```
/*
__/\\\\\\\\\\\\\_____/\\\\\\\\\\\__/\\\\\\\\\\\\____
 _\/\\\/////////\\\__\/////\\\///__\/\\\////////\\\__
  _\/\\\_______\/\\\______\/\\\_____\/\\\______\//\\\_
   _\/\\\\\\\\\\\\\/_______\/\\\_____\/\\\_______\/\\\_
    _\/\\\/////////_________\/\\\_____\/\\\_______\/\\\_
     _\/\\\__________________\/\\\_____\/\\\_______\/\\\_
      _\/\\\___________/\\\___\/\\\_____\/\\\_______/\\\__
       _\/\\\__________\//\\\\\\\\\______\/\\\\\\\\\\\\/___
        _\///____________\/////////_______\////////////_____

-> Name:  s19_maker.h
-> Brief: Header file for s19_maker.cpp
-> Date: May 26, 2017   (Created)
-> Author: Paul Duchesne (B00332119)
-> Contact: pl332718@dal.ca
*/

#ifndef S19_MAKER_H
#define S19_MAKER_H

void init_srec(unsigned int address); // Called to start each new srec

void output_srec_buffer(); // Emits the current buffer of srecords: Includes emitting S1 hex code (0x5331)

void write_srec_byte(unsigned char byte);

void write_srec_word(unsigned short word);

void write_S9(unsigned int s9_srec_address);

#endif
```

```
/*
__/\\\\\\\\\\\\\_____/\\\\\\\\\\\__/\\\\\\\\\\\\____
 _\/\\\/////////\\\__\/////\\\///__\/\\\////////\\\__
  _\/\\\_______\/\\\______\/\\\_____\/\\\______\//\\\_
   _\/\\\\\\\\\\\\\/_______\/\\\_____\/\\\_______\/\\\_
    _\/\\\/////////_________\/\\\_____\/\\\_______\/\\\_
     _\/\\\__________________\/\\\_____\/\\\_______\/\\\_
      _\/\\\___________/\\\___\/\\\_____\/\\\_______/\\\__
       _\/\\\__________\//\\\\\\\\\______\/\\\\\\\\\\\\/___
        _\///____________\/////////_______\////////////_____

-> Name:  parser.h
-> Brief: Header file for parser.h.cpp
-> Date: May 18, 2017
-> Author: Paul Duchesne (B00332119)
-> Contact: pl332718@dal.ca
*/

#include "library.h"

#ifndef PARSER_H
#define PARSER_H

ADDR_MODE parse(std::string op, int& value0, int& value1);

#endif
```

```
/*
__/\\\\\\\\\\\\\_____/\\\\\\\\\\\__/\\\\\\\\\\\\____
 _\/\\\/////////\\\__\/////\\\///__\/\\\////////\\\__
  _\/\\\_______\/\\\______\/\\\_____\/\\\______\//\\\_
   _\/\\\\\\\\\\\\\/_______\/\\\_____\/\\\_______\/\\\_
    _\/\\\/////////_________\/\\\_____\/\\\_______\/\\\_
     _\/\\\__________________\/\\\_____\/\\\_______\/\\\_
      _\/\\\___________/\\\___\/\\\_____\/\\\_______/\\\__
       _\/\\\__________\//\\\\\\\\\______\/\\\\\\\\\\\\/___
        _\///____________\/////////_______\////////////_____

-> Name:  first_pass.h
-> Brief: Header file for first_pass.h
-> Date: May 21, 2017   (Created)
-> Author: Paul Duchesne (B00332119)
-> Contact: pl332718@dal.ca
*/

#ifndef FIRST_PASS_H
#define FIRST_PASS_H

void first_pass();

bool is_last_token(); // True if the token previously parsed was the last token, false if there
                                        // is another token after

void error_detected(std::string error_msg);

#endif
```

```
/*
__/\\\\\\\\\\\\\_____/\\\\\\\\\\\__/\\\\\\\\\\\\____
 _\/\\\/////////\\\__\/////\\\///__\/\\\////////\\\__
  _\/\\\_______\/\\\______\/\\\_____\/\\\______\//\\\_
   _\/\\\\\\\\\\\\\/_______\/\\\_____\/\\\_______\/\\\_
    _\/\\\/////////_________\/\\\_____\/\\\_______\/\\\_
     _\/\\\__________________\/\\\_____\/\\\_______\/\\\_
      _\/\\\___________/\\\___\/\\\_____\/\\\_______/\\\__
       _\/\\\__________\//\\\\\\\\\______\/\\\\\\\\\\\\/___
        _\///____________\/////////_______\////////////_____

-> Name:  main.h
-> Brief: Header file for main.cpp
-> Date: May 15, 2017   (Created)
-> Author: Paul Duchesne (B00332119)
-> Contact: pl332718@dal.ca
*/

#ifndef MAIN_H
#define MAIN_H

std::string fft();
std::string fnt();

#endif
```

```
/*
__/\\\\\\\\\\\\\_____/\\\\\\\\\\\__/\\\\\\\\\\\\____
 _\/\\\/////////\\\__\/////\\\///__\/\\\////////\\\__
  _\/\\\_______\/\\\______\/\\\_____\/\\\______\//\\\_
   _\/\\\\\\\\\\\\\/_______\/\\\_____\/\\\_______\/\\\_
    _\/\\\/////////_________\/\\\_____\/\\\_______\/\\\_
     _\/\\\__________________\/\\\_____\/\\\_______\/\\\_
      _\/\\\___________/\\\___\/\\\_____\/\\\_______/\\\__
       _\/\\\__________\//\\\\\\\\\______\/\\\\\\\\\\\\/___
        _\///____________\/////////_______\////////////_____

-> Name:  library.h
-> Brief: library file for enumeration declarations and other constants
-> Date: May 15, 2017   (Created)
-> Author: Paul Duchesne (B00332119)
-> Contact: pl332718@dal.ca
*/

#ifndef LIBRARY_H
#define LIBRARY_H

// Used for lower and upper bounds during checks
#define MINBYTE -128
#define MAXBYTE  256
#define MINWORD -32768
#define MAXWORD  65536

// Variables used by fft and fnt
extern std::string current_record;
extern std::string current_token;
extern int err_cnt;
extern int line_num;

// Global fine types for input and output across all files
extern std::ifstream fin;
extern std::ofstream outfile;
extern std::ofstream srec_file;

enum SEARCHTYPE {
        I,  // INSTRUCTION
        D   //  DIRECTIVE
};

enum SYMTBLTYPE {
    REG,
    KNOWN,
    UNKNOWN
};

enum INST_TYPE {
    NONE,
    SINGLE,
    DOUBLE,
    JUMP
};

enum BYTE_WORD {
    WORD,
    BYTE,
    OFFSET
};

enum STATE {
    CHK_FIRST_TOKEN,
    CHK_NEXT_TOKEN,
    INST,
    DIRECT,
    CHK_SRC_OP,
    CHK_DST_OP,
    CHK_JMP_OP
};

enum ADDR_MODE {
    REG_DIRECT,
    INDEXED,
    RELATIVE,
    ABSOLUTE,
    INDIRECT,
    INDIRECT_AI,
    IMMEDIATE,
    WRONG
};

```

```
std::string fft(std::istream& fin);
std::string fnt();

#endif
```

```
/*
__/\\\\\\\\\\\\\_____/\\\\\\\\\\\__/\\\\\\\\\\\\____
 _\/\\\/////////\\\__\/////\\\///__\/\\\////////\\\__
  _\/\\\_______\/\\\______\/\\\_____\/\\\______\//\\\_
   _\/\\\\\\\\\\\\\/_______\/\\\_____\/\\\_______\/\\\_
    _\/\\\/////////_________\/\\\_____\/\\\_______\/\\\_
     _\/\\\__________________\/\\\_____\/\\\_______\/\\\_
      _\/\\\___________/\\\___\/\\\_____\/\\\_______/\\\__
       _\/\\\__________\//\\\\\\\\\______\/\\\\\\\\\\\\/___
        _\///____________\/////////_______\////////////_____

-> Name:  symtbl.h
-> Brief: Header file for the symtbl table
-> Date: May 15, 2017   (Created)
-> Author: Paul Duchesne (B00332119)
-> Contact: pl332718@dal.ca
*/

#ifndef SYMTBL_H
#define SYMTBL_H

#include "library.h"

struct symtbl_entry {
        std::string label;  // The name given by the user
        int value;          // Value stored in the symbol
        SYMTBLTYPE type;    // REG, KNOWN, or UNKNOWN
        int line;           // Line number label found on (Updated if an UNKNOWN is filled in)
        symtbl_entry *next; // Pointer to next entry on list
};


void init_symtbl();

void add_symbol(std::string label, int value, SYMTBLTYPE type);

symtbl_entry* get_symbol(std::string lbl);

void output_symtbl();

bool valid_symbol(std::string token);

void symtbl_unknown_check();

#endif
```

```
/*
__/\\\\\\\\\\\\\_____/\\\\\\\\\\\__/\\\\\\\\\\\\____
 _\/\\\/////////\\\__\/////\\\///__\/\\\////////\\\__
  _\/\\\_______\/\\\______\/\\\_____\/\\\______\//\\\_
   _\/\\\\\\\\\\\\\/_______\/\\\_____\/\\\_______\/\\\_
    _\/\\\/////////_________\/\\\_____\/\\\_______\/\\\_
     _\/\\\__________________\/\\\_____\/\\\_______\/\\\_
      _\/\\\___________/\\\___\/\\\_____\/\\\_______/\\\__
       _\/\\\__________\//\\\\\\\\\______\/\\\\\\\\\\\\/___
        _\///____________\/////////_______\////////////_____

-> Name:  emitter.h
-> Brief: Implementation file for emitter.ch
-> Date: May 24, 2017   (Created)
-> Author: Paul Duchesne (B00332119)
-> Contact: pl332718@dal.ca
*/

#ifndef EMITTER_H
#define EMITTER_H

struct single_overlay {
        union {
                struct {
                        unsigned short reg:4;
                        unsigned short as:2;
                        unsigned short bw:1;
                        unsigned short opcode:9;
                };
                unsigned short us_single;
        };
};

struct double_overlay {
        union {
                struct {
                        unsigned short dst:4;
                        unsigned short as:2;
                        unsigned short bw:1;
                        unsigned short ad:1;
                        unsigned short src:4;
                        unsigned short opcode:4;
                };
                unsigned short us_double;
        };
};

struct jump_overlay {
        union {
                struct {
                        unsigned short offset:10;
                        unsigned short opcode:6;
                };
                unsigned short us_jump;
        };
};

void emit(std::string inst, std::string operand, INST_TYPE type, int& LC);

#endif
```

```
/*
__/\\\\\\\\\\\\\_____/\\\\\\\\\\\__/\\\\\\\\\\\\____
 _\/\\\/////////\\\__\/////\\\///__\/\\\////////\\\__
  _\/\\\_______\/\\\______\/\\\_____\/\\\______\//\\\_
   _\/\\\\\\\\\\\\\/_______\/\\\_____\/\\\_______\/\\\_
    _\/\\\/////////_________\/\\\_____\/\\\_______\/\\\_
     _\/\\\__________________\/\\\_____\/\\\_______\/\\\_
      _\/\\\___________/\\\___\/\\\_____\/\\\_______/\\\__
       _\/\\\__________\//\\\\\\\\\______\/\\\\\\\\\\\\/___
        _\///____________\/////////_______\////////////_____

-> Name: inst_dir.h
-> Brief: Header file for the inst_dir file
-> Date: May 15, 2017   (Created)
-> Author: Paul Duchesne (B00332119)
-> Contact: pl332718@dal.ca
*/

#ifndef INST_DIR_H
#define INST_DIR_H

#include "library.h"

struct inst_dir {
        std::string mnemonic;
        INST_TYPE type;
        int opcode;
        BYTE_WORD b_w;
};

inst_dir* get_inst_dir(std::string input, SEARCHTYPE stype);

#endif
```

```
/*
__/\\\\\\\\\\\\\_____/\\\\\\\\\\\__/\\\\\\\\\\\\____
 _\/\\\/////////\\\__\/////\\\///__\/\\\////////\\\__
  _\/\\\_______\/\\\______\/\\\_____\/\\\______\//\\\_
   _\/\\\\\\\\\\\\\/_______\/\\\_____\/\\\_______\/\\\_
    _\/\\\/////////_________\/\\\_____\/\\\_______\/\\\_
     _\/\\\__________________\/\\\_____\/\\\_______\/\\\_
      _\/\\\___________/\\\___\/\\\_____\/\\\_______/\\\__
       _\/\\\__________\//\\\\\\\\\______\/\\\\\\\\\\\\/___
        _\///_____________\/////////_______\////////////_____

-> Name:  second_pass.h
-> Brief: Header file for second_pass.h
-> Date: May 24, 2017   (Created)
-> Author: Paul Duchesne (B00332119)
-> Contact: pl332718@dal.ca
*/

#ifndef SECOND_PASS_H
#define SECOND_PASS_H

void second_pass();

void error_detected_no_cnt(std::string error_msg);

#endif
```