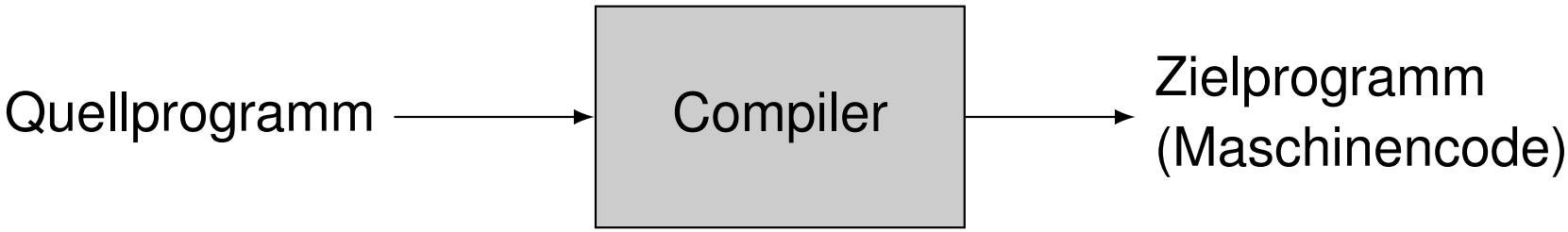
Karlsruher Institut für Technologie

Praxis der Softwareentwicklung
libFirm-Graphen

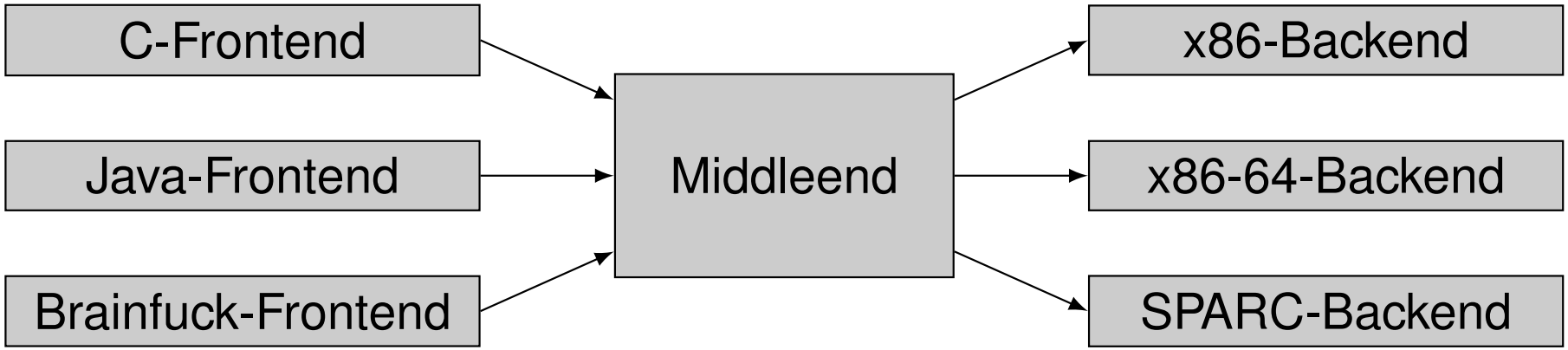
Marcel Radermacher, Simon Bischof
Sebastian Buchwald, Martin Hecker

IPD Snelting, Lehrstuhl für Programmierparadigmen

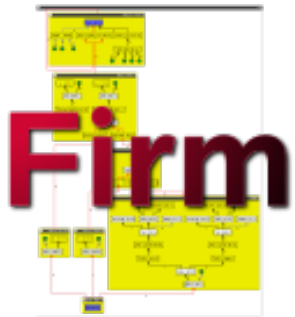
Compiler – Funktionalität



Compiler – Aufbau

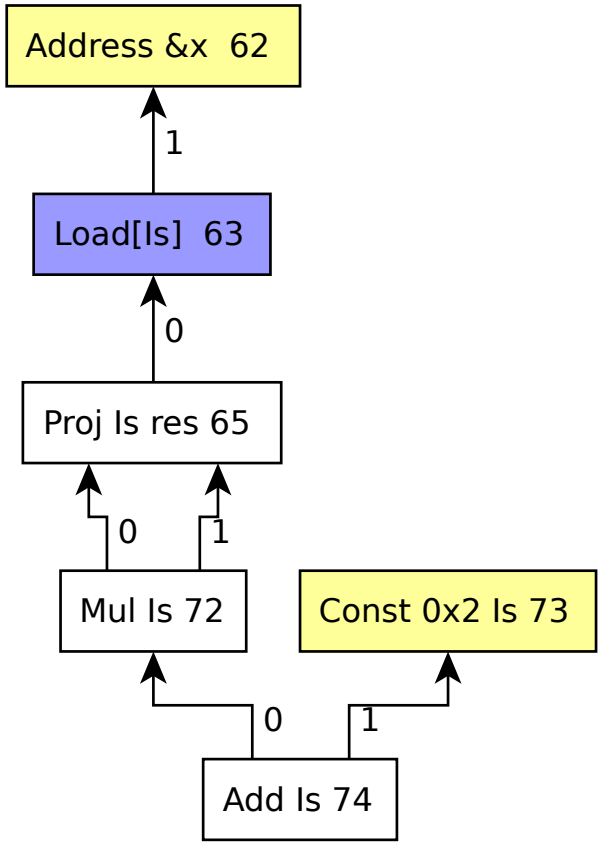


Firm und libFirm



- libFirm ist die Implementierung der low-level-Programmrepräsentation Firm.
- Low-level: Näher an der Maschine als an der Quellsprache.
- Komplett Graph-basiert
- Enthält zahlreiche Optimierungen.
- Sehr ausgereift (für ein Forschungsprojekt).
- Open Source: `http://pp.ipd.kit.edu/firm/`

Datenabhängigkeiten



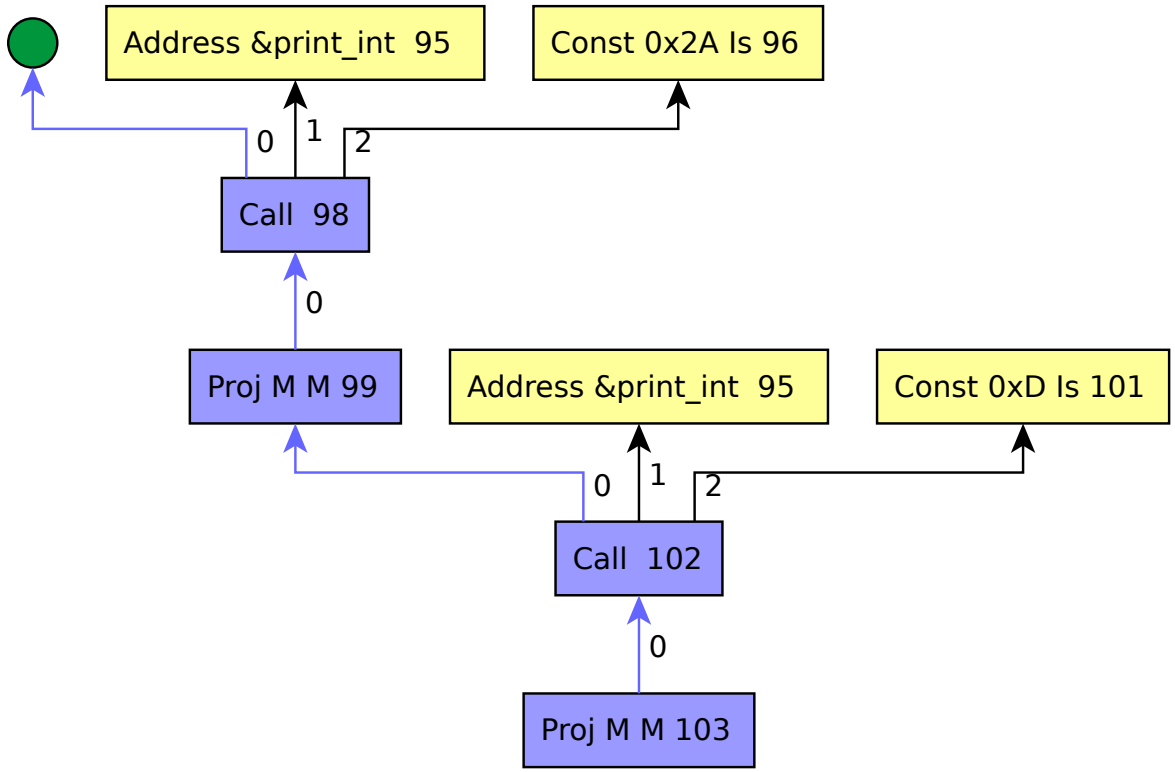
Beispiel: `x * x + 2`

- Operationen sind Knoten in einem Graph.
- Kanten geben Datenabhängigkeiten an.

„Speicher“-abhängigkeiten

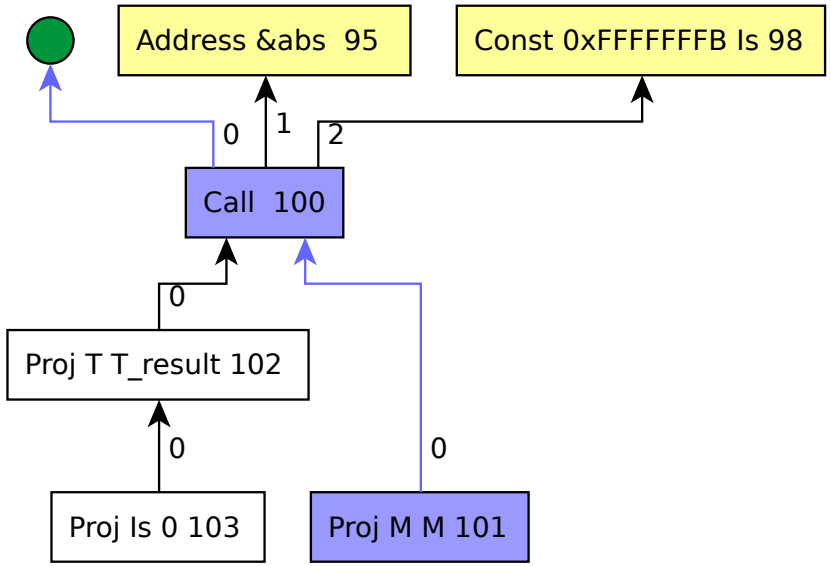
Beispiel: print_int(42); print_int(13);

- Operationen können Nebeneffekte haben (Speicher verändern, Bildschirmausgaben).
- Ordnung muss durch weitere Abhängigkeiten erzwungen werden.



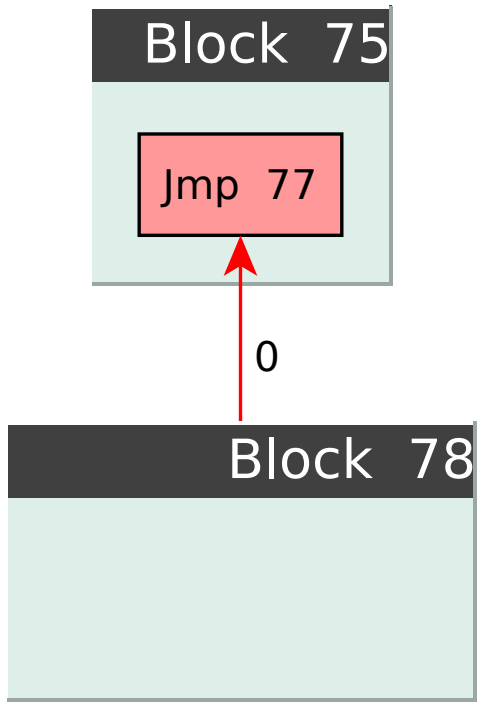
Tupelwerte und Projektionsknoten

Beispiel: `abs(-5)`



- Manche Operationen liefern mehrere Werte zurück. Diese werden in einem Tupelwert zusammengefasst.
- Mit Hilfe der `Proj`-Operation kann man einzelne Werte aus einem Tupel extrahieren.

Grundblöcke und Steuerfluss



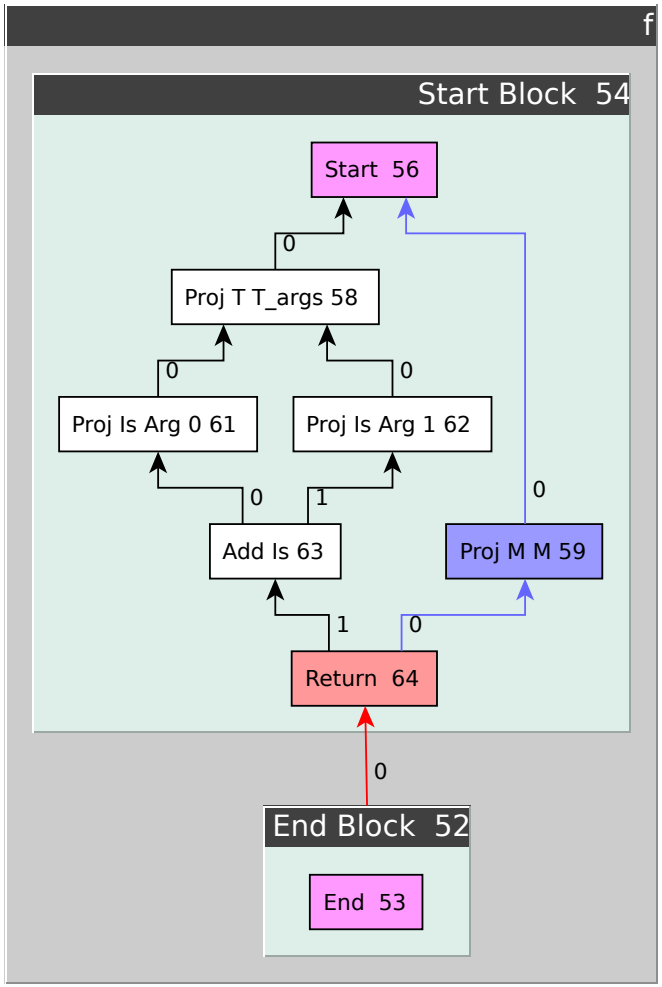
- Grundblöcke sind normale Knoten, die Sprungbefehle als Vorgänger besitzen.
- Jeder Knoten ist einem Grundblock zugeordnet (Vorgänger Nummer -1).

Methoden

- Eine Funktion beginnt am `Start`-Knoten im Startblock.
- Der `Start`-Knoten erzeugt einen initialen Speicherwert und die Funktionsargumente.
- Sie endet am `End`-Knoten im Endblock.
- Der Endblock hat `Return`-Operationen als Vorgänger.

Beispiel: Komplette Methode

Beispiel: **int** f(**int** a, **int** b) { **return** a + b; }



Demo

Karlsruher Institut für Technologie

Live-Demo