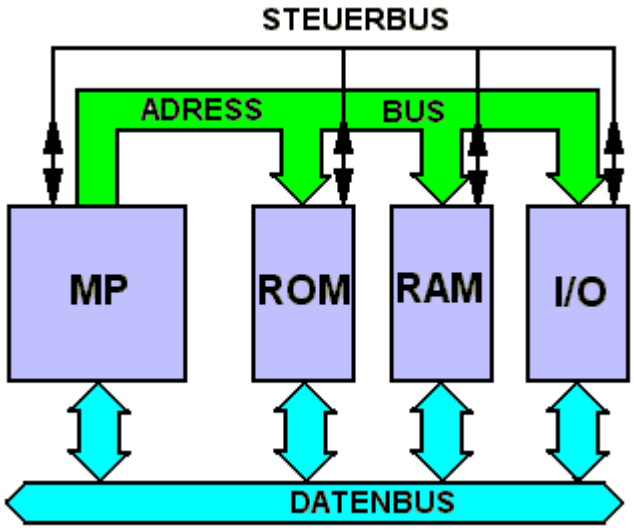
Die Struktur eines Mikrocomputersystems



Der hier dargestellte Mikrocomputer besteht aus den folgenden Bausteinen:

MP: Mikroprozessor mit Steuer- und Rechenwerk

ROM: Read Only Memory (Nur Lese Speicher), Festwertspeicher

RAM: Random Access Memory (Wahlfreier Zugriff) , Schreib- Lese Speicher

I: Input Device (Eingabebaugruppe), Dateneingabe

O: Output Device (Ausgabebaugruppe), Datenausgabe

Der Mikroprozessor, die Speicher und die Ein-/Ausgabebaugruppen sind über drei Busse miteinander verbunden:

Adreßbus:

Datenbus:

Steuerbus:

Aufgaben:

a. Der Adressbus dieses Mikrocomputersystems umfasst 16 Leitungen **(A0 bis A15)**.
Wie viele Speicherstellen lassen sich damit adressieren?

b. Welche Signale liegen auf dem Adressbus, wenn die Speicherstelle 21082_{10} angesprochen werden soll?

c. Der Datenbus dieses Mikrocomputersystems umfasst 8 Leitungen **(D0 bis D7)**.
Wie viele unterschiedliche Zeichen können übertragen werden?

d. Welche Signalzustände liegen auf den Leitungen, wenn das ASCII-Zeichen "u" übertragen wird?

Aufgaben:

a. Wenn der Arbeitsspeicher des IBM-PC nur 640 KByte umfassen kann, wie viele dezimale Speicherzellen (Byte) sind das?

b. Der Arbeitsspeicher beginnt bei der hexadezimalen Adresse 0 und endet bei welcher hexadezimalen Adresse?

c. Der Grafikspeicher fängt bei der nächsten Speicherzelle an. Welche hexadezimale Adresse ist das?

d. Für ROM-Erweiterungen (Steckkarten) ist der Speicherbereich ab C0000 hexadezimal vorgesehen. Wie viel Speicherplatz bleibt dann für die Grafik?

e. Das BIOS-ROM beginnt ab der Adresse E0000 hexadezimal. Wie viel Speicher in KByte bleibt für das BIOS dann übrig?

Wie viele Adressleitungen werden benötigt?

Bestimmung der benötigten Adressleitungen für eine vorgegebene Speichergröße:

SG = Speichergröße (dezimal)
n = Anzahl der Adressleitungen
2 = Basis des binären Zahlensystems

$$\mathbf{SG = 2^n}$$

Durch Logarithmieren kann die Gleichung nach n umgestellt werden:

$$\log SG = n * \log 2$$

$$\mathbf{n = \frac{\log SG}{\log 2}}$$

Aufgabe: Speicheradressen

Der Hauptspeicher eines PC ist mit 4 Speichermodulen (SIMMs) zu je 64 MByte bestückt.

a. Wie viele Adressleitungen braucht der Mikroprozessor mindestens, um den gesamten Speicher zu adressieren?

b. Welche Anfangs- und Endadressen haben die einzelnen Speicherblöcke?

c. Die zweiten 32 MByte des dritten Speichermoduls sind defekt. Von welcher Anfangsadresse bis zu welcher Endadresse funktioniert der Speicher nicht mehr?

Aufgabe Speicheradressen

Im Adressraum des Arbeitsspeichers liegt ein Teil des Betriebssystems MS-DOS ab der Speicheradresse 00C80 (hex). Der Programmteil ist 5 KByte lang.

Bei welcher Adresse befindet sich das **Ende** des Programmteils?

Aufgabe Datenbus

Über einen **8-Bit-Datenbus** können mit einer Transferoperation natürliche Zahlen zwischen 0 und 255 (dez.) übertragen werden.

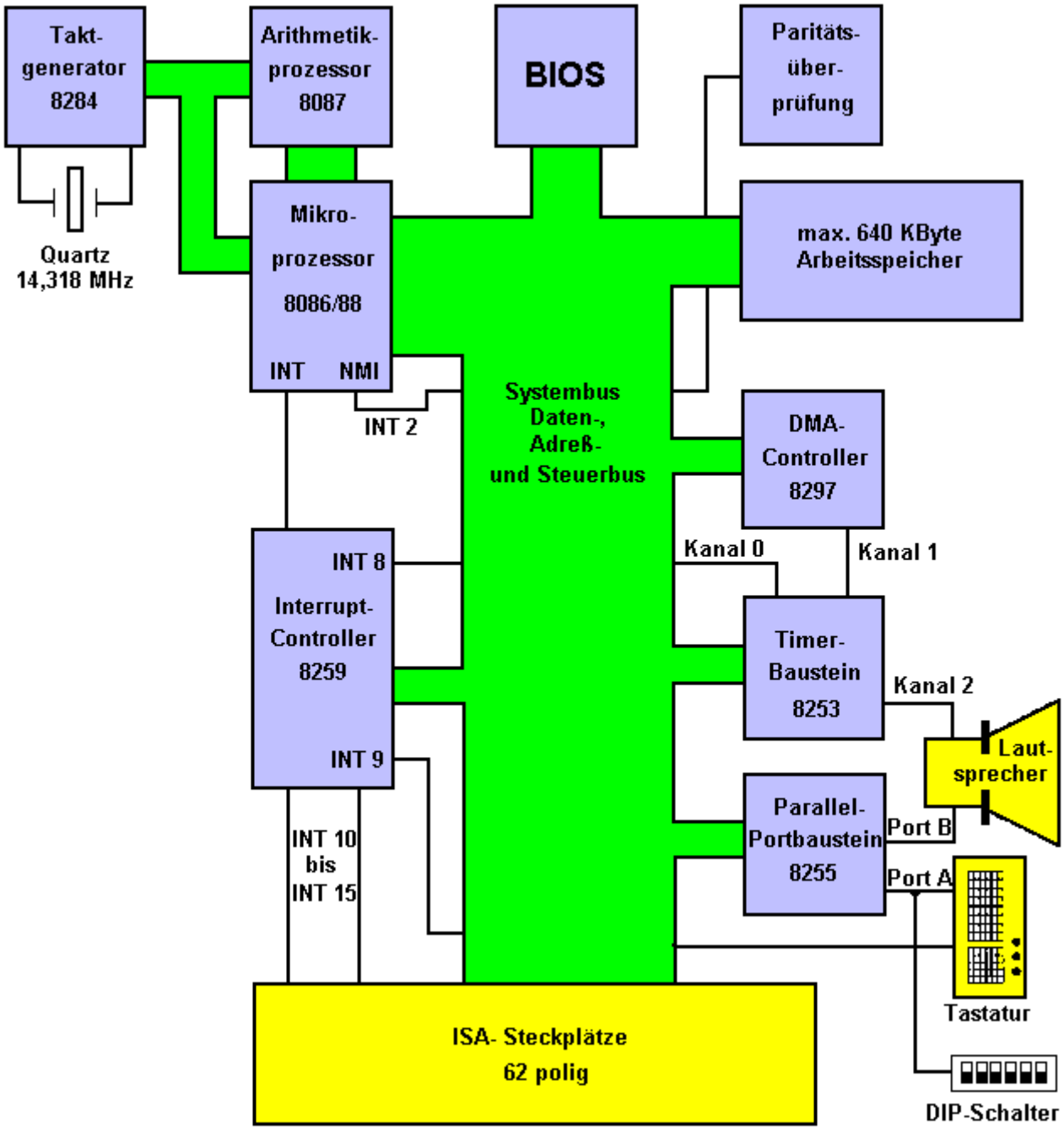
a. Welcher Zahlenbereich der natürlichen Zahlen kann über einen Datenbus von 16-Bit mit einer Transferoperation übertragen werden?

b. Welcher Zahlenbereich der natürlichen Zahlen (einschließlich 0) über einen 32-Bit-Datenbus?

Aufgabe Speicheradressen

Der folgende ASCII-Text beginnt ab der Speicherstelle **0042B** (hex):
"Das Attributbyte im Textspeicher der Grafikkarte bestimmt die Art der Zeichendarstellung".
Die "" sind nicht Bestandteil des Textes. Im Text sind Leerzeichen, aber kein Zeilenvorschub und Wagenrücklauf enthalten.
Bei welcher Speicherstelle endet der Text?

BUS-Systeme



In dieser Grafik ist die Struktur der ersten IBM-PC dargestellt. Dieser PC konnte nur **1MByte Speicher** adressieren.
1 MByte entspricht 1024 * 1 KByte. 1 KByte entspricht 1024 Byte.
Das Herz der Mutterplatine (Motherboard) ist der Mikroprozessor. Hier kam der 8086 oder der 8088 von Intel zum Einsatz. Beide Prozessoren sind 16-Bit-Prozessoren, doch der 8088 arbeitet im Unterschied zum 8086 mit einem externen 8-Bit-Datenbus.

Der 8086/88-Prozessor kann 1 **Mbyte Speicher** (das sind 1048576 Speicherzellen) adressieren.

- **Arbeitspeicher**
  In diesem Speicherbereich liegt zu Beginn der Arbeitsspeicher (RAM) der maximal

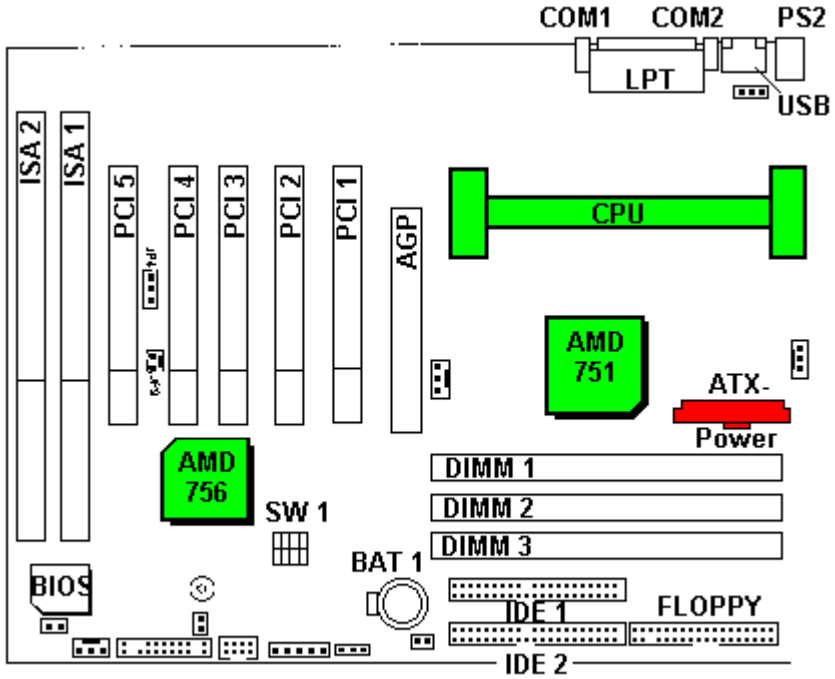
640 KByte groß sein darf.

- **Speicher der Grafikkarte**
  Darauf folgt der Speicher der Grafikkarte. Der Speicher befindet sich auf der Grafik-Steckkarte (Arbeitsspeicher und Video-BIOS).

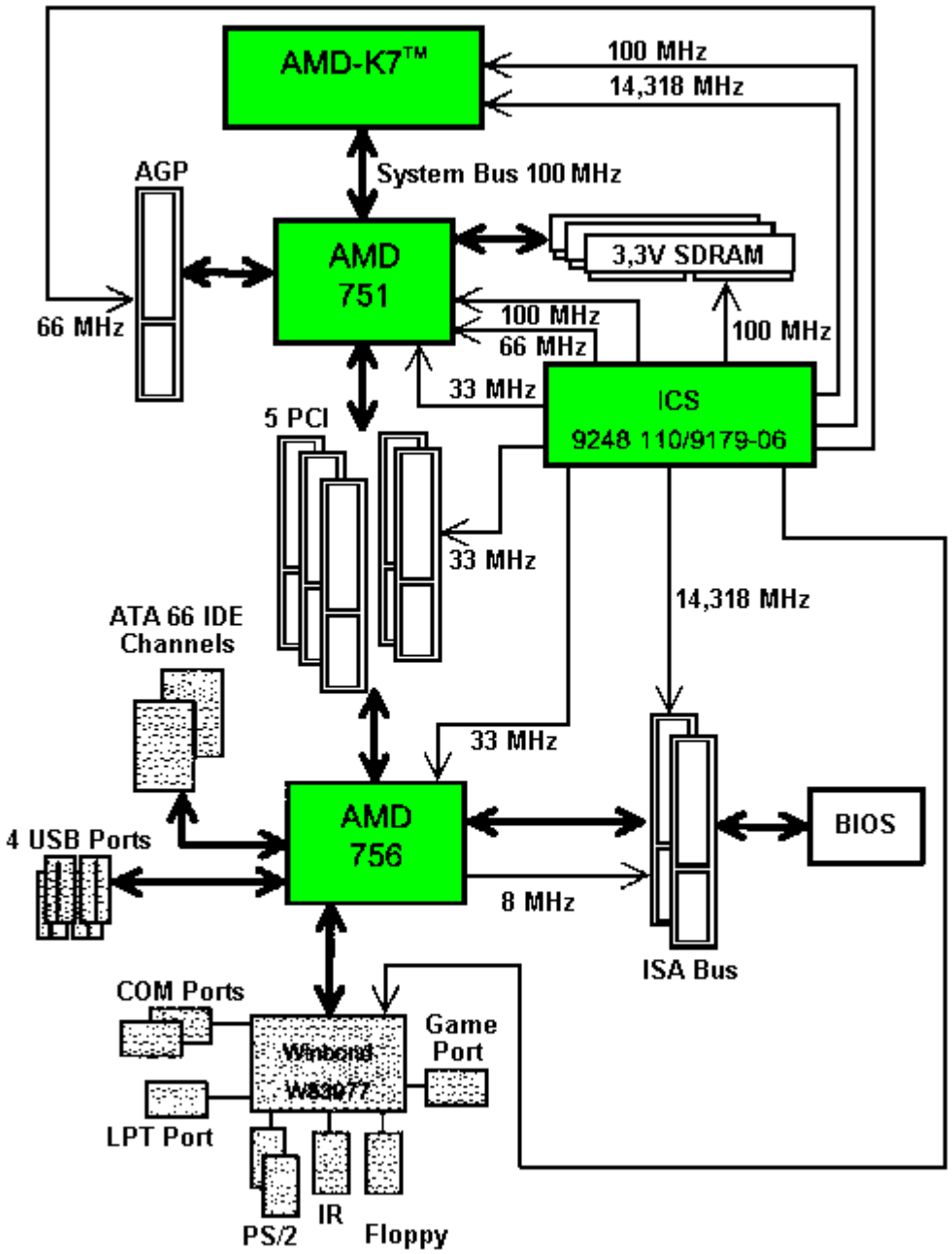
- **Reservierter Speicherbereich für weitere Steckkarten**
  Hier befindet sich häufig Festwertspeicher mit Programmen für den Festplatten-Controller.

- **BIOS**
  Den Abschluß des 1MByte-Speicherbereiches bildet ein Festwertspeicher (ROM), der als BIOS (Basic Input Output System) bezeichnet wird und die zur Verwaltung der PC-Harware nötigen Betriebssystem-Routinen enthält.

Ein Auszug aus der Bestückung eines Mainboards mit AMD-ATHLON-Prozessor (August 2000)



Prinzip-Schaltbild eines Mainboards mit AMD-ATHLON-Prozessor (August 2000)



Aufgabe Bus-Systeme

Beschreiben Sie bitte kurz die Aufgabe der folgenden Busse:

a. Adressbus:

b. Datenbus:

c. Steuerbus:

Aufgabe Chipsatz des Motherboards

Welche Aufgaben hat der sogenannte Chipsatz auf dem Motherboard zu erfüllen?
(Kreuzen Sie bitte alle richtigen Aussagen an)

a. Der Chipsatz enthält den Memory-Cache-Controller.

b. Der Chipsatz übernimmt die Tastatursteuerung.

c. Der Chipsatz enthält die **PCI-Bridge**, die die Datenströme zwischen Prozessor und PCI-Bus steuert.

d. Der Chipsatz enthält das BIOS des Computers.

e. Der Chipsatz steuert den Datenverkehr mit den Schnittstellen.

Aufgabe Speicher- und Ein-/Ausgabebaugruppen

Erläutern Sie bitte kurz die folgenden Begriffe:
a) RAM:

b) ROM:

c) I/O-Baugruppen:

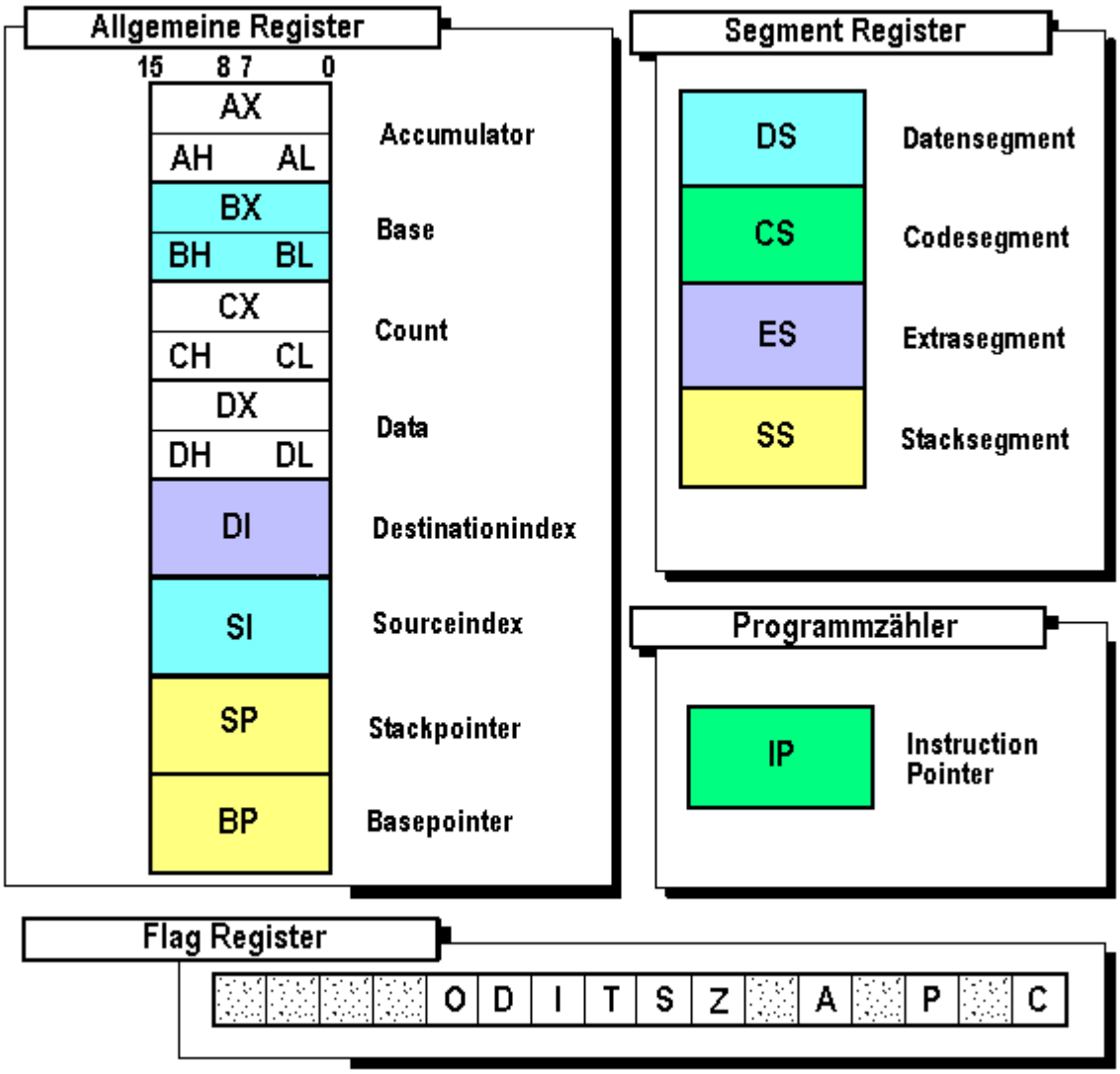
Aufgabe Grafikspeicher

Der Textspeicher der Grafikkarte beginnt bei der Adresse **B8000**. Bei welcher Adresse endet der Textspeicher, wenn zwei komplette Bildschirmseiten mit 80 Spalten und 24 Zeilen für **ASCII-Zeichen** zur Verfügung stehen?
(Bedenken Sie bitte, das zu jedem ASCII-Zeichen ein Attribut-Byte gehört, welches die Zeichenfarbe, die Hintergrundfarbe und den Blinkmodus bestimmt).

Aufgabe Speicheradressierung

Wie viele Adressleitungen braucht ein Mikroprozessor mindestens, wenn er 1 GigaByte an Speicherzellen adressieren können soll?

Die Register des 8086/8088-Mikroprozessors



Die wesentlichen Funktionen eines jeden Prozessors werden mit Hilfe seiner Register ermöglicht. Daten werden aus dem Speicher in Register geladen, dort mit Hilfe von Maschinensprachebefehlen verarbeitet und anschließend wieder in den Speicher zurückgeschrieben.
Die Register sind bis zu den modernen Prozessoren erhalten geblieben. Allerdings sind ab **80386** die allgemeinen Register auf 32 Bit erweitert worden. Weiterhin sind **Register für den Protected Mode** hinzugekommen.

Die Allzweckregister
Die acht Allzweckregister des 8086-Prozessors (alle 16 Bit breit) sind in den meisten Befehlen sowohl als Ursprungsregister als auch als Zielregister für Berechnungen und Datenübertragungen sowie als Zeiger für Adressbereiche und als Zähler beteiligt. Jedes dieser Allzweckregister kann einen 16-Bit-Wert aufnehmen, kann mit dem Inhalt einer Speicherstelle geladen werden und den Inhalt des Registers im Speicher ablegen, sowie in arithmetischen und logischen Operationen eingesetzt werden.

Beispiel

```
1    mov  ax, 5
2    mov  dx, 9
3    add  ax, dx

1 Lade das ax-Register mit dem Wert 5
2 Lade das dx-Register mit dem Wert 9
3 Addiere den Inhalt der beiden Register. Das Ergebnis
  befindet sich dann wieder im ax-Register (Akkumulator).
```

Dieses kleine Beispiel kann mit Hilfe von DEBUG.EXE ausprobiert werden.

```
Starten des Debuggers:
DEBUG  <CR>
-     (Eingabeaufforderung (Promt) des Debuggers)
-r    (Anzeigen der Registerinhalte)
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1196  ES=1196  SS=1196  CS=1196  IP=0100   NV UP EI PL NZ NA PO NC
```

Eingabe des Assemblerprogramms:
Die Eingabe von "a" bewirkt, daß in der Regel ab der Offsetadresse 100 (hex) die Maschinenbefehle eingegeben werden können.

```
-a
1196:0100 mov ax,5
1196:0103 mov dx,9
1196:0106 add ax,dx
1196:0108
```

abgeschlossen wird die Eingabesequenz dadurch, dass einfach kein Befehl, sondern <CR> eingegeben wird.

Die eingegebenen Befehle können durch das Kommando "u" ab der gewünschten Offsetadresse wieder angezeigt werden.

```
-u100
1196:0100 B80500   MOV     AX,0005
1196:0103 BA0900   MOV     DX,0009
1196:0106 01D0     ADD     AX,DX
```

Ausführen des Programmbeispiels
Das Maschinenprogramm wird mit der Eingabe der Zeichen
"p=[ab Adresse] [Anzahl der Befehle]"
gestartet. Für jeden Maschinenbefehl wird die Veränderung aller Register angezeigt. Die Veränderungen werden hier unterlegt dargestellt.

```
-p=100 3    (ab Offsetadresse 100 drei Befehle ausführen)
AX=0005  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1196  ES=1196  SS=1196  CS=1196  IP=0103   NV UP EI PL NZ NA PO NC
1196:0103 BA0900        MOV     DX,0009

AX=0005  BX=0000  CX=0000  DX=0009  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1196  ES=1196  SS=1196  CS=1196  IP=0106   NV UP EI PL NZ NA PO NC
1196:0106 01D0          ADD     AX,DX

AX=000E  BX=0000  CX=0000  DX=0009  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1196  ES=1196  SS=1196  CS=1196  IP=0108   NV UP EI PL NZ NA PO NC
1196:0108 76FE          JBE     0108
```

Das BX-Register
Das BX-Register (B für engl. Base) kann als Zeiger in Speicherbereiche verwendet werden. Dabei stellt BX allerdings nur einen Teil der Adresse dar. Das Speichersegment, auf das zugegriffen wird, wird über das Segmentregister (DatenSegment) DS bestimmt. Dieses Register läßt sich nicht direkt mit einem Wert laden. Es muß über ein anderes Register geladen werden.

```
mov ax,0
mov dx,ax
mov bx,0A
mov al,[bx]
```

Dieses kleine Programm holt einen Wert aus dem Speicher (*Segmentadresse 0000, Offsetadresse 000A*) Dieser Wert wird über den Zeiger bx-Register in das Register al geladen.

Eingeben des Programms mit Hilfe von Debug:

```
-a100
1596:0100 mov ax,0
1596:0103 mov ds,ax
1596:0105 mov bx,0a
1596:0108 mov al,[bx]
1596:010A
-
```

Ausführen

```
-p=100 4
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1596  ES=1596  SS=1596  CS=1596  IP=0103   NV UP EI PL NZ NA PO NC
1596:0103 8ED8          MOV     DS,AX

AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0000  ES=1596  SS=1596  CS=1596  IP=0105   NV UP EI PL NZ NA PO NC
1596:0105 BB0A00        MOV     BX,000A
```

```
AX=0000  BX=000A  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0000  ES=1596  SS=1596  CS=1596  IP=0108   NV UP EI PL NZ NA PO NC
1596:0108 8A07          MOV     AL,[BX]                     DS:000A=25

AX=0025  BX=000A  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0000  ES=1596  SS=1596  CS=1596  IP=010A   NV UP EI PL NZ NA PO NC
1596:010A 8946FE        MOV     [BP-02],AX                   SS:FFFE=F676
            -
```

Auf diese Weise wurde ein Wert (8-Bit) aus dem Speicher gelesen und im al-Register abgelegt.

Den Text “Hallo Welt” ausgeben

```
-a100
178A:0100 jmp 0200
178A:0103 db 0a,0d,"Hallo Welt!",0a,0d,"$"
178A:0113
-a200
178A:0200 mov ah,9
178A:0202 mov dx,0103
178A:0205 int 21
178A:0207 mov ah,0
178A:0209 int 21
178A:020B
-g

Hallo Welt!

Program terminated normally
-
```

Programmschleifen

Wir wollen uns einmal ansehen, wie lange es dauert, einen Registerinhalt von 65535 (dez) bis auf 0 herunterzuzählen. Dazu ziehen wir vom Inhalt des cx-Registers solange 1 ab, bis es zu 0 geworden ist.

```
      mov cx,FFFF  ;Lade Zähler mit 65535 dez.
loop: sub cx,1     ;Ziehe eins ab
      jnz loop     ;Wenn das Ergebnis nicht 0, dann weiter
      mov ah, 4C   ;Beende das Programm
      int 21       ;durch Übergabe an DOS
```

Das **cx-Register** wird zunächst mit FFFF (65535 dez) geladen.
Der nächste Befehl **subtrahiert 1** vom Inhalt des cx-Registers.
Dann wird das **zero-flag getestet**. Ist es nach der Subtraktion nicht 0, dann wird an der Speicherstelle **loop:** weitergemacht.
Erst wenn das cx-Register **bis auf 0 heruntergezählt wurde, wird das Programm angehalten.**

Das geht sehr schnell und da eigentlich nichts besonderes geschieht, ist es nicht sehr eindrucksvoll. Schachtelt man aber zwei Schleifen, so daß der Mikroprozessor 65535*65535 Subtraktionen ausführen muß, dann dauert es schon eine Weile

Der Programmzeiger (IP)

Der Programmzeiger (IP, Instruction Pointer) enthält immer den Offset der Speicherstelle, die den Befehl enthält, der als nächster ausgeführt werden soll.

Wenn ein Befehl ausgeführt wird, dann rückt der Programmzeiger automatisch zum nächsten Befehl vor. Normalerweise wird der Befehl an der nächsten Speicherstelle ausgeführt, aber bei einigen Befehlen, wie Unterprogrammaufrufen und Sprungbefehlen, erhält der Programmzeiger einen neuen Wert und verzweigt zu einer anderen Stelle im Programm.
Mit dem Programmzeiger allein kann man die Adresse, an der der nächste Befehl steht, nicht vollständig angeben. Durch die bereits erwähnte Segmentierung des Speichers wird die Situation etwas komplizierter. Beim Laden eines Befehls wird zunächst eine Basisadresse aus dem cs-Register geladen und dann liefert der Programmzeiger den Offset zu dieser Basisadresse.

Die anderen Register

Alle Register haben im Zusammenhang mit der Programmausführung spezielle Aufgaben, die teilweise relativ komplex sind. Hier soll aber nicht das gesamte Spektrum der Assemblerprogrammierung behandelt werden.

Ein Beispiel für die Programmierung mit einem Macro-Assembler (MASM, TASM)
Microsoft stellt einige Entwicklungswerkzeuge für die Assemblerprogrammierung kostenlos zur Verfügung. Die Firma Borland vermarktet ihre Macro-Assembler.
Macro-Assembler sind recht komplexe Werkzeuge, die einiges Wissen erfordern, um damit Programmstrukturen aufzubauen. Die richtige Verwendung der unterschiedlichen Programm-Segmente oder die Verwendung der Register zur Speicherverwaltung setzen umfangreiche Kenntnisse der Computerhardware und deren Verwaltung voraus. Ein Beispiel für die Verwendung von MASM soll dargestellt werden.

```
; Ein Beispiel für die Funktion des Microsoft MASM - Assemblers
; Wie schnell ist der Microprozessor?
; Es werden 4.294.967.296 = 4 Milliarden Subtraktionen ausgeführt
; R. Krüger 03.01.2008

cr          equ   0dh         ; Zeilenruecklauf
lf          equ   0ah         ; Zeilenvorschub
stdout      equ   1           ; Textausgabe zur Standardausgabe

; Der DOS-Interrupt wird als Makro beschrieben
Dos         MACRO intnum
            mov ah, intnum    ; Funktionsnummer in AH
            int 21h           ; DOS aufrufen
            ENDM              ; Makroende

DGROUP      group _DATA,STACK; Daten- und Stacksegment zusammenfassen
; Das Code-Segment wird Wordweise ausgerichtet
_TEXT       segment word public 'CODE'

; Code-Segment:_TEXT, Daten-Segment:DGROUP, Stacksegment = Stack
            assume  cs:_TEXT,ds:DGROUP,ss:STACK

main        proc far
            mov ax, DGROUP    ; Datensegment einstellen
            mov ds,ax         ; ds enthält den Beginn des Datensegments
            mov bx,stdout     ; Handle für Standard Output einstellen
            mov cx,msg_len    ; Länge der Meldung einstellen
            mov dx,offset msg ; Offsetadresse der Meldung einstellen
            Dos 40h           ; Meldung ausgeben

            ; Äußere Schleife mit 0 laden (65536)
            mov, dx, 0h

@@Inn:      mov cx, 0h        ; innerer Schleifenzähler mit 65536 laden
@@Loop:     sub cx, 1         ; Inneren Schleifenzähler herunterzählen
            jnz @@Loop  ;     Solange wiederholen bis Register auf 0
            sub dx, 1         ; Äußer Schleife um 1 herunterzählen,
                              ; Dann wieder die innere Schleife durchlaufen
            jnz   @@Inn

            Dos   4ch         ; Programm mit Code 0 beenden

main        endp
_TEXT       ends

_DATA       segment word public 'DATA'
msg         db cr,lf,'Start mit 4.294.967.296 Subtraktionen',cr,lf
msg_len     equ   $-msg
_DATA       ends

STACK       segment para stack 'STACK'
db          256 dup (?)
```

```
STACK       ends

end         main
```