Kap. 6:
Architektur von Rechensystemen

Quellen

- U. Rembold, P. Levi: "Einführung in die Informatik für Naturwissenschaftler und Ingenieure", 3. Auflage, Hanser-Verlag, 1999 (Kap. 7)
- D. Werner u.a.: "Taschenbuch der Informatik", Fachbuchverlag Leipzig, 1995 (Kap. 3.3)
- M. Broy: "Informatik - Eine grundlegende Einführung", Teil II, Springer-Verlag, 1992 (Kap. 3)
- W. K. Giloi: "Rechnerarchitektur", 2. Auflage, Springer-Verlag, 1993

6.1 Einführung und Überblick

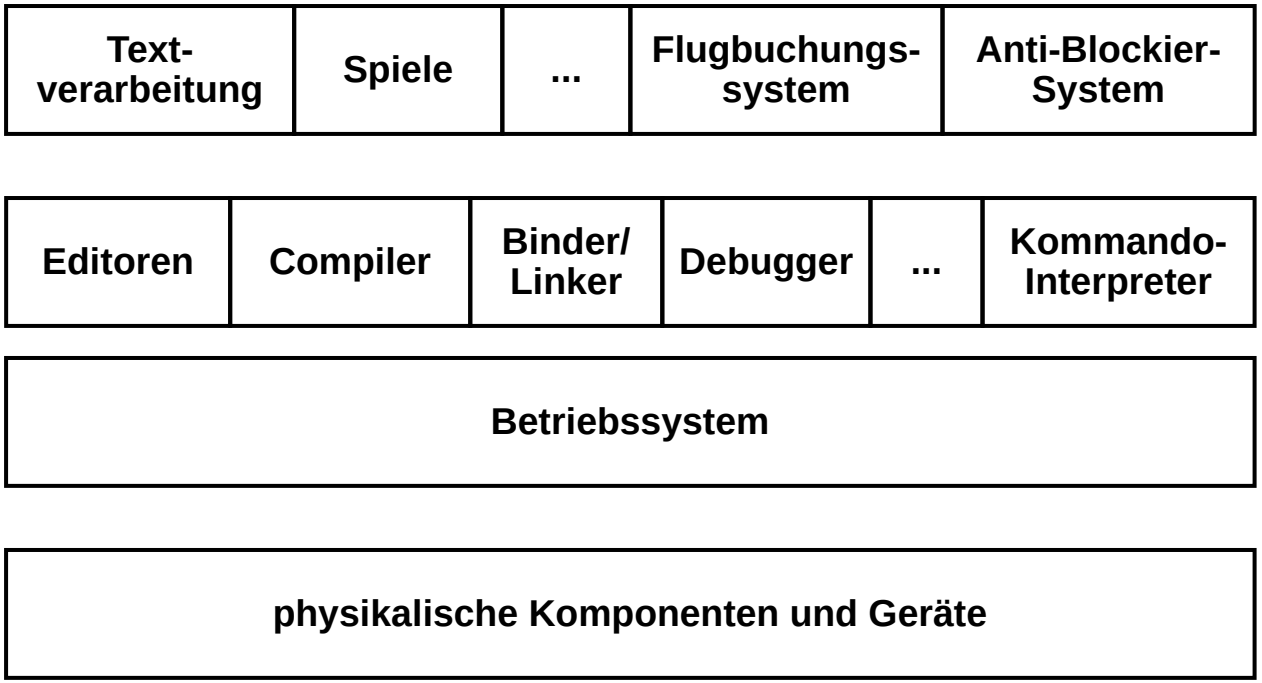
Rechnerarchitektur **engl.: computer architecture**

Baukunst, Baustil,
auch einzelnes Produkt der Baukunst

- **Ziele**
  - Entwicklung von Universalrechnern (für breite Anwendungsgebiete)
  - Entwicklung von Spezialrechnern (z.B. für Handy)
  - Weiterentwicklung der Konstruktionsprinzipien von Rechensystemen
- **Materialien des Rechnerarchitekten:**
  - Bauelemente (Steine, Pfannen, ...): logische Funktionseinheiten, integrierte Schaltkreise (Integrated Circuits, ICs).
  - Teilsysteme/Subsysteme (Zimmer, Dach, ...): Zusammensetzungen einer Menge von Bauelementen (z.B. Prozessoren, Speicher, Verbindungseinrichtungen, E/A-Systeme).
- **Typische Nebenbedingungen:**
  - hohe Leistung (Performance) bei niedrigem Preis
  - Fehlertoleranz, geringer Energieverbrauch ...

Grobe Schichtung eines Rechensystems

- **vgl. Kap. 1:**



Software (SW) eines Rechensystems

- **Anwendungsprogramme**
  - **zur unmittelbaren Lösung von Benutzerproblemen**
- **Systemprogramme**
  - **zur Verwaltung des Betriebs eines Rechners**
  - **zur Unterstützung der Entwicklung von Anwendungen (vgl. Vorlesung "Programmieren 1")**
- **Das *Betriebssystem* (BS, engl. *operating system*, OS) ist das grundlegendste aller Systemprogramme. Es verwaltet alle Betriebsmittel eines Rechensystems und kontrolliert ihre Zuteilung und offeriert den Nutzern des Rechensystems eine virtuelle Maschine, die einfacher zu verstehen und zu programmieren ist als die unterlagerte Hardware.**

Def

  - **Betriebsmittel (engl. *resources*) sind dabei alle zuteilbaren und benutzbaren Hardware- und Software- Komponenten eines Rechensystems.**
  - **BS-Beispiele: UNIX, Windows 10, IBM MVS, OS/2, VMS, OSEK, ...**

Das Betriebssystem als "virtuelle Maschine"

- **Ziele:**
  - **Abschirmen des Programmierers vor der Komplexität der Hardware durch eine SW-Schicht (das Betriebssystem) über der "nackten" HW.**
  - **Schnittstelle zum Programmierer soll aus einfachen Abstraktionen auf hohem Niveau mit entsprechenden Operationen bestehen.**
  - **Die Schnittstelle soll einfacher zu verstehen und langlebiger sein als die sich schnell entwickelnde, technologieabhängige Hardware-Schicht.**
- **Typische Abstraktionen:**
  - **Prozess (Programm in Ausführung) mit virtuellem Adressraum und Threads (elementare Aktivitätsträger, "Leichtgewichtsprozesse")**
  - **Datei (Hintergrundspeicher-Abstraktion)**
  - **Speichersegment (Hauptspeicher-Abstraktion)**
  - **Nachrichten**
  - **Synchronisationsobjekte**

Das Betriebssystem als "Betriebsmittelverwalter"

- **Aufgabe des BS in dieser alternativen Sichtweise:**
  - **Durchsetzen einer geordneten und kontrollierten Zuteilung (Allokation, engl. *allocation*) der Prozessoren, Speicher, E/A-Geräte und sonst. Betriebsmittel an die sich in Ausführung befindlichen Programme.**
  - **Gemeinsame Benutzung "teurer" HW-Betriebsmittel ermöglichen (z.B. Streamer, Laserdrucker).**
  - **Gemeinsame Benutzung logischer Betriebsmittel (z.B. Dateien) ermöglichen, um Kooperation der Benutzer zu unterstützen.**
  - **Schutz aller Betriebsmittel vor unberechtigter Benutzung und Zusicherung von Geheimhaltung der Information, wenn der Benutzer dies wünscht.**
  - **Vermitteln im Falle von Betriebsmittelanforderungen von Programmen und Benutzern, die im Konflikt zueinander stehen.**
  - **Abrechnung der Kosten der Betriebsmittelnutzung (*Accounting*).**

Vorlesung „Betriebssysteme“

„Das Betriebssystem als virtuelle Maschine“

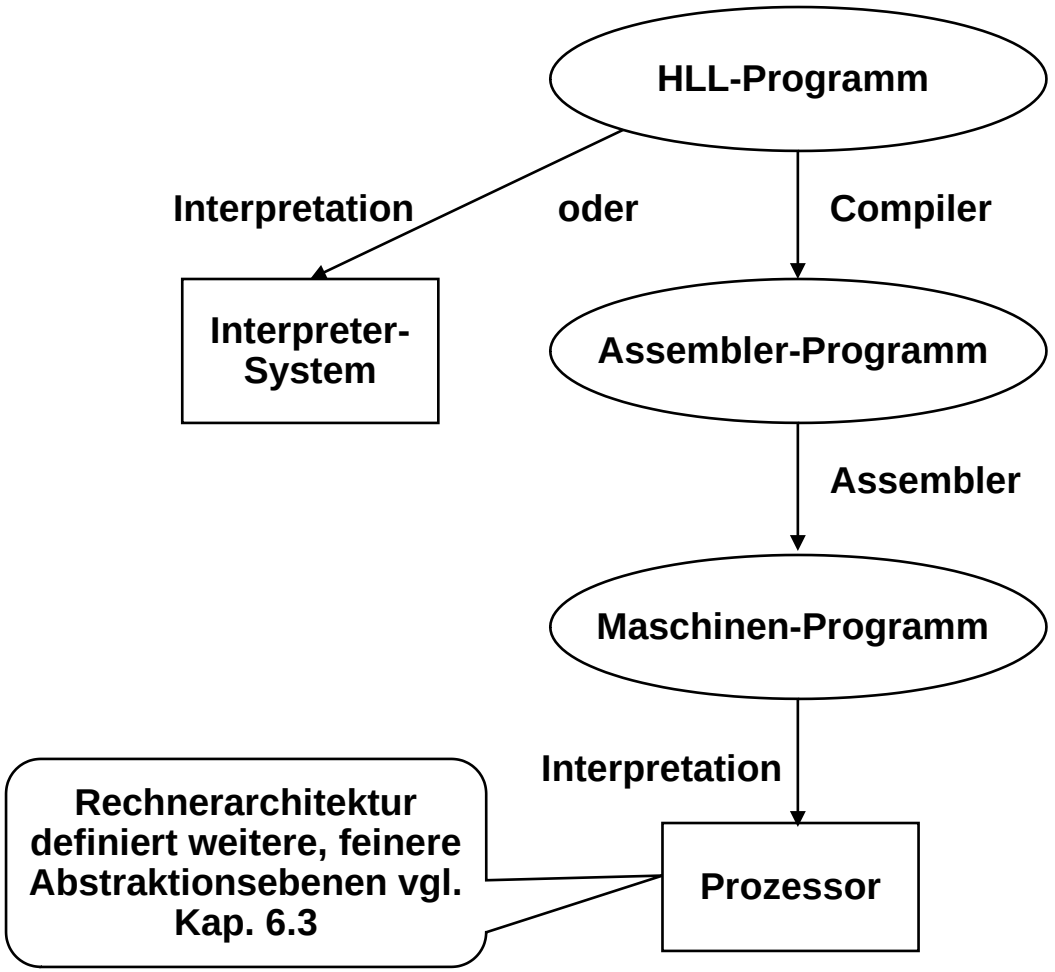
- **Die Vorstellung solcher Abstraktionen und ihre Realisierung im Betriebssystem ist Gegenstand der Vorlesung "Betriebssysteme" (3. Semester).**

„Das Betriebssystem als Betriebsmittelverwalter"

- **Die Vorstellung entsprechender Verfahren und Mechanismen ist ebenfalls Gegenstand der Vorlesung "Betriebssysteme".**

Verfeinerte Schichtung eines Rechensystems

- **aus Programmierer-Sicht:**



Ziel dieses Kapitels

- **Vorstellung von Ansätzen für die Architektur von Rechensystemen**

- **Konzentration auf**
  - **das grundlegende von-Neumann-Architekturmodell (6.2)**
  - **Aufbau von Prozessoren (6.3), basierend auf einfachen digitalen Schaltungen, wie sie in Kap. 5.3 betrachtet wurden.**
  - **Aufbau von Systemen im Großen (6.4).**

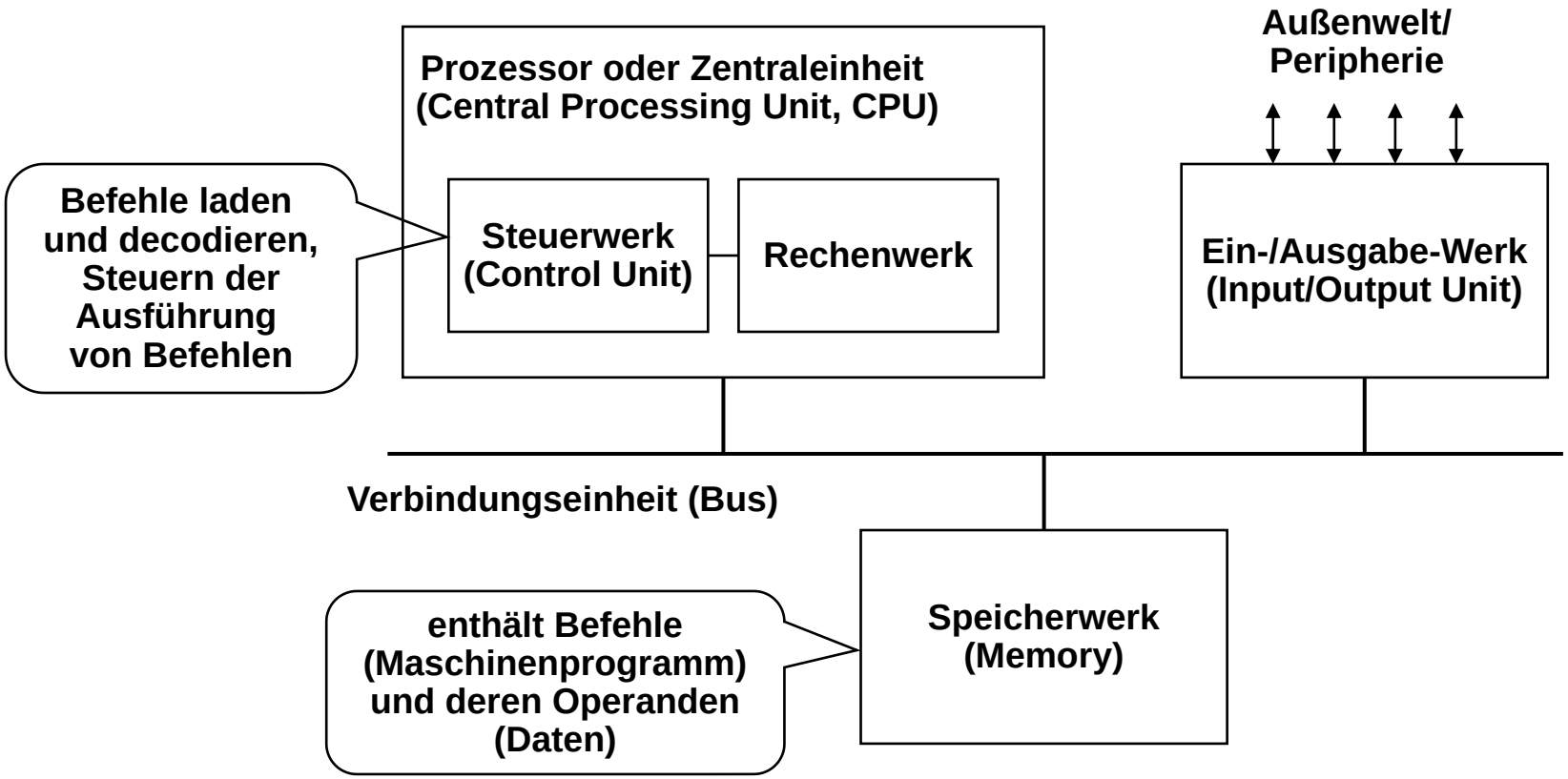
- **Detaillierte Behandlung vieler hier nur angerissener Ansätze erfolgt in der Veranstaltung „Betriebssysteme" (3. Semester).**

6.2 Von-Neumann-Architektur

- John von Neumann formuliert 1945 in einem Bericht die Idee vom im Hauptspeicher gehaltenen Programm zusammen mit den zu verarbeitenden Daten (vgl. Kap. 1).
- Darauf aufbauender Bericht von Burks, Goldstine und von Neumann (1946) gilt als wegweisend für die Rechnerarchitektur.
- Die von-Neumann-Architektur war ausgerichtet auf den *minimalen Hardware-Aufwand*.
- Heutige Architekturen haben das von-Neumann-Operationsprinzip beibehalten, aber
  - hoher Hardware-Aufwand innerhalb eines Prozessors ist heute vertretbar wegen niedriger Kosten durch Hochintegration und wird zur Leistungssteigerung genutzt (vgl. 6.3).
  - hoher Hardware-Aufwand durch Vervielfachung von Teilsystemen (z.B. Prozessoren) wird für Höchstleistungssysteme ("Supercomputing") und zur Realisierung von hochverfügbaren Systemen eingesetzt (vgl. 6.4).

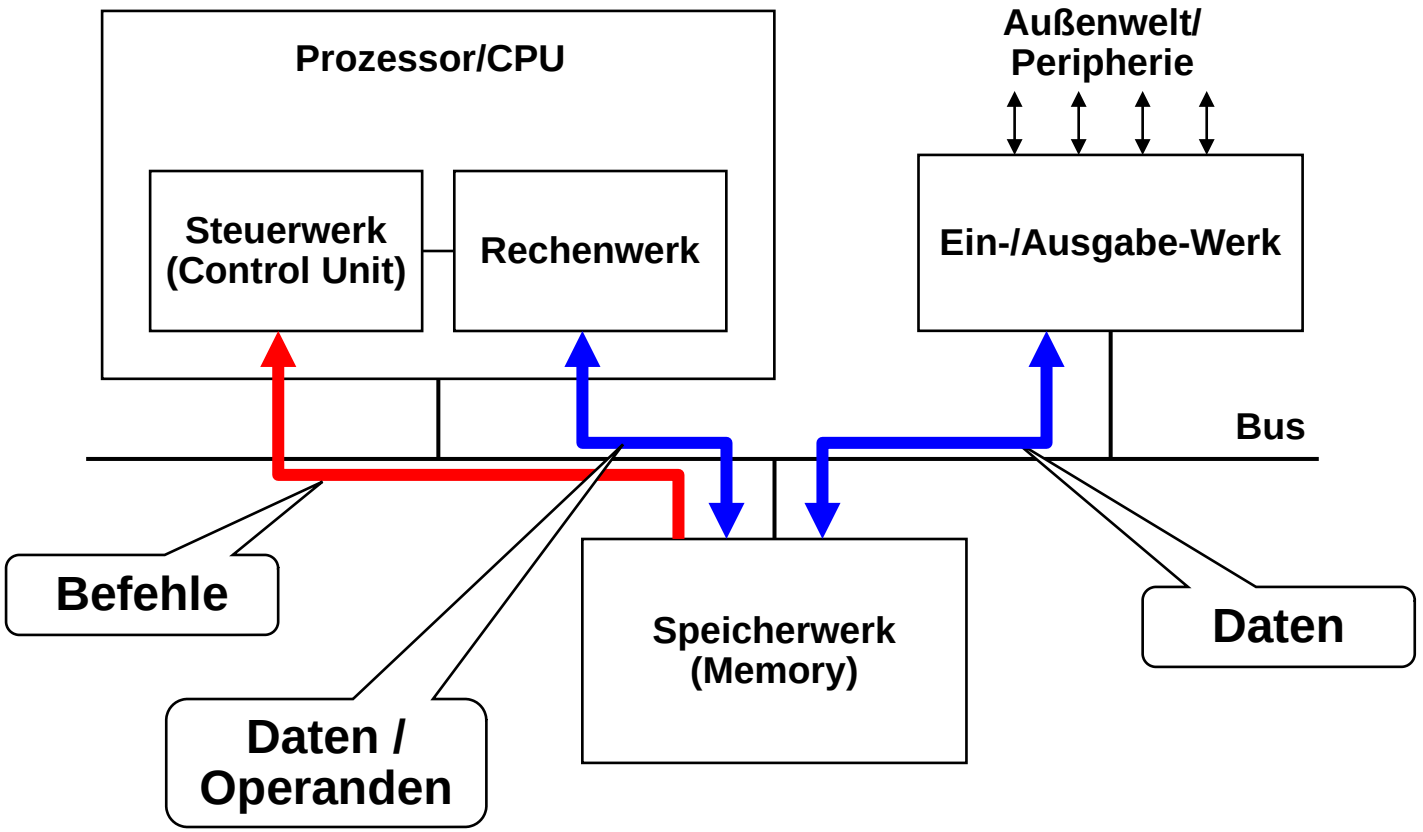
Überblick

- **Funktionseinheiten der von-Neumann-Architektur**



Überblick (2)

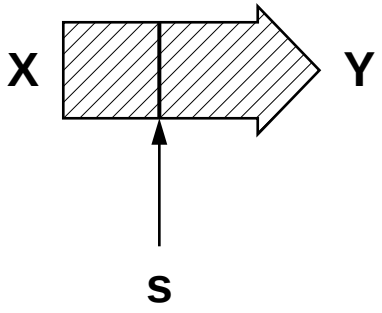
- **Fluss von Daten und Befehlen**



Bus

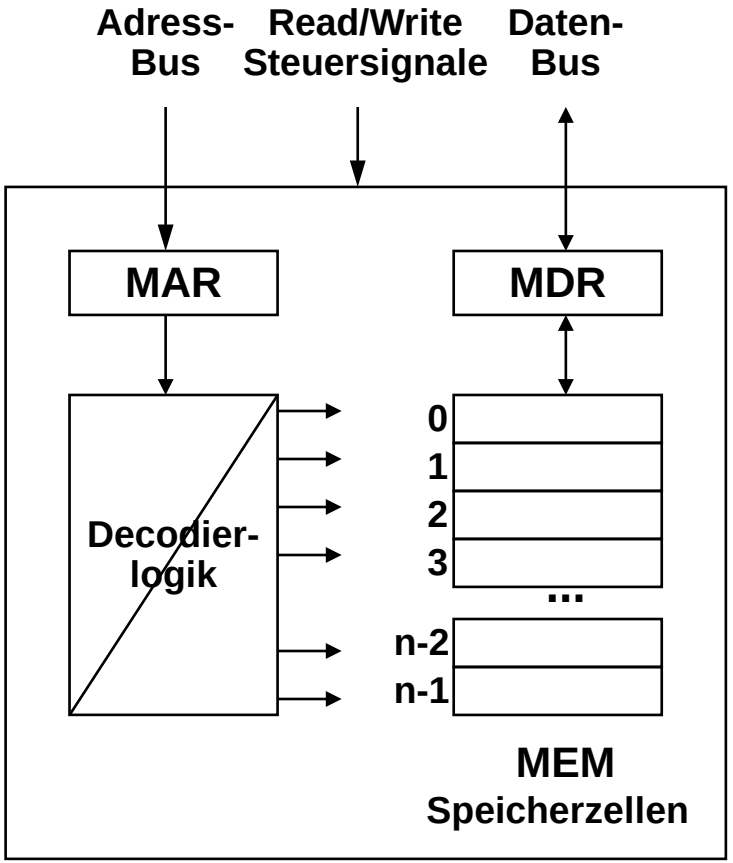
- parallele Verbindungsstruktur zwischen Werken (Prozessor-interne Busse)
- Nutzung als
  - Adressbus
  - Datenbus
  - Bus für Steuersignale
- Einbeziehung von Toren (vgl. 5.3):

$X=(x_1,...,x_n)$
$Y=(y_1,...,y_n)$



Speicherwerk

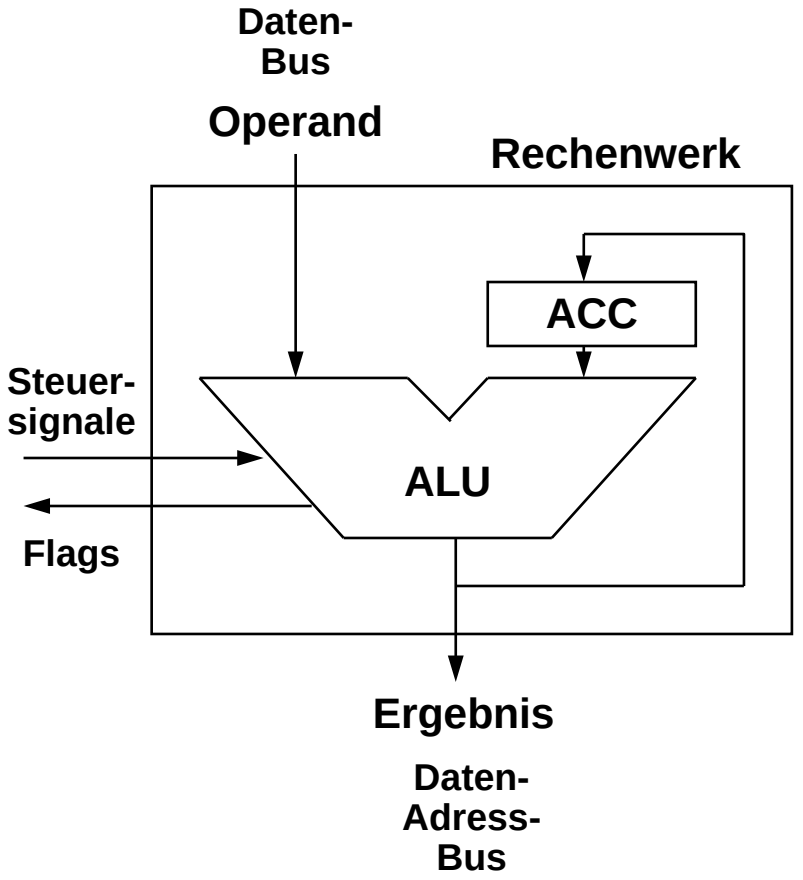
- enthält
  - Speicher-Adressregister, engl. Memory Address Register MAR
  - Speicher-Datenregister, engl. Memory Data Register MDR
  - Folge von gleich großen Speicherzellen (MEM)
- Die Speicherzellen besitzen fortlaufende Adressen 0...n-1.
- Jede Speicherzelle kann einen Maschinenbefehl oder ein Datum binär codiert aufnehmen. (Ein Datum kann auch eine Adresse sein. Die Art des Inhalts einer Zelle ist nicht erkennbar.)
- Lesen: Inhalt der durch MAR ausgewählten Zelle steht anschließend in MDR
- Schreiben: Inhalt des MDR wird in der durch MAR ausgewählten Zelle abgelegt



Speicherwerk

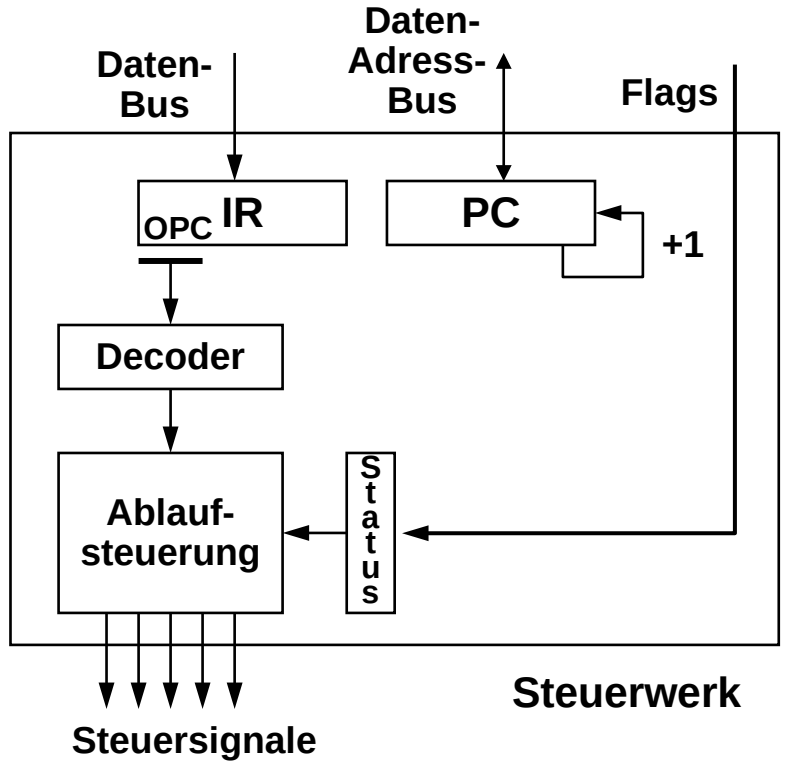
Rechenwerk

- enthält
  - Arithmetisch-Logische Einheit (ALU), vgl. Kap. 5.3
  - ein Ergebnis-Register, das *Akkumulator* (ACC) genannt wird.
- kann mittels der ALU alle arithm.-log. Operationen ausführen,
  ACC ist dabei immer einer der beiden Operanden (sog. Einadressbefehle).
- kann das Ergebnis einer arithm.-log. Operation im Akkumulator speichern.
- kann Ergebnis-Bedingungen als *Flags* anzeigen.
- Anmerkung: nachfolgende Architekturen enthalten insbesondere mehrere zusätzliche Register oder einen Satz (z.B. 16) von adressierbaren, allgemein verwendbaren Registern



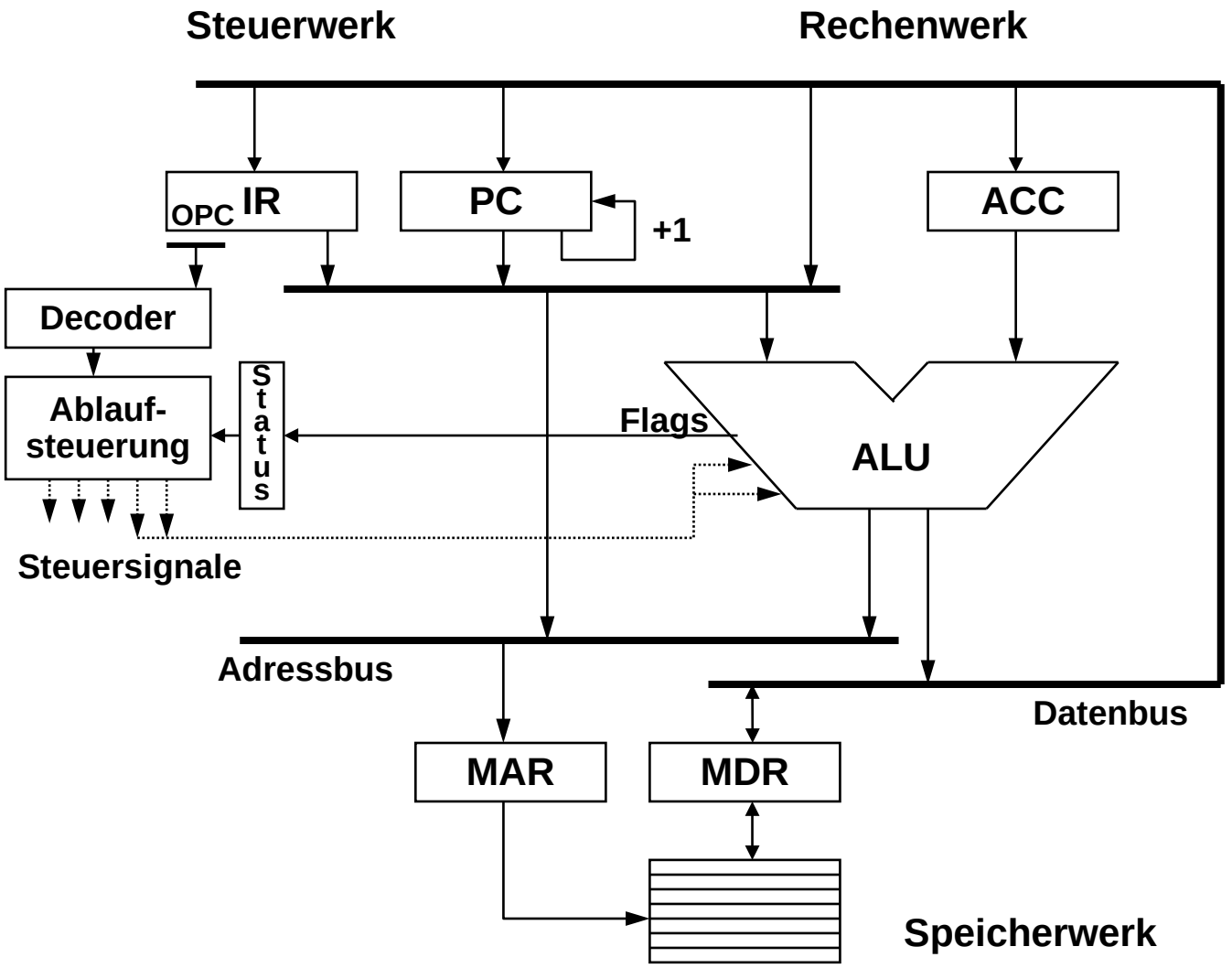
Steuerwerk

- enthält
  - Befehlszähler(-Register) PC, engl. *Program Counter*, mit Inkrementierer (+1)
  - Befehlsregister IR, engl. *Instruction Register*
  - Befehlsdecodierer
  - Ablaufsteuerung mit Statusregister



- Befehlszähler PC enthält die Adresse der Speicherzelle mit dem nächsten auszuführenden Befehl
- Befehlsregister IR enthält den in Ausführung befindlichen Maschinenbefehl. Jeder Befehl enthält einen Operationscode OPC und einen Operandenadressteil.
- Der Befehlsdecodierer entschlüsselt den Operationscode.
- Die Ablaufsteuerung realisiert eine zentrale Steuerschleife und generiert Steuersignale für alle Werke.

Zusammenfassung (ohne E/A)



Befehlsarten

- **Transportbefehle**
  - **Laden des Akkumulators mit dem Inhalt einer Speicherzelle oder eines Registers des EA-Werks.**
  - **Speichern des Inhalts des Akkumulators in eine Speicherzelle oder in ein Register des EA-Werks.**

- **Arithm.-log. Operationen**
  - **Add, Sub, NOT, AND, OR, Shift-Right, Rotate-Left, ...**
  - **Vergleichsbefehle (Setzen Flags, Übernahme in Statusregister)**

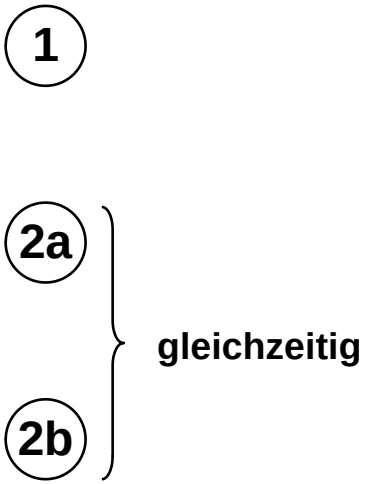
Befehlsarten (2)

- **Sprungbefehle (Programmverzweigungen)**
  - **unbedingter Sprungbefehl:**
    - **Adresse des nächsten auszuführenden Befehls ergibt sich aus dem Inhalt des Adressteils des gerade ausgeführten Befehls.**
    - **Man unterscheidet relative von absoluten Sprungadressen**
  - **bedingter Sprungbefehl:**
    - **Sprung erfolgt nur, wenn eine im Maschinenbefehl formulierte Bedingung, z.B. bzgl. der im Statusregister gehaltenen Flags erfüllt ist (=0?); typisch sind hier das Carry- und Zero-Flag**
    - **andernfalls wird der auf den gerade ausgeführten Befehl folgende ausgeführt (d.h. sequenziell auszuführende Befehle stehen in aufeinander folgenden Speicherzellen).**

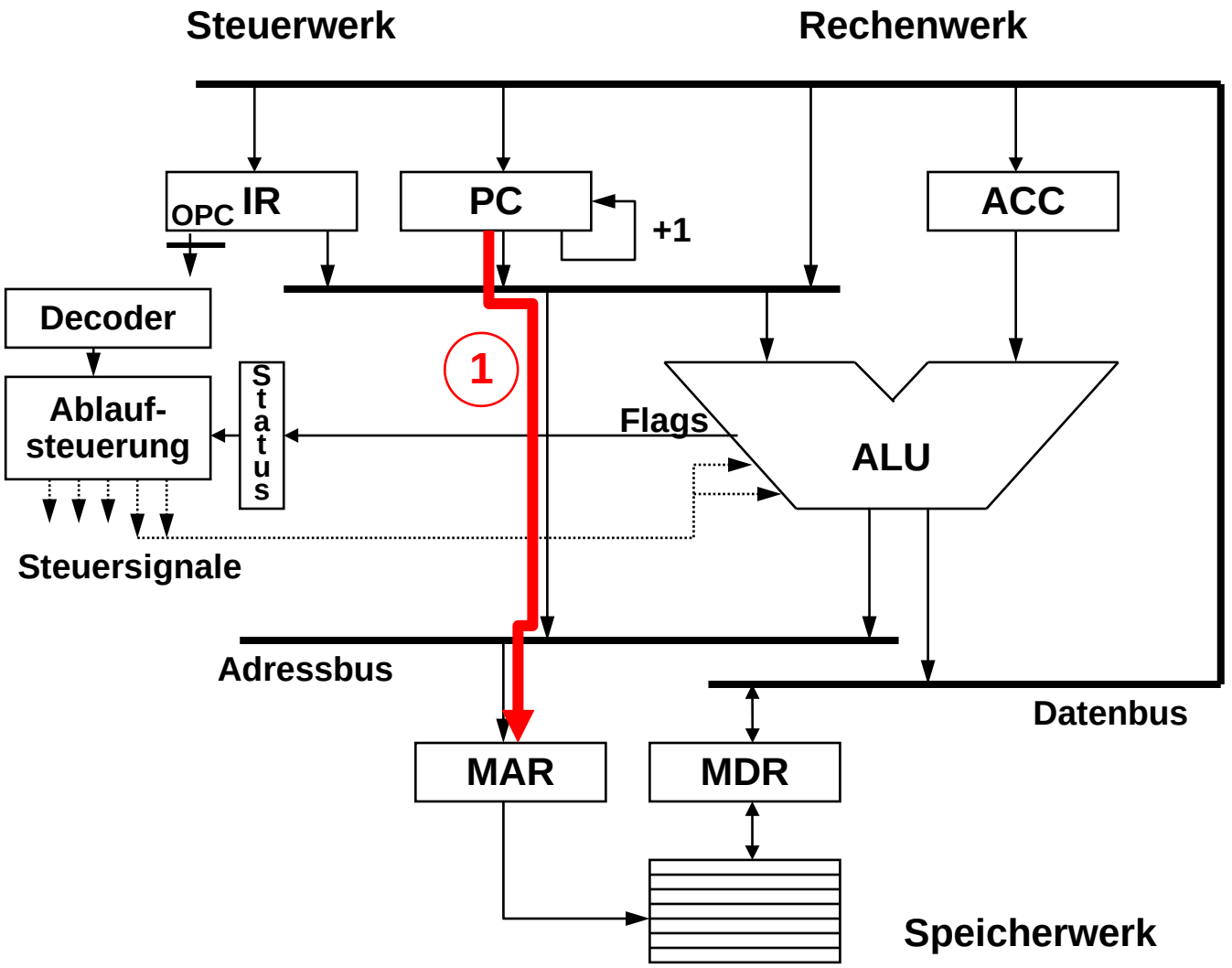
Arbeitsweise

- **Wichtig: Streng sequenzielles, zyklisches Vorgehen.**
- **Jeder Zyklus besteht aus einer *Befehlsholphase* und einer *Befehlsausführungsphase*.**

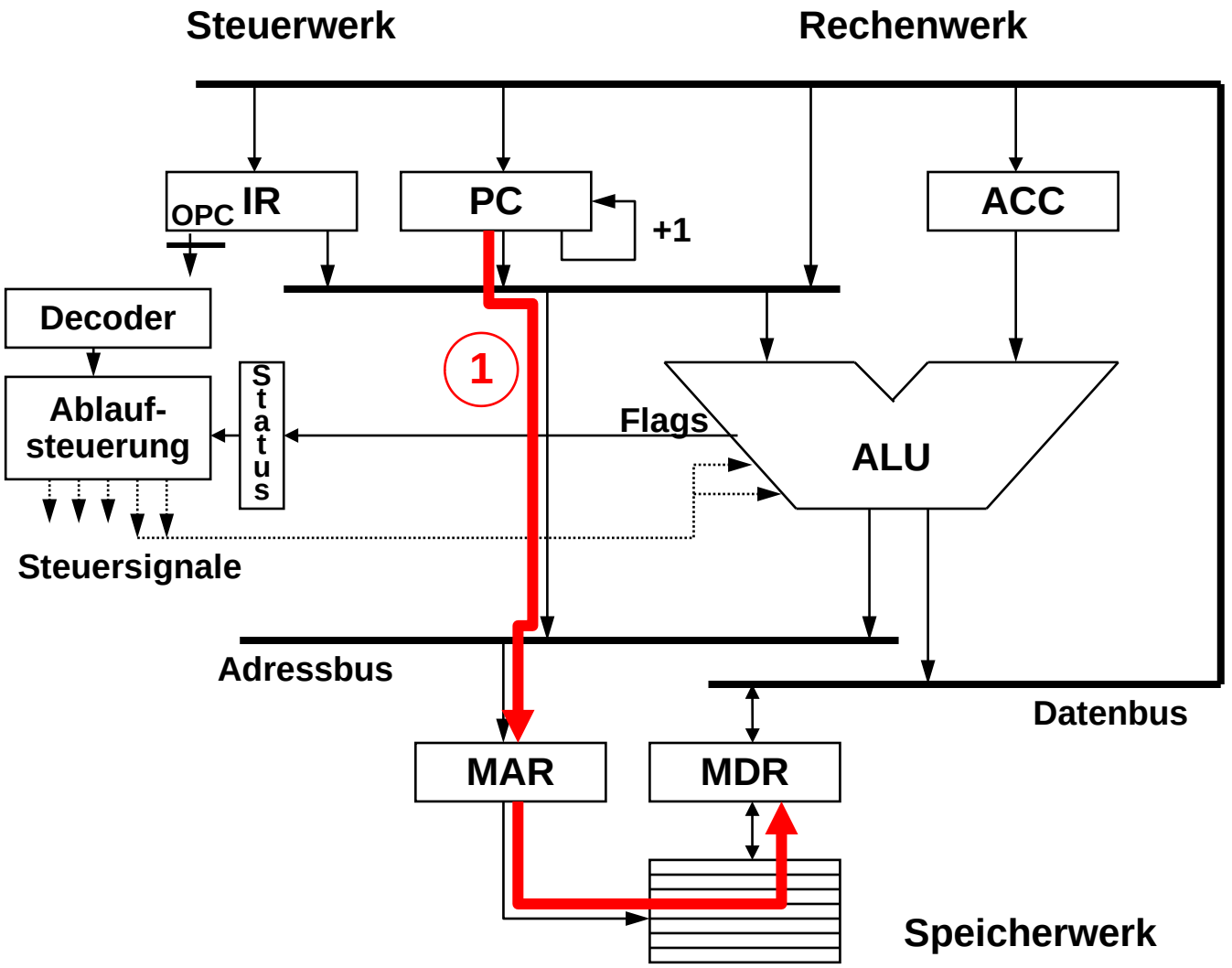
- **Befehlsholphase (engl. *Instruction Fetch*):**
  - **Lesen des Inhalts der Speicherzelle, deren Adresse im Befehlszähler PC steht, in das Befehlsregister.**
  - **Inkrementieren des Befehlszählers (= Adresse des nächsten Befehls entsprechend der Aufschreibung).**
  - **Decodierung (*Instruction Decode*) des Operationscodes im Befehlsregister IR.**



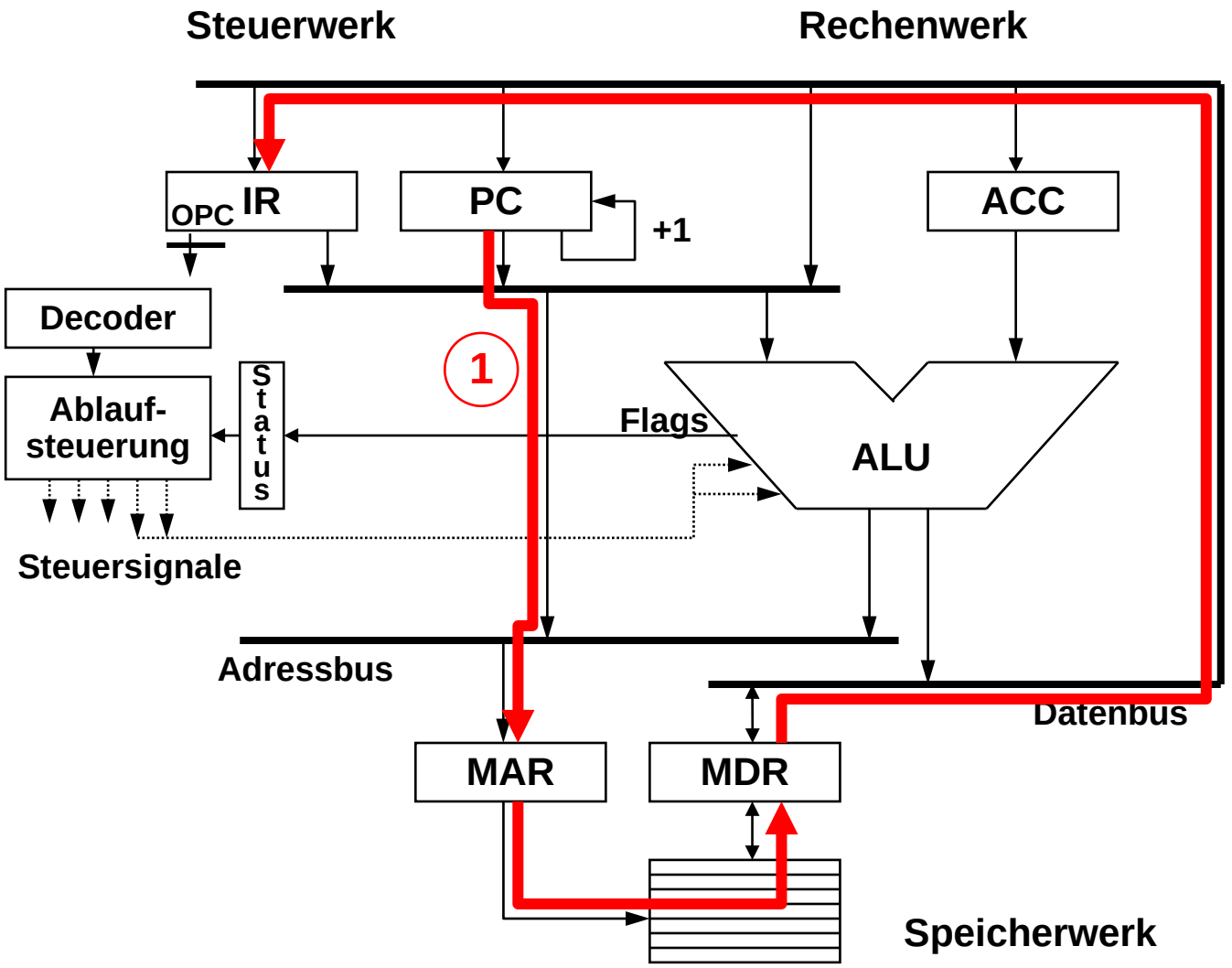
Ablauf der Befehlsholphase (a)



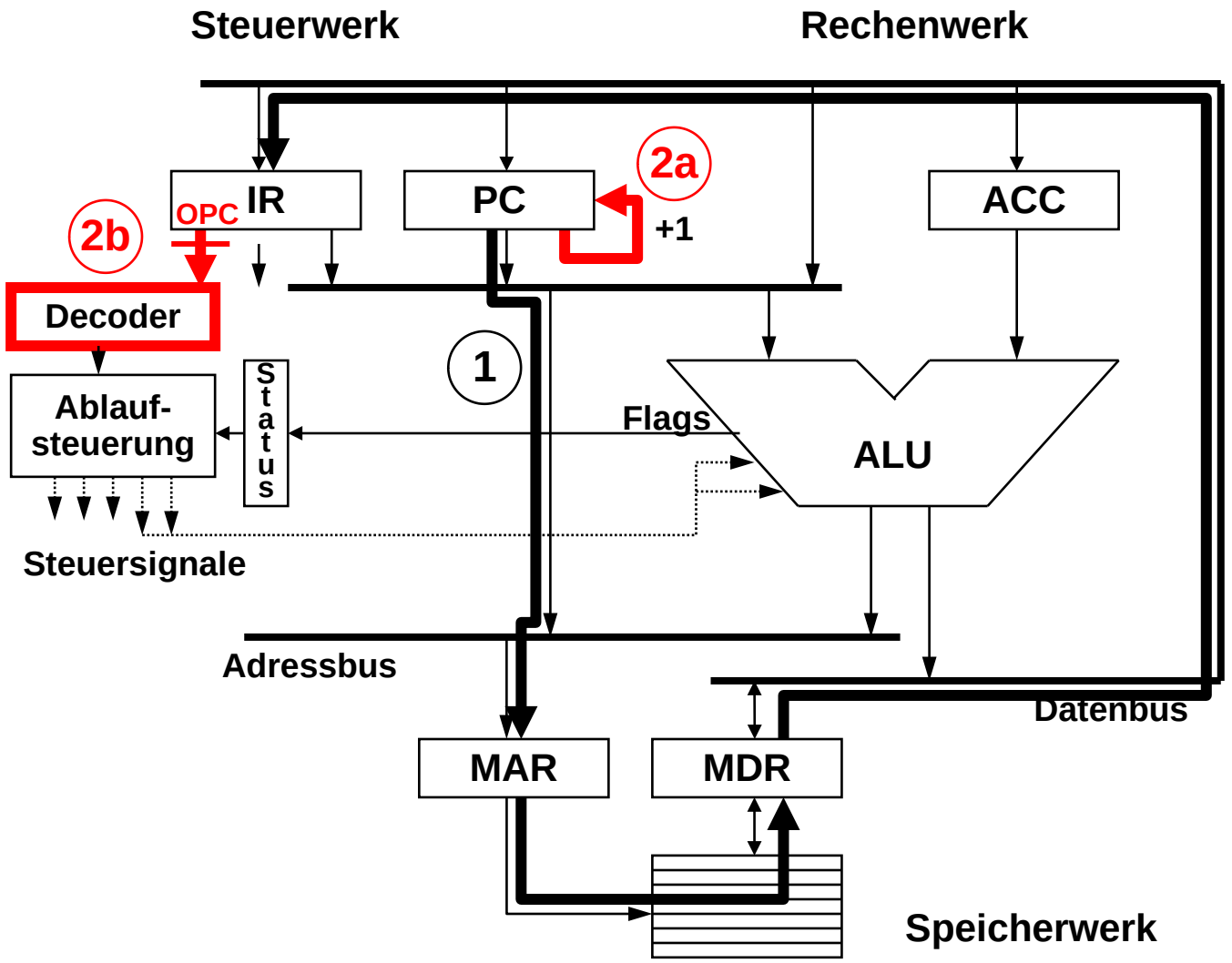
Ablauf der Befehlsholphase (b)



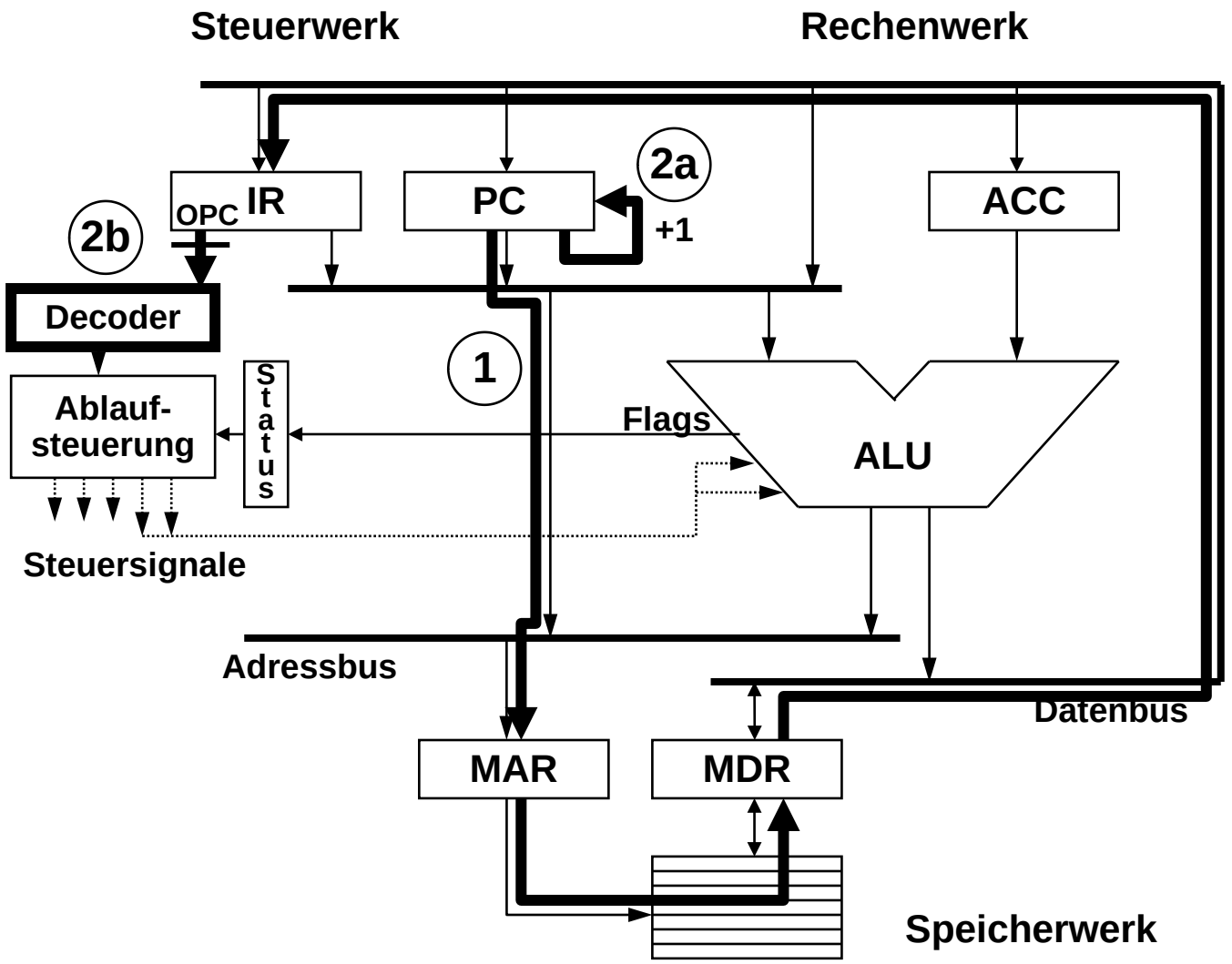
Ablauf der Befehlsholphase (c)



Ablauf der Befehlsholphase (d)



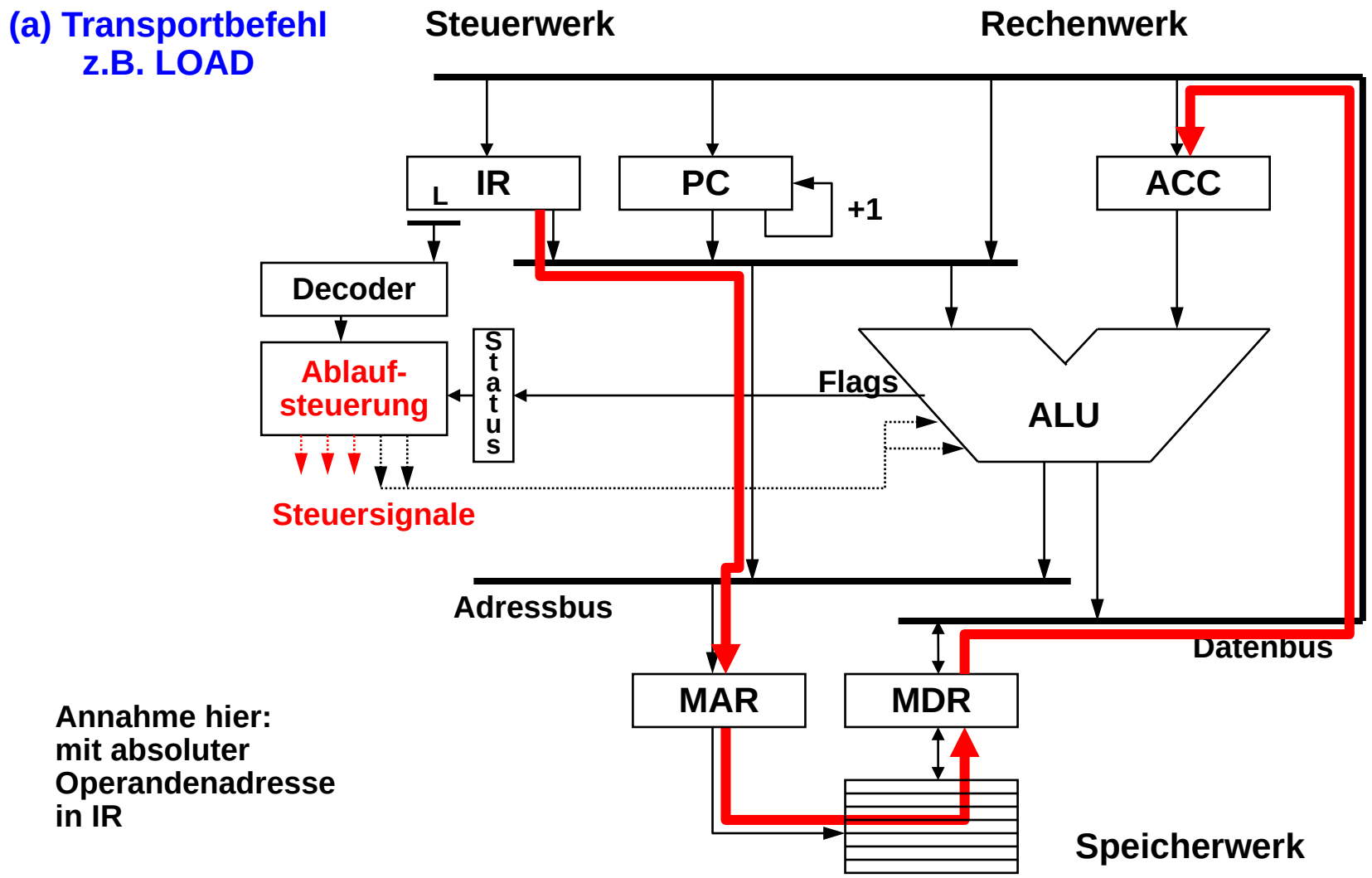
Ablauf der Befehlsholphase



Arbeitsweise (2)

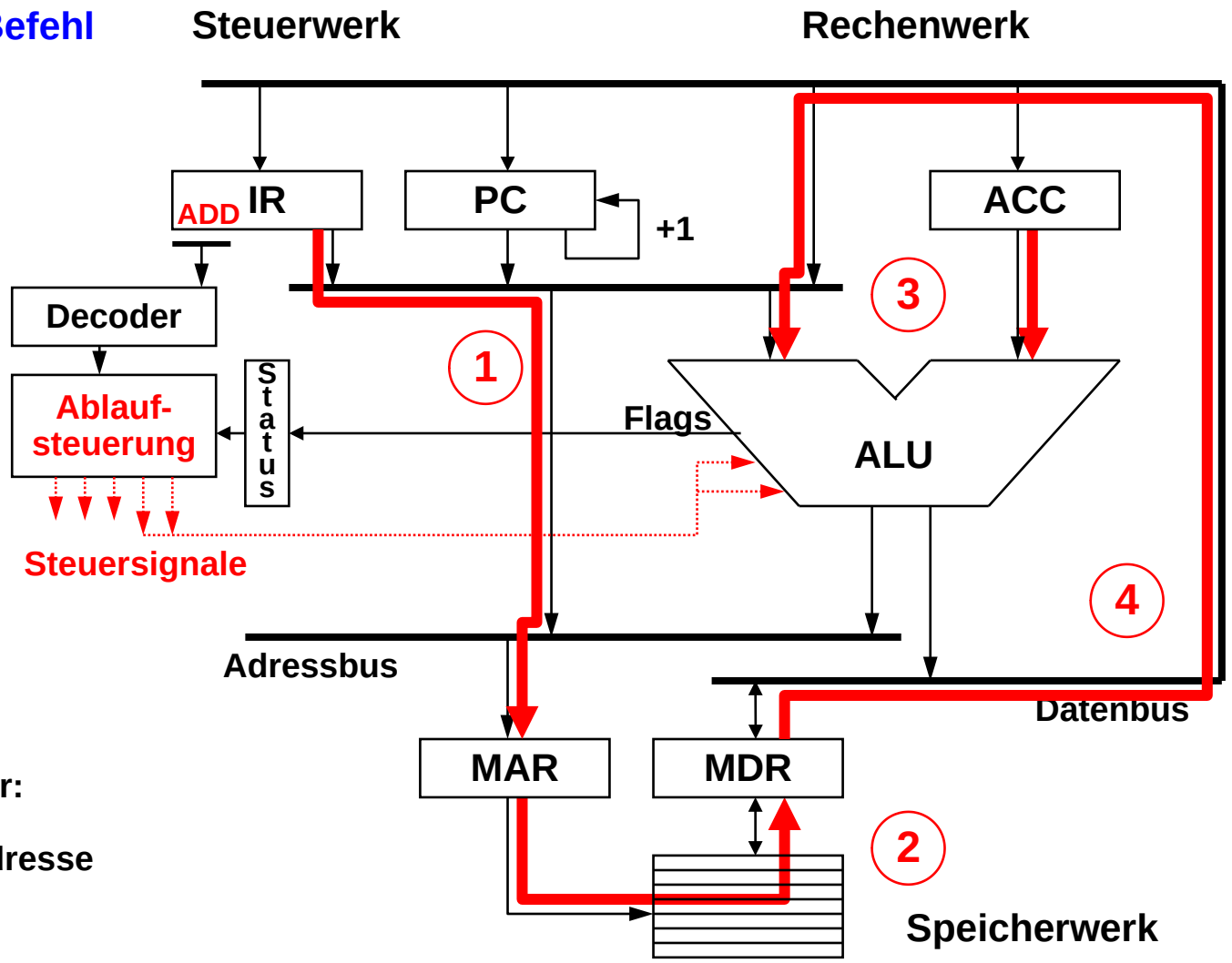
- **Befehlsausführungsphase (engl. *Instruction Execution*):**
  - **bei Transportbefehlen:**
    - **Berechnung der Operandenadresse aus dem Adressteil des Befehls.**
    - **Lesen des Operanden aus dem Speicher oder aus einem Register des E/A-Werks in den Akkumulator.**
    - **Analog für Schreiben.**
  - **bei arithmetisch-logischen Operationen:**
    - **Berechnung der Operandenadresse aus dem Adressteil des Befehls.**
    - **Lesen des Operanden aus dem Speicher.**
    - **Ausführung der gewünschten Operation mit dem gelesenen Operanden und dem Akkumulatorinhalt als 2. Operanden**
    - **Speicherung des Ergebnisses im Akkumulator.**
  - **bei Sprungbefehlen:**
    - **Überprüfen der Sprungbedingung bei bedingten Sprüngen.**
    - **Durchführung eines Sprungs durch Laden des Befehlszählers mit der berechneten Sprungzieladresse.**

Ablauf der Befehlsausführungsphase (a)



Ablauf der Befehlsausführungsphase (b)

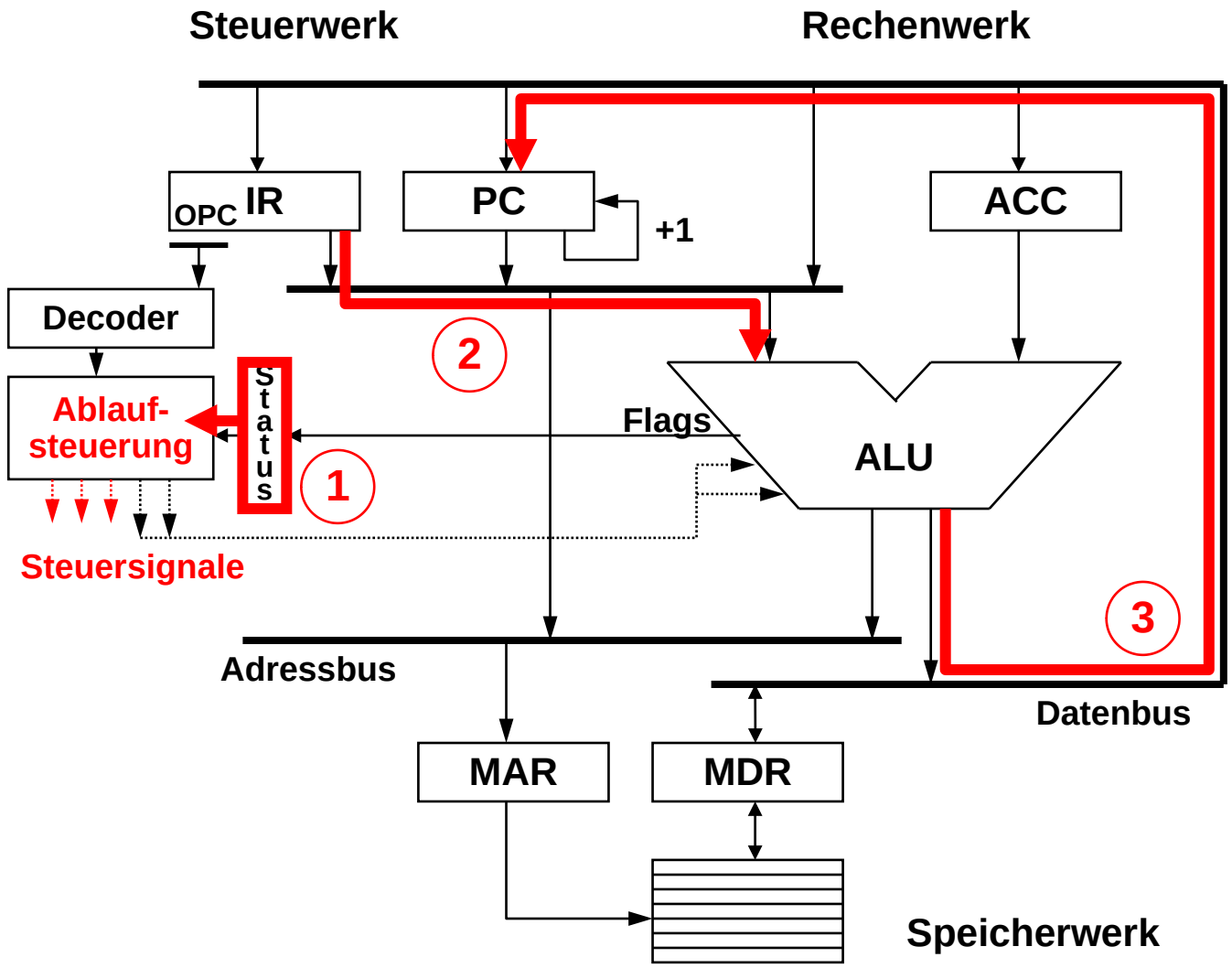
(b) Arith.-log. Befehl
z.B. ADD



Annahme hier:
mit absoluter
Operandenadresse
in IR

Ablauf der Befehlsausführungsphase (c)

(c) bedingter Sprungbefehl z.B. BRANCH



Instruktionen (Op Codes) am Beispiel Z80

- **Maschinencode vs. Assemblersprache am Beispiel des Z80-Mikroprozessors, Aufgabe: Register HL = DE * C**

Obj Code	Label	Statement	Kommentar
210000	MLTPLY	LD HL,0	(a) Doppelreg. HL mit 0 initialisieren
0608		LD B,8	(a) Register B mit 8 laden (8-Bit-Mult!)
CB39	MLOOP	SRL C	(b) Register C log. rechts schieben
3001		JR NC,NOADD	(c) Bedingter Sprung
19		ADD HL,DE	(b) Doppelreg. DE zu HL addieren
CB23	NOADD	SLA E	(b) Reg. E arithm. rechts schieben
CB12		RL D	(b) Carry Flag links in D einrotieren
10F5		DJNZ MLOOP	(c) B:=B-1, Schleife falls nicht Null

Wiederholung aus Kap. 1, mit Typ des Befehls (a), (b), (c)

von-Neumann-Flaschenhals

- Durch die beschriebene 2-phasige Arbeitsweise entsteht ein sequenzieller Kontrollfluss in der Abarbeitung von Maschinenbefehlen.
- Die Ausführung einer im Speicher abgelegten Folge von Maschinenbefehlen wird nur durch Sprungbefehle unterbrochen.
- Jede auszuführende arithm.-log. Operation impliziert mindestens
  - das Holen des Befehls aus dem Speicher,
  - das Holen des Operanden.
- Zusätzlich werden häufig Transportbefehle für das Kopieren von Daten zwischen Speicher und CPU (und damit auch zwischen verschiedenen Speicherzellen) sowie für den Zugriff auf gespeicherte Adressen notwendig.

⇒ Der Bus zwischen CPU und Speicher ist ein Engpass.

von-Neumann-Flaschenhals (2)

- **Der durch das von-Neuman-Operationsprinzip bedingte, streng sequenzielle Kontrollfluss zusammen mit dem Engpass zum Speicher werden als *von-Neumann-Flaschenhals* (*Bottleneck*) bezeichnet.**
- **Heutige, auf dem von-Neumann-Prinzip basierende Architekturen versuchen, die Effekte dieses Flaschenhalses durch geeignete Maßnahmen zu mildern (z.B. Caches, vgl. 6.3).**
- **Versuche, ganz andere Operationsprinzipien einzuführen, die den von-Neumann-Flaschenhals von vorn herein vermeiden, scheiterten (vgl. Vorlesung Betriebssysteme):**
  - ***Datenflussprinzip*:**
    - **Befehle werden ausführbar, sobald die Operanden verfügbar sind.**
    - **hoher Grad an Nebenläufigkeit möglich**
  - ***Reduktionsprinzip*:**
    - **Umformung von symbolischen Ausdrücken, „funktionale“ Verarbeitung**

6.3 Prozessorarchitektur

- **Wesentliche Entwicklungsfaktoren:**
  - **Weitestgehendes Beibehalten der von-Neuman-Architekturprinzipien**
  - **Laufend wachsende Integrationsdichte (Technologie-Fortschritt)**
    - **kleinere Transistorgröße ⇒ höhere mögliche Taktfrequenz ⇒ Leistungssteigerung**
    - **mehr Transistorfunktionen auf einem Chip. Beispiel:**
      1971: Intel 4004 4-Bit-Processor: 2.250 Transistor-Elemente
      1999: Intel Pentium III: ≈ $3*10^7$ Transistor-Elemente
      2019: AMD EPYC Rome: ≈ $4*10^{10}$ Transistor-Elemente
      ⇒ Basis für architektonische Maßnahmen zur Leistungssteigerung
  - **Erhöhung der Anzahl der Register**
    - **Akkumulator → 16 Mehrzweckregister → Register Files in RISC-Prozessoren**
    - **Zwei- oder Drei-Adress-Befehle, verschiedene Adressierungsarten**
  - **Erhöhung der Wortbreite**
    - **1974: typisch 4 Bit; 1999: typisch 32/64 Bit, aktuell: 64 Bit**

6.3 Prozessorarchitektur (2)

- **Wesentliche Entwicklungsfaktoren (Forts.):**
  - **Prozessorfamilien-Konzept**
    - **Kompatibilität zu früheren Mitgliedern (alte Programme bleiben ausführbar)**
    - **Beispiel „x86“: Intel 8086 (1978) bis Pentium, ...**
  - **Erhöhung der internen Parallelarbeit**

- **Beispiel aus der Praxis**

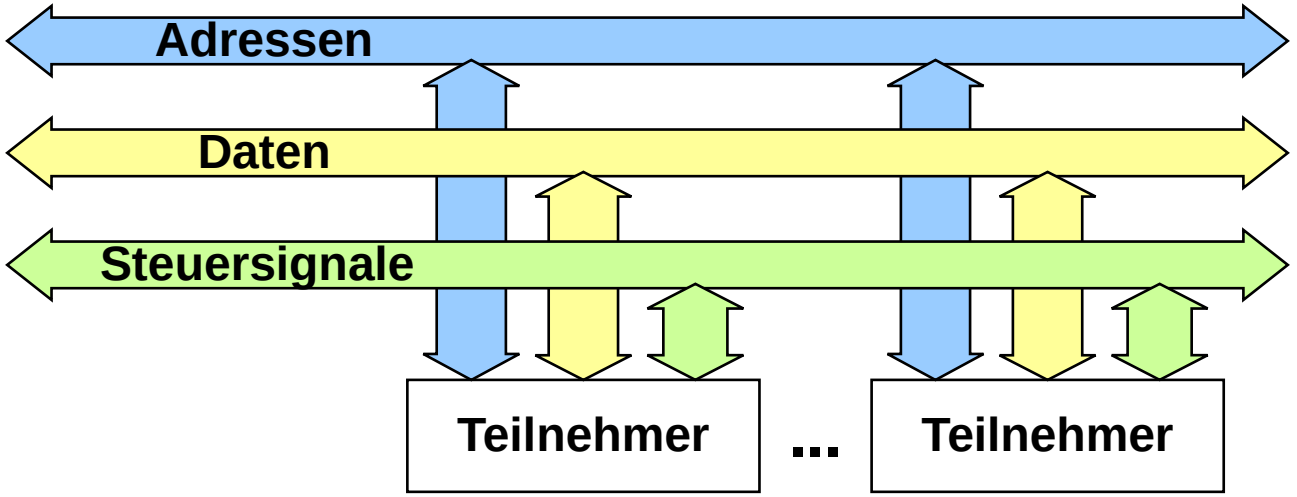
  - **1979: Der erste Rechner des Dozenten:**
    - **8-bit-CPU „Z-80“, 1.78 MHz, 16/48 kiB Hauptspeicher, Massenspeicher per 5.25"-FDD (SD, SS) = 180 kB pro Diskette, > 3000 DM**
  - **2020: Sein aktueller Notebook-Rechner – nach Kaufkraft günstiger!**
    - **64-bit-CPU „Core i7 8565U“, 1.8-4.6 GHz, 24 GiB Hauptspeicher, 1 TB SSD**
  - **Demnach Fortschrittsfaktoren über 41 Jahre bzw. pro Jahr,**

⇨

- **CPU-Takt * Busbreite * #Threads: 165393/1.34, Hauptspeicher: 1572864/1.43, HDD: 5.555.555/1.46** (Moore!)
  (aber: SSD vs. HDD, eigentlich noch höher)

Entwicklung von Systembussen

- Generalisierung der parallelen Verbindungsstruktur zwischen den Werken der von-Neumann-Architektur.
- Systembus enthält
  - Adressleitungen (z.B. 32-64)
  - Datenleitungen (z.B. 32-128)
  - Steuerleitungen (ca. 10-30)
  - gelegentlich werden Adress- und Datenbus auf denselben Leitungen im Zeitmultiplex-Verfahren (abwechselnde Nutzung) betrieben.



Entwicklung von Systembussen (2)

- Mehrere Teilnehmer können Buszyklen zum Transfer von Daten initiieren (*Multi-Master*).
  - Beispiel: DMA (Direct Memory Access) für E/A-Einheiten und Speicher/ Speicher-Operationen
- Einheit für die geordnete Buszuteilung an Teilnehmer heißt *Arbiter* oder *Arbitrierungseinheit*.
- Verschiedene Arten von Buszyklen (Einzel-Transfers, Burst-Mehrwort-Transfers)
- genutzt für System-Busse und für Peripherie-Busse
- Beispiele: VME-Bus, ISA-Bus, PCI-Bus; SCSI-Bus, ...

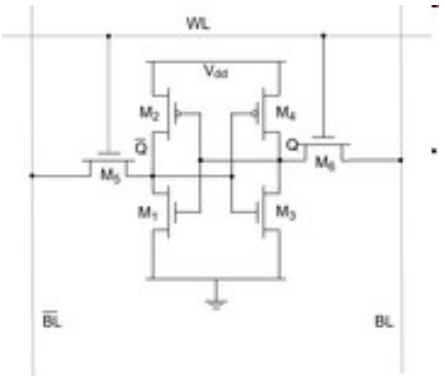
Entwicklung der Speicherwerke

- **Arten von Speichern:**
  - **RAM (Random Access Memory):**
    - beliebig oft wiederbeschreibbar,
    - wahlfrei (in beliebiger Reihenfolge) zugreifbarer Speicher
    - Inhalt ist nach Stromabschaltung verloren
  - **ROM (Read-Only Memory):**
    - beliebig oft nur lesbarer Speicher
    - Inhalt wird z.B. während der Fabrikation oder in speziellem Brennvorgang festgelegt
    - Inhalt bleibt nach Stromabschaltung erhalten
  - **EPROM (Erasable Programmable Read-Only Memory):**
    - seltene Schreibvorgänge
    - nach Schreiben Funktionalität eines ROM-Speichers
    - Speicherinhalt als Ganzes löschbar (durch UV-Bestrahlung)
    - Variante EEPROM (Electrically Erasable Programmable Read-Only Memory), heute verbreitet als Flash-Speicher etwa in USB-Sticks

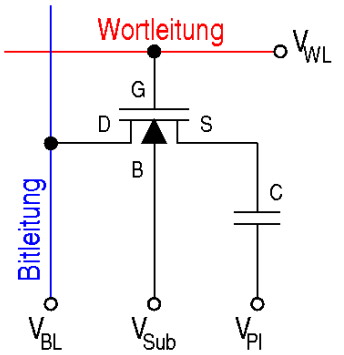
TMS2516
Q: Wikipedia

Entwicklung der Speicherwerke (2)

- Flip-Flops realisieren 1-Bit-Speicherzellen.
- Speicher aus Flip-Flops
  - werden als statische Speicher (SRAM) bezeichnet,
  - Anwendung z.B. für Caches (schnelle Zwischenspeicher, s.u.)
- Dynamische Speicher (DRAM)
  - i.d.R. genutzt für Arbeitsspeicher
  - geringerer HW-Aufwand (Eintransistorzelle, Kondensatorladung) ⇒ höherer Integrationsgrad
  - geringerer Stromverbrauch
  - aber: Zellen verlieren Inhalt ⇒ Notwendigkeit für periodischen Refresh
  - aber: zerstörendes Lesen ⇒ komplexe Lese/Schreib-Zyklen, langsamer



6-Transistor-SRAM-Zelle



1-Transistor-DRAM-Zelle

Q: Wikipedia

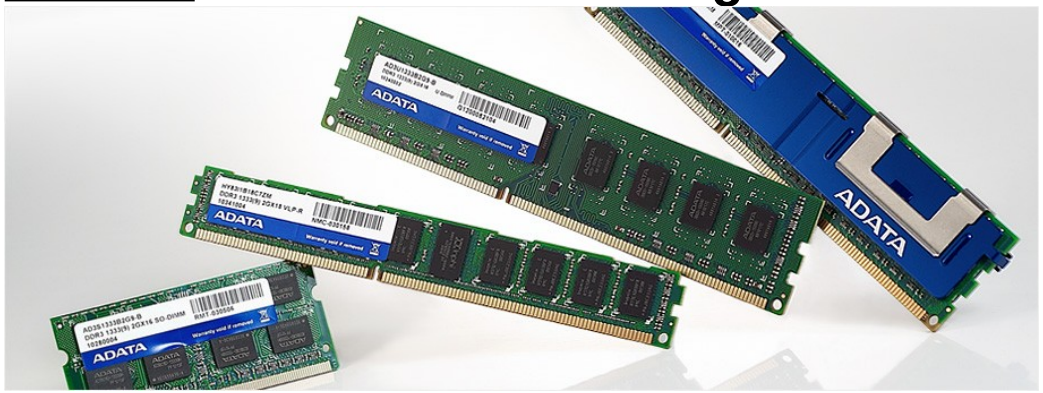
Entwicklung der Speicherwerke (3)

- **In einem Speicherbaustein werden "viele" (z.B. 2^{30} = 1 Gi) Speicherzellen i.d.R. matrixartig angelegt. Eine einzelne Zelle kann so über Zeilen- und Spaltenadresse ausgewählt werden.**

- **Bei gleicher Kapazität sind verschiedene Organisationsformen in Hinblick auf die von einem Chip gelieferte Wortbreite in Bits üblich:**
  - **z.B. 8Gi x 1, 1Gi x 8**

DDR4-SDRAM
Double Data Rate Synchronous DRAM, Gen. 4
SO-DIMM
(Small Outline) Dual Inline Memory Module

- **Speichermodule enthalten vollständige Ansteuerungslogik**



Entwicklung der Speicherwerke (3)

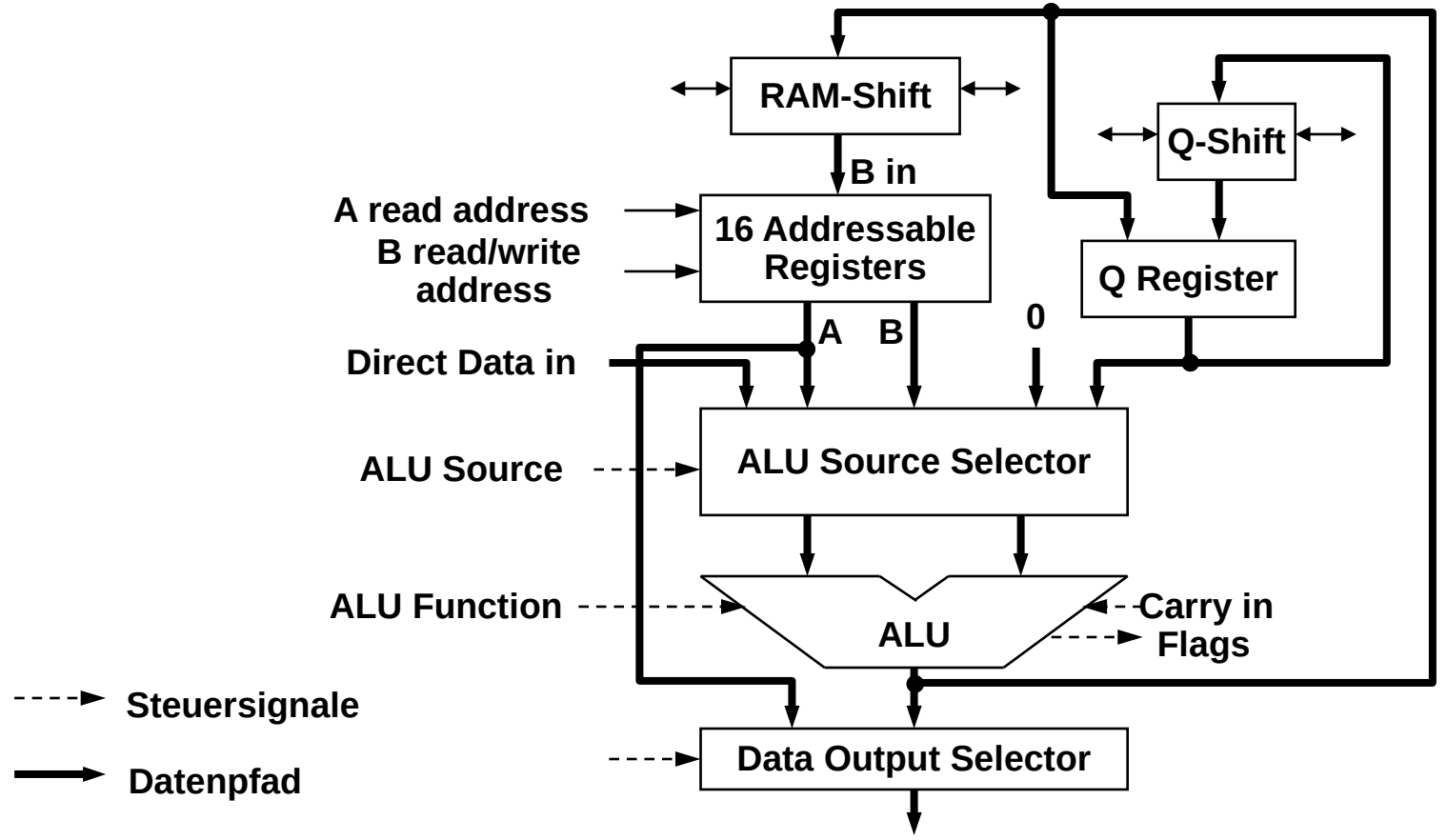
- **Kenngrößen für Speicher**
  - **Speicherkapazität**
  - **Zugriffszeit (engl. *access time*): Dauer bis Bereitstehen der Daten**
  - **Zykluszeit: einschl. Regenerierungszeit, u.U. viel länger!**
  - **Bandbreite: Datenrate vom/zum Speicher**
- **Leistungssteigerungsmaßnahmen für Speicher**
  - **Mehrwortzugriffe zur Erhöhung der Bandbreite**
  - **Speicherverschränkung (Interleaving)**
    - **Wörter mit fortlaufenden Adressen werden zyklisch in mehreren Speichermodulen (*memory banks*) angesiedelt.**
    - **Paralleles Arbeiten der Speichermodule verkürzt nach außen sichtbare Zugriffszeit des Gesamtspeichers**
  - **Integration von wenigen schnellen Schreib-Puffern**
    - **verkürzen sichtbare Schreib-Zugriffszeit**

Entwicklung von Rechenwerken

- Integration von zusätzlichen Registern zur Unterstützung der Multiplikation (Multiplikandenregister, Multiplikator-Register als "Verlängerung" des Akkumulators)
- Integration von mehreren allgemein verwendbaren Registern (Registered ALU, RALU)
- Integration von Shiftern
- Integration von Maskenregistern für Bitkettenmanipulation
- Integration zusätzlicher ALU-Funktionen
- Vergrößerung der Wortbreite (auf z.B. heute 32 oder 64 Bit)
- Integration von großen Register Files (z.B. 128 Register)
- Register-Renaming: mehrere Versionen des Register-Satzes
- Spezialisierte Rechenwerke für bestimmte Funktionen (Gleitpunktarithmetik, Dezimalarithmetik, Pixel-Operationen)
- Heute als eigenständige Funktionseinheiten innerhalb des Prozessors betrachtet (s.u.)

Beispiel: AMD 2901 4-Bit Slice (1974)

- RALU mit 16 Mehrzweckregistern in einem Chip
- Aneinanderfügen mehrerer Slices (Scheiben) zur Realisierung der gewünschten Prozessor-Wortbreite



Entwicklung von Steuerwerken

- Die Gesamtheit der vom Steuerwerk zu generierenden Steuersignale für alle Funktionseinheiten wird als **Steuerwort** bezeichnet.

- Für jeden Maschinenbefehl ist abhängig von dessen Operationscode eine spezifische **Folge von Steuerworten** durch die **Ablaufsteuerung** des Steuerwerks zu generieren.

- Realisierungsalternativen:
  - **fest verdrahtet** durch entsprechende Gatteranordnungen
  - **mikroprogrammiert**, d.h. für jeden Maschinenbefehl existiert ein Mikroprogramm einer internen, für den normalen Programmierer nicht sichtbaren "Mikro"-Maschine, dessen Ausführung die Folge der Steuerwörter erzeugt.

Entwicklung von Steuerwerken (2)

- **Ein mikroprogrammiertes Steuerwerk enthält**
  - **Mikroprogrammspeicher, i.d.R. als ROM-Speicher**
  - **Mikrobefehlszähler**
  - **Mikrobefehlssteuerwerk (Sequencer)**

⇒ **"Rechner im Rechner" (zusätzliche Abstraktionsebene)**

Mikroprogrammiertes Steuerwerk

01/2018:
Microcode-Patches für
Intel-Prozessoren sollen
Sicherheitslücken schließen

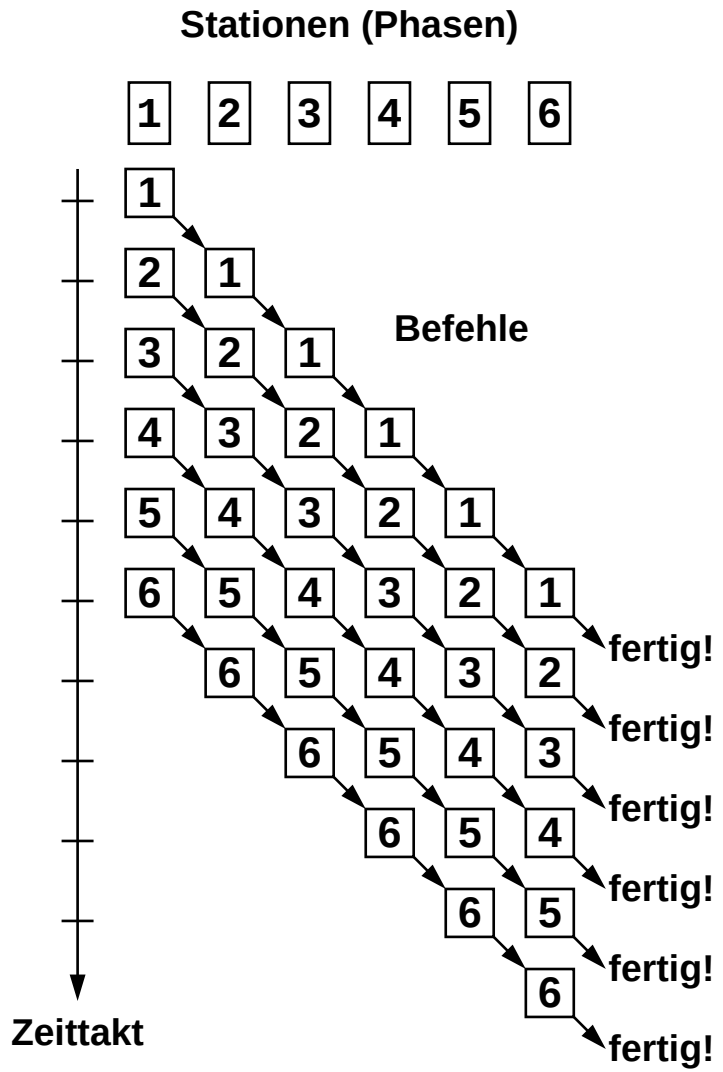
Klassische architektonische Maßnahmen

- Leistungssteigerung des Prozessors durch
  - spezialisierte Funktionseinheiten / Koprozessoren
    - z.B. für Gleitpunkt- oder Graphik-Operationen
  - ⇨ Pipelining (Fließbandprinzip)
  - ⇨ Vervielfachung von Funktionseinheiten

- Leistungssteigerung der Prozessor/Speicher-Schnittstelle
  - Prefetching
    - Heranschaffen von Speicherworten, bevor diese benötigt werden
  - ⇨ Caches
    - getrennt / gemeinsam für Befehle (Instruction Cache) und Daten (Data Cache)
    - für Adressübersetzungsinformation bei der Realisierung von virtuellem Speicher (Translation Look-aside Buffer in Memory Management Unit)

Pipelining (Fließbandverarbeitung)

- **Zu einem Zeitpunkt können sich *mehrere Befehle in unterschiedlichen Phasen der Ausführung* in der CPU befinden.**
- **Phasen orientieren sich an der Befehlsbearbeitung, im einfachsten Fall:**
  - **Befehl bearbeiten,**
  - **gleichzeitig nächsten Befehl holen und vorbereiten,**
- **Typische feinere Phaseneinteilung:**
  - **(1) Befehl holen**
  - **(2) Befehl decodieren**
  - **(3) Operandenadresse berechnen**
  - **(4) Operand holen**
  - **(5) Operation ausführen**
  - **(6) Ergebnis wegschaffen**



Pipelining (2)

- **Probleme**
  - **Hauptproblem bilden Sprungbefehle:**
    - **Die von der Aufschreibung her nachfolgenden und bereits teilweise bearbeiteten Befehle werden durch einen ausgeführten Sprung ungültig.**
    - **Pipeline muss entleert werden.**

    ⇒ Performance geht verloren !
  - **Phasen müssen gleich lang sein;**
    "Der Langsamste bestimmt das Tempo".
  - **Nicht alle Befehle benötigen alle Phasen (Löcher).**

Mehrfache Funktionseinheiten (Superskalarität)

- Ein Prozessor heißt ***superskalar***, wenn er über mehrere Verarbeitungseinheiten verfügt, die Instruktionen parallel ausführen. (Diese wenden intern i.d.R. selbst Pipelining an).
- Diese Nebenläufigkeit kann dazu führen, dass mehr als 1 Befehl pro Takt ausgeführt wird.

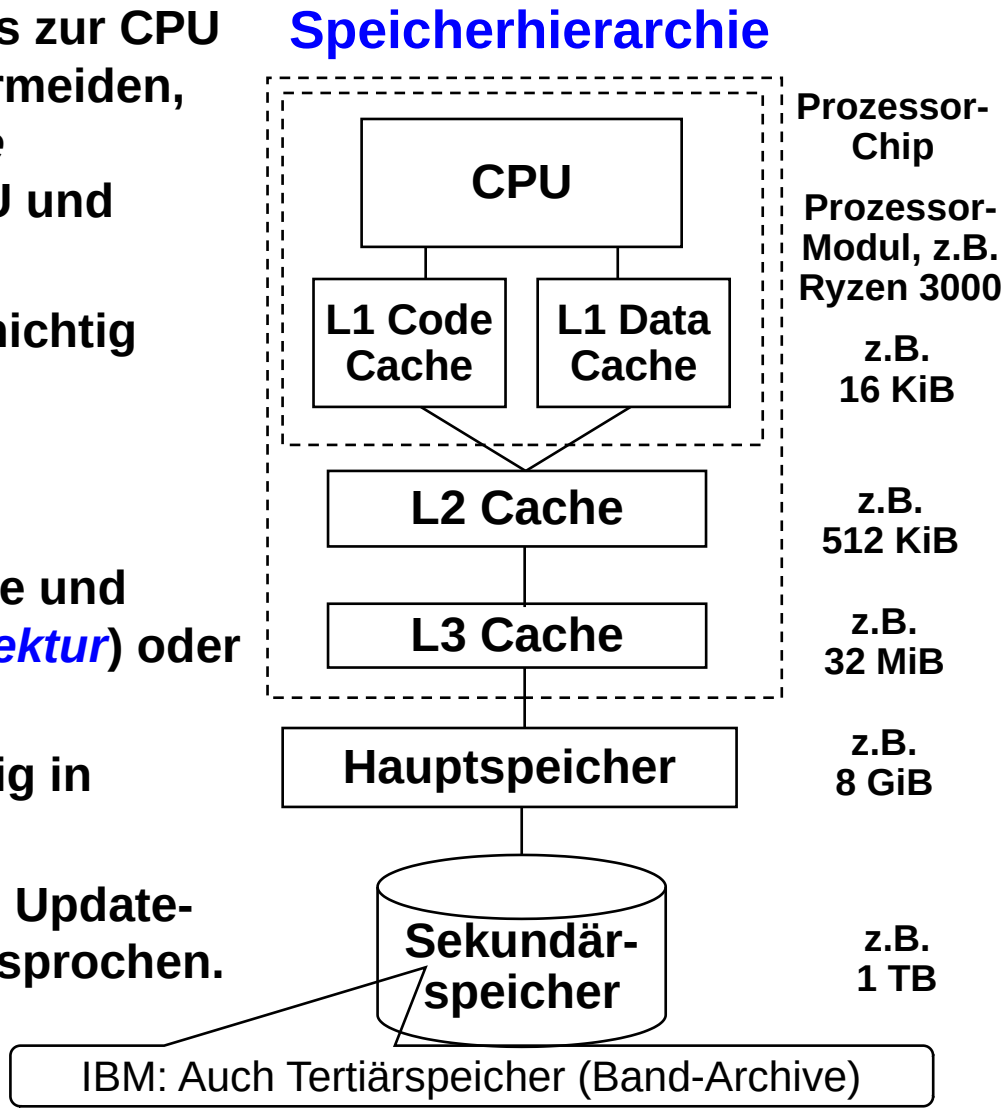
- Beispiele:

	Intel Pentium Pro	Motorola/IBM PowerPC
– Gleitpunkteinheiten:	2	1
– Integer-Einheiten:	2	3
– Memory Interface Units:	1	
– Load/Store-Einheit:		1
– Branch Processing Units:		1

- Relativ lose Kopplung der Verarbeitungseinheiten, "dezentrales" Steuerwerk

Caches

- Um Zugriffe auf den im Verhältnis zur CPU langsamen Hauptspeicher zu vermeiden, werden sog. Caches als *schnelle Zwischenspeicher* zwischen CPU und Hauptspeicher vorgesehen.
- Cache-Systeme können mehrschichtig aufgebaut sein:
  - First Level (L1) Cache,
  - Second Level (L2) Cache, ...
- Caches können getrennt für Code und Daten existieren (*Harvard-Architektur*) oder beides aufnehmen.
- Caches sind heute standardmäßig in Prozessor-Module integriert.
- Cache-Organisationsformen und Update-Strategien werden in LV „BS“ besprochen.



CISC / RISC

- CISC: Complex Instruction Set Computer
  - Beispiele: Intel 386, 486, Pentium; Motorola 680x0
- RISC: Reduced Instruction Set Computer
  - Beispiele: MIPS, SPARC, HP-PA, Alpha, PowerPC
- Abspaltung der RISC-Prozessor-Entwicklung in den 70er Jahren
- Zielrichtung: "Entschlacken der CPU":
  - wenige Instruktionsformate und Adressierungsmodi
  - weniger Instruktionen (<100),
  - Instruktionen besitzen kurze Ausführungszeiten mit 1-2 Takten durch fest verdrahtetes Steuerwerk statt zeit- und platzaufwendiger mikroprogrammierter Ablaufsteuerung,
  - Load/Store-Architektur, arithm.-log. Befehle haben nur Register-Operanden
  - große Anzahl von Registern (32)
  - Nutzung moderner Compiler-Technologie

Aktuelle Techniken

- ***Out-of-Order-Execution*:**
  - **Die Befehle werden im Prozessor in anderer Reihenfolge abgearbeitet, als dies vom Programmierer/Compiler vorgesehen wurde. Durch eine Datenflussanalyse werden Datenabhängigkeiten zwischen Befehlen korrekt beachtet.**
- ***Spekulative Ausführung* (*Speculative Execution*):**
  - **Eine *Verzweigungsvorhersageeinheit* (*Branch Prediction Unit*) "rät",**

    ob ein auszuführender Sprungbefehl zum Sprung führt oder nicht, und bereitet schon einige Befehle am Sprungziel in der Pipeline vor. Erfolgt der Sprung nicht, werden die vorbereiteten Ergebnisse verworfen.
- **Erweiterung ist das vorsorgliche *Ausführen beider Alternativen* eines bedingten Sprungs.**
- **Unterstützung durch moderne Compiler-Technologie.**

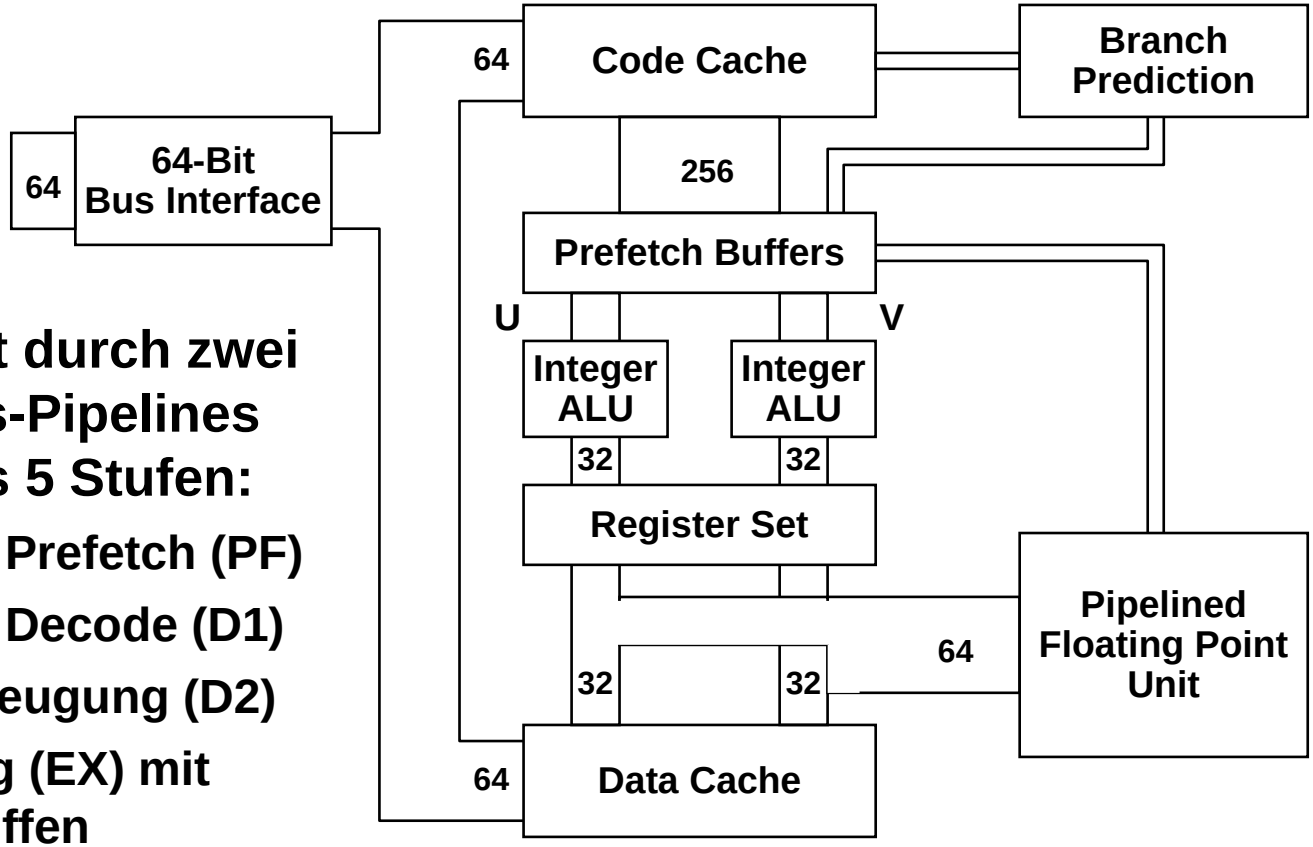
01/2018:
Sicherheitslücken „Meltdown“ und „Spectre“ betreffen insb. die *Branch Prediction Unit*

Aktuelle Techniken (2)

- **Zerlegen einer CISC-Instruktion in mehrere schnelle RISC-Instruktionen, die gleichzeitig auf verschiedenen Funktionseinheiten ausgeführt werden**
  - **Verwischen der Abgrenzung von CISC/RISC !**

- **Anwendung von SIMD-Techniken der Vektorverarbeitung in der Befehlsabarbeitung.**
  Beispiele:
  - **Pentium MMX-Befehle / 3DNow! (AMD)**
  - **Pentium III Streaming für Gleitpunkt und Integer**
  - **Core i3/5/7 2. Gen.: AVX – Advanced Vector Extensions**

Beispiel: Intel Pentium



Superskalarität durch zwei Integer Befehls-Pipelines U, V mit jeweils 5 Stufen:

- Instruction Prefetch (PF)
- Instruction Decode (D1)
- Adress-Erzeugung (D2)
- Ausführung (EX) mit Cachezugriffen
- Write-Back (WB) in Register Set

6.4 Systemarchitektur

Begriffe:

Def

- **Enge / lose Kopplung:**
  Zwei Prozessoren heißen *eng gekoppelt* (*tightly coupled*), wenn sie einen gemeinsamen physikalischen Speicher (*shared memory*) besitzen. Ansonsten können sie nur durch Nachrichtenaustausch (*message passing*) kommunizieren und heißen dann *lose gekoppelt*.
- **Homogenes / heterogenes System:**
  Sind alle Komponenten eines Systems identisch, so heißt das System *homogen*, ansonsten *inhomogen* oder *heterogen*.
- **Symmetrisches / asymmetrisches System:**
  Sind alle Komponenten eines Systems gleichwertig in Hinblick auf ihre Rolle im System, so heißt das System *symmetrisch*, ansonsten *asymmetrisch*.

Klassifikation von Architekturen nach Flynn (1972)

- **Betrachtet werden Instruktions- und Datenströme, auf denen die Instruktionen arbeiten, als Kennzeichen der Arbeitsweise einer Rechnerarchitektur:**
  - **SI / MI: Single / Multiple Instruction Streams (Programmzähler)**
  - **SD / MD: Single / Multiple Data Streams**

Klassifikation von Architekturen (2)

- **Klassen:**
  - **SISD: Single Instruction Stream / Single Data Stream:**
    - **traditioneller Uniprozessor (vgl. 6.1, 6.2).**
  - **SIMD: Single Instruction Stream / Multiple Data Streams:**
    - **Vektorrechner: parallele Ausführung einer Instruktion auf mehreren Datenelementen**
    - **klass. Architektur für Hochleistungsrechner.**
  - **MISD: Multiple Instruction Streams / Single Data Stream:**
    - **wird von den meisten Autoren als leer angesehen.**
  - **MIMD: Multiple Instruction Streams / Multiple Data Streams:**
    - **Systeme mit mehreren eng oder lose gekoppelten Prozessoren,**
    - **parallele und verteilte Rechensysteme**
    - **Ansatz der aktuellen Generation von Hochleistungsrechnern**

Vorteile von MIMD-Systemen

- **Wirtschaftlichkeit**
  - **bestes Preis/Leistungsverhältnis.**
  - **Gesetz von Grosch („4-fache Leistung für doppelten Preis“) gilt nicht mehr.**
  - **Benutzung vieler gleicher Standard-Systeme**
- **hohe nominelle Rechenleistung**
  - **1000 Prozessoren je 100 MIPS = 100 GIPS**
    - ⇒ **Zykluszeit eines Einprozessorsystems wäre 10 Picosek.**
    - ⇒ **Licht legt in dieser Zeit weniger als 3 mm zurück!**
    - ⇒ **Abwärme-Problem !**
- **Anpassbarkeit auf gegebene räumliche Verteiltheit von Anwendungen**
  - **z.B. Filialnetz**
  - **Fabrikautomation**
- **Schrittweise Erweiterbarkeit**
- **Potentiell hohe Verfügbarkeit, Fehlertoleranz**

Multiprozessor-/Multicomputer-Systeme

- **Multiprozessor-System:**
  - **Ein MIMD-Rechensystem mit eng gekoppelten, identischen Prozessoren heißt *(homogenes) Multiprozessor-System*.**
  - **Das grundlegende Kommunikationsmodell ist Memory Sharing.**
  - **Hauptaspekt: einfache Programmierbarkeit.**

- **Multicomputer-System:**
  - **Ein MIMD-Rechensystem mit lose gekoppelten, identischen Prozessoren heißt *Multicomputer-System* oder *Polyprozessor-System*.**
  - **Das grundlegende Kommunikationsmodell ist Message Passing.**
  - **Hauptaspekt: Skalierbarkeit für große Anzahlen von Prozessoren.**

6.4.1 Multiprozessor-Systeme

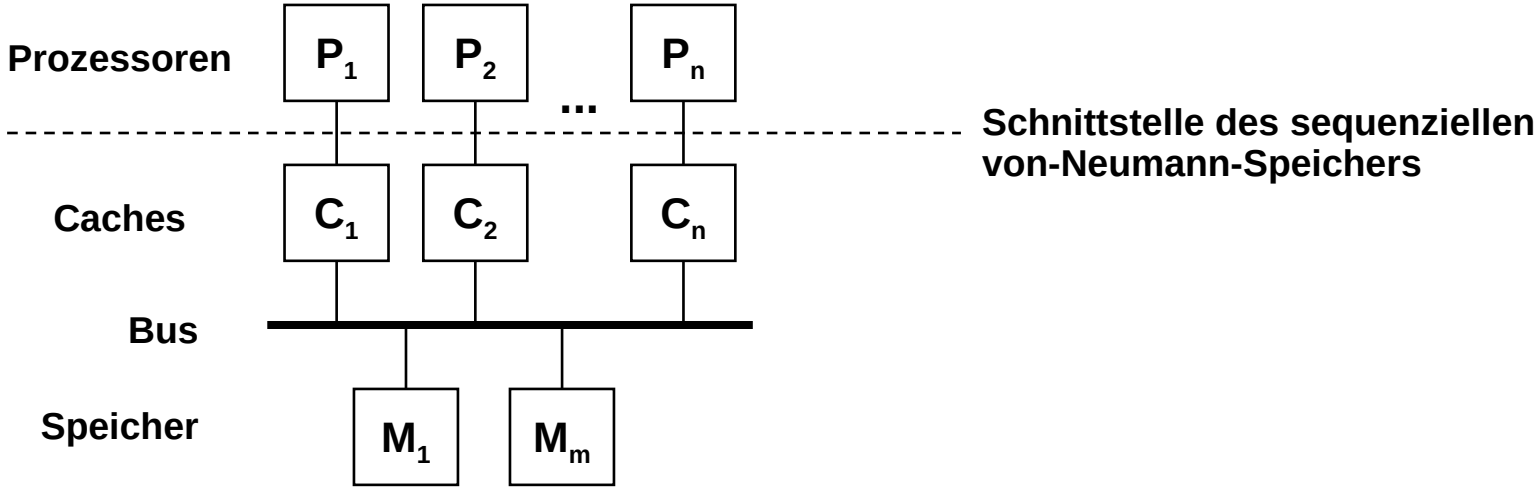
- **Multiprozessorsysteme**
  - **Traditionelle Architektur mit enger Kopplung der Prozessoren.**
  - **Unterstützung für kleine bis mittlere Anzahl von Prozessoren (2 bis ca. 32).**
  - **Verwendung von Caches, um den von-Neumann-Flaschenhals zu verringern.**
  - **Um die obere Anzahl von Prozessoren zu erreichen, ist ein intelligentes Cache-Kohärenz-Protokoll notwendig.**

Alternativen für enge Kopplung

- Basis für Kopplung zwischen Prozessor- und Speicher-Modulen:
- Bussysteme
  - Traditionelle Form.
  - Ein Bus besteht aus Adress- und Datenleitungen (z.B. je 32) sowie aus Kontrollleitungen (ca. 10-30) zur Koordinierung der Busbenutzung (vgl. 5.3).
  - Architektur kann einzelnen Bus oder mehrere parallele oder hierarchische Busse vorsehen.
  - Architektur mit einem Bus ist heute vorherrschender Typ bei Systemen bis ca. 32-64 Prozessoren.
- Switch-basierte Verbindungsnetzwerke
  - Bieten Unterstützung für eine große Zahl von Prozessoren.
  - Kreuzschienenverteiler (Crossbar Switch)
  - Butterfly-Netzwerk, 3D-Torus, Baumstruktur, ...
- Details werden in der LV „Betriebssysteme" vorgestellt.

Beispiel 1

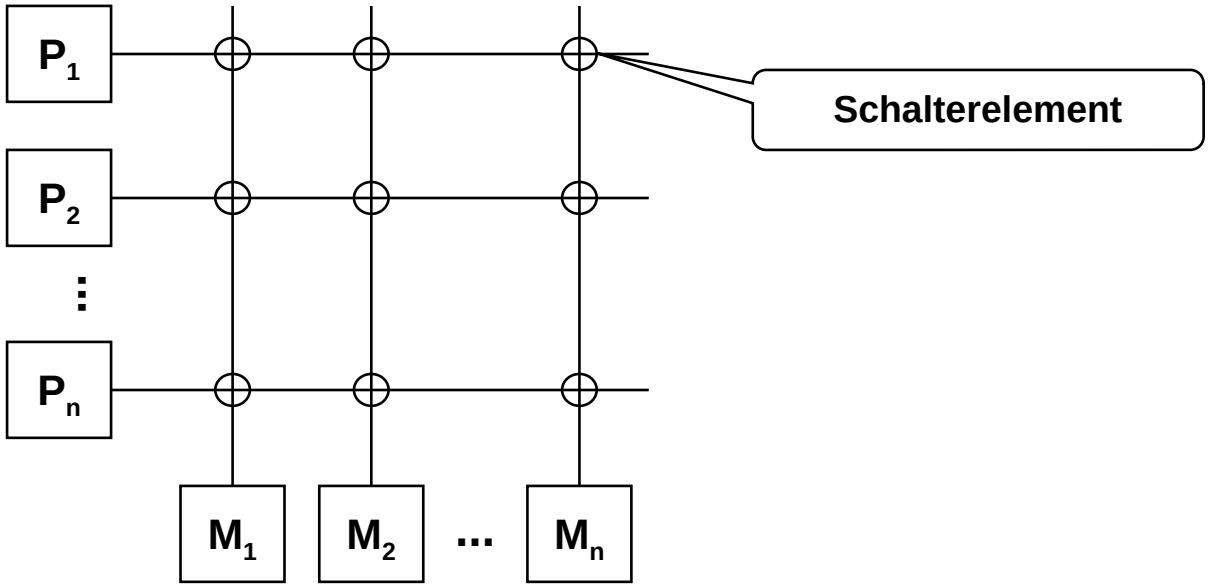
- **1-Bus-basiertes Multiprozessor-System:**



- **Symmetrische Multiprozessor-Systeme (SMP) mit *einem* Bus sind heute der vorherrschende Typ von kommerziellen Rechensystemen.**
- **Durch Einführen von Cache-Speichern und intelligente Cache-Kohärenz-Verfahren kann die Anzahl der sinnvoll betreibbaren Prozessoren an einem Bus auf ca. 32-64 erhöht werden.**
- **Beispiele: PC mit 2 Prozessoren, UNIX-Server-Rechner vieler Hersteller mit i.d.R. bis zu 32 Prozessoren.**

Beispiel 2

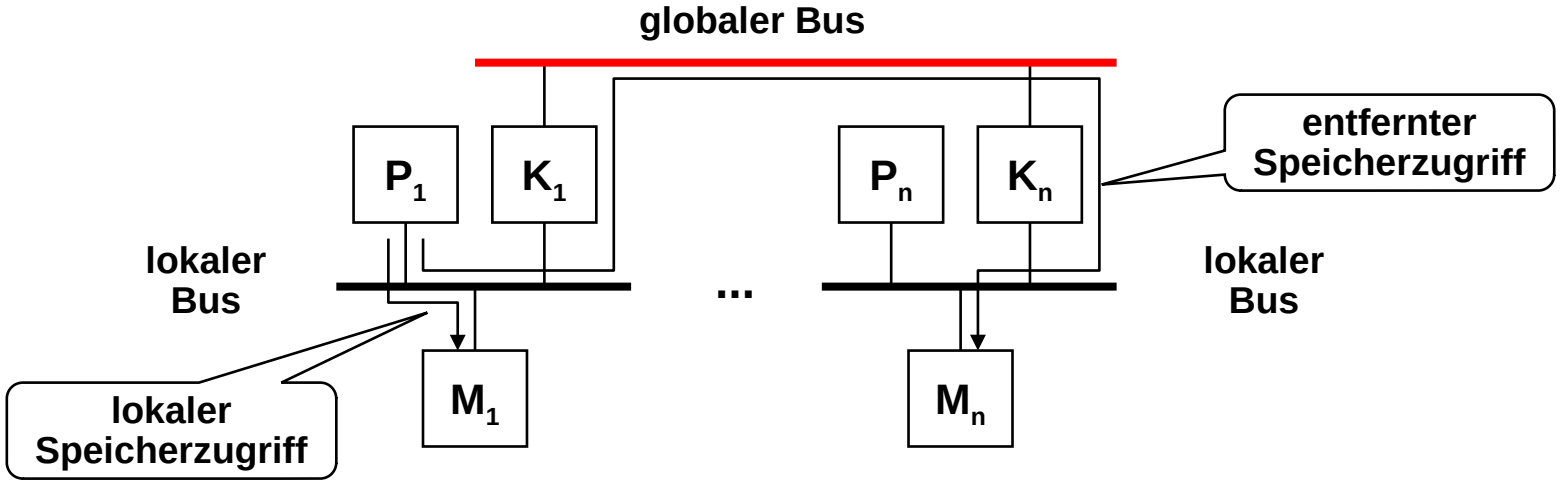
- **Kreuzschienenverteiler-basiertes Multiprozessor-System:**



- **Über den Kreuzschienenverteiler (Crossbar Switch) können mehrere Prozessoren gleichzeitig auf verschiedene Speichermodule zugreifen.**
- **Kreuzschienenverteiler sind teuer und skalieren schlecht (n^2 Schalter). Sie werden daher i.d.R. nur für kleine n (bis ca. 8) eingesetzt.**

Beispiel 3

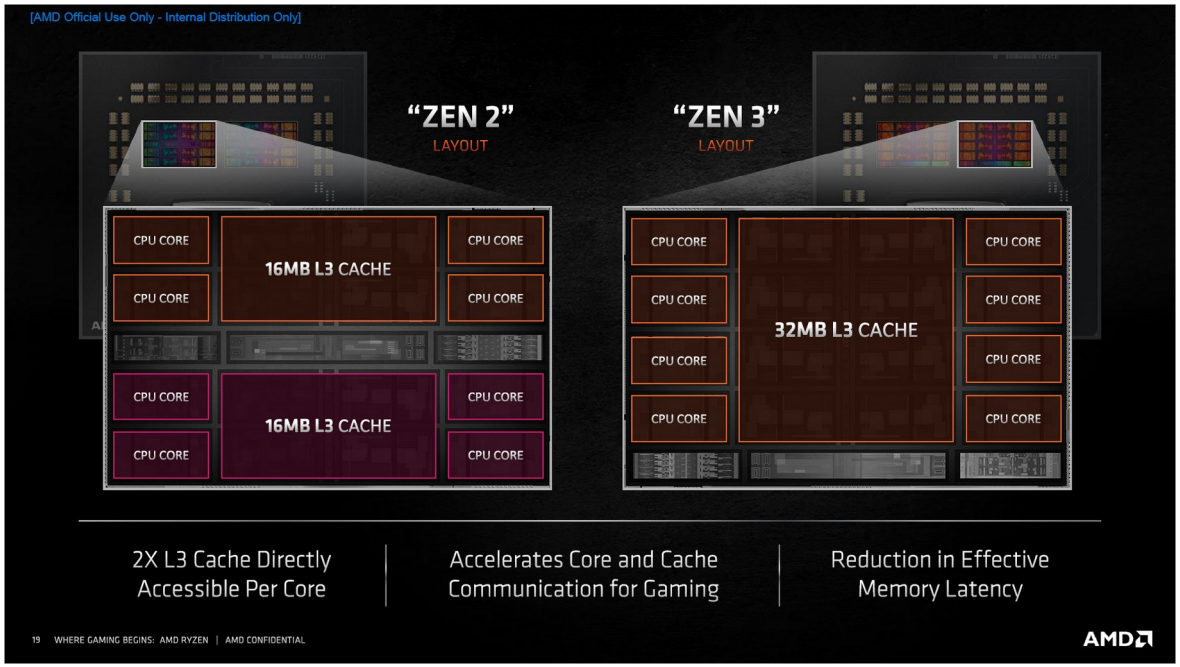
- **Multiprozessor-System mit hierarchischem Bussystem:**



- **Weiterleitung entfernter Speicherzugriffe durch Kommunikationsprozessoren K_i.**
- **Entfernte Zugriffe dauern länger als lokale (Non-Uniform Memory Access).**
- **Mehr als 2 Stufen in der Bushierarchie sind möglich.**
- **Beispiel: Cm* (CMU, 1977): 50 Prozessoren, 3-stufiges Bussystem.**
- **Bus-basierte NUMA-Systeme finden heute im Hochleistungsbereich Einsatz, NUMA-Systeme auf Basis von Verbindungsnetzwerken sind ebenfalls üblich (Beispiel: HP Exemplar X-Class)**

Aktuelles Beispiel: AMD „Infinity Fabric“, Zen3

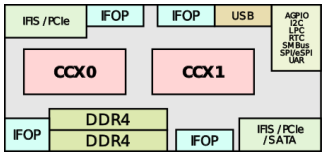
Core Complex (CCX): 4 bzw. 8 CPU-Kerne mit gemeinsamem L3-Cache

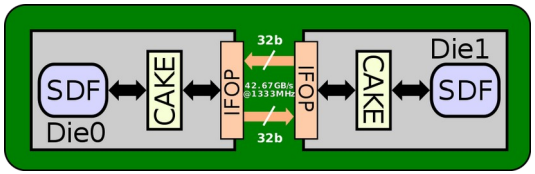


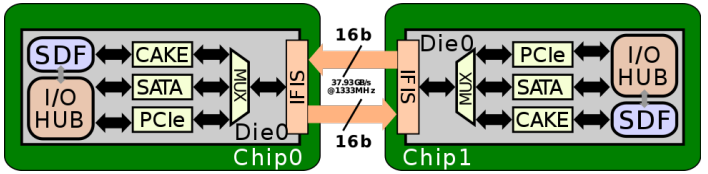
Q: AMD (links), https://en.wikichip.org/wiki/amd/infinity_fabric (rechts)

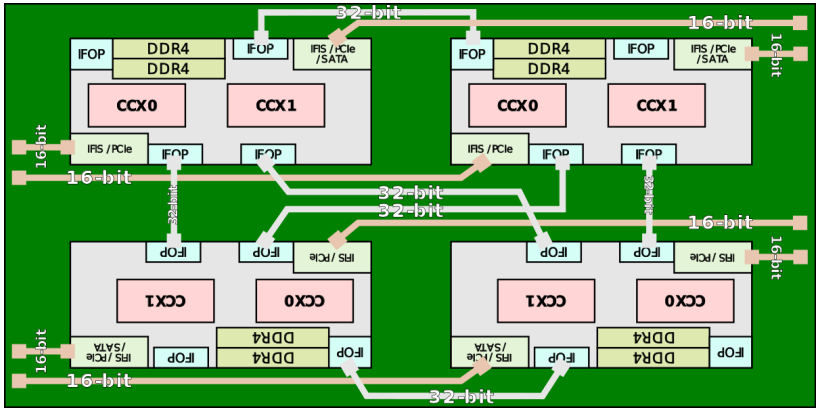
Konsequenzen: Unterschiedliche Latenzen zwischen CPU Cores: 25 ms innerhalb eines CCX,
ca. 70 ns zwischen CCX (Q: c‘t 2/21, S. 133)

CCX-Kopplung mit IF (Mesh), Chip-intern und extern



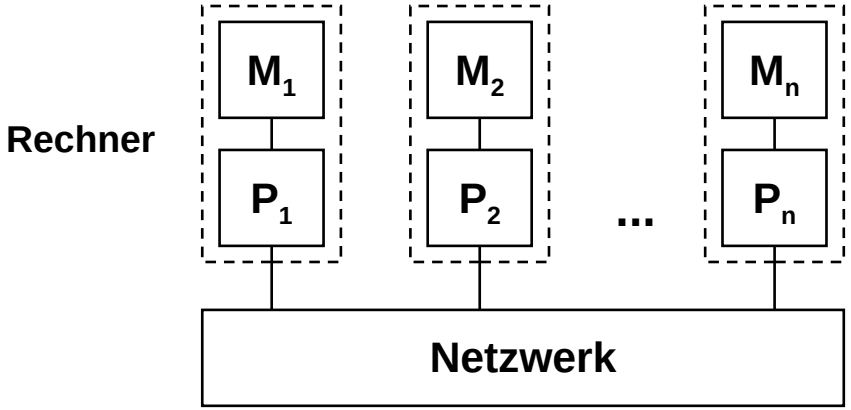






6.4.2 Multicomputer-Systeme

- **Einfache Konstruktion:**
  - **Rechner = Prozessor mit privatem lokalen Speicher (Vermehrt werden als Rechnerknoten SMP-Multiprozessor-Systeme eingesetzt).**
  - **Keine entfernten Speicherzugriffe (NO Remote Memory Access).**
  - **Kommunikation durch lose Nachrichten-Kopplung.**
  - **Prinzipielle Architektur:**



Multicomputer-Systeme (2)

- **Eigenschaften:**
  - **Unterstützung für große Anzahl von Rechnern (> 1024).**
  - **Netzwerk kann Verbindungsnetzwerk, LAN oder WAN sein.**
  - **aktuelle Architektur der meisten massiv-parallelen Systeme (MPP: Massively Parallel Processing).**
  - **Abschwächung der Cache-Kohärenz-Problematik durch explizite nachrichtenorientierte Programmierung (*Message Passing*).**
- **Typisches Anwendungsgebiet:**
  Parallele numerische Anwendungen z.B. für
  - **Wettervorhersage, Klimasimulationen**
  - **Flotten/Crew-Optimierung von Luftverkehrsgesellschaften**
  - **Simulation der Explosion von Kernwaffen, usw.**
- **erreichte Rechenleistung: > 100 PetaFLOPS (10^{17} Gleitpunkt-Op./s)**

Multicomputer-Systeme (3)

- **Beispiele (2005):**
  - **IBM Blue Gene/L DD2 LLNL, DOE, Rochester, USA.**
    - **131072 (2^{17}) PowerPC 440 Prozessoren @ 0,7 GHz, je 2,8 GFLOPS**
    - **R_{max} = 280600 GFLOPS (Vervierfachung in einem Jahr!!)**

OS: CNK/Linux

  - **IBM BGW eServer IBM, Thomas J. Watson RC, USA.**
    - **40960 PowerPC 440 Prozessoren @ 0,7 GHz , je 2,8 GFLOPS**
    - **R_{max} = 91290 GFLOPS**

OS: CNK/Linux

  - **IBM ASC Purple pSeries p5 575 1.9 GHz LLNL, DOE, USA.**
    - **10240 POWER5 Prozessoren @ 1,9 GHz , je 7,6 GFLOPS**
    - **R_{max} = 63390 GFLOPS**

OS: AIX

(Quelle: www.top500.org, 26. Ranking, Nov. 2005)

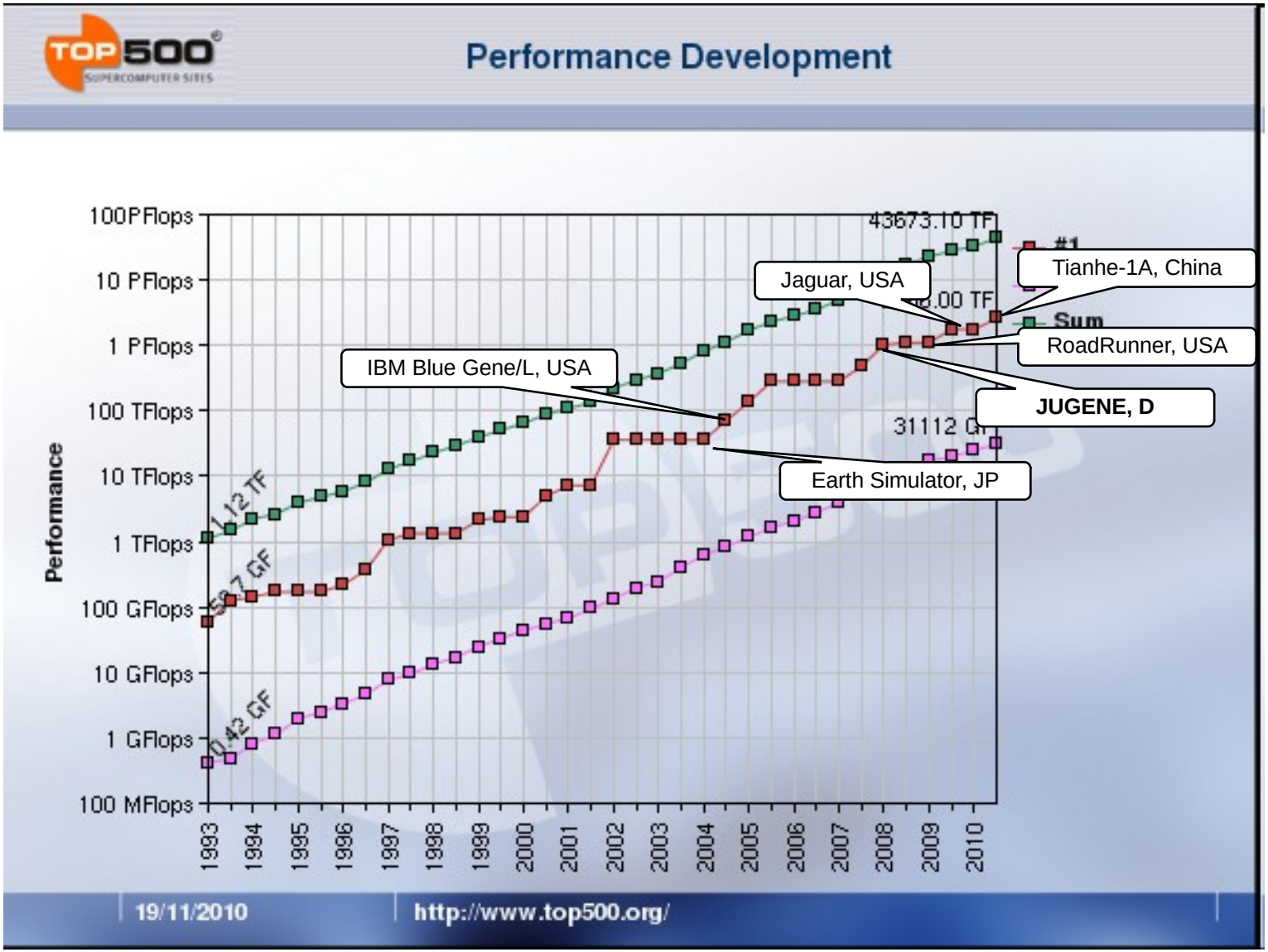
Multicomputer-Systeme (3)

OS in allen Fällen: Linux

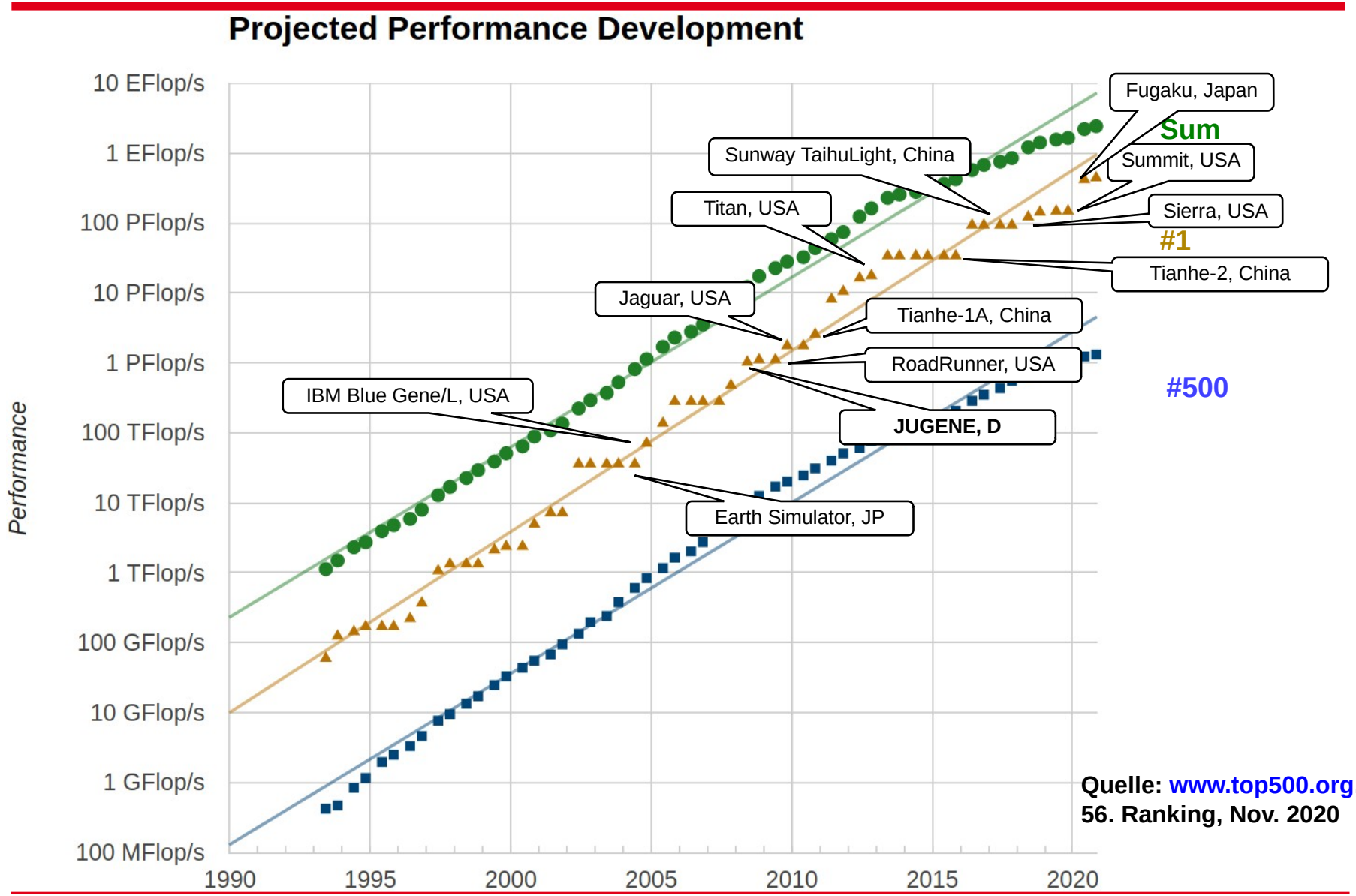
- Akutelle Beispiele (Top 5, 11-2020):
  - Fugaku RIKEN CCS, Japan.
    - 7.630.848 Cores aus A64FX 48C Prozessoren @ 2,2 GHz
    - R_{max} = 442.010 TFLOPS
  - Summit DOE, SC, Oak Ridge Natl. Lab., USA.
    - 2.414.592 Cores aus IBM Power9 22C Prozessoren @ 3,07 GHz + NVIDIA Volta GV100
    - R_{max} = 148.600 TFLOPS
  - Sierra DOE, NNSA, LLNL, USA.
    - 1.572.480 Cores aus IBM Power9 22C Prozessoren @ 3,1 GHz + NVIDIA Volta GV100
    - R_{max} = 94.640 TFLOPS
  - Sunway TaihuLight NSCC, Wuxi, China.
    - 10.649.600 Cores aus Sunway SW26010 260C Prozessoren @ 1,45 GHz
    - R_{max} = 93.014,6 TFLOPS
  - Selene NVIDIA Corp., USA.
    - 555.520 Cores aus AMD EPYC 7742 64C Prozessoren @ 2,25 GHz + NVIDIA DGX100
    - R_{max} = 63.460 TFLOPS
  - (Quelle: www.top500.org, 56. Ranking, Nov. 2020)

Multicomputer-Systeme (4)

Multicomputer-Systeme (4)



Multicomputer-Systeme (4)

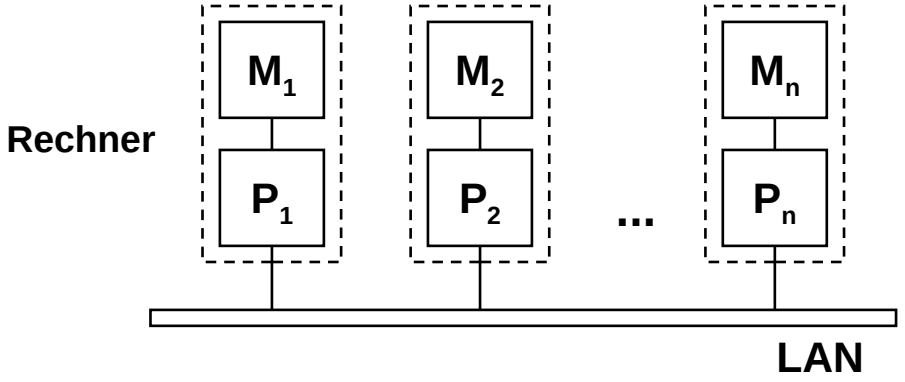


Multicomputer-Systeme (6)

- **Historisches Beispiel:**
  - **SUPRENUM** **GMD, Sankt Augustin, D (u. a. Partner)**
    - **1986-1989 entwickelt, Projekt endete 1991**
    - **1990 einer der weltweit schnellsten Rechner!**
    - **256 Rechenknoten, 32 Kommunikationsknoten, 16 Diagnoseknoten**
      - **16 Cluster, je 16 Rechen-, 2 Komm.-, 1 Diag.-Knoten pro Cluster**
    - **Motorola 68020 Prozessoren @ ca. 14 MHz, 8 MB pro Knoten**
    - **R_{max} = 5 GFLOPS bei insg. 2 GB Hauptspeicher**
    - **Kernideen:**
      - **Optimierter Datenaustausch**
      - **angepasste numerische Verfahren („Multigrid“)**
    - **Anschlussfinanzierung fehlte $\Rightarrow$ kein kommerzieller Erfolg.**

Beispiel 1

- **LAN-basiertes Rechner-Netz:**

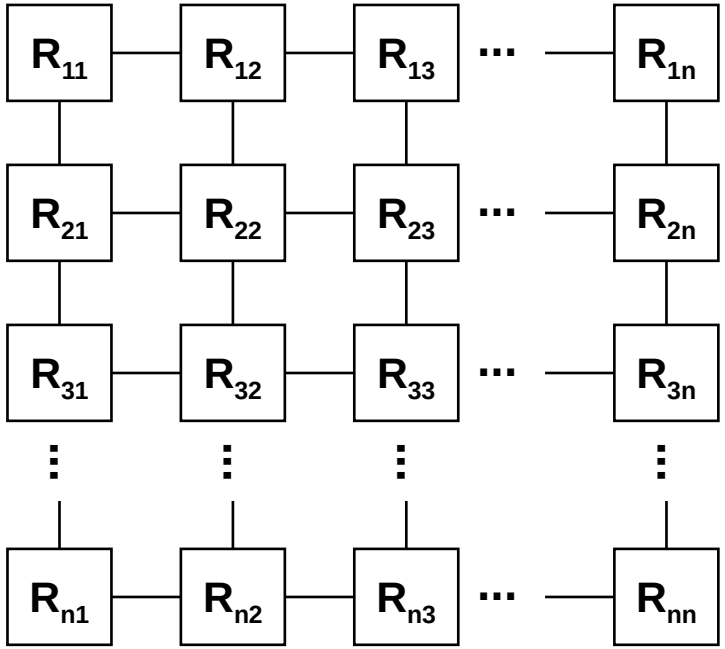


Beispiel:
„Beowulf“-Cluster
(seit ca. 1995)

- **Rechner sind übliche kommerzielle Workstations ohne Bildschirm.**
- **Netzwerk ist ein Local-Area-Netzwerk mit 10-1000 Mbit/sec Übertragungsrate.**
- **Beispiel: Rechner-Farm für 3-D Rendering (vgl. Entstehung "Toy Story").**
- **WAN/Internet-Beispiel: Projekt „SETI-at-home“**

Beispiel 2

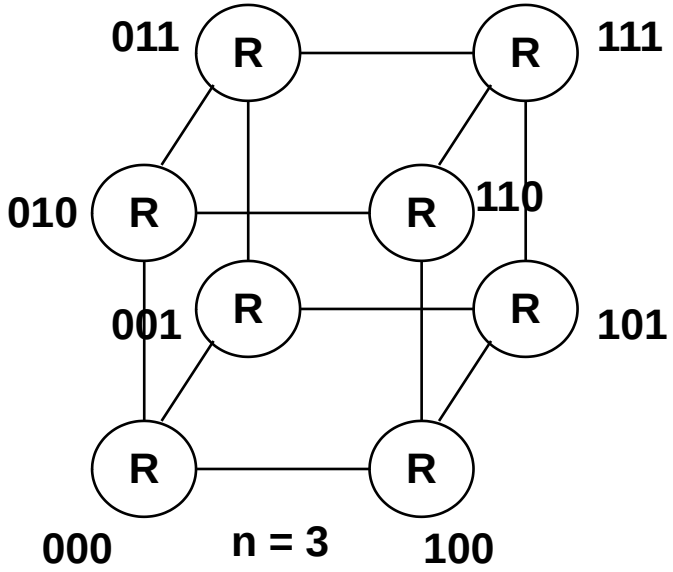
- **Netzwerk mit Gitter-Topologie:**

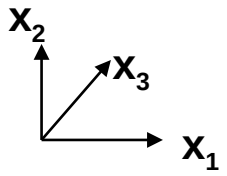


- **Rechner z.B. für Hochleistungs-Gleitpunktarithmetik.**
- **Parallele numerische Anwendungen.**

Beispiel 3

- **Netzwerk mit Hypercube-Struktur:**





Adresse: $x_1x_2x_3$

- **Hypercube = n-dimensionaler Würfel mit 2^n Knotenrechnern.**
- **Einfaches Routing (Pfadfindung für Nachrichten an entfernte Rechner): Adressen von Nachbarn unterscheiden sich an einer Bit-Stelle.**
- **Längster Weg einer Nachricht wächst nur logarithmisch mit der Anzahl der Knoten.**
- **Beispiele: Intel Hypercube, NCube: 512/1024 Prozessoren.**