

# Computer-Architektur

Gregor Sönnichsen

AG Games Engineering  
Universität Bayreuth

15. Dezember 2020

# Inhaltsverzeichnis

Einführung

1 Hardware

2 Programmierung

3 Betriebssystem

Ende

# Einführung

## Motivation



# Einführung

## Motivation



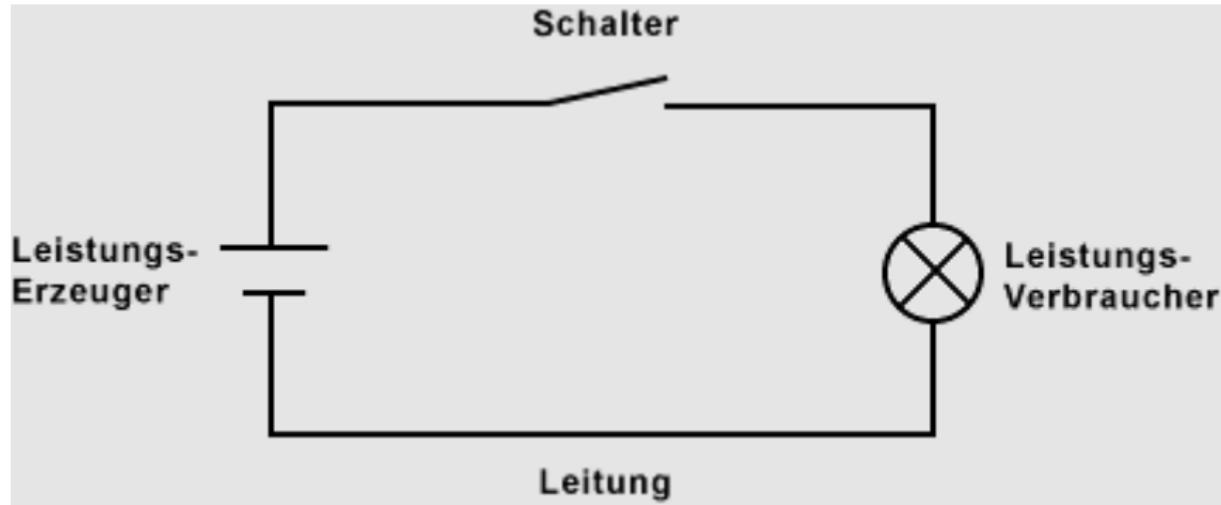
# Einführung

## Schichtenmodell des Rechneraufbaus



# 1 Hardware

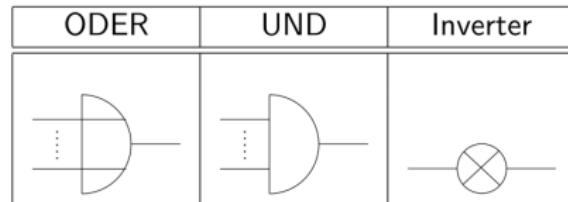
## Stromschaltungen



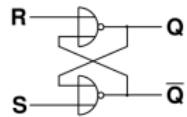
# 1 Hardware

## Konstruktion

Basisgatter:



Flip-Flop:



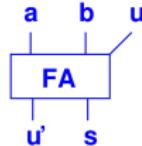
- für  $S = R = 0$  gibt es **zwei stabile Zustände**:
  - $Q = 0, \bar{Q} = 1$
  - $Q = 1, \bar{Q} = 0$

Addierer:

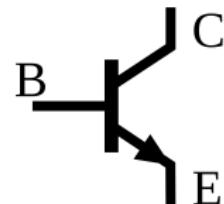
Volladdierer (full adder) (mit Eingabeüberlauf):

$$+ : \{0,1\}^3 \rightarrow \{0,1\}^2$$

$a_0$	$b_0$	$u$	$u'$	$s$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Transistor:



Weitere wichtige Schaltungen: Taktgenerator, (De-)Kodierer, Arithmetisch-Logische Einheit, Bus, Adressrechner, Druckplatten

# 1 Hardware

## Konstruktion

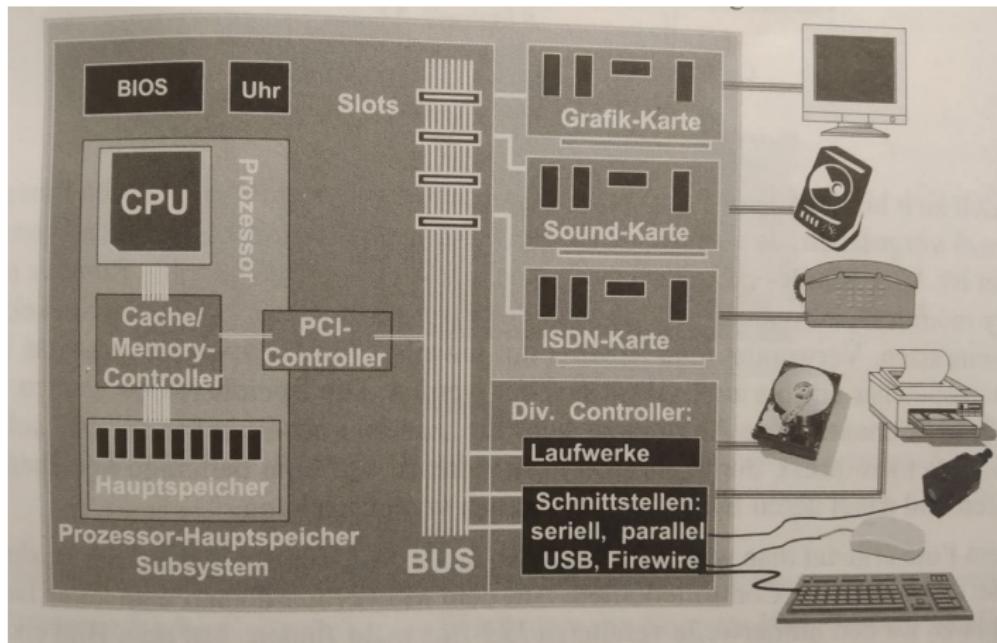
Computer basieren auf elementaren elektronischen, logischen Schaltungen. Es gibt spezielle Schaltungen, die

- ▶ das Speichern/Auslesen von Spannung/Nicht-Spannung bzw. 0/1 ermöglichen
- ▶ elementare binäre Rechenoperationen durchführen
- ▶ Taktung ermöglichen
- ▶ Ein-/Ausgaben ermöglichen

**Prozessor:** Eine Schaltung, die Verbindungen nach außen hat, mithilfe derer der Inhalt der inneren Register gesetzt werden kann. Die Inhalte gewisser Register werden als Befehle interpretiert, die der Prozessor dann ausführt.

# 1 Hardware

## Überblick über größere Komponenten



# 1 Hardware

## Trivia

- ▶ Computer-Schaltkreise heutzutage sind äußerst komplex
- ▶ Firmen wie Intel, AMD und theoretische Informatiker beschäftigen sich mit der computergestützten Generierung und Optimierung von Schaltkreisen



## 2 Programmierung

### Assemblerprogrammierebene

- ▶ Basis-Assemblerbefehle als Abbildung gewisser Registerkonfigurationen
  - ▶ Compiler als Übersetzer
  - ▶ OS überträgt Binärkode in Prozessorregister
- ⇒ Demonstration

# 2 Programmierung

## Ebene höherer Programmiersprachen

Abstraktionen gewisser Assemblerprogrammierpatterns:

- ▶ Arrays
- ▶ Variablen, Typen
- ▶ Loops
- ▶ Funktionen
- ▶ Verzweigungen
- ▶ Klassen
- ▶ Heap

```
1    li $t0, 20
2
3    # Initialize registers
4    li  $t1, 0          # initialize counter (i)
5    li  $t2, 0          # initialize sum
6
7    # Main loop body
8 loop:
9    addi $t1, $t1, 1    # 1) i = i + 1
10   add $t2, $t2, $t1   # 2) sum = sum + i
11   beq $t0, $t1, exit  # 3) if i = N, goto exit
12   j loop              # 4) goto loop
```



Compiler als Übersetzer

```
1  int t0 = 20;
2
3  int t2 = 0;
4
5
6  for (int t1 = 0; t1 < t0; t1++) {
7      t2 = t2 + t1;
8
9
10 }
```

### 3 Betriebssystem

#### Übersicht

**Betriebssystem:** Ein Programm, dass Verwaltungsaufgaben übernimmt und anderen Programmen einen einfachen Kontext zur Verfügung stellt

- ▶ Benutzeroberfläche
- ▶ Speicherverwaltung
- ▶ Dateiverwaltung
- ▶ Prozessverwaltung
- ▶ Ein-/Ausgabe
- ▶ Systemsicherheit

### 3 Betriebssystem

#### Prozessverwaltung

**Problem 1:** Moderne Programme passen nicht komplett in die Prozessorregister und werden üblicherweise auf einem persistenten Speichermedium abgelegt.

**Lösung:** Wenn der User ein Programm ausführt, dann kopiert das OS den Binärkode in den RAM und gibt dem Prozessor die Startadresse des Codes im RAM.

# 3 Betriebssystem

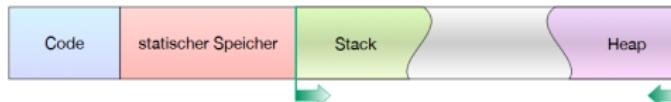
## Prozessverwaltung

**Problem 1:** Moderne Programme passen nicht komplett in die Prozessorregister und werden üblicherweise auf einem persistenten Speichermedium abgelegt.

**Lösung:** Wenn der User ein Programm ausführt, dann kopiert das OS den Binärkode in den RAM und gibt dem Prozessor die Startadresse des Codes im RAM.

**Problem 2:** Code braucht zur Laufzeit extra Speicher, es kann dem Code aber erstmal nicht bekannt sein, wo noch Platz frei ist im RAM. Zudem müssen irgendwo Metadaten abgelegt werden.

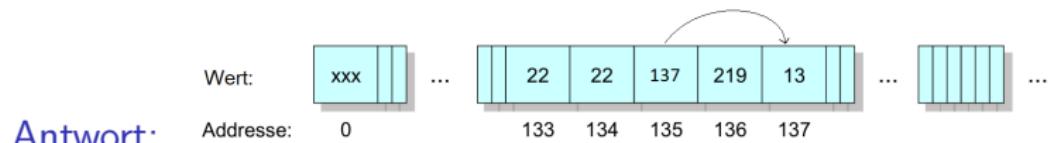
**Lösung:** *Prozesskontext* im RAM: Stack, Heap, Prozess-ID, Datei-Handles etc.



# 3 Betriebssystem

## Speicheradressierung und Ein-/Ausgabe

Frage 1: Was sind nun diese Pointer?

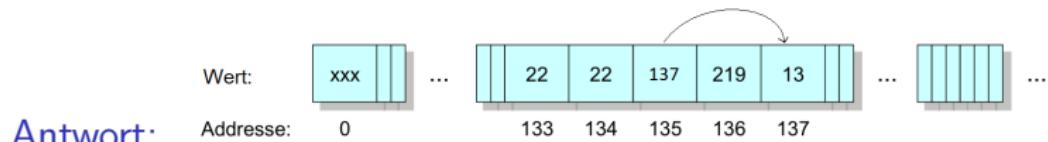


Antwort:

# 3 Betriebssystem

## Speicheradressierung und Ein-/Ausgabe

Frage 1: Was sind nun diese Pointer?

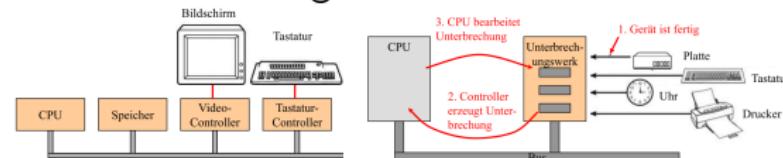


Antwort:

Frage 2: Wie kommt die Eingabe in den Prozessor und die Ausgabe an die Peripherie?

Antwort: spezielle Register, die von Peripherie angesteuert werden

Antwort: dedizierter Bereich im RAM, in dem diese Informationen ausgetauscht werden



# 3 Betriebssystem

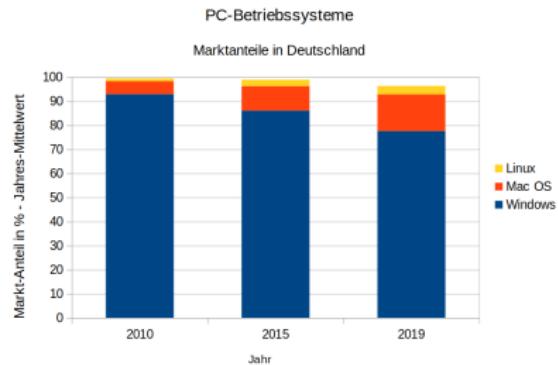
## Trivia

Betriebssysteme stellen ein Programmierinterface bereit, um auf die implementierten Dienste zuzugreifen:

- ▶ Unix und abgeleitete: POSIX
- ▶ Windows: WinApi

Betriebssysteme und ihre Derivate:

- ▶ Unix: Linux, MacOS, Android
- ▶ Windows



# Was sonst noch geschah

Geschichten      Design  
Computerspiele      *Die Informatik ist  
nicht genug*  
Soziologie  
Architektur      Psychologie  
**Philosophie**  
Der Welt etwas zurückgeben



# Was sonst noch geschah

Geschichten      Design  
Computerspiele      *Die Informatik ist  
nicht genug*  
Soziologie  
Architektur      Psychologie  
Philosophie  
Der Welt etwas zurückgeben



Ende