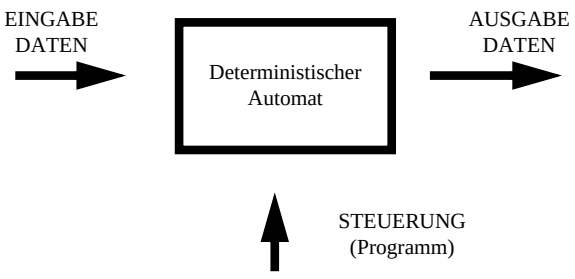
Grundlagen der Informatik für Ingenieure

Background:

3. Grundlagen der Rechnerarchitektur

3.1 Einfaches Rechnermodell

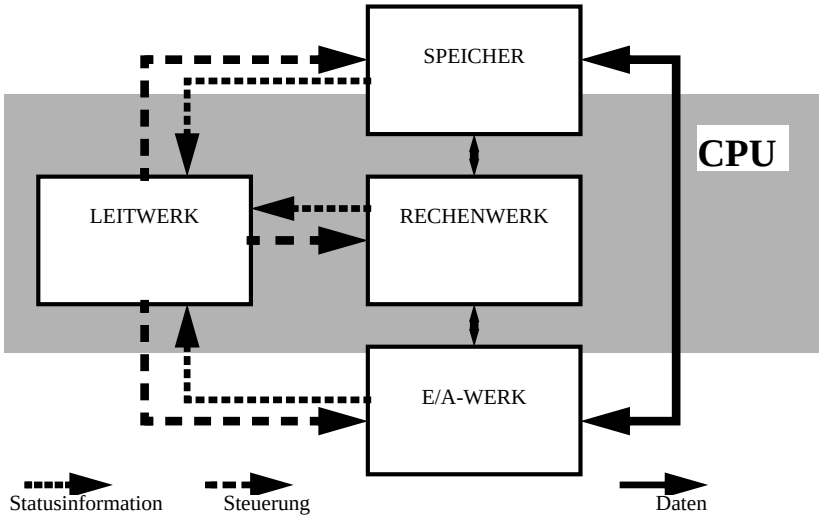
- Blackbox



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

3.1 Einfaches Rechnermodell

- von Neumann-Architektur



3.1 Einfaches Rechnermodell

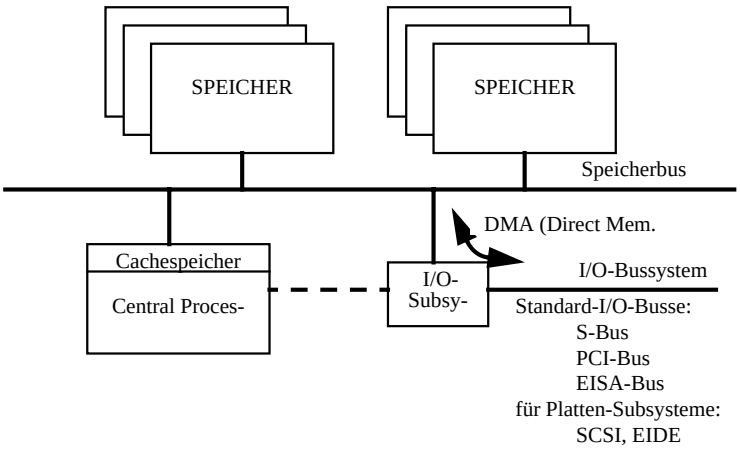
- John (Johann) von Neumann, geb. 28.12.1903 in Budapest, vielseitiger Mathematiker in den USA; gest. 08.02.1957 in Washington.
- Leitwerk:
  - Die HW-Steuerlogik zur Steuerung des Rechen-, Speicher- und E/A-Werks.
  - Je nach Konstruktion des Prozessors und Verfügbarkeit von “Chipfläche” können komplexere Operationen durch eine Abfolge von einfacheren Funktionen realisiert werden: **Taktfolge**; **Microprogrammsteuerung** (Siehe auch: Microprozessortechnologien Kap. B3.5)
  - **Status**: Rückmeldungen der Einheiten über ihren Zustand
- Rechenwerk:
  - HW 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

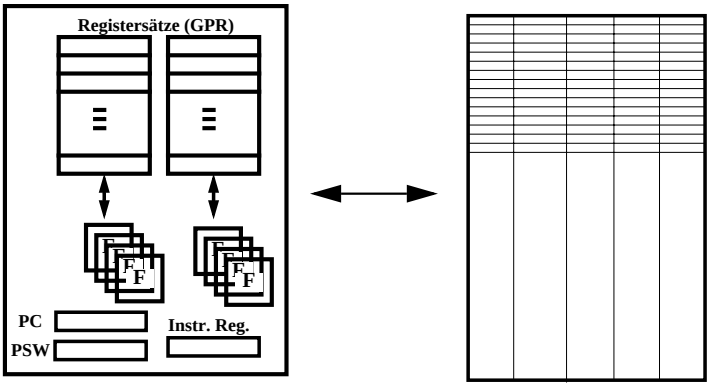
- Speicher (Arbeitsspeicher):
  - Lineare Anordnung von (2-wertigen) Speicherzellen zur Speicherung **binär verschlüsselter** Daten für
    - die Steuerung des Systems (Programme; **Code**) und
    - Daten
  - Speicherzellen**(Bits)** werden in Gruppen - **Byte(8 Bit), Worte(16, 32, 64 Bit), Seiten(2k-, 4k-, 8k-Byte), Segmente (variable, falls System auch seitenorientiert, dann vielfaches einer Seite)** - zusammengefaßt und durch Speicheradressen lokalisiert.
- E/A-Werk:
  - Stellt die HW-Schnittstelle zur Außenwelt bereit; z. B. zu
    - Plattenlaufwerken (sekundärer, persistenter Speicher)
    - Screen, Tastatur, Maus
    - Drucker, Scanner, Video, Audio
    - Schnittstellen zur Steuerung techn. Prozesse
    - .....

3.2 Beispiel...

- ..einer Realisierung



3.3 CPU/Speichermodell



PC: Programcounter (Befehlsadreßregister); PSW: Programmstatuswort
Instr. Reg.: Instruction (Befehls-) Registe

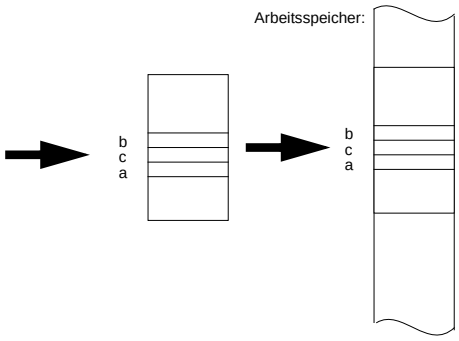
Speichereinheit: 1 Byte = 8 Bit (in der Regel kleinste adressierbare Einheit)
Speicherwort: in der Regel Standardregisterbreite (16, 32, 64 Bit)

3.3 CPU/Speichermodell

- Wie gewinnt nun ein Programm Einfluß auf die Steuerung der Hardware?
  - das Register “PC” (**Programcounter = Befehlsadreßregister**) enthält die Speicheradresse des als nächstes zur Ausführung kommenden **Maschinenbefehls.**
    Dieser wird aus dem Speicher in des **Befehlsregister (Instruction Register)** geladen.
    Das “Bitmuster” ausgezeichneter (prozessorspezifisch!) Bereiche steuert die Microoperationen des Leitwerks. 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; d. h. :
  In dem Speicherplatz mit der symbolischen Bezeichnung “a” soll das Ergebnis der Operation “+” abgespeichert werden, ausgeführt auf die Inhalte der Speicherzellen “b” und “c”.
- Was ist zu tun, damit der Prozessor genau diese Operation ausführt?



3.3 CPU/Speichermodell

- A. Abbildung der symbolischen Operandenadressen in phys. Adressen (Datensegment)
  - 1. Schritt: Compiler erzeugt ein (relatives) Speicherabbild
  - 2. Schritt: Zur Laufzeit lädt die Speicherverwaltung des Betriebssystem die Daten in einen freien Speicherbereich und stellt die Anfangsadresse durch ein spezielles Register (z. B. durch ein Segmentierungsregister(X2); prozessorabhängig) zur Verfügung.
    Die physikalische Adresse erhält man durch Addition des Registerinhalts (X2) mit der relativen Adresse des Datums (D2).
- Speichersegmente:
  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)
  - B. Abbildung der Java-Anweisung “a = b + c;” in einzelne Maschinenbefehle (Assemblersprache) könnte z. B. wie folgt aussehen:
    (relativ einfach strukturierter 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
  - Zugriffsmodus: nur “ausführen” (*execute*) erlaubt.
- Einfluß der Speicherwortbreite/Registerbreite:
  - Wertebereiche der darstellbaren Zahlen
  - Genauigkeit der realen (und complexen) 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 überkommende) 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.
  Insbesondere die RISC-Prozessoren entwickelten sich zu
- SUPERSCALAR-Prozessoren
  - Functional Units werden mehrfach auf einem Chip realisiert und z. T. geeignet miteinander verknüpft (pipelining). Damit entsteht die potentielle Fähigkeit der Prozessoren, mehrere Befehle gleichzeitig abzuarbeiten.
  - Probleme, die dabei gelöst werden müssen:
    Wie findet man (der Compiler/Codegenerator) genügend geeignete Befehlssequenzen zur parallelen Abarbeitung (Datenunabhängigkeit).
- Einige typische Vertreter:
  - CISC: Motorola 68k; Intel 386, 486, 586, Pentium II?, Pentium III?
  - RISC: SPARC, Ultra-SPARC, PowerPC, HP-PA, MIPS, 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.

- Generell gilt:
  - Berechnungsgrundlage: Befehle und Operanden befinden sich im Cache, die Bandbreite zum Speicher wird also nicht berücksichtigt.
  - Je höher die Taktrate und je höher die Zahl der Befehle pro Takt, desto geringer ist die tatsächlich erreichbare Leistungsausbeute bei "normalen" Programmsystemen; 10 - 30% sind nicht ungewöhnlich.

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

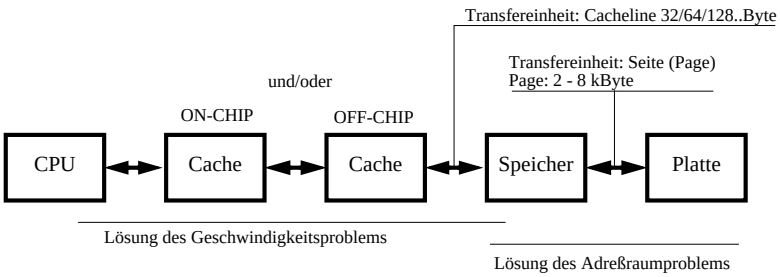
- Leistungssteigerung in den vergangenen 20 Jahren[1]

Verhältnis	2. Hälfte '70er	heute
Mips: ca. 1 : 1.000	"VAX"-MIPS = 1	1000 Mips
MFlops: ca. 1: 3.000	200 kFlops	>1000MFlops
Speicherzyklen: ca. 1: 100	750 ns	7,5 ns
Plattenspeicher: ca. 1 : 8	40ms	5 ms

- Technologische Überwindung des Problems verschieden schneller Hardware-komponenten:
  - verschränkte Speicherbänke, Interleaving, Pipelining, Burstmode, etc
  - Verbreiterung der Zugriffswege; 2, 4, 8, 16-fach
  - Einführung von extremschnellen (aber teueren) "transparenten" 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.