República Bolivariana de Venezuela
Universidad Nacional Experimental Politécnica
"Antonio José de Sucre"
Vice Rectorado Barquisimeto
Departamento de Ingeniería Electrónica

Practica 5
laboratorio de diseño de sistemas de computación

Integrantes:
Gerardo Alfonzo Campos Fonseca
V. 27085179
José Andrés Cortez Teran
V. 26540824

Barquisimeto, Agosto del 2021

Índice

Índice de figuras

Programa lectura.asm

El código del programa se encuentra en un único archivo llamado **lectura.asm**. Este programa funciona mediante una aplicación de consola usando la api de 32 bits de Windows. Para la lectura del archivo se usan las funciones CreateFile en modo lectura, GetFileSize y ReadFile el tamaño. GetFileSize se utiliza para crear un bloque de memoria lo suficientemente grande para almacenar el archivo en memoria, mas el nuevo registro. Para escribir el archivo una vez modificado se utilizan lstrlen para poder saber la cantidad de bytes a escribir y finalmente se usa WriteFile.

```
.386
.MODEL flat, stdcall
OPTION CASEMAP:NONE

Include windows.inc
Include kernel32.inc
Include masm32.inc
include  user32.inc

IncludeLib kernel32.lib
IncludeLib masm32.lib
includelib user32.lib

; lectura de archivo
; calculos respectivos al archivo
; escritura de archivo stdout

; pedir informacion adicional

; re estructurar todo
; imprimir nuevo archivo

;escribir y guardar en el nuevo archivo


Main PROTO

Print_Text Macro txt:REQ
  Invoke StdOut,ADDR txt
EndM

Get_Input Macro prompt:REQ,buffer:REQ
  Invoke StdOut,ADDR prompt
  Invoke StdIn,ADDR buffer, LengthOf buffer
EndM

.DATA
Msg1   DB "Please Type the file is name or path: ",0AH,0DH,0
Msg4   DB "Press Enter to Exit",0AH,0DH,0
```

```
CRLF    DB 0DH,0AH,0
MsgPos DB "Please type the position where the new register will be inserted. ",0AH,0DH,0
Msgaux DB "should be almost 1 and less than: ",0
MsgNom DB "Please type the name: ",0AH,0DH,0
MsgCed DB "Please type the ID: ",0AH,0DH,0
MsgN1  DB "Please type first grade: ",0AH,0DH,0
MsgN2  DB "Please type second grade: ",0AH,0DH,0
MsgN3  DB "Please type third grade: ",0AH,0DH,0
coma    DB ",",0

Cant_lineas DB 1
Aux_string  DB 100 dup(0) ;para el nuevo campo de la bbdd
new_file    DB "BBDD.txt",0

.DATA?
inbuf DB 100 DUP (?)
hFile       DD ?
hFileWrite  DD ?
FileSize    DD ?
hMem        DD ?
BytesRead   DD ?
pos     DB 10 DUP(?)
ID      DB 23 DUP(?)
nam       DB 80 DUP(?)
N1      DB 10 DUP(?)
N2        DB 10 DUP(?)
N3      DB 10 DUP(?)

bytewr      DD ?          ;variable adicional creada por necesidad para el proc
BufferSize  DD ?
Buffer_div_size DD ?
hMem_div    DD ?

.CODE
Start:

  ;*************************************************************************
    ;                           handling the files                          *
  ;*************************************************************************

    ; file to read
  Get_Input Msg1, inbuf
  ;se usa la api de windows para abrir la fila
  invoke  CreateFile,ADDR inbuf,GENERIC_READ,0,0,\
  OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0
  mov     hFile,eax

  ;obtencion del  tamao de la fila para pedir memoria dinamica
  invoke  GetFileSize,eax,0
```

```
89  mov     FileSize,eax
90  inc     eax
91  ;pedir memoria dinamica
92  invoke  GlobalAlloc,GMEM_FIXED,eax
93  mov     hMem,eax
94  add     eax,FileSize
95  mov     BYTE PTR [eax],0   ; Set the last byte to NULL so that StdOut
96  ; can safely display the text in memory.
97  ;finalmente se lee la fila
98  invoke  ReadFile,hFile,hMem,FileSize,ADDR BytesRead,0
99
100 ;se escribe el fichero
101 invoke  StdOut,hMem
102 Print_Text CRLF ;salto de linea
103 Print_Text CRLF ;salto de linea
104 invoke  CloseHandle,hFile
105
106 ;*************************************************************************
107   ;*                          file to write                                *
108 ;*************************************************************************
109
110 ;ADDR inbuf
111 invoke  CreateFile,ADDR inbuf ,GENERIC_WRITE,0,0,\
112         CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,0
113 mov     hFileWrite,eax
114   ;invoke CreateFile,lpName,GENERIC_WRITE,NULL,NULL,CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL
115
116
117 ;cantidad de bytes se consigue restando 2 direcciones de memoria xD
118
119
120 ;*************************************************************************
121   ;                          calculo de lineas                            *
122 ;*************************************************************************
123
124
125 mov ecx, 1 ;inicializamos el contador en 1 posible linea
126 mov esi,  hMem ; vamos a trabajar con la fila, por eso usamos el handler de memoria...
127 mov eax, 0 ; limpiamos eax, solo usaremos al (byte...)
128 cont_lineas:
129   mov al, [esi]
130
131   cmp eax, 10 ;compara buscando caracter de salto de linea '\r' '\n' con \n = 10 decimal A
    hex, gracias olly
132   jne n_linea_nueva
133   inc ecx
134
135   n_linea_nueva:
136
```

```
137   cmp eax , 0 ; compara con el caracter de fin de archivo, end buffer...
138 je  f_lineas
139
140   inc esi
141
142 jmp cont_lineas
143
144 f_lineas:
145
146 mov  edi, OFFSET Cant_lineas; guardamos la cantidad de lineas que hay! primero la memoria a
      un registro
147 inc ecx ; cantidad de lineas +1
148 push ecx ;guardamos el valor numerico
149
150 xor ecx, 30H; mascara para llevarlo a ascii
151 mov [edi] , ecx ; luego el valor a la direccion
152
153 ;*************************************************************************
154   ;*                          lectura de valores                          *
155 ;*************************************************************************
156
157 ;*************************************************************************
158   ;*                          verificacion de posicion                    *
159 ;*************************************************************************
160
161 verificacion:
162   Print_Text MsgPos
163   Print_Text Msgaux
164   Print_Text Cant_lineas
165   Get_Input CRLF,  pos; pedir la posicion
166
167   mov  edi, OFFSET pos
168   xor eax,eax ; limpiamos eax
169   mov eax, [edi]
170   and eax, 15 ;llevamos eax a numerico
171
172   cmp eax,0
173 jg  segundo
174 jmp verificacion
175
176 segundo:
177   pop ecx ;recuperamos el valor numerico
178   push ecx ;lo volvemos a guardar para uso posterior al revisar
179   cmp ecx,eax
180 jl verificacion
181
182 push eax; Guardando el valor en el que se desa ingresar para recuperarlo
183 ;mas tarde facilmente
184
```

```
185 ;*************************************************************************
186   ;*                      Pedir el resto de los datos                      *
187 ;*************************************************************************
188
189 Get_Input MsgCed, ID; pedir la cedula
190 Get_Input MsgNom, nam; pedir el nombre
191 Get_Input MsgN1, N1; pedir la nota 1
192 Get_Input MsgN2, N2; pedir la nota 2
193 Get_Input MsgN3, N3; pedir la nota 3
194
195
196 ;*************************************************************************
197   ;*                       Creacion de la estring                          *
198 ;*************************************************************************
199
200 invoke lstrcat,offset Aux_string,OFFSET ID
201 invoke lstrcat,offset Aux_string,OFFSET coma
202 invoke lstrcat,offset Aux_string,OFFSET nam
203 invoke lstrcat,offset Aux_string,OFFSET coma
204 invoke lstrcat,offset Aux_string,OFFSET N1
205 invoke lstrcat,offset Aux_string,OFFSET coma
206 invoke lstrcat,offset Aux_string,OFFSET N2
207 invoke lstrcat,offset Aux_string,OFFSET coma
208 invoke lstrcat,offset Aux_string,OFFSET N3
209
210 Invoke lstrlen, offset Aux_string
211 mov BufferSize,eax
212
213
214 ;*************************************************************************
215   ;                       Escritura en el archivo                         *
216 ;*************************************************************************
217
218 pop eax ; recuperamos el valo de la linea donde vamos a insertar
219 pop ecx ;recuperamos el valor de la cantidad de lineas que hay
220
221 cmp eax,1
222 je first_line
223 cmp ecx, eax
224 je last_line
225 jmp in_line
226
227 first_line:
228
229   invoke WriteFile,hFileWrite,offset Aux_string,BufferSize,ADDR bytewr,NULL;escritura de la
    cadena
230   Invoke WriteFile,hFileWrite,offset CRLF,2,ADDR bytewr,NULL;escritura del salto de linea
231   invoke WriteFile,hFileWrite,hMem,FileSize,ADDR bytewr,NULL   ;escritura del archivo
232   jmp fin;listo
```

```
233
234 last_line:
235
236   invoke WriteFile,hFileWrite,hMem,FileSize,ADDR bytewr,NULL;escritura del archivo
237   Invoke WriteFile,hFileWrite,offset CRLF,2,ADDR bytewr,NULL;escritura del salto de linea
238   invoke WriteFile,hFileWrite,offset Aux_string,BufferSize,ADDR bytewr,NULL ; escritura del
      nuevo registro
239   jmp fin;listo
240
241
242 in_line:
243 ;eax contiene el lugar en el que se va a guardar, usamos un respaldo en edi
244 mov edi,eax
245
246 mov ecx, 1 ;inicializamos el contador en 1 posible linea
247 mov esi,  hMem ; vamos a trabajar con el archivo, por eso usamos el handler de memoria...
248 mov eax, 0 ; limpiamos eax, solo usaremos al (byte...)
249 cont_lineas2:
250   mov al, [esi]
251
252   cmp eax, 10 ;compara buscando caracter de salto de linea '\r' '\n' con \n = 10 decimal A
      hex
253   jne n_linea_nueva2
254   ;Se llego al punto en el que se escribiria la nueva linea
255   inc ecx
256   cmp edi, ecx
257   je escritura
258
259   n_linea_nueva2:
260   cmp eax , 0 ; compara con el caracter de fin de archivo, end buffer...
261 je  f_lineas2
262
263   inc esi
264 jmp cont_lineas2
265
266 f_lineas2:
267   jmp fin
268
269 escritura:
270   ; eax tiene un caracter
271   ;esi direccion del hmem+cant caracteres     sirve
272   ; edi tiene el numero de linea a donde va
273   ;ecx tiene el contador de cuantos lineas  van
274   inc  esi
275   push esi ; guardamos la direccion de hmem+cant_caracteres
276
277   mov ecx, hMem; direccion inicial del archivo
278   sub esi, ecx ;en esi se tiene cuantos caracteres hay
279
```

```
280     mov edi, offset Buffer_div_size
281     mov [edi] , esi ; luego el valor a la direccion
282
283     invoke WriteFile,hFileWrite,hMem,Buffer_div_size,ADDR bytewr,NULL;escritura del archivo
        primera parte
284
285
286     invoke WriteFile,hFileWrite,offset Aux_string,BufferSize,ADDR bytewr,NULL ; escritura del
        nuevo registro
287     Invoke WriteFile,hFileWrite,offset CRLF,2,ADDR bytewr,NULL;escritura del salto de linea
288
289     pop eax ; guardamos en eax la direccion de donde me quede en el archivo
290     mov hMem_div,eax
291     mov ecx, FileSize
292     sub ecx, esi ; en ecx quedan cuantos caracteres faltan
293     mov [edi] , ecx ; luego el valor a la direccion
294     invoke WriteFile,hFileWrite,hMem_div,Buffer_div_size,ADDR bytewr,NULL;escritura del
        archivo
295
296   fin:
297
298   ;*************************************************************************
299     ;                               Fin Programa                            *
300   ;*************************************************************************
301
302   ;se cierran los ficheros
303   invoke  CloseHandle,hFileWrite
304
305   ;se libera la memoria dinamica
306   invoke  GlobalFree,hMem
307   ;espera enter para salir y poder leer
308
309   Print_Text CRLF ;salto de linea
310   Get_Input Msg4,inbuf ;mensaje de salida
311
312   ;sale del programa
313   Invoke ExitProcess,0
314
315 End Start
```

Funcionamiento del programa

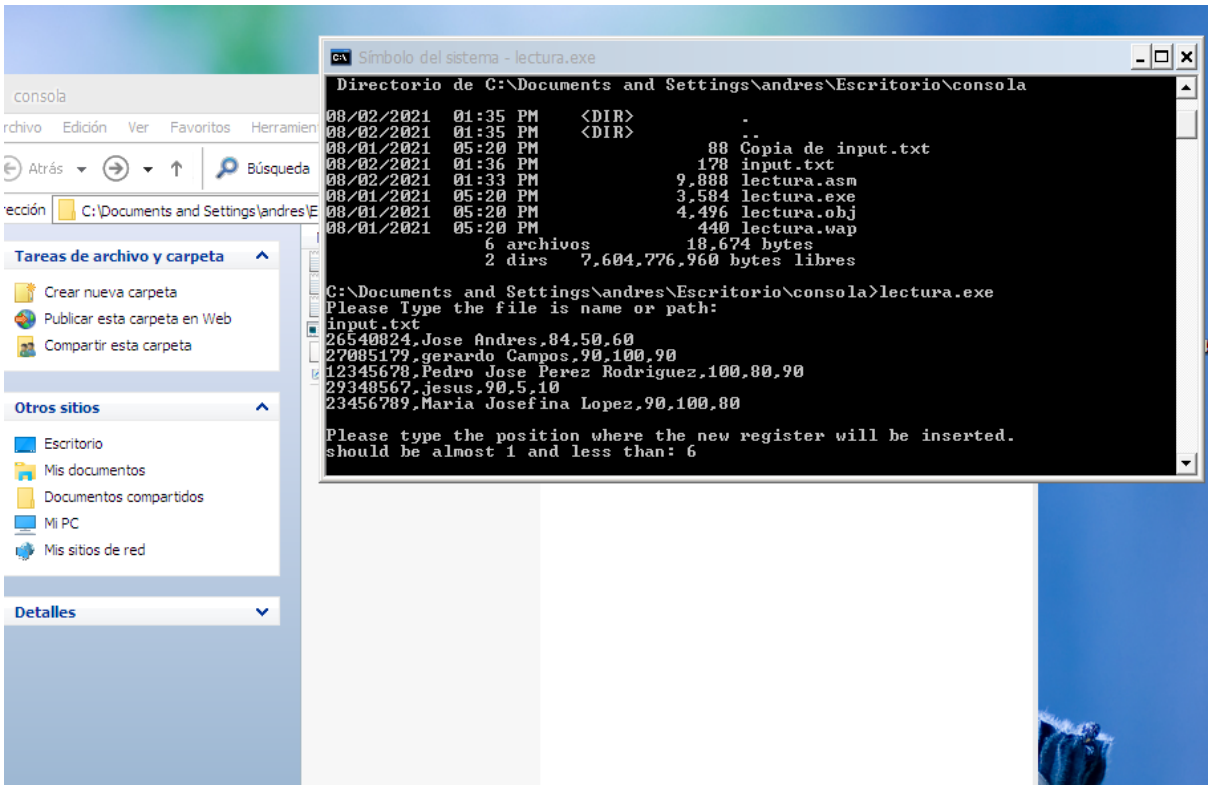

Figura 1: Abriendo el archivo

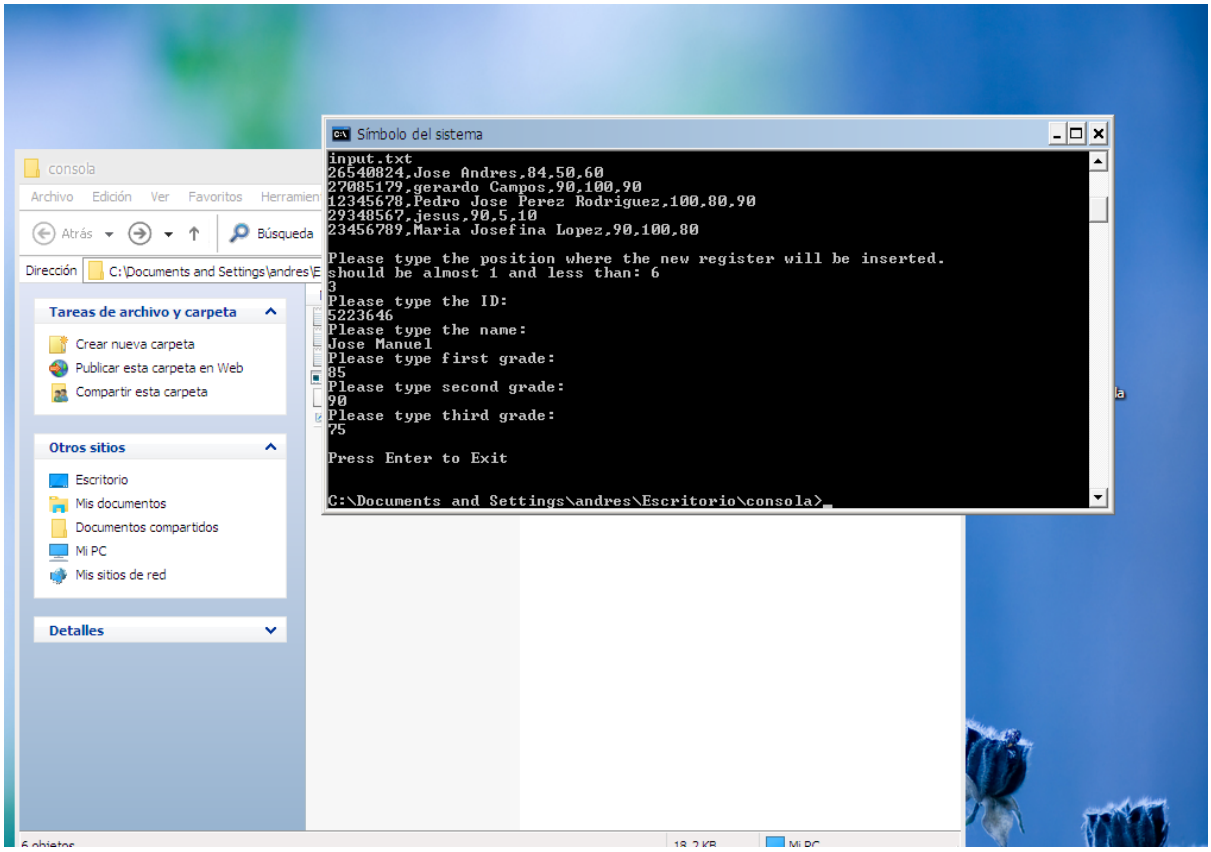

Figura 2: Agregando un nuevo registro

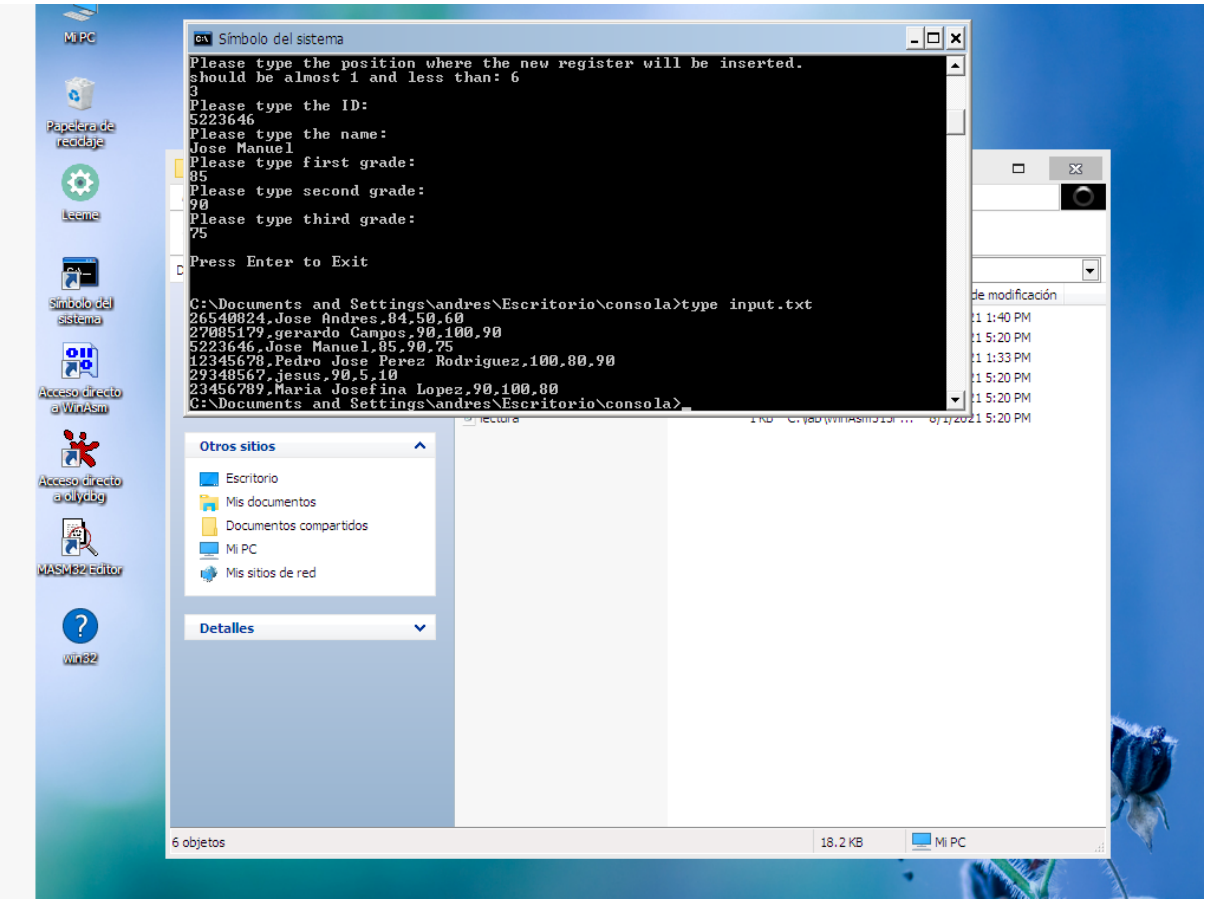

Figura 3: Mostrando en pantalla el archivo resultante

Referencias Bibliográficas

Iczelion. (s.f.). http://www.movsd.com/icz.htm

Irvine, K. R. (2002). Assembly Language for Intel Assembly Language for Intel-Based Computers, 4 Computers, 4th Edition Edition [fecha de consulta: 19/7/2021]. https://www.csie.ntu.edu.tw/~acpang/course/asm_2004/slides/chapt_08Solve.pdf

Microsoft. (2017). CreateFileA function (fileapi.h) [fecha de consulta: 22/07/2021]. https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfilea

Microsoft. (2021a). CloseHandle function (handleapi.h) [fecha de consulta: 22/7/2021]. https://docs.microsoft.com/en-us/windows/win32/api/handleapi/nf-handleapi-closehandle

Microsoft. (2021b). GetDlgItemTextA function (winuser.h) [fecha de consulta: 24/7/2021]. https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getdlgitemtexta

Microsoft. (2021c). GetFileSize function (fileapi.h) [fecha de consulta: 22/7/2021]. https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-getfilesize

Microsoft. (2021d). GlobalAlloc function (winbase.h) [fecha de consulta: 22/7/2021]. https://docs.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-globalalloc

Microsoft. (2021e). How to get the string value of editbox in the variable in win32 application? [fecha de consulta: 24/7/2021]. https://social.msdn.microsoft.com/Forums/vstudio/en-US/17c2d97a-011b-4fb1-9563-4f095d9321e4/how-to-get-the-string-value-of-editbox-in-the-variable-in-win32-application?forum=vcgeneral

Microsoft. (2021f). ReadFile function (fileapi.h) [fecha de consulta: 22/7/2021]. https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-readfile

OVERFLOW, S. (2015). Getting string input and displaying input with DOS interrupts MASM [fecha de consulta: 19/7/2021]. https://stackoverflow.com/questions/29504516/getting-string-input-and-displaying-input-with-dos-interrupts-masm

to 2012, T. M. F. A. 2. (s.f.). how to read from file in masm [fecha de consulta: 22/7/2021]. http://www.masmforum.com/board/index.php?topic=16266.0