```

ADD AX,BX

```

AX=6236 BX=2000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0109  NU UP EI PL NZ NA PE NC
073F:0109 01D8          ADD     AX,BX
-t
AX=8236 BX=2000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010B  OV UP EI NG NZ NA PE NC
073F:010B 89C3          MOV     BX,AX
    
```

MOV BX,AX

```

AX=8236 BX=2000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010B  OV UP EI NG NZ NA PE NC
073F:010B 89C3          MOV     BX,AX
-t
AX=8236 BX=8236 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010D  OV UP EI NG NZ NA PE NC
073F:010D 01D8          ADD     AX,BX
    
```

ADD AX,BX

```

AX=8236 BX=8236 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010D OV UP EI NG NZ NA PE NC
073F:010D 01D8 ADD AX,BX
-t
AX=046C BX=8236 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010F OV UP EI PL NZ NA PE CY
073F:010F B81A00 MOV AX,001A
    
```

MOV AX,001A

```

AX=046C BX=8236 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010F OV UP EI PL NZ NA PE CY
073F:010F B81A00 MOV AX,001A
-t
AX=001A BX=8236 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0112 OV UP EI PL NZ NA PE CY
073F:0112 BB2600 MOV BX,0026
    
```

MOV BX,0026

```

AX=001A BX=8236 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0112 OV UP EI PL NZ NA PE CY
073F:0112 BB2600 MOV BX,0026
-t
AX=001A BX=0026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0115 OV UP EI PL NZ NA PE CY
073F:0115 00D8 ADD AL,BL
    
```

ADD AL,BL

```

AX=001A BX=0026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0115 OV UP EI PL NZ NA PE CY
073F:0115 00D8 ADD AL,BL
-t
AX=0040 BX=0026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0117 NV UP EI PL NZ AC PO NC
073F:0117 00DC ADD AH,BL
    
```

ADD AH,BL

```

AX=0040 BX=0026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0117 NV UP EI PL NZ AC PO NC
073F:0117 00DC ADD AH,BL
-t
AX=2640 BX=0026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0119 NV UP EI PL NZ NA PO NC
073F:0119 00C7 ADD BH,AL
    
```

ADD BH,AL

```

AX=2640 BX=0026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0119  NU UP EI PL NZ NA PO NC
073F:0119 00C7          ADD     BH,AL
-t
AX=2640 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011B  NU UP EI PL NZ NA PO NC
073F:011B B400          MOV     AH,00
    
```

MOV AH,0

```

AX=2640 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011B  NU UP EI PL NZ NA PO NC
073F:011B B400          MOV     AH,00
-t
AX=0040 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011D  NU UP EI PL NZ NA PO NC
073F:011D 00D8          ADD     AL,BL
    
```

ADD AL,BL

```

AX=0040 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011D  NU UP EI PL NZ NA PO NC
073F:011D 00D8          ADD     AL,BL
-t
AX=0066 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011F  NU UP EI PL NZ NA PE NC
073F:011F 049C          ADD     AL,9C
    
```

ADD AL,9C

```

AX=0066 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011F  NU UP EI PL NZ NA PE NC
073F:011F 049C          ADD     AL,9C
-t
AX=0002 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0121  NU UP EI PL NZ AC PO CY
073F:0121 0000          ADD     [BX+SI],AL      DS:4026=00
    
```

【第二个实验】

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

mov ax,1 （从 2000:0 开始的内存单元）

add ax,ax

jmp 2000:0003 （观察跳到什么地方了？）

（1）将三条指令写入从 2000: 0 开始的内存单元中

```

-a 2000:0
2000:0000 mov ax,1
2000:0003 add ax,ax
2000:0005 jmp 2000:0003
2000:0007
    
```

(2) 逐条执行，直到 ax 中计算得出 2 的 8 次方

MOV AX,0001

```

AX=4E20 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=2000 IP=0000  NU UP EI PL NZ AC PO CY
2000:0000 B80100      MOV     AX,0001
-t
AX=0001 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=2000 IP=0003  NU UP EI PL NZ AC PO CY
2000:0003 01C0      ADD     AX,AX
    
```

ADD AX,AX

```

AX=0001 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=2000 IP=0003  NU UP EI PL NZ AC PO CY
2000:0003 01C0      ADD     AX,AX
-t
AX=0002 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=2000 IP=0005  NU UP EI PL NZ NA PO NC
2000:0005 EBFC      JMP     0003
    
```

JMP 0003

```

AX=0002 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=2000 IP=0005  NU UP EI PL NZ NA PO NC
2000:0005 EBFC      JMP     0003
-t
AX=0002 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=2000 IP=0003  NU UP EI PL NZ NA PO NC
2000:0003 01C0      ADD     AX,AX
    
```

到此可见，程序发生循环，又跳回了 ax+ax 的步骤，实现循环乘 2，经过循环，得到 ax 中为 0100H,即得到所需结果

```

AX=0100 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=2000 IP=0005  NU UP EI PL NZ NA PE NC
2000:0005 EBFC      JMP     0003
    
```

【第三个实验】

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。（内存 ffff:0005~ffff:000C(共 8 个字节单元中)处）


```
-d ffff:0
FFFF:0000 EA C0 12 00 F0 30 31 2F-30 31 2F 39 32 00 FC 55 .....01/01/92..U
FFFF:0010 60 10 00 F0 BB 13 A3 01-08 00 70 00 B1 13 A3 01 .....p.....
FFFF:0020 08 00 70 00 60 10 00 F0-60 10 00 F0 60 10 00 F0 ..p.....
FFFF:0030 A5 FE 00 F0 87 E9 00 F0-55 FF 00 F0 60 10 00 F0 .....U.....
FFFF:0040 60 10 00 F0 60 10 00 F0-80 10 00 F0 60 10 00 F0 .....
FFFF:0050 00 13 00 F0 00 11 00 F0-20 11 00 F0 40 11 00 F0 .....e.....
FFFF:0060 A0 11 00 F0 C0 11 00 F0-E0 11 00 F0 20 12 00 F0 .....
FFFF:0070 C0 12 00 F0 C0 12 00 F0-40 12 00 F0 60 10 00 F0 .....e.....
```

可见在该 DOSBOX 环境下的主板生产日期为 92 年 1 月 1 日，要注意，这是虚拟环境中虚拟的生产日期，并不是真实的日期

尝试修改为 00 年 12 月 12 日

```
-e ffff:0005
FFFF:0005 30.31 31.32 2F.
FFFF:0008 30.31 31.32 2F. 39.30 32.30

-d ffff:0000
FFFF:0000 EA C0 12 00 F0 30 31 2F-30 31 2F 39 32 00 FC 55 .....01/01/92..U
FFFF:0010 60 10 00 F0 BB 13 A3 01-08 00 70 00 B1 13 A3 01 .....p.....
FFFF:0020 08 00 70 00 60 10 00 F0-60 10 00 F0 60 10 00 F0 ..p.....
FFFF:0030 A5 FE 00 F0 87 E9 00 F0-55 FF 00 F0 60 10 00 F0 .....U.....
FFFF:0040 60 10 00 F0 60 10 00 F0-80 10 00 F0 60 10 00 F0 .....
FFFF:0050 00 13 00 F0 00 11 00 F0-20 11 00 F0 40 11 00 F0 .....e.....
FFFF:0060 A0 11 00 F0 C0 11 00 F0-E0 11 00 F0 20 12 00 F0 .....
FFFF:0070 C0 12 00 F0 C0 12 00 F0-40 12 00 F0 60 10 00 F0 .....e.....
```

可见修改并没有起到作用，问题在于，向地址 C0000~FFFFFF 的内存单元写入数据的操作是无效的，因为这等于改写只读存储器中的内容

【第四个实验】

使用 Debug，将下面的程序段写入内存，逐条执行，根据指令执行后的实际运行情况填空。（逐条执行，每条指令执行结果截图）

```
mov ax,ffff
mov ds,ax
mov ax,2200
mov ss,ax
mov sp,0100
mov ax,[0]           ;ax=_____
add ax,[2]           ;ax=_____
mov bx,[4]           ;bx=_____

add bx,[6]           ;bx=_____
push ax              ;sp=_____ ;修改的内存单元的地址是_____
内容为_____
push bx              ;sp=_____ ;修改的内存单元的地址是_____
内容为_____
pop ax               ;sp=_____ ;ax=_____
pop bx               ;sp=_____ ;bx=_____
push [4]             ;sp=_____ ;修改的内存单元的地址是_____
内容为_____
push [6]             ;sp=_____ ;修改的内存单元的地址是_____
内容为_____
```

(1) 将指令写入内存

```

-a 2000:0000
2000:0000 mov ax,ffff
2000:0003 mov ds,ax
2000:0005 mov ax,2200
2000:0008 mov ss,ax
2000:000A mov sp,0100
2000:000D mov ax,[0]
2000:0010 add ax,[2]
2000:0014 mov bx,[4]
2000:0018 add bx,[6]
2000:001C push ax
2000:001D push bx
2000:001E pop ax
2000:001F pop bx
2000:0020 push [4]
2000:0024 push [6]
    
```

(2) 逐条执行指令

MOV AX,FFFF

```

AX=0100 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=2000 IP=0000  NU UP EI PL NZ NA PE NC
2000:0000 B8FFFF      MOV     AX,FFFF
-t
AX=FFFF BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=2000 IP=0003  NU UP EI PL NZ NA PE NC
2000:0003 8ED8      MOV     DS,AX
    
```

MOV DS,AX

```

AX=FFFF BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=2000 IP=0003  NU UP EI PL NZ NA PE NC
2000:0003 8ED8      MOV     DS,AX
-t
AX=FFFF BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=073F CS=2000 IP=0005  NU UP EI PL NZ NA PE NC
2000:0005 B80022      MOV     AX,2200
    
```

MOV AX,2200

```

AX=FFFF BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=073F CS=2000 IP=0005  NU UP EI PL NZ NA PE NC
2000:0005 B80022      MOV     AX,2200
-t
AX=2200 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=073F CS=2000 IP=0008  NU UP EI PL NZ NA PE NC
2000:0008 8ED0      MOV     SS,AX
    
```

MOV SS,AX

```

AX=2200 BX=4026 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=073F CS=2000 IP=0008 NU UP EI PL NZ NA PE NC
2000:0008 8ED0          MOV     SS,AX
-t
AX=2200 BX=4026 CX=0000 DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=000D NU UP EI PL NZ NA PE NC
2000:000D A10000       MOV     AX,[0000]          DS:0000=C0EA
    
```

MOV AX,[0]

```

AX=2200 BX=4026 CX=0000 DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=000D NU UP EI PL NZ NA PE NC
2000:000D A10000       MOV     AX,[0000]          DS:0000=C0EA
-t
AX=C0EA BX=4026 CX=0000 DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0010 NU UP EI PL NZ NA PE NC
2000:0010 03060200     ADD     AX,[0002]          DS:0002=0012
    
```

ADD AX,[2]

```

AX=C0EA BX=4026 CX=0000 DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0010 NU UP EI PL NZ NA PE NC
2000:0010 03060200     ADD     AX,[0002]          DS:0002=0012
-t
AX=C0FC BX=4026 CX=0000 DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0014 NU UP EI NG NZ NA PE NC
2000:0014 8B1E0400     MOV     BX,[0004]          DS:0004=30F0
    
```

MOV BX,[4]

```

AX=C0FC BX=4026 CX=0000 DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0014 NU UP EI NG NZ NA PE NC
2000:0014 8B1E0400     MOV     BX,[0004]          DS:0004=30F0
-t
AX=C0FC BX=30F0 CX=0000 DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0018 NU UP EI NG NZ NA PE NC
2000:0018 031E0600     ADD     BX,[0006]          DS:0006=2F31
    
```

ADD BX,[6]

```

AX=C0FC BX=30F0 CX=0000 DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0018 NU UP EI NG NZ NA PE NC
2000:0018 031E0600     ADD     BX,[0006]          DS:0006=2F31
-t
AX=C0FC BX=6021 CX=0000 DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=001C NU UP EI PL NZ NA PE NC
2000:001C 50          PUSH    AX
    
```

PUSH AX

```

AX=C0FC BX=6021 CX=0000 DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=001C NU UP EI PL NZ NA PE NC
2000:001C 50          PUSH    AX
-t

AX=C0FC BX=6021 CX=0000 DX=0000 SP=00FE BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=001D NU UP EI PL NZ NA PE NC
2000:001D 53          PUSH    BX
    
```

PUSH BX

```

AX=C0FC BX=6021 CX=0000 DX=0000 SP=00FE BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=001D NU UP EI PL NZ NA PE NC
2000:001D 53          PUSH    BX
-t

AX=C0FC BX=6021 CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=001E NU UP EI PL NZ NA PE NC
2000:001E 58          POP     AX
    
```

POP AX

```

AX=C0FC BX=6021 CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=001E NU UP EI PL NZ NA PE NC
2000:001E 58          POP     AX
-t

AX=6021 BX=6021 CX=0000 DX=0000 SP=00FE BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=001F NU UP EI PL NZ NA PE NC
2000:001F 5B          POP     BX
    
```

POP BX

```

AX=6021 BX=6021 CX=0000 DX=0000 SP=00FE BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=001F NU UP EI PL NZ NA PE NC
2000:001F 5B          POP     BX
-t

AX=6021 BX=C0FC CX=0000 DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0020 NU UP EI PL NZ NA PE NC
2000:0020 FF360400    PUSH    [0004]          DS:0004=30F0
    
```

PUSH [4]

```

AX=6021 BX=C0FC CX=0000 DX=0000 SP=0100 BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0020 NU UP EI PL NZ NA PE NC
2000:0020 FF360400    PUSH    [0004]          DS:0004=30F0
-t

AX=6021 BX=C0FC CX=0000 DX=0000 SP=00FE BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0024 NU UP EI PL NZ NA PE NC
2000:0024 FF360600    PUSH    [0006]          DS:0006=2F31
    
```

PUSH [6]

```

AX=6021 BX=C0FC CX=0000 DX=0000 SP=00FE BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0024  NU UP EI PL NZ NA PE NC
2000:0024 FF360600      PUSH      [0006]      DS:0006=2F31
-t

AX=6021 BX=C0FC CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0028  NU UP EI PL NZ NA PE NC
2000:0028 0000      ADD      [BX+SI],AL      DS:C0FC=00

-d 2200:00f0
2200:00f0  00 00 21 60 00 00 28 00-00 20 a3 01 31 2f f0 30  ..!`..(.. ..1/.0
2200:0100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00  .....
2200:0110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00  .....
2200:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00  .....
2200:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00  .....
2200:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00  .....
2200:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00  .....
2200:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00  .....
    
```

结束最后一步操作后，可见栈内存放着最后放入的两个数据，DS:0006 和 DS:0004 的数据：2F31 和 30F0

(3) 根据实际运行情况填空

```

mov ax,ffff
mov ds,ax
mov ax,2200
mov ss,ax
mov sp,0100
mov ax,[0]      ;ax= C0EA
add ax,[2]      ;ax= C0FC
mov bx,[4]      ;bx= 30F0

add bx,[6]      ;bx= 6021
push ax         ;sp= 00FE ;修改的内存单元的地址是 2200:00FE
内容为 C0FC
push bx        ;sp= 00FC ;修改的内存单元的地址是 2200:00FC
内容为 6021
pop ax         ;sp= 00FE ;ax= 6021
pop bx        ;sp= 0100 ;bx= C0FC
push [4]       ;sp= 00FE ;修改的内存单元的地址是 2200:00FE
内容为 30F0
push [6]       ;sp= 00FC ;修改的内存单元的地址是 2200:00FC
内容为 2F31
    
```

【第五个实验】

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 中相关寄存器中内容的变化。（逐条执行，每条指令执行结果截图）如果有问题请说明原因

汇编指令

mov ax,1000H

mov ds,ax

mov ds,[0]

add ds,ax

(1) 将指令写入内存后逐条执行

```
-r
AX=6021 BX=C0FC CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0028 NU UP EI PL NZ NA PE NC
2000:0028 0000 ADD [BX+SI],AL DS:C0FC=00
-a 2000:0028
2000:0028 mov ax,1000
2000:002B mov ds,ax
2000:002D mov ds,[0]
2000:0031 add ds,ax
                ^ Error
```

要注意，段寄存器不可以参与算术运算

(2) 将 add ds, ax 指令替换为如下三条指令后逐条执行

mov bx,ds

add ax,bx

mov ds,ax

(3) 逐条执行

MOV AX,1000

```
AX=6021 BX=C0FC CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=0028 NU UP EI PL NZ NA PE NC
2000:0028 B80010 MOV AX,1000
-t
AX=1000 BX=C0FC CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=002B NU UP EI PL NZ NA PE NC
2000:002B 8ED8 MOV DS,AX
```

MOV DS,AX

```
AX=1000 BX=C0FC CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=FFFF ES=073F SS=2200 CS=2000 IP=002B NU UP EI PL NZ NA PE NC
2000:002B 8ED8 MOV DS,AX
-t
AX=1000 BX=C0FC CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=1000 ES=073F SS=2200 CS=2000 IP=002D NU UP EI PL NZ NA PE NC
2000:002D 8E1E0000 MOV DS,[0000] DS:0000=0000
```


MOV DS,[0]

```

AX=1000 BX=C0FC CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=1000 ES=073F SS=2200 CS=2000 IP=002D  NU UP EI PL NZ NA PE NC
2000:002D 8E1E0000      MOV     DS,[0000]      DS:0000=0000
-t
AX=1000 BX=C0FC CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=0000 ES=073F SS=2200 CS=2000 IP=0031  NU UP EI PL NZ NA PE NC
2000:0031 8CDB      MOV     BX,DS
    
```

MOV BX,DS

```

AX=1000 BX=C0FC CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=0000 ES=073F SS=2200 CS=2000 IP=0031  NU UP EI PL NZ NA PE NC
2000:0031 8CDB      MOV     BX,DS
-t
AX=1000 BX=0000 CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=0000 ES=073F SS=2200 CS=2000 IP=0033  NU UP EI PL NZ NA PE NC
2000:0033 01DB      ADD     AX,BX
    
```

ADD AX,BX

```

AX=1000 BX=0000 CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=0000 ES=073F SS=2200 CS=2000 IP=0033  NU UP EI PL NZ NA PE NC
2000:0033 01DB      ADD     AX,BX
-t
AX=1000 BX=0000 CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=0000 ES=073F SS=2200 CS=2000 IP=0035  NU UP EI PL NZ NA PE NC
2000:0035 8ED8      MOV     DS,AX
    
```

MOV DS,AX

```

AX=1000 BX=0000 CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=0000 ES=073F SS=2200 CS=2000 IP=0035  NU UP EI PL NZ NA PE NC
2000:0035 8ED8      MOV     DS,AX
-t
AX=1000 BX=0000 CX=0000 DX=0000 SP=00FC BP=0000 SI=0000 DI=0000
DS=1000 ES=073F SS=2200 CS=2000 IP=0037  NU UP EI PL NZ NA PE NC
2000:0037 0000      ADD     [BX+SI],AL      DS:0000=00
    
```

实现了原功能

【第六个实验】

仔细观察下图中的实验过程，然后分析：为什么 2000:0~2000:f 中的内容会发生改变？

```

C:\>debug
-a
0B39:0100 mov ax,2000
0B39:0103 mov ss,ax
0B39:0105 mov sp,10
0B39:0108 mov ax,3123
0B39:010B push ax
0B39:010C mov ax,3366
0B39:010F push ax
0B39:0110
-
-e 2000:0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-
-d 2000:0 f
2000:0000 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B39 ES=0B39 SS=0B39 CS=0B39 IP=0100 NU UP EI PL NZ NA PO NC
0B39:0100 B80020 MOV AX,2000
-t
AX=2000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B39 ES=0B39 SS=0B39 CS=0B39 IP=0103 NU UP EI PL NZ NA PO NC
0B39:0103 8ED0 MOV SS,AX
-t
AX=2000 BX=0000 CX=0000 DX=0000 SP=0010 BP=0000 SI=0000 DI=0000
DS=0B39 ES=0B39 SS=2000 CS=0B39 IP=0108 NU UP EI PL NZ NA PO NC
0B39:0108 B82331 MOV AX,3123
-d 2000:0 f
2000:0000 00 00 00 00 00 00 00 20-00 00 08 01 39 0B 9D 05 .....
-
    
```

在执行 `mov ss,ax` 时，它的下一条指令 `mov sp,10` 也紧跟着被执行。另外，`-t` 指令将会引发单步中断，CPU 执行 1 号中断程序，将标志寄存器入栈，TF、IF 设置为 0，CS、IP 等段寄存器入栈，再对 CS、IP 做特定修改。可见图中右边 6 个字节从右至左分别为标志寄存器的值，CS，IP

【实验心得】

1. 将本地盘挂载为虚拟盘时可挂载多个，类似日常使用电脑的硬盘分区
2. 写入指令时先要注意 CS:IP 的位置，再进行写入
3. 从地址 0~9FFFF 的内存单元中读取数据，实际上就是在读取主随机存储器中的数据，向地址 A0000~BFFFF 的内存单元中写数据，就是向显存中写数据，这些数据会被显示卡输出到显示器上。向地址 C0000~FFFFF 的内存单元写入数据的操作是无效的，因为这等于改写只读存储器中的内容
4. 在执行对段寄存器 SS 的修改时，它的下一条指令也紧跟着被执行
5. `-t` 指令将会引发单步中断，CPU 执行 1 号中断程序，将标志寄存器入栈，TF、IF 设置为 0，CS、IP 等段寄存器入栈，再对 CS、IP 做特定修改
6. `-u` 指令亦可像 `-d` 指令一样指定 IP 的范围，如果不指定 Max IP，则最多翻译 20H 个字节的指令
7. `-e` 指令逐个修改内存单元中的数据时，若不想修改，直接按空格即可

8. 段寄存器不能用于算术运算的指令（`add ax,ds` 与 `add ds,ax` 均不可）
9. 栈指针 SP 始终指向栈顶数据，而当栈为空时，指向栈底的高一位内存单元
10. POP 指令和 PUSH 指令本质上都是两次操作，push 为先 SP-2 再送入，pop 为先送出再 SP+2
11. 当 add 指令中发生溢出时，将会略去高位，且，例如 `add al, bl` 溢出时，不会因为 AL 溢出而将溢出的 1 放入 AX 的第三位
12. 汇编指令中不分大小写，写指令时不需要带“H”

【课上问题】

1. 南北桥芯片：一块主板上，以 CPU 插槽为北，北桥芯片是最靠近 CPU 插槽的芯片，负责与 CPU 联系并控制内存。南桥芯片是离 CPU 插槽较远的芯片，负责 I/O 总线之间的通讯（如通过 USB，SATA 等与外存储设备通信）。
2. 主频：即 CPU 的时钟频率，CPU 每个时钟周期执行一条指令。时钟频率的 SI 单位是 Hz，表示 1s 内能执行的指令数，很大程度反映着 CPU 的快慢
3. 外频：主板上具有一个统一的时钟信号发生器，控制着 CPU 与主板上其他单元（如内存）的通信，这一时钟频率为外频，随着 CPU 主频的提升，外频可能低于主频，传输数据的速度就会遇到“瓶颈”
4. 倍频：在 CPU 中，倍频=主频/外频。
5. Fsb：前端总线，不能与外频混淆。它的速度指的是 CPU 和北桥芯片之间数据传输的速度
6. 32 位系统对内存的支持：32 位系统能寻址的内存单元最多为 2^{32} 个，即最多能支持 4GB 的内存
7. Debug 程序与我们所用的 DOS 环境下的编译器，对指令的支持不尽相同，如对 `mov ax,[2]`，编译器将其认为是 2H 赋给 ax，debug 将其认为是（DS: 0002）赋给 ax。对 16 位数，debug 中不可加 H，在 IDE 中则必须加 H，不然编译结果会出错