UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍA
DIVISION DE ELECTRONICA Y COMPUTACION

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

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

Reporte de Actividad Práctica

(reporte de investigación, síntesis de lectura o cuestionario)

Práctica 7: "Manejo de Archivos"

Alumno: Torres Rivera Rodrigo
Código: 397423431

Sección: D01

Profesor: Valentín Martínez López

Fecha 11 de Noviembre de 2025
firma de revisado

Nombre de la Práctica: Manejador de Archivos

Objetivo de la Práctica: **Desarrollar un programa en lenguaje ensamblador que lea un archivo de texto con extensión `.asm`, cuente el número de caracteres de cada línea (incluyendo los caracteres de salto de línea) y genere un archivo de salida con extensión `.lst`. En este archivo de salida, cada línea debe comenzar con el número de caracteres acumulados hasta ese momento, expresado en formato de tres dígitos, seguido del contenido original de la línea correspondiente del archivo de entrada.**

Antecedentes:

Se requiere contar con el siguiente material:
-Computadora con sistema operativo compatible.
-Emulador EMU8086 instalado.
-Incluir la libreria "emu8086.inc", para las macro funciones
-Estar familiarizado con el uso de las interrupciones en especial Int 21h

Desarrollo:

1. ***Lectura de un archivo de texto***:
   - El programa debe leer un archivo de texto con extensión `.asm` cuyo nombre se introducirá por teclado.
   - El archivo de entrada debe poder ser leído desde cualquier ruta (directorio o unidad).
2. ***Procesamiento del archivo***:
   - El programa debe contar el número de caracteres de cada línea del archivo de texto.
   - Debe mantener un contador acumulativo de caracteres que incluya los caracteres de cada línea y los caracteres de salto de línea (LF y CR).
3. ***Generación de un archivo de salida***:
   - El programa debe crear un nuevo archivo con el mismo nombre que el archivo de entrada, pero con extensión `.lst`.
   - En cada línea del archivo de salida, se debe imprimir el número de caracteres acumulados hasta ese momento, seguido del contenido original de la línea correspondiente del archivo de entrada.
   - El número de caracteres acumulados debe imprimirse en formato de tres dígitos (por ejemplo, `000`, `012`, `027`, etc.).
4. ***Requisitos adicionales***:
   - El programa debe imprimir en pantalla el nombre del alumno que lo desarrolló.
   - El programa debe utilizar una variable para almacenar la cantidad de caracteres acumulados.
   - El programa debe manejar correctamente la interrupción `21h` o funciones de librerías para realizar las operaciones de entrada/salida.
   - El programa debe terminar correctamente, asegurando que el registro `SP` tenga el valor `FFF8` al finalizar. Este valor indica que los procedimientos y el retorno al sistema operativo se han realizado adecuadamente.
5. ***Consideraciones técnicas***:
   - El programa debe ser capaz de manejar archivos de texto desde cualquier ubicación (directorio o unidad).
   - El archivo de salida debe generarse en la misma ubicación que el archivo de entrada.
   - El programa debe ser robusto y manejar correctamente los casos en los que el archivo de entrada no exista o no pueda leerse.

6. ***Registro SP (Stack Pointer)***:
   - El registro SP debe tener el valor `FFF8h` al finalizar el programa correctamente. Este valor indica que los procedimientos y el retorno al sistema operativo se han realizado de manera adecuada.
   - Si el registro SP tiene este valor debido al uso incorrecto de instrucciones que lo modifiquen, pero no se cierran correctamente los procedimientos o el retorno al sistema operativo, la práctica será rechazada.

Resultados Obtenidos:

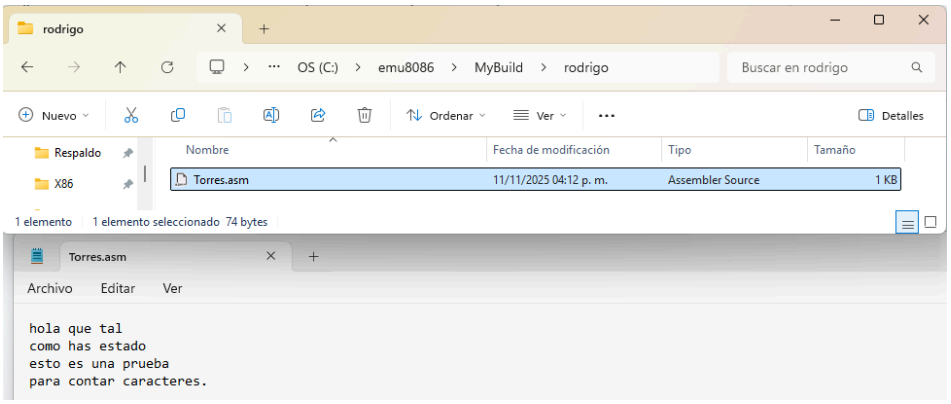
Ejemplo de entrada y salida:

- Si el archivo de entrada contiene:

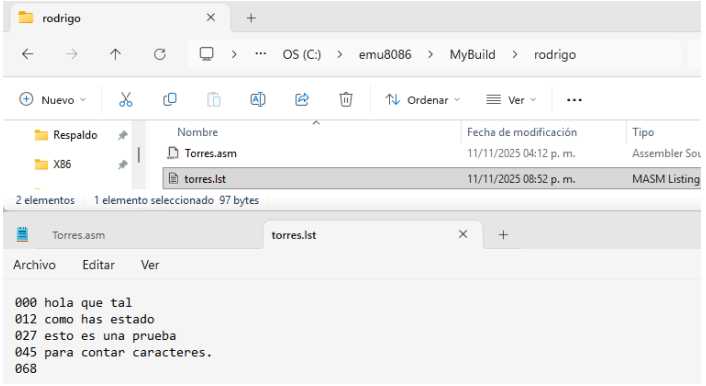
```
hola que tal
como has estado
esto es una prueba
para contar caracteres.
```

- El archivo de salida debe contener:

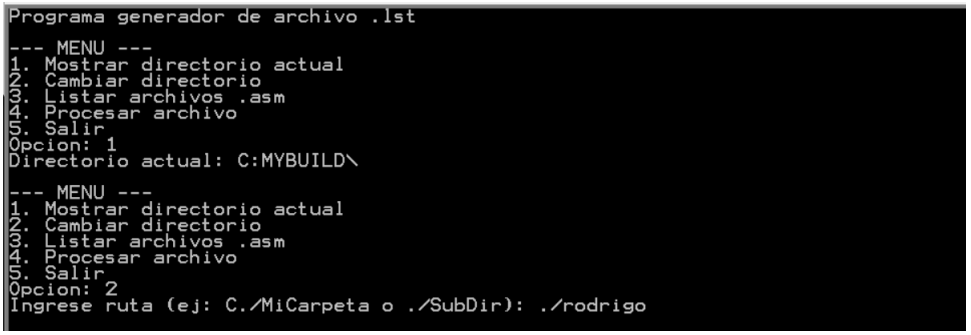
```
000 hola que tal
012 como has estado
027 esto es una prueba
045 para contar caracteres.
068
```



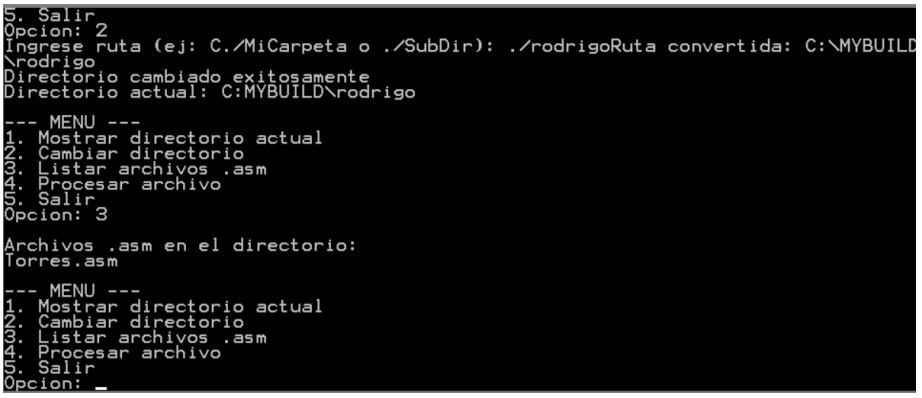
Tras la ejecución del programa tendremos:



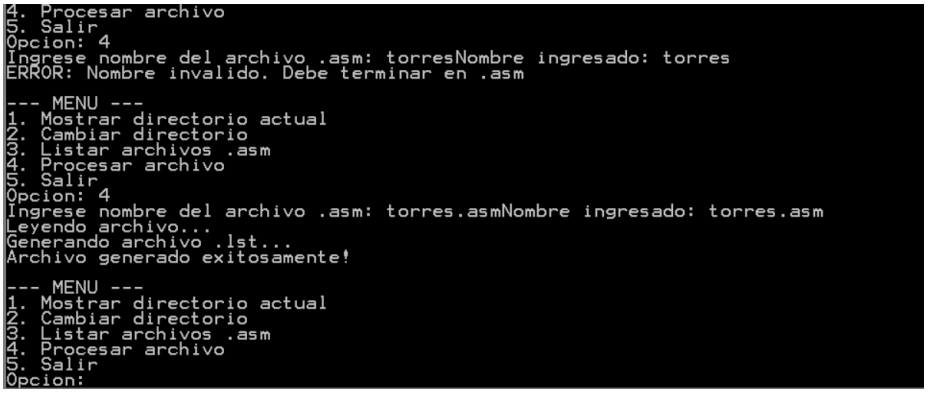
El programa cuenta con un menú para manejar las diferentes funciones, esto es posible gracias al uso de las Macro funciones, que nos facilitan la escritura de Mensajes y la Lectura de información. Primero listamos el directorio en el que estamos ubicados usando la opción 1, para verificar si es accesible o no el archivo de ejemplo *torres.asm*.



Como no estamos en la ruta del archivo nos cambiamos utilizando la opción 2, se nos pide ingresemos la ruta la cual por comodidad se puede ingresar como ./directorio para evitar escribir el directorio padre, se hace notar que debemos escribir la ruta en formato de Linux, debido a que Emu8086 cambia el layout del teclado y no acepta todos los caracteres. listamos los archivos .asm del directorio para verificar que se encuentra el archivo *torres.asm* de prueba usamos la opción 3 del menú:



Tras verificar que el archivo está disponible lo procesamos con la opción 4, observe que si nos equivocamos el programa puede manejar un error en el nombre del archivo:



Igualmente existen medidas de control en caso de fallo en las demás opciones anteriores, así mismo si escribimos una opción no disponible en el menú, para terminar usamos la opción 5.

Conclusiones:

El lenguaje Ensamblador es capaz de realizar todas las operaciones de los lenguajes de alto nivel, debido a que al final todo el código de alto nivel debe “traducirse” a código de bajo nivel que la máquina puede entender y el ensamblador no es más que una máscara simple de este código.