11. Übung - Programmierung

H_0, AM_0

SS 18

H_0

Einfacher Kern von haskell

- nur tail-rekursive Funktionen über Int
- formale Parameter jeder Funktion der Reihe nach von x1 bis xk benennen (entspr. Position im HS)
- *tail-rekursive* Funktionen $\Rightarrow$ enthalten auf der rechten Seite entweder
  1. keinen Funktionsaufruf
  2. genau einen Funktionsaufruf an der "äußersten Position" (d.h. nicht eingeschachtelt) oder
  3. eine Fallunterscheidung, deren Zweige wiederum nach (1), (2) oder (3) aufgebaut sind

  $\Rightarrow$ kein Laufzeitkeller zur Verwaltung rekursiver Funktionsaufrufe nötig

$H_0 - AM_0$ Übersetzung

- Parameter x1 bis xk werden im HS auf den Adressen 1, ..., k abgespeichert
- vor einem Funktionsaufruf $\rightarrow$ Parameter 1,...,k der Funktion erst auf dem DK ausrechnen
- anschließend Parameter mit STORE k; ... STORE 1; im HS ablegen und zur Funktion springen
- Ausgabe immer nur von Speicheradresse 1 $\rightarrow$ WRITE 1;

Übersetzung mit baumstrukturierten Adressen: $H_0 \leftrightarrow AM_0$

Übersetzung der rhs $\rightarrow$ "Right hand side"

rhstrans(e, a) :=	*exptrans*(e) STORE 1; WRITE 1; JMP 0;
rhstrans($f\ e_1 ... e_n, a$) :=	*exptrans*(e_1)... exptrans(e_n) STORE n; . . . STORE 1; JMP f ;
rhstrans(if be then r_1 else r_2 , a) :=	*bexptrans*(b_e) JMC a.3; *rhstrans*(r_1, a.1) a.3 : *rhstrans*(r_2, a.2)

Dabei sind e_i arithmetische Ausdrücke, r_i Right hand sides und a, f Funktionsnamen

Aufgabe 2 a)

```
        READ 1; READ 2;
        LOAD 2; LIT 3; LOAD 1; STORE 3, STORE 2; STORE 1;
        JMP test;
test:   LOAD 1; LIT 0; 9: EQ;
        JMC test.3;
        LOAD 3; STORE 1; WRITE 1; JMP 0;
test.3: LOAD 1; LIT 1; SUB;
        LOAD 3;
        LOAD 2; LOAD 3; MUL; LOAD 3; ADD;
        STORE 3; STORE 2; STORE 1;
        JMP test;
```