Facultad de Ingeniería
Semestre 2020-2
Compiladores
Proyecto Final

Alejandro Hernández Rodríguez

Objetivo

El objetivo del proyecto es desarrollar un compilador del lenguaje C. Para poder desarrollarlo, se siguió las instrucciones de la página oficial de Nora Sandler. En la página se mencionan los aspecto más importantes de un compilador y como desarrollarlos. Además de los conocimientos en ensamblador necesarios para poder traducir el lenguaje de alto nivel a bajo nivel.

El proyecto está dividido en diferentes etapas, cada una presenta un nivel diferente de dificultad y aspectos necesarios para la correcta implementación del compilador.

Stage 1

En la primera etapa se ven los aspectos más básicos del compilador para la manipulación de enteros, con esto tenemos que definir ya la función main de un programa en C. Poder realizar el análisis léxico del lenguaje, el parseo de los tokens generados por el lexer, el árbol de sintaxis abstracto generado.

Stage 2

En la segunda etapa compilamos los operadores unarios, y vimos cómo se representaban en el árbol de sintaxis abstracta.

Stage 3

En la tercera etapa compilamos operadores binarios, el reto fue en modificar el generador de código y agregar otro tipo de nodo al árbol AST.

Stage 4

La cuarta etapa es muy parecida a la tercera, únicamente agregamos operadores lógico binarios, como AND, OR, etc.

Descripción

El compilador está compuesto por cuatro programas, el parser, el lexer, el árbol AST y el generador de código. Cada uno es una pieza importante en el proceso de compilación del archivo.

Archivo.c → Lexer → Parser → AST → Generador de código → Archivo.s

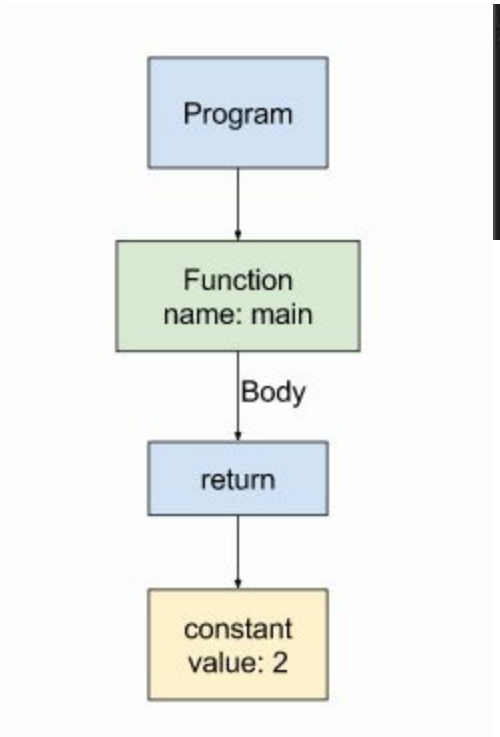
Lexer

El lexer es el encargado del análisis sintáctico del texto. Define y clasifica las palabras reservadas del lenguaje. En general se encarga de "Tokenizar" el código para después poder generar el AST.

Parser

El parser es el encargado de convertir la cadena de tokens generada por el lexer para la creación de un árbol de sintaxis abstracta. Esto lo hace definiendo que tipo de nodo generar a partir cada token.

AST




```
Árbol de Sintaxis Abstracta
 program :  valid/return_2.c
|--->(izquierdo) function :  main
   |--->(izquierdo) return :  return
      |--->(izquierdo) constant :  2
```

El Abstract Syntax Tree es generado en el parseo del código. Representa la jerarquía de expresiones y operaciones en el código. Esta representación es la que utilizaremos para generar el código ensamblador. En la izquierda se observa el árbol del blog de Nora y en la parte de arriba generado por el compilador.

Cada nodo tiene los atributos tipo, padre, hijo izquierdo, hijo derecho, dato.
El atributo tipo es el tipo de token que representa constant, unaryop, etc.
El atributo padre referencia al padre del nodo.
El atributo dato referencia el token leído.

Generador de Código

Por último y no menos importante está el generador de código, la tarea de este será recorrer el AST e ir generando el código ensamblador correspondiente con cada nodo del árbol.