****

IBM-PC汇编语言

实验报告

|  |  |
| --- | --- |
| 实验内容： | 查找匹配字符串 |
| 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

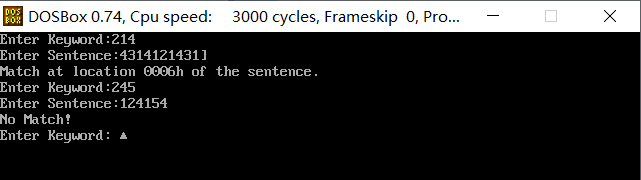
max2 db 50

act2 db ?

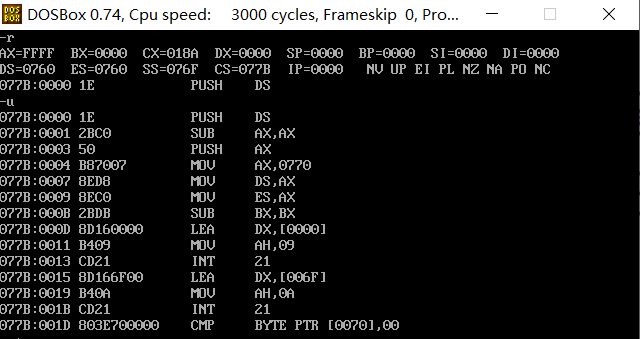
stokn2 db 50 dup(?);改动

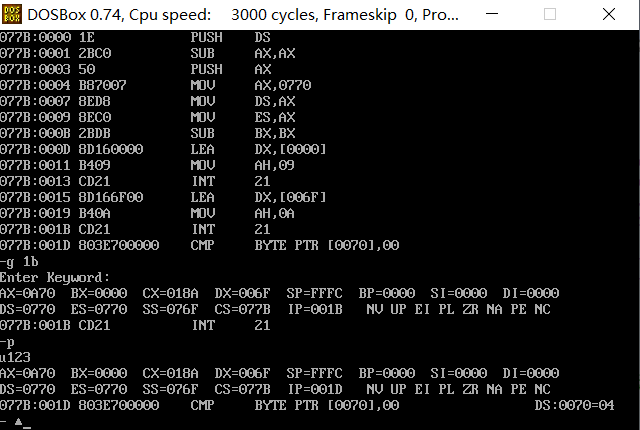
1. **调试说明**

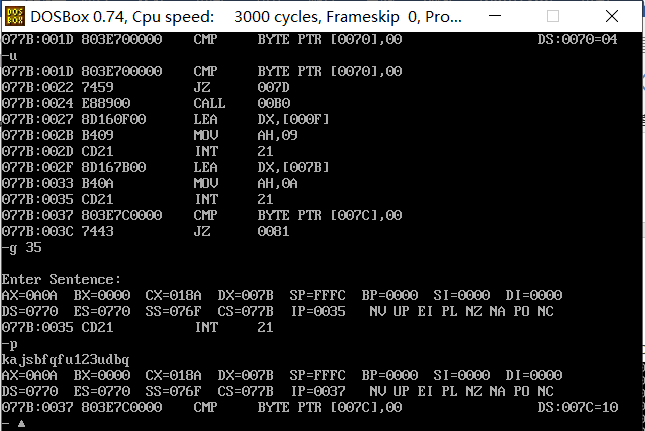
**2.1实验结果**

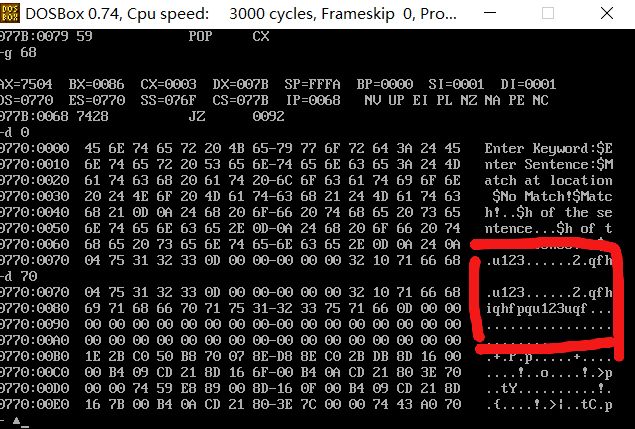
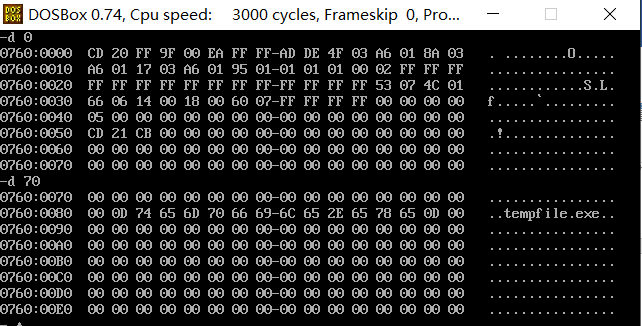
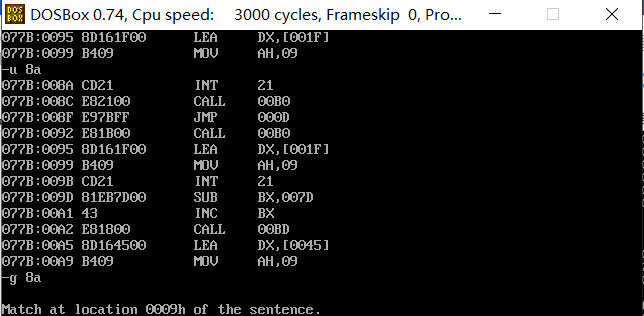


**2.2实验调试**









**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