Zur Erinnerung: Übersetzung von C_0-Programmen

C_0	AM_0
`scanf("%i", &x);`	`READ` HS-Adresse von `x;`
`printf("%d", x);`	`WRITE` HS-Adresse von `x;`
`z = 5;`	`LIT 5;` `STORE` HS-Adresse von `z;`

`z = x / y;`	`LOAD` HS-Adresse von `x;`
	`LOAD` HS-Adresse von `y;`
	`DIV;`
	`STORE` HS-Adresse von `z;`

`if (x > y)`
`then { ... }`
	`LOAD` HS-Adresse von `x;`
	`LOAD` HS-Adresse von `y;`
	`GT;`
	`JMC a.1;`
	AM_0-Code des then-Blocks
	`a.1:` weiterer Code

Zur Erinnerung: Übersetzung von C_0-Programmen

C_0	**AM_0**

`if (x > y)`
`then { ... }`
`else { ... }`
	`LOAD` HS-Adresse von `x`;
	`LOAD` HS-Adresse von `y`;
	`GT`;
	`JMC a.1`;
	AM_0-Code des then-Blocks
	`JMP a.3`;
	`a.1:` AM_0-Code des else-Blocks
	`a.3:` weiterer Code

`while (x < y) { ... }`	`a.1:` `LOAD` HS-Adresse von `x`; `LOAD` HS-Adresse von `y`; `LT`; `JMC a.2`; AM_0-Code des while-Blocks `JMP a.1`; `a.2:` weiterer Code

Übung 1 (b)

BZ	DK	HS	Inp	Out
(1,	ε,	[ ],	$0:1$,	ε)
(2,	ε,	$[1/0]$,	1,	ε)
(3,	ε,	$[1/0, 2/1]$,	ε,	ε)
(4,	0,	$[1/0, 2/1]$,	ε,	ε)
(5,	$1:0$,	$[1/0, 2/1]$,	ε,	ε)
(6,	$0:1:0$,	$[1/0, 2/1]$,	ε,	ε)
(7,	$1:0$,	$[1/0, 2/1]$,	ε,	ε)
(8,	0,	$[1/0, 2/1]$,	ε,	ε)
(5,	0,	$[1/0, 2/1]$,	ε,	ε)
(6,	$0:0$,	$[1/0, 2/1]$,	ε,	ε)
(7,	0,	$[1/0, 2/1]$,	ε,	ε)
(9,	ε,	$[1/0, 2/1]$,	ε,	ε)
(10,	ε,	$[1/0, 2/1]$,	ε,	1)

Befehle der AM_1

Die Befehle der AM_0 werden durch folgende Änderungen ergänzt:

Befehl	**Auswirkung**
LOAD(b, o)	Lädt den Inhalt von Adresse $adr(r, b, o)$ auf den Datenkeller und inkrementiert den Befehlszähler.
STORE(b, o)	Nimmt das oberste Element vom Datenkeller, speichert dieses an Adresse $adr(r, b, o)$ und inkrementiert den Befehlszähler.
WRITE(b, o)	Schreibt den Inhalt an Adresse $adr(r, b, o)$ auf das Ausgabeband und inkrementiert den Befehlszähler.
READ(b, o)	Liest das oberste Element vom Eingabeband, speichert es an Adresse $adr(r, b, o)$ und inkrementiert den Befehlszähler.

$b \in \{\text{global}, \text{lokal}\}$

r : aktueller REF

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

Befehle der AM_1

Befehl	**Auswirkung**
LOADI(o)	Ermittelt Wert ($= x$) an Adresse $r+o$, lädt den Inhalt von Adresse x auf den Datenkeller und inkrementiert den Befehlszähler.
STOREI(o)	Ermittelt Wert ($= x$) an Adresse $r + o$, nimmt das oberste Element vom Datenkeller, speichert dieses an Adresse x und inkrementiert den Befehlszähler.
WRITEI(o)	Ermittelt Wert ($= x$) an Adresse $r + o$, schreibt den Inhalt an Adresse x auf das Ausgabeband und inkrementiert den Befehlszähler.
READI(o)	Ermittelt Wert ($= x$) an Adresse $r+o$, liest das oberste Element vom Eingabeband, speichert es an Adresse x und inkrementiert den Befehlszähler.
LOADA(b, o)	Legt $adr(r, b, o)$ auf den Datenkeller und inkrementiert den Befehlszähler.

Befehle der AM_1

Befehl	**Auswirkung**
PUSH	Legt oberstes Element vom Datenkeller auf den Laufzeitkeller und inkrementiert den Befehlszähler.
CALL adr	Legt den inkrementierten Befehlszählerwert auf den Laufzeitkeller, setzt den Befehlszähler auf adr, legt aktuellen REF auf den Laufzeitkeller und ändert REF auf die Länge des Laufzeitkellers.
INIT n	Legt n-mal 0 auf den Laufzeitkeller.
RET n	Löscht alles nach dem REF-Zeiger, nimmt oberstes Element vom Laufzeitkeller und setzt dieses als REF, nimmt das nun oberste Element vom Laufzeitkeller und setzt dieses als Befehlszähler und löscht die obersten n Elemente vom Laufzeitkeller.

Übung 2 (a)

$tab_{primzerl+lDecl}$ = [primzerl/(proc, 1), z/(var, lokal, -2), i/(var, lokal, 1)]
$tab_{main+lDecl}$ = [primzerl/(proc, 1), z/(var, lokal, 1)]

trans(#include <stdio.h> void primzerl(int z)... void main() ...)

=
INIT size(ε); CALL 2; JMP 0;
blocktrans({int i; i = 2; if (z > 1) {while (z % i != 0)... }}, $tab_{primzerl}$, 1, 1)
blocktrans({int z; scanf("%d", &z); primzerl(z);}, tab_{main}, 2, 0)

=
INIT 0; CALL 2; JMP 0;
1:
INIT size(int i;);
stseqtrans(i=2; if (z > 1) {while (z % i != 0) i = i + 1; ... },
update(int i;, lokal, $tab_{primzerl}$), 1)
RET 1;
2:
INIT size(int z;);
stseqtrans(scanf("%d", &z); primzerl(z);,
update(int z;, lokal, tab_{main}), 2)
RET 0;

=
INIT 0; CALL 2; JMP 0;
1:
INIT 1;
sttrans(i = 2;, $tab_{primzerl+lDecl}$, 1.1)
sttrans(if (z > 1) {while (z % i != 0) i = i + 1; ...}, $tab_{primzerl+lDecl}$, 1.2)
RET 1;
2:
INIT 1;
sttrans(scanf("%d", &z);, $tab_{main+lDecl}$, 2.1)
sttrans(primzerl(z);, $tab_{main+lDecl}$, 2.2)
RET 0;

Übung 2 (a)

$tab_{primzerl+lDecl}$ = [primzerl/(proc, 1), z/(var, lokal, -2), i/(var, lokal, 1)]
$tab_{main+lDecl}$ = [primzerl/(proc, 1), z/(var, lokal, 1)]

sttrans(i = 2;, $tab_{primzerl+lDecl}$, 1.1)
sttrans(if (z > 1) {while (z % i != 0) i = i + 1; ...}, $tab_{primzerl+lDecl}$, 1.2)
sttrans(scanf("%d", &z);, $tab_{main+lDecl}$, 2.1)
sttrans(primzerl(z);, $tab_{main+lDecl}$, 2.2)

```
=       INIT 0; CALL 2; JMP 0;
1:      INIT 1;
        simpleexptrans(2, tab_primzerl+lDecl)
        STORE(lokal, 1);
        boolexptrans(z > 1, tab_primzerl+lDecl)
        JMC 1.2.1;
        sttrans({while (z % i != 0) i = i + 1; z = z / i; ...}, tab_primzerl+lDecl, 1.2.2)
1.2.1:  RET 1;
2:      INIT 1;
        READ(lokal, 1);
        simpleexptrans(z, tab_main+lDecl)
        PUSH;
        CALL 1;
        RET 0;
```

Übung 2 (a)

$tab_{primzerl+lDecl}$ = [primzerl/(proc, 1), z/(var, lokal, -2), i/(var, lokal, 1)]
$tab_{main+lDecl}$ = [primzerl/(proc, 1), z/(var, lokal, 1)]

```
        simpleexptrans(2, tab_primzerl+lDecl)
        boolexptrans(z > 1, tab_primzerl+lDecl)
        sttrans({while (z % i != 0) i = i + 1; z = z / i; ...}, tab_primzerl+lDecl, 1.2.2)
        simpleexptrans(z, tab_main+lDecl)

=       INIT 0; CALL 2; JMP 0;
1:      INIT 1;
        LIT 2;
        STORE(lokal, 1);
        LOAD(lokal, -2);
        LIT 1;
        GT;
        JMC 1.2.1;
        stseqtrans(while (z % i != 0) i = i + 1; z = z / i; ..., tab_primzerl+lDecl, 1.2.2)
1.2.1:  RET 1;
2:      INIT 1;
        READ(lokal, 1);
        LOAD(lokal, 1);
        PUSH;
        CALL 1;
        RET 0;
```

Übung 2 (a)

$tab_{primzerl+lDecl}$ = [primzerl/(proc, 1), z/(var, lokal, -2), i/(var, lokal, 1)]
$tab_{main+lDecl}$ = [primzerl/(proc, 1), z/(var, lokal, 1)]

stseqtrans(while (z % i != 0) i = i + 1; z = z / i; ..., $tab_{primzerl+lDecl}$, 1.2.2)

```
=       INIT 0; CALL 2; JMP 0;
1:      INIT 1;
        LIT 2;
        STORE(lokal, 1);
        LOAD(lokal, -2);
        LIT 1;
        GT;
        JMC 1.2.1;
```
sttrans(while (z % i != 0) i = i + 1;, $tab_{primzerl+lDecl}$, 1.2.2.1)
sttrans(z = z / i;, $tab_{primzerl+lDecl}$, 1.2.2.2)
sttrans(printf("%d", i);, $tab_{primzerl+lDecl}$, 1.2.2.3)
sttrans(primzerl(z);, $tab_{primzerl+lDecl}$, 1.2.2.4)
```
1.2.1:  RET 1;
2:      INIT 1;
        READ(lokal, 1);
        LOAD(lokal, 1);
        PUSH;
        CALL 1;
        RET 0;
```

Übung 3 (a)

```
1.3.1:  LOAD(lokal, 1);
        LOADI(-2);
        LT;
        JMC 1.3.2;
        LOAD(lokal, 1);
        LIT 2;
        MUL;
        STORE(lokal, 1);
        LOAD(lokal, 1);
        PUSH;
        LOAD(lokal, -2);
        PUSH;
        CALL 2;
        JMP 1.3.1;
1.3.2:  LOAD(lokal, 1);
        STOREI(-3);
```

Übung 3 (b)

BZ	DK	LK	REF	Inp	Out
$(14,$	$\varepsilon,$	$0:0:1,$	$3,$	$4,$	$\varepsilon)$
$(15,$	$\varepsilon,$	$4:0:1,$	$3,$	$\varepsilon,$	$\varepsilon)$
$(16,$	$1,$	$4:0:1,$	$3,$	$\varepsilon,$	$\varepsilon)$
$(17,$	$\varepsilon,$	$4:0:1:1,$	$3,$	$\varepsilon,$	$\varepsilon)$
$(\ 3,$	$\varepsilon,$	$4:0:1:1:18:3,$	$6,$	$\varepsilon,$	$\varepsilon)$
$(\ 4,$	$\varepsilon,$	$4:0:1:1:18:3,$	$6,$	$\varepsilon,$	$\varepsilon)$
$(\ 5,$	$4,$	$4:0:1:1:18:3,$	$6,$	$\varepsilon,$	$\varepsilon)$
$(\ 6,$	$2:4,$	$4:0:1:1:18:3,$	$6,$	$\varepsilon,$	$\varepsilon)$
$(\ 7,$	$1,$	$4:0:1:1:18:3,$	$6,$	$\varepsilon,$	$\varepsilon)$
$(\ 8,$	$\varepsilon,$	$4:0:1:1:18:3,$	$6,$	$\varepsilon,$	$\varepsilon)$
$(\ 9,$	$4,$	$4:0:1:1:18:3,$	$6,$	$\varepsilon,$	$\varepsilon)$

Übung 3 (b)

BZ	DK	LK	REF	Inp	Out
$(10,$	$2:4,$	$4:0:1:1:18:3,$	$6,$	$\varepsilon,$	$\varepsilon)$
$(11,$	$2,$	$4:0:1:1:18:3,$	$6,$	$\varepsilon,$	$\varepsilon)$
$(12,$	$\varepsilon,$	$2:0:1:1:18:3,$	$6,$	$\varepsilon,$	$\varepsilon)$
$(18,$	$\varepsilon,$	$2:0:1,$	$3,$	$\varepsilon,$	$\varepsilon)$
$(19,$	$\varepsilon,$	$2:0:1,$	$3,$	$\varepsilon,$	$2)$
$(\ 0,$	$\varepsilon,$	$2:0:1,$	$3,$	$\varepsilon,$	$2)$