**Software Design Document (SDD)**

**Assignment #2 DXE Disassembler for XE computer**

**CS530, Spring 2020**

**Team:**

Ron Landagan, cssc1917 (Team Lead),

Wiliton Rogdrigues, cssc1922 (Developer),

Joshua Vargas, cssc1973 (Developer),

Isaiah Dorado, cssc1929 (Developer)

**Overview & Goals:**

The goal of this project is to develop, test, and deliver a disassembler program for the XE variant of the SIC/XE family of machines. This process will take place over the course of seven weeks, and should include C++ source files, header files, a Makefile, and a README file.

**Project Description:**

The disassembler program will take in two files: an XE object file and its accompanying symbol file. From these two files, the disassembler will reverse-engineer the software such that it generates two additional files: an XE source file and an XE listing file. If the required files aren’t available, then the program will exit.

The main program will consist of a reader that will read the XE object file, and translate each instruction, step-by-step, writing the XE listing file and referring to the symbol file when necessary. After the listing file has been fully written, it will be modified to create the XE source file.

**Plan of Action and Milestones:**

The group will keep in touch through text messages, and documents are shared through Google Drive. The code will be modified through a Github repository. The milestones are laid out as such:

**(Week 1) March 05 –** Project group’s first meeting. Establish goals and group structure. Begin establishing program structure.

**(Week 2) March 12 –** Finalize and review Software Design Document. Finalize program structure and begin compiling a list of programming tasks.

**(Week 3) March 19 –** Finalize list of programming tasks and begin assigning them to various project team members.

**(Week 4) March 26 –** Continue programming and implementing programming tasks.

**(Week 5) April 02 –** Finalize programming phase and begin to plan testing phase.

**(Week 6) April 09 –** Write various sample files and begin testing the program against them. Begin writing Makefile and README files.

**(Week 7) April 16 –** Finalize testing and troubleshooting. Finalize Makefile and README files.

**Requirements:**

- be written in C++

- open both an XE object file, <filename>.obj, and its symbol file, <filename>.sym

- generate both an XE source file, <filename.sic>, and an XE listing file, <filename>.lis

- executable program shall be named “dxe”

- compiled using a Makefile

- all files will contain team members’ names and RedIDs

- Project team will also turn in a README file as per the instructions on Blackboard

**System Design/Specification:**

assignment2.cpp will follow the following structure:

**begin main algorithm**

**if** <filename>.obj and <filename>.sym exist **then**

**begin** building LIS file

initialize LIS file to be empty

**begin** analyze <filename>.obj

**while** line from OBJ file exists **do**

read first character

**if** char = ‘H’ **then**

**begin** reading header record

insert starting address and control section title into LIS file

insert current address, program name, and START statement into LIS file

**end** (read header record)

**if** char = ‘T’ **then**

**begin** reading text record

read current address

**while** text record != empty **do**

initialize full statement string to empty

append current address to full statement

**if** current address holds a label in the SYM file **then**

append appropriate label to full statement

**else**

append a blank space to full statement

**if** current address belongs to a literal **then**

append appropriate literal to full statement

**else**

append appropriate instruction and operand to full statement

insert full statement into LIS file

increment the current address by the appropriate amount

erase appropriate number of characters from record

**end** (read text record)

**if** char = ‘E’ **then**

**begin** reading end record

insert end statement into LIS file

**end** (read end record)

**end** (OBJ file analysis)

**begin** analyze <filename>.sym

**end** (SYM file analysis)

**end** (build LIS file)

**begin** building SIC file

**while** line from LIS file exists **do**

write the substring of line between characters 6 and 32 into SIC file

**end** (build SIC file)

**else**

print error to console

**end** (main algorithm)

**Development Environment:**

- Choice in IDE will depend on the team member’s personal preference

- Project will be compiled using the G++ compiler

**Run/Test Environment:**

- Weeks 6 and 7 will be dedicated to testing

- Testing will use the Makefile for compilation

- Sample files written on week 6 will be used for testing