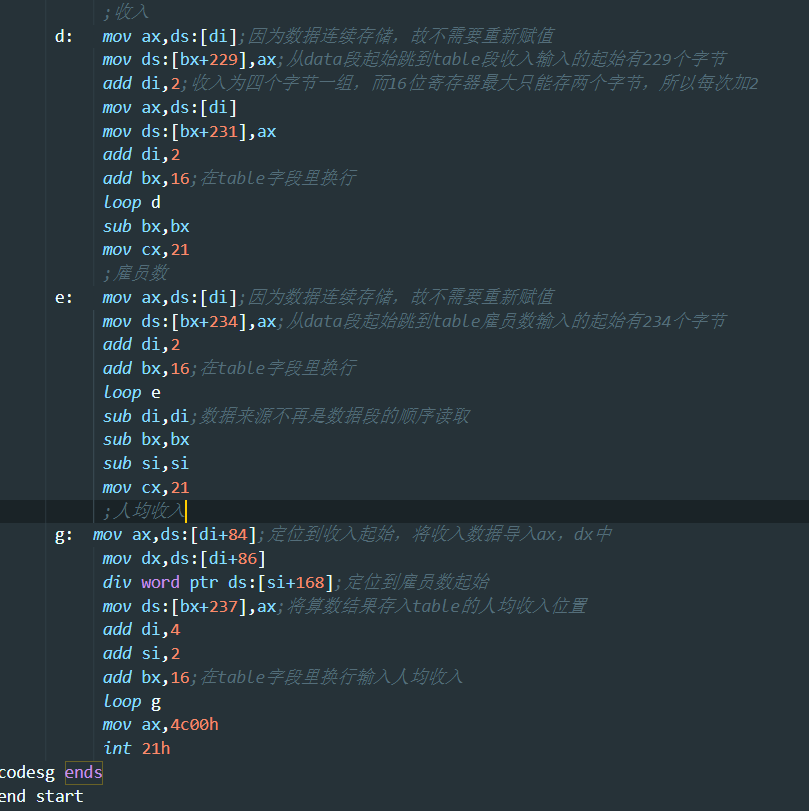
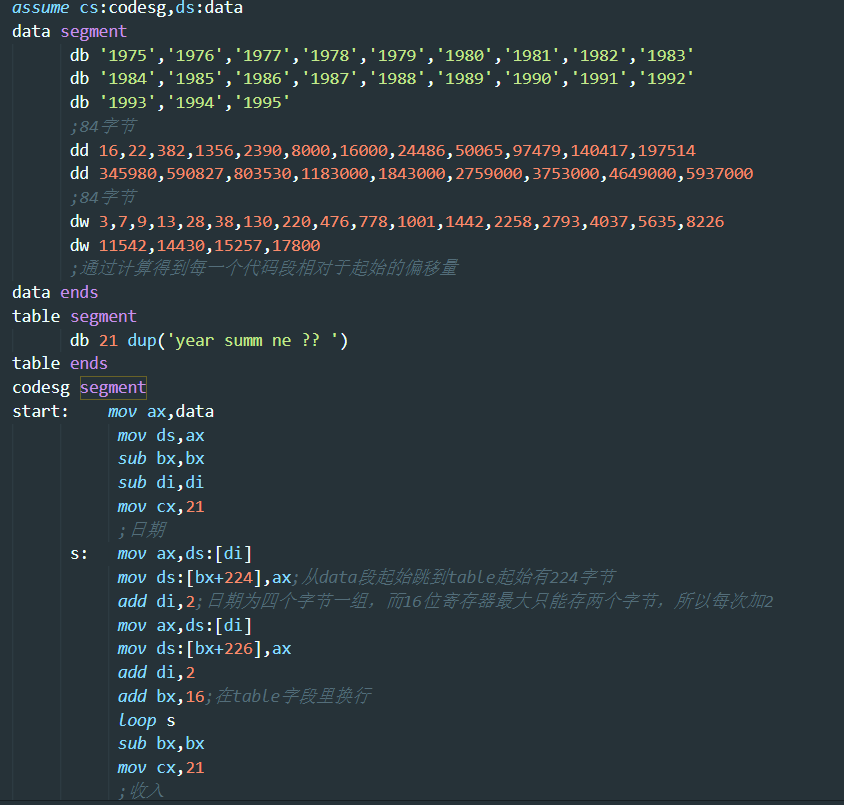
实验7

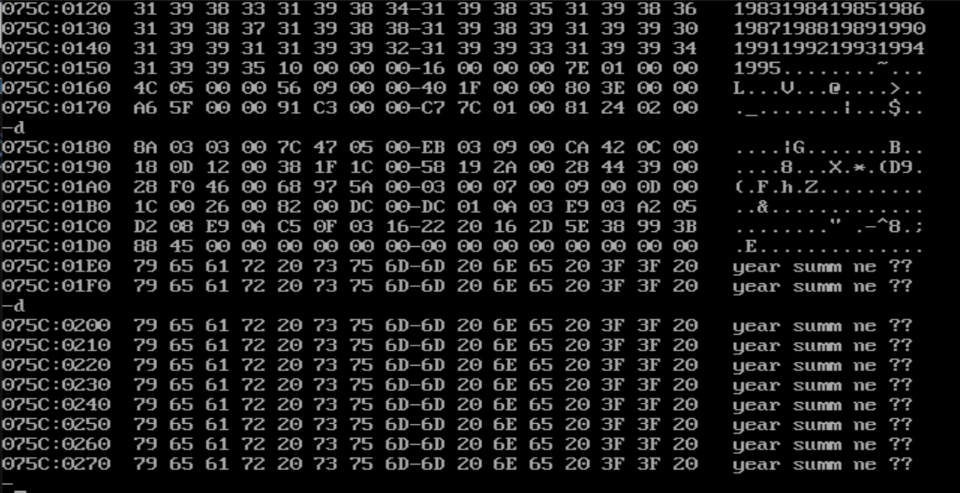
代码如下：



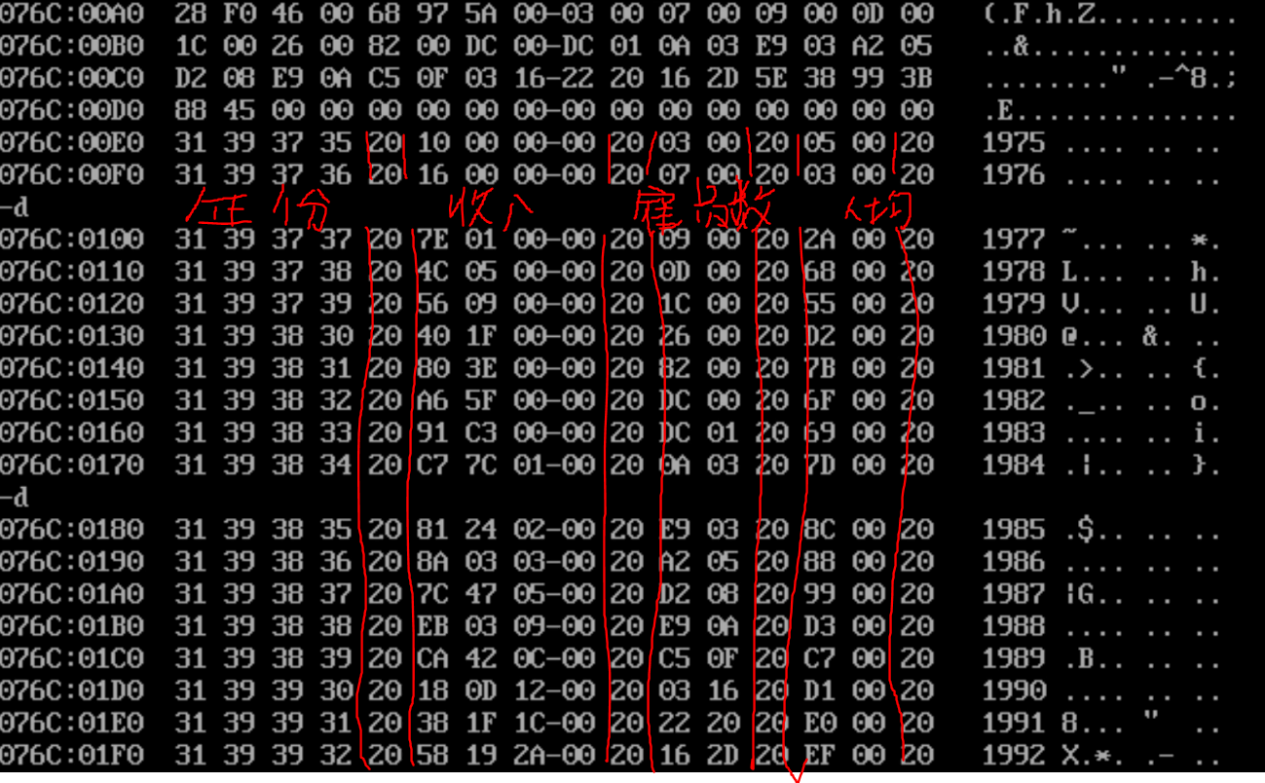
结果如下：

未输入前，数据存储在ds:0100处



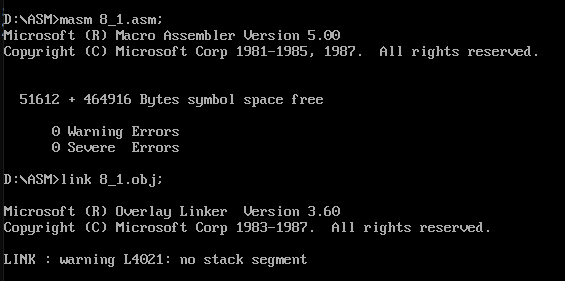


程序运行后，结果存入table处，如下图所示：



实验8

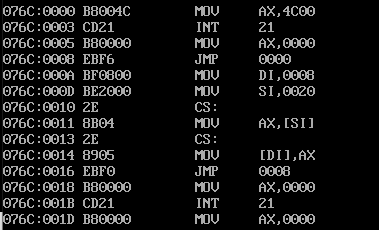
如图所示，程序成功通过编译、链接。



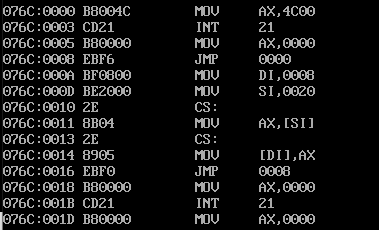
程序成功运行。

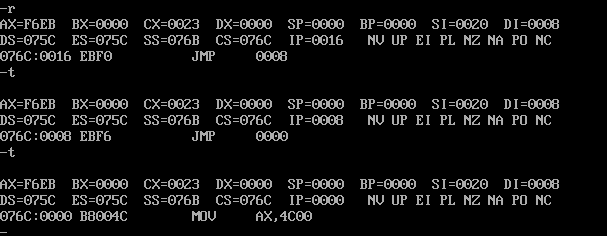


解释：



首先，程序先从start开始运行，运行到s末尾这一段过程中，我们可以看到程序将s2的机器码EBF6移动到了s的开头部分（即两个NOP所占位置），所以执行s0的指令后，ip回到s位置，而又重新运行之前被转移到此的跳转指令jmp short s1。

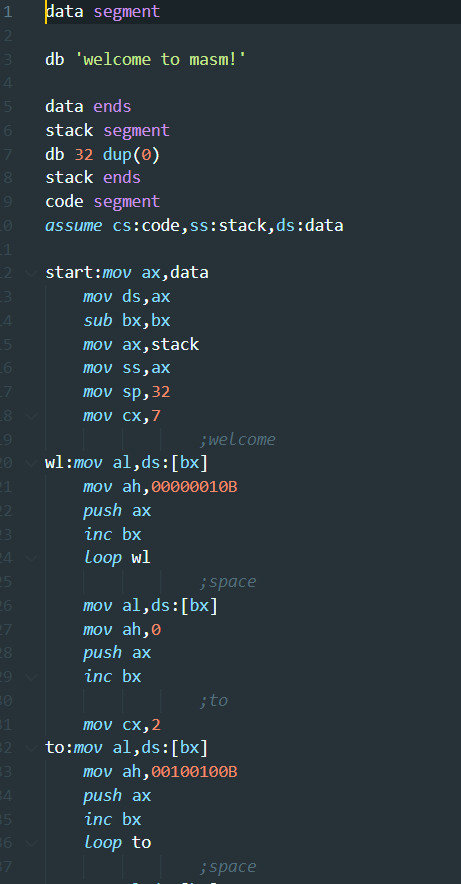


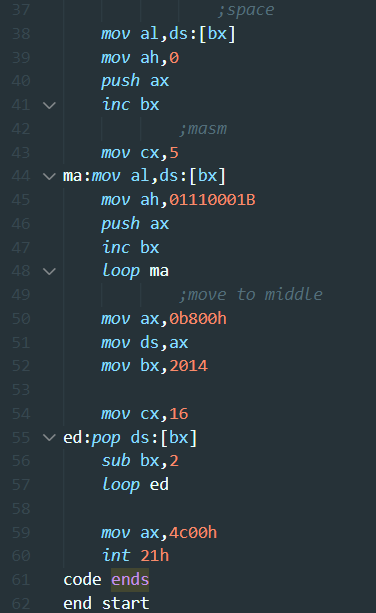


因为jmp指令只是通过改变位移量实现ip跳转，所以根据命令EBF6，ip应该减去10，而因为前面的代码长度正好为10个字节，所以正好跳转到了mov ax,4C00h处，故可以正常退出。

实验9

代码如下：





实验结果如下：



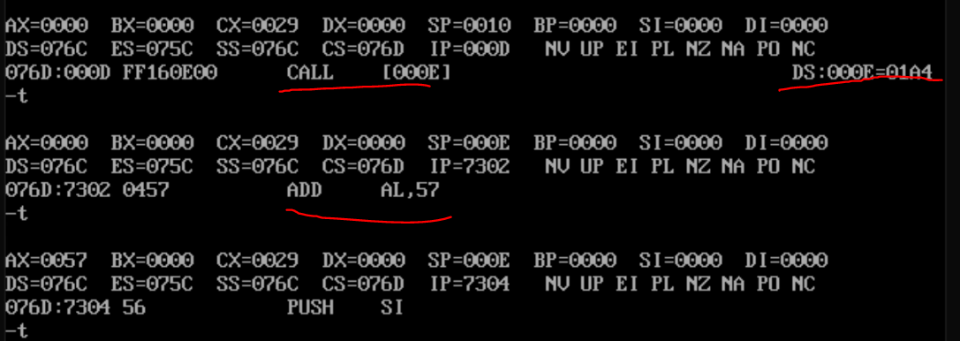
检测点10.5

（1）

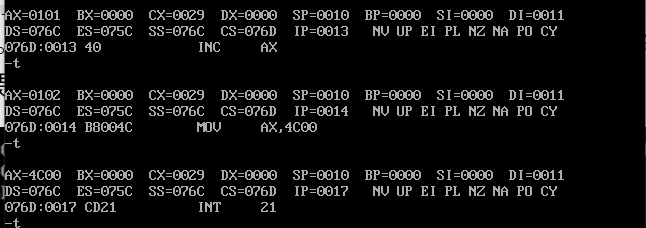
若根据源程序运行，ax应为3。

在执行call指令后，下一地址（即第一个inc ax的地址）的ip应该入栈，尔后程序执行jmp ds:[0EH] ，即程序从 ds:[0EH] 中取出数据赋值给 ip，而ds:[0EH]所指正为刚刚入栈的ip地址。于是又重新返回到第一个inc ax处，继续执行命令。

但是根据debug结果，发现并不是这样，那个栈位置被其他命令占据。



ax也被改写为0057h，随后按照地址顺序一直往下执行，直到遇见出口才结束。



ax=0102h

（2）

若根据程序执行，ax=1，bx=0。

在执行过程中，s的地址首先进入栈的低地址，即ss:[0]，之后cs的地址也进入ss:[2]

之后执行call指令，因为入栈操作是从高地址开始，所以cs与下一操作NOP的ip的地址分别存在了栈的高位，之后跳转进入ss:[0]，因为s:[0]存有ss的ip信息，于是其ip改为s的ip，进入s。

之后ax被赋予s的偏移地址ip，之后ax与ss:[0ch]（入栈的NOP的偏移地址）做减法，因为s比nop高一位，所以结果为1。之后bx被赋予cs的地址，与ss:[0eh]（入栈的cs地址），结果为0。

实验十

代码如下：

assume cs:code

data segment

db 'Hello world',0 ;修改成welcome to masm 也是可以的，通用，可移植性好。

data ends

code segment

start:

mov dh,24 ;dh 行号，0-24

mov dl,8 ;dl 列号，0-79

mov cl,1 ;cl 颜色

mov ax,data

mov ds,ax

mov si,0 ;ds:si指向字符串首地址

call show\_str

mov ax,4c00h

int 21h

show\_str:

push dx

push ax

push di

push es

push cx

push ds

push si

;实际行号

mov ax,160

mul dh ;行号偏移地址存在al中

mov dh,0

;实际列号

dec dl

add dl,dl ;列偏移地址存在dl中

add ax,dx

mov di,ax ;总体偏移地址在di中

mov ax,0b800h

mov es,ax ;所以es:[di]指向显存中的第一个位置

mov ah,cl ;颜色存在ah中

mov cx,0

s:

mov al,ds:[si]

mov cl,al

jcxz over

mov es:[di],ax

inc si

add di,2

jmp short s

over:

pop si

pop ds

pop cx

pop es

pop di

pop ax

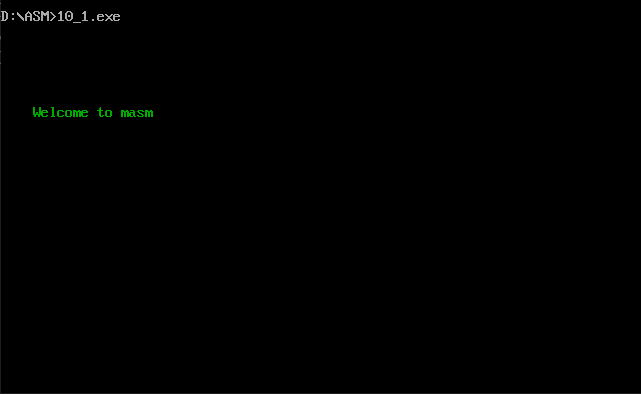
pop dx

ret

code ends

end start

结果如下：



（2）

代码如下：

assume cs:code

data segment

dw 8 dup(0)

data ends

code segment

start:

; mov ax,data

; mov ds,ax

mov ax,4240h ;被除数低16位

mov dx,000fh ;被除数高16位

mov cx,0ah ;除数

call divdw

mov ax,4c00h

int 21h

divdw:

push ax

mov ax,data

mov ds,ax

mov ax,dx ;被除数：dx存放高位,ax存放低位，现在只计算H/N

mov dx,0

div cx ;ax存放着商，dx存放着余数

mov ds:[0],ax ;左侧数据的高位，也是所求表达式的商的高位

;对与右侧数据而言，余数dx的值为高位数据，L（原始被除数低16的值）

;仅为直观 mov dx,dx

pop ax ;右侧数据的被除数的低位

push ax ;再恢复元数据

div cx ;余数在dx（高位），低位仍为ax;ax存放着商，dx存放着余数

mov ds:[2],ax ;此内存单元的值为所求表达式的商的低位

mov ds:[4],dx ;此内存单元的值为所求表达式的余数

pop ax

mov ax, ds:[2]

mov dx, ds:[0]

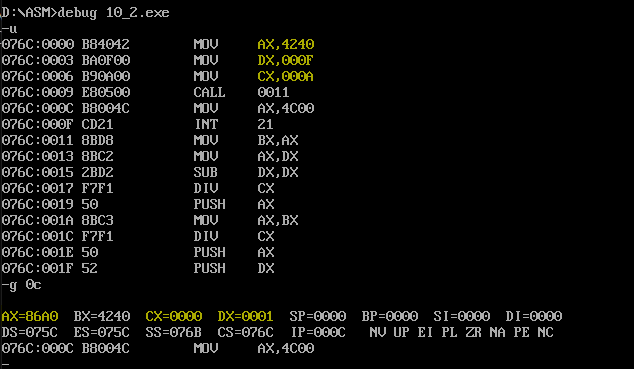
mov cx, ds:[4]

ret

code ends

end start

结果如下：



（3）

代码如下：

*assume* cs:code

data segment

  dw 123,12666,1,8,3,38

data ends

dataend segment

  db 32 dup (0)

dataend ends

stack segment

 dw 24 dup (0)

stack ends

code segment

start:

*mov* ax,data

*mov* ds,ax

*mov* di,0

*mov* ax,stack

*mov* ss,ax

*mov* sp,48

*mov* ax,dataend

*mov* es,ax

*mov* si,0

*mov* cx,6

s:

*call* divw

*dec* cx

    jcxz del

*inc* cx

*jmp* next

del:

*inc* cx

*dec* si      *;divw函数结束，指向dataend段中最后一个逗号*

*mov* al,0

*mov* es:[si],al  *;将该逗号去掉，将值修改为0,即字符串结束标志*

next:

*loop* s

*mov* dh,8        *;在8行3列显示为绿色*

*mov* dl,3

*mov* cl,2

*call* show\_str

*mov* ax,4c00h

*int* 21h

divw:   *;将data中的每一个十进制数值拆分存储在dataend段中*

*;12666=317aH,需要16位除法,被除数32位,dx高位，ax低位，ax商，dx余数*

*push* ax

*push* bx

*push* cx

*push* dx

*push* es

*mov* cx,0

*mov* ax,ds:[di]

*mov* bx,10

divnei:

*mov* dx,0

*div* bx

*add* dl,30h

*push* dx     *;存放余数（逆序）*

*inc* cx      *;用cx记录数据的位数，为pop计数*

*push* cx     *;存放cx,防止因使用jcxz而导致的cx被修改*

*mov* cx,ax

    jcxz divneiover

*pop* cx      *;内部未结束,继续计数*

*jmp* short divnei

divneiover: *;一个数据分离完毕，进行存放*

*;cx已经对数据的位数计数，直接使用*

*pop* cx  *;内部结束，结束计数*

    divcun:

*pop* ax

*mov* es:[si],al

*inc* si

*loop* divcun

*;一个数据处理完毕，隔一个逗号*

*mov* al,2ch

*mov* es:[si],al  *;逗号的ascii是2ch*

*inc* si

*;一个数据处理完毕，loop到下一个数据,divwai开始*

*inc* di

*inc* di      *;word型数据，加两字节*

*;数据处理完毕，返回原程序处*

*pop* es

*pop* dx

*pop* cx

*pop* bx

*pop* ax

*ret*

show\_str:

*push* dx

*push* ax

*push* di

*push* es

*push* cx

*push* ds

*push* si

*mov* ax,dataend

*mov* ds,ax

*mov* si,0

*;实际行号*

*inc* dh

*mov* ax,160

*mul* dh

*mov* dh,0

*;实际列号*

*dec* dl

*add* dl,dl

*add* ax,dx

*mov* di,ax

*mov* ax,0b800h

*mov* es,ax

*mov* ah,cl

*mov* cx,0

s:

*mov* al,ds:[si]

*mov* cl,al

    jcxz over

*mov* es:[di],ax

*inc* si

*add* di,2

*jmp* short s

over:

*pop* si

*pop* ds

*pop* cx

*pop* es

*pop* di

*pop* ax

*pop* dx

*ret*

code ends

end start

结果如下：

