Finding Dead Code

- Easy answer: if the variable being window
 dead
 - Intuition: the value you are gene generating this value is pointless

1:
$$A = B + C$$

2: $C = A + B$
3: $T1 = A + B$
4: $D = T1 + C$
5: $T2 = D + T1$
6: $D = A + B$
7: WRITE(D)

Easy answer: if the variable being written by an instruction is not live, the code is

- Easy answer: if the variable being window
 dead
 - Intuition: the value you are gene generating this value is pointless

1:
$$A = B + C$$

2: $C = A + B$
3: $T1 = A + B$
4: $D = T1 + C$
5: $T2 = D + T1$
6: $D = A + B$
7: WRITE(D)

Easy answer: if the variable being written by an instruction is not live, the code is

- Easy answer: if the variable being window
 dead
 - Intuition: the value you are gene generating this value is pointless

1:
$$A = B + C$$

2: $C = A + B$
3: $T1 = A + B$
4: $D = T1 + C$
5: $T2 = D + T1$
6: $D = A + B$
7: WRITE(D)

Easy answer: if the variable being written by an instruction is not live, the code is

- Easy answer: if the variable being window
 dead
 - Intuition: the value you are gene generating this value is pointless

1:
$$A = B + C$$

2: $C = A + B$
3: $T1 = A + B$
4: $D = T1 + C$
5: $T2 = D + T1$
6: $D = A + B$
7: WRITE(D)

Easy answer: if the variable being written by an instruction is not live, the code is

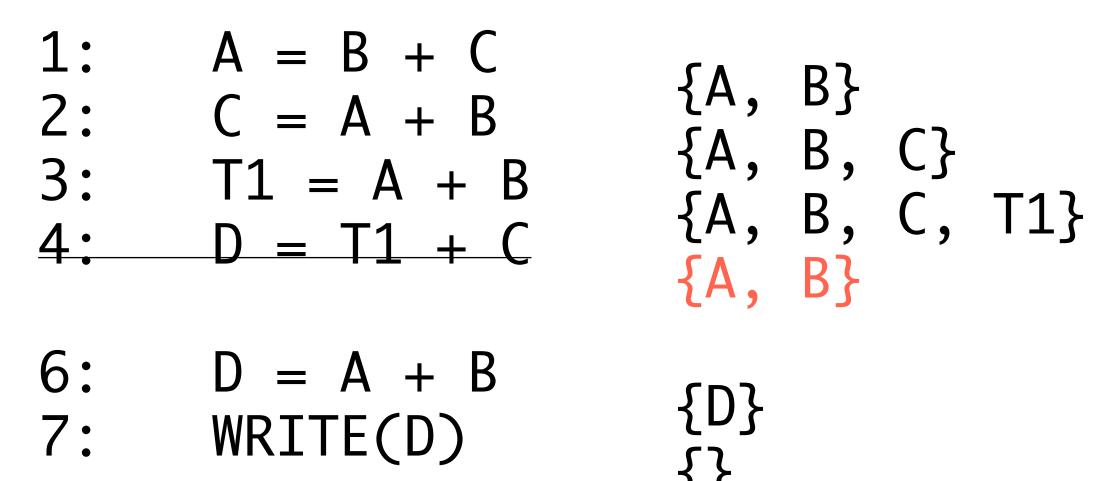
- After you remove dead code, it changes liveness information
- Recompute and iterate

1:
$$A = B + C$$

2: $C = A + B$
3: $T1 = A + B$
4: $D = T1 + C$
6: $D = A + B$
7: WRITE(D)

{A, B}
{A, B, C}
{A, B, C, T1} $\{A, B\}$ {D} {}

- After you remove dead code, it changes liveness information
- Recompute and iterate



 $\{A, B\}$ {D} {}

- After you remove dead code, it changes liveness information
- Recompute and iterate

1:
$$A = B + C$$

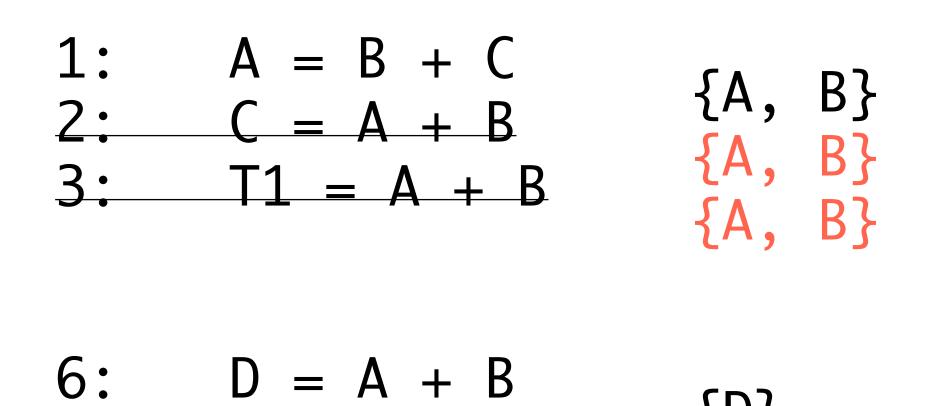
2: $C = A + B$
3: $T1 = A + B$

6: D = A + B7: WRITE(D)

{A, B}
{A, B}
{A, B}

{D} {}

- After you remove dead code, it changes liveness information
- Recompute and iterate



7: WRITE(D)

{D} {}

- After you remove dead code, it changes liveness information
- Recompute and iterate

1:
$$A = B + C$$

{A, B}

{D} {}

can we do this faster?

- Recomputing and iterating is slow!
- We can speed this up by computing **use-def** chains
 - Track how uses of variables are connected to definitions of those variables
- Can trace backwards from live code along use-def chains
 - Instruction is "backwards reachable" from live code \rightarrow instruction is live
 - Instruction is not backwards reachable \rightarrow no definition from this instruction eventually propagates to live code, instruction is dead
- This generalizes to a program analysis technique called program slicing

next: register allocation