Compilador de un sublenguaje de C

Project Manager
Atonal Jiménez Aarón

System Architect
Esquivel Cázares Mauricio

System Tester
Camargo Hernandez Alan Mauricio

System Integrator
Guevara Galván Pedro Josué

Índice

REQUERIMIENTOS

1. Entrada de archivo con extensión “.c“ con contenido:

```
int main( ) {
        return <entero>;
}
```

   Y su salida debe ser un **ejecutable**.

2. Se debe ejecutar desde la línea de comandos o terminal para sistemas unix-like.

3. El ejecutable debe tener el mismo nombre que el archivo compilado, debe estar en la misma ruta que se compiló.

4. Con la bandera “-o <nombre_ejecutable>” se debe poder modificar el nombre del ejecutable por uno diferente al del archivo original. El nombre del ejecutable por default es el nombre del archivo original.

```
$ ./nqcc --o <nombre_ejecutable>
```

5. La bandera “-s” devuelve el ensamblador del archivo compilado y no genera un ejecutable.

```
$ ./nqcc --s <nombre_archivo>
```

6. La bandera “-t” devuelve la lista de tokens gráficamente en la pantalla, no genera archivo ni ejecutable del archivo compilado.

```
$ ./nqcc --t <nombre_archivo>
```

7. La bandera “-a” devuelve el árbol AST del parser en consola y no se genera ejecutable del archivo compilado.

```
$ ./nqcc --a <nombre_archivo>
```

8. La bandera “-help” devuelve las instrucciones del compilador.

```
$ ./nqcc --help
```

9. Si se escribe “$ compilador ” en consola y se oprime enter sin especificar el nombre del archivo a compilar, marca error y devuelve el mensaje “falta nombre del archivo”.

```
$ ./nqcc
$ Falta nombre del archivo
```

10. Si se escribe “$ compilador <nombre archivo>” en consola, devuelve el ejecutable del archivo compilado en la misma carpeta que esta el archivo .c.

11. Una vez compilado si existen errores en el archivo o se insertaron palabras reservadas equivocadas, falta algún punto y coma, no se cerraron las llaves de main, o hay palabras demás que no deben ir dentro del archivo, muestra un mensaje “error léxico o sintáctico” indicando el número de línea en la consola..

```
$ ./nqcc <nombre_archivo>
$ Error error léxico o sintáctico, línea --
```

12. La línea de comandos solo soporta 1 bandera a la vez y no se pueden encadenar más banderas a la instrucción.

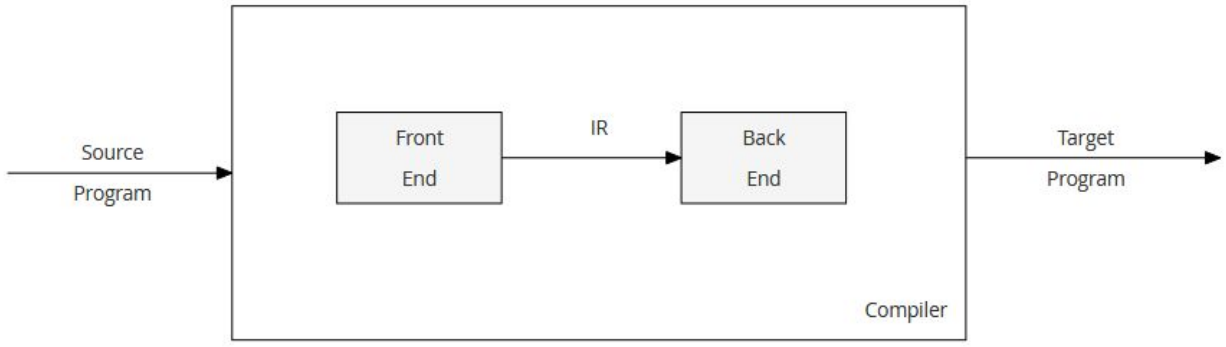
13. Se entregará un manual para correr las pruebas y además otro manual de como compilar el programa(como clonar y correr las pruebas de los ejemplos del git de Nora Sandler).

14. Cada entrega debe tener su versión (git tag version) para la consulta del cliente y así saber que version es la que está compilando.

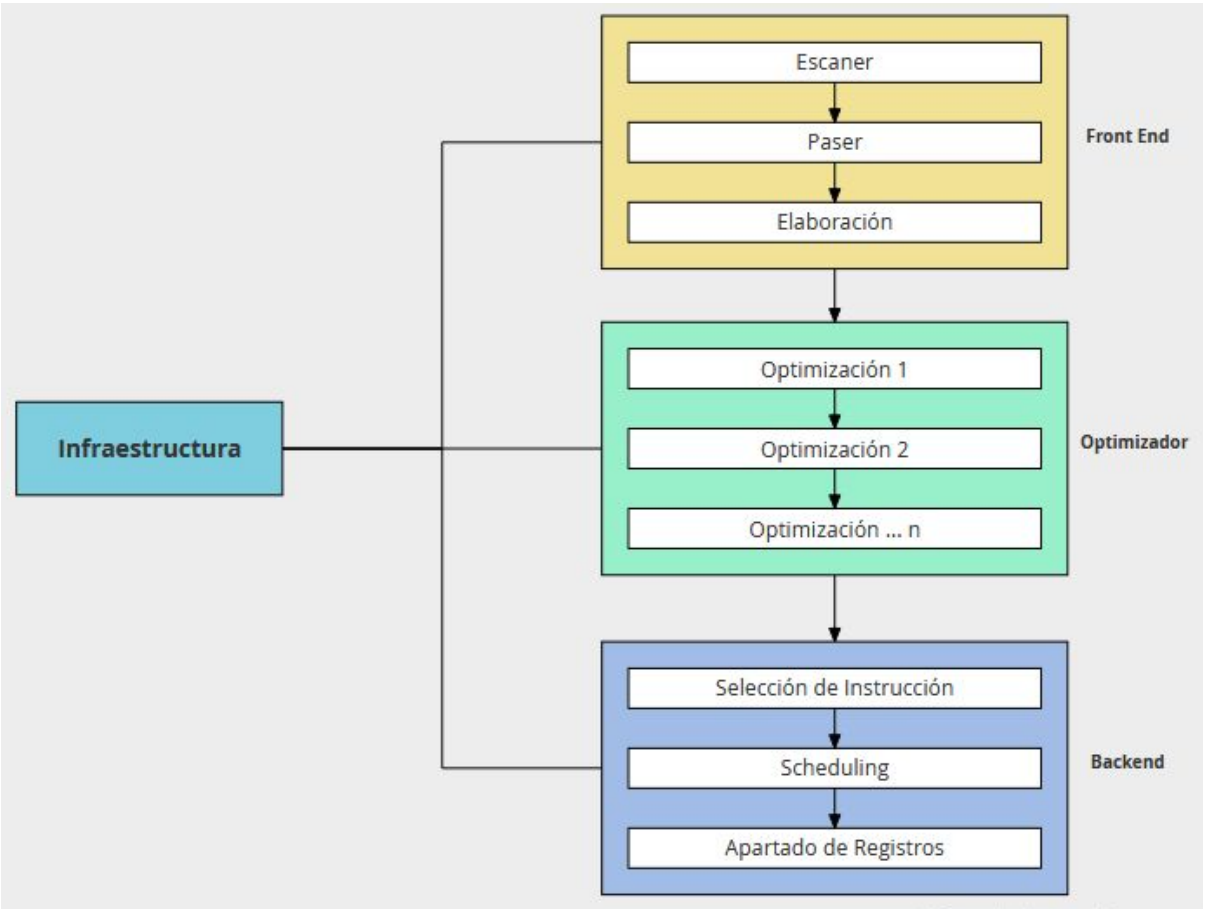
```
$ compilador git tag version
$Versión 1.0.0
```

ARQUITECTURA

El principal objetivo de nuestro compilador es preservar el significado de la entrada del programa y en algunos casos la mayoría de veces, debemos mejorar la salida del programa para hacer procesos más eficientes y con esto, utilizar de mejor manera los procesos de la computadora.



La infraestructura de nuestro nuestro compilador está basado en una base de 3 estructuras: Font end, Optimizador y Back end como se observa en el diagrama de manera lineal como se indican en las flechas.



La implementación del compilador se basa en el desarrollo tradicional de un compilador, es decir, análisis léxico (a su salida devuelve una lista de tokens), análisis sintáctico (analiza las sentencias mediante un árbol AST) y finalmente la generación de código (que genera el ensamblador de dicha entrada) el cual a su salida generará un archivo ejecutable.

Sistemas Operativos

El compilador está diseñado para correr y funcionar en sistemas Unix-like el cual cuenta con su propio manual para consultar cómo correr y compilar en cualquier momento.
El sistema debe contar con 2 prerrequisitos para poder correr el compilador.

Prerrequisitos

Software	• Elixir - Version 1.8.1 onwards • Erlang - Versión 19.0 • Git (Para poder clonar el repositorio) • Gcc (Para poder compilar y correr el programa)

ROADMAP

- 9 Abril (Primera entrega).
  Versión 1.0.0
  Para esta primera entrega el compilador, la capacidad de este es limitada y solo puede leer en el "return" un entero positivo.

```
int main( ) {
        return <entero>;
}
```

- 7 Mayo (segunda entrega).
  Versión ?
  En esta entrega el compilador debe ser capaz de leer en el "return" constantes positivas o negativas. Soporta las siguientes
    Negation '-'
    Bitwise complement '~'
    Logical negation '!='

```
int main( ) {
        return <constante>;
}
```

- 28 Mayo (Tercera entrega).
  Versión ??
  El compilador debe ser capaz de leer y hacer operaciones básicas:
    Adition "+"
    Subtraction "-"
    Multiplication "*"
    Division "/"

```
int main( ) {
        return <constante>;
}
```

- 7 junio (Entrega Final).
  Versión ??
  El compilador debe leer operaciones más complejas en el 'return':
  Logical AND '&&'
  Logical OR '||'
  Equal to '=='
  Not equal to '!='
  Less than '<'
  Less than or equal to '<='
  Greater than '>'
  Greater than or equal to '>='

  Ejemplo:

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

PLAN DE TRABAJO

- Primera entrega
  En esta semana se reunirá el equipo completo para analizar las especificaciones del programa (Project Manager, System Architect , System Integrator and System Tester), arquitectura, lista de requerimientos, sistemas en los que debe ejecutarse el programa, análisis de riesgos y el cómo se va a dividir el trabajo para que se cumpla el objetivo de la entrega en la fecha especificada.

- Segunda entrega .
  El equipo se reunirá para analizar los nuevos requerimientos de la entrega y se espera que se haga (minino) una entrega del avance del compilador (commit).

- Tercera entrega
  Se evaluará el riesgo de agregar los nuevos requerimientos al compilador (este código va por buen camino) y cómo impactaría al sistema (errores en cadena).

- Entrega final

MANUAL DE USUARIO
(Configuración, compilación y ejecución)

- Prerrequisitos
    - Tener sistema operativo Unix
    - Instalar Git
    - Instalar Elixir

    - Instalación de Git

      Para poder instalar Git podemos acceder a la página oficial de Git y encontrar las instrucciones para el sistema operativo Unix que tengamos.

      https://git-scm.com/download/linux

    - Instalación Elixir

      De igual forma podemos instalar Elixir a través de la página oficial:

      https://elixir-lang.org/install.html#unix-and-unix-like

      En esta página encontraremos la línea que debemos correr para que se nos instale Elixir.

❖ Compilacion y Ejecucion

Para obtener los archivos del compilador necesitamos primero clonarlos del repositorio de git, para esto abriremos la terminal y nos posicionaremos en la dirección en la que queremos que se clone el repositorio y pondremos la siguiente línea:

-git clone https://github.com/hiphoox/compilers2019_2.git

```
Cloning into 'compilers2019_2'...
remote: Enumerating objects: 151, done.
remote: Counting objects: 100% (151/151), done.
remote: Compressing objects: 100% (108/108), done.
remote: Total 1596 (delta 53), reused 112 (delta 41), pack-reused 1445
Receiving objects: 100% (1596/1596), 71.20 MiB | 13.24 MiB/s, done.
Resolving deltas: 100% (722/722), done.
```

Una vez descargado el repositorio tenemos que acceder a la carpeta descargada usando:

-cd compilers2019_2

Dentro de la carpeta tenemos que cambiar de branch para encontrar la carpeta del compilador:

-git checkout xforce

```
$ git checkout xforce
Switched to branch 'xforce'
Your branch is up to date with 'origin/xforce'.
```

Después accederemos a la carpeta del compilador

-cd compiler

Para poder usar el compilador primero necesitamos compilar el proyecto

-mix escript.build

Para correr las pruebas solo tenemos que poner la siguiente línea:

-mix test