Comsats University Islamabad, Attock Campus



BACHELOR OF COMPUTER SCIENCE

***Name:***

Muhammad Osman

***Registration no.***

FA21-BCS-029

***Course:***

Compiler Construction

***Submitted To:***

Sir Bilal Bukhari

Qno.3

Give example for optimization used in mini compiler?

**Optimization in a mini compiler improves performance by refining the code during compilation.**

**1. Constant Folding**

Constant folding evaluates constant expressions at compile time instead of runtime.

**Evaluates expressions at compile time:**

**Input:**

int result = 4 \* 5;

**Optimized:**

int result = 20;

**2. Dead Code Elimination**

Removes code that will never execute or doesn't affect the program's outcome.

**int x = 10; x = 20;**

**Optimized:**

int x = 20;

**3. Common Subexpression Elimination (CSE)**

Eliminates redundant calculations of the same expression.

**Example:**

for (int i = 0; i < n; i++) {

int temp = a + b;

arr[i] = temp \* i;

}

**Optimized:**

int temp = a + b;

for (int i = 0; i < n; i++) {

arr[i] = temp \* i;

}

**4. Loop-Invariant Code Motion**

Moves computations that don’t change within a loop outside the loop.

**Example:**

for (int i = 0; i < n; i++) {

int temp = x + y; // This doesn't change per iteration.

arr[i] = temp \* i;

}

**Optimized:**

int temp = x + y;

for (int i = 0; i < n; i++) {

arr[i] = temp \* i;

}

**5. Strength Reduction**

Replaces expensive operations with cheaper ones.

**Example:**

for (int i = 0; i < n; i++) {

int x = i \* 2;

}

**Optimized:**

for (int i = 0; i < n; i++) {

int x = i << 1; // Replace multiplication with a bit shift.

}

**6. Peephole Optimization**

Performs local optimizations in small instruction windows.

**Example (assembly):**

MOV R1, 10

ADD R1, 0

**Optimized (assembly):**

MOV R1, 10

**7. Inline Expansion**

Replaces a function call with the actual function body to eliminate the overhead of a call.

**Example:**

int square(int x) {

return x \* x;

}

int y = square(5);

**Optimized:**

int y = 5 \* 5;

**8. Register Allocation**

Assigns frequently used variables to CPU registers to avoid repeated memory accesses.

Example: Variables a, b, and c are accessed repeatedly, so they are assigned to registers instead of memory.

These techniques aim to reduce execution time, memory usage, or both, making the mini compiler's output more efficient.