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 Linux (Ubuntu) y Windows usaremos el sistema ensamblador AT&T

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.

Entregas de trabajo:

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.

Integrations:	Description:
Lexer:	Esta integración validará esto: - 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:	Esta integración nos permitirá establecer la siguiente funcionalidad básica :. - 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:	Esta integración nos permitirá establecer la siguiente funcionalidad básica: - 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'

Primera entrega

- Compila un código fuente en C y devuelve un número entero cuando ejecuta el archivo .exe.

- Para lograr este objetivo, nos basaremos en cómo crear un compilador de acuerdo con la documentación de Nora, como así como las pruebas que se deben ejecutar.

Puede compilar varias pruebas que tenemos, si lo desea de la misma manera, pero aquí solo haremos una

Segunda entrega

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

- Para lograr este objetivo, nos basaremos en cómo crear un compilador de acuerdo con la documentación de Nora, como así como las pruebas que se deben ejecutar.

Puede compilar varias pruebas que tenemos, si lo desea de la misma manera, pero aquí solo haremos una

Tercera entrega

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

- Para lograr este objetivo, nos basaremos en cómo crear un compilador de acuerdo con la documentación de Nora, como así como las pruebas que se deben ejecutar.

Puede compilar varias pruebas que tenemos, si lo desea de la misma manera, pero aquí solo haremos una

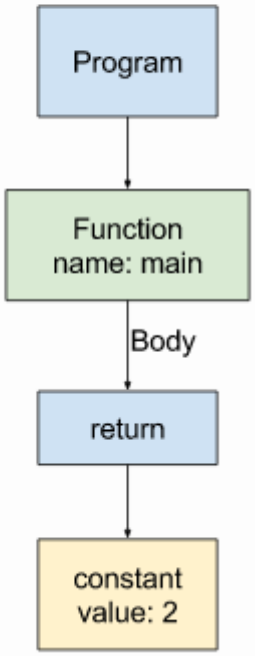
Las banderas

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

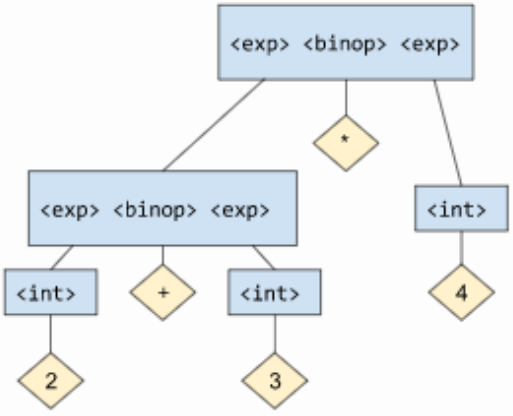
Flag:	Must do:
-t	(-t) Mostramos el código fuente de nuestra lista del tokens. Debe marcar una pareja relacional para reconocer cada token.
-a	(-a) El lenguaje de desarrollo debe ser un patrón coincidente para construir fácilmente un árbol de sintaxis abstracta (AST).
-s	(-s) El ensamblaje debe escribir instrucciones de conjunto de 64 bits. La sintaxis de ensamblado debe ser AT y T por defecto en GCC.
-c	Compila el programa.
-h	Nuestra muestra la ayuda

Primera entrega

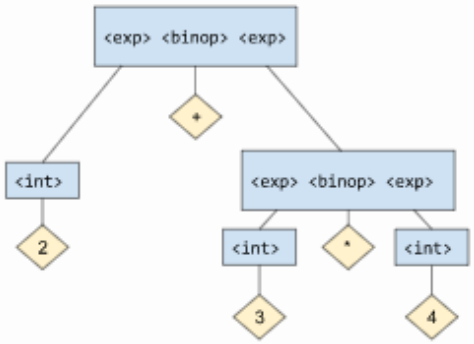
Here's a diagram of the AST for return_2.c:



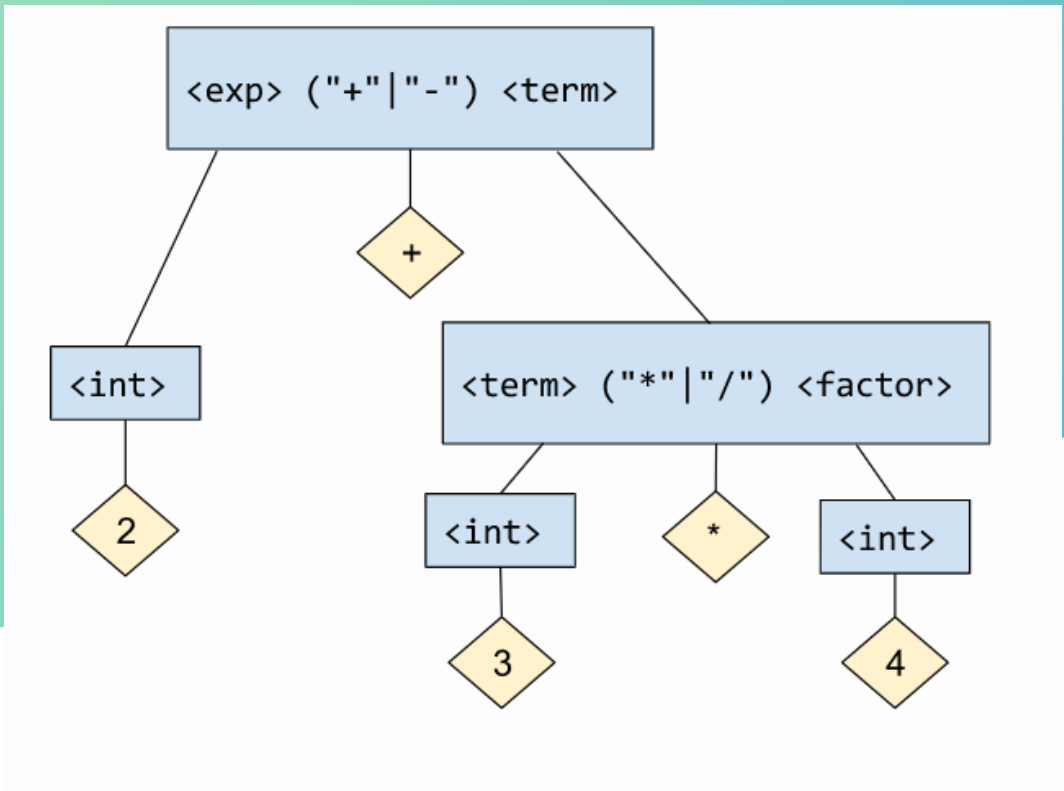
Tecera entrega



Tree #1



Tree #2


<exp> ("+"|"-") <term>
+
<int>
2
<term> ("*"|"/") <factor>
<int>
3
*
<int>
4