Lecture 32

Code Optimizations

Awanish Pandey

Department of Computer Science and Engineering
Indian Institute of Technology
Roorkee

April 23, 2025

Code Optimization

- Criteria for code improving transformation

Code Optimization

- Criteria for code improving transformation
  - Preserve the meaning

Code Optimization

- Criteria for code improving transformation
  - Preserve the meaning
  - Must speed up the program

Code Optimization

- Criteria for code improving transformation
  - Preserve the meaning
  - Must speed up the program
  - Must be worth the effort

Code Optimization

- Criteria for code improving transformation
  - Preserve the meaning
  - Must speed up the program
  - Must be worth the effort
  - The analysis must be fast

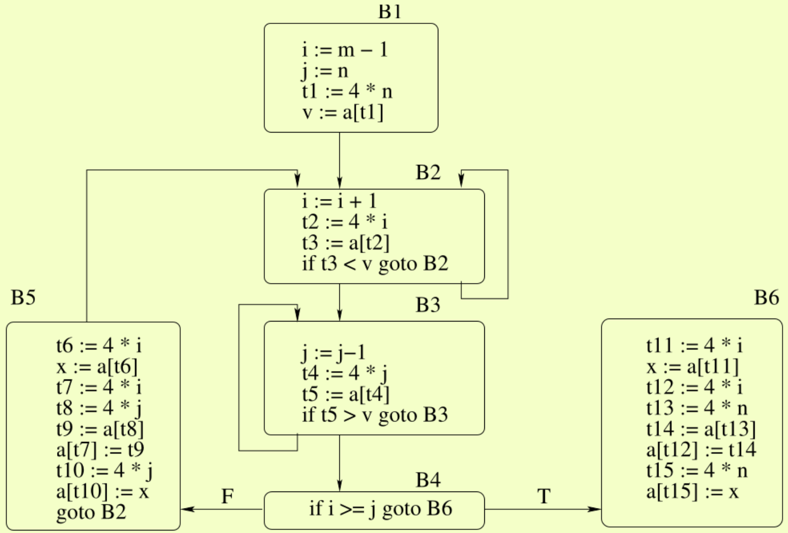
Code Optimization

- Criteria for code improving transformation
    - Preserve the meaning
    - Must speed up the program
    - Must be worth the effort
    - The analysis must be fast
- Local transformation: within basic blocks

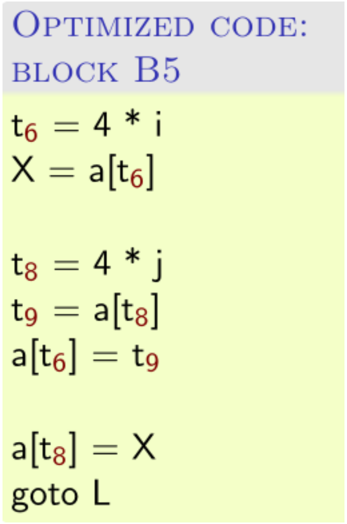
Code Optimization

- Criteria for code improving transformation
  - Preserve the meaning
  - Must speed up the program
  - Must be worth the effort
  - The analysis must be fast
- Local transformation: within basic blocks
- Global transformation: across basic blocks

Impact of Code Optimization



Common SubExpression Elimination



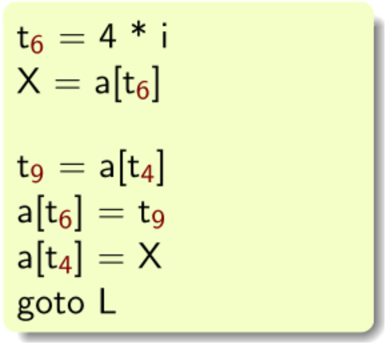
OPTIMIZED CODE: BLOCK B5

$t_6 = 4 * i$
$X = a[t_6]$

$t_8 = 4 * j$
$t_9 = a[t_8]$
$a[t_6] = t_9$

$a[t_8] = X$
goto L

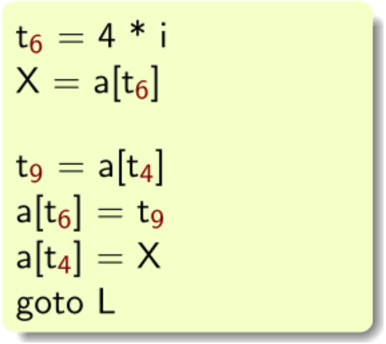
Global CSE



$t_6 = 4 * i$
$X = a[t_6]$

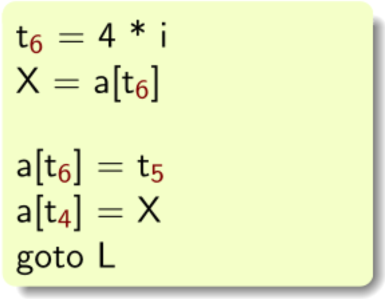
$t_9 = a[t_4]$
$a[t_6] = t_9$
$a[t_4] = X$
goto L

Global CSE



$t_6 = 4 * i$
$X = a[t_6]$

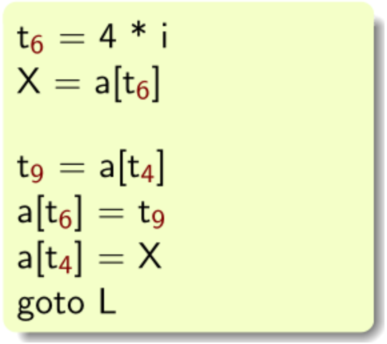
$t_9 = a[t_4]$
$a[t_6] = t_9$
$a[t_4] = X$
goto L



$t_6 = 4 * i$
$X = a[t_6]$

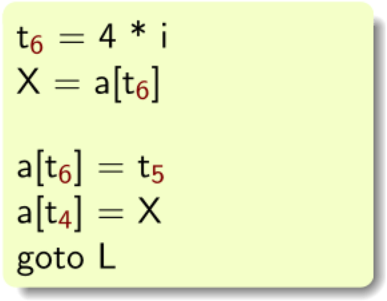
$a[t_6] = t_5$
$a[t_4] = X$
goto L

Global CSE



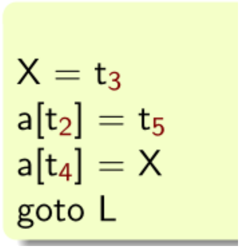
$t_6 = 4 * i$
$X = a[t_6]$

$t_9 = a[t_4]$
$a[t_6] = t_9$
$a[t_4] = X$
goto L



$t_6 = 4 * i$
$X = a[t_6]$

$a[t_6] = t_5$
$a[t_4] = X$
goto L



$X = t_3$
$a[t_2] = t_5$
$a[t_4] = X$
goto L

Example of Optimizations

- Common Sub Expression Elimination

Example of Optimizations

- Common Sub Expression Elimination
- Copy Propagation

Copy propagation works by identifying assignments where the source and destination variables have the same value. It then replaces all uses of the destination variable with the value of the source variable, effectively propagating the copy operation.

For example, consider the following code:

```
int a = 1 + 2;
int b = a;
int ans = b + 6;
```

After applying copy propagation, the code becomes:

```
int a = 1 + 2;
int ans = a + 6;
```

Example of Optimizations

- Common Sub Expression Elimination
- Copy Propagation
- Constant Folding
- Dead Code elimination

Example of Optimizations

- Common Sub Expression Elimination
- Copy Propagation
- Constant Folding
- Dead Code elimination
  - Code unreachable from any path

Example of Optimizations

- Common Sub Expression Elimination
- Copy Propagation
- Constant Folding
- Dead Code elimination
  - Code unreachable from any path
  - Code producing a value not used

Example of Optimizations

- Common Sub Expression Elimination
- Copy Propagation
- Constant Folding
- Dead Code elimination
  - Code unreachable from any path
  - Code producing a value not used
  - Decision statement whose true and false targets are same

Example of Optimizations

- Common Sub Expression Elimination
- Copy Propagation
- Constant Folding
- Dead Code elimination
  - Code unreachable from any path
  - Code producing a value not used
  - Decision statement whose true and false targets are same
  - Decision statement whose boolean expression can be computed at compile time

Example of Optimizations

- Common Sub Expression Elimination
- Copy Propagation
- Constant Folding
- Dead Code elimination
  - Code unreachable from any path
  - Code producing a value not used
  - Decision statement whose true and false targets are same
  - Decision statement whose boolean expression can be computed at compile time
- Code Motion

Example of Optimizations

- Common Sub Expression Elimination
- Copy Propagation
- Constant Folding
- Dead Code elimination
  - Code unreachable from any path
  - Code producing a value not used
  - Decision statement whose true and false targets are same
  - Decision statement whose boolean expression can be computed at compile time
- Code Motion
- Loop Unrolling

Example of Optimizations

- Common Sub Expression Elimination
- Copy Propagation
- Constant Folding
- Dead Code elimination
  - Code unreachable from any path
  - Code producing a value not used
  - Decision statement whose true and false targets are same
  - Decision statement whose boolean expression can be computed at compile time
- Code Motion
- Loop Unrolling
- Loop Jamming

Example of Optimizations

- Common Sub Expression Elimination
- Copy Propagation
- Constant Folding
- Dead Code elimination
  - Code unreachable from any path
  - Code producing a value not used
  - Decision statement whose true and false targets are same
  - Decision statement whose boolean expression can be computed at compile time
- Code Motion
- Loop Unrolling
- Loop Jamming
- Loop Unswitching

Example of Optimizations

Constant folding is an optimization technique in which the expressions are calculated beforehand to save execution time.

- Common Sub Expression Elimination
- Copy Propagation
- Constant Folding
- Dead Code elimination
  - Code unreachable from any path
  - Code producing a value not used
  - Decision statement whose true and false targets are same
  - Decision statement whose boolean expression can be computed at compile time
- Code Motion frequency reduction
- Loop Unrolling
- Loop Jamming Loop jamming is combining two or more loops in a single loop.
- Loop Unswitching It moves a conditional inside a loop outside of it by duplicating the loop's body, and placing a version of it inside each of the if and else clauses of the conditional. This can improve the parallelization of the loop. See wiki for example.
- Induction Variable Simplification simplify a bunch of code to a less size code.