Übung 9: Transformation von Ausdrücken und Befehlen

(Besprechung am Montag, den 7. Juli 2003)

1. Transformation von Standardfunktionen in Ausdrücken

In der Vorlesung wurde die Transformation von Ausdrücken behandelt, die Literale, Konstanten oder Variablen einfachen Typs sowie vordefinierte unäre und binäre Operationen enthalten können.

Erweitert die Transformation um die Behandlung der Aufrufe von Standardfunktionen wie `sin`, `cos` und dgl. (Die Funktionen können 1, 2 und mehr einfache Parameter haben, die jeweils in eine Zelle passen.)

Nimm an, der Code für eine Standardfunktion *f* stünde an einer Speicheradresse α_f und ende mit einem (noch zu definierenden) Befehl **`retsf`** mit maximal einem Modifikator. Die Instruktion zum Aufruf soll **`jpsf`** heißen und praktischerweise α_f als Modifikator haben. Zum Retten des Programmzählers soll ein Register `OPC` zur Verfügung stehen.

Wie könnten die neuen Instruktionen **`retsf`** und **`jpsf`** definiert werden?
Wie kann die Transformation eines Aufrufs $f(e_1, ...e_k)$ einer Standardfunktion *f* definiert werden? (Spielt Transformation und Ausführung an einem geeigneten Beispiel durch.)
Unter welcher Bedingung funktioniert diese Transformation?

2. Transformation von Ausdrücken mit zusammengesetzten Operanden

Wie müsstet Ihr die Transformation von Ausdrücken ändern, wenn auch zusammengesetzte Werte (für deren Abspeicherung mehr alseine Zelle benötigt wird) als Werte von Aggregaten, Konstanten, Variablen und Funktionsergebnissen auftreten können?

3. Transformation von realistischen numerischen Fallunterscheidungen

Diskutiert, wann sich bei der Transformation von numerischen Fallunterscheidungen die Standardübersetzung mit indexierten Sprüngen lohnt, und , und wann man diese Anweisung eher als eine Kette von Booleschen Fallunterscheidungen übersetzen sollte.

Sollte die Syntax der numerischen Fallunterscheidung dazu geändert werden?