**汇编语言程序设计课程实验报告**

**实验5**

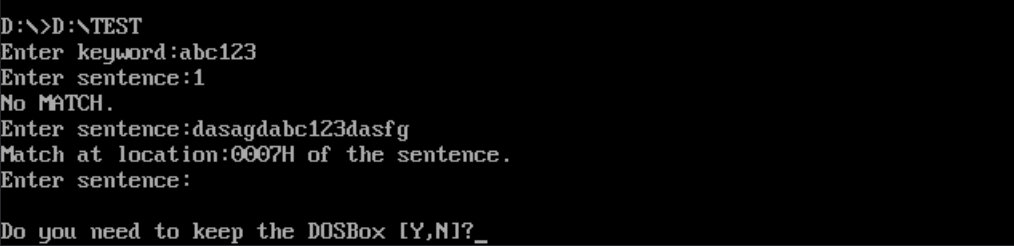
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 |  | 院系 | 计算机工程与科学学院 | 学号 |  |
| 实验目的： | | | | | |
| 查找匹配字符串SEARCH。 | | | | | |
| 实验要求： | | | | | |
| 程序接收用户键入的一个关键字以及一个句子。如果句子中不包含关键字则显示“No match!”；如果句子中包含关键字则显示“Match!”，且把该句子中的位置用十六进制数显示出来。 | | | | | |

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。

1. 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。

1. 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。

1. 程序流程图



1. 程序代码

|  |  |
| --- | --- |
|  | data segment |
|  | mess1     db 'Enter keyword:','$' |
|  | mess2     db 'Enter sentence:','$' |
|  | mess3     db 'Match at location:','$' |
|  | mess4     db 'No MATCH.','$' |
|  | mess5     db 'H of the sentence.','$' |
|  | Max1      db 100 |
|  | Act1      db ? |
|  | keyword   db 100 dup(?) |
|  | Max2      db 100 |
|  | Act2      db ? |
|  | Sentence  db 100 dup(?) |
|  | ChangeRow db 0DH,0ah,'$' |
|  | diff      dw 0 |
|  | location  db 0 |
|  | data ends |
|  |  |
|  | stack segment |
|  | dw 128 dup(?) |
|  | stack ends |
|  |  |
|  | code segment |
|  | assume cs:code,ds:data,ss:stack |
|  | start: |
|  | mov    ax,data |
|  | mov    ds,ax |
|  |  |
|  | lea    dx,mess1 |
|  | mov    ah,09h |
|  | int    21h |
|  |  |
|  | lea    dx,max1 |
|  | mov    ah,0ah |
|  | int    21h |
|  |  |
|  | ;keyword为0，则退出 |
|  | cmp    Act1,0 |
|  | je     exit |
|  |  |
|  | input: |
|  | sub    ax,ax |
|  | sub    bx,bx |
|  | sub    cx,cx |
|  | sub    dx,dx |
|  | call   change |
|  | lea    dx,mess2 |
|  | mov    ah,09h |
|  | int    21h |
|  |  |
|  | lea    dx,max2 |
|  | mov    ah,0ah |
|  | int    21h |
|  |  |
|  | call   change |
|  | mov    al,Act1                     ;length of keyword |
|  | CBW |
|  | mov    cx,ax |
|  | mov    al,Act2                     ;length of sentence |
|  | cmp    al,0 |
|  | je     no\_match |
|  | sub    al,Act1 |
|  |  |
|  | CBW |
|  | mov    diff,ax |
|  | js     no\_match |
|  |  |
|  | lea    bx,Sentence |
|  | mov    di,0 |
|  | mov    si,0 |
|  | loop\_inside: |
|  | mov    ah,[bx+si] |
|  | cmp    ah,keyword[di] |
|  | jne    next |
|  | inc    di |
|  | inc    si |
|  | loop   loop\_inside |
|  | succ: |
|  |  |
|  | sub    si,di |
|  | inc    si |
|  | lea    dx,mess3 |
|  | mov    ah,09h |
|  | int    21h |
|  |  |
|  | mov    bx,si |
|  |  |
|  | push   ax |
|  | push   bx |
|  | push   cx |
|  | push   dx |
|  | call   btoh |
|  | pop    dx |
|  | pop    cx |
|  | pop    bx |
|  | pop    ax |
|  |  |
|  | lea    dx,mess5 |
|  | mov    ah,09h |
|  | int    21h |
|  | jmp    input |
|  |  |
|  | next: |
|  | push   ax |
|  | sub    si,di |
|  | inc    si |
|  | cmp    diff,si |
|  | jl     no\_match |
|  | mov    di,0 |
|  | mov    al,Act1 |
|  | CBW |
|  | mov    cx,ax |
|  | pop    ax |
|  | jmp    loop\_inside |
|  |  |
|  | btoh 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 |
|  | 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 |
|  |  |