```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <arpa/inet.h>
#include <assert.h>
#include "instruction.h"


char all_labels [100][100];
int label_locations[100];

//  getting branch labels
char * get_br(char* line){
    static char branch [50] = "";
    int track = 0;
    for (int i = 0; i < strlen(line); i++){
        if (line[i] == 'b'){
            for (int j = i; j < strlen(line); j++){
                if (line[j + 1] == 'L' || line[j + 1] == 's'){
                    return branch;
                }
                branch[track] = line[j];
                track++;
            }
        }
    }
}
//  all branch handling
uint16_t br(char* line, int loc, int num_instruct){
    if (strcmp(get_br(line), "brn") == 0){
        return emit_br(true, false, false, (loc - num_instruct) - 1);
    } else if (strcmp(get_br(line), "brp") == 0){
        return emit_br(false, false, true, (loc - num_instruct) - 1);
    } else if (strcmp(get_br(line), "brz") == 0){
        return emit_br(false, true, false, (loc - num_instruct) - 1);
    } else if (strcmp(get_br(line), "brzp") == 0){
        return emit_br(false, true, true, (loc - num_instruct) - 1);
    } else if (strcmp(get_br(line), "brnp") == 0){
        return emit_br(true, false, true, (loc - num_instruct) - 1);
    } else if (strcmp(get_br(line), "brnz") == 0){
        return emit_br(true, true, false, (loc - num_instruct) - 1);
    } else if (strcmp(get_br(line), "brnzp") == 0){
        return emit_br(true, true, true, (loc - num_instruct) - 1);
    } else {
        return emit_br(false, false, false, (loc - num_instruct) - 1);
    }
}
```

```
//  gets labels locations
int get_labeloc(char* label){
    int x = 0;
    for (int i = 0; i < 50; i++){
        const char* lab = all_labels[i];
        if (lab != NULL && strcmp(label, lab) == 0){
            x = i;
            return label_locations[x];
            break;
        }
    }
    exit(2);
    return -1;
}

char * get_label(char* line){
    static char label [40] = "";
    int label_track = 0;
    int start = 0;
    if (strstr((const char*) line, (const char*) "ld")){
        for (int i = 0; i < strlen(line); i++){
            if (line[i] == ','){
                start = i + 2;
                break;
            }
        }
    } else if (strstr((const char*) line, (const char*) "brz")){
        for (int i = 0; i < strlen(line); i++){
            if (line[i] == 'z'){
                start = i + 2;
                break;
            }
        }
    } else if (strstr((const char*) line, (const char*) "jsr")){
        for (int i = 0; i < strlen(line); i++){
            if (line[i] == 'r'){
                start = i + 2;
                break;
            }
        }
    } else if (strstr((const char*) line, (const char*) "LABEL")){
        return "LABEL";
    } else if (strstr((const char*) line, (const char*) "morelabel")){
        return "morelabel";
    }
    while (start < strlen(line) - 2){
        label[label_track] = line[start];
        start++;
        label_track++;
```

```
    }
    if (strstr((const char*) line, (const char*) "LABEL")) {
        return "LABEL";
    } else {
        return label;
    }
}

uint16_t ldi(char* line){
    char value [30] = "";
    for (int i = 0; i < strlen(line); i++){
        if (line[i] == '%'){
            value[0] = line[i + 2];
        }
    }
    uint16_t z = emit_ldi(atoi((const char*) &value), 0);
}

uint16_t ld(char* line){
    char value [30] = "";
    for (int i = 0; i < strlen(line); i++){
        if (line[i] == '%'){
            value[0] = line[i + 2];
        }
    }
    uint16_t z = emit_ld(atoi((const char*) &value), 0);
    return z;
}

//  values case
uint16_t val(char* line){
    char word [30] = "";
    int track = 0;
    for (int i = 0; i < strlen(line); i++){
        if (line[i] == '$'){
            for (int j = i + 1; j < strlen(line); j++){
                if (line[j] == '#'){
                    break;
                }
                word[track] = line[j];
                track++;
            }
        }
    }
    uint16_t z = emit_value(atoi((const char*) &word));
    return z;
}

uint16_t ldr(char* line){
```

```
    char word [30] = "";
    char DR [30] = "";
    char SR1 [30] = "";
    char SR2 [30] = "";
    int tracker = 0;
    if (strstr((const char*) line, (const char*) "$")){
        for (int i = 0; i < strlen(line); i++){  // loop through line
            if (line[i] == '%'){  // check for register
                if (tracker == 0){   // determining register
                    DR[0] = line[i + 2];
                    tracker++;
                } else {  // determining register
                    SR1[0] = line [i + 2];
                }
            } else if (line[i] == '$'){  // is number
                int track = 0;
                for (int j = i + 1; j < strlen(line); j++){
                    if (line[j] == '#'){
                        break;
                    }
                    word[track] = line[j];  // could cause problems
                    track++;
                }
            }
        }
    }
        uint16_t z = emit_ldr(atoi((const char*) &DR),
        atoi((const char*) &SR1), atoi((const char*) &word));
    return z;
}
//  not case
uint16_t not_test(char* line){
    char word [30] = "";
    char DR [30] = "";
    char SR1 [30] = "";
    char SR2 [30] = "";
    int tracker = 0;
    for (int i = 0; i < strlen(line); i++){  // loop through line
        if (line[i] == '%'){  // check for register
            if (tracker == 0){   // determining register
                DR[0] = line[i + 2];
                tracker++;
            } else if (tracker == 1){  // determining register
                SR1[0] = line [i + 2];
                tracker++;
            }
        }
    }
    uint16_t z = emit_not(atoi((const char*) &DR),
```

```
    atoi((const char*) &SR1));
    return z;
}

uint16_t str(char* line){
    char word [30] = "";
    char DR [30] = "";
    char SR1 [30] = "";
    char SR2 [30] = "";
    int tracker = 0;
    if (strstr((const char*) line, (const char*) "$")){
        for (int i = 0; i < strlen(line); i++){  // loop through line
            if (line[i] == '%'){  // check for register
                if (tracker == 0){   // determining register
                    DR[0] = line[i + 2];
                    tracker++;
                } else {  // determining register
                    SR1[0] = line [i + 2];
                }
            } else if (line[i] == '$'){  // is number
                int track = 0;
                for (int j = i + 1; j < strlen(line); j++){
                    if (line[j] == '#'){
                        break;
                    }
                    word[track] = line[j];  // could cause problems
                    track++;
                }
            }
        }
    }
    uint16_t z = emit_str(atoi((const char*) &DR),
    atoi((const char*) &SR1), atoi((const char*) &word));
    return z;
}
uint16_t check(char* line){
    int num = 0;
    for (int i = 0; i < strlen(line); i++){
        if (line[i] == '%'){
            num++;
        }
    }
    if (num != 3){
        return false;
    } else {
        return true;
    }
}
```

```
uint16_t add(char* line) {
    char word [30] = "";
    char DR [30] = "";
    char SR1 [30] = "";
    char SR2 [30] = "";
    int tracker = 0;
    if (strstr((const char*) line, (const char*) "$")){
        for (int i = 0; i < strlen(line); i++){  // loop through line
            if (line[i] == '%'){  // check for register
                if (tracker == 0){   // determining register
                    DR[0] = line[i + 2];
                    tracker++;
                } else {  // determining register
                    SR1[0] = line [i + 2];
                }
            } else if (line[i] == '$'){  // is number
                int track = 0;
                for (int j = i + 1; j < strlen(line); j++){
                    if (line[j] == '#'){
                        break;
                    }
                    word[track] = line[j];  // could cause problems
                    track++;
                }
            }
        }
        uint16_t z = emit_add_imm(atoi((const char*) &DR),
        atoi((const char*) &SR1), atoi((const char*) &word));
        return z;
    } else {
        if (check(line) == false){
            exit(2);
        }
        for (int i = 0; i < strlen(line); i++){  // loop through line
            if (line[i] == '%'){  // check for register
                if (tracker == 0){   // determining register
                    DR[0] = line[i + 2];
                    tracker++;
                } else if (tracker == 1){  // determining register
                    SR1[0] = line [i + 2];
                    tracker++;
                } else {
                    SR2[0] = line [i + 2];
                    tracker++;
                }
            }
        }
        uint16_t z = emit_add_reg(atoi((const char*) &DR),
        atoi((const char*) &SR1), atoi((const char*) &SR2));
```

```
        return z;
    }
}

void usage() {
    fprintf(stderr, "Usage: ./xas file");
    exit(1);
}

int main(int argc, char** argv) {
    if (argc != 2) {
        usage();
    }


    int label_curloc = 0;
    uint16_t instructions [100];
    int instruct_loc = 0;
    char word [50];
    char line[200];
    int line_num = 0;
    int curinstruct = 0;


    FILE* fileName = fopen(argv[1], "r");
    while (fgets(line, 100, fileName) != NULL) {  // traverse file
        if (strstr((const char*) line, (const char*) "add")){
            instructions[instruct_loc] = add(line);
            instruct_loc++;
        } else if (strstr((const char*) line, (const char*) "putc")){
            uint16_t temp = emit_trap(0x21);
            instructions[instruct_loc] = temp;
            instruct_loc++;
        } else if (strstr((const char*) line, (const char*) "val")){
            instructions[instruct_loc] = val(line);
            instruct_loc++;
        } else if (strstr((const char*) line, (const char*) "ldi")){
            instructions[instruct_loc] = ldi(line);
            instruct_loc++;
        } else if (strstr((const char*) line, (const char*) "ldr")){
            instructions[instruct_loc] = ldr(line);
            instruct_loc++;
        } else if (strstr((const char*) line, (const char*) "lea")){
            char value [30] = "p";
            for (int i = 0; i < strlen(line); i++){
                if (line[i] == '%'){
                    value[0] = line[i + 2];
                    break;
                }
```

```
    }
    if (value[0] == 'p'){
        exit(2);
    }
    instructions[instruct_loc] = emit_lea(atoi((const char*)
    &value[0]), 0);
    instruct_loc++;
} else if (strstr((const char*) line, (const char*) "not")){
    instructions[instruct_loc] = not_test(line);
    instruct_loc++;
} else if (strstr((const char*) line, (const char*) "ld")){
    instructions[instruct_loc] = ld(line);
    instruct_loc++;
} else if (strstr((const char*) line, (const char*) "br")){
    instruct_loc++;
} else if (strstr((const char*) line, (const char*) "halt")){
    uint16_t temp = emit_trap(0x25);
    instructions[instruct_loc] = temp;
    instruct_loc++;
} else if (strstr((const char*) line, (const char*) "jsrr")){
    char value [30] = "";
    for (int i = 0; i < strlen(line); i++){
        if (line[i] == '%'){
            value[0] = line[i + 2];
        }
    }
    uint16_t temp = emit_jsrr(atoi((const char*) &value));
    instructions[instruct_loc] = temp;
    instruct_loc++;
} else if (strstr((const char*) line, (const char*) "js")){
    uint16_t temp = emit_jsr(0);
    instructions[instruct_loc] = temp;
    instruct_loc++;
} else if (strstr((const char*) line, (const char*) "jmp")){
    char value [30] = "";
    for (int i = 0; i < strlen(line); i++){
        if (line[i] == '%'){
            value[0] = line[i + 2];
        }
    }
    uint16_t temp = emit_jmp(atoi((const char*) &value));
    instructions[instruct_loc] = temp;
    instruct_loc++;
} else if (strstr((const char*) line, (const char*) "sti")){
    char value [30] = "";
    for (int i = 0; i < strlen(line); i++){
        if (line[i] == '%'){
            value[0] = line[i + 2];
        }
```

```
        }
        uint16_t temp = emit_sti(atoi((const char*) &value), 0);
        instructions[instruct_loc] = temp;
        instruct_loc++;
    } else if (strstr((const char*) line, (const char*) "str")){
        instructions[instruct_loc] = str(line);
        instruct_loc++;
    }  else if (strstr((const char*) line, (const char*) ":")){
        char label[100] = "";
        for (int i = 0; i < strlen(line); i++){
            if (line[i] == ':'){
                break;
            }
            label[i] = line[i];
        }
        strcpy(all_labels[label_curloc], label);
        label_locations[label_curloc] = instruct_loc;
        label_curloc++;
    } else if (strstr((const char*) line, (const char*) "st")
    && line[2] != 'a' && line[2] != 'o'){
        char value [30] = "";
        for (int i = 0; i < strlen(line); i++){
            if (line[i] == '%'){
                value[0] = line[i + 2];
            }
        }
        uint16_t temp = emit_st(atoi((const char*) &value), 0);
        instructions[instruct_loc] = temp;
        instruct_loc++;
    } else if (strstr((const char*) line, (const char*) "getc")){
        uint16_t temp = emit_trap(0x20);
        instructions[instruct_loc] = temp;
        instruct_loc++;
    } else if (strstr((const char*) line, (const char*) "putsp")){
        uint16_t temp = emit_trap(0x24);
        instructions[instruct_loc] = temp;
        instruct_loc++;
    } else if (strstr((const char*) line, (const char*) "puts")){
        uint16_t temp = emit_trap(0x22);
        instructions[instruct_loc] = temp;
        instruct_loc++;
    } else if (strstr((const char*) line, (const char*) "enter")){
        uint16_t temp = emit_trap(0x23);
        instructions[instruct_loc] = temp;
        instruct_loc++;
    }
    line_num++;
}
```

```
fclose(fileName);
FILE* file2 = fopen(argv[1], "r");
int line_track = 0;
int num_instructions = 0;
while (fgets(line, 100, file2) != NULL) {  // traverse file second time
    if (strstr((const char*) line, (const char*) "ldi")){
        int loc = get_labeloc(get_label(line));
        instructions[num_instructions] = emit_ldi(getbits(instructions
        [num_instructions], 9, 3), (loc - num_instructions)-1);
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "ldr")){
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "ld")){
        int loc = get_labeloc(get_label(line));
        instructions[num_instructions] = emit_ld(getbits(instructions
        [num_instructions], 9, 3), (loc - num_instructions)-1);
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "add")){
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "val")){
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "br")){
        int loc = get_labeloc(get_label(line));
        uint16_t tempor = br(line, loc, num_instructions);
        instructions[num_instructions] = tempor;
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "jsrr")){
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "jsr")){
        int loc = get_labeloc(get_label(line));
        instructions[num_instructions] =
        emit_jsr((loc - num_instructions) - 1);
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "putc")){
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "puts")){
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "getc")){
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "halt")){
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "jmp")){
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "lea")){
        int loc = get_labeloc(get_label(line));
        instructions[num_instructions] = emit_lea(getbits(instructions
        [num_instructions], 9, 3), (loc - num_instructions)-1);
        num_instructions++;
    } else if (strstr((const char*) line, (const char*) "not")){
```

```
            num_instructions++;
        } else if (strstr((const char*) line, (const char*) "sti")){
            int loc = get_labeloc(get_label(line));
            instructions[num_instructions] = emit_sti(getbits(instructions
            [num_instructions], 9, 3) , (loc - num_instructions)-1);
            num_instructions++;
        } else if (strstr((const char*) line, (const char*) "str")){
            num_instructions++;
        } else if (strstr((const char*) line, (const char*) "st")
        && line[2] != 'a' && line[2] != 'o'){
            int loc = get_labeloc(get_label(line));
            instructions[num_instructions] = emit_st(getbits(instructions
            [num_instructions], 9, 3)  , (loc - num_instructions)-1);
            num_instructions++;
        } else if (strstr((const char*) line, (const char*) "enter")){
            num_instructions++;
        } else if (strstr((const char*) line, (const char*) "putsp")){
            num_instructions++;
        }
        line_track++;
    }  //  write to file
    FILE* a = fopen("a.obj", "wb");
    uint16_t origin = 0x3000;
    origin = htons(origin);
    fwrite(&origin, 1 , sizeof(origin), a);
    for (int i = 0; i < instruct_loc; i++){
        printf("\n");
        print_instruction(instructions[i]);
        uint16_t temp = htons(instructions[i]);
        fwrite(&temp, 1, sizeof(temp), a);
    }
    fclose(a);
    exit(0);
    return 0;
}
```