Reporte

Integrantes

Yasmin Cisneros Cimadevila C411
Jessy Gigato Izquierdo C411

Arquitectura del Compilador

La estructura del compilador la modelamos de la siguiente manera:

- Tokenizador
- Parser (en este se genera el AST)
- Proceso de Semántica:
    - Colector de tipos
    - Constructor de tipos
    - Analisis de dependencia cíclica
    - Chequeo de tipos
- Generación de Código:
    - AST COOL-CIL
    - AST CIL-MIPS
    - Generación del archivo .mips

Tokenización y Parser

Para la construcción del lexer y el parser se utiliza la bilbioteca `ply` de python, que provee herramientas para la generación del AST a partir de una gramática. El método de parseo utilizado fue `LALR`.

Semántica

Del proceso de analisis de la semántica se obtiene un nuevo AST en el que cada una de sus ramas ha adquirido el significado semántico que debe tener, es decir, se asegura de que se cumplan las reglas definidas que asignan un significado lógico a las expresiones dadas por la sintaxis del lenguaje. La evaluación de las reglas semánticas define los valores de los atributos de los nodos del AST para la cadena de entrada. Para el caso de los operadores, cada nodo de este tipo, al terminar el análisis sabe si la operación asociada es aplicable o no. En el colector y constructor de tipos se recopilan los mismos, tanto los existentes en el lenguaje como los creados por las clases definidas en el código de entrada, y por último en el chequeo de tipos se comprueba si en el contexto dado puede ser utilizado.

Generación de Código (Pendiente)

En este punto ya tenemos un AST sintáctica y semánticamente correcto, y que quiere generar un programan en MIPS equivalente al código de entrada que COOL. Como pasar de COOL a MIPS directamente resulta complicado, se pasa primero del AST de COOL a un AST de CIL y luego a partir de este último se construye un AST de MIPS a partir del cual se obtiene un archivo `.mips` listo para ser ejecutado.