如果你对习题答案有疑问，欢迎与作者白洪欢联系。作者电子邮件地址：iceman@zju.edu.cn。

习题

4.1　设有一段内存如下所示，请在以下每条指令的右边写出指令执行后目标操作数的值。

1000:20A0	12h
1000:20A1	34h
1000:20A2	56h
1000:20A3	78h
1000:20A4	9Ah
1000:20A5	0Bh

```
mov  ax, 1000h       ; AX=1000h
mov  ds, ax          ; DS=1000h
mov  al, ds:[20A0h] ; AL=12h
mov  ax, ds:[20A0h] ; AX=3412h
mov  dx, [20A1h]    ; DX=5634h
mov  cx, 20A1h       ; CX=20A1h
mov  ax, cx          ; AX=20A1h
mov  dh, ah          ; DH=20h
mov  bx, 20A0h       ; BX=20A0h
mov  ax, [bx]        ; AX=3412h
mov  dx, bx          ; DX=20A0h
mov  al, [bx+1]      ; AL=34h
mov  ah, [bx+3]      ; AH=78h
mov  si, 2           ; SI=2
mov  ax, [bx+si]     ; AX=7856h
mov  cx, [bx+si+2]  ; CX=0B9Ah
mov  dx, [bx+si-2]  ; DX=3412h
mov  di, 20A2h       ; DI=20A2h
mov  bp, [di-1]      ; BP=5634h
mov  byte  ptr [di-1], 12h    ; byte ptr [20A1h]=12h
mov  byte  ptr [20A0h], 34h  ; byte ptr [20A0h]=34h
mov  word  ptr [20A2h], 0B9Ah ; word ptr [20A2h]=0B9Ah
                              ; byte ptr [20A2h]=9Ah
                              ; byte ptr [20A3h]=0Bh
mov  word  ptr [20A4h], 7856h ; word ptr [20A4h]=7856h
                              ; byte ptr [20A4h]=56h
                              ; byte ptr [20A5h]=78h
```

4.2　试根据以下要求写出相应的汇编语言指令

(1) 把立即数1234h赋值给寄存器AX

```
mov ax, 1234h
```

(2) 把寄存器AH的值赋值给寄存器DL

```
mov dl, ah
```

(3) 把地址为2000:8F3Eh的内存字节赋值为0

```
mov ax, 2000h
mov ds, ax
mov byte ptr ds:[8F3Eh], 0
```

(4) 把地址为2000:8F3Eh的内存字赋值为1234h

```
mov ax, 2000h
mov ds, ax
mov word ptr ds:[8F3Eh], 0
```

(5) 把地址为2000:8F3Eh的内存字节赋值给寄存器CL

```
mov ax, 2000h
mov ds, ax
mov dl, ds:[8F3Eh]
```

(6) 把地址为2000:8F3Eh的内存字赋值给寄存器DI

```
mov ax, 2000h
mov ds, AX
mov di, ds:[8F3Eh]
```

4.3 设DS=2000h，BX=3F80h, SI=3F70h, DI=2。请分别使用[BX+位移]、[BX+DI+位移]、[SI+位移]这三种间接寻址方式把以下左图所示的内存单元值改变为右图所示的结果。要求用MOV指令来做，最多只用6条指令，且不得使用立即数作为操作数。

2000:3F80	12h
2000:3F81	34h
2000:3F82	56h
2000:3F83	78h

2000:3F80	78h
2000:3F81	56h
2000:3F82	34h
2000:3F83	12h

(1) 使用[BX+位移]

```
mov ax, [bx]     ; AX=3412h
mov dx, [bx+2]   ; DX=7856h
mov [bx+3], al   ; byte ptr ds:[3F83h]=12h
mov [bx+2], ah   ; byte ptr ds:[3F82h]=34h
mov [bx+1], dl   ; byte ptr ds:[3F81h]=56h
mov [bx], dh     ; byte ptr ds:[3F80h]=78h
```

(2) 使用[BX+DI+位移]

```
mov ax, [bx+di-2]; AX=3412h
mov dx, [bx+di]  ; DX=7856h
mov [bx+di+1], al; byte ptr ds:[3F83h]=12h
mov [bx+di], ah  ; byte ptr ds:[3F82h]=34h
mov [bx+di-1], dl; byte ptr ds:[3F81h]=56h
mov [bx+di-2], dh; byte ptr ds:[3F80h]=78h
```

(3) 使用[SI+位移]

```
mov ax, [si+10h] ; AX=3412h
mov dx, [si+12h] ; DX=7856h
mov [si+13h], al ; byte ptr ds:[3F83h]=12h
mov [si+12h], ah ; byte ptr ds:[3F82h]=34h
mov [si+11h], dl ; byte ptr ds:[3F81h]=56h
mov [si+10h], dh ; byte ptr ds:[3F80h]=78h
```

4.4 设有两段内存如下所示，且假定DS=1000h, SS=2000h, BX=8D90h, SI=2, DI=2，BP=8D90h。请问执行以下这段指令后，寄存器AX、BX、CX、DX的值分别等于多少？

1000:8D90	42h
1000:8D91	57h
1000:8D92	3Fh

2000:8D90	55h
2000:8D91	8Bh
2000:8D92	99h

1000:8D93	79h

2000:8D93	0Eh

```
        mov  ax, [bx]        ; AX=5742h
        mov  cx, [bp+si]     ; CX=0E99h
        mov  dx, ss:[bx+di] ; DX=0E99h
        mov  bx, ds:[bp+di] ; BX=793Fh
```

4.5 请指出以下指令的错误之处，并说明原因。

```
(1) mov  ax, bh      ; 源操作数与目标操作数类型不一致，AX是字，BH是字节
(2) mov  al, si      ; 源操作数与目标操作数类型不一致，AL是字, SI是字节
(3) mov  ds, 1000h   ; 段寄存器不能用立即数赋值
(4) mov  cs, ax      ; 代码段寄存器不能被赋值
(5) mov  byte ptr  ds:[10F0h], byte ptr  ds:[3F79h]
                      ; 源操作数与目标操作不能都为内存变量
(6)  mov  bx, ip     ; IP不可访问
(7)  mov  ah, [si+di]; 间接寻址方式表示的地址中只能包含一个变址寄存器
(8) mov  word  ptr  [bx+bp+2], 1234h
                              ; 间接寻址方式表示的地址只能包含一个基址寄存器
(9)  mov  byte  ptr  [bx-di+1], 1; 基址寄存器与变址寄存器只能相加，不能相减
```

4.6 设SS=2000h, SP=1000h，SI=1234h, DI=5678h, BX=9ABCh。试画出执行以下各条指令时堆栈指针SS:SP及堆栈内容的变化，并指出最后寄存器SI、DI、BX、SS、SP的值为多少。

```
        push  si
        push  di
        push  bx
        pop   bx
        pop   si
        pop   di
    SI=5678h
    DI=1234h
    BX=9ABCh
    SS=2000h
    SP=1000h
```

4.7 试根据以下要求写出相应的汇编语言指令。

(1) 交换寄存器AX与BX的值

```
    xchg ax, bx
```

(2) 从端口21h读取一个字节到寄存器AL

```
    in al, 21h
```

(3) 把寄存器AL的值写入端口378h

```
    mov dx, 378h
    out dx, al
```

(4) 设有一内存字X，X的偏移地址与段地址按顺序存放在从地址1000:10F0起的内存单元中，请写出汇编指令把X的值赋值给寄存器AX。(要求使用LDS指令)

```
        mov ax, 1000h
        mov ds, ax
        lds bx, dword ptr ds:[10F0h]
        mov ax, [bx]
```

(5) 设有一内存字节Y，Y的偏移地址与段地址按顺序存放在从地址1000:10F0起的内存单元中，请写出汇编指令把Y的值加1。(要求使用LES指令)

```
         mov ax, 1000h
         mov ds, ax
         les di, dword ptr ds:[10F0h]
         inc byte ptr es:[di]
```

4.8 设有如下所示的一段内存，并假定DS=1000h, BX=8FC0h, SI=1。请问执行以下两条指令后，寄存器AX与DX的值等于多少？

1000:8FC0	12h
1000:8FC1	34h
1000:8FC2	56h

```
mov  ax, [bx+si] ; AX=5634h
lea  dx, [bx+si] ; DX=8FC1h
```

4.9 设AL=7Fh，则执行指令CBW后，寄存器AX的值等于多少？

AX=007Fh

4.10 设AX=89BCh，则执行指令CWD后，寄存器AX与DX的值分别等于多少？

AX=89BCh

DX=0FFFFh

4.11 设有如下一段内存，并假定DS=1000h, BX=7FF9h, AL=3，则执行指令XLAT后，寄存器AL的值等于多少？

1000:7FF8	11h
1000:7FF9	22h
1000:7FFA	33h
1000:7FFB	44h
1000:7FFC	55h
1000:7FFD	66h

AL=55h

4.12 试根据以下要求写出相应的汇编语言指令

(1) 求1234h+5678h的和，要求结果存放在寄存器BX中

```
mov bx, 1234h
add bx, 5678h
```

(2) 求12-34的差，要求结果存放在寄存器CH中

```
mov ch, 12
sub ch, 34
```

(3) 求125*8的积，要求结果存放在寄存器AX中

```
mov al, 125
mov bl, 8
mul bl
```

(4) 求10000h/11h的商和余数，要求商存放在寄存器AX中，余数存放在寄存器DX中

```
mov dx, 1    ; DX=0001h
mov ax, 0    ; AX=0000h
mov bx, 11h
div bx
```

(5) 求12345678h+4243BCBCh的和，要求结果存放在寄存器DX:AX中

```
mov ax, 5678h
mov dx, 1234h
add ax, 0BCBCh
adc dx, 4243h
```

(6) 求56781234h-3F7DE980h的差，要求结果存放在寄存器DX:AX中

```
mov ax, 1234h
mov dx, 5678h
sub ax, 0E980h
```

```
    sbb dx, 3F7Dh
```

(7) 求1234h*5678h的积，要求结果存放在寄存器DX:AX中

```
    mov ax, 1234h
    mov bx, 5678h
    mul bx
```

(8) 求10000/99的商和余数，要求商存放在寄存器AL中，余数存放在寄存器AH中

```
    mov ax, 10000
    mov bl, 99
    div bl
```

4.13 试写出对存放在DX:AX中的32位数进行求补(neg)的汇编指令序列。

```
    neg ax      ; 若AX不等于0，则neg后会产生进位
    pushf       ; 保存进位标志CF
    neg dx
    popf        ; 恢复进位标志CF
    sbb dx, 0   ; 若在对AX进行neg时产生了进位，则必须从DX中扣除
```

4.14 若以下指令序列的操作数均为非符号数，请写出每个指令序列执行后标志位CF、ZF的值，并指出CMP指令的前后操作数的大小关系。

```
(1) mov  al, 80h
    cmp  al, 7Fh     ; CF=0, ZF=0, 80h > 7Fh
(2) mov  bx, 1
    cmp  bx, 0FFFFh ; CF=1, ZF=0, 1 < 0FFFFh
(3) mov  cx, 0FFF0h
    cmp  cx, 0FFFEh ; CF=1, ZF=0, 0FFF0 < 0FFFEh
(4) mov  dh, 60h
    cmp  dh, 7Ch     ; CF=1, ZF=0, 60h < 7Ch
```

4.15 若以下指令序列的操作数均为符号数，请写出每个指令序列执行后标志位CF、ZF、SF、OF的值，并指出CMP指令的前后操作数的大小关系。

```
(1) mov  al, 99h
    cmp  al, 34h     ; CF=0, ZF=0, SF=0, OF=1, 99h < 34h
(2) mov  ah, 81h
    cmp  ah, 0FFh    ; CF=1, ZF=0, SF=1, OF=0, 81h < 0FFh
(3) mov  bx, 1234h
    cmp  bx, 8086h   ; CF=1, ZF=0, SF=1, OF=1, 1234h > 8086h
(4) mov  cx, 0FFFFh
    cmp  cx, 0FFFEh ; CF=0, ZF=0, SF=0, OF=0, 0FFFFh > 0FFFEh
(5) mov  dx, 3F7Dh
    cmp  dx, 1000h   ; CF=0, ZF=0, SF=0, OF=0, 3F7Dh > 1000h
```

4.16 请写出执行以下指令序列后寄存器AX的值。

```
(1) mov  ax, 9
    add  al, 6
    daa          ; AX=0015h

(2) mov  ax, 72h
    add  al, 69h
    daa           ; AX=0041h

(3) mov  ax, 76h
    sub  al, 49h
    das           ; AX=0027h
(4) mov  ax, 0639h
    add  al, 33h
```

```
    aaa             ; AX=0702h
(5) mov  ax, 0702h
    sub  al, 3
    aas             ; AX=0609h
(6) mov  al, 6
    mov  cl, 9
    mul  cl
    aam             ; AX=0504h
(7) mov  ax, 0702h
    aad             ; AX=0048h
```

4.17 设AX=5A7Fh，请写出分别执行以下指令后寄存器AX的值是多少。

```
(1) or   ax,  1234h ; AX=5A7Fh
(2) and  ax,  5678h ; AX=5278h
(3) not  ax         ; AX=0A580h
(4) neg  ax         ; AX=0A581h
(5) xor  ax, 7F9Dh  ; AX=25E2h
(6) test ax, 6000h  ; AX=5A7Fh
```

4.18 设BX=0D75Ah, CL=2, CF=1, 请写出分别执行以下指令后寄存器BX和标志位CF的值是多少。

```
(1) shl bh, 1       ; BX=0AE5Ah, CF=1
(2) shr bx, cl      ; BX=35D6h, CF=1
(3) sar bx, cl      ; BX=0F5D6h, CF=1
(4) sal bl, 1       ; BX=0D7B4h, CF=0
(5) rol bx, cl      ; BX=5D6Bh, CF=1
(6) ror bx, 1       ; BX=6BADh, CF=0
(7) rcl bx, cl      ; BX=5D6Bh, CF=1
(8) rcr bh, 1       ; BX=0EB5Ah, CF=1
```

4.19 试写出把32位数3F7E59ACh逻辑左移2位的汇编指令序列，要求结果存放在DX:AX中。

```
    mov ax, 59ACh
    mov dx, 3F7Eh
    mov cx, 2
again:
    shl ax, 1
    rcl dx, 1
    loop again
```

4.20 试写出把32位数3F7E59ACh逻辑右移2位的汇编指令序列，要求结果存放在DX:AX中。

```
    mov ax, 59ACh
    mov dx, 3F7Eh
    mov cx, 2
again:
    shr dx, 1
    rcr ax, 1
    loop again
```

4.21 设从地址1000:10A0起存放了一个字符串，该字符串长度为100h字节。请编写一段程序按正方向把该字符串复制到从地址2000:20F0起的内存单元中。

```
    mov ax, 1000h
    mov ds, ax
    mov si, 10A0h
    mov ax, 2000h
    mov es, ax
    mov di, 20F0h
    mov cx, 100h
    cld
```

```
rep movsb
```

4.22 设从地址1000:10A0与2000:3BF0起，分别存放了一个长度为100h字节的字符串，请编写一段程序按正方向比较两个字符串是否完全相同，若相同则跳转到equal，若不相同则跳转到unequal。

```
mov ax, 1000h
mov ds, ax
mov si, 10A0h
mov ax, 2000h
mov es, ax
mov di, 3BF0h
mov cx, 100h
cld
repe cmpsb
je  equal
jmp unequal
```

4.23 设从地址4FA0:125B起存放了一个字符串，该字符串长度为100h字节。请编写一段程序在该字符串中按正方向查找空格，若找不到任何空格则转跳到not_found，否则把首次找到的空格的偏移地址赋值给寄存器BX。

```
mov ax, 4FA0h
mov es, ax
mov di, 125Bh
mov cx, 100h
mov al, ' '
cld
repne scasb
jne not_found
dec di
mov bx, di
```

4.24 设从地址3F80:0000起存放了一个字符串，该字符串以'$'结束。请编写一段程序计算该字符串的长度(不包括结束符'$')并赋值给寄存器CX。

```
mov ax, 3F80h
mov es, ax
mov di, 0
mov cx, 0FFFFh
mov al, '$'
cld
repne scasb
not cx       ; CX = 0FFFFh - CX
dec cx
```

4.25 试编写一段程序，把从地址8000:12F0开始的8000h个内存单元赋值为0。

```
mov ax, 8000h
mov es, ax
mov di, 12F0h
mov cx, 8000h
mov al, 0
cld
rep stosb
```

4.26 设从地址2B7C:1080开始存放了一个长度为100h字节的字符串，该字符串含有至少一个'$'符。请编写一段程序找出字符串中的最后一个'$'符，并把该字符的偏移地址赋值给寄存器BX。

```
mov ax, 2B7Ch
mov es, ax
```

```
    mov di, 1080h+100h-1; DI=117Fh
    mov cx, 100h
    mov al, '$'
    std
    repne scasb
    inc di
    mov bx, di
```

4.27 设从地址2B7C:1080开始存放了一个长度为100h字节的字符串，该字符串含有至少一个'$'符。请编写一段程序把该字符串中的所有'$'符都替换成空格。

```
    mov ax, 2B7Ch
    mov es, ax
    mov di, 1080h
    mov cx, 100h
    mov al, '$'
    cld
next:
    jcxz done
    repne scasb
    jne done
    mov byte ptr es:[di-1], ' '
    jmp next
done:
```

4.28 设CS=2000h, IP=10A0h，并且从地址2000:10A0起存放了跳转指令的机器码，这些机器码是以下五种情况之一。试准对这五种不同情况，分析一下执行本条指令后，CS与IP的值将等于多少。(提示：0EBh为短跳，0E9h为近跳，0EAh为远跳)

(1) 0EBh 9Ch
(2) 0EBh 70h
(3) 0E9h 80h 70h
(4) 0E9h 70h 0FFh
(5) 0EAh 9Dh 5Fh 00h 10h

4.29 假定寄存器AX与DX存放的值为非符号数，SI与DI存放的值为符号数，请用比较指令和条件跳转指令完成以下判断。

(1) 若AX 大于 DX则跳转至above

```
    cmp ax, dx
    ja above
```

(2) 若DX 小于或等于 AX则跳转至below_equal

```
    cmp dx, ax
    jbe below_equal
```

(3) 若AX 等于 DX则跳转至equal

```
    cmp ax, dx
    je equal
```

(4) 若AX 大于或等于 DX则跳转至above_equal

```
    cmp ax, dx
    jae above_equal
```

(4) 若CX 等于 0则跳转至CX_is_zero

```
    jcxz CX_is_zero
```

(5) 若SI 小于 DI则跳转至less

```
    cmp si, di
    jl less
```

(6) 若SI 小于或等于 DI则跳转至less_euqal

```
cmp si, di
jle less_equal
```

(7) 若SI 大于 DI则跳转至greater

```
cmp si, di
jg greater
```

(8) 若SI 大于或等于 DI则跳转至greater_equal

```
cmp si, di
jge greater_equal
```

(9) 若SI 等于 DI 则跳转至equal

```
cmp si, di
je equal
```

(10)比较SI与DI，若产生溢出则跳转至overflow

```
cmp si, di
jo overflow
```

(11)比较AX与DX，若产生符号位则跳转至negative

```
cmp ax, dx
js negative
```

4.30 试编写一段程序完成1+2+3...+100的计算，要求使用LOOP指令来做，结果存放到寄存器AX中。

```
    xor ax, ax ; AX=0
    mov cx, 100
again:
    add ax, cx
    loop again
```

4.31 请问以下这段程序执行完后，寄存器AX、BX、CX的值等于多少？

```
    mov  ax, 0FFFBh
    mov  bx, 0FFFFh
    mov  cx, 000Ah
step1:
    add  bx, ax
    inc  ax
    loopnz  step1
step2:              ; AX=0, BX=0FFF0h, CX=5
    inc  ax
    shr  bx, 1
    test bx, 1
    loopz  step2
step3:              ; AX=4, BX=0FFFh, CX=1
    inc  bx
    loop  step3
done:
```

运行结果：AX=4, BX=1000h, CX=0

4.32 设有如下所示的一段程序，并假定CS=1210h，IP=0000h, SS=1210h, SP=0FFFEh。试分析这段程序的执行过程，并画出在程序执行过程中堆栈指针SS:SP及堆栈内容的变化，最后写出程序终止时寄存器BX、CX、DX的值。

```
1210:0000 BB0000        mov bx, 0000
1210:0003 B90300        mov cx, 0003
1210:0006 BA0100        mov dx, 0001
1210:0009 E80400        call    0010h; ①
```

```
1210:000C B44C          mov ah, 4Ch
1210:000E CD21          int 21h       ; 程序终止时，BX=6，CX=0，DX=4
1210:0010 03DA          add bx, dx    ; BX=1, BX=3, BX=6
1210:0012 E80300        call     0018h; ② ④ ⑥
1210:0015 E2F9          loop     0010h; CX=2, CX=1, CX=0
1210:0017 C3            ret           ; ⑧
1210:0018 42            inc dx        ; DX=2, DX=3, DX=4
1210:0019 C3            ret           ; ③ ⑤ ⑦
```

程序执行过程中堆栈的变化如下(数字表示执行该步之后)：

1210:FFF8	??h	
1210:FFF9	??h	
1210:FFFA	15h	←SS:SP ② ④ ⑥
1210:FFFB	00h	
1210:FFFC	0Ch	←SS:SP ① ③ ⑤ ⑦
1210:FFFD	00h	
1210:FFFE	??h	←SS:SP ⑧
1210:FFFF	??h	

4.33 设有如下所示的一段内存，当执行中断指令INT 16h后，寄存器CS与IP的值将等于多少？

0000:0054	40h
0000:0055	02h
0000:0056	58h
0000:0057	02h
0000:0058	2Dh
0000:0059	04h
0000:005A	70h
0000:005B	00h

先计算int 16h中断向量所在的位置 = 0000:16h*4 = 0000:0058h。根据上图，从该地址起连续存放了4个字节：2Dh, 04h, 70h, 00h。这4个字节中的前面2个字节构成一个字042Dh，后面2个字节也构成一个 字 0070h 。这两个字中前面一个是偏移地址，后面一个是段地址，合在一起构成一个远指针：0070h:042Dh。这正是int 16h的中断向量。所以当执行int 16h之后，IP等于042Dh，CS等于0070h。

4.34 设有如下一段内存，且假定SS=2000h，SP=0FFF0h，则执行指令RETF后，寄存器CS与IP的值将等于多少？

2000:FFEC	06h
2000:FFED	7Bh
2000:FFEE	8Fh
2000:FFEF	30h
2000:FFF0	0Ah
2000:FFF1	00h
2000:FFF2	10h
2000:FFF3	12h

执行指令RETF后，IP=000Ah, CS=1210h。

4.35 设有如下一段内存，且假定SS=3000h，SP=0FFF8h，则执行指令IRET后，寄存器CS与IP的值将等于多少？标志位CF的值又是多少？

3000:FFF6	06h
3000:FFF7	8Dh
3000:FFF8	80h
3000:FFF9	35h
3000:FFFA	27h
3000:FFFB	18h
3000:FFFC	03h
3000:FFFD	72h

执行指令IRET后，IP=3580h, CS=1827h, CF=1。因为执行IRET后将分别从堆栈中弹出IP、CS、FL，所以IP=3580h，CS=1827h，FL=7203h。CF在FL中是第0位，所以CF=1。

习题

5.1 请指出以下变量名中哪些是不正确的，并说明错误原因。

```
(1) aaa          ; 正确
(2) bbb          ; 正确
(3) VIDEO-3D     ; 不正确。变量名中不能含有减号。
(4) It_is_ok!    ; 不正确。变量名中不能含有感叹号。
(5) ?IsItRight   ; 正确
(6) 2Small       ; 不正确。变量名不能用数字开头。
(7) MP2$MP3@MP4  ; 正确
(8) FFFFh        ; 正确
```

5.2 请指出以下变量或数组定义中哪些是不正确的，并说明错误原因。

(1) `xyz db 12h, 34h, 1234h`
不正确。db规定每一项必须为字节，但1234h超过0FFh。

(2) `abc dw 1, 23h, 4, 'A', 'B', 'C', 1234h, 0FFFFh`
正确。1可以看作是0001h，'A'可以看作是0041h。

(3) `big dd 12345678h, 1234h, 5678h, FFFFFFFFh`
不正确。16进制数如果以字母开头，则必须加前缀0。

(4) `top db -1, 10001001B, 8Fh, 178Q`
不正确。八进制数中不能含有大于7的位。

(5) `pot dw (10+5)/2, NOT 1, (3 OR 4) XOR 0FFh, 7Fh AND (80h-3)`
正确。

(6) `yes dw $ - 100h`
正确。

5.3 用16进制形式写出以下常量表达式的值

(1) `(10/3)*3`
9h

(2) `(5 MOD 3) OR (8 XOR 6)`
0Eh

(3) `(0FFh SHR 3) SHL 2`
7Ch

(4) `(-10) AND (7F8Dh-8000h)`
0FF84h

(5) `6651h XOR (NOT (1-3)) XOR 6651h`
0001h

5.4 请指出以下程序中不正确的语句，并说明错误原因。

```
data  segment
abc   db  4, 3, 2, 1
      db  'ABCD'
```

```
xyz   dw  1234h, 5678h, 0, 0FFFFh
data  ends
code  segment
assume  cs:code, ds:data
main:
    mov  ax, data
    mov  ds, ax
    mov  ah, abc
    add  ah, abc+2   ; 正确。相当于add ah, [abc+2]。
    mov  al, abc*4   ; abc不是一个常量，所以不可以使用*运算符
    mov  si, [abc+2] ; [abc+2]的类型是字节，不是字，与SI不一致。
    mov ch, abc[4]  ; 正确。CH='A'。
    mov cl, [xyz+2] ; [xyz+2]的类型是字，不是字节，与CL不一致。
    mov  dx, xyz[1]  ; 正确。DX=7812h。
    mov  ah, 4Ch
    int  21h
code  ends
end   main
```

5.5　定义一个数据段data如下所示，并假定data段的段址为1000h。请根据data中的变量和数组定义，用16进制形式写出从地址1000:0000到1000:0013之间共20个内存单元的值。

```
data  segment
ONE   equ  1
TWO   equ  (ONE+ONE)
abc   db  'ABC', 2 dup ('D'), "EF"
      db  ONE, TWO, ONE+TWO
len   =  $ - offset abc
xyz   dw  offset xyz, seg  xyz
      dw  abc, data, len
data  ends
```

地址	值	解释
1000:0000	41h	'A'
1000:0001	42h	'B'
1000:0002	43h	'C'
1000:0003	44h	'D'
1000:0004	44h	'D'
1000:0005	45h	'E'
1000:0006	46h	'F'
1000:0007	01h	ONE
1000:0008	02h	TWO
1000:0009	03h	ONE+TWO
1000:000A	0Ah	offset xyz
1000:000B	00h	
1000:000C	00h	seg xyz
1000:000D	10h	
1000:000E	00h	abc即offset abc
1000:000F	00h	
1000:0010	00h	data
1000:0011	10h	
1000:0012	0Ah	len
1000:0013	00h	

5.6　以下程序中，假设data段址为120Fh。请用16进制形式在code段的每条MOV指令的右边注明指令执行后各个寄存器的值。（允许使用DEBUG进行调试）

```
data  segment
```

```
abc   db  1, 2, 3
xyz   dw  10, 20h, 30
data  ends
code  segment
assume  cs:code, ds:data
main:
    mov  ax, data
    mov  ds, ax
    mov  ah, [abc+1]
    mov  al, abc
    mov  bh, abc+2
    mov  bl, abc[3]
    mov  ax, [xyz]
    mov  bx, xyz+2
    mov  cx, xyz[1]
    mov  dx, [xyz-2]
    mov  si, word ptr [abc+1]
    mov  ah, byte ptr [xyz+4]
    mov  al, byte ptr xyz[3]
    mov  ah, 4Ch
    int  21h
code  ends
end   main
```

5.7 以下程序中，假设data段址为120Fh，extr段址为1210h。请用16进制形式在code段的每条MOV指令的右边注明指令执行后各个寄存器的值。（允许使用DEBUG进行调试）

```
data  segment
abc   db  1, 2, 3
xyz   dw  3344h, 5566h, 7788h
data  ends
extr  segment
bee   db  4, 5, 6
see   dw  1234h, 5678h, 9ABCh
extr  ends
code  segment
assume  cs:code, ds:data, es:extr
main:
    mov  ax, data  ; AX=120Fh
    mov  ds, ax    ; DS=120Fh
    mov  ax, extr  ; AX=1210h。在默认情况下，段长度总是等于10h的倍数，
                   ; 所以，data段与extr段的实际长度尽管是9字节，但都
                   ; 自动扩充为10h字节。
    mov  es, ax    ; ES=1210h
    mov  bx, offset abc ; BX=0000h
    mov  ah, [bx]       ; AH=01h
    mov  al, [bx+2]     ; AL=03h
    mov  cx, [bx+1]     ; CX=0302h
    mov  bx, [bx+3]     ; BX=3344h
    mov  si, 2          ; SI=0002h
    mov  dx, xyz[si]    ; DX=5566h
    mov  bp, [xyz+si+2] ; BP=7788h
    mov  di, offset see ; DI=0003h
    mov  ah, es:[di-1]  ; AH=06h
    mov  bx, es:[di+4]  ; BX=9ABCh
    mov  cx, [di+4]     ; CX=7788h。注意这里的默认段址是DS。
    mov  ah, 4Ch        ; AH=4Ch
    int  21h
code  ends
```

```
end   main
```

5.8 阅读以下程序，以16进制形式写出程序终止时字类型数组xyz中四个元素的值。（允许使用DEBUG进行调试）

```
data  segment
xyz   dw  1234h, 5678h, 9ABCh, 0FFFFh
table  dw  offset here, offset exit
       dw  offset xyz,  data
data  ends
code  segment
assume  cs:code, ds:data
main:
    mov  ax, data
    mov  ds, ax
    mov  ax, [xyz]        ; AX=1234h
    xchg ax, [xyz+2]      ; AX=5678h, xyz[2]=1234h
    mov  [xyz], ax        ; [xyz]=5678h
    mov  bx, offset table
    inc  word  ptr  [bx-2]; xyz[6]=0000h
    jz   there            ; ➔ there
here:
    les  di, dword  ptr  table[4] ; DI=offset xyz, ES=data
    not  word  ptr  es:[di+4]     ; xyz[4]=6543h
    mov  di, es:[di]              ; DI=[xyz]=5678h
    xor  word  ptr xyz[2], di     ; xyz[2]=1234h xor 5678h=444Ch
    mov  bp, table[2]             ; BP=offset exit
    jmp  bp                       ; ➔ exit
there:
    mov  ax, [bx-4]       ; AX=9ABCh
    sub  [bx-2], ax       ; xyz[6]=0000h-9ABCh=6544h
    jmp  [table]          ; ➔ here
exit:
    mov  ah, 4Ch
    int  21h
code  ends
end   main
```

程序终止时，xyz[0]=5678h, xyz[2]=444Ch, xyz[4]=6543h, xyz[6]=6544h

5.9 编写一个程序，在屏幕上输出“Hello,world!”及回车。要求程序中定义三个段：数据段、代码段、堆栈段。

```
data segment
hi db 'Hello,world!', 0Dh, 0Ah, '$'
data ends
code segment
assume cs:code, ds:data, ss:stk
main:
    mov ax, data
    mov ds, ax
    mov ah, 9
    mov dx, offset hi
    int 21h
    mov ah, 4Ch
    int 21h
code ends
stk segment stack
dw 100h dup(0)
stk ends
```

```
end main
```

5.10 编写一个程序，在屏幕上输出“Hello,world!”及回车。要求程序中只定义一个段：代码段。

```
code segment
assume cs:code, ds:code
main:
   push cs
    pop  ds  ; DS=CS
    mov ah, 9
    mov dx, offset hi
    int 21h
    mov ah, 4Ch
    int 21h
hi db 'Hello,world!', 0Dh, 0Ah, '$'
code ends
end main
```

5.11 编写一个程序，在屏幕上输出“Hello,world!”及回车20遍。要求数据段中只定义一个字符串“Hello,world!”。

```
data segment
hi db 'Hello,world!', 0Dh, 0Ah, '$'
data ends
code segment
assume cs:code, ds:data
main:
    mov ax, data
    mov ds, ax
    mov cx, 20
again:
    mov ah, 9
    mov dx, offset hi
    int 21h
   loop again
    mov ah, 4Ch
    int 21h
code ends
end main
```

5.12 编写一个程序，从键盘输入一个字符，然后在屏幕上输出该字符60遍。(提示：输入一个字符可调用INT 21h中断的01h号功能，输入的字符在调用INT 21h/AH=01h后自动返回到寄存器AL中；输出一个字符可调用INT 21h的02h号功能，待输出的字符要求存放在寄存器DL中。)

```
code segment
assume cs:code, ds:code
main:
    mov ah, 1
    int 21h    ; 输入一个字符到AL
    mov dl, al ; DL=AL=输入的字符
    mov cx, 60 ; 循环次数
again:
    mov ah, 2
    int 21h    ; 输出DL中的字符
   loop again
    mov ah, 4Ch
    int 21h
code ends
end main
```

5.13 已定义一个数据段如下所示，请编写一个代码段，要求把数据段中所定义的字符串按逆序输出，结果应显示“A quick brown fox jumps over the lazy dog.”。

```
data  segment
abc   db  '.god yzal eht revo spmuj xof nworb kciuq A'
len   =  $ - offset abc
data  ends
code segment
assume cs:code, ds:data
main:
    mov ax, data
    mov ds, ax
    mov si, offset abc
    add si, len
    dec si            ; 以上三句也可写成一句：mov si, offset abc+len-1
                      ; DS:SI → 数组abc中的最后一个字节
    mov cx, len      ; CX=字符串长度
again:
    mov ah, 2
    mov dl, [si]
    int 21h
    dec si
   loop again
    mov ah, 4Ch
    int 21h
code ends
end main
```

5.14 已定义data段与extra段如下所示，请编写一个代码段，要求把data段中所定义的字符数组source复制到extra段中的字符数组destination中。

```
data  segment
source  db  'From source to destination!',0
len     =   $ - offset source
data  ends
extra  segment
destination  db  len  dup(?)
extra  ends
code segment
assume cs:code, ds:data, es:extra
main:
   mov ax, data
   mov ds, ax
   mov ax, extra
   mov es, ax
   mov si, offset source
   mov di, offset destination
   mov cx, len
   cld
   rep movsb
   mov ah, 4Ch
   int 21h
code ends
end main
```

5.15 编写一个程序，以10进制格式输入两个个位数，输出这两个个位数之和。

```
code segment
assume cs:code, ds:code
```

```
main:
    mov cx, 0       ; 输入次数
input:
    mov ah, 1
    int 21h         ; 输入一个字符 → AL
    cmp al, '0'     ; 若输入字符 < '0', 则
    jb input        ; 重新输入
    cmp al, '9'     ; 若输入字符 > '9', 则
    ja input        ; 重新输入
    sub al, '0'     ; 把字符转化为数值，如'3' → 3
    inc cx          ; 输入次数加1
    cmp cx, 1       ; 若是第一次输入，则
    je first_number ; 输入的是被加数
second_number:      ; 否则，输入的是加数
    mov bh, al      ; 把加数保存到BH中
    jmp add_two_numbers
first_number:
    mov bl, al      ; 把被加数保存到BL中
    jmp input       ; 再次输入
add_two_numbers:
    add bl, bh      ; 两数相加
    cmp bl, 10      ; 若和小于10，则
    jb  less_than_10 ; 直接输出个位
    mov ah, 2       ; 否则，先输出十位数'1'
    mov dl, '1'
    int 21h         ; 输出十位数'1'
    sub bl, 10      ; 把和减去10，剩余个位
less_than_10:
    mov ah, 2
    mov dl, bl
    add dl, '0'     ; 把个位数转化成字符，如3 → '3'
    int 21h         ; 输出个位数
    mov ah, 4Ch
    int 21h
code ends
end main
```

5.16 编写一个程序，以10进制格式输入两个正整数，输出这两个数的和、差、积、商。

```
data segment
buf db 6, 0, 6 dup(?) ; 输入缓冲区
x   dw ?                ; 第一个数
y   dw ?                ; 第二个数
big dd 100000           ; 大数
out_buf db 10 dup(?), '$' ; 输出缓冲区
incorrect_msg db 'Incorrect input!', 7, 0Dh, 0Ah, '$'
div_overflow_msg db 'Divided by zero!', 7, 0Dh, 0Ah, '$'
ellipses_msg db '...$'
data ends

code segment
assume cs:code, ds:data, es:data
main:
    mov ax, data
```

```
    mov ds, ax
    mov es, ax
    cld
input1:
    mov si, offset buf
    mov di, offset x
    call input              ; 输入第一个数，存放到x中
    jnc input2
    jmp incorrect_input
input2:
    mov si, offset buf
    mov di, offset y
    call input              ; 输入第二个数，存放到y中
    jnc add_two_numbers
    jmp incorrect_input
add_two_numbers:
    xor ax, ax
    xor dx, dx
    mov ax, [x]
    add ax, [y]
    adc dx, 0               ; 两数相加
    call output             ; 输出两数之和
    call crlf               ; 回车换行
sub_two_numbers:
    xor ax, ax
    xor dx, dx
    mov ax, [x]
    sub ax, [y]
    sbb dx, 0               ; 两数相减
    jnc positive            ; 若没有进位则结果为正数
negative:                   ; 否则结果为负数
    push ax
    push dx
    mov ah, 2
    mov dl, '-'
    int 21h                 ; 显示负号
    pop dx
    pop ax
    neg ax
    pushf
    neg dx
    popf
    sbb dx, 0               ; 求负数的相反数
    call output
    call crlf
    jmp mul_two_numbers
positive:
    call output
    call crlf
mul_two_numbers:
    xor ax, ax
    xor dx, dx
    mov ax, [x]
    mul [y]                 ; 两数相乘
    call output
    call crlf
```

```
div_two_numbers:
    xor ax, ax
    xor dx, dx
    mov ax, [x]
    mov bx, [y]
    or bx, bx
    jz div_overflow            ; 若除数为0则除法溢出
    div bx                      ; 两数相除
    push dx
    xor dx, dx
    call output                 ; 输出商
    mov ah, 9
    mov dx, offset ellipses_msg
    int 21h                     ; 输出省略号
    pop dx
    mov ax, dx
    xor dx, dx
    call output                 ; 输出余数
    call crlf
    jmp exit
div_overflow:
    mov ah, 9
    mov dx, offset div_overflow_msg
    int 21h
    jmp exit
incorrect_input:
    mov ah, 9
    mov dx, offset incorrect_msg
    int 21h
exit:
    mov ah, 4Ch
    int 21h

crlf proc near
    push ax
    push dx
    mov ah, 2
    mov dl, 0Dh
    int 21h
    mov ah, 2
    mov dl, 0Ah
    int 21h
    pop dx
    pop ax
    ret
crlf endp

input proc near
    push ax
    push bx
    push cx
    push dx
    push si
    push di
    push bp
input_again:
    mov ah, 0Ah
```

```
    mov dx, si
    int 21h                     ; 输入一行字符
    call crlf
    xor cx, cx
    mov cl, [si+1]
    jcxz input_again          ; 若输入字符个数为0则重新输入
    add si, 2                   ; SI → 输入的首个字符
    xor ax, ax
    xor dx, dx
    xor bx, bx
    mov bp, 10
calculate_number:
    mul bp
    mov bl, [si]
    cmp bl, '0'
    jb  invalid_number
    cmp bl, '9'
    ja  invalid_number
    sub bl, '0'
    add ax, bx
    adc dx, 0
    jnz number_overflow     ; 若输入的数超过16位最大值(65535)则溢出
    inc si
    loop calculate_number    ; 把字符串转化成数值
    mov [di], ax                ; 保存该数
    clc
    jmp input_done
invalid_number:
number_overflow:
    mov word ptr [di], 0
    stc
input_done:
    pop bp
    pop di
    pop si
    pop dx
    pop cx
    pop bx
    pop ax
    ret
input endp

output proc near
    push ax
    push bx
    push cx
    push dx
    push si
    push di
    push bp
    mov di, offset out_buf
    mov bx, word ptr [big]
    mov cx, word ptr [big+2] ; CX:BX=100000
    cmp dx, cx
    ja  big_number
    jb  small_number
```

```
    cmp ax, bx
    jae big_number              ; 若大于等于100000则进行大数处理
    jmp small_number
big_number:
    xor bp, bp
div_big_number:
    sub ax, bx
    sbb dx, cx
    jc  adjust
    inc bp
    jmp div_big_number          ; 计算100000的倍数
adjust:
    add ax, bx
    adc dx, cx
    push ax
    push dx
    xor dx, dx
    mov ax, bp                      ; 先把10万的倍数转化成字符串
    mov si, 1                       ; 字符串前不需添'0'
    call separate_combine_digit ; 把数值转化成字符串
    pop dx
    pop ax                           ; 再把余下的值转化成字符串
    mov si, 0                       ; 若少于5位则字符串前需添'0'
    call separate_combine_digit  ; 把数值转化成字符串
    jmp output_done
small_number:
    mov si, 1
    call separate_combine_digit
output_done:
    mov byte ptr [di], '$'         ; 字符串后面补'$'
    mov ah, 9
    mov dx, offset out_buf
    int 21h                          ; 输出结果
    pop bp
    pop di
    pop si
    pop dx
    pop cx
    pop bx
    pop ax
    ret
output endp

separate_combine_digit proc near
    push ax
    push bx
    push cx
    push dx
    push si
    push bp
    mov bp, 10
    xor cx, cx
separate_digit:
    div bp
    add dl, '0'
    push dx
```

```
    inc cx
    xor dx, dx
    or ax, ax
    jnz separate_digit
    or si, si
    jnz combine_digit
    push cx
    sub cx, 5
    neg cx
    mov al, '0'
    rep stosb
    pop  cx
combine_digit:
    pop ax
    stosb
    loop combine_digit
    pop bp
    pop si
    pop dx
    pop cx
    pop bx
    pop ax
    ret
separate_combine_digit endp

code ends
end main
```

5.17 编写一个程序，输出ASCII码从20h到0FFh之间的所有ASCII字符以及它们的二进制、十进制、十六进制值。

```
code segment
assume cs:code, ds:code
main:
    mov cx, 0FFh-20h+1  ; 字符个数
    mov al, 20h           ; 从20h开始
again:
    call ascii            ; 显示ASCII码对应的字符
    call space            ; 显示空格
    call binary           ; 显示二进制值
    call space
    call decimal          ; 显示十进制值
    call space
    call hex               ; 显示十六进制值
    call crlf              ; 回车换行
    inc al                  ; ASCII码加1
    loop again
    mov ah, 4Ch
    int 21h

crlf proc near
    push ax
    push dx
    mov ah, 2
    mov dl, 0Dh
    int 21h
    mov ah, 2
```

```
    mov dl, 0Ah
    int 21h
    pop dx
    pop ax
    ret
crlf endp

space proc near
    push ax
    push dx
    mov ah, 2
    mov dl, ' '
    int 21h
    pop dx
    pop ax
    ret
space endp

ascii proc near
    push ax
    push dx
    mov ah, 2
    mov dl, al
    int 21h
    pop dx
    pop ax
    ret
ascii endp


binary proc near
    push ax
    push  cx
    push dx
    mov cx, 8
binary_next:
    shl al, 1               ; 左移一位
    jc show_1               ; 若产生进位则此位必定是1
show_0:
    mov dl, '0'             ; 否则此位是0
    jmp show_this_digit
show_1:
    mov dl, '1'
show_this_digit:
    push ax
    mov ah, 2
    int 21h
    pop ax
    loop binary_next
    pop dx
    pop cx
    pop ax
    ret
binary endp

decimal proc near
    push ax
```

```
    push bx
    push cx
    push dx
    xor cx, cx
    mov bl, 10
decimal_separate_digit:
    mov ah, 0
    div bl
    mov dl, ah
    add dl, '0'
    push dx
    inc cx
    or al, al
    jnz decimal_separate_digit
decimal_combine_digit:
    pop dx
    mov ah, 2
    int 21h
    loop decimal_combine_digit
    pop dx
    pop cx
    pop bx
    pop ax
    ret
decimal endp

hex proc near
    push ax
    push bx
    push cx
    push dx
    mov cx, 2
hex_next:
    push cx
    mov cl, 4
    rol al, cl           ; 循环左移4位，高4位移与低4位交换位置
    push ax
    and al, 0Fh          ; 去掉高4位
    cmp al, 10           ; 若小于10
    jb  less_than_10    ; 则必定在0到9之间，只需直接加'0'就转化成字符
    sub al, 10           ; 否则，必定在10到15之间，只需先减去10
    add al, 'A'          ; 再加上'A'，就可以转化成'A'到'F'之间的字符
    jmp hex_show_digit
less_than_10:
    add al, '0'
hex_show_digit:
    mov ah, 2
    mov dl, al
    int 21h
    pop ax
    pop cx
    loop hex_next
    pop dx
    pop cx
    pop bx
    pop ax
```

```
    ret
hex endp

code ends
end main
```