

Intermediate Code Generation

Basic Blocks

Copyright 2022, Pedro C. Diniz, all rights reserved.

Students enrolled in the Compilers class at the University of Porto have explicit permission to make copies of these materials for their personal use.

Basic Blocks

- How to Split Code into Basic Blocks?
 - Identify “head” instructions, or *leaders* of the basic blocks
 - Jumps (the instruction after) and Targets of jump are possible points of control-flow divergence
 - “Move forward” to next “header” instruction
 - Repeat until all headers and subsequent sequential instructions are identified

Basic Blocks

- Maximal sequence of instructions with the property:

If first instruction is executed, then all subsequent instructions are also executed

- Why is this Important?
 - Basis of Program Reasoning – a single control-flow thread.
 - Locus of Opportunity for Analysis and Transformations.
- How to Identify the Basic Blocks?

Algorithm for Identifying Basic Blocks

- Idea: Identify Leader or Header of Basic Blocks
 - The instruction that initiates the execution of a basic block
 - Where control-flow may diverge or converge define end of a basic block
 - Block: all instructions between leader and the next leader (excluding it)
- Algorithm:
 - The first instruction in the code is a leader;
 - Any instruction target of a jump (conditional or unconditional) is a leader;
 - Any instruction that immediately follows a jump is a leader;
 - Repeatedly scans instructions until no additional leaders can be identified;
 - Scan instructions from each leader to the next to identify basic blocks;

Example

```

01:  i = 1
02:  j = 1
03:  t1 = 10 * i
04:  t2 = t1 + j
05:  t3 = 8 * 12
06:  t4 = t3 - 8
07:  M[t4] = 0
08:  j = j + 1
09:  if j <= 10 goto 03
10:  i = i + 1
11:  if j <= 10 goto 02
12:  i = 1
13:  t5 = i - 1
14:  t6 = 8 * t5
15:  M[t6] = 1
16:  i = i + 1
17:  if i <= 10 goto 13
18:  j = 1
19:  i = j + 2
20:  return i
    
```

Example

```

01: i = 1
02: j = 1
03: t1 = 10 * i
04: t2 = t1 + j
05: t3 = 8 * 12
06: t4 = t3 - 8
07: M[t4] = 0
08: j = j + 1
09: if j <= 10 goto 03
10: i = i + 1
11: if j <= 10 goto 02
12: i = 1
13: t5 = i - 1
14: t6 = 8 * t5
15: M[t6] = 1
16: i = i + 1
17: if i <= 10 goto 13
18: j = 1
19: i = j + 2
20: return i
    
```

Algorithm:

- The first instruction in the code is a leader;
- Any instruction target of a jump (conditional or unconditional) is a leader;
- Any instruction that immediately follows a jump is a leader;
- Repeatedly scans instructions until no additional leaders can be identified;
- Scan instructions from each leader to the next to identify basic blocks;

Example

```

01: i = 1
02: j = 1
03: t1 = 10 * i
04: t2 = t1 + j
05: t3 = 8 * 12
06: t4 = t3 - 8
07: M[t4] = 0
08: j = j + 1
09: if j <= 10 goto 03
10: i = i + 1
11: if j <= 10 goto 02
12: i = 1
13: t5 = i - 1
14: t6 = 8 * t5
15: M[t6] = 1
16: i = i + 1
17: if i <= 10 goto 13
18: j = 1
19: i = j + 2
20: return i

```

Algorithm:

- The first instruction in the code is a leader;
- Any instruction target of a jump (conditional or unconditional) is a leader;
- Any instruction that immediately follows a jump is a leader;
- Repeatedly scans instructions until no additional leaders can be identified;
- Scan instructions from each leader to the next to identify basic blocks;

Example

```

01: i = 1
02: j = 1
03: t1 = 10 * i
04: t2 = t1 + j
05: t3 = 8 * 12
06: t4 = t3 - 8
07: M[t4] = 0
08: j = j + 1
09: if j <= 10 goto 03
10: i = i + 1
11: if j <= 10 goto 02
12: i = 1
13: t5 = i - 1
14: t6 = 8 * t5
15: M[t6] = 1
16: i = i + 1
17: if i <= 10 goto 13
18: j = 1
19: i = j + 2
20: return i
    
```

Algorithm:

- The first instruction in the code is a leader;
- Any instruction target of a jump (conditional or unconditional) is a leader;
- Any instruction that immediately follows a jump is a leader;
- Repeatedly scans instructions until no additional leaders can be identified;
- Scan instructions from each leader to the next to identify basic blocks;

Example

```

01: i = 1
02: j = 1
03: t1 = 10 * i
04: t2 = t1 + j
05: t3 = 8 * 12
06: t4 = t3 - 8
07: M[t4] = 0
08: j = j + 1
09: if j <= 10 goto 03
10: i = i + 1
11: if j <= 10 goto 02
12: i = 1
13: t5 = i - 1
14: t6 = 8 * t5
15: M[t6] = 1
16: i = i + 1
17: if i <= 10 goto 13
18: j = 1
19: i = j + 2
20: return i

```

Algorithm:

- The first instruction in the code is a leader;
- Any instruction target of a jump (conditional or unconditional) is a leader;
- Any instruction that immediately follows a jump is a leader;
- Repeatedly scans instructions until no additional leaders can be identified;
- Scan instructions from each leader to the next to identify basic blocks;

Example

01:	i = 1	BB1
02:	j = 1	BB2
03:	t1 = 10 * i	BB3
04:	t2 = t1 + j	
05:	t3 = 8 * 12	
06:	t4 = t3 - 8	
07:	M[t4] = 0	
08:	j = j + 1	
09:	if j <= 10 goto 03	
10:	i = i + 1	BB4
11:	if j <= 10 goto 02	
12:	i = 1	BB5
13:	t5 = i - 1	BB6
14:	t6 = 8 * t5	
15:	M[t6] = 1	
16:	i = i + 1	
17:	if i <= 10 goto 13	
18:	j = 1	BB7
19:	i = j + 2	
20:	return i	

Algorithm:

- The first instruction in the code is a leader;
- Any instruction target of a jump (conditional or unconditional) is a leader;
- Any instruction that immediately follows a jump is a leader;
- Repeatedly scans instructions until no additional leaders can be identified;
- Scan instructions from each leader to the next to identify basic blocks;

Example

- Why is this Important?
 - Basis of Program Reasoning – single control-flow thread.

Understanding which BBs reach which other BBs

- Locus of Opportunity for Analysis and Transformations.

Data-Flow is “easy” to understand. Why?

Control-Flow and data-Flow Analysis Later

