Lösung

Aufgabe 1. Rechtslineare Sprachen $(2 + 3 + 3 = 8$ Punkte)

Gegeben sei die Grammatik $G = (N, T, P, S), N = \{S, A, B, C, D\}, T = \{0, 1\}$,

$$\begin{aligned} P = \{S &\rightarrow 0S|1A, \\ A &\rightarrow 0S|1B, \\ B &\rightarrow 0S|1C, \\ C &\rightarrow 0S|1D, \\ D &\rightarrow 0S|1D|\lambda\}. \end{aligned}$$

(a) Von welchem / welchen Chomsky-Typ(en) ist diese Grammatik?

Lösung: 0, 2, 3. Nicht kontextsensitiv, wegen Lambda-Regel. 0.5 Punkte / richtige Typzuordnung.

(b) Geben Sie einen regulären Ausdruck RA an, sodass gilt: $L(RA) = L(G)$.

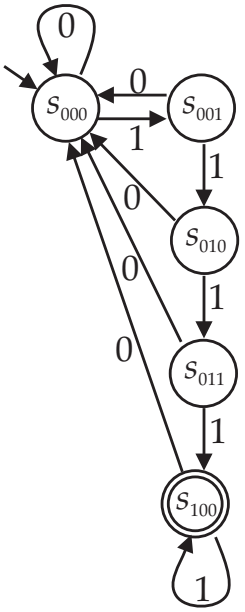
Lösung (Bsp.): $RA = (0 + 1)^\star 1111$

Pro Wortklasse, die falsch in $L(RA)$ bzw. nicht in $L(RA)$ ist: -1 Punkt.

Textuelle Beschreibung der Sprache: bis zu 1,5 Punkten (aber nicht zusätzlich zum RA).

(c) Geben Sie einen endlichen Automaten A an, sodass gilt: $L(A) = L(G)$. Definieren Sie den Automaten vollständig.

Lösung (Bsp.):



Aufgabe 2. Kellerautomaten (11 Punkte)

Gegeben sei für $E = \{a, b, c, d\}, \mathbb{N}^+ = \{1, 2, \ldots\}$ die Sprache

$$L = \{w \in E^\star | w = a^m b^n c^o d^p, m + n = o - p \text{ und } m, n, o, p \in \mathbb{N}^+\}.$$

Konstruieren Sie einen deterministischen Kellerautomaten KA mit $L(KA) = L$. Definieren Sie den Automaten vollständig.

Hinweis: Die Lösung erfordert bspw. eine Erweiterung des Kellerautomaten für die Sprache $\{a^n b^n \mid n \in \mathbb{N}\}$.

Lösung.

Der Automat liest zu Beginn $a^m b^n$ ein und wechselt dann den Zustand. Mit c^o wird der Keller gelöscht bzw. wird der Rest wieder in den Keller geschrieben. Mit den verbleibenden Eingabezeichen d^p kann der Keller restlos gelöscht werden. Somit ergibt sich der Kellerautomat $KA = (E, S, K, \delta, s_0, k_0, F)$ mit $E = \{a, b, c, d\}$, $S = \{s_0, s_1, s_2, s_3, s_4\}$, $K = \{k_0, a, b, c\}$, $F = \{s_4\}$ und

$$\begin{array}{lll}
\delta: & (s_0, a, k_0) & \to (s_0, ak_0) \\
& (s_0, a, a) & \to (s_0, aa) \\
& (s_0, b, a) & \to (s_0, ba) \\
& (s_0, b, b) & \to (s_0, bb) \\
& (s_0, c, b) & \to (s_1, \lambda) \quad \text{aus dem Keller löschen} \\
& (s_1, c, b) & \to (s_1, \lambda) \quad \text{aus dem Keller löschen} \\
& (s_1, c, a) & \to (s_1, \lambda) \quad \text{aus dem Keller löschen} \\
& (s_1, c, k_0) & \to (s_2, ck_0) \\
& (s_2, c, c) & \to (s_2, cc) \\
& (s_2, d, c) & \to (s_3, \lambda) \quad \text{aus dem Keller löschen} \\
& (s_3, d, c) & \to (s_3, \lambda) \quad \text{aus dem Keller löschen} \\
& (s_3, \lambda, k_0) & \to (s_4, k_0)
\end{array}$$

oder

$$
\begin{array}{lll}
\delta': & (s_0, a, k_0) & \rightarrow (s_0, ak_0) \\
& (s_0, b, k_0) & \rightarrow (s_0, bk_0) \quad \text{Sonderfall: } m = 0 \\
& (s_0, c, k_0) & \rightarrow (s_2, ck_0) \quad \text{Sonderfall: } m, n = 0 \\
& (s_0, \delta, k_0) & \rightarrow (s_4, k_0) \quad \text{Sonderfall: } m, n, o, p = 0 \\
& (s_0, a, a) & \rightarrow (s_0, aa) \\
& (s_0, b, a) & \rightarrow (s_0, ba) \\
& (s_0, b, b) & \rightarrow (s_0, bb) \\
& (s_0, c, a) & \rightarrow (s_1, \lambda) \quad \text{Sonderfall: } n = 0 \\
& (s_0, c, b) & \rightarrow (s_1, \lambda) \quad \text{aus dem Keller löschen} \\
& (s_1, c, b) & \rightarrow (s_1, \lambda) \quad \text{aus dem Keller löschen} \\
& (s_1, c, a) & \rightarrow (s_1, \lambda) \quad \text{aus dem Keller löschen} \\
& (s_1, c, k_0) & \rightarrow (s_2, ck_0) \\
& (s_1, \delta, k_0) & \rightarrow (s_4, k_0) \quad \text{Sonderfall: } p = 0 \\
& (s_2, c, c) & \rightarrow (s_2, cc) \\
& (s_2, d, c) & \rightarrow (s_3, \lambda) \quad \text{aus dem Keller löschen} \\
& (s_3, d, c) & \rightarrow (s_3, \lambda) \quad \text{aus dem Keller löschen} \\
& (s_3, \lambda, k_0) & \rightarrow (s_4, k_0)
\end{array}
$$

Bewertung.

- Die Aussage $m, n, o, p \in \mathbb{N}^+$ bedingt eigentlich, dass die Null nicht dazu gehört. Wer hier eine andere Annahme trifft, der soll allerdings nicht für seine Mehrarbeit bestraft werden.
- Wenn jemand beispielsweise in Worten (also keine formale Darstellung von δ) das Prinzip des Kellerautomaten darstellt, wie dieser arbeiten muss, dann werden dafür bis zu 5 Teilpunkte vergeben.
- Bis zu 2 Teilpunkte werden für eine fehlerfreie Darstellung von $KA = (E, S, K, \delta, s_0, k_0, F)$ mit $E = \{a, b, c, d\}$, $S = \{s_0, s_1, s_2, s_3, s_4\}$, $K = \{k_0, a, b, c\}$ und $F = \{s_4\}$ vergeben.
- Bis zu 9 Teilpunkte für eine fehlerfreie Darstellung von δ.
- Fehler führen zu halben Punktabzügen.

Aufgabe 3. Cocke-Younger-Kasami-Algorithmus (8 Punkte)

Gegeben sei die Grammatik $G = (N, T, P, S)$ mit $N = \{S, A, B, C, D\}$, $T = \{0, 1\}$ und

$$\begin{aligned} P = \{S &\rightarrow AA|AB, \\ A &\rightarrow 0|AA, \\ B &\rightarrow CD, \\ C &\rightarrow 1, \\ D &\rightarrow AC|BC\}. \end{aligned}$$

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

$$w = 10001 \in L(G).$$

Hinweis: Geben Sie zusätzlich zum Ausfüllen der Tabelle explizit an, ob $w \in L(G)$.
Lösung.
Grammatik erzeugt 0^i1^j.
Aus der Grammatik ergibt sich folgende Tabelle für das Verfahren von Cocke, Younger und Kasami.

(Bitte nur doppelt umrahmten Teil betrachten.)

	1	1	0	0	0	1	1	1	1
$m = 1$	C	C	A	A	A	C	C	C	C
$m = 2$	-	-	S, A	S, A	D	-	-	-	
$m = 3$	-	-	S, A	D	-	-	-		
$m = 4$	-	-	D	-	-	-			
$m = 5$	-	B	-	-	-				
$m = 6$	-	D	-	-					
$m = 7$	B	-	-						
$m = 8$	D	-							
$m = 9$	-								

Da S im letzten Feld ($m = 5$) nicht enthalten ist, liefert der Algorithmus das Ergebnis, dass das angegebene Wort nicht durch die angegebene Grammatik erzeugt werden kann.

6 Punkte für Algorithmus.
2 Punkte für Aussage $w \notin L(G)$.

Aufgabe 4. Komplexität und Berechenbarkeit $(4 + 1 + 4 = 9$ Punkte)

(a) Definieren Sie oder erklären Sie exakt in eigenen Worten, welche Probleme die folgenden vier aus der Vorlesung bekannten Komplexitätsklassen jeweils enthalten. Die Beschreibung muss alle enthaltenen Probleme umfassen, Beispiele genügen nicht.

P: **Lösung:** Probleme, die von einer deterministischen TM in Polynomialzeit berechnet werden können.

$$P = \bigcup_{k \in \mathbb{N}} DTime(n^k)$$

NP: **Lösung:** Probleme, die von einer nichtdeterministischen TM in Polynomialzeit berechnet werden können.

$$P = \bigcup_{k \in \mathbb{N}} NTime(n^k)$$

NP-schwer: **Lösung:** Probleme, auf die jedes Problem aus NP in Polynomialzeit reduziert werden kann. ($\frac{1}{2}$ Punkt Abzug, falls verkehrt herum.)

$$NP\text{-schwer} = \{X \mid \forall Y \in NP : Y \leq_{pol} X\}$$

NP-vollständig: **Lösung:** Probleme aus NP-schwer, die auch noch in NP liegen.

$$NP\text{-vollständig} = \{X \mid X \in NP \cap NP\text{-schwer}\}$$

Bewertung: Definition **oder** sprachliche Beschreibung genügen. Wer das bekannte Inklusions-Diagramm malt, bekommt einen Punkt für NP-vollständig, denn diese Klasse lässt sich allein durch Angabe der Inklusionen beschreiben. Wenn im Diagramm NP-schwer fehlt, gibt es noch einen halben Punkt. Wer das Reduktionsdiagramm malt, bekommt den Punkt für NP-schwer.

(b) Was ist eine universelle Turingmaschine? **Lösung:** TM U, die bei Eingabe einer TM $T = (E, S, B, \delta, s_0, F)$ und $w \in E^\star$ das Verhalten von T bei Eingabe von w simuliert.

(c) Wie kann man eine universelle Turingmaschine nutzen, um zu zeigen, dass das Halteproblem Turing-aufzählbar ist? (Es sei bereits bekannt, dass das Halteproblem nicht entscheidbar ist.)

Hinweis: Das Halteproblem entspricht der Sprache

$$H = \{(c_T, w) \mid c_T \text{ kodiert eine Turingmaschine } T \text{ und } T \text{ hält auf Eingabe } w\}.$$

Lösung: Sei die TM T und das Wort $w \in E^\star$ die Eingabe für das Halteproblem. Um zu zeigen, dass T auf Eingabe von w anhält, kann U eine Simulation von T auf w durchführen. Wenn die Simulation schließlich stoppt, kann U akzeptierend halten. Im anderen Fall läuft die Simulation endlos weiter und man kann nie mit Sicherheit sagen, ob sie anhalten wird.

(Ähnlich kann man bei einem Computerprogramm vorgehen, indem man es startet und abwartet, ob es anhält. Der Computer spielt dabei die Rolle der universellen TM.)

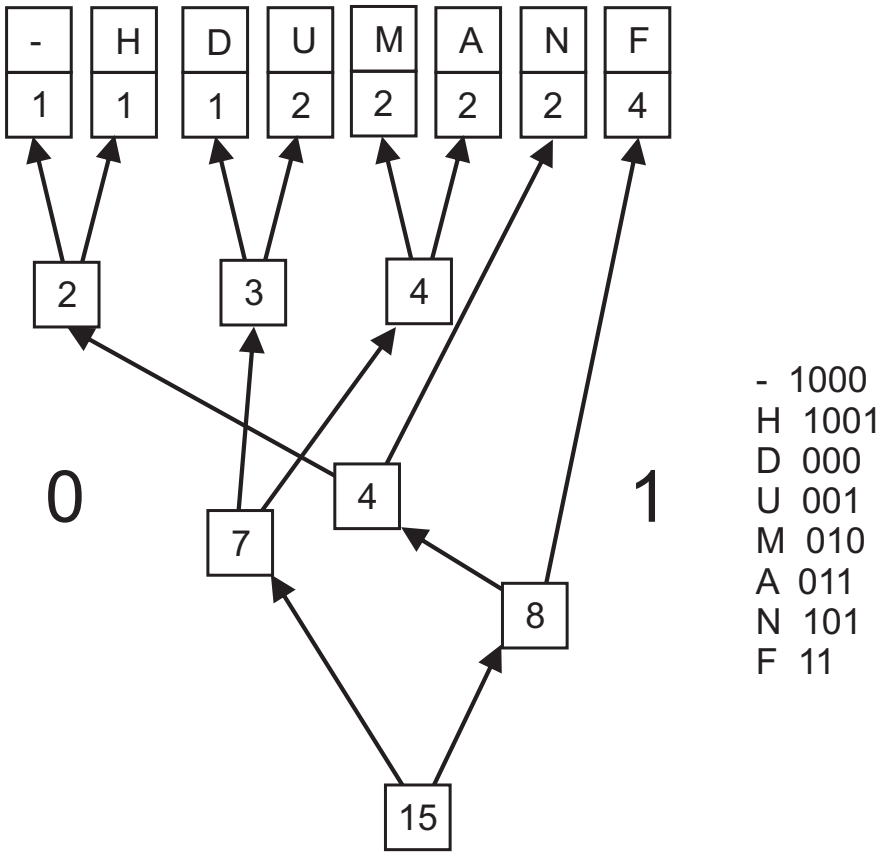
Aufgabe 5. Huffman-Kodierung (5 + 3 = 8 Punkte)

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

HUFFMAN-DUFFMAN

(a) 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.



L(c) = (1*4+1*4+1*3+2*3+2*3+2*3+2*3+4*2)/15
= 2 13/15

(b) Geben Sie die Codelänge der Huffman-Kodierung aus (a) an (Bruch genügt). **Lösung:** $2\frac{13}{15}$

Aufgabe 6. Gleitpunktzahlen (8 Punkte)

Geben Sie in der Tabelle die Werte der Zahlen an, welche im Gleitpunktzahlenformat „1.4.2“ dargestellt sind. Dabei stehen 1 Bit für das Vorzeichen, 4 Bit für die Charakteristik und 2 Bit für die Mantisse. Die Interpretation der Charakteristik geschehe analog zum IEEE 754 Standard (einschließlich denormalisierter Zahlen und Sonderwerte).

Hinweise:

- Für jeden richtig berechneten und zugeordneten Wert erhalten Sie einen Punkt.
- Es reicht, Produkte und Summen von Zweierpotenzen anzugeben, ohne sie auszurechnen.

Lösung und Bewertung.
Die Werte müssen nicht ausgerechnet werden. Es reicht, wenn das Endergebnis ableitbar ist. Ansonsten siehe Aufgabenstellung (insbesondere keine Folgefehler)! $q = 2^{4-1} - 1 = 7$.

GPZ-Darstellung	Wert
0.0000.00	denormalisiert: 0
0.0000.01	denormalisiert: $2^{1-7}(0 + 2^{-2}) = 2^{-8} = 0,00390625$
0.0001.00	$2^{1-7}(1 + 0) = 2^{-6} = 0,015625$
0.0001.01	$2^{1-7}(1 + 2^{-2}) = 2^{-6} + 2^{-8} = 0,01953125$
0.1000.00	$2^{8-7}(1 + 0) = 2$
0.1000.01	$2^{8-7}(1 + 2^{-2}) = 2^{1} + 2^{-1} = 2,5$
0.1111.00	denormalisiert: ∞
0.1111.11	denormalisiert: NaN

Es werden keine Teilpunkte für irgendwelche Annahmen vergeben. Ebenso gibt es keine Teilpunkte für richtige Vorzeichen, Mantissen oder Charakteristiken. Die Aufgabenstellung ist/war eindeutig!

Aufgabe 7. Schaltnetze (12 Punkte)

Gegeben seien die beiden 2-Bit-Binärzahlen

$$A = a_1 a_0 \text{ und } B = b_1 b_0.$$

Beide können beliebige Werte aus $\mathbb{B}^2$ annehmen. Konstruieren Sie ein Schaltnetz für die Berechnung der 4-Bit-Binärzahl $E = e_3 e_2 e_1 e_0$, die gegeben ist als Multiplikation

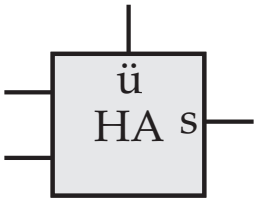
$$E = A \cdot B.$$

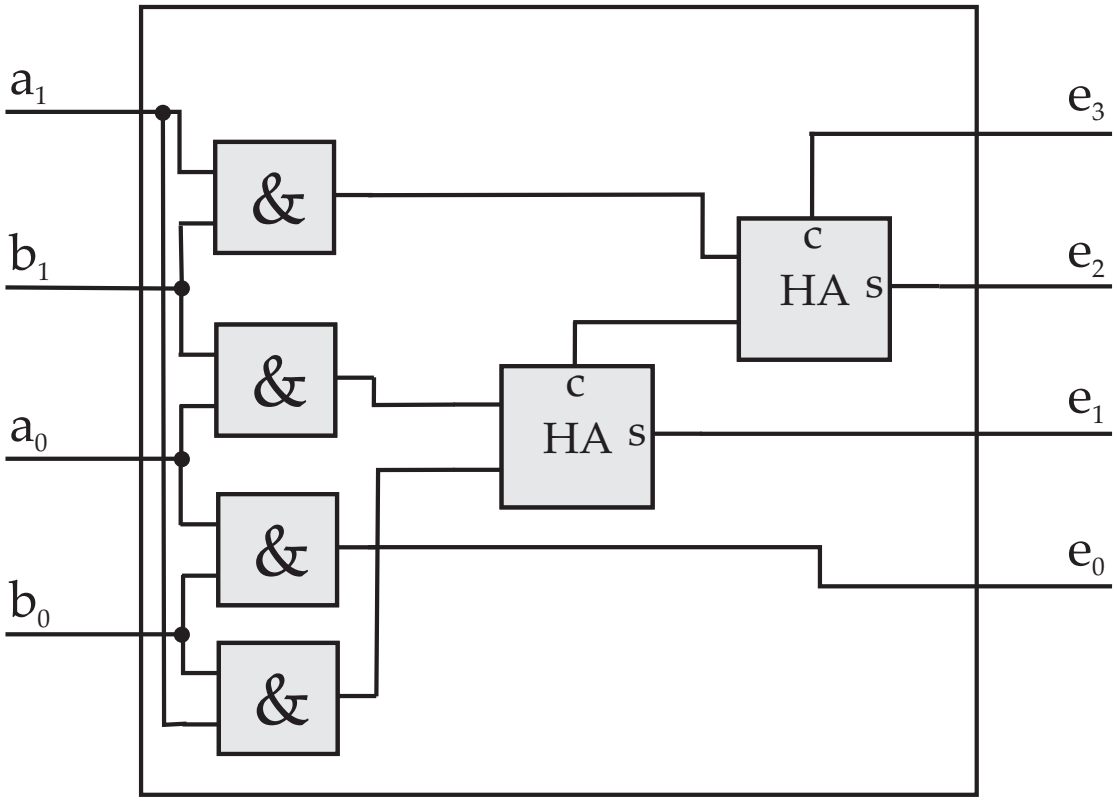
Zeichnen Sie Ihre Lösung in das vorgegebene Feld auf der nächsten Seite. Benutzen Sie für das Schaltnetz **nur Halbaddierer (HA) und &-Gatter** (s. u.) als Bausteine.

Hinweise:

- HA erhalten zwei Bits an den Eingängen und berechnen deren Summe und Übertrag.
- &-Gatter berechnen für zwei Bits $x, y \in \mathbb{B}$ die Und-Verknüpfung $x \wedge y$. Dies ist gleichzeitig die 1-Bit-Multiplikation $x \cdot y$.
- Orientieren Sie sich bei der Konstruktion an der umseitig angegebenen Umformung.
- Überprüfen Sie Ihre Lösung stichprobenweise anhand der umseitig angegebenen Wahrheitstafel.

Bausteine:





Umformung:

$E = A \cdot B = (a_1 \cdot 10 + a_0) \cdot (b_1 \cdot 10 + b_0)$
$= 100 \cdot a_1 \cdot b_1 + 10 \cdot a_0 \cdot b_1 + 10 \cdot a_1 \cdot b_0 + a_0 \cdot b_0$

$$
\begin{array}{rcccc}
 & & a_1 \cdot b_1 & 0 & 0 \\
 & & & a_0 \cdot b_1 & 0 \\
\Longrightarrow & & & a_1 \cdot b_0 & 0 \\
 & + & & & a_0 \cdot b_0 \\
\hline
 & e_3 & e_2 & e_1 & e_0 \\
\hline\hline
\end{array}
$$

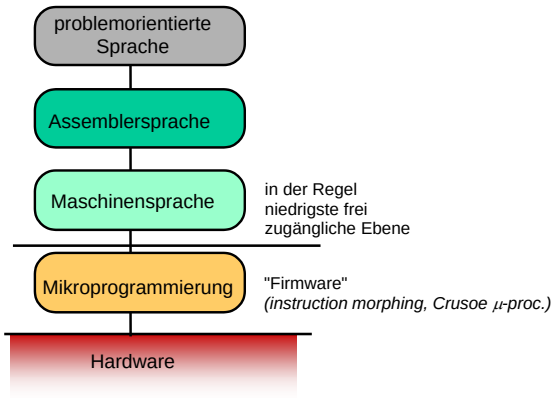
a_1	a_0	b_1	b_0	e_3	e_2	e_1	e_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Aufgabe 8. Programmiersprachen (4 + 2 + 4 = 10 Punkte)

(a) Ordnen Sie folgende Programmiersprachen nach ihrem Abstraktionsgrad von der dem System zugrundeliegenden Hardware: Maschinensprache, Java, Mikroprogrammierung, Assembler, C.

Lösung: Java muss über C stehen oder auf der gleichen Ebene.

Punktevergabe: 4 - kleinstmögliche Anzahl an Permutationen, sodass man auf die richtige Lösung kommt.

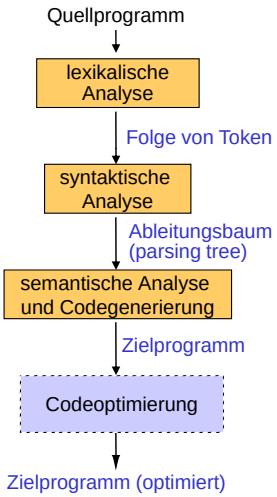


(b) Warum muss der Assembler zwei Läufe durchführen, um aus einem Assemblerprogramm ein gültiges Maschinenprogramm zu erzeugen?

Lösung: Weil die Labels im ersten Durchlauf aufgelöst werden müssen.

(c) Welche vier Schritte führt ein Compiler typischerweise auf dem Weg vom Quelltext zum Zielprogramm aus?

Lösung: Es reichen die Begriffe aus den Kästen. Die Reihenfolge war nicht gefragt.



Aufgabe 9. Adressierung (6 Punkte)

Gegeben sei folgender Speicherinhalt.

Index
2

Speicher	
Adresse	Inhalt
0	8
1	1
2	0
3	3
4	5
5	12
6	7
7	6
8	4
9	2

Welchen Wert liest der Befehl "`LOAD 2`" jeweils aus, wenn es sich um unmittelbare, direkte, indirekte oder indizierte Adressierung handelt?

2 (1 Punkt)	0 (1 Punkt)	8 (2 Punkte)	5 (2 Punkte)
unmittelbar	direkt	indirekt	indiziert

Aufgabe 10. Betriebssysteme (2 + 4 + 4 = 10 Punkte)

(a) Was ist der Unterschied zwischen einem Prozess und einem Thread?

Lösung:
Jeder Prozess hat einen eigenen Adressraum, während verschiedene Threads, die zu einer Anwendung gehören, den gleichen Adressraum verwenden.

(b) Welche Probleme können beim Zeitscheibenverfahren (Round-Robin) mit sehr großen Zeitsegmenten auftreten und welche bei sehr kurzen Zeitsegmenten? (Gemeint ist eine unterschiedliche Umlaufdauer bei gleicher Prozessanzahl.)

Lösung:

- Bei zu kurzen Zeitsegmenten: Overhead durch Implementierung der Round-Robin-Technologie nimmt im Verhältnis zu; ...
- Bei zu langen Zeitsegmenten: Parallelität geht immer mehr verloren (im Extremfall wird jeder Prozess vollständig abgearbeitet, bevor der nächste drankommt); ...

Sinnvolle Argumente: zwischen 1 und 2 Punkte. Es können auch mehr als 2 Punkte für „zu kurz“ bzw. „zu lang“ vergeben werden. (Max. 4 Punkte.)

(c) Was ist ein Seitenfehler? Wie reagiert das Betriebssystem darauf?

Lösung: Ein Seitenfehler tritt auf, wenn eine benötigte Seite (Block in Adressraum) nicht im Hauptspeicher ist. (1 Punkt)

Reaktion:

- Falls es einen freien Seitenrahmen ρ gibt: Seite wird in Seitenrahmen ρ kopiert. (1,5 Punkte)
- Falls es keinen freien Seitenrahmen gibt: Inhalt eines Seitenrahmens ρ wird auf den Hintergrundspeicher kopiert (meist an den ursprünglichen Platz). Danach wird wie im ersten Fall verfahren. (1,5 Punkte)