

華中科技大學

课程实验报告

课程名称：汇编语言

专业班级: _____

学 号: _____

姓 名: _____

指导教师: _____

报告日期: 2019 年 11 月 01 日

网络空间安全学院

实验二 寻址方式以及结构化数据的访问

1.1 实验目的与内容

1.1.1 实验目的

- 1、熟悉编写和调试具有多个段的应用程序。
- 2、熟悉汇编中一般循环程序的设计思路。
- 3、掌握不同的寻址方式，掌握更灵活定位内存地址的方法。
- 4、灵活运用寻址方式进行结构化数据的访问。

1.1.2 实验内容

重点参考教材实验 4-7 以及实验二、三 ppt。

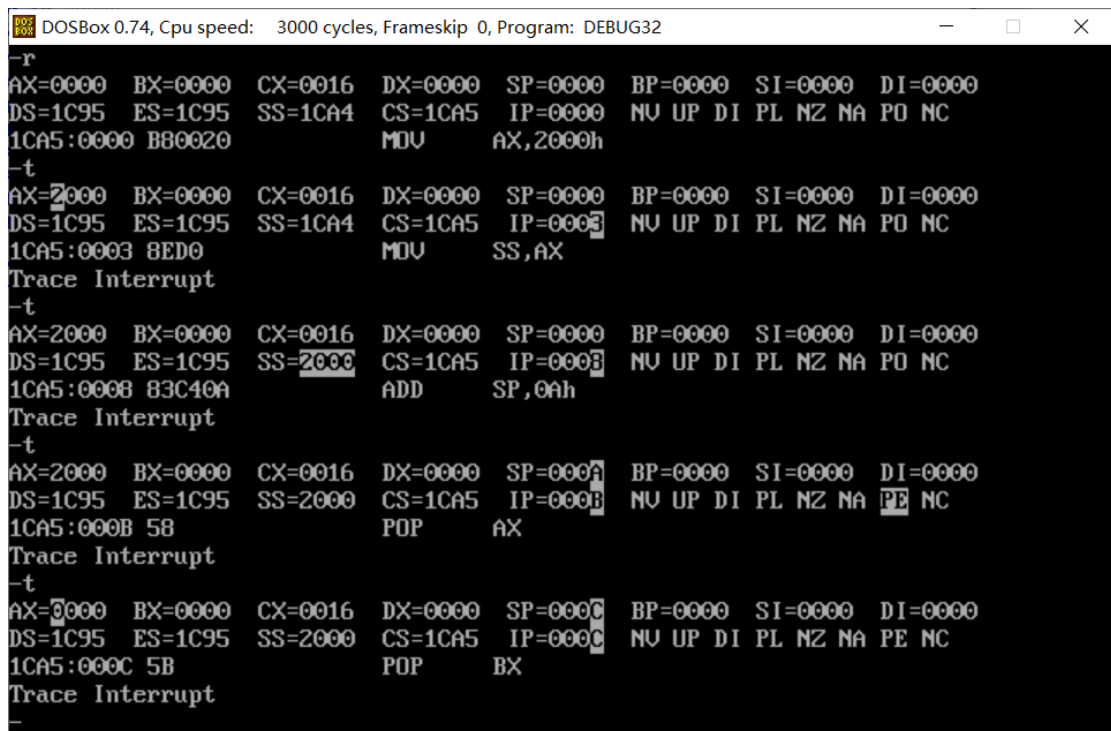
1.1.3 实验工具

Dosbox0.74。

1.2 实验过程

1.2.1 任务一：王爽版 实验 3

(1) 过程截图：



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG32
-r
AX=0000 BX=0000 CX=0016 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=1C95 ES=1C95 SS=1CA4 CS=1CA5 IP=0000 NU UP DI PL NZ NA PO NC
1CA5:0000 B80020 MOV AX,2000h
-t
AX=2000 BX=0000 CX=0016 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=1C95 ES=1C95 SS=1CA4 CS=1CA5 IP=0003 NU UP DI PL NZ NA PO NC
1CA5:0003 8ED0 MOV SS,AX
Trace Interrupt
-t
AX=2000 BX=0000 CX=0016 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=1C95 ES=1C95 SS=2000 CS=1CA5 IP=0003 NU UP DI PL NZ NA PO NC
1CA5:0008 83C40A ADD SP,0Ah
Trace Interrupt
-t
AX=2000 BX=0000 CX=0016 DX=0000 SP=000A BP=0000 SI=0000 DI=0000
DS=1C95 ES=1C95 SS=2000 CS=1CA5 IP=000B NU UP DI PL NZ NA PE NC
1CA5:000B 5B POP AX
Trace Interrupt
-t
AX=0000 BX=0000 CX=0016 DX=0000 SP=000C BP=0000 SI=0000 DI=0000
DS=1C95 ES=1C95 SS=2000 CS=1CA5 IP=000C NU UP DI PL NZ NA PE NC
1CA5:000C 5B POP BX
Trace Interrupt
-
```

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG32
AX=0000 BX=0000 CX=0016 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=1C95 ES=1C95 SS=2000 CS=1CA5 IP=0000  NU UP DI PL NZ NA PE NC
1CA5:000D 50          PUSH    AX
Trace Interrupt
-t
AX=0000 BX=0000 CX=0016 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=1C95 ES=1C95 SS=2000 CS=1CA5 IP=000E  NU UP DI PL NZ NA PE NC
1CA5:000E 53          PUSH    BX
Trace Interrupt
-t
AX=0000 BX=0000 CX=0016 DX=0000 SP=000A BP=0000 SI=0000 DI=0000
DS=1C95 ES=1C95 SS=2000 CS=1CA5 IP=000F  NU UP DI PL NZ NA PE NC
1CA5:000F 5B          POP     AX
Trace Interrupt
-t
AX=0000 BX=0000 CX=0016 DX=0000 SP=000C BP=0000 SI=0000 DI=0000
DS=1C95 ES=1C95 SS=2000 CS=1CA5 IP=0010  NU UP DI PL NZ NA PE NC
1CA5:0010 5B          POP     BX
Trace Interrupt
-t
AX=0000 BX=0000 CX=0016 DX=0000 SP=000E BP=0000 SI=0000 DI=0000
DS=1C95 ES=1C95 SS=2000 CS=1CA5 IP=0011  NU UP DI PL NZ NA PE NC
1CA5:0011 B8004C      MOV     AX,4C00h
Trace Interrupt
-a_
-t
AX=4C00 BX=0000 CX=0016 DX=0000 SP=000E BP=0000 SI=0000 DI=0000
DS=1C95 ES=1C95 SS=2000 CS=1CA5 IP=0014  NU UP DI PL NZ NA PE NC
1CA5:0014 CD21      INT     21h ;End Program
Trace Interrupt
-t
Program terminated RC = 0
-a_

```

(2) 题目问题回答：

- 1) 生成可执行文件过程略（masm、link 即可生成）
- 2) 每一步相关寄存器内容和栈顶内容：

每步后	AX	BX	栈顶的内容
初始值	0	0	0
MOV AX,2000H	2000H	0	0
MOV SS,AX	2000H	0	0
ADD SP,0	2000H	0	0
ADD SP,10	2000H	0	0
POP AX	0	0	0
POP BX	0	0	0
PUSH AX	0	0	0
PUSH BX	0	0	0
POP AX	0	0	0
POP BX	0	0	0
POP AX	0	0	0
POP BX	0	0	0

3) 查看 PSP 的内容：（即数据段 DS 的前 100H 个字节）

```

-d 1C95:0
1C95:0000 CD 20 FF 9F 00 EA FF FF-AD DE BA 5D A2 01 00 00 M ...j..-^:l"...
1C95:0010 18 01 10 01 18 01 92 01-FF FF FF FF FF FF FF .....
1C95:0020 FF FF FF FF FF FF FF-FF FF FF FF 8B 1C F6 FF .....
1C95:0030 00 20 14 00 18 00 95 1C-FF FF FF FF 00 00 00 00 .....
1C95:0040 05 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
1C95:0050 CD 21 CB 00 00 00 00 00-00 00 00 00 00 00 00 M!K.....
1C95:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
1C95:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....

```

1.2.2 任务二：王爽版 实验 4

(1) (2) 编程，向内存 0:200-0:23F 依次传送数据 0~63，只能使用 9 条指令。

1) 源代码：见附件

2) 过程截图：（运行完后 -d 查看 0:200~0:23F 的内存空间）

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG32
C:\>debug32 exp41.exe
Debug32 - Version 1.0 - Copyright (C) Larson Computing 1994

CPU = 486, Real Mode, Id/Step = 0402, A20 disabled
-u
1CA5:0000 MOV AX,0020h
1CA5:0003 MOV DS,AX
1CA5:0005 MOV BX,0000h
1CA5:0008 MOV CX,0040h
1CA5:000B MOV [BX],BX
1CA5:000D INC BX
1CA5:000E LOOP 000B
1CA5:0010 MOV AX,4C00h
1CA5:0013 INT 21h
1CA5:0015 POP [265F1]
1CA5:0019 ADD [D8261],AX
1CA5:001D ADD [BX+SI],AL
-g CS:0013
AX=4C00 BX=0040 CX=0000 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0020 ES=1C95 SS=1CA4 CS=1CA5 IP=0013 NU UP DI PL NZ AC PO NC
1CA5:0013 CD21 INT 21h ;End Program
Instruction Breakpoint
-p
Program terminated RC = 0
-a_

-d 0:200 23F
0000:0200 00 01 02 03 04 05 06 07-08 09 0A 0B 0C 0D 0E 0F .....
0000:0210 10 11 12 13 14 15 16 17-18 19 1A 1B 1C 1D 1E 1F .....
0000:0220 20 21 22 23 24 25 26 27-28 29 2A 2B 2C 2D 2E 2F !"#$/&'()*+,-./
0000:0230 30 31 32 33 34 35 36 37-38 39 3A 3B 3C 3D 3E 3F 0123456789:;<=>?
-a_

```

(3) 补全程序：

1) 填空：mov ax,cs

mov cx,17h

2) 源代码：见附件

3) 过程截图: (运行程序后, -d 查看 0:200 处内存空间, 其内容为程序的机器码)

```
C:\>debug32 exp43.exe
Debug32 - Version 1.0 - Copyright (C) Larson Computing 1994

CPU = 486, Real Mode, Id/Step = 0402, A20 disabled
-u
1CA5:0000 BCC8      MOV     AX,CS
1CA5:0002 8ED8      MOV     DS,AX
1CA5:0004 B82000     MOV     AX,0020h
1CA5:0007 8EC0      MOV     ES,AX
1CA5:0009 BB0000     MOV     BX,0000h
1CA5:000C B91700     MOV     CX,0017h
1CA5:000F 8A07      MOV     AL,[BX]
1CA5:0011 268807     MOV     ES:[BX],AL
1CA5:0014 43        INC     BX
1CA5:0015 E2F8      LOOP    000F
1CA5:0017 B8004C     MOV     AX,4C00h
1CA5:001A CD21      INT     21h
-a

-d 0:200 217
0000:0200 8C C8 8E D8 B8 20 00 8E-C0 BB 00 00 B9 17 00 8A .H.XB ..e;..9...
0000:0210 07 26 88 07 43 E2 F8 00 .&..CbX.
```

4) 题目问题回答:

a) 复制的是什么: 复制的是指令的机器码

从哪里到哪里: 从代码段 CS 赋值到 0:200 处

b) 复制的有多少字节: 复制有 23 (17h) 个字节

如何知道复制的字节数量: 先待定复制的字节数, 将源程序 debug 使用 -u 反汇编查看代码所占的内存空间, 确定复制的字节数

1.2.3 任务三: 王爽版 实验 5

(1) 编译连接程序并回答问题

1) 过程截图: ((2)、(3) 题截图类似, 略)

```
-g CS:0020
AX=4C00 BX=0000 CX=0042 DX=0000 SP=0010 BP=0000 SI=0000 DI=0000
DS=1CA5 ES=1C95 SS=1CA6 CS=1CA7 IP=0020 NU UP DI PL NZ NA PO NC
1CA7:0020 CD21      INT     21h ;End Program
Instruction Breakpoint
```

2) 题目问题回答:

①data 段数据为: 0123h,0456h,0789h,0abch,0defh,0fedh,0cbah,0987h

②cs=0042、ss=1CA7、ds=1CA5

③data 段地址为 X-2, stack 段地址为 X-1

(2) 编译连接程序并回答问题

1) 题目问题回答:

①data 数据段为: 0123h, 0456h, 其余 12 个字节为 0

②CS=1CA7、SS=1CA6、DS=1CA5

③data 段地址为 X-2, stack 段地址为 X-1

④若段中的数据占 N 个字节, 程序加载后, 该段实际占空间为 $([N/16]+1)*16$ 字节 注: [x]表示向下取整

(3) 编译连接程序并回答问题

1) 题目问题回答:

① data 数据段为: 0123h, 0456h, 其余 12 个字节为 0

② CS=1CA5、SS=1CA9、DS=1CA8

③data 段地址为 X+3, stack 段地址为 X+4

(4) 若将 (1)、(2)、(3) 题最后一条伪指令改为 “end” (即不指名程序入口), 则哪个程序仍然可以正确执行? 请说明原因。

1) 题目问题回答:

仅 (3) 题的程序可以正确执行

原因: 若不指名程序入口, 则程序会从加载进内存的第一个单元开始执行, 前两题开始定义的是数据段和堆栈段, CPU 会依然当作指令执行, 会带来错误; 而第 (3) 题开始即为代码段, 所以可以正常执行。

(5) 编写 code 段代码, 将 a 段和 b 段中的数据依次相加, 将结果存到 c 段

1) 源代码: 见附件

2) 过程截图: (c 段的段地址在程序中为 1CA7)

(6) 编写 code 段代码，用 push 指令将 a 段前八个字型数据逆序存到 b 段中

1) 源代码：见附件

2) 过程截图：（b 段的段地址在程序中为 1CA7）

```
-u
1CA8:0000 B8A51C      MOV     AX,1CA5h
1CA8:0003 8ED8          MOV     DS,AX
1CA8:0005 B8A71C      MOV     AX,1CA7h
1CA8:0008 8ED0          MOV     SS,AX
1CA8:000A BC1000      MOV     SP,0010h
1CA8:000D BB0000      MOV     BX,0000h
1CA8:0010 B90800      MOV     CX,0008h
1CA8:0013 FF37          PUSH    Word Ptr [BX]
1CA8:0015 83C302      ADD     BX,02h
1CA8:0018 E2F9          LOOP    0013
1CA8:001A B44C          MOV     AH,4Ch
1CA8:001C CD21      INT     21h
-g CS:001c
AX=4CA7 BX=0010 CX=0000 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=1CA5 ES=1C95 SS=1CA7 CS=1CA8 IP=001C  NU UP DI PL NZ AC PO NC
1CA8:001C CD21      INT     21h ;End Program
Instruction Breakpoint
-d 1ca7:0 7
1CA7:0000 08 00 07 00 06 00 05 00 .....
-d 1ca7:0 f
1CA7:0000 08 00 07 00 06 00 05 00-04 00 03 00 02 00 01 00 .....
-
```

1.3 实验总结

本次实验更加熟悉了汇编语言的相关操作。首先学会了用 debug 的-g 命令和-p 命令调试带有循环的源程序。同时对汇编源程序的数据段及其程序加载后生成的段有了一定的了解，知道了代码段前有 256 个字节的 PSP 部分用来程序与 DOS 通信。最后，通过实验中编写汇编源程序的部分段代码来实现某项功能，熟悉了如何操作汇编程序的段地址、偏移地址及相关的寻址方法，了解了不同寄存器、以及代码中的标号、变量定义等在程序中的作用。

本次实验没有太多问题，遇到的大部分问题大多是程序算法上的问题，通过看书了解汇编语言的相关知识即可解决。

1.4 尚未解决问题

无

1.5 附件

1.5.1 实验 4（1）（2）源代码

编程，向内存 0:200-0:23F 依次传送数据 0~63，只能使用 9 条指令。

```
assume cs:code
```

```
code segment
```

```
    mov ax,0020h
```

```
    mov ds,ax    ;0020h 做 ds 段地址
```

```
    mov bx,0;bx 的值从 0 到 63 既做数据又做偏移地址
```

```

        mov cx,64    ;cx 的值作为循环变量

s:   mov [bx],bx
    inc bx
    loop s

    mov ax,4c00h
    int 21h
code ends
end

```

1.5.2 实验 4（3）源代码

补全程序，将“**mov ax,4c00h**”之前的指令复制到内存 0:200 处

```

assume cs:code
code segment
    mov ax,cs        ;cs 为代码段将其作为数据段（填空处）
    mov ds,ax
    mov ax,0020h
    mov es,ax        ;附加段储存段地址
    mov bx,0
    mov cx,17h       ;通过 debug 反汇编确定代码长度（填空处）
s:   mov al,[bx]      ;将数据段数据赋值到 0:200 处
    mov es:[bx],al    ;因为不能存储器到存储器，要寄存器过渡
    inc bx
    loop s
    mov ax,4c00h
    int 21h
code ends
end

```

1.5.3 实验 5（5）源代码

编写 code 段代码，将 a 段和 b 段中的数据依次相加，将结果存到 c 段

```

assume cs:code

a segment
    db 1,2,3,4,5,6,7,8
a ends

b segment
    db 1,2,3,4,5,6,7,8
b ends

```



```

c segment
    db 0,0,0,0,0,0,0,0
c ends

code segment
start:    mov ax,a
          mov ds,ax    ;ds 指向 a 段

          mov ax,c
          mov es,ax    ;es 指向 c 段

          mov bx,0;bx 指向偏移地址
          mov cx,8;循环 8 次

s:        mov al,[bx]
          add al,[bx+16]    ; (al) =a[i]+b[i]
          mov es:[bx],al    ;将和存到 c 段中

          inc bx
          loop s

          mov ah,4ch
          int 21h
code ends
end start

```

1.5.4 实验 5（6）源代码

编写 code 段代码，用 push 指令将 a 段前八个字型数据逆序存到 b 段中

```

assume cs:code
a segment
    dw 1,2,3,4,5,6,7,8,9,0ah,0bh,0ch,0dh,0eh,0fh,0ffh
a ends

b segment
    dw 0,0,0,0,0,0,0,0
b ends

code segment
start:    mov ax,a
          mov ds,ax    ;ds 指向 a 段

          mov ax,b

```

```
mov ss,ax    ;ss 指向 b 段
mov sp,16    ;sp 指向栈底

mov bx,0
mov cx,8

s:  push [bx];压栈
    add bx,2
    loop s

mov ah,4ch
int 21h
code ends

end start
```