

# 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： 毕欣怡

学号： 19120194

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 keyword 和句子 sentence 的名字和空间。运用 label 伪指令，创建一个名字是 keyword 的单字节变量，在 label 指令的后续指令中为 keyword 分配 20 个字节的空間。同理为 sentence 创建并分配 120 个字节的空間。

```
keyword label byte ;label伪指令，创建一个keyword，不分配空间
max1 db 20 ;keyword的最大空间，命名是max1
act1 db ? ;keyword的实际距离，命名是act1
kw db 20 dup(?) ;keyword的空间创建，20个db的空白空间
sentence label byte
max2 db 120
act2 db ?
sen db 120 dup(?)
;string1 db 120,120 dup(?)该行语句与sentence相同，第一个120表示缓冲区最大容量
;string2 db 20,20 dup(?)
```

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。

```
docompare:
    sub cx,cx
    mov cl,act1 ;关键字实际长度，存在cl
    lea si,kw ;取关键字有效地址，存在si
    lea bx,sen ;取句子有效地址，存在bx
    add bl,al ;开始比较的位置
    mov di,bx ;将句子开始比较的有效地址，存在di

    ;下面两行操作前，需要将比较的两个串分别放在si,di，将比较位数放在cl
    cld ;df=0 正向比较
    repe cmpsb ;重复串比较操作直到相等，跳到match
    jz domatch ;相等zf=1,跳到match

    inc al ;ax记录串偏移量，递增
    cmp al,act2 ;判断是否越界，al表示现在所在句子的地方，act2表示句子实际长度
    jnb notmatch ;al不小于act2，越界，跳到没有匹配
    jmp docompare ;否则还没比较完成，继续比较
```

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 CMPS 操作。因为需要在句子中从左到右扫描是否存在关键字，所以设置了一个 al 寄存器用来保存现在开始扫描的头位置，并且每次不匹配且不越界的一次循环结束加一。

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。

```
add al,1 ;找到匹配位置，因为计算机默认从0开始计数，所以实际匹配位置是al+1
sub cx,cx
mov ch,al ;备份一份匹配位置，该数据是十六进制，需要将其显示在屏幕上
mov bl,al ;显示第一位十六进制
mov cl,4
shr bl,cl ;右移4位，右移位数>1，将右移量存在cl。右移后高位为0，低位即原来的十六进制高位
mov al,bl ;将右移后的数据放入al，进行查表操作，得到的结果是十六进制高位对应的数字的ASCII码
lea bx,tab ;将表格偏移地址放入bx，查表操作前操作
xlat ;查表指令，将以bx为首地址，偏移地址为al的内容送到al
mov dl,al ;此时存在al中的是十六进制高位的数字对应的ASCII码
mov ah,2
int 21h ;此时调用输出语句，输出到屏幕上的是原来应该输出到屏幕上的十六进制的高位

and ch,0fh ;匹配位置与0fh，得到的结果是应该输出的十六进制的低位，高位为0
mov al,ch
lea bx,tab
xlat
mov dl,al ;进行和高位一样的查表操作并输出，此时屏幕上显示的是应该输出的十六进制的低位
mov ah,2
int 21h
```

匹配失败直接输出“匹配失败”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

①Enter keyword:abc

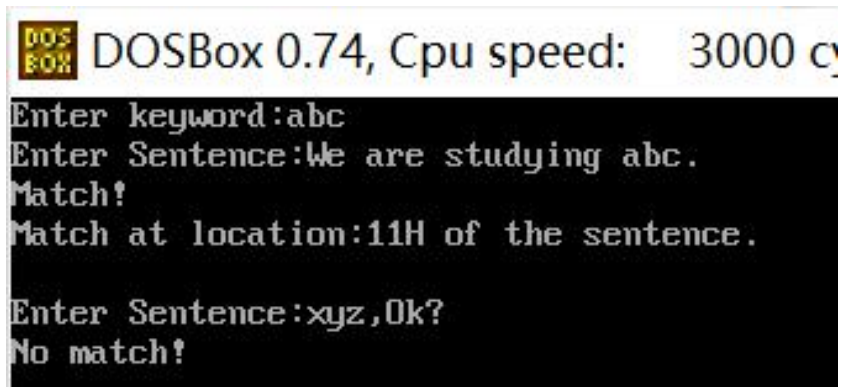
Enter Sentence: We are studying abc.

Match!

Match at location:11H of the sentence.

Enter Sentence: xyz, Ok?

No match!



```
DOS BOX DOSBox 0.74, Cpu speed: 3000 cycles, Fra...
Enter keyword:abc
Enter Sentence:We are studying abc.
Match!
Match at location:11H of the sentence.

Enter Sentence:xyz,Ok?
No match!
```

②Enter keyword:www

Enter Sentence: We are studying abc.

No match!

Enter Sentence:wwwwwwwww

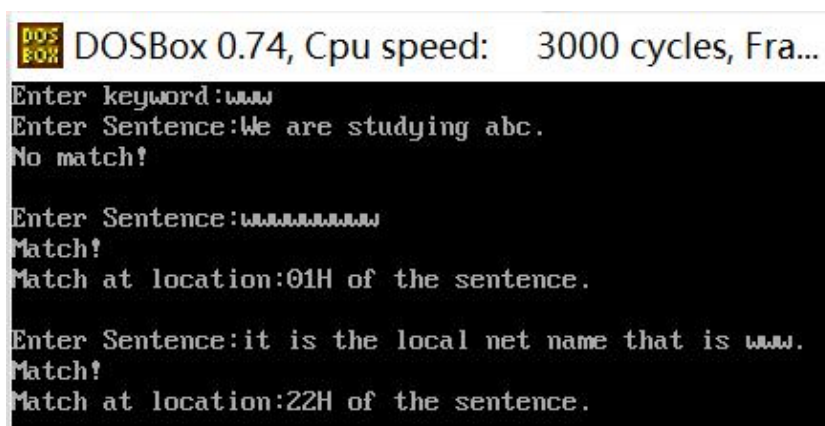
Match!

Match at location:01H of the sentence.

Enter Sentence:it is the local net name that is www.

Match!

Match at location:22H of the sentence.



```
DOS BOX DOSBox 0.74, Cpu speed: 3000 cycles, Fra...
Enter keyword:www
Enter Sentence:We are studying abc.
No match!

Enter Sentence:wwwwwwwww
Match!
Match at location:01H of the sentence.

Enter Sentence:it is the local net name that is www.
Match!
Match at location:22H of the sentence.
```

### 三、实验体会

本次实验要求的字符串匹配需要用到串处理指令。在实验前的一节课正好学到了该指令，因此在写代码的过程中，串比较部分的代码并没有多大的困难；在搞清楚串比较指令需要的两个串存放的寄存器，以及设置好比较的位数和方向之后，串比较操作很快就完成了。

真正困住我的脚步的是显示到屏幕上的十六进制数。上节课将十进制数显示到屏幕上用到了除法，但是这次的实验不能一概而论。在老师的指导下，我用了汇编语言的查表指令，将保存在寄存器中的十六进制的位置的每一位分别取出，再调用查表 XLAT 指令将其转换成对应的 ASCII 码输出，这样显示在屏幕上的就是实际串匹配位置的十六进制表示。

通过这次实验，我不仅对刚学的串处理指令有了更加深刻的了解，也再次熟悉了查表指令。纸上得来终觉浅，绝知此事要躬行。

附上代码如下：

```
001 ;查找匹配字符串
002 data segment
003     mess1 db 'Enter keyword:$'
004     mess2 db 'Enter Sentence:$'
005     mess3 db 'Match!$'
006     mess4 db 'Match at location:$'
007     mess5 db 'H of the sentence.$'
008     mess6 db 'No match!$'
009     mess7 db 13,10,'$' ;取代四行的回车换行语句
010 ;XLAT指令所需，建立表格
011     tab db 030h,031h,032h,033h,034h,035h,036h,037h,038h,039h,041h,042h,043h,044h,045h,046h
012     keyword label byte ;label伪指令，创建一个keyword，不分配空间
013         max1 db 20 ;keyword的最大空间，命名是max1
014         act1 db ? ;keyword的实际距离，命名是act1
015         kw db 20 dup(?) ;keyword的空间创建，20个db的空白空间
016     sentence label byte
017         max2 db 120
018         act2 db ?
019         sen db 120 dup(?)
020 ;string1 db 120,120 dup(?)该行语句与sentence相同，第一个120表示缓冲区最大容量
021 ;string2 db 20,20 dup(?)
022 data ends
023
024 code segment
025     assume cs:code,ds:data,es:data
026     start:
027     main proc far
028         push ds
029         sub ax,ax
030         push ax
031         mov ax,data
032         mov ds,ax
033         mov es,ax ;数据压栈保存
```



```

034
035     lea dx,mess1
036     mov ah,9
037     int 21h;输出提示语句：请输入关键字
038     lea dx,keyword
039     mov ah,0ah
040     int 21h;从键盘读入关键字存在缓冲区，回车结束
041     ;sub ch,ch
042     ;mov cl,[string2+1];关键字长度
043     ;lea si,string2+2;关键字起始字符
044     lea dx,mess7
045     mov ah,9
046     int 21h;回车换行
047
048 resentence:
049     lea dx,mess2
050     mov ah,9
051     int 21h;输出提示语句：请输入句子。句子可以重复输入，所以该部分可以循环
052     lea dx,sentence
053     mov ah,0ah
054     int 21h;从键盘读入句子存在缓冲区，回车结束
055     ;sub ch,ch
056     ;mov ch,[string1+1];句子长度
057     ;lea di,string1+2;句子起始字符
058     lea dx,mess7
059     mov ah,9
060     int 21h;回车换行
061
062     mov ax,0;设置句子开始比较的位置，从头开始
063 docompare:
064     sub cx,cx
065     mov cl,act1 ;关键字实际长度，存在cl
066     lea si,kw ;取关键字有效地址，存在si

067     lea bx,sen ;取句子有效地址，存在bx
068     add bl,al ;开始比较的位置
069     mov di,bx ;将句子开始比较的有效地址，存在di
070
071     ;下面两行操作前，需要将比较的两个串分别放在si,di，将比较位数放在cl
072     cld ;df=0 正向比较
073     repe cmpsb ;重复串比较操作直到相等，跳到match
074     jz domatch ;相等zf=1,跳到match
075
076     inc al ;ax记录串偏移量，递增
077     cmp al,act2 ;判断是否越界，al表示现在所在句子的地方，act2表示句子实际长度
078     jnb notmatch ;al不小于act2，越界，跳到没有匹配
079     jmp docompare;否则还没比较完成，继续比较
080
081 domatch::字符串匹配，输出十六进制的位置
082     lea dx,mess3
083     mov ah,9
084     int 21h;输出提示语句：匹配！
085     lea dx,mess7
086     mov ah,9
087     int 21h;回车换行
088
089     lea dx,mess4
090     mov ah,9
091     int 21h;输出提示语句
092
093     add al,1 ;找到匹配位置，因为计算机默认从0开始计数，所以实际匹配位置是al+1
094     sub cx,cx
095     mov ch,al ;备份一份匹配位置，该数据是十六进制，需要将其显示在屏幕上
096     mov bl,al ;显示第一位十六进制
097     mov cl,4
098     shr bl,cl ;右移4位，右移位数>1，将右移量存在cl。右移后高位为0，低位即原来的十六进制高位
099     mov al,bl ;将右移后的数据放入al，进行查表操作，得到的结果是十六进制高位对应的数字的ASCII码

```

```

100 lea bx,tab;将表格偏移地址放入bx,查表操作前操作
101 xlat ;查表指令,将以bx为首地址,偏移地址为al的内容送到al
102 mov dl,al ;此时存在al中的是十六进制高位的数字对应的ASCII码
103 mov ah,2
104 int 21h ;此时调用输出语句,输出到屏幕上的是原来应该输出到屏幕上的十六进制的高位
105
106 and ch,0Fh;匹配位置与0Fh,得到的结果是应该输出的十六进制的低位,高位为0
107 mov al,ch
108 lea bx,tab
109 xlat
110 mov dl,al ;进行和高位一样的查表操作并输出,此时屏幕上显示的是应该输出的十六进制的低位
111 mov ah,2
112 int 21h
113
114 lea dx,mess5
115 mov ah,9
116 int 21h;输出提示语句
117 lea dx,mess7
118 mov ah,9
119 int 21h;回车换行
120 lea dx,mess7
121 mov ah,9
122 int 21h;回车换行
123 jmp resentence;程序要求句子可重复输入,无条件跳转到输入句子部分
124
125 notmatch:;字符串不匹配
126 lea dx,mess6
127 mov ah,9
128 int 21h;输出提示语句:字符串不匹配
129 lea dx,mess7
130 mov ah,09
131 int 21h;回车换行
132 lea dx,mess7
133
133 mov ah,09
134 int 21h;回车换行
135 jmp resentence;重复输入句子
136
137 exit:ret
138
139 main endp
140 code ends
141 end start

```