

上海大学



SHANGHAI UNIVERSITY

2021-2022 学年秋季学期

上海大学 计算机学院

《汇编语言程序设计》

实验 3

实验名称： 查找匹配字符串

专业： 19 级直招计科 2 班

姓名： 汪雨卿

学号： 19120191

实验名称： 查找匹配字符串

一、实验目的

查找匹配字符串 SEARCH。

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。

二、实验内容

(1) 实验原理

①查找匹配字符串

本实验要求，用户键入一个关键字和一个句子，在句子中查找匹配字符串。

键入操作在实验二已有所涉猎。因为本实验要求的是在键入的句子中查找匹配字符串，故需要将键入的关键字和句子都存在缓存区中。调用 0A 号操作可以实现该目的。

查找匹配字符串可以使用串处理指令：串比较 CMPS，串扫描 SCAS。加上前缀 repe 相等则重复，即可完成在句子中按字节重复查找关键字，并记录匹配位置的操作。

需要注意的是，如果是从句子的头开始比较，则调用 CLD 指令，设置 DF=0，地址自动增量；如果是从句子的尾开始比较，则调用 STD 指令，设置 DF=1，地址自动减量。

②输出位置的十六进制

在调用串比较指令的时候，用一个寄存器记录比较的时候句子的起始位置。当串比较指令返回结果为匹配时，寄存器记录的数据加一就是关键字匹配在句子中能够的位置，且保存在该寄存器的数据即要输出的十六进制的位置。

若是直接将该位置调用 02 号命令输出，则显示在屏幕上的是该位置的十进制。所以需要 XLAT 指令，将十六进制的位置显示在屏幕上。

XLAT 指令，首先在数据段建立一个表格，将 1~15 的十六进制的 ASCII 码保存在表格中。再将位置分成高位和低位，分别取出高位和低位的数字，再分别调

用 XLAT 指令，即可以将十六进制的高位和低位分别以其 ASCII 码输出，最后显示在屏幕上的就是十六进制的位置。

(2) 实验步骤

- (1) 启动 MASM 6.0 或 MASM for Windows 集成编程环境。
- (2) 分支指令形式编写 .ASM 源程序。
- (3) 对其进行汇编及连接，产生 .EXE 文件。
- (4) 作必要的调试。

(3) 实验记录

在数据段创建关键字 KEY 和句子 BUFFER 的名字和空间。运用 label 伪指令，创建一个名字是 KEY 的单字节变量，在 label 指令的后续指令中为 KEY 分配 20 个字节的空間。同理为 BUFFER 创建并分配 80 个字节的空間。

```
006 ; LABEL伪指令给变量设置别名，共享内存位置
007 BUFFER LABEL BYTE
008     MAX1 DB 80 ;最大长度
009     ACT1 DB ? ;实际输入长度
010     STOKN1 DB 80 DUP(?) ;空间的创建
011
012 KEY LABEL BYTE
013     MAX2 DB 80
014     ACT2 DB ?
015     STOKN2 DB 80 DUP(?)
016
017 TEMP LABEL BYTE
018     MAX3 DB 80
019     ACT3 DB ?
020     STOKN3 DB 80 DUP(?)
021
```

本操作也可以用 string 操作代替，需要注意在创建语句中指定缓冲区最大容量，即第一个变量的大小。且在每次从键盘读入关键字和句子的时候，需要将其各自的偏移地址和字符串长度分别保存在寄存器。这有可能导致寄存器不够用而数据冲突，故本实验采用上一种方法。

```
100
101 MOV DH,AH ;字符串长度减关键字长度作为循环次数
102 ADD DH,1
103
104 LEA DI,STOKN1 ;初始化DI,为字符串指针
105 LEA SI,STOKN2 ;初始化SI,为关键字指针
106 MOV BX,0 ;初始化字符串偏移量
107
108 CMP_SEG:
109     CLD ;设置SI,DI移动方向为+1
110     REPZ CMPSB ;比较字符串和关键字
111     JZ MATCH_1 ;匹配成功,跳转至MATCH_1
112     JNZ LOOP_1 ;匹配失败,循环继续匹配
113
114 LOOP_1:
115     INC BX ;字符串偏移量
116     LEA DI,STOKN1 ;重置DI
117     LEA SI,STOKN2 ;重置SI
118     MOV CL,ACT2 ;CL存放关键字长度
119     ADD DI,BX ;确定新的字符串起始位置
120     DEC DH ;循环次数减一
121     CMP DH,0 ;判断循环是否终止
122     JE NO_MATCH ;若循环终止还未匹配,则跳转至NO_MATCH
123     JNE CMP_SEG ;否则继续匹配
124
```

在数据段编写完成后，代码段需要完成分别输入关键字和句子，并将它们的偏移地址分别保存在 si 和 di，从而进行 CMP_SEG 操作。利用 LOOP_1 的循环，对关键字和句子进行匹配。

运用 repe cmpsb 操作指令，重复按字节比较字符串，直到比较相等则结束。在此之前，设置串比较操作的方向，即 CLD，令 DF=0 正向递增比较。如果串比较匹配则跳转到输出匹配位置的代码段；如果匹配失败则将句子比较的开始位置向后移动一位，并且判断是否越界，没有越界继续上述操作，否则不匹配。比较结束。

```

125 MATCH_1:
126     LEA DX,MATCH      ;匹配成功
127     MOV AH,09H
128     INT 21H
129
130     LEA DX,LOCATION_1
131     MOV AH,09H
132     INT 21H
133
134     MOV AX,BX
135     PUSH BX           ;进栈保护数据
136     INC AX            ;使起始位置从一开始
137     CALL PRINT_2      ;将关键字所在位置以十六进制数形式打印
138
139     POP BX
140
141
142     JMP START
143
144 NO_MATCH:
145     LEA DX,NOMATCH    ;匹配失败
146     MOV AH,09H
147     INT 21H
148
149
150     JMP START
151 ERROR_1:
152     LEA DX,ERROR      ;输入错误
153     MOV AH,09H
154     INT 21H
155

```

匹配失败直接输出错误情况的提示语句并跳回句子输入，继续下一次匹配操作。

匹配成功则需要将匹配位置的十六进制输出到屏幕。因为直接输出到屏幕是十六进制对应的十进制，所以需要进行 XLAT 操作将十六进制的数字显示在屏幕上。即分别将十六进制的高位和低位分别取出并通过查表，将其转换为数字对应的 ASCII 码再输出，即可将十六进制输出到屏幕上。

(4) 数据处理

①Whether to perform string matching?

Please input Y/N:Y

Enter keyword: Hello World!!! Happy Everyday!

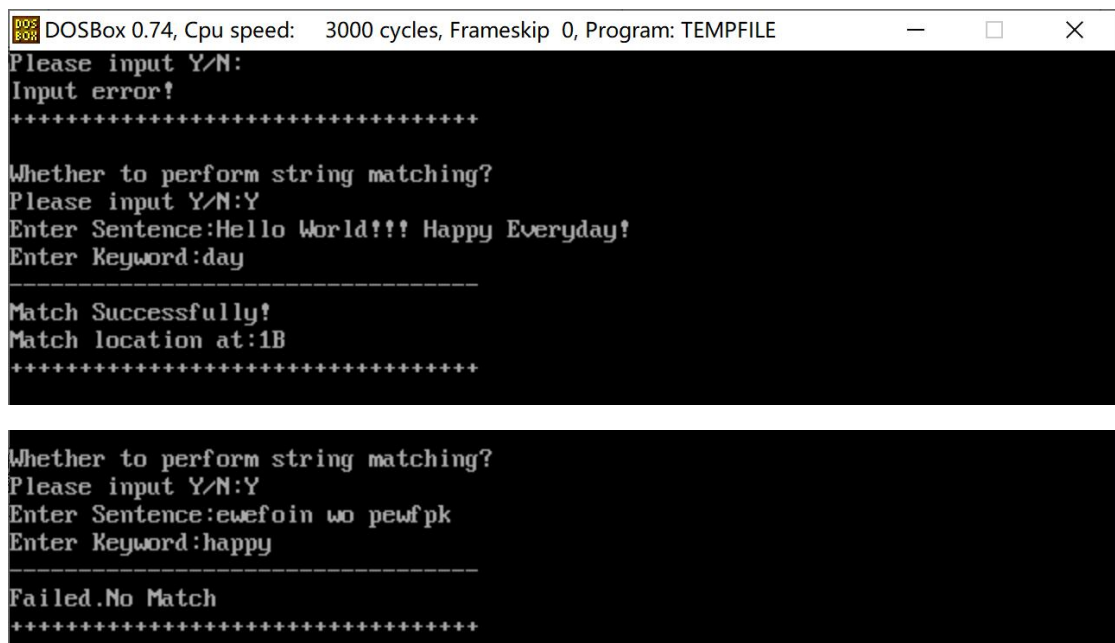
Enter Sentence: day

Match!

Match at location:1B of the sentence.

Enter Sentence: ewefoin wo pewfpk, Ok?

Failed. No match.



The image shows a DOSBox 0.74 terminal window with a black background and white text. The window title bar reads 'DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: TEMPFILE'. The terminal output is as follows:

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: TEMPFILE
Please input Y/N:
Input error!
+++++

Whether to perform string matching?
Please input Y/N:Y
Enter Sentence:Hello World!!! Happy Everyday!
Enter Keyword:day
-----
Match Successfully!
Match location at:1B
+++++

Whether to perform string matching?
Please input Y/N:Y
Enter Sentence:ewefoin wo pewfpk
Enter Keyword:happy
-----
Failed.No Match
+++++
```

三、实验体会

本次实验着重于对于字符串的操作中，通过查阅资料我了解到了如何在数据段中开辟空间，如何获取到字符串的长度、首地址等等。同时，对于字符串的一些基础操作有了一定的了解。本次实验通过实践，让我很好的熟悉了如何在实操中对字符串进行操作。

本实验让我感到有一定难度的部分在于字符串的匹配环节。如何能高效的判断关键字在字符串并且记录其位置，并不像在常用的高级语言中编程那样简洁。我首先参考了在数据结构中学习的字符串匹配算法，结合到汇编语言中的不同寄存器。可以将寄存器视为数组，而偏移量视为指针或者下标。进而实现算法的转

换。通过不断的尝试和调试最终解决了该难点。

最后，本实验输出 16 进制数据的部分，也复习了前两课学习到的输出两位数据等程序。温故而知新，这次的实验帮助我更好的理解和学习了汇编语言。

附上代码如下：

```
001 DATAS SEGMENT
002     SENTENCE DB 13,10,'Enter Sentence:','$'
003     KEYWORD DB 13,10,'Enter Keyword:','$'
004     ERROR DB 13,10,'Input error!','$'
005
006     ; LABEL伪指令给变量设置别名，共享内存位置
007     BUFFER LABEL BYTE
008         MAX1 DB 80 ;最大长度
009         ACT1 DB ? ;实际输入长度
010         STOKN1 DB 80 DUP(?) ;空间的创建
011
012     KEY LABEL BYTE
013         MAX2 DB 80
014         ACT2 DB ?
015         STOKN2 DB 80 DUP(?)
016
017     TEMP LABEL BYTE
018         MAX3 DB 80
019         ACT3 DB ?
020         STOKN3 DB 80 DUP(?)
021
022     DIVIDING_LINE_1 DB 13,10,'+',13,10,'$' ;此处分割线
023     DIVIDING_LINE_2 DB 13,10,'-',13,10,'$' ;此处分割线
024
025     ASK_1 DB 13,10,'Whether to perform string matching?','$'
026     ASK_2 DB 13,10,'Please input Y/N:','$'
027
028
029     MATCH DB 13,10,'Match Successfully!','$'
030     NOMATCH DB 13,10,'Failed.No Match','$'
031     LOCATION_1 DB 13,10,'Match location at:','$'
032     RESULT DB 13,10,'The content of the new string is:','$'
033     CRLF DB 13,10,'$'
034     CHAR DB '$'
035 DATAS ENDS
036
037 STACKS SEGMENT
038     ;此处输入堆栈段代码
039 STACKS ENDS
```

```

040
041 CODES SEGMENT
042     ASSUME CS:CODES,DS:DATAS,SS:STACKS,ES:DATAS
043 START:
044     MOV AX,DATAS
045     MOV DS,AX
046     MOV ES,AX
047
048     LEA DX,DIVIDING_LINE_1      ;回车换行
049     MOV AH,09H
050     INT 21H
051
052     LEA DX,ASK_1
053     MOV AH,09H
054     INT 21H
055
056     LEA DX,ASK_2
057     MOV AH,09H
058     INT 21H
059
060     MOV AH,01H                  ;设置带回显的键盘输入
061     INT 21H
062
063     CMP AL,'Y'
064     JE BEGIN_1
065
066     CMP AL,'N'
067     JE END_0
068     JNE ERROR_1
069
070 BEGIN_1:
071     LEA DX,SENTENCE ;提示输入
072     MOV AH,09H
073     INT 21H
074
075     MOV AH,8
076     INT 33
077
078

```

```

078
079     LEA DX,BUFFER ;输入字符串
080     MOV AH,0AH
081     INT 21H
082
083     LEA DX,KEYWORD ;提示输入
084     MOV AH,09H
085     INT 21H
086
087     LEA DX,KEY ;输入关键字
088     MOV AH,0AH
089     INT 21H
090
091     LEA DX,DIVIDING_LINE_2      ;回车换行
092     MOV AH,09H
093     INT 21H
094
095     MOV CL,ACT2 ;CL存放关键字长度
096
097     MOV AH,ACT1
098     SUB AH,CL
099     JB ERROR_1 ;字符串长度小于关键字长度，直接error
100
101     MOV DH,AH ;字符串长度减关键字长度作为循环次数
102     ADD DH,1
103
104     LEA DI,STOKN1 ;初始化DI,为字符串指针
105     LEA SI,STOKN2 ;初始化SI,为关键字指针
106     MOV BX,0 ;初始化字符串偏移量
107
108 CMP_SEG:
109     CLD ;设置SI,DI移动方向为+1
110     REPZ CMPSB ;比较字符串和关键字
111     JZ MATCH_1 ;匹配成功,跳转至MATCH_1
112     JNZ LOOP_1 ;匹配失败,循环继续匹配
113

```



```

114 LOOP_1:
115     INC BX           ;字符串偏移量
116     LEA DI,STOKN1   ;重置DI
117     LEA SI,STOKN2   ;重置SI
118     MOV CL,ACT2     ;CL存放关键字长度
119     ADD DI,BX       ;确定新的字符串起始位置
120     DEC DH          ;循环次数减一
121     CMP DH,0        ;判断循环是否终止
122     JE NO_MATCH     ;若循环终止还未匹配，则跳转至NO_MATCH
123     JNE CMP_SEG     ;否则继续匹配
124
125 MATCH_1:
126     LEA DX,MATCH     ;匹配成功
127     MOV AH,09H
128     INT 21H
129
130     LEA DX,LOCATION_1
131     MOV AH,09H
132     INT 21H
133
134     MOV AX,BX
135     PUSH BX          ;进栈保护数据
136     INC AX           ;使起始位置从一开始
137     CALL PRINT_2     ;将关键字所在位置以十六进制数形式打印
138
139     POP BX
140
141
142     JMP START
143
144 NO_MATCH:
145     LEA DX,NOMATCH   ;匹配失败
146     MOV AH,09H
147     INT 21H
148
149
150     JMP START

```

```

151 ERROR_1:
152     LEA DX,ERROR     ;输入错误
153     MOV AH,09H
154     INT 21H
155
156
157     JMP START
158
159 PRINT_2:
160     MOV AH,0         ;对于超过9个数的字符，ASCII中并没有直接与之对应的字符，因此应分别输出两位数
161     MOV BL,16
162     DIV BL           ;将AX中内容除以16，商放在AL，余数放在AH
163     MOV BH,AH
164     CMP AL,10
165     JB NUMBER_1
166     JAE LETTER_1
167     NUMBER_1:
168     ADD AL,30H       ;将个位数+30H转化为数字字符
169     JMP OUTPUT_1
170     LETTER_1:
171     ADD AL,37H       ;将个位数+37H转化为字母字符
172     OUTPUT_1:
173     MOV DL,AL
174     MOV AH,02H       ;打印个位数
175     INT 21H
176
177     MOV AL,BH
178     CMP AL,10
179     JB NUMBER_2
180     JAE LETTER_2
181     NUMBER_2:
182     ADD AL,30H       ;将个位数+30H转化为数字字符
183     JMP OUTPUT_2
184     LETTER_2:
185     ADD AL,37H

```



```
186 OUTPUT_2:
187     MOV DL,AL
188     MOV AH,02H ;打印十位数
189     INT 21H
190     RET
191
192
193 END_0:
194     MOV AH,4CH
195     INT 21H
196 CODES ENDS
197     END START
198
199
200
201
```