CS323: Compilers
Spring 2023

Week 11: Instruction Scheduling (contd..), Control Flow Graphs

Acknowledgements: Milind Kulkarni

List scheduling - Example

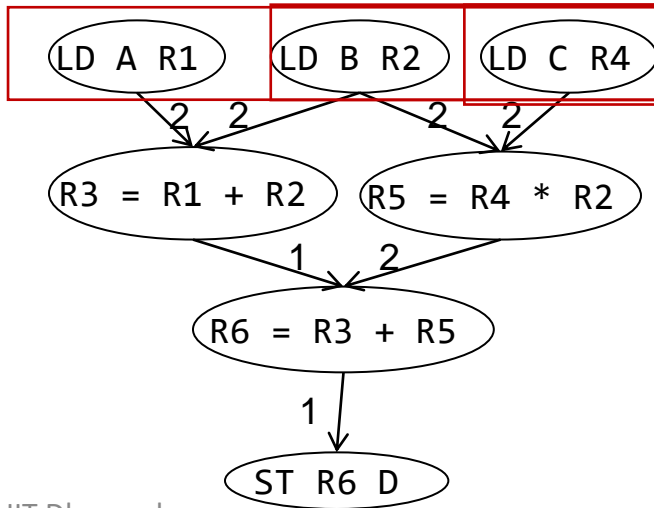|	Cycle #	Available Instruction(s)	Scheduled Instruction(s)	Completed Instruction(s)
	0	1, 2, 4	1*	
	1	2, 4		
	2	2, 4	2*	1
	3	4		
	4	3,4	3,4	2
	5			3
	6	5	5	4
	7			
	8	6	6	5
	9	7	7	6
	10			7

```
1.  LD  A,  R1
2.  LD  B,  R2
3.  R3  =  R1  +  R2
4.  LD  C,  R4
5.  R5  =  R4  *  R2
6.  R6  =  R3  +  R5
7.  ST  R6,  D
```

*an instruction from the list of available instructions is picked at random and scheduled

LD A R1   LD B R2   LD C R4
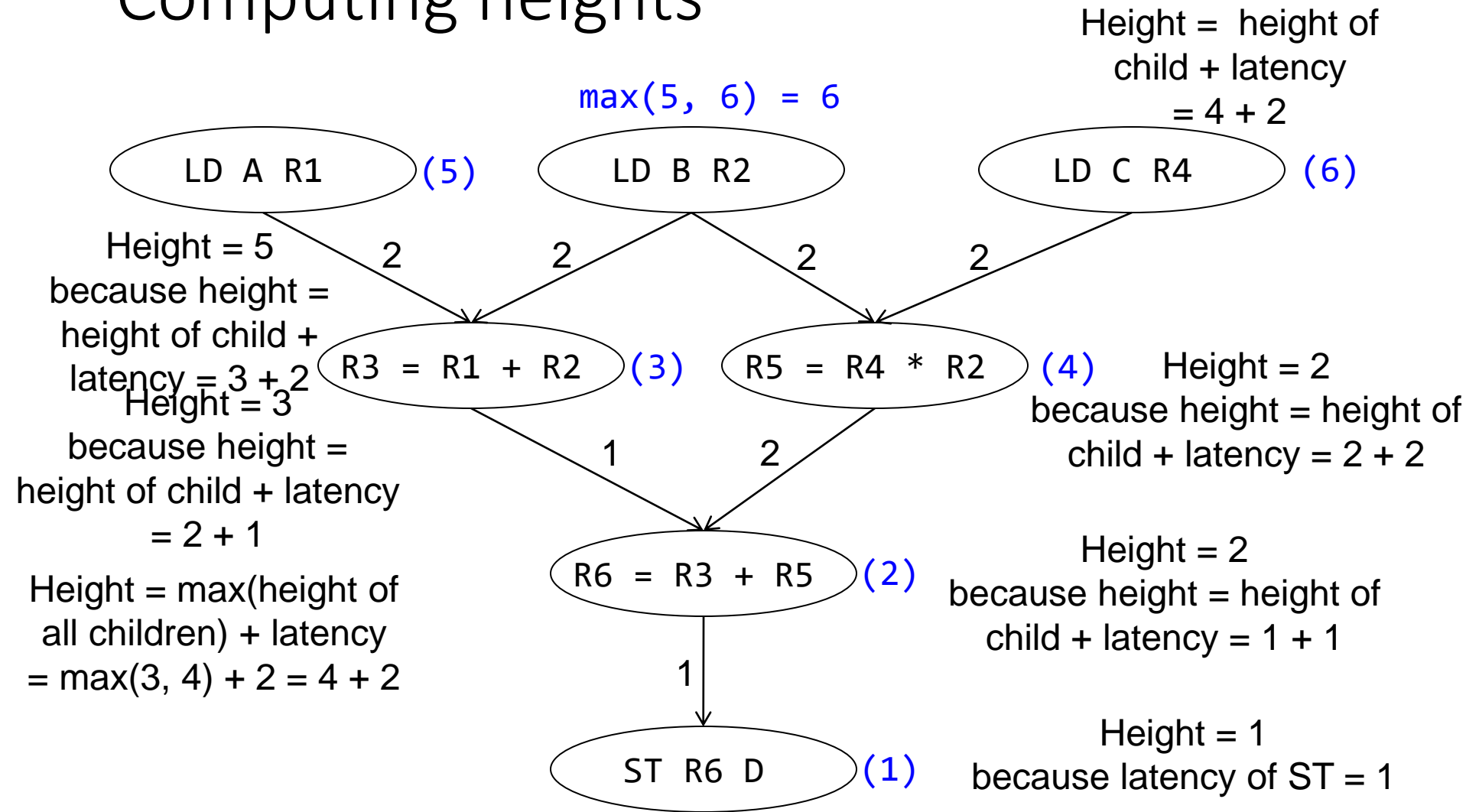
R3 = R1 + R2   R5 = R4 * R2

R6 = R3 + R5

ST R6 D

2

List scheduling

1. LD A, R1
2. LD B, R2
3. R3 = R1 + R2
4. LD C, R4
5. R5 = R4 * R2
6. R6 = R3 + R5
7. ST R6, D

Cycle	ALU0	ALU1	LD/ST
0			1
1			1
2			2
3			2
4	3		4
5			4
6	5		
7			
8	6		
9			7
10			

Height-based scheduling

- Important to prioritize instructions

    - Instructions that have a lot of downstream instructions dependent on them should be scheduled earlier

- Instruction scheduling NP-hard in general, but **height-based scheduling** is effective

- Instruction height = latency from instruction to farthest-away leaf

    - Leaf node height = instruction latency

    - Interior node height = max(heights of children + instruction latency)

- Schedule instructions with highest height first

Computing heights

max(5, 6) = 6

Height = height of child + latency = 4 + 2

LD A R1   (5)

LD B R2

LD C R4   (6)

Height = 5 because height = height of child + latency = 3 + 2

2   2   2   2

Height = 3 because height = height of child + latency = 2 + 1

R3 = R1 + R2   (3)

R5 = R4 * R2   (4)   Height = 2 because height = height of child + latency = 2 + 2

1   2

Height = max(height of all children) + latency = max(3, 4) + 2 = 4 + 2

R6 = R3 + R5   (2)   Height = 2 because height = height of child + latency = 1 + 1

1

ST R6 D   (1)   Height = 1 because latency of ST = 1

Height-based list scheduling

1. LD A, R1
2. LD B, R2
3. R3 = R1 + R2
4. LD C, R4
5. R5 = R4 * R2
6. R6 = R3 + R5
7. ST R6, D

Cycle	ALU0	ALU1	LD/ST
0			2
1			2
2			4
3			4
4	5		1
5			1
6	3		
7	6		
8	7		
9			
10			

Instruction Scheduling - Exercise

- 2 ALUs (fully pipelined) and one LD/ST unit (not pipelined) are available.
- Either of the ALUs can execute ADD (1 cycle). Only one of the ALUs can execute MUL (2 cycles).
- LDs take up an ALU for 1 cycle and LD/ST unit for two cycles.
- STs take up an ALU for 1 cycle and LD/ST unit for one cycle.

*i) Draw reservation tables, ii)DAG for the code shown iii) schedule using height based list scheduling.*

```
1:  LD A R1            11:  ST R10   E
2:  LD B R2            12:  ST R7    F
3:  LD C R3
4:  LD D R4
5:  R5 = R1 + R2
6:  R6 = R5 * R3
7:  R7 = R1 + R6
8:  R8 = R6 + R5
9:  R9 = R4 + R7
10: R10 = R9 + R8
```

7

Basic Blocks and Flow Graphs

- Basic Block
  - Maximal sequence of consecutive instructions with the following properties:
    - The first instruction of the basic block is the *only entry point*
    - The last instruction of the basic block is either the halt instruction or the *only exit point*

- Flow Graph
  - Nodes are the basic blocks
  - Directed edge indicates which block follows which block

Basic Blocks and Flow Graphs - Example

```
if A = B then
    C := 1;
    D := 2;
else
    E := 3
fi
A := 1;
```
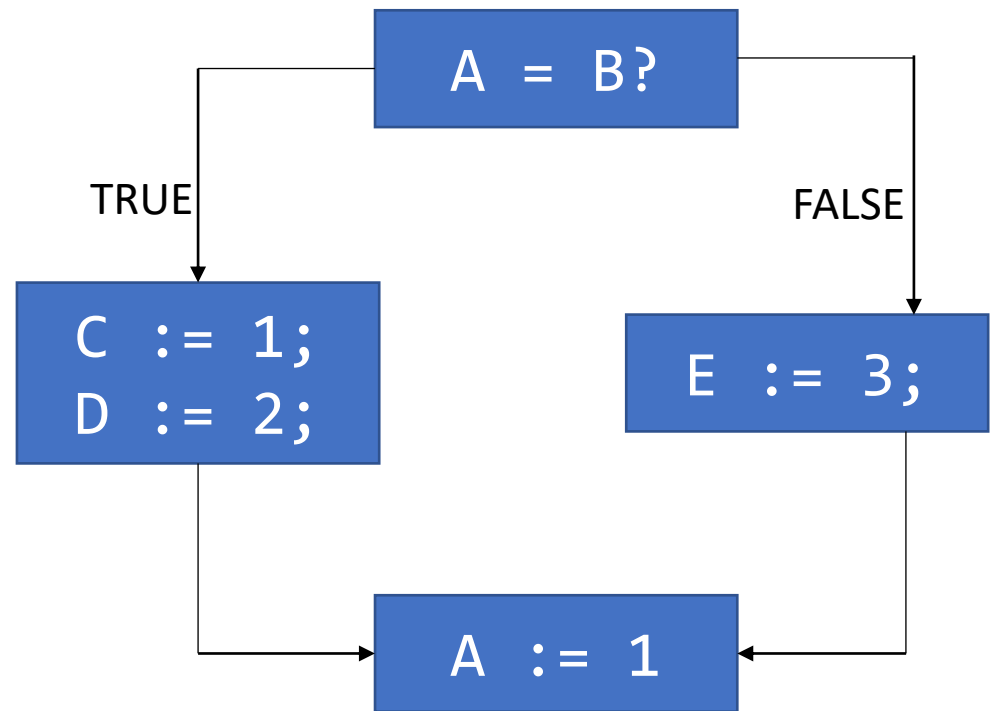

A data flow graph

Flow Graphs

- Capture how control transfers between basic blocks due to:
  - Conditional constructs
  - Loops
- Are necessary when we want optimize considering larger parts of the program
  - Multiple procedures
  - Whole program

Flow Graphs - Representation

- We need to label and track statements that are jump targets
  - **Explicit targets** – targets mentioned in jump statement
  - **Implicit targets** – targets that follow conditional jump statement
    - Statement that is executed if the branch is not taken
- Implementation
  - Linked lists for Basic Blocks
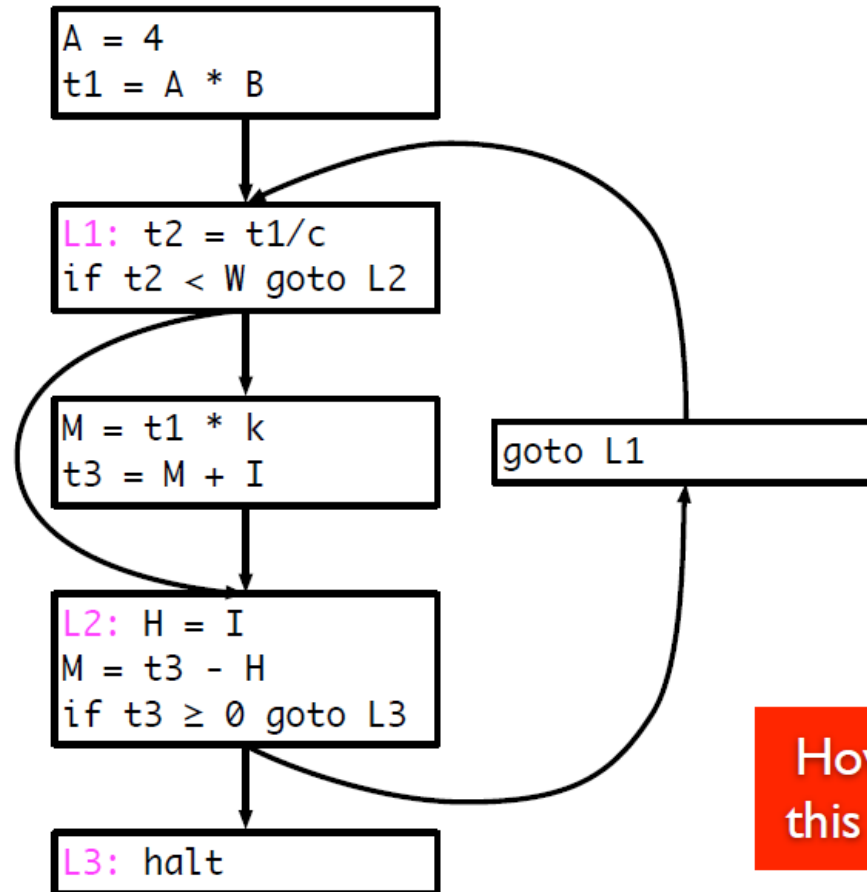 - Graph data structures for flow graphs

Running example

```
A = 4
t1 = A * B
repeat {
  t2 = t1/C
  if (t2 ≥ W) {
    M = t1 * k
    t3 = M + I
  }
  H = I
  M = t3 - H
} until (T3 ≥ 0)
```

Running example

```
1        A = 4
2        t1 = A * B
3   L1:  t2 = t1 / C
4        if t2 < W goto L2
5        M = t1 * k
6        t3 = M + I
7   L2:  H = I
8        M = t3 - H
9        if t3 ≥ 0 goto L3
10       goto L1
11  L3:  halt
```

CFG for running example


```
A = 4
t1 = A * B
```

```
L1: t2 = t1/c
if t2 < W goto L2
```

```
M = t1 * k
t3 = M + I
```

```
goto L1
```

```
L2: H = I
M = t3 - H
if t3 ≥ 0 goto L3
```

```
L3: halt
```

How do we build this automatically?

Constructing a CFG

- To construct a CFG where each node is a basic block

    - Identify *leaders*: first statement of a basic block

    - In program order, construct a block by appending subsequent statements up to, but not including, the next leader

- Identifying leaders

    - First statement in the program

    - Explicit target of any conditional or unconditional branch

    - Implicit target of any branch

Partitioning algorithm

- Input: set of statements, *stat(i)* = i[th] statement in input

- Output: set of *leaders*, set of basic blocks where *block(x)* is the set of statements in the block with leader *x*

- Algorithm

```
leaders = {1}          //Leaders always includes first statement
for i = 1 to |n|       //|n| = number of statements
    if stat(i) is a branch, then
        leaders = leaders ∪ all potential targets
end for
worklist = leaders
while worklist not empty do
    x = remove earliest statement in worklist
    block(x) = {x}
    for (i = x + 1; i ≤ |n| and i ∉ leaders; i++)
        block(x) = block(x) ∪ {i}
    end for
end while
```

Running example

```
 1        A = 4
 2        t1 = A * B
 3  L1:   t2 = t1 / C
 4        if t2 < W goto L2
 5        M = t1 * k
 6        t3 = M + I
 7  L2:   H = I
 8        M = t3 - H
 9        if t3 ≥ 0 goto L3
10        goto L1
11  L3:   halt
```

Leaders = ?
Basic blocks = ?

leaders = {1}   //Leaders always includes first statement
for i = 1 to |n| //|n| = number of statements
    if *stat(i)* is a branch, **then**
        leaders = *leaders* ∪ all potential targets
end for
worklist = *leaders*

Running example

```
 1        A = 4
 2        t1 = A * B
 3  L1:   t2 = t1 / C
 4        if t2 < W goto L2
 5        M = t1 * k
 6        t3 = M + I
 7  L2:   H = I
 8        M = t3 - H
 9        if t3 ≥ 0 goto L3
10        goto L1
11  L3:   halt
```

Leaders = {1}
Basic blocks =

Running example

```
 1        A = 4
 2        t1 = A * B
 3   L1:  t2 = t1 / C
 4        if t2 < W goto L2
 5        M = t1 * k
 6        t3 = M + I
 7   L2:  H = I
 8        M = t3 - H
 9        if t3 ≥ 0 goto L3
10        goto L1
11   L3:  halt
```

Leaders = {1}
Basic blocks =

Running example

```
 1        A = 4
 2        t1 = A * B
 3  L1:   t2 = t1 / C
 4        if t2 < W goto L2
 5        M = t1 * k
 6        t3 = M + I
 7  L2:   H = I
 8        M = t3 - H
 9        if t3 ≥ 0 goto L3
10        goto L1
11  L3:   halt
```

Leaders = {1,3}
Basic blocks =

Running example

```
1        A = 4
2        t1 = A * B
3   L1:  t2 = t1 / C
4        if t2 < W goto L2
5        M = t1 * k
6        t3 = M + I
7   L2:  H = I
8        M = t3 - H
9        if t3 ≥ 0 goto L3
10       goto L1
11  L3:  halt
```

Leaders = {1,3}
Basic blocks =

Running example

```
 1         A = 4
 2         t1 = A * B
 3   L1:   t2 = t1 / C
 4         if t2 < W goto L2
 5         M = t1 * k
 6         t3 = M + I
 7   L2:   H = I
 8         M = t3 - H
 9         if t3 ≥ 0 goto L3
10         goto L1
11   L3:   halt
```

Leaders = {1,3,5}
Basic blocks =

Running example

```
 1        A = 4
 2        t1 = A * B
 3  L1:   t2 = t1 / C
 4        if t2 < W goto L2
 5        M = t1 * k
 6        t3 = M + I
 7  L2:   H = I
 8        M = t3 - H
 9        if t3 ≥ 0 goto L3
10        goto L1
11  L3:   halt
```

Leaders = {1,3,5}
Basic blocks =

Running example

```
 1        A = 4
 2        t1 = A * B
 3  L1:   t2 = t1 / C
 4        if t2 < W goto L2
 5        M = t1 * k
 6        t3 = M + I
 7  L2:   H = I
 8        M = t3 - H
 9        if t3 ≥ 0 goto L3
10        goto L1
11  L3:   halt
```

Leaders = {1,3,5,7}
Basic blocks =

Running example

```
1          A = 4
2          t1 = A * B
3   L1:    t2 = t1 / C
4          if t2 < W goto L2
5          M = t1 * k
6          t3 = M + I
7   L2:    H = I
8          M = t3 - H
9          if t3 ≥ 0 goto L3
10         goto L1
11  L3:    halt
```

Leaders = {1,3,5,7}
Basic blocks =

Running example

```
1          A = 4
2          t1 = A * B
3   L1:    t2 = t1 / C
4          if t2 < W goto L2
5          M = t1 * k
6          t3 = M + I
7   L2:    H = I
8          M = t3 - H
9          if t3 ≥ 0 goto L3
10         goto L1
11  L3:    halt
```

Leaders = {1,3,5,7}
Basic blocks =

Running example

```
1          A = 4
2          t1 = A * B
3   L1:    t2 = t1 / C
4          if t2 < W goto L2
5          M = t1 * k
6          t3 = M + I
7   L2:    H = I
8          M = t3 - H
9          if t3 ≥ 0 goto L3
10         goto L1
11  L3:    halt
```

Leaders = {1,3,5,7,10}
Basic blocks =

Running example

```
1         A = 4
2         t1 = A * B
3  L1:    t2 = t1 / C
4         if t2 < W goto L2
5         M = t1 * k
6         t3 = M + I
7  L2:    H = I
8         M = t3 - H
9         if t3 ≥ 0 goto L3
10        goto L1
11 L3:    halt
```

Leaders = {1,3,5,7,10,11}
Basic blocks = ?

worklist = *leaders*
while *worklist* not empty **do**
   x = remove earliest statement in *worklist*
   block(x) = {*x*}
   for ($i = x + 1; i \leq |n|$ and $i \notin$ *leaders*; i++)
      block(x) = *block(x)* ∪ {*i*}
   end for
end while

Running example

```
 1         A = 4
 2         t1 = A * B
 3  L1:    t2 = t1 / C
 4         if t2 < W goto L2
 5         M = t1 * k
 6         t3 = M + I
 7  L2:    H = I
 8         M = t3 - H
 9         if t3 ≥ 0 goto L3
10         goto L1
11  L3:    halt
```

Leaders = {1,3,5,7,10,11}   Block(1) = ?
Basic blocks =

Running example

```
 1         A = 4
 2         t1 = A * B
 3  L1:    t2 = t1 / C
 4         if t2 < W goto L2
 5         M = t1 * k
 6         t3 = M + I
 7  L2:    H = I
 8         M = t3 - H
 9         if t3 ≥ 0 goto L3
10         goto L1
11  L3:    halt
```

Leaders = {1,3,5,7,10,11}
Basic blocks =

Block(1) = ?
Start from statement 2 and add till either the end or a leader is reached

Running example

```
 1         A = 4
 2         t1 = A * B
 3  L1:    t2 = t1 / C
 4         if t2 < W goto L2
 5         M = t1 * k
 6         t3 = M + I
 7  L2:    H = I
 8         M = t3 - H
 9         if t3 ≥ 0 goto L3
10         goto L1
11  L3:    halt
```

Leaders = {1,3,5,7,10,11}   Block(1) = {1,2}
Basic blocks =

Running example

```
 1          A = 4
 2          t1 = A * B
 3   L1:    t2 = t1 / C
 4          if t2 < W goto L2
 5          M = t1 * k
 6          t3 = M + I
 7   L2:    H = I
 8          M = t3 - H
 9          if t3 ≥ 0 goto L3
10          goto L1
11   L3:    halt
```

Leaders = {1,3,5,7,10,11}   Block(3) = ?
Basic blocks =

Running example

```
 1         A = 4
 2         t1 = A * B
 3  L1:   t2 = t1 / C
 4         if t2 < W goto L2
 5         M = t1 * k
 6         t3 = M + I
 7  L2:   H = I
 8         M = t3 - H
 9         if t3 ≥ 0 goto L3
10         goto L1
11  L3:   halt
```

Leaders = {1,3,5,7,10,11}   Block(3) = {3,4}
Basic blocks =

Running example

```
 1        A = 4
 2        t1 = A * B
 3  L1:   t2 = t1 / C
 4        if t2 < W goto L2
 5        M = t1 * k
 6        t3 = M + I
 7  L2:   H = I
 8        M = t3 - H
 9        if t3 ≥ 0 goto L3
10        goto L1
11  L3:   halt
```

Leaders = $\{1,3,5,7,10,11\}$ Block(5) = ?
Basic blocks =

Running example

```
 1        A = 4
 2        t1 = A * B
 3  L1:   t2 = t1 / C
 4        if t2 < W goto L2
 5        M = t1 * k
 6        t3 = M + I
 7  L2:   H = I
 8        M = t3 - H
 9        if t3 ≥ 0 goto L3
10        goto L1
11  L3:   halt
```

Leaders = {1,3,5,7,10,11}   Block(5) = {5,6}
Basic blocks =

Running example

```
 1        A = 4
 2        t1 = A * B
 3   L1:  t2 = t1 / C
 4        if t2 < W goto L2
 5        M = t1 * k
 6        t3 = M + I
 7   L2:  H = I
 8        M = t3 - H
 9        if t3 ≥ 0 goto L3
10        goto L1
11   L3:  halt
```

Leaders = {1,3,5,7,10,11}   Block(7) = ?
Basic blocks =

Running example

```
 1        A = 4
 2        t1 = A * B
 3  L1:   t2 = t1 / C
 4        if t2 < W goto L2
 5        M = t1 * k
 6        t3 = M + I
 7  L2:   H = I
 8        M = t3 - H
 9        if t3 ≥ 0 goto L3
10        goto L1
11  L3:   halt
```

Leaders = {1,3,5,7,10,11}   Block(7) = {7,8,9}
Basic blocks =

Running example

```
1         A = 4
2         t1 = A * B
3  L1:    t2 = t1 / C
4         if t2 < W goto L2
5         M = t1 * k
6         t3 = M + I
7  L2:    H = I
8         M = t3 - H
9         if t3 ≥ 0 goto L3
10        goto L1
11 L3:    halt
```

Leaders = {1,3,5,7,10,11}   Block(10) = ?
Basic blocks =

Running example

```
1          A = 4
2          t1 = A * B
3    L1:   t2 = t1 / C
4          if t2 < W goto L2
5          M = t1 * k
6          t3 = M + I
7    L2:   H = I
8          M = t3 - H
9          if t3 ≥ 0 goto L3
10         goto L1
11   L3:   halt
```

Leaders = {1,3,5,7,10,11}   Block(10) = {10}
Basic blocks =

Running example

```
1        A = 4
2        t1 = A * B
3  L1:   t2 = t1 / C
4        if t2 < W goto L2
5        M = t1 * k
6        t3 = M + I
7  L2:   H = I
8        M = t3 - H
9        if t3 ≥ 0 goto L3
10       goto L1
11 L3:   halt
```

Leaders = {1,3,5,7,10,11}   Block(11) = {11}
Basic blocks =

Running example

```
1        A = 4
2        t1 = A * B
3   L1:  t2 = t1 / C
4        if t2 < W goto L2
5        M = t1 * k
6        t3 = M + I
7   L2:  H = I
8        M = t3 - H
9        if t3 ≥ 0 goto L3
10       goto L1
11  L3:  halt
```

Leaders =   {1, 3, 5, 7, 10, 11}
Basic blocks =   { {1, 2}, {3, 4}, {5, 6}, {7, 8, 9}, {10}, {11} }