Grundbegriffe der Informatik
Aufgabenblatt 5

Matr.nr.:	
Nachname:	
Vorname:	
Tutorium:	Nr.
	Name des Tutors:

Ausgabe: 25. November 2015

Abgabe: 4. Dezember 2015, 12:30 Uhr
im GBI-Briefkasten im Untergeschoss
von Gebäude 50.34

Lösungen werden nur korrigiert, wenn sie

- rechtzeitig,
- in Ihrer eigenen Handschrift,
- mit dieser Seite als Deckblatt und
- in der oberen **linken** Ecke zusammengeheftet

abgegeben werden.

Vom Tutor auszufüllen:

erreichte Punkte

Blatt 5:	/ 18	(Physik: 18)
Blätter 1 – 5:	/ 84	(Physik: 81)

Aufgabe 5.1 (1 + 1 + 4 = 6 Punkte)
Es sei $\text{Val} = \{\mathtt{0}, \mathtt{1}\}^8$, es sei $\text{Adr} = \{\mathtt{0}, \mathtt{1}\}^{32}$ und es sei $\text{Mem} = \text{Val}^{\text{Adr}}$. Die Addition modulo 2^8 zweier Zahlen in Binärdarstellung der Länge 8 ist gegeben durch die Abbildung

$$\begin{aligned}\text{add}_{\text{Val}} \colon \text{Val} \times \text{Val} &\to \text{Val},\\ (u, v) &\mapsto \text{bin}_8((\text{Num}_2(u) + \text{Num}_2(v)) \ \mathbf{mod}\ 2^8),\end{aligned}$$

und die Addition modulo 2^{32} zweier Zahlen in Binärdarstellung der Länge 32 beziehungsweise 8 ist gegeben durch die Abbildung

$$\begin{aligned}\text{add}_{\text{Adr}} \colon \text{Adr} \times \text{Val} &\to \text{Adr},\\ (a, v) &\mapsto \text{bin}_{32}((\text{Num}_2(a) + \text{Num}_2(v)) \ \mathbf{mod}\ 2^{32}).\end{aligned}$$

Ein Stapel ist eine Datenstruktur mit drei grundlegenden Operationen:
a) „push“ legt einen Wert auf den Stapel;
b) „pop“ nimmt den zuoberst liegenden Wert vom Stapel;
c) „peek“ liefert den zuoberst liegenden Wert, ohne ihn vom Stapel zu nehmen.

In unserem Speichermodell kann ein Stapel mit höchstens $(2^8 - 1)$-vielen Werten durch eine Adresse repräsentiert werden, deren Wert die Anzahl der Werte auf dem Stapel in Binärdarstellung ist und deren Folgeadressen die Werte auf dem Stapel enthalten. Die Abbildungen init_stack, is_empty, push, pop und peek bilden eine Schnittstelle zur Verwaltung von Stapeln und sind gegeben durch:

$$\begin{aligned}\text{init\_stack} \colon \text{Mem} \times \text{Adr} &\to \text{Mem},\\ (m, a) &\mapsto \text{memwrite}(m, a, \text{bin}_8(0)),\end{aligned}$$

$$\begin{aligned}\text{is\_empty} \colon \text{Mem} \times \text{Adr} &\to \mathbb{B},\\ (m, a) &\mapsto \begin{cases} \mathbf{w}, & \text{falls memread}(m, a) = \text{bin}_8(0),\\ \mathbf{f}, & \text{sonst},\end{cases}\end{aligned}$$

$$\begin{aligned}\text{push} \colon \text{Mem} \times \text{Adr} \times \text{Val} &\to \text{Mem},\\ (m, a, v) &\mapsto \text{memwrite}(m', \text{add}_{\text{Adr}}(a, \text{memread}(m', a)), v),\\ &\qquad \text{wobei } m' = \text{memwrite}(m, a, \text{add}_{\text{Val}}(\text{memread}(m, a), \text{bin}_8(1))),\end{aligned}$$

$$\begin{aligned}\text{pop} \colon \text{Mem} \times \text{Adr} &\to \text{Mem},\\ (m, a) &\mapsto \begin{cases} m, & \text{falls is\_empty}(m, a),\\ \text{memwrite}(m, a, \text{add}_{\text{Val}}(\text{memread}(m, a), \text{bin}_8(2^8 - 1))), & \text{sonst},\end{cases}\end{aligned}$$

$$\begin{aligned}\text{peek} \colon \text{Mem} \times \text{Adr} &\to \text{Val},\\ (m, a) &\mapsto \text{memread}(m, \text{add}_{\text{Adr}}(a, \text{memread}(m, a))).\end{aligned}$$

Für jeden Speicher $m \in \mathrm{Mem}$, jede Adresse $a \in \mathrm{Adr}$ und jeden Wert $v \in \mathrm{Val}$, initislisiert init_stack(m, a) einen Stapel bei a in m, prüft is_empty(m, a), ob der Stapel bei a in m leer ist oder nicht, legt push(m, a, v) den Wert v auf den Stapel bei a in m, nimmt pop(m, a) den zuoberst liegenden Wert des Stapels bei a in m und liefert peek(m, a) den zuoberst liegenden Wert des Stapels bei a in m.

a) Es sei $m \in \mathrm{Mem}$ und es sei $a = \mathrm{bin}_{32}(0)$. Geben Sie den Wert

$$\mathrm{peek}(\mathrm{pop}(\mathrm{push}(\mathrm{push}(\mathrm{init\_stack}(m, a), \texttt{00101111}), a, \texttt{00001100}), a), a)$$

an.

b) Es sei $m \in \mathrm{Mem}$, es sei $a = \mathrm{bin}_{32}(0)$ und es sei

$$m' = \mathrm{push}(\mathrm{push}(\mathrm{init\_stack}(m, a), \texttt{11111111}), a, \texttt{00000001}).$$

Geben Sie den Wert $\mathrm{add}_{\mathrm{Val}}(\mathrm{peek}(m', a), \mathrm{peek}(\mathrm{pop}(m', a), a))$ an.

c) Definieren Sie induktiv, unter ausschließlicher Verwendung der Abbildungen $\mathrm{add}_{\mathrm{Val}}$, is_empty, pop und peek, eine Abbildung $\mathrm{sum}\colon \mathrm{Mem} \times \mathrm{Adr} \to \mathrm{Val}$ derart, dass für jeden Speicher $m \in \mathrm{Mem}$ und jede Adresse $a \in \mathrm{Adr}$ gilt, dass sum(m, a) die Binärdarstellung der Summe modulo 2^8 aller Werte, interpretiert als Binärdarstellungen von Zahlen, auf dem Stapel bei a in m ist, wobei die leere Summe per Definition 0 ist.

Lösung 5.1

a) `00101111`

b) `00000000`

c)

$$\begin{aligned}
&\mathrm{sum}\colon \mathrm{Mem} \times \mathrm{Adr} \to \mathrm{Val},\\
&\qquad (m, a) \mapsto \begin{cases} \mathrm{peek}(m, a), & \text{falls } \mathrm{is_empty}(m, a) = \mathbf{w},\\ \mathrm{add}_{\mathrm{Val}}(\mathrm{sum}(\mathrm{pop}(m, a), a), \mathrm{peek}(m, a)), & \text{sonst.} \end{cases}
\end{aligned}$$

Aufgabe 5.2 (4 Punkte)

Es seien a_1 und a_2 zwei verschiedene 20bit Adressen. Im Speicher stehe in Adresse a_1 die Zweierkomplementdarstellung einer nicht-negativen ganzen Zahl x, für die 2^x mit 24bit in Zweierkomplementdarstellung darstellbar ist. Ergänzen Sie die fehlenden Konstanten und Adressen im unvollständigen Minimalmaschinenprogramm

$$
\begin{array}{rll}
 & \texttt{LDC} & \boxed{} \\
 & \texttt{STV} & \boxed{} \\
\textit{while}: & \texttt{LDC} & \boxed{} \\
 & \texttt{NOT} & \\
 & \texttt{ADD} & \boxed{} \\
 & \texttt{STV} & \boxed{} \\
 & \texttt{JMN} & \textit{end} \\
 & \texttt{LDV} & \boxed{} \\
 & \texttt{ADD} & \boxed{} \\
 & \texttt{STV} & \boxed{} \\
 & \texttt{JMP} & \textit{while} \\
\textit{end}: & \texttt{HALT} &
\end{array}
$$

derart, dass nach dessen Ausführung 2^x in Zweierkomplementdarstellung im Speicher bei Adresse a_2 steht. Beachten Sie, dass alle arithmetischen Ausdrücke, in denen x vorkommt, keine Konstanten sind, und, dass $2^0 = 1$ gilt.

Lösung 5.2
Es gilt $2^0 = 1$ und für jedes $n \in \mathbb{N}_0$ gilt $2^{n+1} = 2 \cdot 2^n = 2^n + 2^n$. Somit gilt $2^1 = 2^0 + 2^0 = 1 + 1$, $2^2 = 2^1 + 2^1 = (1+1) + (1+1)$, $2^3 = 2^2 + 2^2 = [(1+1) + (1+1)] + [(1+1) + (1+1)]$, und so weiter. Initialisieren wir den Wert bei a_2 mit 1 und wiederholen wir x-mal, dass wir den Wert bei a_2 mit sich selbst addieren und das Ergebnis bei a_2 ablegen, so ist der Wert bei a_2 nach der nullten Wiederholung 2^0, nach der ersten Wiederholung 2^1, nach der zweiten Wiederholung 2^2, und nach der x-ten Wiederholung 2^x. Ein Minimalmaschinenprogramm für diesen

Algorithmus ist das Folgende:

```
       LDC 1
       STV a2
while: LDC 0
       NOT
       ADD a1
       STV a1
       JMN end
       LDV a2
       ADD a2
       STV a2
       JMP while
end:   HALT
```

Aufgabe 5.3 (4 + 4 = 8 Punkte)
Für jede positive ganze Zahl $n \in \mathbb{N}_+$ ist der ganzzahlige binäre Logarithmus von n, geschrieben $\lfloor \log_2 n \rfloor$, jene nicht-negative ganze Zahl $k \in \mathbb{N}_0$, für die $2^k \leq n < 2^{k+1}$ gilt. Es seien a_1, a_2 und a_3 drei paarweise verschiedene 20bit Adressen.

a) Im Speicher stehe in Adresse a_1 die Zweierkomplementdarstellung einer positiven ganzen Zahl x_1. Schreiben Sie ein Minimalmaschinenprogramm, nach dessen Ausführung die Zweierkomplementdarstellung von x_1 **div** 2 im Akkumulator steht und das höchstens die Adressen a_1 und a_2 verwendet.

b) Im Speicher stehe in Adresse a_1 die Zweierkomplementdarstellung einer positiven ganzen Zahl x_2. Schreiben Sie ein Minimalmaschinenprogramm, nach dessen Ausführung die Zweierkomplementdarstellung von $\lfloor \log_2 x_2 \rfloor$ im Speicher bei Adresse a_3 steht. Dabei dürfen Sie das Programm aus der vorangegangenen Teilaufgabe verwenden, indem sie `DIV` a_1 dort im Programm schreiben, wo das Programm aus der vorangegangenen Teilaufgabe Zeichen für Zeichen ohne den (letzten) Befehl `HALT` eingefügt werden soll.
Hinweise:
- Für jedes $k \in \mathbb{N}_0$ gilt genau dann $k = \lfloor \log_2 x_2 \rfloor$, wenn x_2 **div** $2^k = 1$.
- Für jedes $y \in \mathbb{N}_0$ gilt $x_2 \textbf{ div } 2^{y+1} = (x_2 \textbf{ div } 2) \textbf{ div } 2^y$.
- Auch wenn Sie für Teilaufgabe a) keine Lösung gefunden haben, können Sie Teilaufgabe b) bearbeiten.

Lösung 5.3

a) Die Zweierkomplementdarstellung u von x_1 **div** 2 ist gleich der Zweierkomplementdarstellung w von x_1 ohne das niederwertigste Bit. Zur Berechnung von u rotieren wir w um eins nach rechts und setzen das höchstwertige Bit auf `0`. Letzteres erreichen wir dadurch, dass wir den rotierten

Wert bitweise mit `011...1` verunden. Das Bitmuster `011...1` bekommen wir indem wir `111...1` bitweise mit `000...01` exklusiv verodern und um eins nach rechts rotieren. Als Minimalmaschinenprogramm:

```
LDV a1
RAR
STV a1
LDC 0
NOT
STV a2
LDC 1
XOR a2
RAR
AND a1
HALT
```

b) Wir berechnen nacheinander x_2, x_2 **div** 2, $(x_2$ **div** $2)$ **div** 2, und so weiter, bis wir das erste Mal 1 erhalten. Außerdem merken wir uns die Anzahl durchgeführter Ganzzahldivisionen. Als Minimalmaschinenprogramm:

```
       LDC 0
       STV a3
while: LDC 1
       EQL a1
       JMN end
       DIV a1
       STV a1
       LDC 1
       ADD a3
       STV a3
       JMP while
  end: HALT
```