Von-Neumann-Rechner

(John von Neumann : 1903-1957)

C. BABBAGE (1792 – 1871): Programmgesteuerter (mechanischer) Rechner

Quelle: http://www.cs.uakron.edu/~margush/465/01_intro.html
Analytical Engine - Calculate general formulas under the control of a looping program stored on punch cards (1834)

Konrad Zuse (1910-1995): Erster programmgesteuerte Rechner, der die in modernen Rechnern zu findenden Prinzipien verkörpert, war die Z3. Sie arbeitete elektromechanisch.
Anzahl der Relais: ca. 2500
Taktfrequenz: ca. 5 Hertz; Addition ca. 0,8 Sek., Multiplikation ca. 3 Sek.
Gewicht: ca. 1 Tonne

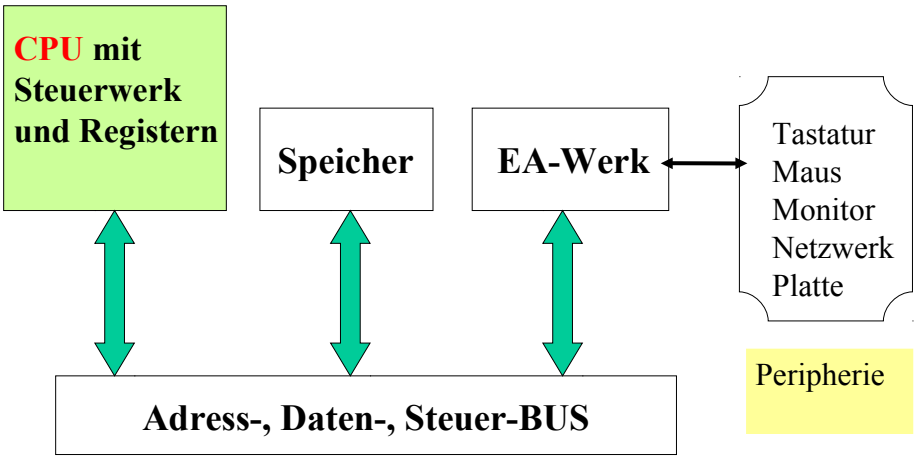
John von Neumann (1903 - 1957)
Er schlug 1946 ein Konzept zur Gestaltung eines universellen Rechners vor. Die heutigen Rechenanlagen orientieren sich an der Struktur dieses klassischen Universalrechners.

Der VN-Rechner besteht aus 5 Funktionseinheiten: Steuerwerk + Rechenwerk (= CPU), Speicher und E/A-Werk und Bus.

Anmerkung
Die wesentlichen Ideen der so genannten Von-Neumann-Architektur wurden jedoch schon 1936 von Konrad Zuse ausgearbeitet, in zwei Patentschriften von 1937 dokumentiert und größtenteils bereits 1938 in der Z1 realisiert.
(Quelle: http://www.heise.de/ct/artikel/Der-Rechenplaner-1022815.html)

Von-Neumann-Rechner (1946)

(Moderne Princeton-Architektur)



Princeton-Architektur des von-Neumann-Rechners

Das im Jahre 1946 von John von Neumann vorgeschlagene Konzept beschreibt die Struktur des klassischen **Universalrechners**.

Konzept des Universalrechners
Das Programm zur Lösung eines Problems ist im Speicher abgelegt. Die Struktur des VNR ist **unabhängig** von dem zu lösenden Problem.

Steuerwerk und Rechenwerk werden zu einer Einheit zusammengefasst, die als Prozessor oder CPU (= Central Processing Unit) bezeichnet wird.

Alle Komponenten der CPU werden durch das **Steuerwerk** beeinflusst:

- Laden des nächsten Befehls aus dem Speicher
- Dekodierung der Befehle
- Steuerung der Befehlsausführung
- Laden und Ablage von Daten aus dem bzw. in den Speicher

Eine Voraussetzung für das Zusammenwirken der Funktionseinheiten ist die Übertragung von Daten, Befehlen, Adressen, sowie Kontrollinformationen über den **BUS**:

- Der **Datenbus** überträgt Daten; er arbeitet bidirektional. Seine Breite entspricht i.a. der Anzahl der Bits einer Speicherzelle des Arbeitsspeichers. Der Inhalt einer Speicherzelle kann somit in einem Takt übertragen werden.
- Der **Adressbus** dient der Übertragung der von der Steuereinheit berechneten Speicheradressen zum Speicher und zum EA-Werk.
- Der **Steuerbus** (auch Kontrollbus genannt) dient der Übermittlung von Steuersignalen zwischen Steuerwerk und den übrigen Funktionseinheiten. Über diesen Bus wird signalisiert, für welche Funktionseinheiten die jeweils auf Adress- und Datenbus anliegenden Daten bestimmt sind. Weiterhin wird die Zugriffsart Lesen oder Schreiben festgelegt.

Von-Neumann-Prinzipien

1. Alle Daten (Programmbefehle, Adressen, Werte) sind **binär** codiert. Die möglichen Zustände innerhalb einer Speicherzelle werden mit 0 oder 1 bezeichnet.

2. Der **Speicher** ist in gleichgroße Zellen unterteilt, die fortlaufend durchnummeriert sind. Über die Nummer (=Adresse) einer Speicherzelle kann deren Inhalt abgerufen oder verändert werden.

3. **Vereinigung von Daten- und Programmspeicher**
   Programm und Daten sind in demselben Speicher abgelegt.

4. Ein Programm besteht aus **Arbeitsschritten (= Befehle)**. Die Befehle eines Programms sind in aufeinander folgenden Speicherzellen abgelegt.

5. Es gibt folgende Befehlsarten:
   Arithmetische Befehle (Addieren, Multiplizieren usw.)
   Logische Befehle (Vergleiche, logisches NICHT, UND, ODER, ...)
   Transportbefehle (z.B. vom Speicher zum Rechenwerk; für die Ein-/ Ausgabe)
   Durch **Sprungbefehle** kann von der Bearbeitung der Befehle in der gespeicherten Reihenfolge abgewichen werden.
   Befehle zur Programmablaufsteuerung
   (Programmende, Programmzähler initialisieren, ..)

Geeignete Schaltwerke im **Steuerwerk** sorgen für die richtige Entschlüsselung (Decodierung).

Der Befehlszyklus

Beim Start eines Programms wird das **Befehlszähler PC** in der CPU auf die Anfangsadresse des Programms gesetzt.

1. Phase: Ladephase (Fetch-Phase)
Den durch PC adressierten Befehl aus dem Speicher in das Befehlsregister laden. Ein Befehl besteht aus dem **Operationsteil** und dem **Adressteil**.

2. Phase: Update-Phase
Befehlszähler verändern: PC = PC + 1

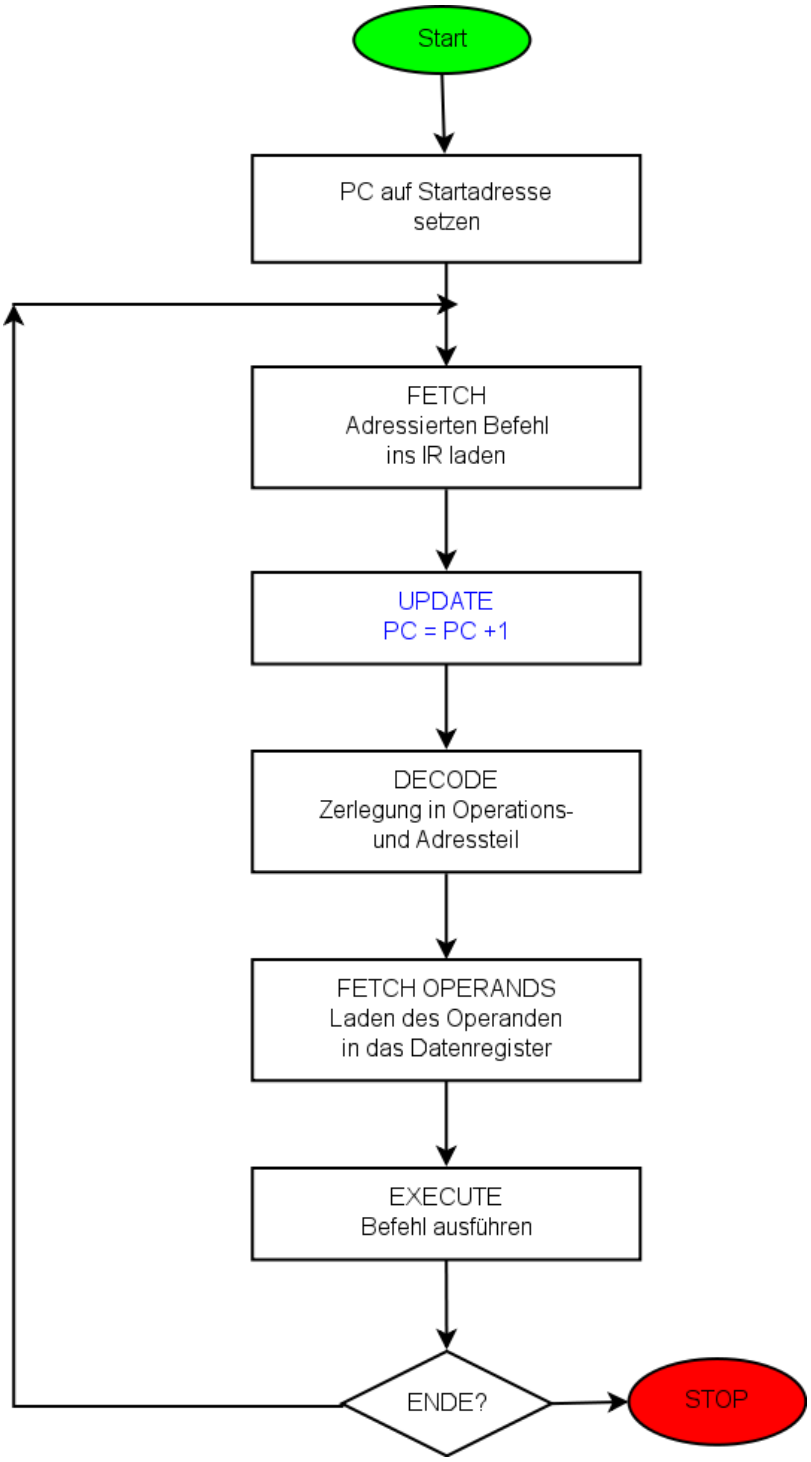
3. Phase: Decode-Phase
Der Befehl wird im Dekodierer (Teil des Steuerwerks) entschlüsselt.

4. Phase: Fetch Operands
Laden von Daten (Operanden) aus dem Speicher in das Datenregister.

5. Phase: Execution-Phase
Befehlsausführung im Rechenwerk.

Weiter mit Phase 1

Von-Neumann-Befehlszyklus



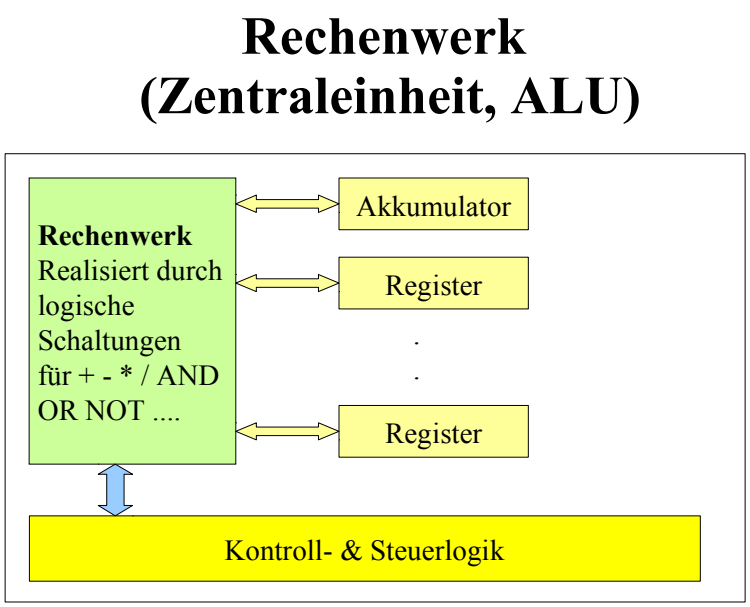
Das Rechenwerk (ALU)

Das Rechenwerk (ALU Arithmetic Logical Unit) ist die Komponente des Rechners, in der arithmetische (z.B. Addition, Subtraktion, Multiplikation, Division) und logische Verknüpfungen (z.B. **UND**, **ODER**, **NICHT**) durchgeführt werden.

Die für die Verknüpfungen notwendigen Operanden werden dem Rechenwerk aus dem Speicher zugeführt. Die schnellen Speicher im Rechenwerk heißen **Register**.

In einem speziellen Register, dem **Akkumulator**, werden die Zwischenergebnisse gespeichert.

In weiteren Registern werden Statusinformationen, wie z.B. ein Überlauf bei der Zahlenbereichsüberschreitung vermerkt.



Assembler

Da die Programmbefehle in Maschinensprache schwer lesbar sind, verwendet man eine Assemblersprache. Symbolische Bezeichner (**Mnemonics**) entsprechen den Maschinenbefehlen. Im Sprachgebrauch wird oft auch die Assemblersprache als Maschinensprache bezeichnet.

Die virtuelle Java-Maschine

javap ist ein Disassembler für Klassendateien. Er macht den Maschinencode der virtuellen Maschine lesbar.

Ein Beispiel Die Datei ***Einfach.class*** als Java-Quellcode:

```
public class Einfach1 {

        public static void main(String[] args) {

                int zahl1= 20;
                int zahl2=7;

                int ergebnis = zahl1 + zahl2 +5;

        }

}
```

Der erzeugte Code (in Auszügen)

axel@informatik1:~/Dokumente/java/Test/bin> javap -l -v Einfach1

```
Compiled from "Einfach1.java"
public class Einfach1 extends java.lang.Object
  SourceFile: "Einfach1.java"
….


{
public Einfach1();
…..........................

public static void main(java.lang.String[]);
  LineNumberTable:
   line 10: 0
   line 11: 3
   line 13: 6
   line 15: 12
```

```
 LocalVariableTable:
  Start  Length  Slot  Name   Signature
  0      13      0    args       [Ljava/lang/String;
  3      10      1    zahl1      I
  6      7       2    zahl2      I
  12     1       3    ergebnis   I


 Code:
  Stack=2, Locals=4, Args_size=1
  0:   bipush  20
  2:   istore_1
  3:   bipush  7
  5:   istore_2
  6:   iload_1
  7:   iload_2
  8:   iadd
  9:   iconst_5
  10:  iadd
  11:  istore_3
  12:  return


}
```

Simulation
Um die Funktionsweise des Von-Neumann-Computers im Prinzip zu verstehen, vereinfachen wir die Vorgänge und programmieren in einer **hypothetischen Maschinensprache**. Wir sind dann unabhängig vom jeweils eingebauten Prozessormodell. Es soll eine reduzierte Befehlsliste zur Verfügung stehen.

→ Didaktischer Computer
Link: http://www.oberstufeninformatik.de/dc/index.html