
— Informatik I —

Modul 5: Rechnerarchitekturen und Rechnerorganisation



Universität
Zürich^{UZH}

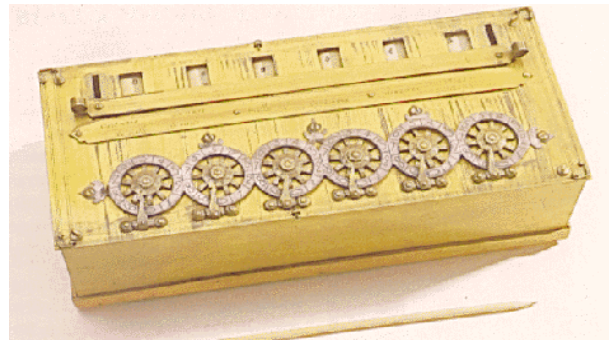


Modul 5: Rechnerarchitekturen & -organisation

- ❑ von-Neumann Architektur
- ❑ Aufbau und Funktionsweise
- ❑ Organisation
- ❑ Peripherie
- ❑ Technologieentwicklung

Überblick über die Rechnerentwicklung

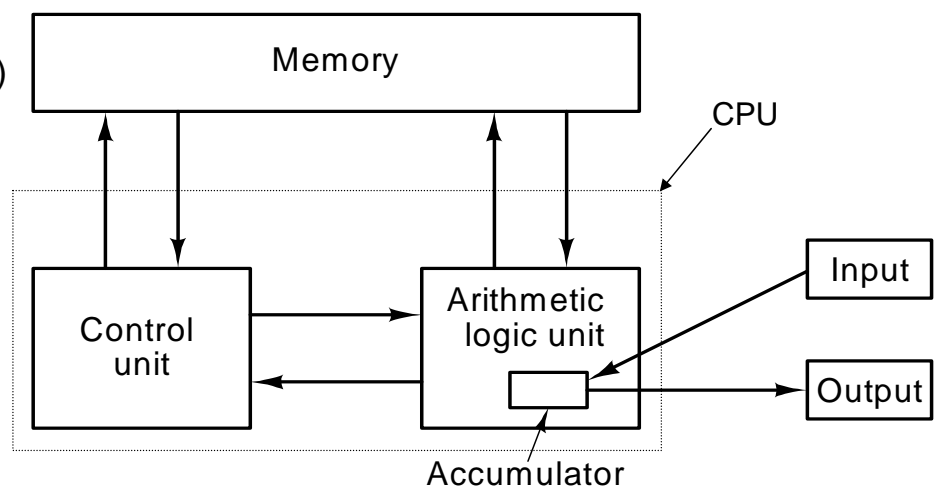
- ❑ 1642: Blaise Pascal (1623-1662)
 - Addition
- ❑ 1672: Gottfried Wilhelm Leibniz (1646-1716)
 - +, -, *, /
- ❑ 1774: Philipp Matthäus Hahn (1739-1790)
 - Zuverlässigkeit
- ❑ 1792-1871: Charles Babbage
Difference Engine und Analytical Engine
 - Planung, nie voll funktionsfähig
- ❑ 1936: Konrad Zuse (Z1, Z2, Z3, Z4)
 - Erster funktionsfähiger, programmgesteuerter Rechner (Z3)
- ❑ 1944: Howard Aiken (Harvard Mark I)
 - Addition 0,3 s, Multiplikation 6 s
- ❑ 1946: J. P. Eckert, J. W. Mauchly
ENIAC (Electronic Numerical Integrator And Computer)
 - Multiplikation 3 ms
- ❑ 1946-1952: J. Von Neumann, A. W. Burcks, H. H. Goldstine
EDVAC (Electronic Discrete Variable Automatic Computer)



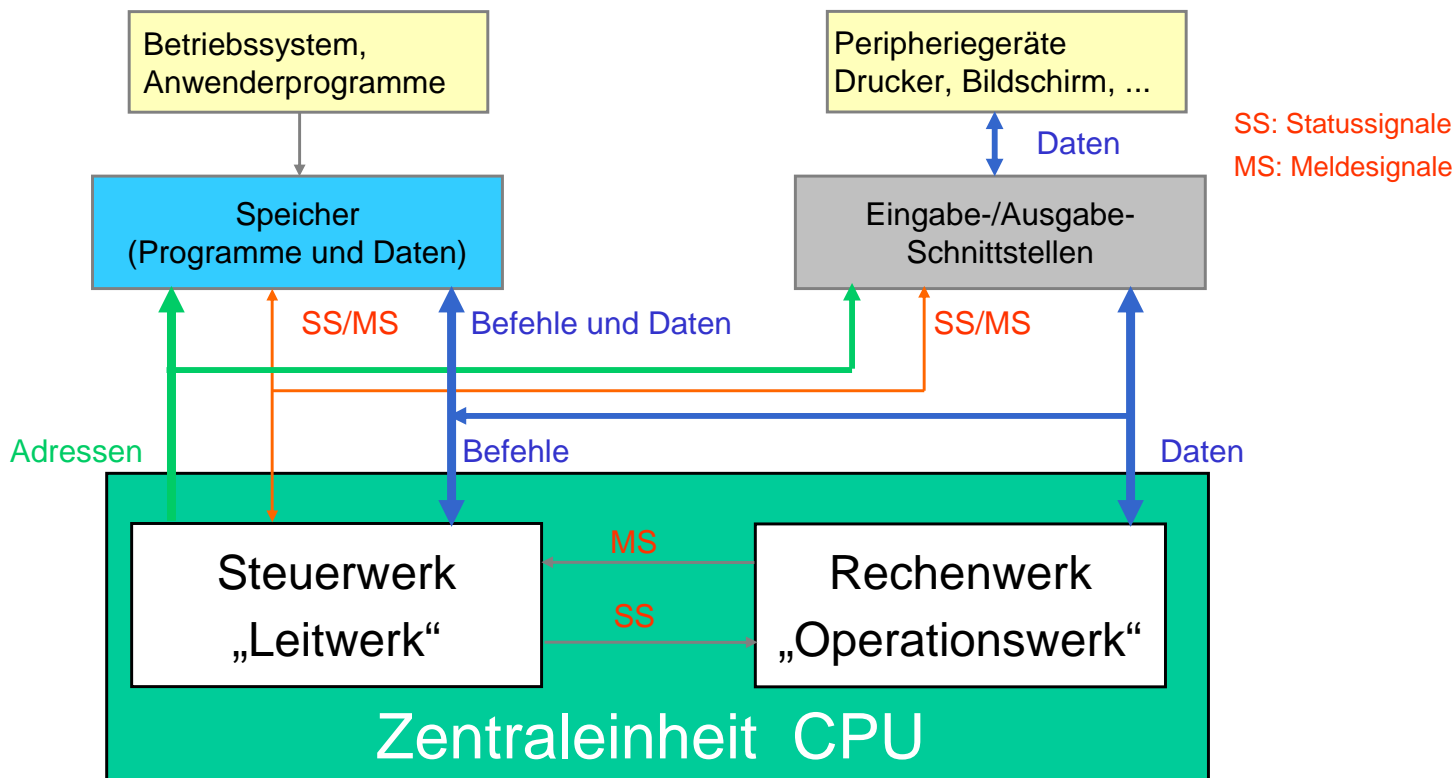
Die von-Neumann Architektur

- ❑ Die von-Neumann Architektur stellt die Basis von fast allen heutigen Hardware-Architekturen dar.
- ❑ Die Architektur umfaßt die folgenden sechs Hauptkomponenten:

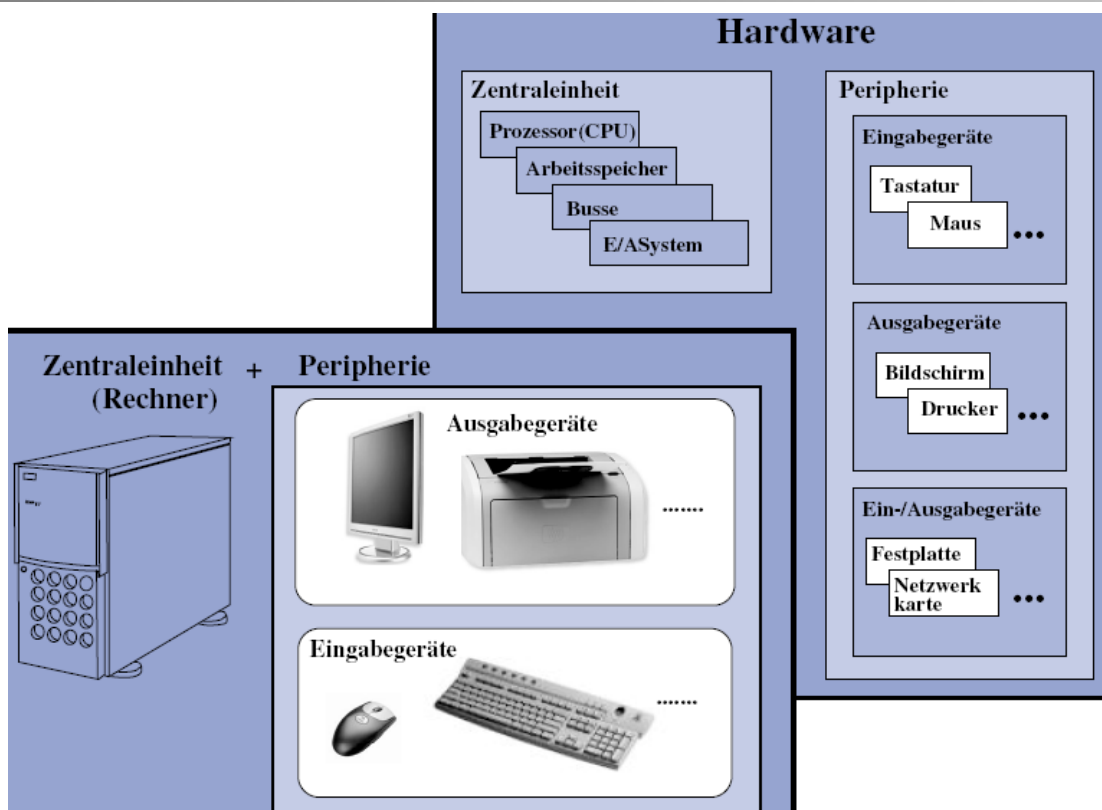
- Rechner (CPU, Central Processing Unit)
 - Steuerungseinheit (Control Unit)
 - ALU
- Speicher (Memory)
- Ein/Ausgabeeinheiten (Input/Output Units)



Digitalrechner nach dem von-Neumann Konzept (1)



Digitalrechner nach dem von-Neumann Konzept (2)



EVA-Prinzip

- ❑ Computer arbeiten nach dem EVA-Prinzip (*Eingabe, Verarbeitung und Ausgabe*).
 - **Eingabe:** Über eine Eingabeeinheit wie z.B. eine Tastatur, eine Maus, einen Memorystick gelangen Daten in den Computer.
 - **Verarbeitung** dieser Daten findet dann in der Zentraleinheit statt.
 - **Ausgabe** erfolgt über ein Ausgabegerät wie Bildschirm, Drucker, Festplatte.

EVA in den verschiedenen Computergenerationen

Computergeneration	Eingabeeinheit	Verarbeitungseinheit	Ausgabeeinheit
Röhrenrechner	Schalttafeln	Trommelspeicher	Leuchtanzeigen, Drucker mit Endlospapier
Transistorrechner	Lochkarten	Magnetkernspeicher	Drucker mit Endlospapier
IC-Rechner	Terminal mit Tastatur, Magnetband	IC-Schaltungen	Terminal, Drucker mit Endlospapier, Magnetband
Mikrocomputer	Tastatur, Maus, Festplatte usw.	Mikroprozessor	Bildschirm, Festplatte, Farb-/Laserdrucker usw.

Das EVA-Verfahren lässt sich durch die gesamte Geschichte der Computer verfolgen

Modul 5: Rechnerarchitekturen & -organisation

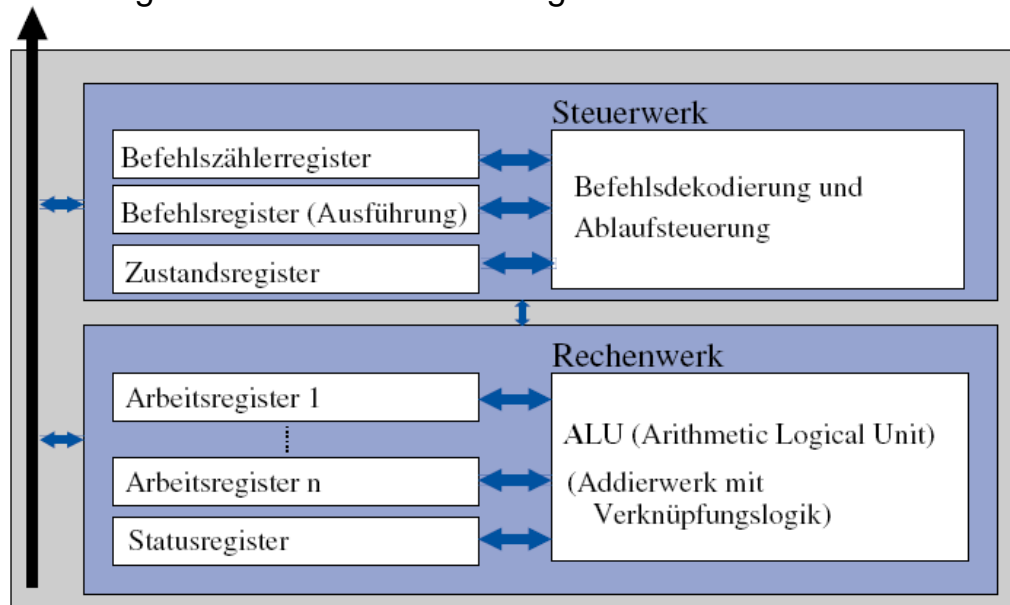
- ❑ von-Neumann Architektur
- ❑ **Aufbau und Funktionsweise**
- ❑ Organisation
- ❑ Peripherie
- ❑ Technologieentwicklung

Zentraleinheit

- Die Zentraleinheit besteht im wesentlichen aus den Komponenten der *Hauptplatine* (*Mainboard* oder *Motherboard*).
 - Der **Mikroprozessor** (*CPU = Central Processing Unit*): Ausführung der Programme, Steuerung und Verwaltung der Hardware verantwortlich.
 - Der **RAM-Arbeitsspeicher** (*RAM = Random Access Memory*): enthält Programme, die gerade ausgeführt werden, und verwendete Daten.
 - Der **ROM-Speicher** (*ROM = Read-only-Memory*): enthält meist ein Programm (*BIOS* bei IBM-PCs), das beim Einschalten die wichtigsten Hardwarekomponenten überprüft und dann das *Booten* des Betriebssystems von einem Speichermedium (Festplatte, CD) veranlaßt.
 - Die **Busse und Schnittstellen**: Kommunikation zwischen einzelnen Bestandteilen des Mainboards, zum Anschluss von Peripheriegeräten (z.B. Grafikkarten, Netzwerkkarten, Festplatten, Druckern).
 - Der **Chipsatz**: fest auf dem Mainboard untergebrachte Schaltkreise, z.B. Für die Steuerung sämtlicher Anschlüsse des Mainboards.

Mikroprozessor (1)

- Mikroprozessoren sind integrierte elektronische Schaltkreise
 - Während sie ursprünglich nur aus wenigen Transistoren bestanden, enthalten sie heute mehrere Millionen Transistoren, wobei die Integrationsdichte auf Grund des technologischen Fortschritts ständig zunimmt.



Mikroprozessor (2)

- Ein Mikroprozessor setzt sich aus folgenden Komponenten zusammen:
 - **ALU** (*Arithmetic Logical Unit=Arithmetisch-logische Einheit*): **Rechenwerk**, das mathematische Operationen und logische Verknüpfungen durchführt.
 - **Register**: spezielle Speicherplätze innerhalb des Prozessorkerns. Die ALU rechnet mit den Werten, die sich in den Arbeitsregistern befinden. Typische Prozessoren verfügen nur über relativ wenige Register, wie z. B. 2, 4, 6, 8 usw.
 - **Steuerwerk**: übernimmt mittels zweier spezieller Register (Befehlszählerregister und Befehlsregister) die Kontrolle über die Ausführung des Programmcodes und initiiert andere Steuerungsfunktionen, verwaltet auch den Stack-Zeiger.
 - *Befehlstabelle (Instruction Table)* erlaubt Maschinenbefehle eines Programms zu dekodieren.
 - Über **Busse (Datenleitungen)** ist der Prozessor mit den Komponenten verbunden:
 - *Datenbus*: Austausch von Daten mit dem Arbeitsspeicher
 - *Adressbus*: Übertragen der zugehörigen Speicheradressen
 - *Steuerbus*: Ansteuerung der Peripherie-Anschlüsse

Register

- Register
 - sind prozessorinterne Speicherplätze, die jeweils ein (binäres) Datum bestimmter Länge (z. B. 32-Bit) aufnehmen können.
 - besitzen zu anderen Prozessorkomponenten, wie Ablaufsteuerung und Verknüpfungslogik, sehr enge Verbindungen. Nur mit den Daten in den Arbeitsregistern können direkte logische Operationen durchgeführt werden.
- Registertypen
 - *Arbeitsregister* werden in Befehlen durch Namen wie z.B. D1...D7 oder EAX...EDX angesprochen. Sie können Daten (Datenregister) und je nach Prozessortyp auch Adressen (Adressregister) aufnehmen.
 - Das *Befehlszählerregister* beinhaltet immer die Adresse des nächsten auszuführenden Befehls (häufiger Name ist *Instruction Pointer = IP*).
 - Das *Befehlsregister* kann einen (binären) Maschinenbefehl aufnehmen.
 - Das *Stackregister* speichert den Prozessorstatus und den Programmzähler ab.

(Laufzeit-)Stack oder „Kellerspeicher“

- ❑ Ein besonderer Speicherbereich, der normalerweise im Arbeitsspeicher angelegt ist (software stack), ist nach dem **Kellerprinzip** (*LIFO: Last-in-first-out*) organisiert und wird Kellerspeicher genannt.

- ❑ **Funktion:**
 - Abspeichern des Prozessorstatus und des Programmzählers beim Unterprogrammaufruf und Aufruf von Unterbrechungs-Routinen
 - Parameterübergabe
 - Kurzzeitige Lagerung von Daten bei der Ausführung

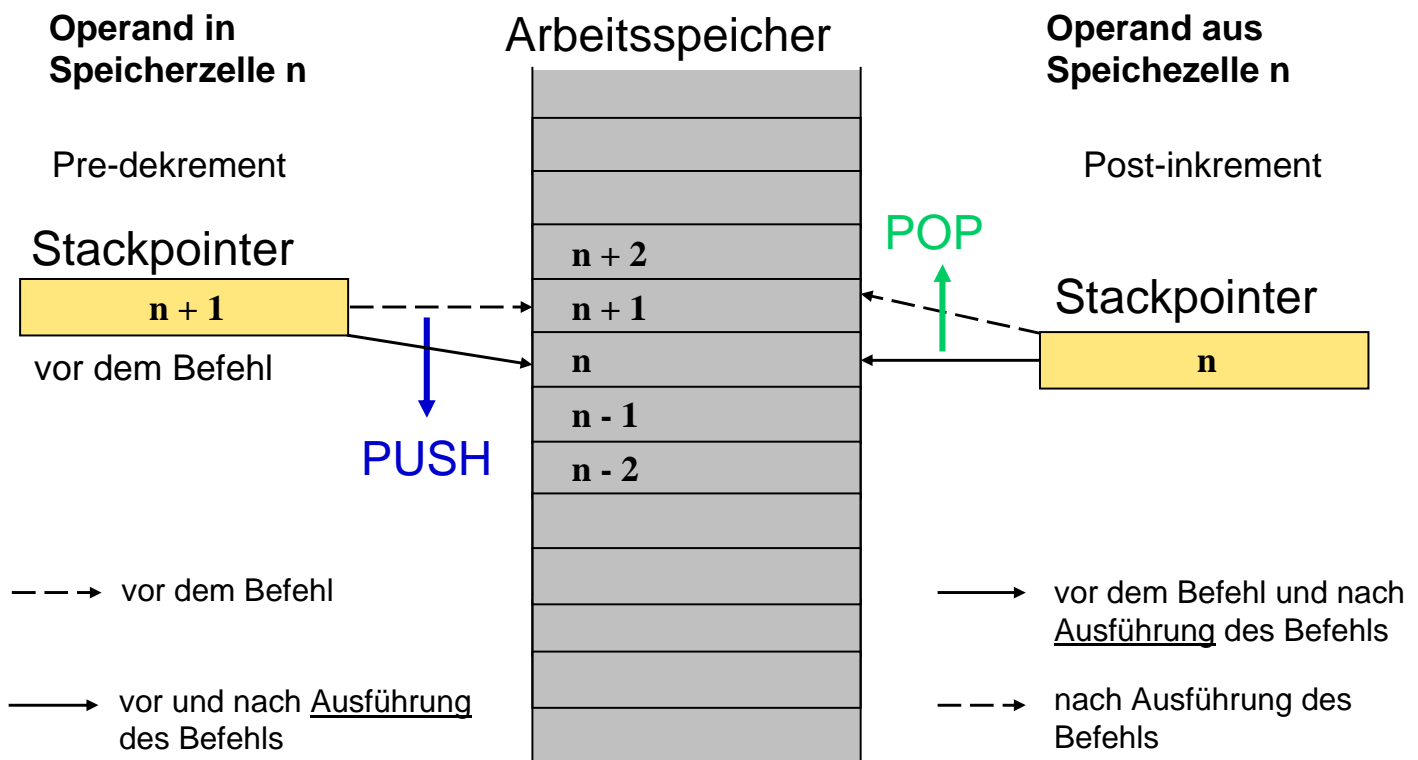
- ❑ Bei modernen Prozessoren existieren häufig mehrere getrennte Stackspeicher:
 - System Stack, User Stack, Data Stack

Hardware-Unterstützung des Stacks

- ❑ **Stackregister (Stapelzeiger, Stack Pointer SP):**
 - Enthält die Adresse des zuletzt in den Stack eingetragenen Datums

- ❑ **Spezielle Befehle zur Datenübertragung in den bzw. aus dem Stack:**
 - **PUSH:**
Inhalt eines Registers wird in den Stack übertragen
 - **POP (PULL):**
Inhalt eines Registers wird vom Stack geladen

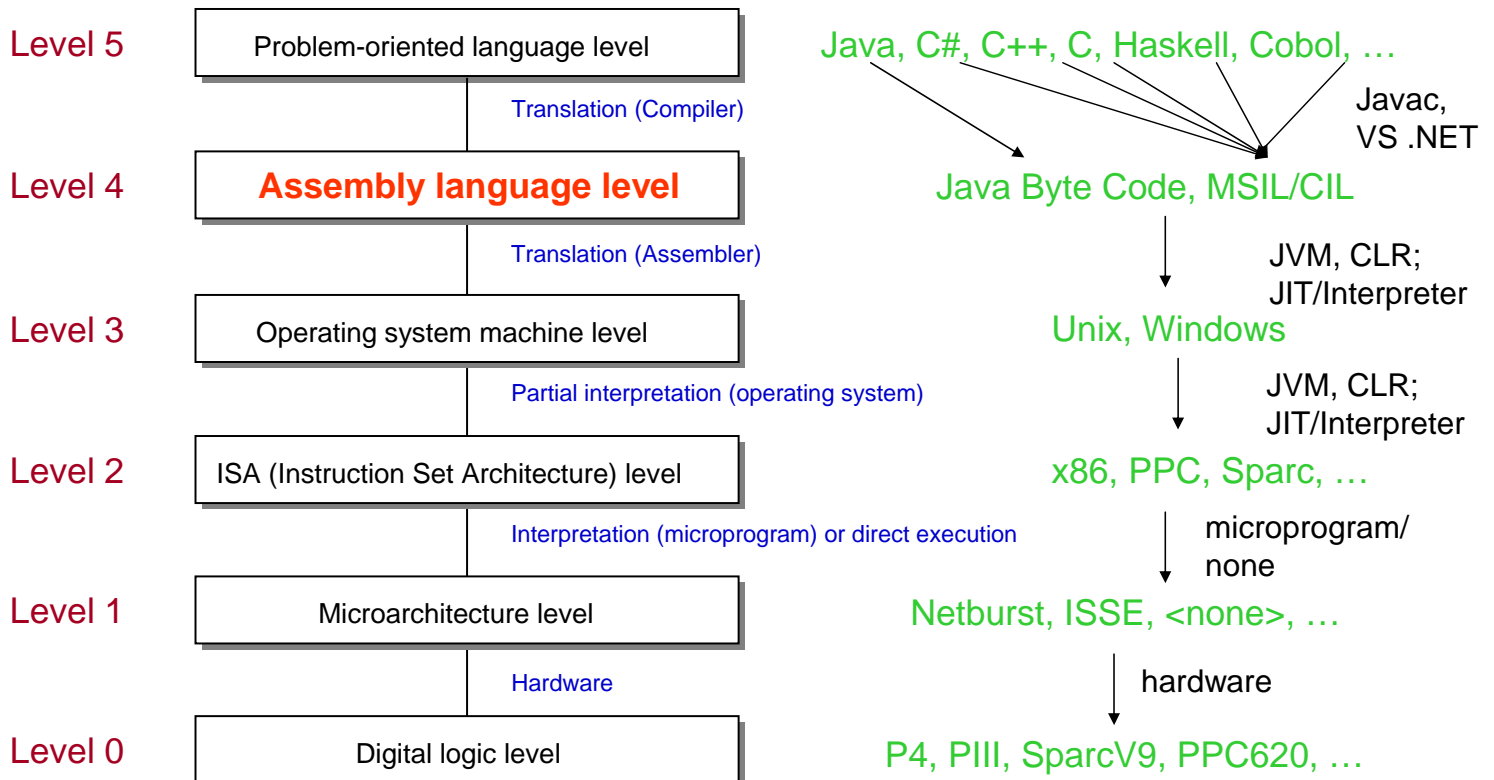
Verwaltung des Stack-Registers



Funktionsweise eines Prozessors

1. Das Befehlszählerregister des Steuerwerks enthält die Adresse des nächsten Maschinenbefehls. Die Adresse des Befehls wird über den Adressbus an den Arbeitsspeicher übermittelt.
2. Der Befehl wird aus dem Arbeitsspeicher über den Datenbus in das Befehlsregister übertragen. Mittels Dekodierlogik wird der Befehl analysiert und die Ausführung angestoßen.
3. Der Befehl wird ausgeführt; abhängig vom jeweiligen Befehl wird dabei zusätzlich das Lesen von Daten aus dem Arbeitsspeicher, die Ansteuerung von Peripherieschnittstellen, das Rechnen in der ALU oder die Durchführung eines Sprungs im Programm erforderlich. Der Status der jeweiligen Operation wird im *Statusregister (Flagregister)* angezeigt
4. Falls ein Sprung stattfand, wird das Befehlszählerregister auf die entsprechende neue Adresse gesetzt, ansonsten wird das Befehlszählerregister um 1 erhöht.
5. Der Prozessor fährt wieder mit dem 1. Schritt fort.

Sechs-Ebenen Modell der Befehle/Sprachen



Maschinenbefehle (1)

- *Binäre Maschinenbefehle* (im Befehlsregister verarbeitet) sind für den Menschen praktisch unlesbar. Daher wurde zur Erleichterung der Programmierung eine **symbolische Schreibweise für Maschinenbefehle** eingeführt.
- Symbolische Maschinenbefehle werden **Assembler-Befehle** genannt. Da fast jeder Prozessortyp, abhängig von der Dekodierlogik, unterschiedliche Maschinenbefehle besitzt, gibt es auch je Prozessortyp unterschiedliche Assemblerbefehle.
 - Beispiele symbolischer Maschinenbefehle (Assembler-Befehle):
 - **MOV BX, \$7A35**
Hole aus dem Arbeitsspeicher den Wert, der an der Adresse 7A35 steht, und lege ihn im Arbeitsregister BX ab.
 - **ADD BX, 20**
Addiere den Wert 20 zum Inhalt des Rechenregisters BX.

Maschinenbefehle (2)

- **CMP BX, 50**
Vergleiche den Wert im Register BX mit 50. Falls in BX der Wert 50 steht, wird ein bestimmtes Bit (Flag) im Zustandsregister gesetzt.
- **JE \$B7F4**
Falls der vorherige Vergleich „gleich“ ergeben hat (Flag ist im Zustandsregister gesetzt), springe zur Programmadresse B7F4. JE steht für „jump if equal“, also „Springe ..., wenn gleich“.

□ Binäre Maschinenbefehle für den Prozessor

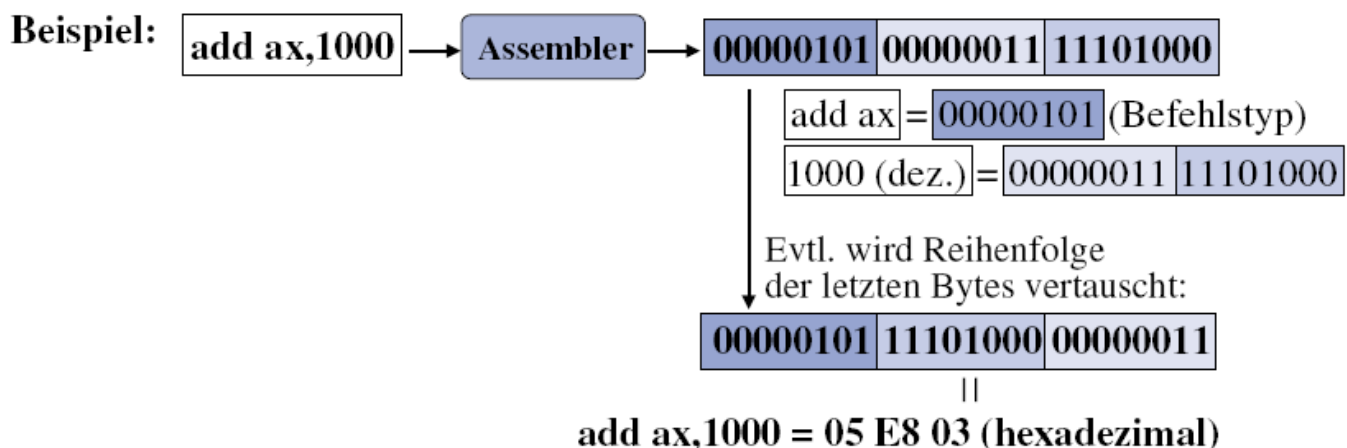
- Für einen Ablauf müssen Assembler-Befehle in binäre, für den Prozessor verständliche Maschinenbefehle übersetzt werden.
- Dafür werden für die jeweiligen Prozessoren automatische Übersetz-Programme, auch *Assembler* genannt, zur Verfügung gestellt, die aus den symbolischen Maschinenbefehlen eines Assembler-Programms die entsprechenden binären Maschinenbefehle erzeugen, die der Prozessor dann „verstehen“ kann.

Maschinenbefehle (3)

- Aufbau und Art von Maschinenbefehlen unterscheiden sich bei verschiedenen Prozessortypen und -herstellern sehr stark. Es sind aber gemeinsame Grundprinzipien vorhanden.

Aufbau eines binären Maschinenbefehls:

Operationscode	Adressierungsart	Operandenteil
----------------	------------------	---------------



Maschinenbefehle (4)

- Maschinenbefehle bestehen aus mehreren Teilen. Diese umfassen im Allgemeinen:
 - den eigentlichen Befehl (OP-CODE),
 - einen Operandenteil mit Angabe der Adressierungsart und
 - einen Operandenwert oder eine Adresse.
- Jedes Bit hat eine spezielle Bedeutung und durch die Interpretation im Steuerwerk werden die verschiedenen Reaktionen veranlasst.
- Je nach Art des Befehls können Maschinenbefehle auch verschiedene Längen und eine unterschiedliche Anzahl von Operanden haben.

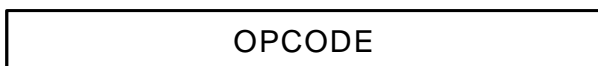
Datentypen

	8 bit	16 bit	32 bit	64 bit	128 bit
Pentium II					
□ Signed integer	X	X	X		
□ Unsigned integer	X	X	X		
□ Float			X	X	
UltraSPARC II					
□ Signed integer	X	X	X	X	
□ Unsigned integer	X	X	X	X	
□ Float			X	X	X
JVM					
□ Signed integer	X	X	X	X	
□ Unsigned integer					
□ Float			X	X	

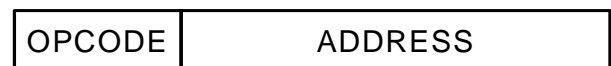
Befehlstypen

- *Arithmetische und logische Befehle* erlauben Berechnungen und logische Entscheidungen in einem Programm.
- *Sprungbefehle* ermöglichen unter Berücksichtigung entsprechender logischer oder arithmetischer Bedingungen Abweichungen vom linearen Fluß eines Programms. Mit diesen wird, zusammen mit arithmetischen und logischen Befehlen, die Vielfalt im Verhalten eines Programms und die sprichwörtliche „Intelligenz“ von heutigen Computern erreicht.
- *Transportbefehle* dienen zum Transport der Daten zwischen Prozessor, Arbeitsspeicher und Ein-/Ausgabeeinheiten.
- *Prozessorkontrollbefehle* werden zur Priorisierung von wichtigen Aufgaben und allgemein zur internen Organisation und zur Verwaltung des Prozessors benötigt.

Grundsätzliche Befehlsformate



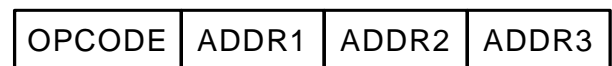
(a)



(b)



(c)



(d)

- **Beispiel:**
 $C := A + B$
 - „Zero-address“-Befehl
 - Kellerspeicherarchitekturen: `push A; push B; ADD; pop C`
 - „One-address“-Befehl
 - Der Akkumulator ist Operand und Resultat: `load A; ADD B; st C`
 - „Two-address“-Befehl
 - Ein Operand wird zum Resultat: `ADD B, A; move A, C`
 - „Three-address“-Befehl
 - $C := A + B$: `ADD C, A, B`

Beispiele für Befehlssatzarchitekturen

□ Pentium:

- Entstammt den klassischen x86 CISC Architekturen
- CISC nach außen, aber RISC nach innen!
- Andere CISC-Beispiele: Athlon und viele ältere Prozessoren (VAX, IBM, ...)

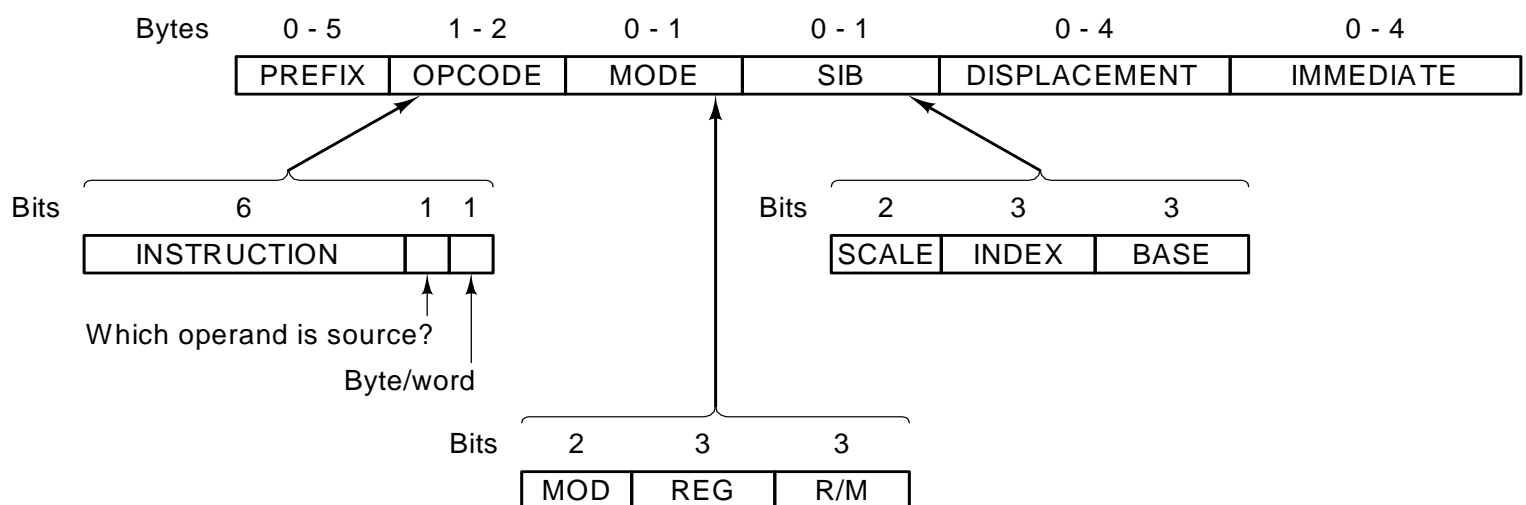
□ UltraSPARC (Ultra Scalable Processor Architecture):

- Entstammt den frühen RISC-Projekten (wie der MIPS Prozessor)
- RISC, aber erweitert
- Beispiele in SUN-Rechnern und industriellen Kontrollsystemen
- Andere RISC-Beispiele: Alpha, MIPS, Power, PowerPC

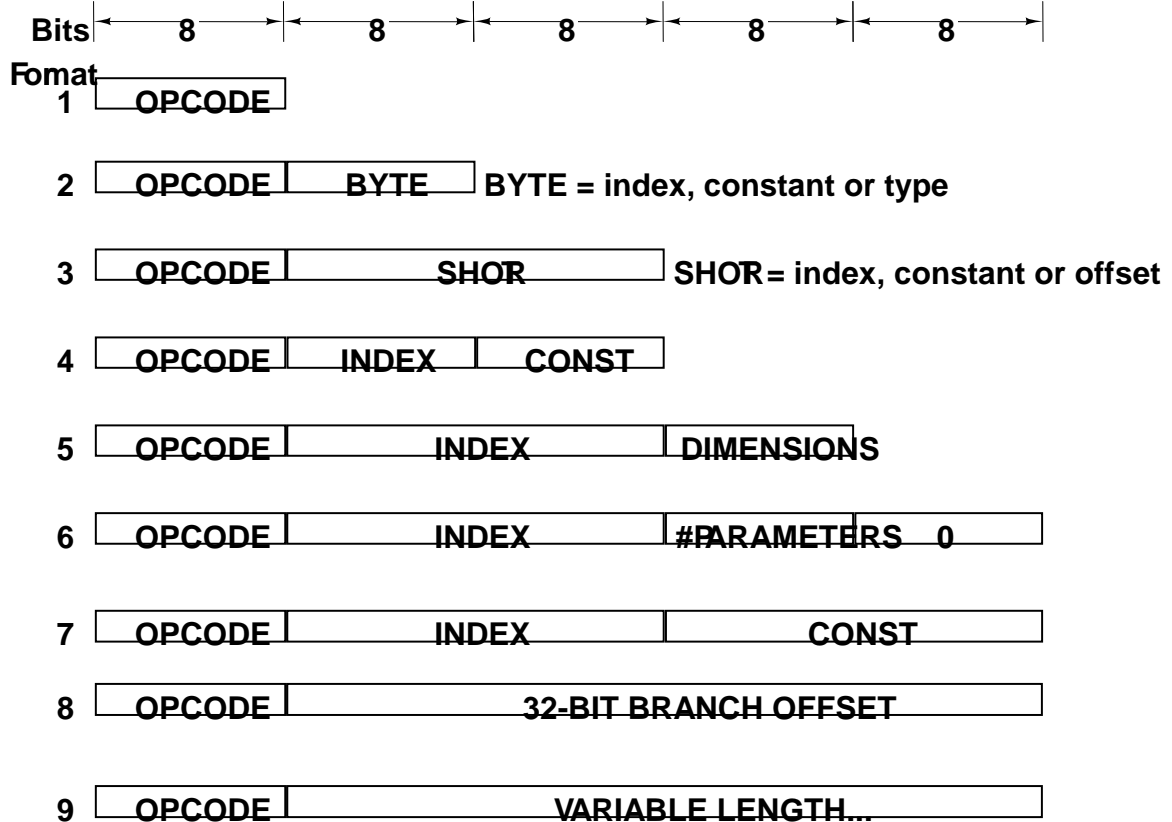
□ JVM (Java Virtual Machine):

- Entweder ein virtueller Prozessor oder reale HW (beispielsweise picoJava)
- Rechner mit einem Kellerspeicher (Operationen werden im Keller ausgeführt)
- Durch die Sprache Java stark beeinflusst.
- Andere Beispiele virtueller Maschinen: CLR, P-Code

Pentium II Befehlsformat

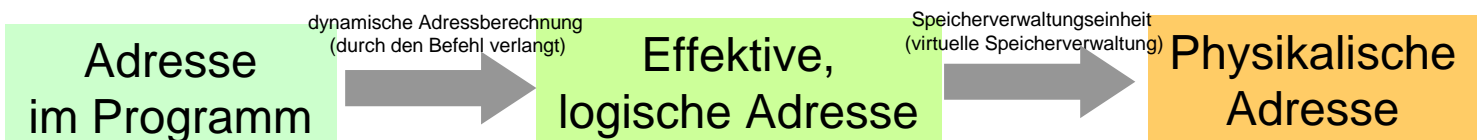


JVM Befehlsformat



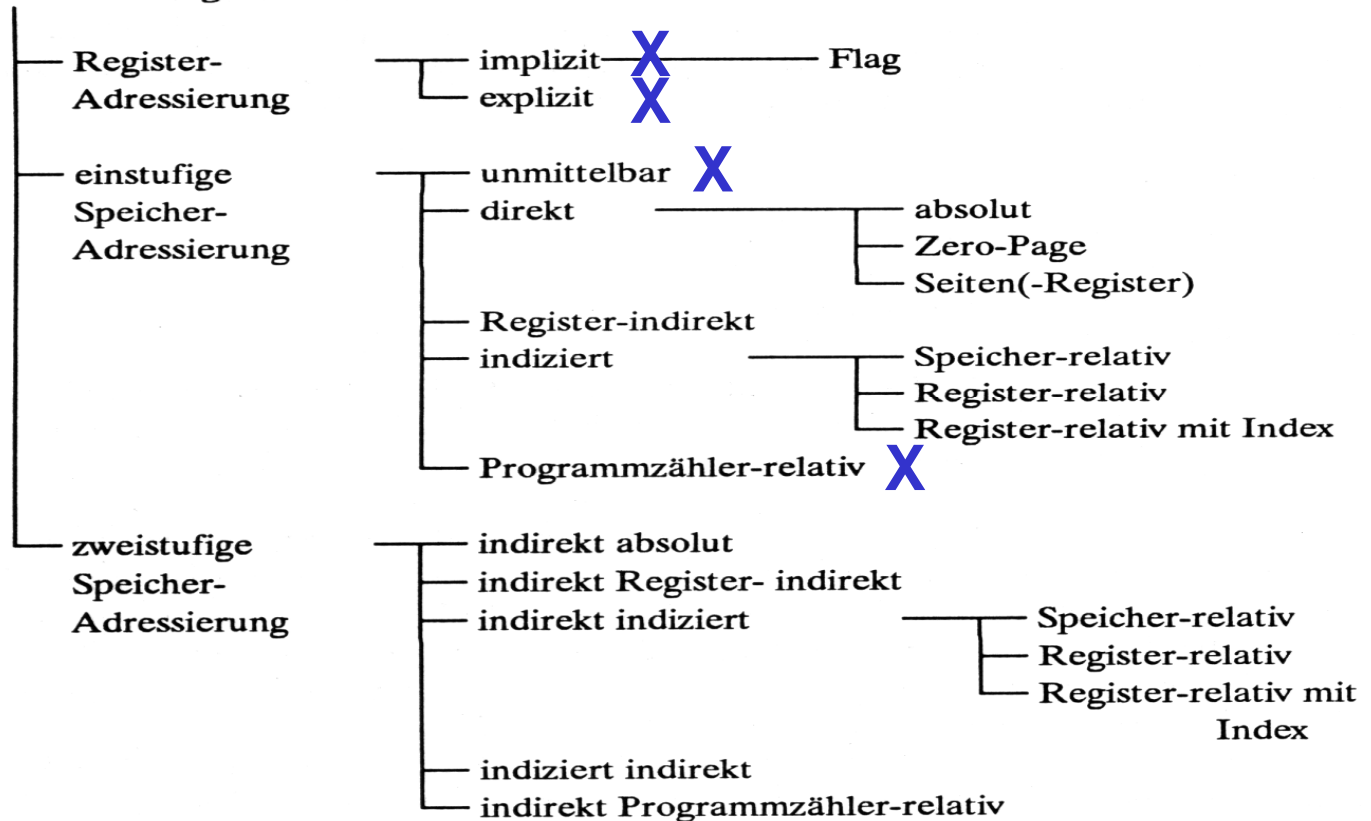
Adressierungsarten

- Die Adressierungsarten bieten verschiedene Möglichkeiten eines Prozessors, die Adresse eines Operanden oder eines Sprungziels im Speicher zu berechnen.
- Früher:
 - Adresse der Operanden und Sprungziele absolut im Befehl vorgegeben
- Nachteile:
 - Absolute Adressen müssen bereits zur Programmierzeit festgelegt werden
 - Programme sind lageabhängig im Speicher
 - Bei Tabellenzugriffen im Speicher muß die Adresse im Befehl geändert werden
 - Keine Festwertspeicher als Programmspeicher möglich
- Heute:
 - Adresse wird zur Laufzeit berechnet (dynamische Adreßberechnung)



Adressierungsarten — Überblick (1)

Adressierungsarten



Adressierungsarten — Überblick (2)

- Register-Adressierung
 - Operand steht bereits im Register
 - ➔ kein Speicherzugriff erforderlich

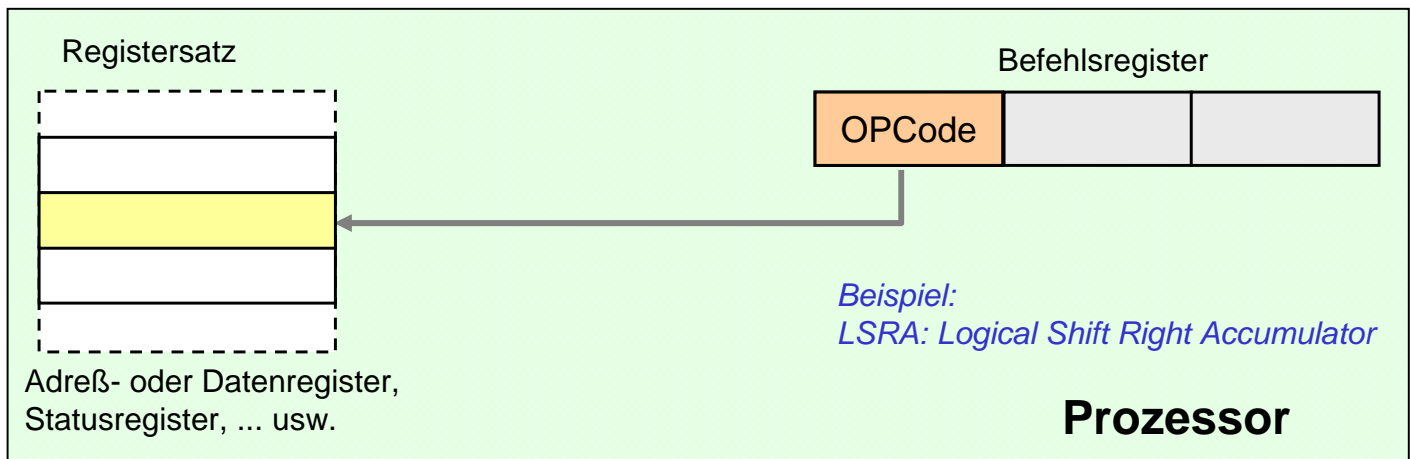
- Einstufige Speicher-Adressierung
 - Eine Adreßberechnung zur Ermittlung der effektiven Adresse notwendig, d.h. keine mehrfachen Speicherzugriffe zur Adreßermittlung

- Zweistufige Speicher-Adressierung
 - Mehrere sequentielle Adressberechnungen und Speicherzugriffe.
 - Ergebnis der ersten Berechnung liefert die Adresse einer Speicherzelle, deren Inhalt wieder eine Adresse oder ein Offset zur weiteren Berechnung ist

Implizite Adressierung

(inhärente Adressierung, implied-, inherent addressing)

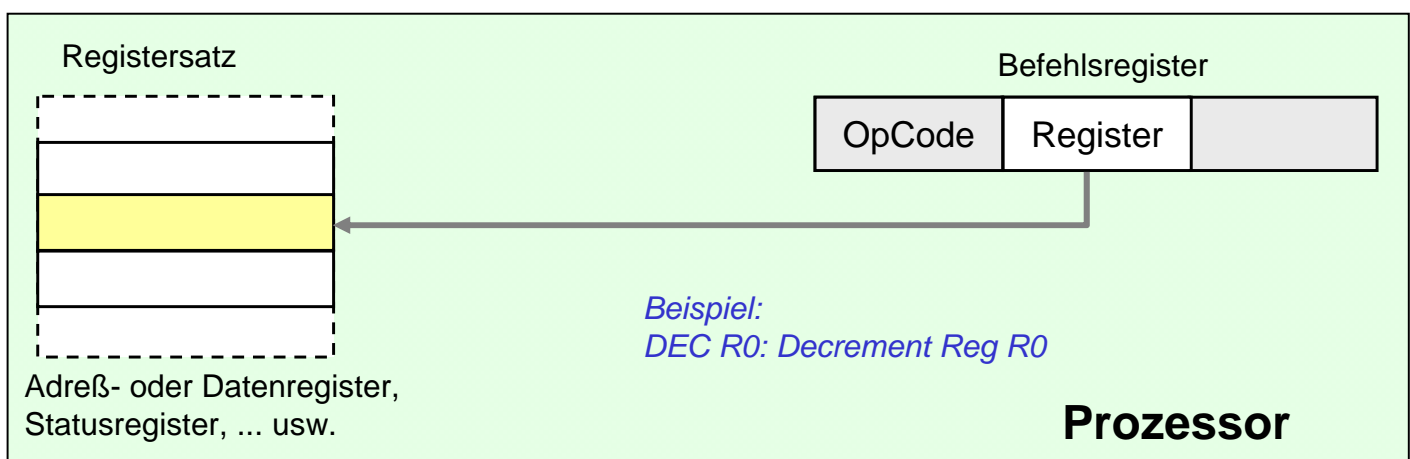
- Die Nummer, d.h. die effektive Adresse des angesprochenen Registers ist codiert im Operations-Feld (OpCode) enthalten
- **Assemblerschreibweise:** <Mnemo> A (A Akkumulator)
- **Effektive Adresse:** EA ist codiert im OpCode enthalten



Explizite Register-Adressierung

(register operand addressing)

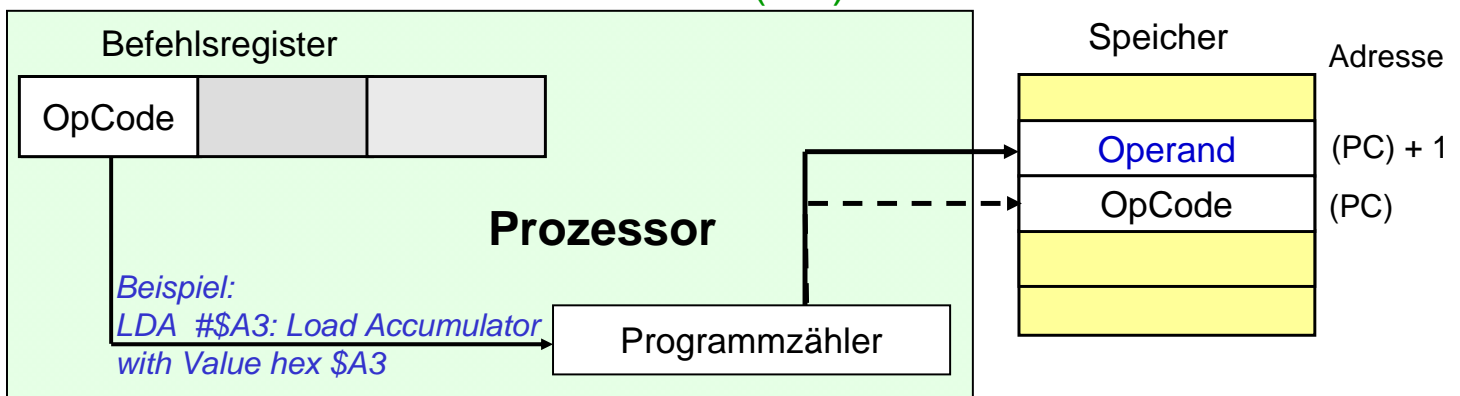
- Die Adresse (Nummer) des Registers wird im Operandenfeld des Befehls angegeben.
- **Assemblerschreibweise:** <Mnemo> Ri (Register i)
- **Effektive Adresse:** EA = i



Unmittelbare Adressierung

(*immediate addressing*)

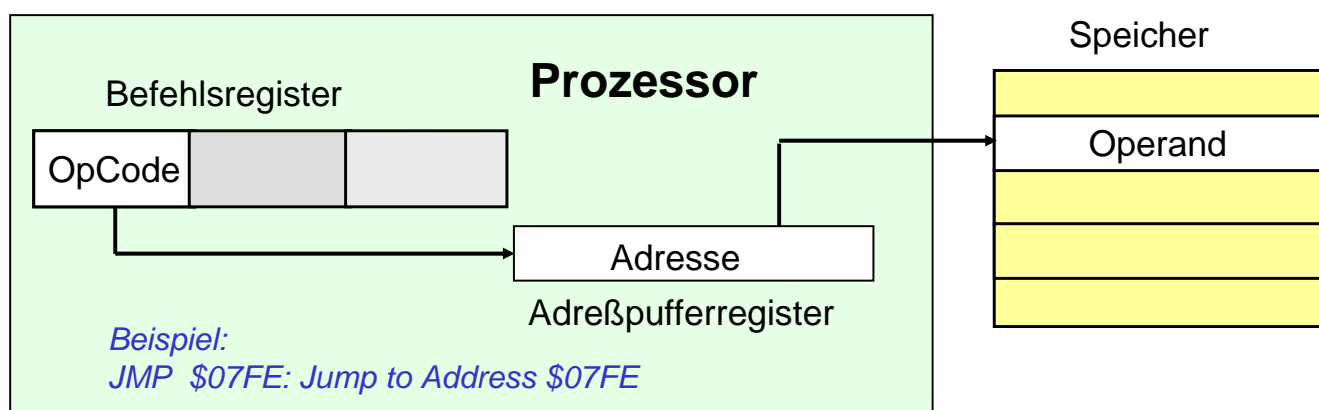
- ❑ Der Befehl enthält nicht die Adresse des Operanden oder einen Zeiger darauf, sondern den Operanden selbst.
- ❑ OpCode und Operand belegen im Speicher aufeinanderfolgende Speicherworte
- ❑ **Assemblerschreibweise:** `<Mnemo> #<Operand>`
- ❑ **Effektive Adresse:** $EA = (PC) + 1$



Absolute Adressierung

(*extended direct addressing*)

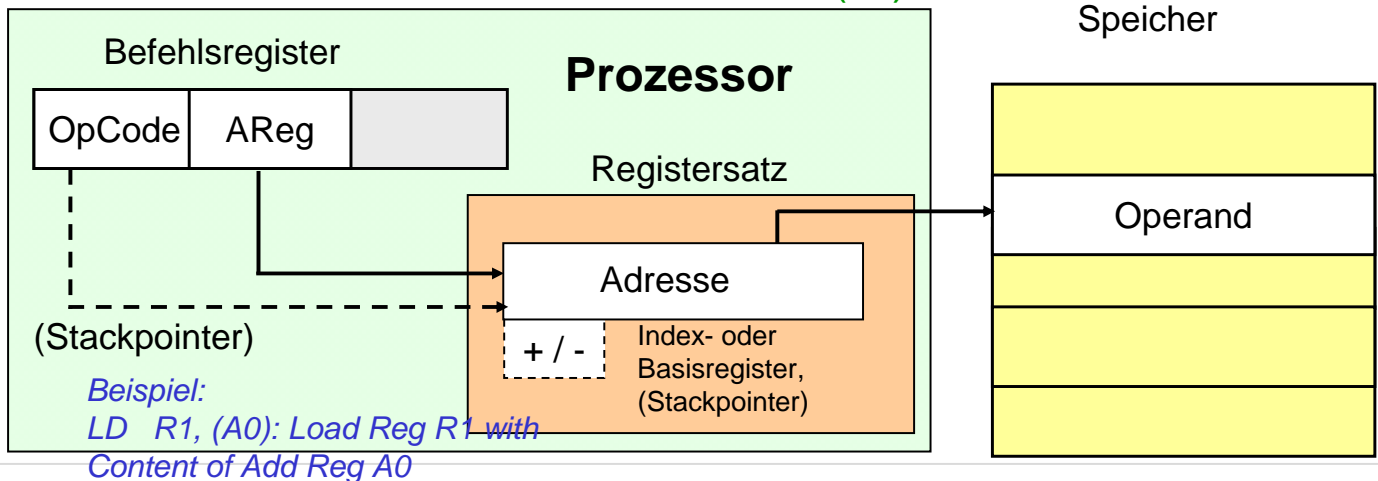
- ❑ Der Befehl enthält im Speicherwort, das dem OpCode folgt, die absolute, d.h. vollständige Adresse des Operanden im (logischen) Adressraum



Register-indirekte Adressierung (1)

(*register indirect addressing*)

- Hier enthält das durch seine Nummer im Register-Feld des OpCodes angegebene Adreßregister die Adresse des Operanden (*pointer*, "Zeiger", auch: Zeigeradressierung)
- **Assemblerschreibweise:** <Mnemo> (Ri)
- **Effektive Adresse:** EA = (Ri)



© 2012 Burkhard Stiller, 2006 Pearson Studium

M5 – 35

ifi

Register-indirekte Adressierung (2)

- Bei der im Register stehenden Adresse handelt es sich oft um die Anfangs- oder Endadresse eines Tabellenbereichs im Speicher → Registerinhalt automatisch modifizieren
- **Post-increment:**
 - Nach der Ausführung des Befehls wird der Inhalt des Registers (um 1) erhöht und zeigt danach auf die nachfolgende Speicherzelle
- **Assemblerschreibweise:** <Mnemo> (Ri)+
- **Effektive Adresse:** EA = (Ri)
 - Beispiel:
INC (R0)+ (*Inkrementiere zunächst den Inhalt des Speicherwortes, das durch das Register R0 adressiert wird, und danach den Inhalt von R0*)
- **Pre-decrement:**
 - Vor der Ausführung des Befehls wird der Inhalt des Registers erniedrigt und zeigt danach auf die vorhergehende Speicherzelle

© 2012 Burkhard Stiller, 2006 Pearson Studium

M5 – 36

ifi

Indizierte Adressierung

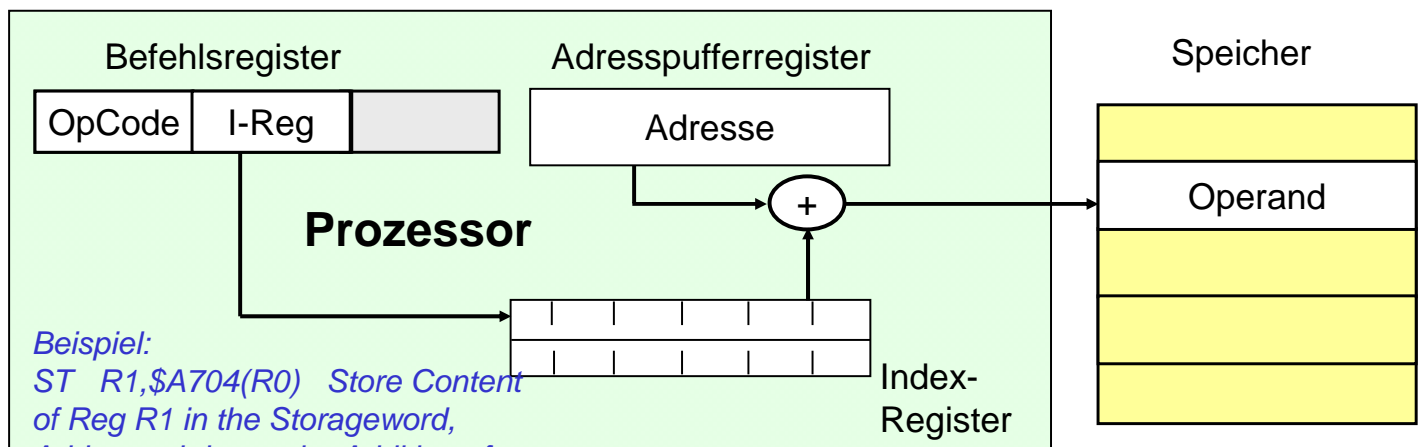
(*indexed addressing*, relative Adressierung)

- Bei ihr wird die effektive Adresse durch die Addition des Inhalts eines Registers zu einem angegebenen Basiswert berechnet. (Adreßdistanz zu einem Basiswert, Tabellenverarbeitung)
- Je nachdem, in welcher Form der Basiswert vorgegeben wird, kann man unterscheiden zwischen:
 - Speicherrelative Adressierung
 - Registerrelative Adressierung

Speicher-relative Adressierung

(*memory relative addressing*)

- Hier wird Basiswert als absolute Adresse im Befehl vorgegeben
- **Assemblerschreibweise:** <Mnemo> <Adresse>(li)
- **Effektive Adresse:** $EA = ((PC) + 1) + (li)$

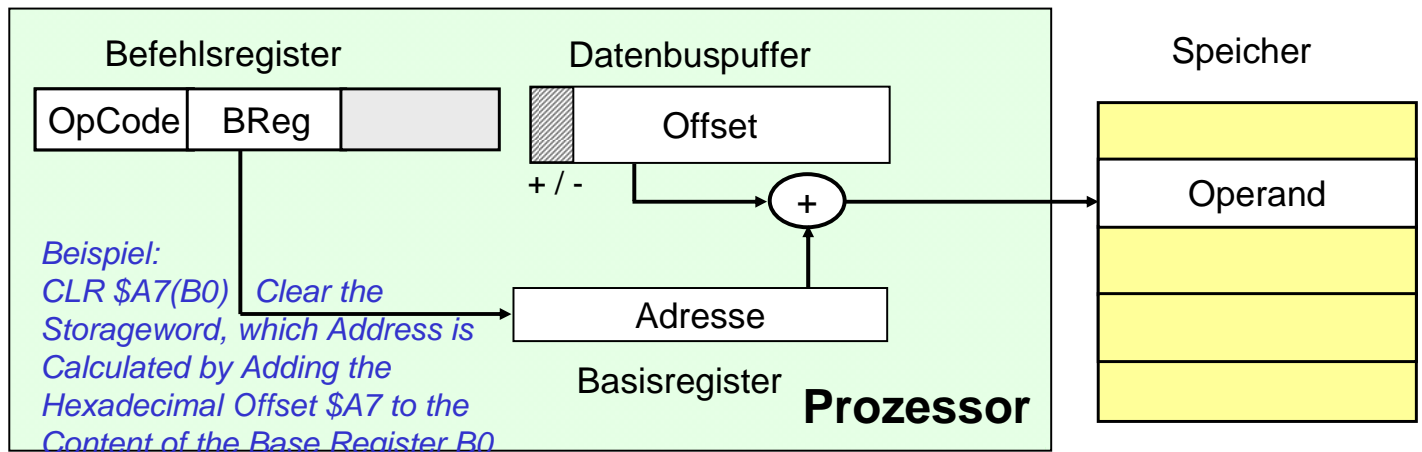


Beispiel:
`ST R1,$A704(R0)` Store Content
of Reg R1 in the Storgeword,
Addressed due to the Addition of
Content of Reg R0 to the Base of
\$A704

Register-relative Adressierung

(register relative addressing, based mode)

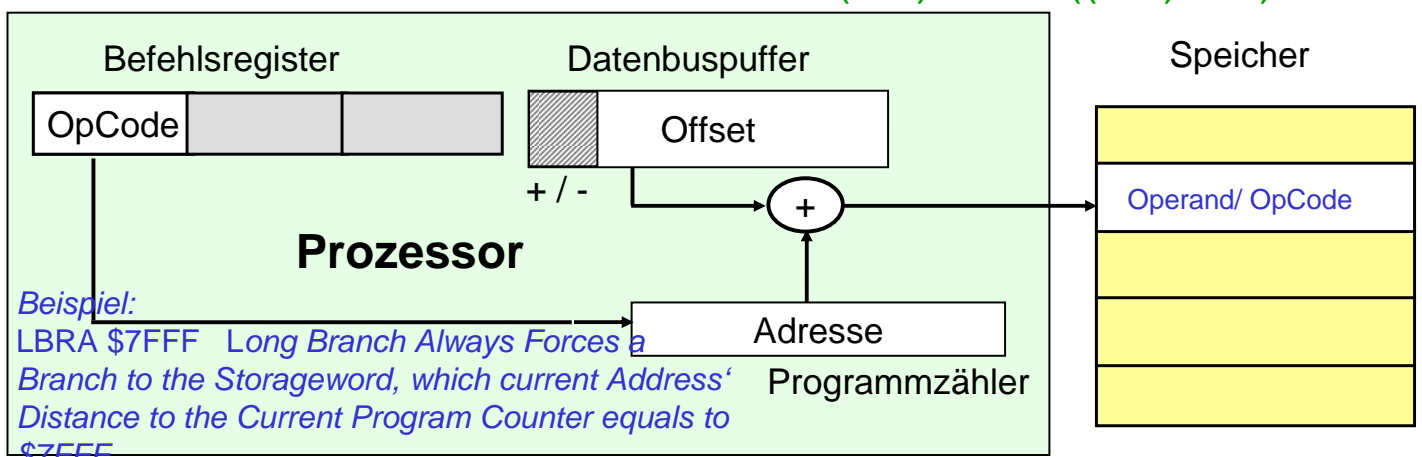
- Basiswert befindet sich in Basisregister, auf das durch das BReg-Feld im OpCode verwiesen wird. Im Befehl wird ein Offset angegeben, der zum Inhalt des Basisregisters addiert wird.
- **Assemblerschreibweise:** <Mnemo> <Offset>(Bi)
- **Effektive Adresse:** $EA = (Bi) + ((PC) + 1)$



Programmzähler-relative Adressierung

(program counter relative addressing)

- Effektive Adresse entsteht durch die Addition eines im Befehl angegebenen Offsets zum aktuellen Programmzählerstand.
- Diese Adressierungsart erlaubt es, Programme im Hauptspeicher "frei" zu verschieben
- **Assemblerschreibweise:** <Mnemo> <Offset>(PC)
- **Effektive Adresse:** $EA = (PC) + 2 + ((PC) + 1)$



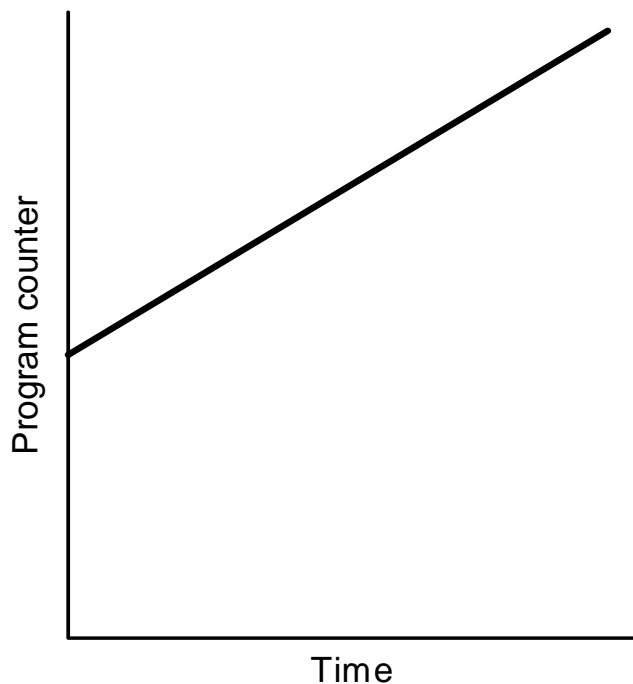
Vergleich der wichtigsten Adressierungsmodi

Modus	Pentium II	UltraSPARC II	JVM
<input type="checkbox"/> Immediate	X	X	X
<input type="checkbox"/> Direct	X		
<input type="checkbox"/> Register	X	X	
<input type="checkbox"/> Register indirect	X		
<input type="checkbox"/> Indexed	X	X	X
<input type="checkbox"/> Based-indexed		X	
<input type="checkbox"/> Stack			X

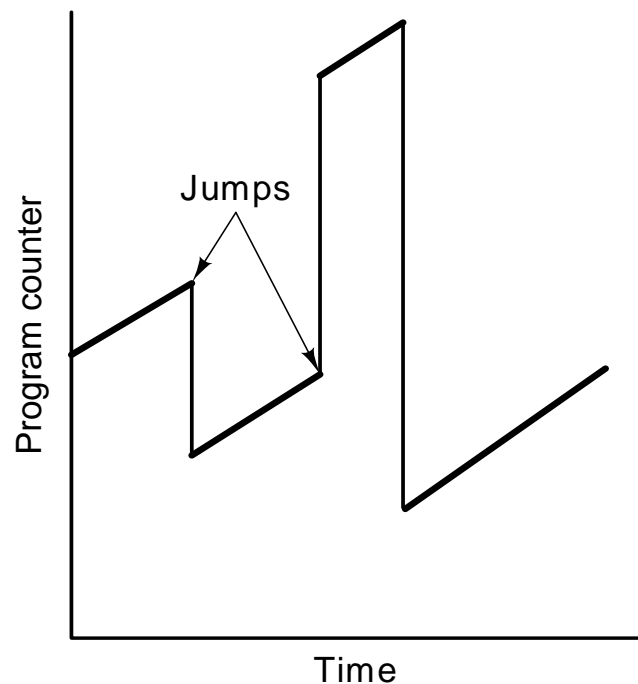
Nichtlineare Programmausführung

- Gründe für eine nichtlineare Programmausführung:
 - Sprünge und Verzweigungen (jumps, branches)
 - Prozeduraufrufe, Unterprogramme, Methodenaufrufe (Procedure calls, subroutines, method invocation)
 - Multiple Threads, parallele Prozesse, Coroutinen (Multithreading, parallel processes, co-routines)
 - Hardware-Unterbrechungen aus prozessorexternen Gründen (Hardware interrupts)
 - Software-Unterbrechungen oder Traps, aus prozessorexternen Gründen (Software interrupts)
- Nichtlineare Programmausführung stellt den Normalfall dar!

Programmausführung



Linear, without branches



With branches

Unterbrechungen (Ausnahmesituationen)

❑ Hardware-Interrupts:

- Fehler in der Hardware können zu negativen Rückmeldungen führen
- Bei Anfragen von der Hardware (wie z. B. Festplatte oder Soundkarte) unterbricht der Prozessor unter Umständen die Abarbeitung der Befehlssequenz des laufenden Prozesses und führt zunächst die angeforderte Kommunikation mit der Hardware durch, bevor er an der unterbrochenen Stelle wieder fortsetzt.

❑ Software-Interrupts:

- Ein Programm ruft ein anderes Programm oder ein Unterprogramm auf
- Anwendung des Stackspeichers zum Ablegen des Prozessorstatus
- Die Verwendung von Ctrl-C bricht laufende Prozesse ab

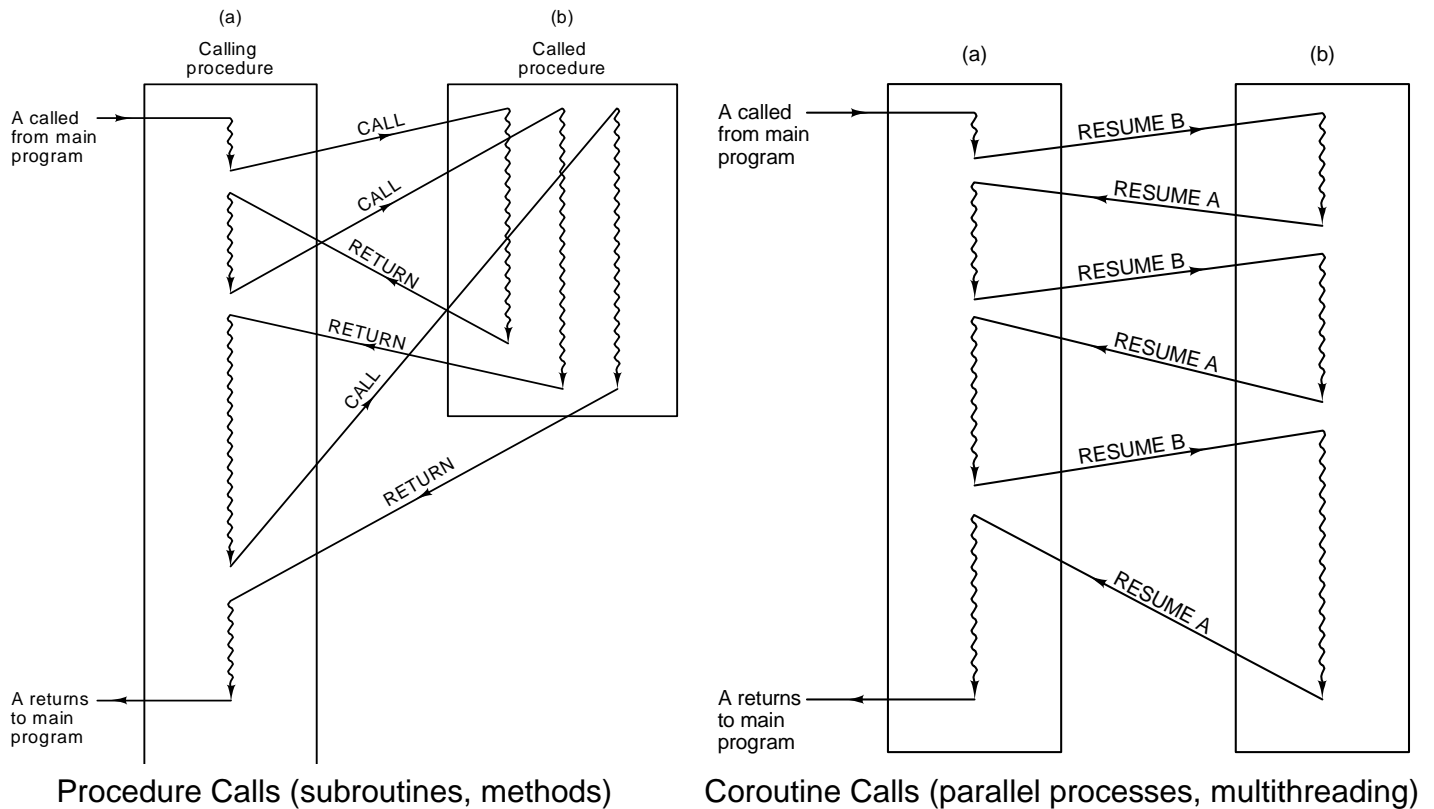
Behandlung von Ausnahmesituationen

- Während des Betriebs eines Prozessorsystems können Ausnahmesituationen (Exceptions) auftreten
- Ursachen:
 - Fehler im Betriebssystem bei der Ausführung des Anwenderprogramms oder Fehler der Hardware
 - Wunsch externer Systemkomponenten, die Aufmerksamkeit des Prozessors zu erhalten
- Eine solche Ausnahme-Situation erfordert eine **vorübergehende Unterbrechung** oder den **Abbruch** des laufenden Programms
 - Die Ausnahme-Behandlung erfolgt durch eine Ausnahmeroutine (Interrupt Service Routine)
- Auswahl und Aktivierung der Ausnahmeroutine wird durch Hardware-Komponente im Steuerwerk unterstützt:
 - Unterbrechungs-System (Interrupt System)
 - Die Ausnahmeroutine hat Ähnlichkeit mit dem Aufbau eines Unterprogramms

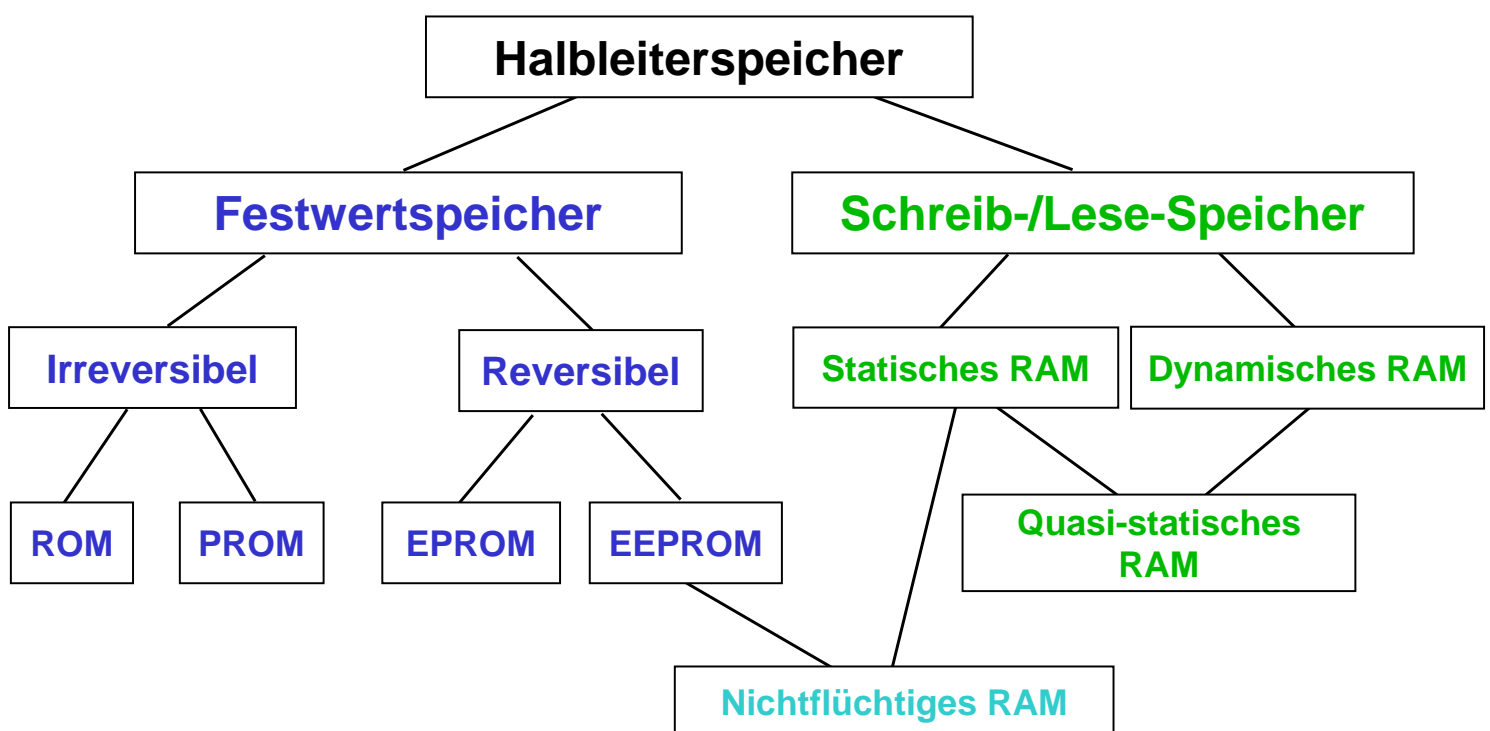
Vergleich: Ausnahmeroutine, Unterprogramm

- Wesentliche Unterschiede in **Aktivierung and Beendigung**:
 - A: *call subroutine* bei Unterprogramm
 - A: *INT*-Befehl oder Hardware-Aktivierung bei Ausnahmebehandlung
 - B: *ret*-Befehl bei Unterprogrammen (*return from subroutine*)
 - B: *reti*-Befehl bei Ausnahmebehandlung (*return from interrupt*)
- Einsprungsadresse ins Unterprogramm direkt im Programm, bei Ausnahmebehandlung über Interrupt-Tabelle
 - Unterprogrammaufruf sichert meist nur PC auf den Stack, Ausnahmebehandlungs-Aufruf meist auch das PSW
 - Unterprogrammaufrufe werden *immer* durchgeführt, die meisten Ausnahmebehandlungen werden nur aktiviert, falls das Interrupt Enable Bit im PSW gesetzt ist.

Prozedur- und Coroutinen-Aufrufe



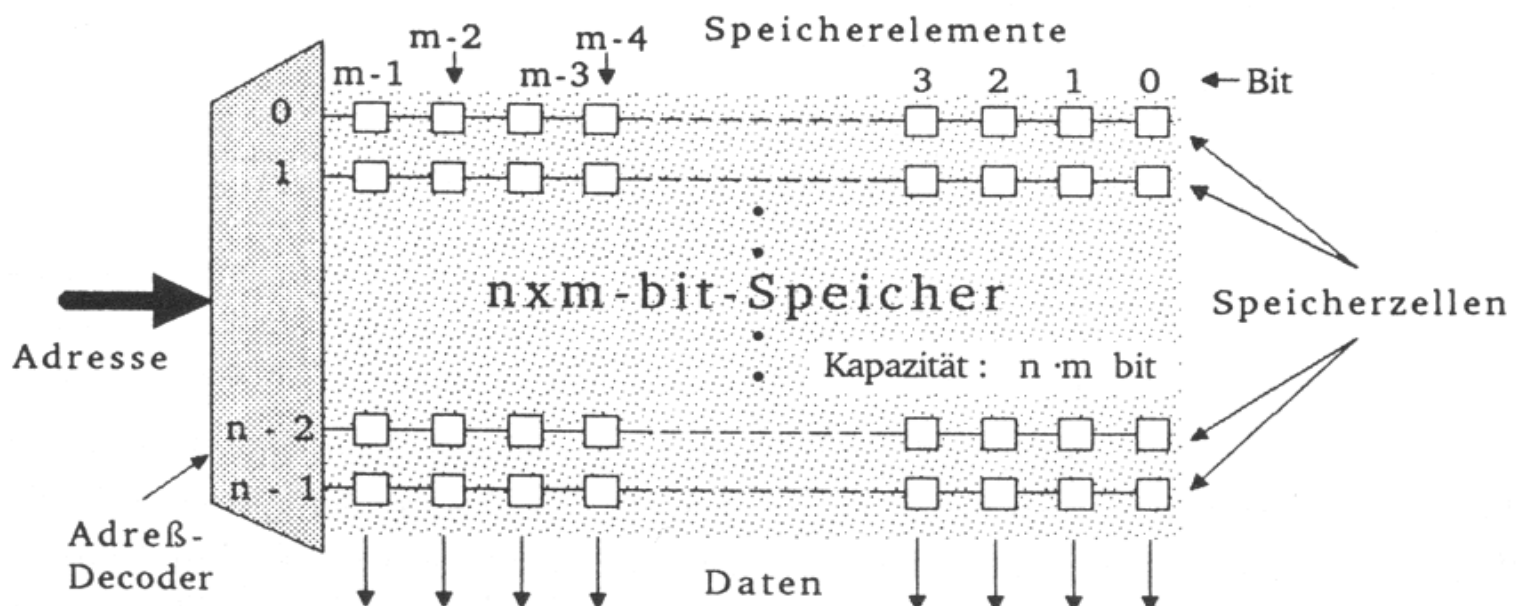
Klassifizierung von Halbleiterspeichern



Speicher

- Zwei Arten von Speicher existieren in einem Rechner:
 - Arbeitsspeicher und Peripheriespeicher (Massenspeicher)
- Arbeitsspeicher:
 - Gedächtnis (*memory*) des Rechners
 - Es werden Programme und Daten gespeichert, die “jeder Zeit sofort” zur Verfügung stehen müssen
- Permanente Ablage von Daten → Langzeitgedächtnis
 - Festwertspeicher (ROM), nicht flüchtig, z.B. OS-Kern, Systemtabellen
 - Vorübergehender Ablage von Daten → Kurzzeitgedächtnis
 - Schreib/Lesespeicher (RAM), flüchtig, z.B. Anwenderprogramme
- Technologie:
 - Heute ausschließlich Halbleiterspeicher:
 - schnell, klein, zuverlässig, preisgünstig ($\ll 1$ CHF / MByte)
 - Früher Ringkernspeicher (Magnetspeicher), Röhren, Relais:
 - Langsam, groß, teuer, z.B. $\frac{1}{4}$ MByte Speichererweiterung einer IBM/360-75 kostete 1967 1,6 Million CHF

Begriffe (1)



Begriffe (2)

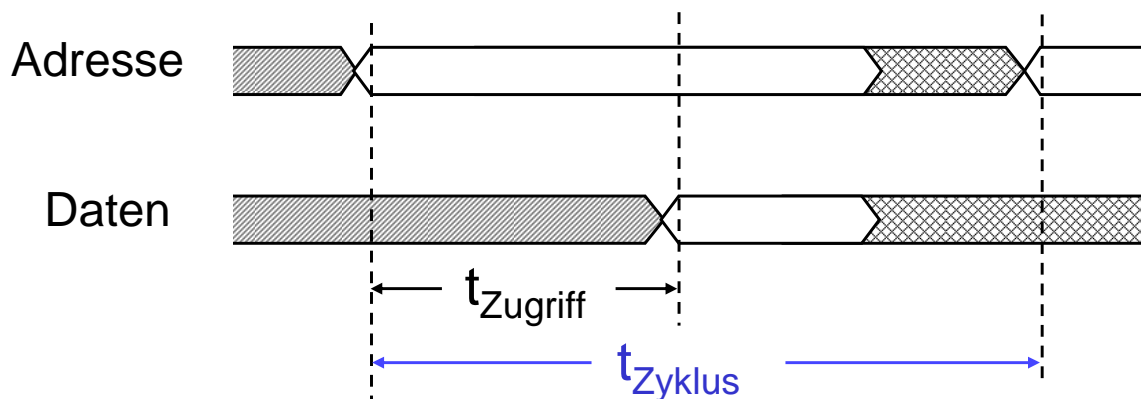
- **Speicherelement:**
 - 1 Bit Speicher
- **Speicherzelle (-platz, -stelle):**
 - Feste Anzahl von Speicherelementen, durch eine einzige Adresse ausgewählt, z.B. 8, 16, 32 Bit
- **Speicherwort:**
 - Maximale Anzahl von Speicherelementen, die in einem Buszyklus zwischen μP und Speicher übertragen werden können → Speicherwortbreite = Datenbusbreite
- **Wahlfreier Zugriff:**
 - Jede Speicherzelle kann direkt angesprochen werden, ohne vorher andere Zellen ansprechen zu müssen.
 - Die Selektion erfolgt über einen Adreßdeko­der. Die Adresse wird in einen 1-aus-n Code umgeformt (Deko­der).
- **Organisation:**
 - Organisation eines Speicherbausteins bzw. eines Speichers wird durch Anzahl n seiner Speicherzellen und die Anzahl m der Speicherelemente/Zelle definiert.

Begriffe (3)

- Angabe in der Form: $n \times m$ Bit
- Beispiel:
 - 4k x 8 Bit Speicher enthält 4096 Speicherzellen je 8 Bit
 - Aufbau aus acht 4k x 1 Bit Bausteinen oder zwei 4k x 4 Bausteinen oder ...
- **Kapazität:**
 - Informationsmenge (in Anzahl Bit), die im Speicher untergebracht werden kann:
 $n \times m$ Bit
- **Größen zur Charakterisierung der Arbeitsgeschwindigkeit eines Speicherbausteins:**
 - Zugriffszeit (*access time*):
 - Maximale Zeitdauer, die vom Anlegen einer Adresse an den Speicher bis zur Ausgabe der gewünschten Daten vergeht
 - Zykluszeit (*cycle time*):
 - Minimale Zeitdauer, die zwischen zwei hintereinander folgenden Aufschaltungen von Adressen an den Speicher vergehen muß

Zugriffszeit und Zykluszeit

- Die Zykluszeit kann erheblich länger als die Zugriffszeit sein!
- Gründe:
 - Speicherzelle muß sich nach einem Zugriff “erholen”
 - Bei einigen Speicherarten wird die Information durch das Auslesen zerstört und muß erst wieder eingeschrieben werden (refresh)
- Idealfall: Zykluszeit = Zugriffszeit
- Realität: Meist Zykluszeit > Zugriffszeit (bis zu 80%)



Speichermodultypen (1)

- Single In-line Pin Package (SIPP): **Veraltet**



SIPP-Modul mit seinen 30 pin-förmigen Anschlüssen und einer Datenbreite von 8 Bit.

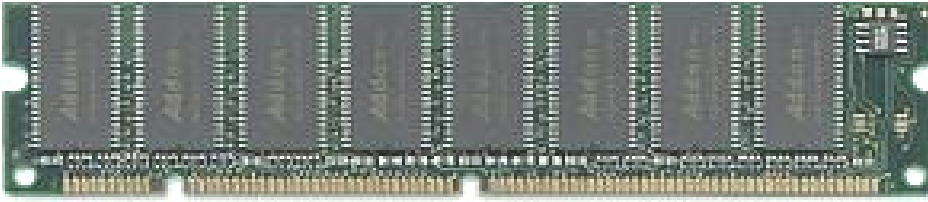
- Single In-line Memory Module (SIMM): PS/2-Module



SIMM-Module in der 30- und 72-poligen Ausführung mit Datenbreiten von 8 und 32 Bit.

Speichermodultypen (2)

□ Dual In-line Memory Module (DIMM):



Die 168-poligen DIMMs besitzen eine Datenbusbreite von 64 Bit.

□ Rambus In-line Memory Modul (RIMM)

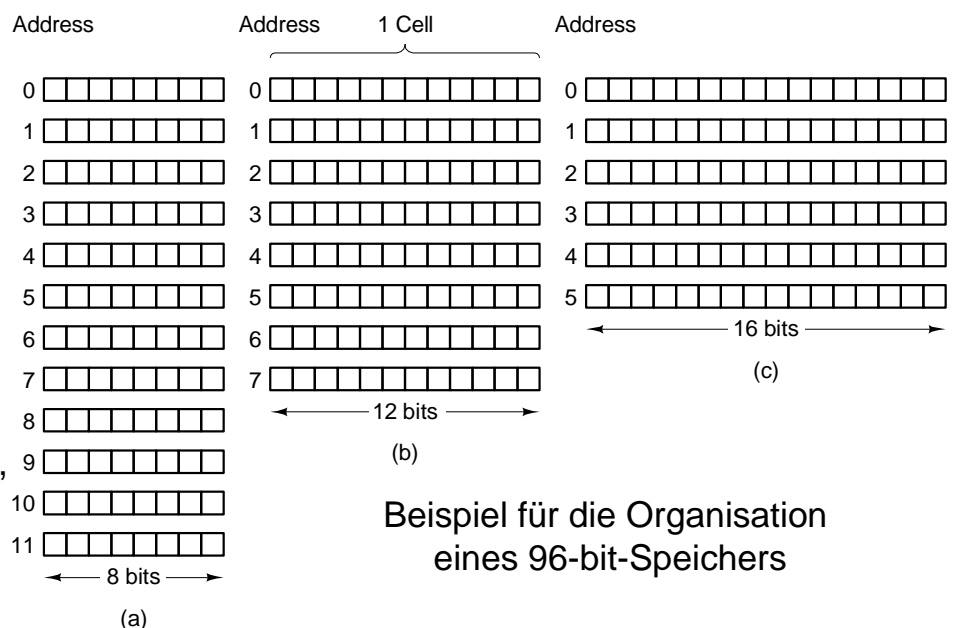


Speichertechnologie von Intel. 400 bis 800 MHz sind möglich. Metallabdeckung zur Kühlung der Rambus-DRAMs.

Organisation des Arbeitsspeichers

□ Arbeitsspeicher:

- Lineare Liste von Speicherworten
- Aufgebaut aus Speicherbausteinen
- Zugriffszeit hängt allein von der Art der verwendeten Speicherbausteine ab
- Die Breite des Arbeitsspeichers entspricht i.A. der Breite des Datenbus (8, 16, 32, 64 Bit). Dies entspricht der maximalen Informationsmenge, auf die in einem Buszyklus zugegriffen werden kann.

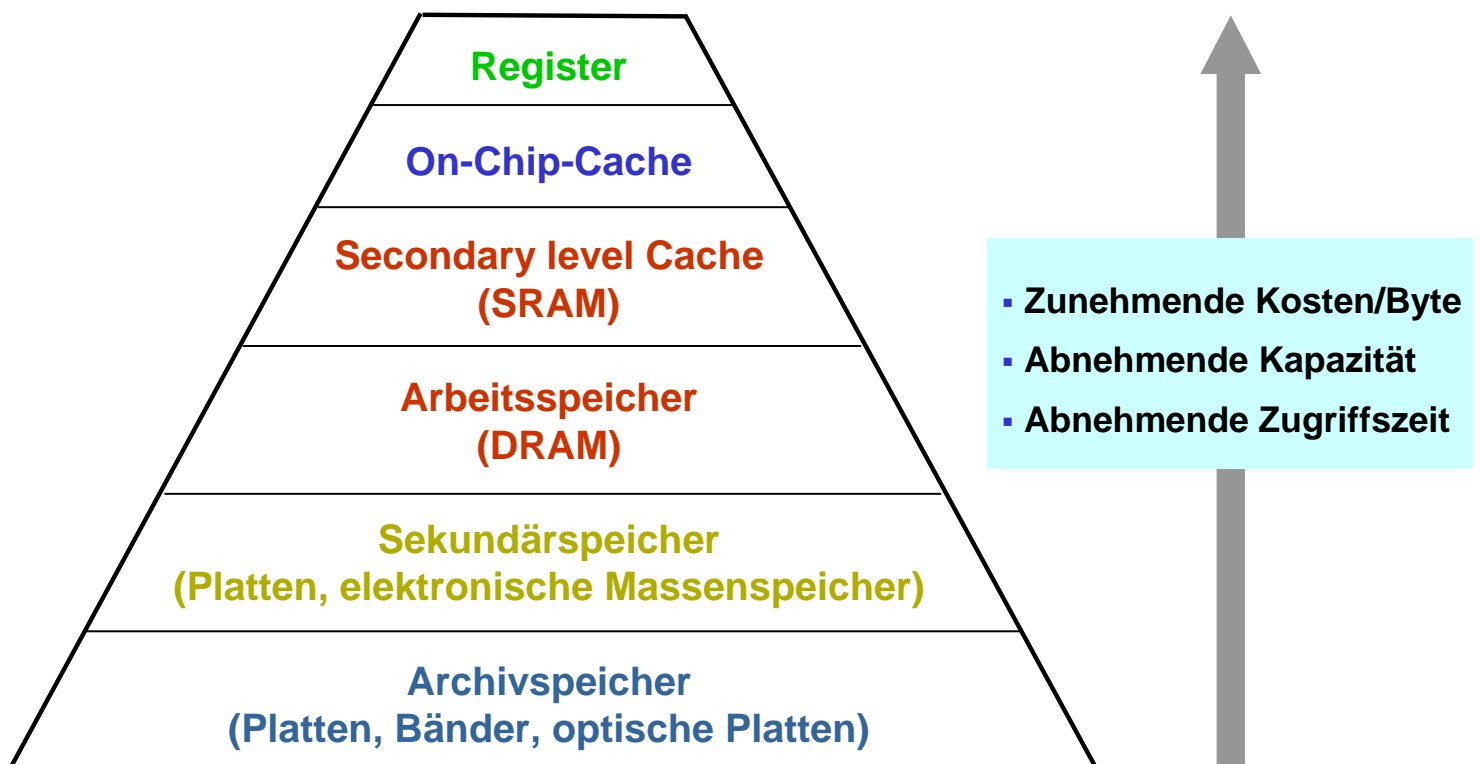


Beispiel für die Organisation eines 96-bit-Speichers

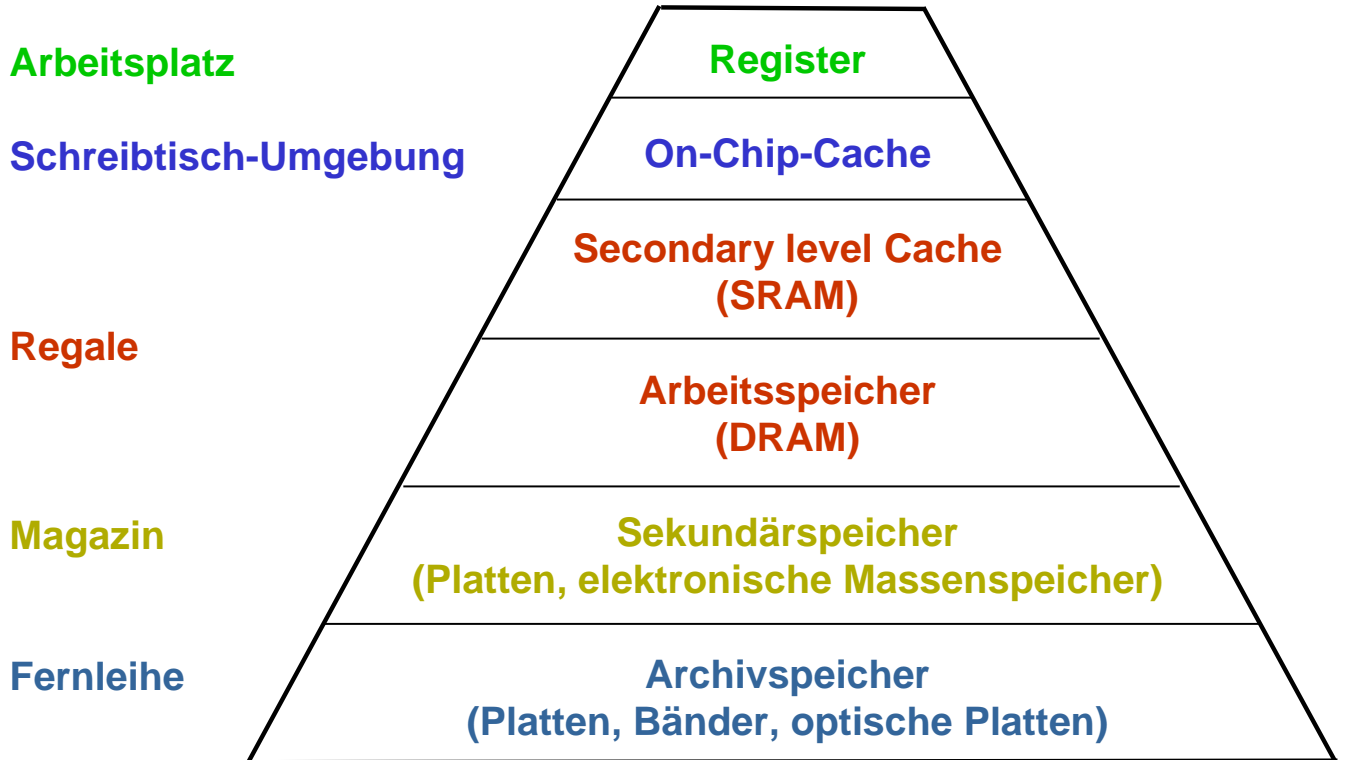
Speicherhierarchie (1)

- Ein *technologisch einheitlicher* Speicher mit *kurzer Zugriffszeit* und *großer Kapazität* ist aus *Kostengründen* i.a. nicht realisierbar.
- Lösung:
 - Schichtenweise Anordnung verschiedener Speicher und Verschiebung der Information zwischen den Schichten (Speicherhierarchie)
 - Speicherhierarchie zum Ausgleich der unterschiedlichen Zugriffszeiten der CPU und des Hauptspeichers.
- Zwei Strategien sind möglich:
 - Cache-Speicher: Kurze Zugriffszeiten
 - Beschleunigung des Prozessorzugriffs
 - Virtueller Speicher:
 - Vergrößerung des tatsächlich vorhandenen Hauptspeichers
 - Z.B. bei gleichzeitiger Bearbeitung mehrerer Prozesse.

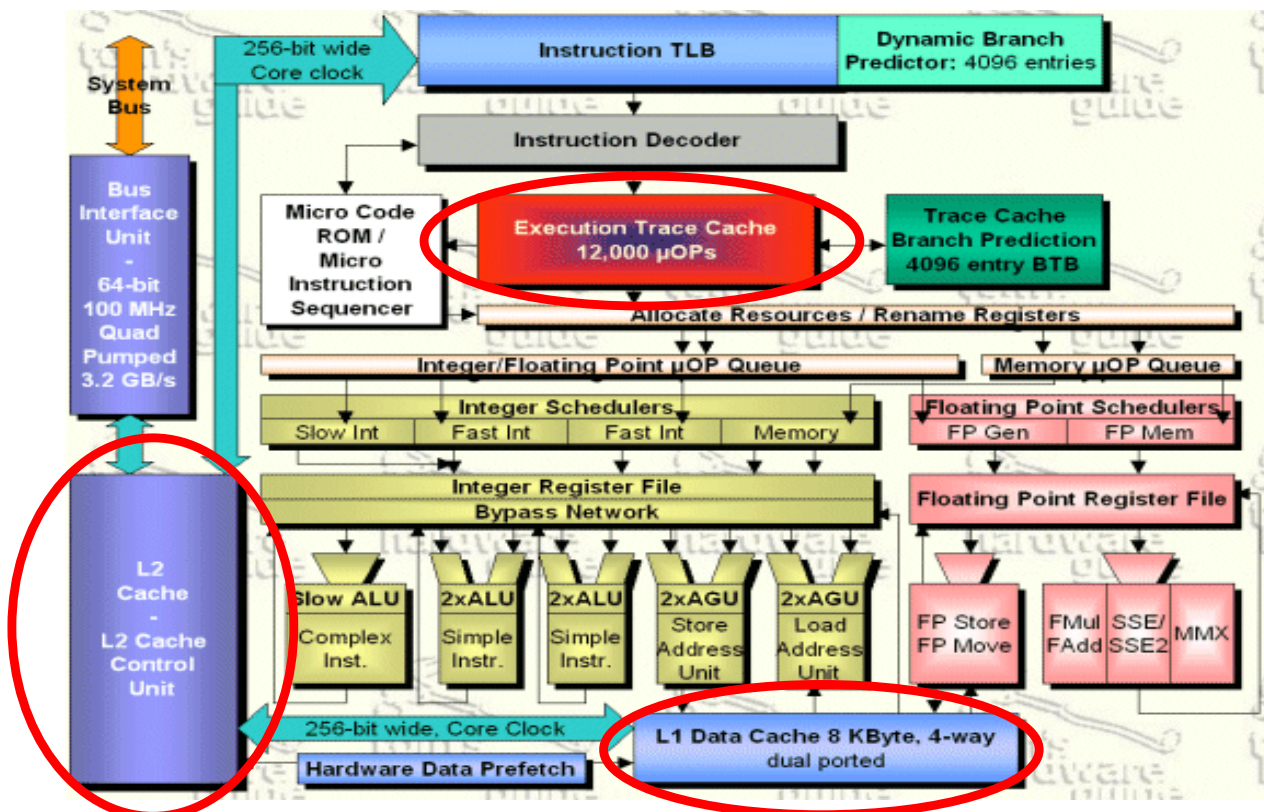
Speicherhierarchie (2)



Speicherhierarchie (3) — Vergleich

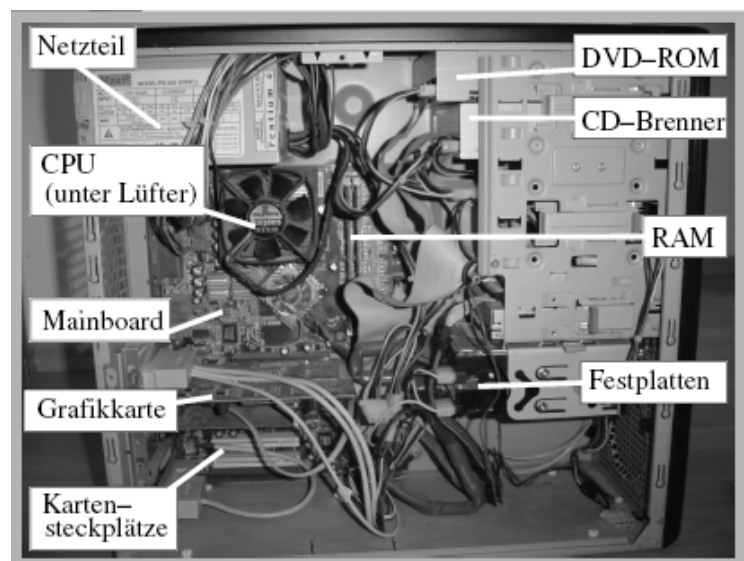
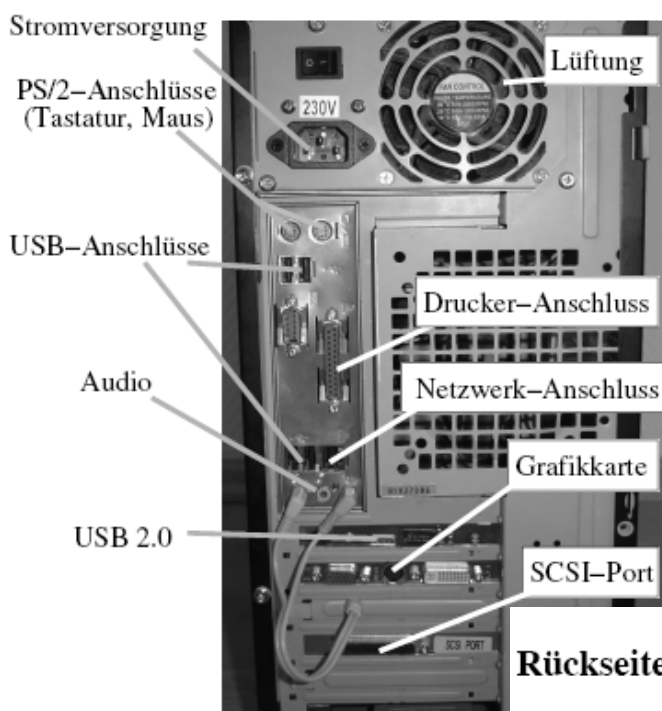


Cache-Speicher im Pentium 4



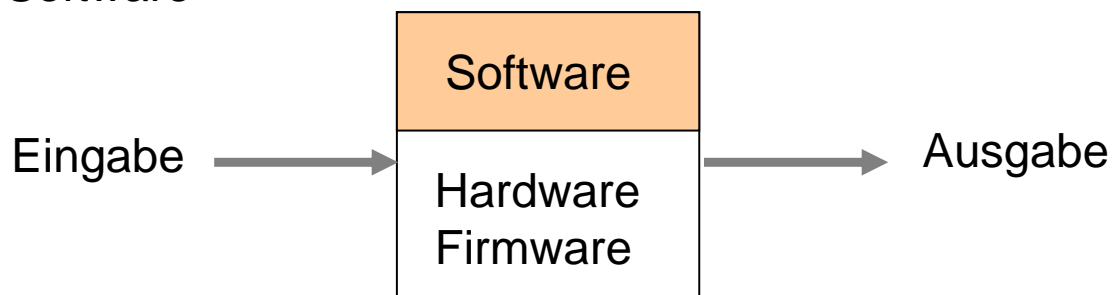
- ❑ von-Neumann Architektur
- ❑ Aufbau und Funktionsweise
- ❑ **Organisation**
- ❑ Peripherie
- ❑ Technologieentwicklung

Die heutigen Rechner (Personal Computer, PC)



Komponenten eines Digitalrechners

- **Hardware (HW)**
Alle mechanischen und elektronischen Bauelemente
- **Software (SW)**
Alle Programme, die auf dem Rechner ablaufen
- **Firmware (FW)**
Mikroprogramme in ROMs, Mittelstellung zwischen Hardware und Software



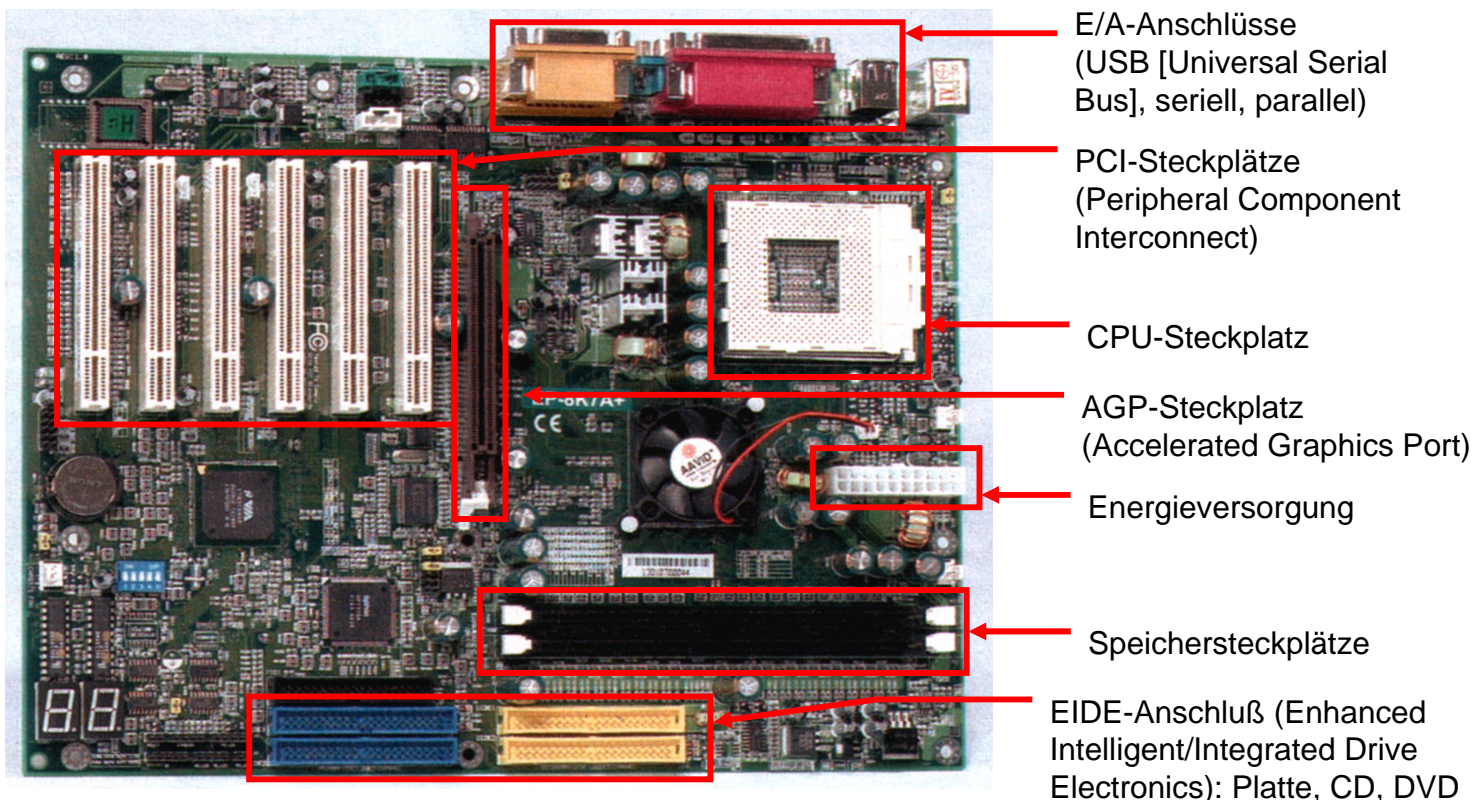
BIOS (Basic Input/Output System)

- Das BIOS ist ein Chip, der sich bei IBM-PCs auf dem Mainboard befindet und *Firmware* enthält, die hier die Basis-Steuerlogik für den Start des Rechners beinhaltet.
 - Das BIOS besteht heute aus Flash-PROM und meist zusätzlich noch aus einem kleinen, batteriegepufferten RAM-Baustein, in dem Konfigurationsparameter und auch die System-Uhrzeit gespeichert und ohne externe Spannungsversorgung gehalten werden, zumindest solange die Batterie funktionsfähig ist.
 - Bei jedem Start eines Rechners wird zuerst ein Programm im BIOS ausgeführt, das bestimmte Tests durchführt, dazugehörige Kontrollmeldungen anzeigt und danach ein Betriebssystem von der Festplatte lädt und startet.
- Das Betriebssystem (SW) erweitert schließlich die Fähigkeiten der HW, FW zu einem nutzbaren Gesamtsystem (siehe M7).

Funktionen des BIOS

- ❑ *POST – Power-On Self Test (Selbsttest beim Einschalten)*: Testet als Erstes die wichtigsten Hardwarekomponenten (Grafikkarte, RAM etc.)
- ❑ *Einfache Kommunikation mit der Hardware*: Über diese Funktionalität läßt sich z.B. die Rechneruhr einstellen oder man kann festlegen, ob von einer CD oder von der Festplatte das Betriebssystem zu laden ist.
- ❑ *Übergabe der Kontrolle an den Datenträger*: Nach einem erfolgreichen POST-Test übergibt das BIOS die Kontrolle an den Datenträger, von dem das System gestartet werden soll. Es wird hierbei das Programm im so genannten *Master Boot Record (Startsektor; kurz MBR)* des Boot-Laufwerks in den Arbeitsspeicher geladen und gestartet.

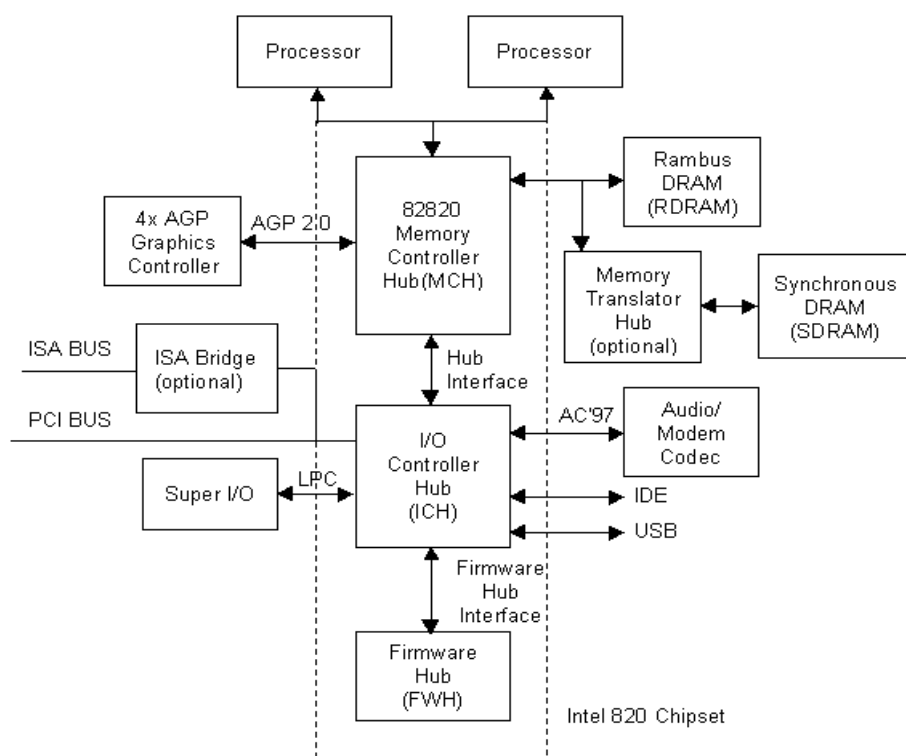
Motherboard/Mainboard



Chip-Satz

- Der Chip-Satz ist das Bindeglied zwischen den einzelnen Komponenten eines Computersystems.
- Die Chipsätze von verschiedenen Herstellern können teilweise Leistungsunterschiede von bis zu 10% haben.
- Der Chipsatz legt fest, welche Komponenten verwendet werden können:
 - Systembus
 - Speichertyp
 - Schnittstellen
 - Prozessortyp

Beispiel: Intel 820 System Block Diagram

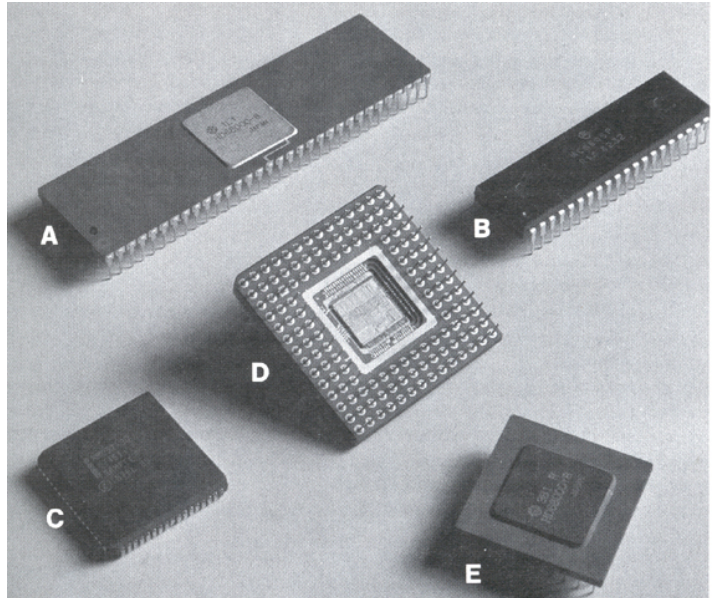


HW eines Prozessors/Mikroprozessors

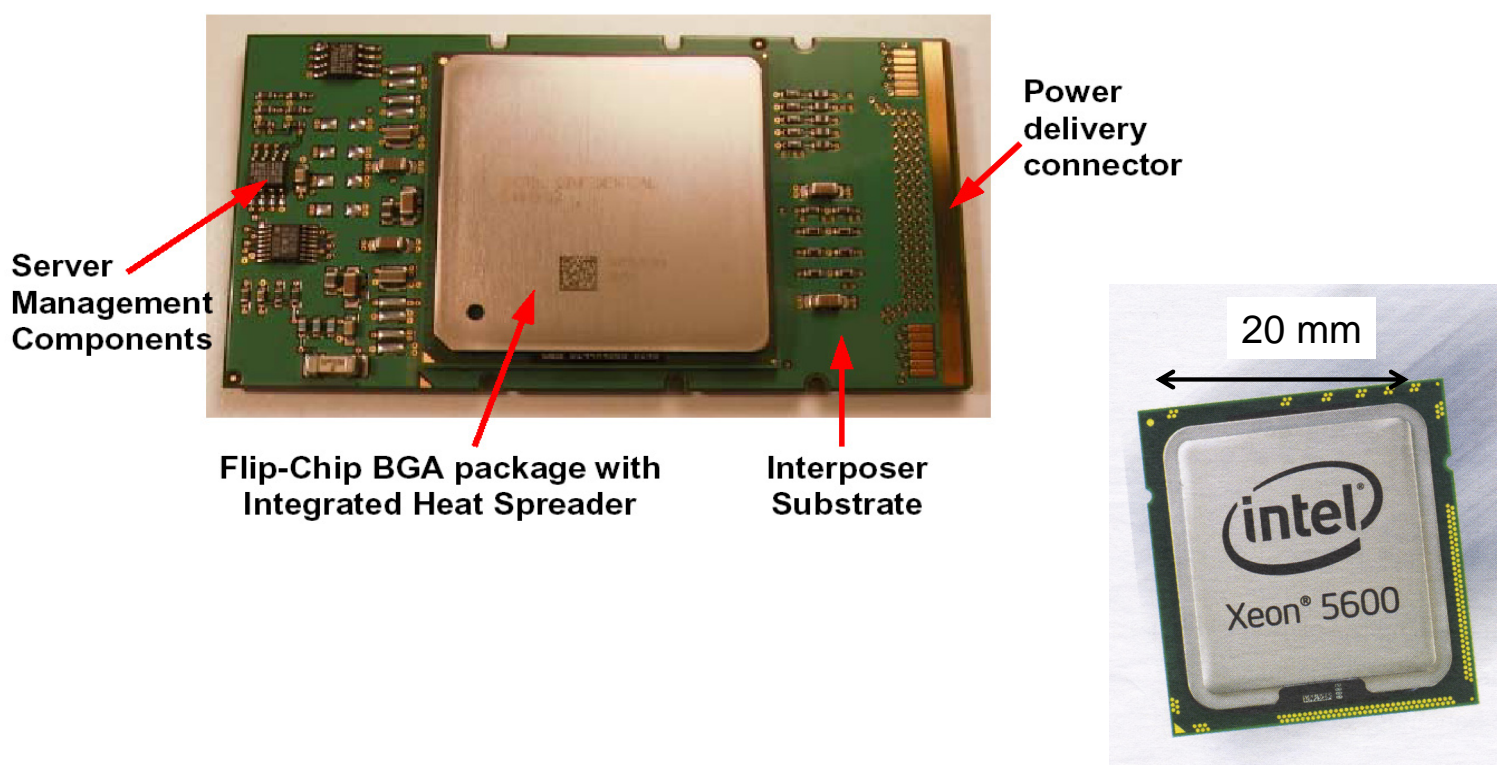
- Gehäuse und Anschlüsse
 - Ein Chip muß zur Erhöhung der mechanischen Stabilität, zur Herausführung der Anschlüsse sowie zur Ableitung der Wärme (z.B. bis 80 Watt) in ein Gehäuse untergebracht werden.

- Gehäusematerialien:
 - Plastik
 - Keramik

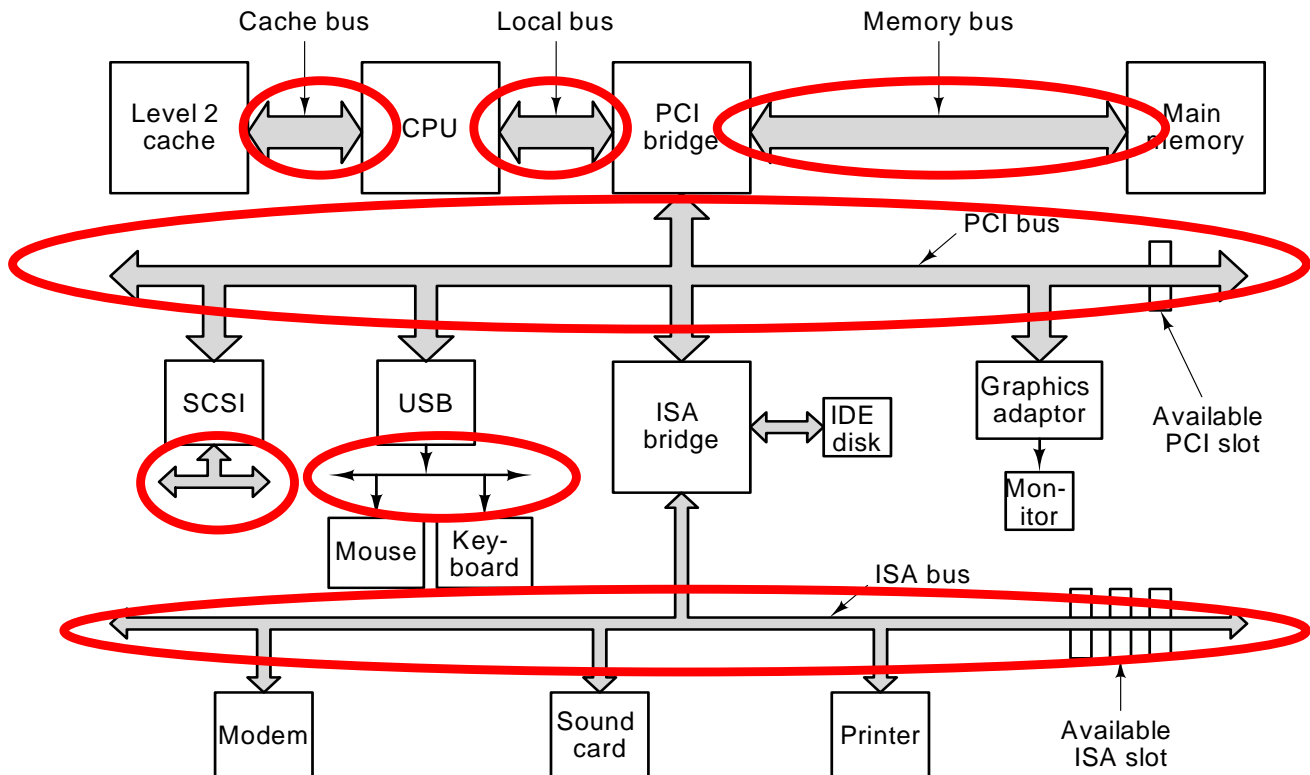
- Gebräuchlichste Gehäusetypen:
 - A bis E



Intel Itanium CPU Package und Xeon 5600 Serie



Die Großfamilie der Busse

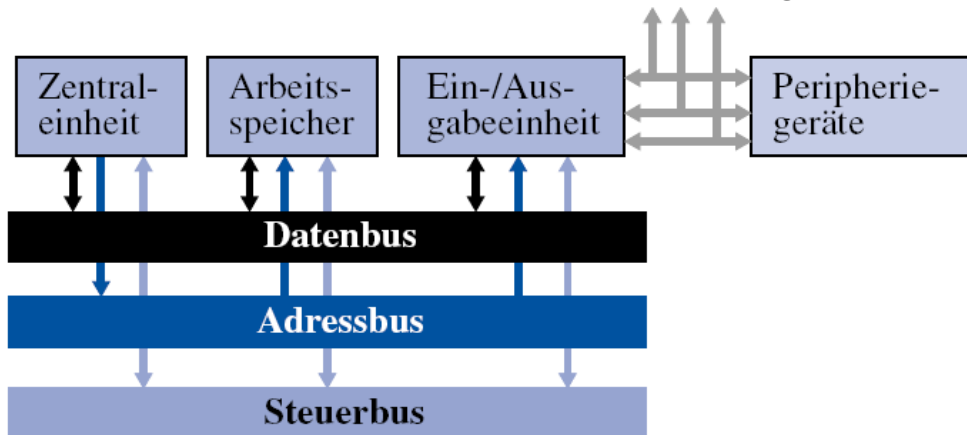


Busse und Schnittstellen

- Busse und Schnittstellen werden sowohl zur **Kommunikation zwischen den Bestandteilen des Mainboards** als auch zum Anschluß aller Arten von Peripheriegeräten benötigt, wie z.B. für Grafikkarten, Festplatten, Drucker.
- Dem **Transport von Daten** zwischen den Einheiten auf dem Mainboard, dem Prozessor, dem Arbeitsspeicher und der Ein-/Ausgabeeinheit dient ein so genanntes internes *Bussystem*.
- Aus Geschwindigkeitsgründen werden auf dem internen Bussystem **mehrere Bits parallel übertragen**. Die Anzahl der parallel zu übertragenden Bits hängt von der an den HW-Chips verfügbaren Busleitungen ab und korreliert sinnvollerweise mit der Bitlänge der Prozessorregister und der darin zu verarbeitenden maximalen Datengrößen.

Bustypen

- **Datenbus:** Er dient der *bidirektionalen Übertragung von Daten* zwischen den Einheiten.
- **Adressbus:** Er dient der *unidirektionalen Übermittlung von Adressen zum Speicher (oder zu den Ein-/Ausgabeeinheiten)*.
- **Steuerbus:** Er dient zur Koordination exklusiver Zugriffe auf den Daten- und Adressbus (Bus reservieren, freigeben, ...).

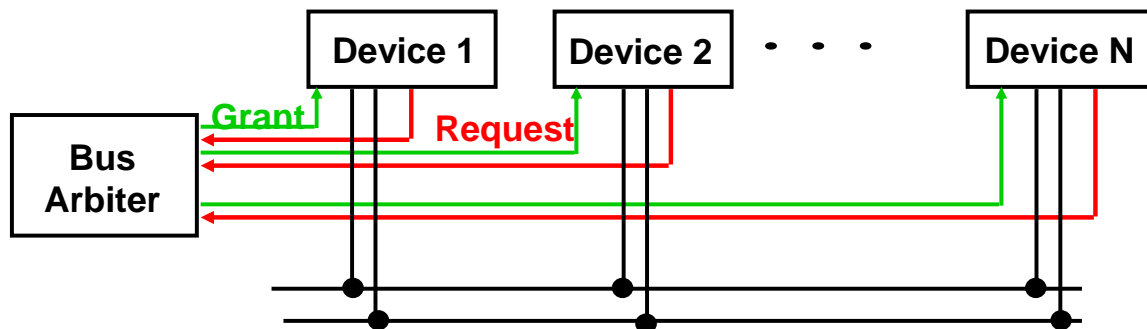


Kommunikationsverfahren

- Für die Kommunikation zwischen Prozessor, den einzelnen Bestandteilen des Mainboards und den angeschlossenen Peripheriegeräten werden verschiedene Techniken eingesetzt.
 - *Polling von Daten*
Bei diesem Verfahren fragt der Prozessor in bestimmten Zeitabständen bei einem Gerät nach, ob Daten zur Übertragung anstehen.
 - *Interrupt Requests (IRQs)*
Bei diesem Verfahren kann eine Kommunikation mit dem Prozessor von einem Gerät durch Auslösung bestimmter Signale, so genannter IRQs (Interrupt Requests), begonnen bzw. angefordert werden.
- Ein Systemsteuerbaustein wird benötigt, wenn mehrere Komponenten selbstständig auf den Systembus zugreifen und so zum *Bus Master* werden (Zuteilungsverfahren).
- Der Bus-Arbiter regelt hierbei die Hierarchie der Zugriffsberechtigungen und gewährt den Zugriff auf den Systembus.

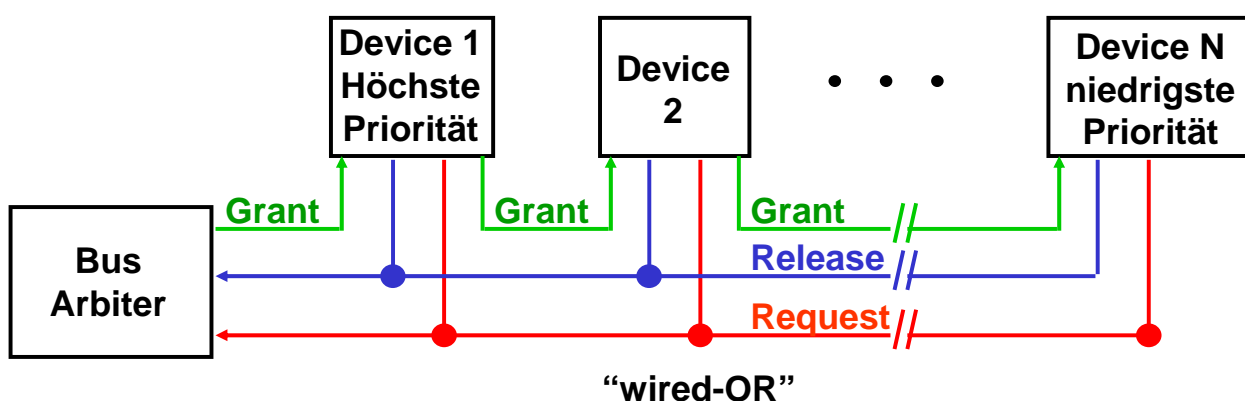
Systembus-Zuteilungsverfahren (1)

- Zentrales Verfahren (unabhängige Anforderung):
 - Jede Komponente (device) fordert über ihren Bus-Arbiter den Zugriff auf den Systembus von einem *zentralen Prioritäts-Decoder* an,
 - Dieser vergibt den Bus bei mehreren gleichzeitigen Anforderungen nach einem festen Algorithmus.
 - Im eigentlichen Sinn stellt also er den „Schiedsrichter“ dar.
 - Zwei Schritte für eine Zuteilung sind notwendig:
 - Request = Anfrage der Zuteilung
 - Grant = Zuteilung erfolgt



Systembus-Zuteilungsverfahren (2)

- Dezentrales Verfahren (Daisy-Chain):
 - Alle Arbiter werden kettenförmig miteinander verbunden.
 - Jeder schiebt einen *Bus-Grant*, den er selbst nicht benötigt (kein eigener *Bus-Request*), an den nächsten Arbiter weiter.
 - Der erste Arbiter in der Kette hat die höchste Priorität
 - Zusätzlich dritter Schritt für eine Zuteilung notwendig:
 - Release = Weitergabe der Zuteilung

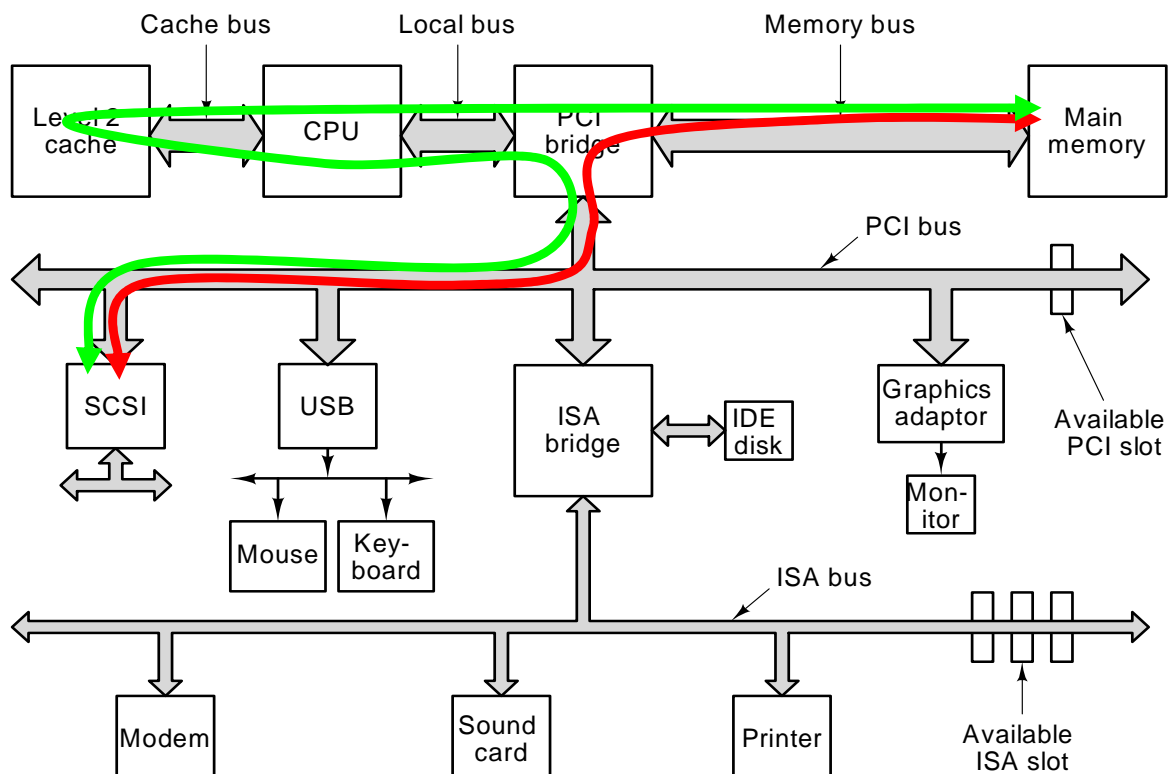


Anschlüsse und Schnittstellen

□ Möglichkeiten zur Übertragung der Kommunikationsdaten:

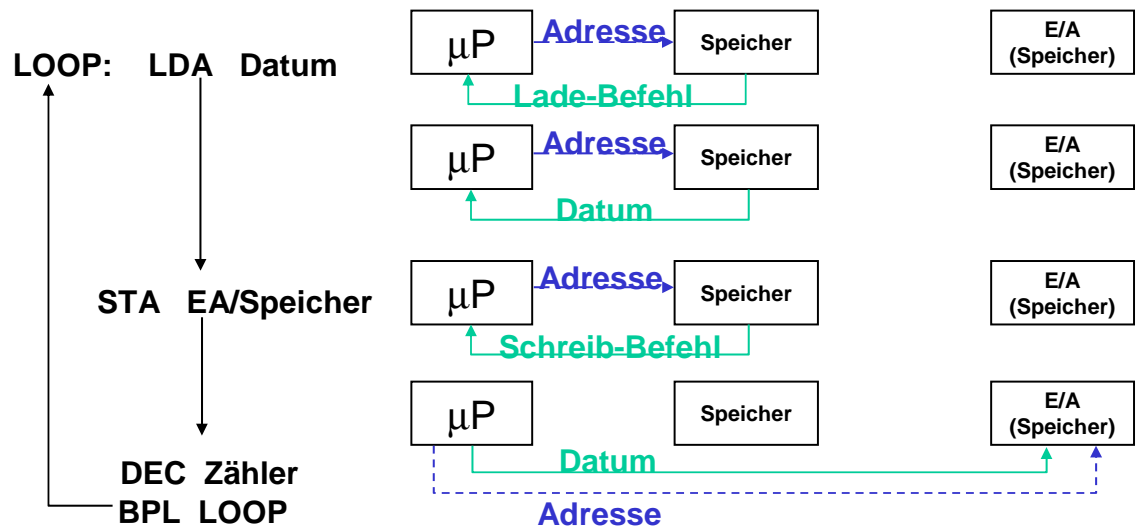
- *E/A-Port- bzw. Basisadressen (Programmed I/O)*
legen den Beginn eines vereinbarten Adreßblocks fest, der für den Austausch der Daten zwischen dem Prozessor und dem jeweiligen Gerät verwendet wird.
- *Memory Mapped I/O*
Prozessoren besitzen nicht immer spezielle Adreßbereiche für Ein-/Ausgabegeräte. Adressen für E/A-Geräte können prinzipiell auch im Adreßbereich des Arbeitsspeichers liegen.
- *DMA-Kanäle (Direct Memory Access) und Bus Mastering*
sind spezielle Verfahren zur direkten Übertragung von Gerätedaten in den Arbeitsspeicher und umgekehrt, ohne daß die Daten den Prozessor passieren müssen.

PIO (Programmed I/O) und DMA (Direct Memory Access)



Programmierte Ein- und Ausgabe (PIO)

- Datenübertragung zwischen Speicher und Peripherie mittels Prozessor:



- Nachteil: Mindestens vier Speicherzugriffe pro Datum nötig
→ langsamer Datentransfer, Prozessor belastet

Direkter Speicherzugriff (DMA)

- Hierbei erfolgt der Datentransfer ohne Beteiligung des Mikroprozessors direkt zwischen den beteiligten Komponenten.
- Ein spezieller Baustein, DMA-Controller genannt, koordiniert den Datentransfer.
 - Vorteile:
 - Speicherzugriffe für das Holen der Lade-, Speicher- und Schleifenbefehle entfallen, da die Datenübertragung hardwaremäßig (ohne Programm) ausgeführt wird
→ nur zwei (ggf. sogar nur ein) Speicherzugriff pro Datum nötig
 - Der Prozessor wird entlastet und kann derweil andere Dinge tun, sofern diese nicht den Systembus benötigen.
 - Nachteile:
 - Cache-Inkonsistenzen müssen behandelt werden
- Achtung:
 - Es gibt keinen klaren Gewinner! Je nach Anwendung kann PIO oder DMA besser sein.

Anschlüsse für Erweiterungskarten

- Es gibt vor allem im PC-Bereich verschiedene Arten von Schnittstellen zum Anschluss von Ergänzungs- bzw. Erweiterungskarten:
 - Der *PCI-Anschluss (Peripheral Component Interface)* ist der Standard-Kartenanschluss für PCs (32-Bit, 33 MHz).
 - Der *AGP-Anschluss (Accelerated Graphics Port)* ist ein spezieller Anschluss für Grafikkarten (64-Bit, \geq 66 MHz).
 - Der *PCMCIA-Anschluss (Personal Computer Memory Card International Association)* wird auch als *PC-Card-Anschluss* und häufig bei Notebooks als externer Anschluss für spezielle kleine Einsteckkarten eingesetzt.

Anschlüsse für Laufwerke

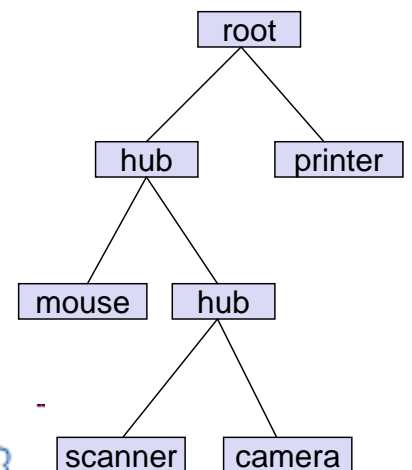
- Für Festplatten, CD-ROM-Laufwerke und andere Massenspeicher gibt es zwei wichtige Arten von Schnittstellen.
 - *EIDE (Enhanced Integrated Device Electronics)* ist auf den meisten PC-Mainboards integriert. Mit jedem der beiden Anschlüsse können je zwei Geräte verbunden werden: ein *Master* und ein *Slave*.
 - *SCSI (Small Computer System Interface)*: Eine SCSI-Schnittstelle erlaubt den Anschluss von sieben Geräten bzw. *Wide-SCSI* läßt sogar den Anschluß von 15 Geräten zu.
- **Nachfolgetechnologie:**
 - *S-ATA (Serial-Advanced Technology Attachment)* ist eine Nachfolgetechnologie zu IDE / EIDE (auch P-ATA) mit schneller, serieller Datenübertragung.

Anschlüsse für weitere Peripherie

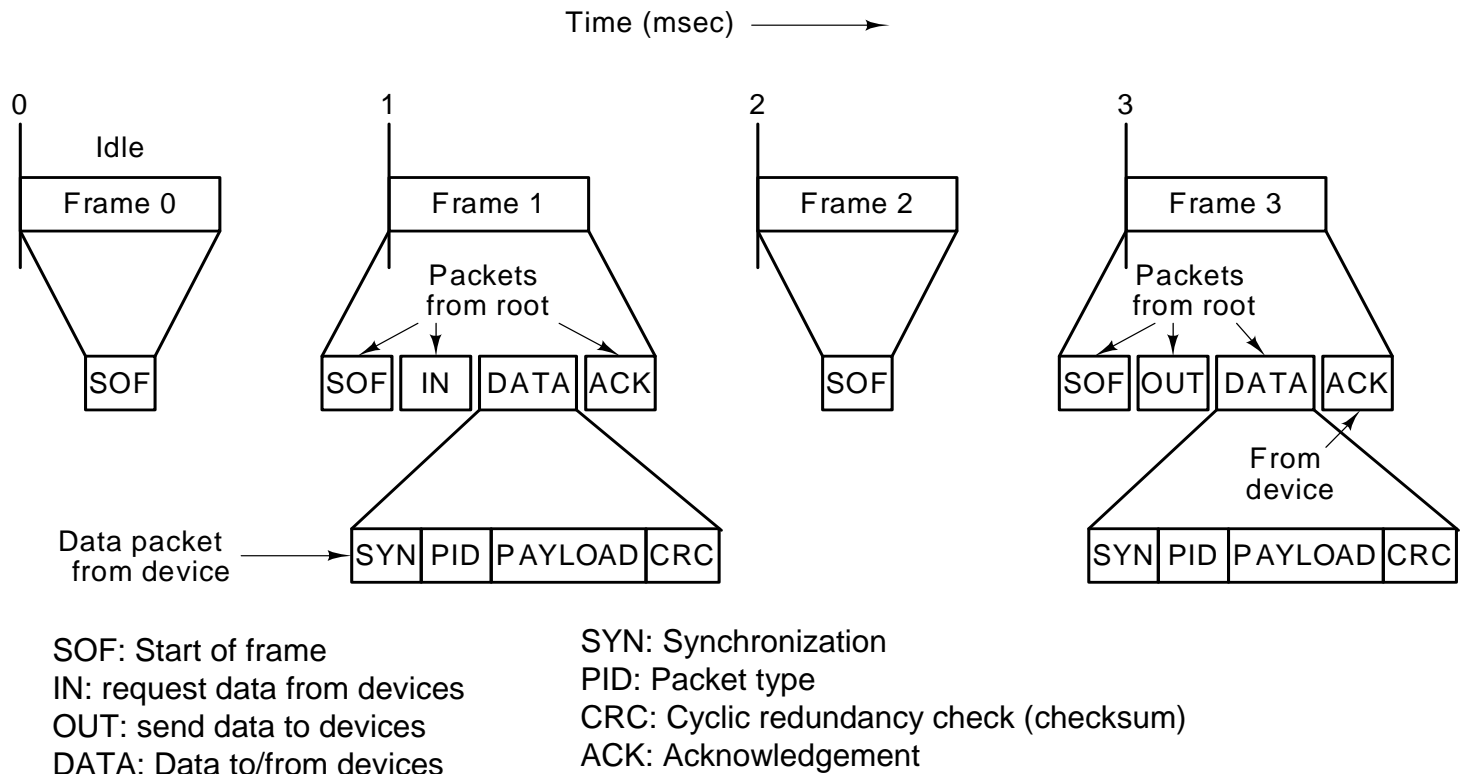
- Heutige Rechner sind mit neuartigen, seriellen Schnittstellen für externe Geräte ausgestattet
- Diese unterstützen das so genannte *Hot-Plugging*-Verfahren:
 - Geräte können im laufenden Betrieb angeschlossen und wieder entfernt werden.
- Der *USB-Anschluss (Universal Serial Bus)* arbeitet als *USB 1.0 und 1.1* mit einer Datenübertragungsrate von 12 Mbit/s und als *USB 2.0* mit einer Übertragungsrate von 480 Mbit/s. USB 3.0 erreicht neuerdings eine Rate von bis zu 5 Gbit/s.
- Die *IEEE-1394-Schnittstelle* (auch *FireWire* genannt) arbeitet mit einer Übertragungsrate von 400 Mbit/s bzw. das neuere *FireWire 800* mit einer Übertragungsrate von 800 Mbit/s.

USB (Universal Serial Bus 1.0 – Basic Idea)

- Connection of low-bandwidth peripherals
 - Mouse, keyboard, scanner, printer, camera, PDA, ...
- Plug-and-Play
 - Simple connection, every time possible, no configuration
- Services
 - 12 Mbit/s shared bandwidth (USB2.0: 480 Mbit/s
USB 3.0: 4,8 Gbit/s)
 - Isochronous and bulk data transfer
- System
 - Tree topology
 - Root hub at root polls devices (leaves)
 - 4 wires/cable: 2 power supply, 2 data
- See <http://www.usb.org> for more info.



USB Frames (Root Issues Frames Every 1 ms)



Serielle und parallele Datenübertragung (Bus)

- Bei den unterschiedlichen Anschlußsystemen ist die jeweilige Art der Datenübertragung ein wichtiges Merkmal.
 - Bei der *seriellen Datenübertragung* werden die einzelnen Bits nacheinander.
 - Bei der *parallelen Übertragung* werden die Bits dagegen gleichzeitig auf mehreren nebeneinander liegenden Leitungen übertragen, wie z. B. 8, 16, 32 oder 64 Bits.
- Da eine serielle Datenübertragung weniger Strom benötigt und auch für größere Entfernungen als eine parallele Datenübertragung eingesetzt werden kann, gewinnt die serielle Übertragung von Daten über lange Entfernungen hinweg immer mehr Oberhand gegenüber der parallelen. Hingegen ist die parallele Datenübertragung innerhalb eines Rechners entscheidend für dessen interne Geschwindigkeit.

- ❑ von-Neumann Architektur
- ❑ Aufbau und Funktionsweise
- ❑ Organisation
- ❑ **Peripherie**
- ❑ Technologieentwicklung

Peripherie

- ❑ E/A-/Speichergeräte, die an einen Rechner angeschlossen sind.
 - Es existieren viel zu viele Peripherieelemente und –systeme, als daß auch nur in einem annähernd sinnvollen Ansatz ein Überblick gegeben werden könnte.
 - **HD**, SSD, **CD**, **DVD**, scanner, laser printer, CRT, LCD, audio, mouse, trackball, touchpad, keyboard, hard disk, camera, 3-D-glasses, 3-D-printer, force feedback joystick, ink printer, robot, card reader, PC-card, **modem**, LAN adapter, WLAN adapter, GSM adapter, fax, sensors, actors, ...

Geräte

Eingabe Ausgabe Massenspeicher

Tastatur, Maus, Scanner, Digitalkamera

X

CD-ROM-Laufwerk

X

X

Grafikkarte, Bildschirm, Drucker, Lautsprecher

X

Netzwerkkarte, Modem, Soundkarte

X

X

Festplatte, USB-Stick, CD-Brenner

X

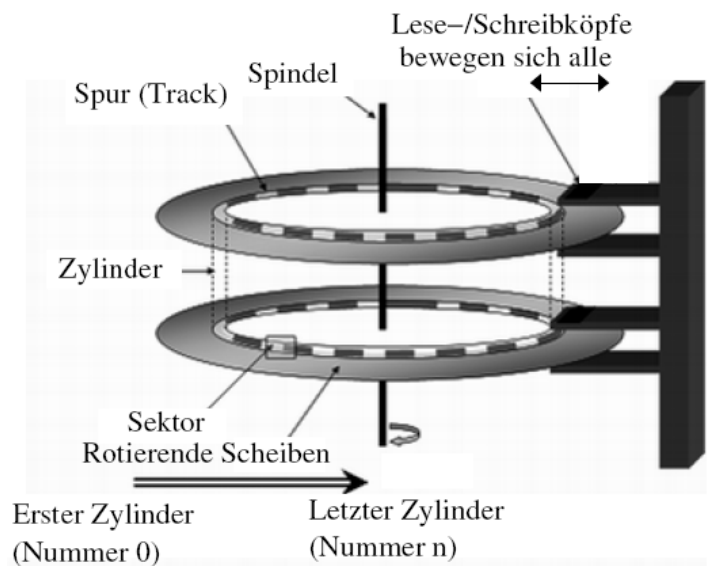
X

X

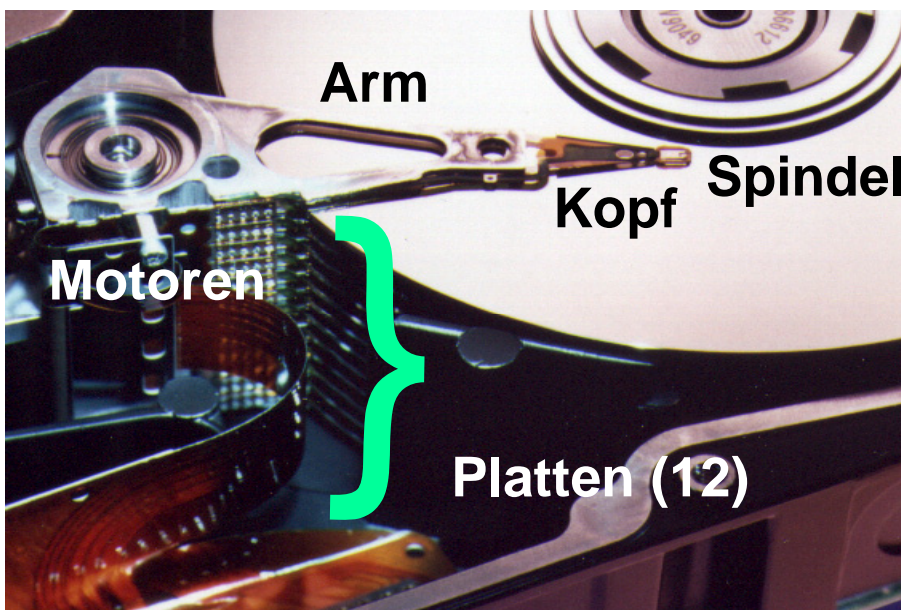
Festplatte (Harddisk, HD) (1)

□ Verschiedene Speicherplatten:

- Information ist magnetisch auf beiden Seiten der Platten gespeichert.
- Bits sind in Spuren, die wiederum in Sektoren unterteilt sind (z.B. 512 Byte), gespeichert.
- Motoren bewegen die Köpfe (≥ 1 /Plattenseite) über die Spuren einer selektierten Oberfläche. Warten bis der zu lesende bzw. zu schreibende Sektor unter dem Kopf vorbeikommt.
- Zylinder: Alle Spuren unter den Köpfen



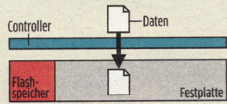
Festplatte (Harddisk, HD) (2)



Festplatte (Harddisk, HD) (3)

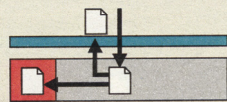
Tempo durch Controlling

Die Hybridplatte besitzt einen speziellen Controller, der wichtige Dateien für einen schnellen Zugriff in den Flash kopiert.



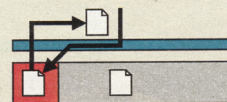
Schreiben

Zu speichernde Daten leitet der Controller an die HDD weiter. Die SSD ist für Schreibzugriffe gesperrt.



1. Zugriff

Ein Counter im Controller zählt die Dateizugriffe. Angeforderte Files werden in den Flash kopiert.



2. Zugriff

Ruft der Nutzer die Dateien nochmals auf, leitet der Controller die Anfrage zum schnellen Flashspeicher um.

DIE HYBRIDPLATTE

Die Momentus XT besteht aus einer 500 GByte großen HDD-Platte und vier GByte SSD-Speicher.

HDD

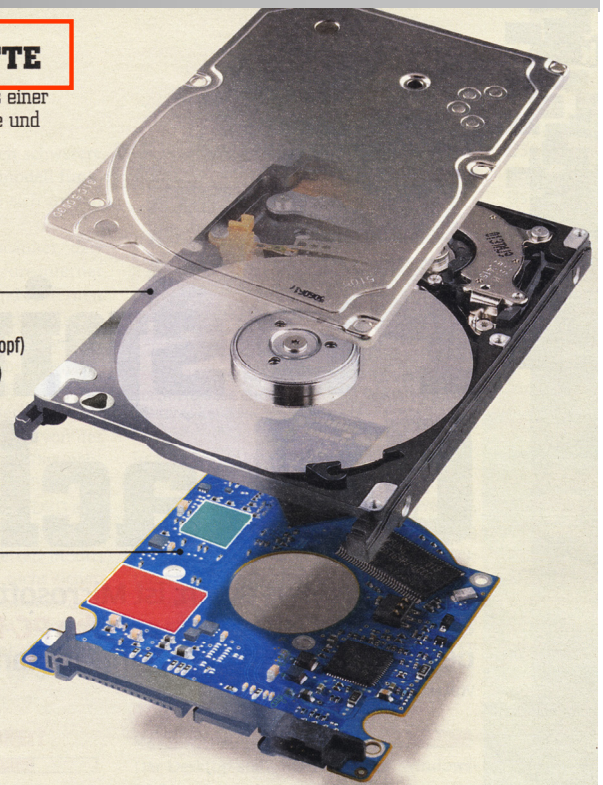
Datenspeicher: Magnetscheibe
Zugriff: mechanisch (Schreib-Lese-Kopf)

- + günstig (ca. 0,08 Euro pro GByte)
- + unbegrenzte Schreibvorgänge
- schockempfindlich
- langsam
- hoher Stromverbrauch
- hörbares Arbeitsgeräusch

SSD

Datenspeicher: Halbleiter
Zugriff: elektronisch

- teuer (ca. 2,60 Euro pro GByte)
- begrenzte Schreibvorgänge
- + schockresistent
- + schnell
- + geringer Stromverbrauch
- + geräuschlos

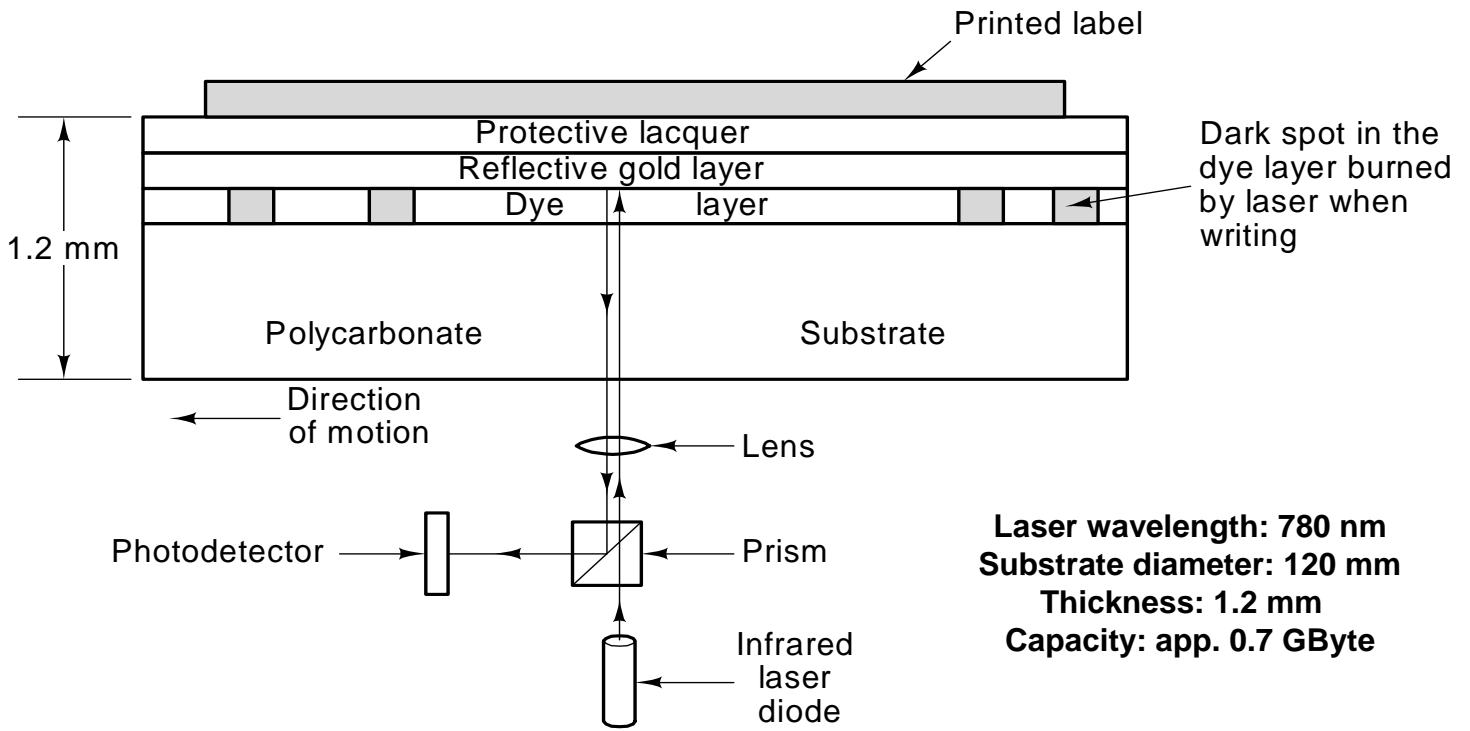


HDD: Hard Disk Drive
SSD: Solid State Disk

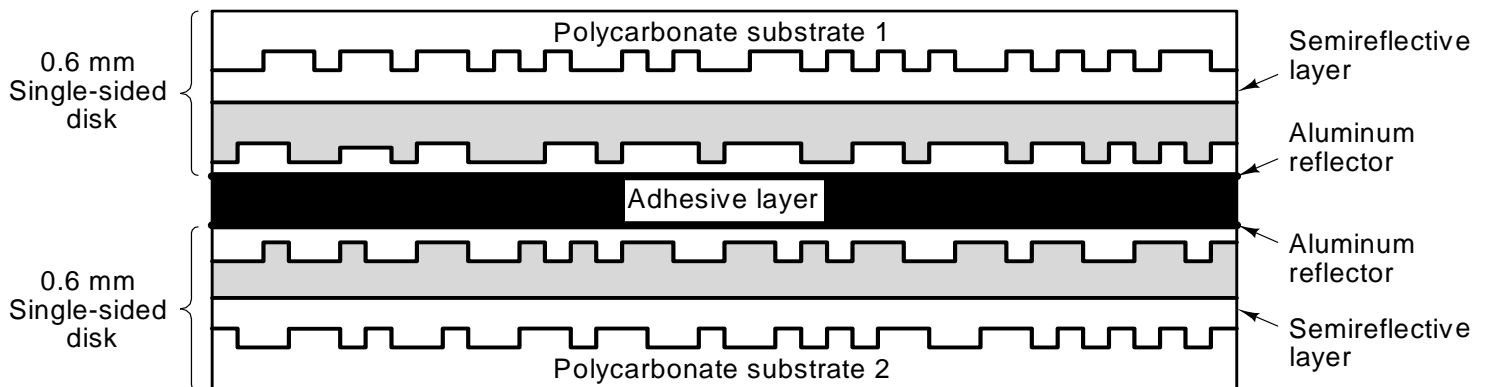
Optische Disks

- Die *CD* ist ein optischer Datenträger und wurde 1982 von Sony und Philips zur Ablösung der Schallplatte vorgestellt.
- Die Oberfläche einer CD besteht aus einer sehr dünnen Metallschicht, die von einem Laserstrahl abgetastet wird. Vertiefungen in dieser Oberfläche (*Pits*) wechseln sich mit der normalen, unversehrten Fläche (*Land*) ab und bilden so das Bitmuster, das von CD-Spielern als Daten, Musik oder Video interpretiert wird.
- Musik, Speicherung von Daten im Computerbereich; damit existieren mehrere Datenformate für die CD-ROM.
- Die DVD (Digital Versatile Disc) besitzt mit 4,7 GByte eine nochmals erheblich höhere Speicherdichte. Es gibt drei zueinander inkompatible Formate.

Compact Disk (CD)

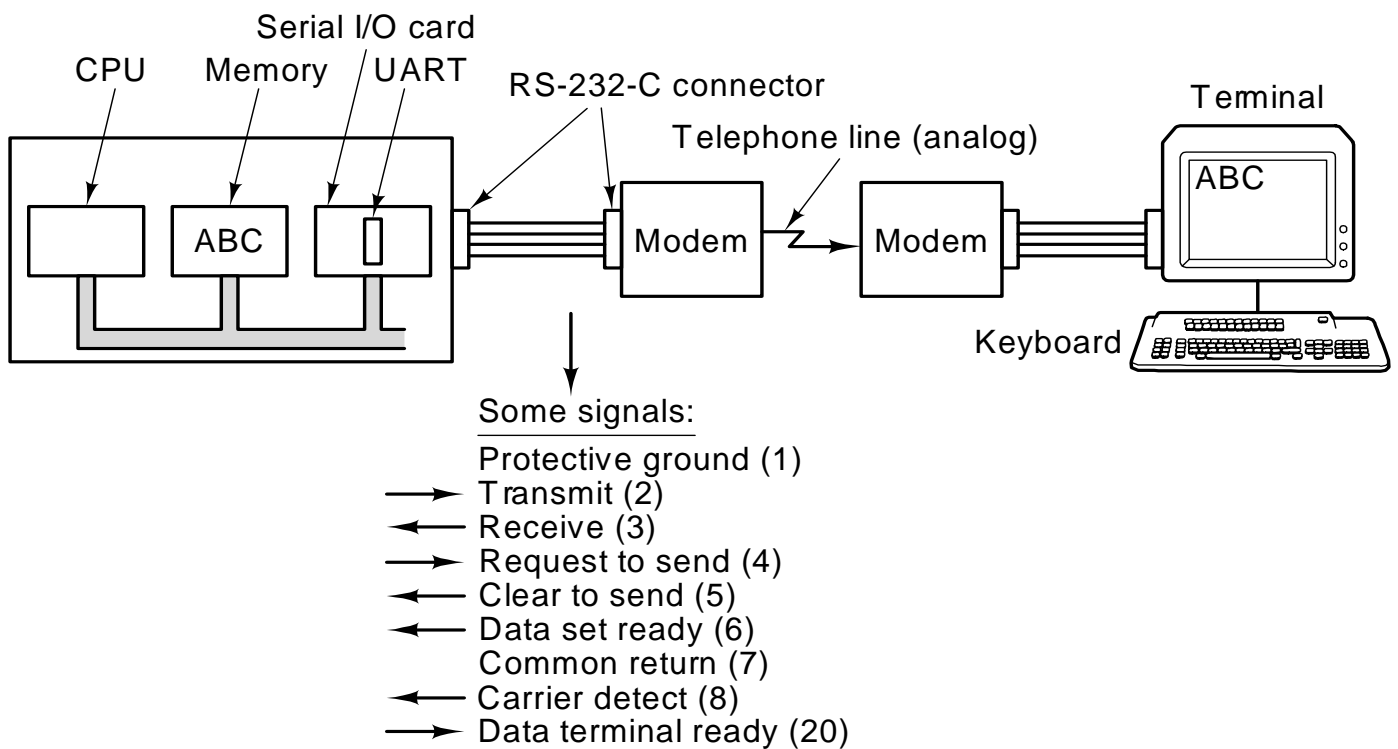


DVD (Double-sided, Dual-layer)



Laser wavelength: 635 or 650 nm
Substrate diameter: 120 mm
Thickness: 1.2 mm
Capacity: 4.7, 8.54, 9.4, or 17 GByte

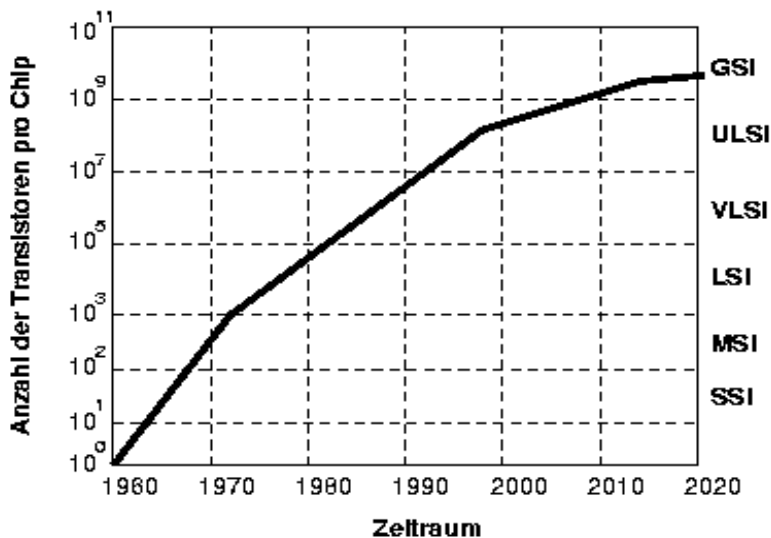
Modem – Verbindung von Rechnern



Modul 5: Rechnerarchitekturen & -organisation

- ❑ von-Neumann Architektur
- ❑ Aufbau und Funktionsweise
- ❑ Organisation
- ❑ Peripherie
- ❑ Technologieentwicklung

Technologieentwicklung



SSI: Small Scale Integration

MSI: Medium Scale Integration

LSI: Large Scale Integration

VLSI: Very Large Scale Integration

ULSI: Ultra Large Scale Integration

GSI: Giga Scale Integration

Leistungssteigerung in Rechnersystemen

- Welche Möglichkeiten gibt es prinzipiell zur Leistungssteigerung in Rechnersystemen?

- Strukturelle Maßnahmen:
 - z.B. Zahl der Transistoren erhöhen
 - Parallelarbeit
 - Abfolge der Arbeitsschritte „verweben“
 - Pipelining

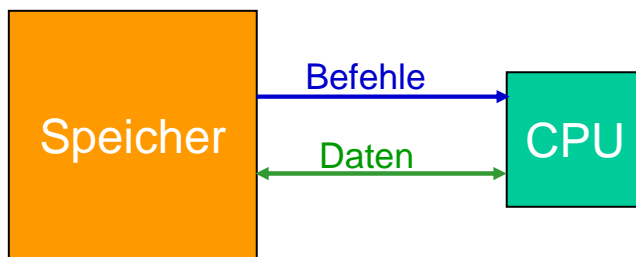
- Technologische Maßnahmen:
 - Anwendung schnellerer Technologien
 - Re-design ist nötig.
 - Alternative Ansätze
 - Vollständig neue Modelle nötig.

Strukturelle Maßnahmen (1)

- Unterscheidung bezüglich der gleichzeitig bearbeiteten Befehls- und Datenströme:

- SISD (Single Instruction Single Data)

Ein Datenstrom wird entsprechend einer seriellen Befehlsfolge verarbeitet (von-Neumann-Rechner)



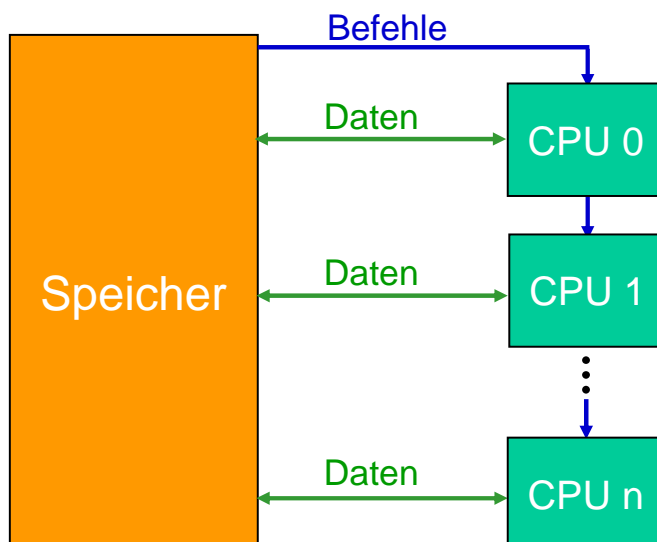
Beispiele:

IBM-PC, IBM 370,
Micro-VAX von DEC

Strukturelle Maßnahmen (2)

- SIMD (Single Instruction Multiple Data)

Alle Prozessoren führen gleichzeitig dieselben Befehle auf verschiedenen Daten aus (Array-Prozessoren)



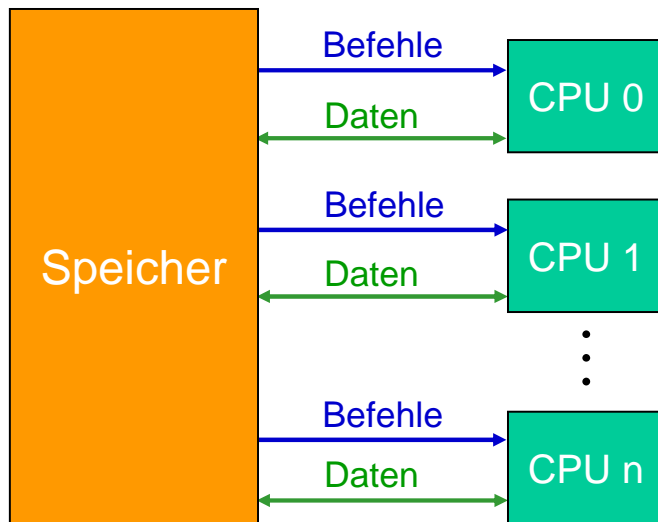
Beispiel Bildverarbeitung:

Jedem Prozessor wird ein
Bildausschnitt zugeordnet.

Strukturelle Maßnahmen (3)

– MIMD (Multiple Instruction Multiple Data)

Alle Prozessoren führen gleichzeitig verschiedene Befehle auf verschiedenen Daten aus



Beispiele

Multiprozessor-
Systeme:

IBM 3084, Cray-2,

Multiprozessor PCs

Strukturelle Maßnahmen (4)

– MISD (Multiple Instruction Single Data)

Es wird nur ein Datenstrom bearbeitet. Bestimmte Ausführungseinheiten übernehmen die Ausführung bestimmter Teile einer Operation (Pipeline-Verarbeitung), was zu einer Parallelität auf Befehlsebene führt.

- Moderne“ Prozessoren: ab Intel 80286
- Bei vielen Autoren bleibt diese Klasse leer – darüber lässt sich diskutieren!

□ Weitere Einteilung:

• Mehrprozessorsysteme:

- Mehrere Prozessoren mit unabhängigen Programmen arbeiten mit einem gemeinsamen Hauptspeicher

• Feldrechner:

- Mehrere Prozessoren arbeiten am gleichen Programm, aber mit verschiedenen Daten (Bsp: Bildverarbeitung)

• System mit **funktionsspezialisierten** Prozessoren:

- Mehrere Spezialprozessoren arbeiten unter einer CPU und mit einem Hauptspeicher

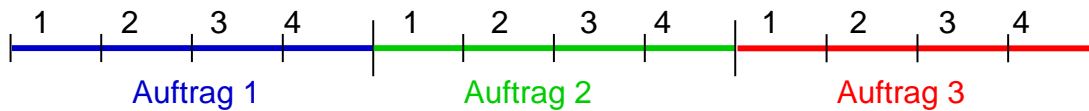
• **Fließbandverarbeitung (Pipeline-Struktur):**

- In einer Kette von Prozessoren übernimmt jeder die Ausführung bestimmter Teile einer Operation.

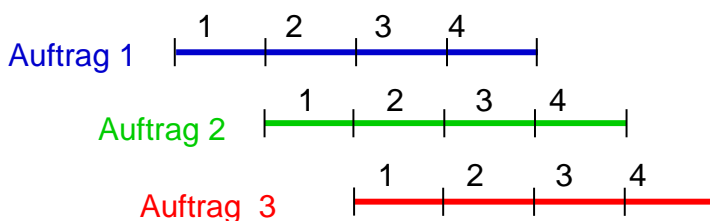
Pipeline-Verarbeitung

- Ausführung von drei gleichartigen Verarbeitungsaufträgen in vier Teilverarbeitungsschritten

- Serielle Verarbeitung:



- Pipeline-Verarbeitung:

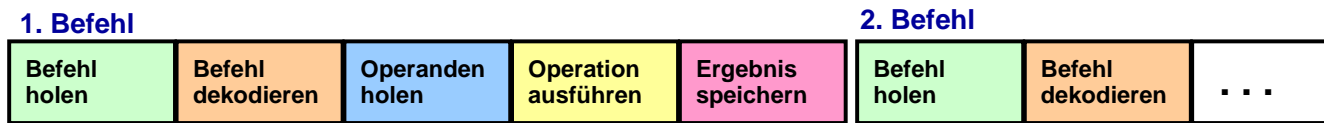


Anwendung: Fünfstufige Befehlspipeline

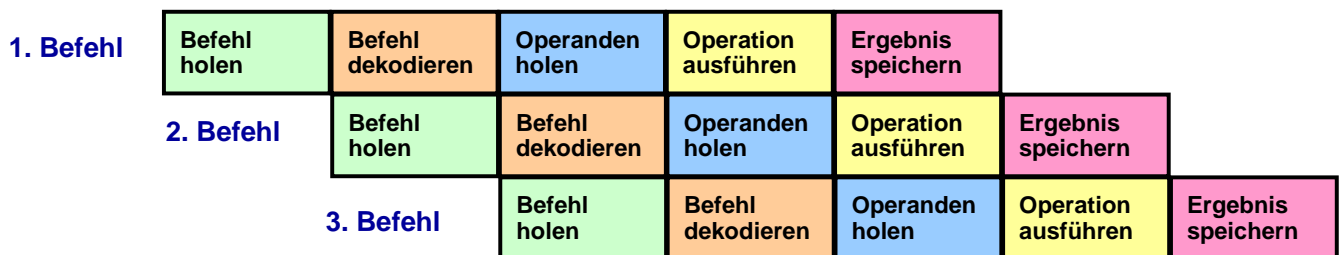
- Befehls-Holphase (instruction fetch):
 - Der Befehl wird aus dem Arbeitsspeicher (bzw. dem Befehls-cache) ins Befehlsregister geladen. Der Befehlszähler wird weitergeschaltet.
- Decodierphase (instruction decode):
 - Aus dem Operationscode des Maschinenbefehls werden prozessorinterne Steuersignale erzeugt.
- Operanden-Holphase (operand fetch):
 - Das Steuerwerk schaltet die Operanden auf die Busse zum Rechenwerk. Diese Operanden stehen im Registersatz. Bei Lade-/Speicherbefehlen oder Verzweigungen wird die effektive Adresse durch das Adreßwerk berechnet.
- Ausführungsphase (execution phase, ALU Operation):
 - Die verlangte Operation wird vom Rechenwerk ausgeführt.
- Abspeicherphase (result write back phase):
 - Ergebnis wird in Register/Speicher abgelegt. Befehle ohne Ergebnis durchlaufen diese Phase passiv. Bei Lade-/Speicherbefehlen wird Adresse auf den Adreßbus gelegt und das Datum zwischen Registersatz/Arbeitsspeicher übertragen.

Pipelining

Sequentielle Ausführung:



Pipelining:



Alternative Rechnerarchitekturen

- Die Datenverarbeitung mit **neuronalen Netzen** stellt eine gänzlich andere Art der Verarbeitung von Daten dar.
- In künstlichen neuronalen Netzen wird ein *vereinfachtes Modell des zentralen Nervensystems (Gehirn) von Lebewesen* nachgebildet. Dieses sind Netzwerke von vielfach miteinander verbundenen Neuronen, die jeweils Prozesselemente sind.
- Ein solches Netzwerk führt keine vorher eingegebenen Befehle (Programm) aus, sondern muß erst durch „Lernen“ dazu gebracht werden, auf Eingabedaten die gewünschten Ausgabereaktionen zu erzeugen.
- Neuronale Netze werden heute üblicherweise auf Prozessoren durch entsprechende Software simuliert; höhere Leistungen werden dabei durch den Einsatz spezieller Prozessoren für schnelle Vektor-/Matrixverarbeitung erreicht.