

**UNIVERSIDAD DE GUADALAJARA**  
**CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIA**  
**DIVISION DE ELECTRONICA Y COMPUTACION**

**DEPARTAMENTO DE CIENCIAS COMPUTACIONALES**

**SEMINARIO DE SOLUCIÓN DE  
PROBLEMAS DE TRADUCTORES DE  
LENGUAJE I (I7026)**

**Reporte de Práctica 7**

[Practica 7: " Manejo de archivos"](#)

**Alumno: Oswaldo Luna Grados**  
**Código: 211718256**

Sección: D07

Fecha 7 de noviembre del 2018  
firma de revisado

Profesor: Valentín Martínez López

Objetivo: Escribir un programa que sea capaz de leer un archivo de texto del que debe determinar el número de caracteres de cada línea, el programa debe crear un archivo nuevo en el que al inicio de cada línea indique el número de caracteres que existen hasta el momento.

Puede usarse interrupción 21 o funciones de librerías.

Suponga que el archivo de texto contiene lo siguiente:

hola que tal  
como has estado  
esto es una prueba  
para contar caracteres.  
Se deberá crear un archivo de texto como el que sigue:

000 hola que tal  
012 como has estado  
027 esto es una prueba  
045 para contar caracteres.  
068

En la primera línea del archivo de salida se imprime al principio 000 por que no se han contado ningún carácter.

La primera línea tiene 12 caracteres por lo que en archivo de salida la segunda línea empieza con 012

La segunda línea tiene 15 caracteres al sumarse con los caracteres de la primera línea el resultado es 27 y se imprimen en la tercera línea y esto se repite hasta obtener la cantidad total de caracteres del archivo.

Un archivo informático está identificado por un nombre y una descripción, el cual almacena información en formato binario (es decir ceros y unos). En lenguajes de alto nivel manejan los grupos de información (archivos), escondiendo la complejidad sobre el manejo y compilación de los mismos. En lenguajes de alto nivel la manipulación de archivos se reduce a tareas simples como por ejemplo, creación, lectura, escritura. En lenguaje ensamblador, la manipulación de archivo requiere de mayor detalle.

Lo primero es leer el archivo

El cual se llama file.txt y lo máximo de caracteres que lea el archivo es 110 y lo guardara en la cadena llamada leído y cerramos el archivo

```

read:
mov dx, offset filename
mov ah, 3dh
int 21h
jc error
mov handle,ax
mov bx,handle
mov cx, 110 ;maximo de caracteres en el archivo
mov dx, offset leido
mov ah, 3fh
int 21h

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

```

Hice que imprimiera en pantalla la cadena leída y guardada en leído.

```

print:
;imprimir el contenido de leido
mov dx, offset leido
mov ah, 9
int 21h

```

Después empezamos el contar los caracteres por renglón inicializando los registros en cero luego ponemos 0 puesto que no se ha leído nada aun.

```

cont:
xor dx,dx ;contar caracteres
xor ax,ax
xor bx,bx
xor cx,cx
mov cx, 110 ;maximo de caracteres a leer
xor si,si
xor di,di

mov string[di], '0'
inc ap1
mov di,ap1
mov string[di], ' '
inc ap1

```

En recorrer lo que hace es poner en los registros di y si la posición actual de las cadenas leídas y escrita con el número de caracteres. También revisa si hay un salto de carácter o final de la cadena leída si no es ninguna de las otras va a contcarac donde se pone en contChar el número en hexadecimal de caracteres contados actualmente. Y vuelve a seguir donde se pasa el carácter actual y después incrementa la posición de cada cadena y vuelve a recorrer

```

recorre:
mov si,ap
mov di,ap1
cmp leido[si], '$'
jz fin
cmp leido[si], 10
jz suma
jnz contcarac
seguir:
mov al,leido[si]
mov string[di],al
seguir2:
inc ap
inc ap1
loop recorre

```

```

contcarac:
mov bx,ap
inc contChar
call seguir

```

Si encuentra salto de línea va a suma donde pone el valor actual de caracteres contados en ascii en la cadena escrita.

```

suma:
mov di,ap1
mov string[di],10
inc ap1
call hex2ascii ;agrega la cantidad de caracteres
mov di,ap1
mov string[di], ' '
call seguir2

```

Hex2ascii hace que devuelva el valor en decimal.

```

hex2ascii proc
xor ax,ax
mov al, contChar
mov bx, 0ah
div bl
mov bl, al
add al, 48
mov di,ap1
mov string[di],al
inc ap1
xor ax,ax
mov al, bl
mov bl, 0ah
mul bl
mov bx, ax
mov al, contChar
sub ax, bx
add al, 48
mov di,ap1
mov string[di],al
inc ap1
ret
hex2ascii endp

```

En cuanto encuentre el final de la cadena leída guardara en la cadena escrita el número de caracteres después de dar un salto de renglón.

```
fin:
; se sumara el total y terminara
mov string[di],13
inc ap1
mov di,ap1
mov string[di],10
inc ap1
mov di,ap1
call hex2ascii
mov di,ap1
```

Ya que tenemos la cadena escrita con el número de caracteres por renglón creamos un nuevo archivo llamado file2.txt  
Imprimimos la nueva cadena en pantalla, escribimos en el archivo la cadena escrita y cerramos el archivo.

```
; create file
create:
    mov ah,3ch
    mov cx,00000000b
    lea dx,filename2
    int 21h
    jc error
    mov handle,ax

printf:
xor dx,dx ;print
mov dx, offset string
mov ah, 9
int 21h

write:
mov ah,40h
mov bx,handle
mov cx,110
lea dx,string
int 21h

; close file
mov ah,3eh
mov bx,handle
int 21h
error:
int 20h
```

The screenshot displays three overlapping windows on a Windows desktop:

- original source code**: A text editor showing assembly code. The code includes instructions like `mov si, ap`, `mov di, ap1`, `cmp leido[si], '$'`, `jz fin`, `cmp leido[si], 10`, `jz suma`, `jnz contcarac`, `seguir:`, `mov al, leido[si]`, `mov string[di], al`, `seguir2:`, `inc ap`, `inc ap1`, and `loop rep`. The line `jz suma` is highlighted in yellow.
- emulador: practica7\_OswaldoLuna.com**: A debugger window. It features a menu bar (file, math, debug, view, external, virtual devices, virtual drive, help), a toolbar (Load, reload, step back, single step, run), and a status bar (step delay ms: 0). The registers window shows:
 

	H	L
AX	00	73
BX	00	01
CX	00	6C
DX	00	00
ES	0700	0262

 The memory window shows addresses 0700:0262 to 0700:026F with values like 74, 116, 57, 087, 75, 117, 12, 018, 8A, 138, 84, 132, 82, 088. The disassembly window shows instructions like `MOV SI, [001F7h]`, `MOV DI, [001F9h]`, `CMP b.[SI] + 00116h, 024h`, `JZ 02D7h`, `CMP b.[SI] + 00116h, 0Ah`, and `JZ 02BBh`.
- emulator screen (80x25 chars)**: A terminal window displaying the output of the program. The text shown is:
 

```

esta es una prueba
para contar el numero de caracteres
de un archivo de texto
¿isuerte!

printf:
xor dx, dx
mov dx, 0
mov ah, 9
int 21h

write:
mov ah
mov bx
mov cx
lea dx
int 21h

mov ah
mov bx
mov cx
lea dx
int 21h
      
```

The screenshot shows a Windows desktop with several open applications. The primary application is a debugger named 'emulator: practica7\_OswaldoLuna.com\_'. The debugger's interface includes a menu bar (file, math, debug, view, external, virtual devices, virtual drive, help), a toolbar (Load, reload, step back, single step, run), and a status bar (step delay ms: 0).

The main window of the debugger is divided into several panes:

- Registers:** A table showing the state of various registers. The '0700: 02A1' register is highlighted in blue.
- Assembly:** A list of assembly instructions. The instruction 'mov bx, 0ah' is highlighted in blue.
- Memory:** A table showing memory addresses and their contents. The address '0700: 0187' is highlighted in blue.
- String:** A text area displaying the string 'esta es una prueba para contar el numero de caract'.

The assembly code in the main window is as follows:

```

xor ax, ax
mov al, contChar
div bx, 0ah
mov bl, al
add di, 48
mov string[di], al
inc ap1
xor ax, ax
mov al, bl
mul bl, 0ah
mov bx, 0ah
sub ax, 0ah

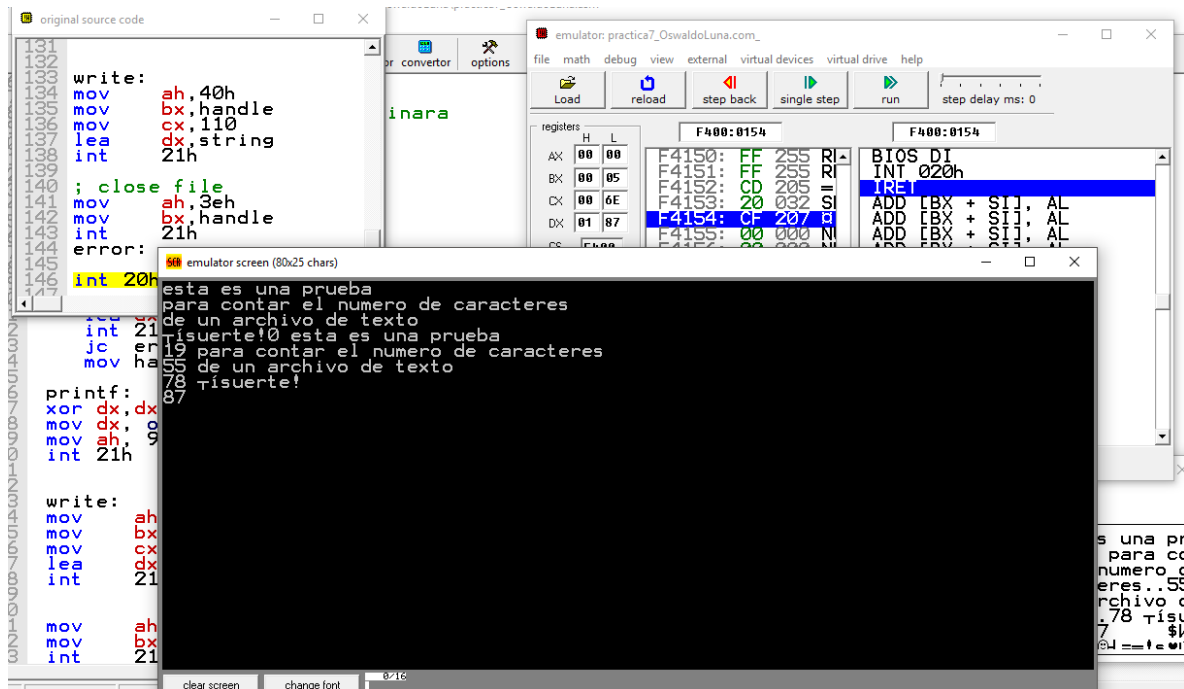
```

The memory dump shows the following data:

Address	Value
0700: 0187	30
0700: 0188	75
0700: 0189	65
0700: 018A	62
0700: 018B	20
0700: 018C	20
0700: 018D	20
0700: 018E	20
0700: 018F	20
0700: 0190	20
0700: 0191	20
0700: 0192	20
0700: 0193	20
0700: 0194	20
0700: 0195	20
0700: 0196	20
0700: 0197	20
0700: 0198	20
0700: 0199	20
0700: 019A	20
0700: 019B	20
0700: 019C	20
0700: 019D	20
0700: 019E	20
0700: 019F	20
0700: 01A0	20
0700: 01A1	20
0700: 01A2	20
0700: 01A3	20
0700: 01A4	20
0700: 01A5	20
0700: 01A6	20
0700: 01A7	20
0700: 01A8	20
0700: 01A9	20
0700: 01AA	20
0700: 01AB	20
0700: 01AC	20
0700: 01AD	20
0700: 01AE	20
0700: 01AF	20
0700: 01B0	20
0700: 01B1	20
0700: 01B2	20
0700: 01B3	20
0700: 01B4	20
0700: 01B5	20
0700: 01B6	20
0700: 01B7	20
0700: 01B8	20
0700: 01B9	20
0700: 01BA	20
0700: 01BB	20
0700: 01BC	20
0700: 01BD	20
0700: 01BE	20
0700: 01BF	20
0700: 01C0	20
0700: 01C1	20
0700: 01C2	20
0700: 01C3	20
0700: 01C4	20
0700: 01C5	20
0700: 01C6	20
0700: 01C7	20
0700: 01C8	20
0700: 01C9	20
0700: 01CA	20
0700: 01CB	20
0700: 01CC	20
0700: 01CD	20
0700: 01CE	20
0700: 01CF	20
0700: 01D0	20
0700: 01D1	20
0700: 01D2	20
0700: 01D3	20
0700: 01D4	20
0700: 01D5	20
0700: 01D6	20
0700: 01D7	20
0700: 01D8	20
0700: 01D9	20
0700: 01DA	20
0700: 01DB	20
0700: 01DC	20
0700: 01DD	20
0700: 01DE	20
0700: 01DF	20
0700: 01E0	20
0700: 01E1	20
0700: 01E2	20
0700: 01E3	20
0700: 01E4	20
0700: 01E5	20
0700: 01E6	20
0700: 01E7	20
0700: 01E8	20
0700: 01E9	20
0700: 01EA	20
0700: 01EB	20
0700: 01EC	20
0700: 01ED	20
0700: 01EE	20
0700: 01EF	20
0700: 01F0	20
0700: 01F1	20
0700: 01F2	20
0700: 01F3	20
0700: 01F4	20
0700: 01F5	20
0700: 01F6	20
0700: 01F7	20
0700: 01F8	20
0700: 01F9	20
0700: 01FA	20
0700: 01FB	20
0700: 01FC	20
0700: 01FD	20
0700: 01FE	20
0700: 01FF	20

The string 'esta es una prueba para contar el numero de caract' is displayed in the string pane. The debugger is currently at the instruction 'mov bx, 0ah'.

Y termina guardando en un nuevo archivo e imprimiendo el resultado en pantalla



## Conclusión

El manejo de archivos es la interrupción del procesador para leer en la memoria secundaria y obtener la información de ahí la cual se puede modificar eliminar copiar etc. El cual el objetivo es leer dentro del archivo y analizar la cantidad de caracteres antes que aparezca un salto de línea es decir un 13 en hexadecimal. Logrando así poder reconocer cuantos caracteres hay en cada renglón aunque a final de cuentas todo el archivo no tiene saltos de renglón sino que por ascii parece que existe el salto de renglón aunque todo está en un mismo arreglo. También se comprendió más el manejo de cadenas y como moverse en esta y agregar en esta parte por parte.