

Objetivo

Diremos a continuación se definirían que harán nuestras entregas :

Versión 1:

- Nuestro compilador debe poder devolver un número entero, compilando un código de fragmento simple.

Versión 2:

- Nuestro compilador debe poder devolver un resultado operado con operaciones unarias como negar resultado, positivo, bit a bit y negación lógica.

Versión 3:

- Nuestro compilador debe poder operar una operación binaria como suma, resta, multiplicación y división.

Para alcanzar este alcance propuesto, nos basaremos en el tutorial de “Nora Sandler”, hasta su tercera entrega.

¿Cómo trabajamos?

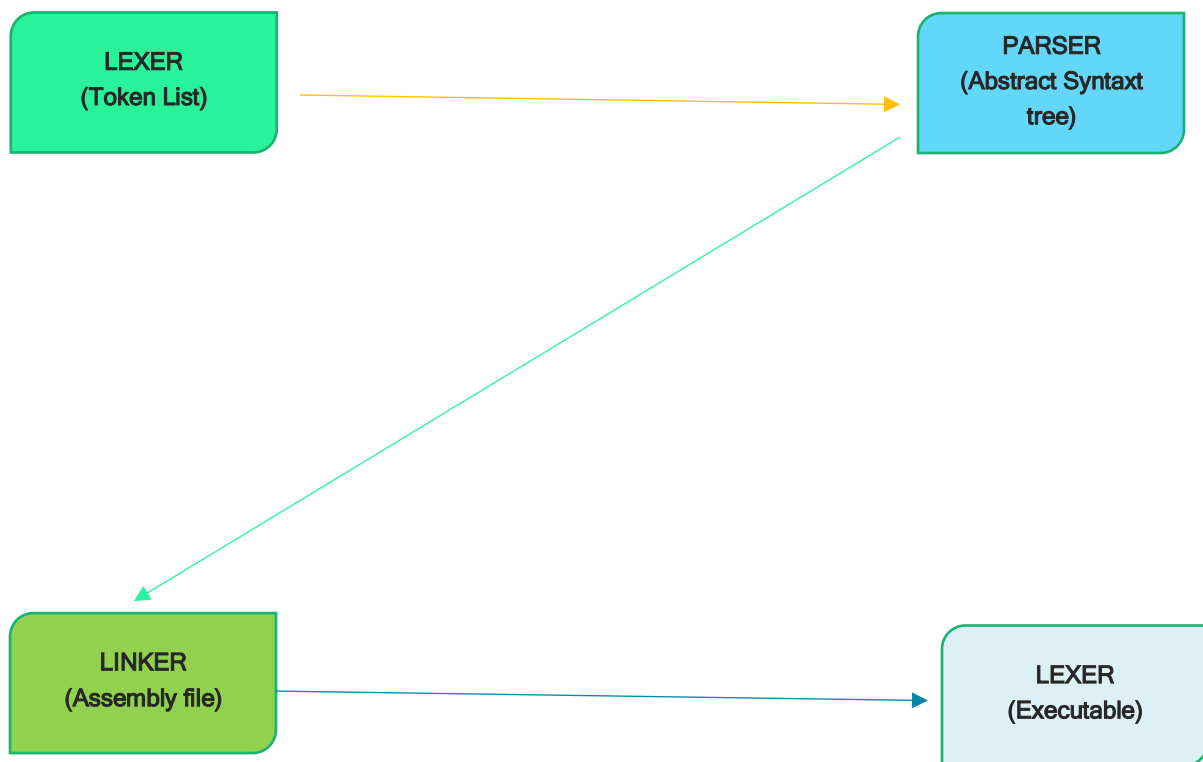
El compilador será desarrollado con arquitectura de x64 bits nuestro principal entorno operativo y donde el compilador funcione correctamente será UNIX (Ubuntu) y Windows

Integrations:	Description:
Lexer:	<p>Esta integración validará esto:</p> <ul style="list-style-type: none"> Valida la lista de tokens. La salida será una lista de tuplas de cadenas de átomos. Si hay un error, se mostrará una lista de tuplas con el token, así como la columna y fila incorrectas.
Parser:	<p>Esta integración nos permitirá establecer la siguiente funcionalidad básica :.</p> <ul style="list-style-type: none"> Genere un AST con la lista de tuplas creadas por Lexer. Si hay algún error, mostrará una lista de tuplas con el token que genera el error, la columna y la fila.
Code generator:	<p>Esta integración nos permitirá establecer la siguiente funcionalidad básica:</p> <ul style="list-style-type: none"> Tome el AST generado por el analizador para construir el código en ensamblador, desde las hojas hasta la raíz. La salida será una cadena con el código representativo en ensamblador.

Linker:

✚ Linker: es un programa del sistema informático que toma uno o más archivos objeto generados por un compilador o un ensamblador y los combina en un solo archivo ejecutable, archivo de biblioteca u otro archivo 'objeto'

How does it work?



Architecture.

Este compilador ha sido diseñado para ser lo más manejable posible. Por lo que significa que una vez que esté funcionando, no se debe alterar su código. Para ello, se han abstraído con una serie de actores que simplemente operan sobre una serie de archivos para el lenguaje deseado. Entonces, esta arquitectura solo funciona con este compilador.

What should the compiler do?

El objetivo principal es ejecutar lo siguiente, pero de múltiples formas:

```
int main(){  
return "constant";  
}
```

La salida debe ser un archivo ejecutable, tratando de seguir las instrucciones a continuación y tratando de hacerlo de la manera correcta.

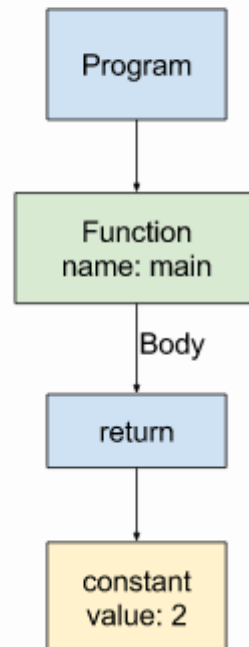
Flag:	Must do:
-t	Devuelve la lista del token de nuestro archivo compilado.
-a	Devuelve el árbol de sintaxis abstracto del archivo compilado.
-s	Debe generar un archivo de ensamblaje.
-c	Compila el programa.
-h	Nuestra muestra la ayuda

Delivery date of the work:

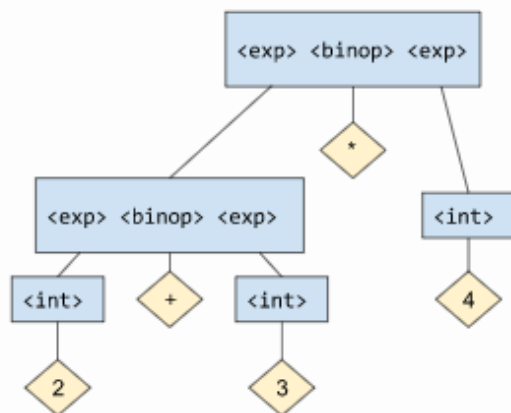
Integers.	13-08-21	Nuestro compilador debe poder devolver un número entero, compilar un fragmento de código simple y realizar múltiples pruebas.
Unary operators.	13-08-21	Nuestro compilador debe poder devolver un resultado operado con operaciones unarias como negar resultado, positivo, bit a bit y negación lógica, haciendo múltiples pruebas.
Binary Operators.	13-08-21	Nuestro compilador debe poder operar una operación binaria como suma, resta, multiplicación y división, haciendo múltiples pruebas.

Primera entrega

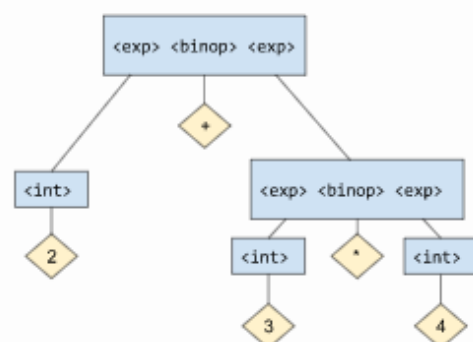
Here's a diagram of the AST for return_2.c:



Tecera entrega



Tree #1



Tree #2

