Grundlagen der Informatik

- Logische und mathematische Grundlagen
- Digitale Daten
- Computerprogramme als Binärdaten
  - von Neumann-Rechnerarchitektur
  - Einführung in Maschinen-Code
  - Speicherorganisation
- Betriebssysteme
- Rechnernetzwerke

Computerprogramme als Binärdaten

Entscheidender historischer Schritt auf dem Weg zum heutigen Allzweckcomputer:

- Die Folge von Instruktionen, die ein Computer als ”Programm“ ausführen soll, lässt sich wie andere Daten in Binärform kodieren.
- Bei geeigneter Binärkodierung von Programmen lässt sich die Ausführung eines Programms auf die logische Manipulation dieser Binärdaten zurückführen.
- Programme lassen sich daher nicht nur ausführen, sondern auch wie beliebige andere Daten behandeln.

→ Zum Beispiel durch andere Programme manipulieren, speichern oder übers Netz verschicken.

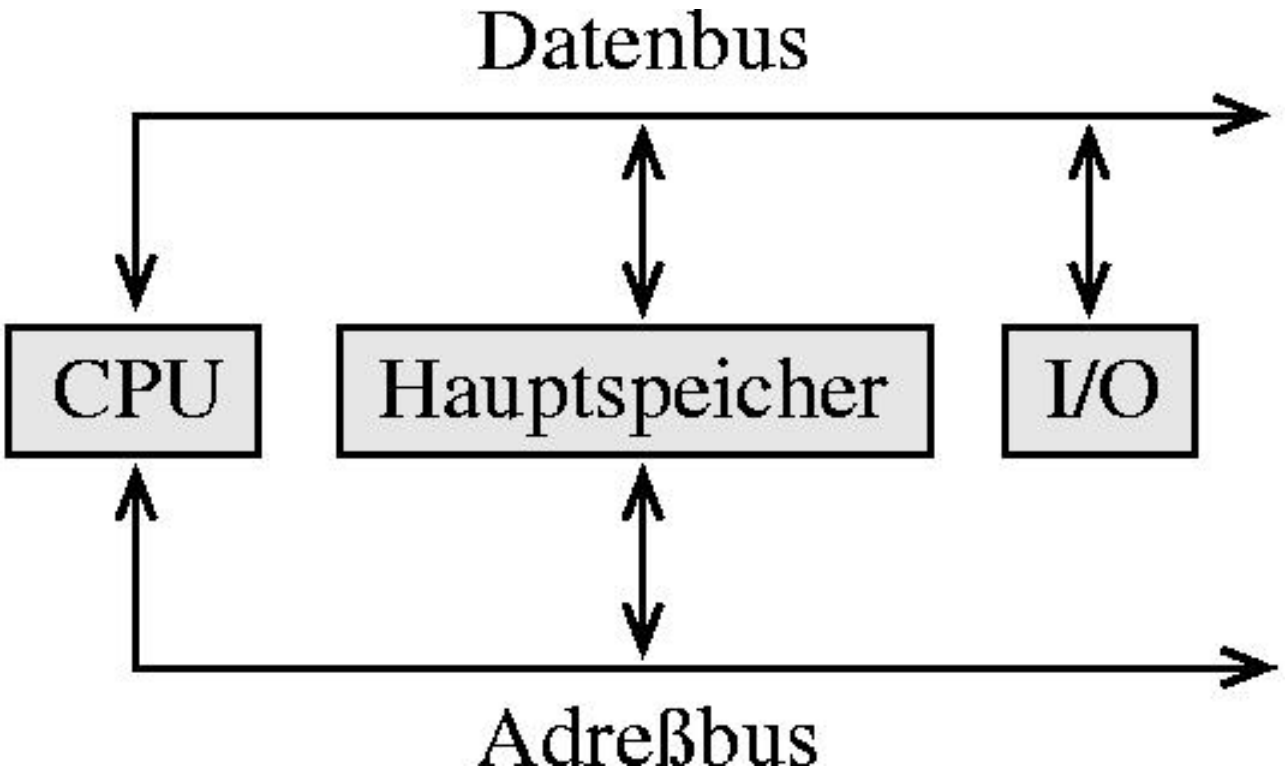
Grundlagen der Informatik

- Logische und mathematische Grundlagen
- Digitale Daten
- Computerprogramme als Binärdaten
  - ➔ von Neumann-Rechnerarchitektur
  - Einführung in Maschinen-Code
  - Speicherorganisation
- Betriebssysteme
- Rechnernetzwerke

von Neumann-Modell

- Keine Beschreibung realer Computer, sondern ein abstraktes *Muster* für die prinzipielle Organisation von Computern.
- *Grundidee*: Statt zur Lösung jedes neuen Problems eine neue Maschine zu bauen,
  - ◊ gibt es eine einzige, völlig problemunabhängige Maschine,
  - ◊ und jedes neue Lösungsverfahren wird nur noch als ein neues Programm realisiert
  - ◊ und zusammen mit seinen Daten im Speicher der Maschine unterschiedslos abgelegt.
- *Prinzipielle Aufteilung in Komponenten*:
  - ◊ Speicher,
  - ◊ Zentraleinheit (*Central Processing Unit*, *CPU*),
  - ◊ Schnittstelle zur Ein-/Ausgabe von Daten (I/O=Input/Output)
  - ◊ sowie Verbindungen dazwischen (*Busse*).

von Neumann-Modell



Hauptspeicher

- Der *Hauptspeicher* ist in eine feste Zahl von 2^n *Speicherzellen* aufgeteilt (auch *Maschinenworte* genannt).
  - Jede Speicherzelle besteht aus der gleichen Anzahl *m* von *Bits* (typisch heutzutage *m* = 16 oder *m* = 32).
  - 1 Bit = eine elementare Speichereinheit zum Speichern eines einzelnen 0=1–Wertes.
  - Alle *m* Bits einer Speicherzelle werden immer als Ganzes ausgelesen bzw. überschrieben.
- Die Speicherzellen haben fortlaufende *Adressen*: Binärzahlen der Länge *n* von 000 : : : 0 = 0 bis 111 : : : 1 = 2^n - 1
- Neben dem Hauptspeicher gibt es einzelne weitere Speicherzellen für spezielle Aufgaben (*Register*), die nicht unbedingt genau *m* Bits haben.

Busse

- Datenleitungen, die die einzelnen Elemente miteinander verbinden.
- *Breite*: Anzahl Bits, die gleichzeitig übertragen werden können.

  → Ein Bus zum Auslesen/Überschreiben von ganzen Maschinenworten hat also Breite *m*.
- *Auslesen*: Kopieren ohne zu verändern.

CPU

Besteht aus *Steuerwerk* und *Rechenwerk*.

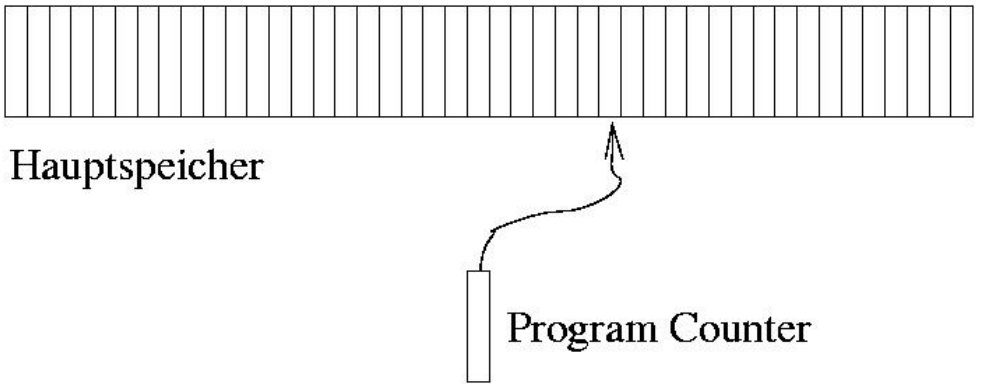
- *Steuerwerk*: Steuert die Ausführung von Programmen.
- *Rechenwerk*: Erledigt die logischen und arithmetischen Datenmanipulationen.

  → Auch *Arithmetic–Logical Unit* (*ALU*) genannt.

Ausführung von Programmen

- *Vereinfachende Annahme*: Jede Instruktion in einem Programm belegt genau ein Maschinenwort.
- Kern des Steuerwerks ist der *Programmzähler* (*Program Counter*, *PC*).

  ◊ Eine weitere Speicherzelle, aber mit *n* Bits (statt *m*).
  ◊ *Inhalt*: Adresse der jeweils als nächstes auszuführenden Instruktion.



- Eine Art Uhr gibt einen "Takt" vor, nach dem sich alle Operationen richten.

Inhalt von Instruktionen

- Im Bitmuster einer Instruktion ist kodiert,
  - wieviel an Daten die Instruktion braucht und wo sie zu finden sind,
  - wie die logischen Verschaltungen in der CPU konfiguriert werden müssen, damit genau die beabsichtigte arithmetisch–logische Operation auf den Daten ausgeführt wird, sowie
  - an welchen Hauptspeicheradressen bzw. in welchen Registern Ergebnisse abzulegen sind.

- *Also*: Der Durchlauf einer Instruktion durch die logischen Verschaltungen der CPU stößt die Öffnung der richtigen Kanäle an:
  - Von den Datenspeichern in die CPU.
  - Durch die datenmanipulierenden logischen Verschaltungen der CPU hindurch.
  - Von der CPU zurück in die Datenspeicher.

Ausführung einer Instruktion

in 5 Taktzyklen (idealisiert):

- Der Inhalt der Adresse, die im Program Counter steht, wird ausgelesen und in das Steuerwerk verbracht.
- Dieses Bitmuster durchläuft die logischen Verschaltungen des Steuerwerks und konfiguriert damit die richtigen Schaltungen
- Die Daten werden aus denjenigen Speicherstellen, deren Ausgangskanäle zur CPU dadurch geöffnet wurden, in die CPU ausgelesen.
- Die Daten durchlaufen diejenigen datenmanipulierenden logischen Verschaltungen der CPU, zu denen die Kanäle geöffnet wurden.
- Das Ergebnis wird an der Speicherstelle abgelegt, deren Eingangskanal zur CPU geöffnet wurde.

Grundlagen der Informatik

- Logische und mathematische Grundlagen
- Digitale Daten
- Computerprogramme als Binärdaten
  - von Neumann-Rechnerarchitektur
  - ➔ Einführung in Maschinen-Code
  - Speicherorganisation
- Betriebssysteme
- Rechnernetzwerke

Maschinencodes

- Gibt es wohl soviele wie es Baureihen von Computern gibt.
- Unterscheiden sich auf den ersten Blick drastisch voneinander.
- Es gibt aber entscheidende gemeinsame Merkmale.

Wichtige gemeinsame Merkmale:

- Ein paar Bits sind in jeder Instruktion reserviert zur Identifizierung der Art der Instruktion (z.B. Addition).
- Ein ausgefeilter Adressierungsmechanismus bietet verschiedene Möglichkeiten zur Angabe von Hauptspeicher- und Register-adressen.
- Spezielle Instruktionen zum bedingten oder unbedingten Überschreiben des Program Counters (”Sprungbefehle“) machen die Abarbeitung von Programmen erst wirklich flexibel.

Sprungbefehle

- Die Instruktionen eines Programms folgen im Speicher typischerweise ohne Lücke aufeinander.

  → Eine ”normale“ Instruktion wird damit beendet, dass der Program Counter auf die Adresse des nächsten Maschinenworts nach der zuletzt abgearbeiteten Instruktion gesetzt wird.

  → Bedeutet schlicht und einfach: Der Program Counter wird um 1 hochgezählt.

- *Sprungbefehl*: Eine Instruktion, deren Effekt darin besteht, den Program Counter auf einen anderen Wert zu setzen.
  - Daten eines Sprungbefehls: der neue Wert.
  - Ein Sprungbefehl sorgt also dafür, dass die Ausführung des Programms an eine andere Stelle im Maschinencode ”springt“ und dort fortfährt.

Bedingter Sprungbefehl

- Wertet zunächst einen logischen Ausdruck aus.
- Wenn der logische Ausdruck als ”wahr“ zu verstehen ist, wird gesprungen.
- Ansonsten wird der Program Counter wie bei einer ”normalen“ Instruktion hochgesetzt.
- *Zusammengefasst*: Ein bedingter Sprungbefehl ist ein Sprungbefehl,
  - ◊ der nur dann wirklich ausgeführt wird, falls eine gewisse logische Bedingung erfüllt ist,
  - ◊ und der keinerlei Effekt hat, wenn diese logische Bedingung *nicht* erfüllt ist.

Schematisches Beispiel für Übersetzung in Maschinen-Code

```
summe = 0;
for ( i=1; i<n; i++ )
   summe += i;
```

```
summe = 0; i = 1;
while ( i<n ) {
   summe = summe + i;
   i = i+1;
}
```

- Instruktion 1: Überschreibe den Inhalt der Speicherzelle namens `summe` mit dem Wert 0.
- Instruktion 2: Überschreibe den Inhalt der Speicherzelle `i` mit dem Wert 1.
- Instruktion 3: Lese die Werte in n und `i` aus, subtrahiere den zweiten vom ersten und schreibe das Ergebnis in Register X.
- Instruktion 4: Falls der Inhalt von X kleiner oder gleich 0 ist, springe nach Instruktion 8.
- Instruktion 5: Addiere die Werte von `summe` und `i` und überschreibe den Inhalt von `summe` mit dem Ergebnis.
- Instruktion 6: Erhöhe den Wert von `i` um 1.
- Instruktion 7: Springe nach Instruktion 3.
- Instruktion 8: ...

Erläuterung

- So wie auf der letzten Folie muss man sich ungefähr die Struktur von Maschinencode vorstellen.
- Vor allem die Sprungbefehle sind ziemlich genau so verwendet worden, wie es in einer Übersetzung des kleinen Programms auf der letzten Folie in Maschinencode tatsächlich aussehen würde.
- Natürlich besteht realer Maschinencode
  - nicht aus informellen deutschen Sätzen wie auf der letzten Folie,
  - sondern aus Bitmustern von gewisser Länge,
  - die aber im Prinzip genau das bewirken, was die deutschen Sätze aussagen.
- Realer Maschinencode ist aber um einiges komplexer.

Programm-Verzweigungen

- Sämtliche Sprachkonstrukte zur Steuerung des Programmverlaufs in Java oder anderen Programmiersprachen werden bei der Übersetzung in ähnlicher Weise in (bedingte) Sprungbefehle aufgelöst.

- Welche Sprachkonstrukte sind gemeint:
  - Konditionale: ”if“–Verzweigung,
  - Iteratoren: ”while“–Schleife, ”for“–Schleife
  - Methodenaufrufe, Beendigung einer Methode.

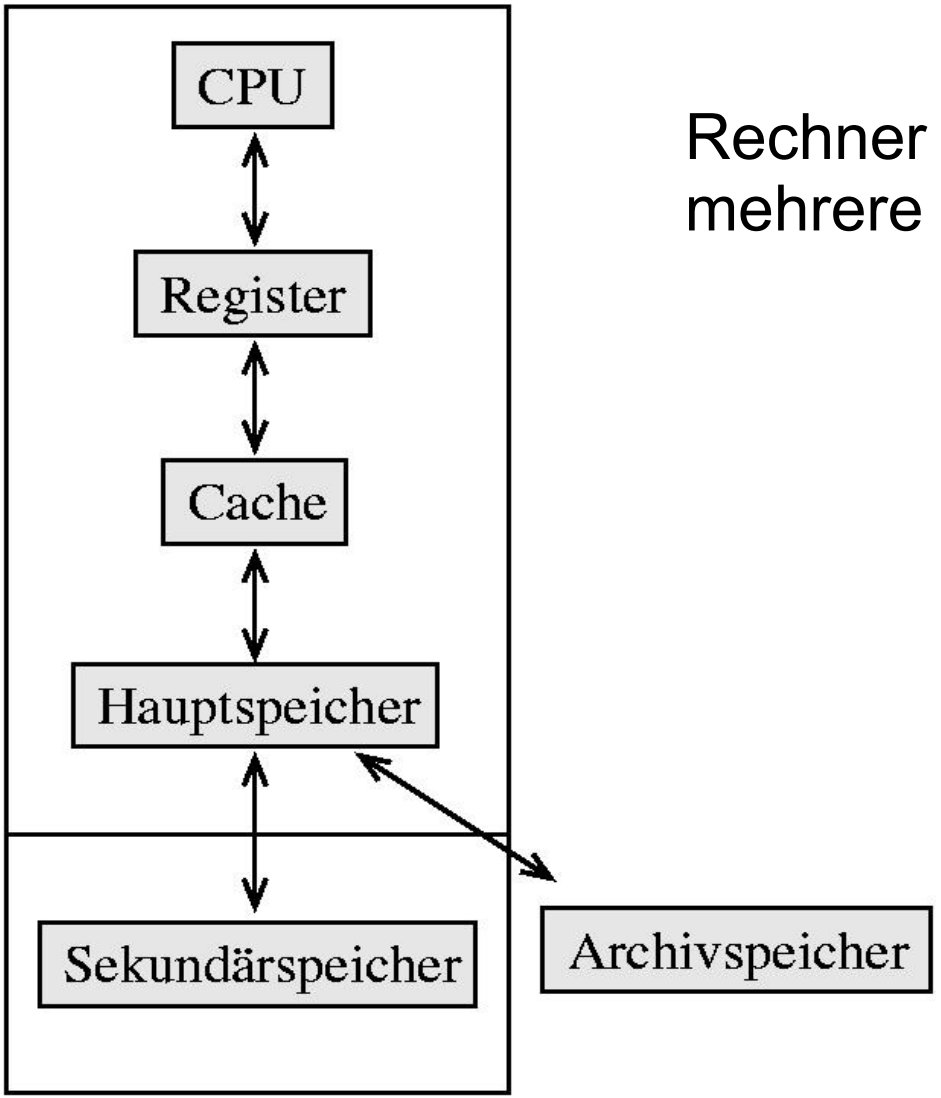
- Das Schema bietet auch eine geeignete Möglichkeit, die Bedeutung solcher Sprachkonstrukte höherer Programmier-sprachen wie Java präzise und unzweideutig zu definieren.

- Davon wird später in der Vorlesung noch ausgiebig Gebrauch gemacht werden.

Grundlagen der Informatik

- Logische und mathematische Grundlagen
- Digitale Daten
- Computerprogramme als Binärdaten
  - von Neumann-Rechnerarchitektur
  - Einführung in Maschinen-Code
  - ➔ Speicherorganisation
- Betriebssysteme
- Rechnernetzwerke

Speicherhierarchie



Rechner verfügen über mehrere Arten von Speichern

Busse in der Speicherhierarchie

- Busse gibt es immer nur zwischen unmittelbar übereinander-liegenden Speichern (plus CPU):
  - ◊ CPU ↔ Register,
  - ◊ Register ↔ Cache,
  - ◊ Cache ↔ Hauptspeicher,
  - ◊ Hauptspeicher ↔ Sekundärspeicher,
  - ◊ Hauptspeicher ↔ Archivspeicher.
- Will man zum Beispiel Daten aus dem Sekundärspeicher in die CPU laden, dann muss man sie
  - ◊ aus dem Sekundärspeicher in den Hauptspeicher,
  - ◊ aus dem Hauptspeicher in den Cache,
  - ◊ aus dem Cache in ein Register und
  - ◊ aus dem Register schließlich in die CPU laden.

Register

- Enthalten die Daten und Programmanweisungen, die unmittelbar zur Verarbeitung durch die CPU anstehen.
- Sind teilweise für Spezialaufgaben eingerichtet (z.B. Program Counter) und teilweise für die Zwischenspeicherung beliebiger Inhalte durch die CPU.

Cache

- Eine Reaktion der Hardwarebauer auf den ”Von–Neumannschen Bottleneck“ und auf die Erfahrungen mit dem typischen Lokalitätsverhalten von realen Computerprogrammen.

- ”Von–Neumannscher Bottleneck“:

  Ein Zugriff auf den Hauptspeicher benötigt bei weitem mehr Zeit als die eigentliche Ausführung einer Instruktion.

- Lokalitätsverhalten:

  Die Daten, auf denen ein Programm arbeitet, lassen sich typischerweise derart in eine lineare Reihenfolge bringen, dass zwei Maschinenworte, auf die unmittelbar nacheinander zugegriffen wird, in der Regel sehr nah beieinander liegen.

Lokalitätsverhalten

- Für Zugriffe auf auszuführende Instruktionen gilt das Lokalitätsprinzip in der Regel, da
  - Sprungbefehle relativ selten sind und
  - die Mehrzahl der Ausführungen von Sprungbefehlen den Program Counter im Allgemeinen auch nur um einen kleinen Wert verändern.
- Für Zugriffe auf Daten gilt:

  Bei vielen, insbesondere laufzeitintensiven Anwendungen wird die Laufzeit durch lineare Durchläufe durch große Datenmengen dominiert.
- *Konkrete Beispiele*:
  - Durchlauf durch Tabellen in einer Datenbank,
  - numerische Berechnungen (z.B. auf Matrizen) in graphischer Datenverarbeitung, Simulation u. ä.

Realisierung eines Cache

- Ein im Vergleich zum Hauptspeicher relativ kleiner Zwischenspeicher zwischen CPU–Registern und Hauptspeicher.
- Dafür aber schnelle Transferzeit zwischen CPU–Registern und Cache.

→ Eigentlicher Sinn und Zweck von Caches.

- Wenn die CPU den Inhalt einer Hauptspeicheradresse anfordert,
  - ◊ wird erst nachgeschaut, ob er schon im Cache zwischengespeichert ist,
  - ◊ und falls nicht, wird nicht nur dieses einzelne Maschinenwort in den Cache geladen,
  - ◊ sondern gleich eine ganze Sequenz von Maschinenworten auf einmal.
- *Konsequenz*:
  Wenn der (System-)Programmierer die Daten seines Programms ”cache–bewusst“ organisiert hat, ist nach dem Lokalitätsprinzip die Wahrscheinlichkeit sehr groß, dass die Mehrzahl der in nächster Zeit angeforderten Daten dadurch mit in den Cache geladen werden.

Sekundärspeicher

- Heutzutage in der Regel Festplatte.
- Im Gegensatz zum Hauptspeicher permanente Speicherung der Daten (auch wenn der Computer ausgeschaltet wird).
  - um Größenordnungen höhere Speicherkapazität
  - aber auch um Größenordnungen höhere Zugriffszeiten.
- *Konsequenz für Hardwarebauer*:
  Der Zugriff auf den Sekundärspeicher wird so organisiert, dass nicht einzelne Maschinenworte, sondern immer gleich größere Speicherbereiche (*Seiten*) eingelesen werden.
- *Konsequenz für* (*System*-)*Programmierer*:
  Auf einer Seite im Sekundärspeicher sollten möglichst immer Daten abgespeichert werden, die mit großer Wahrscheinlichkeit fast gleichzeitig benötigt werden (*Clustering*).

  → Wird in Datenbanksystemen u.ä. systematisch gemacht.