PROGRAMMIERUNG

ÜBUNG 11: C_1 UND ABSTRAKTE MASCHINE AM_1

Eric Kunze

eric.kunze@mailbox.tu-dresden.de

INHALT

Implementierung von C_1 und abstrakte Maschine AM_1

C_1 **UND** AM_1

- **bisher:** Implementierung von C_0 mit AM_0
- **jetzt:** Erweiterung auf C_1 mit AM_1

C_1 **UND** AM_1

- **bisher:** Implementierung von C_0 mit AM_0
- **jetzt:** Erweiterung auf C_1 mit AM_1
  - Erweiterung um Funktionen *ohne* Rückgabewert
  - Einschränkungen von C_0 bleiben erhalten

C_1 **UND** AM_1

- **bisher:** Implementierung von C_0 mit AM_0
- **jetzt:** Erweiterung auf C_1 mit AM_1
  - Erweiterung um Funktionen *ohne* Rückgabewert
  - Einschränkungen von C_0 bleiben erhalten
- **Implementierung** durch
  - Syntax von C_1
  - Befehle und Semantik einer abstrakten Maschine AM_1
  - Übersetzer $C_1 \leftrightarrow AM_1$

ABSTRAKTE MASCHINE AM_1

Die AM_1 besteht aus

- einem Ein- und Ausgabeband,
- einem Datenkeller,
- einem Laufzeitkeller,
- einem Befehlszähler und
- einem Referenzzeiger (REF).

Im Vergleich zur AM_0 ist also aus dem Hauptspeicher ein *Laufzeitkeller* geworden und der *Referenzzeiger* ist hinzugekommen.

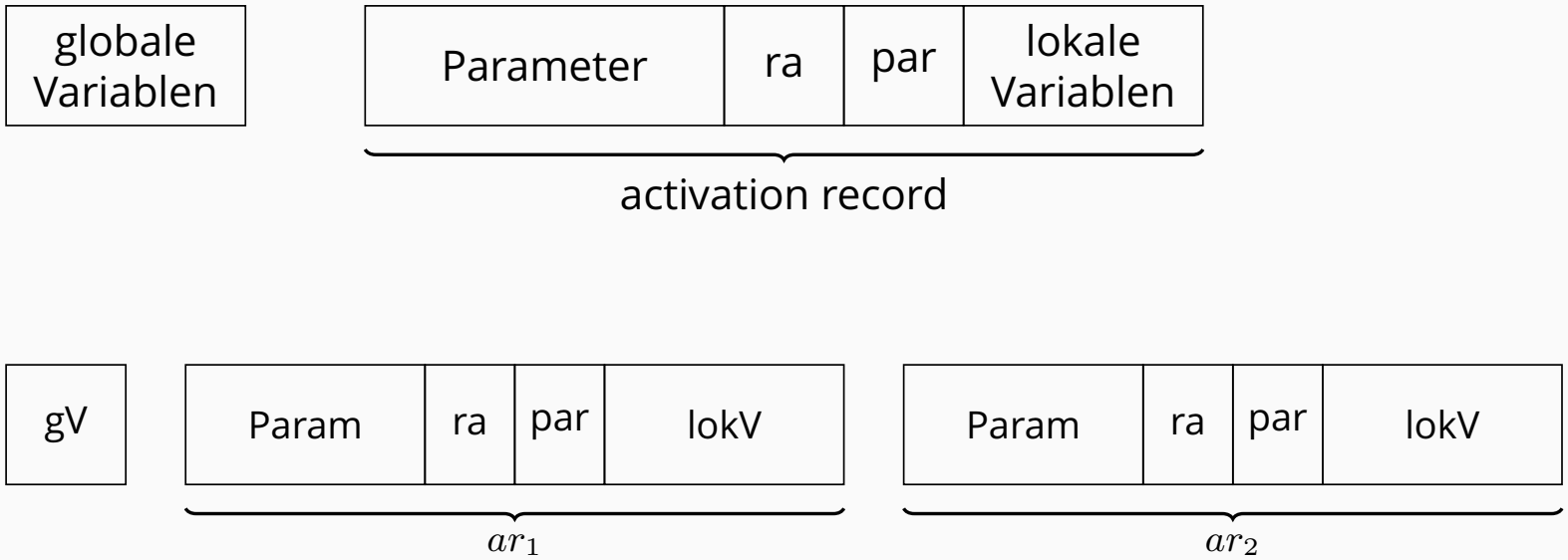
Den Zustand der AM_1 beschreiben wir daher nun mit einem 6-Tupel

$$(m, d, h, r, inp, out) = (\text{BZ}, \text{DK}, \text{LZK}, \text{REF}, \text{Input}, \text{Output})$$

FUNKTIONSAUFRUFE & DER LAUFZEITKELLER

Wofür brauchen wir den REF? $\rightarrow$ Funktionsaufrufe & Rücksprünge

Struktur des Laufzeitkellers:



Funktionsaufrufe übersetzen:

- Parameter `LOAD` & `PUSH`
- Funktion `CALL`

BEFEHLSSEMANTIK DER AM_1

$b \in \{\text{global}, \text{lokal}\}$
r ... aktueller REF

$$adr(r, b, o) = \begin{cases} r + o & \text{wenn } b = \text{lokal} \\ o & \text{wenn } b = \text{global} \end{cases}$$

Befehl	**Auswirkungen**
LOAD(b,o)	Lädt den Inhalt von Adresse $adr(r, b, o)$ auf den Datenkeller, inkrementiere Befehlszähler
STORE(b,o)	Speichere oberstes Datenkellerelement an $adr(r, b, o)$, inkrementiere Befehlszähler
WRITE(b,o)	Schreibe Inhalt an Adresse $adr(r, b, o)$ auf das Ausgabeband, inkrementiere Befehlszähler
READ(b,o)	Lies oberstes Element vom Eingabeband, speichere an Adresse $adr(r, b, o)$, inkrementiere Befehlszähler

BEFEHLSSEMANTIK DER AM_1

Befehl	Auswirkungen
LOADI(o)	Ermittle Wert ($= b$) an Adresse $r + o$, Lade Inhalt von Adresse b auf Datenkeller, inkrementiere Befehlszähler
STOREI(o)	Ermittle Wert ($= b$) an Adresse $r + o$, nimm oberstes Datenkellerelement, speichere dieses an Adresse b, inkrementiere Befehlszähler
WRITEI(o)	Ermittle Wert ($= b$) an Adresse $r + o$, schreibe den Inhalt an Adresse b auf Ausgabeband, inkrementiere Befehlszähler
READI(o)	Ermittle Wert ($= b$) an Adresse $r + o$, lies das oberste Element vom Eingabeband, speichere es an Adresse b , inkrementiere Befehlszähler
LOADA(b,o)	Lege $adr(r, b, o)$ auf Datenkeller, inkrementiere Befehlszähler
PUSH	oberstes Element vom Datenkeller auf Laufzeitkeller, Befehlszähler inkrementieren
CALL *adr*	Befehlszählerwert inkrementieren und auf LZK legen, Befehlszähler auf *adr* setzen, REF auf LZK legen, REF auf Länge des LZK ändern
INIT n	n-mal 0 auf den Laufzeitkeller legen
RET n	im LZK alles nach REF-Zeiger löschen, oberstes Element des LZK als REF setzen, oberstes Element des LZK als Befehlszähler setzen, n Elemente von LZK löschen

Übungsblatt 11

AUFGABE 1 – TEIL (A)

```
1 while (*p > i) { f(p); i = i + 1; }
2 p = \&i;
```

$$tab_{g+\text{lDecl}} = \{\texttt{f}/(\text{proc}, 1), \texttt{g}/(\text{proc}, 2), \texttt{i}/(\text{var}, \text{lokal}, 1), \texttt{p}/(\text{var-ref}, -2)\}$$

AUFGABE 1 – TEIL (A)

```
1 while (*p > i) { f(p); i = i + 1; }
2 p = \&i;
```

$tab_{g+\text{lDecl}} = \{\texttt{f}/(\text{proc}, 1), \texttt{g}/(\text{proc}, 2), \texttt{i}/(\text{var}, \text{lokal}, 1), \texttt{p}/(\text{var-ref}, -2)\}$

Lösung.

```
2.2.1  LOADI(-2) ; LOAD(lokal,1) ; GT ; JMC 2.2.2 ;
       LOAD(lokal,-2) ; PUSH ; CALL 1 ;
       LOAD(lokal,1) ; LIT 1 ; ADD ; STORE(lokal,1) ;
       JMP 2.2.1 ;

2.2.2  LOADA(lokal,1) ; STORE(lokal,-2) ;
```

AUFGABE 1 – TEIL (B)

Gegeben ist folgender AM_1-Code:

```
1  INIT 1;
2  CALL 18;
3  INIT 0;
4  LOAD(lokal,-2);
5  LIT 0;
6  GT;
7  JMC 17;
8  LIT 2;
9  LOADI(-3);
10 MUL;
11 STOREI(-3);
12 LOAD(lokal,-2);
13 LIT 1;
14 SUB;
15 STORE(lokal,-2);
16 JMP 4;
17 RET 2;
18 INIT 0;
19 READ(global,1);
20 LOADA(global,1);
21 PUSH;
22 LOAD(global,1);
23 PUSH;
24 CALL 3;
25 WRITE(global,1);
26 JMP 0;
```

Führen Sie 11 Schritte der AM_1 auf der Konfiguration $\sigma = (22, \varepsilon, 1:3:0:1, 3, \varepsilon, \varepsilon)$ aus.

AUFGABE 2 – TEIL (A)

```
1  READ 1;
2  READ 2;
3  LOAD 1;
4  LIT 0;
5  GT;
6  JMC 20;
7  LOAD 2;
8  LOAD 1;
9  SUB;
10 STORE 1;
11 LOAD 2;
12 LOAD 1;
13 GT;
14 JMC 19;
15 LOAD 2;
16 LIT 2;
17 DIV;
18 STORE 2;
19 JMP 3;
20 WRITE 1;
```

AUFGABE 2 – TEIL (B)

Ablauf der abstrakten Maschine:

BZ	DK	HS	Inp	Out
7	ε	[1/3, 2/1]	ε	ε
8	3	[1/3, 2/1]	ε	ε
9	1:3	[1/3, 2/1]	ε	ε
10	2:1:3	[1/3, 2/1]	ε	ε
11	2:3	[1/3, 2/1]	ε	ε
12	5	[1/3, 2/1]	ε	ε
13	ε	[1/3, 2/5]	ε	ε
3	ε	[1/3, 2/5]	ε	ε
4	5	[1/3, 2/5]	ε	ε
5	5:5	[1/3, 2/5]	ε	ε
6	0	[1/3, 2/5]	ε	ε
14	ε	[1/3, 2/5]	ε	ε
15	ε	[1/3, 2/5]	ε	3