# Fragen und Antworten zu Einführung in die Technische Informatik

Tonico

11. Oktober 2011

## Inhaltsverzeichnis



# 1 Einleitung

Die Fragen wurden aus diversen Ausarbeitungen (vor allem die von Paulchen) und After-Test-Threads extrahiert und um eigene Fragen ergänzt. Die Kapitelnummerierung sollte mit der im Buch übereinstimmen.

Die Ausarbeitung ist bei weitem nicht vollständig und ich bin mir fast sicher dass noch einige Fehler enthalten sind, also am besten alles nochmal nachprüfen! ;)

# Hardware

## 2 Logische Schaltungen

**2.1** Geben Sie die Übertragungskennlinie $u_a = f(u_e)$ eines Inverters an. Wo liegt die verbotene Zone? Wodurch sind die Grenzen der verbotenen Zone festgelegt? (S. 8)

Die Verbotene Zone liegt im steilen Teil der Übertragungskennlinke wo die Steigung dem Betrag nach >= 1 ist. Vgl. Grafik im Buch.

**2.2** NOR Schaltplan (3 Stück 3 Variablen) mit NAND darstellen. (S. 9)

Zum Merken, Umformungen für NAND Gatter:

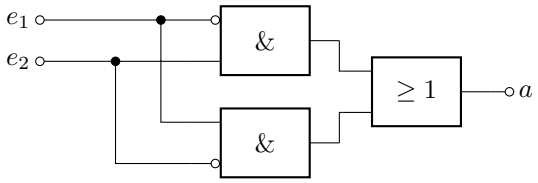
$$\begin{aligned} \neg A &= \neg(A \wedge 1) \\ A \wedge B &= \neg(\neg(A \wedge B) \wedge 1) \\ A \vee B &= \neg(\neg(A \wedge 1) \wedge \neg(B \wedge 1)) \end{aligned}$$

Umformungen für NOR Gatter:

$$\begin{aligned} \neg A &= \neg(A \vee 0) \\ A \wedge B &= \neg(\neg(A \vee 0) \vee \neg(B \vee 0)) \\ A \vee B &= \neg(\neg(A \vee B) \vee 0) \end{aligned}$$

**2.3** Zeichnen Sie eine Gatterschaltung für eine XOR-Funktion! (S. 9)

$a = e_1 \otimes e_2 = (e_1 \wedge \neg e_2) \vee (\neg e_1 \wedge e_2)$.




**2.4** Was sind 0-aktive Eingänge bei ICs (integrierten Schaltungen)? (S. 10)

Der Eingang wird mit logisch 0 aktiv und ist bei logisch 1 im Ruhezustand.

**2.5** Was versteht man unter „Fan-Out"? (S. 11)

Die maximale Anzahl an Eingängen die an einem Ausgang eines Gatters angeschlossen werden können ohne seine Funktionalität zu beeinträchtigen.

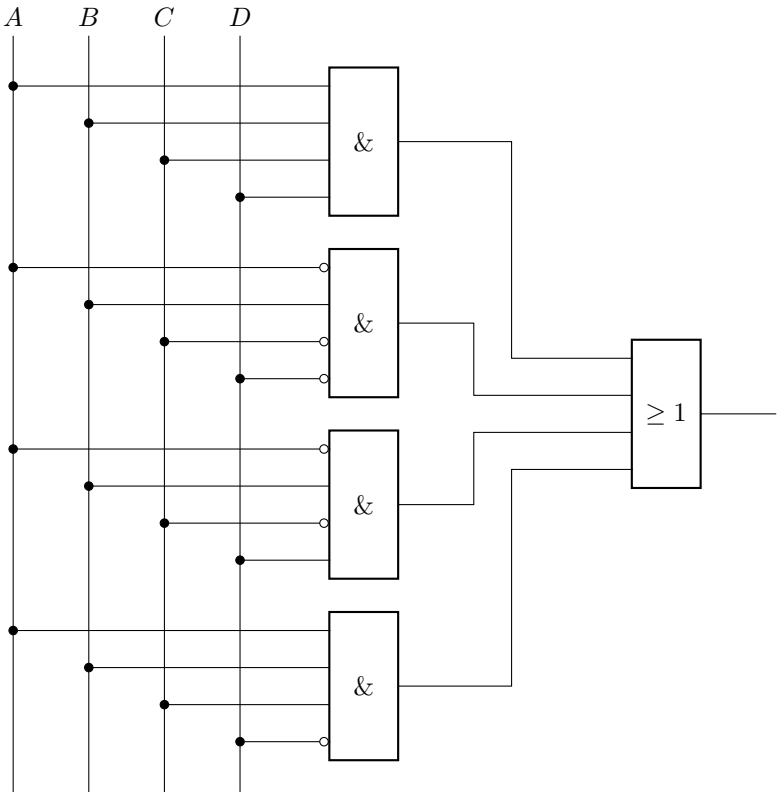
**2.6** Werden bei elektronischen Schaltungen mehrere Eingänge an einem Ausgang angeschlossen, wirkt eine kapazitive Belastung auf diesen Ausgang. Bei jedem Umschaltvorgang muss nun die gesamte Kapazität über den Innenwiderstand des Ausgangs auf- bzw. entladen werden. Geben Sie die Formel für den Aufladevorgang an und berechnen Sie, nach welcher Zeit (ausgedrückt als Vielfaches der Zeitkonstante $\tau$) 30% des Endwertes der Spannung $U_0$ erreicht werden. (S. 12)

Aufladevorgang: $u_C(t) = u_0 \cdot (1 - e^{-\frac{t}{\tau}})$ mit $u_0 \ldots$ asymptotischer Endwert

Entladevorgang: $u_C(t)) = u_0 \cdot e^{-\frac{t}{\tau}}$ mit $u_0 \ldots$ Anfangswert

30% beim Aufladen nach: $t = -\tau \cdot \ln \frac{7}{10} = \tau \cdot \ln \frac{10}{7} =$ (in den Taschenrechner, falls erlaubt)

**2.7** Vereinfachen Sie die folgende Schaltung mit Hilfe eines KV-Diagramms und zeichnen Sie die resultierende Schaltung.



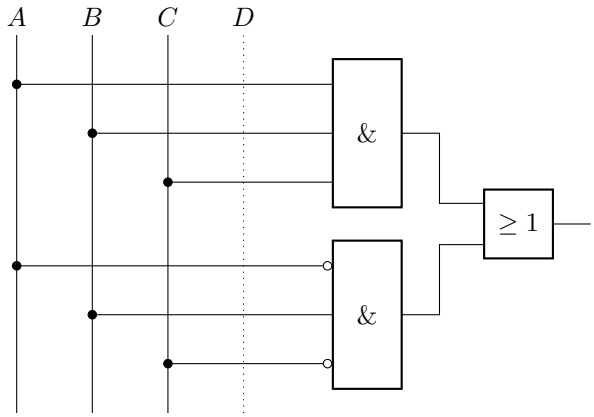

(S. 21)

KV-Diagramm:

| | | $A$ | | $\overline{A}$ | |
|---|---|---|---|---|---|
| $B$ | | 1 | 0 | 1 | 0 |
| | | 1 | 0 | 1 | 0 |
| $\overline{B}$ | | 0 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0 |
| | | $C$ | $\overline{C}$ | $C$ | |

$D$, $\overline{D}$, $D$

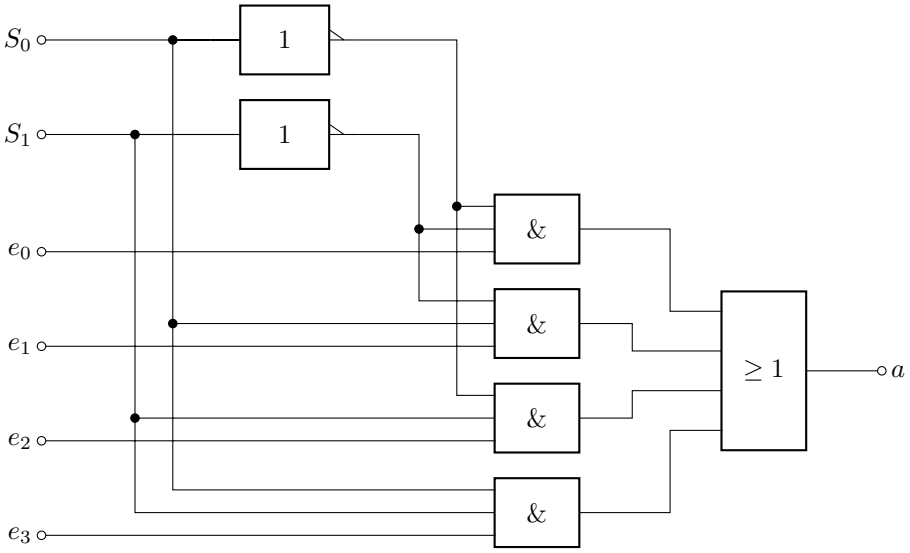
Vereinfachte Ausgangsfunktion $a = (A \wedge B \wedge C) \vee (\overline{A} \wedge B \wedge \overline{C})$ und resultierende Schaltung:




**2.8** Welche Funktion hat ein Enable-Eingang bei einem Decoder? (S. 22)

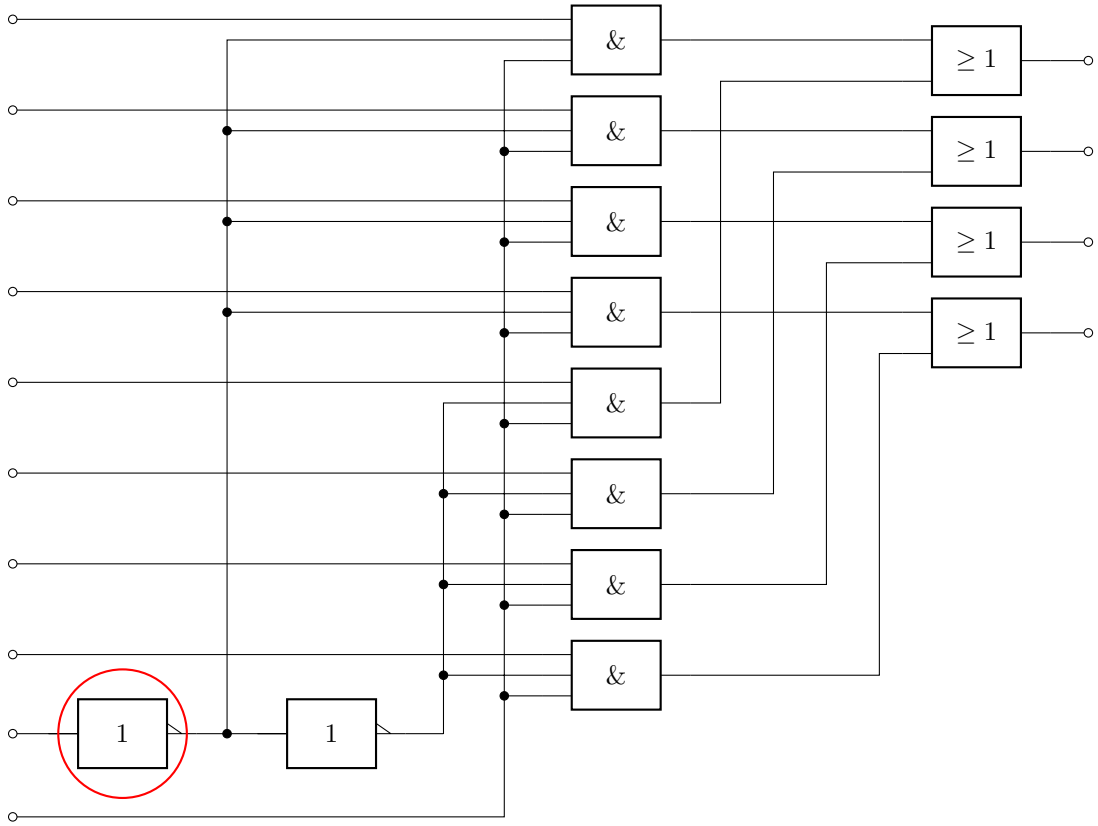
Das Eingangssignal wird decodiert wenn am Enable-Eingang logisch 1 anliegt, ansonsten wird an den Ausgängen logisch 0 ausgegeben.

**2.9** Gegeben ist ein Multiplexer MUX mit vier Eingängen $e_0$, $e_1$, $e_2$ und $e_3$ sowie zwei Steuereingängen $S_0$und $S_1$. Ergänzen Sie die Schaltung!




(S. 22)

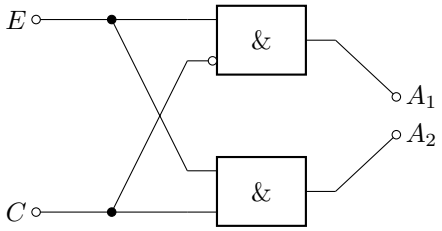
**2.10** Wozu dient der durch einen Kreis gekennzeichnete Inverter der dargestellten Multiplex-Schaltung?




(S. 23)

Der Inverter dient der Entkopplung zwischen der treibenden Schaltung am S-Eingang und dem Multiplexer.

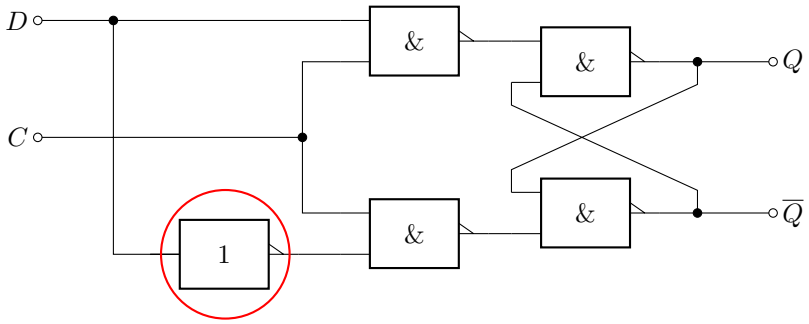
**2.11** Konstruieren Sie eine Gatterschaltung für einen Demultiplexer mit einem Informationseingang $E$ und einem Controlleingang $C$ sowie den Ausgängen $A_1$und $A_2$, wobei $C$ die Werte 0 oder 1 annehmen kann! (S. 24)




**2.12** Ist $R = S = 1$ bei einem RS-Latch zulässig? (S. 27)

Nein, dies würde bedeuten, dass $Q$ und $\overline{Q}$ logisch 0 wären.

**2.13** Gegeben ist die Schaltung eines D-FF. Erklären Sie die Funktion des gekennzeichneten Gatters!




(S. 27)

Der Inverter sorgt dafür, dass aus einem RS-Latch ein D-FF wird, indem das Signal von $D$ an den Eingang $S$ und das negierte Signal von $D$ auf den Eingang $R$ des ehemaligen RS-Latch gelegt wird. Dadurch sind $S$ und $R$ stets zueinander invertiert.

**2.14** Unterschied D- zu JK-FlipFlop (S. 28)

Beim D-FF bestimmen die Eingänge $C$ und $D$ den Folgezustand $Q$:

| $C$ | $D$ | $Q$ |
|---|---|---|
| 0 | X | Unverändert |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

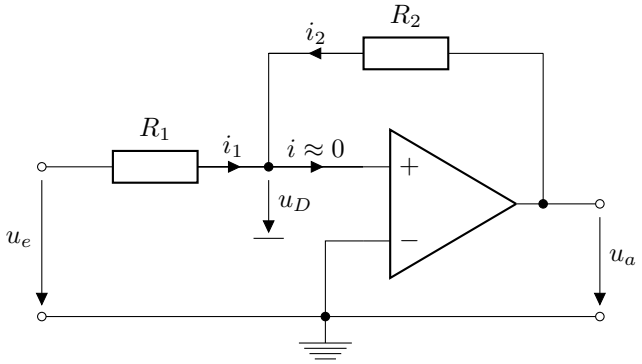
Beim JK-FlipFlop bestimmen die Eingänge $J$, $K$ und der Aktuelle Zustand $Q_{t-1}$ den Folgezustand $Q_t$:

| $J$ | $K$ | $Q_{t-1}$ | $Q_t$ |
|---|---|---|---|
| 0 | 0 | X | Unverändert |
| 0 | 1 | X | 0 |
| 1 | 0 | X | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**2.15** Gegeben ist eine Asynchronzähler, bestehend aus vier Latches und einem Zählstand von 7. Welche falschen Zählerzwischenstände treten beim Übergang von 7 auf 8 auf? (S. 35)

$(0110)_2$, $(0100)_2$, $(0000)_2$

**2.16** Gegeben ist die Schaltung eines invertierenden Operationsverstärkers. Wie groß ist die Spannungsverstärkung $v_u$? (S. 38)

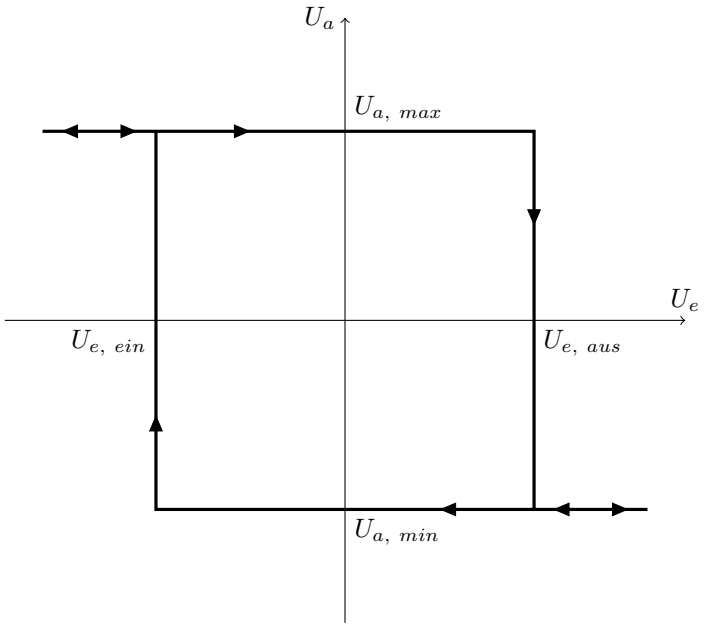



Die Spannungsverstärkung ist $v_u = \frac{u_a}{u_e} \approx -\frac{R_2}{R_1}$.

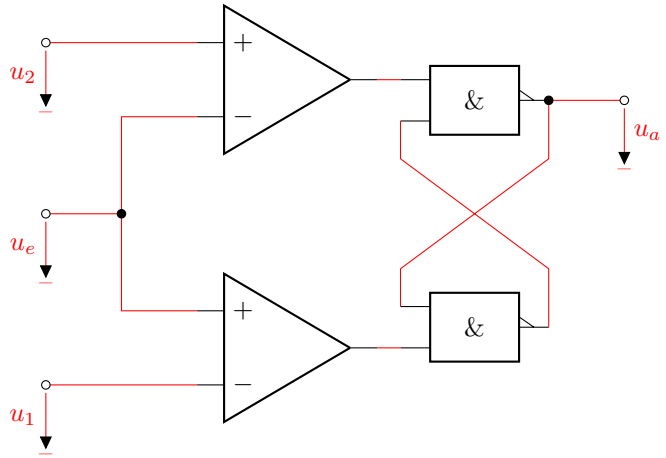
**2.17** Was ist die Hysterese beim Schmitt-Trigger? (S. 42)

Der Sachverhalt, dass die Übertragungskennlinie bei einem vollständigen Schaltzyklus unterschiedliche Wege beim Schalten durchläuft. Die Größe der Hysterese ergibt sich aus der Differenz zwischen unterer und oberer Schaltschwelle: $U_{HST} = U_{SO} - U_{SU}$.

**2.18** Zeichnen Sie die Übertragungskennlinie eines invertierenden Schmitt-Triggers! (S. 43)




**2.19** Vervollständigen Sie durch Einzeichnen der fehlenden Verbindungen die angegebene Schaltung eines Präzisions-Schmitt-Triggers!



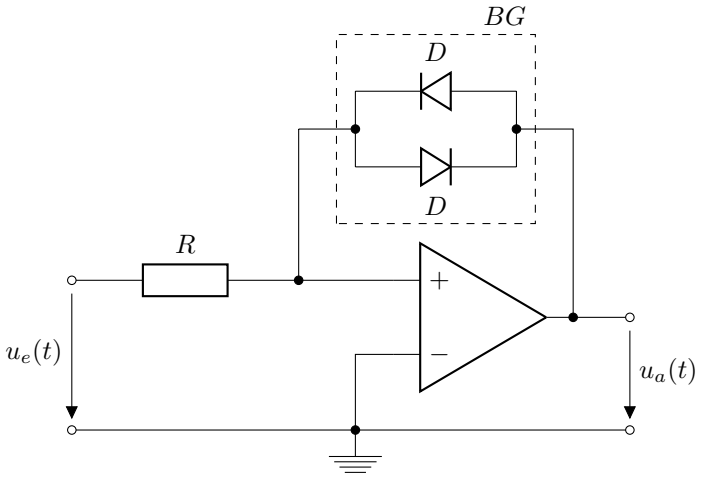

(S. 46)

**2.20** Was versteht man unter dem Tastgrag $g$ einer Rechteckimpulsfolge des Taktgenerators eines Rechners? (S. 46)

Der Tastgrad $g$ beschreibt das Verhältnis zwischen Impulsdauer $\tau_i$ und Impulsperiodendauer $T$:

$$g = \frac{\tau_i}{T}$$

**2.21** Erklären Sie die Funktion der anti-parallelen Diodenschaltung im Gegenkopplungszweig des Operationsverstärkers eines Zero-Crosssing-Detectors!




(S. 48)

Dient als Begrenzerschaltung zur Begrenzung der positiven und negativen Ausgangsspannung.

**2.22** Aufzeichnen des zeitlichen Verlaufs der Ausgangsspannung bei einem Zero-Crossing-Detector (Verlauf der Eingangsspannung gegeben) (S. 48)

Vgl. Grafik im Buch.

**2.23** Was ist die Funktion des nachtriggerbaren Univibrators? (S. 50)

Ein Univibrator besitzt einen Ausgang mit zwei Zuständen von denen einer für eine gewisse Verweilzeit stabil bleibt. Bei einem nachgetriggerten Univibrator wird die Verweilzeit bei jedem Triggerimpuls von neuem gestartet.

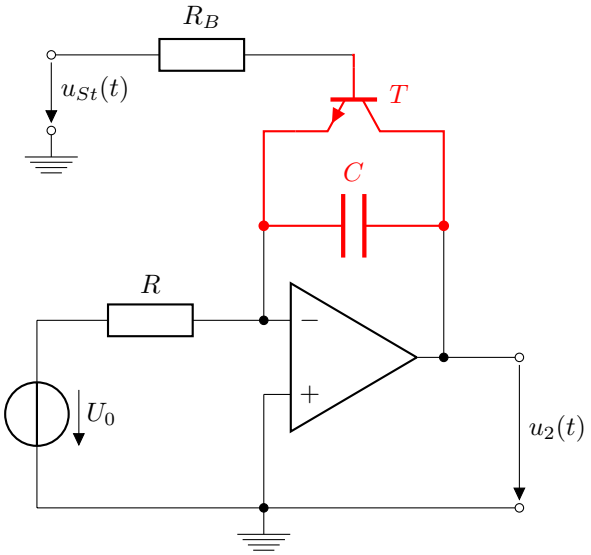
**2.24** Wozu dient die Ziehkapazität bei einem Quarzoszillator? (S. 52)

Zum Abgleich der Resonanzfrequenz bei Serienresonanz, die Ziehkapazität wird mit dem Quarzoszillator in Serie geschaltet.

**2.25** Wofür ist das dritte NAND-Gatter beim Quarzoszillator da? (S. 52)

Um eine Rechteck-Impulsfolge mit Flanken von hinreichender Flankensteilheit zu erreichen.

**2.26** Gegeben ist ein Sägezahngenerator mit Miller-Integrator und Schalttransistor $T$. Vervollständigen Sie die Schaltung. Das heißt, ergänzen Sie die fehlenden Bauteile und Leitungen und geben Sie an, wo sich der invertierende und der nicht invertierende Eingang des Operationsverstärkers befinden. (S. 54)




**2.27** Sägezahngenerator - Zeitdiagramm. (S. 54)

Vgl. Grafik im Buch.

**2.28** Geben Sie je 2 Beispiele für Tabellen- und Funktionsspeicher. (S. 63)

1. Tabellenspeicher: RAM, ROM.
2. Funktionsspeicher: PAL, PLA.

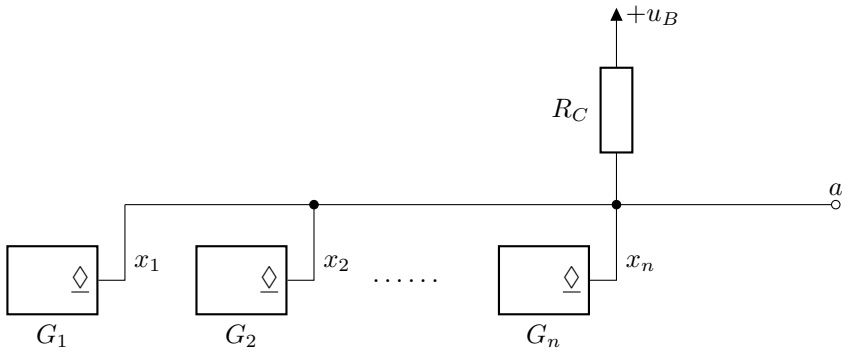
**2.29** Erläutern Sie die Funktion eines Tristate-Gatters! (S. 60)

Durch einen Steureingang kann der Output ausgeschaltet werden wodurch sich drei Zustände für den Output ergeben:

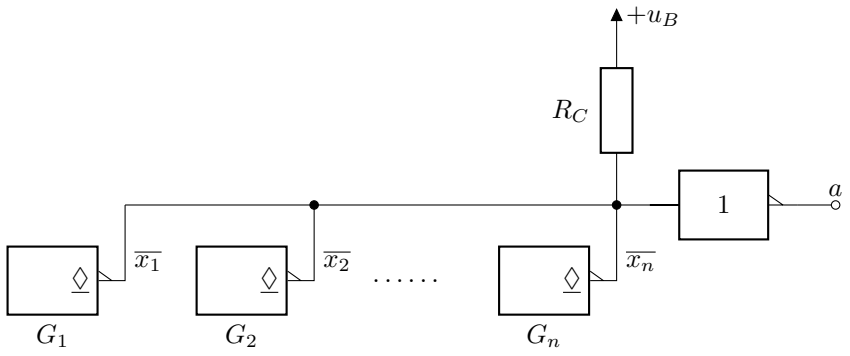
| In | S | Out |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| X | 0 | hochohmig |

Ausgänge solcher Gatter können zusammengeschaltet werden ohne dass dabei die Gatter wegen großer Ströme beschädigt werden.

**2.30** Welche Schaltungen liegen in den folgenden Abbildung vor?




$$a = x_1 \wedge x_2 \wedge \ldots \wedge x_n$$




$$a = x_1 \vee x_2 \vee \ldots \vee x_n$$

(S. 62)

Es handelt sich um Wired-AND bzw. Wired-OR Schaltungen die mit Open-Collector Ausgängen realisiert wurden.

**2.31** Erklären Sie die Abkürzung ASIC! (S. 64)

Application Specific Integrated Circuit. Ein ASIC ist ein Baustein und dient zum Speichern einer Funktion. Z.B. ein Programmable Array Logic (PLA) Baustein.

**2.32** Der Alarm eines Kührlschranks soll anschlagen, wenn der Temperatursensor über 12°C misst. Die Temperatur wird vom Sensor ganzzahlig in 0°C bis 15°C in 4 Bit-Darstellung geliefert.

Tragen Sie entsprechende Werte in unten stehende Wahrheitstabelle ein.

| Bit1 | Bit2 | Bit3 | Bit4 | >12°C |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

1. Wie lautet die minimale Boolesche Funktion $y$, die den Alarm ansteuert, also logisch 1 ausgibt, wenn die Temperatur 12°C überschreitet.
2. Skizzieren Sie die Funktion $y$ in untenstehendem PLA.
3. Berechnen Sie die maximale Taktfrequenz unter der Annahme, dass eine Negation 10ns und ein Gatter 20ns Durchlaufzeit hat.

(S. 65)

1. $y = Bit1 \wedge Bit2 \wedge (Bit3 \vee Bit4) = (Bit1 \wedge Bit2 \wedge Bit3) \vee (Bit1 \wedge Bit2 \wedge Bit4)$
2. Einfach Verbindungspunkte einzeichnen (vgl. Prüfungsbogen vom 9.12.2008, Bsp. 3).
3. $T = \frac{1}{20+20}$ (es werden keine NOT-Gatter durchlaufen)

**2.33** Ist es einfacher eine disjunktive oder konjunktive Aussageform in ein in NAND/NAND Struktur aufgebautes PLA zu übertragen? (S. 65)

Es ist einfacher die disjunktive Aussageform zu verwenden, weil die zu setzenden Verbindungen direkt daraus ablesbar sind. Z.B. ist $(A \wedge B) \vee (B \wedge C)$ logisch äquivalent zu $\neg((A \wedge B) \wedge (\neg A \wedge \neg C))$.

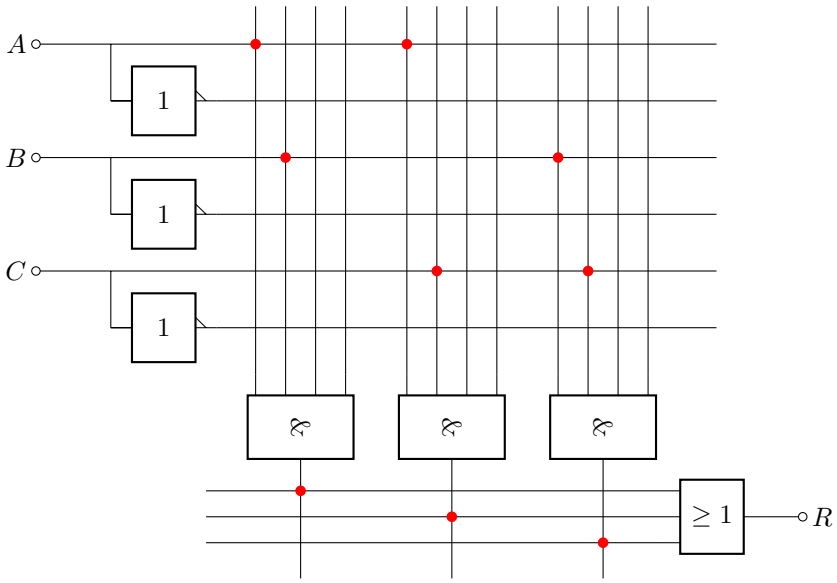
**2.34** Welche Möglichkeiten kennen Sie, Verbindungen in einem IC nach der Fertigung zu trennen bzw. zu schließen? (S. 65)

1. Einmal Programmierbar mit Schmelzsicherung: Nicht benötigte Leitungen werden durch einen Stromimpuls durchgeschmozen.

2. Mehrmals Programmierbar durch Transistorschalter. Die Verbindungen werden durch einen Transistorschalter implementiert. Zu jedem Transistor ist ein Bit gespeichert, über das gesteuert wird ob die Verbindung durchgeschaltet werden soll. Die Bits können mehrfach programmiert werden.

**2.35** Ein (2-von-3) Voter soll mittels eines PLA realisiert werden. Dieser (2-von-3) Voter hat drei Eingänge, die wir mit $A$, $B$ und $C$ bezeichnen, und einen Ausgang, den wir mit $R$ bezeichnen. An jedem dieser Eingänge kann entweder der Wert 0 oder der Wert 1 anliegen. Die Aufgabe eines (2-von-3) Voters is es nun, eine Mehrheitsentscheidung zu treffen und jenen Wert (entweder 0 oder 1) am Ausgang zu liefern, der am häufigsten unter den drei Eingängen auftritt. In der Praxis verwendet man einen (2-von-3) Voter in fehlertoleranten Systemen. Wenn ein Eingang einen falschen Wert liefert, können die beiden anderen das falsche Ergebnis überstimmen und der richtige Wert liegt am Ausgang an.

Bestimmen Sie zuerst die Wahrheitstabelle der logischen Funktion eines (2-von-3) Voters. Minimieren Sie dann diese Funktion und tragen Sie das Ergebnis in das PLA ein.




(S. 65)

| $A$ | $B$ | $C$ | $R$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

$$R = (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge B \wedge C)$$
$$= (A \vee B) \vee (A \vee C) \wedge (B \wedge C)$$

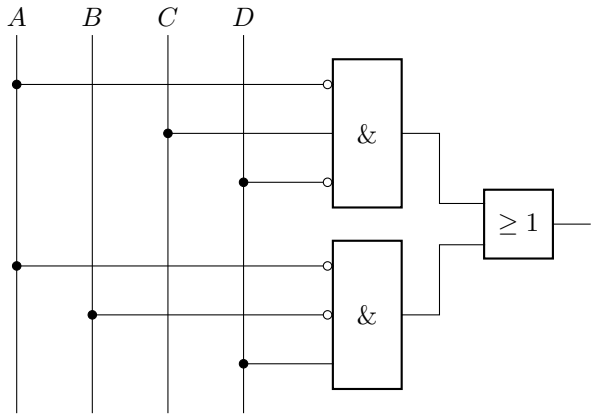
**2.36** Es soll eine Schaltung mittels PLA entworfen werden, die stets logisch 1 ausgibt, wenn die Zahl entweder durch 4 oder durch 6 teilbar ist. Die Zahl wird dargestellt durch $A$, $B$, $C$, und $D$ wobei $A$ das LSB und $D$ das MSB ist. (S. 65)

Wahrheitstabelle mit Ausgang $a$:

| # | $A$ | $B$ | $C$ | $D$ | $a$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 |
| 3 | 1 | 1 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 | 1 |
| 5 | 1 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 0 | 0 |
| 8 | 0 | 0 | 0 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 | 0 |
| 11 | 1 | 1 | 0 | 1 | 0 |
| 12 | 0 | 0 | 1 | 1 | 1 |
| 13 | 1 | 0 | 1 | 1 | 0 |
| 14 | 0 | 1 | 1 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 | 0 |

| | | $A$ | | $\overline{A}$ | |
|---|---|---|---|---|---|
| $B$ | 0 | 0 | 0 | 0 | $D$ |
| | 0 | 0 | 0 | 1 | $\overline{D}$ |
| $\overline{B}$ | 0 | 0 | 0 | 1 | |
| | 0 | 0 | 1 | 1 | $D$ |
| | $C$ | | $\overline{C}$ | $C$ | |

Die minimierte Ausgabefuntion lautet $a = (\overline{A} \wedge C \wedge \overline{D}) \vee (\overline{A} \wedge \overline{B} \wedge D)$. Die resultierende Schaltung:



# 3 VHDL

**3.37** Welche drei Sichtweisen gibt es beim Entwurf elektronischer Schaltungen laut dem Y-Modell? (S. 70)

1. Geometrie
2. Struktur
3. Verhalten

Die drei Sichten lassen sich in einem Y-Modell zusammenfügen.

**3.38** Welche 5 Entwurfsebenen gibt es in VHDL? (S. 70)

1. Systemebene
2. Algorithmische Ebene
3. Transistor-Transfer-Ebene
4. Logik-Ebene
5. Schaltkreisebene

**3.39** Welchen Baustein beschreibt der folgende VHDL-Code? Geben Sie die Funktionen aller Ein- und Ausgänge genau an!

```
library ieee;
use ieee.std_logic_1164.all;

entity mystery is
  port(C, SI : in std_logic;
       SO : out std_logic;
       PO : out std_logic_vector(7 downto 0));
end mystery;

architecture behav of mystery is
  signal tmp: std_logic_vector(7 downto 0);
  begin
    process (C)
      begin
        if (C'event and C='1') then
          for i in 0 to 6 loop
            tmp(i+1) <= tmp(i);
          end loop;
          tmp(0) <= SI;
        end if;
    end process;
    SO <= tmp(7);
    PO <= tmp;
end behav;
```

(S. 83)

Es handelt sich um ein Schieberegister mit Takteingang C und paralleler Ausgabe der am seriellen Eingang SI empfangenen Information an den Ausgängen P0-P7. Am zusätzlichen Ausgang SO wird die empfangene Information seriell um 7 Taktzyklen verzögert ausgegeben.

**3.40** Welche drei Arbeitsschritte sind zur Erstellung eines Moduls bzw. einer Komponente in VHDL erforderlich? (S. 73)

1. Schnittstellenbeschreibung
2. Beschreibung der Architektur
3. Festlegung der Konfiguration

# 4 Mikroprozessoren

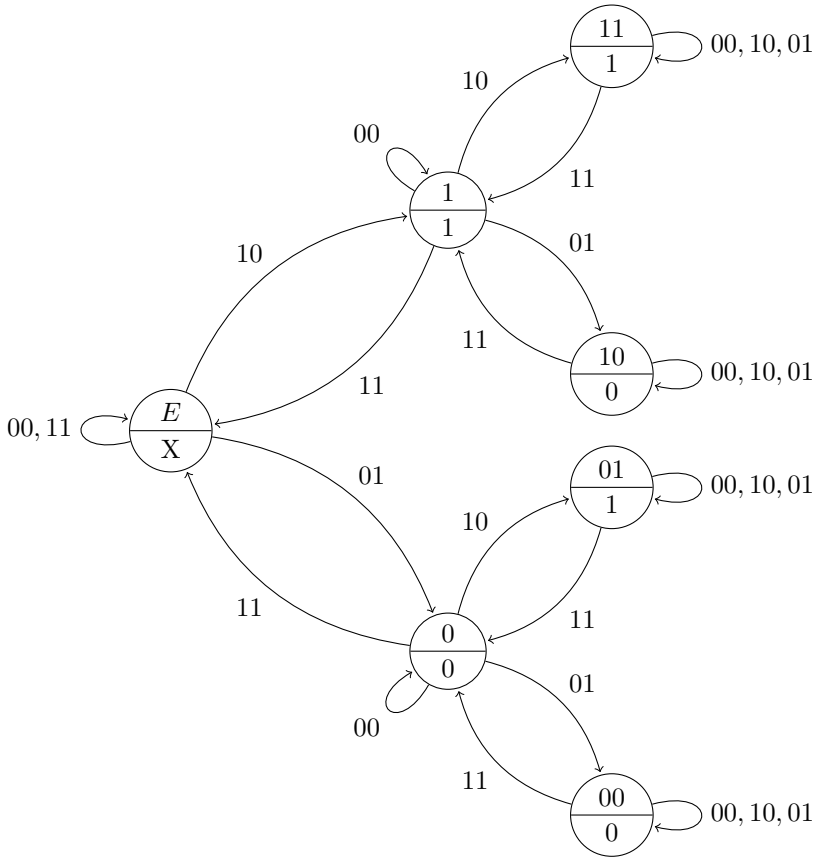
**4.41** Von welchen zwei Faktoren hängen Zustandsänderungen bei Schaltwerken ab? (S. 87)

Vom aktuellen Zustand und von der Eingangsinformation.

**4.42** Ein Hardware-Stack (Kellerspeicher) soll mittels eines Moore-Schaltwerks implementiert werden. Dieser Stack hat Platz für maximal zwei Elemente (ein sehr kleiner Stack) und kann Elemente vom Typ 0 oder vom Typ 1 aufnehmen. Es gibt zwei Operationen, die wir mit `push` und `pop` bezeichnen wollen. `push` legt ein Element oben auf den Kellerspeicher, `pop` entfernt das oberste Element. Dabei wird nach dem LIFO-Prinzip (last in, first out) gearbeitet, d.h. `pop` entfernt immer jenes Element, welches als letztes auf dem Kellerspeicher mit `push` abgelegt wurde.

Das Schaltwerk hat zwei Eingänge, die wir mit $push_0$ und $push_1$ bezeichnen. Wenn beide Eingänge 0 sind, verändert sich der Zustand des Stacks nicht. Wenn $push_0$ den Wert 1 hat, wird ein Element vom Typ 0 auf den Stack gelegt, wenn $push_1$ den Wert 1 hat, wird ein Element vom Typ 1 auf den Stack gelegt. Wenn sowohl $push_0$ als auch $push_1$ den Wert 1 haben, soll eine `pop`-Operation durchgeführt werden, d.h. es wird das oberste Element entfernt. Wenn der Stack voll ist (d.h. es sind bereits zwei Elemente gespeichert), bewirkt die `push`-Operation keine Änderung. Wenn der Stack leer ist, hat die `pop`-Operation keine Wirkung.

Am einzigen Ausgang der Schaltung, den wir *top* bezeichnen, wird das oberste Element des Stacks ausgegeben. Dabei soll eine 0 für ein Element vom Typ 0 und 1 für ein Element vom Typ 1 ausgegeben werden. Geben Sie den Zustandsgraphen und eine dichte Zustandscodierung für dieses Moore-Schaltwerk an (verwenden Sie dabei K als LSB). Es ist nicht notwendig, die Übergangsfunktion anzugeben, allerdings soll die Ausgangsfunktion berechnet werden.




(S. 99)

Bitmuster Reihenfolge: $push_0$, $push_1$. Der Zustände sind wie folgt definiert:

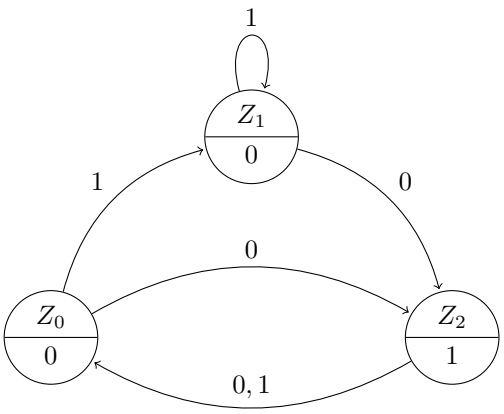
| Zustand | Stack | *top* |
|---|---|---|
| $E$ | leer | X |
| 0 | 0 | 0 |
| 1 | 1 | 1 |
| 00 | 00 | 0 |
| 01 | 01 | 1 |
| 10 | 10 | 0 |
| 11 | 11 | 1 |

Für sieben Zustände kommt man mit 3 Latches aus ($7 \leq 2^3$), die Zustandstabelle mit dichter Codierung:

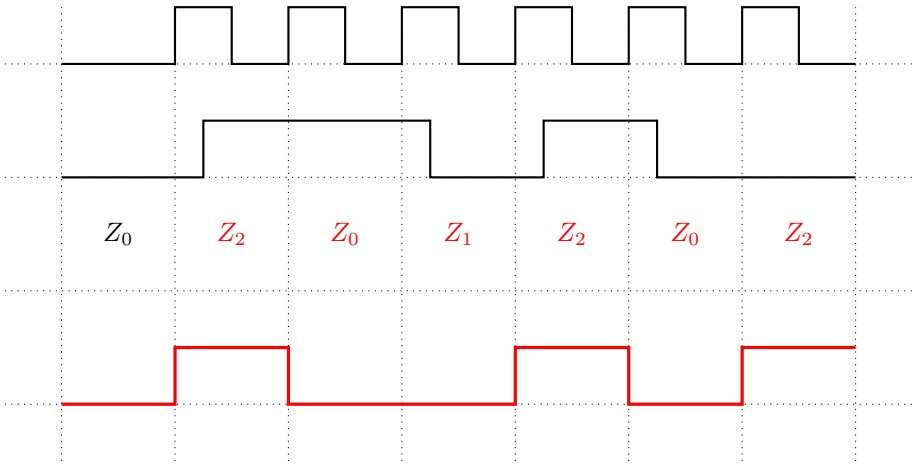
| Zustand | M | L | K |
|---|---|---|---|
| $E$ | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 00 | 1 | 0 | 0 |
| 01 | 0 | 1 | 1 |
| 10 | 1 | 1 | 0 |
| 11 | 1 | 0 | 1 |

Die minimierte Ausgabefunktion $A$ hängt nur noch von $K$ ab: $A = K$

**4.43** Gegeben ist das folgende Zustandsdiagramm eines Moore-Schaltwerkes mit drei Zuständen ($Z_0$ bis $Z_2$), einem Eingang $E$ und einem Ausgang $A$. Vervollständigen Sie den Zeitverlauf der Schaltung. Die Triggerung erfolgt durch die positiven Flanken des Taktsignals. Im Zustandsdiagramm des Mooreschaltwerks ist die (triviale) Bitreihenfolge für den Eingang $E$ und für den Ausgang $A$ eingezeichnet.




(S. 103)




**4.44** Nennen Sie drei Ausgänge zur Statusanzeige einer ALU. (S. 127)

1. Carry Flag, zeigt an ob bei einer Berechnung ein Übertrag stattgefunden hat
2. Zero Flag, ist logisch 1 wenn alle Bits im Ergebnis logisch 0 sind.
3. Sign Flag, ist logisch 1 falls die Zahl einen negativen Wert hat und 0 andernfalls.

**4.45** Was versteht man unter einer Bus Arbitration Logic (bzw. einem Busarbiter)? (S. 129)

Die Arbitration Logic entscheidet, welches Register seine Daten auf den Bus legen darf, um zu verhindern, dass sich mehrere Register gegenseitig beeinflussen.

**4.46** Was versteht man unter einem Scratchpad eines Prozessors? (S. 130)

Ein Scratchpad (auch Register File) bezeichnet Prozessorregister die als Zwischenspeicher verwendet werden.

**4.47** Der im Buch beschriebene Mikroprozessor Mikro16 enthält leider keinen eigenen Befehl, um einen beliebigen numerischen Wert (das heißt, ein immediate value) in ein Register zu laden. Geben Sie eine Sequenz von Instruktionen für den Mikro16 an, der den Wert 17 in das Register `R11` lädt. Sie können dazu beliebige, andere Register überschreiben. Bedenken Sie jedoch, dass die Register `R0`, `R1` und `R2` nicht überschrieben werden können, weil sie die Konstanten 0, 1 bzw. -1 enthalten. (S. 139)

```
1  R3  <- lsh(1 + 1);     # 1 + 1 = (10)_2; lsh((10)_2) = (100)_2 = (4)_10
2  R3  <- lsh(R3);        # lsh((100)_2) = (1000)_2 = (8)_10
3  R3  <- lsh(R3);        # lsh((1000)_2) = (10000)_2 = (16)_10
4  R4  <- 1;
5  R11 <- R3 + R4;        # 16 + 1 = 17
```

**4.48** Es soll ein Mikro-Programm für den Mikro16 Prozessor entworfen werden, das eine Bitfolge um $n > 0$ Stellen arithmetisch nach links verschiebt. Die zu shiftende Bitfolge liegt in Register `R4`. Die Anzahl auszuführender arithmetischer Shift-Operationen ist im Register `R5` abgelegt. Initial sind die Register `R0` mit der Konstante 0, `R1` mit der Konstante 1 und `R2` mit der Konstante -1 belegt. Zusätzlich enthält Register `R10` den Wert $(8000)_{16}$. (S. 140)

```
1  (R5); If Z goto 5
2  R4 <- lsh(R4)
3  R5 <- R5 + R2
4  goto 1
```

**4.49** Was ist VLSI, Vorteile davon gegenüber anderen Chips (S. 142)

VLSI bedeutet Very Large Scale Integration, alle Bauteile und Leitungen werden gleichzeitig auf einem Siliziumplättchen hergestellt. Der Vorteil ist, dass man sehr viele Bauteile auf einem Chip integrieren kann. Durch die Reduzierung der Wegstrecken kann sich Strom schneller ausbreiten was eine Steigerung der Geschwindigkeit zur folge hat.

**4.50** Das in der Informatik bekannte Moore's Law sagt voraus, dass die Dichte der Transistoren auf einem integrierten Schaltkreis exponentiell wächst. Obwohl diese Vorhersage schon 1965 getätigt worden ist, trifft sie immer noch zu. Welchen Vorteil hat eine Erhöhung der Bauteildichte bei Mikroprozessoren? (S. 142)

Eine höhere Dichte führt zur Reduzierung der Wegstrecken. Da sich der Strom mit nur etwa der 0,7-fachen Lichtgeschwindigkeit in den Schaltungsleitungen ausbreitet bedeutet das eine Steigerung der Prozessorgeschwindigkeit.

# 5 Computersysteme

**5.51** Was versteht man unter dem Program Counter (PC) bei einem (Mikro-)Prozessor? (S. 144)

Ein Prozessorregister, welches die Adresse des auszuführenden Maschinenbefehls speichert. Nachdem ein Maschinenbefehl abgearbeitet wurde, erhöht der Interpreter den Program-Counter und liest den nächsten Befehl ein.

**5.52** Das Instruktionsformat der x86-Prozessorreihe von Intel (der auch die neuesten Pentium-Prozessoren angehören) verwendet Maschineninstruktionen mit variabler Länge. Eine x86-Maschineninstruktion kann zwischen einem und 15 Bytes lang sein. Nennen Sie einen Vorteil und einen Nachteil von Instruktionen mit variabler Länge im Vergleich zu Instruktionen mit fixer Länge. (S. 144)

Vorteil: Der Maschinencode kann mit weniger Codewörtern codiert werden. Nachteil: Der Interpreter ist umfangreicher, komplexer und dadurch langsamer.

**5.53** Welche zwei Möglichkeiten gibt es, die Adressen der Ports für Input-/Output-Operationen zu vergeben? (S. 145)

1. Independent I/O: Ports haben vom Hauptspeicher unabhängige Adressen und erfordern eigene I/O-Transferoperationen.
2. Memory-mapped I/O: Ein Teil des Adressraums wird für Ports verwendet, somit können Ports mit normalen Transferoperationen angesprochen werden.

**5.54** Welche zwei wichtigen Schritte werden beim Aufruf einer Unterprozedur durch einen Subroutine Call verrichtet? (S. 149)

1. Retten des Program Status Words, der Registerinhalte und des Program Counters.
2. Laden des PC mit der Prozedur-Startadresse.

**5.55** Was sind Interrupts (S. 150)

Interrupts unterbrechen den aktuellen Programmablauf und springen in bestimmte Interrupt-Service-Routinen. Auslöser für einen Interrupt sind Ausnahmefehler oder ein externes Ereignis. Interrupts treten asynchron zum Programmablauf ein. Beim Auftreten eines Interrupts über eine spezielle Hardware (Interrupt-Logik) die CPU.

**5.56** Was ist nach Abarbeitung eines Interrupts zu tun um den normalen Programmablauf wieder herzustellen? (S. 150)

Wiederherstellen des Contexts, d.h. alle Prozessorregister, den PC und das Statusregister.

**5.57** Welche zwei Möglichkeiten gibt es, die Adresse einer Interrupt Service Routine (ISR) zu bestimmen? (S. 150)

1. Fixe Zuordnung: Jeder Interrupt-Quelle ist eine fixe ISR-Adresse zugeordnet.
2. Interruptvektor: Eine modifizierbare Interrupt-Tabelle gibt die Zuordnung der Interrupt-Quellen zu den ISR-Adressen an.

**5.58** Der Stackpointer (SP) zeigt bei der Ausführung eines Programms auf die Adresse `0xFFA8`. Das Programm führt daraufhin zwei `push`-Befehle durch, die jeweils ein 16-Bit-Wort auf den Stack legen. Auf welche Adresse zeigt der Stackpointer nach diesen Operationen? (S. 152)

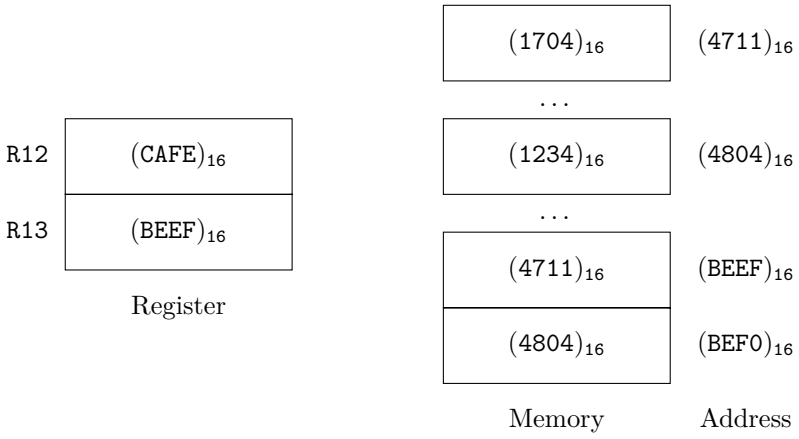
`0xFFA8 - 0x2 = 0xFFA6`

**5.59** Wo werden die Return-Adressen bei Unterprozeduraufrufen gespeichert? (S. 153)

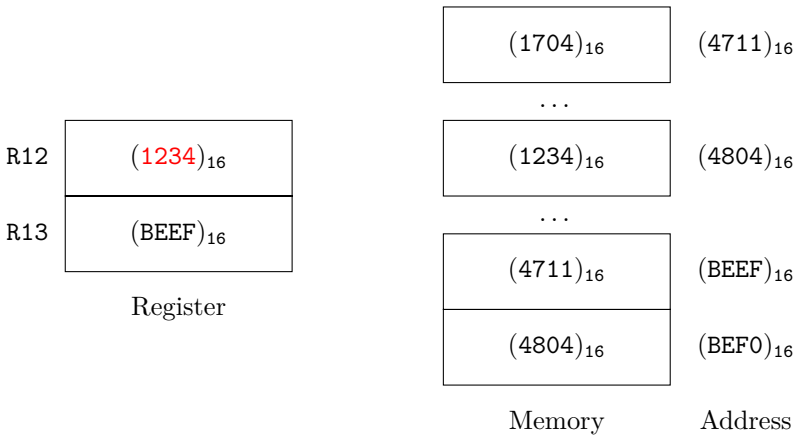
Auf dem Stack.

**5.60** Geben Sie die Werte in allen gezeigten Registern und Speicherzellen im Endzustand an, die sich aus dem Ausgangszustand nach dem Abarbeiten der folgenden Instruktion $I$ ergeben.

Instruktion $I$ (indirect addressing): `R12` $\leftarrow$ `memory[memory[R13]` $+$ `(F3)`$_{16}$`]`




(S. 156)




**5.61** Was ist indirekte Adressierung, geben Sie ein Beispiel. (S. 158)

Bei indirekter Adressierung erfolgt die Speicheradressierung in zwei Stufen. Die erste Berechnung liefert die Adresse eines Speicherwortes das entweder die Adresse oder einen Offset der folgenden Berechnung enthält. Beispiel: `R7 <- memory[memory[(1234)16]]`

**5.62** Durch welche drei im Buch erläuterten Arten der Parallelverarbeitung kann die Performance eines Prozessors gesteigert werden? (S. 160)

1. Vektorverarbeitung: Eine Operation wird an mehreren Operanden gleichzeitig durchgeführt.
2. Superskalare Verarbeitung: Durch mehrfach vorhandene Verarbeitungseinheiten können Befehle parallel abgearbeitet werden.
3. Instruction Pipelining: Eine Instruktion wird in etwa gleich lange Schritte zerlegt welche jeweils von einer Stufe abgearbeitet werden. Die Stufen arbeiten parallel.

**5.63** Warum ist ein hohe Trefferquote der Branch-Prediction Einheit (verantwortlich für die Vorhersage des Ziels von bedingten Sprüngen - Predicted Branches) bei modernen Prozessoren so wichtig? Denken Sie daran, dass die Prozessoren ein sehr lange Pipeline (oft mehr als 20 Stufen) verwenden. (S. 164)

Um zu vermeiden, dass die Pipeline nach einem bedingten Sprung für ungültig erklärt und neu geladen werden muss, was zu Performanceverlusten führt.

**5.64** Ein 256 MB-Speicherbauteil hat eine durchschnittliche Zugriffszeit von 12 ns. Um den Zugriff zu beschleunigen, wird ein Cache von 4 MB verwendet, der eine Zugriffszeit von durchschnittlich nur 4 ns hat. Ein Programm zur Bestimmung von Primzahlen weist eine exzellente Lokalität der Speicherzugriffe auf, sodass eine Hit-Rate von 97% erzielt werden kann. Wie groß ist die effektive Speicherzugriffszeit bei der Abarbeitung dieses Programms? (S. 169)

$$t_{eff} = h \cdot t_{cache} + (1 - h) \cdot t_{main} = 0,97 \cdot 4 + 0,03 \cdot 12 = 4,24$$

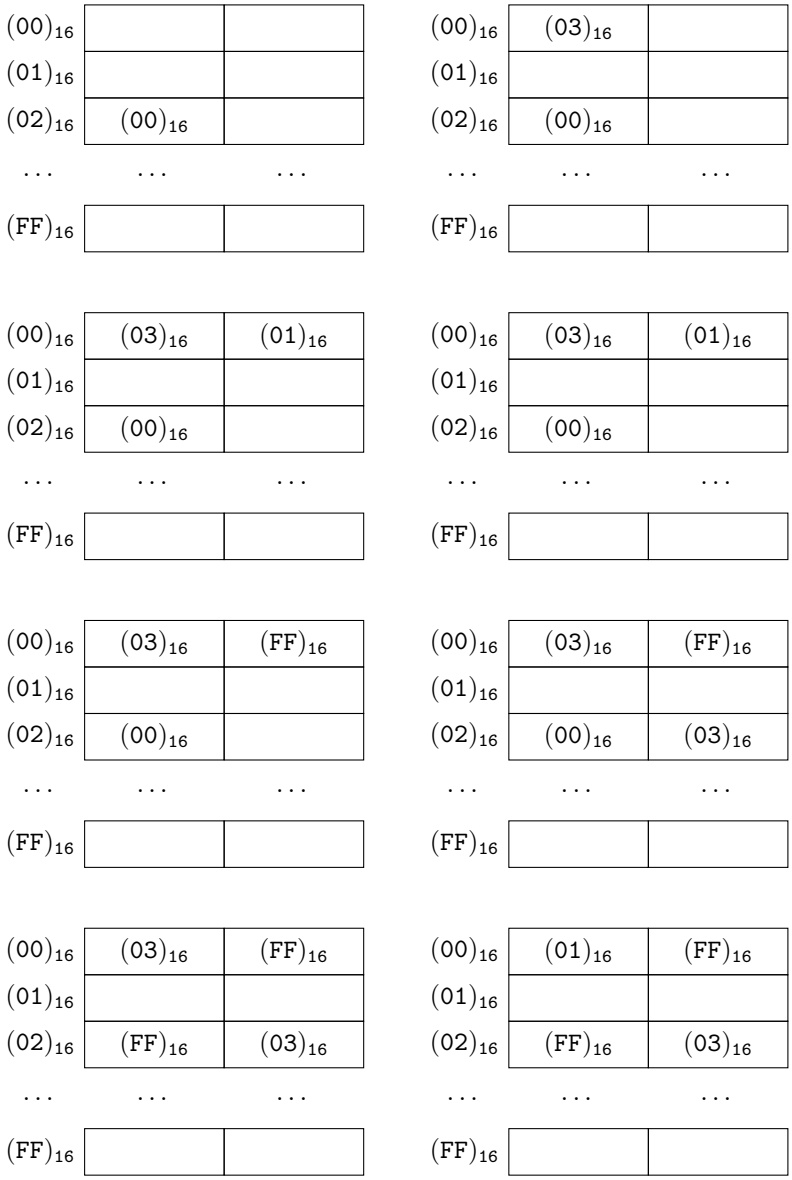
**5.65** Was ist ein Cache? (Hauptspeicher, Cache, Register in richtiger Reihenfolge anordnen.) (S. 169)

Ein Cache ist ein sehr schneller aber teurer Speicher (SRAM) der Zugriffe von der CPU auf den langsameren Hauptspeicher (DRAM) zwischenspeichert. Die Hierarchie von langsam bis schnell:

1. Hauptspeicher
2. Cache
3. Prozessorregister

**5.66** Gegeben ist ein assoziativer Zweiweg-Cache mit LRU-Verfahren. Geben Sie an, welche Werte sich im 16-Bit-Tag-RAM befinden, wenn der Cache anfangs leer ist und die folgende Reihe von (32-Bit) Speicherzugriffen erfolgt. Beachten Sie, dass uns der Datenbereich des Caches nicht interessiert und er daher auch nicht eingezeichnet werden soll.

Speicherzugriffe (in folgender Reihenfolge): $(0002)_{16}$, $(0300)_{16}$, $(0100)_{16}$, $(0300)_{16}$, $(FF00)_{16}$, $(0302)_{16}$, $(FF02)_{16}$, $(0100)_{16}$. (S. 171)




**5.67** Erklären Sie DMA. (S. 174)

Bei Direct Memory Access (DMA) findet eine Datetübertragung ohne CPU direkt zwischen Hauptspeicher und I/O-Geräten über den Bus statt. Für die Steuerung der Übertragung ist der DMA Controller zuständing, er fordert den Bus an und gibt ihn wieder frei.

**5.68** Erklären Sie Cycle-Stealing. (S. 174)

Bei Direct Memory Access (DMA) findet eine Datetübertragung ohne CPU direkt zwischen Hauptspeicher und I/O-Geräten über den Bus statt. Der DMA Controller fordert den Bus an

und beginnt mit der Übertragung. Bei der Cycle-Stealing Variante kann der Prozessor erst wieder auf den Bus zugreifen, wenn diese Aktion beendet ist.

**5.69** Durch welche Parameter lassen sich Magnetpeicher charakterisieren? (S. 180)

1. Kapazität
2. Zugriffszeit
3. Datenrate

**5.70** Was ist ein Head-Crash? (S. 182)

Wenn ein Schreib-Lesekopf in einer Festplatte eine rotierende Platte berührt.

**5.71** Was ist die Latenzzeit und wie berechnet man die mittlere Zugriffszeit bei einer Festplatte? (S. 183)

Die Latenzzeit ist die Zeit die abgewartet werden muss bis der gewünschte Sektor am Arm vorbei rotiert und dauert maximal so lange wie eine Plattenumdrehung. Mittlere Zugriffszeit = Positionierzeit des Armes + mittlere Latenzzeit.

**5.72** Was sind Bad-Blocks? (S. 183)

Stellen auf der Festplatte die nicht mehr einwandfrei beschreibbar sind. Diese können softwaremäßig markiert werden um sie vor Zugriffen zu schützen.

**5.73** Was ist der technische Unterschied zwischen einer DVD und einer CD? (S. 185)

Bei einer DVD liegen zwei Schichten vor, die der Laser durch Änderung der Brennweite auswählen kann. Weiters besitzt die DVD pro Schicht eine höhere Kapazität als die CD. Das wird durch eine geringeren Abstand zwischen den Spuren und eine reduzierte Mindestlänge der Pits erreicht.

**5.74** Was sind Dialoggeräte? (S. 191)

Dazu gehören Geräte zur Kommunikation zwischen Mensch und Maschine. Z.B. Tastatatur, Maus, Bildschirm, Touchscreen, Scanner, Joystick, Drucker, ...

**5.75** USB: Wie viele Hubs können zwischen einem PC und dem Gerät hängen (S. 194)

Bis zu sieben.

# Netzwerke

## 6 Aufbau

**6.76** Wofür stehen die Abkürzungen ANSI, BSI, DIN, AFNOR? (S. 211)

ANSI: American National Standards Institute

BSI: Britisch Standards Institute

DIN: Deutsches Institut für Normung e.V.

AFNOR: Association Français de Normalisation

**6.77** Wofür stehen die Abkürzungen ISO, IEEE, NBS, IEC, ECMA und IFIP? (S. 211)

ISO: International Organization for Standartization

IEEE: Institute of Electrical and Electoronics Engineers

NBS: National Bureau of Standards

IEC: International Electronical Commission

ECMA: European Computer Manufacturers Association

IFIP: International Federation for Information Processing

# 7 Architekturen

**7.78** Wozu dient das OSI-Schichtenmodell? (S. 215)

Es legt die Aufgaben der einzelnen theoretischen Layer fest.

**7.79** ISO-OSI Diagramm ergänzen (Verbindungen einzeichnen, Layer benennenm beispiele für layer geben) (S. 215)

**Layer 1 – Physical Layer:**

1. Kümmert sich um die Übertragung einzelner Bits.
2. Das L1-Protokoll umfasst die Konventionen und Regeln, nach denen einzelne Bits zu übertragen sind.
3. Die angebotenen Services bieten die Möglichkeit, einen Strom von Bits über das jeweilige physikalische Medium zu senden bzw. zu empfangen.
4. Bittaktgeneration wird mittels eines Amplitudenentscheiders bewirkt, Wortsynchronisation erfolgt mittels eines Synchronisationswortes.
5. Beispiele: Übertragungsmedien: Funk, Kupfer, Lichtleiter; Geräte: Netzwerkkarte, Hub

**Layer 2 – Data Link Layer:**

1. Gewährleistet fehlerfreie Datenübertragung und regelt den Zugriff auf das Übertragungsmedium.
2. Vom Network-Layer kommenden Daten werden in Stücke portitioniert und mit einem Header und einem Trailer zu einem Frame verarbeitet. Der Header enthält u.a. die Adresse des Empfängers, der Trailer eine CRC-Prüfsumme.
3. Frames werden an einen IMP gesendet, der den Empfang mit einem Acknowledgement Frame bestätigt.
4. Das L2-Protokoll umfasst Konventionen und Regeln, nach denen der Austausch von Frames erfolgt.
5. In Broadcast Subnets kümmert sich der MAC Sublayer um die Zuteilung des Übertragungsmediums.

6. Error Control dient der Erkennung und Korrektur von Fehlern die während der Übertragung auftreten.
7. Flow Control ermöglicht dem Empfänger zu bestimmen, mit welcher Geschwindigkeit die Gegenseite senden darf.
8. Die angebotenen Services bieten also neben minderwertiger Datenübertragung vor allem Möglichkeiten zur fehlerfreien Datenübermittlung mit garantierter Ankunft.
9. Beispiele: Hardware: Bridge, Switch

**Layer 3 – Network Layer:**

1. Zurständig für den Betrieb des Communication Subnets. Bietet die Möglichkeit echter End-zu-End-Verbindungen (Network Connections) zwischen Hosts, die nicht direkt miteinander verbunden sein müssen.
2. Vom Transport Layer kommende Daten werden mit einem Header und einem Trailer zu Paketen verarbeitet. Der Header enthält u.a. die Adresse des Empfängers, der Trailer eine Prüfsumme.
3. Die Schnittstellen zum Transport Layer heißen Network Service Access Points (NSAPs). Ein NSAP ist netzwerkweit durch eine eindeutige Adresse gekennzeichnet. Eine Network Connection ist eine Verbindung zwischen zwei NSAPs.
4. Die Hauptaufgabe ist das Routing, es bietet dafür connection-oriented und connectionless Services an.
   (a) Bei connection-oriented Services wird vor dem Datenaustausch eine dedizierte Verbindung zwischen zwei NSAPs hergestellt und nach der Übertragung wieder explizit aufgelöst.
   (b) Bei connection-less Services werden Pakete unabhängig voneinander durch das Netzwerk geschleust. Die Reihenfolge beim Empfänger kann von der der Sendereihenfolge verschieden sein.
5. Für das Routing existieren verschiedene Service-Quallitäten betreffend der Zuverlässigkeit der Übermittlung und der Garantie, dass Daten ankommen.
6. Beispiele: Protokolle: IP, ICMP; Hardware: Bridge, Switch

**Layer 4 – Transport Layer:**

1. Bietet den höheren Layern einen einheitlichen Zugriff auf den Network Layer.
2. Die Layer 1-3 sind für einen Benutzer unbeeinflussbar. Der Transport Layer reduziert die Abhängigkeit zu diesen Layern durch Bereitstellen von Methoden zur sicheren Übertragung von Transport-Paketen.
3. Schnittstellen zum Session Layer werden Transport Service Access Points (TSAP) genannt und werden netzwerkweit durch eindeutige Adressen identifiziert. Einer Transport Connection verbindet zwei TSAPs.
4. Eine wesentliche Aufgabe des Transport Layers ist die Bereitstellung eines einheitlichen Schemas für TSAP Adressen. Aus einer TSAP Adresse muss eine NSAP Adresse hervor gehen.
5. Dieser Layer bietet bequeme connection-oriented und connection-less Services an. Z.B. kann eine Transport Connection intern auf mehrere Network Connections aufgeteilt, oder eine Network Connection für mehrere Transport Connections genutzt werden.

6. Beispiele: Protokolle: TCP, UDP, SCTP; Hardware: Router

**Layer 5 – Session Layer:**

1. Zuständig für die Kommunikation zwischen Prozessen auf verschiedenen Hosts.
2. Bietet im Prinzip erweiterte Transpor Layer Services, unter anderem Atomic Actions für Messages.

**Layer 6 – Presentation Layer:**

1. Befasst sich mit der Syntax und der Semantik der übertragenen Information.
2. Dieser Layer konvertiert zu übertragende Daten in Formate die auch andere Hosts richtig interpretieren können.
3. Beispiele: Latin1 zu UTF8, Einer- zu Zweirkomplementdarstellung.

**Layer 7 – Application Layer:**

1. Enthält die eigentliche Applikation.
2. Beispiele: E-Mail, SSH, Telnet, FTP.

**7.80** Layer 1 benennen und zwei Aufgaben nennen. (S. 215)

Der Physical Layer bietet Services zum Übertragen einzelner Bits. Aufgaben: Bittaktgeneration, Wortsynchronisation.

**7.81** Datalink-Layer: Hauptaufgabe erklären und drei Mechanismen nennen, die er dazu nützt (S. 215)

Gewährleistet fehlerfreie Übertraung und regelt Zugriff auf das Übertragungsmedium. Zur fehlerfreien Übertragung werden Prüfsummen im Frame mitgeschickt, mit denen der Empfänger Fehler erkennen kann. Ein Empfänger kann mit einem Acknowledement-Frame den Empfang bestätigen. Der Zugriff auf das Übertragungsmedium wird vom MAC Sublayer geregelt.

**7.82** In welcher OSI-Schicht ist IP angesiedelt? (S. 215)

IP, das Internet Protokoll, ist im Layer 3 (Network Layer) angesiedelt.

**7.83** Welche Aufgabe hat Schicht 1 im OSI-Schichtenmodell? (S. 215)

Der Physical Layer beschreibt die Konventionen und Regeln, nach denen einzelne Bits zu übertragen sind. Die angebotenen Services bieten die Möglichkeit, einen Strom von Bits über das jeweilige physikalische Medium zu senden bzw. zu empfangen. Zu den Aufgaben zählen auch Bittaktgeneration und Wortsynchronisation.

**7.84** Nennen Sie die Hauptaufgabe des Layer 3 (Network Layer) im OSI-Referenzmodell. Geben Sie das vorherrschende Network Layer Protokoll im Internet an. (S. 216)

Die Hauptaufgabe ist das Routing, d.h. das Festlegen über welche IMPs ein Paket zu seinem Ziel geschickt wird. Das vorherrschende Protokoll ist das IP-Protokoll.

**7.85** Durch welche Maßnahme wird bei Token Ring ein unidirektionaler Datenverkehr bewirkt? (S. 222)

Beim Token Ring werden die Hosts durch eine Kette derart verbunden, dass eine Ring-Topologie entsteht. Das Ring-Interface einer Maschine ist im wesentlichen ein D-Latch, wodurch der Ring als verteiltes zyklisches Schieberegister angesehen werden kann.

**7.86** Wie wird eine kollisionsfreie Datenübertragung beim Token Ring erreicht? (S. 222)

Dadurch, dass nur der Host der gerade das Token hat Daten senden kann.

**7.87** Wie wird ein Token-Loss behandelt? (S. 222)

Durch Überwachung des Netzwerkes, z.B. durch einen einzelnen Host.

**7.88** Erläutern Sie die Grundidee von DSL. (S. 225)

Die Grundidee von DSL (Digital Subscriber Line) ist die Nutzung des bestehenden Telefonnetzes zum Senden und Empfangen digitaler Daten im ungenutzten Frequenzspektrum, da nur ein schmaler Frequenzbereich für Telefonie benötigt wird.

**7.89** Was sind die Unterschiede zwischen ADSL und SDSL? (S. 227)

ADSL bedeutet Asynchronous DSL und SDSL Synchronous DSL. Bei ADSL sind Up- und Downloadraten unterschiedlich, bei SDSL sind sie gleich.

**7.90** Was versteht man unter Frequency Hopping? (S. 226)

Wenn bei einer Übertragung sehr schnell zwischen verschiedenen Frequenzen innerhalb eines Frequenzbereiches hin und her gesprungen wird. Es senkt die Wahrscheinlichkeit, dass die Übertragung gestört wird.

# 8 Protokolle

**8.91** Sie wollen von zu Hause 5 Rechner an das Internet anschließen. Ihr Internet Provider stellt aber nur eine Class B Adresse zur Verfügung. Geben Sie eine Netzwerktopologie an, sodass jeder Rechner mit einem Server im Internet kommunizieren kann. (S. 234)

1. Ein Router mit zwei Adaptern stellt ein NAT Service zur Verfügung. Dem ersten Adapter wird die Class B Adresse vom Provider zugewiesen, dem zweiten Adapter die private Adresse 192.168.0.1.

2. Die Hosts bekommen Adressen aus dem Bereich 192.168.0.* zugewiesen wobei aber 192.168.0.0 für das Subnetz, 192.168.0.1 für den Gateway und 192.168.0.255 für Broadcast reserviert sind. Die Netzwerkmaske wird auf 255.255.255.0 gesetzt, und als default Gateway wird die Adresse des Routers eingetragen.

**8.92** Gegeben ist das Klasse C Netzwerk mit der Adresse 192.168.11.0. Teilen Sie dieses Klasse C Netzwerk so auf, dass

(a) Subnetze für jeweils zumindest 104 Hosts entstehen.

(b) Subnetze für jeweils zumindest 24 Hosts entstehen.

Geben Sie die jeweils maximal mögliche Anzahl der Subnetze und die jeweiligen Subnetzmasken an. (S. 235)

(a) max. Anzahl Subnetze mit jeweils 104 Hosts: $2^1 = 2$
Subnetzmaske: 255.255.255.128

(b) max. Anzahl Subnetze mit jeweils 104 Hosts: $2^3 = 8$
Subnetzmaske: 255.255.255.224

**8.93** Wieviele Adressen sind bei einem Netzwerk mit der Subnetmask 255.255.255.192 verwendbar? Was ist 192.168.0.63 in diesem Netzwerk? Wer bearbeitet eine Anfrage auf die IP Adresse 192.168.0.193? (S. 235)

Es sind $2^6 - 2 = 62$ Adressen für Hosts verwendbar, eine Adresse ist für das Netzwerk und eine für Broadcast reserviert. 192.168.0.63 ist die erste mögliche Hostadresse in diesem Subnetz. Die IP 192.158.193 ist eine Broadcastadresse und wird von allen Hosts in diesem Subnetz bearbeitet.

**8.94** Was ist der Unterschied zwischen TCP und UDP? (S. 236)

Das Transmission Control Protocol ermöglicht eine Punkt-zu-Punkt Verbindung zwischen Hosts. Das Protokoll besitzt eine Fluss- und Fehlerkontrolle und garantiert, dass versendete Daten in der richtigen Reihenfolge ankommen. Das User Datagram Protocol ist das Gegenstück und erlaubt auch Broadcasting und Multicasting. UDP garantiert nicht, dass Datenpakete beim Empfänger in der richtigen Reihenfolge, fehlerfrei oder überhaupt ankommen, es ist dafür aber schneller.

# Betriebssysteme

## 9 Betriebssysteme Übersicht

**9.95** Was versteht man unter einem System Call? (S. 245)

System Calls stellen die Schnittstelle zwischen Programmen und der Funktionalität des Betriebssystem dar. Einem System Call ist eine Prozedur zugeordnet welche vom Betriebssystem gestartet wird. Die System Calls sind in Form von Funktionen durch ein API definiert.

**9.96** Welche Dienste bietet ein Betriebssystem an? (S. 244)

1. Prozessmanagement
2. Interprozess-Kommunikation
3. Speichermanagement
4. Zugriff auf E-/A-Geräte
5. Zugriff auf Dateien
6. Fehlerbehandlung
7. Accounting

**9.97** Was ist ein Trap? Geben Sie eine Beispiel. (S. 245,332)

Ein Trap ist ein Software-Interrupt-Befehl bei dem der aktuelle Prozess unterbrochen, dessen Kontext gesichert, und das Programm dessen Adresse im Trap-Vektor eingetragen wurde ausführt. Beispiel: System-Call.

# 10 Prozesse

**10.98** Was versteht man unter logischer Parallelität von Prozessoren? (S. 252)

Wenn mehrere Prozesse scheinbar parallel auf einem Prozessor ausgeführt werden. Die scheinbare parallelität ergibt sich dadurch, dass jedem Prozess der Prozessor für ein bestimmtes Zeitfenster zugeteilt wird.

**10.99** Prozesszustände

1. Welches potentielle Problem gibt es beim direkten Übergang von `BLOCKED` nach `RUNNING`?
2. Unterschied zwischen `BLOCKED` und `SUSPENDED`?

(S. 257)

1. Beim gerade ausgeführten Prozess kann es zu Datenverlusten kommen.
2. Bei `BLOCKED` wartet der Prozess von sich aus auf das Eintreten eines Ereignisses, bei `SUSPENDED` hingegen wird der Prozess von einem anderen Prozess eingefroren.

**10.100** Nennen Sie zumindest drei Bestandteile des Prozessdeskriptors! (S. 260)

1. Process Identification: Process-ID (PID)
2. Process State Information: Zustand (`RUNNING`, `READY`, ...), Priorität und Registerinhalte
3. Process Control Information: Besitzer, Zugriffsrechte, File Handles, ...

**10.101** Wenn der Parent-Prozess in einem Unix-Betriebssystem auf das Terminieren seines Child-Prozesses wartet (mittels des `wait()` System-Calls), dann wird der Prozess so lange blockiert, bis der Child-Prozess einen `exit()` System-Call macht. Dabei wird das Ergebnis des Child-Prozesses (sein Rückgabewert) an den wartenden Parent-Prozess übermittelt. Was passiert nun mit einem Child-Prozess, der terminiert und sein Ergebnis an den Parent-Prozess übermitteln will, wenn der Parent-Prozess nicht vorher ein `wait()` aufgerufen hat? (S. 261)

Der Child-Prozess geht in Zustand (DEAD bzw. ZOMBIE) bis der Parent-Prozess mittels `wait()` den Rückgabewert übernimmt.

**10.102** Welche zwei Komponenten benötigt jeder Thread für sich alleine? (S. 263-265)

1. Context: Registersatz der Prozessoren
2. Stack: für lokale Variablen und Rücksprungadressen

**10.103** Was sind Threads und was sind typische Eigenschaften von Threads? (S. 263)

Ein Thread ist Teil eines Prozesses der ausgeführt wird, ein Prozess kann mehrere Threads ausführen. Die Threads eines Prozesses teilen sich das Codesegement, das Datensegment und andere Ressourcen wie Dateideskriptoren. Jeder Thread besitzt einen eigenen Stack und seinen eigenen Context. Die Erzeugung eines Threads erfordert einen viel geringeren Overhead als die Erzeugung eines neuen Prozesses.

**10.104** Nennen Sie mindestens zwei Probleme bei der Verwendung von Threads! (S. 265)

1. Wegen Sharing des gleichen Adressraums gibt es keine Schutzmechanismen zwischen Threads.
2. Die Verwendung von Bibliotheken die nicht reentrant sind, insbesondere Probleme bei parallelem Zugriff auf globale Variablen.

**10.105** Gegeben ist ein Programm $P$, welches von zwei Threads $A$ und $B$ ausgeführt wird. Das Programm ist in der linken Box angegeben.

```
 1  int j;
 2
 3  void f(int i)
 4  {
 5          int k;
 6          k = i + j;
 7          j = k;
 8          return;
 9  }
10
11  int main()
12  {
13          int i;
14
15          i = read_int();
16          j = read_int();
17          f(i);
18          return 0;
19  }
```

```
Thread A:
15 (read_int() liefert 1)
16 (read_int() liefert 42)

Thread B:
15 (read_int() liefert 2)
16 (read_int() liefert 10)
17
6

Thread A:
17
6

Thread B:
7
8
18
ENDE (Wert von j angegeben)

Thread A:
7
8
18
ENDE (Wert von j angegeben)
```

Die Ausführungsreihenfolge der Threads ist in der rechten Box angegeben. Es sind dabei jeweils die Zeilennummern der Instruktionen angegeben, die ein Thread ausführt. Es beginnt Thread $A$ mit der Instruktion auf Zeile 15. Dabei ist für jeden Aufruf der Funktion `read_init()` angegeben, welchen Wert die Funktion zurück liefert. Nachdem Thread $A$ zwei Instruktionen ausgeführt hat, schaltet der Scheduler zu Thread $B$ um, der dann vier Instruktionen ausführt, usw.

Geben Sie an, welchen Wert die Variable $j$ hat, (a) nachdem Thread $B$ die Ausführung beendet hat und (b) nachdem auch Thread $A$ die Ausführung beendet hat. (S. 265)

(a) $j_B = 12$

(b) $j_A = 11$

**10.106** Wann wird ein Programm (oder eine Funktion) reentrant genannt? (S. 266)

Wenn das Programm (oder die Funktion) für die gleichzeitige Benutzung im selben Adressraum geeignet ist.

**10.107** Welche Anforderungen werden an einen Scheduler gestellt? (S. 268)

1. Fairness: Verteilung soll gerecht sein
2. Effizienz: optimale Auslastung
3. Durchsatz: Anzahl der verarbeiteten Jobs soll maximal sein
4. Response: möglichst schnelle Antwortzeiten
5. Prozessorwechselzeit: Berechnungsdauer der Scheduling-Entscheidung soll minimal sein

**10.108** Was ist Round Robin Scheduling?(S. 269)

Ein Scheduling-Verfahren das allen Prozessen nacheinander maximal einen Zeitschlitz lang Zugriff auf Ressourcen gewährt.

**10.109** Mit welchem Scheduling verfahren wird RR ergänzt? (S. 269)

Mit First Come First Served (FCFS). Bei FCFC wird der erste Prozess aus der `READY` Liste gewählt.

**10.110** Was passiert wenn man eine sehr kleine Zeitscheibe für die Prozesse verwendet (S. 269)

Der Scheduling-Overhead reduziert die Prozessorleistung.

**10.111** Zustandsgraph vom Scheduling vervollständigen (S. 270)

Knoten: `RUNNING`, `DEAD`, `READY[]`, `BLOCKED[]`

Kanten: `RUNNING` → `RUNNING`: Zeitscheibe abgelaufen, Prozess darf weitermachen
`RUNNING` → `DEAD`: Prozess terminiert
`RUNNING` → `READY[]`: Zeitscheibe abgelaufen, Prozess am Ende der Liste einreihen, erste Prozess in der Liste wird `RUNNIG` und aus der Liste entfernt
`RUNNING` → `BLOCKED[]`: Prozess blockiert, Prozess am Ende der Liste einreihen

**10.112** Ein Betriebssystem verwendet den Shortest Remaining Time (SRT) Algorithmus für das Prozess-Scheduling. Die folgende Tabelle zeigt den Zeitpunkt, zu dem die jeweiligen Prozesse in den Zustand `READY` wechseln und deren verbleibende Restlaufzeit. Geben Sie an, in welcher Reihenfolge die Prozesse ausgeführt werden. Dazu tragen Sie in die Tabelle ein, zu welchen Zeitpunkten ein Prozess in den Zustand `RUNNING` übergeht (den Zeitpunkt *und* den dazugehören Prozess in die Tabellen eintragen).

Dabei nehmen Sie einmal an, dass das Betriebssystem nicht-preemptives Scheduling verwendet, und danach, dass preemptives Scheduling unterstützt wird.

| Prozess | `READY`-Zeitpunkt | Restlaufzeit |
|---|---|---|
| A | 0 | 2 |
| B | 3 | 7 |
| C | 4 | 3 |
| D | 5 | 1 |
| E | 8 | 5 |
| F | 12 | 4 |

(S. 271)

Nicht-preemtiv: A:0, B:3, D:10, C:11, F:14, E:18

Preemptives: A:0, B:3, C:4, D:5, C:6, E:8, F:13, B:17

# 11 Interprozess-Kommunikation

**11.113** Gegeben sind drei Versuche zur Synchronisation von zwei Prozessen $A$ und $B$. Prozess-Synchronisation heißt in diesem Fall, die Codestücke müssen sicherstellen, dass sich höchstens ein Prozess (entweder $A$ oder $B$) zur selben Zeit im kritischen Abschnitt `critical_section()` befindet. Zusätzlich darf auch kein Deadlock auftreten. Dabei sind `turn`, `lock`, `lockA`, und `lockB` gemeinsame (shared) Variablen der jeweiligen Prozesse $A$ und $B$. Für jeden der drei Versuche: Begründen Sie kurz. warum der Versuch korrekt ist oder erklären Sie, wo der Fehler liegt.

*Hinweis*: Es kann sich sowohl um Fehler handeln, die dazu führen, dass beide Prozesse gleichteitig im kritischen Abschnitt laufen, als auch um Fehler, die zu einem Deadlock führen. (S. 276)

(a) Verusch 1:

```
# Init
int turn = 1;
```

```
# Prozess A
while (turn != 1) { }; // wait
critical_section(); // kritischer Abschnitt
turn = 2
```

```
# Prozess B
while (turn != 2) { }; // wait
critical_section(); // kritischer Abschnitt
turn = 1;
```

**Kommentar/Begründung:** Korrekt: Prozess $B$ kann erst in den kritischen Abschnitt wenn die Bedingung `(turn != 2)` falsch wird. Dies kann nur geschehen nachdem Prozess $A$ den kritischen Abschnitt verlassen hat. Es kommt zu keinem Deadlock da die Bedingung `(turn != 1)` gleich nach dem Init falsch ist und der Prozess $A$ in den kritischen Abschnitt gelangt.

(b) Versuch 2:

```
# Init
boolean lock = false;
```

```
# Prozess A
while (lock == true) { }; // wait
lock = true;
critical_section(); // kritischer Abschnitt
lock = false
```

```
# Prozess B
while (lock == true) { }; // wait
lock = true;
critical_section(); // kritischer Abschnitt
turn = false;
```

**Kommentar/Begründung:** Falsch: Die Bedingung `(lock == true)` ist nach dem Init falsch, wodurch der aktive Prozess die Schleife verlassen kann. Wenn dieser Prozess unterbrochen wird bevor er `lock = true` setzen kann, so kann der andere Prozess in den kritischen Abschnitt. Wird der Prozessor wieder dem anderen Prozess zugeteilt bevor der kritische Abschnitt verlassen wurde, so befinden sich beide im kritischen Abschnitt.

Versuch 3:

```
# Init
boolean lockA = false;
boolean lockB = false;
```

```
# Prozess A
lockA = true;
while (lockB == true) { }; // wait
critical_section(); // kritischer Abschnitt
lockA = false;
```

```
# Prozess B
lockB = true;
while (lockA == true) { }; // wait
critical_section(); // kritischer Abschnitt
lockB = false;
```

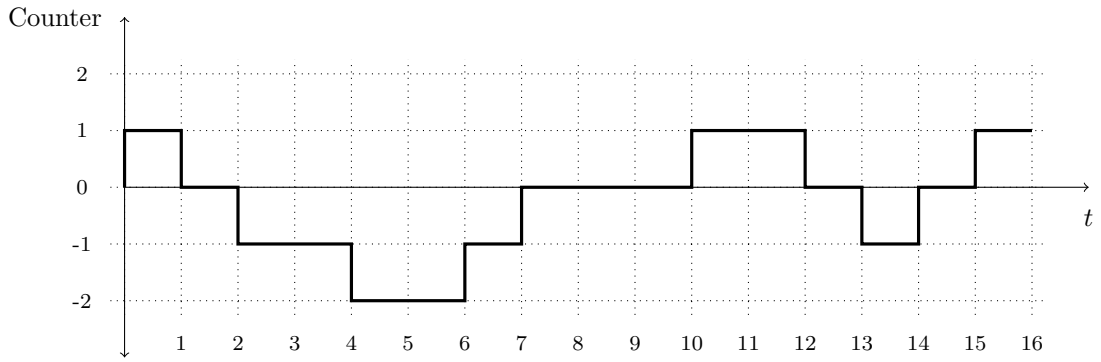
**Kommentar/Begründung:** Falsch: Es kann zu einem Deadlock kommen, wenn gleich nach `lockA = true` in Prozess $A$ dem Prozess $B$ der Prozessor zugeteilt wird und `lockB = true` setzt. Keiner der beiden Prozesse kann nun die While-Schleife verlassen weil die Bedingung niemals falsch wird.

**11.114** Gegeben ist ein Semaphore `S` mit einem Counter, der auf den Wert 1 initialisiert ist. Dabei ist zu beachten, dass die Ordnung der Prozessqueue dieses Semaphores prioritätsgesteuert funktioniert.

Die folgende Tabelle liest eine Menge von Prozessen auf, die diesen Semaphore verwenden wollen (d.h. die `S_P(S)` aufrufen). Dabei ist der Zeitpunkt gegeben, zu dem die jeweiligen Prozesse den Aufruf `S_P(S)` tätigen, sowie die Zeitspanne, die jeder Prozess den Semaphore hält, bevor er mittels `S_V(S)` diesen wieder freigibt. Außerdem ist die Priorität jedes Prozesses angegeben. Dabei bedeutet ein höherer Wert eine höhere Priorität.

| Prozess | Priorität | Zeitpunkt von `S_P(S)` | Haltedauer |
|---|---|---|---|
| A | 1 | 1 | 5 |
| B | 2 | 2 | 3 |
| C | 4 | 4 | 1 |
| D | 0 | 12 | 2 |
| E | 2 | 13 | 1 |

Geben Sie in einem entsprechenden Diagramm den Wert des Counters von Semaphore `S` als Funktion der Zeit an, d.h. zeichnen Sie ein, wann und wie sich der Counter auf Grund der Benutzung durch die Prozesse ändert. (S. 280-281)




**11.115** Geben Sie die 4 notwendingen Bedingungen für die Entstehungen von Deadlocks an! (S. 289)

1. Eine Ressource kann von höchstens einem Prozess besetzt werden.
2. Prozesse die eine Ressource besetzen können weitere beantragen.
3. Ein Prozess blockiert wenn eine beantragte Ressource besetzt ist.
4. Nur der Prozess der eine Ressource besetzt hat kann sie wieder freigeben.

**11.116** Nennen Sie einen Nachteil von asynchronen Methoden zur Interprozesskommunikation (IPC) im Vergleich zu synchronen Methoden. Geben Sie ein Beispiel für asynchrone IPC-Methoden, die unter Unix und auch in MS Windows realisierbar sind. (S. 288)

Bei asynchronen Methoden wird ein Prozess durch ein Signal unterbrochen und führt eine Service-Routine aus. Es kann jedoch zu Race-Conditions kommen wenn z.B. mehrere Prozesse auf die selbe Ressource zugreifen. Unter Unix und in MS Windows gibt es Methoden mit denen ein Programm einen Handler für Signale (z.B. SIGABRT) installieren kann.

# 12 Speicherverwaltung

**12.117** Warum virtuelle Speicherverwaltung? (S. 293)

Jedem Prozess wird ein eigener (virtueller) Adressraum zur Verfügung gestellt, dadurch braucht ein Programmierer keine Rücksicht auf das Vorhandensein von physikalischem Speicher nehmen. Zudem kann fragmentierter physikalischer Speicher dem Prozess als kontinuierlich präsentiert werden. Weiters ermöglicht die virtuelle Speicherverwaltung die Implementierung von Memory Protection zwischen Prozessen.

**12.118** Wie groß ist der Adressraum, der mit 16-Bit Adressen adressiert werden kann? (S. 293)

$2^{16} = 65636$ Byte $= 64$ KByte

**12.119** Wieviele Adressen kann man mit 32 bzw. 64 Bit adressieren? (S. 293)

$2^{32}$ bzw. $2^{64}$ Adressen.

**12.120** Ihre Oma kauft sich ein System mit 16 GB Hauptspeicher und 1500 GB Externspeicher. Sie fragt Sie, welches Betriebssystem sie verwenden soll: (a) Windows XP Home 64 Bit Edition oder (b) Windows Vista Ultimate 32 Bit Edition. Begründen Sie Ihre Antwort. (S. 293)

(a) Weil die 32 Bit Edition nur maximal 4 GB des Hauptspeichers adressieren kann.

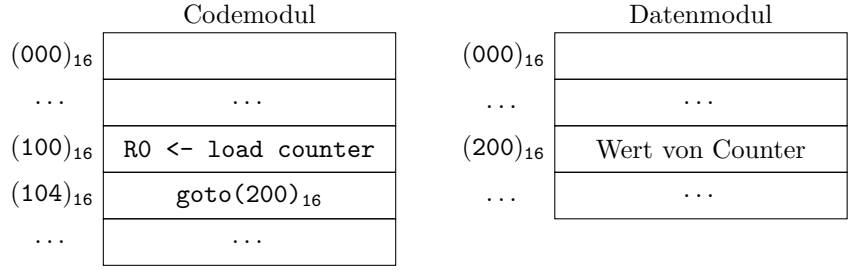
**12.121** Geben Sie zumindest drei verschiedene Segmente an, die sich in einem ausführbaren Programm befinden. (S. 294)

1. Codesegment für ausführbare Maschinenbefehle
2. Datensegment für den Stack
3. Datensegment für Variablen

**12.122** Was ist die Aufgabe eines Linkers? (S. 296)

Fügt separat kompilierte Module zu einem ausführbaren Programm zusammen. Das vereinfacht die Entwicklung, weil Linken wesentlich einfacher als Kompilieren ist und existierende Module nicht neu kompiliert werden müssen.

**12.123** Ein Linker soll ein Codemodul und ein Datenmodul zu einem ausführbaren Programm zusammenfügen. Sowohl das Codemodul als auch das Datenmodul liegen als Relocatable Object Code vor. Das Codemodul soll im ausführbaren Programm an der virtuellen Adresse `0x08048000` beginnen. Das Datenmodul soll an der virtuellen Adresse `0x04000000` liegen. Im Folgenden sind Ausschnitte aus dem Codemodul und dem Datenmodul gegeben.



Geben Sie an, nach welchen virtuellen Adressen (a) der `load` Befehl, (b) der `goto` Befehl und (c) der Wert der Variablen *counter* nach dem Linken (im ausführbaren Programm) gespeichert sind. Geben Sie außerdem an, (d) zu welcher virtuellen Zieladresse der `goto` Befehl springt und (e) zu welcher Adresse die *unresolved external address* des `load` Befehls aufgelöst wird. (S. 296)

(a) `0x08048100`

(b) `0x08048104`

(c) `0x04000200`

(d) `0x08048200`

(e) `0x04000200`

**12.124** Speicherverwaltung: Was ist Swapping/Paging?

**Swapping:** (S. 300)

1. Ein Prozessimage wird zur Gänze zwischen dem pyhsikalischen und virtuellen Speicher hin- und herbewegt.
2. Jedem Prozess wird genau ein physikalisches Codesegment zugeordnet, das selbe Codesegment kann aber mehreren Prozessen zugeordnet sein.
3. Daten-Segmente werden in der Regel nur einem Prozess zugeordnet, d.h. jedem Prozess wird ein eigenes Daten-Segment zugeordnet.
4. Alle Referenzen auf (Daten-)Speicherzellen werden über ein Index-Register (aka Segment-Register, Base Address Register) durchgeführt, dem bei der Erzeugung des Prozesses die physikalische Anfangsadresse des jeweiligen Daten-Segments zugewiesen wird. Die fix zugeordnete Adresse jeder Variable ist relativ zum Beginn des Datensegmentes zu interpretieren.
5. Die Physikalische Adresse ergibt sich durch Addition des Segment-Registers zur virtuellen Adresse.
6. Für das Laden der Images in den physikalischen Speicher gibt es zwei Varianten. Die Images werden entweder in Partitionen mit fixer oder variabler Größe in den physikalischen Speicher geladen.

**Paging:** (S. 303)

1. Der virtuelle Adressraum wird in gleichgroße Pages unterteilt denen jeweils genau ein Page Frame im physikalischen Adressraum zugeordnet ist.
2. Durch Ausnutzung der Lokalität der Referenzen werden nur benötigte Pages im physikalischen Speicher gehalten.
3. Die Menge der benötigten Pages in einem Zeitintervall ab einem gewissen Zeitpunkt wird Working Set genannt.
4. Wenn der physikalische Speicher nicht alle Working Sets aufnehmen kann kommt es zu Trashing: die Maschine muss ständig Pages vom Externspeicher laden.
5. Die Zuordnung einer Page zu einem Page Frame geschieht mittels einer Page Table: bei jeder Page-Nummer ist eine Frame-Nummer eingetragen.
6. Die virtuelle Adresse ist aufgeteilt in Page-Nummber-Bits und Offset-Bits.

7. Physikalische Adresse ermitteln:
   (a) Die Anzahl der Bits für das Offset $o$ kann aus der Page-Size $s$ berechnet werden: $o(s) = \log_2(s)$. Beispiel: Page Size = 2 KByte = $2^{11}$ Byte $\Rightarrow$ Offset = 11.
   (b) Page Nummer: Virtuellen Adresse als Binärzahl anschreiben und die Offset-Bits wegschneiden.
   (c) Frame Number: Mit Hilfe der Page-Nummer aus der Page-Table raussuchen.
   (d) Physikalische Adresse: Die Frame Nummer als Binärzahl anschreiben und die Offset-Bits ankleben.
8. Auf dieses Art ergibt sich implizite Memory Protection, da ein Prozess nur seine eigenen Pages referenzieren kann.
9. Zusätzliche Access-Modes in der Page-Table erlauben Überwachung der Speicherzugriffe auf Page-Ebene.
10. Bei einem Page-Fault, also wenn ein Zugriff auf eine virtuelle Adresse erfolgt die noch nicht geladen wurde, wird der auslösende Prozess unterbrochen und darf wieder fortsetzen wenn das entsprechende Page-Frame geladen wurde.

**12.125** Was versteht man unter Swapping? (S. 300)

Das Gesamte Prozessimage wird zwischen virtuellem und physikalischem Speicher hin- und herbewegt. Zur Berechnung der physikalischen Adresse werden Index-Register verwendet: die darin gespeicherte Basisadresse wird zur virtuellen Adresse addiert.

**12.126** Was versteht man unter Paging? (S. 303)

Der virtueller Adressraum eines Prozesses wird in Pages unterteilt. Analog dazu wird der physikalische Speicher in Page-Frames unterteilt. Der Vorteil gegenüber Swapping ist, dass nur jene Pages im physikalischen Speicher gehalten werden die tatsächlich gebraucht werden. Zur Berechnung der Physikalischen Adresse wird eine Page-Table geführt, welche jeder Page einen Page-Frame zuordnet.

**12.127** Ein Speicherverwaltungssystem mit Paging verwendet 32-Bit virtuelle Adressen und 24-Bit physikalische Adressen. Die Größe einer Page beträgt 2 KB (2048 Bytes). Gegeben ist eine Pagetable. Wandeln Sie mit Hilfe dieser Tabelle die folgenden vier virtuellen 32-Bit-Adressen in die entsprechende physikalische Adresse um. Geben Sie dabei jeweils an, ob ein Page Fault auftritt.

Virtuelle Adressen:

$(000008BF)_{16}$
$(0006E1F2)_{16}$
$(0006E821)_{16}$
$(00001047)_{16}$

Page Table:

| Page Number | Frame Number |
|---|---|
| $(0)_{16}$ | not present |
| $(1)_{16}$ | not present |
| $(20)_{16}$ | $(AF)_{16}$ |
| ... | ... |
| $(D9)_{16}$ | not present |
| $(DA)_{16}$ | $(C04)_{16}$ |
| $(DB)_{16}$ | not present |
| $(DC)_{16}$ | $(7C)_{16}$ |
| $(DD)_{16}$ | not present |
| $(DE)_{16}$ | $(B48)_{16}$ |
| ... | ... |

(S. 303)

$2048 = 2^{11}$ d.h. die letzten 11 Bit sind für den Offset.

- $(\texttt{8BF})_{16} = (\texttt{1000 1011 1111})_2$
  → Page Number = $(\texttt{1})_2 = (\texttt{1})_{16}$
  → Page Fault
- $(\texttt{6E1F2})_{16} = (\texttt{110 1110 00001 1111 0010})_2$
  → Page Number = $(\texttt{1101 1100})_2 = (\texttt{DC})_{16}$
  → Frame Number = $(\texttt{7C})_{16} = (\texttt{111 1100})_2$
  → Physikalische Adresse = $(\texttt{11 1110 0001 1111 0010})_2 = (\texttt{3E1F2})_{16}$
- $(\texttt{6E821})_{16} = (\texttt{110 1110 1000 0010 0001})_2$
  → Page Number = $(\texttt{1101 1101})_2 = (\texttt{DD})_{16}$
  → Page Fault
- $(\texttt{1047})_{16} = (\texttt{1 0000 0100 0111})_2$
  → Page Number = $(\texttt{10})_2 = (\texttt{2})_{16}$
  → Frame Number = $(\texttt{AF})_{16} = (\texttt{1010 1111})_2$
  → Physikalische Adresse = $(\texttt{101 0111 1000 0100 0111})_2 = (\texttt{57847})_{16}$

**12.128** Ein Speicherverwaltungssystem mit Paging verwendet 32-Bit virtuelle Adressen und 24-Bit physikalische Adressen. Die Größe einer Page beträgt 8 KB (8192 Bytes). Gegeben ist eine Pagetable. Wandeln Sie mit Hilfe dieser Tabelle die folgenden vier virtuellen 32-Bit-Adressen in die entsprechende physikalische Adresse um. Geben Sie dabei jeweils an, ob ein Page Fault auftritt.

Virtuelle Adressen:

$(\texttt{000078BF})_{16}$
$(\texttt{001DD1AB})_{16}$
$(\texttt{001DA472})_{16}$
$(\texttt{00004A02})_{16}$

Page Table:

| Page number | Frame number |
|---|---|
| $(\texttt{1})_{16}$ | not present |
| $(\texttt{2})_{16}$ | not present |
| $(\texttt{3})_{16}$ | $(\texttt{1B})_{16}$ |
| ... | ... |
| $(\texttt{E9})_{16}$ | not present |
| $(\texttt{EA})_{16}$ | $(\texttt{207})_{16}$ |
| $(\texttt{EB})_{16}$ | not present |
| $(\texttt{EC})_{16}$ | $(\texttt{7C})_{16}$ |
| $(\texttt{ED})_{16}$ | not present |
| $(\texttt{EE})_{16}$ | $(\texttt{380})_{16}$ |
| ... | ... |

(S. 303)

$8192 = 2^{13}$ d.h. die letzten 13 Bit sind für den Offset.

- $(\texttt{78BF})_{16} = (\texttt{0111 1000 1011 1111})_2$
  → Page Number = $(\texttt{11})_2 = (\texttt{3})_{16}$
  → Frame Number = $(\texttt{1B})_{16} = (\texttt{1 1011})_2$
  → Physikalische Adresse = $(\texttt{11 0111 1000 1011 1111})_2 = (\texttt{378BF})_{16}$

- $(\texttt{1DD1AB})_{16} = (\texttt{0001 1101 1101 0001 1010 1011})_2$
  $\rightarrow$ Page Number $= (\texttt{1110 1110})_2 = (\texttt{EE})_{16}$
  $\rightarrow$ Frame Number $= (\texttt{380})_{16} = (\texttt{11 1000 0000})_2$
  $\rightarrow$ Physikalische Adresse $= (\texttt{111 0000 0001 0001 1010 1011})_2 = (\texttt{7011AB})_{16}$
- $(\texttt{1DA472})_{16} = (\texttt{1 1101 1010 0100 0111 0010})_2$
  $\rightarrow$ Page Number $= (\texttt{1110 1101})_2 = (\texttt{ED})_{16}$
  $\rightarrow$ Page Fault
- $(\texttt{4A02})_{16} = (\texttt{0100 1010 0000 0010})_2$
  $\rightarrow$ Page Number $= (\texttt{10})_2 = (\texttt{2})_{16}$
  $\rightarrow$ Page Fault

**12.129** Wo wird virtuelle Speicherverwaltung im Rechner umgesetzt? Wie heißt sie und wie ist sie implementiert? (S. 305)

Sie wird in Hardware von der Memory Management Unit umgesetzt und wird mit Paging, Segmentieurng oder Swapping implementiert.

**12.130** Was bedeutet MMU? (S. 305)

Memory Management Unit, sie übernimmt Funktionen der virtuellen Speicherverwaltung, z.B. virtuelle in physikalische Adressen umrechnen.

**12.131** Ein Betriebssystem verwendet den LFU (Last Frequently Used) Algorithmus für das Page Replacement. Wenn zwischen Pages unterschieden werden muss, die gleich oft verwendet worden sind, wird die älteste Page aus dem Speicher entfernt (jene Page, die zuerst geladen worden ist).

Einem Prozess stehen vier physikalische Frames zu Verfügung. Im Folgenden ist die Reihenfolge der Pagenummern angegeben, auf die der Prozess zugreift (Referenzstring). Geben Sie nach jedem Zugriff an, welche Pages sich im physikalischen Speicher befinden.

Referenzstring: 1 2 1 3 4 2 5 5 6 7 8 5 8 9 9 10

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| | | | 3 | 3 | 3 | 5 | 5 |
| | | | | 4 | 4 | 4 | 4 |
| 1 | 1 | 1 | 1 | 1 | 9 | 9 | 9 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 10 |
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 6 | 7 | 8 | 8 | 8 | 8 | 8 | 8 |

(S. 306)

# 13 Ressourcen Management

**13.132** Was sind Transactions (im Context Files)? (S. 323)

Änderungen von zusammengehörigen Dateien welche von einem konsistenten Zustand in einen anderen übergeführt werden. Eine Transaktion ist unteilbar und wird garantiert ganz oder gar nicht durchgeführt.

**13.133** Was ist ein Device Driver (S. 323)

Device Driver stellen Standardbefehle als Standardinterfaces für alle I/O Geräte zur Verfügung. Ein Prozess kommuniziert nicht direkt sondern über die Schnittstelle des Drivers mit den Devices. Der Driver kümmert sich dabei um die Eigenheiten der Geräte und hat Möglichkeiten zur Optimierung (z.B. Cache bei Festplatte).

## 14 Sicherheit

**14.134** Was versteht man im Bereich Sicherheit unter dem Begriff Interruption? Geben Sie zwei Beispiele für Angriffe an, die in diese Kategorie fallen! (S. 329)

Der Informationsfluss wird unterbrochen. Beispiele: Durchtrennen der Kommunikationsleitung, Überlastung eines Servers mit Anfragen, Zerstörung der Hardware.

**14.135** Was versteht man im Bereich Sicherheit unter dem Begriff Interception? Geben Sie zwei Beispiele für Angriffe an, die in diese Kategorie fallen! (S. 329)

Ein Angreifer verschafft sich Zugriff auf übertragene Informationen. Z.B. Netzwerksniffer, unautorisiertes Kopieren von Dateien.

**14.136** Was versteht man im Bereich Sicherheit unter dem Begriff Modifikation? Geben Sie zwei Beispiele für Angriffe an, die in diese Kategorie fallen! (S. 330)

Bei diesem Angriff werden übertragene Informationen verändert. Z.B. Ändern von Dateien oder Netzwerkpaketen.

**14.137** Was versteht man im Bereich Sicherheit unter dem Begriff Fabrikation? Geben Sie zwei Beispiele für Angriffe an, die in diese Kategorie fallen! (S. 330)

Der Angreifer bringt gefälschte Daten in den Datenfluss ein. Z.B. Erstellen von zusätzlichen Einträgen in die Passwortdatei oder Eintragen neuer Datensätze in eine Datenbank.

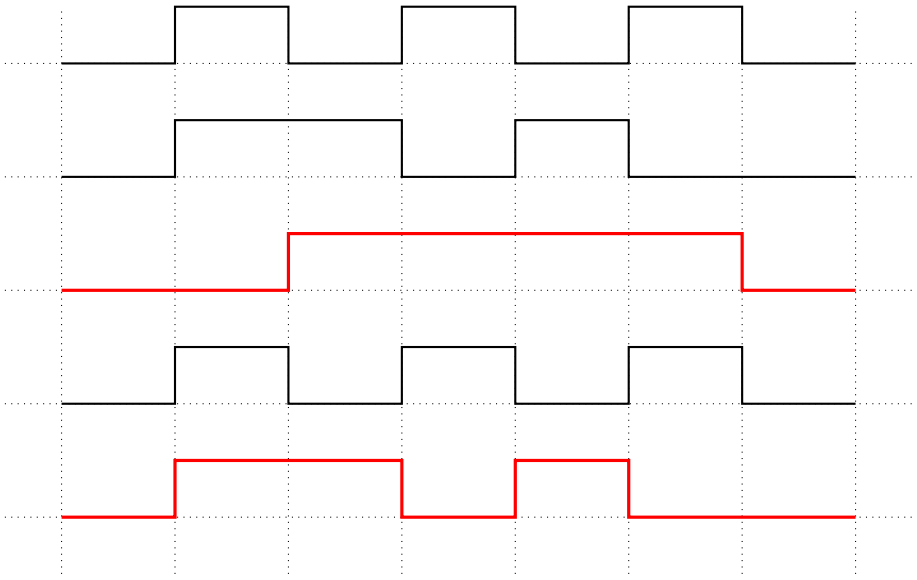
**14.138** Was ist Trusted Computing? (S. 337)

Unter Trusted Computing versteht man, dass sich die verwendete Hard- und Software stets wie erwartet verhält, und dass dieses Verhalten durch Hard- und Software erzwungen wird. Die Trusted Computing Leistungen werden dabei hauptsächlich durch zwei Hardwarekomponenten erbracht: das Trusted Platform Module (TPM) und die Core Root of Trust Measurment (CRTM) Komponente.

# Sonstiges

**14.139** GPS (Global Positioning System) ist ein weltweites Navigationssystem, das es erlaubt, mit Hilfe von Signalen, die von Satelliten ausgesandt werden, die eigene Position zu bestimmen. Das vom Satelliten gesendete Signal ist dabei die Kombination eines Taktsignals mit den eigentlichen Nutzdaten. Dazu werden der Takt und das Nutzsignal mittels einer Antivalenzoperation

miteinander verknüpft und dann gesendet. Beim Empfänger müssen die Nutzdaten aus dem Satellitensignal wieder extrahiert werden. Dabei kann angenommen werden, dass der Empfänger über ein synchronisiertes Taktsignal verfügt. Gegeben sind die Taktsignale beim Satelliten und beim Empfänger sowie das Nutzsignal. Im ersten Schritt ermitteln Sie das Signal, welches der Satellit zu Erde schickt (Sendesignal). Danach müssen Sie überlegen, wie der Empfänger wieder an die Nutzdaten kommt. Geben Sie dazu die notwendige Funktion an (Funktion von Sendesignal und Takt) und tragen Sie das Ergebnis in das Diagramm ein!



Sei $s$ das Sendesignal, $t$ der Takt. Das Nutzsignal $n$ wird durch erneute Anwendung der Antivalenzoperation gewonnen: $n = (s \wedge \neg t) \vee (\neg s \wedge t)$.

**14.140** Ein Codierschalter arbeitet mit BCD-Code. Stellen Sie die Zahl $(265)_{10}$ in BCD-Code dar!

```
0010 0110 0101
```