Rechner - Grundbegriffe

Auf der Grundlage eines Programms kann ein Computer

- Daten mit seiner Umgebung austauschen,
- mathematische und logische Operationen durchführen,

} E V A - Prinzip

- in Abhängigkeit von anfallenden Daten den Programm-ablauf zur Laufzeit verändern.

d.h.: er denkt nicht mit!

Er kann so mathematisch-logische Aufgaben lösen, wenn deren Lösungsweg entsprechend formuliert werden kann.

Aus Anwendersicht ist der Computer ein System zusammen-wirkender Hardware- u. Software-Komponenten.

„EVA“: Eingabe, Verarbeitung, Ausgabe

Rechner - Grundbegriffe

Definition: Als **System** bezeichnen wir

- Eine Menge von Komponenten
  (Gegenständen, Individuen, Größen, Prozessen, Ideen),
- die untereinander in einer kausalen Wechselwirkung stehen
- und von ihrer Umgebung entweder als abgeschlossen oder als in einer wohldefinierten Beziehung stehend betrachtet werden können
- sowie
  die Gesamtheit der unter ihnen herrschenden Beziehungen.

  Im Sinne dieser Definition kann jedes System in beliebig viele Teil- oder Subsysteme zerlegt werden, sofern dies zweckmäßig erscheint.

Rechner - Grundbegriffe

Die Existenz von Systemen ist meist mit der Erfüllung eines Zweckes verbunden (vgl. technische, politische, soziale, Regal-, Gleichungs- oder Planetensysteme).

Ein Rechensystem (Computer) ist ein System zusammenwirkender Hardware- u. Software-Komponenten (s.o.).

- **Hardware**: Materielle Komponenten eines Rechners

  „Gesamtheit o. Teil der apparativen Ausstattung von Rechensystemen“ (DIN 44 300)

  - d.h.:

  - physische, gegenständliche Bestandteile: Elektronik, Elektrik, Optik, Mechanik etc.
  - Hardware-Dokumentation ist nicht Teil der Hardware!

Software: Immaterielle Komponenten eines Rechners

„Gesamtheit o. Teil der Programme für Rechensysteme, wobei die Programme zusammen mit den Eigenschaften der Rechensysteme den Betrieb der Rechensysteme, die Nutzung der Rechensysteme zur Lösung gestellter Aufgaben o. zusätzliche Betriebs- u. Anwendungsarten der Rechensysteme ermöglichen“ (DIN 44 300)

- d.h.:

- alle Programme und Daten
- Auch **Firmware**, d.h.: in Chips gespeicherte (veränderliche) Programme, notwendig für den Betrieb von Rechnern (z.B.: BIOS), ISDN-Adaptern, Modems u.ä.

Basic Input Output System

Rechner - Grundbegriffe

Die technische Umsetzung der Beziehungen und Regeln für das Zusammenwirken zwischen Hardware- u./o. Software-Komponenten u./o. ihren Anwender/inne/n wird als die **Schnittstelle** zwischen den beteiligten Seiten bezeichnet.

Zu Schnittstellen zählen u.a.:

- die Anschlüsse für externe Geräte wie Maus, Drucker, Scanner (serielle, parallele, SCSI, USB etc.): *Hw/Hw-Schnittstelle*
- die Maschinenbefehle eines Rechners: *Hw/Sw-Schnittstelle*
- die Spezifikationen, mit denen Programme o. ext. Speicher unter einem Betriebssystem verwendbar sind: *Sw/Sw-Schnittstelle*
- die Programmaufrufe für den Einsatz systemnaher Programme (z.B. zur Fenstererstellung) in Anwendungsprogrammen: Application Programming Interface *Sw/Sw-Schnittstelle API*
- die Menügestaltung, Dialogführung und sonstige Bedienung eines Programms: *Mensch-Maschine-Schnittstelle MMI*

Rechner - Grundbegriffe

Der innere Aufbau (Design) eines Systems, das Zusammenspiel (Interaktion) seiner Komponenten und seine Beziehung zu anderen Systemen wird als die **Architektur** des Systems bezeichnet; sie ist maßgeblich durch die Systemschnittstellen bestimmt.

Systeme (Subsysteme, Komponenten), deren Schnittstellen sämtlich mit jenen eines anderen Systems übereinstimmen, heißen zu diesem System **kompatibel** und können entsprechend ganz o. teilweise an seiner Stelle eingesetzt werden.

> Kompatibilität sagt zunächst nichts über interne Struktur und Güte aus, sondern lediglich über das Verhalten nach außen.

Rechner - Aufbau

Hauptkomponenten eines programmierbaren Digitalrechners:
(J.P.Eckert, J.W.Mauchly, J.v.Neumann, 1944)

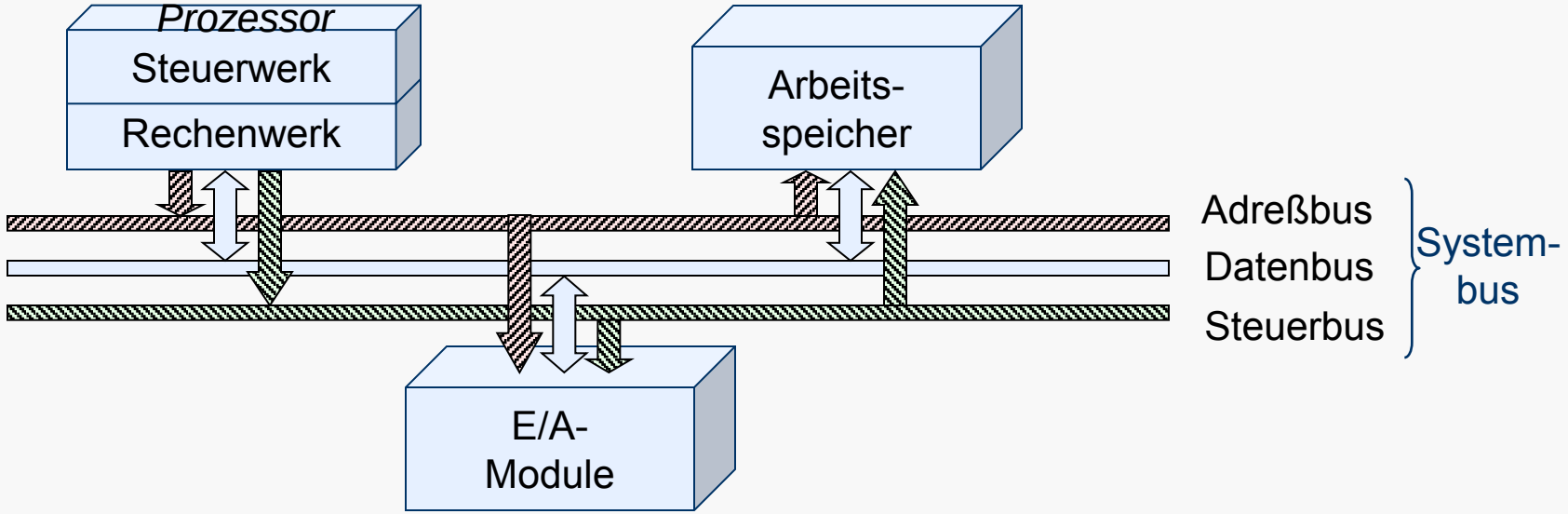
- Prozessor (*processor*)
- Arbeitsspeicher, Hauptspeicher (*memory*)
- Ein-/ Ausgabemodule (*I/O moduls*)

- Steuereinheit
- Arbeitsspeicherband
- Lese- / Schreibkopf

(Turing 1936)

Alle Einheiten arbeiten taktgesteuert und kommunizieren miteinander (über den Systembus).

Der Prozessor besteht aus Rechenwerk und Steuerwerk (Leitwerk).

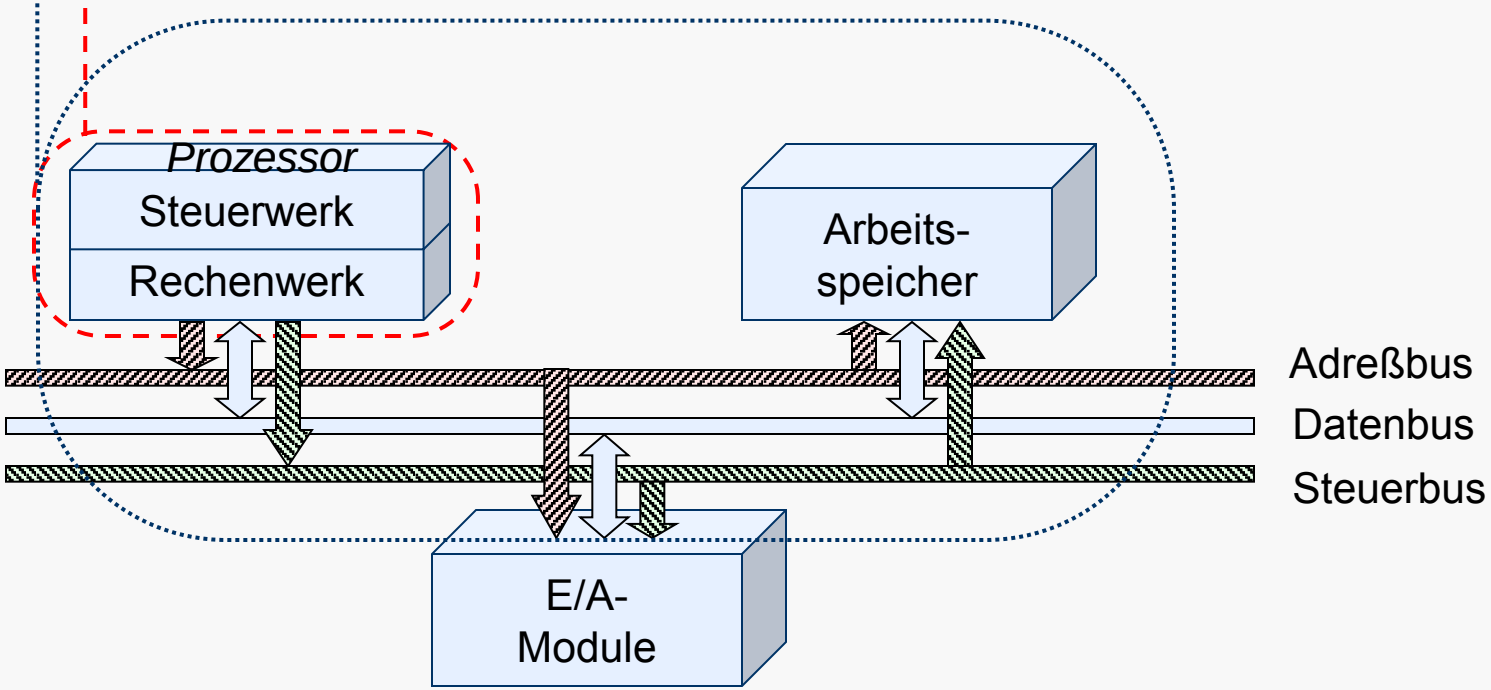


Rechner - Aufbau

Begriffsklärung:

Die CPU: (Central Processing Unit: Zentralprozessor, Zentraleinheit, zentrale Verarbeitungseinheit)

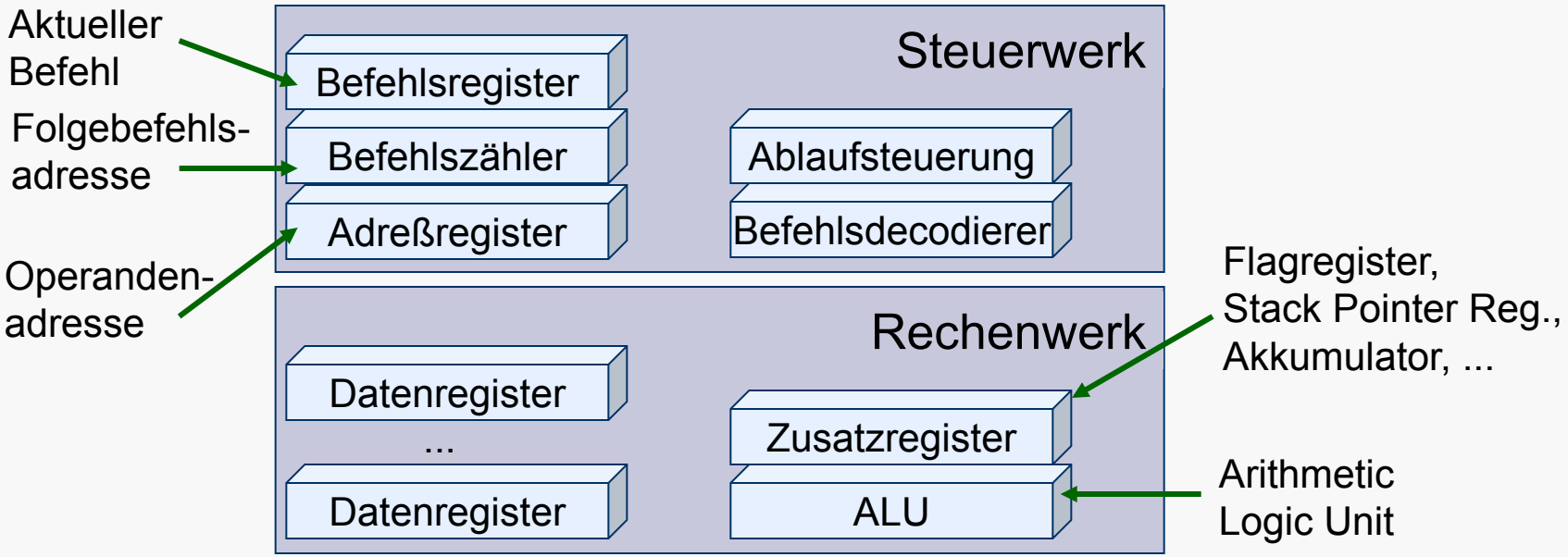
- ursprünglich: Prozessor + Arbeitsspeicher [+ E/A-Schnittstellen] (IEEE, DIN)
- heute meist: Prozessor [+ E/A-Schnittstellen] (Hersteller-Jargon)



Rechner - Aufbau

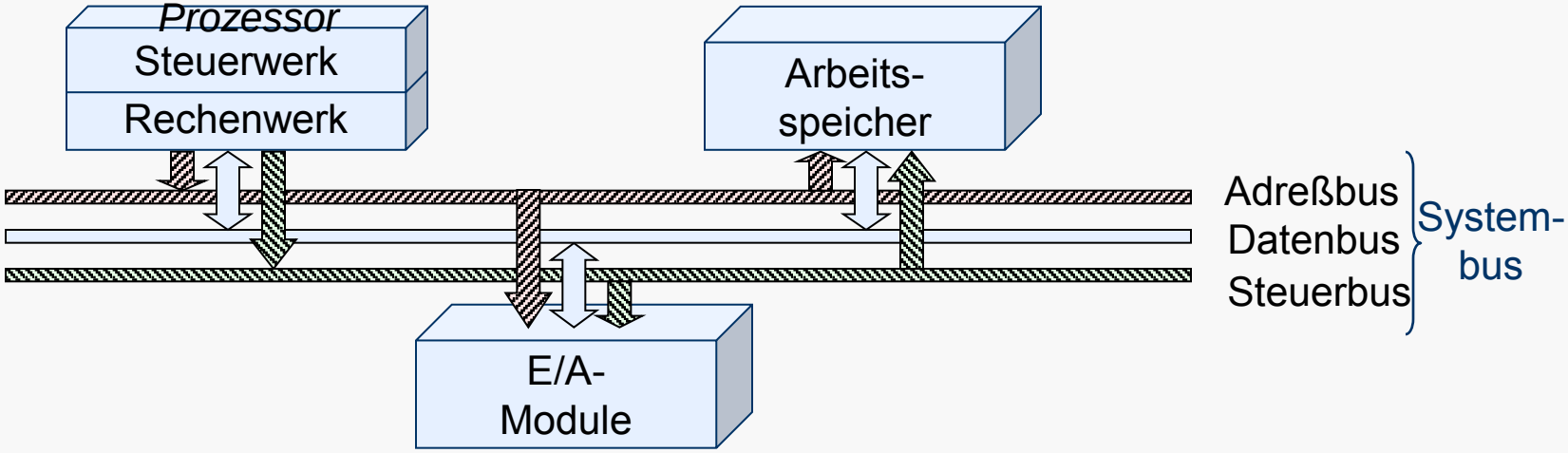
Aufbau des Prozessors:

- Steuerwerk
  - koordiniert Vorgänge in der CPU u. Transporte auf das Bussystem
- Rechenwerk
  - verknüpft Daten nach dem Prinzip der Boole'schen Algebra
    (mehrere Milliarden Arithmetik- o. Vergleichsoperationen /sec)



Rechner - Aufbau

- Zur Kommunikation innerhalb des Rechners werden Bussysteme eingesetzt.
- Busse sind Bündel parallel genutzter Leitungen zur Signalübertragung zwischen den Rechner-Komponenten (externer Bus) und innerhalb des Prozessors (interner B.).
- Zur besseren Auslastung / Vermeidung von Engpässen gibt es separate **Daten-**, **Adreß-** und **Steuerbusse**. Der Datenbus arbeitet **bidirektional**, der Adreßbus (meist) und der Steuerbus dagegen **unidirektional**.



Rechner - Aufbau

- Die Anzahl der Leitungen eines Busses wird dessen **Breite** genannt.

  Die Adreßbus- und vor allem die Datenbus-Breite beeinflussen stark die Leistungsfähigkeit eines µP's:

- Die **Datenbus**-Breite gibt an, wieviele Daten-Bits in einem Takt aus dem / im Rechenwerk übertragen werden (extern / intern – z.B.: 32/64).

- Aus der **Adreßbus**-Breite ergibt sich die Größe des adressierbaren Arbeitsspeichers:

  Ein *n*-Bit-Adreßbus ermöglicht einen 2^n-Byte-Speicher
  z.B.: 16 Leitungen $\Rightarrow$ 2^{16} Byte = 65.536 Byte = 64 KByte - bzw.:
  2^{16} Byte = 2^{6+10} Byte = 2^6 KByte = 64 KByte

  <u>Anmerkungen:</u>

  - 1 Byte ist die kleinste adressierbare Speicherzelle
  - Bei nicht ausreichendem Arbeitsspeicher werden Daten auf den (langsameren) virtuellen Speicher (Festplatte) ausgelagert.

Rechner - Funktionsweise

Sieben Operationsprinzipien des Digitalrechners (John v. Neumann, 1946):

1. Ein Rechner besteht (mindestens) aus:
   - **Arbeitsspeicher** für Programme und Daten (zur Ausführungszeit);
   - **Steuerwerk** zur Interpretation des Programms;
   - **Rechenwerk** zur Ausführung arithmetisch-logischer Operationen;
   - **Ein-/Ausgabewerk** für die Kommunikation mit der Umgebung.
2. Die Rechnerstruktur ist unabhängig von dem zu bearbeitenden Problem.
3. Programme u. Daten sind veränderlich und im selben Speicher resident.
4. Der Arbeitsspeicher ist in Zellen unterteilt, die durch ihre fortlaufende Numerierung (Adresse) gekennzeichnet sind.
5. Programme sind Befehlsfolgen, die sequentiell abgearbeitet werden (implizite Fortschaltungsregel).
6. Sprunganweisungen ermöglichen Abweichung von der Sequentialität; sie können (auch) durch Verarbeitungsergebnisse ausgelöst werden.
7. Interne Signale (Befehle, Daten) sind binär (bzw. dual) codiert.

Novum!

Der Befehlssatz eines Computers:

- Transportbefehle
  - Bewegen von Daten zwischen Speicher und/oder Register
- Verarbeitungsbefehle
  - Art der Operation(Addition, Subtraktion, Logische Operationen, ...)
  - Angaben über Speicher- u./o. Registeradressen der Operanden
- Sprungbefehle
  - Unbedingte Sprünge
  - Bedingte Sprünge
  - Sprünge mit Rückkehrabsicht (jump subroutine)
  - Rücksprünge (return from subroutine)
- Ein-/Ausgabebefehle
  - Transportbefehle zu besonderen Adressen außerhalb des Hauptspeichers (z.B. Bildschirm, Drucker)

Rechner - Funktionsweise

- Soll ein Programm ausgeführt werden („laufen“), wird es vom permanenten Speicher (früher: Lochstreifen, heute: Festplatte) in den (flüchtigen) Arbeitsspeicher kopiert - „geladen“.

- Befehle sowie die dazugehörigen Daten u. (Folgebefehls- oder Daten-) Adressen werden vom Arbeitsspeicher in die für sie vorgesehenen Prozessor-Register übertragen („geholt“) und sequentiell abgearbeitet.

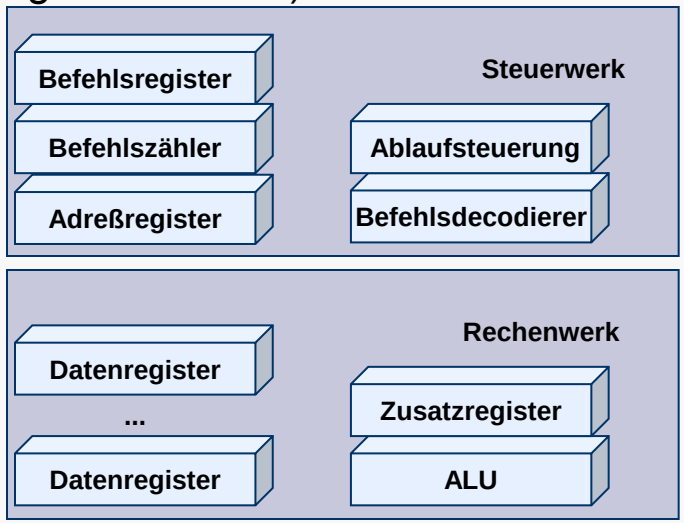
- Jeder Befehl enthält die Information darüber, wo die zu verarbeitenden Daten liegen und welche Operation mit ihnen auszuführen ist.

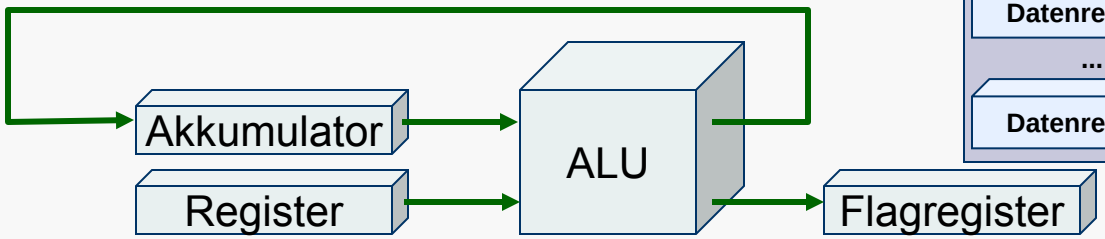
- Register sind z.T. spezialisiert - etwa:
  - der Akkumulator zur Speicherung eines Operanden, der nach Operation durch das Ergebnis überschrieben wird;
  - das Flagregister zur Anzeige von Besonderheiten bei der Ausführung von Operationen („Übertrag“, „Ergebnis negativ“ o.ä.);
  - das Kellerregister (Stack Pointer, Stapelzeiger) zur Speicherung wichtiger Daten bei d. Behandlg v. Interrupts o. Unterprogrammen.

Rechner - Funktionsweise

Befehlszyklus:

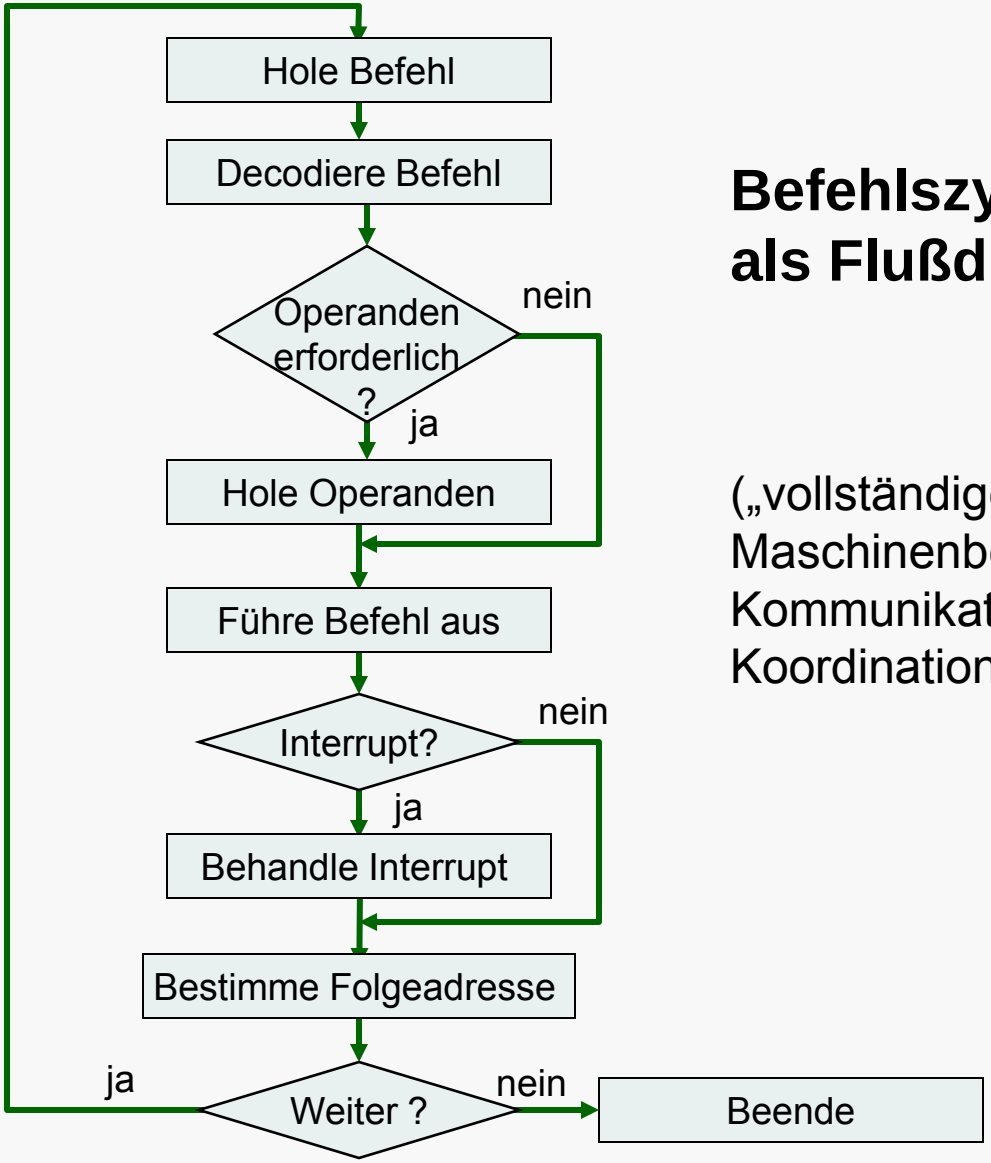
1. Hole Befehl ins Befehlsregister
   - Befehls-Adresse steht im Befehlszähler (PC: Program Counter)
2. Decodiere den Befehl und führe ihn aus
   - Befehl decodieren
   - Befehlstypische Ausführungsschritte vollziehen
   - Ggf. weitere Daten vom Speicher holen
     ⇨ Ausführungszeit ist befehlsabhängig!





3. Bestimme Folgeadresse
   - Je nach Ergebnis der Befehlsausführung:
     Folgeadresse ist nächste Befehls-Adresse oder eine Sprungadresse.
4. Prüfe auf Unterbrechungsanforderung
   - Reaktion auf äußere Ereignisse (engl. Interrupt)

Rechner - Funktionsweise



Befehlszyklus als Flußdiagramm

(„vollständiges Programm“: Folge von Maschinenbefehlen für Problemlösung und Kommunikation mit I/O u. Speichermedien, Koordination laufender Programme etc.)

Betriebssystem

Zur Entlastung (Ermöglichung) der Programmierarbeit werden häufig benötigte, aufgabenunabhängige, aufwendige, maschinenspezifische Programmteile als **Module** realisiert und oft zum sog. **Betriebssystemkern** zusammengefaßt.

austauschbare Funktionseinheiten

z.B. Defragmentierung, Komprimierung

Mit weiteren nützlichen **Dienstprogrammen** bildet er das **Betriebssystem** (BS, engl.: operating system) d. Rechners.

Im Sinne der System-Definition bilden das Betriebssystem Programme, deren Zusammenwirken eine ‚wohldefinierte' Beziehung der Rechner-Ressourcen untereinander und zum Anwender (bzw.: zur Anwendung) herstellt.

Komponenten

Umgebung

Betriebssystem

Ein Betriebssystem (BS) erfüllt 2 Aufgaben (außen / innen):

- Übernahme der (schwierigen) Programmierung einzelner Hw-Komponenten und deren Zusammenspiels. Für den Anwender verhält sich das Betriebssystem wie eine **Virtuelle Maschine**.

  Wichtig:

  Einführung von Abstraktionsstufen / **Information hiding**

- Bei mehrfacher gleichzeitiger Nutzung (Programme, Menschen): Regelung des Zugriffs auf Hw-Komponenten, Daten, Rechenzeit: **Ressourcen-Verwaltung**.

Betriebssystem

BS: Summe permanent installierter Programme, die – mit den Hw-Eigenschaften – die möglichen Betriebsarten des Rechensystems festlegen und insb. die Ausführung von Anwendungsprogrammen steuern und überwachen.

Das BS verwaltet die Ressourcen des Rechensystems:

- Betriebsmittel (CPU, Arbeitsspeicher, Festplatte, Peripherie),
- Benutzeraufträge (Tasks) und
- Datenbestände

alle Vorrichtungen, die nicht zur CPU gehören

und steuert die dazugehörigen Arbeitsabläufe – z.B.:

- Berücksichtigung von Unterbrechungssignalen (Interrupts)
- Auslagerung des laufenden Programms, damit ein anderes CPU-Zeit bekommt (time slicing)

Das BS wird sofort nach Einschalten des Rechners vom permanenten Speicher in den Arbeitsspeicher geladen: Bootstrapping, **booten**

Betriebssystem

Beispiel: Startvorgang des IBM-PC

1. Basis-**Hardwaretest** (POST - Power On Self Test)
   - Überprüfung Prozessor
   - Überprüfung div. Schaltkreise (DMA / Interrupt)
   - Überprüfung RAM / BIOS
2. Bestimmung der **Konfiguration**
   - Bestimmung der RAM-Größe
   - Bestimmung der Schnittstellenparameter (z.B. Portadressen, Anzahl und Art der Festplatten, ...)
3. Grund-**Initialisierung**
   - Initialisierung der Schaltkreise und Schnittstellen
4. **ROM**-Scan (Suche nach BIOS-Erweiterungen)
5. **BOOT**
   - Laden des Betriebssystemkerns

Betriebssystem

Architektur-Modelle von BSen:

- Monolithische Architektur

  (homogenes Sw-Gebilde, effizient, unflexibel)
- Mehrschichten-Modell

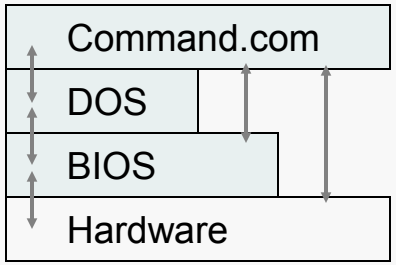
  (Verteilung der Funktionalität über mehrere Ebenen: MS-DOS)
- Mikrokern-Architektur

  (Minimale Grundausstattung, nach Bedarf mehr: Windows NT ff.)
- Kern-Schale-Architektur

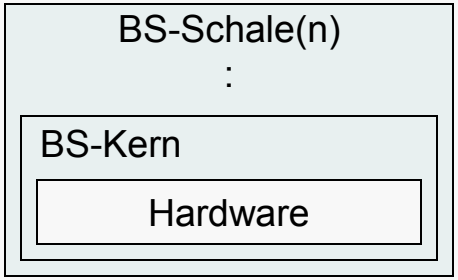
  (Abschottung der Hw durch mehrere Sw-Schalen: UNIX)
- Virtuelle Maschine

  (Basis-BS, auf dem das eigentliche BS installiert wird: IBM ...)

Quasi-konsistente MS-DOS-Schichten:



UNIX-Shells:



Betriebssystem

Kategorisierung von Betriebssystemen:

- Single-User / Single-Tasking
  Einzelnutzer-, Einzelprozeß-System, -Betrieb
  Beispiele: MS-DOS, Windows 3.11, System 7, ...
- Single-User / Multi-Tasking
  Einzelnutzer-, Mehrprozeß-System, -Betrieb
  Beispiele: Windows 9x, Windows NT, OS/2
- Multi-User / Multi-Tasking
  Mehrnutzer-, Mehrprozeß-System, -Betrieb
  Beispiele: VMS, UNIX, ...

Bei Multi-Tasking unterscheidet man (bei PCs) zwischen

- **kooperativ**em bzw. nicht-präemptivem Multi-Tasking (nichtverdrängendem Mehrprozeßbetrieb):
  Applikation entscheidet selbst, wann sie Kontrolle über Rechenzeit abgibt (z.B. bis Windows 3.11) – und
- **präemptiv**em Multi-Tasking (verdrängendem Mehrprozeßbetrieb):
  BS entzieht die Rechenzeit den Applikationen oder weist sie ihnen zu (z.B. ab Windows 95).

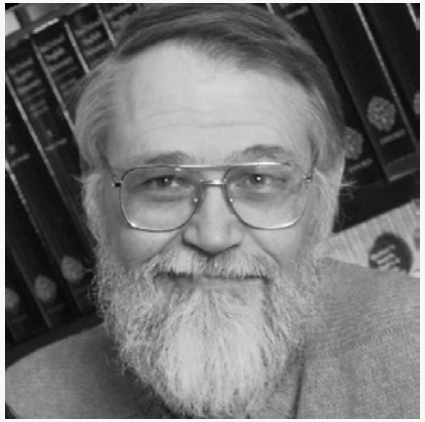
„das Multi-Tasking“

Die Applikationen können sich zudem in mehrere sog. **Threads** („Ausführungsstränge“) aufspalten, die parallel ausgeführt werden.

(z.B.: Menü-Nutzung o.ä. während langwieriger Speicherung)

Betriebssysteme

- Bestrebungen zur Hw-Abstraktion seit Z3 (K.Zuse, 1941)
- 1960er: Einführung des Dialogbetriebs
  (zuvor: Stapelbetrieb , Operator)
- Maßgebliche Rolle dabei: Entwicklung von Unix durch Beiträge von
  Ken Thompson (*1943)
  Dennis Ritchie (1941-2011)
  Brian Kernighan (*1942)



Brian Kernighan

Ken Thompson Dennis Ritchie

- Mitte der 60er: K.Thompson, D.Ritchie (Bell Labs, AT&T) und MIT-Forscher entwickeln für Mainframe GE645 (General Electric) Multiuser-BS MULTICS:

  MULTiplexed **I**nformation and **C**omputing **S**ystem

- MULTICS wird schließlich nicht eingesetzt: „techn. Mängel“

  ⇨ Weiterentwicklung durch Thompson („Space Travel“)

  ⇨ Verballhornung durch B.Kernighan: „UNICS“

- 1969: Betriebssystem UNICS, ab 1970: „Unix“ (in Assembler)

  Gleichzeitig (Thompson, Ritchie): Programmiersprache A (BCPL-basiert) ⇨ B ⇨ C.

- 1973: Unix erstes BS größtenteils in der Hochsprache C (kaum 1000 Zeilen Maschinencode) ⇨ Portierbarkeit!

- US-Kartell-Bestimmungen verhindern die Vermarktung von Unix durch AT&T
- Ab ca. 1975: Unix zum Selbstkostenpreis an Universitäten

  Univ. of California: Berkeley Software Distribution (BSD)

  ⇨ Erweiterungen
- 1984: IEEE/POSIX ⇨ US-Standardisierungsvorgabe
- 1988: ANSI-C
- 1980-1990: Xenix (Microsoft, ab Mitte 80er: Santa Cruz Operation)
- 1991: Linux 0.02 (Linus Torvalds, FIN)
- 1993: Windows NT

Betriebssysteme

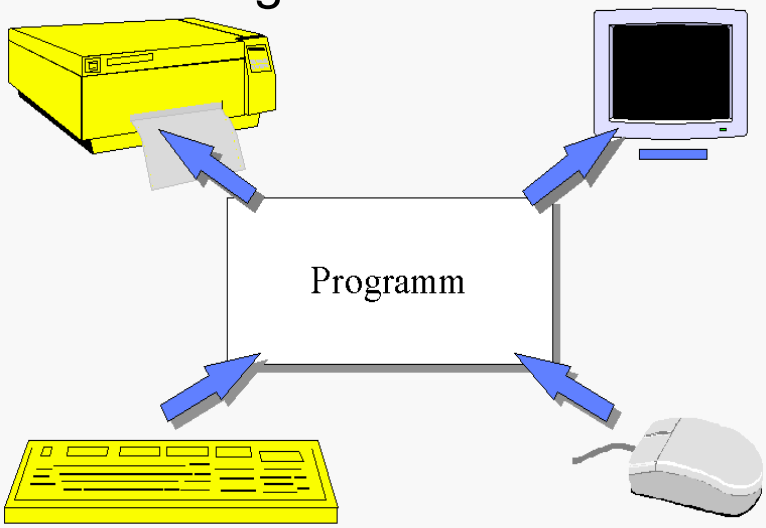
- Heute: ca. ein Dutzend Betriebssysteme; Marktführer: MS
- Gründe für d. PC-Erfolg (Debüt: IBM 1981/ mit MS-DOS)
  - Mächtiger Anbieter (IBM):
    Gewähr für Fortbestand & Anbindung an Großrechner
  - Modulares Bau-Konzept:
    Nach Bedarf aufrüstbar & Günstige Nachbauten Dritter
  - Orientierung am Massenmarkt („Schneeball-Effekt“):
    Erschwinglicher Preis & Kurze Einarbeitung
- Paradigmenwechsel im Anspruch an Rechner:
  ≤80er: Erzeugung v. Ergebnissen; Darstellung auf Drucker
  heute: Ständiger Dialog; Ergebnisse in div. Formen/Medien
- Wichtig für die Realisierung: Forschung am Xerox PARC
  Mitte 70er: Konzept für graf. Benutzungsschnittstelle (GUI)
  Apple: 1983 „Lisa“, '84 Macintosh; MS:1985 Windows 1.01

Grafische Benutzungsschnittstellen

⇨ Steigerung d. Hw-Abstraktion durch zusätzliche Sw

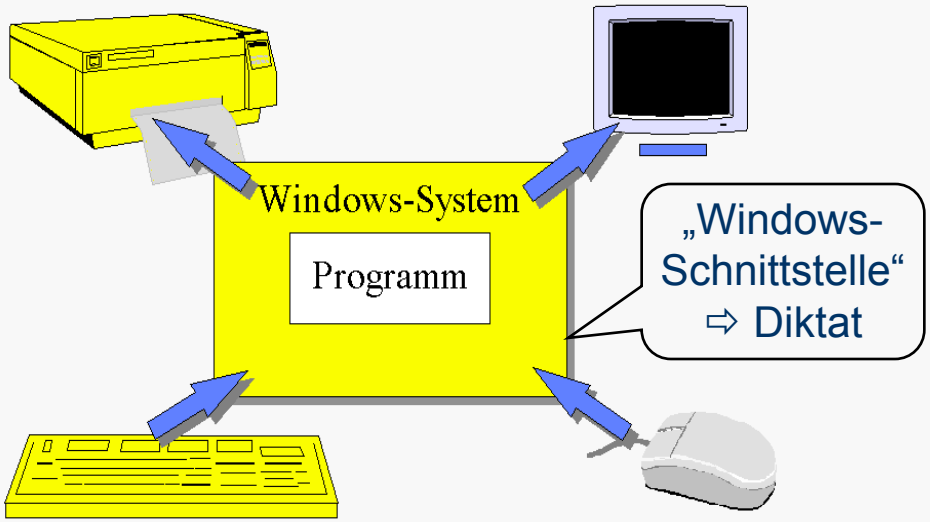
Anwendungsprogrammierung unter **MS-DOS**:

"Standard-I/O" u.standardisierte Rechenzeit-Zuweisung; abweichende Handhabung durch Programmierer/in



Windows-Programmierung:

Kommunikation mit Peripherie ins System integriert: Geräte werden "wie unter Windows" angesprochen.



Bilder: W. Doberenz, Th. Kowalski: "Programmieren lernen in Visual Basic 5", Hanser 1997

Zur Verdeutlichung: Word for DOS auf ≥20 Disketten ausgeliefert; darunter: 1-2 für das Programm selbst, Rest für diverse Treiber

Grafische Benutzungsschnittstellen

- Ziel der GUI: Verlagerung des Schreibtisches in den PC
  „Desktop“: Rechner auf dem Tisch – eher: Tisch-Oberfläche!
  Vorstellung: Papier-Stapel („Batch“) vs. lose Blätter (Fenster)
- Einige Forderungen an GUI und ihre Auswirkungen
  - Intuitive Handhabung ⇨ Minimierung d. Voraussetzungen
    (z.B.: Brief-/Mail-Schreiben nach Klick statt Befehl-Eingabe)
    ➥ Design / Ergonomie
  - Quasi-parallele Bearbeitung ⇨ Präemptives Multitasking
    (z.B.: Erstellung v. Präsentation während Internet-Suche)
  - Arbeits-Abläufe einer Person ⇨ Prozeß-Kommunikation
    (z.B.: Datei-Löschen i. allen betroffenen Fenstern anzeigen)
    ➥ Plattform / Betriebssystem
  - Einheitliches „Look & Feel“ ⇨ Kapselung d. Ein-/Ausgabe
    (z.B.: „Drag&Drop“ programm-übergreifend realisiert)
    ➥ Applikationen / Bibliotheken

- Dialog-Komponenten immer komplexer / differenzierter
  (z.B.: gezielte Warn-Meldungen mit Fallunterscheidungen)
- Dialog (quasi-)unabhängig vom Programm
  (z.B.: Menüleisten mit: „Datei - Bearbeiten - Ansicht -...“)

⇨ Loslösung der eigentlichen Applikation von Ein-/Ausgabe

⇨ Abgabe der Ablaufkontrolle an GUI-Umgebung/-Plattform

Konsequenzen für die Applikations-Entwicklung:

- für die Ergonomie: sehr groß
- für das DV-Ergebnis: keine
- für die Sw-Konzeption: gering
- für die Codierung: deutlich, v.a. für kleine Sw-Projekte

Zusätzlicher Codierungsaufwand durch Berücksichtigung

- der Plattform-Aufrufe
  (z.B.: Ausgabe im Fenster) - aber auch
- der plattform-eigenen Erfordernisse (Plattform-‘Eigenleben‘)
  (z.B.: Änderung der Fenstergröße, Löschung des Fensters)

Vorteile der Anwendung grafischer Benutzungsschnittstellen:

- Intuitive Dialoggestaltung
  (z.B.: Animierte Tastatur-Bilder für Einloggen)
  ➥ Leichte Erlernbarkeit von Programmen

- Ähnliche Strukturierung unterschiedlicher Programme
  (z.B.: „Datei - Bearbeiten - Ansicht - ... /Senden an:)
  ➥ Schnelle Einbeziehung neuer Lebensbereiche in DV

- Fortentwicklung des Bildschirms zum „Eingabegerät“
  ➥ Interaktion: Computer als Arbeits-/Lernpartner

- WYSIWYG
  ➥ Planbarkeit von Arbeit u. Ergebnis

Akronym: „What You See Is What You Get“
(bzgl. eines Text-Druckbildes)

B. Kernighan: " WYSIAYG:
What You See Is All You Get"