SNS COLLEGE OF TECHNOLOGY

(Autonomous)

COIMBATORE-35

Basic blocks and Flow graphs & Next Use Information

(1) prod := 0
(2) i := 1
(3) t_1 := 4* i
(4) t_2:= a[t_1]
(5) t_3 := 4* i
(6) t_4 := b[t_3]
(7) t_5 := t_2 * t_4
(8) t_6 := prod + t_5
(9) prod := t_6
(10) t_7 := i +1
(11) i := t_7
(12) if i <= 20 goto (3)

B_1:
(1)prod := 0
(2) i := 1

B_2:
(3) t_1 := 4* i
(4) t_2:= a[t_1]
(5) t_3 := 4* I
(6) t_4 := b[t_3]
(7) t_5 := t_2 * t_4
(8) t_6 := prod + t_5
(9) prod := t_6
(10) t_7 := i +1
(11) i := t_7
(12) if i <= 20 goto (3)

Basic blocks and Flow graphs

Basic Block:

A basic block is a sequence of consecutive statements in which flow of control enters at the beginning and leaves at the end without halt or possibly of the branching except at the end.

- **Flow Graph:** A graph representation of three address statements, called flow graph.
- Nodes in the flow graph represent computations.
- Edges represent the flow of control.
- Used to do better job of register allocation and instruction selection.

Basic blocks and Flow graphs

Basic Blocks

- A *basic block* is a sequence of consecutive instructions with exactly one entry point and one exit point (with natural flow or a branch instruction)

```
     MOV 1,R0
     MOV n,R1
     JMP L2
L1:  MUL 2,R0
     SUB 1,R1
L2:  JMPNZ R1,L1
```

```
     MOV 1,R0
     MOV n,R1
     JMP L2
```

```
L1:  MUL 2,R0
     SUB 1,R1
```

```
L2:  JMPNZ R1,L1
```

Basic blocks and Flow graphs

2

Flow Graphs

- A *flow graph* is a graphical depiction of a sequence of instructions with control flow edges
- A flow graph can be defined at the intermediate code level or target code level

```
    MOV 1,R0
    MOV n,R1
    JMP L2
L1: MUL 2,R0
    SUB 1,R1
L2: JMPNZ R1,L1
```

```
    MOV 0,R0
    MOV n,R1
    JMP L2
L1: MUL 2,R0
    SUB 1,R1
L2: JMPNZ R1,L1
```

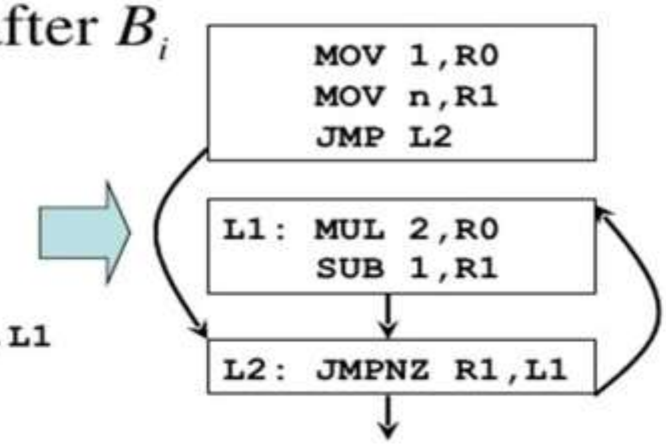
Basic blocks and Flow graphs

4

Basic Blocks and Control Flow Graphs

- A *control flow graph* (CFG) is a directed graph with basic blocks B_i as vertices and with edges $B_i \rightarrow B_j$ iff B_j can be executed immediately after B_i

```
    MOV 1,R0
    MOV n,R1
    JMP L2
L1: MUL 2,R0
    SUB 1,R1
L2: JMPNZ R1,L1
```

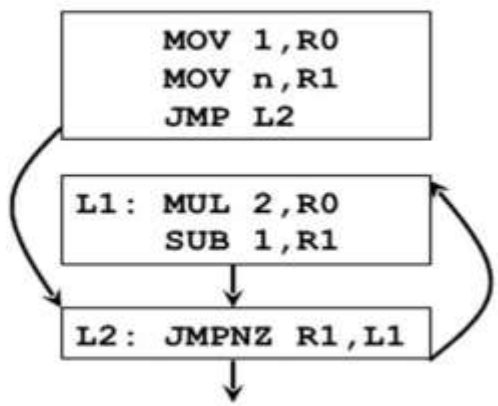


Basic blocks and Flow graphs

5

Successor and Predecessor Blocks

- Suppose the CFG has an edge $B_1 \rightarrow B_2$
  - Basic block B_1 is a *predecessor* of B_2
  - Basic block B_2 is a *successor* of B_1



Basic blocks and Flow graphs

7

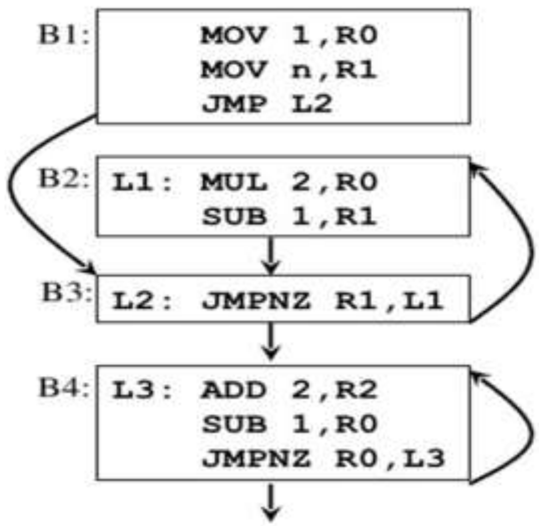
Loops

- A *loop* is a collection of basic blocks, such that
  - All blocks in the collection are *strongly connected*
  - The collection has a unique *entry*, and the only way to reach a block in the loop is through the entry

Basic blocks and Flow graphs

8

Loops (Example)



Strongly connected components:

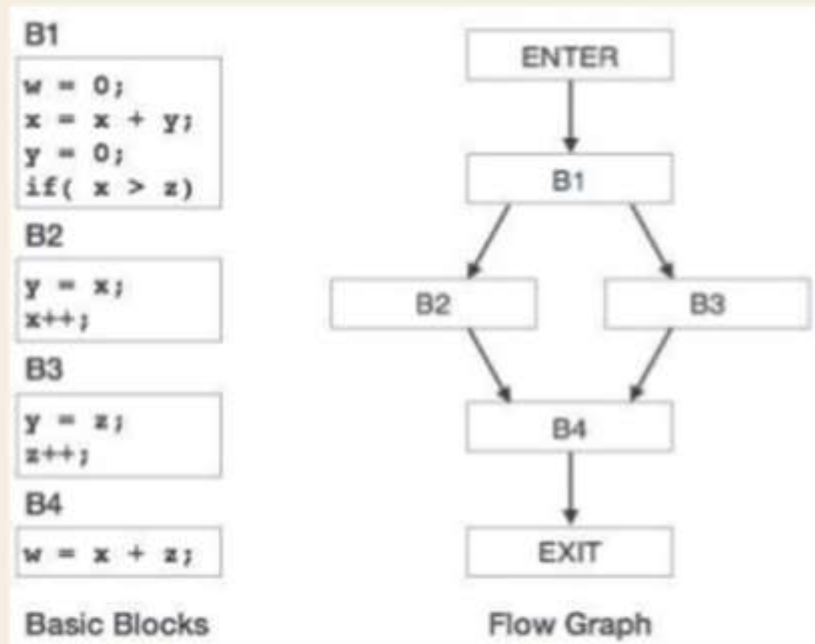
SCC={ {B2,B3}, {B4} }

Entries: B3, B4

Basic blocks and Flow graphs

Construction of the Representation

1. Partition the intermediate code into basic blocks
2. The basic blocks become the nodes of a flow graph and the edges indicate the *flow* (which blocks follow which)



Basic blocks and Flow graphs

Constructing Basic Blocks

Determine a set of leaders

1) The first instruction is a leader
2) Instruction L is a leader if there is an instruction `if ... goto L` or `goto L`
3) Instruction L is a leader if it immediately follows an instruction `if ... goto B` or `goto B`

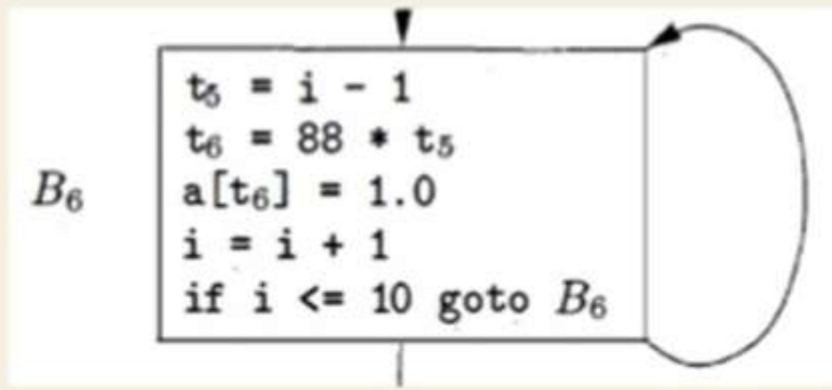
```
1)  i = 1
2)  j = 1
3)  t1 = 10 * i
4)  t2 = t1 + j
5)  t3 = 8 * t2
6)  t4 = t3 - 88
7)  a[t4] = 0.0
8)  j = j + 1
9)  if j <= 10 goto (3)
10) i = i + 1
11) if i <= 10 goto (2)
12) i = 1
13) t5 = i - 1
14) t6 = 88 * t5
15) a[t6] = 1.0
16) i = i + 1
17) if i <= 10 goto (13)
```

Basic blocks and Flow graphs

Constructing the Flow Graph

There is an edge from block *B* to block *C* iff it is possible for the first instruction in block *C* to immediately follow the last instruction in block *B*

1. There is a conditional or unconditional jump from the end of B to the beginning of C

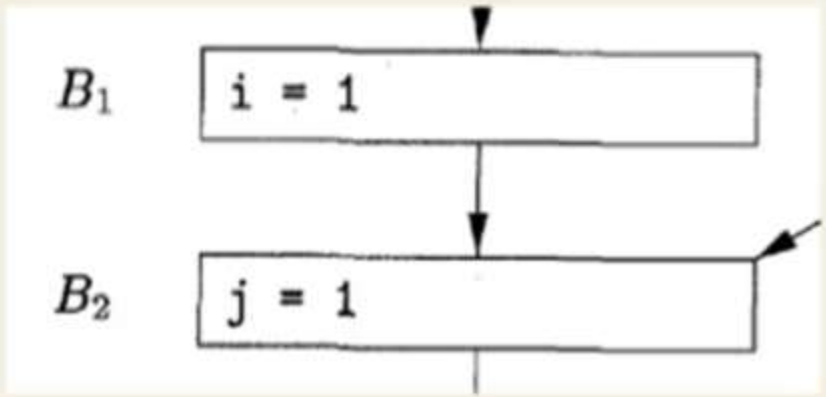


Basic blocks and Flow graphs

Constructing the Flow Graph

There is an edge from block *B* to block *C* iff it is possible for the first instruction in block *C* to immediately follow the last instruction in block *B*

2. C immediately follows B in the original order of the three-address instructions, and B does not end in an unconditional jump

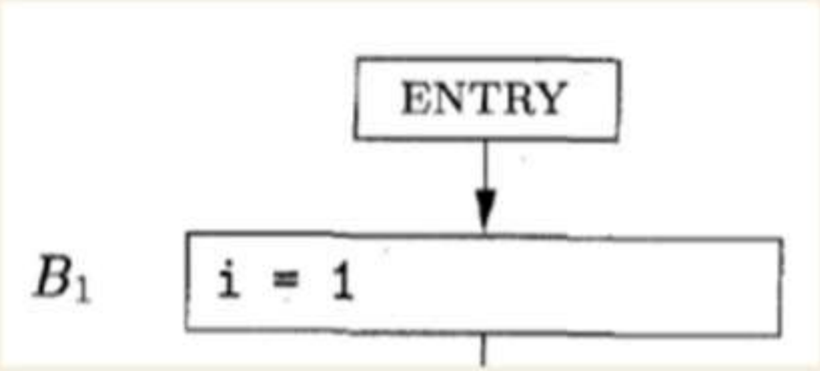


Basic blocks and Flow graphs

Constructing the Flow Graph

Often we add two nodes, called the entry and exit.

1. There is an edge from the entry to the first executable node.

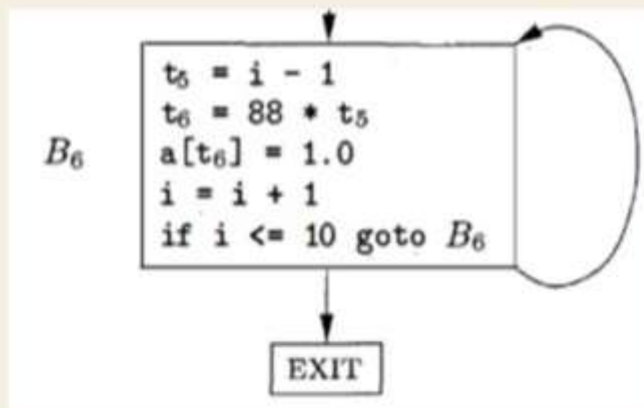


Basic blocks and Flow graphs

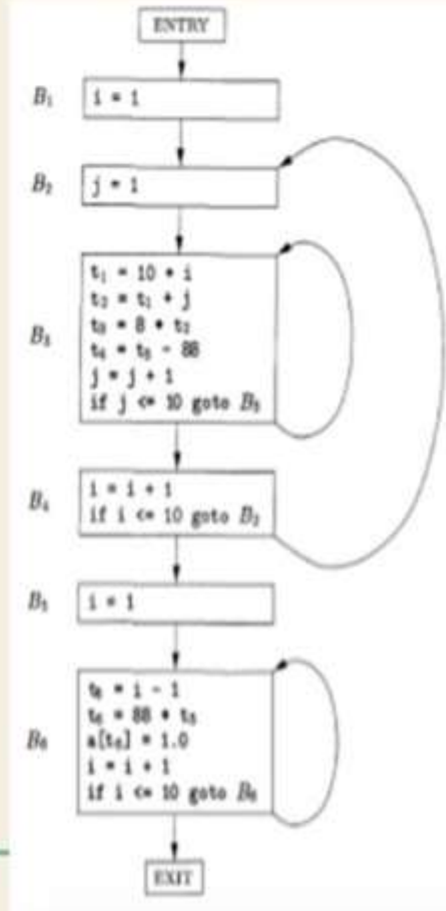
Constructing the Flow Graph

Often we add two nodes, called the entry and exit.

2. There is an edge to the exit from any basic block that contains an instruction that could bet eh last executed instruction of the program



```
1)  i = 1
2)  j = 1
3)  t1 = 10 * i
4)  t2 = t1 + j
5)  t3 = 8 * t2
6)  t4 = t3 - 88
7)  a[t4] = 0.0
8)  j = j + 1
9)  if j <= 10 goto (3)
10) i = i + 1
11) if i <= 10 goto (2)
12) i = 1
13) t5 = i - 1
14) t6 = 88 * t5
15) a[t6] = 1.0
16) i = i + 1
17) if i <= 10 goto (13)
```



Summarization