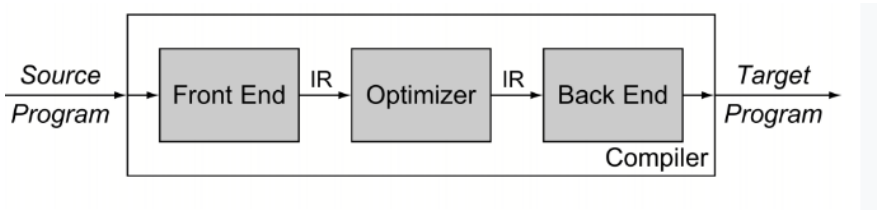
ARQUITECTURA

El objetivo de los compiladores es el preservar el significado del programa que se está compilando, siendo este un aspecto semántico. Además, de que 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.

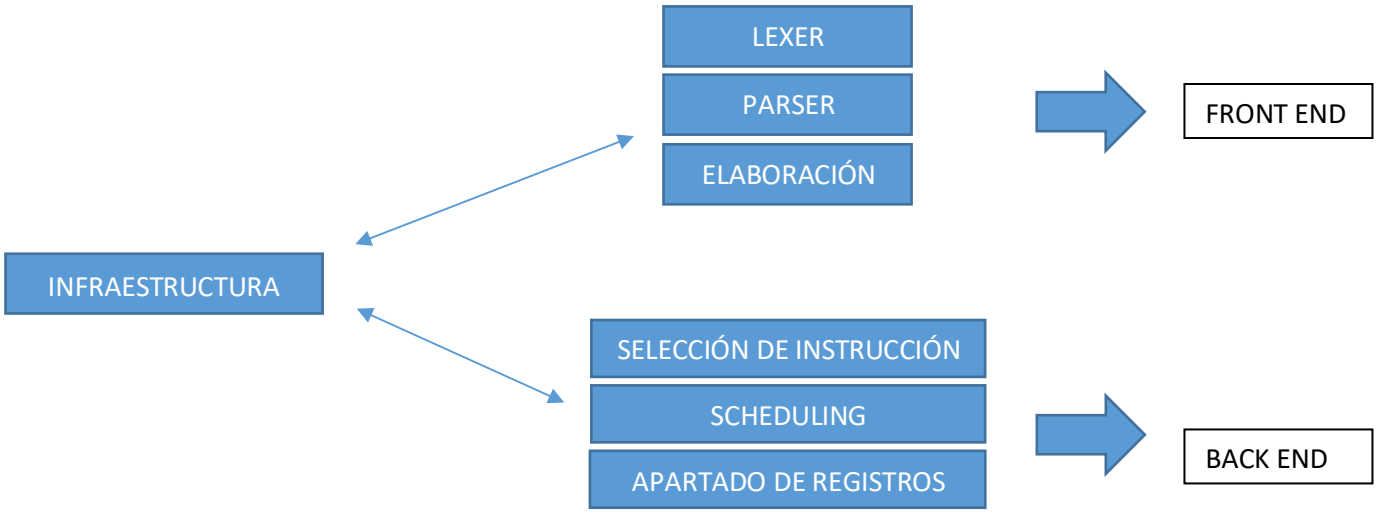
Siendo estos los principios de los compiladores, la estructura de nuestro compilador es la siguiente.



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

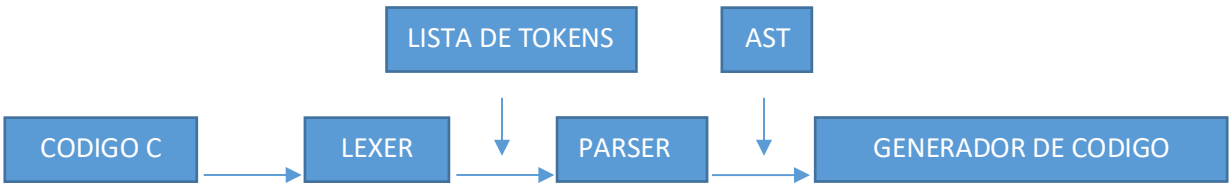
La tarea principal del Front End es el poder leer el programa, checar la sintaxis o las reglas que debe seguir el programa y checar la semántica o el significado del mismo.

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.



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 cual va a 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 de modo que nos otorgara un archivo ejecutable.



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 (Proyect 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 (Proyect 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 (Proyect Manager).

● Segunda entrega

El equipo se reunirá para analizar los nuevos requerimientos de la entrega, de manera que la implementación sea la más eficaz posible.

● Tercera entrega

En esta entrega se analizará cómo es que al ir aumentando los requerimientos el sistema seguirá comportándose de buena manera o en otro caso, ver en qué aspectos afecta para con ello, corregirlos.

- Entrega final

En esta entrega se verá en funcionamiento el sistema con todos los requerimientos solicitados por el cliente.