Software Compis Company

Compiler System (v1.0)
Week 1 Integers (First demo, April 9th, 6 weeks)

Specific Requirement Software (v1.0)

Index

1. Introduction

The present document intends to present the functional and non-functional characteristics that the final product is expected to fulfill, as well as the scope, the identification of requirements, of high level and low level, on which the successive stages of the project.

1.1 Purpose

As part of the analysis of the project, this document compiles the functional and non-functional requirements of the Compiler System and the systems that compose it, so that those involved in the project share the scope and details of this. It also describes the expected operating environment and the assumptions considered. The following document is intended for administrators and clients of the compiler system

1.2 Scope

This document describes the operation and functionality requirements of the systems that make up the system focused on the needs of the compiler.
The goal is to translate a written program (or text) into a "source" language, which we will call source program, into an equivalent in another language called "object", which we will call program or object code. If the source program is correct (belongs to the language formed by the correct programs), such a translation can be made.
The activities of project management, legal contracts, business analysis, and solution design are beyond the scope of this document.

1.3 Definitions, acronyms, and abbreviations

FR: Functional requirement
NFR: Non-functional requirements
PF: Product functions

1.4 References

Name	Description	Link
Work breakdown (v1.0) Road Map (v1.0)		WorkBreakdown&Roadmap.docx
Morgan. Kaufmann. Engineering. A. Compiler. 2nd.Edition		Morgan.Kaufmann.Engineering.A.Compiler.2nd.Edition.pdf

1.5 Global appraisal

This document describes in a general way the current problem and the conditions of the software solution to be developed. It is organized from the general to the particular, describing functional and non-functional considerations for the design and development of the product.

2. Global description

This section of the document describes some general factors that affect the product and requirements. This section does not declare the specific requirements. Instead, it maintains a background of those requirements that are defined in detail in the next section of the document to make it easier to understand.

2.1 Product perspective

In this delivery, the compiler system only contemplates a program that returns an integer value. We also unify and configure the basic subsystems of our compiler to obtain different structures based on a series of input instructions.

The product does not contemplate any other requirement that is not explicitly specified in this document.

2.2 Product functions

ID	Name	Description
PF_0	Compiler.py	The compiler is responsible for analyzing the type of input argument and calling each subsystem of the compiler
PF_1	file_clean.py	The file_clean reads a file from the directory to save the content in a source text
PF_2	lexer_module.py	The lexer_module (also called the scanner or tokenizer) is the phase of the compiler that breaks up a string (the source code) into a list of tokens. A token is the smallest unit the parser can understand.
PF_3	parser_module.py	The parser_module is transforming our list of tokens into an abstract syntax tree. An 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.
PF_4	linker_module.py	The load or linker editor will resolve calls to routines, including them from other library objects if necessary, and obtain absolute addresses, so that the absolute machine code executable will be available.
PF_5	code_generator.py	In this final phase the object code, usually relocatable or assembler machine code, is finally generated. Relative memory positions or registers are then selected for the variables and each statement of the intermediate code is translated into a sequence of instructions that execute the task

2.3 User characteristics

Operators (clients): Users familiar with the operation of compilers, programming languages and terminal management to run the system.

2.4 Restrictions

● The infrastructure and security of the compiler system must comply with the policies established by the documentation described in Nora Sandler's blog.
● The compiler system must be run on the UNIX operating system.

2.5 Attention and dependencies

● The client will provide the data and basic concepts of a compiler to build the compiler system.
● Any new requirements will be added to this document in future versions.

2.6 Prorate the requirements

In future versions, the program will have an error message (or several) that will allow you to determine the sources of the incorrectness as clearly as possible.

The architecture we define now will make it easy to add more language features later on.

3. The specific requirements

3.1 Functional requirements

ID	Name	Description
FR_0	Executable Program	The compiler generates an executable program from a source code
FR_1	UNIX instructions	Compiler instructions will comply with unix standards
FR_2	Instruction flag	This version only carries an entry flag.
FR_3	Executable name	-o will be the flag that will edit the name of the executable
FR_4	Assembler flag	-s or -- assembler will be the flag that will return only the assembly code. NO EXECUTABLE
FR_5	Tokens flag	-t or --tokens will be the flag that will return only the list of tokens. NO EXECUTABLE
FR_6	AST flag	-a or –ast will be the flag that will return the AST. NO EXECUTABLE
FR_7	Help flag	-h or -help will be the flag that will return the instructions to use and run the compiler. NO EXECUTABLE

3.2 Non-functional requirements

ID	Name	Description
NFR_0	Execution	Execution of the compiler by command line
NFR_1	UNIX S.O.	The compiler will only be executed in Unix OS

4 Annexes

4.1 Directory

Name	Position	Contact (Discord)
NORBERTO ORTIGOZA	CEO Compilers *2019-2*	hiphoox#1865
BUSTAMANTE HERNANDEZ LUIS FERNANDO	Integrator	Luis Bustamante#6714
MIRAMONTES SARABIA LUIS ENRIQUE	Architect	lauermeer#5479
LOPEZ GOMEZ NSERGIO ULISES	Tester	sergio#7167
MOLINA MEDINA MARCO ANTONIO	Project Manager	Marco.Molina#0678