****

**Lab Terminal # 04**

**Submitted by:** Isha Eman

## **Reg #:** fa21-bcs-046

**Give examples of four optimization in your mini compiler.**

1. **Constant Folding**: Evaluate constant expressions during compile time. int x = 5 + 3; should be replaced by int x = 8;

**Code Example:**-

public AstNode Optimize(AstNode node) {

if (node is BinaryOpNode binaryOp &&

binaryOp.Left is IntLiteralNode left &&

binaryOp.Right is IntLiteralNode right)

{

int result = 0;

if(binaryOp.Operator==TokenType.PLUS){

result = left.Value + right.Value;

}

else if(binaryOp.Operator==TokenType.MINUS){

result = left.Value - right.Value;

}

return new IntLiteralNode(result);

}

//recursive call to visit the rest of nodes

return node;

}

**Example**

int x = 5 + 3; // Before Optimization

// After constant folding:

int x = 8; // 5 + 3 is computed at compile-time

1. **Dead Code Elimination**: Removing unreachable code. Remove the code after an unconditional return.

**Code Example:**-

public List<AstNode> RemoveDeadCode(List<AstNode> statements)

{

List<AstNode> optimizedStatements = new List<AstNode>();

bool unreachable = false;

foreach(AstNode statement in statements){

if(unreachable) continue; //skip unreachable statements

if(statement is ReturnStatementNode){

unreachable = true;

}

optimizedStatements.Add(statement);

}

return optimizedStatements;

}

**Example**

void func() {

int x = 10;

return; // After this point, all code is unreachable

int y = 20; // This code is eliminated during optimization

}

1. **Instruction Selection:** Select more efficient assembly instructions, such as replacing

 x = x + 1 by increment x if there is a increment instruction available in target architecture.

**Code Example:**-

public void GenerateCode(AstNode node)

{

switch (node)

{

case BinaryOpNode binaryOpNode:

if(binaryOpNode.Operator == TokenType.PLUS &&

binaryOpNode.Right is IntLiteralNode right &&

right.Value == 1 ) {

GenerateCode(binaryOpNode.Left);

\_instructions.Add($"increment");

}

else { //do usual code generation

GenerateCode(binaryOpNode.Left);

GenerateCode(binaryOpNode.Right);

\_instructions.Add($"{binaryOpNode.Operator.ToString()}");

}

break;

}

}

**Eaxmple:-**

x = x + 1; // Before optimization

// After instruction selection (assuming target architecture supports increment):

increment x; // More efficient, single instruction

1. **Peephole Optimization:** Examine a small window of instructions for specific inefficient pattern and replaced with a better one. Example load x, add 1 may be replaced by a single instruction increment x.

**Code Example:**-

public List<string> PeepholeOptimize(List<string> instructions){

List<string> optimizedInstructions = new List<string>();

for(int i =0; i< instructions.Count; i++){

if(i + 1 < instructions.Count && instructions[i].StartsWith("load") && instructions[i+1] == "add 1"){

optimizedInstructions.Add($"increment {instructions[i].Split(' ')[1]}");

i++;

}

else {

optimizedInstructions.Add(instructions[i]);

}

}

return optimizedInstructions;

}

**Example:-**

load x // Before peephole optimization

add 1

// After peephole optimization:

increment x // More efficient single instruction