**SSN COLLEGE OF ENGINEERING, KALAVAKKAM**

**DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING**

**UCS1602 - Compiler Design**

Assignment-8 Implementation of code optimization techniques

Name:Guntumadugu Anil Kumar

Roll:195001035

sec:CSE-A

source code:

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#define NOL 50

#define SOL 50

int main(){

char ch, fname[25];

FILE \*fp;

char \*line = NULL;

size\_t len = 0;

ssize\_t read;

printf("Enter name of a file: ");

gets(fname);

fp = fopen(fname, "r");

if (fp == NULL){

perror("Error while opening the file.\n"); exit(-1);

}

printf("Input file contents: ");

char \*\*tac, \*\*rhs, \*\*lhs;

tac = malloc(NOL \* sizeof(char \*)); for (int i = 0; i < NOL; i++)

tac[i] = malloc((SOL + 1) \* sizeof(char));

int loc = 0;

while ((read = getline(&line, &len, fp)) != -1){ printf("%s", line);

if (read > 2){

strcpy(tac[loc++], line);

}

}

fclose(fp);

int \*leaders;

leaders = malloc(loc \* sizeof(int));

leaders[0] = 0;

int lnum = 0;

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

char \*gt = strstr(tac[i], "goto");

if (gt){

leaders[++lnum] = i;

leaders[++lnum] = i + 1;

}

}

char \*token;

rhs = malloc(loc \* sizeof(char \*)); for (int i = 0; i < loc; i++)

rhs[i] = malloc((SOL + 1) \* sizeof(char));

lhs = malloc(loc \* sizeof(char \*)); for (int i = 0; i < loc; i++)

lhs[i] = malloc((SOL + 1) \* sizeof(char));

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

token = strtok(tac[i], ":=");

if (token == NULL)

strcpy(lhs[i], "\n");

else

strcpy(lhs[i], token);

token = strtok(NULL, ":=");

if (token == NULL)

strcpy(rhs[i], "\n");

else

strcpy(rhs[i], token);

}

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

int len = strlen(rhs[i]);

if (len == 5 && strstr(rhs[i], "0") != NULL){ if (rhs[i][1] == '+'){

if (rhs[i][0] == '0'){

rhs[i][0] = rhs[i][2];

rhs[i][1] = ' ';

rhs[i][2] = ' ';

}

else if (rhs[i][2] == '0'){

rhs[i][1] = ' ';

rhs[i][2] = ' ';

}

}

else if (rhs[i][1] == '\*'){

if (rhs[i][0] == '0'){

char replace[] = "";

strncat(replace, "0", 1);

strcpy(rhs[i], replace);

}

else if (rhs[i][2] == '0'){

char replace[] = "";

strncat(replace, "0", 1);

strcpy(rhs[i], replace);

}

}

}

}

printf("\n ------------------------------------------ \nAlgebraic Identity\n ------------------------------------------ \n");

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

printf("%s := %s \n", lhs[i], rhs[i]);

}

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

int len = strlen(rhs[i]);

if (len == 5 && isdigit(rhs[i][0]) && isdigit(rhs[i][2])){ if (rhs[i][1] == '+'){

int x = rhs[i][0] - '0';

int y = rhs[i][2] - '0';

rhs[i][0] = (x + y) + '0';

rhs[i][1] = ' ';

rhs[i][2] = ' ';

}

else if (rhs[i][1] == '-'){

int x = rhs[i][0] - '0';

int y = rhs[i][2] - '0';

rhs[i][0] = (x - y) + '0';

rhs[i][1] = ' ';

rhs[i][2] = ' ';

}

else if (rhs[i][1] == '\*'){

int x = rhs[i][0] - '0';

int y = rhs[i][2] - '0';

rhs[i][0] = (x \* y) + '0';

rhs[i][1] = ' ';

rhs[i][2] = ' ';

}

else if (rhs[i][1] == '/'){

int x = rhs[i][0] - '0';

int y = rhs[i][2] - '0';

rhs[i][0] = (x / y) + '0';

rhs[i][1] = ' ';

rhs[i][2] = ' ';

}

}

}

printf("\n ------------------------------------------- \nConstant Folding\n -------------------------------------------\n");

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

printf("%s := %s \n", lhs[i], rhs[i]);

}

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

int len = strlen(rhs[i]);

if (len == 5){

if (rhs[i][0] == '2' && rhs[i][1] == '\*'){

if (rhs[i][2] >= 'a' && rhs[i][2] <= 'z'){ rhs[i][0] = rhs[i][2];

rhs[i][1] = '+';

}

}

else if (rhs[i][1] == '\*' && rhs[i][2] == '2'){ if (rhs[i][0] >= 'a' && rhs[i][0] <= 'z'){

rhs[i][1] = '+';

rhs[i][2] = rhs[i][0];

}

}

}

}

printf("\n ------------------------------------------- \nStrength Reduction\n ------------------------------------------ \n");

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

printf("%s := %s \n", lhs[i], rhs[i]);

}

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

printf("line %d ====> %s := %s \n", i, lhs[i], rhs[i]);

}

printf("\nNumber of basic blocks: %d\n", lnum + 1);

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

printf("| Leader | Line |\n");

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

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

printf("| %d | %d |\n",(i+1),leaders[i]);

}

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

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

char \*gt = strstr(tac[leaders[i]], "goto"); char \*t = strstr(tac[leaders[i]], "true"); if (gt && t){

int goto\_num\_units, goto\_num;

int last = strlen(tac[leaders[i]]); if (isdigit(tac[leaders[i]][15])){

goto\_num\_units = tac[leaders[i]][15] - '0';

goto\_num = tac[leaders[i]][14] - '0';

goto\_num = goto\_num \* 10 + goto\_num\_units;

}

else{

goto\_num = tac[leaders[i]][14] - '0';

}

if (goto\_num < leaders[i]){

printf("If we consider line %s, dead code found from %d to line %d\n", tac[leaders[i]], leaders[i], loc);

}

else{

printf("If we consider line %s, dead code found from line %d to line %d\n", tac[leaders[i]], leaders[i], goto\_num);

}

}

}

Output:

A screenshot of a computer

Description automatically generated with medium confidence







