

Campus de Quetzaltenango  
Facultad de Ingeniería   
Ingeniería en Informática y Sistemas  
Sexto Semestre  
Compiladores  
Ing. Dhaby Xiloj

**“Proyecto: Compilador LOOP”**

Alicia del Rosario Arroyave Fernandez

Carné: 1575517

Jasmine Lineth del Pilar Morales Granados

Carné: 1559917

Guillermo Danilo de Jesús Herrera Castillo

Carné:1509916

Quetzaltenango, 18 de octubre de 2019.

**El proyecto**

El proyecto consiste en la creación de un intérprete para un lenguaje Orientado a Objetos creado únicamente para este curso, llamado LOOP o Lenguaje Orientado a Objetos en Pseudocódigo. El lenguaje es una combinación de ideas de otros lenguajes, agregando características nuevas, logrando así crear un lenguaje que puede ser utilizado por cualquier persona que quiera aprender a programar, pero en idioma castellano. El resultado debe de generar un código en Assembler para el PIC 18F4550 y a su vez utilizando el opcode de este pic poder convertirlo a hexadecimal.

**Objetivos**

La realización del proyecto propone mejorar las habilidades en el uso de herramientas para la creación de intérpretes y compiladores, así como también mejorar la lógica de creación de lenguajes. De igual manera ayudar al usuario a realizar una traducción a Assembler de manera más simple.

**¿Qué hace el programa?**

Este programa le permite seleccionar un archivo de texto y la aplicación comprobará que código cumple con la estructura de un programa escrito con el lenguaje LOOP.

La primera instrucción que hace el programa es pre-procesar el archivo. En esta etapa se lee el archivo de entrada y se busca si tiene bibliotecas para añadir, de ser así, agrega la biblioteca al inicio del programa y se almacena un archivo temporal en las carpetas del usuario, el cual será el archivo que se usará durante la ejecución del programa.

Luego el programa se encarga de leer todo el documento y verificar que todos los lexemas pertenezcan al leguaje. En este punto el programa ya sabe que todos los lexemas son válidos y se debe empezar a validar las reglas gramaticales y de estas crear un árbol sintáctico el cual es pasado a la siguiente etapa como resultado de analizar las reglas que definen al programa LOOP.

Ya que se verificó que se cumplieran las reglas, es momento de verificar reglas un poco más complejas, como la asignación de tipos a las variables. Para esto se recorre el árbol y se verifica que las asignaciones cumplan con los tipos y que todas las variables sean declaradas, y así poder ver si son llamadas desde otro lado por medio de la tabla de símbolos.

Una vez el árbol ha sido verificado, ya está listo para su ejecución. En esta parte se empieza con la función principal y se ejecutan sus instrucciones líneas a línea, se van asignando y cambiando los valores en la tabla de símbolos y se realizan las operaciones de archivos correspondientes; así sucesivamente hasta que se termine el árbol, dando como resultado un código en Assembler. Al terminar esto, el programa realiza la traducción de Assembler a Hexadecimal para poder devolver de esta manera un archivo .hex.

**ESTRUCTURA**

Los programas escritos en lenguaje loop tienen la siguiente estructura:

* Carga de Librerías
* Clases
* Propiedades
* Métodos
* Condiciones, operaciones, ciclos… etc.

Además, la sintaxis hace uso de identación para indicar jerarquía y cada línea puede o no acaban con “;”

**Tipos de datos**

Los tipos de identificadores que el lenguaje debe reconocer como básicos serán:

1. **entero:** solo manejará valores enteros como1, 30, 234912.
2. **real:** podrá manejar numeración de punto flotante como1.25, 0.59, 0.002**.**
3. **cadena:** puede almacenar cadenas de texto de cualquier longitud como \hola", \mundo".
4. **booleano:** permitirá guardar valores comoverdaderoyfalso, donde las palabrasverdaderoy falso están reservadas, además debe poder almacenarlos y usarlos con su equivalente 1 y 0.
5. **nulo:** solamente será usado para funciones que no retornen ningún valor al terminar suejecución.

**Declaración de variables**

Estructura de declaración de una variable

< tipo > < variable [, variable] > [= < valor inicial >]

**Estructuras selectivas:** utilizadas para manipular la ejecución del código, pudiendo utilizar una estructura con la opción de usar otro trozo de código si la condición no se cumple

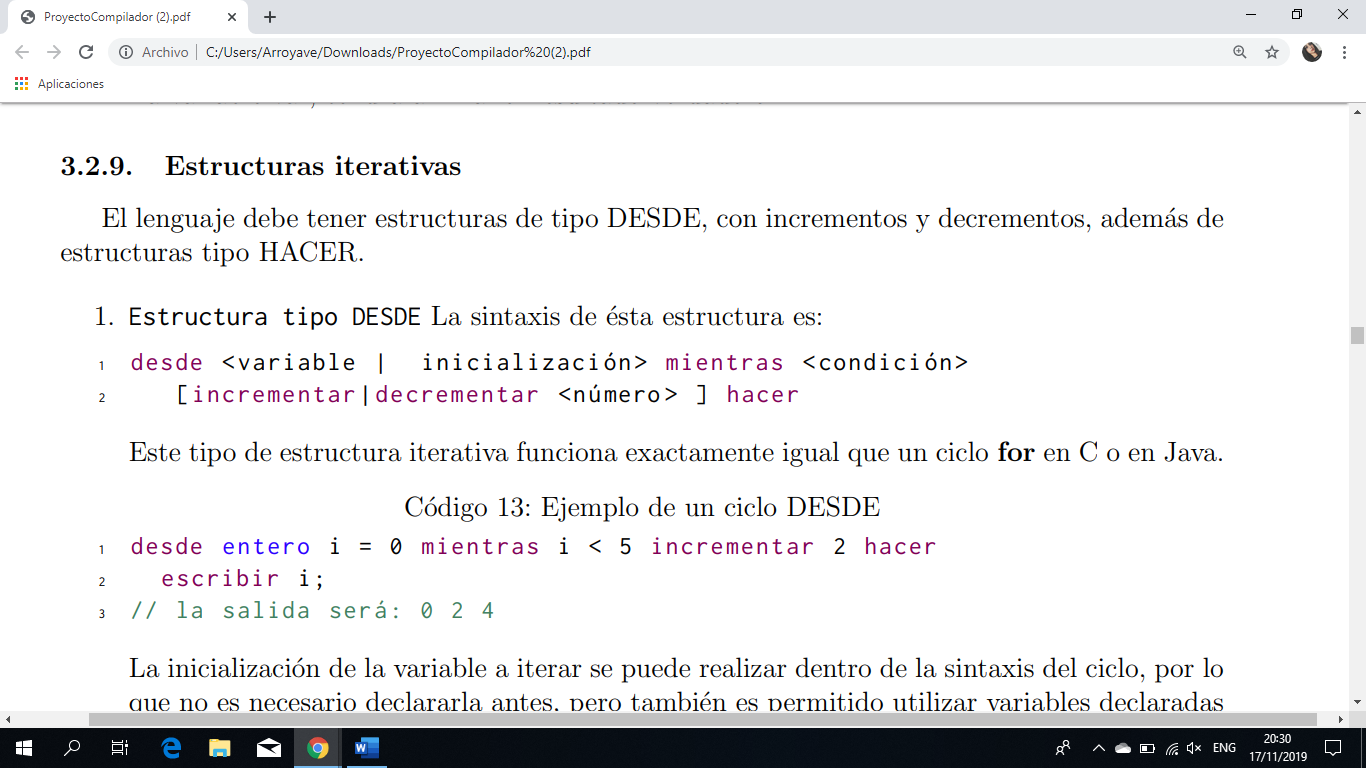
Si (if), debe tener la siguiente sintaxis:

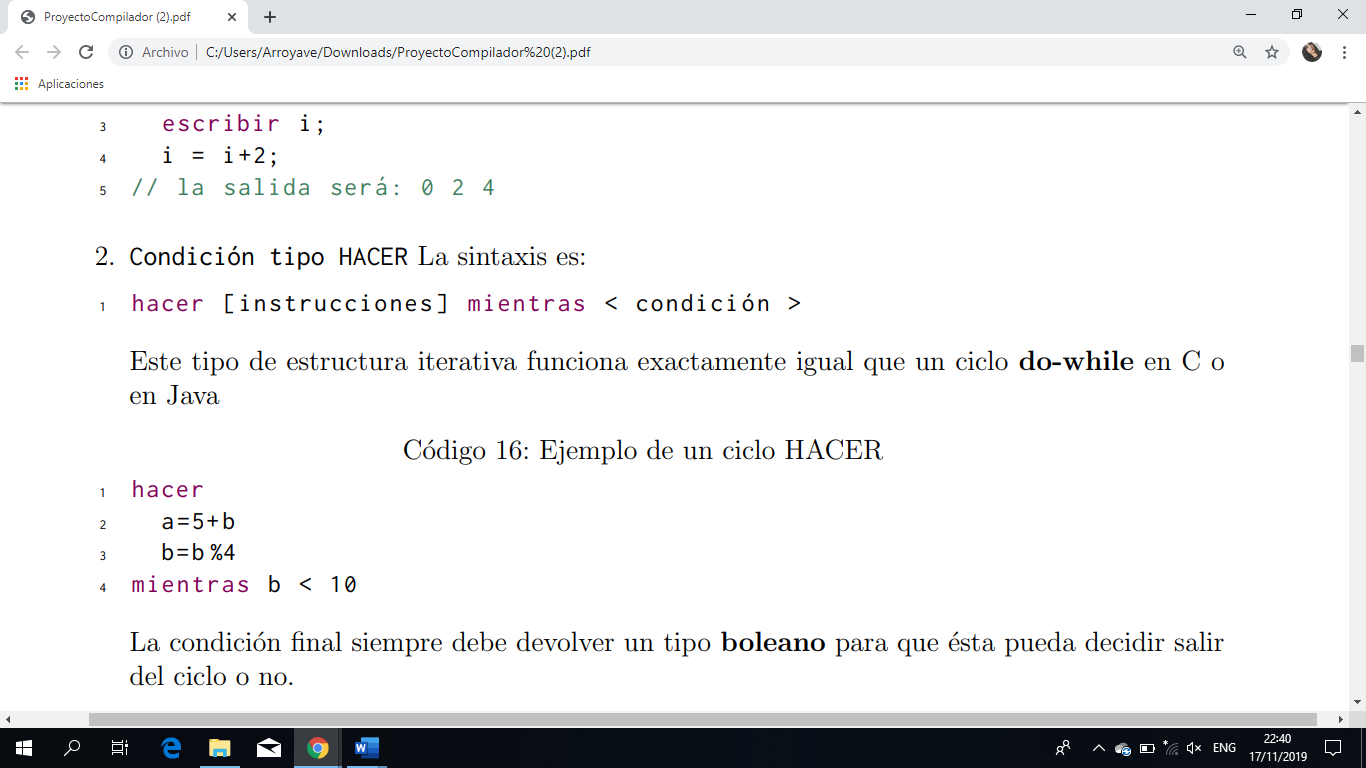
si < condicion > entonces [ sino [ condicion entonces ] ... ]

La sintaxis nos permite crear múltiples condiciones anidadas, según sea necesario.

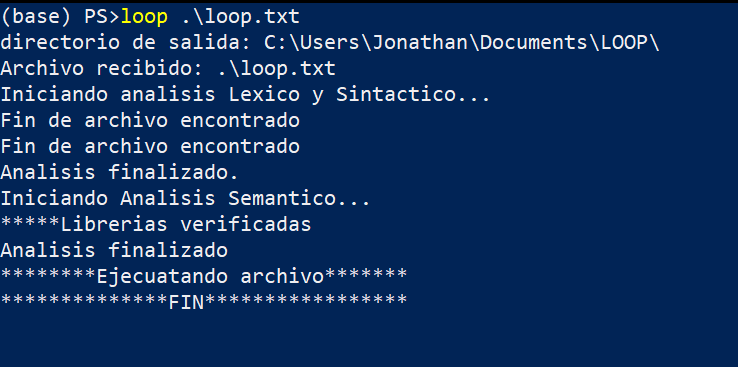
**Condiciones**: las condiciones son operaciones booleanas, las cuales evaluarán la validez de una comparación entre dos variables, o bien, constantes, para lo cual se cuenta con cuatro operadores:

* MAYOR QUE >
* MENOR QUE <
* IGUAL QUE ==
* DISTINTO QUE !=



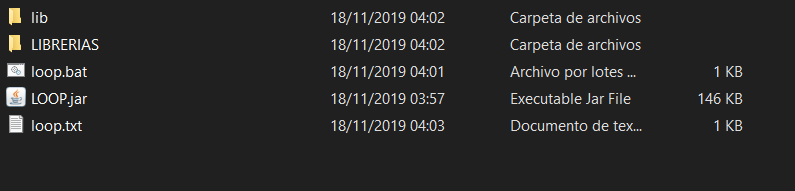


**Utilización**



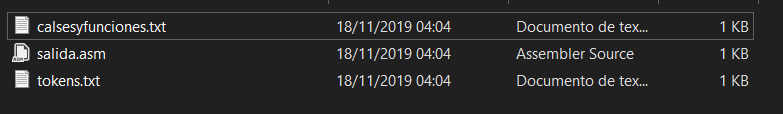
**En la carpeta exec esta todo lo necesario para que corra el programa:**

1. **Librerías Jcupy Jfex**
2. **Bat para ejecutar el jar**
3. **Un ejemplo loop.txt**
4. **Loop.jar**
5. **Y librerías que se pueden cargar al programa**

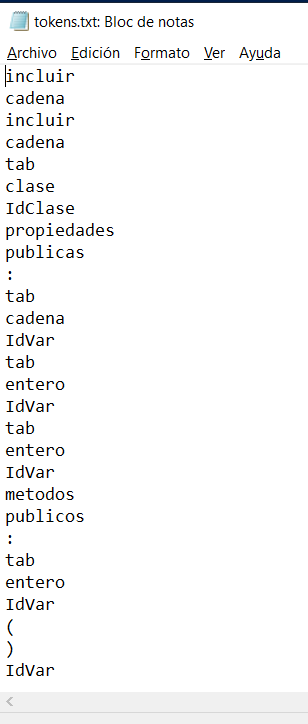


**La salida del programa es en los archivos del usuario -> documentos -> loop**

* **Tabla de Tokens.**
* **Funciones, variables y métodos.**
* **Salida asm**



**Archivo Tokens**



**Archivos variables, clases y métodos**

