**XENON COMPILER**

Development by Javeritos Inc.

National Autonomous University of Mexico

Faculty of Engineering

Computer Engineering

Compilers

Ing. Norberto Jesús Ortigoza Marqués

Developers

André Marqueda

Javier Solano

Alberto Castillo

Daniel Zarco

Business Requirements Document

# Project Details

|  |  |
| --- | --- |
| **Project Name** | Xenon compiler |
| **Project Type** | Phase II |
| **Project Star Date** | March 17 |
| **Project End Date** | April 14 |
| **Project Sponsor** | Norberto Ortigoza Márquez |
| **Division** | Compilers |
| **Project Manager** | Daniel Alberto Zarco Manzanares |

# Overview

This document defines the high-level requirements of Xenon compiler. It will be for the following activities:

1. Creative solutions design.
2. Developing test plans, test scripts, and their test subcases.
3. Determining project completion.

# Document sources

|  |  |  |
| --- | --- | --- |
| **Name** | **Business Unit** | **Role** |
| Norberto Ortigoza Márquez | Information Technology | Client |

# Purpose and Scope

This User Requirements Specification (URS) details of C language compiler (Initiative Xenon) which will be used to compile a source code wrote in C and execute the executable generated by compiler. The compiler must be supporting integers, unary and binary operators.

**Responsibilities**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Initials** | **Department** | **Responsibilities** | **Title** |
| Daniel Alberto Zarco Manzanare | DZ | Direction and Management | Manager | Project Manager |
| André Marqueda | AM | Architecture and Planning | Architect | Architect Design |
| Alberto Castillo | AC | Development | Developer | Dev Analyst |
| Javier Solano | JS | Version Management | Integrator | System Analyst |

# Design Requirements

|  |  |
| --- | --- |
| **Reference** | **Requirements** |
| **PHASE I** | |
| U1 | Compile a C source code and return a integer when executes de .exe file.  int main(){  return 25;  } |
| U2 | Assembly must write in 64-bits set instructions.   |  | | --- | | Lexing done!  .section \_\_TEXT,\_\_text,regular,pure\_instructions  .p2align 4, 0x90  .globl \_main ## -- Begin function main  \_main: ## @main  mov $25, %rax  push %rax  pop %rbx  ret  push %rax  pop %rbx  push %rax | |
| U3 | Development language must be a matching pattern to easily build an Abstract Syntax Tree (AST), however, phase I the right side’s tree must be nil.   |  | | --- | | %Arbol{  hijoIzq: %Arbol{  hijoIzq: %Arbol{  hijoIzq: %Arbol{  hijoIzq: nil,  hijoder: nil,  nodopadre: :constant,  valor: 25  },  hijoder: nil,  nodopadre: :statement,  valor: :return  },  hijoder: nil,  nodopadre: :funcion,  valor: :main  },  hijoder: nil,  nodopadre: :program,  valor: nil  } | |
| U4 | Source code must have main function where return line code has a decimal integer. |
| U5 | Source code have a single function calls main() which return a decimal integer.  int main(){  return 25;  } |
| U6 | The return value only be a decimal integer and can be a variable into a decimal range.  return int 25 |
| U7 | The assembly syntax must be a AT&T by default in GCC. |
| U8 | The parser (scanner) must show token’s list form source code. Must check a relational couple to recognize every token.   |  | | --- | | {:type, 1, [:intKeyWord]},  {:ident, 1, [:mainKeyWord]},  {:lParen, 1, []},  {:rParen, 1, []},  {:lBrace, 1, []},  {:ident, 2, [:returnKeyWord]},  {:num, 2, 25},  {:semicolon, 2, []},  {:rBrace, 3, []} | |
| **PHASE II** | |
| U8 | Compiler must support bitwise, negation, and logical negation operators. Following the form  Negation  int main(){  return -1;  }  Bitwise operator (Complement to one)  int main(){  return ~5;  }  Negation operator (0 is false, 1 is true)  int main(){  return !-5;  } |
| U9 | Compiler’s output shows negative number when source code has the statement  int main(){  return -5;  }  Bash$ -5 |
| U10 | Compiler’s output shows complement to one when bitwise operator is called in source code  Bitwise operator (Complement to one)  int main(){  return ~5;  }  Bash$ -6 |
| U11 | Compiler’s output shows logical negation by cero (0) and one (1), to false and true, respectively.  Negation operator (0 is false, 1 is true)  int main(){  return !5;  }  int main(){  return !!5;  }  Bash$ 0  Bash$ 1 |

# Documentation and regulation

|  |  |
| --- | --- |
| **References** | **Requirements** |
| U12 | Nora Sandler’s compiler tutorial |
| U13 | Norberto’s information classroom |

# Change History

|  |  |  |  |
| --- | --- | --- | --- |
| **Edition** | **Effective date** | **Description of change** | **Revised without changes**  **Date / Sign** |
| 1.1 | February 27 | Start Phase I | March 2 |
| 1.1 | March 16 | Final Phase I | March 14 |
| 1.2 | March 17 | Start Phase II | March 20 |
| 1.2 | April 14 | Final Phase II | April 13 |

**Approbation and Validation**

|  |  |
| --- | --- |
| **Prepared by Management Department** |  |
| Daniel Alberto Zarco Manzanares |  |
| Project Manager | sign |
|  |  |
| **Reviewed and approved by:** |  |
| Norberto Ortigoza Márquez |  |
| Client | sign |