钱晓捷新版汇编语言程序设计习题答案

第一章　汇编语言基础知识

1.17、举例说明 CF和 OF标志的差异。

溢出标志 OF 和进位标志 CF 是两个意义不同的标志

进位标志表示无符号数运算结果是否超出范围，运算结果仍然正确；溢出标志表示有符号数运算结果是否超出范围，运算结果已经不正确□例 1：3AH＋7CH＝B6H

无符号数运算：58＋124＝182，范围内，无进位

有符号数运算：　58+124＝182，范围外，有溢出□例 2：AAH＋7CH＝（1）26H

无符号数运算：170＋124＝294，范围外，有进位

有符号数运算：－86＋124＝28，范围内，无溢出

1.20、8086 有哪 4 种逻辑段，各种逻辑段分别是什么用途？（解答）

代码段（Code Segment）用来存放程序的指令序列。处理器利用 CS : IP 取得下一条要执行的指令

□堆栈段（Stack Segment）确定堆栈所在的主存区域。处理器利用 SS : SP 操作堆栈中的数据

□数据段（Data Segment）存放当前运行程序所用的数据。处理器利用 DS : EA 存取数据段中的数据

□附加段（Extra Segment）是附加的数据段，也用于数据的保存。处理器利用 ES : EA 存取数据段中的数据

第二章 8086 指令系统

2.1 已知 DS ＝ 2000H、BX＝0100H、SI＝0002H，存储单元[20100H]～[20103H]依次存放 12 34 56 78H，[21200H]～[21203H]依次存放 2A 4C B7 65H，说明下列每条指令执行完后 AX 寄存器的内容。

```
（1） mov ax,1200h                ;AX＝1200h
（2） mov ax,bx                   ; AX＝0100h
（3） mov ax,[1200h]              ; AX＝4C2Ah
（4） mov ax,[bx]                 ; AX＝3412h
（5） mov ax,[bx+1100h]           ; AX＝4C2Ah
（6） mov ax,[bx+si]              ; AX＝7856h
（7） mov ax,[bx][si+1100h]       ; AX＝65B7h
```

2.2 指出下列指令的错误

(1)　mov cx,dl　　两操作数类型不匹配

(2) mov ip,ax IP　　指令指针禁止用户访问

(3) mov es,1234h　　立即数不允许传给段寄存器

(4) mov es,ds　　段寄存器之间不允许传送

(5) mov al,300　　两操作数类型不匹配

(6) mov [sp],ax　　目的操作数应为[BP]

(7) mov ax,bx+di　　源操作数应为 [BX+DI]

(8) mov 20h,ah　　立即数不能作目的操作数

2.3 已知数字 0 ~ 9对应的格雷码依次为：18H、34H、05H、06H、09H、0AH、0CH、11H、12H、14H，它存在于以 table 为首地址（设为 200H）的连续区域中。请为如下程序段的每条指令加上注释，说明每条指令的功能和执行结果。

lea bx,table　　；获取 table 的首地址，BX=200H

mov al,8　；传送欲转换的数字，AL=8

xlat　；转换为格雷码，AL=12H

2.4 什么是堆栈，它的工作原则是什么，它的基本操作有哪两个，对应哪两种指令？

堆栈是一种按“先进后出”原则存取数据的存储区域。

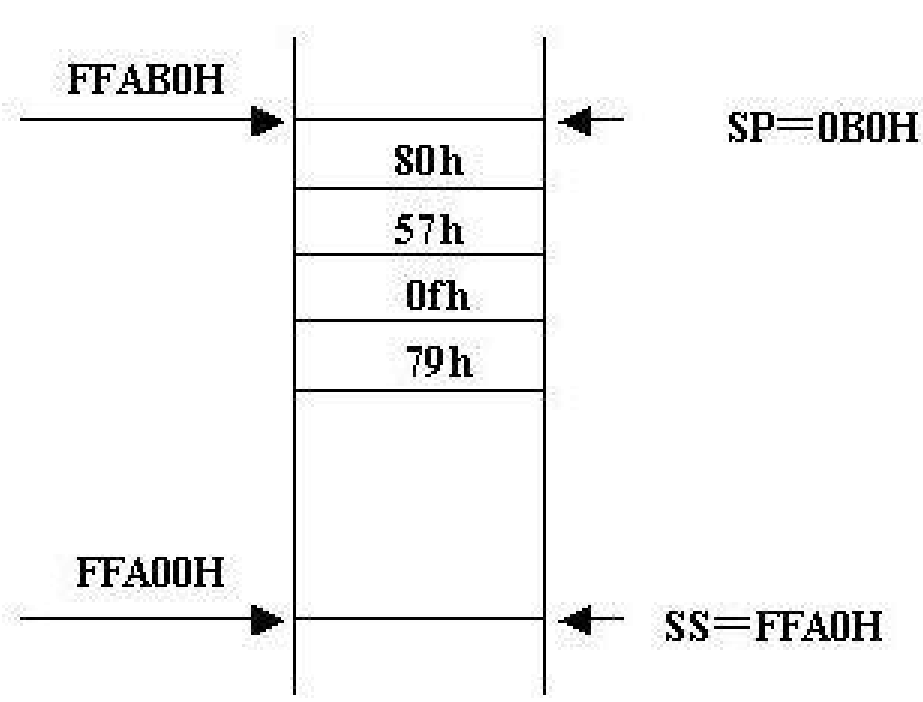
堆栈的两种基本操作是压栈和出栈，对应的指令是 PUSH和 POP。

2.5 已知 SS＝FFA0H、SP＝00B0H，画图说明执行下面指令序列时，堆栈区和 SP 的内容如何变化？

mov ax,8057h

push ax

mov ax,0f79h



push ax

pop bx　　;bx=0f79h

pop [bx] ;DS:[0f79h]=8057h

2.6 给出下列各条指令执行后 AL 值，以及 CF、ZF、SF、OF 和 PF 的状态：

mov al,89h AL=89h CF ZF SF OF PF

add al,al AL=12h 1 0 0 1 1

add al,9dh AL=0afh 0 0 1 0 1

cmp al,0bch AL=0afh 1 0 1 0 1

sub al,al AL=00h 0 1 0 0 1

dec al AL=0ffh 0 0 1 0 1

inc al AL=00h 0 1 0 0 1

2.7 设 X、Y、Z 均为双字数据，分别存放在地址为 X、X+2；Y、Y+2；Z、Z+2 的存储单元中，它们的运算结果存入 W 单元。阅读如下程序段，给出运算公式。

```
mov ax,X
mov dx,X+2
add ax,Y
adc dx,Y+2
add ax,24
adc dx,0
sub ax,Z
sbb dx,Z+2
mov W,ax
mov W+2,dx
```

W=X+Y+24-Z

2.8 请分别用一条汇编语言指令完成如下功能：

（1）把 BX 寄存器和 DX 寄存器的内容相加，结果存入 DX 寄存器。 ADD DX,BX

（2）用寄存器 BX 和 SI 的基址变址寻址方式把存储器的一个字节与AL 寄存器的内容相加，并把结果送到 AL 中。 ADD AL,[BX+SI]

（3）用 BX 和位移量 0B2H 的寄存器相对寻址方式把存储器中的一个字和 CX 寄存器的内容相加，并把结果送回存储器中。 ADD [BX+0B2H],CX

（4）用位移量为 0520H 的直接寻址方式把存储器中的一个字与数 3412H 相加，并把结果送回该存储单元中。 ADD WORD PTR [0520H],3412H

（5）把数 0A0H 与 AL 寄存器的内容相加，并把结果送回 AL 中。 ADD AL,0A0H

2.9；设 X、Y、Z、V 均为 16 位带符号数，分别装在 X、Y、Z、V 存储单元中，阅读如下程序段，得出它的运算公式，并说明运算结果存于何处。

为了避免与操作数地址混淆，将题中 X,Y,Z,V 字操作数改为 A,B,C,D

```
mov ax,X   ；ax=A
imul Y    ；dx,ax = A*B (  将操作数看作符号数，以下同）
mov cx,ax
mov bx,dx    ；bx,ax <-- dx,ax =A*B
mov ax,Z   ；ax = C
cwd  ；dx,ax =C  （扩展符号后为双字）
add cx,ax
adc bx,dx     ；bx,cx <-- bx,cx+dx,ax=A*B+C
sub cx,540
sbb bx,0     ；bx,cx<-- A*B+C-540
mov ax, V    ；ax= D
cwd  ；dx,ax= D  （扩展符号后为双字）
sub ax, cx
sbb dx, bx      ；dx,ax = dx,ax - bx,cx = D-(A*B+C-540)
idiv X      ；运算结果：[D-(A*B+C-540h)]/A  ；ax 存商，dx 存余数
```

2.10;

（1） xchg [si],30h xchg 的操作数不能是立即数

（2 ） pop cs 不能对 CS直接赋值

（3） sub [si],[di] 两个操作数不能都是存储单元

（4）push ah 堆栈的操作数不能是字节量

（5） adc ax,ds adc 的操作数不能是段寄存器

（6） add [si],80h 没有确定是字节还是字操作

（7） in al,3fch in 不支持超过 FFH的直接寻址

（8） out dx,ah out 只能以 AL/AX为源操作数

2.11；给出下列各条指令执行后的结果，以及状态标志 CF、OF、SF、ZF、PF 的状态。

指令	AX的值	CF	OF	SF	ZF	PF
Mov ax,1407h	1470h	-	-	-	-	-
And ax,ax	1470h	0	0	0	0	0
Or ax,ax	1470h	0	0	0	0	0
Xor ax,ax	0	0	0	0	1	1
Not ax	0ffffh	-	-	-	-	-
Test ax,0f0f0h	0ffffh	0	0	1	0	1

注意: 1. mov, not 指令不影响标志位

2. 其他逻辑指令使 CF=OF=0,根据结果影响其他标志位。

2.12；假设例题 2.32的程序段中，AX＝08H，BX＝10H，请说明每条指令执行后的结果和各个标志位的状态。

指令	注释	执行结果	CF	OF	SF	ZF	PF
mov si,ax	si=ax	si=0008h	-	-	-	-	-
shl si,1	si=2*ax	si=0010h	0	0	0	0	0
add si,ax	si=3*ax	si=0018h	0	0	0	0	1
mov dx,bx	dx=bx	dx=0010h	-	-	-	-	-
mov cl,03h	cl=03h		-	-	-	-	-
shl dx,cl	dx=8*bx	dx=0080h	0	u	0	0	0
sub dx,bx	dx=7*bx	dx=0070h	0	0	0	0	0
add dx,si	dx=7*bx+3*ax	dx=0088h	0	0	0	0	1

注意：

1. 左移 N次相当于乘于 2 的 N次方，右左移 N次相当于除乘于 2 的 N次方。

2. 移位指令根据是否移入“1”到 CF,设置 CF,根据移位后的结果影响 SF,ZF,PF。根据最高符号位是否改变设置 OF，如改变 OF=1.

3. ‘u’表示无定义，‘-’表示无影响。

2.13　编写程序段完成如下要求：

（1）　用位操作指令实现 AL（无符号数）乘以 10

;不考虑进位 mov bl,al

mov cl,3

shl al,cl

add al,bl ;shl bl,1

add al,bl

；考虑进位 xor ah,ah

mov bx,ax

mov cl,3

shl ax,cl

add ax,bx ;shl bx,1

add ax,bx

(2)　用逻辑运算指令实现数字 0 ~ 9的 ASCII 码与非压缩 BCD 码的互相转换

数字 0～9 的 ASCII 码是：30h～39h

非压缩 BCD码的 0～9 是：00h～09h

方法一：

```
and al,0fh ;     实现 ASCII 到非压缩 BCD码的转换
or al,30h ;     实现非压缩 BCD码到 ASCII 的转换
```

方法二：

```
xor al,30h     ；求反 D5D4位，其他不变
；即高 4 位为 3，则变为 0；高 4 位为 0，则变为 3
mov cl,4
again: shr dx,1 ;       实现逻辑右移
□□□“sar       dx,1”□□□□□□□□
 rcr ax,1
 dec cl
 jnz again
```

(3) 把 DX:.AX 中的双字右移 4 位

```
    MOV CL, 4
    SHR AX, CL
    MOV BL, DL
    SHR DX, CL
    SHL BL, CL
    OR AH, BL
```

2.14；已知 AL＝F7H（表示有符号数-9），分别编写用 SAR 和 IDIV 指令实现的除以 2 的程序段，并说明各自执行后，所得的商是什么？

（1）用 sar 编写

```
        mov al,0f7h      ;-9送 al
        sar al,1          ;结果：al=0fbh 即-5
```

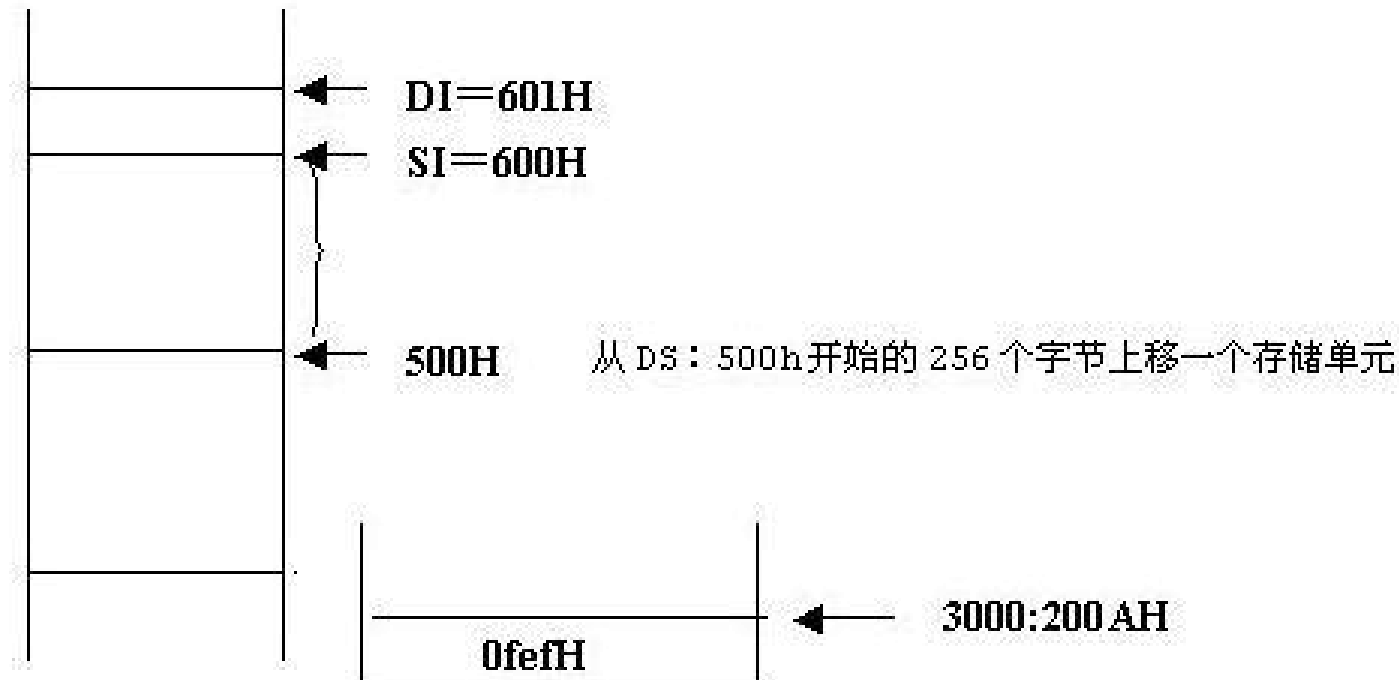
（2）用 idiv 编写

```
    mov al,0f7h    ；-9送 al
    cbw            ;字节符号扩展位字
    mov bl,2       ;注意除数不可为立即数
    idiv bl        ；结果：商为 al=fch  (-4)
                   ；  余数为 ah=ffh (-1)
```

结论：符号数的除法 用 idiv 准确

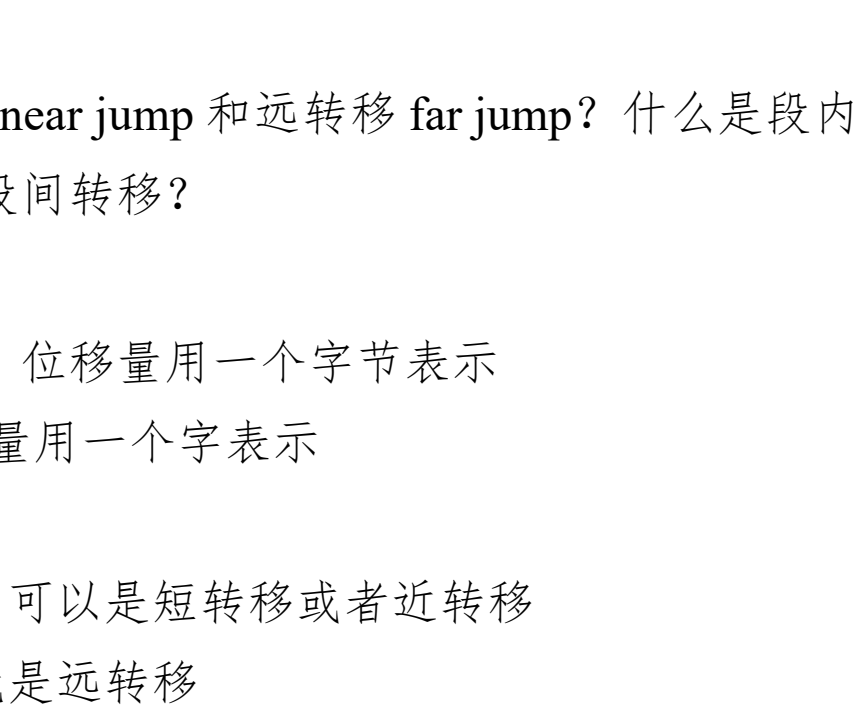
2.15、已知数据段 500h ~600h 处存放了一个字符串，说明下列程序段执行后的结果：

```
mov si,600h
    mov di,601h
    mov ax,ds
        mov es,ax
        mov cx,256
        std
        rep movsb
```



2.16、说明下列程序段的功能

```
cld
    mov ax,0fefh
    mov cx,5
   mov bx,3000h
    mov es,bx
   mov di,2000h
    rep stosw
```



2.17、指令指针 IP 是通用寄存器还是专用寄存器？有指令能够直接它赋值吗？哪类指令的执行会改变它的值？调试程序 DEBUG 环境下，如何改变 IP 数值？

2.18、控制转移类指令中有哪三种寻址方式？

2.19；什么是短转移 short jump、近转移 near jump 和远转移 far jump？什么是段内转移和段间转移？8086 有哪些指令可以实现段间转移？

短转移：指段内－128～127 之间的转移，位移量用一个字节表示

近转移：指段内±32K 之间的转移，位移量用一个字表示

远转移：指段间 1MB 范围的转移

段内转移：指在同一个代码段内的转移，可以是短转移或者近转移

段间转移：指转移到另外一个代码段，就是远转移

8086/8088CPU 的 JMP、CALL 和 INT n 指令可以实现段间转移

2.20；8086 的条件转移指令的转移范围有多大？实际编程时,你如何处理超出范围的条件转移？

8086 的条件转移的转移范围：在当前指令地址的 +127---- -128 之内。

如条件转移的转移范围超出此范围，可在此范围内安排一条无条件转移，再转移到范围外的目标地址。

2.21；假设 DS=2000H，BX=1256H，SI=528FH，位移量 TABLE=20A1H，[232F7H]=3280H，[264E5H]=2450H，试问执行下列段内间接寻址的转移指令后，转移的有效地址是什么?

(1)JMP Bx ；转移的有效地址 EA=BX=1256h

(2)JMP tABLE[Bx] ；转移的有效地址 EA=[ds:20a1h+1256h]=[232f7]=3280h

(3)JMP [Bx][si] ；转移的有效地址 EA=[ds:1256h+528fh]=264e5h=2450h

2.22、判断下列程序段跳转的条件

（1） xor ax,1e1eh

je equal

；AX=1e1eh（异或后为 0）

（2）test al,10000001b

jnz there

；AL的 D0或 D7至少有一位为 1

（3） cmp cx,64h

jb there

；CX（无符号数）＜ 64h

2.23；设置 CX = 0，则 LOOP 指令将循环多少次？例如

mov cx ，0 ；不循环，因为一进入循环就判 cx=0? 如 cx=0 就退出循环

delay ： loop delay

循环次数是 2 的 16 次方，即 2^16＝65536。

2.24 假设 AX 和 SI 存放的是有符号数，DX 和 DI 存放的是无符号数，请用比较指令和条件转移指令实现以下判断：

（1）若 DX > DI，转到 above 执行

cmp dx,di

ja above ；＝jnbe above

（2）若 AX > SI，转到 greater 执行

cmp ax,si

jg greater ；＝jnle greater

（3）若 CX = 0，转到 zero 执行

cmp cx,0 jcxz zero

jz zero

（4）若 AX－SI 产生溢出，转到 overflow 执行；

cmp ax,di

jo overflow

（5）若 SI≤AX□□□　less_eq　执行；

```
cmp si,ax
cmp ax,si
jle less_eq
jge less_eq
```

（6）若 DI≤DX□□□　below_eq 执行。

```
cmp di,dx
cmp dx,di
jbe below_eq
jae below_eq
```

2.25 有一个首地址为 array 的 20 个字的数组，说明下列程序段的功能。

```
 mov cx,20
    mov ax,0
    mov si,ax
sum_loop:
    add ax,array[si]
    add si,2
    loop sum_loop
    mov total,ax
 ；答：将首地址为 array 得 20 个字的数组求和，并将结果存入 total　单元中。
```

2.26 按照下列要求，编写相应的程序段：

(1) 起始地址为 string 的主存单元中存放有一个字符串（长度大于 6），把该字符串中的第 1 个和第 6 个字符（字节量）传送给 DX 寄存器。

```
  mov si,0
  mov dl,string[si]        ；第 1 个字符送 dl 寄存器
  mov si,5
  mov dh,string[si]        ；第 6 个字符送 dh 寄存器
```

(2) 从主存 buffer 开始的 4 个字节中保存了 4 个非压缩 BCD 码，现按低（高）地址对低（高）位的原则，将它们合并到 DX 中。

```
  xor si,si       ；si 清零
  mov al,buffer[si]        ；第一字节
  inc si
  mov ah,buffer[si]        ；第二字节
  mov cl,4
  shl ah,cl       ；BCD码移到高半字节
  or al,ah        ；组合成压缩 BCD码
```

```
                ；存入 dl 寄..
inc si
mov al,buffer[si]       ；第三字节
inc si
mov ah,buffer[si]       ；第四字节
mov cl,4
shl ah,cl       ；BCD码移到高半字节
or al,ah        ；组合成压缩 BCD码
mov dh,al     ；存入 dh 寄..
```

(3) 编写一个程序段，在 DX 高 4 位全为 0 时，使 AX = 0；否则使 AX = -1。

```
test dx,0f000h
jz zero
  mov ax,-1
  jmp done
  zero: mov ax,0
  done: ret
```

(4) 有两个 64 位数值，按 小端方式”存放在两个缓冲区 buffer1 和 buffer2 中，编写程序段完成 buffer1－buffer2 功能。

```
  lea bx,buffer1
  lea dx,buffer2
  mov cx,8    ；8 个字节
  xor si,si       ；si=0
  clc    ；CF=0
```

(5) 假设从 B800h : 0 开始存放有 100 个 16 位无符号数，编程求它们的和，并把 32 位的和保存在 DX.AX 中。

```
  mov ax,0b800h
  mov ds,ax    ；段地址
  xor si,si       ；地址偏移量 si=0
  xor dx,dx     ；和的高字 dx=0
  mov cx,99    ；加的次数
  mov ax,[si]      ；第一个数
again: inc si       ；指向下一个字单元
  inc si
  add ax,[si]       ；加下一个数
  jnc noc     ；无进位转
  inc dx     ；有进位 dx=dx+1
```

；次数-1

jnz cx,again　　；非 0 继续加

ret

(6)　已知字符串 string 包含有 32KB 内容，将其中的　符号替换成空格。

mov si,offset string

mov cx,8000h　；32k=2^15=8000h

again: cmp　[si],'$'

jnz next

mov [si],20h　　；if　[si]='$'　　[si]<--　　'　'

next: inc si

loop again

(7)　有一个 100 个字节元素的数组，其首地址为 array，将每个元素减 1（不考虑溢出）存于原处。

xor si,si　　；si<--0

mov cx,100　；循环次数

again: dec array[si]

dec cx

jnz again

(8)　统计以□$结尾的字符串 srting 的字符个数。

xor si,si ;si<--0

coun: cmp　string[si],'$'

je done

inc si

.