/**
*

- Members: John Jacobsen, Carson Elmer, Daniel Valoria
- Class User Accounts: cssc0400, cssc0401, cssc0402
- REDIDs: 820405580, 820252778, 820103915
-
- CS530, Spring 2019
- Assignment #2, XE Disassembler
- Filename: SoftwareEngineeringDocument.pdf
- Purpose: Provide detailed description of how the project works.

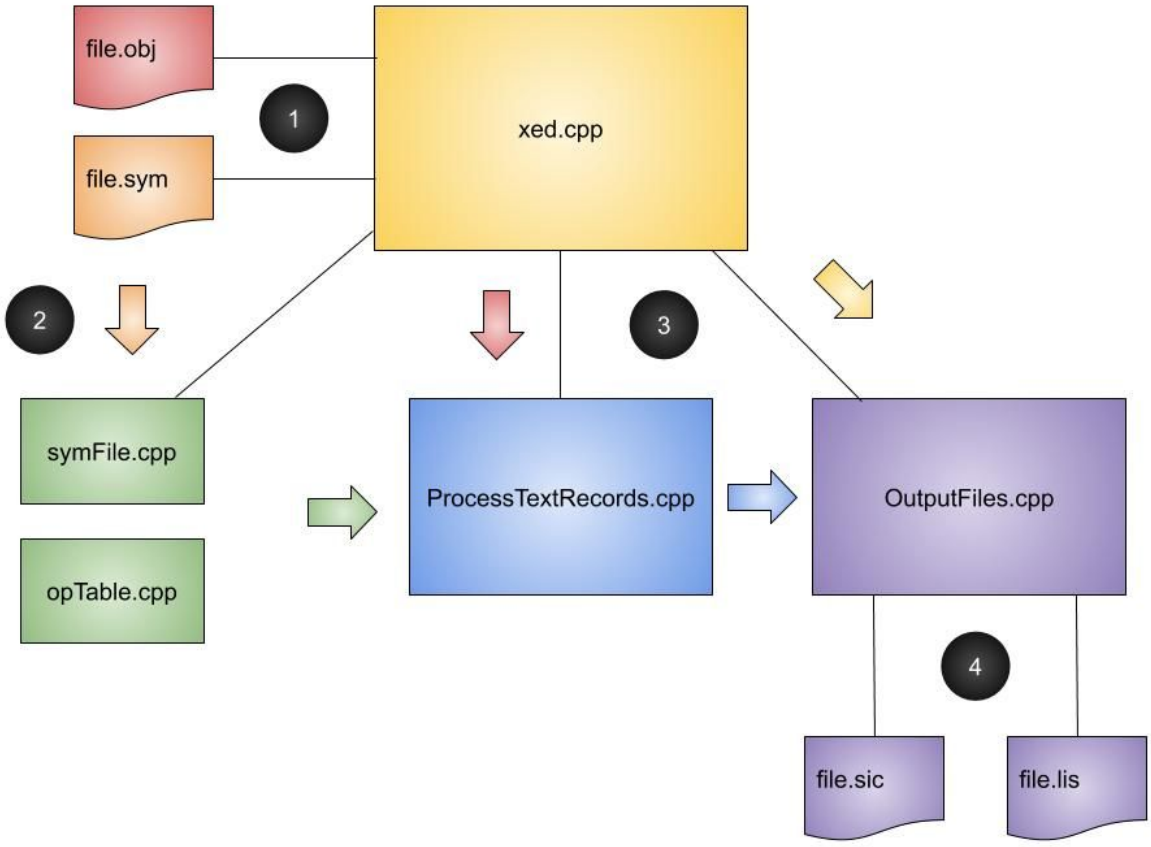
****
/

I. System Planning
   A. Task Assignments
      1. John: Reading and Processing Input Files (file.obj, file.sym)
      2. Carson: Generating and Preparing Output Files (file.sic, file.lis)
      3. Daniel: Disassembling Records
   B. Notes
      1. We used Increment/Evolution development to build our project because we did not know what and how much we would need for each of the core tasks:
         a) Reading and Processing Input Files
         b) Disassembling the Records
         c) Writing the disassembled code to Output Files
      2. Although we began working independently, the group worked together and helped each other on all of the tasks.
      3. The group regularly uploaded updated code to our GitLab repository.
   C. Timeline
      1. Week 1 (3/13 - 3/20):
         a) Creating Modules/Assignment List
         b) Draft Rough Timeline
      2. Week 2 (3/20 - 3/27):
         a) Planning System Design
         b) Working Independently on Assigned Tasks
      3. Weeks 3&4 (3/27 - 4/10):
         a) Working Independently on Assigned Tasks
         b) Completing Project

4. Weeks 5&6 (4/10 - 4/17):
    a) Verification/Testing
    b) Generating Makefile, README.md, Software Design Document

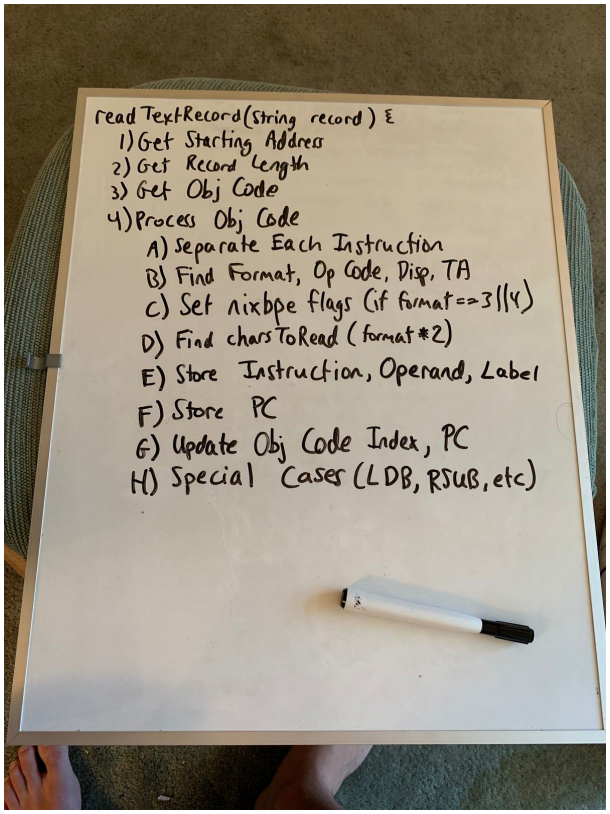
II. System Design

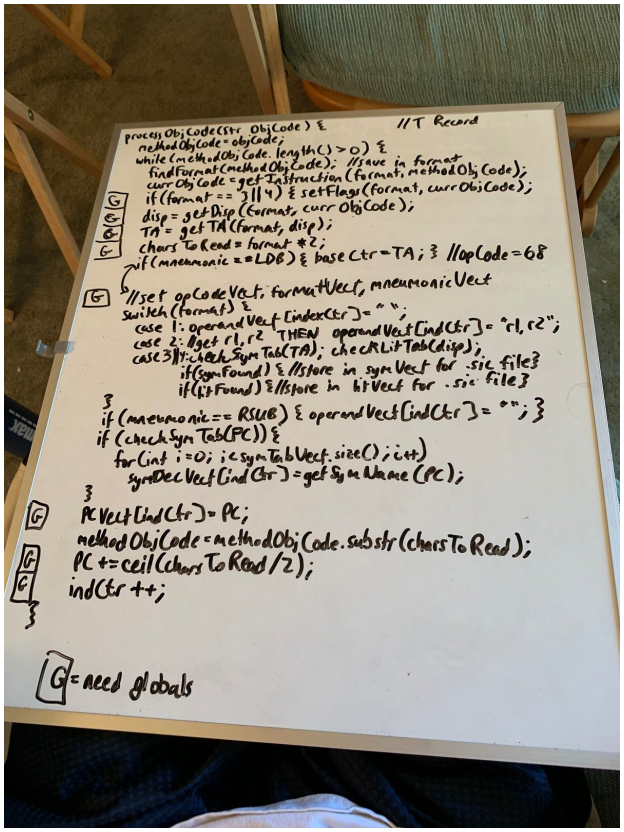
A. Basic Structure

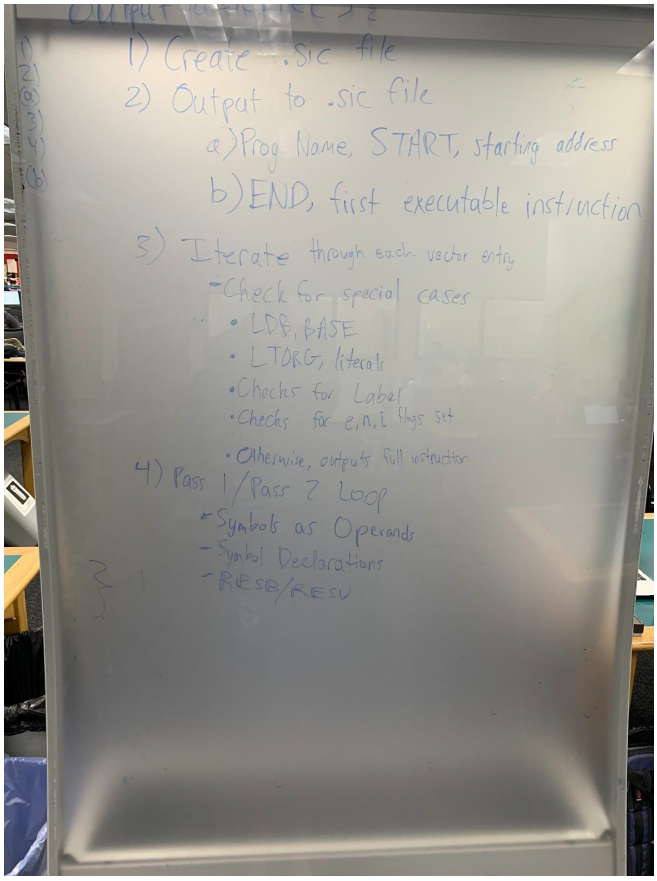


1) *xed.cpp* (driver program) takes in *file.obj* and *file.sym* as input.
2) *symFile.cpp* process the information from *file.sym*. *opTable.cpp* holds the OPTAB.
3) *ProcessTextRecords.cpp* uses the text record from *xed.cpp*, as well as the SYMTAB and OPTAB from *ReadTables.cpp* to disassemble the text record.
4) *OutputFiles.cpp* outputs the source code (*file.sic*) and its accompanying listing file (*file.lis*) using the disassembled records from *ProcessTextRecords.cpp* and *xed.cpp*.

Pseudo:


readTextRecord(String record) {
1) Get Starting Address
2) Get Record Length
3) Get Obj Code
4) Process Obj Code
A) Separate Each Instruction
B) Find Format, Op Code, Disp, TA
C) Set nixbpe flags (if format==3||4)
D) Find charsToRead (format*2)
E) Store Instruction, Operand, Label
F) Store PC
G) Update Obj Code Index, PC
H) Special Cases (LDB, RSUB, etc)


processObjCode(Str ObjCode) { //T Record
methodObjCode = objCode;
while(methodObjCode.length() > 0) {
findFormat(methodObjCode); //save in format
currObjCode = getInstruction(format, methodObjCode);
if(format == 3||4) { setFlags(format, currObjCode);
disp = getDisp(format, currObjCode);
TA = getTA(format, disp);
charsToRead = format*2;
if(mnemonic == LDB) { baseCtr = TA; } //opCode = 68
//set opCodeVect, formatVect, mnemonicVect
switch(format) {
case 1: operandVect[indexCtr] = "";
case 2: //get r1, r2 THEN operandVect[indCtr] = "r1, r2";
case 3||4: checkSymTab(TA); checkLitTab(disp);
if(symFound) { //store in symVect for .sic file }
if(litFound) { //store in litVect for .sic file }
}
if(mnemonic == RSUB) { operandVect[indCtr] = ""; }
if(checkSymTab(PC)) {
for(int i=0; i<symTabVect.size(); i++)
symDecVect[indCtr] = getSymName(PC);
}
PCVect[indCtr] = PC;
methodObjCode = methodObjCode.substr(charsToRead);
PC += ceil(charsToRead/2);
indCtr++;
}
G = need globals



B. Core Files:

1. **opTable.cpp**
   Purpose: Holds the OPTAB.
2. **OutputFile.cpp**
   Purpose: Format output for .sic and .lis file.
3. **ProcessTextRecord.cpp**
   Purpose: Disassemble Text Records from the .obj file.
4. **symFile.cpp**
   Purpose: Reads .sym file, produces SYMTAB and LITTAB.
5. **xed.cpp**
   Purpose: Driver program. Processes .obj file for disassembly.

C. Additional Files:

1. **opTable.h**
   Purpose: Header file for opTable.cpp.
2. **OutputFiles.h**
   Purpose: Header file for OutputFiles.cpp.
3. **ProcessTextRecord.h**
   Purpose: Header file for ProcessTextRecord.cpp.
4. **symFile.h**

Purpose: Header file for symFile.cpp.

5. **xed.h**
   Purpose: Header file for xed.cpp.
6. **Makefile**
   Purpose: Compile program.
7. **README.md**
   Purpose: Document key parts of the project.

III. Verification & Test Design

A. Error Checking Methods:

1. Ran and debugged the sample files provided on blackboard.
2. Ran and debugged online example SIC/XE programs.
3. Running through the program by hand to verify our results.

B. Verification (Test) Files:

1. **sample.obj**
2. **sample.sym**
3. **sample.sic**
4. **sample.lis**
5. **sample2.obj**
6. **sample2.sym**
7. **sample2.sic**
8. **sample2.lis**