1. 选择题

1.下面的数值都是8位二进制数的补码，最大值的是( D )。  
A.1111 1111B B.0000 1111B   
C.11110000B D.0011 1111B

2.有关8086中物理地址说法不正确的是( C )  
A.物理地址长度是20位 B.物理地址由专门的地址加法器计算   
C.段地址加偏移地址得到物理地址 D.段首地址的低4位是0

3.下列指令中，源操作数不同的是( D )。  
A.MOV AL，-1 B.MOV AL, 0FFH   
C.MOV AL,255 B.MOV AL，10000001B

4 已知DS:100H开始的4个连续内存字节空间中,依次存放1、2、3和4四个数,则MOV AX,[101]执行后,AX值为( C )。  
A.2H B.23H C.302H  D.203H

5.下来指令中，使用的段寄存器不一样的是( A )。  
A.MOV AX,[100H] B. MOV AX,[BP]   
C.POP AX D.PUSH AX

除了B以外，其他的都是使用DS作为默认的段寄存器，因为它们都是操作数据的指令。而B是使用SS作为默认的段寄存器，因为它是操作堆栈的指令，BP是堆栈指针寄存器,它用来存放堆栈的偏移地址

6.已知AL为-1,则NEG AL执行后,CF和OF 的值是( C )。  
A.1,1  B.0,0 C.1,0 D.0，1

CF的值：在补码操作中，如果原始值不是0，CF将被设置为1。由于原始AL是-1（非0），所以CF将被设置为1。

OF的值：在这种情况下，没有发生溢出。因为-1的补码（即1）在8位二进制中可以正确表示。因此，OF将保持为0。

7.下列指令可以实现AX=BX+2的是(D )。  
A.MOV AX,BX+2 B.MOV AX,[BX+2]   
C.LEA AX,BX+2 D.LEA AX,[BX +2]

MOV bx, OFFSET a ; 将a的首地址加载到bx中

LEA bx, [a] ; 将a的首地址加载到bx中

8.已知AX 为-7,BL为-3,则指令IDIV BL执行后，AH的值为( B )。  
A.1 B.-1 C.2  D.-2

AH:AL / BL-> -7/-3=2 …-1 -> 余数储存在ah-> AH=-1

9.已知DS 不是0，下列指令中，有可能取出N号中断向量的段地址的是(D)。  
A. MOV BX,ES:[N\*4] B.MOV BX.ES:[N\*4+2]   
C. MOV BX,[N\*4] D.MOV BX,[N\*4+2]

中断向量是用来存放中断处理程序的入口地址的表，它位于内存的最低部分，从0000:0000开始。每个中断向量占用4个字节，前两个字节是偏移地址，后两个字节是段地址。所以，N号中断向量的段地址就是0000:N4+2的字的内容。这个指令的意思是将内存地址为DS:N4+2的字的内容移动到BX寄存器中。如果DS不是0，那么这个地址就不是0000:N4+2，而是DS:N\*4+2，所以这个指令就有可能取出N号中断向量的段地址。

10.已知字变量X的偏移地址是2000H,值为3000H,则JMPX后,则会跳到物理地址是( B )的地方执行。  
A.CS:2000H B.DS:3000H

JMP指令是无条件跳转的指令，它将指定的地址作为下一条指令的地址。这个指令的意思是将X的内容作为下一条指令的地址。由于X的内容是3000H，所以下一条指令的地址是DS:3000H。物理地址是由段地址和偏移地址的组合得到的，所以物理地址就是DS:3000H。

1. 判断题(判断正误，如果错误，指出错误的原因或改正，20分)

1.MOV cx,[cx]

正确

2.IMUL AL,3

imul的源操数不能是立即数

MOV BL 3

IMUL AL BL

3.SHL[1234H],1

[这是一条非法的指令，因为SHL指令的目的操作数不能是内存操作数3](https://wenku.baidu.com/view/d56eecf27c192279168884868762caaedd33baf0.html)。错误，应改为先将内存中的数据移动到一个寄存器中，再用SHL指令。

4.ADD DI,CF

[这是一条非法的指令，因为ADD指令的源操作数不能是标志位4](https://bbs.csdn.net/topics/330139524)。错误，应改为先将CF位移动到一个寄存器中，再用ADD指令

5.JG指令执行前，一定要使用CMP指令

[因为JG指令只是根据标志寄存器中的SF和OF位来判断是否跳转5](https://www.docin.com/p-2447961987.html)，而不一定要使用CMP指令来改变这些标志位，也可以使用其他的算术或逻辑指令。

6. JNZ WORD PTR[100];

这是一条非法的指令，因为JNZ指令的操作数不能是内存操作数。错误，应改为JNZ 100或者JNZ [100]

7.MOV CS, CODE: CODE

这是一条非法的指令，因为MOV指令不能用于改变CS寄存器的值。错误

8. MOVAX,X+Y;X和Y都是常量

MOVAX,X+Y;X和Y都是常量。这是一条合法的指令，它的含义是将X+Y的结果作为立即数赋值给AX寄存器

9. MOV AX，X+Y;X和Y都是变量

应改为先将X和Y的值分别移动到两个寄存器中，再用ADD指令相加，然后再用MOV指令赋值给AX寄存器。

10.Iret指令如果用在Far类型的函数中，也能返回到原来的调用位置，只是堆栈没有恢复。

错误，因为Iret指令不仅能返回到原来的调用位置，还能恢复堆栈和标志寄存器的值

三、简答题  
1. .定义一个数据段DATA，其中包括:字变量X，初值不定:长度是50的字节变量Y，初值为1,自变量Z，初值为X的偏移地址，字符串变量S,初值是2022

data segment

x dw ?

y dw 50 dup(1)

z dw offset x

s db ‘2022’, ’$’

data ends

2. 简述在: DOSBOX环境下，使用 DEBUG的相关命令，给地址是DS:100内存赋值1234H的过程。假设 DEBUG在E:\zbh 下

MOUNT C E:\zbh ; 使用 mount 命令来挂载包含 DEBUG 工具的目录

C: ; 切换到挂载的驱动器

DEBUG ;启动DEBUG

-E 100 34 12

Q

3.寄存器中的数均以2进制存储。统计AX中1的个数，并存放BL中。

mov cx, 16 ; 意味着循环16次

mov bx, 0 ; 将 BX 清零，用于存储结果

mov ax, 1234h ; 假设 AX 寄存器的值是 1234h，您可以根据需要更改此值

count\_ones:

shl ax, 1 ; 将 AX 中的位左移一位,左移若移了1，CF就=1了

jnc no\_increment ; 如果最高位（进位标志）不是 1，则跳过递增

inc bl ; 如果最高位是 1，则递增 BL

no\_increment:

loop count\_ones ; 循环直到 CX 减到 0

jnc意思是，cf=1就执行下一行代码，cf不等于1就跳转no\_increment

4.从255端口读入一个字节，将最高位取反，最低位清0后，从256端口输出。

in al, 255 ; 从端口 255 读入字节到 AL 寄存器

xor al, 80h ; 取反 AL 寄存器的最高位（80h=1000 0000b）

and al, 0FEh ; 清除 AL 寄存器的最低位（0feh=1111 1110b）

out 256, al ; 将 AL 寄存器的内容输出到端口 256

（一个位与 1 进行异或操作会翻转该位的值，而与 0 进行异或操作则保持该位的值不变）

5.假设BL的绝对值在10与99之间，将BL以10进制的形式，输出到屏幕上。

; 假设BL中的值是一个10到99之间的数字

mov ah, 0 ; 清空ah, 准备用于除法

mov al, bl ; 将BL的值移入al进行操作

; 获取十位

mov dl, 10 ; 设置除数为10

div dl ; ax / 10，商在al中，余数在ah中

add al, '0' ; 将十位转换为ASCII

mov dl, al ; 准备输出十位

mov ah, 02h ; 函数号，输出字符

int 21h ; 调用DOS中断来输出字符

; 获取个位

mov dl, ah ; 将余数（个位）移入dl

add dl, '0' ; 将个位转换为ASCII

mov ah, 02h ; 函数号，输出字符

int 21h ; 再次调用DOS中断来输出字符

这个题就是把dl=10，然后al除以10转换ascii输出，ah除以10转码输出  
  
四、程序设计(编写完整的程序，包括给出段定义，第一题10分，第二题20分，共30分)  
 1、统计长度为100的字节数组ARR中大写字母的个数，并存放在AL中。

(思路：用si指针遍历arr，[si]复制给al，al与A和Z比较，不是大写字母则inc si，移动到下一位，继续循环，是大写字母，则inc bl，最后数据储存到al)

data segment

arr db 100 dup(?) ; 定义一个长度为 100 的字节数组 ARR

data ends

code segment

start:

; 初始化数据段寄存器

mov ax, data

mov ds, ax

; 初始化指针和计数器

lea si, arr ; 将数组 ARR 的地址加载到 SI 寄存器

mov cx, 100 ; 设置循环计数器为 100（数组的长度）

mov bl, 0 ; 初始化 BL，用于计数

count\_uppercase:

; 检查当前字节是否为大写字母

mov al, [si] ; 将当前字节加载到 AL

cmp al, 'A' ; 检查是否大于等于 'A'

jb not\_uppercase

cmp al, 'Z' ; 检查是否小于等于 'Z'

ja not\_uppercase

; 如果是大写字母，递增计数器

inc bl

not\_uppercase:

; 移动到数组中的下一个字节

inc si

; 循环直到处理完数组中的所有字节

loop count\_uppercase

; 将计数器的结果存储到 AL 中

mov al, bl

; 程序结束处理

mov ah, 4Ch

int 21h ; 调用 DOS 中断来退出程序

code ends

end start  
 2、数据段中有3个长度均为100的有符号字缓冲区x、Y和V,要求编程计算.V=F(X,Y),即对X、Y缓冲区对应的每个字，按函数F计算后将结果送入对应的v缓冲区中。要求:求F(X，Y)功能写成一个子程序，且按照c语言的方式处理参数，即传入参数采用堆栈并从右向左传递、传出参数使用寄存器AX传递，假设每次运算的结果仍然是16位。F(X，Y)定义如下:

V=F(X,Y)=X(当Y=0时)

V=F(X,Y)=|X|+Y/100(当X<0且Y>0)

V=F(X,Y)=X/Y+100(其他)

data segment

; 定义长度为 100 的有符号字缓冲区 X, Y 和 V

x dw 100 dup(?) ; X 缓冲区

y dw 100 dup(?) ; Y 缓冲区

v dw 100 dup(?) ; V 缓冲区

data ends

code segment

start:

; 初始化数据段寄存器 DS

mov ax, data

mov ds, ax

; 初始化 SI, DI 和 CX 寄存器

; SI 指向 X 缓冲区，DI 指向 Y 缓冲区

; CX 用作循环计数器

mov si, offset x

mov di, offset y

mov cx, 100

process\_loop:

; 循环处理 X 和 Y 缓冲区中的每个元素

; 将 Y 中的当前值压入堆栈

push word ptr [di]

; 将 X 中的当前值压入堆栈

push word ptr [si]

; 调用子程序 CalculateF

call CalculateF

; 将计算结果存储到 V 缓冲区的当前位置

mov [si-100+offset v], ax

; 移动到下一个元素

add si, 2

add di, 2

; 循环计数器递减并检查是否为零

loop process\_loop

; 程序结束处理，返回到操作系统

mov ah, 4Ch

int 21h

CalculateF proc near

; 子程序 CalculateF 开始

; BP 寄存器用作基指针，存储堆栈地址

mov bp, sp

; 获取传入的参数 Y 和 X

mov ax, [bp+4] ; Y

mov bx, [bp+2] ; X

; 根据 F(X, Y) 函数的定义计算结果

cmp ax, 0

je X\_when\_Y\_is\_zero

cmp bx, 0

jl X\_negative\_Y\_positive

jmp Other

X\_when\_Y\_is\_zero:

; 情况 1：当 Y = 0 时，V = X

mov ax, bx

ret

X\_negative\_Y\_positive:

; 情况 2：当 X < 0 且 Y > 0 时，V = |X| + Y/100

cmp ax, 0

jle Other

neg bx ; 取 X 的绝对值

add ax, bx ; 加上 Y

sar ax, 1 ; 等效于 Y/100（位移操作）

sar ax, 1

ret

Other:

; 情况 3：其他情况，V = X/Y + 100

cmp bx, 0 ; 检查除数 Y 是否为零

je division\_by\_zero

; 准备除法：被除数在 DX:AX，除数在 BX

mov dx, 0 ; 清空 DX，因为我们假设 X 是一个 16 位的数

xchg bx, ax ; 交换 AX 和 BX 的值，现在 AX 中是 Y，BX 中是 X

idiv bx ; 执行除法：DX:AX / BX，商在 AX 中，余数在 DX 中

add ax, 100 ; 将 100 加到商中

ret

division\_by\_zero:

; 处理除以零的情况

; 这里我们暂时简单地将结果设置为 0

mov ax, 0

ret

CalculateF endp

code ends

end start