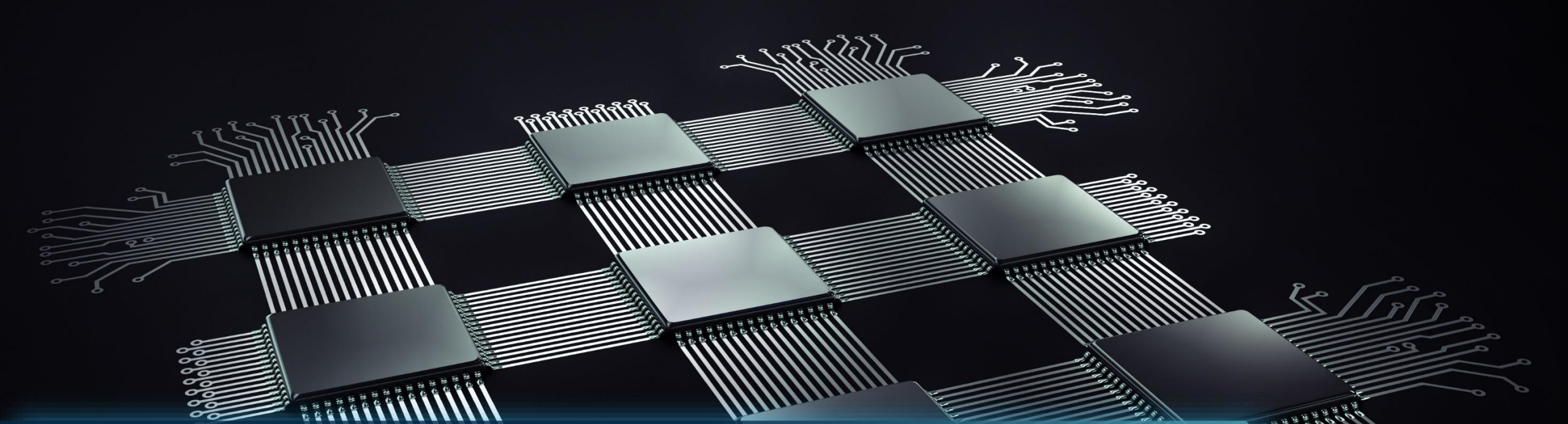


Operation System Security

01 - Einführung

SECURNITE

Heutige Agenda

Vorstellung des Dozenten

Modulaufbau

Check-In

Was ist ein Computer?

CPU

Speicher

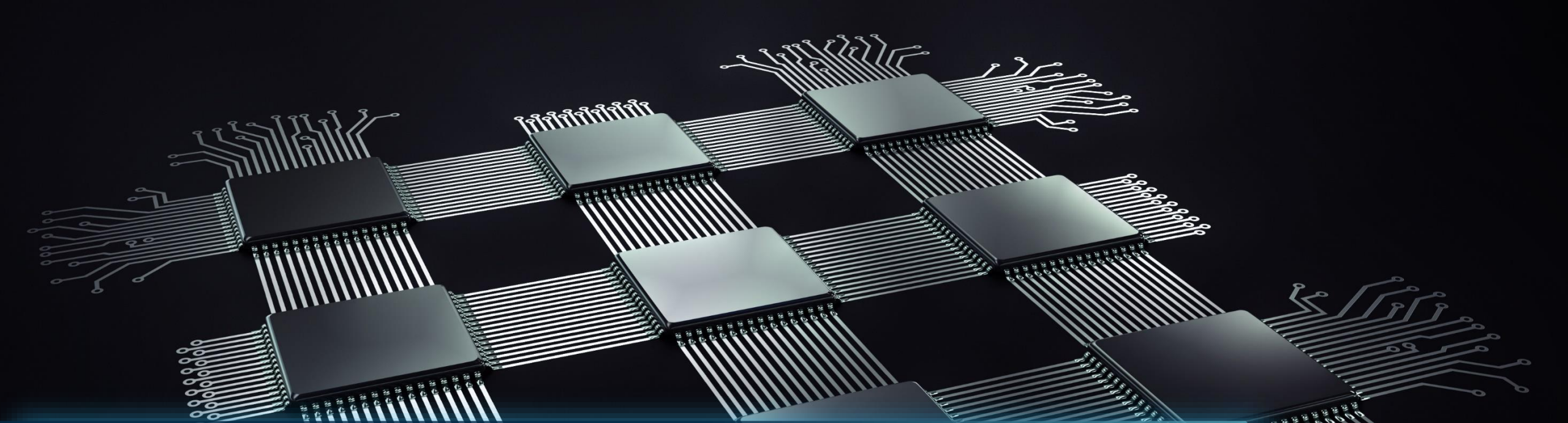
Busse

Hardware & Software

Bestandteile von Betriebssystemen

Einführung in die Labore

SECURNITE



Ihr Dozent

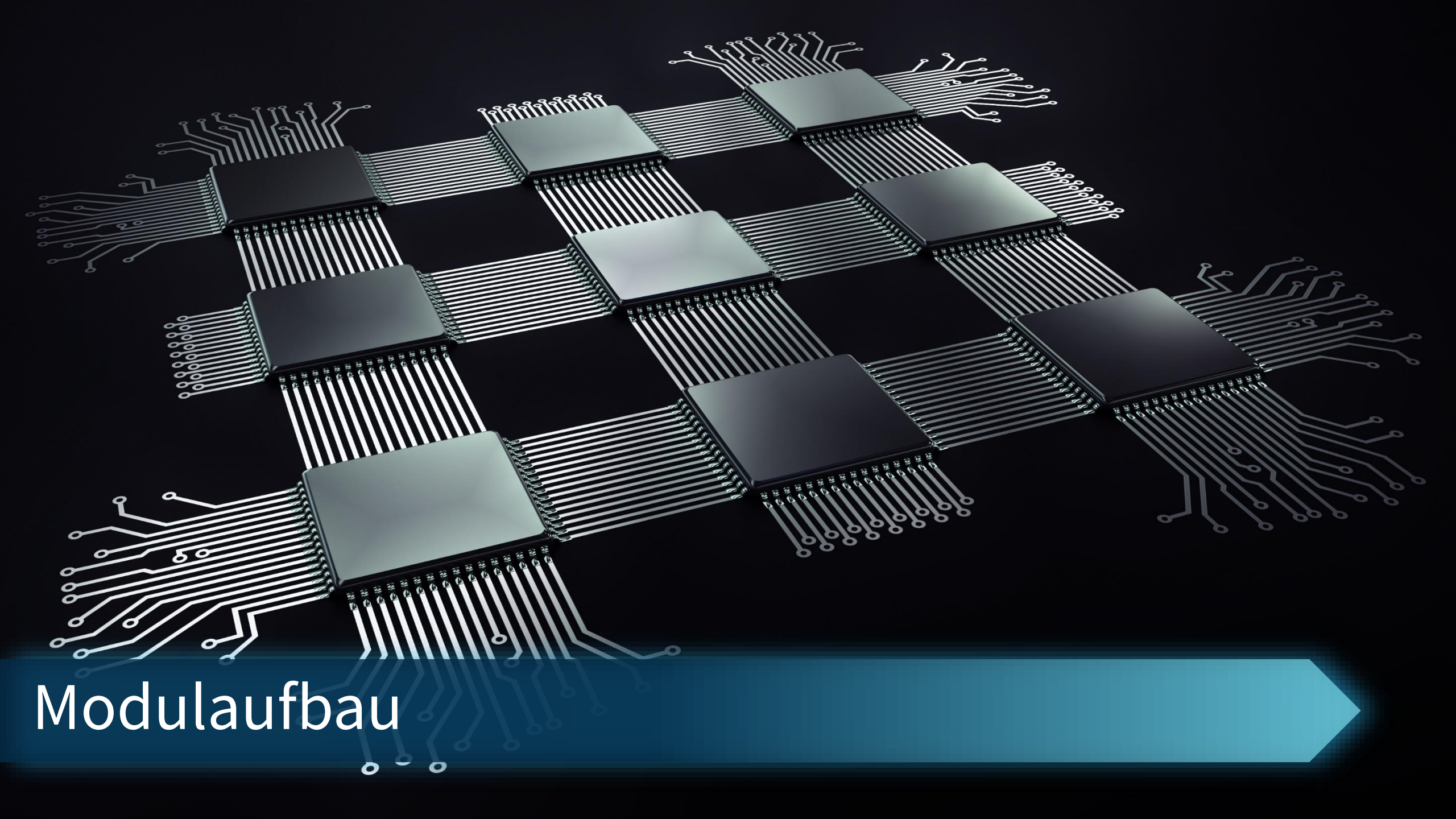
SECURNITE

Dr. Thomas Punz

- Experte für ganzheitliche Informationssicherheit von der detailliertesten Technik bis zur wegweisenden Strategie
- Doktor der Wissenschaften in experimenteller Elementarteilchenphysik (ETH)
- Geschäftsführer der SECURNITE Gruppe

Kontakt

thomas.punz@hslu.ch



Modulaufbau

Lernziele

Fachkompetenzen

- Grundsätze für gute Bestriessystemsicherheit kennen und entsprechende Situationen einschätzen zu können
- Gefahren in Bezug auf Betriebssysteme vorbeugen, erkennen und Gegenmaßnahmen ergreifen.

Methodenkompetenzen

- Durchführung von Laborübungen mit bereitgestelltem und eigenem Equipment
- Vorgehen beim Erkennen von Risiken in Bezug auf OS-Sicherheit

Personalkompetenz

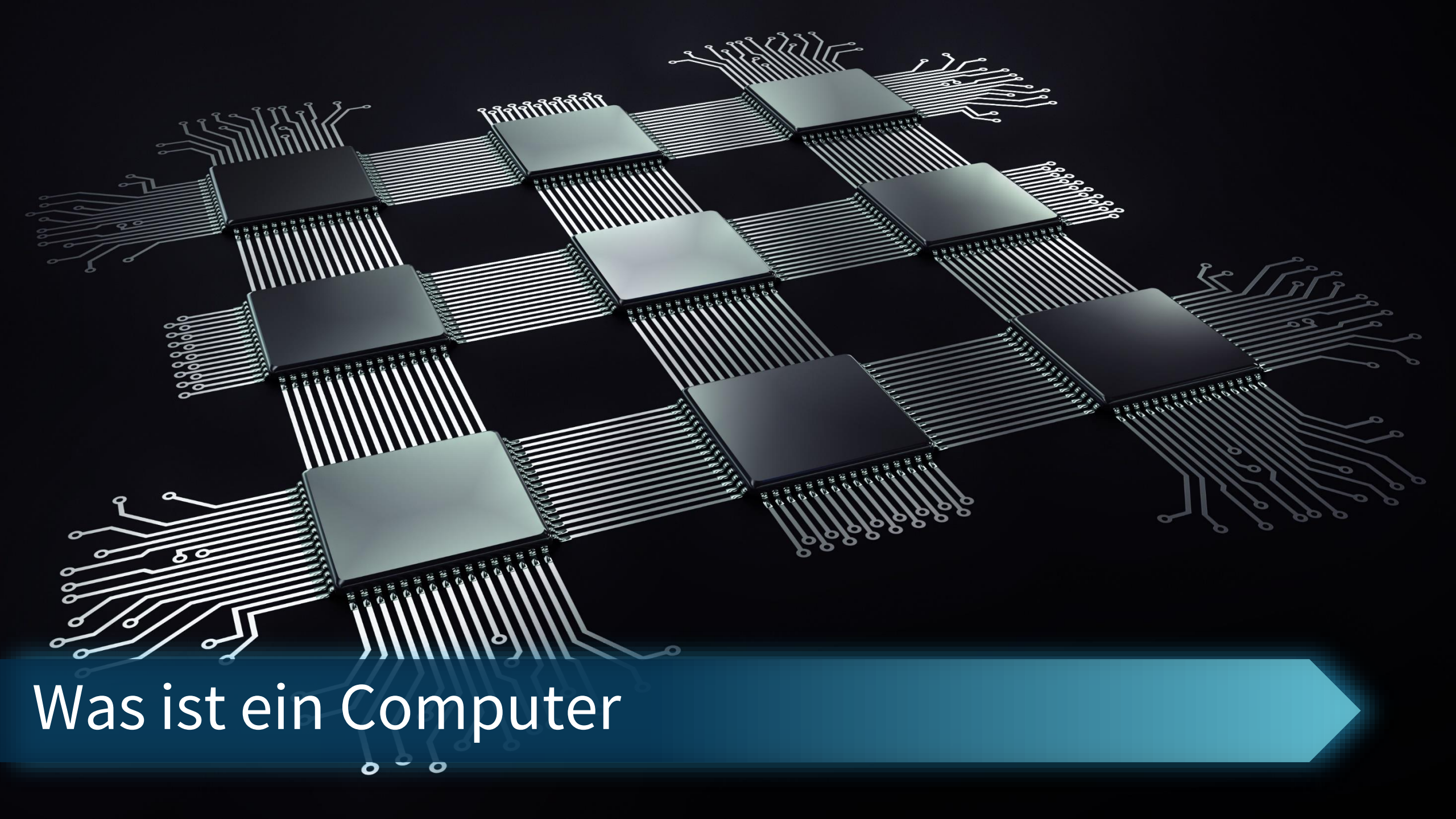
- Logisches Vorgehen
- Anwenden des bereits Erlernten

SECURNITE

6

Organisatorisches

- Die Modulnote besteht aus einer schriftlichen Prüfung (100%)
- Das Vorlesungsmaterial wird im Vorfeld jeder Semesterwoche im Ilias bereitgestellt
- Die Veranstaltungen werden aufgezeichnet und im Nachgang im Ilias zur Verfügung gestellt



Was ist ein Computer

Von-Neumann-Architektur

Was ist die Von-Neumann-Architektur?

- Die Von-Neumann-Architektur (VNA) ist ein **Referenzmodell für Computer**, bei dem ein gemeinsamer Speicher sowohl Computerprogrammbefehle als auch Daten hält.
- Benannt nach dem Mathematiker **John von Neumann**, der 1945 wesentliche Arbeiten zu diesem Thema veröffentlichte.

Entwicklungsgeschichte

- Von Neumann beschrieb das Konzept 1945 in seinem Papier “First Draft of a Report on the EDVAC”.
- Revolutionär, da zuvor entwickelte Rechner an ein festes Programm gebunden waren.
- Mit der VNA war es nun möglich, Änderungen an Programmen schnell und ohne Hardwareänderungen durchzuführen.

Von-Neumann-Architektur

Grundprinzipien der VNA

1. **Gemeinsamer Speicher**: Ein Speicher für Befehle und Daten.
2. **Rechenwerk und Steuerwerk**: Trennung von Berechnung und Steuerung.
3. **Programmierbarkeit**: Flexibilität durch Softwaresteuerung.

Heutige Anwendung

- Die meisten modernen Computer basieren auf dem Grundprinzip der VNA.
- Allerdings sind sie oft komplexer und leistungsfähiger als die ursprünglichen VNA-Modelle.

Fazit

Die Von-Neumann-Architektur hat die Computerentwicklung revolutioniert und bildet die Grundlage für die meisten heute bekannten Computer.

Definition: Computer

- (englisch compute, latein computare, deutsch (zusammen-)rechnen)
- oder Rechner ist ein Apparat, der mit Hilfe einer programmierbaren Rechenvorschrift **Informationen** verarbeiten kann (Datenverarbeitungsanlage)

SECURNITE

Definition: Computer

- **Eingabe**, **Verarbeitung** und **Ausgabe** von Daten frei programmierbar

Digitalcomputer

- mit digitalen Geräteeinheiten digitale Daten verarbeiten

Analogcomputer

- mit analogen Geräteeinheiten analoge Daten verarbeiten

Kategorien

Minirechner (mittlere Datentechnik)

- Bürocomputer
- Einsatzbereiche: **Anwendungs-programme**, **Datenbanken** (in kleinerem Umfang)

Workstation

- **Arbeitsplatzcomputer** für Rechenleistung und Grafikleistung
- Einsatzgebiete: Bildbearbeitung, Computer Aided Design (CAD) und Datenbanken

Supercomputer (number cruncher)

- **Hochleistungsrechner**, speziell auf komplizierte technisch-wissenschaftliche Aufgaben ausgelegt
- zentrale **rechenintensive Aufgaben**

Grossrechner (Mainframes, Hosts)

- in **Rechenzentren** eingesetzt
- grosse Anzahl von Terminals und Peripheriegeräten
- Einsatzgebiete: Zentralcomputer großer Organisationen

Kategorien

Tragbare Computer

- Laptop, Notebook, Tablets
- Smartphone

Pervasive & Ubiquitous Computing

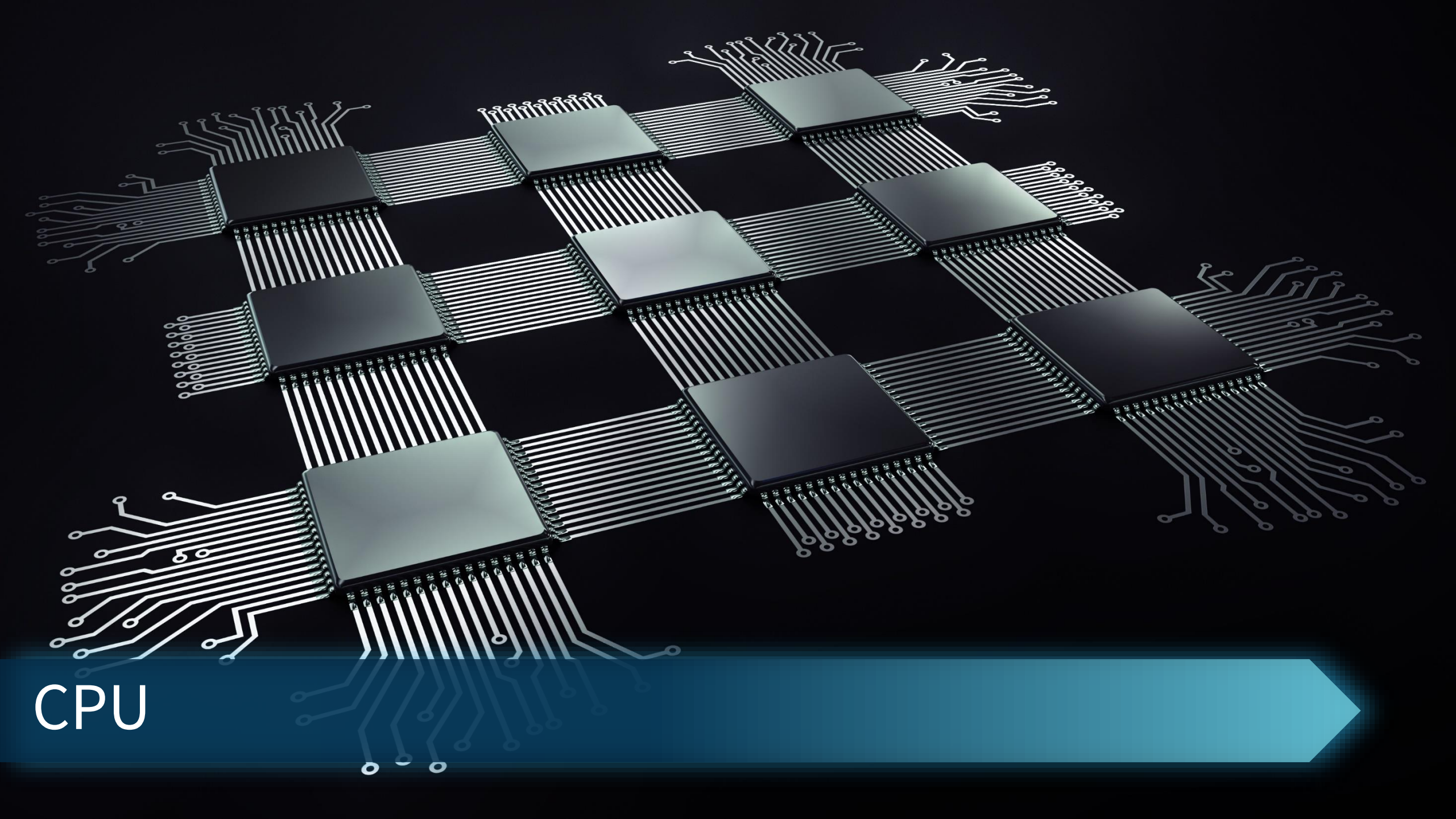
- Verborgene Computer z.B. innerhalb von IoT Geräten
- Sprachsteuerung
- Von Überall zugreifbar

Mikrocomputer

- **Einzelplatzsystem**
- Zentraleinheit auf einem Chip integriert (Mikroprozessor)

Personal Computer (PC)

- **Synonym für Mikrorechner**
- ursprüngliche Idee:
  - leistungsfähiger Computer für die meisten der am Arbeitsplatz benötigten Funktionen
  - sollte möglichst unabhängig von anderen Systemen arbeiten (stand Alone)



CPU

Central Processing Unit

- Zentrale **Verarbeitungseinheit**
- Teil des Computers, der die Hauptarbeit erledigt und alle anderen Komponenten **steuert**
- liest **Programmcode** (Anweisungen, Befehle) aus **Hauptspeicher**, **entschlüsselt** Programmcode
- **führt** Programmcode **aus** und **modifiziert** dabei evtl. Daten und Programmcode im Hauptspeicher
- kann Eingaben **lesen** und Resultate **ausgeben**
- Geschwindigkeit wird in **Herz** gemessen

Central Processing Unit

Wesentliche Bestandteile

- **Control Unit**: liest Programmbefehle aus und führt sie aus
- **Internes Register / Speicher**: Hält Befehle und Daten
- **Arithmetic Logic Unit**: Führt logische (Vergleiche) oder arithmetische (mathematische) Instruktionen aus

Control Unit

Steuerwerk, **Programmsteuerwerk** oder **Leitwerk** (englisch **Control Unit**, kurz **CU**)

Seine Hauptaufgabe besteht darin, den Ablauf der Befehlsverarbeitung zu steuern. Hierzu sendet das Steuerwerk Steuersignale über den **Steuerbus** an andere Funktionseinheiten und empfängt von ihnen Informationen.

1.FETCH (Holen): Das Steuerwerk greift auf die Arbeitsspeicheradresse zu, die im **Befehlszähler** steht. Die Daten an dieser Adresse werden ins **Befehlsregister** geladen, und der Befehlszähler wird inkrementiert.

2.DECODE (Dekodieren): Der **Decoder** analysiert den Inhalt des Befehlsregisters. Er hat Ausgänge für jeden möglichen Maschinenbefehl und Operanden. Diese Ausgänge setzen entsprechend dem Befehlsinhalt Steuersignale für die anderen Funktionseinheiten.

3.FETCH OPERANDS (Operanden holen): Je nach ausgewähltem Befehl und Operanden werden die benötigten Daten geladen oder es wird direkt mit der **EXECUTE (Ausführung)** begonnen.

4.EXECUTE (Ausführung): Alle notwendigen Daten sind in Registern vorhanden. Jetzt werden die Operationen gemäß dem Befehl ausgeführt. Einfache Befehle wie **MOV** oder **ADD** benötigen weniger Zeit, während komplexe Befehle wie **MUL** und **DIV** mehr Zeit beanspruchen können.

5.WRITE BACK (Zurückschreiben): In dieser Phase werden die Ergebnisse zurückgeschrieben, falls erforderlich.

Das Steuerwerk ist eine entscheidende Komponente der CPU und sorgt dafür, dass alle Aufgaben in der richtigen Reihenfolge ablaufen.

Interne Register

Register in einem Prozessor sind **Speicherbereiche für Daten**, auf die Prozessoren besonders schnell zugreifen können. Sie befinden sich direkt in der Nähe der Rechenwerke. In einem Prozessorkern stehen Register an der Spitze der Speicherhierarchie und sind daher die **schnellste Möglichkeit, Daten zu manipulieren**, da der Speicherzugriff unabhängig vom Daten- oder Adressbus erfolgt.

1.Hardware-Register: Diese sind physisch vorhanden und haben keine speziellen Namen. Sie entsprechen dem physischen Adressraum.

2.Architektur-Register: Diese sind für die Software “sichtbar” und entsprechen dem logischen Adressraum. Bei modernen Prozessoren sind Hardware- und Architektur-Register unterschiedliche Dinge.

3.Skalare Register: Sie enthalten einen einzelnen Datenwert, meistens Integer (Ganzzahlen), die auch als Speicheradressen verwendet werden können.

4.Vektor-Register: Diese Register enthalten mehrere Datenwerte (üblicherweise zwei bis 64) als Vektor. Die einzelnen Datenwerte können jeweils eine Größe von 8 bis 64 Bit haben und Ganz- oder Gleitkommazahlen sein.

5.Datenregister (Akkumulator): Diese werden benutzt, um Operanden für die **Arithmetisch-logische Einheit (ALU)** und deren Resultate zu speichern. Moderne Prozessorkerne besitzen oft mehrere Datenregister mit Akkumulatorfunktion.

Die Registersätze verschiedener Prozessorarchitekturen unterscheiden sich in der Größe, der Anzahl und der Art der zur Verfügung stehenden Register. Sie sind ein wesentlicher Bestandteil der CPU und ermöglichen effiziente Berechnungen und Datenmanipulationen.

Central Processing Unit

Leistung / Durchsatz

$$\frac{Befehle}{Programm} \times \frac{Taktzyklen}{Befehl} \times \frac{Ausführungsszeit}{Taktzyklus} = NIT \times CPI \times CCT$$

- **NIT** (Number of Instructions per Task)
- **CPI** (Clock Per Instruction) Anzahl Ausführungseinheiten (effektiver Durchschnitt). Wird über alle Befehlsausführungen in einem Programm gemittelt.
- **CCT** (Clock Cycle Time) je nach verwendeter Hardwaretechnologie, Architekturtechniken

Beispiel

- Programm benötigt 1 Milliarde Anweisungen zur Ausführung auf einem Prozessor mit 2 GHz
- 50 % der Anweisungen werden in 3 Taktzyklen, 30 % in 4 Taktzyklen und 20 % in 5 Taktzyklen ausgeführt

Wie lang ist die Ausführungszeit für das Programm?

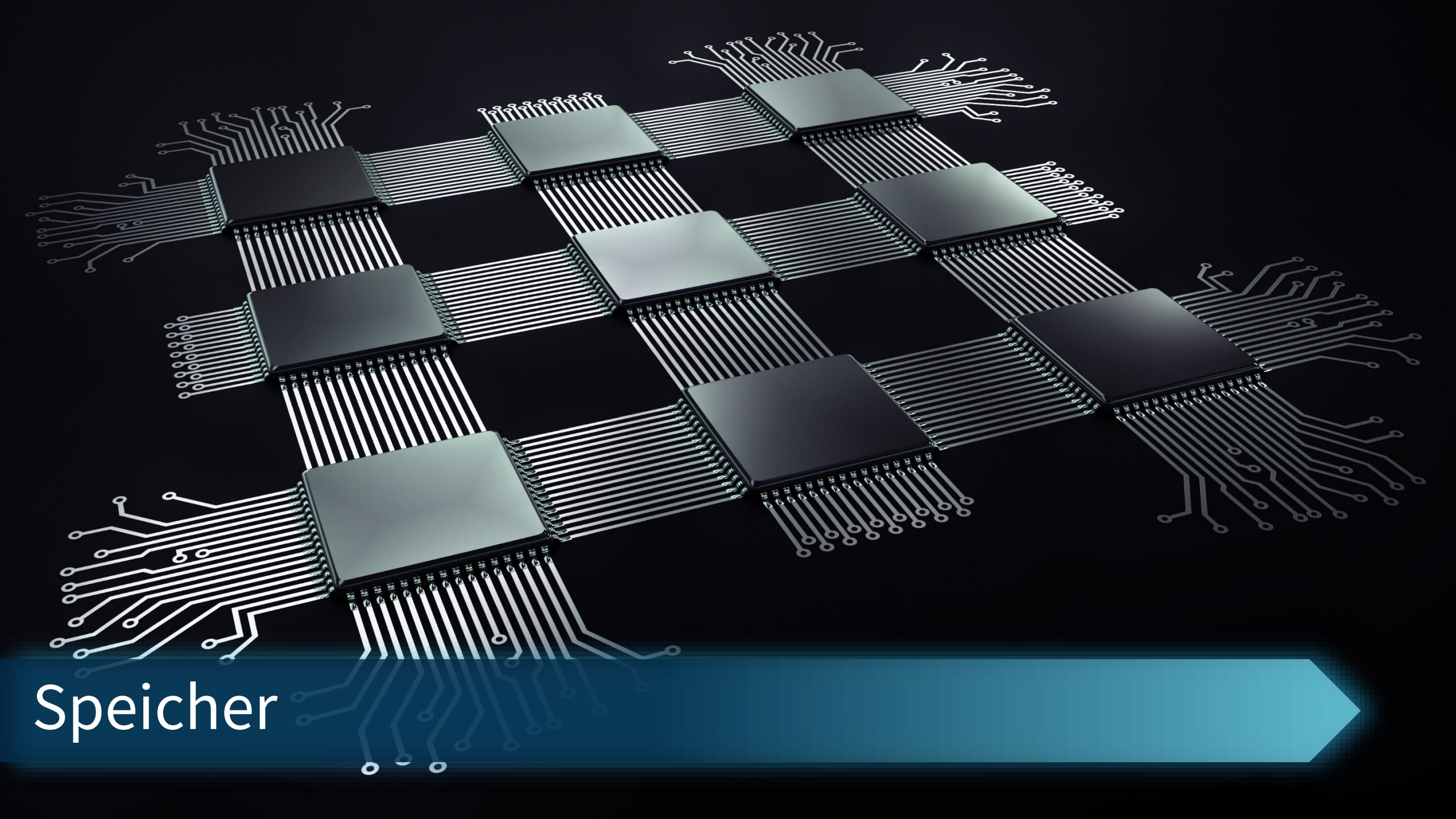
NIT = 10^9

CCT = $\frac{1}{2\ GHz}$ = 0,5 * 10^{-9}sec

	#	Frequenz	
	3 *	0,5	= 1,5
	4 *	0,3	= 1,2
	5 *	0,2	= 1,0

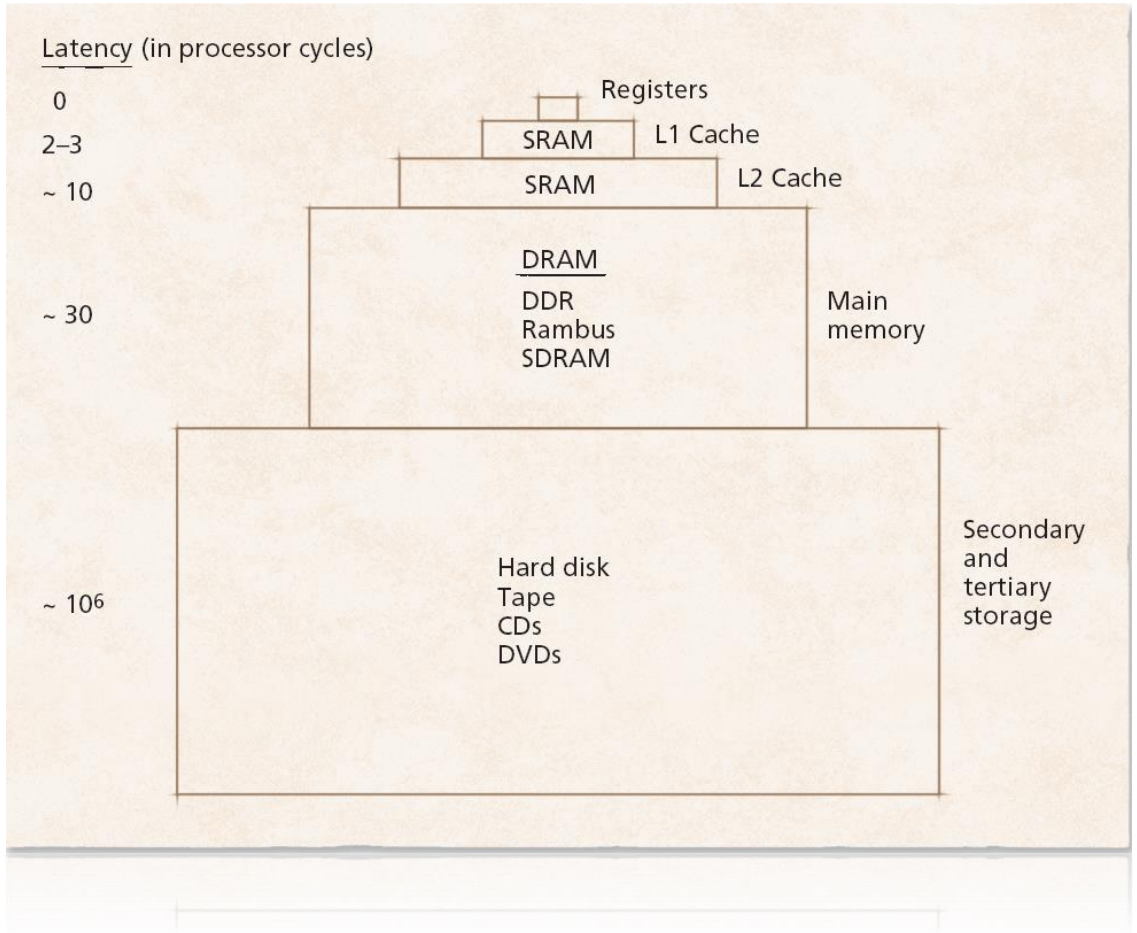
CPI = 3,7

Ausführungszeit = $1.0 \times 10^9 \times 3.7 \times 0.5 \times 10^{-9}$ sec = 1.85 sec



Speicher

Speicher



Speicherkapazität

- Speichermodul hat Speicherkapazität, welche Anzahl an Bytes angibt, die es speichern kann
- Einheiten für Kapazitäten:

Unit	Symbol	Number of Bytes
kilobyte	**KB**	**2^{10} = 1024**
megabyte	**MB**	**2^{20} (over 1 million)**
gigabyte	**GB**	**2^{30} (over 1 billion)**
terabyte	**TB**	**2^{40} (over 1 trillion)**

Speicher

Random Access Memory (RAM)

- Direct Access Memory
- Read / Write
- **Flüchtig**
- Bevor **Anweisung** ausgeführt wird, wird Programm in Hauptspeicher geladen
- Während Ausführung führt Hauptspeicher Anweisungen zur CPU zu und hält Daten aus CPU

Read Only Memory (ROM)

- Direct Access Memory
- Read Only
- **Nicht flüchtig**
- Hält **permanent** Daten oder Anweisungen, welche von CPU benötigt werden

Cache

- schneller **Zwischenspeicher** (Notizspeicher), geringere Zugriffszeit und Grösse
- sollte die als nächste benötigten **Befehle** und **Daten** aus dem HS enthalten: **räumliche** (demnächst benötigte Daten liegen wahrscheinlich bei aktuellen Daten) und **zeitliche** (aktuelle Daten werden wahrscheinlich wieder benötigt) **Zugriffsnähe** (Lokalitätsprinzip, actual working set)
- getrennter Cache für Programm und Daten erlaubt **gleichzeitiges Lesen** durch Prozessor
- meist **mehrstufig**:
  - **L1-Cache** bedient CPU-Zugriffe und holt Daten / Befehle bei Fehlzugriffen aus L2-Cache
  - **L2-Cache** bedient L1-Zugriffe und holt Daten / Befehle bei Fehlzugriffen aus HS

Einführung in die Funktionsweise von Computern

Sekundärer Speicher

- Speichert Informationen **permanent**
- Kostengünstig
- Langsamere **Zugriffszeiten**
- Beispiele
  - HDD / SSD
  - USB Sticks
  - Floppy Disks

Tertiärer Speicher

- Speichert Informationen **permanent** und über **sehr lange Zeit**
- Meist für **Backup** und **Transfer**
- Beispiele
  - Tapes

Speicherhierarchie

- Speicher wird nach **Kategorien** sortiert

- **Hierarchie** (schnellster, teuerster Speicher zuerst)
  - Register
  - L1 Cache (auf CPU Chip)
  - L2 Cache (auf CPU Chip)
  - Hauptspeicher
  - Sekundärer and tertiärer Speicher

- Hauptspeicher ist die letzte Ebene, bei der Daten direkt vom Prozessor referenziert werden

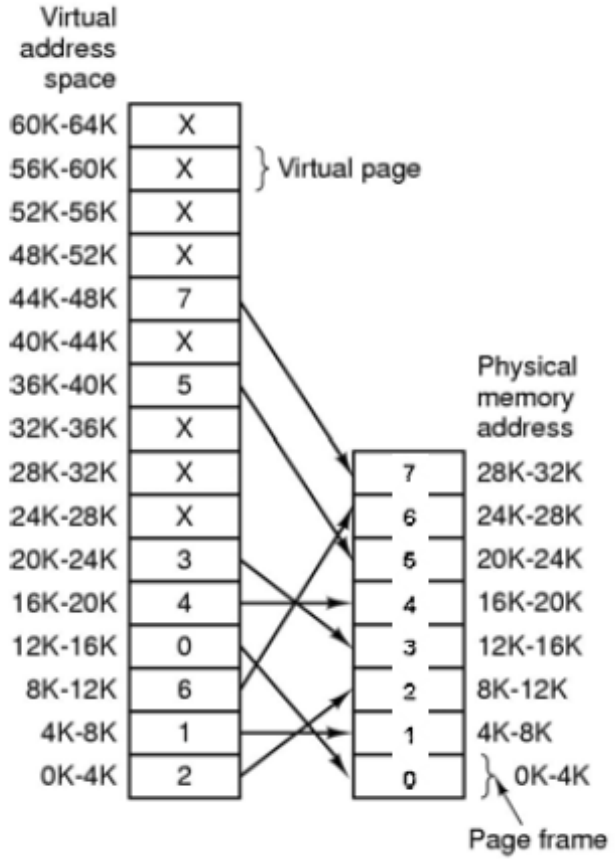
Hauptspeicher

- RAM
- Kleinste Speichereinheit: 1 Bit, hat 2 Zustände
- Mit 2 Speichereinheiten sind $2^2 = 4$ Zustände darstellbar
- Mit 8 Bit = 1 Byte = $2^8 = 256$ Zustände darstellbar
- Bytes = **kleinste adressierbaren Speichereinheit**

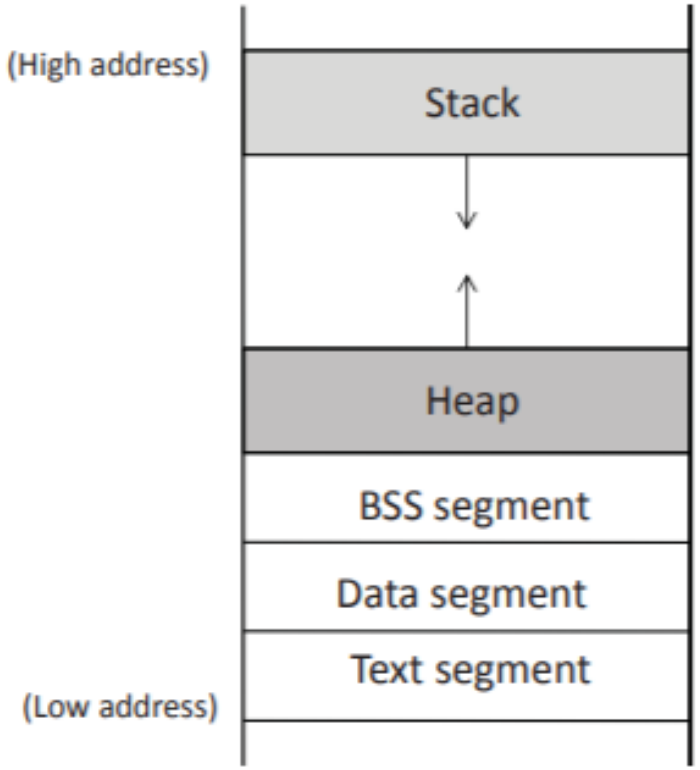
- In Bits, Bytes und Folgen von Bytes werden Befehle, Zahlen, Zeichen, usw. im Hauptspeicher abgespeichert und von der CPU verarbeitet
- Kenngrössen:
  - **Wortlänge** ("Breite") m einer Zelle: Anzahl der Bits, die auf einmal gelesen bzw. geschrieben werden, z.B. 32 Bit oder 64 Bit
  - **Grösse** ("Länge") N des HS: i.d.R. in Bytes (oder Worten) angegeben, z.B. 16 GB oder 32 GB

Informationen im Hauptspeicher speichern

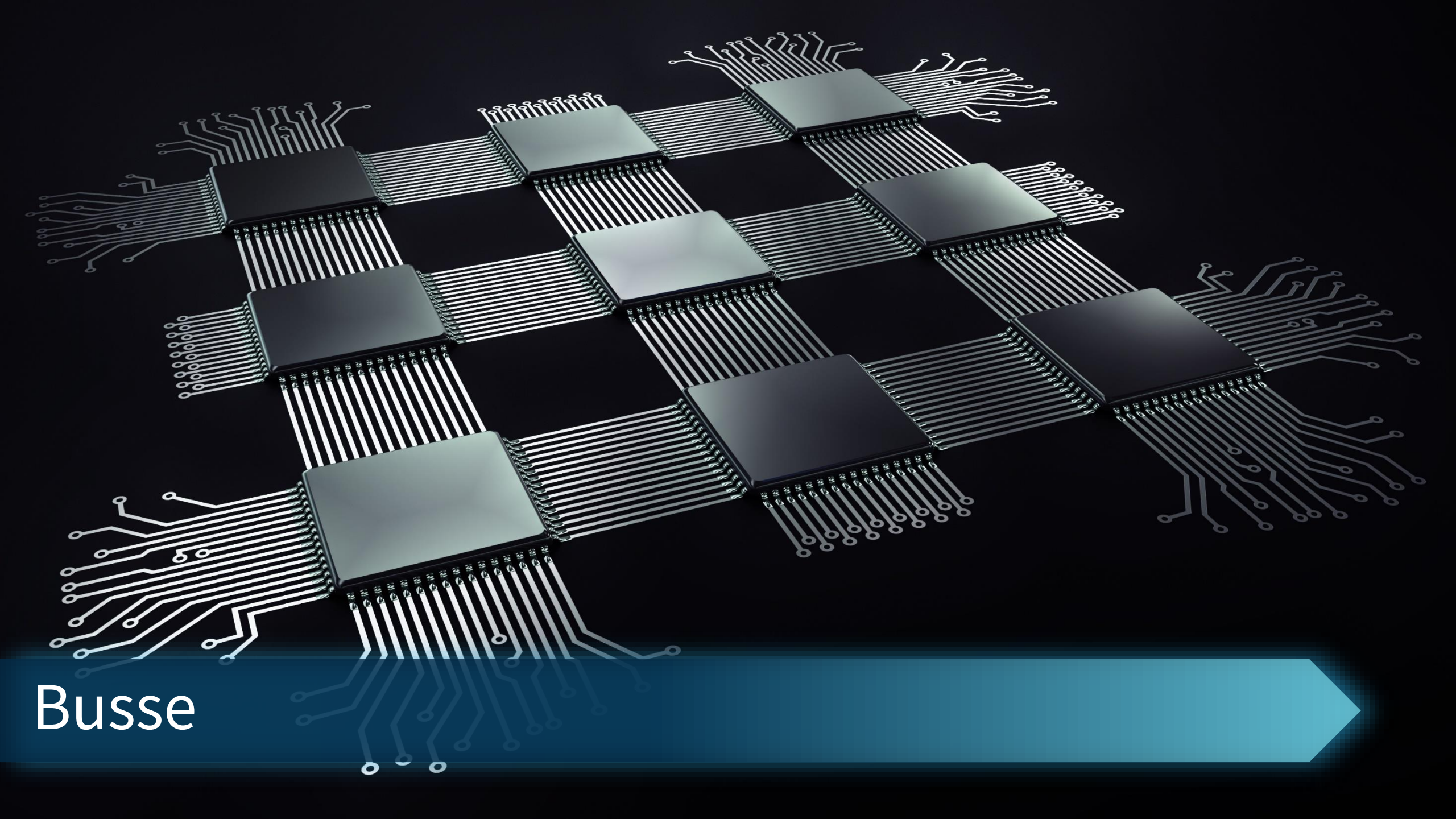
- **Problem**: Speicherbedarf für Prozesse (Programmcodes und Daten) grösser als Hauptspeicher
- Lösungen
  - **Swapping**
    - Aus- und Einlagern kompletter Prozesse
    - Anzahl, Grösse und Ort der Partitionen sind variabel
    - macht Relozierung (Anpassung der Speicheradressen) erforderlich
    - Speicherverwaltung mit Bitmaps
  - **Paging**
    - Teile des in Ausführung befindlichen Programms, die gerade gebraucht werden, werden im Hauptspeicher vorgehalten
    - Die anderen Programmteile liegen teilweise auf der Festplatte
    - Prozesse verwenden einen virtuellen Adressraum
    - Die Einheiten heissen Seiten (Pages), haben feste Größe



Informationen im Hauptspeicher speichern



- Um ein Programm auszuführen, werden seine Daten in unterschiedliche Speichersegmente unterteilt:
  - **Text Segment**: Ausführbarer Code (read only)
  - **Data Segment**: Statische / Globale Variablen, initialisiert vom Programm
  - **BSS Segment**: uninitialisierte Statische / Globale Variablen, werden vom Betriebssystem mit Nullen gefüllt
  - **Heap**: Bereich für dynamische Speicherzuweisung
  - **Stack**: lokale Variablen, die in Funktionen definiert werden, Return Adressen, Funktionsargumente

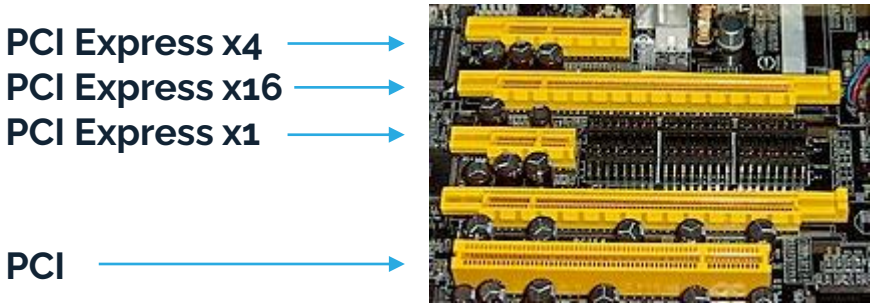


Busse

Busse

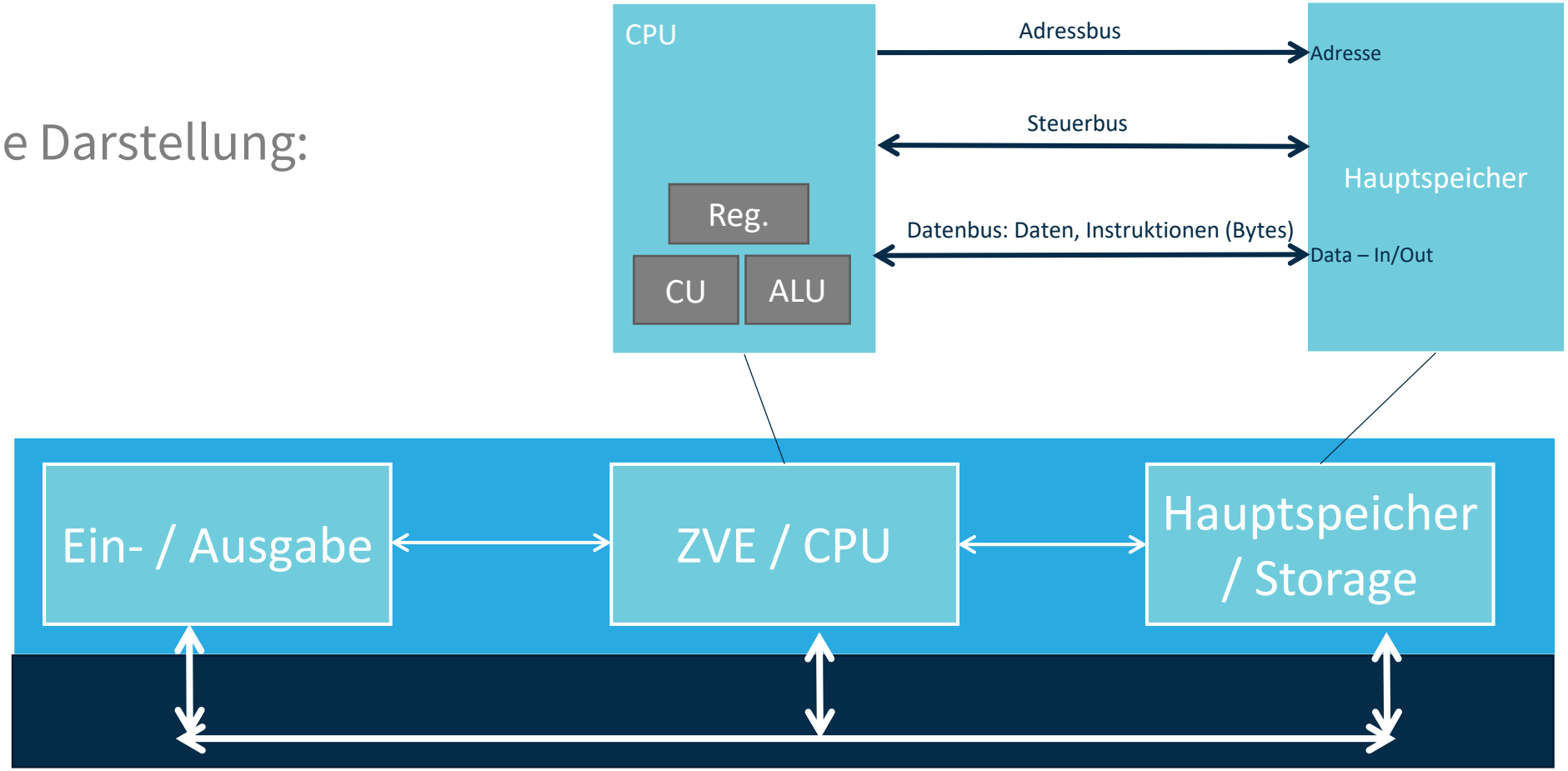
- **„Highway des Computers“,** der Informationen zwischen Einheiten transferiert
- angepasste Kommunikationspfade zwischen Komponenten
- bestehen häufig aus mehreren genormten Leitungen (Geschwindigkeit, Steuerung)
- Arten:
  - **Speicherbus:** schnell, kurz, unidirektional
  - **E/A-Bus:** länger, langsamer
  - **Steuerbus**: unidirektional
  - **Datenbus**: bidirektional
  - **Adressbus**: unidirektional

Beispiele:

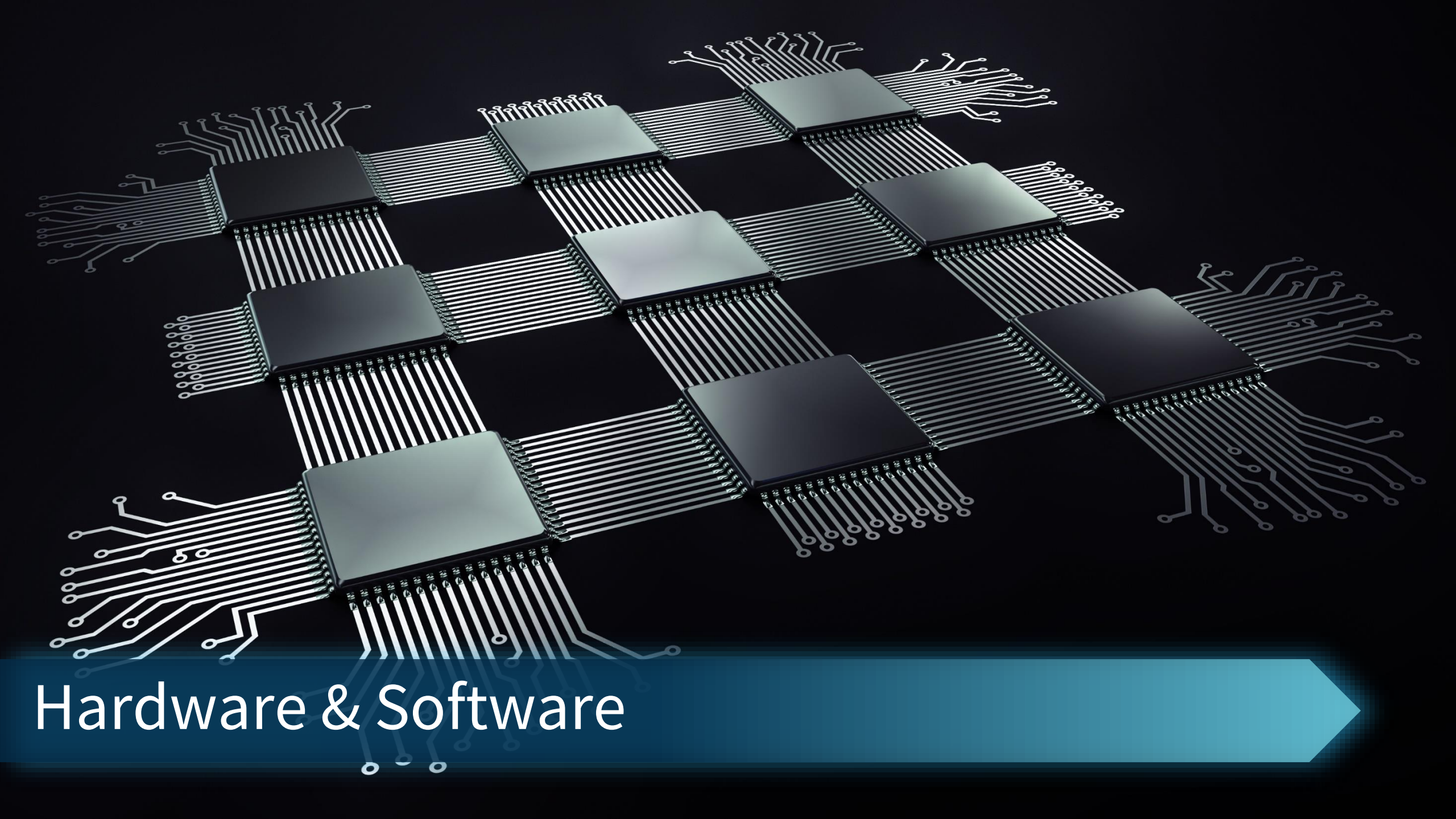


Busse

Schematische Darstellung:



Prinzipiell sind alle Busse angreifbar, wobei externe Busse z.B. für die Ein-/Ausgabe eine grössere Angriffsfläche haben.



Hardware & Software

Hardwarearchitekturen

- Physische elektronische Komponenten eines Computers

- Können in verschiedene **Kategorien** gegliedert werden:
  - **Mikroarchitekturen**: 8-Bit-Mikroprozessoren
  - **Miniarchitekturen**: 16-Bit-Prozessoren und –Rechner
  - **Spezial-Architekturen**: Burroughs, Cray-Rechner
  - **32-Bit-Architekturen**
    - 32 Bit Wort-Verarbeitung
    - Adresslänge von 31 oder 32 Bit
  - **64-Bit-Architekturen:** evolutionäre Erweiterung der Vollarchitekturen

- Software, die für aktuelle Architekturen entwickelt wird, ist lange gültig

Hardware & Software

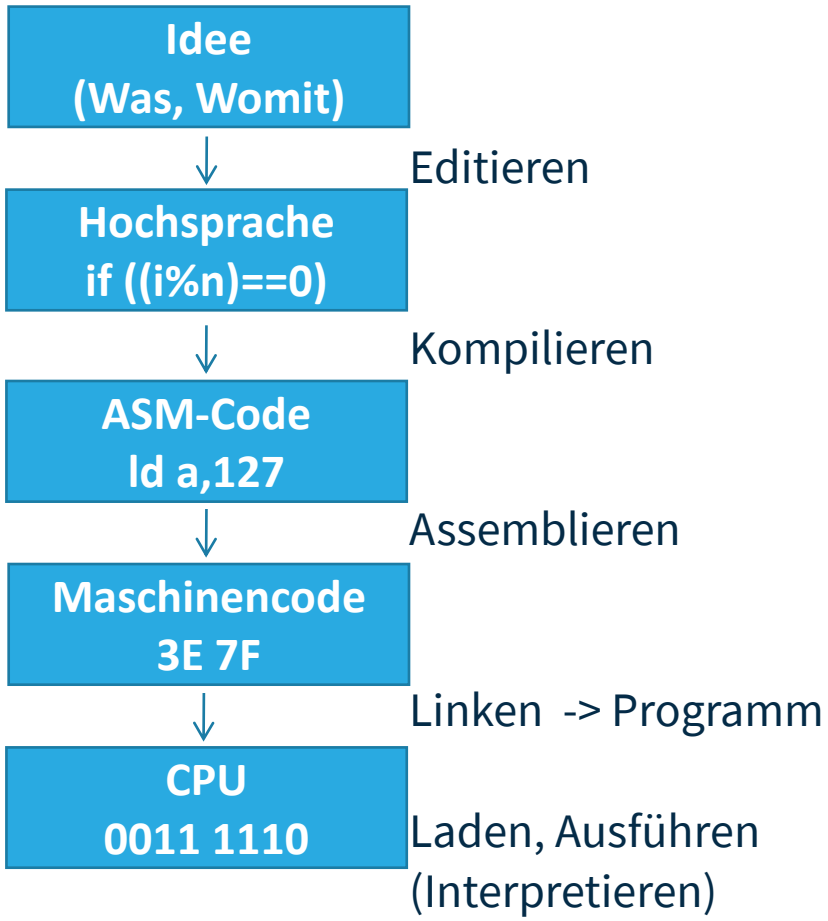
Hardware

- Alle physischen elektronischen Komponenten eines Computers sind Hardware

Software (= Programm)

- Sammlung von Computerprogrammen, welche der Hardware sagen, **was** sie tun soll und **wie**
- Sequenz von Befehlen, die der Computer ausführen kann
- wird in Computersprache geschrieben… aber CPU versteht nur bytes and bits

Software



Softwaresprachen & Sprachlevel

- Drei Programmiersprachen Level:
  - Maschinensprache
  - Assemblersprache
  - Hochsprache (z.B. C, Java)

- Jede Art von CPU versteht eine eigene, spezifische Maschinensprache
- Andere Level wurden erschaffen, um es Menschen einfacher zu machen, Programme zu lesen und zu entwickeln

Software

Assembler

- Repräsentiert Befehle in Maschinensprache
- Konvertiert Assemblersprache in Maschinensprache
- Beschleunigt Programmierung und reduziert Potential für Fehler

Maschinensprache

- Wird vom Hardware Design des Computers definiert
- Besteht aus 1en and 0en, die den Computer anweisen Basisoperationen auszuführen

Bestandteile von Betriebssystemen

Bootvorgang

Boot(-vorgang)

- Das Laden der ersten Software, welche den Computer startet
- Lädt typischerweise das Betriebssystem oder ein Boot Manager
- Kurz für “bootstrap” – ein Streifen am Schaft des Schuhs, den man ziehen kann, um das Anziehen zu vereinfachen
- “cold boot” – Anschalten des Computers aus dem Abschaltzustand
- “warm boot” – Reset eines Computers, der bereits an ist

Bootvorgang

BIOS / UEFI

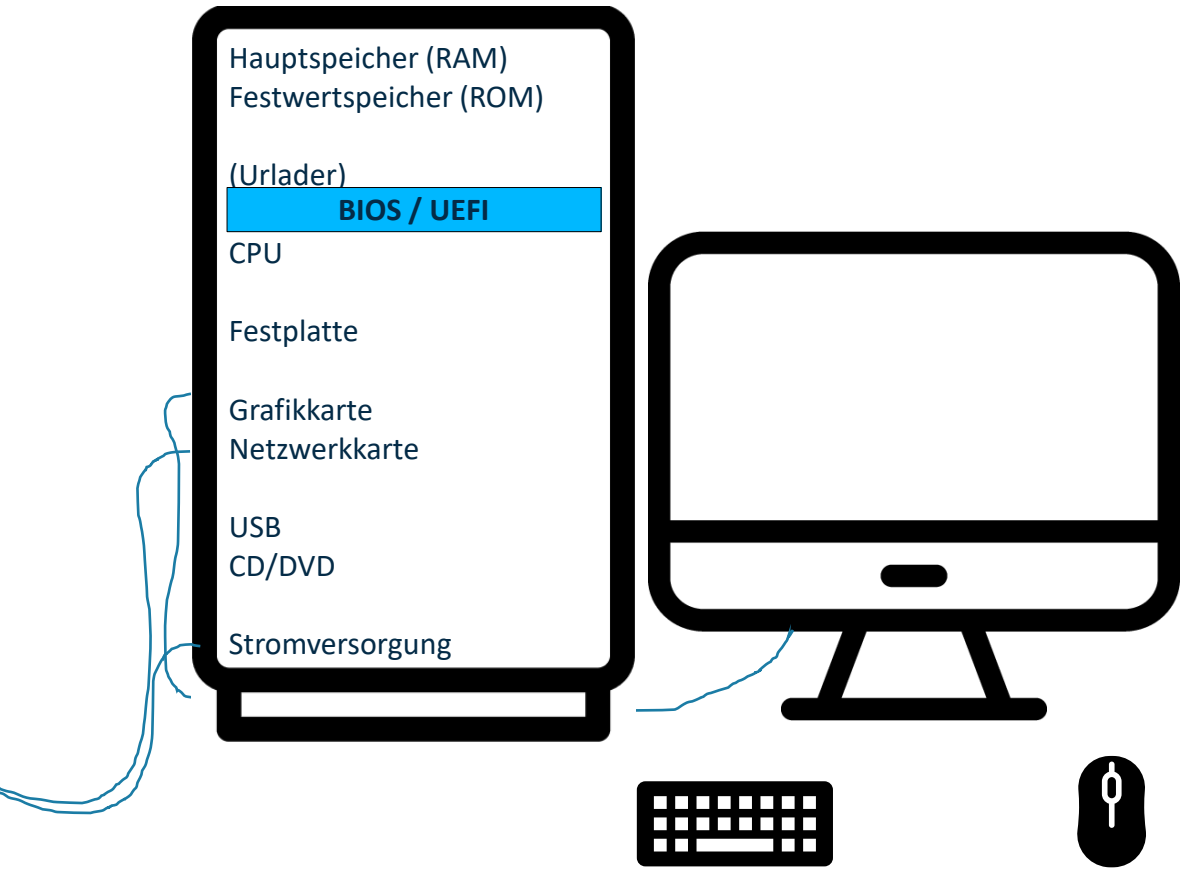
- Firmware der Computer Hauptplatine (Mainboard)
- **BIOS** = Basic Input Output System, **UEFI** = Unified Extensible Firmware Interface
- Persistenter Code, der beim Powerup schon ausgeführt wurde

Bootvorgang

Boot Manager

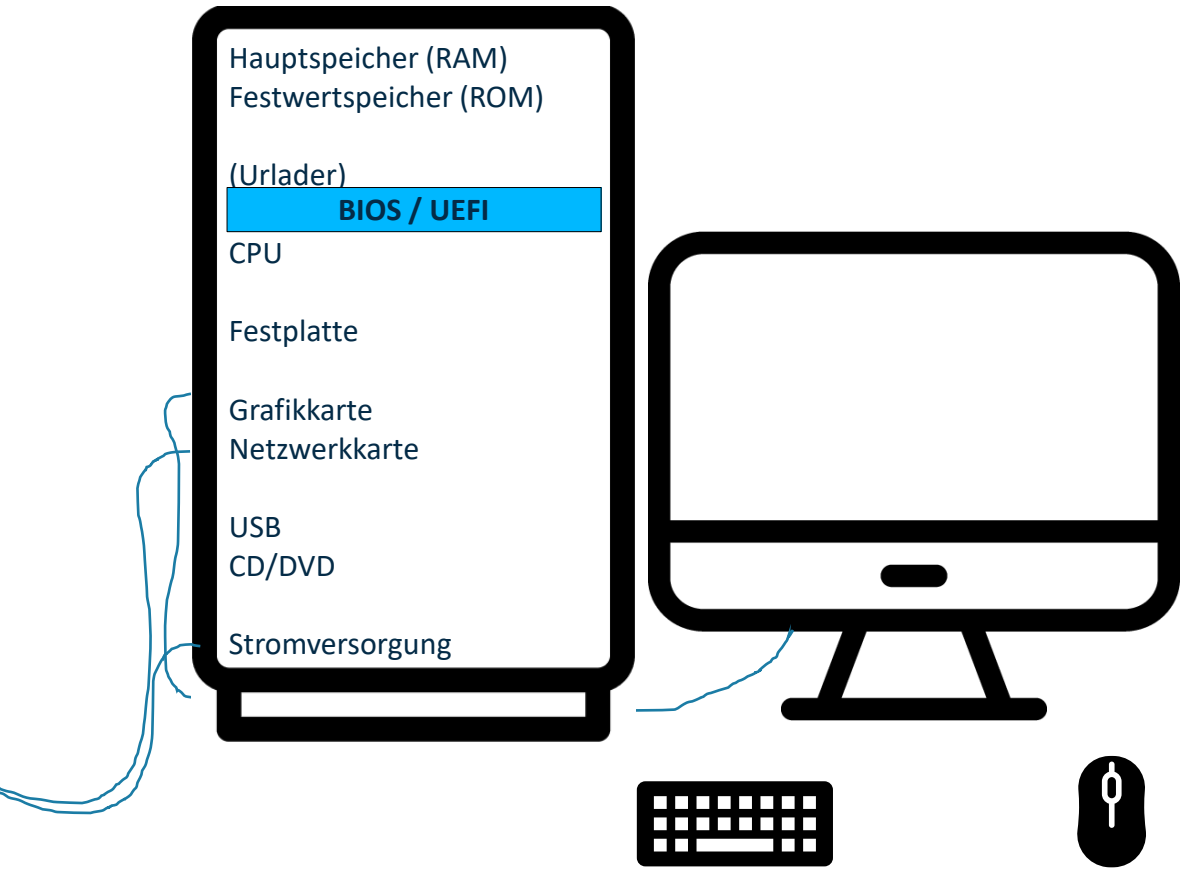
- Code, welcher vom Firmware Bootstrap geladen wird
- Ermöglicht die Auswahl eines **Boot Image** oder die Spezifikation von **Boot Parametern** etc.
- Fügt einen weiteren “layer” zum Bootprozess hinzu und erhöht die **Flexibilität**, z.B. durch “multiboot” Konfigurationen

Definition: Betriebssystem



- isoliert **Anwendersoftware** von der **Hardware**: Betriebssystem läuft auf Hardware, Anwendersoftware auf Betriebssystem
- Läuft immer
- verwaltet **Ressourcen** des Computers, stellt sie zur Verfügung, kontrolliert und koordiniert deren **Nutzung**
- Wie „**virtuelle CPU**“: durch Abstraktion und Transparenz einfacher zu handhaben als reale CPU → schützt vor Fehlbedienung

Boot Ablauf: Vom Power On zum Betriebssystem



- Power On Self-Test (POST)
- Initialisierung der Hardware
- Optional: Aufforderung zur Eingabe eines BIOS- oder UEFI- Passworts
- Optional: Startbildschirm
- Optional: BIOS- oder UEFI-Konfigurationsmenü
- Aufrufen von Erweiterungen, die auf Steckkarten untergebracht sind
- Urlader: Feststellen, von welchem Datenträger gebootet werden soll und Laden des Software-Bootloader von diesem Datenträger.

Definition: Urlader

- Programm, welches Bytes von einer **Bootpartition** in den Speicher schreibt und dann der **CPU Kontrolle** über die neu geladenen Bytes gibt (executable)
- Programm, welches das erste Programm lädt (“loader loader”)
- Erfordert Firmware support (“hardware bootstrap”)

Boot Ablauf: Vom Power On zum Betriebssystem

- **Urlader** lädt Programm von Bootsektor der CD oder USB Stick, welches während der Ausführung den **Lader** nachlädt und im Bootsektor der Festplatte (MBR oder Protective-MBR) speichert
- Bei jedem nachträglichen Start lädt der Urlader den Lader (Betriebssystem) vom Bootsektor eines bootfähigen Mediums
- Optional: der Urlader kann auch den Boot Manager laden, welcher dann wiederrum das Betriebssystem lädt

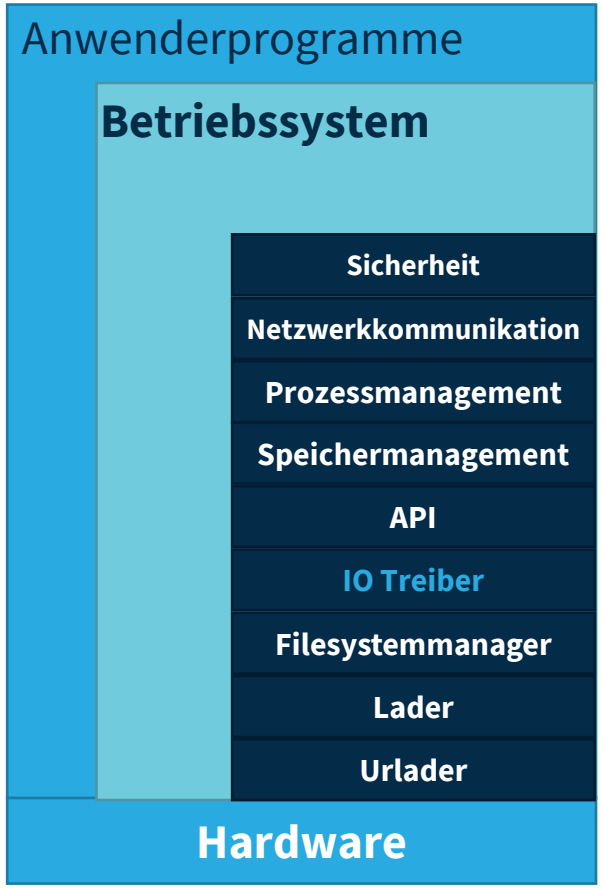
Dateiverwaltungssysteme

- **Systemsoftware**
- Zugriff auf Dateien für Benutzer oder Anwendungen
- **Aufgaben**:
  - Speichern von Daten
  - Gewährleistung der Integrität und Verfügbarkeit
  - Leistungsverhalten optimieren
  - E/A-Schnittstellenroutinen zur Verfügung stellen
  - E/A-Aufträge mehrerer Benutzer koordinieren

Dateiverwaltungssysteme

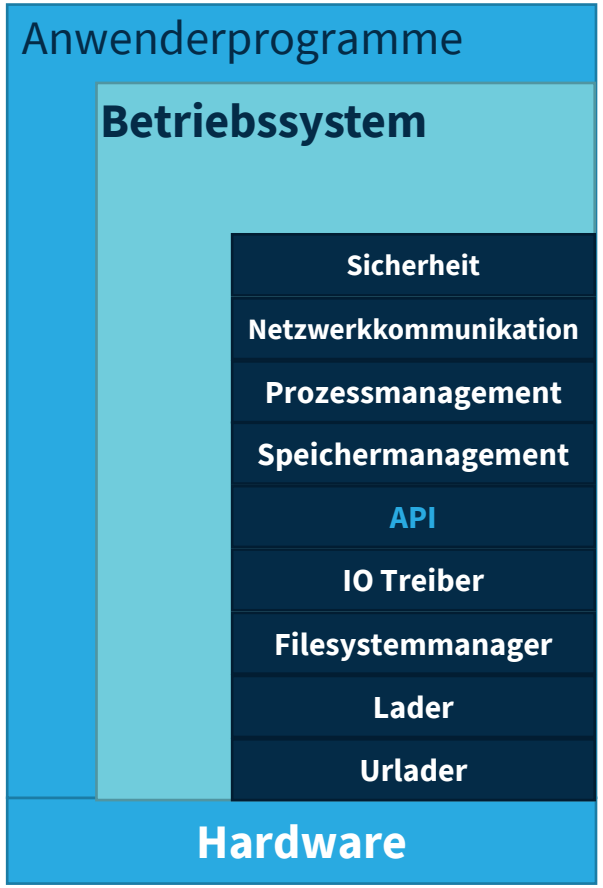
- **Anforderungen:**
  - Dateien erstellen, löschen, lesen und ändern
  - kontrollierbaren Zugriff auf Dateien anderer Benutzer
  - festlegen, welche Zugriffsrechte für Dateien gelten
  - Dateien umstrukturieren können
  - Daten von einer Datei in eine andere transferieren
  - Dateien sichern und wiederherstellen

IO Treiber



IO Treiber

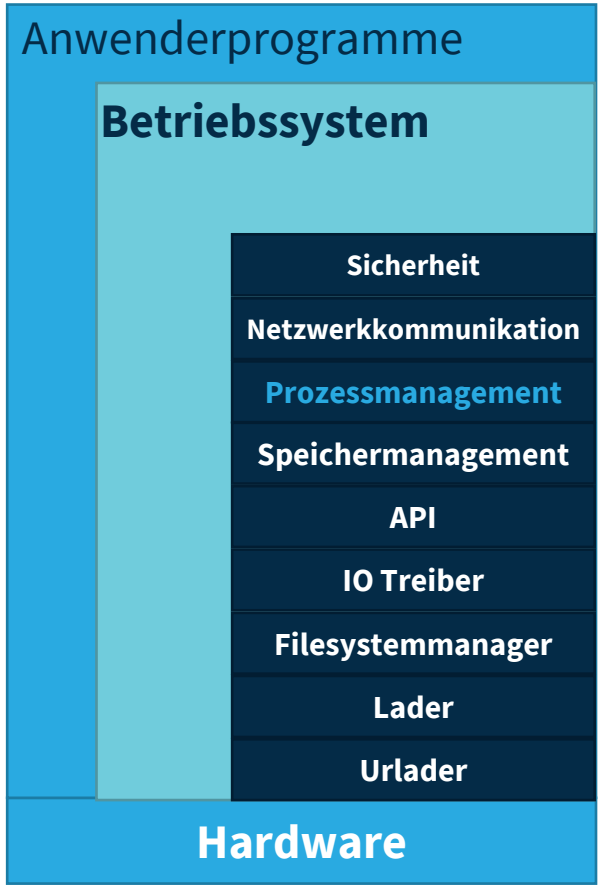
- **Softwaremodule** zur Ansteuerung von Hardware
- In der Regel integriert ins Betriebssystem
  - Kürzere Programme
  - Besser wartbare Programme
  - Softwareentwickler brauchen keine Hardware Kenntnisse
- Mit der Hauptplatine gekoppelte IO haben primitive Treiber im BIOS



API

- **Programmierschnittstellen** für Systemaufrufe
- Ermöglichen Übersichtlichkeit, Wartung, Pflege, Austauschbarkeit von Programmen durch Abstraktion und Transparenz

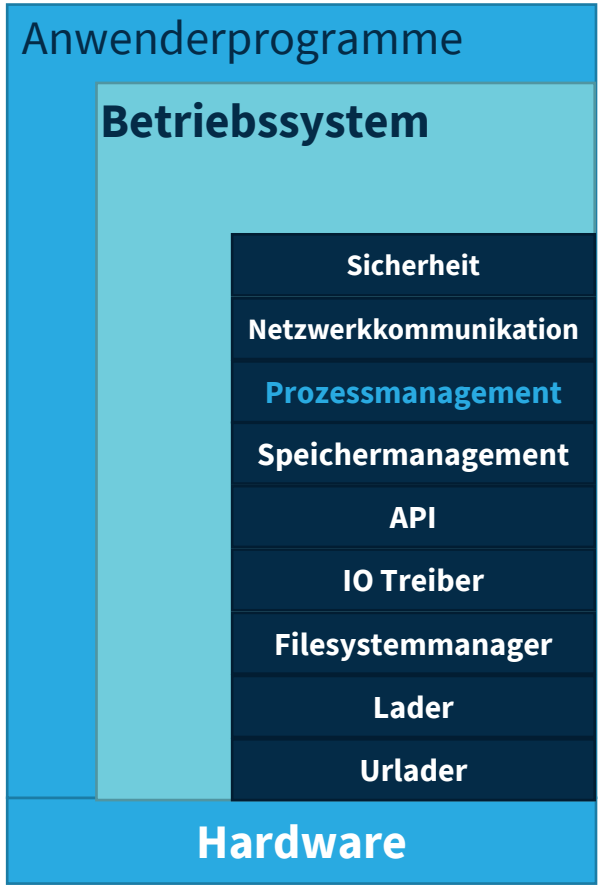
Prozessmanagement



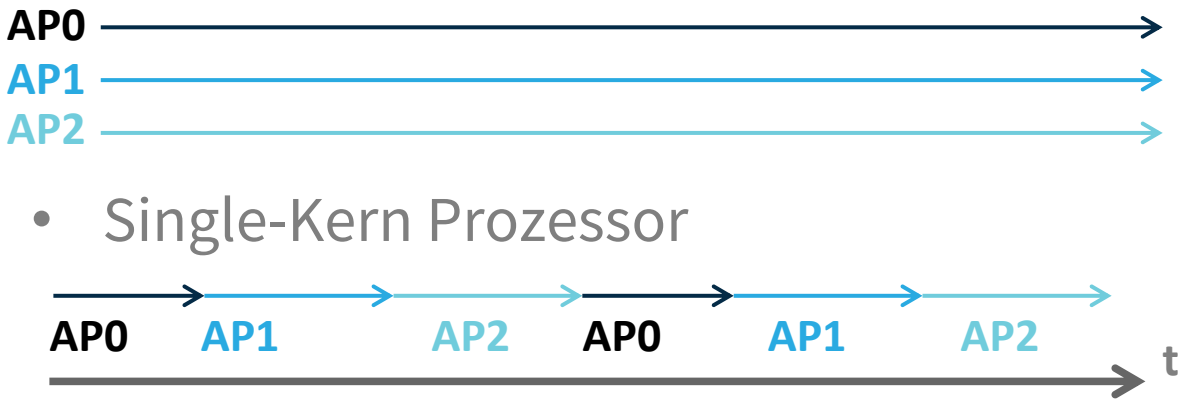
Prozess

- **Programme** unter Kontrolle des Betriebssystems
- Betriebssystem sorgt dafür, dass das Programm eines Prozesses „ordentlich“ ausgeführt wird.

Prozessmanagement

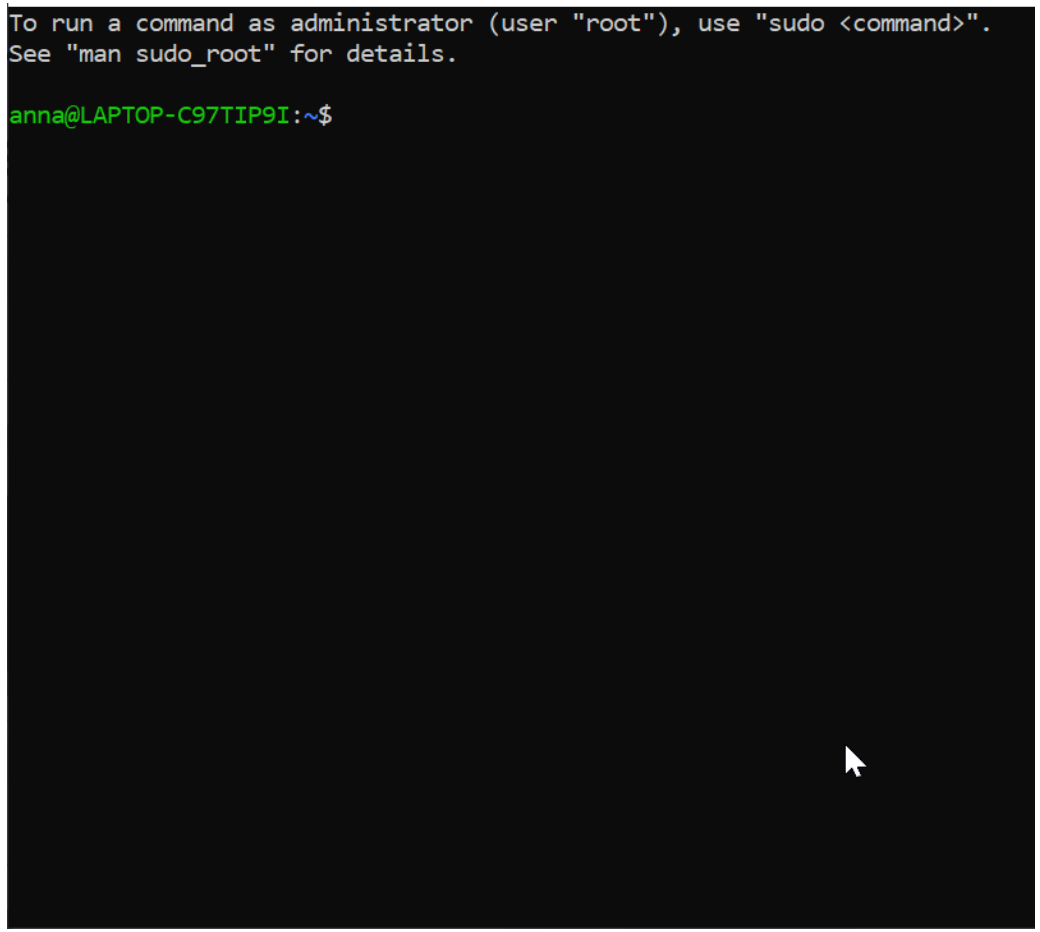


- **Scheduling**
- Interprozess **Kommunikation**
- Moderne Betriebssysteme können mehrere Programme parallel oder quasiparallel starten und verwalten
- Mehrkernprozessor



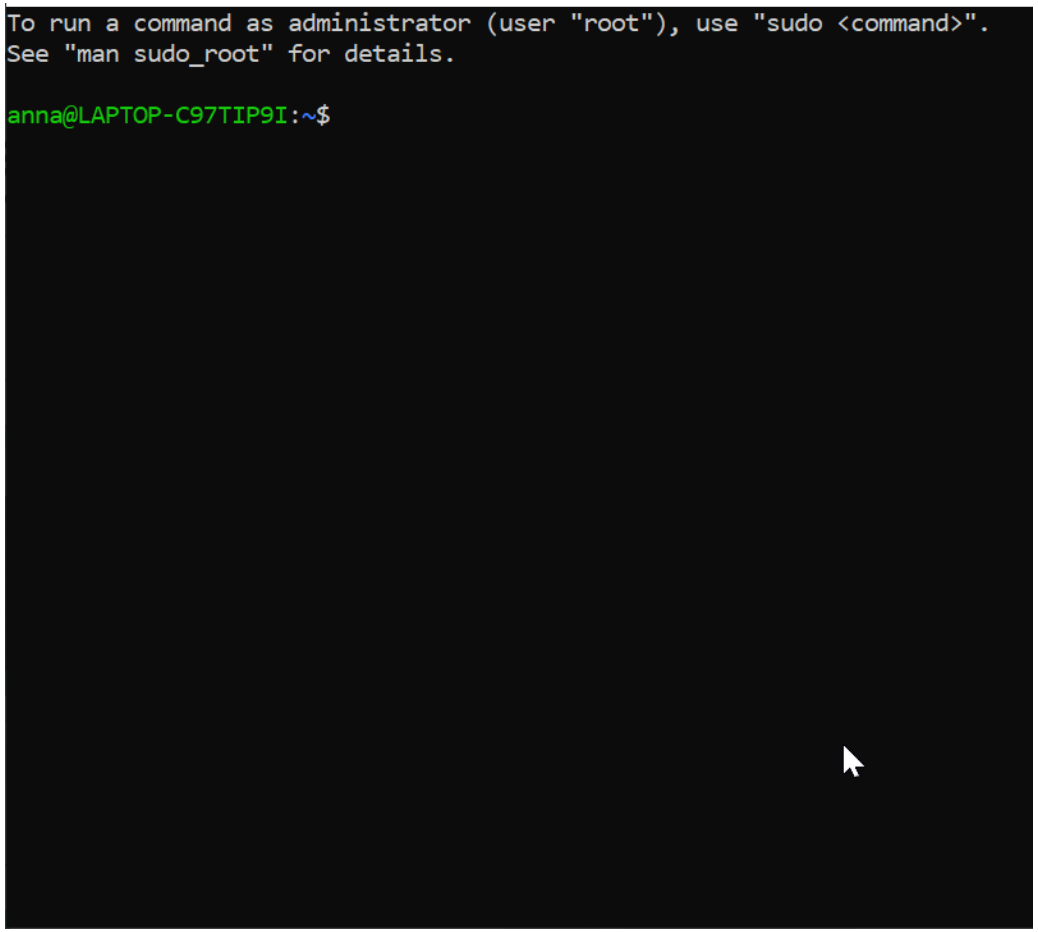
- Single-Kern Prozessor

Bedienschnittstelle



- **Dienstprogramm**
- Verschiedene Benutzeroberflächen (Shells)
  - Character User Interface (CUI) oder Kommandointerpreter (z.B. Unix)
  - Graphical User Interface (GUI) / Grafische Benutzeroberfläche (z.B. MacOS, Windows)
- Braucht nicht Bestandteil des Betriebssystems zu sein

Bedienschnittstelle



- Ermöglicht
  - Wartung und Pflege des Dateisystems
  - File-Handling
  - Starten und Überwachung von Anwenderprogrammen
  - Ermittlung des Systemstatus, Systemwiederherstellung
  - Gewährleitung der Systemadministration
    - Accounts
    - Abrechnungsdaten
    - Rechte
    - Systemkonfiguration
  - Programmentwicklung auch für Fremdrechner

Dilbert

«Evolution» der Programmentwicklung und Bedienschnittstelle

Einführung in die Labore

Disclaimer

- Die nachfolgenden Slides sind nicht Prüfungsrelevant
- Falls Sie gerade live zuschauen, bitte dranbleiben!
- …aber falls Sie diese Präsentation während der Prüfungsvorbereitung sehen, können Sie den Rest überspringen.

Wofür sind die Labs?

- Praktische Seite Operation System Security

- Einzelne Themen und Konzepte im Detail kennenlernen

Vier grosse Labor-Übungen

- SW XX: Host Intrusion Detection
- SW XX: Mandatory Access Control
- SW XX: Buffer Overflow
- SW XX: Kapselung

Vor SW XX wird Feedback zu den Labs eingeholt

SECURNITE

Ablauf der Labore

- Selbständiges Bearbeiten der Labs mit Anleitung
- Fragen via Zoom

- Sie können die Labs auch ausserhalb der Labore lösen
- Potenziell brauchen Sie mehr Zeit als die zwei Stunden
- Ausserhalb der Zoom-Zeit: Discord oder E-Mail
- Theorie-Inputs am Ende der Vorlesungen, nicht während der Labs