Logo, company name

Description automatically generated

**COMSATS University Islamabad (CUI)**

**Lab Terminal**

**Submitted to: Sir Bilal Bukhari**

**Submitted By: Sohaib Safeer**

**Reg No: FA21-BCS-054**

**Course: Topics in Computer Science**

**Date: 3 jan, 2025**

Question 4 : give example for optimization used in mini compiler

**Example of Constant Folding Optimization in a Mini Compiler**

**Scenario:**

Consider the following code snippet:

C

int x = 2 \* (3 + 4);

**Without Optimization:**

1. **Parsing:** The compiler would parse the expression, breaking it down into its constituent parts:
   * x (variable)
   * = (assignment operator)
   * 2 (constant)
   * \* (multiplication operator)
   * ( (left parenthesis)
   * 3 (constant)
   * + (addition operator)
   * 4 (constant)
   * ) (right parenthesis)
2. **Intermediate Code Generation:** The compiler would generate intermediate code representing these operations, such as:
3. LOAD 3
4. PUSH
5. LOAD 4
6. PUSH
7. ADD
8. LOAD 2
9. PUSH
10. MUL
11. STORE x
12. **Code Generation:** The generated code would then be translated into machine code for the target architecture, resulting in instructions to load constants, perform arithmetic operations, and store the result in memory.

**With Constant Folding Optimization:**

1. **Constant Expression Evaluation:** During the parsing or intermediate code generation phase, the compiler would recognize the constant expression (3 + 4) and evaluate it directly to 7.
2. **Simplified Intermediate Code:** The intermediate code would be simplified to:
3. LOAD 2
4. PUSH
5. LOAD 7
6. PUSH
7. MUL
8. STORE x
9. **Code Generation:** The generated machine code would now be more efficient, as it directly loads the constant 7 instead of performing the intermediate addition operation.

**Benefits:**

* **Reduced Runtime:** By performing constant calculations at compile time, the number of runtime instructions is reduced, leading to faster execution.
* **Smaller Code Size:** The optimized code may be smaller in size as it eliminates unnecessary instructions.
* **Improved Performance:** Overall, constant folding can significantly improve the performance of the generated code, especially in cases where constant expressions are frequently used.

This is a simple example of how constant folding can be applied in a mini compiler. More sophisticated optimizations, such as common subexpression elimination, dead code elimination, and strength reduction, can also be incorporated to further enhance the performance of the generated code.