Compiler Design (CD)
GTU # 2170701

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Unit – 7
Code Optimization

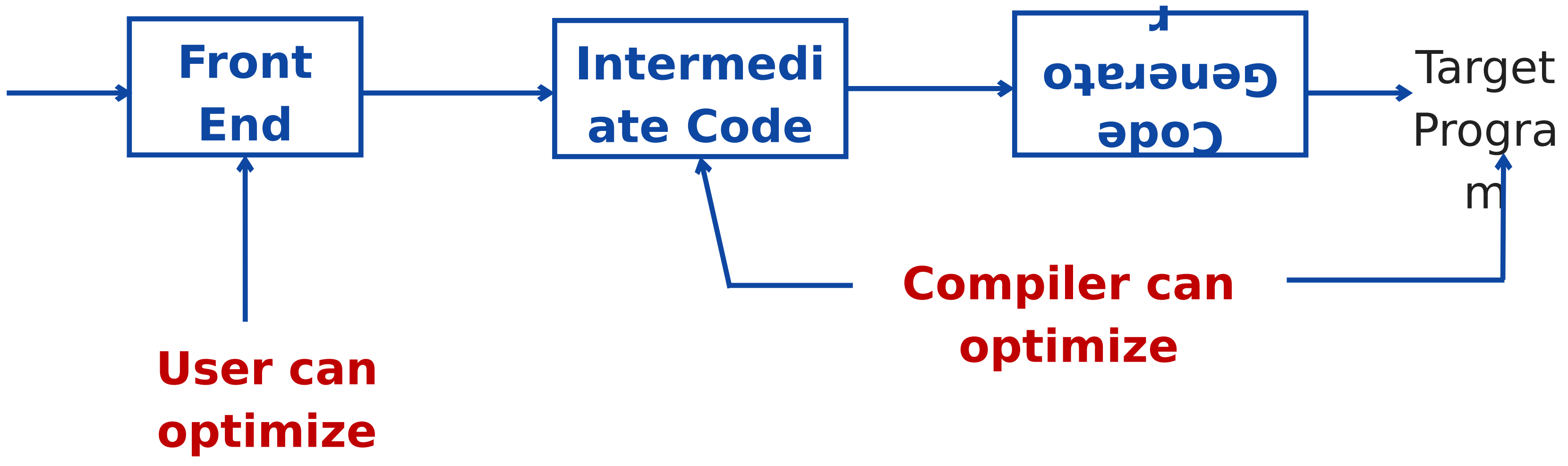
Prof. Dixita B. Kagathara
Computer Engineering
Department
rshan Institute of Engineering & Technology, Rajkot

dixita.kagathara@darshan.ac.in
+91 - 97277 47317 (CE
Department)

Topics to be covered

- Optimization technique
- Peephole optimization
- Loops in flow graph
- Global data flow analysis
- Dataflow properties

Code Optimization



Optimization Techniques

Dedicated Faculty, Committed Education

Darshan

Institute of Engineering & Technology

Compile time evaluation

- Compile time evaluation means shifting of computations from run time to compile time.
- There are two methods used to obtain the compile time evaluation.

Folding

- In the folding technique the computation of constant is done at compile time instead of run time.

    Example : length = (22/7)*d

- Here folding is implied by performing the computation of 22/7 at compile time.

Constant propagation

- In this technique the value of variable is replaced and computation of an expression is done at compilation time.

    Example : pi = 3.14; r = 5;

    Area = pi * r * r;

- Here at the compilation time the value of pi is replaced by 3.14 and r by 5

Common sub expressions elimination

- The common sub expression is an expression appearing repeatedly in the program which is computed previously.
- If the operands of this sub expression do not get changed at all then result of such sub expression is used instead of re-computing it each time.
- Example:

t1 := 4 * i
t2 := a[t1]
t3 := 4 * j
~~t4 : = 4 * i~~
t5:= n
t6 := b[t4]+t5

Copy Propagation

- Copy propagation means use of one variable instead of another.
- Example:

```
~~x = pi;~~
area = x * r * r;
```

$\downarrow$

```
area = pi * r * r;
```

Code Motion

- Optimization can be obtained by moving some amount of code outside the loop and placing it just before entering in the loop.
- This method is also called loop invariant computation.
- Example:

```
While(i<=ma
x-1)
{

sum=sum+a[
i];
}
```

$\longrightarrow$

```
;
}
```

Reduction in Strength

- priority of certain operators is higher than others.
- For instance strength of * is higher than +.
- In this technique the higher strength operators can be replaced by lower strength operators.
- Example:

```
for(i=1;i<=50;i++)
{
        count = i*7;  →
}                                                  ;



                             }
```

- Here we get the count values as 7, 14, 21…. and so on.

Dead code elimination

- The variable is said to be dead at a point in a program if the value contained into it is never been used.
- The code containing such a variable supposed to be a dead code.
- Example:

```
i=0;
if(i==1)
{
        a=x+5;
}
```

Dead Code

- If statement is a dead code as this condition will never get satisfied hence, statement can be eliminated and optimization can be done.

Peephole Optimization

Peephole optimization

- Peephole optimization is a simple and effective technique for locally improving target code.
- This technique is applied to improve the performance of the target program by examining the short sequence of target instructions (called the peephole) and replacing these instructions by shorter or faster sequence whenever possible.
- Peephole is a small, moving window on the target program.

Redundant Loads & Stores

- Especially the redundant loads and stores can be eliminated in following type of transformations.
- Example:

```
MOV R0,x
MOV x,R0
```

- We can eliminate the second instruction since x is in already R0.

Flow of Control Optimization

- The unnecessary jumps can be eliminated in either the intermediate code or the target code by the following types of peephole optimizations.
- We can replace the jump sequence.

Goto L1
...... $\rightarrow$
L1: goto L2

- It may be possible to eliminate the statement L1: goto L2 provided it is preceded by an unconditional jump. Similarly, the sequence can be replaced by:

If a<b goto L1
...... $\rightarrow$ *2*
L1: goto L2

Algebraic simplification

- Peephole optimization is an effective technique for algebraic simplification.
- The statements such as x = x + 0 or x := x* 1 can be eliminated by peephole optimization.

Reduction in strength

- Certain machine instructions are cheaper than the other.
- In order to improve performance of the intermediate code we can replace these instructions by equivalent cheaper instruction.
- For example, x^2 is cheaper than $x * x$.
- Similarly addition and subtraction are cheaper than multiplication and division. So we can add effectively equivalent addition and subtraction for multiplication and division.

Machine idioms

- The target instructions have equivalent machine instructions for performing some operations.
- Hence we can replace these target instructions by equivalent machine instructions in order to improve the efficiency.
- Example: Some machines have auto-increment or auto-decrement addressing modes.
- These modes can be used in code for statement like i=i+1.

Loops in Flow Graphs

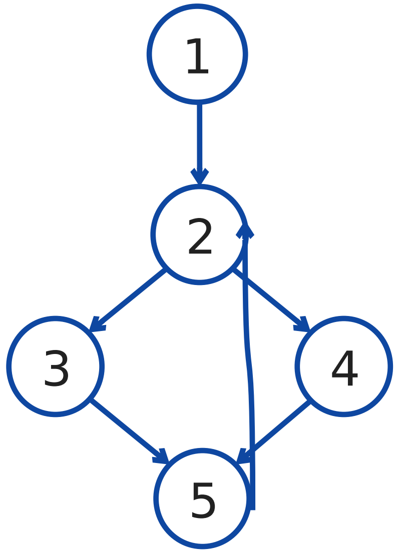
Dedicated Faculty, Committed Education

Darshan

Institute of Engineering & Technology

Dominators

- In a flow graph, a node d dominates n if every path to node n from initial node goes through d only.
- This can be denoted as ’d dom n'.
- Every initial node dominates all the remaining nodes in the flow graph.
- Every node dominates itself.

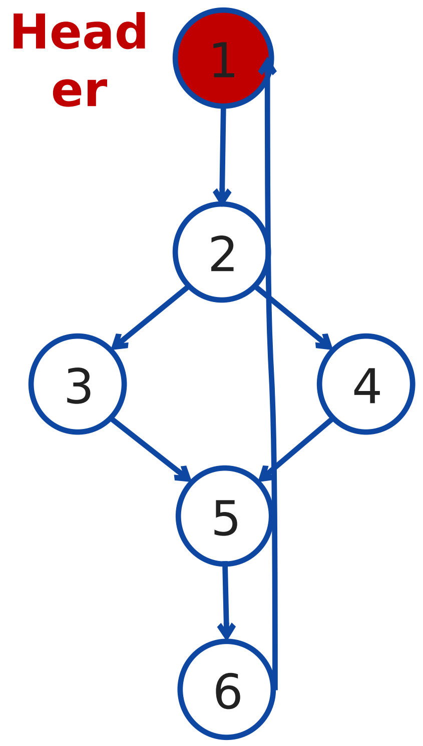


- Node 1 is initial node and it dominates every node as it is initial node.
- Node 2 dominates 3, 4 and 5.

Natural Loops

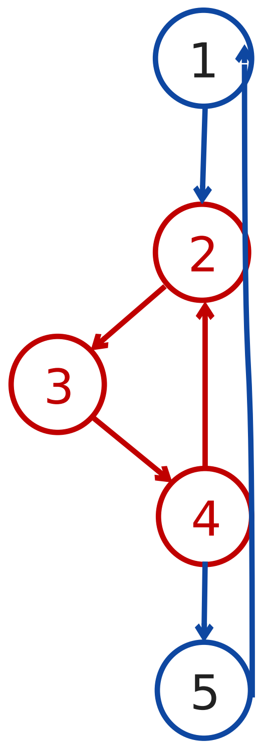
- There are two essential properties of natural loop:
  1. A loop must have single entry point, called the header. This point dominates all nodes in the loop.
  2. There must be at least one way to iterate loop.

  6→1 is natural loop.



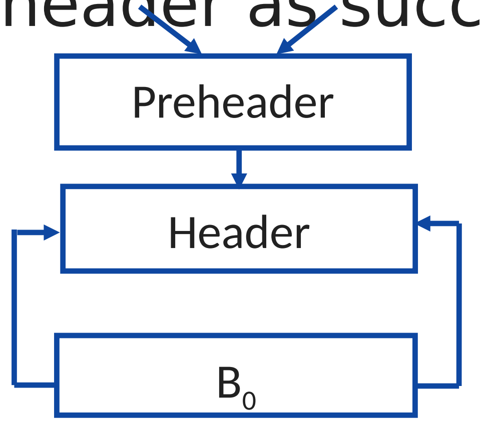
Inner Loops

- The inner loop is a loop that contains no other loop.
- Here the inner loop is 4→2 that mean edge given by 2-3-4.



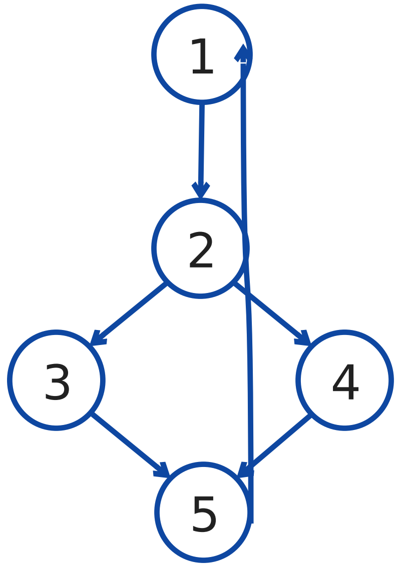
Preheader

- Several transformation require us to move statements "before the header".
- Therefore begin treatment of a loop by creating a new block, called the preheader.
- The preheader has only the header as successor.



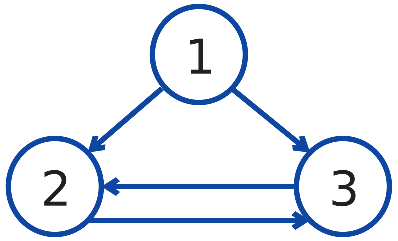
Reducible Flow Graph

- The reducible graph is a flow graph in which there are two types of edges forward edges and backward edges.
- These edges have following properties,
  1. The forward edge forms an acyclic graph.
  2. The back edges are such edges whose head dominates their tail.



Nonreducible Flow Graph

- A non reducible flow graph is a flow graph in which:
  1. There are no back edges.
  2. Forward edges may produce cycle in the graph.



Global Data Flow Analysis

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Global Data Flow Analysis

- Data flow equations are the equations representing the expressions that are appearing in the flow graph.
- Data flow information can be collected by setting up and solving systems of equations that relate information at various points in a program.
- The data flow equation written in a form of equation such that,

- Data flow equation can be read as “the information at the end of a statement is either generated within a statement, or enters at the beginning and is not killed as control flows through the statement”.

Global Data Flow Analysis

- The details of how dataflow equations are set up and solved depend on three factors.

1. The notions of generating, i.e., on the data flow analysis problem to be solved. Moreover, for some problems, instead of proceeding along with flow of control and defining out[s] in terms of in[s], we need to proceed backwards and define in[s] in terms of out[s].
2. g and killing depend on the desired information Since data flows along control paths, data-flow analysis is affected by the constructs in a program. In fact, when we write out[s] we implicitly assume that there is unique end point where control leaves the statement; in general, equations are set up at the level of basic blocks rather than statements, because blocks do have unique end points.
3. There are subtleties that go along with such statements as procedure calls, assignments through pointer variables, and even assignments to array variables.

Dataflow Properties

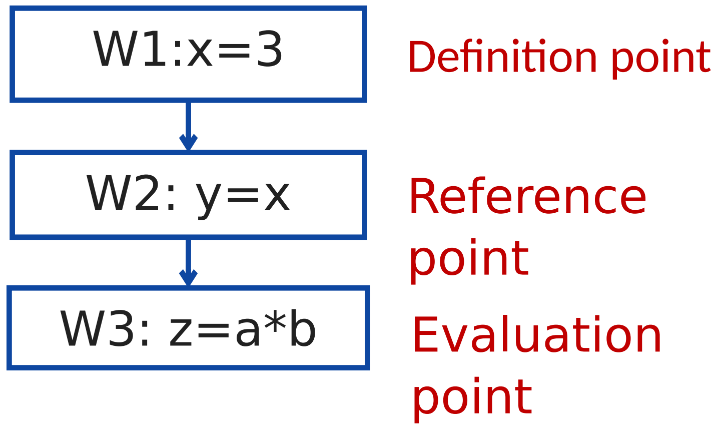
Dedicated Faculty, Committed Education

Darshan

Institute of Engineering & Technology

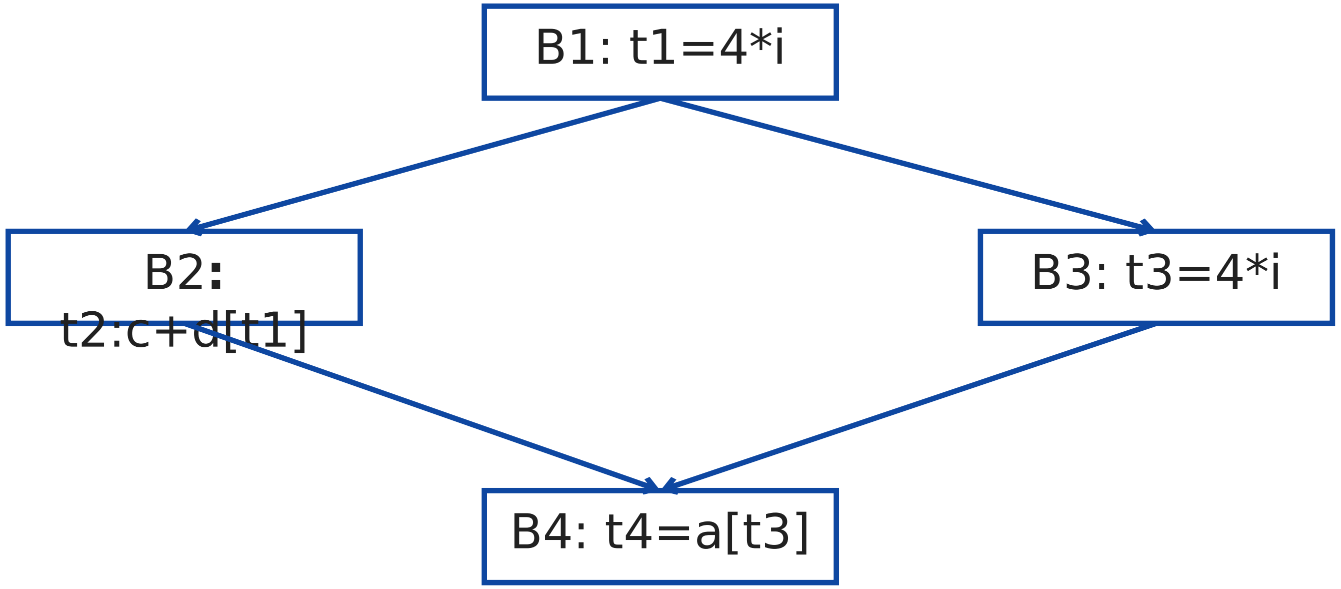
Data Flow Properties

- A program point containing the definition is called Definition point.
- A program point at which a reference to a data item is made is called Reference point.
- A program point at which some evaluating expression is given is called Evaluation point.



Available Expression

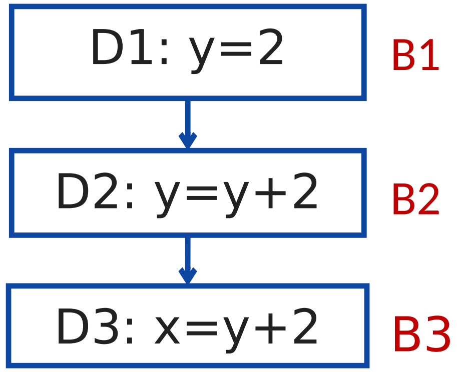
- An expression is available at a program point w if and only if along all paths are reaching to w.
  1. The expression is said to be available at its evaluation point.
  2. Neither of the two operands get modified before their use.



- Expression is the available expression for , and because this expression has not been changed by any of the block before appearing in .

Reaching Definition

- A definition reaches at the point if there is a path from to along which is not killed.
- A definition of variable is killed when there is a redefinition of .



- The definition is reaching definition for block , but the definition is not reaching definition for block , because it is killed by definition in block .

Live variable

- A live variable is live at point if there is a path from to the exit, along which the value of is used before it is redefined.
- Otherwise the variable is said to be dead at the point.
- Example:

```
b = 3
c = 5
a = f(b * c)
```

- The set of live variables are {b, c} because both are used in the multiplication on line 3.

Busy Expression

- An expression is said to be busy expression along some path if and only if an evaluation of exists along some path and no definition of any operand exist before its evaluation along the path.

Thank You