UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

MATERIA: COMPILADORES

GRUPO: 3

PROFESOR: ING. NORBERTO JESUS ORTIGOZA MARQUEZ

MATERIA: COMPILADORES

SEMESTRE 2020-1

EQUIPO: CODEX

PROJECT MANAGER - MÁRTINEZ LÓPEZ ÁNDRES

SYSTEM ARCHITECT - ROMERO BASURTO FABIAN

SYSTEM TESTER - PONCE SEDANO JESUS ALEJANDRO

SYSTEM INTEGRATOR - LÓPEZ ALEXIS URIEL

Índice

INTRODUCCIÓN

Un compilador es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación.

La construcción de un compilador involucra la división del proceso. Generalmente estas fases se agrupan en dos tareas: el análisis del programa fuente, siendo el Front-end y la síntesis del programa objeto siendo el back-end.

- *Front-end*: es la parte que analiza el código fuente, comprueba su validez, genera el árbol de derivación, añadiendo a estos, están las fases comprendidas entre el análisis Léxico y la generación de código intermedio.
- *Back-end*: es la parte que genera el código máquina, específico de una plataforma, a partir de los resultados del front-end.

Esta división permite que el mismo *Back End* se utilice para generar el código máquina de varios lenguajes de programación distintos y que el mismo *Front End* que sirve para analizar el código fuente de un lenguaje de programación concreto sirva para generar código máquina en varias plataformas distintas.

REQUERIMIENTOS

1. Compilar un archivo con extensión “.c“ con contenido:

```
int main( ) {
return <entero>;
}
```

Y su salida debe ser un ejecutable.

2. Se debe ejecutar desde la línea de comandos o terminal para sistemas unix.

3. El ejecutable debe tener el mismo nombre que el archivo compilado, debe estar en la misma ruta que se compiló.

4. Con la bandera “-o <nombre_ejecutable>” se debe poder modificar el nombre del ejecutable por uno diferente al del archivo original. El nombre del ejecutable por default es el nombre del archivo original.

5. La bandera “-s” genera el código de ensamblador del archivo compilado y no genera un ejecutable.

6. La bandera “-t” devuelve la lista de tokens del archivo compilado.

7. La bandera “-a” devuelve el árbol sintactico abstracto AST del archivo compilado.

8. La bandera “-h” devuelve las opciones del compilador.

9. Detectar elementos faltantes o errores.

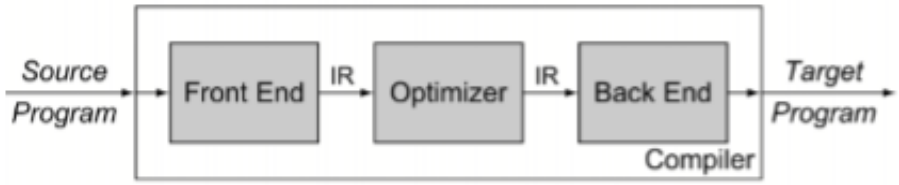
10. Se entregará un manual para compilar el programa, además de otro manual de como correr las pruebas de los ejemplos del git de Nora Sandler.

11.Trabajar el control de versiones en la plataforma Github.

ARQUITECTURA

El objetivo de los compiladores es el preservar el significado del programa que se está compilando, siendo este un aspecto semántico. Además, de qué si es posible, mejorar el programa de entrada de modo que la salida del programa sea lo más eficiente posible, tomando en cuenta todas las herramientas del entorno.

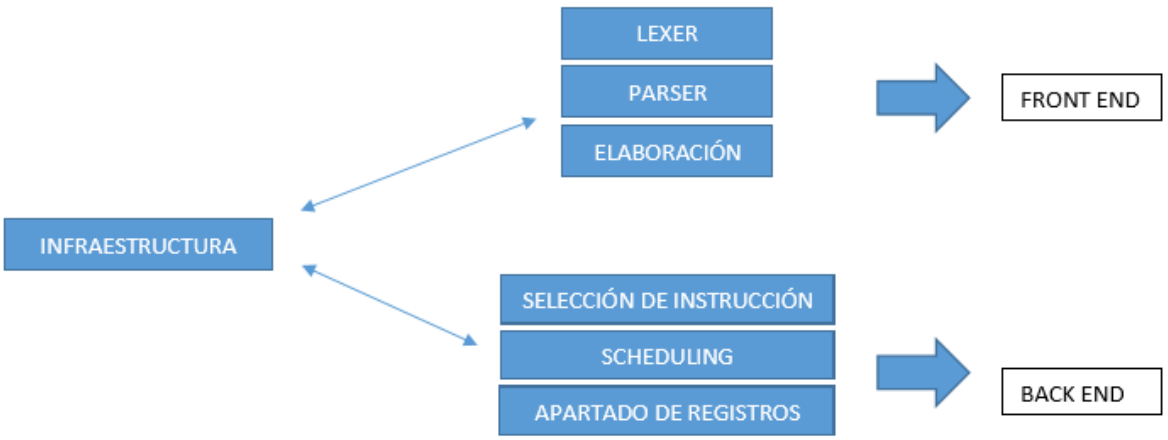
Siendo estos los principios de los compiladores, la estructura de nuestro compilador es la siguiente.



La estructura de nuestro compilador se basa en dos procesos principalmente, Front End y Back End.

La tarea principal del Front End es el poder leer el programa, checar la sintaxis o las reglas que debe seguir el programa y checar la semántica o el significado del mismo.

La tarea del Back End es ser el encargado de poder comunicarse con el ensamblador o el sistema de modo que pueda comunicarse con la computadora.

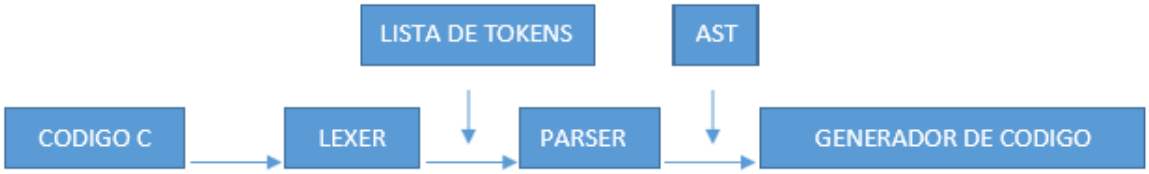


IMPLEMENTACIÓN

La implementación de nuestro compilador se basa en el desarrollo estándar de un compilador, es decir, análisis léxico en el Lexer, el cual no le importará el orden del programa, solamente checará los componentes léxicos los cuales son un conjunto de caracteres que tienen un significado coherente en cierto lenguaje de programación (tokens), este apartado nos devolverá una lista de tokens.

La lista que nos devuelve el lexer debe ser analizada en el Parser, en donde se iniciará un proceso sintáctico, basándose en un árbol de tipo AST (Abstract Syntax Trees), se escogió este tipo estructura ya que nos sirven para manejar información semántica de un código, además que al elegir esta estructura de árbol, en posteriores entregas, no se va a tendrá que modificar mucho el código debido a que los flujos de información serán la misma, ya que se basa en una relación jerárquica.

Finalmente, la Generación de Código, la cual se va a encargar de generar un código de ensamblador y el encargado de generar un archivo ejecutable de ensamblador es el linker.



ROAD MAP.

● 20 Septiembre - Enteros (Primera entrega)

Para esta primera entrega el compilador, la capacidad de este es limitada y solo puede leer en el “return” con un valor positivo.

● 11 Octubre – Operadores Unarios (segunda entrega).

En esta entrega el compilador debe ser capaz de leer en el “return” valores positivos o negativos y operadores unarios, como Bitwise (~) y la negación lógica (!).

● 8 Noviembre – Operadores Binarios (Tercera entrega)

El compilador debe ser capaz de leer y hacer operaciones básicas con operadores binarios, como:

- Adición “+”
- Sustracción “-”
- Multiplicación “*”
- División “/”

● 29 Noviembre – Operaciones Complejas (Entrega Final)

El compilador debe leer operaciones más complejas en el “return”:

- Lógica AND ‘&&’
- Lógica OR ‘||’
- Igual que ‘==’
- Diferente que ‘!=’
- Menor que ‘<’
- Menor o igual que ‘<=’
- Mayor que ‘>’
- Mayor o igual que ‘>=’.

Ejemplo:

```
int main( ) {
        return 3+4 <= 4 || 1&&2 != 3 > -6;
}
```

PLAN DE TRABAJO.

● Primera entrega

En esta entrega se reunió el equipo completo (Project Manager, System Architect, System Integrator and System Tester) para analizar las especificaciones del programa, la arquitectura, requerimientos, además del sistema operativo en el cual deberá ser ejecutado el programa, también la manera en la que se va a dividir el trabajo para que se cumplan los objetivos en tiempo y forma.

Semana del 11 al 17 de agosto

- Instalar el software necesario para llevar a cabo el proyecto.
- Sentar alguna de las bases para nuestro compilador.

Semana del 18 al 24 de agosto

- Concretar el plan de desarrollo del proyecto para la 1° entrega.
- Dividir por bloques de manera adecuada nuestro compilador (System Arquitect).
- Conocer mejor los detalles que integrara el compilador.

Semana del 25 al 31 de agosto

- Planificar la forma adecuada de llevar el proyecto con el fin de poder integrar la parte dos del programa (System Integrator).
- Planificar e idear pruebas para comprobar la primera parte de nuestro compilador (Tester).
- Comprobar que la forma en que se está planeado es adecuada y de acuerdo a lo solicitado por el profesor (Project Manager).

Semana del 1 al 7 de Septiembre

- Programar en Elixir con el formato adecuado para cumplir con los requerimientos solicitados.
- Asegurarse de que se esté llevando a cabo para la arquitectura indicada, así como para el sistema operativo. (System Architect).
- Verificar que la modulación del programa es eficaz de tal manera que permita su mejora e innovación posterior (System Integrator).
- Contar con los casos de prueba bien estructurados y detallados (Tester).
- Corroborar con el profesor si se están cumpliendo los requerimientos o si se necesita corregir (Project Manager).

Semana del 8 al 13 de Septiembre

- Verificar el funcionamiento correcto de cada una de las partes de nuestro proyecto (System Integrator).
- Comprobar y documentar el por qué sí se está llevando a cabo para la arquitectura y sistema operativo deseado (System Archiect).
- Aplicar las pruebas para cada uno de los módulos y de manera conjunta (Tester).
- Cuantificar que tan eficaz es nuestro programa hasta el momento (Tester)
- Verificar que se ha cumplido con cada uno de los requerimientos del proyecto, así como las diferentes versiones del programa en Github (Project Manager).

Semana del 14 al 20 de Septiembre

- Corregir los errores encontrados y optimizar algunas partes del código.
- Documentar el manual de usuario para la correcta ejecución de nuestro compilador hasta el momento.
- Realizar la presentación para la primera entrega.
- Corroborar el funcionamiento de cada una de las funciones del código.

● Segunda entrega

El equipo se reunirá para analizar los nuevos requerimientos de la entrega, de manera que la implementación sea la más eficaz posible, así mismo diseñar como se integraran los nuevos requerimientos (operadores unarios) de tal forma que no involucre un problema en el desarrollo del código.

Semana del 21 al 27 de Septiembre

- Diseñar la manera correcta de implementar los cambios solicitados para el compilador (System Integrator).
- Verificar que los cambios planeados sigan cumpliendo con los requerimientos de la arquitectura planteada (System Architect).
- Corroborar que lo planificado cumpla con los nuevos requerimientos y que no pongan en riesgo el código realizado anteriormente (Project Manager – System Integrator)

Semana del 28 de Septiembre al 04 de Octubre

- Programar los cambios planificados en el Lexer, Parser y Generador de código.
- Diseñar e implementar las pruebas para que los nuevos requerimientos sean puestos a prueba (Tester)
- Verificar que lo implementado hasta el momento sea lo que el profesor está pidiendo (Project Manager).

Semana del 05 al 11 de Octubre

- Corregir los posibles errores generados.
- Asegurarse que las pruebas hayan sido exitosas (Tester).
- Verificar que el compilador funcione con normalidad en el sistema operativo UNIX (System Architect).
- Verificar que lo codificado en la entrega anterior no se haya visto afectado de manera negativa (System Integrator).
- Corroborar que lo implementado hasta el momento cumpla con los requerimientos solicitados para esta entrega y actualizar la documentación para la correcta ejecución del compilador (Project Manager).

● Tercera entrega

En esta entrega se analizará cómo es que al ir aumentando los requerimientos el sistema seguirá comportándose de buena manera, y en otro caso, ver en qué aspectos se ve afectado el código y plantear la solución a los nuevos requerimientos solicitados y profundizar en la forma que se está implementando el árbol AST.

Semana del 12 al 18 de Octubre

- Planificar y plantear los cambios solicitados para esta entrega.
- Evaluar lo que se ha hecho en las dos últimas entregas (Project Manager).

- Analizar si el Lexer, el Parser y el Generador de código pueden ser más eficiente (Project Manager).
- Evaluar si el árbol AST se está llevando de la manera adecuada hasta el momento (System Architect).
- Conocer a detalle los nuevos requerimientos y concretar los puntos de vista generados a partir de las primeras dos entregas (Project Manager).

Semana del 19 al 25 de Octubre

- Programar los cambios planificados para la tercera entrega.
- Mejorar el Lexer, Parser y Generador de código lo más posible para hacer eficiente al compilador.
- Mejorar o implementar los cambios pertinentes para el árbol AST (System Integrator - System Architect).
- Planear las pruebas necesarias para ejecutar los nuevos requerimientos (Tester).

Semana del 26 de Octubre al 01 de Noviembre

- Verificar con el profesor que se está haciendo lo correcto según lo planteado (Project Manager).
- Verificar que toda la estructura del código desde la primera entrega hasta este momento se esté llevando de manera correcta (System Integrator).
- Corroborar que las pruebas se estén ejecutando con éxito (Tester).
- Comprobar y analizar los errores (Tester).

Semana del 02 al 08 de Noviembre

- Corregir los errores encontrados.
- Actualizar la documentación para el correcto funcionamiento del compilador (Project Manager).
- Verificar que se está llevando a cabo para la arquitectura solicitada(System Architect).
- Verificar la integridad de todas las partes del código (System Integrator).

● Entrega final

En esta entrega se verá en funcionamiento el sistema con todos los requerimientos solicitados por el cliente de manera correcta y algunas nuevas especificaciones.

Semana del 09 al 15 de Noviembre

- Evaluar los puntos a mejorar de las entregas pasadas (Project Manager).
- Diseñar la implementación de los nuevos requerimientos.
- Analizar y concluir el por qué se fue implementado de esa manera el compilador (System Architect).

Semana del 16 al 22 de Noviembre

- Programar los cambios planificados.
- Diseñar las pruebas para ejecutar lo nuevo implementado (Tester).
- Corroborar que se ha estado cumpliendo con lo solicitado por el profesor (Project Manager).
- Aclarar cualquier duda restante sobre los requerimientos del compilador (Project Manager).
- Entender en su totalidad el funcionamiento y el motivo por el cual se realizó de esa forma el compilador

Semana del 23 al 29 de Noviembre

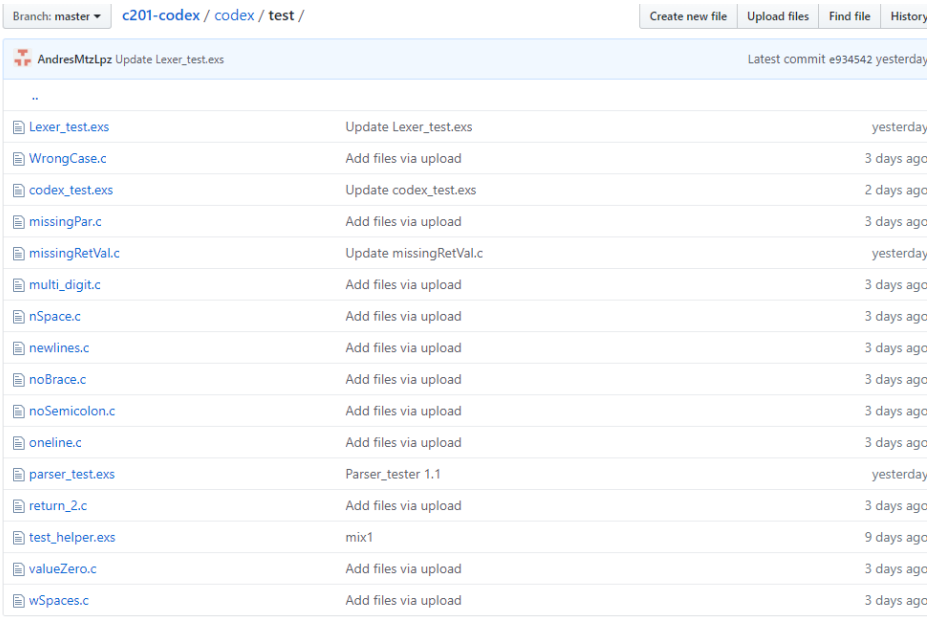
- Ejecutar las pruebas planeadas (Tester).
- Verificar la integridad del código desde la primera entrega (System Integrator).
- Corroborar que en el repositorio de GitHub se encuentre el historial de lo que se ha venido realizado (System Integrator).
- Actualizar la documentación para la entrega final (Project Manager).
- Checar que se esté cumpliendo con todos los requerimientos solicitados por el profesor, así como su arquitectura (System Architect).
- Detallar lo que se aprendió de este proyecto.
- Concluir el proyecto para su entrega (Project Manager).

TESTS

La forma de hacer las pruebas es usando el proceso automatizado que nos proporciona la herramienta de mix, en el cual se debe estar dentro de la carpeta codex, y después ejecutar el comando de "*mix escript.build*". Esto para poder generar un ejecutable de nuestro compilador.

```
[andres@pc-aml codex]$ mix escript.build
Compiling 1 file (.ex)
Generated escript codex with MIX_ENV=dev
```

Después para correr las pruebas que Nora Sandler propone hay que agrupar los archivos de pruebas (archivos .c), en la carpeta de test.



Finalmente para hacer las pruebas únicamente se escribe el comando de "mix test", esto para poder realizar las pruebas.

```
[andres@pc-aml codex]$ mix test
........................

Finished in 0.1 seconds
24 tests, 0 failures

Randomized with seed 354966
[andres@pc-aml codex]$
```

Como se puede notar, la herramienta de mix nos ayuda al momento de probar los archivos que Nora propone además que hace las pruebas tanto para la sección del lexer y del parser, siendo 12 pruebas para cada sección.

CONCLUSIÓN.

Con la elaboración de esta primera entrega del proyecto se trabajó en equipo con el fin de mejorar nuestra manera de comunicarnos, asignando roles a los miembros del equipo, de modo que se pueda hacer un trabajo o entrega integral.

La idea principal de la asignación de roles a cada miembro del equipo fue para que cada uno sea responsable del área correspondiente, además de aumentar la eficiencia, ya que siempre el dividir problemas grandes en subproblemas es más conveniente.

Consideramos que el trabajar con actualizaciones de código usando un controlador de versiones como lo es Github, ayuda o aclara bastante al momento de ver los nuevos cambios que cada integrante del equipo va añadiendo, por lo que el poder colaborar añadiendo o editando códigos de los compañeros ayuda bastante, ya que es un apoyo colaborativo el cual brinda un avance colectivo y eficaz.

En conclusión, este tipo de proyectos nos ayudan bastante, debido que nosotros hemos estado acostumbrados a hacer proyectos en su mayoría sin analizar, en cambio este proyecto se prestó bastante para ver realmente como es que un proyecto se debe elaborar, tratando de agrupar ciertas partes del proyecto haciendo el problema de este más pequeño o menos complejo, para después conjuntarlos.