



**UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA**  
**Ingeniería en Computación**  
**Lenguaje ensamblador**

**Documentación.**  
**Editor hexadecimal 'Hexed'**

**Profesor:**  
M.C Luis Anselmo Zarza López

**Grupo:**  
602-A

**Realizado por:**  
Hernández Montellano Carlos

3 de julio de 2017

# Índice

1 Requisitos.....	1
1.1 Requisitos para ejecución.....	1
1.2 Requisitos para compilación.....	1
2 Compilación.....	1
3 Ejecución.....	1
4 Funcionalidad, características y limitaciones del programa.....	2
5 Código fuente.....	3
5.1 Menú.....	3
5.2 Modificación de buffer.....	4
5.3 Modificación de buffer.....	4
5.4 Desplegado de los datos hexadecimales.....	5
5.5 Desplegado de los datos hexadecimales.....	6

# 1 Requisitos

## 1.1 Requisitos para ejecución

- *Sistema Operativo:* DOS ~ Windows XP.

## 1.2 Requisitos para compilación

- *Sistema Operativo:* DOS ~ Windows XP.
- Turbo Assembler (TASM)

# 2 Compilación

A continuación se mostrará el procedimiento para compilar el código fuente con TASM. Es necesario tener las variables de entorno adecuadamente configuradas para que el shell reconozca los comandos de ejecución de TASM.

1. Establecer la ruta del shell donde se encuentran los códigos fuente.
2. Generar los códigos objeto de *HEXED.ASM* y *FUN.ASM*, para esto ejecutar las siguientes instrucciones.

***tasm fun.asm***

***tasm fun.asm***

3. Ligar los códigos objeto, para esto ejecutar la siguiente instrucción.

***tlink hexed fun***

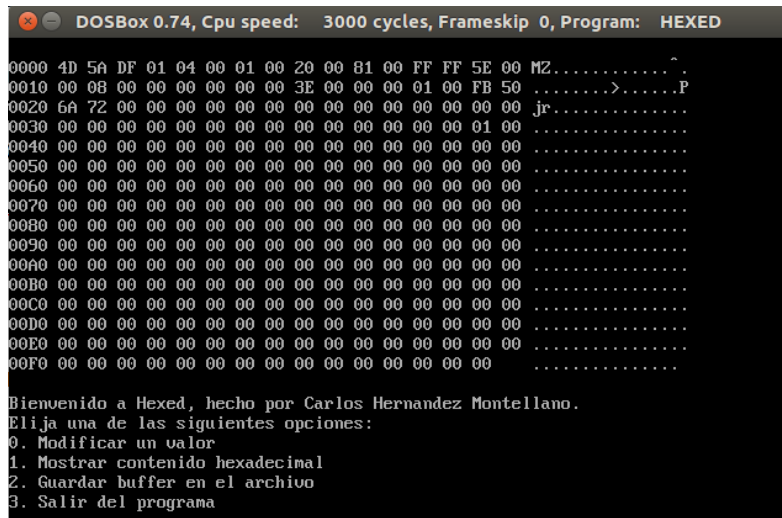
# 3 Ejecución

Para poder ejecutar el programa correctamente, es necesario que en el mismo directorio se encuentre el archivo "*archivo.txt*" del cual es de donde se obtendrá la información hexadecimal, de lo contrario el programa terminará en error. Se puede renombrar un archivo binario o cualquier otro archivo para poder ser abierto con el programa. Para ejecutar el programa utilizar la siguiente instrucción con la ruta apropiada en el shell.

***hexed.exe***

## 4 Funcionalidad, características y limitaciones del programa

- Puede abrir archivos de hasta **8 bits**. Si se intenta abrir un archivo más grande, el programa mostrará sólo las primeras 255 posiciones.



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: HEXED

0000 4D 5A DF 01 04 00 01 00 20 00 81 00 FF FF 5E 00 M2.....^
0010 00 08 00 00 00 00 00 00 3E 00 00 00 01 00 FB 50 .....>.....P
0020 5A 72 00 00 00 00 00 00 00 00 00 00 00 00 00 jr.....
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 .....
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

Bienvenido a Hexed, hecho por Carlos Hernandez Montellano.
Elija una de las siguientes opciones:
0. Modificar un valor
1. Mostrar contenido hexadecimal
2. Guardar buffer en el archivo
3. Salir del programa
```

*Figura 1: Caso en el que se intentó abrir el ejecutable de Hexed. Se muestra la última posición FEh como 00.*

- Se pueden hacer modificaciones de los valores mediante la opción '0' del menú. Se preguntará al usuario la posición en hexadecimal del dato que se quiere modificar y el nuevo valor.
- Se puede volver a mostrar la tabla de datos en hexadecimal eligiendo la opción '1' del menú.
- Se pueden guardar los primeros 255 bits de un archivo sin alterar los siguientes bits restantes del archivo. Para esto se puede elegir la opción '2' del menú.

## 5 Código fuente

A continuación se muestran fragmentos del código fuente que representan la implementación de algunas de las funcionalidades.

### 5.1 Menú

La sección del menú despliega todas las opciones contenidas en una sola cadena, después se lee la opción que se eligió y posteriormente se compara con las distintas opciones para saltar a la parte del código correspondiente a la opción elegida.

- El menú se despliega después de haber mostrado la información hexadecimal por primera vez sin la solicitud del usuario.
- Si el usuario introduce cualquier otro valor que no esté en el menú, este volverá a desplegarse.

```
98      ;Aqui empieza interaccion con el usuario
99 u_menu: call reto
100      call reto
101      mov ah, 09h
102      mov dx, offset cad4
103      int 21h
104      call reto
105      call lee1
106      call reto
107      cmp al, 0
108      jne u_nomod
109      call modify
110 u_nomod: cmp al, 1
111          je u_des
112          cmp al, 2
113          jne u_nosav
114          call save
115          call save
116          mov ah, 09
117          mov dx, offset cad5
118          int 21h
119 u_nosav: cmp al, 3
120          je exit
121          jmp u_menu ;Si no se elijio salir del programa, se vuelve a mostrar el menu
122      ;Cerrar archivo
```

Figura 2: Fragmento del código donde se implementó el menú.

## 5.2 Modificación de buffer

La forma de modificar los datos del archivo es pidiéndole al usuario que ingrese la posición del dato que quiere modificar y el nuevo valor que desee.

Hay que tomar en cuenta que posiciones no existentes y valores no hexadecimales no están validados y pueden corromper la ejecución del programa o el archivo si se guarda una entrada corrupta.

Esta función es llamada directamente desde el menú.

```
170 ;Funcion para modificar un valor del buffer
171 modify: pusha
172         mov ah, 09h ;Se le pide al usuario la posición
173         mov dx, offset cad1
174         int 21h
175         call lee4 ;Se lee la posición que ingresó el usuario
176         mov [position], ax
177         mov ah, 02
178         mov dl, 0Ah
179         int 21h
180         mov dl, 0Dh
181         int 21h
182         mov bx, offset buffer
183         add bx, [position]
184         mov [position], bx ;Se guarda la posición absoluta en memoria
185         mov dx, offset cad2
186         mov ah, 09
187         int 21h
188         mov dx, [bx]
189         call des2 ; Se le notifica al usuario el valor de esa posición
190         call reto
191         mov dx, offset cad3 ; Se le pide al usuario que ingrese el nuevo valor
192         int 21h
193         call lee2
194         cld
195         mov di, [position]
196         stosb ;Se copia el nuevo valor dentro del bufer. No se guarda en archivo
197         mov ah, 02
198         mov dl, 0Ah
199         int 21h
200         mov dl, 0Dh
201         int 21h
202         popa
203         ret
```

Figura 3: Se muestra el fragmento de código encargado de modificar el buffer de datos. Este código no guarda el buffer en el archivo una vez ha sido modificado.

## 5.3 Guardar buffer en el archivo

Esta función se encarga de guardar el buffer en el archivo. Se llama desde el menú.

```
204 ;Funcion que se encarga de guardar el buffer en el archivo.
205 save: pusha
206         mov ah, 42h ;Con este servicio se retorna al apuntador principio del
207         mov al, 0h ;archivo para que el apuntador sobrescriba el texto
208         mov bx, fid
209         mov cx, 0h
210         mov dx, 0h
211         int 21h
212         mov ah, 40h ;Se utiliza el archivo de escritura
213         mov bx, fid
214         mov cx, [sizer] ;Se indica el tamaño del buffer
215         mov dx, offset buffer ;Se indica la dirección del buffer
216         int 21h
217         jc error ;Si hay error, se despliega mensaje de error y aborta el programa
218         popa
219         ret
```

Figura 4: Fragmento de código que guarda el buffer en el archivo.

## 5.4 Desplegado de los datos hexadecimales

La forma de desplegar el contenido hexadecimal consiste en ir recorriendo el buffer byte a byte e ir desplegando el valor hexadecimal de cada uno. Cada renglón está constituido de la representación de 16 bytes. La representación ASCII es llamada por otra función en cada renglón impreso.

;Después de abrir y leer el archivo, se empieza el procedimiento para desplegar los datos hex.

mov [sizer], ax ;Guardar bytes leídos [0 - FFh]

u\_des: mov cx, [sizer] ;Recuperar bytes leídos desde variable. Cuando el usuario

      ;decide desplegar, empieza desde aquí

      ;Se imprime cabecera

      mov ah, 09

      mov dx, offset cabecera

      int 21h

      call reto

      mov bl, 0 ;Contador de renglones

      mov bh, 0 ;Contador de índice

      mov dx, 0

      call des4

      call spc

      cld

      mov si, offset buffer

h\_des1: lodsb

      mov dl, al

      call des2

      mov dl, ''

      mov ah, 02

      int 21h

      inc bl

      inc bh

      cmp bl, 16

      je h\_resbl

h\_rloop: loop h\_des1

      cmp bl, 0

      je exit

      ;Última impresión ASCII, se dan los espacios necesarios para que este se despliegue

correctamente

      mov ch, 0

      mov bh, 16

      sub bh, bl

      mov al, bh

      mov bh, 03

      mul bh

      mov cl, al

h\_spclp: call spc

      loop h\_spclp

      mov dx, si

      mov bh, 0

      sub dx, bx

      call ascii

## 5.5 Desplegado de los datos en ASCII

En esta sección del código se recibe en DX la posición donde se quedó SI y se le restan 16 posiciones. Después utilizando LODSB se cargan las últimas 16 posiciones del buffer y son desplegadas con el servicio 02H para AH.

Esta función sólo es llamada desde el procedimiento de mostrar los datos hexadecimales del buffer.

```
148      ;Despliega caracteres ASCII o si no un '.', Rango [21h ~ 7Eh]
149      ;Recibe dirección inicial en DX
150 ascii: pusha
151      mov ch, 0
152      mov cl, bl
153      mov si, dx
154      cld
155      mov ah, 02
156 ascloop: lodsb
157      cmp al, 21h
158      jl aprintp
159      cmp al, 7eh
160      jg aprintp
161      mov dl, al
162      int 21h
163      jmp asiglp
164 aprintp: mov dl, '.'
165      int 21h
166 asiglp: loop ascloop
167 ascex: popa
168      ret
```

Figura 5: Fragmento para desplegar caracteres ASCII de la últimas 16 posiciones del buffer.