

# Department of CSE

## SSN College of Engineering

Vishakan Subramanian - 18 5001 196 - Semester VI

29 April 2021

---

### UCS 1602 - Compiler Design

---

#### Exercise 8: Code Optimization Using C

##### **Aim:**

Implement a program in C to perform code expression for simple code expressions, including simple mathematical identities and strength reduction.

## Code: Optimizer.c:

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4
5 int main(int argc, char *argv[]){
6     stdin = fopen(argv[1], "r");
7     char line[100];
8
9     printf("\n---Parsed Code Starts---\n\n");
10    system("cat Code.txt");
11    printf("\n\n---Parsed Code Ends---\n");
12
13    printf("\n---Optimized Code Starts---\n\n");
14
15    scanf(" %s[^\n]", line);
16
17    while (strcmp(line, "END") != 0){
18
19        //FOR: x=y+0; x=y*1;
20        if ((line[3] == '+' && line[4] == '0') || (line[3] == '*' && line
21[4] == '1')){
22            if (line[0] == line[2]){
23                scanf(" %s[^\n]", line);
24                continue;
25            }
26            printf("%c=%c\n", line[0], line[2]);
27        }
28
29        //FOR: x=y-0; x=y/1;
30        else if ((line[3] == '-' && line[4] == '0') || (line[3] == '/' &&
31line[4] == '1')){
32            if (line[0] == line[2]){
33                scanf(" %s[^\n]", line);
34                continue;
35            }
36            printf("%c=%c\n", line[0], line[2]);
37        }
38
39        //FOR: x=0+x; x=1*y
40        else if ((line[3] == '+' && line[2] == '0') || (line[3] == '*' &&
41line[2] == '1')){
42            if (line[0] == line[4]){
43                scanf(" %s[^\n]", line);
44                continue;
45            }
46            printf("%c=%c\n", line[0], line[4]);
47        }
48    }
```

```

45     }
46
47     //FOR: x=y*2
48     else if (line[3] == '*' && line[4] == '2'){
49         printf("%c=%c+%c\n", line[0], line[2], line[2]);
50     }
51
52     //FOR: x=2*y
53     else if (line[3] == '*' && line[2] == '2'){
54         printf("%c=%c+%c\n", line[0], line[4], line[4]);
55     }
56
57     //FOR: x=pow(y,2);
58     else if (line[2] == 'p' && line[3] == 'o' && line[4] == 'w' &&
line[5] == '(' && line[8] == '2'){
59         printf("%c=%c*%c\n", line[0], line[6], line[6]);
60     }
61
62     else{
63         printf("%s\n", line);
64     }
65
66     scanf(" %s[^\n]", line);    //next line
67 }
68
69 printf("\n---Optimized Code Ends---\n");
70
71 return 0;
72 }

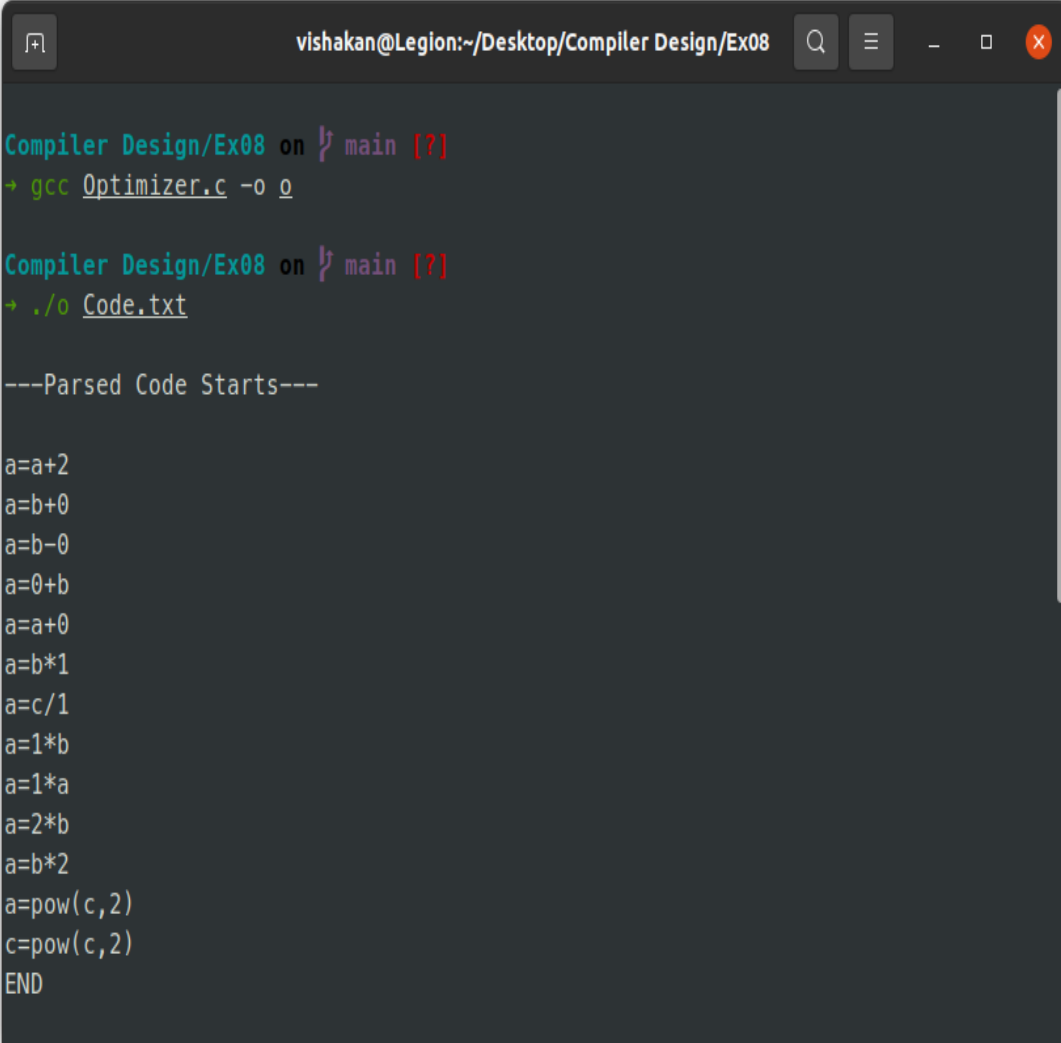
```

## Parsed Code:

```
1 a=a+2
2 a=b+0
3 a=b-0
4 a=0+b
5 a=a+0
6 a=b*1
7 a=c/1
8 a=1*b
9 a=1*a
10 a=2*b
11 a=b*2
12 a=pow(c,2)
13 c=pow(c,2)
14 END
```

## Output: Parsed Code:

Figure 1: Console Output - Parsed Code.

A terminal window with a dark background and light-colored text. The window title bar shows the user 'vishakan@Legion' and the directory '~/Desktop/Compiler Design/Ex08'. The terminal content shows a shell prompt 'main [?]' followed by the command 'gcc Optimizer.c -o o'. Another prompt 'main [?]' is followed by './o Code.txt'. Below this, the text '---Parsed Code Starts---' is displayed, followed by a list of assignment statements: 'a=a+2', 'a=b+0', 'a=b-0', 'a=0+b', 'a=a+0', 'a=b\*1', 'a=c/1', 'a=1\*b', 'a=1\*a', 'a=2\*b', 'a=b\*2', 'a=pow(c,2)', 'c=pow(c,2)', and 'END'.

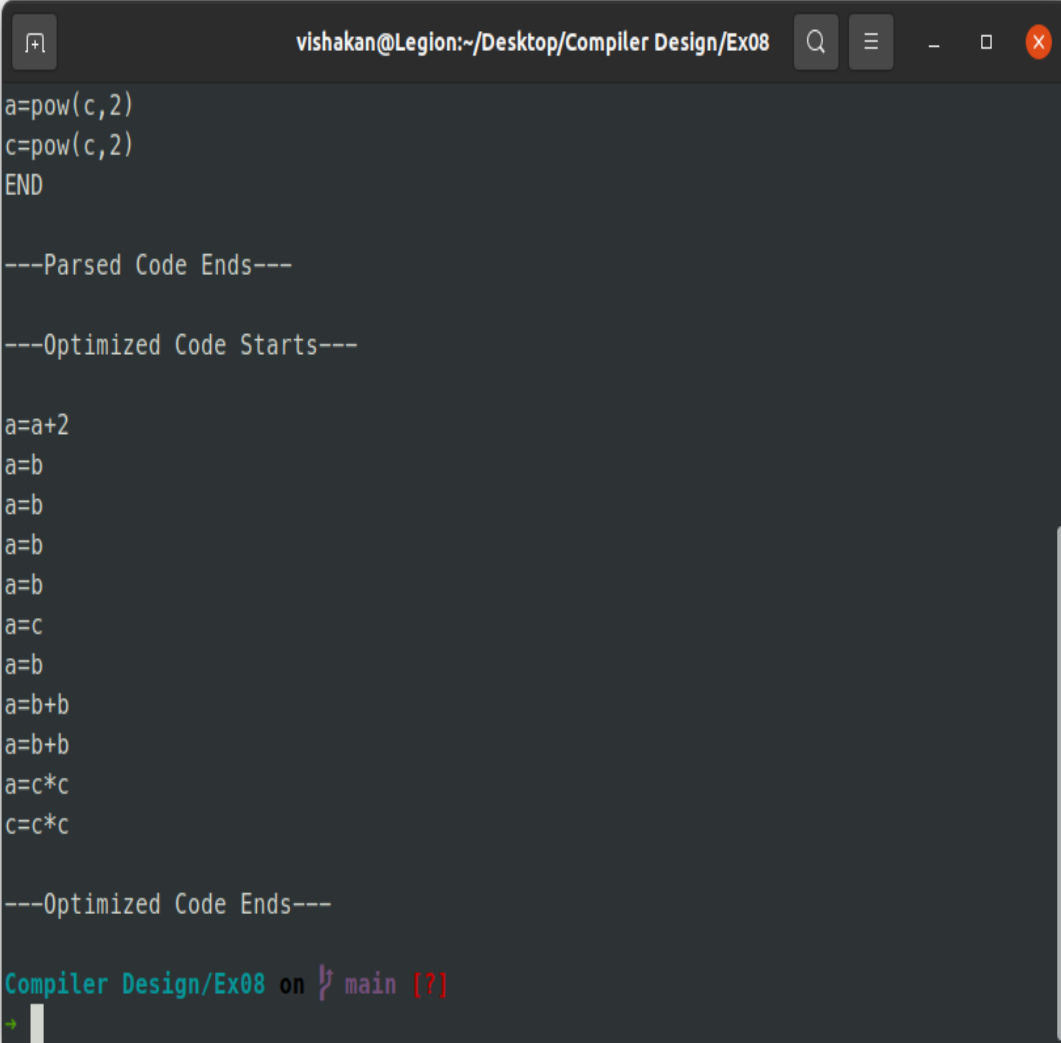
```
vishakan@Legion:~/Desktop/Compiler Design/Ex08
Compiler Design/Ex08 on main [?]
→ gcc Optimizer.c -o o
Compiler Design/Ex08 on main [?]
→ ./o Code.txt

---Parsed Code Starts---

a=a+2
a=b+0
a=b-0
a=0+b
a=a+0
a=b*1
a=c/1
a=1*b
a=1*a
a=2*b
a=b*2
a=pow(c,2)
c=pow(c,2)
END
```

## Output: Optimized Code:

Figure 2: Console Output - Optimized Code.

A terminal window with a dark background and light text. The title bar at the top reads 'vishakan@Legion:~/Desktop/Compiler Design/Ex08'. The terminal content shows the output of a compiler optimization process. It starts with the original code: 'a=pow(c,2)', 'c=pow(c,2)', and 'END'. This is followed by a separator '---Parsed Code Ends---'. Then, the optimized code is displayed, starting with '---Optimized Code Starts---'. The optimized code consists of several assignment and arithmetic statements: 'a=a+2', 'a=b', 'a=b', 'a=b', 'a=b', 'a=c', 'a=b', 'a=b+b', 'a=b+b', 'a=c\*c', and 'c=c\*c'. This is followed by the separator '---Optimized Code Ends---'. At the bottom, a prompt indicates the current file is 'Compiler Design/Ex08' and the current function is 'main'.

```
vishakan@Legion:~/Desktop/Compiler Design/Ex08
a=pow(c,2)
c=pow(c,2)
END

---Parsed Code Ends---

---Optimized Code Starts---

a=a+2
a=b
a=b
a=b
a=b
a=c
a=b
a=b+b
a=b+b
a=c*c
c=c*c

---Optimized Code Ends---

Compiler Design/Ex08 on main [?]
```

## Learning Outcome:

- I learnt about code optimization.
- I learnt why code optimization is useful in the compilation process.
- I understood the different logical techniques that go behind the implementation of a code optimizer.
- I was able to implement some simple optimization techniques using C.
- I successfully optimized basic code containing redundant operations.
- I successfully optimized code using the strength reduction technique, in which I converted powers of 2 to simple multiplication.
- From the implementation, I was able to visualize different other similar code optimization techniques that I learnt in Compiler Theory and how it proves to be effective in code execution.