Theorieausarbeitung TGI Test 3

Überblick Programmiersprachen:

- Generation 1 → Direkte Maschinencodes (01010101), abhängig von Architektur + Prozessor (zB. Mikro16).
- G2, Maschinenbefehle (Assembler → Interpreter), abhängig von Architektur
- G3, Höhere Programmiersprachen (Compiler; Pascal, C, Modula)
- G4, Fourth Generation Language → anwendungsbezogen, zB. Natrual
- G5, Very High Level Language (VHLL), ermöglicht Darstellung auf Problemebene, zB. Prolog.

Maschinenbefehle:

- Arithmetische → Addition, Subtraktion, Division, Inkrement, Dekrement
- Logische → AND, OR, XOR, Invertieren
- Datentransferbefehle → MOV, I/O, Manipulation des Stacks
- Schiebebefehle → shifts
- Kontrollbefehle → jumps

Befehle für Datentransfer:

- **Zwischen Registern und/oder Hauptspeicher**
  - MOV Befehle
  - Kopieren eines Datenwortes (8,16,32bit)
  - Daten kopieren bis Bedingung eintritt
- **Zwischen Register/Hauptspeicher und peripherem Gerät**
  - memory mapped I/O (→ Reg- des peripheren Geräts wird wie Hauptspeicher behandelt; Verwendung von norm. MOV Befehlen)
  - independent I/O (→ Register des Geräts wird eine Nummer (port) zugewiesen; spezielle Befehle die von ports lesen und auf ports schreiben können)
- **DMA** (direct meomory access) → Eigener Controller verwaltet I/O Zugriffe auf Peripherie; CPU setzt nur DMA Kommando ab; Interrupt nach Beendigung.

Stack (Keller oder Stapelspeicher) → Befehle zw. Registern und Stack. 2 Operationen: push / pop.
Beispiel dazu (Ü6):

Aufgabe 1: Theoriefragen

a) Erläutern Sie die Begriffe CISC und RISC. Worin liegen die Unterschiede zwischen diesen beiden Architekturen?

b) Erläutern Sie die Funktionsweise eines Stacks bzw. Kellerspeichers anhand des folgenden Beispiels. Tragen Sie die nach Ablauf der Befehlssequenz resultierenden Werte entsprechend ein (vgl. Foliensatz 13 - Befehlssatz, Folie 11 ff.). Der Stackpointer (SP) zeigt zu Beginn auf die Speicheradresse $(FFFF)_{16}$.

```
R1 ← 1
push_16(R1)
R1 ← 1+1
R2 ← lsh(1+1)
push_16(R1)
push_16(R2)
pop_16(R1)
pop_16(R2)
```

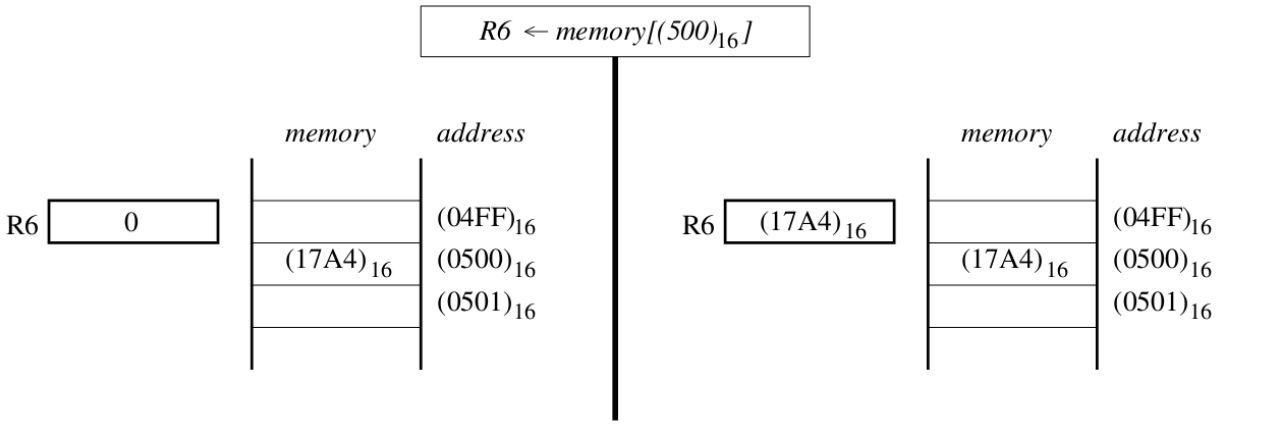
Register

RAM

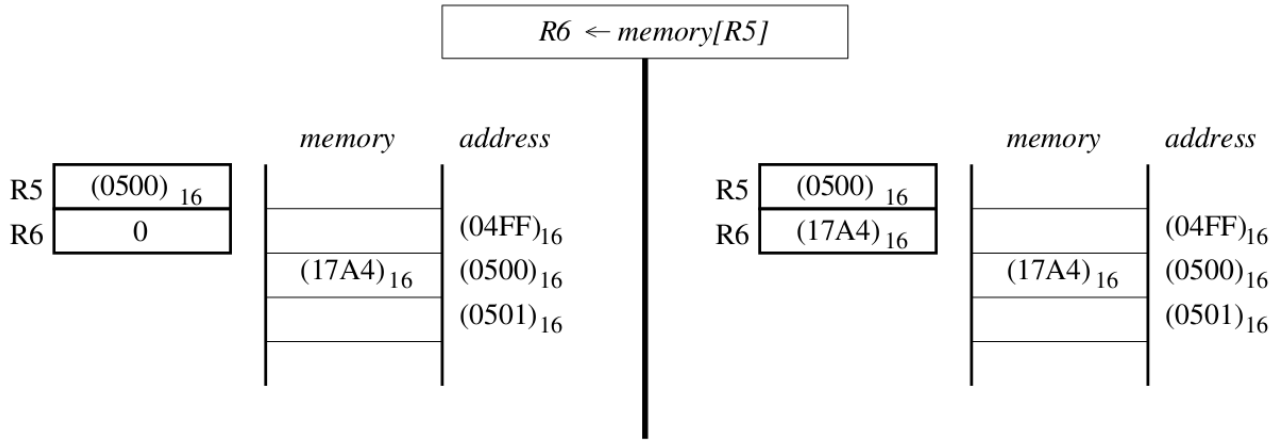
Adressierungsarten:

- *Direct-Addressing Mode*
- *Register-Indirect Mode*
- *Base-Register-Addressing*
- *Register-Indirect-With-Predecrement*
- *Register-Indirect-With-Postincrement*
- *(Memory-)Indirect-Addressing Mode*

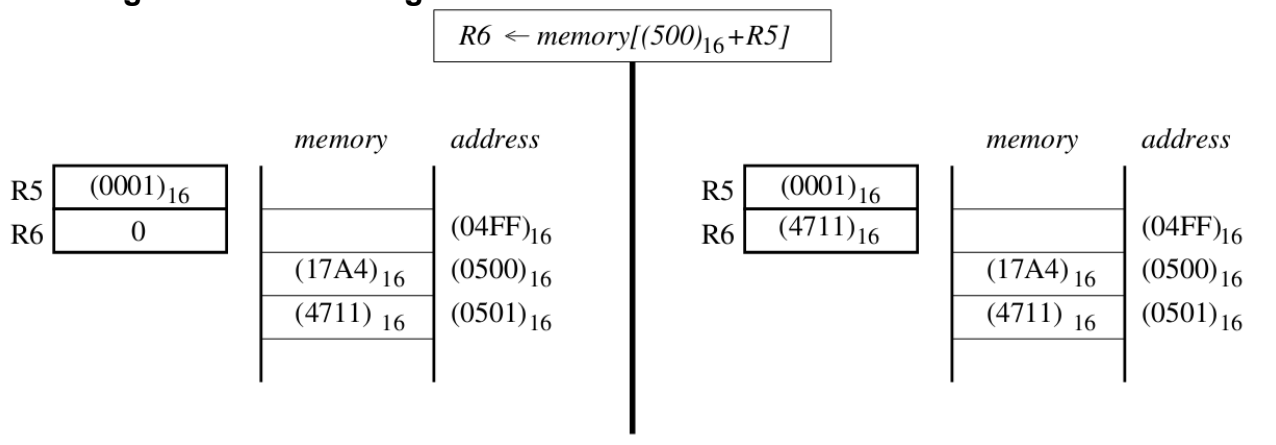
Direct Adressing Mode:



Register Indirect Mode:

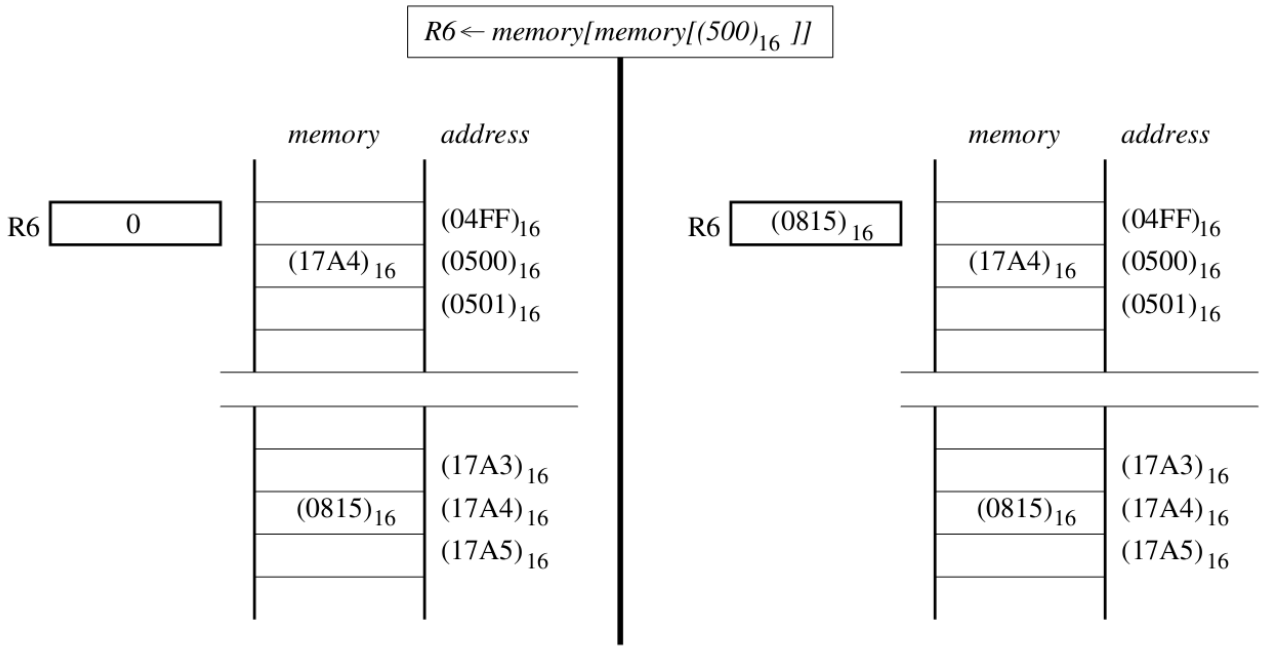


Base Register Adressierung:



Pre/Post Increment / Decrement
memory[+(Z0)]
memory[-(Z0)]
memory[(Z0)+]
memory[(Z0)-]

Memory Indirect Adressing Mode:



CISC und RISC: Im Laufe d. Zeit → Maschinenbefehle immer mächtiger, es entstand CISC (Complex instruction set computer). Probleme dabei → Steuerwerk wurde immer komplexer, und daher eher langsamer, Compiler nutzen viele der Befehle wenig oder nicht.

Dadurch entstand Idee von RISC (Reduced instruction set computer) → Weniger Befehle, einfach gleichartig aufgebaut. BUCH.

Control Flow Befehle: Unbedingte Sprünge (→ Programm setzt an neuer Adresse fort, Kontrolle wird zu einer neuen Adresse transferiert).

Bedingte Sprünge: Kontrolle wird nur unter bestimmten Umständen zu einer Adresse transferiert (zB. Z Flag, N flag). Komfortablere Sprungbefehle zB. Direkt mit Zahlen operieren (verzweige wenn der Wert im RA gleich Wert von RB ist (→ RT = RA-RB; branch if Z).

Interrupts: unterbrechen herkömmlichen Programmauflauf (kann durch prozessorinterne [Division durch 0, illegaler Maschinenbefehl] oder externe [illegaler Speicherzugriff, DMA Controller beendet Speicherbehandlung] Ereignisse ausgelöst werden) und springen ins ISR (interrupt service routine).

Funktionsaufrufe:

- Funktion Prozedur Unterprogramm
  - fasst häufig benutzen Code zusammen
  - kann von anderen Code Teilen genutzt werden
- Kommunikation
  - Argumente und Rückgabewert können am Stack oder in Registern übergeben werden.

Beispiel dazu:
(Ü8)

Aufgabe 4: Prozesse – Prozesszustand

Gegeben ist der folgende Prozess in Pseudo-Code Notation:

```
   int main() {
1:   r= get_input();
2:   f();
3:   u();
4:   return 0;
   }

   p() {
5:    b= a*3
   }

   f() {
6:    a= r*r;
7:    p();
8:    F= b;
   }

   u() {
9:    a= r*2;
10:   p();
11:   U= b;
   }
```

Nehmen Sie an, dass die Prozessausführung bei 1 beginnt und dass mit `get_input()` der Wert 5 eingelesen wird. Tragen Sie in die folgende Tabelle für jeden Schritt den Zustand des Prozesses *am Beginn* des jeweiligen Schrittes ein (siehe Foliensatz 20-21–Prozessmanagement, Folien 13ff).

Hinweis: Alle Variablen sind als global deklariert!

PC	r	a	b	F	U	Stack
1						
2						
6						
7						
5						
8						
3						
9						
10						
5						
11						
4						

Pipelining:
Verarbeitungsschritte MIPS:

IF … instruction fetch (Holt Instruktionen aus Speicher)
ID … decode and source register read (Dekodiert die Instruktionen und greift lesend auf Register zu)
EXE … execution (Führt ALU Operation aus)
MEM … memory data access (Führt lesende und schreibende Zugriffe auf Speicher aus).
WB … Target register write (Schreibt Ergebnisse in die Register zurück)

Kennwerte (5 stufige Pipeline, 2ns Taktung):
Absolute Ausführungszeit (10 Takte → 20 ns)
Realer Durchsatz → Wieviel ns pro Befehl; Wieviel Millionen Instruktionen pro Sek (MIPS).
Theoretischer Durchsatz → 1 Befehl in ? Ns, ? MIPS.

Performance Gewinn: k stufige Pipeline → Besserer Durchsatz um Faktor k; verbessert nicht die Ausführungszeit eines einzelnen Befehls.
Begrenzungen: Einlaufen d. Pipeline, Hazards

Hazards: **Strukturelle Hazards** (= mehrere Pipeline Stufen benötigen die selbe Ressource, Maßnahmen stall)
Control Hazards (aka Branch Hazards; Nachfolgebefehl hängt vom Ausgang des Sprunges ab; Maßnahmen: prediction, delayed branching, stall)
Data Hazards (= Berechnung erfordert Ergebnis des Vorgängerbefehls; Maßnahmen: forwarding, Code Optimierung, stall).

Typen von Data Hazards:
read after write → Register wird gelesen, bevor vorheriger Befehl es beschrieben hat.
Write after read → Register wird beschrieben bevor vorheriger Befehl es gelesen hat.
Writer after write → Register wird beschrieben jedoch danach von vorherigem Befehl überschrieben.

Beispiel(Ü7):

Aufgabe 1: Pipelining – Wiederholung

Ein Prozessor hat eine Pipeline mit den vier Stufen *Fetch*(F), *Decode*(D), *Execute*(E) und *Store Result*(S). Durch die Pipelinestruktur kann es zu *RAW (Read After Write) Data Hazards* kommen. Um das zu vermeiden, kann ein Befehl erst dann in der Stufe *D* verarbeitet werden, wenn der abhängige Befehl bereits die Stufe *S* abgeschlossen hat. Die Ausführung eines Befehls kann dazu in der Stufe *D* beliebig lange verzögert werden (*Stall*). Folgendes Programm wird auf dem Prozessor ausgeführt:

```
mult R1, R0, R1              # Multipliziere R0 * R1, Ergebnis --> Register 1
add  R3, R1, R1              # Addiere Register R1 + R1, Ergebnis --> Register 3
sll  R4, R2, 1               # Shift left von R2, Ergebnis --> Register 4
div  R3, R0, R3              # Dividiere R0 / R3, Ergebnis --> Register 3
sll  R2, R2, 1               # Shift left von R2, Ergebnis --> Register 2
add  R8, R6, R5              # Addiere Register R6 + R5, Ergebnis --> Register 8
jr   (R8)                    # Springe an die Speicheradresse in R8
```

a) Zeichnen Sie die Belegung der Pipeline für das gegebene Programm unter der Annahme, dass die Pipeline am Beginn und am Ende leer ist.

b) Berechnen Sie aus dem Ergebnis von Teilaufgabe a) wieviele Takte eine Instruktion bei paralleler Ausführung (mit Pipeline) im Durchschnitt benötigt.

c) Ordnen Sie die Befehle so um, dass die Ausführungszeit minimiert wird. Die Funktion des Programms muss dabei erhalten bleiben.

Superscalar Pipelining: Mehrere Datenpfade, zB. Integer, FP, load/store; Pro Taktzyklus mehrere Befehle bearbeitet, zB. CPI = 0.25; Pro TZ werden mehr Befehle geladen, zB 2 Befehle über 64bit.

Speicher:
Von Neumann: Speicher enthält Programme + Daten; Flaschenhals → Caches
Harvard: Programm + Datenspeicher getrennt; Befehle, Daten gleichzeitig, geladen, geschrieben.

Interleaved Memory: Meist wird seqzenziell auf Speicher zugegriffen, Aufteilung des Speichers in gleich große Bänke, Bereiche. Aufeinanderfolgende Adressen liegen in anderer Bank.
Beispiel (Ü7):

Aufgabe 6: Interleaved Memory

Sie arbeiten auf einem System mit 8-fach *Interleaved Memory*. Die Adressen sind dabei so verschränkt, dass aufeinanderfolgende Speicherworte jeweils in einer anderen Speicherbank liegen. Die Geschwindigkeit in einem solchen System ist natürlich umso höher, je besser eine Adress-Zugriffssequenz *Memory Interleaving* ausnutzt.

Geben Sie an, wie sich die Speicheradressen auf die einzelnen Bänke verteilen. Reihen Sie außerdem die folgenden Sequenzen nach der Geschwindigkeit, mit der die Datenwörter zur Verfügung stehen (absteigend von der schnellsten zur langsamsten). Begründen Sie, warum die einzelnen Sequenzen *Memory Interleaving* gut oder weniger gut ausnutzen.

Sequenz 1:	3	19	27	13	14	30	6	25
Sequenz 2:	18	4	16	22	27	1	21	15
Sequenz 3:	2	10	18	26	7	15	47	31
Sequenz 4:	11	31	29	37	26	0	24	17

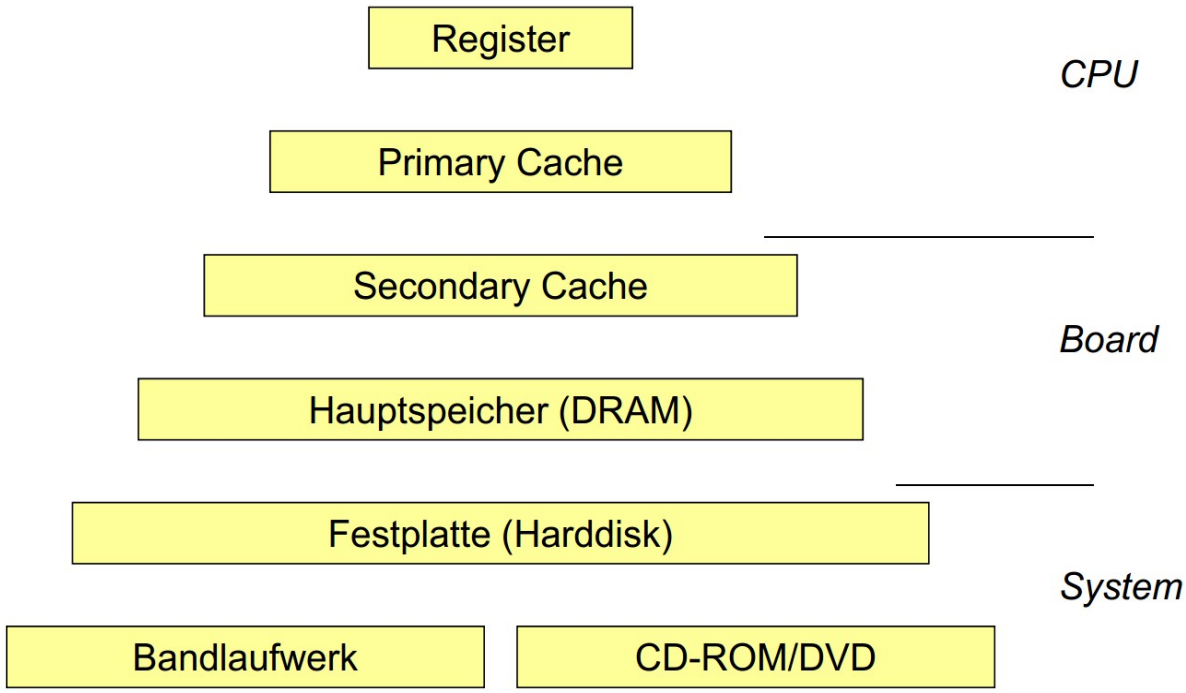
Problem: Anwender und Programmierer erwarten schnellen Speicher mit unbegrenzter Kapazität → große + schnell Speicher → unbezahlbar.

Speichertyp	Kapazität	Zugriffszeit	Kosten / Einheit
Synchronous SRAM chip	18 Mbit	3.4ns	2€/MB
DDR3-1600 Modul	4GB	10-40ns / ~200ns mit Speichercontroller	4€/GB
Flash SSD	250GB	0,01ms read / 0,35ms write	1,4€/GB
Festplatte	2TB	10ms	50€/TB
Blue Ray Double Layer 12x	50GB	180ms	40€/TB

Temporal Locality: Wurde auf einen Speicherinhalt erst kürzlich zugegriffen, so ist die Wahrscheinlichkeit eines baldigen neuerlichen Zugriffs relativ hoch.

Spacial Locality: Wird auf einen Speicherinhalt gerade zugegriffen, so ist es relativ wahrscheinlich, dass der nächste Zugriff in dessen Nachbarschaft erfolgen wird.

Speicherstruktur:



Prinzip: SRAM ermöglicht schnellen zugriff der CPU auf Daten/Befehle. Bei miss, werden Daten aus dem Hauptspeicher nachgeladen.
Direct Mapped Cache: Cache Index → Teil der Adresse, bestimmt Position im Cache. Für Speicherzelle im DRAM ist Position im Cache fixiert → direct mapped. Restliche Adressbits werden zu den Daten gespeichert → Cache Tag. Valid Bit dient der Erkennung bisher unbenutzter Positionen. Für jeden Blcok → eine mögliche Position im Cache. Beispiel (Ü7).

Aufgabe 3: Blockgröße von Caches / 1

Nehmen Sie einen *Direct-Mapped* Cache mit 8 Blöcken und einer Blockgröße von genau einem Datenwort an. Ergänzen Sie in der nachfolgenden Tabelle, ob für die angegebene Adresse der Zugriff jeweils ein *hit* oder *miss* ist. Geben Sie weiters an, aus welchem Cache-Block die Daten gelesen werden und tragen Sie bei einem *miss* zusätzlich ein, von welchen Adressen die Daten in diesen Block geladen werden. Zählen Sie am Ende die Anzahl der *miss*-Einträge und vergleichen Sie diese in der Übung mit dem Ergebnis von Aufgabe 4.

Hinweis: Es wird angenommen, dass der Cache zu Beginn leer ist.

Adresse	hit/miss	Cache-Block	Inhalt
0	miss	0	0
4	miss	4	4
1			
6			
12			
0	hit	0	
5			
1			
7			
13			
2			
4			
7			
6			
14			
2			
5			
15			
7			
15			
# misses			

Cache Performance: westentlich für Gesamt Performance. Entscheidend dafür → Hit Time, Hit Rate, Miss Penalty. Je höher Performance der CPU, desto größer der Einfluss der Cache Performance.

Hit / Miss: *Hit → WriteThrough* (= aktualisiere Cache und aktualisiere auch sofort den Hauptspeicher.) ; *Copy Back* (= aktualisiere Cache und markiere den Block „dirty“; aktualisiere Hauptspeicher erst später, wenn Block aus Cache entfernt)
Miss → Write Around (= Ignoriere Cache und schreibe direkt in den Speicher, meist in Kombination mit WT); *Fetch on Write* (= Ersetze aktuellen Inhalt des Cache und aktualisiere Tag; Falls Blockgröße >1, lade restlichen zum Block gehörigen Daten aus Hauptspeicher nach.)

Fully Associative Cache: Jeder Block darf auf jede beliebige Position im Cache gelegt werden, optimale Hit Rate, bei direct mapped nur mäßige, aber komplizierte Verwaltung, Suche im gesamten Cache.

N-way associative Cache: Für jeden Block gibt es N versch. Möglichkeiten der Platzierung im Cache. Vernünftige Hit Rate.

Direct mapped:Eindeutig über Index; Set Associative: Auswahl innerhalb von Set; Fully Associative Cache: beliebige Auswahl.

Ersetzungsstrategien: LRU (das zuletzt nicht benutzte LOL deutsch); LFU (am wenigsten benutzt); reference bit; random; FIFO(first in, first out).

Peripherie:
Chipsatz: Befindet sich auf dem Mainboard, verbindet CPU an Peripherie, Controller für periphere Geräte.

Northbridge (= CPU, PCIe x16)
Southbridge (= S-ATA, USB, Super I/O, BIOS Flash, System Management Bus, HD Audio, PCI 32bit, PCIe x1(=PCIe- IDE Adapter -IDE- IDE Devices)).

Controller und Co Prozessoren: Controller → I/O Aufgaben, DMA Controller → kontrolliert Zugriff der I/O Controller auf den Hauptspeicher, ohne das CPU eingreifen muss. Arten (= Externspeicher(S), Serial I/O(S), Netzwerke (S oder PCIe), Mathe Co Prozessor(in CPU), Grafik Co Prozessor (auf Graka, PCIe x16 oder CPU, i3,i5).

Alte Schnittstellen: IDE/ATA für Speichermedien (HDD,CD), heute per Adapter über PCIe. PCI (parallel, 32bit), evtl. Sound oder TV Karten, neuere HW über PCIe oder USB. Super I/O für Legacy Schnittstellen, wie V24/RS232, Centronics (Parallel Printer Port) oder PS2.

Schnittstellen aktuell
Intern: PCIe, SAS (über PCIe), SATA; Extern: eSATA, USB, HDMI, VGA, DVI, DisplayPort, optisches Audio, etc...

PCIe: Serielle Point 2 Point Verbindung. Vollduplex pro Link mit min. 250 MB/s. Hot Plug. SATA: Serielle P2P Verbindung mit jeweils einem Leitungspaar für Sende, bzw. Empfangsrichtung; Hot Plug; Für Speichermedien; zur Zeit max. 6Gbit/s.

Externe Schnittstellen nach Art: Monitor (= VGA, DVI, HDMI, DP), Bus-artig: USB,

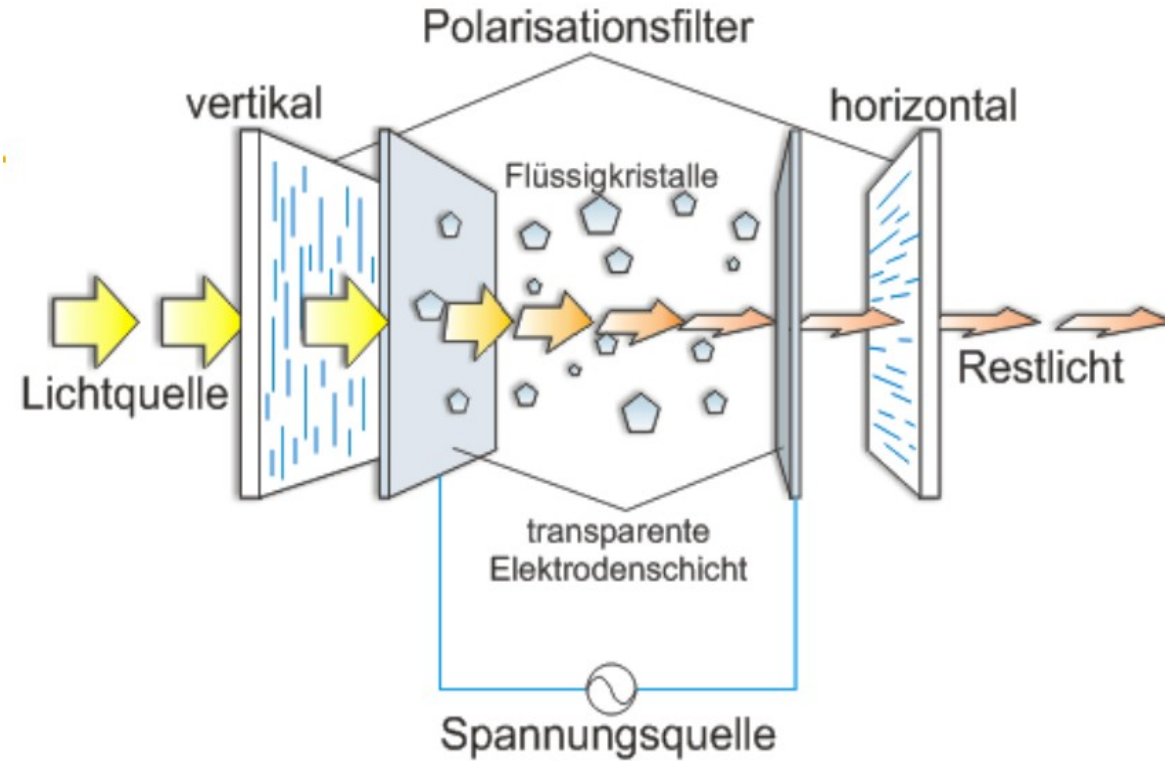
Firewire, eSATA(kein Bus). Netzwerk LAN, WAN. Funk: WLAN, Bluetooth. Optisch: IrDA, S/PDIF, HD-Audio. Seriell/parallel, analog/digital.

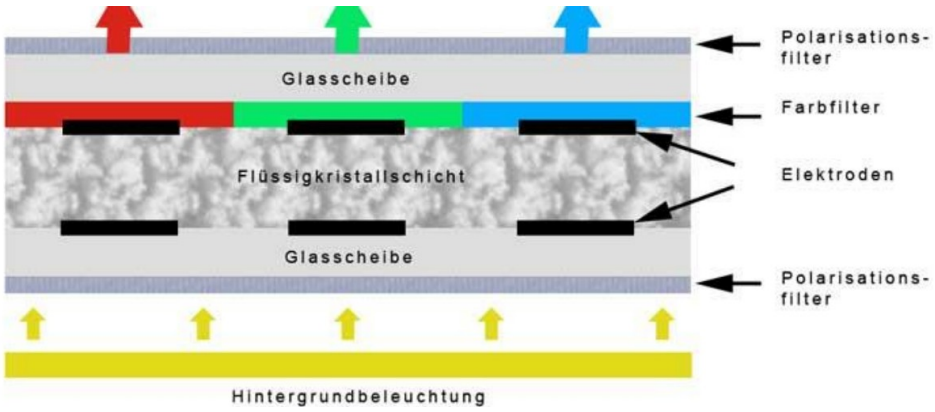
USB: Seriell, Hot plug, max 127 Geräte. Je Chipsatz unterschiedliche Anzahl an USB Ports.

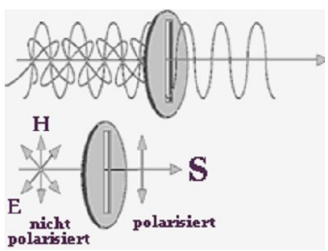
Kommunikation: Logisch sternförmiges Netz (Phys. Baum). Controller steuert Kommunikation mit externem Device, schreibt liest Gerätepuffer → polling. Kommunikation durch Anstecken, Anmelden des Gerätes als Device 0, Controller weist Adresse zu, liest Spez. Des Gerätes (Anzahl + Größe der Puffer, I/O, Freq. Für polling, Strombedarf, etc.).

USB 2.0 (1 Datenleitungspaar) – USB 3.0 10 x so schnell, mit nur 2 zusätzlichen Datenleitungspaaren. Bei USB 3.0 entfällt Polling, ist ab und aufwärts kompatibel, Höhere Stromstärken.

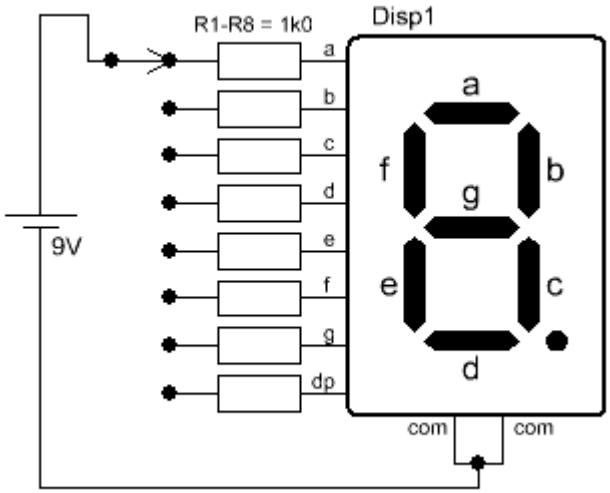
LCD Display: Flüssigkristalle aus druchsichtigen Molekülen, die Lichtwellen polarisieren. Elektisches Feld orientiert diese Moleküle.



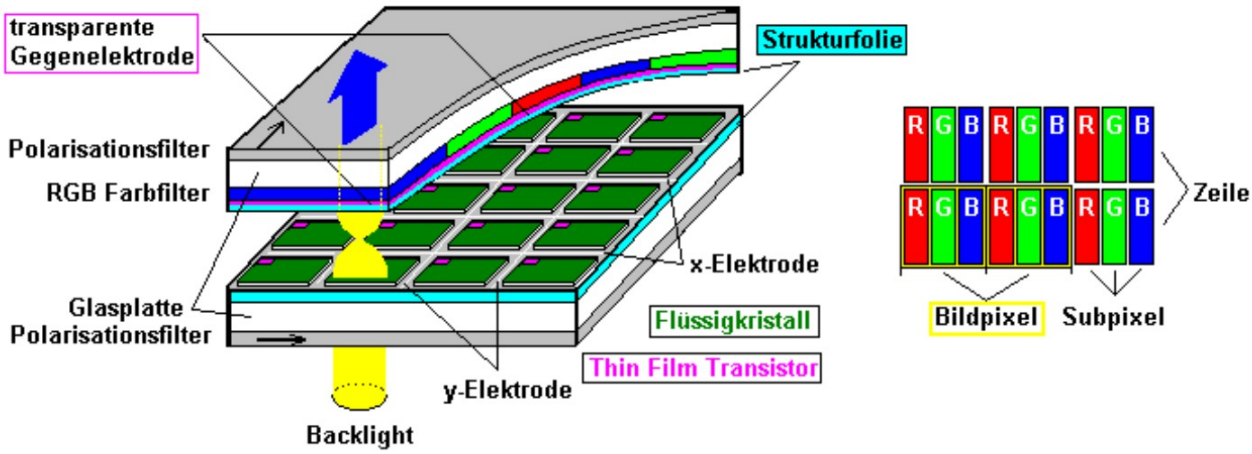


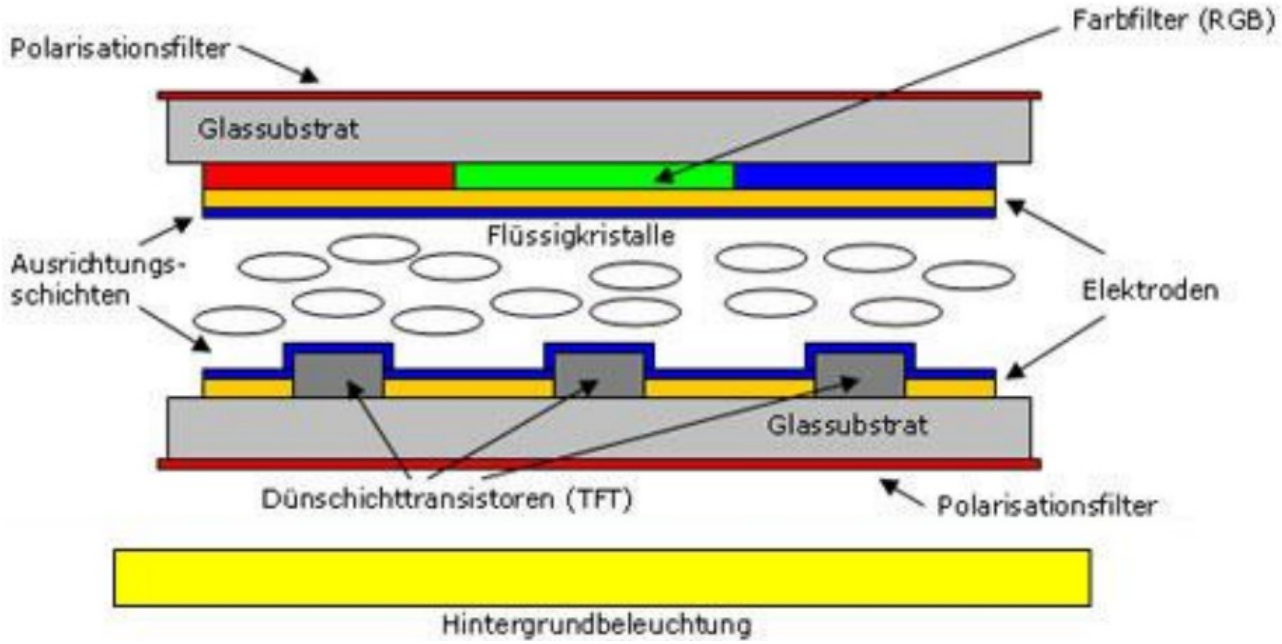


Segmente können unabhängig voneinander Helligkeit ändern; steuern mit Spannung Ausrichtung der Kristalle und ändern damit die Durchlässigkeit für polarisiertes Licht; sind in gleichmäßigem Raster angeordnet (Pixel); speziell 7 Segment Anzeige →



Weiterentwicklung → Active Matrix Display (= verwendet zur Ansteuerung eine Matrix aus **TFTs** (thin film transistor).





Jedes Subpixel wird mit einem TFT Gesteuert. Auflösung FULL HD → 1920x1080 x3 für Subpixel → > 6 Mio.

Farbtiefe: n bit / Farbe → n = 3 x n → 2^n Farben.
Bildwiederholfrequenz: schon relevant.

Reaktionszeit: Grau zu Grau Zeit, praktisch 20ms (min. 2 ms).
Leistungsaufnahme: je nach Auflösung, BG Beleuchtung ca 55W.

Fertigungsfehler: Ständig leuchtend (= Transistor kann nicht eingeschaltet werden); Ständig schwarz (= Transistor hat Kurzschluss oder kann nicht abgeschaltet werden).

# Fehler pro 1 Mio Pixel	Fehlertyp 1	Fehlertyp 2	Fehlertyp 3	Fehlercluster Typ 1/2	Fehlercluster Typ 3
Fehlerklasse ↓	ständig leuchtendes Pixel	ständig schwarzes Pixel	1-2 defekte Subpixel, ständig leuchtend oder ständig schwarz	mehrere Fehler vom Typ 1 oder 2 in einem Bereich von 5 × 5 Pixeln	mehrere Fehler vom Typ 3 in einem Bereich von 5 × 5 Pixeln
I	0	0	0	0	0
II	2	2	5	0	2
III	5	15	50	0	5
IV	50	150	500	5	50

Ansteuerung Monitor über onboard in CPU (i3,i5) oder **PCIe Graka**.

Graka: Für Berechnung von Grafik verantwortlich, kann auch für math. Berechnungen genutzt werden. Minimal → onboard. Westentliche Bestandteile → GPU, Grafikspeicher inkl Framebuffer (Größe bestimmt Farbtiefe und Auflösung); Display Anschlüsse.

Display Anschlüsse: VGA (= Video Graphics Array; analoge Schnittstelle, bei hohen Auflösungen hohe Anforderung an Kabel und Signalaufbereitung; Bei digitalen Displays nicht optimal da überflüssige Wandlung (=digital(PC) → analog(Schnittstelle) → digital (Display). Danke VGA !

DVI (Digital Visual Interactive), für TFT Monitore ab 99. analoge und digitale Daten gleichzeitig. Für digitale Daten → TMDS Standard. Bei Verbindungsaufbau (Single Link → 3.72 Gbit/s; Dual Link 7,44GBit/s und Auflösung, Bittiefe, Bildfrequenz, Verschlüsselung); hot plug;

TMDS (Transition Minimized Differential Signaling): Standard für unkomprimierte, digitale MM Daten. Zur Elimination von elektromagn. Störungen, wie sie bei der analogen Übertragung auftreten können. Max. 165 MP/s bei 60Hz → max. Auflösung 1600 x 1200 (UXGA) bei CRT, 1920 x 1200 (WUXGA) bei digitalen Displays; 3 differenzielle TMDS Leitungen mit je max. 1,650 Gbit/s für Daten; 3 differenzielle Taktleitung mit der Taktfreq. Von 1/10 der Datenrate (max. 165 MHz). Max. Auflösung DVI 2560 x 1600 WQXGA Pixel bei 60 Hz.

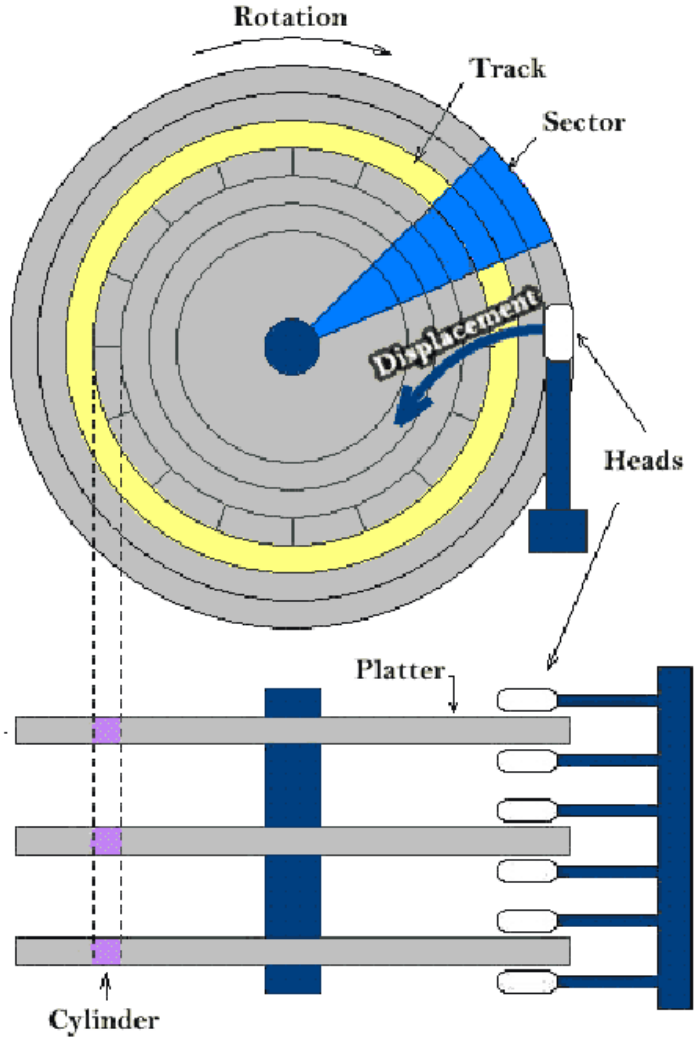
HDMI (High Definition Multimedia Interface): seit 2003 soll ab 2015 Standard sein. Hot plug; Audio und Video; Kopierschutz DRM; volldigitale Übertragung → TMDS; 5 versch. Kabeltypen zw. 2.2 Gbit/s – 8,16 Gbit/s. 1080 i mit 340 MHz, 720 p mit 75MHz.

Display Port: Für Computer Monitore, seit 06. Statt DVI und VGA bis 2015. DRM, auch als interne Schnittstelle gedacht; 6, 8, 10, 12 oder16 bits/Farbkomponente (Subpixel); halbduplex AUX Kanal mit 720 Mbit/s. Kommunikation mit Display über Monitor Command Control Set (MCCS); Für USB Verbindungen wie Kamera, Mikrofon, Touch; unterstützt stereoskopische 3D Formate; bis zu 63 Video und Audio Streams gleichzeitig; Nachteil bei hoher Auflösung wie 4096 x 2560 nur 2m Kabel.

Datenraten:

Anforderung	Datenrate in Gbit/s	
HD-Video (720p 60 Hz 24b/px)	ca. 1,30	
FullHD-Video (1080p 60 Hz 24b/px)	ca. 3,00	
Schnittstelle		
PCIe x16 2.0/3.0	64,00	96 Gbit/s
USB 2.0/3.0	0,48	5 Gbit/s
eSATA 2.0/3.0	3,00	6 Gbit/s
DVI single/dual	3,72 bzw. 7,44	
HDMI 1.4a	2,2 bzw. 8,16	
DisplayPort 1.2	1,296 bis 17,28	

Magnetspeicher:
Plattenstapel mit mehreren magnetisierbar beschichteten Scheiben aus Glas, Aluminium oder Magnesium; Beschreiben / Lesen mittels Schreib- Lesekopf; Schreiben (→ Stromdurchflossener Schreibkopf erzeugt Magnetfeld in bestimmte Richtung) Lesen (→ Ausrichtung des Magnetfeldes auf dem Trägermaterial induziert Lesespannung [+,-].



Longitudinales Aufzeichnungsverfahren (Waagrecht auf Trägeroberfläche; Datendichte 31 Gbit/cm²) vs Perpendikuläres Aufzeichnungsverfahren (senkrecht zur Trägeroberfläche; 155 Gbit/cm²)→ Bitsong.

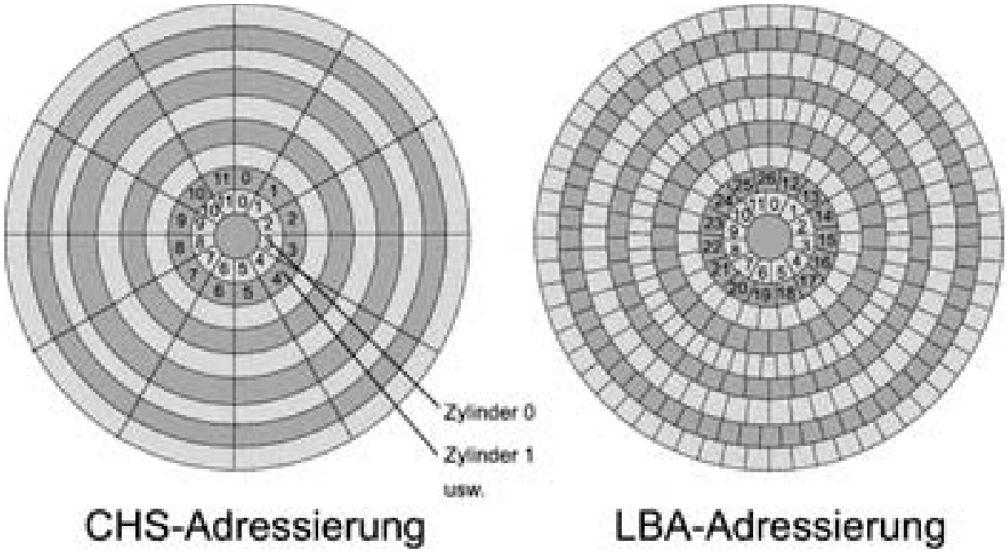
Aufbau: Übereinander liegende Scheiben mit konzentrischen Spuren, tracks; Zylinder: die jeweils aufeinanderliegenden tracks; Addressierung über → CHS C: Cylinder(welche tracks); Head (welche platter+Seite); Sector (welcher Abschnitt).

Kapazität: T → Anzahl der Tacks pro platter Seite; S → Anzahl der Sektoren pro Track; K → Anzahl der Köpfe pro Festplatte. Anzahl der Sektoren auf einer platter Seite → T*S; Anzahl der Gesamtsektoren einer Festplatte: T*S*K; Gesamtkapazität: Anzahl der Gesamtsektoren * 512 Bytes pro Sektor → T*S*K*512 B

Zone Bit Recording: Verschwendung durch Einteilung der Plattenoberfläche in

gleiche Anzahl Sektoren pro Kreisabschnitt → Deswegen Zonenaufzeichnung: mit variabler Anzahl von Sektoren mit gleichbleibender Sektorlänge → innen weniger Sektoren als außen; LBA.

LBA: Logische Blockadressierung, Adressierung der Blöcke unabhängig von Festplattengeometrie; Blöcke durchgehend beginnend bei 0, nummeriert; ein LBA Block = Sektor bei CHS.



Kenngrößen:
Rotationsgeschwindigkeiten: 5400 – 7200 rpm Consumer; 7200 – 15000 rpm Server
Zugriffszeiten: 12 – 8 ms Consumer; 8 – 3 ms Server
Cache: 8-64 MB
Interface: SATA, USB, SAS, IDE/ATA

Optische Speicher: CDROM / CDRW (650-879MB); DVD ROM / DVD RW / DVD RAM (SL 4.7, DL 8.5 GB); Blu Ray (SL 25GB, DL 50).

CD Rom Aufbau: halbdurchlässiger Spiegel lässt Laserlicht durch auf CD Oberfläche, diese lenkt rücklaufendes Licht zur Fotodiode, diese erkennt Signal.

Datencodierung CD: Hierarchische, hochredundante Codierung der Daten; Fehlerkorrektur auf 3 Ebenen (Channelbits → unterste Ebene, EFM Codierung (8-14 [DVD 8-16, BluRay eigene Codierung]); Frames → 42*14 Bit = 588 Bit; eventuell keine Korrektur; Sektoren → 98 Frames); Redundanz in einem Frame → 588 Bit, nur 192 Bit Nutzdaten; Speicherbedarf eines Sektors → 98*588 Bit = 57624 Bit (7203 Byte).

Datenraten:

	CD-ROM	DVD-ROM	BLU-RAY
Markteinführung	1982	1996	2006
Datenrate	1,228 Mbit/s	11,08 Mbit/s	36 Mbit/s
Laserwellenlänge	780 nm Infrarot	680 nm Rot	405 nm Violett
Spurabstand	1,6 µm	0,74 µm	0,32 µm
Ø Laserpunkt	2,1 µm	1,3 µm	0,6 µm

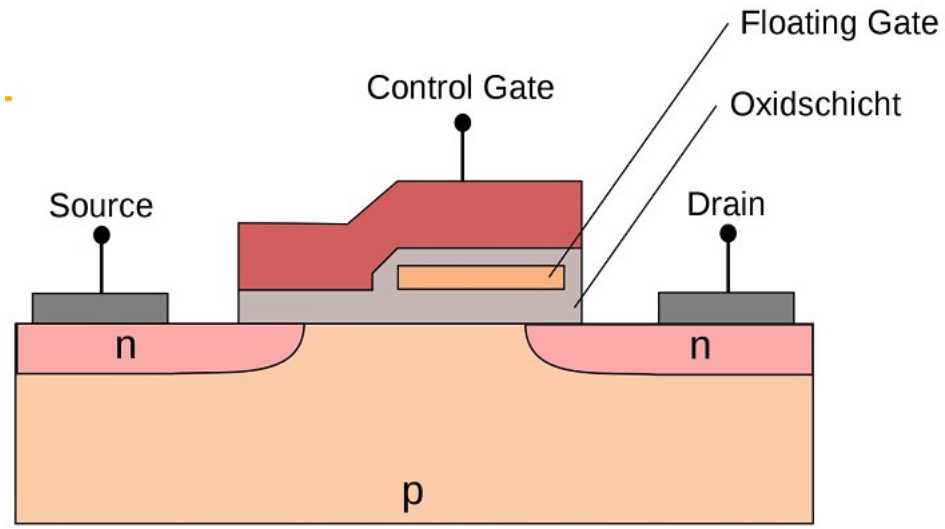
Flash Speicher: Kommt aus Bereich von DigiCam, Speicher Karten für mobile Devices, erschütterungsresistent, Bios Flash, aber auch schon SSDs; sind EEPROMs, „Flash“ wegen des Block weisen Löschvorgangs.

Pro/Kontra: Schneller sequentieller Zugriff, hohe Transferdaten; Stoßfestigkeit; großer nutzbarer Temperaturbereich – Lebensdauer abhängig von Löschzyklen → MLC: einige 10000, SLC: einige 100000.

Controller: Verteilt Löschvorgänge möglichst gleichmäßig; Reservezellen ca 7-10%.

TRIM: BS sollte TRIM unterstützen, um der SSD mitzuteilen, welche Daten nicht mehr in Verwendung sind.

Funktionsprinzip: Speicherung der einzelnen Bit erfolgt als Ladung im Floating Gate des Floating Gate Transistors; durch diese 2 Zustände kann die Informationsmenge eines Bits gespeichert werden; zum Löschen und Beschreiben → Spannung zwischen Gate und Substrat angelegt, Richtung ist entscheidend.



NAND Flash: in Reihe geschaltete FG Transistoren, höhere Speicherdichte als NOR, billiger, Adressierung erfolgt Page- und blockorientiert; Page: 512-4096 Bytes bei größeren Flashbausteinen; Pages sind zu Block gruppiert, Blockgröße 16 – 512KB; Löschen →

blockweise, Lesen/Schreiben Pageweise.

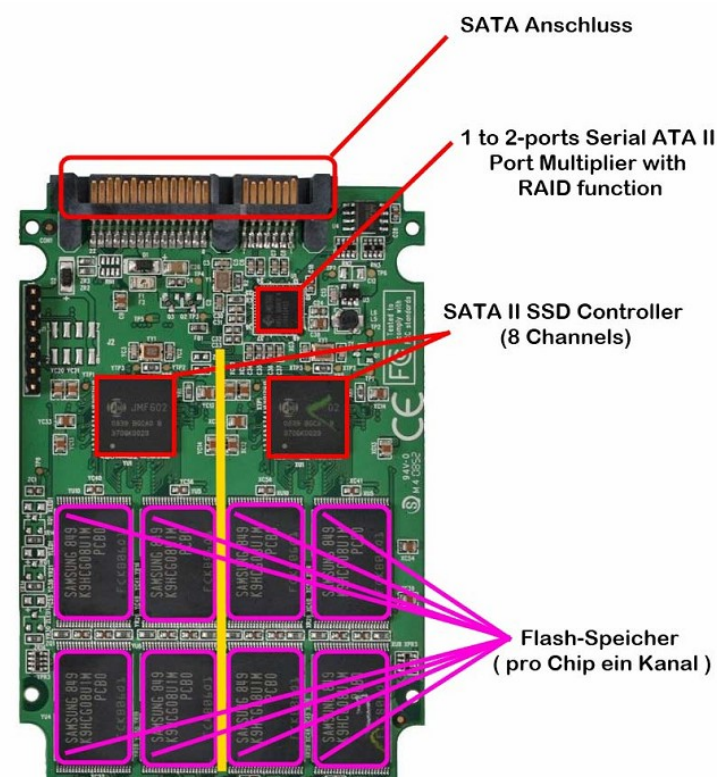
NOR Flash: Parallel geschaltete, einzeln adressierbare Speicherzellen, Byte; Geringere Speicherdichte, teurer, schneller lesbar als NAND. Löschen Blockweise, Lesen/Schreiben Byteweise zu meist 64 KB Blöcken.

SLC (= Single Level Cell, 1Bit) → speichert nur 1 Bit im Floating Gate; durch einzelnes Spannungslevel robuster gegen Löschvorgänge; schneller beschreib- , lesbar als MLC.

MLC (= Multi Level Cell, 2-4Bits) → multiple Spannungslevel für 2-4 Bits, anfälliger für Fehler durch Löschen.

SSD Aufbau:

- Flash-Controller meist mehrkanalig ausgelegt (paralleler Zugriff, hier 8)
- Optional mehrere Flash-Controller möglich (hier 2)
- Interne Organisation als RAID möglich
- Restliche Speicherchips hinten aufgebracht



Netzwerktypen: PAN (Personal Area Network), LAN (Local Area Network), MAN (Metropolitan Area Network), WAN (Wide Area Network).

Übertragungstechniken: Circuit, Packet Switching

Struktur: Point2Point, Broadcast

Standardisierungen: ITU (International Telecommunication Union), ISO (Internation Organization for Standardization), OSI (Open Systems Interconnection)

IEEE (Institute of Electrical and Electronics Engineers):

Nummer	Thema
802.1	Übersicht/Architektur LANs
802.2	Logische Verbindungssteuerung
802.3	Ethernet
802.4	Token Bus
802.5	Token Ring
802.11	Drahtlose LANs
802.15	Persönliche Netze
802.16	Drahtlose Breitbandnetze

Dienste:

Einheiten: aktive Elemente auf einzelnen Schichten (Software → Prozess; Hardware → intelligenter E/A Chip).

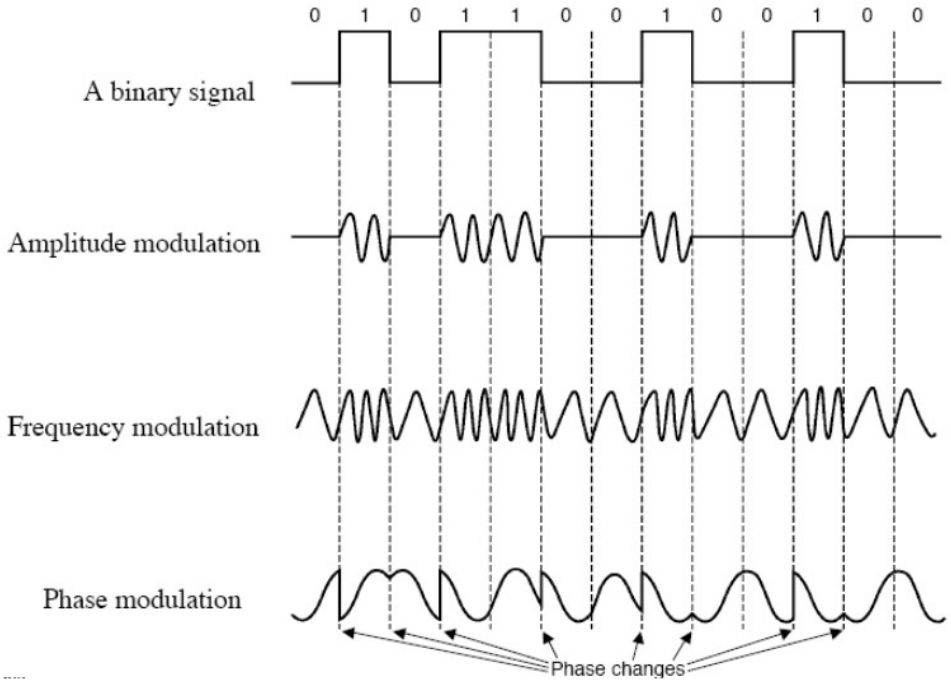
Partnereinheiten: Einheiten der gleichen Schicht
Einheiten auf Schicht n implementieren einen von Schicht n+1 benutzten Dienst (Schicht n → Dienstanbieter (Service Provider); Schicht n+1 → Dienstnutzer, Service User).
Dienste werden auf Dienstzugriffspunkten (ServiceAccessPoints, SAPs bereitgestellt.
SAPs → eindeutige Adressen.

Dienst Operationen: Anfrage, Request (Einheit fordert den Dienst an, eine bestimmte Aufgabe zu erfüllen); Anzeige, Indication (Einheit wird über ein Ereignis informiert); Antwort, Response (Einheit möchte auf Ereignis antworten), Bestätigung, Confirm (Antwort auf eine frühere Anfrage).

OSI Schicht	Einordnung	DoD Schicht	Einordnung	Protokollbeispiel	Einheiten	Kopplungselemente
7. Anwendungen	Anwendungsorientiert	Anwendung	Ende zu Ende	HTTP,FTP, HTTPS,SMTP,LDAP,NCP	Daten	GateWay, Content Switch, Layer-4-7 Switch
6. Darstellung						
5. Sitzung						
4. Transport	Transportorientiert	Transport		TCP,UDP,SCTP,SPX	TCP = Segmente, UDP = Datagramme	
3. Vermittlung		Vermittlung		ICMP,IGMP,IP,IPsec,IPX	Pakete	Router, Layer-3 Switch
2. Sicherung		Netzzugriff	Punkt zu Punkt	Ethernet, Token Ring, FDDI, ARCNET	Frames	Bridge, Switch
1. Bitübertragung					Bits	Repeater, Hub

Bitübertragungsschicht: Übertragung von Bits, Strecke → Koaxialkabel, Lichtleiter, Funkstrecken; Stecker ; Übertragungstechnik: Modulationsarten, Codierungstechniken, Bittaktregeneration, Wortsynchronation.

Modulationsarten:



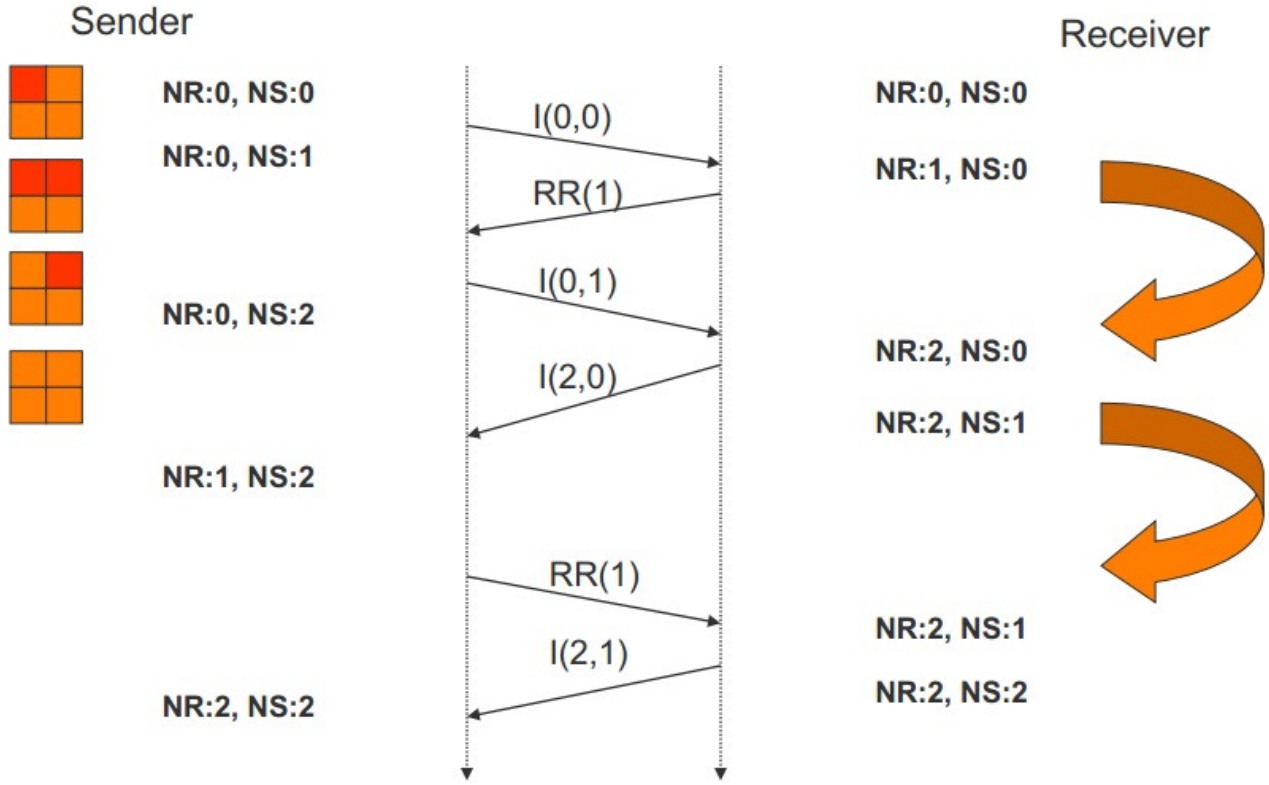
Sicherungsschicht: Dieser Layer bildet Grundstein zur fehlerfreien Datenübertragung.

- Zusammenstellung von Datenrahmen (Frames)
- Header, Adressierung, Länge des Frames
- Nutzdaten
- Trailer, Datensicherung (zB CRC checksum)
- Anfang, Ende (Zeichenzählung, Anfangs und Endzeichensetzung, Flags)
- Unbestätigte verbindungslose Dienste (Empfang von Frames nicht bestätigt)
- Bestätigte verbindungslose Dienste (Rahmen werden einzeln bestätigt; Bei Ausbleiben der Bestätigung → wiederholte Versendung
- Verbindungsorientierte Dienste (Verbindungsaufbau, Rahmenübertragung, Verbindungsabbau)
- Fehlerüberwachung (Erkennung, Überwachung, + - Bestätigungen, Folgenummern, Timerüberwachung etc.)
- Flusssteuerung (Schneller Sender vs. Langsamer Empfänger)

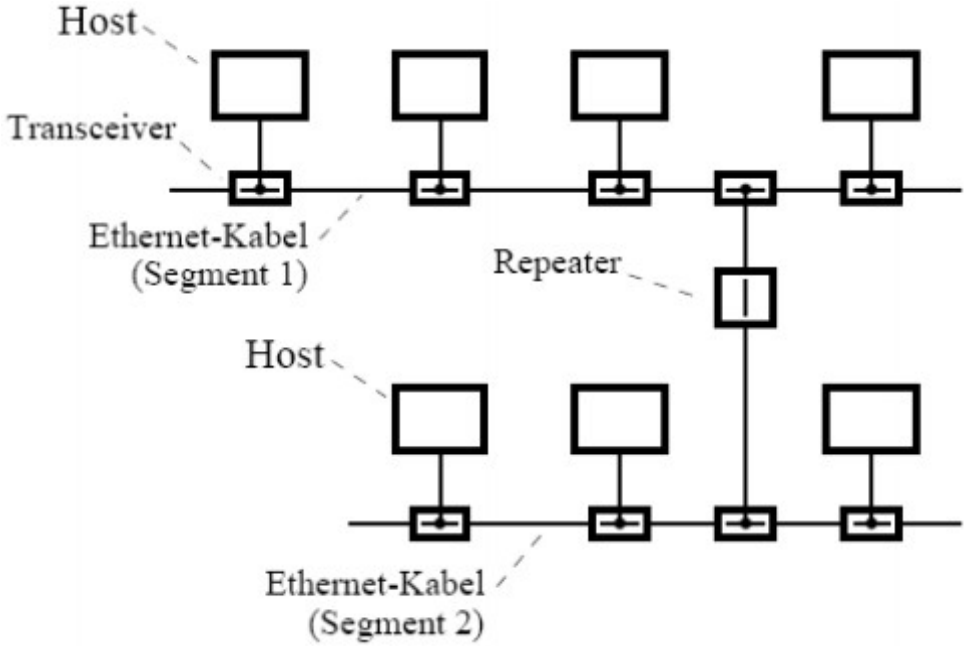
Protokolle d. Sicherungsschicht:
Rahmentypen:

Supervisory and Numbered Information Frame	Aktion	Richtung	Reaktion
Receiver Ready (RR)	Empfangsbereit	Sender ↔ Receiver	Adaption Zähler
Receiver Not Ready (RNR)	Überlauf oder unerwarteter Rahmen	Sender ↔ Receiver	Keine Zähler-adaption
Information Frame (I)	Übertragen von Daten nr Rahmen empfangen ns Rahmen versendet	Sender ↔ Receiver	I(nr,ns) RR(nr) RNR(nr)

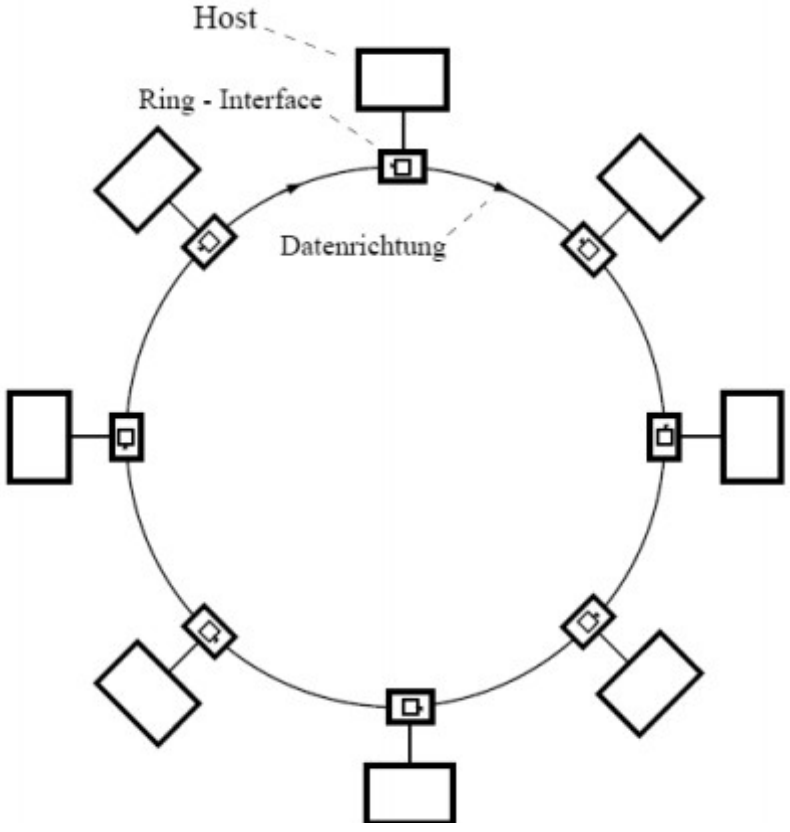
Sliding Window Protocol:



Medium Access Control (MAC):
IEEE 802.3 Carrier Sense Multiple Access with Collision Detection
Zugriff versch. Stationen auf Übertragungsmedium im Zeitmultiplexverfahren (TDMA); Listen while Talk; Jam Signal



802.5 Token Ring
Kollisionsfreie Übertragung von Paketen; Ring- Topologie; Token Passing; Tokenhaltezeit; Ringwartung (= Token Verlust; Mehrere Token; Monitor)



Vermittlungsschicht: Pakete vom Ursprung ans Ziel bringen (Point2Point). Wegewahl (Routing) innerhalb Netzwerke, sowie Netzwerkgrenzen hinweg (Datagrams, Virtuelle Verbindungen).

Transportschicht: Logische Kanäle (Multiplexen und Demultiplexen); Zerlegung von Nachrichten in kleinere Einzelpakete; Einhaltung Reihenfolge; Wiederholungsanforderungen; Fehlerkontrolle von Endsystem zu Endsystem.

Sitzungsschicht: Auf und Abbau von Sitzungen; Überwachung des Betriebs während Sitzung (→ Dialogsteuerung; Tokenmanagement); Synchronisation (→ Atomic Actions, Check Points)

Darstellungsschicht: Festlegung der Syntax und Semantik der zu übertragenden Daten (Codierung der Daten); Schutz der Daten vor Zugriff Unberechtigter (→ Auth., Verschlüsselung).

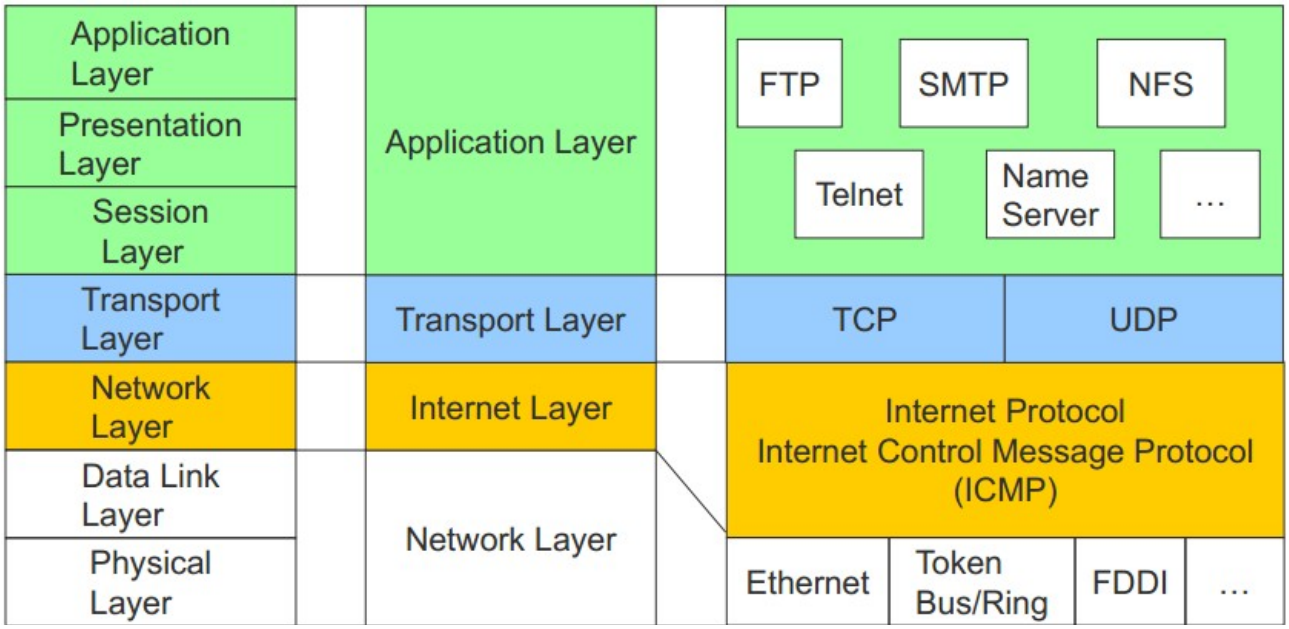
Verarbeitungsschicht: Anbieten von Diensten für eigentliche Applikationen (→ Email Service, Übertragung von Dateien, Remote Job Entry, Verzeichnisabfragen …)

Übertragungsmedien: Koaxialkabel; Verdrilltes (verdrillte Form bewirkt Reduzierung der elektromagnetischen Störungen) Adernpaar TP.

Netzwerkelemente:

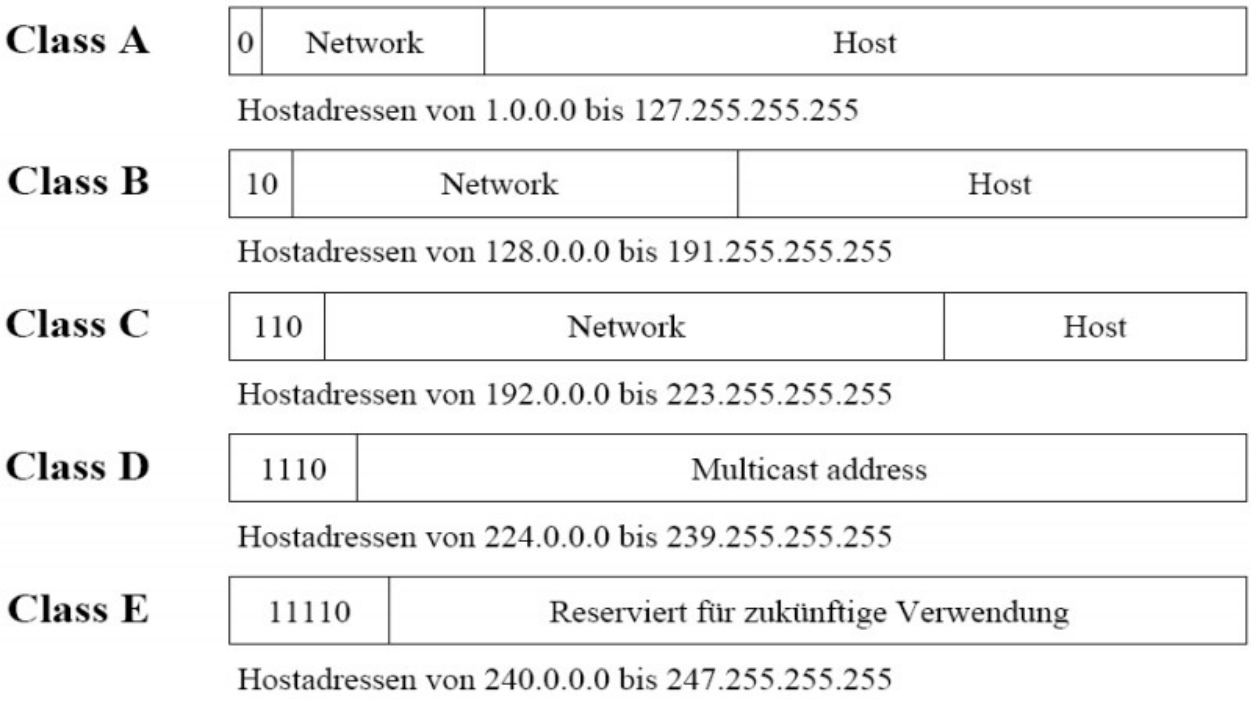
- Repeater (Schicht 1)
  - Keine Paket-Inspektion
  - Wird verwendet um entfernte Netze zu verbinden (Signalverstärkung)
- Bridge
  - Inspiziert eingehende Pakete (Schicht 2)
  - Stellt die Pakete nur zu, wenn die Empfänger-MAC sich auch wirklich auf der Gegenseite befindet
- Hub
  - Multi-Port Repeater
- Switch
  - Multi-Port Bridge
- Router
  - Arbeitet auf Schicht 3
  - Kann Routing-Tabellen verarbeiten um Pakete entsprechend zuzustellen.

IP Family:



IP Adresse: 32Bit numerisches 4-Tupel (128.130.2.3), beinhaltet Netzwerknummer und Host;
„This Host“: 0.0.0.0 ;
Limited Local Broadcast: 255.255.255.255;
Private Ips: 127.0.0.0-127.255.255.255,
10.0.0.0 – 10.255.255.255,
172.16.0.0 – 172.17.255.255,
192.168.0.0 – 192.168.255.255

IP Adressraum:



A: 126 Netzwerke mit jeweils 16M Hosts, /8 Netz
*B:*16384 N, jeweils 65534 H, /16 Netz
C: 2M Netzwerke, 254 H, /24 Netz

TCP (Transmission Control Protocol): Verbindungsorientiertes Protokoll, Sockets: IP Adresse und 16 Bitnummer (Port).

UDP (User Datagram Protocol): Verbindungsloses Protokoll; Sockets: IP Adresse und 16 Bitnummer (Port).

Subnetting: Netzbereiche gruppieren, unterteilen, um Routing zu ermöglichen.

Subnets: Werden durch Nr. und Subnetmask identifiziert, zB. 192.168.1.0, Mask: 255.255.255.0

Wie rechnen:
IP AND Subnetmask → Erste IP, Netzwerkname.
IP AND NOT(Subnetmask) → Letzter Teil, Hostnummer
IP OR NOT(Subnetmask) → Letzte IP, Broadcast

IP4 vs. IP6: Derzeit Adressbereich 4 Milliarden, Abhilfe schafft IP6 (16Byte Adressen als 8x2Byte Hex Tupel (128Bit)).

OS: Management von Ressourcen, Präsentation einer einheitlichen Schnittstelle für Anwendungen. Wichtige Gebiete: Prozessmanagement, Speichermanagement, Dateisystem, I/O ; System Calls → Funktionen die ein OS den Anwendungen zur Verfügung stellt.

Programm: Statisches Objekt im Dateisystem, Folge von Instruktionen, unveränderlich über die Zeit.
Prozess: dynamisches Abbild von Programm, läuft in einer bestimmten, veränderlichen Umgebung ab, Folge von ausgeführten Instruktionen auf der CPU, existiert nur für bestimmte Zeit, mehrere Prozesse können das selbe Programm ausführen.

Prozessablauf: Prozesse werden erzeugt durch Betriebsystem (init task) oder durch andere Prozesse. Prozesse werden freiwillig(exit()), aufgrund von Fehler, durch anderen Prozess beendet.

Prozesszustände: RUNNING(Prozess kann auf der CPU exekutiert werden), READY (Prozess kann bereit sein, exekutiert zu werden), BLOCKED (Prozess kann auf externe Ereignisse warten).

Zustandsübergänge: ausgelöst durch Prozess selbst, Scheduler, Ressourcenfreigabe.

Scheduling warum?: zuviele Prozesse, nur 1 CPU, OS muss zwischen Prozessen wechseln, Auswahl des neuen Prozesses → Scheduling.

Ziele: hoher Durchsatz, niedrige Antwortzeit, CPU Auslastung, Fairness.

Non preemptive vs preemptive: Ad non: Prozess kann CPU behalten bis fertig, oder

freiwillig Kontrolle abgeben. Preemptive: Prozess kann von OS gegen seinen Willen unterbrochen werden → Priority !

Scheduling: Angabe von Zeitpunkten, zu denen Prozesse zu laufen beginnen, max. 1 Prozess befindet sich im Zustand RUNNING, mehrere Prozesse können auf CPU warten (READY).

Synchronisation: Problem, zB. Zugriff auf gemeinsame Ressource wie zB. Drucker oder auch gemeinsame Variable.

Critical Section: Reihe von Operationen welche nicht unterbrochen werden dürfen.

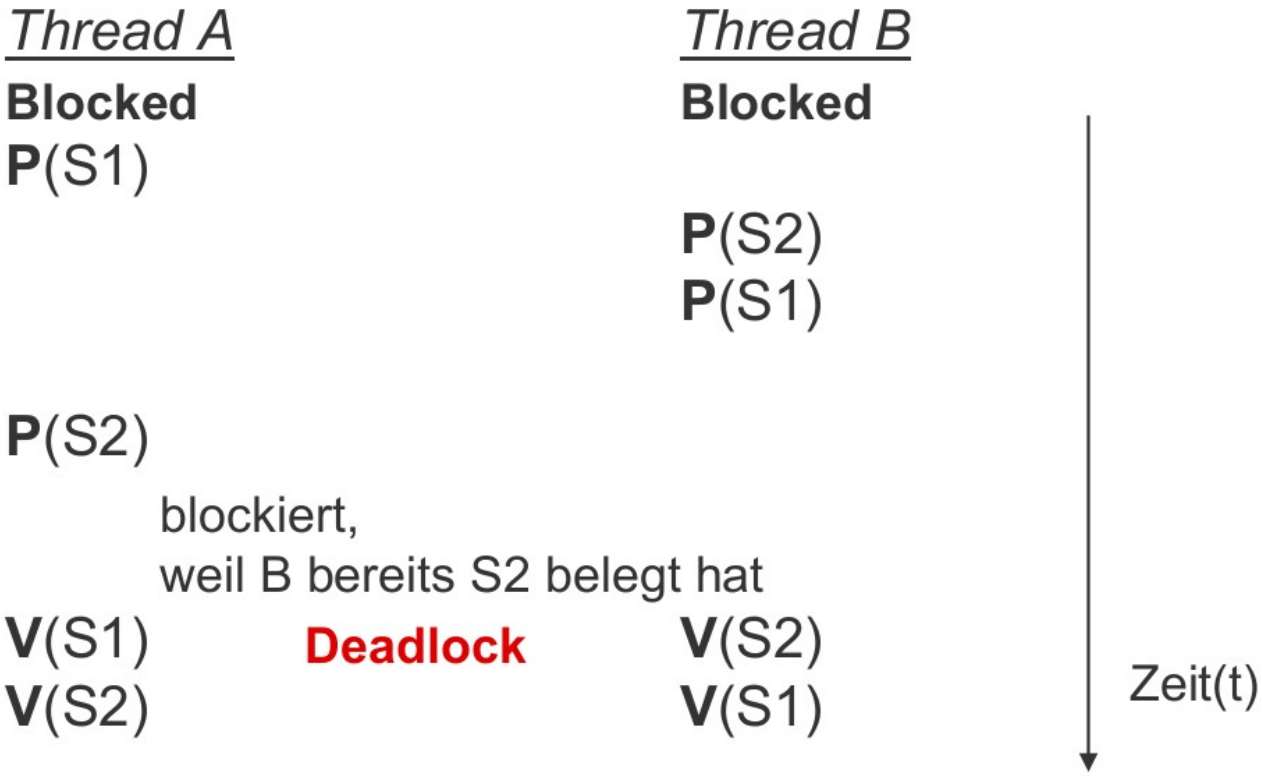
Mutual exclusion: Tatsache, dass sich nur ein Prozess in einer critical Section befinden darf.

Synchrone Methoden: Empfänger muss Nachricht aktiv abholen (Message Passing, Semaphore)

Asynchrone Methoden: Empfänger wird durch Nachricht unterbrochen (Signale, Interrupts)

Semaphore: dienen zur Synchronisation: Aufbau (= Counter, Queue), Operationen P (= Counter--, wenn <= blockiere den aufrufenden Prozess und trage ihn in die Queue ein) und V(= Counter++, wenn <0 deblockiere einen Prozess, der in Queue wartet).

Deadlock: Gruppe von Prozesses wartet jeweils auf Aktion eines anderen Prozesses innerhalb der Gruppe → Stillstand, da kein Fortschritt mehr erzielt wird.



Bedingungen für Deadlock: Mutual Exclusion (Zugriff auf Ressourcen ist exklusiv), Hold and Wait (Prozesse können Ressourcen anfordern, obwohl sie schon welche besitzen), No preemption (Ressourcen werden nur durch Prozesse freigegeben), Circular Wait (Zumindest 2 Prozesse warten aufeinander).

Maßnahmen zur Verhinderung: Deadlock Avoidance (Prozessüberwachung, Anforderungen die möglicherweise zu Deadlock führern, werden nicht erlaubt / verzögert), Deadlock Prevention (1 oder mehrere der 4 Bedingungen für Deadlock werden im System verboten), Deadlock Detection (Deadlock wird erkannt und beteiligte Prozesse beendet), Ignorieren.

Prozessortechnologie:
Befehlssatz: Desktop CPUs hauptsächlich CISC Prozessoren, seit 2004 64bit Standard.
Taktung: Taktraten stagnieren, Leistungserhöhung durch Verkleinerung, „schlaues“ Design, Pipelining etc.
Pipelining: super important, versch. Pipelinestufen für Integer, FP, etc.
Platz: Mehr Platz → niedrigere Geschwindigkeit.

N-Bit Prozessoren: besitzen n-Bit Register und n-Bit fähige ALU.

Instruktionen: Meist länger als Register/ALU Breite (zB. Mikro 16 hat 32bit Instruktionen); Instruktionen können variable Länge haben (SSE Instruktionen sind länger als normale).

Adressbus: Meist in Registerbreite weil oft durch Register adressiert wird.

Pipelining: Mehrere Pipeline Stufen, zB. Intel Pentium II → 11.

64 Bit (x64, AMD64, Intel64) → Größerer Adressraum, >4GB RAM und größere Register. Nachteile (= 64Bit OS notwendig, Treibersupport required).

Mehrkernprozessoren → Mehrere vollständige Hauptprozessoren auf 1 Chip; Multi Level Caches (= L1 immer Prozessornah, L2 je nach Anzahl der Kerne, L3 meist nur bei multiplen Kernen); Mehrkernprozessoren bringen erhöhten Instruktionsdurchsatz.

Probleme bei SMP (=Symmetric Multiprocessor): CPU Hopping, Cache Kohärenz(Daten können in mehreren Caches gleichzeitig vorhanden sein, ändert ein Kern seinen Cacheinhalt → Inkonsistenzen, WB/WT ähnlich problematisch; Lösung:Software[Compiler übernimmt Konsistenzsicherung, Nachteil → Cache nicht optimal genutzt], Hardware[Abgleich der Caches durch HW Controller, auf CPU integriert]), Speicherbus, Parallelisierbarkeit (Nicht alle Programme sind gleich parallelisierbar).

HyperThreading: Vollständiger Registersatz für jedes Element, Verwendet vorhande Pipelines parallel, wird vom OS als eigenständiger Kern betrachtet, Gewinn von 10-30% Leistung; Nachteile: Performance abhängig von Befehlsstruktur.