Protokoll
Grundlagen des Übersetzerbau (1)

13. März 2013

Prüfer: Prof. Dr. Michael Philippsen
Beisitz: Dipl.-Inf. Stefan Kempf
Papier + Stift bereitgestellt

1. Was passiert wenn der Compiler Code übersetzt (sollte anhand eines kleinen Stücken Codes erklärt werden). Lexer baut aus Zeichenketten Token (bei Identifiern: Eintrag in Namenstabelle + Verweis auf diese im Token !)
   Parser baut daraus den AST.
   Typprüfung prüft Funktionsaufrufe dann anhand des Prototypen (beachte Sachen wie Typweitung)
2. Verändern des Übungscompilers: Unterstützung für eine beliebige Anzahl an Argumenten (vgl. `va_args`).
   Hierzu wurde vorneweg der Stackaufbau bei Funktionsaufrufen gefragt (auch wie das bei beliebiger Anzahl an Argumenten klappt.)
   Lexer erweitern (hinzufügen eines Tokens für ...), Grammatik des Parsers anpassen, bei der Typprüfung dann aufpassen.
3. *Zusatzfrage:* Bei ARM werden Parameter per Register übergeben, wie klappt das da dann ? (Antwort war: Register werden auf Stack gespeichert, da dann wieder wie oben)
4. Registervergabe bei Ausdrucksbaum, musste zuvor aus Formel erstellt werden `x+y + 2*x + (x-y) oder ähnlich`
   Erstellen von Zwischencode (beliebige Anzahl an Registern) und anschliessende Registervergabe per Graphfärben.
   Dazu: Ermitteln der Lebensspannen, Aufbau des Interferenzgraphen (nur teilweise durchzuführen).
   Im Anschluss erklären was mit dem Interferenzgraph passiert.
   Hier wurde noch gefragt wie man dann noch die Befehlsreihenfolge optimieren kann (Stichwort: data hazzards) siehe Foliensatz 11 ab Seite 20

5. `while`-Schleife durch Sprünge darstellen.