

## Lehreinheit H2 – Hardware, EVA-Prinzip, Von-Neumann-Architektur

### Zeitraumen

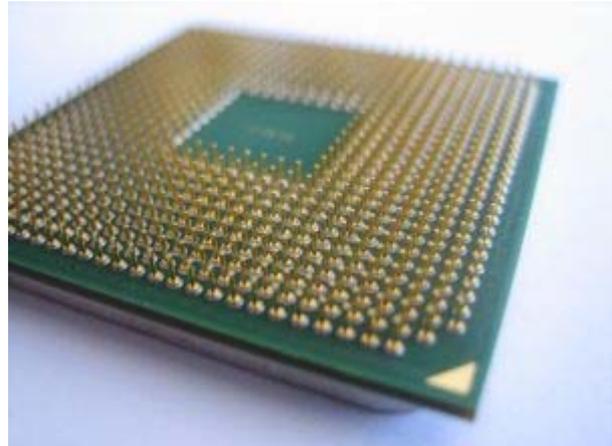
100 Minuten

### Zielgruppe

Sekundarstufe I

### Inhaltliche Voraussetzung

Keine



### Lehrziel

- Funktionsweise der Hardware verstehen, den Computer von „Innen“ kennen lernen.
- Aufbau des Von-Neumann-Rechners, Speicherprogrammierung, einfachen Befehlszyklus verstehen.
- Erkennen, dass im Computer die Arbeit in Einzelschritte unterteilt wird.
- Durch die Animation die Funktionen der einzelnen Hardware-Komponenten kennen lernen, da man ja im Computer selbst die Funktionsweise der Hardware-Komponenten nicht erkennen kann.

Durch einen extrem knapp gehaltenen Befehlsvorrat soll zwar das Prinzip der Speicherprogrammierung verstanden werden. Es wird jedoch, um für Schülerinnen und Schüler der Sekundarstufe I verständlich zu bleiben, auf Ein-/Ausgabe und auf Elemente der Speicherhierarchie verzichtet.

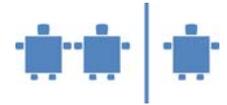
### Motivation

Das Funktionsprinzip eines Computers im Sinne eines Automaten findet man im alltäglichen Leben auch bei anderen „Maschinen“. Schaut man sich die einzelnen Hardware-Komponenten des Computers nur an, kann kaum Verständnis für deren Funktionsweise erzielt werden, da alles zu abstrakt ist. Durch die Animation, das Selbst-Erleben, sowie das anschließende Zerlegen des Computers kann das Verständnis gefördert werden.

Das Prinzip der Speicherprogrammierung erlaubt es, den Computer als universelle Informationsverarbeitungsmaschine anzusehen. Je nachdem, welches Programm geladen wird, wird aus dieser universellen informationsverarbeitenden Maschine eine Spezialmaschine für Textverarbeitung, erstellen von Präsentationen, Video-Schnitt, oder eben auch – wie der Name sagt – für die Ausführung von Rechenoperationen.

### Requisiten

- Kärtchen mit Beschriftungen für Eingabe, Ausgabe, Arbeitsspeicher (RAM), CPU, Steuerwerk, Rechenwerk, Register, E/A-Bus
- Vorlage für die Speicherzellen im Arbeitsspeicher
- Kärtchen mit Befehlen, Rechenaufgaben (vorbereitetes Programm in Pseudo-Assembler)
- Leere Kärtchen oder Zettelchen, auf die (mit wasserlöslichem Folienstift) im Lauf der Simulation entstehende Speicher- und Registerinhalte geschrieben werden können
- Wasserlösliche Folienstifte
- Schreibsachen
- Tafel, Farbkreide
- Alte, aber wenn möglich funktionstüchtige Computer; mind. einen, im Idealfall aber mehrere, dann sehen die Kinder mehr bzw. können stärker selbst aktiv sein.
- Bauteile alter Computer



**Partizipanden**

- 4 - 6 TN für die Animation (bei kleinen Gruppen können einige Aufgaben von einer Person übernommen werden)
- mindestens 2 Beobachter
- gesamte Klasse beim Aufschrauben der Computer

**Vorgehensweise**

1. **Einstieg:** Übungsleiter/in fragt die Teilnehmer:

Was versteht ihr unter dem Begriff „Automat“? Könnt ihr Beispiele für Automaten nennen?

- Getränkeautomat
- DVD-Player
- Radio
- ...
- Computer

Wie könnte man einen Automaten beschreiben, was haben alle Automaten gemeinsam?

Wir haben eine Eingabe und eine Ausgabe und irgendetwas passiert dazwischen. Das sehen wir meistens nicht.

Was entspricht bei den bereits genannten Beispielen der Ein- und Ausgabe?

| <b>Automat</b>  | <b>Eingabe</b>              | <b>Ausgabe</b>        |
|-----------------|-----------------------------|-----------------------|
| Getränkeautomat | Geld, Auswahl-Knopf drücken | Gewähltes Getränk     |
| DVD-Player      | DVD, Knöpfe bedienen        | Film am Bildschirm    |
| Radio           | Sender, Auswahl, Lautstärke | Ton                   |
| Computer        | Befehle, Daten              | ... Berechnungen etc. |

Bleiben wir nun beim Computer. Wodurch kommen Ein- und Ausgaben zustande? Wie heißen die Geräte die ich dafür benötige?

*Die Kinder kennen sicherlich bereits viele dieser Geräte und werden sie auch selbst nennen können. Den Kindern unbekannte Geräte sollten an dieser Stelle auch kurz erklärt werden.*

*Eingabegeräte:* Maus, Tastatur, Joystick, Controller, Scanner, Mikrophon

*Ausgabegeräte:* Bildschirm, Drucker, Plotter, Lautsprecher, Beamer

*Ein- und Ausgabegeräte:* CD-Laufwerk, DVD-Laufwerk, Diskettenlaufwerk, Schnittstelle f. USB-Stick

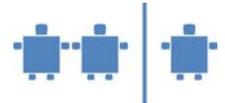
Eingabe und Ausgabe gibt es wohl bei jedem Automat. Wie sieht es aber mit den als Ein-/Ausgabegerät bezeichneten Komponenten aus? Findet man diese auch bei anderen Automaten und wenn ja, wie unterscheiden sie sich von den oben genannten?

Hinweis auf Speicher als spezifisches Element von Computern

Was im Inneren eines Automaten/Computers vorgeht wissen wir aber als Benutzer nicht wirklich. Das soll sich nun ändern.

2. **Vorbereitung zur Gewinnung von Verständnis für Speicherprogrammierung** als Grundlage für das Verständnis eines Computers nach den Von-Neumann-Prinzipien

Das wesentliche Element eines Computers ist, dass er eine „allgemeine Berechnungsmaschine“ ist. „Allgemein“ bedeutet, dass er beliebige Berechnungen durchführen kann. Dies wird dadurch möglich, dass er nicht, wie etwa ein einfacher Taschenrechner, die vier Grundrechnungsarten auf Zahlen ausführen kann, sonst aber nichts, sondern dass man ihm durch ein Programm mitteilen kann, welche



Berechnungen er durchführen soll. Dieses Programm merkt er sich so wie die Daten, auf denen dieses Programm auszuführen ist, in seinem **Arbeitsspeicher**. Dieser gliedert sich somit logisch in einen **Programmspeicher** und einen **Datenspeicher**. Diese Prinzipien, nach denen ein allgemeiner Berechnungsautomat durch Speicherprogrammierung sich wie eine Spezialmaschine verhalten kann, wurden vom ungarischen Mathematiker John von Neuman in den vierziger Jahren des 20. Jahrhunderts in Princeton, USA, vorgeschlagen und anschließend von einer Reihe von Forschern in unterschiedlichen Ländern mit Telefon-Relais, Radoröhren und letztlich mit elektronischen Schaltkreisen realisiert. Wir wollen uns dem Aufbau und der Arbeitsweise einer solchen allgemeinen Informationsverarbeitungsmaschine dadurch nähern, dass wir erst einmal betrachten, wie wir Menschen selbst Informationen verarbeiten oder Berechnungen ausführen.

*Arbeitsanweisungen für die gesamte Klasse vor der Animation:*

a. **Wieviel ist**

$$3 * 7 = \underline{\quad}, \quad 4 * 4 = \underline{\quad}$$

$$35 * 17 = \underline{\quad} ?$$

Wir erkennen an diesem Beispiel, dass wir einfache (elementare) Rechnungen direkt im Kopf ausführen können. Wir haben die jeweiligen Berechnungen so lange auswendig gelernt, bis wir sie „im Schlaf“ ausführen können. Die Aufgabe der Multiplikation mehrstelliger Zahlen können wir nicht mehr auswendig. Aber wir haben ein Verfahren gelernt, mit dem wir eigentlich beliebig große Zahlen multiplizieren können.

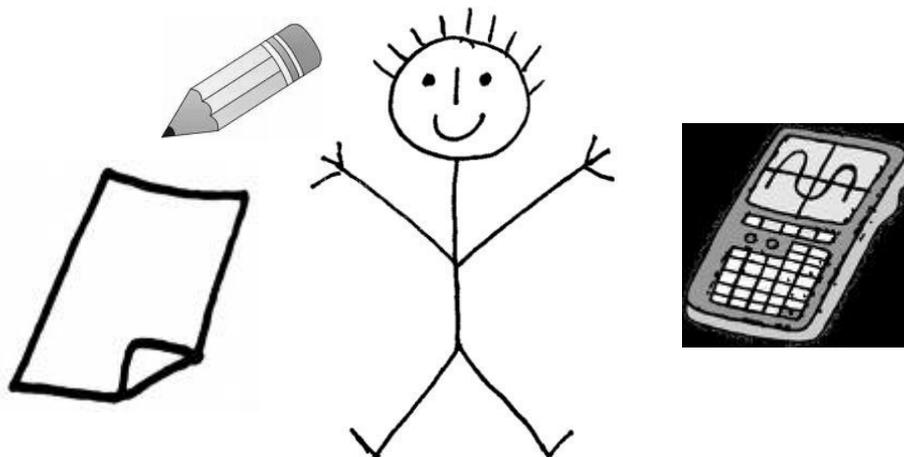
**Wo haben wir die auswendig gelernten Ergebnisse des kleinen Ein-Mal-Eins abgespeichert?**

**Wo haben wir das Verfahren zur Multiplikation großer Zahlen abgespeichert?**

**Womit führen wir dieses Verfahren der Multiplikation großer Zahlen aus?**

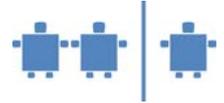
b. **Was sind unsere Rechenutensilien?**

- Rechenaufgaben, Zettel/Block zum schreiben, Taschenrechner, Ablage für die Ergebnisse (**Tafelbild**)



c. Auch der Computer benötigt verschiedene Teilbereiche zum Rechnen.

Der **Block** entspricht dem **Arbeitsspeicher**. Hier werden Notizen gemacht und die Rechnungen können einmal angeschrieben werden. Die eigentlichen Rechnungen führt der Computer wie der Mensch mit einem **Rechner** aus, dieser heißt **CPU** (vom Englischen *Central Processing Unit* – Zentrale



Recheneinheit) und besteht aus dem **Steuerwerk** und dem **Rechenwerk**. Das Steuerwerk ordnet den Ablauf und das Rechenwerk führt die eigentlichen Berechnungen durch. Rechner kann für den Menschen einfach das Gehirn sein, oder auch ein Taschenrechner.

- d. Wie dies funktioniert, wollen wir erst einmal an einem Beispiel überlegen, bei dem wir selbst Berechnungen durchführen wollen:

**Nehmen wir an, wir wollen die Fläche eines Rechtecks berechnen. Wie gehen wir vor?**

Dazu erinnern wir uns an die Formel aus dem Mathematikunterricht

**Fläche = Länge \* Breite** oder

**$F = l * b$  bzw.  $F = a * b$ .**

kennengelernt, wobei  $l$  für Länge und  $b$  für Breite steht.

Mit dieser Formel können wir das Ausmaß rechteckiger Flächen berechnen und, wenn bekannt ist, wie viel Farbe man braucht um einen m<sup>2</sup> zu streichen, dadurch ermitteln, wie viel Farbe wir brauchen um die Fläche anzustreichen oder wie viel wir zahlen müssen, um diese Farbe zu kaufen.

In obige Formel können wir beliebige Werte einsetzen, also etwa

**$l = 2, b = 1$  oder**

**$l = 10, b = 100, \dots$**

**Wodurch unterscheiden sich die drei zuletzt auf der Tafel angeschriebenen Zeilen (blauer Text)?**

Die Kinder sollten erkennen, dass die Formel  **$F = l * b$**  eine Berechnungsvorschrift ist, während die beiden letzten Zeilen Daten sind, die zu dieser Berechnungsvorschrift passen.

Um dies zu verdeutlichen, sollten wir noch eine zweite Rechenaufgabe stellen. Etwa, man solle die Fläche eines rechtwinkligen Dreiecks berechnen. Wieder erinnern wir uns an eine Formel aus dem Mathematikunterricht, etwa

**$F = a * b / 2$ .**

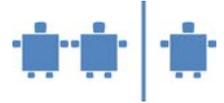
Wieder können wir Werte für  $g$  und  $h$  festsetzen, etwa

**$a = 3, b = 6$  oder**

**$a = 10, b = 100, \dots$**

Wir können nun ein Kind zum Rechtecksflächenrechner, ein anderes Kind zum Dreiecksflächenrechner ernennen und erkennen dadurch:

- Beide Kinder sind grundsätzlich in der Lage beliebige Rechnungen, die sie einmal gelernt haben, auszuführen. Sie könnten prinzipiell wohl auch die Oberfläche oder das Volumen von Quadern, den Umfang von Dreiecken oder von Rechtecken, etc. berechnen. Aber dadurch, dass wir sie mit einer konkreten Aufgabe betraut haben, geben sie uns jetzt eben ein spezielles Ergebnis (Rechtecksfläche, Dreiecksfläche) zurück.
- Damit sie die Rechnungen ausführen können, müssen wir ihnen die Abmessungen eines konkreten Gegenstands mitteilen. Die Formel funktioniert immer. Sie ist von den Werten die wir als Abmessungen bekanntgeben unabhängig.
- Möglicherweise können wir unsere beiden Rechenspezialisten überfordern. Etwa dann, wenn wir extrem große Zahlen angeben, sodass sie sich Zwischenergebnisse nicht mehr merken können. – Das kann in Extremfällen auch bei einem Computer passieren.
- Sowohl in der Oberflächenformel wie in der Dreiecksformel kommt ein „ $b$ “ vor. Doch diese gleiche Benennung von Werten mit unterschiedlicher Bedeutung ist ein „Zufall“. Die beiden „ $b$ “s haben nichts gemeinsam. Wie die einzelnen Eingaben in der Rechnung zu verwenden sind, entscheidet einzig die jeweilige Formel.



- Übertragen wir die Aufgabe nun von unseren beiden Mitschüler/innen auf einen Rechner, würde aus der Formel ein kleines elementares Programm werden.

e. **Übertragung der Rechenaufgabe vom Menschen auf eine Maschine**

Betrachten wir nochmals unsere beiden Rechenspezialisten. **Warum konnten sie ihre jeweilige Aufgabe lösen?**

- Sie konnten uns verstehen, wenn wir ihnen Abmessungen zuriefen.
- Sie konnten Addieren.
- Sie konnten Multiplizieren.
- Sie konnten sprechen (oder an die Tafel schreiben), um uns das Ergebnis ihrer Berechnung mitzuteilen.

Also, sie hatten gewisse Fähigkeiten, die sich einerseits auf die Ein-/Ausgabe bezogen und die andererseits Bezug zu den konkreten Rechenaufgaben hatten. Hätten wir etwa Logarithmieren oder Exponentieren von ihnen verlangt, wären sie gescheitert, weil sie diese Vokabel und die entsprechenden Rechenoperationen noch nicht erlernt haben, diese also noch nicht in ihrem **Befehlsvorrat** vorhanden sind.

Der Befehlsvorrat den wir ausführen können, hängt bei uns Menschen von der gesunden Funktion bestimmter Organe (Ohren, Augen; Nervensystem, Gehirn; Mund, Hände, ...) ab. Beim Computer hängt er dem entsprechend davon ab, aus welchen Bauteilen ein Computer zusammengesetzt ist. Diese, und ihr Zusammenwirken wollen wir nun besprechen.

3. Um den Befehlsvorrat zu verstehen, müssen wir eine Vorstellung über den **Aufbau des Rechners** haben. Er besteht aus:

*An dieser Stelle sollte kein langer Monolog über die einzelnen Einheiten, aus denen ein Computer aufgebaut ist, kommen. Einige der Teile kennen die Kinder ja bereits. Daher scheint es eher sinnvoll, mit der Frage:*

*Aus welchen Teilen setzt sich ein Computer zusammen?*

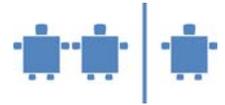
*zu beginnen und Anhand der Rückmeldungen der Kinder mit der in Abb. 1 dargestellten schematischen Zeichnung des von-Neumann-Rechners zu beginnen. Im Entstehen der Zeichnung gibt man, so dies nicht von den Kindern kommt, die nachfolgenden Erklärungen ab. Schließlich ergänzt man die Zeichnung zu dem in Abb. 1 gezeigten Bild.*

*Die nachfolgenden Beschreibungen umfassen sämtliche relevanten und möglicherweise von den Kindern genannten Komponenten. Für die anschließende Animation sind jedoch lediglich die Komponenten Bus, Speicher (RAM); CPU bestehend aus Rechenwerk (ALU) und Steuerwerk (Befehlszähler und Befehlsregister) nötig. So andere Komponenten nicht von den Kindern erwähnt werden, sollten sie derzeit auch nicht besprochen werden. Die Zeichnung reduziert sich entsprechend.*

**Eingabeeinheit:** Sie hat die Aufgabe, Eingaben an den Computer weiterzuleiten, indem sie diese an den E/A-Bus weitergibt. Von dort geht die Eingabe zuerst in den Arbeitsspeicher (RAM).

Während der Ausführung eines Programms sind die Eingaben, die vom Programm zu verarbeitenden Daten. Während der Programmierung ist die Eingabe der Text des Programms, das von einem Editor (einem speziellen Textverarbeitungsprogramm) aufgenommen wird und dann von einem Compiler in Maschinenbefehle übersetzt wird. Dies wollen wir aber hier noch nicht besprechen.

**Ausgabeeinheit:** Sie dient dazu, die vom Computer erarbeiteten Ergebnisse wieder in eine für Menschen lesbare Form zu bringen (Bildschirm, Drucker, ...). Sie erhält die Daten über den E/A-Bus aus dem Arbeitsspeicher.



**Bus:** Das Bussystem eines Rechners ist dafür zuständig, Daten zwischen den einzelnen Einheiten des Computers zu transportieren. Daher auch der an einen Autobus erinnernde Begriff. Tatsächlich handelt es sich um eine gebündelte Datenleitung.

In modernen Rechnern besteht das Bussystem aus mehreren unterschiedlichen Bussen mit unterschiedlichen Leistungsparametern. Der E/A-Bus ist für den Datentransport zwischen externen Geräten (Eingabe, Ausgabe, externe Speicher) und dem Arbeitsspeicher zuständig. Für die Kommunikation zwischen Arbeitsspeicher und den schnellen Registern der CPU sorgt der Daten- bzw. der Befehlsbus. Wir wollen in der Folge der Einfachheit halber allerdings nur von einem gemeinsamen Bus ausgehen.

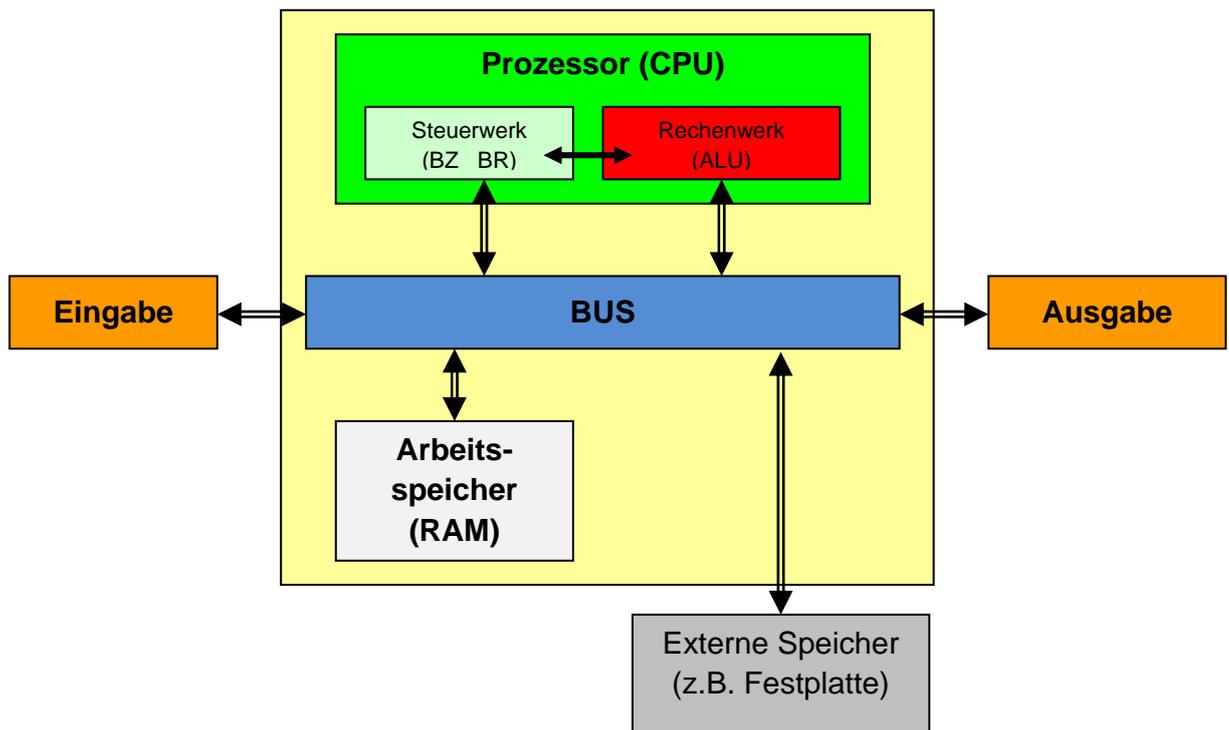


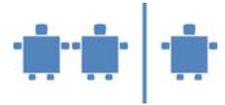
Abb. 1.: *Schema eines von-Neuman Rechners:*

**Speicher** dienen dazu, Daten und Programme während einer Verarbeitung festzuhalten (*Arbeitsspeicher*) oder auch über längere Zeiträume verfügbar zu haben (*Permanentspeicher*). In ihrer Funktion sind alle Speicher gleich. Sie können sich Folgen von Nullen und Einsen „merken“. In ihrer Technologie und daher in ihrer Leistungscharakteristik unterscheiden sie sich jedoch einerseits bezüglich der Zeit, die es dauert einen Wert abzuspeichern oder wieder auszulesen und andererseits auch bezüglich ihrer Flüchtigkeit. Der Inhalt des Arbeitsspeichers und der Register verschwindet, wenn der Strom abgeschaltet wird. Der Inhalt von Magnetspeichern (etwa Festplatte, Diskette, USB-Stick, ...) oder optischen Speichern (CD) bleibt jedoch erhalten, auch wenn kein Strom anliegt.

**Register** sind schnelle Speicher in teurer Technologie gefertigt. Sie können mit der ALU Schritt halten, weil diese in derselben Technologie gefertigt ist.

**Arbeitsspeicher / RAM:** Der Arbeitsspeicher ist jener Bereich, aus dem die CPU sowohl ihr Programm ausliest als auch die von diesem Programm zu verarbeitenden Daten.

RAM steht für Random Access Memory. Dies bedeutet, dass die Zugriffsdauer von der konkreten Adresse, auf die zugegriffen wird, unabhängig ist. Früher waren diese Speicher in Magnetkern-Technologie ausgeführt. Heute sind Arbeitsspeicher Halbleiterspeicher, so wie die Register. Allerdings sind sie in einer langsameren und daher billigeren Technologie ausgeführt.



Im Arbeitsspeicher stehen die von einem Programm zu verarbeitenden Daten an jenen Stellen, an die sie von diesem Programm eingelesen wurden wie auch das Programm selbst.

Fertige Ergebnisse kommen wieder hierher zurück und können hier von der Ausgabe abgeholt werden. Hier werden Daten und Programme gespeichert.

**Extern Speicher / Festplatte:** Die Festplatte kann als Standardbeispiel für externe Permanentspeicher angesehen werden. Informationen, die auf der Platte stehen, bleiben dort erhalten, auch wenn man den Rechner abschaltet. Die Platte ist allerdings kein Direktzugriffsspeicher und die Zugriffsdauer hängt davon ab, wo Information gerade auf der Platte steht und auch davon, auf welche Stelle der Platte zuletzt zugegriffen wurde. Selbstverständlich können auch auf der Platte beliebige Informationen abgespeichert werden. So etwa umfangreiche Datenfolgen, aus denen etwas zu berechnen ist (etwa die Buchhaltung eines Unternehmens) oder – wir haben die Formeln aus dem Mathematikunterricht ja auch vielleicht in einem Formelheftchen stehen – Programme, die wir für unseren Rechner vorbereitet haben, die aber momentan nicht zur Ausführung gelangen.

Doch so wie wir das Formelheftchen nicht rechnen lassen können, sondern erst die Formel lesen und verstehen müssen, damit wir im Kopf dann die zugehörigen Daten einsetzen und die Rechnung ausführen, müssen auch auf der Platte abgespeicherte Programme erst in den Rechner geladen werden, bevor sie ausgeführt werden können. *(Details dazu in den Betriebssystem Einheiten).*

**CPU:** Die *Central Processing Unit* ist der eigentliche Rechner im Computer. Sie besteht aus schnellen Registern, in denen Operanden auf denen die eigentlichen Befehle (Operatoren) ausgeführt werden, abgelegt sind, aus der ALU und aus dem Steuerwerk.

**Rechenwerk / ALU:** Die *Arithmetic Logical Unit* ist der Kern der CPU und das eigentliche Rechenwerk. Hier werden die Befehle des Maschinenprogramms auf die der ALU zugeordneten Register ausgeführt. Also hier wird addiert, subtrahiert, multipliziert, ... Aber es werden eben auch logische Berechnungen (und-sowie oder-Verknüpfungen) durchgeführt.

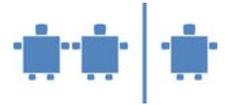
Das **Steuerwerk** besteht im wesentlichen aus einem Befehlsregister und dem Befehlszähler. Es sorgt dafür, dass der jeweils als nächstes auszuführende Befehl aus dem Arbeitsspeicher (RAM) geholt wird und dieser, wenn es sich um einen arithmetischen oder logischen Befehl handelt, von der ALU ausgeführt wird. Wenn es sich um einen Befehl handelt, der den Programmablauf steuert (Verzweigung, Sprung), wird er nur im Steuerwerk ausgeführt.

Der **Befehlszähler** ist ein Register, in dem die Speicheradresse des eben vom Steuerwerk aus dem RAM geholten Befehls steht. Ist dies ein arithmetischer oder logischer Befehl wird der Befehlszähler im Takt der Ausführung dieses Befehls so erhöht, dass er nun die Adresse des nächsten, im Speicher unmittelbar nachfolgenden Befehls enthält. Bei Sprungbefehlen wird er auf den Wert gesetzt, der als Sprungziel im Programm angegeben ist.

4. Damit wir die eben besprochenen Rechenaufgaben einem Computer übertragen können, müssen wir einmal dessen **Befehlsvorrat** kennen lernen.

Wenn wir die in Abb. 1 aufgezeichneten Komponenten des Computers als seine „Organe“ auffassen, wie könnte dann sein Befehlsvorrat sein, wenn wir jedes „Organ“, also jede Komponente ihrer Funktion entsprechend einsetzen wollen?

*Es genügt, hier ein wenig zum Nachdenken anzuregen. Letztlich muss man ja auf den vorbereiteten Befehlsvorrat kommen. Die Frage einzustreuen hilft allerdings, um einen zu langen Monolog zu vermeiden.*



Unterschiedliche Rechner (genau genommen: unterschiedliche CPUs) haben eine unterschiedliche Architektur und daher auch einen unterschiedlichen Befehlsvorrat. Wir nehmen an, er sei sehr einfach und bestünde aus folgenden Befehlen:

|                           |     |  |
|---------------------------|-----|--|
| 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 <i>Wert</i> . 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.                   |

Diese Befehle passen durchaus noch zu den von unseren beiden „RechenexpertInnen“ durchgeführten Operationen, wenn wir *LADE Speicheradresse* durch *LIES vomNotizblock* und *SPEICHERE Speicheradresse* durch *NOTIERE aufNotizblock* ersetzen.

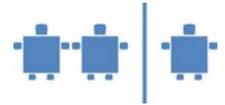
*Wir stellen hier nur einen knappen Befehlsvorrat vor. Insbesondere verzichten wir in dieser Einheit noch auf Vergleichs- oder Sprungbefehle und insbesondere auch auf Ein-/Ausgabebefehle.*

*Es ist somit gesondert zu besprechen, wie das Programm und wie die Daten in den Speicher kommen. Die für die Sekundarstufe II vorgesehene Einheit hat einen umfangreicheren Befehlssatz. Hier gibt es auch ein-/Ausgabe-Operationen.*

- Da wir nun die Komponenten des Rechners kennen gelernt haben, können wir unsere beiden Berechnungsaufgaben so formulieren, dass sie dieser Rechner, den wir in Kürze simulieren werden, verstehen kann.

*Aus Zeitgründen wird man nicht beide sondern nur eine der beiden Berechnungsaufgaben nun üben. Wir empfehlen, dafür Flächenberechnung des Rechtecks zu wählen, weil bei der Dreiecksberechnung möglicherweise eine hier nicht nötige mathematische Hürde auftritt.*

*Um das Prinzip zu erklären, genügt es die Fläche eines beliebigen Rechtecks berechnen zu lassen. Wir empfehlen jedoch, die Summe zweier unterschiedlicher Rechtecksflächen berechnen zu lassen, um öfter als einmal durch den Lese-/Schreibzyklus zwischen Speicher und CPU zu gehen.*



*Gleichviel ob gemeinsam erarbeitet oder bereits vorbereitet, empfiehlt es sich, dieses Programm auf einen Flip-Chart-Bogen (einen Bogen Packpapier) und nicht auf die Tafel zu schreiben. Dadurch kann man das fertige Programm in einem Schritt „in den Speicher laden“ und spart so relativ viel unproduktive Zeit.*

Wir wollen also die Summe von zwei Rechtecksflächen berechnen. Dazu greifen wir auf die schon eingangs erarbeitete Formel

$$F = l * b$$

zurück. Für die Gesamtfläche G haben wir damit

$$G = F1 + F2 = l1 * b1 + l2 * b2.$$

Da wir annehmen, dass die Daten, die wir für unsere Berechnung benötigen, also Länge und Breite von Fläche1 und Fläche2 bereits im Speicher haben, müssen wir erst für diese Daten im Speicher Platz reservieren. Ebenso benötigen wir Platz für die Gesamtfläche G.

Wir benötigen also Speicherplatz für 4 Eingabewerte und einen Ergebniswert

*Man könnte natürlich auch zusätzlichen Speicherplatz für die Zwischenergebnisse F1 und F2 vorsehen. Das würde das zu entwickelnde Programm für einige Kinder vermutlich nachvollziehbarer machen. Allerdings geht dadurch das Prinzip der Mehrfachverwendung von Speicherzellen verloren. Wir wählen daher in der Ausarbeitung des Beispiels die Variante zwar für G Speicher zu reservieren, verwenden aber die „nicht mehr nötigen“ Zellen für die Werte von b1 und b2 für F1 und F2.*

Wir schaffen also Platz für die 4 Eingabevariablen (l1, b1, l2, b2) und für die Ausgabevariable G. Wir nehmen dabei an, dass wir für jede dieser Variablen, wie auch für jeden dann zu schreibenden Befehl ein Speicherwort benötigen. Also reservieren wir dafür in dem für unser Programm vorgesehenen Speicherbereich:

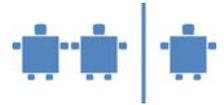
|    |              |
|----|--------------|
| S1 | Platz für G  |
| S2 | Platz für l1 |
| S3 | Platz für b1 |
| S4 | Platz für l2 |
| S5 | Platz für b2 |

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            |

*Mit fortgeschrittenen Klassen kann man diesen Teil des Programms gemeinsam entwickeln und den Rest aus Zeitgründen ergänzen. Wenn die Zeit knapper ist, kann man das gesamte Programm auf einem Flip-Chart oder Packpapier-Bogen vorbereiten.*



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 |

Was steht nun im Akkumulator?

Was würde im Akkumulator stehen, hätten wir vergessen, in Zeile S12 einen CLEAR-Befehl zu geben?

Brauchen wir nun, als S17 weder einen Befehl, der den Flächeninhalt von Fläche2 in den Speicher zurück schreibt? –Warum nicht?

Wir können also feststellen, 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.   |

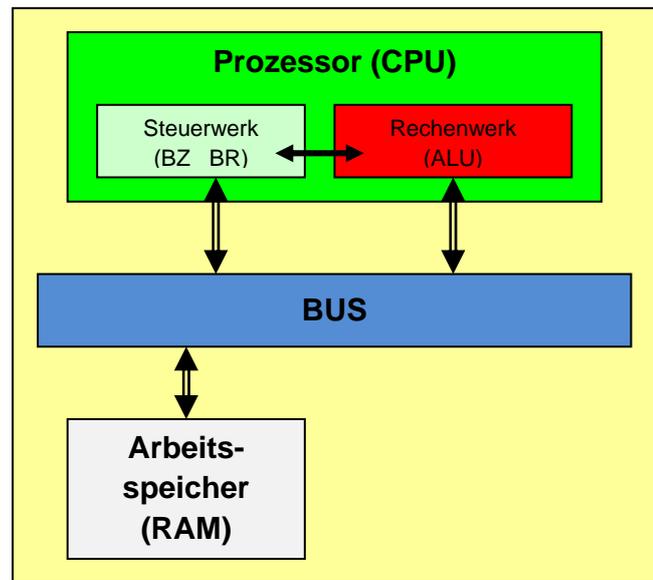
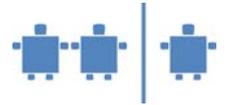
*Hier ist nun zu ergänzen, dass das STOP etwas zu früh kommt, weil man ja eigentlich noch die Ausgabe programmieren müsste, doch dies wollen wir, wie die Programmierung der Eingabe entfallen lassen.*

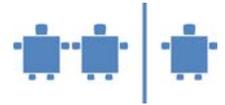
*Den Befehl MULT\*K Wert haben wir hier nicht benötigt. Wenn man an Stelle des Rechteckprogramms ein Dreiecksprogramm geschrieben hätte, hätten wir ihn zur Multiplikation mit  $\frac{1}{2}$  benötigt. Dies ist durch eine SHIFT-Operation aufgrund des binären Zahlensystems einfach realisierbar.*

## 6. Simulation des von Neumann-Rechners

Die Kinder sollen im Folgenden die Aufgabe dieses Rechners übernehmen indem jede Funktionseinheit im Computer durch ein Kind besetzt wird. Das folgende Schema für die Animation zeigt die Aufteilung in die Aufgabenbereiche. Hierfür kann man entweder je Funktionseinheit einen Tisch verwenden, oder man stellt einige Tische zusammen, sodass der hellgelbe Bereich sich auf einem großen Tisch befindet.

*Schema für die Animation:*





Bevor wir versuchen, auch für die Flächenberechnung des Dreiecks ein Programm zu schreiben, wollen wir das Flächenprogramm auf einem von Schülerinnen und Schülern nachgebildeten Computer ausführen lassen. Dazu benötigen wir:

Bus: 1 TN

CPU: 1 TN Steuerwerk oder wenn genügend Personen vorhanden sind:

1 TN Befehlszähler und

1 TN Befehlsregister , sowie

1 TN Rechenwerk (ALU) oder wenn genügend Personen vorhanden sind:

1 TN Eingaberegister der ALU

1 TN Akkumulator

RAM: 1 TN

Die TN, die nicht an der Animation teilnehmen können, bekommen **Beobachterrollen** zugeteilt.

1) Beobachten des Steuerwerks

2) Beobachten des Rechenwerks

3) Globale Beobachtung bzw. Beobachtung der einzelnen „Bauteile“

Jene TN, die eine Beobachterrolle inne haben, haben auch die Rolle der **Benutzer**. Das bedeutet, sie definieren die Werte der Daten, die vom Programm verarbeitet werden sollen.

*Sollte die Gruppe aus nur wenigen Kindern bestehen, kann man problemlos die Betreuung der drei Register einer Person zuordnen. Ebenso kann man Eingabe und Ausgabe einer Person zuordnen. Es sollten aber jedenfalls unterschiedliche Personen für Ein-/Ausgabe, Bus, Steuerwerk, Rechenwerk, Register und RAM, also mindestens 6 Personen vorhanden sein.*

*Bei mehreren Kindern kann man auch noch das Steuerwerk trennen und unterschiedliche Personen mit den Aufgaben Befehlszähler und Befehlsregister beauftragen.*

Die Aufgabenzuweisung erfolgt nun entsprechend den Aufgaben, die die jeweiligen Komponenten auch tatsächlich im Computer haben.

*Die oben angeführten 20 Befehle sind bereits auf Kärtchen vorgeschrieben. Wählt man eine andere Aufgabe und mithin ein anderes Programm, wären entsprechende sequentiell durchnummerierte Kärtchen zu verwenden, so dass auf jedes Kärtchen ein Befehl geschrieben wird.*

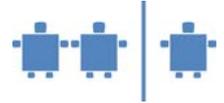
Da für den Computer Code auch nur eine spezielle Form von Daten ist, lesen wir erst einmal das Programm ein. Dies bedeutet:

- Eingabe gibt die Programmzeilen bzw. Programmkärtchen in der richtigen Reihenfolge an dem Bus.
- Der Bus legt sie mit S1 beginnend auf die jeweils nächste freie Speicherstelle, S1 also auf die Speicherstelle s1, S2 auf s2 u.s.w.

*Diesen Schritt wirklich auszuführen, fehlt meist die Zeit. Es genügt eigentlich, darauf hinzuweisen, dass wir das Programm schon entwickelt haben und dieses nun wie es ist, in den Speicher eingelesen wird/wurde.*

- Weiters müssen noch die Eingabedaten, also die Werte für  $I_1$ ,  $b_1$ ,  $I_2$ , und  $b_2$  an die Stellen  $s_2$  bis  $s_5$  des Speichers geschrieben werden. Diese Aufgabe kann man den „allgemeinen Beobachtern“ zuordnen, sie dürfen auch (einfache!) Werte selbst vorschlagen.

Ist dieser Vorgang abgeschlossen, können wir das Programm starten.



- Dazu wird der Befehlszähler-TN (Steuerwerk) einmal auf 0 gesetzt. Er bekommt ein leeres Blatt Papier. Dann muss ihm noch mitgeteilt werden, wo im Speicher der erste ausführbare Befehl liegt (dies mitzuteilen ist noch Aufgabe des Betriebssystems). In unserem Fall ist das Zelle 6. Er notiert also 6 auf seinem Zettel.
- Der Befehlsregister-TN (Steuerwerk-TN) liest (laut !) nun den Inhalt jener Zelle des Arbeitsspeichers, die auf dem Zettel des Befehlszählers notiert ist. (Anfangs also den Inhalt von 5). Er schreibt den Wert auf ein Kärtchen und legt dieses in das Befehlsregister.

*Zur Beschleunigung des Verfahrens empfiehlt es sich, hierfür einen Stapel von Kärtchen zu haben, auf denen jeweils ein Befehl und Unterstrichungen für die zu diesem Befehl gehörenden Parameter eingetragen sind. Der BR-TN muss dann nur das richtige Kärtchen nehmen und die vom Speicher gelesenen Parameter (Adressen) eintragen.*

- Der BR-TN prüft, ob die eben gelesene Operation eine Rechenoperation oder eine Speicheroperation ist. Ist im Befehlsregister (BR) eine arithmetische Operation, gibt sie der BR-TN an das Rechenwerk (ALU) weiter. Ist eine Speicheroperation im BR, wird das Kärtchen an den Bus-TN weitergegeben.
- Bei Datentransfers (Operation LADE oder SPEICHERE) wird das Befehlszettelchen an den Bus-TN gegeben. Dieser schreibt den Wert, der in der angegebenen Abholadresse steht auf ein Kärtchen und legt dieses auf die angegebene Zieladresse (und sagt auch laut, was sie oder er eben ausführte).  
Der ursprünglich in der Zieladresse gespeicherte Wert ist damit nicht mehr ersichtlich. Er wurde durch den neuen Wert überschrieben.
- Wenn der Rechenwerk-TN einen Befehl bekommen hat, führt er diesen auf den angegebenen Registern (also zwischen seinem Eingabe-Register und dem Akkumulator) aus. Dazu fragt es die im Befehl angesprochenen Register nach deren jeweils aktuellen Wert. Diese lesen ihn laut vor.

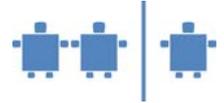
Das Rechenwerk notiert die Berechnung auf einem Kärtchen, führt sie aus, liest das berechnete Ergebnis laut vor. Dieses Ergebnis kommt als neuer Wert auf seinen Kärtchenstapel des Akkumulators. (Man sieht immer nur den jeweils aktuellen Wert).

*Die Beobachter sollen dabei prüfen, ob das Rechenwerk auch tatsächlich korrekt rechnet.*

- Nachdem ein Befehl ausgeführt wurde, ruft die ausführende Komponente (Bus, Rechenwerk) deutlich „fertig“. Auf dieses „fertig“ muss der Befehlszähler-TN warten. Jede Befehlsausführung, gleichviel ob vom Steuerwerk direkt ausgeführt oder vom Rechenwerk ausgeführt, führt zur Erhöhung des Befehlszählers. Der Befehlszähler-TN erhöht also die auf seinem Notizblatt (BZ) notierte Zahl um 1.
- Sobald diese Erhöhung erfolgte, also der Befehl fertig ausgeführt ist, liest der Befehlsregister-TN (Steuerwerk-TN) den laut Befehlszähler nächsten Befehl vom Speicher, notiert ihn wieder auf einem Kärtchen, legt dieses ins Befehlsregister und entscheidet, ob der am Zettelchen stehende Befehl direkt auszuführen ist oder an das Rechenwerk weitergegeben werden soll.

Das oben entwickelte, nunmehr im Speicher abgelegte Programm wird so Schritt für Schritt abgearbeitet, bis letztlich in Schritt 20 das Rechenwerk STOP liest und somit das Verfahren beendet.

*Ein unschöner Aspekt dieses Verfahrens ist, dass das Steuerwerk direkt aus dem Speicher lesen kann. Wir erlauben dies, um das Verfahren zu beschleunigen und die Architektur nicht zu komplex werden zu lassen. Man kann dies aber problematisieren und darauf hinweisen, dass es in heutigen Rechnern nicht einen Bus*



*sondern ein ganzes Bus-System gibt. Der Datenverkehr läuft, wie beschrieben, über den Datenbus, der „Befehlsverkehr“ nach ähnlichen Regeln über den Befehlsbus.*

Nach der Animation wird mit den Beobachtern gemeinsam das Schema der Animation an der **Tafel** festgehalten, also ähnlich der obigen Grafik des Von-Neumann-Rechners. Die beobachtenden Kinder sollten hier die einzelnen Zusammenhänge zwischen den Stationen zu erklären versuchen. Sollten die Kinder das nicht erwähnen: Wichtig ist auch, dass alle Teile zusammenspielen müssen. Ist also der Rechner/die ALU langsam wird alles langsam.

*Am Tafelbild sollte man die Namen der Kinder, die die einzelnen Stationen gespielt haben zu den Hardware-Komponenten dazuschreiben. Somit können die Kinder besser einen Bezug zu den Hardware-Teilen herstellen.*

Der Befehlszyklus soll nochmals gemeinsam besprochen werden:

### **Der Befehlszyklus – Erklärung der einzelnen Schritte:**

1. *Befehl holen*
  - a. Befehl wird aus dem Hauptspeicher (RAM) geholt, im Befehlszähler steht drinnen, welcher Befehl der nächste ist.
  - b. Befehl wird ins Befehlsregister geschrieben.
2. *Befehlszähler erhöhen*
  - a. Befehlszähler wird um die Befehlslänge erhöht (in unserem Fall war dies stets 1)
3. *Befehl decodieren*
  - a. Berechnen, wo die Operanden im RAM (Hauptspeicher) stehen
  - b. Steuersignale für den Befehl/Operator werden berechnet
4. *Operanden holen*
  - a. Die Operanden werden aus dem RAM (Hauptspeicher) geholt und in den Registern gespeichert.
5. *Befehl ausführen*
  - a. ALU (Rechenwerk) berechnet nun das Ergebnis.
  - b. Ergebnis wird wieder ins Register geschrieben.
6. *Ergebnis speichern*
  - a. Ergebnis wird im RAM (Hauptspeicher) abgelegt und steht dort für die Anforderung der Ausgabe bereit.

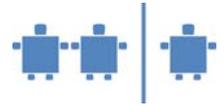
Somit wird das soeben Durchgespielte gefestigt und auch visuelle Lern-Typen werden im Verständnis unterstützt. In dieser Phase sollten die Beobachter vorwiegend tätig sein und beschreiben, wie der „Rechner“ gearbeitet hat. Wenn das nicht möglich ist, sollen die Kinder, die die einzelnen Komponenten gespielt haben, helfen und berichten, was ihre Aufgabe war.

*Aus zeitlichen Gründen wird es innerhalb einer Doppelstunde kaum mehr möglich sein, dass die Kinder auch ein Programm für die Berechnung der Dreiecksfläche – vorzugsweise in Gruppenarbeit – erarbeiten. Es wäre allerdings jedenfalls sinnvoll, ihnen diese Aufgabe zu stellen und die Ergebnisse in einer der nächsten Informatikstunden zu besprechen.*

7. Nun werden die Computer gemeinsam mit den Kindern aufgeschraubt und die wichtigsten Bauteile, sofern erkennbar, benannt.

*Anweisungen und Regeln falls genügend Rechner zur Verfügung sind und die Kinder selbst am Computer herumschrauben:*

- a. Die Kinder werden in Gruppen unterteilt, jede Gruppe erhält einen Rechner.



- b. **Verhaltenregeln** werden erklärt:
- Immer zuerst die Einzelschritte vorzeigen lassen
  - Teile vorsichtig berühren, nicht an den Kontakten
  - Darauf achten, dass nichts auf das Motherboard fällt
  - Zusammenwarten bis alle Gruppen gleich weit sind
  - Während Erklärungen Hände weg vom Computer und aufpassen!

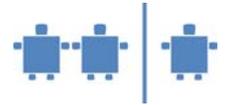
*Beim Zerlegen ist es ganz wichtig, dass ein Bezug zur Animation durch die Kinder hergestellt wird (eventuell Namen der Kinder merken, z.B. wer war das Rechenwerk usw.)*

*Beim Speicher: Unterschiede zwischen magnetischem, optischem und flüchtigem Speicher erwähnen. Wenn möglich sollten den TN auch Teile wie geöffnete Festplatten, ausgebaute Arbeitsspeicher usw. gezeigt werden.*

### **Quellen/Weiterführende Literatur**

Gumm, Heinz-Peter; Sommer, Manfred: Einführung in die Informatik. Oldenbourg Wissenschaftsverlag, München, 2002

Rechenberg Peter: Was ist Informatik? Eine allgemeinverständliche Einführung, 2. Auflage, Hanser Verlag, München Wien, 1994.



## Anhang

Code für die Berechnung der Fläche eines Dreiecks, ausgehend von der Formel  $F = g * h / 2$ .

In diesem Fall haben wir nur 3 Variable,  $F, g, h$ . In der Beispiellösung wird Speicherplatz in dieser Reihenfolge angelegt. Tatsächlich wäre jede beliebige Reihenfolge korrekt.

|    |             |
|----|-------------|
| S1 | Platz für F |
| S2 | Platz für g |
| S3 | Platz für h |

Es folgen die Leseoperationen für  $g$  und  $h$ .

|    |            |  |
|----|------------|--|
| S4 | CLEAR      | setze Akkumulator auf 0  |
| S5 | LADE s2    | lade Wert von g in ALU-Register                                  |
| S6 | ADD        | bring diesen Wert in den Akkumulator                             |
| S7 | LADE s3    | lade Wert von h in das ALU-Register                              |
| S8 | MULT       | multipliziere den zuletzt gelesenen Wert mit Wert im Akkumulator |
| S9 | MULT*K 1/2 | Hier haben wir wieder eine Operation mit einer Konstanten.       |

Eigentlich sollte durch 2 dividiert werden. Doch dies entspricht der viel einfacher durchzuführenden Multiplikation mit  $\frac{1}{2}$  (oder  $2^{-1}$ ). Die tatsächliche Realisierung wird durch ein Verschieben des Registerinhalts nach Rechts, also Right-Shift r3 ausgeführt.

Wir haben somit das gewünschte Ergebnis im Akkumulator stehen. Dieses ist noch in den Speicher zu schreiben.

|     |              |   |
|-----|--------------|---|
| S10 | SPEICHERE s1 | speichert das nunmehr im Akkumulator enthaltene Endergebnis in Zelle s1 |
| S11 | STOP         | beendet dieses Programm   |