# CS530Assignment

Amiel Nava RedID:824264864

Chase Parker RedID:821253141

Kenny Nguyen RedID:821649498

Jeremy Espinosa RedID:821770060

The program runs with the following command:

./dissem sample-obj.txt sample.sym.txt

Will output:

list.txt

The attached Zip file contains our Disassembler code.

The purpose of these are to read in Object codes and Symbol files while an Assembly listing file is the output.

The program uses input.cpp to read in the Object codes and Symbols files.

Using these the Disassembler class is created to decode these into an Assembly file.

When testing within our main file we would call upon the input.cpp and Disassembler.cpp to produce our output,

The Disassembler calls upon header\_record.cpp, mod\_record.cpp, and end\_record.cpp to read the inputted files and

generate the output from them.

## First Diagram

Setting up a plan for our assignment

We also split up the work by input, header record, text record, end record and output. We ended up changing some variable types and method names.

input.cpp main.cpp read\_obj-file (intarge, char \*\*argv) call read Aptermine if its text, neader, modifier, or end record and call that function to process line call output neader.h read\_sym\_file(intarge, char \*\* argv) OPSEJ insertsymbols( mnemonics[] Struct Format? text-record CPP parse text-line (char \*\*line) int type process\_obj-code (int byte-length, charmon code) validate-obj-code (char \*\*opcode) int address char \*lubel char \*mnemonic int operand - address header\_record.cpp int op-code Purse\_header\_line (char \*\* line) mod record cpp

Parse\_mod\_line (char \*line) Hash-table: modify-addiess (int add-loc) end\_record.cpp Key -> add less parseend-line (char miline) Value -> format structure output. cpp print-table ( Write-File L Format 3 & address, label, runemonic, operand, object code}

Diagram 2

Setup a structure for the assembly line code that way the output method can just iterate assembly code lines and print it or write it.

Struct Instruction line?

int type;
string address;
String unemonic;
String unemonic;
String operand Address;

Diagram 3

Put all parts together to have a running program

```
main (argc, argv)

Disassembler disassembler = inputMethod(argc, argv)

outputMethod (disassembler);

inputMethod (argc, argv)

line = readLine(

Dissasembler disassembler = header Method (line, &disassembler);

// loop all text lines

line = read Line()

process TextLine (line, &disassembler);

// loop all modifier lines

line = read Line()

modify Method (line, &disassembler);

line = read line()

end Method (line, &disassembler);

Yeturn disassembler;
```