Einheit H2 – Funktionsweise eines Computers
H-AB2.1

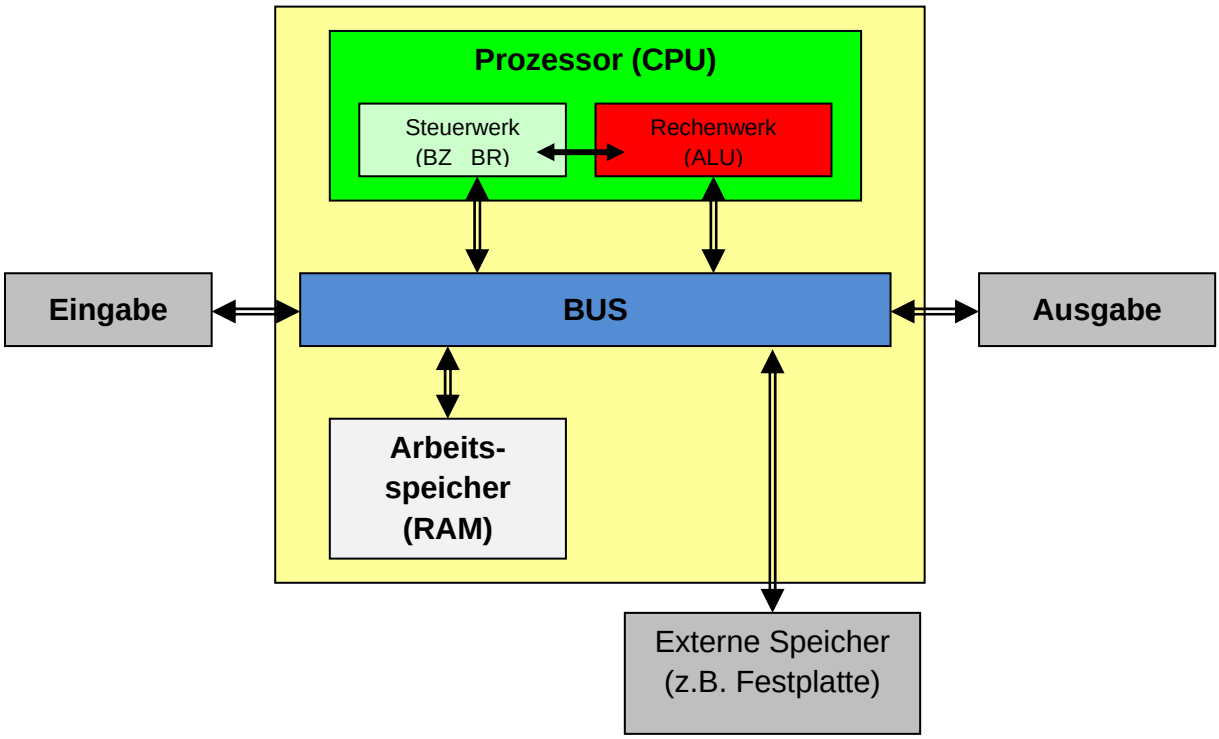
Auf diesem Arbeitsblatt zeigen wir Euch den Aufbau und die Funktionsweise eines Computers.

Die wesentlichste Eigenschaft eines Computers ist, dass er durch Programmierung von einer Universalmaschine zur Lösung beliebiger Berechnungs- oder Informationsverarbeitungsaufgaben zu einer Spezialmaschine gemacht werden kann. Die dafür erforderlichen Prinzipien hat der ungarische Mathematiker John von Neumann in den Vierzigerjahren des 20. Jahrhunderts vorgeschlagen.

Aus diesen Prinzipien ergibt sich eine Architektur, die aus

- Eingabeeinheit,
- Ausgabeeinheit,
- Speicher,
- Steuereinheit und
- Rechenwerk, sowie
- Verbindungsstrukturen zwischen diesen Komponenten (heute üblich, ein Bus)

besteht. Im Speicher befinden sich sowohl die Daten, die der Berechnung zu Grunde liegen als auch das Programm (die entsprechend dargestellte Berechnungsformel), das auf diese Daten angewandt werden soll. Die folgende Abbildung zeigt schematisch die Beziehungen zwischen den Komponenten eines von Neumann-Rechners .



So wie uns Menschen nur verstehen, wenn wir ihre Sprache sprechen, versteht uns auch der Computer letztlich nur, wenn wir seine Maschinensprache sprechen. Wir wollen hier ein Beispiel für eine sehr einfache Maschinensprache, den Befehlsvorrat unseres Computers angeben.

`LADE Speicheradresse`	---	holt den in der angegebenen Speicheradresse gespeicherten Wert in die (in das Eingaberegister der) ALU.
`SPEICHERE Speicheradresse`	---	speichert den von der ALU berechneten im Akkumulator abgelegten Wert in die angegebene Speicheradresse
`CLEAR`	---	setzt den Wert der ALU, insbesondere des Akkumulators, auf 0 (Null).
`ADD`	---	addiert den in der Eingabe der ALU enthaltenen Wert zum Wert des Akkumulators. Das Ergebnis steht im Akkumulator
`MULT`	---	multipliziert den in der Eingabe der ALU enthaltenen Wert mit dem Wert des Akkumulators. Das Ergebnis steht im Akkumulator
`MULT*K Wert`	---	multipliziert den im Akkumulator enthaltenen Wert mit dem im Befehl angegebenen Wert *Wert*. Das Ergebnis wird anschließend wieder in den Akkumulator geschrieben.
`STOP`	---	beendet das Programm. Damit kommt der Rechner tatsächlich allerdings nicht zum Stillstand sondern die Kontrolle wird an das Betriebssystem übertragen.

Aus diesen Befehlen können wir nun ein Programm in Maschinensprache (Assembler) entwickeln. Beginnen wir mit der Formel G = F1 + F2 = l1 * b1 + l2 * b2.

Dazu ist erst wichtig, festzustellen, dass wir in der Formel 4 Eingabevariablen (*l1, b1, l2, b2*) und eine Ausgabevariable *G* haben. Für diese fünf Werte müssen wir Speicherplatz reservieren. Wir nehmen dabei an, dass wir für jede dieser Variablen, wie auch für jeden dann zu schreibenden Befehl ein Speicherwort benötigen. Also reservieren wir

S1	Platz für G
S2	Platz für l1
S3	Platz für b1
S4	Platz für l2
S5	Platz für b2

Die Werte, die in diesen Speicheradressen stehen sollten kannten wir bei der Festlegung der Berechnungsformel noch nicht. Also lassen wir diese Speicherstellen vorerst leer und beginnen nun mit dem eigentlichen Programm.
Zuerst berechnen wir das Ausmaß von Fläche1. Also setzen wir die ALU auf 0, laden wir die Länge in die ALU und addieren sie zum Wert im Akkumulator, laden die Breite in die ALU und multiplizieren mit dem Wert im Akkumulator und speichern das Ergebnis zurück. Wir speichern es in eine jener beiden Zellen, die wir nun nicht mehr benötigen, also s2 oder s3.

S6	CLEAR	setzt ALU auf 0
S7	LADE s2	lies Wert l1 aus Speicheradresse s2
S8	ADD	addiere diesen Wert zum Wert des Akkumulators
S9	LADE s3	lies Wert b1 aus Speicheradresse s3
S10	MULT	multipliziere den zuletzt gelesenen Wert mit dem Wert des Akkumulators
S11	SPEICHERE s3	speichere das Ergebnis in die nicht mehr benötigte Zelle s3

Nun können wir nach eben diesem Schema die Berechnung für die zweite Fläche programmieren. Wichtig ist dabei, nicht zu vergessen, dass die ALU erst wieder auf 0 gesetzt werden muss.

S12	CLEAR	setzt ALU auf 0
S13	LADE s4	lies Wert l2 aus Speicheradresse s4
S14	ADD	addiere diesen Wert zum Wert des Akkumulators
S15	LADE s5	lies Wert b2 aus Speicheradresse s5
S16	MULT	multipliziere den zuletzt gelesenen Wert mit dem Wert des Akkumulators

Nun stellen wir fest, dass wir eigentlich nur mehr die zuerst berechnete Fläche addieren müssen.

S17	LADE s3	lädt das in s3 abgespeicherte Zwischenergebnis, Flächeninhalt von F1, wieder in die ALU
S18	ADD	bildet die Summe der beiden Flächeninhalte
S19	SPEICHERE s1	schreibt das Ergebnis nach s1
S20	STOP	beendet das Programm.

Ihr könnt nun versuchen, an dieser Stelle das Programm für die Berechnung der Fläche eines Dreiecks nach der Formel F = g * h / 2 zu entwerfen und dieses dann mit Partnern so wie das Programm zur Oberflächenberechnung zu simulieren.
Vorsicht, hierbei wird mit einer Konstanten gearbeitet. Ihr benötigt daher den Befehl MULT*K, um mit ½ zu multiplizieren. Natürlich könnte man such einen Divisionsbefehl verwenden. Doch wäre dies im Fall der Division durch 2 nicht besonders geschickt. Warum?

S1 _____

S2 _____

S3 _____

S4 _____

S5 _____

S6 _____

S7 _____

S8 _____

S9 _____

S10 _____

S11 _____