汇编考试大秘籍 周布伟

* **选择与判断**

**BCD码：**压缩的用4位二进制表示十进制，非压缩的用8位二进制表示十进制，高4位为0。

* 若80H表示压缩BCD码，其值为10进制80。√

压缩的BCD，一个字节可以表示2位10进制数

* 若80H表示非压缩BCD码，则其值为10进制数128。×

非压缩BCD码只低4位有效，故为0

* 若80H表示压缩BCD码，则其值为10进制数128。× 值为80

**段寄存器：**段寄存器存放段的首地址的高16位，是段地址；偏移地址是相对于首地址的位置。段寄存器的赋值通常用mov指令，但cs和ip不能用mov，mov时不能用常数，要么用寄存器，要么用内存。段寄存器不能用做加减乘除运算。

* 8086中偏移地址默认从0开始。√

段地址是段首地址的高16位

* 8086中段地址表示一个段的首地址的高16位。√
* 8086中段地址即为段的首地址。×

8086中段地址表示一个段首地址的高16位

* MOV CS，AX可以设置CS的值 ×

段寄存器可以使用通用寄存器和内存操作数赋值，但CS不可以

**8个通用寄存器：**4个访问内存：bx, bp, si, di；默认的段寄存器不是ds：bp、sp(默认ss)；内存访问方式：5个寻址方式只有直接寻址不用寄存器；loop循环寄存器在cx里。标志寄存器：CF(加减)/OF(溢出)/ZF(结果是0)

* mov ax, x √

这是直接寻址，编译成[…]，默认段寄存器在ds

* 加一减一不影响CF，not所有的标志寄存器都不影响
* 已知AL=80H，则NEG AL执行后，OF的值为1 √

80h求补neg后会产生溢出

* MOV AX,[AX]×

AX不能作为内存操作数

* ADD BX,[SP]指令是否正确×

SP不能表示内存地址

* xchg AX,[SP]指令是否正确×

SP不能表示内存地址

* ADD BX，[SI+2]可以实现BX=BX+2+SI×

BX和内存内容相加

**字地址、字节地址：**要用word ptr…运算中遇到常数和[寄存器]，无论加减、移位都是错的，有一个是寄存器则没关系

* mov [bx], 1 MOV [1000H ]，1000H ×

不知道是字节还是字

* mul [bx] ×
* 在内存1234中从33开始取一个字，取出来是403h

**mov：**对段寄存器ds的赋值，字节与字之间和内存与内存之间不能赋值、常数放内存标注字节还是字；

* offset x 编译时就是2，如果是x则是[2]
* mov x, offset x 相当于mov x, 2
* mov x, x 相当于mov x, [2]
* MOV CF,1 可以使CF为1 ×

CF不能通过MOV来赋值

**xlat：**不用操作数，把bx+al所指的单元内容给al，预先要将表的首地址(偏移地址)给bx，将信息在表中的行号给al；

**压栈、出栈**以及sp的变化；

* PUSH AL PUSH 12H ×

PUSH操作只能是16位操作数

* PUSHF指令执行完成后，SP的值加2 ×

标志压栈操作，和普通的压栈操作一样，出栈加2，入栈减2

* POPF指令执行完成后，SP的值加2 √

**lea：**取内存的偏移地址，不能是寄存器、常数的地址，offset是给编译器看的，直接加变量，不能加任何东西，所有的变量在编译的时候，如果没有加offset，所有的变量在编译的时候都是用[偏移地址]表示，加了则直接是偏移地址没有[]，地址本身是个数值。

* lea BX,[BX+2+SI]可以实现BX=BX+2+SI√
* LEA BX, BX+2+SI可以实现可以实现BX=BX+2+SI ×

**加减乘除：**带借位进位、加一减一，除法，字节被除数放ax，字被除数放dx:ax中，乘法，字节乘数放al中，字乘数放ax中

* DIV 80H × MUL 20 ×

MUL和 DIV的操作数不是常数，只能是通用寄存器或者内存操作数

* 已知AX=-1，CWD指令执行完成后，DX的值是-1 √

CWD是对AX的符号位扩展，AX的符号位为1，所以DX=0FFFFH=-1

* 已知AL=-1，则NEG AL执行后，CF的值为1。 √

AL为非0，则CF为1

* ADD [BX],[100]可实现两内存内容相加。×

内存和内存不能直接相加

* ADD AX，CF可以实现AX值加CF ×
* 32位加16位，看16位是否为有符号数，有符号数则扩展，无符号数直接高位加0

**逻辑运算：**与(清零)、或(置一)、异或(取反)、非(全部取反、不影响标志位)，注意位数！

**移位运算：**移位不为1要放cl

* 已知AL为2，SHL AX，AL，可以实现AX左移2次。 ×

源操作数不能是AL，只能是CL或者1

**跳转：**jmp和call可以段间跳转，可以用内存间接的方式，也可以直接用通用寄存器。jmp l跳到l标号所对应的地方，jmp x 跳到x内容表示的位置；比较跳转中有符号jg、jl，无符号ja、jb，比较前不用cmp没关系。

* jmp [bx] ×

字节还是字

* JMP AX JMP CX指令是否正确 √
* cmp a, b 与 sub a, b 标志一样，没有区别；cmp a, b 与 sbb a, b 引起的标志不一样

**中断：**int 21h，输入输出，计算机中只有2进制，其他进制显示。

**指令：**

end start：指示源码到此结束，指示程序开始执行点(start标号处)

assume：不是给寄存器赋值，只是在编译的时候要不要在变量前加段前缀

mov ah, 4ch int 21h：程序到此结束

函数的定义：int 21h后面或者start前面，在这两个位置可以定义变量，代码段定义则是cs:…

**编译、debug的使用：**

编译 MASM myfile.asm

连接 LINK myfile.obj

debug:

mount c e:\

c:

debug

* 显示存储单元命令

❖ -D [ADDRESS] 或 [RANGE]

❖ 默认：前面没用过D命令，则地址为CS:IP， 否则从前一个D结束地址显示。

* 修改存储单元内容命令

❖ -E ADDRESS [LIST] ；

❖ -E ds:100 "This is the text example" 01 02 03

* 检查和修改寄存器内容命令

❖ -R [REGISTER NAME] ；寄存器名字

* 汇编命令

❖ -A [ADDRESS]

* 反汇编命令

❖ -U [ADDRESS]或[RANGE]

* 运行命令

❖ -G [=ADDRESS ] [ADDRESS2 [ADDRESS3]

* 跟踪命令

❖ -T [=ADDRESS] [VALUE] ；VALUE运行的 条数

* 继续命令

❖ -P [=ADDRESS] [VALUE]

❖ CALL等指令当成一条。

* 退出命令

❖ -Q

汇编考试大秘籍 周布伟

* **简答与编程**
* 基本的程序框架

stack segment stack

dw 512 dup(?)

stack ends

data segment

... ;在数据段定义数据

data ends

code segment

assume cs:code,ds:data,ss:stack

start: mov ax,data

mov ds,ax

... ;在代码段填入指令序列

mov ah,4ch

int 21h

... ;子程序代码

code ends

end start

* 求一个数的相反数

neg运算

先not运算后add 1

* 求0~15的平方

将0~15的平方预先放在从T开始的缓存中

MOV BX,OFFSET T

MOV AL,n

XLAT

* 由输入值得到输出值

将X和Y的对应关系保存到表格中，可以是

一维数组，可保存至ROM

对于输入值，采用二分查找进行计算

* x、y、z均为双精度无符号数，分别存放在地址为X,X+2；Y, Y+2；Z, Z+2的存储单元中，用指令序列实现w ← x+y+24-z ，并用W, W+2单元存放w。

MOV CX, X

ADD CX，Y

MOV BX, X+2

ADC BX, Y+2 ; ；x+y

ADD CX, 24

ADC BX，0 ； x+y+24

SUB CX, Z

SBB BX, Z+2 ; x+y+24-z

MOV W，CX

MOV W+2，BX

* x,y,z,v均为16位带符号数，计算(v-(x\*y+z-540))/x

MOV AX, X

IMUL Y ; x\*y

MOV CX, AX

MOV BX, DX

MOV AX, Z

CWD

ADD CX, AX

ADC BX, DX ; x\*y+z

SUB CX, 540 ;？

SBB BX, 0 ; x\*y+z-540

MOV AX, V

CWD

SUB AX, CX

SBB DX, BX ; v-(x\*y+z-540)

IDIV X ; (v-(x\*y+z-540))/x

* 在DX和AX中存放着一个32位数据，试将其左移1位。

SHL AX,1

RCL DX,1

* 把(BL)中的8位数高低4位互换

MOV CL,4

ROL/ROR BL, CL

MOV DL,BL

MOV CL,4

SHR BL, CL

SHL DL, CL

OR BL, DL

* 完成分段函数

CMP AL,0

JG L1

JL L2

MOV AH,0

JMP DONE

L1: MOV AL,1

JMP DONE

L2:MOV AL,-1

DONE: …

* 找出100个有符号字节数中间的最大数。

设有符号数首地址是ARR

MOV AL, 80H

MOV CX，100

MOV SI, 0

AGAIN：CMP AL，ARR[SI]

JGE NEXT

MOV AL, ARR[SI]

NEXT:

INC SI

LOOP AGAIN

* 从键盘读入一个2位的十进制的正整数，存入BL中。

MOV AH,1

INT 21H ；等待输入十位

AND AL,0FH ；取数

MOV BL,10

MUL BL ；乘以10

MOV BL,AL ；暂存

MOV AH,1 ；等待输入个位

INT 21H

AND AL,0FH ；取数

ADD BL,AL ；相加

* 计算AX的绝对值

cmp ax,0

jns nonneg ;分支条件：AX≥0

neg ax ;条件不满足，求补

nonneg: mov result,ax ;条件满足

* 显示BX最高位

shl bx,1 ;BX最高位移入CF

jnc one ;CF＝0，即最高位为0，转移

mov dl,’1’

;CF＝1，即最高位为1，DL←’1’

jmp two ;一定要跳过另一个分支体

one: mov dl,’0’ ;DL←’0’

two: mov ah,2

int 21h ;显示

* 判断2次方程有无实根

;a、b、c均为字节变量：-127~127

mov al,\_b

imul al

mov bx,ax ;BX中为b^2

mov al,\_a

imul \_c

mov cx,4

imul cx ;AX中为4ac（DX无有效数据?）

cmp bx,ax ;比较二者大小

jge yes ;条件满足？

mov tag,0

;第一分支体：条件不满足，tag←0

jmp done ;跳过第二个分支体

yes: mov tag,1

;第二分支体：条件满足，tag←1

done: .exit 0

* 根据键盘输入的1~8数字转向不同的处理程序

start1: mov dx,offset msg ;提示输入数字

mov ah,9

int 21h

mov ah,1 ;等待按键

int 21h

cmp al,'1' ;数字 < 1？

jb start1

cmp al,'8' ;数字 > 8？

ja start1

and ax,000fh ;将ASCII码转换成数值

dec ax

shl ax,1 ;等效于add ax,ax

mov bx,ax

jmp table[bx]

;（段内）间接转移：IP←[table+bx]

start2: mov ah,9

int 21h

.exit 0

disp1: mov dx,offset msg1 ;处理程序1

jmp start2

* 将一个字符串中所有大写字母改为小写

mov bx,offset string

again: mov al,[bx] ;取一个字符

or al,al ;是否为结尾符0

jz done ;是，退出循环

cmp al,'A' ;是否为大写A～Z

jb next

cmp al,'Z'

ja next

or al,20h

;是，转换为小写字母（使D5=1）

mov [bx],al ;仍保存在原位置

next: inc bx

jmp again ;继续循环

done: .exit 0

* 把BX中的二进制数以十六进制的形式显示在屏幕上。

mov ch, 4

mov cl, 4

rotate: rol bx, cl

mov al, bl

and al, 0fh

cmp al, 0ah

jl next

add al, 37h ;’A’-’F’

jmp done

next: add al, 30h’;’0’-’9’

done: mov dl, al

mov ah, 2

int 21h

dec ch

jnz rotate

* 数据段中Score缓冲区中有5个学生的成绩（字节型）。将各自的名次算出填充到Rank缓冲区中。

MOV CL,5

MOV SI,0

AGAIN:

MOV AL,SCORE[SI]

MOV DI,0

MOV CH,5

GOON:

CMP AL,SCORE[DI]

JAE NEXT

INC RANK[SI]

NEXT:

INC DI

DEC CH

JNZ GOON

INC SI

DEC CL

JNZ AGAIN

* 子程序实现回车换行

CSEG SEGMENT

ASSUME CS:CSEG

START:

…

CALL LF\_CR

…

MOV AH,4CH

INT 21H

LF\_CR PROC

MOV DL,0DH

MOV AH,2

INT 21H

MOV DL,0AH

MOV AH,2

INT 21H

RET

LF\_CR ENDP

CSEG ENDS

END START

* 编写子程序完成a-b=c

DATA SEGMENT

A DW 2

B DW 1

SUM DW 0

DATA ENDS

SUBSEG SEGMENT

ASSUME CS:SUBSEG,DS:DATA

HTON\_F PROC FAR

PUSH BP

MOV BP,SP

PUSH AX

PUSH BX

MOV BX,[BP+8]

MOV AX,[BP+6]

SUB AX,BX

MOV

[BP+10],AX

POP BX

POP AX

POP BP

RET 4

HTON\_F ENDP

SUBSEG ENDS

CSEG SEGMENT

ASSUME CS:CSEG, DS:DATA

START:

MOV AX,DATA

MOV DS,AX

SUB SP, 2

PUSH B

PUSH A

CALL HTON\_F

POP SUM

MOV AH, 4CH

INT 21H

CSEG ENDS

END START

* 递归求解：Sum(n)=1+2…+n

n DW 100

Result DW ?

SUB SP,2

PUSHn

CALL SUM

POP Result

SUM PROC FAR

PUSH BP

MOV BP,SP

PUSH AX

MOV AX,[BP+6]

CMP AX,0

JNZ SUM1

MOV AX,0

JMP EXIT

SUM1: SUB SP,2

DEC AX

PUSH AX

CALL SUM

POP AX

ADD AX, [BP+6]

EXIT:

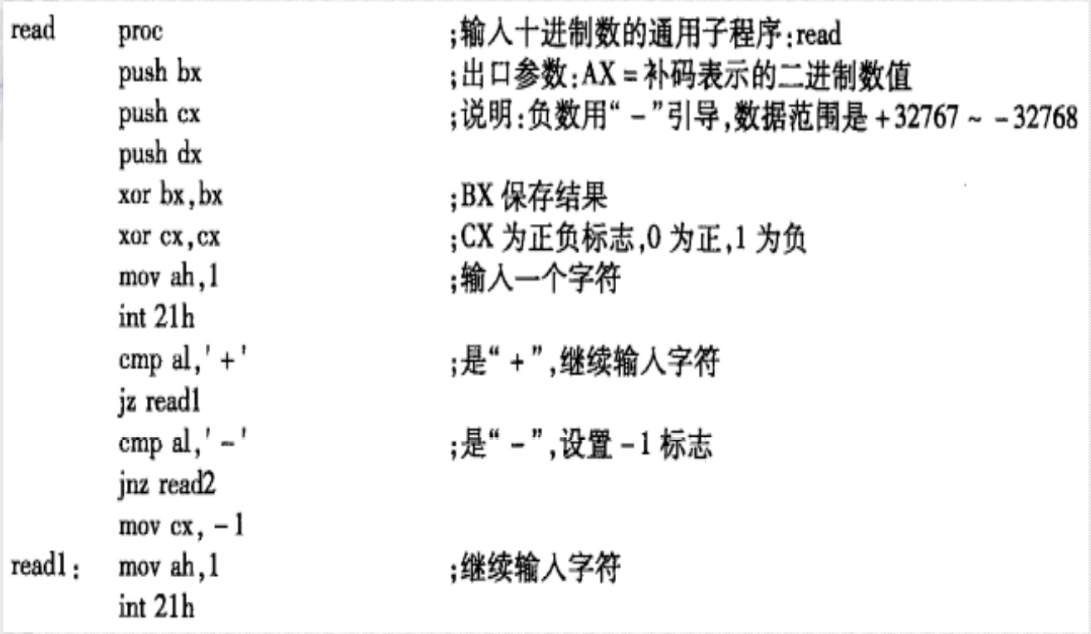
MOV [BP+8],AX

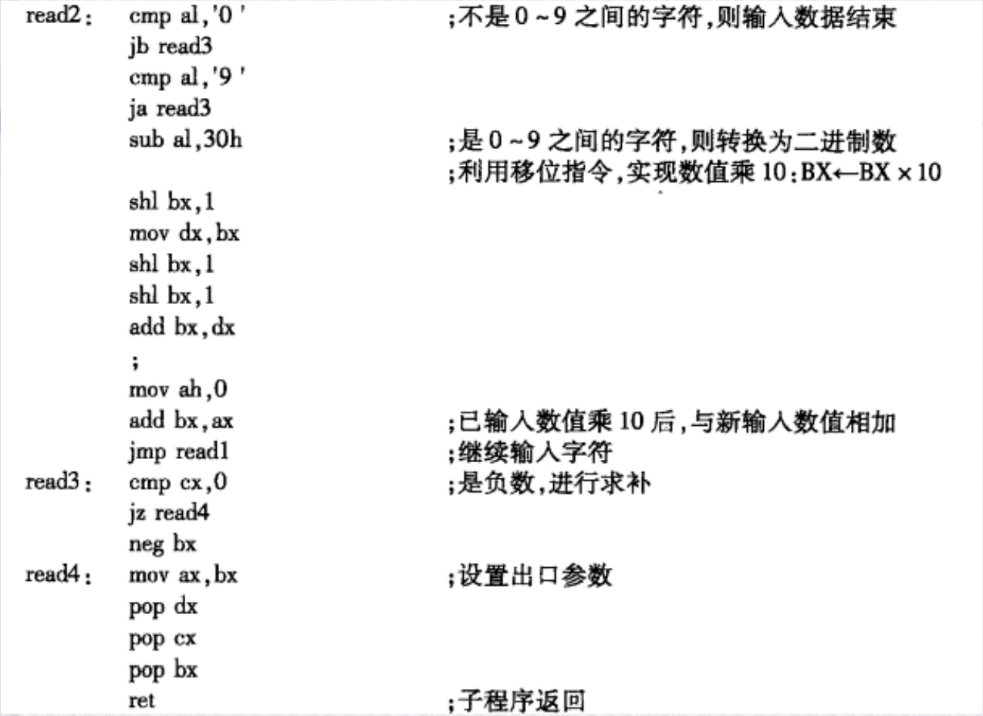
POP AX

POP BP

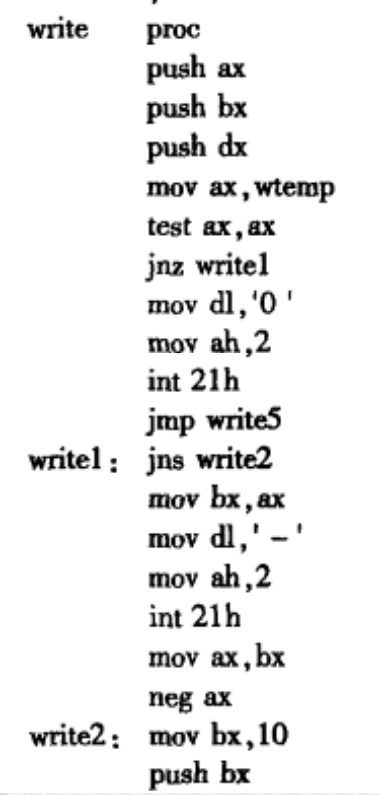
RET 2

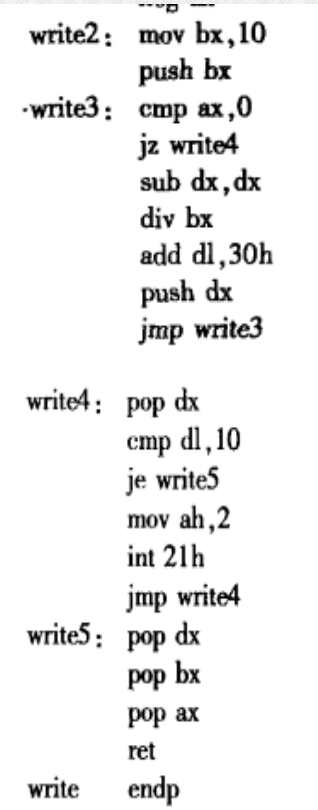
* 从键盘输入有符号十进制数子程序

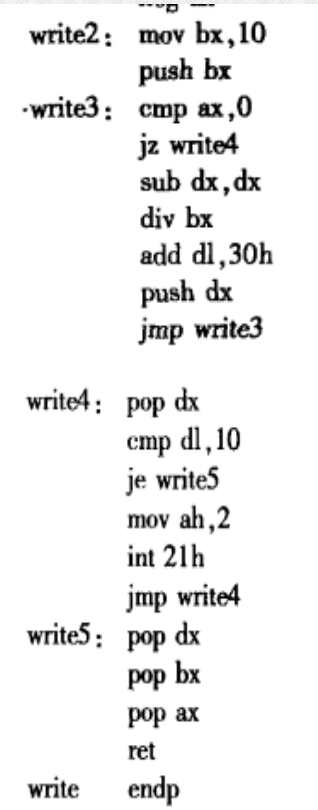




* 输出有符号十进制数子程序







* 数据段中X、Y、Z、V均为字变量，存放的是16位带符号数。编写汇编语言程序计算表达式值（V–（X\*Y+Z-720））/X。

;exp1-1

stack segment stack

dw 512 dup(?)

stack ends

data segment

x dw 1

y dw 720

z dw 1

v dw 6 ;最终计算结果应该是5h

data ends

code segment

assume cs:code, ds:data, ss:stack

start: mov ax, data

mov ds, ax

mov ax, x

imul y ;带符号

mov cx, ax

mov dx, bx

mov ax, z

cwd ;32位计算，符号扩展

add cx, ax

adc bx, dx

sub cx, 720

sbb bx, 0 ;与0 sbb 完成借位

mov ax, v

mov dx, 0

sub ax, cx

sbb dx, bx

idiv x

mov ah, 4ch

int 21h

code ends

end start

* BL中的只有一位为1。编写程序测试，并输出提示信息“The X Bit is 1”，要求：**使用地址表实现分支功能**，在分支中输出提示信息。

;exp1-2

stack segment stack

dw 512 dup(?)

stack ends

data segment

x db 00100000b

s0 db 'The 0 Bit is 1', 0dh, 0ah, '$'

s1 db 'The 1 Bit is 1', 0dh, 0ah, '$'

s2 db 'The 2 Bit is 1', 0dh, 0ah, '$'

s3 db 'The 3 Bit is 1', 0dh, 0ah, '$'

s4 db 'The 4 Bit is 1', 0dh, 0ah, '$'

s5 db 'The 5 Bit is 1', 0dh, 0ah, '$'

s6 db 'The 6 Bit is 1', 0dh, 0ah, '$'

s7 db 'The 7 Bit is 1', 0dh, 0ah, '$'

adtab dw l0, l1, l2, l3, l4, l5, l6, l7

data ends

code segment

assume cs:code, ds:data, ss:stack

start: mov ax, data

mov ds, ax

mov bl, x

mov cx, 8 ;循环计数器

mov dx, -1 ;计位器

again: inc dx

test bl, 1 ;看最低位

ror bl, 1 ;不影响标志位

loope again

shl dx, 1

mov si, dx

jmp adtab[si]

l0: lea dx, s0

jmp s

l1: lea dx, s1

jmp s

l2: lea dx, s2

jmp s

l3: lea dx, s3

jmp s

l4: lea dx, s4

jmp s

l5: lea dx, s5

jmp s

l6: lea dx, s6

jmp s

l7: lea dx, s7

s: mov ah,9

int 21h

mov ah, 4ch

int 21h

code ends

end start

;exp1-2 另一种方式

stack segment stack

dw 512 dup(?)

stack ends

data segment

x db 00100000b

s0 db 'The 0 Bit is 1', 0dh, 0ah, '$'

s1 db 'The 1 Bit is 1', 0dh, 0ah, '$'

s2 db 'The 2 Bit is 1', 0dh, 0ah, '$'

s3 db 'The 3 Bit is 1', 0dh, 0ah, '$'

s4 db 'The 4 Bit is 1', 0dh, 0ah, '$'

s5 db 'The 5 Bit is 1', 0dh, 0ah, '$'

s6 db 'The 6 Bit is 1', 0dh, 0ah, '$'

s7 db 'The 7 Bit is 1', 0dh, 0ah, '$'

adtab dw s0, s1, s2, s3, s4, s5, s6, s7

data ends

code segment

assume cs:code, ds:data, ss:stack

start: mov ax, data

mov ds, ax

mov bl, x

mov cx, 8

mov dx, -1

again: inc dx

test bl, 1

ror bl, 1

loope again

shl dx, 1

mov si, dx

mov dx, adtab[si]

mov ah,9

int 21h

mov ah, 4ch

int 21h

code ends

end start

* 求一个班50名学生成绩的平均值,并将结果显示出来。

;exp1-3

stack segment stack

dw 512 dup(?)

stack ends

data segment

score dw 5 dup (60,85,95,78,90,77,66,90, 55,98) ;平均分79分

data ends

code segment

assume cs:code, ds:data, ss:stack

start: mov ax, data

mov ds, ax

mov cx, 50

mov si, 0

mov bl, 50

xor ax, ax

again1: add ax, score[si]

add si, 2 ; score 是字类型

loop again1

div bl

mov cx, 2

mov bl, 10

again2: and ax, 0ffh ;清除余数

div bl

push ax

loop again2

mov cx, 2

again3: pop bx

add bh, 30h ;取余数转换成字符

mov dl, bh

mov ah, 02h

int 21h

loop again3

mov ah, 4ch

int 21h

code ends

end start

* 编写一个**子程序**计算z=f(x,y)=x\*y+x-y（x,y,z有符号数字数），要求：

（1）输入参数通过堆栈传送，输出参数通过AX传送；

（2）主程序调用后，结果需要输出到屏幕上。

;exp2-1

stack segment stack

dw 512 dup(?)

stack ends

data segment

x dw 14

y dw 4

z dw ?

data ends

code segment

assume cs:code, ds:data, ss:stack

start: mov ax, data

mov ds, ax

sub sp, 2

push y

push x

call fun

pop ax

mov z, ax

mov cx, 3

mov bl, 10

again1: and ax, 0ffh

div bl

push ax

loop again1

mov cx, 3

again2: pop bx

add bh, 30h

mov dl, bh

mov ah, 02h

int 21h

loop again2

mov ah, 4ch

int 21h

fun proc

push bp

mov bp, sp

push ax

push dx

xor dx, dx

mov ax, [bp+6]

imul word ptr[bp+4]

add ax, [bp+4]

adc dx, 0

sub ax, [bp+6]

sbb dx, 0

mov [bp+8], ax ;默认不超过8位

pop dx

pop ax

pop bp

ret 4

fun endp

code ends

end start

* VC++中编写内联汇编，将字节数组的每个元素高低四位互换。char dbyte[] = {0x12,0x34,0x56,0x78,0x9a};

//exp2-2

#include <stdio.h>

int main(void)

{

char dbyte[5] = { 0x12,0x34,0x56,0x78,0x9a };

int i;

\_\_asm

{

mov esi, 0

mov cx, 5

again: mov bl, dbyte[esi]

mov ax, 0

next: ror bl, 1

inc ax

cmp ax, 4

jne next

mov dbyte[esi], bl

inc esi

loop again

}

for (i = 0; i < 5; i++)

{

printf("%#x ", dbyte[i]);

}

return 0;

}