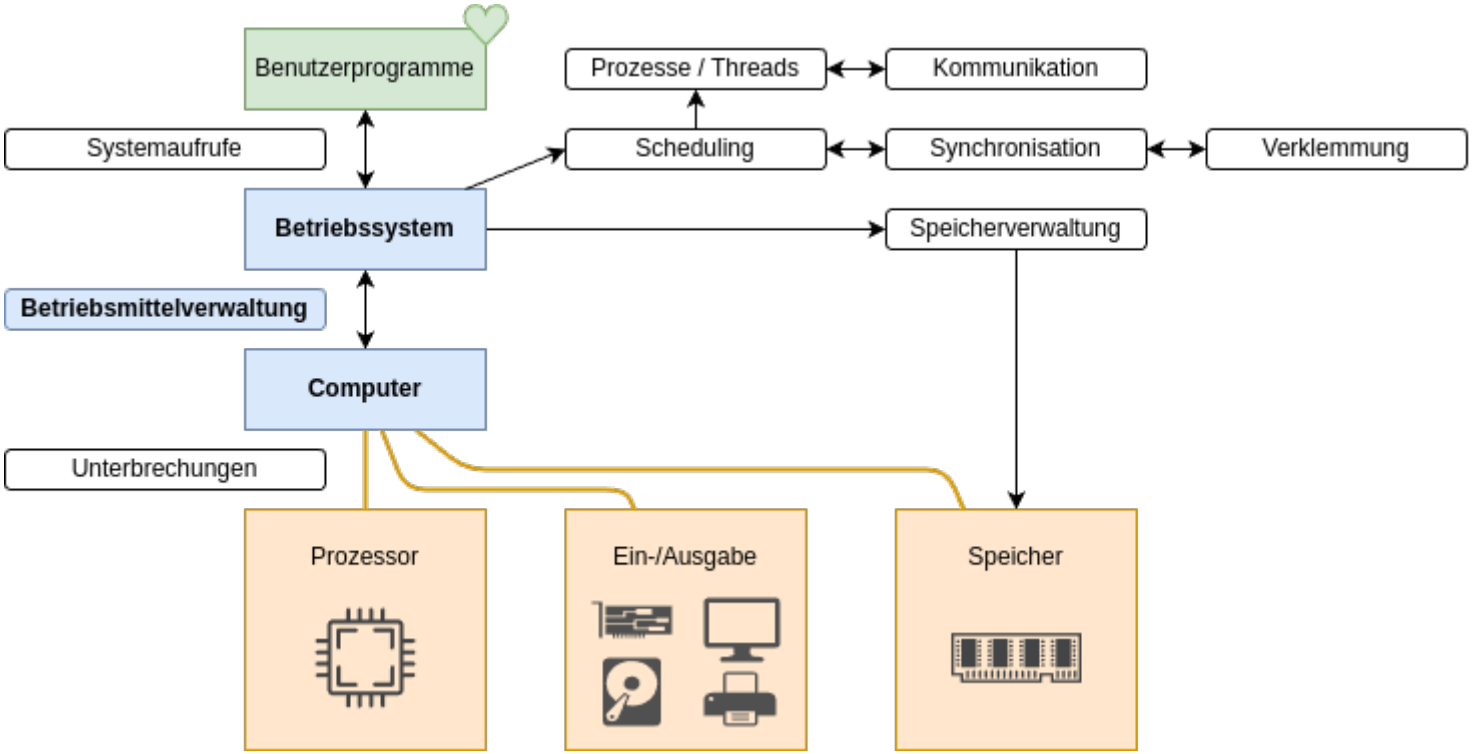
HOCHSCHULE FULDA
UNIVERSITY OF APPLIED SCIENCES

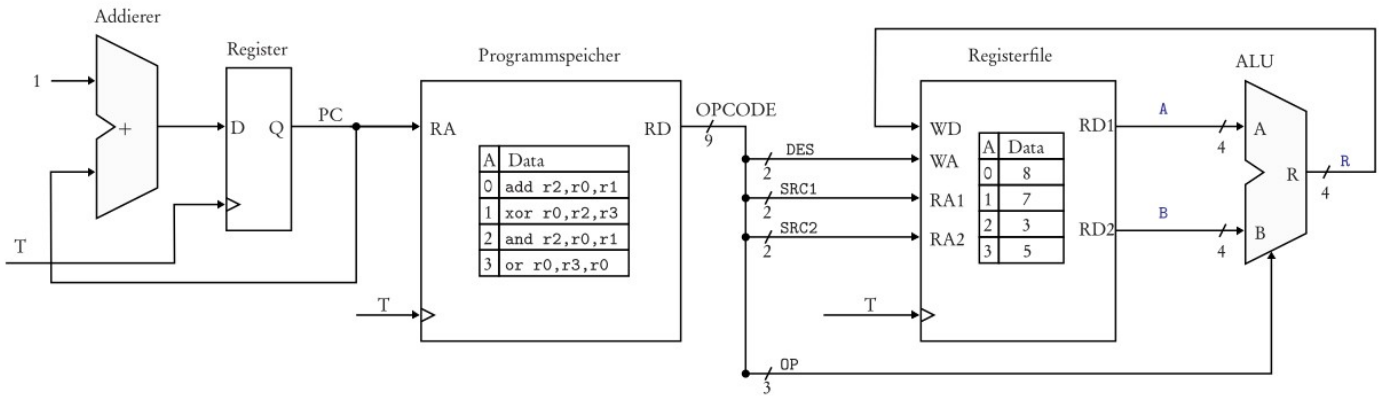
Betriebssysteme

Stage 2 – Rechnerarchitektur

Übersicht

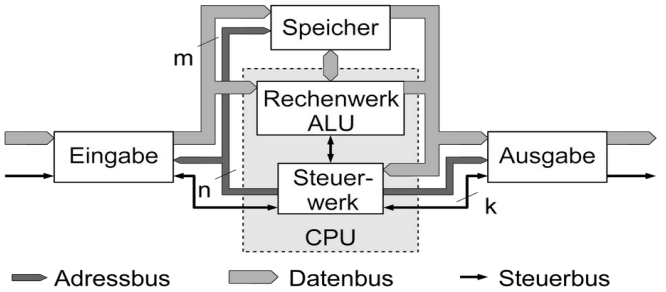

Benutzerprogramme
Systemaufrufe
Betriebssystem
Betriebsmittelverwaltung
Computer
Unterbrechungen
Prozesse / Threads
Kommunikation
Scheduling
Synchronisation
Verklemmung
Speicherverwaltung
Prozessor
Ein-/Ausgabe
Speicher

Rückblick: Minimalrechner



Von-Neumann-Architektur

Der Universalrechner

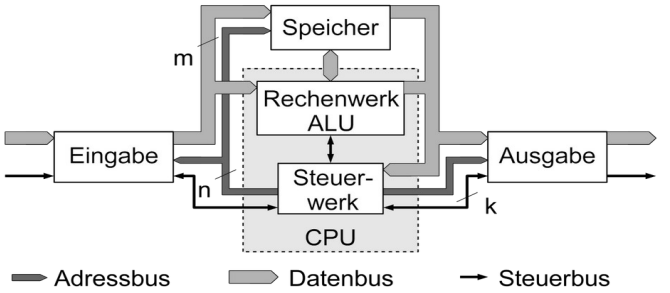


- Idee des Universalrechner
  - Nicht an ein “festes Programm” gebunden
  - Verfügt über Eingabe und Ausgabe (I/O)

- Von-Neumann-Architektur
  - Entwickelt durch John von Neumann (1946)
  - Die CPU besteht aus Rechenwerk und Steuerwerk
  - Die CPU kommuniziert mit dem Speicher, der Instruktionen und Daten enthält
  - Darüber hinaus gibt es Eingabe- und Ausgabegeräte
  - Programm und Daten werden binär kodiert und liegen im gleichen Speicher

- Konzept
  - Programme liegen als Folge von Maschineninstruktionen vor (in aufeinander folgenden Speicheradressen)
  - Die CPU führt die Instruktionen Schritt für Schritt aus
  - Programmzähler zeigt auf die Speicheradresse der jeweils nächsten auszuführenden Instruktion

Von-Neumann-Architektur: Bussystem

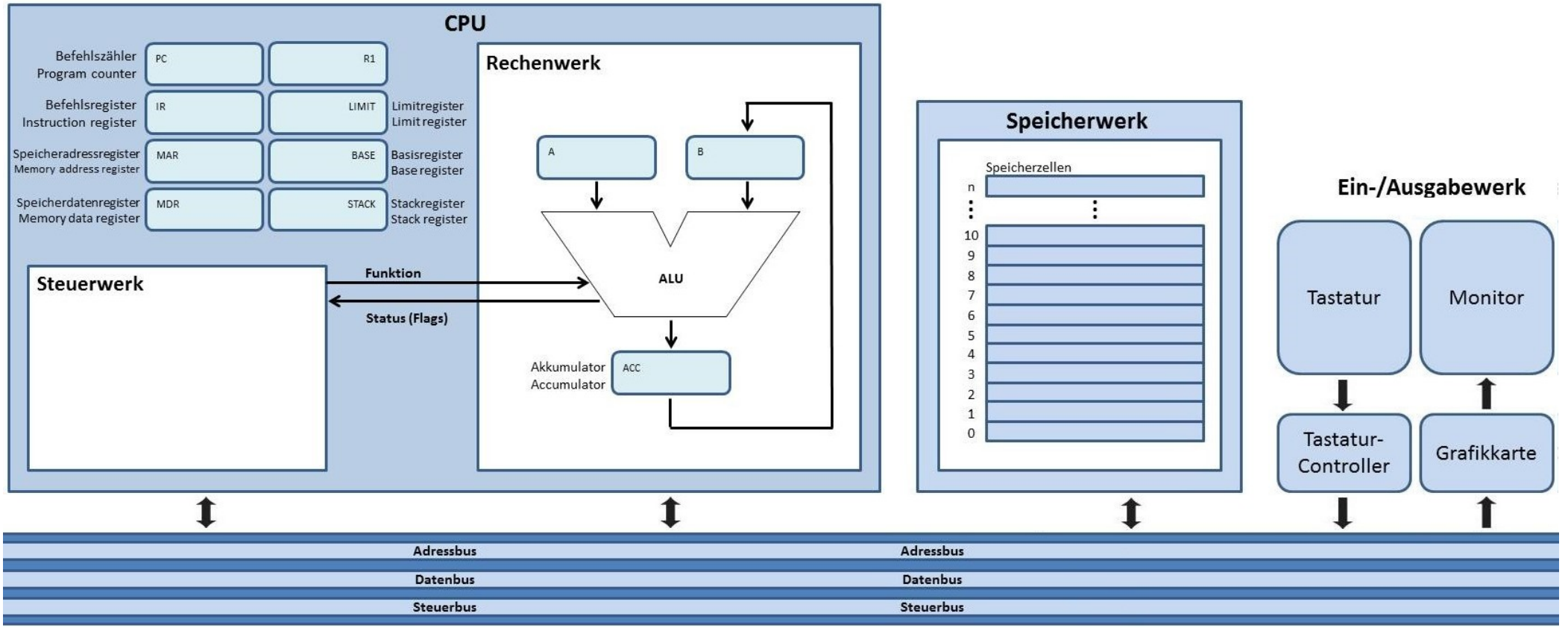


- **Datenbus**
  - Überträgt Daten zw. CPU, Speicher und Geräten
  - Anzahl der Leitungen → Gleichzeitig übertragbare Bits pro Takt
  - Moderne CPUs (Intel Pentium, Core i7, AMD64) nutzen 64-Bit
- **Adressbus**
  - Überträgt Speicheradressen zur Adressierung von RAM, ROM, I/O-Geräten
  - Anzahl der Leitungen (z.B. 32) → Anzahl adressierbarer Speicherstellen
  - Moderne CPUs (Intel Core i7, AMD64) nutzen 36-48 Bit
  - Moderne Betriebssysteme nutzen 64 Bit (logische) Adressen
- **Steuerbus**
  - Überträgt Kommandos von der CPU und Statusmeldungen der Geräte
  - Enthält z.B. Leitungen für Unterbrechungsanforderungen an die CPU (Interrupts)
  - Moderne CPUs besitzen I.d.R. nicht mehr als 10 Leitungen

Von-Neumann-Zyklus (Fetch-Decode-Execute-Cycle)

- Die CPU wiederholt kontinuierlich den Von-Neumann-Zyklus
  - **FETCH**
    - Nächster abzuarbeitender Befehl wird aus dem Speicher in das Befehlsregister geladen
  - **DECODE**
    - Das Steuerwerk “dekodiert” den Befehl in Schaltinstruktionen für das Rechenwerk
  - ***FETCH OPERANDS***
    - Falls es sich um einen Befehl handelt, der Operanden benötigt, werden diese aus dem Speicher geladen
  - **EXECUTE**
    - Das Rechenwerk führt den Befehl aus
  - ***UPDATE PROGRAM COUNTER***
    - Der Befehlszähler wird auf den nächsten Befehl gesetzt
  - **WRITE-BACK**
    - Das Ergebnis des Befehls wird gespeichert (in einem CPU-Register oder Hauptspeicher)

Von-Neumann-Architektur



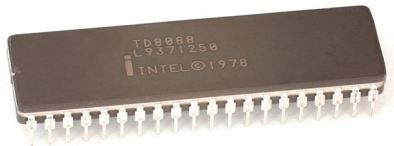
Rückblick “Geschichte der Rechner und Betriebssysteme”

- Effizienzsteigerung bei der Hardware
  - Moorsches Gesetz

    Die Anzahl der Transistoren integrierter Schaltungen verdoppelt sich etwa alle 18 Monate

    Folge: Kleinere Chips, komplexere Schaltungen, größere Caches

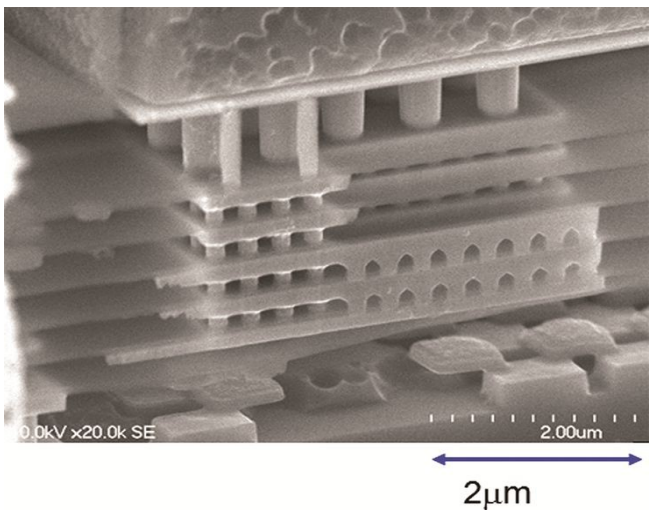
  → CPU ist viel schneller als Speicher und Peripherie



Intel 8088 (Appaloosa@Wikipedia)

- Bessere Auslastung der Hardware ist wünschenswert
  - Mehrere Programme gleichzeitig
  - Mehrere Benutzer:innen gleichzeitig

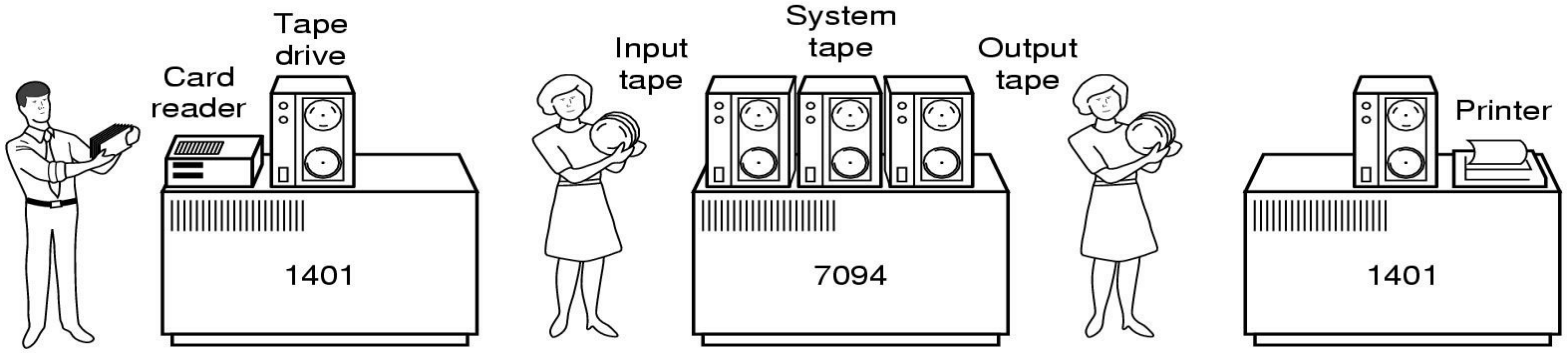
  → Dadurch entstehen jedoch viele Herausforderungen



Der I/O-Flaschenhals

- Die CPU ist viel schneller als Kartenleser und Drucker
  → Verschwendung von Rechenzeit

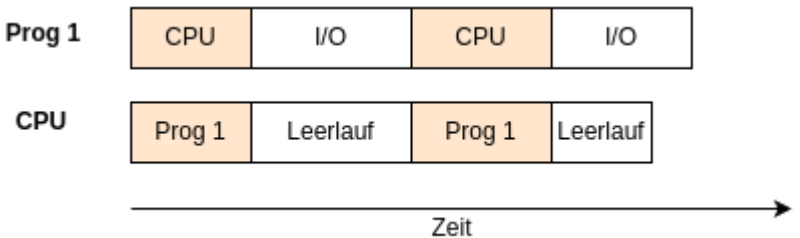
- Lösungen: Off-line processing: Stapelverarbeitung & Spooling
  - Lagerung von Daten in Puffern, bevor diese bearbeitet werden



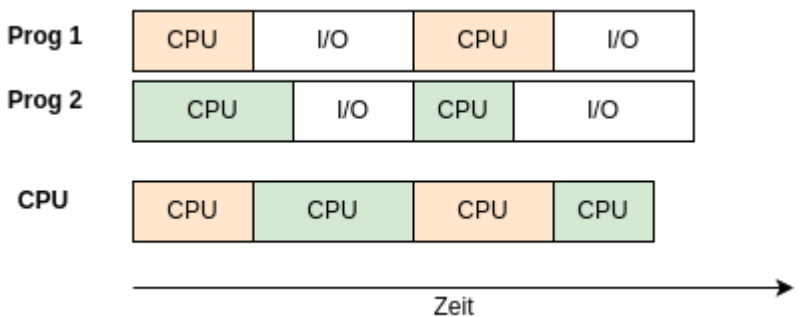
aus [Tanenbaum]: Moderne Betriebssysteme

Mehrprogrammbetrieb

- Trotz Spooling keine effiziente CPU-Nutzung im Einprogrammbetrieb
  - Es wechseln sich quasi CPU-Stöße und I/O-Stöße gegenseitig ab
  - Bei I/O-Stößen muss die CPU sinnlos warten



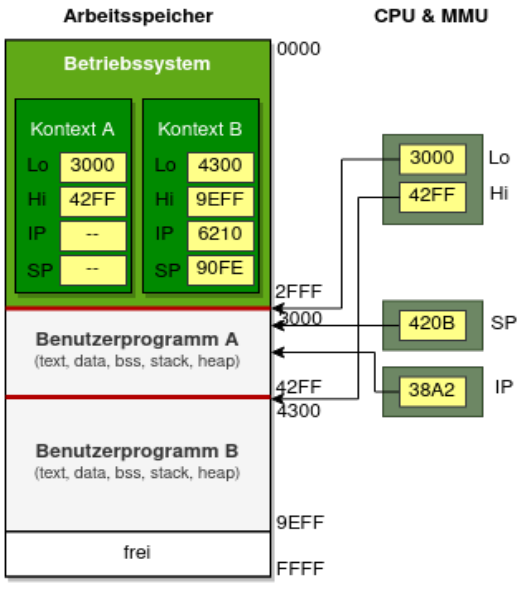
- Einführung von Mehrprogrammbetrieb
  - Nebenläufiges ausführen mehrerer Tasks, scheinbar “gleichzeitig”



Mehrprogrammbetrieb / Multitasking

- Herausforderungen
  - Aufteilen des Speichers in Partitionen
  - Gleichzeitiges bereithalten mehrerer Aufträge im Arbeitsspeicher
    → Umschalten, statt Warten

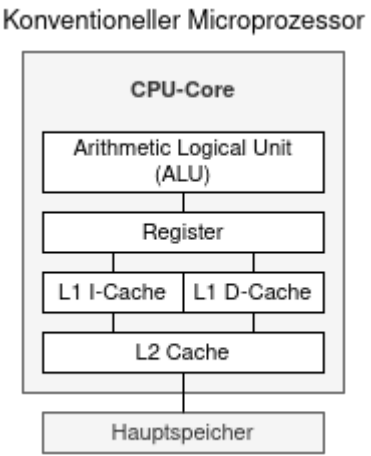
- Erforderlich
  - Dynamische Speicherverwaltung
  - Speicherschutz (→ MMU)
  - Prozessverwaltung (→ Kontextwechsel)



Computerhardware: CPU

Computerhardware: Prozessor (CPU)

- Jede CPU besitzt
  - einen spezifischen Befehlssatz
    (Reduced Instruction Set (RISC) vs. Complex Instruction Set (CISC))
  - diverse interne Register
    (Zwischenspeichern von Operanden und Ergebnissen, Spezialregister wie Befehlszähler (IP), Stackpointer (SP), usw.)
  - Heute: Sehr komplexe Befehlssätze und Register auf x86 (CISC bzw. Hybrid) Architektur (über 3500 Instruktionen (z.B. SSE4, AVX, FMA, CMPS))

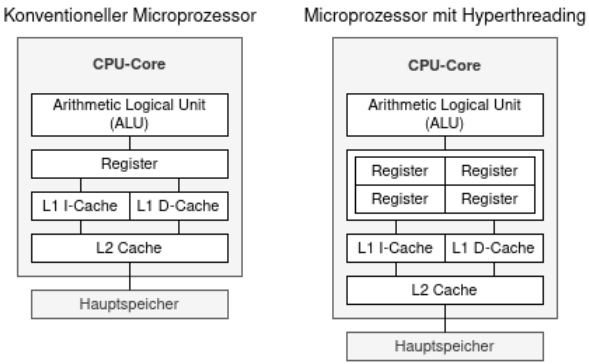


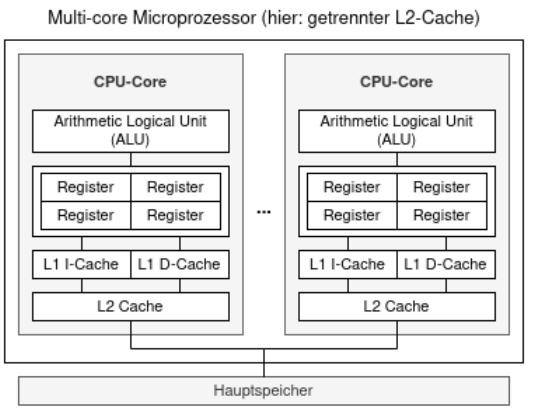
Prozessor: Hyperthreading, Multicore, Multiprozessor

- **Hyperthreading** (auf x86-Plattform eingeführt mit Intel Pentium 4)
  - **Mehrere Registersätze** erlauben dem Prozessor zwei oder mehr “Threads” zu verwalten und sehr schnell zwischen diesen umschalten. Die Threads laufen abwechselnd.

- **Multicore**
  - **Mehrere echte Prozessorkerne auf einem Chip** erlauben die parallele Ausführung von Prozessen. Cache (>= L2) kann ggf. von mehreren Prozessorkernen gemeinsam genutzt werden.
    (Bsp: Intel Xeon Phi: 72 Cores, 288 Threads)

- **Multiprocessor**
  - **Mehrere physische Prozessoren** (getrennte Chips) werden parallel betrieben. Neue Herausforderungen, z.B. durch verteilten Speicher.
    (Stichwort NUMA, ccNUMA)





Prozessor: Multitasking, Multithreading

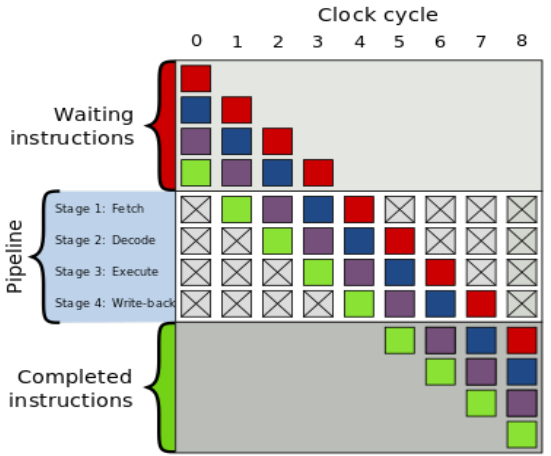
- Multitasking
  - Ist immer mit Mehrprogrammbetrieb gekoppelt, vermeidet aber die Nachteile des Mehrprogrammbetriebs
  - Das Betriebssystem kann Prozesse automatisch verdrängen, falls diese zu lange rechnen
  - Im Gegensatz zu Mehrprogrammbetrieb wird die Rechenzeit “gleichmäßig” auf alle Programme verteilt; Lösung zum Beispiel per “Zeitscheibenverfahren” (typischerweise im Bereich 10-200 ms).

- Multithreading
  - Threads sind zusammenhängende Folgen von Anweisungen innerhalb eines Programms (mit gemeinsamen Adressraum (Text, Data, Heap), aber eigenem Programmzähler (IP) und Stack). Diese können z.B. auf Multicore Systemen echt parallel ausgeführt werden.
  - Keine einheitliche Schnittstellendefinition: POSIX-Threads, UI-Threads, Win32-Threads, usw.

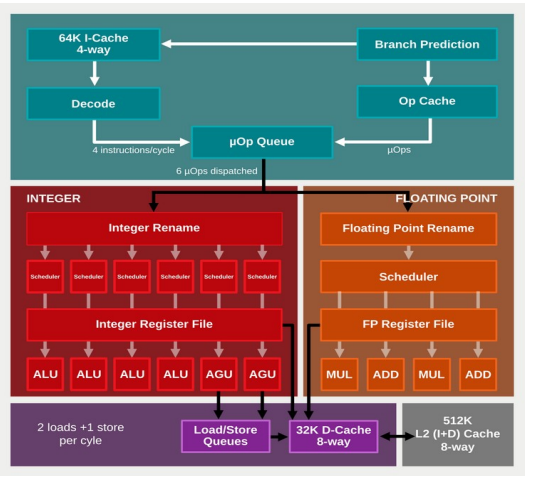
- Das Betriebssystem entscheidet, wann welcher Prozess läuft
  - Keine a-priori Annahme über zeitlichen Verlauf oder Ausführungsgeschwindigkeit möglich

Prozessor: Effizienzsteigerung

- Effizienzsteigerung
  - Pipeline-Konzept: “Parallelisierung” bzw. vorausschauendes laden, decodieren, ausführen und zurück schreiben → Mit jedem CPU-Takt eine Instruktion
  - Superskalare CPU: Simultanes laden und Nutzung mehrerer Verarbeitungseinheiten für das Decodieren, Ausführen (mehrere Fest- und Gleitkommaeinheiten), usw.
  - Heute: Out-of-Order execution, Speculative execution, Branch prediction, und viele weitere Optimierungen (→ Folge u.a. Sicherheitslücken wie Spectre und Meltdown)



4-stage pipeline (Bildquelle Wikipedia)



AMD Zen (Bildquelle Wikipedia)

Beispiel: CPUs, Befehlssätze, Multi-Core, NUMA, usw.

```
[root@azrael major]# lscpu
Architecture:                x86_64
  CPU op-mode(s):            32-bit, 64-bit
  Address sizes:             38 bits physical, 48 bits virtual
  Byte Order:                Little Endian
CPU(s):                      8
  On-line CPU(s) list:       0-7
Vendor ID:                   GenuineIntel
  Model name:                Intel(R) Core(TM) Ultra 7 268V
    CPU family:              6
    Model:                   189
    Thread(s) per core:      1
    Core(s) per socket:      8
    Socket(s):               1
    Stepping:                1
    CPU(s) scaling MHz:      18%
    CPU max MHz:             5000.0000
    CPU min MHz:             400.0000
    BogoMIPS:                6604.80
    Flags:                   fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr s
                             good nopl xtopology nonstop_tsc cpuid aperfmperf tsc_known_freq pni pclmulqdq dtes64 monitor ds_cpl
                             dline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb ssbd ibrs ibpb stib
                             i2 erms invpcid rdt_a rdseed adx smap clflushopt clwb intel_pt sha_ni xsaveopt xsavec xgetbv1 xsave
                             wp_act_window hwp_epp hwp_pkg_req hfi vnmi umip pku ospke waitpkg gfni vaes vpclmulqdq tme rdpid bu
                             apabilities
Virtualization features:
  Virtualization:            VT-x
Caches (sum of all):
  L1d:                       320 KiB (8 instances)
  L1i:                       512 KiB (8 instances)
  L2:                        14 MiB (5 instances)
  L3:                        12 MiB (1 instance)
NUMA:
  NUMA node(s):              1
  NUMA node0 CPU(s):         0-7
Vulnerabilities:
  Gather data sampling:      Not affected
  Ghostwrite:                Not affected
  Indirect target selection: Not affected
  Itlb multihit:             Not affected
  L1tf:                      Not affected
  Mds:                       Not affected
  Meltdown:                  Not affected
  Mmio stale data:           Not affected
  Old microcode:             Not affected
  Reg file data sampling:    Not affected
  Retbleed:                  Not affected
  Spec rstack overflow:      Not affected
  Spec store bypass:         Mitigation; Speculative Store Bypass disabled via prctl
  Spectre v1:                Mitigation; usercopy/swapgs barriers and __user pointer sanitization
  Spectre v2:                Mitigation; Enhanced / Automatic IBRS; IBPB conditional; PBRSB-eIBRS Not affected; BHI BHI_DIS_S
  Srbds:                     Not affected
  Tsa:                       Not affected
  Tsx async abort:           Not affected
  Vmscape:                   Mitigation; IBPB before exit to userspace
```

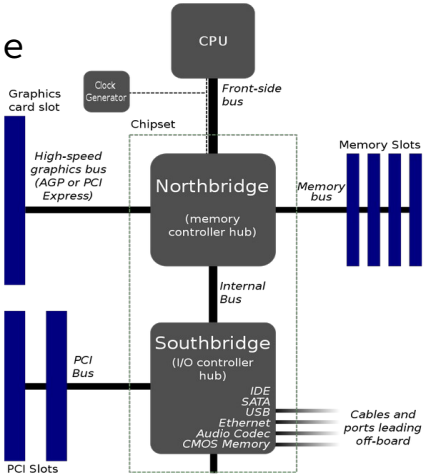
Mein Notebook

```
seprds01:~ # lscpu
Architecture:                x86_64
  CPU op-mode(s):            32-bit, 64-bit
  Address sizes:             46 bits physical, 48 bits virtual
  Byte Order:                Little Endian
CPU(s):                      40
  On-line CPU(s) list:       0-39
Vendor ID:                   GenuineIntel
  Model name:                Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz
    CPU family:              6
    Model:                   79
    Thread(s) per core:      2
    Core(s) per socket:      10
    Socket(s):               2
    Stepping:                1
    CPU(s) scaling MHz:      35%
    CPU max MHz:             3400.0000
    CPU min MHz:             1200.0000
    BogoMIPS:                4800.09
    Flags:                   fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse
                              nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 mon
                              xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb cat_
                              invpcid rtm cqm rdt_a rdseed adx smap intel_pt xsaveopt cqm_llc cqm_
Virtualization features:
  Virtualization:            VT-x
Caches (sum of all):
  L1d:                       640 KiB (20 instances)
  L1i:                       640 KiB (20 instances)
  L2:                        5 MiB (20 instances)
  L3:                        50 MiB (2 instances)
NUMA:
  NUMA node(s):              2
  NUMA node0 CPU(s):         0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38
  NUMA node1 CPU(s):         1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39
Vulnerabilities:
  Gather data sampling:      Not affected
  Indirect target selection: Not affected
  Itlb multihit:             KVM: Mitigation: VMX disabled
  L1tf:                      Mitigation; PTE Inversion; VMX conditional cache flushes, SMT vulnera
  Mds:                       Mitigation; Clear CPU buffers; SMT vulnerable
  Meltdown:                  Mitigation; PTI
  Mmio stale data:           Mitigation; Clear CPU buffers; SMT vulnerable
  Reg file data sampling:    Not affected
  Retbleed:                  Not affected
  Spec rstack overflow:      Not affected
  Spec store bypass:         Mitigation; Speculative Store Bypass disabled via prctl
  Spectre v1:                Mitigation; usercopy/swapgs barriers and __user pointer sanitization
  Spectre v2:                Mitigation; Retpolines; IBPB conditional; IBRS_FW; STIBP conditional;
  Srbds:                     Not affected
  Tsx async abort:           Mitigation; Clear CPU buffers; SMT vulnerable
```

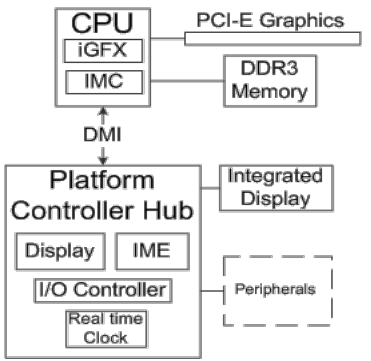
Einer der HFD Backup-Server

Computerhardware: Bussystem

- Chipsatz: Verbindendes Element zwischen CPU und dem Rest der Hardware
  - Von-Neumann-Architektur heute nicht mehr hundertprozentig umgesetzt
- Historisch: Chipsatz bestehend aus Northbridge und Southbridge
  - Frontside Bus (FSB) verbindet CPU mit Northbridge (Daten-, Adress- und Steuerbus)
  - Schnelle Geräte an Northbridge angebunden, langsame an Southbridge (ab ca. 2008 wandert Speichercontroller in die CPU)
- Heute: Plattform Controller Hub (PCH)
  - Anbindung schneller Geräte (PCI-e, RAM) direkt an die CPU
  - Anbindung der anderen Geräte an den PCH
- Viele weitere Bussysteme in Rechnern
  - Parallel: PCI, SCSI, PATA (IDE), PCMCIA, ...
  - Seriell: PCI-Express, SATA, Ethernet, USB, ...



Chipsatz (North- und Southbridge)
CC-BY-SA (Gribeco @Wikipedia)



Platform Controller Hub
CC-BY-SA (Anas hashmi @Wikipedia)

Computerhardware: Speicher

Speicher: Speicherpyramide

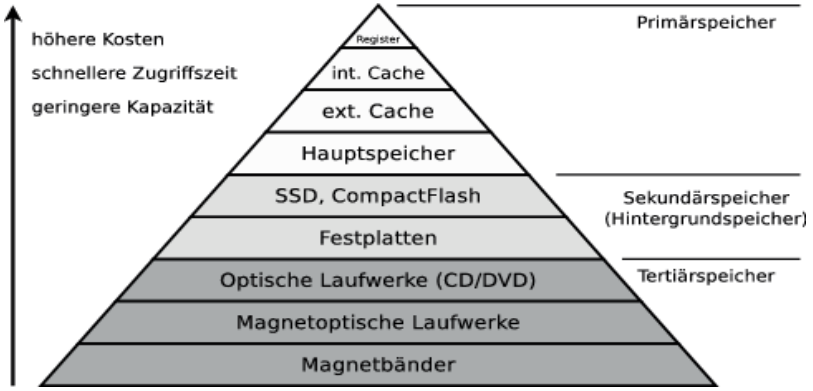
- Anforderung: Schnell, Groß, Billig
  - You can’t winn ‘em all ...
  - Lösung: Mehrstufige Speicher-Hierarchie

- Speicherpyramide (von oben nach unten)

  - Primärspeicher
    - Direkter Zugriff durch die CPU
    - CPU-Register, Cache, Hauptspeicher

  - Sekundärspeicher
    - Zugriff über Speichercontroller
    - Festplatten (HDD, SSD)

  - Teritärspeicher
    - Nicht dauerhaft mit dem Rechner verbunden
    - Mangetbänder, Magnetoptisch, Optisch (CD/DVD)



Copyright SpringerLink Betriebssysteme Kompakt

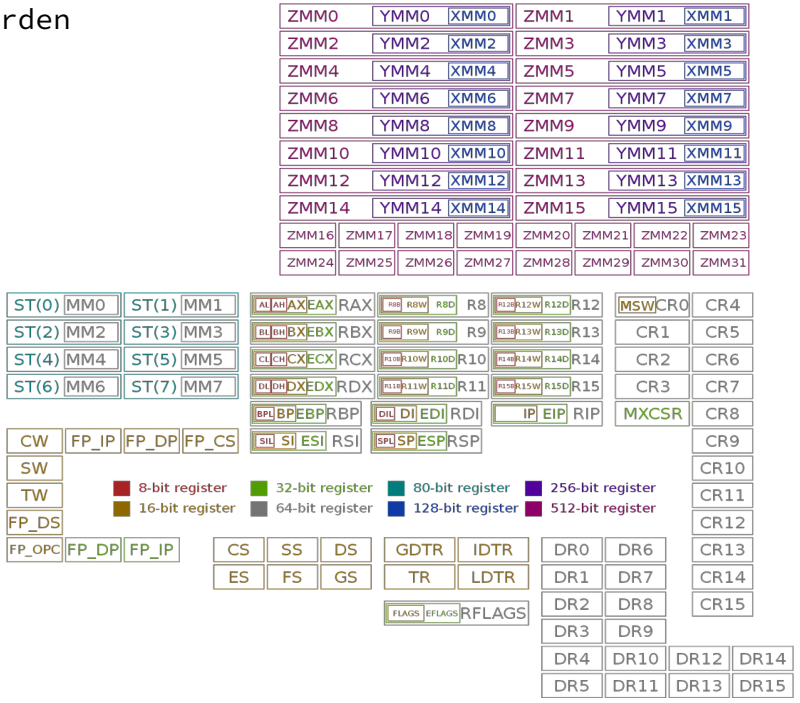
Iomega Jaz
Magnet-Wechselspeicher (0,5 – 2 GB)
CC-BY-SA (KMJ @Wikipedia)

LTO Ultrium Bandcartridge
Komprimiert: 45 TB, 1GB/s
CC-BY-SA (Austinmurphy @Wikipedia)

Speicher: CPU-Register

- CPUs enthalten diverse Register
  - Register enthalten Daten, die von der CPU unmittelbar benötigt werden
  - Arbeiten mit der Taktrate der CPU

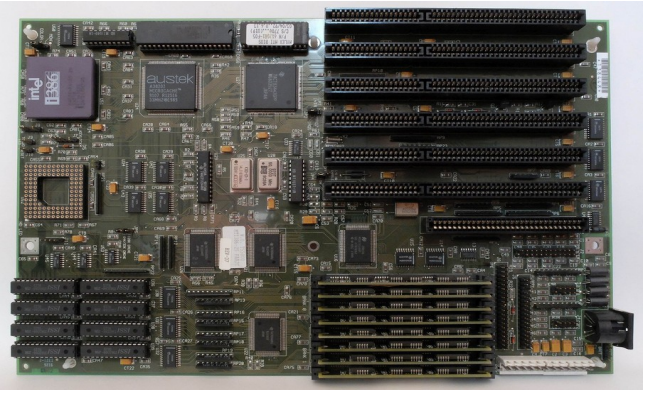
- Beispiele
  - Befehlsregister (instruction register, IR)
    Enthält den aktuellen Befehl
  - Befehlszähler (program counter, PC)
    Enthält die Speicheradresse des nächsten Befehls
  - Stackregister (stack pointer, SP)
    Enthält die Speicheradresse des “oberen” Ende des Stack
  - Adressregister
    Zur Ermittlung von Speicheradressen von Befehlen und Operanden (Basisadressregister und Indexregister)
  - Datenregister
    Speicherung von Operanden und Resultaten der ALU
    Beispiel: EAX, EBX, ECX (32-Bit) bzw. RAX, RBX, RCX (64-Bit)



Registers available in the x86-64 instruction set
(CC-BY-SA Immae @Wikipedia)

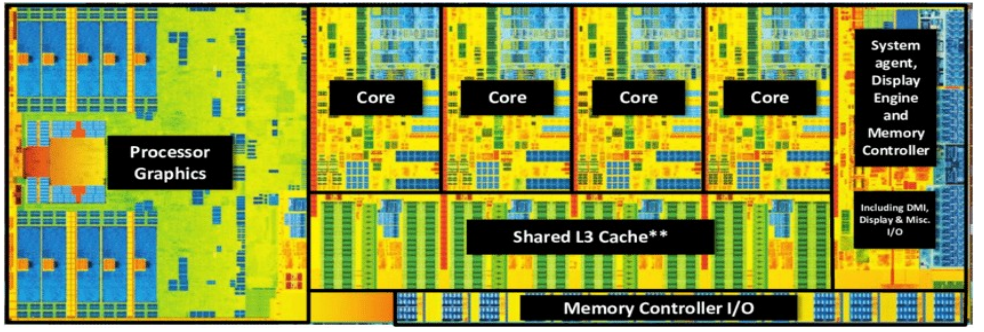
Primärspeicher: Cache

- Kleiner aber sehr schneller Speicher direkt in der CPU
  - Enthält Kopien von Teilen des Hauptspeichers
- In mehreren Hierarchien angeordnet
  - First-, Second-, Third Level Cache
    Mindestens der First Level Cache, der selbst in einen Befehlscache und einen Datencache unterteilt ist, steht der CPU (bzw. einem CPU-Kern) exklusiv zur Verfügung. In den weiter unten liegenden Hierarchien wird Cache von den CPU-Kernen ggf. gemeinsam genutzt.



Mainboard mit i386 CPU und externem L2-Cache
(CC-BY-SA Drahtlos @Wikipedia)

- Funktionsweise
  - Organisation in Cache-Lines
    Kleinste Einheit innerhalb des Cache. Eine Cache-Line (heute i.d.R. 64-128 Byte) enthält eine Kopie eines bestimmten Bereichs des Hauptspeichers.
  - Cache-Hit
    Aktuelle Daten werden im Cache gefunden
    (Zugriff auf die Daten im Bereich ~2 Taktzyklen)
  - Cache Kohärenz
    Stellt sicher, dass die Daten in unterschiedl. Caches identisch sind (z.B. L1 Cache je CPU-Core)

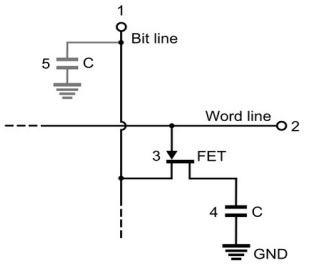


Die-Shot: Quad-Core Haswell CPU (Quelle)

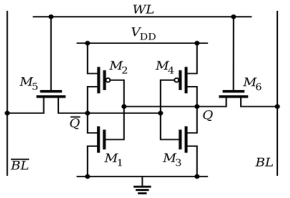
Primärspeicher: Hauptspeicher

- In der Regel Flüchtige Speicher
  - Dynamic Random Access Memory (DRAM)
    - Speicherung mittels Kondensator
    - Regelmäßiges “Aufgefrischen” der Kondensatoren nötig
    - Verwendung zum Beispiel als Hauptspeicher (RAM)
  - Static Random Access Memory (SRAM)
    - Speicherung mittels bistabiler Kippstufe (Flipflop)
    - Daten bleiben erhalten solange Betriebsspannung anliegt
    - Verwendung zum Beispiel für CPU-Register und Cache

DRAM Module
(CC-BY-SA, An-d @Wikipedia)

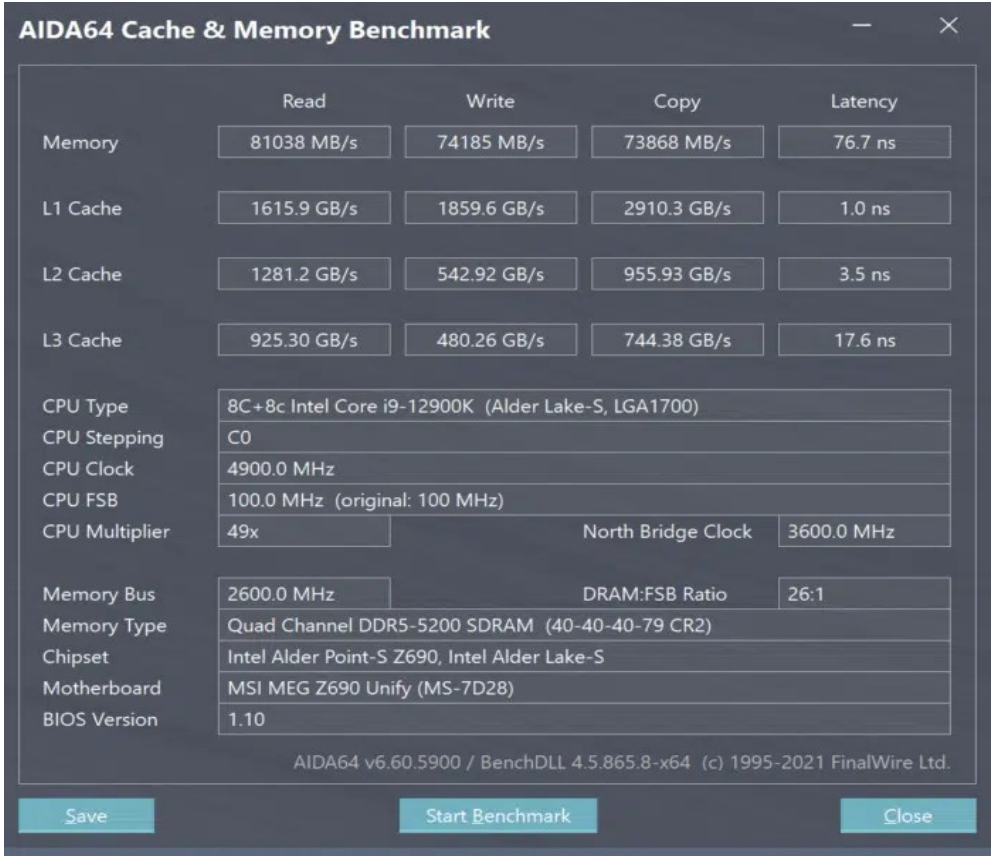


Prinzipieller Aufbau einer DRAM-Zelle
(CC-BY-SA, Tosaka @Wikipedia)



Prinzipieller Aufbau einer SRAM-Zelle
(CC-BY-SA, Inductiveload @Wikipedia)

Beispiel Speicherhierarchie



Bildquelle: https://www.thefpsreview.com

Sekundärspeicher

- Hard Disk Drive (HDD)
  - Rotierende metallische Scheiben (“spinning rust”)
    (z.B. 5.400, 7.200, 10.800, 15.000 Rotationen/Minute)
  - Schwungarm mit Schreib-/Lesekopf für jede Scheibenseite
  - Luftpolster zw. Platte und Schwungarm von ca. 20 Nanometern
  - 100x größer und billiger, aber auch 1000x langsamer als Hauptspeicher

Festplatte (CC BY-SA, Eric Gaba @Wikipedia)

- Solid State Drive (SSD)
  - Nicht-Flüchtige digitale Speicherzellen (Flash-Speicher)
  - Vorteile: Keine beweglichen Teile, geringer Energieverbrauch,
    geringes Gewicht, keine Geräuschentwicklung, …
  - Nachteile: Hoher Preis, Verschleiß (begrenzte Anzahl Schreibzyklen),
    sicheres Löschen ist ggf. schwierig

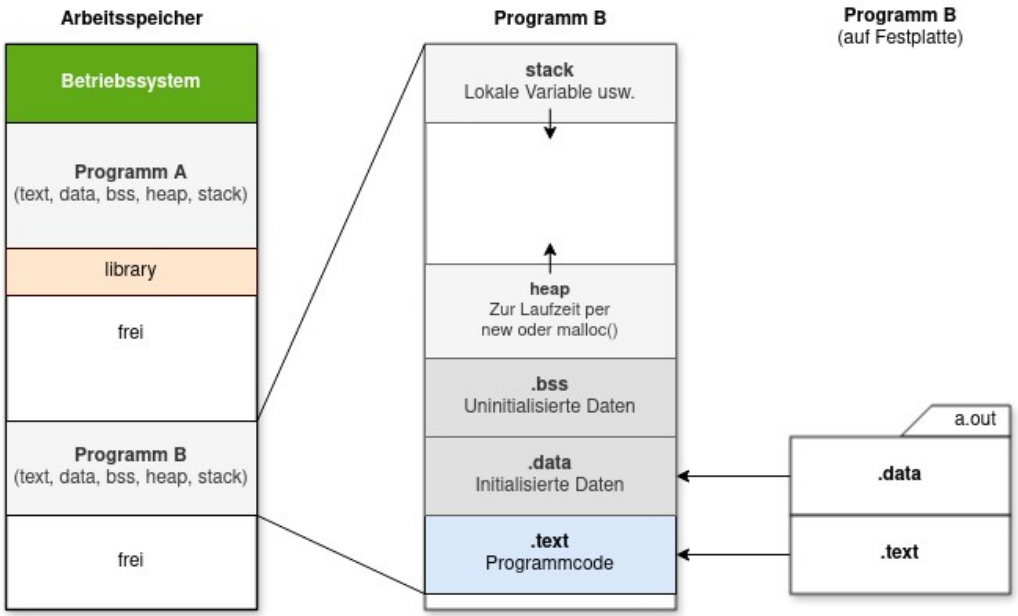
SSD (CC-BY-SA, Jacek Halicki @Wikipedia)

- NVMe
  - Schnittstelle zur Anbindung von SSD mittels PCI-Express

NVMe SSD (CC-BY-SA, D-Kuru @Wikipedia)

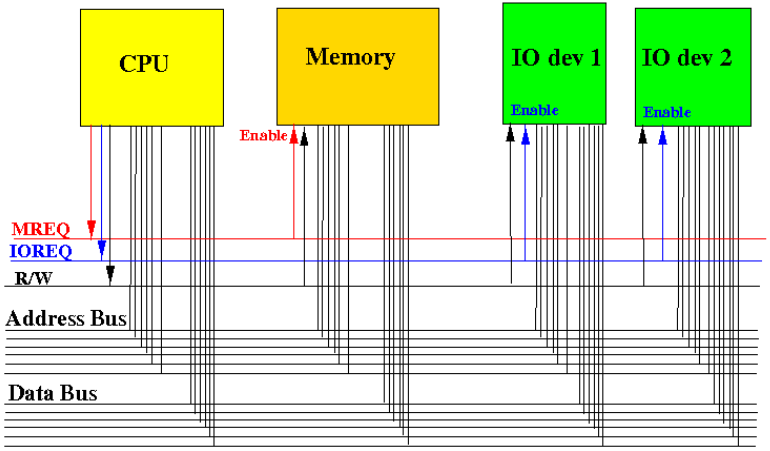
Einschub: Benutzerprogramm im Speicher

- Verschiedene Speichersegmente haben verschiedene Anforderungen
  - Ausführbar vs. Beschreibbar
  - Statisch vs. dynamisch wachsend
  - Großer Abstand
    - zw. Stack und Heap
    - zw. versch. Programmen
- Anforderungen aus Sicht der Prozesse
  - Gesamter Speicher exklusiv nutzbar
  - Isolation von anderen Adressräumen
  - Keine “Verunreinigungen”

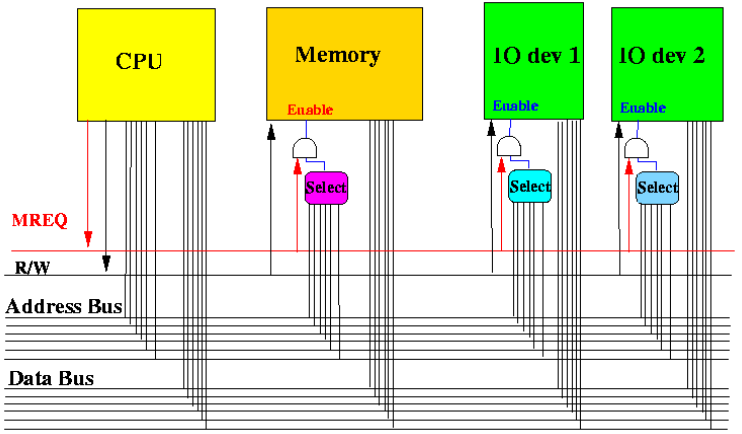


Einschub: Adressraum

- “Landkarte” des Adressraums
  - Realen Arbeitsspeicher
    - Betriebssystem
      (Betriebssystemcode und dessen Datenstrukturen)
    - Benutzerprogramme
      (Text-, Daten-, Heap-, Stack-Segmente)
  - I/O-Geräte (mit direkt adressierbarem Speicher)
  - Viele große Lücken
- I/O-Adressierung
  - Standard I/O
    Umschaltung zwischen Adressräumen
  - Memory-mapped I/O
    Aufteilen des Adressraums



Standard-I/O (Bildquelle)



Memory-mapped-I/O (Bildquelle)

Speicherhierarchie

- Funktionsweise der Speicherhierarchie
  - Lesender Zugriff auf ein Element im Speicher (z.B. Festplatte)
    - Kopie des gelesenen Elements wandert entlang der Speicherhierarchie “nach oben”
  - Schreibender Zugriff erfordert das zurückschreiben der Daten “nach unten”
    - Kein direktes Zurückschreiben nach ganz unten (zum Original)
    - Aktualisierung jeder Hierarchieebene notwendig
- Erforderliche Konzepte
  - Strategie zum Zurückschreiben in den langsameren Speicher
    - Write-through
      Daten werden bei Änderungen sofort an die untere Ebene weitergegeben
    - Write-back
      Daten werde erst dann an die untere Ebene weitergegeben, wenn diese in der oberen Ebene ersetzt (verdrängt) werden.
  - Strategie zum Ersetzen von Daten im schnelleren Speicher
    - Speicherersetzungsalgorithmen (z.B. Least Recently Used (LRU))

Speicherhierarchie

- Programme haben i.d.R. die Eigenschaft der Lokalität
  (vgl. The Working Set Model for Program Behavior, P. Denning, MIT (Link))

  - Zeitliche Lokalität

    Es besteht eine relativ hohe Wahrscheinlichkeit, dass die zuletzt verwendete Speicheradresse in naher Zukunft erneut verwendet wird

  - Räumliche Lokalität

    Es besteht eine relativ hohe Wahrscheinlichkeit, dass nach einem Speicherzugriff auf eine benachbarte Speicheradresse zugegriffen wird

  - Beispiel:

```
[…]

int len = 100;
int sum = 0;

for (int i = 0; i < len; i++) {
      sum += array[i];
}

[…]
```

Eingabe / Ausgabe (I/O)

I/O: Input / Output

- Ein-/Ausgabegeräte (E/A, bzw. I/O)
  - Wesentlicher Bestandteil der Von-Neumann-Architektur
  - All das, was nicht CPU oder Primärspeicher ist
  - Werden nach deren kleinster Übertragungseinheit unterschieden

- Arten von I/O-Geräten
  - Zeichenorientierte Geräte
    - Bei jeder Ankunft eines Zeichen wird mit der CPU kommuniziert
    - Beispiele: Maus, Tastatur, Drucker, Magnetband
  - Blockorientierte Geräte
    - Datenübertragung erst dann, wenn ein kompletter Block (z.B. 4 kB) vorliegt
    - Beispiele: Festplatte / Solid State Drive (SSD)

I/O-Geräte und Root-Dateisystem

- UNIX und UNIXoide Systeme folgen dem Filesystem Hierarchy Standard (FHS)
  - Oberste Hierarchie-Ebene ist das Root-Dateisystem

- Aufbau
  - Der / dient als Verzeichnis-Trennzeichen
  - Die Wurzel (/) befindet sich selbst in einer Partition (z.B. auf einer Festplatte)
  - Unterhalb von / befinden sich weitere
    - „echte“ Partitionen (z.B. einer Festplatte)
    - „virtuelle Dateisysteme“ (z.B. /dev, /sys, /proc)

- Wichtiges Konzept in UNIX: “Special files” (in /dev)
  - Zwei Arten: Block-Dateien (block special file) und Zeichen-Dateien (character special file)
  - Darstellung von I/O-Geräten als Dateien (Beispiel: /dev/console, /dev/tty, /dev/random)
  - Grundidee: “Everything is a file” (im Betriebssystem Plan-9 konsequent weitergedacht)

```
brw-rw----   1 root  disk      259,   0 Apr 18 17:38 nvme0n1
crw-rw-rw-   1 root  root        1,   8 Apr 18 17:38 random
crw--w----   1 root  tty         4,   0 Apr 18 17:38 tty0
```

Beispiel Special Files in /dev

Das Betriebssystemen

Was ist ein Betriebssystem?

„Ein Computer ist, wenn er genau betrachtet wird, nur eine Ansammlung von Plastik und Metall, das zur Leitung von Strom benötigt wird. Dieser „Industriemüll“ kann somit nicht ausschließlich das sein, was wir unter einem modernen Computer verstehen, etwas, das dem Computer „Leben“ einhaucht und ihn zu dem Werkzeug unseres Jahrhunderts macht.

Es ist das Betriebssystem, das die Kontrolle über das Plastik und Metall (Hardware) übernimmt und anderen Softwareprogrammen (Excel, Word, . . .) eine standardisierte Arbeitsplattform (Windows, Unix, OS/2) schafft.“

Ewert et al., Literatur zu „Freehand 10“

Definition: Betriebssystem

„Die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften der Rechenanlage die Grundlage der möglichen Betriebsarten des digitalen Rechensystems bilden und insbesondere die Abwicklung von Programmen steuern und überwachen.“

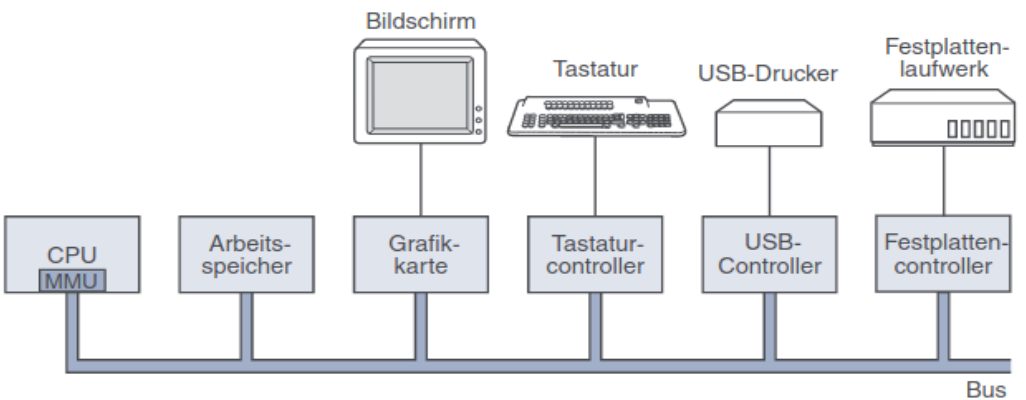
DIN 44300

„Eine Softwareschicht, die alle Teile des Systems verwaltet und dem Benutzer eine Schnittstelle oder virtuelle Maschine anbietet, die leichter zu verstehen und zu programmieren ist [als die darunterliegende reale Maschine].“

Tanenbaum

Das Betriebssystem als "Betriebsmittelverwalter"

- Des Betriebssystems in der Sichtweise **Betriebsmittelverwalter**
  - Durchsetzen einer geordneten und kontrollierten Zuteilung der Betriebsmittel an die in Ausführung befindlichen Programme
  - Ermöglichen der gemeinsamen Benutzung *teurer/knapper* Betriebsmittel
    - z.B. Hardware-Betriebsmittel: Soundkarte, Drucker, ...
    - z.B. logische Betriebsmittel: Dateien, Puffer, ...
  - Schutz der Betriebsmittel vor unberechtigter Benutzung
  - Vermitteln im Falle von Konflikten
  - Abrechnung der Kosten der Betriebsmittelnutzung (Accounting)
  - Buchführung über die Abläufe (Auditing)



Das Betriebssystem als "virtuelle Maschine"

- Des Betriebssystems in der Sichtweise **virtuelle Maschine**
  - Abschirmen des Programmierers vor der Komplexität der Hardware
    - durch Nutzung einer Software-Schicht über der "nackten" Hardware → dem Betriebssystem
  - Schnittstelle zum Programmierer
    - einfache Abstraktionen auf hohem Niveau mit entsprechenden Operationen (z.B. Dateikonzept, Lesen und Schreiben von Blöcken)
    - Diese Schnittstelle ist einfacher zu verstehen und langlebiger

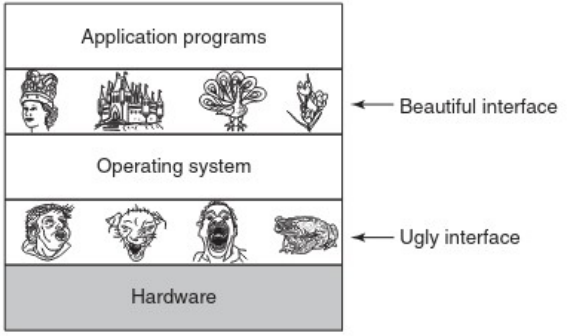


Figure 1-2. Operating systems turn ugly hardware into beautiful abstractions.
[Tanenbaum]

Beispiel: SCSI-Operationen für Festplatten:

FORMAT UNIT	RECOVER DATA	VERIFY
INQUIRY	RECOVER ID	WRITE
MODE SELECT	RELEASE UNIT	WRITE AND VERIFY
MODE SENSE	REQUEST SENSE	WRITE BUFFER
NO OPERATION	RESERVE UNIT	WRITE EXTENDED
PRIORITY RESERVE	REZERO UNIT	WRITE LONG
READ	SEEK	WRITE SAME
READ BUFFER	SEEK EXTENDED	.....
READ CAPACITY	SEND DIAGNOSTIC	
READ DEFECT DATA	SET LIMITS	
READ EXTENDED	START/STOP UNIT	
READ LONG	TEST UNIT READY	
REASSIGN BLOCKS		

Beispiel für “Komplexität der Hardware”:
SCSI-Befehle zur Kommunikation mit einer Festplatte

Betriebsarten

- Die Betriebsart bestimmt, wie das Betriebssystem Aufträge abarbeitet

- Stapelverarbeitung (*batch processing*)
  - Aufgaben werden nacheinander und ohne Benutzerinteraktion abgearbeitet
  - Früher: Ausführung einzelner Jobs in Form von Lochkartenstapeln
  - Heute: Nach wie vor wichtig (Hintergrund-Jobs, High-Performance-Computing, usw.)

- Dialogbetrieb
  - Ständige Interaktion mit Benutzer:in
  - Neuartige I/O-Geräte (Tastatur, Bildschirm, Maus) erfordern entsprechende Reaktionszeiten

- Echtzeitbetrieb
  - Garantierte Antwortzeiten für die Bearbeitung einer bestimmten Aufgabe

Zusammenfassung

- Grundlegende Funktionsweise von Computern
  - Von-Neumann-Architektur / Von-Neumann-Zyklus

- Computerhardware
  - CPU (und Chipsatz)
  - Speicher (Speicherhierarchie)
  - Ein-/Ausgabe

- Das Betriebssystem
  - Betriebsmittelverwalter bzw. (erweiterte) virtuelle Maschine
  - Abstraktion der Komplexität des darunterliegenden Rechnersystems

- Übung 2: Vertiefung Kommandozeile und Shellscripts