```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char mnemonic[18][3][10] = {
    {"00", "STOP", "IS"},
    {"01", "ADD", "IS"},
    {"02", "SUB", "IS"},
    {"03", "MULT", "IS"},
    {"04", "MOVER", "IS"},
    {"05", "MOVEM", "IS"},
    {"06", "COMP", "IS"},
    {"07", "BC", "IS"},
    {"08", "DIV", "IS"},
    {"09", "READ", "IS"},
    {"10", "PRINT", "IS"},
    {"11", "DS", "DL"},
    {"12", "DC", "DL"},
    {"13", "START", "AD"},
    {"14", "END", "AD"},
    {"15", "ORIGIN", "AD"},
    {"16", "EQU", "AD"},
    {"17", "LTORG", "AD"}};

char register1[4][10] = {"AREG", "BREG", "CREG", "DREG"};
char condition[6][10] = {"LT", "LE", "GT", "GE", "EQ", "ANY"};
char symbol_table[10][2][10] = {{"LOOP", "102"}, {"NEXT", "107"}, {"LAST", "113"},
{"A", "115"}, {"B", "116"}};
char lit_table[10][2][10] = {{"=5", "108"}, {"=1", "109"}, {"=1", "111"}, {"=2", "117"}};
int s1 = 10, l1 = 10, p1 = 10, l_cnt = 0, blank_cnt = 0, remain1 = 0;
int main() {
   FILE *fr, *fw;
   int start = 0, loc = 0, equ = 0, ltorg = 0, end = 0, dl = 0, is = 0;
   char *field, record[200], const1[10], left_op[20];
   int pool_ptr = 0, i = 0;

   fr = fopen("ass_ic.txt", "r");
   fw = fopen("icnew.txt", "w");

   while (!feof(fr)) {
         int fcnt = 0;
         int found = 0, index;
         loc++;
         if (loc != 1) {
         fprintf(fw, "%d%s", loc, "+");
      }
      fgets(record, 200, fr);
      field = strtok(record, " ");
      while (field != NULL) {
         fcnt++;
         if (fcnt == 2) {
            if (ltorg == 1 && strcmp(field, "#") == 0) {
               for (i = 0; i < 18; i++) {
                  if (strcmp(mnemonic[i][1], "DC") == 0) {
                     fprintf(fw, "%s%s%s\t", "(DL,", mnemonic[i][0], ")");
                  }
               }
            }
            if (end == 1 && strcmp(field, "#") == 0) {
               for (i = 0; i < 18; i++) {
```

```
            if (strcmp(mnemonic[i][1], "DC") == 0) {
                fprintf(fw, "%s%s%s\t", "(DL,", mnemonic[i][0], ")");
            }
        }
    }
    for (i = 0; i < 18; i++) {
        if (strcmp(mnemonic[i][1], field) == 0) {
            found = 1;
            index = i;
            break;
        }
    }
    if (found == 1) {
        char class1[10] = "", mnemonic1[10] = "", op_code[10] = "";
        strcpy(class1, mnemonic[index][2]);
        strcpy(mnemonic1, mnemonic[index][1]);
        strcpy(op_code, mnemonic[index][0]);
        if (strcmp(class1, "AD") == 0) {
            if (strcmp(mnemonic1, "START") == 0) {
                start = 1;
                fprintf(fw, "%s%s%s", "(AD,", op_code, ")");
            }
            if (strcmp(mnemonic1, "EQU") == 0) {
                equ = 1;
                fprintf(fw, "%s%s%s", "(AD,", op_code, ")");
                loc--;
            }
            if (strcmp(mnemonic1, "LTORG") == 0) {
                ltorg = 1;
                fprintf(fw, "%s%s%s", "(AD,", op_code, ")");
                loc--;
                pool_ptr++;
                break;
            }
            if (strcmp(mnemonic1, "END") == 0) {
                end = 1;
                fprintf(fw, "%s%s%s", "(AD,", op_code, ")");
                loc--;
                break;
            }
        } else if (strcmp(class1, "DL") == 0) {
            dl = 1;
            fprintf(fw, "%s%s%s\t", "( DL,", op_code, ")");
        } else if (strcmp(class1, "IS") == 0) {
            is = 1;
            fprintf(fw, "%s%s%s\t", "(IS,", op_code, ")");
        }
    }
}
if (fcnt == 3) {
    if (dl == 1 && equ != 1 && end != 1) {
        fprintf(fw, "%s%s%s\t", "(C,", field, ")");
    }
    if (is == 1) {
        for (i = 0; i < 4; i++) {
            if (strcmp(register1[i], field) == 0) {
                fprintf(fw, "%s%d%s\t", "(", i + 1, ")");
            }
        }
        for (i = 0; i < 6; i++) {
```

```
        if (strcmp(condition[i], field) == 0) {
            fprintf(fw, "%s%d%s\t", "(", i + 1, ")");
        }
    }
}
if (start == 1) {
    strcpy(const1, field);
    loc = atoi(const1);
    fprintf(fw, "%s%d%s\t", "(C,", loc, ")");
    loc = loc - 1;
    start = 0;
}
if (equ == 1) {
    for (i = 0; i < s1; i++) {
        if (strcmp(symbol_table[i][0], field) == 0) {
            fprintf(fw, "%s%d%s", "(S,", i + 1, ")");
    equ = 0;
    break;
        }
    }
}
if (ltorg == 1) {
    char *ptr, *s;
    ptr = strchr(field, '=');
    if (ptr) {
        s = strtok(field, "=");
        fprintf(fw, "%s%s%s \t\t    ", "(C,", s, ")");
    } else {
        ltorg = 0;
    }
```

```
        }
        if (end == 1) {
            char *ptr, *s;
            ptr = strchr(field, '=');
            if (ptr) {
                s = strtok(field, "=");
                fprintf(fw, "%s%s%s \t\t", "(C,", s, ")");
            } else {
                end = 0;
            }
        }
    }
    if (fcnt == 4) {
        char *ptr;
        ptr = strchr(field, '=');
        if (ptr) {
            int get_lit;
            for (i = 0; i < l1; i++) { // Iterate over lit_table directly
                if (strcmp(lit_table[i][0], field) == 0) {
                    get_lit = i;
                    fprintf(fw, "%s%d%s", "(L,", (get_lit + 1), ")");
                    break; // Once found, exit loop
                }
            }
        }
        else {
            int complete = 0;
            for (i = 0; i < s1; i++) {
                if (strcmp(symbol_table[i][0], field) == 0) {
                    fprintf(fw, "%s%d%s", "(S,", i + 1, ")");
```

```
                            complete = 1;
                            break;
                        }
                    }
                    if (complete == 0) {
                        // Handle undefined symbols here
                    }
                }
            }
            field = strtok(NULL, " ");
            if (fcnt != 1) {
                fprintf(fw, "\n");
            }
        }
    }
    fclose(fr);
    fclose(fw);
    return 0;
}
```