

Programmierung

08. Übungsblatt

Zeitraum: 29. Mai – 02. Juni 2017

Übung 1 (AGS 13.10)

- (a) Geben Sie für das folgende C_0 -Programm die Übersetzung in ein linearisiertes AM_0 -Programm an. Zwischenschritte der Übersetzung brauchen Sie nicht anzugeben.

```
#include <stdio.h>

int main() {
    int x, y, a;
    scanf("%i", &y);
    scanf("%i", &a);
    x = 0;
    while (x < a) {
        x = x + 1;
        y = y * y;
    }
    printf("%d", y);
    return 0;
}
```

- (b) Folgendes AM_0 -Programm sei gegeben:

1: READ 1;	6: SUB;
2: READ 2;	7: JMC 9;
3: LOAD 1;	8: JMP 5;
4: LOAD 2;	9: WRITE 2;
5: LIT 0;	

Protokollieren Sie den schrittweisen Ablauf dieses Programms auf der AM_0 mit der Anfangskonfiguration $(1, \varepsilon, [], 0 : 1, \varepsilon)$.

Übung 2 (AGS 14.2)

Gegeben sei folgendes C_1 -Programm *Primzerl*:

```
#include <stdio.h>

void primzerl(int z) {
    int i;
    i = 2;
    if (z > 1) {
        while (z % i != 0) i = i + 1;
        z = z / i;
        printf("%d", i);
        primzerl(z);
    }
}
```

```

void main() {
    int z;
    scanf("%d", &z);
    primzerl(z);
}

```

- (a) Berechnen Sie – in sinnvollen Zwischenschritten – das baumstrukturierte Programm $bPrimzerl_1 = trans(Primzerl)$ mit Hilfe der in der Vorlesung und im Skript angegebenen Übersetzungsfunktionen.
- (b) (Zusatzaufgabe) Wandeln Sie $bPrimzerl_1$ in ein Programm $Primzerl_1$ mit linearisierten Adressen um und berechnen Sie die operationelle Semantik $\mathcal{P}[[Primzerl_1]](6)$. Dokumentieren Sie den Zustand der AM_1 nach Ausführung jedes Befehls.
- (c) (Zusatzaufgabe) Stellen Sie den Aufbau des Laufzeitkellers beim zweiten Erreichen der Befehlszähleradresse 9 ähnlich den Darstellungen im Vorlesungsskript dar. Markieren Sie dabei insbesondere die Aktivierungsblöcke sowie den Referenzzeiger REF.

Übung 3 (AGS 14.18)

- (a) Gegeben sei folgendes Fragment eines C_1 -Programms mit den Funktionen f und g :

```

while(c < *b) {
    c = c * 2;
    g(c, b);
}
*a = c;

```

Übersetzen Sie die Sequenz dieser Statements in entsprechenden AM_1 -Code mit baumstrukturierten Adressen (mittels *stseqtrans*). Sie müssen keine Zwischenschritte angeben. Nehmen Sie an, die *while*-Anweisung sei das dritte Statement in f , und es sei

$tab_{f+Decl} = [f/(proc, 1), g/(proc, 2), a/(var-ref, -3), b/(var-ref, -2), c/(var, lokal, 1)]$.

- (b) Gegeben sei folgender AM_1 -Code:

```

1: INIT 1;      8: LOADI(-2);    14: READ(global, 1);
2: CALL 13;     9: LIT 2;         15: LOADA(global, 1);
3: INIT 0;     10: DIV;          16: PUSH;
4: LOADI(-2);  11: STOREI(-2);  17: CALL 3;
5: LIT 2;      12: RET 1;        18: WRITE(global, 1);
6: GT;         13: INIT 0;       19: JMP 0;
7: JMC 12;

```

Führen Sie ein schrittweises Ablaufprotokoll der AM_1 aus, ausgehend vom Startzustand $\sigma = (14, \varepsilon, 0 : 0 : 1, 3, 4, \varepsilon)$. Sie müssen nur Zellen ausfüllen, deren Wert sich im Vergleich zur letzten Zeile geändert hat.

Zusatzaufgabe 1 (AGS 14.19 ★)

- (a) Gegeben sei folgendes Fragment eines C_1 -Programms mit den Funktionen f und g :

```

if (a > 1) {
    f(&a, b);
} else {
    *b = a;
}

```

Übersetzen Sie die Sequenz dieser Statements in entsprechenden AM_1 -Code mit baumstrukturierten Adressen (mittels *stseqtrans*). Sie müssen keine Zwischenschritte angeben. Nehmen Sie an, die **if-else**-Anweisung sei das dritte Statement in *g*, und es sei

$$tab_{g+Decl} = [f/(proc, 1), g/(proc, 2), a/(var, lokal, 1), b/(var-ref, -2)].$$

(b) Gegeben sei folgender AM_1 -Code:

```

1: INIT 2;           8: LIT 7;           15: PUSH;
2: CALL 13;         9: STOREI(-3);      16: LOAD(global, 2);
3: INIT 0;          10: RET 2;          17: PUSH;
4: LOAD(lokal, -2); 11: INIT 0;         18: CALL 3;
5: LIT 0;           12: READ(global, 1); 19: WRITE(global, 1);
6: GT;              13: READ(global, 2); 20: JMP 0;
7: JMC 12;          14: LOADA(global, 1);

```

Erstellen Sie ein Ablaufprotokoll der AM_1 , indem Sie sie schrittweise ablaufen lassen. Die Startkonfiguration ist $(13, \varepsilon, 3 : 0 : 3 : 0, 4, 4, \varepsilon)$. Sie müssen nur Zellen ausfüllen, deren Wert sich im Vergleich zur letzten Zeile geändert hat.

Zusatzaufgabe 2 (AGS 13.5 ★)

Gegeben sei folgendes C_0 -Programm.

```

#include <stdio.h>

int main() {
    int x, y, z;
    scanf("%i", &x);
    scanf("%i", &y);
    if (x < y) z = x;
    else z = y;
    printf("%d", z);
    return 0;
}

```

(a) Geben Sie für dieses Programm die bereits linearisierte Übersetzung der operationellen Semantik an. Zwischenschritte brauchen Sie nicht anzugeben.

(b) Gegeben sei folgende bereits linearisierte Übersetzung eines C_0 -Programms *operation2*:

```

1: READ 1;           6: GT;              11: STORE 2;         16: JMP 4;
2: LIT 1;            7: JMC 17;          12: LOAD 1;          17: WRITE 2;
3: STORE 2;          8: LOAD 2;          13: LIT 1;
4: LOAD 1;           9: LOAD 1;          14: SUB;
5: LIT 1;            10: MUL;            15: STORE 1;

```

Die AM_0 habe bereits die folgende Konfiguration: $(4, \varepsilon, [1/2, 2/1], \varepsilon, \varepsilon)$.

Führen Sie die Berechnung der AM_0 auf der Grundlage des gegebenen linearisierten Programms solange weiter, bis der Befehlszähler ≥ 15 ist.

(c) Geben Sie für das AM_0 -Programm der Teilaufgabe ?? das C_0 -Programm *operation2* an. Verwenden Sie entsprechend der Reihenfolge der Speicherplätze 1 bis *n* die Variablennamen x_1 bis x_n .