UNIVERSIDAD DE GUADALAJARA

Red Universitaria e Institución Benemérita de Jalisco

Alumno:

Romo Rodríguez José Alberto (216853747)

Materia:

I7028 | Seminario de Solucion de Problemas de Traductores de Lenguajes II

NRC:

103841

Sección:

D02

Maestro:

Lopez Franco Michel Emanuel

Horario:

Lunes y Miercoles | 13:00 - 14:55

Tarea:

Codigo Objeto

Fecha de Entrega:

25 de Mayo del 2023

Introduccion.

Generar código objeto a partir de un análisis léxico, sintáctico y semántico es un proceso esencial en la compilación de un programa. Permíteme explicarte paso a paso cómo se lleva a cabo este proceso.

Análisis léxico:
El análisis léxico es la primera etapa en la compilación de un programa. Su objetivo es escanear el código fuente y dividirlo en unidades léxicas, también conocidas como tokens. Estos tokens pueden ser palabras clave (if, while, int, etc.), identificadores (nombres de variables o funciones), operadores (+, -, *, etc.), constantes numéricas o literales de cadena.

Por ejemplo, si tenemos el siguiente código fuente en C:

```
int main() {
    int a = 5;
    return a;
}
```

El análisis léxico generaría tokens como:

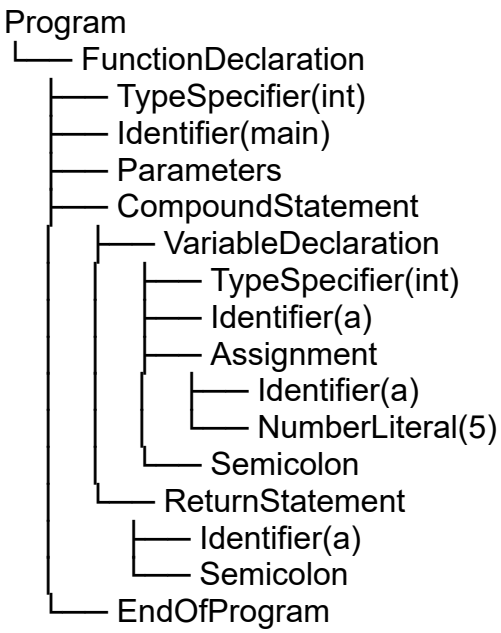
```
TOKEN_INT
TOKEN_MAIN
TOKEN_LPAREN
TOKEN_RPAREN
TOKEN_LBRACE
TOKEN_INT
TOKEN_IDENTIFIER(a)
TOKEN_ASSIGN
TOKEN_NUMBER(5)
TOKEN_SEMICOLON
TOKEN_RETURN
TOKEN_IDENTIFIER(a)
TOKEN_SEMICOLON
TOKEN_RBRACE
```

Análisis sintáctico:

Una vez que tenemos los tokens generados, pasamos al análisis sintáctico, también conocido como análisis gramatical. En esta etapa, se verifica si la estructura del código fuente cumple con la gramática definida por el lenguaje de programación. Se utiliza una representación formal llamada gramática libre de contexto para describir la estructura sintáctica del lenguaje.

El análisis sintáctico construye un árbol de análisis llamado árbol sintáctico o árbol de derivación, que representa la estructura jerárquica del código fuente. Este árbol sintáctico se construye aplicando reglas gramaticales del lenguaje de programación a los tokens generados en el análisis léxico.

Usando el ejemplo anterior, el árbol sintáctico podría ser:



Análisis semántico:

El análisis semántico tiene como objetivo verificar la coherencia y corrección del significado del programa. Durante esta etapa, se realizan comprobaciones adicionales que no pueden ser capturadas por el análisis léxico o sintáctico, como la declaración y uso correcto de variables, la compatibilidad de tipos y la detección de errores semánticos.

El análisis semántico también construye una tabla de símbolos, que es una estructura de datos que almacena información sobre las variables, funciones y otros elementos del programa. La tabla de símbolos se utiliza posteriormente en la generación de código objeto.

Continuando con el ejemplo, en el análisis semántico se verificaría que la variable "a" está correctamente declarada antes de su uso, que su tipo es compatible con la expresión de retorno, entre otras comprobaciones.

Generación de código objeto:

Una vez que se ha completado el análisis semántico y se ha verificado la corrección del programa, se puede proceder a la generación de código objeto. El código objeto es una representación intermedia del programa que puede ser ejecutada por la máquina objetivo, o bien, puede ser posteriormente traducida a código máquina.
Durante la generación de código objeto, se recorre el árbol sintáctico y se traduce cada nodo en instrucciones de código objeto. Estas instrucciones son específicas para la arquitectura de la máquina objetivo.

Siguiendo el ejemplo, se generaría código objeto en lenguaje ensamblador o en código de bytes, dependiendo de la arquitectura objetivo.

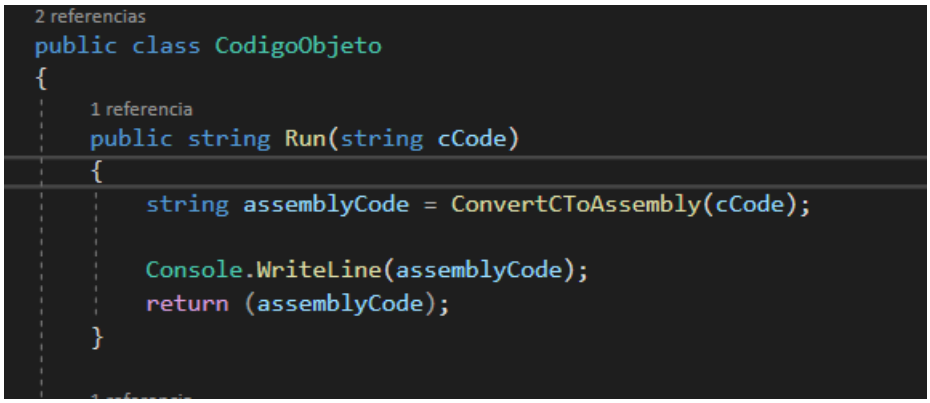
Es importante tener en cuenta que el proceso de generación de código objeto puede ser más complejo en la práctica, ya que implica considerar optimizaciones, manejo de registros, manejo de memoria, entre otros aspectos.

En resumen, la generación de código objeto a partir del análisis léxico, sintáctico y semántico es un proceso fundamental en la compilación de un programa. Cada etapa cum-

ple una función específica para garantizar la correcta traducción del código fuente a un formato ejecutable o procesable por la máquina objetivo.

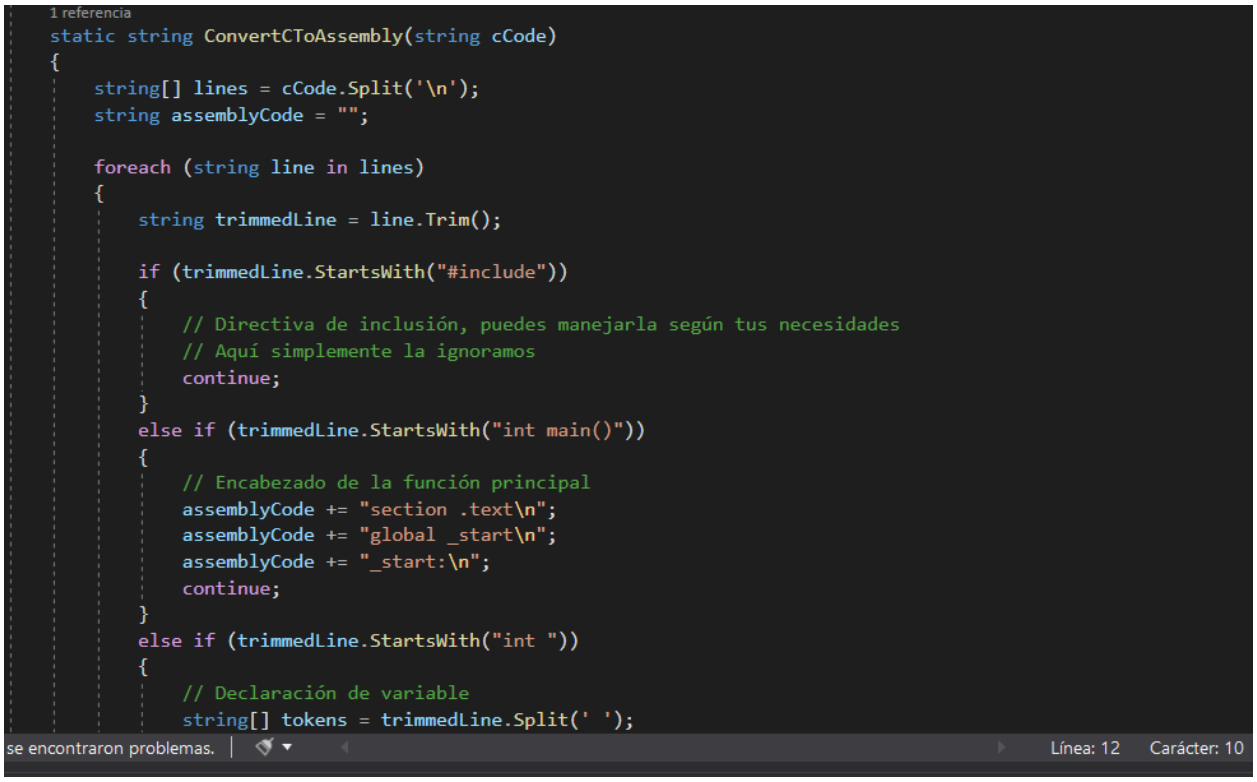
Desarrollo.

Para realizar la generacion del codigo de tipo ensamblador se comienza teniendo todos nuestros resultados listos simplemente para ser procesados:



Para ello se hace una funcion nueva que se encarga de ello.

El programa comienza solicitando el programa en cuestion que sabemos esta verificado.



A continuación, el programa llama a la función ConvertCToAssembly, pasándole el código en estilo C como argumento.

La función ConvertCToAssembly toma el código en estilo C y lo divide en líneas.
A medida que recorre cada línea, la función realiza ciertas operaciones dependiendo de su contenido.

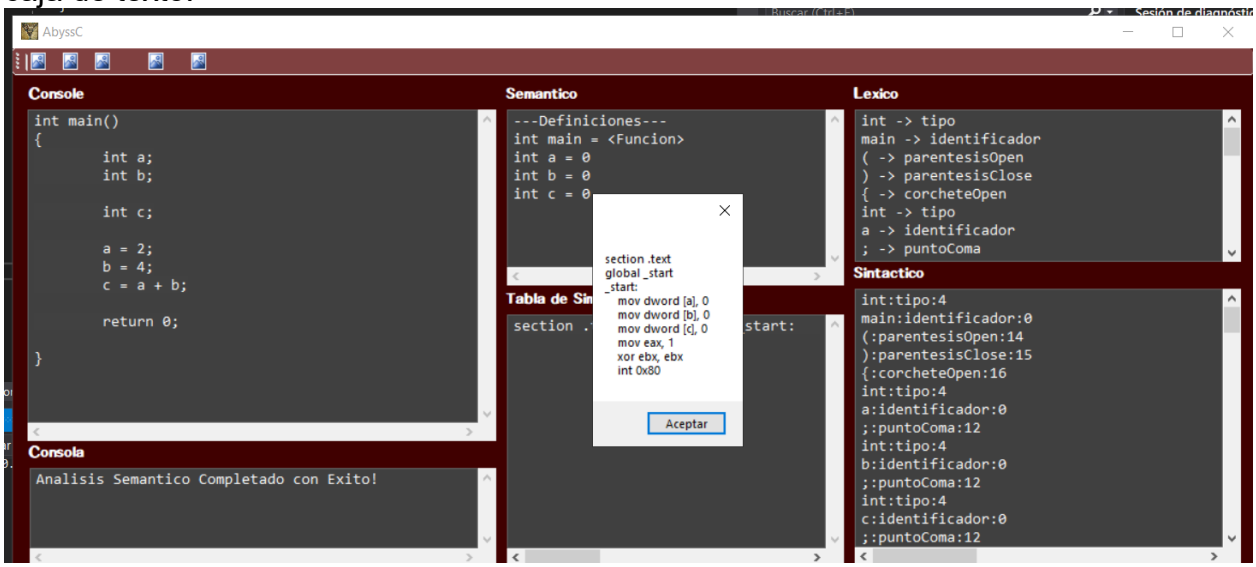
La línea que comienza con int main() (encabezado de la función principal) indica el inicio

de la sección de texto del ensamblador.

Las líneas que comienzan con int (declaraciones de variables) se convierten en instrucciones de ensamblador para asignar memoria y establecer los valores iniciales de las variables.

Después de recorrer todas las líneas, la función ConvertCToAssembly devuelve el código en ensamblador resultante.

El programa principal muestra el código en ensamblador por la consola ademas de una caja de texto:



En resumen, el código en C# proporcionado permite convertir código en estilo C a su equivalente en ensamblador, utilizando una lógica simple para analizar el código en estilo C y generar el código en ensamblador correspondiente.

Ahora con este codigo podemos ejecutar el programa en un compilador directamente de ensamblador y obtener conclusiones.