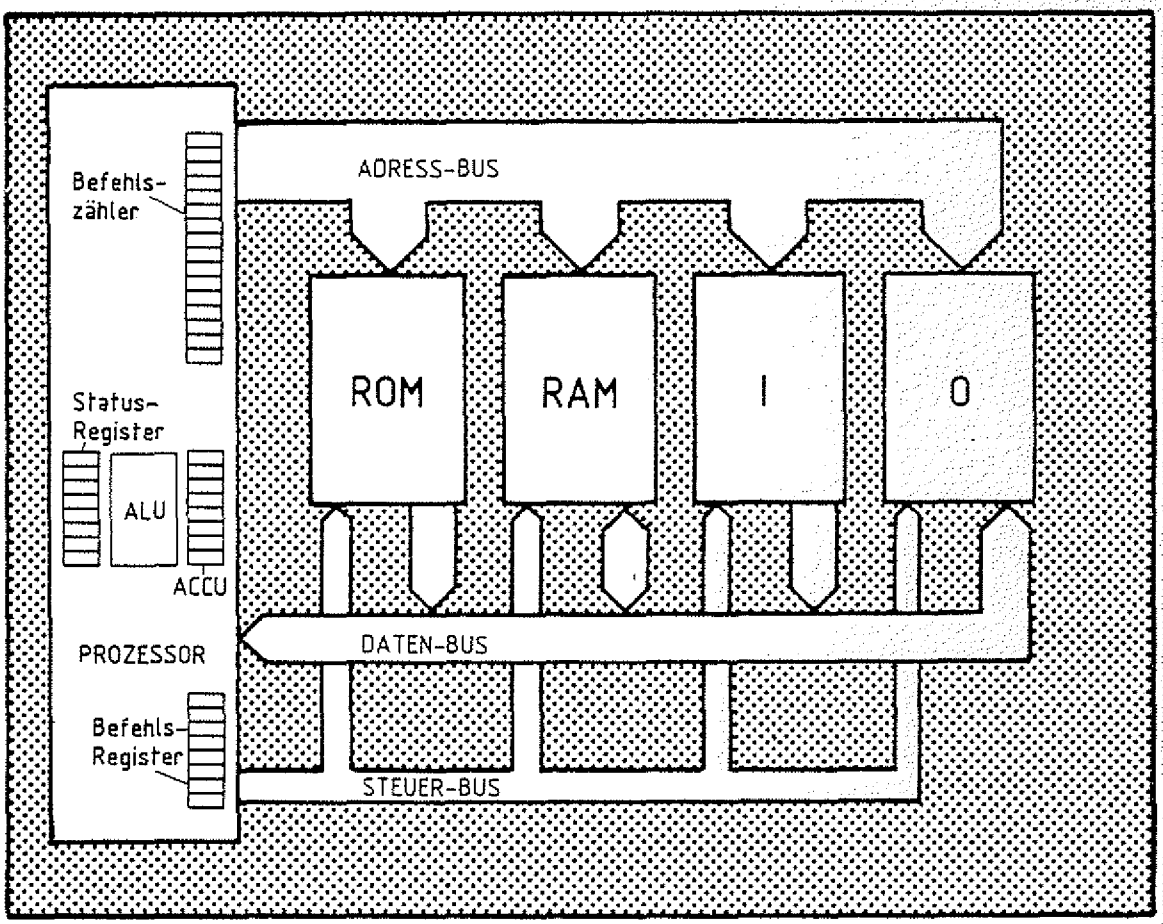
FACHTHEORETISCHE ÜBUNG MIKROCOMPUTER – TECHNIK

MIKROPROZESSOR-MIKROCOMPUTER BFZ/MFA 10.4.



BFZ MFA

Diese Übung ist Bestandteil eines Mediensystems, das im Rahmen eines vom Bundesminister für Bildung und Wissenschaft, vom Bundesminister für Forschung und Technologie sowie der Bundesanstalt für Arbeit geförderten Modellversuches zum Einsatz der "Mikrocomputer-Technik in der Facharbeiterausbildung" vom BFZ-Essen e.V. entwickelt wurde.

Inhaltsverzeichnis

FACHTHEORETISCHE ÜBUNG MIKROCOMPUTER – TECHNIK

MIKROPROZESSOR-MIKROCOMPUTER

BFZ/MFA 10.4.

THEORIETEIL 1

1.1. Einleitung

Das zentrale Element eines Mikrocomputers ist der Mikroprozessor. Seine immer wiederkehrenden Aufgaben sind es ...

Aufgaben des Mikroprozessors

- den Datenverkehr auf den Datenleitungen in Verbindung mit Adreß- und Steuer-Bus zu steuern, um ...
- Bitkombinationen aus dem Speicher in den Prozessor zu transportieren, ...
- die diesen Bitkombinationen entsprechenden Befehle zu ermitteln (Befehlsentschlüsselung) und diese dann auszuführen, ...

Befehlsentschlüsselung

- alle hierzu notwendigen Daten (auch sie sind nur Bitkombinationen) aus dem Speicher zu holen oder Informationen im Speicher abzulegen ...
- sowie den Datenaustausch mit den Ein- und Ausgabe-Einheiten zu steuern.

Die grundsätzliche Arbeitsweise verschiedener Prozessortypen ist gleich. Im internen Aufbau (Prozessor-Architektur) können sie sich allerdings erheblich unterscheiden; es gibt jedoch einige Elemente, die in allen Mikroprozessoren wiederzufinden sind. Die Kenntnis der Bedeutung und Funktion dieser Elemente ist unbedingte Voraussetzung für das Verständnis der Arbeitsweise eines Mikrocomputers.

1.2. Die Elemente des Mikroprozessors (CPU)

Die interne Struktur eines Mikroprozessors ist in Bild 1 dargestellt. Die in jedem Prozessor vorhandenen Elemente sind ...

- Ablaufsteuerung,
- Befehlszähleinrichtung mit Adreßregister,
- Befehlsregister mit Befehlsdecoder,
- Arithmetische und logische Einheit mit Akkumulator und Statusregister.

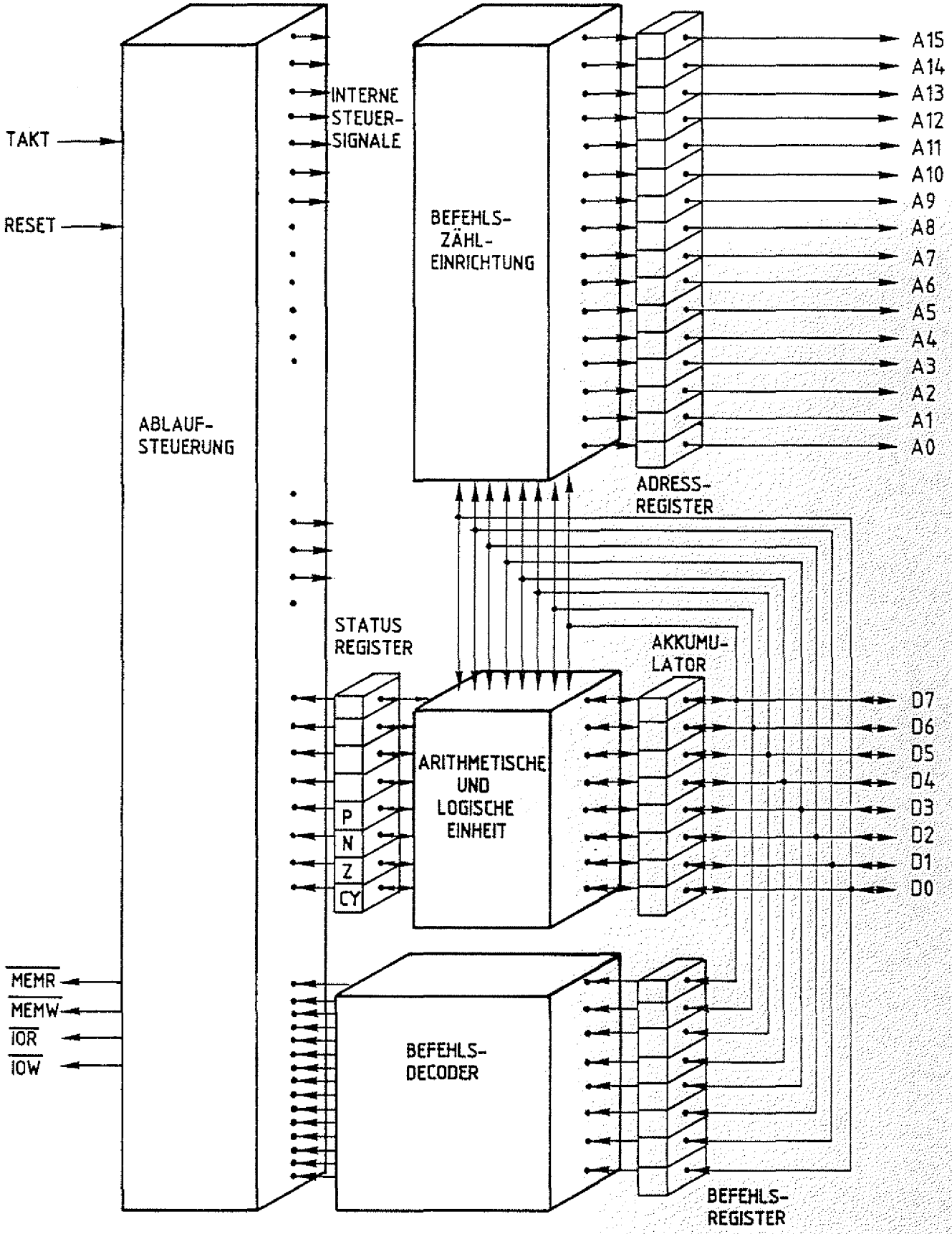


Bild 1: Interne Struktur eines Mikroprozessors

Die ersten drei Elemente gehören zur Control-Unit (CU = Steuerwerk) des Prozessors. Sie sorgen dafür, daß die Bitkombinationen aus dem Speicher geholt werden, und daß die ihnen entsprechenden Befehle erkannt und ausgeführt werden. Zur Arithmetischen und Logischen Einheit (ALU, Rechenwerk) gehört das Status-Register und der Akkumulator. Alle Speicher in einem Prozessor nennt man üblicherweise Register. Der Akkumulator (kurz Akku) ist ein besonderes Register. Es ist eng mit der ALU verknüpft und speichert die Daten, die in der ALU verarbeitet werden sollen bzw. die als Ergebnis einer Rechenoperation in der ALU anfallen. Darauf werden wir im folgenden noch eingehen (siehe 1.2.4.).

ALU
Statusregister
Register
Akkumulator

Ein 8-Bit-Prozessor besitzt einen 8-Bit-breiten Daten-Bus, über den er neben den Daten die auszuführenden Befehle aus dem Speicher holt. Ein durch eine Bitkombination verschlüsselter Befehl an den Prozessor kann daher nur 8-Bit-breit sein. In diesem Zusammenhang spricht man in der Computertechnik auch von Wortbreite oder Wortlänge. Die Wortbreite des in Bild 1 dargestellten Prozessors beträgt 8 Bit oder 1 Byte. Dagegen besitzt z.B. ein 16-Bit-Prozessor eine Wortbreite von 2 Byte. Wortbreite und Daten-Bus-Breite müssen nicht immer übereinstimmen.

Wortbreite

Die Wortbreite eines Prozessors bestimmt die Anzahl der möglichen Bitkombinationen, die er unterscheiden kann. Ein 8-Bit-Prozessor kann max. 256 (2^8) verschiedene Bitkombinationen unterscheiden. Welche Bitkombination welche Reaktion im Prozessor bewirkt, legt der Hersteller des Mikroprozessors fest. Zu jedem Mikroprozessor gehört daher eine Befehlsliste für den Anwender. Im folgenden werden wir einige Befehle aus der Befehlsliste für den Prozessortyp 8085 kennenlernen.

Befehlsliste

Unabhängig von der Art der Befehle vollziehen sich im Prozessor immer wieder die folgenden Schritte:

- Der Prozessor sendet auf den Adreßleitungen die Adresse der Speicherzeile aus, in der der nächste zu verarbeitende Befehl steht.

- Er liest mit dem Steuersignal $\overline{\text{MEMR}}$ diesen Befehl und speichert ihn im Befehlsregister.
- Er stellt fest, um welchen der ihm "bekannten" (vom Hersteller festgelegten) Befehle es sich handelt. Dies nennt man Entschlüsselung des Befehls. Abhängig von dieser Entschlüsselung holt er entweder noch zusätzlich erforderliche Daten für die Befehlsausführung aus dem Speicher oder führt den Befehl sofort aus.
- Wenn der Prozessor den Befehl ausgeführt hat, fährt er mit dem zuerst genannten Schritt fort.

Flußdiagramm

Dieser Vorgang läßt sich übersichtlich in Form eines Fluß-Diagramms darstellen (Bild 2).

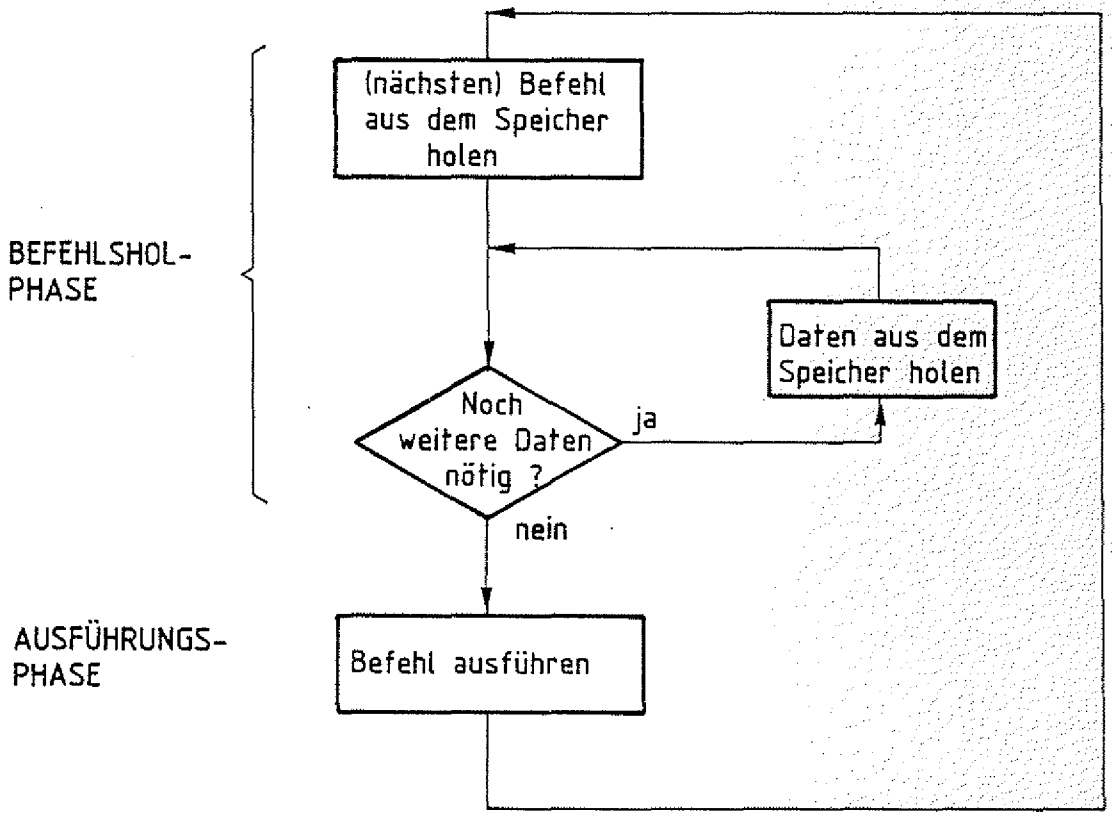


Bild 2: Flußdiagramm zur Befehlsabarbeitung

Befehlshol-Ausführungs-Phase

Jede Befehlsabarbeitung kann daher in zwei Phasen unterteilt werden, die Befehlshol- und die Ausführungs-Phase. Wie die einzelnen Prozessorelemente an diesem Ablauf beteiligt sind, wird im folgenden beschrieben.

1.2.1. Befehlszähleinrichtung und Adreßregister

Im Speicher sind die Befehle hintereinander abgespeichert. Die Befehlszähleinrichtung steuert die Reihenfolge, in der die Befehle abgearbeitet werden. Hierfür besitzt sie intern einen 16-Bit-Zähler, den man Befehls- oder Programmzähler (PC = Programm Counter) nennt.

Programmzähler

Der Zähler wird während jeder Befehlsausführung jeweils um Eins erhöht und liefert so die Adresse der Speicherzeile, deren Inhalt (Befehl oder Daten) verarbeitet wird.

Das Adreßregister wird zu Beginn einer jeden Befehlsholphase mit dem Inhalt des Programmzählers geladen. Während der Befehlsausführung kann bei bestimmten Befehlen der übernommene Progammzählerstand im Adreßregister überschrieben werden. Dies ist beispielsweise der Fall, wenn bei der Befehlsausführung eine Ein- oder Ausgabe-Baugruppe angesprochen wird und die Port-Adresse in das Adreßregister geladen werden muß.

Adreßregister

1.2.2. Befehlsregister und Befehlsdecoder

Das über den Programmzähler und das Adreßregister adressierte Befehlsbyte gelangt während der Befehlsholphase in das Befehlsregister. Im Befehlsdecoder wird ermittelt, welchem Befehl dieses Befehlsbyte entspricht. Das Ergebnis dieser Decodierung oder Entschlüsselung wird der Ablaufsteuerung mitgeteilt.

1.2.3. Ablaufsteuerung

Die Ablaufsteuerung erzeugt alle internen und externen Steuersignale für das Holen und Ausführen der Befehle. Jeder Befehl löst eine bestimmte Folge (Sequenz) von Steuerfunktionen aus, so daß man bei der Ablaufsteuerung auch von einem Steuer-Sequenzer spricht. Die Steuer-Sequenzen aller möglichen Befehle (max. 256) werden in der Ablaufsteuerung in einem sogenannten Mikroprogrammspeicher aufbewahrt. Dieser Mikroprogrammspeicher wird vom Hersteller der CPU programmiert.

Mikroprogramm-Speicher

Mikroprozessoren sind taktgesteuerte Schaltungen.

Taktsteuerung

Zur Erzeugung des Taktsignals besitzen einige Prozessoren einen internen Oszillator, bei anderen muß das Taktsignal extern erzeugt werden. Damit der Prozessor nach dem Anlegen der Betriebsspannung nicht willkürlich mit der Programmabarbeitung beginnt, muß er in den Grundzustand gebracht werden. Dafür besitzen die Prozessoren einen Rücksetz- (RESET-) Eingang. Ein kurzzeitiger Impuls an diesem Eingang bewirkt unter anderem, daß der Programmzähler gelöscht und die Befehlsholphase eingeleitet wird. Nach einem RESET beginnt der Prozessor also immer bei der Speicherstelle 0000H mit der Befehlsabarbeitung.

Taktimpuls-erzeugung

RESET

1.2.4. Arithmetische und Logische Einheit und Akkumulator

Die eigentliche Verarbeitung der Daten erfolgt in der Arithmetischen und Logischen Einheit (Rechenwerk). Man nennt die Daten, die verarbeitet werden, auch Operanden. Die ALU kann einen oder zwei Operanden verarbeiten. Wie und in welcher Form die Verarbeitung erfolgt, d.h. welche Operation in der ALU ausgeführt wird, hängt von dem gerade ausgeführten Befehl ab. Man unterscheidet zwischen arithmetischen und logischen Operationen. Arithmetische Operationen sind z.B. ...

Operand

Operation

arithmetische Operation

— Addition zweier Operanden:

```
  01010011
+ 10000100
----------
  11010111
```

— Subtraktion zweier Operanden:

```
  10011101
- 00101100
----------
  01110001
```

— Inkrementieren eines Operanden: (Um 1 erhöhen)

```
  01110111
+        1
----------
  01111000
```

— Dekrementieren eines Operanden: (eine Eins abziehen)

```
  10001000
-        1
----------
  10000111
```

logische Operation

Dagegen sind logische Operationen z.B. ...

- eine Bit-für-Bit UND-Verknüpfung zweier Operanden:

```
  10011101
  11000111
  --------
  10000101
```

- eine Bit-für-Bit ODER-Verknüpfung zweier Operanden:

```
  00101001
  11100001
  --------
  11101001
```

Für die Zuführung der beiden Operanden besitzt die ALU zwei Eingangskanäle. Der eine Operand gelangt über den Daten-Bus in die ALU, der andere über den Akkumulator. Das Ergebnis einer arithmetischen oder logischen Operation in der ALU wird im Akku abgelegt. Alle Mikroprozessoren besitzen Befehle, die das Laden (engl. load) des Akkus mit Daten aus dem Speicher bzw. das Ablegen des Akku-Inhalts im Speicher (engl. store) bewirken. Außerdem läuft der Datenverkehr von und zu den Ein- und Ausgabe-Einheiten über den Akkumulator ab. Der Akkumulator hat somit eine zentrale Funktion im Mikroprozessor.

Daten von und zu E/A-Einheiten über den Akku

Manchmal soll nach einer arithmetischen oder logischen Operation in der ALU aufgrund eines besonderen Ergebnisses eine Entscheidung getroffen werden (Programmverzweigung). Ein besonderes Ergebnis ist z.B. der Überlauf bei einer Addition, der zustande kommt, wenn die Summe aus den beiden Operanden eine Zahl ergibt, die sich mit acht Bit nicht mehr darstellen läßt.

```
                 11100011        (227)
               + 00100001      + ( 33)
               ----------      -------
Überlauf ---->  1 00000100        (260)  > 255
```

Bei einer Subtraktion kann es vorkommen, daß beide Operanden gleich groß sind und als Ergebnis Null auftritt. Diese und andere Besonderheiten eines Ergebnisses werden nach einer Operation in der ALU im sogenannten STATUS-Register durch Setzen oder Löschen einzelner Bits angezeigt.

Statusregister

Nachfolgend sind Bezeichnung und Bedeutung einiger Status-Bits zusammengefaßt.

Carry-Bit (CY) = 1: Überlauf ist aufgetreten
= 0: kein Überlauf

Zero-Bit (Z) = 1: Ergebnis ist Null
= 0: Ergebnis ist ungleich Null

Negativ-Bit (N)* = 1: Ergebnis ist negativ (Bit7 im Akku ist 1)
= 0: Ergebnis ist positiv (Bit7 im Akku ist 0)

Parity-Bit (P) = 1: Anzahl der Bits mit dem Wert 1 ist geradzahlig
= 0: Anzahl der Bits mit dem Wert 1 ist ungerade

Flags

Die einzelnen Bits, die jeweils einen bestimmten Zustand anzeigen, nennt man auch Flags (Flagge). Der Zustand der Flags wird unmittelbar der Ablaufsteuerung zugeführt, da es Befehle für den Prozessor gibt, die in Abhängigkeit vom Zustand der Flags ausgeführt oder ignoriert werden.

* Wird bei der vorzeichenbehafteten Darstellung von Dualzahlen verwendet.

1.3. Befehle und Befehlsabarbeitung

Befehlsliste

Die Befehle, die ein Mikroprozessor ausführen kann, werden vom Bausteinhersteller in einer Befehlsliste beschrieben. Einige typische Prozessorbefehle und ihre Wirkungen werden im folgenden dargestellt. Dazu nehmen wir das kleine Programmbeispiel zu Hilfe, das Sie in der Übung "Speicher-Einheiten BFZ/MFA 10.3." in den Speicher eingegeben haben.

Speicheradresse (hex.)	Inhalt (hex.)
0000	DB
0001	01
0002	D3
0003	02
0004	C3
0005	00
0006	00

Dieses Programm veranlaßt den Mikroprozessor ...

- den Signalzustand an den Eingängen der Eingabe-Baugruppe mit der Port-Adresse 01 in den Akku zu holen,
- den Akku-Inhalt an die Ausgabe-Baugruppe mit der Port-Adresse 02 zu übergeben,
- die Befehlsabarbeitung an der Speicherstelle 0000 (1. Befehl) fortzusetzen.

Mehr-Byte-Befehle

Das Programm besteht aus drei Befehlen. Zur Speicherung dieser drei Befehle sind jedoch sieben Speicherzeilen erforderlich. Ein vollständiger Befehl besteht also aus dem Befehlsbyte und eventuell erforderlichen Zusatzangaben. Man unterscheidet daher Ein-, Zwei- und Drei-Byte-Befehle. Das obige Programm enthält z.B. zwei Zwei-Byte- und einen Drei-Byte-Befehl.

Die Abarbeitung der einzelnen Befehle vollzieht sich in den folgenden Schritten:

1. Befehl, Befehlsholphase:
   - Nach einem RESET lädt der Prozessor den Inhalt der Speicherzeile 0000H - also den Binärwert von DB - in das Befehlsregister und entschlüsselt den Befehl (Bild 3).
     (DBH ≙ 11011011) Diese Bitkombination veranlaßt den Prozessor, eine Eingabe-Baugruppe zu lesen. Zur Ausführung dieses Befehls benötigt er noch die Port-Adresse der Eingabe-Baugruppe. Vom Hersteller des Prozessors ist festgelegt, daß diese Port-Adresse in der dem Befehlsbyte folgenden Speicherzeile abgelegt sein muß (Der Programmierer muß dafür sorgen, daß sie dort auch zu finden ist).

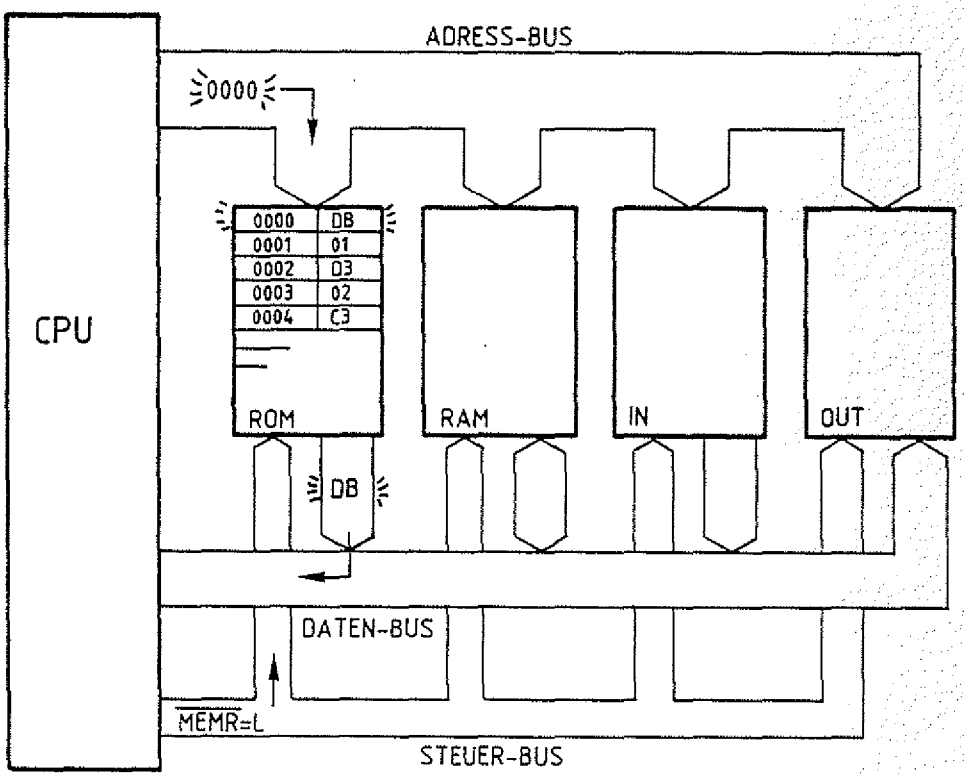


Bild 3: Lesen des Befehlsbytes DB aus dem Speicher

— Zum Lesen des Inhalts der nächsten Speicherzeile erhöht der Prozessor den Befehlszählerstand um Eins, transportiert ihn zum Adreßregister und aktiviert das Steuersignal $\overline{\text{MEMR}}$. Der Speicher übergibt daraufhin den Inhalt der Speicherzeile 0001 an den Akkumulator. Jetzt hat der Prozessor alle Informationen für die Ausführung des 1. Befehls (Bild 4).

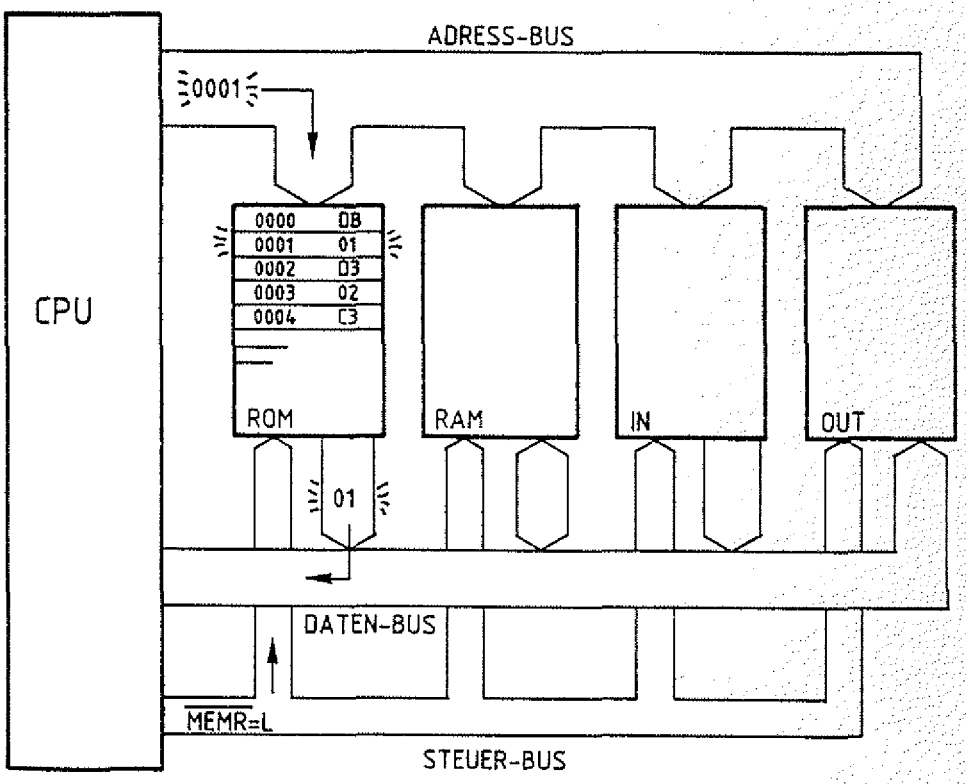


Bild 4: Lesen der Port-Adresse 01 aus dem Speicher

1. Befehl, Ausführungsphase:

   — Für die Befehlsausführung lädt der Prozessor die gerade gelesene Port-Adresse in das Adreßregister und aktiviert das Steuersignal $\overline{IOR}$. Die angewählte Eingabe-Baugruppe schaltet daraufhin den Signalzustand der Eingangsleitungen auf den Daten-Bus, den der Prozessor dann mit dem Wegschalten des Steuersignals in den Akku übernimmt. Der erste Befehl ist abgearbeitet (Bild 5).

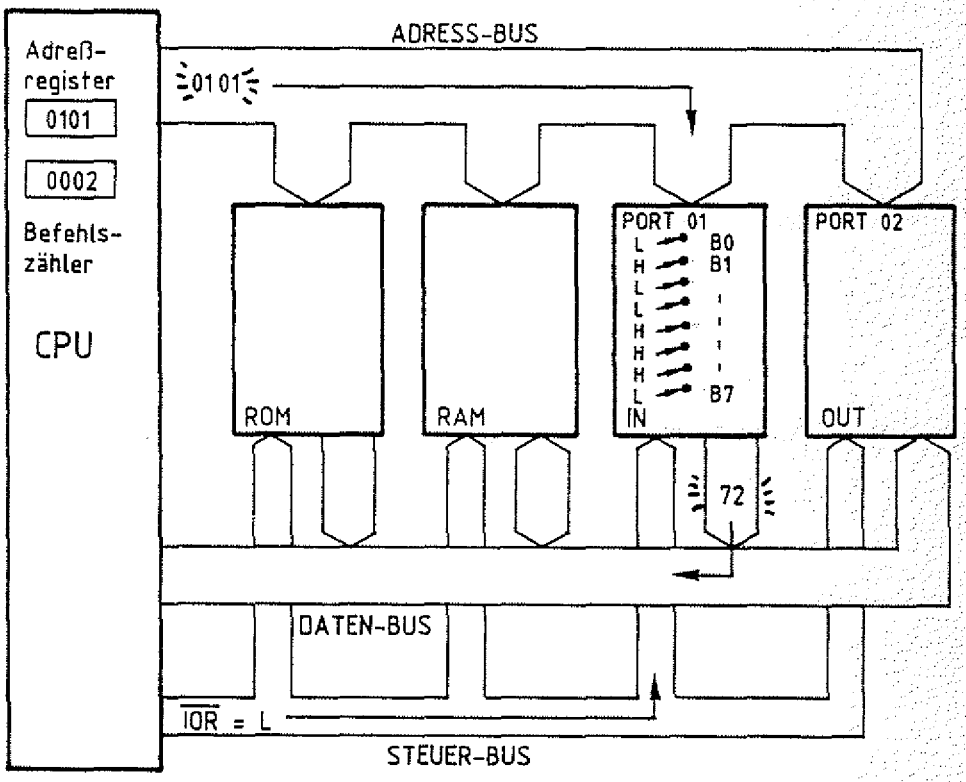


Bild 5: Lesen der Eingabe-Daten von Port 01
(Ausführen des Befehls DB)

Die Befehlszähleinrichtung des Prozessors erhöht während der Befehlsholphase den Befehlszähler jeweils mit dem Lesen einer Speicherstelle, so daß schon während der Befehlsausführung der Befehlszähler die Anfangsadresse des nächsten Befehls enthält. Man sagt auch, der Befehlszähler zeigt auf den nächsten Befehl (Bild 5).

2. Befehl, Befehlsholphase:

— Für das Lesen des nächsten Befehls lädt der Prozessor das Adreßregister mit dem Inhalt des Befehlszählers, aktiviert das Steuersignal $\overline{MEMR}$ und übernimmt das erste Byte des zweiten Befehls in das Befehlsregister. Die Befehlsdecodierung signalisiert der Ablaufsteuerung, daß Daten aus dem Akku an eine Ausgabe-Baugruppe übergeben werden sollen (D3H ≙ 11010011). Auch bei diesem Befehl muß die Port-Adresse der Baugruppe in der dem Befehl folgenden Speicherzeile stehen..

— Der Prozessor liest den Speicherinhalt unter der Adresse 0003H und transportiert ihn für die Befehlsausführung in das Adreßregister.

2. Befehl, Ausführungsphase:

— Für die Ausführung des Befehls schaltet der Prozessor noch den Akku-Inhalt auf den Daten-Bus und aktiviert das Steuersignal $\overline{IOW}$, mit dem die angewählte Ausgabe-Baugruppe die Daten übernimmt. Der zweite Befehl ist abgearbeitet.

Transportbefehle

Die beiden ersten Befehle, die Daten im Computer hin- und hertransportieren, gehören zur Gruppe der Transportbefehle. Der nächste auszuführende Befehl unter der Speicheradresse 0004H ist kein Transportbefehl. Dieser Befehl (C3H ≙ 11000011) veranlaßt den Prozessor, den Befehlszählerstand zu verändern, damit die Befehlsabarbeitung an einer anderen Speicherstelle fortgesetzt wird. Die Speicheradresse, mit der der Befehlszähler geladen werden soll, muß in den zwei, dem Befehlsbyte folgenden Speicherzeilen stehen (Adressen erfordern zwei Byte). Hinter dem Befehlsbyte (C3H) muß der untere Adreßteil (A0 bis A7) und danach der obere (A8 bis A15) abgelegt werden. Den unteren Adreßteil nennt man auch niederwertiges und den oberen höherwertiges Adreßbyte. Dieser Befehl, mit dem Sprünge im Speicher ausgeführt werden können, heißt Sprungbefehl (engl. jump). Er gehört zur Gruppe der Programmsteuerbefehle.

Niederwertiges u. höherwertiges Adreßbyte

Sprungbefehl

Programm-steuerbefehle

3. Befehl, Befehlsholphase:

- Der Prozessor liest die Speicherzeile 0004H und transportiert ihren Inhalt (C3H) in das Befehlsregister.
- Da er für die Ausführung des Sprungbefehls die Sprungadresse benötigt, liest er die nächste Speicherzeile (0005H), in der der niederwertige Adreßteil abgelegt sein muß.
- Für den höherwertigen Adreßteil greift der Prozessor noch einmal auf den Speicher zu und liest den Inhalt der Speicherzeile 0006H.

3. Befehl, Ausführungsphase:

- Für die Befehlsausführung überschreibt der Prozessor den Befehlszählerinhalt mit der gerade gelesenen Adresse. Der Befehl ist abgearbeitet.

Da der Befehlszählerinhalt mit der Adresse 0000H überschrieben wurde, beginnt der Prozessor wieder bei der Speicheradresse 0000H mit der Befehlsholphase. Das Programm wird fortlaufend abgearbeitet und hat zur Folge, daß der Signalzustand an den Eingängen der Eingabe-Baugruppe an den Ausgängen der Ausgabe-Baugruppe eingestellt wird.

1.4. Befehlsarten

Verarbeitungsbefehle

Neben den Transport- und Programmsteuerbefehlen gibt es noch die wichtige Gruppe der Verarbeitungsbefehle, die Operationen in der ALU auslösen. Die drei folgenden Befehlslisten für ...

- Transportbefehle
- Verarbeitungsbefehle
- Programmsteuerbefehle

enthalten nur einen kleinen Teil der beim Prozessor 8085 verfügbaren Befehle. Jede Liste enthält

- den binären und hexadezimalen Befehlscode,
- Angaben zur Anzahl der Bytes des Befehls,
- Angaben zur Wirkung des Befehls und
- den Mnemonischen Code des Befehls (Erklärung folgt).

T R A N S P O R T B E F E H L E				
Befehlsbyte			Bef.	
Mnemo.	hex.	binär	Länge	Wirkung/Beispiel
LDA	3A	00111010	3	Der Inhalt der Speicherzeile, deren Adresse in den beiden dem Befehl folgenden Speicherzeilen steht, wird in den Akku geladen. Beispiel: 3A bewirkt, daß der Inhalt 04 der Speicherzeile 1304H 13 in den Akku geladen wird.
STA	32	00110010	3	Der Akku-Inhalt wird im Speicher abgelegt. Die Adresse der Speicherzeile steht in den beiden dem Befehl folgenden Speicherzeilen. Beispiel: 32 bewirkt, daß der Akku-Inhalt FE unter der Adresse 70FEH abge- 70 speichert wird.
MVI A	3E	00111110	2	Der Akku wird mit dem Inhalt der dem Befehl folgenden Speicherzeile geladen. Beispiel: 3E bewirkt, daß der Akku mit F9 dem Wert F9H geladen wird.
IN	DB	11011011	2	Der Akku wird mit dem Signalzustand der Eingabe-Baugruppe geladen, deren Port-Adresse in der folgenden Speicherzeile steht. Beispiel: DB bewirkt, daß der Signalzustand an 45 den Eingängen der Eingabe-Baugruppe mit der Adr. 45H in den Akku geladen wird.
OUT	D3	11010011	2	Der Akku-Inhalt wird an eine Ausgabe-Baugruppe übergeben. Die Port-Adresse steht in der dem Befehl folgenden Speicherzeile. Beispiel: D3 bewirkt, daß der Akku-Inhalt an die AC Ausgabe-Baugruppe mit der Adr. ACH übergeben wird.

V E R A R B E I T U N G S B E F E H L E				
Befehlsbyte			Bef.	Wirkung/Beispiel
Mnemo.	hex.	binär	Länge	
CMA	2F	00101111	1	Jedes Bit des Akku-Inhaltes wird invertiert (aus 1 wird 0, aus 0 wird 1).
INR	3C	00111100	1	Der Akku-Inhalt wird um 1 erhöht, d.h. inkrementiert.
DCR	3D	00111101	1	Der Akku-Inhalt wird um 1 erniedrigt, d.h. dekrementiert.
ADI	C6	11000110	2	Das dem Befehl folgende Daten-Byte wird zum Akku-Inhalt addiert. Beispiel: C6 bewirkt, daß zum Akku-Inhalt 5BH 5B addiert wird.
SUI	D6	11010110	2	Das dem Befehl folgende Daten-Byte wird vom Akku-Inhalt abgezogen. Beispiel: D6 bewirkt, daß der Akku-Inhalt 3D um 3DH vermindert wird.
ANI	E6	11100110	2	Der Akku-Inhalt wird Bit-für-Bit mit demjenigen Daten-Byte UND-verknüpft, das in der dem Befehl folgenden Speicherzeile steht. Beispiel: E6 wenn vor der Befehlsausführung im 5A Akku 83H steht, so bewirkt der Befehl, daß im Akku der Wert 02H steht.
ORI	F6	11110110	2	Der Akku-Inhalt wird Bit-für-Bit mit demjenigen Daten-Byte ODER-verknüpft, das in der dem Befehl folgenden Speicherzeile steht. Beispiel: F6 wenn vor der Befehlsausführung im 0F Akku 83H steht, so bewirkt der Befehl, daß im Akku der Wert 8FH steht.

P R O G R A M M S T E U E R B E F E H L E				
Befehlsbyte			Bef.	Wirkung/Beispiel
Mnemo.	hex.	binär	Länge	
JMP	C3	11000011	3	Die Programmabarbeitung wird an der Speicherzeile fortgesetzt, deren Adresse in den beiden dem Befehl folgenden Speicherzeilen steht. Beispiel: C3 bewirkt, daß der nächste Befehl 12 von der Speicherzeile F712H F7 gelesen wird.
JZ	CA	11001010	3	Dieser Sprungbefehl zu der Speicheradresse, die in den beiden folgenden Speicherzeilen steht, wird nur ausgeführt, wenn die letzte Rechenoperation Null ergab. Beispiel: CA bewirkt, daß der nächste Befehl 12 nur dann von der Speicherzeile F7 F712H gelesen wird, wenn das Zero-Bit 1 ist. Wenn nicht, wird der Sprungbefehl ignoriert.
JNZ	C2	11000010	3	Dieser Sprungbefehl zu der Speicheradresse, die in den folgenden Speicherzeilen steht, wird nur dann ausgeführt, wenn die letzte Rechenoperation nicht Null (Zero-Bit = 0) ergab.
HLT	76	01110110	1	Dieser Befehl bewirkt, daß die Befehlsabarbeitung gestoppt wird. Nur ein RESET (oder Interrupt) kann die Befehlsabarbeitung wieder einleiten.

1.5. Schreibweise von Programmen

Maschinencode

Der Binär-Code der Befehle (und die hexadezimale Schreibweise) wird auch Maschinencode genannt, weil nur er vom Prozessor (der Maschine) verstanden wird. Diesen Code kann man sich jedoch schlecht merken und man braucht zum Lesen und Schreiben der Programme immer eine Befehlsliste. Daher hat man zu jedem Befehlsbyte eine dem Anwender verständliche Abkürzung (Merk- oder Mnemo-Code) eingeführt, die in den meisten Fällen direkt die Wirkung des jeweiligen Befehls erkennen läßt. Diese Abkürzungen sind in den vorangegangenen Befehlslisten in der Spalte "Mnemo." aufgeführt und heißen Assembler-Code (engl. assemble = zusammensetzen).

Mnemo-Code

Assembler-Code

Die folgenden Beispiele zeigen, wie diese Abkürzungen entstanden sind.

Transportbefehle

LDA = Load ACCU direct,
Lade den Akku direkt

STA = Store ACCU direct,
speichere den Akku direkt

MVI A = Move immediate,
bewege unmittelbar (in den Akku)

IN = Input,
Eingabe

OUT = Output,
Ausgabe

Verarbeitungsbefehle

CMA = Complement ACCU,
komplementiere (invertiere) Akku

INR A = Increment Register,
inkrementiere Register A (Akku)

DCR A = Decrement Register,
dekrementiere Register A (Akku)

ADI = Add immediate to ACCU,
addiere unmittelbar zum Akku

SUI = Subtract immediate from ACCU,
subtrahiere unmittelbar vom Akku

ANI = And immediate with ACCU,
UND unmittelbar mit Akku

ORI = Or immediate with ACCU,
ODER unmittelbar mit Akku

Programmsteuerbefehle

JMP = Jump unconditional,
springe unbedingt (immer)

JZ = Jump on Zero,
springe wenn Null (bedingt)

JNZ = Jump on no Zero
springe wenn nicht Null (bedingt)

HLT = Halt,
anhalten

Programme, die im Hex- oder Binär-Code vorliegen, heißen Maschinenprogramme und solche, die im Assembler-Code vorliegen, nennt man Assemblerprogramme.

Maschinenprogramme
Assemblerprogramme

Häufig schreibt man für die Dokumentation die Bytes eines Befehls in eine Zeile und gibt dabei nur noch die Speicheradresse des Befehlsbytes an. Dadurch werden auch die Maschinenprogramme für den Anwender übersichtlich. In Bild 6 sind die verschiedenen Schreibweisen für Programme gegenübergestellt.

Adresse	Inhalt
0000	DB
0001	01
0002	D3
0003	02
0004	C3
0005	00
0006	00

Maschinenprogramm

Adresse	Befehl
0000	DB 01
0001	D3 02
0004	C3 00 00
0007	..

Maschinenprogramm

Adresse	Befehl
0000	IN 01
0002	OUT 02
0004	JMP 0000
0007	...

Assemblerprogramm

Bild 6: Programmschreibweisen

Programmierer entwickeln ihre Programme zunächst nur im Assembler-Code, weil sie die Kürzel nach einiger Zeit wie die Worte unserer Sprache beherrschen. Man spricht häufig in diesem Zusammenhang auch von der Assemblersprache.

Assembler-Sprache

Nachdem ein Programm fertiggestellt ist, muß es in den Maschinen-Code übersetzt werden, d.h. statt der Kürzel wie IN, OUT, JMP usw. muß der entsprechende Hex-Code (DBH, D3H, C3H, ...) eingesetzt werden. Diese Arbeit nennt man assemblieren, man läßt sie meist von einem Computer ausführen. Das dafür notwendige Programm heißt auch Assembler.

assemblieren