Karlsruher Institut für Technologie

# Praktikum Compilerbau
# Sitzung 8 – Optimierung: CSE

**Prof. Dr.-Ing. Gregor Snelting**
**Matthias Braun und Sebastian Buchwald**

IPD Snelting, Lehrstuhl für Programmierparadigmen

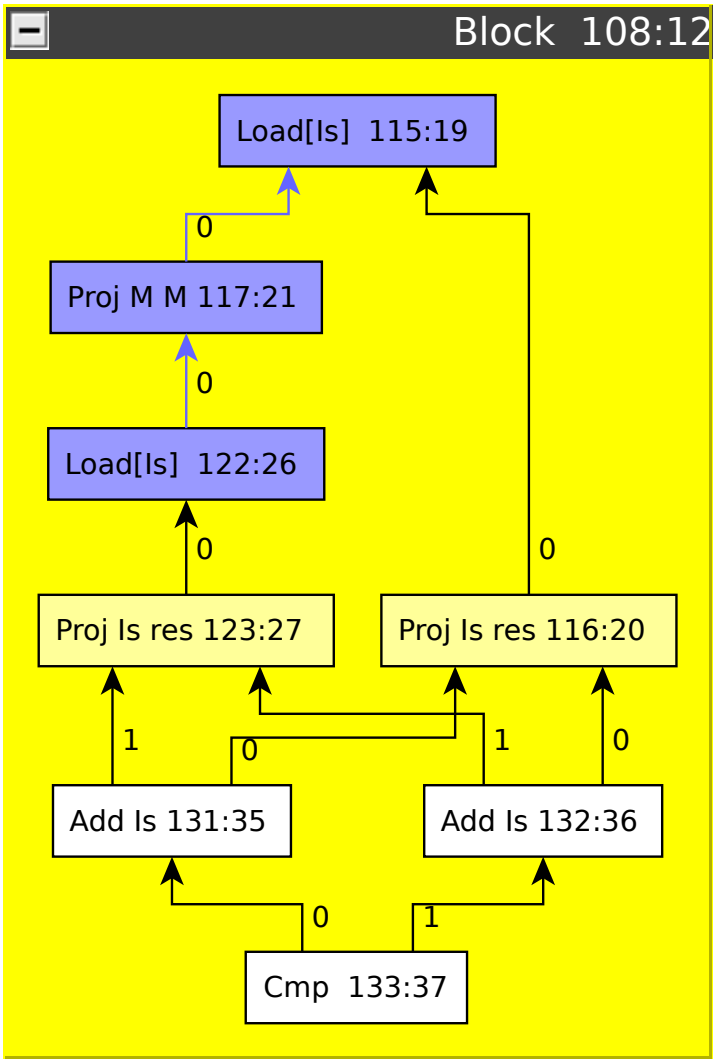
1. Letzte Woche
2. Einführung
3. GVN als Datenflussanalyse
4. CSE mit libFirm
5. Sonstiges

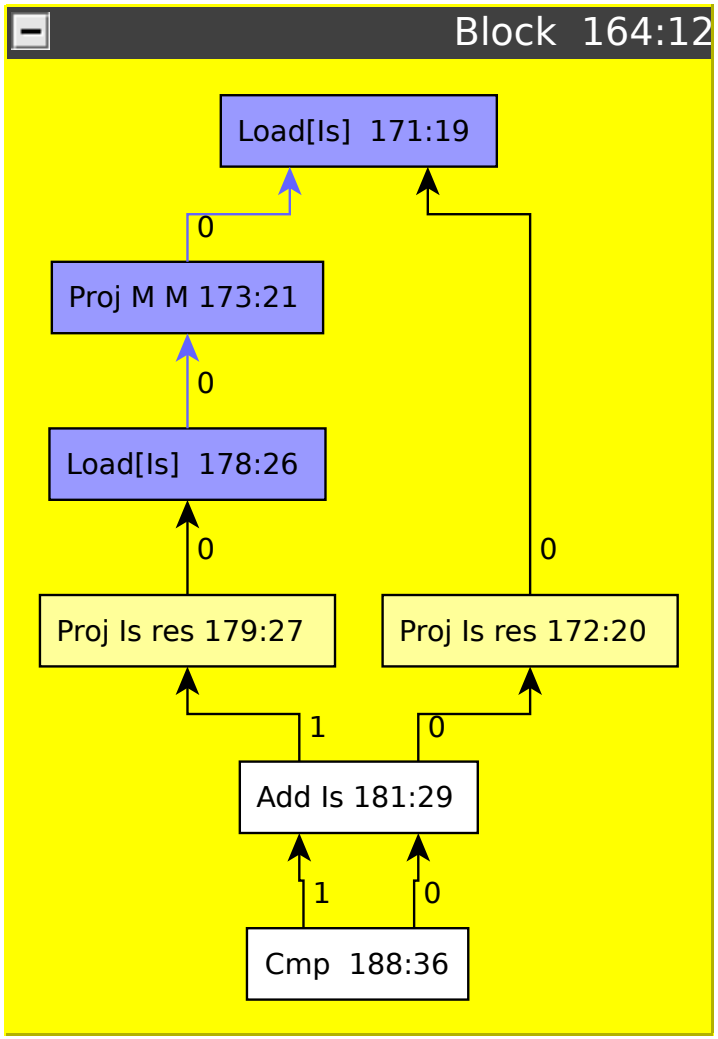
# Letzte Woche

- Was waren die Probleme?
- Hat soweit alles geklappt?

1. Letzte Woche
2. Einführung
3. GVN als Datenflussanalyse
4. CSE mit libFirm
5. Sonstiges

# Elimination gemeinsamer Teilausdrücke



# Globale Wertnummerierung

Globale Wertnummerierung (engl. global value numbering, GVN) ist eine Analyse zur Identifikation gleicher Werte.

Was soll GVN mindestens erkennen?

- Gleiche Operationen auf gleichen Werten ergeben den gleichen Wert

Mögliche Erweiterungen

- Algebraische Identitäten
  - $x + x = 2 \cdot x = x << 1$
  - $x + 0 = 1 \cdot x = x$
  - ...

1. Letzte Woche
2. Einführung
3. GVN als Datenflussanalyse
4. CSE mit libFirm
5. Sonstiges

# GVN als Datenflussanalyse

- $C_{x,y} \in \{\cong, \ncong\}$ gibt an ob Werte $x$ und $y$ kongruent sind
  - $\bot$ bedeutet Werte sind kongruent $\cong$
  - $\top$ bedeutet Werte sind nicht kongruent $\ncong$
- Verband ist das kartesische Produkt aller $C_{x,y}$
- Punktweise Anwendung von $\sqcup$ bei mehreren Vorgängern
- Transferfunktionen
  - $f_{x_2=\phi(x_0,x_1)}(I) = I\left[C_{x_2,y} \leftarrow \begin{cases} \cong & y_{op} = \phi \wedge \forall i : op_i(x_2) \cong op_i(y) \\ \cong & C_{x_0,x_1} \wedge C_{x_0,y} \\ \ncong & \text{sonst} \end{cases}\right]$
  - $f_{z=x+y}(I) = I\left[C_{z,u} \leftarrow \begin{cases} \cong & u_{op} = + \wedge \exists\sigma\forall i : op_i(z) \cong op_{\sigma(i)}(u) \\ \ncong & \text{sonst} \end{cases}\right]$
  - ...

1. Letzte Woche
2. Einführung
3. GVN als Datenflussanalyse
4. CSE mit libFirm
5. Sonstiges

# GVN mit libFirm

- Partitionen als grundlegende Datenstruktur
  - alle kongruenten Werte liegen in einer Partition
- Initialisierung
  - alle Knoten in eine Partition oder
  - unterschiedliche Partitionen für verschiedene Blöcke und Opcodes
    - reduziert Partitionsgröße erheblich
    - ermöglicht Opcode-spezifische Hashfunktionen
- inkrementelles Verfeinern der Partitionen bis alle Werte innerhalb der Partitionen kongruent sind

# Hashfunktionen

- Wir benutzen Hashfunktionen um verschiedene Werte zu identifizieren
- Hashfunktionen arbeiten auf Partitionen
  - Effiziente Zuordnung Knoten $\mapsto$ Partition nötig
- Für kommutative Operationen sollte Hashfunktion unabhängig von der Reihenfolge der Operanden sein
- Wrapper-Klasse(n) mit überschriebener Hashfunktion verwenden

# Der Partitionierungs-Algorithmus

1. Berechne initiale Partitionierung
2. Füge alle Partitionen zur Arbeitsliste hinzu
3. Entferne ein Element x aus der Arbeitsliste
   - 3.1 Berechne für jedes Element von x den Hashwert
   - 3.2 Partitioniere *x* anhand der verschiedenen Hashwerte
   - 3.3 Füge alle neuen Partitionen zur Arbeitsliste hinzu
   - 3.4 Füge alle Partitionen mit Verwendungen von Elementen der neuen Partitionen zur Arbeitsliste hinzu
4. Falls Arbeitsliste nicht leer: gehe zu 3

# Transformation: CSE

- Wähle aus jeder Partition einen Repräsentanten
  - Was ist ein guter Repräsentant wenn man algebraische Identitäten berücksichtigt?
- Ersetze alle Knoten der Partition durch den Repräsentanten
  - Möglich weil kongruente Werte zum selben Grundblock gehören

1. Letzte Woche
2. Einführung
3. GVN als Datenflussanalyse
4. CSE mit libFirm
5. Sonstiges

# Feedback! Fragen? Probleme?

- Anmerkungen?
- Probleme?
- Fragen?