Aufgabe 1. Minimierung und reguläre Ausdrücke (6 + 1 + 4 = 11 Punkte)

/ 11

Bei der Minimierung eines endlichen Automaten habe sich folgende Minimierungstabelle ergeben.

s_1	X1				
s_2	X0	X0			
s_3	X2	X1	X0		
s_4	X2	X1	X0	X2	
s_5	X1	X1	X0	X1	X1
	s_0	s_1	s_2	s_3	s_4

(a) Geben Sie für jedes $k \in \{0, 1, 2\}$ jeweils die Aufteilung der Zustandsmenge auf die Mengen der Zustände des Automaten an, die zueinander k-äquivalent sind, sowie die Mengen der äquivalenten Zustände. Geben Sie auch einelementige Mengen an.

Lösung:

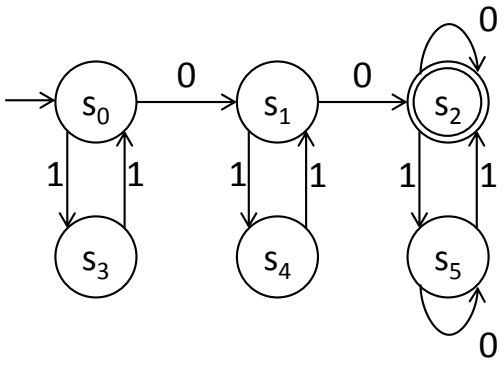
- Mengen 0-äquivalenter Zustände: $\{s_0, s_1, s_3, s_4, s_5\}, \{s_2\}$
- Mengen 1-äquivalenter Zustände: $\{s_0, s_3, s_4\}, \{s_1\}, \{s_2\}, \{s_5\}$
- Mengen 2-äquivalenter Zustände: $\{s_0\}, \{s_1\}, \{s_2\}, \{s_3\}, \{s_4\}, \{s_5\}$
- Mengen äquivalenter Zustände: $\{s_0\}, \{s_1\}, \{s_2\}, \{s_3\}, \{s_4\}, \{s_5\}$

(b) Der minimierte Automat A sei gegeben durch $A = (\{0, 1\}, \{s_0, s_1, s_2, s_3, s_4, s_5\}, \delta, s_0, \{s_2\})$.

Geben Sie eine alternative Menge der Endzustände $\overline{F}$ an, die sich aus obiger Minimierungstabelle ergeben könnte.

Lösung: $\overline{F} = \{s_0, s_1, s_3, s_4, s_5\}$

(c) Gegeben sei der endliche Automat $A' = (\{0, 1\}, \{s_0, s_1, s_2, s_3, s_4, s_5\}, \delta', s_0, \{s_2\})$ mit δ':



Beschreiben Sie die Sprache $L(A')$ durch einen regulären Ausdruck α.
Lösung:
$\alpha = (11)^\star 0(11)^\star 0(0 + 10^\star 1)^\star$

Aufgabe 2. Kellerautomat (9 Punkte)

/ 9

Die Sprache $L \subseteq \{\ (,),[,]\ \}^\star$ sei gegeben durch folgende Definition:

- $\lambda \in L$,
- falls $w, w' \in L$, dann auch: ww', (w) und $[w] \in L$.

L ist also die Sprache aller wohlgeformten Klammerausdrücke über (,), [und].

Es gilt beispielsweise:
$\lambda,\ ((())),\ ()[](([][])),\ [](),\ ([]()) \in L$
$)(,\ ()[),\ ()([)],\ ((,\ ][,\ (]](][\notin L$

Geben Sie einen deterministischen Kellerautomaten $KA = (E, S, K, \delta, s_0, k_0, F)$ an mit $L(KA) = L$. Geben Sie diesen vollständig an.

Zeigen Sie, dass Ihr Kellerautomat das Testwort ()[()] akzeptiert.

Lösung:
$KA = (\{\ (,),[,]\ \}, \{s_0, s_1\}, \{k_0, (, [\ \}, \delta, s_0, k_0, \{s_0\})$

$$
\begin{aligned}
&\delta: \\
&\quad \delta(s_0, (, k_0) = (s_1, (k_0) \\
&\quad \delta(s_0, [, k_0) = (s_1, [k_0) \\
&\quad \delta(s_1,), (\ ) = (s_1, \lambda) \\
&\quad \delta(s_1,], [\ ) = (s_1, \lambda) \\
&\quad \delta(s_1, (, [\ ) = (s_1, ([\ ) \\
&\quad \delta(s_1, [, (\ ) = (s_1, [(\ ) \\
&\quad \delta(s_1, (, (\ ) = (s_1, ((\ ) \\
&\quad \delta(s_1, [, [\ ) = (s_1, [[\ ) \\
&\quad \delta(s_1, \lambda, k_0) = (s_0, k_0)
\end{aligned}
$$

Erkennung des Testwortes ()[()]:

$$
\begin{aligned}
&(s_0, ()[()], k_0) \vdash (s_1,)[()], (k_0) \vdash (s_1, [()], k_0) \vdash (s_0, [()], k_0) \vdash \\
&(s_1, ()], [k_0) \vdash (s_1,)], ([k_0) \vdash (s_1,], [k_0) \vdash (s_1, \lambda, k_0) \vdash (s_0, \lambda, k_0)
\end{aligned}
$$

Aufgabe 3. Grammatiken und Chomsky-Hierarchie (5 + 2 + 1 + 2 = 10 Punkte)

/ 10

Gegeben sei die Grammatik G mit

$$\begin{aligned} G &= (\{S, B, Y, T\}, \{a, b\}, P, S) \\ P &= \{S \to YTb \mid a, \\ &\quad Y \to BY \mid aa, \\ &\quad Ba \to aaB, \\ &\quad BT \to Tb, \\ &\quad aTb \to ab\} \end{aligned}$$

(a) Geben Sie vier verschiedene Wörter $w_1, w_2, w_3, w_4 \in L(G)$ an.

Lösung: $w_1 = a, w_2 = aab, w_3 = aaaabb, w_4 = aaaaaaaabbb$.

(b) Geben Sie die Sprache $L(G)$ (mathematisch oder umgangssprachlich präzise) an.

Lösung:

$$L(G) = \left\{a^{2^n} b^n \mid n \in \mathbb{N}_0\right\}$$

(c) Geben Sie das größte i an, sodass gilt: Die Grammatik G ist vom Chomsky-Typ i.

Lösung: $i = 0$. Denn es gibt nicht-kontextfreie, verkürzende Regeln.

(d) Angenommen, jemand beauftragt Sie, die Grammatik G in Chomsky-Normalform zu bringen. Wie würden Sie vorgehen, um zu beweisen, dass das nicht möglich ist?

Begründen Sie kurz.

Lösung: CNF ist kontextfrei, was die obige Grammatik offenbar nicht ist. Um das zu zeigen, könnte man das Pumping-Lemma für kontextfreie Sprachen verwenden.

Aufgabe 4. Cocke-Younger-Kasami-Algorithmus (7 Punkte)

/ 7

Gegeben sei die Grammatik $G = (N, T, P, S)$ mit $N = \{S, A, B, C\}, T = \{a, b, c\}$ und

$$\begin{aligned} P = \{S &\rightarrow AB \mid a, \\ A &\rightarrow BC \mid a, \\ B &\rightarrow CA \mid AS \mid b, \\ C &\rightarrow SC \mid AB \mid c\}. \end{aligned}$$

Überprüfen Sie mithilfe des Algorithmus von Cocke, Younger und Kasami, ob

$$w = bacca \in L(G).$$

Geben Sie zusätzlich zum Ausfüllen der Tabelle explizit an, ob $w \in L(G)$.

Lösung:

	b	a	c	c	a
$m = 1$	B	S, A	C	C	S, A
$m = 2$	$\emptyset$	C	$\emptyset$	B	
$m = 3$	A	$\emptyset$	$\emptyset$		
$m = 4$	$\emptyset$	$\emptyset$			
$m = 5$	S, C				

Da S im letzten Feld enthalten ist, liefert der Algorithmus das Ergebnis, dass das angegebene Wort durch die angegebene Grammatik erzeugt werden kann.
$\Rightarrow bacca \in L(G)$

Aufgabe 5. Komplexitäts- und Berechenbarkeitstheorie (9 Punkte)

/ 9

Gegeben seien die folgenden Informationen zu den Problemen A, B, C, D, E und F:

- $CLIQUE \leq_{pol} A \leq_{pol} B \leq_{pol} F$,
- $B \in NP$,
- $C \leq_{pol} PRIMES \leq_{pol} D$,
- E ist äquivalent zum Halteproblem.

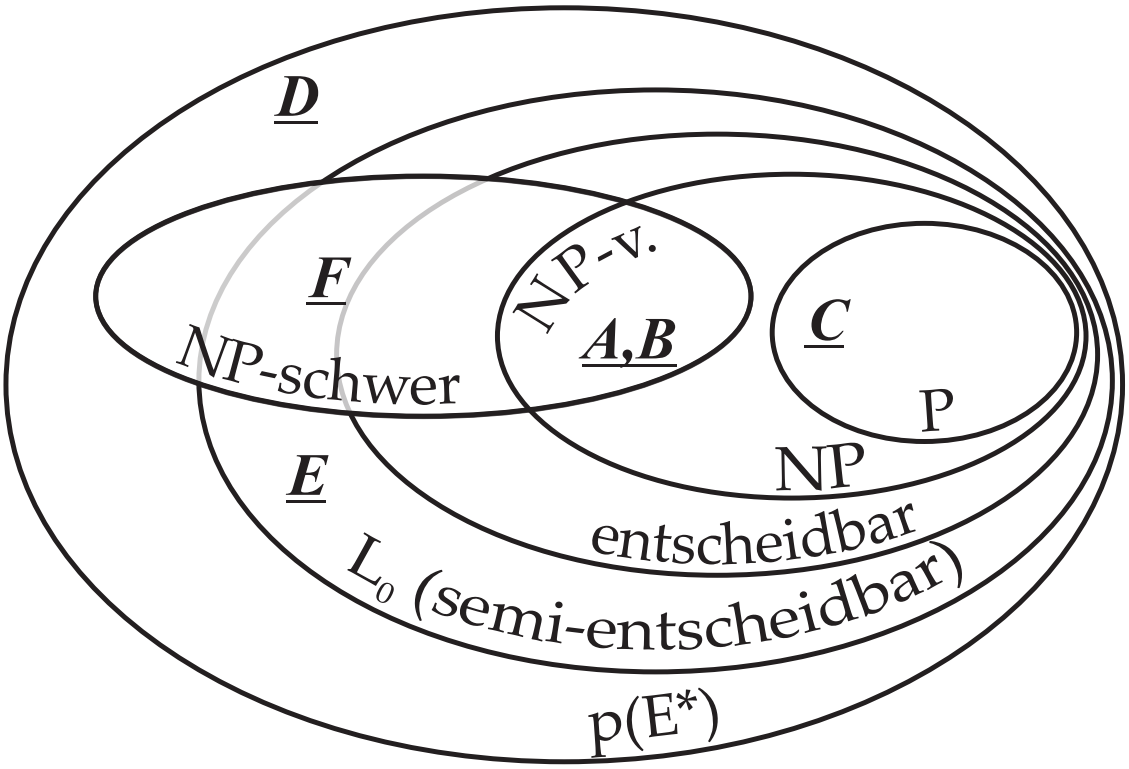
Darüber hinaus sei bekannt, dass

- $CLIQUE \in NP-$vollständig,
- $PRIMES \in P$,

Zeichnen Sie A, B, C, D, E und F so genau wie möglich in das abgebildete Diagramm ein.

Hinweis: Sie erhalten 1,5 Punkte pro Problem, das an „genau der richtigen Stelle“ eingeordnet ist, also so tief wie möglich, aber nicht tiefer als aus den obigen Informationen ableitbar. Für NP-schwer müssen Sie nur „$NP-$vollständig“ und „nicht $NP-$vollständig“ unterscheiden.

Lösung:



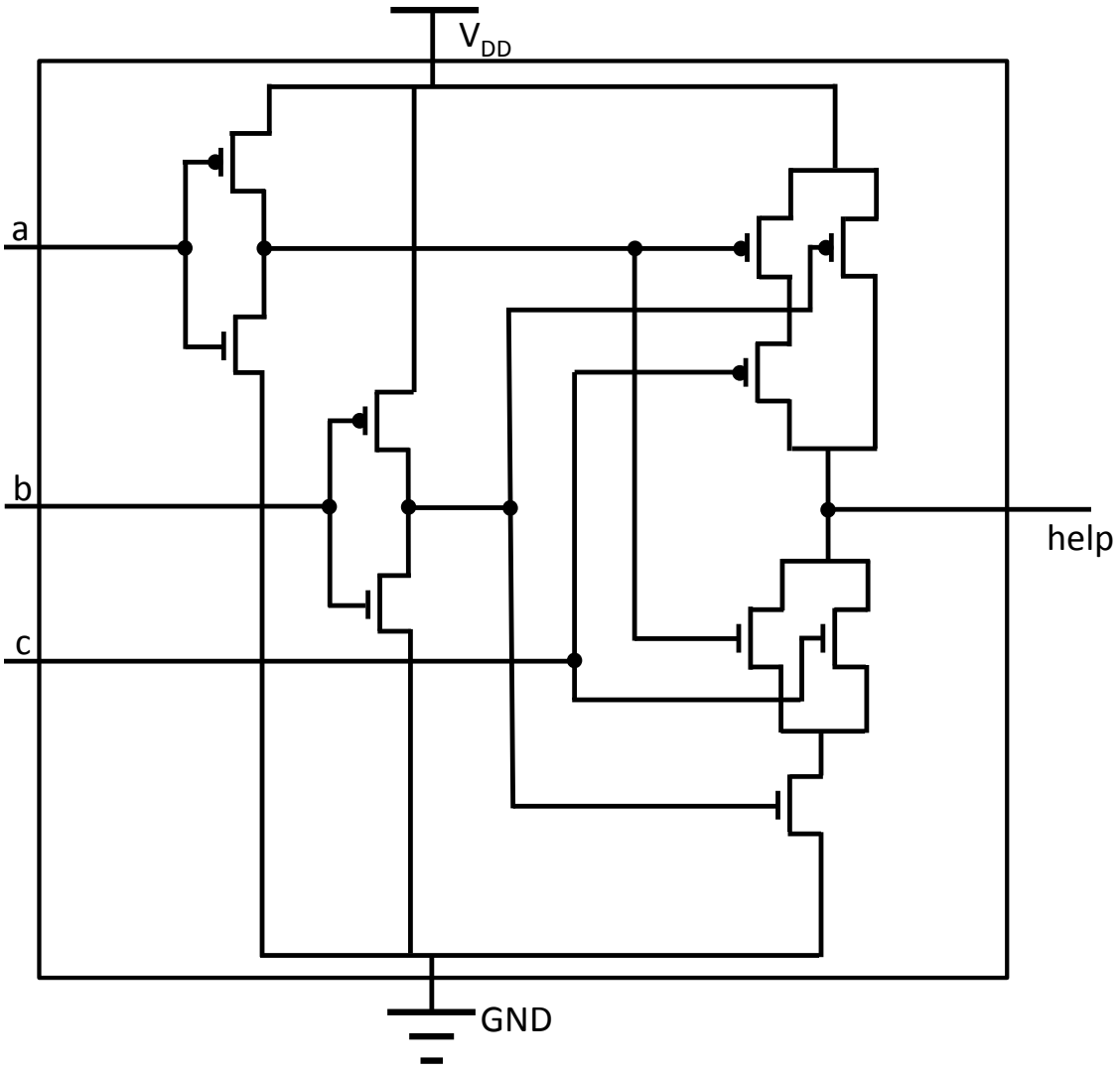
Aufgabe 6. CMOS (9 Punkte)

/ 9

Gegeben sei folgende Schaltfunktion „help“:

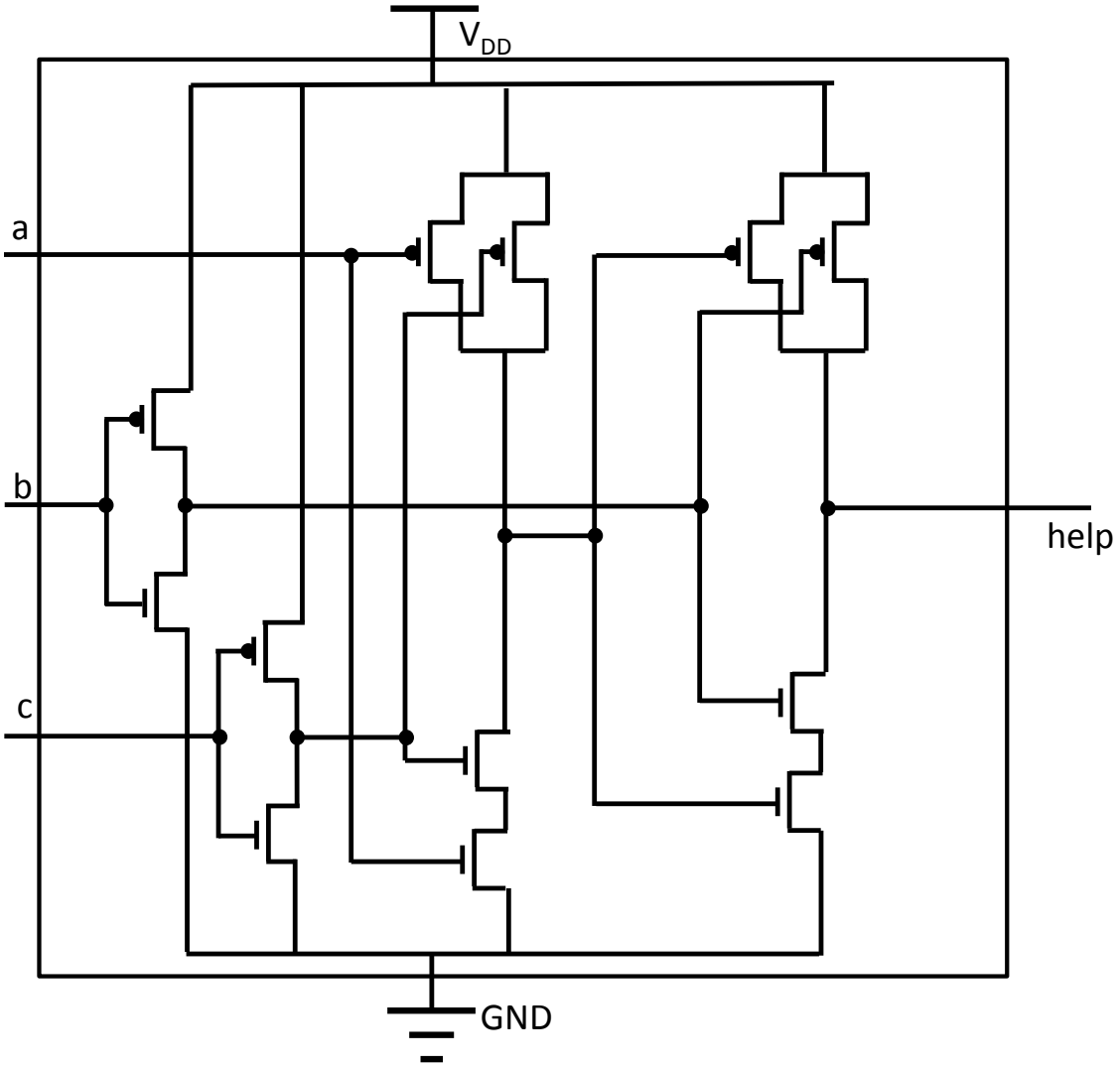
$$\text{help}(a, b, c) = b \vee (a \wedge \neg c).$$

Geben Sie eine CMOS-Schaltung für diese Funktion an. Zeichnen Sie diese auf Transistorebene (verwenden Sie also keine elementaren CMOS-Bausteine als Blackboxen).
Lösung:



zweiter, etwas komplizierterer Lösungsweg (über die Umwandlung in Gatter):

$b \vee (a \wedge \neg c) = \neg(\neg b \wedge \neg(a \wedge \neg c)) = \neg b \,\text{NAND}\, \neg(a \wedge \neg c) = \neg b \,\text{NAND}(a \,\text{NAND}\, \neg c)$



Aufgabe 7. Zahlendarstellung (1 + 1 + 1 + 1 + 1 + 2 = 7 Punkte)

/ 7

Gegeben sei ein 4-stelliger Binärstring $w \in \{0,1\}^4$.

Geben Sie jeweils die kleinste und die größte darstellbare Dezimalzahl an, wenn w als folgende Zahlendarstellung interpretiert wird:

(a) Vorzeichen-Betrag-Darstellung

Lösung:
$w_{VB} \in \{-7, \ldots, 7\}$

(b) 1-Komplement-Darstellung

Lösung:
$w_{1K} \in \{-7, \ldots, 7\}$

(c) 2-Komplement-Darstellung

Lösung:
$w_{2K} \in \{-8, \ldots, 7\}$

(d) BCD-Darstellung

Lösung:
$w_{BCD} \in \{0, \ldots, 9\}$

(e) Exzess-3-Darstellung

Lösung:
$w_{E-3} \in \{-3, \ldots, 12\}$

(f) Gleitpunktdarstellung mit 1-Bit-Charakteristik und 2-Bit-Mantisse

Hinweis: Gehen Sie davon aus, dass es – im Gegensatz zur IEEE-754-Darstellung – keine Sonderwerte oder denormalisierte Zahlen gibt.
Lösung:
$w_{GPZ} \subset [-3,5; 3,5]$
da $\pm 2^1 \cdot (1 + 2^{-1} + 2^{-2})$ mit $q = 0 = 2^{n-1} - 1$

Aufgabe 8. Huffman-Kodierung (9 Punkte)

/ 9

Folgende Zeichenkette sei repräsentativ für Daten, die noch kommen sollen:

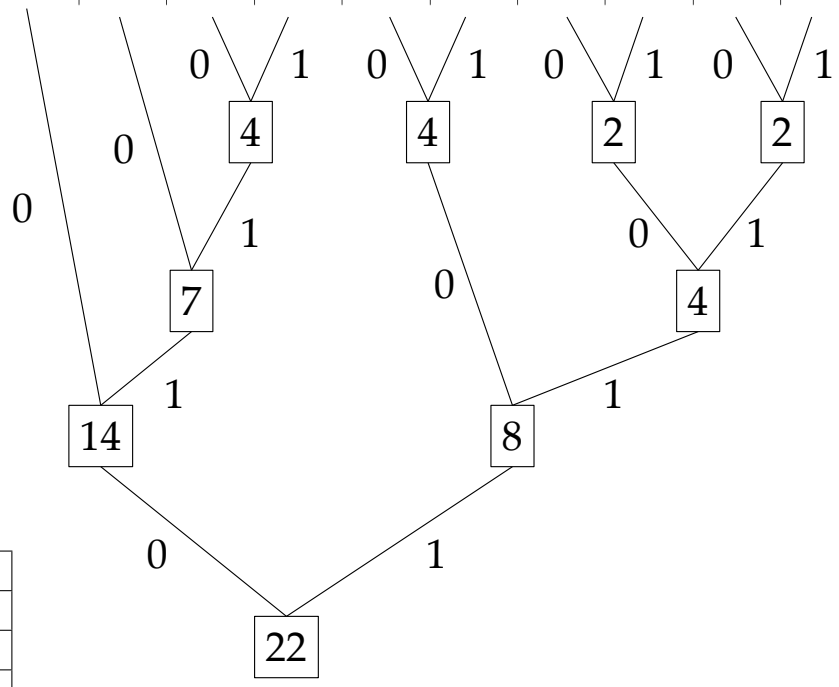
ABRAKADABRA-SIMSALABIM

Erzeugen Sie anhand der durch die Zeichenkette gegebenen Häufigkeitsverteilung eine Huffman-Kodierung.

Tragen Sie dazu die Häufigkeiten der Zeichen in die erste Tabelle ein, erstellen Sie einen entsprechenden Baum mit Angabe der Häufigkeiten an den Knoten und tragen Sie die Kodierung der Zeichen in die zweite Tabelle ein.

Lösung:

Zeichen	A	B	R	S	I	M	K	D	-	L
Häufigkeit	7	3	2	2	2	2	1	1	1	1



Zeichen	Code
A	00
B	010
R	0110
S	0111
I	100
M	101
K	1100
D	1101
-	1110
L	1111

Aufgabe 9. Assembler und Adressierung (8 Punkte)

/ 8

Die Befehle einer Assembler-Sprache seien folgendermaßen aufgebaut:

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

Dabei werde das Ergebnis der Operation *OpCode* basierend auf den durch $Q1$ (und bei Rechenoperationen $Q2$) bezeichneten Operanden an der durch Z bezeichneten Adresse abgelegt.

Gegeben sei folgender Ausschnitt aus einem Assemblerprogramm.

```
STORE 1001, 1002
STORE #1002, 1001
STORE *1001, 1003
MULTIPLY 1003, 1002, *1003
SUBTRACT 50, #2499, 1004
```

Dabei werden für die Adressierungsarten folgende Notationen verwendet:

- Unmittelbare Adressierung: Präfix '#'
- Direkte Adressierung: ohne Präfix
- Indirekte Adressierung: Präfix '$*$'

Gegeben sei weiter ein 4-zeiliger Assoziativ-Cache, der nach dem *Least-Recently-Used*-Prinzip arbeite und zu Beginn die eingetragenen Werte enthalte:

Cache-Zeile	Tag-Feld (Hauptspeicheradresse)	Datum	Zugriffs-Zeitpunkte
0	1001, 1001, 1004	50, 1002, 1	1, 3, 4, 12
1	1002	50	2, 5, 8
2	1003	50	6, 7, 9
3	50	2500	10, 11

Bei jeder Assembler-Operation werden die beteiligten Operanden im Cache abgelegt.

Aktualisieren Sie die obige Tabelle mit den während des Programmlaufs anfallenden Werten. Schreiben Sie dabei die Zugriffszeitpunkte in die letzte Spalte.

Aufgabe 10. Speicherorganisation (6 + 1 = 7 Punkte)

/ 7

(a) Geben Sie für folgende Speicher die Energieabhängigkeit sowie die Zugriffsart an. Markieren Sie dies durch ein Kreuz in der entsprechenden Zelle der folgenden Tabelle.

Lösung:

	Energieabhängigkeit		Zugriffsarten		
	flüchtig	nicht flüchtig	wahlfrei	sequentiell	block-adressierbar
Register	X		X		
Optischer Speicher		X			X
Hauptspeicher	X		X		
Cache	X		X		
Magnetplattenspeicher		X			X
Solid-State-Disk		X			X

(b) Nennen Sie je einen Vor- und einen Nachteil für die Verwendung von Solid-State-Disks gegenüber herkömmlichen Magnetplattenspeichern.

Lösung:
Vorteil:

- deutlich schneller
- geräuschlos
- weniger Energieverbrauch

Nachteil

- begrenzte Anzahl an Schreibvorgängen, Abnutzung

Aufgabe 11. Betriebssysteme (1 + 1 + 1 + 1 = 4 Punkte)

/ 4

Beantworten Sie die folgenden Fragen kurz und stichwortartig.

(a) Nennen Sie zwei Betriebsarten, die eine Rechenanlage unterstützen kann.

Lösung: Stapelbetrieb, Multiprogrammbetrieb, Dialogbetrieb, Echtzeitbetrieb, Client-/Server-Betrieb.

(b) Nennen Sie zwei mögliche Zustände eines Prozesses.

Lösung: initiiert, bereit, aktiv, blockiert, terminiert.

(c) Wie nennt man Prozesse, die sich einen Adressraum teilen?

Lösung: Threads.

(d) Was ist „Round Robin“?

Lösung: Ein Zuteilungsverfahren für Prozessorzeit an Prozesse.