1. Sección .data y .bss

En esta sección se reservan espacios en la memoria para almacenar datos estáticos y variables que serán utilizadas durante la ejecución del programa. Los buffers y variables definidas aquí son fundamentales para el manejo de archivos y la manipulación de cadenas de texto.

- **file, file2:** Buffers para almacenar los nombres de archivo.
- **buffer, buffer2:** Buffers para almacenar el contenido de los archivos.
- **linea1, linea2:** Buffers para almacenar las líneas actuales de los archivos.
- **overwrite, anterior, despues:** Buffers para almacenar el contenido antes, durante y después de la línea que se está editando.
- **text:** Buffer para almacenar el contenido nuevo después de la edición.
- **input:** Buffer para almacenar la entrada del usuario.
- **char:** Buffer para imprimir caracteres en formato hexadecimal.
- **diffBuffer:** Buffer para almacenar las diferencias entre dos archivos.
- Otras variables como **lineas** y **r15** que se utilizan para contar líneas y manejar parámetros de función.

2. Sección .text

Aquí se encuentran las instrucciones de código ensamblador que conforman el programa principal. Las principales funciones y bloques de código incluyen:

- **_start**

Es el punto de entrada del programa. Aquí se comprueba si se proporcionan argumentos al ejecutar el programa y se dirige el flujo de control a la función correspondiente según la cantidad de argumentos.

- **_noArg**

Esta función maneja el caso en que el usuario no proporciona argumentos al ejecutar el programa. Solicita al usuario que ingrese el nombre de un archivo mediante un mensaje en pantalla y espera la entrada del usuario.

- **_openFile**

Abre el archivo especificado por el usuario y guarda su contenido en un buffer para su posterior procesamiento. Si no puede abrir el archivo, muestra un mensaje de error y termina la ejecución del programa. RDI recibe el nombre de archivo.

- **ReadLine**

Lee una línea del archivo, muestra su contenido al usuario y permite que este lo edite. La edición se realiza en un bucle donde el usuario puede modificar la

línea actual. Una vez que el usuario confirma los cambios, se procede a guardar la línea editada en el archivo. RSI recibe el puntero de la línea actual.

- **ReadLine.save**

Guarda los cambios realizados por el usuario en el archivo. Primero tenemos que guardar los cambios de la línea actual en el buffer overwrite de la memoria. Después guardamos el contenido posterior a la línea actual, ya que RSI recibe el puntero de la otra línea, simplemente lo guardamos en un buffer. Luego con la cantidad de líneas leídas del contador, tenemos que hacer un loop que guarde el contenido anterior a la línea actual, empezando el RSI al inicio del buffer.

Con todo ese proceso hecho, podemos abrir el archivo en modo escritura y escribimos el nuevo texto.

- **_oneArg**

Maneja el caso en que el usuario proporciona un solo argumento al ejecutar el programa. Este argumento puede ser para mostrar ayuda, leer, editar o ver el archivo en formato hexadecimal. Dependiendo del argumento proporcionado, el programa realiza la acción correspondiente.

- **compareLines**

Compara las diferencias entre la linea actual del archivo 2 con respecto al archivo 1. RDI recibe la línea actual del archivo 2, mientras que RSI recibe la línea actual del archivo 1.

- **storeLine1 y storeLine2**

Almacenan las líneas actuales de los archivos en los buffers linea1 y linea2. Estas líneas se utilizan para comparar, encontrar y mostrar las diferencias durante el proceso. RDI y RSI reciben el puntero de la línea actual del buffer 1 y 2 respectivamente.

- **cleanBuffer**

Limpia un buffer, estableciendo todos sus elementos en cero. Esta función se utiliza para garantizar que los buffers estén vacíos antes de almacenar nuevos datos.

- **fwrite**

Escribe en un archivo los cambios realizados por el usuario. Esta función es llamada después de que el usuario confirma los cambios realizados en una línea del archivo.

- **delReturn**

Elimina el retorno de carro del nombre de archivo. Esto es útil para limpiar la entrada del usuario cuando se proporciona un nombre de archivo.

- **writeString**

Imprime una cadena en la salida estándar. Esta función es utilizada para mostrar mensajes y mensajes de error durante la ejecución del programa.

- **strLen**

Calcula la longitud de una cadena. Esta función es utilizada para determinar la cantidad de caracteres en una cadena dada. RSI recibe el puntero y este revisa byte por byte si el byte es igual a 0. RCX es el contador.

- **readInput**

Lee una entrada del usuario y la almacena en un buffer. Esta función se utiliza para obtener la entrada del usuario durante la ejecución del programa.

- **fopen, fread, fclose**

Funciones para abrir, leer y cerrar archivos respectivamente. Estas funciones son llamadas para manipular archivos durante la ejecución del programa.

- **strchr**

Encuentra el siguiente retorno de carro en una cadena. Esta función se utiliza para buscar el siguiente retorno de carro en una cadena dada.

- **strLen0ah**

Calcula la longitud de una cadena hasta encontrar un retorno de carro. Esta función se utiliza para determinar la longitud de una cadena hasta que se encuentre un retorno de carro.

- **printf**

Imprime una cadena de texto con longitud especificada. Esta función es una implementación básica de la función de impresión formateada.

- **countLines**

Cuenta las líneas en un archivo. Esta función es utilizada para determinar la cantidad de líneas en un archivo dado.

- **clearScreen**

Limpia la pantalla. Esta función se utiliza para limpiar la pantalla antes de imprimir un nuevo contenido, lo que mejora la legibilidad de la salida.

- **openFile**

Abre un archivo. Esta función se utiliza para abrir un archivo dado su nombre y modo de apertura.

- **printArchivo**

Imprime un mensaje indicando el archivo con el que se está trabajando. Esta función se utiliza para mostrar información sobre el archivo que está siendo editado o procesado.

- **guardarDespues y guardarAnterior**

Guardan el contenido antes y después de la línea actual. Estas funciones son utilizadas durante la edición de un archivo para almacenar el contenido antes y después de la línea que se está modificando.

- **cmpStr**

Compara dos cadenas de texto. Esta función es utilizada para comparar dos cadenas y determinar si son iguales o diferentes.

- **readFileName y readFilenameCmdLine**

Leen un nombre de archivo proporcionado por el usuario. Estas funciones se utilizan para obtener nombres de archivo desde la entrada del usuario, ya sea desde la línea de comandos o durante la ejecución del programa.

- **power y convertToASCII**

Funciones auxiliares utilizadas para realizar operaciones matemáticas simples y convertir caracteres a su valor ASCII respectivamente.

- **viewHex**

Muestra el contenido de un archivo en formato hexadecimal. Esta función es utilizada para mostrar el contenido de un archivo en formato hexadecimal, lo que puede ser útil para la depuración y visualización de datos binarios. RDI recibe el buffer del archivo y en un ciclo, por cada byte que pasa por el RDI llamamos la función itoa (convertToASCII) para guardar cada byte en un memoria y luego imprimir cada byte en formato hexadecimal.

- **storeSecondFilename**

Guarda el nombre del segundo archivo para la comparación de diferencias. Esta función se utiliza para almacenar el nombre del segundo archivo que se utilizará para comparar diferencias durante la ejecución del programa.

3. Sección .rodata

Esta sección contiene cadenas de texto constantes que se utilizan en el programa, como mensajes de error, ayuda, instrucciones, etc. Estos mensajes son mostrados al usuario durante la ejecución del programa para proporcionar información o notificar sobre errores.