Grundlagen der Informatik

- Computer-Hardware -

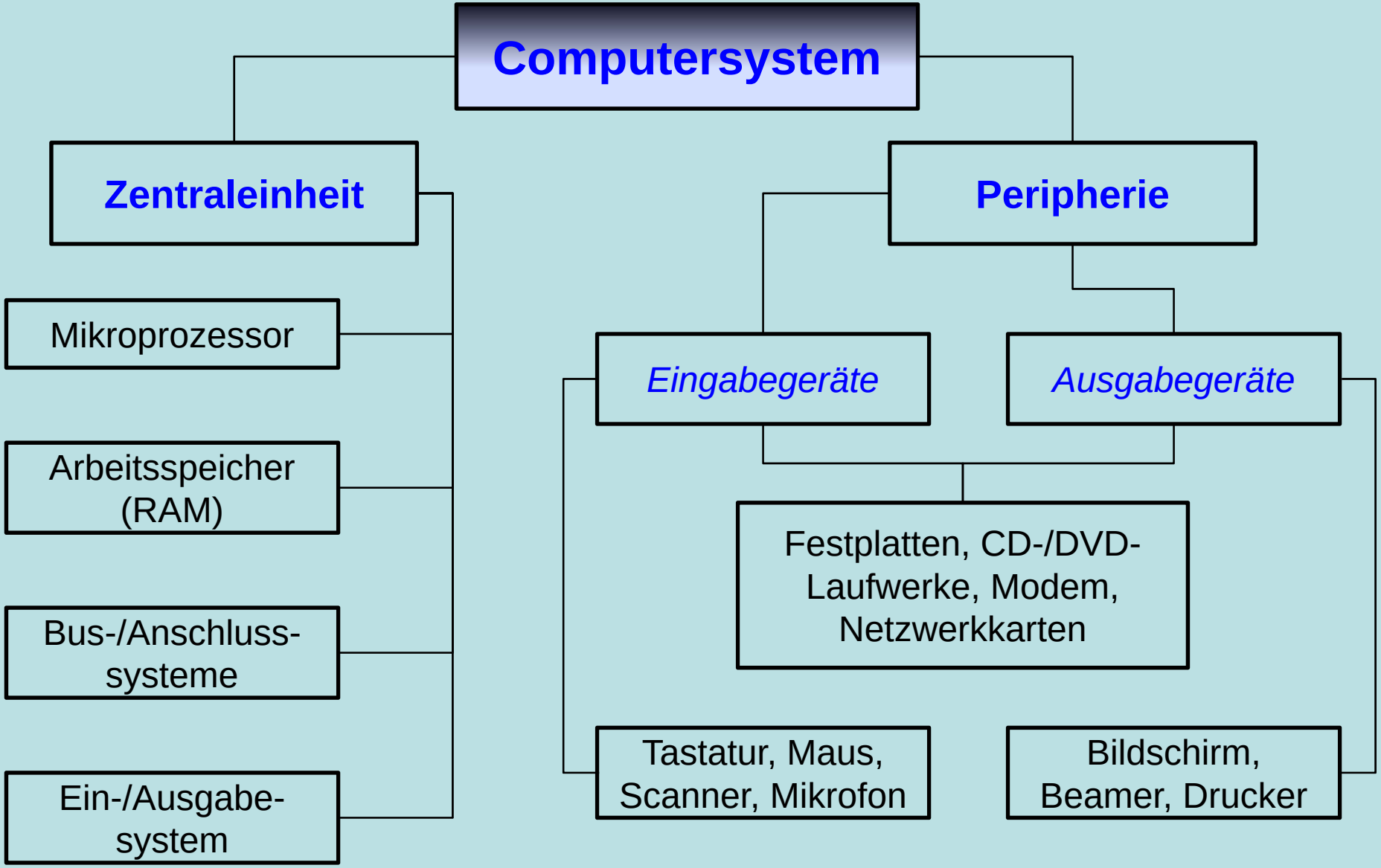
Prof. Dr. Klaus Volbert

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

Hochschule für angewandte Wissenschaften
Fakultät Informatik und Mathematik

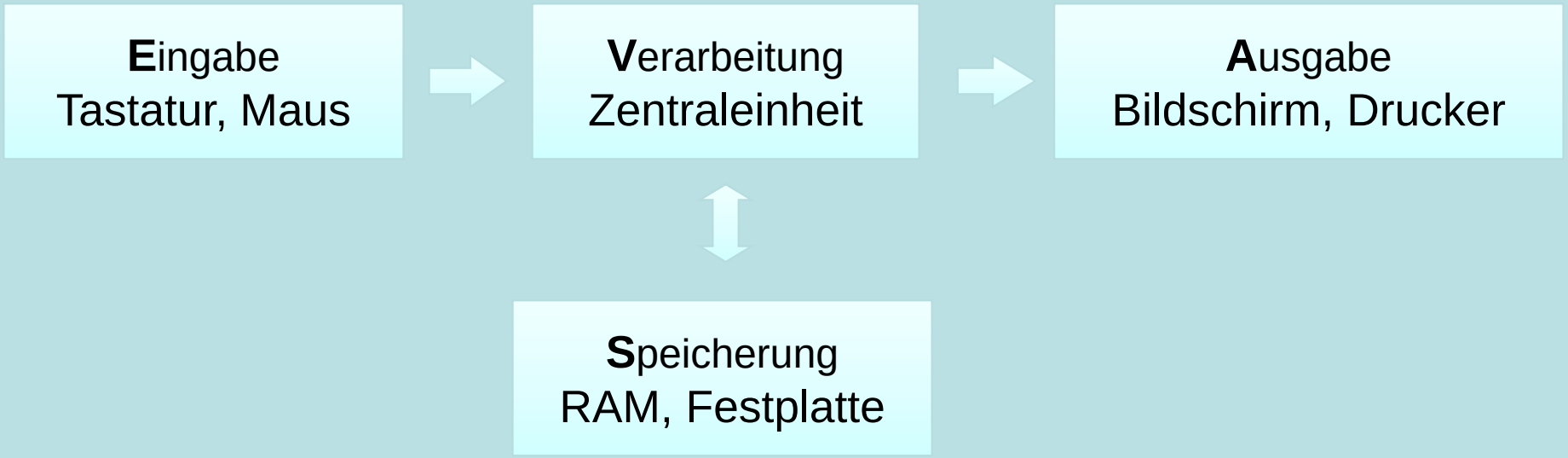
Wintersemester 2010/11
Regensburg, 26. Oktober 2010

Übersicht


Computersystem
Zentraleinheit
Peripherie
Mikroprozessor
Arbeitsspeicher (RAM)
Bus-/Anschluss-systeme
Ein-/Ausgabe-system
Eingabegeräte
Ausgabegeräte
Festplatten, CD-/DVD-Laufwerke, Modem, Netzwerkkarten
Tastatur, Maus, Scanner, Mikrofon
Bildschirm, Beamer, Drucker

EVA-Prinzip

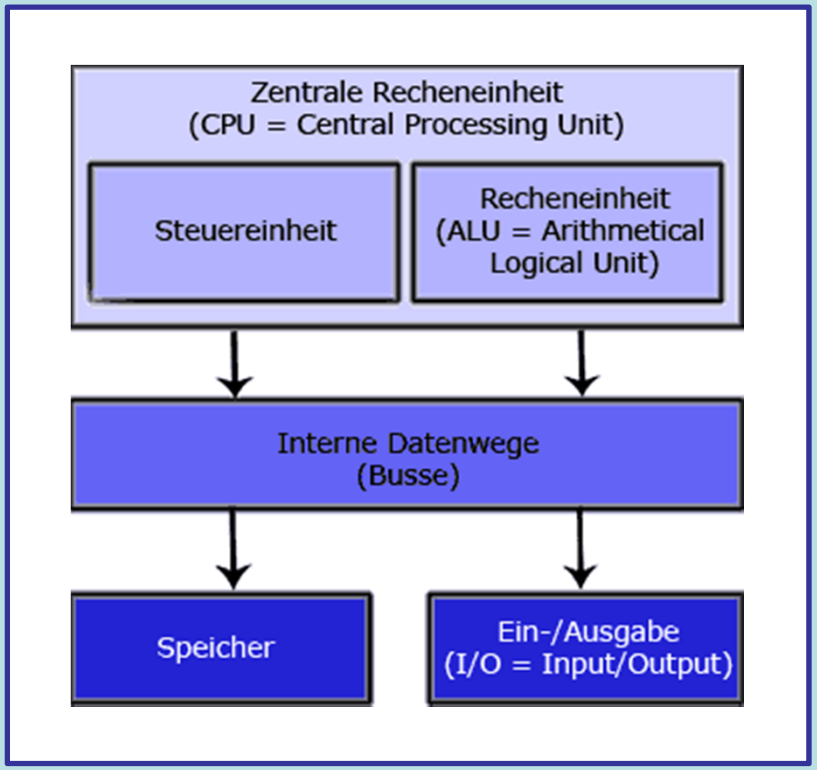
- Eingabe, Verarbeitung, Ausgabe



- Das EVA-Prinzip lässt sich sowohl durch die gesamte Geschichte der Computer als auch durch die verschiedenen Teilbereiche der Informatik verfolgen (z.B. Algorithmik)

Von-Neumann-Rechner

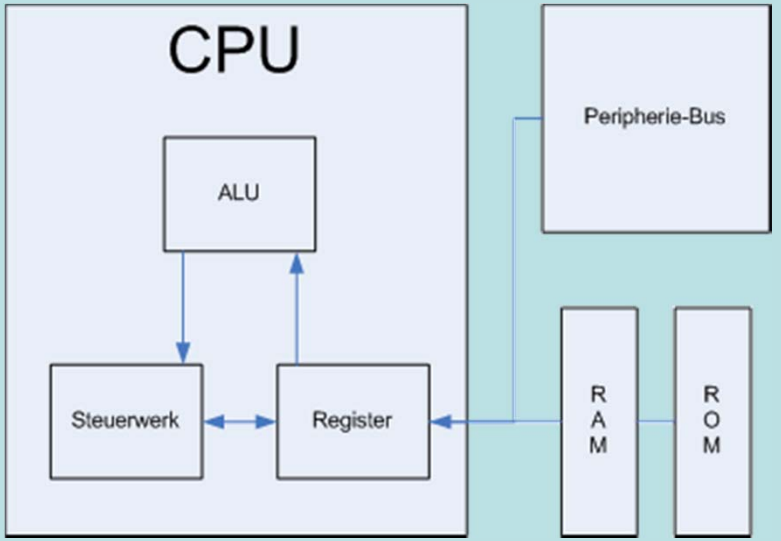
- Wichtige Merkmale
  - Programmgesteuerter Rechner mit
    - Zentraler Recheneinheit (CPU) mit ALU und Steuereinheit
    - Speicher
    - Ein- und Ausgabeeinheiten
    - Internen Datenwegen (Busse)
  - Verwendung des Binärsystems
  - Universalrechner
    - Struktur des Rechners ist unabhängig von dem zu bearbeitenden Problem (Eingabe von Programmen, Sprünge)
  - Programme und Daten liegen im gemeinsamen Speicher
    - Speicherplätze sind gleich breit (Maschinenwort) und werden adressiert
  - Ein Programm ist eine Abfolge von Maschinen-Befehlen
  - Befehle geben nur die Speicheradresse an, wo Daten abgelegt sind, und nicht die Daten selbst



Von-Neumann-Zyklus (Schematisch)

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Befehlsverarbeitung in einem Von-Neumann-Rechner:



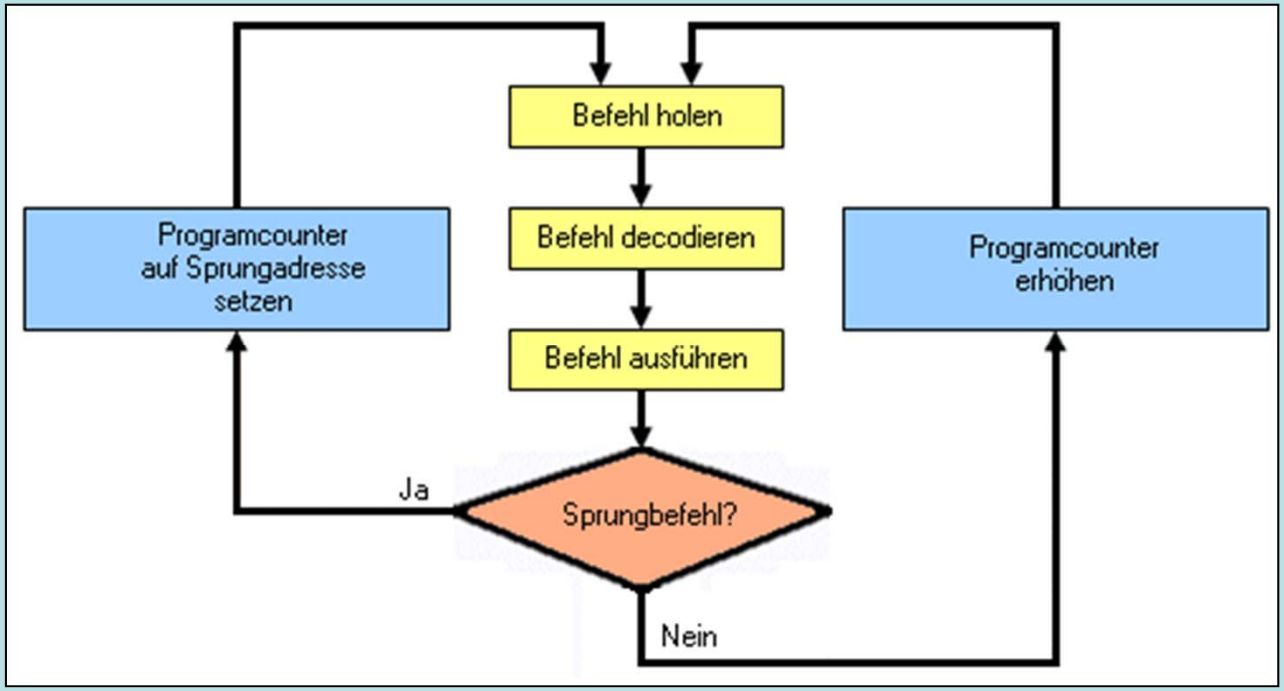
  1. FETCH (Befehl holen)
     Laden des akt. Befehls (via Befehlszähler) aus dem Speicher in das Befehlsregister
  2. DECODE (Befehl dekodieren)
     Dekodierung des Befehls im Steuerwerk
  3. FETCH OPERANDS (Operanden holen)
     Laden von Operanden aus dem Speicher in das Datenregister
  4. EXECUTE (Operation ausführen)
     Ausführen des Befehls im Rechenwerk
  5. UPDATE INSTR PTR (Befehlszähler aktualisieren)
     Befehlszähler modifizieren (Sprünge berücksichtigen)

Beispielabfolge:

```
mov   #5,&4711
add   &4711,R3
incd  R3
```

- Ein Teilschritt kann mehrere Takte dauern
- Moderne Prozessoren führen heutzutage in einer Sekunde mehrere Milliarden Takte aus

Von-Neumann-Zyklus (Ablaufdiagramm)



- Anmerkung Hardware-Interrupts:
  - Die Befehlssequenz kann ggf. durch äußere Ereignisse unterbrochen werden (z.B. Festplatte, Netzwerkkarte, Soundkarte, etc.)
  - Bei Unterbrechung („einem Interrupt“) werden zunächst die für die Unterbrechung definierten Befehle durchgeführt
  - Anschließend wird an der unterbrochenen Stelle im Programm wieder aufgesetzt

Nachteile des Von-Neumann-Rechners

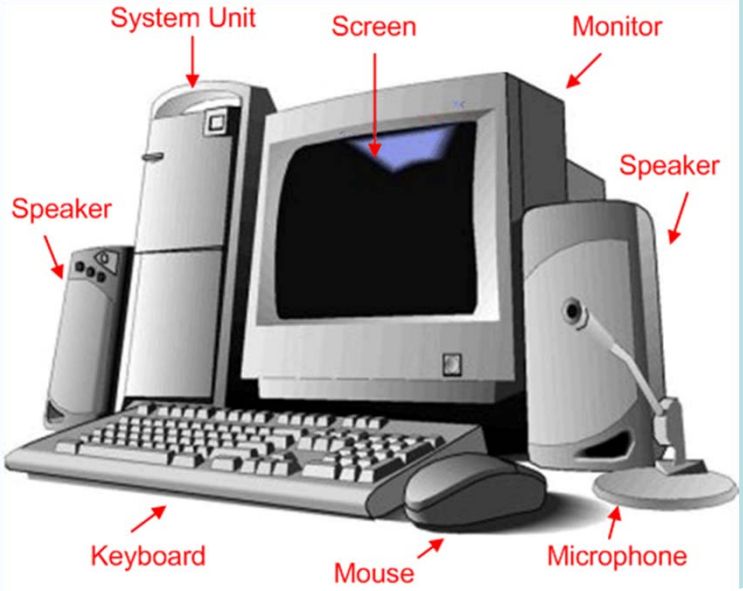
- Im Wesentlichen hängen die Nachteile mit der Speicherstruktur zusammen:
  - Befehle und Daten können nicht anhand des Bitmusters unterschieden werden
  - Variable und konstante Daten können nicht unterschieden werden
  - Falsche Adressierung kann dazu führen, dass Speicherinhalte geändert werden, die nicht geändert werden dürfen (Befehle/Konstanten)

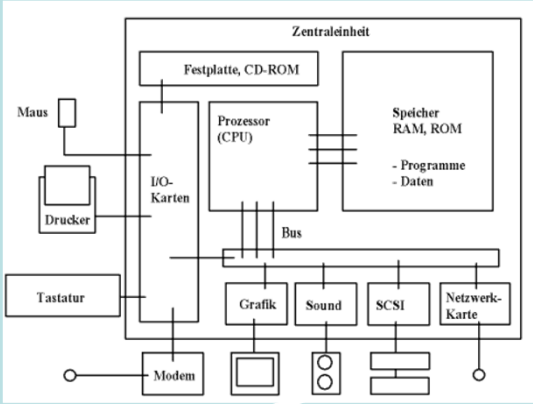
    → erfordert Mechanismus zum expliziten Speicherschutz

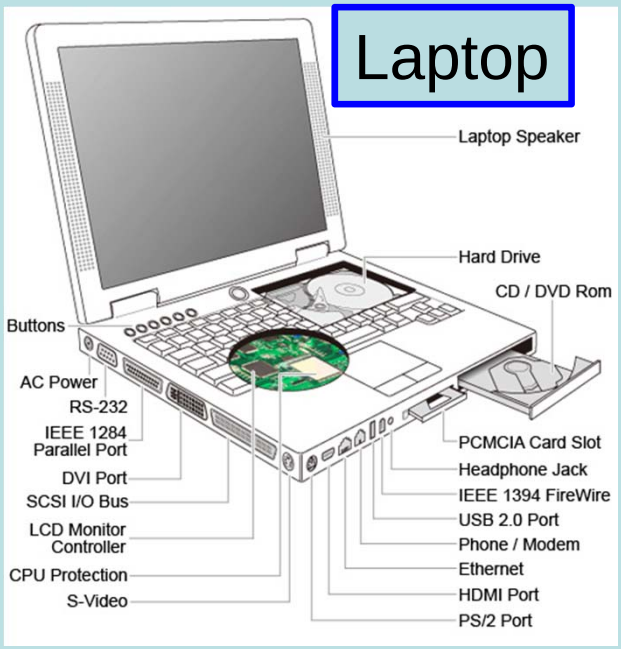
- Von-Neumann-Flaschenhals
  - Da Daten und Befehle im gleichen Speicher liegen, kommt es beim Zugriff auf den Speicher über den Systembus zu einem Flaschenhals (engl. bottleneck)

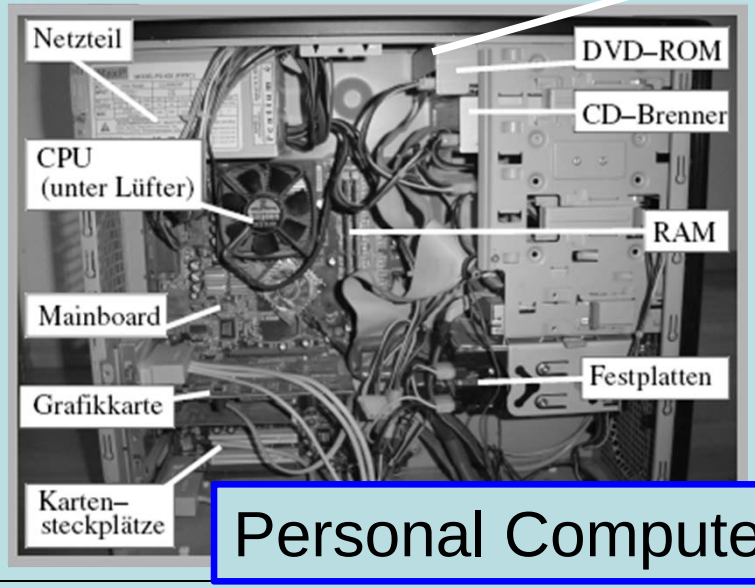
    → Moderne Rechner schwächen das Problem durch einen Cache ab

Heutige Computer (Ansichten)

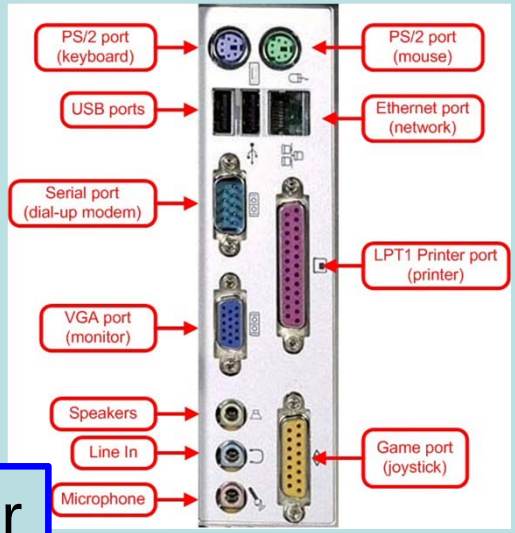


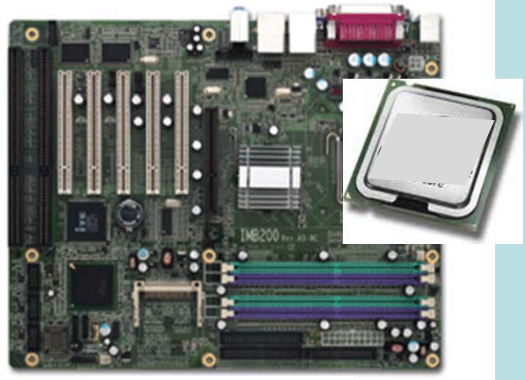






Personal Computer





Mainboard

Beispiel: Dualansicht (C++, Maschine) I

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
	int a=5;

	cout << "Hallo Welt!" << endl;
	cout << "Der Wert für a ist: " << a << endl;

	return 0;
}
```

Beispiel: Dualansicht (C++, Maschine) II

```
#include "stdafx.h"
#include <iostream>

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
01291490  push        ebp
01291491  mov         ebp,esp
01291493  sub         esp,0CCh
01291499  push        ebx
0129149A  push        esi
0129149B  push        edi
0129149C  lea         edi,[ebp-0CCh]
012914A2  mov         ecx,33h
012914A7  mov         eax,0CCCCCCCCh
012914AC  rep stos    dword ptr es:[edi]
        int a=5;
012914AE  mov         dword ptr [a],5
```

Hello World in Assembler (Intel 80x86)

Beispiel: Dualansicht (C++, Maschine) III

```
        cout << "Hallo Welt!" << endl;
012914B5  mov         esi,esp
012914B7  mov         eax,dword ptr [__imp_std::endl (129A314h)]
012914BC  push        eax
012914BD  push        offset string "Hallo Welt!" (129784Ch)
012914C2  mov         ecx,dword ptr [__imp_std::cout (129A318h)]
012914C8  push        ecx
012914C9  call        std::operator<<<std::char_traits<char> >
                      (129114Fh)
012914CE  add         esp,8
012914D1  mov         ecx,eax
012914D3  call        dword ptr
[__imp_std::basic_ostream<char,std::char_traits<char> >::operator<<
(129A31Ch)]
012914D9  cmp         esi,esp
012914DB  call        @ILT+405(__RTC_CheckEsp) (129119Ah)
```

Hello World in Assembler (Intel 80x86)

Beispiel: Dualansicht (C++, Maschine) IV

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

```
        cout << "Der Wert für a ist: " << a << endl;
012914E0  mov         esi,esp
012914E2  mov         eax,dword ptr [__imp_std::endl (129A314h)]
012914E7  push        eax
012914E8  mov         edi,esp
012914EA  mov         ecx,dword ptr [a]
012914ED  push        ecx
012914EE  push        offset string "Der Wert f\xfcr a ist: "
(1297830h)
012914F3  mov         edx,dword ptr [__imp_std::cout (129A318h)]
012914F9  push        edx
012914FA  call        std::operator<<<std::char_traits<char> >
(129114Fh)
012914FF  add         esp,8
01291502  mov         ecx,eax
01291504  call        dword ptr
[__imp_std::basic_ostream<char,std::char_traits<char> >::operator<<
(129A300h)]
0129150A  cmp         edi,esp
```

Hello World in Assembler (Intel 80x86)

Beispiel: Dualansicht (C++, Maschine) V

```
0129150C  call        @ILT+405(__RTC_CheckEsp) (129119Ah)
01291511  mov         ecx,eax
01291513  call        dword ptr
[__imp_std::basic_ostream<char,std::char_traits<char> >::operator<<
(129A31Ch)]
01291519  cmp         esi,esp
0129151B  call        @ILT+405(__RTC_CheckEsp) (129119Ah)
        return 0;
01291520  xor         eax,eax
}
01291522  pop         edi
01291523  pop         esi
01291524  pop         ebx
01291525  add         esp,0CCh
0129152B  cmp         ebp,esp
0129152D  call        @ILT+405(__RTC_CheckEsp) (129119Ah)
01291532  mov         esp,ebp
01291534  pop         ebp
01291535  ret
```

Hello World in Assembler (Intel 80x86)

Zentraleinheit („Gehirn“ des Systems)

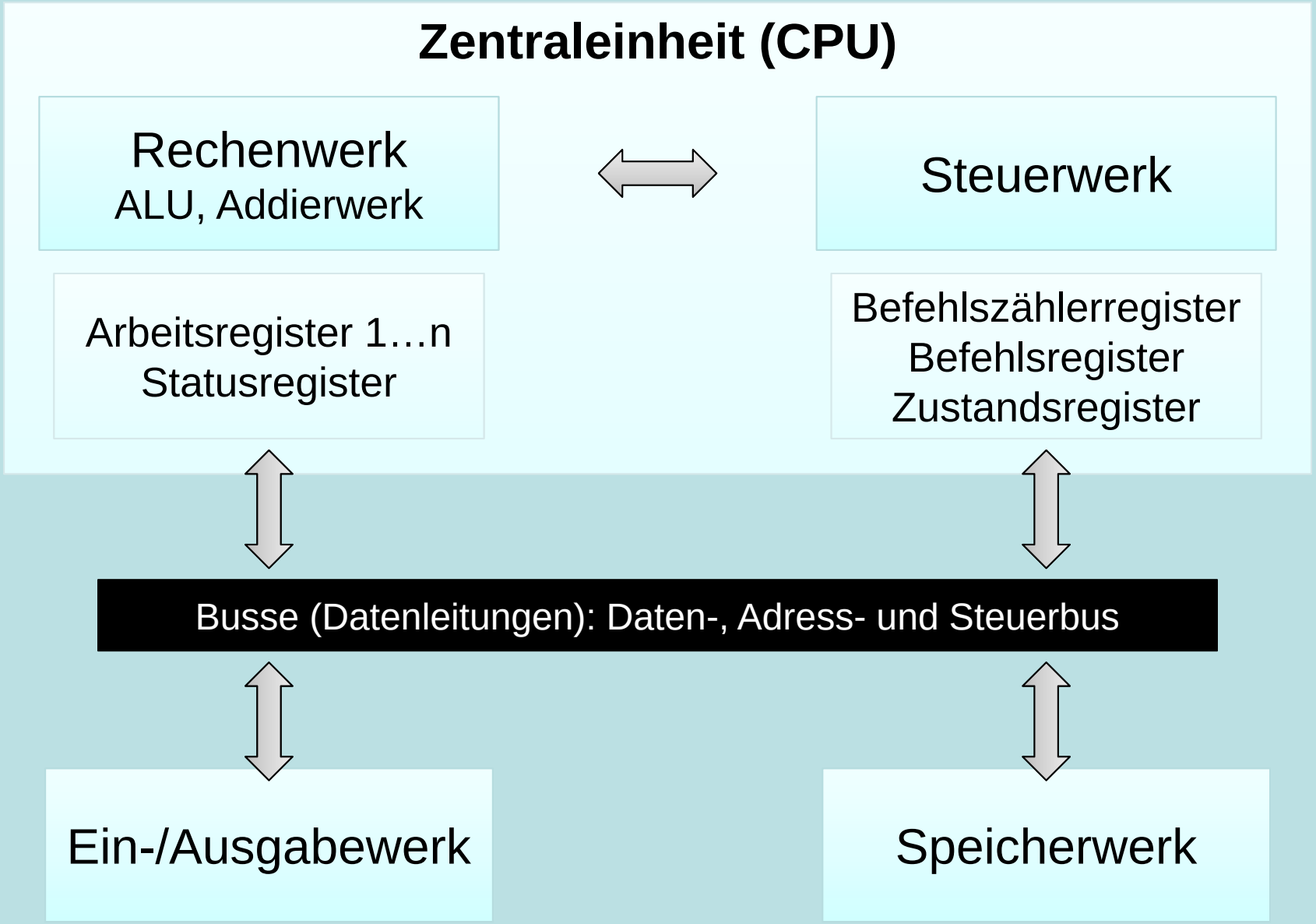
- Komponenten der Hauptplatine (engl. mainboard, motherboard)
  - Mikroprozessor (Herzstück des Rechners)
    - Ausführung der Programme / Steuerung und Verwaltung der Hardware
  - Arbeitsspeicher (Random Access Memory, RAM)
    - Daten und aktuell auszuführende Programme
  - Nur-Lese-Speicher (Read Only Memory, ROM)
    - Enthält heute nur noch das Basic-Input-Output-System (Firmware, die beim Einschalten die wichtigsten Hardware-Komponenten prüft und dann das Hochfahren des Betriebssystems veranlasst; Befehlsabfolge ist starr festgelegt, häufig ist die Firmware ohne Tausch eines Bausteins aktualisierbar: ggf. kritische Aktualisierung)
  - Busse und Schnittstellen
    - Kommunikation zwischen den einzelnen Bestandteilen der Hauptplatine
    - Anschluss weiterer Peripherie
  - Chipsatz
    - Steuerung sämtlicher Anschlüsse des Mainboards

Mikroprozessor (Prozessor, CPU)

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Integrierter Schaltkreis mit folgenden Komponenten
  - Rechenwerk: Arithmetisch-logische Einheit (ALU)
    - Mathematische und Logische Operationen
  - Register
    - Speicherplätze innerhalb des Prozessorkerns, z.B. R1…R15, EAX…EDX
    - Aufgeteilt in Arbeits- und Statusregister, Speicherung: 1 Maschinenwort
  - Steuerwerk
    - Steuert mittels Befehlszähler- und Befehlsregister den Programmablauf
    - Initiiert ggf. Steuerfunktionen und verwaltet den Stack-Zeiger
  - Befehlstabelle (engl. instruction table)
    - Enthält die Maschinenbefehle zwecks Decodierung des Programms
  - Busse (Datenleitungen)
    - Datenbus: Austausch von Daten mit dem Arbeitsspeicher
    - Adressbus: Übertragen der zugehörigen Speicheradressen
    - Steuerbus Ansteuerung der Peripherie-Anschlüsse

Komponenten eines Mikroprozessors



Maschinensprache

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Maschinenbefehl (kurz: Befehl)
  - Ein Befehl ist eine eindeutig spezifizierte Arbeitsanweisung an den Prozessor. Er ordnet eine Operation an, die in der Regel an spezifizierten Daten (Operanden) vorzunehmen ist.
- Maschinensprache
  - Befehlssatz des Prozessors
- Binäre Maschinenbefehle
  - Verarbeitung im Befehlsregister
  - Binäre Darstellung für den Menschen praktisch unlesbar (Kombination aus 0en und 1en)
- Symbolische Maschinenbefehle (Assembler-Befehle)
  - Einführung von Symbolen zwecks Verbesserung der Lesbarkeit
  - Individuell für fast jeden Prozessortyp

Beispiele für Assembler-Befehle

- `mov BX,$7A35 // bewege`
  Hole aus dem Arbeitsspeicher den Wert, der an der Adresse 0x7A35 steht, und lege ihn im Arbeitsregister BX ab
- `add BX,20 // addiere`
  Addiere den Wert 20 zum Inhalt des Rechenregisters BX
- `cmp BX,50 // vergleiche`
  Vergleiche den Wert im Register BX mit 50. Falls in BX der Wert 50 steht, wird ein bestimmtes Bit (Flag) im Zustandsregister (Status-Register) gesetzt
- `jeq $B7F4 // springe, wenn gleich`
  Falls der vorherige Vergleich „gleich“ ergeben hat, d.h. das Flag war gesetzt, springe zur Programmadresse 0xB7F4
- Anmerkungen
  - Reihenfolge der Ausführungen ist vom Prozessortyp abhängig

Binäre Maschinenbefehle

HOCHSCHULE REGENSBURG UNIVERSITY OF APPLIED SCIENCES

- Ein Assembler (engl. Monteur), „automatisches Montier-Programm“, setzt ein Maschinen-Programm bestehend aus symbolischen Befehlen in ein Programm bestehend aus binären Befehlen um („in die Sprache des Prozessors“)
- Da es sich um eine 1:1 Abbildung von Maschinen- auf Assemblersprache handelt, ist die Umkehrung möglich (Bezeichnung: Disassemblierung)
- Beispiel (Intel 80x86):

Assembler-Befehl	Maschinenbefehl	Kommentar
`add edx,0x0F`	83 C2 0F	Addiere 15 zum Inhalt des EDX-Registers
`sub eax,[v2]`	2B 45 F8	Subtrahiere den Inhalt von v2 von EAX
`neg ecx`	F7 D9	Negiere den Inhalt des ECX-Registers

Aufbau von Maschinenbefehlen

- Aufbau und Art unterscheiden sich bei verschiedenen Prozessortypen und -herstellern sehr stark
- Generelles Grundprinzip:

Operationsteil Adress-/Operandenteil

- Bestandteile eines Maschinenbefehls
  - Eigentlicher Befehl
  - Operandenteil mit Angabe der Adressierungsart
  - Operandenwert oder Adresse
- Beispiel:
  - `add ax,1000` → 0000010100000011111101000

    add ax (1000)$_{10}$
  - Anmerkung: Wort-Abbildung variiert (Low Byte/High Byte First)

Typische Adressierungsarten

- Registeradressierung
  - Ein Register im Prozessor wird adressiert
- Unmittelbare Adressierung
  - Operanden sind Bestandteil(e) des Befehls
- Absolute Adressierung
  - Eine Speicherzelle im Hauptspeicher wird adressiert
- Relative Adressierung
  - Die Speicherzelle im Hauptspeicher, die adressiert wird, ergibt sich aus dem übergebenen Wert plus dem Wert eines Spezialregisters (typ. Befehlszähler)
- Indirekte Adressierung
  - Die Speicherzelle im Hauptspeicher, die adressiert wird, steht an der adressierten Position
- Indizierte Adressierung
  - Die Speicherzelle im Hauptspeicher, die adressiert wird, ergibt sich aus dem übergebenen Index (Wert eines Registers) und dem übergebenen Offset