INTERFAZ

Este es un código escrito en Python utilizando la biblioteca PyQt5 para crear una interfaz gráfica de usuario (GUI). El código representa un compilador o ensamblador, con un menú principal y diferentes páginas para realizar diversas acciones, como ver un manual, ver créditos y ejecutar el ensamblador.

Módulos y Librerías Importadas

-PyQt5
-QApplication: Clase principal para manejar la aplicación.
-QLabel, QPushButton, QWidget, QStackedLayout, QGridLayout, QVBoxLayout, QPlainTextEdit, QHBoxLayout: Widgets de PyQt5 para construir la interfaz gráfica.
-QPixmap: Utilizado para manejar imágenes y pixeles en PyQt5.
-QCursor: Define el cursor del mouse.
-QtCore, QtGui: Módulos principales de PyQt5.
-QUrl: Maneja las URLs en PyQt5.
-QDesktopServices: Proporciona funciones para acceder al escritorio del sistema.

Otros

-sys: Proporciona acceso a funciones específicas del sistema.

Variables Globales

-widgets: Un diccionario que contiene widgets que serán utilizados en el menú del compilador. Incluye "Regresar menu principal", "Iniciar", y "DIE".

-Clase pagina2:
Representa la segunda página de la aplicación.
Contiene widgets como imágenes (DIE y unam_eti) y botones (boton1, boton2, boton3).
Define métodos para realizar acciones al presionar ciertos botones (B_manual, Ventana_programa, Ventana_creditos).

-Clase Pagina_Creditos;
Representa la página de créditos de la aplicación.
Incluye una lista de miembros con nombres, nombres de archivos de imágenes y escalas.
Define métodos para agregar miembros, establecer escalas y limpiar el diseño.

-Clase ensamblador:
Representa la página del ensamblador.
Contiene un área de código (code_editor), un botón para ensamblar (assemble_button) y una etiqueta para mostrar resultados (result_label).
Define el método assemble_code para procesar y ensamblar el código ingresado.

-Clase MainMenu:
Representa el menú principal de la aplicación.
Crea y organiza las diferentes páginas y widgets del menú.
Define métodos para cambiar entre páginas (show_main_menu, ventana_2, Mostar_pag_Creditos, Mostar_pag_Programa).

Flujo Principal
Se crea una aplicación (QApplication) y se muestra el menú principal (MainMenu).
El usuario puede navegar entre las páginas haciendo clic en los botones correspondientes.
Cada página realiza acciones específicas, como abrir un archivo PDF, mostrar créditos o ensamblar código.

LÓGICA

Este script es un ensamblador simple diseñado para el lenguaje de ensamblaje Z80. El programa realiza dos pasadas a través del código fuente. Durante la primera pasada, se construye la tabla de símbolos y se valida la sintaxis. Durante la segunda pasada, se genera el código objeto y se reemplazan las etiquetas con sus direcciones correspondientes.

Funciones Principales:
macroEnsamble(archivo)
Esta función realiza el ensamblaje del código fuente, manejando las macros y generando una lista de traducción.

pasada1()
La primera pasada se encarga de construir la tabla de símbolos y realizar una validación de la sintaxis.

pasada2()
En la segunda pasada, se genera el código objeto y se reemplazan las etiquetas con sus direcciones correspondientes.

agregarTS()
Esta función agrega la tabla de símbolos al final de la lista de traducción.

ensamblar(codigo)
Función principal para ensamblar el código proporcionado. Invoca las pasadas necesarias y genera el archivo LST.

Variables Globales:
tablaRegistros
Un diccionario que mapea los registros a sus códigos binarios correspondientes.

tablaPares
Diccionario que asigna códigos binarios a pares de registros.

tablaBits
Diccionario que mapea bits a sus códigos binarios.

tablaCondiciones
Diccionario que asigna códigos binarios a condiciones.

palabrasReservadas
Una lista de palabras reservadas que no pueden usarse como símbolos.

tablaSimbolos
Un diccionario que almacena la tabla de símbolos, donde los símbolos se asignan a direcciones.

tablaMacros
Un diccionario que almacena información sobre las macros, incluidos sus parámetros y líneas.

traduccion
Una lista que almacena la traducción final del código fuente.

cl
Un contador que lleva la cuenta de la dirección actual de la memoria.

Funciones de Bajo Nivel:
validarArchivo(archivo)
Valida si el archivo existe y tiene la extensión .asm.

leerArchivo(archivo)
Lee el contenido de un archivo.

formatoLinea(linea)
Formatea una línea del código fuente, eliminando espacios y tabulaciones innecesarios.

crearLST(nombre)
Crea y guarda el archivo LST a partir de la traducción final.

agregarSimbolo(simbolo)
Añade un símbolo a la tabla de símbolos después de validar su formato y unicidad.

buscarEtiqueta(simbolo)
Busca un símbolo en la tabla de símbolos y devuelve su dirección si existe.

//
Comentarios y Observaciones:
El código realiza una validación sintáctica durante las pasadas, lanzando excepciones en caso de errores.
Las funciones validarMnemonico(linea, pasada1) y ensamblar(codigo) podrían beneficiarse de una mayor modularización para mejorar la legibilidad y facilitar la comprensión del código.
La salida del programa se guarda en un archivo LST, y se proporciona información detallada, incluyendo la tabla de símbolos.
Este código es un ensamblador simple que puede ser extendido y mejorado para admitir más características del lenguaje Z80 y para manejar casos más complejos.