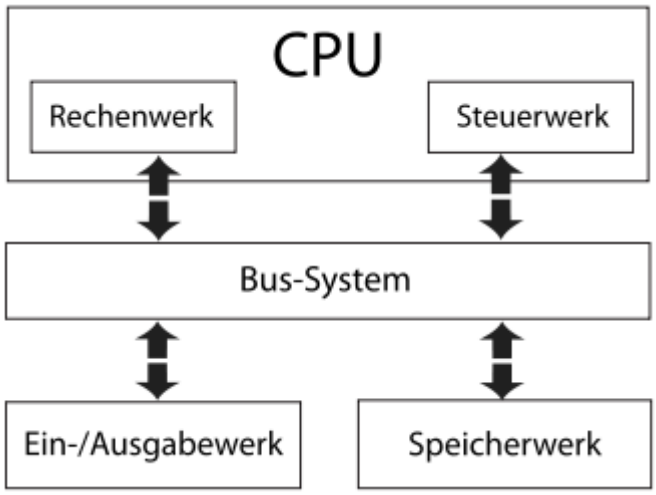
Von Neumann Rechner



Daten und Befehle liegen im gleichen Speicher vor.

John v. Neumann teilte die konzeptionelle Einteilung des Computers in vier Bereiche: das Rechenwerk (mit der ALU=arithmetical logical unit), das Steuerwerk (CU=Control Unit) (mit Befehlszähler (PC=Programm Counter), Befehlsregister und Befehlsdecoder), den Speicher und Ein/Ausgabeeinheiten.
Das Rechenwerk sollte die vier Grundrechenarten beherrschen.

Rechnermodell

Das folgende Rechnermodell entspricht diesem Konzept in wesentlichen Merkmalen .
Es enthält die Register „Akkumulator“ (AC), „Befehlszählregister“ (BZ) und einem von 0 an adressierbaren (Haupt-)Speicher (DS) zur Speicherung von Programmen und Daten.
Falls keine Sprunganweisung zu einer anderen Adresse vorliegt, wird der Inhalt des Befehlszählregisters nach Ausführung einer Anweisung jeweils um 1 erhöht.
Bei Befehlen mit Adresse (wie z. B. LOAD *adr*) stehen der Befehl und die Adresse in einer Speicherzelle des Rechners. Entsprechendes gilt für Befehle mit Konstanten (wie z. B. LOADNUM *num*).
Der Binärcode besteht jeweils aus 4 Bit für die Anweisungsnummer und weiteren 8 Bit für die Adresse *adr* bzw. Nummer *num*, auf die sich der Befehl bezieht, als Binärzahl.

Der Befehlssatz des Modellrechners

Binärcode	**Mnemonic**	**Formale Beschreibung**	**Erläuterungen**
000000000000	READ	AC ← Eingabe	Der Eingabewert wird in den Akkumulator geschrieben.
000100000000	PRINT	Ausgabe ←AC	Der Akkumulatorinhalt wird ausgegeben.
0010*adr*	LOAD *adr*	AC ←DS[*adr*]	Der Inhalt der Speicherzelle *adr* wird in den Akkumulator kopiert.
0011*adr*	LOADI *adr*	AC ←DS[DS[*adr*]]	*load indirect:* Die Speicherzelle *adr* dient als Adressspeicher. Sie enthält die Adresse der Speicherzelle, aus der der Inhalt in den Akkumulator kopiert werden soll.
0100*num*	LOADNUM *num*	AC ←*num*	Die Konstante *num* wird in den Akkumulator geschrieben.
0101*adr*	STORE *adr*	DS[*adr*] ←AC	Der Inhalt des Akkumulators wird nach Speicherzelle *adr* kopiert.
0110*adr*	STOREI *adr*	DS[DS[*adr*]] ←AC	*store indirect:* Die Speicherzelle *adr* dient als Adressspeicher. Sie enthält die Adresse der Speicherzelle, in die der Inhalt des Akkumulators kopiert werden soll.
0111*adr*	ADD *adr*	AC ←AC + DS[*adr*]	Der Inhalt der Speicherzelle *adr* wird zum Akkumulatorinhalt addiert.
1000*adr*	SUB *adr*	AC ←AC – DS[*adr*]	Der Inhalt der Speicherzelle *adr* wird vom Akkumulatorinhalt subtrahiert.
1010*adr*	JUMP *adr*	BZ ←*adr*	Das Programm wird an Speicheradresse *adr* fortgesetzt.
1011*adr*	JUMPE *adr*	Falls AC=0: BZ ←*adr*	Falls der Inhalt des Akkumulators gleich Null ist, wird das Programm an Speicheradresse *adr* fortgesetzt.
1100*adr*	JUMPG *adr*	Falls AC>0: BZ ←*adr*	Falls der Inhalt des Akkumulators größer als Null ist, wird das Programm an Speicheradresse *adr* fortgesetzt.
110100000000	HALT		Die Programmausführung wird beendet.

Beispiel:
Auf dem Eingabeband steht die Zahl 2.

Speicher-Adresse	Anweisung	Erläuterung	BZ	AC	Speicherzelle 1000 0000	Speicherzelle 1000 0001	Ausgabe
00000000	0100 0000 0100	Die Zahl 4 wird in den Akkumulator geladen.	00000001	4			
00000001	0101 1000 0000	Die Zahl 4 wird aus dem Akkumulator in die Speicherzelle 1000 0000 gespeichert.	00000010	4	4		
00000010	0000 0000 0000	Die Zahl vom Eingabeband wird gelesen und im Akkumulator gespeichert.	00000011	2	4		
00000011	0111 1000 0000	Der Inhalt der Speicherzelle 1000 0000 wird zum Akkumulator-inhalt addiert.	00000100	6	4		
00000100	0001 0000 0000	Die Summe wird aus dem Akkumulator ausgegeben.	00000101				6
00000101	1101 0000 0000	Das Programm hält.					

Aufgabe:

a) Interpretiere das folgende Programm, entsprechend anhand der Eingaben 3 und 5. (Zunächst wird die Zahl 3 eingegeben, danach die Zahl 5). Verfolge die Inhalte des Befehlszählregsiters BZ, de Akkumulators AC sowie der Speicherzellen 1000 0000, 1000 0001, 1000 0010, 1000 0011.

Speicher-Adresse	Anweisung
0000 0000	0000 0000 0000
0000 0001	0101 1000 0000
0000 0010	0000 0000 0000
0000 0011	0101 1000 0001
0000 0100	0101 1000 0011
0000 0101	0100 0000 0001
0000 0110	0101 1000 0010
0000 0111	0010 1000 0000
0000 1000	1000 1000 0010
0000 1001	0101 1000 0000
0000 1010	1011 0000 1111
0000 1011	0010 1000 0011
0000 1100	0111 1000 0001
0000 1101	0101 1000 0011
0000 1110	1010 0000 0111
0000 1111	0010 1000 0011
0001 0000	0001 0000 0000
0001 0001	1101 0000 0000

b) Interpretiere das folgende Programm, entsprechend anhand des folgenden Beispiels: Nacheinander werden die Zahlen 3,4,4,3,2,1,3,5 (als Binärzahlen) eingegeben. Nicht belegte Speicherzellen sollen dabei vorher jeweils schon mit dem Wert 0000 0000 initialisiert worden sein.

Speicher-Adresse	Anweisung
0000 0000	0100 0000 0101
0000 0001	0101 1000 0000
0000 0010	0100 0000 0001
0000 0100	0101 1000 0111
0000 0101	0100 1000 1000
0000 0110	0101 1000 0011
0000 0111	0000 0000 0000
0000 1000	0101 1000 0001
0000 1001	1000 1000 0000
0000 1010	1011 0001 0010
0000 1011	0010 1000 0001
0000 1100	0111 1000 0011
0000 1101	0101 1000 0010
0000 1110	0011 1000 0010
0000 1111	0111 1000 0111
0001 0000	0110 1000 0010
0001 0001	1010 0000 0111

0001 0010	1101 0000 0000