



燕山大学

汇编语言程序设计实验报告

Assembler Language Programming Experiment Report

学 院：信息科学与工程学院（软件学院）

班 级：18 级软件工程 6 班

姓 名：乔翱

学 号：201811040809

指导教师：郝晓冰 李可 陈贺敏

教 务 处

2020 年 10 月

目 录

实验 1 汇编语言源程序的输入.....	3
1.1 实验目的.....	3
1.2 实验原理.....	3
1.3 实验仪器.....	3
1.4 实验步骤.....	3
1.5 实验要求.....	6
实验 2 数据的建立与传送程序.....	7
2.1 实验目的.....	7
2.2 实验原理.....	7
2.3 实验仪器.....	12
2.4 实验步骤.....	12
2.5 实验要求.....	12
实验 3 分支程序设计.....	14
3.1 实验目的.....	14
3.2 实验原理.....	14
3.3 实验仪器.....	14
3.4 实验步骤.....	14
3.5 实验要求.....	21
实验 4 统计学生成绩程序.....	22
4.1 实验目的.....	22
4.2 实验原理.....	22
4.3 实验仪器.....	22
4.4 实验步骤.....	22
4.5 实验要求.....	25
实验 5 学生成绩名次表实验.....	27
5.1 实验目的.....	27
5.2 实验原理.....	27
5.3 实验仪器.....	29
5.4 实验步骤.....	29
5.5 实验要求.....	29

实验 1 汇编语言源程序的输入

1.1 实验目的

1. 通过实验了解和熟悉微机系统的配置。
2. 学习在 DEBUG 状态下输入汇编源程序的方法。
3. 初步掌握调试(在 DEBUG 状态下)的过程。

1.2 实验原理

1、本实验要求在 DEBUG 状态下输入汇编源程序,并用 DEBUG 命令进行调试。用单步跟踪的方法验证指令的功能。

2、以下是给定的参考程序,并在实验时在每条指令的“;”符号右边按要求填写指令的执行结果。

注:(1)微机进入 DEBUG 状态下之后,一切立即数和地址数据均被默认为十六进制数,在输入时数的后面不加后缀“H”;

(2)在 DEBUG 状态下执行程序时,“INT 20H”指令可使系统执行完该指令前的程序时返回到“-”提示符状态,并且恢复 CS 和 IP 寄存器原来的值。

1.3 实验仪器

微机一台

1.4 实验步骤

1、开机后进入 DOS 系统,

C > DEBUG↵ (↵回车符)

— (为 DEBUG 提示符)

当显示器出现提示符“-”时,说明已进入 DEBUG 状态,这时,可用 DEBUG 命令进行操作。

2、用 DEBUG 的 Register 命令检查所有寄存器内容,并作记录。命令格式:

R [寄存器名]

该命令的功能是显示寄存器的内容,或修改某一指定寄存器内容,若[寄存器名]缺省,则显示所有寄存器内容。

例如:—R

3、用 DEBUG 的 Assemble 命令输入汇编源程序。

格式: A [内存地址]

注: 用 “[]” 符号括起来的部分表示可以省略。

该命令的功能是从指定的内存地址开始 (括号不要输入) 逐条输入汇编语言源程序并汇编成机器码存入内存。若地址缺省, 则接上一个 A 命令最后一条指令之后输入汇编语句, 若没有用过 A 命令, 则从 CS: 0100H 地址开始输入。

例如: -A 0CD3: 0100-

在输入 A 命令之后, 或每输入一条指令之后, 显示器的左端给出了内存的段地址和偏移地址。

每条指令均用回车 (✓) 结束。若输入的指令有语法错误, DEBUG 拒绝接收, 并给出提示, 此时可以重新输入。程序的最后一条指令输入完之后, 再按一次回车键 (✓), 即可结束汇编命令, 回到 DEBUG 提示符 “-” 状态。

4、用 DEBUG 的 Unassemble 命令反汇编。

命令格式:

U [起始地址[终止地址]]

该命令的功能是从起始地址到终止地址反汇编目标码, 缺省值是接上一个 U 命令或从 CS: 0100H 地址开始。例如:

-U

显示器上将显示程序的内存地址、指令机器码的汇编源程序三列对照清单。

5、用 DEBUG 的 Trace 命令单步跟踪程序。

命令格式:

T [=起始地址] [指令条数]

该命令的功能是从指定的起始地址开始逐条执行指令, 每执行完一条指令, 屏幕显示所有寄存器内容和下一条指令地址和指令。若 [=起始地址] 缺省, 则 T 命令从 CS: IP 地址开始执行指令。例如: -T✓

重复这一过程, 即可看到每条指令执行后, 所有寄存器和标志寄存器的标志位内容。此时, 要检查内存单元的数据, 可用 DEBUG 的 D 命令。

6、用 DEBUG 的 Dump 命令显示存储器单元的内容。命令格式:

D[起始地址[终止地址]]

该命令的功能是从起始地址到终止地址, 连续显示存储器单元的内容。若地址缺省, 则接上一个 D 命令或从 DS: 0100H 地址开始显示。例如: -D✓

参考程序:

```
MOV AX, 2000          ; AL=00H
MOV DS, AX             ; DS=2000H
NOT AX                 ; AX=FFFFH
XOR AX, AX             ; AX=0000H
DEC AX                 ; AX=FFFFH
INC AX                 ; AX=0000H
MOV BX, 2030           ; BH=20H
MOV SI, BX             ; SI=2030H
MOV [SI], BL           ; [2030H]=30H
MOV WORD PTR[SI], 10F   ; [2030H]= 0FH [2031H]=01H
MOV DI, SI             ; DI=2030H
MOV [DI+50], BH        ; [DI+50H]=20H
MOV BH, [SI]           ; BH=0FH
MOV BL, [DI+50]        ; BL=20H
MOV SP, 5000
PUSH AX                ; AX= 0000H [SS: 4FFEh]=00H [SS: 4FFFh]=00H
PUSH BX                ; BX=0F20H [SS: 4FFCh]=20H [SS: 4FFDh]=0FH
POP AX                 ; AX=0F20H
POPF                   ; F=00000000
NEG BX                 ; BX=F0E0H
XCHG BX, AX            ; BX=0F20H
STD                    ; F=01010001
STI                    ; F=01110001
CLD                    ; F=00110001
CLI                    ; F=00010001
ADC DI, 2050           ; DI=4081H F=00000010
ADC SP, DI             ; SP=9081H F=10010010
ADC AX, 1500           ; AX=05E0H F=00000001
SUB AX, BX             ; AX=F6C0H BX=0F20H
SHL AH, 1              ; AH=ECH
```

RCL AX, 1	; AX=D981H
SHR BH, 1	; BH=07H
RCR BL, 1	; BL=90H
MOV CL, 4	
MOV DX, 80F0	
ROL DX, CL	; DX=0F08H CL=04H
INT 20	; CS=F000H IP=1480H

1.5 实验要求

1、整理每条指令执行的结果，填到打印清单的右半部分(应注意内存数据检查的正确性)。

2、比较实验记录与理论分析的结果是否相同，若有不同，找出差别及问题所在。

对于一些指令的用法和作用不是特别熟悉，导致实验记录与理论分析的结果不同，通过查阅书籍以及其他的资料了解了这些指令的用法。

std 指令：置位方向标志，DF=0，串操作后地址增大。

sti 指令：复位方向标志，DF=1，串操作后地址减少。

cli 指令：复位方向标志，IF=0，禁止可屏蔽中断。

sti 指令：置位方向标志，IF=1，允许可屏蔽中断。

pushf 指令：标志寄存器压入堆栈。

popf 指令：堆栈顶部一个字量数据弹出到标志寄存器。

程序员可以改变段地址和偏移地址，但是在这个过程中如果需要改变段寄存器 SS 和 SP 必须禁止中断，当改变完成后再恢复中断（也就是说在 cli 指令后需要有与其配对的 sti 指令，否则计算机最常见的反应就是崩溃）

3、总结本次实验的体会。

通过本次实验，了解和熟悉了微机系统的配置，学会了如何在 DEBUG 状态下输入汇编语言的方法，掌握了在 DEBUG 状态下调试程序的过程。

通过本次实验，更加深刻地体会到了汇编语言中一些指令的用法和作用，发现自己对于某些指令的功能掌握的不是很好，例如：popf、std 等指令，通过查阅相关资料了解了这些指令的用法以及功能，更好的理解了这些指令，对汇编程序的一些常用指令有了一个很好的掌握。

实验 2 数据的建立与传送程序

2.1 实验目的

- 1、继续学习 DEBUG 命令。
- 2、验证指令的功能。

2.2 实验原理

在 DEBUG 状态下，分别输入下面各程序段，每输入完一个程序段，用 G 命令进行连续方式执行程序，在连续执行时，要记录程序的执行结果。

参考程序：

1. 在内存 10000H 单元开始，建立 00H~0FH~00H 31 个数，要求 00H~0FH 数据逐渐增大，0FH~00H 逐渐减小。该程序从内存 CS:0100H 地址开始输入。

```
MOV AX, 1000H
MOV DS, AX
MOV SI, 0
MOV CL, 0FH
XOR AX, AX
PPE1: MOV [SI], AL
      INC SI
      INC AL
      DEC CL
      JNZ PPE1
      MOV CX, 10H
PPE2: MOV [SI], AL
      INC SI
      DEC AL
      LOOP PPE2
      INT 20H
```

注：转移指令的符号地址直接用绝对偏移地址，该地址在用 A 命令汇编输入时，可以看到程序全部运行完之后，可用 DEBUG 的 Dump 命令查看建立的数据块内容。例如：

—D1000: 00 1E

在 debug 下执行程序，执行完利用-D 查看内存单元，可以看到建立的数据块内容
实验运行截图如下：

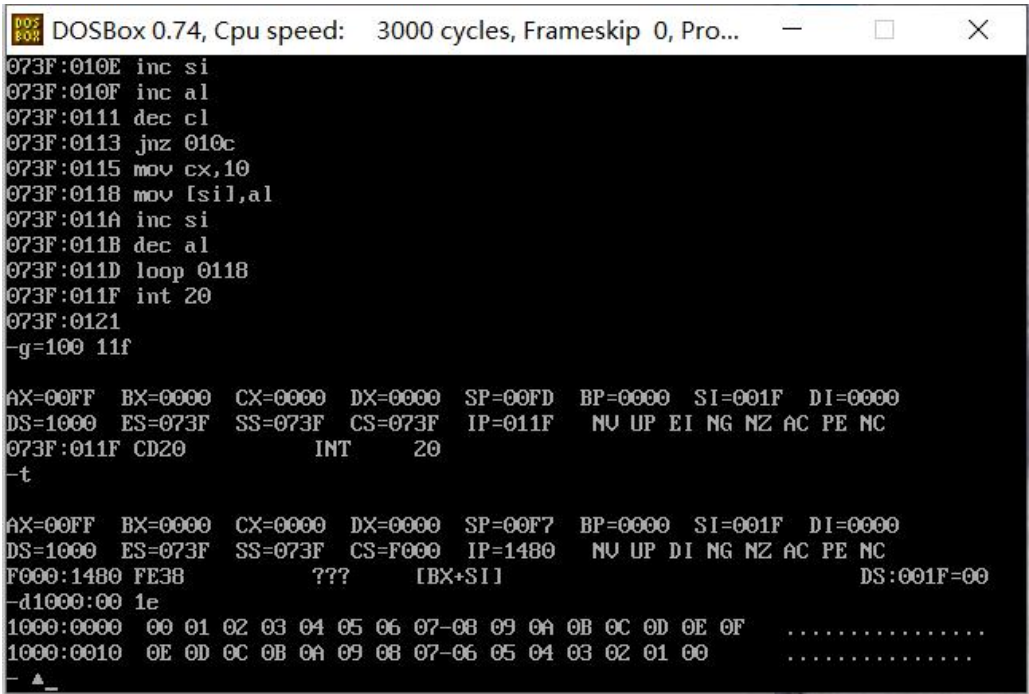


图 2.1 建立数据块运行程序截图

2、把上一个程序的执行结果(建立的 31 个字节数据块,其首地址在 10000H)，分几种方式传送到以下指定的区域。

该程序从内存 CS: 0150H 开始输入。把数据块传送到 15050H 开始的存贮区域中。

参考程序：

```
MOV AX, 1000H
MOV DS, AX
MOV SI, 0
MOV DI, 5050H
MOV CX, 1FH ; 数据块长度是 31
PPPA: MOV AL, [SI]
      MOV [DI], AL
      INC SI
      INC DI
      LOOP PPEA
      INT 20H
```

检查内存数据块的传送情况，可用“D”命令。

利用循环和传送指令，实现数据的传送。首先先是对数据段寄存器 DS 进行初始化，对 DI 和 SI 初始化，赋值计数寄存器 CX 为要传送的数据的个数（1FH），接着进行循环操作，传送数据。

利用-d1000:5050 506e，查看数据块传送到 15050H 开始的存贮区域的结果，实验截图如下：

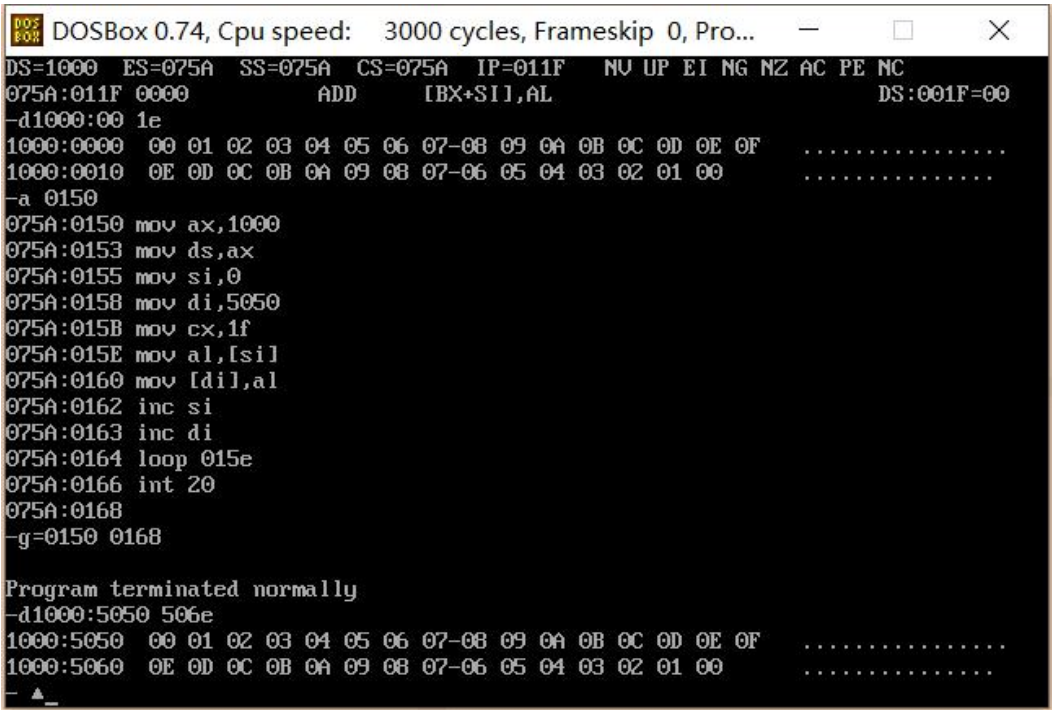


图 2.2 利用循环和传送指令传送数据块程序运行截图

(b) 用串传送指令 MOVSB, 把数据块传送到 15150H 开始的区域, 该程序从内存 CS:0200H 开始输入。

检查程序最后的执行结果，可用“D”命令, 例如：

—D1000: 5150 ✓

编写的程序代码：

```
MOV AX,1000
MOV DS,AX
MOV ES,AX
MOV SI,0
MOV DI,5150
MOV CX,1F
```

```
CLD
PPP:  MOVSB
      LOOP PPP
      INT 20
```

利用 MOVSB 指令和 loop 循环传送数据块到指定位置，利用 -d 查看运行结果，实验截图如下：

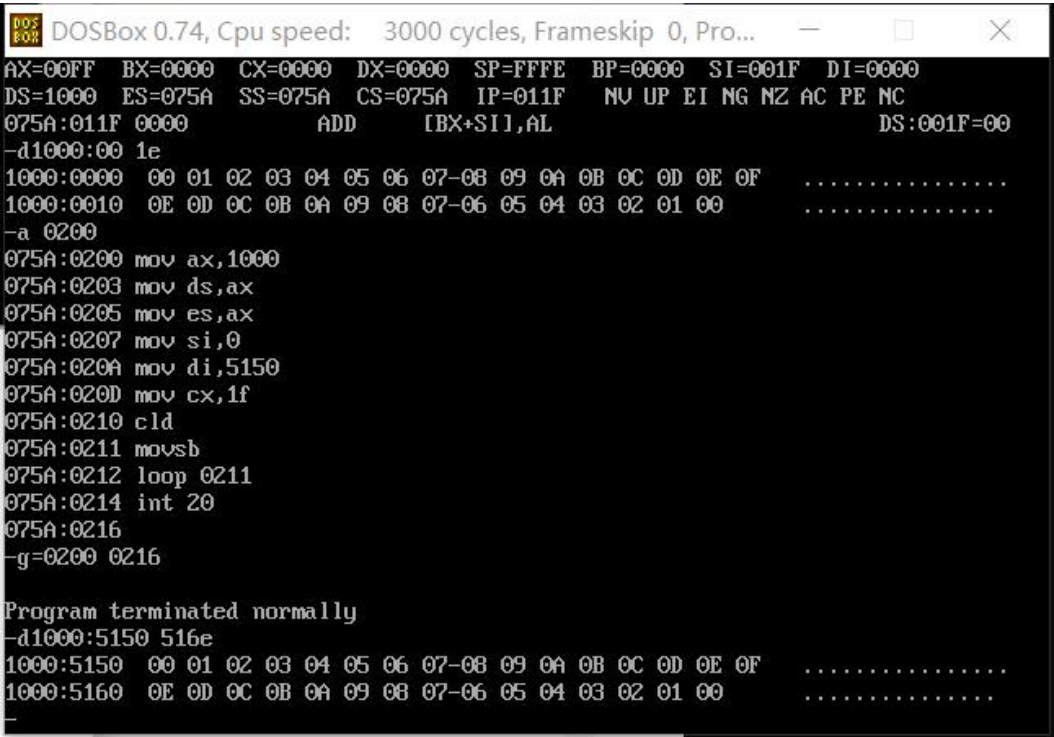


图 2.3 利用串传送指令 MOVSB 传送数据块程序运行截图

(c) 用重复串操作指令“REP MOVSB”把数据块传送到 15250H 开始的区域。该程序从 CS: 50H 地址开始输入。

检查程序的最后执行结果时，可用：

```
-D1000: 5250H
```

编写的程序代码

```
MOV AX, 1000
MOV DS, AX
MOV ES, AX
MOV SI, 0
MOV DI, 5250
MOV CX, 1F
```

```
CLD
REP MOVSB
INT 20
```

利用 rep movsb 循环传送数据块到指定位置，利用 -d 查看运行结果，实验截图如下：

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...
AX=00FF BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=001F DI=0000
DS=1000 ES=075A SS=075A CS=075A IP=011F  NU UP EI NG NZ AC PE NC
075A:011F 0000          ADD     [BX+SI],AL          DS:001F=00
-d1000:00 1e
1000:0000 00 01 02 03 04 05 06 07-08 09 0A 0B 0C 0D 0E 0F .....
1000:0010 0E 0D 0C 0B 0A 09 08 07-06 05 04 03 02 01 00 .....
-a 250
075A:0250 mov ax,1000
075A:0253 mov ds,ax
075A:0255 mov es,ax
075A:0257 mov si,0
075A:025A mov di,5250
075A:025D mov cx,1f
075A:0260 cld
075A:0261 rep movsb
075A:0263 int 20
075A:0265
-g=0250 0265

Program terminated normally
-d1000:5250 526e
1000:5250 00 01 02 03 04 05 06 07-08 09 0A 0B 0C 0D 0E 0F .....
1000:5260 0E 0D 0C 0B 0A 09 08 07-06 05 04 03 02 01 00 .....
```

图 2.4 利用 rep movsb 传送数据块程序运行截图

(d) 用串操作的减量工作方式，把数据块传送到 25050H 开始的区域。该程序从 CS:0300H 开始输入。

检查程序的最后执行结果，用 D 命令：

-D2000: 5050✓

```
MOV AX, 1000
MOV DS, AX
ADD AX, AX
MOV ES, AX
MOV SI, 1E
MOV DI, 506E
MOV CX, 1F
STD
REP MOVSB
INT 20
```

用串操作的减量工作方式，即使用 STD 指令传送数据块，程序运行截图如下：

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...
AX=00FF BX=0000 CX=0000 DX=0000 SP=FFFE BP=0000 SI=001F DI=0000
DS=1000 ES=075A SS=075A CS=075A IP=011F  NU UP EI NG NZ AC PE NC
075A:011F 0000      ADD     IBX+SI,AL      DS:001F=00
-d1000:00 1e
1000:0000 00 01 02 03 04 05 06 07-08 09 0A 0B 0C 0D 0E 0F .....
1000:0010 0E 0D 0C 0B 0A 09 08 07-06 05 04 03 02 01 00 .....
-a 0300
075A:0300 mov ax,1000
075A:0303 mov ds,ax
075A:0305 add ax,ax
075A:0307 mov es,ax
075A:0309 mov si,1e
075A:030C mov di,506e
075A:030F mov cx,1f
075A:0312 std
075A:0313 rep movsb
075A:0315 int 20
075A:0317
-g=0300 0317

Program terminated normally
-d2000:5050 506e
2000:5050 00 01 02 03 04 05 06 07-08 09 0A 0B 0C 0D 0E 0F .....
2000:5060 0E 0D 0C 0B 0A 09 08 07-06 05 04 03 02 01 00 .....
    
```

图 2.5 利用串操作的减量工作方式传送数据块程序运行截图

2.3 实验仪器

微机一台。

2.4 实验步骤

参照实验一的步骤，按照本实验程序的内容，分别输入各段程序，连续执行程序，记录每个程序段的最后结果。若想把源数据块重新换一批数据，可以用 DEBUG 的 Fill 命令填充新的数据。例如：

—F1000 : 00 L1F 33✓

从 1000: 0000H 开始的 31 个字 节被替换成 33H。

2.5 实验要求

1、整理每个程序段在实验时的记录内容。

2、比较每个程序段的特点。

本次实验首先建立了一个数据段，接着采用四种不同的方式进行了数据段的传送。第一种方法是采用 mov 传送指令加 loop 循环指令进行数据块的传送，第二种是采用串传送指令

movsb 进行数据块的传送，movsb 是专门传送串的指令，第三种是采用 reg+movsb 指令，reg 的作用也是循环，是为了传送串循环，第四种是采用减量的方式，即每次串传送后地址减少。

本次实验通过四种方式实现了程序块的传送，四种方式最后实现的效果都是一样的，实验后我认为采用 reg+movsb 进行数据块的传送最方便，从实验中可以看出串传送指令对于串的操作很方便，极大的方便了对于串的一系列操作。

3、分析本次实验出现的问题，找出问题所在。

通过本次实验主要学会了汇编语言中的循环程序设计，以及一些串传送指令，来对数据块进行传送。由于平时对于串传送指令的掌握不够熟练，在做本次实验时查询了课本以及其他资料，了解并且掌握了串传送指令的使用，本次实验涉及到的串传送指令主要是 MOVSB 和 Rep 指令。

MOVSB 指令将数据段中的字节或字数据，传送至 ES 指向的段，字字节传送：ES:[DI]<-DS:[SI]，然后 SI<-SI+1, DI<-DI+1。

重复前缀指令 Rep 利用计数器 CX 保存串长度，可以理解为“当数据串没有结束，则继续传送”，每执行一次串指令，CX 减 1；直到 CX=0，重复执行结束。

通过本次实验更熟练的掌握了汇编程序的编写，能快速编写出一个简单的循环结构的汇编程序，进一步理解了相关指令的功能，学会了如何使用串指令来进行对于串的操作，利用串指令可以很方便的对串进行操作，体会到了串指令操作数据串的方便。

实验3 分支程序设计

3.1 实验目的

- 1.练习分支程序的编写方法。
- 2.练习汇编语言程序的上机过程。

3.2 实验原理

- 1.通过分支程序设计调试和运行，进一步熟悉掌握汇编程序执行的软件环境。
- 2.通过分支程序的执行过程，熟悉 EDIT 的使用，建立 OBJ 文件 EXE 文件的方法。

3.3 实验仪器

微机一台。

3.4 实验步骤

- 1.给出三个有符号数，编写一个比较相等关系的程序：

- (1)如果这三个数都不相等，则显示 0；
- (2)如果这三个数中有两个数相等，则显示 1；
- (3)如果这三个数都相等，则显示 2；

EDIT 状态下的源程序：

```
.model small
.stack
.data
    d1 dw ?
    d2 dw ?
    d3 dw ?
    msg db "Input: $"
    msg1 db "output: $"
.code
.startup
    mov dx, offset msg
    mov ah, 9
```

```
        int 21h
        call dispCrLf
mov ah , 1
        int 21h
mov dl, ax
        call dispCrLf
mov ah , 1
        int 21h
mov d2, ax
call dispCrLf
mov ah , 1
        int 21h
mov d3, ax
call dispCrLf
mov dl, 0
cmp ax, d2 ;比较 d3 和 d2
jz L1; d3=d2, 跳转到 L1
cmp ax, d1; 比较 d3 和 d1
jz L2; d3=d1, 跳转到 L2
mov ax, d2
cmp ax, d1; 比较 d1 和 d2
jz L2; d1=d2, 跳转到 L2
jmp outc
L1: inc dl
cmp ax, d1; 比较 d3 和 d1
jz L2; d3=d1, 跳转到 L2
jmp outc
L2: inc dl
outc:
        push dx
        mov dx, offset msg1
```



```
        mov ah, 9
        int 21h
    call dispCrLf
    pop dx
    add dl, 30h
    mov ah, 2
    int 21h
    mov ah, 4ch
        int 21h
dispCrLf proc
            push ax
            push dx
            mov dl, 0dh
            mov ah, 2
            int 21h
            mov dl, 0ah
            mov ah, 2
            int 21h
            pop dx
            pop ax
            ret
dispCrLf endp
.exit
end
```

程序执行结果截图：

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...
Z:\>mount d F:\MASMPlus\Bin
Drive D already mounted with local directory f:\MASMPlus\
Z:\>path c:\d:\;
Z:\>MOUNT C "D:\dosbox\DOSBox-0.74"
Drive C is mounted as local directory D:\dosbox\DOSBox-0.74\
Z:\>C:
C:\>TEST.EXE
Input:
2
2
3
output:
1
C:\>a_

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...
Z:\>mount d F:\MASMPlus\Bin
Drive D already mounted with local directory f:\MASMPlus\
Z:\>path c:\d:\;
Z:\>MOUNT C "D:\dosbox\DOSBox-0.74"
Drive C is mounted as local directory D:\dosbox\DOSBox-0.74\
Z:\>C:
C:\>TEST.EXE
Input:
1
2
3
output:
0
C:\>

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...
Z:\>mount d F:\MASMPlus\Bin
Drive D already mounted with local directory f:\MASMPlus\
Z:\>path c:\d:\;
Z:\>MOUNT C "D:\dosbox\DOSBox-0.74"
Drive C is mounted as local directory D:\dosbox\DOSBox-0.74\
Z:\>C:
C:\>TEST.EXE
Input:
5
5
5
output:
2
C:\>
    
```

图 3.1 比较数字是否相等程序截图

DEBUG 状态下:

```

D:\>debug
+a
073F:0100 mov ax,1
073F:0103 mov bx,0
073F:0106 mov cx,-1
073F:010B cmp ax,bx
073F:010D jne 0111
073F:010F inc dl
073F:0111 cmp ax,cx
073F:0113 jne 0117
073F:0115 inc dl
073F:0117 cmp bx,cx
073F:0119 jne 011d
073F:011B inc dl
073F:011D cmp dl,3
073F:0120 jne 0124
073F:0122 dec dl
073F:0124 add dl,30
073F:0127 mov ah,02
073F:0129 int 21
073F:012B int 20
073F:012D
+g

```

图 3.2 DEBUG 下比较数字是否相等程序截图

2. 从键盘上输入一串字符到输入缓冲区, 找出其中的大写字母和小写字母, 并分别统计它们的个数, 结果放到变量 num1 和 num2 中, 要求在屏幕输出显示所有字母及变量 num1 和 num2 的值。

EDIT 状态下的源程序:

```

.model small
.stack
.data
    buf db 255,?,255 dup(?)
    num1 db ?
    num2 db ?
    msg db "Input: $"
    msg1 db "number of capital leters: $"
    msg2 db "number of lowercase leters: $"
.code
.startup

```

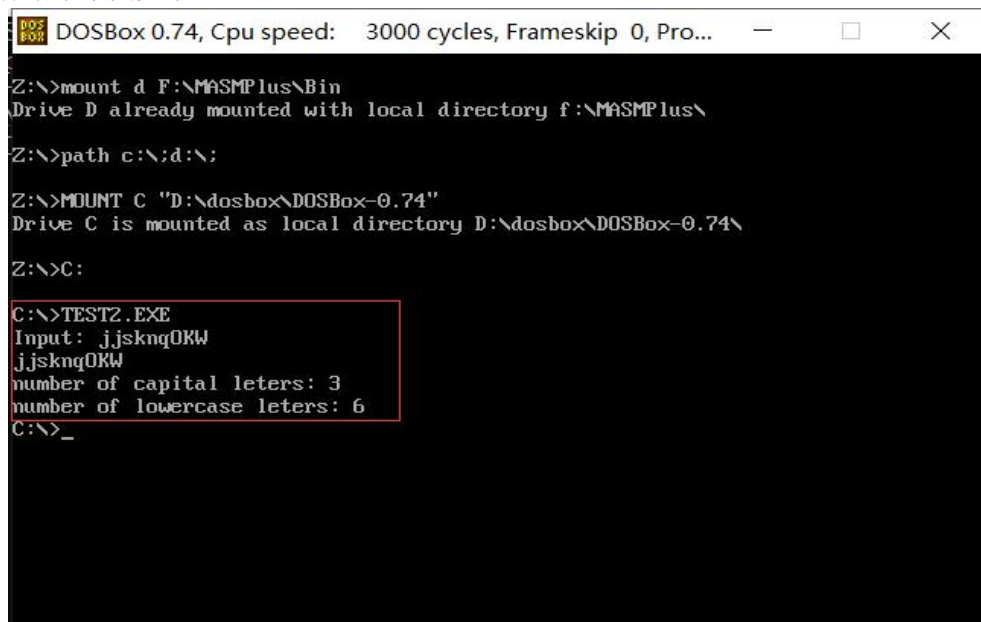
```
mov num1,0
mov num2,0
mov dx,offset msg
mov ah,9
int 21h
lea dx,buf
mov ah,0ah
int 21h
;bl 为输入的字母个数
mov bl,buf+1
cmp bl,0
jz exit
lea si,buf+2
mov cl,bl
; 比较
again: cmp byte ptr[si], 'A'
jb next
cmp byte ptr[si], 'z'
ja next
cmp byte ptr[si], 'Z'
jbe number1
cmp byte ptr[si], 'a'
jae number2
number1: inc num1
        jmp next
number2: inc num2
        jmp next
next: inc si
      loop again
mov bh,0
add bx,dx
```

```
mov byte ptr[bx+2], '$'
mov buf, 13
mov buf+1, 10
mov ah, 9
int 21h
mov dl, 0dh
mov ah, 2
int 21h
mov dl, 0ah
mov ah, 2
int 21h
mov dx, offset msg1
mov ah, 9
int 21h
mov dl, num1
add dl, 30h
mov ah, 2
int 21h
mov dl, 0dh
mov ah, 2
int 21h
mov dl, 0ah
mov ah, 2
int 21h
mov dx, offset msg2
mov ah, 9
int 21h
mov dl, num2
add dl, 30h
mov ah, 2
int 21h
```

```
exit: mov ah,4ch
      int 21h

.exit
end
```

执行结果截图如下:



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...
Z:\>mount d F:\MASMPlus\Bin
Drive D already mounted with local directory f:\MASMPlus\
Z:\>path c:\n:d\;
Z:\>MOUNT C "D:\dosbox\DOSBox-0.74\"
Drive C is mounted as local directory D:\dosbox\DOSBox-0.74\
Z:\>C:
C:\>TEST2.EXE
Input: jjsknqOKW
jjsknqOKW
number of capital letters: 3
number of lowercase letters: 6
C:\>_
```

图 3.3 统计大小写字母个数程序截图

3.5 实验要求

1. 该源程序在 DEBUG 和 EDIT 两种状态下运行。
2. 实验报告中要有源程序和执行结果。
3. 总结本次实验的体会

本次实验编写了两个汇编语言程序，一个是判断三个数字是否相等，一个是统计输入的字母中大小写字母的个数，通过编写者两个程序更好的理解了如何编写汇编语言的分支循环程序，对于汇编语言中的一些相关指令有了更加深刻的认识。

汇编语言和高级语言不一样，汇编语言没有 if 条件语句，编写分支语句只能通过无条件跳转指令 JMP 和条件跳转指令（JA, JB 等），在编写分支程序的时候就需要注意指令的执行顺序，可以通过画流程图来理解整个程序的执行，在编写汇编程序之前先画程序流程图对于汇编程序的编写有很大的帮助。

总体来说，通过本次实验能够熟练的进行分支程序的编写，掌握了条件语句对应的汇编语言指令的实现。

实验 4 统计学生成绩程序

4.1 实验目的

进一步掌握分支程序和循环程序的编写方法。

4.2 实验原理

设有 10 个学生的成绩分别为 56、69、84、82、73、88、99、63、100 和 80 分。试编制程序分别统计低于 60 分、60~69 分、70~79 分、80~89 分、90~99 分及 100 分的人数存放到 s5、s6、s7、s8、s9 及 s10 单元中。

这一题目的算法很简单，成绩分等部分采用分支结构，统计所有成绩则用循环结构完成。程序框图如下图所示。

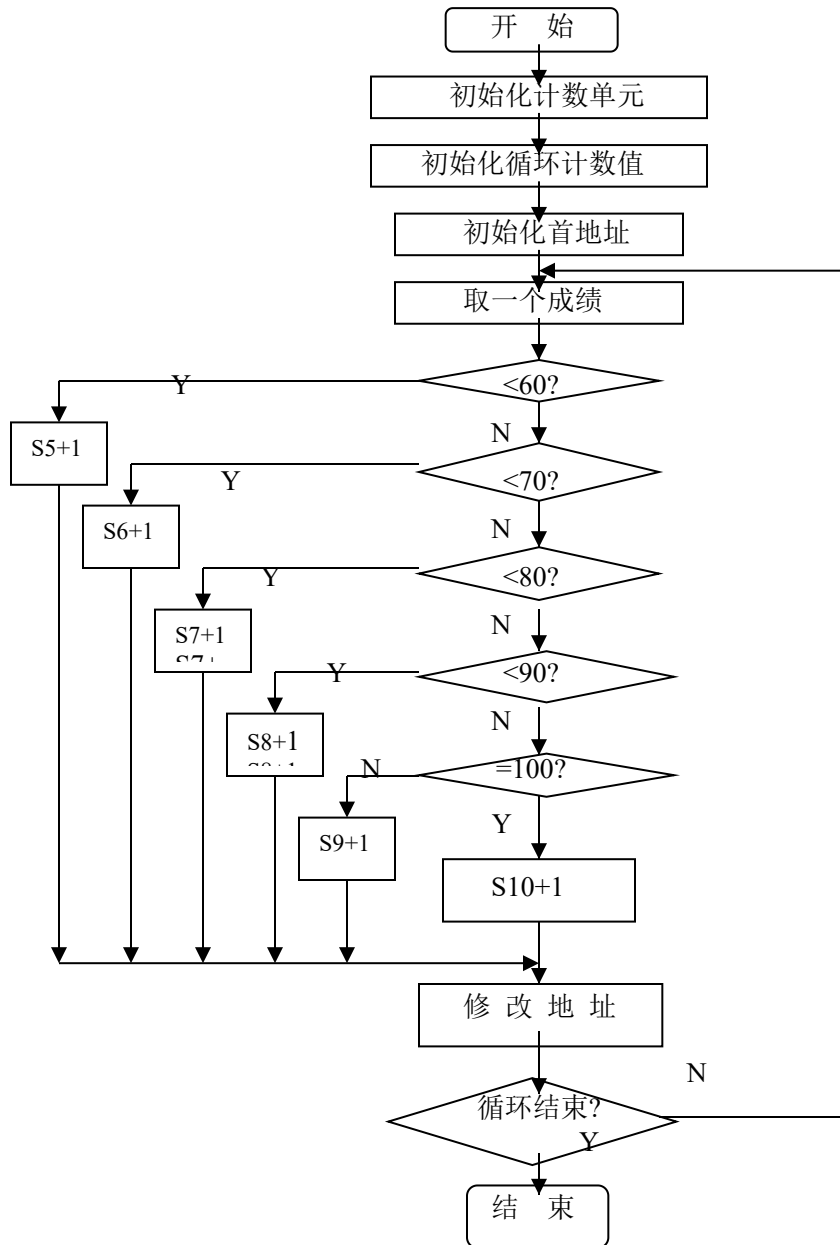
4.3 实验仪器

微机一台。

4.4 实验步骤

参考给的流程图编写源程序，进行调试。

流程图如下：



程序框图

参考程序：

```
.model small
```

```
.stack
```

```
.data
```

```
GRADE DW 56,69,84,82,73,88,99,63,100,80
```

```
S5 DW 0
```


S6 DW 0

S7 DW 0

S8 DW 0

S9 DW 0

S10 DW 0

.code

.startup

MOV S5, 0

MOV S6, 0

MOV S7, 0

MOV S8, 0

MOV S9, 0

MOV S10, 0

MOV CX, 10

MOV BX, OFFSET GRADE

COMPARE:

MOV AX,[BX] ; GET A RESULT

CMP AX, 60 ; <60?

JL FIVE

CMP AX, 70 ; <70?

JL SIX

CMP AX, 80 ; <80?

JL SEVEN

CMP AX, 90 ; <90?

JL EIGHT

CMP AX, 100 ; =100?

JNE NINE

INC S10

JMP CHANGE_ADDR

NINE:

INC S9

```
        JMP  CHANGE_ADDR  
EIGHT:  
        INC S8  
        JMP  CHANGE_ADDR  
SEVEN:  
        INC S7  
        JMP  CHANGE_ADDR  
SIX:  
        INC S6  
        JMP CHANGE_ADDR  
FIVE:  
        INC S5  
CHANGE_ADDR:  
        ADD BX, 2  
        LOOP COMPARE  
  
.exit  
end
```

4.5 实验要求

1. 读懂所给的程序。
2. 编写将存放于 S5、S6、S7、S8、S9 及 S10 单元中的数据在屏幕上显示出来的部分程序。

```
        MOV CX,6;输出  
        MOV SI,OFFSET S5  
  
AGAIN:  
        MOV DX,[SI]  
        ADD DX,30H  
        MOV AH,02H  
        INT 21H  
        MOV DL,20H
```

```
MOV AH,02H
```

```
INT 21H
```

```
ADD SI,2
```

```
LOOP AGAIN
```

3. 总结本次实验的体会

本次实验是一个编写一个统计学生成绩的汇编程序，对于该程序，首先是声明了几个变量用来存放各个成绩段的人数，接着对这些变量进行了初始化。完后通过 loop 循环对于每一个学生的成绩进行判断，判断该同学的成绩属于哪个成绩段，则对应的存放该成绩段的人数的变量加一。最后编写输出各个程序段人数的汇编程序，对于输出，也是通过循环来一个一个输出，调用 2 号功能，进行显示。

在本次实验的进行过程中，对于比较大小的跳转指令掌握不是特别熟练，通过查阅资料了解了这些条件跳转指令的具体用法和功能。

对于无符号数比较大小时用高于、低于的相关条件跳转指令：

- (1) 低于：JB/JANE
- (2) 不低于：JNB/JAE
- (3) 不高于：JBE/JAE
- (4) 高于：JNBE/JA

对于有符号数比较用大于、小于的相关条件跳转指令：

- (1) 小于：JL/JNGE
- (2) 不小于：JNL/JGE
- (3) 不大于：JLE/JNG
- (4) 大于：JNLE/JG

通过本次实验，熟练掌握了如何编写汇编语言中分支程序和循环程序，分支程序无论是在高级语言中还是在汇编语言中都是应用最广泛的，也是最基本的程序结构。在汇编语言里进行分支程序设计中，测试某些条件时经常用到的是比较指令（CMP）、条件转移指令和无条件转移指令（JMP）。

实验 5 学生成绩名次表实验

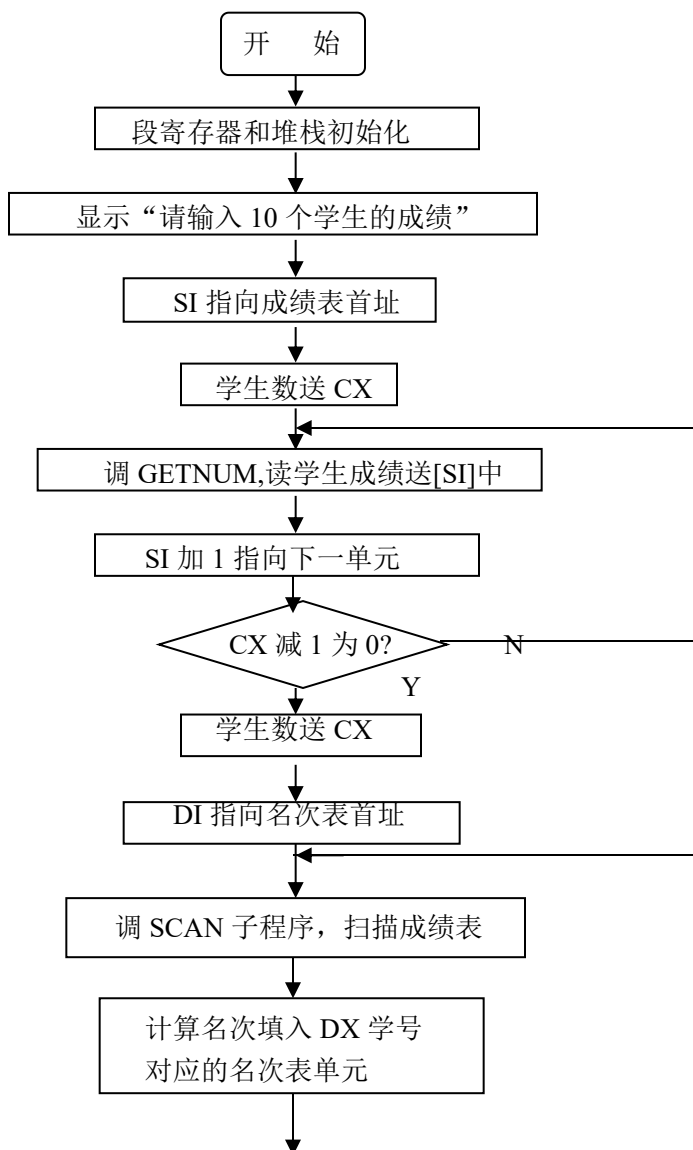
5.1 实验目的

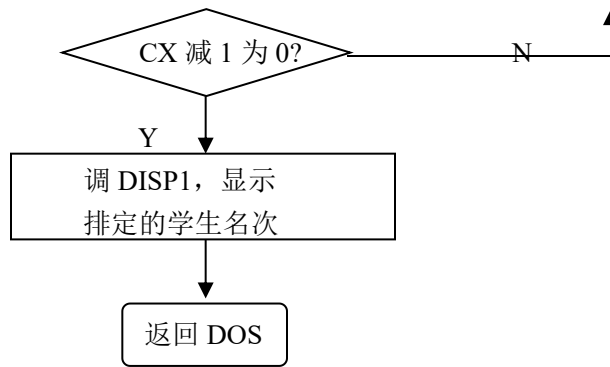
进一步熟悉排序方法。

5.2 实验原理

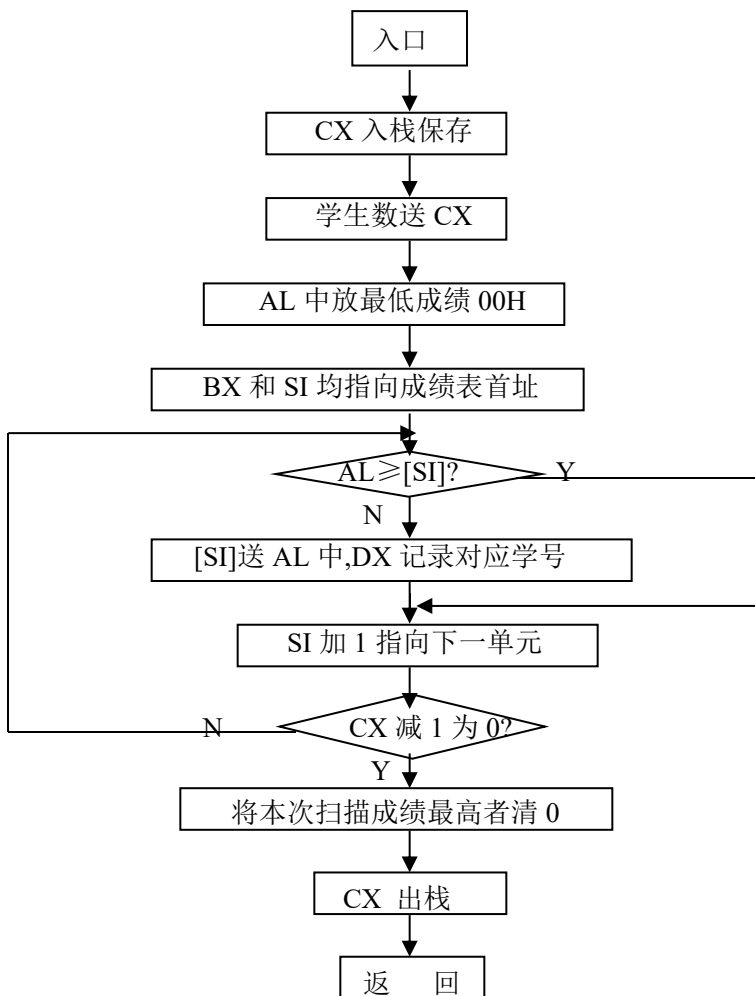
将 0~100 之间的 10 个成绩存入首址为 1000H 的单元中。1000H+i 表示学号为 i 的学生成绩,编写程序能在 2000H 开始的区域排出名次表。2000H+i 为学号 i 的学生的名次。

参考主程序:





程序 SCAN:



5.3 实验仪器

微机一台。

5.4 实验步骤

理解了给出的流程图，看懂给出的参考程序，进一步体验汇编程序中的指令的用法，学会如何汇编语言中排序程序的编写。

5.5 实验要求

- 1. 参考流程图编写实验程序，进行调试。
- 读懂参考程序，对程序中每个模块的功能加了注释，参考程序如下：

```
CRLF    MACRO
        MOV        AH, 02H
        MOV        DL, 0DH
        INT        21H
        MOV        AH, 02H
        MOV        DL, 0AH
        INT        21H
ENDM
DATA    SEGMENT
        STUNUM     EQU        10
        MESS       DB          ' INPUT 10 STUDENTS SCRE: ' ,
        ODH, 0AH, '$'
        ERROR      DB          ' INPUT ERROR! ', ODH, 0AH, '$'
        ORG        1000H
        SCORE      DB          10 DUP (?)
        ORG        2000H
        SEQU       DB          10 DUP (?)
DATA     ENDS
STACK    SEGMENT
        STA        DW          12 DUP (?)
        TOP        DW          ?
```

汇编语言程序设计实验报告

```
STACK      ENDS
CODE       SEGMENT
ASSUME     CS:CODE, DS:DATA, ES:DATA, SS:STACK
START:     MOV      AX, DATA
           MOV      DS, AX
           MOV      ES, AX
           MOV      SP, TOP          ; 初始化
           MOV      AH, 09H
           MOV      DX, OFFSET MESS
           INT      21H              ; 显示提示信息
           MOV      SI, OFFSET SCORE ; 成绩表首址
           MOV      CX, STUNUM       ; 学生数送 CX
UUU:       CALL     GETNUM            ; 读取键入数值送 DX
           MOV      [SI], DL          ; 存入成绩表缓冲区
           INC      SI                ; 指向下一单元
           LOOP     UUU
           MOV      CX, STUNUM        ; 学生数
           MOV      DI, OFFSET SEQU   ; 名次表首址
VVV:       CALL     SCAN              ; 扫描子程序
           MOV      AL, STUNUM        ; 学生数
           SUB      AL, CL
           INC      AL                ; 计算名次
           MOV      BX, DX
           MOV      [DI+BX], AL       ; 记 DX 学号对应名次
           LOOP     VVV
           MOV      CX, STUNUM        ; 学生数
           MOV      SI, OFFSET SEQU   ; 名次表首址
WWW:       MOV      AL, [SI]
           CALL     DISP1
           CRLF
           INC      SI
```

```

                LOOP    WWW                ; 显示排定的学生名次
                MOV     AX, 4C00H
                INT     21H
SCAN PROC      NEAR                ; 子程序, 每扫描一遍成绩表缓冲区, 找出其中成绩
最高者由 DX 指针指示对应学生 之后将该成绩清除, 以便下一次扫描
PUSH    CX
MOV     CX, STUNUM                ; 学生数
MOV     AL, 00H                  ; 最低成绩
MOV     BX, OFFSET SCORE
MOV     SI, BX                    ; 指向成绩表首址
CCC:    CMP     AL, [SI]
        JAE     JJJ                ; AL 中的成绩不低于成绩表指针 SI
所指单元的成绩则转 JJJ
        MOV     AL, [SI]            ; AL 存放较高的成绩
        MOV     DX, SI
        SUB     DX, BX                ; DX 为对应学号
JJJ:    INC     SI                    ; 指向下一单元
        LOOP    CCC
        ADD     BX, DX
MOV     BYTE PTR[BX], 00H        ; 本次扫描成绩最高者清 0
        POP     CX
        RET
SCAN     ENDP
; 输出两位数字
DISP1 PROC      NEAR
        PUSH    CX
        MOV     BL, AL
        MOV     DL, BL
        MOV     CL, 04
        ROL     DL, CL
        AND     DL, 0FH

```



```

        CALL    DISPL    ;输出高位
        MOV     DL, BL
        AND     DL, 0FH
        CALL    DISPL    ;输出低位
        POP     CX
        RET

DISP1    ENDP
;输出一个字符
DISPL    PROC     NEAR

        ADD     DL, 30H
        CMP     DL, 3AH
        JB      DDD      ;结果是 0-9 的数字
        ADD     DL, 27H   ;结果大于 9，输出的是小写的 a-e
DDD:     MOV     AH, 02H
        INT     21H      ;显示一个字符
        RET
DISPL    ENDP
GETNUM   PROC     NEAR

        PUSH    CX
        XOR     DX, DX
GGG:     MOV     AH, 01H
        INT     21H
        CMP     AL, 0DH
        JZ      PPP      ;没有输入，直接按回车，跳转到 PPP
        CMP     AL, 20H
        JZ      PPP      ;输入的是空格，跳转到 PPP
        SUB     AL, 30H
        JB      KKK      ;输入小于 0, 跳转到 KKK, 报错
        CMP     AL, 0AH
        JB      GETS     ;输入小于 10，输入数据合法，跳转到 GETS
        CMP     AL, 11H

```

```

        JB      KKK      ;输入在 3AH-40H, 输入数据不合法, 跳转到 KK, 报错
        SUB     AL, 07H
        CMP     AL, 0FH
        JBE     GETS     ;输入的是大写字母
        CMP     AL, 2AH
        JB      KKK      ;输入小于 a 的字符, 报错
        CMP     AL, 2FH
        JA      KKK      ;输入大于 e 的字母, 报错
        SUB     AL, 20H   ;输入的是小写字母
    
```

;移位, 输入两位数

```

GETS:    MOV     CL, 04
         SHL     DX, CL
         XOR     AH, AH
         ADD     DX, AX
         JMP     GGG
    
```

;输入信息不符合要求, 输出错误提示

```

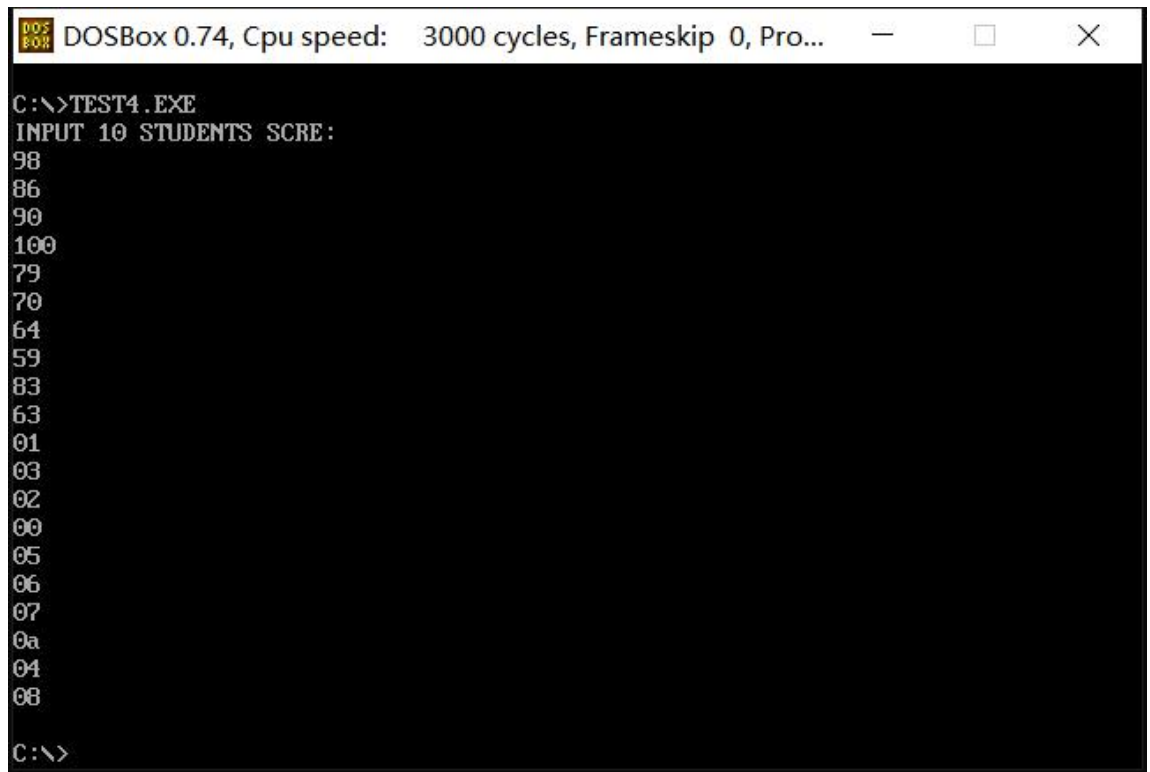
KKK:     MOV     AH, 09H
         MOV     DX, OFFSET  ERROR
         INT     21H
PPP:     PUSH    DX
         CRLF
         POP     DX
         POP     CX
         RET
    
```

```

GETNUM   ENDP
CODE     ENDS
END      START
    
```

2. 记录实验结果。

实验结果截图：



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Pro...
C:\>TEST4.EXE
INPUT 10 STUDENTS SCRE:
98
86
90
100
79
70
64
59
83
63
01
03
02
00
05
06
07
0a
04
08
C:\>
```

图 5.1 学生成绩名次表实验结果截图

3. 总结本次实验的体会

本次实验是对学生成绩表进行一个排名,通过本次实验进一步了解了汇编语言中分支程序和循环程序如何编写,并且了解了如何编写汇编语言中的排序程序。本次实验中还使用多个子程序,通过本次实验学会了如何编写子程序。

通过本次实验体会到了汇编语言对于程序的编写相对于高级语言来说还是比较复杂的,但是汇编语言的执行速度更快并且代码小,这也是汇编语言的优点。在程序需要具有较快的执行时间或者只能占用较小的内存的时候,汇编语言非常合适。

通过五次的汇编实验,能够熟练编写汇编程序,并且熟练掌握了汇编程序的指令的作用以及使用方法,能够快速准确的编写一个较为复杂的循环结构或者选择结构的汇编程序,对汇编语言有了更加深刻的理解。

封面设计： 贾丽

地 址： 中国河北省秦皇岛市河北大街 438 号

邮 编： 066004

电 话： 0335-8057068

传 真： 0335-8057068

网 址： <http://jwc.ysu.edu.cn>