

**Lab Terminal**

**Submitted By:**

Ramsha Kokab

FA21-BCS-010

**Course Instructor:**

Syed Bilal Haider Bukhari

**Course:**

Compiler Construction [Lab]

**Date:**

3rd January, 2024

**DEPARTMENT OF COMPUTER SCIENCE**

**COMSATS UNIVERSITY ISLAMABAD, ATTOCK CAMPUS**

### Q3: Explain 4 optimizations in your mini compiler.

1. **Constant Folding**

* Occured during the Semantic Analysis or Intermediate Code Generation phase.
* The compiler’s optimization module would scan for expressions involving constants and simplify them directly.

*Input Code:*

int x = 3 + 5;

*Optimized Code:*

int x = 8;

1. **Dead Code Elimination**

* Occured during the **Intermediate Code Optimization** phase. The compiler analyzes the control flow graph (CFG) or intermediate representation to identify and eliminate such code.

*Input Code:*

int x = 5;

return 0;

x = x + 1; // This code is never executed.

*Optimized Code:*

int x = 5;

return 0;

1. **Strength Reduction**

* The compiler would check for multiplication or division operations during the **Intermediate Code Generation** phase and replace them with efficient operations.

*Input Code:*

int y = x \* 2;

*Optimized Code:*

int y = x + x;

1. **Common Subexpression Elimination**

* During the **Intermediate Code Optimization** phase, the compiler scans the IR to detect duplicate computations and replaces them with a single stored result.

*Input Code:*

int z = (x + y) \* 2;

int w = (x + y) / 3;

*Optimized Code:*

int temp = x + y;

int z = temp \* 2;

int w = temp / 3;