Project Scope

Project Title: C Compiler on Elixir

Client: Ortigoza Márquez Norberto J.

Authors: Ruiz Aguilar Eduardo (Project Manager)

Aguilera Ortíz Alfredo (Integrator)

Hernández Escobar Oswaldo (Tester)

Rodríguez García Dulce Coral (Architect)

Due Date: March 17, 2020

Version: 1.0

Circulation: Internal

1. Introduction & Context

A compiler translates (or compiles) a program written in a high-level programming language that is suitable for human programmers into the low-level machine language that is required by computers. During this process, the compiler will also attempt to spot and report obvious programmer mistakes. Using a high-level language for programming has a large impact on how fast programs can be developed. The main reasons for this are:

- Compared to machine language, the notation used by programming languages is closer to the way humans think about problems.
- The compiler can spot some obvious programming mistakes.
- Programs written in a high-level language tend to be shorter than equivalent programs written in machine language.

A common division into phases is described below. In some compilers, the ordering of phases may differ slightly, some phases may be combined or split into several phases or some extra phases may be inserted between those mentioned below.

- Lexical analysis: This is the initial part of reading and analyzing the program text. The text is read and divided into tokens, each of which corresponds to a symbol in the programming language, e.g., a variable name, keyword or number.
- Syntax analysis: This phase takes the list of tokens produced by the lexical analysis and arranges these in a tree-structure (called the syntax tree) that reflects the structure of the program. This phase is often called parsing. An abstract syntax tree (AST) is one way to represent the structure of a program. In most programming languages, language constructs like conditionals and function declarations are made up of simpler constructs, like variables and constants. ASTs capture this relationship; the root of the AST will be the entire program, and each node will have children representing its constituent parts.

- Code generator: The intermediate language is translated to assembly language (a textual representation of machine code) for a specific machine architecture.
- Assembly and linking: The assembly-language code is translated into binary representation and addresses of variables, functions, etc., are determined.
- Optimizer: Although this stage is strictly required for any commercial compiler, we´ll just mention its functions in these lines because for this project purposes its implementation won´t be mandatory. The optimizer is responsible of finding and fixing semantic mistakes such as redundancy or unreachable stages of code.

2. Project overview & Scope

- Nora suggests a 32-bit architecture for the compiler, but in agreement with our client a 64-bit architecture was established.
- Compiler will be coded using Elixir programming language and capable of reading C files.
- The compiler will only handle single function programs (main function), consisting of a single return value at first, and then, capable of handling statements, only integer literals, neither hexadecimal nor octal.
- The verification of the compiler consists on compiling a series of programs using proven correctly automatized tests and checking if the compilations are successful.
- The compiler must produce the x64 assembly.
- The lexer accepts a file and returns a list of tokens.
- The parser accepts the list of tokens and transforms it into an abstract syntax tree (AST).
- For creating the AST, the parser is divided into four different functions, the next logic is used, “for each non-terminal element or derivation rule a function must be created” since our program only leads to four divisions, the functions are:
    - Parse_program()
    - Parse_function()

- Parse_statement()
- Parse_expression()

```
program = Program(function_declaration)
function_declaration = Function(string, statement) //string is the function name
statement = Return(exp)
exp = Constant(int)
```

- Since C grammar fits best as LL-type grammars our grammar construction algorithms will be for these types of grammars.
- Postorder traversal algorithm will be used for the AST.

3. Deliveries

The project division is based on the work presented by Nora Sandler on her website.

a. First Delivery: Integers

After this stage, the compiler must be able to compile a program that returns a single integer. We'll also set up the three basic passes of our compiler. Here's an example of a program we'd like to compile:

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

b. Second Delivery: Unary Operators

On this stage, three unary operators will be added, must be noticed that these operators only take one value.

- Negation (-)

  i.e. -5, -1, -7654, -86 in other words, a regular negative number.

- Bitwise complement (~)

This operator flips every bit in a number, for example:

- 4 is written as 100 in binary.
- The bitwise complement of 100 is 011.
- 011 in decimal is 3.
- So $\sim 4 = 3$.

- Logical negation (!)

  The boolean "not" operator. This treats 0 as "false" and everything else as "true".

  - $!0 = 1$
  - ! (anything else) $= 0$

c. Third Delivery: Binary Operators

For this stage, some binary operators will be added, must be noticed that these operators take two values.

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)

d. Fourth Delivery: More Binary Operators

Finally, eight more operators will be added, essentially logical operators like;

- Logical AND (&&)
- Logical OR (||)
- Equal to (==)
- Not equal to (! =)
- Less than (<)
- Less than or equal to (<=)
- Greater than (>)
- Greater than or equal to (>=)

*Note: Since the first delivery, must be noticed that updates in the lexing, parsing and code generation stages are mandatory for supporting the operators.

4. Out of Scope

Everything not specifically documented as in scope is explicitly out of scope.

5. Statement of Agreement

As the client/customer I´m in full awareness of agreement with the terms and conditions stablished in this document.