INHALTSVERZEICHNIS

VERSION: 14.9.00

Literatur:

Messmer	PC - Hardwarebuch	Addison-Wesley
Wohak	8086 / 286 Assembler	IWT Verlag
Schief	Einführung in die Mikroprozessoren und Mikrocomputer	Attempto Verlag

Nicht für Anfänger:

Althaus	Das neue PC Profibuch	Sybex
Uphoff	Die Programmierung der EGA/VGA Grafikkarte	Addison-Wesley
Lai	MS-DOS Device-Treiber	Addison-Wesley
Lipp	Der Drucker-Workshop	Systhema

1 Einführung

1.1 Allgemeines

1.1.1 Anwendungsgebiete für Mikrocomputer

Aus den ständig wachsenden Anwendungsgebieten seien einige beispielhafte Bereiche herausgegeriffen:

Nachrichtentechnik:
Fernschreiber, Fernkopierer, elektronisches Fernsprechvermittlungssystem, Bildtelephon, Infrarot-Nachrichtenübertragung, Kommunikation in Breitbandkabelnetzen, Videotext, Bildschirmtext, Verkehrsfunk-Decoder, Rundsichtradar, Funksprechgeräte, Spracherkennung, Sprachwiedergabe.

"Intelligente" Meßgeräte
Oszilloskope, Logik-Analysatoren, Spektrum-Analysatoren, Bauelemente-Tester, Sortierlogiken, rechnende Waagen, Digitalmultimeter, Meßbrücken, "smarte" Meßinstrumente (Meßinstrumente, bei denen keine Fehlbedienung möglich ist), Dosimetriesysteme, Signalsynthesizer.

Physikalische und chemische Geräte:
Überwachung von Reaktoren und Beschleunigern, Pattern-Analyzer, Massenspektrometer, Gaschromatographen, Analyseautomaten, Aufdampfanlagen, Warnanlagen für Umweltschutz, Klimaanlagen.

Medizin, Biologie:
Diagnosegeräte, Röntgengerätesteuerung, Computertomographie, Narkosegeräte, Elektronenmikroskope, Laborauswertungen, Patientenüberwachung, psychologische Testsysteme, Verwaltungscomputer für Praxen.

Industrielle Fertigung:
Fertigungssteuerung und-kontrolle, Roboter, Automaten, CNC-Maschinensteuerung, Regelsysteme, Prozeßsteuerung, Fließbandregelung, Aufzugsteuerung, Pressensteuerung, Testsysteme, für IC's und Leiterplatten, CAD, CAM, (Computer Aided Design, -Manufacturing).

Datenverarbeitung:
Personal-Computer, Workstations, Taschenrechner, Registrierkassen, allgemeine Bürotechnik (Organisation von Karteien, Finanzbuchhaltung, Kopierautomaten, Textverarbeitung usw.), Bankanlagen (automatisches Quittieren von Schecks u.ä.).

Computer - Peripherie:
Bildschirmgeräte (Terminals, Monitore), Floppy-Disk- und Festplatten-Laufwerke, Bandgeräte, Lochkartenleser, Strichcode-Leser, Klarschriftleser, Drucker, Plotter, Scanner, Programmiergeräte.

Kfz - Elektronik:
Motorsteuerung zur Optimierung des Kraftstoffverbrauchs, Antiblockier-Bremssystem, Antikollisionsradar, Bord-Computer, Überwachung aller wichtigen Funktionen, Kfz-Diagnosegeräte

Dezentrale Bordcomputer:
In Flugzeugen, Schiffen, Kraftfahrzeugen, Raumfahrzeugen.

Militärelektronik:
In allen Waffensystemen.

Haushaltselektronik:
Waschmaschinen, Wäschetrockner, Geschirrspüler, Nähmaschinen-Programmierung, Heizungs- und Lichtregelung, Raumüberwachung.

Unterhaltungs - und Freizeitelektronik:
Rundfunk- und Fernsehgeräteelektronik, Plattenspieler-, CD- und Bandgerätesteuerung, Videorecorder, Fernsteuerung, Videospiele, Filmkamera und -projektor, elektronische Brettspiele, Tonsynthese in Orgeln, Heimcomputer, Spielautomaten, Spielzeuge, Uhren.

1.1.2 Gegenüberstellung: Klassische digitale Elektronik Elektronik mit Mikrocomputern

1.1.2.1 Klassische digitale Elektronik: ("fest verdrahtete" digitale Elektronik)

Für jedes Problem der digitalen Elektronik (Messen, Steuern, Regeln) wird eine spezielle logische Schaltung entwickelt: Aus der Problemstellung wird (z.B. über Wahrheitstafeln, Maxterm - Minterm - Methode und Minimierung mit KV - Tafeln) die spezielle Schaltung mit logischen Grundbausteinen (UND, ODER, NICHT, NOR, Flipflops usw.) aufgebaut. Jede Änderung der Problemstellung hat eine neue Schaltung zur Folge.
Beispiel:
Ein Zähler wird durch Flipflops mittels einer KV-Tafel erstellt.
Soll der Zähler anders zählen (z.B. rückwärts), muß ein neuer Zähler erstellt werden.

1.1.2.2 Elektronik mit Mikrocomputern: ("programmierte" digitale Elektronik)

Die bei der Lösung eines Problems zu vollziehenden logischen Schritte und Operationen werden von einem Mikrocomputer durchgeführt. Die Lösung des Problems, dh.die Folge der auszuführenden logischen Schritte - das Programm - ist in einem Speicher niedergelegt; dieser Programmspeicher sagt dem Mikrocomputer, was er tun soll. Eine Änderung der Problemstellung hat "lediglich" eine Änderung des Anwenderprogramms ("Software") zur Folge, das heißt, der Inhalt des Programmspeichers wird ausgetauscht, während der Computer-Aufbau ("Hardware") weitgehend unverändert bleibt.
An die Stelle der individuell verdrahteten Schaltung tritt die individuelle Programmierung des universellen Mikrocomputers. Die Lösung eines digital - elektronischen Problems wird weitgehend auf die Entwicklung eines geeigneten Programms verlagert.

1.2 Darstellung von Information durch Bitmuster

Mikrocomputer sind digital-elektronische Schaltungen, welche Information nur durch zweiwertige Zustandsgrößen ("binar") darstellen. Ein Mikrokomputer ist fähig selbständig Programme (die aus Befehlen bestehen,) abzuarbeiten

Diese beiden Zustände eines Mikrocomputers werden mit 1 und 0 bezeichnet, bzw. bei Realisierung durch Spannungen mit H (High) und L (Low).
Einen solchen zweiwertigen Zustand nennt man

1 Bit (= 1 Binary Digit = 1 Binärziffer)

Gleichzeitig ist 1 Bit die Einheit für binäre Information.
Das heißt zum Beispiel, daß eine "8 Bit Information" aus 8 Stellen mit jeweils 0 oder 1 besteht, etwa 10010110.
Ein Bitmuster ist eine Bitfolge, also aus Nullen und Einsen bestehend.
Beispiel eines Bitmusters: 10101110.
Die Darstellung von Information durch zweiwertige Bitmuster hat den Vorteil, daß diese Bitmuster elektronisch leicht realisiert werden können.

Ist an einer Stelle die 1 vorhanden, sagt man: "Das Bit ist gesetzt".
Ist an einer Stelle die 0 vorhanden, sagt man: "Das Bit ist rückgesetzt".

Größere Einheiten für Informationen sind:

1 Byte = 8 Bit

1 Kbyte = 1024 Byte = (2^{10} Byte)

1 Mbyte = 1024 Kbyte = 1048576 Byte = (2^{20} Byte)

1 Gbyte = 1024 Mbyte = 1048576 Kbyte = 1073741824 Byte
= (2^{30} Byte)

1.2.1 Darstellen von Zahlen durch Bitmuster

Zahlensysteme
Die allgemein eingeführte (bekannte ?) Schreibweise von Zahlen als Folge von Ziffern stellt die Kurzschreibweise für eine Summe dar.
Die einzelnen Summanden ergeben sich aus der jeweiligen Ziffer multipliziert mit dem Stellenwert.
Der Stellenwert ist eine Potenz zur Basis B (B - te Potenz), wobei B eine natürliche Zahl ist und größer oder gleich 2 ist. Für jede Ziffer b_i gilt: $0 \leq b_i < B$
Um Zahlen verschiedener Zahlensysteme zu unterscheiden, kennzeichnet man eine Zahl durch die Basis B im Index. Wenn über die Basis kein Zweifel herrscht, kann ihre Kennzeichnung entfallen.

1.2.1.1 Dezimalsystem

B = 10
Ziffern: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Beispiel:
$1987_{10} = 1 \cdot 10^3 + 9 \cdot 10^2 + 8 \cdot 10^1 + 7 \cdot 10^0$

1.2.1.2 Dualsystem

B = 2
Zifffern: 0, 1
Beispiel:
$110101_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \quad (= 53_{10})$

Bemerkung:
Die Dualzahldarstellung ergibt also ein Bitmuster und ist deshalb sehr geeignet für Computer. Für eine übersichtliche Darstellung von Bitmustern benutzt man das Oktalsystem oder das Hexadezimalsystem.

1.2.1.3 Oktalsystem

B = 8
Zifffern: 0, 1, 2, 3, 4, 5, 6, 7
Beispiel:
$327_8 = 3 \cdot 8^2 + 2 \cdot 8^1 + 7 \cdot 8^0 \quad (= 215_{10} = 11010111_2)$

Bemerkung:
Eine Dualzahl läßt sich ohne viel zu rechnen als Oktalzahl schreiben, indem man jeweils 3 Bit von rechts her zu einer Gruppe zusammenfaßt und jeweils als Oktalzahl schreibt.
Eventuell ist dazu nötig, die Dualzahl mit führenden Nullen aufzufüllen.

Beispiel:
$11010111 = 011\ 010\ 111 = 3\ 2\ 7_8$

1.2.1.4 Hexadezimalsystem (Sedezimalsystem)

B = 16
Zifffern: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
Beispiel:
$D7_{16} = D \cdot 16^1 + 7 \cdot 16^0 \quad (= 215_{10} = 11010111_2)$

Bemerkung:
Eine Dualzahl läßt sich ohne viel zu rechnen als Hexadezimalzahl schreiben, indem man jeweils 4 Bit von rechts her zu einer Gruppe zusammenfaßt und jeweils als Hexadezimalzahl schreibt.
Eventuell ist dazu nötig, die Dualzahl mit führenden Nullen aufzufüllen.

Beispiel:
$1000011010111 = 0001\ 0000\ 1101\ 0111 = 1\ 0\ D\ 7_{16}$

1.2.2 Darstellen von Zeichen durch Bitmuster

Da der Mikrocomputer nur 0 und 1 versteht, müssen auch Zahlen und Texte, die man dem Mikrocomputer mitteilen will, als Bitmuster dargestellt werden. Alle Mikrocomputer verwenden als Zeichencode den ASCII (American Standard Code for Information Interchange) Code mit 256 Zeichen.

1.2.3 Darstellen von Maschinenbefehlen durch Bitmuster

Die Befehle, mit denen dem Mikrocomputer mitgeteilt wird, was er zu tun hat, werden vom Anwender in Form eines Programmes zusammengestellt, welches dann im Arbeitsspeicher des Mikrocomputer niedergelegt wird. Für jeden Mikrocomputer gibt es einen charakteristischen Befehlssatz, in welchem jedem einzelnen Befehl ein bestimmtes Bitmuster zugeordnet ist.

Beispiel für die Interpretation eines Bitmusters durch einen Mikrocomputer
Das Bitmuster 47_{16} kann also folgendes für den Mikrocomputer bedeuten:

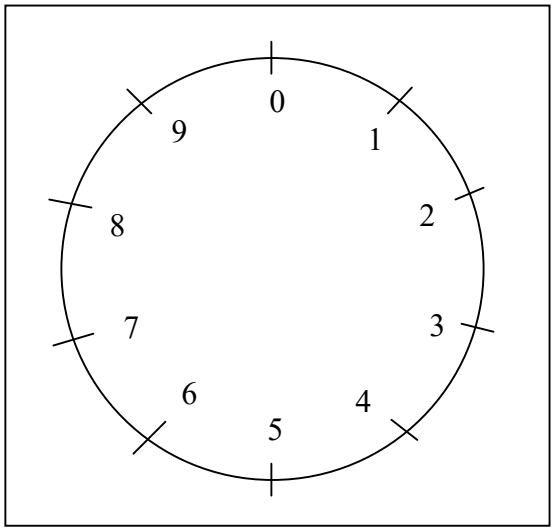
- Zahl: 71_{10}
- ASCII Zeichen: G
- Befehl: MOV B, A

Der Mikrocomputer muß also instruiert werden, wie er das Bitmuster interpretieren soll. Wie dies geschieht, wird später behandelt.

1.3 Rechnen in verschiedenen Zahlensystemen

1.3.1 Darstellung von negativen Zahlen

1.3.1.1 Motivation



4 ⊕ 3 = 7
7 ⊕ 2 = 9

aber:
9 ⊕ 1 = 0
8 ⊕ 3 = 1
7 ⊕ 6 = 3

Wenn man auf einer Kreisscheibe (z.B.einem Tacho) "addiert" ("Tachoaddition"), dann gilt für diese Tachoaddition ⊕

Wenn aber
9 ⊕ 1 = 0
dann kann man
9 als -1,
8 als -2,
7 als -3, usw. auffassen.

Wenn man die Zahlen 0 bis 9 in zwei gleich große Teile zerlegt,

dann sind die positiven Zahlen:
0 ,1,2,3,4

und die negativen Zahlen:
9 ⇔ -1
8 ⇔ -2
7 ⇔ -3
6 ⇔ -4
5 ⇔ -5

Wenn die negativen Zahlen zusammenaddiert nicht ihren negativen Zahlenbereich verlassen (z.B. verlässt −3 ⊕ −4 (= −7 < −5) den negativen Zahlenbereich), dann kann man mit negativen Zahlen über die Tachoaddition addieren:
−1 ⊕ −3 = 9 ⊕ 7 = 6 = −4

1.3.1.2 Dualzahlen im Mikrocomputer

Im Mikrocomputer werden Zahlen als Bitmuster fester Länge L = n + 1 dargestellt.
Außerdem benötigt die Einführung der negativen Dualzahlen die Voraussetzung einer festen Länge der Dualzahlen.
Im Folgenden sei L = 8.
Also haben alle Dualzahlen die Länge 8.
Damit lassen sich 256 verschiedene Zahlen darstellen.

Positive Zahlen: (0 bis 127)
Alle positiven Zahlen beginnen mit einer 0 in der höchstwertigsten Stelle der Dualzahl.
Die restlichen 7 Stellen stellen als Dualzahl den Wert der Zahl dar.
0: 0000 0000
1: 0000 0001
2: 0000 0010
.
.
127: 0111 1111

Das Einerkomplement einer Dualzahl erhält man, indem man in dieser Dualzahl jede 0 durch 1 ersetzt und jede 1 durch eine 0.
Das Zweierkomplement einer Dualzahl erhält man, indem man zum Einerkomplement dieser Dualzahl 1 addiert.
Beispiel:
Einerkomplement von 101 = 010
Zweierkomplement von 101 = 011

Negative Zahlen: (-1 bis -128)
Es wird das Zweierkomplement der positiven Zahl genommen:
-1: 1111 1111
-2: 1111 1110
-3: 1111 1101
.
.
-128: 1000 0000

allgemein:

$-x := 2^{n+1} - x$ für $0 \leq x \leq 2^n$

Bemerkungen:
1) Alle negativen Zahlen beginnen mit einer 1 in der höchstwertigsten Stelle der Dualzahl.

2) L=8: `-1 (dezimal) = 11111111 (dual)`
L=16: `-1 (dezimal) = 1111111111111111 (dual)`

3) Der Mikroprozessor kann Zahlen entweder als vorzeichenbehaftet, oder als Zahlen ohne Vorzeichen verarbeiten. Die Wahl liegt beim Anwender ! Für beide Fälle gibt es Befehle.
Z.B: MUL (vorzeichenlos), IMUL (vorzeichenbehaftet)

1.3.2 Rechnen im Dualsystem

1.3.2.1 Addition

```
     1   1   1   0                          1   1   1   1
 +  ₁1   1   0   1                      +  ₁1  ₁1  ₁0   1   0
----------------                       --------------------
 1   1   0   1   1                        1   0   1   0   0   1
```

1.3.2.2 Subtraktion

```
     1  ₁0  ₁0   1   1                      1  ₁0   1  ₁0   1  ₁0   1
 -    ¹  ₁1   1   1   0                 -    ¹   1  ₁0   1  ₁0   1   0
--------------------                   ----------------------------
     0   0   1   0   1                      0   1   0   1   0   1   1
```

1.3.2.3 Subtraktion mit dem Zweierkomplement

x - y = x + (-y) und Nichtbeachten des Überlaufs aus der höchstwertigsten Stelle.

Beispiel:

```
     1   0   0   1   1
 -       1   1   1   0
--------------------
             ?
```

Einerkomplement von 0 1 1 1 0 = 1 0 0 0 1
Zweierkomplement von 0 1 1 1 0 = 1 0 0 0 1 + 1 = 1 0 0 1 0

```
     1   0   0   1   1
 +   1   0   0   1   0
--------------------
 1 |  0   0   1   0   1 |
```

1.3.2.4 Umwandlung von Dezimalzahlen in Dualzahlen

$30_{10} = ?_2$

```
30 : 2 = 15  Rest  0
15 : 2 =  7  Rest  1
 7 : 2 =  3  Rest  1
 3 : 2 =  1  Rest  1
 1 : 2 =  0  Rest  1
```

Damit:
$30_{10} = 1\,1\,1\,1\,0_2$
Bemerkung:
Die Reste von unten nach oben gelesen ergeben die gesuchte Dualzahl.
Dieses Prinzip gilt auch bei der Umwandlung von Dezimalzahlen in anderen Zahlensystemen.

1.3.3 Rechnen im Hexadezimalsystem

1.3.3.1 Addition

```
     D   E   9   3                    6   2   5   8
 +  ¹1  ¹B   8   8                +  ¹3  ¹F  ¹E   D
 ----------------                ----------------
     F   A   1   B                    A   2   4   5
```

1.3.3.2 Subtraktion

```
     F  ¹A  ¹1   B                    A  ¹2  ¹4  ¹5
 -  ¹D  ¹E   9   3                -  ¹3  ¹F  ¹E   D
 ----------------                ----------------
     1   B   8   8                    6   2   5   8
```

1.3.3.3 Umwandlung von Dezimalzahlen in Hexadezimalzahlen:

$15000_{10} = ?_{16}$

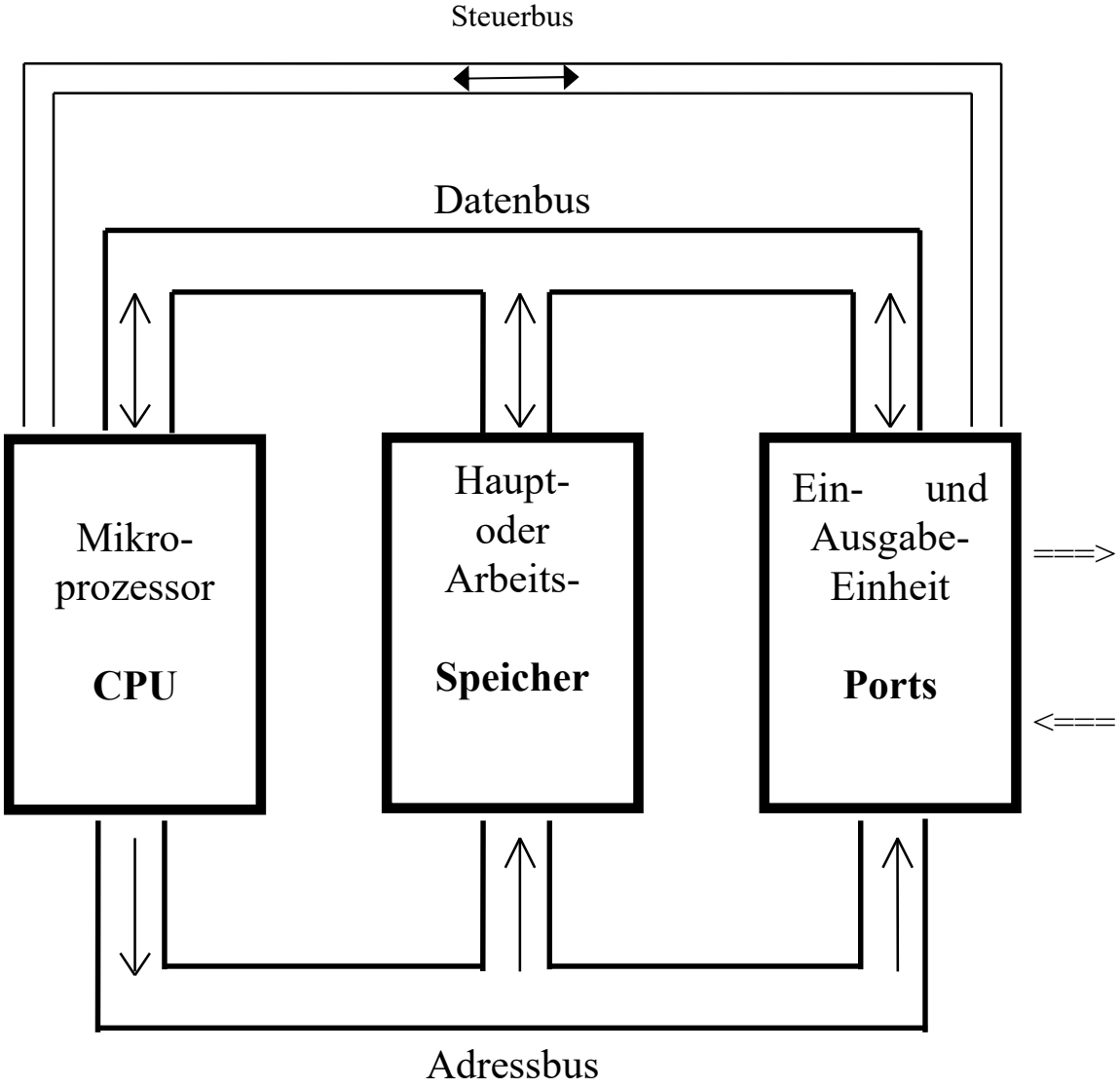
```
15000 : 16 = 937  Rest  8
  937 : 16 =  58  Rest  9
   58 : 16 =   3  Rest 10 (A)
    3 : 16 =   0  Rest  3
```

Damit:
$15000_{10} = 3A98_{16}$

Bemerkung:
Die Reste von unten nach oben gelesen ergeben die gesuchte Hexadezimalzahl

1.4 Überblick über einen Mikrocomputer

Das Blockschaubild eines Mikrocomputers:



Der Mikrocomputer besteht aus der *CPU* (Central Processor Unit), *Arbeitsspeicher* und der *Ein / Ausgabe -Einheit*.

Der wichtigste Bestandteil des Mikrocomputers ist die CPU.
Ihre Aufgabe ist es die Verarbeitung des Programms durchzuführen und den Funktionsablauf des ganzen Mikrocomputers zu steuern.
Die CPU besteht im wesentlichen aus dem *Rechenwerk* und dem *Steuerwerk*.
Im Rechenwerk (ALU = Arithmetic Logic Unit) werden die arithmetisch und logischen Operationen durchgeführt.
Das Steuerwerk sorgt für die Koordination des Mikrocomputers bei der Ausführung der Befehle.
Die CPU holt Befehle aus dem Arbeitsspeicher und führt sie aus. Die Gesamtheit der nacheinander auszuführenden Befehle zur Erfüllung einer bestimmten Aufgabe heißt *Programm*.
Die Zeit, die benötigt wird, um einen Befehl aus dem Arbeitsspeicher zu holen und auszuführen, heißt *Befehlszyklus*. Sie ist je nach Befehlsart verschieden lang.

Der *Arbeitsspeicher* besteht aus elektronischen Speichern zur Aufbewahrung von Bitmustern und ist nur ein Teil des gesamten *Adressraums* (Ein/Ausgabe-Einheit gehört nicht zum Adressraum dazu), den ein Mikroprozessor ansprechen kann.
Der Arbeitsspeicher hat beim Mikrocomputer zwei Funktionen. Er enthält das Programm und die momentan "in Arbeit " befindlichen Daten. Die Adresse (Hausnummer) ist eine eindeutige Bezeichnung eines Speicherplatzes des Arbeitspeichers oder der Ein/Ausgabe-Einheit.

Bem:
IBM-kompatible Computer haben nur einen Teil des Betriebssystems (das BIOS) in einem nicht löschbaren Bereich des Adressraums (= die Bytes, die durch die CPU adressierbar sind) - dem sogenannten BIOS-ROM - eingebaut.
Der Rest des Betriebssystems wird von Diskette oder Festplatte nachgeladen.

Um Daten zwischen dem Mikrocomputer und peripheren Geräten auszutauschen, ist die Ein / Ausgabe -Einheit im Mikrocomputer notwendig, über die der Datenfluß erfolgt.
Die Anschlußstellen der Ein / Ausgabe -Einheiten zur Umgebung heißen Ports.

Ein *Datenbus* besteht aus parallel verlaufenden Leitungen, auf denen die Daten zwischen der CPU und dem Arbeitsspeicher oder der Ein / Ausgabe -Einheit übertragen werden können.
Wenn der Datenbus z.B. aus 8 Leitungen besteht, können alle 8 Bit parallel (gleichzeitig) übertragen werden. Die Daten können in 2 Richtungen übertragen werden (bidirektional).
Die *Datenbusbreite* ist die Anzahl der Leitungen, aus denen der Datenbus besteht.
Verkehrstechnisch gesprochen gibt die Datenbusbreite die Anzahl der Fahrbahnen zum Datentransport an.

Ein *Adressbus* besteht aus parallel verlaufenden Leitungen, über die eine bestimmte Adresse des Arbeitsspeichers (oder ein anderer Teil des Adressraums) oder der Ein / Ausgabe -Einheit angesprochen werden kann. Man kann sich einen Adressbus als einen Kabelstrang eines Telefonnetzes vorstellen. Hier lassen sich ja auch viele Teilnehmer durch eine bestimmte Nummer (Telefonnummer) eindeutig anwählen.
Die Adressen werden nur in einer Richtung übertragen, nämlich von der CPU zum Arbeitsspeicher (unidirektional).
Die *Adressbusbreite* ist die Anzahl der Leitungen, aus denen der Adressbus besteht.
Für die Anzahl der Speicheradressen gilt:

$$\text{Anzahl der Speicheradressen} = 2^{\text{Adressbusbreite}}$$

Der *Steuerbus* überträgt die Steuerimpulse zu den Komponenten des Mikroprozessors.

Unter *Wortlänge* eines Mikrocomputers versteht man diejenige Anzahl von Bits, welche die CPU in ihrem Rechenwerk "auf einmal" parallel verarbeiten kann.

Zum Verständnis:
Die Wortlänge hat entscheidenden Einfluß auf die Leistungsfähigkeit des Mikroprozessors. Mit einer Wortlänge von 4 Bit können in einem Mikroprozessor nur 4-Bit Operationen (z.B. Addition, Subtraktion) durchgeführt werden. Die Verarbeitung z.B. einer 16-Bit Zahl muß dann in 4 Schritten erfolgen. Mit einem 16-Bit Prozessor kann diese 16-Bit Zahl dagegen mit einem Befehl verarbeitet werden.
Das gilt auch dann, wenn der Mikroprozessor z.B. eine Wortlänge von 16 Bit hat, der Datenbus aber nur eine Datenbusbreite von 8 Bit hat (8088 Mikroprozessor von Intel). Natürlich verliert man dadurch Geschwindigkeit gegenüber Mikroprozessoren mit einer Datenbusbreite von 16 Bit.

Bem:
Ein 16-Bit Prozessor ist ein Prozessor mit einer 16-Bit Wortlänge.
Ein 32-Bit Prozessor ist ein Prozessor mit einer 32-Bit Wortlänge, usw.

1.5 Verschiedene Mikroprozessortypen

Typ	Datenbus-breite	Ersthersteller	Zweithersteller
Wortlänge = 4 Bit			
TMS1000-Serie	4 Bit	Texas Instruments	
COP-Serie	4 Bit	National Semic.	
Wortlänge = 8 Bit			
8080	8 Bit	Intel	Siemens, AMD, National, Texas, ...
8085	8 Bit	Intel	Siemens, AMD, NEC,Mitsubishi, ...
Z 80 / 180	8 Bit	Zilog	Sharp, Thomson, NEC, Toshiba, ...
HD 64180	8 Bit	Hitachi	
NSC 800	8 Bit	National Semic.	SMC
MC 6800/02/..	8 Bit	Motorola	Hitachi, National, Thomson, Harris
Wortlänge = 12 Bit			
IM 6100	12 Bit	Intersil	Harris
Wortlänge = 16 Bit			
8086 / 80186	16 Bit	Intel	Siemens, NEC, Harris, OKI, ...
8088 / 80188	8 Bit	Intel	Siemens, Fujitsu, AMD, OKI, ...
V 30	16 Bit	NEC	Zilog
V20	8 Bit	NEC	Zilog
80286	16 Bit	Intel	Siemens, AMD, Harris
68000/08/10	16 Bit	Motorola	Rockwell, Thomson, Hitachi, ...
Z 8000/01/02/..	16 Bit	Zilog	Sharp, Thomson
Z 280	8 / 16 Bit	Zilog	
NS 16000/16/32	16 Bit	National Semic.	
TMS 9900/95	16 Bit	Texas Instr.	
Wortlänge = 32 Bit			
80386	32 Bit	Intel	AMD, C&T, Cyrix, Texas Instr.
80386SX / SL	16 Bit	Intel	AMD, C&T, Cyrix, Texas Instr., ...
80486/80486SX	32 Bit	Intel	AMD, Cyrix, IBM
Pentium (80586)	64 Bit	Intel	
68020/30/40/60	32 Bit	Motorola	Thomson, Hitachi, Valvo
NS 32032 / 332 / 532	32 Bit	National Semic.	Texas Instr.
NS 32016	16 Bit	National Semic.	Texas Instr.
Z 80000	32 Bit	Zilog	
WE 32100	32 Bit	AT&T	Zilog
V 60 / 70	32 Bit	NEC	

2 Prinzip, Aufbau und Funktionsweise eines Mikrocomputers

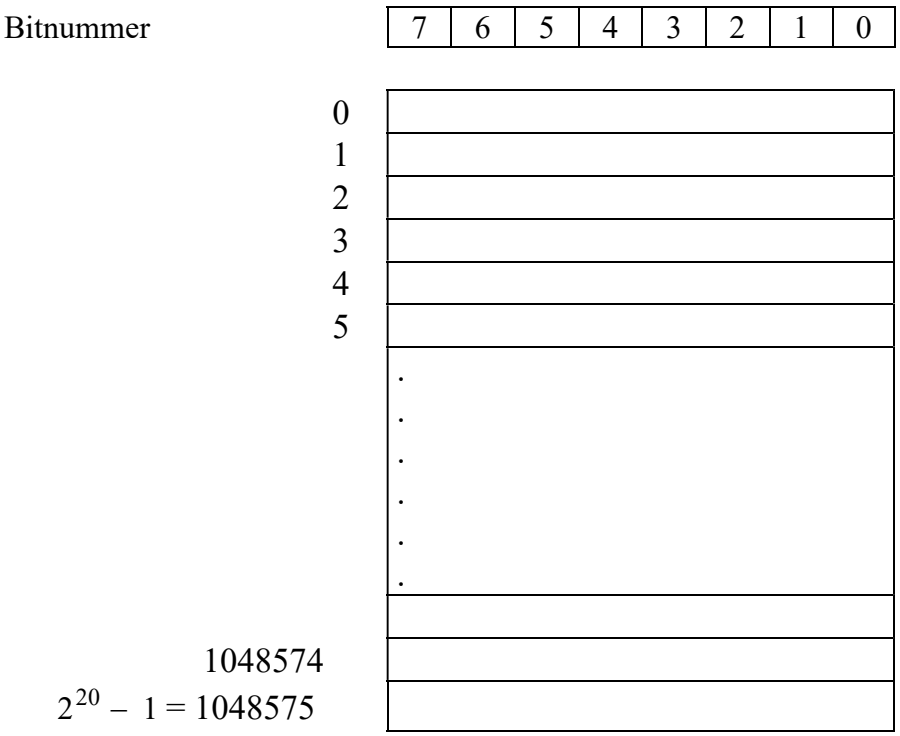
2.1 Allgemeine Beschreibung eines Mikrocomputers

Das schrittweise Verfahren zur Lösung eines gegebenen Problems nennt man einen Algorithmus.
Ein Algorithmus ist also eine Vorschrift nach der ein Problem Schritt für Schritt gelöst wird.
Er muß nach einer endlichen Zahl von Schritten zum Abschluß gebracht werden.
Sobald die Lösung für ein Problem in Form eines Algorithmus dargestellt ist, kann der Algorithmus von einem Computer abgearbeitet werden.
Ein Algorithmus ist ein Verfahren (Verarbeitungsvorschrift), das so präzise formuliert ist, daß es von Maschinenbefehlen durchführbar ist.
Damit ein Algorithmus von einem Computer abgearbeitet werden kann, muß er als eine Folge von Maschinenbefehlen (kurz Befehl) dargestellt werden.
Jeder Maschinenbefehl besteht aus einer Folge von Nullen und Einsen.
Im ersten Teil eines Maschinenbefehls (z.B. das erste Byte) steht (in kodierter Form) die Art des Befehls (z.B. Addition, Kopieren einer Konstanten in den Arbeitsspeicher) und die Länge (Anzahl der Bytes) des Maschinenbefehls.
Im zweiten Teil stehen die Konstanten oder Adressen, die zur Ausführung des Maschinenbefehls benötigt werden.
Jeder Maschinenbefehl steht im Arbeitsspeicher.
Ein Arbeitsspeicher ist ein Speicher, der aus nummerierten Speicherzellen (Speicherplätzen) besteht. Die Nummer einer Speicherzelle nennt man ihre Adresse.
Beispiel:

Arbeitsspeicher 2^{20} x 8 Bit, also mit Kapazität 1 MB.

Dieser Arbeitsspeicher hat 2^{20} Speicherzellen, wobei jede Speicherzelle die Länge 8 Bit hat

Schematische Darstellung eines Arbeitsspeichers: (10^6 x 8 Bit)



Die kleinste adressierbare Einheit ist also 1 Byte.
Jedes Byte des Arbeitsspeichers ist durch eine eindeutige Adresse ansprechbar, die zwischen 0 und $2^{20} - 1$ liegt.
Außer den Maschinenbefehlen stehen auch noch Daten im Arbeitsspeicher.
Daten sind wie Maschinenbefehle aus einer Folge von Nullen und Einsen aufgebaut.

Die wesentliche Aufgabe des Mikroprozessors ist es die Maschinenbefehle im Arbeitsspeicher auszuführen und falls die Maschinenbefehle es verlangen, die dort stehenden Daten zu ändern.

2.1.1 Compiler und Assembler

Eine Maschinensprache besteht aus Maschinenbefehlen.
Maschinenbefehle sind durch Befehlsformate charakterisiert und bestehen aus 0 en und 1 en.
Eine Assemblersprache besteht aus symbolischen Befehlen wie z.B. ADD AX, BX
Jedem Maschinenbefehl ist ein symbolischer Befehl zugeordnet.
Für den Menschen als Benutzer ist die Maschinensprache schwer lesbar.
Deshalb bevorzugt er die bequeme symbolische Schreibweise der Assemblersprache.
Ein Assembler ist ein Übersetzungsprogramm, das ein Programm in der Assemblersprache (kurz: Assemblerprogramm) in ein Programm in der Maschinensprache (kurz: Maschinenprogramm) übersetzt.
Ein Compiler ist ein Übersetzungsprogramm, das ein Programm in einer menschennahen, leicht verständlichen "Hochsprache" (z.B. Pascal, C, C++, Java, Basic, Prolog, usw.) zuerst in die Assemblersprache übersetzt und danach in die Maschinensprache.

Bem:
Der Compiler bzw. Assembler untersucht die symbolischen Darstellung der Befehle auf Fehler in der Syntax (Rechtschreibung) und meldet sie.
Wenn z.B. bei ADD AX, BX das D weggelassen wurde, also AD AX, BX geschrieben wurde, meldet der Assembler einen Fehler.

2.1.2 Mechanismus der Befehlsabarbeitung

Woher weiß der Mikroprozessor, welchen Befehl er als nächsten ausführen soll ?
Der Mikroprozessor hat einen Befehlszähler (engl: Instruction Pointer, Programm Counter), in dem die Adresse steht, wo der Befehl anfängt, der als nächstes zu bearbeiten ist.

Sobald dieser Befehl aus dem Arbeitsspeicher geholt und ausgeführt wird, "bestimmt" der Mikroprozessor die Adresse des nächsten zu bearbeitenden Befehls und schreibt diese Adresse in den Befehszähler.

Wie "bestimmt" der Mikroprozessor die Adresse des nächsten zu bearbeitenden Befehls (den er in den Befehlszähler schreibt) ?
Der Mikroprozessor erkennt am "Aussehen" des gerade bearbeiteten Befehls wie die Adresse des nächsten zu bearbeitenden Befehls aussehen muß und zwar:
a)
Bei vielen Befehlen wird zur Adresse des gerade bearbeiteten Befehls die Länge (in Bytes) dieses Befehls hinzuaddiert.
b)
Bei Sprungbefehlen steht die Adresse des nächsten zu bearbeitenden Befehls im Sprungbefehl selbst drin.

Die Reihenfolge der Ausführung der einzelnen Befehle geschieht also nach folgendem sich wiederholenden Schema:

1)
Befehl ausführen, dessen (Anfangs) Adresse im Befehlzähler IP steht.
Nach Ausführung dieses Befehls bekommt der Befehlszähler IP einen neuen Wert.

2)
weiter mit 1)

Bem:
Wenn das erste Byte des Befehls vom Mikroprozessor gelesen wird, weiß dieser aufgrund des Informationsgehaltes dieses Bytes, wie lang dieser Befehl ist und liest deshalb die restlichen Bytes des Befehls in den Mikroprozessor.

2.1.3 Register eines Mikroprozessors

Befehle benutzen Register oder verändern Register.
Register sind ein kleines Gedächtniss (schnelle Speicherelemente) im Mikroprozessor. Sie können benutzt werden, um Daten vom Arbeitsspeicher in den Mikroprozessor zu kopieren (oder umgekehrt vom Mikroprozessor in den Arbeitsspeicher).
Der Mikroprozessor kann auf Register schneller zugreifen als auf den normalen Arbeitsspeicher.
Jedes Register hat einmalige Eigenschaften und bestimmte Fähigkeiten, die kein anderes Register und keine Speicherstelle des Arbeitsspeichers aufweist.

Beispiele für Register:
1) Befehlszähler (IP)

2) Akkumulator (AX)
Wird immer benötigt, wenn man eine Multiplikation oder eine Division durchführen will.

2.1.4 Struktur eines Maschinenbefehls

Ein Maschinenbefehl wird vom Mikroprozessor ausgeführt.
Ein Maschinenbefehl besteht aus einer Kombination von Nullen und Einsen.

Ein Maschinenbefehl besteht aus zwei Teilen:

Operations - Code (Op - Code)	Operandenteil

Op - Code : WAS soll gemacht werden ?
- Datentransport
- Addieren
- AND, OR
- Bits testen

Operandenteil : WOMIT, mit welchen Operanden (Daten) soll die im Op - Code angegebene Operation durchgeführt werden ?
Der Operandenteil enthält entweder
- Konstante, die unmittelbar weiterverarbeitet werden, oder
- Adressen, die angeben wo die zu verarbeitenden Konstanten zu finden sind

Die Länge eines Befehls ist die Anzahl der Bytes, aus denen der Op - Code und der Operandenteil bestehen.

2.1.5 Befehlsformat

Der Operandenteil (und der Op-Code) können jeweils eine bestimmte Struktur besitzen.

Beispiel (Operandenteil besitzt bestimmte Struktur):

Befehlsformat:

00000011	Adresse - low	Adresse - high

Symbolisch: ADD AX, [a]

verbal: Addiert zum Akkumulator den Inhalt der Adresse a hinzu.

konkreter Fall:

03	A 1	B2

(hexadezimale Darstellung)

symbolisch: ADD AX, [B2A1]

Beispiel:

Befehlsformat:

10000001	Data - low	Data - high

Symbolisch: ADD AX, d

verbal: Addiert zum Akkumulator die Konstante d hinzu.

konkreter Fall:

81	C3	FF

(hexadezimale Darstellung)

symbolisch: ADD AX, FFC3

Bemerkung:
Der Befehlssatz eines Mikroprozessors ist die Menge aller Befehle, die der Mikroprozessor ausführen kann.
Die Befehlsformate dieser Befehle sind in Handbüchern zu den jeweiligen Mikroprozessoren zu finden.

2.1.6 Das Zeitverhalten

Bei einem Mikroprozessor werden alle Elemente synchron geschaltet, die logischen Operationen des Systems zu bestimmten Zeitpunkten vollzogen. Dies geschieht durch den Taktgeber (Clockgenerator).

Die Zeit, die ein Mikroprozessor braucht, um einen Befehl auszuführen, hängt ab von der

(1) Taktfrequenz
(2) Der Anzahl der Taktzyklen, die für den Befehl benötigt werden.

Beispiel:
Der FMP arbeitet mit einer Taktfrequenz von 10 MHz.
Ein Taktzyklus benötigt somit:

$$T = \frac{1}{f} = \frac{1}{10\ MHz} = 100\ ns$$

Der Befehl
`ADD AX, a`
benötigt 10 Takte, braucht also 10 * 100 ns, (siehe später)
wobei a einen Direktwert (Konstante) bedeutet.

Die Anzahl der Taktzyklen, die zusätzlich benötigt werden, um falls erforderlich die effektive Adresse (EA) eines Speicheroperanden zu bilden, hängt vom Befehl ab.
Die Anzahl der Taktzyklen eines Befehls sind dem Prozessorhandbuch zu entnehmen.

3 Beispiel: Der fiktive Mikroprozessor **FMP**

Bem: Alle Zahlen im Kapitel FMP sind, falls nicht anders gekennzeichnet, hexadezimal !!

Ein (Beispiel) Programm im Arbeitsspeicher, der mit dem FMP verbunden ist:
(Adressen und Speicherinhalte sind alle hexadezimal angegeben)

Adresse	Speicherinhalt	symbolisch	Befehlslänge
0001	?		
0002	?		
	.. ..		
8900	F1	MOV AX, [890C]	3
8901	0C		
8902	89		
8903	A1	ADD AX, [890D]	3
8904	0D		
8905	89		
8906	A2	ADD AX, 05	2
8907	05		
8908	F2	MOV [890E], AX	3
8909	0E		
890A	89		
890B	ED	HLT	1
890C	03		
890D	04		
890E	99 ($\rightarrow$ 0C)		
	.. ..		
A189	11		
	.. ..		
FFFF	?		

Die Register des FMP (und die Inhalte vor der Befehlsausführung)

AX:	15	
IP:	89	00

Bemerkung:
1) Breite des Adressbus des FMP : 2 Byte
Breite des Datenbus des FMP : 1 Byte

2) Nach Inbetriebnahme des FMP haben alle Register einen bestimmten (Anfangs)Wert.
Dieser Wert kann durch einen Schalter eingestellt werden.
In diesem Beispiel wurden die Register auf folgende Werte (siehe oben) eingestellt:
AX = 15
IP = 8900

3.1 Der Befehlssatz des FMP

Befehlsformat :	F1	adress-low	adress-high

Symbolisch: MOV AX, [a]

verbal: Kopiere den Inhalt der Adresse a in AX

Anzahl Takte: 10

Befehlsformat :	F2	adress-low	adress-high

Symbolisch: MOV [a], AX

verbal: Kopiere AX an die Adresse a (in den Inhalt der Adresse a)

Anzahl Takte: 30

Befehlsformat :	A1	adress-low	adress-high

Symbolisch: ADD AX, [a]

verbal: Der Inhalt der Adresse a wird zu AX hinzuaddiert.

Anzahl Takte: 25

Befehlsformat :	A2	d

Symbolisch: ADD AX, d

verbal: Die Konstante d wird zu AX hinzuaddiert.

Anzahl Takte: 15

Befehlsformat :	0C	adress-low	adress-high

Symbolisch: SUB AX, [a]

verbal: Der Inhalt von a wird von AX subtrahiert.

Anzahl Takte: 45

Befehlsformat : | ED |

Symbolisch: HLT

verbal: Der Prozessor stellt die Befehlsbearbeitung ein.

Anzahl Takte: 10

Befehlsformat : | F3 | d |

symbolisch: MOV AX, d
verbal: Kopiere die Konstante d in den Akkumulator AX
Anzahl Takte: 10

für später: (wird noch behandelt)

Befehlsformat : | BE | d |

Symbolisch: JMP d

verbal:
1) Zum Inhalt des Befehlszählers wird die Befehlslänge (wie bei jedem anderen Befehl auch) hinzuaddiert.
2) Zum Inhalt des Befehlszählers wird die Distanz d (-128 bis +127) hinzuaddiert. (IP + d $\rightarrow$ IP)

Anzahl Takte: 7

3.2 Problem

Wie bestimmt der Mikroprozessor aus den 2 Bytes einer Adresse die richtige Adresse ?
Im Arbeitsspeicher steht zum Beispiel (siehe oben)

8900	F1	MOV AX, [890C]
8901	0C	
8902	89	

Es gibt zwei Möglichkeiten, wie der Mikroprozessor diese zwei (0C und 89) Bytes zu einer Adresse zusammenbasteln könnte:

1) 0C89
2) 890C (FMP bastelt diese Adresse !)

Der FMP geht nach folgendem Schema vor:
Das Byte an der niederwertigeren Adresse im Arbeitsspeicher wird zum niederwertigeren Byte in der Adresse.
Das Byte 0C ist das niederwertigere Adressbyte, da es die kleinere Adresse (8901 ist kleiner als 8902) besitzt. Also ist es auch das niederwertigere (steht rechts) Byte der Adresse.

Anschaulich:
Die Adresse wird im Arbeitsspeicher "von unten nach oben gelesen"

3.3 Dynamisches Verhalten der Registerinhalte und der Speicherinhalte

Annahme:
Der Inhalt des Registers IP wird am Anfang (manuell, mit Hilfe eines Schalters) auf 8900 gesetzt.

IP (alt)	IP (neu)	AX	[890E]	Befehl
8900		15	99	vor der Befehlsausführung
8900	8903	03	99	MOV AX, [890C]
8903	8906	07	99	ADD AX, [890D]
8906	8908	0C	99	ADD AX, 05
8908	890B	0C	0C	MOV [890E], AX
890B	890C	0C	0C	HLT

3.3.1 Programmverlauf

1)
Der Mikroprozessor veranlaßt, daß der Inhalt der Adresse, die im IP steht, aus dem Arbeitsspeicher in den Mikroprozessor gebracht wird.
Der Inhalt der Adresse 8900, also F1, wird daher in den Mikroprozessor gebracht.
Der Mikroprozessor interpretiert diesen Wert als den Operationsteil des Kopierbefehls (mit Befehlsänge 3), der den Speicherinhalt der Adresse, die dem Operationsteil folgt, in den Akkumulator AX kopiert. Außerdem wird der IP um die Befehlslänge erhöht.
Konkret:
Der Inhalt der Adresse 890C, also 03, wird in AX gebracht und der IP um die Befehlslänge 3 erhöht.
Im IP steht also 8903

2)
Der Mikroprozessor veranlaßt, daß der Inhalt der Adresse, die im IP steht, aus dem Arbeitsspeicher in den Mikroprozessor gebracht wird.
Der Inhalt der Adresse 8903, also A1, wird daher in den Mikroprozessor gebracht.
Der Mikroprozessor interpretiert diesen Wert als den Operationsteil des Addierbefehls (mit Befehlsänge 3), der den Speicherinhalt der Adresse, die dem Operationsteil folgt, zum Akkumulatorinhalt von AX hinzuaddiert. Außerdem wird der IP um die Befehlslänge erhöht.
Konkret:
Der Inhalt der Adresse 890D, also 04, wird zu AX, also 03, hinzuaddiert und der IP um die Befehlslänge 3 erhöht.
In AX steht jetzt also 7 (= 03 + 04).
Im IP steht also 8906

3)
Der Mikroprozessor veranlaßt, daß der Inhalt der Adresse, die im IP steht, aus dem Arbeitsspeicher in den Mikroprozessor gebracht wird.
Der Inhalt der Adresse 8906, also A2, wird daher in den Mikroprozessor gebracht.
Der Mikroprozessor interpretiert diesen Wert als den Operationsteil des Addierbefehls (mit Befehlslänge 2), der den Wert,der dem Operationsteil folgt, zum Akkumulatorinhalt von AX hinzuaddiert. Außerdem wird der IP um die Befehlslänge erhöht.
Konkret:
05 wird also zu AX, also 07, hinzuaddiert und der IP um die Befehlslänge 2 erhöht.
In AX steht jetzt also 0C (= 07 + 05).
Im IP steht also 8908

4)
Der Mikroprozessor veranlaßt, daß der Inhalt der Adresse, die im IP steht, aus dem Arbeitsspeicher in den Mikroprozessor gebracht wird.
Der Inhalt der Adresse 8908, also F2, wird daher in den Mikroprozessor gebracht.
Der Mikroprozessor interpretiert diesen Wert als den Operationsteil des Kopierbefehls (mit Befehlsänge 3), der den Akkumulatorinhalt von AX an die Adresse, die dem Operationsteil folgt, kopiert. Außerdem wird der IP um die Befehlslänge erhöht.
Konkret:
Der Inhalt von AX, also 0C, wird daher an die Adresse 890E kopiert (die Adresse 890E hat daher den Inhalt 0C) und der IP um die Befehlslänge 3 erhöht.
Im IP steht also 890B

5)
Der Mikroprozessor veranlaßt, daß der Inhalt der Adresse, die im IP steht, aus dem Arbeitsspeicher in den Mikroprozessor gebracht wird.
Der Inhalt der Adresse 890B, also ED, wird daher in den Mikroprozessor gebracht.
Der Mikroprozessor interpretiert diesen Wert als den Operationsteil des Haltebefehls (mit Befehlsänge 1), der die Befehlsbearbeitung beendet.
Dieser Befehl hat keinen Operandenteil.

3.3.2 Programmbeschreibung

Programmbeschreibung:
Es werden 2 Zahlen addiert (mit Adresse 890C und 890D).
Zu diesem Ergebnis werden 5 dazu addiert und das Ergebnis an der Adresse 890E abgespeichert.
kurz: [890C] + [890D] + 05 $\rightarrow$ [890E]

3.3.3 Belegung im Arbeitsspeicher

Das Programm und seine Daten belegen im Arbeitsspeicher folgenden Speicherplatz:
Programm: von 8900 bis 890B
Daten: von 890C bis 890E

3.3.4 Bemerkung

2 Speicheroperanden können nicht direkt addiert werden wie z.B:
ADD [890D] [890C],
da dieses Befehlsformat nicht im Befehlssatz vorkommt.

3.3.5 Frage

Was geschieht, wenn der Befehlszähler am Anfang auf 8901 gesetzt wird ?

Antwort:
Der Mikroprozessor liest den Inhalt der Adresse, die im IP steht (8901), aus dem Arbeitsspeicher in den Mikroprozessor und versucht ihn als Op - Code eines Befehls zu interpretieren.Er "schaut" also in seinem Befehlssatz nach, ob es ein Befehlsformat mit dem Op - Code 0C gibt. Es gibt so ein

Befehlsformat:	0C	adress-low	adress-high

Der Mikroprozessor faßt daher die dem Op - Code 0C folgenden 2 Byte als eine Adresse auf und subtrahiert daher den Inhalt der Adresse A189, also 11 vom Akkumulatoreninhalt AX und erhöht den Inhalt des Befehlszählers um 3 auf 8904.
Der Mikroprozessor liest den Inhalt der Adresse, die im IP steht (8904), aus dem Arbeitsspeicher in den Mikroprozessor und versucht ihn als Op - Code eines Befehls zu interpretieren. Er "schaut" also in seinem Befehlssatz nach, ob es ein Befehlsformat mit dem Op - Code 0D gibt. Es gibt kein solches

Befehlsformat:	0D	adress-low	adress-high

Da er kein Befehlsformat mit dem Op - Code 0D in seinem Befehlssatz "findet", hört er auf zu arbeiten. Mit drücken der Reset Taste kann der FMP wieder gestartet werden.

3.3.6 Aufgabe

1) Schreiben Sie ein Programm, das den Inhalt der Adresse 4711 zum Inhalt der Adresse 4712 addiert und von diesem Ergebnis den Inhalt der Adresse 4713 subtrahiert. Das Ergebnis soll an die Adresse 4720 kopiert werden.
kurz: [4711] + [4712] - [4713] $\rightarrow$ [4720]

2) Wie kann Speicherplatz (bei den Daten) eingespart werden ?

3.3.7 Aufgabe

Annahme:
IP = 1000, AX = 0

Schreiben Sie ein Programm, das 15 * 17 berechnet.

Lösung: Im Arbeitsspeicher steht ab der Adresse 1000 insgesamt 15 mal der Befehl:
ADD AX, 17

Frage: Wieviel Byte Speicherplatz benötigt dieses Programm ?
Antwort: 15 * 2 Byte = 30 Byte

Frage: Kann dieses Programm mit Befehlen aus dem Befehlssatz des FMP wesentlich kleiner gemacht werden ?
Antwort: Nein

Frage: Um welchen Befehl müßte der FMP Befehlssatz erweitert werden, damit der Additionsbefehl ADD AX, 17 zwar 15 mal ausgeführt wird, aber nicht 15 mal im Speicher steht ?

Antwort: Um einen Sprungbefehl. Dieser Befehl muß den Inhalt von IP um soviel erniedrigen, damit dort die Adresse steht, deren Inhalt der Beginn des esten Bytes des ADD Befehls ist.

3.3.8 Der Sprungbefehl

Wir geben der Firma MIKROP den Auftrag, den Befehlssatz unseres FMP Mikroprozessors um den folgenden (präzise beschriebenen) Sprungbefehl zu erweitern.

Befehlsformat :

BE	d

Symbolisch: JMP d

verbal:
1) Zum Inhalt des Befehlszählers wird die Befehlslänge (wie bei jedem anderen Befehl auch) hinzuaddiert.
2) Zum Inhalt des Befehlszählers wird die Distanz d (-128 bis +127) hinzuaddiert. (IP + d $\rightarrow$ IP)

Anzahl Takte: 7

Bemerkung:
Der Befehl

BE	FC

interpretiert also das BE folgende Byte als eine vorzeichenbehaftete Dualzahl (Distanz -4) und bedeutet also nicht JMP 252 sondern JMP -4

3.3.9 Beispiel für den FMP Sprungbefehl in einem Programm

Adresse	Speicherinhalt	symbolisch	Befehlslänge
0001	?		
0002	?		
	..		
	..		
	..		
	..		
1000	A2	ADD AX, 17	2
1001	17		
1002	BE	JMP -4	2
1003	FC		
..	..		
..	..		
..	..		
..	..		

Die Register des FMP (und die Inhalte vor der Befehlsausführung)

AX:	00	
IP:	10	00

Dynamisches Verhalten der Registerinhalte und der Speicherinhalte
(Alle Zahlen sind hexadezimal aufzufassen)

IP (alt)	IP (neu)	AX	Befehl
1000		00	vor der Befehlsausführung
1000	1002	17	ADD AX, 17
1002	1000 (= 1002 + 2 - 4)	17	JMP -4
1000	1002	2E	ADD AX, 17
1002	1000 (= 1002 + 2 - 4)	2E	JMP -4
..			
..			

Der FMP befindet sich in einer Endlos - Schleife. Bei jedem Schleifendurchgang wird der Inhalt von AX um 17 erhöht. Leider verläßt der FMP die Schleife nicht nach 15 Durchgängen, wie von uns in der obigen Aufgabe gefordert wurde.
Deshalb gibt es bei realen Mikroprozessoren sogenannte bedingte Sprungbefehle.

Aufgabe:
Schreiben Sie ein möglichst wenig speicherverbrauchendes Programm, das 2 * a + 2 * b berechnet. Braucht 2 * a + 2 * b oder 2 * (a + b) mehr Speicherplatz ?