UNIVERSIDAD TECNOLÓGICA CENTROAMERICANA

Compiladores 2 Sec. 1247

Actividad

Proyecto: Compilador Ada 95

Alumnos

Harold Rodrigo Mendoza Raudales – 11541045

José Ricardo Fernández Ortega – 11541162

CATEDRÁTICO

Ing. Carlos Enrique Vallejo

FECHA

24/06/2019

LUGAR

Campus Tegucigalpa

Técnicas de Programación y Problemas Encontrados

Para la elaboración de este proyecto, se requirió aprender a utilizar nuevos programas y técnicas de programación que facilitan la creación de nuestro compilador. El proyecto comenzó con la elaboración de un archivo de extensión **.flex**, en el cual se definen todos los tokens y las expresiones regulares que va a requerir el programa.

Una vez tokenizada toda la información, la pasamos a nuestro archivo conocido como **Parser.cup**, el cual es el que contiene todas las gramáticas de nuestro programa. Dentro de este archivo es donde se lleva a cabo mayor parte de la programación del compilador, ya que los valores constantemente se mueven a través de las gramáticas.

Es en este archivo además donde se genera un árbol sintáctico abstracto, el cual una vez que se ejecuta el programa nos muestra, en forma de un árbol, la estructura de nuestro programa. Este árbol debe contener poca información, principalmente la información que vaya a ser utilizada por otros nodos.

Como en cualquier lenguaje de programación se pueden crear variables y otras estructuras de programación, debemos de encontrar una manera para tener control sobre estas variables. Para esto se utiliza una técnica en la que se elabora una tabla con toda la información de todas las variables del programa, esta se conoce como **Tabla de Símbolos**. Esta tabla fue muy útil en el desarrollo del compilador, ya que con esta se pueden validar errores, como si una variable no fue declarada previamente y se está utilizando, etcétera.

Además de una tabla de símbolos, también requerimos la ayuda de otra tabla conocida como **Tabla de Cuádruplos**. Esta tabla lo que hace es ordenar cada línea del programa de tal manera que se pueda generar código de tres direcciones, para facilitarnos también la generación del código intermedio y final.

Teniendo la tabla de cuádruplos lista pasamos a la generación del **código intermedio**, el cual por medio de la tabla de cuádruplos se genera como código de tres direcciones. Teniendo completo el código intermedio comenzamos a generar el **código final**, el cual nos muestra un programa de extensión **.asm** que puede ser ejecutado en **QTSpim**, el programa que fue utilizado para la ejecución de los programas en MIPS.

Al momento de ejecutar el programa en la consola de comandos, se nos presentaba un problema que decía "**Code Too Long"**, como su nombre lo explica se debía a que nuestro código en el parser.cup era demasiado largo. La solución que implementamos para este problema fue la creación de más clases **.java** que tuvieran métodos para correr los códigos más largos dentro del parser.cup.

También se nos presentaban problemas de **shift-reduce**, esto se debía a ambigüedad en la gramática, y se solucionaba simplemente cambiando la estructura de la gramática de tal manera que esta ya no causara ambigüedad.