TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Systems Architecture, Operating Systems Group

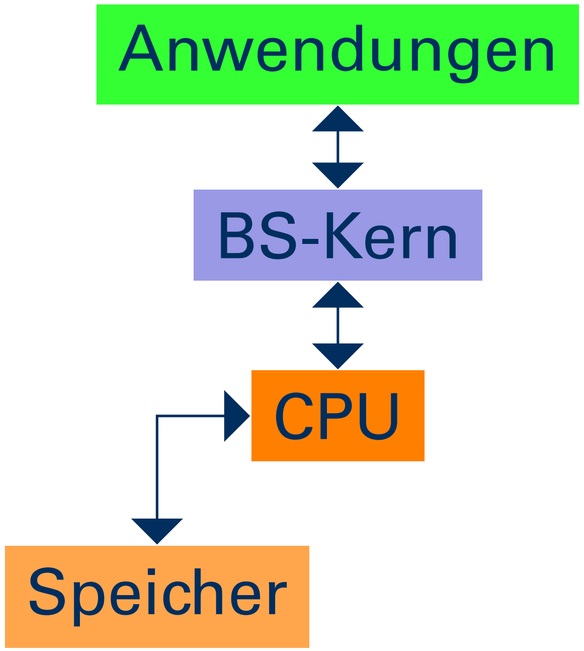
HARDWARE UND GERÄTETREIBER

MICHAEL ROITZSCH

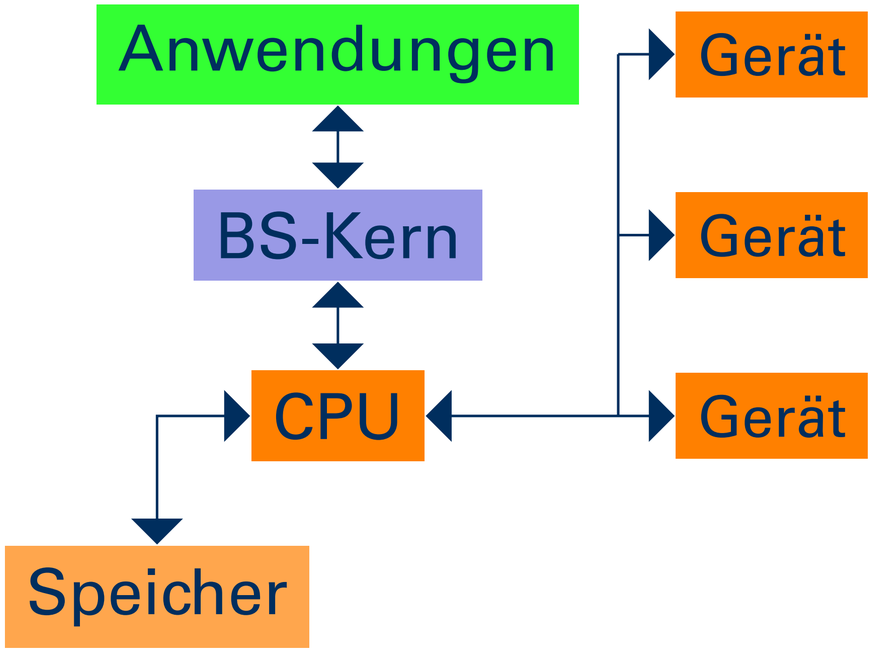
Übersicht

- Kommunikation zwischen Hardware und CPU
  - Interrupts
  - I/O-Ports
  - I/O-Speicher
  - Busse
- Verwaltung von Geräten
  - Dynamisches Hinzufügen/Entfernen
  - Anbindung an das Betriebssystem
- Warum ist das alles so schwer?

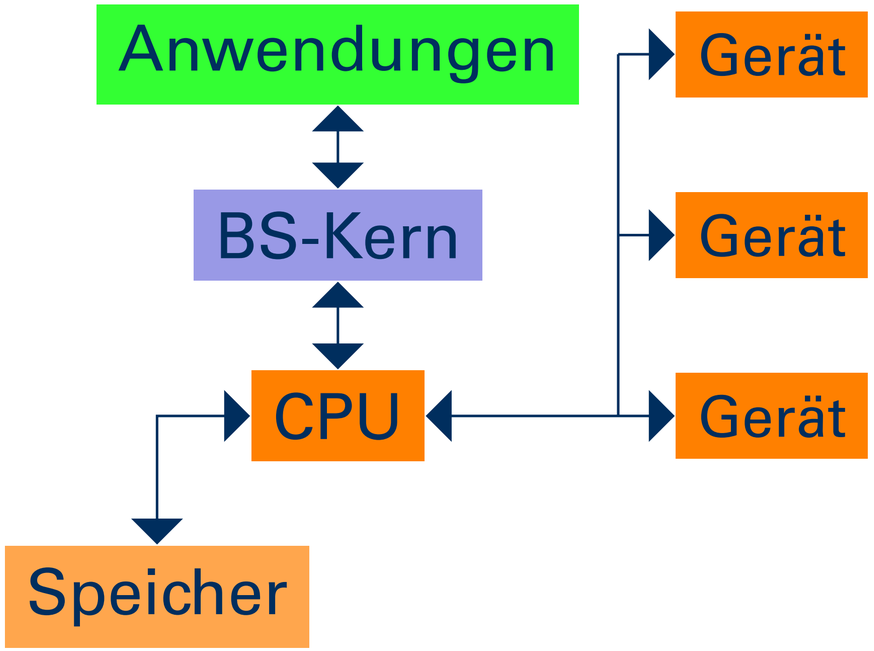
Bausteine



Bausteine



Bausteine

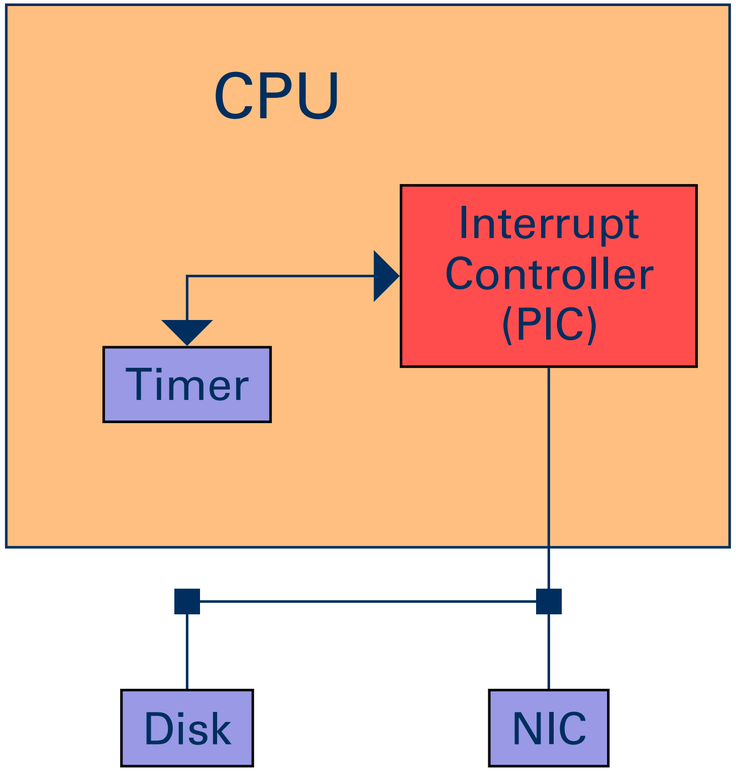


Beispiele für Geräte

- Eingabegeräte (*Human Interface Devices (HID)*)
- Netzwerkkarten (*Network Interface Cards (NIC)*)
- Grafik-Karten
- Speichermedien
- CPU-nahe Geräte (Timer, Interrupt-Controller)

Kommunikation zwischen Gerät und CPU

- Interrupt signalisiert Zustandsänderung des Geräts
- Interrupt-Controller (PIC):
  - Umwandlung von HW-Interrupts in CPU Exceptions
  - Priorisierung, Kombination von Interrupts
  - Maskieren von Interrupts
  - SMP: senden/empfangen von Inter-Prozessor-Interrupts (IPI)



I/O-Ports

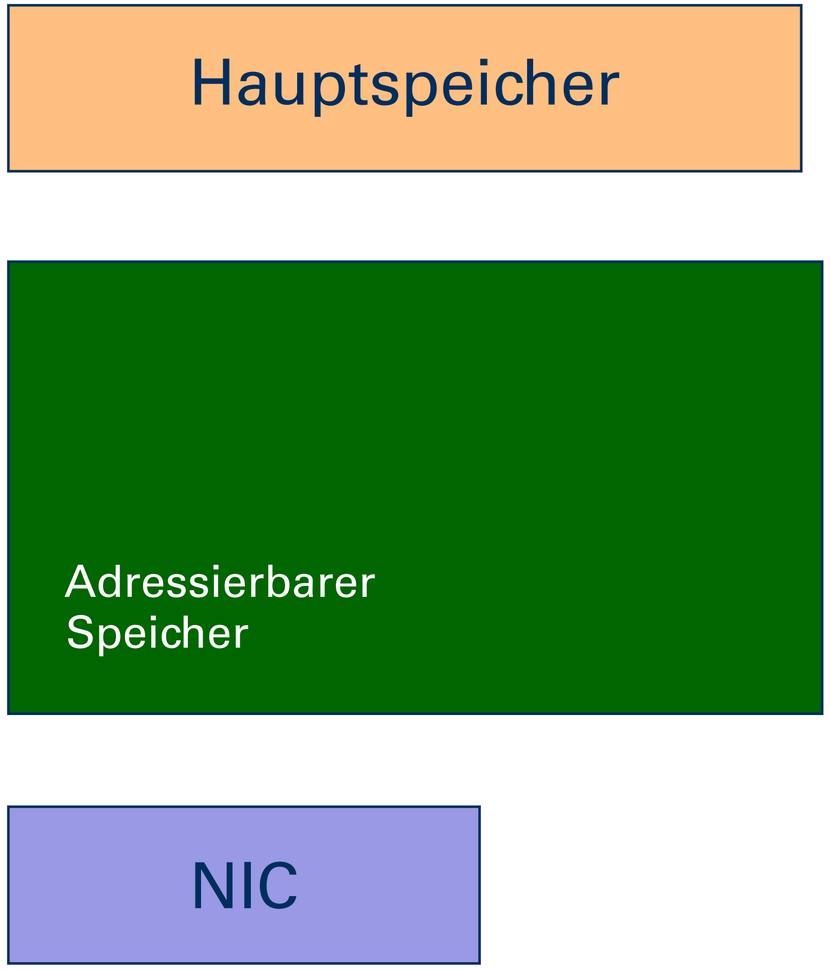
- Alternative Möglichkeit zum Zugriff auf x86-Gerätespeicher („Historisch gewachsen“ ™)
- Spezieller Adressbereich: einmalig 65536 I/O-Port-Bytes
- Nicht Teil des physischen Speichers
- Zugriff mittels spezieller Instruktionen

  `inb, inw, in`

  `outb, outw, out`
- Durchreichen an Anwendungen möglich (IO Privilege Level, IO Bitmap)

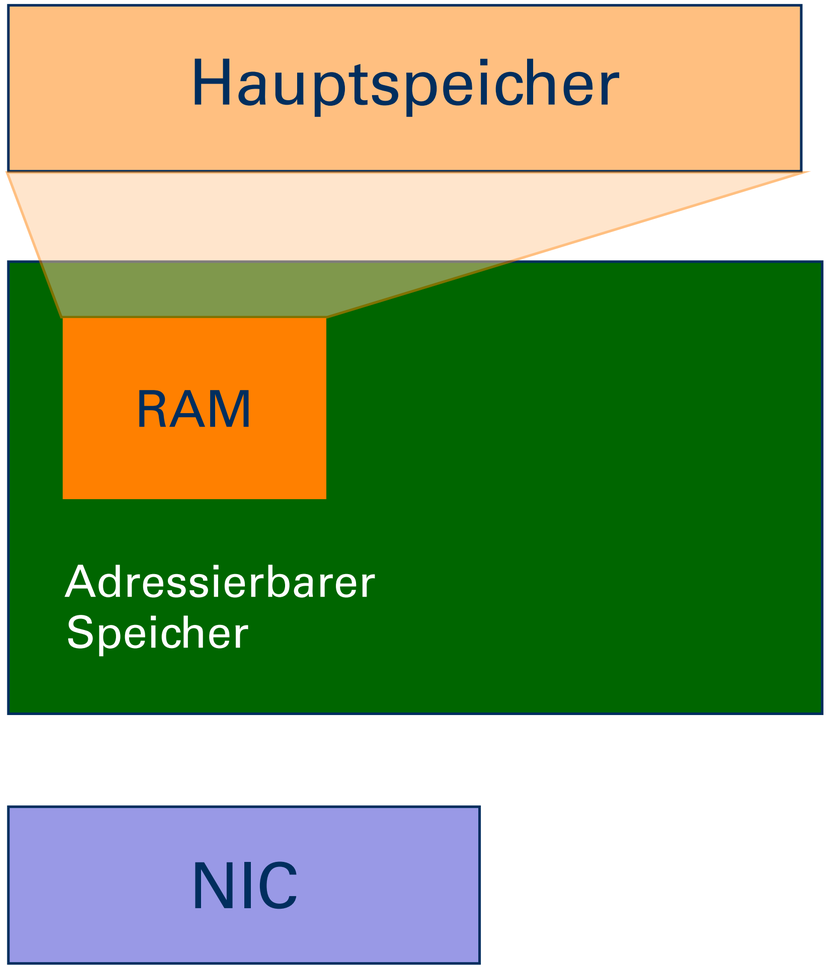
I/O-Speicher

- Komplexere Geräte: eigener Speicher
- Von CPU als Teil des physischen Speichers zugegriffen
- Aufteilung des physischen Speichers plattformabhängig
- Aufgabe des BIOS während des Bootvorgangs



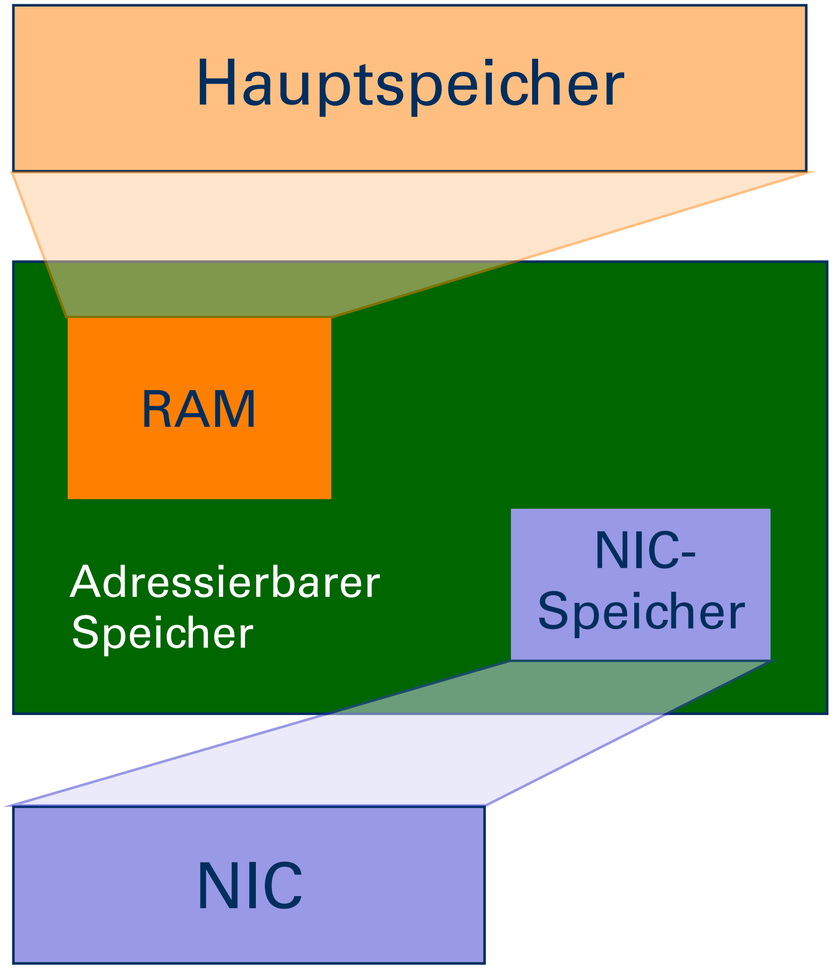
I/O-Speicher

- Komplexere Geräte: eigener Speicher
- Von CPU als Teil des physischen Speichers zugegriffen
- Aufteilung des physischen Speichers plattformabhängig
- Aufgabe des BIOS während des Bootvorgangs



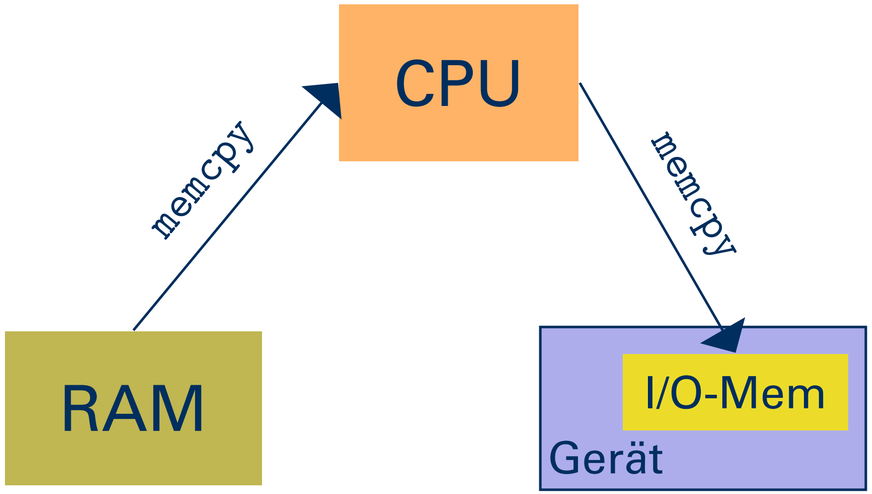
I/O-Speicher

- Komplexere Geräte: eigener Speicher
- Von CPU als Teil des physischen Speichers zugegriffen
- Aufteilung des physischen Speichers plattformabhängig
- Aufgabe des BIOS während des Bootvorgangs



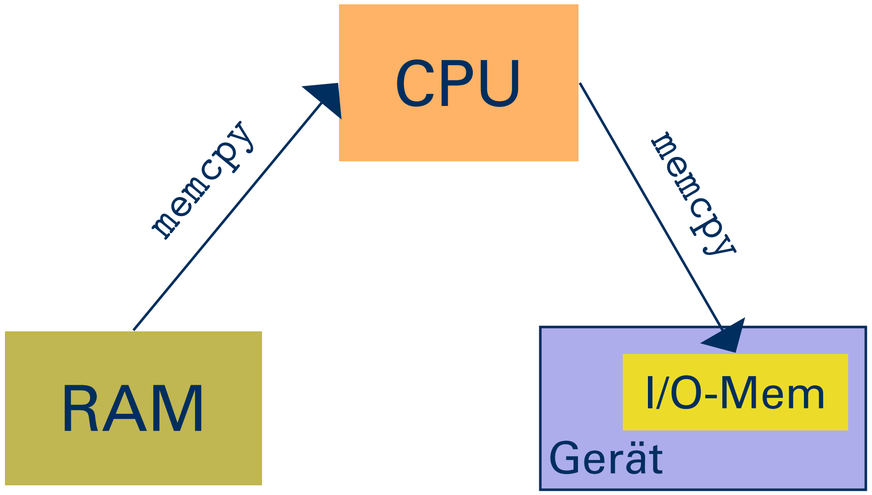
Direct Memory Access

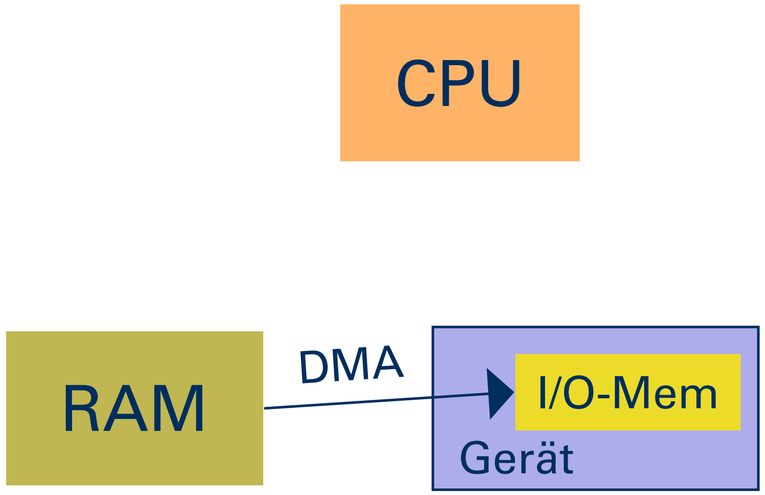
- I/O-Ports: Zugriff auf maximal 4 Byte gleichzeitig
- I/O-Speicher: kann `mempcy()`-Instruktionen (z. B. `rep movs`, SSE-Erweiterungen) benutzen
- Weitere Optimierung: *DMA*



Direct Memory Access

- I/O-Ports: Zugriff auf maximal 4 Byte gleichzeitig
- I/O-Speicher: kann `mempcy()`-Instruktionen (z. B. `rep movs`, SSE-Erweiterungen) benutzen
- Weitere Optimierung: *DMA*



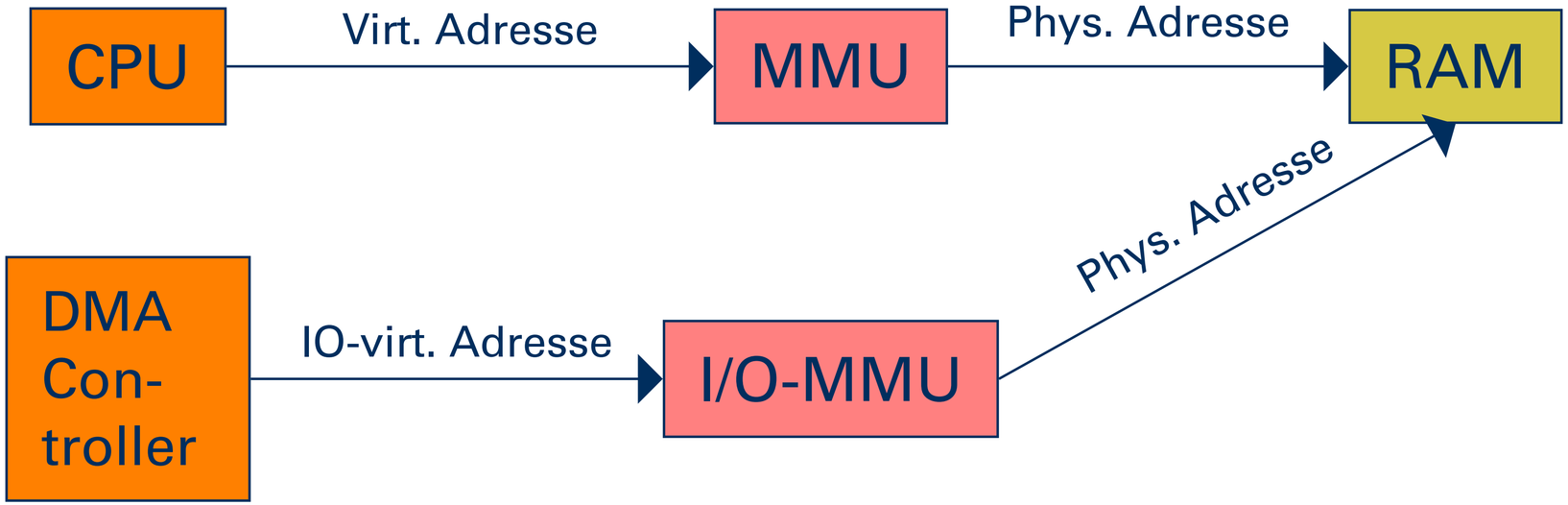


DMA: Sicherheitsproblem

- Physische Adressierung
- Überschreiben beliebiger Speicher-Regionen
- Also: Überschreiben von Kerndaten möglich

DMA: Sicherheitsproblem

- Physische Adressierung
- Überschreiben beliebiger Speicher-Regionen
- Also: Überschreiben von Kerndaten möglich
- Lösung: *I/O-MMU*



Geräte-Zustand

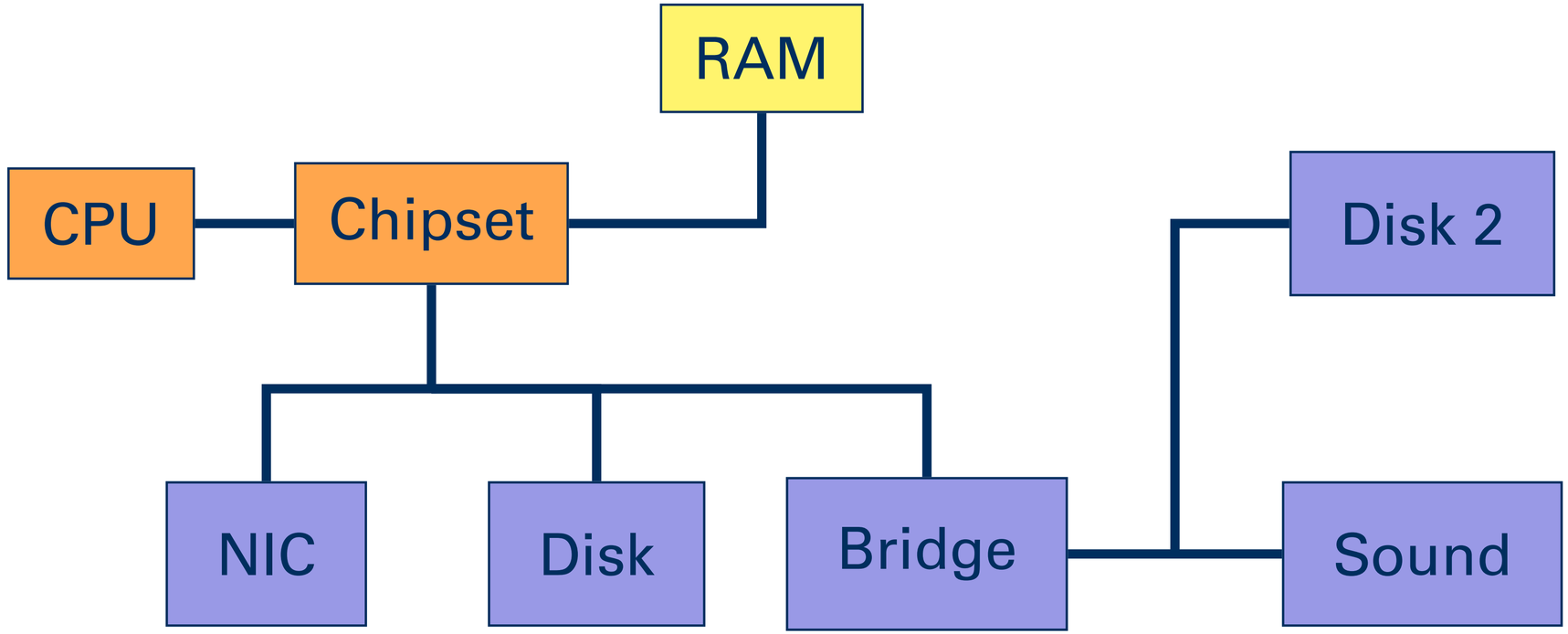
- BS liest und modifiziert Zustand via I/O-Speicher und I/O-Ports → Kernaufgabe eines *Gerätetreibers*
- Geräte-spezifische Aktionen
  - Welchen Zustand gibt es?
  - Wie wird er zugegriffen?
  - Wie führt man Geräte-Aktionen aus?
- Herkunft der Informationen variiert
  - Plattform-weite Definition (z. B. PC-Plattform)
  - Standard f. bestimmte Geräteklasse (z. B. SATA-Festplatten)
  - Geräte-Spezifikation (vom Hersteller)

Gerätetreiber

- Interrupt-Behandlung (Inspektion und Manipulation des Zustands)
- Kapselt geräte-spezifisches Wissen
- Integration in restliche BS-Infrastruktur

Wie findet man Geräte? Wie die zugehörigen Treiber?

Busse: Vernetzung von Geräten



Aufgaben: Identifikation, Adressierung, Weiterleiten von Kommandos/Interrupts

Geräte-Erkennung

- Während des Bootvorgangs
- BS „scannt“ die PCI-Busse
  - Lesen der Geräte-ID
  - Rückgabewert ! = $0x\mathit{FFFFFFFF}$ → Gerät vorhanden
- Findend es zugehörigen Treibers
  - Treiber liefert Liste der unterstützten Geräte-IDs
  - BS ruft `init()`-Funktion auf, wenn entsprechendes Gerät gefunden

Probleme bei Treiber-Entwicklung

- Entwicklung in „unsicherer“ Programmiersprache
  - C → Pointer-Arithmetik, Bit-Operationen etc.
- Kompliziertes Kern-Interface
- Komplizierte Hardware
  - Fehlende HW-Spezifikationen