Outline - Optimization Using Data-flow Analysis

Introduction and Preliminaries

Objectives

Three-Address Code and Control Flow Graph

Optimization Techniques (Independent from the Target Machine)

Principles of Data-flow Analysis

Elimination of Redundant Computations with Available Expressions

Elimination of Useless Instructions with Live Variables

Constant Propagation

Generalization

Summary

Two kinds of optimization techniques

Optimization independent from the target machine

- Objective: optimize the performance of the program.
- "source level" or "assembly level" pgm transformations.

Example (Optimization independent from the target machine)

- constant propagation, constant folding
- common sub-expressions elimination
- dead code elimination
- code motion

Optimization dependent from the target machine

- Objective: optimize the use of hardware resources.

Example (Optimization dependent from the target machine)

- machine instruction,
- memory hierarchy (registers, cache, pipeline, etc.).

Main principles of optimisation techniques

Input: initial intermediate code

Output: optimized intermediate code

Several steps:

1. generation of a control flow graph (CFG)
2. analysis of the CFG
3. transformation of the CFG
4. generation of the output code

Analysis and transformations

Analysis	**Transformation**
Available expressions common sub-expressions	Elimination of redundant computation
Live Variables	Elimination of useless code
Constant propagation	Replacing variables by their constant value
Induction Variable	Strength reduction
Loop Invariant	Moving the invariant code outside the loop
Dead-code elimination	Suppress useless instructions (which do not influence the execution)
Constant folding	Performing operations between constants
Copy propagation	Suppress useless variables (i.e., equal to another one or to a constant)
Algebraic simplification Strength reduction	Replace costly computations by less expensive ones