SSN COLLEGE OF ENGINEERING, KALAVAKKAM

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UCS1602 - Compiler Design

Programming Assignment 6

Implementation of Code optimization techniques

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 20500104*

Source code:

```c
#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: ");
    scanf("%s", 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  \nConstantFolding\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:

```
Enter name of a file: ip.txt
Input file contents: prod:=0
i:=1
t1:=b*c
t2:=a-t1
t3:=b*c
t4:=t2+t3
prod:=t4
if true goto (2)
t5:=i+0
i:=t5
if(i<20) goto (3)
t6:=4*i
t7:=b*0
t8:=0*b
t9:=0+i
if true goto (21)
t10:=2+3
t11:=3*3
t12:=9/3
t13:=9-2
t14:=a*2
t15:=2*b
```

```
------------------------
Algebraic Identity
------------------------
prod := 0

i := 1

t1 := b*c

t2 := a-t1

t3 := b*c

t4 := t2+t3

prod := t4

if true goto (2)
 :=
```

```
prod := 0

i := 1

t1 := b*c

t2 := a-t1

t3 := b*c

t4 := t2+t3

prod := t4

if true goto (2)
 :=

t5 := i+0

i := t5

if(i<20) goto (3)
 :=

t6 := 4*i

t7 := b*0

t8 := 0*b

t9 := 0+i

if true goto (21)
 :=

t10 := 2+3

t11 := 3*3

t12 := 9/3

t13 := 9-2

t14 := a*2

t15 := 2*b
```

```
ConstantFolding
----------------------
prod := 0

i := 1

t1 := b*c

t2 := a-t1

t3 := b*c

t4 := t2+t3

prod := t4

if true goto (2)
 :=

t5 := i+0

i := t5

if(i<20) goto (3)
 :=

t6 := 4*i

t7 := b*0

t8 := 0*b

t9 := 0+i

if true goto (21)
 :=

t10 := 2+3

t11 := 3*3

t12 := 9/3

t13 := 9-2

t14 := a*2

t15 := 2*b
```

```
----------------------
Strength Reduction
----------------------
prod := 0

i := 1

t1 := b*c

t2 := a-t1

t3 := b*c

t4 := t2+t3

prod := t4

if true goto (2)
 :=

t5 := i+0

i := t5

if(i<20) goto (3)
 :=

t6 := 4*i

t7 := b*0

t8 := 0*b

t9 := 0+i

if true goto (21)
 :=

t10 := 2+3

t11 := 3*3

t12 := 9/3

t13 := 9-2

t14 := a*2

t15 := 2*b
```

```
t14 := a*2

t15 := 2*b

line 0 ====> prod := 0

line 1 ====> i := 1

line 2 ====> t1 := b*c

line 3 ====> t2 := a-t1

line 4 ====> t3 := b*c

line 5 ====> t4 := t2+t3

line 6 ====> prod := t4

line 7 ====> if true goto (2)
 :=

line 8 ====> t5 := i+0

line 9 ====> i := t5

line 10 ====> if(i<20) goto (3)
 :=

line 11 ====> t6 := 4*i

line 12 ====> t7 := b*0

line 13 ====> t8 := 0*b

line 14 ====> t9 := 0+i

line 15 ====> if true goto (21)
 :=

line 16 ====> t10 := 2+3

line 17 ====> t11 := 3*3

line 18 ====> t12 := 9/3

line 19 ====> t13 := 9-2

line 20 ====> t14 := a*2
```

```
line 6 ====> prod := t4

line 7 ====> if true goto (2)
 :=

line 8 ====> t5 := i+0

line 9 ====> i := t5

line 10 ====> if(i<20) goto (3)
 :=

line 11 ====> t6 := 4*i

line 12 ====> t7 := b*0

line 13 ====> t8 := 0*b

line 14 ====> t9 := 0+i

line 15 ====> if true goto (21)
 :=

line 16 ====> t10 := 2+3

line 17 ====> t11 := 3*3

line 18 ====> t12 := 9/3

line 19 ====> t13 := 9-2

line 20 ====> t14 := a*2

line 21 ====> t15 := 2*b


Number of basic blocks: 7
---------------------------------
|    Leader    |     Line        |
---------------------------------
|       1      |       0         |
|       2      |       7         |
|       3      |       8         |
|       4      |       10        |
|       5      |       11        |
|       6      |       15        |
|       7      |       16        |
---------------------------------
```

```
Number of basic blocks: 7
---------------------------------
|    Leader    |     Line        |
---------------------------------
|       1      |       0         |
|       2      |       7         |
|       3      |       8         |
|       4      |       10        |
|       5      |       11        |
|       6      |       15        |
|       7      |       16        |
---------------------------------
If we consider line if true goto (2)
, dead code found from 7 to line 22
If we consider line if true goto (21)
, dead code found from line 15 to line 21
```

Learning Outcome:

- Understood the working of yacc for debugging of programs.
- Understood the role of yacc in running a program
- Understood how to write yacc programs