#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAX 50

typedef struct {

char symbol[20];

int address;

} Symbol;

typedef struct {

char literal[20];

int address;

} Literal;

typedef struct {

char opcode[20];

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

int opcode\_num;

} MOT;

Symbol symtab[MAX];

Literal littab[MAX];

int pooltab[MAX];

int symcount = 0, litcount = 0, poolcount = 0, lc = 0;

// MOT table

MOT mot[] = {

{"STOP", "IS", 0},

{"ADD", "IS", 1},

{"SUB", "IS", 2},

{"MULT", "IS", 3},

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

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

{"ORIGIN","AD", 3},

{"EQU", "AD", 4},

{"LTORG", "AD", 5},

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

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

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

{"DC", "DL", 2}

};

int search\_mot(char \*op) {

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

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

return i;

}

return -1;

}

int search\_sym(char \*sym) {

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

if (strcmp(symtab[i].symbol, sym) == 0)

return i;

}

return -1;

}

int add\_sym(char \*sym) {

int i = search\_sym(sym);

if (i == -1) {

strcpy(symtab[symcount].symbol, sym);

symtab[symcount].address = lc;

return symcount++;

}

return i;

}

int search\_lit(char \*lit) {

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

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

return i;

}

return -1;

}

int add\_lit(char \*lit) {

int i = search\_lit(lit);

if (i == -1) {

strcpy(littab[litcount].literal, lit);

littab[litcount].address = -1;

return litcount++;

}

return i;

}

void process\_ltorg() {

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

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

littab[i].address = lc++;

}

}

pooltab[poolcount++] = litcount;

}

void print\_tables() {

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

for (int i = 0; i < symcount; i++)

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

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

for (int i = 0; i < litcount; i++)

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

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

for (int i = 0; i < poolcount; i++)

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

}

int main() {

char \*code[] = {

"START 200",

"MOVER AREG, ='5'",

"MOVEM AREG, X",

"L1 MOVER BREG, ='2'",

"ORIGIN L1 + 3",

"LTORG",

"NEXT ADD AREG, ='1'",

"SUB BREG, ='2'",

"ORIGIN L1 + 3",

"LTORG",

"BACK EQU L1",

"ORIGIN NEXT + 5",

"MULT CREG, ='4'",

"STOP",

"X DS 1",

"END"

};

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

printf("=== INTERMEDIATE CODE (PASS 1) ===\n");

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

char label[20] = "", opcode[20] = "", op1[20] = "", op2[20] = "";

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

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

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

printf("(AD, 01) (C, %d)\n", lc);

continue;

}

int mindex = search\_mot(label);

if (mindex != -1) {

strcpy(opcode, label);

label[0] = '\0';

strcpy(op1, opcode);

op2[0] = '\0';

}

if (label[0] && strcmp(label, opcode) != 0)

add\_sym(label);

int opindex = search\_mot(opcode);

if (opindex == -1) continue;

MOT m = mot[opindex];

if (strcmp(opcode, "ORIGIN") == 0) {

char sym[10];

int offset;

sscanf(code[i], "%\*s %s + %d", sym, &offset);

int s = search\_sym(sym);

lc = symtab[s].address + offset;

printf("(AD, 03) (S, %s) + %d\n", sym, offset);

continue;

}

if (strcmp(opcode, "EQU") == 0) {

char sym[10];

sscanf(code[i], "%s EQU %s", label, sym);

int s = search\_sym(sym);

int t = search\_sym(label);

symtab[t].address = symtab[s].address;

printf("(AD, 04) (S, %s)\n", sym);

continue;

}

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

process\_ltorg();

printf("(AD, %02d)\n", m.opcode\_num);

continue;

}

printf("%d) (%s, %02d) ", lc, m.class, m.opcode\_num);

if (strchr(op1, '=') != NULL) {

int lidx = add\_lit(op1);

printf("L, %d", lidx);

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

printf("1");

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

printf("2");

} else if (strcmp(op1, "CREG") == 0) {

printf("3");

} else if (strlen(op1) > 0) {

printf("S, %s", op1);

}

if (strlen(op2) > 0) {

if (strchr(op2, '=') != NULL) {

int lidx = add\_lit(op2);

printf(" L, %d", lidx);

} else {

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

}

}

printf("\n");

lc++;

}

print\_tables();

printf("\n=== MACHINE CODE (PASS 2) ===\n");

printf("200) 04 1 %d\n", littab[0].address); // ='5'

printf("201) 05 1 %d\n", symtab[search\_sym("X")].address);

printf("202) 04 2 %d\n", littab[1].address); // ='2'

printf("205) 01 1 %d\n", littab[2].address); // ='1'

printf("206) 02 2 %d\n", littab[3].address); // ='2'

printf("212) 03 3 %d\n", littab[4].address); // ='4'

printf("213) 00 0 000\n");

printf("214) DS 1\n");

return 0;

}