Single Programmer Affidavit
I the undersigned promise that the attached assignment is my own work. While I was free to discuss ideas with others, the work contained is my own. I recognize that should this not be the case, I will be subject to penalties as outlined in the course syllabus.
Your Name Here: Andrick Mercado

File Name: main.cpp
Functions:

(int) main: This program accepts two files of type .obj and .sym, and it has to be in this order otherwise the code won't work, and after it checks if it has received two inputs we call file_manager.

(void) file_manager: Parameters are both the .obj and .sym files and use global variables, makes sure files exist, and if they do we extract the lines of each file to corresponding vectors after it calls dissem.

(void) Dissem: No parameters uses global variables, traverses all lines in the obj_lines vector and checks if the first char is one of this 'H', 'T', 'E' and calls the corresponding function that extracts the info

(void) header_record: No parameters and uses global variables, saves the program length in a variable as well as current address the given address in the header finally prints it out in the out.lst file with lis_ofstream.

(void) text_record(int line): Params: current line we are extracting the object code from. This function loops through every object code based on the given text_lenth from the text record, we iterate through every object code and call the function instruction_disasembler according to each object code, this function returns the number that corresponds to the length of the given object code. After finishing the while loop we check if RESB should be used, given that the next line is a text record if it's not a text record it calculates the RESB to be added based on the public variable program_Length.

(void) end_record: prints the end as well as the first program name

(int) instruction_disasembler(int line, int current_index): (int) text_record(int line, int current_index): return length of object code (int) line: current line we are extracting the object code form (int) current_index: pointer or index of current object code start point. This function initially checks if lit should be added on the current line or address if it does find a match of current_address and lit_addresses and lit_mentions is true (only true when previously reference without yet being implemented) we proceed to add a new lit with ltorg procedure. The next for loop is to check if lit should be inserted using the byte procedure only if the current address matches the lit_addresses. Lastly, we check the opcode of the instruction and retrieve its format, after we call the corresponding function based on the format. (returns type int) In this case will be the length of the object code.

void format_2(int opCode, int line, int current_index): (int) opcode: holds the opcode value (int) line: current line we are extracting the object code form (int) current_index: holds the indices of the current object code. This function initially fins the opcode name based on the given value from the parameter, after we loop the addresses we either print the label and the opcode or just the opcode. finally, we print the register and its corresponding address.

(int) format_3_and_4(int opCode, int line, int current_index) (int) opcode: holds the opcode value (int) line: current line we are extracting the object code from (int) current_index: holds the indices of the current object code @returns int which corresponds to the length of the object code 3 or 4 This function initially finds the mnemonic given the opcode value, after we get the nixbpe from the object code, next we see if a label needs to be placed or not (can be from lit or sym), next we get the corresponding displacement to check if it's in twos-complement if not we keep the regular value. Everything after that has its corresponding comments essentially where we determine the addressing mode in order to print out the correct mnemonic and operand.

*** Full Explanations in the main file ***
made use of this website for references to finding and disasembling object code

https://www.unf.edu/~cwinton/html/cop3601/supplements/sicsim.page.html

General explanation here:

```
//this two functions are to extract information
void file_manager(char *objFile);
void dissem();

//this three functions disasemble each object code depending on record type
void header_record(int line);
void text_record(int line);
void end_record();

//this will be called by text_record and will return address length
int instruction_disasembler(int line, int current_index);

//this two functions extract the object code from current index of a given line and print out results
void format_2( int opCode, int line, int current_index);
int format_3_and_4( int opCode, int line, int current_index);
```

main.h: contains the header file for main, in it are delclarations for all functions in this class as well as global varriables there are few comments explaining each, lastly we have a struct which holds all the sic/xe Addressing Modes slide (1.3.2 SIC/XE Addressing Modes, slide 66).

Mnemonic	Format	Opcode
ADD m	3/4	18
ADDF m	3/4	58
ADDR r1,r2	2	90
AND m	3/4	40
CLEAR r1	2	4
COMP m	3/4	28
COMPF m	3/4	88
COMPR r1,r2	2	A0
DIV m	3/4	24
DIVF m	3/4	64
DIVR r1,r2	2	9C
FIX	1	C4
FLOAT	1	C0
HIO	1	F4
J m	3/4	3C
JEQ m	3/4	30
JGT m	3/4	34
JLT m	3/4	38
JSUB m	3/4	48
LDA m	3/4	00
LDB m	3/4	68
LDCH m	3/4	50
LDF m	3/4	70
LDL m	3/4	08
LDS m	3/4	6C
LDT m	3/4	74
LDX m	3/4	04
LPS m	3/4	D0
MUL m	3/4	20
MULF m	3/4	60
MULR r1,r2	2	98
NORM	1	C8
OR m	3/4	44
RD m	3/4	D8
RMO r1,r2	2	AC
RSUB	3/4	4C
SHIFTL r1,n	2	A4
SHIFTR r1,n	2	A8
SIO	1	F0
SSK m	3/4	EC
STA m	3/4	0C
STB m	3/4	78
STCH m	3/4	54
STF m	3/4	80
STI m	3/4	D4
STL m	3/4	14
STS m	3/4	7C
STSW m	3/4	E8
STT m	3/4	84
STX m	3/4	10
SUB m	3/4	1C

https://www.unf.edu/~cwinton/html/cop360

SUBF m	3/4	5C
SUBR r1,r2	2	94
SVC n	2	B0
TD m	3/4	E0
TIO	1	F8
TIX m	3/4	2C
TIXR r1	2	B8
WD m	3/4	DC

- Here we see we can store in this manner

{Mnemonic, opcode, format}

for each entry thats Not format 1

Libraries included for this program:

```
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <cstdlib>
#include <vector>
#include <iomanip>
#include <bitset>
```

Global Variables:

```
ifstream obj_ifstream;//.obj file input stream
ifstream sym_ifstream;//.sym file input stream
ofstream lis_ofstream;//.lis file output stream

vector<string> obj_lines;//saves all lines into this vector from the .obj file
vector<string> sym_lines;//saves all lines into this vector from the .sym file
vector<string> lit_Const;//saves the lit constants
vector<string> lit_Names;//saves the lit names
vector<int> lit_Length_Address;//saves the lit lengths
vector<int> lit_Mentions;//in order to see if we have referenced it before assingment (THE LIT)
vector<unsigned int> lit_Addresses;//holds the lit addresses
vector<string> sym_Symbol_Names;//holds symbol names of .sym file
vector<unsigned int> sym_Adresses;//holds symbol addresses of .sym file
unsigned int program_Length;//holds the program length given by the header record
unsigned int current_Address;//will always hold the current address of each line or object code
unsigned int current_Base_Address;//will always hold the value given by ldb
```