**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:**

The program will follow the following structure:

1. Build LIS File
   1. Open **OBJ** file + Open **SYM** file
   2. Read Header Record (Read col 1 “H”)
      1. Read Program name in col 2-7
         1. Write program name in first line
      2. Read starting address in col 8-13
         1. Initialize counter variable
      3. Read length of object program in col 14-19
         1. Set max length for counter
      4. Begin writing LIS file
   3. Read Text Record (Read col 1 “T”)
      1. Read first address in col 2-7
      2. Read length of object code in record in col 8-9
         1. Save into a variable
      3. Read Object Code in col 10-69
         1. Write current address to LIS file in col 1-4
            1. If current address matches a label in the SYM table, write label in col 7-12
         2. Read first 3 characters
         3. Convert 3 characters to binary
         4. Read first 6 bits
            1. Get Opcode from the OBJ file
            2. Get format from Opcode
            3. If format is 3 or 4, use the bits 7 to 12 tho find the *nixbpe* flags
            4. Return instructions format
         5. Read full instruction
            1. Get all *nixbpe* flags

Find operand using flags

* + - * 1. Write Instruction and operand

Write instruction in col 15-20

Write operand in col 23-31

* + - * 1. Write Object code in col 33-end
      1. Increment counter variable by instruction size
      2. Repeat until text record is finished
    1. Go to next line
  1. Read Modification Record (Read col 1 “M”)
  2. Read End record (Read col 1 “E”)
     1. Write address
     2. Write “END” instruction
     3. Write program name
  3. Read Sym Table to fill in the rest of labels and assembler directives.

1. Build SIC File
   1. Open LIS file
   2. Begin writing SIC file
      1. Read through character-by-character
      2. Write only source statements into SIC file

**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