Intro to dataflow analysis

CSE 501

Spring 15

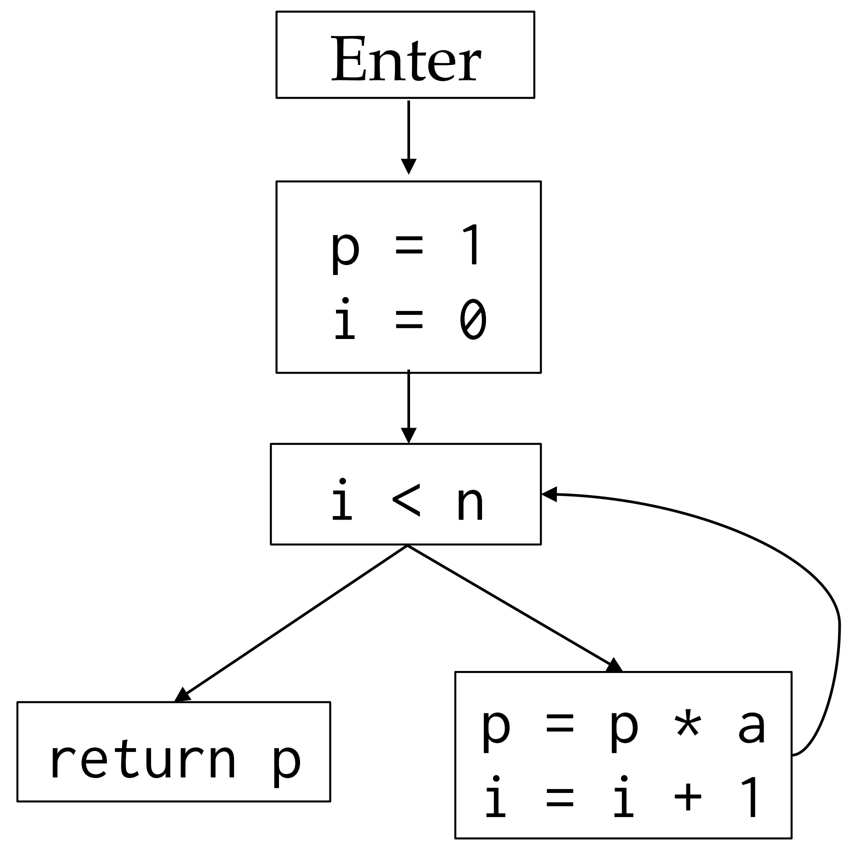
Announcements

- Paper commentaries
  - Please post them 24 hours before class

- Application paper presentations
  - Good training for conference talks!
  - Will help go through slides the day before
  - Part of class participation grade
  - Will post signups on course website

Control-flow Graph

- Directed graph
  - Each node is a statement
  - Edges represents possible flow of control
- Statements
  - Assignments
  - Branches
  - Enter / return
  - Declarations usually omitted



Dataflow Analysis

- Collect program information without actually running it
  - Too good to be true?

- Many uses:
  - Compiler optimizations
  - Bug detection
  - (will see more in subsequent lectures)

Dataflow Framework

- <G, L, F, M>
- G = flow graph
- L = (semi-)lattice
- F / M = flow / transfer functions

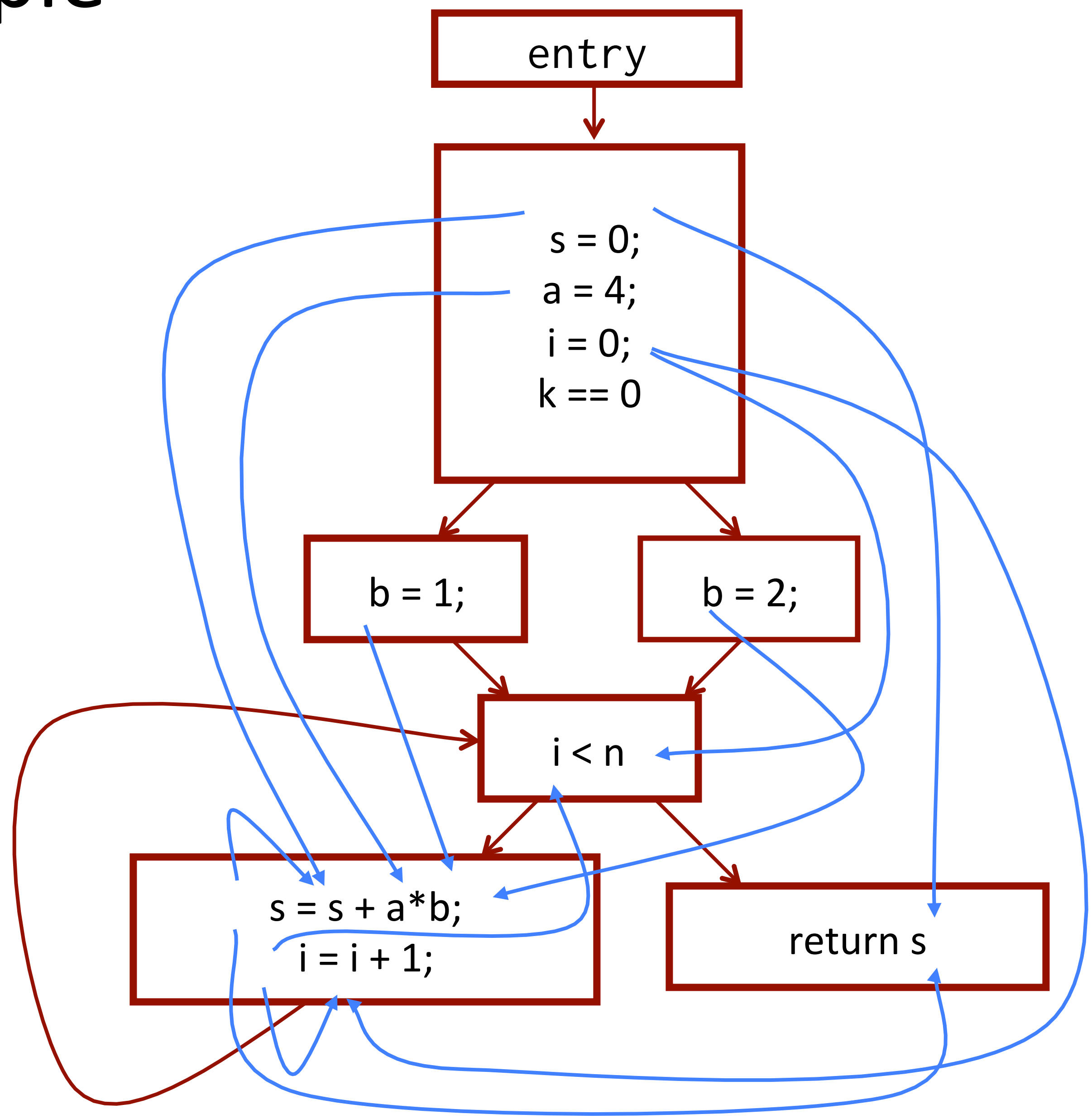
Example: Reaching Definitions

- Concept of definition and use
  - z = x+y
  - is a definition of z
  - is a use of x and y
- A definition reaches a use if
  - value written by definition
  - may be read by use

Example: Reaching Definitions

- Problem:
  - For each basic block,
    find all definitions that reach it

Example

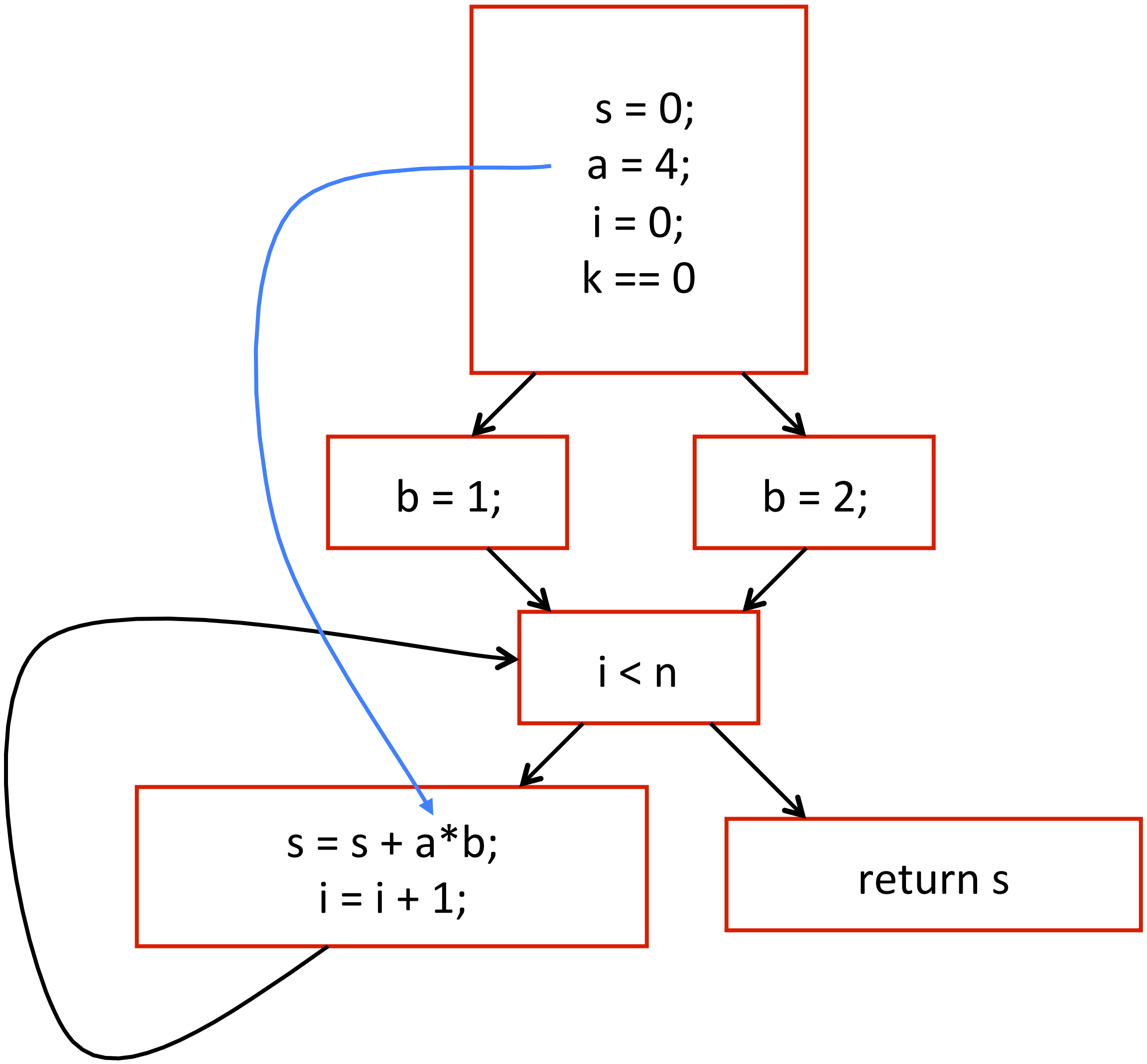


Why bother?

- Is a use of a variable a constant?
  - Check all reaching definitions
  - If all assign variable to same constant
  - Then use is in fact a constant

- Can replace variable with constant

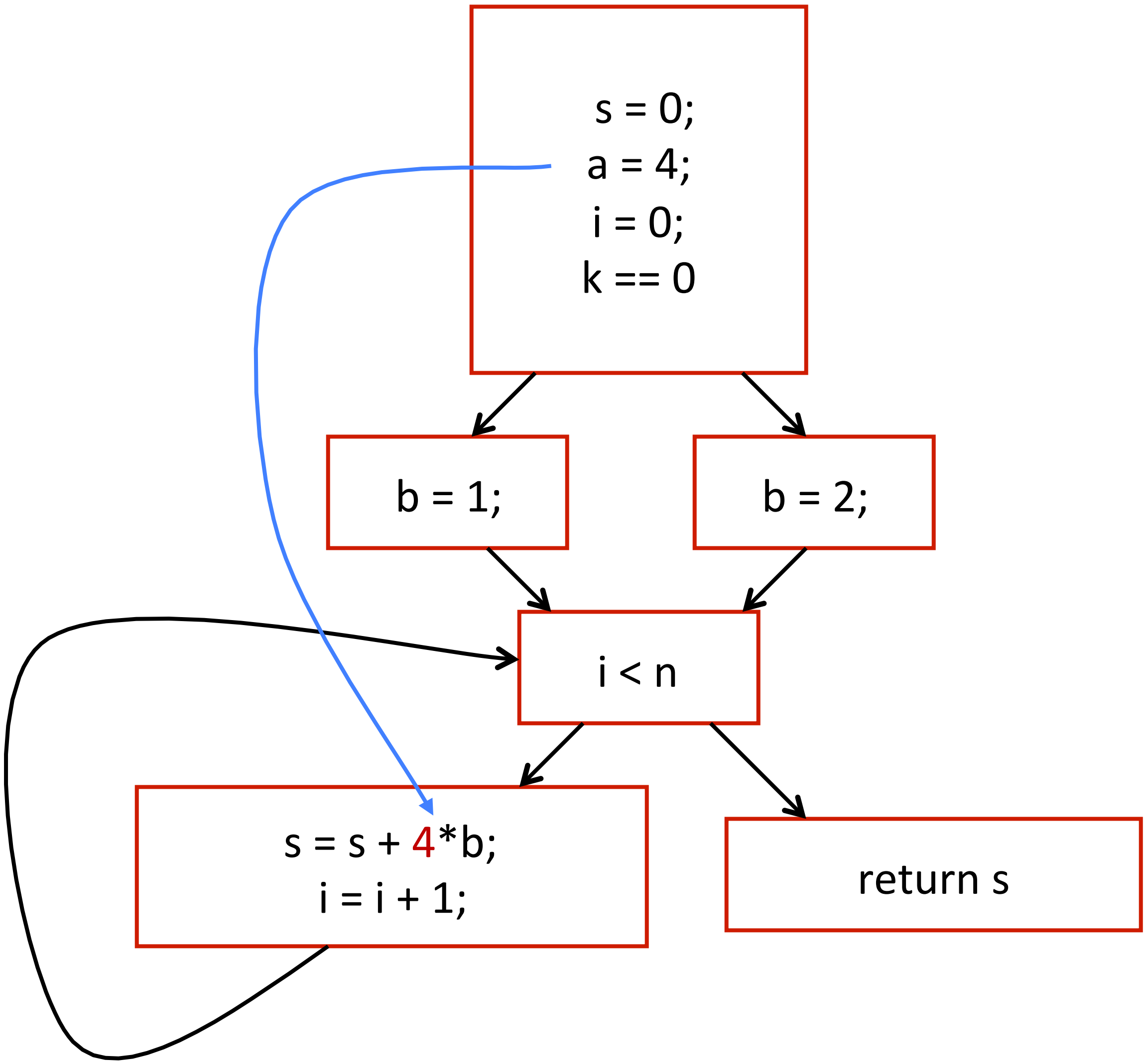
Is **a** constant in s = s+a*b?



Yes!

$a = 4$

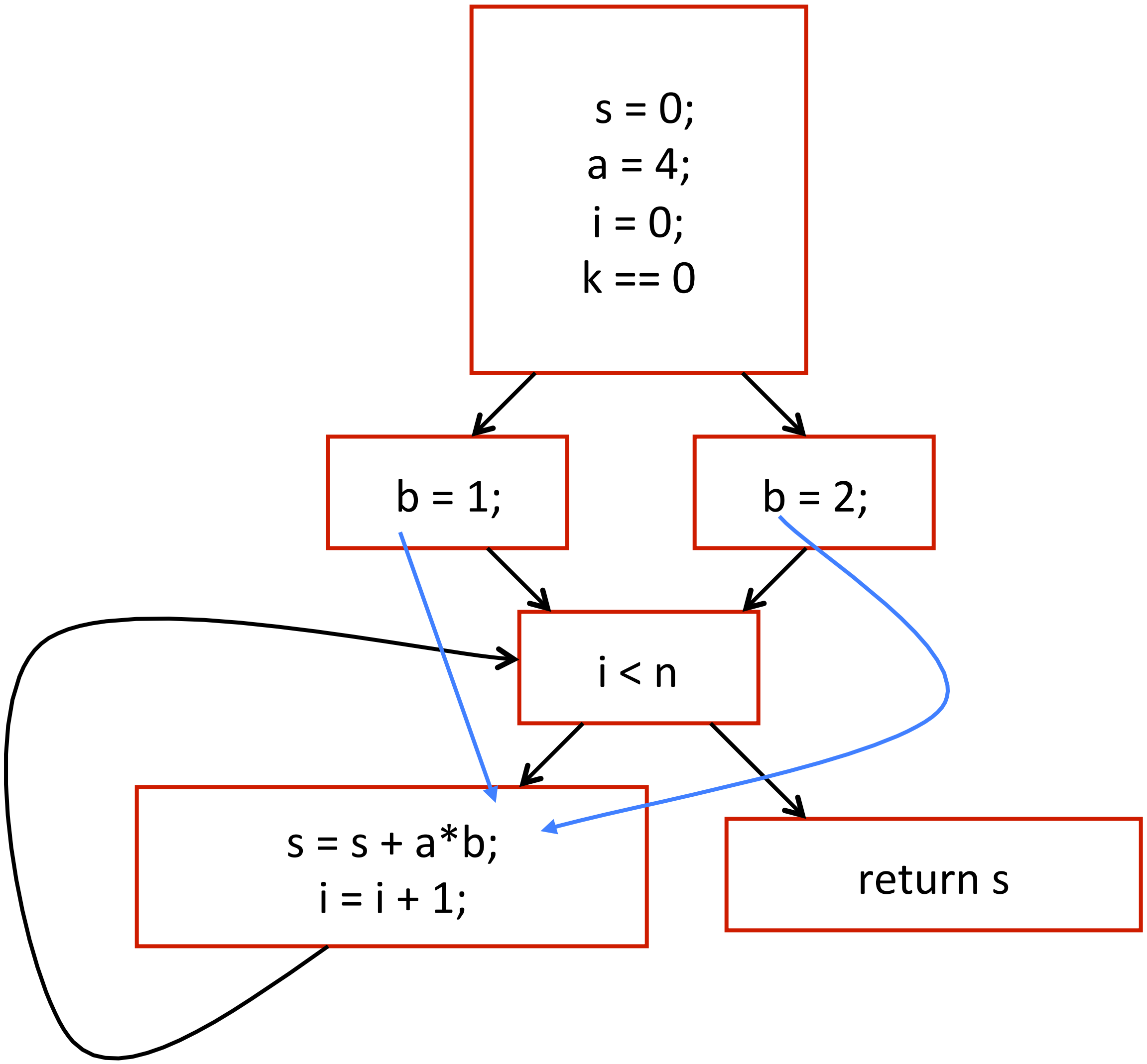
Constant Propagation Transform



Yes!

$a = 4$

Is **b** Constant in s = s+a*b?



No!

b = 1

b = 2

Computing Reaching Definitions

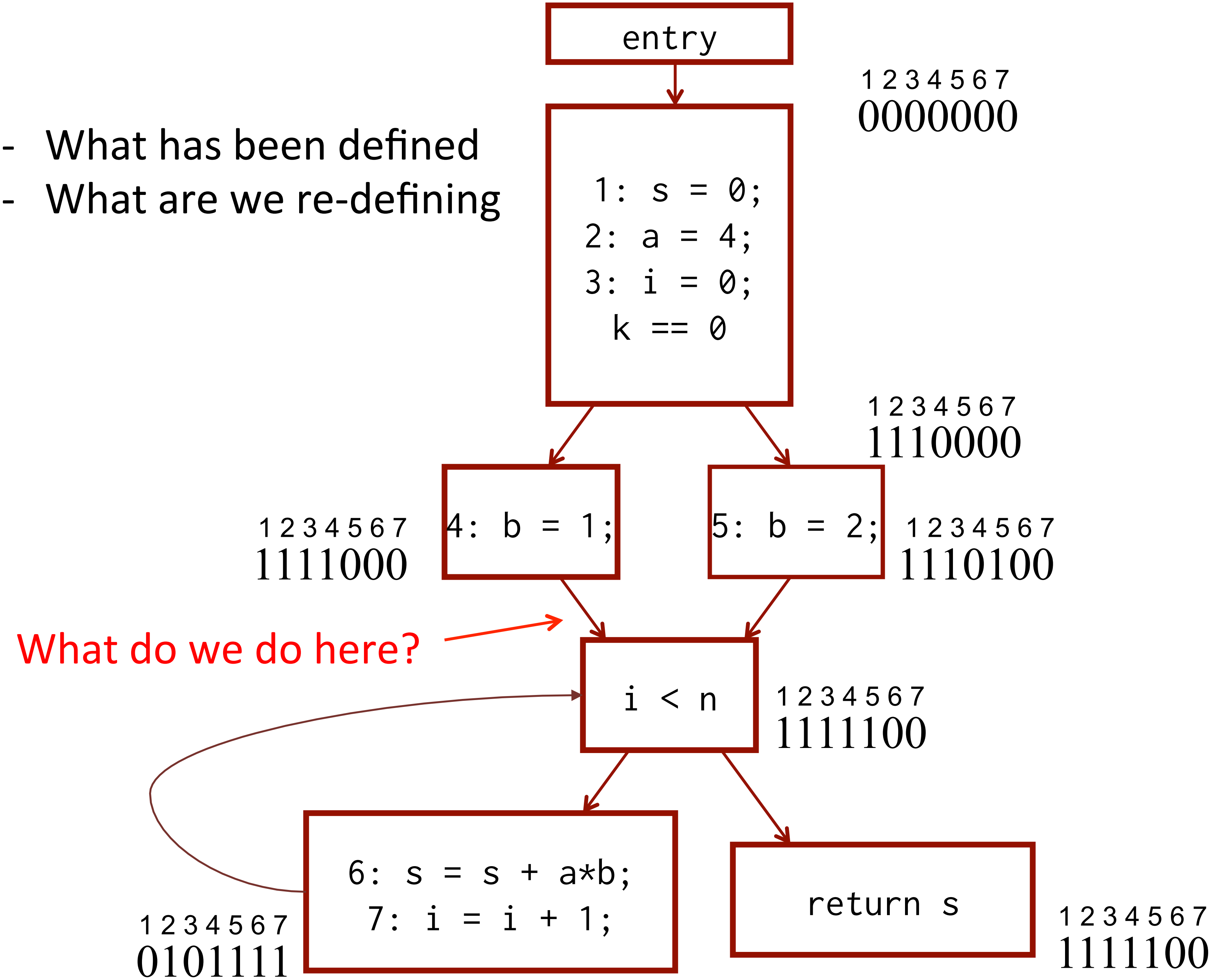
- Generate control flow graph of function
- Compute with sets of definitions
  - represent sets using bit vectors
  - each **definition** has a position in the bit vector
- At each basic block, compute
  - definitions that reach start of block
  - definitions that reach end of block

Setting up

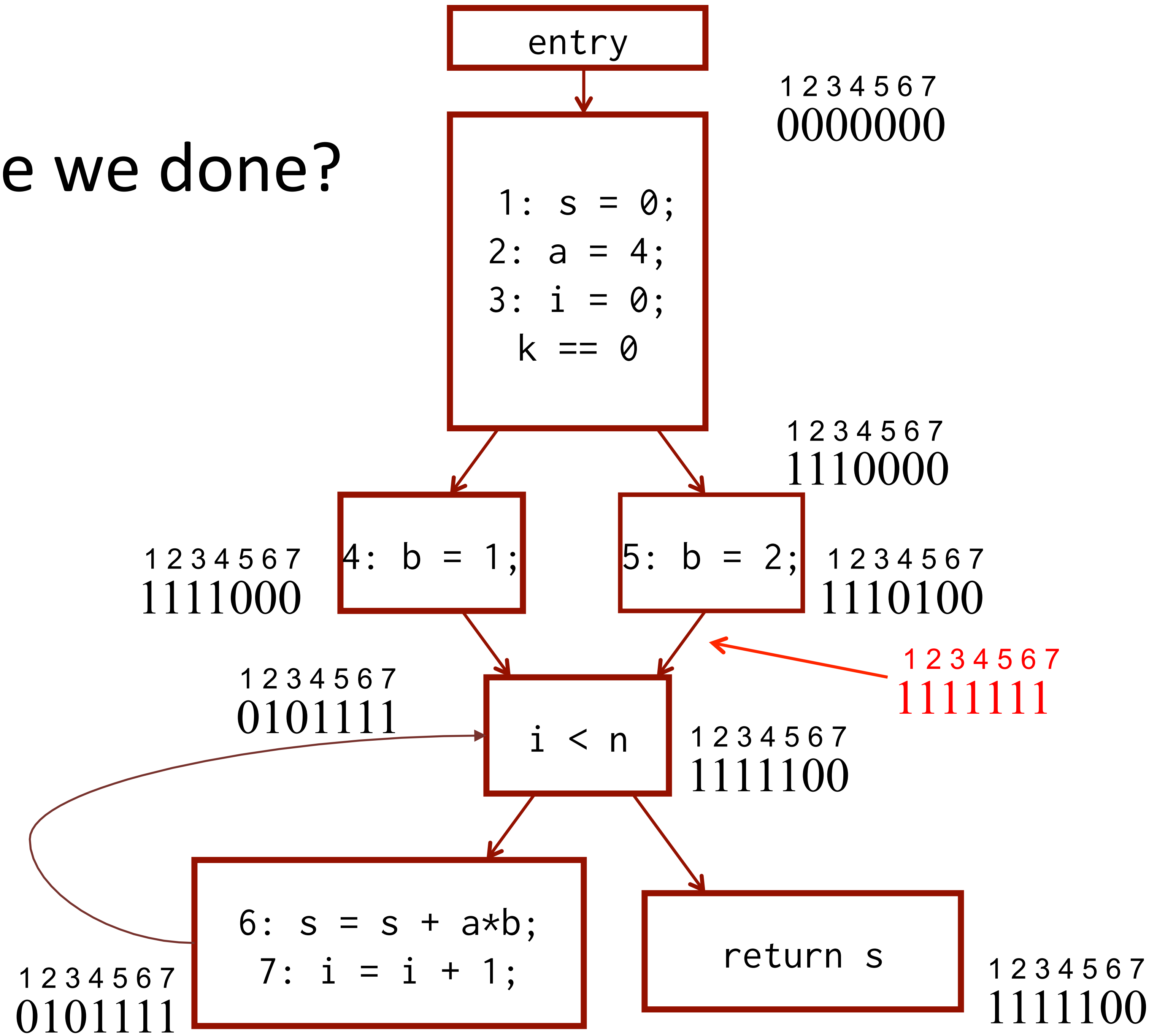
- Boundary condition:
  - Nothing get propagated out of the exit block

- Initial assumptions:
  - All blocks produce no definitions

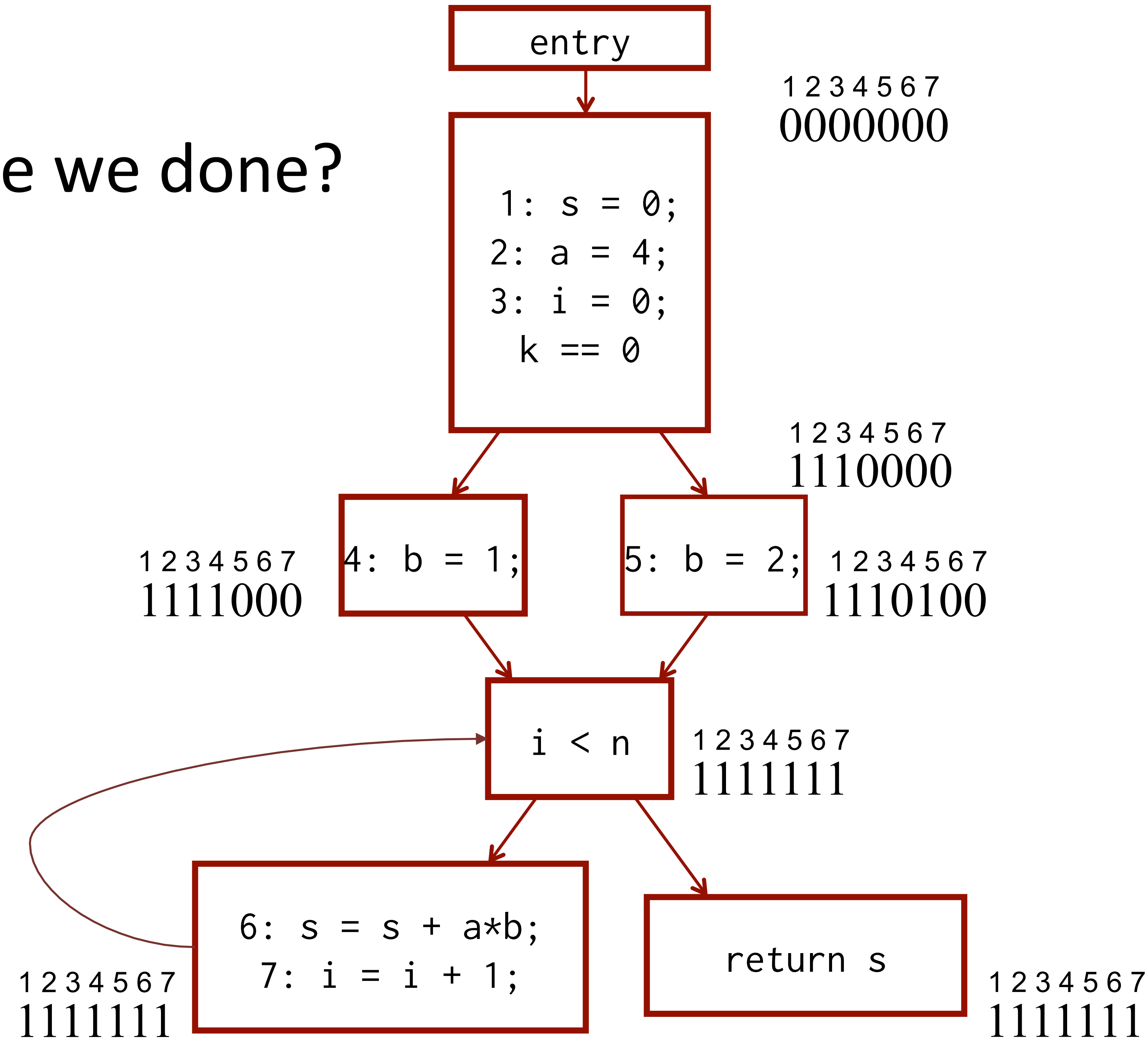
- What has been defined
- What are we re-defining



Are we done?


entry
1 2 3 4 5 6 7
0000000
1: s = 0;
2: a = 4;
3: i = 0;
k == 0
1 2 3 4 5 6 7
1110000
1 2 3 4 5 6 7
1111000
4: b = 1;
5: b = 2;
1 2 3 4 5 6 7
1110100
1 2 3 4 5 6 7
1111111
1 2 3 4 5 6 7
0101111
i < n
1 2 3 4 5 6 7
1111100
6: s = s + a*b;
7: i = i + 1;
return s
1 2 3 4 5 6 7
0101111
1 2 3 4 5 6 7
1111100

Are we done?



entry
1: s = 0;
2: a = 4;
3: i = 0;
k == 0
1 2 3 4 5 6 7
0000000
4: b = 1;
1 2 3 4 5 6 7
1111000
5: b = 2;
1 2 3 4 5 6 7
1110000
1 2 3 4 5 6 7
1110100
i < n
1 2 3 4 5 6 7
1111111
6: s = s + a*b;
7: i = i + 1;
1 2 3 4 5 6 7
1111111
return s
1 2 3 4 5 6 7
1111111

Dataflow Framework

- <G, L, F, M>
- G = flow graph
- L = (semi-)lattice
- F / M = flow / transfer functions

Computing Reaching Definitions

- Generate control flow graph of function

  Flow graph

- Compute with sets of definitions
  - represent sets using bit vectors ← Semi-lattice
  - each definition has a position in the bit vector

- At each basic block, compute ← Transfer function
  - definitions that reach start of block
  - definitions that reach end of block

Transfer functions

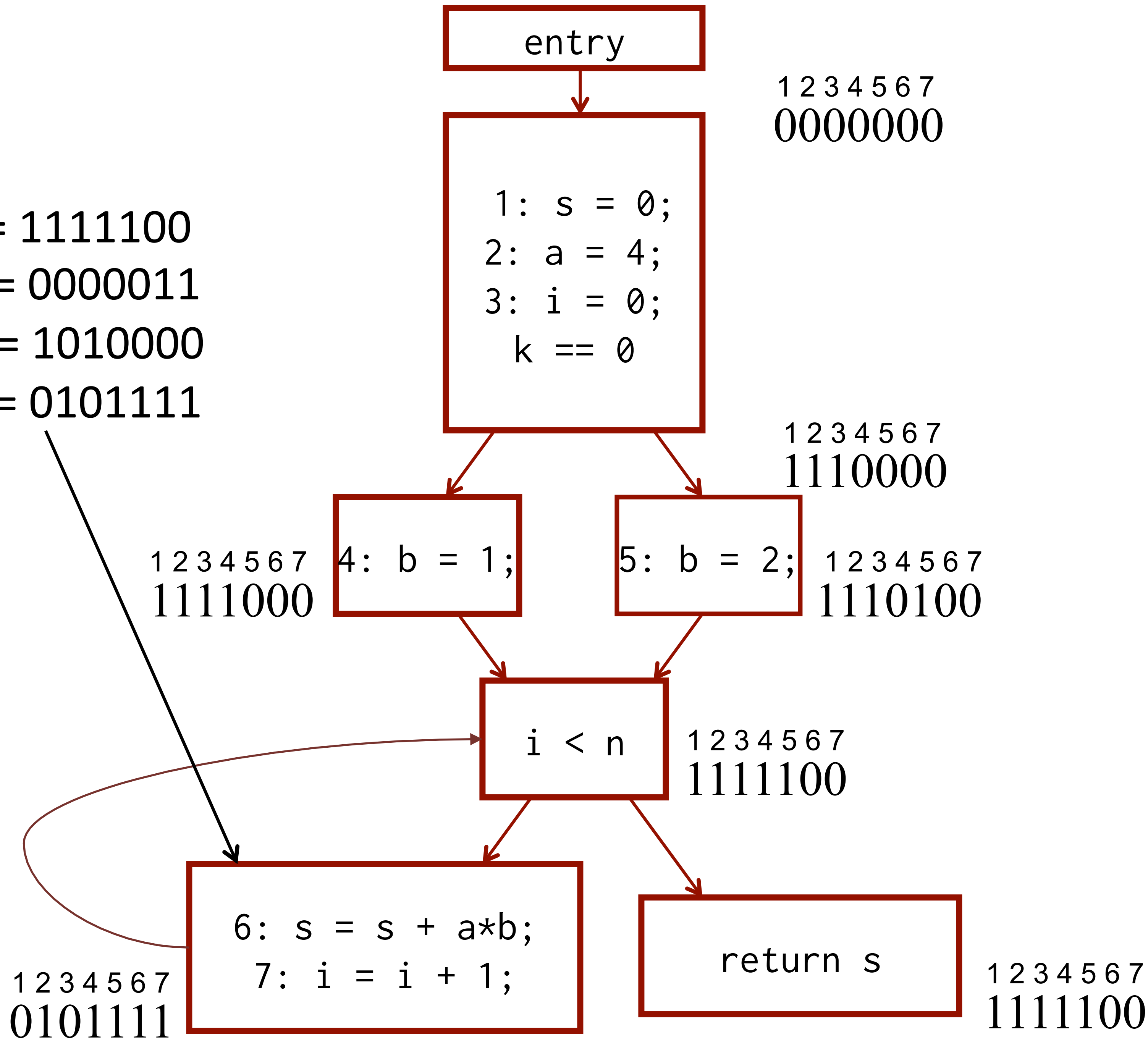
- Each basic block has
  - IN - set of definitions that reach beginning of block
  - OUT - set of definitions that reach end of block
- For this analysis, define:
  - GEN - set of definitions generated in block
  - KILL - set of definitions killed in block
- Analyzer scans each basic block to derive GEN and KILL sets for each function, and then compute OUT

IN = 1111100

GEN = 0000011

KILL = 1010000

OUT = 0101111



Dataflow Equations

- IN[b] = OUT[b1] U ... U OUT[bn]
  - where b1, ..., bn are predecessors of b in CFG
- OUT[b] = (IN[b] - KILL[b]) U GEN[b]
- IN[entry] = 0000000
- Result: system of equations from each basic block

Solving Equations

- Initialize with solution of OUT[b] = 0000000
- Repeatedly apply equations
  - IN[b] = OUT[b1] U ... U OUT[bn]
  - OUT[b] = (IN[b] - KILL[b]) U GEN[b]
- Until reach fixed point
  - Until equation application has no further effect
- Solve using iterative algorithm

Solving Equations

```
Input: flow graph (CFG)
// boundary condition
OUT[Entry] = 0...0
// initial conditions
for each basic block B other than entry
  OUT[B] = 0...0
// iterate
while (any out[] changes value) {
  for each basic block B other than entry {
    IN[B] = U (OUT[p]), for all predecessor block p of B
    OUT[B] = (IN[B] − KILL[B]) U GEN[B]
  }
}
```

Reaching Definitions Summary

Lattice	Sets of definitions represented by bit-vectors
Transfer function	OUT[B] = f_b(IN[B]) f_b(x) = (x – KILL[x]) U GEN[x]
Meet operation	IN[B] = U OUT[Predecessors]
Boundary condition	OUT[entry] = 0….0
Initial condition	OUT[B] = 0….0

Questions

- Does the algorithm halt?
  - yes, because transfer function is monotonic
  - if increase IN, increase OUT
  - in limit, all bits are 1
- If bit is 0, does the corresponding definition ever reach basic block?
- If bit is 1, does the corresponding definition always reach the basic block?