Instituto Tecnológico de Costa rica

Escuela de Ingeniería en Computación

Compiladores e Intérpretes [IC - 5701]

Proyecto 3

Generación Código Destino (MIPS)

Estudiantes:

Eduardo Rojas Gómez 2023152827

Yosimar Montenegro Montenegro 2023147365

Profesor(a) a cargo:

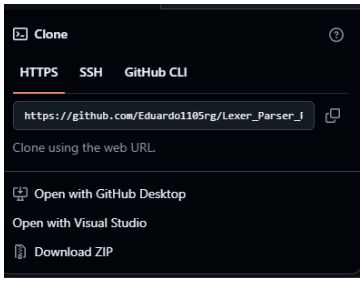
Allan Rodríguez Dávila

II Semestre | 2025

Enlace al repositorio del proyecto: https://github.com/Eduardo1105rg/Lexer_Parser_PY1_Compi.git

Instrucciones de compilación:

Lo primero paso se deberá de clonar el repositorio del proyecto o descargar el zip de este.



Como se muestra en la imagen adjunta, en esa parte de a página de git se realiza este proceso.

Una vez se tiene el proyecto en la computadora, se tiene que abrir desde un editor de código, una vez que se abre el proyecto se ejecutara los siguientes comandos en la terminar para compilar y ejecutar:

- Primero, ejecutar el comando ‘./grandlew clean’ para limpiar el .build, esto con el fin de poder crear los nuevos archivos.

- Después ejecutar el comando ‘./grandlew generateLexer’ para generar el lexer y posteriormente el comando ‘./grandlew generateParser’ para generar el parser.

- En caso de ser necesario en la capeta ‘src/test/resources/test-programs’ se definen los archivos que se desean probar, es decir los archivo con la gramática del lenguaje que se está desarrollando.

- Ahora usando el comando ‘./gradlew run --args="src/test/resources/test-programs/ejemplo1.txt"’ se ejecuta todo el programa.

Una vez ejecutado se producirá un archivo Tokens.txt en el cual se tienen los tokens y lexemas que se encontraron al leer el archivo de entrada, además en la consola se mostraran los errores léxicos, sintácticos y semánticos que se encontraron al leer el archivo fuente, además después de los errores se imprimirán las tablas de símbolos creada durante el proceso de análisis.

Adicionalmente se genera un archivo llamado tac_output.txt con el código intermedio en formato tres direcciones basado en el archivo fuente.

Por último, en un archivo se generará un archivo que contendrá el código destino (lenguaje ensamblador mips) el cual estaría listo para ser ejecutado en programas como QTSPIM.

Descripción del problema:

Se solicita el desarrollo de un compilador para una gramática definida en BNF, exte compilador contara con la fase de análisis (léxico, sintáctico y semántico), creación de código en formato tres direcciones y generación de código destino (en el lenguaje MIPS). Todo el proyecto debe de ser implementado en el lenguaje de programación JAVA y se deben de usar las librerías JFLEX y CUP.

Diseño del programa: decisiones de diseño y algoritmos usados:

Este proyecto se desarrolló utilizando grandle y en el archivo build.grandle se configuraron las tareas para la ejecución del proyecto.

Para este tercer proyecto se tomo lo desarrollado en el proyecto 1 y el 2 para continuar con el trabajo. Para las correcciones de los proyectos anteriores fueron mínimas y para la generación de código en MIPS se realizó utilizando la lista de cuads del código de tres direcciones, para realizar la traducción recorremos la lista varias veces para realizar la traducción de distintas partes del programa. Se crearon varias clases para realizar todo el proceso de traducción las cuales son las que recorren toda la lista de cuads.

Librerías usadas: creación de archivo, lexer y parser, etc:

- Java.io: Librería para la entrada y salida.

- Jflex: Para la generación del analizador léxico.

- Cup: Para la generación del analizador sintáctico.

- Librerías para tipos de dato de java: Adicionalmente se usan librería como List o ArrayList para trabajar con tipos de datos en java.

Análisis de resultados: objetivos alcanzados, objetivos no alcanzados, y razones por las cuales no se alcanzaron los objetivos (en caso de haberlos):

- **Objetivos alcanzados:**
    - Análisis léxico, sintáctico y semántico: Si.
    - Generador código en formato tres direcciones.
- **Objetivos no alcanzados:**
    - Traducción a código MIPS: Esta parte no fue alcanzado por completo, faltaron distintas partes como el manejo de recursión y listas. Aunque la generación general de un programa de MIPS es completada correctamente, las funciones, las operaciones de expresiones básicas son logradas correctamente. En si podemos considerar que no logramos alcanzar bien los objetivos debido a que algunos errores con el TAC costaron mas de lo que teníamos considerado.

Justificación de la toma de decisiones:

Para la traducción a código destino se opto por trabajar con el código tres direcciones generado en el proyecto 2, se recorre toda la lista de cuad que se genera. Primero recorremos una vez todos el TAC para obtener todos los datos que vamos a tener en él .data, después de eso se crearon dos funciones para analizar todo el TAC y obtener las funciones que se hallan identificado, cada vez que se identifica una función se registran los parámetros y las variables que van a tener y posteriormente se calcula el tamaño de la pila que se va a ocupar la función y se guarda.

También se creó una clase la cual se encarga de registrar todos los datos relacionados con las funciones para que sea más fácil de procesarlas.

Posteriormente de analizar las funciones, se procede a recorrer otra vez la lista de cuads para que esta vez si se vaya a realizar poco a poco la traducción a código destino a partir del TAC que se tiene. Toda esta parte se realiza mediante estructuras de if para ir reconociendo poco a poco la estructura.

Cabe recalcar que para la parte de las estructuras de control también siguen un proceso similar de traducción.

Lo anterior, aunque bien tal vez no sea la forma más eficiente de realizarlo, era la que mejor se adaptaba a lo que ya teníamos planificado en el proyecto dos.