Experiment No.8

Title: - Code Optimization

Aim:- Implement Code optimization techniques.
(a) Common Sub Expression Elimination
(b) Strength Reduction
(c) Constant Folding

Theory:-

A piece of code is said to be optimized if it consumes less processing time and/or less memory space. An optimizing transformation of a program is defined as one which yields better or more desirable program compare to the program we started out with. The resulting program is considered better because it either runs faster than the initial program or it uses less storage. An optimizing transformation may be applied either to source program or to object program.
Types of optimizations
1 Machine dependent optimization – Machine dependent optimization focuses on how features of the native machine language and its architecture can be exploited to generate optimized code. There are some sources of machine dependent optimization

- Jumps to jumps must be avoided
- Jumps to calls must be avoided
- Call recursion elimination

2 Machine independent optimization – There are two broad categories of machine-independent optimization. Function preserving transformations and loop optimizations.
Common sub expression elimination- If an expression gets repeated more than once and values of its variables do not change between its occurrences, then the expression is called common sub-expression. This common sub-expression can be computed once and its value can be used in all remaining occurrences.

Eg:	t1 = 4* I	After optimization	t1 = 4 * i
	x= a[t1]	----------□	x = a[t1]
	t2= 4* I		y = a[t1]
	y= a[t2]		

Constant Folding- If an expression involves all constants, it can be evaluated compile time itself. It reduces execution time of the program.

	After optimization	
Eg: A = 3.12/2	----------□	A = 1.56

strength reduction is a compiler optimization where expensive operations are replaced with equivalent but less expensive operations. The classic example of strength reduction converts "strong" multiplications inside a loop into "weaker" additions – something that frequently occurs in array addressing.
Examples of strength reduction include:

- replacing a multiplication within a loop with an addition
- replacing an exponentiation within a loop with a multiplication

example:

```
c = 8;
for (i = 0; i < N; i++)
{
    y[i] = c * i;
}
```

can be replaced with successive weaker additions

```
c = 8;
k = 0;
for (i = 0; i < N; i++)
{
    y[i] = k;
    k = k + c;
}
```

Exercise:-