COAL PROJECT

"NOTE PAD"

CLASS: BSCS 3C

GROUP MEMBERS:

1. Fiza Muhammad Amin (2012202)
2. Qurat Ul Ain Sikandar (2012418)
3. Alina Mubarak Ali (2012196)
4. Shaz Syed (2012230)
5. Zohair Tahir (2012234)

Table of Contents

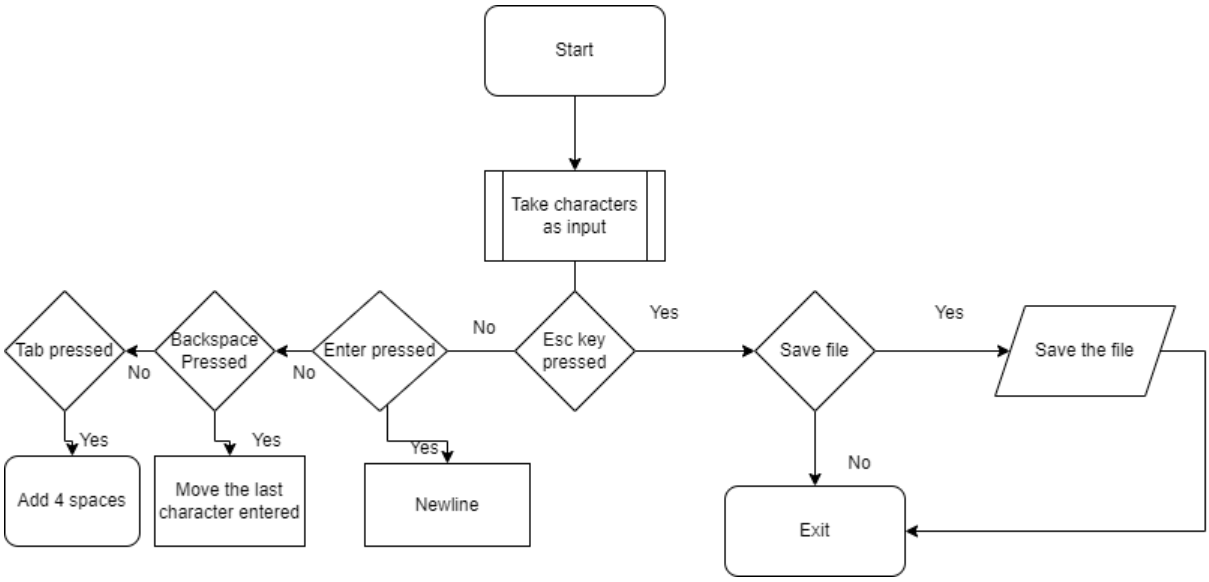
PROBLEM STATEMENT

Using emu8086 write a program in Assembly Language that can act as a notepad. Recall this type of program allows the user to type on screen. They may hit enter to go to next line, tab to start new paragraph, space to create blank space etc. As an added feature allow the user to save his work to a text file.

Project Scope

- Input characters to a limit of 500 characters.
- When Esc key pressed you can save the file in system or exit without saving.
- When enter pressed cursor points at the next line.
- When backspace pressed it removes the last character entered.
- When backspace pressed and cursor is at the start of the line so the cursor moves to the previous line.
- When tab pressed, 4 spaces are entered.

FLOW CHART


Start
Take characters as input
Esc key pressed
No
Yes
Save file
Yes
Save the file
No
Exit
Enter pressed
No
Yes
Newline
Backspace Pressed
No
Yes
Move the last character entered
Tab pressed
No
Yes
Add 4 spaces

SOURCE CODE

Project - Notepad

```
macro printMsg p1
lea dx, p1  ;loading the offset address of the parameter passed
mov ah, 9
int 21h
endm

macro newline
mov dl,10 ;interrupt for newline
mov ah,2
int 21h
mov dl,13 ;interrupt for return carriage
mov ah,2
int 21h
endm

macro storeValue p1
; al = 97
; p1 = al
; cx = 0
cmp cx, MAX_SIZE ;Checking max limit, limit the numbers of character to avoid crossing the size limit of 16-bit registers, max is 65,535 but for demo we made it to 500
je takeInput
```

```
; si = array address
; [si] = data at array address

mov [si], p1    ;move the content of p1 register to the array (replaces old array value)
add si, 1       ;points at the next value of the array (since we're using db (define byte), each gap to next value is of 1 bytes)
inc cx          ;text count increases by 1 for every character entered
endm

macro checkPreviousLine
push cx         ;storing text count because it will overwritten by 10h/03h

;Dealing if at the start of the newline and want to go back to the previous line
mov ah, 03h     ;get cursor position on the screen
mov bh, 0
int 10h         ;dh=row, dl=column
;CH = cursor start line, CL = cursor bottom line.

pop cx          ;getting the text counter value back to the cx

cmp dl, 0
jne delete_and_effect   ;if not at the start of line (column = 0) just continue without setting cursor
```

```
pop dx          ;getting the last postion of row and column where enter was
pressed

;Setting cursor to the previous line
mov ah, 2       ;set cursor position on the screen
mov bh, 0
inc dl          ;adjusting coloum by increasing it by 1 so it is after the last character
of previous line
int 10h         ;dh=row, dl=column
endm


.model small
.stack 100h

.data
errMessage db 'Wrong key pressed.. Please press either "y" or "n"$'
displayMessage db 'Would you like to save? Press "y" to save and exit or "n" to
exit without saving$'

file db "D:\SOFTWARES\emu8086\vdrive\C\work.txt"
handle dw ?

MAX_SIZE dw 500 ;Total number of characters allowed
tabIndent dw 4  ;Number of spaces per tab
```

array db 500 dup (?) ;an array which has '0' stored 500 times, max is 65,535 but for demo we made it to 500

```
.code
mov ax, @data
mov ds, ax

mov cx, 0 ;cx sometimes have garbage value on starting the program, to make sure that it's at 0, we used this command
mov si, offset array ;stores the starting address of array into si

takeInput:
mov ah, 7 ;stores the keyboard input into AL register
int 21h

cmp al, 27 ;whenever terminator ESC is pressed, you exit the loop 19 is the decimal code for terminator's ascii code
je processInput

cmp al, 13 ;whenever enter is pressed, a newline is displayed in text editor 13 is the decimal code for enter's ascii code
je enter

cmp al, 8 ;whenever backspace is pressed, the carriage is moved one space back to overwrite previously written keyword 8 is the decimal code for backspace's ascii code
je backspace
```

```
;Handling Tab by spliting it into backspaces as a global setting
cmp al, 9 ;whenever tab is pressed instead of letting the default tab print we split it into backspaces defined by the tab indentation variable
mov bx, tabIndent
je tab

storeValue al ;store input to the array

mov ah, 2 ;outputs value stored in dl on the screen
mov dl, al
int 21h


jmp takeInput

processInput:
newline
printMsg displayMessage

mov ah, 1 ;stores the keyboard input into AL register
int 21h   ;outputs it on a black screen
cmp al, 'y'
je save

cmp al, 'n'
```

```
je exit

newline
printMsg errMessage

jmp processInput

save:
push cx ;pushing cx to store the text count on the stack

;create and open file
mov ah, 3ch
mov cx, 0  ;normal file
mov dx, offset file  ;loading filepath, tells the program where to create the file
int 21h ;creates file when int 21h is called

mov handle, ax ;handle is used to read/write file

;write to file:
mov ah, 40h  ;syntax
mov bx, handle  ; handle to the the file
mov dx, offset array ;characters stored in this array
pop cx ; cx hold the total text count which represent the number of bytes to
write
int 21h
```

```
;close file handle
mov ah, 3eh ;syntax
mov bx, handle ;handle to close
int 21h

jmp exit

enter:
storeValue al

push cx     ;storing text count because it will overwritten by 10h/03h

mov ah, 03h ;get cursor position on the screen
mov bh, 0
int 10h     ;dh=row, dl=column

pop cx      ;getting the text counter value back to the cx

push dx     ;push the cursor position where newline is entered

newline
jmp takeInput

backspace:
cmp cx, 0    ;if index of array is pointing to zero, we skip this to avoid going
below array starting address
```

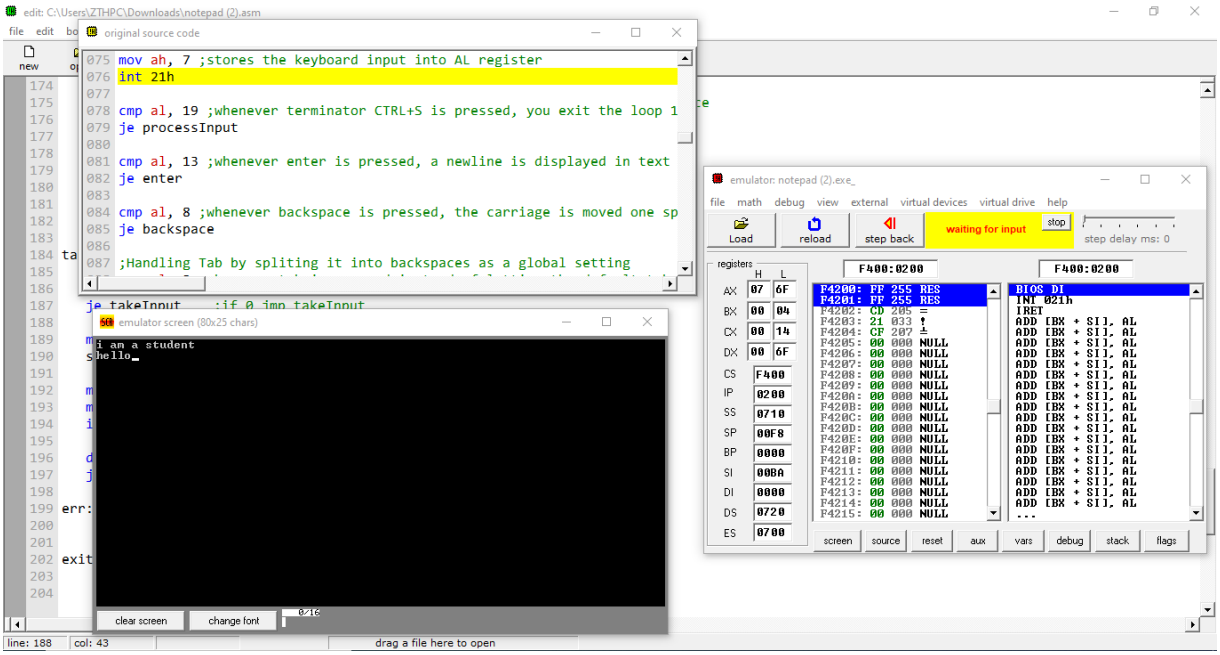
```
je takeInput

checkPreviousLine

delete_and_effect:
sub si, 1    ;move array pointer back 1 space to point previous value
sub cx, 1    ;reduce the text count by 1

;Giving effect of deleting character like in real notepad
mov dl, 8    ;backspace again to reset pointer to the deleted character space
mov ah, 2
int 21h

mov dl, 255  ;ascii for blank
mov ah, 2
int 21h

mov dl, 8    ;backspace again to reset pointer to the deleted character space
mov ah, 2
int 21h

mov [si], 0 ;store null at the place where we 'deleted' character in array

jmp takeInput
```

```
tab:

cmp bx, 0       ;comparing bx
je takeInput    ;if 0 jmp takeInput

mov al, 32      ;mov 32 in al, which is ascii for space
storeValue al   ;calling storeValue to store space in array

mov dl, al      ;printing the space on the screen
mov ah, 2
int 21h

dec bx          ;decrementing bx by 1
jmp tab

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

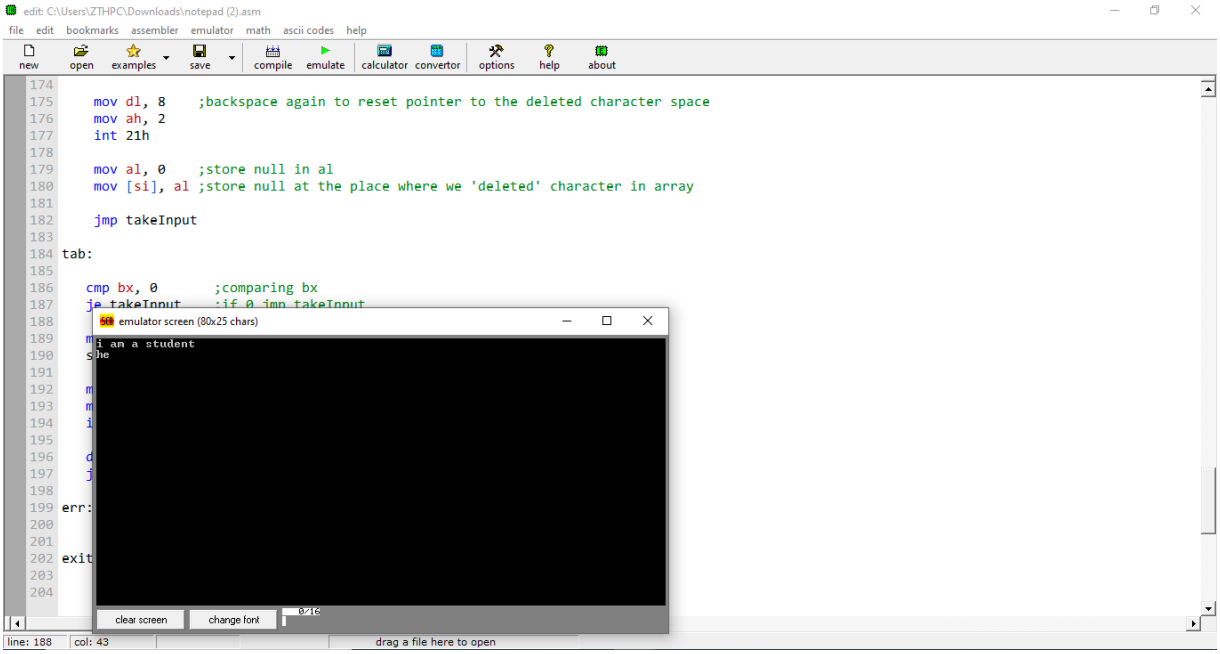
SCREENSHOTS

Output

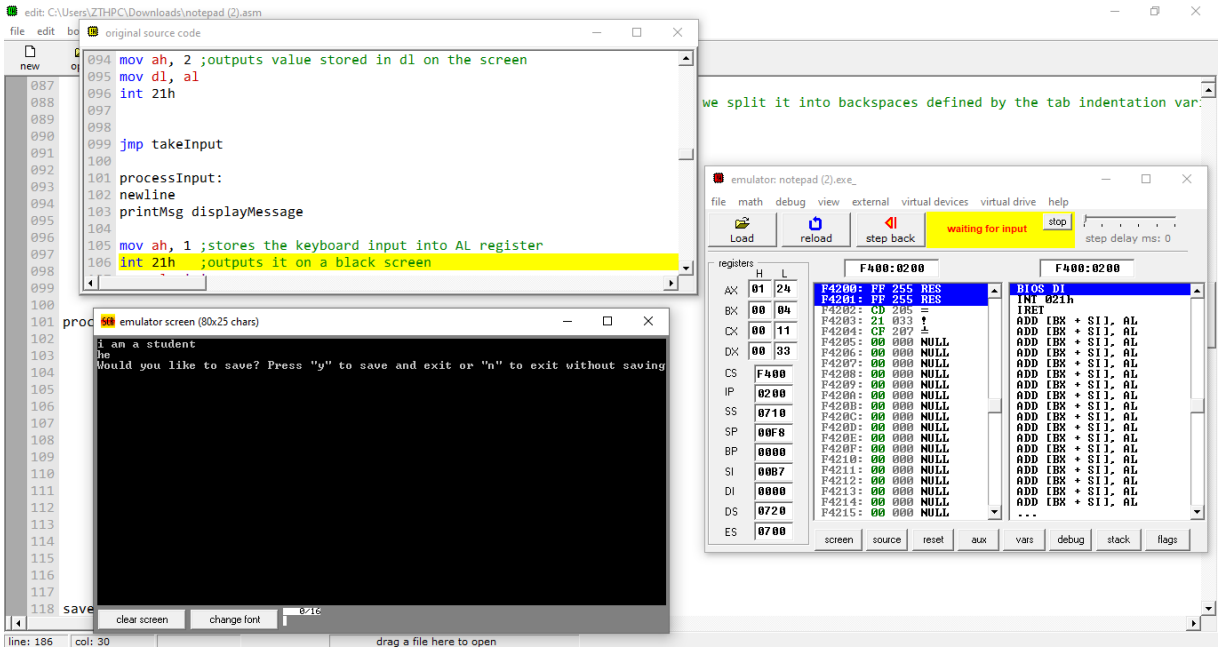
Enter pressed and cursor moves to the next line



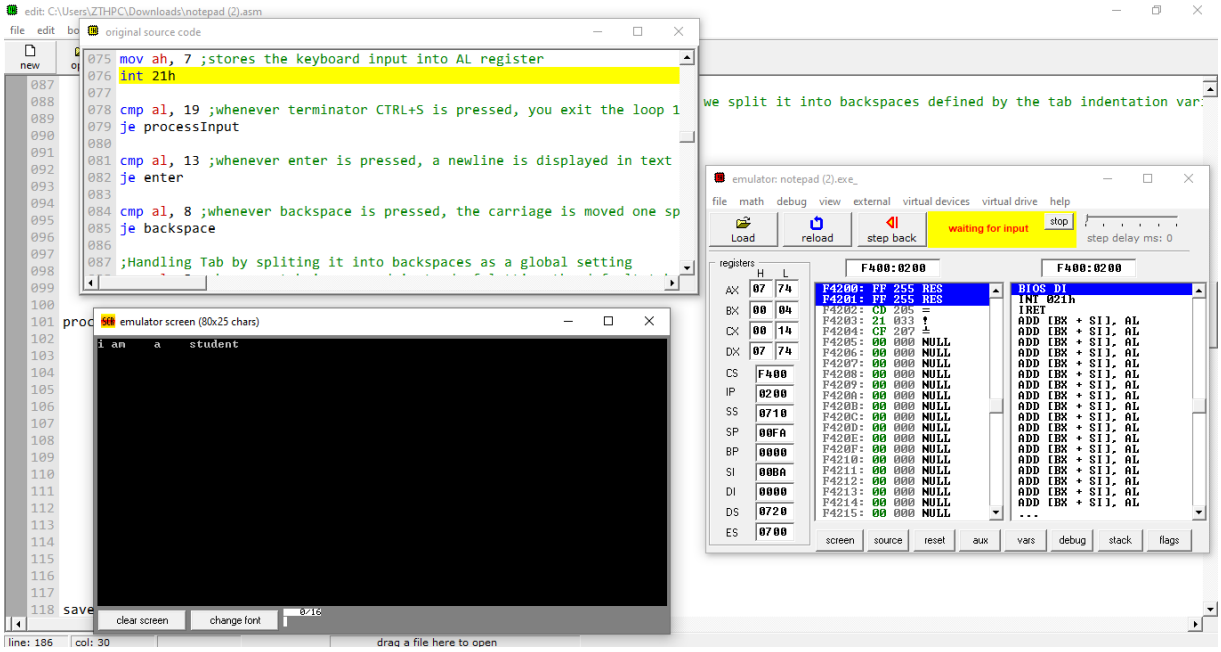
Backspace removes the characters entered



Esc key pressed



Tab pressed



File saved in folder