3. Funktionsweise eines Rechners

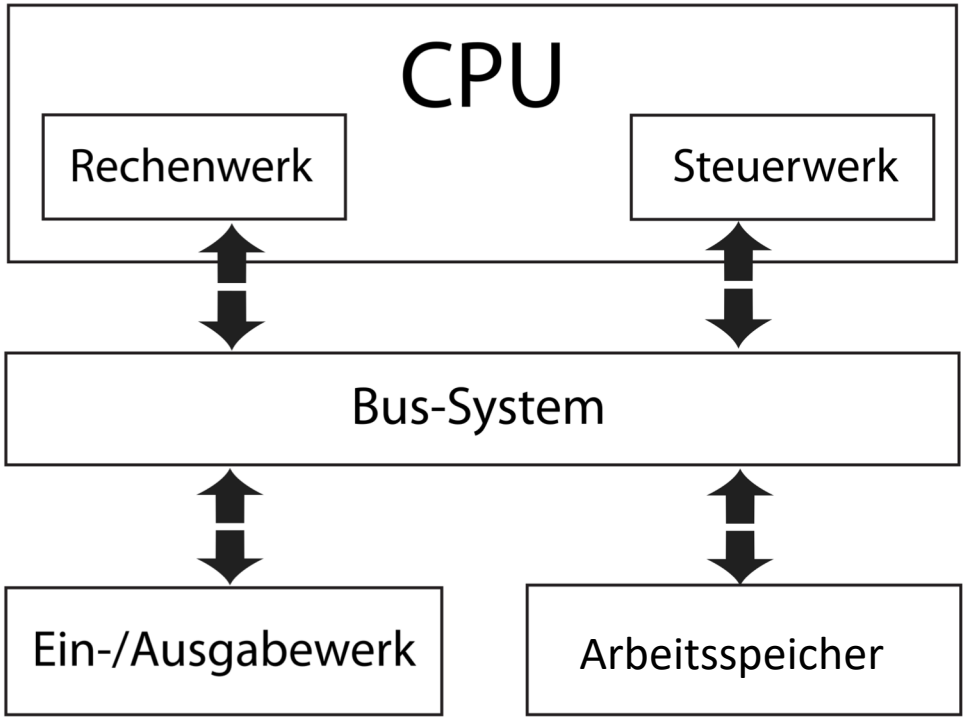
3.2 Registermaschine als Modell

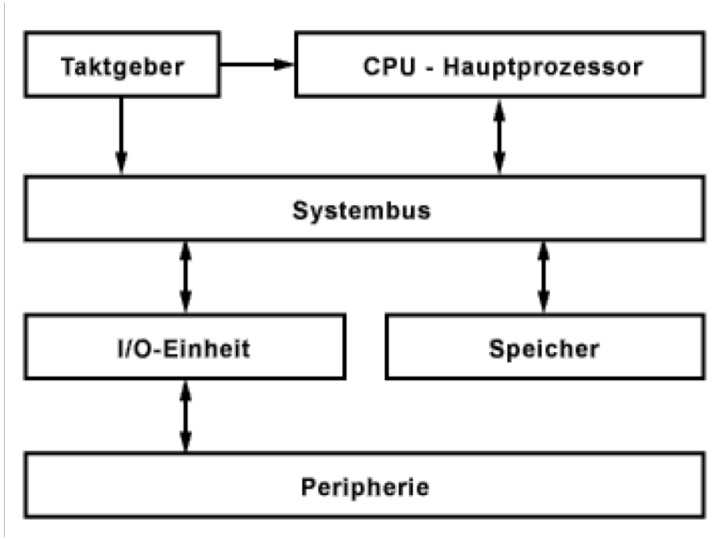
3.2.1 Die von-Neumann-Architektur

John von Neumann (1903 -1957) war einer der wichtigsten Mathematiker des 20. Jahrhunderts und gilt als Pionier der Informatik.

1945 schlug John von Neumann aufbauend auf den Arbeiten von Konrad Zuse ein Konzept für die Funktionsweise eines Rechners vor, das im Wesentlichen bis heute Bestand hat.

3.2.1 Die von-Neumann-Architektur





Rechenwerk	Steuerwerk
Führt arithmetische und logische Operationen aus.	Interpretiert die Befehle eines Programms, führt sie nach dem von-Neumann-Zyklus aus, koordiniert Rechenwerk, Ein- und Ausgabesteuerung und den Arbeitsspeicher.

Ein-/Ausgabewerk	Arbeitsspeicher
Kommuniziert mit den Peripheriegeräten und steuert die Ein-und Ausgabe von Daten	Speichert sowohl Programme als auch Daten.

- Die Programme und Daten werden im Arbeitsspeicher (RAM = Random Access Memory) abgelegt.

- Der Befehl wird nach **von-Neumann Zyklus** abgearbeitet:

FETCH
DECODE
FETCH OPERANDS
EXECUTE

- Programme, Daten, Zwischen- und Endergebnisse werden alle im Arbeitsspeicher abgelegt (sind durch Rechenwerk veränderbar).

- Der Arbeitsspeicher besteht aus einer großen Anzahl von Schaltelementen (**Bits**) mit zwei möglichen Zuständen. Eine festgelegte Anzahl (üblicherweise 8 Bit (= 1 Byte), 16 Bit, 32 Bit oder 64 Bit) davon wird zu adressierbaren Speicherzellen zusammengefasst, welche gezielt angesprochen werden können.

- Ein Programm besteht aus einer Folge von Befehlen; diese werden nacheinander ausgeführt. Sprungbefehle ermöglichen es den sequenziellen Ablauf zu umgehen

- Die Kapazität eines Speichers ergibt sich aus
  Anzahl Speicherzellen * Größe der Speicherzellen in Byte
  Einheiten:
  1 kByte = 2^{10}Byte = 1024 Byte; 1MByte = 1024 kByte; 1Gbyte = 1024 MByte

Vorteil:
Der streng sequentielle Ablauf einer Von-Neumann-Architektur ist der entscheidende Vorteil gegenüber anderen, parallelen Architekturen (z.B. Rechnerverbund). Aus der Sicht des Programmierers ist damit ein einfacher, deteministischer, Programmablauf garantiert.

Nachteile:

- Im Speicher lassen sich Befehle und Daten nicht unterscheiden.

- Jeglicher Datenverkehr von und zu CPU wird über den BUS abgewickelt.
  Der BUS begrenzt dadurch die Geschwindigkeit, da die Transfergeschwindigkeit kleiner als die Verarbeitungsgeschwindigkeit der CPU ist.
  (**Von-Neumann-Flaschenhals**)
  Ein schneller in der CPU integrierter Cache-Speicher kann dieses Problem etwas abschwächen.

Dezimalsystem:

2	5	4	1
10^3er	10^2er	10^1er	10^0er

$(2541)_{10} =$
$2 \cdot 10^3 + 5 \cdot 10^2 + 4 \cdot 10^1 + 2 \cdot 10^0$

Eine n Stellen lange Dezimalzahl kann die ganzen Zahlen von 0 bis $10^n - 1$ darstellen.

Binärsystem:

1	0	0	1
2^3er	2^2er	2^1er	2^0er

$(1001)_2 =$
$1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (9)_{10}$

Eine k Bit lange Binärzahl kann die ganzen Zahlen von 0 bis $2^k - 1$ darstellen.

Umrechnen zwischen den Systemen:

Binär → Dezimal:

- Multiplikation jeder Ziffer mit dem zugehörigen Stellenwert
- Addition der Ergebnisse liefert die entsprechende Dezimalzahl

Beispiel:

$$(1001101)_2 = 64 + 8 + 4 + 1 = 77$$

1	**0**	**0**	**1**	**1**	**0**	**1**
64	32	16	8	4	2	1

Umrechnen zwischen den Systemen:

Dezimal → Binär:

Methode 1:

Schreibe die Zahl als Summe von Zweierpotenzen

Beispiel:

$$73 = 64 + 9 = 64 + 8 + 1 = (1001001)_2$$

1	0	0	1	0	0	1
64	32	16	8	4	2	1

Umrechnen zwischen den Systemen:

Dezimal → Binär:

Methode 2:

Wiederhole bis zum Ergebnis 0:
Ganzzahlige Division durch 2, notiere den Rest.

Die Reste rückwärts gelesen ergeben die Binärzahl

Beispiel:

$$
\begin{aligned}
75 : 2 &= 37 \quad R\,1 \\
37 : 2 &= 18 \quad R\,1 \\
18 : 2 &= 9 \quad R\,0 \\
9 : 2 &= 4 \quad R\,1 \\
4 : 2 &= 2 \quad R\,0 \\
2 : 2 &= 1 \quad R\,0 \\
1 : 2 &= 0 \quad R\,1
\end{aligned}
$$

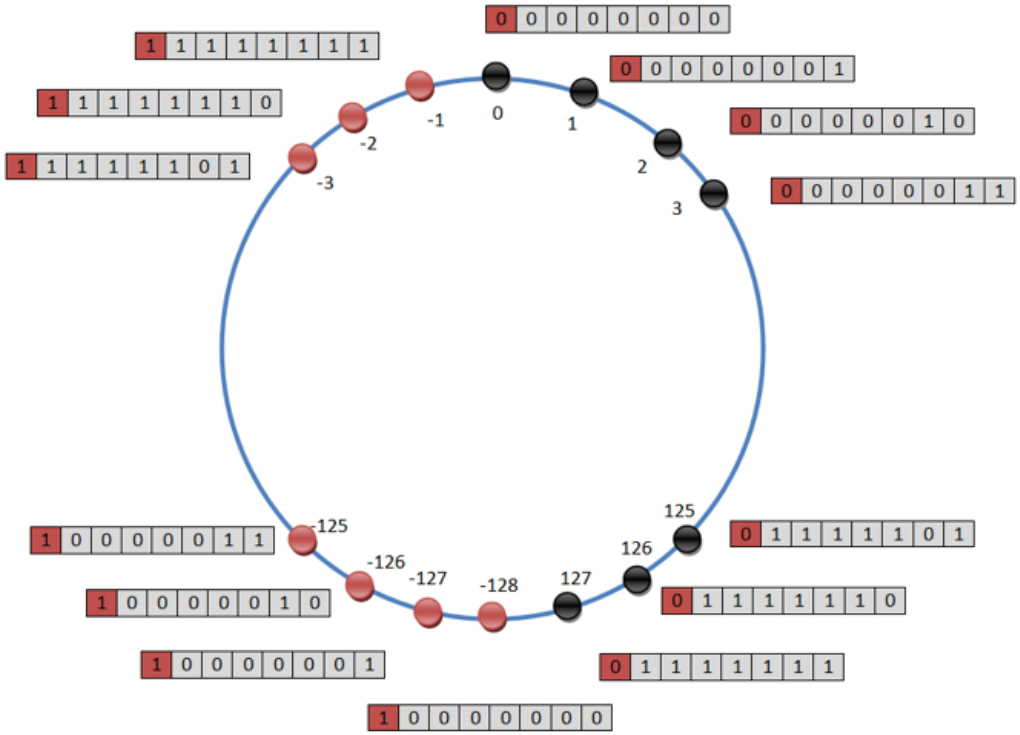
$$73 = (1001011)_2$$

Erklärung:
Im Dezimalsystem erhält man die Ziffern einer Zahl, wenn man durch 10 dividiert:

$$
\begin{aligned}
3017 : 10 &= 301 \quad R\,7 \\
301 : 10 &= 30 \quad R\,1 \\
30 : 10 &= 3 \quad R\,0 \\
3 : 10 &= 0 \quad R\,3
\end{aligned}
$$

Binärdarstellung von ganzen Zahlen:

Mit einer n-Bit Darstellung kann man die 2^n Zahlen $[-2^{n-1}; 2^n - 1]$ darstellen.
Das erste Bit gibt an, ob die Zahl negativ ist. Negative Zahlen haben an der ersten Stelle eine 1.
Darstellung am Zahlenkreis für 8 Bit:

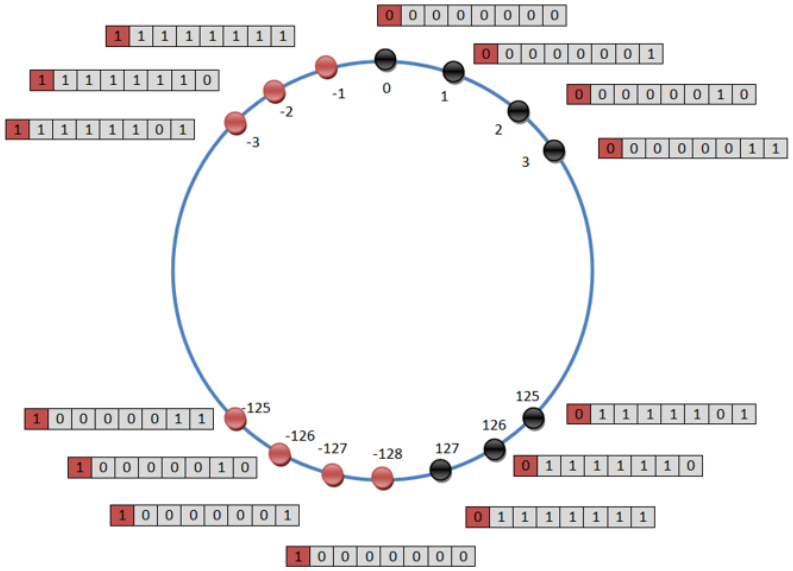


Zur Umrechnung verwendet man
am einfachsten das Einerkomplement.

Dazu kehrt man jedes Bit der Zahl um.

Beispiel:

$$\overline{(01111101)_2} = (10000010)_2 = (-126)_{10}$$



Allgemein gilt:

$$(z)_2 + (\bar{z})_2 + 1 = 0$$

Begründung:

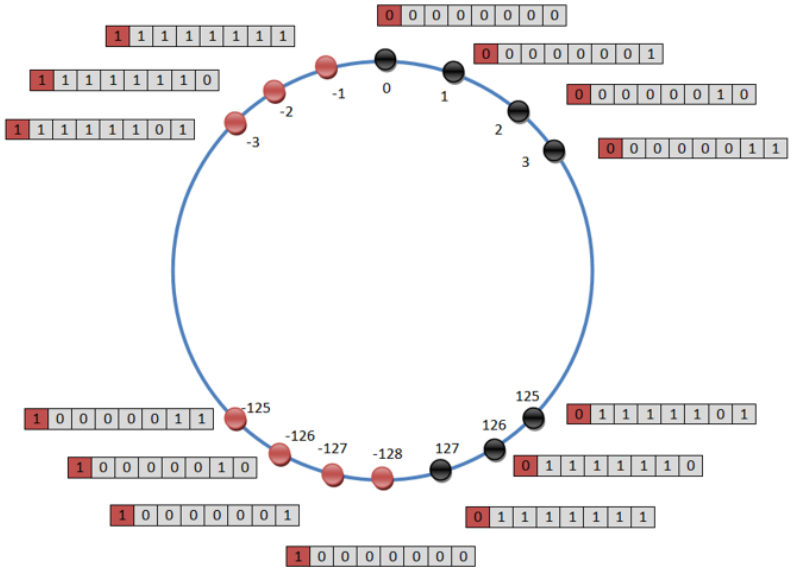
$$(z)_2 + (\bar{z})_2 = (11111111)_2$$

Am Zahlenkreis erkennt man:

$$(11111111)_2 + 1 = 0$$

Umrechnen zwischen den Systemen:

Binär → Dezimal:



Ist das erste Bit eine 0, kann man direkt umrechnen.

Beispiel:

$(01001011)_2 = 64 + 8 + 2 + 1 = 75$

Ist das erste Bit eine 1, verwendet man die Beziehung

$\mathbf{(z)_2 + (\bar{z})_2 + 1 = 0 \quad \Leftrightarrow \quad (z)_2 = -(\bar{z})_2 - 1}$

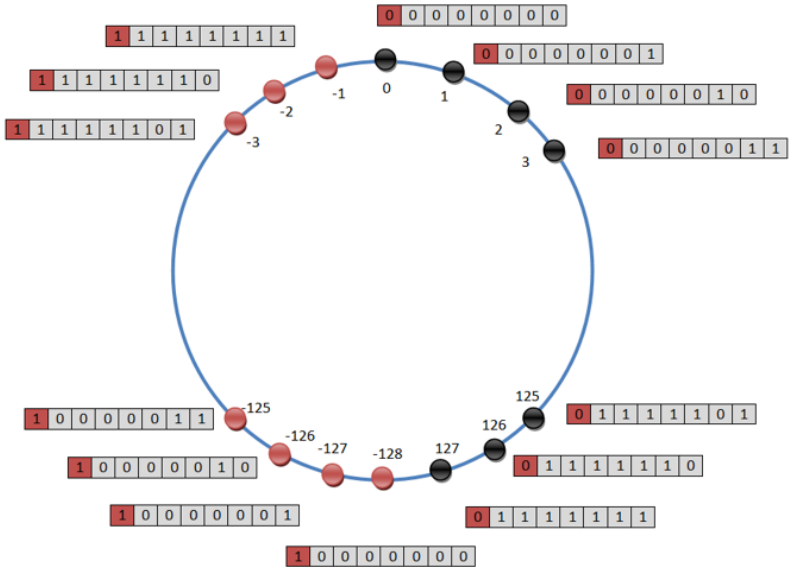
Beispiel:

$(z)_2 = (10000010)_2 = -(\bar{z})_2 - 1 = -(01111101)_2 - 1 =$

$= -(64 + 32 + 16 + 8 + 4 + 1) - 1 = -125 - 1 = -126$

Umrechnen zwischen den Systemen:

Dezimal → Binär:



Ist die Zahl positiv, rechnet man direkt um

Beispiel:

$108 = 64 + 32 + 8 + 4 = (01101100)_2$

Ist die Zahl negativ, verwendet man die Beziehung

$\mathbf{(z)_2 + (\bar{z})_2 + 1 = 0 \quad \Leftrightarrow \quad -(z)_2 = (\bar{z})_2 + 1}$

Beispiel:

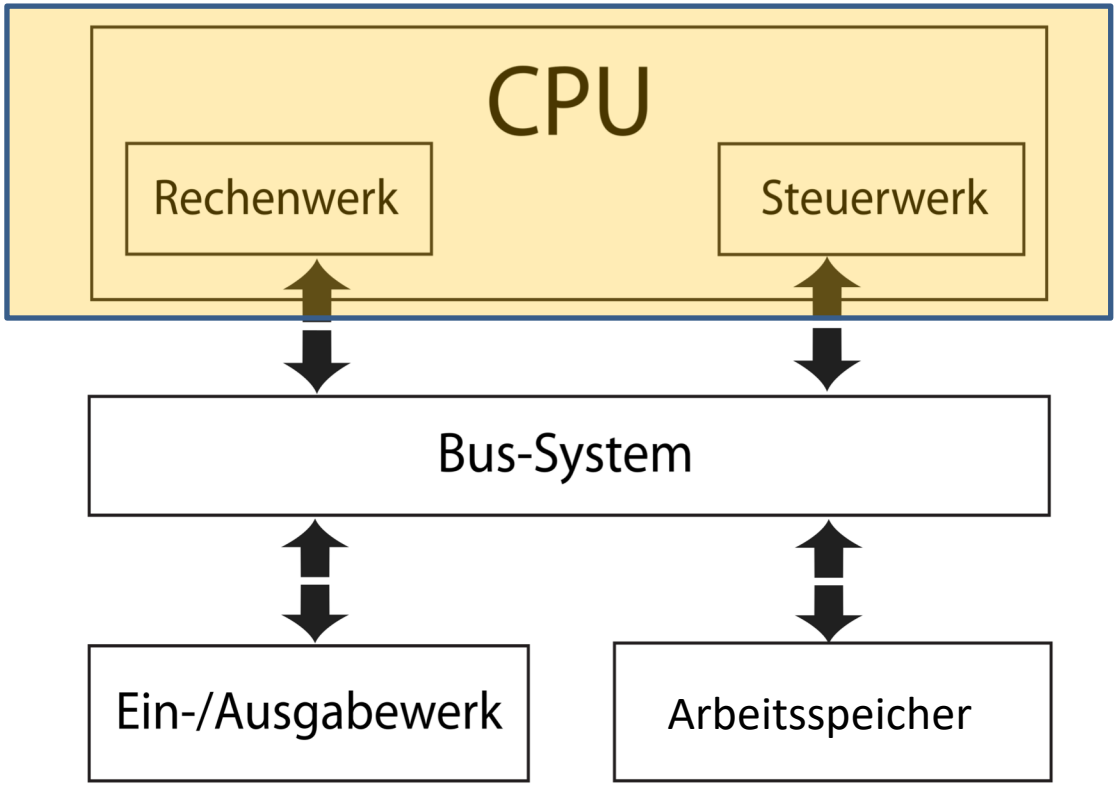
$-108 = ?$

Rechne zunächst 108 in das Binärsystem um:

$108 = (01101100)_2$

Bilde das Einerkomplement und addiere 1:

$\overline{(01101100)_2} + 1 = (10010011)_2 + 1 = (10010100)_2$

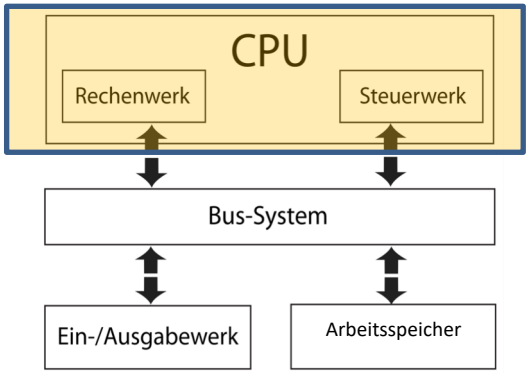


Aufgabe des Prozessors:

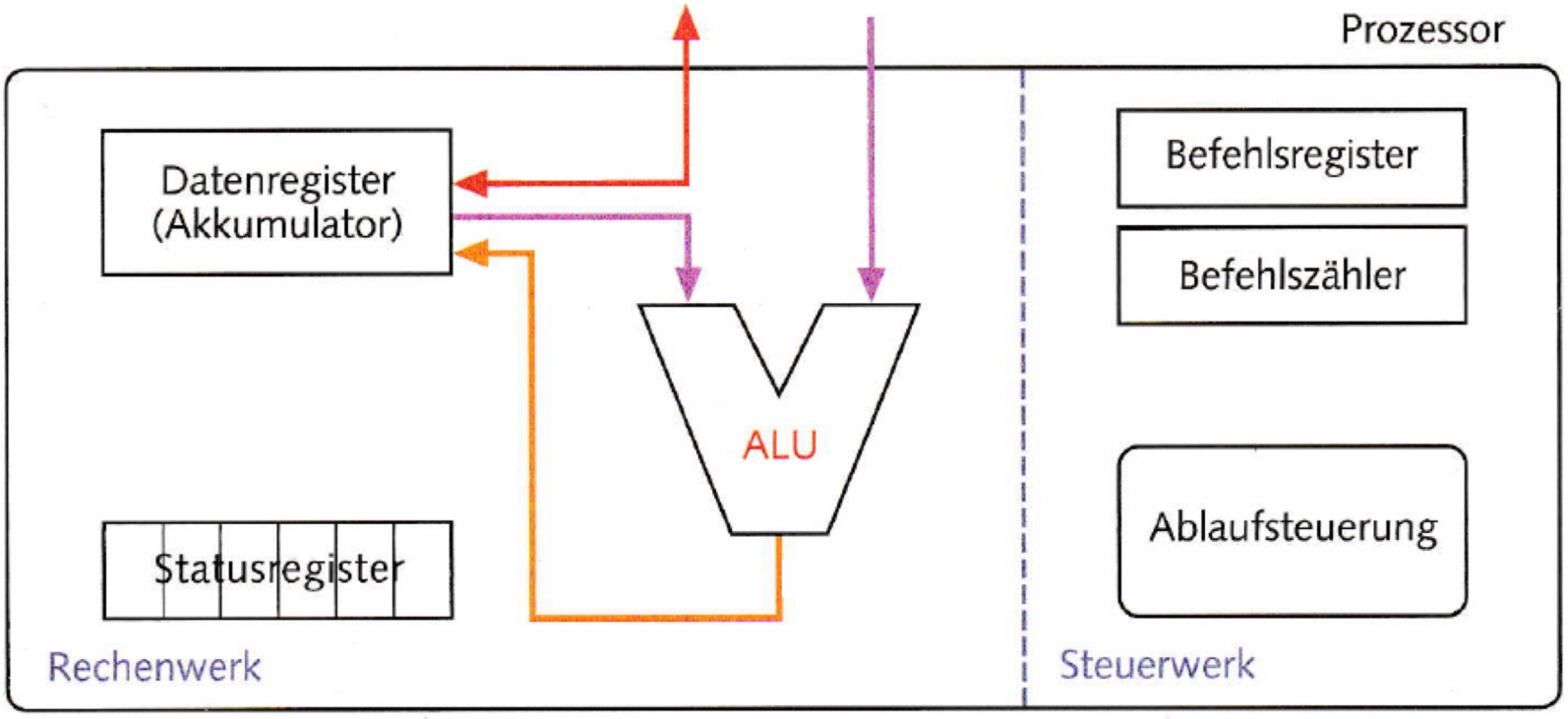
Ausführen eines im RAM abgelegten Programms
(mehrerer aufeinanderfolgender Befehle):

Jeder Befehl ist in einer oder mehreren Speicherzellen (mit jeweils einer Speicheradresse) gespeichert

Also: Durchlaufen der Speicherzellen und Ausführen der jeweiligen Befehle



Detaillierteres Modell eines Prozessors
aus dem Simulationsprogramm Minimaschine
(https://schule.awiedemann.de/minimaschine.html)



Das Steuerwerk - Befehlszähler

- Enthält die Speicheradresse des als nächstes auszuführenden Befehls
- Wird (normalerweise) nach jedem ausgeführten Befehl um 1 erhöht

Das Steuerwerk - Befehlsregister

- Jeder Befehl wird vor der Ausführung vom Arbeitsspeicher (RAM) in den Befehlsregister geladen

- Das Befehlsregister enthält immer den momentan auszuführenden Befehl

Das Steuerwerk - Ablaufsteuerung

- Steuert und bewirkt unmittelbar die Berechnung von Befehlen

Das Rechenwerkwerk - Arithmetisch-logische Einheit (ALU)

- Berechnet unmittelbar die jeweiligen Befehle
- Beispiel möglicher Operationen:
  - Addition
  - Subtraktion
  - UND
  - ODER
  - NICHT
  - …

Das Rechenwerkwerk – Datenregister (Akkumulator)

- Kleiner Datenspeicher im Prozessor
- Schnellerer Zugriff als auf Arbeitsspeicher
- Zwischenspeicher für Werte aus dem Arbeitsspeicher oder Berechnungsergebnisse

Das Rechenwerkwerk - Statusregister

- Enthält sog. „Flags“: Bits, die je nach dem Ergebnis der letzten Berechnung an oder aus sind:
- Beispiele:
  - N-Flag: an, wenn das Ergebnis der letzten Berechnung negativ war
  - Z-Flag: an, wenn das letzte Ergebnis null war
  - V-Flag: an, wenn die letzte Berechnung einen Überlauf verursachte
- Dient zur bequemeren Operation des Prozessors

Der von-Neumnann-Zyklus

Einteilung in vier Phasen:

- Fetch
- Decode
- Fetch Teil 2 (Fetch Operands)
- Execute

Der von-Neumnann-Zyklus - Fetch

- Befehl einlesen, dessen Adresse im Adresszähler gespeichert ist
- Inkrementieren des Befehlszählers um 1

Der von-Neumnann-Zyklus - Decode

- Befehl wird entschlüsselt

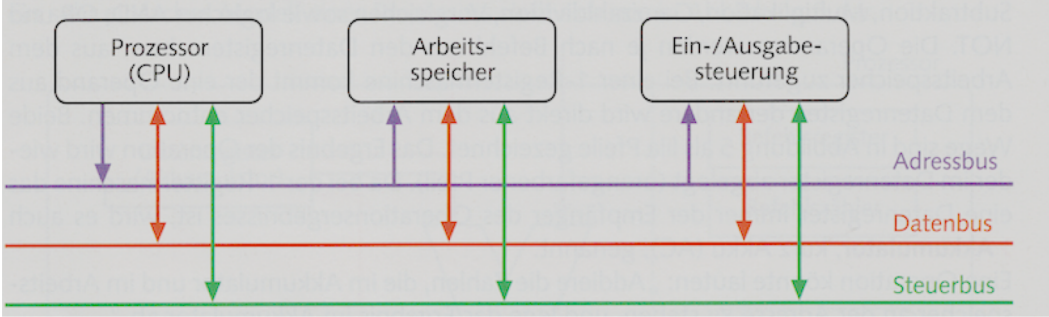
Der von-Neumnann-Zyklus - Fetch Teil 2 (Fetch Operands)

- Ggf. notwendige Operanden aus dem Arbeitsspeicher einlesen
- Inkrementieren des Befehlszählers um die Anzahl der eingelesenen Operanden
  → zum nächsten Befehl

Der von-Neumnann-Zyklus - Execute

- Berechnen des Befehls mit den ggf. eingelesenen Operanden durch die ALU
- Write Back (= Rückschreiben) Speichern des Ergebnisses des Befehls im entsprechenden Register
- Bei Sprüngen im Programm: Umstellen des Befehlszählers auf Zieladresse des Sprungs

3.2.4 Kommunikation zwischen Prozessor und Arbeitsspeicher



Die Kommunikation erfolgt über ein BUS-System.
Ein BUS hat eine feste Anzahl paralleler Leitungen, die jeweils nur die zwei Zustände „ein“ und „aus“ kennen

Adressbus:
Dient zur Übergabe von Adressen. Jede Speicherzelle des Arbeitsspeichers wird adressiert.

Datenbus:
Übernimmt den eigentlichen Transfer. Die Daten werden einzeln, d.h. im Umfang einer Speicherzelle übertragen.

Steuerbus:
Koordiniert die Nutzung von Adress- und Datenbus.

Der Prozessor beherrscht einige einfache Befehle.
Die Menge aller Befehle nennt man **Maschinensprache**. Ein **Maschinenprogramm** ist eine Aneinanderreihung dieser Maschinenbefehle.

Ein Maschinenbefehl ist für den Prozessor ein elementarer Verarbeitungsschritt (z.B. „Lade eine Zahl in den Akkumulator“) und besteht aus zwei Telen:

Operationserkennung:
Bestimmt die Operation, die ausgeführt werden soll

Operandenteil:
Zusätzliche Angaben zur Operation (z.B. Adressen, Werte)

Ein Maschinenbefehl wird wie eine reine Zahl als Bitfolge im Speicher abgelegt.
Die bestimmte Bitfolge legt die Operationserkennung eindeutig fest.
Dem Inhalt einer Speicherzelle ist nicht anzusehen, ob es sich um einen Maschinenbefehl, reine Daten oder eine Adresse handelt.

Die Arbeit des Steuerwerks und damit die Ausführung der Maschinenbefehle erfolgt in einem festen Rhythmus. Der Taktgeber führt dem Prozessor Taktimpulse zu (z.B. 10^9 Impulse pro Sekunde, d.h. Taktfrequenz 1 GHz)

Ein Maschinenbefehl erfordert je nach Komplexität mehrere Taktzyklen

Zu Beginn der Programmausführung wird der Befehlszähler auf die Anfangsadresse des Programms gesetzt.

Das Steuerwerk des Prozessors arbeitet zyklisch und wiederholt den Befehlszyklus bis ein Befehl den Ablauf stoppt.