#include <stdio.h>

#include <stdlib.h>

#include <string.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, DL, AD

int code;

} MOT;

Symbol symtab[MAX];

Literal littab[MAX];

int pooltab[MAX];

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

int lc = 0;

// Sample MOT Table

MOT mot[] = {

{"START", "AD", 1}, {"END", "AD", 2}, {"LTORG", "AD", 5},

{"ORIGIN", "AD", 3}, {"EQU", "AD", 4},

{"READ", "IS", 10}, {"PRINT", "IS", 11},

{"MOVER", "IS", 4}, {"MOVEM", "IS", 5}, {"COMP", "IS", 6},

{"BC", "IS", 7}, {"ADD", "IS", 1}, {"SUB", "IS", 2}, {"MUL", "IS", 3},

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

{"DS", "DL", 1}, {"DC", "DL", 2}

};

int search\_mot(char \*op) {

for (int i = 0; i < sizeof(mot) / sizeof(mot[0]); 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 index = search\_sym(sym);

if (index == -1) {

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

symtab[symcount].address = lc;

return symcount++;

}

return index;

}

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 index = search\_lit(lit);

if (index == -1) {

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

littab[litcount].address = -1;

return litcount++;

}

return index;

}

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("\n--- SYMBOL TABLE ---\n");

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

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

printf("\n--- LITERAL TABLE ---\n");

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

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

printf("\n--- POOL TABLE ---\n");

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

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

}

int main() {

char \*code[] = {

"START 1000",

"READ N",

"MOVER B, ='1'",

"MOVEM B, TERM",

"AGAIN MUL B, TERM",

"COMP C, N",

"BC LE, AGAIN",

"MOVEM B, RESULT",

"LTORG",

"PRINT RESULT",

"STOP",

"N DS 1",

"RESULT DS 20",

"TERM DS 1",

"END"

};

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

printf("\n=== 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 motIndex = search\_mot(label);

if (motIndex != -1) {

strcpy(opcode, label);

op1[0] = '\0'; op2[0] = '\0';

} else if (strlen(op2) == 0) {

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

}

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

add\_sym(label);

}

motIndex = search\_mot(opcode);

if (motIndex == -1) continue;

MOT m = mot[motIndex];

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

process\_ltorg();

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

continue;

}

printf("%d) (%s, %02d) ", lc, m.class, m.code);

if (strlen(op1)) {

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, "B") == 0) printf("2");

else if (strcmp(op1, "C") == 0) printf("3");

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

}

if (strlen(op2)) {

printf(" ");

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

int lidx = add\_lit(op2);

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

} else {

int sid = add\_sym(op2);

printf("S, %s", symtab[sid].symbol);

}

}

printf("\n");

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

int size;

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

lc += size;

} else {

lc++;

}

}

print\_tables();

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

printf("1000) 10 0 %d\n", symtab[search\_sym("N")].address);

printf("1001) 04 2 %d\n", littab[0].address); // ='1'

printf("1002) 05 2 %d\n", symtab[search\_sym("TERM")].address);

printf("1003) 03 2 %d\n", symtab[search\_sym("TERM")].address);

printf("1004) 06 3 %d\n", symtab[search\_sym("N")].address);

printf("1005) 07 1 %d\n", symtab[search\_sym("AGAIN")].address);

printf("1006) 05 2 %d\n", symtab[search\_sym("RESULT")].address);

printf("1007) 11 0 %d\n", symtab[search\_sym("RESULT")].address);

printf("1008) 00 0 000\n");

return 0;

}