Practical 7

Name : Ansh Agrawal

Roll No. : 20

Batch : A2

In [1]:

```
class Optimizer:
    def __init__(self, tac_code):
        self.tac_code = tac_code
        self.optimized_code = []
        self.optimization_log = []

    def apply_copy_propagation(self):
        temp_vars = {}
        for line in self.tac_code:
            if line.startswith('t') and '=' in line:
                var, expr = line.split('=')
                var = var.strip()
                expr = expr.strip()
                if expr in temp_vars:
                    optimized_line = f'{var} = {temp_vars[expr]}'
                    if optimized_line != line:
                        self.optimized_code.append(optimized_line)
                        self.optimization_log.append(f'Copy Propagation: {optimized_line}')
                else:
                    self.optimized_code.append(line)
                temp_vars[var] = expr
            else:
                self.optimized_code.append(line)

    def apply_constant_propagation(self):
        const_vars = {}
        new_code = []
```

```
        for line in self.optimized_code:
            if line.startswith('t') and '=' in line:
                var, expr = line.split('=')
                var = var.strip()
                expr = expr.strip()
                if expr.isdigit():
                    const_vars[var] = int(expr)
                    new_line = f'{var} = {int(expr)}'
                    if new_line != line:
                        new_code.append(new_line)
                        self.optimization_log.append(f'Constant Propagation: {new_line}')
                elif expr in const_vars:
                    new_line = f'{var} = {const_vars[expr]}'
                    if new_line != line:
                        new_code.append(new_line)
                        self.optimization_log.append(f'Constant Propagation: {new_line}')
                else:
                    new_code.append(line)
            else:
                new_code.append(line)
        self.optimized_code = new_code

    def apply_constant_folding(self):
        for i, line in enumerate(self.optimized_code):
            if line.startswith('t') and '=' in line:
                var, expr = line.split('=')
                var = var.strip()
                expr = expr.strip()
                try:
                    result = eval(expr)
                    new_line = f'{var} = {result}'
                    if new_line != line:  # Check if it changed
                        self.optimized_code[i] = new_line
                        self.optimization_log.append(f'Constant Folding: {new_line}')
                except Exception:
                    continue

    def apply_common_subexpression_elimination(self):
        subexprs = {}
        new_code = []
        for line in self.optimized_code:
            if line.startswith('t') and '=' in line:
                var, expr = line.split('=')
                var = var.strip()
```

```
                expr = expr.strip()
                if expr in subexprs:
                    new_line = f'{var} = {subexprs[expr]}'
                    if new_line != line:
                        new_code.append(new_line)
                        self.optimization_log.append(f'Common Subexpression Elimination: {new_line}')
                else:
                    subexprs[expr] = var
                    new_code.append(line)
            else:
                new_code.append(line)
        self.optimized_code = new_code

    def optimize(self):
        self.optimized_code = self.tac_code.copy()
        self.optimization_log.append('Initial TAC Code')

        self.apply_copy_propagation()
        self.optimization_log.append('After Copy Propagation')

        self.apply_constant_propagation()
        self.optimization_log.append('After Constant Propagation')

        self.apply_constant_folding()
        self.optimization_log.append('After Constant Folding')

        self.apply_common_subexpression_elimination()
        self.optimization_log.append('After Common Subexpression Elimination')

        return self.optimized_code, self.optimization_log

# Shorter example TAC code
tac_code = [
    't1 = a + b',
    't2 = t1',
    't3 = 3 + 4',
    't4 = t3 * 2',
    't5 = t4 + t1'
]

optimizer = Optimizer(tac_code)
optimized_code, optimization_log = optimizer.optimize()

print("Optimized TAC Code:")
```

```python
for line in optimized_code:
    print(line)

print("\nOptimization Steps:")
for log in optimization_log:
    print(log)
```

```
Optimized TAC Code:
t1 = a + b
t2 = t1
t3 = 7
t4 = t3 * 2
t5 = t4 + t1
t1 = t1
t2 = t1
t3 = t3
t4 = t4
t5 = t5

Optimization Steps:
Initial TAC Code
Copy Propagation: t2 = a + b
After Copy Propagation
After Constant Propagation
Constant Folding: t3 = 7
Constant Folding: t3 = 7
After Constant Folding
Common Subexpression Elimination: t1 = t1
Common Subexpression Elimination: t2 = t1
Common Subexpression Elimination: t3 = t3
Common Subexpression Elimination: t4 = t4
Common Subexpression Elimination: t5 = t5
After Common Subexpression Elimination
```