汇编语言程序设计 期末大作业

吕鑫 201411212012

实验内容

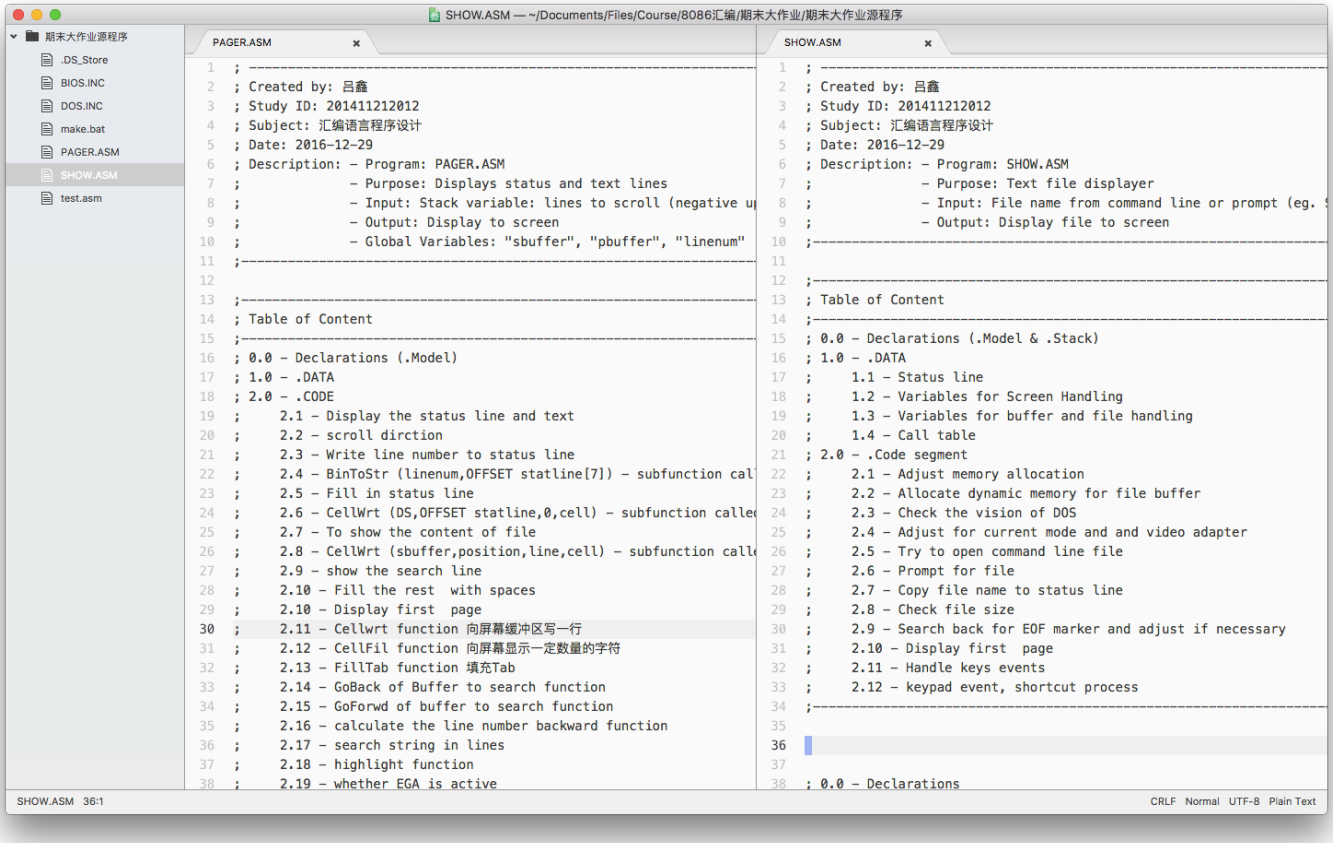
- 阅读并深入理解 Show.exe 软件汇编语言源程序(包括 show.asm 及 pager.asm)
- 修改 show.asm 及 pager.asm，为 SHOW 软件增加文本搜索功能，具体要求如下:
    1. 不影响原程序功能
    2. 给定待搜索字符串，实现在打开的文本文件中搜索出全部匹配字符串
    3. 高亮显示匹配的字符串
    4. 统计并显示匹配的数量
    5. 设计并实现友好的人机界面，可以方便输入及修改待搜索字符串

- 作业要求
    1. 对主要算法及程序结构进行详细分析及设计
    2. 画出主要算法流程图
    3. 用汇编语言实现搜索算法及人机交互界面
    4. 列写相应源程序，并加以注释
    5. 将新代码融入原程序，生成出新的可执行文件
    6. 给出执行结果，并对结果进行必要分析
    7. 给出完整报告

源码分析

我在阅读源程序的过程中，一边阅读代码和注释，一边整理书写和排版，同时在原先已有的部分注释的基础上，补充了没有的部分的注释（中英混杂……），并且给文件重新划分了结构，并写了分层目录, 代码分析见注释。

文件结构如图:



show.asm

```
; ---------------------------------------------------------------------------
; Created by: 吕鑫
; Study ID: 201411212012
; Subject: 汇编语言程序设计
; Date: 2016-12-29
; Description: - Program: SHOW.ASM
;                        - Purpose: Text file displayer
;                        - Input: File name from command line or prompt (eg. SHOW.
EXE TEST.ASM)
;                        - Output: Display file to screen
;---------------------------------------------------------------------------
-------

;---------------------------------------------------------------------------
; Table of Content
;---------------------------------------------------------------------------
; 0.0 - Declarations (.Model & .Stack)
; 1.0 - .DATA
;         1.1 - Status line
;         1.2 - Variables for Screen Handling
;         1.3 - Variables for   buffer and file handling
;         1.4 - Call table
```

```
; 2.0 - .Code segment
;           2.1 - Adjust memory allocation
;           2.2 - Allocate dynamic memory for file buffer
;           2.3 - Check the vision of DOS
;           2.4 - Adjust for current mode and and video adapter
;           2.5 - Try to open command line file
;           2.6 - Prompt for file
;           2.7 - Copy file name to status line
;           2.8 - Check file size
;           2.9 - Search back for EOF marker and adjust if necessary
;           2.10 - Display first    page
;           2.11 - Handle keys events
;           2.12 - keypad event, shortcut process
;-----------------------------------------------------------------------------



; 0.0 - Declarations

PAGE      60,132
TITLE     SHOW

DOSSEG
    .MODEL  small

    INCLUDE dos.inc
    INCLUDE bios.inc

    .STACK  100h

    .DATA



    ; 1.1 - Status line
    PUBLIC  statline, linenum, statSearch, searchPos
    statSearch   DB   "/",100 dup(" ")
    statline         DB   " Line:         " ; 在状态栏显示行号
    statfile         DB   " File:                " ; 在状态栏显示正在读取的文件名字
    stathelp         DB   "Search:ENTER Quit:ESC Move:  PGUP PGDN HOME END "
    linenum          DW   1

    ; 1.2 - Variables for   screen handling
    PUBLIC  cell, rows, columns, vidadr, statatr, scrnatr, cga ; 依次声明屏幕显示单元
、行、列、缓冲地址、配色1、配色2、CGA标识（对应vidadr）
    cell      LABEL   WORD      ; Cell (character and attribute)
    char      DB      " "       ; Initialize to space
    attr      DB      ?     ; Attribute
```

```
    columns   EQU     80        ; Number of columns
    rows      DW      24        ; Number of rows - status line takes one more
    mode      DB      ?     ; Initial mode
    pag   DB      ?     ; Initial display page
    newvid    DB      0     ; Video change flag
    cga   DB      1     ; CGA flag - default yes
    vidadr    DW      0B800h    ; Video buffer address - default CGA
    mono      EQU     0B000h    ; Monochrome address
    statatr   DB      030h      ; Color default - black on cyan
    bwstat    EQU     070h      ; B&W default - black on white
    scrnatr   DB      017h      ; Color default - white on blue
    bwscrn    EQU     007h      ; B&W default - white on black



    ; 1.3 - Variables for   buffer and file handling
    PUBLIC  buffer,pbuffer,sbuffer,fsize,namebuf ; 依次声明缓冲、缓冲偏移、缓冲段基址、文
件大小
    buffer    LABEL   DWORD
    pbuffer   DW      0     ; Position in buffer (offset)
    sbuffer   DW      ?     ; Base of buffer (segment)
    lbuffer   DW      ?     ; Length of buffer
    fhandle   DW      ?     ; Holds file handle on open
    fsize           DW    ?     ; File size after dosopen
    searchPos DW        0
    prompt    DB      13,10,13,10,"Enter filename: $"
    prompt2   DB      13,10,"File problem. Try again? $"
    namebuf   DB      66,?
    filename  DB      66 DUP (0)        ; Buffer for file name
    err1      DB      13,10,"Must have DOS 2.0 or higher",13,10,"$"
    err2      DB      13,10,"File too big",13,10,"$"

    ; 1.4 - Call table
    ; 这里定义了函数调用
    exkeys    DB      71,72,73,79,80,81 ; Extended key codes
    lexkeys   EQU     $-exkeys      ; Table of keys
    extable   DW      homek
    DW    upk
    DW    pgupk
    DW    endk
    DW    downk
    DW    pgdnk
    DW    nonek

; 2.0 - .Code

  .CODE
    EXTRN     pager: PROC, isEGA:PROC   ; Routines in other module
```

```
    start:
                    mov   ax,@DATA      ; Initialize data segment as DATA
                  mov     ds,ax
                  cli                               ; Turn off interrupts
                  mov     ss,ax             ; Initialize stack segment as DATA
                  mov     sp,OFFSET STACK   ; Initialize stack pointer
                  sti                               ; 打开中断


    ;2.1 - Adjust memory allocation
                  mov     bx,sp         ; Convert stack pointer to paragraphs
                  mov     cl,4          ; to get stack size
                  shr     bx,cl
                  add     ax,bx         ; Add SS to get end of program
                  mov     bx,es         ; Get start of program
                  sub     ax,bx         ; Subtract start from end
                  @ModBlok ax           ; Release memory after program

    ;2.2 - Allocate dynamic memory for file buffer
                  @GetBlok 0FFFh            ; Try to allocate 64K
                  mov     sbuffer,ax        ; Save buffer segment
                  mov     lbuffer,bx        ; Save actual length allocated

    ;2.3 - Check the vision of DOS
                  @GetVer           ; Get DOS version
                  cmp     al,2          ; Requires DOS 2.0
                  jge     video
                  @DispStr err1         ;   else error and quit
                  int     20h

    ;2.4 - Adjust for current mode and and video adapter, so as to check if the EG
A is active, and return the number of row, if not, return 0
    ; 判断EGA or VGA
    video:
                    call isEGA                      ; EGA (or VGA)?
                  or      ax,ax                     ; If 0 must be CGA or MA
                  je      modechk                   ; Leave default, 0 now, jump t
o modechk
                  mov     rows,ax                   ; or Load rows
                  dec     cga                           ; Not CGA
    ;识别显示模式
    modechk:
                    @GetMode            ; Get video mode
                  mov     mode,al       ; Save initial mode and page
                  mov     pag,bh        ; store the page
                  mov     dl,al         ; Work on copy
                  cmp     dl,7          ; Is it mono 7?
                  je      loadmono  ; Yes? Set mono
```

```
                cmp     dl,15           ; Is it mono 15?
                jne     graphchk  ; No? Check graphics

    loadmono:
                  mov   vidadr,mono       ; Load mono address
                mov     statatr,bwstat    ; Set B&W defaults for status line
                mov     scrnatr,bwscrn    ;   and screen background
                dec     cga             ; Not CGA
                cmp     al,15           ; Is it mono 15?
                jne     cmdchk          ; No? Done
                mov     dl,7            ; Yes? Set standard mono
                jmp     SHORT chmode

    graphchk:
                  cmp   dl,7            ; 7 or higher?
                jg      color           ; 8 to 14 are color (7 and 15 done)
                cmp     dl,4            ; 4 or higher?
                jg      bnw                 ; 5 and 6 are probably black and white
                je      color           ; 4 is color
                test    dl,1      ; Even?
                jz      bnw                 ; 0 and 2 are black and white

        color:                              ; 1 and 3 are color
                cmp     dl,3            ; 3?
                je      cmdchk          ; Yes? Done
                mov     dl,3            ; Change mode to 3
                jmp     SHORT chmode
        ;配色方案
            bnw:
                  mov   statatr, bwstat   ; Set B&W defaults for status line
                mov     scrnatr, bwscrn   ; and   screen background
                cmp     dl,2            ; 2?
                je      cmdchk          ; Yes? Done
                mov     dl,2            ; Make it 2
        ;设置显示模式
        chmode:
                  @SetMode dl               ; Set video mode
                @SetPage 0                  ; Set video page
                mov     newvid,1        ; Set flag

        ;2.5 - Try to open command line file
        cmdchk:
                  mov   bl,es:[80h]         ; Get length
                sub     bh,bh
                mov     WORD PTR es:[bx+81h],0; Convert to ASCIIZ
                push    ds
                @OpenFil 82h,0,es       ; Open argument
                pop     ds
```

```
                jc      getname         ; If error, get from prompt
                mov     fhandle,ax          ;   else save handle
                push    ds
                @GetFirst 82h,,es       ; Let DOS convert to file name
                pop     ds
                jnc     opened          ; If OK file is open


        ;2.6 - Prompt for file
        getname:                                        ;get the filename
                  @DispStr prompt               ; Prompt for file
                @GetStr namebuf,0           ; Get response as ASCIIZ
                @OpenFil filename,0         ; Try to open response
                jc      badfile                     ; If successful, continue
                mov     fhandle,ax              ; Save handle
                @GetFirst filename          ; Let DOS convert to file name
                jnc     opened                      ; If OK, file is opened

        badfile:                                    ; 打开失败了
                  @DispStr prompt2              ; else prompt to try    again
                @GetKey 0,1,0
                and     al,11011111b        ; Convert key to uppercase
                cmp     al,"Y"          ;       If yes,
                je      getname         ;   try again
                jmp     quit            ;   else quit

        ;2.7 - Copy file name to status line
        opened:
                  mov   si, 9Eh         ; Load FCB as as source
                mov     di, OFFSET statfile[5]    ; Load status line as destinat
ion
                mov     al, es:[si]         ; Load first byte
                inc     si
        copy:
                  mov   [di], al        ; Save and load bytes until 0
                inc     di
                mov     al, es:[si]
                inc     si
                or      al, al              ; Check for 0, 这里只要不是0, 就会继续复制
                loopne  copy

        ;2.8 - Check    file size
                @GetFilSz fhandle       ; Get file size
                or      dx,dx                   ; Larger than 64K?
                jne     big                         ; Yes? Too big
                mov     fsize,ax            ; Save file size
                mov     cx,4                    ; Convert to paragraphs
                shr     ax,cl
```

```
                cmp     ax,lbuffer        ; Is it larger than buffer
                jle     fileread          ; No? Continue

      big:
              @DispStr err2           ;  错误，因为文件太大了
            @Exit   2

  fileread:                                       ; 文件读取
              push      ds
            @Read   buffer,fsize,fhandle
            pop     ds
            jnc     readok        ; If no read error continue
            jmp     getname       ;   else try again


      ;2.9 - Search back for EOF marker and adjust    if necessary
readok:
              mov   di,ax         ; Load file length
            push  es          ; Save ES and load buffer segment
            mov     es,sbuffer
            std               ; Look backward for 255 characters
            mov     cx,0FFh
            mov     al,1Ah        ; Search for EOF marker
            repne   scasb
            cld
            jcxz    noeof         ; If none, we're OK
            inc     di            ;   else adjust and save file size
            mov     fsize,di

      noeof:
          pop   es


      ;2.10 - Display first   page
          xor   ax,ax         ; Start at 0
          push      ax
      firstpg:
          call      pager


      ;2.11 - Handle keys events
      nextkey:
              @GetKey 0,0,0           ; Get a key
      nextkey2:
              cmp   al,0          ; Is it a null?
            je      extended      ; Yes? Must be extended code
            cmp     al,27         ; Is it ESCAPE?
              je quit
```

```
        cmp al, 13          ; 搜索
        je  searchDriver
        call Enterk
      jmp     nextkey       ; No? Ignore unknown command

searchDriver:
        call searchk
        jmp nextkey


quit:
        cmp rows, 23
        je searchDriver
        @ClosFil fhandle        ; Yes? Close file 这里是ESC
      @FreeBlok sbuffer     ; Release buffer
      cmp     newvid,1      ; Restore video?
      jne     thatsall      ; No?
      @SetMode mode         ; Restore video mode, page, and cursor
      @SetPage pag
thatsall:                       ;行数滚动更新
        mov   dx,rows       ; Load last row and first column
      xchg  dl,dh
      mov     cx,dx         ; Make row the same
      mov     dl,79
      @Scroll 0             ; Clear last line
      sub     dl,dl
      @SetCurPos            ; Set cursor

    @Exit     0         ; Quit

extended:
        @GetKey 0,0,0           ; Get extended code
      push    es
      push    ds            ; Load DS into ES
      pop     es
      mov     di,OFFSET exkeys  ; Load address and length of key list
      mov     cx,lexkeys+1
      repne   scasb         ; Find position
      pop     es
      sub     di,(OFFSET exkeys)+1  ; Point to key
      shl     di,1          ; Adjust pointer for word addresses
      call    extable[di]       ; Call procedure
      jmp     nextkey


homek:
    mov   pbuffer,0     ; HOME - set position to 0
      push  pbuffer
```

```
            mov     linenum,1
            call  pager
            retn

    upk:
              mov   ax,-1         ; UP - scroll back 1 line
            push  ax
            call  pager
            retn

        pgupk:
              mov   ax,rows       ; PGUP - Page back
            neg     ax
            push  ax
            call  pager
            retn

        endk:
              mov   ax,fsize          ; END - Get last byte of file
            mov     pbuffer,ax        ; Make it the file position
            mov     linenum,-1        ; Set illegal line number as flag
            mov     ax,rows               ; Page back
            neg     ax
            push  ax
            call  pager
            retn

        downk:
              mov   ax,1          ; down - scroll forward 1 line
            push  ax
            call  pager
            retn

        pgdnk:
              push    rows            ; pgdn - page forward
            call  pager
            retn

        nonek:
              retn                ; ignore unknown key


        ;2.12 - keypad event, shortcut process
        searchk:
              cmp rows,24
              jne result1
              mov ax, 23
              jmp result2
```

```
    result1:
            mov ax ,24
    result2:
        mov rows,ax
            mov ax, searchPos     ; 初始化buffer
    result3:
            mov si,9
            cmp ax,0
            jne  result4
            mov searchPos, ax
            xor ax, ax
            push ax
            call pager
            retn
    result4:
            dec ax
            add si,ax
            mov statSearch[si],' '
            cmp ax,0
            jne result3

    ; enter to search
    enterk:                             ;搜索栏激活判断
            cmp rows,23
            jne cEnter4

    cEnter1:
            mov si,9
            add si,searchPos
            cmp al,8                    ;退格键?
            je cEnter2
            mov statSearch[si], al      ;    输出字串并更新位置
            inc searchPos
            jmp cEnter3
    cEnter2:
            cmp searchPos,0         ;退到头
            je cEnter4
            dec si
            mov statSearch[si],' '
            dec searchPos
            jmp cEnter3
cEnter3:
            xor ax, ax
            push ax
            call pager
    cEnter4:
            retn
```

```
                end    start
```

pager.asm

```
    ; ------------------------------------------------------------------------
--
; Created by: 吕鑫
; Study ID: 201411212012
; Subject: 汇编语言程序设计
; Date: 2016-12-29
; Description: - Program: PAGER.ASM
;                         - Purpose: Displays status and text lines
;                         - Input: Stack variable: lines to scroll (negative up, po
sitive down)
;                         - Output: Display to screen
;                             - Global Variables: "sbuffer", "pbuffer", "linenum"
;-------------------------------------------------------------------------------
-------

;-----------------------------------------------------------------------
; Table of Content
;-----------------------------------------------------------------------
; 0.0 - Declarations (.Model)
; 1.0 - .DATA
; 2.0 - .CODE
;       2.1 - Display the status line and text
;           2.2 - scroll dirction
;           2.3 - Write line number to status line
;           2.4 - BinToStr (linenum,OFFSET statline[7]) - subfunction called
;           2.5 - Fill in status line
;           2.6 - CellWrt (DS,OFFSET statline,0,cell) - subfunction called
;           2.7 - To show the content of file
;           2.8 - CellWrt (sbuffer,position,line,cell) - subfunction called
;           2.9 - show the search line
;           2.10 - Fill the rest    with spaces
;           2.11 - Cellwrt function 向屏幕缓冲区写一行
;           2.12 - CellFil function 向屏幕显示一定数量的字符
;           2.13 - FillTab function 填充Tab
;           2.14 - GoBack of Buffer to search function
;           2.15 - GoForwd of buffer to search function
;           2.16 - calculate the line number backward function
;           2.17 - search string in lines
;           2.18 - highlight function
;           2.19 - whether EGA is active
;           2.20 - Converts integer to string
;           2.21 - wirte in CGA model
;           2.22 - Calculate the tab number
;-----------------------------------------------------------------------
```

```
; 0.0 - Declarations (.Model)
PAGE      60,132
.MODEL  small


; 1.0 - .DATA
.DATA ;*************************************
        EXTRN     statatr:BYTE,scrnatr:BYTE,sbuffer:WORD,pbuffer:WORD
        EXTRN     fsize:WORD,cell:WORD,statline:BYTE,linenum:WORD,statSearch:BYTE,
searchPos:WORD
        EXTRN     rows:WORD,vidadr:WORD,cga:BYTE


; 2.0 - .CODE
.CODE
      PUBLIC  Pager,isEGA


; 2.1 - Display the status line and text
Pager     PROC
      push  bp
      mov     bp,sp

      mov     es,sbuffer        ; Initialize buffer position, es为段基址
      mov     di,pbuffer        ; offset


; 2.2 - scroll dirction
      mov     cx,[bp+4]     ; Get count argument
      mov     ax,10         ; Search for linefeed, 这个是ASCII10

      or      cx,cx         ; Argument 0? 滚动方向判断，大于0则forward，小于0则backward
      jg      forward       ; If above, forward
      jl      backward      ; If below, backward
      jmp     SHORT show        ; If equal, done 这是没滚动
backward:                                                                ;这种是
向上滚
        call    GoBack      ; Adjust backward
      jmp     SHORT show        ; Show screen
forward:                                                                 ;这
种为向下滚动
        call    GoForwd     ; Adjust forward


; 2.3 - Write   line number to status line
show:
```

```
        cld                         ; Go forward
      push  di
      push  es
      push  ds          ; Load DS to ES
      pop       es

; 2.4 - BinToStr (linenum,OFFSET statline[7]) - subfunction called
      push linenum                          ; Arg 1
      mov    ax,OFFSET statline[6]
      push ax                                   ; Arg 2
      call BinToStr                    ; Convert 行号 to string

; 2.5 - Fill in status line 打印显示
      mov    cx,7           ; Seven spaces to fill
      sub    cx,ax          ; Subtract those already done
      mov    al," "     ; Fill with space
      rep    stosb
      pop    es
      mov    bl,statatr     ; Load status attribute
      mov    BYTE PTR cell[1],bl

; 2.6 - CellWrt (DS,OFFSET statline,0,cell) - subfunction called
      push  ds                              ; Arg 1
      mov       ax,OFFSET statline  ; Arg 2
      push  ax
      sub       ax,ax                               ; Arg 3
      push  ax
      push  cell                            ; Arg 4
      call  CellWrt     ; Write status line 确定了status line 的颜色
      pop     di
      mov     bl,scrnatr        ; Load screen attribute 刚刚的配色
      mov     BYTE PTR cell[1],bl
      mov     si,di         ; Update position, buffer offset
      mov     cx,rows       ; Lines per screen 行数/页

; 2.7 - To show the content of file
show1:
        mov   bx,rows       ; Lines of text
      inc     bx                ; Adjust for 0
      sub     bx,cx         ; Calculate current row
      push  cx              ; Save line number
        push    si              ; Save buffer offset

; 2.8 - CellWrt (sbuffer,position,line,cell) - subfunction called, 写入第0行
      push  sbuffer     ; Arg 1
      push  si          ; Arg 2
      push  bx          ; Arg 3
      push  cell            ; Arg 4
```

```
      call  cellwrt     ; Write line, 缓冲区内容写入
      push  ss          ; Restore DS from SS
      pop     ds
        pop     si
      pop     cx            ; Restore line number
        cmp     rows, 23
        jne         NoneResult
        mov bx, rows
        inc bx
        sub bx, cx
        push bx
        push si
        push ax
        call SearchStr
NoneResult:
      mov     si,ax         ; Get returned position
      cmp     ax,fsize      ; Beyond end of file? 文末检测
      jae     fillout       ; Yes? Fill screen with spaces
      loop  show1           ;    else next line
; 2.9 - show the search line
        mov   bl,statatr
        mov   BYTE PTR cell[1],bl
        cmp     rows,23
        jne     fillout
        push    ds          ; Arg 1
        mov   ax,OFFSET statSearch  ; Arg 2
        push    ax
        mov   ax,24         ; Arg 3
        push    ax
        push    cell            ; Arg 4
        call  CellWrt
        jmp   SHORT pagedone    ; Get out if done

; 2.10 - Fill the rest  with spaces
fillout:
        dec   cx            ; Adjust
      jcxz  pagedone
      mov     al,80         ; Columns times remaining lines
      mul     cl
      push  sbuffer     ; Arg 1
      push  ax          ; Arg 2
      push  cell            ; Arg 3
      call  CellFil     ; Fill screen with spaces 通过调用 CellFil (sbuffer,count,ce
ll)
      push  ss          ; Restore DS from SS
      pop     ds

pagedone:
```

```
        pop   bp
      ret     2
Pager     ENDP

; 2.11 - Cellwrt function 向屏幕缓冲区写一行
; Procedure CellWrt (segment,offset,line,cell)
; Input     Stack variables:
;                                     1 - segment of line 缓冲段
;                                  2 - offset   缓冲偏移
;                                  3 - line number 行号
;                                  4 - attribute   配色
; Output    Line to screen buffer

CellWrt   PROC
      push  bp
      mov     bp,sp
      sub     dx,dx         ; Clear as flag for scan
      cmp     cga,1         ; CGA?
      jne     noscan
      mov     dx,03DAh      ; Load port #
noscan:
        mov   es,vidadr     ; Load screen buffer segment
      mov     ds,[bp+10]        ; Buffer segment
      mov     si,[bp+8]     ; Buffer position
      mov     cx,80         ; Cells per row
      mov     ax,[bp+6]     ; Starting row
      mov     bx,80*2       ; Bytes per row
      mul     bl            ; Figure columns per row
      mov     di,ax         ; Load as destination
      mov     bx,di         ; Save start for tab calculation
      mov     ax,[bp+4]     ; Attribute
movechar:
        lodsb                       ; Get character
      cmp     al,13         ; CR?
      je      fillspc
      cmp     al,9          ; Tab?
      jne     notab
      call  filltab     ; Yes? fill with spaces
      jcxz  nextline        ; If beyond limit done
      jmp     SHORT movechar

notab:
        or    dx,dx         ; CGA?
      je      notab2
      call  Retrace     ; Yes? Write during retrace
      loop  movechar
      jmp     SHORT nextline
```

```
notab2:
        stosw                       ; Write
      loop  movechar
      jmp     SHORT nextline    ; Done

fillspc:
        mov   al," "        ; Fill with space
      or      dx,dx         ; CGA?
      je      space2
space1:
        call    Retrace     ; Yes? Write during retrace
      loop  space1
      inc     si            ; Adjust
      jmp     SHORT exit        ; Done

space2:
        rep   stosw         ; Write
      inc     si            ; Adjust for LF
      jmp     SHORT exit        ; Done

nextline:
        mov   ah,10         ; Search for next line feed
chklf:
        lodsb               ; Load and compare
      cmp     al,ah
      loopne  chklf

exit:
        mov   ax,si         ; Return position
      pop     bp
      ret     8
CellWrt   ENDP


; 2.12 - CellFil function 向屏幕显示一定数量的字符
; Procedure CellFil (segment,count,cell)
; Input     Stack variables:
;                                       1 - segment of text (offset 0)
;                                 2 - number    of characters   数量
;                                 3 - attribute and character 配色
; Output    Characters to screen buffer

CellFil   PROC
      push  bp
      mov     bp,sp
      sub     dx,dx         ; Clear as flag for scan
      cmp     cga,1         ; CGA?
      jne     noscan2
```

```
      mov     dx,03DAh      ; Load port #

noscan2:
        mov   es,vidadr     ; Load screen buffer segment
      mov     ds,[bp+8]     ; Buffer segment (position 0)
      mov     cx,[bp+6]     ; Characters to fill
      mov     ax,[bp+4]     ; Attribute
      or      dx,dx         ; CGA?
      je      fillem2
fillem1:
        call    Retrace     ; Yes? Write during retrace
      loop  fillem1
      jmp     SHORT filled      ; Done
fillem2:
        rep   stosw         ; Write

filled:
        pop   bp
      ret     6
CellFil   ENDP


; 2.13 - FillTab function 用空格填充Tab
; Procedure FillTab
; Input     BX points to start of line, DI points to current position  各种指向行位置
; Output    Spaces to screen buffer

FillTab   PROC
      push  bx
      push  cx

      sub     bx,di         ; Get current position in line
      neg     bx
      shr     bx,1          ; Divide by 2 bytes per character

      mov     cx,8          ; Default count 8
      and     bx,7          ; Get modulus
      sub     cx,bx         ; Subtract
      mov     bx,cx         ; Save modulus

      mov     al," "        ; Spaces
      or      dx,dx         ; CGA?
      je      tabem2

tabem1:
        call    Retrace     ; Yes? Write during retrace
      loop  tabem1
      jmp     SHORT tabbed
```

```
tabem2:
        rep   stosw          ; Write

tabbed:
        pop   cx
      sub     cx,bx          ; Adjust count
      jns     nomore         ; Make negative count 0
      sub     cx,cx
nomore:
        pop   bx
      ret
FillTab   ENDP


; 2.14 - GoBack of Buffer to search function
; Procedure GoBack
; Input     CX has number of lines; ES:DI has buffer position
; Output    Updates "linenum" and "pbuffer"

GoBack    PROC
      std                       ; Go backward
      neg     cx              ; Make count positive
      mov     dx,cx           ; Save a copy
      inc     cx              ; One extra to go up one
      or      di,di           ; Start of file?
      je      exback          ; If so, ignore
findb:
        push    cx            ;   else save count
      mov     cx,0FFh         ; Load maximum character count
      cmp     cx,di           ; Near start of buffer?
      jl      notnear         ; No? Continue
      mov     cx,di           ;   else search only to start
notnear:
        repne   scasb           ; Find last previous LF
      jcxz  atstart     ; If not found, must be at start
      pop       cx
      loop  findb
      cmp       linenum,0FFFFh  ; End of file flag?
      jne       notend      ; No? Continue
      add       di,2            ; Adjust for cr/lf
      mov     pbuffer,di        ; Save position
      call  EndCount        ; Count back to get line number
      ret
notend:
        sub   linenum,dx        ; Calculate line number
      jg      positive
      mov     linenum,1     ; Set to 1 if negative
positive:
```

```
        add   di,2          ; Adjust for cr/lf
      mov     pbuffer,di        ; Save position
      ret

atstart:  pop     cx
      sub     di,di         ; Load start of file
      mov     linenum,1     ; Line 1
      mov     pbuffer,di        ; Save position
exback:
        ret
GoBack    ENDP


; 2.15 - GoForwd of buffer to search function
; Procedure GoForwd
; Input     CX has number of lines; ES:DI has buffer position
; Output    Updates "linenum" and "pbuffer"

GoForwd   PROC
      cld                           ; Go forward
      mov     dx,cx         ; Copy count
findf:
        push    cx          ; Save count
      mov     cx,0FFh       ; Load maximum character count
      repne scasb           ; Find next LF
      jcxz  atend           ; If not found, must be at end
      cmp     di,fsize      ; Beyond end?
      jae     atend
      pop     cx
      loop  findf
      add     linenum,dx        ; Calulate line number
      mov     pbuffer,di        ; Save position
      ret

atend:
        pop   cx
      mov     di,pbuffer        ; Restore position
      ret
GoForwd   ENDP


; 2.16 - calculate the line number backward function
; Procedure EndCount
; Input     ES:DI has buffer position
; Output    Modifies "linenum"

EndCount  PROC
      push  di
```

```
      mov     al,13         ; Search for CR
      mov     linenum,0     ; Initialize

findstrt:
        inc   linenum       ; Adjust count
      mov     cx,0FFh       ; Load maximum character count
      cmp     cx,di         ; Near start of buffer?
      jl      notnear2      ; No? Continue
      mov     cx,di         ;   else search only to start
notnear2:
        repne     scasb         ; Find last previous cr
      jcxz    found         ; If not found, must be at start
      jmp         SHORT findstrt

found:
        pop   di
      ret
EndCount  ENDP

; 2.17 - search string in lines
; Procedure SearchStr
; Purpose: search string in certain line and call highlight func
; Input         - line number
;                       - address of string
;                   - endstring address of file
; output        None

SearchStr proc
        push    bp
        mov   bp,sp         ;Arg1
        push  si
        push  di
        push  ax
        push  cx
        push  es
        push  bx
      mov   di,[bp+6]       ;Arg2
      mov   [bp+10],di      ;Arg3
lp1:
        mov   es,sbuffer
        mov   di,[bp+6]             ;initilize primary address with input buffer
      mov   si,OFFSET statSearch
      add   si,9
      mov   cx,searchPos        ;length
      cmp   cx,0
      je    exitf
      cld
```

```
      repz  cmpsb
      jz    lp2
        mov     ax,[bp+6]         ;shifft 1 bit
        inc     ax
        mov   [bp+6],ax
        jmp lp3
lp2:
    mov   ax,[bp+10]          ; encount number of tab
    push    ax
    mov   ax,[bp+6]                     ; primary address load
    push    ax
    call    CalTab                          ; calculate the number
    mov     bx,ax                               ; store res
      mov       ax,[bp+8]           ;lines number
      push  ax
      mov       ax,[bp+6]
      sub     ax,[bp+10]         ; offset inlines
      add       ax,bx              ; total with tab
      push  ax
      mov       ax,searchPos
      push  ax
      call  HighLight               ; highlight the result string
        mov     ax,[bp+6]
    add     ax,searchPos
    mov     [bp+6],ax
    jmp     lp3
lp3:                                                ; check if this is the end of
lines
        mov     ax,[bp+4]
        cmp     di,ax
        jb      lp1                                 ; not equal to 1, not the end
< , jump to lp1
exitf:
        pop bx
        pop es
        pop cx
        pop ax
        pop di
        pop si
        pop bp
        ret 6

        SearchStr endp


; 2.18 - highlight function
; Procedure HighLight
;   Purpose     highlight the search string if exist
```

```
;   Input       - line number
;                       - primary address
;                       - characters number
;Output         None

HighLight proc
        push bp
        mov  bp,sp
        push ax
        push bx
        push cx
        push es
        push di
        mov  es,vidadr
        mov  ax,[bp+8]      ; Arg1
        mov  bx,80*2            ; characters number in the line
        mul  bl
        mov  di,ax
        add  di,[bp+6]
        add  di,[bp+6]          ; Arg2
        mov  cx,[bp+4]         ; Arg3
        cmp  cx,0
        jne  t2
t1:
        pop di
        pop es
        pop cx
        pop bx
        pop ax
        pop bp
        ret 6
t2:
        mov bx,es:[di] ;修改配色
        mov bh,statatr
        mov es:[di],bx
        add di,2
        dec cx
        cmp cx,0
        jne t2
        jmp t1

        HighLight endp



; 2.19 - whether EGA is active
; Procedure isEGA
; Input     None
```

```
; Output    0 if no; lines per screen if yes

isEGA     PROC
      push  bp
      push  es
      mov     ah,12h        ; Call EGA status function
      mov     bl,10h
      sub     cx,cx         ; Clear status bits
      int     10h
      sub     ax,ax         ; Segment 0 and assume no EGA
      jcxz  noega           ; If status still clear, no EGA

      mov     es,ax         ; ES=0
      test  BYTE PTR es:[487h],1000b ; Test active bit
      jnz     noega         ; If set, not active
      mov     ax,1130h      ; Get EGA information
      int     10h
      mov     al,dl         ; Return lines per screen
      cbw

noega:
        pop   es
      pop     bp
      ret
isEGA     ENDP


; 2.20 - Converts integer to string
; Procedure BinToStr (number,address)
; Input     Stack arguments: 1 - Number to convert; 2 - Near address for write
; Output    AX has characters written
BinToStr  PROC
      push  bp
      mov     bp,sp
      mov     ax,[bp+6]     ; Arg 1
      mov     di,[bp+4]     ; Arg 2

      sub     cx,cx         ; Clear counter
      mov     bx,10         ; Divide by 10

; Convert and save on stack backwards

getdigit:
        sub   dx,dx         ; Clear top
      div     bx            ; Divide to get last digit as remainder
      add     dl,"0"        ; Convert to ASCII
      push  dx          ; Save on stack
      or      ax,ax         ; Quotient 0?
```

```
      loopnz  getdigit      ; No? Get another

; Take off the stack and store forward

      neg     cx            ; Negate and save count
      mov     dx,cx
putdigit:
        pop   ax            ; Get character
      stosb                 ; Store it
      loop  putdigit
      mov     ax,dx     ; Return digit count

      pop     bp
      ret     4
BinToStr  ENDP


; 2.21 - wirte in CGA model
; Procedure Retrace
; Input     ES:DI has screen buffer position, AX has cell
; Output    Character to screen buffer

Retrace   PROC
      push  bx
      mov     bx,ax         ; Save character
lscan2:
        in    al,dx         ; Look in the port
      shr     al,1          ;   until it goes low
      jc      lscan2
      cli
hscan2:
        in    al,dx         ; Look in the port
      shr     al,1          ;   until it goes high
      jnc     hscan2
      mov     ax,bx         ; Restore and write it
      stosw
      sti
      pop     bx
      ret
Retrace   ENDP

; 2.22 - Calculate the tab number
; Procedure CalTab
; Input         - initial offset
;                       - end offset
; Output        the number of tab to placeholder
CalTab proc
        push    bp
```

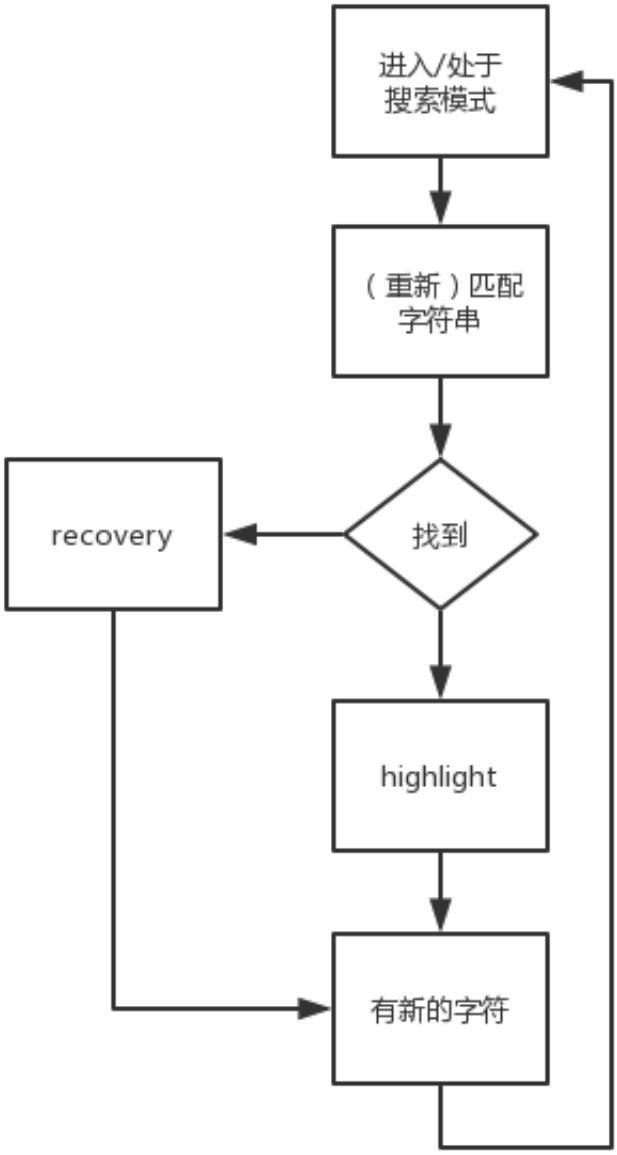
```
        mov   bp,sp
        push  di
        push  bx
        xor     ax, ax
        mov   di,[bp+6]          ;Arg 1
        mov   [bp+8],di              ;Arg 2
lb1:
        mov   bl, es:[di]
        cmp   bl,9
        jne   lb2
        mov   bx,di
        sub   bx,[bp+8]        ; calculate the tab position
        add   bx,ax
        and   bx,7
        sub   bx,8
        neg   bx
        dec   bx               ; exinclude the initial one
        add   ax,bx
lb2:
        inc     di
        cmp   di,[bp+4]
        jb    lb1
lb3:
        pop     bx
        pop     di
        pop     bp
        ret     4
        CalTab endp

  END
```

三个分析题目

> 如何高亮显示

我是通过highlight函数实现，函数以line number， primary address，characters number为输入，对配色进行修改，达到高亮的效果。 这里的输入包含了行号位置高亮字符等信息，所以调用是在search阶段匹配后调用的，详见search匹配部分代码，修改配送的流程如下：



这样的好处是当字符不匹配时方便恢复

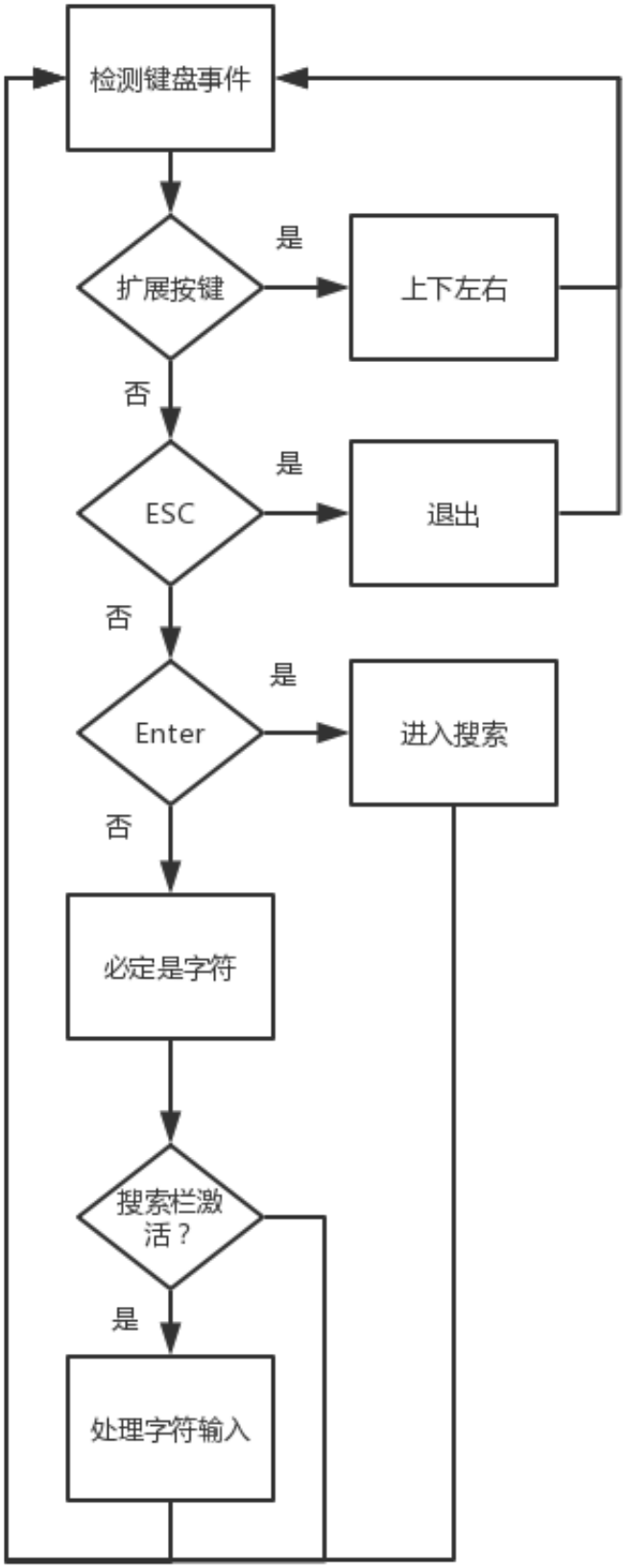
修改配色：

```
mov bx,es:[di]
mov bh,statatr
mov es:[di],bx
```

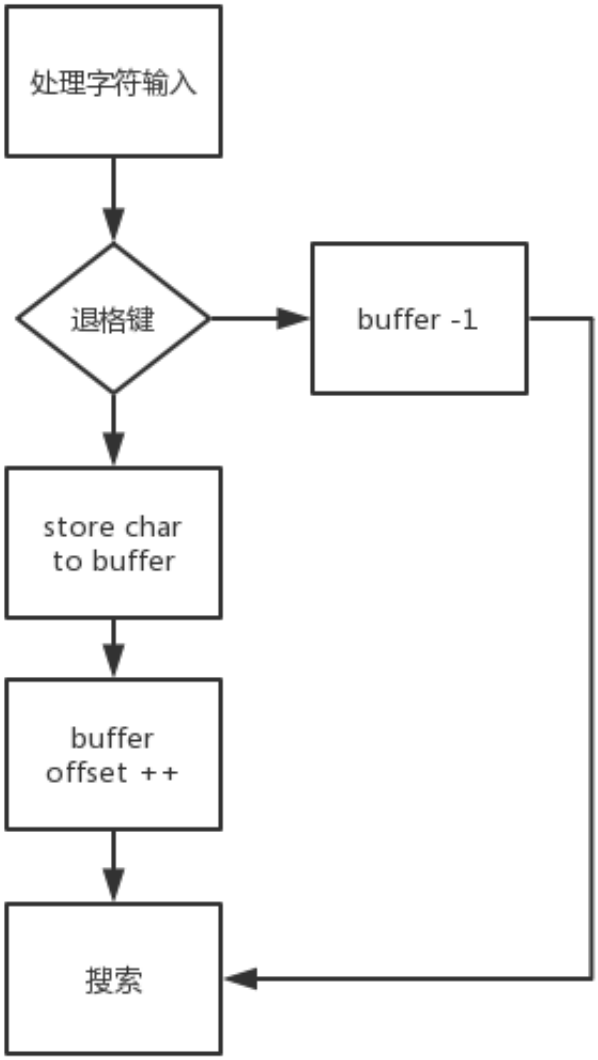
> 如何输入

这部分是对键盘事件的响应，click enter后会开始记录要搜索的字符串,手段就是比对键和ASCII值，模仿源程序就好。

```
- Handle keys events
nextkey:
    @GetKey 0,0,0           ; Get a key
    nextkey2:
              cmp     al,0          ; Is it a null?
              je      extended      ; Yes? Must be extended,  扩展键盘   code
              cmp     al,27         ; ESC退出
              je quit
              cmp al, 13            ; 搜索
              je  searchDriver
              jmp     nextkey       ; No? Ignore unknown command
```

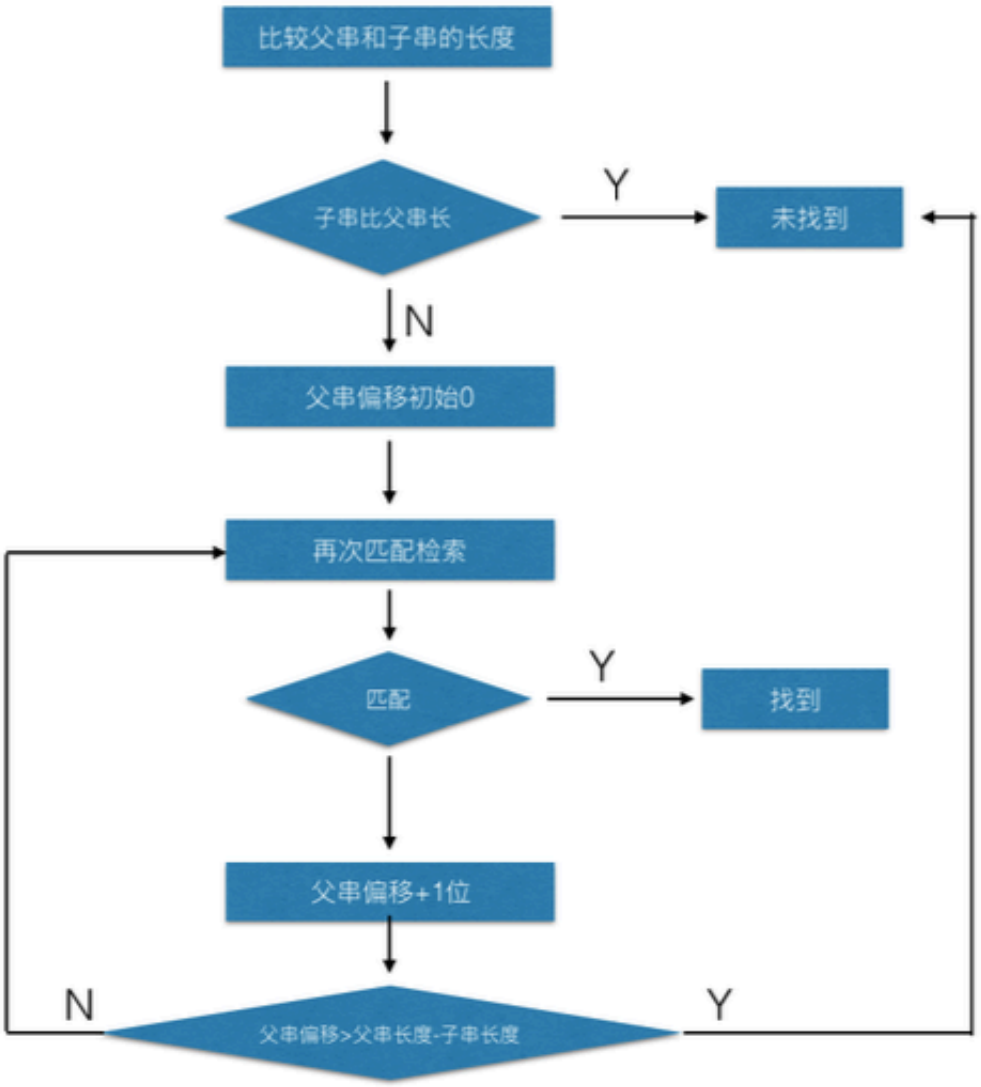

检测键盘事件
扩展按键
是
上下左右
否
ESC
是
退出
否
Enter
是
进入搜索
否
必定是字符
搜索栏激活？
是
处理字符输入

处理字符的流程：



> 如何匹配

这里我是借鉴的之前有一次作业，也是在父串中找子串，算法和当时类似，直接用当时的流程图解释：

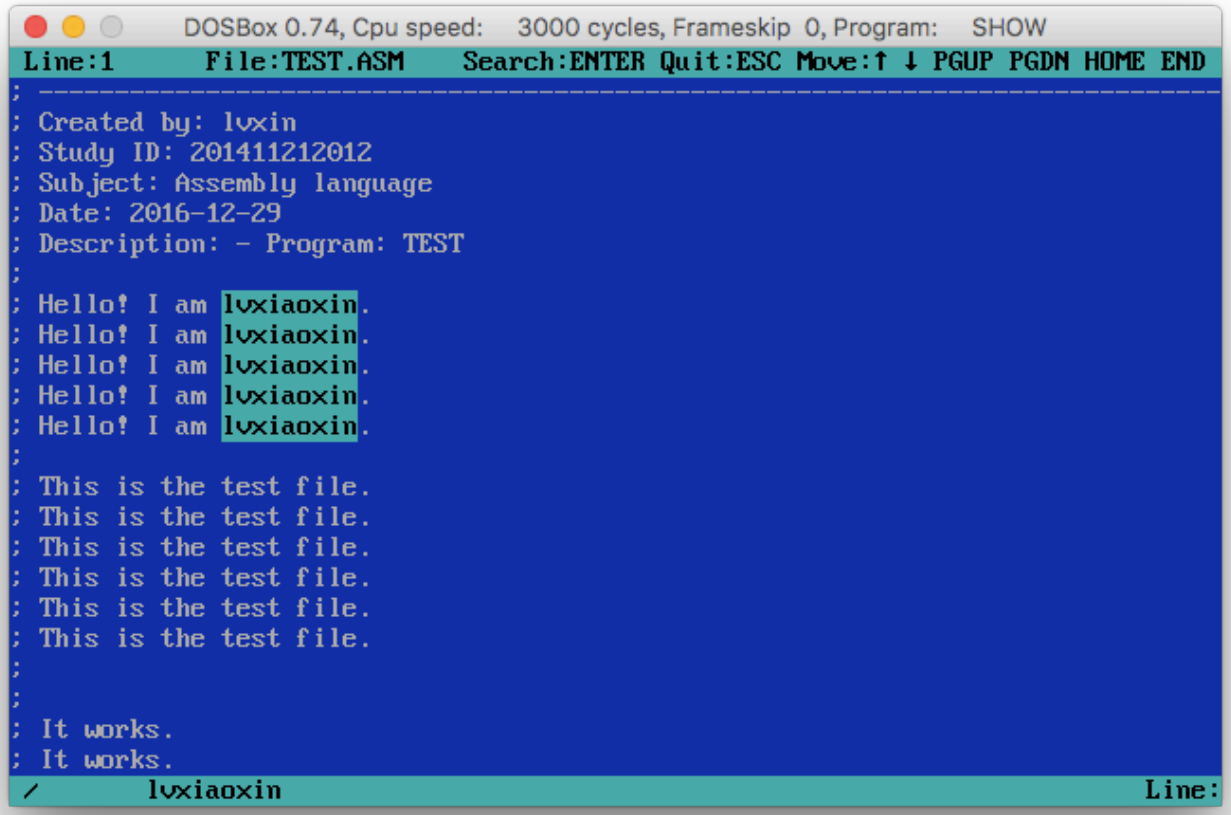


测试结果

运行方法

```
> SHOW.EXE TEST.ASM
```

结果如下图：



遇到的问题

1. 如果输入搜索的字符串过长，我会出现一个很神奇的bug。。。时间紧，没解决
2. 搜索栏和状态栏重叠，解决办法，开大搜索栏缓冲区

实验反思

不得不说，这么大工程量的汇编相比之前的作业来讲难度上升坡度还是太过于陡峭了，尤其在这学期选了5门选修的情况下，很是棘手。之前听许宏旭同学说原实验代码很乱，所以我在读的过程中保留了原有的注释，加入了自己的一些理解，然后做了一些简单的架构重整，然后才开始改动，单单是这个重抄一边的过程就很繁杂。

这个程序的屏幕高亮显示是通过对符合匹配的字符进行修改配色实现的，匹配字符的办法用的是之前有一次实验，检索字符串类似的子串、父串方法，但是我采取的是遍历每一行来找匹配字符，那一行字符的首地址，再加上结合缓冲偏移量，通过这种办法来获取和锁定字串位置。

在调试过程中，因为我使用的是Mac系统，DOSBOX更是及其不方便，经常崩掉，后来为了调试方便，我决定给代码加上分层目录，方便查找，事实证明这也是很好的一种编程风格。

最后，我认为，这门课主要目的不是学习8086汇编本身一种语言，毕竟现在用汇编去写东西的机会已经很少了。主要目的在于加强对一些底层包括寄存器等在内的理解，从而更好地优化自己的高级语言代码。此外，在一些其他领域，比如反汇编看懂别人代码等也是很有用途，所以希望在以后的课程中能适量加入这些应用，激发学生的兴趣。