## Moores Law (1965 geäußerte Vermutung)

- Anzahl der Transistoren je Prozessorchip verdoppelt sich alle 2 Jahre
- Verarbeitungsleistung der Prozessoren verdoppelt sich alle 1,5 jahre
- Speichergröße vervierfacht sich alle 3 Jahre
- Speicherleistung verdoppelt sich alle 10 Jahre
- Zum gleichen Preis die doppelte Leistung in weniger als 2 Jahren

## Zahlensysteme

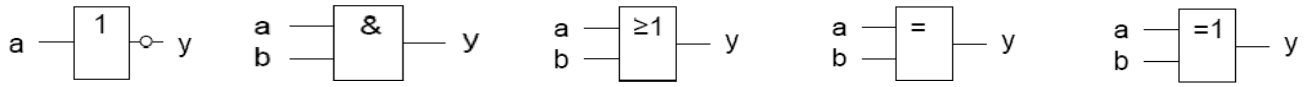
- Integer-Zahlen ohne Vorzeichen (nur positive Zahlen)
- Integer-Zahlen mit vorzeichen (zwei Nullen: 0000, 1111)
- 1er Komplement (Erstes Bit ist Vorzeichen, bildung durch Negierung jedes einzelnen Bits, zwei Nullen)
- 2er Komplement (Erstes Bit ist Vorzeichen, Bildung durch 1er Komplement + 1)

## Code-Systeme

- BCD-Code ( [0-9] dual codiert mit [0000-1001], immer vier Bits stellen eine Dezimalstelle dar)
- Unicode (16 Bit, die ersten 8 Bit sind ASCII-Code)

## Boolesche Algebra

### Symbole




NOT AND OR XNOR XOR

### Regeln

| | | | | |
|---|---|---|---|---|
| Kommutativ | a+b = b+a | | a*b = b*a | |
| Assoziativ | (a+b)+c = a+(b+c) | | (a*b)*c = a*(b*c) | |
| Distributiv | a+(b*c) = (a+b)*(a+c) | | a*(b+c) = (a*b)+(a*c) | |
| Idempotenz | a+a = a | | a*a = a | |
| Absorption | a+(a*b) = a | | a*(a+b) = a | |
| Null & Eins | 0+a = a | 1+a = 1 | 0*a = 0 | 1*a = a |
| Komplement | $a + \overline{a} = 1$ | | $a * \overline{a} = 0$ | |
| de Morgan | $\overline{(a+b)} = \overline{a} * \overline{b}$ | | $\overline{(a*b)} = \overline{a} + \overline{b}$ | |

## Entwicklung kombinatorischer Schaltungen

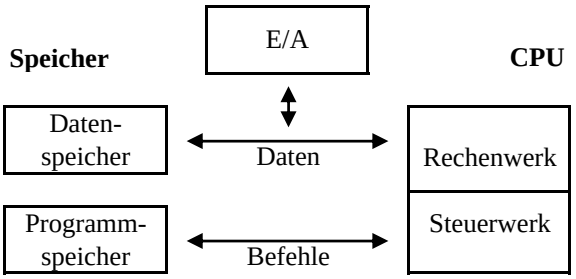
- Funktionstabelle
- Funktionsgleichungen aufstellen (DNFs)
- Minimierung (z.B. mit KV-Diagramm)
- Schaltungsrealisierung

## Schichtenmodell eines Rechnersystems

| | |
|---|---|
| Anwendungsprogramme | Word, PowerPoint |
| Höhere Programmiersprachen | Java, C++ |
| Assemblersprachen, Maschinencode | Assembler |
| festverdrahtete Steuerung | |
| Funktionseinheiten | ALU, Speicher |
| Logikebene | Gatter (UND, ODER,…) |
| Physikalische Ebene | Transistoren /Verbindungen |

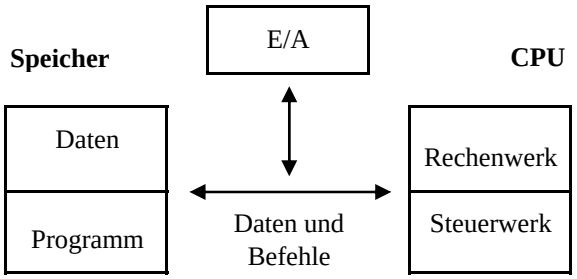
## Grundarchitekturen

Harvard-Architektur



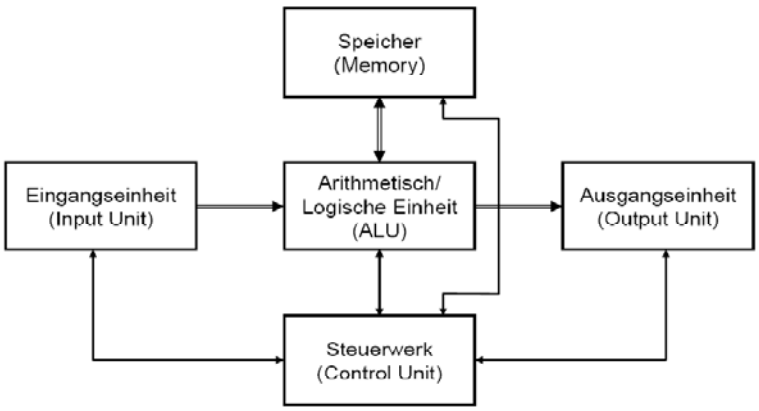

- Befehle und daten in getrennten Speichern
- Je ein Befehls- und Datenbus
- Schneller gleichzeitiger Zugriff auf Code und Daten
- Hauptanwendung: Signalprozessoren

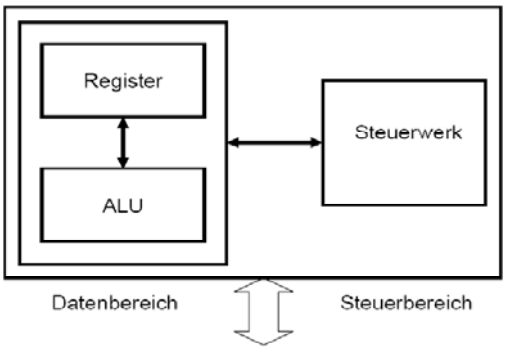
Von-Neumann-Architektur



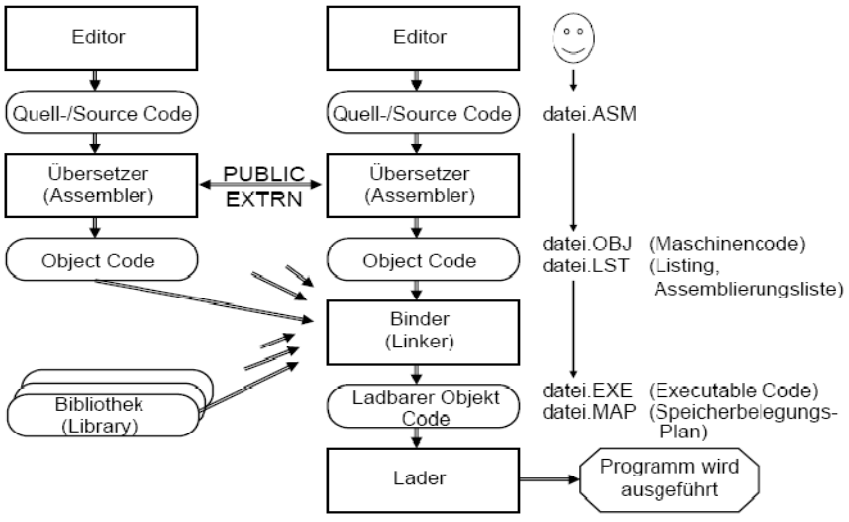

- Befehle und Daten im gemeinsamen Speicher
- Nur ein Bus für Befehle und Daten
- Flexible Aufteilung des Speichers zwischen Code und Daten
- Hauptanwendung: Allgemeine Computer

## Blockschaltbilder: Von-Neumann-Architektur / Aufbau einer CPU




## Programmentwicklung Assembler



## Assembler und Hochsprachen

- Import von Assembermodulen beim Linken (obj-Dateien)
- Verwendung des inline-Assemblers des jeweiligen Compilers. (geeignet für kurze Assemblerroutinen, wird mit asm (Borland) bzw. __asm (Microsoft) eingeleitet)
- Zusammenfassen mehrerer Befehle zu einem Block durch Klammerung ( asm {…Assembler-Code ... } ) möglich.

## Einteilung Interrupts

Ein Interrupt ist eine Unterbrechung. Das laufende Programm wird unterbrochen und ein spezielles Programm gestartet, die so genannte Interrupt Service Routine (ISR). Nach Beendigung der ISR wird an der ursprünglichen Position im Programm fortgefahren.
Die Interrupts des 8086 können eingeteilt werden in:

- *Software-Interrupts*
  Werden vom Programmierer ausgelöst (ähnlich Unterprogrammaufruf).
  z.B. INT 21H = DOS-Funktionsaufruf
- *Exceptions*
  Werden in Ausnahmesituationen durch den Prozessor selbst ausgelöst.
  z.B. Auslösen von Interrupt Nr.0 bei Division durch 0.
- *Externe (asynchrone) Interrupts*
  Werden durch externe Bausteine bzw. Ereignisse ausgelöst.
  z.B. Periodischer Interrupt, ausgelöst durch einen Zeitgeberbaustein (Timer).

## Interruptverarbeitung

1) HW- oder SW-Interrupt innerhalb eines Programms ist aufgetreten; der gerade in Abarbeitung befindliche Befehl wird zu Ende ausgeführt.
2a) Interrupt-Nummer wird mit vier multipliziert
2b) Adresse des nächsten Programmbefehls des Hauptprogramms (Rücksprungadresse) und die Flags werden auf den Stack gesichert.
3) Zugriff auf die Interrupt-Vektortabelle mit der in Schritt 2a berechneten Adresse
4) Interrupt-Vektortabelle enthält für alle Interrupts die zughörigen Einsprungadressen (Programm n: ISR n)
5) Am Ende des Interrupt-Programms befindet sich ein IRET-Befehl. IRET liest die Flags und die Rückkehradresse vom Stack.
6) Der Prozessor kann die Abarbeitung des Programms mit dem auf den Interruptbefehl folgenden Befehl fortsetzen.

## Interrupt-Vektortabelle

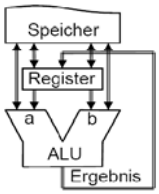
- Ist ein großer block reservierter Speicherplätze (1024 Kbyte)
- Dort liegen die Interruptvektoren (Einsprungadressen) für die 256 möglichen Interrupts
- Jeder Interruptvektor besteht aus Segment und Offset, belegt also 4 Byte.

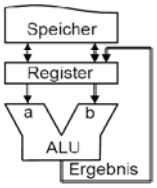
## Lokale Prozessor-Klassifikation

- Stack-Architektur
- Akkumulator-Architektur
- Allgemeinzweck-Register-Architekturen (GPR)
  - Register-Register-Maschinen - RISC
  - Register-Speicher-Maschinen - CISC
  - Speicher-Speicher-Maschinen

## GPR-Architektur

| Complex Instruction Set Computer (CISC) | Reduced Instruction Set Computer (RISC) |
|---|---|
| Mischung aus Akkumulator und Load-Store-Architektur | Load-Store-Architektur |
| Komplexe Instruktionen, Ausführung in mehreren Taktzyklen | Einfache Instruktionen, Ausführung in einem Taktzyklen |
| Jede Instruktion kann auf den Speicher zugreifen | Nur Lade- und Speicherbefehle greifen auf den Speicher zu |
| Kein oder wenig Pipelining | Intensives Pipelining |
| Instruktionen werden vom Mikroprogramm interpretiert | Instruktionen werden durch festverdrahtete HW ausgeführt |
| Instruktionsformat variabler Länge | Instruktionen alle mit festen Länge |
| Kompakter Code | Code weniger kompakt |
| Viele Instruktionen und Adressierungsarten | Wenige Instruktionen und Adressierungsarten |
| Die Komplexität liegt im Mikroprogramm / mächtige Befehle | Die Kompexität liegt im Compiler |
| Einfacher Registersatz | Umfangreiche Registersätze (32-512) |
| | |




## Parallelrechnerarchitekturen (nach Flynn)

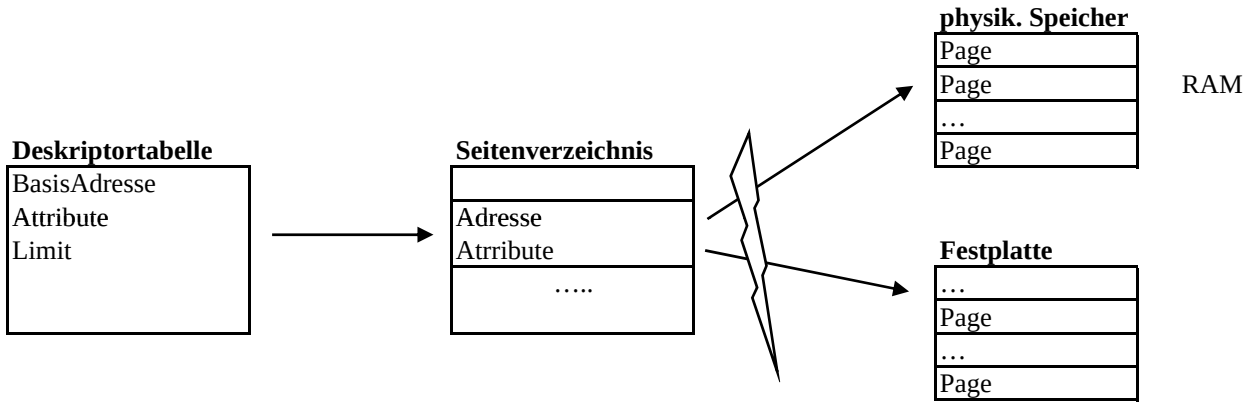
| | |
|---|---|
| **SISD**<br>Single Instruction<br>Single Data<br><br>z.B. Von-Neumann | **MISD**<br>Multiple Instruction<br>Single Data<br><br>z.B. Pipelining bei Unix |
| **SIMD**<br>Single Instruction<br>Multiple Data<br><br>z.B. MMX, 3D-Now | **MIMD**<br>Multiple Instruction<br>Multiple Data<br><br>z.B. Mehrprozessorsys. |

## Operation Modes 80386 bis Pentium

- *Real-Adress Mode*
  Stellt die Programmierumgebung des 8086 zur Verfügung und zusätzlich die Möglichkeit in den Protected oder System Managment Mode zu wechseln. Prozessor arbeitet als hochgetakteter 8086.
- *Virtual-8086 Mode*
  "quasi-operating mode". Erlaubt die Ausführung von 8086-Software in einem protected, multitasking environment
- *Protected Mode*
  Originärer Mode des Prozessors. Alle Befehle und Architektureigenschaften sind nutzbar
- *System Managment Mode (quasi operating mode)*
  An allen Intel-Prozessoren ab 386 verfügbar. Ermöglicht beispielsweise die Implementierung von Power Managment.

## Virtuelle Speicherverwaltung

- Ein virtueller Speicher bildet ein komplexes Programm ab.
- Ist eine so genannte Erweiterung des RAM.

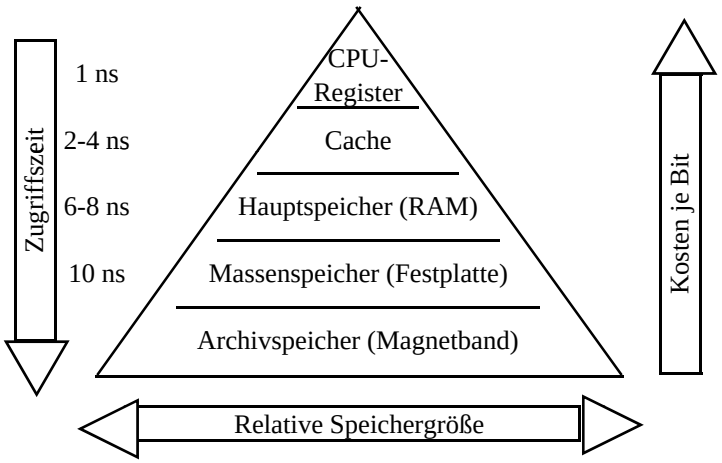



Atrribute:
- Present-Bit
- Zugriffsrechte

Interrupt:
MMU der CPU erkennt,
dass Seite nicht im RAM ist.
-> BS lagert die fehlende Seite ein
und aktualisiert die Seitentabellen.

Strategien für den Seitenwechsel:
1) First In First Out (FIFO)
2) Least Recently Used (LRU)
3) Least Frequently Used (LFU)

## Speicherhierarchie



## Cache

Kaschierer (Verdecker), Versteck, Depot
- schneller Pufferspeicher, der den nachgeordneten langsamen Speicher kaschiert / verdeckt (zwischen CPU und RAM)

Assoziativspeicher:
- Inhaltsadressierbarer Speicher, die Adressierung erfolgt teilweise über die gesuchten Daten selbst

1) Index adressiert mittels Dekoder eine Zeile im Tag-Ram
2) Tag wird dann mit Inhalt der betreffenden Zeile im Tag-Ram verglichen: identisch (Hit), ungleich (Miss)

Schreibzugriff Cache:

1) Durchschreibeverfahren: CPU schreibt Datum gleichzeitig in den Cache und in den Hauptspeicher
   Vorteil: Daten sind immer konsistent
   Nachteil: Zugriffszeit ist durch den Zugriff auf den Hauptspeicher bestimmt
   -> Cache bringt keinen Vorteil beim Schreiben
2) Rückschreibeverfahren: CPU schreibt Datum zunächst nur in den Cache -> schneller Zugriff
   Geschriebene Daten werden durch so genanntes Drity Bit gekennzeichnet
   Rückschreiben erfolgt durch Cache Controller wenn das Datum in den Hauptspeicher ausgelagert werden muss -> CPU arbeitet parallel weiter
   Vorteil: Schneller Schreib-/Lesezugriff
   Nachteil: Datenkonsistenzproblem bei Zugriff weiterer Einheiten auf Hauptspeicher

## MMX

MMX = Multimedia Extensions
MMX-Prozessor kann als Vektorrechner im Kleinen bezeichnet werden (SIMD)
8 zusätzliche Register der FDU mit einer speziell auf Multimedia ausgerichteten Arithmetik

Sättigungsarithmetik: Bei der Ton- und Bildverarbeitung ist eine seperate Arithmetik für Multimedia sinnvoll
Sättigung: ein kleiner Wert für eine Farbintensität kann problemlos auf 0 gesetzt werden, bzw. ein Überlauf im Blauton für ein Bild ist harmlos, wenn einfach der maximale Wert eingestellt wird.

## VGA

- Video Graphics Array
- 640 x 480 Pixel Auflösung, 16 oder 256 Farben
- auf allen Grafikkarten verfügbar
- aktiv in der Boot-Phase eines PC sowie im Reperatur- und Notbetrieb

- über das BIOS der Grafikkarte (oder des PC, aber seltener), zB INT 10H
  pro: einfach zu programmieren
  contra: langsam
- direkter Zugriff auf den Bildspeicher (Planes) durch den Grafikcontroller
  pro: schnell
  contra: aufwändige Programmierung

## Rendering

Bilderzeugung am Computer aus geometrischen Objekten.

1. Application
   - Interaktion mit Anwender
   - Kollisionsberechnung
   - Generierung der Rendering-Primitiven
   - "Culling": Zugriff auf vorgefertigte Graphikobjekte
2. Geometry
   - 2.1. Transformation [Modell- und Sichttransformation (Kameraperspektive) in 3D-Weltkoordinaten]
   - 2.2. Lighting [Beleuchtungsberechnung (Lichteffekte, Reflexionen)
   - 2.3. Projection [Transformation in 3D-Kamerakoordinaten]
   - 2.4. Clipping [Abtrennen aller Objekte außerhalb des Sichtbereichs der Kamera]
   - 2.5. Screen Mapping [Transformation in 2D-Kamerakoordinaten]
3. Rasterizer
   - Sichtbarkeitsberechnung (Z-Buffer)
   - Einrechnen der Texturen ("Tapezieren" der 2D-Polygone)
   - Setup: Umsetzen der 2D-Polygone in Pixel