Premier University

Department of Computer Science and Engineering

Course Title: Compiler Construction

Course Code: CSE 453

Compiler Construction Course's

Assignment on

"Code Optimization and Generation in a Simple Compiler"

Submission Date: 24/11/2025

Submitted By

Rayanul kader Chowdhury Abid

210401020 2162

8th Semester A Section

Task 1: Intermediate Code Generation

Objective: Convert the high-level arithmetic into strict Three-Address Code (TAC).

The input program contains complex expressions (e.g., a + b * c) that must be broken down according to operator precedence (Multiplication * has higher precedence than Addition +). We will introduce temporary variables (v1, v2, etc.) to hold intermediate results.

Generated Three-Address Code:

1. v1 = b * c
2. t1 = a + v1
3. v2 = b * c
4. t2 = a + v2
5. t3 = t1 + t2
6. if t3 > 10 goto L1
7. t4 = t3 * 2
8. L1: t5 = t4 + 1

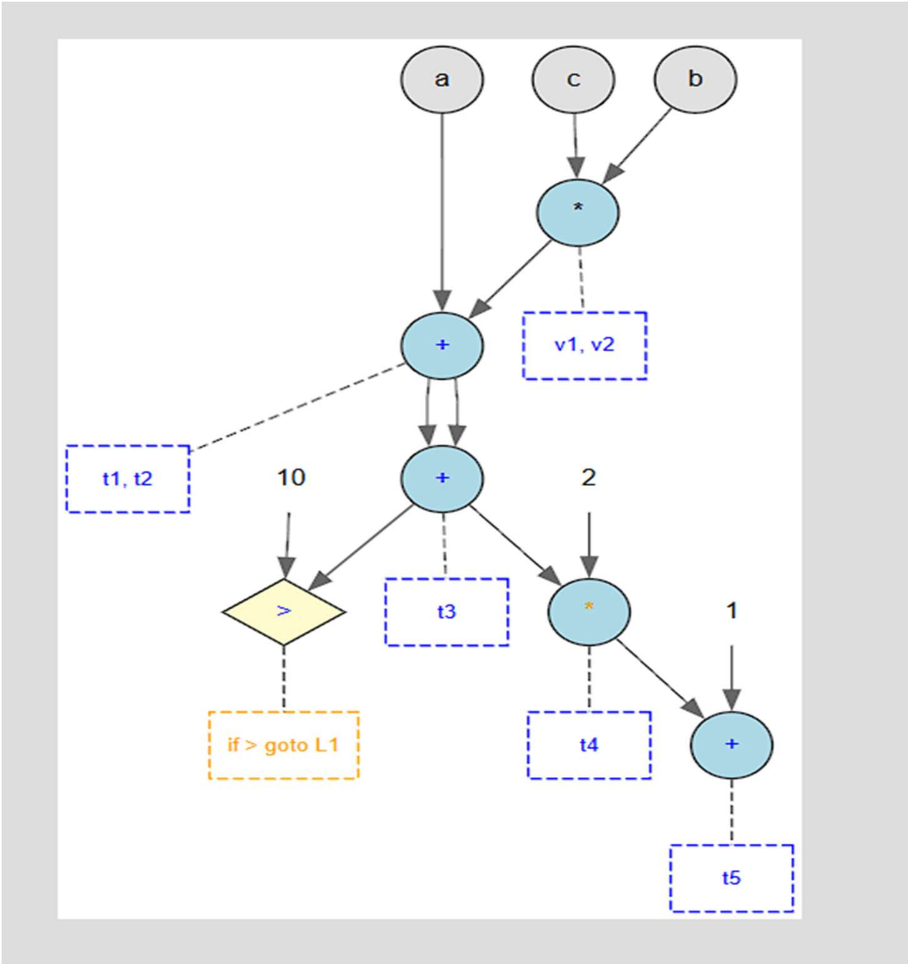
Task 2: DAG Construction and Optimization

Objective: Construct a Directed Acyclic Graph (DAG) to identify Common Sub-expressions (CSE) and eliminate redundancy.

2.1 DAG Construction Analysis

We process the TAC codes line by line to build the nodes:

1. **Leaf Nodes:** Created for a, b, c, 10, 2, 1.
2. **Instruction 1 (v1 = b * c):** Create Node * with children b and c. Attach label v1.
3. **Instruction 2 (t1 = a + v1):** Create Node + with children a and Node(v1). Attach label t1.
4. **Instruction 3 (v2 = b * c):** The computation b * c already exists in the DAG (Node v1). We do not create a new node. Attach label v2 to Node(v1).
5. **Instruction 4 (t2 = a + v2):** The computation a + v2 (which is a + v1) already exists (Node t1). Attach label t2 to Node(t1).
6. **Instruction 5 (t3 = t1 + t2):** Create Node +. Since t1 and t2 point to the same node, both children are Node(t1). Attach label t3.
7. **Instruction 6 (if t3 > 10 goto L1):** Create a comparison Node > with children t3 and 10.
8. **Instruction 7 (t4 = t3 * 2):** Create Node * with children t3 and 2. Attach label t4.
9. **Instruction 8 (t5 = t4 + 1):** Create Node + with children t4 and 1. Attach label t5.



2.2 Redundancy Elimination

- **Common Sub-expression identified:** b * c (Labels v1 and v2 are equivalent).
- **Common Sub-expression identified:** a + (b * c) (Labels t1 and t2 are equivalent).
- **Optimization:** Replace usage of v2 with v1, and t2 with t1. Consequently, t3 = t1 + t2 becomes t3 = t1 + t1 (or 2 * t1).

2.3 Optimized TAC

1. v1 = b * c
2. t1 = a + v1
3. t3 = t1 + t1
4. if t3 > 10 goto L1
5. t4 = t3 * 2
6. L1: t5 = t4 + 1

Task 3: Basic Block Identification

Objective: Partition the optimized TAC into Basic Blocks.

3.1 Identify Leaders

A "Leader" is the first instruction of a basic block.

1. **Leader 1:** The first instruction of the program.
   - (1) v1 = b * c
2. **Leader 2:** The target of a conditional or unconditional jump.
   - (6) L1: t5 = t4 + 1
3. **Leader 3:** The instruction immediately following a conditional jump.
   - (5) t4 = t3 * 2

3.2 Basic Blocks and Control Flow

Block B1 (Entry):

1. v1 = b * c
2. t1 = a + v1
3. t3 = t1 + t1
4. if t3 > 10 goto L1

Block B2 (Else branch):

5. t4 = t3 * 2

Block B3 (Target L1):

6. L1: t5 = t4 + 1

3.3 Flow Diagram

- **B1** flows to **B2** (if condition is False).
- **B1** flows to **B3** (if condition is True).
- **B2** flows to **B3** (Sequential fall-through).

Task 4: Code Generation

Objective: Generate assembly code from Optimized TAC. **Assumptions:**

- We use a register-based machine (R0, R1, R2...).
- Instructions: MOV (Move), ADD (Add), MUL (Multiply), CMP (Compare), JGT (Jump Greater Than).
- Variables a, b, c are in memory.

Assembly Code:

```
MOV R0, b
MUL R0, c
MOV R1, a
ADD R1, R0
ADD R1, R1
CMP R1, 10
JGT L1

MOV R2, R1
MUL R2, 2
MOV t4, R2

L1:
MOV R3, t4
ADD R3, 1
MOV t5, R3
```

Task 5: Cost Estimation

Objective: Estimate the cost of the optimized code. **Cost Model Rule:** Cost = 1 (for instruction) + 1 (for each memory reference).

- Registers and Immediate values have 0 memory cost.

Instruction	Type	Cost Calculation	Total Cost
MOV R0, b	Mem Read	1 (Inst) + 1 (Mem b)	2
MUL R0, c	Mem Read	1 (Inst) + 1 (Mem c)	2
MOV R1, a	Mem Read	1 (Inst) + 1 (Mem a)	2
ADD R1, R0	Reg only	1 (Inst) + 0	1
ADD R1, R1	Reg only	1 (Inst) + 0	1
CMP R1, 10	Immediate	1 (Inst) + 0	1
JGT L1	Control	1 (Inst) + 0	1
MOV R2, R1	Reg only	1 (Inst) + 0	1
MUL R2, 2	Immediate	1 (Inst) + 0	1
MOV t4, R2	Mem Write	1 (Inst) + 1 (Mem t4)	2
MOV R3, t4	Mem Read	1 (Inst) + 1 (Mem t4)	2
ADD R3, 1	Immediate	1 (Inst) + 0	1
MOV t5, R3	Mem Write	1 (Inst) + 1 (Mem t5)	2

Total Cost: 2 + 2 + 2 + 1 + 1 + 1 + 1 + 1 + 1 + 2 + 2 + 1 + 2 = **19**