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:

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

int main(int argc, char *argv[]){
    stdin = fopen(argv[1], "r");
    char line[100];

    printf("\n---Parsed Code Starts---\n\n");
    system("cat Code.txt");
    printf("\n\n---Parsed Code Ends---\n");

    printf("\n---Optimized Code Starts---\n\n");

    scanf(" %s[^\n]", line);

    while (strcmp(line, "END") != 0){

        //FOR: x=y+0; x=y*1;
        if ((line[3] == '+' && line[4] == '0') || (line[3] == '*' && line[4] == '1')){
            if (line[0] == line[2]){
                scanf(" %s[^\n]", line);
                continue;
            }

            printf("%c=%c\n", line[0], line[2]);
        }

        //FOR: x=y-0; x=y/1;
        else if ((line[3] == '-' && line[4] == '0') || (line[3] == '/' && line[4] == '1')){
            if (line[0] == line[2]){
                scanf(" %s[^\n]", line);
                continue;
            }
            printf("%c=%c\n", line[0], line[2]);
        }

        //FOR: x=0+x; x=1*y
        else if ((line[3] == '+' && line[2] == '0') || (line[3] == '*' && line[2] == '1')){
            if (line[0] == line[4]){
                scanf(" %s[^\n]", line);
                continue;
            }
            printf("%c=%c\n", line[0], line[4]);
```

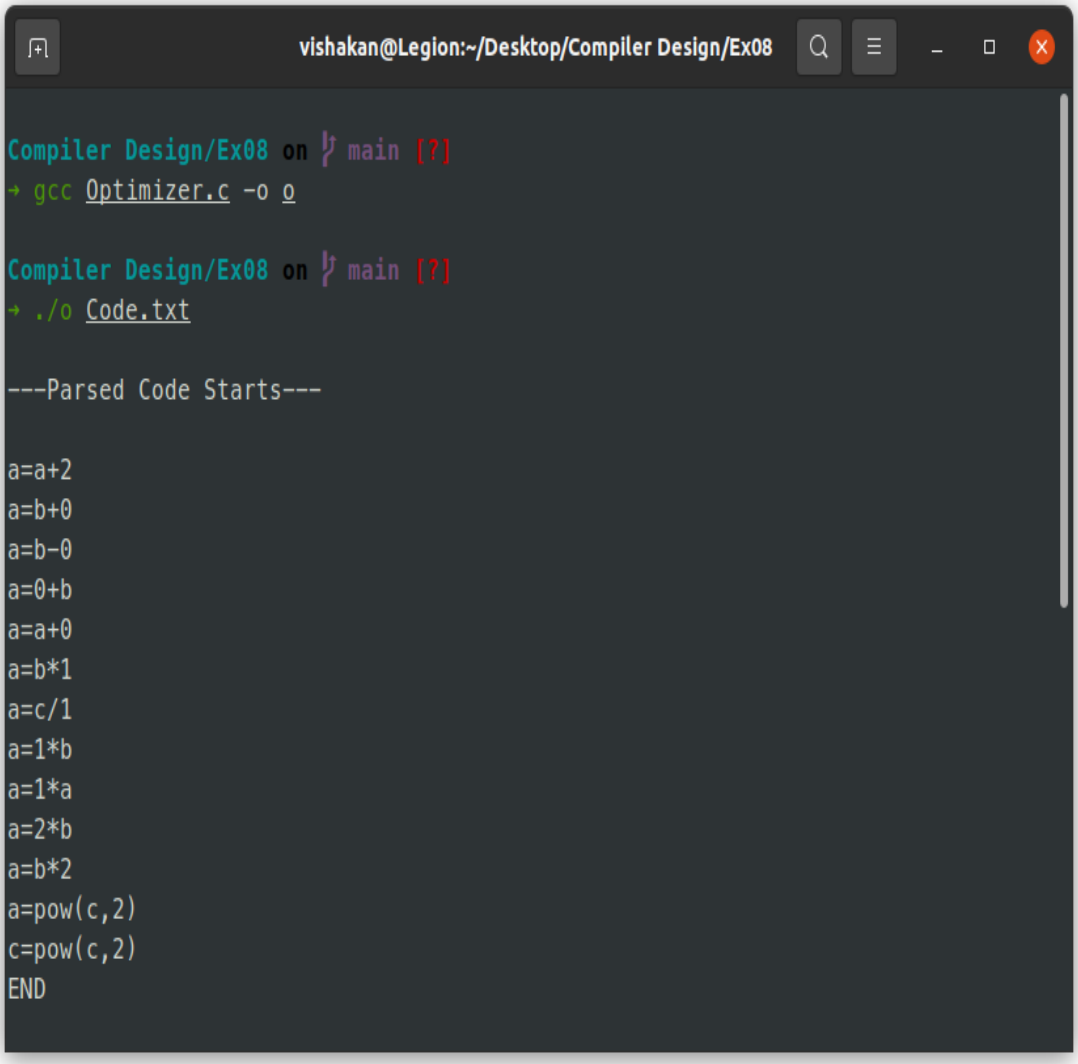
```c
        }

        //FOR: x=y*2
        else if (line[3] == '*' && line[4] == '2'){
            printf("%c=%c+%c\n", line[0], line[2], line[2]);
        }

        //FOR: x=2*y
        else if (line[3] == '*' && line[2] == '2'){
            printf("%c=%c+%c\n", line[0], line[4], line[4]);
        }

        //FOR: x=pow(y,2);
        else if (line[2] == 'p' && line[3] == 'o' && line[4] == 'w' && line[5] == '(' && line[8] == '2'){
            printf("%c=%c*%c\n", line[0], line[6], line[6]);
        }

        else{
            printf("%s\n", line);
        }

        scanf(" %s[^\n]", line);    //next line
    }

    printf("\n---Optimized Code Ends---\n");

    return 0;
}
```

Parsed Code:

```
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
```

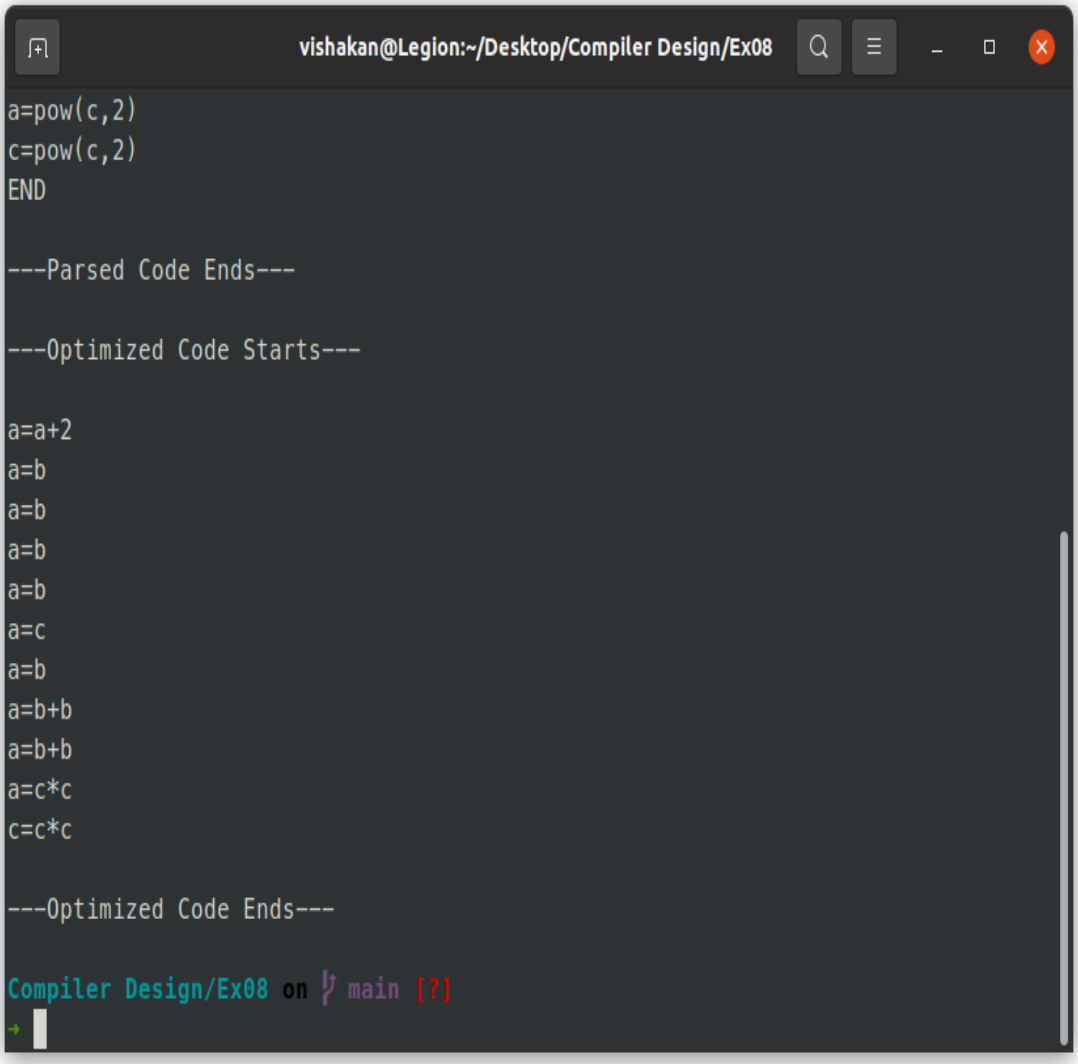
Output: Parsed Code:

Figure 1: Console Output - Parsed Code.



Output: Optimized Code:

Figure 2: Console Output - Optimized Code.



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 sucessfully 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.