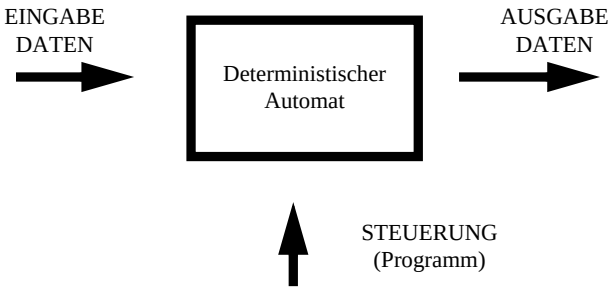
Grundlagen der Informatik für Ingenieure

Background:

3. Grundlagen der Rechnerarchitektur

3.1 Einfaches Rechnermodell

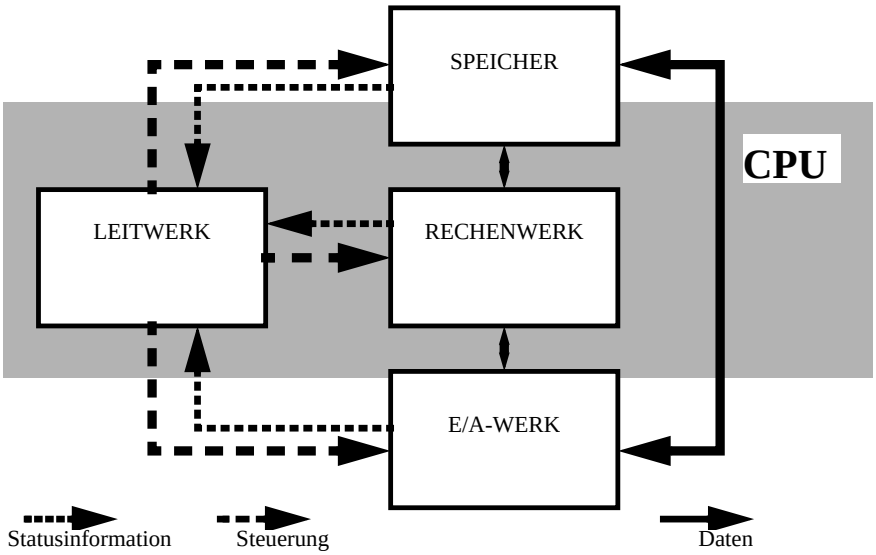
- Blackbox



- Ein Eingabedatenstrom wird durch den Automaten in einen Ausgabedatenstrom umgewandelt.
- Der Vorgang wird durch ein Programm gesteuert

3.1 Einfaches Rechnermodell

- von Neumann-Architektur



3.1 Einfaches Rechnermodell

- John (Johann) von Neumann, geb. 28.12.1903 in Budapest, vielseitiger Mathematiker in den USA; gest. 08.02.1957 in Washington.

- **Leitwerk**:
  - Die Hardware-Steuerlogik zur Steuerung des Rechen-, Speicher- und E/A-Werks.

- **Rechenwerk**:
  - Hardware zur Verknüpfung von Daten
    - Registersätze
    - **Funktions Einheiten**: +; - ; * ; / ; logische Bitmanipulation; ...

- Leitwerk und Rechenwerk werden zur **Central Processing Unit (CPU)** zusammengefaßt. Die CPU wird auch als **Prozessor** bezeichnet.

3.1 Einfaches Rechnermodell

◆ **Speicher (**Arbeitsspeicher):

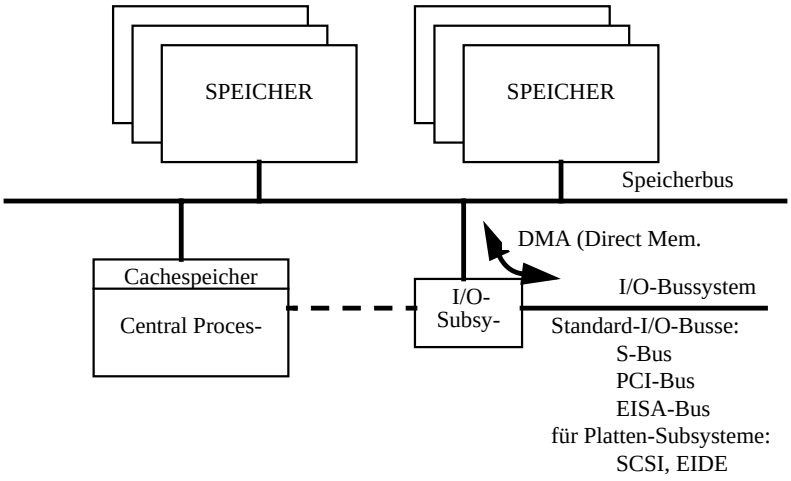
- Lineare Anordnung von (2-wertigen) Speicherzellen zur Speicherung **binär verschlüsselter** Daten für
  - die Steuerung des Systems (Programme; **Code**) und
  - Daten
- Speicherzellen **(Bits)** werden in Gruppen zusammengefaßt:
  - **Byte** (8 Bit)
  - **Worte** (16, 32, 64 Bit)
  - **Seiten** (2k-, 4k-, 8k-Byte)
  - **Segmente** (variabel oder vielfaches einer Seite falls System auch seitenorientiert)
- Speicherzellen(Bits) werden durch Speicheradressen lokalisiert.

3.1 Einfaches Rechnermodell

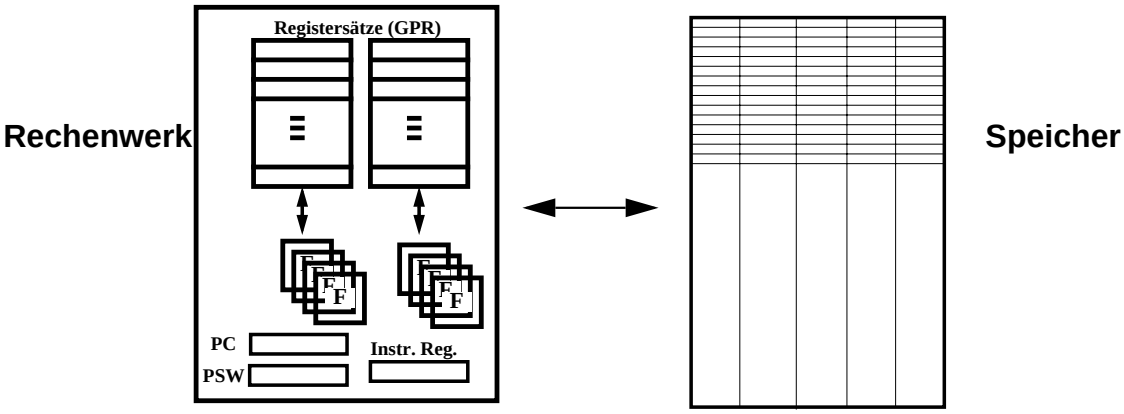
◆ **E/A-Werk:**

- Stellt die HW-Schnittstelle zur Außenwelt bereit; z. B. zu
  - Plattenlaufwerken (sekundärer, persistenter Speicher)
  - Bildschirm
  - Tastatur
  - Maus
  - Drucker
  - Scanner
  - Video
  - Audio
  - Schnittstellen zur Steuerung techn. Prozesse
  - ....

3.2 Beispiel einer Realisierung



3.3 CPU/Speichermodell



- **PC**: Programcounter (Befehlsadreßregister, Befehlszähler);
- **PSW:** Programmstatuswort
- **Instr. Reg.**: Instruction (Befehls-) Register
- **Speichereinheit**: 1 Byte = 8 Bit (in der Regel kleinste adressierbare Einheit)
- **Speicherwort**: in der Regel Standardregisterbreite (16, 32, 64 Bit)

3.3 CPU/Speichermodell

◆ Wie gewinnt ein Programm Einfluß auf die Steuerung der Hardware?

- das Register “PC” (**Programcounter = Befehlsadreßregister**) enthält die Speicheradresse des als nächstes auszuführenden **Maschinenbefehls.**
- Dieser wird aus dem Speicher in des **Befehlsregister (Instruction Register)** geladen.
- Ein “Bitmuster” (Befehlscode) steuert das **Leitwerk** und das **Rechenwerk**.
- Außerdem enthält das Register die notwendigen Informationen über die “**Orte**” (Register- oder Speicheradressen) der zu bearbeitenden Daten.

3.3 CPU/Speichermodell

◆ Die Java-Code-Zeile “***a = b + c;***”

soll von einem Prozessor abgearbeitet werden:

- Die Operation “**+**” wird auf die Inhalte der Speicherzellen “**b**” und “**c**” ausgeführt.
- Das Ergebnis der Operation “**+**” wird in dem Speicherplatz mit der symbolischen Bezeichnung “**a**” abgespeichert.

3.3 CPU/Speichermodell

◆ Was ist zu tun, damit der Prozessor genau diese Operation ausführt?

- Abbildung der symbolischen Operandenadressen in **phys. Adressen** (Datensegment):
  - Compiler erzeugt ein (relatives) **Maschinenprogramm**
  - Zur Laufzeit lädt die Speicherverwaltung des Betriebssystems das **Maschinenprogramm** in einen **freien Speicherbereich.**
  - Die **Anfangsadresse** des freien Speicherbereichs wird durch ein spezielles Register das sog. **Segmentierungsregister** (mit dem Inhalt X2) zur Verfügung.
  - Die **physikalische Adresse** erhält man durch Addition des Registerinhalts (X2) mit der relativen Adresse des Maschinenbefehls bzw. Datums (D2).

3.3 CPU/Speichermodell

- Das **Segmentierungsregister** hat neben der Bereitstellung der Anfangsadresse des Segments auch noch die Aufgabe den **Zugriffsschutz** und die zulässigen **Zugriffsmodi** sicherzustellen:
  - **Zugriffsmodi**:
    - Lesen
    - Lesen/Schreiben,
    - Ausführen (execute)
  - **Zugriffsschutz**:
    - Daten können nur durch den Programmcode des (der) “zugehörigen” Prozesse(s) (Prozessbegriff siehe Background 1) verändert werden.

3.3 CPU/Speichermodell

◆ Was ist zu tun, damit der Prozessor genau diese Operation ausführt? (cont)

• Die Abbildung der Java-Anweisung “a = b + c;” in einzelne Maschinenbefehle (Assemblersprache) könnte z. B. wie folgt aussehen:

```
(relativ einfach strukturierte klass. IBM-Assemblersprache)
Operanden vom Typ INTEGER:
        ......
        LD      R1,D2(X2)
        LD      R2,D2+4(X2)
        ADD     R1,R2 (Ergebnis in R1)
        BFC     PSW, ERR (z. B. Overflow)
        ST      R1, D2+8(X2)
        ....
ERR:    ....
```

– immer noch symbolische Form, aber prozessorspezifisch

– **Assemblierer**/Codegenerator erzeugt **Binärcode**

– **Ladeobjekt enthält Binärcode und Binärdaten**

3.4 Struktur eines Maschinenbefehls

◆ Beispiel:

```
LD      R1,D2(X2)
```

– Befehlscode: 0...7

– Registeradressen: 8...11; 12...15

– Displacement: 16...31 (ggf.: 32...64)

◆ Betriebssystem lädt Programmcode in Codesegment im Arbeitsspeicher

• Zugriffsmodus: nur “ausführen” (*execute*) erlaubt.

3.4 Struktur eines Maschinenbefehls

◆ Einfluß der Speicherwortbreite/Registerbreite:

- Wertebereiche der darstellbaren Zahlen
- Genauigkeit der realen (und komplexen) Zahlen
- Größe des adressierbaren (physikalischen/virtuellen) Speichers
- Größe der Struktureinheiten der Speicher (Segmente/Pages(Seiten))
- Größe von Dateisystemen und Länge von Dateien

3.5 Microprocessor-Technologien

◆ **CISC** - Complex Instruction Set Computer (CPU)

- Übertragung konventioneller CPU-Architekturen der 70er-Jahre auf Microchip-Technologie; microprogrammierbar;
  Komplexe (und überkommene) Strukturen führten u.a. zu Problemen bei der Steigerung der Taktraten.
- Dieser Typ von Prozessoren war die Basis der Betrachtungen in den Vorkapiteln.

◆ **RISC** - Reduced Instruction Set Computer (CPU)

- Neuansatz im Prozessorchipdesign mit den Zielen:
  - Strukturen müssen geeignet sein für hohe Taktraten
  - Ausführung eines Befehls pro Takt
  - Komplexere Befehle werden simuliert

3.5 Microprocessor-Technologien

◆ Die einsetzende technologischen Entwicklungen wie:

- Beherrschung des Produktionsprozesses mit
  - immer höhere Integrationsdichten und
  - höheren Taktraten

führte dazu,

- daß auch CISC - Prozessoren von der RISC-Technologie profitierten
- und andererseits RISC-Prozessoren immer weniger wirklich nur über einen “reduzierten” Befehlssatz verfügen.

3.5 Microprocessor-Technologien

◆ Insbesondere die **RISC**-Prozessoren entwickelten sich zu **SUPERSCALAR**-Prozessoren:

- **Functional Units** werden mehrfach auf einem Chip realisiert.
- Damit entsteht die potentielle Fähigkeit der Prozessoren, mehrere Befehle **gleichzeitig** abzuarbeiten.
- Probleme, die dabei gelöst werden müssen:
  - Wie findet der Compiler genügend geeignete Befehlssequenzen zur parallelen Abarbeitung.
- Einige typische Vertreter:
  - CISC: Motorola 68k; Intel 386, 486, 586, Pentium II?, Pentium III?
  - RISC: SPARC, Ultra-SPARC, PowerPC, HP-PA, DEC-Alpha

3.6 Leistungsmaße

(1) Taktraten (MHz):
Eigentlich nur geeignet für den Vergleich gleicher Architekturen.

(2) M(G)IPS:
Mega (Giga)-Instructions per Second
Die theoretisch erreichbare maximale Instruktionsanzahl die sich aus der "Taktrate x Anzahl der theor. möglichen Instruktionen pro Takt" berechnet.

(3) M(G)FLOPS
Mega (Giga)-Floating-Point-Instructions per Second
Die theoretisch erreichbare maximale Instruktionsanzahl die sich aus der "Taktrate x Anzahl der theor. möglichen Instruktionen pro Takt" berechnet.

◆ **Berechnungsgrundlage**: Befehle und Operanden befinden sich im Cache, die "Bandbreite" zum Speicher wird also nicht berücksichtigt.

3.6 Leistungsmaße

(4) SpecMarks:
Herstellerunabhängige Leistungsangabe auf der Basis eines standardisierten Benchmarks, der auch die Speicherbandbreite und die Fähigkeit des Compilers einschließt, einen effizienten Code zu erzeugen.

(5) LINPACK; LAPACK:
Benchmarks zur Beurteilung insbesondere der numerischen Leistungsfähigkeit eines Rechensystems

(6) weitere anwendungsspezifische Benchmarks z. B. zur Beurteilung von

- Netzwerkleistung : LADDIS (NFS),
- Datenbanksystemen: TPC (Transaktionsleistung)
- oder Grafikleistungen: Xmark, OPEN GL

3.7 Hierachische Speichertechniken

◆ Leistungssteigerung in den vergangenen 20 Jahren[1]

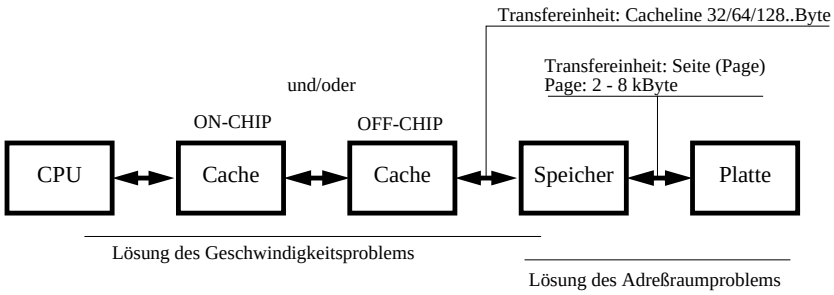
Verhältnis	2. Hälfte ’70er	Ende ’90er
Mips: ca. 1 : 1.000	1 MIPS	1000 Mips
MFlops: ca. 1: 5.000	200 kFlops	1000 MFlops
Speicherzyklen: ca. 1: 100	750 ns	7,5 ns
Plattenspeicher: ca. 1 : 8	40ms	5 ms

◆ Technologische Überwindung des Problems verschieden schneller Hardwarekomponenten:

- verschränkte Speicherbänke, Interleaving, Pipelining, Burstmode, etc
- Verbreiterung der Zugriffswege; 2, 4, 8, 16-fach
- Einführung von extremschnellen (aber teueren) Cachespeichern

1. ohne Betrachtung der sog. Mainframes und Vektorrechner

3.7 Hierachische Speichertechniken



◆ Lösungsprinzipien:

- Entkoppeln von Funktionseinheiten verschiedener Grundgeschwindigkeiten durch Puffer und Anstoß paralleler (vorausschauender) Hintergrundarbeit (Prefetch, Look Ahead);
- Hoffen auf datenlokales Verhalten des Programms.