

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.....	3
Project Scope.....	3
FLOW CHART.....	4
CODE.....	5
SCREENSHOTS.....	14

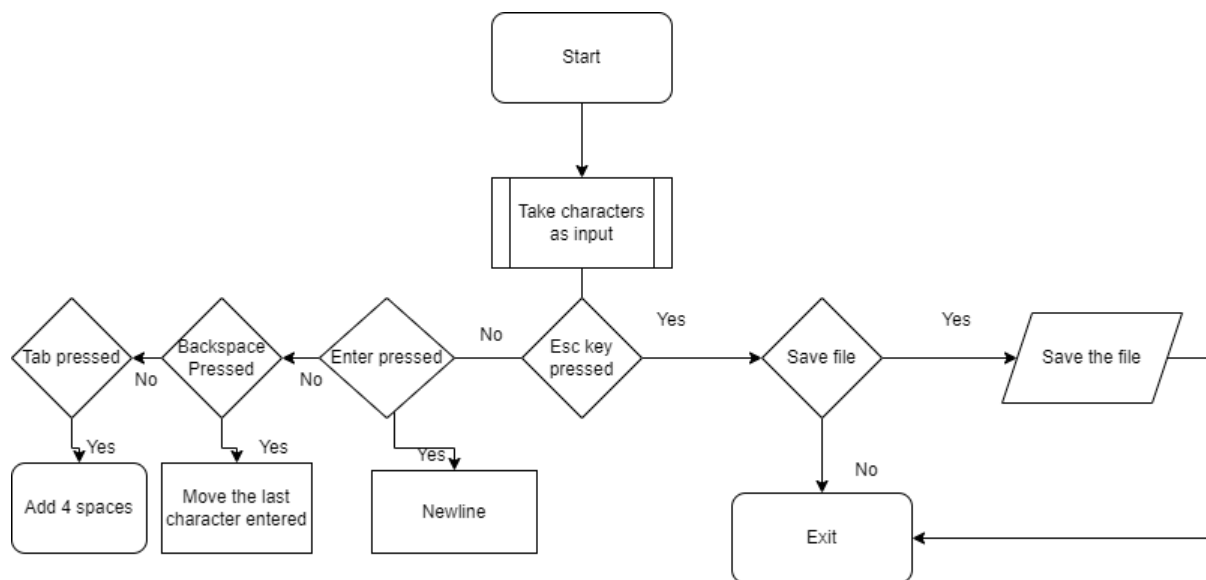
PROBLEM STATEMENT

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



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\vdribe\C\work.txt"
```

```
handle dw ?
```

```
MAX_SIZE dw 500 ;Total number of characters allowed
```

```
tablIndent 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 splitting 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 errorMessage

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

Backspace removes the characters entered

edit: C:\Users\ZTHPC\Downloads\notepad (2).asm

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

```

174
175     mov dl, 8      ;backspace again to reset pointer to the deleted character space
176     mov ah, 2
177     int 21h
178
179     mov al, 0      ;store null in al
180     mov [si], al   ;store null at the place where we 'deleted' character in array
181
182     jmp takeInput
183
184 tab:
185
186     cmp bx, 0      ;comparing bx
187     ja takeInput   ;if 0 jmp takeInput
188
189     emulator screen (80x25 chars)
190
191     i am a student
192
193
194
195
196
197
198
199 err:
200
201
202 exit
203
204

```

clear screen change font 0/16 drag a file here to open

line: 188 col: 43

Esc key pressed

edit: C:\Users\ZTHPC\Downloads\notepad (2).asm

file edit bo original source code

new open examples save compile emulate calculator convertor options help about

```

087
088     mov ah, 2 ;outputs value stored in dl on the screen
089     mov dl, al
090     int 21h
091
092     jmp takeInput
093
094 processInput:
095     newline
096     printMsg displayMessage
097
098     mov ah, 1 ;stores the keyboard input into AL register
099     int 21h ;outputs it on a black screen
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118 save

```

emulator screen (80x25 chars)

i am a student
the
Would you like to save? Press "y" to save and exit or "n" to exit without saving

clear screen change font 0/16 drag a file here to open

line: 186 col: 30

emulator: notepad (2).exe

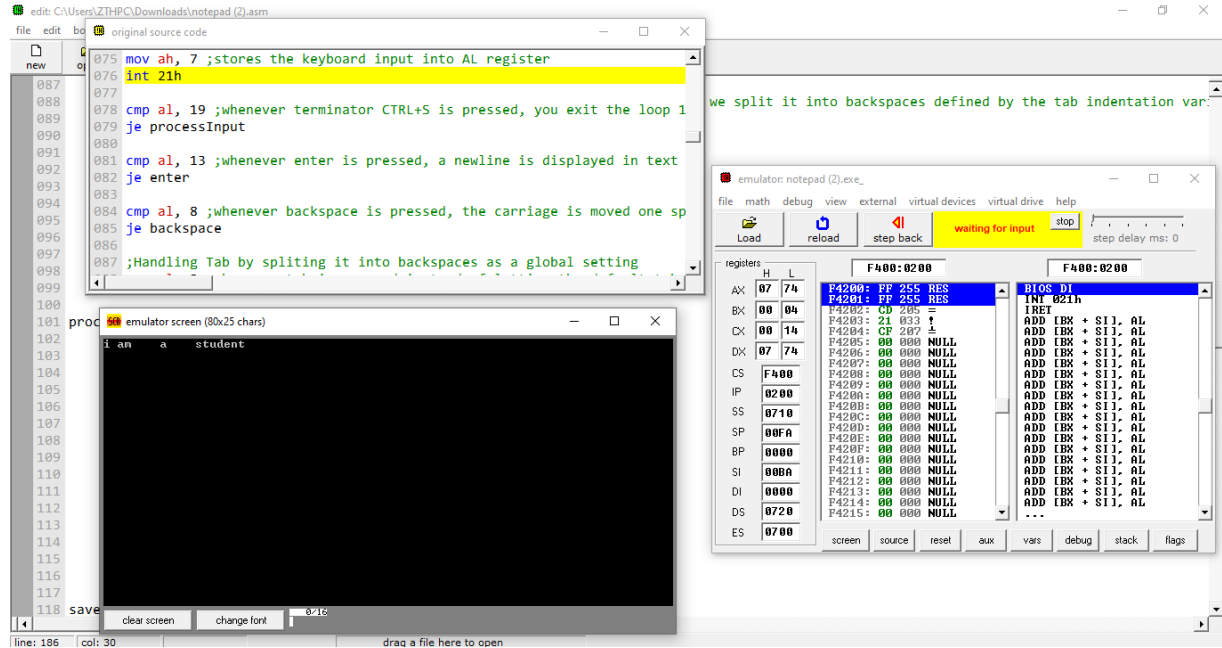
file math debug view external virtual devices virtual drive help

Load reload step back waiting for input stop step delay ms: 0

registers	H	L	F400:0200	F400:0200
AX	01	24	F4200: FF 255 RES	INT 021h
BX	00	04	F4201: FF 255 RES	1RET
CX	00	11	F4202: CD 205 =	ADD [BX + SI], AL
DX	00	33	F4203: 21 033 ?	ADD [BX + SI], AL
CS	F400		F4204: CF 207 1	ADD [BX + SI], AL
IP	0200		F4205: 00 000 NULL	ADD [BX + SI], AL
SS	0710		F4206: 00 000 NULL	ADD [BX + SI], AL
SP	00F8		F4207: 00 000 NULL	ADD [BX + SI], AL
BP	0000		F4208: 00 000 NULL	ADD [BX + SI], AL
SI	0007		F4209: 00 000 NULL	ADD [BX + SI], AL
DI	0000		F420A: 00 000 NULL	ADD [BX + SI], AL
DS	0720		F420B: 00 000 NULL	ADD [BX + SI], AL
ES	0700		F420C: 00 000 NULL	ADD [BX + SI], AL
			F420D: 00 000 NULL	ADD [BX + SI], AL
			F420E: 00 000 NULL	ADD [BX + SI], AL
			F420F: 00 000 NULL	ADD [BX + SI], AL
			F4210: 00 000 NULL	ADD [BX + SI], AL
			F4211: 00 000 NULL	ADD [BX + SI], AL
			F4212: 00 000 NULL	ADD [BX + SI], AL
			F4213: 00 000 NULL	ADD [BX + SI], AL
			F4214: 00 000 NULL	ADD [BX + SI], AL
			F4215: 00 000 NULL	ADD [BX + SI], AL

screen source reset aux vars debug stack flags

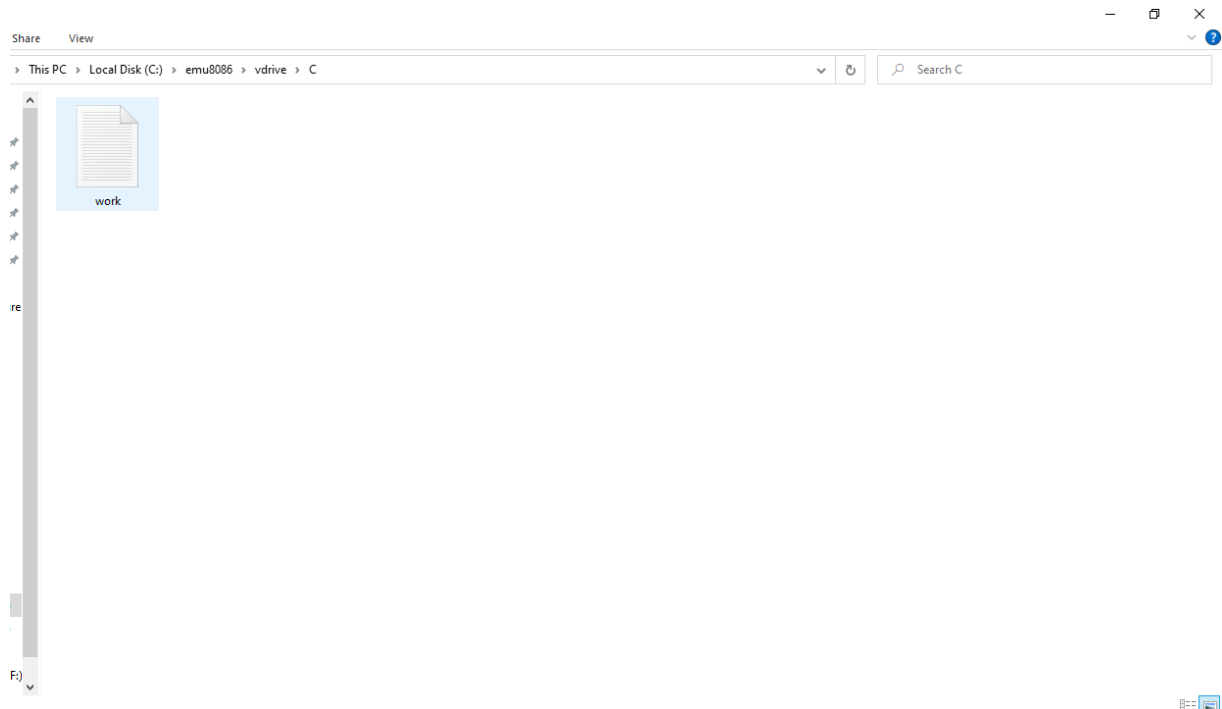
Tab pressed



The screenshot shows an x86-64 emulator with the following components:

- Assembly Code Editor:** Displays assembly code for a program that handles keyboard input. The code includes comments explaining the logic for handling the Tab key by splitting it into backspaces.
- Emulator Screen:** A terminal window showing the output of the program, which displays the text "i an a student".
- Registers Window:** Shows the state of various registers, including AX, BX, CX, DX, IP, CS, BP, SI, DI, DS, and ES.
- Memory Window:** Displays memory addresses and their contents, including BIOS data.

File saved in folder



The screenshot shows a Windows File Explorer window with the following details:

- Address Bar:** Shows the path "This PC > Local Disk (C:) > emu8086 > vdrive > C".
- Search Bar:** Contains the text "Search C".
- File List:** Displays a single file named "work" with a document icon.