

# Universidad Nacional Autónoma de México

## Facultad de Ingeniería



Primera Entrega Compilador

### **Wombat**

Datos del Cliente:

Ing. Norberto Jesus Ortigoza Marquez

Equipo de Desarrollo:

Romero Andrade Cristian

-

PMP, Arquitecto

Arguelles Macosay Mariana

-

Integradora, Tester

Entrega V 0.1: 16 de septiembre del 2019

# Índice

<b>1. Introducción</b>	<b>1</b>
<b>2. Plan de Proyecto</b>	<b>1</b>
2.1. Descripción del Proyecto . . . . .	1
2.2. Entorno de Trabajo . . . . .	2
2.3. Actividades del equipo . . . . .	3
<b>3. Arquitectura del Compilador</b>	<b>3</b>
<b>4. Plan de Pruebas y de “Suites”</b>	<b>4</b>
<b>5. Conclusiones</b>	<b>4</b>
<b>Appendices</b>	<b>4</b>

## 1. Introducción

*Un compilador para un pequeño Subconjunto de C,*

Este proyecto trata acerca de crear un compilador en uno de los lenguajes de programación permitidos, en este caso hemos elegido Elixir como el lenguaje del proyecto. Para poder empezar debemos saber ¿Qué es un compilador?, un compilador es de forma sencilla un programa que traduce un lenguaje a lenguaje máquina es decir un lenguaje de alto nivel (palabras entendibles que puedan expresar que queremos que haga un programa) a lenguaje maquina (conjunto de números cuyo significado es directamente una orden al microprocesador). Para la creación de un compilador existen ciertos procesos o pasos para poder hacer las traducciones o compilaciones dependiendo sea el caso y la estructura que tendrá el compilador, debido a la complejidad de los compiladores, en esta o primera entrega solo haremos una parte de un compilador, aun así manteniendo una estructura completa.

## 2. Plan de Proyecto

### 2.1. Descripción del Proyecto

El proyecto total consiste en un compilador de escala altamente reducida de C la cual consiste en lo siguiente:

```

1  int main() {
2      return (3+4 <+ 4 || 1&& != 3>-6);
3
4  }
```

Para esta primera entrega el alcance llega a compilar el siguiente código

```

1  int main() {
2      return 2;
3  }
```

**Compilador Wombat**

17/09/2019

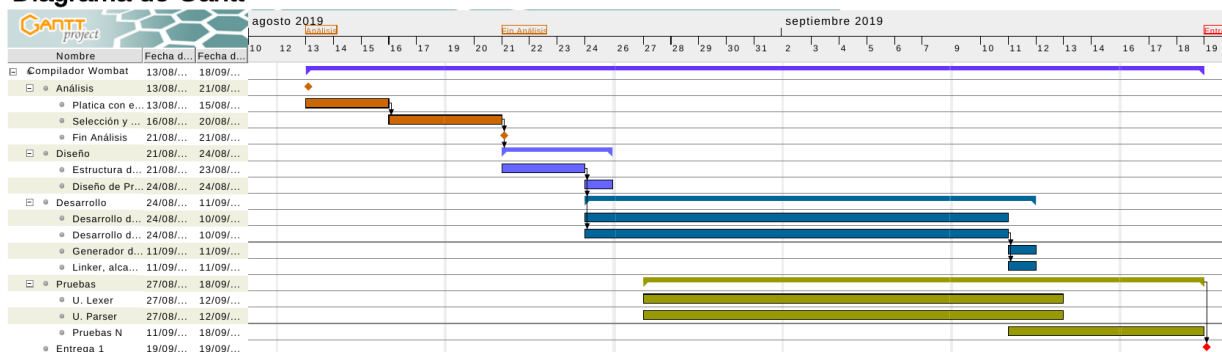
**Tarea**

2

Nombre	Fecha de inicio	Fecha de fin
Compilador Wombat	13/08/19	18/09/19
Análisis	13/08/19	21/08/19
Platica con el Stakeholder	13/08/19	15/08/19
Selección y estudio de herramientas	16/08/19	20/08/19
Fin Análisis	21/08/19	21/08/19
Diseño	21/08/19	24/08/19
Estructura de Compilador	21/08/19	23/08/19
Diseño de Pruebas	24/08/19	24/08/19
Desarrollo	24/08/19	11/09/19
Desarrollo del Lexer, alcance 1	24/08/19	10/09/19
Desarrollo del Análizador, alcance 1	24/08/19	10/09/19
Generador de Código ASM, alcance 1	11/09/19	11/09/19
Linker, alcance 1	11/09/19	11/09/19
Pruebas	27/08/19	18/09/19
U. Lexer	27/08/19	12/09/19
U. Parser	27/08/19	12/09/19
Pruebas N	11/09/19	18/09/19
Entrega 1	19/09/19	19/09/19

**Diagrama de Gantt**

4



Las próximas tareas al cerrar esta entrega, a parte de la adición de los operadores unarios, será obtener el error de compilación junto con el número de línea.

**2.2. Entorno de Trabajo**

- Por petición del cliente todo entrégale relacionado con el proyecto se llevará a través de la plataforma de [GitHub](#). Más adelante se describe como se esta organizado el proyecto.
- Se utilizará Elixir por las facilidades del lenguaje para el desarrollo específico del proyecto.
- El editor es libre para cada desarrollador, aunque se recomienda el uso de *Visual Studio Code* y *Emacs* por el soporte del lenguaje (vscode-elixir y alchemist-mode respectivamente).

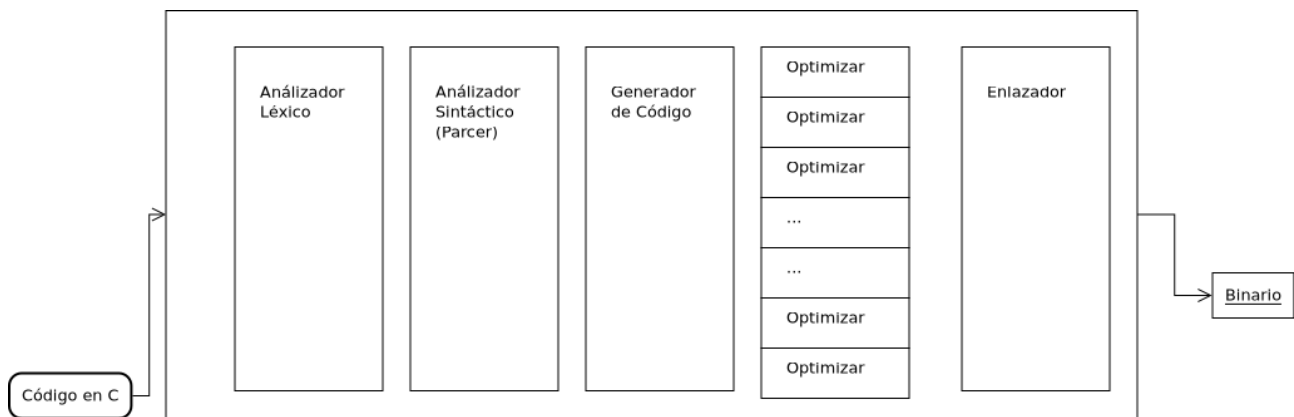
- Se usará *gcc* para enlazar el código ensamblador con la máquina.
- Por ende se utiliza el gestor de versiones git.

### 2.3. Actividades del equipo

Se espera la siguiente rutina para el desarrollo del proyecto:

- Se trabajará en la rama *dev*, cada fusión con *master* se aprobará siempre y cuando la adición de un funcionamiento este completa y no haya cambios con el objetivo del proyecto.
- Toda comunicación va a ser a través de *Discord* y otro medio de mensajería instantánea.

## 3. Arquitectura del Compilador



La arquitectura del compilador es simple utilizado el gestor de proyectos de *elixir mix*, la cual genera las herramientas suficientes para las pruebas y compilación. La estructura que se decidió fue la siguiente:

- `_build`
- `config`
- `ejemplo`
  - `return_2.c`
- `lib`
  - `compiladorwombat.ex`
  - `wc2`
    - `analizador.ex`
    - `arbol.ex`

- 
- code\_gen.tex
  - lexer.ex
  - linker.ex
  - Makefile
  - miscelanea
  - mix.exs
  - mix.lock
  - README.md
  - test
    - compilador\_wombat\_test.exs
    - stage\_1<sup>1</sup>
    - test\_helper.exs

Ya que así se tiene separado las etapas del compilador y es fácil extender y dar mantenimiento a dicha estructura tomando a todos los archivos necesarios de manera independiente.

## 4. Plan de Pruebas y de “Suites”

El plan consiste en pasado un tiempo de desarrollo de los módulos del compilador, se comenzará las pruebas mediante una simple regla, “Lo que se debe de recibir, lo que se debe de mandar”, por ejemplo en el caso del lexer, lo que este recibe es una lista de cadenas de caracteres [“int”, “main”, etc] y lo que debe de mandar en una lista y/o tupla de *tokens* [:kint, :kmain, etc]. Ya completado esta “etapa” de pruebas, se comenzará con las pruebas de **Nora**, la cual se espera solo tener errores en sentencias específicas conservando el funcionamiento original.

## 5. Conclusiones

Gracias a la clase de compiladores nos fue posible definir una buena estructura para el programa, por ahora el programa funciona, aunque solo acepta lo básico y ciertas palabras clave, por lo que todavía no se puede considerar un compilador completo ni usar como uno. Se aprendió que existen diversas herramientas que facilitan la creación de compiladores. Para esta primera entrega podemos decir que se ha realizado con éxito.

# Appendices

Repositorio del Proyecto

<<https://github.com/hiphoox/c201-wombat>>

---

<sup>1</sup>Pruebas de Nora