IT- und Medientechnik

Vorlesung 1: 16.10.2023

Wintersemester 2023/2024 h_da

Heiko Weber, Lehrbeauftragter

Teil 1: IT- und Medientechnik
Inhalt der Vorlesung laut Modulhandbuch

- Darstellung Hard- und Software
- gängige Softwareentwicklungsmethoden
- Aufbau von Netzwerken und des Internets
- Techniken der Datenübertragung mit Intra- und Internet
- Cloud-Dienste und Internet-der-Dinge
- Digitalisierung von Inhalten und elektronische Übertragung in den gängigen Medien

Teil 1: IT- und Medientechnik
Themenübersicht der Vorlesung

- **Hardware und Software**
  - Hardware: CPU, Speicher, Bus, I/O, ...
  - Software: System-, Unterstützungs-, Anwendungssoftware
- **Software-Entwicklung**
  - Quellcode, Programmiersprachen, Dokumentation
  - Software-Entwicklungsphasen, -Entwicklungsmethoden, -Komponenten, -Lizenzen
- **Digitalisierung von Inhalten und Informationen**
  - Daten: Informationen, Bits, Bytes, Binärsystem, Hexadezimalsystem
  - Dateien, Dateisysteme, Datenformate, Kompression
- **Netzwerke und Internet**
  - Netzwerk-Strukturen, Geschichte des Internets, Techniken der Datenübertragung
  - Protokolle und Technologien, Standards, WWW, Cloud, IoT
- **Weitere Technologien**
  - Datensicherung, Virtuelle Maschinen

Teil 1: IT- und Medientechnik
Themenübersicht der Vorlesung

- **Hardware und Software**
  - Hardware: CPU, Speicher, Bus, I/O, ...
  - Software: System-, Unterstützungs-, Anwendungssoftware
- **Software-Entwicklung**
  - Quellcode, Programmiersprachen, Dokumentation
  - Software-Entwicklungsphasen, -Entwicklungsmethoden, -Komponenten, -Lizenzen
- **Digitalisierung von Inhalten und Informationen**
  - Daten: Informationen, Bits, Bytes, Binärsystem, Hexadezimalsystem
  - Dateien, Dateisysteme, Datenformate, Kompression
- **Netzwerke und Internet**
  - Netzwerk-Strukturen, Geschichte des Internets, Techniken der Datenübertragung
  - Protokolle und Technologien, Standards, WWW, Cloud, IoT
- **Weitere Technologien**
  - Datensicherung, Virtuelle Maschinen

Hardware vs. Software

Hardware

- die physischen Bestandteile eines Computers oder Zubehör, welches an einen Computer angeschlossen werden kann

Software

- eine Reihenfolge von Anweisungen, um den Zustand der Hardware-Komponenten eines Computers in einer speziellen Reihenfolge zu verändern
- normalerweise in Programmiersprachen geschrieben, die dann in Maschinensprache übersetzt wird, damit der Computer sie versteht

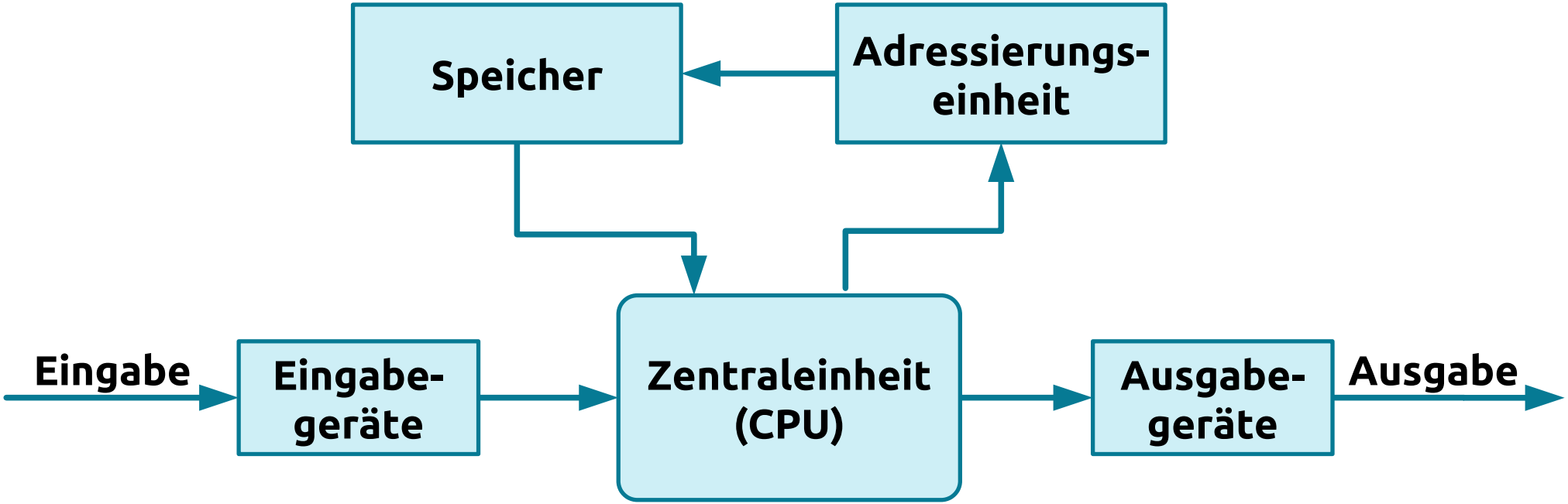
Hardware vs. Software

	Hardware	Software
Definition	Physische Geräte, die benötigt werden um Software zu speichern und auszuführen.	Eine Sammlung von Anweisungen, die es einem Benutzer ermöglichen mit einem Computer zu interagieren. Alle nichtphysischen Bestandteile eines Computers.
Typen	Ein-/Ausgabegeräte, Speicher, Verarbeitungsgeräte, Kontrollgeräte, ...	System-, Unterstützungs-, Anwendungssoftware.
Beispiele	CD-ROM, Monitor, Drucker, Grafikkarte, Mouse, Festplatte, ...	Firefox, Microsoft Word, Ubuntu Linux, OpenOffice, ...
Funktion	Die Hardware dient als Ausführungsplattform für Software.	Anweisungen an die Hardware, um spezielle Aufgaben zu erledigen.

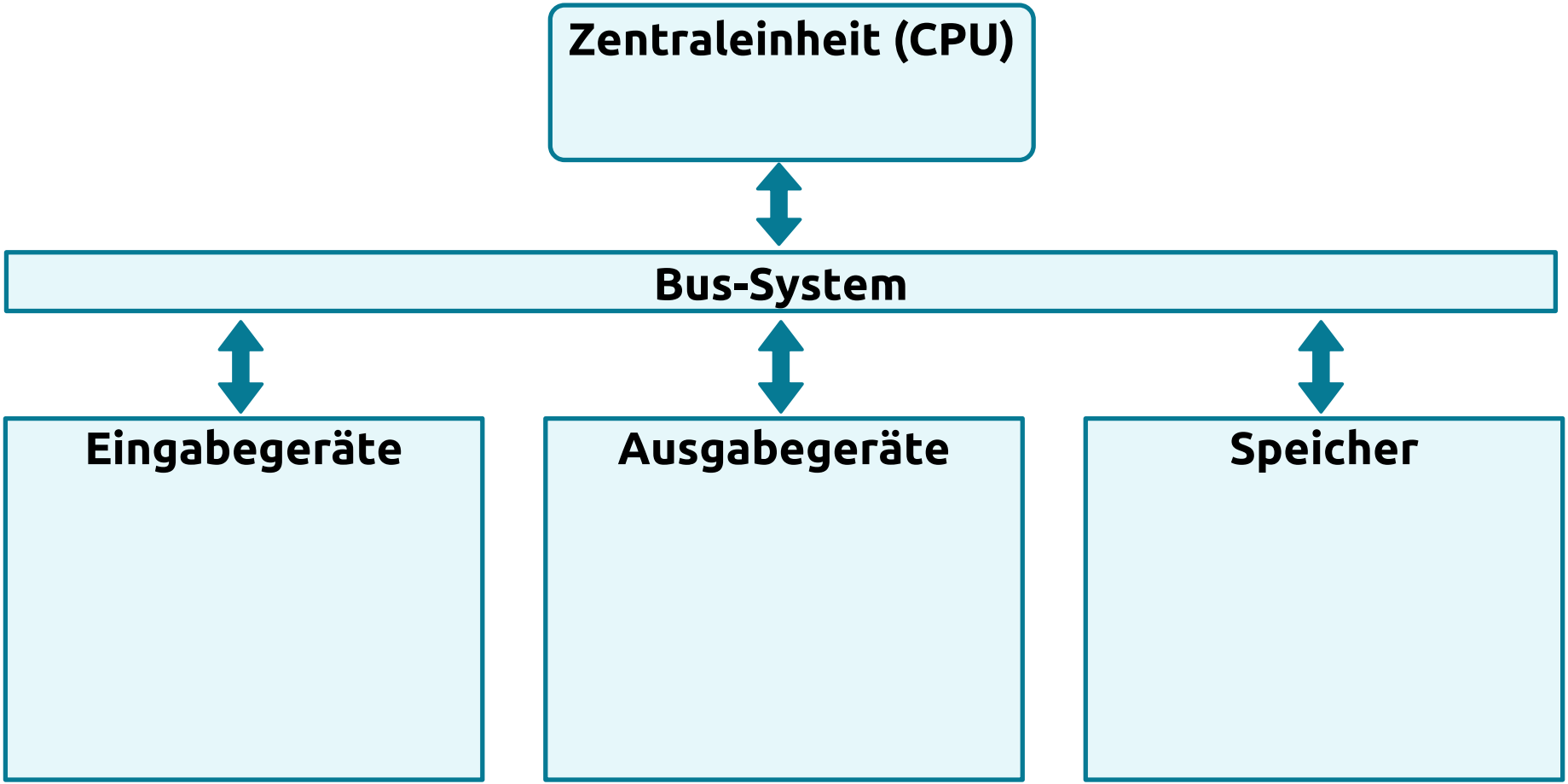
Hardware vs. Software

	Hardware	Software
Abhängigkeiten	Hardware beginnt zu funktionieren, nachdem die Software gestartet wurde.	Zum Ausführen von Software, muss sie auf Hardware installiert sein und auf diese zugreifen können.
Ausfall	Hardware-Ausfall ist mehr oder weniger zufällig, meist in Abhängigkeit mit der Lebensdauer oder Frequenz der Benutzung.	Software-Ausfall ist systematisch, in Abhängigkeit der Programmfehler. Software wird mit dem Alter nicht fehleranfälliger.
Haltbarkeit	Hardware wird mit der Zeit „abgenutzt“.	Software wird nicht „abgenutzt“ mit der Zeit, aber mit der Zeit werden Fehler gefunden.

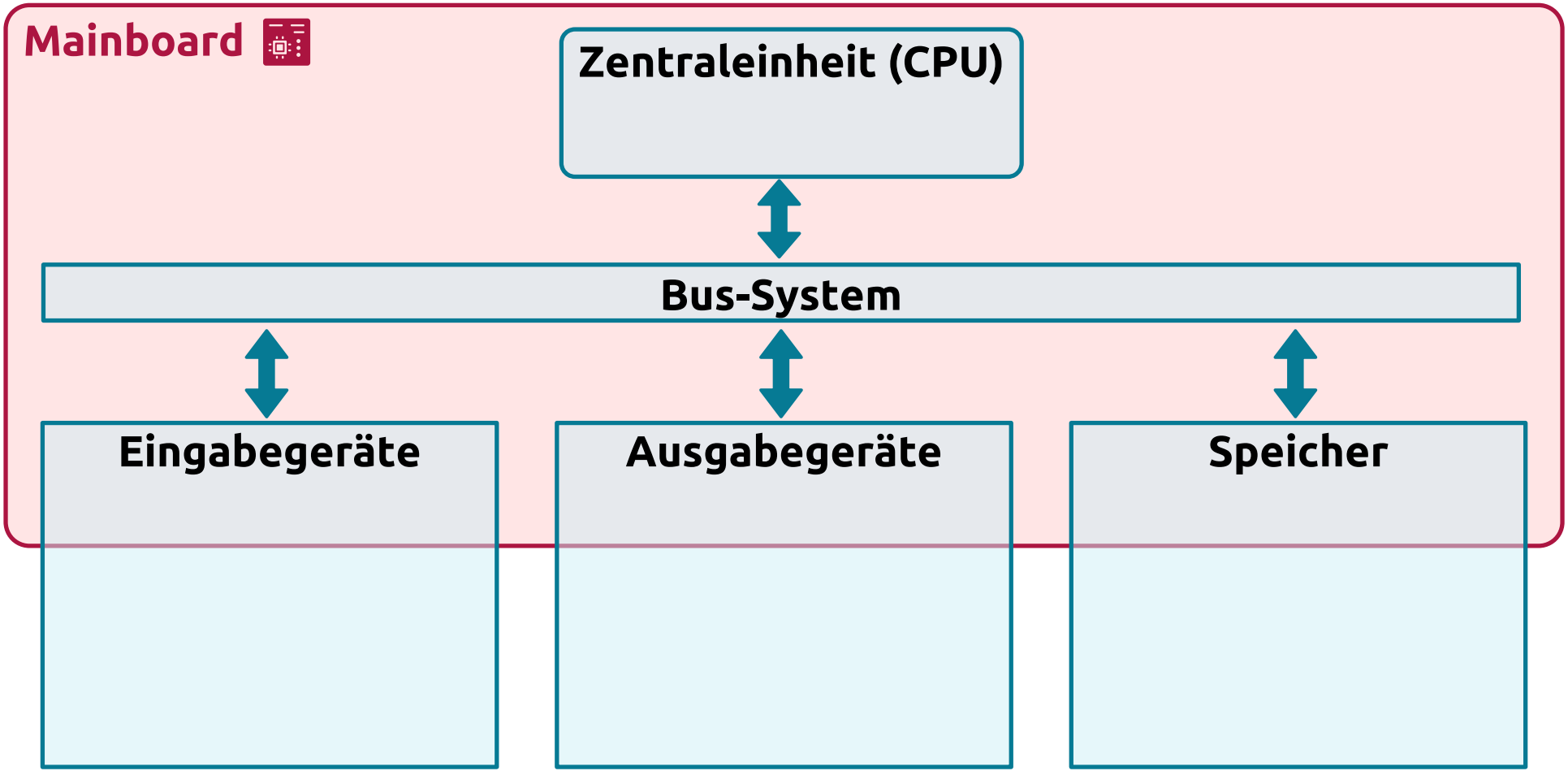
Logische Komponenten eines Computers



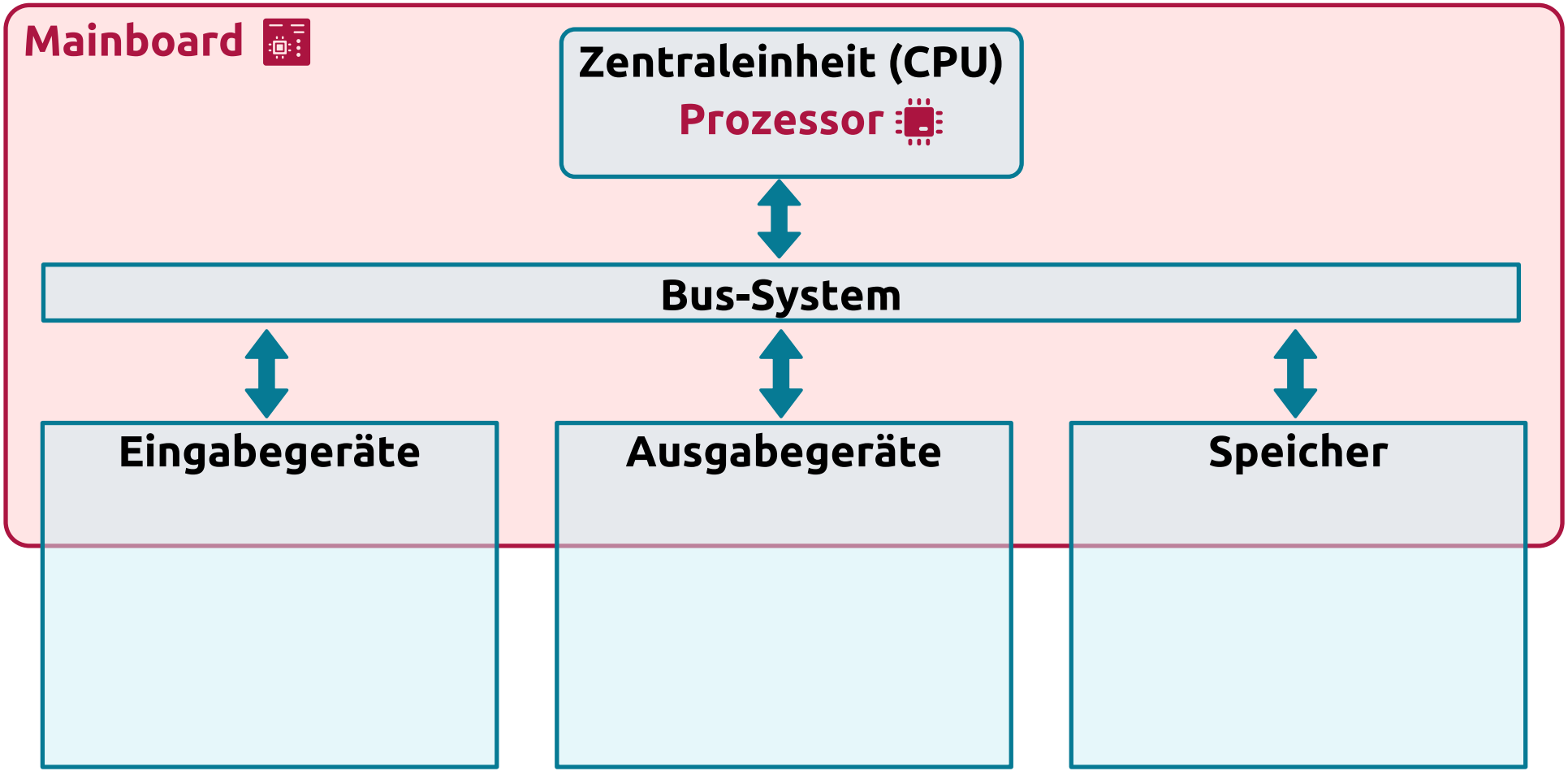
Physische Komponenten eines Computers



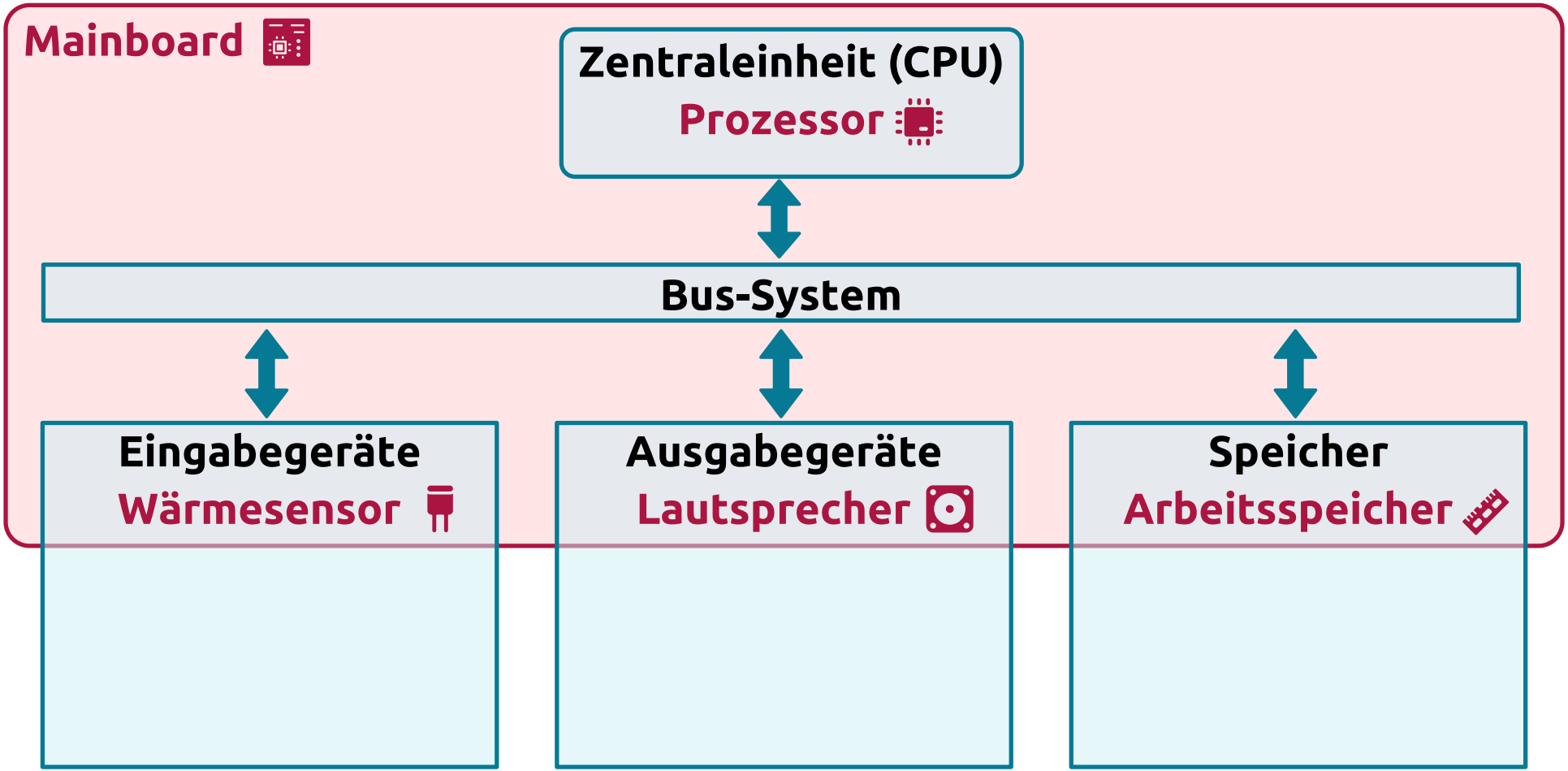
Physische Komponenten eines Computers



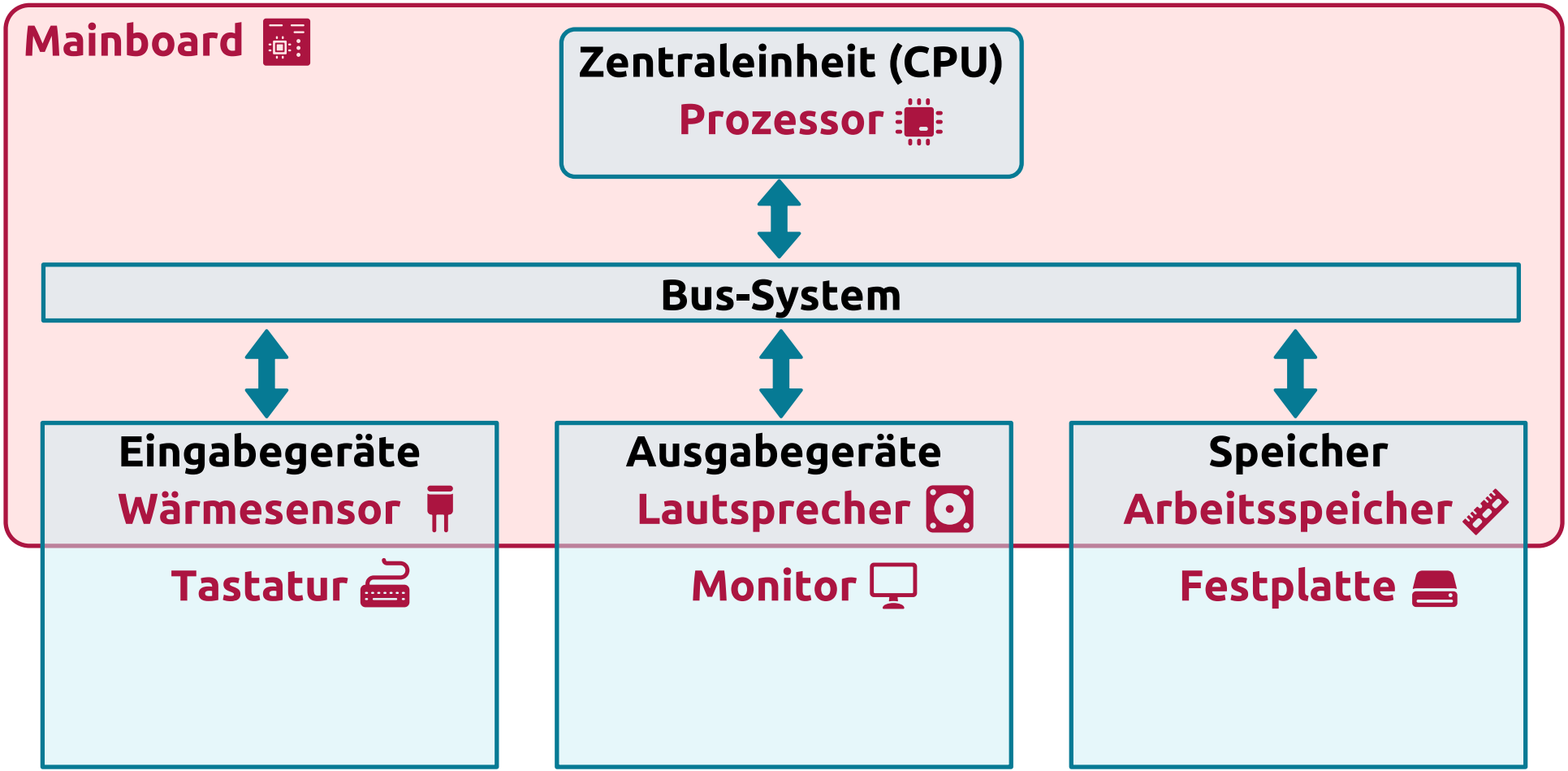
Physische Komponenten eines Computers



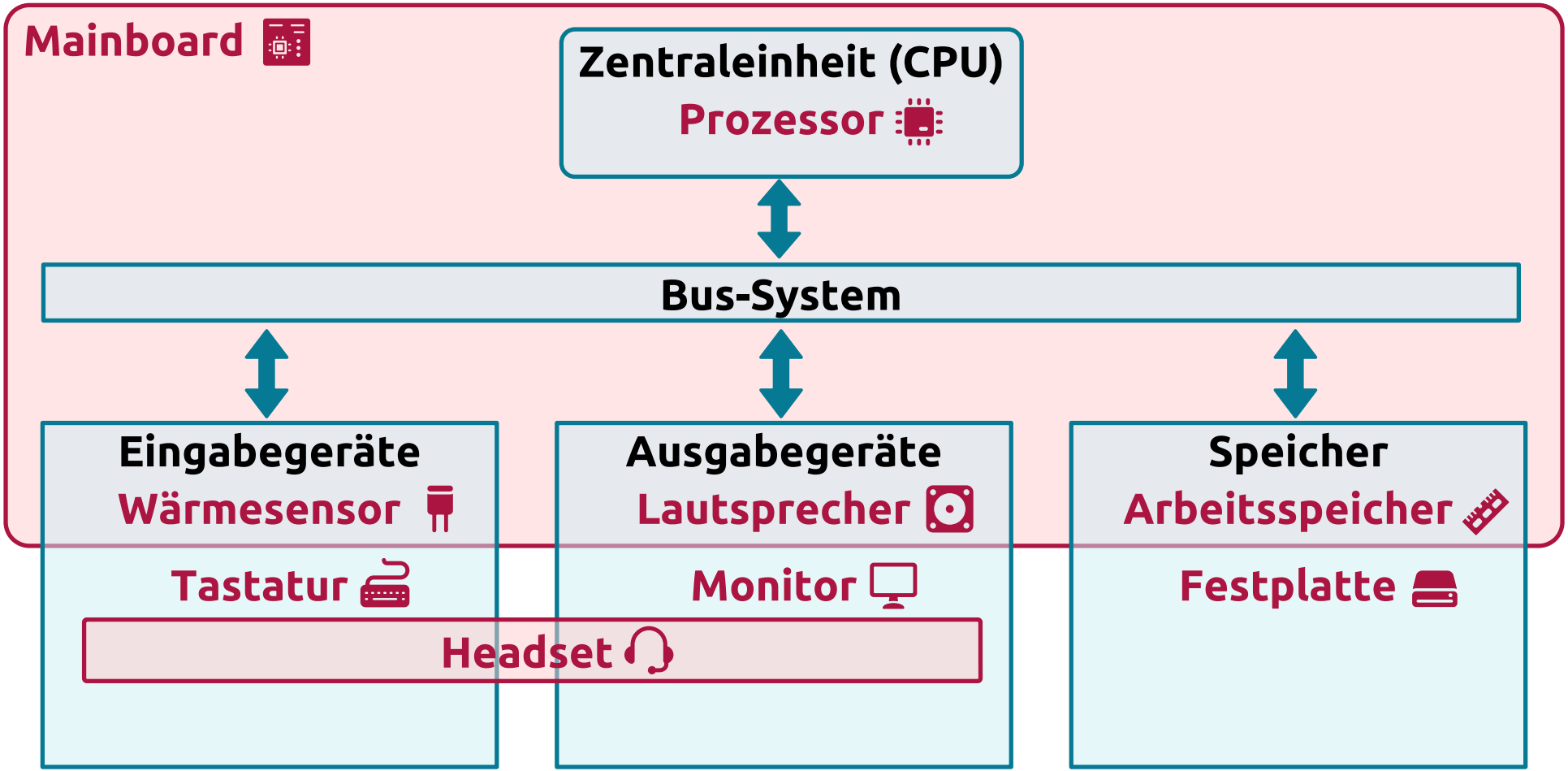
Physische Komponenten eines Computers



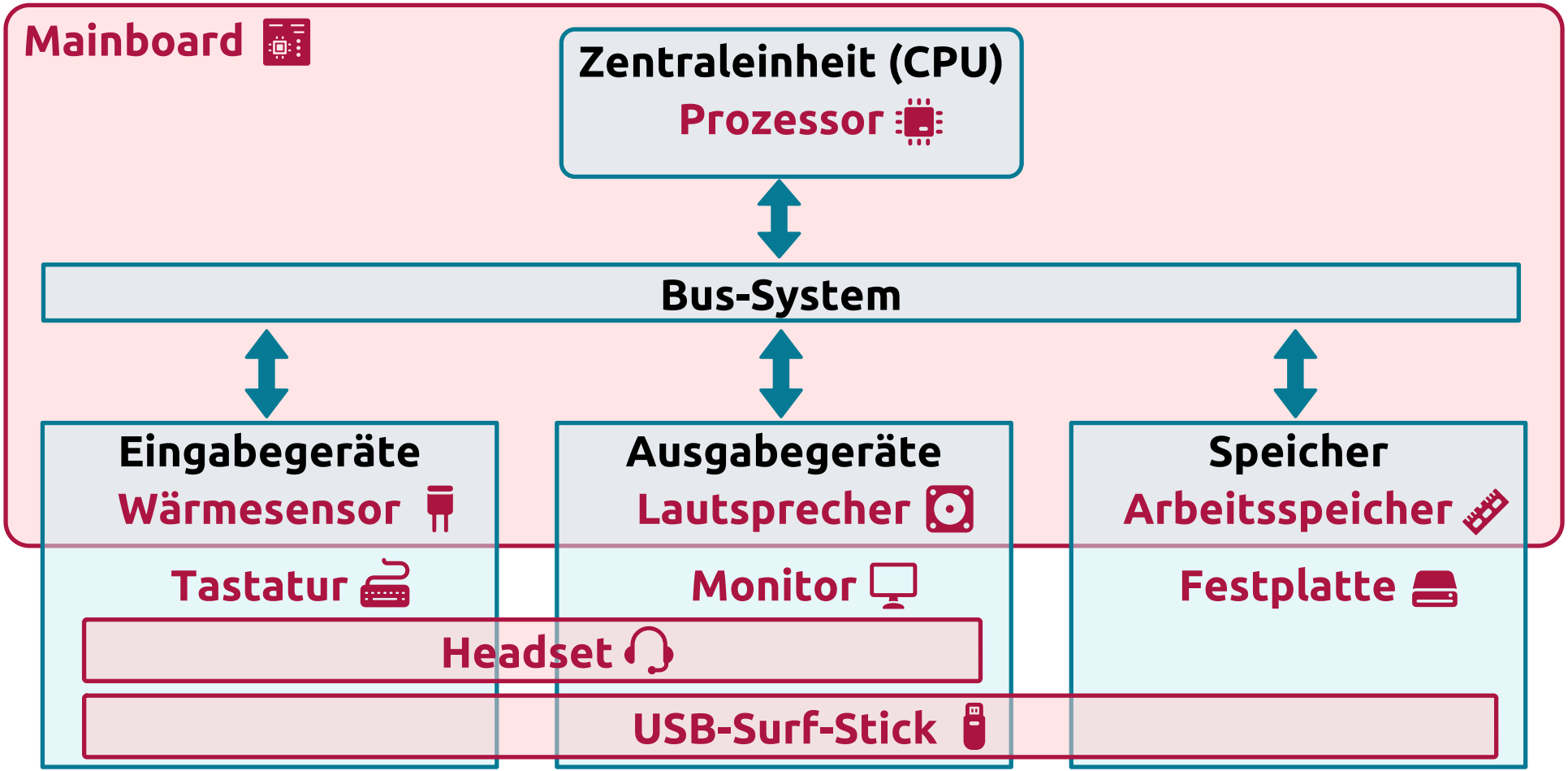
Physische Komponenten eines Computers



Physische Komponenten eines Computers



Physische Komponenten eines Computers



Geschwindigkeit von Computern / Rechenleistung

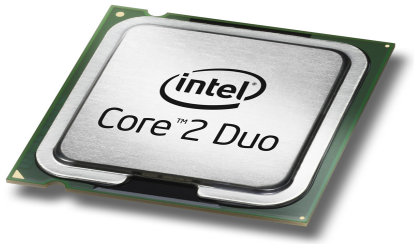
- wird typischerweise angegeben in „Maschinenbefehlen pro Sekunde“
- pro Sekunde = Hertz (abgekürzt Hz) = Einheit für die Frequenz
- Takt = Maschinenbefehlsauslöser
- Taktfrequenz = Maschinenbefehlsauslöser pro Sekunde (angegeben in Hertz)
  - 1.000 = kilo (k)
  - 1.000.000 = Mega (M)
  - 1.000.000.000 = Giga (G)
- Moderne Prozessoren können pro Takt mehrere Maschinenbefehle ausführen.
- Die aktuellen Intel-Prozessoren haben Taktfrequenzen von bis zu 5 GHz, also 5.000.000.000 Takte pro Sekunde und können ca. 100 Maschinenbefehle pro Takt ausführen, also ca. 500.000.000.000 Maschinenbefehle pro Sekunde. Zudem haben aktuelle Prozessoren meistens mehrere Kerne, so dass die Befehle mehrfach parallel ausgeführt werden können.

Speicher-Typen

- **ROM vs. RAM**
  - **ROM** = Read-Only Memory
    - kann gar nicht oder nur unter bestimmten Umständen verändert werden – zum Lesen von Daten
    - z. B. zum Speichern von Firmware verwendet, CD-ROM, …
  - **RAM** = Random-Access Memory
    - darauf kann beliebig zugegriffen werden
    - Daten lesen und speichern
    - wird oft auch als Bezeichnung für Arbeitsspeicher genutzt
- **persistent vs. nichtpersistent**
  - **persistent**: Daten werden „permanent“ gespeichert – also so lange, bis sie explizit gelöscht werden (falls möglich)
  - **nichtpersistent**: Daten leben nur vorübergehend im Speicher – z. B. bis sie durch andere Daten ersetzt werden oder nur so lange der Speicher mit Strom versorgt wird

Speicher-Zugriffszeiten und -datengrößen

	Zugriffszeit in Taktzyklen (ca.)	Größe in Bytes (ca.)
Prozessorregister	1	1.500
Arbeitsspeicher	100	16.000.000.000
Solid-State-Drive (SSD)	100.000	8.000.000.000.000
Festplatte (HDD)	1.000.000	20.000.000.000.000





Software-Typen: Systemsoftware

- anwendungsunabhängige Software, die das Ausführen von weiterer Software ermöglicht
- **Betriebssystem**
  - steuert die grundlegende Kommunikation zwischen Hardware und weiterer Software
  - z. B. Microsoft Windows, Linux, Unix, macOS, Android, iOS
- **Treiber**
  - z. B. Druckertreiber, Grafikkartentreiber
- **Laufzeitumgebungen**
  - z. B. Java-Runtime, C-Runtime, .NET-Runtime

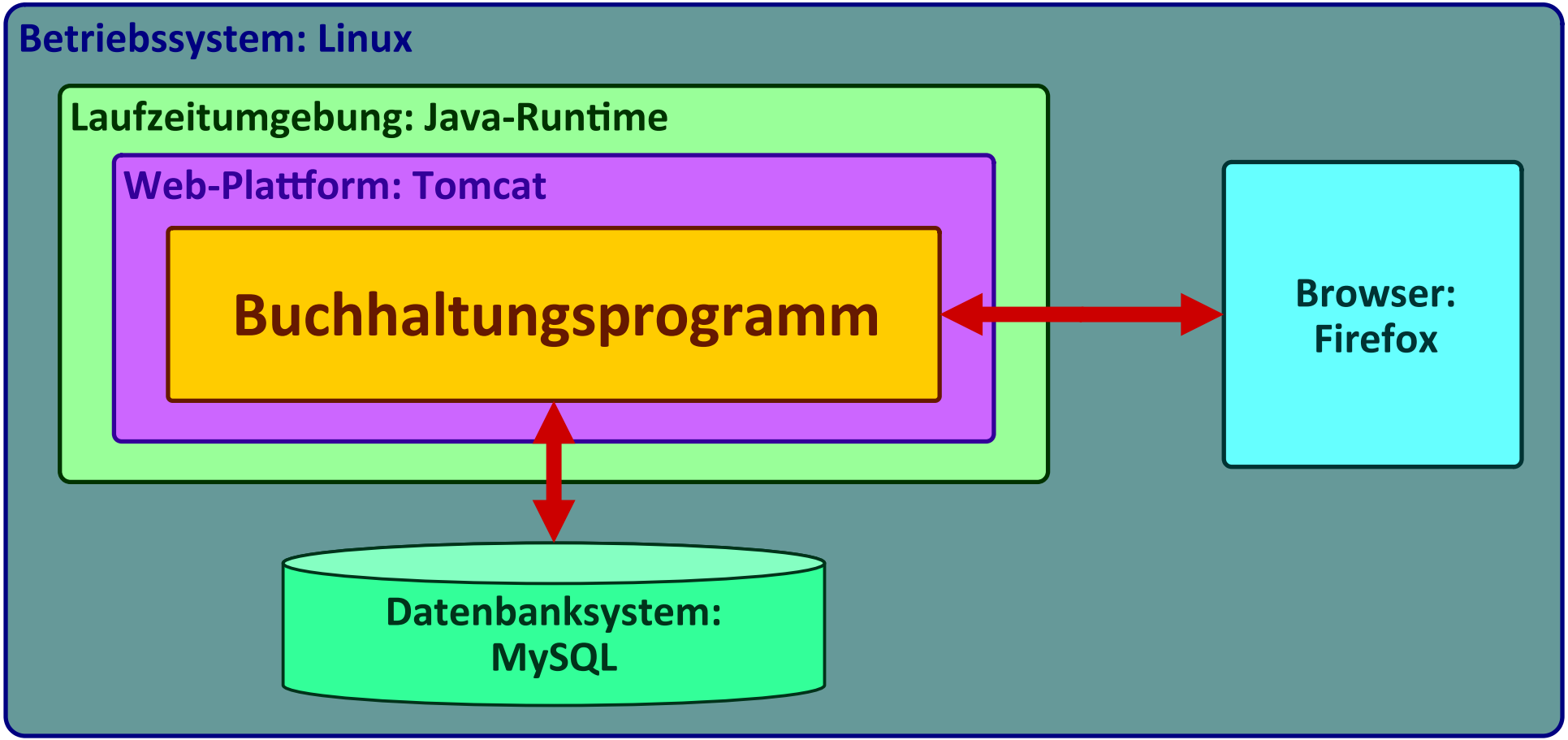
Software-Typen: Unterstützungssoftware

- Software, die bei der Entwicklung und/oder Wartung helfen oder eine sonstige nicht-anwendungsspezifische Leistung erbringen
- einige Subtypen
  - Entwicklungsumgebungen / Editoren (Eclipse, Notepad++, ...)
  - Datenbanksysteme (Oracle, MySQL, MS SQL, …)
  - Compiler/Interpreter (gcc, Visual C, javac, PHP, …)
  - Virenscanner (Avira, Kaspersky, avast, …)
  - Web-Plattformen (Apache Webserver, IIS, Tomcat, ...)
- eigentlich auch eine Gruppe von Systemsoftware, wird aber nach ISO/IEC 2382 als spezieller Typ definiert

Software-Typen: Anwendungssoftware

- Software, die den/der Benutzer:in bei der Ausführung der Aufgaben unterstützt und dadurch erst den eigentlichen, unmittelbaren Nutzen erbringt
- Anwendungssoftware wird von *Systemsoftware* ausgeführt (Betriebssystem, Treiber, Laufzeitumgebungen) und benutzt typischerweise *Unterstützungssoftware* (Datenbanken, Webserver)
- Beispiele
  - Microsoft Word, VLC, LibreOffice, WhatsApp, ...

Beispiel-Software: Buchhaltungsprogramm



Programmiersprachen

- einfach zu verstehende Anweisungen, die in Maschinensprache übersetzt werden müssen, bevor sie vom Computer ausgeführt werden
- Sprachen, deren Syntax und Semantik genau festgelegt ist
- **Syntax**
  - Definition aller zulässigen Wörter / Ausdrücke, die in einer Sprache formuliert werden können
- **Semantik**
  - Bedeutung der zulässigen Wörter / Ausdrücke
  - syntaktisch falsche Wörter / Ausdrücke haben keine Semantik
- **Algorithmus**
  - eine eindeutige, ausführbare Folge von Anweisungen endlicher Länge zur Lösung eines Problems

Programmierparadigmen

- **Imperative Programmierparadigmen**
  - eine Folge von Befehlen, die vorgeben wie, also in welcher Reihenfolge, die Befehle vom Computer ausgeführt werden sollen
  - strukturierte, prozedurale, modulare Programmierung
- **Deklarative Programmierparadigmen**
  - nicht das Wie, sondern das Was steht im Mittelpunkt – sehr mathematisch
  - funktionale, logische, mathematische Programmierung
- **Objektorientierte Programmierparadigmen**
  - Objekte werden in Klassen unterteilt, für die spezielle Funktionen gelten
  - Wiederverwendbarkeit und Datenkapselung
  - Weiterentwicklung der imperativen Programmierung
- **Service-orientierte Programmierparadigmen**
  - Dienste (Services) in einem Netzwerk liefern Teile der Implementierung

Programmierparadigmen

- Es gibt Programmiersprachen, die genau für die Programmierung in einem speziellen Programmierparadigma vorgesehen sind.
- Einige moderne Programmiersprachen (z. B. C++, PHP, Rust) eignen sich für alle Programmierparadigmen.
- Die Art der Programmierung entscheidet, wie viel Kontrolle der eigene Programmcode über die Ausführung des Programms im Computer hat.

		Kontrolle über Programmausführung
imperativ	Das Programm gibt genau vor, wie eine Berechnung durchgeführt wird. Nur der Compiler und die spezielle Rechnerarchitektur können noch beeinflussen, wie das Programm genau ablaufen wird.	**hoch**
deklarativ	Das Programm gibt im Detail vor, was berechnet werden soll, aber überlässt einem Teil der genauen Umsetzung den Internas der Programmiersprache.	
Service-orientiert	Der aufgerufene Service verbirgt komplett die Details, wie das Programm ausgeführt wird. Das Service-aufrufende Programm verlässt sich auf Zusicherungen vom Service, was als Ergebnis geliefert wird.	**niedrig**

Compiler vs. Interpreter

- **Compiler (Ahead-of-time-Compiler)**
  - übersetzt ein komplettes Programmiersprachenprogramm in ein systemspezifisches Maschinensprachenprogramm oder in systemunabhängigen Byte-Code
  - z. B. C, C++, Objective C, Pascal, Rust
- **Interpreter**
  - übersetzt einzelne Anweisungen eines Programmiersprachen-programms zu einzelnen systemspezifischen Anweisungen in Maschinensprache, während das Programm ausgeführt wird
  - z. B. PHP, Perl, Python, Basic
- **Just-in-time-Compiler**
  - übersetzt während der Laufzeit eines Programms Teile eines Programmiersprachenprogramms oder eines Byte-Code-Programms in Teile eines systemspezifischen Maschinensprachenprogramms
  - z. B. Java

Compiler vs. Interpreter: Fehlermeldungen

- **Compiler**
  - Syntaxfehler werden beim Compilieren gemeldet
  - Laufzeitfehler werden beim Ausführen gemeldet
  - Syntaxfehler führen dazu, dass das Programm nicht in Maschinensprache übersetzt werden kann
- **Interpreter**
  - Syntax- und Laufzeitfehler werden beim Ausführen gemeldet
  - Syntaxfehler bringen das Programm zur Laufzeit zum Abbruch

Compiler-Übersetzung

ein Compiler übersetzt die Anweisungen von einer höheren Programmiersprache in die Assembler-Anweisungen für die entsprechende Ziel-Plattform

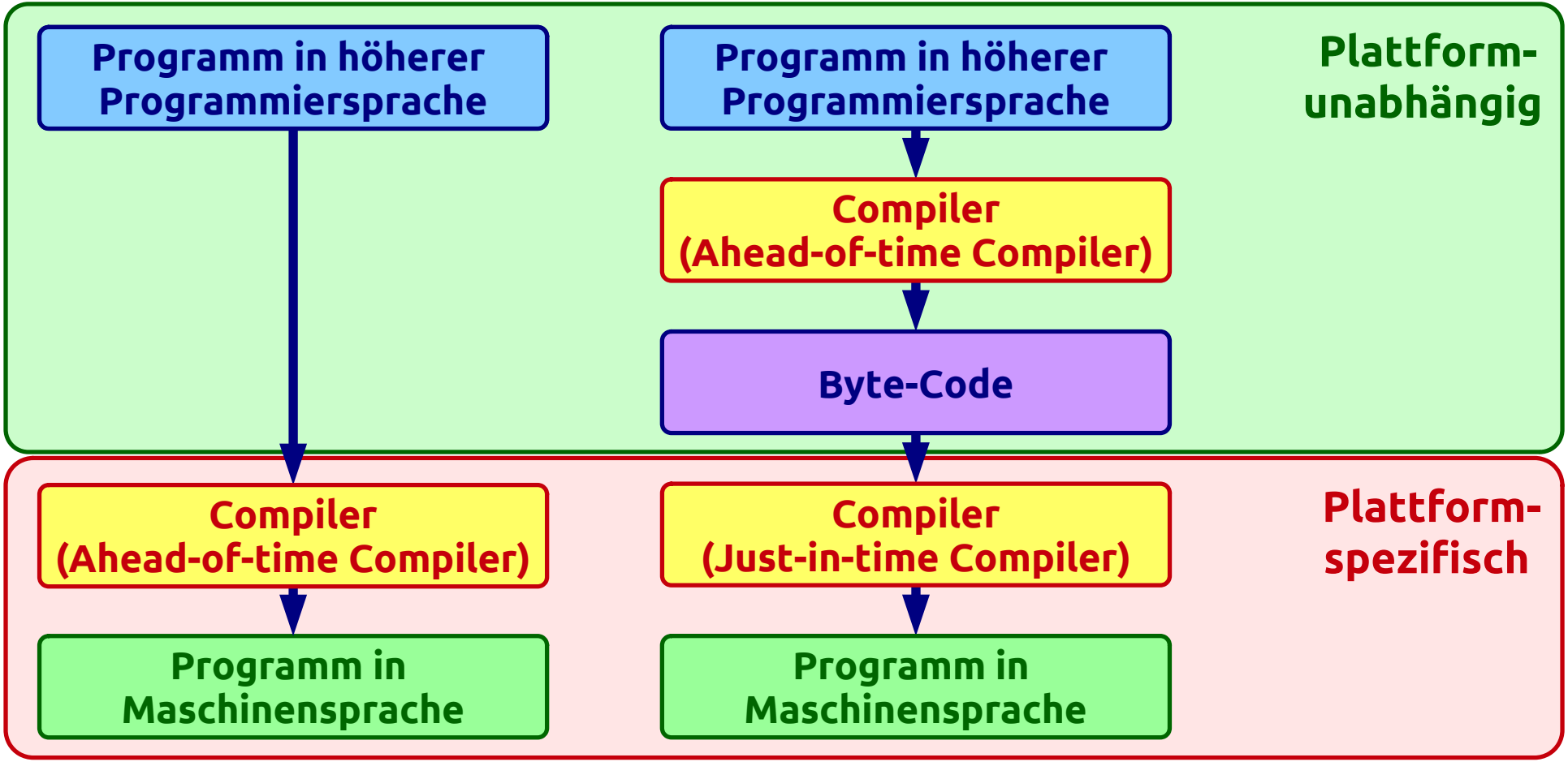
C-Programm:

```
main() {
  unsigned int a = 3;
  unsigned int b = 5;
  return a + b;
}
```

Assembler:

```
main:
pushq %rbp
movq %rsp, %rbp
movl $3, -8(%rbp)
movl $5, -4(%rbp)
movl -4(%rbp), %eax
movl -8(%rbp), %edx
addl %edx, %eax
popq %rbp
ret
```

Compiler-Übersetzung



Softwaresystem

- ein System aus mehreren Softwarekomponenten, die zusammengehören oder zusammenspielen
- z. B. das Buchhaltungs-Softwaresystem besteht aus
  - **Systemsoftwarekomponenten**
    - Betriebssystem: Linux
    - Laufzeitumgebung: Java-Runtime
  - **Unterstützungssoftwarekomponenten**
    - Web-Plattform: Tomcat
    - Datenbanksystem: MySQL
  - **Anwendungssoftwarekomponenten**
    - Buchhaltungssoftware
    - Browser: Firefox *(kann in diesem Fall auch als Unterstützungssoftwarekomponente betrachtet werden)*

Softwarearchitektur

- beschreibt die Anordnung und das Zusammenspiel der verschiedenen Komponenten innerhalb eines Softwaresystems
- ein einfaches und grobes Beispiel ist: