華中科技大学

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 9 日**

网络空间安全学院

实验三 熟悉汇编调试环境、查看寄存器和内存内容

1.1 实验目的与内容

1.1.1 实验目的

1、了解 80x25 彩色字符模式显示缓冲区结构（教材实验 9）

2、了解 DOS 系统 INT21 功能调用（实验 ppt4）

3、了解十进制数据的显示方法（黄色教材例子 4-11）

4、熟悉程序模块及其调用，编写一个较大规模的汇编程序

1.1.2 实验内容

完成教材课程设计 1。

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 6

（1）调试课程中讲解过的程序：问题 7.8 P158 页程序调试

1）过程截图

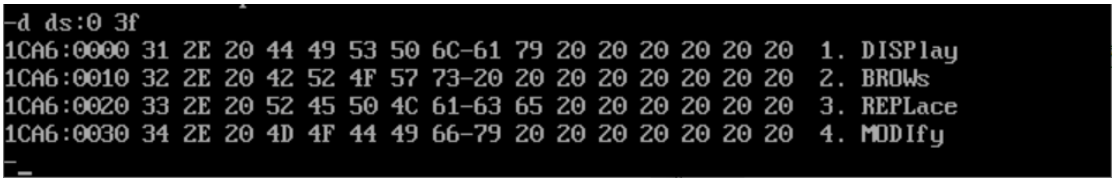
```
-u
1CAA:001E 8800            MOV     [BX+SI],AL
1CAA:0020 46              INC     SI
1CAA:0021 E2F7            LOOP    001A
1CAA:0023 83C310          ADD     BX,10h
1CAA:0026 59              POP     CX
1CAA:0027 E2EA            LOOP    0013
1CAA:0029 B44C            MOV     AH,4Ch
1CAA:002B CD21            INT     21h
1CAA:002D 0D1A80          OR      AX,801Ah
1CAA:0030 00A61810        ADD     [BP+1018],AH
1CAA:0034 0000            ADD     [BX+SI],AL
1CAA:0036 0000            ADD     [BX+SI],AL
-g cs:002b
AX=4C58  BX=0040  CX=0000  DX=0000  SP=0010  BP=0000  SI=0003  DI=0000
DS=1CA5  ES=1C95  SS=1CA9  CS=1CAA  IP=002B  NV UP DI PL NZ NA PO NC
1CAA:002B CD21            INT     21h  ;End Program
Instruction Breakpoint
-d ds:0 3f
1CA5:0000 49 42 4D 20 20 20 20 20-20 20 20 20 20 20 20 20  IBM
1CA5:0010 44 45 43 20 20 20 20 20-20 20 20 20 20 20 20 20  DEC
1CA5:0020 44 4F 53 20 20 20 20 20-20 20 20 20 20 20 20 20  DOS
1CA5:0030 56 41 58 20 20 20 20 20-20 20 20 20 20 20 20 20  VAX
-
```

2）调试过程：先使用-u 指令反汇编确定源程序的机器码、其所占空间已经程序的结束的代码地址。再使用-g 指令使程序执行到最后。最后使用-d 指令查看 ds:0 到 ds:3f 的 40h 个空间，确定字符串已经转换为大写字母。

（2）问题 7.9 编程

1）源代码：见附件

2）过程截图：

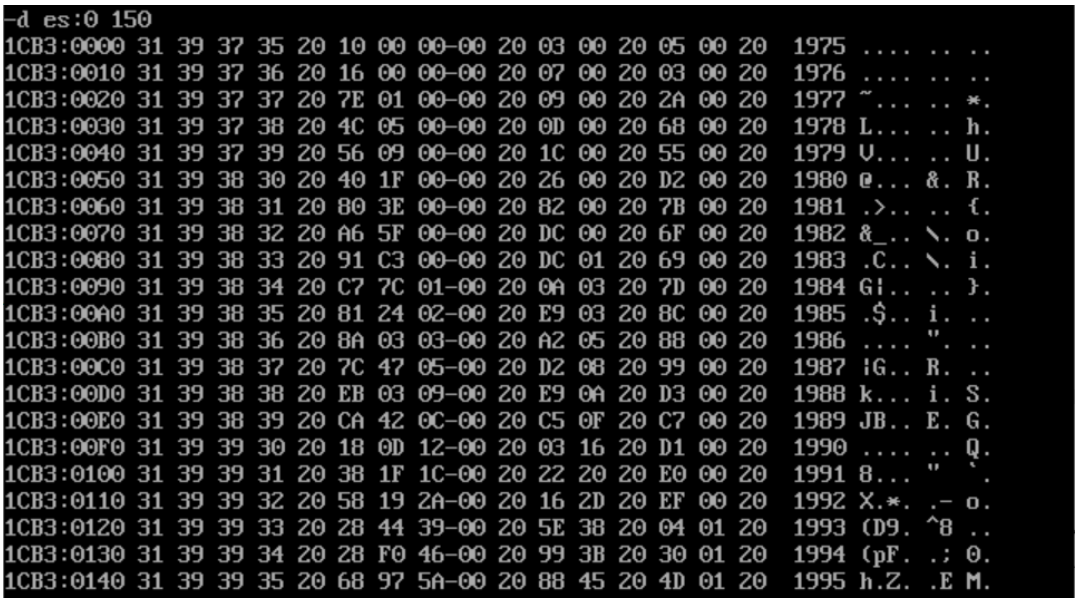


1.2.2 任务二：王爽版 实验 7

编程将 data 段数据按格式写入到 table 段

1）源代码：见附件

2）过程截图：



1.2.3 任务三：王爽版 实验 8

分析一个奇怪的程序

1）源代码：见附件（包含注释）

2）程序重要过程理解：

①mov ax,cs:[si]与 mov cs:[di],ax 两条指令将 s2 标号处的 jmp short s1 指令的机器码存入到了 s 标号处。

②jmp short s 使 IP 跳转到 s 标号处，此时 s 标号处已存有 jmp short s1，执行该条指令。

③jmp 指令实际在实际执行时是对 IP 进行偏移，在汇编过程中会将 jmp 后的标号的地址与该指令执行后的 IP 地址作差，计算出偏移地址。指令编程机器码时标号就变成了需要 IP 偏移的字节数（补码存储，jmp 到指令前为负数，jmp 到指令后为正数）。

④jmp short s1 实际上使 IP 偏移从标号 s2 到 s1 的字节数（经编译确定为-10B）。此时执行 s 标号处的 jmp short s1，IP 的值减小 10，此时 IP 恰好指

向 codesg 段开始。

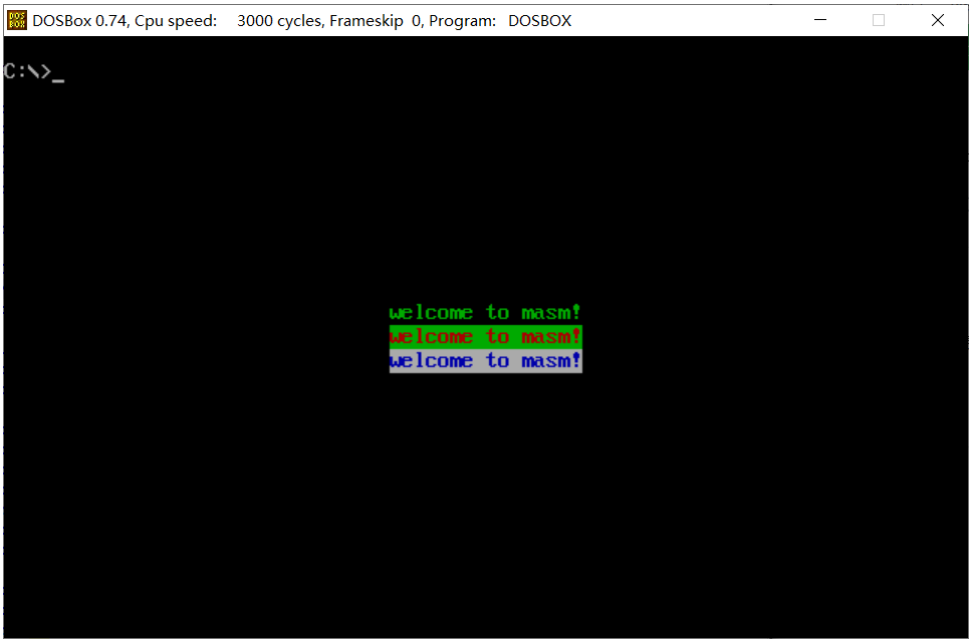
⑤最后执行 mov ax,4c00h 与 int 21h 程序结束

1.2.4 任务四：王爽版 实验 9

编程在屏幕中间分别显示绿色、绿底红色、白底蓝色的字符串‘welcome to masm!’

1）源代码：见附件

2）过程截图：

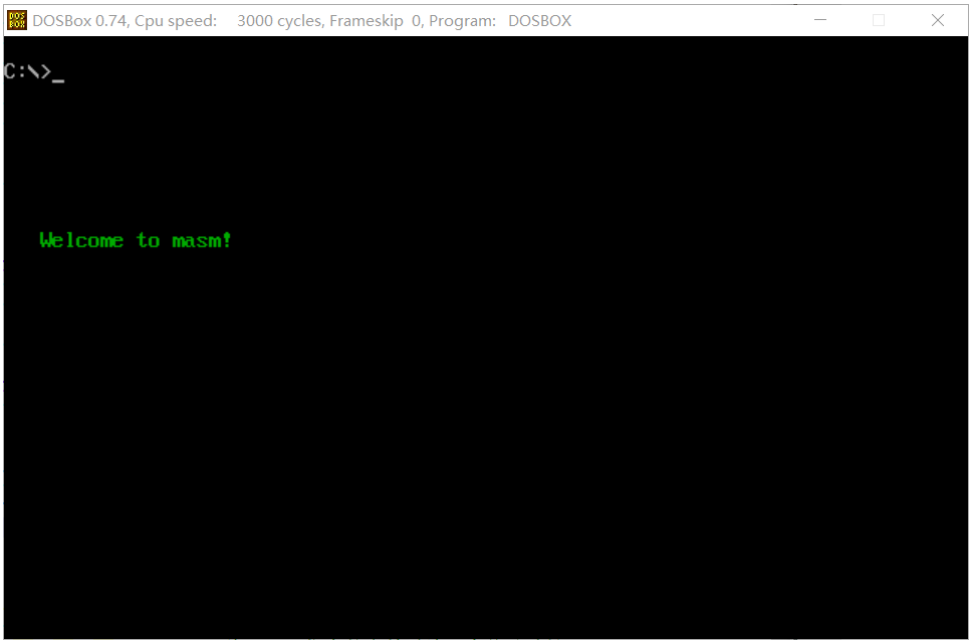


1.2.5 任务五：王爽版 实验 10

（1）显示字符串

1）源代码：见附件

2）过程截图：



（2）解决除法溢出的问题

1）源代码：见附件

2）过程截图：

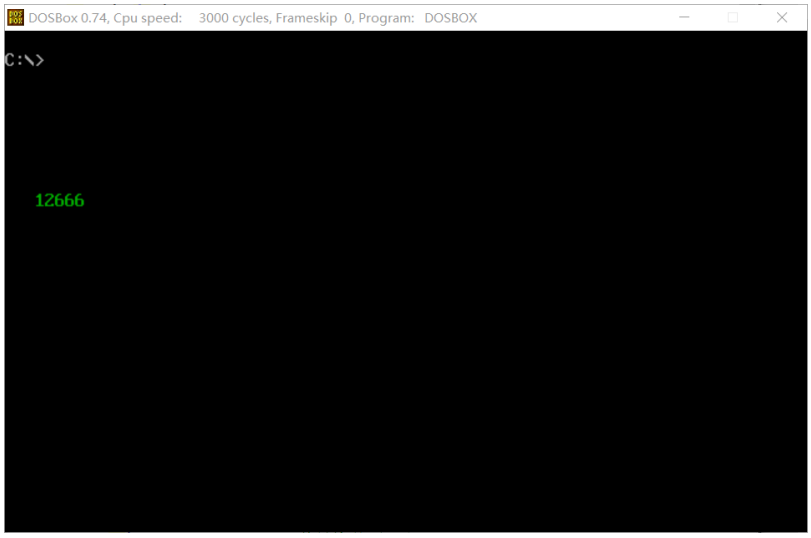
以计算 1000000/10（F4240H/0AH）为例

结果应为:(dx)=0001H,(ax)=86A0H,(cx)=0

```
-g 000c
AX=86A0  BX=0001  CX=0000  DX=0001  SP=0000  BP=0000  SI=0000  DI=0000
DS=1C95  ES=1C95  SS=1CA4  CS=1CA5  IP=000C  NV UP DI PL NZ NA PO NC
1CA5:000C B44C            MOV     AH,4Ch
Instruction Breakpoint
-_
```

（3）数值显示

1）源代码：见附件



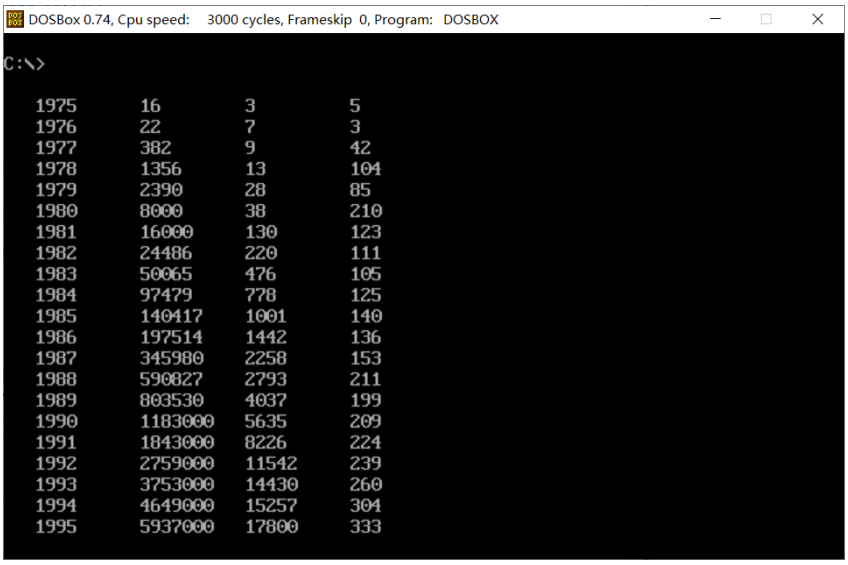
2）过程截图：

1.2.6 任务六：王爽版 课程设计 1

实验 7 的数据显示

1）源代码：见附件

2）过程截图：



1.3 实验总结

本次实验内容颇多，从较基本的寻址方式存储数据到跳转指令的使用、程序的循环结构，再到数据显示已经子过程的书写。使个人对汇编语言有了更深层次的理解，也体会到了汇编与其他语言相似于不同的地方。在汇编语言编程的过程中，深刻感受到汇编语言作为低级语言具有许多繁琐的规定要求，对程序设计有着更加苛刻的要求，同时调试也不如高级语言直接，出现问题也很难确定问题的具体出处，使调试也颇有难度。个人在做这些实验的过程中也遇到了许许多多的问题，也是通过不断地翻书查看文档、调试等方法才逐一解决。其中在做实验的过程中，个人觉得有几点比较值得注意：

1. 注意立即数传送到内存是要标明内存的数据类型，该问题一般 masm 时即可发现解决。
2. 二重循环时要注意在进入内存循环前将外层 cx 的数据入栈保护，在内层循环结束后再将 cx 原数据出栈恢复。确保循环能正常进行。
3. 跳转指令“jmp 标号”实际是 IP 的值的变化，具体见 1.2.3 实验 8 的程序分析
4. 书写子程序时也要考虑类似上文 2 的问题，子程序使用要使用的寄存器都应在子程序开始就入栈保护，在子程序程序结束前出栈恢复（除有返回值的情况），确保调用子程序后主程序中寄存器中原有的数据不会丢失。同时推广到一般情况，当一个寄存器被使用多次时，若寄存器原数据之后还会用到，一定要将该数据保护起来（一般入栈，也可以存到别的寄存器、内存中），并在需要时恢复。该问题经常会在写较长的程序时出现且较难发现。
5. 关于栈，汇编程序中栈均为字（16 位）/双字（32 位）栈，因此对于 8 位的字节数据也是以 16 位存入栈中。值得注意的是，像实验 10（3）中数值显示，字符数据会在过程中存入栈中，因而会以字类型将其出栈，这时其高字节已经是 0 了，所以实际上当处理完整个数值后，在转换的字符串末尾已经有了结尾符 0，不需额外添加，但同时在一些情况下要避免最后多出的 0，要将其覆盖掉。

总的来说，该次收获满满，对自己的汇编语言掌握有很大帮助。

1.4 尚未解决问题

无

1.5 附件

1.5.1 实验 6（2）源代码

内容：编程将 datasg 段中每个单词的前 4 个字母改为大写字母

```
assume cs:codesg,ss:stacksg,ds:datasg

stacksg segment
    dw 0,0,0,0,0,0,0,0
stacksg ends

datasg segment
    db '1. display          '
```

```
        db '2. brows              '
        db '3. replace          '
        db '4. modify             '
datasg ends

codesg segment

start:
            mov ax,datasg
            mov ds,ax
            mov cx,4        ;循环 4 次处理字符串
            mov bx,0        ;bx 存 datasg 段行偏移地址

        s0:
            mov si,3        ;si 存 datasg 段列偏移地址
            push cx                 ;cx 数据入栈保护
            mov cx,4        ;循环 4 次处理前四个字母

        s:
            sub byte ptr [bx+si],20h   ;将小写字母改为大写
            inc si                  ;si 指向下一字符
            loop s

            pop cx                  ;原 cx 数据出栈恢复
            add bx,10h              ;bx 指向下一行字符串
            loop s0

            mov ah,4ch
            int 21h

codesg ends

end start
```

1.5.2 实验 7 源代码

内容：编程将 data 段数据按格式写入到 table 段

```
.386
assume cs:codesg,ds:data,es:table

data segment use16
        db '1975','1976','1977','1978','1979','1980','1981','1982','1983'
        db '1984','1985','1986','1987','1988','1989','1990','1991','1992'
        db '1993','1994','1995'
        ;以上是表示 21 年的 21 个字符串

        dd 16,22,382,1356,2390,8000,16000,24486,50065,97479,140417,197514
```

```
        dd 345980,590827,803530,1183000,1843000,2759000,3753000,4649000,5937000
        ;以上是表示 21 年公司总收入的 21 个 DWORD 型数据

        dw 3,7,9,13,28,38,130,220,476,778,1001,1442,2258,2793,4037,5635,8226
        dw 11542,14430,15257,17800
        ;以上是表示 21 年公司雇员人数的 21 个 word 型数据
data ends

table segment use16
        db 21 dup('year summ ne ?? ')
table ends

codesg segment       use16

start:
                mov ax,data
                mov ds,ax
                mov ax,table
                mov es,ax
                mov bx,0              ;bx 用来指向表示 21 年的字符串和 21 年总收入
                mov di,168                  ;di 指向雇员人数
                mov si,0              ;si 指向 table 段

                mov cx,21

        s:
                mov eax,[bx]
                mov es:[si],eax       ;year

                mov ax,[bx+84]
                mov es:[si+5],ax
                mov ax,[bx+86]
                mov es:[si+7],ax      ;收入

                mov ax,[di]
                mov es:[si+10],ax           ;雇员数

                mov ax,[bx+84]
                mov dx,[bx+86]
                div word ptr es:[si+10]
                mov es:[si+13],ax           ;人均收入

                add bx,4
                add di,2
                add si,10h
                loop s
```

```
        mov ah,4ch
        int 21h

codesg ends

end start
```

1.5.3 实验 8 源代码

内容：一个奇怪的程序

```
assume cs:codesg

codesg segment

        mov ax,4c00h
        int 21h

start:
        mov ax,0
    s:
        nop                     ;跳转后，s 段存有 jmp short s1 指令机器码
        nop                     ;

        mov di,offset s     ;把 s 的偏移地址给 di
        mov si,offset s2    ;把 s2 的偏移地址给 si
        mov ax,cs:[si]      ;把 cs：s2 的内容（jmp short s1）给 ax
        mov cs:[di],ax      ;把 cs：s2 的内容给 cs：di，即给 s 标号段
                                ;此时 s 段内存的为 s2 的第一条指令 jmp short s1

    s0:
        jmp short s             ;跳转到 s 标号段

    s1:
        mov ax,0
        int 21h
        mov ax,0

    s2:
        jmp short s1        ;跳转到 s1 段（实际为跳转到相对该指令结束的偏移地址处）
                                ;该指令距离 s1 有-10B，所以执行该语句时 ip 倒退 10B
                                ;在 s 段中时，执行该语句后 ip 倒退 10K 跳转到 mov ax,4c0h 处

        nop

codesg ends

end start
```

1.5.4 实验 9 源代码

内容：编程在屏幕中间分别显示绿色、绿底红色、白底蓝色的字符串‘welcome to masm!’

```
data segment
    db 'welcome to masm!'    ;输出字符串
    db 00000010b,00100100b,01110001b    ;三种显示样式的属性
data ends

code segment
    assume cs:code,ds:data

start:
        mov ax,data
        mov ds,ax
        mov ax,0b800h
        mov es,ax                ;将显存段地址 b800 赋给 es
        mov si,11*160+80-16      ;si 为显存偏移地址
                                 ;11*160 表示第 12 行起始处，80-16 到达该行中间
        mov di,16                ;di 为三种显示样式偏移地址

        mov ax,0003h
        int 10h                  ;清屏
        mov cx,3                 ;循环三次输出字符串

    s0:
        push cx                  ;将先前 cx 入栈保护
        mov cx,16                ;循环输出 16 个字符
        mov bx,0                 ;bx 为输出字符串偏移地址

    s1:
        mov al,[bx]
        mov es:[si],al           ;将字符串的每个字符传送到显存中
        mov al,[di]
        mov byte ptr es:[si+1],al ;将字符对应的属性存入显存中
        inc bx                   ;bx 向下一个输出字符偏移
        add si,2                 ;si 指向下一个字
        loop s1

        add si,160-16*2          ;si 指向下一行中间
        pop cx                   ;原 cx 值出栈
        inc di                   ;di 指向下一种样式的属性
        loop s0

        mov ah,4ch
```

```
        int 21h

code ends

end start
```

1.5.5 实验 10（1）源代码

内容：显示字符串

```
;显示字符串
;名称：show_str
;功能：在指定的位置，用指定的颜色，显示一个用 0 结束的字符串
;参数：(dh)=行号（取值范围 0~24），(dl)=列号（取值范围 0~79）
;       (cl)=颜色，ds:si 指向字符串首地址
;返回：无

assume cs:code,ds:data
data segment
    db 'Welcome to masm!',0
data ends

code segment

start:
        mov ax,data
        mov ds,ax

        mov ax,0003h
        int 10h                         ;清屏

        mov si,0
        mov dh,8
        mov dl,3
        mov cl,2
        call show_str

        mov ah,4ch
        int 21h

show_str:
        push ax
        push es
        push bx
        push di
        push si                 ;主程序寄存器数据入栈保护

        mov ax,0b800h
        mov es,ax               ;将 es 段地址赋值为显存的 b800h
```

```
        mov al,2
        mul dl
        push ax
        mov al,160
        mul dh
        pop bx
        add ax,bx
        mov di,ax                ;di=ah*160+al*2

    change:
        cmp byte ptr [si],0      ;比较 ds:si 指向的数据与 0
        jz ok                    ;若 ds：si 为 0 则跳转到 ok

        mov al,[si]              ;di 指向显示的指定位置对应的显存偏移地址
        mov es:[di],al           ;将 ds：si 指向的字符赋给显存偏移地址
        mov es:[di+1],cl         ;将 cl 内的显示属性赋给显存偏移地址

        inc si                   ;si 指向下一字符
        add di,2                 ;di 指向下一字
        jmp short change         ;跳转到子过程开始
    ok:
        pop si
        pop di
        pop bx
        pop es
        pop ax                   ;主程序寄存器数据出栈恢复
        ret                      ;子过程结束，返回

code ends

end start
```

1.5.6 实验 10（2）源代码

内容：解决除法溢出的问题

```
;解决除法溢出问题
;名称：divdw
;功能：进行不会产生溢出的除法运算，被除数为 dword 型，除数为 word 型，结果为 dword 型
;参数：(ax)=dword 型数据的低 16 位，(dx)=dword 型数据的高 16 位，(cx)=除数
;返回：(dx)=结果的高 16 位，(ax)=结果的低 16 位，(cx)=余数

assume cs:code

code segment

start:
```

```
        mov ax,4240h
        mov dx,00fh
        mov cx,0ah
        call divdw

        mov ah,4ch
        int 21h

divdw:
        push bx             ;主程序寄存器数据入栈保护

        push ax             ;将 ax 中的被除数低 16 位入栈保护
        mov ax,dx           ;将被除数高 16 位放入 ax  为除法做准备
        mov dx,0            ;dx 清零  为字除法做准备

        div cx              ;被除数高 16 位/除数
        mov bx,ax           ;将 int(H/N)存入 bx 中

        pop ax              ;将原 ax 出栈得被除数低 16 位
        div cx              ;dx 中为 rem(H/N)，作为高位及相当于*65536
                            ;此时(ax)=int( [rem(H/N)*65536+L]/N )，即整体商低 16 位
                            ;(dx)=rem(       [rem(H/N)*65536+L]/N )

        mov cx,dx           ;(dx)=rem(       [rem(H/N)*65536+L]/N ),即整体的余数
        mov dx,bx           ;int(H/N)*65536，即整体商高 16 位

        pop bx              ;主程序寄存器数据出栈恢复
        ret

code ends

end start
```

1.5.7 实验 10（3）源代码

内容：数值显示

```
;word 型转为十进制字符串
;名称：dtoc
;功能：将 word 型数据转变为表示十进制数的字符串，字符串以 0 为结尾符
;参数：(ax)=word 型数据，ds:si 指向字符串首地址
;返回：无

assume cs:code,ds:data

data segment
    db 10 dup(0)
```

```
data ends

code segment

start:
        mov bx,data
        mov ds,bx

        mov si,0
        mov ax,12666
        call dtoc

        mov ax,0003h
        int 10h                 ;清屏
        mov dh,8
        mov dl,3
        mov cl,2
        call show_str

        mov ah,4ch
        int 21h

dtoc:
        push ax
        push bx
        push cx
        push dx
        push si                 ;主程序寄存器数据入栈保护

        mov bx,10               ;10 做除数存到 bx 中
        mov cx,0                ;cx 做计数器记位  置零
    getc:
        mov dx,0                ;将 dx 清零  为除法做准备
        div bx                  ;(dx,ax)除以 bx
        add dx,30h              ;余数 dx 为对应十进制数，+30h 转换为 ASCII 码
        push dx                 ;将余数入栈存储
        inc cx                  ;位数+1

        cmp ax,0                ;若商为 0
        jz putc                 ;跳转到 putc

        jmp getc                ;商不为 0 跳转到 getc
    putc:
        pop [si]                ;将栈中字符出栈存入[si]指向的内存
        inc si                  ;si 指向下一字符
        loop putc
```

```
        pop si
        pop dx
        pop cx
        pop bx
        pop ax                  ;主程序寄存器数据出栈恢复
        ret

show_str:
        push ax
        push es
        push bx
        push di
        push si                 ;主程序寄存器数据入栈保护

        mov ax,0b800h
        mov es,ax               ;将 es 段地址赋值为显存的 b800h

        mov al,2
        mul dl
        push ax
        mov al,160
        mul dh
        pop bx
        add ax,bx
        mov di,ax               ;di=ah*160+al*2
    change:
        cmp byte ptr [si],0     ;比较 ds:si 指向的数据与 0
        jz ok                   ;若 ds：si 为 0 则跳转到 ok

        mov al,[si]             ;di 指向显示的指定位置对应的显存偏移地址
        mov es:[di],al          ;将 ds：si 指向的字符赋给显存偏移地址
        mov es:[di+1],cl        ;将 cl 内的显示属性赋给显存偏移地址

        inc si                  ;si 指向下一字符
        add di,2                ;di 指向下一字
        jmp short change        ;跳转到子过程开始
    ok:
        pop si
        pop di
        pop bx
        pop es
        pop ax                  ;主程序寄存器数据出栈恢复
        ret                     ;子过程结束，返回

code ends
```

```
end start
```

1.5.8 课程设计 1 源代码

内容：实验 7 数据在屏幕显示

```
.386
assume cs:codesg,ds:table

data segment use16
    db '1975','1976','1977','1978','1979','1980','1981','1982','1983'
    db '1984','1985','1986','1987','1988','1989','1990','1991','1992'
    db '1993','1994','1995'
    ;以上是表示 21 年的 21 个字符串

    dd 16,22,382,1356,2390,8000,16000,24486,50065,97479,140417,197514
    dd 345980,590827,803530,1183000,1843000,2759000,3753000,4649000,5937000
    ;以上是表示 21 年公司总收入的 21 个 DWORD 型数据

    dw 3,7,9,13,28,38,130,220,476,778,1001,1442,2258,2793,4037,5635,8226
    dw 11542,14430,15257,17800
    ;以上是表示 21 年公司雇员人数的 21 个 word 型数据
data ends

table segment use16
    db    840 dup(' ')
    ;共 21 行，每行 40 个空格字符，每 10 个字符存一个数据
table ends

codesg segment       use16

start:
        mov ax,data
        mov es,ax
        mov ax,table
        mov ds,ax
        mov bx,0                ;bx 用来指向表示 21 年的字符串和 21 年总收入
        mov di,168              ;di 指向雇员人数
        mov si,0                ;si 指向 table 段

        mov cx,21

    s0:
        mov eax,es:[bx]
        mov [si],eax            ;year

        mov ax,es:[bx+84]
        mov dx,es:[bx+86]
        add si,10
```

```
        call dtoc                   ;收入

        mov ax,es:[di]
        mov dx,0
        add si,10
        call dtoc                   ;雇员数

        mov ax,es:[bx+84]
        mov dx,es:[bx+86]
        div word ptr es:[di]
        mov dx,0
        add si,10
        call dtoc                   ;人均收入

        mov byte ptr [si+3],0       ;结尾赋值为 0 结束符

        add bx,4
        add di,2
        add si,10
        loop s0

        mov ax,0003h
        int 10h                     ;清屏

        mov cx,21                   ;设置 21 次循环
        mov si,0                    ;设置显示的字符串起始偏移地址
        mov dh,3                    ;设置显示行
        mov dl,3                    ;设置显示列
    s1:
        push cx                     ;将 cx 里的循环次数入栈保护
        mov cl,00000111b            ;设置显示属性（黑底白字）
        call show_str
        inc dh                      ;显示行数+1
        add si,40                   ;指向字符串偏移 40B
        pop cx                      ;将栈中循环次数出栈恢复
        loop s1

        mov ah,4ch
        int 21h

dtoc:
    ;功能：将 dword 型转变为十进制数的字符串，字符串以 0 为结尾符
    ;参数：(ax)=dword 型数据的低 16 位，(dx)=dword 型数据的高 16 位，ds：si 指向字符串的
;首地址
    ;返回:无
        push ax
```

```
        push bx
        push cx
        push dx
        push si                 ;主程序寄存器数据入栈保护

        mov bx,0                ;bx 做计数器记位 置零
    getc:
        mov cx,10               ;10 做除数存到 cx 中
        call divdw              ;(dx,ax)除以 cx
        add cx,30h              ;余数 cx 为对应十进制数，+30h 转换为 ASCII 码
        push cx                 ;将余数入栈存储
        inc bx                  ;位数+1

        cmp ax,0                ;若商的低 16 位不为 0
        jnz getc                ;跳转到跳转 getc 继续计算
        cmp dx,0                ;若商的高 16 位不为 0
        jnz getc                ;跳转到跳转 getc 继续计算
                                ;若商的 32 位均为 0 即商为 0 则执行 putc
    putc:
        pop [si]                ;将栈中字符出栈存入[si]指向的内存
        inc si                  ;si 指向下一字符
        sub bx,1
        jnz putc                ;还有位数则继续循环输出字符

        mov byte ptr [si],20h   ;添加字符串结束符 0

        pop si
        pop dx
        pop cx
        pop bx
        pop ax                  ;主程序寄存器数据出栈恢复
        ret

divdw:
        push bx                 ;主程序寄存器数据入栈保护

        push ax                 ;将 ax 中的被除数低 16 位入栈保护
        mov ax,dx               ;将被除数高 16 位放入 ax 为除法做准备
        mov dx,0                ;dx 清零 为字除法做准备

        div cx                  ;被除数高 16 位/除数
        mov bx,ax               ;将 int(H/N)存入 bx 中

        pop ax                  ;将原 ax 出栈得被除数低 16 位
        div cx                  ;dx 中为 rem(H/N)，作为高位及相当于*65536
```

```
                                ;此时(ax)=int( [rem(H/N)*65536+L]/N )，即整体商低 16 位
                                ;(dx)=rem(      [rem(H/N)*65536+L]/N )

        mov cx,dx               ;(dx)=rem(      [rem(H/N)*65536+L]/N ),即整体的余数
        mov dx,bx               ;int(H/N)*65536，即整体商高 16 位

        pop bx                  ;主程序寄存器数据出栈恢复
        ret

show_str:
        push ax
        push es
        push bx
        push di
        push si                 ;主程序寄存器数据入栈保护

        mov ax,0b800h
        mov es,ax               ;将 es 段地址赋值为显存的 b800h

        mov al,2
        mul dl
        push ax
        mov al,160
        mul dh
        pop bx
        add ax,bx
        mov di,ax               ;di=ah*160+al*2

    change:
        cmp byte ptr [si],0     ;比较 ds:si 指向的数据与 0
        jz ok                   ;若 ds：si 为 0 则跳转到 ok

        mov al,[si]             ;di 指向显示的指定位置对应的显存偏移地址
        mov es:[di],al          ;将 ds：si 指向的字符赋给显存偏移地址
        mov es:[di+1],cl        ;将 cl 内的显示属性赋给显存偏移地址

        inc si                  ;si 指向下一字符
        add di,2                ;di 指向下一字
        jmp short change        ;跳转到子过程开始
    ok:
        pop si
        pop di
        pop bx
        pop es
        pop ax                  ;主程序寄存器数据出栈恢复
        ret                     ;子过程结束，返回
```

```
codesg ends

end start
```