Karlsruher Institut für Technologie

aifb
Institut für Angewandte Informatik
und Formale Beschreibungsverfahren

25.07.2016

Lösung zur Klausur über den Stoff der Vorlesung
"Grundlagen der Informatik II"
(90 Minuten)

Name: _____ Vorname: _____

Matr.-Nr.: _____ Semester: _____ (SS 2016)

Ich bestätige, dass ich die folgenden Angaben gelesen und mich von der Vollständigkeit dieses Klausurexemplars überzeugt habe (Seiten 1-22).

_____

Unterschrift des o. g. Klausurteilnehmers bzw. der o. g. Klausurteilnehmerin

Anmerkungen:

1. Legen Sie bitte Ihren Studierendenausweis bereit.
2. Bitte tragen Sie **Name**, **Vorname** und **Matr.-Nr.** deutlich lesbar ein.
3. Die folgenden **10 Aufgaben** sind vollständig zu bearbeiten.
4. Folgende Hilfsmittel sind zugelassen: **keine.**
5. Täuschungsversuche führen zum Ausschluss von der Klausur.
6. Unleserliche oder mit Bleistift geschriebene Lösungen können von der Klausur bzw. Wertung ausgeschlossen werden.
7. Die Bearbeitungszeit beträgt 90 Minuten.

Nur für den Prüfer :

1	2	3	4	5	6	7	8	9	10	-	-	-	-	-	-	gesamt
(8)	(8)	(9)	(8)	(9)	(10)	(10)	(12)	(8)	(8)							(90)

Aufgabenübersicht

Aufgabe 1 **8 Punkte**

2016-N-01 **Chomsky-Normalform**

/ 8

Gegeben sei die folgende kontextfreie Grammatik:

$$G = (\{A, B, C, D, S\}, \{a, b\}, P, S)$$

$$\begin{aligned} P = \{S &\to a \mid Db \mid aD \mid BAa, \\ A &\to a \mid b \mid Ca \mid Sb, \\ B &\to b \mid Bb, \\ C &\to \lambda \mid Cb, \\ D &\to a \mid b \mid Ca \mid SbBS\} \end{aligned}$$

(a) Wandeln Sie die Grammatik um in die **Chomsky-Normalform (CNF)**.

/ 5

(b) Geben Sie **einen Ableitungsbaum** für das Wort $aabbab \in L(G)$ an:

- Entweder mit der Grammatik G;
- Oder mit Ihrer selbsterstellten CNF-Grammatik aus Aufgabenteil (a).

/ 3

Lösung:

*** Lambda-free grammar ***

$$G = (\{A, B, C, D, S\}, \{a, b\}, P, S)$$

$$\begin{aligned} P = \{S &\to a \mid Db \mid aD \mid BAa, \\ A &\to a \mid b \mid Ca \mid Sb, \\ B &\to b \mid Bb, \\ C &\to b \mid Cb, \\ D &\to a \mid b \mid Ca \mid SbBS\} \end{aligned}$$

*** Grammar without chains ***

$$G = (\{A, B, C, D, S\}, \{a, b\}, P, S)$$

$$\begin{aligned} P = \{S &\to a \mid Db \mid aD \mid BAa, \\ A &\to a \mid b \mid Ca \mid S\,b, \\ B &\to b \mid Bb, \\ C &\to b \mid Cb, \\ D &\to a \mid b \mid Ca \mid S\,bBS\,\} \end{aligned}$$

*** Grammar with all terminals isolated ***

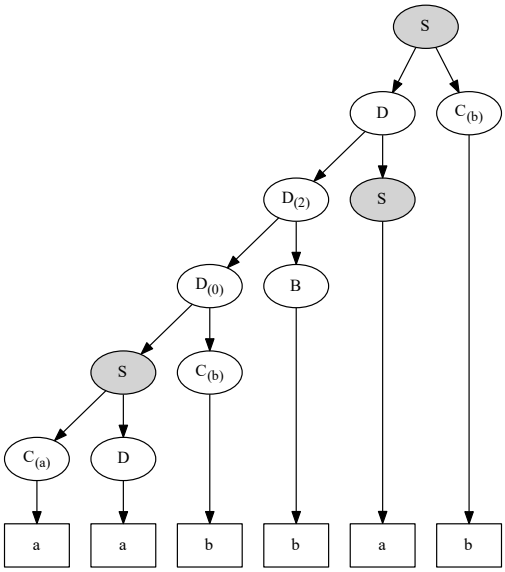
$$G = (\{A, B, C, C_{(a)}, C_{(b)}, D, S\,\}, \{a, b\}, P, S\,)$$

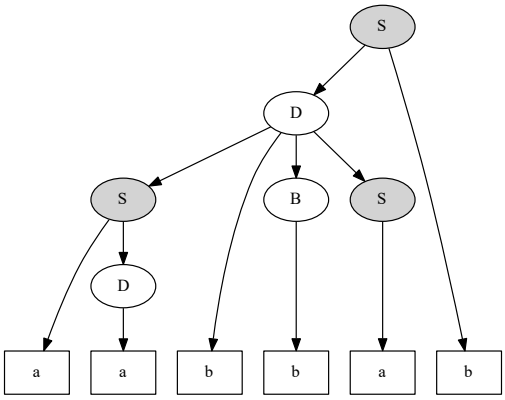
$$\begin{aligned} P = \{S &\to a \mid C_{(a)}D \mid DC_{(b)} \mid BAC_{(a)}, \\ A &\to a \mid b \mid CC_{(a)} \mid S\,C_{(b)}, \\ B &\to b \mid BC_{(b)}, \\ C &\to b \mid CC_{(b)}, \\ D &\to a \mid b \mid CC_{(a)} \mid S\,C_{(b)}BS, \\ C_{(a)} &\to a, \\ C_{(b)} &\to b\} \end{aligned}$$

*** CNF grammar ***

$$G = (\{A, B, C, C_{(a)}, C_{(b)}, D, D_{(0)}, D_{(1)}, D_{(2)}, S\,\}, \{a, b\}, P, S\,)$$

$$\begin{aligned} P = \{S &\to a \mid C_{(a)}D \mid DC_{(b)} \mid D_{(1)}C_{(a)}, \\ A &\to a \mid b \mid CC_{(a)} \mid S\,C_{(b)}, \\ B &\to b \mid BC_{(b)}, \\ C &\to b \mid CC_{(b)}, \\ D &\to a \mid b \mid CC_{(a)} \mid D_{(2)}S, \\ C_{(a)} &\to a, \\ C_{(b)} &\to b, \\ D_{(0)} &\to S\,C_{(b)}, \\ D_{(1)} &\to BA, \\ D_{(2)} &\to D_{(0)}B\} \end{aligned}$$


S
D
C(b)
D(2)
S
D(0)
B
S
C(b)
C(a)
D
a
a
b
b
a
b


S
D
S
B
S
D
a
a
b
b
a
b

Aufgabe 2 **8 Punkte**

2016-N-02 **Reguläre Sprachen**

/ 8

Der Shuffle zweier Worte $u = u_1 \dots u_m$ und $v = v_1 \dots v_n$ besteht aus allen Wörtern, die sich durch Mischen der Zeichen von u und v erzeugen lassen, wobei die relative Ordnung der Zeichen in den Wörtern erhalten bleibt:

$$\begin{aligned} \lambda \odot v &= \{v\} \\ u \odot \lambda &= \{u\} \\ u \odot v &= u_1(u_2 \dots u_m \odot v) \cup v_1(u \odot v_2 \dots v_n) \end{aligned}$$

Sei beispielsweise $u = ab$ und $v = xy$, dann gilt:

$$u \odot v = \{abxy, axby, xaby, axyb, xayb, xyab\}.$$

Der Shuffle zweier Sprachen L_1 bzw. L_2 besteht aus allen Wörtern, die sich durch Shuffle aus Wörtern jeweils aus L_1 und L_2 erzeugen lassen:

$$L_1 \odot L_2 = \bigcup_{u \in L_1, v \in L_2} u \odot v.$$

Begründen Sie, dass wenn L_1 und L_2 **regulär** sind, $L_1 \odot L_2$ ebenfalls **regulär** ist.

Hinweis: Nehmen Sie an, Sie kennen zwei deterministische endliche Automaten A_1 und A_2, die jeweils L_1 und L_2 erkennen. Nutzen Sie A_1 und A_2, um einen nichtdeterministischen endlichen Automaten zu beschreiben, der $L_1 \odot L_2$ erkennt.

Lösung:

Seien $A_1 = (E_1, S_1, \delta_1, s_{01}, F_1)$ und $A_1 = (E_2, S_2, \delta_2, s_{02}, F_2)$. Der Automat A_3 mit $L(A_3) = L_1 \odot L_2$, kann wie folgt definiert werden:

$$A_3 = (E_1 \cup E_2, S_1 \times S_2, \delta_3, (s_{01}, s_{02}), F_1 \times F_2)$$

mit

$$\delta_3((s_1, s_2), e) = \{(\delta_1(s_1, e), s_2), (s_1, \delta_2(s_2, e))\},$$

wobei $(s_1, s_2) \in S_1 \times S_2$ und $e \in E_1 \cup E_2$.

Aufgabe 3 **9 Punkte**

2016-N-03 **Nichtdeterministischer Kellerautomat**

/ 9

Gegeben sei die Sprache aller Wörter $w \in \{0, 1, \$\}^\star$, bei denen **auf jede 0 oder auf jede 1 ein \$ folgt**.

Das heißt, dass in einem Wort der Sprache auf mindestens eines der Zeichen 0, 1 **immer** ein \$ folgen muss; hinter dem jeweiligen anderen Zeichen darf \$ folgen, muss aber nicht. Ansonsten ist die Verteilung von 0, 1 und \$ beliebig. Formal:

$$L = \{w \in \{0, 1, \$\}^\star \mid \exists a \in \{0, 1\} : \forall u, v, b \in \{0, 1, \$\}^\star, |b| = 1 : w = uabv \implies b = \$\}$$

Es gilt beispielsweise:

$$\lambda, 0, 1, 0\$, 1\$, 0\$\$\$, 0\$1, \$01\$, 0\$111\$0\$111, 0\$1\$0\$\$, \$\$\$ \in L,$$
$$01, 10, 0011, 01, 10\$11\$00 \notin L$$

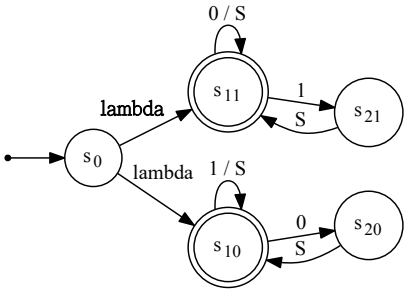
Entwerfen Sie eine **nichtdeterministischen Kellerautomaten**

$$A = \left(\{0, 1, \$\}, \underbrace{\qquad\qquad}_{S}, \delta, \underbrace{\quad}_{s_0}, \underbrace{\qquad\qquad}_{F} \right)$$

welcher die Sprache L erkennt.

Hinweis: Sie müssen den Keller nicht unbedingt benutzen, um die Sprache zu erkennen. Geben Sie aber auf jeden Fall einen Kellerautomaten an (obwohl prinzipiell auch ein endlicher Automat ausreichend wäre).

Lösung: Da der Keller nicht genutzt werden muss, reduziert sich der KA zu einem EA, indem er beim Lesen aus dem Keller immer k_0 voraussetzt, den Kellerinhalt also ignoriert, und auch immer k_0 wieder hineinschreibt. Wir geben daher nur die Zustandsübergänge an.



SKRIPT ID-16551

Aufgabe 4	**8 Punkte**
2016-N-04	**Komplexität**

/ 8

Eine **eindeutige Turingmaschine** ist eine Turingmaschine, die in jeder Kombination aus Zustand und Eingabesymbol potentiell mehrere Folgekonfigurationen besitzt, aber eine Eingabe genau dann akzeptiert, wenn nur **ein einziger Pfad** im Berechnungsbaum in einen zulässigen Endzustand mündet.

Bezeichne UP (Unique Polynomial Time) die Klasse der Entscheidungsprobleme, die von einer eindeutigen Turingmaschine in **Polynomialzeit** gelöst werden können.

Hinweis: Überlegen Sie sich, wie sich eindeutige Turingmaschinen von deterministischen und nichtdeterministischen Turingmaschinen unterscheiden.

(a) Begründen Sie, warum $P \subseteq UP$ gilt.

/ 4

Lösung:

- Die deterministische Turingmaschine ist ein Spezialfall der eindeutigen Turingmaschine, wenn diese für jede Kombination aus Zustand und Eingabesymbol maximal eine Folgekonfiguration besitzt.
- Folglich kann eine deterministische Turingmaschine durch eine eindeutige Turingmaschine simuliert werden.
- Hierdurch ergibt sich, dass die eindeutige Turingmaschine alle Entscheidungsprobleme in Polynomialzeit lösen kann, die auch von einer deterministischen Turingmaschine in Polynomialzeit gelöst werden kann.

(b) Begründen Sie, warum $UP \subseteq NP$ gilt.

/ 4

Lösung:

- Die eindeutige Turingmaschine ist ein Spezialfall der nichtdeterministischen Turingmaschine, wenn diese eine Eingabe genau dann akzeptiert, wenn nur ein einziger Pfad im Berechnungsbaum in einen zulässigen Endzustand mündet.
- Folglich kann jede eindeutige Turingmaschine durch eine nichtdeterministische Turingmaschine simuliert werden.
- Hierdurch ergibt sich, dass die nichtdeterministische Turingmaschine alle Entscheidungsprobleme in Polynomialzeit lösen kann, die auch von einer eindeutigen Turingmaschine in Polynomialzeit gelöst werden kann.

Aufgabe 5 **9 Punkte**

2016-N-05 **Binary Decision Diagram**

/ 9

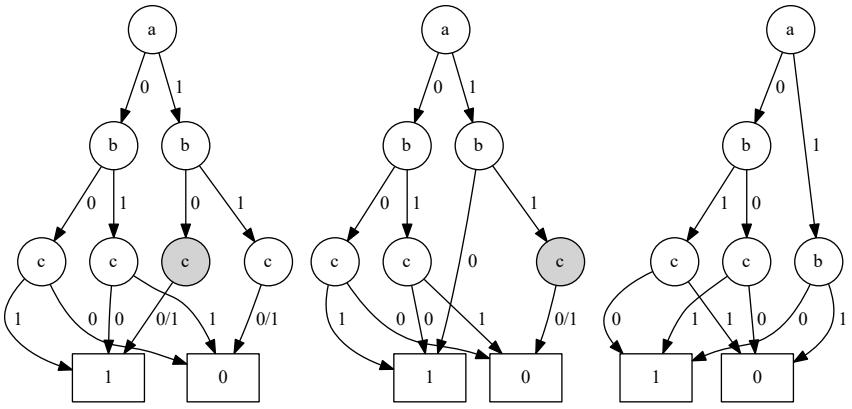
Gegeben sei die Funktion $f : \mathbb{B}^3 \to \mathbb{B}$ durch folgende Wahrheitstabelle:

a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(a) Erstellen Sie das zu f gehörende BDD bei Variablenreihenfolge $a \to b \to c$.

/ 5

Lösung:



SKRIPT ID-16803

(b) Wie könnte eine Funktion $f' : \mathbb{B}^3 \to \mathbb{B}$ aussehen, die sich als BDD **möglichst stark** komprimieren lässt?

/ 1

a	b	c	f'
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tragen Sie Ihren Vorschlag für f' in die Wahrheitsabelle ein.

Lösung:

Alle Tabelleneinträge haben entweder den Wert 0 oder 1.

(c) Wie viele Vereinfachungsschritte (beider Typen laut Algorithmus aus der Vorlesung) werden benötigt, um das BDD zu Ihrer Funktion f' zu erzeugen? Begründen Sie kurz und geben Sie **nur das fertige BDD** an.

/ 3

Lösung:

- Insgesamt sind 15 Vereinfachungsschritte durchzuführen.
- Hierzu werden gleiche Knoten und gleiche Ebenen zusammengefasst.

1

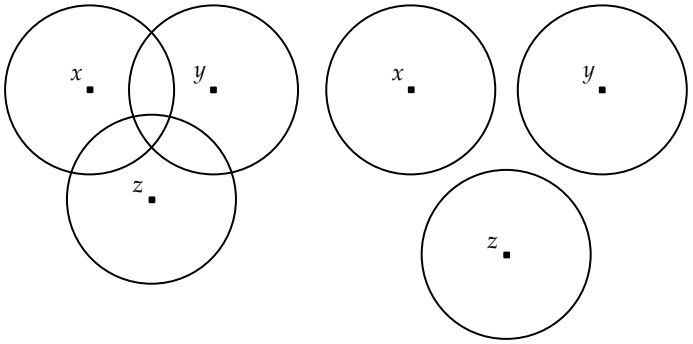
Je nach Lösung in Aufgabenteil (b) kann im BDD alternativ der Wert 0 stehen.

Aufgabe 6 **10 Punkte**

2016-H-06 **Fehlererkenn- und ~korrigierbarkeit**

/ 10

Gegeben seien folgende zwei schematische Darstellungen der drei Codewörter $x, y, z \in \{0,1\}^8$ und ihres "Abstands" voneinander im Rahmen eines Codes. Erklären Sie den Zusammenhang zwischen den euklidischen Abständen in der Abbildung und den Hammingabständen zwischen den Codewörtern in Bezug auf die Behandelbarkeit von Fehlern anhand der folgenden Teilaufgaben.



Erkennbarkeit	x	
Korrigierbarkeit		x

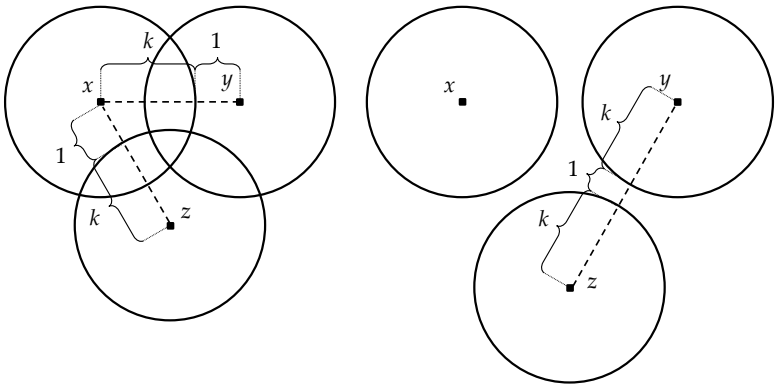
(a) Kreuzen Sie zunächst in der vorgegebenen Tabelle unterhalb der beiden Darstellungen an, ob diese jeweils für die Erkennbarkeit oder die Korrigierbarkeit von Fehlern steht.

/ 1

(b) Zeichnen Sie in die Abbildung die für die Fehlerbehandlung relevanten Abstände ein (es reicht jeweils ein Beispiel zur Verdeutlichung). Beschriften Sie deutlich.

/ 3

Lösung:



(c) Wie lassen sich aus den Abbildungen die Formeln für die k-Fehlererkennbarkeit bzw. die k-Fehlerkorrigierbarkeit von Codes herleiten? Geben Sie die Formeln an und erklären Sie kurz.

/ 4

Lösung:

- Fehlererkennbarkeit:
  - euklidische Abstand zwischen den Codewörtern x, y und z repräsentiert den Hammingabstand der Wörter
  - (liegt kein Codewort in einem Kreis (mit Radius k) um ein anderes Codewort,) ergibt sich die Hammingzahl zu $h_c \geq k + 1$.
  - Kodierung ist dann mindestens k-fehlererkennbar, da eine Veränderung von höchstens k Bits an einem Codewort nicht ein anderes Codewort ergeben kann.
  - daraus folgt: $h_c = k + 1$, also $k = h_c - 1$.
- Fehlerkorrigierbarkeit:
  - alle Kreise mit Radius k um die Codewörter sind hier disjunkt / Hammingzahl berechnet sich aus dem euklidischen Abstand zwischen den Codewörtern
  - Hammingzahl ergibt sich zu $h_c \geq 2k + 1$
  - Kodierung ist mindestens k-fehlerkorrigierbar, da eine Veränderung von höchstens k Bits an einem Codewort nicht zu einem Wort führen kann, das durch Veränderung von höchstens k Bits aus einem anderen Codewort entstanden ist.
  - daraus folgt: $h_c = 2k + 1$, also $k = \left\lfloor \frac{h_c - 1}{2} \right\rfloor$.

(d) Die Codewörter seien nun gegeben als

x	0 0 0 0 1 1 1 1
y	0 0 1 1 1 1 0 0
z	1 1 1 1 0 0 0 0

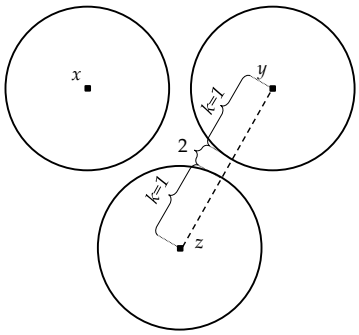
Zeichnen Sie ein neues Diagramm, das die **maximale Korrigierbarkeit** dieser Codewörter darstellt. Wie groß sind die entsprechenden Abstände?

/ 2

Lösung:

Darstellung für die konkreten Werte:

- Hammingabstand: $h_c = 4$
- Korrigierbarkeit: $k = 1$ (mit Abstand 5 wäre es schon $k = 2$, der Abstand zwischen den Kreisbögen wäre dann 1)



Aufgabe 7 **10 Punkte**

2016-N-07 **Zahlendarstellung**

/ 10

(a) Subtrahieren Sie $B = (33)_{10}$ von $A = (132)_{10}$ **bitweise** in 12-Bit Dualdarstellung.

/ 2

Lösung:

A_2	0	0	0	0	1	0	0	0	0	1	0	0
B_2	0	0	0	0	0	0	1	0	0	0	0	1
c	0	0	0	0	1	1	0	0	0	1	1	0
$(A-B)_2$	0	0	0	0	0	1	1	0	0	0	1	1

(b) Subtrahieren Sie $C = (50)_{10}$ von $A = (132)_{10}$ **bitweise** in 12-Bit 2-Komplementdarstellung.

/ 2

Lösung:

A_{2K}	0	0	0	0	1	0	0	0	0	1	0	0
C_{2K}	1	1	1	1	1	1	0	0	1	1	1	0
c	1	1	1	1	0	0	0	1	1	0	0	0
$(A-C)_{2K}$	0	0	0	0	0	1	0	1	0	0	1	0

(c) Wie lautet, dezimal, die betragsmäßig größte Zahl, die mittels einer binären 8-Bit Festkommazahl mit einem Vorzeichenbit, 5 Vorkommastellen und 2 Nachkommastellen dargestellt werden kann?

/ 2

Lösung:

$(0|1)1111111 = 16 + 8 + 4 + 2 + 1 + 0,5 + 0,25 = (+/-)31,75$

(d) Die Zahl $N = 1,5 \cdot 2^5$ kann im Gleitkommaformat mit einem Vorzeichenbit, 5 Bit Charakteristik und 3 Bit Mantisse folgendermaßen dargestellt werden:

$$V = 0; c = 10100; m' = 100$$

Multiplizieren Sie N mit sich selbst, ohne Umweg über die Dezimaldarstellung. Unter Berücksichtigung der Formel, mittels derer normierte Gleitpunktzahlen in Dezimalzahlen umgerechnet werden, führen Sie dafür die folgenden Schritte aus:

- Multiplizieren sie die Mantisse (nicht nur m', sondern m) **bitweise** mit sich selbst.
- Errechnen Sie die Charakteristik aus c und q der ursprünglichen Zahl.
- Bestimmen Sie das neue Vorzeichenbit.
- Geben Sie das Ergebnis N^2 im selben Format an, wie ursprünglich N.

/ 4

Lösung:

Multiplikation der Mantissen:

	1,	1	0	0	·	1,	1	0	0
	1	1	0	0					
		1	1	0	0				
			0	0	0	0			
				0	0	0	0		
1	0,	0	1	0	0	0	0		

Charakteristik (vor Normierung):
$c_{neu} = 2 \cdot c_{alt} - q = 2 \cdot (10100)_2 - q = (101000)_2 - q$

	1	0	1	0	0	0
−	0	0	1	1	1	1
	0	1	1	0	0	1

Vorzeichen: $0 + 0 = 0$

Normierung:
Komma in Mantisse eine Stelle nach links verschieben (durch 2 teilen), Charakteristik um 1 inkrementieren (mit 2 mutiplizieren).
$\Rightarrow c_{Ergebnis} = (11010)_2; m'_{Ergebnis} = (001)_2$

Aufgabe 8 **12 Punkte**

2016-N-08 **Assembler und Cache**

/ 12

Die Befehle einer Assembler-Sprache seien folgendermaßen aufgebaut, wobei `Q` für Quelle steht und `Z` für Ziel:

```
OpCode Q1, (Q2,) Z
```

unmittelbare Adressierung wird durch das Präfix # gekennzeichnet, **indirekte Adressierung** durch *. Der Sprungbefehl `JNZ Q label` springt genau dann zu `label`, wenn der Inhalt von `Q` $\neq 0$ ist.

Das Rechenwerk enthält die Register `R1` und `R2`, die Adressen der Hauptspeicherzellen sind natürliche Zahlen k ($k \in \mathbb{N}_0$). Zwischen Registern und Hauptspeicher befindet sich ein **2-zeiliger Direct-Mapped-Cache** mit **Write-Back-Policy** zur Zwischenspeicherung der geladenen Operanden.

Gegeben sei das folgende Assemblerprogramm.

```
I     |       LOAD   #1  R1
II    |       LOAD   20  21
III   | loop1 LOAD   #1  R2
IV    |       LOAD   21  24
V     | loop2 MULT   24  R2    R2
VI    |       SUB    24  #1    24
VII   |       JNZ    24  loop2
VIII  |       MULT   R2  R1    R1
IX    |       STORE  R1  *20
X     |       SUB    21  #1    21
XI    |       JNZ    21  loop1
XII   | end
```

(a) Dokumentieren Sie in der gegebenen Tabelle den Programmablauf, bis zum ersten Mal der Befehl `XI` erreicht wird. Kennzeichnen Sie für jeden ausgeführten Befehl, ob es dabei zu einem Cache-Hit ($H\,\langle addr\rangle$) oder Cache-Miss ($M\,\langle addr\rangle$) kommt. Sie finden sowohl die Tabelle als auch eine Kopie des Programms auf der nächsten Seite.

Hinweis: Die ersten beiden Schritte (Befehlsausführungen) sind bereits beispielhaft eingetragen. Den Inhalt der Register R1 und R2, des Cache und der verwendeten Hauptspeicherzellen können sie als Hilfe notieren, sie werden jedoch bei der Punktevergabe nicht berücksichtigt.

/ 10

```
I       |       LOAD   #1   R1
II      |       LOAD   20   21
III     | loop1 LOAD   #1   R2
IV      |       LOAD   21   24
V       | loop2 MULT   24   R2     R2
VI      |       SUB    24   #1     24
VII     |       JNZ    24   loop2
VIII    |       MULT   R2   R1     R1
IX      |       STORE  R1   *20
X       |       SUB    21   #1     21
XI      |       JNZ    21   loop1
XII     | end
```

Lösung:

Programmablauf

Schritt	Befehl	Hit / Miss
1	I	-
2	II	M 20
3	III	-
4	IV	M 21
5	V	M 24
6	VI	H 24
7	VII	H 24
8	V	H 24
9	VI	H 24
10	VII	H 24
11	VIII	-
12	IX	M 20
13	X	H 21
14	XI	H 21

Rechenwerk

R1	1, 2
R2	1, 2, 2

Cache (direct mapped, write back)

Zeile	Tag-Feld (Hauptspeicheradresse)	Datum
0	20, 24, 20	2, 2, 1, 0, 2
1	21	2, 1

Hauptspeicher

Adresse	Datum
20	2
21	2
24	2, 0
2	2

(b) In welchem Schritt, bzw. in welchen Schritten wird der Wert in Speicherzelle 24 verändert?

Lösung: / 2

In Schritt 4, wenn Befehl `IV` den Inhalt von Speicherzelle 21 nach 24 schreibt und in Schritt 12, wenn nach einem Cache-Miss auf Adresse 20 diese die 24 aus dem Cache verdrängt und der Wert im Hauptspeicher aktualisiert wird (write back). Die Modifikationen in den Schritten 6 und 9 durch Befehl `VI` finden nur im Cache statt!

(c) **Bonusaufgabe für drei Zusatzpunkte:** Welche Funktion berechnet das Programm für allgemeine Werte $n \in \mathbb{N}$ an Hauptspeicheradresse 20?

Lösung:

$\prod_{i=1}^{n} i!$

Aufgabe 9 **8 Punkte**

2016-N-09 **Betriebssysteme**

/ 8

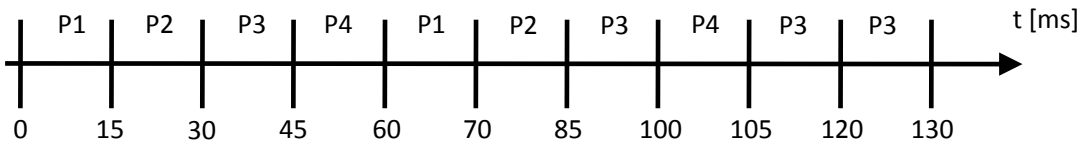
In der Warteschlange eines Prozessors befinden sich die folgenden Prozesse P1 bis P4, die in dieser Reihenfolge beim Prozessor ankommen:

Prozesse	CPU-Zeit in ms	Priorität
P1	25	1
P2	30	2
P3	55	2
P4	20	1

(a) Teilen Sie den Prozessen Rechenzeit gemäß dem Round Robin Verfahren zu. Die Zeitscheibe sei dabei in feste Zeitspannen der Länge $Z = 15$ms unterteilt. Veranschaulichen Sie Ihr Ergebnis auf dem gegebenen Zeitstrahl.

/ 2

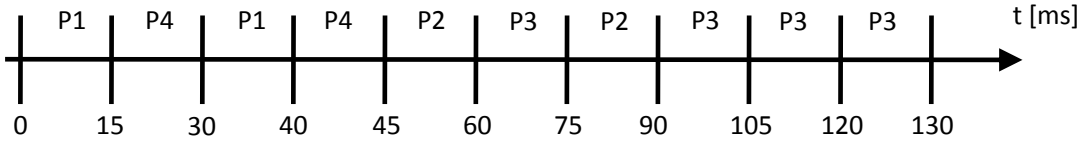
Lösung:



(b) Verwenden Sie nun ein prioritätsbasiertes Round Robin Verfahren, in dem zunächst immer nur den Prozessen der aktuell höchsten Prioritätsstufe Rechenzeit zugewiesen wird. Erst wenn alle Prozesse der höchsten Prioritätsstufe abgearbeitet wurden, wird Prozessen der nächsten Prioritätsstufe Rechenzeit zugeteilt. Die Zeitspannen der Zeitscheibe betragen weiterhin $Z = 15$ms. Veranschaulichen Sie Ihr Ergebnis auf dem gegebenen Zeitstrahl.

/ 2

Lösung:



(c) Erklären Sie den Unterschied zwischen **preemptive** und **nonpreemptive** Scheduling. Zu welchem Typ zählt das Round-Robin Verfahren?

/ 2

Lösung:

- Im preemptive Scheduling dürfen Prozesse vor Fertigstellung unterbrochen werden. Im nonpreemptive Scheduling muss ein einmal gestarteter Prozess komplett abgearbeitet werden, bevor der nächste Prozess gestartet werden kann.
- Das Round-Robin-Verfahren gehört zu den preemptive Scheduling Verfahren.

(d) Erklären Sie, was ein Deadlock ist und geben Sie eine Möglichkeit an, wie Deadlocks verhindert werden können.

/ 2

Lösung:

- Ein Deadlock bezeichnet die gegenseitige Blockierung voneinander abhängiger Prozesse.
- Ein Prozess kann die von ihm benötigten Betriebsmittel vor seiner Ausführung sperren. Falls er nicht alle Betriebsmittel sperren kann (weil diese unter Umständen bereits von einem anderen Prozess gesperrt sind), dann gibt er seine bisher gesperrten Betriebsmittel frei und versucht die notwendigen Betriebsmittel zu einem späteren Zeitpunkt noch einmal zu sperren. *Hinweis: Hier sind auch andere Lösungen denkbar.*

Aufgabe 10 **8 Punkte**

2016-N-10 **Hashing**

/ 8

(a) Verwenden Sie die Divisionsmethode mit dem Divisor 11 um die folgenden Zahlen in die untenstehende Hashtabelle einzutragen. Behandeln Sie auftretende Kollisionen durch lineares Austesten.

$$23, 505, 66, 75, 64, 13, 21, 95$$

/ 4

Lösung:

Schlüssel	23	505	66	75	64	13	21	95
Hash	1	10	0	9	9	2	10	7

RSN	0	1	2	3	4	5	6	7	8	9	10
Schlüssel	66	23	64	13	21			95		75	505

(b) Nennen und beschreiben Sie kurz eine weitere Hashfunktion, die zur Dateiorganisation geeignet ist.

/ 2

Lösung:

In der Vorlesung wurden zwei weitere Methoden vorgestellt:

- **Multiplikative Methode**: Der zu speichernde Schlüssel wird mit einer Zahl $r \in (0, 1)$ multipliziert. Die Nachkommastellen der resultierenden Zahl geben den Index in der Tabelle an.
- **Mittelquadrat-Methode**: Der zu speichernde Schlüssel wird quadriert. Der zu speichernde Index ergibt sich aus dem mittleren Block der quadrierten Zahl.

(c) Geben Sie eine weitere Möglichkeit zur Kollisionsbehandlung neben dem linearen Austesten an.

/ 2

Lösung:

Hier gibt es grundsätzlich viele Möglichkeiten. Die aufgelisteten Verfahren sind nicht erschöpfend.

- Neu einzufügende Elemente, die bereits mit einem existierenden Tabelleneintrag kollidieren, können in einem separaten Überlaufbereich gespeichert werden.
- Im Falle einer Kollision kann eine zweite Hashfunktion zusätzlich zur ersten verwendet werden, um den Tabelleneintrag zu berechnen.