汇编语言（第三版）王爽　　检测点答案

第一章

检测点 1.1

(1) 13

(2) 1024, 0, 1023

(3) 8192, 1024

(4) 2^30, 2^20, 2^10

(5) 64, 1, 16, 4

(6) 1, 1, 2, 2, 4

(7) 512, 256

(8) 二进制

注:符号'^'指求幂运算(如: 2^30 指 2 的 30 次方)

第二章

检测点 2.1

(1)大家做这题的时候一定要注意,要看清楚数制,比如是二进制还是十进制,还是十六进,我当时没注意,做错了!!呵呵!!

第一空: F4A3H

第二空: 31A3H

第三空: 3123H

第四空: 6246H

第五空: 826CH

第六空: 6246H

第七空: 826CH

第八空: 04D8H

第九空: 0482H

第十空: 6C82H

第十一空: D882H

第十二空: D888H

第十三空: D810H

第十四空: 6246H

(2)

```
mov ax,2
add ax,ax
add ax,ax
add ax,ax
```

检测点 2.2

(1) 00010H 1000fH

(2) 0001H 2000H

检测点 2.3

共修改了4次ip 第一次mov ax,bx 第二次是sub ax,ax 第三次是jmp ax 第四次是执行 jmp ax 最后 ip 的值是 0 因为 ax 的值是零!!

检测点 3.1

(1)

第一空：2662H
第二空：E626H
第三空：E626H
第四空：2662H
第五空：D6E6H
第六空：FD48H
第七空：2C14H
第八空：0000H
第九空：00E6H
第十空：0000H
第十一空：0026H
第十二空：000CH
注意：ax 中存取的是字型数据 ， 高地址存放高字节，低地址存放低字节!!（一定要小心）
（2）
①写出 cpu 的执行序列
Mov ax,6622
Jmp 0ff0:0100
Mov ax,2000H
Mov ds,ax
Mov ax ,[0008]
Mov ax,[0002]
②
指令执行顺序
寄存器
CS
IP
DS
AX
BX
初值
2000H
0000
1000H
0
0

mov ax,6622h
2000H
0003
1000H
6622H
0000

jmp 0ff0:0100

1000h
0000
1000H
6622H
0000

mov ax,2000h
1000H
0003
1000H
2000H
0000

mov ds,ax
1000H
0005
2000H
2000H
0000

mov ax,[0008]
1000H
0008
2000H
C389H
0000

mov ax,[0002]
1000H
000B
2000H
EA66H
0000

③数据和程序在计算机中都是以二进制的形式存放的，在区别程序和数据时，关键是看段地址，如果段地址是 ds 段，说明该内存中存放的是数据，如果段地址是 cs 段，说明该内存中存放的是程序指令

检测点 3.2

(1)

Mov ax,2000H

Mov ss,ax

Mov sp,0010H

(2)

Mov ax,1000H

```
Mov  ss,ax
Mov  sp,0000H
```

注意：出栈时和入栈时的顺序，空栈时栈顶指向的是最高端地址！栈满是栈顶指针指向的是最底端地址!!

实验二：用机器指令和汇编指令编程

(1)

```
5BEA
5CCA
30F0
6023
```

```
00FE    2200:0100    5CCA
00FC    2200:00FE    6023
00FE    6023
0100    5CCA
00FE    2200:0100    30F0
00FC    22000:00FC   2E39
```

(2)

因为在 DEBUG 中 T 命令在执行修改寄存器 ss 的指令时，会产生中断，将现场保护起来，下一条指令也紧接着被执行!!

实验三：

该实验自己做吧 我做出来的东西只是我 pc 上的数据，可能在别的 pc 做出来的结果会不一样，在加之数据太多，我做了两次数据都没有完整的记下来，只好作罢!!

实验四： [bx]和 loop 的使用

(1)

```
assume cs:code
code segment
    mov ax,0
    mov ds,ax
    mov cx,64
    mov bx,200h
  s:
    mov ds:[bx],ax
    inc bx
    inc ax
loop s
    mov ax,4c00H
    int 21h
code ends
End
```

(2) 如上题 我一不小心写出来就是九条指令了!!! 呵呵!!

(3) 具体做法是将那段指令写道内存中，察看其在内存中的起始地址和终止地址，然后将该内存的内容复制到 0：200 处就可以了!!

检测点 6.1

(1)

```
Mov cs:[dx],ax
```

(2)

第一空：cs

第二空：26 或者 1ah

第三空：pop cs：[dx]

实验五：编写、调试具有多个段的程序

(1)

① 1931（一直保持不变，由于 pc 不同 答案有可能不一样）

②cs=1943　　ss=1941　　ds=1931　　(由于 pc 不同，所以答案也可能不同，这只是机子上的数据)

③第一空：X—2

　第二空：X—1

(2)

1. 一直不变

2. 答案不一样，就不写了

3. X-2，X-1

4. (N/16+1)*16

(3)

1. 一直不变

2. 答案不一样

3. X+3，X+4

(4)

答：可能执行，如果不指明标号，cpu 顺序执行指令，所有可能正确执行!!!!

(5)

```
;*****************************************
;将 a 段和 b 段指的数据依次相加，将结果保存到 c 段中
;*****************************************
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
d segment
  db 8 dup(0)
d ends
;*************************************
;代码段
```

```
;*********************************
code segment
start:
   mov ax,d
   mov es,ax
   mov ax,a
   mov ds,ax
   mov cx,8
   mov ax,0
   mov bx,0
again1:
   mov dx,ds:[bx]
   mov es:[ax],dx
   inc bx
   inc ax
loop again1
   mov ax,b
   mov ds,ax
   mov ax,0
   mov bx,0
   mov cx,8
   mov dx,0
again2:
   add dx,ds:[bx]
   mov es:[ax],dx
   inc ax
   inc bx
loop again2
   mov ax,4c00h
   int 21h
code ends
end start
```

(6)

```
;*************************************************************
;编写程序,用 push 指令将 a 段中的 word 数据,逆序存储到 b 段中
;*************************************************************
assume cs:code
a segment
  dw 1,2,3,4,5,6,7,8
a ends
b segment
  dw 0,0,0,0,0,0,0,0
b ends
code segment
```

```
start:
      mov ax,b
      mov ss,ax
      mov sp,10h
      mov ax,a
      mov ds,ax
      mov bx,0
      mov cx,8
s:    push [bx]
      add bx,2
      loop s
      mov ax,4c00h
      int 21h
code ends
end start
```

本文来自 CSDN 博客，转载请标明出处：ht (1)

本题略!!!

(2)

```
;*************************************************************
;编程,将 datasg 段中的每个单词的前四个字母改写为大写字母
;*************************************************************
assume cs:codesg,ss:stacksg,ds:datasg
stacksg segment
  dw 0,0,0,0,0,0,0,0
stacksg ends
datasg segment
  db '1. displsy    '
  db '2. brows      '
  db '3. replace    '
  db '4. modify     '
datasg segment
codesg segment
start:
      mov ax,stacksg
      mov ss,ax
      mov sp,16
      mov ax,datasg
      mov ds,ax
      mov dx,0
      mov cx,4
  s0:
    push cx
    mov si,0
    mov cx,4
```

```
  s1:
    mov al,[bx+si+4]
    add al,11011111B
    mov [bx+si+4],al
    inc si
  loop s1
    add dx,16
    pop cx
  loop s0
    mov ax,4c00h
    int 21h
codesg ends
    end start
assume cs:codesg,ds:data,es:table
 data segment
        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 11542,14430,45257,17800
        ;以上是表示 21 年公司雇员人数的 21 个 word 型数据
data ends

table segment
        db 21 dup('year summ ne ?? ')
table ends
codesg segment
start:
      mov ax,data
      mov ds,ax
      mov ax,table
      mov es,ax
      mov cx,21
      mov bx,0
      mov si,0
      mov di,0

  s0:
```

```
;*****************************************
;把年份送到 table 中
;*****************************************
      mov al,[bx]
      mov es:[di],al
      mov al,[bx+1]
      mov es:[di+1],al
      mov al,[bx+2]
      mov es:[di+2],al
      mov al,[bx+3]
      mov es:[di+3],al
;*********************************************
;把收入送到 table 中
;*********************************************
      mov ax,54h[bx]
      mov dx,56h[bx]
      mov es:5h[di],ax
      mov es:7h[di],dx
;*****************************************
;把人数送到 table 中
;*****************************************
      mov ax,0A8h[si]
      mov es:0Ah[di],ax
;*****************************************
;计算人均收入并把其送到 table 中
;*********************************************
      mov ax,54h[bx]
      div word ptr 0A8h[si]
      mov es:0Ch[di],ax

      add si,2
      add di,16
      add dx,4
   loop s0                                  ;循环 21 次
     mov ax,4c00h
     int 21h
codesg ends
end start
```

第九章

检测点 9.1

(1) 程序如下:

```
    assume cs:code
```

```
data segment
  db 0,0,0,0,0,0,0,0
data ends
code segment
start:mov ax,data
      mov ds,ax
      mov bx,0
      jmp word ptr [bx+1]
      mov ax,4c00h
      int 21h
code ends
ends start
```

理由是:要是 jmp 跳转后执行第一条指令,本条指令是 word ptr 是段内转移 必须满足 ip=0 所以 ds: [bx+1]的值必须为零 ， 也就是 data 段的第二个数据必须为零

(2) 第一空: bx

第二空: cs

(3) 本题可以先用 debug 将 内存 2000: 1000 中的内容写为 BE 00 06 00 .....然后再调试可得到

cs=0006h ip=00BEh

检测点 9.2

```
;*********************************************
;实现在内存 2000H 段中查找第一个值为零的字节,
;找到后,将它的偏移地址存储到 dx 中
;*********************************************
assume cs:code
code segment
start:
    mov ax,2000H
    mov ds,ax
    mov bx,0
  s:
    mov cl,[bx]
    mov ch,0
    jcxz ok
    inc bx
  jmp short s
  ok:
    mov dx,bx
    mov ax,4c00h
    int 21h
code ends
```

```
      end start
```

注：利用 jcxz 判断 cx 是否等于零来发生跳转，注意判断的是一个字节!!

检测点 9.3

```
;*************************************************************
;利用 loop 指令,实现在内存 2000H 段中查找第一个值为零的 byte,找到后
;将它的偏移地址存储到 dx 中
;*************************************************************
assume cs:code
code segment
 start:
    mov ax,2000H
    mov ds,ax
    mov bx,0
  s:
    mov cl,[bx]
    mov ch,0
    jmp ok
    inc bx
    loop s
  ok:
    dec bx
    mov dx,bx
    mov ax,4c00h
    int 21h
code ends
     end start
```

实验八：分析一个奇怪的程序

```
assume cs:codesg
codesg segment
     mov ax,4c00h
     int 21h
start:
     mov ax,0
   s:
     nop
     nop
     mov di,offset s
     mov si,offset s2
     mov ax,cs:[si]
```

```
    mov cs:[di],ax
  s0:
    jmp short s
  s1:
    mov ax,0
    int 21h
    mov ax,0
  s2:
    jmp short s1
    nop
codesg ends
    end start
```

注：程序可以正常运行，本程序主要考察了转移指令的使用!!!

实验九：根据材料编程

```
;*************************************************************************
;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!'.
;实验原理见书中 186 页
;*************************************************************************
assume cs:code,ds:data,es:display,ss:stack
data segment
    db 'welcome to masm!'
    db 02H,24H,71H
data ends
stack segment
     db 16 dup(0)
stack ends
display segment
    db 1024 dup(0)
display ends
code segment
start:
    mov ax,data
    mov ds,ax
    mov si,0
    mov ax,0b872H                    ;定义 12 行中间的起始位置
    mov es,ax
    mov cx,3
    mov di,0
;*********************************
;做三次外循环,每次显示一种颜色
;*********************************
 s0:
    push cx
    push ax
```

```
    mov cx,16
    mov bx,0
;************************************************
;做上六次内循环,将数据段定义的字符串写入内存中
;************************************************
  s:

    mov al,[bx]
    mov es:[si],al
    mov al,[di+10h]
    mov es:[si+1],al
    add si,2
    inc bx
  loop s
    sub si,32                          ;将 si 回到初始位置
    add si,160                         ;换行
    pop ax
    pop cx
    inc di                             ;改变颜色
 loop s0
    mov ax,4c00h
    int 21h
code ends
     end start
```

第十章

检测点 10.1

第一空: 1000

第二空: 0000

检测点 10.2

ax=6

注意: 执行 call s 是 ip 的值为 6 接着进栈, 然后执行 pop ax 相当于把 ip 的值放到 ax 中!!

检测点 10.3

ax=1100H

注意: 执行 callfar ptr s 后 cs=1000 ip=3 (10H) 接着进栈 然后执行 pop ax 此时 ax=3 ==〉add ax, ax 此时 ax=6 (100H) ==〉pop bx 此时 bx=1000 ==〉add ax, bx 此时 ax=1100H

检测点 10.4

ax=000B 具体我就不一一分析了, 呵呵!!

检测点 10.5

(1)

ax=3

(1) ax=1 bx=0

实验十: 编写子程序

(1) 显示字符串

```
;*************************************************************************
;显示字符串的子程序 (dh)= 行号(取值范围 0-24),(dl)=列号(取值范围 0-79)
;(cl)=颜色,ds:si 指向字符串的首地址
;*************************************************************************
assume cs:code,ds:data,es:display, ss:stack
data segment
   db 'welcome to masm!',0
data ends
stack  segment
   db  8 dup(0)
stack ends
display segment
   db 1024 dup(0)
display ends
code segment
start:
     mov dh,8
     mov dl,3
     mov cl,2
     mov ax,data
     mov ds,ax
     mov si,0
     call show_str
     mov ax,4c00h
     int 21h
;*****************************************
;显示字符串的子程序
;*****************************************
show_str:
     push cx
     push si
     mov ax,0B800H
     mov es,ax
     mov al,0a0h
     dec dh
     mul dh
     mov bx,ax
     mov al,2
     mul dl
     sub dl,2
     add bx,ax                    ;得到偏移地址=(dh-1)*160+dl*2-2
     mov di,0
     mov ch,0
```

```
     mov al,cl
    s:
     mov cl,ds:[si]
     jcxz next                ;进行判断最后一个字符是否是零,如果是零则结束
     mov es:[bx+di],cl        ;将字符串放到偶地址中
     mov es:[bx+di+1],al      ;颜色属性放到奇地址中
     add di,2
     inc si
     jmp s
next:
   pop si
   pop cx
   ret
code ends
     end start
```

注意:子程序中用到的寄存器一定要将其保存起来,注意参数的传递!!!,只要程序中要进栈和出栈,就一定要定义堆栈段!!!!

2.解决除法溢出问题

```
;*************************************************************************
;解决除法溢出问题,
;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,ss:stack
stack segment
  dw 8 dup(0)
stack ends
code segment
start:
        mov ax,stack
        mov ss,ax
        mov sp,10h
        mov ax,4240h
        mov dx,0fh
        mov cx,0ah
        call divdw
        mov ax,4c00h
        int 21h
divdw:                          ;子程序定义开始
        push ax
```

```
        mov ax,dx
        mov dx,0
        div cx
        mov bx,ax
        pop ax
        div cx
        mov cx,dx
        mov dx,bx           ;dx:ax/cx=高位的商*65536+[高位的余数*65536+低位]/cx
        ret                 ;子程序定义结束
code ends
end start
```

3.数值显示子程序

```
;*************************************************
;数值显示子程序
;编程,将 data 段中的数据以十进制的形式显示出来
;*************************************************
assume cs:code,ds:data
data segment
   db 10 dup(0)
data ends
code segment
start:
   mov ax,12666
   mov bx,data
   mov ds,bx
   mov si,0
   call dtoc
   mov dh,8
   mov dl,3
   mov cl,2
   call show_str
   mov ax,4c00h
   int 21h
;*************************************************************
;将 word 型的数据转变为十进制数字符串的子程序
;*************************************************************
dtoc:
   push ax
   push si
   push dx
   push cx
   push bx
```

```
    mov bx,0

s0:
    mov cx,10d
    mov dx,0
    div cx
    mov cx,ax                          ;将商放到 cx 中判断商是否为零
    jcxz s1
    add dx,30h
    push dx
    inc bx
    jmp s0
s1:
   add dx,30h
   push dx
   inc bx
   mov cx,bx
   mov si,0
s2:
   pop ax
   mov ds:[si],al
   inc si
   loop s2
   pop bx
   pop cx
   pop dx
   pop si
   pop ax
   ret
;*************************************************************
;显示字符串子程序
;*************************************************************
show_str:
      push bx
      push cx
      push si
      mov al,0a0h
      dec dh
      mul dh
      mov bx,ax
      mov al,2
      mul dl
      sub ax,2
      add bx,ax                        ;得到偏移地址=(dh-1)*160+ (dl-2)*2
```

```
 mov ax,0B800H
 mov es,ax
 mov di,0
 mov ch,0
 mov al,cl
s:
 mov cl,ds:[si]
 jcxz next                    ;进行判断最后一个字符是否是零,如果是零则结束
 mov es:[bx+di],cl            ;将字符串放到偶地址中
 mov es:[bx+di+1],al          ;颜色属性放到奇地址中
 add di,2
 inc si
 jmp  short s
next:
   pop si
   pop cx
   pop bx
   ret
code ends
  end start
```

课程设计一 :

```
;*************************************************************
;课程设计 1,将下面的 data 中的数据 按照表格的形式显示在屏幕上
;*************************************************************
assume cs:code,ds:data
data segment
        db 11 dup(0)                    ;在数据段开辟一段空间存放数据转换后的字符
串
        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 11542,14430,45257,17800
        ;以上是表示 21 年公司雇员人数的 21 个 word 型数据
data ends
code segment
start:
      mov ax,data
```

```
        mov ds,ax
        mov cx,21
        sub si,si
        mov bx,0
        mov dh,2
  s0:
      push cx
      mov di,dx          ;此处一定要注意,实现的是 si 能加四,注意数据段中各种数据的类
型!!
      mov cl,02h
      mov dl,0

;*****************************************
;把年份送到 table 中
;*****************************************
      mov ax,[si+11]
      mov ds:[0],ax
      mov ax,[si+2+11]
      mov ds:[2],ax
      mov byte ptr ds:[4], 0
      push si
      mov si,0

      call show_str
      pop si
;*********************************************
;把收入送到 table 中
;*********************************************
      mov ax,[si+84+11]
      mov dx,[si+86+11]
      push si
      call dtoc
      mov dx,di
      mov dl,20
      call show_str
      pop si
;*************************************************************
;把人数送到 table 中,注意人数是双字节,所以每次不能增四,只能增二
;*************************************************************
      sub si,bx
      mov ax,[si+168+11]
      mov bp,ax
      sub dx,dx
      add si,bx
```

```
        push si
        call dtoc
        mov dx,di
        mov dl,40
        call show_str
        pop si
;*****************************************
;计算人均收入并把其送到 table 中
;*****************************************
        mov ax,[si+84+11]
        mov dx,[si+86+11]
        div bp
        sub dx,dx                                   ;只要人均值,余数不要
        push si
        call dtoc
        mov dx,di
        mov dl,60
        call show_str
        pop si

        add si,4
        add bx,2
        inc dh
        pop cx
    loop s0
        mov ax,4c00h
        int 21h
;*********************************************************
;数字转化为字符串子程序
;*********************************************************
dtoc:
push ax
push cx
push dx
mov si,9
  dnext:
mov cx,10
call divdw
add cx,30h
mov [si],cl                         ;将转换好的字符串放到我们定义的数据段中
dec si
cmp dx,0
jne dnext
cmp ax,0
```

```
jne dnext
inc si
pop dx
pop cx
pop ax
ret
;***********************************************
;防溢出除法子程序
;***********************************************
divdw:
        jmp short divstart
datareg dw  4 dup (0)
  divstart:
push bx
push ds
push si
cmp dx,cx ;通过这里实现兼容没有溢出的除法
jb divnoflo
mov bx,cs
mov ds,bx
mov si,offset datareg
mov [si],ax                ;保存低 16 位 L
mov ax,dx   ;求 H/N,得到 int(H/N)和 rem(H/N),分别保存在 ax 和 dx 当中
sub dx,dx ;***这个语句非常重要，对 dx 清零，避免溢出
div cx
mov [si+2],dx ;保存 rem(H/N)
mov bx,512 ;求得 int(H/N)*65536
mul bx
mov bx,128
mul bx
mov [si+4],ax ;保存 int(H/N)*65536
mov [si+6],dx
mov ax,[si+2] ;求得 rem(H/N)*65536
mov bx,512
mul bx
mov bx,128
mul bx
add ax,[si] ;求得 rem(H/N)*65536＋L
div cx ;求得[rem(H/N)*65536＋L]/N ***注意这里进行的除法不能清除 dx,这里不可能会
溢出
mov cx,dx ;求得结果得余数
add ax,[si+4] ;求得结果的低 16 位
mov dx,[si+6] ;求得结果得高 16 位
jmp short dsret
```

```
 divnoflo:
div cx
mov cx,dx
sub dx,dx
dsret:
pop si
pop ds
pop bx
ret
;*************************************************************
;显示字符串子程序
;*************************************************************
show_str:
push ax ;注意子程序中要用到的寄存器要保存起来
push dx
push cx
push es
push di
push si
push bx


mov ax,0
mov al,0a0h
mul dh
mov bx,ax
mov al,2
mul dl

add bx,ax                    ;得到偏移地址=dh*160+dl*2

 mov di,bx                   ;将偏移地址保存起来
 mov ax,0B800H
 mov es,ax
 mov al,cl
 mov cx,0
s:
 mov cl,ds:[si]
 jcxz next                   ;进行判断最后一个字符是否是零,如果是零则结束
 mov es:[di],cl          ;将字符串放到偶地址中
 mov es:[di+1],al        ;颜色属性放到奇地址中
 add di,2
 inc si
 jmp  short s
```

```
next:
     pop bx
     pop si
     pop di
     pop es
     pop cx
     pop dx
     pop ax
   ret

code ends
     end start
```

第十一章

检测点 11.3

(1) 第一空: jnb s0　　第二空: jna s0

(2) 第一空: jb s0　　第二空: ja s0

检测点 11.4

ax=45H

注意: 需要把标志寄存器的各个位写出来, 然后再进行与运算!!

实验十一: 编写子程序

```
;*********************************************
;编写子程序 letterc 将以 0 结尾的字符串中的小写
;字母转变成大写字母,ds:[si]指向字符串首地址
;*********************************************
assume cs:codesg,ds:datasg
datasg segment
  db "Beinner's All-purpose symbolic Instruction Code.",0
datasg ends
codesg segment
start:
   mov ax,datasg
   mov ds,ax
   mov si,0
   call letterc
   mov ax,4c00h
   int 21h
;************************
;子程序 letterc
;************************
letterc:
    push ax
    push si
  s0:
```

```
    mov al,[si]
    cmp al,0
    jmp last
    cmp al,61h
    jb next
    cmp al,7ah
    ja next
    and al,11011111B
    mov [si],al
next:
    inc si
    jmp s0
last:
    pop si
    pop ax
ret
codesg ends
       end start
```

实验 12 编写 0 号中断的处理程序

```
;*************************************************
;编写程序,使得在除法溢出时,在屏幕中间显示字符串
;"divide error!"然后返回 dos
;*************************************************
assume cs:code,ss:stack
stack segment
  db 128 dup(0)
stack ends
code segment
start:
   mov ax,cs
   mov ds,ax
   mov si,offset do0                    ;源地址

   mov ax,0
   mov es,ax
   mov di,200h                          ;目标地址
   mov cx,offset do0end-offset do0
   cld
   rep movsb                            ;将中断处理程序放到内存中

   mov word ptr es:[0*4],200h
   mov word ptr es:[0*4+2],0            ;设置中断向量表
   int 0                                                ;调用零号中断
```

试试

```
   do0:                                                    ;显示中断处理程序
    jmp short do0start
    db "divide errors! "
do0start:
   mov ax,cs
   mov ds,ax
   mov si,202h
   mov ax,0b800h
   mov es,ax
   mov di,12*160+36*2                                      ;计算 dos 中间位置
   mov cx,14
s: mov al,[si]
   mov es:[di],al
   inc si
   loop s
   mov ax,4c00h
   int 21h
do0end:nop
code ends
    end start
```

第十三章

检测点 13.1

(1) 128　注:因为 loop 实现的是段内短转移,目的地址必须在离本指令-128-----127 范围内

```
;*****************************************
;利用七号中断实现 jmp near ptr s 指令
;*****************************************
assume cs:code
data segment
        db 'conversation',0
data ends
code segment
start:
        mov ax,cs
        mov ds,ax
        mov si,offset nr
        mov ax,0
        mov es,ax
        mov di,200h
        mov cx,offset nrend-offset nr
        cld
        rep movsb
```

```
        ;以上 9 句为安装中断例程

        mov word ptr es:[7ch*4],200h
        mov word ptr es:[7ch*4+2],0

        mov ax,data
        mov ds,ax
        mov si,0
        mov ax,0b800h
        mov es,ax
        mov di,12*160
s:      cmp byte ptr [si],0
        je ok
        mov al,[si]
        mov es:[di],al
        inc si
        add di,2
        mov bx,offset s-offset ok
        int 7ch
ok:     mov ax,4c00h
        int 21h

nr:     push bp             ;定义中断例程[开始]
        mov bp,sp
        add [bp+2],bx
nrret:  pop bp
        iret
nrend:  nop                 ;定义中断例程[结束]

code ends
end start
```

检测点 13.2

(1)

错误,因为 bios 是不可写的,不能向里面写程序

(2)

错误 19 号中断是引导操作系统的,必须在在操作系统还没有执行前提供

实验 13 编写、应用中断例程

```
 ;*************************************************
;编写并安装 int 7ch 中断例程,功能为显示一个用 0
;结束的字符串,中断例程安装在 0:200 处
;*************************************************
assume cs:code
```

```
code segment
start:
     mov ax,cs
     mov ds,ax
     mov si,offset display

     mov ax,0
     mov es,ax
     mov di,200h
     mov cx,offset displayend-offset display
     cld
     rep movsb

     mov ax,0
     mov es,ax
     mov word ptr es:[7ch*4],200h
     mov word ptr es:[7ch*4+2],0


     mov ax,4c00h
     int 21h

display:
        push cx
        push ax
 push bx
 push dx
 push bp
 push es
        push di

        mov ax,0b800h
        mov es,ax
        mov ah,0
        mov al,dh
        mov bl,160
        mul bl
        mov bp,ax
        mov dh,0
        add dl,dl
        mov di,dx
```

```
    mov al,cl

  s:
    mov cl,ds:[si]
    mov ch,0
    jcxz next
    mov es:[bp+di],cl
    mov es:[bp+di+1],al
    inc si
    add di,2
    jmp short s
next:
        pop di
 pop es
 pop bp
 pop dx
 pop bx
 pop ax
        pop cx
        iret
displayend:nop

code ends
     end start
```

实验 14 访问 CMOS RAM

```
;*****************************************
assume cs:code,ds:data,es:info_num
data segment
    db '11/11/11 11:11:11$'          ;预设字符串
data ends
info_num segment
    db 9,8,7,4,2,0                   ;端口时间地址列表
info_num ends
code segment
start:
    mov ax,data
    mov ds,ax
    mov si,0                 ;初始指向字符串首
    mov ax,info_num
    mov es,ax
    mov bp,0                 ;指向端口时间地址列表首
    mov cx,6
```

```
s:
    push cx
    mov al,es:[bp]
    out 70h,al
    in al,71h
    mov ah,al          ;暂存 al
    mov cl,4
    shr ah,cl          ;获取 BCD 码高四位
    and al,00001111B     ;获取 BCD 码低四位
    add al,30h
    add ah,30h
    mov ds:[si],ah
    mov ds:[si+1],al     ;将时间信息写入字符串指定位置
    add si,3           ;指向字符串下一写入位置
    inc bp               ;指向端口时间地址列表下一位置
    pop cx
    loop s
    mov ah,2
    mov bh,0
    mov dh,12
    mov dl,50
    int 10h              ;调用系统 BIOS 中断例程设置光标位置
    mov ah,9
    mov dx,0           ;指向字符串首
    int 21h              ;调用 DOS 中断例程显示字符串
    mov ah,2
    mov bh,0
    mov dh,24
    mov dl,0
    int 10h              ;开始没有这一段，其他并没有问题，就是发现调用 21h 例程后光
标位置直接在字符串的下一行
                         ;通过这一段重置光标位置
    mov ax,4c00h
    int 21h
code ends
end start
```

实验 15 安装新的 int9 中断例程

```
;中断时的入栈顺序是 pushf, push cs,  push ip
assume cs : codesg, ss : stacksg
stacksg SEGMENT
     dw 64 dup (0)
stacksg ENDS
```

```
codesg SEGMENT
start:    mov ax, stacksg
    mov ss, ax
    mov sp, 128
    mov ax, 0
    mov es, ax
    mov di, 0200h
    mov si, offset int9
    push cs
    pop ds

    cli
    mov bx, offset table
    mov ax, es : [9 * 4]
    mov ds : [bx], ax
    mov ax, es : [9 * 4 + 2]
    mov ds : [bx + 2], ax
    sti

    mov cx, offset int9end - offset int9
    cld
    rep movsb
    mov word ptr es : [9 * 4 + 2], 0
    mov word ptr es : [9 * 4], 0200h

    mov ax, 4c00h
    int 21h
int9:    jmp short s
    table dw 0, 0
s:    push ax
    push cx
    push si
    push es
    mov ax, 0
    mov ds, ax
    pushf
    call dword ptr ds : [202h]
    in al, 60h
    cmp al, 9eh
    jne int9ret
    mov ax, 0b800h
    mov es, ax
    mov si, 0
    mov cx, 2000
```

```
s1:    mov byte ptr es : [si], 'A'
    add si, 2
    loop s1
int9ret:pop es
    pop si
    pop cx
    pop ax
    iret
int9end:nop
codesg ENDS
END start
```

实验 16 编写包含多个功能子程序的中断例程

```
;中断时的入栈顺序是 pushf, push cs,  push ip
assume cs : codesg, ss : stacksg
stacksg SEGMENT
     dw 16 dup (0)
stacksg ENDS
codesg SEGMENT
start:    mov ax, 0
    mov es, ax
    mov di, 0200h
    mov ax, codesg
    mov ds, ax
    mov si, offset screen
    mov cx, offset scend - offset screen
    cld
    rep movsb

    mov word ptr es : [7ch * 4], 0200h
    mov word ptr es : [7ch * 4 + 2], 0
    mov ax, 4c00h
    int 21h
screen:    jmp short begin
    table dw f0 - screen + 200h, f1 - screen + 200h, f2 - screen + 200h, f3 - screen
+ 200h
    begin:    push bx
        push ds
        mov bx, 0
        mov ds, bx
        cmp ah, 3
        ja sret
        mov bl, ah
        mov bh, 0
```

```
        add bx, bx
        call word ptr [table + bx - screen + 200h]
    sret:    pop ds
        pop bx
        iret
f0:        push bx
        push es
        push cx
        mov bx, 0b800h
        mov es, bx
        mov bx, 0
        mov cx, 2000
    f0s:    mov byte ptr es : [bx], ' '
        add bx, 2
        loop f0s
        pop cx
        pop es
        pop bx
        ret
f1:        push bx
        push es
        push cx
        mov bx, 0b800h
        mov es, bx
        mov bx, 0
        mov cx, 2000
    f1s:    and byte ptr es : [bx + 1], 11111000b
        or es : [bx + 1], al
        add bx, 2
        loop f1s
        pop cx
        pop es
        pop bx
        ret
f2:        push bx
        push es
        push cx
        mov cl, 4
        shl al, cl
        mov bx, 0b800h
        mov es, bx
        mov bx, 0
        mov cx, 2000
    f2s:    and byte ptr es : [bx + 1], 10001111b
```

```
        or es : [bx + 1], al
        add bx, 2
        loop f2s
        pop cx
        pop es
        pop bx
        ret
f3:        push di
        push es
        push si
        push cx
        mov di, 0b800h
        mov es, di
        mov ds, di
        mov si, 160
        mov cx, 2000 - 80
        cld
        rep movsw
        mov di, 1920
        mov cx, 80
  f3s:    mov byte ptr es : [di], ' '
        add di, 2
        loop f3s
        pop cx
        pop si
        pop es
        pop di
        ret
scend:    nop
codesg ENDS
END start
```


实验 17 编写包含多个功能子程序的中断例程

```
assume cs:code,ds:data
data segment
strd db 60000 dup (0) ;读取磁盘内容到这里
data ends
code segment
start: mov ax,data
mov ds,ax
mov si,offset strd
mov ax,0
mov es,ax
```

```
push si
pop es:[204h] ;保存 data 段偏移地址
push ds
pop es:[206h] ;保存 data 段地址
push cs
pop ds
call setup7ch ;安装 7ch 中断
mov ax,0
mov es,ax
cli
push es:[13h*4] ;复制 13h 中断偏移
pop es:[200h]
push es:[13h*4+2] ;复制 13h 中断段
pop es:[202h]
mov word ptr es:[7ch*4],208h ;7ch 中断入口
mov word ptr es:[7ch*4+2],0
sti
mov ah,0
mov bx,9
int 7ch
mov ax,4c00h
int 21h
setup7ch proc ;安装 7ch 子程序
push ax
push si
push di
push cx
mov si,offset new7ch
mov ax,0
mov es,ax
mov di,208h
mov cx,offset new7chend - offset new7ch
cld
rep movsb
pop cx
pop di
pop si
pop ax
```

```
ret
setup7ch endp
new7ch: jmp short nstart ;7ch 中断例程
strc db 'i love this world!'
nstart: push cs
pop ds
push ds
pop es
push bx
cmp ah,1 ;0 为读, 1 为写
jne fread
fwrite:pop bx
call setfld ;根据王爽老师提供的逻辑扇区算法(计算逻辑扇区到物理扇区的子程序).
mov bx,offset strc
mov al,1
mov ah,3
pushf
call dwordptr es:[200h] ;调用旧 int 13h 例程,此时入口地址在 0:200H
iret
fread: pop bx
call setfld
mov ax,es:[206h] ; es 指向 DATA 段地址
mov es,ax
mov bx,es:[204h] ;bx 指向 data 偏移地址
mov al,1
mov ah,2
pushf
call dwordptr cs:[200h]
iret
setfld proc ;计算逻辑扇区到物理扇区的子程序,入口参数 BX , 出口参数: cl 扇区号,CH 磁道号,dl 软驱,dh 柱面号
mov ax,0
mov ax,bx
xor dx,dx
mov bx,1440
div bx
xchg ah,al
```

```
push ax
mov ax,dx
xor dx,dx
mov bh,0
mov bl,18
div bl
mov ch,al
xchg ah,al
mov ah,0
inc ax
mov cl,al
pop dx
ret
setfld endp
new7chend:nop
code ends
end start
```