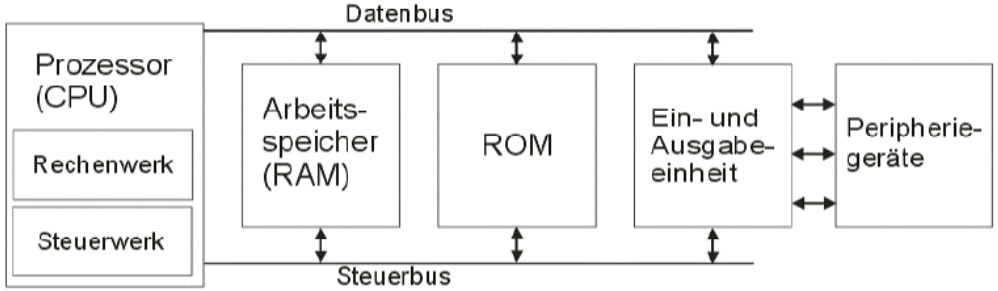
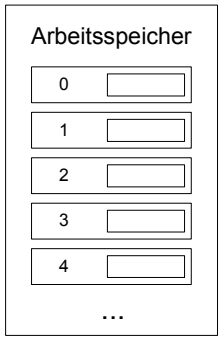
Arbeitsweise eines Computer

Beim letzten Mal wurde der typische Aufbau von Computern nach der Neumann-Architektur



besprochen. Zur Erinnerung ist der Aufbau nochmals abgebildet:
Heute geht es um das Zusammenwirken der einzelnen Komponenten, insbesondere untersuchen wir die Interaktion zwischen Prozessor und Arbeitsspeicher. Für das ROM und die Ein- und Ausgabeeinheit gilt ähnliches wie für den Arbeitsspeicher.

Um die Vorgänge besser verstehen zu können, müssen aber die Bestandteile näher betrachtet werden:



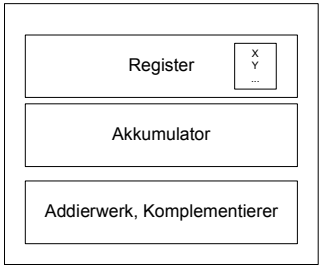
Arbeitsspeicher:

Der Arbeitsspeicher ist durchgängig nummeriert und erlaubt direkten Zugriff auf jede Speicherstelle („wahlfreier Zugriff“).
In ihm stehen Programme **und** Daten. Je nach Programmablauf können damit sogar Daten als Programme interpretiert werden. Die Programme befinden sich in der Regel in der sogenannten *Maschinensprache* im Speicher. Sie sind im Binärcode dargestellt und können vom Prozessor direkt ausgeführt werden. Dabei werden bestimmte Werte von Bytes als Befehl interpretiert, z.B. $A9_{16}$ als LDA interpretiert.

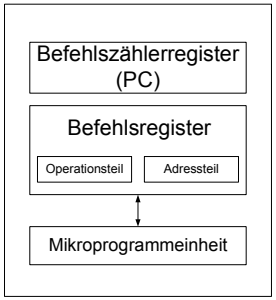
Rechenwerk:

Das Rechenwerk führt die arithmetischen Operationen, wie Addition, Subtraktion, Multiplikation, Division, und logische Operationen, wie z.B. Verneinungen, durch. Deshalb wird es auch arithmetisch-logische Einheit (engl. arithmetic logical unit, ALU) genannt. Es bekommt die Daten (Operanden) vom Steuerwerk zur Verfügung gestellt.
Das Rechenwerk besteht aus:

- dem *Akkumulator*, der immer einen Operand für die Rechnungen enthält und damit bei jeder Operation beteiligt ist.
- *Registern*, die besonders schnelle Speicherstellen sind und z.B. Zwischenergebnisse speichern können. Bei geschickter Nutzung kann man durch sie Programme optimieren.
- *Addierwerk und Komplementierer*, mit dem man Zahlen addieren kann. Alle anderen Rechnungen werden durch Addition und geschickte Zahldarstellung simuliert.
- *Kontroll- und Steuerschaltungen*



Steuerwerk:



Das ist die zentrale Komponente der CPU und besteht aus folgenden Komponenten:

- Das *Befehlsregister* enthält immer den Befehl, der gerade ausgeführt wird. Jeder Befehl besteht in der Regel aus einem Operationsteil und einem Adressteil.
- Der Operationsteil wird an die Mikroprogrammeinheit übergeben, die dann die eigentliche Ausführung (z.B. elektrische Signale) übernimmt.
- Das *Befehlszählregister* (***P**rogramm **C**ounter*) speichert die Adresse des nächsten auszuführenden Befehls. Nachdem ein Befehl ausgeführt ist, wird dieser in der Regel um 1 erhöht.

Mit Hilfe dieser Komponenten ist das Steuerwerk für das Laden der Befehlen aus dem Speicher, Decodieren und Interpretieren der Befehle und für die Steuerung der Bauteile verantwortlich.

Befehlszyklus

1. Phase: Holphase (Ladephase)
Der nächste Befehl, auf den das Befehlszählregister zeigt, wird aus dem Arbeitsspeicher in das Steuerwerk geholt.

2. Phase: Decodierphase
Der aktuelle Befehl wird im Steuerwerk entschlüsselt und interpretiert.

3. Phase: Ausführungsphase
Die Steuersignale zur Ausführung des Befehls wird z.B. durch das Mikroprogramm erzeugt und der Befehl wird schließlich ausgeführt.

Diese drei Phasen werden solange durchlaufen, bis ein END-Befehl auftritt. Die Zeit, die ein Prozessor braucht, um den Zyklus durchzuführen, ist ein Kriterium zur Bewertung seiner Ausführungsgeschwindigkeit. Wichtig ist auch, dass verschiedene Befehle unterschiedliche Ausführungszeiten besitzen.

Einfache Befehle in Maschinensprache:

Zur Verdeutlichung des Zusammenwirkens der einzelnen Komponenten verwenden wir das Programm Wincosi von Dominik N. Weingardt. Die dort verwendeten Maschinensprache ähnelt stark der von einigen Prozessoren in der Realität.

Befehl	**Funktion**
LDA xx	lädt den Wert der Speicherstelle xx in den Akkumulator
STA xx	speichert den Wert des Akkumulators in der Speicherstelle xx
END	beendet ein Programm regulär.
ADD xx	addiert zum Wert, der sich im Akkumulator befindet, den Inhalt der Speicherstelle xx
SUB xx	subtrahiert vom Inhalt des Akkumulators den Inhalt der Speicherstelle xx
MUL xx	multipliziert den Inhalt des Akkumulators mit dem Inhalt der Speicherstelle xx
DIV xx	dividiert den Inhalt des Akkumulators mit dem Inhalt der Speicherstelle xx
JMP xx	springt an die angegebene Adresse xx.

xx steht jeweils für eine Adresse im Arbeitsspeicher. Weitere Befehle werden in der Hilfe von Wincosi ausführlich dargestellt.

