**Software Design Document (SDD)**

**Assignment #2 Assembler for XE Computer**

**CS530, Fall 2020**

**Team:**

Aria Kafie cssc4055

Htet Hnin Su Wai (Su), cssc4043

**Overview & Goals:**

The goal of this project is to develop a limited two-pass assembler for the XE variant of the SIC/XE family of machines. The assembler, named “asxe”, will take SIC (XE only) assembler source files as input and generate listing files for the XE machine. It will implement all XE features and assembler directives up to and including section 2.3.4 of the text (which means, no Control Sections, EXTDEF and EXTREF). The assembler will not generate object code files but will produce listing files (.l) and a symbol table file (.st) for each source file provided.

**Project Description:**

The project involves developing a software tool that translates assembly language code written for the XE variant of the SIC/XE variant of the XIC XE family of machines into listing files while also generating a symbol table. The tool will utilize a limited two-pass approach to accomplish this task efficiently.

**Plan of Action and Milestones:**

*Phase One: Outline/Brainstorm (March 4th - March 11th)*

* Identify the final goals for the project and software implementation.
* Discuss timeframe and team roles/workload distribution.
* Brainstorm potential challenges and solutions.
* Determine the scope of the project and establish milestones.

*Phase Two: Design Architecture (March 12th - March 20th)*

* Design the overall architecture of the assembler.
* Create a Data Flow Diagram (DFD) outlining the flow of data and control.
* Define the structure of the assembler components and their interactions.
* Determine coding roles and responsibilities based on the architecture.

*Phase Three: First Pass Implementation (March 21st - March 28th)*

* Implement the first pass of the assembler according to the designed architecture.
* Develop code to read input files, parse instructions, and build symbol tables.
* Ensure accurate generation of intermediate files containing processed instructions.

*Phase Four: Second Pass Implementation (March 29th - April 5th)*

* Implement the second pass of the assembler to generate listing files.
* Resolve symbols and generate machine code for instructions.
* Write code to produce listing files with assembled instructions, addresses, and errors.

*Phase Five: Test Preparation (April 6th - April 10th)*

* Create test files to cover various scenarios and edge cases.
* Develop a comprehensive testing plan outlining test cases and expected outcomes.
* Review and finalize the test environment setup.

*Phase Six: Testing and Debugging (April 11th - April 14th)*

* Execute test cases to verify the functionality and correctness of the assembler.
* Debug and refine the assembler based on testing results.
* Address any issues or discrepancies found during testing.

*Phase Seven: Documentation and Finalization (April 15th - April 18th)*

* Document the usage and functionality of the assembler.
* Write a comprehensive README file detailing installation instruction, usage guidelines, and any dependencies.
* Finalize the project, ensuring all requirements are met.
* Prepare for final submission on April 18th.

**Requirements:**

* Knowledge of C/C++ Programming Language.
* C++ Programming Language and GCC Compiler
* MAC OS or Windows 10 Operating Systems.
* Must be able to convert the source file swiftly.

**System Design/Specification:**

Note – this can include User Stories, Object Design, or Structured Design. You can use diagrams and text, pictures (of work on white boards/etc) and text, be creative and have fun.

**First Pass:**

Read input source file line by line.

Parse each line and extract labels, operation codes, operands, and comments.

Generate the location counter (LOCCTR) for each instruction.

Build the symbol table with labels and corresponding addresses.

Generate an intermediate file containing processed instructions.

**Second Pass:**

Read the intermediate file generated in the first pass.

Resolve symbols and generate machine code for instructions.

Generate the listing file with assembled instructions, addresses, and any errors encountered.

Write the symbol table to a separate file.

**Development Environment:**

Operating System: Windows 10

Integrated Developing Environment: Visual Studio Code

Compiler: GCC Compiler  
Edoras from SDSU

Version Control: Git

**Run/Test Environment:**

We will run and test the environment on GCC Compiler and generate test cases to ensure the program runs successfully. For the final product, the program will be compiled and tested on Edoras server to ensure we do not have any errors.

**Code and Test Plan:**

1. Code Review: The code will be reviewed by both team members to ensure the accuracy of syntax and satisfaction of all requirements.
2. Unit Testing: Each method in the program will be tested individually to make sure each method works accordingly.
3. Integration Testing: Combined code will be tested with test files generated to make sure the entire program works as intended.