汇编实验作业

——320150938841 陈冠斌

目录

[实验二 2](#_Toc483031947)

[I. 2](#_Toc483031948)

[实验三 4](#_Toc483031949)

[I. 4](#_Toc483031950)

[II. 5](#_Toc483031951)

[实验四 6](#_Toc483031952)

[I. 6](#_Toc483031953)

[II. 8](#_Toc483031954)

[III. 9](#_Toc483031955)

[IV.自编：输入小写/大写字母，输出大写字母 9](#_Toc483031956)

[V.自编：输入‘$’则结束，输入小写/大写字母，转变为大写/小写字母，其它字符不变，输出字符。 10](#_Toc483031957)

[实验五 12](#_Toc483031958)

[I. 12](#_Toc483031959)

[II.两个数a,b的三种关系(<,=,>)，并按格式输出 13](#_Toc483031960)

[III.输出问候信息，没有输出名字。 16](#_Toc483031961)

[IV. 输出问候信息，且输出名字。 18](#_Toc483031962)

[IV.自编：比较两个字符串的大小 22](#_Toc483031963)

[实验六 25](#_Toc483031964)

[I.优化——十位数字为0则不显示 25](#_Toc483031965)

[II. 27](#_Toc483031966)

[III. 28](#_Toc483031967)

[IV.程序已优化修改 29](#_Toc483031968)

[实验七 34](#_Toc483031969)

[I. 34](#_Toc483031970)

[实验八 35](#_Toc483031971)

[I. 35](#_Toc483031972)

# 实验二

## I.

Code：

assume cs:codesg

codesg segment

mov ax,2000h

mov ss,ax

mov sp,10h

add sp,4

push ax

push bx

pop ax

pop bx

mov ax,4c00h

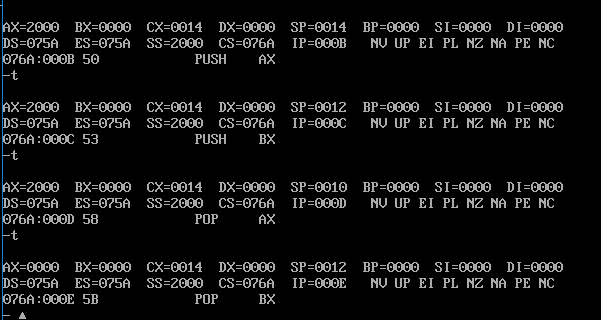
int 21h

codesg ends

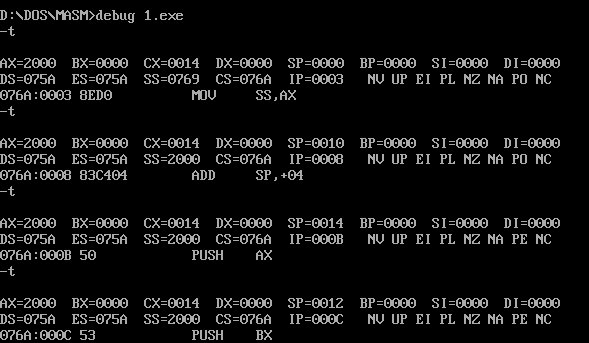
end

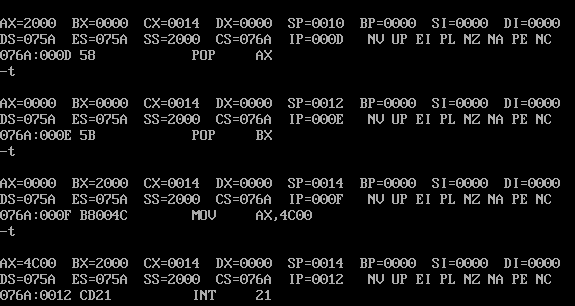
Analysis：

初始时ax=0FFFFh，bx=0。程序实现的是ax和bx的交换。



每一步执行后寄存器的内容：





# 实验三

## I.

Code：

code segment

assume cs:code

g1:mov ah,1

int 21h ;input to al

cmp al,'$'

jz exit ;if al='$' then exit

mov dl,al

mov ah,2 ;output from dl

int 21h

jmp g1 ;cycle

exit:mov ah,4ch ;just need ah,don't need al

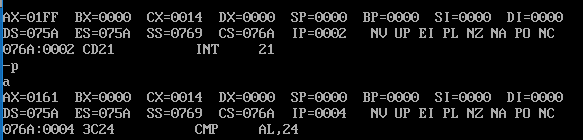
int 21h ;-p to check

code ends

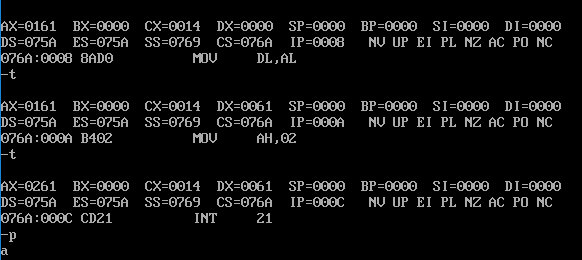
end g1

Analysis：

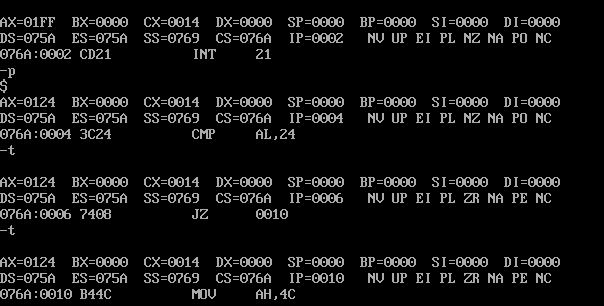
1.输入'a'，al变为61(十进制是97，即'a'的ascii)



2.从dl输出一个字节的内容，为'a'



3.当读入‘$’，IP从0006转到0010



Result：



## II.

Code：

data segment

buf db 'Hello',13,10,'END$'

data ends

stack segment

db 100 DUP (0)

stack ends

code segment

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

begin: mov ax,data

mov ds,ax

lea bx,buf ;bx=PianYi address of buf

la:mov dl,[bx]

cmp dl,'$'

jz exit

cmp dl,'a'

jb k

cmp dl,'z'

ja k

sub dl,20h ;if dl in ['a'~'z'],then sub 20h(change into DaXie)

k:mov ah,2

int 21h

inc bx

jmp la

exit:mov ah,4ch

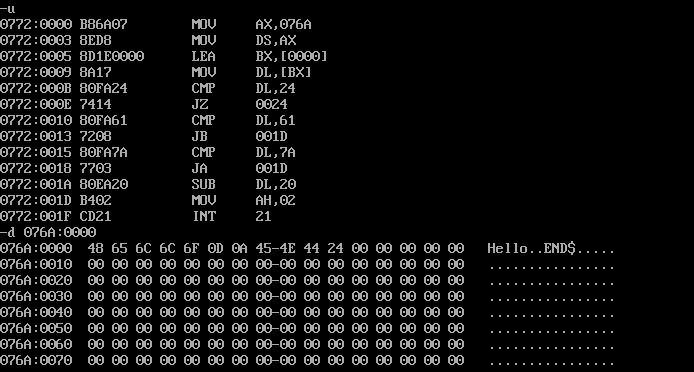
int 21h

code ends

end begin

Analysis：

1.通过-u获得data的偏移地址，然后通过-d data\_address:0000看到数据段的内容，其中字符是转化为ascii存储的，而数字却不会



2.字符占据一个内存单元，而程序是按照一个内存单元(inc bx)进行处理的。而数据段中13,10是换行符，其中13,10是十进制，转换为十六进制是0dh,0ah

Result：



# 实验四

## I.

Code：

assume cs:code

code segment

mov ax,cs

mov ds,ax

mov ax,0020h

mov es,ax

mov bx,0

mov cx,offset a

s:mov al,[bx] ;xx = ds:0000

mov es:[bx],al ;xx:xx

inc bx

loop s

a:mov ax,4c00h

int 21h

code ends

end

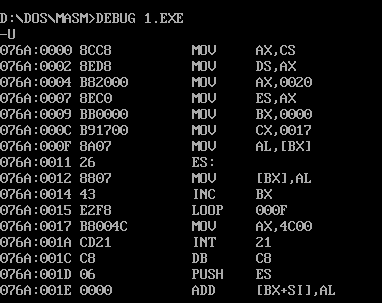
Analysis：

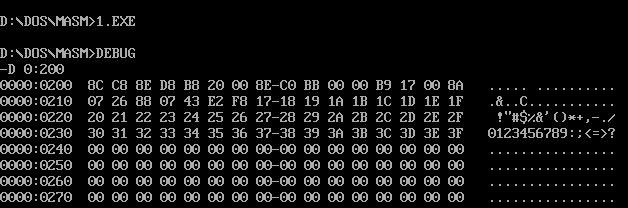
1.代码段的偏移首地址为0，mov cx,offset a为cx=a的偏移地址，而每次复制1行，总共复制偏移地址为0~cx-1的行，即为题目目的：将mov ax,4c00h之前的指令进行拷贝。

2.mov al,[bx] [bx]指的是ds:0000，若没有特殊说明，默认为ds

mov es:[bx],al 寄存器:[bx](偏移地址)

Result：





## II.

Code：

assume cs:code

code segment

mov ax,0020h

mov ds,ax

mov bx,0

mov cx,64

s:mov ds:[bx],bl

inc bx

loop s

mov ah,4ch

int 21h

code ends

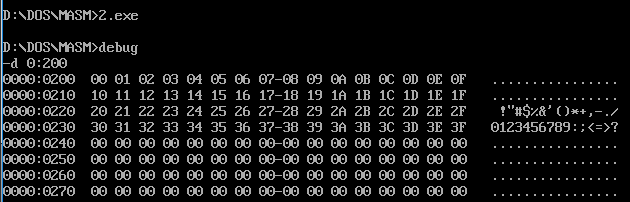
end

Analysis：

1.偏移地址和数值是相同的，但是偏移地址是两个字节，数值是一个字节，所以一个用bx，一个用bl，从而节约内存

2.9条指令包含：退出指令(2),设置基地址(ds)和偏移地址(bx)，设置循环次数(cx)，循环

Result：



## III.

Code：

stack segment

db 200 dup(0)

stack ends

code segment

assume cs:code,ss:stack

begin:mov ah,1

int 21h

sub al,20h

mov dl,al

mov ah,2

int 21h

mov ah,4ch

int 21h

code ends

end begin

Analysis：



## IV.自编：输入小写/大写字母，输出大写字母

Code：

stack segment

db 200 dup(0)

stack ends

code segment

assume cs:code,ss:stack

begin:mov ah,1

int 21h

and al,11011111B

mov dl,al

mov ah,2

int 21h

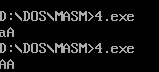
mov ah,4cH

int 21h

code ends

end begin

Result：



## V.自编：输入‘$’则结束，输入小写/大写字母，转变为大写/小写字母，其它字符不变，输出字符。

Code：

code segment

assume cs:code

begin:mov ah,1

int 21h

;input '$' is the symbol of the end of program

cmp al,24h ;'$':24h

jz exit

cmp al,41h ;'A'

jb output ;<'A'

cmp al,5Ah ;'Z'

jna upper ;'A'~'Z'

cmp al,7Ah ;'z'

ja output ;>'z'

cmp al,61h ;'a'

jnb lower ;'a'~'z'

loop output ;>'Z' and <'a'

upper: ;change to lowercase

or al,00100000B

loop output

lower: ;change to uppercase

and al,11011111B

loop output

output: ;output a modified char

mov dl,al

mov ah,2

int 21h

loop begin

exit:mov ah,4ch

int 21h

code ends

end begin

Analysis：

对于不同情况分别处理

Result：



对于V，另外的方法：找到小写/大写字母，xor 00100000(左数第三位：小写为1，变为0，即大写；大写为0，变为1，即小写)

Code：

code segment

assume cs:code

begin:mov ah,1

int 21h ;input from screen(al)

cmp al,24h ;'$'

jz exit

cmp al,41h ;'A'

jb output ;<'A'

cmp al,5Ah ;'Z'

jna operator ;'A'~'Z'

cmp al,61h ;'a'

jb output ;>'Z' and <'a'

cmp al,7Ah ;'z'

ja output ;>'z'

;left 'a'~'z'

operator: ;'A'~'Z' and 'a'~'z'

xor al,00100000B

output:mov dl,al

mov ah,2

int 21h ;output from screen(dl)

loop begin

exit:mov ah,4Ch

int 21h

code ends

end begin

Result：



# 实验五

## I.

Code：

stack segment stack

dw 256 dup(?)

stack ends

data segment

a db 10h

b db 30h

data ends

code segment

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

start:mov ax,data

mov ds,ax

mov a,al

mov b,al

cmp a,al

jge age

mov dl,'B'

mov bl,'A'

jmp disp

age:mov dl,'A'

mov bl,'B'

disp:mov ah,2

int 21h

mov dl,'>'

int 21h

mov dl,bl

int 21h

mov ah,4ch

int 21h

code ends

end start

Analysis：

固定输出模式，输出(dl)>(bl),当A>B,(dl)=A,(bl)=B；否则(dl)=B,(bl)=A。但题目是错的，忽略了等号情况

Result：



## II.两个数a,b的三种关系(<,=,>)，并按格式输出

Code：

stack segment stack

dw 256 dup(?)

stack ends

data segment

a db 10h

b db 30h

data ends

code segment

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

start:mov ax,data

mov ds,ax

;output 'A= '

mov ah,2

mov dl,'A'

int 21h

mov dl,'='

int 21h

mov dl,' '

int 21h

;input A

mov ah,1

int 21h

mov a,al

;output ' B= '

mov ah,2

;Space

mov dl,20h

int 21h

mov dl,'B'

int 21h

mov dl,'='

int 21h

mov dl,' '

int 21h

;input B

mov ah,1

int 21h

mov b,al

;compare

cmp a,al

ja bigger

jb smaller

jmp equal

bigger:mov bl,'>'

jmp output

smaller:mov bl,'<'

jmp output

equal:mov bl,'='

output:mov ah,2

;HuanHang:\r\n

mov dl,0Ah

int 21h

mov dl,0Dh

int 21h

mov dl,a

int 21h

mov dl,bl

int 21h

mov dl,b

int 21h

mov ah,4ch

int 21h

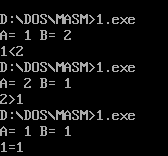
code ends

end start

Analysis：

输入两次，从而第一个的数值要放在内存中保留。根据比较的结果，到达不同的位置，实现不同的输出

Result：



Flowchart：

输出'<'

输出'='

输出'>'

输出a

比较a,b大小

输出b

## III.输出问候信息，没有输出名字。

Code：

stack segment stack

db 64 dup(?)

stack ends

data segment

buff db 50,?,50 dup(?)

;\n\r

mes1 db 0dh,0ah

db 'What is your name?:$'

mes2 db 0dh,0ah

db 'hello!','$','abc'

data ends

code segment

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

start:mov ax,data

mov ds,ax

;output mes1

mov dx,offset mes1

mov ah,9

int 21h

;input to buff

mov dx,offset buff

mov ah,0ah

int 21h

;output mes2

mov dx,offset mes2

mov ah,9

int 21h

mov ah,4ch

int 21h

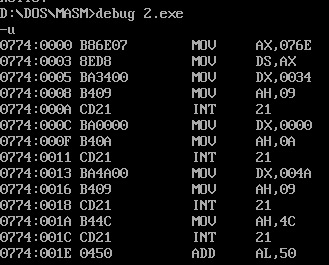
code ends

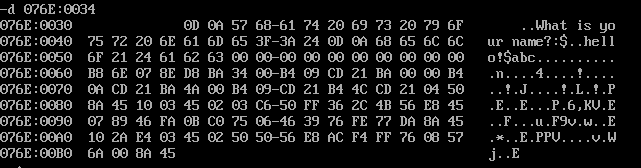
end start

Analysis：

1.mov dx,offset mes1：获得mes1的偏移地址

mov ah,9 int 21h：显示器显示字符串，DS：DX置字符串首址，字符串以'$'结束





2.同一数据段的数据是连接起来的。而如buff的标识符，作用是获得某一位置的地址。可以理解为是通过标识符，分成很多个区间(字符串)，但在内存分配上事实上并不是这样，它就是一个连续的内存空间。

## IV. 输出问候信息，且输出名字。

Code：

stack segment stack

db 64 dup(?)

stack ends

data segment

buff db 50,?,50 dup(?)

;\n\r

mes1 db 0dh,0ah

db 'What is your name?:$'

mes2 db 0dh,0ah

db 'hello!','$','abc'

data ends

code segment

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

start:mov ax,data

mov ds,ax

;output mes1

mov dx,offset mes1

mov ah,9

int 21h

;input to buff

mov dx,offset buff

mov ah,0ah

int 21h

;output mes2

mov dx,offset mes2

mov ah,9

int 21h

;output buff

;need to add "exit character" '$'

;wrong: mov [bl],'$'

; ds:

mov bh,00h

mov si,offset buff

mov bl,[si+1]

mov al,'$'

mov [bx+2],al

;output ' '

mov dl,' '

mov ah,2

int 21h

mov dx,offset buff

;ignore first and secend character

add dx,2

mov ah,9

int 21h

mov ah,4ch

int 21h

code ends

end start

Analysis：

1.buff db 50,?,50 dup(?)

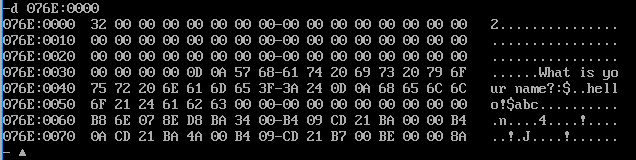
mov ah,10 int 21h：键入并显示字符串，DS：DX置字符串首址，第1单元置允许键入的字符数(含一个结束符)，第2单元中置键入的实际字符数。

数据段中“？”代表空的单元，显示值为0，用于为某些数据预留存储单元。

第一个50为允许键入的字符数；第二个数需要输入字符串后得到，所以设置为“？”；然后50 dup(?)指的是连续50个值为0的数







2.对于代码

; ds:

mov bh,00h

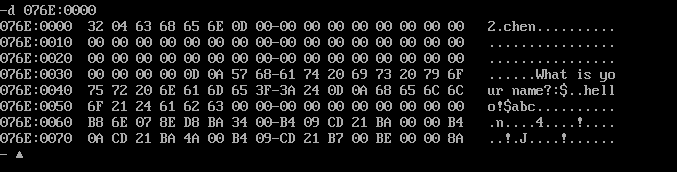
mov si,offset buff

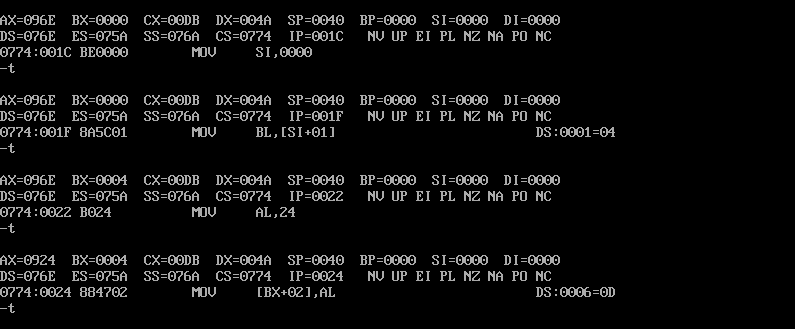
mov bl,[si+1]

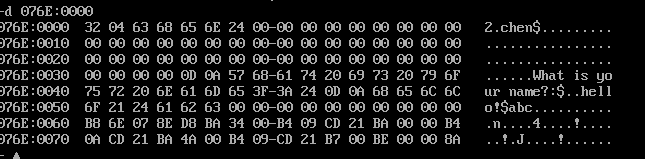
mov al,'$'

mov [bx+2],al

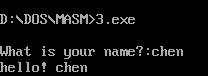
buff为数据段首地址，偏移地址值为0，其实可以直接mov bl,[si](si=1)。字符串的长度等于[1]的内容。字符串在data:0002~data:(0002+(data:0001)-1)，所以要在data: (0002+(data:0001))添加一个'$'(即为mov [bx+2],al)，从而使得字符串有结束符标志，实现后续字符串的输出。







Result：



另外输出字符串的方法(不加结束标识符,获得字符串长度，循环输出)：

stack segment stack

db 64 dup(?)

stack ends

data segment

buff db 50,?,50 dup(?)

;\n\r

mes1 db 0dh,0ah

db 'What is your name?:$'

mes2 db 0dh,0ah

db 'hello!','$','abc'

data ends

code segment

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

start:mov ax,data

mov ds,ax

;output mes1

mov dx,offset mes1

mov ah,9

int 21h

;input to buff

mov dx,offset buff

mov ah,0ah

int 21h

;output mes2

mov dx,offset mes2

mov ah,9

int 21h

;output ' '

mov dl,' '

mov ah,2

int 21h

mov ch,0

lea si,[buff+2]

mov bx,offset buff

mov cx,[bx+1]

mov ch,0

;mov ah,2

s:mov dl,[si]

int 21h

inc si

loop s

mov ah,4ch

int 21h

code ends

end start

## IV.自编：比较两个字符串的大小

Code：

stack segment stack

db 100 dup(?)

stack ends

data segment

buf1 db 50,?,50 dup(?)

buf2 db 50,?,50 dup(?)

tip1 db 0dh,0ah

db 'Buf1= ','$'

tip2 db 0dh,0ah

db 'Buf2= ','$'

data ends

code segment

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

start:mov ax,data

mov ds,ax

;output tip1

mov dx,offset tip1

mov ah,9

int 21h

;input to buf1

mov dx,offset buf1

mov ah,0ah

int 21h

;add '$' to buf1

mov bh,00h

mov si,dx

mov bl,[si+1]

mov al,'$'

mov [bx+si+2],al

mov cx,bx

;output tip2

mov dx,offset tip2

mov ah,9

int 21h

;input to buf2

mov dx,offset buf2

mov ah,0ah

int 21h

;add '$' to buf1

mov bh,00h

mov si,dx

mov bl,[si+1]

mov al,'$'

mov [bx+si+2],al

;output buf1

mov ah,2

mov dl,0dh

int 21h

mov dl,0ah

int 21h

mov dx,offset buf1

add dx,2

mov ah,9

int 21h

;cx=min(length(buf1),length(buf2))

cmp cx,bx

jna next

mov cx,bx

next:

;compare each position

mov si,offset buf1

mov di,offset buf2

mov bx,2

s:mov al,[bx+si]

cmp al,[bx+di]

ja bigger

jc smaller

add bx,1

loop s

;judge whose len is cx

mov al,[bx+si]

cmp al,'$'

jne bigger

mov al,[bx+di]

cmp al,'$'

jne smaller

same:mov dl,'='

jmp output

bigger:mov dl,'>'

jmp output

smaller:mov dl,'<'

jmp output

output:

;output symbol

mov ah,2

int 21h

;output buf2

mov dx,offset buf2

add dx,2

mov ah,9

int 21h

mov ah,4ch

int 21h

code ends

end start

;ab abc

;abc ab

;abc abc

;ab d

;d ab

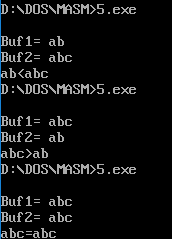
Analysis：

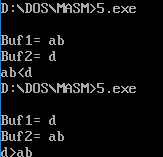
1.给两个字符串添加结束标识符

2.依次比较两个字符串的每一位(通过偏移地址)

3.若同一位置的字符为小于/大于关系，则结果为小于/大于关系

4.直到比较到‘$’(较小数的该位字符为'$')，此时比较长度，若有字符串该位不为‘$’，则该字符串长度必然更长(另外一个字符串到了该位就结束了)，而前面两个字符串相等，所以该字符串更大；若两个字符串该位都为'$'，则大小相等。





# 实验六

## I.优化——十位数字为0则不显示

Code：

stack segment

db 200 dup(0)

stack ends

code segment

assume cs:code,ss:stack

start:

;input x

mov ah,1

int 21h

mov bl,al

;output '\n'

mov ah,2

mov dl,0dh

int 21h

;mov ah,2

mov dl,0ah

int 21h

;input y

mov ah,1

int 21h

;change to ascii

sub al,30h

sub bl,30h

;multiply: ax=al\*bl

;ax=al\*bl<10\*10=100

mul bl

;ax div 10 result: Shang:al,Yu:ah

;ax<100 , thus al(Shang)<10 ; ah(Yu)<ChuShu=10

mov bl,10

div bl

;change to number

add ax,3030h

;program need to use ax, thus store the information of ax by "bx=ax"

mov bx,ax

;output '\n'

mov ah,2

mov dl,0dh

int 21h

;mov ah,2

mov dl,0ah

int 21h

;output two numbers

mov ah,2

;if bl=0 ,don't input

cmp bl,30h

jz output

mov dl,bl

int 21h

output:mov dl,bh

int 21h

mov ah,4ch

int 21h

code ends

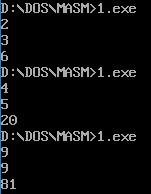
end start

Analysis：

首先(ax)=(al)\*(bl)，即两个数的乘积。(ax)<10\*10=100,

其次(al)=(ax)/10,(ah)=(ax)%10。(al)<100/10=10，(ah)<除数=10。

所以(al)和(ah)的值都大于等于0，小于10，这种情况下，用10进制和16进制的结果都是一样的。



## II.

Code：

assume cs:code

code segment

start:mov ah,1

int 21h

;'\n'

cmp al,0dh

jz tc

;'0'~'9'

cmp al,'0'

jb next

cmp al,'9'

ja a

mov dl,al

mov ah,2

int 21h

jmp start

;'A'~'Z'

a:cmp al,41h

jb next

cmp al,5ah

ja b

c:mov dl,'C'

mov ah,2

int 21h

next:jmp start

;'a'~z'

b:cmp al,61h

jb next

cmp al,7ah

ja next

jmp C

tc:mov ah,4ch

int 21h

code ends

end start

Analysis：

1.若输入换行，则退出。注意是0dh而不是0ah。

mov ah,1

int 21h

cmp al,0dh

jz tc

2.程序是按照ascii大小顺序(升序)进行判断，先'0'~'9'，再‘A’~'Z'，最后'a'~‘z’。

Result：



## III.

Code：

stack segment

dw 64 dup(?)

stack ends

code segment

assume cs:code,ss:stack

;26 characters 'A'~'Z'

start:mov cx,001ah

mov bl,41h

mov ah,02h

a1:mov dl,bl

int 21h

inc bl

push cx

;loop 0fffh times

mov cx,0fffh

a2:loop a2

pop cx

dec cx

;if cx<>0 , jmp a1 the combination of dec & jnz

jnz a1

mov ah,4ch

int 21h

code ends

end start

Analysis：

1.push cx，pop cx是因为程序在中途使用cx，会改变cx的值，而我们需要原来cx的值。所以用栈把cx入栈，在需要的时候出栈cx。

2.mov cx,0fffh。注意不能是fffh，字母不能为首字母。如ffff需要写成0ffff。

3.mov cx,0fffh

a2:loop a2

为不执行任何指令，循环0fff次。

用于：

1.当上一个指令执行需要较长时间，我们需要等待

2.输入显示慢一点。或者输出一个数，执行这两行语句，等待一段时间，让人有时间看清楚，再输出另外一个数

实际上，删除这两句对这个代码没影响

Result：



## IV.程序已优化修改

Code：

stack segment

db 64 dup(?)

stack ends

code segment

assume cs:code,ss:stack

start:

;output '\n'

mov dl,0dh

mov ah,2

int 21h

mov dl,0ah

int 21h

mov bl,0

next:mov ah,2

;output ShiWei

mov dl,bl

mov cl,4

shr dl,cl ;div 16(2^4)

or dl,30h

int 21h

;output GeWei

mov dl,bl

and dl,0fh ;mod 16(2^4,(0f+1))

or dl,30h

mov ah,2 ;change to character

int 21h

mov al,bl

inc al

daa

cmp al,30h

jnc finish

mov bl,al

;'0':'f' can't show on the first place

;loop ffff times

mov cx,0ffffh

delay:loop delay

jmp next

finish:mov ah,4ch

int 21h

code ends

end start

Analysis：

1.一个十六进制s的数转化为十进制t

t的十位数：s/16，即shr al,cl

t的个位数：s%16，即and dl,0fh

2.or dl,30h

指的是增加30h，因为dl本来是0~9，所以可以这样做。但是若dl & 30h<>0,则增加小于30，不能这样做。

3.mov cx,0ffffh

delay: loop delay

这样做使得一个一个输出，否则全部直接显示(计算速度很快)

4.daa

这里是把0a变为10，1a变为20，2a变为30，然后退出

网上解释：

十进制数加调整指令DAA(Decimal Adjust After Addition)

该指令是用于调整AL的值，该值是由指令ADD或ADC运算二个压缩型BCD码所得到的结果。

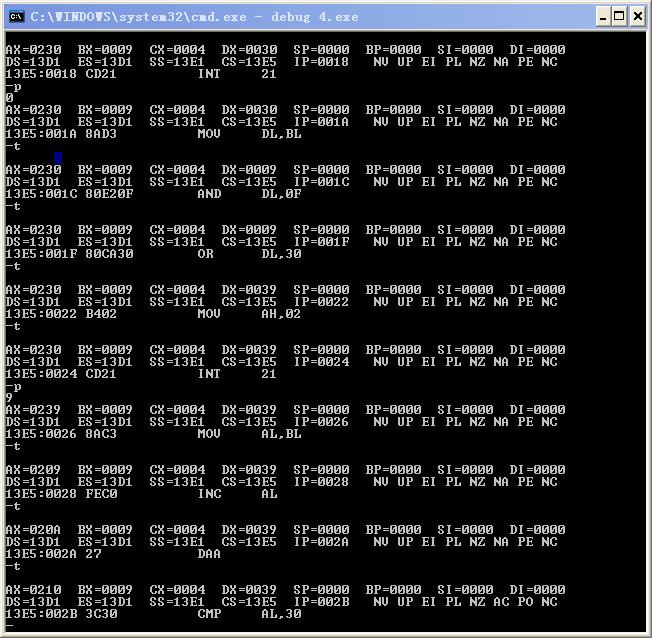
压缩型BCD码是一个字节存放二个BCD码，低四位和高四位都是一个BCD码。

其调整规则如下：

如果AL的低四位大于9，或标志位AF=1，那么，AL=AL+6，并置AF=1；

如果AL的高四位大于9，或CF=1，那么，AL=AL+60H，并置CF=1；

如果以上两点都不成立，则，清除标志位AF和CF。



5.关于调试

特殊情况：10,20,30

但是要执行很久……

可以一开始设置数为9，然后循环一次就到了特殊情况



另外的方法：

判断个数数字，若为9，则bl的值增加7(注意是7不是6)。

stack segment

db 64 dup(?)

stack ends

code segment

assume cs:code,ss:stack

start:

;output '\n'

mov dl,0dh

mov ah,2

int 21h

mov dl,0ah

int 21h

mov bl,0

next:mov ah,2

;output ShiWei

mov dl,bl

mov cl,4

shr dl,cl ;div 16(2^4)

add dl,30h

int 21h

;output GeWei

mov dl,bl

and dl,0fh ;mod 16(2^4,(0f+1))

add dl,30h

mov ah,2 ;change to character

int 21h

;JinWei

cmp dl,39h ;number '9' (30 number'0')

jz special

inc bl

jmp judge

special:add bl,7

;when bl=30 , exit

judge:cmp bl,30h

jz finish

mov cx,0ffffh

delay:loop delay

jmp next

finish:mov ah,4ch

int 21h

code ends

end start

# 实验七

## I.

Code：

code segment

assume cs:code

mov ah,6

mov al,0

mov ch,0

mov cl,0

mov dh,24

mov dl,79

mov bh,7

int 10h

poscurse:

mov ah,2

mov dh,16

mov dl,30

mov bh,0

int 10h

mov cx,20

getchar:

mov ah,1

int 21h

cmp al,'Q'

jz exit

loop getchar

mov ah,6

mov al,1

mov ch,8

mov cl,30

mov dh,16

mov dl,50

mov bh,7

int 10h

jmp poscurse

exit:mov ah,4ch

int 21h

code ends

end

# 实验八

Code：

## I.

stack segment

dw 100 dup(?)

stack ends

data segment

buff dw 3c6dh

data ends

code segment

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

start:mov ax,data

mov ds,ax

lea si,buff

mov bx,[si]

mov ch,4

a1:mov cl,4

rol bx,cl

mov al,bl

;just exist the last position

and al,0fh

;change to ascii

add al,30h

;if '0'~'9',not add 7h;otherwise 'A'~'F', add 7h(10(3Ah)->A(41h))

cmp al,3ah

jl a2

add al,7h

a2:mov dl,al

mov ah,2

int 21h

dec ch

jnz a1

mov ah,4ch

int 21h

code ends

end start

Analysis：

1.

dec ch

jnz al

mov ah,4ch

int 21h

指的是若执行完dec ch后，ch为0，则不跳转，直接结束程序



2.rol，ror

rol 数字，位数

数字(二进制)循环左移/右移位数

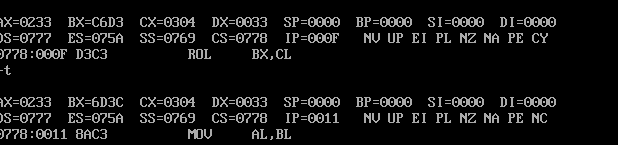
mov cl,4

rol bx,cl

在十六进制下循环左移一位，需要左移4位

每次执行这两条语句，当前最后一位分别为千位，百位，十位，个位

如1234->2341->3412->4123->1234



Result：

