汇编语言程序设计课程实验报告

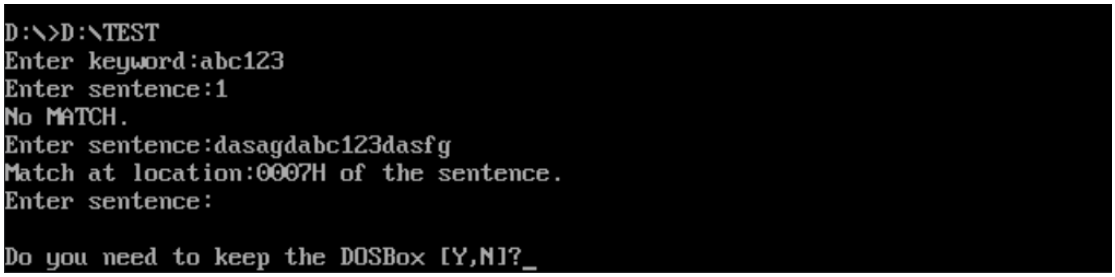
实验 5

姓名		院系	计算机工程与科学学院	学号	
实验目的：					
查找匹配字符串 SEARCH。					
实验要求：					
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!”，且把该句子中的位置用十六进制数显示出来。					

1. 程序分析

实验的基本思路是使用多个判断，两重循环的方式控制程序，首先先判断关键字与句子长度是否合法，为 0 则直接结束程序，其次判断句子长度是否小于关键字长度，小于则直接输出不匹配，不进入字符比较部分；然后使用 di，si 寄存器进行变址开始逐位比较，如果发现不匹配，更新回退句子变址寄存器 si，判断句子剩下的字符串长度是否足够匹配，如足够，则更新 cx，bi 的值，继续回到内层循环匹配，不足够则输出不匹配，清空寄存器内容开始新一轮的输入。如果字符匹配成功，则判断 cx 是否为 0，为 0 则说明匹配成功，输出成功匹配信息，情况寄存器内容转入新一轮的输入循环。

2. 程序运行结果

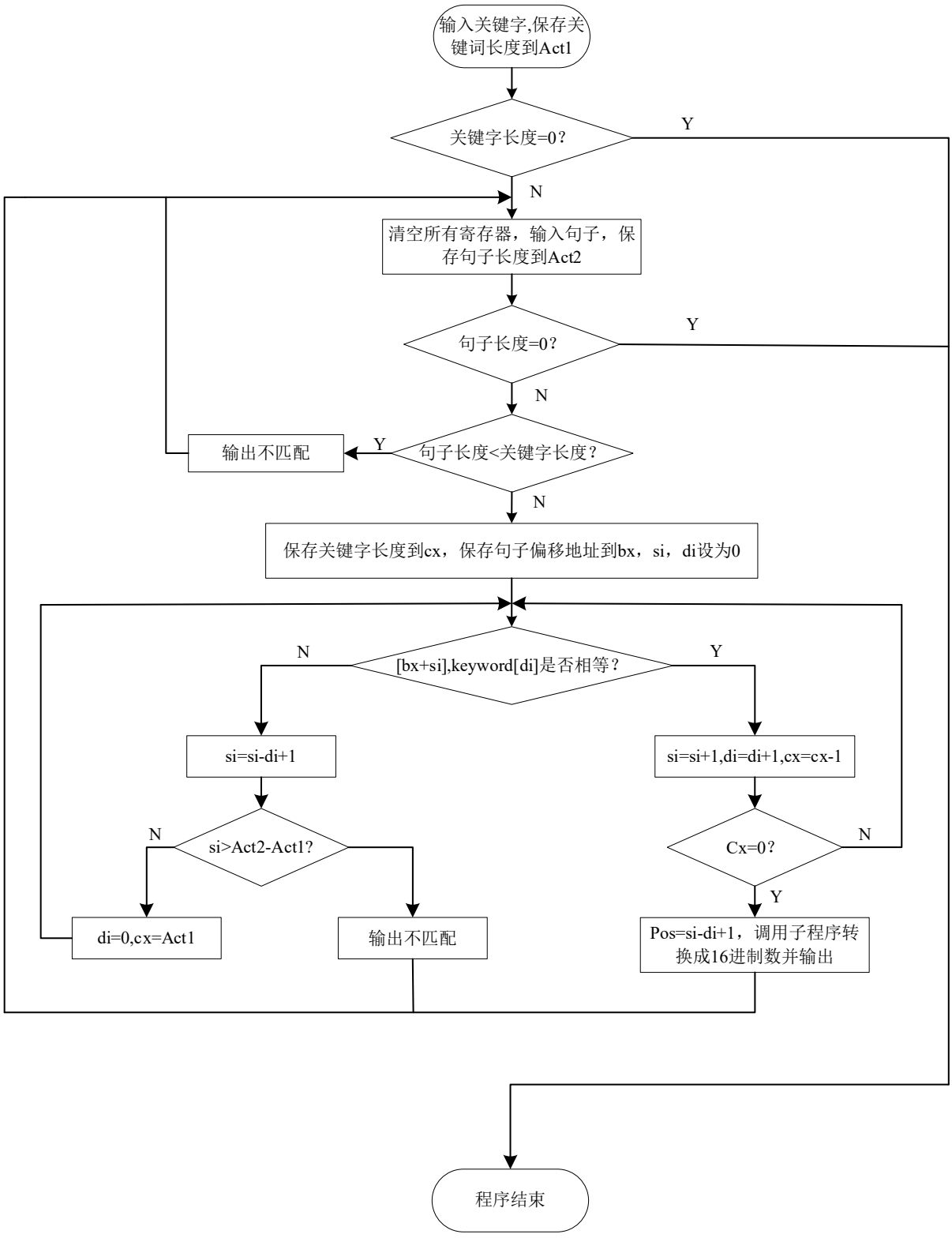


由运行结果可知，当输入的关键字和句子有效时(不为空)，则会持续循环输入匹配，当其中一个为空时则退出程序。

3. 实验体会

这次实验的难度较大，有多重循环和多个判断控制，需要设计很多的异常检测控制使得程序能够正常的运行和退出，关于字符匹配的设计部分，因为需要输出关键字在句子中的位置，一开始我想的是在每个外循环进内循环的时候将句子当前匹配的位置入栈，后面我发现，si 与 di 是同时增的，只有 si 会回退，而回退的只有 si 一个变量 di 要么增加到关键字长度要么在中间匹配失败直接归 0 重新匹配，si 回退的大小恰好是 di 的增量，所以只需要讲 si－di 就能够同时实现回退和记录位置的功能，省去了多次的入栈出栈操作。

4. 程序流程图


输入关键字,保存关键词长度到Act1
关键字长度=0?
Y
N
清空所有寄存器，输入句子，保存句子长度到Act2
句子长度=0?
Y
N
句子长度<关键字长度?
Y
输出不匹配
N
保存关键字长度到cx，保存句子偏移地址到bx，si，di设为0
[bx+si],keyword[di]是否相等？
N
Y
si=si-di+1
si=si+1,di=di+1,cx=cx-1
si>Act2-Act1?
N
Cx=0?
N
Y
di=0,cx=Act1
输出不匹配
Pos=si-di+1，调用子程序转换成16进制数并输出
程序结束

5. 程序代码

```
1.  data segment
2.      mess1       db 'Enter keyword:','$'
3.      mess2       db 'Enter sentence:','$'
4.      mess3       db 'Match at location:','$'
5.      mess4       db 'No MATCH.','$'
6.      mess5       db 'H of the sentence.','$'
7.      Max1        db 100
8.      Act1        db ?
9.      keyword     db 100 dup(?)
10.     Max2        db 100
11.     Act2        db ?
12.     Sentence    db 100 dup(?)
13.     ChangeRow db 0DH,0ah,'$'
14.     diff        dw 0
15.     location    db 0
16. data ends
17.
18. stack segment
19.             dw 128 dup(?)
20. stack ends
21.
22. code segment
23.                 assume cs:code,ds:data,ss:stack
24.     start:
25.                 mov     ax,data
26.                 mov     ds,ax
27.
28.                 lea     dx,mess1
29.                 mov     ah,09h
30.                 int     21h
31.
32.                 lea     dx,max1
33.                 mov     ah,0ah
34.                 int     21h
35.
36.     ;keyword 为 0，则退出
37.                 cmp     Act1,0
38.                 je      exit
39.
40.     input:
41.                 sub     ax,ax
42.                 sub     bx,bx
43.                 sub     cx,cx
```

```
44.                     sub     dx,dx
45.                     call    change
46.                     lea     dx,mess2
47.                     mov     ah,09h
48.                     int     21h
49.
50.                     lea     dx,max2
51.                     mov     ah,0ah
52.                     int     21h
53.
54.                     call    change
55.                     mov     al,Act1                 ;length of keyword
56.                     CBW
57.                     mov     cx,ax
58.                     mov     al,Act2                 ;length of sentence
59.                     cmp     al,0
60.                     je      no_match
61.                     sub     al,Act1
62.
63.                     CBW
64.                     mov     diff,ax
65.                     js      no_match
66.
67.                     lea     bx,Sentence
68.                     mov     di,0
69.                     mov     si,0
70.             loop_inside:
71.                     mov     ah,[bx+si]
72.                     cmp     ah,keyword[di]
73.                     jne     next
74.                     inc     di
75.                     inc     si
76.                     loop    loop_inside
77.             succ:
78.
79.                     sub     si,di
80.                     inc     si
81.                     lea     dx,mess3
82.                     mov     ah,09h
83.                     int     21h
84.
85.                     mov     bx,si
86.
87.                     push    ax
```

```
88.                     push    bx
89.                     push    cx
90.                     push    dx
91.                     call    btoh
92.                     pop     dx
93.                     pop     cx
94.                     pop     bx
95.                     pop     ax
96.
97.                     lea     dx,mess5
98.                     mov     ah,09h
99.                     int     21h
100.                    jmp     input
101.
102.        next:
103.                    push    ax
104.                    sub     si,di
105.                    inc     si
106.                    cmp     diff,si
107.                    jl      no_match
108.                    mov     di,0
109.                    mov     al,Act1
110.                    CBW
111.                    mov     cx,ax
112.                    pop     ax
113.                    jmp     loop_inside
114.
115.    btoh PROC NEAR
116.                    MOV     CH,4
117.        rotate:     MOV     CL,4
118.                    ROL     BX,CL
119.                    MOV     AL,BL
120.                    and     AL,0fh
121.                    add     AL,30h
122.                    cmp     al,3ah
123.                    jl      printit
124.                    add     al,7h
125.        printit:
126.                    MOV     dl,al
127.                    MOV     ah,2
128.                    int     21h
129.                    dec     ch
130.                    jnz     rotate
131.                    ret
```

```
btoh endp

    exit:
                mov     ah,4ch
                int     21h

    no_match:
                lea     dx,mess4
                mov     ah,09h
                int     21h
                jmp     input
change PROC far
                mov     dx,OFFSET ChangeRow
                mov     ah,09h
                int     21H
                RET
change ENDP
code ends
end start

```