#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 10

// Structures

typedef struct {

char symbol[10];

int address;

} Symbol;

typedef struct {

char literal[10];

int address;

} Literal;

typedef struct {

char opcode[10];

char type[3]; // IS, AD, DL

int opcode\_num;

} MOTEntry;

Symbol symtab[MAX];

Literal littab[MAX];

int pooltab[MAX];

int sym\_count = 0, lit\_count = 0, pool\_count = 0;

// Sample MOT

MOTEntry mot[] = {

{"START", "AD", 1},

{"END", "AD", 2},

{"LTORG", "AD", 3},

{"DS", "DL", 1},

{"DC", "DL", 2},

{"MOVER", "IS", 4},

{"MOVEM", "IS", 5},

};

int LC = 0;

// Get MOT entry

MOTEntry\* get\_mot(char \*opcode) {

for (int i = 0; i < sizeof(mot)/sizeof(mot[0]); i++) {

if (strcmp(mot[i].opcode, opcode) == 0)

return &mot[i];

}

return NULL;

}

// Add to symbol table

void add\_symbol(char \*label) {

strcpy(symtab[sym\_count].symbol, label);

symtab[sym\_count++].address = LC;

}

// Add literal

int add\_literal(char \*lit) {

for (int i = 0; i < lit\_count; i++) {

if (strcmp(littab[i].literal, lit) == 0)

return i;

}

strcpy(littab[lit\_count].literal, lit);

littab[lit\_count].address = -1; // To be resolved

return lit\_count++;

}

// Allocate literals (on LTORG or END)

void allocate\_literals() {

for (int i = 0; i < lit\_count; i++) {

if (littab[i].address == -1) {

littab[i].address = LC++;

}

}

pooltab[pool\_count++] = lit\_count;

}

void print\_tables() {

printf("\nSYMBOL TABLE:\n");

for (int i = 0; i < sym\_count; i++)

printf("%s -> %d\n", symtab[i].symbol, symtab[i].address);

printf("\nLITERAL TABLE:\n");

for (int i = 0; i < lit\_count; i++)

printf("%s -> %d\n", littab[i].literal, littab[i].address);

printf("\nPOOL TABLE:\n");

for (int i = 0; i < pool\_count; i++)

printf("#%d\n", pooltab[i]);

}

// Simulate 2-pass assembly

int main() {

char \*code[] = {

"START 200",

"MOVER AREG, ='5'",

"MOVEM AREG, X",

"L1 MOVER BREG, ='3'",

"LTORG",

"X DS 1",

"END"

};

int n = sizeof(code)/sizeof(code[0]);

printf("INTERMEDIATE CODE:\n");

for (int i = 0; i < n; i++) {

char label[10] = "", opcode[10] = "", op1[10] = "", op2[10] = "";

sscanf(code[i], "%s %s %[^,], %s", label, opcode, op1, op2);

if (strcmp(label, "START") == 0) {

sscanf(code[i], "%s %d", opcode, &LC);

MOTEntry\* m = get\_mot(opcode);

printf("(%s, %02d) C, %d\n", m->type, m->opcode\_num, LC);

continue;

}

if (strchr(code[i], ':') != NULL || strlen(op2) == 0) { // label present

sscanf(code[i], "%s %s %s", label, opcode, op1);

if (label[1] != 0) // not empty

add\_symbol(label);

}

if (strcmp(opcode, "LTORG") == 0 || strcmp(opcode, "END") == 0) {

allocate\_literals();

MOTEntry\* m = get\_mot(opcode);

printf("(%s, %02d)\n", m->type, m->opcode\_num);

continue;

}

MOTEntry\* m = get\_mot(opcode);

if (!m) continue;

printf("%d) (%s, %02d) ", LC, m->type, m->opcode\_num);

if (strchr(op1, '\'') != NULL) { // literal

int idx = add\_literal(op1);

printf("L, %d\n", idx);

} else if (strcmp(op1, "AREG") == 0 || strcmp(op1, "BREG") == 0) {

printf("1 ");

}

if (strlen(op2) && op2[0] != '=') {

printf("S, %s\n", op2);

} else if (strlen(op2)) {

int idx = add\_literal(op2);

printf("L, %d\n", idx);

}

LC++;

}

print\_tables();

return 0;

}