Manual Técnico

Introducción

Descripción General del Programa

Este programa de ARM64 está diseñado para gestionar y procesar datos en una hoja de cálculo simplificada. Ofrece funcionalidades avanzadas como el ingreso de datos, la manipulación de celdas y la ejecución de operaciones aritméticas y lógicas. Entre las principales características se encuentran la capacidad de importar datos desde archivos CSV, el manejo de diferentes comandos para realizar cálculos y manipulaciones en la hoja de cálculo, y la generación de reportes detallados en archivos de texto.

Objetivos y Alcance

El principal objetivo de este programa es proporcionar una herramienta flexible y eficiente para la gestión de datos en una hoja de cálculo. Está diseñado tanto para usuarios individuales que requieren una solución ligera y poderosa, como para propósitos educativos, permitiendo a los estudiantes y profesionales comprender y visualizar el funcionamiento de diferentes operaciones de datos.

El alcance del programa incluye:

- Ingreso de datos por teclado y desde archivos CSV.
- Manipulación y edición de celdas en la hoja de cálculo.
- Ejecución de operaciones aritméticas (suma, resta, multiplicación, división) y lógicas (OR, AND, XOR, NOT) en las celdas.
- Generación de reportes en archivos de texto, documentando el proceso y resultado de las operaciones realizadas.

Requisitos del Sistema

Para ejecutar este programa, se requiere un entorno de desarrollo compatible con ARM64, como QEMU para emulación, y un ensamblador ARM64, como `aarch64-linux-gnu-as`, para compilar el código ensamblador. Además, se necesita un entorno de terminal para interactuar con el programa, y un editor de texto para modificar el código fuente si es necesario.

Estructura del Código

Organización de Archivos y Directorios

El programa está organizado en un solo archivo de código ARM64 que contiene todas las definiciones de variables, funciones principales, subrutinas auxiliares y macros. La estructura del código está dividida en secciones claras y bien definidas, que incluyen la inicialización de datos, las funciones de manejo de archivos, y las subrutinas de operaciones matemáticas y lógicas.

- **Sección de Datos (`.data` y `.bss`)**:
    - Aquí se declaran y reservan todas las variables y buffers necesarios para el funcionamiento del programa.
    - Se incluyen definiciones de cadenas de texto, buffers de entrada y salida, y estructuras para almacenar datos de la hoja de cálculo.
- **Sección de Texto (`.text`)**:
    - Contiene todas las funciones y subrutinas del programa.
    - Incluye la lógica principal del programa, desde el punto de entrada (`_start`) hasta las funciones específicas para cada comando y operación.
- **Macros**:
    - Definidas al inicio del código para facilitar la repetición de ciertas operaciones, como imprimir (`print`, `printr`), leer (`Input`), y manipular la posición en la hoja de cálculo (`setPosicion`).

La estructura modular del código permite una fácil lectura y mantenimiento, facilitando la comprensión de cada componente y su función dentro del programa.

Convenciones de Nombres y Estilo de Código

El código sigue un conjunto de convenciones de nombres y estilo que aseguran claridad y consistencia:

- **Nombres de Variables**:
    - Nombres descriptivos que indican claramente el propósito de la variable, como `Hoja_Calculo`, `bufferComando`, `filename`, `retorno`, etc.
    - Uso de mayúsculas y guiones bajos (_) para separar palabras en los nombres de variables y hacerlos más legibles.
- **Nombres de Funciones**:
    - Funciones y subrutinas tienen nombres que describen su propósito o acción, como `Abrir_Archivo`, `Cerrar_Archivo`, `LeerCSV`, `Imprimir_Hoja`, `Comando_Suma`, `Verificar_Operando`, etc.
    - Uso de prefijos comunes para agrupar funciones relacionadas, por ejemplo, `Comando_` para todas las funciones que manejan comandos específicos.
- **Estructura de Bloques**:

        - Uso claro de comentarios para delinear secciones del código y explicar la lógica de bloques específicos.
        - Agrupación de instrucciones relacionadas en bloques lógicos, facilitando la lectura y comprensión del flujo del programa.
- **Comentarios**:
        - Comentarios detallados que explican la funcionalidad de cada bloque de código, especialmente en partes complejas o críticas.
        - Uso de comentarios en línea para describir el propósito de instrucciones específicas y variables.

Estas convenciones y la organización clara del código aseguran que el programa sea fácil de mantener y expandir, permitiendo a los desarrolladores y usuarios entender rápidamente el propósito y funcionamiento de cada componente.

Sección de Datos

Definiciones de Variables y Buffers

En la sección de datos de este programa, se definen todas las variables y buffers necesarios para el manejo de la entrada de datos, almacenamiento temporal y manipulación de cadenas. Aquí se declaran las variables utilizadas para la gestión de los datos de entrada y salida, así como los buffers para el procesamiento interno.

- **Variables Principales**:
    - `Hoja_Calculo`: Matriz principal donde se almacenan todos los datos de la hoja de cálculo.
    - `val`: Espacio temporal utilizado para almacenar valores durante el procesamiento.
    - `bufferComando`: Buffer para almacenar los comandos ingresados por el usuario.
    - `filename`: Buffer para almacenar el nombre del archivo CSV que se leerá.
    - `buffer`: Buffer general para almacenar datos temporales.
    - `Descriptor`: Variable para almacenar el descriptor de archivo durante la operación de lectura/escritura.
    - `ListaIndices`: Buffer para almacenar los índices de las columnas durante la importación de datos.
    - `num`: Buffer temporal para almacenar un número durante la conversión de ASCII a entero.
    - `col_imp`: Variable para almacenar el índice de la columna importada.
    - `Caracter`: Variable para almacenar un carácter leído desde la entrada estándar.
    - `count`: Variable que almacena la cantidad de números leídos desde el archivo CSV o ingresados manualmente.
    - `op1`, `op2`: Variables temporales utilizadas para almacenar operandos durante las operaciones.
    - `celda`: Espacio reservado para almacenar referencias a celdas.
    - `Num_f`: Buffer temporal para almacenar un número de fila.
    - `retorno`: Variable utilizada para almacenar el resultado de las operaciones.
- **Buffers de Cadenas y Mensajes**:
    - `clear`: Comando para limpiar la pantalla.
    - `dosPuntos`: Buffer que contiene los dos puntos ":".
    - `encabezado`: String que contiene el encabezado del programa, incluyendo información de la universidad, el curso y el autor.
    - `salto`: Buffer que contiene el carácter de salto de línea.

- `espacio`, `espacio2`: Buffers que contienen caracteres de espacio.
- `Columns`: String que contiene los títulos de las columnas de la hoja de cálculo.
- `filas`: String que contiene los títulos de las filas de la hoja de cálculo.
- `ComandoImportar`, `ComandoGuardar`, `Comando_Suma`, `Comando_Resta`, `Comando_Multiplicacion`, `Comando_Dividir`, `Comando_Potenciar`, `Comando_OLOGICO`, `Comando_YLOGICO`, `Comando_OXLOGICO`, `Comando_NOLOGICO`, `Comando_LLenar`, `Comando_Promedio`, `Comando_Minimo`, `Comando_Maximo`: Buffers que contienen los nombres de los diferentes comandos soportados por el programa.
- `mensajeErrorComando`, `mensajeNoCeldasValidad`, `mensajeIngresarValor`, `mensajeErrorRango`, `mensajeValorNoValido`, `errorExponenteNegativo`, `errorDivisionCero`, `errorImport`, `errorSuma`, `errorAbrir_Archivo`, `getIndexMsg`, `LeidoCorrectamente`, `errorCeldas`, `errorDesbordamiento`: Mensajes predefinidos para manejar errores y advertencias.

Inicialización de Datos

Al inicio del programa, las variables y buffers se inicializan para asegurarse de que están en un estado adecuado antes de cualquier operación. Esto se logra mediante la sección `.bss`, donde las variables se inicializan con ceros o espacios reservados para datos temporales.

- **Estructura `.bss`**:
    - Define las variables que no requieren un valor inicial específico.
    - Utiliza `.space` y `.zero` para reservar espacio en memoria para las variables.

Esta sección proporciona una base sólida para el manejo y manipulación de datos a lo largo del programa. La correcta definición y inicialización de estas variables son cruciales para el funcionamiento eficiente y preciso del programa.

Funciones Principales y Funciones de Archivo

`_start`: Punto de Entrada del Programa

La función `_start` es el punto de entrada principal del programa. Inicializa el entorno y establece el flujo de control principal del programa. Primero, limpia la pantalla, luego imprime el encabezado, y finalmente inicia el bucle de entrada de comandos, permitiendo al usuario interactuar con la hoja de cálculo.

`menu`: Menú Principal

La función `menu` despliega las opciones principales para el usuario y espera la selección de un comando. El menú principal permite al usuario elegir entre las distintas operaciones soportadas por el programa, como importar datos, guardar el estado actual de la hoja de cálculo, o realizar operaciones aritméticas y lógicas.

`metodoCSV`: Ingreso de Datos desde un Archivo CSV

`metodoCSV` gestiona la lectura de datos desde un archivo CSV. Solicita el nombre del archivo, lo abre, lee los datos y los almacena en el buffer principal (`Hoja_Calculo`). Esta función facilita la carga de datos desde fuentes externas, asegurando una mayor flexibilidad para el usuario.

`Imprimir_Hoja`: Impresión de la Hoja de Cálculo

La función `Imprimir_Hoja` imprime el estado actual de la hoja de cálculo en la consola, mostrando filas y columnas formateadas para facilitar la lectura. Esta función es esencial para que el usuario visualice los datos de manera clara y ordenada.

`int_to_Ascii`: Conversión de Entero a Cadena ASCII

Esta función convierte valores enteros a su representación en formato de cadena ASCII. Esta conversión es vital para la impresión de números y la generación de reportes. Permite que los datos numéricos se presenten de manera legible y comprensible para el usuario.

`Ascii_to_int`: Conversión de Cadena ASCII a Entero

La función `Ascii_to_int` convierte cadenas ASCII que representan números en sus valores enteros correspondientes. Es una función auxiliar vital para procesar las entradas de texto del usuario y convertirlas en datos numéricos utilizables dentro del programa.

Funciones de Archivo

`Abrir_Archivo`: Apertura de Archivos

La función `Abrir_Archivo` maneja la apertura de archivos, estableciendo los permisos necesarios y asegurando que el archivo esté listo para operaciones de lectura. Si ocurre un error durante la apertura, la función maneja este error y notifica al usuario.

`Cerrar_Archivo`: Cierre de Archivos

`Cerrar_Archivo` cierra un archivo abierto, liberando cualquier recurso asociado con el descriptor de archivo. Esto es esencial para evitar fugas de memoria y asegurar que los datos se guarden correctamente.

`LeerCSV`: Lectura de Archivos CSV

La función `LeerCSV` lee datos desde un archivo CSV y los almacena en la `Hoja_Calculo`. Procesa cada línea del archivo, manejando la separación por tabuladores y el almacenamiento de los valores en las celdas correspondientes de la hoja de cálculo.

Funciones Aritméticas

`Comando_Suma`: Ejecución del Comando de Suma

La función `Comando_Suma` realiza la suma de dos operandos que pueden ser números, celdas o el valor almacenado en la variable `retorno`. Verifica y extrae los operandos, realiza la suma y almacena el resultado en `retorno`.

`Comando_Resta`: Ejecución del Comando de Resta

Similar a `Comando_Suma`, esta función realiza la resta de dos operandos. Verifica y extrae los operandos, realiza la resta y almacena el resultado en `retorno`.

`Comando_Multiplicacion`: Ejecución del Comando de Multiplicación

La función `Comando_Multiplicacion` realiza la multiplicación de dos operandos. Verifica y extrae los operandos, realiza la multiplicación y almacena el resultado en `retorno`.

`Comando_Dividir`: Ejecución del Comando de División

Esta función realiza la división de dos operandos, manejando el posible error de división por cero. Verifica y extrae los operandos, realiza la división y almacena el resultado en `retorno`.

Funciones Lógicas

`Comando_OLOGICO`: Ejecución del Comando OR Lógico

La función `Comando_OLOGICO` realiza la operación OR lógico entre dos operandos. Verifica y extrae los operandos, realiza la operación OR lógico y almacena el resultado en `retorno`.

`Comando_YLOGICO`: Ejecución del Comando AND Lógico

La función `Comando_OLOGICO` realiza la operación AND lógico entre dos operandos. Verifica y extrae los operandos, realiza la operación AND lógico y almacena el resultado en `retorno`.

`Comando_NOLOGICO`: Ejecución del Comando NOT Lógico

La función `Comando_OLOGICO` realiza la operación NOT lógico entre dos operandos. Verifica y extrae los operandos, realiza la operación NOT lógico y almacena el resultado en `retorno`.

`Comando_OXLOGICO`: Ejecución del Comando OR Lógico

La función `Comando_OLOGICO` realiza la operación XOR lógico entre dos operandos. Verifica y extrae los operandos, realiza la operación XOR lógico y almacena el resultado en `retorno`.

Estas funciones principales, de archivo, aritméticas y lógicas son la columna vertebral del programa, permitiendo la interacción del usuario con la hoja de cálculo, el manejo de datos, la realización de operaciones y la generación de reportes. Han sido diseñadas para ser eficientes y fáciles de mantener, asegurando que el programa funcione de manera robusta y confiable.

Conclusiones

Este programa de ARM64 es una herramienta robusta para gestionar y procesar datos en una hoja de cálculo simplificada. Ofrece una variedad de funcionalidades avanzadas, incluyendo la capacidad de importar datos desde archivos CSV, manipular y editar celdas, y realizar operaciones aritméticas y lógicas. Además, permite generar reportes detallados en archivos de texto, documentando el estado y las operaciones realizadas en la hoja de cálculo. La combinación de estas características proporciona una solución eficiente y flexible para usuarios individuales y propósitos educativos.