Dokumentation: Optimierung

Daniel Marzin David Knötel Björn Karger Martin Görick

1 Einleitung

2 Ablauf

Für die Optimierung steht der Code in einem String, wobei die Befehle je eine Zeile einnimmt. Dieser Code wird zuerst in die einzelnen Befehle geparst, jedoch vorher in die einzelnen Funktionen unterteilt. Nach dem parsen wird mit der Optimierung begonnen.
Die Optimierung wird für jede Funktion durchgeführt. Wenn die Funktion gewählt ist, wird der Flussgraph für diese erstellt. Dafür werden die *next* und *previous* Blöcke der enthaltenen Blöcke gesetzt und die Label angepasst. Nach dem erstellen des Flussgraphen wird die Registermap erstellt. In dieser stehen die Definitionen und Verwendungen der Register. Nachdem der Flussgraph und die Registermap erstellt wurde, wird mit der Optimierung begonnen. Die jeweiligen Funktionen werden im Abschnitt 4 beschrieben. Als erstes wird mit der Dead Register Elimination angefangen. Danach findet das Constant Folding und Propagation statt. Im Anschluss wird die Lebendigkeitsanalyse über die Befehle *load* und *store* durchgeführt. Dann werden doppelte Befehle mittels Common Expression entfernt. Am Schluss werden die Blöcke, welche nur unbedingte Sprünge enthalten entfernt. Nun wird vor der Ausgabe des optimierten Codes die Labels angepasst.

3 Programm Aufbau

4 Funktionen

4.1 Dead Register Elimination

Die Tote Register Elimination wird in der Funktion 1 durchgeführt.

Listing 1: Dead Register Elimination Funktion

```
public void eliminateDeadRegisters()
```

Dafür wird am Anfang die Definition von toten Register aus dem Register entfernt. Da es Operanden in den gelöschten Befehlen geben kann, welche nun keine Verwendung mehr haben, können diese ebenfalls gelöscht werden. Deswegen werden alle gelöschten Befehle im Register durchgegangen und die Operanden untersucht ob diese weiterhin verwendet werden. Werden diese nicht weiter benutzt, dann werden diese auch entfernt.

4.2 Constant Folding

Constant Folding wird in der Funktion 2 durchgeführt.

Listing 2: Constant Folding Funktion

```
public void constantFolding()
```

Es werden zuerst alle Befehle des Blockes durchlaufen und Constant Folding durchgeführt, wenn dies möglich ist. Danach wird über den veränderten Befehlen eine Constant Propagation abgearbeitet. Sollten dabei weitere Änderungen entstehen wird erneut Constant Folding durchgeführt.

4.3 Globale Lebendigkeitsanalyse für load und store

Die globale Lebendigkeitsanalyse wird in der Funktion 3 abgehandelt.

Listing 3: Funktion zur Globale Lebendigkeitsanalyse

```
public void globalLiveVariableAnalysis()
```

Es wird zuerst die def- und use-Mengen der Blöcke erstellt. Dies entsteht anhand der *store* und *load* Befehle im jeweiligen Block. Sind die Mengen in den Blöcken ermittelt, können die überflüssigen *store*-Befehle entfernt werden. Dafür werden die Befehle eines Blockes von hinten durchgegangen und es wird überprüft ob die Operanden lebendig sind. Wenn ein Befehl Operanden hatte, welche nun keine Verwendung mehr haben, werden diese aus dem Register gelöscht.

4.4 Common Expressions

In der Funktion 4 erfolgt das abarbeiten der Common Expression.

Listing 4: Funktion zur Entfernung doppelter Befehle

```
public void removeCommonExpressions()
```

Die Common Expression wird für jeden Block durchgeführt, dabei wird überprüft ob diese doppelte Befehle enthalten. Dabei werden nur die Befehle einer vorher festlegten Liste beachtet (Derzeit: ADD, SUB, MUL, DIV und LOAD). Sollten doppelte Befehle gefunden werden, wird das aktuelle mit dem bestehenden Kommando ersetzt. Am Ende wird für den Block Constant Propagation durchgeführt.

4.5 Entfernung leerer Blöcke

Das entfernen von Blöcken die nur unbedingte Sprünge enthalten findet in der Funktion 5 statt.

Listing 5: Funktion zur Entfernung leerer Blöcke

```
public void deleteEmptyBlocks()
```

Zuerst wird geprüft, ob ein Block nur unbedingte Sprünge aufweist. Wenn dies ermittelt ist, wird der Block entfernt und alle Vorgängerblöcke untersucht ob diese auf den gelöschtes Block verweisen. Dafür wird der Verweis auf den aktuellen Block mit dem des Zielblocks getauscht und das Register angepasst. Danach wird noch der Flussgraph angepasst, damit dieser weiterhin verwendet werden kann. Am Ende werden die gelöschten Blöcke entfernt.