Vorlesung Systemnahe Informatik
Sommersemester 2023

Prof. Dr. Peter Martini, Dr. Matthias Frank, Lennart Buhl M.Sc.

2. Übungszettel

Ausgabe:	Dienstag, 18. April 2023.
Abgabe:	Sonntag, 23. April 2023
Besprechung:	In den Übungen ab Montag, 24. April 2023.
Hinweis:	Abgabe erfolgt freiwillig per PDF über eCampus, siehe Hinweise in Aufgabe 1 auf dem 1. Übungszettel Sie können Kontakt zu Ihrer Tutor/in aufnehmen durch E-mail an cs4+ueb-si-XX@cs.uni-bonn.de mit XX als Gruppennummer.

Aufgabe 1: SPOZ-I-Assembler

Wir schreiben das Jahr 2042. Nach einer extremen Verschärfung der Finanzkrise können die Universitäten nur noch sehr preiswerte Rechner anschaffen. Zum Glück hat der führende Prozessorhersteller *Inrola* gerade einen neuen Chip namens *Untium* auf den Markt gebracht. Dieser Chip ist so preiswert, weil er nur noch genau eine Instruktion hat – nämlich „subtract-and-branch-on-less-zero“. Dieser Befehl hat vier Parameter, Quelle 1 ($Q1$), Quelle 2 ($Q2$), Zielregister (ZR) und Sprungadresse (SA). Wird der Befehl aufgerufen, berechnet er $Q1 - Q2$, schreibt das Ergebnis in ZR und springt dann nach SA, wenn das Ergebnis, das er in ZR geschrieben hat, kleiner als 0 ist (sonst folgt die Bearbeitung des Befehls in der folgenden Zeile). Der Untium-Prozessor verfügt über 6 Ganzzahlregister, \$1 bis \$6. Als Quelle kann entweder ein Register oder eine ganzzahlige Konstante angegeben werden, ZR ist immer ein Register. SA gibt die Nummer des Befehls an, zu dem eventuell gesprungen werden soll.
Als Studierende kommt Ihnen die Ehre zuteil, den ersten Rechner dieses Typs, den SPOZ-I zu testen. Da zu einem so frühen Zeitpunkt noch keine Compiler zur Verfügung stehen, müssen sie den 1-Befehl-Assembler benutzen. Ein Assemblerprogramm besteht aus Zeilen der Form:

$Q1, Q2, ZR, SA$ // optional kann hier ein Kommentar stehen

a) Finden Sie heraus, wie eine Addition durchgeführt werden kann. Dabei seien die Summanden in den Registern \$1 und \$2 vorgegeben. Die Summe soll nach der Berechnung in \$3 stehen.

b) Formulieren Sie eine Codepassage, die folgender for-Schleife entspricht:

```
1  for (a = const; a <= b; a+= const2) {
2          ...
3  }
```

Dabei sei a mit *const* initialisiert in \$1, b in \$2 und *const*2 in \$3 vorgegeben.

c) Implementieren Sie einen Algorithmus, der zwei ganze Zahlen gegeben in \$1 und \$2 multipliziert und das Produkt in \$3 ablegt. Achten Sie auf die Sonderbehandlung des Vorzeichens negativer ganzer Zahlen bei der Multiplikation.

Aufgabe 2: Alpha-Notation, Schleifen

Bei dieser Aufgabe stehen Ihnen ausschließlich die folgenden Befehle zur Verfügung ($k \in \mathbb{Z}$, op $\in \{+, -, *, /\}$). Sie dürfen beliebig viele Sprunglabel *label* verwenden. Sie können davon ausgehen, dass es sich bei x, y, z, $anfang$ und *ende* um entsprechende Speicherzellen handelt, die beispielsweise mit $\rho(x)$, $\rho(anfang)$, usw. angesprochen werden können.

$\alpha := \rho(i)$	$\alpha := \rho(i)$ op $\rho(j)$	$\rho(i) := \alpha$ op k
$\rho(i) := \alpha$	$\rho(i) := \rho(j)$	$\alpha := \alpha$ op k
$\alpha := \alpha$ op $\rho(i)$	$\alpha := \rho(i)$ op k	$\alpha := k$
$\alpha := \rho(i)$ op α	if $\alpha = 0$ then goto *label*	goto *label*

Geben Sie an, wie man die folgenden bekannten Schleifenkonstrukte (angelehnt an C Syntax) in α-Notation realisieren kann:

a) `do { irgendwas; } while (x != y)`

b) `for (int z = anfang; i < ende; z++) { irgendwas_anderes; }`

c) `while (x == y) { nochwas_anderes; }`

Aufgabe 3: Alpha-Notation, Quersumme berechnen

Bei dieser Aufgabe stehen Ihnen ausschließlich die folgenden Befehle zur Verfügung ($k \in \mathbb{Z}$, $i, j \in \{h_1, h_2, e, x\}$, op $\in \{+, -, *, /, \%\}$). Hierbei entspricht der Operator „/“ der ganzzahligen Division ohne Rest und „%“ dem Modulo (Divisionsrest). Sie dürfen beliebig viele Sprunglabel *label* verwenden.

$\alpha := \rho(i)$	$\rho(i) := \rho(i)$ op $\rho(j)$	$\rho(i) := \alpha$ op k
$\rho(i) := \alpha$	$\alpha := \rho(i)$ op α	$\alpha := \alpha$ op k
$\alpha := k$	if $\alpha = 0$ then goto *label*	goto *label*

Schreiben sie nun ein Programm, das die Quersumme der ganzzahligen Dezimal-Zahl, die in Speicherstelle x steht, berechnet und das Ergebnis in Speicherstelle e ablegt. Die Speicherstelle x darf nur einmal lesend verwendet werden und die Speicherstelle e darf nur einmal beschrieben werden. Sie dürfen in dieser Aufgabe maximal zwei Hilfsspeicherzellen h_1 und h_2 benutzen (siehe oben). Nach der Berechnung springen Sie zur Marke „*ende*“.