

重庆大学

学生实验报告

实验课程名称 汇编语言程序设计

开课实验室 DS1501

学院 大数据与软件学院 年级 软件工程 专业班 01 班

学生姓名 _____ 学号 _____

开课时间 2024 至 2025 学年第 1 学期

| | |
|------|-----|
| 总成绩 | |
| 教师签名 | 陈蜀宇 |

大数据与软件工程学院 制

《汇编语言程序设计》实验报告

开课实验室： DS1501 机房

2024年10月31日

一【实验目的】

1. 熟悉 DEBUG 命令；
2. 掌握数据在内存中的存放方式和内存操作数的几种寻址方式，段寄存器和物理地址计算；
3. 熟悉数据传送指令、算术运算指令、逻辑指令、数据串传送程序和数据串传送指令 MOVS、STOS 及重复前缀 REP；
4. 掌握简单的汇编语言程序编写与调试。

二【实验环境】

1. PC 微机；
2. DOS 操作系统或 Windows 操作系统；
3. MASM.EXE, LINK.EXE, DEBUG.COM 或宏汇编集成环境。
4. DOSBOX.EXE（64 位 Windows 操作系统需要）。

DOSBOX.EXE 使用方法

1. 安装；
2. 运行；
3. 在输入框状态下 Z:>mount C D:\masm ==> “Z:>” 这个是提示符 “C” 作为虚拟 C 盘 “D:\masm” 要虚拟的文件夹位置；简洁的讲，把要虚拟的文件夹位置换掉上面的 D:\masm；
4. 在刚才的提示符下输入 C:，这样就切换到虚拟的 c 盘，也就是你要的目录；
5. 按照 DOS 操作系统环境要求运行 C: 目录下的 MASM.EXE, LINK.EXE, DEBUG.COM 等软件。如：C:>MASM 123.ASM 、C:>LINK 123.OBJ 、 C:>debug 123.exe 等。

三【实验要求】

1. 仔细阅读有关 DEBUG 命令的内容，对有关命令，要求事先准备好使用的例子；
2. 阅读数据传送指令、算术运算指令、逻辑指令、数据串传送程序和数据串传送指令 MOVS、STOS 及重复前缀 REP 的内容；
3. 用 DEBUG 的有关命令调试本实验中的求累加和程序、多字节加法程序、数据串搬家程序段。

四【预备知识】Debug 的使用

(1) 什么是 Debug?

Debug 是 DOS、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 各种寄存器中的内容、内存的情况和在机器码级跟踪程序的运行。

(2) 我们用到的 Debug 功能

- 用 Debug 的 R 命令查看、改变 CPU 寄存器的内容；
- 用 Debug 的 D 命令查看内存中的内容；
- 用 Debug 的 E 命令改写内存中的内容；
- 用 Debug 的 U 命令将内存中的机器指令翻译成汇编指令；
- 用 Debug 的 T 命令执行一条机器指令；
- 用 Debug 的 A 命令以汇编指令的格式在内存中写入一条机器指令。

五【实验内容】

1. 内存操作数及各种寻址方式使用；

- 2. 求累加和程序；
- 3. 多字节加法程序；
- 4. 数据串搬家程序；
- 5. 段寄存器概念及字符串传送指令练习。

六【实验步骤】

1. 内存操作数及各种寻址方式使用

```
MOV AX, 1234
MOV [1000], AX
MOV BX, 1002
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+1], DL
MOV WORD PTR[BX+SI+2], 2846
```

步骤:

(1) 用A命令键入上述程序，并用T命令逐条运行。

The screenshot shows a DOS terminal window titled 'C:\WINDOWS\system32\DEBUG.exe'. The command '-A' has been entered, followed by the assembly code provided in the question. The command '-T' has also been entered. Below the assembly code, the CPU register dump is shown:

```
AX=1234 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1397 ES=1397 SS=1397 CS=1397 IP=0103 NV UP EI PL NZ NA PO NC
1397:0103 A30010 MOU [1000],AX DS:1000=0000
```

(2) 每运行一条有关内存操作数的指令，要用D命令检查并记录有关内存单元的内容并注明是什么寻址方式。注意D命令显示结果中右边的ASCII字符及双字节数存放法。

The screenshot shows a DOS terminal window displaying the result of the 'D 100' command. The output shows memory starting at address 100, with each byte value followed by its ASCII representation. The command '1397:0100' is visible at the top.

```
-D 100
1397:0100 B8 34 12 A3 00 10 BB 02-10 C6 07 20 B2 39 43 88 .4.....9C.
1397:0100 17 FE CA BE 03 00 88 10-88 50 01 C7 40 02 46 28 .....P..@.F<
1397:0120 CC 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 ...
1397:0130 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 ...
1397:0140 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 ...
1397:0150 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 ...
1397:0160 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 ...
1397:0170 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 ...
```

```

C:\WINDOWS\system32\DEBUG.exe
-T
AX=1234 BX=1003 CX=0000 DX=0038 SP=FFEE BP=0000 SI=0003 DI=0000
DS=1397 ES=1397 SS=1397 CS=1397 IP=0118 NV UP EI PL NZ NA PO NC
1397:0118 885001 MOU [BX+SI+01],DL DS:1007=00
-T
AX=1234 BX=1003 CX=0000 DX=0038 SP=FFEE BP=0000 SI=0003 DI=0000
DS=1397 ES=1397 SS=1397 CS=1397 IP=011B NV UP EI PL NZ NA PO NC
1397:011B C740024628 MOU WORD PTR [BX+SI+02],2846 DS:1008=0000
-T
AX=1234 BX=1003 CX=0000 DX=0038 SP=FFEE BP=0000 SI=0003 DI=0000
DS=1397 ES=1397 SS=1397 CS=1397 IP=0120 NV UP EI PL NZ NA PO NC
1397:0120 0000 ADD [BX+SI],AL DS:1006=38
-D 100
1397:0100 B8 34 12 A3 00 10 BB 02-10 C6 07 20 B2 39 43 88 .4.....9C.
1397:0110 17 FE CA BE 03 00 88 10-88 50 01 C7 40 02 46 28 .....P..@.F<
1397:0120 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00
1397:0130 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00
1397:0140 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00
1397:0150 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00
1397:0160 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00
1397:0170 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00
-
```

2. 求累加和程序:

程序:

```

MOV BX, 1000
MOV CX, 10
SUB AX, AX
LOP: ADD AL, [BX]
      ADC AH, 0
      INC BX
J:   LOOP LOP
INT 3

```

步骤:

- (1) 用A命令将程序键入到100H开始的内存中，在键入时记下标号LOP和J的实际地址，在键入LOOP指令时LOP用实际地址值代替；

The screenshot shows the DEBUG environment. On the left, assembly code is being typed into the command window:

```

-a
0AE8:0100 mov bx,1000
0AE8:0103 mov cx,10
0AE8:0106 sub ax,ax
0AE8:0108 add al,[bx]
0AE8:010A adc ah,0
0AE8:010D inc bx
0AE8:010E loop 0108
0AE8:0110 int 3

```

On the right, three windows show the memory dump results:

- aa.com:** Shows the memory dump from address 0000 to 0010. It contains the assembled code.
- BX:** Shows the value of BX register, which is 0000.
- CX:** Shows the value of CX register, which is 0010.
- Writing 00010 bytes:** Confirmation message indicating the successful write of 10 bytes.

- (2) 用命令N AA 将此程序命名为文件AA(文件名可任取);
- (3) 用R命令将 BX: CX 改为程序长度值(即最后一条指令后面的地址减去开始地址);
- (4) 用命令W 100将此程序存到AA命名的磁盘文件中;
- (5) 用命令Q退出DEBUG;
- (6) 用命令DEBUG AA再次调入DEBUG和文件AA，可用U命令检查调入程序;

The screenshot shows the DEBUG environment again. The command window displays:

```

C:\DOCUMENTS\XPMUser>debug aa.com
-u 0100 110
0B2F:0100 BB0010      MOU     BX,1000
0B2F:0103 B91000      MOU     CX,0010
0B2F:0106 29C0        SUB     AX,AX
0B2F:0108 0207        ADD     AL,[BX]
0B2F:010A 80D400      ADC     AH,00
0B2F:010D 43          INC     BX
0B2F:010E E2F8        LOOP    0108

```

Below the command window, a message indicates the program has been successfully loaded.

- (7) 用E命令在内存地址1000H处键入16个数字;

```
-e 1000  
0B2F:1000 06.1 F9.2 72.3 03.4 5A.5 5B.6 58.7 C3.8  
0B2F:1008 1E.9 06.a 1F.b 83.c F9.d 01.e 75.f 1B.10
```

(8) 用命令G=100 J(J用实际地址代替), 使程序运行并停在断点J上, 检查AX, BX的值是否符合你的预计值; 如: G=0100 010D

【说明: G就是连续执行内存代码, 可以在G后面指定内存地址 如G=0000:0100; 地址可以是偏移地址, 也可以是逻辑地址。如不指定地址, G就会从指令指针ip指的内存处开始执行; G命令还可以指定执行断点, 如, G=0000:0100 0000:0200 就是代码从100的内存地址执行到200的内存地址然后断住。】

```
-g=100 010d  
AX=0001 BX=1000 CX=0010 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000  
DS=0B2F ES=0B2F SS=0B2F CS=0B2F IP=010D NU UP EI PL ZR NA PE NC  
0B2F:010D 43 INC BX
```

(9) 用T命令运行一步, 观察程序方向(IP值)和CX值是否与你的估计一样, 若不一样, 检查程序是否有错;

```
-t  
AX=0001 BX=1001 CX=0010 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000  
DS=0B2F ES=0B2F SS=0B2F CS=0B2F IP=010E NU UP EI PL NZ NA PO NC  
0B2F:010E E2F8 LOOP 0108
```

(10) 重复G、J与T, 再检查AX是否正确;

(11) 用G命令使程序运行到结束, 检查AX值是否正确。

【说明: G=100 是指从偏移量: 0100 开始执行, 直到程序结束为止。】

```
-g=100  
AX=0594 BX=1010 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000  
DS=0B2F ES=0B2F SS=0B2F CS=0B2F IP=0110 NU UP EI PL NZ AC PO NC  
0B2F:0110 CC INT 3
```

3. 多字节加法程序

程序:

```
MOV DI, 1000  
MOV CX, 8  
MOV SI, 2000  
CLC  
LOP: MOV AL, [SI]  
      ADC [DI], AL  
      INC SI  
      INC DI  
      LOOP LOP  
      INT 20
```

步骤:

(1) 用命令键入此程序。

```

-a
13A8:0100 MOV DI,1000
13A8:0103 MOV CX,8
13A8:0106 MOV SI,2000
13A8:0109 CLC
13A8:010A MOV AL,[SI]
13A8:010C ADC [DI],AL
13A8:010E INC SI
13A8:010F INC DI
13A8:0110 LOOP 010A
13A8:0112 INT 20
13A8:0114
-E 1000
13A8:1000 00.11 00.11 00.11 00.11 00.11 00.11 00.11 00.11 00.11
13A8:1008 00.
-E 2000
13A8:2000 00.22 00.22 00.22 00.22 00.22 00.22 00.22 00.22 00.22
-G=100

```

(2) 用E命令在1000H开始处键入一个8字节被加数，在2000H开始处键入一个8字节加数，均为低字节在前面。

```

-E 1000
13A8:1000 00.11 00.11 00.11 00.11 00.11 00.11 00.11 00.11 00.11
13A8:1008 00.
-E 2000
13A8:2000 00.22 00.22 00.22 00.22 00.22 00.22 00.22 00.22 00.22
-G=100

Program terminated normally
-D 1000
13A8:1000 33 33 33 33 33 33 33 33-00 00 00 00 00 00 00 00 00 00 33333333.-----
13A8:1010 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00 00 -----
13A8:1020 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00 00 -----
13A8:1030 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00 00 -----
13A8:1040 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00 00 -----
13A8:1050 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00 00 -----
13A8:1060 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00 00 -----
13A8:1070 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00 00 -----

```

4. 数据串搬家程序

(1) 用A命令键入下列程序：

```

MOV SI, 1000
MOV DI, 1500
MOV CX, 0FH
LOP: MOV AL, [SI]
      MOV [DI], AL
      INC SI
      INC DI
      LOOP LOP
      INT 20

```

```

-a
13A8:0100 mov si,1000
13A8:0103 mov di,1500
13A8:0106 mov cx,0f
13A8:0109 mov al,[si]
13A8:010B mov [di],al
13A8:010D inc si
13A8:010E inc di
13A8:010F loop 0109
13A8:0111 int 20
13A8:0113

```

(2) 用A命令DB伪指令在1000H键入下列字符串：

‘IBM PC COMPUTER’

```

-D 1000
13A8:1000 49 42 4D 5F 50 43 20 43-4F 4D 50 55 54 45 52 00 IBM_PC COMPUTER.
13A8:1010 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00 00
13A8:1020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00
13A8:1030 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00

```

(3) 用G命令运行此程序，并用D命令检查目的地址处的字符与源串是否一致。

【说明： G=100 是指从偏移量： 0100 开始执行， 直到程序结束为止。】

```

-G=100
Program terminated normally
-D 1000 LF
0AE8:1000 49 42 4D 5F 50 43 20 43-4F 4D 50 55 54 45 52 IBM_PC COMPUTER
-D 1500 LF
0AE8:1500 49 42 4D 5F 50 43 20 43-4F 4D 50 55 54 45 52 IBM_PC COMPUTER

```

5. 段寄存器概念及字符串传送指令练习

(1) 用A命令键入下列程序：

```

MOV SI, 0
MOV DI, 0
MOV AX, 1000
MOV DS, AX
MOV AX, 1500
MOV ES, AX
MOV CX, 0F
CLD
REP MOVSB
INT 20

```

```

-A
13A8:0100 MOU SI,0
13A8:0103 MOU DI,0
13A8:0106 MOU AX,1000
13A8:0109 MOU DS,AX
13A8:010B MOU AX,1500
13A8:010E MOU ES,AX
13A8:0110 MOU CX,F
13A8:0113 CLD
13A8:0114 REP MOUSB
13A8:0116 INT 20
13A8:0118

```

(2) 用A命令DB伪指令在1000: 0000处键入字符串'IBM_PC COMPUTER'，并用D命检查

(3) 用D命令检查0F00: 1000处内容，看是否一样。为什么？

```

-A 1000:0000
1000:0000 DB 'IBM_PC COMPUTER'
1000:000F
-D 1000:1000
1000:1000 49 42 4D 5F 50 43 20 43-4F 4D 50 55 54 45 52 65 IBM_PC COMPUTERRe
1000:1010 74 65 72 73 20 20 20 20-20 53 70 65 63 69 66 69 ters Specifi
1000:1020 65 73 20 63 6F 6D 6D 61-6E 64 2D 6C 69 6E 65 20 es command-line
1000:1030 69 6E 66 6F 72 6D 61 74-69 6F 6E 20 72 65 71 75 information requ
1000:1040 69 72 65 64 20 62 79 0D-0A 20 20 20 20 20 20 20 20 ired by..
1000:1050 20 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20 20
1000:1060 20 20 20 74 68 65 20 66-69 6C 65 20 79 6F 75 20 the file you
1000:1070 77 61 6E 74 20 74 6F 20-74 65 73 74 2E 0D 0A 0D want to test....

```

(4) 用G命令运行此程序，检查目的地址1500: 0000处内容。并记下DS与ES值。

【说明： G=100 是指从偏移量： 0100 开始执行， 直到程序结束为止。】

```

-G=100
Program terminated normally
-D 1500:0000
1500:0000 49 42 4D 5F 50 43 20 43-4F 4D 50 55 54 45 52 00 IBM_PC COMPUTER.
1500:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1500:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1500:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1500:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1500:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1500:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1500:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00

```

此时的DS和ES的值分别为：

(5) 修改此程序，使ES与DS为同一值，以减少一条指令，而源物理地址和目的物理地址(是多少?)仍和原来一样。

```

-A
13A8:0100 MOV SI,0
13A8:0103 MOV DI,0
13A8:0106 MOV AX,1000
13A8:0109 MOV DS,AX
13A8:010B MOV AX,1500
13A8:010E MOV ES,AX
13A8:0110 MOU CX,F
13A8:0113 CLD
13A8:0114 REP MOUSB
13A8:0116 INT 20
13A8:0118

命令提示符 - DEBUG

-A 1000:0000
1000:0000 DB 'IBM_PC COMPUTER'
1000:000F
-D 1000:1000
1000:1000 49 42 4D 5F 50 43 20 43-4F 4D 50 55 54 45 52 65 IBM_PC COMPUTER
1000:1010 74 65 72 73 20 20 20 20-20 53 70 65 63 69 66 69 Specifi
1000:1020 65 73 20 63 6F 6D 6D 61-6E 64 2D 6C 69 6E 65 20 es command-line
1000:1030 69 6E 66 6F 72 6D 61 74-69 6F 6E 20 72 65 71 75 information requ
1000:1040 69 72 65 64 20 62 79 0D-0A 20 20 20 20 20 20 20 20 ired by..
1000:1050 20 20 20 20 20 20 20-20 20 20 20 20 20 20 20 20
1000:1060 20 20 20 74 68 65 20 66-69 6C 65 20 79 6F 75 20 the file you
1000:1070 77 61 6E 74 20 74 6F 20-74 65 73 74 2E 0D 0A 0D want to test...
-G=100

Program terminated normally
-D 1500:0000
1500:0000 49 42 4D 5F 50 43 20 43-4F 4D 50 55 54 45 52 00 IBM_PC COMPUTER.
1500:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1500:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1500:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1500:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1500:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1500:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1500:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00

```

6. 自编程序

用16位减法指令编一个32位(4字节)数减法程序，两个源数及结果存放地址同多字节加法程序，调试并做记录。

4字节减法程序源代码

```

1397:0100 MOV SI,1000
1397:0103 MOV CX,4
1397:0106 MOV DI,2000
1397:0109 CLC
1397:010A MOV AL,[SI]

```

1397:010C SBB [DI],AL

1397:010E INC SI

1397:010F INC DI

1397:0110 LOOP 010A

1397:0112 INT 20

1397:0114

调试和运行

```
-d 2000 l 4
1397:2000 33 33 33 33
-d 1000 l 4
1397:1000 55 55 55 55
-g=100

Program terminated normally
-d 1000 l 4
1397:1000 55 55 55 55
-d 2000 l 4
1397:2000 DE DD DD DD
.....
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=1397 ES=1397 SS=1397 CS=1397 IP=0100 NV UP EI PL NZ NA PO NC
1397:0100 BE0010 MOU SI,1000
-
```

结果运行正确

七【实验过程原始记录(数据、图表、计算等)】

如果有则写。

八【思考题】

1. MOV BYTE PTR[BX], 20
MOV WORD PTR[BX+SI+2], 2846

上述指令中BYTE PTR及WORD PTR伪操作不加行不行?试一试。

```
-A  
1397:0100 MOU BYTE PTR[BX],20  
1397:0103 MOU [BX],20  
1397:0103 MOU WORD PTR[BX+SI+2],2846  
1397:0108 MOU [BX+SI+2],2846  
1397:0108
```

2. 用G命令运行多字节加法程序，并用D命令检查其结果(存放在哪里？)，是否正确？

```
-E 1000
13A8:1000 00.11 00.11 00.11 00.11 00.11 00.11 00.11 00.11 00.11
13A8:1008 00.
-E 2000
13A8:2000 00.22 00.22 00.22 00.22 00.22 00.22 00.22 00.22 00.22
-G=100
```

3. 将多字节加法程序中的INT 20H指令改为INT 3，有何区别？若这条指令不加，行不行？试一试。

```
-a  
0AE8:0100 MOU    DI,1000  
0AE8:0103 MOU    CX,8  
0AE8:0106 MOU    SI,2000  
0AE8:0109 CLC  
0AE8:010A MOU    AL,[SI ]  
0AE8:010C ADC    [DI ],AL  
0AE8:010E INC    SI  
0AE8:010F INC    DI  
0AE8:0110 LOOP   010A  
0AE8:0112 INT    3  
0AE8:0113
```

```
-E 1000  
0AE8:1000 59.11 5B.11 C3.11 53.11 51.11 2E.11 C6.11 06.11  
-E 2000  
0AE8:2000 20.22 66.22 69.22 6C.22 65.22 20.22 6E.22 6F.22  
-G=100  
  
AX=0022 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=2008 DI=1008  
DS=0AE8 ES=0AE8 SS=0AE8 CS=0AE8 IP=0112 NV UP EI PL NZ NA PO NC  
0AE8:0112 CC INT 3
```

4. 数据串搬家程序中的目的地址改为1002H，再运行此程序，看能不能把1000H开始的字符串搬到1002H开始的地方去？修改程序以做到这一点。

不能结果不正确