

Data-Flow Analysis

Adapted From Lectures by
Prof. Saman Amarasinghe (MIT)

Outline

- Data-flow analysis
- Available expressions
- Algorithm for calculating available expressions
- Bit sets
- Formulating a data-flow analysis problem
- DU chains
- SSA form

Data-Flow Analysis

- Local Analysis
 - Analyze the effect of each instruction
 - Compose effects of instructions to derive information from beginning of basic block to each instruction
- Data-Flow Analysis
 - Iteratively propagate basic block information over the control-flow graph until no changes
 - Calculate the final value at the beginning of the basic block
- Local Propagation
 - propagate the information from the beginning of the basic block to each instruction

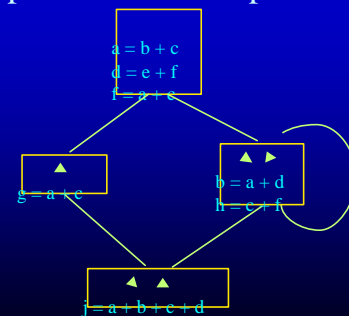
Outline

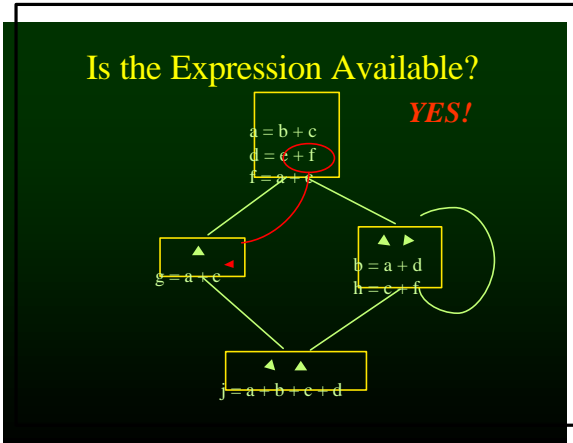
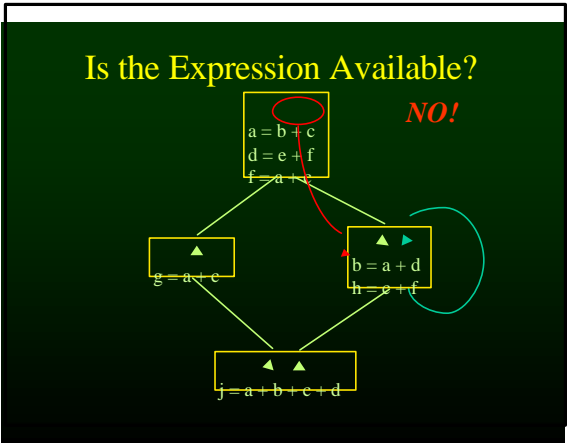
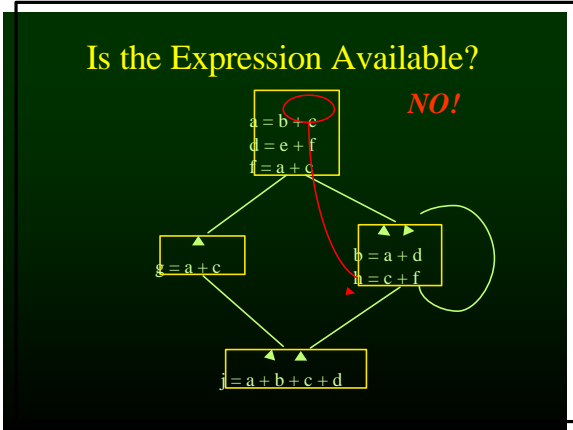
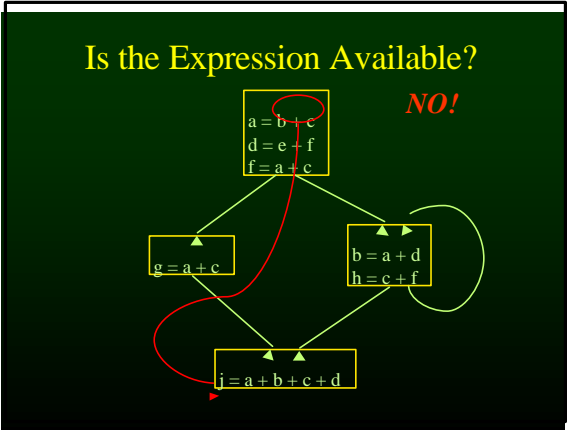
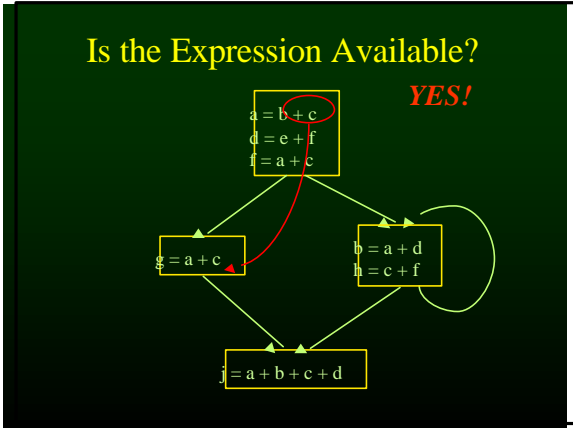
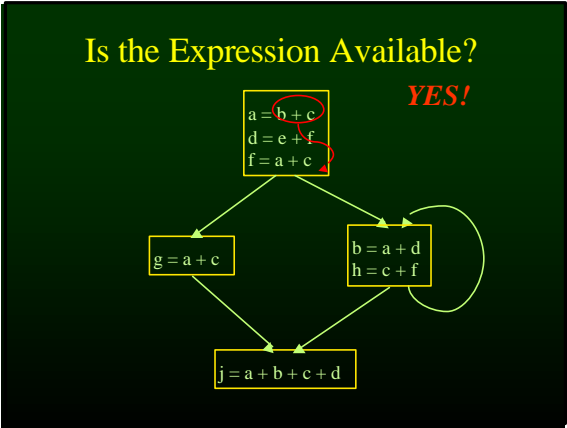
- Overview of data-flow analysis
- Available expressions
- Algorithm for calculating available expressions
- Bit sets
- Formulating a data-flow analysis problem
- DU chains
- SSA form

Example: Available Expression

- An expression is available **if and only if**
 - All paths of execution reaching the current point passes through the point where the expression was defined
 - No variable used in the expression was modified between the definition point and the current point

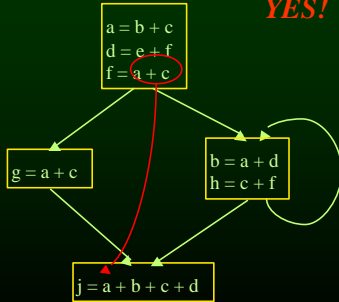
Example: Available Expression



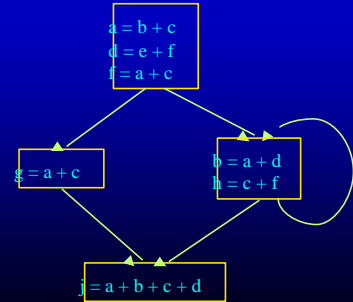


Is the Expression Available?

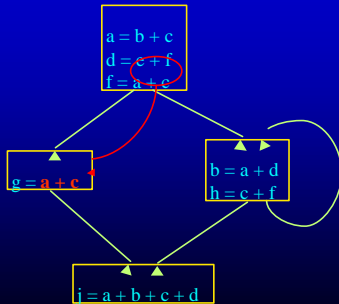
YES!



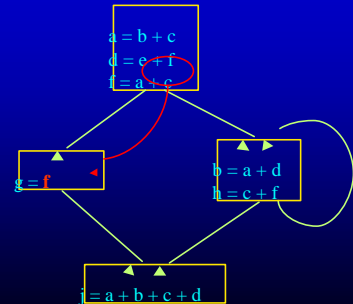
Use of Available Expressions



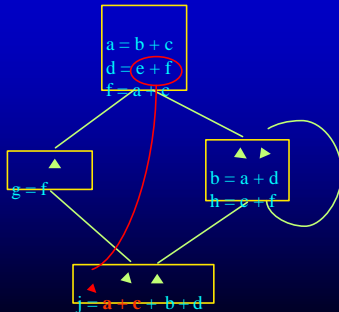
Use of Available Expressions



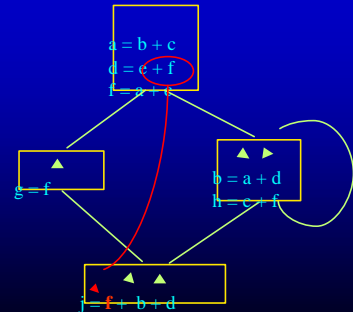
Use of Available Expressions

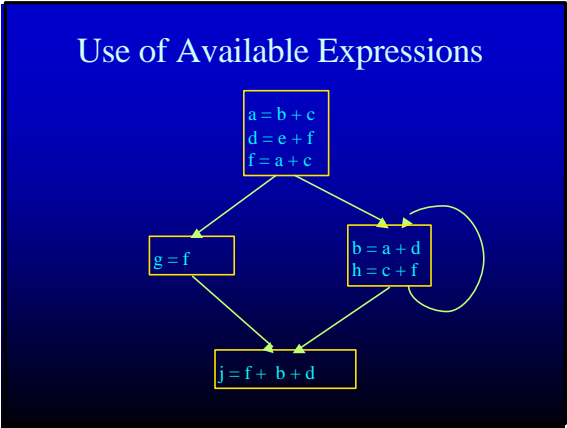


Use of Available Expressions

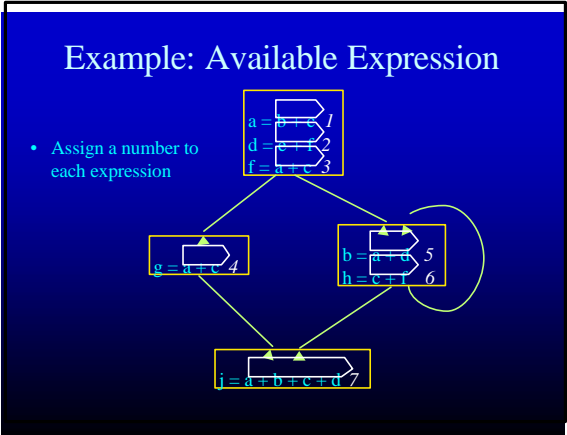


Use of Available Expressions



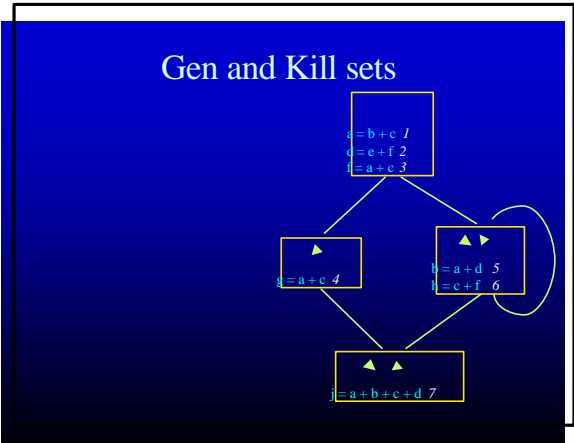


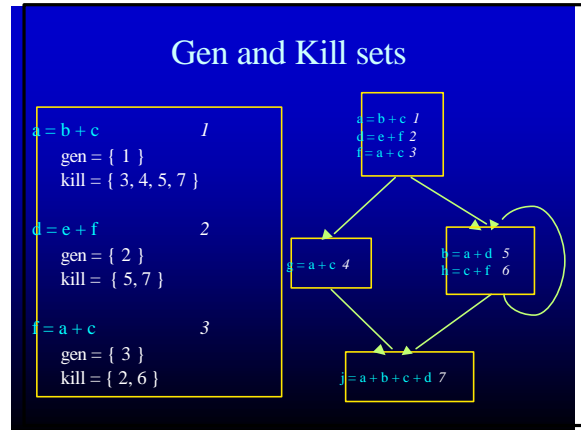
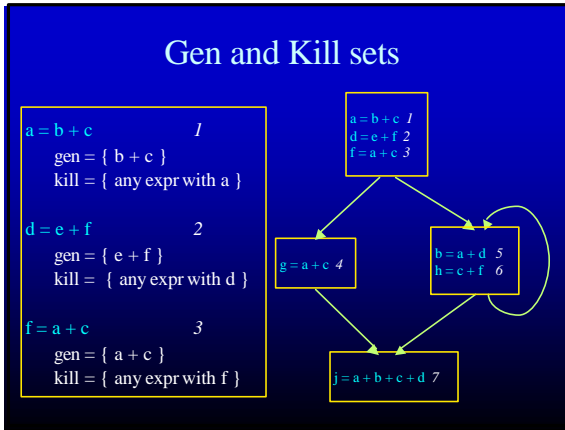
- ### Outline
- Overview of data-flow analysis
 - Available expressions
 - Algorithm for calculating available expressions
 - Bit sets
 - Formulating a data-flow analysis problem
 - DU chains
 - SSA form



- ### Gen and Kill sets
- Gen set
 - If the current basic block (or instruction) creates the definition, it is in the gen set
 - The entry should be in the output no matter what
 - Kill set
 - If the current basic block (or instruction) redefines a variable in the expression, it is in the kill set
 - Expression is not valid after that

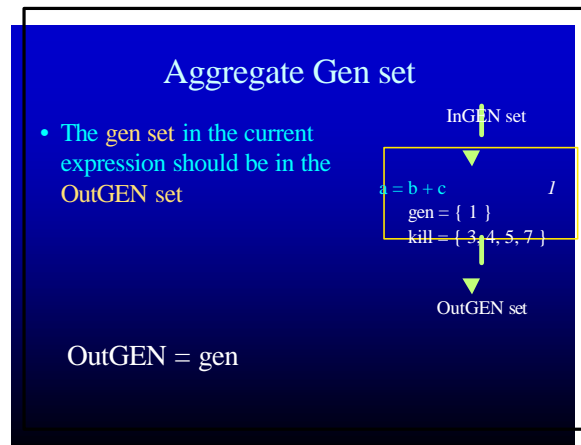
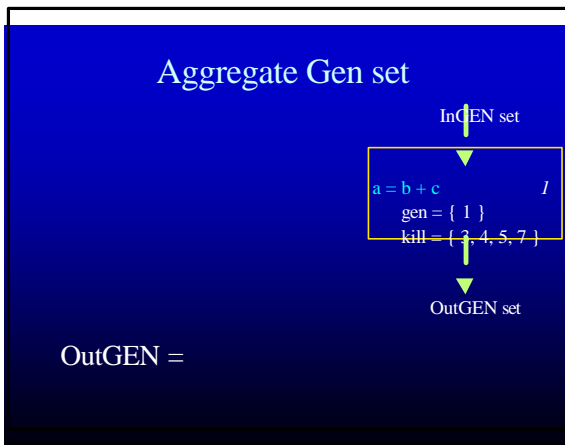
- ### Algorithm for Available Expression
- Assign a number to each expression
 - Calculate gen and kill sets for each instruction





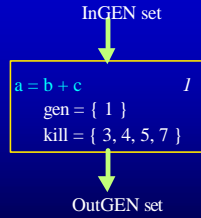
- ### Algorithm for Available Expression
- Assign a number to each expression
 - Calculate **gen** and **kill** sets for each instruction
 - Calculate **aggregate gen** and **kill** sets for each basic block

- ### Aggregate Gen and Kill sets
- | | |
|---|--|
| $a = b + c$ 1 gen = { 1 } kill = { 3, 4, 5, 7 } | <ul style="list-style-type: none"> • Propagate all the gen and kill sets from top of the basic block to the bottom of the basic block |
| $d = e + f$ 2 gen = { 2 } kill = { 5, 7 } | |
| $f = a + c$ 3 gen = { 3 } kill = { 2, 6 } | |



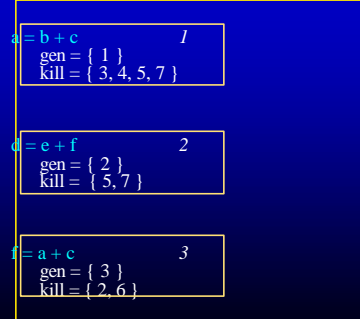
Aggregate Gen set

- The gen set in the current expression should be in the OutGEN set
- Any expression in the InGEN set that is not killed should be in the OutGEN set

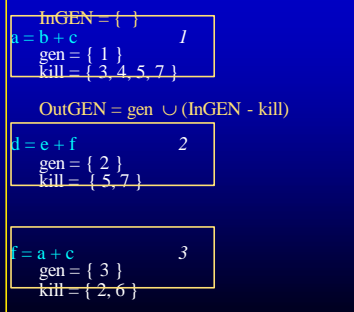


$$\text{OutGEN} = \text{gen} \cup (\text{InGEN} - \text{kill})$$

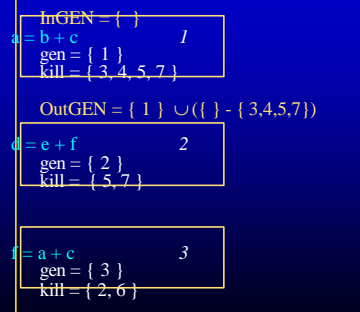
Aggregate Gen set



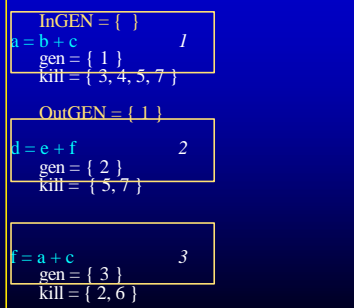
Aggregate Gen set



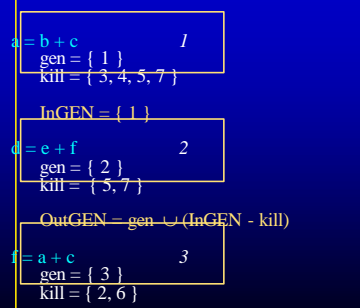
Aggregate Gen set



Aggregate Gen set



Aggregate Gen set



Aggregate Gen set

$a = b + c$ 1
 $gen = \{ 1 \}$
 $kill = \{ 3, 4, 5, 7 \}$

 $InGEN = \{ 1 \}$

 $d = e + f$ 2
 $gen = \{ 2 \}$
 $kill = \{ 5, 7 \}$

 $OutGEN = \{ 2 \} \cup (\{ 1 \} - \{ 5, 7 \})$

 $f = a + c$ 3
 $gen = \{ 3 \}$
 $kill = \{ 2, 6 \}$

Aggregate Gen set

$a = b + c$ 1
 $gen = \{ 1 \}$
 $kill = \{ 3, 4, 5, 7 \}$

 $InGEN = \{ 1 \}$

 $d = e + f$ 2
 $gen = \{ 2 \}$
 $kill = \{ 5, 7 \}$

 $OutGEN = \{ 1, 2 \}$

 $f = a + c$ 3
 $gen = \{ 3 \}$
 $kill = \{ 2, 6 \}$

Aggregate Gen set

$a = b + c$ 1
 $gen = \{ 1 \}$
 $kill = \{ 3, 4, 5, 7 \}$

 $d = e + f$ 2
 $gen = \{ 2 \}$
 $kill = \{ 5, 7 \}$

 $InGEN = \{ 1, 2 \}$

 $f = a + c$ 3
 $gen = \{ 3 \}$
 $kill = \{ 2, 6 \}$
 $OutGEN = gen \cup (InGEN - kill)$

Aggregate Gen set

$a = b + c$ 1
 $gen = \{ 1 \}$
 $kill = \{ 3, 4, 5, 7 \}$

 $d = e + f$ 2
 $gen = \{ 2 \}$
 $kill = \{ 5, 7 \}$

 $InGEN = \{ 1, 2 \}$

 $f = a + c$ 3
 $gen = \{ 3 \}$
 $kill = \{ 2, 6 \}$
 $OutGEN = \{ 3 \} \cup (\{ 1, 2 \} - \{ 2, 6 \})$

Aggregate Gen set

$a = b + c$ 1
 $gen = \{ 1 \}$
 $kill = \{ 3, 4, 5, 7 \}$

 $d = e + f$ 2
 $gen = \{ 2 \}$
 $kill = \{ 5, 7 \}$

 $InGEN = \{ 1, 2 \}$

 $f = a + c$ 3
 $gen = \{ 3 \}$
 $kill = \{ 2, 6 \}$
 $OutGEN = \{ 1, 3 \}$

Aggregate Gen set

$GEN = \{ 1, 3 \}$

 $a = b + c$ 1
 $gen = \{ 1 \}$
 $kill = \{ 3, 4, 5, 7 \}$

 $d = e + f$ 2
 $gen = \{ 2 \}$
 $kill = \{ 5, 7 \}$

 $f = a + c$ 3
 $gen = \{ 3 \}$
 $kill = \{ 2, 6 \}$

Aggregate Kill set

InKILL set

$a = b + c$ 1
gen = { 1 }
kill = { 3, 4, 5, 7 }

↓

OutKILL set

OutKILL =

Aggregate Kill set

- The kill set in the current expression should be in the OutKILL set

InKILL set

$a = b + c$ 1
gen = { 1 }
kill = { 3, 4, 5, 7 }

↓

OutKILL set

OutKILL = kill

Aggregate Kill set

- The kill set in the current expression should be in the OutKILL set
- Any set in the InKILL should be in OutKILL

InKILL set

$a = b + c$ 1
gen = { 1 }
kill = { 3, 4, 5, 7 }

↓

OutKILL set

OutKILL = kill \cup InKILL

Aggregate Kill set

$a = b + c$ 1
gen = { 1 }
kill = { 3, 4, 5, 7 }

$d = e + f$ 2
gen = { 2 }
kill = { 5, 7 }

$f = a + c$ 3
gen = { 3 }
kill = { 2, 6 }

Aggregate Kill set

InKILL = { }

$a = b + c$ 1
gen = { 1 }
kill = { 3, 4, 5, 7 }

OutKILL = kill \cup InKILL

$d = e + f$ 2
gen = { 2 }
kill = { 5, 7 }

$f = a + c$ 3
gen = { 3 }
kill = { 2, 6 }

Aggregate Kill set

InKILL = { }

$a = b + c$ 1
gen = { 1 }
kill = { 3, 4, 5, 7 }

OutKILL = { 3, 4, 5, 7 } \cup { }

$d = e + f$ 2
gen = { 2 }
kill = { 5, 7 }

$f = a + c$ 3
gen = { 3 }
kill = { 2, 6 }

Aggregate Kill set

$\text{InKILL} = \{ \}$

| | |
|-----------------------|---|
| $a = b + c$ | 1 |
| gen = { 1 } | |
| kill = { 3, 4, 5, 7 } | |

$\text{OutKILL} = \{ 3, 4, 5, 7 \}$

| | |
|-----------------|---|
| $d = e + f$ | 2 |
| gen = { 2 } | |
| kill = { 5, 7 } | |

| | |
|-----------------|---|
| $f = a + c$ | 3 |
| gen = { 3 } | |
| kill = { 2, 6 } | |

Aggregate Kill set

| | |
|-----------------------|---|
| $a = b + c$ | 1 |
| gen = { 1 } | |
| kill = { 3, 4, 5, 7 } | |

$\text{InKILL} = \{ 3, 4, 5, 7 \}$

| | |
|-----------------|---|
| $d = e + f$ | 2 |
| gen = { 2 } | |
| kill = { 5, 7 } | |

$\text{OutKILL} = \text{kill} \cup \text{InKILL}$

| | |
|-----------------|---|
| $f = a + c$ | 3 |
| gen = { 3 } | |
| kill = { 2, 6 } | |

Aggregate Kill set

| | |
|-----------------------|---|
| $a = b + c$ | 1 |
| gen = { 1 } | |
| kill = { 3, 4, 5, 7 } | |

$\text{InKILL} = \{ 3, 4, 5, 7 \}$

| | |
|-----------------|---|
| $d = e + f$ | 2 |
| gen = { 2 } | |
| kill = { 5, 7 } | |

$\text{OutKILL} = \{ 5, 7 \} \cup \{ 3, 4, 5, 7 \}$

| | |
|-----------------|---|
| $f = a + c$ | 3 |
| gen = { 3 } | |
| kill = { 2, 6 } | |

Aggregate Kill set

| | |
|-----------------------|---|
| $a = b + c$ | 1 |
| gen = { 1 } | |
| kill = { 3, 4, 5, 7 } | |

$\text{InKILL} = \{ 3, 4, 5, 7 \}$

| | |
|-----------------|---|
| $d = e + f$ | 2 |
| gen = { 2 } | |
| kill = { 5, 7 } | |

$\text{OutKILL} = \{ 3, 4, 5, 7 \}$

| | |
|-----------------|---|
| $f = a + c$ | 3 |
| gen = { 3 } | |
| kill = { 2, 6 } | |

Aggregate Kill set

| | |
|-----------------------|---|
| $a = b + c$ | 1 |
| gen = { 1 } | |
| kill = { 3, 4, 5, 7 } | |

| | |
|-----------------|---|
| $d = e + f$ | 2 |
| gen = { 2 } | |
| kill = { 5, 7 } | |

$\text{InKILL} = \{ 3, 4, 5, 7 \}$

| | |
|-----------------|---|
| $f = a + c$ | 3 |
| gen = { 3 } | |
| kill = { 2, 6 } | |

$\text{OutKILL} = \text{kill} \cup \text{InKILL}$

Aggregate Kill set

| | |
|-----------------------|---|
| $a = b + c$ | 1 |
| gen = { 1 } | |
| kill = { 3, 4, 5, 7 } | |

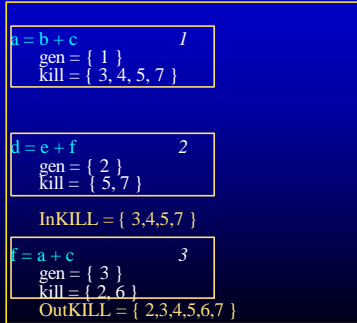
| | |
|-----------------|---|
| $d = e + f$ | 2 |
| gen = { 2 } | |
| kill = { 5, 7 } | |

$\text{InKILL} = \{ 3, 4, 5, 7 \}$

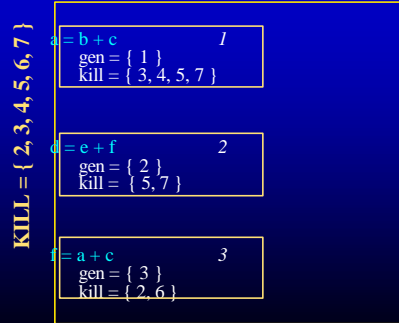
| | |
|-----------------|---|
| $f = a + c$ | 3 |
| gen = { 3 } | |
| kill = { 2, 6 } | |

$\text{OutKILL} = \{ 3, 4, 5, 7 \} \cup \{ 2, 6 \}$

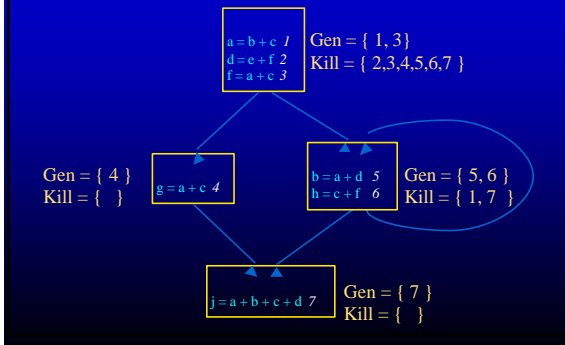
Aggregate Kill set



Aggregate Kill set



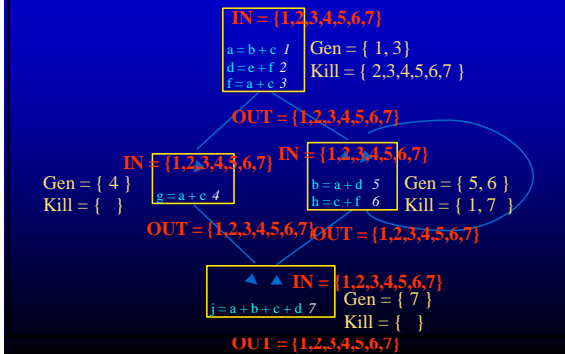
Aggregate Gen and Kill sets



Algorithm for Available Expression

- Assign a number to each expression
- Calculate gen and kill sets for each instruction
- Calculate aggregate gen and kill sets for each basic block
- Initialize available set at each basic block to be the entire set

Aggregate Gen and Kill sets



Algorithm for Available Expression

- Assign a number to each expression
- Calculate gen and kill sets for each instruction
- Calculate aggregate gen and kill sets for each basic block
- Initialize available set at each basic block to be the entire set
- Iteratively propagate available expression set over the CFG

Propagate available expression set

- If the expression is generated (in the gen set) then it is available at the end
 - should be in the OUT set

OUT = gen

Propagate available expression set

- If the expression is generated (in the gen set) then it is available at the end
 - should be in the OUT set
- Any expression available at the input (in the IN set) and not killed should be available at the end

OUT = gen \cup (IN - kill)

Propagate available expression set

- Expression is available only if it is available in all input paths

IN = \bigcap OUT

OUT = gen \cup (IN - kill)

Aggregate Gen and Kill sets

22

IN = \bigcap OUT
OUT = gen \cup (IN - kill)

Node 1: IN = {1,2,3,4,5,6,7}, Gen = {1,3}, Kill = {2,3,4,5,6,7}, OUT = {1,2,3,4,5,6,7}

Node 2: IN = {1,2,3,4,5,6,7}, Gen = {4}, Kill = {}, OUT = {1,2,3,4,5,6,7}

Node 3: IN = {1,2,3,4,5,6,7}, Gen = {5,6}, Kill = {1,7}, OUT = {1,2,3,4,5,6,7}

Node 4: IN = {1,2,3,4,5,6,7}, Gen = {7}, Kill = {}, OUT = {1,2,3,4,5,6,7}

Aggregate Gen and Kill sets

IN = \bigcap OUT
OUT = gen \cup (IN - kill)

Node 1: IN = {1,2,3,4,5,6,7}, Gen = {1,3}, Kill = {2,3,4,5,6,7}, OUT = {1,2,3,4,5,6,7}

Node 2: IN = {1,2,3,4,5,6,7}, Gen = {4}, Kill = {}, OUT = {1,2,3,4,5,6,7}

Node 3: IN = {1,2,3,4,5,6,7}, Gen = {5,6}, Kill = {1,7}, OUT = {1,2,3,4,5,6,7}

Node 4: IN = {1,2,3,4,5,6,7}, Gen = {7}, Kill = {}, OUT = {1,2,3,4,5,6,7}

Aggregate Gen and Kill sets

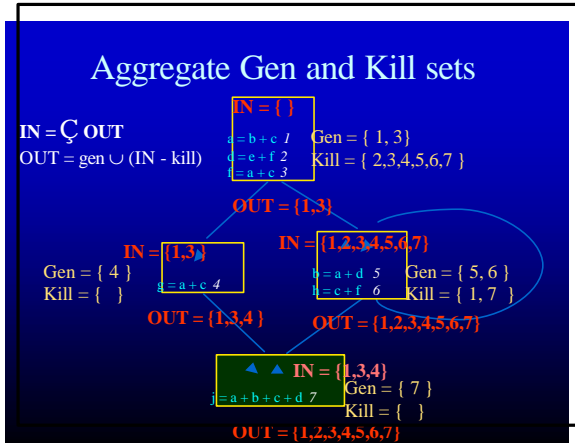
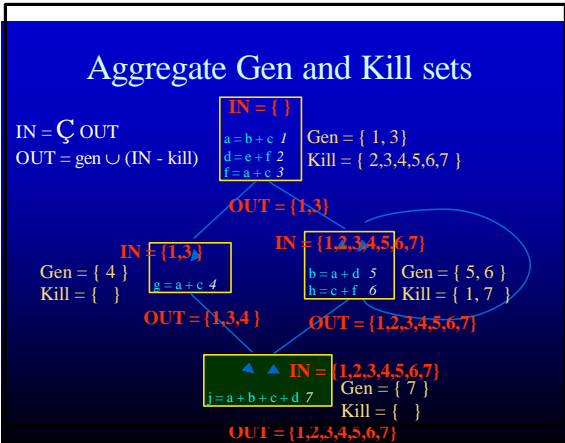
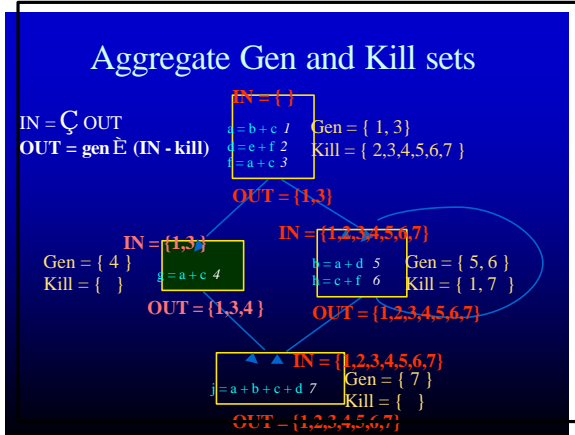
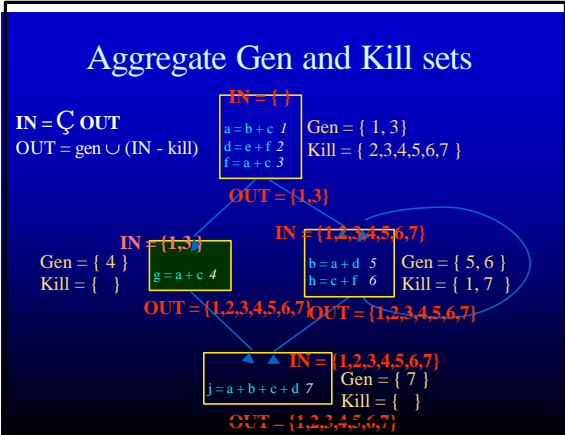
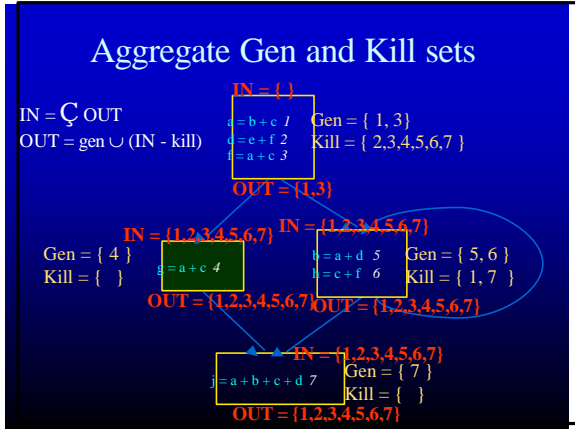
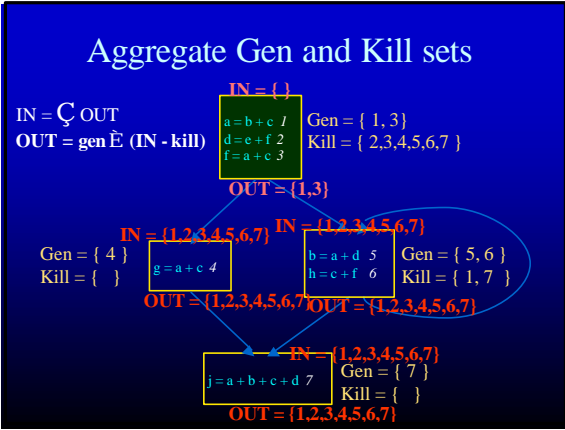
IN = \bigcap OUT
OUT = gen \cup (IN - kill)

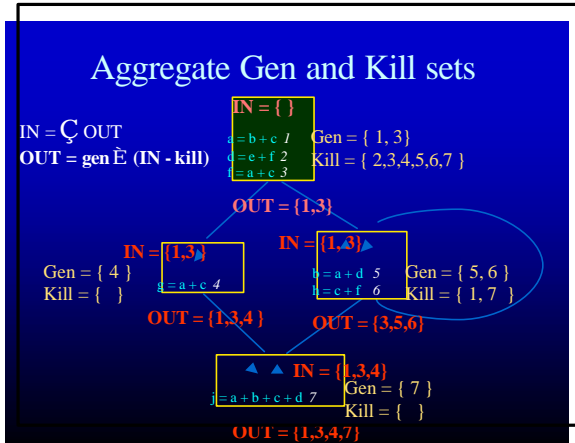
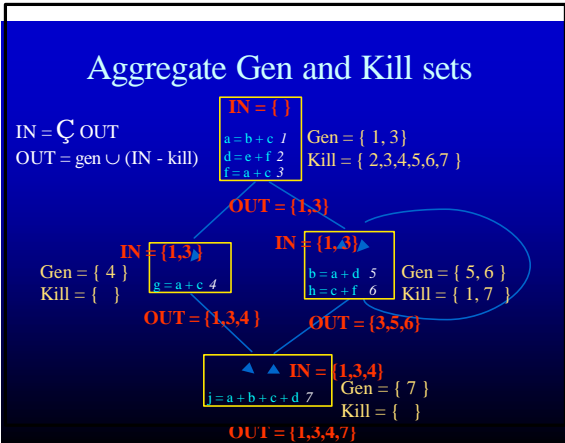
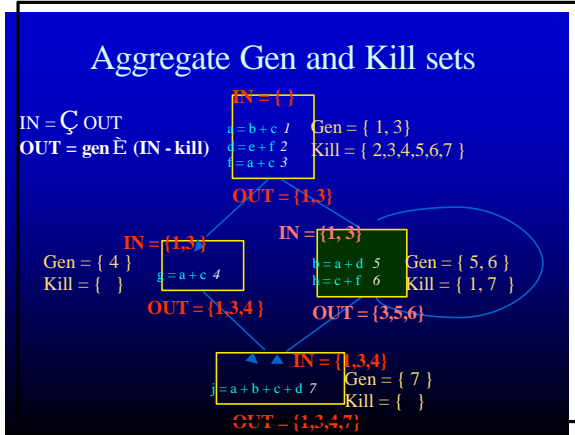
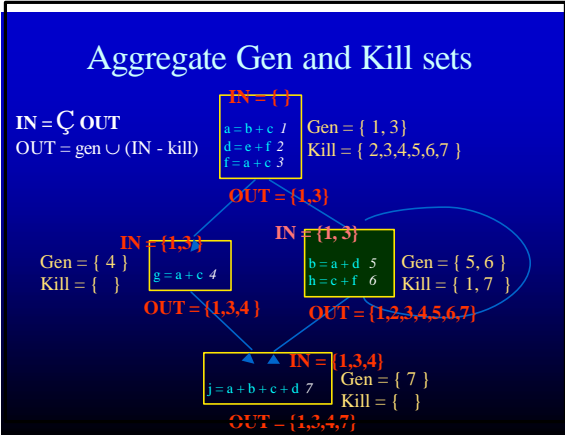
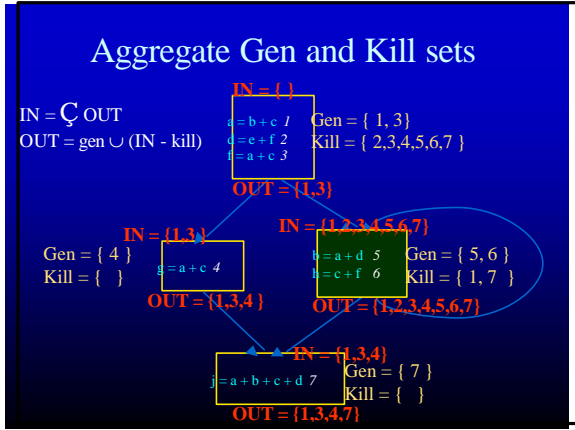
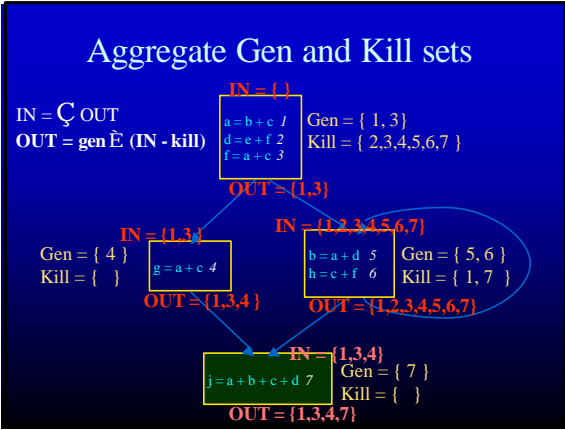
Node 1: IN = {}, Gen = {1,3}, Kill = {2,3,4,5,6,7}, OUT = {1,2,3,4,5,6,7}

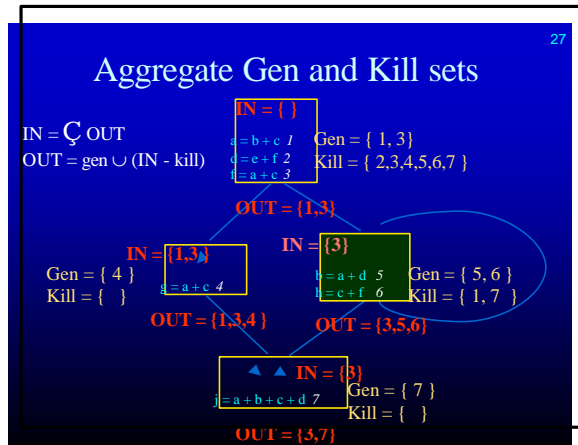
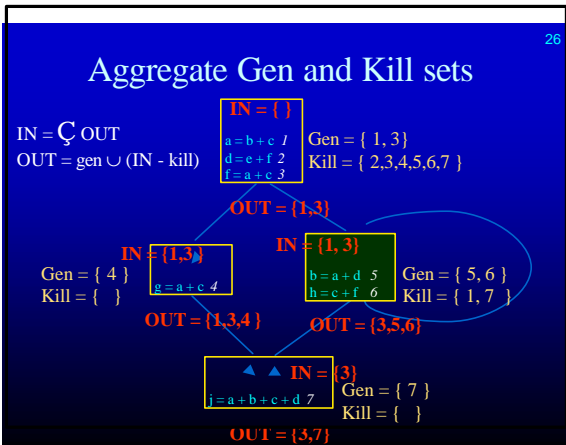
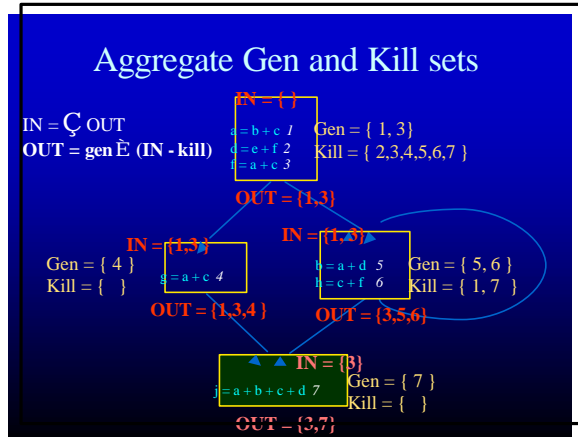
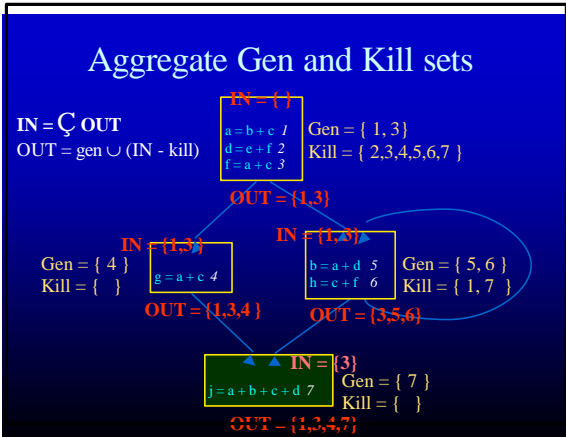
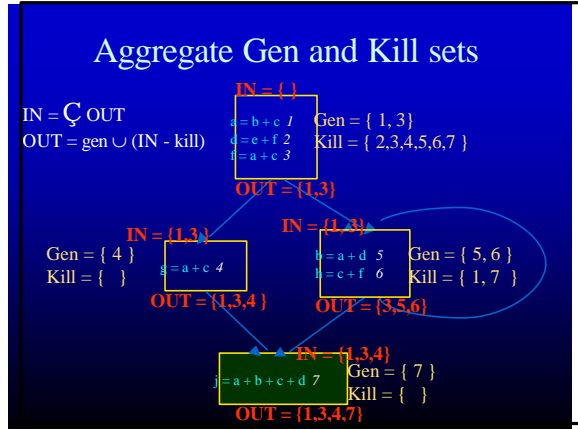
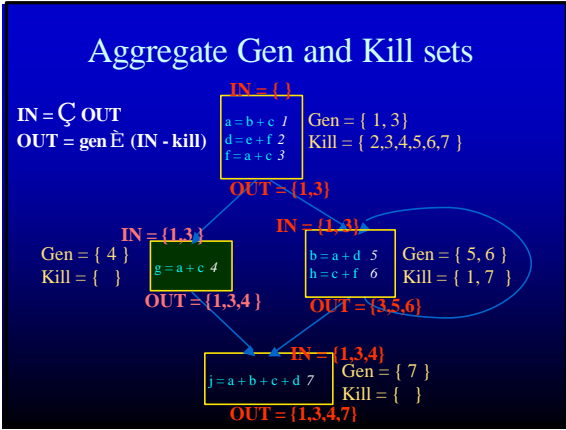
Node 2: IN = {1,2,3,4,5,6,7}, Gen = {4}, Kill = {}, OUT = {1,2,3,4,5,6,7}

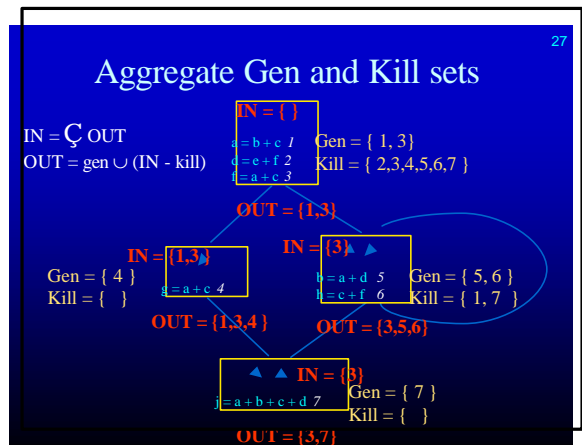
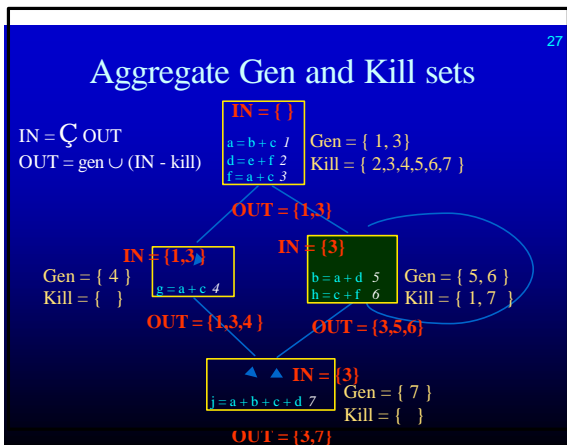
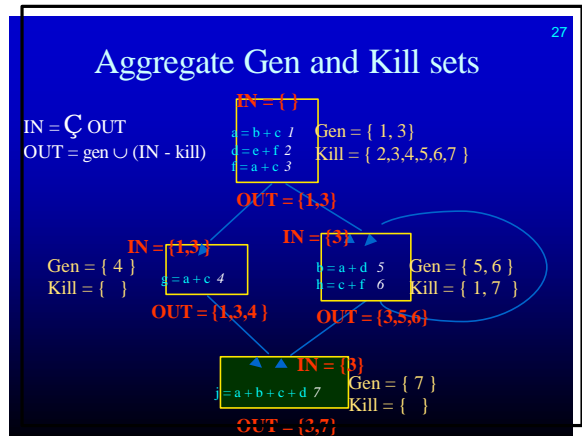
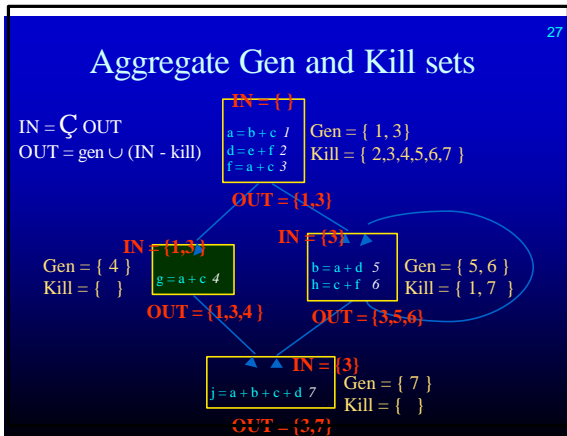
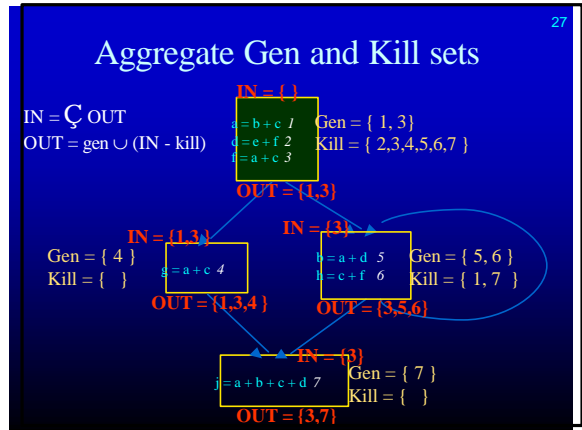
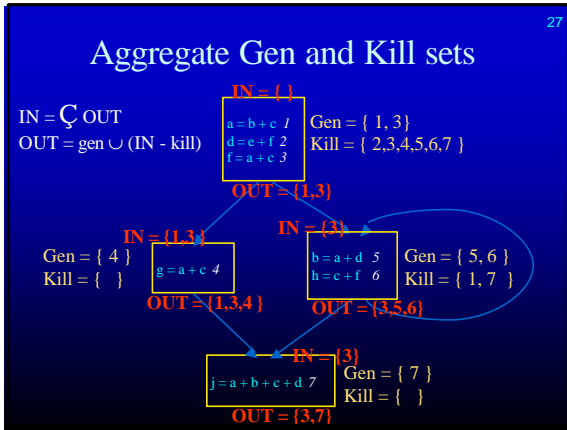
Node 3: IN = {1,2,3,4,5,6,7}, Gen = {5,6}, Kill = {1,7}, OUT = {1,2,3,4,5,6,7}

Node 4: IN = {1,2,3,4,5,6,7}, Gen = {7}, Kill = {}, OUT = {1,2,3,4,5,6,7}







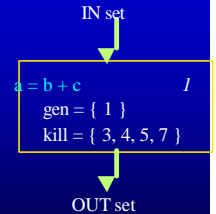


Algorithm for Available Expression 28

- Assign a number to each expression
- Calculate gen and kill sets for each instruction
- Calculate aggregate gen and kill sets for each basic block
- Initialize available set at each basic block to be the entire set
- Iteratively propagate available expression set over the CFG
- Propagate within the basic block

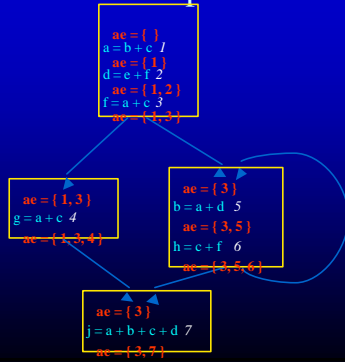
Propagate within the basic block

- Start with the IN set of available expressions
- Linearly propagate down the basic block
 - same as data-flow step
 - single pass since no back edges



$$\text{OUT} = \text{gen} \cup (\text{IN} - \text{kill})$$

Available Expressions



Outline 29

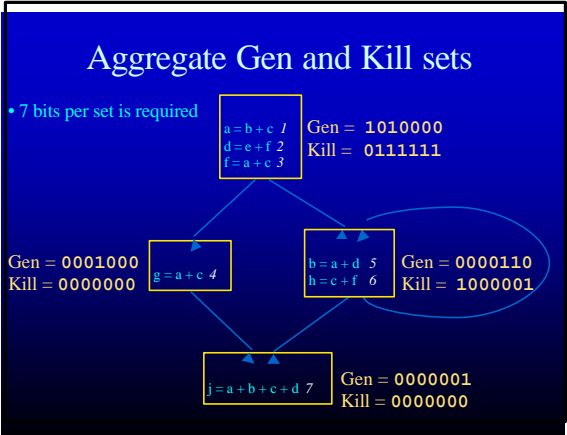
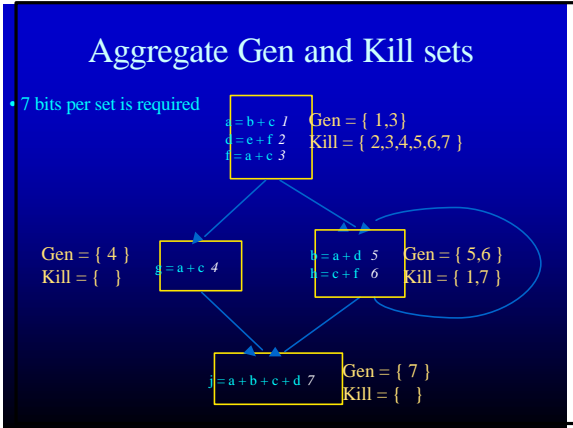
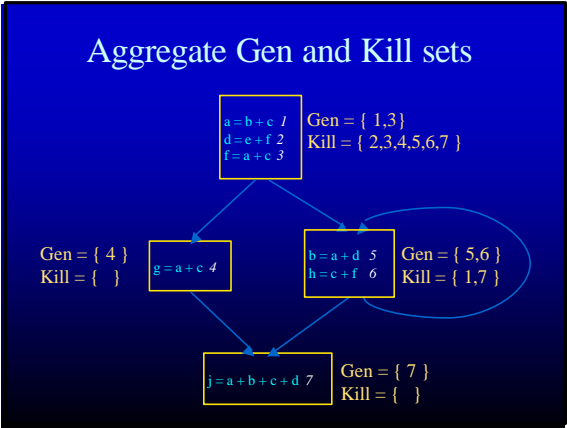
- Overview of data-flow analysis
- Available expressions
- Algorithm for calculating available expressions
- Bit sets
- Formulating a data-flow analysis problem
- DU chains
- SSA form

Bitsets

- Assign a bit to each element of the set
 - Union \Rightarrow bit OR
 - Intersection \Rightarrow bit AND
 - Subtraction \Rightarrow bit NEGATE and AND
- Fast implementation
 - 32 elements packed to each word
 - AND and OR are single instructions

Kill Set vs. Preserve Set

- Kill Sets
 - $\text{OUT} = \text{gen} \cup (\text{IN} - \text{kill})$
 - Using bit vectors: $\text{OUT} = \text{gen} \vee (\text{IN} - \text{kill})$
 - Subtraction \Rightarrow bit NEGATE and AND
 - $\text{OUT} = \text{gen} \vee (\text{IN} \wedge \neg \text{kill})$
 - Preserve Sets
 - $\text{PRSV} = \text{Entire Set} - \text{KILL}$
 - $\text{OUT} = \text{gen} \cup (\text{IN} \cap \text{prsv})$
 - $\text{OUT} = \text{gen} \vee (\text{IN} \wedge \text{prsv})$
- (Refer to Muchnick book)



Formulating a data-flow analysis problem

- Problem independent
 - calculate **gen** and **kill** sets for the basic block
 - iterative propagation of information until convergence
 - propagation of information within the basic block

Formulating a data-flow analysis problem

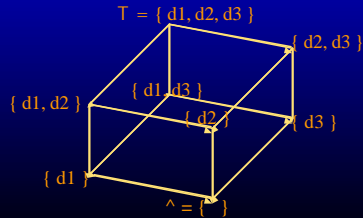
- Lattice
 - Abstract quantities over which the analysis will operate
Example: sets of available expressions
- Flow functions
 - how each control-flow and computational construct affects the abstract quantities
Example: the OUT equation for each statement

Lattice

- A lattice L consists of
 - a set of values
 - two operations meet ($\hat{\cup}$) and join ($\hat{\cup}$)
 - a top value (\top) and a bottom value ($\hat{\cup}$)

Lattice

- Example: the lattice for the reaching definition problem when there are only 3 definitions

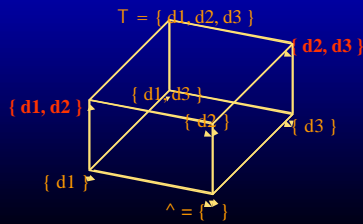


Meet and Join Operations

- Meet and Join forms a closure
 - For all $a, b \in L$ there exist a unique c and $d \in L$ such that
 - $a \hat{\cup} b = c$
 - $a \hat{\cap} b = d$
- Meet and Join are commutative
 - $a \hat{\cup} b = b \hat{\cup} a$
 - $a \hat{\cap} b = b \hat{\cap} a$
- Meet and Join are associative
 - $(a \hat{\cup} b) \hat{\cup} c = b \hat{\cup} (a \hat{\cup} c)$
 - $(a \hat{\cap} b) \hat{\cap} c = b \hat{\cap} (a \hat{\cap} c)$
- There exist a unique top element (\top) and bottom element (\wedge) in L such that
 - $a \hat{\cup} \wedge = a$
 - $a \hat{\cap} \top = a$

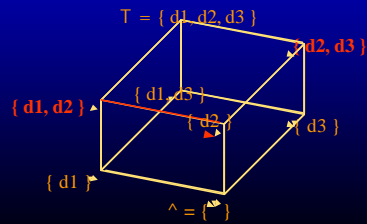
Meet and Join Operations

$$\{d1, d2\} \hat{\cup} \{d2, d3\} = ???$$



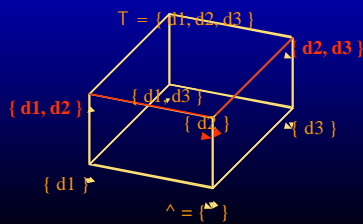
Meet and Join Operations

$$\{d1, d2\} \hat{\cap} \{d2, d3\} = ???$$



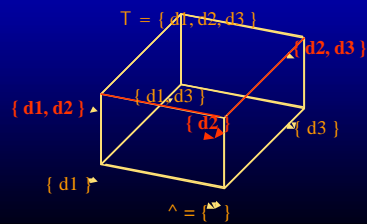
Meet and Join Operations

$$\{d1, d2\} \hat{\cup} \{d2, d3\} = ???$$



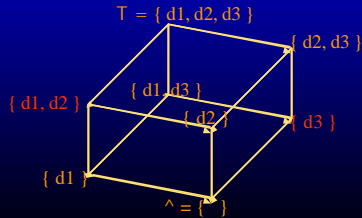
Meet and Join Operations

$$\{d1, d2\} \hat{\cup} \{d2, d3\} = \{d2\}$$



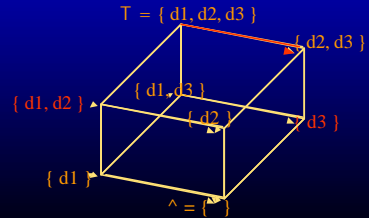
Meet and Join Operations

$$\{d1, d2\} \cup \{d3\} = ???$$



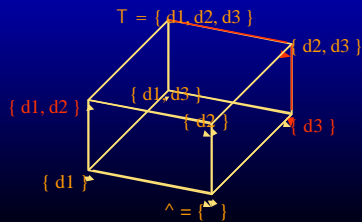
Meet and Join Operations

$$\{d1, d2\} \cup \{d3\} = ???$$



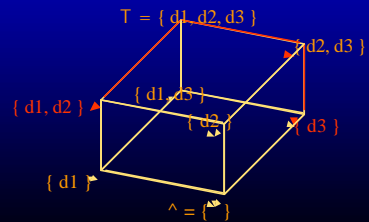
Meet and Join Operations

$$\{d1, d2\} \cup \{d3\} = ???$$



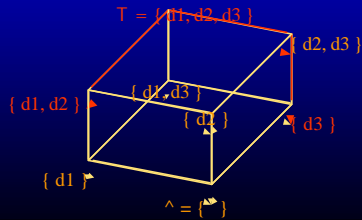
Meet and Join Operations

$$\{d1, d2\} \cup \{d3\} = ???$$



Meet and Join Operations

$$\{d1, d2\} \cup \{d3\} = \{d1, d2, d3\}$$



Meet and Join Operations

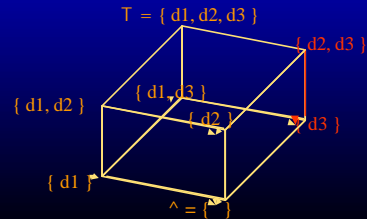
- Meet Operation
 - Set Intersection
 - Follow the lines downwards from the two elements in the lattice until they meet at a single unique element
- Join Operation
 - Set Union
 - There is a unique element in the lattice from where there is a downwards path (with no shared segment) to both elements

Partial Order

- Define $a \subseteq b$ if and only if $a \wedge b = a$
- Properties
 - Reflexive: $a \subseteq a$
 - Antisymmetric: $a \subseteq b$ and $b \subseteq a \Rightarrow a = b$
 - Transitive: $a \subseteq b$ and $b \subseteq c \Rightarrow a \subseteq c$

Partial Order

- Define $a \subseteq b$ if and only if $a \wedge b = a$
- Properties
 - $a \subseteq b \Rightarrow$ there exist a path from b to a

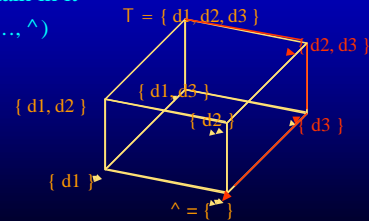


Lattice Height

- The height of the lattice is the longest ascending chain in it
 - $(T, a, b, c, \dots, \wedge)$

Lattice Height

- The height of the lattice is the longest ascending chain in it
 - $(T, a, b, c, \dots, \wedge)$



– Height is $(T, \{d2, d3\}, \{d3\}, \wedge) = 4$

Flow Functions

- Example: $OUT = f(IN)$
- $f: L \rightarrow L$ where L is a lattice
- Properties
 - Monotone: $\forall a, b \in L \quad a \subseteq b \Rightarrow f(a) \subseteq f(b)$
- Fixed Point
 - A fixed point is an element $a \in L$ such that $f(a) = a$

Intuition about Termination

- Data-flow analysis starts assuming most optimistic values (T)
- Each stage applies a flow function
 - $V_{new} \subseteq V_{prev}$
 - Moves downwards in the lattice
- Until stable (values don't change)
 - A fixed point is reached at every basic block
- Lattice has a finite height \Rightarrow should terminate

40

Outline

- Overview of data-flow analysis
- Available expressions
- Algorithm for calculating available expressions
- Bit sets
- Formulating a data-flow analysis problem
- **DU chains**
- **SSA form**

Def-Use and Use-Def Chains

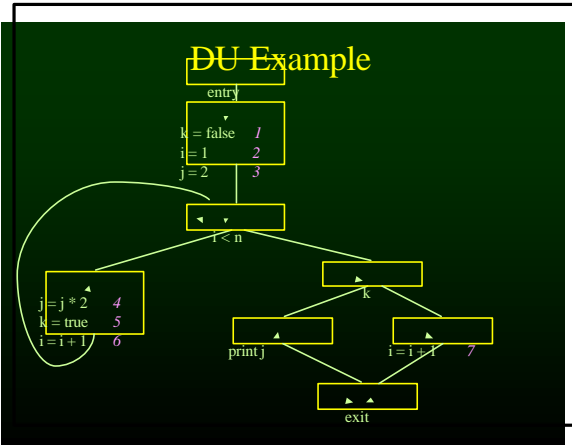
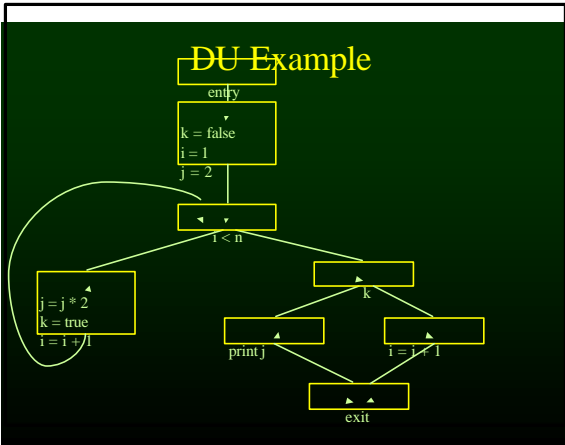
- **Def-Use (DU) Chain**
 - Connects a definition of each variable to all the possible uses of that variable
- **Use-Def (UD) Chain**
 - Connects a use of a variable to all the possible definitions of that variable

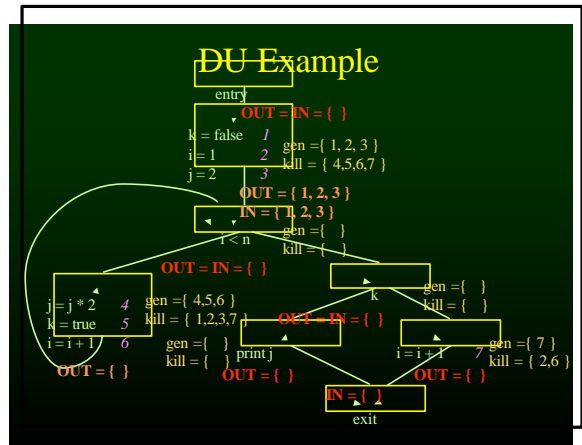
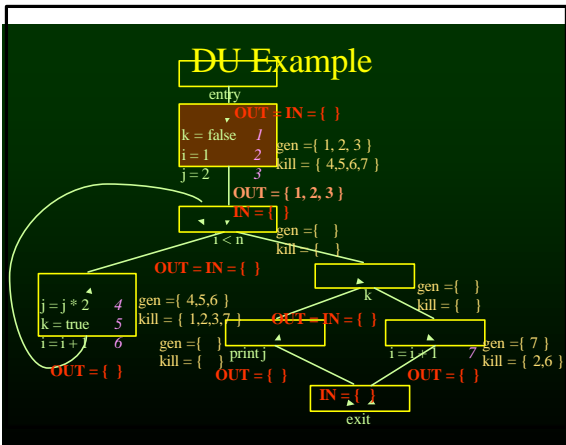
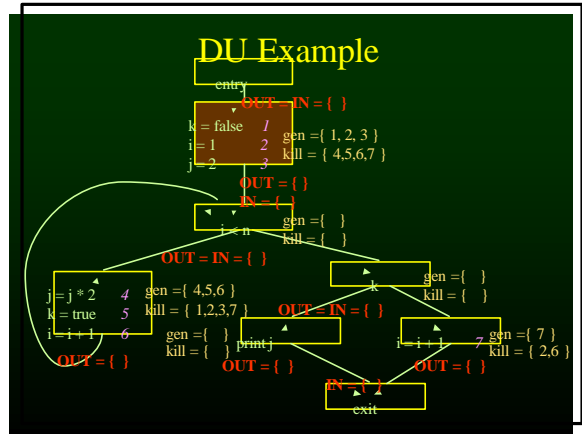
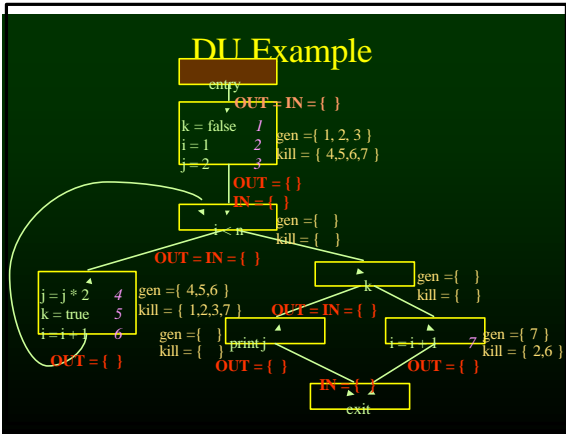
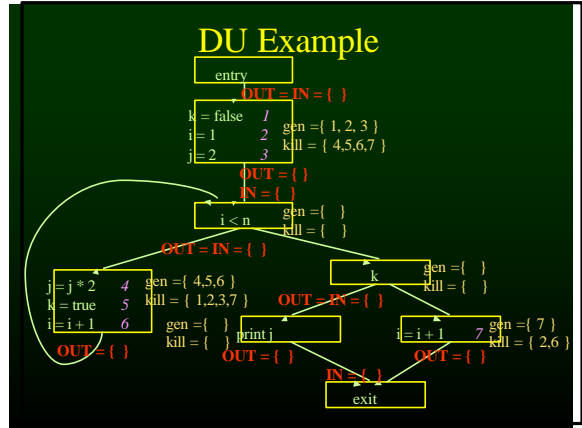
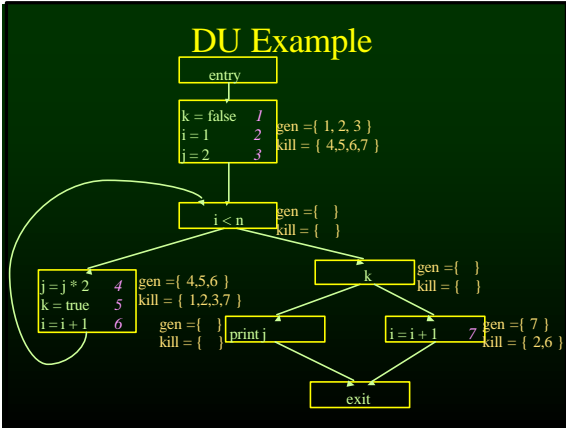
DU Chain Data-Flow Problem Formulation

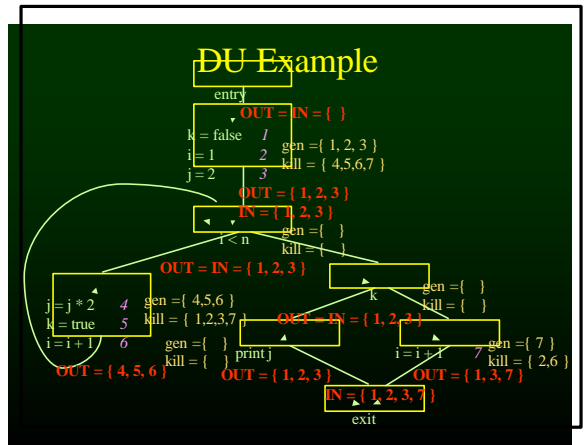
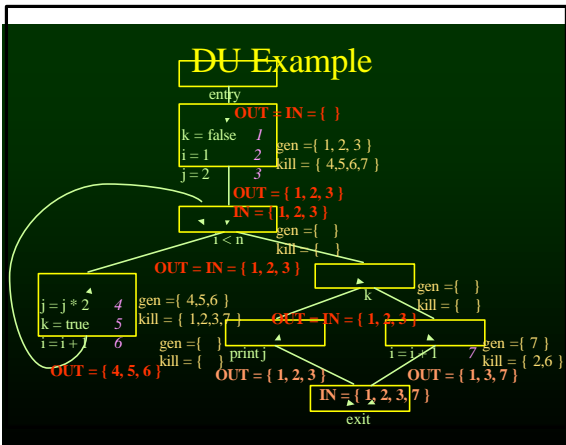
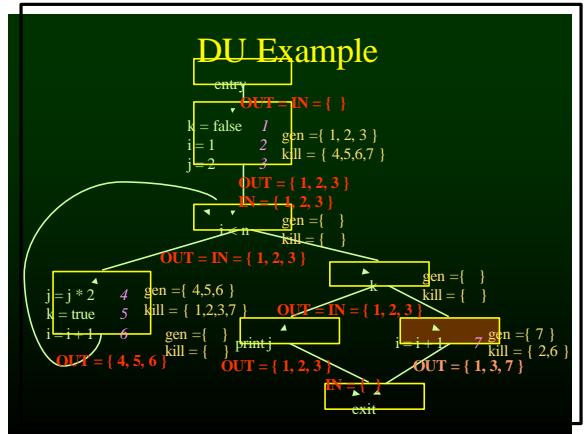
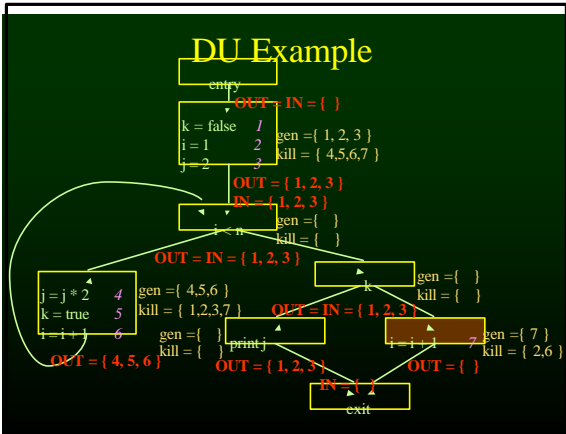
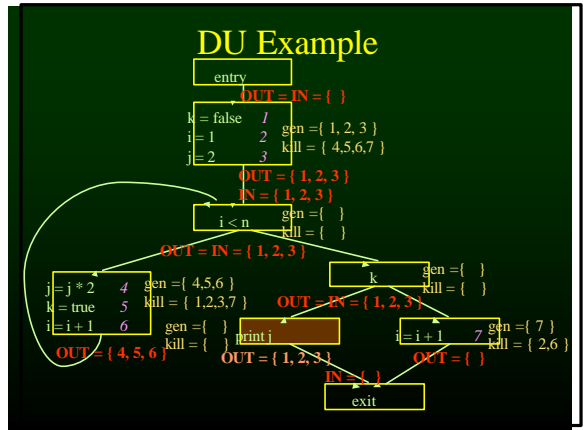
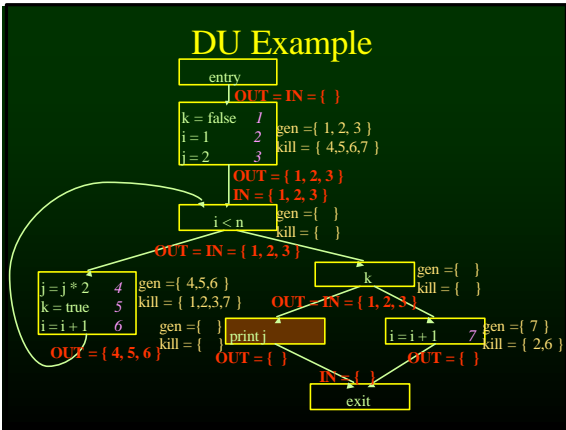
- **Lattice:** The set of definitions
 - Bitvector format: a bit for each definition in the procedure
- **Flow direction:** Forward Flow
- **Flow Functions:**
 - $gen = \{ b_0 \dots b_n \mid b_k = 1 \text{ iff the } k\text{th definition} \}$
 - $kill = \{ b_0 \dots b_n \mid b_k = 1 \text{ iff } k\text{th variable is redefined} \}$
 - $OUT = gen \cup (IN - kill)$
 - $IN = \vec{E} \cup OUT$

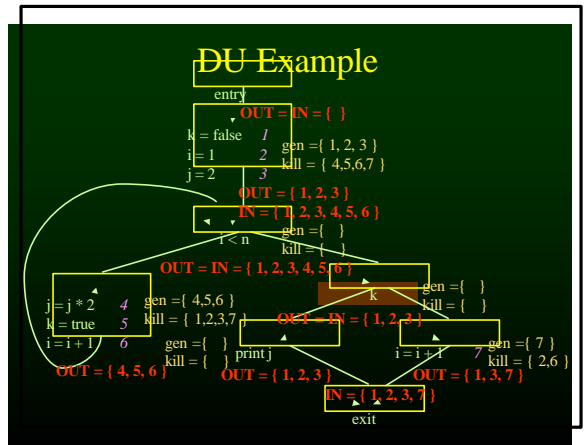
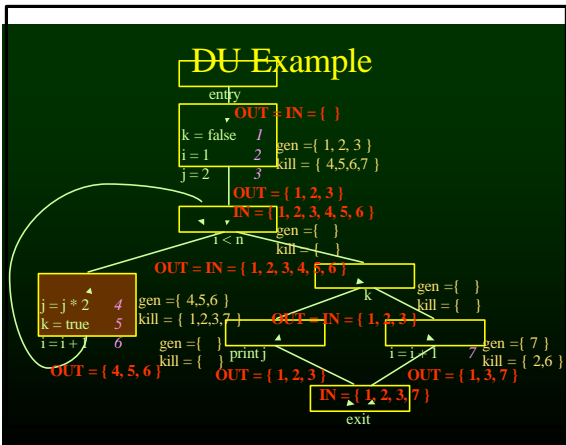
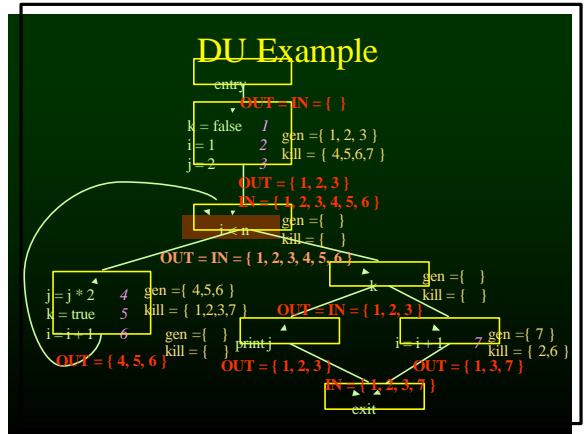
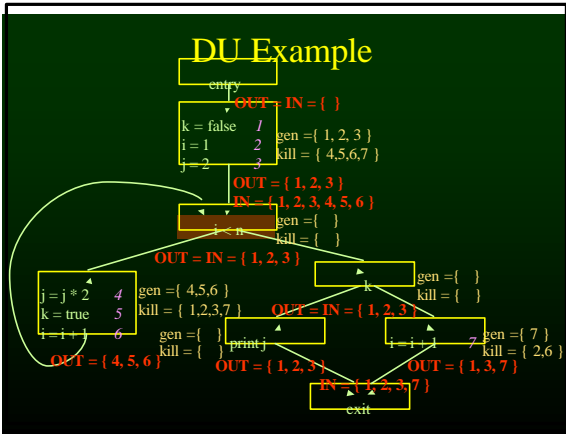
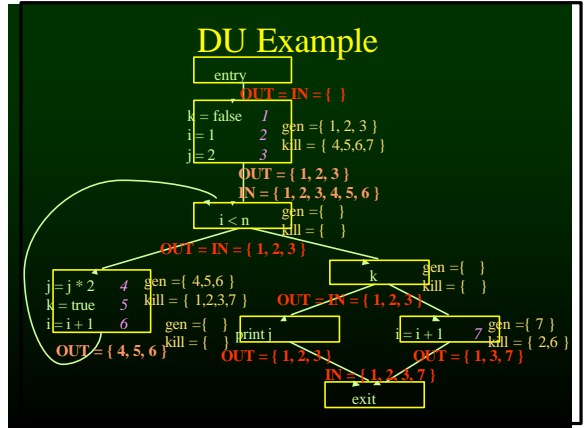
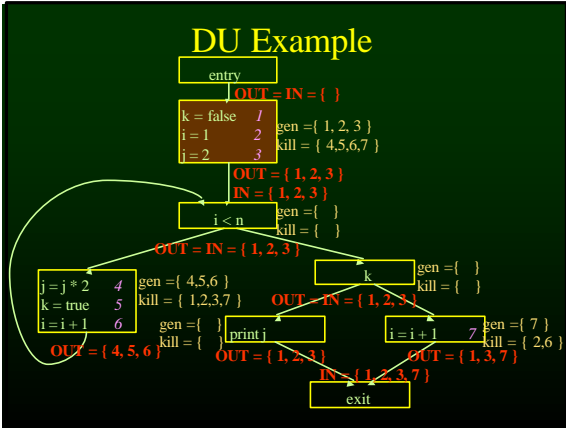
Formulate the UD Chain Data-Flow Problem

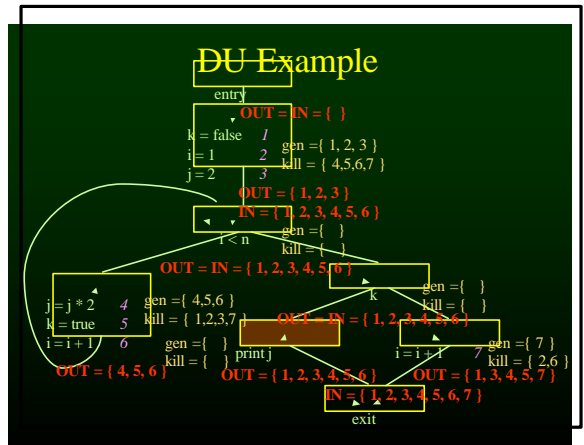
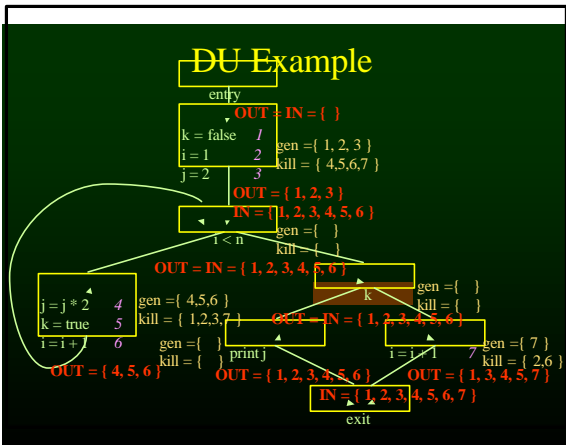
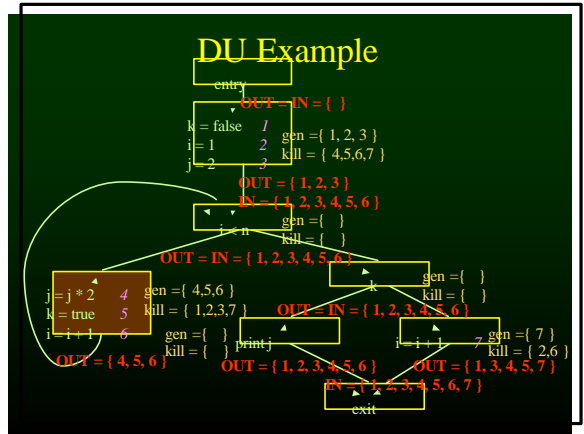
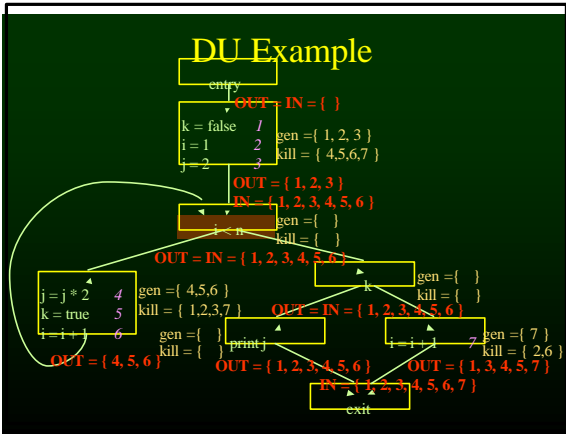
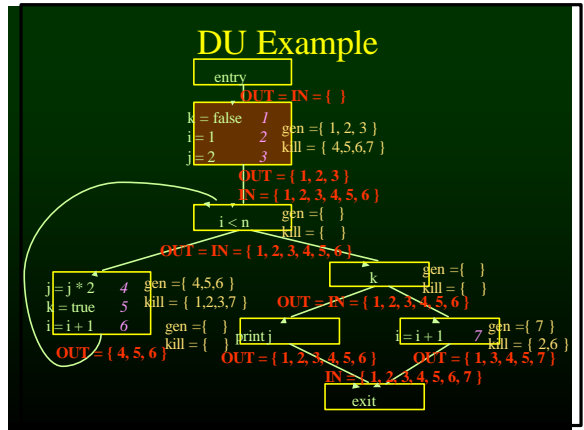
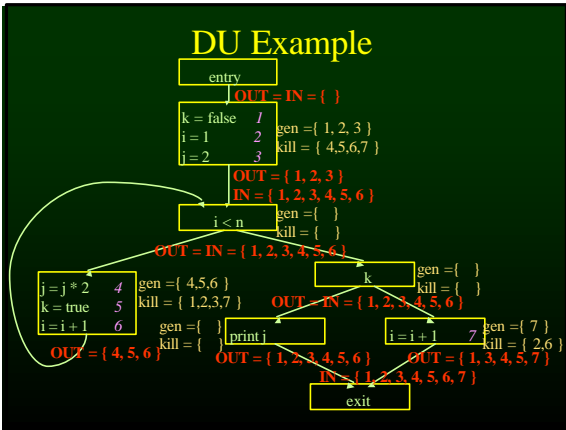
- **Lattice:**
 - Bitvector format:
- **Flow direction:** Forward/Backward Flow
- **Flow Functions:**
 - $gen = \{ b_0 \dots b_n \mid b_k = 1 \}$
 - $kill = \{ b_0 \dots b_n \mid b_k = 1 \}$
 - $OUT =$
 - $IN =$

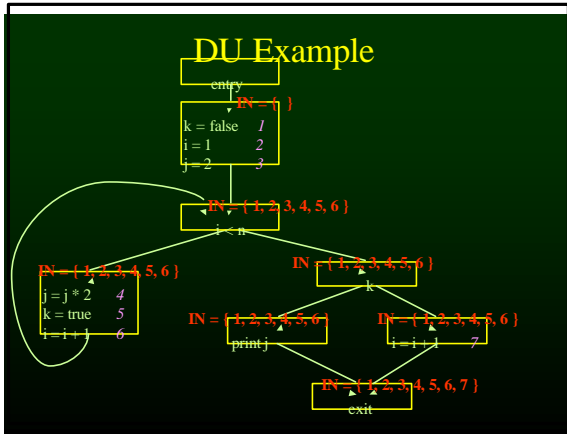
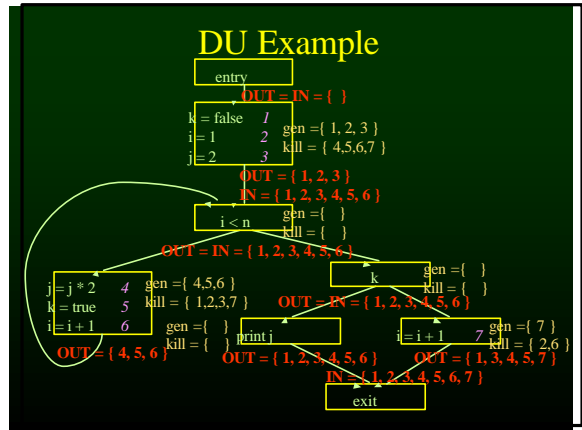
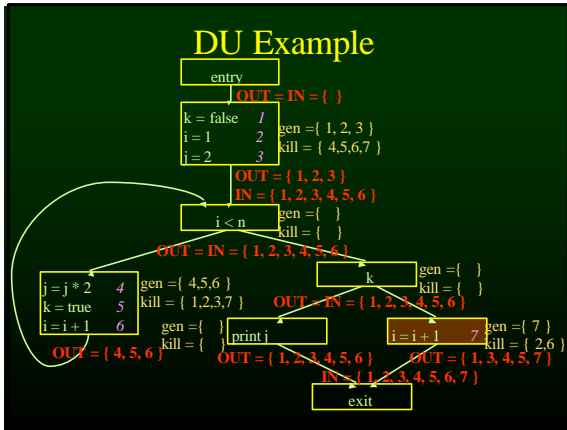












- ### DU Chains
- At each use of a variable, points to all the possible definitions
 - Very useful information
 - Used in many optimizations
 - Incorporate this information in the representation
 - SSA Form

- ### Outline
- Overview of control-flow analysis
 - Available expressions
 - Algorithm for calculating available expressions
 - Bit sets
 - Formulating a data-flow analysis problem
 - DU chains
 - **SSA form**

- ### Static Single Assignment (SSA) Form
- Each definition has a unique variable name
 - Original name + a version number
 - Each use refers to a definition by name
 - What about multiple possible definitions?
 - Add special merge nodes so that there can be only a single definition (Φ functions)

Static Single Assignment (SSA) Form

```

a = 1
b = a + 2
c = a + b
a = a + 1
d = a + b
    
```

Static Single Assignment (SSA) Form

```

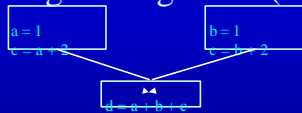
a1 = 1
b1 = a1 + 2
c1 = a1 + b1
a2 = a1 + 1
c2 = a2 + b1
    
```



```

a1 = 1
b1 = a1 + 2
c1 = a1 + b1
a2 = a1 + 1
c2 = a2 + b1
    
```

Static Single Assignment (SSA) Form



Static Single Assignment (SSA) Form

