# IT ABI Zusammenfassung

*Jannis Müller, Robin Rausch*

*03 April 2021*

## Inhaltsverzeichnis

**1 Zahlensysteme** **6**
1.1 Dezimalsystem . . . 6
1.2 Dualsystem . . . 6
1.3 Oktalsystem . . . 6
1.4 Hexadezimalsystem . . . 6

**2 Aufbau Computer** **7**
2.1 Von Neumann Prinzip . . . 7

**3 Betriebssysteme** **8**
3.1 Kernel . . . 8
3.1.1 Monolithischer Kernel . . . 8
3.1.2 Mirkokernel . . . 8
3.1.3 Hybridkernel . . . 9
3.2 Master Boot Record (MBR) . . . 9
3.3 GPT . . . 10
3.4 UEFI und BIOS . . . 10

**4 Dateisysteme** **10**
4.1 FAT . . . 10
4.2 NTFS . . . 11
4.2.1 Dataruns . . . 12

**5 Prozessverwaltung** **12**
5.1 Prozesse . . . 12
5.1.1 Threads . . . 12
5.1.2 Prozesszustände . . . 13
5.2 Mehrprozessbetrieb . . . 13
5.3 Multitasking . . . 14
5.3.1 Kontext . . . 14
5.4 Scheduling . . . 14
5.5 Scheduling Strategien . . . 15
5.5.1 First Come First Serve . . . 15
5.5.2 Shortest Job First . . . 15
5.5.3 Round Robin . . . 15
5.5.4 Priority . . . 15

**6 Speicherkonzepte** **15**
6.1 Speicherhierarchie . . . 15

**7 Speicherverwaltung** **15**
7.1 Direkte Speicherverwaltung . . . 15
7.2 Swapping . . . 15
7.3 virtueller Speicher . . . 16
7.4 segmentorientierter Speicher . . . 16
7.5 seitenorientierter Speicher - *Paging* . . . 17
7.5.1 Beispielaufgaben . . . 18
7.5.2 Buch Aufgabe 1, S.243 . . . 18
7.5.3 Buch Aufgabe 3, S.243 . . . 19
7.5.4 Buch Aufgabe 4, S.243 . . . 19
7.5.5 Buch Aufgabe 5, S.243 . . . 20
7.5.6 Buch Aufgabe 6, S.243 . . . 20
7.5.7 Buch Aufgabe 7, S.243 . . . 21
7.6 Speicherverwaltungsmethodenchronologie . . . 21

**8 Datensicherung** **21**
8.1 Sicherungsmedien . . . 22
8.2 Datenverlust . . . 22
8.3 Datenvernichtung . . . 22
8.4 Sicherungsverfahren . . . 23
8.4.1 Vollständige Sicherung . . . 23
8.4.2 Differenzielle Sicherung . . . 23
8.4.3 Inkrementelle Sicherung . . . 23
8.4.4 Sicherungsstrategie: Generationen Konzept . . . 24
8.5 Vernetzte Sicherung . . . 24
8.5.1 verteilte Sicherung . . . 24
8.5.2 lokale Sicherung . . . 24
8.5.3 Netzwerk Sicherung . . . 24
8.5.4 Kapazität des Netzwerks . . . 25
8.5.5 Logfiles . . . 25
8.6 RAID Systeme . . . 25
8.6.1 RAID 0 . . . 25
8.6.2 RAID 1 . . . 25
8.6.3 RAID 5 . . . 26
8.6.4 RAID 10 . . . 26
8.6.5 RAID 50 . . . 26
8.6.6 Gegenüberstellung der RAID Systeme . . . 26
8.6.7 RAM vs ROM . . . 26

**9 Netze** **27**
9.1 Einführung . . . 27
9.1.1 Beispiele für Netze . . . 27
9.1.2 Gemeinsamkeiten aller Netze . . . 27
9.2 Einfaches Computernetz . . . 27
9.2.1 Netz Topologien . . . 30
9.2.2 Netze in Unternehmen . . . 31

9.2.3 Private Netze . . . 31
9.3 Servermodelle . . . 31
9.3.1 Ein-Server-Modell . . . 31
9.3.2 Multi-Server-Modell . . . 31
9.4 Kommunikationsarten . . . 31
9.4.1 Simplex . . . 31
9.4.2 Halbduplex . . . 31
9.4.3 Vollduplex . . . 32
9.5 Netzwerkkomponenten . . . 32
9.5.1 Port . . . 32
9.5.2 Client . . . 32
9.5.3 Netzwerkkarte . . . 32
9.5.4 Repeater . . . 32
9.5.5 HUB . . . 32
9.5.6 Bridge . . . 33
9.5.7 Switch . . . 33
9.5.8 Router . . . 33
9.5.9 Einordnung der Komponenten ins OSI-Modell . . . 33
9.6 Round Trip Delay Time - RTDT . . . 34
9.7 Bezeichnungen von Netzen . . . 34
9.8 OSI 7-Schichten Modell . . . 34

**10 Ethernet** **35**
10.1 MAC Adresse . . . 36
10.1.1 Aufbau MAC Adresse . . . 36

**11 Netzwerkprotokolle** **37**
11.1 Datenübertragung . . . 37

**12 Vollständiges Ethernet-Paket** **37**

**13 CSMA/CD - Verfahren** **38**
13.1 Kollision . . . 38

**14 ARP-Protokoll** **39**

**15 Subnetting** **40**
15.1 Die IP Adresse . . . 40
15.2 Netzwerkklassen . . . 41
15.3 Subnetzmaske . . . 41
15.4 CIDAR . . . 41
15.5 VLSM . . . 42
15.6 Routing Tabellen . . . 42
15.7 DMZ - Demilitarisierte Zone . . . 42
15.8 IPv6 . . . 43

**16 Schaltnetze** **44**
16.1 Einfache Bauteile . . . 44
16.1.1 NOT . . . 44
16.1.2 AND . . . 44
16.1.3 OR . . . 44
16.1.4 NAND . . . 45
16.1.5 NOR . . . 45
16.1.6 XOR . . . 45
16.1.7 XNOR . . . 46
16.2 Wahrheitstabelle . . . 46
16.3 KV-Diagramm . . . 46
16.3.1 Don't-Care-Positions . . . 47
16.4 Disjunktive Normalform . . . 47
16.5 Impulsdiagramm . . . 48
16.6 Zustandsfolgetabelle . . . 48

**17 Schaltwerke** **49**
17.1 Schaltwerke . . . 49
17.2 Taktgeber . . . 49
17.3 Flip-Flops . . . 49
17.3.1 RS Flip Flop . . . 49
17.3.2 T Flip Flop . . . 50
17.3.3 JK Flip Flop . . . 50
17.3.4 D Flip Flop . . . 50
17.4 Zähler . . . 51
17.5 Addierschaltungen . . . 51
17.6 Subtrahierschaltungen . . . 52
17.7 Schieberegister . . . 52
17.8 Entprellen von Signalen . . . 53

**18 Mirkocontroller** **53**
18.1 Aufbau . . . 53

**19 Assembler** **54**
19.1 Wichtige Befehle . . . 54
19.2 Adressierungsarten bei Mikrocontroller . . . 54
19.3 Datenpointer . . . 54
19.4 Zeitschleife erstellen ohne Timer . . . 55
19.5 Zeitberechnung mit der Näherungsformel . . . 55
19.6 Interrupt . . . 56

**20 Objektorientierte Programmierung** **56**
20.1 Grundverständnis . . . 56
20.2 Klassen . . . 56
20.3 Objekte . . . 57
20.4 Datentypen . . . 57
20.5 Arrays . . . 57
20.6 Verebung . . . 57
20.7 Polymorphie . . . 57

20.7.1 Polymorphie . . . . . 57
20.7.2 Dynamische Polymorphie . . . . . 58
20.8 UML . . . . . 58
20.8.1 Struktogramm . . . . . 58
20.8.2 Klassendiagramm . . . . . 59
20.8.3 Objektdiagramm . . . . . 59
20.8.4 Assoziationen . . . . . 59
20.8.5 Polymorphie . . . . . 59
20.8.6 Vererbungen . . . . . 59
20.8.7 Sequenzdiagramm . . . . . 59
20.8.8 Zustandsdiagramm . . . . . 59

**21 Datenbanken** **59**
21.1 Grundverständnis Datenbanken . . . . . 59
21.2 Verwendung . . . . . 60
21.3 Entity-Relationship-Diagramm . . . . . 60
21.4 Primärschlüssel . . . . . 60
21.5 Sekundärschlüssel . . . . . 60
21.6 Realtionsschreibweise . . . . . 61
21.7 SQL . . . . . 61
21.7.1 SQL Abfragen . . . . . 61

**22 Fachsprache** **62**
22.1 Fachwörter . . . . . 62
22.2 Typfeld Kennungen . . . . . 62

**23 Epilog** **62**
23.1 Literatur . . . . . 62

# 1 Zahlensysteme

## 1.1 Dezimalsystem

| | |
|---|---|
| Nennwerte: | 0,1,[. . . ],8,9 |
| Basis: | 10 |
| Größter Nennwert: | 9 |

| $10^1$ | $10^0$ |
|---|---|
| 10 | 1 |
| 0 | 0 |
| 0 | 1 |
| 0 | 2 |
| 0 | 3 |
| 0 | 4 |
| 0 | 5 |
| 0 | 6 |
| 0 | 7 |
| 0 | 8 |
| 0 | 9 |
| 1 | 0 |
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |
| 1 | 4 |
| 1 | 5 |

## 1.2 Dualsystem

| | |
|---|---|
| Nennwerte: | 0,1 |
| Basis: | 2 |
| Größter Nennwert: | 1 |

| $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|
| 8 | 4 | 2 | 1 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

## 1.3 Oktalsystem

| | |
|---|---|
| Nennwerte: | 0,1,[. . . ],6,7 |
| Basis: | 8 |
| Größter Nennwert: | 7 |

| $8^1$ | $8^0$ |
|---|---|
| 8 | 1 |
| 0 | 0 |
| 0 | 1 |
| 0 | 2 |
| 0 | 3 |
| 0 | 4 |
| 0 | 5 |
| 0 | 6 |
| 0 | 7 |
| 1 | 0 |
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |
| 1 | 4 |
| 1 | 5 |
| 1 | 6 |
| 1 | 7 |

## 1.4 Hexadezimalsystem

| | |
|---|---|
| Nennwerte: | 0,1,[. . . ],8,9,A,B,C,D,E,F |
| Basis: | 16 |
| Größter Nennwert: | F |

| $16^1$ | $16^0$ |
|---|---|
| 8 | 1 |
| 0 | 0 |
| 0 | 1 |
| 0 | 2 |
| 0 | 3 |
| 0 | 4 |
| 0 | 5 |
| 0 | 6 |
| 0 | 7 |
| 0 | 8 |
| 0 | 9 |
| 0 | A |
| 0 | B |
| 0 | C |
| 0 | D |
| 0 | E |
| 0 | F |

# 2 Aufbau Computer

## 2.1 Von Neumann Prinzip

**Quellcode:**

$$\text{zahl1} = 1;$$
$$\text{zahl2} = 4;$$
$$\text{zahl1} + \text{zahl2} = \text{sum};$$

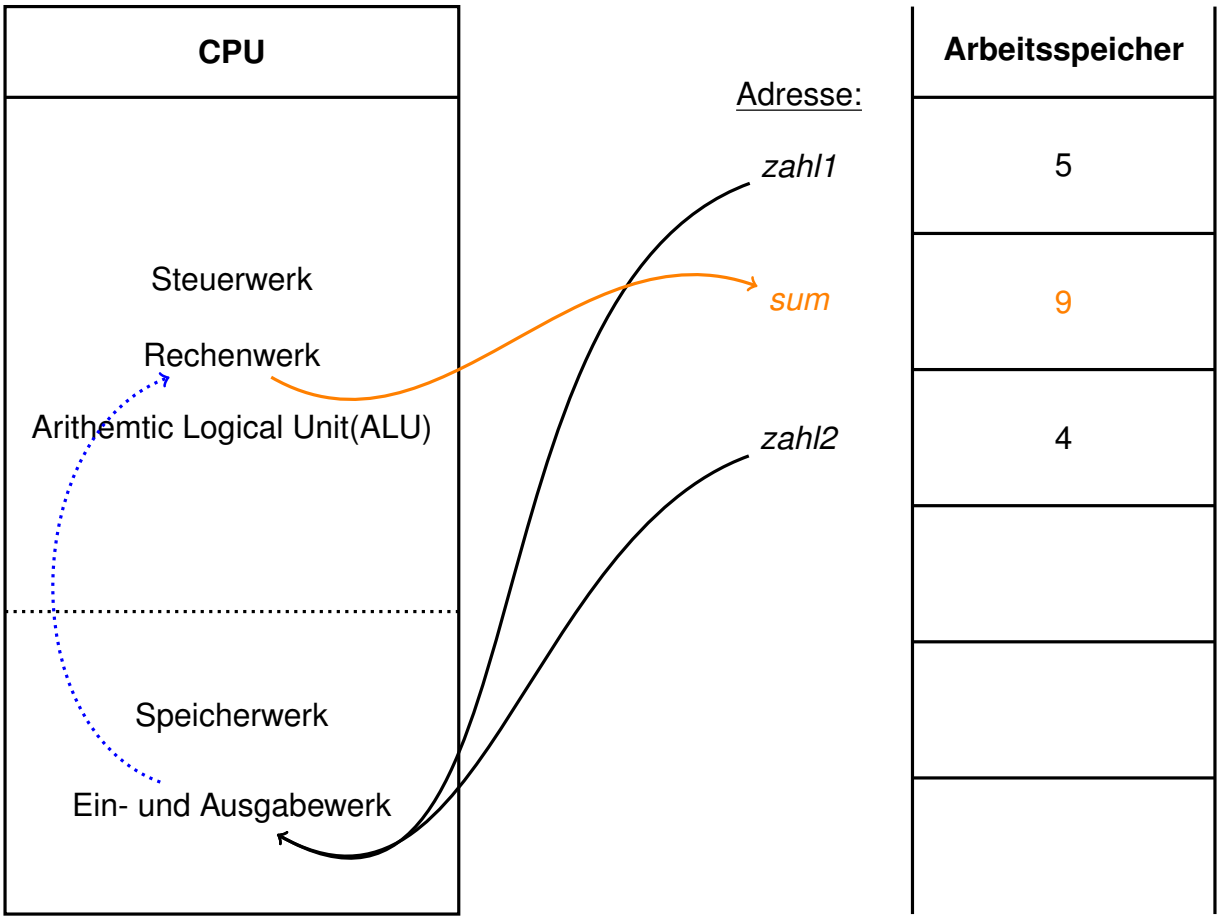



Abbildung 1: Von Neumann Prinzip

# 3 Betriebssysteme

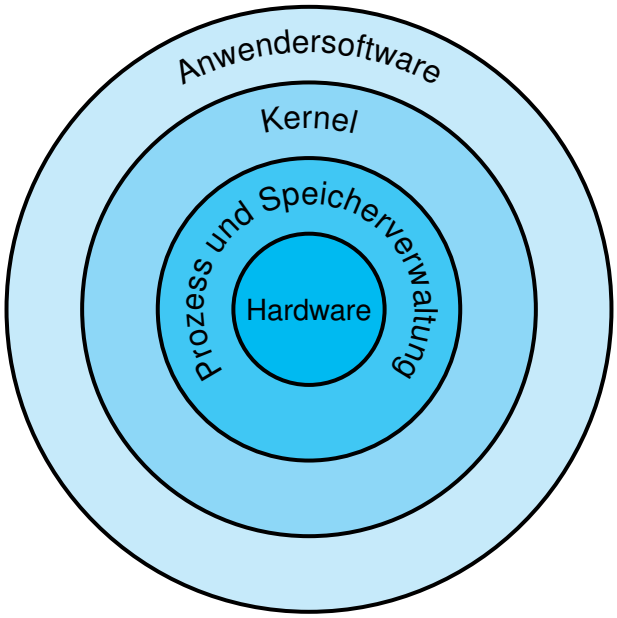



Abbildung 2: Aufbau eines Betriebssystemes

## 3.1 Kernel

Der Kernel ist die Schnittstelle zwischen der Anwendersoftware und der Speicher- und Prozessverwaltung. Er kann auch als Betriebssystem-Kern bezeichnet werden.

### 3.1.1 Monolithischer Kernel

**+** sehr schnell, da alles im Kernel geschieht und es kaum Schnittstellen hat

**-** wenn eine Komponente defekt ist oder einen Fehler hat, ist der Kernel nichtmehr funktionsfähig

**-** Updates müssen im Gesamten direkt erfolgen(z.B. Treiberupdates)

Der Monolitische Kernel wird bei Betriebsystemen wie z.B. DOS, Linux und Android verwendet.

### 3.1.2 Mirkokernel

**+** wenn eine Komponente defekt ist oder einen Fehler hat, kann man den Kernel trotzdem weiterhin verwenden

**-** Prozesse dauern länger, da auch Prozesse/Treiber ausgelagert sind

Der Mirkokernel wird bei Betriebsystemen wie z.B. Echtzeitsbetriebssystemen oder RISC OS verwendet.

### 3.1.3 Hybridkernel

**+** Nutzung beider Vorteile

**-** Nutzung beider Nachteile

Der Hybridkernel wird bei Betriebsystemen wie z.B. Windows oder MAC OS verwendet.

## 3.2 Master Boot Record (MBR)

Der Master Boot Record (kurz MBR) enthält ein Startprogramm für BIOS-basierte Computer und eine Partitionstabelle. Außerdem werden durch den MBR die Partitionen einer Festplatte eingeteilt. Zudem kann man die Betriebssysteme und Dateisysteme der einzelnen Partitionen herauslesen.

Tabelle 1: Partitionstabelleneinträge

| Adresse | Inhalt |
|---|---|
| 00 | 80hex = bootfähige Partition<br>00hex = nicht bootfähige Partition |
| 01 | CHS-Eintrag des ersten Sektors |
| 04 | Typ der Partition (Dateisystem/Partitionsytp) |
| 05 | CHS-Eintrag des letzten Sektors |
| 08 | Startsektor nach LBA |
| 0C | Anzahl der Sektoren in der Partition |

Tabelle 2: Master Boot Record

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | eb | 48 | 90 | 10 | 8e | d0 | bc | 00 | b0 | b8 | 00 | 00 | 8e | d8 | 8e | c0 |
| 0010 | fb | be | 00 | 7c | bf | 00 | 06 | b9 | 00 | 02 | f3 | a4 | ea | 21 | 06 | 00 |
| ... | | | | | | | | | | | | | | | | |
| 01a0 | 10 | ac | 3c | 00 | 75 | f4 | c3 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01b0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | 01 |
| 01c0 | 01 | 00 | 83 | fe | ff | ff | 3f | 00 | 00 | 00 | 41 | 29 | 54 | 02 | 00 | fe |
| 01d0 | ff | ff | 82 | fe | ff | ff | 80 | 29 | 54 | 02 | fa | e7 | 1d | 00 | 00 | fe |
| 01e0 | ff | ff | 83 | fe | ff | ff | 7a | 11 | 72 | 02 | fa | e7 | 1d | 00 | 00 | fe |
| 01f0 | ff | ff | 05 | fe | ff | ff | 74 | f9 | 8f | 02 | 0c | 83 | 6c | 04 | 55 | aa |

Bootloader | Disk Signatur | Nullbytes | Partitionstabelle | Magic Number

### 3.3 GPT

Die Guide Partition Table ist der bessere MBR und hat folgende Vor- und Nachteile gegenüber dem MBR:

+ Funktioniert auch bei 64-Bit Systemen
+ beliebig viele primäre Partitionen
+ verfügt über Schutz- und Sicherheitsmerkmale(Backups & Prüfsummen)
- Nur mit UEFI kompatibel

### 3.4 UEFI und BIOS

UEFI ist das modernere und bessere BIOS. UEFI verfügt gegenüber BIOS folgende Vorteile:

- schnelleres Starten durch paralleles Laden der Treiber
- Datenträger können auch mit mehr als 2TB booten
- Funktioniert auch bei modernen 64-Bit Systemen
- Netzwerke können auch gebootet werden

## 4 Dateisysteme

### 4.1 FAT

Die FAT (*File Allocation Table, Dateizuordnungstabelle*) stellt ein Inhaltsverzeichnis der Festplatte dar und ist notwendig, um die auf dem Datenträger bereitstehenden Daten zu finden. In einem 16-Bit System (FAT 16) sind genau $2^{16} = 65536$ Adressenverwaltungseinträge in der FAT möglich. Bei einer Clustergröße von 512 Byte beträgt die maximale Partitionsgröße 32 MiB ($2^{16} \cdot 512$ Byte).

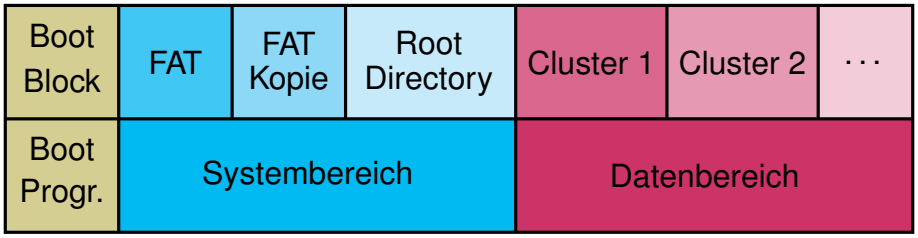



Abbildung 3: Aufbau des FAT Dateisystems

Durch die geringe Clustergröße können Dateien meist nur fragmentiert abgelegt werden. In einer Dateizuordnungstabelle werden Dateipfade gespeichert und können ausgelesen werden. Das richtige Zusammensetzen der Fragmente ist so möglich.
Ein Nachteil des FAT Systemes liegt darin, dass die FAT an einer festen Position auf dem Datenträger gespeichert ist. Bei Festplatten (Speichermedien mit rotierender Disk und Lesekopf) muss der Lesekopf bei jedem Lese/Schreib Vorgang zur FAT und zurück bewegt werden. Dann muss gewartet werden, bis die FAT den Lesekopf passiert. Das kostet viel Zeit. Heutzutage findet man das FAT System in der Flash Drive Technik.

## 4.2 NTFS

Das NTFS (*New Technology File System*) Dateisystem wurde erstmals mit Windows NT (*New Technology*) eingeführt und ist seither das preferierte Dateisystem von Windows. NTFS verwaltet Dateien anders als bei FAT in der MFT (*Mater File Table*). Diese enthält die sogenannten Records. Zu jeder abgelegten Datei existiert also ein Record in der MFT der folgende Informationen enthält: Header, Informationen (z.B. Erstellungsdatum), den Dateinamen, Zugriffsrechte und die eigentlichen Daten bzw. die Position der Daten im Datenbereich. Dateien kleiner 1.5 KiB können direkt in den Record in der MFT gespeichert werden, wohingegen Dateien größer 1.5 KiB in den Datenbereich des Datenträgers gespeichert werden. Der Record verweist dann auf die entsprechende Stelle. Der Datenbereich ist bei NTFS variabel. Daraus resultieren kürzere Lese/Schreib Zeiten. Außerdem ist die MFT zwei mal zwischen den Datenbereichen enthalten.

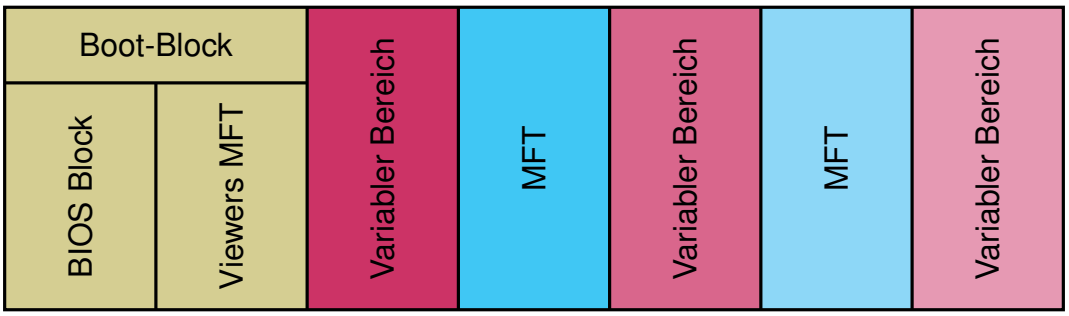



Abbildung 4: Aufbau einer NTFS Partition

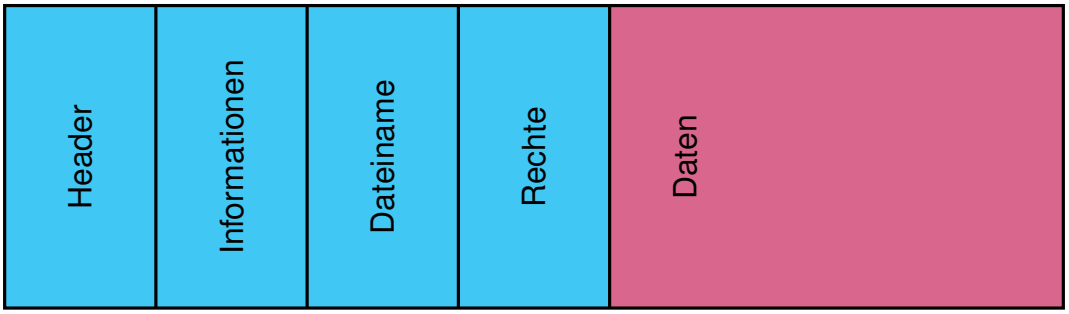



Abbildung 5: NTFS Record Dateien kleiner 1.5KiB

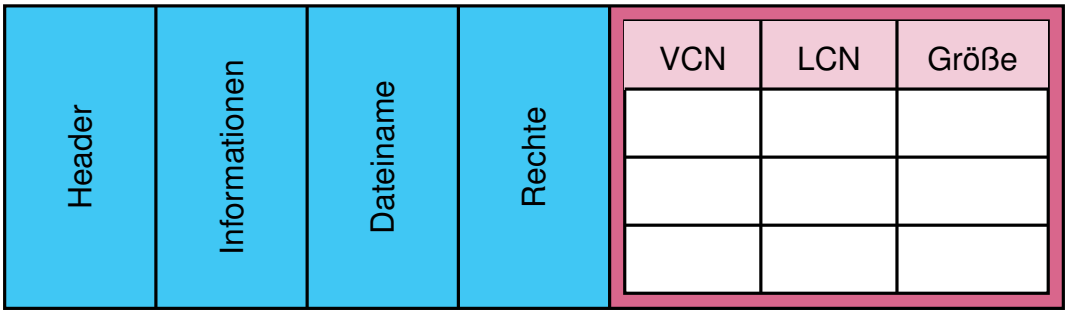



Abbildung 6: NTFS Record Dateien größer 1.5KiB

NTFS verfügt über eine hohe Fehlertoleranz. Eine geschriebene Datei wird mit der sich noch im RAM befindender Original Datei verglichen, um Fehler zu vermeiden. Wird ein Fehler erkannt, wird der betreffende Block als fehlerhaft markiert. Der Schreibvorgang wird dann an einem Anderen versucht. Dieses Prinzip nennt man Hot-Fixing. NTFS verfügt zudem über eine sogenannte LOGFILE. Alle Transaktionen werden in diese eingetragen. So kann nach einem Absturz verlorenes Datengut wiederhergestellt werden. NTFS Dateinamen können bis zu 255 Unicode Zeichen lang sein.

### 4.2.1 Dataruns

Ist ein Dateneintrag bei NTFS zu groß, muss dieser in den Datenbereich ausgelagert werden, liegt eine sehr große Datei vor, so muss diese zusätzlich fragmentiert werden. Diese Fragmentierung wird mit Hilfe des B-Tree Prinzipes in den Datenbereich des Records eingetragen. Mittles LCN-VCN Mapping ist es nun möglich die fragmentierte Datei auf dem Datenträger richtig und vollständig zusammen zu setzen.

# 5 Prozessverwaltung

Die Betriebsmittel des Computersystems müssen zwischen den verschiedenen laufenden Programmen und Systemaufgaben verteilt werden. Zu diesem Zweck werden die einzelnen Aufgaben als Prozesse ausgeführt, die vom Betriebssystem verwaltet werden.

## 5.1 Prozesse

Ein Prozess besteht aus dem Programmcode und dem Prozesskontext. Der Prozesskontext wiederum setzt sich zusammen aus den Registerinhalten des Prozessors, Speicherbereichen für die Daten und weiteren Betriebsmitteln.
Ein Programm kann sich aus mehreren Prozessen (Windows: Tasks) zusammensetzen. Ein Prozess kann Unterprozesse besitzen. So ist hier B ein Unterprozess von A, D E und F wiederum sind Unterprozesse von B.

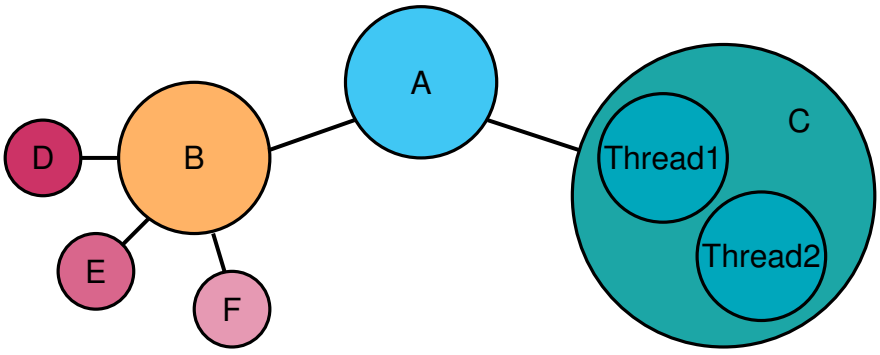



Abbildung 7: Prozesse, Unterprozesse und Threads

### 5.1.1 Threads

Prozesse können neben eigenständigen Unterprozessen auch noch Threads haben. Threads sind leichtgewichtige Prozesse innerhalb eines übergeordneten Prozesses. Threads teilen

sich mit anderen Threads verschiedene Betriebsmittel. Threads, die dem selben Prozess zugeordnet sind, verwenden den gleichen Adressraum. Dadurch ist eine Kommunikation zwischen diesen Threads möglich.

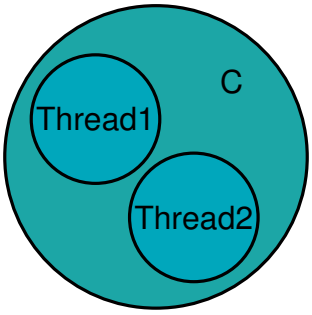



Abbildung 8: Threads in Prozess C

### 5.1.2 Prozesszustände

Ein Prozess kann folgende Zustände annehmen:

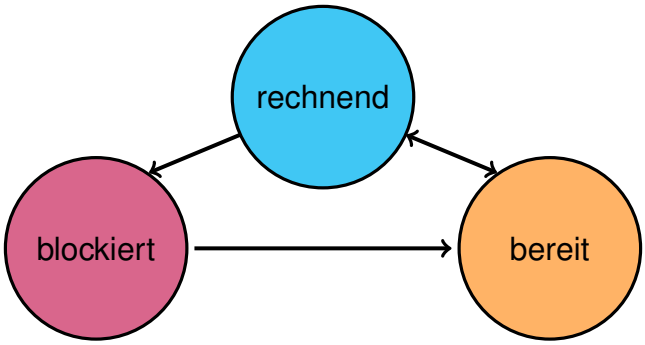
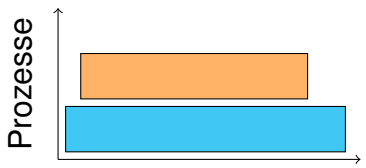



Abbildung 9: Prozessmodell

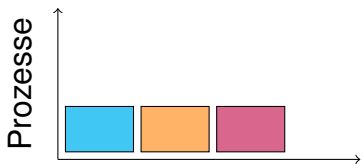
## 5.2 Mehrprozessbetrieb

Der Kern eines Prozessors kann immer nur einen Prozess bearbeiten. Prozesse nutzen die zur Verfügung stehende Rechenzeit allerdings nicht optimal, da viele Prozesse auf langsamere äußere Einflüsse warten müssen (bsp. Tastatureingabe). Die Wartezeit eines Prozesses kann effizient genutzt werden, indem diese Wartezeit einem anderen Prozess zugeteilt wird. Man unterscheidet hierbei zwischen zwei Prinzipien:

1. parallel, wenn jeder Kern einer CPU genau einen Prozess bearbeitet
2. quasiparallel, wenn ein Kern einer CPU mehrere Prozesse nacheinander bearbeitet.




parallel




quasiparallel

## 5.3 Multitasking

Es wird zwischen zwei Arten von Multitasking unterschieden:

1. Koorperatives Multitasking, hier ist es jedem Prozess selbst überlassen, wann er die Kontrolle an das Betriebssystem zurück gibt. Geringer Verwaltungsaufwand, aber es besteht die Gefahr, dass ein unkoorperativer Prozess das System blockiert.

2. Präemptives Multitasking, hier steuert das Betriebssystem die Vergabe der Rechenzeit. Der Scheduler kann einem Prozess Zeit oder Ereignis gesteuert Rechenzeit entziehen. Somit ist die Bearbeitung wichtigerer Prozesse zu jedem Zeitpunkt möglich, ein unkoorperativer Prozess legt das System nicht lahm.

### 5.3.1 Kontext

Um einen Prozess unterbrechen zu können, muss dieser nach der Unterbrechung dieselbe Umgebung vorfinden können, welche der Prozess hatte, bevor er unterbrochen wurde. Der Zustand des Prozesses wird also vor der Unterbrechung gespeichert, und bleibt das so lange, bis ihm wieder Rechenzeit zugeteilt wird. Unmittelbar bevor er wieder aktiv wird, wird der gespeicherte Zustand geladen. Aus Sicht des Prozesses scheint es so, als wäre er nie angehalten worden.
Wichtig für dieses Verhalten ist, dass Prozesse abgeschirmt voneinander sein müssen. Prozess A darf den Kontext von Prozess B nicht kennen. Prozess A erfährt immer nur seinen eigenen Kontext.

## 5.4 Scheduling

Konkurrieren mehrere Prozesse um die Rechenzeit, teilt der Scheduler ihnen gemäß folgender Kriterien die Rechenzeit zu:

1. **Fairness**, jeder Prozess erhält einen gerechten Anteil der Rechenzeit
2. **Effizienz**, der Prozessor arbeitet möglichst effizient
3. **Antwortzeit**, diese wird für interaktiv arbeitende Benutzer minimiert
4. **Verweilzeit**, die Wartezeit für die Ausgabe von Stapelaufträgen wird minimiert
5. **Durchsatz**, Maximierung der Auftragszahl für ein bestimmtes Zeitintervall

## 5.5 Scheduling Strategien

### 5.5.1 First Come First Serve

### 5.5.2 Shortest Job First

### 5.5.3 Round Robin

### 5.5.4 Priority

# 6 Speicherkonzepte

## 6.1 Speicherhierarchie

Speicher werden nach ihrer Schnelligkeit in eine Hierarchie eingeordnet, von kurzer zu langer Zugriffszeit.

1. Prozessorregister
2. Prozessorcache
3. Hauptspeicher
4. Festplatte
5. Wechseldatenträger

# 7 Speicherverwaltung

## 7.1 Direkte Speicherverwaltung

Beim Einprogrammbetrieb verwaltet das Betriebssystem den Hauptspeicher als einen großen Speicherblock. Dieser besteht aus System und Benutzerbereich.

| | | |
|---|---|---|
| Betriebssystem | Programm | freier Speicher |

*Problem:*
Die Programmgröße ist durch die Speichergröße begrenzt. Der Hauptspeicher ist somit zu klein für alle aktiven Prozesse.
*Lösung: Swapping*

## 7.2 Swapping

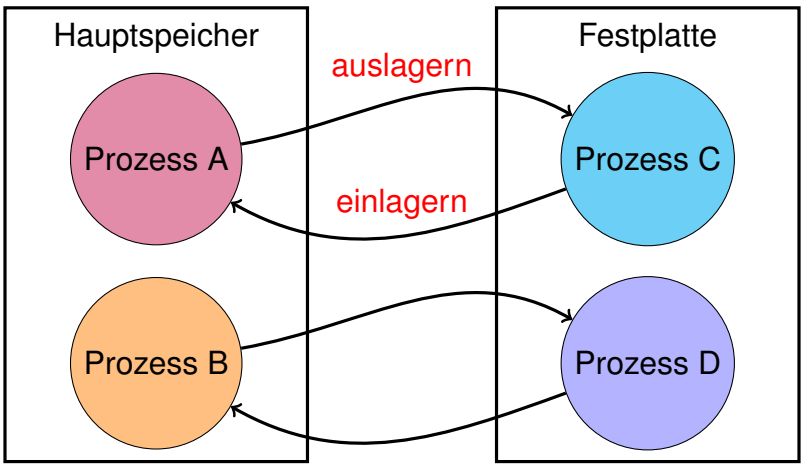
Beim Swapping wird ein Bereich der Festplatte als sogenannter *'Swap-Space'* deklariert. In diesen können Prozesse ausgelagert werden. Daten und Programmcode des Prozesses werden vollständig in den Swap Space ausgelagert.

**Ein ausgelagerter Prozess kann nicht ausgeführt werden.**

Demnach wären hier nur Prozess A und Prozess B ausführbar.

Ein Prozess wird ausgelagert wenn:

- Ein Prozess, der neu entsteht nicht im Hauptspeicher untergebracht werden kann.




- Ein Prozess, seinen Hauptspeicherplatz erweitern möchte, jedoch kein zusätzlicher Speicher verfügbar ist.
- Ein *wichtigerer* Prozess aus dem Swap-Space eingelagert werden muss und für ihn kein Hauptspeicherbereich verfügbar ist.

## 7.3 virtueller Speicher

Festplattenspeicher und Hauptspeicher werden zu einem Speicherblock vereint. Dadurch stehen mehr Speicheradressen zur Verfügung, wie tatsächlich im Hauptspeicher vorhanden sind = virtueller Speicher.

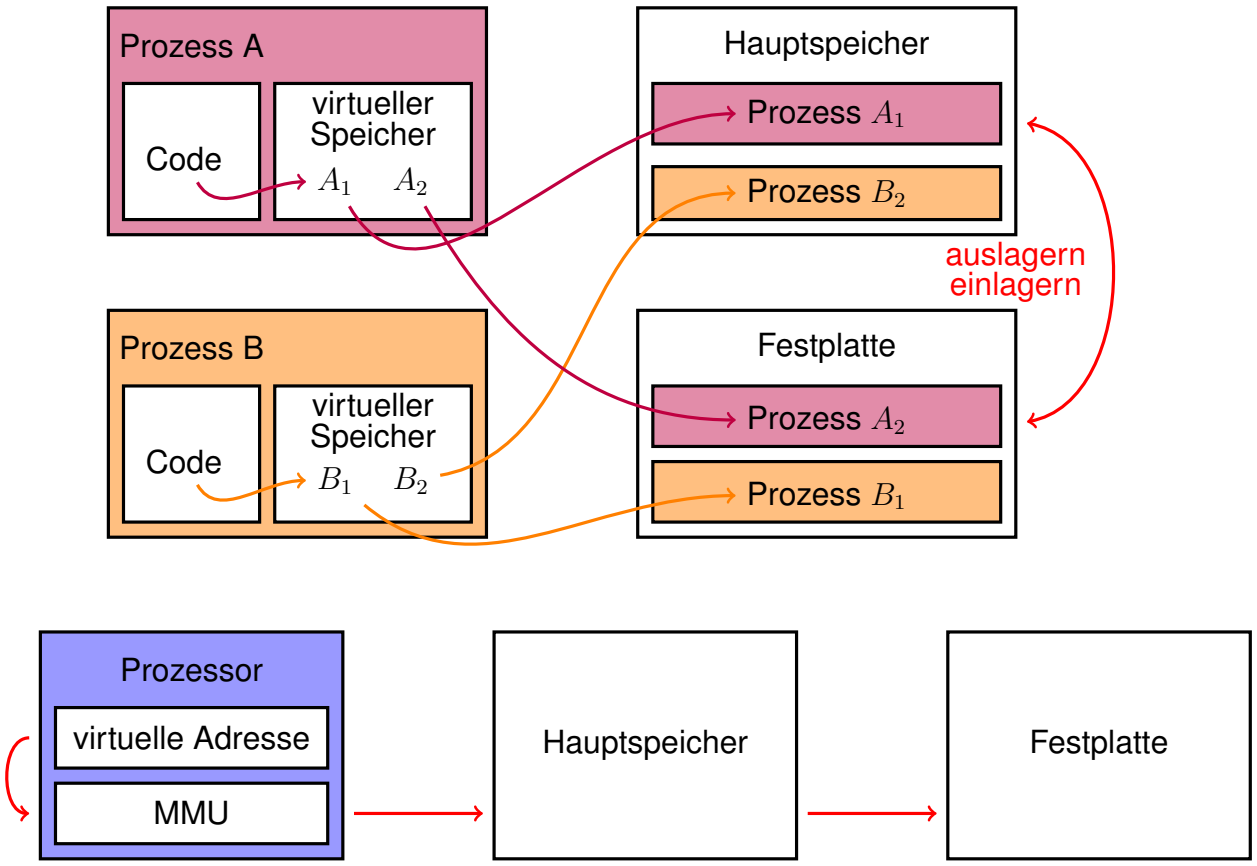
Im Gegensatz zum Swapping können beim virtuellen Speicher Prozesse auch nur teilweise im Hauptspeicher stehen, während der Rest meist auf dem Festplattenspeicher liegt.

Jedem Prozess wird ein virtueller Speicher geboten, in den alles außer arbeitenden Code und Daten (bleiben im Hauptspeicher) eingelagert wird. Wird von einem Prozess auf den Speicher zugegriffen, verweist die virtuelle Adresse auf eine reelle Adresse des Hauptspeichers. Dieser Verweis geschieht durch die *MMU (Memory Management Unit)*. Befinden sich die Daten in einem ausgelagerten Festplattenblock, wird dieser in den Hauptspeicher eingelagert.
Durch den Einsatz eines Virtuellen Speichers ergeben sich folgende Vorteile:

- Programmbereiche und Datenbereiche sind in ihrer Länge nicht durch die reelle Größe des Hauptspeichers begrenzt.
- Es können gleichzeitig mehrere Programme ausgeführt werden, deren Gesamtlänge die Hauptspeichergröße überschreitet.

## 7.4 segmentorientierter Speicher

Der Adressraum wird in Segmente variabler Größe entsprechend der der logischen Einheiten des Programms unterteilt. Jedem Segment ist ein Speicherbereich im Haupt oder Festplattenspeicher zugeteilt. Die Virtuelle Adresse besteht aus zwei Teilen, der Segmentnummer und dem Offset. Dieser gibt die Position innerhalb des Segments an.




Der Prozess gibt mit der virtuellen Adresse an, auf welches seiner Segmente und auf welche Position innerhalb dieses Segmentes er zugreifen möchte. Die Umrechnung von virtuellen in reelle Adressen geschieht mit der Segmenttabelle. Jeder Prozess besitzt eine eigene Tabelle. Die Segmenttabelle eines Prozesses enthält für jedes Segment einen Eintrag mit folgenden Informationen:

- Reelle Anfangsadresse des Segments
- Valid-Bit (gibt an, ob sich das Segment auf dem Hauptspeicher oder auf der Festplatte befindet)
- Länge des Segments
- Zugriffsrechte

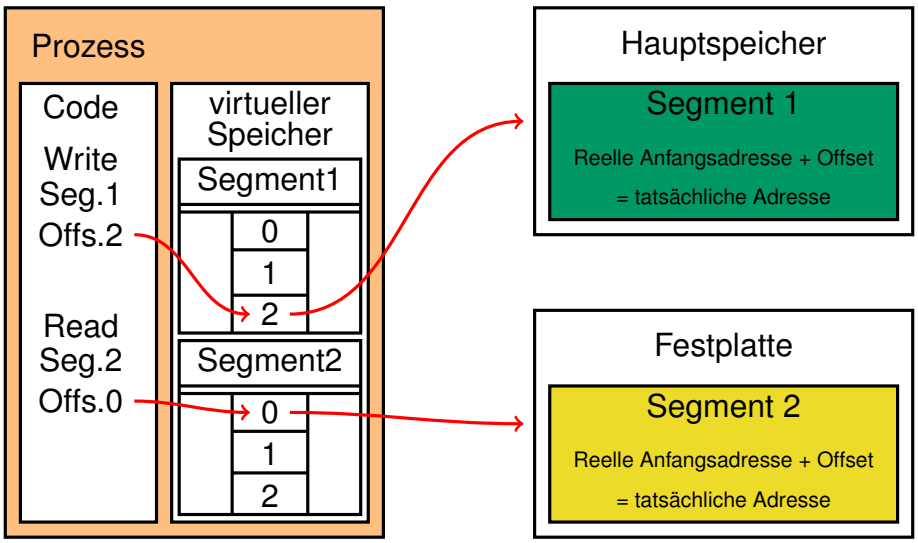
## 7.5 seitenorientierter Speicher - *Paging*

virtuelle Adresse Seitentabelle reelle Adresse

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 2KiB | | 0 | RA | | 2KiB | 2048 |
| 1 | 2KiB | | 1 | RA | | 2KiB | 4096 |
| 2 | 2KiB | | 2 | RA | | 2KiB | 6144 |

verweist

Die Seitentabelle stellt die *Schnittstelle (Interface)* zwischen virtueller und reeller Adresse dar. Die virtuelle Adresse, welche hauptsächlich zum leichteren Programmieren eingeführt




wurde, verweist auf die zugehörige Seite der Seitentabelle. Die Seitentabelle stellt demnach ein 1:1 Abbild der virtuellen Adresstabelle dar. Diese wiederum adressiert die reelle Startadresse im *Hauptspeichers (RAM)*. Demnach muss für jede reelle Adresse eine Seitenspalte existieren.
**Wichtig:** die Größe der Segmente der virtuellen und reellen Adresse sind *immer* gleichgroß! (Hier: 2kiB)

### 7.5.1 Beispielaufgaben

### 7.5.2 Buch Aufgabe 1, S.243

| Seitengröße | 0 | 4096 | 8182 | 12288 | 16384 | 20480 |
|---|---|---|---|---|---|---|
| Seite | 0 | 1 | 2 | 3 | 4 | 5 |
| Startadresse | 12288 | P3 | 32678 | 0 | 8192 | P12 |

Die Größe einer Seite beträgt 4096 Byte. Es wird nun geprüft, in welche Seite die virtuelle Anfangsadresse "reinpasst". Z.b.: die virtuelle Anfangsadresse 9000 passt nur in die Seite 2. Diese geht von 8182-12288. Der Offset wird berechnet indem man die gesamt Seitengrö-

| Seiten Nummer | virtuelle Anfangs-adresse | reelle Anfangs-adresse | Offset | reelle Adresse |
|---|---|---|---|---|
| 0 | 2000 | 12288 | 2000 | 14288 |
| 1 | 5000 | P3 | 904 | P3 + 904 |
| 2 | 9000 | 32768 | 808 | 33576 |
| 3 | 14000 | 0 | 1712 | 1712 |
| 4 | 18000 | 8192 | 1616 | 9808 |
| 5 | 21000 | P12 | 520 | P12 + 520 |

ße (0; 4096; 8192; 12288; . . . ) von der virtuellen Anfangsadresse abzieht. (Bsp.: Offset für *Offset* $9000 = 9000 - 8192 = 808$)
Die reelle Adresse wird berechnet indem man die reelle Anfangsadresse und den Offset addiert. (Bsp.: reelle Adresse für $18000 : 8192 + 1616 = 9808$)

Nun wird die reelle Anfangsadresse der Seite 2 in den Festplattenblock *P23* verschoben, dadurch wird ein Hauptspeichersegment frei. In dieses wird nun der Festplattenblock *P12* geladen. Die Tabelle ändert sich nun wie folgt:

| Seiten Nummer | virtuelle Anfangs-adresse | reelle Anfangs-adresse | Offset | reelle Adresse |
|---|---|---|---|---|
| 0 | 2000 | 12288 | 2000 | 14288 |
| 1 | 5000 | P3 | 904 | P3 + 904 |
| 2 | 9000 | P23 | 808 | P23 + 808 |
| 3 | 14000 | 0 | 1712 | 1712 |
| 4 | 18000 | 8192 | 1616 | 9808 |
| 5 | 21000 | 32768 | 520 | 33288 |

### 7.5.3 Buch Aufgabe 3, S.243

Ein Prozessor besitzt virtuelle Adressen mit *32 Bit*, physische Adressen (reell) mit *28 Bit* und Seiten mit *2 KiB*.
Wie viele Bit werden für die virtuellen und physischen (reell) Seitennummern benötigt?

*ausführliche Lösung:*
Um *2KiB* darstellen zu können, werden *11Bit* benötigt:

$$2KiB = 2048Byte = 2^{11}$$

Die gesamte Länge der virtuellen Adressen beträgt *32Bit*, wovon *11Bit* dazu verwendet werden, jede einzelne Adresse in einem Adressblock von *2KiB* zu adressieren. Die *11Bit* stellen demnach den Offset dar. Daraus ergibt sich:
Es bleiben noch *21Bit*, die dazu verwendet werden können, die Anzahl von Adressblöcken anzugeben.

$$32Bit - 11Bit = 21Bit$$

$$28Bit - 11Bit = 17Bit$$

Es ergeben sich demnach $2^{21}$ adressierbare virtuelle Speicheradressen mit je *2KiB*, und $2^{17}$ adressierbare reelle Adressen mit je *2KiB*.

### 7.5.4 Buch Aufgabe 4, S.243

virtuelle Adressen = *48Bit* reelle Adressen = *36Bit*
Hauptspeichergröße = *128MiB* Seitengröße = *4KiB*

a) $4KiB = 4096 = 2^{12}$ $12Bit = Offset$
$48Bit - 12Bit = 36Bit \longrightarrow$ virtuelle Speicheradressen = $2^{36}$
$36Bit - 12Bit = 24Bit \longrightarrow$ reelle Speicheradressen = $2^{24}$

b) $\frac{128MiB}{4KiB} = 32786 = 2^{15}$

1 Prozess = $2^{24} \longrightarrow$ große Festplattenauslagerung
verfügbarer RAM = $2^{15}$ weil RAM kleiner als Prozess

### 7.5.5 Buch Aufgabe 5, S.243

virtuelle Adresse = *32Bit* reelle Adresse = *24Bit* Seitengröße = *2KiB*

a) Wie viele Seitentabelleneinträge werden in diesem System benötigt?
*Lösung:*

*Seitentabelleneinträge = virtuelle Adressen*

$$32Bit - 2^{11} = 21Bit$$

$$21Bit = 2^{21} = \textit{Seitentabelleneinträge}$$

b) Wie groß ist jeder Seitentabelleneintrag auf das nächste volle Byte aufgerundet?
*Lösung:*

$$24Bit - 11Bit = 13Bit$$

*13Bit* aufgerundet = *16Bit*

$$16Bit = 2Byte$$

c) Wie viel Speicherplatz ist für die Seitentabelle erforderlich?
*Lösung:*

$$\frac{2^{21} \cdot 2Byte}{1024} = 4MiB$$

### 7.5.6 Buch Aufgabe 6, S.243

Bestimmen Sie die Seitentabelle mit Seitennummer und Seitenrahmen für:

virtueller Speicher = *16MiB* reeller Speicher = *4MiB* 4 Seitenrahmen

Seitenrahmen = reelle Speichergröße

$$\frac{4MiB}{4Seitenrahmen} = \textit{eine Seitenrahmengröße}$$

| Seitennummer | 0 | 1 | 2 | 3 | bis 15 |
|---|---|---|---|---|---|
| Seitenrahmen | 1MiB | 1MiB | 1MiB | 1MiB | P0-P11 |

virtueller Speicher = *2GiB* reeller Speicher = *265MiB* 16 Seitenrahmen

| Seitennummer | 0 | 1 | [...] | 15 | bis 127 |
|---|---|---|---|---|---|
| Seitenrahmen | 16MiB | 16MiB | 16MiB | 16MiB | P0-P111 |

virtueller Speicher = *2GiB* reeller Speicher = *8MiB* 16 Seitenrahmen

| Seitennummer | 0 | 1 | [...] | 15 | bis 4095 |
|---|---|---|---|---|---|
| Seitenrahmen | 0.5MiB | 0.5MiB | 0.5MiB | 0.5MiB | P0-P4079 |

### 7.5.7 Buch Aufgabe 7, S.243

virtueller Speicher = *16Bit* reeller Speicher = *8Bit* Offset = *4Bit*

Die angeforderte virtuelle Adresse lautet: 1000 0011 0000 1111

Die Seitentabelle verweist für die Seite 1000 0011 0000 auf den Seitenrahmen 0101

a)
***Gegebenheiten:***

virtueller Adress Rahmen = Seitenrahmen (SR) = reeller Adress Rahmen

virtuelle Adressen = Seiteneinträge

| virtueller Speicher = VS<br>virtuelle Adresse = VA | ges. Seitengröße = SG<br>Seitenadressen = SA | reeller Speicher = RS<br>reelle Adresse = RA |
|---|---|---|
| *16Bit-4Bit* Offset = *12Bit* VA<br>Offset $2^4 = 16Byte = VARahmen$<br>$\frac{16 \cdot 2^{12}}{1024} = 64KByte = VS$ | Offset $2^4 = 16Byte = SR$<br>$\frac{16 \cdot 2^{12}}{1024} = 64KByte = SG$ | *8Bit-4Bit* Offset = *4Bit* RA<br>Offset $2^4 = 16Byte = RARahmen$<br>$2^4 \cdot 16Byte = 265Byte = RS$ |

b)

| Frame Number = 0101 | Offset der Adresse = 1111 |
|---|---|

**Die reelle Adresse lautet demnach: 0101 1111**

## 7.6 Speicherverwaltungsmethodenchronologie

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| direkte Speicherverwaltung | Swapping | seitenorientierte Speicherverwaltung | segmentorientierte Speicherverwaltung |

# 8 Datensicherung

Ein Backup dient als Schutz vor:

- Schadsoftware z.B. Viren
- versehentliches Datenüberschreiben oder Löschen
- Hardware Schäden
- logische Fehler in einer Datei
- Diebstahl

## 8.1 Sicherungsmedien

Bekannte Sicherungsmedien sind:

- DVD's
- USB - Sticks
- DLT (Digital Line Tape)
- DAT Laufwerk

Wichtig bei Sicherungsmedien ist, dass die Backup Zeit so klein wie möglich bleibt! Eine Formel zur Berechnung der Backup Zeit wäre:

$$t_b = \frac{m}{r_b}$$

$t_b$ = Backup Zeit $m$ = Datenmenge $r_b$ = Schreibrate in Bit/s

## 8.2 Datenverlust

Durch falsche Lagerung, äußere Gewalteinwirkung, aussetzen von bestimmten Strahlungen (Magnetfeld), kann es zu Schäden der Dateien auf dem Datenträger kommen. Auch bei versehentlichem Löschen einer Datei kann diese noch wiederhergestellt werden. Beim Löschen oder Formatieren werden Dateien nicht wirklich *vernichtet*, generell werden sie nur ausgeblendet bzw. Verzeichniseinträge werden gelöscht. Man unterscheidet zwischen drei *Recovery Verfahren:*

**1. Einfache Rettung:**
Eine Datei wurde versehentlich gelöscht, der Eintrag über die gelöschte Datei im Inhaltsverzeichnis des Datenträgers ist noch vorhanden. Die Datei kann mit Hilfe einer *Recovery Software* wiederhergestellt werden.

**2. Aufwendige Rettung:**
Der Eintrag im Verzeichnis ist nicht mehr vorhanden. Die Datei muss mithilfe spezieller Dateisignaturen, die den Dateityp, den Dateianfang und das Dateiende enthalten, erkannt und wiederhergestellt werden.

**3. Schwierige Rettung:**
Ist die Datei darüber hinaus auch noch fragmentiere, erschwert das die Rettung erheblich. Die Recovery Software muss die Inhalte der Datenfragmente analysieren und sie dann zusammenfügen. Mit zusätzlicher Software kann der Datenverlust durch Erkennung von Verschleißerscheinungen vermieden werden.

## 8.3 Datenvernichtung

Es ist ratsam sensible Daten vor der Entsorgung eines Datenträgers durch verschiedene Maßnahmen annährend vollständig zu vernichten (es besteht keine 100% -ige Garantie, dass die Daten nicht wiederhergestellt werden können).
Angemessene Maßnahmen können:

- Säurebad
- zusätzliche Fragmentierung
- Überschreiben mit *Zufallsdaten oder Bitmustern*
- Hardwarezerstörung

sein.

## 8.4 Sicherungsverfahren

Abhängig vom Maschinentyp, Dateisensibilität und Anwendungsbereich können verschiedene Verfahren angewendet werden.

### 8.4.1 Vollständige Sicherung

Alle Daten werden auf einem Backup-Medium z.B. externer Festplatte gesichert.

| Vorteile | Nachteile |
|---|---|
| Alle Daten liegen komplett vor. Keine lange Suche bei der Wiederherstellung. Zur Wiederherstellung verlorener Dateien wird nur die letzte Vollsicherung benötigt. | überflüssige Kopiervorgänge, Backups brauchen viel Zeit, hohe Datenmengen |

$\longrightarrow$ Jeden Tag wird eine Vollsicherung durchgeführt.

### 8.4.2 Differenzielle Sicherung

Es werden die Daten gespeichert, die seit der letzten Vollsicherung erstellt oder verändert wurden. Praxisbezug: Es wird Täglich die Datenmenge gesichert, welche seit der letzten Vollsicherung erstellt oder verändert wurde.

| Vorteile | Nachteile |
|---|---|
| Die Wiederherstellung ist unkomplizierter und schnell | Es wird mehr Zeit und Speicherplatz auf dem Sicherungsmedium im Vergleich zur *Inkrementellen Sicherung* benötigt. |

$\longrightarrow$ Eine Vollsicherung wird einmal pro Woche durchgeführt. Die einzelnen Tage werden nicht unabhängig gesichert. Das Backup vom Mittwoch enthält die Sicherungen von Montag, Dienstag und Mittwoch.

### 8.4.3 Inkrementelle Sicherung

*Zuwachssicherung*. Nur die Daten, die seit dem letzten Backup erstellt oder geändert wurden werden gesichert. Praxisbezug: Es wird Täglich nur die Datenmenge gesichert, welche erstellt oder verändert wurde.
$\longrightarrow$ Eine Vollsicherung wird einmal pro Woche durchgeführt. Die einzelnen Tage werden unabhängig voneinander gesichert. Das Backup vom Mittwoch enthält nur den Tag Mittwoch.

| Vorteile | Nachteile |
| --- | --- |
| Schnelle Datensicherung und wenig Speicherplatzbedarf auf dem Sicherungsmedium | Hoher zeitlicher Aufwand bei der Wiederherstellung, da die letzte Vollsicherung und darauffolgende Zuwachssicherungen benötigt werden. |

### 8.4.4 Sicherungsstrategie: Generationen Konzept

Die einzelnen Backups können nach dem *Generationen Konzept* logisch geordnet werden:

**Söhne**
Die am einzelnen Tag ausgeführten Backups werden als Söhne bezeichnet.

**Väter**
Die Vollsicherungen (einmal pro Woche) werden als Väter bezeichnet.

**Großväter**
Die Vollsicherungen im Zeitraum eines Monates werden als Großväter zusammengefasst.

## 8.5 Vernetzte Sicherung

Vor allem in Firmen arbeiten die Mitarbeiter, jeder eigenständig, an Computern. Um die entstandenen Datenmengen zu sichern, werden Sicherungen meist über ein Netzwerk durchgeführt. Dafür können folgende Strategien angewendet werden:

### 8.5.1 verteilte Sicherung

Jeder Rechner ist mit einem eigenen Sicherungsmedium im Netzwerk (Backup Server) ausgestattet. Jeder Rechner wird einzeln gesichert, es kann aber über das Netzwerk von jedem Rechner aus auf die Backups jedes anderen zugegriffen werden. Alle Backups sind voneinander digital getrennt.

### 8.5.2 lokale Sicherung

Jeder Rechner ist mit einem eigenen Sicherungsmedium, unabhängig von einem Netzwerk ausgestattet. Es kann nicht über das Netzwerk auf Backups anderer Rechner zugegriffen werden. Alle Backups sind voneinander physisch getrennt.

### 8.5.3 Netzwerk Sicherung

Alle Rechner besitzen eine Anknüpfung zum Netzwerk, und erstellen ihre Backups zusammen auf einem Backup Server. Die Effizienz der Backup Medien steigt, da alle Rechner auf diese sichern können. Es kann zu Engpässen oder Netzwerküberlastung kommen, wenn z.B. alle Rechner gleichzeitig hohe Datenmengen sichern wollen. Alle Daten sind weder digital noch physisch getrennt, da sie alle auf das selbe Sicherungsmedium (Backup Server) gesichert werden. Der Backup Server ist jedoch physisch von den einzelnen Rechnern getrennt, was die Sicherheit erhöht.

### 8.5.4 Kapazität des Netzwerks

Die Transportkapazität des Netzwerkes stellt bei der Netzwerk und lokalen Sicherung einen limitierenden Faktor dar. Die Backup Zeit $t$ lässt sich mit folgender Formel berechnen:

$$t = \frac{m}{G}$$

$m$= Datenmenge $G$= Übertragungsgeschwindigkeit $\frac{Mbit}{s}$

### 8.5.5 Logfiles

Alle bisher genannten Verfahren haben den Nachteil, dass bei z.B. Virusbefall auf das letzte vorhandene Backup zurückgegriffen werden muss, und somit Transaktionsdaten das laufenden Tages verloren gehen. Um dies zu verhindern ist es ratsam eine *Logfile* zu führen, in der jede Änderung an Dateien gespeichert wird. Die *Logfile* enthält alle Datenänderungen seit dem letzten Backup.

## 8.6 RAID Systeme

*RAID* = Redundant Array of Independent Disks
ist ein Sicherungsmedien Verband, mit dem folgende Ziele verfolgt werden können:

- Erhöhung der Ausfallsicherung durch Redundanz
- Steigerung der Transferraten
- Austausch von Sicherungsmedien während des Systembetriebs.

### 8.6.1 RAID 0

Eine zu sichernde Datei wird in verschiedene Segmente aufgeteilt. Jedes Segment wird auf einem seperaten Sicherungsmedium gesichert. Dadurch werden hohe Transferraten durch Aufteilung der Datenmenge auf verschiedene Sicherungsmedien erreicht. Es besteht keine Datensicherheit. Geht ein Sicherungsmedium kaputt, Sind alle Dateien im RAID verband verloren, da jede Datei in Segmente unterteilt wird, und somit ein Segment von jeder Datei auf jedem Sicherungsmedium liegt.

### 8.6.2 RAID 1

Bei diesem Verfahren werden alle Daten doppelt abgelegt. Geht ein Sicherungsmedium kaputt, sind alle Dateien auf einem zweiten im RAID System verknüpften Sicherungsmedium verfügbar. Kaputte platten können einfach ausgetauscht werden. Es kann nur 50% der Gesamtkapazität der im RAID Verband verknüpften Sicherungsmedien genutzt werden, da alles doppelt vorhanden ist. Es besteht kein Performance Gewinn.

### 8.6.3 RAID 5

Es werden mindestens drei Sicherungsmedien benötigt. Dieses Verfahren unterscheidet sich von RAID 0 in nur einem Punkt. Es werden zusätzlich Paritäten ermittelt, und auf einem Sicherungsmedium gespeichert, um bei einem Ausfall Dateien vollständig zu rekonstruieren.
Die nutzbare Sicherungskapazität wird wie folgt berechnet:

$$K_5 = s \cdot (n-1)$$

$K_5$= nutzbare Kapazität $s$= kleinste Platte im Array $n$ Anzahl der Platten

### 8.6.4 RAID 10

verbindet RAID 0 und 1. Im Prinzip funktioniert dieses System wie RAID 0, nur dass alle Dateien zusätzlich doppelt vorhanden sind. Durch dieses System erhält man eine hohe Datensicherheit, und gute Performance.

### 8.6.5 RAID 50

Kombination aus RAID 5 und 0. Die Performance im Vergleich zu RAID 10 steigt nicht. Es werden mindestens 6 Sicherungsmedien benötigt.
Die nutzbare Sicherungskapazität wird wie folgt berechnet:

$$K_{50} = s \cdot \left(\frac{n}{2} - 1\right)$$

$K_{50}$= nutzbare Kapazität $s$= kleinste Platte im Array $n$ Anzahl der Platten

### 8.6.6 Gegenüberstellung der RAID Systeme

| **System** | **Kenngrößen** |
|---|---|
| RAID 0 | hoher Datendurchsatz,<br>100% Kapazität nutzbar,<br>Keine Sicherheit |
| RAID 1 | Hohe Sicherheit,<br>kein Performance Gewinn,<br>50% der Kapazität nutzbar |
| RAID 5 | Sicherheit,<br>Performance Gewinn,<br>>50% Kapazität nutzbar |

### 8.6.7 RAM vs ROM

Im Computer sind sowohl RAM, das für Random Access Memory steht, als auch ROM, das für Read-Only Memory steht, vorhanden. RAM ist ein flüchtiger Speicher, der die Dateien, an denen Sie arbeiten, vorübergehend speichert. ROM ist ein nichtflüchtiger Speicher, der Anweisungen für Ihren Computer dauerhaft speichert.

# 9 Netze

## 9.1 Einführung

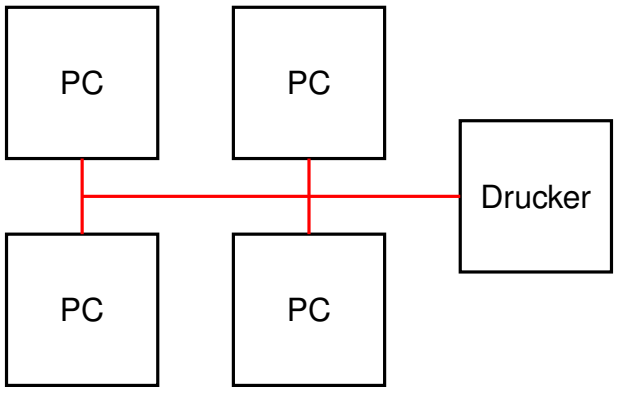
### 9.1.1 Beispiele für Netze

- Verkehrsnetz
- Smart Home
- Mobilfunknetz
- Stromnetz
- Soziale Netze (Familie, Freunde etc. . . )

### 9.1.2 Gemeinsamkeiten aller Netze

- Verbinden Dinge, Sachen, Knoten (Connection)
- Es bestehen immer Protokolle / Regeln (StVO)
- Dienen zum Informationsaustausch / Transport von Informationen

## 9.2 Einfaches Computernetz




Die Vernetzung verschiedener Systeme ermöglicht eine Verbesserung der Leistungsfähigkeit und reduziert Kosten. Computernetze erzielen diese Gegebenheiten hauptsächlich auf drei Arten:

- Freigeben von Informationen / Daten
- Freigeben von Hard- und Software
- Zentralisieren von Verwaltung und Unterstützung

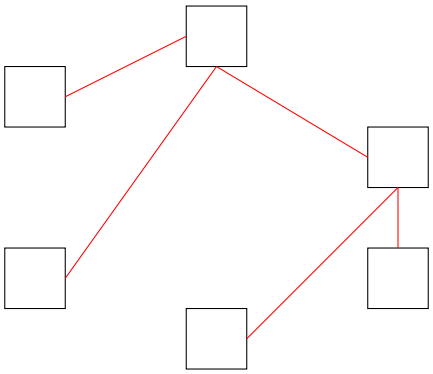
Spezifisch können Endgeräte eines Netzes folgende Komponenten freigeben:

- Betriebsmittel: Drucker, Maus, Monitor
- Sicherungsmedien: Festplatte, Laufwerke

- Sich auf diesen Sicherungsmedien befindende Daten
- Software

### 9.2.1 Netz Topologien

#### Baum Topologie



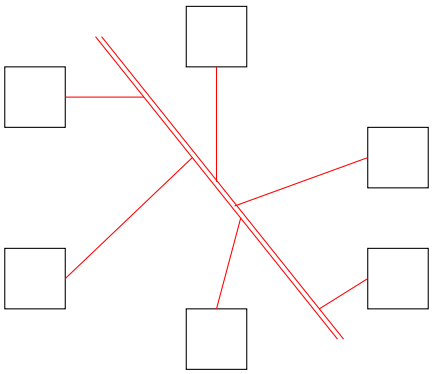
Vorteile:

- Bei Ausfall einer Komponente bricht nur ein Teil des Netzes zusammen
- leichte Skalierbarkeit

Nachteile:

- Durchsatzproblem an der Wurzel / jeder Netzkomponente $\longrightarrow$ höhere Laufzeiten

#### Bus-Topologie



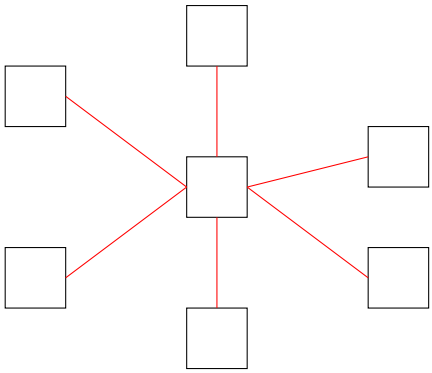
Vorteile:

- hohe Ausfallsicherheit
- leichte Skalierbarkeit

Nachteile:

- Ausfall der Hauptleitung $\longrightarrow$ Totalausfall
- Hauptleitung benötigt hohe Bandbreite

#### Stern Topologie



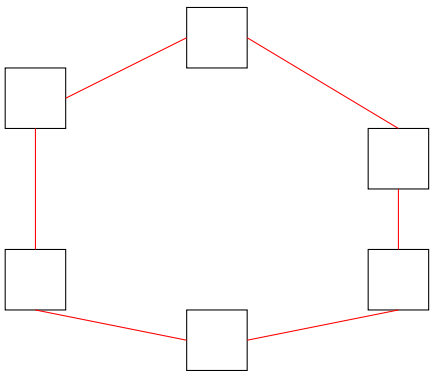
Vorteile:

- Leicht umsetzbar / skalierbar
- sehr schnell
- Beim Ausfall einer Komponente ist *nur* diese betroffen

Nachteile:

- Beim Ausfall der zentralen Wurzel $\longrightarrow$ Totalausfall
- Clients müssen immer über zentrale Wurzel kommunizieren.

#### Ring-Tropologie



Vorteile:

- simpler Aufbau
- leichte Skalierbarkeit

Nachteile:

- Unterbrechung des Rings $\longrightarrow$ Totalausfall

### 9.2.2 Netze in Unternehmen

Das Verwenden eines Netzes hat signifikante Vorteile für ein Unternehmen / Betrieb. Ressourcen und Betriebsmittel können gemeinsam genutzt werden (Zentral-Drucker). Dadurch können Kosten eingespart werden. Die Kommunikation innerhalb des Unternehmens wird durch verschiedene Netze (internes Telefon, Kommunikation zwischen den Systemen) um ein vielfaches erleichtert. Sollte das Unternehmen im Laufe der Zeit an Größe zulegen, ist die Skalierbarkeit des Systems im Allgemeinen unbegrenzt. Zusätzliche Systeme oder Server können problemlos hinzugefügt werden. Die Systemleistung eines einzelnen Computers kann außerdem gesteigert werden. Aufwendige Anwendungen können auf einem leistungsstarken Server ausgeführt werden, sind also nicht an die locale Leistung eines einzelnen Computers gebunden. Die Komponente der Teamarbeit im globalen Sinne wird durch ein globales Netzwerk zwischen Unternehmen außerdem vereinfacht.

### 9.2.3 Private Netze

Besitzt ein Haushalt einen Home Server mit Internetanbindung, so ist es Privatpersonen möglich, von nahezu überall auf der Welt auf ihre persönlichen Daten zugreifen zu können. Diese Option steht Privatpersonen außerdem durch Cloud Services zur Verfügung. Nachteil: Die Daten werden externen Firmen zur Verfügung gestellt, diese Option ist mit Vorsicht zu genießen. Privatpersonen können durch das Internet außerdem auf, generell betrachtet, Online Services zugreifen (Online Banking, Online Shopping). Die Kommunikation wird auch für Privatpersonen erheblich erleichtert. Durch Chat, Voice Chat oder sogar Video Chat Anwendungen ist die Kommunikation nahezu an jedem Ort der Erde möglich. Auch das Unterhaltungsprogramm profitiert vom Internet. Dienste wie YouTube oder Netflix wären ohne das Internet undenkbar.

## 9.3 Servermodelle

### 9.3.1 Ein-Server-Modell

Ein Computer / Server übernimmt alle zentralen Dienste (Standard)

### 9.3.2 Multi-Server-Modell

Mehrere Computer / Server teilen sich die Verwaltung (für große Netze)

## 9.4 Kommunikationsarten

### 9.4.1 Simplex

Einer spricht, der Rest hört zu

### 9.4.2 Halbduplex

Wechselseitiges Sprechen und Hören

### 9.4.3 Vollduplex

Jeder spricht und hört gleichzeitig

## 9.5 Netzwerkkomponenten

### 9.5.1 Port

Ein Port ist der Teil einer Netzwerk-Adresse, der die Zuordnung von TCP- und UDP-Verbindungen und -Datenpaketen zu Server- und Client-Programmen durch Betriebssysteme bewirkt. Zu jeder Verbindung dieser beiden Protokolle gehören zwei Ports, je einer auf Seiten des Clients und des Servers.

### 9.5.2 Client

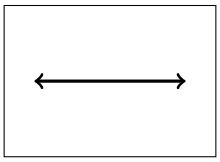
Der Client stellt einen Rechner im Netz dar. Er simuliert einen Computer an einem gewissen Standpunkt. Der Client besitzt eine Network-Interface-Card.


### 9.5.3 Netzwerkkarte

Eine Netzwerkkarte besitzt eine individuelle MAC-Adresse und befindet sich in eigentlich jedem Rechner. Sie stellt die Verbindung zwischen dem Rechner und dem Netzwerkmedium dar. Außerdem verfügt sie über eine Netzwerk-Schnittstelle.

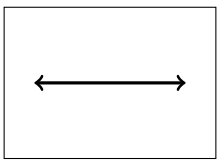
### 9.5.4 Repeater

Der Repeater hat jeweils einen Eingang und einen Ausgang. Er verstärkt Signale durch regenerieren/synchronisieren des Taktes. Dazu arbeitet er auf Bitebene und verursacht eine kleine Latenz.



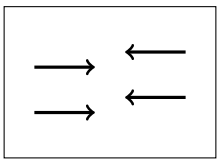
### 9.5.5 HUB

Der HUB ist ein Multiport Repeater. D.h. er hat mehrere Ein- und Ausgänge und arbeitet auch auf Bitebene.



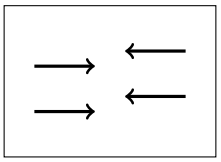
### 9.5.6 Bridge

Die Bridge fungiert als Brücke zwischen zwei LAN-Netzwerken. Sie kennt die relevanten MAC-Adressen in beiden Netzen und verbindet so die Netze miteinander.



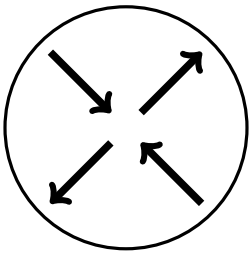
### 9.5.7 Switch

Der Switch ist eine Multiport-Bridge. D.h. er hat mehrere Ein- und Ausgänge und verbindet so mehrere Netzwerke miteinander. Er verwendet Filtertabellen um den schnellsten Weg für Signale zu finden. Er arbeitet ebenso mit MAC-Adressen.



### 9.5.8 Router

Der Router verbindet Netzwerke indem er Datenpackete weiterleitet oder blockiert. Er arbeitet mit IP-Adressen und wird oft auch als intelligenter Switch bezeichnet. Der Router wird für das gesamte Netz verwendet.



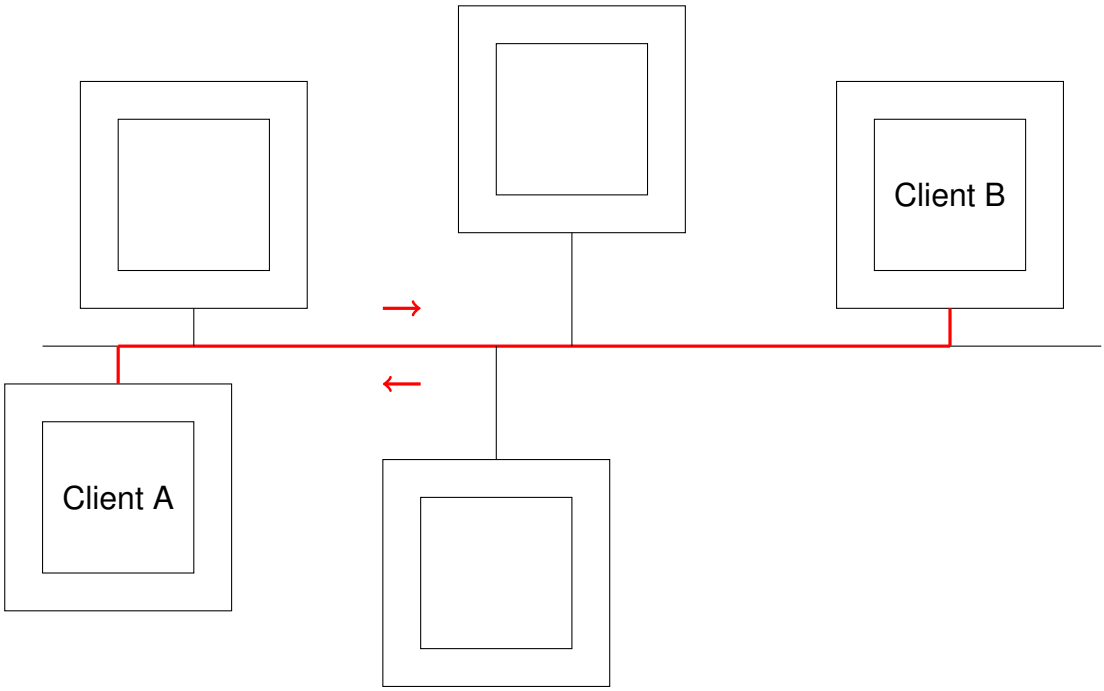
### 9.5.9 Einordnung der Komponenten ins OSI-Modell

Die Netzwerkkomponenten lassen sich in die Schichten 1 bis 4 einordnen. Die Schicht 1 beinhaltet alle Kabel(Kupferkabel, Ethernetkabel, Glasfaserkabel) und den Repeater. Zu Schicht 2 gehören Switches und Bridges. In Schicht 3 ist der Router und in Schicht 4 handelt es sich um die Netzwerkkarte und die Ports.

| Nummer | Bezeichnung | Komponenten |
|---|---|---|
| 1 | Bitübertragungsschicht | Repeater, Kabel |
| 2 | Sicherungsschicht | Switches, Bridges |
| 3 | Vermittlungsschicht | Router |
| 4 | Transportschicht | Netzwerkkarte, Ports |

## 9.6 Round Trip Delay Time - RTDT

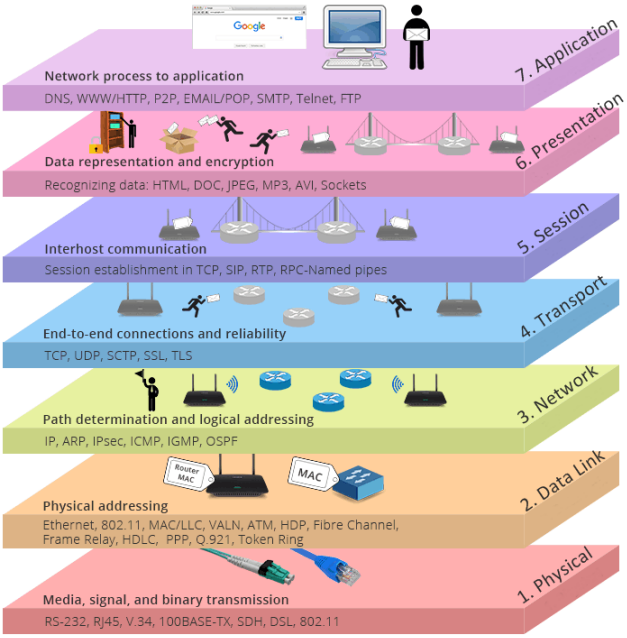
Die Paketumlaufzeit bzw. Round Trip Time gibt die Zeit an, die ein Datenpaket in einem Rechnernetz benötigt, um von der weitesten entfernten Quelle zum Ziel und zurück zu reisen. Es handelt sich also um die Summe aus Laufzeit von Punkt A nach Punkt B und der Laufzeit von Punkt B nach Punkt A.




## 9.7 Bezeichnungen von Netzen

## 9.8 OSI 7-Schichten Modell

Zweck des OSI-Modells ist, Kommunikation über unterschiedlichste technische Systeme hinweg zu beschreiben und die Weiterentwicklung zu begünstigen.




**Hauptaufgaben der Schichten:**

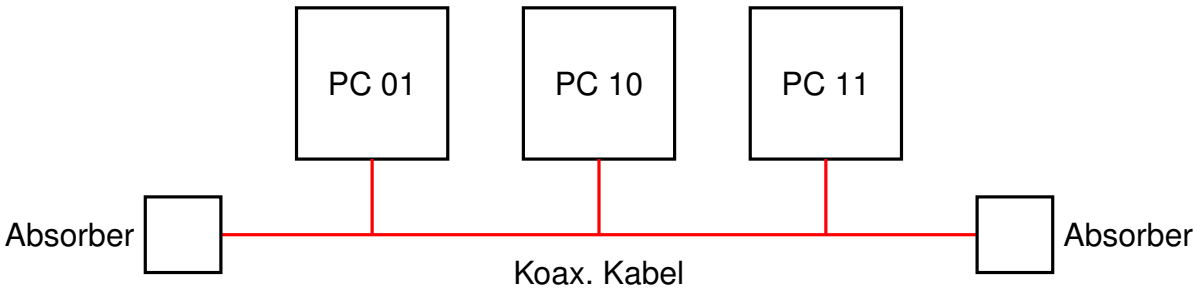
- Schicht 7: Anwendungen für Benutzer
- Schicht 6: Darstellung der Daten in verständliche Formate (jpg, ASCII)
- Schicht 5: Steuerung der Verbindung
- Schicht 4: Zuordnung der Datenpakete zu den Ports
- Schicht 3: Vermittelt Datenpakete
- Schicht 2: Fehlerfreie Übertragung
- Schicht 1: Bit-Übertragung

Tabelle 3: Bezeichnungen von Netzen

| Abkürzung | Beschreibung | Kabellänge | Anwendungen |
|---|---|---|---|
| CAN | Control Area Network | 5km | Steuerungstechnik |
| PAN | Personal Area Network | 100m | Handynetz |
| LAN | Local Area Network | 100m | Home Netz |
| MAN | Metropolitan Area Network | 100km | Unternehmen mit mehreren Standorten |
| WAN | Worldwide Area Network | Weltweit | Internet, Telefon |


# 10 Ethernet

Mit der Erfindung des Ethernets wollte man individuellen Stationen / Systemen den Zugriff auf ein gemeinsames Medium zur Datenübertragung ermöglichen. Dieses gemeinsame Medium wurde durch ein Koaxial Kabel realisiert.
Soll ein Datenpaket verschickt werden (z.B. ein Zahlenwert), so muss der Sender das Ziel des Paketes kennen, um die Information zum korrekten Empfänger schicken zu können.
Will *PC 01* die Zahl *5d* an *PC 10* senden, ergibt sich folgender Rahmen:




| | |
|---|---|
| 0010 *(Kennung des Ziels, 4Bit)* | 0101 *(Daten, 4Bit)* |

Damit der Empfänger den Erhalt der Daten quittieren kann, muss jedoch auch die Kennung des Senders im Rahmen enthalten sein:

| | | |
|---|---|---|
| 0010 *(Kennung des Ziels)* | 0001 *(Kennung der Quelle)* | 0101 *(Daten)* |

Der verschickte Rahmen lautet demnach:

0010 0001 0101

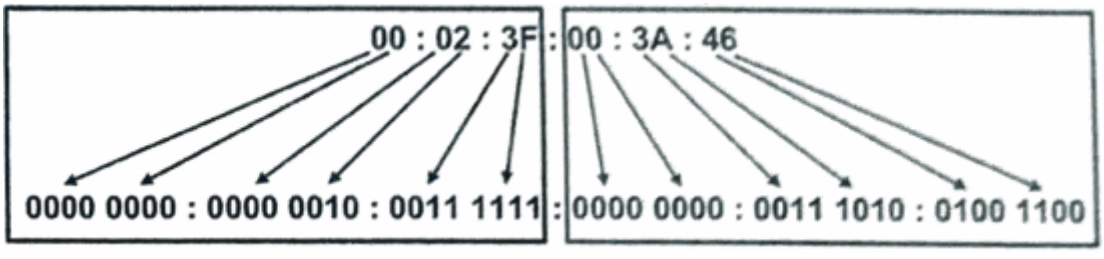
Um zu wissen, ob es sich bei den empfangenen Impulsen tatsächlich um eine gesendete Information handelt, wird ein Datenrahmen immer durch eine Präambel angekündigt. Es könnten sonst Missverständnisse durch Signalrauschen oder Störungen auftreten. Die endgültige Struktur eines Datenrahmens:

| | | | | |
|---|---|---|---|---|
| Präambel (7Byte) | SFD (1Byte) | Ziel (MAC) | Quelle (MAC) | Nutzdaten |

## 10.1 MAC Adresse

Um Informationen im Ether / Internet zuverlässig und zielgenau verschicken zu können, muss jedes Endgerät im Netz seine eigene individuelle Kennung besitzen!
$\longrightarrow$ *MAC Adresse*
Die MAC Adresse ist in den ROM der Network Interface Card eines jeden Gerätes eingebrannt. Die MAC ist also keine virtuelle Softwarekennung, sondern eine durchaus physisch mit dem Gerät verbundene Kennnummer.

### 10.1.1 Aufbau MAC Adresse




3 Byte OUI (Herstellerkennung) 3Byte Gerät-Seriennummer

# 11 Netzwerkprotokolle

Hat die Netzwerkkarte einen Datenrahmen empfangen, muss ermittelt werden, welches Protokoll zur Weiterverarbeitung verwendet werden soll. Innerhalb des Datenrahmens muss also der Typ und das Ziel der Daten festgelegt sein. Diese Informationen stehen in einem 2Byte großem Typenfeld. Für jedes Protokoll existiert eine eigene Kennung:

| | |
|---|---|
| ARP | 0x0806 |
| IPv4 | 0x0800 |
| IPv6 | 0x86DD |

Allerdings kommt dem Typenfeld eine weitere Bedeutung:

| | |
|---|---|
| Wert kleiner als 0x0600 | Länge des Datenrahmens |
| Wert größer als 0x0600 | Protokollkennung |

Alle Protokollkennungen müssen also größer als 1536d oder 0x0600h sein!

**Achtung:**
Das ICMP Netzwerkprotokoll ist ein Protokoll der IP-Familie und folgt somit dem IPv4-Protokoll!
Ebenso gehört das ICMPv6 Netzwerkprotokoll zur IPv6 Familie und folgt somit auch dem IPv6 Protokoll!

## 11.1 Datenübertragung

Um sicher zu gehen, dass alle Informationen fehlerfrei übertragen wurden, wird dem Datenrahmen ein 4Byte großes Prüffeld (CRC) angehängt. Dieses Prüffeld ergibt sich aus einer Polynomdivision.

| Präambel | SFD | Ziel-Mac | Quell-MAC | Typ | Nutzdaten | CRC |
|---|---|---|---|---|---|---|

Min: 64Byte - Max: 1518Byte

Unterschreitet der Anteil der Nutzdaten 46Byte, wird durch Padding aufgefüllt.

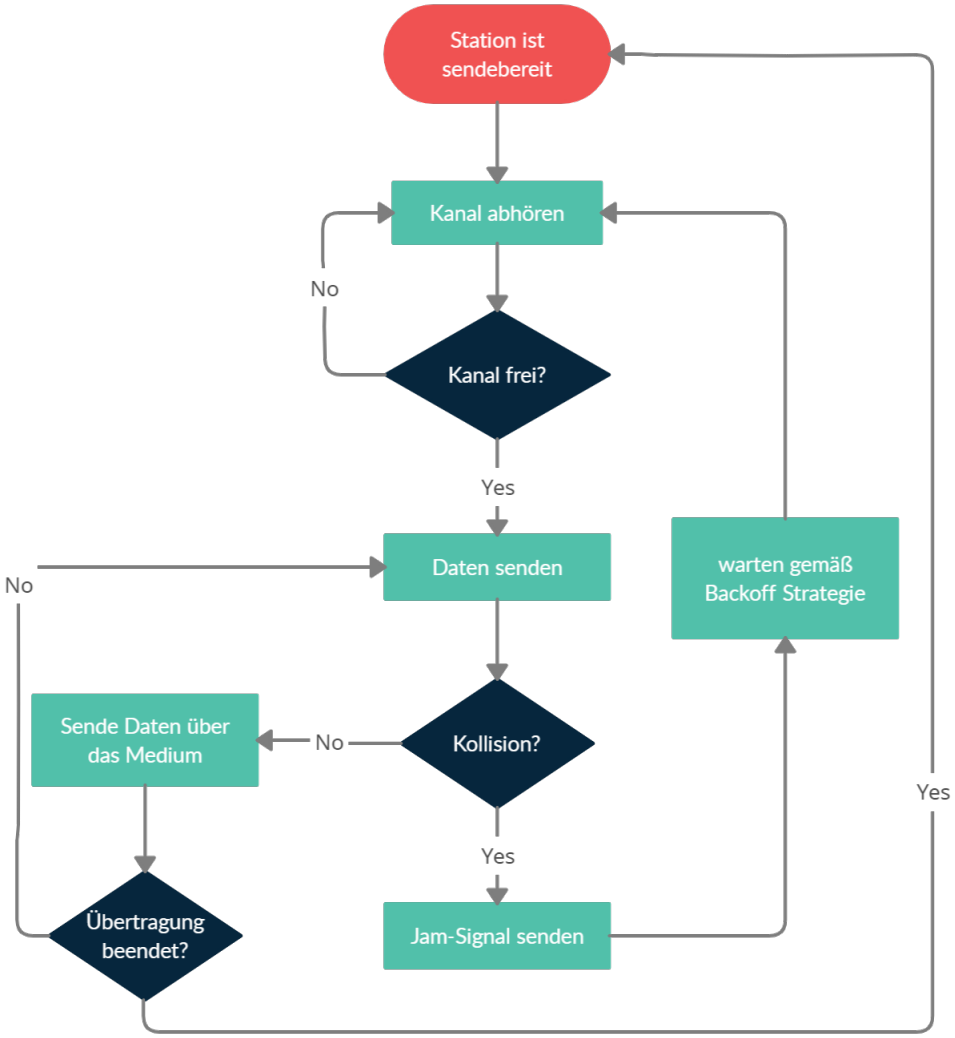
# 12 Vollständiges Ethernet-Paket

| Präambel 7 | SFD 1 | Ziel-Mac 6 | Quell-MAC 6 | Typ 2 | Nutzdaten 46-1500 | CRC 4 |
|---|---|---|---|---|---|---|

Zahlenwert im Feld = Bytegröße des Feldes.

# 13 CSMA/CD - Verfahren

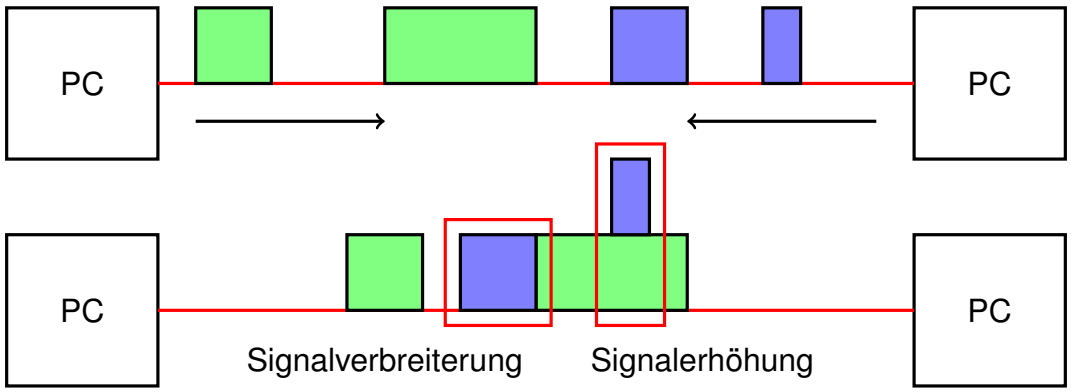
Programmablaufplan des CSMA/CD Verfahrens:




## 13.1 Kollision

**Problem:**
Da Alle Systeme mit nur einem Übertragungsmedium vernetzt sind, um Materialkosten zu sparen, und Skalierbarkeit zu erhalten, muss der Leiter bidirektional verwendet werden. Dadurch kann es zu Kollisionen von Impulsen kommen.
$\longrightarrow$ Es kommt zu einer Signalerhöhung / Signalverbreiterung und damit zu einer fehlerhaften Übertragung.

**Lösung:**
CSMA/CD Verfahren - Jam Signal **Situation:**




Das sendende System vergleicht ständig das gesendete Signal mit dem Signal auf der Leitung. Kommt es zu Abweichungen bricht das System die Übertragung ab und sendet ein Jam-Signal welches andere Systeme im Netz über die Kollision informiert. Alle Systeme unterbrechen die Übertragung.
Ab wann dürfen die einzelnen Systeme wieder anfangen Daten zu senden?

Jedes Signal auf der Leitung braucht eine bestimmte Zeit, um zwischen den beiden entferntesten Systemen im Netz einmal hin, und wieder zurück zu laufen. Diese Zeit wird mit RTT bezeichnet. Ist die RTT abgelaufen, befinden sich keine Signale mehr im Netz, es kann neu gesendet werden.

Nach dem Ethernet-Standard 802.3 ist die RTT auf 51,2 Mikrosekunden festgelegt.

Kommt es nach dem Warten der RTT dennoch zu einer weiteren Kollision, variieren die Systeme ihre Wartezeit indem sie ein vielfaches der RTT warten. Als Vielfaches können die Faktoren 0,1,2 und 3 gewählt werden.
$\longrightarrow$ Es wird unterschiedlich lang gewartet.

Kommt es erneut zu einer Kollision, wird der Bereich der Vorfaktoren von 0 bis 7 erweitert.

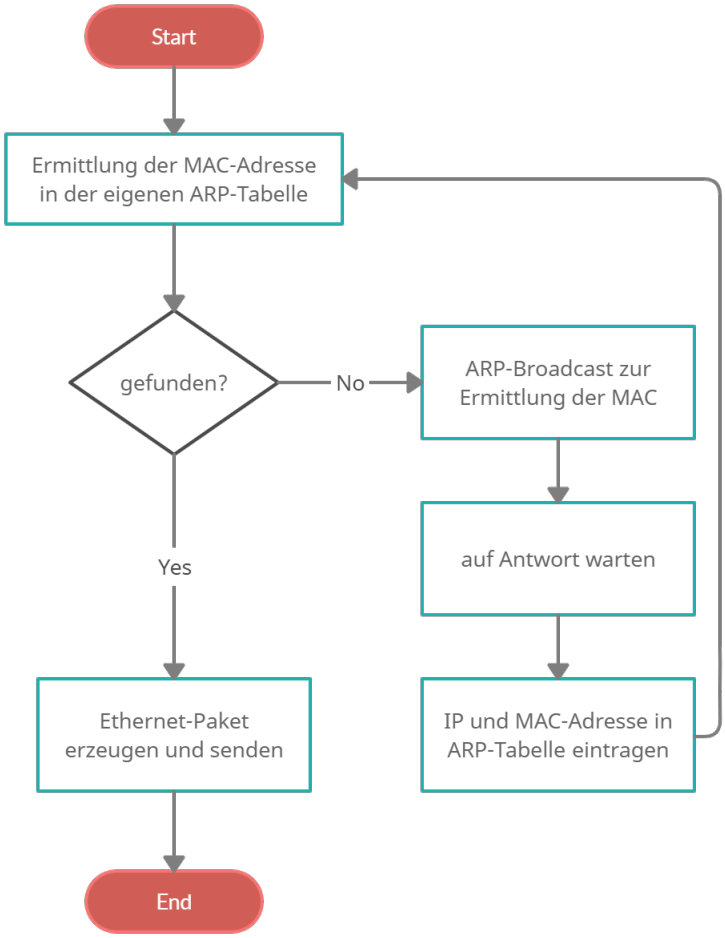
$$\text{Wartezeit} = k \cdot RTD$$

k = 0 bis $2^i$-1 i = Anzahl Versuche

Nach 10 Versuchen wird i nicht mehr erhöht, nach 16 erfolglosen Versuchen wird der Sendeversuch abgebrochen.

# 14 ARP-Protokoll

Programmablaufplan des ARP Verfahrens:




# 15 Subnetting

Subnetting ermöglicht es Netzwerkadministratoren beispielsweise, das eigene Firmennetzwerk in Subnetze aufzuteilen, ohne dies im Internet bekannt zu machen. Das heißt, der Router, der schließlich das Netzwerk mit dem Internet verbindet, wird weiterhin als einfache Adresse angegeben.

Alle Subnetze eines Netzes funktionieren unabhängig voneinander und die Datenvermittlung läuft schneller. Warum ist das so? Subnetting macht das Netzwerk überschaubarer. Ein Broadcast, bei dem ein Teilnehmer Daten an das gesamte Netz sendet, verläuft ohne Ordnung durch Subnetze relativ unkontrolliert.

Durch Subnets werden Datenpakete durch den Router viel gezielter an die Empfänger geleitet. Befinden sich Sender und Empfänger im gleichen Subnetz, können die Informationen direkt zugestellt und müssen nicht umgeleitet werden.

## 15.1 Die IP Adresse

Bei dem Netzwerkprotokoll IPv4 (heute aktuell: IPv6) besteht eine IP-Adresse aus 32 Bit. Diese sind in vier Abschnitte mit je einem Oktett aufgeteilt.
Beispiel IP Adresse:

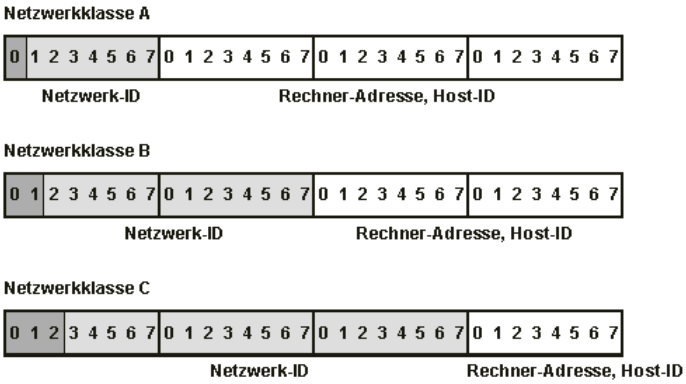
| | |
|---|---|
| dezimal | 192.168.0.1 |
| binär | 11000000.10101000.0000000.00000001 |

Zu beachten ist dabei, dass jedes Oktett als eigenständig bei der Berechnung der Wertigkeit angesehen wird.
Der höchste darstellbare Wert eines Oktettes beläuft sich demnach auf 255.

| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|
| 128 | 46 | 32 | 16 | 8 | 4 | 2 | 1 |

## 15.2 Netzwerkklassen

Eine IP Adresse ist immer einem bestimmten System, welches sich in einem bestimmten Netz befindet zuzuordnen. Demnach besitzt eine IP Adresse einen Netzwerk und einen Hostbereich. Die IP Adresse muss dazu nicht in der Mitte geteilt sein (2 Oktette Netzwerk, 2 Oktette Hostbereich), sondern kann dazu beliebig zwischen jedem Bit geteilt werden. Netzen, mit 1 Oktett, 2 Oktetten und 3 Oktetten Netzwerkbereich wurden besondere Bezeichnungen zugewiesen:




## 15.3 Subnetzmaske

Um ein bestehendes Computernetz zur besseren Übersicht und Verwaltung in mehrere kleine Netze aufteilen zu können, bedarf es einer Subnetzmaske. Diese gibt den Netzwerkbereich einer IP Adresse an. Mit Hilfe der Subnetzmaske kann außerdem überprüft werden, ob sich zwei Geräte im gleichen Subnetz befinden oder nicht. Verknüpft man IP Adresse und Subnetzmaske durch eine AND Verknüpfung, erhält man die Netzwerkkennung des Subnetzes, indem sich die IP Adresse befindet. Stimmen die Netzwerkkennungen zweier Systeme überein, befinden sie sich im selben Subnetz.

## 15.4 CIDAR

Die CIDAR ist eine vereinfachte Schreibweise für die Subnetzmaske. Da eine Subnetzmaske dadurch definiert wird, dass sie in binär Schreibweise eine fortlaufende Kette von gesetzten Bits haben muss, die nicht durch ein nicht gesetztes Bit unterbrochen werden darf, kann man die Schreibweise dahingehend vereinfachen, dass einfach die gesetzten Bits gezählt werden:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

In diesem Falle wäre die CIDAR /17

## 15.5 VLSM

VLSM ist ein erweitertes Subnetting. Hierzu wird die Subnetmaske in eine variable Länge gebracht um das Subnetz in mehrere verschieden große Teile zu unterteilen und sie hierarchisch nach ihrer Größe sortiert. Somit ist es möglich, Subnetze mit einer jeweils verschiedenen Anzahl an Hosts zu erschaffen, ohne dass dafür eine große Menge an IP-Adressen verschwendet werden muss.

**Beispiel:**

Aufgabenstellung:

Es wird ein neues Gebäude gebaut. Dafür steht der IP-Bereich 11.137.4.0/23 zur Verfügung. Die ersten 2 von den 6 Etagen sollen jeweilt 100 IP-Adressen haben und die restlichen 4 jeweils 50 IP-Adressen. Gib die IP-Range jeder Etage an!

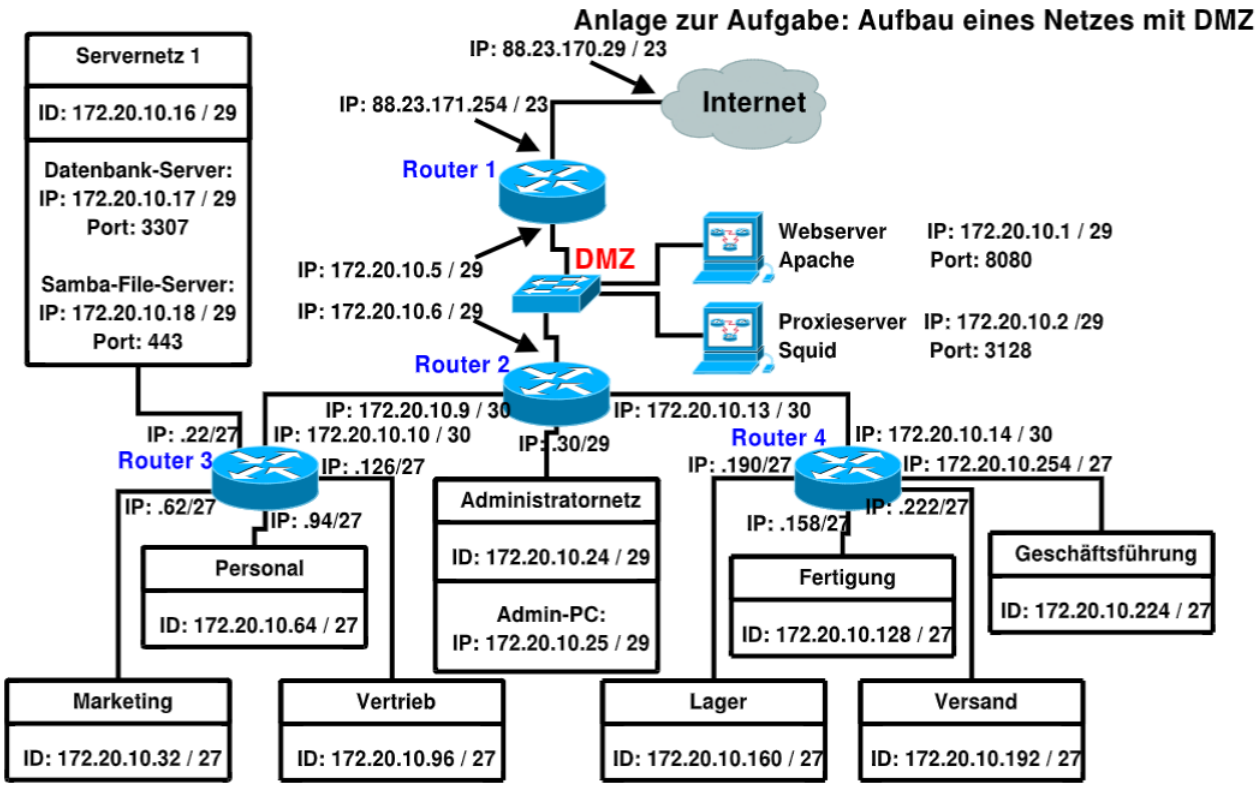
Lösung:

**Etage 1:** 11.137.4.1 - 11.137.4.126
**Etage 2:** 11.137.4.129 - 11.137.4.254
**Etage 3:** 11.137.5.1 - 11.137.5.62
**Etage 4:** 11.137.5.65 - 11.137.5.126
**Etage 5:** 11.137.5.129 - 11.137.5.190
**Etage 6:** 11.137.5.193 - 11.137.5.254

## 15.6 Routing Tabellen

## 15.7 DMZ - Demilitarisierte Zone

Eine Demilitarisierte Zone bezeichnet ein Computernetz mit sicherheitstechnisch kontrollierten Zugriffsmöglichkeiten auf die daran angeschlossenen Server. Die in der DMZ aufgestellten Systeme werden durch eine oder mehrere Firewalls gegen andere Netze abgeschirmt.




## 15.8 IPv6

Vorteile und Nachteile von IPv6 zu IPv4:

| Vorteile | Nachteile |
|---|---|
| 128 Bit = $2^{128}$ Adressen | Wenn jedes Gerät eine feste, statische Adresse bekommt, kommt es evtl. zu Sicherheitsrisiken |
| Kunden bekommen ein /64 Netz | |
| NAT wird nichtmehr gebraucht | |
| Broadcast-Adressen werden nichtmehr gebraucht | |
| ARP wird nichtmehr gebraucht | |
| bis zu 4GiB Datenversandt | |
| verbessertes Multicast | |

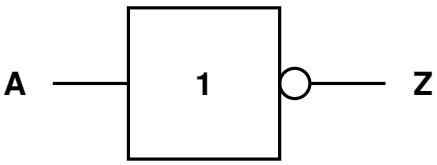
# 16 Schaltnetze

Schaltnetze sind Zusammensetzungen von logischen Gattern ohne Speicherverhalten. Sie bestehen aus einem oder mehreren einfachen Bauteilen.

## 16.1 Einfache Bauteile

### 16.1.1 NOT

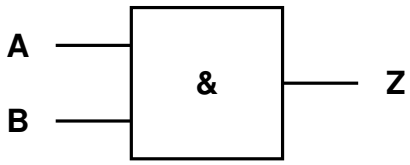
Das Not(Negation)-Bauteil negiert das Signal. Somit wird eine Eins zur Null und Nullen werden zu Einsen.




| A | Z |
|---|---|
| 1 | 0 |
| 0 | 1 |

### 16.1.2 AND

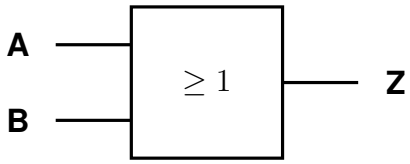
Das AND-Bauteil vereint zwei Eingänge zu einem Ausgang. Das Ausgangssignal wird nur Eins, wenn beide Eingangssignale auf Eins standen.




| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

### 16.1.3 OR

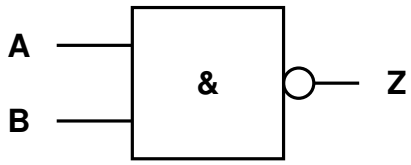
Das OR-Bauteil vereint zwei Eingänge zu einem Ausgang. Das Ausgangssignal wird dann Eins, wenn mindestens einer der Beide Eingangssignale auf Eins steht.




| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

### 16.1.4 NAND

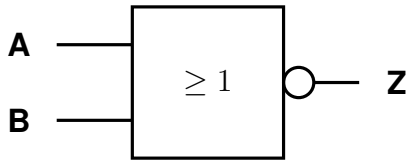
Das NAND-Bauteil vereint zwei Eingänge zu einem Ausgang. Das Ausgangssignal wird dann Eins, wenn mindestens einer der beiden Eingangssignale auf Null steht.




| A | B | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

### 16.1.5 NOR

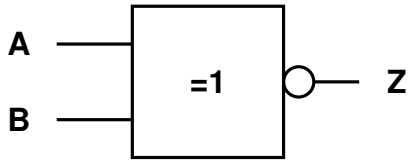
Das NOR-Bauteil vereint zwei Eingänge zu einem Ausgang. Das Ausgangssignal wird dann Eins, wenn beide Eingangssignale auf Null steht.




| A | B | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

### 16.1.6 XOR

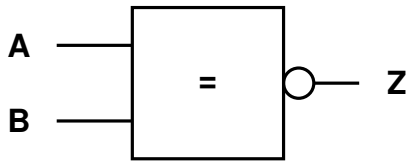
Das XOR-Bauteil vereint zwei Eingänge zu einem Ausgang. Das Ausgangssignal wird dann Eins, wenn die Eingangssignale ungleich sind.




| A | B | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

### 16.1.7 XNOR

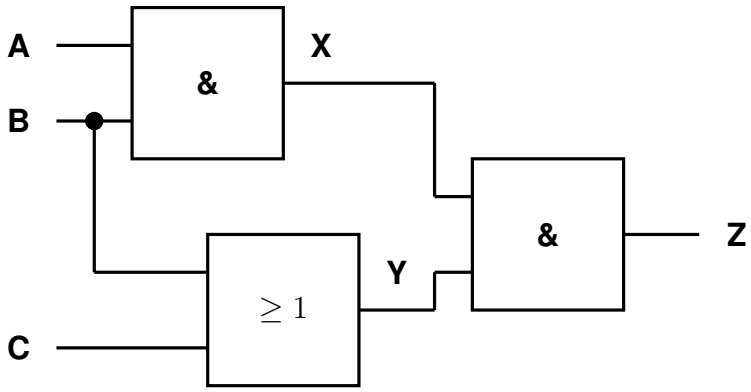
Das XNOR-Bauteil vereint zwei Eingänge zu einem Ausgang. Das Ausgangssignal wird dann Eins, wenn die beiden Eingangssignale gleich stehen.




| A | B | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## 16.2 Wahrheitstabelle

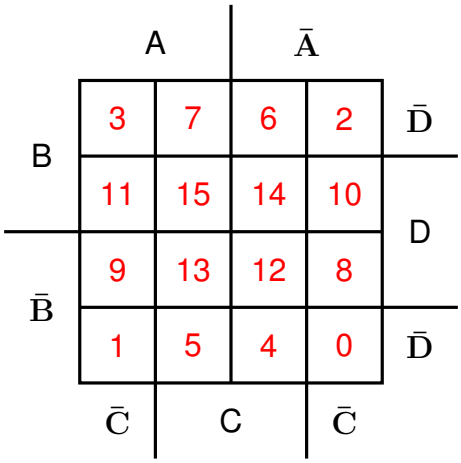
Die Wahrheitstabelle stellt alle möglichen Zustände einer Schaltung dar. Sie sieht für folgende Schaltung folgendermaßen aus:




| C | B | A | Z |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

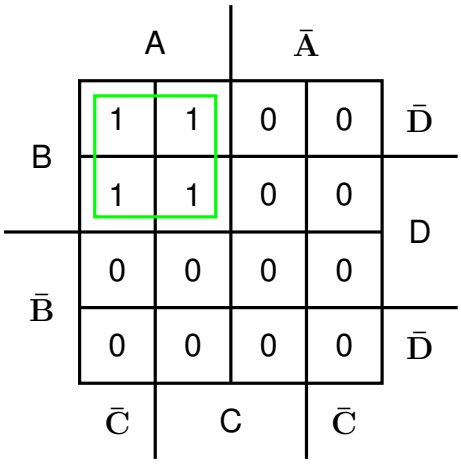
## 16.3 KV-Diagramm

Die Nummern im KV-Diagramm ergeben den dezimalen Wert, des entstehenden Binärcodes(siehe Wahrheitstabelle). Die Disjunktiven Normalformen der Schaltfunktionen lassen sich mit dem KV-Diagramm erheblich leichter kürzen! Man sucht aneinander liegende Einsen und bestimmt deren Gemeinsamkeiten. Jede „1“ darf nur in einem Größeren Bereich verwendet werden, somit müssen möglichst viele Felder zu Einem gruppiert werden.




**Beispiel:**
Für folgende Wahrheitstabelle sieht das KV-Diagramm folgendermaßen aus:

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




$Z = A \wedge B$

### 16.3.1 Don't-Care-Positions

Don't-Care-Positions sind Zustände bei welchen der Zustandswert egal ist. Im KV-Diagramm wird hierfür ein X eingetragen. Don't-Care-Positions können auch mit als Kästen zusammengefasst werden.

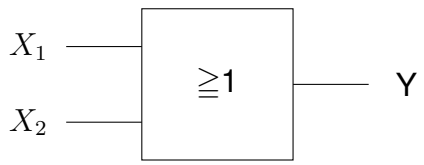
## 16.4 Disjunktive Normalform

Die Disjunktive Normalform ist ein ungekürzter Term für die Bestimmung eines Ausgangssignales. Mit ihr kann das Ausgangssignal bei bekannten Eingangswerten bestimmt werden.

## 16.5 Impulsdiagramm

Das Impulsdiagramm zeigt die Highs und Lows der Ein- und Ausgänge, also quasi eine bildliche Wahrheitstabelle

**ODER**

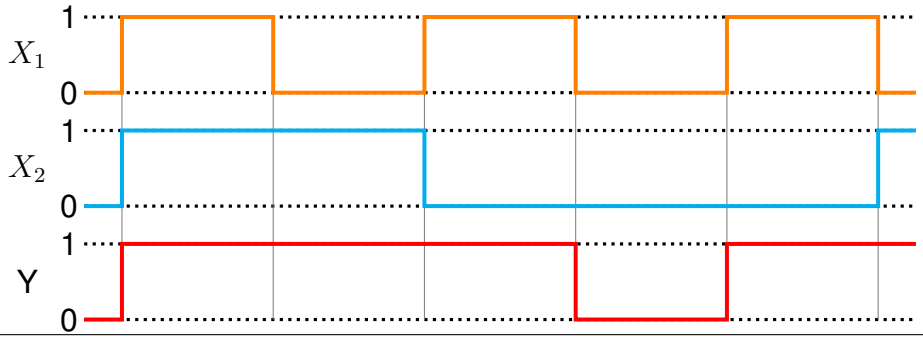



**Formel**

$$\mathbf{Y = X_1 \vee X_2}$$

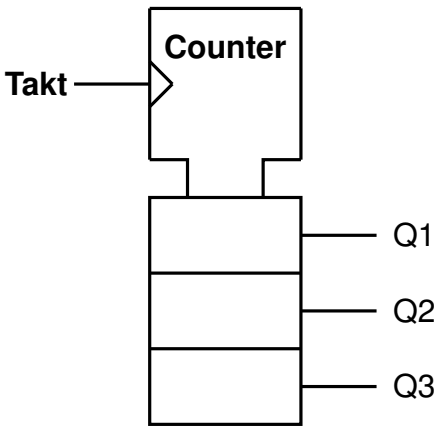
| Wahrheitstabelle | | |
|---|---|---|
| Eingang $X_1$ | Eingang $X_2$ | Ausgang Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Impulsdiagramm**




## 16.6 Zustandsfolgetabelle

Die Zustandsfolgetabelle gibt die aktuelle und Übergangsfunktion für ein Schaltwerk oder Schaltnetz.




| | | | $t_n$ | | | $t_{n+1}$ |
|---|---|---|---|---|---|---|
| **Q3** | **Q2** | **Q1** | **Q3** | **Q2** | **Q1** |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |

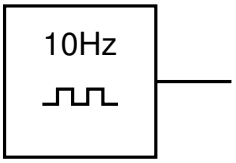
# 17 Schaltwerke

## 17.1 Schaltwerke

Schaltwerke besitzen Speicherverhalten, d.h. der Ausgabewert hängt nicht nur von der Eingabe, sondern auch vom Zustand des Schaltwerks ab.

## 17.2 Taktgeber

Der Taktgeber gibt einen Takt. Dieser wechselt zwischen Einsen und Nullen. Meist wird die Geschwindigkeit als Frequenz angegeben. Die Zeit $t$ zwischen den Wechseln, wird berechnet durch: $t = \frac{1}{f}$



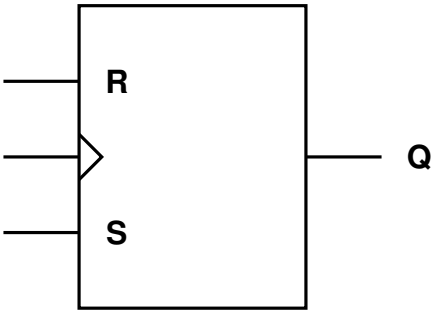

## 17.3 Flip-Flops

Flip Flops sind Speichereinheiten, welche ein Signal zwischenspeichern können.

### 17.3.1 RS Flip Flop

Das RS Flip Flop kann Signale speichern und behält diese bis sie rückgesetzt werden. Hierfür gibt es 3 Eingänge. Der erste Eingang ist R für Rücksetzen, der zweite ist der Takt und

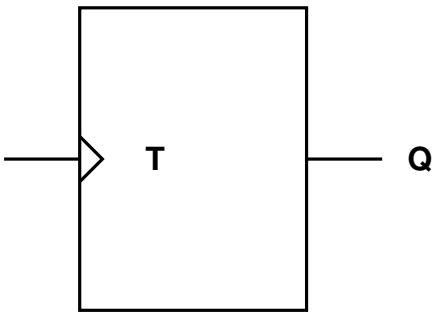
der dritte ist S fürs Setzen. Es ist Taktflanken gesteuert und das Signal tritt bei aufsteigender Flanke ein.




| Takt | R | S | $Q^{n+1}$ |
|---|---|---|---|
| ↑ | 0 | 0 | $Q^n$ |
| ↑ | 1 | 0 | 0 |
| ↑ | 0 | 1 | 1 |
| ↑ | 1 | 1 | undefined |

### 17.3.2 T Flip Flop

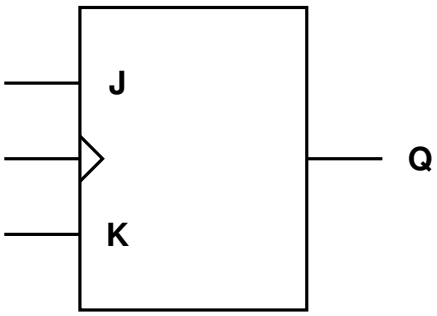
Das t Flip Flop ändert bei jeder aufsteigenden Taktflanke das Ausgangssignal Q.




| Takt T | $Q^{n+1}$ |
|---|---|
| ↑ | $\overline{Q^n}$ |

### 17.3.3 JK Flip Flop

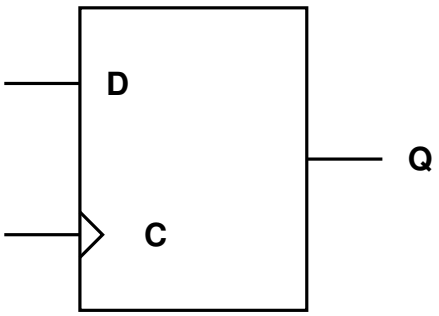
Das JK Flip Flop reagiert ebenfalls auf die aufsteigende Taktflanke und ändert die Signale wie folgt:




| Takt | J | K | $Q^{n+1}$ |
|---|---|---|---|
| ↑ | 0 | 1 | 0 |
| ↑ | 1 | 0 | 1 |
| ↑ | 1 | 1 | $\overline{Q^n}$ |
| sonst | X | X | $Q^n$ |

### 17.3.4 D Flip Flop

Das D Flip Flop reagiert ebenfalls auf die aufsteigende Taktflanke und ändert dabei den Ausgang Q zum Eingang D.

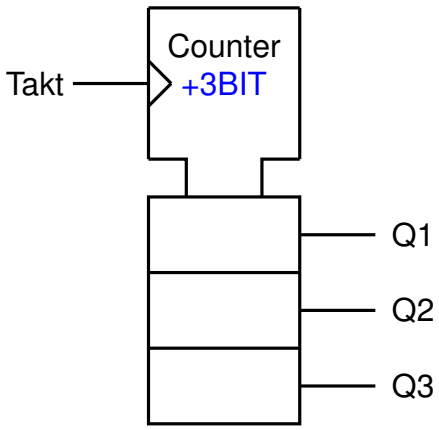



| Takt | D | $Q^{n+1}$ |
|---|---|---|
| ↑ | 0 | 0 |
| ↑ | 1 | 1 |
| sonst | X | $Q^n$ |

## 17.4 Zähler

Der positive/normale Zähler/Counter zählt bei jedem Takt um Eins hoch bis er sein Limit erreicht hat. Danach beginnt er wieder von Neuem. Negative Zähler zählen von Oben nach unten. Hierbei muss außerdem unterschieden werden ob es sich um einen Synchron- oder Asynchronzähler handelt. Synchronzähler zählen bei aufsteigender Taktflanke und Asynchronzähler bei fallender Taktflanke.

**+** positiver Zähler
**-** negativer Zähler

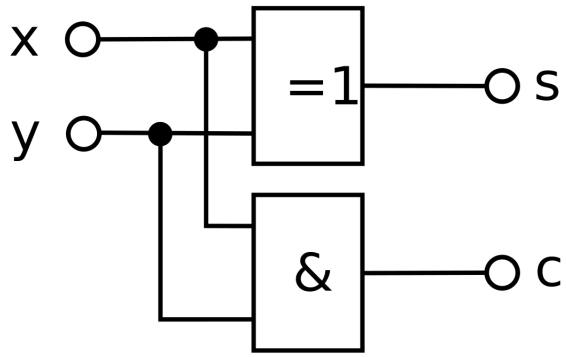



| $t_n$ | | | $t_{n+1}$ | | |
|---|---|---|---|---|---|
| **Q3** | **Q2** | **Q1** | **Q3** | **Q2** | **Q1** |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |

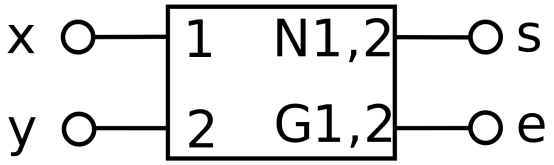
## 17.5 Addierschaltungen

Addierschaltungen lassen sich aufteilen in Halb- und Volladdierer. Diese unterscheiden sich in der Größe der Summanden.

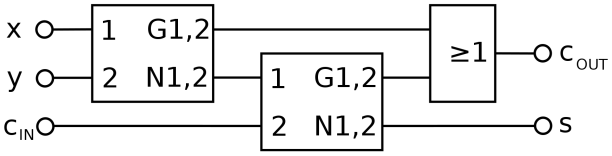
Halbaddierer aus UND und XOR:



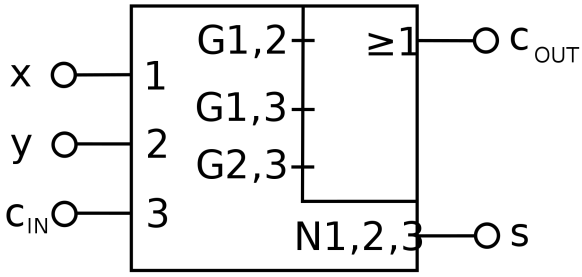

Halbaddierer Schaltsymbol:




Volladdierer aus zwei Halbaddern und einem OR:




Volladdierer Schaltsymbol:

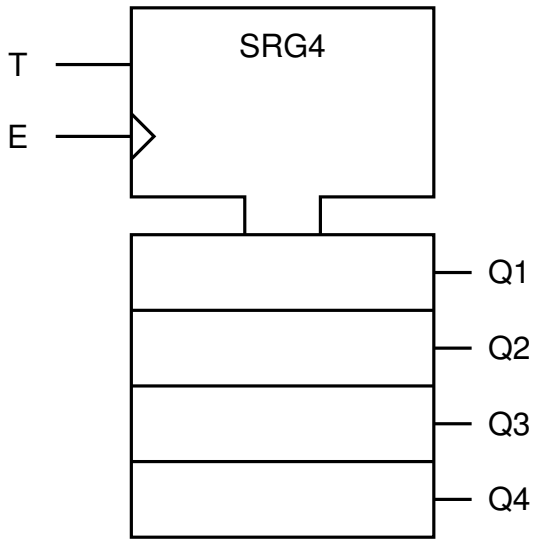



## 17.6 Subtrahierschaltungen

Subtrahierschaltungen lassen sich aufteilen in Halb- und Vollsubtrahierer. Diese unterscheiden sich in der Größe der Subtrahenden.

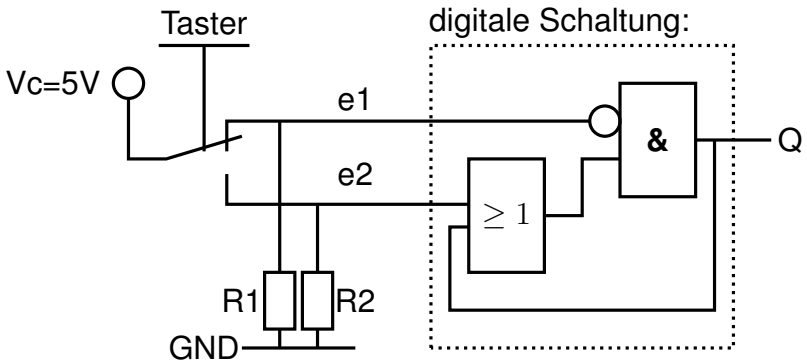
## 17.7 Schieberegister

Mehrere in Reihe geschaltete Flipflops schieben ihren Speicherinhalt bei jedem Arbeitstakt um ein Flipflop weiter – anschaulich gesehen ähnlich einer Eimerkette. Die Anzahl der im Register vorhandenen Speicherplätze ist konstant.




## 17.8 Entprellen von Signalen

Ein Signal ist prellfrei, wenn das Signal sauber als Ein- oder Ausschaltsignal entziffert werden kann. Hier ein Beispiel zum Entprellen eines Tasters:




Die Wiederstände R1 und R2 dienen zum Entladen der Leitungen e1 und e2 nach Umschalten des Tasters. Nur die markierte digitale Schaltung dient zum Entprellen des Tasters. Oft wird aber auch mit Flip Flops ein Signal entprellt.

# 18 Mirkocontroller

## 18.1 Aufbau

Ein Mirkocontroller besteht aus CPU, RAM, internen und externen Speichern, Interrupts, Timern und Ports. Wir beschäftigen uns hier ausschließlich mit dem 8051-Controller. Außer-

| **Variante** | **Größe** |
|---|---|
| Port X | Byte |
| Port X.Y | Bit |
| Akku | Byte |
| Register $R_n$ | Byte |

dem gibt es noch Datenpointer, welche ein gesamtes Byte aus einer Tabelle in den Akku laden können. Interrupts sind Prozesse, welche ganz nach oben auf den Stack gelegt werden und dann direkt abgearbeitet werden. Sie unterbrechen damit eine Abfolge von Prozessen da ihre Priorität höher ist. Der Stack ist ein Stapel an Prozessen, welcher von der CPU abgearbeitet werden muss.

# 19 Assembler

Eine Assemblersprache, kurz auch Assembler genannt, ist eine Programmiersprache, die auf den Befehlsvorrat eines bestimmten Computertyps ausgerichtet ist. Sie ist die letzte/tiefste Sprache vor der Maschienensprache. Alle Hochsprachen basieren auf Assembler oder C.

## 19.1 Wichtige Befehle

## 19.2 Adressierungsarten bei Mikrocontroller

## 19.3 Datenpointer

Der Datenpointer kann mithilfe des Akkus auf eine 1 Byte-große Zeile in einer Tabelle verweisen und diese in den Akku laden. Vor der Benutzung eines Datenpointers, muss man diesen zuerst initialisieren. Hierzu schreibt man:

```
MOV DPTR, #tabelle
```

Um nun eine Zeile aus der Tabelle in den Akku zu laden, schreibt man die Nummer der Zeile in den Akku und benutzt den Befehl MOVC:

```
MOV A, #2
MOVC A, @A+DPTR
MOV A, P2 ;Speichert Zeile in Port 2
```

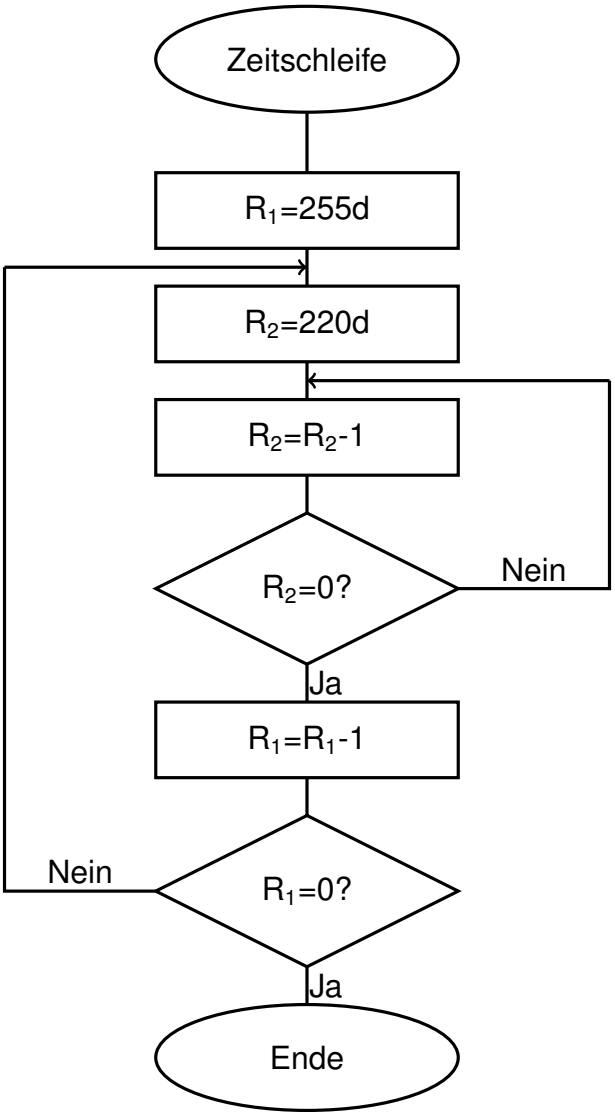
Die Tabelle muss folgendermaßen gestaltet werden:

```
tabelle:
db 0101 0111b
db 0001 1010b
db 1010 1000b
...
```

'db' steht für define Byte und 'b' für Binär. Statt binär kann man natürlich auch 'h' für Hexadezimal oder 'd' für Dezimal nehmen. Wichtig ist außerdem, dass wenn bei Hexadezimal-Darstellung die erste Ziffer einer Zeile mit einem Buchstaben beginnt, eine 0 davor muss. Beispiel: *db A2h* muss als *db 0A2h* geschrieben werden.

## 19.4 Zeitschleife erstellen ohne Timer

Aufgrund Covid-19 ist der Timer kein Bestandteil des diesjährigen Abiturs, weshalb wir Wartezeiten nur mit Schleifen erstellen können. Hierbei verwenden wir Register. Man setzt die 8-Bit Register auf eine Zahl und bildet mit mindestens 2 Registern eine Schleife. Zuerst wird der Wert des inneren Registers immer um 1 reduziert, bis es 0 ist, dann wird das äußere Register um 1 reduziert und das Innere Register beginnt von neu. Hierbei wird je nach Geschwindigkeit des Mikrocontrollers eine Wartezeit erzeugt.




## 19.5 Zeitberechnung mit der Näherungsformel

Die Formel lautet:

Zeit = Maschienenzyklengeschwindigkeit des Mikrocontrollers * Wert R1 * Wert R2 * Maschienenzyklen für DJNZ Befehl

**Beispiel:**
R1=255 & R2=220 & Geschw. des Mikrocontrollers=1MZ & DJNZ-Befehl=2MZ

$$t = 2 * 255 * 220 = 112200\mu s = 112,2ms = 0,1122s$$

Wenn man die ganz genaue Zeit eines Programmes berechnen will, muss man natürlich noch alle Initialisierungsbefehle, usw. hinzuaddieren.

## 19.6 Interrupt

In der Informatik versteht man unter einem Interrupt eine kurzfristige Unterbrechung der normalen Programmausführung, um einen, in der Regel kurzen, aber zeitlich kritischen, Vorgang abzuarbeiten. Das Program, welches nach dem Interrupt ausgeführt wird nennt man Interrupt Service Routine(ISR). Um den Interrupt verwenden zu können, muss man ihn zuerst initialisieren. Dazu muss man in der Formelsammlung des Mirkocontrollers nachschauen, welche Ports Interrupt-fähig sind. An diesen Ports kann man nun seinen Interrupt aktivieren und Einstellen wann er reagieren soll.
Eine mögliche Initialisierung könnte folgendermaßen aussehen:

```
Init:   SETB IT0
        SETB EX0
        SETB EAL
```

Um die ISR zu schreiben, muss man die Einsprung Adresse des Interrupts finden.
Dies sieht dann folgendermaßen aus:

```
ORG 0003h:
marke0:  ...
         ...
```

# 20 Objektorientierte Programmierung

## 20.1 Grundverständnis

Die objektorientierte Programmierung ist ein auf dem Konzept der Objektorientierung basierendes Programmierparadigma. Die Grundidee besteht darin, die Architektur einer Software an den Grundstrukturen desjenigen Bereichs der Wirklichkeit auszurichten, der die gegebene Anwendung betrifft.

## 20.2 Klassen

Unter einer Klasse versteht man in der objektorientierten Programmierung ein abstraktes Modell bzw. einen Bauplan für eine Reihe von ähnlichen Objekten. Die Klasse dient als Bauplan für die Abbildung von realen Objekten in Softwareobjekte und beschreibt Attribute und Methoden der Objekte.

## 20.3 Objekte

In den meisten objektorientierten Programmiersprachen wird ein Objekt oder Exemplar (auch Instanz genannt) aus einer Klasse erzeugt, mittels Konstruktion (siehe auch Konstruktoren und Destruktoren). Diese Instanz besitzt dann zur Laufzeit ihren eigenen Datentyp, eigene Eigenschaften und Methoden. Ein Beispiel für Klassen und Objekte ist ein Mitarbeiter Programm wobei die Klasse ein Formular mit auszufüllenden Daten zu den Mitarbeitern ist und jeder Mitarbeiter ein eigenes Objekt ist, bei dem die Daten eingetragen sind.

## 20.4 Datentypen

Unter den wichtigen Datentypen in der Hochsprache C# befinden sich:

| | |
|---|---|
| int | Integer sind positive oder negative ganze Zahlen |
| double | Kommazahlen und Ganze Zahlen |
| float | 4 Byte große Gleitkommazahl |
| string | Zeichenfolgen/Text |
| bool | Wahr oder falsch, 1 oder 0, binärer Zustand |
| char | einzelner Character |

## 20.5 Arrays

Ein Array/Feld ist in der Informatik eine Datenstruktur-Variante, mit deren Verwendung viele gleichartig strukturierte Daten [...] verarbeitet werden sollen. Der Zugriff auf bestimmte Inhalte eines Felds erfolgt mit Hilfe von Indizes, die dessen Position bezeichnen.

*int[] myArray = new int[3];*

Ein Array namens *myArray* wurde erstellt und ihm wurden drei Integer-Speicheradressen reserviert.

## 20.6 Verebung

Wenn Klassen erben, besitzen diese eine abstrakte *Mutterklasse*. Diese vererbt Methoden und Attribute. Für jedes Objekt gibt es nun 2 Objekte. Eines der Mutterklasse und eines der Unterklasse. Ein gutes Beispiel hierfür ist die Mutterklasse Fahrzeug mit ihrer *Kindklasse* PKW. Beim Aufrufen des Konstruktors von PKW wird ebenso der Konstruktor der Klasse Fahrzeug aufgerufen.

## 20.7 Polymorphie

### 20.7.1 Polymorphie

Polymorphie bedeutet, dass ein Bezeichner abhängig von seiner Verwendung, Objekte unterschiedlichen Datentyps annehmen kann. Somit kann man mit einem Verweis/Pointer mehrere Objekte/Datentypen aufrufen. Eine Methode ist polymorph, wenn sie in verschiedenen Klassen die gleiche Signatur hat, jedoch erneut implementiert ist. Polymorphie wird oft

auch als Überladen beschrieben. Ebenso ist das Überladen einer Methode mit unterschiedlichen oder unterschiedlich vielen Parametern Polymorphie.

### 20.7.2 Dynamische Polymorphie

Man spricht von der dynamischen Polymorphie, wenn Methoden von einer Basisklasse/Mutterklasse abgeleitet und unterschiedlich implementiert werden. Dies wird auch als Überschreiben von Methoden bezeichnet.

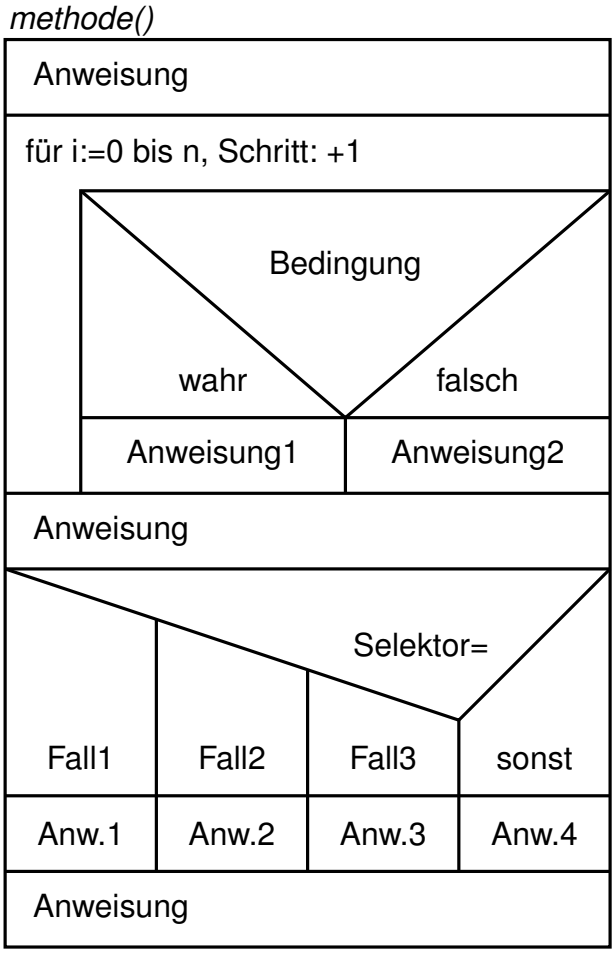
## 20.8 UML

Die Unified Modeling Language, kurz UML, ist eine grafische Modellierungssprache zur Spezifikation, Konstruktion, Dokumentation und Visualisierung von Software-Teilen und anderen Systemen. Sie wird von der Object Management Group entwickelt und ist sowohl von ihr als auch von der ISO genormt.

### 20.8.1 Struktogramm

Ein Struktogramm wird verwendet um einen Ablauf eines Programmes dar zu stellen.

```
public void methode(){
    Anweisung;
    for(int i=0; i<=n; i++){
        if(Bedingung){
            Anweisung1;
        }else{
            Anweisung2;
        }
    }
    Anweisung;
    switch(Bedingung){
        case Fall1: Anweisung1; break;
        case Fall2: Anweisung2; break;
        case Fall3: Anweisung3; break;
        default: Anweisung4;
    }
    Anweisung;
}
```



### 20.8.2 Klassendiagramm

Klassendiagramme können vereinfacht und detailliert dargestellt werden. Sie veranschaulichen alle Attribute und Methoden einer Klasse.

### 20.8.3 Objektdiagramm

Objektdiagramme werden im Bezug zum Klassendiagramm gezeichnet. Sie sind eine Instanz der Klasse und die Attribute sind mit Werten gefüllt.

### 20.8.4 Assoziationen

Klassendiagramme können außerdem die Assoziationen zwischen verschiedenen Klassen darstellen:

### 20.8.5 Polymorphie

Polymorphie beschreibt das Erben zwischen Klassen und veranschaulicht dies.

### 20.8.6 Vererbungen

Vererbungen können außerdem vereinfacht dargestellt werden.

### 20.8.7 Sequenzdiagramm

Sequenzdiagramme stellen den Ablauf eines Programmes dar. Hierbei sieht man außerdem die Schritte in den verschiedenen Klassen.

### 20.8.8 Zustandsdiagramm

Beim Zustandsdiagramm kann man herauslesen, wie das Programm in einem gewissen Zustand fortfahren kann. Hierbei wird jeder Zustand aufgezeichnet und alle möglichen Optionen ergänzt.

# 21 Datenbanken

## 21.1 Grundverständnis Datenbanken

Eine Datenbank, auch Datenbanksystem genannt, ist ein System zur elektronischen Datenverwaltung. Die wesentliche Aufgabe einer Datenbank ist es, große Datenmengen effizient, widerspruchsfrei und dauerhaft zu speichern und benötigte Teilmengen in unterschiedlichen, bedarfsgerechten Darstellungsformen für Benutzer und Anwendungsprogramme bereitzustellen. Eine Datenbank besteht aus mehreren Tabellen. Jede Tabelle hat eigene Spalten. Diese Tabellen können zusammenarbeiten und effizient Daten speichern. Daten müssen nicht doppelt oder dreifach gespeichert werden.

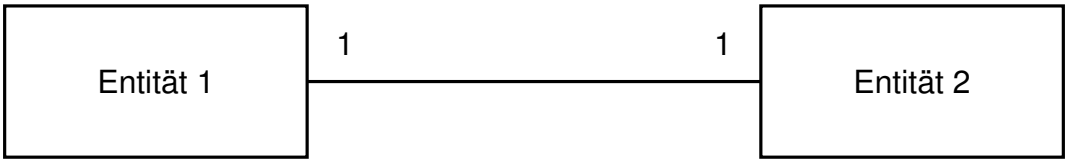
## 21.2 Verwendung

Um Daten dauerhaft, effizeint und widerspruchsfrei zu speichern.

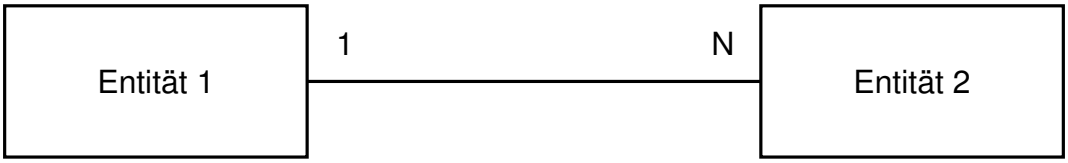
## 21.3 Entity-Relationship-Diagramm

Das ER-Diagramm gibt einen schnellen und übersichtlichen Überblick über die Tabellen einer Datenbank. Das ER-Diagramm wird zum Beispiel beim Erstellen und Planen einer Datenbank verwendet. Bevor man das ER-Diagramm zeichnen will, sollte man sich aber alle Beziehungen der Tabellen ansehen. Treten n:m-Beziehungen auf, müssen diese zuerst aufgelöst werden:

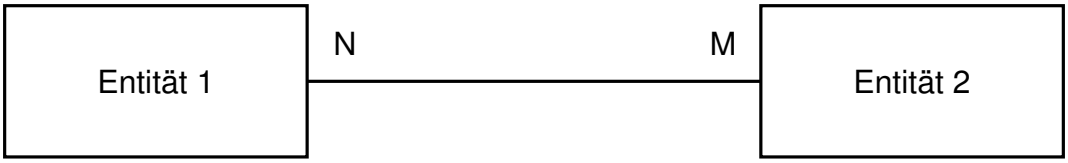
**1:1 Beziehung:**



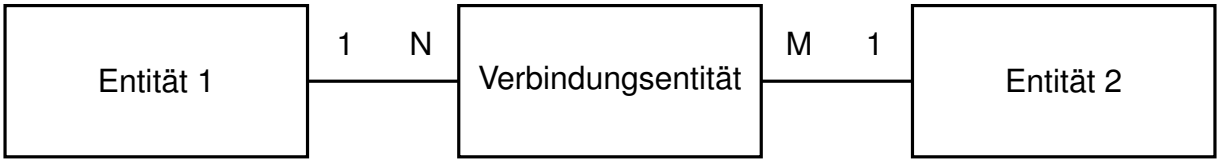

**1:n Beziehung:**




**n:m Beziehung:**




**n:m Beziehung(aufgelöst):**




## 21.4 Primärschlüssel

Der Primärschlüssel findet sich in jeder Tabelle einer Datenbank. Er ist meist die ID der jeweiligen Einträge, da diese sich nicht doppelt. Mit dem Primär- und Sekundärschlüssel können Tabellen verknüpft werden.

## 21.5 Sekundärschlüssel

Der Sekundärschlüssel ist der Primärschlüssel der jeweils anderen Tabelle. Dieser befindet sich außerdem in der lokalen Tabelle.

## 21.6 Realtionsschreibweise

Bei der Realtionsschreibweise erkennt man direkt die wichtigsten Informationen und die Spalten einer Datenbank. Hierbei muss deutlich klar gemacht werden, was der Primär- und was der Sekundärschlüssel sind.
**Beispiel:**
Angestellter(**PerNr**, Name, Vorname, AbtNr)
Abteilung(**AbtNr**, Bezeichnung)

Legende:
**Fett**=Primärschlüssel, Unterstrichen=Sekundärschlüssel

## 21.7 SQL

SQL ist eine Datenbanksprache zur Definition von Datenstrukturen in relationalen Datenbanken sowie zum Bearbeiten und Abfragen von darauf basierenden Datenbeständen.

### 21.7.1 SQL Abfragen

Um gefilterte Informationen aus Datenbanken zu entnehmen werden SQL-Abfragen geschrieben. Mit diesen kann man mehrere Tabellen verknüpfen und gefiltert nach den gewünschten Daten suchen.

```
SELECT Buchungen.BuchungsID, Buchungen.KundenID,
Kunden.Name FROM Kunden, Buchungen
WHERE Kunden.KundenID = Buchungen.KundenID
AND KUNDEN.Name LIKE "Müller"
```

Mit diesem Befehl werden alle Buchungsnummern, Kundennummern und Kundennamen von Kunden mit dem Namen Müller ausgegeben.

Der Aufbau einer SQL-Abfrage muss folgende Reihenfolge einhalten:

```
SELECT <Spalte>,<Aggregatfunktion>
FROM <Tabelle>
WHERE <Bedingung>
GROUP BY <Spalte>
HAVING <Aggregatfunktion> <VerglOp> <Wert>
ORDER BY <Spalte>[ASC|DESC], <Aggregatfunktion>
LIMIT [start,] anzahl
```

Hier einige Befehle:

| | |
|---|---|
| DISTINCT | Vermeidet doppelte Datensätze(kommt nach SELECT) |
| ASC | Alphabetische/Aufsteigende Sortierung |
| DESC | Absteigende Sortierung |
| SUM() | Summe |
| COUNT() | Anzahl |
| AVG() | Durchschnittlicher Wert |
| MAX() | Höchster Wert |
| MIN() | Kleinster Wert |

Beispiel:
**SELECT fahrradtypen.Typbezeichnung, SUM(DATEDIFF(bis, von) * Tagesmietpreis) AS "Mieteinnahmen"**
**FROM fahrraeder, fahrradtypen, vermietungen**
**WHERE fahrraeder.FahrradNr = vermietungen.FahrradNr**
**AND fahrradtypen.TypNr = vermietungen.TypNr**
**GROUP BY fahrradtypen.Typbezeichnung**
**HAVING SUM(DATEDIFF(bis, von) * Tagesmietpreis) >= 1000**

Mit diesem Befehl werden alle Mieteinnahmen und Fahrradtyp ausgegeben bei denen die Mieteinnahmen größer als 1000€ sind.

# 22 Fachsprache

## 22.1 Fachwörter

## 22.2 Typfeld Kennungen

# 23 Epilog

Diese Zusammenfassung wurde erstellt und geteilt von *Robin Rausch* und *Jannis Müller*. Für Falschaussagen oder Fehlinformationen übernehmen die Autoren keinerlei Gewähr. Es ist empfehlenswert, sich selbst mit den Inhalten vertraut zu machen und die hier dargestellten Inhalte gegebenenfalls zu korrigieren. Bei Fragen oder inhaltlichen Fehlern kontaktieren sie bitte umgehend einen der Autoren und weisen auf die von Ihnen festgestellten Mängel hin.

## 23.1 Literatur

Fachwissen sowie einige Inhalte wurden übernommen aus:
*Informatik und Informationstechnik - für Gymnasien und höhere Bildungsgänge im beruflichen Schulwesen (3. Auflage)* verlegt von *Europa Lehrmittel*.

| Fachbegriff | Erklärung |
|---|---|
| Algorithmus | Eine genau definierte Handlungsvorschrift zur Lösung von Problemen in endlich vielen Schritten |
| Attribut | Eigenschaft einer Klasse |
| ASCII | American Standard Code for Information Interchange |
| ARP | Adress Resolution Protocol |
| BCD-Code | Binary Coded Decimal (dualkodierte Dezimalziffer) |
| Client-Server Netz | Ein Client-Server Netz besteht zwischen einem Server und einem Computer. Der Server bietet dem Client verschiedene Dienste an, der Client kann Dienste vom Server anfordern. |
| CRC / FCS | Cyclic Redundancy Check / Frame Check Sequence |
| CSMA/CD | Carrier Sense Multiple Access / Collision Detection |
| Geheimnisprinzip | Lesen oder Schreiben der Attributwerte eines Objekts nur durch Operation möglich |
| Kapselung | Attribute sind durch Operationen von der Außenwelt abgeschlossen (Attribute private, Operationen public) |
| Klasse | Beschreibt Gemeinsamkeiten einer Menge gleichartiger Objekte |
| Klassenattribut | Ein Attributwert für alle Objekte der Klasse |
| MAC Adresse | Media Access Control Adresse |
| MTU | Maximum Transmission Unit (Nutzdatenblock Ethernet Frame) |
| Objekt | Gegenstand der Programmierung (materieller Gegensand, Organisationseinheit, Begriff, Aspekt eines Systems) |
| OUI | Organizationally Unique Identifier |
| OSI (Modell) | Open Systems Interconnedtion (Modell) |

| Fachbegriff | Erklärung |
|---|---|
| Peer-to-Peer Netz | Ein Peer-to-Peer Netz besteht aus zwei oder mehreren gleichberechtigten Systemen. |
| Padding | Füllbits zum Auffüllen von Segmenten, in denen der Datenanteil nicht die Mindestgröße ausfüllt. |
| Redundanz | Beschreibt diejenigen Informationen oder Daten, die in einer Informationsquelle mehrfach vorhanden sind. Eine Informationseinheit ist dann redundant, wenn sie ohne Informationsverlust weggelassen werden kann. |
| RTD | Round Trip Delay |
| RTT | Round Trip Time |
| SFD / SOF | Start Frame Delimiter / Start of Frame |
| Virtuelle Kommunikation | Beschreibt die Kommunikation der zwei Systeme auf der entsprechenden Ebene über einheitliche Sprache bzw. Protokolle |

| | |
|---|---|
| 0x0800 | IPv4 Internet Protocol, Version 4 |
| 0x0806 | ARP Adress Resolution Protocol |
| 0x0842 | WoL Wake on Lan |
| 0x8035 | RARP Reverse Adress Resolution Protocol |
| 0x809B | EtherTalk Appletalk |
| 0x80F3 | AARP Appletalk Adress Resolution Protocol |
| 0x8100 | VLAN Tag |
| 0x8137 | Novell IPX |
| 0x8138 | Novell |
| 0x86DD | IPv6 Internet Protocol, Version 6 |
| 0x8863 | PPPoE Discovery |
| 0x8864 | PPPoE Session |
| 0x8870 | Jumbo Frames |
| 0x8892 | Echtzeit-Ethernet PROFINET |
| 0x88A2 | ATA over Ethernet Coraid AoE |
| 0x88A4 | EtherCAT Echtzeit-Ethernet |
| 0x88A8 | Provider Bridging |
| 0x88AB | Echtzeit-Ethernet POWERLINK |
| 0x88CD | Echtzeit-Ethernet SERCOS III |
| 0x8906 | Fibre Channel over Ethernet |
| 0x8914 | FCoE Initialization Protocol FIP |