Local Optimizations - Part 1

Y.N. Srikant

Department of Computer Science
Indian Institute of Science
Bangalore 560 012

NPTEL Course on Compiler Design

Outline of the Lecture

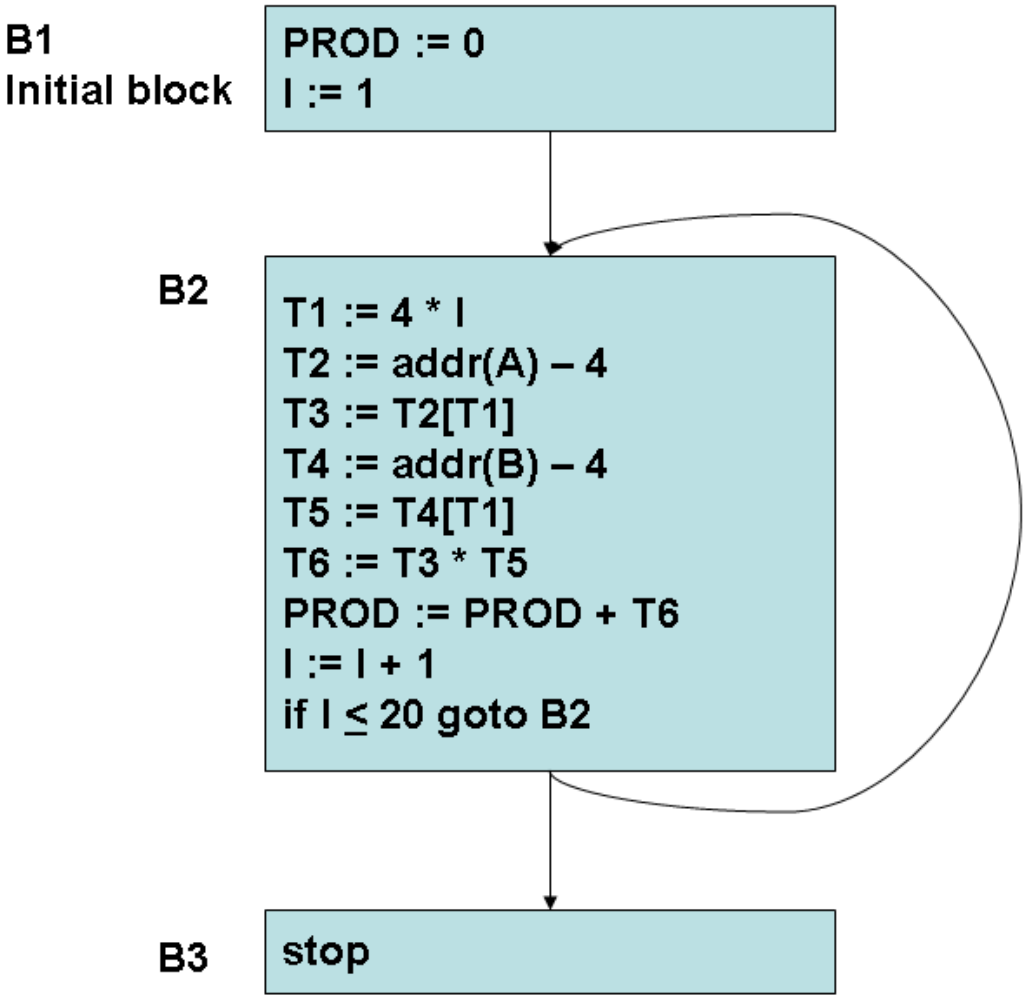
- Basic blocks and control flow graphs
- Local optimizations
- Building a control flow graph
- Directed acyclic graphs and value numbering

DAGs and value-numbering will be covered in part 2 of the lecture.

Basic Blocks and Flow Graphs

- Basic blocks are sequences of intermediate code with a *single entry* and a single exit
- We consider the quadruple version of intermediate code here, to make the explanations easier
- Flow graphs show control flow among basic blocks
- Basic blocks are represented as *directed acyclic blocks*(DAGs), which are in turn represented using the value-numbering method applied on quadruples
- Optimizations on basic blocks

Example of a Control Flow Graph



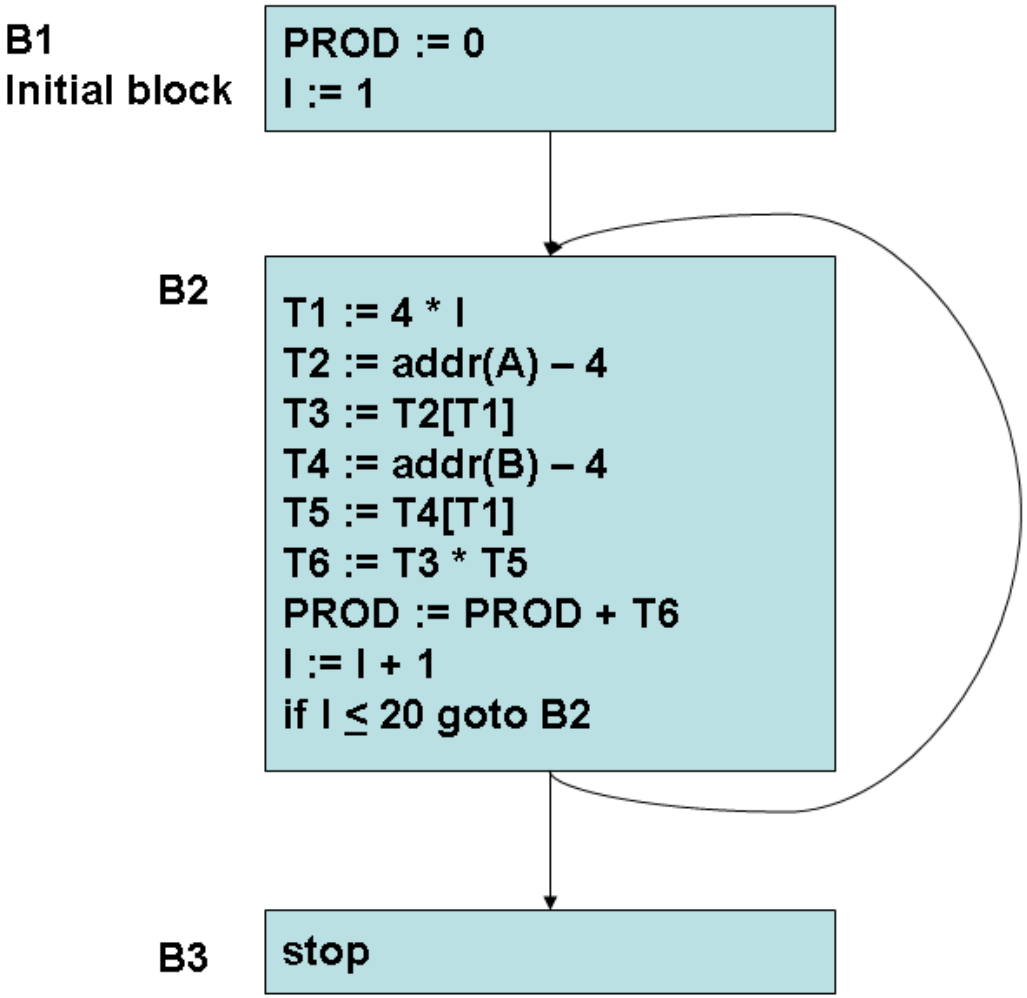
Local Optimizations

- Local common subexpression elimination
- Dead code (instructions that compute a value that is never used) elimination
- Reordering statements that do not depend on one another
- Reordering computations using algebraic laws
- The above two optimizations can be applied only on DAG or tree representation

Algorithm for Partitioning into Basic Blocks

1. Determine the set of *leaders*, the first statements of basic blocks
   - The first statement is a leader
   - Any statement which is the target of a conditional or unconditional *goto* is a leader
   - Any statement which immediately follows a *conditional goto* is a leader
2. A leader and all statements which follow it upto but not including the next leader (or the end of the procedure), is the basic block corresponding to that leader
3. Any statements, not placed in a block, can never be executed, and may now be removed, if desired

Example of a Control Flow Graph



Control Flow Graph

- The nodes of the CFG are basic blocks
- One node is distinguished as the initial node
- There is a directed edge $B1 \longrightarrow B2$, if B2 can immediately follow B1 in some execution sequence; i.e.,
  - There is a conditional or unconditional jump from the last statement of B1 to the first statement of B2, or
  - B2 immediately follows B1 in the order of the program, and B1 does not end in an unconditional jump

Basic Block Representation

A basic block is represented as a record consisting of

1. a count of the number of quadruples in the block
2. a pointer to the leader of the block
3. pointers to the predecessors of the block
4. pointers to the successors of the block

Note that jump statements point to basic blocks and not quadruples so as to make code movement easy

Example of a Control Flow Graph