Let's Focus on Copy Instructions

- Optimizations that help optimize away copy instructions:
  - Copy Propagation
  - Dead Code Elimination

- Can all copy instructions be eliminated using this pair of optimizations?

Example Where Copy Propagation Fails

- Use of copy target has multiple (conflicting) reaching definitions

Another Example Where the Copy Instruction Remains

- Copy target (Y) still live even after some successful copy propagations
  - Bottom line:
    - copy instructions may still exist when we perform register allocation
Copy Instructions and Register Allocation

- What clever thing might the register allocator do for copy instructions?

- If we can assign both the source and target of the copy to the same register:
  - then we don’t need to perform the copy instruction at all!
  - the copy instruction can be removed from the code
    - even though the optimizer was unable to do this earlier
- One way to do this:
  - treat the copy source and target as the same node in the interference graph
  - then the coloring algorithm will naturally assign them to the same register
  - this is called “coalescing”

Example Revisited: With Coalescing

- With coalescing, X and Y are now guaranteed to end up in the same register
  - the copy instruction can now be eliminated
- Great! So should we go ahead and do this for every copy instruction?

Should We Coalesce X and Y In This Case?

- It is legal to coalesce X and Y for a “Y = X” copy instruction if:
  - initial definition of Y’s live range is this copy instruction, AND
  - the live ranges of X and Y do not interfere otherwise
- But just because it is legal doesn’t mean that it is a good idea...

Simple Example: Without Coalescing

X = ...;
A = 5;
Y = X;
B = A + 2;
Z = Y + B;
return Z;

Valid coloring with 3 registers

- Without coalescing, X and Y can end up in different registers
  - cannot eliminate the copy instruction

No! That would result in incorrect behavior if this branch is taken.
Why Coalescing May Be Undesirable, Even If Legal

- What is the likely impact of coalescing X and Y on:
  - live range size(s)?
  - recall our discussion of live range splitting
  - colorability of the interference graph?
- Fundamentally, coalescing adds further constraints to the coloring problem
- If we coalesce in this case, we may:
  - save a copy instruction, BUT
  - cause significant spilling overhead if we can no longer color the graph

When to Coalesce

- Goal when coalescing is legal:
  - coalesce unless it would make a colorable graph non-colorable
- The bad news:
  - predicting colorability is tricky!
    - it depends on the shape of the graph
    - graph coloring is NP-hard
- Example: assuming 2 registers, should we coalesce X and Y?

Representing Coalescing Candidates in the Interference Graph

- To decide whether to coalesce, we augment the interference graph
- Coalescing candidates are represented by a new type of interference graph edge:
  - dotted lines: coalescing candidates
    - try to assign vertices the same color
    - (unless that is problematic, in which case they can be given different colors)
  - solid lines: interference
    - vertices must be assigned different colors

How Do We Know When Coalescing Will Not Cause Spilling?

- Key insight:
  - Recall from the coloring algorithm:
    - we can always successfully N-color a node if its degree is < N
- To ensure that coalescing does not cause spilling:
  - check that the degree < N invariant is still locally preserved after coalescing
    - if so, then coalescing won’t cause the graph to become non-colorable
    - no need to inspect the entire interference graph, or do trial-and-error
- Note:
  - We do NOT need to determine whether the full graph is colorable or not
  - Just need to check that coalescing does not cause a colorable graph to become non-colorable
Simple and Safe Coalescing Algorithm

- We can safely coalesce nodes X and Y if \(|X| + |Y| < N\)
  - Note: \(|X|\) = degree of node X counting interference (not coalescing) edges
- Example:
  \[
  \begin{align*}
  X & \quad Y \\
  |X| + |Y| &= (1 + 2) = 3 \\
  \\
  X/Y & \quad \text{Degree of coalesced node can be no larger than 3}
  \end{align*}
  \]
  - if \(N \geq 4\), it would always be safe to coalesce these two nodes
    - this cannot cause new spilling that would not have occurred with the original graph
    - if \(N < 4\), it is unclear
  
  How can we (safely) be more aggressive than this?

What About This Example?

- Assume \(N = 3\)
- Is it safe to coalesce X and Y?

\[
\begin{align*}
|X| + |Y| &= (1 + 2) = 3 \\
\end{align*}
\]

- Notice: X and Y share a common (interference) neighbor: node A
  - hence the degree of the coalesced X/Y node is actually 2 (not 3)
  - therefore coalescing X and Y is guaranteed to be safe when \(N = 3\)
  - How can we adjust the algorithm to capture this?

Building on This Insight

- When would coalescing cause the stack pushing (aka “simplification”) to get stuck?
  1. coalesced node must have a degree \(\geq N\)
     - otherwise, it can be pushed on the stack, and we are not stuck
  2. AND it must have at least \(N\) neighbors that each have a degree \(\geq N\)
     - otherwise, all neighbors with degree \(< N\) can be pushed before this node
       - reducing this node’s degree below \(N\) (and therefore we aren’t stuck)
- To coalesce more aggressively (and safely), let’s exploit this second requirement
  - which involves looking at the degree of a coalescing candidate’s neighbors
    - not just the degree of the coalescing candidates themselves
Briggs’s Algorithm

- Nodes X and Y can be coalesced if:
  - \( \text{number of neighbors of } X/Y \text{ with degree } \geq N < N \)
- Works because:
  - all other neighbors can be pushed on the stack before this node, and then its degree is < N, so then it can be pushed
- Example: \( N = 2 \)

George’s Algorithm

Motivation:
- imagine that X has a very high degree, but Y has a much smaller degree
  - (perhaps because X has a large live range)
- With Briggs’s algorithm, we would inspect all neighbors both X and Y
  - but X has a lot of neighbors!
- Can we get away with just inspecting the neighbors of Y?
  - showing that coalescing makes coloring no worse than it was given X?

Briggs’s Algorithm

- Nodes X and Y can be coalesced if:
  - \( \text{number of neighbors of } X/Y \text{ with degree } \geq N < N \)
- More extreme example: \( N = 2 \)
Summary

• **Coalescing** can enable register allocation to **eliminate copy instructions**
  - if both source and target of copy can be allocated to the same register
• However, coalescing must be applied with care to **avoid causing register spilling**
• Augment the interference graph:
  - **dotted lines** for coalescing candidate edges
  - try to allocate to same register, unless this may cause spilling
• **Coalescing Algorithms**:
  - simply based upon **degree of coalescing candidate nodes (X and Y)**
  - Briggs’s algorithm
    • look at degree of neighboring nodes of X and Y
  - George’s algorithm
    • asymmetrical: look at neighbors of Y (degree and interference with X)