Partial Dead Store Elimination

Zhixuan Chen • Fanghao Zhang • Mingye Chen • Yichen Zhong

EECS 583 Fall 2023

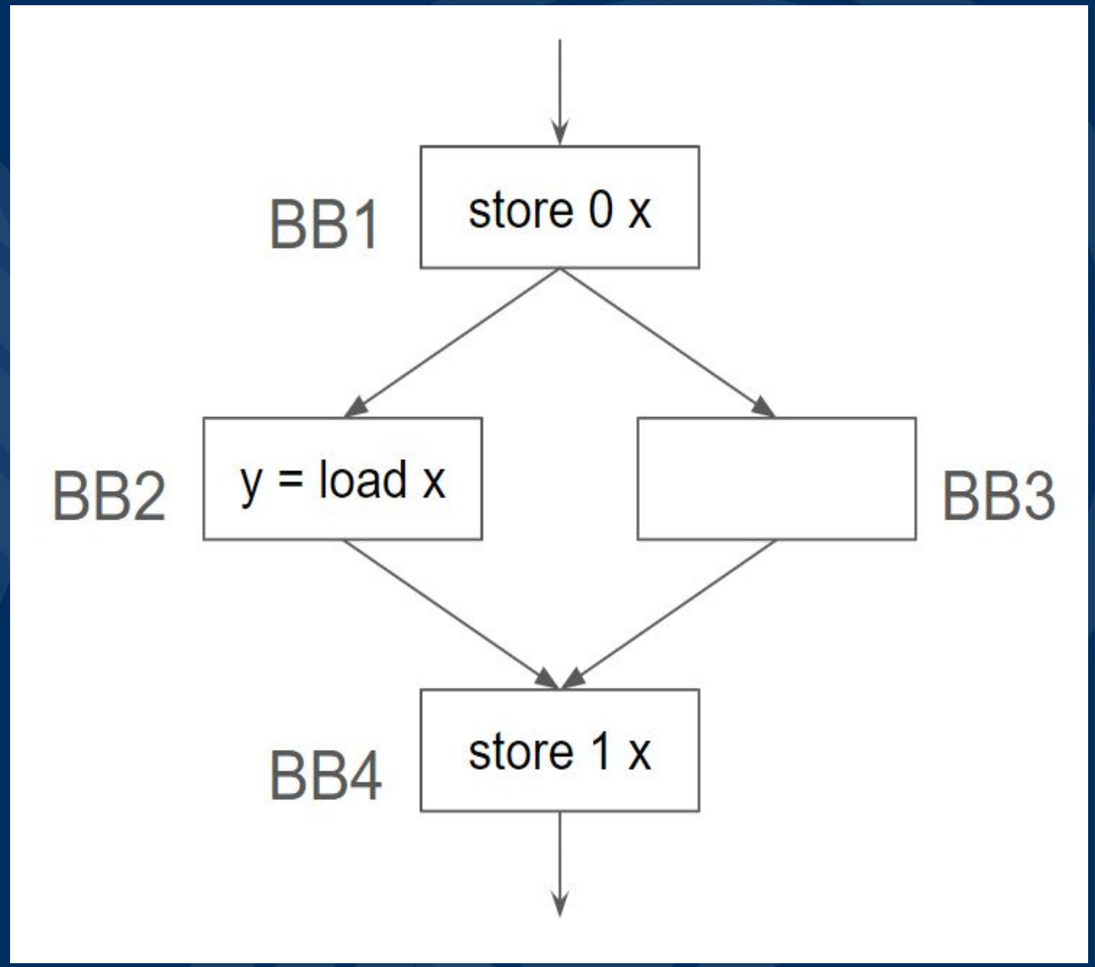
COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

Content

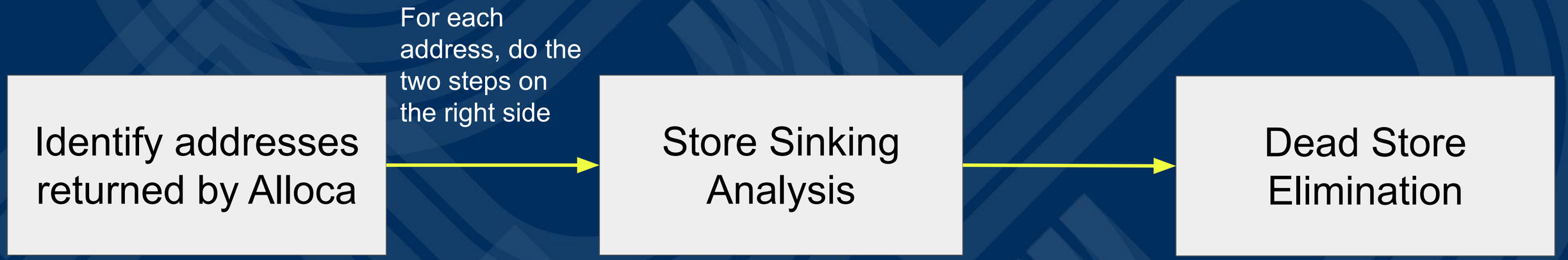
- Partial Dead Store Description (1 min)
- Overall Algorithm (1 min)
- Detailed breakdown of Algorithm:
  - GEN-KILL (1.5min)
  - IN-OUT (1.5 min)
  - Insertion Point (1min)
  - Fall-through Case (1min)
- Results (2.5min)
  - Show the CFG comparison (might use additional PDF)
- Limitation (0.5min)

Partial Dead Store

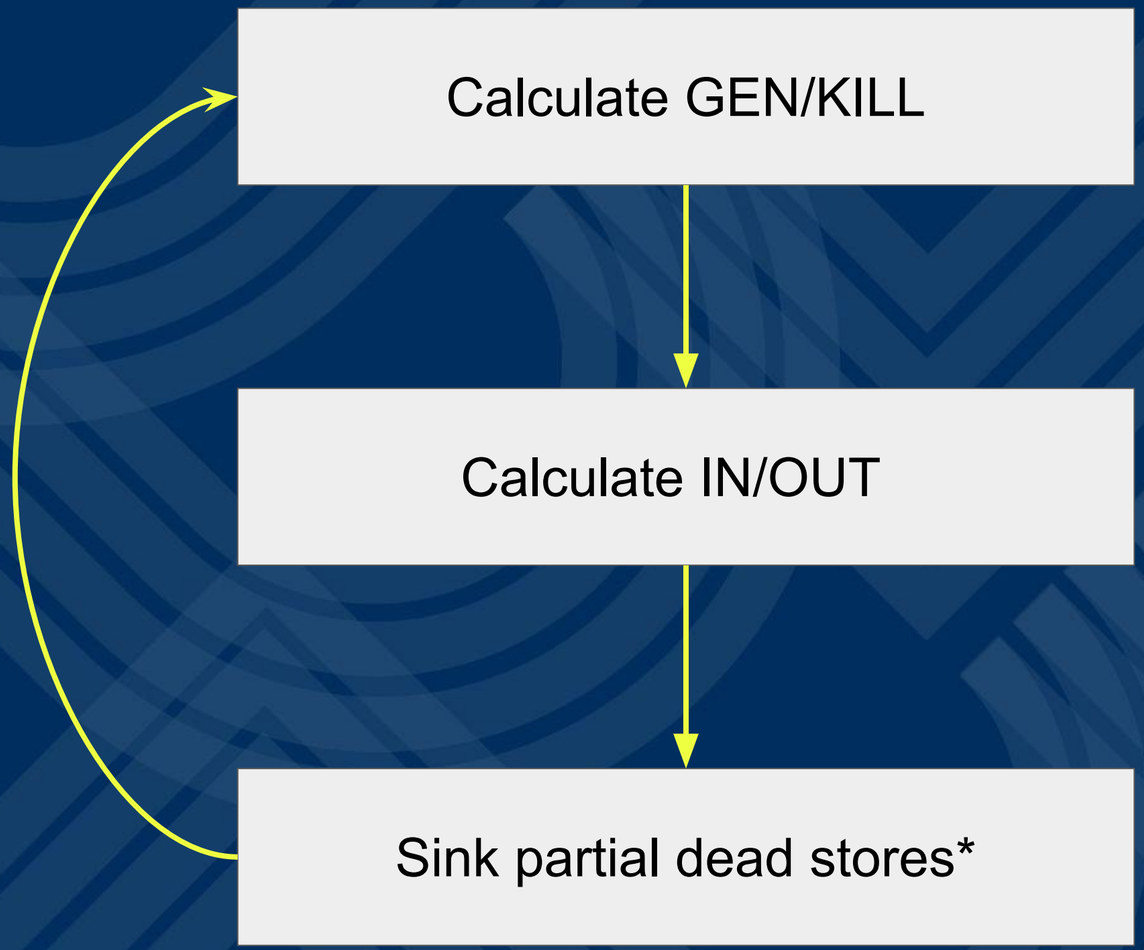
Informal definition: Store a value into an address but never load (thus never use) this value from that address later on **some** path



Algorithm Overview



Store Sinking Analysis



* The idea of code sinking is based on Jens Knoop, Oliver Rüthing, and Bernhard Steffen. 1994. Partial dead code elimination. SIGPLAN Not. 29, 6 (June 1994), 147–158.

Calculate GEN/KILL

Algorithm 1 Compute GEN/KILL w.r.t address *addr*

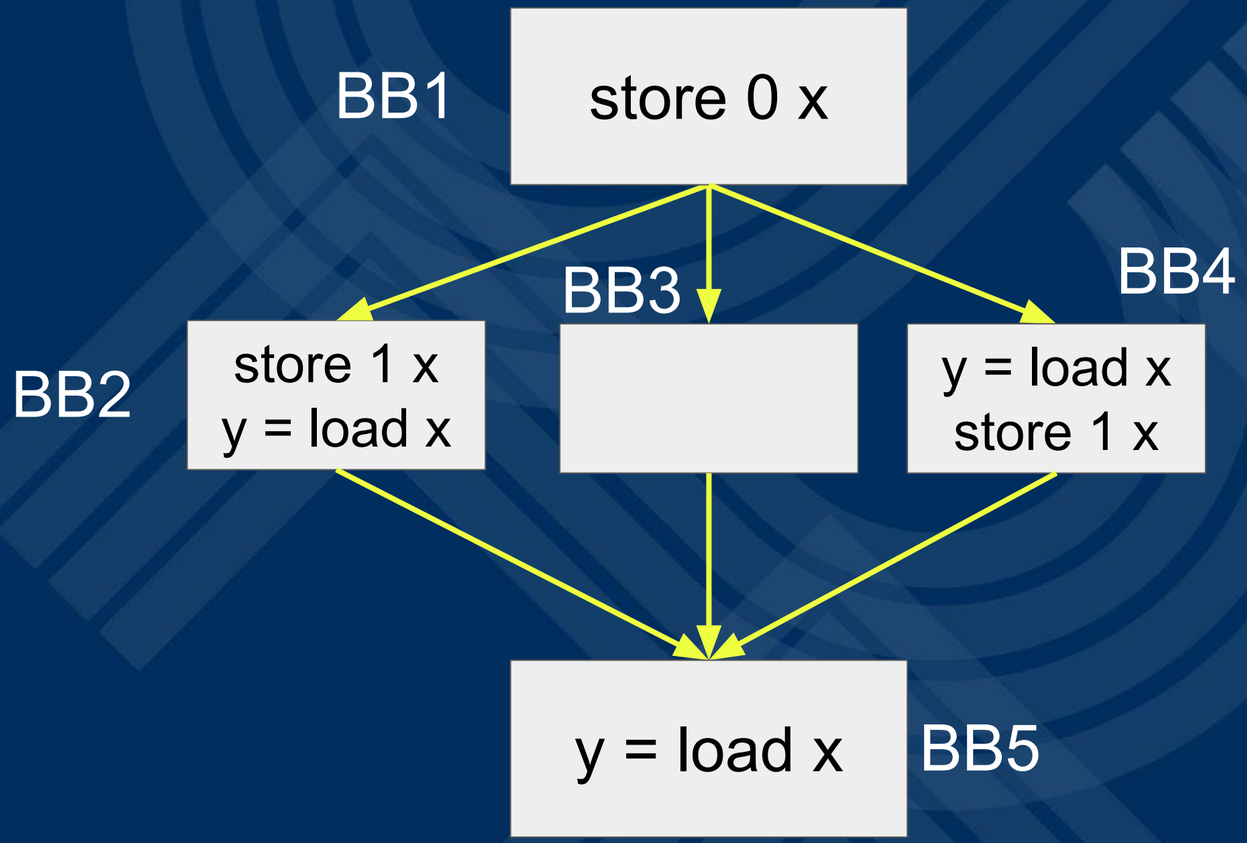
```
for Basic Block BB in Function F do
    GEN(BB), KILL(BB) ← None, False
    if fall-through case is detected then
        GEN(BB), KILL(BB) ← None, True
        continue
    end if

    for Instruction I in BB do
        if I is a store instruction w.r.t. addr then
            GEN(BB), KILL(BB) ← I, False
        else if I is a load instruction w.r.t. addr then
            GEN(BB), KILL(BB) ← None, True
        end if
    end for
end for
```

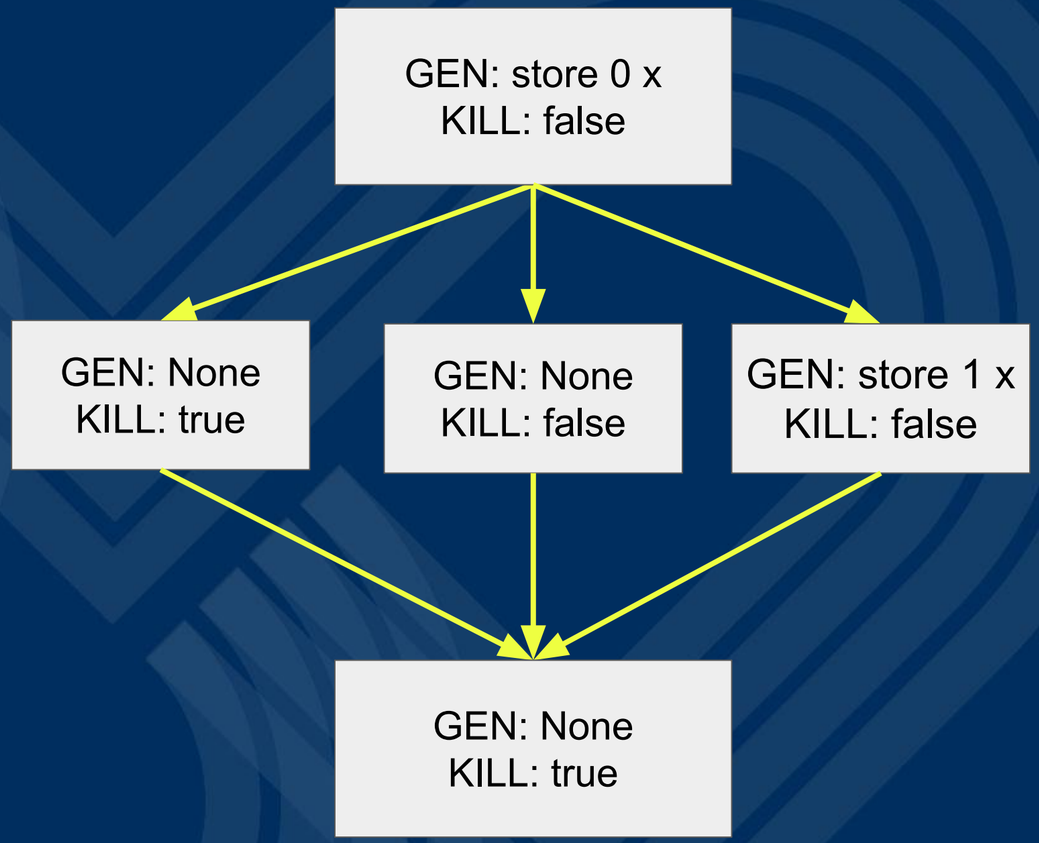
TL;DR:

GEN is to figure out whether there is a store in the current block that can be sunk downwards.

KILL is to figure out whether there is a load in the current block that blocks the sinking of stores.

Calculate GEN/KILL





Calculate IN/OUT

Algorithm 2 Compute IN/OUT w.r.t address *addr*

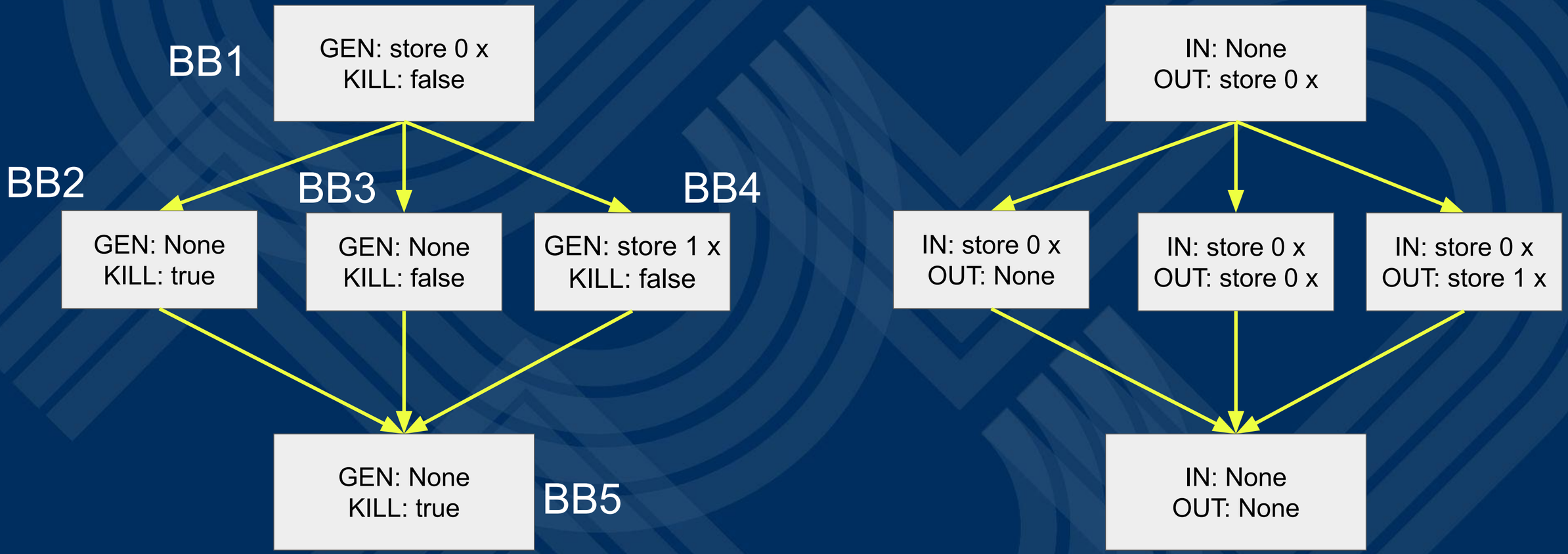
```
for Basic Block BB in Function F do
    IN(BB), OUT(BB) ← None, None
end for

change ← True
while change is True do
    for Basic Block BB in Function F do
        IN(BB) ← ⋂_{pred(BB)} OUT(pred)
        OUT ← None

        if GEN(BB) is not None then
            OUT(BB) ← GEN(BB)
        else if KILL(BB) is False then
            OUT(BB) ← IN(BB)
        end if

        change ← changed?
    end for
end while
```

TL;DR:

IN is to figure out which store can be sunk to the beginning of the current block.

OUT is to figure out which store can be sunk past the current block.

Calculate IN/OUT



Sink Partial Dead Stores

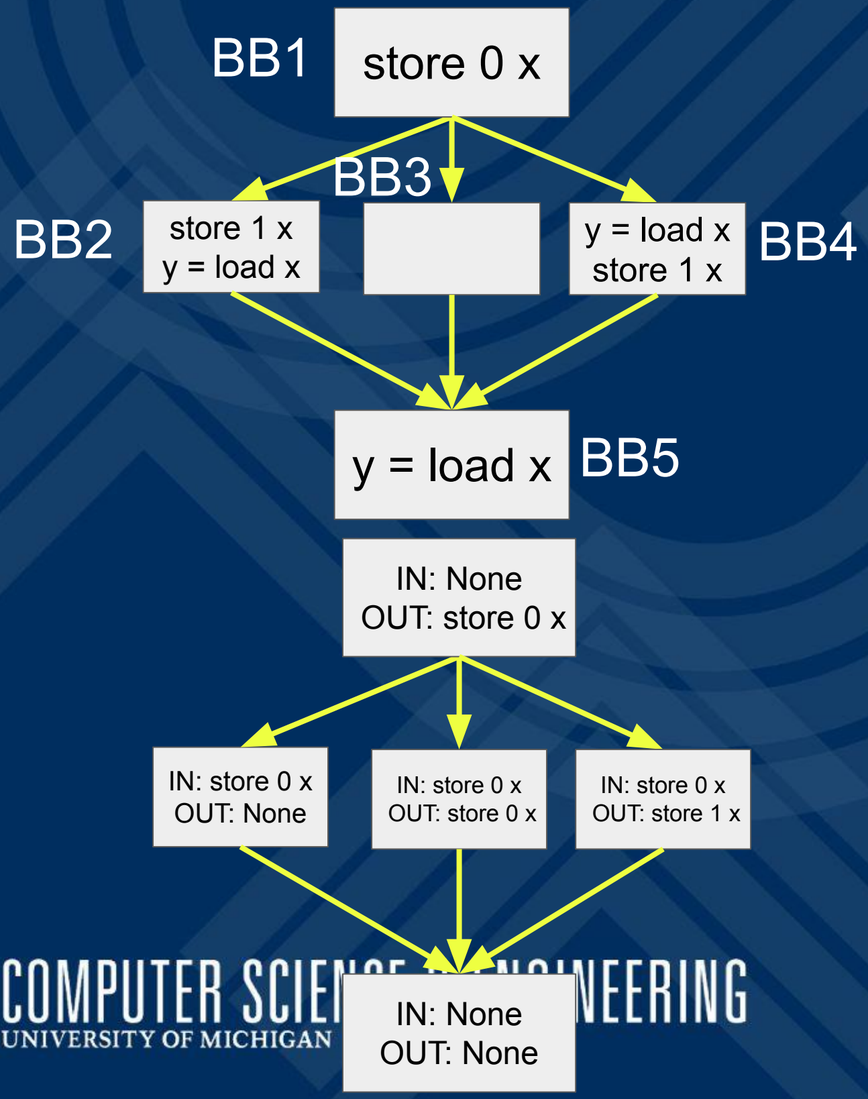
Algorithm 3 Sink partial dead stores w.r.t. address *addr*

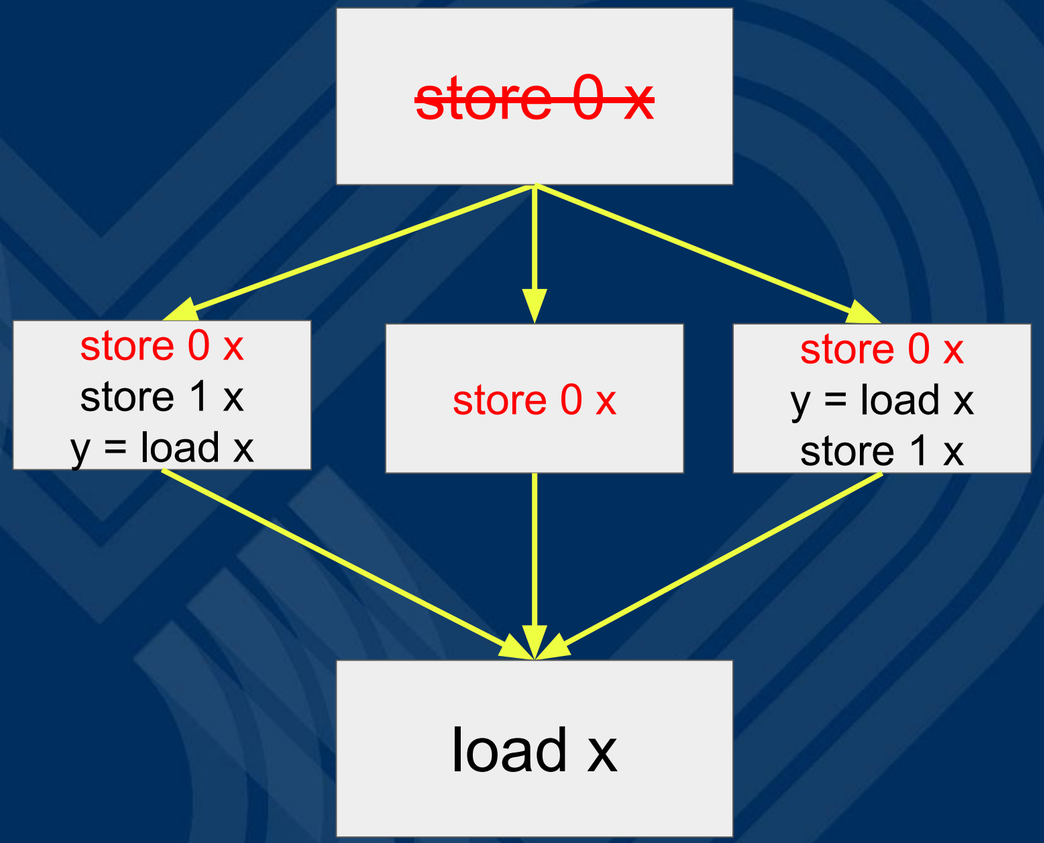
```
partial_dead_stores ← {}

for Basic Block BB in Function F do
    if IN(BB) is not None then
        Clone IN(BB) to the beginning of BB
        Insert IN(BB) into partial_dead_stores
    end if
end for

for Instruction I in partial_dead_stores do
    Remove I from its parent Basic Block
end for
```

← Insert clones of original stores

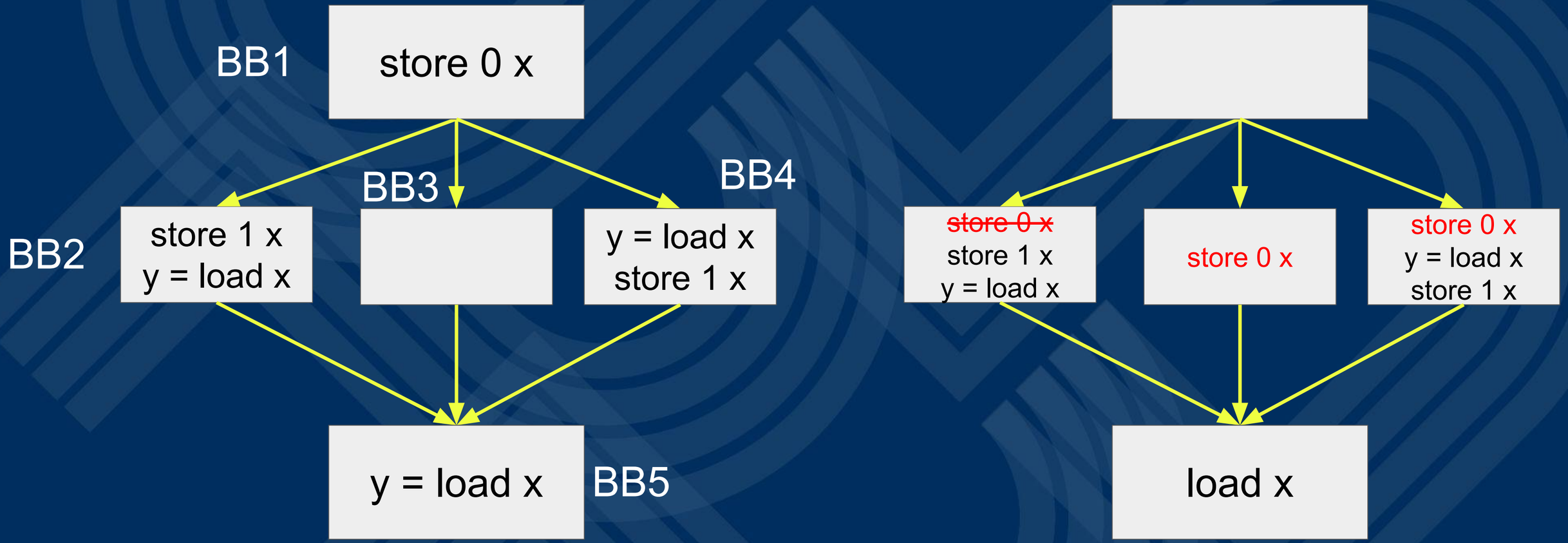
← Remove original stores

Sink Partial Dead Stores

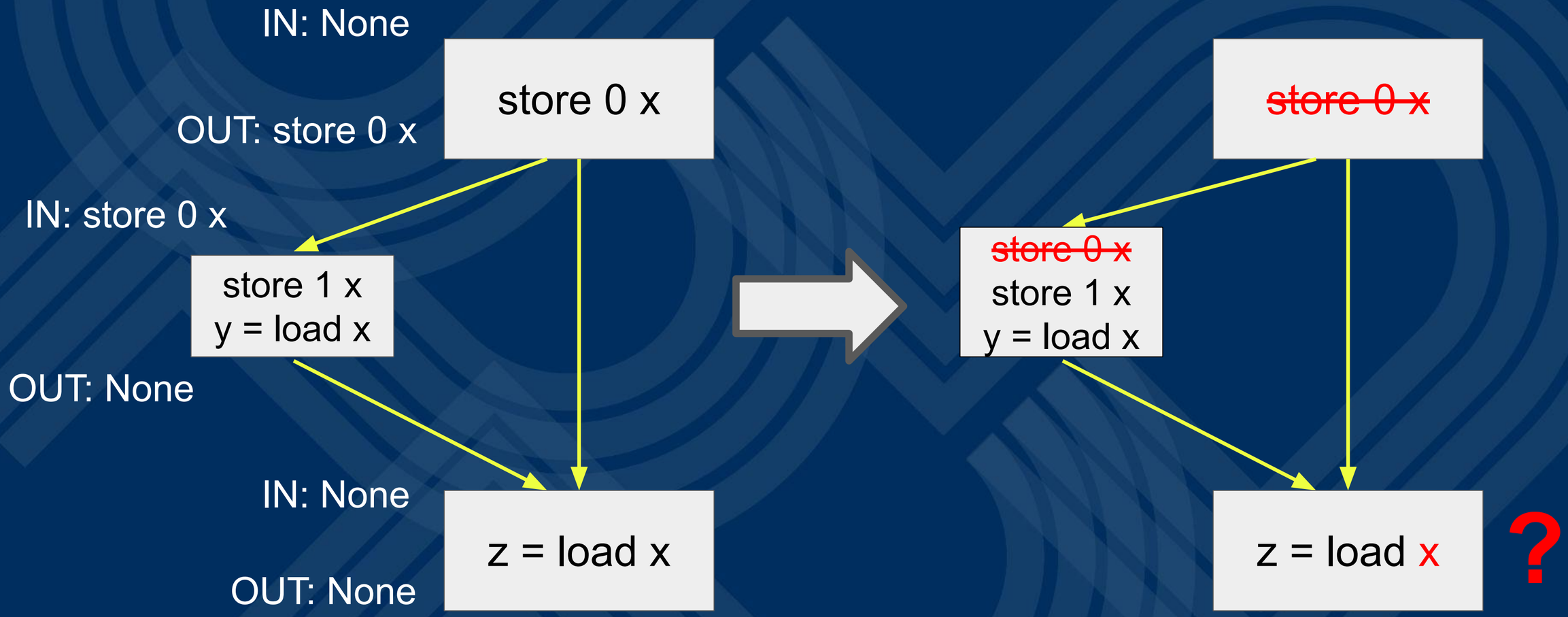




Eliminate Dead Stores in each BB



Handle Fall-through Case*



Benchmarks - for correctness only

Our benchmarks cover different scenarios:

- Basic
- Multiple sinkable stores
- Sinking-Sinking effect
- Sinking-Elimination effect
- Fall-through case

Demo Show

Limitations

- Only considered acyclic control flows
- Efficiency concerns - Conduct sinking analysis for each address, thus having to run the algorithm a lot of times if the program is large
- Fall-through case handling - Too conservative
- Only tested on correctness benchmarks, so how much performance is improved is unknown

Thank you!

Partial Dead Store Elimination

Zhixuan Chen • Fanghao Zhang • Mingye Chen • Yichen Zhong

EECS 583 Fall 2023

COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN