

Lecture 5: Data-Flow Analysis Examples

17-355/17-665/17-819: Program Analysis

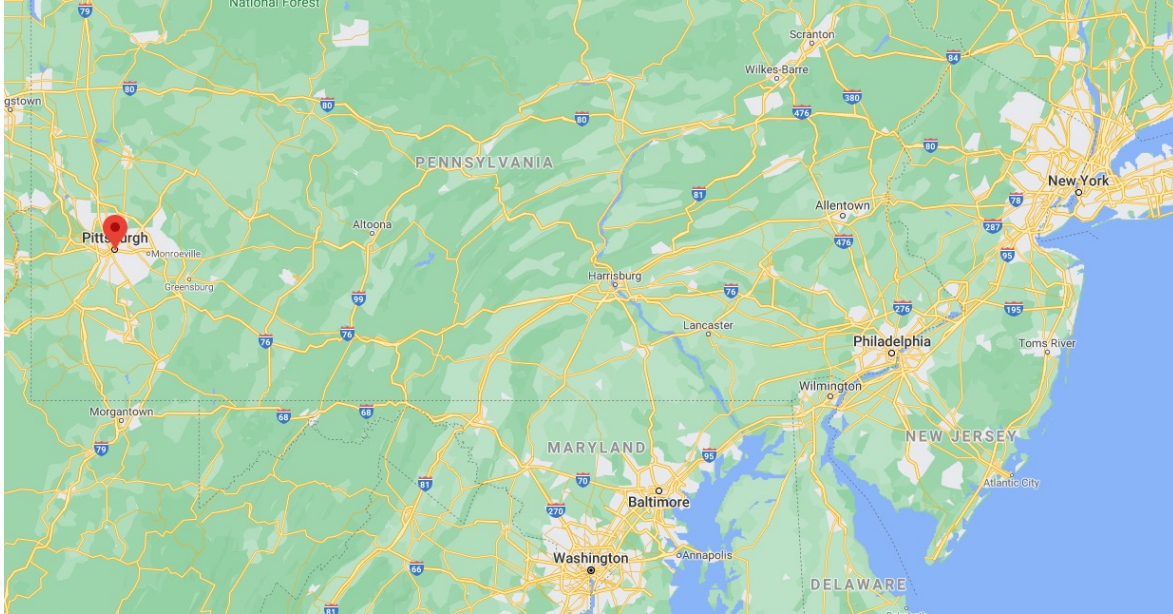
Rohan Padhye

February 1, 2022

* Course materials developed with Jonathan Aldrich and Claire Le Goues

Trivia

“You are here” maps don’t lie - Banach mapping theorem



What mathematical concept is common to both these facts?

Python 3.8:

```
exec(s:='print("exec(s:=%r)"%s)')
```

Review: Data-Flow Analysis

- a lattice (L, \sqsubseteq)
- an abstraction function α
- a flow function f
- initial dataflow analysis assumptions, σ_0

Classic Data-Flow Analyses

- Zero Analysis
- Integer Sign Analysis
- Constant Propagation
- Reaching Definitions
- Live Variables Analysis

- Available Expressions
- Very Busy Expressions
- ...

Integer Sign Analysis

- Extension of Zero Analysis to track integers zero, less-than-zero, greater-than-zero, or $\overline{_}(_)_/_$ (unknown).
- **Q:** Why do we care about sign?
- **Exercise 1:** What would the lattice for simple Sign Analysis look like?

Integer Sign Analysis

- Extension of simple Sign Analysis to track when $x < 0$, $x \leq 0$, $x = 0$, $x \geq 0$, $x > 0$, $x \neq 0$, or unknown ($\overline{_} \setminus _ (\text{ツ}) _ / \overline{_}$).
- **Q:** Why do we care about all these cases?
- **Exercise 2:** What would the lattice for precise Sign Analysis look like?

Constant Propagation

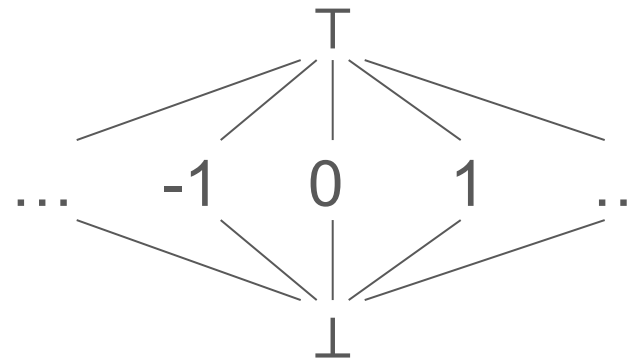
- Extension of Zero / Sign Analysis to track exact values of variables, if they are **constant** at a given program point (across all paths).
- E.g. x is 42 at line 10
- **Q**: Why is this useful?

Constant Propagation

$$\sigma \in Var \rightarrow L_{CP}$$

L_{CP} is $\mathbb{Z} \cup \{\top, \perp\}$

$$\forall l \in L_{CP} : \perp \sqsubseteq l \wedge l \sqsubseteq \top$$



Constant Propagation

$$\begin{aligned}\sigma &\in Var \rightarrow L_{CP} \\ \sigma_1 \sqsubseteq_{lift} \sigma_2 &\text{ iff } \forall x \in Var : \sigma_1(x) \sqsubseteq \sigma_2(x) \\ \sigma_1 \sqcup_{lift} \sigma_2 &= \{x \mapsto \sigma_1(x) \sqcup \sigma_2(x) \mid x \in Var\} \\ \top_{lift} &= \{x \mapsto \top \mid x \in Var\} \\ \perp_{lift} &= \{x \mapsto \perp \mid x \in Var\} \\ \alpha_{CP}(n) &= n \\ \alpha_{lift}(E) &= \{x \mapsto \alpha_{CP}(E(x)) \mid x \in Var\} \\ \sigma_0 &= \top_{lift}\end{aligned}$$

Constant Propagation

$$f_{CP}[\![x := n]\!](\sigma) =$$

$$f_{CP}[\![x := y]\!](\sigma) =$$

$$f_{CP}[\![x := y \text{ op } z]\!](\sigma) =$$

$$f_{CP}[\![\text{goto } n]\!](\sigma) =$$

$$f_{CP}[\![\text{if } x = 0 \text{ goto } n]\!]_T(\sigma) =$$

$$f_{CP}[\![\text{if } x = 0 \text{ goto } n]\!]_F(\sigma) =$$

$$f_{CP}[\![\text{if } x < 0 \text{ goto } n]\!](\sigma) =$$

Constant Propagation

$$f_{CP}[\![x := n]\!](\sigma) = \sigma[x \mapsto \alpha_{CP}(n)]$$

$$f_{CP}[\![x := y]\!](\sigma) = \sigma[x \mapsto \sigma(y)]$$

$$f_{CP}[\![x := y \text{ op } z]\!](\sigma) = \sigma[x \mapsto \sigma(y) \text{ oplift } \sigma(z)]$$

where $n \text{ oplift } m = n \text{ op } m$

$$\text{and } n \text{ oplift } \perp = \perp \quad (\text{and symmetric})$$

$$\text{and } n \text{ oplift } \top = \top \quad (\text{and symmetric})$$

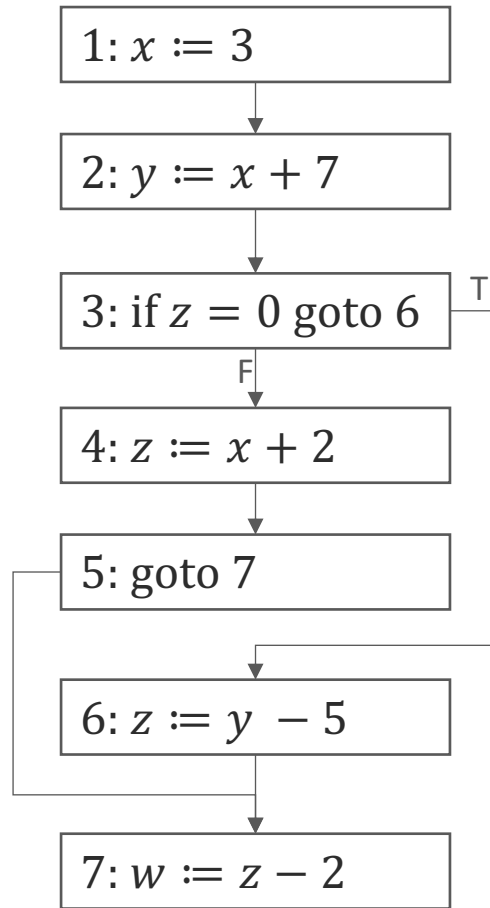
$$f_{CP}[\![\text{goto } n]\!](\sigma) = \sigma$$

$$f_{CP}[\![\text{if } x = 0 \text{ goto } n]\!]_T(\sigma) = \sigma[x \mapsto 0]$$

$$f_{CP}[\![\text{if } x = 0 \text{ goto } n]\!]_F(\sigma) = \sigma$$

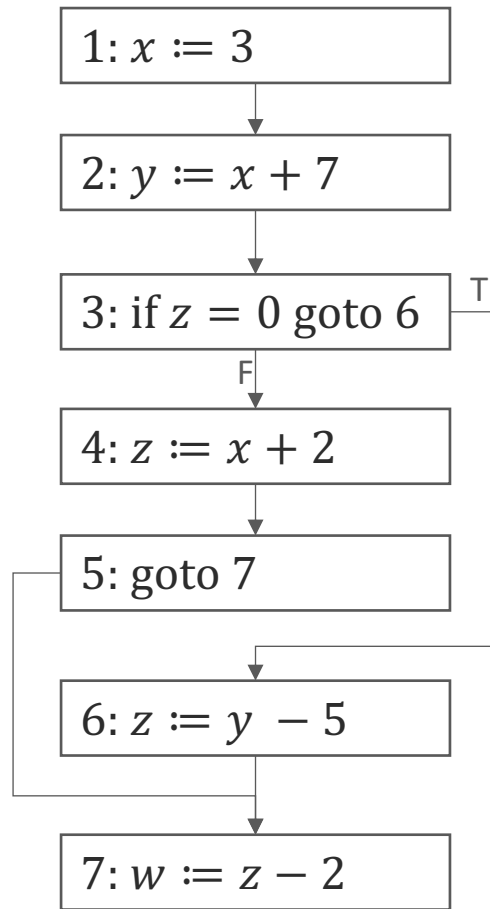
$$f_{CP}[\![\text{if } x < 0 \text{ goto } n]\!](\sigma) = \sigma$$

Constant Propagation



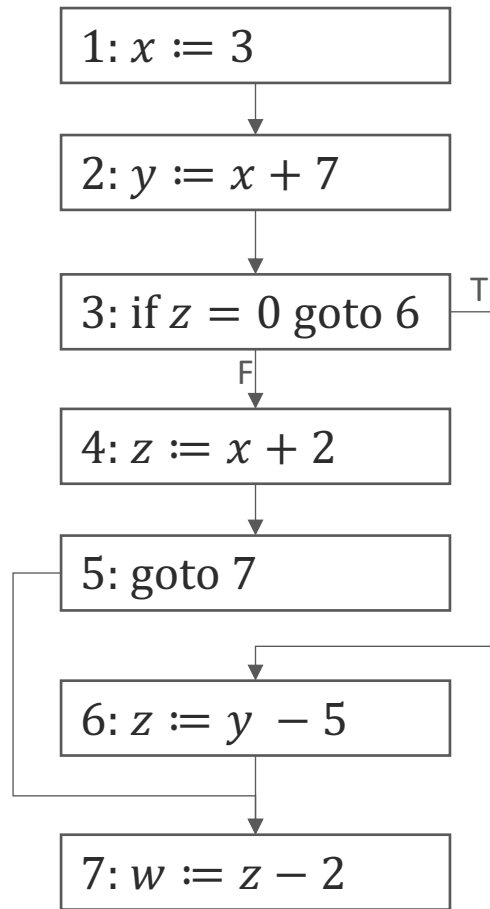
```
1 :  x := 3
2 :  y := x + 7
3 :  if z = 0 goto 6
4 :  z := x + 2
5 :  goto 7
6 :  z := y - 5
7 :  w := z - 2
```

Constant Propagation



stmt	worklist	x	y	z	w

Constant Propagation



stmt	worklist	x	y	z	w
0	1,2,3,4,5,6,7	\top	\top	\top	\top
1	2,3,4,5,6,7	3	\top	\top	\top
2	3,4,5,6,7	3	10	\top	\top
3	4,5,6,7	3	10	$0_T, \top_F$	\top
4	5,6,7	3	10	5	\top
5	6,7	3	10	5	\top
6	7	3	10	5	\top
7	\emptyset	3	10	5	3

Reaching Definitions

- Where might a variable have last been defined?
 - Equivalent: what definitions of a variable *reach* this program point?
 - E.g. At line 7, the value of x was last obtained from assignments at lines 2 and 3.
- Lots of applications in compilers (“def-use chains”)
- Let DEFS = set of all definitions
 - e.g. $\{x_1, x_2, y_3\}$

Reaching Definitions

$$\begin{array}{lcl} \sigma & \in & \mathcal{P}^{\text{DEFS}} \\ \sigma_1 \sqsubseteq \sigma_2 & \text{iff} & \\ \sigma_1 \sqcup \sigma_2 & = & \\ \top & = & \\ \perp & = & \\ \sigma_0 & = & \end{array}$$

Reaching Definitions

$$\begin{array}{lcl} \sigma & \in & \mathcal{P}^{\text{DEFS}} \\ \sigma_1 \sqsubseteq \sigma_2 & \text{iff} & \sigma_1 \subseteq \sigma_2 \\ \sigma_1 \sqcup \sigma_2 & = & \sigma_1 \cup \sigma_2 \\ \top & = & \text{DEFS} \\ \perp & = & \emptyset \\ \sigma_0 & = & \emptyset \end{array}$$

Reaching Definitions

$$f_{RD}[[I]](\sigma) =$$

Reaching Definitions

$$f_{RD}[[I]](\sigma) = \sigma - KILL_{RD}[[I]] \cup GEN_{RD}[[I]]$$

Reaching Definitions

$$f_{RD}[[I]](\sigma) = \sigma - KILL_{RD}[[I]] \cup GEN_{RD}[[I]]$$

$$KILL_{RD}[[n: x := \dots]] =$$

$$KILL_{RD}[[I]] =$$

$$GEN_{RD}[[n: x := \dots]] =$$

$$GEN_{RD}[[I]] =$$

Reaching Definitions

$$f_{RD}[[I]](\sigma) = \sigma - KILL_{RD}[[I]] \cup GEN_{RD}[[I]]$$

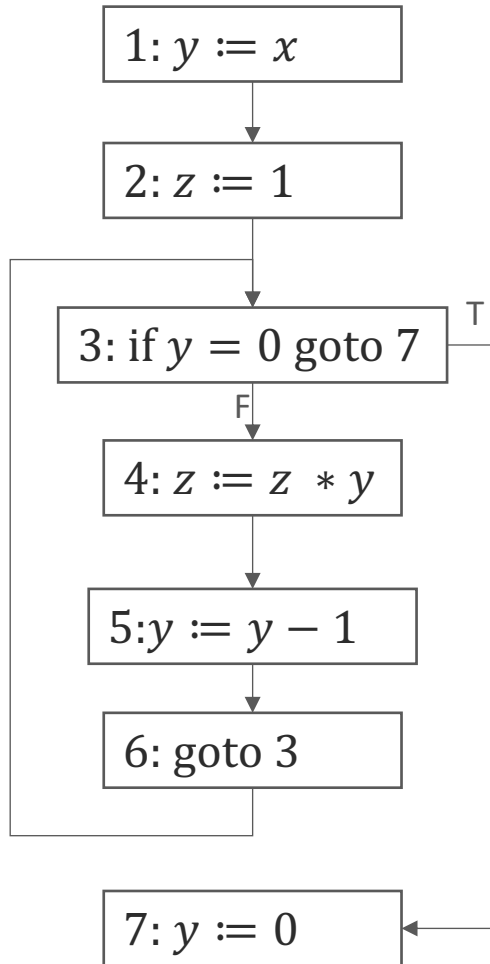
$$KILL_{RD}[[n: x := \dots]] = \{x_m \mid x_m \in \mathbf{DEFS}(x)\}$$

$$KILL_{RD}[[I]] = \emptyset \quad \text{if } I \text{ is not an assignment}$$

$$GEN_{RD}[[n: x := \dots]] = \{x_n\}$$

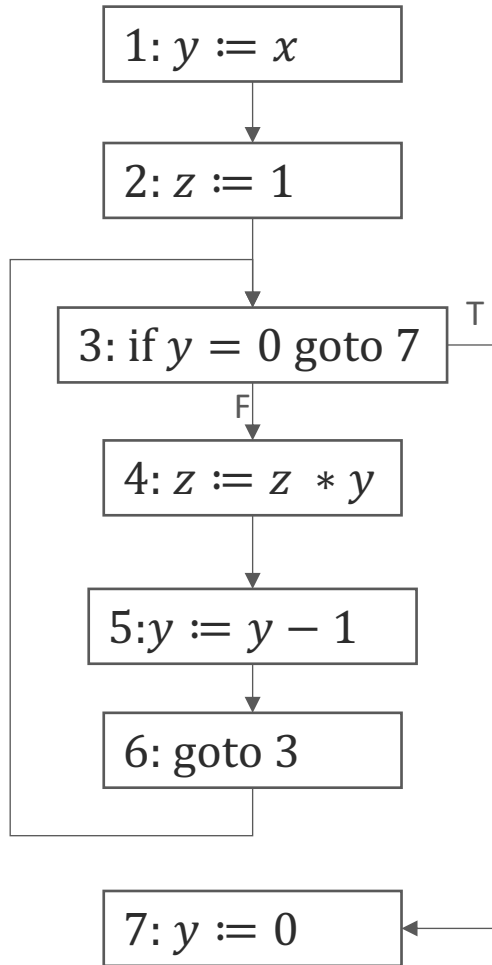
$$GEN_{RD}[[I]] = \emptyset \quad \text{if } I \text{ is not an assignment}$$

Reaching Definitions



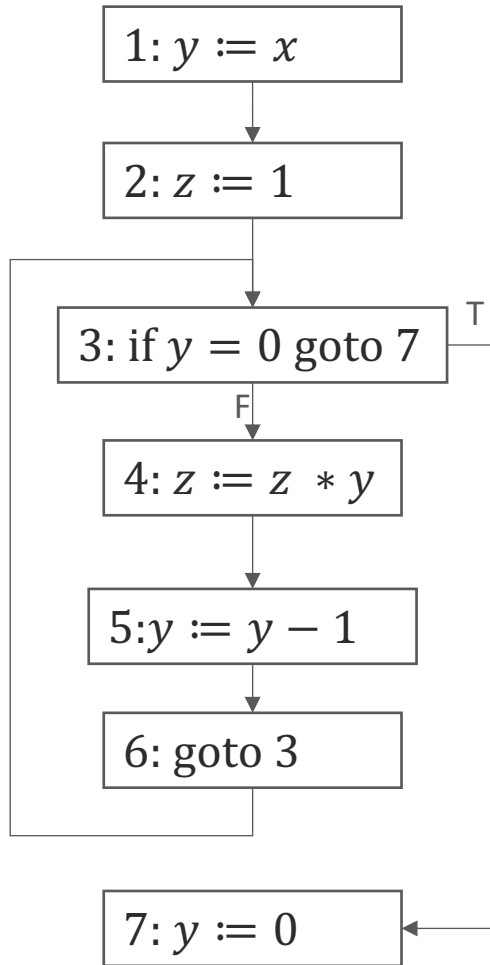
```
1 :  y := x
2 :  z := 1
3 :  if y = 0 goto 7
4 :  z := z * y
5 :  y := y - 1
6 :  goto 3
7 :  y := 0
```

Reaching Definitions



stmt	worklist	defs

Reaching Definitions



stmt	worklist	defs
0	1,2,3,4,5,6,7	\emptyset
1	2,3,4,5,6,7	$\{y_1\}$
2	3,4,5,6,7	$\{y_1, z_1\}$
3	4,5,6,7	$\{y_1, z_1\}$
4	5,6,7	$\{y_1, z_4\}$
5	6,7	$\{y_5, z_4\}$
6	3,7	$\{y_5, z_4\}$
3	4,7	$\{y_1, y_5, z_1, z_4\}$
4	5,7	$\{y_1, y_5, z_4\}$
5	7	$\{y_5, z_4\}$
7	\emptyset	$\{y_7, z_1, z_4\}$

Live Variables

- Which variables will be used in the future (are “live”)?
- E.g. x is live at line 7 because it’s current value will be used at line 10.
- Another set-based analysis (like *reaching definitions*).
- Data-flow values propagate *backwards* !!!

Live Variables

$$\begin{array}{lcl} \sigma & \in & \mathcal{P}^{\text{Var}} \\ \sigma_1 \sqsubseteq \sigma_2 & \text{iff} & \\ \sigma_1 \sqcup \sigma_2 & = & \\ \top & = & \\ \perp & = & \end{array}$$

Live Variables

$$\begin{array}{lcl} \sigma & \in & \mathcal{P}^{\text{Var}} \\ \sigma_1 \sqsubseteq \sigma_2 & \text{iff} & \sigma_1 \subseteq \sigma_2 \\ \sigma_1 \sqcup \sigma_2 & = & \sigma_1 \cup \sigma_2 \\ \top & = & \text{Var} \\ \perp & = & \emptyset \end{array}$$

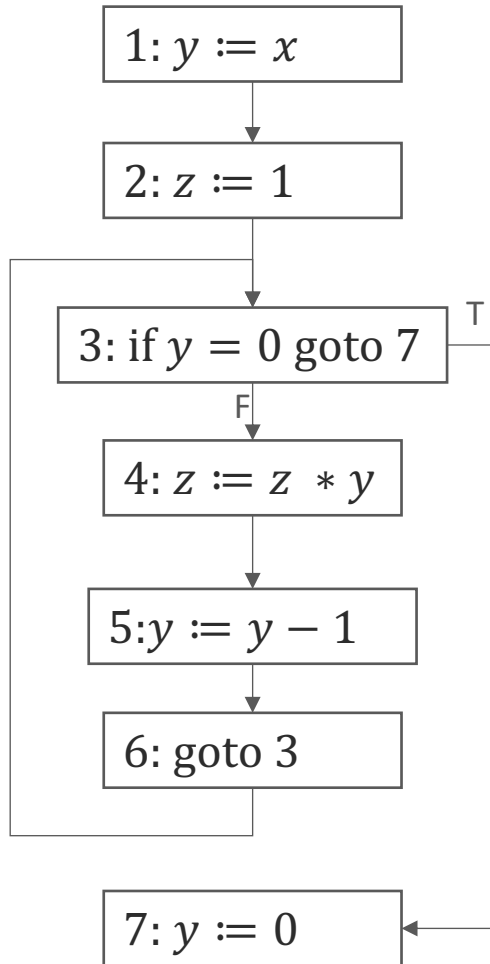
Live Variables

Flow functions map backward! (out \rightarrow in)

$$\text{KILL}_{LV}[[I]] = \{x \mid I \text{ defines } x\}$$

$$\text{GEN}_{LV}[[I]] = \{x \mid I \text{ uses } x\}$$

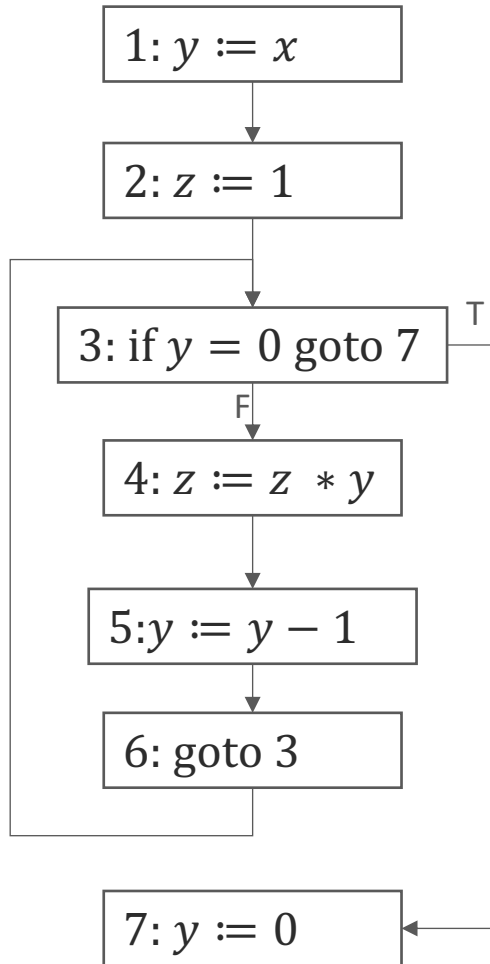
Live Variables



Assume result is in z

```
1 :   $y := x$ 
2 :   $z := 1$ 
3 :  if  $y = 0$  goto 7
4 :   $z := z * y$ 
5 :   $y := y - 1$ 
6 :  goto 3
7 :   $y := 0$ 
```

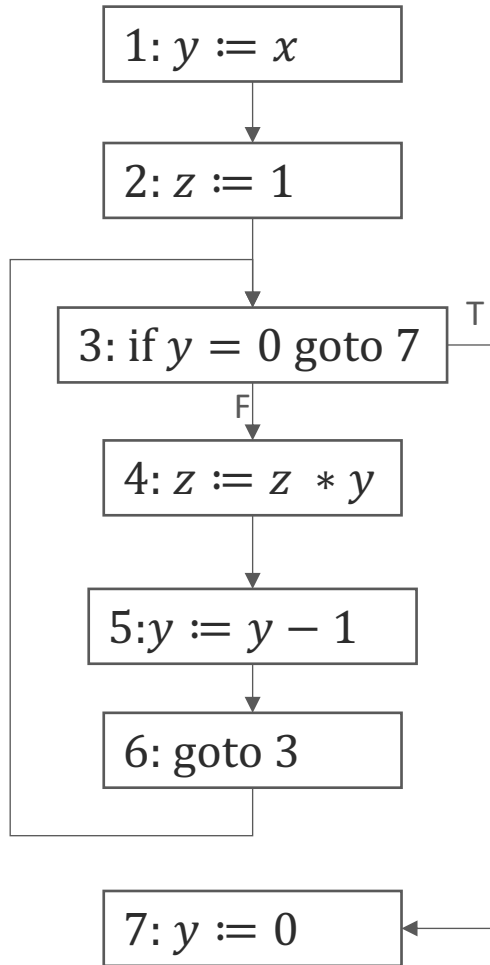
Live Variables



Assume result is in z

stmt	worklist	live

Live Variables



Assume result is in z

stmt	worklist	live
end	7,3,6,5,4,2,1	$\{z\}$
7	3,6,5,4,2,1	$\{z\}$
3	6,5,4,2,1	$\{z, y\}$
6	5,4,2,1	$\{z, y\}$
5	4,2,1	$\{z, y\}$
4	3,2,1	$\{z, y\}$
3	2,1	$\{z, y\}$
2	1	$\{y\}$
1	\emptyset	$\{x\}$