****

IBM-PC汇编语言

实验报告

|  |  |
| --- | --- |
| 实验内容： | 分类统计字符串个数 |
| 姓 名： | 乔 楠 |
| 学 号： | 0919170210 |
| 专业班级： | 物联网工程1702班 |
| 指导教师： | 贺建飚 |

**实验二 查找匹配字符串**

1. **设计说明**

1.1实验目的

(1)掌握循环程序的设计方法。

(2)掌握汇编语言源程序的编辑、汇编、连接及调试过程。

(3)进一步熟悉利用DEBUG程序修改参数的方法，并检查和验证结果的正确性。

(4) 学会针对不同的问题，选用不同的组织循环的方法。

1.2实验要求

掌握循环程序的过程和汇编实现原理，进一步熟练掌握DEBUG操作指令以及汇编语言源程序的编辑、汇编、连接及调试过程。

1.3主要符号说明

定义输出字符串,均为提示用语

mess1 db 'Enter Keyword:','$'

mess2 db 'Enter Sentence:','$'

mess3 db 'Match at location ','$'

mess4 db 'No Match!','$'

mess5 db 'Match!',0dh,0ah,'$'

mess6 db 'h of the sentence.',0dh,0ah,'$'

mess7 db 'h of the sentence.',0dh,0ah,'$'

定义byte类型数据结构，大小为10,用作存储keyword字符串

stoknin1 label byte ;定义输入字符串

max1 db 10

act1 db ?

stokn1 db 10 dup(?);改动

stoknin2 label byte

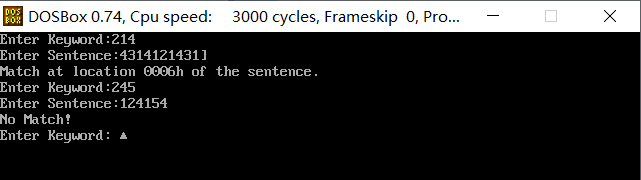
max2 db 50

act2 db ?

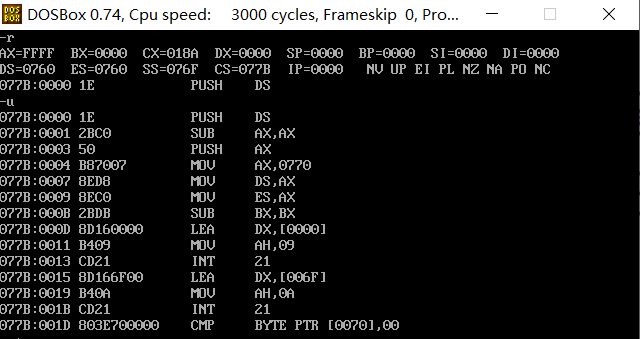
stokn2 db 50 dup(?);改动

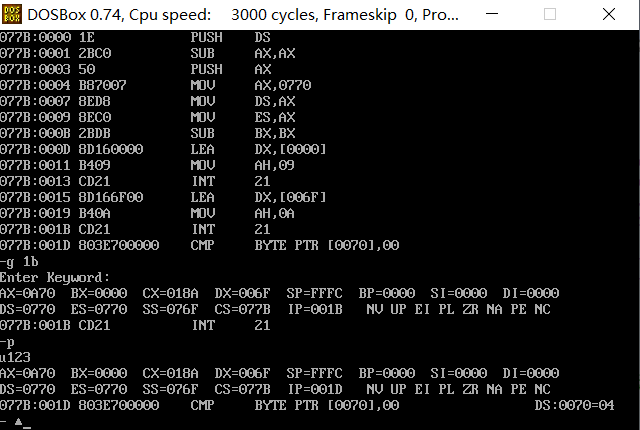
1. **调试说明**

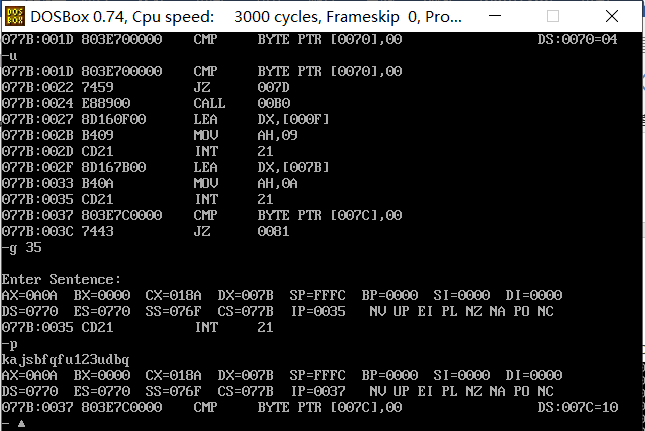
**2.1实验结果**

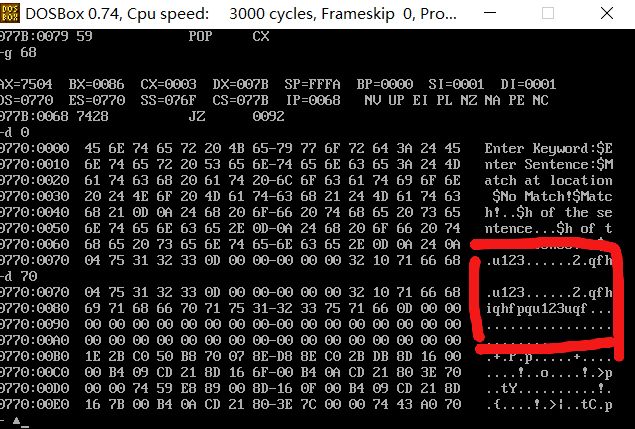
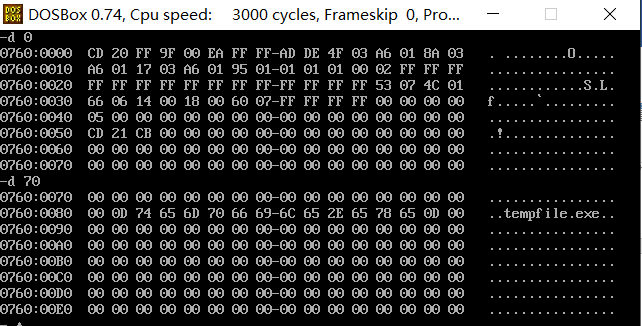
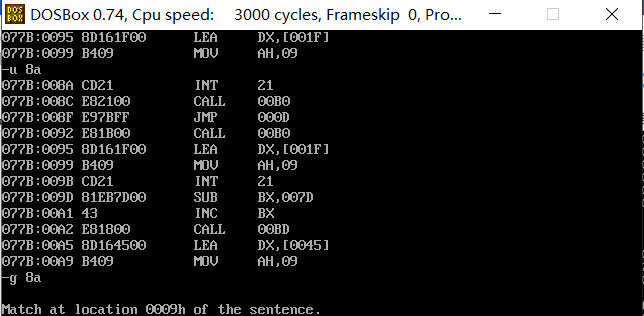


**2.2实验调试**









**2.3观察现象**

1. 数据段在代码运行前后显著变化，keyword的内容和sentence内容被写入数据段。

2. 在程序运行前，数据段的变量并没有进入数据段。，

**2.4实验心得**

与第一次实验相比这次用到了数据段，应该注意数据段的定义和使用。此外应该注意单个字符时的“边缘”情况，因为这时指针的移动和关键字的长度是一致的，但是这并不意味着，两个字符串匹配。同时，清晰地调试内容让我更深刻的理解数据段、代码段中运行的内部过程，在我眼前，代码逐渐透明化。这对我学习汇编语言十分有帮助！

1. **程序框图**



1. **程序清单**

掌握循环排序算法的思想：

查找匹配字符串：程序接收用户键入的一个关键字以及一个句子。如果句子中不包含关键字则显示‘NO match！’；如果句子中包含关键字则显示‘MATCH’，且把该字在句子中的位置用十六进制数显示出来。

**实验代码如下：**

;string match

;南有乔木

;;data

datarea segment

mess1 db 'Enter Keyword:','$' ;定义输出字符串

mess2 db 'Enter Sentence:','$'

mess3 db 'Match at location ','$'

mess4 db 'No Match!','$'

mess5 db 'Match!',0dh,0ah,'$'

mess6 db 'h of the sentence.',0dh,0ah,'$'

mess7 db 'h of the sentence.',0dh,0ah,'$'

;es

stoknin1 label byte ;定义输入字符串

max1 db 10

act1 db ?

stokn1 db 10 dup(?);改动

stoknin2 label byte

max2 db 50

act2 db ?

stokn2 db 50 dup(?);改动

datarea ends

prognam segment

main proc far

;代码段初始化

assume cs:prognam, ds:datarea, es:datarea

start::

push ds

sub ax,ax

push ax

mov ax,datarea

mov ds,ax

mov es,ax

sub bx,bx;

;输入关键字字符串

keyword:

lea dx,mess1

mov ah,09h

int 21h

lea dx,stoknin1

mov ah,0ah

int 21h

cmp act1,0

je exit

sentence:

call crlf

lea dx,mess2

mov ah,09h

int 21h

lea dx,stoknin2

mov ah,0ah

int 21h

cmp act2,0

je nmatch

mov al,act1

cbw

mov cx,ax

push cx

mov al,act2

sub al,act1

js nmatch;关键字多于查找字，必然不匹配！

inc al

mov di,0

mov si,0

lea bx,stokn2

compare1:

mov ah,[bx+di]

cmp ah,stokn1[si]

jne compare2;bx记得加一

inc si

inc di

;inc bx

dec cx;剩余关键字长度，实时更新

cmp cx,0

je match

jmp compare1

compare2:

inc bx;比较字符串往下走

dec al;剩余字符串可比较长度

cmp al,0

je nmatch

mov si,0;在compare1中增加，但是此时必须清空重新比较

mov di,0

pop cx;更新此刻寄存器的cx

push cx;但又不删除

jmp compare1;继续比较

exit:

call crlf

ret

;输出no match

nmatch:

call crlf

lea dx,mess4

mov ah,09h

int 21h

call crlf

jmp keyword

match:

call crlf

lea dx,mess3

mov ah,09h

int 21h

sub bx,offset stokn2

inc bx

call trans

lea dx,mess6

mov ah,09h

int 21h

jmp keyword

crlf proc near

mov dl,0dh

mov ah,2

int 21h

mov dl,0ah

mov ah,2

int 21h

ret

crlf endp

trans proc near

mov ch,4

rotate:

mov cl,4

rol bx,cl

mov al,bl

and al,0fh

add al,30h

cmp al,3ah

jl printit

add al,7h

printit:

mov dl,al

mov ah,2

int 21h

dec ch

jnz rotate

ret

trans endp

main endp

prognam ends

end start

**实验三 分类统计字符个数**

1. **设计说明**

**1.1实验目的**

掌握分支程序设计的原理与方法，重点掌握各种程序中形成和判断条件而产生的程序段的设计方法和技巧。

**1.2实验要求**

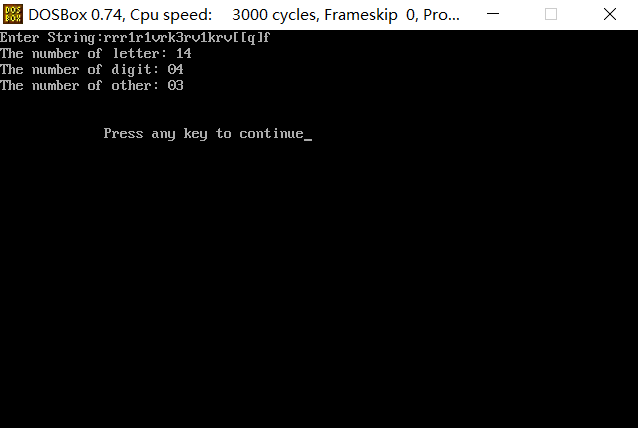
掌握分支程序的调用及基本子程序的编写，判断程序运行的顺序。

**1.3实验内容**

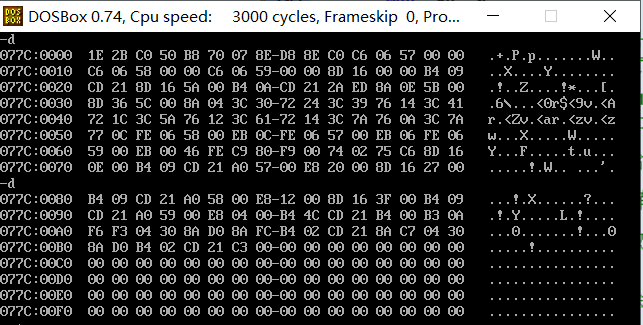
程序接受用户输入一行字符，并按照字符串中字符所属的类型分类，将结果输出（例如：aaabb1234&中，字母5，数字4，其他字符1）

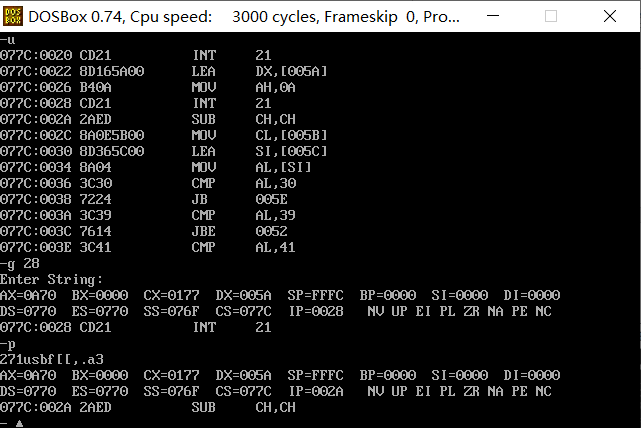
1. **调试说明**

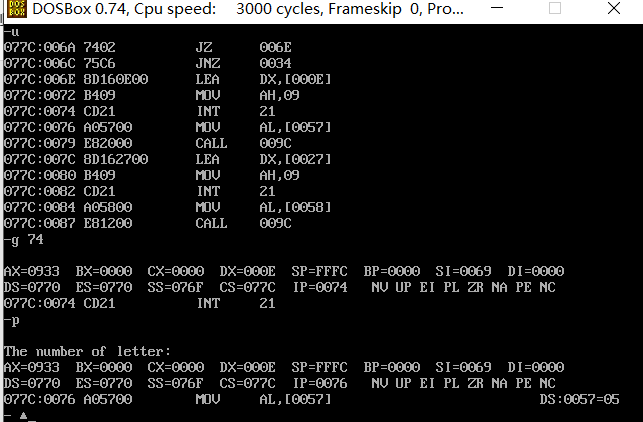
**2.1实验结果**

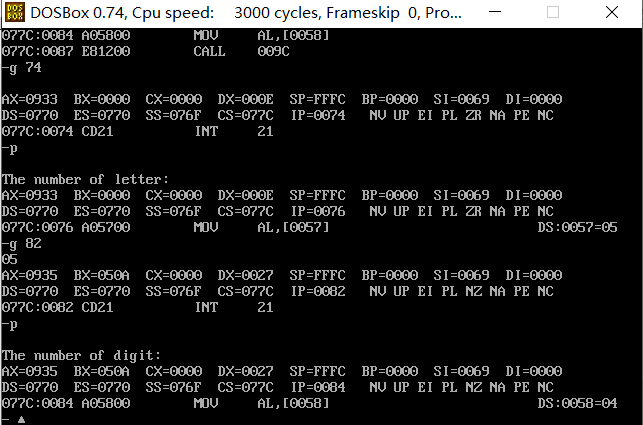


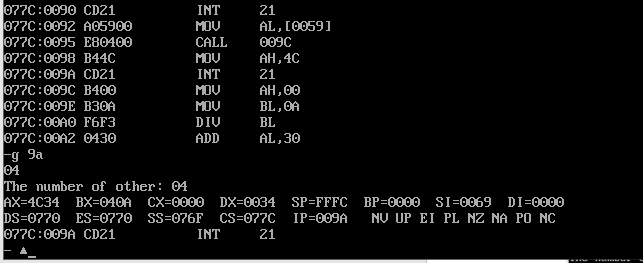
**2.2实验调试**

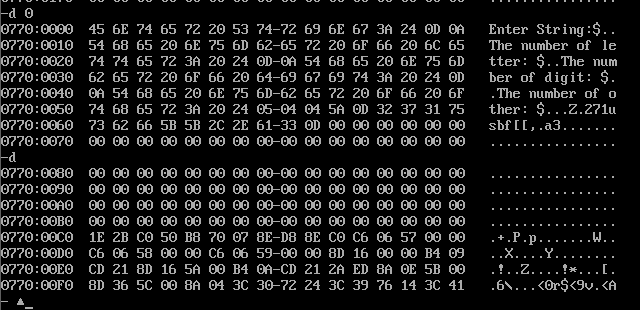


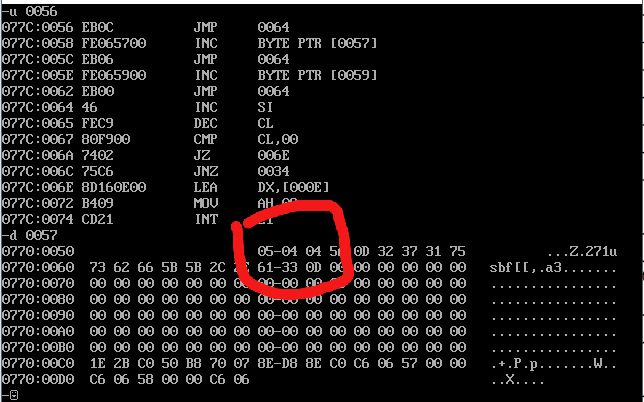












**2.3观察现象**

1. 在程序运行前，数据段的变量并没有进入数据段。

2. 在0770：0057 0058 0059处存有digit，letter，other的内容，分别为04，04，05

**2.4实验心得**

我认为本次实验主要是要清楚各分支程序。什么时候该跳转到哪个分支，并在分支中做好各项处理工作是本次实验的重点。

并且此次实验也让我们明白了，汇编的输出都是“字符”，也就是说要想输出两位数字必须做一点处理，将它转换为相应的字符“一个一个”地输出，从而解决了实验中的疑问。

还有一点就是关于输入的问题，我采用的是单字符输入，边输入边处理。因为用字符串的话，输入之后，仍然要“一个一个”字符取出来去做判断，还不如直接就判断了。

1. **程序框图**



1. **程序清单**

**实验代码如下：**

;string match

;南有乔木

;;data

datarea segment

mess1 db 'Enter String:','$' ;定义输出字符串

mess2 db 0dh,0ah,'The number of letter: ','$'

mess3 db 0dh,0ah,'The number of digit: ','$'

mess4 db 0dh,0ah,'The number of other: ','$'

letter db ? ;变量

digit db ?

other db ?

string label byte ;定义输入字符串

max db 90

act db ?

str1 db 90 dup(?);改动

datarea ends

prognam segment

;main proc far

assume cs:prognam, ds:datarea, es:datarea

start::

push ds

sub ax,ax

push ax

mov ax,datarea

mov ds,ax

mov es,ax

mov letter,0

mov digit,0

mov other,0

lea dx,mess1

mov ah,09h

int 21h

lea dx,string

mov ah,0ah

int 21h

sub ch,ch

mov cl,[string+1]

lea si,string+2

judge:

mov al,[si]

cmp al,'0'

jb othersg1

cmp al,'9'

jna digitsg1

cmp al,'A'

jb othersg1

cmp al,'Z'

jna lettersg1

cmp al,'a'

jb othersg1

cmp al,'z'

jna lettersg1

cmp al,'z'

ja othersg1

digitsg1:

inc digit

jmp loop1

lettersg1:

inc letter

jmp loop1

othersg1:

inc other

jmp loop1

loop1:

inc si

dec cl

cmp cl,0

jz print1

jnz judge

print1:

lea dx,mess2

mov ah,09h

int 21h

mov al,letter

call disp

lea dx, mess3

mov ah,09h

int 21h

mov al,digit

call disp

lea dx, mess4

mov ah,09h

int 21h

mov al,other

call disp

exit:

mov ah, 4ch

int 21h

disp: ;十进制数形式显示AL中的内容.

mov ah, 0

mov bl, 10

div bl ;div 无符号:div src 16位操作:商ax=(dx,ax)/src,余数dx

add al, 30h ;比如说al=15h,即21，表示letter数量，然后，这个过程就是，ax=0015h(21),除以bl，bl值为10

mov dl, al ;则除完了的结果为2余1，则ah=01,al=02,即ax=0102h;,那么al+30h即为表示该数字的ASCII码值，因为0的ASCII值为30h

mov bh, ah ;则ax=0132h,dl=32,bh=01;

mov ah, 02h ;显示输出dx,则显示32码对应的数字，2

int 21h

mov al, bh ;把01给al,然后算出ASCII码，然后给dx，然后显示

add al, 30h

mov dl, al ;除数B有 8位和16位两种，保存在一个reg寄存器里 或是内存单元中。

;被除数A 默认存放在AX中（16位以内） 或 AX和DX中（32位，DX存放高16位，AX存放低16位）

;结果： 如果除数B是8位，那么除法的结果AL保存商，AH保存余数，

;如果除数B是16位，那么除法的结果 AX保存商，DX保存余数。

mov ah, 02h

int 21h

ret

;---------------------------------------

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;main endp

prognam ends

end start

**实验四 查找电话号码**

1. **设计说明**

1.1实验目的与要求

学习用汇编语言设计与编写子程序。

题目：查找电话号码phone

1.2实验内容

(1) 要求程序建立一个可存放50项的电话号码表，每项包括人名(20个字符)及电话号码(8个字符)两部分；

(2) 程序可接收输入人名及相应的电话号码，并把它们加入电话号码表中；

(3) 凡有新的输入后，程序应按人名对电话号码表重新排序；

(4) 程序可接收需要查找电话号码的人名，并从电话号码表中查出其电话号码，再在屏幕上以如下格式显示出来。

name tel．

X X X X X X X X

1.3算法设计

关于查找电话号码的问题有多种程序能实现,但是所用的设计思想都是一样的,其实现的主要思想都是使用子程序结构.主程序的主要部分如下:

.显示提示符 Input name:

.调用子程序 input\_name 接收人名；

.调用子程序 stor\_name 把人名存入电话号码表tel\_tab中；

.显示提示符 Input a telephone number:

.调用子程序inphone接收电话号码,并把它存入电话号码表tel\_tab中；

.如输入已结束则调用name\_sort子程序对电话号码表按人名排序；

.显示提示符 Do you want a telephone number?(Y/N)；

.回答N则退出程序；

.回答Y则再显示提示符 name ；

.调用子程序 input\_name 接收人名；

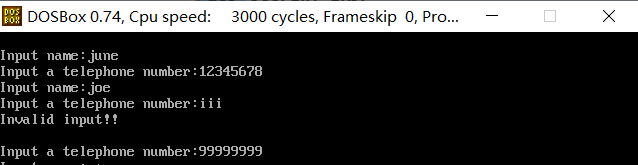
.调用子程序 name\_search在电话号码表中查找所要的电话号码；

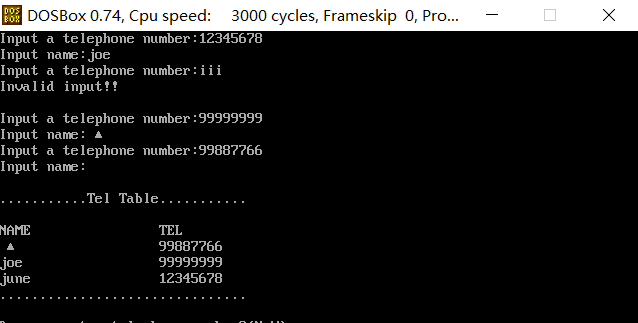
.调用子程序 printline按要求格式显示人名和电话号码；

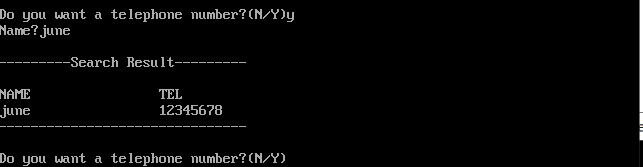
.重复查号提示符直至用户不再要求查号为止．

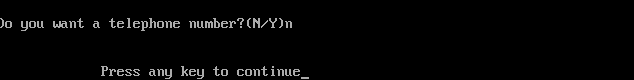
1. **调试说明**

**2.1实验结果**

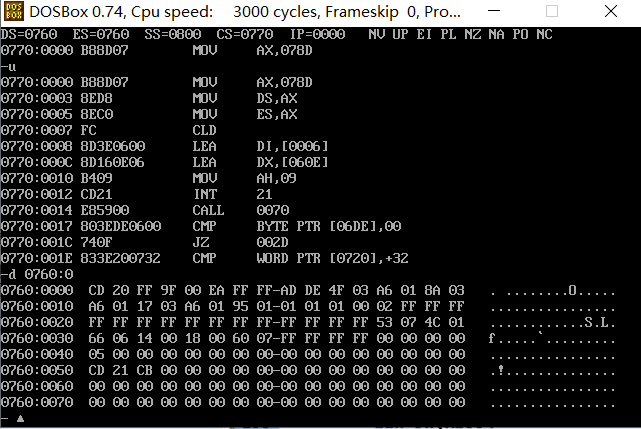


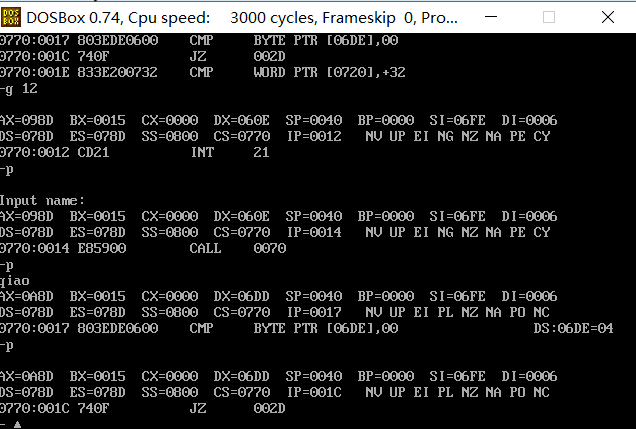


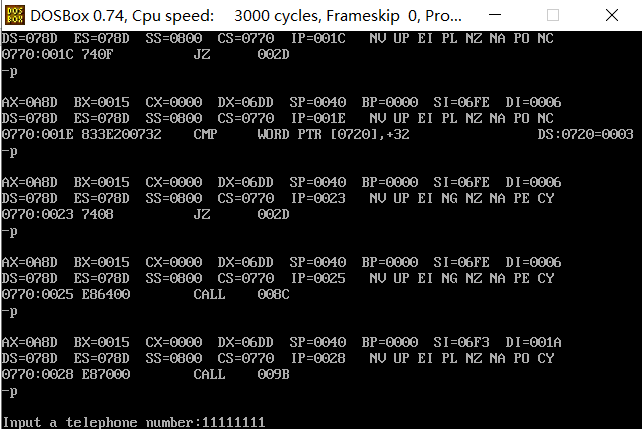


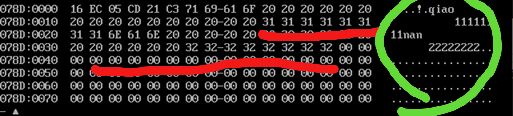


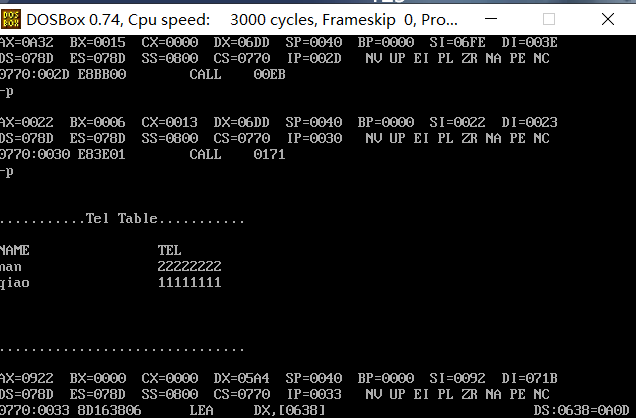
**2.2实验调试**











**2.3观察现象**

1. 在程序运行前，数据段的变量并没有进入数据段。

2. 在078D：0000中依次存入

**2.4实验心得**

我认为本次实验主要是要清楚各分支程序。什么时候该跳转到哪个分支，并在分支中做好各项处理工作是本次实验的重点。

并且此次实验也让我们明白了，汇编的输出都是“字符”，也就是说要想输出两位数字必须做一点处理，将它转换为相应的字符“一个一个”地输出，从而解决了实验中的疑问。

还有一点就是关于输入的问题，我采用的是单字符输入，边输入边处理。因为用字符串的话，输入之后，仍然要“一个一个”字符取出来去做判断，还不如直接就判断了。

1. **程序框图**



1. **程序清单**

**实验代码如下：**

;tele

;南有乔木

.MODEL SMALL

.STACK 64

;....................................................

.DATA

TEL\_TAB DB 50 DUP(20 DUP(?),8 DUP(?)) ;电话号码表（电话号码和人名）

MESS1 DB 0DH,0AH,0DH,0AH,11 DUP('.'),'Tel Table',11 DUP('.'),0DH,0AH,'$'

MESS2 DB 31 DUP('.'),0DH,0AH,'$'

MESS3 DB 0DH,0AH,0DH,0AH,9 DUP('-'),'Search Result',9 DUP('-'),0DH,0AH,'$'

MESS4 DB 31 DUP('-'),0DH,0AH,'$'

MESSNAME DB 0DH,0AH,'Input name:','$' ;提示输入姓名

MESSTEL DB 0DH,0AH,'Input a telephone number:','$' ;提示输入电话号码

MESSFIND DB 0DH,0AH,'Do you want a telephone number?(N/Y)','$';提示是否要进行查找操作

MESSERROR DB 0DH,0AH,'Invalid input!!',0DH,0AH,'$'

MESSN DB 0DH,0AH,'Name?','$' ;输入要查找的人名

MESSERROE DB 0DH,0AH,'Invlid input!please input again!','$' ;输入错误的提示

MESSDIS DB 0DH,0AH,'NAME',16 DUP(20H),'TEL',0DH,0AH,'$';输出号码表的格式

MESSNOTFIND DB 0DH,0AH,'The name is not in the table!!',0DH,0AH,'$'

INPUTNAME LABEL BYTE ;输入姓名的缓冲区

NMAXLEN DB 21

NAMELEN DB ?

I\_NAME DB 21 DUP(?)

INPUTTEL LABEL BYTE ;输入电话号码的缓冲区

TMAXLEN DB 9

TELLEN DB ?

I\_TEL DB 9 DUP(?)

TEMP DB 20 DUP(' '),8 DUP(' '),0DH,0AH,'$' ;输出电话表时借助的存储位置

TEMP\_TEL DW 0

COUNT DW 0 ;记录电话号码表的记录的个数

ENDADDR DW 0

SWAP DW 0

;....................................................

.CODE

MAIN PROC FAR

MOV AX,@DATA

MOV DS,AX

MOV ES,AX

CLD

LEA DI,TEL\_TAB

INPUT\_LOOP:

LEA DX,MESSNAME ;提示输入姓名

MOV AH,09H

INT 21H

CALL INPUT\_NAME ;输入姓名（调用输入姓名的子程序）

CMP NAMELEN,0 ;如果没有输入姓名则跳转

JZ NEXT

CMP COUNT,50 ;如果电话号码表满也跳转

JE NEXT

CALL STOR\_NAME ;调用保存姓名的子程序

CALL IPHONE ;输入电话号码和保存电话号码到电话号码表中

JMP INPUT\_LOOP ;循环输入电话号码和姓名

NEXT:

CALL NAME\_SORT

CALL DISPLAY ;输出电话号码表

SEARCH\_LOOP:

LEA DX,MESSFIND ;输入要进行查找的提示

MOV AH,09H

INT 21H

MOV AH,01H ;输入Y或y则查找，输入N或n则退出，其它则提示非法

INT 21H

CMP AL,'N'

JE EXIT

CMP AL,'n'

JE EXIT

CMP AL,'Y'

JE FIND

CMP AL,'y'

JE FIND

MOV AH,09H

LEA DX,MESSERROR

INT 21H

JMP SEARCH\_LOOP

FIND: MOV AH,09H

LEA DX,MESSN

INT 21H

CALL INPUT\_NAME ;输入要查找的姓名（调用输入姓名的子程序）

CALL NAME\_SEARCH ;调用查找姓名的子程序

CALL PRINTLINE

JMP SEARCH\_LOOP

EXIT: MOV AH,4CH

INT 21H

MAIN ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;输入姓名的子例程

INPUT\_NAME PROC NEAR

MOV AH,0AH

LEA DX,INPUTNAME ;输入姓名

INT 21H

MOV BH,0

MOV BL,NAMELEN

MOV CX,21

SUB CX,BX

NAME\_LOOP: MOV I\_NAME[BX],20H ;除了姓名以外的字符均以空格代替

INC BX

LOOP NAME\_LOOP

RET

INPUT\_NAME ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;把人名存入电话号码表tel\_tab的子例程

STOR\_NAME PROC NEAR

INC COUNT

CLD

LEA SI,I\_NAME ;将输入的姓名放入电话号码表中

MOV CX,10

REP MOVSW

RET

STOR\_NAME ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;接收电话号码，并把它存入电话号码表中的iphone子例程

IPHONE PROC NEAR

I\_START: LEA DX,MESSTEL ;提示输入电话号码提示

MOV AH,09H

INT 21H

LEA DX,INPUTTEL ;输入电话号码

MOV AH,0AH

INT 21H

MOV BX,0

MOV CH,0

MOV CL,TELLEN

CHECK: MOV AL,I\_TEL[BX] ;检查电话号码输入是否合法

INC BX

CMP AL,30H

JB I\_NEXT ;如果输入不合法就输出提示信息

CMP AL,39H

JA I\_NEXT ;如果输入不合法就输出提示信息

LOOP CHECK

MOV BH,0

MOV BL,TELLEN

MOV CX,9

SUB CX,BX

TEL\_LOOP:

MOV I\_TEL[BX],20H ;除了电话号码以外的字符都以空格代替

INC BX

LOOP TEL\_LOOP

CLD

LEA SI,I\_TEL ;将电话号码存放在电话号码表中

MOV CX,4

REP MOVSW

RET

I\_NEXT:MOV AH,09H

LEA DX,MESSERROR

INT 21H

JMP I\_START

IPHONE ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;NAME\_SORT子程序，此时DI已经指向了电话号码表中的最后一项的末尾

NAME\_SORT PROC NEAR

SUB DI,56

MOV ENDADDR,DI ;用ENDADDER来记录电话号码表中的倒数第二项的偏移地址

SORT\_NEXT1:MOV SWAP,0

MOV SI,OFFSET TEL\_TAB ;SI得到电话号码表的偏移地址

SORT\_NEXT2:MOV CX,20

CLD

MOV DI,SI

ADD DI,28 ;DI指向SI的下一个电话号码项

MOV AX,DI ;AX保存DI的值

MOV BX,SI ;BX保存SI的值

REPE CMPSB ;等于则重复

JBE SORT\_NEXT3 ;不用交换跳转

CALL CHANGE ;调用交换子例程

SORT\_NEXT3:MOV SI,AX ;SI指向上一个DI的位置

CMP SI,ENDADDR ;判断SI是否位于倒数第二个电话号码项

JBE SORT\_NEXT2

CMP SWAP,0

JNZ SORT\_NEXT1 ;如果后面的项发生了交换就要重新进行扫描了，知道没有交 ;换的项为止

RET

NAME\_SORT ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;子程序CHANGE交换两个字符串（为了便于说明设为串1和串2）的子程序

CHANGE PROC NEAR

CLD

MOV CX,14

MOV SI,BX ;SI指向串1

LEA DI,TEMP ;将串1的字符串(电话号码的一项）移到TEMP

REP MOVSW

CLD

MOV CX,14 ;此时SI已经移动到了串2的位置

MOV DI,BX ;将DI指向串1

REP MOVSW ;这时把串2移动到了串1中

CLD

MOV CX,14 ;此时DI已经指向了串2

LEA SI,TEMP ;SI指向TEMP

REP MOVSW ;将TEMP中的内容移到了串2中

MOV SWAP,1

RET

CHANGE ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;子程序NAME\_SEARCH，在电话号码表中查找所要的电话号码

NAME\_SEARCH PROC NEAR

LEA SI,TEL\_TAB ;si指向电话号码表

MOV BX,SI ;bx用于保存si的值

MOV DX,COUNT ;dx保存了count电话号码表的项数的值，用来控制查找的次数

SEA\_LOOP: LEA DI,I\_NAME ;di指向要查找的姓名

MOV CX,10

REPE CMPSW ;比较si di相等重复比较

JCXZ SEA\_EXIT ;当cx为0就跳转，此时找到了要查找的字符

ADD BX,28

MOV SI,BX ;si移动到电话号码表的下一项

DEC DX

JNZ SEA\_LOOP

NOTAB: MOV CX,-1 ;如果查找结束以后发现没有找到就设cx为-1

RET

SEA\_EXIT: MOV SI,BX ;保存查找到的电话号码记录到temp

LEA DI,TEMP

MOV CX,14

REP MOVSW

RET

NAME\_SEARCH ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;子例程DISPLAY,按要求显示人名以及电话号码

DISPLAY PROC NEAR

MOV AH,09H ;显示要输出电话号码表的提示信息

LEA DX,MESS1

INT 21H

MOV AH,09H

LEA DX,MESSDIS

INT 21H

LEA SI,TEL\_TAB ;si指向电话号码表的开始地址

MOV BX,COUNT ;bx保存了电话号码表的项数

LOOP\_DIS: CLD

LEA DI,TEMP ;di指向temp,将电话号码表中要输出的项保存在temp

MOV CX,14

REP MOVSW

MOV AH,09H

LEA DX,TEMP ;输出temp中的电话号码项

INT 21H

DEC BX

JNZ LOOP\_DIS

MOV AH,09H

LEA DX,MESS2

INT 21H

RET

DISPLAY ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;printline子例程，用于输出查找的结果

PRINTLINE PROC NEAR

MOV AH,09H

LEA DX,MESS3

INT 21H

CMP CX,-1

JE NOTFIND

MOV AH,09H

LEA DX,MESSDIS

INT 21H

MOV AH,09H

LEA DX,TEMP

INT 21H

JMP P\_EXIT

NOTFIND: MOV AH,09H

MOV DX,OFFSET MESSNOTFIND

INT 21H

P\_EXIT: MOV AH,09H

LEA DX,MESS4

INT 21H

RET

PRINTLINE ENDP

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

END MAIN

**实验五 Fibonacci**

1. **设计说明**

**1.1实验原理**

调用FIB函数求Fibonacci数,结果高位存在RESULT2H，低位存在RESULT2L，最后调用OUTPUT函数将结果以十进制的形式显示出来。

**1.2实验目的**

　　（1） 掌握循环程序的设计方法

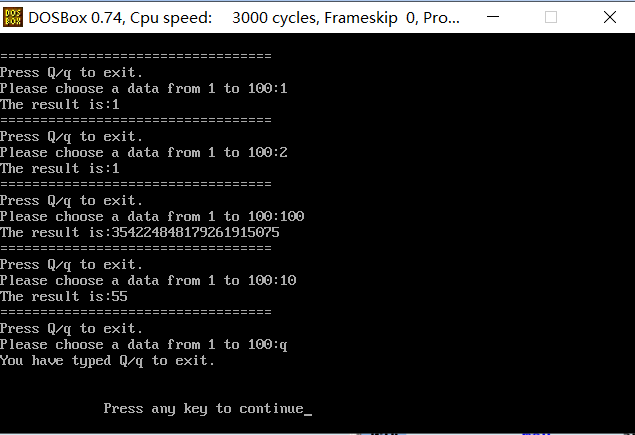
（2）掌握汇编语言源程序的编辑、汇编、连接及调试过程。

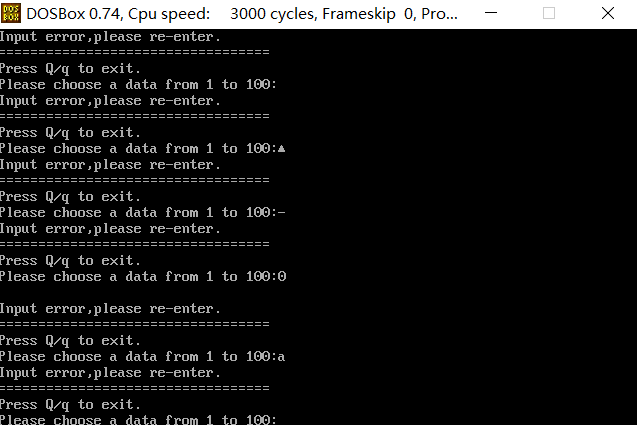
（3）进一步熟悉利用DEBUG程序修改参数的方法，并检查和验证结果的正确性。

（4） 学会针对不同的问题，选用不同的组织循环的方法。

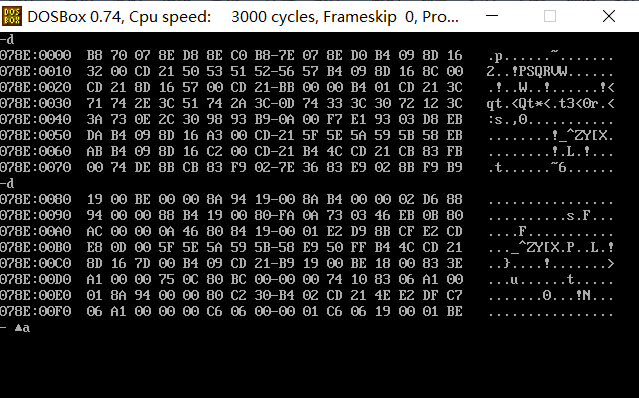
1. **调试说明**

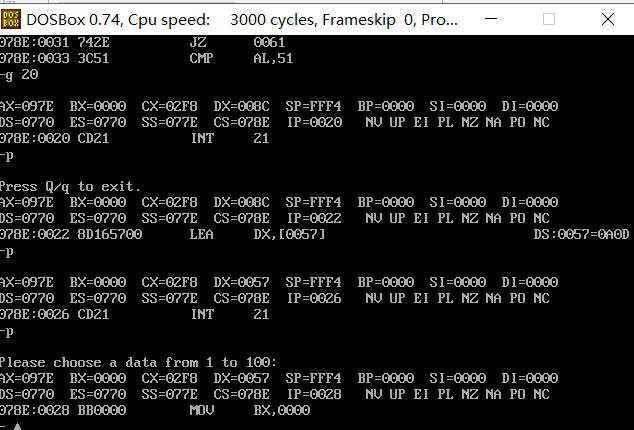
**2.1实验结果**

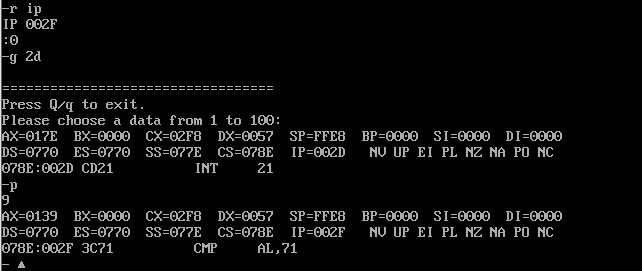


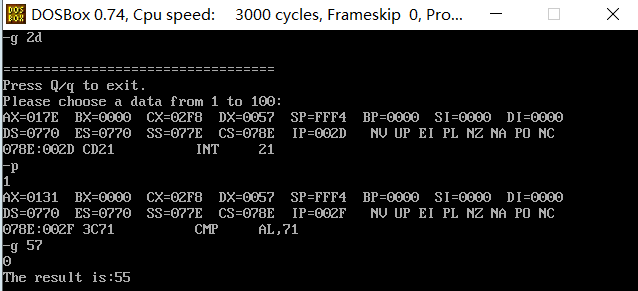


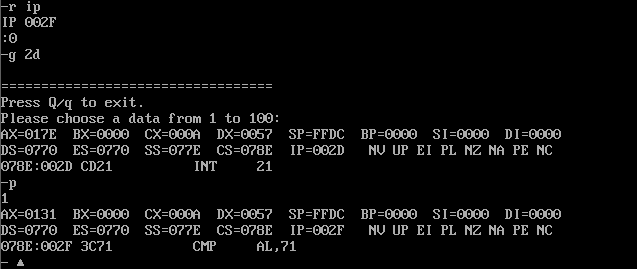
**2.2实验调试**



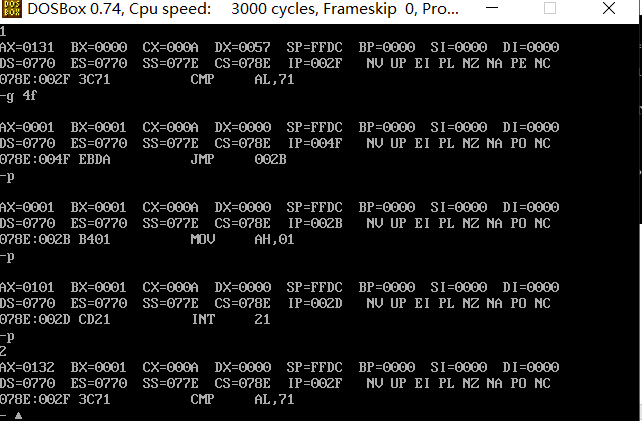




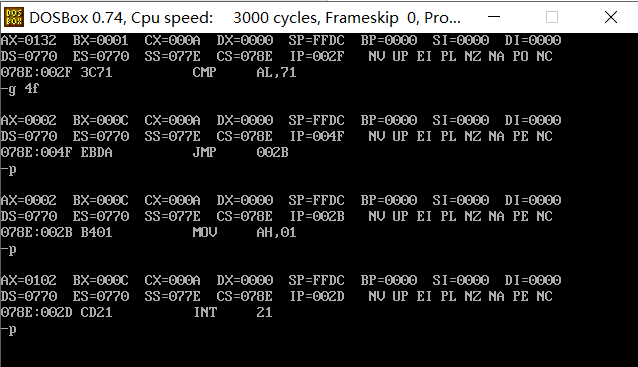




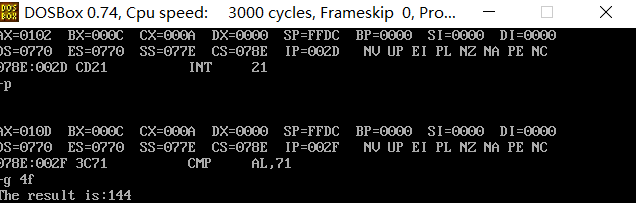
输入1



输入2

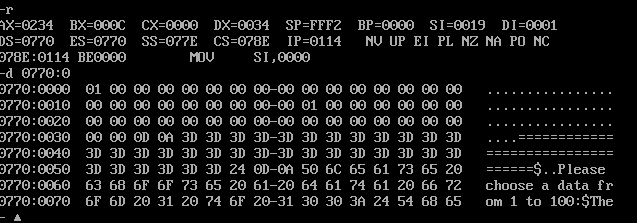


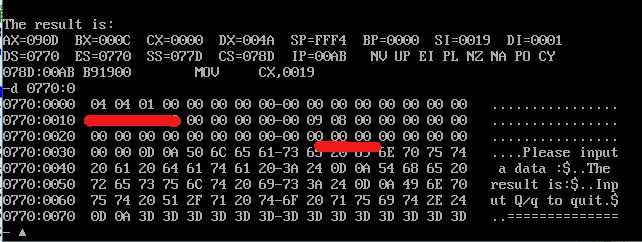
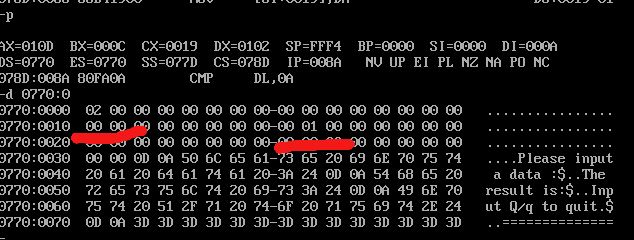
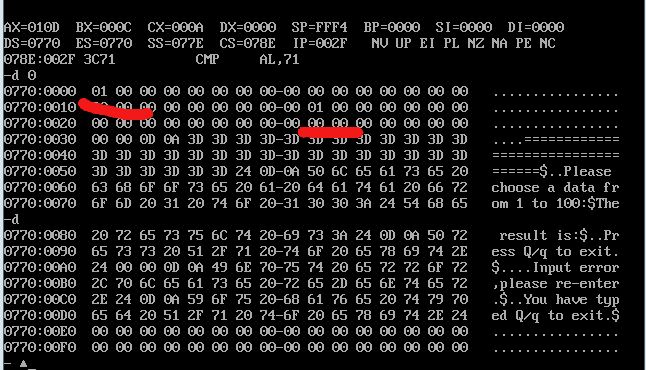
输入回车



结果为144

**以下为数据段中result，x处数据的变化**





**2.3观察现象**

1. 在程序运行前，数据段的变量并没有进入数据段。

2. 在0770：0000为result，0770：0019为x

3. 几次循环之后，数据存入内存。

**2.4实验心得**

我认为本次实验主要是要清楚各分支程序。什么时候该跳转到哪个分支，并在分支中做好各项处理工作是本次实验的重点。

并且此次实验也让我们明白了，汇编的输出都是“字符”，也就是说要想输出两位数字必须做一点处理，将它转换为相应的字符“一个一个”地输出，从而解决了实验中的疑问。

还有一点就是关于输入的问题，我采用的是单字符输入，边输入边处理。因为用字符串的话，输入之后，仍然要“一个一个”字符取出来去做判断，还不如直接就判断了。

1. **程序框图**



1. **程序清单**

**实验代码如下：**

;fibonicci

;南有乔木

datas segment

result db 1d,24 dup(0)

x db 1d,24 dup(0)

mess0 db 0dh,0ah,'==================================','$'

mess1 db 0dh,0ah,'Please choose a data from 1 to 100:','$'

mess2 db 'The result is:','$'

mess3 db 0dh,0ah,'Press Q/q to exit.','$'

flag dw ?

messerror db 0dh,0ah,'Input error,please re-enter.','$'

quit db 0dh,0ah,'You have typed Q/q to exit.','$'

datas ends

stacks segment

dw 128 dup(?)

stacks ends

codes segment

main proc far

assume cs:codes,ds:datas,ss:stacks,es:datas

start::

mov ax,datas ;ds寄存器设置到当前的数据段

mov ds,ax

mov es,ax

mov ax,stacks ;ss寄存器设置到当前堆栈段

mov ss,ax

reinput:

mov ah,09

lea dx,mess0

int 21h

push ax ;进栈

push bx

push cx

push dx

push si

push di

mov ah,09 ;显示操作符

lea dx,mess3

int 21h

lea dx,mess1

int 21h

;输入规模n，并且转换为十进制数

mov bx,0

input:

mov ah,01

int 21h

cmp al,'q' ;判断输入是否为q

jz end\_ ;跳转到end\_

cmp al,'Q'

jz end\_

cmp al,0dh

jz cont

cmp al,'0' ;输入是否为数字

jb error ;跳转到error

cmp al,':'

jnb error

sub al,30h

cbw

xchg ax,bx

mov cx,10d

mul cx

xchg ax,bx

add bx,ax

jmp input

error: ;提示出错信息

mov ah,09

lea dx,messerror

int 21h

pop di ;出栈

pop si

pop dx

pop cx

pop bx

pop ax

jmp reinput ;跳转回开始状态

end\_: ;提示已退出程序信息

mov ah,09

lea dx,quit

int 21h

mov ah,4ch

int 21h

ret

cont: cmp bx,0

jz error

mov cx,bx ;输入的数放到cx中

cmp cx,2

jle call\_f

;如果小于等于2，就直接输出结果

sub cx,2 ;否则以cx-2作为外循环，把两个数相加

next: mov di,cx

mov cx,25

mov si,0

adde: mov dl,x[si]

mov dh,result[si]

add dl,dh ;将两个存储单元的数中进行相加

mov result[si],dl

mov x[si],dh ;将上次的相加结果放入X中

cmp dl,10d

jae great ;如果大于10D，则转移

inc si

jmp goon

great: sub result[si],10d ;将尾数存入本单元

inc si

add x[si],1 ;高位加1

goon: loop adde

mov cx,di

loop next

call\_f:

call fibp

pop di ;出栈

pop si

pop dx

pop cx

pop bx

pop ax

jmp reinput ;跳转回开始状态，进行下一次计算

mov ah,4ch

int 21h

main endp

fibp proc near

print: lea dx,mess2 ;输出显示提示

mov ah,09

int 21h

mov cx,25

mov si,24

display1: cmp flag,0 ;标志位判断输出的高位是否为0

jnz n2

cmp result[si],0

jz n3

add flag,1

n2: mov dl,result[si] ;以十进制输出

add dl,30h

mov ah,02h

int 21h

n3: dec si

loop display1

mov flag,0

mov result[0],1d

mov x[0],1d

mov si,1

mov cx,24

initial: mov result[si],0

mov x[si],0

add si,1

loop initial

mov si,0

ret

fibp endp

codes ends

end start