

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 $\text{Val} = \{0, 1\}^8$, es sei $\text{Adr} = \{0, 1\}^{32}$ 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

$$\begin{aligned} \text{add}_{\text{Val}}: \text{Val} \times \text{Val} &\rightarrow \text{Val}, \\ (u, v) &\mapsto \text{bin}_8((\text{Num}_2(u) + \text{Num}_2(v)) \bmod 2^8), \end{aligned}$$

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

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

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:

$$\begin{aligned} \text{init_stack}: \text{Mem} \times \text{Adr} &\rightarrow \text{Mem}, \\ (m, a) &\mapsto \text{memwrite}(m, a, \text{bin}_8(0)), \end{aligned}$$

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

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

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

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

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

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

$$\text{peek}(\text{pop}(\text{push}(\text{push}(\text{init_stack}(m, a), a, 00101111), a, 00001100), a), a)$$

an.

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

$$m' = \text{push}(\text{push}(\text{init_stack}(m, a), a, 11111111), a, 00000001).$$

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

c) Definieren Sie induktiv, unter ausschließlicher Verwendung der Abbildungen add_{Val} , is_empty , pop und peek , eine Abbildung $\text{sum}: \text{Mem} \times \text{Adr} \rightarrow \text{Val}$ derart, dass für jeden Speicher $m \in \text{Mem}$ und jede Adresse $a \in \text{Adr}$ gilt, dass $\text{sum}(m, a)$ 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)

$\text{sum}: \text{Mem} \times \text{Adr} \rightarrow \text{Val}$,

$$(m, a) \mapsto \begin{cases} \text{peek}(m, a), & \text{falls } \text{is_empty}(m, a) = \mathbf{w}, \\ \text{add}_{\text{Val}}(\text{sum}(\text{pop}(m, a), a), \text{peek}(m, a)), & \text{sonst.} \end{cases}$$

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 Minimalmaschi-

nenprogramm

```
LDC   
STV   
while: LDC   
NOT  
ADD   
STV   
JMN end  
LDV   
ADD   
STV   
JMP while  
end: HALT
```

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
```

Korrektur: Es wird Lösungen geben, in denen auch MIMA-Befehle geändert sind (LDV statt LDC, usw.)

Wie bewerten wir das?

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.

- 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 $x_1 \mathbf{div} 2$ im Akkumulator steht und das höchstens die Adressen a_1 und a_2 verwendet.
- 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 a1` 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 $x_2 \mathbf{div} 2^k = 1$.
- Für jedes $y \in \mathbb{N}_0$ gilt $x_2 \mathbf{div} 2^{y+1} = (x_2 \mathbf{div} 2) \mathbf{div} 2^y$.
- Auch wenn Sie für Teilaufgabe a) keine Lösung gefunden haben, können Sie Teilaufgabe b) bearbeiten.

Lösung 5.3

- a) Die Zweierkomplementdarstellung u von $x_1 \mathbf{div} 2$ 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 $011\dots 1$ verunden. Das Bitmuster $011\dots 1$ bekommen wir indem wir $111\dots 1$ bitweise mit $000\dots 01$ 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 , $x_2 \mathbf{div} 2$, $(x_2 \mathbf{div} 2) \mathbf{div} 2$, 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
```