Computer Organization & Assembly Language Lab Project

Command Line Text Editor

Submitted to: Sir. Daniyal Baig

Introduction

- A Simple console based text editor written in 8086 MASM Style Assembly language.
- The Command Line or Console is a text-based interface that was used early in DOS OS and still being used today.
- This project is a simple clone of that same retro environment

Features

- Lets user enter a custom name for their text document.
- It has a cursor that can be used to navigate around the characters on screen using arrow keys.
- User can delete characters while navigating through.
- User can get to a newline using Enter.

Features

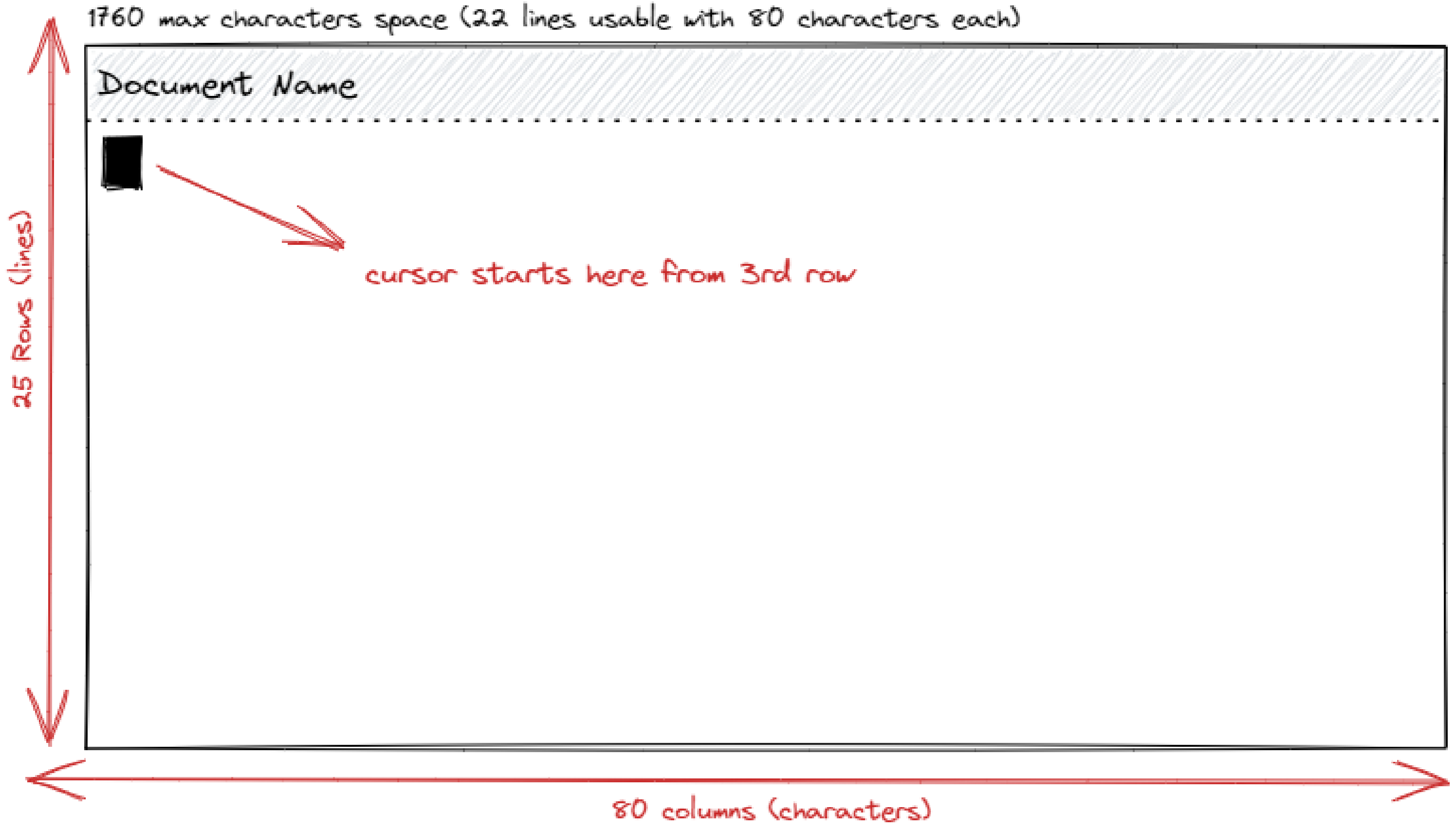
- Furthermore there will be file handling mechanism using which we can save and open files and work on them.
- User can close the program using escape key.
- For editor to work in that manner it will have special pre-defined shortcut keys that can decide the function.

Editor Interface

80x25 (emu 8086 Console Window)

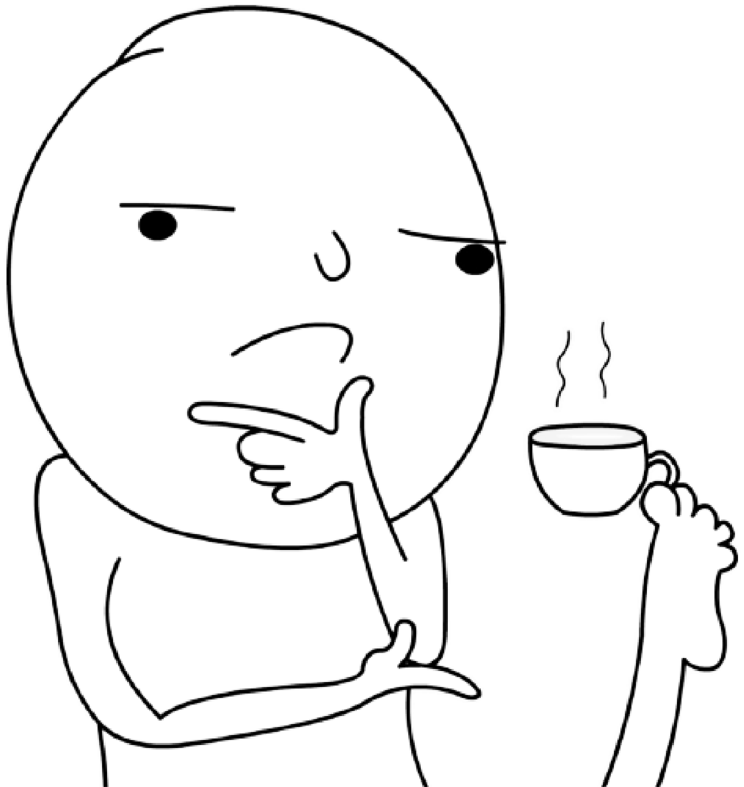
80x25 (Matrix Array to store characters)

1760 max characters space (22 lines usable with 80 characters each)



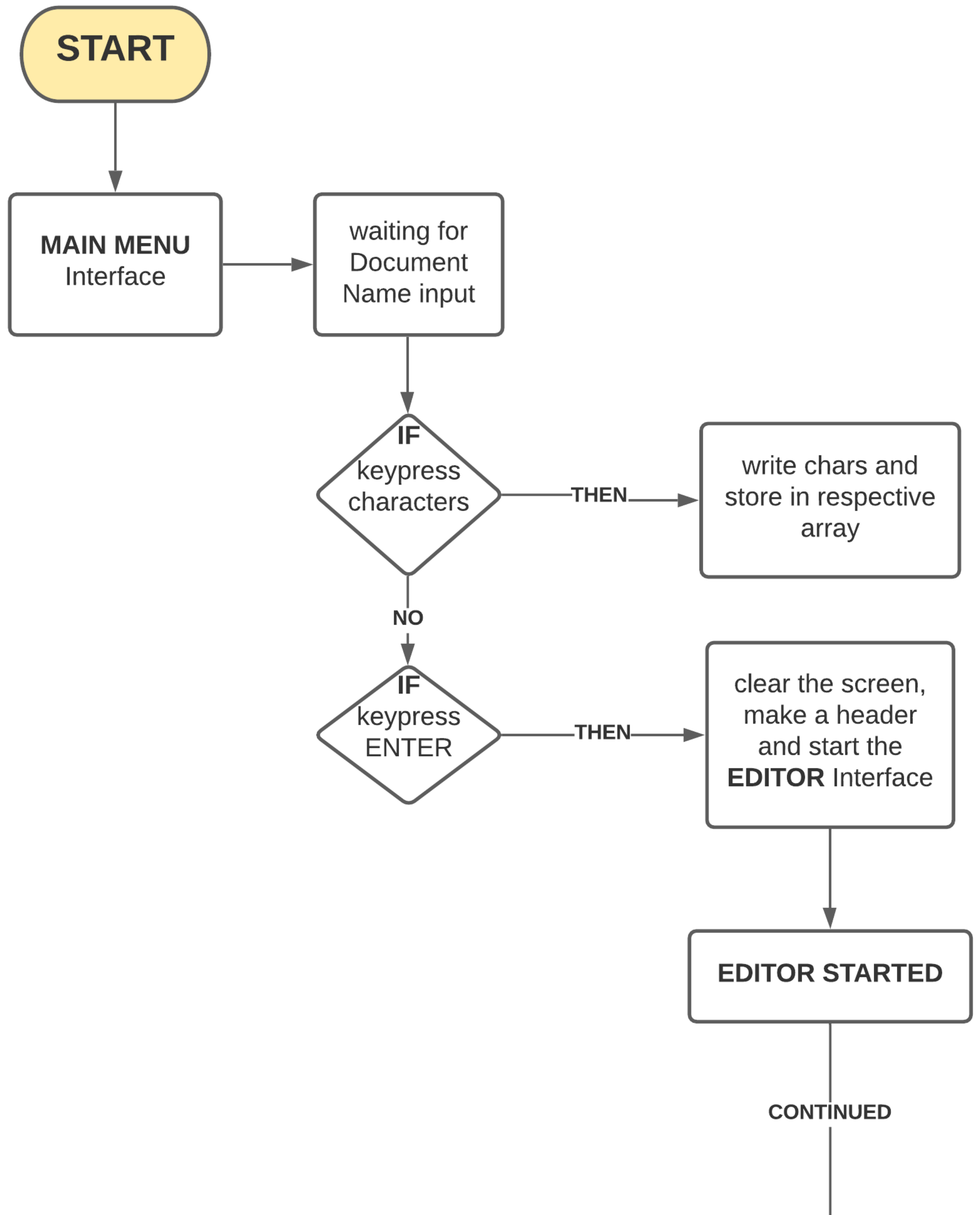
but how does it work?

- Basically It’s divided into 3 main phases
- LET’S SEE HOW IT WORKS ;)



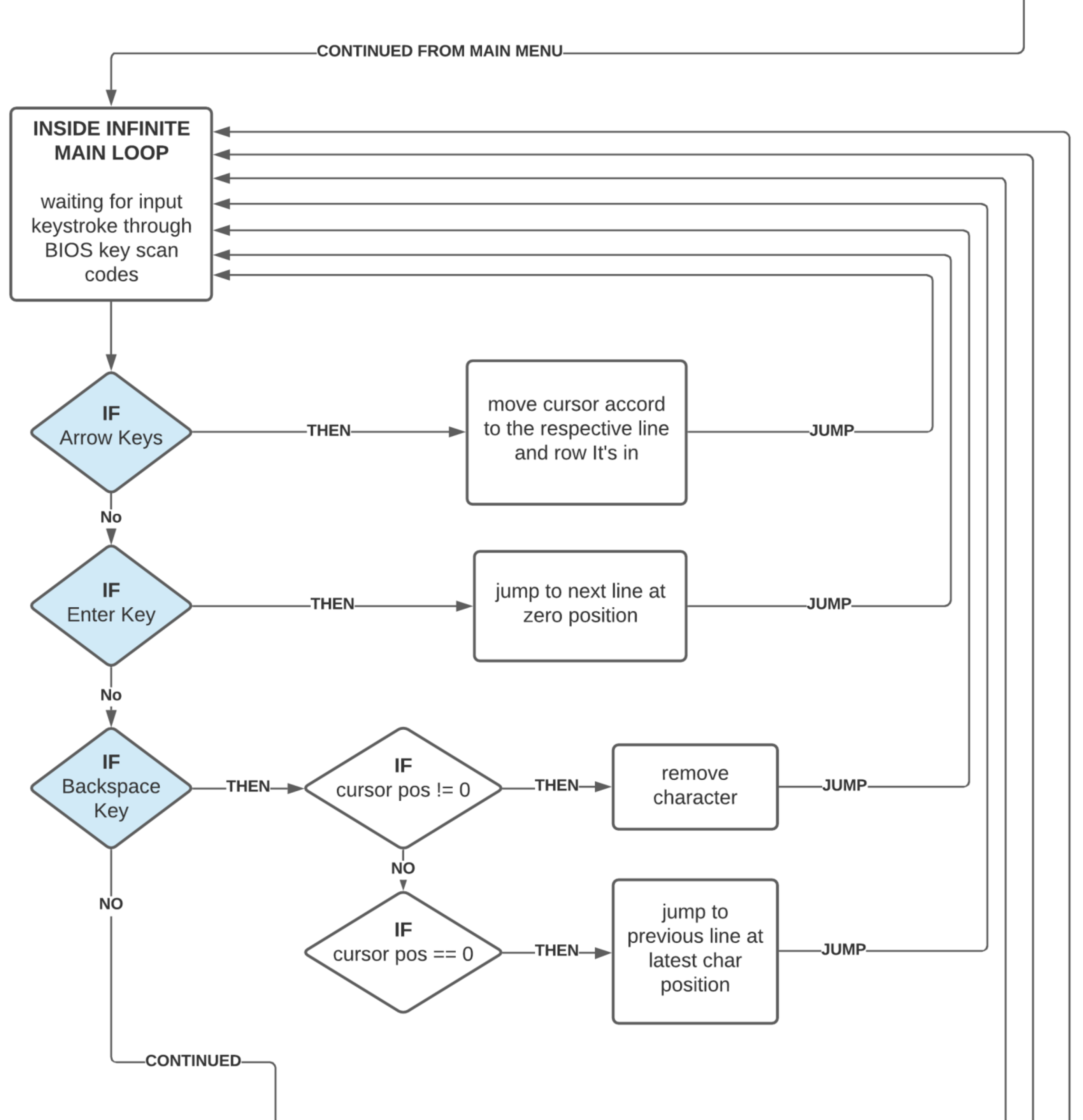
Editor Working # 1

- A simple main menu to prompt user for document name


START
MAIN MENU Interface
waiting for Document Name input
IF keypress characters
THEN
write chars and store in respective array
NO
IF keypress ENTER
THEN
clear the screen, make a header and start the EDITOR Interface
EDITOR STARTED
CONTINUED

Editor Working # 2

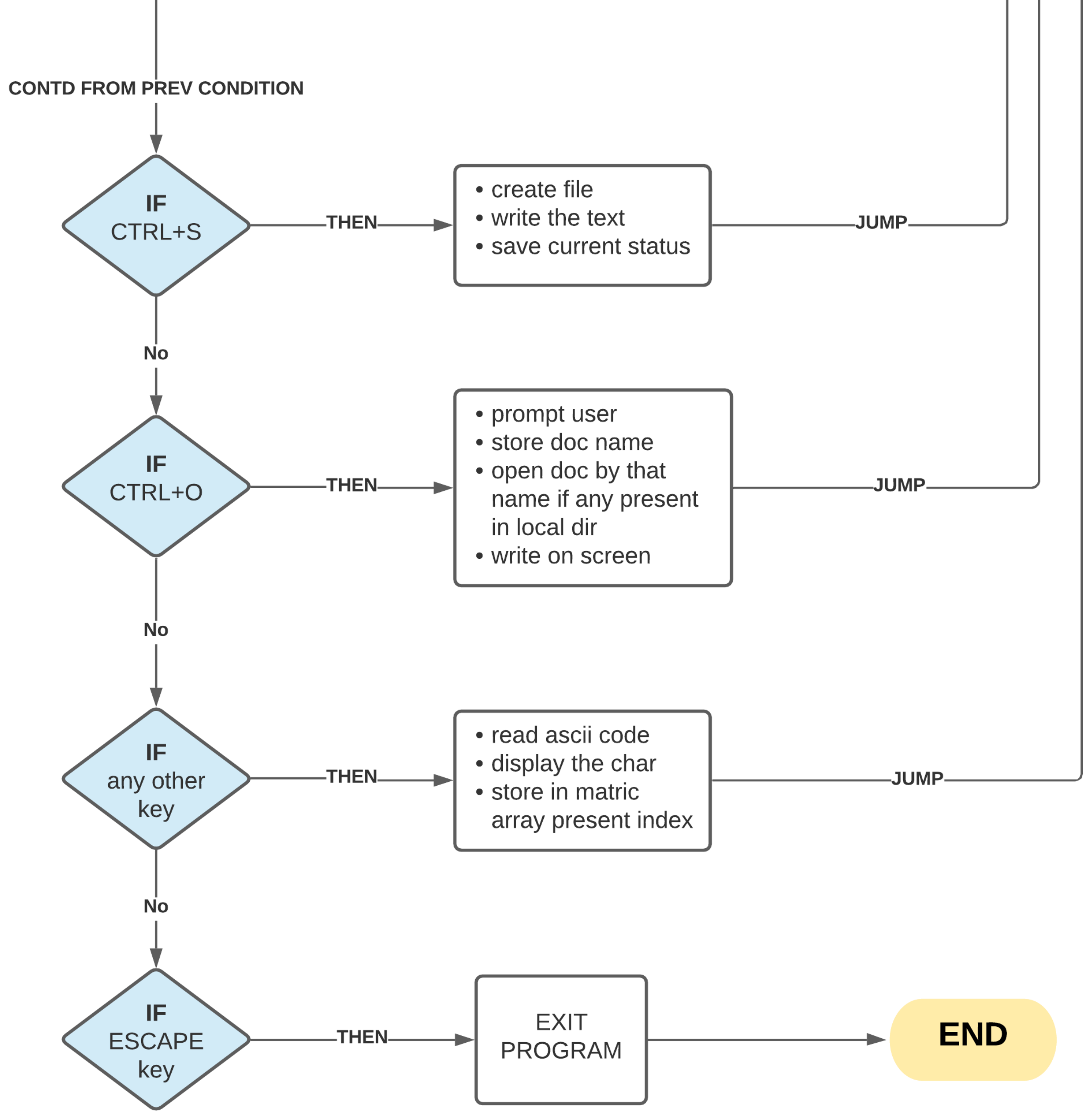
- Graphics Interrupt (10h) has a big role in this whole project
- All the navigation is done through using that interrupt
- Every function has it's own key



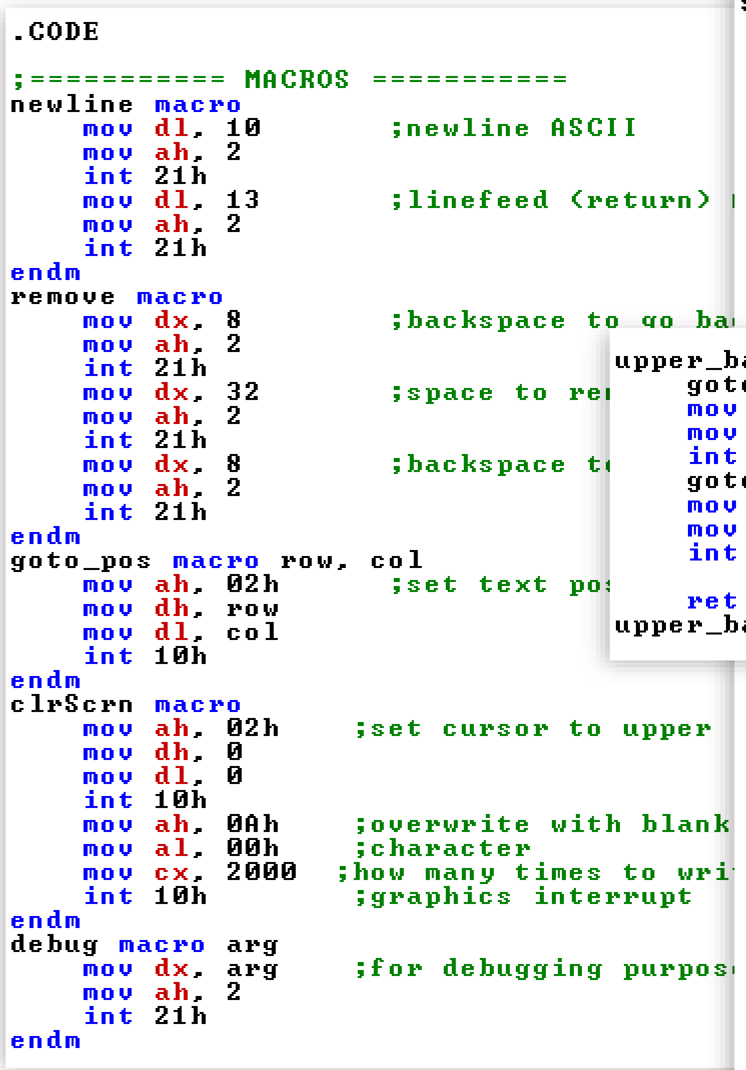
CONTINUED FROM MAIN MENU
INSIDE INFINITE MAIN LOOP
waiting for input keystroke through BIOS key scan codes
IF Arrow Keys
THEN
move cursor accord to the respective line and row It's in
JUMP
No
IF Enter Key
THEN
jump to next line at zero position
JUMP
No
IF Backspace Key
THEN
IF cursor pos != 0
THEN
remove character
JUMP
NO
IF cursor pos == 0
THEN
jump to previous line at latest char position
JUMP
NO
CONTINUED

Editor Working # 3

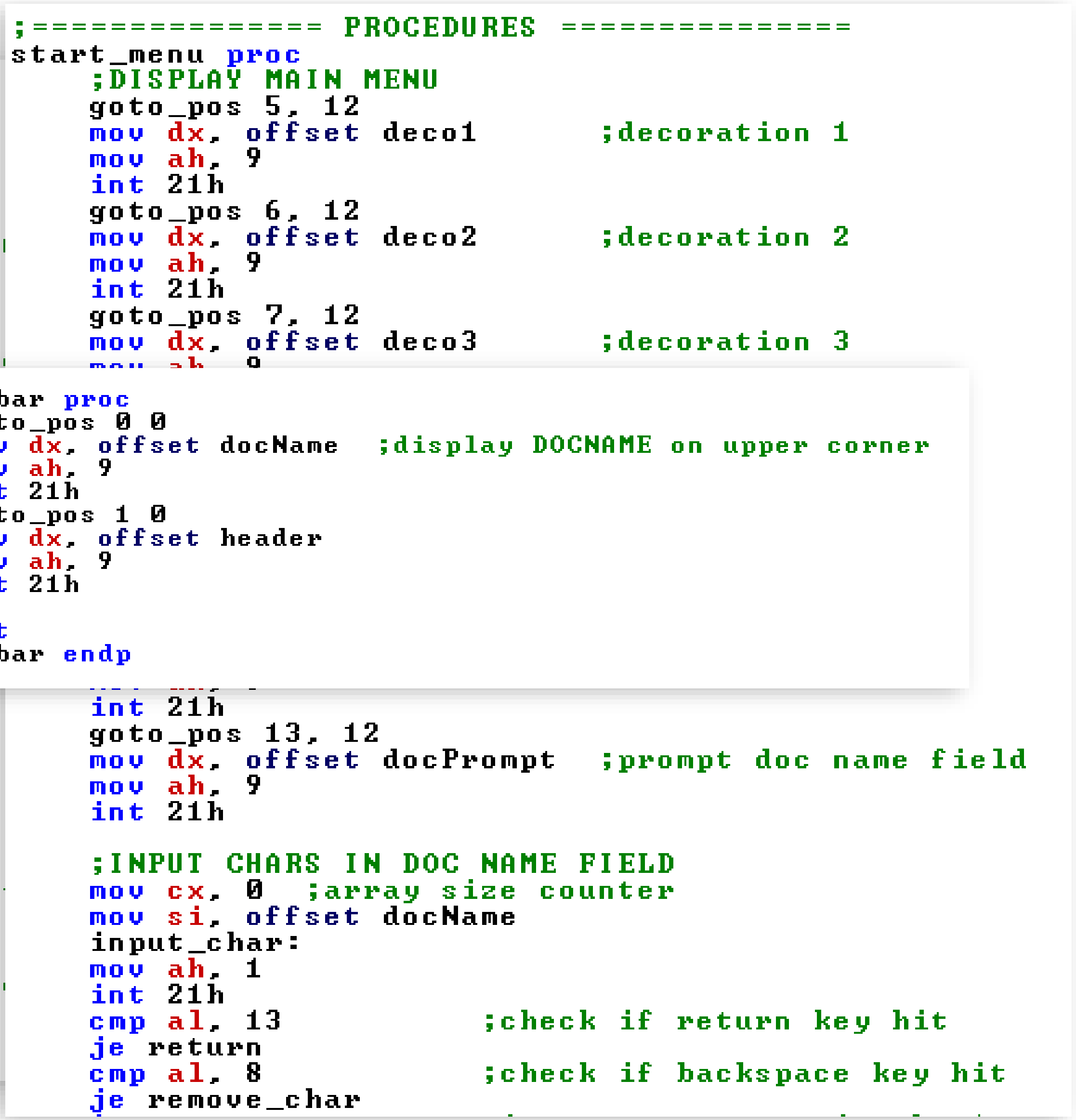
- For every function there is a label, also some macros and procedures
- Whenever we write something on screen it stores that into an array called matrix



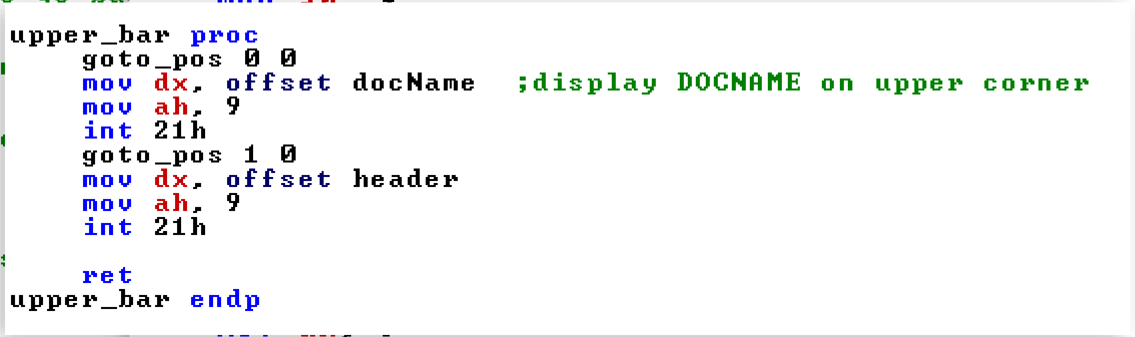
Code Structure - Macros & Procedures



```
.CODE

;=========== MACROS ===========
newline macro
     mov dl, 10        ;newline ASCII
     mov ah, 2
     int 21h
     mov dl, 13        ;linefeed (return)
     mov ah, 2
     int 21h
endm
remove macro
     mov dx, 8         ;backspace to go ba
     mov ah, 2
     int 21h
     mov dx, 32        ;space to re
     mov ah, 2
     int 21h
     mov dx, 8         ;backspace t
     mov ah, 2
     int 21h
endm
goto_pos macro row, col
     mov ah, 02h       ;set text pos
     mov dh, row
     mov dl, col
     int 10h
endm
clrScrn macro
     mov ah, 02h      ;set cursor to upper
     mov dh, 0
     mov dl, 0
     int 10h
     mov ah, 0Ah      ;overwrite with blank
     mov al, 00h      ;character
     mov cx, 2000    ;how many times to wri
     int 10h          ;graphics interrupt
endm
debug macro arg
     mov dx, arg      ;for debugging purpos
     mov ah, 2
     int 21h
endm
```

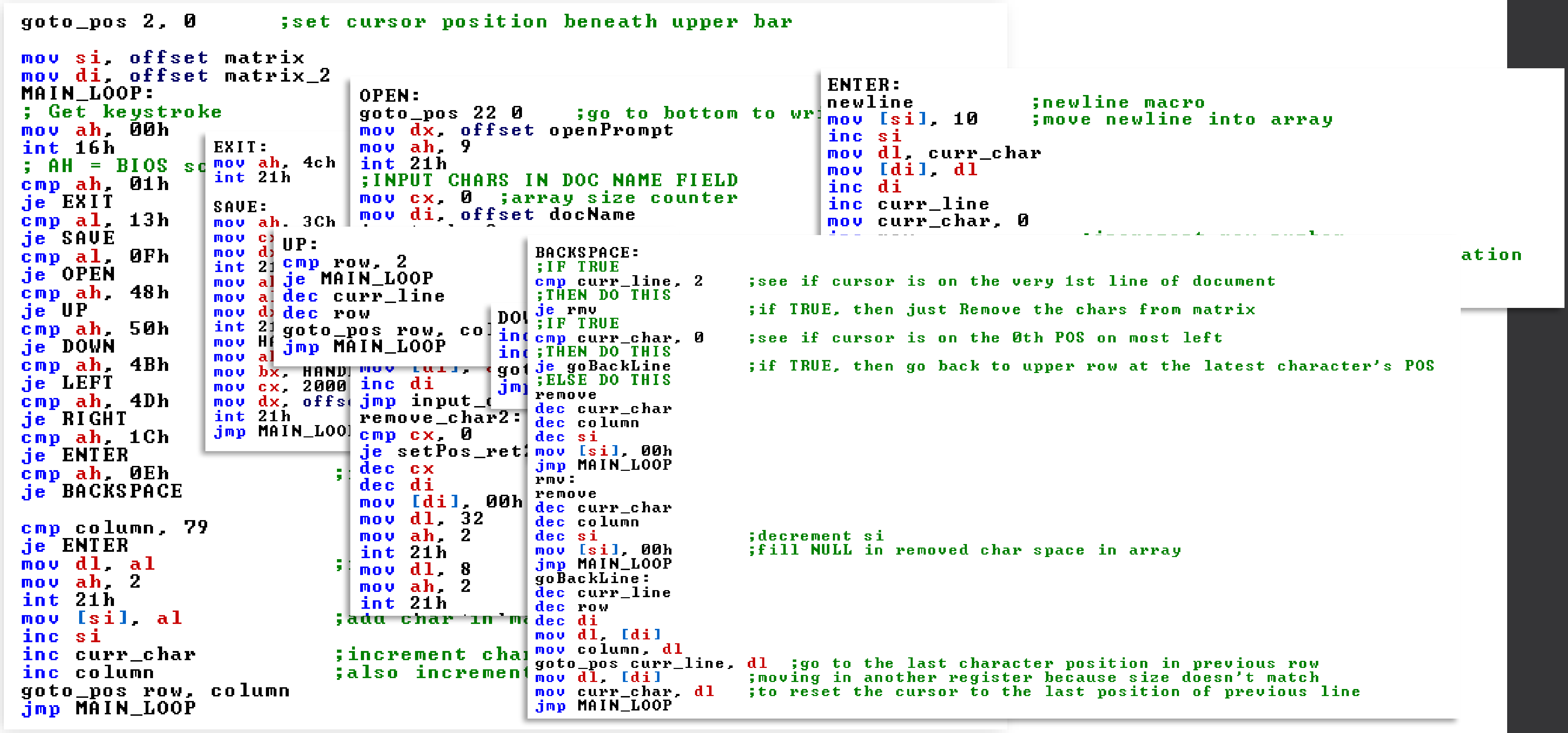


```
;================ PROCEDURES ================
start_menu proc
    ;DISPLAY MAIN MENU
    goto_pos 5, 12
    mov dx, offset deco1      ;decoration 1
    mov ah, 9
    int 21h
    goto_pos 6, 12
    mov dx, offset deco2      ;decoration 2
    mov ah, 9
    int 21h
    goto_pos 7, 12
    mov dx, offset deco3      ;decoration 3
    mov ah, 9

    mov ah, 9
    int 21h
    goto_pos 13, 12
    mov dx, offset docPrompt  ;prompt doc name field
    mov ah, 9
    int 21h

    ;INPUT CHARS IN DOC NAME FIELD
    mov cx, 0  ;array size counter
    mov si, offset docName
    input_char:
    mov ah, 1
    int 21h
    cmp al, 13            ;check if return key hit
    je return
    cmp al, 8             ;check if backspace key hit
    je remove_char
```



```
upper_bar proc
    goto_pos 0 0
    mov dx, offset docName  ;display DOCNAME on upper corner
    mov ah, 9
    int 21h
    goto_pos 1 0
    mov dx, offset header
    mov ah, 9
    int 21h

    ret
upper_bar endp
```

Code Structure – MAIN PROCEDURE



```
goto_pos 2, 0       ;set cursor position beneath upper bar

mov si, offset matrix
mov di, offset matrix_2
MAIN_LOOP:
; Get keystroke
mov ah, 00h
int 16h
; AH = BIOS sc
cmp ah, 01h
je EXIT
cmp al, 13h
je SAVE
cmp al, 0Fh
je OPEN
cmp ah, 48h
je UP
cmp ah, 50h
je DOWN
cmp ah, 4Bh
je LEFT
cmp ah, 4Dh
je RIGHT
cmp ah, 1Ch
je ENTER
cmp ah, 0Eh                   ;
je BACKSPACE

cmp column, 79
je ENTER
mov dl, al                    ;
mov ah, 2
int 21h
mov [si], al                  ;add char in ma
inc si
inc curr_char                 ;increment cha
inc column                    ;also increment
goto_pos row, column
jmp MAIN_LOOP
```

```
EXIT:
mov ah, 4ch
int 21h

SAVE:
mov ah, 3Ch
mov c
mov d
int 2
mov a
mov a
mov d
int 2
mov H
mov a
mov bx, HAND
mov cx, 2000
mov dx, offs
int 21h
jmp MAIN_LOO
```

```
UP:
cmp row, 2
je MAIN_LOOP
dec curr_line
dec row
goto_pos row, col
jmp MAIN_LOOP
```

```
OPEN:
goto_pos 22 0     ;go to bottom to wri
mov dx, offset openPrompt
mov ah, 9
int 21h
;INPUT CHARS IN DOC NAME FIELD
mov cx, 0   ;array size counter
mov di, offset docName

inc di
jmp input_c
remove_char2:
cmp cx, 0
je setPos_ret
dec cx
dec di
mov [di], 00h
mov dl, 32
mov ah, 2
int 21h
mov dl, 8
mov ah, 2
int 21h
```

```
DOV
inc
inc
got
jmp
```

```
ENTER:
newline             ;newline macro
mov [si], 10        ;move newline into array
inc si
mov dl, curr_char
mov [di], dl
inc di
inc curr_line
mov curr_char, 0
```

```
                                                                             ation
```

```
BACKSPACE:
;IF TRUE
cmp curr_line, 2      ;see if cursor is on the very 1st line of document
;THEN DO THIS
je rmv                ;if TRUE, then just Remove the chars from matrix
;IF TRUE
cmp curr_char, 0      ;see if cursor is on the 0th POS on most left
;THEN DO THIS
je goBackLine         ;if TRUE, then go back to upper row at the latest character's POS
;ELSE DO THIS
remove
dec curr_char
dec column
dec si
mov [si], 00h
jmp MAIN_LOOP
rmv:
remove
dec curr_char
dec column
dec si                ;decrement si
mov [si], 00h         ;fill NULL in removed char space in array
jmp MAIN_LOOP
goBackLine:
dec curr_line
dec row
dec di
mov dl, [di]
mov column, dl
goto_pos curr_line, dl   ;go to the last character position in previous row
mov dl, [di]          ;moving in another register because size doesn't match
mov curr_char, dl     ;to reset the cursor to the last position of previous line
jmp MAIN_LOOP
```

Takeaway

- It’s a reflection of how you would create a program to write and edit text back in 80s era, when personal computers were just beginning to be norm.
- It shows you how you can use Assembly to work with operating system device drivers, even create something to listen to your keystrokes like a keylogger running in background

QnA

- If you have any questions, feel free to ask