UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ingenieria

Division de Ingeniería Eléctrica

Departamento de Ingeniería en Computación

Proyecto

Diseño e implementacion de un Compilador

Grupo 4

Semestre 2020-2

COMPILADORES

Profesor: Ing. Norberto Jesús Ortigoza Márquez

Integrantes:
Palacios Flores, Gonzalo Emilio
Ramírez Rivera, Giezy Alberto
Sandoval Juárez, Luis Arturo
Vivanco Quintanar, Diego Armando

08 de junio de 2020

1. Objetivo

El alumno pondrá en práctica los conocimientos adquiridos de cada uno de los temas del curso de Compiladores para diseñar e implementar un compilador C basado en el diseño de un compilador desarrollado por Nora sandler.

2. Introducción

El desarrollo del proyecto no solo se enfoca en la parte técnica que es la implementación del compilador (cuyos requerimientos se abordan en el siguiente apartado) sino también involucra el uso de conocimientos adquiridos en otras asignaturas de la carrera como lo son Ingeniería de Software, Estructura y Programación de Computadoras, Lenguajes Formales y Autómatas y Sistemas Operativos principalmente.

Pero antes de abordar el desarrollo de nuestro proyecto primero veamos algunas definiciones y conceptos.

2.1. Compilador

Un compilador es un programa que convierte o traduce el código fuente de un programa hecho en lenguaje de alto nivel, a un lenguaje de bajo nivel (lenguaje de máquina). En pocas palabras, es un software que se encarga de traducir el programa hecho en lenguaje de programación, a un lenguaje de máquina que pueda ser comprendido por el equipo y pueda ser procesado o ejecutado por este.

Figura 1: Fases del funcionamiento de un compilador.

Los compiladores son procesos complejos debido a que tienen varias fases por las que un programa fuente debe de pasar antes de convertirse en un programa ejecutable, los pasos son los siguientes:

- **Analizador léxico:**
  El analizador léxico o lexicográfico (Scanner en inglés) es la primera etapa del proceso de compilación, el cual se encarga de dividir el programa en Tokens, los cuales, según una tabla de símbolos definida por el mismo lenguaje.

  De esta forma cada token del programa es clasificado según su significado para ser procesados en la segunda etapa del proceso de compilación.

- **Analizador sintáctico:**
  El analizador sintáctico (Parse en inglés), es la segunda fase del proceso de compilación y tiene como finalidad la generación de un Árbol sintáctico, el cual no es más que una estructura de datos compleja que permite representar de una forma más simple al programa fuente.

  Los compiladores modernos utilizan estructuras de objetos para representa a un programa, de esta forma existe una clase específica para representa cada posible token de nuestra tabla de símbolos.

- **Analizador semántico:**
  El analizador semántico es el último paso antes de empezar a compilar realmente el código, prepara el programa para ser compilado. El analizador semántico parte del árbol sintáctico abstracto y tiene la finalidad de validar los puntos más finos del programa, como por ejemplo, validar compatibilidad en tipos de datos, que la variable utilizada en una instrucción este previamente declara o que estén dentro del contexto, si implementamos una interface que todos los métodos estén definidos, etc.

  El analizador semántico es el que analiza que todo el programa tenga un significado exacto y que este no pueda fallar en tiempo de ejecución.

En la figura 3 podemos ver el esquema donde se relacionan cada una de las fases anteriormente mencionadas.

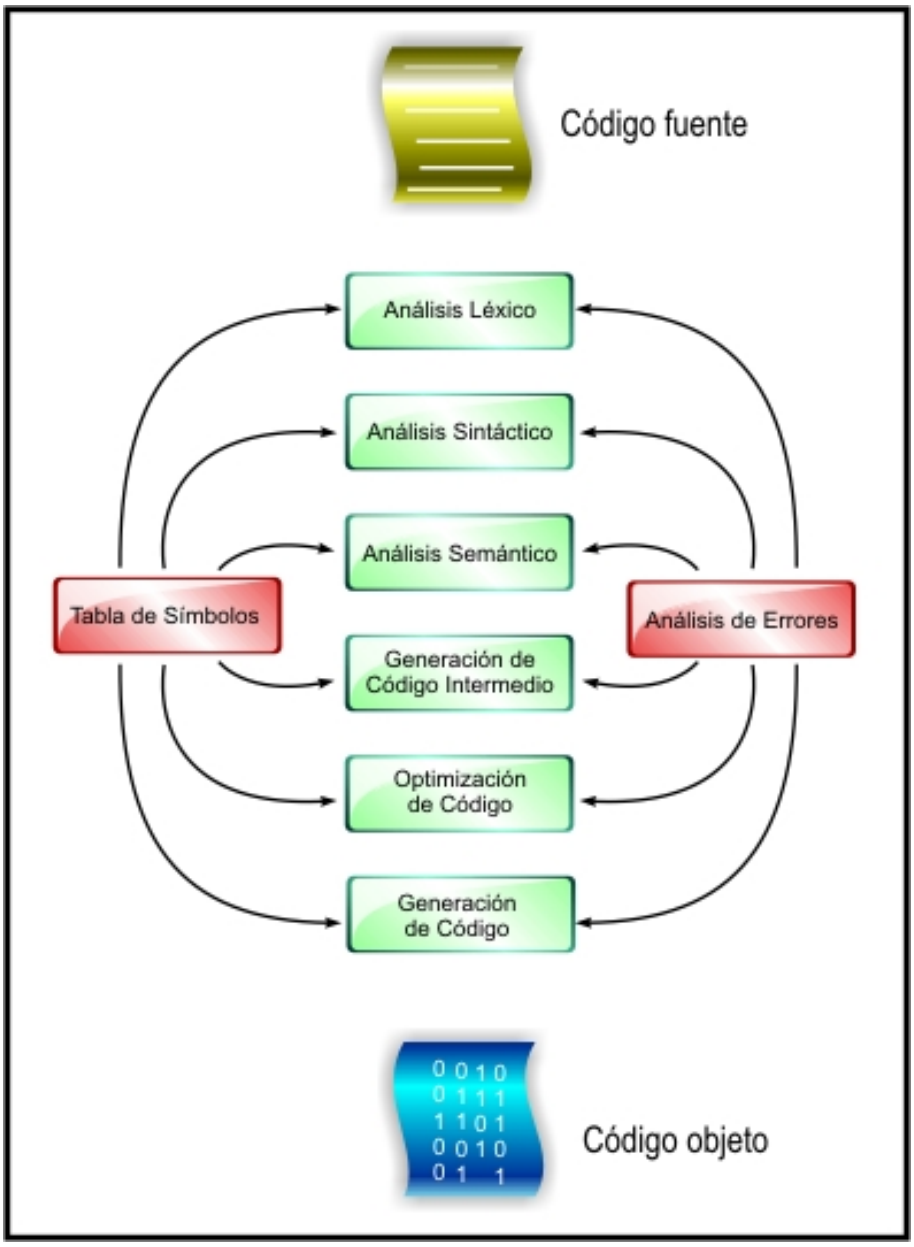

Figura 2: Esquema de las fases de un compilador.

2.2. Elixir

Elixir es un lenguaje de programación dinámico y funcional, diseñado para crear aplicaciones escalables y mantenibles. Elixir fue creado en el año 2012 por José Valim. En comparación con otros lenguajes, Elixir es una tecnología bastante nueva, sin embargo, esto no debería engañarnos, que sea un lenguaje nuevo no significa que sea frágil o que existan bugs. De hecho, Elixir se distingue por ser estable y robusto, principalmente porque los programas creados en este lenguaje se ejecutan en la máquina virtual de Erlang (BEAM).

Cuando nos encontremos desarrollando en Elixir lo haremos utilizando el paradigma de programación funcional, dejaremos a un lado el tema de objetos y nos concentraremos principalmente en funciones y módulos.

3. Requerimientos

El proyecto se presentará en 4 entregas, cumpliendo con lo siguiente:

- El compilador debe ser del lenguaje de programación C.
- En la primer entrega el compilador deberá estructurarse con los módulos correspondientes (Lexer, Parser, Generador de Código, etc), deberá reconocer todos los caracteres que soporta el Lenguaje C.
- En la segunda entrega el compilador deberá soportar los operadores unarios: **Negation (-), Bitwise complement (∼)** y **Logical negation (!)**.
- En la tercera entrega el compilador deberá soportar los operaciones de suma, resta, multiplicación y división, deberá de poder mostrar el resultado de dichas operaciones.
- En una cuarta entrega el compilador reconocerá otros operadores binarios (**&&, ||, ==, !=,** $<$, $<=$, $>=$, $>$).
- El compilador se llevará a cabo en el lenguaje de programación Elixir.
- Para desarrollar el proyecto, se contempla una arquitectura de 64 bits.
- Con la implementación realizada se podrá generar el código en ensamblador, además de un binario ejecutable en un ambiente Windows, aunque se harán las modificaciones pertinentes para que el compilador pueda ser utilizado en un ambiente Mac.

4. Plan de Trabajo

4.1. Fechas Importantes

Actividad	Fecha
Primera entrega del Proyecto	16-27 marzo 2020
Segunda entrega del Proyecto	13-24 abril 2020
Tercer entrega del Proyecto	26-29 mayo 2020
Cuarta entrega del Proyecto	15-19 junio 2020

Tabla 1: Fechas para las actividades del curso en el semestre 2020-2.

4.2. Roles de los integrantes del Equipo

Rol	Integrantes
Project Manager	Sandoval Juárez, Luis Arturo
Tester	Ramírez Rivera, Giezy Alberto
Architec	Palacios Flores, Gonzalo Emilio
Integrator	Vivanco Quintanar Diego Armando

Tabla 2: Roles de cada integrante del equipo.

4.3. Actividades Generales

A continuación, se plantean las actividades generales que se llevarán a cabo para el desarrollo del proyecto:

- Se tendrán juntas de trabajo 3 veces por semana donde se discutirá el avance del proyecto de cada uno de los integrantes del equipo, se discutirá el plan de trabajo para la siguiente semana.
- Se verificará que el avance individual de los integrantes esté acorde con lo planteado en el cronograma general para cada entrega.
- Se hará una revisión de todos los cambios realizados al repositorio del equipo.
- Se discutirán dudas y sugerencias acerca de las tareas asignadas.
- Se discutirá el avance del proyecto con el experto de dominio y cliente.

4.4. Registro de Actividades

Task ID	Task	Details	Assigned	Start	Due
		PRIMERA ENTREGA			
1	Requerimientos e investigacion		PM, Architect, Tester,	17-feb-20	21-feb-20
2	sanitazer		Proyect Manager	21-feb-20	22-feb-20
3	NQCC		PM, Architect, Tester	22-feb-20	25-feb-20
4	Lexer		Tester	25-feb-20	29-feb-20
5	Parser		Integrator	29-feb-20	08-mar-20
6	Generador de Código		Architect	08-mar-20	13-mar-20
7	Pruebas		Tester	29-feb-20	13-mar-20
		SEGUNDA ENTREGA			
8	Lexer		Proyect Manager	22-mar-20	25-mar-20
9	Parser		Integrator	18-mar-20	22-mar-20
10	Generador de Código		Architect	22-mar-20	09-abr-20
11	Pruebas		Tester	09-abr-20	14-abr-20
		TERCERA ENTREGA			
12	Lexer		Proyect Manager	22-abr-20	28-abr-20
13	Parser		Integrator	28-abr-20	10-may-20
14	Generador de Código	Problemas en inicio de codigo ensamblador "nombre" linux	Architect	10-may-20	21-may-20
15	Pruebas		Pruebas	17-may-20	28-may-20
		CUARTA ENTREGA			
16	Lexer		Proyect Manager	28-may-20	01-jun-20
17	Parser		Integrator	28-may-20	28-may-20
18	Counter		Tester	28-may-20	29-may-20
19	Generador de Código	Errores en los registros de 64 bits	Architect	28-may-20	16-jun-20
20	Pruebas		Tester	05-jun-20	15-jun-20

Figura 3: Registro de las actividades al momento de realizar el compilador.

5. Arquitectura

La estructura general del compilador está fundamentada en las recomendaciones del tutorial de Nora Sandler. Por lo anterior el compilador cuenta con cuatro módulos fundamentales, estos son: Lexer, Parser, Code Generator, Linker. En la figura 4 se puede observar la arquitectura de nuestro compilador.

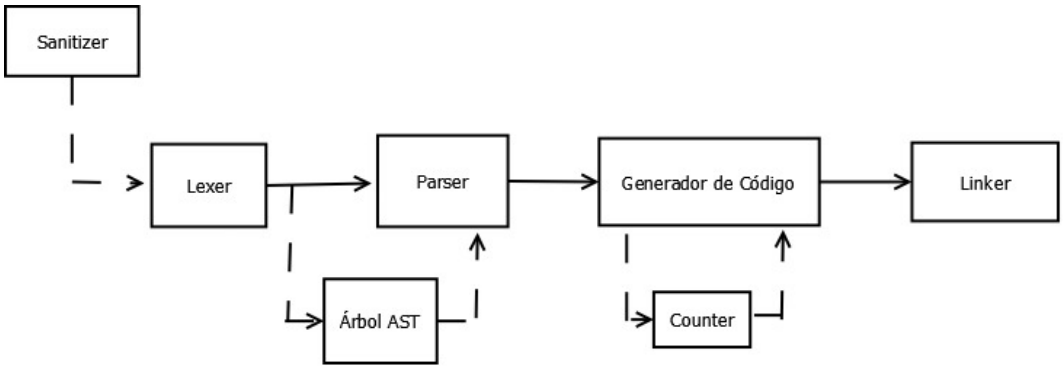

Figura 4: Modulos que conforman el compilador.

- **Lexer**
  Este módulo convierte el texto en una lista de tokens con información suficiente para indicar la línea y el carácter compuesto el cual es necesario para lanzar los errores mediante un raise.

- **Parser**

  Además se utiliza el algoritmo recursivo descendente para que, por cada derivación, haya una función asociada y por lo tanto se pueda validar; y tener una mejor estructura al momento de recorrer la lista de tokens. Al igual que en el lexer, el manejo de errores se hace mediante excepciones. Genera un árbol AST el cual contiene la estructura para que el árbol se pueda recorrer y mantenga la semántica del programa.

- **Árbol AST**
  Provee el molde para generar el árbol AST.Posteriormene el Parser lo manda a llamar para seguir estructurando el árbol completo.

- **Generador de Código:**

  Se utiliza el algoritmo de pos orden para recorrer el árbol y recoger primero las operaciones y expresiones de menor precedencia y así ir acomodando todo el código conforme debe ser procesado.

- **Counter**

  Genera el número de etiqueta que ayudara en el modulo de Generación de Código.

6. Pruebas

Para la realización de las pruebas se utilizó el código provisto por Nora Sandler para el testeo del proyecto, dichas pruebas se realizaron de maneara automática y se verifico que el compilador se comportara de la forma esperada para casos en donde la sintaxis y gramática del programa fueran correctas y casos en donde esta condición no se satisficiere.

7. Conclusiones

- **Palacios Flores, Gonzalo Emilio**
  En este proyecto observamos como funciona un compilador, sus partes, complejidad, construcción, lógica, funciones así como aprender un nuevo lenguaje de programación para tal implementación. Este proyecto fue el más complejo que he hecho a lo largo de la carrera ya que aplique conocimientos de varias materias para que pudieramos implementar tal proyecto, observamos que se necesita verdaderamente de una organización tanto para aprender un nuevo lenguaje distinto, con una lógica diferente que hace que poder implementarlo sea un poco más dificíl, una comunicación aún mas estricta ya que podemos extrapolar este proyecto como a un sistema que podriamos implementar en un trabajo real tanto en programación como en documentación, trabajar en equipo siendo este la parte más compleja debido a que muchas veces es dificil comunicar tus ideas, investigar sobre un compilador, aprender la forma y estructura para la implementación y sobre todo no dejar de lado ninguna idea, en muchas ocaciones por quedarnos una idea teniamos errores que muchas veces no supimos como resolver, esto hizo que tuvieramos que regresar pasos atrás para poder corregir problemas tanto en el equipo como en el proyecto. Este proyecto fue en verdad una de las mejores experiencia que he vivido en mi carrera universitaria, tuve bastantes experiencias gratificantes para las cuales me dio gusto tener.

- **Ramírez Rivera, Giezy Alberto**
  Con el uso de las diferentes pruebas implementadas a lo largo del desarrollo del proyecto nos dimos cuenta de la importancia de automatizar pruebas. No sólo nos una visión más amplia del porcentaje desarrollado del proyecto sino que facilita mucho la detección de errores para su pronta solución. En diversas ocasiones pudimos notar errores con el uso de las pruebas, errores que no hubiéramos sido capaces de detectar de no ser por ellas ya que el resto de casos funcionaban bien. Las pruebas también nos ayudaron a detectar más fácilmente el origen de los errores ya que nos muestran la salida esperada contra la real, permitiéndonos observar si hay algún error menor como un tabulador de más en una cadena o un error de lógica. Consideramos importante mencionar la diferencia de las pruebas unitarias y las pruebas de integración ya que ambas son importantes. Mientras que las pruebas unitarias nos permitieron comprobar que cada módulo funcionara de la forma que se espera, las pruebas de integración nos permitieron comprobar que la modificación de un módulo no afectara el siguiente.

- **Sandoval Juárez, Luis Arturo** En lo personal el ambiente del proyecto se me hizo algo familiar ya que he trabajado en muchos proyectos de esta índole, por lo que el rol de Project Manager no me fue una tarea difícil, además de que mis compañeros fueron muy solidarios en la realización del proyecto. La parte de la programación y del uso de Elixir me costaron un poco de trabajo al principio, pero conforme el proyecto fue avanzando, poco a poco fui entendiendo el proceso de crear los módulos y de relacionarlos para el funcionamiento del Compilador. Como conclusión final puedo decir que el trabajo en equipo es clave en el resultado final del proyecto, si se tiene una buena organización el proyecto puede salir si bien no de una manera muy fácil el proceso si puede llegar a ser un poco menos agobiante o incómodo.

- **Vivanco Quintanar, Diego Armando**

  En general el proyecto, aunque pareciera que fue sencillo involucro muchas cosas, desde los conceptos de aboles, hasta los de gramáticas vistos en cursos anteriores, así como el de otros conceptos en la planeación de proyectos, organización y el análisis de requerimientos que los clientes demandan en algún proyecto. El trabajo en equipo fue un factor muy importante al momento de llevar a cabo la construcción del proyecto, nos dimos cuenta por ejemplo que, en cualquier proyecto, las habilidades de cada integrante del equipo contribuyen al desarrollo de ciertas partes o módulos del trabajo que en este casi fueron los módulos del Compilador.

  En el equipo un compañero tenia mas habilidad y experiencia con el ensamblador, mientras que otra tenía experiencia en la organización y el dirigir a personas en un proyecto, así pues, algún otro compañero tenia mas habilidad para la programación e ir aprendiendo de manera muy rápida la sintaxis del nuevo lenguaje, que en este caso es Elixir. Sin duda el proyecto me dejo varias lecciones, las cuales son las de aprender un lenguaje nuevo de programación y no solo quedarme con lo que la Facultad nos proporciona, también me hizo ver que debo reforzar ciertos conocimientos como ensamblador y las gramáticas vistas en cursos anteriores, también aprendí la importancia de tener una buena comunicación con un equipo y la manera de ver que las capacidades de cada uno de los integrantes sean explotadas al máximo para tener un resultado eficaz en la finalización de un proyecto.

8.

Referencias

[1] Ruiz Catalan, J. (2010). Compiladores. San Fernando de Henares, Madrid: RC libros.

[2] AHO, Alfredo, SETH, Ravi, et al. Compiladores. Principios, técnicas y herramientas. Addison-Wesley Iberoamerica, 2000.

[3] TREMBLAY, Jean-paul. SORENSON, Paul The Theory and Practice of Compiler Writing Mac, Graw-Hill, 1985.

9. Apéndice

En el siguiente link se encuentra nuestro repositorio con todos los archivos y documentación del Compilador.
https://github.com/hiphoox/c202-ramex.git

10. Anexo

10.1. WBS

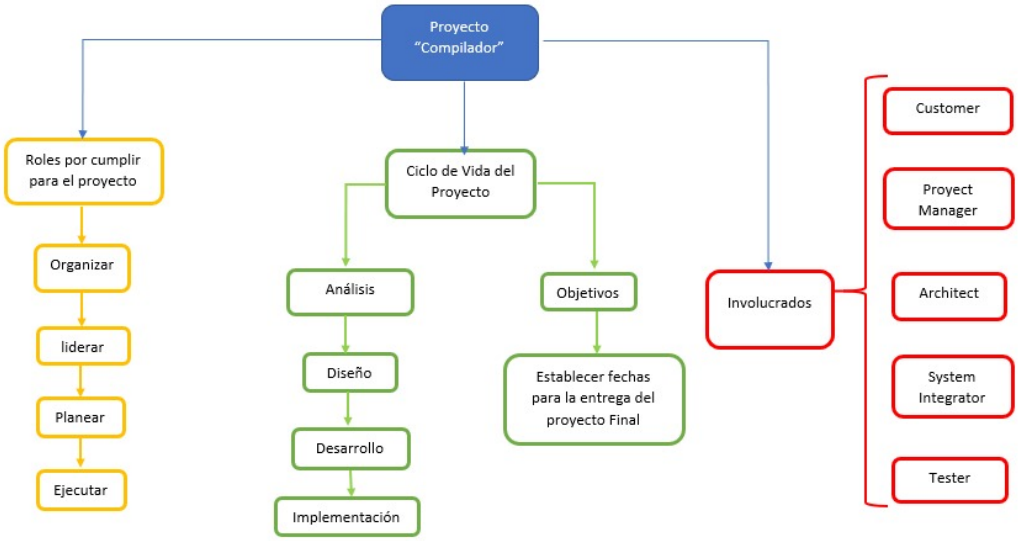

Figura 5: Mapa conceptual del BWS.