Generador de Código Intermedio - Proyecto Final de Compiladores

Portada

Samuel Roberto Acevedo Sandoval -> A01026893 Sebastian Enrique Moncada Gonzales -> A01027028
Curso: Compiladores **Fecha:** 29 Mayo 2025

Introducción

Este proyecto consiste en la implementación de un generador de código intermedio para un lenguaje estilo C-, que traduce programas fuente a código en lenguaje ensamblador **MIPS**. La elección de MIPS se debe a su arquitectura simplificada, ampliamente utilizada en cursos de arquitectura de computadoras y compiladores. Su formato RISC (Reduced Instruction Set Computer) permite un entendimiento claro de la traducción desde estructuras de alto nivel hasta instrucciones de bajo nivel.

El código generado tiene como objetivo ser funcional y comprensible dentro de entornos como **SPIM** o **MARS**, simuladores ampliamente usados en el ámbito académico para la arquitectura MIPS.

Este generador de código se conecta con las etapas anteriores del compilador (análisis léxico, sintáctico y semántico) para convertir un árbol de sintaxis abstracta (AST) en instrucciones MIPS.

Manual del Usuario

A continuación se muestra cómo compilar y ejecutar el generador de código con un programa escrito en C-.

Requisitos:

- Python 3.x
- Archivos previos del compilador:
    - `scanner.py`, `parser.py`, `analyze.py`, `globalTypes.py`, etc.
    - `main.py` (punto de entrada del compilador)
    - `cgen.py` (módulo de generación de código)

Estructura del Proyecto

```
📁 A01026893_CodGen
├── main.py
├── cgen.py
├── scanner.py
├── parser.py
├── analyze.py
```

```
├── globalTypes.py
└── ...otros módulos...
```

Ejemplo paso a paso

Supongamos que tenemos un programa en C- llamado `test.c` con el siguiente código:

```
int main(void) {
  int x;
  int y;
  x = input();
  y = input();
  output(x + y);
  return 0;
}
```

1. Coloca el código en el archivo `test.c`
2. Corre el programa principal:

   ```
   python3 main.py
   ```

3. Esto generará un archivo `output.s` con el código en MIPS
4. Abre el simulador MARS o SPIM y carga `output.s`
5. Corre el programa
6. Introduce los dos números uno por uno en consola cuando se solicite
7. Verás el resultado (la suma) impreso en consola

Capturas de pantalla (opcional si es en documento impreso):

- Compilación en terminal
- Archivo `.s` generado
- Ejecución en SPIM/MARS con entrada/salida

Apéndices

Apéndice A: Primera Entrega (Análisis Léxico)

- Implementación del scanner (`scanner.py`)
- Tabla de tokens y tipos
- Código de funciones auxiliares

Apéndice B: Segunda Entrega (Análisis Sintáctico)

- Implementación del parser (`parser.py`)
- Gramática en BNF/EBNF
- Árbol de derivación generado

Apéndice C: Tercera Entrega (Análisis Semántico)

- Implementación de la tabla de símbolos
- Verificación de tipos y declaraciones
- Mecanismos de manejo de errores semánticos

Apéndice D: Definición del Lenguaje C-

- Gramática completa en BNF
- Lista de palabras reservadas
- Operadores soportados: `+`, `-`, `*`, `/`, `==`, `!=`, `<`, `<=`, `>`, `>=`
- Estructuras válidas: declaraciones, asignaciones, condicionales, loops, funciones
- Funciones incorporadas: `input()`, `output()`

Conclusiones

El generador de código es una de las últimas etapas del compilador y representa un puente crítico entre el análisis semántico y la ejecución real en máquina o simulador. La implementación en MIPS ofrece una representación clara y educativa del proceso de compilación y ejecución, permitiendo una mejor comprensión del comportamiento interno de los lenguajes de programación.