

Chapter 11

1. test11.1

写出下面每条指令执行后,ZF.PF.SF等标志位的值

sub al,al ;ZF=1 PF=1 SF=0

mov al,1 ;1 1 0(有问题)

push ax ;1 1 0

pop bx ;1 1 0

add al,bl ;0 0 0

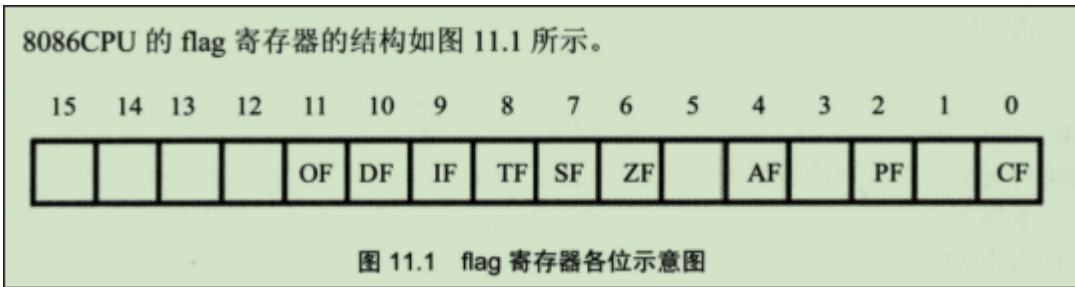
add al,10 ;0 1 0

mul al ;0 1 0

- PF是flag的第2位，奇偶标志位，记录指令执行后结果二进制中1的个数是否为偶数，结果为偶数时，PF=1
- ZF是flag的第6位，零标志位，记录指令执行后结果是否为0，结果为0时，ZF=1
- SF是flag的第7位，符号标志位，记录有符号运算结果是否为负数，结果为负数时，SF=1

add、sub、mul、div、inc、or、and等运算指令影响标志寄存器

mov、push、pop等传送指令对标志寄存器没影响。



2. test 11.2

;CF OF SF ZF PF

sub al,al ;0 0 0 1 1

mov al,10h ;0 0 0 1 1

add al,90h ;0 0 1 0 1

mov al,80h ;0 0 1 0 1

add al,80h ;1 1 0 1 1

mov al,0fch ;1 1 0 1 1

add al,05h ;1 0 0 0 0

mov al,7dh ;1 0 0 0 0

add al,0bh ;0 1 1 0 1

检测点涉及的相关内容：

- CF是flag的第0位，进位标志位，记录无符号运算结果是否有进/借位，结果有进/借位时，SF=1
- PF是flag的第2位，奇偶标志位，记录指令执行后结果二进制数中1的个数是否为偶数，结果为偶数时，PF=1
- ZF是flag的第6位，零标志位，记录指令执行后结果是否为0，结果为0时，ZF=1
- SF是flag的第7位，符号标志位，记录有符号运算结果是否为负数，结果为负数时，SF=1
- OF是flag的第11位，溢出标志位，记录有符号运算结果是否溢出，结果溢出时，OF=1

add、sub、mul、div、inc、or、and等运算指令影响flag

mov、push、pop等传送指令对flag没影响

指令 `cmp ax,bx` 的逻辑含义是比较 `ax` 和 `bx` 中的值，如果执行后：

`zf=1`，说明 $(ax)=(bx)$

`zf=0`，说明 $(ax) \neq (bx)$

`cf=1`，说明 $(ax) < (bx)$

`cf=0`，说明 $(ax) \geq (bx)$

`cf=0` 并且 `zf=0`，说明 $(ax) > (bx)$

`cf=1` 或 `zf=1`，说明 $(ax) \leq (bx)$

指令	含义	检测的相关标志位
<code>je</code>	等于则转移	<code>zf=1</code>
<code>jne</code>	不等于则转移	<code>zf=0</code>
<code>jb</code>	低于则转移	<code>cf=1</code>
<code>jnb</code>	不低于则转移	<code>cf=0</code>
<code>ja</code>	高于则转移	<code>cf=0</code> 且 <code>zf=0</code>
<code>jna</code>	不高于则转移	<code>cf=1</code> 或 <code>zf=1</code>

下面，我们以 `cmp ah,bh` 为例，总结一下 CPU 执行 `cmp` 指令后，`sf` 和 `of` 的值是如何来说明比较的结果的。

(1) 如果 `sf=1`，而 `of=0`

`of=0`，说明没有溢出，逻辑上真正结果的正负=实际结果的正负；

因 `sf=1`，实际结果为负，所以逻辑上真正的结果为负，所以 `(ah)<(bh)`。

(2) 如果 `sf=1`，而 `of=1`：

`of=1`，说明有溢出，逻辑上真正结果的正负 \neq 实际结果的正负；

因 `sf=1`，实际结果为负。

实际结果为负，而又有溢出，这说明是由于溢出导致了实际结果为负，简单分析一下，就可以看出，**如果因为溢出导致了实际结果为负，那么逻辑上真正的结果必然为正**。这样，`sf=1`，`of=1`，说明了 `(ah)>(bh)`。

(3) 如果 `sf=0`，而 `of=1`

`of=1`，说明有溢出，逻辑上真正结果的正负 \neq 实际结果的正负；

因 `sf=0`，实际结果非负。而 `of=1` 说明有溢出，则结果非 0，所以，实际结果为正。

实际结果为正，而又有溢出，这说明是由于溢出导致了实际结果非负，简单分析一下，就可以看出，**如果因为溢出导致了实际结果为正，那么逻辑上真正的结果必然为负**。这样，`sf=0`，`of=1`，说明了 `(ah)<(bh)`。

(4) 如果 `sf=0`，而 `of=0`

`of=0`，说明没有溢出，逻辑上真正结果的正负=实际结果的正负；

因 `sf=0`，实际结果非负，所以逻辑上真正的结果非负，所以 `(ah) \geq (bh)`。

3. test 11.3

```
1 ;统计F000:0处32个字节中,大小在[32,128]的数据的个数
2 assume cs:code
3 code segment
4 start:
5 mov ax,0f000h
6 mov ds,ax
7
8 mov bx,0
9 mov dx,0 ;初始化累加器
10 mov cx,32
11
12 s: mov al,ds:[bx]
13     cmp al,32
14     jb s0 ;当小于32,跳到s0,继续循环
15     cmp al,128
16     ja s0 ;当大于128,跳到s0,继续循环
17     inc dx
18 s0: inc bx
19     loop s
20
21     mov ax,4c00h
22     int 21h
23
24 code ends
```

```

25 end start
26
27 ;统计F000:0处32个字节中,大小在(32,128)的数据的个数
28 assume cs:code
29 code segment
30 start:
31 mov ax,0f000h
32 mov ds,ax
33
34 mov bx,0
35 mov dx,0 ;初始化累加器
36 mov cx,32
37
38 s: mov al,ds:[bx]
39     cmp al,32
40     jna s0 ;当不大于32,跳到s0,继续循环
41     cmp al,128
42     jnb s0 ;当不小于128,跳到s0,继续循环
43     inc dx
44 s0: inc bx
45     loop s
46
47     mov ax,4c00h
48     int 21h
49
50 code ends
51 end start

```

4. test 11.4

```

1 mov ax,0 ;ax=0
2 push ax ;
3 popf ;
4 mov ax,0fff0h ;ax=fff0h
5 add ax,0010h ;
6 pushf ;
7 pop ax ;ax=0047h 因为flag寄存器中的次低位(第1bit)为1
8 and al,11000101B ;ax=0045h 与SF ZF PF CF 相与
9 and ah,00001000B ;ax=0045h

```

lab 11

1. 转换大写

```

1 assume cs:code
2
3 data segment
4     db "Beginner's All-purpose Symbolic Instruction Code.",0
5 data ends
6
7 code segment
8     begin:

```

```

9          mov ax,data
10         mov ds,ax
11         mov si,0      ;ds:si同时指向源以及目的地址
12         call letterc
13
14         mov ax,4c00h
15         int 21h
16
17         ;名称;letterc
18         ;功能:将以0结尾的字符串中的小写字母转变为大写字母
19         ;参数:ds:si指向字符串的首地址
20
21         letterc:
22             push si
23             push cx
24
25         s1: push cx
26             mov cx,ds:[si]
27             jcxz ok      ;当为0时,则结束循环
28             and byte ptr ds:[si],11011111B;否则,转换为大写
29             inc si
30             pop cx
31             loop s1
32         ok:
33             pop cx
34             pop si
35             ret
36
37         code ends
38         end begin

```

```
DOSBox 0.74-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
C:\>link lab11:

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

LINK : warning L4021: no stack segment

C:\>debug lab11.exe
-g b

AX=076A BX=0000 CX=0030 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076E IP=000B NU UP EI PL NZ AC PE NC
076E:000B B8004C MOU AX,4C00
-d ds:0 f
076A:0000 42 45 47 49 4E 4E 45 52-07 53 00 41 4C 4C 0D 50 BEGINNER.S.ALL.P
-d ds:0
076A:0000 42 45 47 49 4E 4E 45 52-07 53 00 41 4C 4C 0D 50 BEGINNER.S.ALL.P
076A:0010 55 52 50 4F 53 45 00 53-59 4D 42 4F 4C 49 43 00 URPOSE.SYMBOLIC.
076A:0020 49 4E 53 54 52 55 43 54-49 4F 4E 00 43 4F 44 45 INSTRUCTION.CODE
076A:0030 0E 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0040 B8 6A 07 8E D8 BE 00 00-E8 05 00 B8 00 4C CD 21 .j.....L.!
076A:0050 56 51 51 8B 0C E3 07 80-24 DF 46 59 E2 F4 59 5E UQQ.....$.FY..Y^
076A:0060 C3 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83 .@P.....RP..H.
076A:0070 C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6 ..P....P..s.....
```

Chapter 12

- 我们如何让一个内存单元成为栈顶?将它的地址放入SS:SP中
- 我们如何让一个内存单元中的信息被CPU当做指令来执行?将它的地址放入CS:IP中
- 我们如何让一段程序成为N号中断的中断处理程序?将它的入口地址放入中断向量表的N号表项中偏移地址放入0:4N字单元中,段地址放入0:4N+2字单元中

1. test 12.1

(1) 用 Debug 查看内存, 情况如下:

0000:0000 68 10 A7 00 8B 01 70 00-16 00 9D 03 8B 01 70 00

则 3 号中断源对应的中断处理程序的入口地址为: _____。

(2) 存储 N 号中断源对应的中断处理程序入口的偏移地址的内存单元的地址为: _____。

存储 N 号中断源对应的中断处理程序入口的段地址的内存单元的地址为: _____。

(1)0070:018b

(2)4N 0000

2. 中断的详细过程

中断过程(中断向量:中断处理程序的入口地址)

1. 取得中断类型码N
2. pushf
3. TF=0,IF=0

4. push CS
5. push IP
6. (IP)=(N * 4) (CS)=(N * 4+2)

中断处理程序

1. 保存用到的寄存器
2. 处理中断
3. 恢复用到的寄存器
4. 用iret指令返回 pop IP pop CS popf

编写0号中断向量中断处理程序

1. 编写可以显示"overflow!"的中断处理程序:do0:
2. 将do0送入内存0000:0200处
3. 将do0的入口地址0000:0200存储在中断向量表0号表项中

3. 基本流程

```
1  assume cs:code
2
3  code segment
4  start: do0安装程序
5          设置中断向量表
6          mov ax,4c00h
7          int 21h
8
9  do0:    保存现场
10         显示字符串"overflow!"
11         恢复现场
12         mov ax,4c00h
13         int 21h
14
15 code ends
16
17 end start
```

lab 12

1. example

```
1  assume cs:code
2
3  code segment
4
5  start:
6      mov ax,cs
7      mov ds,ax
8      mov si,offset do0 ;设置ds:si指向源地址
9
10     mov ax,0
```

```

11      mov es,ax
12      mov di,200h          ;设置es:di指向目的地址
13
14      mov cx,offset do0end-offset do0          ;设置cx为传输长度,利用编译器来计算
do0长度;
15      ;因为汇编编译器可以处理表达式
16
17      cld                  ;设置传输方向为正
18
19      rep movsb
20
21
22      mov ax,0
23      mov es,ax
24      mov word ptr es:[0*4],200h
25      mov word ptr es:[0*4+2],0      ;设置中断向量表
26
27      mov ax,4c00h
28      int 21h
29
30      do0: jmp short do0start ;指令占2B
31          db "overflow!"
32
33      do0start:
34          mov ax,cs
35          mov ds,ax
36          mov si,202h          ;设置ds:si指向字符串(字符串存放在代码段中,偏移地址,0:200处的指令
为jmp short do0start,这条指令占两个字节,所以"overflow!"的偏移地址为202h
37
38          mov ax,0b800h
39          mov es,ax
40          mov di,12*160+36*2 ;设置es:di指向显存空间的中间位置
41
42          mov cx,9            ;设置显示字符串长度
43      s: mov al,ds:[si]
44          mov es:[di],al
45          inc si
46          add di,2
47          loop s
48
49          mov ax,4c00h
50          int 21h            ;用来返回DOS的
51
52      do0end: nop
53
54      code ends
55      end start

```

2. 实验源码

```

1  assume cs:code
2
3  code segment

```



```

4
5 start:
6     ;安装,主程序负责将中断处理程序转移到起始地址为0:200内存单元中.
7     mov ax,cs
8     mov ds,ax
9     mov si,offset do0      ;ds:si指向源地址
10
11     mov ax,0
12     mov es,ax
13     mov di,200h           ;es:di指向目的地址(即中断处理程序的起始地址)
14
15     mov cx,offset do0end - offset do0    ;设置程序所占的字节数大小
16
17     cld                   ;设置df=0,正向传送
18
19     rep movsb             ;循环传送
20
21     mov ax,0
22     mov es,ax
23     mov word ptr es:[0*4],200h
24     mov word ptr es:[0*4+2],0    ;设置中断向量
25
26
27
28     mov ax,4c00h
29     int 21h              ;返回DOS
30
31     ;do0即为中断处理程序,负责显示.
32
33
34     do0: jmp short do0start ;指令占2B
35     db "divide error!"      ;字符共13个
36
37 do0start:
38     ;flag,cs,ip已经由中断过程保存了
39     push ds
40     push es
41     push si
42     push di              ;保存现场
43     push ax
44     push cx
45
46     mov ax,cs
47     mov ds,ax
48     mov si,202h          ;ds:si指向字符地址,此处可以使用绝对地址,因为这是固定在
底部
49
50     mov ax,0b800h
51     mov es,ax
52     mov di,12*160+34*2    ;es:di指向显存地址
53
54     mov cx,13             ;设置循环次数
55     mov ah,2              ;设置为绿色

```

```

56      s:  mov al,ds:[si]
57          mov es:[di],ax
58          inc si
59          add di,2
60          loop s
61
62          pop cx
63          pop ax
64          pop di          ;恢复现场
65          pop si
66          pop es
67          pop ds
68
69
70          mov ax,4c00h
71          int 21h          ;默认情况下是返回到导致中断的程序处(iret),这里执行中断处理程
序后返回DOS系统
72          ;iret:pop IP;pop CS;popf
73          ;int :push popf;push CS;push IP
74
75
76  do0end:nop
77
78
79  code ends
80  end start

```

DOSBox 0.74-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```

51604 + 464940 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>link lab12;

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.


divide error!


LINK : warning L4021: no stack segment

C:\>lab12

C:\>debug
-a
073F:0100 mov ax,1000
073F:0103 mov bl,1
073F:0105 div bl
073F:0107
-g
C:\>_

```

Chapter 13

1. 问题一:编写,安装中断7ch的中断例程

```
1 ;功能:求一word型数据的平方
2 ;参数:(ax)=要计算的数据
3 ;返回值:dx,ax中存放结果的高16位和低16位
4 assume cs:code
5
6 code segment
7
8 start:
9     mov ax,cs ;安装,主程序负责将中断处理程序转移到起始地址为0:200内存单元中
10    mov ds,ax
11    mov si,offset sqr
12
13    mov ax,0
14    mov es,ax
15    mov di,200h
16
17    mov cx,offset sqrend-offset sqr
18
19    cld
20
21    rep movsb
22
23
24    mov ax,0
25    mov es,ax
26    mov word ptr es:[7ch*4],200h
27    mov word ptr es:[7ch*4+2],0 ;设置中断向量
28
29    mov ax,4c00h
30    int 21h
31
32    sqr:mul ax
33    iret
34 sqrend: nop
35
36 code ends
37 end start
```

2. 问题一:编写,安装中断7ch的中断例程

```
1 ;功能:将一个全是字母,以0结尾的字符串,转化为大写
2 ;参数:ds:si指向字符串的首地址
3 ;返回值:无
4 assume cs:code
5
6 code segment
7
8 start:
9     ;安装,主程序负责将中断处理程序转移到起始地址为0:200内存单元中
10    mov ax,cs
```

```

11         mov ds,ax
12         mov si,offset capital
13
14         mov ax,0
15         mov es,ax
16         mov di,200h           ;目的地址
17
18         mov cx,offset capitalend-offset capital
19
20         cld
21
22         rep movsb
23
24         mov ax,0
25         mov es,ax
26         mov word ptr es:[7ch*4],200h
27         mov word ptr es:[7ch*4+2],0 ;设置中断向量
28
29         mov ax,4c00h
30         int 21h
31
32 capital: push cx
33         push si
34
35 change:  mov cl,[si]
36         mov ch,0
37         jcxz ok
38         and byte ptr ds:[si],11011111b
39         inc si
40         jmp short change
41
42 ok:     pop si
43         pop cx
44         iret
45
46 capitalend:
47         nop
48
49 code ends
50 end start
51
52 ;use
53 assume cs:code
54
55 data segment
56     db 'conversation',0
57 data ends
58
59 code segment
60 start:
61     mov ax,data
62     mov ds,ax
63     mov si,0

```

```

64         int 7ch
65
66         mov ax,4c00h
67         int 21h
68     code ends
69 end start

```

```

DOSBox 0.74-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
DS=076A ES=075A SS=0769 CS=0000 IP=0206  NU UP DI PL NZ NA PO NC
0000:0206 E306      JCXZ   020E
-d ds:0 f
076A:0000  43 4F 6E 76 65 72 73 61-74 69 6F 6E 00 00 00 00  CONVERSATION....
-t
AX=076A BX=0000 CX=006E DX=0000 SP=FFF6 BP=0000 SI=0002 DI=0000
DS=076A ES=075A SS=0769 CS=0000 IP=0208  NU UP DI PL NZ NA PO NC
0000:0208 8024DF      AND    BYTE PTR [SI],DF      DS:0002=6E
-t
AX=076A BX=0000 CX=006E DX=0000 SP=FFF6 BP=0000 SI=0002 DI=0000
DS=076A ES=075A SS=0769 CS=0000 IP=020B  NU UP DI PL NZ NA PE NC
0000:020B 46          INC    SI
-t
AX=076A BX=0000 CX=006E DX=0000 SP=FFF6 BP=0000 SI=0003 DI=0000
DS=076A ES=075A SS=0769 CS=0000 IP=020C  NU UP DI PL NZ NA PE NC
0000:020C EBF4      JMP    0202
-g
Program terminated normally
-d ds:0 f
076A:0000  43 4F 4E 56 45 52 53 41-54 49 4F 4E 00 00 00 00  CONVERSATION....
-

```

3. 对int,iret和栈的深入理解

```

1  assume cs:code
2
3  code segment
4
5      start:
6          mov ax,0b800h
7          mov es,ax
8          mov di,160*12
9
10         mov bx,offset s-offset se    ;设置从标号se到标号s的转移位移
11         mov cx,80
12
13         s: mov byte ptr es:[di],','
14             add di,2
15             int 7ch                    ;如果(cx)不等于0,转移到标号s处
16         se: nop
17
18         mov ax,4c00h
19         int 21h
20

```

```

21
22         lp: push bp
23             mov bp,sp
24             dec cx
25             jcxz lppret
26             add [bp+2],bx
27     lppret: pop bp
28             iret
29
30 code ends
31
32 end start
33

```

4. test 13.1

(1) 在上面的内容中，我们用 7ch 中断例程实现 loop 的功能，则上面的 7ch 中断例程所能进行的最大转移位移是多少？

(2) 用 7ch 中断例程完成 jmp near ptr s 指令的功能，用 bx 向中断例程传送转移位移。

应用举例：在屏幕的第 12 行，显示 data 段中以 0 结尾的字符串。

```

assume cs:code
data segment
    db 'conversation',0
data ends
code segment
start: mov ax,data

```

(1)ffffh

(2)源代码

```

1  ;安装程序
2  ;功能:使用7ch中断例程完成jmp near ptr s指令的功能
3  ;参数:cx:循环次数      bx:传送转移位移
4  ;返回值:无
5  assume cs:code
6
7  code segment
8
9  start:
10         ;安装,主程序负责将中断处理程序转移到起始地址为0:200内存单元中
11         mov ax,cs
12         mov ds,ax
13         mov si,offset lp
14
15         mov ax,0
16         mov es,ax
17         mov di,200h      ;目的地址
18
19         mov cx,offset lpend-offset lp
20
21         cld

```

```

22
23         rep movsb
24
25         mov ax,0
26         mov es,ax
27         mov word ptr es:[7ch*4],200h
28         mov word ptr es:[7ch*4+2],0 ;设置中断向量
29
30         mov ax,4c00h
31         int 21h
32
33 lp:      push bp
34
35         mov bp,sp      ;保存bp
36         dec cx
37         jcxz lpret
38         add [bp+2],bx   ;调用指令的下一条指令的IP+跳转的偏移量
39
40 lpret:   pop bp        ;恢复bp
41         iret
42
43
44 lpend:   nop
45
46 code ends
47 end start
48
49 ;调用程序
50 ;在屏幕的第12行,显示data段中以0结尾的字符串
51 assume cs:code
52
53 data segment
54     db 'conversation',0
55 data ends
56
57 code segment
58 start:
59     mov ax,data
60     mov ds,ax
61     mov si,0
62
63     mov ax,0b800h
64     mov es,ax
65     mov di,12*160
66
67     s:  cmp byte ptr ds:[si],0
68         je ok ;如果是0跳出循环
69         mov ah,2 ;设置绿色
70         mov al,ds:[si]
71         mov es:[di],ax
72         inc si
73         add di,2
74         mov bx,offset s- offset ok ;跳转的偏移量

```

```

75         int 7ch
76
77     ok: mov ax,4c00h
78         int 21h
79 code ends
80 end start

```

运行结果:

```

DOSBox 0.74-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

C:\>tinstall_13

C:\>t13_1

C:\>masm t13_1;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

51566 + 464978 Bytes symbol space free
conversation
    0 Warning Errors
    0 Severe Errors

C:\>link t13_1;

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

LINK : warning L4021: no stack segment

C:\>t13_1

C:\>_

```

5. test 13.2

(1)我们可以编程改变ffff:0处的指令,使得CPU不去执行BIOS中的硬件系统检测和初始化程序. ×

(2)int 19h中断例程,可以由DOS提供 ×

(1) 开机后, CPU 一加电, 初始化(CS)=0FFFFH, (IP)=0, 自动从 FFFF:0 单元开始执行程序。FFFF:0 处有一条转跳指令, CPU 执行该指令后, 转去执行 BIOS 中的硬件系统检测和初始化程序。

(2) 初始化程序将建立 BIOS 所支持的中断向量, 即将 BIOS 提供的中断例程的入口地址登记在中断向量表中。注意, 对于 BIOS 所提供的中断例程, 只需将入口地址登记在中断向量表中即可, 因为它们是固化到 ROM 中的程序, 一直在内存中存在。

(3) 硬件系统检测和初始化完成后, 调用 int 19h 进行操作系统的引导。从此将计算机交由操作系统控制。

(4) DOS 启动后, 除完成其他工作外, 还将它所提供的中断例程装入内存, 并建立相应的中断向量。

6. BIOS和DOS提供的中断例程,都用ah来传递内部子程序的编号

1. BIOS:int 10h

```

1  mov ah,2      ;置光标 表示调用第10h号中断例程的2号子程序
2  mov bh,2      ;第0页

```



```

3  mov dh,5      ;行号
4  mov dl,12     ;列号
5  int 10h
6
7  ;在屏幕的5行12列显示3个红底高亮闪烁绿色的'a' (b1=0cah)
8  assume cs:code
9  code segment
10 start:
11     mov ah,2    ;置光标 表示调用第10h号中断例程的2号子程序
12     mov bh,2    ;第0页
13     mov dh,5    ;行号
14     mov dl,12   ;列号
15     int 10h
16
17     mov ah,9    ;在光标位置显示字符
18     mov al,'a'  ;字符
19     mov b1,0cah ;红底高亮闪烁绿色
20     mov bh,0    ;第0页
21     mov cx,3    ;字符重复个数
22     int 10h
23
24     mov ax,4c00h ;mov al,4ch ;程序返回
25                   ;mov al,0   ;返回值
26     int 21h
27
28 code ends
29 end start

```

2. DOS:int 21h

```

1  ;在屏幕的5行12列显示字符串"welcome to masn!"
2  assume cs:code
3
4  data segment
5      db 'welcome to masn','$'
6  data ends
7
8  code segment
9  start:
10     mov ah,2    ;置光标 表示调用第10h号中断例程的2号子程序
11     mov bh,2    ;第0页
12     mov dh,5    ;行号
13     mov dl,12   ;列号
14     int 10h
15
16     mov ax,data
17     mov ds,ax
18     mov dx,0    ;ds:dx指向字符串的首地址data:0
19     mov ah,9
20     int 21h
21
22     mov ax,4c00h ;mov al,4ch ;程序返回
23                   ;mov al,0   ;返回值

```

```

24         int 21h
25
26     code ends
27 end start

```

lab 13

1. 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

1.调用程序

```

1  ;功能:显示一个用0结束的字符串
2  ;参数:(dh)=行号,(dl)=列号,(cl)=颜色,ds:si指向字符串首地址
3  ;返回:无
4
5  assume cs:code
6  data segment
7      db "welcome to hujie!",0
8  data ends
9
10 code segment
11 start:
12     mov dh,10
13     mov dl,10
14     mov cl,2
15     mov ax,data
16     mov ds,ax
17     mov si,0
18     int 7ch
19     mov ax,4c00h
20     int 21h
21
22
23
24 code ends
25 end start
26

```

2.安装程序

```

1  ;安装程序:中断向量号:7ch,中断向量: 0:200处
2  assume cs:code
3
4  code segment
5  start:
6      mov ax,cs
7      mov ds,ax
8      mov si,offset 1p
9
10     mov ax,0
11     mov es,ax
12     mov di,200h           ;目的地址

```

```

13
14     mov cx,offset lpend-offset lp
15
16     cld
17
18     rep movsb
19
20     mov ax,0
21     mov es,ax
22     mov word ptr es:[7ch*4],200h
23     mov word ptr es:[7ch*4+2],0 ;设置中断向量
24
25     mov ax,4c00h
26     int 21h
27
28 lp:   push es
29       push ax
30       push cx
31       push bx
32       push di ;保存现场
33
34       mov ax,0b800h
35       mov es,ax
36       dec dh ;行
37       mov al,dh
38       mov bl,160
39       mul bl
40       mov di,ax
41
42       dec dl ;列
43       mov al,dl
44       mov bl,2
45       mul bl
46       add di,ax ;es:bx指向显存地址
47
48 change: push cx
49         mov ah,cl ;保存颜色
50         mov al,ds:[si]
51         mov es:[di],ax ;字符显示
52
53         mov cl,al
54         sub ch,ch ;cx中存放的是对应的ASCII码
55
56         jcz ok ;判断其字符码是否为0
57         inc si ;指向下一个字符
58         add di,2 ;指向下一个显存地址
59         pop cx ;恢复颜色
60         jmp short change
61
62 ok:   pop di
63       pop bx
64       pop cx
65       pop ax

```

3.程序结果

```
DOSBox 0.74-2, Cpu speed: 3000 cycles, Frameskip 0, Program: LAB13
2 Warning Errors
0 Severe Errors

C:\>masm labin13;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

51484 +welcome to hujie! ol space free

0 Warning Errors
0 Severe Errors

C:\>link labin13;
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

LINK : warning L4021: no stack segment

C:\>labin13

C:\>lab13
```

2. 编写并安装int 7ch中断例程,功能为完成loop指令的功能

1.调用程序

[illegible]

```

18         mov ax,4c00h
19         int 21h
20
21 code ends
22 end start
23

```

2.安装程序

```

1  ;安装程序:中断向量号:7ch,中断向量: 0:200处
2  assume cs:code
3
4  code segment
5  start:
6      mov ax,cs
7      mov ds,ax
8      mov si,offset lp
9
10     mov ax,0
11     mov es,ax
12     mov di,200h           ;目的地址
13
14     mov cx,offset lpend-offset lp
15
16     cld
17
18     rep movsb
19
20     mov ax,0
21     mov es,ax
22     mov word ptr es:[7ch*4],200h
23     mov word ptr es:[7ch*4+2],0 ;设置中断向量
24
25     mov ax,4c00h
26     int 21h
27
28 lp:   push bp
29       mov bp,sp
30       dec cx
31       jcxz lpret
32       add [bp+2],bx         ;完成IP+base从而转到标号s处
33
34 lpret: pop bp
35       iret
36
37 lpend: nop
38
39 code ends
40 end start

```

3.程序结果

```
DOSBox 0.74-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>lab13_2
C:\>lab13_2
C:\>lab13_2
C:\>lab13_2
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>lab13_2
C:\>_
```

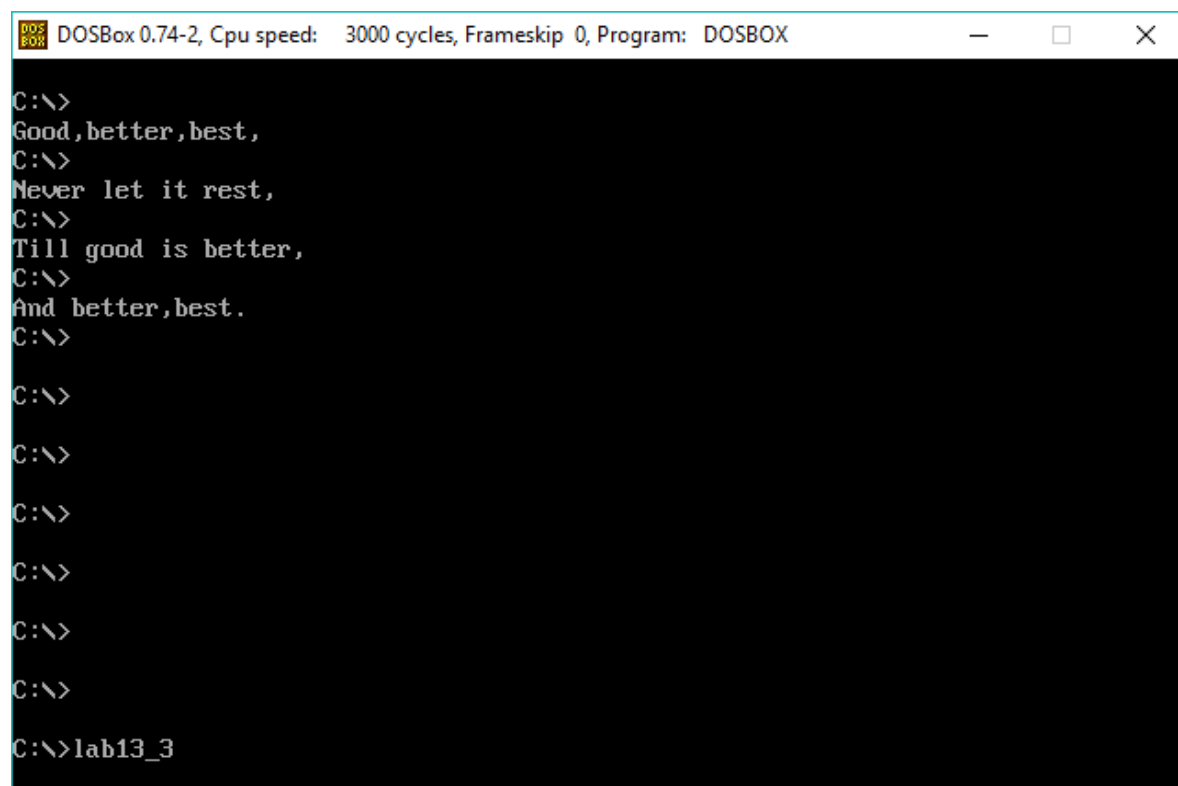
3. 分别在屏幕的第2,4,6,8行显示4句英文诗,补全程序

1.程序

```
1  assume cs:code
2  code segment
3      s1: db 'Good,better,best','','$'
4      s2: db 'Never let it rest','','$'
5      s3: db 'Till good is better','','$'
6      s4: db 'And better,best.','','$'
7      s : dw offset s1,offset s2,offset s3,offset s4
8      row:db 2,4,6,8
9
10     start: mov ax,cs
11             mov ds,ax
12             mov bx,offset s
13             mov si,offset row
14             mov cx,4
15     ok: mov bh,0           ;第0页
16             mov dh,ds:[si] ;行
17             mov dl,0       ;列
18             mov ah,2       ;置光标
19             int 10h
20
21             mov dx,ds:[bx] ;ds:dx指向字符串的首地址
22             mov ah,9       ;在光标位置处显示字符
23             int 21h
24             inc si         ;指向需要显示的下一行的行数的偏移地址
25             add bx,2       ;指向下一行字符串的首地址
26             loop ok
27
```

```
28      mov ax,4c00h
29      int 21h
30  code ends
31  end start
```

2.运行结果



```
DOSBox 0.74-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>
Good,better,best,
C:\>
Never let it rest,
C:\>
Till good is better,
C:\>
And better,best.
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>lab13_3
```

3.分析

由于是不同的行显示不同内容,故最好的策略是将行号与要内容分开,具体实现是

1. 将每行的内容分别放在标号也即该行的首地址中
2. 采用字大小,用标号记录其各行的标号
3. 采用字节大小(屏幕总共25行),用标号记录相应的行号
4. 将以上内容放入cs段的起始位置

Chapter 14

1. test 14.1

1.编程,读取CMOS RAM的2号单元的内容

```
1  mov al,2
2  out 70h,al ;往70h端口写入2
3  in al,71h ;从71h端口读入2号单元的一个字节(mov)
```

2.编程,向CMOS RAM的2号单元写入0

```

1  mov al,2
2  out 70h,al ;往70h端口写入2
3  mov al,0
4  out 71h,al ;往71h端口写入0字节

```

2. test 14.2

编程,用加法和移位指令计算 $(ax)=(ax)*10$

```

1  shl ax,1
2  mov bx,ax
3  mov cl,2
4  shl ax,cl
5  add bx,ax

```

lab 14

1. 编程,以"年/月/日 时:分:秒"的格式,显示当前的日期,时间.

1.编程思路

若单独显示不同的年 月 日 时 分 秒,则代码变得过于冗长和重复,考虑到代码的精简性,于是将其单元号都放入一个标号单元中,同理单独显示的字符也都放入一个标号中.考虑到程序的移植性,将其放入CS段中.

2.程序

```

1  assume cs:code
2
3  stack segment
4      db 16 dup (0)
5  stack ends
6
7  code segment
8      s: db 9,8,7,4,2,0 ;年 月 日 时 分 秒
9      s1:db 47,47,32,58,58,0 ;单独显示符号的ASCII码
10  start:
11      mov cx,6
12      mov ax,stack
13      mov ss,ax
14      mov sp,16
15
16      mov ax,cs
17      mov ds,ax
18      mov bp,offset s
19      mov di,offset s1
20      mov si,64
21
22  change: push cx
23      mov al,ds:[bp]
24      out 70h,al
25      in al,71h
26      mov ah,al
27      mov cl,4

```



```

28      shr ah,c1
29      and al,00001111b      ;转成两位十进制数
30
31      add ah,30h
32      add al,30h
33
34      mov bx,0b800h
35      mov es,bx
36      mov byte ptr es:[160*12+si],ah ;显示年 月 日 时 分 秒的十位数码
37      mov byte ptr es:[160*12+si+2],al;显示年 月 日 时 分 秒的个位数码
38      mov al,ds:[di]
39      mov byte ptr es:[160*12+si+4],al;显示单独字符
40
41      pop cx
42      inc bp
43      inc di
44      add si,6
45      loop change
46
47      mov ax,4c00h
48      int 21h
49
50 code ends
51 end start

```

3.运行结果

```

DOSBox 0.74-2, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>lab14
C:\>_

```

Chapter 15

1. 可屏蔽中断(外中断)

1. 取中断类型码n;(从数据总线送入CPU)
2. 标志寄存器入栈,IF=0,TF=0;
3. CS,IP入栈;
4. $(IP)=(n * 4)$, $(CS)=(n * 4 + 2)$

2. 不可屏蔽中断(外中断)

1. 标志寄存器入栈,IF=0,TF=0;
2. CS,IP入栈
3. $(IP)=(8)$, $(CS)=(0AH)$

3. 在BIOS键盘缓冲区中,一个键盘输入用一个字单元存放,高位字节存放扫描码,低位字节存放字符码

1. 键盘产生扫描码
2. 扫描码送入60h端口
3. 引发9号中断
4. CPU执行int 9中断例程处理键盘输入(这步能改变)

4. int过程的模拟过程变为

1. 标志寄存器入栈
2. IF=0,TF=0
3. `call dword ptr ds:[0]`

```
1  pushf                ;标志寄存器入栈
2
3  pushf
4  pop ax
5  and ah,11111100h
6  push ax
7  popf                 ;IF=0,TF=0
8
9  call dword ptr ds:[0];CS,IP入栈
```