5.3 BASIC BLOCKS AND FLOW GRAPHS

Basic Blocks and Flow Graphs are important concepts in Compiler Design. They are mainly used during code optimization and control-flow analysis.

A compiler divides a program into small groups of statements called Basic Blocks and represents the flow between these blocks using a Flow Graph.

1. Basic Idea

Source Program

Intermediate Code

Divide into Basic Blocks

Construct Flow Graph

Perform Analysis / Optimization

Optimized Code

2. Basic Block

Definition:

A Basic Block is a sequence of consecutive statements in which:

- Control enters at the beginning.
- Control leaves at the end.
- There is no branching inside the block.

In simple words:

A Basic Block is a group of statements that are executed sequentially without any jump or branch in the middle.

3. Example of Basic Block

Consider the following three-address code:

1. a = b + c

2. d = a - e

3. f = d * 2

4. if f > 10 goto 7

5. g = f + 1

6. goto 8

7. g = f - 1

8. h = g * 2

This code can be divided into four basic blocks.

Block B1:

a = b + c

d = a - e

f = d * 2

if f > 10 goto 7

Block B2:

g = f + 1

goto 8

Block B3:

g = f - 1

Block B4:

h = g * 2

4. Leaders

To identify Basic Blocks, the compiler first identifies Leaders.

A Leader is the first statement of a basic block.

Rules for Finding Leaders:

There are three important rules.

Rule 1: First statement is always a leader.

The first instruction of the program is a leader.

Example:

1. a = b + c ← Leader

Rule 2: Target of a jump is a leader.

If a statement is the target of a conditional or unconditional jump, it is a leader.

Example:

4. if f > 10 goto 7

Therefore:

7. g = f - 1 ← Leader

Rule 3: Statement immediately following a jump is a leader.

The statement immediately after a conditional or unconditional jump is also a leader.

Example:

4. if f > 10 goto 7

5. g = f + 1 ← Leader

Also:

6. goto 8

7. g = f - 1 ← Leader

5. Algorithm for Constructing Basic Blocks

Given a sequence of three-address instructions:

Step 1: Identify all leaders.

Step 2: A leader and all statements following it up to, but not including, the next leader form a basic block.

Step 3: Continue until all statements are included in a block.

Example:

Consider:

1. a = b + c

2. d = a - e

3. if d > 10 goto 6

4. f = d + 1

5. goto 7

6. f = d - 1

7. g = f * 2

Identify Leaders:

- Statement 1 → First statement → Leader.
- Statement 4 → Follows conditional jump → Leader.

- Statement 6 → Target of jump → Leader.
- Statement 7 → Target of goto 7 → Leader.

Therefore:

Leaders = 1, 4, 6, 7

Basic Blocks:

B1

1. a = b + c

2. d = a - e

3. if d > 10 goto 6

B2

4. f = d + 1

5. goto 7

B3

6. f = d - 1

B4

7. g = f * 2

6. Flow Graph Definition

A Flow Graph is a directed graph that represents the flow of control between basic blocks.

It consists of:

- Nodes: Represent basic blocks.
- Edges: Represent the flow of control.

General Structure:

B1

B2

B3

B4

Here:

- B1, B2, B3, B4 are basic blocks.

- Arrows represent possible control flow.

7. Flow Graph for the Example

For the previous example:

1. a = b + c

2. d = a - e

3. if d > 10 goto 6

4. f = d + 1

5. goto 7

6. f = d - 1

7. g = f * 2

The flow graph is:

B1

1, 2, 3

B2

4, 5

B3

6

B4

7

8. Components of a Flow Graph

1. Nodes

Each node represents a basic block.

Example: B1, B2, B3, B4.

2. Edges

An edge represents a possible transfer of control.

For example:

- B1 → B2
- B1 → B3

- B2 → B4
- B3 → B4

3. Entry Node

The first basic block is called the Entry Block.

In the example, B1 is the entry block.

4. Exit Node

The block from which program control leaves the flow graph represents the Exit Block.

9. Types of Control Flow

There are mainly three common types.

1. Sequential Flow

Control moves from one block directly to the next.

Example:

B1 → B2

2. Conditional Flow

Control can move to one of two blocks depending on a condition.

Example:

B1

B2

B3

This occurs with:

- if
- if-else
- while
- for

3. Unconditional Flow

Control always moves to one specific block.

Example:

B1 → B2

This occurs with goto statements.

10. Basic Blocks vs Flow Graphs

Basic Block	Flow Graph
Group of consecutive statements	Graph representing control flow
Contains instructions	Contains basic blocks as nodes
No branching inside	Shows branching between blocks
Used for local analysis	Used for control-flow analysis
Forms a unit of optimization	Shows relationships between blocks

11. Importance of Basic Blocks

Basic Blocks are useful because they:

1. Simplify program analysis.
2. Help perform local optimization.
3. Make control flow easier to understand.
4. Help identify unreachable code.
5. Help identify loops.
6. Support data-flow analysis.
7. Help in efficient code generation.

12. Importance of Flow Graphs

Flow Graphs are useful for:

- Code optimization.
- Dead code elimination.
- Loop detection.
- Data-flow analysis.
- Control-flow analysis.
- Register allocation.
- Common subexpression analysis.
- Finding unreachable blocks.

Example:

If a block has no path from the entry block, the compiler can identify it as unreachable code.

13. Basic Blocks and Flow Graphs in Optimization

The relationship can be represented as:

Intermediate Code

Identify Leaders

Form Basic Blocks

Construct Flow Graph

Analyze Control Flow

Apply Optimization

Optimized Code

14. Important Exam Points

Basic Block:

A basic block is a sequence of consecutive statements with a single entry point and a single exit point, where control enters only at the beginning and leaves only at the end.

Leader:

The three rules for identifying leaders are:

1. The first statement is a leader.
2. The target of a jump is a leader.
3. The statement immediately following a jump is a leader.

Flow Graph:

A flow graph is a directed graph in which nodes represent basic blocks and edges represent possible flow of control between the blocks.