

Historischer Überblick

Parameter	Kursinformationen
Veranstaltung:	Digitale Systeme / Eingebettete Systeme
Semester:	Wintersemester 2025/26
Hochschule:	Technische Universität Freiberg
Inhalte:	<u>Übersicht der historischen Entwicklung von Rechentechnik</u>
Link auf GitHub:	https://github.com/TUBAF-IfI-LiaScript/VL_EingebetteteSysteme/blob/master/01_HistorischerUeb erblick.md
Autoren:	Sebastian Zug, André Dietrich, Fabian Bär & GitHub Copilot Teaching-Agent



Fragen an die Veranstaltung

- Worin lag der „große Wurf“ des Intel 4004?
- Was bedeutet die Angabe 8bit, 16bit usw. ?
- Erklären Sie die Schichten der Rechnerstruktur.
- Worin unterschieden sich ENIAC und die Z3?
- Welche technologischen Durchbrüche ermöglichen die Miniaturisierung von Computern?
- Warum war die Harvard-Architektur für frühe Computer wichtig?
- Welche Rolle spielte die Erfindung des Transistors für die Computerentwicklung?
- Wie entwickelten sich Speichertechnologien im Laufe der Computergeschichte?

Weiterführende Literaturhinweise

- Dirk W. Hoffmann, Grundlagen der Technischen Informatik, Hanser-Verlag, 2007
- Raul Rojas, Sechzig Jahre Computergeschichte - Die Architektur der Rechenmaschinen Z1 und Z3 [Link](#)
- Webseiten zur Rechnergeschichte
 - <http://www.horst-zuse.homepage.t-online.de/z1.html>
 - <http://www.computerhistory.org/babbage/adalovelace>

Rückblick, Fragen und Diskussionen

C# ist furchtbar aufgebläht ... wer braucht so etwas wie Properties?

— Kursteilnehmer

Sprachunterschiede

C++ und C# verfolgen unterschiedliche Ziele:

- C++ gibt maximale Kontrolle, liegt nah an der Hardware, erfordert oft viel Boilerplate für Kapselung. Oberstes Ziel ist Performance und Transparenz.
- C# ist eine *Managed Language*, die viele alltägliche Aufgaben übernimmt (z.B. Speicherverwaltung, Typsicherheit) und Features wie **Properties** bereitstellt, um **Datenkapselung** einfacher umzusetzen.

Properties

```
// C# Property
class Person {
    private int age;

    public int Age {
        get { return age; }
        set {
            if (value < 0) throw new ArgumentException("Alter darf nicht negativ sein");
            age = value;
        }
    }

    var p = new Person();
    p.Age = 25; // Aufruf des setters
    Console.WriteLine(p.Age); // Aufruf des getters
```

In C++ müsste man dafür Getter- und Setter-Methoden explizit definieren, in C# erledigt die Property beides in einer eleganten Syntax.

Memo: Es gibt keine "richtige" oder "falsche" Programmiersprache. Jede hat ihre Stärken und Schwächen. Wichtig ist, die Konzepte zu verstehen, die hinter den Sprachfeatures stehen.

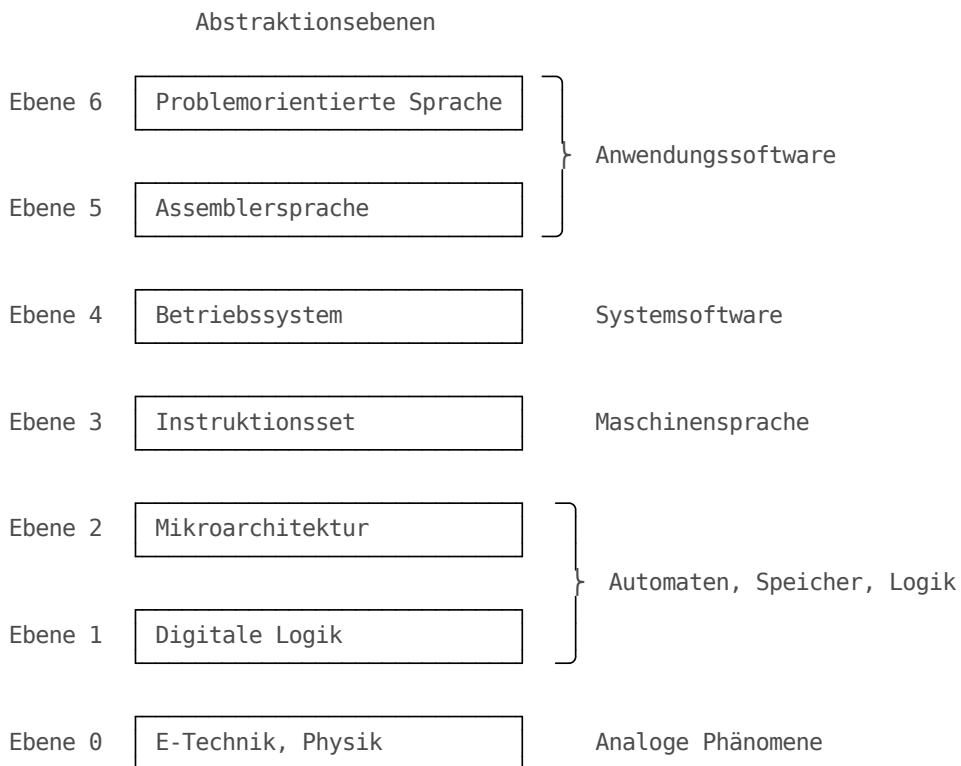
Format	Informatik Studierende	Nicht-Informatik Studierende
Verbesserungsvorschlag	0	0
Fragen	1	0
generelle Hinweise	0	0

Von Rechenmaschinen zum uC: 200 Jahre Rechentechnik

Erinnern Sie sich an den Arduino-Code aus der letzten Vorlesung? `pinMode()`, `digitalWrite()`, `delay()` - diese einfachen Befehle sind das Ergebnis einer 200-jährigen Evolution. Heute reisen wir durch die Zeit und entdecken, wie aus mechanischen Zahnrädern die Mikrocontroller in Ihrem Arduino wurden.

Hier ist das Faszinierende: Die Grundprinzipien, die Charles Babbage 1837 in seiner Analytical Engine erdachte, stecken heute noch in Ihrem ATmega328P. Nur sind sie millionenfach kleiner und milliardenfach schneller geworden.

Mission dieser Vorlesung: Verstehen Sie, warum Ihr Arduino so funktioniert, wie er funktioniert - durch die Brille der Geschichte!



Begrifflichkeiten

Bevor wir in die Geschichte eintauchen, klären wir die Grundbegriffe. Ihr Arduino ist übrigens ein perfektes Beispiel für alle diese Definitionen: Er löst Probleme (LED-Steuerung), verarbeitet Daten (Sensormessungen) und besteht aus zusammenarbeitenden Baueinheiten (CPU, RAM, Flash-Speicher).

Ein Computer oder Digitalrechner ist eine Maschine, die Probleme für den Menschen lösen kann, indem sie die ihr gegebenen Befehle ausführt. (Tannenbaum, Computerarchitektur)

Ein Computer oder Rechner ist ein Gerät, das mittels programmierbarer Rechenvorschriften Daten verarbeitet. (Wikipedia)

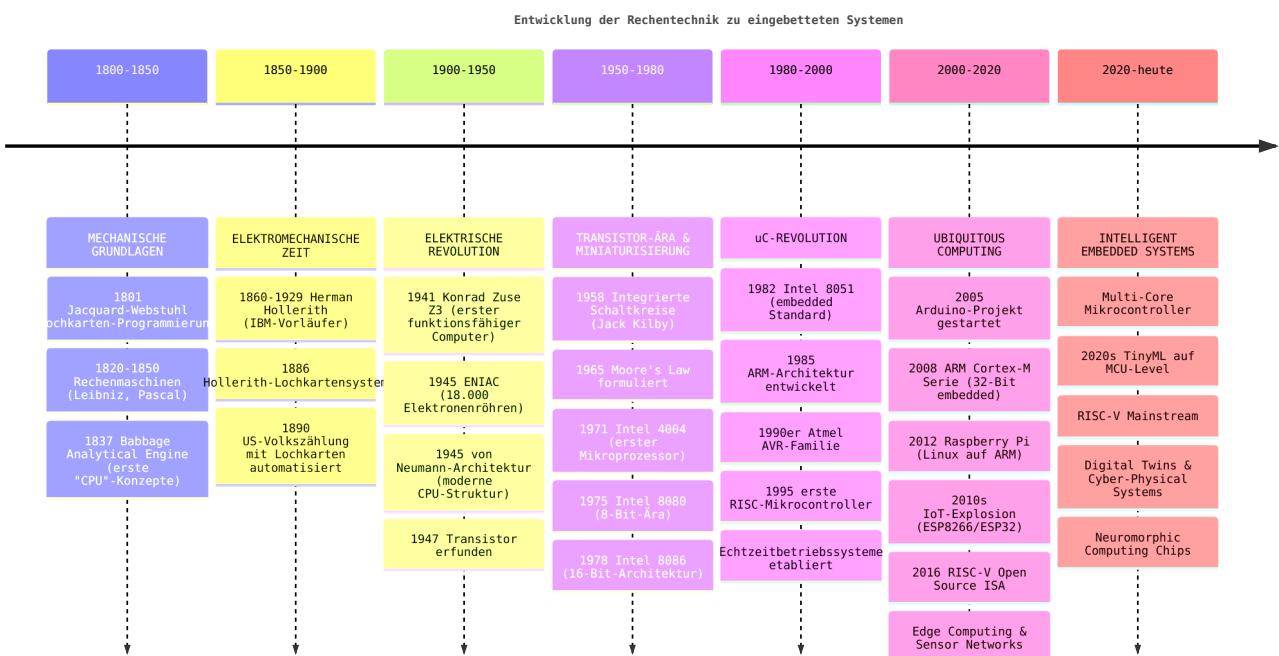
Rechenanlage (Computer) ... Die Gesamtheit der Baueinheiten, aus denen ein Datenverarbeitungssystem aufgebaut ist. (DIN 44300)

Interessant: Diese Definitionen von 1945, 2020 und 1970 beschreiben alle das Gleiche - nur die Implementierung hat sich dramatisch verändert. Von raumfüllenden Röhrencomputern zu daumennagelgroßen Mikrocontrollern.

Welche Gemeinsamkeiten und Unterschiede sehen Sie in diesen Definitionen?

Zeitstrahl: Von Zahnrädern zu Mikrocontrollern

Bevor wir in die Details eintauchen, verschaffen wir uns einen Überblick über die wichtigsten Meilensteine. Dieser Zeitstrahl zeigt Ihnen die Evolution von mechanischen Rechenmaschinen bis zu Ihrem Arduino - eine faszinierende 200-Jahres-Reise!



Faszinierend: Diese Timeline zeigt die kontinuierliche Evolution der technischen Informatik! Von Babbages mechanischen Konzepten 1837 über die Transistor-Revolution bis zu Ihrem Arduino heute - jede Epoche brachte entscheidende Durchbrüche. Besonders bemerkenswert: Moore's Law, ARM-Architektur und RISC-V haben direkt zu den Mikrocontrollern geführt, die Sie heute in embedded Systems verwenden. Ihr Arduino ist das Ergebnis von 200 Jahren Ingenieurskunst!

Rechenmaschinen

Ausgangspunkt für die Vereinfachung des Rechnens ist das Konzept des Stellenwertsystems. In einer positionsunabhängigen Darstellung bedarf es immer neuer Symbole um größere Zahlen auszudrücken. Im römischen Zahlensystem sind dies die bekannten Formate **MDCCLXV**. Können Sie den Zahlenwert rekonstruieren - es ist das Gründungsjahr der Bergakademie.

Antwort: $MDCCLXV = 1000 + 500 + 200 + 50 + 10 + 5 = 1765$ - Gründung der Bergakademie Freiberg!

Interessant: Additive Zahlensysteme wie das römische sind für maschinelle Verarbeitung ungeeignet, da jede Ziffer einzeln dekodiert werden muss.

Die Idee, den Wert einer Ziffer von ihrer Position innerhalb der ganzen Zahl abhängig zu machen, geht auf den indischen Kulturkreis zurück. Die sogenannten "arabischen" Zahlen integrieren dafür einen zentrale Voraussetzung, die "0". Ohne die Null ist es nicht möglich, den Wert einer einzelnen Ziffer zu vervielfachen.

Die Null ist entscheidend für digitale Logik! In Ihrem Arduino repräsentiert `0V = "0"` und `5V = "1"`. Ohne das Konzept der Null gäbe es keine Boolesche Algebra, keine Gatter, keine Mikrocontroller. Die indische Innovation der Null ermöglichte letztendlich die gesamte Digitaltechnik.

Binärsystem-Verbindung: Das Stellenwertsystem ist fundamental für alle Computer! Ihr Arduino arbeitet intern nur mit Binärzahlen (Basis 2) - `pinMode(13, OUTPUT)` wird zu `DDRB |= (1<<5)`, was binär `00100000` bedeutet. Das Prinzip ist identisch zu unserem Dezimalsystem, nur mit zwei statt zehn Ziffern.

Der Abakus greift diesen Ansatz auf und strukturiert den Rechenprozess. Dabei unterscheidet man verschiedene Systeme. Es existieren Vorgehensmuster für die Umsetzung der Grundrechenarten und des Wurzelziehens.

Der Abakus (ca. 2400 v. Chr.) ist eines der ältesten bekannten Rechenhilfsmittel. Er nutzt das Stellenwertsystem, indem Perlen auf Stäben verschoben werden, um Zahlen darzustellen und Rechenoperationen durchzuführen.

CPU-Analogie: Der Abakus ist erstaunlich ähnlich zu Ihrem Arduino! Die Perlen = Register (Speicher für Zwischenergebnisse), die Stäbe = Datenbusse (Transportwege), die Rechenregeln = Algorithmus (Ihr Arduino-Code), der Benutzer = Control Unit (Steuerwerk). Sogar Parallel-Processing gibt es: mehrere Stäbe gleichzeitig bearbeiten!



Abakus im Stadtzentrum von Lübeck Dietmar Rabich, Abakus des Wissenschaftspfads, Lübeck, Schleswig-Holstein, Deutschland, https://commons.wikimedia.org/wiki/File:L%C3%BCbeck,_Wissenschaftspfad,_Abakus -- 2017 -- 0373.jpg

Eine weitreichendere Unterstützung beim eigentlichen Rechenprozess bieten die Napierischen Rechenstäbe (John Napier 1550 - 1617), die insbesondere die Multiplikation einer Ziffer mit einer beliebig großen Zahl unterstützen.

Embedded-System-Prinzip: Napierische Rechenstäbe sind das erste adaptive "Look-Up-Table"-System! Ihr uC nutzt dasselbe Prinzip: In der `sin()`-Funktion stehen vorberechnete Werte in Tabellen im Flash-Speicher. Statt zu rechnen, wird nachgeschlagen - genau wie bei Napier 1617! Modern: CORDIC-Algorithmen in Mikrocontrollern.



[Video auf YouTube ansehen](#)

Fehler 153

Fehler bei der Konfiguration des Videoplayers



Die notwendige manuelle Addition bei größeren Faktoren löste die Rechenmaschine von Wilhelm Schickard. Die Automatisierung der Addition ist mechanisch gelöst und zum Beispiel unter [Link](#) beschrieben.



Nachbau der Rechenmaschine von Wilhelm Schickard [\[Schickard\]](#)



Video auf YouTube ansehen

Fehler 153

Fehler bei der Konfiguration des Videoplayers



Blair Pascal, Gottfried Wilhelm Leibniz und andere Entwickler trieben die Entwicklung weiter, erweiterten die Stellsysteme, integrierten weitere Rechenarten hatten aber insgesamt mit den mechanischen Herausforderungen und fehlender Fertigungsgenauigkeit zu kämpfen.

Es ist unwürdig, die Zeit von hervorragenden Leuten mit knechtischen Rechenarbeiten zu verschwenden, weil bei Einsatz einer Maschine auch der Einfältigste die Ergebnisse sicher hinschreiben kann. (Gottfried Wilhelm Leibniz)

Abstraktere Konzepte

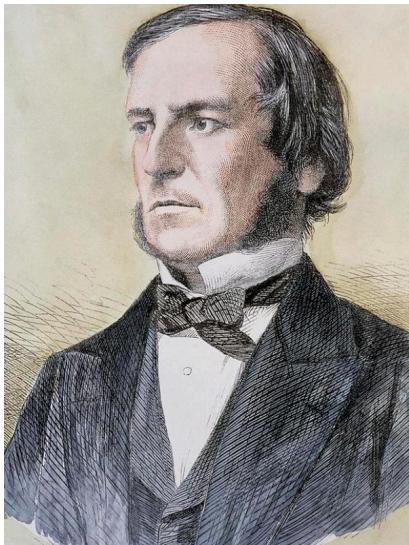
Bisher stand die Berechnung von einzelnen Ergebnissen auf der Basis einer Sequenz von Eingaben im Vordergrund. Ende des 18. Jahrhunderts entwarfen Visionäre neue Konzepte, die allgemeingültige Lösungen integrierten.

Diesen Aspekt kann man auf technischer und theoretischer Ebene betrachten.

Die Grundlagen moderner Rechner legten die Arbeiten von Georg Boole (1815 - 1864), der eine boolesche Algebra (oder einen booleschen Verband) definierte, die die Eigenschaften der logischen Operatoren UND, ODER, NICHT sowie die Eigenschaften der mengentheoretischen Verknüpfungen Durchschnitt, Vereinigung, Komplement verallgemeinert. Gleichwertig zu booleschen Algebren sind boolesche Ringe, die von UND und ENTWEDER-ODER (exklusiv-ODER) beziehungsweise Durchschnitt und symmetrischer Differenz ausgehen.

Arduino-Hardware-Verbindung: Booles Algebra von 1854 läuft EXAKT in Ihrem Arduino! Jeder `if(digitalRead(2) && digitalRead(3))` ist ein AND-Gatter. Die Hardware-Register wie PORTB werden mit OR (`|=`) und AND (`&=`) manipuliert. Booles abstrakte Mathematik wurde 1947 zu Transistor-

Gattern und 2009 zu Ihrem Arduino!



Georg Boole Autor unbekannt, https://commons.wikimedia.org/wiki/File:George_Boole_color.jpg

$$\begin{aligned}f(a, b, c) &= (a \wedge b) \vee (\neg c) \\f(a, b, c) &= (a \cdot b) + (\bar{c})\end{aligned}$$

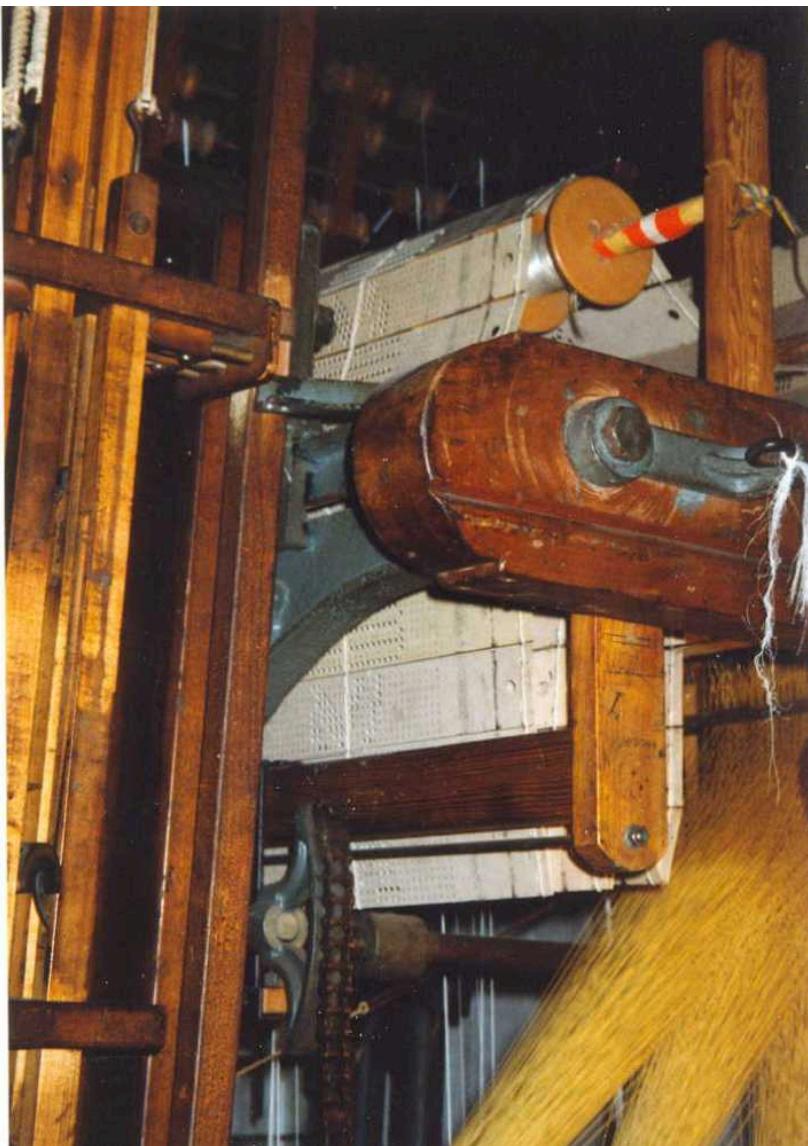
Praktisches Beispiel: Diese Formel könnte in Arduino-Code so aussehen: `if((sensor_A && sensor_B) || !sensor_C)` - identisch zu Booles Notation von 1854! Ihr ATmega328P führt solche Operationen in seinen ALU-Gattern hardware-nativ aus.

Im Rahmen dieser Vorlesung werden wir die boolesche Algebra als Grundlage für die digitale Logik verwenden.

Warum ist das revolutionär? Boole schuf die mathematische Grundlage für ALLE digitale Elektronik. Ohne seine Algebra gäbe es keine Transistor-Gatter, keine CPUs, keine Mikrocontroller. Ihr Arduino ist Booles Theorie in Silizium gegossen!

Joseph-Marie Jacquard (1752 - 1834) - Automatischer Webstuhl

Jacquards "Musterwebstuhl" realisierte die Ansteuerung der Webmechanik durch eine Lochkartensteuerung. Im Jahr 1805 wurde das Verfahren erstmals vorgestellt. Dadurch konnten endlose Muster von beliebiger Komplexität mechanisch hergestellt werden.



Die Lochkartensteuerung einer Jacquard-Maschine im Historischen Zentrum Wuppertal (Autor: Markus Schweiß, Die Lochkartensteuerung einer Jacquard-Maschine im Historischen Zentrum Wuppertal, https://de.wikipedia.org/wiki/Joseph-Marie_Jacquard#/media/Datei:Jacquard01.jpg)

Auf den Karte waren Informationen über das in einem Schritt zu webende Muster enthalten. Ein Loch bedeutete Fadenhebung, kein Loch eine Fadensenkung. Dabei konnten die Lochkarten in einer Endlosschleife gekoppelt werden, um wiederkehrende Strukturen umzusetzen.

Charles Babbage (1791 - 1871) - Analytical Engine

Ausgangspunkt war die Konstruktion einer Rechenmaschine für die Lösung polynomialer Funktionen. Dabei entstand die Vision einer universellen Rechenapparatur, die auf der Basis eines programmierbaren Systems Berechnungen löst. Die erste Beschreibung wurde 1837 veröffentlicht.

- Energiebereitstellung über eine Dampfmaschine
- 8000 mechanische Komponenten
- Eingabe der Daten und Befehle über Lochkarten
- Nutzerinterface: Drucker, ein Kurvenplotter und eine Glocke
- Zahlendarstellung: dezimale Festkommazahlen, pro Stelle ein Zahnrad
- Arbeitsspeicher zwischen 1,6 und 20 kB (umstritten)

The result of my reflections has been that numbers containing more than thirty places of figures will not be required for a long time to come.

Auch Genies irren. Babbage blieb den dezimalen Zahlen treu, obwohl die binäre Darstellung schon damals bekannt war. Die mechanische Realisierung von Zahnrädern für jede Ziffer machte die Maschine extrem komplex und fehleranfällig.

Eine weitere Parallele zu heute: Babbage dachte, 30 Dezimalstellen würden "für lange Zeit" ausreichen. Heute rechnen wir mit 32-Bit-Zahlen - das sind etwa 10 Dezimalstellen. Manchmal liegen die Visionäre richtig, aber aus den falschen Gründen!

Die Maschine wurde zu Lebzeiten von Babbage nicht realisiert und nur in Teilen durch seinen Sohn implementiert. Aktuell existieren in verschiedenen Museen unterschiedliche Neubauten.



[Video auf YouTube ansehen](#)

Fehler 153

Fehler bei der Konfiguration des Videoplayers





Video auf YouTube ansehen

Fehler 153

Fehler bei der Konfiguration des Videoplayers



Eine der zentralen Persönlichkeiten, die die Möglichkeiten der Analytical Engine erkannte, war Ada Lovelace.

„[Die Analytical Engine] könnte auf andere Dinge als Zahlen angewandt werden, wenn man Objekte finden könnte, deren Wechselwirkungen durch die abstrakte Wissenschaft der Operationen dargestellt werden können und die sich für die Bearbeitung durch die Anweisungen und Mechanismen des Gerätes eignen.“

Ada Lovelace legte in den Notes zu einem Vortrag von Babbage einen schriftlichen Plan zur Berechnung der Bernoulli-Zahlen in Diagrammform vor, welcher als das erste veröffentlichte formale Computerprogramm gelten kann.

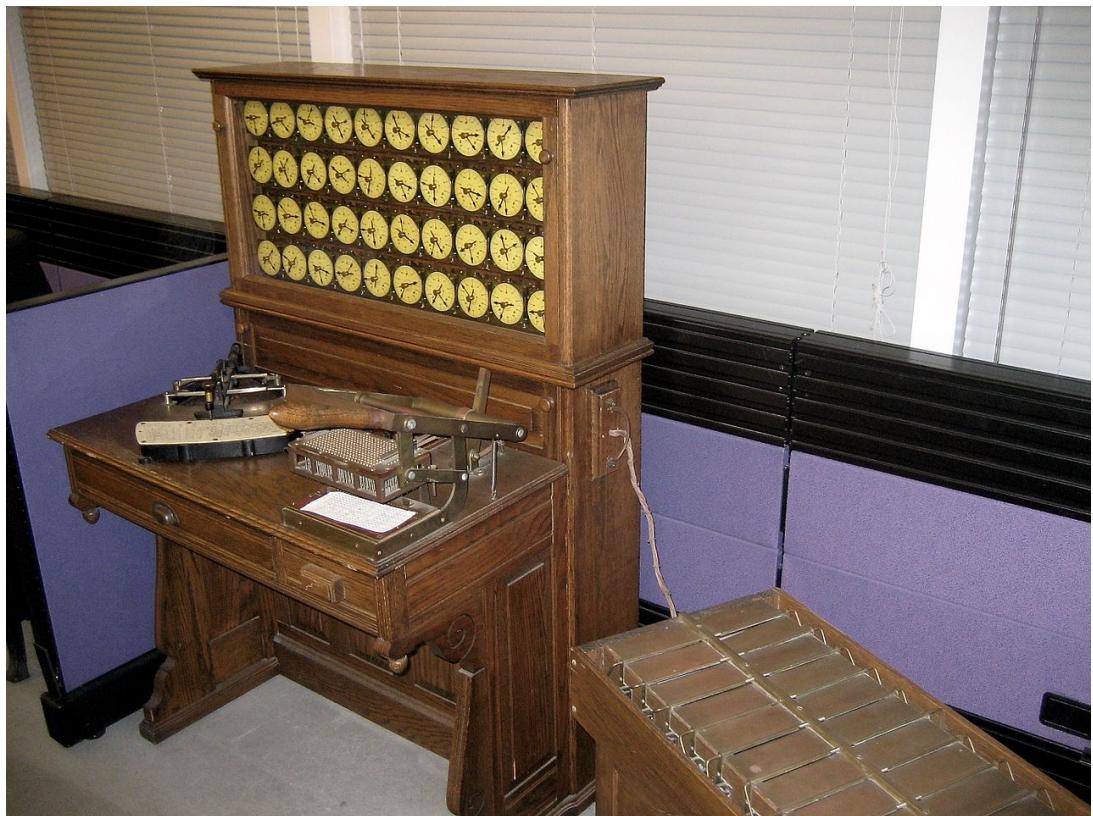
Es werde Licht ... elektrische Systeme

Der Verfügbarkeit des elektrischen Stromes als Energiequelle löste einige der technischen Hürden bei den mechanischen Rechenmaschinen, eröffnete aber auch neue Möglichkeiten bei der Eingabe von Daten.

Herman Hollerith (1860 - 1929) interpretierte die Lochkarten als Medium neu. Sein Konzept für die Lösung/Auswertung von organisatorischen Problemstellungen sah diese als Basis für die Datenerfassung.

Das System für die Erfassung von Daten auf Lochkarten bestand aus der Tabelliermaschine, dem Lochkartensortierer, dem Lochkartenlocher und dem Lochkartenleser. Damit konnte die Volkszählung in den USA 1890 innerhalb von 2 Jahren ausgewertet werden.

1924 wurde die von ihm gegründete Firma schließlich in International Business Machines Corporation (IBM) umbenannt.



Nachbau einer Hollerith Maschine (Autor: Adam Schuster, Replica of early Hollerith punched card tabulator and sorting box (right) at Computer History Museum, <https://commons.wikimedia.org/wiki/File:HollerithMachine.CHM.jpg>)



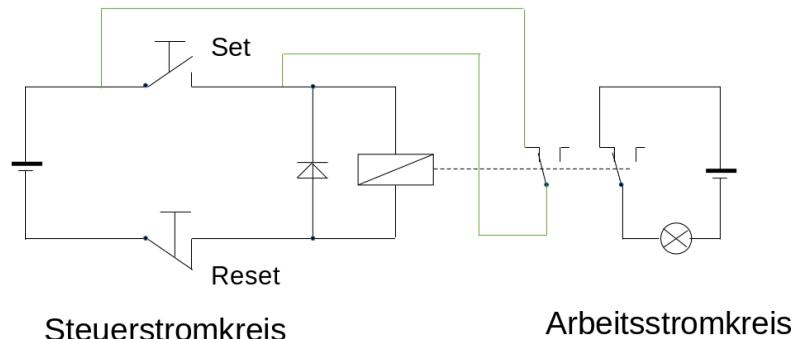
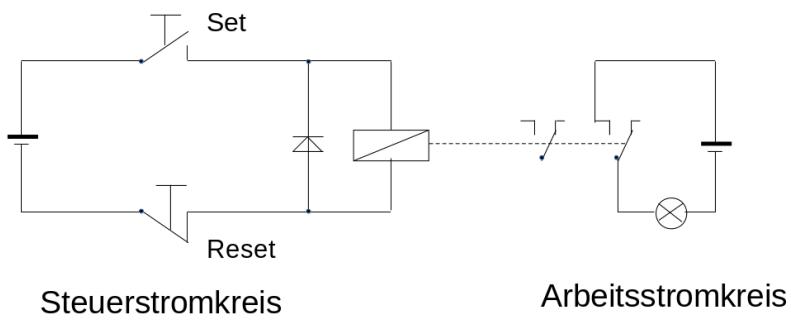
[Video auf YouTube ansehen](#)

Fehler 153

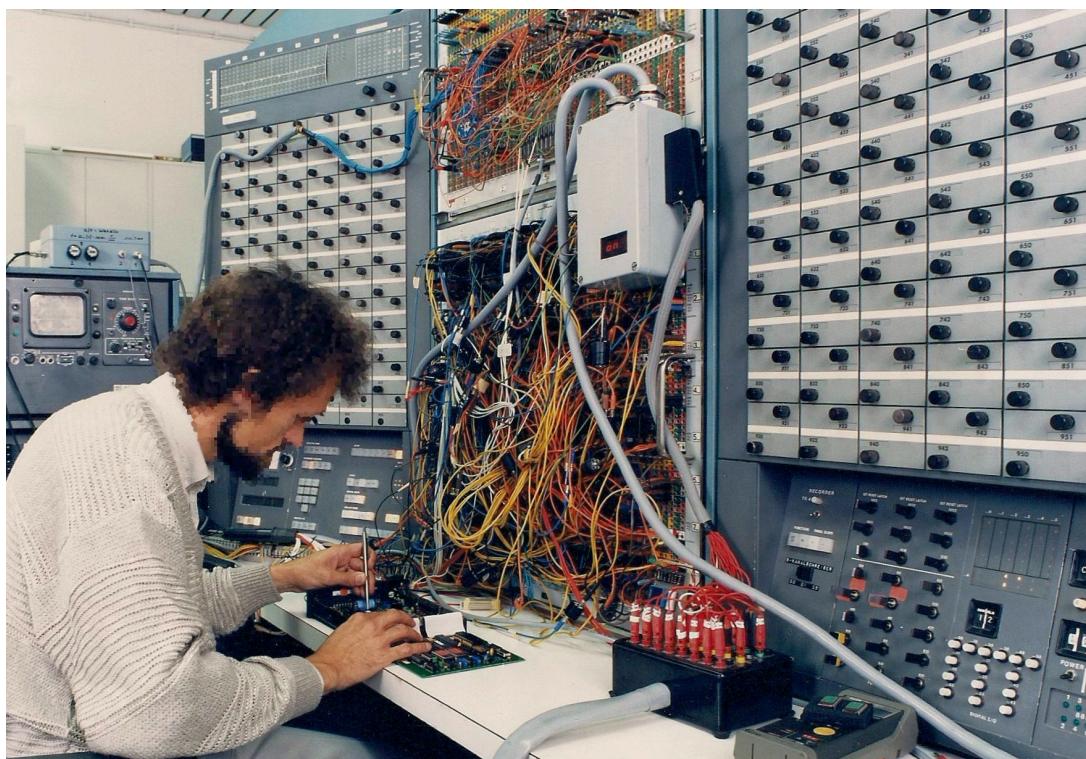
Fehler bei der Konfiguration des Videoplayers



Nicht nur auf der Ebene der Datenerfassung, sondern auch für die Datenspeicherung eröffnete sich auf der Basis des elektrischen Stromes eine Revolution. Relais konnten Zustände nun speichern und logische Operationen abbilden. Damit manifestierte sich aber auch die Festlegung auf eine binäre Informationsdarstellung - An, Aus (1 und 0).



Bis in die 80er Jahre bildeten Analogrechner einen alternativen Ansatz. Anders als bei den diskret arbeitenden Digitalrechnern wurde hier im Werte- und Zeitverlauf kontinuierlich gearbeitet. Dabei wurde das Systemverhalten von komplexen Systemen mit elektrischen Schaltungen nachgebildet.

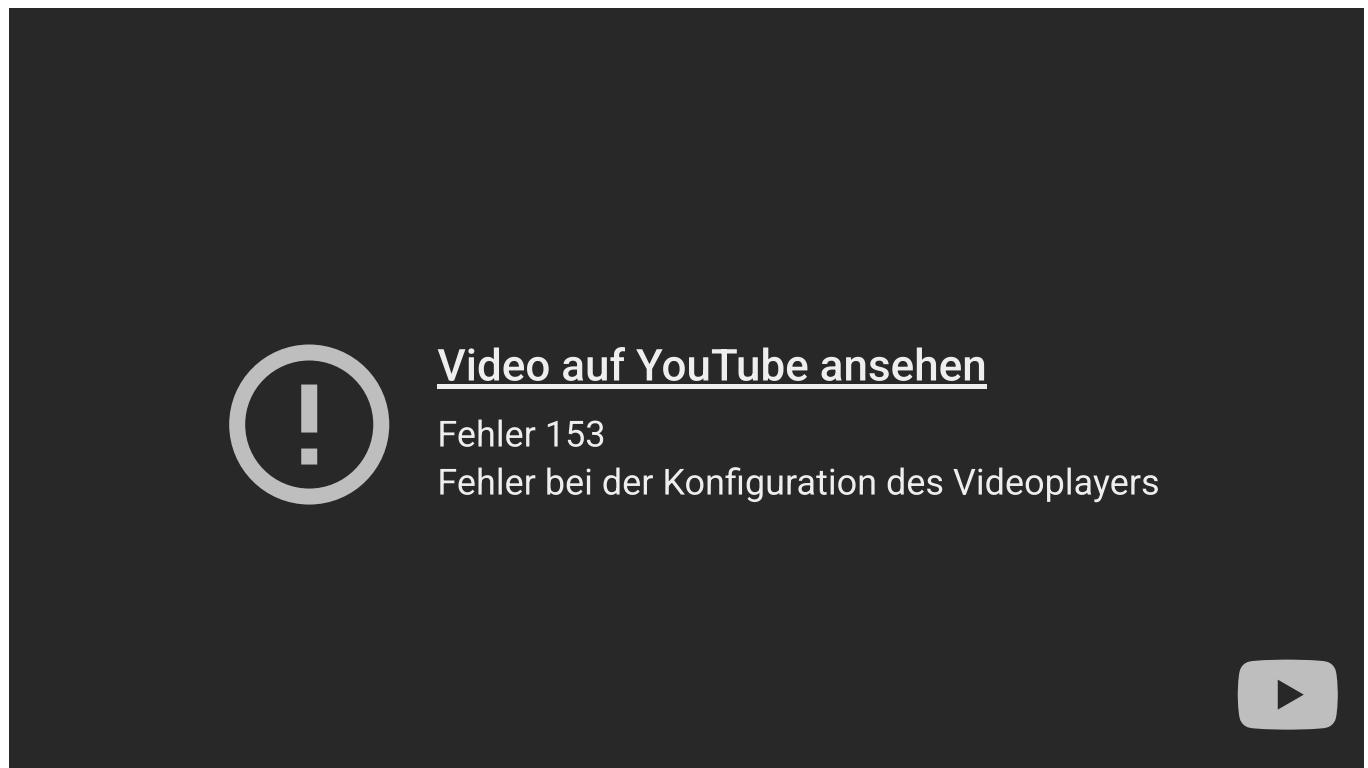


Analogrechneranwendung für die Simulation der Regelstrecke für einen Regelkreis (Autor: SchmiAlf, Analogrechneranwendung für die Simulation der Regelstrecke für einen Regelkreis mit einem externen Steuergerät (Mikrocontroller) angeschlossen als Hardware-in-the-Loop, mit Analogrechner EAI-8800, ca. 1985, https://commons.wikimedia.org/wiki/File:Analogrechner_HW-in-Loop_Ausschnitt.jpg)

Zuse Z3

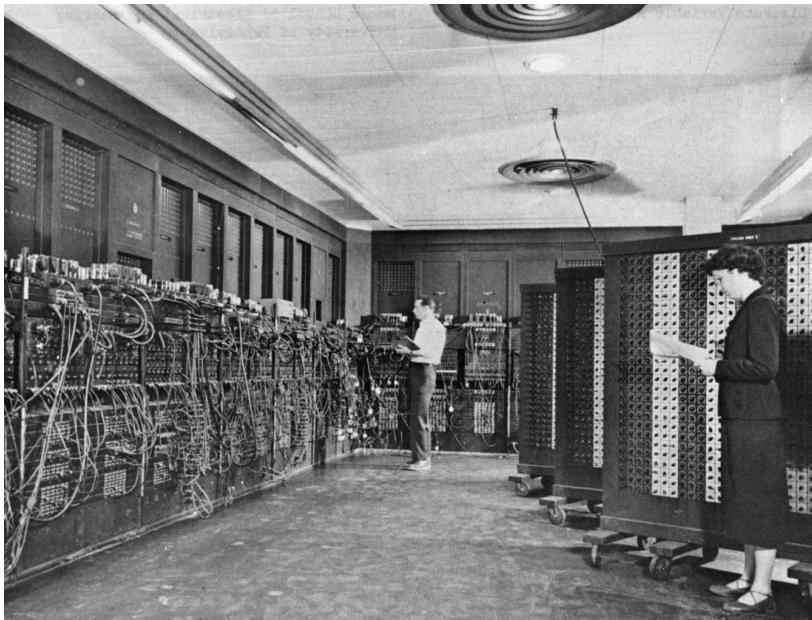
Die Z3 war der erste funktionsfähige Digitalrechner weltweit und wurde 1941 von Konrad Zuse in Zusammenarbeit mit Helmut Schreyer in Berlin gebaut. Die Z3 wurde in elektromagnetischer Relaistechnik mit 600 Relais für das Rechenwerk und 1400 Relais für das Speicherwerk ausgeführt.

- 10 Hertz Taktfrequenz
- basierend auf 2200 Relais
- 22-stellige Binärzahlen (im Gleitkomma-Format !)
- dezimale Ein-/Ausgabe
- Speicher mit 64 Worten
- Steuereinheit mit Sequenzer
- Addition in 3 Takten, Multiplikation in 16 Takten
- keine Sprungoperationen!



Bereits Vorwegname der Kernelemente moderner Architekturen:

- Gleitkommaformat
- Mikroprogrammierung
- Pipeline
- 20 Akkumulatoren, 1 Multiplizierer, 3 Funktionstabellen
- programmiert durch Kabel-Verbindungen
- E/A mittels Lochkarten
- gebaut für ballistische Berechnungen



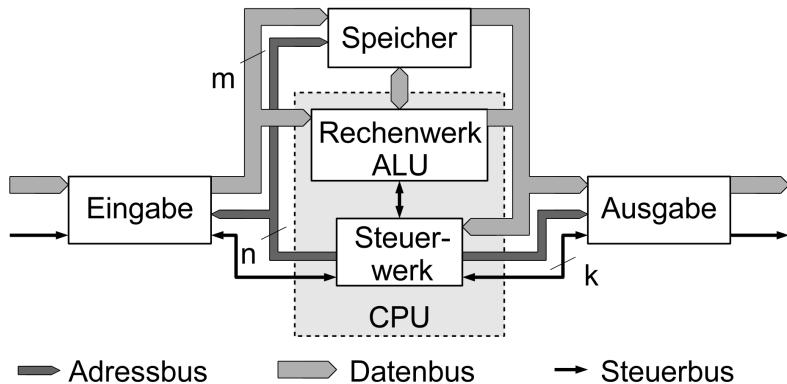
ENIAC Betriebsraum (Autor unbekannt, ENIAC in Philadelphia, Pennsylvania. Glen Beck (background) and Betty Snyder (foreground) program the ENIAC in building 328 at the Ballistic Research Laboratory ,
<https://commons.wikimedia.org/wiki/File:Eniac.jpg>)

Ein großes Problem bei der Entwicklung des ENIAC war die Fehleranfälligkeit der Elektronenröhren. Wenn nur eine der 17.468 Röhren ausfiel, rechnete die gesamte Maschine fehlerhaft.

Konzeptionelle Entwürfe

In seinem Papier *First Draft on the Report of EDVAC* beschreibt John von Neumann 1945 die Basiskomponenten eines Rechners:

- ALU (Arithmetic Logic Unit) – Rechenwerk für die Durchführung mathematischer/logischer Operationen
- Control Unit – Steuerwerk für die Interpretation der Anweisungen eines Programmes
- BUS – Bus System, dient zur Kommunikation zwischen den einzelnen Komponenten (Steuerbus, Adressbus, Datenbus)
- Memory – Speicherwerk sowohl für Programme als auch für Daten
- Ein-/Ausgabe – Nutzerinterface



Von-Neumann Architektur (Medvedev, Schaltbild einer Von-Neumann-Architektur auf deutsch.:
https://commons.wikimedia.org/wiki/File:%22von_Neumann%22_Architektur_de.svg)

Die Transistor-Ära

1948 stellen Shockley, Bardeen und Brattain den ersten Transistor an den Bell Labs her. Dafür erhalten sie 1956 den Nobelpreis für Physik. Der Transistor verdrängt langsam die Röhre als Verstärker und Schalter. Die neue Technik ermöglicht die Erstellung integrierter Schaltungen.



[Video auf YouTube ansehen](#)

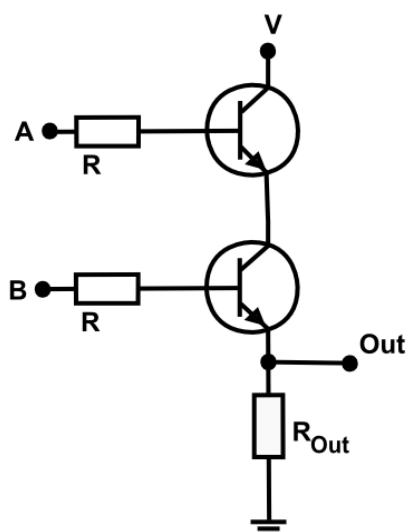
Fehler 153

Fehler bei der Konfiguration des Videoplayers



Und wie wird daraus nun ein Rechner? Die intelligente Verschaltung mehrerer Transistoren ermöglicht die Umsetzung von logischen Schaltungen wie AND, OR usw.

Transistor AND Gate



Transistorverschaltung für AND-Logik (Autor: Datenblatt der Firma Fairchild, DM7408, August 1986)

Diese wiederum fassen wir nun in entsprechenden ICs zusammen. Wir haben die elektrische Ebene verlassen und sind endgültig auf der logischen Ebene angekommen.

DM7408

Quad 2-Input AND Gates

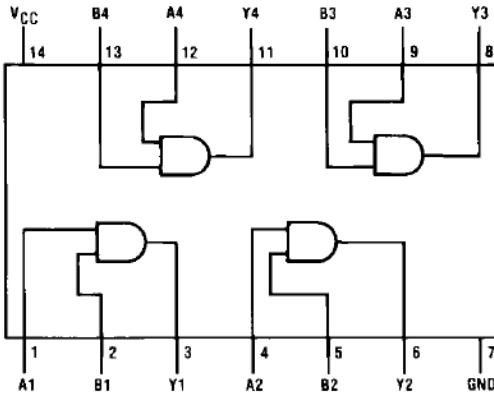
General Description

This device contains four independent gates each of which performs the logic AND function.

Ordering Code:

Order Number	Package Number	Package Description
DM7408N	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide

Connection Diagram



Function Table

Inputs		Output
A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H

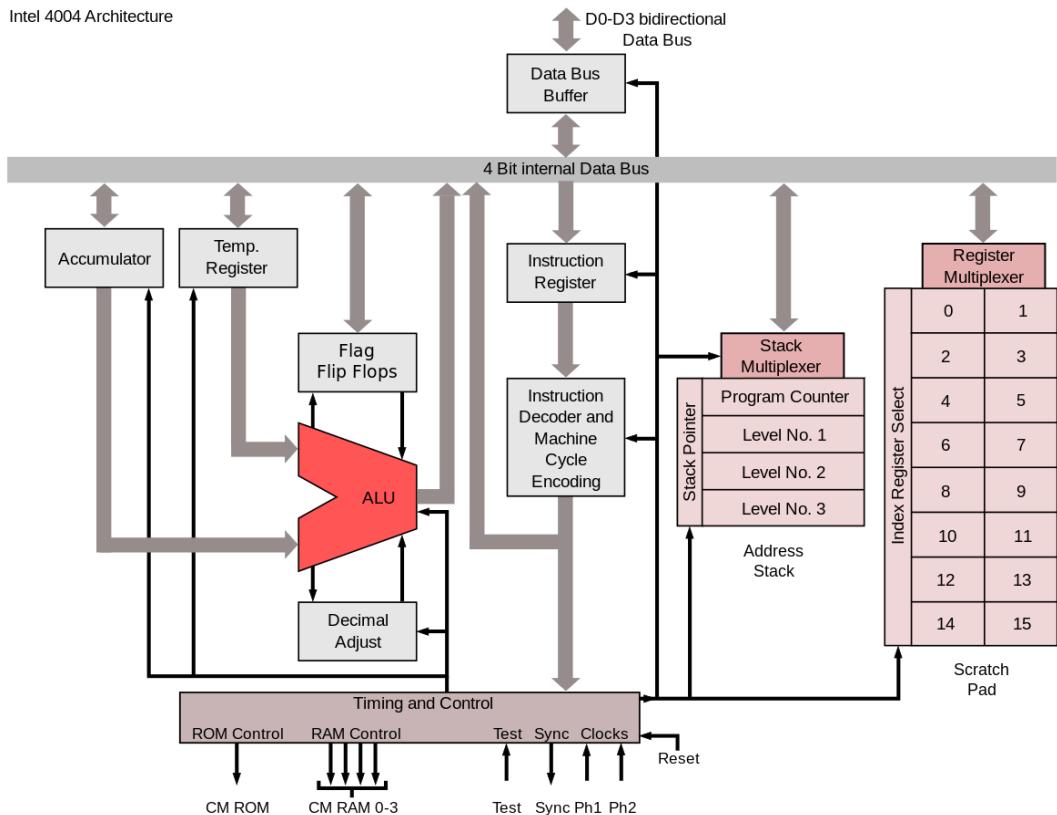
H = HIGH Logic Level
 L = LOW Logic Level

AND-IC (Autor: Datenblatt der Firma Fairchild, DM7408, August 1986)

[ANDTransistor] EBattleP, diagram of a transistor AND gate. Reference: <http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/and.html#c1>, <https://commons.wikimedia.org/wiki/File:TransistorANDgate.png>

Intel 4004

Beispiel: Intel 4004-Architektur (1971)



Intel 4004 Architekturdarstellung (Autor: Appaloosa, Intel 4004,

https://upload.wikimedia.org/wikipedia/commons/thumb/8/87/4004_arch.svg/1190px-4004_arch.svg.png

- Anzahl Transistoren: 2300
- Taktfrequenz: 500 bis 740 kHz
- Zyklen pro Instruktion: 8
- Daten-Adressraum: 5120 Bit (Harvard-Architektur)
- Anzahl Befehle: 46
- Bauform: 16 Pin (DIP)

Halten Sie nach der GoldCap-Variante Ausschau!

Jetzt wird's spannend: Vergleichen wir das mit Ihrem Arduino! Der ATmega328P hat etwa 100.000 Transistoren (40x mehr), läuft mit 16 MHz (20x schneller) und hat 131 Befehle.

Intel 4004 (1971) vs. ATmega328P (2008):

Eigenschaft	Intel 4004	ATmega328P	Faktor
Transistoren	2.300	~100.000	43x
Taktfrequenz	740 kHz	16 MHz	22x
Befehle	46	131	3x
Wortbreite	4 Bit	8 Bit	2x
RAM	80 Byte	2048 Byte	26x

Achtung! Die Angaben zum ATmega328P sind Schätzungen, da die Architektur proprietär ist (vgl. zum Beispiel [Link](#)).

Was sagt die Befehlsanzahl aus?

Offenbar ist das Steuerwerk des ATmega deutlich komplexer und kann kompliziertere Befehle verarbeiten. Es fehlen dem Intel 4004 viele der heute üblichen Operationen (z.B. Multiplikation, Sleep Modi, Watch-Dog Reset). Das bedeutet, dass Programme auf dem 4004 länger sind (mehr Programmspeicher) und zusätzlich zur geringen Taktrate mehr "Schritte" brauchen.

Unterstützung für die Interpretation aus dem Nutzerhandbuch, dass das Instruction set beschreibt:

4004 Instruction Set

BASIC INSTRUCTIONS (* = 2 Word Instructions)

Hex Code	MNEMONIC	OPR D ₃ D ₂ D ₁ D ₀	OPA D ₃ D ₂ D ₁ D ₀	DESCRIPTION OF OPERATION
00	NOP	0 0 0 0	0 0 0 0	No operation.
1 -	*JCN	0 0 0 1 A ₂ A ₂ A ₂ A ₂	C, C ₂ C ₃ C ₄ A ₁ A ₁ A ₁ A ₁	Jump to ROM address A ₂ A ₂ A ₂ A ₂ , A, A ₁ A ₁ A ₁ (within the same ROM that contains this JCN instruction) if condition C, C ₂ C ₃ C ₄ is true, otherwise go to the next instruction in sequence.
2 -	*FIM	0 0 1 0 D ₂ D ₂ D ₂ D ₂	R R R 0 D ₁ D ₁ D ₁ D ₁	Fetch immediate (direct) from ROM Data D ₂ D ₂ D ₂ D ₂ , D, D ₁ D ₁ D ₁ to index register pair location RRR.

■ ■ ■

8 -	ADD	1 0 0 0	R R R R	Add contents of register RRRR to accumulator with carry.
9 -	SUB	1 0 0 1	R R R R	Subtract contents of register RRRR to accumulator with borrow.
A -	LD	1 0 1 0	R R R R	Load contents of register RRRR to accumulator.
B -	XCH	1 0 1 1	R R R R	Exchange contents of index register RRRR and accumulator.
C -	BBL	1 1 0 0	D D D D	Branch back (down 1 level in stack) and load data DDDD to accumulator.
D -	LDM	1 1 0 1	D D D D	Load data DDDD to accumulator.
F0	CLB	1 1 1 1	0 0 0 0	Clear both. (Accumulator and carry)
F1	CLC	1 1 1 1	0 0 0 1	Clear carry.
F2	IAC	1 1 1 1	0 0 1 0	Increment accumulator.

Intel 4004 Instruction Set (Autor: Intel 4004 Assembler, <http://e4004.szy.org/asm.html>)

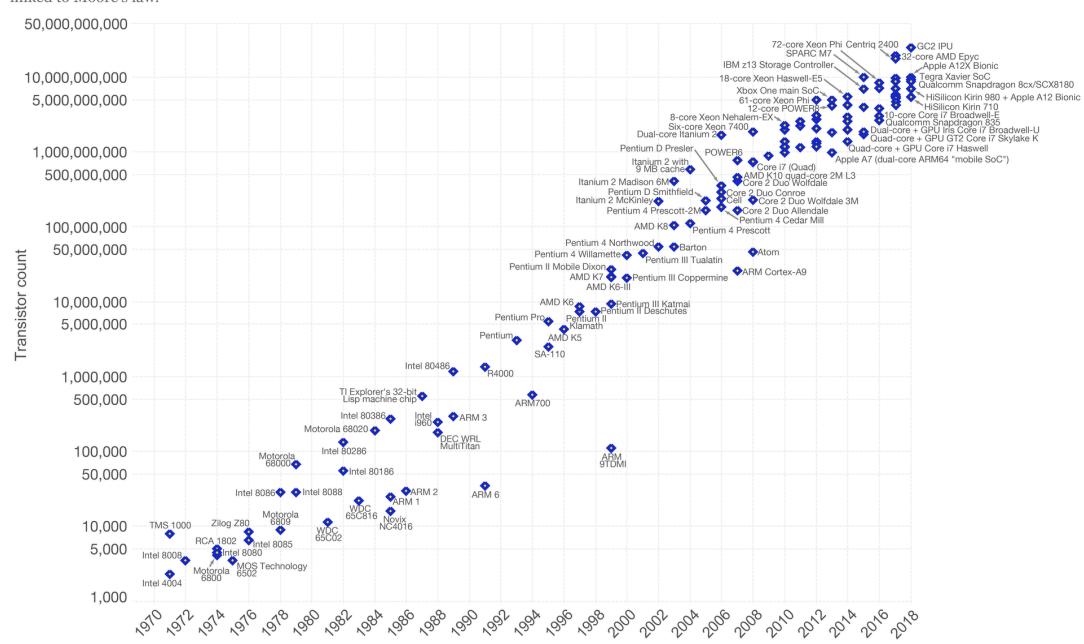
Weitere Entwicklung

Jahr	Entwicklung	Beschreibung
1971	Intel 4004	Erste kommerzielle CPU auf einem einzigen Chip (4-Bit).
1978	Intel 8086	16-Bit-Architektur und Grundlage für die x86-Architektur.
1979	Motorola 68000 mit 32-Bit interner Architektur	
1985	RISC-Architektur	Einführung der RISC (Reduced Instruction Set Computer)-Architektur für effizientere CPU-Designs.
1989	Intel i860 und erste GPUs	Erste Grafikprozessoren (GPUs) für 3D-Beschleunigung und spezielle Berechnungen.
1993	Intel Pentium	Einführung von Super-Skalaren und schnellerer Rechenleistung bei x86-Prozessoren.
1999	Nvidia GeForce 256	Erste „GPU“ zur Hardware-Beschleunigung von 3D-Grafikberechnungen.
2000	AMD Athlon 64	Erste 64-Bit-Desktop-CPU.
2006	Mehrkernprozessoren (Intel Core, AMD Athlon X2)	Einführung von Mehrkernprozessoren für verbesserte Leistung und parallele Verarbeitung.
2010	Nvidia Fermi-Architektur	CUDA-Architektur für Parallelverarbeitung und GPGPU (General Purpose GPU Computing).
2011	Intel Sandy Bridge	Einführung der integrierten Grafik mit CPU und GPU auf demselben Chip für höhere Effizienz.

Technologie	Erste Anwendung	Merkmale
2015	AMD HBM (High Bandwidth Memory)	Einführung von HBM für schnelle und effiziente Speichernutzung bei GPUs.
2016	Nvidia Pascal-Architektur und Deep Learning	GPUs mit optimierter Leistung für KI und Deep Learning (Tesla P100).
2020	Apple M1 SoC	Erster ARM-basierter Desktop-Prozessor von Apple
2022	Nvidia Ada Lovelace-Architektur	Fortschrittliche Architektur für Raytracing und KI-Beschleunigung mit DLSS 3-Technologie.

Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's Law: The number of transistors on integrated circuit chips (1971-2010). Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Licensed under CC-BY-SA by the author Max Roser.

Moore's Law (Autor: https://commons.wikimedia.org/wiki/File:Moore%27s_Law_Transistor_Count_1971-2018.png, Max Roser)

Warum ist das alles für Ihr Verständnis wichtig?

Weil ein solides Hardwarewissen Ihnen hilft Software besser zu verstehen ... und falsche Statements zu erkennen!

Die historische Entwicklung zeigt Muster auf, die sich wiederholen - auch bei den Fehlprognosen:

I think there is a world market for about five computers. (Thomas J. Watson Jr., chairman of IBM, 1943)

Where a calculator as the ENIAC is equipped with 18000 vacuum tubes and weighs 30 tons, computers in the future may have only 1000 vaccum tubes and weigh 1 1/2 tons. (Popular Mechanics, 1949)

There is no reason anyone would want a computer in their home. (Ken Olsen, founder of Digital Equipment Corporation, 1977)

The Internet will catastrophically collapse in 1996. (Robert Metcalfe, Ethernet inventor, 1995)

Two years from now, spam will be solved. (Bill Gates, Microsoft, 2004)

The iPhone has no chance of getting any significant market share. (Steve Ballmer, Microsoft CEO, 2007)

No one will ever need more than 640 kB of memory for a personal computer. (Bill Gates, 1981 - angeblich)

Smartphones will never replace cameras. (Kodak executives, 2000s)

Hausaufgabe

- Setzen Sie sich mit den Unterschieden zwischen der Z1 und der Z3 auseinander.
- Woher stammt der Begriff "Bug" in Bezug auf die Programmierung?