DIVISIÓN DE TECNOLOGÍAS PARA LA INTEGRACIÓN CIBER-HUMANA

Arquitectura de Computadoras

Jorge Ernesto López Arce Delgado

Proyecto Final

Irving Eduardo Celedón Sánchez

Yoshua Isaac Díaz Castillo

MIPS es un tipo de arquitectura de microprocesador que utiliza un enfoque de computación de conjunto de instrucciones reducido (RISC), es decir, es una forma de organizar cómo funciona el procesador de una computadora, centrándose en la simplicidad y la velocidad, lo que la convierte en un modelo ideal para comprender los principios fundamentales de los sistemas computacionales.

Este proyecto desarrolla una aplicación de ensamblador que permita interpretar, validar y traducir instrucciones de tipo MIPS a sus representaciones binarias correspondientes.

A través de esta aplicación, el usuario puede cargar instrucciones, visualizarlas, procesarlas y exportar los resultados obtenidos.

El objetivo general será, por supuesto, desarrollar la aplicación ensambladora para instrucciones tipo MIPS que permita interpretar, procesar y convertir instrucciones mediante una interfaz gráfica elaborada sobre Python.

Los objetivos particulares que tiene el proyecto son

- Implementar la lógica central de un ensamblador tipo MIPS que soporte parsing, validación y traducción de instrucciones.
- Diseñar una arquitectura modular que permita mantener la separación entre la lógica de negocio, el manejo de archivos, la interfaz gráfica y la administración de configuraciones.
- Crear una interfaz gráfica intuitiva que permita cargar archivos de instrucciones, visualizar su contenido y obtener el resultado convertido.
- Implementar un sistema de configuración general del programa, incluyendo soporte para traducciones y opciones personalizables.

- Realizar pruebas funcionales utilizando archivos de ejemplo y verificando la correcta conversión de instrucciones.
- Documentar el proceso de desarrollo siguiendo la guía institucional.

El proyecto está organizado en múltiples carpetas y archivos libres, siendo que cada uno cuenta con responsabilidades claras, como por ejemplo:

main.py -

Punto de entrada de la aplicación, se encarga de inicializar controlador principal.

core - Contiene la lógica de negocio del sistema, incluyendo:

assembler.py - Procesa instrucciones, analiza tokens y genera representaciones binarias.

assembler_controller.py - Conecta la lógica del ensamblador con la interfaz gráfica.

settings_manager.py - Maneja configuraciones globales (como idioma o preferencias del usuario) y su permanencia a través de las sesiones.

io - Funciones para lectura, escritura y validación de archivos.

El ensamblador implementado funciona siguiendo los pasos típicos de un proceso de ensamblado:

- Lectura de instrucciones del usuario o desde un archivo.
- Parsing: Segmentación de cada instrucción en sus componentes (opcode, registros, inmediatos, etiquetas).
- Validación sintáctica: Comprobación de formatos, operandos y estructura.

- Correspondencia con el diccionario de instrucciones, ubicado en resources/assembler_data.py.
- Traducción: Conversión de la instrucción a código numérico, representación binaria o formato de salida seleccionado.
- Entrega del resultado a la interfaz gráfica para visualizar o exportar.

El sistema permite manejar múltiples instrucciones y proporciona mensajes claros en caso de errores de sintaxis o incompatibilidad.

La GUI facilita la interacción del usuario con el ensamblador. Está dividida en carpetas que actúan como paquetes:

mw (Main Window) - Ventana principal del sistema, permite cargar archivos, visualizar instrucciones, ejecutar el ensamblador y mostrar resultados e incluye botones, campos de texto y áreas para mostrar la conversión generada.

widgets_factory.py y **styles.py - G**eneran widgets personalizados con estilos consistentesy mejoran la experiencia de usuario y mantienen un diseño uniforme.

sw (Settings Window) - Controla las ventanas de configuración y permite cambiar idioma, tema visual u otras preferencias

En la carpeta **tests** dentro del repositorio de github se incluyen:

- Archivos de instrucciones de ejemplo.
- Capturas de pantalla que prueban la funcionalidad del sistema, incluyendo:
- Carga de archivos .txt
- Conversión de instrucciones

- Guardado de resultados
- Manejo de múltiples instrucciones
- Estas pruebas demuestran el correcto funcionamiento del ensamblador, así como su integración completa con la interfaz gráfica.

Conclusiones

El desarrollo de este proyecto permitió comprender de manera práctica los principios fundamentales de la arquitectura MIPS y el proceso de ensamblado de instrucciones. La implementación de un ensamblador real, con parsing, validación y traducción de instrucciones, fortaleció los conocimientos en arquitectura de computadoras, programación estructurada y diseño de sistemas modulares.

La creación de una interfaz gráfica funcional facilitó la interacción con el ensamblador y mejoró la usabilidad del sistema, convirtiéndolo en una herramienta accesible para estudiantes y usuarios interesados en el funcionamiento de la arquitectura MIPS.

El proyecto resultó en una aplicación robusta, modular y bien organizada, capaz de procesar correctamente instrucciones tipo MIPS y de servir como base para futuras extensiones como simulación de pipeline o ejecución paso a paso.