Lösung

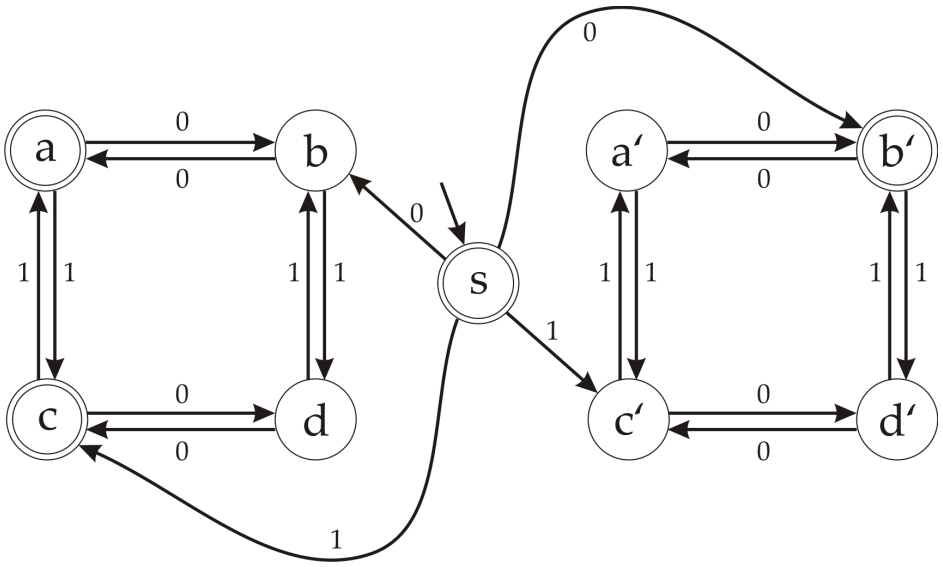
Aufgabe 1. (5 + 4 + 3 = 12 Punkte)

Gegeben sei der nichtdeterministische endliche Automat $A = (E, S, \delta, s, F)$ mit

$$E = \{0, 1\}, S = \{a, b, c, d, a', b', c', d', s\}, F = \{a, c, b', s\}.$$

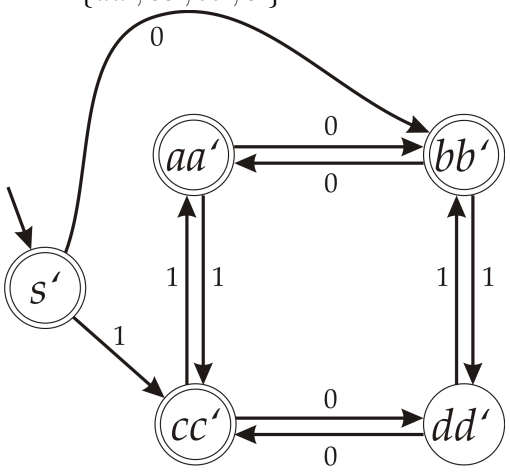
Durch das unten angegebene Zustandsdiagramm sei δ definiert.

(a) Erzeugen Sie einen äquivalenten deterministischen endlichen Automaten A' nach dem aus der Vorlesung bekannten Verfahren. Nutzen Sie die Tabelle für den Algorithmus. Definieren Sie den neuen Automaten vollständig.



Lösung:

$A' = (E, S', \delta', s', F')$,
$E = \{0, 1\}$, $S' = \{aa', bb', cc', dd', s'\}$,
$F' = \{aa', cc', bb', s'\}$.



Zustandsmengen	0	1
$\{s\}$	$\{b, b'\}$	$\{c, c'\}$
$\{b, b'\}$	$\{a, a'\}$	$\{d, d'\}$
$\{c, c'\}$	$\{d, d'\}$	$\{a, a'\}$
$\{a, a'\}$	$\{b, b'\}$	$\{c, c'\}$
$\{d, d'\}$	$\{c, c'\}$	$\{b, b'\}$

(b) Welche Zustände im neuen, deterministischen Automaten A' sind k-äquivalent für $k = 0, 1$? Einelementige Mengen müssen Sie nicht angeben.

k	k-äquivalente Zustandsmengen
0	{aa', bb', cc', s'}
1	{aa', s'}

(c) Welche Sprache $L(A)$ erkennt A (oder der äquivalente Automat A')? Definieren Sie $L(A)$ mathematisch oder formulieren Sie umgangssprachlich präzise.

Lösung: $L(A) = \{w \in \{0,1\}^* \mid |w|_0 \bmod 2 = 0 \text{ oder } |w|_1 \bmod 2 = 0\}$.

Das ist fast derselbe Automat wie in der Bonusklausur, nur dass zwei Zustände mehr Endzustände sind. Das hat zur Folge, dass man jetzt nicht nur die Wörter erkennt, die eine gerade Anzahl von Einsen und Nullen enthalten, sondern auch diejenigen, die eine gerade Anzahl von Einsen oder Nullen enthalten.

Beispiele: $00011, \lambda, 10000 \in L(A)$. (Nur Beispiele geben hier keine Punkte.)

Aufgabe 2. (10 Punkte)

Gegeben sei die Sprache $L = \{w \in \{0,1\}^* \mid \exists n \in \mathbb{N}_0 \colon w = (01)^{2n}\}$.

Geben Sie eine Turingmaschine $M = (S, E, B, \delta, s_0, F)$ an, die für Wörter $w \in L$ die gesamte Eingabe mit Einsen überschreibt und für alle anderen Wörter $w' \notin L$ die gesamte Eingabe mit Nullen überschreibt. In beiden Fällen soll M nach der Bearbeitung in einem Endzustand stoppen. Definieren Sie M vollständig.

Hinweise:

- Wenn die Eingabe das leere Wort ist, muss auch nichts überschrieben werden; M soll in diesem Fall nur in einem Endzustand halten.
- Sie können wie üblich annehmen, dass der Kopf von M zu Beginn auf dem linkesten Zeichen $e \in E$ der Eingabe steht.

Lösung:
$M = (S, E, B, \delta, s_0, F)$.
$S = \{s_0, s_1, s_2, s_3, s_N, s_{NN}, s_{EE}\}$.
$E = \{0, 1\}$.
$B = \{0, 1, \star\}$.
$F = \{s_{EE}, s_{NN}\}$.

Zustand	0	1	$\star$
s_0	$(s_1, 1, R)$	$(s_N, 0, N)$	$(s_{EE}, \star, N)$
s_1	$(s_N, 0, N)$	$(s_2, 1, R)$	$(s_N, \star, N)$
s_2	$(s_3, 1, R)$	$(s_N, 0, N)$	$(s_N, \star, N)$
s_3	$(s_N, 0, N)$	$(s_0, 1, R)$	$(s_N, \star, N)$
s_N	$(s_N, 0, R)$	$(s_N, 1, R)$	$(s_{NN}, \star, L)$
s_{NN}	$(s_{NN}, 0, L)$	$(s_{NN}, 0, L)$	
s_{EE}			

Aufgabe 3. $(2 + 4 + 4 = 10$ Punkte)

Gegeben sei die Grammatik $G = (N, T, P, S)$ mit:

$$\begin{aligned} N &= \{A, B, C, S\}, \\ T &= \{a, b, c, d\}, \\ P &= \{S \to ABCS|\lambda, \\ &\quad A \to aA \mid a, \\ &\quad B \to bB \mid b, \\ &\quad C \to cC \mid c, \\ &\quad AB \to dd\}. \end{aligned}$$

(a) Zu welchem Typ / welchen Typen der Chomsky-Hierarchie gehört diese Grammatik? Nennen Sie alle passenden Typen.

Lösung: Nur Typ 0, denn die Regel $AB \to dd$ ist nicht rechtslinear und kontextfrei; außerdem ist sie zwar monoton, aber nicht kontextsensitiv (die Sprache $L(G)$ ist also trotzdem eine Typ-1-Sprache).

(b) Geben Sie einen regulären Ausdruck R an mit $L(R) = L(G)$.

Lösung: $((a^+b^+ + a^\star dd)\, c^+)^\star$, mit $x^+ := xx^*$

(c) Von welchem Chomsky-Typ ist nach dem Ergebnis aus (b) die von G definierte Sprache $L(G)$? Geben Sie eine Grammatik G' dieses Typs an, sodass gilt: $L(G') = L(G)$. Definieren Sie die Grammatik vollständig.

Lösung: Vom Typ 3. Rechtslineare Grammatik:

$$\begin{aligned} G' &= (\{S, A, B, C, D_1, D_2\}, \{a, b, c, d\}, P', S) \\ P' &= \{S \to aA \mid aB \mid aD_1 \mid dD_2 \mid \lambda \\ &\quad A \to aA \mid aB \\ &\quad B \to bB \mid bC \\ &\quad C \to cC \mid cS \\ &\quad D_1 \to aD_1 \mid dD_2 \\ &\quad D_2 \to dC\} \end{aligned}$$

Aufgabe 4. (3 + 3 + 3 = 9 Punkte)

Seien P und NP die aus der Vorlesung bekannten Komplexitätsklassen, SAT das Erfüllbarkeitsproblem der Aussagenlogik und $\leq_{pol}$ die aus der Vorlesung bekannte Relation mit der Bedeutung:

$$X \leq_{pol} Y \Leftrightarrow X \text{ ist in Polynomialzeit reduzierbar auf } Y.$$

(a) Beschreiben Sie anhand von P und NP, was eine (Zeit-) Komplexitätsklasse ist:
- Was sind die Elemente einer solchen Klasse?
- Welche gemeinsamen Eigenschaften haben diese Elemente?
- Was ist dabei der Unterschied zwischen P und NP?

(b) Zu welcher der Klassen P, NP gehört SAT? Durch welche weitere bekannte Eigenschaft von SAT kann man dessen Position in der Komplexitätsordnung dieser Klasse noch genauer charakterisieren?

(c) Für zwei Probleme X, Y sei folgendes gezeigt worden:

$$X \leq_{pol} SAT \leq_{pol} Y.$$

In welche Komplexitätsklassen kann man X und Y folglich einordnen?

Lösung:

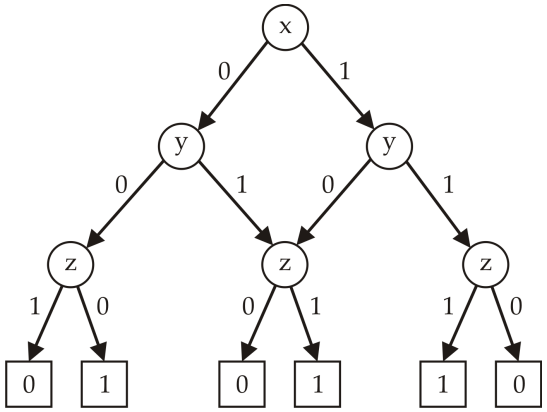
(a)
- Die Elemente sind Probleme.
- Die gemeinsame Eigenschaft ist, dass sie sich innerhalb gewisser Zeitschranken lösen lassen (für P und NP ist diese Zeitschranke durch die Menge der Polynome definiert).
- Bei P muss die Berechnung deterministisch sein, bei NP darf sie nichtdeterministisch sein.

(b) SAT liegt in NP, aber nicht in P. Durch dessen NP-Vollständigkeit bzw. NP-Schwere gehört es in dieser Charakterisierung der Problem-Komplexität (also bzgl. der Polynomialzeitreduktion) zu den schwersten Problemen in NP.

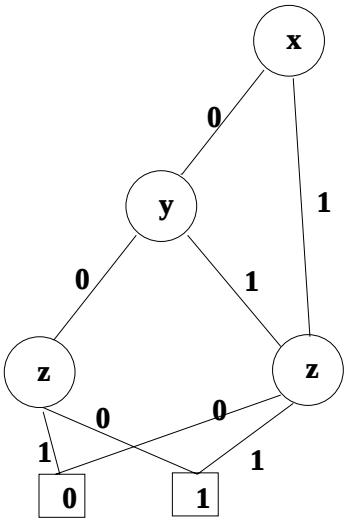
(c) $X \in NP$, Y ist NP-schwer.

Aufgabe 5. (5 + 3 = 8 Punkte)

(a) Gegeben sei die folgende Baumdarstellung einer Booleschen Funktion $f\colon \mathbb{B}^3 \to \mathbb{B}$. Erzeugen Sie das zu f gehörende BDD (Binary Decision Diagram).



Lösung:



(b) Geben Sie die Funktion f aus Aufgabe (a) als Booleschen Ausdruck an.

$f(x, y, z) =$ **Lösung:**

$f(x, y, z) = x.z + y.z + \bar{x}.\bar{y}.\bar{z}$

$f(x, y, z) = x.z + y.z + \bar{y}.\bar{z}$

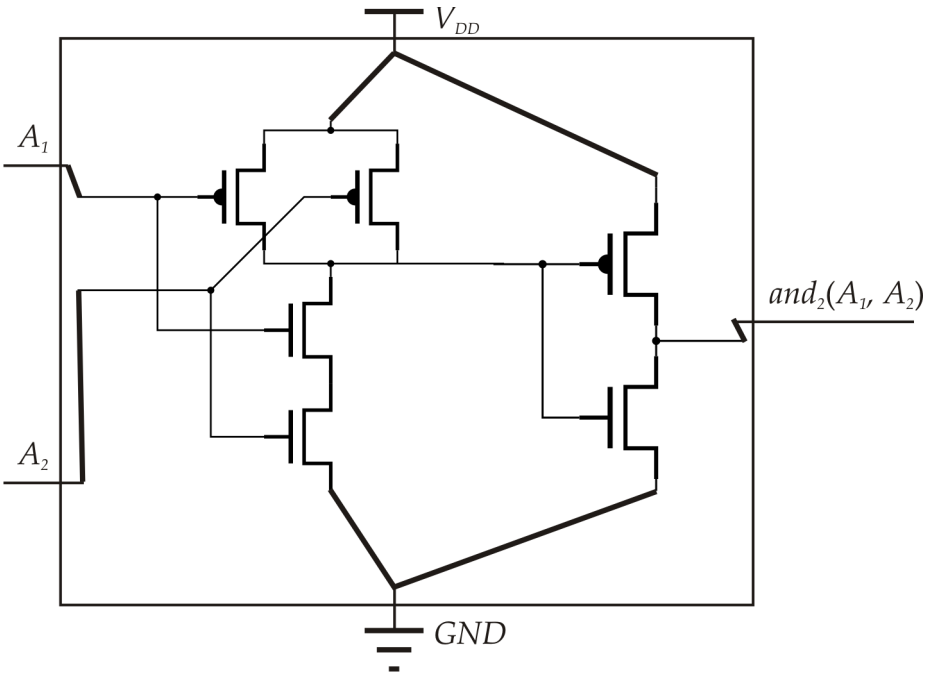
$f(x, y, z) = x.z + \bar{x}.y.z + \bar{x}.\bar{y}.\bar{z}$

Aufgabe 6. $(5 + 2 + 2 = 9$ Punkte)

Die Boolesche Funktion $and_2 \colon \mathbb{B}^2 \to \mathbb{B}$ sei für $A_1, A_2 \in \mathbb{B}$ definiert als

$$and_2(A_1, A_2) = A_1 \wedge A_2.$$

(a) Zeichnen Sie in das vorgegebene Feld eine CMOS-Schaltung, die and_2 implementiert.



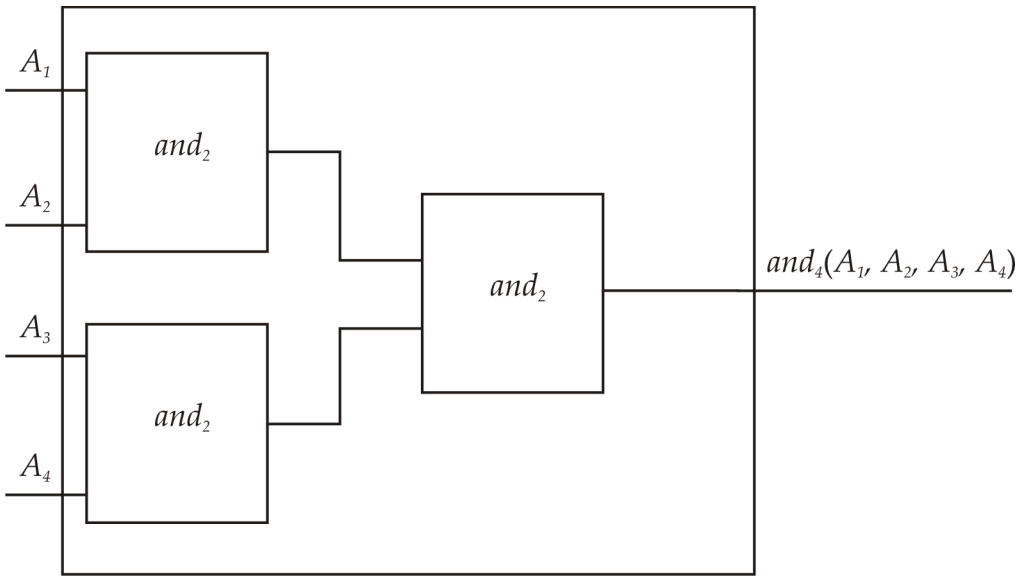
Hinweis: Überlegen Sie zunächst, welche CMOS-Schaltungen Sie bereits kennen und wie man and_2 durch diese (eventuell durch Verknüpfung mehrerer) darstellen kann.

Lösung: Bspw. durch Hintereinanderschaltung von $NAND$ und NOT.

(b) Es sei nun für jedes feste $n \in \mathbb{N}, n \geq 3$, die Funktion $and_n \colon \mathbb{B}^n \to \mathbb{B}$ gegeben, die auf Basis von and_2 aus Aufgabe (a) für $A_1, ..., A_n \in \mathbb{B}$ rekursiv definiert ist als

$$and_n(A_1, ..., A_n) = and_2(and_{n-1}(A_1, ..., A_{n-1}), A_n)$$

Zeichnen Sie eine Schaltung, die and_4 implementiert. Benutzen Sie dafür das and_2-Gatter aus Aufgabenteil (a) als Blackbox (Sie müssen hier also keine CMOS-Elemente mehr zeichnen). Benutzen Sie keine anderen Gatter.



Hinweis: Sie dürfen das and_2-Gatter auch verwenden, wenn Sie Aufgabenteil (a) nicht gelöst haben.

Lösung: Bspw. als „$((A_1 \wedge A_2) \wedge (A_3 \wedge A_4))$“, siehe Bild.

(c) Wie muss man die and_2-Gatter anordnen, damit die Signallaufzeit einer Schaltung für and_n möglichst kurz ist? Wie groß ist die Signallaufzeit dann in Abhängigkeit von n (in O-Notation)?

Hinweis: Mit „Signallaufzeit“ ist die Zeit gemeint, die die Schaltung vom Anlegen eines Signals am Eingang bis zur korrekten Ausgabe am Ausgang braucht, wenn man annimmt, dass das and_2-Gatter eine konstante Zeit zum Schalten braucht.

Lösung: Durch Hierarchische Anordnung und damit Eingeführte parallele Abarbeitung bekommt man $O(log(n))$: Bsp. „$((A_1 \wedge A_2) \wedge (A_3 \wedge A_4))$“

Aufgabe 7. $(1 + 2 + 4 = 7$ Punkte)

Es sei für ein Alphabet A eine 3-Bit-Kodierung $c\colon A \to \mathbb{B}^3$ gegeben, die folgende Codes enthält:

c: 001
010
011
100
101
110

(a) Was ist die Hammingzahl der Kodierung c?

hc = 1

(b) Erweitern Sie die Kodierung c um höchstens 1 Bit, sodass 1-Bit-Fehler erkannt werden können.

1-Prüfbit -> gerade Anzahl von 1s

c: 1 001
1 010
0 011
1 100
0 101
0 110

(c) Erweitern Sie die Kodierung c um höchstens 3 Bits, sodass 2-Bit-Fehler erkannt werden können.

2-bits Kontrolziffer : sagt auf welche Stelle die 1 oder die 0 ist

1-Prüfbit Bit –> gerade Anzahl von 1s

c: 01 1 001
10 1 010
11 0 011
11 1 100
10 0 101
01 0 110

Aufgabe 8. (6 Punkte)

Gegeben seien die Zahlen A, B in der bekannten IEEE-754-Darstellung:

$$A \hat{=} 0\,10000011\,10010100000000000000000$$
$$B \hat{=} 0\,10000010\,10011000000000000000000$$

Die Darstellung hat folgende Form:

$$v\, c_7 ... c_0\, m_1 ... m_{23}.$$

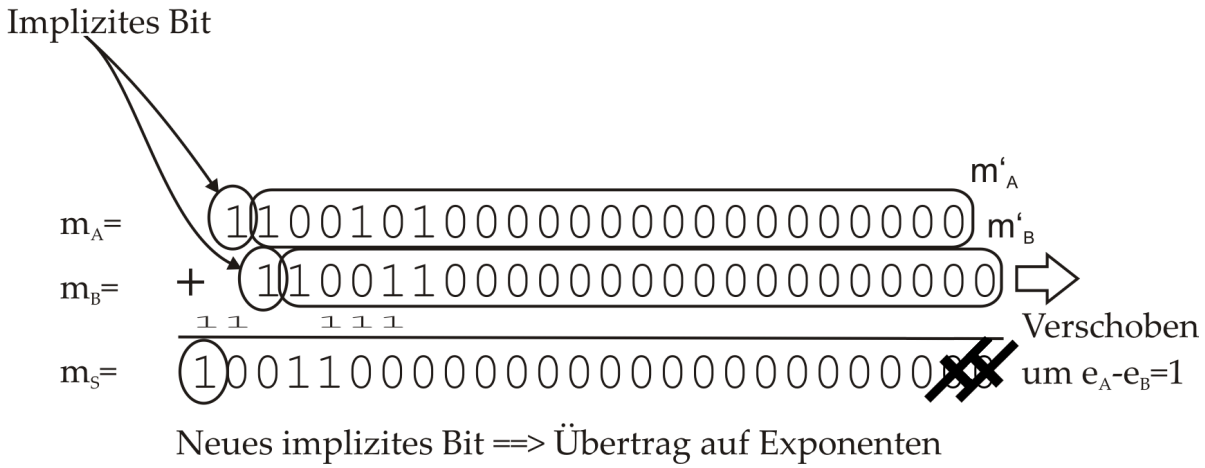
v ist das Vorzeichen-Bit, $c_0, ..., c_7$ sind die Bits der Charakteristik und $m_1, ..., m_{23}$ die Bits der Mantisse. Beachten Sie, dass das höchstwertige Bit der Mantisse links ist.

Berechnen Sie die Summe $S = A + B$ der beiden Zahlen und geben Sie S in IEEE-754-Darstellung an.

Hinweis: Eine Umwandlung in reelle Zahlen ist nicht erforderlich.

Lösung:

- Ohne Umwandlung in reelle Zahlen.



e_A ist größerer Exponent, muss wegen neuem impl. Bit um 1 erhöht werden:

c_S=10000100

Vorzeichen bleibt gleich. Lösung:
S: 0 10000100 00110000000000000000000

- Mit Umwandlung in reelle Zahlen.

A:

$v = 0$, $c = 10000011$ –> $2^7 + 2^1 + 2^0 = 131$ -> e = 131 -127 = 4

10010100000000000000000 -> $2^{-1} + 2^{-4} + 2^{-6}$

Zahl = $2^4(1 + m') = 2^4(1 + 2^{-1} + 2^{-4} + 2^{-6}) = 25.25$

B:

$v = 0$, $c = 10000010$ –> $2^7 + 2^1 = 130$ -> e = 131 -127 = 3

10011000000000000000000 -> $2^{-1} + 2^{-4} + 2^{-5}$

Zahl = $2^3(1 + m') = 2^3(1 + 2^{-1} + 2^{-4} + 2^{-5}) = 12.75$

S:

Summe = 38 $v = 0$, $38 = 2^5 + 2^2 + 2^1 = 2^5(1 + 2^{-3} + 2^{-4})$ –> c = 5+127 = 132 : 10000100 und $m' = 00110000000000000000000$

Aufgabe 9. (2 + 2 + 3 = 7 Punkte)

Der Speicher einer Rechenanlage sei in folgenden zwei Stufen organisiert:

- Prozessorcache, Zugriffszeit 1 ns,
- Hauptspeicher, Zugriffszeit 20 ns.

Bei der Ausführung eines Programms treten 5 % Cachefehler auf.

(a) Berechnen Sie die durchschnittliche Zugriffszeit.

Lösung: $0,95 * 1ns + 0,05 * 20ns = 1,95ns$

(b) Wieviel Zeit für Speicherzugriffe benötigt ein Programm auf dieser Rechenanlage, wenn es auf der gleichen Rechenanlage ohne Cache dafür 20 Sekunden benötigen würde?

Lösung: $20s/20ns = 10^9$ Anzahl Instruktionen

$10^9 * 1,95ns = 1,95s$

(c) Wieviel Zeit für Speicherzugriffe benötigt das Programm aus (b) auf einer Rechenanlage, bei der für die beiden Speicherstufen folgende Angaben gelten?

- Prozessorcache, Zugriffszeit 0,5 ns,
- Hauptspeicher, Zugriffszeit 50 ns.

Lösung: $10^9 * (0,95 * 0,5 + 0,05 * 50)ns = 10^9 * (0,475 + 2,5)ns = 2,975s$

Aufgabe 10. (4 + 2 = 6 Punkte)

Die Befehle der in der Vorlesung vorgestellten Assembler-Sprache sind folgendermaßen aufgebaut, wobei 'Q' für Quelle steht und 'Z' für Ziel:

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

Für unmittelbare Adressierung wird das Präfix '#' verwendet. Ein bedingter Sprungbefehl ist JNZ (JumpNotZero), der den Befehlszähler genau dann zu Label L springen lässt, falls $Q \neq 0$. Die Notation des Sprungbefehls ist:

```
JNZ Q L
```

Gegeben sei der folgende Ausschnitt eines Assemblerprogramms.

```
    STORE R1, R2
    STORE #1, R3
L1: MULTIPLY R3, R2, R3
    SUBTRACT R1, #1, R1
    JNZ R1, L1
    STORE R3, R2
```

(a) Zu Beginn stehe in Register $R1$ ein Wert $n > 0, n \in \mathbb{N}$. Wie hängt der Endwert von $R2$ mit dem Anfangswert von $R1$ zusammen?

(b) Was geschieht, wenn zu Beginn der Wert $n = 0$ in $R1$ steht?

Lösung: (a) Es wird für $n > 0$ die Potenzfunktion $R1^{R1} = n^n$ berechnet; (b) für $n = 0$ entsteht eine Endlosschleife bzw. irgendwann ein Überlauf der Variablen R1.

Aufgabe 11. (4 + 2 = 6 Punkte)

Es sei die Menge $\Pi = \{P_1, P_2, P_3, P_4\}$ von Prozessen gegeben, deren benötigte CPU-Zeit $t(P_i)$ für $i \in \{1, 2, 3, 4\}$ in der folgenden Tabelle dargestellt ist.

Prozess P_i	Benötigte CPU-Zeit $t(P_i)$
P_1	56 ms
P_2	12 ms
P_3	10 ms
P_4	32 ms

(a) Benutzen Sie ein *prioritätsgesteuertes Zuteilungsverfahren*, bei dem für die Prioritäten $p(P_i)$, $p(P_j)$ der Prozesse P_i, P_j gilt:

$$t(P_i) \leq t(P_j) \Rightarrow p(P_i) \geq p(P_j),$$

um die Rechenzeit zuzuteilen. Zeichnen Sie auf dem ersten vorgegeben Zeitstrahl die Zuteilung der Rechenzeit auf die Prozesse ein (unter der Annahme, dass die Rechnung bei 0 ms startet).

(b) Benutzen Sie nun das *Zeitscheibenverfahren (Round-Robin)* mit 10 ms Zeiteinheit für die Zuteilung und zeichnen Sie das Ergebnis auf dem zweiten vorgegebenen Zeitstrahl ein (unter der Annahme, dass die Rechnung bei 0 ms startet).