



Von-Neumann-Rechner

Prof. Matthias Werner
Professur Betriebssysteme

Von der Schaltung zum Processing

- ▶ Wir können Daten darstellen, speichern und logisch bzw. arithmetisch manipulieren
- ▶ Aber:
 - ▶ Algorithmen sind teilweise nur unter großen Aufwand „verdrahtbar“
 - ▶ Flexibilität ist gewünscht
- ▶ Was fehlt dafür noch?
 - ▶ Lesen/Decodieren von Instruktionen
 - ▶ Kontrolle des Programmablaufs
 - ▶ Sprünge

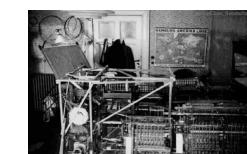
Die ersten ‚Rechner‘

- ▶ 1801: Webstühle mit hölzernen Lochkarten
- ▶ 1822: Dampfbetriebene **Analytical Engine** von Charles Babbage
 - ▶ Mechanischer Rechner, Eingabedaten auf Lochkarten
 - ▶ Unterstützung von Berechnungen und bedingten Anweisungen
 - ▶ Entwicklung der Programmierkonzepte durch Ada Byron
 - ▶ Niemals gebaut
- ▶ 1890: **Hollerith Desk** zur Unterstützung der amerikanischen Volkszählungsbehörde
 - ▶ Lochkartenleser, Zähleinheiten
 - ▶ Gebaut von der **Tabulating Machine Company**, später **International Business Machines (IBM)**
 - ▶ Nutzung von Lochkarten für Ausgabe



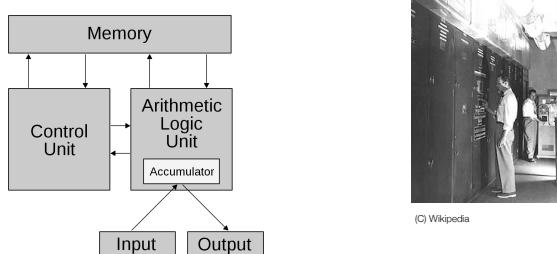
Die ersten ‚Rechner‘

- ▶ 1944: **Harvard Mark I** entwickelt als Kooperation der Harvard-Universität und von IBM
 - ▶ Militärische Anwendung für ballistische Tabellen
 - ▶ Erster programmierbarer digitaler Rechner aus den USA
 - ▶ Schalter, Relays, mechanische Kupplungen
 - ▶ Erster ‚Bug‘ von Grace Hopper gefunden
- ▶ 1941: Konrad Zuse stellt die Z3 fertig
 - ▶ Erster programmierbarer elektromechanischer Rechner
 - ▶ Programme und Daten auf gelochten Filmrollen
 - Papiermangel durch 2. Weltkrieg
 - ▶ Abbildung der booleschen Algebra auf Relays
 - ▶ Unabhängig von den Entwicklungen in den USA



Von Neumann Architektur

- ▶ 1946: **ENIAC** als erster vollständig elektronischer Rechner aus den USA
 - ▶ Änderung der Verkabelung für neue Programme
- ▶ **EDVAC**: Konzept des *stored program computer* von John von Neumann
 - ▶ Speicher enthält Programm und Daten

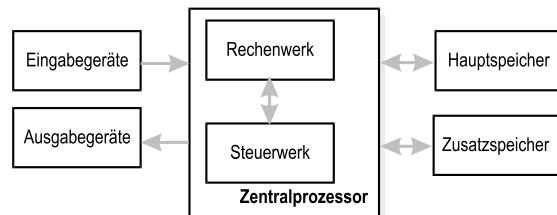


Informationsverarbeitung



- ▶ Abstrakt gesehen besteht Informationsverarbeitung aus den drei Stufen:
 - ▶ Datengewinnung und -eingabe
 - ▶ Verarbeitung und Speicherung
 - ▶ Datenausgabe
- ▶ Diese Stufen sollten sich in Geräten zur Informationsverarbeitung widerspiegeln

Von-Neumann-Rechner

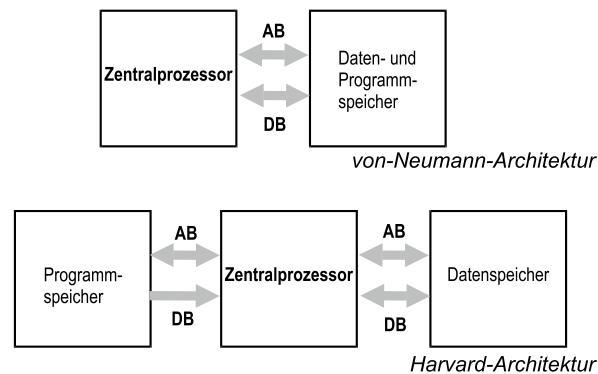


- ▶ **John v. Neumann** (eigentlich Margittai Neumann János Lajos), ungarischer Mathematiker; veröffentlichte erstmals die nach ihm benannten Prinzipien des Rechnerentwurfs, nachdem er für das Manhattan-Projekt beim Entwurf des EDVAC (*Electronic Discrete Variable Automatic Computer*) mitgearbeitet hatte

Von-Neumann-Prinzipien

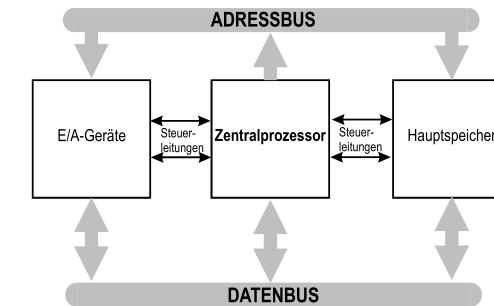
- ▶ Ein Rechner besteht aus **Rechenwerk**, **Steuerwerk**, Speicher und Ein-/ und Ausgabegeräten
- ▶ Die Zentraleinheit arbeitet **taktgesteuert**
- ▶ Die Signale werden **binär** kodiert
- ▶ Der Inhalt eines Speicherwortes wird über (s)eine **Adresse** angesprochen
- ▶ Der Rechner verarbeitet **externe Programme**, die intern gespeichert werden
- ▶ Programmbefehle und Daten werden im **einheitlichen Hauptspeicher** gespeichert
- ▶ Programme und Daten werden **sequentiell** abgearbeitet; der sequentielle Programmfluss kann durch (bedingte oder unbedingte) Sprünge verändert werden
- ▶ Jede theoretisch mögliche Berechnung ist (im Rahmen der Kapazität des Rechners) **berechenbar**

Harvard-Architektur vs. Von-Neumann-Architektur



Bus

- Ein **Bus** (*bidirectional universal switch*) ist eine Verbindungseinheit zwischen verschiedenen logisch getrennten Funktionseinheiten

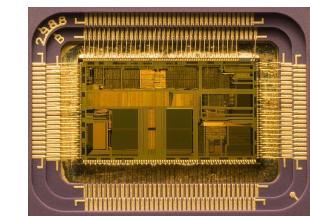


Zentrale Verarbeitungseinheit

- Die **Zentrale Verarbeitungseinheit** ist das Kernstück jedes Von-Neumann-Rechners
- Bezeichnung: **CPU** = central processing unit (ZVE, Zentrale Verarbeitungseinheit)
- Sie liegt häufig als ein IC (*integrated circuit*, Integrierte Schaltung) vor
 - Mikroprozessor (*microprocessor*, μ P)
- Bekannte Prozessorfamilien sind:
 - x86, IA-32, Intel-64
 - MIPS
 - m68k / m88k
 - SPARC
 - PowerPC
 - ARM

Zentrale Verarbeitungseinheit (Forts.)

- In der CPU findet die eigentliche Informationsverarbeitung statt.
- Die CPU besteht mindestens aus **Rechenwerk** (ALU) und **Steuerwerk**
 - **Rechenwerk:** macht die Arbeit
 - **Steuerwerk:** bestimmt, welche Arbeit gemacht werden soll
- Neben der CPU kann es noch andere Prozessoren geben, die spezialisierte Aufgaben erfüllen (**Koprozessoren**), z. B. für:
 - Gleitkommaarithmetik
 - Speichermanagement
 - Bus-Controller

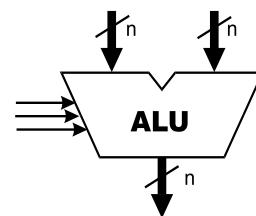


Intel 80486DX2

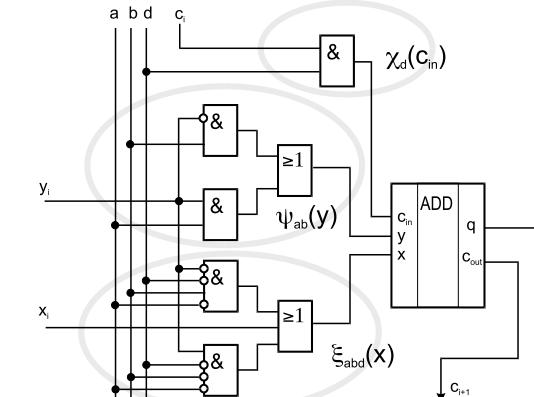
Bildquelle: Wikipedia

ALU / Rechenwerk

- Die ALU steht für **arithmetic/ logic unit** (Arithmetisch/logische Einheit, ALE)
- Begriffe Rechenwerk und ALU sind nicht klar unterscheidbar
- Zu den Aufgaben des Rechenwerks gehören:
 - Ausführung **arithmetischer Operationen** wie Addition, Subtraktion, Inkrementierung, Dekrementierung und Bildung von Zahlenkomplementen
 - Ausführung **logischer Operationen** wie bitweise Negation, Disjunktion, Konjunktion, Antivalenz, etc.
 - Ausführen von **Vergleichoperationen** zwischen Zahlen
 - Ausführung von **Schiebeoperationen**, also zyklisches und nichtzyklisches Rechts- oder Linksschieben
 - mitunter deshalb auch: ALSU (arithmetic/logic/shifting unit)

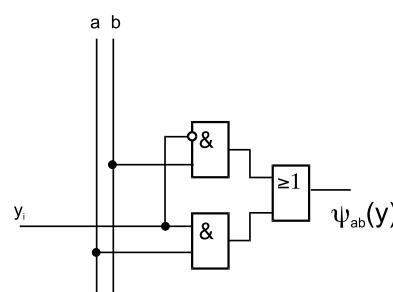


Kernzelle einer einfachen Beispiel-ALU



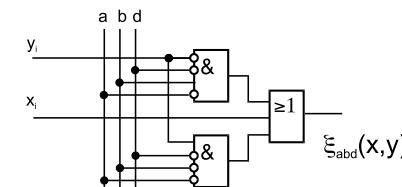
ALU - Parameterschaltung

Durch verschiedene Eingangssignale a und b kann die Parameterschaltung alle einstelligen Binärfunktionen für y darstellen



a	b	$\psi_{ab}(y)$
0	0	0
0	1	\bar{y}
1	0	y
1	1	1

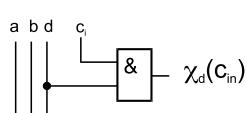
ALU - Parameterschaltung (Forts.)



a	b	d	$\xi_{abd}(x,y)$
0	0	0	$x \vee y$
0	1	0	$x \vee \bar{y}$
1	0	0	x
1	1	0	\bar{x}
-	-	1	x

- Warum unterscheiden sich die Parameterschaltungen für beide Eingangssignale?
- Ziel ist eine große Flexibilität
- Durch Hinzunahme der OR-Funktion wird die ALU **logisch vollständig**

ALU - Parameterschaltung (Forts.)



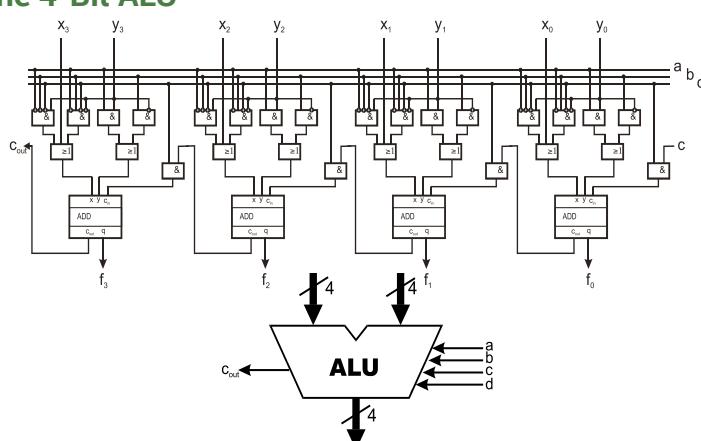
d	$\chi_d(c_{in})$
0	0
1	c_{in}

- Ist die Steuerleitung $d = 0$, so wird **kein** Übertrag zwischen den einzelnen ALU-Kernzellen übertragen
- aus arithmetischen Operationen werden logische, bei denen das Ergebnis einer Bitstelle nicht von anderen Bitstellen abhängt

Funktionen der ALU

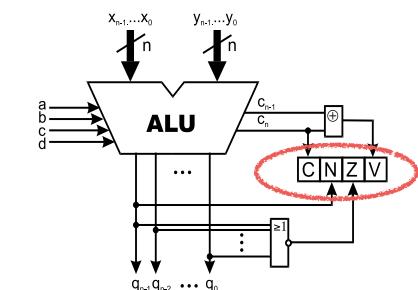
a	b	c	d	$\psi_{ab}(y)$	$\xi_{abd}(x, y)$	Ergebnisfunktion $f_{abcd}(x, y)$	Bezeichnung
<i>Logische Funktionen</i> $f_{abcd}(x, y) = \psi_{ab}(y) \oplus \xi_{abcd}(x, y)$							
0	0	-	0	0	$x \vee y$	$x \vee y$	Disjunktion
0	1	-	0	\bar{y}	$x \vee \bar{y}$	$x \wedge y$	Konjunktion
1	0	-	0	y	x	$x \oplus y$	Antivalenz
1	1	-	0	1	x	\bar{x}	Einerkomplement
<i>Arithmetische Funktionen</i> $f_{abcd}(x, y) = \psi_{ab}(y) + \xi_{abcd}(x, y) + c$							
0	0	0	1	0	x	x	Identität (Transfer)
0	0	1	1	0	x	$x + 1$	Inkrement
0	1	0	1	\bar{y}	x	$x - y$	Subtraktion (1er Kpl.)
0	1	1	1	\bar{y}	x	$x - y + 1$	Subtraktion (2er Kpl.)
1	0	0	1	y	x	$x + y$	Addition
1	0	1	1	y	x	$x + y + 1$	Addition mit Übtr.
1	1	0	1	1	x	$x - 1$	Dekrement
1	1	1	1	1	x	x	Identität (Transfer)

Einfache 4-Bit ALU



Status

- Zur Realisierung von **Vergleichsoperationen** dient ein Statusregister (flag register)
- Flags werden entsprechend dem Ergebnis der letzten Operation gesetzt
- Typische Flags sind:
 - Carry ($C = c_n$):** eine arithmetische Operation hat einen Überlauf erzeugt
 - Zero ($Z = q_0 \vee \dots \vee q_{n-1}$):** ein Ergebnis besteht nur aus Nullen
 - Negative ($N = q_n$):** ein Ergebnis stellt eine negative Zahl dar
 - Overflow ($V = c_{n+1} \oplus c_n$):** bei einer 2er Komplement-Berechnung trat ein Überlauf auf



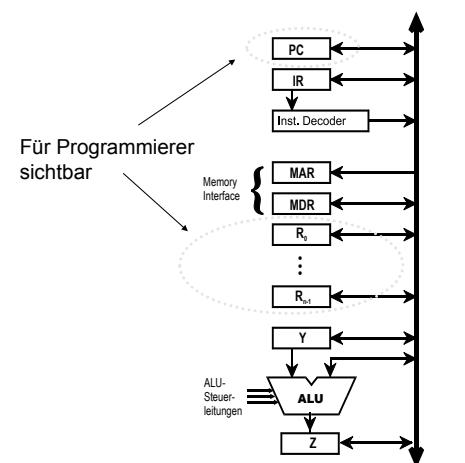
Auswertung von Vergleichen

- ▶ Vergleich (compare) erfolgt durch Ausführung einer Subtraktion ohne Speichern des Rechenergebnisses
- ▶ Das Ergebnis des Vergleichs ist aus den Flags ablesbar

Relation	vorzeichenlose Zahlen	Zahlen im 2er-Komplement
$x = y$	$Z=1$	$Z=1$
$x \neq y$	$Z=0$	$Z=0$
$x \geq y$	$C=1$	$N=V$
$x < y$	$C=0$	$N \neq V$
$x > y$	$C=1$ und $Z=0$	$N=V$ und $Z=0$
$x \leq y$	$C=0$ oder $Z=1$	$N \neq V$ oder $Z=1$

Sichtbare und unsichtbare Register

- ▶ PC (program counter, Befehlszähler, BZ, häufig auch IP *instruction pointer*)
→ hält die Adresse der Speicherzelle, in der der nächste auszuführende Befehl steht
- ▶ IR (instruction register, Befehlsregister, BR)
→ speichert den gerade ausgeführten Befehl

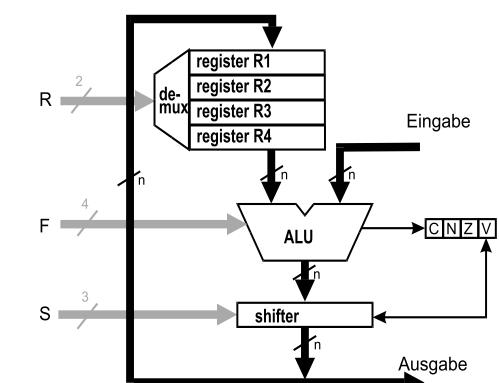


Registersatz

- ▶ In der CPU gibt es in der Regel ein oder mehrere Register, deren Verwendungszweck nicht von vornherein bestimmt ist
→ **Allzweckregister** (*general purpose register*)
- ▶ Diese Register kann ein Programmierer frei verwenden.
- ▶ Bestimmte Register in der CPU sind auf spezielle Aufgaben spezialisiert
→ **Spezialregister** (*special purpose register*)
- ▶ Anzahl und Einsetzbarkeit von Registern stellen ein wesentliches Merkmal einer Architektur dar.
- ▶ Die Gesamtheit der für den Programmierer nutzbaren Register wird Registersatz genannt.

Steuerung des Rechenwerkes

- ▶ Schwarze Busse: **Datensignale**
Informationen die verarbeitet werden
- ▶ Graue Busse: **Steuersignale**
bestimmen, wie die Informationen verarbeitet werden
- ▶ Die Steuersignale können jedoch wieder zu Datenwörtern zusammengefasst werden



Das Steuerwerk

- ▶ Das Steuerwerk (auch *control unit* - Steuereinheit) koordiniert Rechenwerk, Ein-/Ausgabeeinheit, Speichersystem
- ▶ Es fungiert als Nervenzentrum:
 - ▶ sendet Steuersignale an andere Einheiten
 - ▶ ergründet deren Status
- ▶ Wesentliche Funktionen:
 - ▶ **Sequencing:** Generierung von Steuersignalen zur Abarbeitung einer beliebigen, gegebenen Instruktion (innerhalb der CPU) ➔ Kapitel 4
 - ▶ Steuerung/Überwachung von Speicher, E/A-System

Interagieren mit dem Speicher

- ▶ Speicher dient zum Ablegen von Informationen - Daten, Programmen, Ergebnissen
 - ▶ es ist aus HW-Sicht i. A. **nicht** zu erkennen, **was** im Speicher liegt (Datentyp)
 - ▶ Die Interpretation obliegt der auf den Speicher zugreifenden Einheit
- ▶ Um Zugriff zum Speicher zu erhalten, muss ein Adresssignal erzeugt werden. Außerdem muss ein Steuersignal erzeugt werden, das besagt, ob gelesen oder geschrieben wird
- ▶ Es gibt verschiedene Arten von Speicher
 - ▶ Schreiben + Lesen möglich: *random access memory (RAM*, etwa: Speicher mit wahlfreiem Zugriff)
 - ▶ nur Lesen möglich: *read only memory (ROM*, Nur-Lese-Speicher)

Interagieren mit dem Speicher (Forts.)

- ▶ Um mit dem Speicher kommunizieren zu können, gibt es eine Reihe von Spezialregistern in der CPU (vgl. Folie 92):
 - ▶ MAR (*memory adress register*, **Speicheradressierungsregister**, SAR)
 - ▶ speichert die Adresse eines gesuchten Wertes im Speicher
 - ▶ MDR (*memory data register*, **Speicherdatenregister**, SDR)
 - ▶ speichert den Wert, der in den Speicher geschrieben werden soll oder der aus dem Speicher gelesen wurde

Interagieren mit dem Speicher (Forts.)

- ▶ MAR und MDR stehen dem Programmierer nicht direkt zur Verfügung.
- ▶ Warum werden MAR und MDR gebraucht? Es könnte ja auch direkt auf/vom das/dem betreffende(n) Register gelesen werden.
 - ▶ Werte auf dem Bus müssen u.U. länger anliegen
 - ▶ die CPU-interne Verarbeitung wäre solange blockiert
 - ▶ Adressen können sich aus komplexen Ausdrücken zusammensetzen
 - ▶ MAR dient als Zwischenspeicher
 - ▶ Busse speichern keinen Wert
 - ▶ Register dienen als Busrepräsentation (Portal)
- ▶ Aus ähnlichen Gründen gibt es innerhalb der CPU bestimmte Register, die als Portal zu nichtspeichernden Einheiten dienen
- ▶ Bei der ALU **Operandenregister** (meist mit X und Y bezeichnet) und **Ergebnisregister** (meist mit Z bezeichnet)

Speicherlesen- und Schreiben

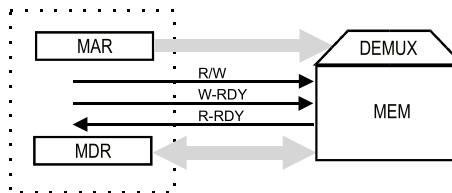
Lesen

- ▶ Lade Adresse nach MAR
- ▶ Setze R/W auf „Lesen“
- ▶ Warte bis R-RDY = „okay“
(MFC = memory function complete)

- ▶ Übernehme Daten in MDR

Schreiben

- ▶ Lade Adresse nach MAR
- ▶ Lade Datenwert in MDR
- ▶ Setze R/W auf „Schreiben“
- ▶ Setze W-RDY auf „okay“
- ▶ Warte bis R-RDY = „okay“
(MFC = memory function complete)



Ein- und Ausgabe

- ▶ Die Ein- und Ausgabe an Geräte funktioniert analog zum Speicherlesen und -schreiben
- ▶ Durch eine Adresse wird ein Gerät ausgewählt. Aus sich der CPU heißt eine solche Adresse **Port**
- ▶ Ein auf ein Port geschriebener oder von dort gelesener Wert wird in bestimmten Registern zwischengespeichert.
- ▶ In manchen Architekturen wird nicht zwischen Speicher und anderen Geräten unterschieden
 - Geräte haben **speziellen Adressbereich**
- ▶ Andere Architekturen besitzen **spezielle Steuerleitungen** für Ein- und Ausgabe