# Kleincomputer KC85

## Beschreibung zu M027 DEVELOPMENT

**veb mikroelektronik**
**>wilhelm pieck<**
**mühlhausen**
**im veb kombinat mikroelektronik**

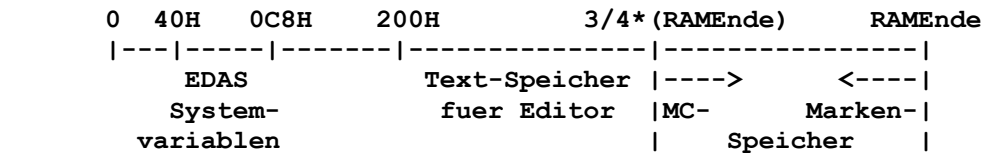
### 1. Einleitung

Der Modul M 027 "DEVELOPMENT" ist ein 8 K Byte Festwertspeichermodul. Er enthaelt vier 2 K Byte EPROM`s, einen 2-Pass-Assembler mit Editor (EDAS) fuer die UDOS-Sprachversion, einen Disassembler (DISASS), einen Testmonitor (TEMO) sowie zwei Druckertreiberroutinen. Der Modul belegt den Speicherbereich C000H bis DFFFH und kann ueber das Kommando SWITCH zu- und abgeschalten werden (siehe Modulhandhabung Anhang G). Beim Einsatz im KC 85/3 muss vor der Arbeit mit dem Development-Modul der interne BASIC-ROM abgeschaltet werden (SWITCH 2 0), da dieser den gleichen Speicherbereich belegt.
Im Programm EDAS sind der Texteditor, mit dem Programme erstellt werden koennen, und der Assembler, der diese Programme in den abarbeitbaren Zustand uebersetzt, enthalten. Der Nutzer kann zwischen beiden Programmteilen umschalten, ohne dass der im Rechner vorhandene Programmtext zerstoert wird bzw. nachgeladen werden muss.
Das Systemhandbuch und die dazugehoerigen Uebersichten sind fuer eine effektive Assemblerprogrammierung unbedingt erforderlich. Darin sind unter anderem Hinweise zur Benutzung der Unterprogramme des CAOS-Betriebssystems und zum Einbinden von selbsterstellten Programmen ins Menue des Betriebssystems enthalten.
Das Programm DISASS dient dem blockweisen Rueckuebersetzen von im Speicher stehenden Maschinenprogrammen in die Assemblermnemonik. Der Testmonitor ist zur Unterstuetzung der Assembler- und Maschinenprogrammierungen vorgesehen. Weiterhin steht ein Unterprogramm zur Bestimmung der Laenge und Art von Maschinenbefehlen zur Verfuegung.
Der Disassembler kann in Verbindung mit dem Schrittbetrieb des Testmonitors zur Darstellung der Befehlsmnemonik eingesetzt werden. Nach dem Einschalten des Moduls und dem Aufruf des Kommandos MENU erscheinen im Menue des Betriebssystems zusaetzlich die folgenden Kommandos:

EDAS
REEDAS
DISASS
CDISASS
TEMO
RETEMO
V24K6311
V24K6313

## 2. Editor

Nach dem Programmaufruf mit EDAS wird die Groesse des zur Verfuegung stehenden RAM-Speicherbereiches ermittelt (ohne diesen zu zerstoeren), und es wird die oberste Speicheradresse (RAM-Ende in HEX) zur Bestaetigung angeboten. Durch Eingabe eines niedrigeren Wertes kann ein Teil des Speicherbereiches vor dem Zugriff von EDAS geschuetzt werden, um z. B. weitere Maschinenprogramme (wie Druckertreiberroutine) abspeichern zu koennen. Eine falsche Zahleneingabe oder eine Eingabe ausserhalb des zugelassenen Zahlenbereiches wird zurueckgewiesen. Anschliessend wird der EDAS zugewiesene RAM-Speicher wie folgt initialisiert:




Danach tritt das Programm in den Kommando-Modus ein und das Editor-Menue mit den Kommandos zur Bearbeitung des Quelltextes wird angezeigt. Dabei sind auf dem Bildschirm rechts oben stets die freien Speicherplaetze (in HEX) eingeblendet. Den Kommando-Modus erreicht man beim Start ueber EDAS. Ist mit EDAS ein Quellprogramm erstellt worden, kann als Folge mit REEDAS gestartet werden. Dabei erfolgt keine Initialisierung des Speichers (Warmstart).

Quelltexte bleiben dann, wie schon erwaehnt, erhalten.

Die Anzeige des Kommandomenues erfolgt in der folgenden Form:

```
>>> EDAS V1.3 <<<      FREE:nnnn
-----------------------------------
*MENU
*EXIT
*CLEAR
*SAVE
*LOAD
*PRINT
*ASM
*FIND
*TOP
*BOTOM
*EDIT
*VERIFY
*
```

## 2.1. Kommando-Modus

Das Editor-Menue ist aehnlich wie das CAOS-Betriebssystem-Menue aufgebaut, nur erscheint ein "*" als Prompt-Zeichen. Nach dem "*"-Zeichen sind eines der im folgenden beschriebenen Editor-Kommandos einzugeben (in Grossbuchstaben!) bzw. die entsprechende Zeile im Menue mit dem Cursor anzuwaehlen. Die Kommandos MENU, CLEAR, SAVE, LOAD, PRINT und ASM fuehren nach Abarbeitung zurueck in den Kommando-Modus; FIND, TOP, BOTOM und EDIT leiten den Editier-Modus ein.

Im folgenden wird die Wirkungsweise der einzelnen Kommandos beschrieben:

MENU: Anzeige aller Kommandofunktionen.

EXIT: Verlassen von EDAS und Rueckkehr zur CAOS-Systemschleife.

CLEAR: Loeschen des gesamten Textspeichers. Die Markentabelle und das uebersetzte Programm bleiben erhalten.

SAVE: Abspeichern des im Hauptspeicher befindlichen Quelltextes auf Kassette. Nach Aufruf von SAVE ist ein (maximal 8 Zeichen langer) Name einzugeben. Diese Eingabe ist nach Einschalten des Kassettenrecorders mit <ENTER> abzuschliessen. Der Name wird automatisch durch den Dateityp ASM ergaenzt. Waehrend der Namenseingabe und der Kassettenausgabe kann jederzeit mit der BREAK-Taste abgebrochen werden.

LOAD:
Einlesen eines (mit dem EDAS-SAVE-Kommando abgespeicherten) Quelltextes von Kassette. Der einzulesende Text wird stets vor die erste Zeile der zuletzt bearbeiteten Textseite eingeschoben. Deshalb ist vor LOAD entweder der Textspeicher mit CLEAR zu loeschen, oder der Cursor im vorhandenen Quelltext so zu positionieren, dass der Text von der Kassette an die richtige Stelle geladen wird. Wie beim SAVE-Kommando ist ein maximal achtstelliger Name einzugeben, der vom Rechner durch den Dateityp ASM ergaenzt wird. Name und Typ muessen mit denen des zu ladenden Files uebereinstimmen. Anschliessend ist der Recorder zu starten. Alle eingelesenen Bloecke werden mit Blocknummer bzw. Dateiname (bei 1. Block) angezeigt. Bei Lesefehlern und vor Programmanfang folgt danach ein "*", sonst das Zeichen ">". Der Ladevorgang wird abgebrochen, wenn der Textspeicher vollstaendig belegt ist, oder wenn ein EOF-Zeichen (03H) im Text erkannt wurde. Tritt waehrend des Ladevorgangs ein Lesefehler auf ("*" erscheint), so kann nach Rueckspulen der Kassette erneut versucht werden, den fehlerhaften Block zu lesen. Gelingt das trotz mehrmaligen Probierens nicht, so ist der Ladevorgang mit der BREAK-Taste abzubrechen. Bei Abbruch ist der bis dahin eingelesene Text verfuegbar. Auf die BREAK-Taste reagiert der Rechner nur, wenn ein Signal vom Kassettenrecorder anliegt.

**PRINT:**

Ausdruck des Textspeichers. Es koennen folgende Parameter angegeben werden:

PRINT [m [n]]  m,n in HEX-Zahlen

m - Anzahl der Bildschirmzeilen fuer eine Druckzeile (beim Fehlen wird m=1 angenommen)
n - Anzahl der Druckzeilen je Seite (bei n=0 keine Seitenformatierung)

Werden weniger als 2 Argumente angegeben, so erfolgt die Druckausgabe ueber Bildschirm, bei Angabe von 2 Argumenten auf den Drucker (User-Kanal 1).

Beispiele:

PRINT 1 0 - Ausgabe auf Drucker ohne Seitenformatierung
PRINT 2 3C - Ausgabe auf Drucker mit Unterbrechung nach jeder Seite (3CH = 60 Zeilen)

Die Zeichenausgabe wird mit BRK abgebrochen und mit STOP angehalten (Fortsetzung mit beliebiger Taste). Ist kein Ausgabegeraet angeschlossen und PRINT wurde ohne Parameter eingegeben, erfolgt die Ausgabe auf den Bildschirm.

Soll ein Drucker mit selbsterstellter Ausgaberoutine angesteuert werden, so muss vor dem Start von EDAS auf Systemadresse 0B7BEH die Adresse der Ausgaberoutine eingetragen werden. Die Ausgaberoutine wird dann ueber den CAOS-Programmverteiler (UOT1) angesprungen und bekommt im Akkumulator den Zeichencode eines ASCII-Zeichens. Die Routine ist mit RET abzuschliessen, Register muessen nicht gerettet werden. Als Steuercodes sind 0AH fuer Line Feed und 0DH fuer Carriage Return zu verwenden.

**ASM:**

Der Editor wird verlassen und der Assembler aufgerufen (s. Abschn. 3).

**FIND:**

Durchsuchen des Quelltextes nach einem Suchwort.

Nach Aufruf ist ein Suchtext einzugeben. Dieser kann maximal 34 Zeichen lang sein. Die Texteingabe kann mit BREAK abgebrochen werden. Der im Speicher befindliche Quelltext wird nach diesem Suchtext durchmustert. Dabei wird zwischen Gross- und Kleinbuchstaben unterschieden. Begonnen wird in der Zeile nach der aktuellen Cursorposition. Wird der gesuchte Text gefunden, so steht der Cursor anschliessend am Anfang der entsprechenden Zeile, sonst nach der letzten Textzeile. Der Suchvorgang kann, ohne den Suchtext neu eingeben zu muessen, im Editier-Modus mit der Taste F2 wiederholt werden. Vor dem Durchsuchen sollte die Cursorposition mit TOP auf den Textanfang gestellt werden.

**TOP:**
**Cursor auf den Textanfang, Anzeige der ersten Textseite.**

**BOTOM:**
**Cursor auf die Zeile nach dem Textende, Anzeige der letzten Textseite (30 Zeilen, oder weniger fuer kuerzere Texte).**

**EDIT:**
**Anzeige der aktuellen Textseite (zuletzt bearbeitete Seite).**

**VERIFY:**
**Ueberpruefung eines auf Magnetband gespeicherten Programms auf fehlerfreie Aufzeichnung (siehe Bedienungsanleitung bzw. Systembeschreibung).**

## 2.2. Editier-Modus

Im Editier-Modus wird jeweils eine zu bearbeitende Textseite angezeigt. Auf diese Seite kann beliebig geschrieben und vorhandener Text veraendert werden. Die Textseite wird bei Betaetigung der Tasten BREAK, SHIFT+CUU oder SHIFT+CUD in den Textspeicher uebernommen. Dabei werden Leerzeilen gestrichen, wenn nicht mindestens ein Leerzeichen auf dieser Zeile eingegeben wurde. Einfuegen vor der ersten Zeile ist nicht moeglich. Wird die erste Zeile ueberschrieben, kann Quelltext eingefuegt werden.

Die Sondertasten haben folgende Bedeutungen:

F1 - Tabulator (in Leerzeilen automatisch gesetzt) wenn kein Text auf der Tabulatorposition steht, ansonsten wird der Text auf die naechste Tabulatorposition verschoben. Steht auf der Position jedoch schon Text, bleibt dieser erhalten und der zu verschiebende wird geloescht.

F2 - Wiederholung der Textsuche (s. FIND)

BRK - Beendigung des Editier-Modus und Rueckkehr ins Menue

INS - Einfuegen eines Leerzeichens links vom Cursor

DEL - Loeschen eines Zeichens mit Verdichten der Zeile

CLR - Loeschen eines Zeichens ohne Verdichten der Zeile

HOME - Cursor zum linken oberen Bildschirmrand

CUU - Cursor zum Zeilenanfang der vorherigen Zeile (bei oberster Zeile nur zum Zeilenanfang)

CUD - Cursor zum Zeilenanfang der folgenden Zeile. Am Seitenende wird, falls der Text weitergeht, der Bildschirm gerollt und die naechste Zeile angezeigt.

CUL - Cursor ein Zeichen nach links (bei Zeilenanfang keine Wirkung)

CUR - Cursor ein Zeichen nach rechts (bei Zeilenende keine Wirkung)

CR - Einfuegen einer Leerzeile nach Cursorzeile. Am Seitenende wird der Bildschirm gerollt.

SHIFT+DEL - Loeschen der Cursorzeile

SHIFT+HOME - Loeschen der aktuellen Textseite

SHIFT+CUL - Cursor zum Zeilenanfang

SHIFT+CUU - Eine Seite (30 Zeilen) zurueckblaettern *1

SHIFT+CUD - Eine Seite (30 Zeilen) vorwaertsblaettern *1

*1 Der Cursor bleibt an der Position stehen.

Fehlermeldungen:

Bei BRK, SHIFT+CUU und SHIFT+CUD kann die Anzeige >>>NO MEMORY<<< erscheinen, falls die aktuelle Textseite nicht mehr vollstaendig in den Textspeicher uebernommen werden kann. Durch Betaetigung einer beliebigen Taste wird der uebernommene Teiltext angezeigt.

# 3. Assembler

## 3.1. Allgemeines

Der im folgenden beschriebene EDAS-Assembler gestattet es dem Anwender, Programme in symbolischer Form auf dem Kleincomputer zu schreiben, d. h. durch Eingabe leicht zu merkender Abkuerzungen. Dadurch ist es moeglich, ohne Kenntnis des eigentlichen binaeren Formats des Maschinenbefehls Maschinenprogramme zu erstellen. Darueber hinaus brauchen auch die Werte von Speicheradressen nicht beruecksichtigt zu werden, sondern es genuegt an deren Stelle mit frei waehlbaren symbolischen Namen (sog. "Marken" oder "Labels") zu arbeiten, die die Speicherstellen kennzeichnen. Als Operanden koennen Namen von Speicherstellen, Registernamen und Konstanten auftreten.

Neben den eigentlichen vom Prozessor ausfuehrbaren Anweisungen umfasst die hier beschriebene Assemblersprache auch Anweisungen fuer den Uebersetzungsvorgang, die es dem Programmierer erlauben, die Arbeitsweise des Uebersetzungsprogramms zu beeinflussen. Solche nicht ausfuehrbaren Anweisungen nennt man "Pseudobefehle". Ein Programm, das aus einer Folge von ausfuehrbaren und nicht ausfuehrbaren Anweisungen besteht, nennt man Quellprogramm oder Source-Programm. In jeder Zeile dieses Quellprogramms koennen vor dem eigentlichen Befehl und den Operanden dieses Befehls auch eine Marke stehen und nach dem Befehl ggf. ein Kommentar zur Erlaeuterung.

Das gesamte Quellprogramm wird vom Assemblerprogramm in die eigentliche Maschinensprache (in binaerer Form) des U880-Prozessors (im folgenden auch CPU - central processor unit) uebersetzt, wobei jedem Assemblerbefehl genau ein Maschinenbefehl entspricht (s. Abb. 3.1). Das auf diese Weise erstellte Maschinenprogramm kann nun auf dem Computer direkt ausgefuehrt werden (im Gegensatz z. B. zu Basic-Programmen, die den Basic-Interpreter benoetigen).

Es sei darauf hingewiesen, dass es mehrere verschiedene Assemblersprachen fuer die U880 bzw. Z80 CPU gibt. Der hier beschriebene Assembler verarbeitet Quellprogramme in ZILOG-Mnemonik (wie z. B. die Assembler der Buerocomputer A5120/30 unter den Betriebssystemen UDOS und SCPX).

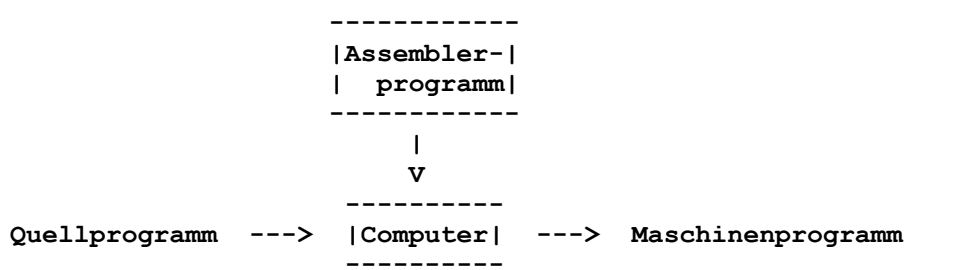



Abb. 3.1:
Assembliervorgang

## 3.2. Arbeitsweise des Prozessors

Mit dem U880-Prozessor stehen dem Programmierer ueber 600 Operationscodes fuer arithmetische, logische, Programmorganisations-, Datentransfer- sowie Ein-/Ausgabe-Befehle zur Verfuegung.

Saemtliche Befehle eines abzuarbeitenden Maschinenprogramms stehen in externen Speicherbausteinen (dem sog. Hauptspeicher) des Rechners. Die Befehlsablaeufe ("Befehlszyklen") sehen alle im Prinzip gleich aus (s. Abb. 3.2).

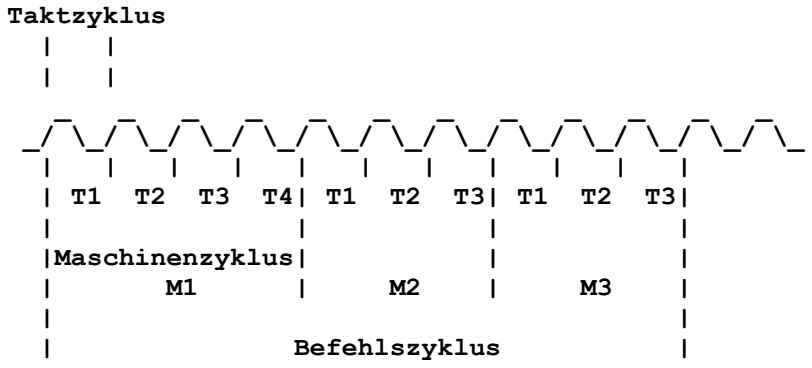



Abb. 3.2:
Beispiel eines Befehlszyklus

Man ersieht aus Abb. 3.2, dass folgende Hierarchie besteht:

Die Befehlszyklen bestehen aus mehreren "Maschinenzyklen". Die Anzahl der Maschinenzyklen pro Befehl ist unterschiedlich (zwischen 1 und 6). Jeder Maschinenzyklus besteht seinerseits aus mehreren "Taktzyklen" (3 bis 6 Taktzyklen). Die Dauer eines Taktzyklus ist durch die Frequenz des Taktgenerators der CPU gegeben. Beim KC 85/2 und KC 85/3 betraegt die Taktfrequenz 1,75 MHz, d. h. die Dauer eines Taktzyklus betraegt etwa 571 ns. Im Abschnitt 3.5 ist fuer die einzelnen Befehle der CPU die Gesamtanzahl der Taktzyklen jeweils angegeben. Man kann hieraus die Ausfuehrungszeit eines Befehls ermitteln, was z. B. fuer Programme mit Zeitschleifen notwendig ist (s. Abschn. 4).

## 3.2.1. Registerstruktur

Der U880-Prozessor hat die in Abb. 3.3 dargestellte Registerstruktur.

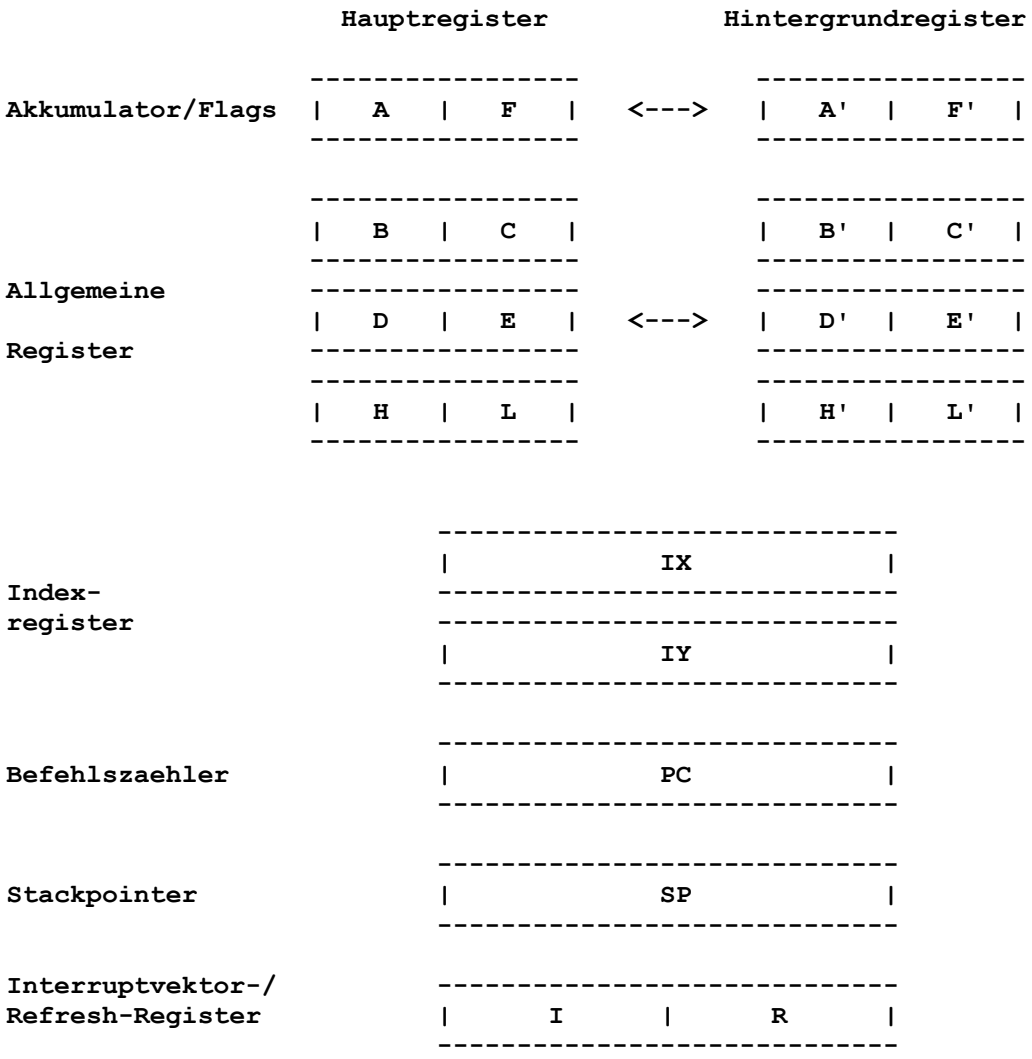



Abb. 3.3:
Registerstruktur

Die einzelnen Register bestehen aus 16-Bit-Speichern, die entsprechend der in der Abbildung dargestellten Aufteilung wahlweise als 8-Bit-Register oder als 16-Bit-Registerpaare benutzt werden koennen. Der Zahlenbereich der 8-Bit-Register geht von 0 bis 255 (bzw. -128 bis +127 bei vorzeichenbehafteten Zahlen) und der Zahlenbereich der 16-Bit-Register von 0 bis 65535 (bzw. -32768 bis +32767).

Nach dem Einschalten des Rechners wird immer der Hauptregistersatz angesprochen. Das Umschalten auf die Hintergrundregister geschieht durch 2 Austauschbefehle (s. Abschn. 3.5.3) getrennt fuer Akkumulator/Flag und Allgemeinregister. Danach beziehen sich alle Befehle bis zum erneuten Umschalten auf die Hintergrundregister.

Akkumulator

Das 8-Bit-Register A dient bei arithmetischen und logischen Befehlen zur Aufnahme eines Operanden. Der andere Operand kommt aus einem anderen Register oder aus dem Speicher. Das Ergebnis der Operation wird wieder im Akkumulator abgelegt, und steht dort fuer eine weitere Verarbeitung zur Verfuegung.

Allgemeine Register

Diese koennen als 8-Bit-Register (B, C, D, E, H, L) oder als 16-Bit-Registerpaare (BC, DE, HL) benutzt werden. Die Register kann man frei als Zwischenspeicher verwenden, jedoch beziehen sich bestimmte Befehle auf einzelne Register. So dient das HL-Registerpaar der indirekten Speicheradressierung. Das DE-Registerpaar kann mit dem Registeraustauschbefehl ueber HL demselben Zweck dienen. Bei einigen Befehlen werden beide Registerpaare als Adressenspeicher fuer Quell- und Zieladressen benutzt (z. B. LDIR).

Das Register B bzw. BC wird vorwiegend als Zaehlregister verwendet.

Befehlszaehler

Das 16-Bit-Register PC enthaelt den aktuellen Befehlszaehlerstand. Beim Einschalten des Rechners wird der Befehlszaehler auf Null gesetzt. Bei Sprung- und Unterprogrammbefehlen wird er mit einem neuen Wert geladen, sonst wird er automatisch jeweils um die Befehlslaenge erhoeht.

Stackpointer

Der Stackpointer SP enthaelt die 16-Bit-Adresse der aktuellen Spitze des Kellerspeichers der CPU. Der Kellerspeicher arbeitet nach dem Prinzip, dass die zuletzt gespeicherten Daten wieder als erste ausgegeben werden ("last in - first out"). Er dient vorwiegend zur Aufnahme der Rueckkehradressen bei Unterprogrammaufrufen und Interruptroutinen. Ausserdem kann er zum Ablegen (PUSH) und Wiedereinlesen (POP) von 16-Bit-Daten aus den Registern verwendet werden. Durch Setzen des Stackpointers im Initialisierungsprogramm des Rechners wird die Lage des fuer den Kellerspeicher zur Verfuegung stehenden Teils des Operativspeichers (RAM-Bereich) festgelegt. Die Groesse ist zunaechst nicht begrenzt. Beim Programmerstellen ist aber zu beachten, dass fuer das jeweilige Programm ausreichend Kellerspeicherplaetze zur Verfuegung stehen.

Der Stackpointer wird beim Abspeichern im Keller um 2 Byte verkleinert und beim Einlesen um 2 Byte erhoeht. Er zeigt immer auf den zuletzt eingespeicherten Wert.

Ein Unterprogrammaufruf kellert den Befehlszaehlerstand ein, ein Rueckkehrbefehl kellert den PC wieder aus. Ausserdem koennen in den Unterprogrammen Daten mit PUSH und POP zwischengespeichert werden. Damit nicht versehentlich Befehlszaehler und Daten verwechselt werden, ist bei der Verwendung von PUSH und POP groesste Sorgfalt auf ein symmetrisches Ein- und Auskellern zu legen.

Beim KC 85/2 und KC 85/3 zeigt der Stackpointer nach dem Einschalten und nach RESET auf die Adresse 01D4H.

Indexregister

Die Indexregister IX und IY werden zur indizierten Adressierung (s. Abschn. 3.2.4) verwendet oder koennen als 16-Bit-Datenregister verwendet werden.

Bei der indizierten Adressierung kann auf einen Speicherbereich in einer Umgebung von -128 bis +127 um den im Register gespeicherten Adressenwert direkt zugegriffen werden.

Das IX-Register des KC 85/2 und KC 85/3 wird fuer System-interruptroutinen genutzt und sollte in Anwenderprogrammen nicht veraendert werden.

Interruptvektorregister

Dieses Register beinhaltet den hoeherwertigen Adressteil der Tabelle fuer die Interruptroutinen. Es hat beim KC 85/2 und KC 85/3 nach Einschalten oder RESET den Wert 1 (s. dazu Abschn. 3.2.3).

Refreshregister

Dieses 7-Bit-Register wird bei jeder Befehlsverarbeitung um 1 erhoeht. Es dient zum Auffrischen der Inhalte der dynamischen RAM-Speicher und ist fuer den Programmierer nicht von Bedeutung.

## 3.2.2. Flag-Bits

Die CPU verfuegt ueber zwei Status-(Flag-)Register (s. Abb. 3.3). Durch Veraenderung einzelner Bits wird ueber die Art des Ergebnisses der letzten Prozessoroperation Auskunft gegeben. Diese Auskunft wird hauptsaechlich dazu benutzt, bedingte Spruenge vorzunehmen, d. h. je nach Ergebnis der Pruefung eines dieser Bedingungsbits die eine oder aber eine andere Aktion durchzufuehren. Die Stellung der einzelnen Bits innerhalb des Flag-Registers F zeigt folgende Tabelle:

```
 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0
---|---|---|---|---|---|---|---
 S | Z | X | H | X |P/V| N | C
```

Hierbei bedeuten:

- C - Uebertragsbit (Carry-Flag)
- N - Additions-/Subtraktionsbit (Add/Subtract-Flag)
- P/V - Paritaets-/Ueberlaufbit (Parity/Overflow-Flag)
- H - Halb-Byte-Ueberlaufbit (Half-Carry-Flag)
- Z - Nullbit (Zero-Flag)
- S - Vorzeichenbit (Sign-Flag)
- X - nicht verwendet

Das Carry-Flag wird gesetzt (=1), wenn bei der Addition zweier Operanden ein Uebertrag von Bit 7 entsteht, sowie wenn bei der Subtraktion ein Bit geborgt werden muss (das Ergebnis negativ wird). Darueber hinaus fungiert das Carry-Flag als Bit-Speicher bei Verschiebe- und Rotationsbefehlen.

Das Zero-Flag wird gesetzt, wenn das Ergebnis einer Operation den Wert Null ergibt. Bei Einzelbitbefehlen dient es zur Uebergabe ausgelesener Bits.

Die Funktion des P/V-Flags haengt von der verwendeten Operation ab. Bei logischen und Verschiebebefehlen wird die Paritaet des Ergebnisses angezeigt (gerade Paritaet: P/V = 1; ungerade Paritaet P/V = 0). Bei arithmetischen Befehlen wird das P/V-Flag als Vorzeichen-Ueberlaufkennzeichnung benutzt; es wird z. B. gesetzt, wenn das Ergebnis zweier Vorzeichenzahlen ausserhalb des zulaessigen Bereiches von -128 bis +127 liegt.

Das Sign-Flag zeigt nach Additionen und Subtraktionen an, ob das Ergebnis positiv ist (S = 0) oder negativ (S = 1).

Das Half-Carry-Flag wirkt wie das Carry-Flag, jedoch wird der Uebertrag von Bit 3 auf Bit 4 angezeigt.

Mit dem Add/Subtract-Flag wird gekennzeichnet, ob als letzter Befehl eine Addition (N = 0) oder eine Subtraktion (N = 1) durchgefuehrt wurde.

Die genaue Reaktion der Flags auf die einzelnen Befehle kann der Befehlsliste (Anhang A) entnommen werden.

## 3.2.3. Interruptsystem

Soll der Rechner auf externe Ereignisse reagieren, so hat man die Moeglichkeit, entweder den betreffenden Eingabekanal staendig abzufragen oder das laufende Programm mittels Interrupt zu unterbrechen, und nach Reaktion auf das Eingabesignal (Interruptprogramm) das urspruengliche Programm fortzusetzen. Die Tastaturabfrage und das Kassetteninterface des KC 85/2 und KC 85/3 arbeiten z. B. mit Interrupt.

Die U880-CPU hat zwei getrennte Signaleingaenge zur Ausloesung von Interrupts:

NMI - nichtmaskierbarer Interrupt hoechster Prioritaet
INT - maskierbarer Interrupt (kann in 3 verschiedenen Interrupt-Modi betrieben werden)

### 3.2.3.1. Nichtmaskierbarer Interrupt (NMI)

Die NMI-Signalzufuehrung kann nicht gesperrt werden. Ein NMI-Signal fuehrt also in jedem Fall zu einer Unterbrechung des laufenden Programms und zu einem erzwungenen Unterprogrammsprung zur Speicheradresse 0066H. An dieser Stelle muss die Interruptbehandlungsroutine vom Programmierer eingetragen sein. Die Interruptroutine muss am Ende mit einem Ruecksprung ins unterbrochene Programm (mit dem Befehl RETN) abgeschlossen werden.

### 3.2.3.2. Maskierbarer Interrupt (INT)

Die INT-Signalzufuehrung kann mit Hilfe der Befehle

EI - eingeschaltet (enable interrupt) und mittels
DI - ausgeschaltet (disable interrupt)

werden (maskieren).

Ist das Interruptsystem nicht freigegeben (DI), so werden INT-Anforderungen ignoriert.

Die Steuerung des INT-Eingangs der CPU erfolgt ueber zwei Merker IFF1 und IFF2 (Interruptflipflops). Der Befehl EI setzt beide auf 1 und DI beide auf 0. IFF2 dient als Merker der Stellung von IFF1 bei der NMI-Behandlung (waehrend der NMI-Behandlung, d. h. bis RETN, ist kein INT moeglich).

Wird nun ein anstehendes INT-Signal erkannt, so fuehrt die CPU hardwaremaessig eine DI-Operation aus, d. h. IFF1 und IFF2 werden auf 0 gesetzt. Sollen weiterhin Interrupts zugelassen werden, so ist spaetestens vor Verlassen der Interruptbehandlungsroutine mit RETI der Befehl EI zu programmieren.

Der maskierbare Interrupt INT kann in 3 Arbeitsweisen betrieben werden, die mit den Befehlen IM 0, IM 1 bzw. IM 2 eingeschaltet werden. Der KC 85/2 und der KC 85/3 arbeiten stets im Interrupt-Mode IM 2. Eine Veraenderung wuerde zum Absturz des CAOS-Betriebssystems fuehren.

**Interrupt-Mode IM 0**

**Wird ein INT-Signal akzeptiert, so wird gleichzeitig ein auf dem Datenbus vom Interruptausloeser (peripherer Baustein) bereitzustellender 1-Byte-Befehl eingelesen und anschliessend ausgefuehrt. Im Normalfall werden dazu die Restart-Befehle**

**RST n (n = 0, 8, 10H, 18H, 20H, 28H, 30H, 38H)**

**verwendet, die einen Unterprogrammaufruf zur Speicheradresse n bewirken, an der die Interruptbehandlungsroutine beginnen muss.**

**Interrupt-Mode IM 1**

**Beim Akzeptieren des INT-Signals wird unabhaengig von den anderen Eingaengen ein Unterprogrammsprung zur Adresse 0038H durchgefuehrt (aehnlich wie NMI).**

**Interrupt-Mode IM 2**

**Diese Betriebsart der CPU ist die leistungsfaehigste und gestattet die individuelle Behandlung unterschiedlicher peripherer Bausteine. Der KC 85/2 und KC 85/3 arbeiten in diesem Modus.**

**Die Interruptbehandlung laeuft nach folgendem Schema ab:**

**In der CPU wird aus dem Wert des Interruptregisters I und dem vom peripheren Baustein auf dem Datenbus bereitzustellenden Interruptvektor IV eine 16-Bit-Adresse gebildet. Das Interruptregister I bildet dabei den hoeherwertigen Adressteil und der Interruptvektor IV den niederwertigen. Von dieser und der folgenden Speicheradresse wird die eigentliche Startadresse der Interruptbehandlungsroutine ausgelesen und ein Unterprogrammsprung dorthin durchgefuehrt. Die Interruptbehandlungsroutine muss mit RETI beendet werden, wobei ggf. zuvor das Interruptsystem mit EI einzuschalten ist.**

**Mit dem Interruptregister I wird also die Lage der Tabelle der Startadressen fuer die Interruptbehandlungsroutinen im Speicher festgelegt. Durch den vom peripheren Baustein bereitgestellten Interruptvektor IV, der geradzahlig sein muss, wird eine der 128 moeglichen Startadressen ausgewaehlt, an der die Interruptroutine beginnt.**

**Beim KC 85/2 und KC 85/3 hat das Interruptregister nach RESET den Wert 1, d. h. die Interrupttabelle steht im Speicher von 0100H bis 01FFH. Von diesen Adressen stehen fuer den Anwender im CAOS-System aber nur die Werte von 01D4H bis 01E3H zur Verfuegung, d. h. es ist Platz fuer 8 Anwenderinterruptadressen. Die Anwenderbausteine muessen die entsprechenden Interrupt-vektoren 0D4H, 0D6H,..., 0E0H oder 0E2H liefern.**

## 3.2.4. Adressierungsarten

Der Befehlssatz des Prozessors beinhaltet 6 verschiedene Adressierungsarten zur Bereitstellung von Register-, Speicher- oder Ein/Ausgabe-Adressen fuer zu spezifizierende Datenwoerter:

### Direkte Adressierung

Der Operationscode beinhaltet vollstaendig die entsprechende Adresse.

```
Z. B.:  LD      A,B
        LD      (0200H),HL
```

### Implizite Adressierung

Der Operationscode bezieht sich fest auf bestimmte Speicherplaetze oder Register.

```
Z. B.:  EXX
        SCF
```

### Unmittelbare Adressierung

Dem Operationscode folgt unmittelbar eine 8- oder 16-Bit-Konstante im Speicher.

```
Z. B.:  LD      A,6
        XOR     20H
```

### Indirekte Adressierung

Die 16-Bit-Adresse befindet sich in einem Registerpaar der CPU. Der Befehl bezieht sich indirekt auf diese Adresse.

```
Z. B.:  LD      A,(HL)
        LDIR
```

### Indizierte Adressierung

Der Operationscode beinhaltet ein Datenbyte (zwischen -128 und +127), dass zum Inhalt des Doppelregisters IX oder IY addiert die vollstaendige Adresse ergibt.

```
Z. B.:  LD      A,(IX+6)
```

## 3.3. Syntax der Assemblersprache

Der hier beschriebene EDAS-Assembler verarbeitet die ZILOG Mnemonik. Dabei gelten allerdings einige Einschraenkungen und Besonderheiten, auf die in diesem Abschnitt an entsprechender Stelle eingegangen wird.

Ein Assemblerprogramm (Quellprogramm) besteht, wie schon erwaehnt, aus einer Folge von Anweisungen (sog. Statements), die zusammen das Anwenderprogramm ergeben. Jede der Quellprogrammzeilen ist aus einem Markenfeld, einem Operationscodefeld (mnemonische Ausdruecke), einem Operandenfeld und einem Kommentarfeld aufgebaut.

Beispiel fuer eine Quellprogrammzeile:

```
Markenfeld  Operationscodefeld  Operandenfeld  Kommentarfeld

START:      LD                  A,6            ;Akku laden
```

Je nach Art der Befehle koennen oder muessen einzelne dieser Felder wegfallen. Die einzelnen Felder muessen durch eine beliebige Anzahl von Leerzeichen oder Tabulatoren voneinander getrennt werden.

Im Quellprogramm wird ausser in Zeichenketten nicht zwischen Gross- und Kleinbuchstaben unterschieden.

### 3.3.1. Marken

Marken sind symbolische Bezugspunkte innerhalb des Programms. Sie werden verwendet, um in einer anderen Anweisung auf den momentanen Befehlszaehlerstand, auf eine andere Marke oder auf eine Konstante Bezug nehmen zu koennen. Eine Marke muss in der ersten Spalte der Programmzeile beginnen und sollte 6 Zeichen nicht ueberschreiten. Das erste Zeichen muss ein Buchstabe sein, als weitere sind Buchstaben, Ziffern, der Punkt und das Unterstreichungszeichen zulaessig. Marken koennen mit Doppelpunkt abgeschlossen werden (muessen aber nicht).

Verboten fuer Marken sind folgende Zeichenketten:

```
Registernamen: A, B, C, D, E, H, L, I, R,
               AF, BC, DE, HL, IX, IY, SP


Flagbedingungen (s. Abschn. 3.3.3):

               C, NC, Z, NZ, M, P, PE, PO
```

### 3.3.2. Operationscodes

Im Operationscodefeld steht eine der ZILOG-Maschinenbefehlsmnemoniken oder eine Assembler-Pseudoanweisungen (s. dazu Abschn. 3.3.5, 3.5 bzw. Anhang A und B).

Das Operationscodefeld beginnt fruehestens in der 2. Spalte der Programmzeile, d. h., wenn das Markenfeld leer ist, muss vor dem Operationscode mindestens ein Trennzeichen (Leerzeichen oder Tabulator) stehen.

### 3.3.3. Operanden

Je nach Art des Operationscodes muss das Operandenfeld entweder leer sein, oder es enthaelt einen oder zwei (durch ein Komma getrennte) Operanden, die eine Adresse (Speicher, Register oder Ein/Ausgabe-Kanal), eine Konstante oder eine Flag-Bedingung repraesentieren. Die Operanden ergaenzen die jeweiligen Anweisungen durch eine Information darueber, mit welchen Parametern die Operation durchzufuehren ist.

Es sind folgende Schluesselwoerter fuer die Operandenfelder reserviert (vgl. dazu Abschn. 3.2.1 und 3.2.2):

- Die Namen der internen Register der CPU, die jeweils den 8-Bit- Inhalt eines dieser Register ansprechen. Die Registernamen sind: A, B, C, D, E, F, H, L, I und R.

- Die Namen der Doppelregister und Registerpaare. Doppelregister sind die 16-Bit-Register IX, IY, SP und PC. Ueber die Registerpaar-Bezeichnungen AF, BC, DE und HL lassen sich die oben bezeichneten 8-Bit-Register der CPU paarweise (als 16-Bit-Woerter) vom Assemblerbefehl ansprechen.

- Die in der CPU integrierten Zweitregister werden in der Assemblersprache mit einem Hochkomma gekennzeichnet. Die Namen sind AF', BC', DE' und HL'. Von diesen Namen ist jedoch lediglich die Kombination AF' als Operand zulaessig (in der Anweisung EX AF,AF').

- Der Zustand der 4 vom Assemblerprogramm testbaren Bedingungs-Bits wird in der Flag-Bedingung wie folgt notiert:

| Bedingungs-Bit-Bezeichnung | Bedingung erfuellt (Flag-Bit = 1) | Bedingung nicht erfuellt (Flag-Bit = 0) |
|---|---|---|
| Uebertrags-Bit (Carry) | C | NC |
| Null-Bit (Zero) | Z | NZ |
| Vorzeichen-Bit (Sign) | M (negativ) | P (positiv) |
| Paritaets-/ Ueberlauf-Bit (Parity/Overflow) | PE (gerade-even) | PO (ungerade-odd) |

(vgl. Abschn. 3.2.2).

Als Konstanten oder Speicheradressen koennen arithmetische Ausdruecke stehen, die durch beliebige Anzahl von Additionen (+) und Subtraktionen (-) von Dezimalzahlen, Hexadezimalzahlen, Marken, Zeichenkettenkonstanten, $ und # bestehen (ohne Leerzeichen dazwischen !). Die Zeichen $ und # sind dabei synonym zur Kennzeichnung des momentanen Befehlszaehlerstandes verwendbar. Hexadezimalzahlen muessen mit einer Ziffer (0 bis 9) beginnen und mit der Kennzeichnung H enden (z. B. 200H, 3FFFH, 0E000H).

Zeichenkettenkonstanten sind in Hochkommas (Apostroph) einzuschliessen. Sie stehen fuer den ASCII-Wert (gemaess Tabelle in Anhang D) der jeweiligen Zeichen (z. B. 'A' steht fuer 41H, 'NO' fuer 4E4FH). Tritt bei der Berechnung der Ausdruecke durch den Assembler ein arithmetischer Ueberlauf auf, so wird dieser nicht beruecksichtigt und geht verloren:

Z. B.: 0E000H+32768 ergibt 6000H

### 3.3.4. Kommentare

Kommentare dienen zu Dokumentationszwecken und zur Erhoehung der Uebersichtlichkeit der Quellprogramme. Sie sind kein funktioneller Bestandteil des Programms und werden beim Assembliervorgang uebersprungen.

Ein Kommentar darf in jeder Spalte der Programmzeile beginnen und er endet mit dem Zeilenende. Das erste Zeichen eines jeden Kommentars muss ein Semikolon (;) sein.

## 3.3.5. Pseudooperationen

Pseudooperationen sind Anweisungen an den Assembler zur Steuerung der Uebersetzung des Quellprogramms. Es gibt daher zu Pseudoanweisungen keinen U880-Maschinencode. Sie sind aber wie ausfuerbare Anweisungen aufgebaut; koennen also mit einer Marke versehen und mit einem Kommentar beendet werden.

Normalerweise beginnt ein Assemblerprogramm mit einer ORG-Pseudoanweisung. Sie legt fest, auf welche Adresse der Beginn des Maschinenprogramms gelegt wird. Wird sie weggelassen, so legt der EDAS-Assembler den Anfang standardmaessig auf den fuer die Speicherablage des Maschinenprogramms reservierten Bereich (3000H ohne RAM-Modul, 6000H bei Verwendung eines 16K RAM-Moduls), sofern der Gesamtspeicherbereich nicht eingeschraenkt wurde.

Fuer Ausdruck kann im Operandenfeld eine Marke, eine hexadezimale Zahl oder eine Verknuepfung stehen.

Der EDAS-Assembler verarbeitet folgende Pseudoanweisungen:

```
        ORG     Ausdruck
```

Setzt den Adresszaehler auf den Wert des Ausdrucks. Ueblicherweise wird damit der Speicherbeginn eines Maschinenprogramms definiert. Die Anweisung kann aber auch benutzt werden, um im Programm freie Speicherplaetze zu reservieren:

```
                ORG     $+Ausdruck
```

reserviert die durch den Ausdruck festgelegte Anzahl von Bytes.

```
Marke   EQU     Ausdruck
```

Weist der Marke den Wert des Ausdrucks zu. Damit kann im Assemblerprogramm mit symbolischen Bezeichnungen anstelle von Konstanten gearbeitet werden.

```
        DEFB    Ausdruck
```

"Definiere Byte" - legt den durch den Ausdruck festgelegten Byte-Wert auf die naechste Speicherstelle. Z. B. DEFB 25

```
        DEFW    Ausdruck
```

"Definiere Wort" - legt den durch den Ausdruck festgelegten 2-Byte-Wert (Wort) auf die naechsten beiden Speicherstellen. Dabei wird zunaechst das niederwertige Byte und danach das hoeherwertige Byte abgelegt.

```
        DEFM    'Text'
```

Legt die ASCII-Werte der durch 'Text' definierten Zeichenkette im Speicher ab. Die Zeichenkette muss in Hochkommas eingeschlossen sein. Z. B.:

```
DEFM    'HALLO'
```

legt die Bytefolge 48H, 41H, 4CH, 4CH, 4FH in den Speicher.
Alle weiteren im orginalen ZILOG-Assembler moeglichen Pseudooperationen (auch END) fuehren zu einer Fehlermitteilung des Assemblers.

Innerhalb eines in Hochkommas (') eingeschlossenen Textes darf statt der Space-Taste nicht die CUR-Taste verwendet werden, da es sonst beim Uebersetzen des Quelltextes zu Fehlern fuehrt.

## 3.4. Abarbeitung des Assemblers

Der EDAS-Assembler wird vom EDAS-Editor-Menue ("*" ist Bereitschaftszeichen) mit dem Kommando ASM aufgerufen:

*ASM <ENTER>

Ist dabei kein Quelltext im Speicher vorhanden, so wird der Assembler sofort wieder verlassen und es wird in die Editor-Kommandoschleife zurueckgekehrt.

Zum vollstaendigen Assemblieren gehoeren zwei Assemblerlaeufe (Paesse), die normalerweise hintereinander ablaufen. Im ersten Pass wird die Markentabelle erzeugt, im zweiten Pass der Maschinencode und eine Druckliste. Ist im Quelltext keine ORG-Anweisung vorhanden, wurde das Maschinenprogramm automatisch ab der Adresse 3000H bei 16K RAM bzw. 6000H bei 32K RAM (mit 16K RAM-Modul) abgelegt, sofern der Gesamtspeicherbereich nicht eingeschraenkt wurde.

Die Ablage der Markentabelle im Speicher erfolgt von der beim EDAS-Aufruf anzugebenden Speicherendadresse rueckwaerts (als Stack).

### 3.4.1. Aufrufbedingungen

Der Assembler verlangt nach Aufruf zunaechst die Eingabe von Options zur Steuerung des Uebersetzungsvorgangs. Es erscheint die Ausschrift:

```
OPTIONS(+,2,B,L,O,P,S)? :_
```

Durch Eingabe der entsprechenden Buchstaben bzw. Zeichen in beliebiger Reihenfolge und an beliebigen Stelle im Cursorfenster koennen die Options angegeben werden. Dabei bedeutet:

+ - Erhalten einer schon vorhandenen Markentabelle und Hinzufuegen der neuen Marken zu dieser Tabelle.

2 - Es wird nur der zweite Pass durchgefuehrt, d. h. es wird keine (neue) Markentabelle erzeugt.

B (Brief - Kurzform) - Die ueber das Ausgabegeraet kommenden Listen- bzw. Fehlerzeilen werden auf eine Laenge von 39 Zeichen begrenzt, um z. B. die Ausgabe ueber den Bildschirm uebersichtlicher zu gestalten.

L (Listing) - Erzeugung einer Assemblerdruckliste und deren Ausgabe ueber das Ausgabegeraet (s. P). Die Ausgabe kann mit der Taste STOP unterbrochen und mit BREAK abgebrochen werden. Im Drucklisting ist in der 3. bis 6. Spalte die Speicheradresse und in der 8. bis 16. Spalte der erzeugte Maschinencode enthalten. Danach folgt die Quellprogrammzeile.

O (Output) - Erzeugung des Maschinencodes und Ablage im Speicher. Dabei ist Vorsicht geboten, wenn ORG-Anweisungen im Quelltext stehen. Die Ueberpruefung, dass von EDAS benutzte Speicherbereiche nicht ueberschrieben werden, muss der Benutzer selbst vornehmen. Ggf. muss der EDAS-Arbeitsbereich beim EDAS-Aufruf entsprechend eingeschraenkt werden. Eine ORG-Anweisung <3000H fuehrt zu einem fehlerhaften Maschinencode. Werden uebersetzte Programme auf bereits durch andere Programme belegte Speicherbereiche ausgegeben muss am Programmmanfang eine entsprechende ORG-Anweisung stehen.

P (Printer) - Fuer die Dauer des Assembliervorgangs wird das Ausgabegeraet auf den Anwenderausgabekanal 1 eingestellt. Damit ist es z. B. moeglich, das Assemblerlisting (L und P gleichzeitig) auf einen Drucker (siehe Pkt. 6) auszugeben.

S (Save) - Nach Abarbeitung der uebrigen Assembleroptions wird ein weiterer (dritter) Pass zur Kassettenausgabe des Maschinencodes gestartet. Dabei wird zunaechst ein Programmname (bis zu 8 Zeichen) abgefordert. Dieser wird intern mit dem Dateityp COM ergaenzt. Der Kassettenpass kann mit BREAK abgebrochen werden.

Bei der Kassettenausgabe ist zu beachten, dass der erzeugte Maschinencode aus nur einem zusammenhaengenden Speicherbereich bestehen darf (es darf nur hoechstens eine ORG-Anweisung und zwar am Anfang des Quelltextes stehen). Nur unter dieser Voraussetzung kann der gespeicherte Maschinencode mit dem LOAD-Befehl des Monitors korrekt geladen werden.

Nachdem die Assembler-Options eingegeben wurden (die Zeile ist mit <ENTER> abzuschliessen), beginnt der Assembliervorgang. Die Beendigung des ersten Passes wird am Bildschirm durch die Ausschrift

END PASS 1

angezeigt. Bei der Uebersetzung ermittelte Fehler werden ueber das Ausgabegeraet (Bildschirm oder Drucker, je nachdem, ob P als Option eingegeben wurde) durch den Ausdruck der fehlerhaften Zeile und einer Fehlernummer in der ersten Spalte ausgegeben. Nach Beendigung des zweiten Assemblerpasses erfolgt die Mitteilung

ERRORS: nnnn (nnnn ist 4-stellige Dezimalzahl)

mit der Ausgabe der Anzahl der im Quellprogramm enthaltenen Fehler.

Bei der Programmuebersetzung ist die folgende Vorgehensweise zweckmaessig:

Zunaechst werden keine Options angegeben, d. h. es wird sofort <ENTER> gedrueckt. Damit wird das Quellprogramm nur auf syntaktische Fehler getestet, und nur die fehlerhaften Zeilen werden am Bildschirm angezeigt. Auf Grund der Programmgroesse des Editor/Assemblers kann keine umfassende Syntaxpruefung erfolgen, d.h. Fehler im Operandenfeld werden nur bedingt erkannt. Beim Vorhandensein von Fehlern sind diese mit dem Editor zu beseitigen und anschliessend ist erneut ohne Options zu uebersetzen. Hat man schliesslich ein syntaktisch korrektes Quellprogramm (ERRORS: 0000 wird angezeigt), so sollte man diesen Quelltext zunaechst mit SAVE auf Kassette abspeichern. Danach kann die Assemblierung mit der Erzeugung des Maschinencodes erfolgen, der mit "O" im Speicher abgelegt oder mit "S" auf Kassette ausgegeben wird. Wenn gewuenscht, kann noch eine Druckliste ueber Bildschirm (L) oder auf den Drucker (L P) ausgegeben werden.

Zum Testen des Maschinenprogramms ist EDAS ueber den EXIT-Befehl zu verlassen. Werden die EDAS-Speicherbereiche nicht veraendert, so steht nach Aufruf von REEDAS der Programmtext noch zur Verfuegung.

Mit dem Assembler ist es auch moeglich, Programmteile, die nicht gleichzeitig im Speicher Platz haben, zu verbinden (zu "Linken") Dazu kann man wie folgt vorgehen:

Die Quellprogramme werden nacheinander eingelesen und assembliert, wobei die Option "+" mit anzugeben ist. Angezeigte Markenfehler "3" sind hierbei zu ignorieren. Auf diese Weise wird eine Gesamtmarkentabelle aller zu verbindenden Programme erzeugt. Anschliessend werden die Programme erneut geladen, unter Verwendung der Options "+" und "2" assembliert und der Maschinencode erzeugt.

## 3.4.2. Fehlermitteilungen

**Fehlerhafte Quellprogrammzeilen werden beim Assemblieren in der ersten Spalte mit einer Fehlernummer (gefolgt von einem "*") versehen und ueber das Ausgabegeraet (Bildschirm oder Drucker) ausgedruckt.**

**Die Fehlernummern haben folgende Bedeutungen:**

**"1" - Es fehlt ein Semikolon in der Kommentarzeile**
**"2" - Marke mehrfach definiert**
**"3" - Marke nicht definiert**
**"4" - Falsche Mnemonik (Befehl unbekannt)**
**"5" - Falsches Zahlenformat**
**"6" - Operandenfehler bei JR, IX oder IY**
**(ausserhalb des Bereiches -128,..., +127)**
**"7" - Keine Marke in EQU-Anweisung**
**"8" - Hochkomma fehlt**
**"9" - Operandenfehler bei EX-Befehl**
**"A" - Falsche Flag-Bedingung**
**(zulaessig sind Z, NZ, C, NC, PE, PO, M, P)**
**"B" - Plus, Minus oder Komma im Operanden fehlt**

**Dabei ist zu beachten, dass der Fehler "2" nicht in den Programmzeilen angezeigt wird, in denen die Marke definiert ist (im Markenfeld steht), sondern in den Zeilen, in denen die Marke als Operand auftritt.**

**Ausserdem sei darauf hingewiesen, dass vom EDAS-Assembler keine vollstaendige Analyse aller moeglichen Fehlerquellen durchgefuehrt wird. (Der EDAS-Assembler umfasst nur 2K Byte Speicherplatz!) Die gaenigsten Fehler werden jedoch identifiziert.**

## 3.5. Befehlssatz des Assemblers

In diesem Abschnitt wird der syntaktische Aufbau und die Wirkungsweise der einzelnen Assemblerbefehle beschrieben. Der U880-Prozessor verfuegt ueber einen sehr umfangreichen Befehlssatz, der nicht nur 8-Bit-, sondern auch 16-Bit-Befehle umfasst. Mit der grossen Befehlsliste stehen dem Programmierer meist verschiedene Moeglichkeiten zur Verfuegung, ein und dasselbe Problem mehr oder weniger elegant zu loesen. Viel haengt dabei von der Uebung und den Erfahrungen des Programmerstellers ab.

Besonders zu beachten ist, dass z. B. verschiedene Assemblerbefehle nur fuer spezielle Operanden gelten. Die Gueltigkeit des verwendeten Befehls ist also anhand der in diesem Abschnitt angegebenen Uebersicht ueber alle erlaubten Befehle sorgfaeltig zu ueberpruefen.

Der Befehlssatz laesst sich in folgende Gruppen einteilen:

- Lade- und Austauschbefehle
- Blocktransfer- und Blocksuchbefehle
- Arithmetik- und Logikbefehle
- Programmverzweigungsbefehle
- Unterprogrammbefehle
- Rotations- und Verschiebebefehle
- Einzelbitbefehle
- Steuerbefehle
- Ein- und Ausgabebefehle

Die Ladebefehle tranportieren Daten zwischen den Registern oder zwischen Registern und dem Speicher. Registeraustauschbefehle erschliessen dem Programmierer die Hintergrundregister der CPU, Blocktransferbefehle transportieren ganze Datenbloecke zwischen verschiedenen Speicherbereichen. Mit einem Blocksuchbefehl kann man einen Speicherbereich nach einem Datenbyte durchmustern.

Die Arithmetik- und Logikbefehle arbeiten mit einem Akkumulator (A, HL, IX oder IY) als ersten Operanden und einem Register, Speicherstelle oder Konstanten als zweiten Operanden. Das Ergebnis der Operation (z. B. einer Addition) steht wieder im Akkumulator und es werden die entsprechenden Flag-Bits gesetzt.
Die Programmverzweigungsbefehle gestatten es, in Abhaengigkeit von den Flag-Bits Spruenge (relative oder absolute) zu anderen Programmteilen durchzufuehren.

Im Unterschied zu den Sprungbefehlen bieten die Unterprogrammbefehle die Moeglichkeit, nach Abarbeitung des aufgerufenen Programmstuecks wieder das urspruengliche Programm fortzusetzen.

Die Rotations-, Verschiebe- und Einzelbitbefehle verwendet man, um den Inhalt einzelner Bits der Register oder von Speicherstellen abzufragen oder zu veraendern.

Die Steuerbefehle dienen zur Beeinflussung des Interruptsystems der CPU.

Mit den Ein- und Ausgabebefehlen kann man spezielle Toradressen zur Kommunikation mit externen Bausteinen (PIO, SIO, CTC) ansprechen und Daten ein- oder ausgeben.

Der U880-Prozessor hat variable Befehlswortlaenge (1 bis 4 Byte). Die Byteanzahl und die Kodierung der einzelnen Befehle kann man der Befehlscode-Tabelle im Anhang A entnehmen. Der Befehlscode erscheint auch im Drucklisting des Assemblers (Option L).

In den Abschnitten 3.5.1 bis 3.5.12 werden alle Assemblerbefehle und deren Wirkungsweise erlaeutert. Dabei werden die folgenden Abkuerzungen verwendet:

Abkuerzungsverzeichnis:

r, r' - Einfachregister A, B, C, D, E, H, L
dd - Doppelregister BC, DE, HL, SP
qq - Doppelregister AF, BC, DE, HL
pp - Doppelregister BC, DE, SP
n - 8-Bit-Konstante
nn - 16-Bit-Konstante, Adresse
d - Verschiebung bei Adressierung ueber Indexregister, im Bereich -128 <= d <= +127
b - Bit, das in den Einzelbitbefehlen behandelt werden soll, 0 <= b <= 7
m - Inhalt des durch HL adressierten Speicherplatzes. Fuer m kann sowohl (HL) als auch M stehen. Register L beinhaltet dabei die niederwertigen 8 Bits und Register H die hoeherwertigen 8 Bits der Adresse des Speichplatzes.
p - Einer der Werte 00H,08H,10H,18H,20H,28H,30H,38H
CY - Carry-Flag
T - Anzahl der Taktzyklen des Befehls

Bei Befehlen, die Flag-Bedingungen fuer Programmspruenge auswerten sind 2 Taktzyklen angegeben. Dabei bezieht sich die zweite Zahl auf den Fall, dass kein Sprung durchgefuehrt und mit dem naechsten im Speicher stehenden Befehl weitergearbeitet wird.

T*571 Nanosekunden ist die Befehlsausfuehrungszeit auf dem KC 85/2 und KC 85/3.

**Flag-Beeinflussung der Befehle (s. 3.2.2):**

In der letzten Spalte der folgenden Befehlsuebersicht ist fuer die einzelnen Flag-Bits C, N, P/V, H, Z, S deren Wert nach der Ausfuehrung des Befehls angegeben. Dabei bedeutet:

1 - gesetzt (=1)
0 - zurueckgesetzt (=0)
\- - unveraendert
* - entsprechend dem Ergebnis der Operation,d. h.:
gesetzt wenn erfuellt, zurueckgesetzt wenn nicht erfuellt
x - unbestimmt
V - Overflow-Funktion
P - Parity-Funktion
F - Inhalt des Interrupt-Flip-Flops IFF2 (vgl. Abschn 3.2.3.2)

### 3.5.1. 8-Bit-Ladebefehle

Die Ladebefehle transportieren 8-Bit-Daten intern zwischen Registern oder zwischen Registern und dem Speicher. Dabei steht im Operandenfeld als erstes der Zielspeicherplatz und danach (durch Komma getrennt) der Quellspeicherplatz. Der Inhalt des Quellspeicherplatzes wird bei diesen Befehlen nicht veraendert.

| Mnemonik | T | Wirkungsweise des Befehls | SZHPNC V |
|---|---|---|---|
| LD r,r' | 4 | Inhalt des Registers r' wird in das Register r umgespeichert | ------ |
| LD r,n | 7 | Die 8-Bit-Konstante n wird in das Register r geladen | ------ |
| LD r,m | 7 | Inhalt des durch Registerpaar HL adressierten Speicherplatzes m wird in das Register r geladen. Fuer m ist sowohl die Schreibweise (HL) als auch M zulaessig | ------ |
| LD r,(IX+d) | 19 | Inhalt des durch Register IX (bzw. IY) plus Verschiebung d adressierten Speicherplatzes wird in Register r geladen | ------ |
| LD r,(IY+d) | 19 | | ------ |
| LD m,r | 7 | Transportiert ein Byte aus Register r auf den durch Registerpaar HL adressierten Speicherplatz m. Fuer m kann (HL) oder M stehen | ------ |
| LD (IX+d),r | 19 | Bringt Daten aus dem Register r an den Speicherplatz, dessen Adresse durch den Inhalt des IX (bzw. IY)-Registers plus Verschiebung d spezifiziert ist | ------ |
| LD (IY+d),r | 19 | | ------ |

| | | | |
|---|---|---|---|
| LD m,n | 10 | Bewirkt den Transport der Konstanten n an den durch Registerpaar HL adressierten Speicherplatz m. Fuer m kann (HL) oder m stehen | ------ |
| LD (IX+d),n | 19 | Bewirkt den Transport der Konstanten n an den Speicherplatz, dessen Adresse durch den Inhalt des IX (bzw. IY)-Registers plus Verschiebung d spezifiziert ist | ------ |
| LD (IY+d),n | 19 | | ------ |
| LD A,(BC) | 7 | Der Inhalt des durch Registerpaar BC adressierten Speicherplatzes wird in den Akkumulator (A-Register) geladen | ------ |
| LD A,(DE) | 7 | Der Inhalt des durch Registerpaar DE adressierten Speicherplatzes wird in den Akkumulator (A-Register) geladen | ------ |
| LD A,(nn) | 13 | Der Inhalt des Speicherplatzes nn wird in den Akkumulator (A-Register) geladen | ------ |
| LD (nn),A | 13 | Der Inhalt des Akkumulators (A-Register) wird auf den Speicherplatz nn geladen | ------ |
| LD (BC),A | 7 | Der Inhalt des Akkumulators (A-Register) wird auf den Speicherplatz geladen, dessen Adresse im Registerpaar BC steht | ------ |
| LD (DE),A | 7 | Der Inhalt des Akkumulators (A-Register) wird auf den Speicherplatz geladen, dessen Adresse im Registerpaar DE steht | ------ |
| LD A,I | 9 | Der Inhalt des Interruptregisters I wird in den Akkumulator (A-Register) geladen | **0F0- |
| LD A,R | 9 | Der Inhalt des Refresh-Registers I wird in den Akkumulator (A-Register) geladen | **0F0- |
| LD I,A | 9 | Der Inhalt des Akkumulators (A-Register) wird in das Interruptregister I geladen | ------ |
| LD R,A | 9 | Der Inhalt des Akkumulators (A-Register) wird in das Refresh-Register R geladen | ------ |

## 3.5.2. 16-Bit-Ladebefehle

Transportieren 16-Bit-Daten intern zwischen den Registern oder zwischen Registern und dem Speicher. Der Inhalt des Quellspeichers wird dabei nicht veraendert.
Spezielle 16-Bit-Befehle sind die PUSH- und POP-Befehle. Mit Ihnen werden 16-Bit-Daten aus Doppelregistern in den Kellerspeicher gebracht bzw. zurueck in die Doppelregister geholt. Man verwendet sie haeufig zum Retten von Registerinhalten z. B. in Unterprogrammen:

```
UP:     PUSH    HL
        PUSH    DE
        PUSH    BC
          ...       ) Unterprogramm-
          ...       ) Befehle
          ...       )
        POP     BC
        POP     DE
        POP     HL
        RET
```

Nach Beendigung des Unterprogramms besitzen die Register BC, DE, HL die gleichen Inhalte wie vor dem Aufruf.
Zu beachten ist, dass 16-Bit-Daten im Speicher auf 2 Byte mit aufeinanderfolgenden Adressen nn und nn+1 abgespeichert werden. Die unteren 8 Bit (niederwertiger Teil) stehen auf der Adresse nn und die oberen 8 Bit (hoeherwertiger Teil) auf der Adresse nn+1. Alle 16-Bit-Woerter (auch in Maschinencodes) werden vom Prozessor grundsaetzlich auf diese Weise (erst niederwertiger, dann hoeherwertiger Teil) abgespeichert.

| Mnemonik | T | Wirkungsweise des Befehls | SZHPNC V |
|---|---|---|---|
| LD dd,nn | 10 | Die 16-Bit-Konstante nn wird in das Doppelregister dd geladen | |
| LD IX,nn | 14 | Die 16-Bit-Konstante nn wird in das Indexregister IX (bzw. IY) geladen | |
| LD IY,nn | 14 | | |
| LD HL,(nn) | 16 | Inhalt der Speicherplaetze nn und nn+1 wird in das Doppelregister HL geladen<br>Inhalt von nn+1 --> Register H<br>Inhalt von nn --> Register L | |
| LD pp,(nn) | 20 | Inhalt der Speicherplaetze nn und nn+1 wird in das Doppelregister pp geladen<br>Inhalt von nn+1 --> hoeherwertiges Reg<br>Inhalt von nn --> niederwertiges Reg | |
| LD IX,(nn) | 20 | Inhalt der Speicherplaetze nn und nn+1 wird in das Indexregister IX (bzw. IY) geladen<br>Inhalt von nn+1 --> hoeherwert. Teil<br>Inhalt von nn --> niederwert. Teil | |
| LD IY,(nn) | 20 | | |

```
            |  |                                          |
LD   (nn),HL|16| Inhalt des Doppelregisters HL wird auf  |------
            |  | die Adressen nn und nn+1 transportiert  |
            |  | Register H       -->  Inhalt von nn+1   |
            |  | Register L       -->  Inhalt von nn     |
            |  |                                          |
LD   (nn),pp|20| Inhalt des Doppelregisters pp wird auf  |------
            |  | die Adressen nn und nn+1 transportiert  |
            |  | hoeherwertiges Reg -->  Inhalt von nn+1 |
            |  | niederwertiges Reg -->  Inhalt von nn   |
            |  |                                          |
LD   (nn),IX|20| Inhalt des Indexregisters IX (bzw. IY)  |------
LD   (nn),IY|20| wird auf die Adressen nn und nn+1 trans-|------
            |  | portiert                                |
            |  | hoeherwert. Teil   -->  Inhalt von nn+1 |
            |  | niederwert. Teil   -->  Inhalt von nn   |
            |  |                                          |
LD   SP,HL  | 6| Inhalt des Doppelregisters HL wird in   |------
            |  | den Stackpointer (Kellerzeiger) trans-  |
            |  | portiert                                |
            |  |                                          |
LD   SP,IX  |10| Inhalt des Indexregisters IX (bzw. IY)  |------
LD   SP,IY  |10| wird in den Stackpointer (Kellerzeiger) |------
            |  | transportiert                           |
            |  |                                          |
PUSH   qq   |11| Inhalt des Doppelregisters qq wird in   |------
            |  | Kellerspeicher (Stack) transportiert:   |
            |  | erniedrigen von SP                      |
            |  | hoeherwertiges Reg --> Inhalt von SP-Adr|
            |  | erniedrigen von SP                      |
            |  | niederwertiges Reg --> Inhalt von SP-Adr|
            |  |                                          |
PUSH   IX   |15| Inhalt des Indexregisters IX (bzw. IY)  |------
PUSH   IY   |15| wird in den Kellerspeicher (Stack)      |------
            |  | transportiert:                          |
            |  | erniedrigen von SP                      |
            |  | hoeherwert. Teil  -->  Inhalt von SP-Adr|
            |  | erniedrigen von SP                      |
            |  | niederwert. Teil  -->  Inhalt von SP-Adr|
            |  |                                          |
POP   qq    |10| Inhalt der von Stackpointer SP adres-   |------
            |  | sierten 2 Speicherstellen wird in das   |
            |  | Doppelregister qq uebertragen:          |
            |  | Inhalt von SP-Adr --> niederwertiges Reg|
            |  | erhoehen von SP                         |
            |  | Inhalt von SP-Adr --> hoeherwertiges Reg|
            |  | erhoehen von SP                         |
            |  |                                          |
POP   IX    |14| Inhalt der von Stackpointer SP adres-   |------
POP   IY    |14| sierten 2 Speicherstellen wird in das   |------
            |  | Indexregister IX (bzw. IY) uebertragen: |
            |  | Inhalt von SP-Adr --> niederwertig. Teil|
            |  | erhoehen von SP                         |
            |  | Inhalt von SP-Adr --> hoeherwertig. Teil|
            |  | erhoehen von SP                         |
            |  |                                          |
```

## 3.5.3. Registeraustauschbefehle

Diese Befehle dienen dem schnellen Austausch von Doppelregisterinhalten und erschliessen dem Programmierer die Hintergrundregister.
Zum Beispiel

```
UP:     EXX
        ...  ) Unterprogramm-
        ...  ) Befehle
        ...  )
        EXX
        RET
```

rettet die Registerinhalte BC, DE, HL fuer das Hauptprogramm. Im Unterschied zur Verwendung von PUSH und POP gehen die Inhalte der Hintergrundregister ggf. im Unterprogramm verloren.

Durch den Befehl EX DE,HL kann fuer in DE enthaltene Adressen auch die indirekte Adressierung (ueber HL) verwendet werden.

| Mnemonik | T | Wirkungsweise des Befehls | SZHPNC V |
|---|---|---|---|
| EX DE,HL | 4 | Die 16-Bit-Inhalte der Registerpaare DE und HL werden ausgetauscht<br>DE <---> HL | ------ |
| EX AF,AF' | 4 | Die 16-Bit-Inhalte der Registerpaare AF und AF' werden ausgetauscht<br>AF <---> AF' | ------ |
| EXX | 4 | Die 16-Bit-Inhalte der nachstehenden Registerpaare werden ausgetauscht<br>BC <---> BC'<br>DE <---> DE'<br>HL <---> HL' | ------ |
| EX (SP),HL | 19 | Der Inhalt des Registers L wird gegen den Inhalt der Speicherstelle ausgetauscht, die durch den Inhalt des Stackpointers SP adressiert ist. Der Inhalt des Registers H wird gegen den Inhalt der Speicherstelle ausgetauscht, die durch den Inhalt des Stackpointers SP plus 1 adressiert ist.<br>H <---> (SP+1)<br>L <---> (SP) | ------ |

| | | | |
|---|---|---|---|
| EX (SP),IX | 23 | Der niederwertige Teil des Indexregisters IX (bzw. IY) wird gegen den Inhalt der Speicherstelle ausgetauscht, die durch den Inhalt des Stackpointers SP adressiert ist. Der hoeherwertige Teil des Indexregisters IX (bzw. IY) wird gegen den Inhalt der Speicherstelle ausgetauscht, die durch den Inhalt des Stackpointers SP plus 1 adressiert ist.<br>hoeherwertiger Teil <---> (SP+1)<br>niederwertiger Teil <---> (SP) | ------ |

## 3.5.4. Blocktransfer- und -suchbefehle

Mit einem einzigen Befehl koennen beliebig grosse Datenmengen im Speicher verschoben werden, bzw. es kann in einem Speicherbereich nach einem Datenbyte gesucht werden. Die Suche wird dabei beendet, wenn das Byte gefunden wurde.
Beispiel: Die Befehlsfolge

```
LD      HL,0E000H
LD      DE,2000H
LD      BC,800H
LDIR
```

verschiebt einem Teil des CAOS-Betriebssystems in den RAM-Bereich.

| Mnemonik | T | Wirkungsweise des Befehls | SZHPNC<br>V |
|---|---|---|---|
| LDI | 16 | Transport eines Datenbytes von der Speicherstelle, die durch das Registerpaar HL adressiert wird nach der Speicherstelle, die durch das Registerpaar DE adressiert wird. Die Register DE und HL werden um 1 erhoeht, und das Register BC wird um 1 vermindert.<br>BC = 0 --> PV = 0<br>BC <> 0 --> PV = 1 | --0*0- |
| LDIR | 21<br>16 | Transport mehrerer Datenbytes ab der Speicherstelle, die durch das Registerpaar HL adressiert wird nach der Speicherstelle, die durch das Registerpaar DE adressiert wird. Bie Byteanzahl ist im Registerpaar BC enthalten. Nach jeder Byteuebertragung wird der Inhalt von HL und von DE um 1 erhoeht und BC um 1 vermindert. Die Uebertragung endet, wenn der Inhalt von BC Null ist. | --000- |

| | | | |
|---|---|---|---|
| LDD | 16 | Der Befehl wirkt wie LDI, nur werden die Register DE und HL um 1 vermindert | --0*0- |
| LDDR | 21<br>16 | Der Befehl wirkt wie LDIR, nur werden die Register DE und HL um 1 vermindert | --000- |
| CPI | 16 | Vergleich des Inhalts des durch HL adressierten Speicherplatzes mit dem Inhalt des Akkumulators (A-Register):<br>A = (HL) --> Z = 1<br>A <> (HL) --> Z = 0<br>Anschliessend wird das Register HL um 1 erhoeht und das Registerpaar BC um 1 vermindert. Das Registerpaar BC kann als Bytezaehler arbeiten:<br>BC = 0 --> PV = 0<br>BC <> 0 --> PV = 1 | ****1- |
| CPIR | 21<br>16 | Vergleich des Inhalts des Akkumulators (A-Register) mit dem Inhalt eines adressierten Speicherbereiches. Die Anfangsadresse des Bereiches ist im Registerpaar HL enthalten, Die Laenge des Bereiches im Registerpaar BC. Der Vergleich endet, wenn A = (HL) oder wenn BC = 0 ist. Der Befehl ist also ein wiederholter Aufruf von CPI mit den Abbruchbedingungen Z = 1 oder PV = 0. | ****1- |
| CPD | 16 | Der Befehl wirkt wie CPI, nur das Register HL wird vermindert | ****1- |
| CPDR | 21<br>16 | Der Befehl wirkt wie CPIR, nur das Register HL wird vermindert | ****1- |

## 3.5.5. 8-Bit-Arithmetik- und -Logikbefehle

Diese Befehle arbeiten mit Daten, die sich im Akkumulator (Register A als ersten Operanden) und mit Daten in anderen Registern oder auf Speicherplaetzen (als zweiten Operanden) befinden. Das Ergebnis dieser Operationen wird im Akkumulator abgelegt.

| Mnemonik | T | Wirkungsweise des Befehls | SZHPNC V |
|---|---|---|---|
| ADD A,r | 4 | Der Inhalt von Register r wird zum Akkumulatorinhalt addiert | ***V0* |
| ADD A,m | 7 | Der Inhalt des durch das Registerpaar HL adressierten Speicherplatzes wird zum Inhalt des Akkumulators addiert | ***V0* |
| ADD A,n | 7 | Die Konstante n wird zum Inhalt des Akkumulators addiert | ***V0* |
| ADD A,(IX+d) | 19 | Der Inhalt des durch das Indexregister IX (bzw. IY) plus Adressverschiebung d adressierten Speicherplatzes wird zum Inhalt des Akkumulators addiert. Das Ergebnis steht im Akkumulator. | ***V0* |
| ADD A,(IY+d) | 19 | | ***V0* |
| ADC A,r | 4 | Der Inhalt von Register r plus Carry-Flag wird zum Akkumulatorinhalt addiert | ***V0* |
| ADC A,m | 7 | Der Inhalt des durch das Registerpaar HL adressierten Speicherplatzes plus Carry-Flag wird zum Inhalt des Akkumulators addiert | ***V0* |
| ADC A,n | 7 | Die Konstante n plus Carry-Flag wird zum Inhalt des Akkumulators addiert | ***V0* |
| ADC A,(IX+d) | 19 | Der Inhalt des durch das Indexregister IX (bzw. IY) plus Adressverschiebung d adressierten Speicherplatzes plus Carry-Flag wird zum Inhalt des Akkumulators addiert. Das Ergebnis steht im Akkumulator. | ***V0* |
| ADC A,(IY+d) | 19 | | ***V0* |
| SUB r | 4 | Der Inhalt von Register r wird vom Akkumulatorinhalt subrahiert | ***V1* |
| SUB m | 7 | Der Inhalt des durch das Registerpaar HL adressierten Speicherplatzes wird vom Inhalt des Akkumulators subtrahiert | ***V1* |
| SUB n | 7 | Die Konstante n wird vom Inhalt des Akkumulators subtrahiert | ***V1* |
| SUB (IX+d) | 19 | Der Inhalt des durch das Indexregister IX (bzw. IY) plus Adressverschiebung d | ***V1* |
| SUB (IY+d) | 19 | | ***V1* |

```
             |   | adressierten Speicherplatzes wird vom   |
             |   | Inhalt des Akkumulators subtrahiert. Das|
             |   | Ergebnis steht im Akkumulator.          |
             |   |                                         |
SBC   A,r    | 4 | Der Inhalt von Register r plus Carry-   |***V1*
             |   | Flag wird vom Akkumulatorinhalt subtra- |
             |   | hiert                                   |
             |   |                                         |
SBC   A,m    | 7 | Der Inhalt des durch das Registerpaar HL|***V1*
             |   | adressierten Speicherplatzes plus Carry-|
             |   | Flag wird vom Inhalt des Akkumulators   |
             |   | subtrahiert                             |
             |   |                                         |
SBC   A,n    | 7 | Die Konstante n plus Carry-Flag wird vom|***V1*
             |   | Inhalt des Akkumulators subtrahiert     |
             |   |                                         |
SBC A,(IX+d) |19 | Der Inhalt des durch das Indexregister  |***V1*
SBC A,(IY+d) |19 | IX (bzw. IY) plus Adressverschiebung d  |***V1*
             |   | adressierten Speicherplatzes plus Carry-|
             |   | Flag wird vom Inhalt des Akkumulators   |
             |   | subtrahiert. Das Ergebnis steht im Akku-|
             |   | mulator.                                |
             |   |                                         |
AND   r      | 4 | Logische UND-Verknuepfung eines Regi-   |**1P00
AND   m      | 7 | sters, Speicherbytes oder Konstanten mit|**1P00
AND   n      | 7 | dem Akkumulator. Das spezifizierte Byte |**1P00
AND   (IX+d) |19 | wird bitweise mit dem Inhalt des Akkumu-|**1P00
AND   (IY+d) |19 | lators konjunktiv verknuepft. Das logi- |**1P00
             |   | sche UND ist nur dann 1, wenn beide Bits|
             |   | 1 sind.                                 |
             |   |                                         |
OR    r      | 4 | Logische ODER-Verknuepfung eines Regi-  |**0P00
OR    m      | 7 | sters, Speicherbytes oder Konstanten mit|**0P00
OR    n      | 7 | dem Akkumulator. Das spezifizierte Byte |**0P00
OR    (IX+d) |19 | wird bitweise mit dem Inhalt des Akkumu-|**0P00
OR    (IY+d) |19 | lators disjunktiv verknuepft. Das logi- |**0P00
             |   | sche ODER ist nur dann 0, wenn beide    |
             |   | Bits 0 sind.                            |
             |   |                                         |
XOR   r      | 4 | Exklusiv-ODER-Verknuepfung eines Regi-  |**0P00
XOR   m      | 7 | sters, Speicherbytes oder Konstanten mit|**0P00
XOR   n      | 7 | dem Akkumulator. Das spezifizierte Byte |**0P00
XOR   (IX+d) |19 | wird bitweise mit dem Inhalt des Akkumu-|**0P00
XOR   (IY+d) |19 | lators exklusiv verknuepft. Das Exklu-  |**0P00
             |   | siv-ODER ist dann 1, wenn ein Bit = 1   |
             |   | und das andere Bit = 0 ist.             |
             |   |                                         |
CP    r      | 4 | Der Inhalt eines Registers, eines Spei- |***V1*
CP    m      | 7 | cherbytes oder eine Konstante wird mit  |***V1*
CP    n      | 7 | dem Akkumulator verglichen. Der ur-     |***V1*
CP    (IX+d) |19 | spruengliche Inhalt des Akkumulators    |***V1*
CP    (IY+d) |19 | bleibt dabei erhalten. Das Vergleichs-  |***V1*
             |   | ergebnis ist durch die Flags-Stellungen |
             |   | erkennbar.                              |
             |   |                                         |
INC   r      | 4 | Der Inhalt des Registers r wird um 1 er-|***V0-
             |   | hoeht                                   |
             |   |                                         |
```

| | | | | |
|---|---|---|---|---|
| INC | m | 11 | Der Inhalt des durch HL adressierten Speicherplatzes wird um 1 erhoeht | ***V0- |
| INC | (IX+d) | 23 | Der Inhalt des durch IX (bzw. IY) plus Verschiebung d adressierten Speicherplatzes wird um 1 erhoeht | ***V0- |
| INC | (IY+d) | 23 | | ***V0- |
| DEC | r | 4 | Der Inhalt des Registers r wird um 1 vermindert | ***V1- |
| DEC | m | 11 | Der Inhalt des durch HL adressierten Speicherplatzes wird um 1 vermindert | ***V1- |
| DEC | (IX+d) | 23 | Der Inhalt des durch IX (bzw. IY) plus Verschiebung d adressierten Speicherplatzes wird um 1 vermindert | ***V1- |
| DEC | (IY+d) | 23 | | ***V1- |
| DAA | | 4 | Korrigiert nach Addition / Subtraktion zweier gepackter BCD-Zahlen den Akkumulatorinhalt so, dass im Akkumulator wieder die gepackte BCD-Darstellung erreicht wird | ***P-* |
| CPL | | 4 | Bitweises Negieren (Komplementieren) des Akkumulatorinhalts | --1-1- |
| NEG | | 8 | Subtrahieren des Akkumulatorinhalts von Null. Entspricht wertmaessig dem Zweierkomplement | ***V1* |
| CCF | | 4 | Komplementieren des Carry-Flags | --x-0* |
| SCF | | 4 | Setzen des Carry-Flags | --0-01 |

### 3.5.6. 16-Bit-Arithmetikbefehle

Arbeiten aehnlich wie die 8-Bit-Arithmetikbefehle, jedoch mit Doppelregistern. Als Akkumulator wird eines der Register HL, IX oder IY benutzt.

| Mnemonik | | T | Wirkungsweise des Befehls | SZHPNC V |
|---|---|---|---|---|
| ADD | HL,dd | 11 | Der Inhalt des Registerpaares dd wird zum Inhalt des Registerpaares HL addiert | --x-0* |
| ADD | IX,IX | 15 | Der Inhalt des Indexregistes IX (bzw.IY) wird mit sich selbst addiert. Diese Verdoppelung ist gleichbedeutend mit einer Linksverschiebung der 16 Bit um eine Position | --x-0* |
| ADD | IY,IY | 15 | | --x-0* |
| ADD | IX,pp | 15 | Der Inhalt von pp wird zum Inhalt des Indexregisters IX (bzw. IY) addiert | --x-0* |
| ADD | IY,pp | 15 | | --x-0* |

| | | | |
|---|---|---|---|
| ADC HL,dd | 15 | Der Inhalt von dd plus Carry-Flag wird zum Inhalt des Registerpaares HL addiert | **xV0* |
| SBC HL,dd | 15 | Der Inhalt von dd plus Carry-Flag wird vom Inhalt des Registerpaares HL subtrahiert | **xV1* |
| INC dd | 6 | Der Inhalt des Doppelregisters dd wird um 1 erhoeht | ------ |
| INC IX<br>INC IY | 10<br>10 | Der Inhalt des Indexregisters IX (bzw. IY) wird um 1 erhoeht | ------<br>------ |
| DEC dd | 6 | Der Inhalt des Doppelregisters dd wird um 1 vermindert | ------ |
| DEC IX<br>DEC IY | 10<br>10 | Der Inhalt des Indexregisters IX (bzw. IY) wird um 1 vermindert | ------<br>------ |

### 3.5.7. Programmverzweigungsbefehle

Es ist zwischen unbedingten und bedingten Spruengen zu unterscheiden. Es sind weiterhin relative Spruenge moeglich, mit denen zu Marken in einer naeheren Umgebung (-126 bis +129 Byte) um die Befehlsadresse verzweigt werden kann. Im Quellprogramm ist dabei zwar die absolute Adresse der Marke anzugeben, im Befehlscode aber erscheint nur die relative Verschiebung zum momentanen Befehlszaehlerstand. Das Maschinenprogramm wird damit unabhaengig von seiner Lage im Speicher.
Bei bedingten Spruengen sind Flag-Bedingungen als Operanden anzugeben (s. Abschn. 3.3.3), und es werden die entsprechenden Flag-Bits getestet. In Abhaengigkeit von diesem Test wird der Sprungbefehl entweder ausgefuehrt oder ignoriert.

| Mnemonik | T | Wirkungsweise des Befehls | SZHPNC<br>V |
|---|---|---|---|
| JP nn | 10 | Unbedingter Sprung nach Adresse nn, indem der Befehlzzaehler PC mit nn geladen wird | ------ |
| JP NZ,nn | 10 | Sprung nach Adresse nn, wenn das Z-Flag gleich 0 ist | ------ |
| JP Z,nn | 10 | Sprung nach Adresse nn, wenn das Z-Flag gleich 1 ist | ------ |
| JP NC,nn | 10 | Sprung nach Adresse nn, wenn das C-Flag gleich 0 ist | ------ |
| JP C,nn | 10 | Sprung nach Adresse nn, wenn das C-Flag gleich 1 ist | ------ |

| | | | | |
|---|---|---|---|---|
| JP | PO,nn | 10 | Sprung nach Adresse nn, wenn das P/V-Flag gleich 0 ist | ------ |
| JP | PE,nn | 10 | Sprung nach Adresse nn, wenn das P/V-Flag gleich 1 ist | ------ |
| JP | P,nn | 10 | Sprung nach Adresse nn, wenn das S-Flag gleich 0 ist | ------ |
| JP | M,nn | 10 | Sprung nach Adresse nn, wenn das S-Flag gleich 1 ist | ------ |
| JR | nn | 12 | Unbedingter relativer Sprung nach Adresse nn | ------ |
| JR | NZ,nn | 12/7 | Relativer Sprung nach Adresse nn, wenn das Z-Flag gleich 0 ist | ------ |
| JR | Z,nn | 12/7 | Relativer Sprung nach Adresse nn, wenn das Z-Flag gleich 1 ist | ------ |
| JR | NC,nn | 12/7 | Relativer Sprung nach Adresse nn, wenn das C-Flag gleich 0 ist | ------ |
| JR | C,nn | 12/7 | Relativer Sprung nach Adresse nn, wenn das C-Flag gleich 1 ist | ------ |
| JP | m | 4 | Unbedingter Sprung zur Adresse, die im Doppelregister HL steht | ------ |
| JP | (IX) | 8 | Unbedingter Sprung zur Adresse, die im | ------ |
| JP | (IY) | 8 | Indexregister IX (bzw. IY) steht | ------ |
| DJNZ | nn | 13/8 | Der Inhalt des Registers B wird um 1 vermindert. Bedingter relativer Sprung zur Adresse nn, wenn der Inhalt des Registers B ungleich 0 ist. | ------ |

## 3.5.8. Unterprogrammbefehle

Es ist wie bei Sprungbefehlen zwischen bedingten und unbedingten Befehlen zu unterscheiden. Der Unterprogrammaufruf geschieht mit einem CALL-Befehl, bei dem, zusaetzlich zum Programmsprung, die dem CALL-Befehl im Speicher folgende Adresse (Rueckkehradresse) in den Kellerspeicher (Stack) gerettet wird. Wird nun das Unterprogramm mit einem RET-Befehl abgeschlossen, so wird diese Rueckkehradresse in den Befehlszaehler aus dem Stack zurueckgeladen, und das Programm wird an der alten Stelle fortgesetzt.

| Mnemonik | T | Wirkungsweise des Befehls | SZHPNC V |
|---|---|---|---|
| CALL nn | 17 | Unbedingter Unterprogrammaufruf zur Adresse nn. Die nach dem CALL-Befehl folgende Speicheradresse wird wie bei einem PUSH-Befehl in den Stack gerettet (Rueckkehradresse). Danach erfolgt ein unbedingter Sprung zur Adresse nn, indem der Befehlszzaehler PC mit nn geladen wird | ------ |
| CALL NZ,nn | 17/10 | Unterprogrammaufruf zur Adresse nn, wenn das Z-Flag gleich 0 ist | ------ |
| CALL Z,nn | 17/10 | Unterprogrammaufruf zur Adresse nn, wenn das Z-Flag gleich 1 ist | ------ |
| CALL NC,nn | 17/10 | Unterprogrammaufruf zur Adresse nn, wenn das C-Flag gleich 0 ist | ------ |
| CALL C,nn | 17/10 | Unterprogrammaufruf zur Adresse nn, wenn das C-Flag gleich 1 ist | ------ |
| CALL PO,nn | 17/10 | Unterprogrammaufruf zur Adresse nn, wenn das P/V-Flag gleich 0 ist | ------ |
| CALL PE,nn | 17/10 | Unterprogrammaufruf zur Adresse nn, wenn das P/V-Flag gleich 1 ist | ------ |
| CALL P,nn | 17/10 | Unterprogrammaufruf zur Adresse nn, wenn das S-Flag gleich 0 ist | ------ |
| CALL M,nn | 17/10 | Unterprogrammaufruf zur Adresse nn, wenn das S-Flag gleich 1 ist | ------ |
| RST p | 11 | Der RST-Befehl ist ein spezieller Unterprogrammaufruf. Es sind folgende 8 Restart-Adressen zugelassen: p = 00H,08H,10H,18H,20H,28H,30H,38H Der hoeherwertige Adressteil ist dabei stets 0. Der RST-Befehl entspricht in seiner weiteren Wirkung dem unbedingten | ------ |

| | | | |
|---|---|---|---|
| | \| \| | Unterprogrammaufruf. | |
| | \| \| | | |
| RET | \|10\| | Unbedingter Unterprogrammrueck-sprung. Die Ausfuehrung erfolgt, indem die Rueckkehradresse wie bei einem POP-Befehl aus dem Stack geholt und in den Befehlszaehler PC geladen wird. | ------ |
| RET NZ | \|11\| 5\| | Unterprogrammrueck-sprung, wenn das Z-Flag gleich 0 ist | ------ |
| RET Z | \|11\| 5\| | Unterprogrammrueck-sprung, wenn das Z-Flag gleich 1 ist | ------ |
| RET NC | \|11\| 5\| | Unterprogrammrueck-sprung, wenn das C-Flag gleich 0 ist | ------ |
| RET C | \|11\| 5\| | Unterprogrammrueck-sprung, wenn das C-Flag gleich 1 ist | ------ |
| RET PO | \|11\| 5\| | Unterprogrammrueck-sprung, wenn das P/V-Flag gleich 0 ist | ------ |
| RET PE | \|11\| 5\| | Unterprogrammrueck-sprung, wenn das P/V-Flag gleich 1 ist | ------ |
| RET P | \|11\| 5\| | Unterprogrammrueck-sprung, wenn das S-Flag gleich 0 ist | ------ |
| RET M | \|11\| 5\| | Unterprogrammrueck-sprung, wenn das S-Flag gleich 1 ist | ------ |
| RETI | \|14\| | Es erfolgt ein Rueck-sprung aus einer Interruptbehandlungsroutine, die durch einen maskierbaren Interrupt ausgeloest wurde. Dem peripheren Baustein, der den Interrupt ausloeste, wird das Ende seines Programms mitgeteilt. Der Baustein gibt daraufhin die von ihm blockierte Interrupt-Kette wieder frei und ermoeglicht damit die Abarbeitung niederwertiger Interrupts. Durch die RETI-Anweisung wird der maskierbare Interrupt nicht freigegeben. Es sollte deshalb vor jeder RETI-Anweisung ein EI-Befehl stehen, der die Annahme spaeter folgender Interruptanforderungen ermoeglicht. | ------ |
| RETN | \|14\| | Es erfolgt ein Rueck-sprung aus einer Interruptbehandlungsroutine, die durch einen nichtmaskierbaren Interrupt (NMI) ausgeloest wurde. Die Anweisung wirkt zunaechst wie ein RET-Anweisung. Zusaetzlich wird der Inhalt von IFF2 nach IFF1 uebertragen, so dass die Abarbeitung maskierbarer Interrupts unmittelbar nach Ausfuehrung des RETN-Befehls frei- | ------ |

| | | gegeben ist, falls sie vor der NMI-An- |
|---|---|---|
| | | forderung freigegeben war. |

## 3.5.9. Rotations- und Verschiebebefehle

Durch diese Befehle wird die Moeglichkeit gegeben, im Akkumulator (Register A), in einem anderen Register oder in einem Speicherplatz Daten einfach zyklisch (bitweise) zu verschieben. Das aus dem Byte herausgeschobene Bit wird dabei im Carry-Flag abgelegt.

| Mnemonik | T | Wirkungsweise des Befehls | SZHPNC V |
|---|---|---|---|
| RLCA | 4 | Linksrotation des Akkumulatorinhalts. Der Inhalt des Akkumulators wird um eine Bitposition nach links verschoben. Das hoechstwertige Bit 7 wird zum Inhalt des Bits 0 und des Carry-Flags.<br><br>----------><br>\| \|<br>CY<----B7<----B0<- | --0-0* |
| RRCA | 4 | Rechtsrotation des Akkumulatorinhalts. Der Inhalt des Akkumulators wird um eine Bitposition nach rechts verschoben. Das niederwertige Bit 0 wird zum Inhalt des Bits 7 und des Carry-Flags.<br><br><----------<br>\| \|<br>->B7---->B0---->CY | --0-0* |
| RLA | 4 | Linksrotation des Akkumulatorinhalts durch das Carry-Flag. Der Inhalt des Akkumulators wird um eine Bitposition nach links verschoben. Das hoechstwertige Bit 7 ersetzt das Carry-Flag, waehrend das Carry-Flag das Bit 0 des Akkumulators ersetzt.<br><br>----------><br>\| \|<br>-CY<----B7<----B0- | --0-0* |
| RRA | 4 | Rechtsrotation des Akkumulatorinhalts durch das Carry-Flag. Der Inhalt des Akkumulators wird um eine Bitposition nach rechts verschoben. Das niederwertigste Bit 0 ersetzt das Carry-Flag, waehrend das Carry-Flag das Bit 7 des Akkumulators ersetzt. | --0-0* |

```
            |  |        <-----------------               |
            |  |       |                  |              |
            |  |        -B7---->B0---->CY-               |
            |  |                                         |
RLC   r     | 8| Linksrotation eines Registers oder      |**0P0*
RLC   m     |15| Speicherbytes analog dem Befehl RLCA    |**0P0*
RLC   (IX+d)|23|                                         |**0P0*
RLC   (IY+d)|23|                                         |**0P0*
            |  |                                         |
RRC   r     | 8| Rechtsrotation eines Registers oder     |**0P0*
RRC   m     |15| Speicherbytes analog dem Befehl RRCA    |**0P0*
RRC   (IX+d)|23|                                         |**0P0*
RRC   (IY+d)|23|                                         |**0P0*
            |  |                                         |
RL    r     | 8| Linksrotation eines Registers oder      |**0P0*
RL    m     |15| Speicherbytes durch das Carry-Flag      |**0P0*
RL    (IX+d)|23| analog dem Befehl RLA                   |**0P0*
RL    (IY+d)|23|                                         |**0P0*
            |  |                                         |
RR    r     | 8| Rechtsrotation eines Registers oder     |**0P0*
RR    m     |15| Speicherbytes durch das Carry-Flag      |**0P0*
RR    (IX+d)|23| analog dem Befehl RRA                   |**0P0*
RR    (IY+d)|23|                                         |**0P0*
            |  |                                         |
SRA   r     | 8| Rechtsverschiebung eines Registers oder |**0P0*
SRA   m     |15| Speicherbytes um ein Bit durch das      |**0P0*
SRA   (IX+d)|23| Carry-Flag. Der Inhalt des hoechstwerti-|**0P0*
SRA   (IY+d)|23| gen Bit 7 bleibt erhalten.              |**0P0*
            |  |                                         |
            |  |        -->                              |
            |  |       |   |                             |
            |  |        -B7---->B0---->0                 |
            |  |                                         |
SLA   r     | 8| Linksverschiebung eines Registers oder  |**0P0*
SLA   m     |15| Speicherbytes um ein Bit durch das      |**0P0*
SLA   (IX+d)|23| Carry-Flag. Das niederwertige Bit 0     |**0P0*
SLA   (IY+d)|23| wird 0.                                 |**0P0*
            |  |                                         |
            |  |       CY<----B7<----B0<----0            |
            |  |                                         |
SRL   r     | 8| Rechtsverschiebung eines Registers oder |**0P0*
SRL   m     |15| Speicherbytes um ein Bit durch das      |**0P0*
SRL   (IX+d)|23| Carry-Flag. Das hoechstwertige Bit 7    |**0P0*
SRL   (IY+d)|23| wird 0.                                 |**0P0*
            |  |                                         |
            |  |       0---->B7---->B0---->CY            |
            |  |                                         |
RLD         |18| Zyklische Verschiebung nach links zwi-  |**0P0-
            |  | schen dem Akkumulator und dem Inhalt des|
            |  | durch HL adressierten Speicherplatzes.  |
            |  | Die unteren 4 Bit des durch HL adres-   |
            |  | sierten Speicherplatzes werden in die   |
            |  | oberen 4 Bitstellen uebertragen und     |
            |  | diese ihrerseits in die unteren 4 Bit-  |
            |  | stellen des Akkumulators. Die unteren 4 |
            |  | Bits des Akkumulators werden in die un- |
            |  | teren 4 Bitstellen der Speicherstelle   |
            |  | transportiert. Die oberen 4 Bits des Ak-|
```

| | | | |
|---|---|---|---|
| | | kumulators bleiben unberuehrt. | |
| | | | |
| RRD | 18 | Zyklische Verschiebung nach rechts zwischen dem Akkumulator und dem Inhalt des durch HL adressierten Speicherplatzes. Die unteren 4 Bit des durch HL adressierten Speicherplatzes werden in die unteren 4 Bitstellen des Akkumulators uebertragen und diese in die oberen der durch HL adressierten Speicherstelle. Die oberen 4 Bits der durch HL adressierten Speicherstelle werden in die unteren 4 Bitstellen transportiert. Die oberen 4 Bits des Akkumulators bleiben unberuehrt. | **0P0- |

## 3.5.10. Einzelbitbefehle

Diese Befehle erlauben es, einzelne Bits in Registern oder auf Speicherplaetzen zu testen, zu setzen oder zu loeschen.

| Mnemonik | T | Wirkungsweise des Befehls | SZHPNC V |
|---|---|---|---|
| | | | |
| BIT b,r | 8 | Die durch b gekennzeichnete Bitposition wird in dem Register r getestet. Das Komplement des zu testenden Bits wird in das Z-Flag geladen. | x*1x0- |
| | | | |
| BIT b,m | 12 | Die durch b gekennzeichnete Bitposition wird in der Speicherstelle getestet, die durch das Register HL adressiert ist. Das Komplement des zu testenden Bits wird in das Z-Flag geladen. | x*1x0- |
| | | | |
| BIT b,(IX+d) | 20 | Die durch b gekennzeichnete Bitposition wird in der Speicherstelle getestet, die durch das Indexregister IX (bzw. IY) plus Verschiebung d adressiert ist. Das Komplement des zu testenden Bits wird in das Z-Flag geladen. | x*1x0- |
| BIT b,(IY+d) | 20 | | x*1x0- |
| | | | |
| SET b,r | 8 | Die durch b gekennzeichnete Bitposition wird in dem Register r gesetzt (auf 1) | ------ |
| | | | |
| SET b,m | 15 | Die durch b gekennzeichnete Bitposition wird in der Speicherstelle gesetzt, die durch das Register HL adressiert ist | ------ |
| | | | |
| SET b,(IX+d) | 23 | Die durch b gekennzeichnete Bitposition wird in der Speicherstelle gesetzt, die durch das Indexregister IX (bzw. IY) plus Verschiebung d adressiert ist | ------ |
| SET b,(IY+d) | 23 | | ------ |

| | | | |
|---|---|---|---|
| RES b,r | 8 | Die durch b gekennzeichnete Bitposition wird in dem Register r geloescht (Null) | ------ |
| RES b,m | 15 | Die durch b gekennzeichnete Bitposition wird in der Speicherstelle geloescht, die durch das Register HL adressiert ist | ------ |
| RES b,(IX+d)<br>RES b,(IY+d) | 23 | Die durch b gekennzeichnete Bitposition wird in der Speicherstelle geloescht, die durch das Indexregister IX (bzw. IY) plus Verschiebung d adressiert ist | ------<br>------ |

### 3.5.11. CPU-Steuerbefehle

Diese Befehle dienen zur Steuerung des Interruptsystems der CPU. Der Interruptmodus ist im KC 85/2 und KC 85/3 auf IM 2 eingestellt und sollte nicht veraendert werden.

| Mnemonik | T | Wirkungsweise des Befehls | SZHPNC V |
|---|---|---|---|
| NOP | 4 | Die CPU fuehrt keine Operation aus. Es werden aber Refresh-Zyklen erzeugt. | ------ |
| HALT | 4 | Die CPU fuehrt solange eine Folge von NOP-Befehlen aus, bis ein Interrupt oder der RESET-Eingang an der CPU aktiv wird. Es werden Refresh-Zyklen erzeugt. | ------ |
| DI | 4 | Der maskierbare Interrupt wird durch Ruecksetzen der Interrupt-Freigabe-Flip-Flops IFF1 und IFF2 der CPU gesperrt. Nichtmaskierbare Interrupts werden anerkannt. | ------ |
| EI | 4 | Der maskierbare Interrupt wird durch Setzen der Interrupt-Freigabe-Flip-Flops IFF1 und IFF2 der CPU freigegeben. Waehrend der Ausfuehrung des Befehls akzeptiert die CPU keine Interruptanforderungen. | ------ |
| IM 0 | 8 | Der Befehl bringt die CPU in den Interruptmodus 0 | ------ |
| IM 1 | 8 | Der Befehl bringt die CPU in den Interruptmodus 1 | ------ |
| IM 2 | 8 | Der Befehl bringt die CPU in den Interruptmodus 2 | ------ |

## 3.5.12. Ein- und Ausgabebefehle

Mit diesen Befehlen koennen Datenbytes zwischen Registern oder Speicheradressen und externen Bausteinen ausgetauscht werden. Der externe Baustein wird dabei ueber eine sog. Portadresse (8-Bit-Wert) angesprochen. Diese Portadresse wird je nach Befehl entweder direkt angegeben (als Konstante) oder muss im Register C zur Verfuegung stehen. Aehnlich den Blocktransferbefehlen existieren auch hier Befehle fuer die Daten-Ein- und -Ausgabe ganzer Speicherbereiche.

| Mnemonik | T | Wirkungsweise des Befehls | SZHPNC V |
|---|---|---|---|
| IN A,(n) | 11 | Die Eingabekanaladresse wird mittels der Konstanten n eingestellt. Zielregister ist der Akkumulator.<br>(n) --> A | ------ |
| IN r,(C) | 12 | Die Eingabekanaladresse wird indirekt ueber das Register C eingestellt. Zielregister ist r.<br>(C) --> r | **0P0- |
| INI | 16 | Die Eingabekanaladresse wird indirekt ueber das Register C eingestellt. Zieladresse ist der durch HL adressierte Speicherplatz. B kann als Bytezaehler arbeiten.<br>B wird um 1 vermindert, HL um 1 erhoeht.<br>(C) --> (HL)<br>B-1 --> B<br>HL+1 --> HL | x*xx1- |
| INIR | 21<br>16 | Die Eingabekanaladresse wird indirekt ueber das Register C eingestellt. Zieladresse ist der durch HL adressierte Speicherplatz. B kann als Bytezaehler arbeiten.<br>B wird um 1 vermindert, HL um 1 erhoeht.<br>Es wird eine Blockuebertragung durchgefuehrt bis B = 0 ist.<br>(C) --> (HL)<br>B-1 --> B<br>HL+1 --> HL<br>Wiederholen bis B = 0 | x1xx1- |
| IND | 16 | Die Eingabekanaladresse wird indirekt ueber das Register C eingestellt. Zieladresse ist der durch HL adressierte Speicherplatz. B kann als Bytezaehler arbeiten.<br>B und HL werden um 1 vermindert.<br>(C) --> (HL)<br>B-1 --> B | x*xx1- |

```
             |   | HL-1  -->  HL                              |
             |   |                                            |
INDR         |21| Die Eingabekanaladresse wird indirekt       |x1xx1-
             |16| ueber das Register C eingestellt. Ziel-     |
             |   | adresse ist der durch HL adressierte       |
             |   | Speicherplatz. B kann als Bytezaehler      |
             |   | arbeiten.                                  |
             |   | B und HL werden um 1 vermindert.           |
             |   | Es wird eine Blockuebertragung durchge-    |
             |   | fuehrt bis B = 0 ist.                      |
             |   | (C)   -->  (HL)                            |
             |   | B-1   -->  B                               |
             |   | HL-1  -->  HL                              |
             |   | Wiederholen bis B = 0                      |
             |   |                                            |
OUT   (n),A  |11| Die Ausgabekanaladresse wird mittels der|------
             |   | Konstanten n eingestellt. Quellregister    |
             |   | ist der Akkumulator.                       |
             |   | A  -->  (n)                                |
             |   |                                            |
OUT   (C),r  |12| Die Ausgabekanaladresse wird indirekt       |------
             |   | ueber das Register C eingestellt. Quell-|
             |   | register ist r.                            |
             |   | r  -->  (C)                                |
             |   |                                            |
OUTI         |16| Die Ausgabekanaladresse wird indirekt       |x*xx1-
             |   | ueber das Register C eingestellt. Quell-|
             |   | adresse ist der durch HL adressierte       |
             |   | Speicherplatz. B kann als Bytezaehler      |
             |   | arbeiten.                                  |
             |   | B wird um 1 vermindert, HL um 1 erhoeht.|
             |   | (HL)  -->  (C)                             |
             |   | B-1   -->  B                               |
             |   | HL+1  -->  HL                              |
             |   |                                            |
OTIR         |21| Die Ausgabekanaladresse wird indirekt       |x1xx1-
             |16| ueber das Register C eingestellt. Quell-|
             |   | adresse ist der durch HL adressierte       |
             |   | Speicherplatz. B kann als Bytezaehler      |
             |   | arbeiten.                                  |
             |   | B wird um 1 vermindert, HL um 1 erhoeht.|
             |   | Es wird eine Blockuebertragung durchge-    |
             |   | fuehrt bis B = 0 ist.                      |
             |   | (HL)  -->  (C)                             |
             |   | B-1   -->  B                               |
             |   | HL+1  -->  HL                              |
             |   | Wiederholen bis B = 0                      |
             |   |                                            |
OUTD         |16| Die Ausgabekanaladresse wird indirekt       |x*xx1-
             |   | ueber das Register C eingestellt. Quell-|
             |   | adresse ist der durch HL adressierte       |
             |   | Speicherplatz. B kann als Bytezaehler      |
             |   | arbeiten.                                  |
             |   | B und HL werden um 1 vermindert.           |
             |   | (HL)  -->  (C)                             |
             |   | B-1   -->  B                               |
             |   | HL-1  -->  HL                              |
             |   |                                            |
```

| | | | |
|---|---|---|---|
| OTDR | 21<br>16 | Die Ausgabekanaladresse wird indirekt ueber das Register C eingestellt. Quelladresse ist der durch HL adressierte Speicherplatz. B kann als Bytezaehler arbeiten.<br>B und HL werden um 1 vermindert.<br>Es wird eine Blockuebertragung durchgefuehrt bis B = 0 ist.<br>(HL) --> (C)<br>B-1 --> B<br>HL-1 --> HL<br>Wiederholen bis B = 0 | x1xx1- |

# 4. Disassembler

## 4.1. Inhalt und Bedienung

Nachdem man den Modul zugewiesen hat, erscheinen mit dem Aufruf des Kommandos MENU auch die Menueworte "DISASS" und "CDISASS".

Der Disassembler ermoeglicht das blockweise Rueckuebersetzen von einem im Speicher des Computers stehenden Maschinenprogramm in die mnemonische Form.

Mit der Eingabe:

DISASS aaaa bbbb (nn) (oooo)

wird der Disassembler aufgerufen. Dabei bedeuten die Parameter:

aaaa - Anfangsadresse des zu disassemblierenden Programms
bbbb - Endadresse +1 des zu disassemblierenden Programms
nn - Anzahl der auf einer Bildschirmseite stehenden Zeilen
oooo - Offset, Verschiebeparameter

Mit dem Verschiebeparameter " Offset " kann man die Ladeadresse eines Programms um den angegebenen Wert veraendern.

Beispiel:
Ein Programm, das den Bereich von 7A00H - 7EFFH belegt, soll disassembliert werden. Da aber moeglicherweise kein Speicher-erweiterungsmodul verfuegbar ist, muss deshalb das Programm auf eine andere Adresse geladen werden. Gewaehlt wird z.B. 2A00H. Das Programm wird mit

LOAD B000

in den Computer auf die Adresse 2A00H geladen. Damit die Adresse des disassemblierten Programms der Anfangsadresse 7A00H entspricht, ist der Verschiebeparameter (oooo) = 5000H einzugeben.

Eingabe:

DISASS 2A00 2EFF 1A 5000.

Der 3. Parameter nn ist mit 1AH angegeben. Das heisst, es werden 26 Zeilen mit einmal auf dem Bildschirm angezeigt. Wird der Parameter nn nicht angegeben, enthaelt eine Bildschirmseite 30 Zeilen. Nach dieser Anzahl von Zeilen wartet das Programm, bei eingeblendetem Cursor, auf eine Tasteneingabe.

Reaktionen der Tasten:

| | |
|---|---|
| BRK | Ruecksprung in das CAOS - Betriebssystem |
| ENTER | Ausgabe der naechsten nn Zeilen |
| SHIFT HOME | Loeschen des Bildschirmes und Ausgabe der naechsten nn Zeilen ( blaettern ) |

Mit dem Menuewort "CDISASS" wird das erzeugte Quellprogramm auf Kassette ausgegeben. Danach kann man mit dem auf Kassette gespeicherten Programm mit EDAS weiterarbeiten. Bei der Ausgabe mit CDISASS muss eine Offset Angabe erfolgen !

CDISASS erzeugt also Quellprogramme, die mit EDAS weiterbearbeitet werden. Dabei empfiehlt es sich, saemtliche Sprungadressen mit Marken zu versehen. Werden keine Marken gesetzt, muss zum Erhalt der richtigen Sprungweite bei relativen Spruengen eine ORG-Anweisung mit der Orginalanfangsadresse des zu disassemblierenden Programms am Anfang des Quelltextes stehen.

Folgende Parameter sind anzugeben:

CDISASS aaaa bbbb oooo

Bedeutung der Parameter:

aaaa - Anfangsadresse
bbbb - Endadresse + 1
oooo - Offset ( wie DISASS )

Fuer die Kassettenaufzeichnung sind acht Zeichen zur Benennung des Dateinamens zugelassen. Als Dateityp wird " ASM " vereinbart.

## 4.2. Ausgabe auf den Bildschirm

Die Ausgabe auf den Bildschirm erfolgt in Form von Befehlsadressen und -mnemonik. Der Disassembler ist speziell fuer den KC 85/2 und KC 85/3 erarbeitet worden. Deshalb werden die Besonderheiten des Betriebssystems CAOS bei der Rueckuebersetzung einbezogen. Beispielsweise fuehrt der Aufruf des Programmverteilers des CAOS - Betriebssystems auf Adresse F003H zu folgender Ausschrift.

```
Beispiel:      0100 CALL 0F003H
               0103 DEFB 003H          ( Unterprogrammnummer 3 )
```

Beim Aufruf des Unterprogramms 23H ( Zeichenkettenausgabe ) werden die der 23H folgenden Bytes bis einschliesslich des ersten Bytes 00H das erkannt wird, als Zeichenkette bzw. als Datenbytes ausgegeben.

```
Beispiel:      0100  CALL  0F003H
               0103  DEFB  023H
               0104  DEFB  00CH        ( Bildschirm loeschen )
               0105  DEFM  'Testprogramm'
               0111  DEFB  0AH         ( Cursor down )
               0112  DEFB  0DH         ( Cursor auf Zeilenanfang )
               0113  DEFM  'Eingabe'
               011A  DEFB  000H        ( Ende der Zeichenkette )
```

## 4.3. Zusätzliche U880-Befehle

Ueber den offiziellen Befehlssatz hinaus kennt der Disassembler einige spezielle Indexregisterbefehle, die wie die 8-Bit-Register benutzt werden koennen. Die bekannten Lade- und Arithmetikbefehle mit den Registern H und L koennen ebenfalls durch Vorsetzen der Umschaltbytes FD und DD auf das niederwertige (low) oder hoeherwertige (high) Byte der Indexregister angewendet werden. Die Bezeichungen lauten HX, HY, LX, LY.

```
Beispiel:    FD 44 =  LD B,HY
             DD 85 =  ADD LX
```

In der Gruppe der 2-Byte-Befehle fehlt eine Spalte mit der Kodierung 110B fuer die Bits 3 - 5. Diese Kodierung erzeugt einen Schiebebefehl nach links mit dem Setzen des Bits 0 ! Der Befehl wird mit SLS uebersetzt.

Zu den 4 - Byte - Indexregisterbefehlen ist zu bemerken, dass das letzte Byte die Registeradresse in den Bits 0 bis 2 enthaelt. Wird in diese genannten Bits eine andere Kodierung eingetragen als 110B, erfolgt ein Eintrag des Operationsergebnisses in ein entsprechendes Register. Im Disassemblerprotokoll steht dafuer der Zusatz $ LD reg.

```
Beispiel:   FD CB 03 01 = RCL ( IY + 03H ) & LD C
            FD CB 04 32 = SLS ( IY + 04H ) & LD C
```

Diese Darstellungsweise funktioniert nicht mit den Bittestbefehlen. Da der Hersteller des U 880D diese zusaetzlichen Befehle nicht mit angibt, kann dafuer keine Garantie uebernommen werden. Der Assembler uebersetzt diese Befehle auch nicht, deshalb sollten sie nicht verwendet werden.

# 5. Testmonitor

## 5.1. Inhalt und Bedienung

### 5.1.1. Allgemeines

Haupteinsatzgebiet des Programmpaketes "Testmonitor" ist die Unterstuetzung beim Test von Maschinenprogrammen. Es dient zur Fehlersuche bzw. Programmanalyse mittels Einzelbefehlsabarbeitung und Unterbrechungspunktsteuerung mit nachfolgender Anzeige der Registerinhalte. Fuer die zu testenden Anwenderprogramme werden eigene USER-Register definiert, welche nach der Anwenderprogrammabarbeitung im RAM gerettet werden und zur Abarbeitung in den CPU-Registersatz geladen werden. Die Einzelschrittabarbeitung erfolgt interruptgesteuert.

Der Kanal 1 des CTC-Bausteins des Rechners wird so programmiert, dass nach dem Sprung in das Anwenderprogramm auf dem ersten Befehl ein Interrupt ausgeloest wird. Anschliessend werden die Register gerettet und angezeigt.

Die Unterbrechungspunktsteuerung ist auf zwei Arten moeglich:

1. Abarbeitung im Echtzeitbetrieb; die Unterbrechung wird durch Eintragen eines RST 38 - Befehls auf den Unterbrechnungspunkt realisiert,

2. die Unterbrechung wird ueber Adressvergleich nach Abarbeitung jedes Befehls realisiert (interruptgesteuert wie Einzelschrittabarbeitung)

Weiterhin beinhaltet der Testmonitor Kommandos zur Verschiebung von Speicherbereichen, Kommandos zur Einzelbyte-Ein- und -Ausgabe ueber die anzugebenden Ein-/Ausgabekanaladressen und weitere allgemein nutzbare Kommandos.

### 5.1.2. Starten des Testmonitors

Der Testmonitor wird aus dem erweiterten Grundmenue ueber die Kommandos

```
        TEMO (n)
oder
        RETEMO
```

initialisiert.

Parameter n
- wird kein Parameter angegeben, so wird der Disassembler aktiviert und es erfolgt bei Registeranzeige die Anzeige der Befehlsmnemonik; im Menuekopf erscheint die Ausschrift ' (DISASS) ' (ACHTUNG! der Disassembler veraendert den Zweitregistersatz auch fuer Anwenderprogramme)
- wird ein beliebiger Parameter angegeben, erfolgt kein Aufruf des Disassemblers, im Menuekopf fehlt dann die Ausschrift '(DISASS)'

Nach dem Start erscheint das Menue des Testmonitors:

```
>>> TEMO <<<   (DISASS)
+ MENU
+ ARITH
+ BREAK
+ GO
+ STEP
+ COPY
+ MOVE
+ REG
+ SBKR
+ IN
+ OUT
+ DISPLAY
+ EXIT
+ MODIFY
+ FILL
+ CHSUM
+ WORKRAM
+
```

Durch den Sprung in das TEMO-Menue ueber Kommando TEMO wird der Unterbrechungspunkt- und Schrittbetrieb initialisiert. Dazu werden

- die USER-Register AF, BC, DE, HL und PC auf Null gesetzt
- der Anwender-Stack auf 160H gelegt (bzw. unterhalb des Systemstacks)
- der Interruptvektor des CTC fuer den Einzelschrittbetrieb vorbereitet

Durch den Sprung in das TEMO-Menue ueber das Kommando RETEMO bleibt der Inhalt der USER-Register unveraendert. Die Initialisierung bezueglich des Disassemblers wird durch das Kommando 'RETEMO' nicht veraendert.

### 5.1.3. Kommandos

Im folgenden werden die einzelnen Kommandos des Testmonitors beschrieben:

- **MENU** - Ausschreiben des Testmonitormenues

- **ARITH aaaa bbbb** - Anzeige der Summe, Differenz und des Sprungoffsets zweier Zahlen

Parameter aaaa, bbbb - Hex-Zahlen

Anzeige nnnn mmmm oo
  nnnn - Summe
  mmmm - Differenz bbbb - aaaa
    oo - Offset fuer relativen Sprung von aaaa nach bbbb

**Beispiel ARITH 1008 1011**

**Anzeige 2019 0009 07**

**- BREAK nnnn - Vereinbarung eines Unterbrechungspunktes im RAM**
**==========**

**Parameter nnnn - Adresse des Unterbrechungspunktes**

**Beim Start eines Anwenderprogrammes mit 'GO' wird ein Byte FFH (entspr. Befehl RST 38H) auf die angegebene Adresse geschrieben und nach Erreichen der Adresse im Programmlauf wieder auf den urspruenglichen Wert zurueckgesetzt, anschliessend wird in den Schrittbetrieb uebergegangen. Die Speicherplaetze 38H bis 3AH werden fuer einen Sprungbefehl in den Testmonitor benoetigt.**

**Wird ein mit GO in Echtzeit gestartetes Programm vor Erreichen des Unterbrechungspunktes mit der RESET-Taste unterbrochen, muss beachtet werden, dass auf den Unterbrechungspunkt der RST 38 eingetragen ist und korrigiert werden muss. Dazu ist es notwendig, mit MODIFY auf der Adresse B700H die Adressen des BREAK-Punktes und des Datenbytes zu ermitteln und das Datenbyte auf der BREAK-Punktadresse zu aendern.**

**Ein nicht eingegebener BREAK-Punkt setzt den Unterbrechungspunkt auf die Adresse 0 beim Start mit GO.**

**Beispiel BREAK 109 - Vereinbarung eines Unterbrechungspunktes auf Adresse 109H**

**- GO nnnn - Sprung in ein Anwenderprogramm mit Initialisierung**
**========  des Unterbrechungspunktes**

**Parameter nnnn - Ansprungadresse**

**Wird ein Programm mit GO gestartet, so muss ein BREAK-Punkt auf den Austrittspunkt des zu testenden Programms vorher vereinbart werden.**

**Wird kein Wert eingegeben, so erfolgt der Programmstart ab der Adresse, auf welche der USER-PC zeigt. Das Programm wird im Echtzeitlauf abgearbeitet, bis ein Befehl RST 38H erreicht wird (Unterbrechungspunkt oder beliebiges Befehls-Byte FFH).**

**Beispiel GO 100 - Start eines Anwenderprogrammes auf Adresse 100H und Eintragen des Unterbrechungspunktes (im vorigen Beispiel 109H)**

**- STEP (nnnn) - Ausfuehren eines Anwenderbefehls mit folgender**
**==========  Anzeige der Register, Flags, Bytes des Folgebefehls und ggf. der Mnemonik des Folgebefehls (bei aktiviertem Disassembler)**

**Parameter nnnn - Anfangsadresse des Schrittbetriebes**

Entfaellt nnnn, so wird der letzte Wert des USER-PC verwendet. Wurde vorher mit 'SBRK' ein Haltepunkt vereinbart, erfolgt die Unterbrechung erst bei Erreichen des Haltepunktes bzw. bei Betaetigung der Taste 'BRK'.

Nach der Anzeige erscheint der Cursor und es kann mit folgenden Tasten fortgesetzt werden:

'ENTER' - Abarbeiten des naechsten USER-Befehls
'Cursor down' - Setzen eines Haltepunktes auf die dem angezeigten Befehl folgenden Adressen zum kompletten Abarbeiten von Programmschleifen und Unterprogrammen einschliesslich CAOS-Systemrufen. Dabei ist zu beachten, dass die Abarbeitung nicht in Echtzeit erfolgt, da das Programm nach jedem Befehl zum Adressvergleich unterbrochen wird. Die zeitliche Dehnung ist etwa 120fach. Es kann jederzeit mit der 'BRK'-Taste unterbrochen und in die Schrittbetriebsanzeige gesprungen werden. (Die zeitliche Dehnung macht sich extrem beim Bildschirmloeschen und -rollen bemerkbar!)
'BRK' - Uebergang in die Kommandoeingabe des Testmonitors

Beispiel STEP 100H - Abarbeitung des Befehls auf Adresse 100H

- COPY aaaa bbbb cccc - kopieren eines Speicherbereiches
  ==========

Parameter aaaa - Quelladresse
bbbb - Zieladresse
cccc - Anzahl Bytes

Das Umspeichern erfolgt ueber einen 'LDIR'-Befehl,ohne Kontrolle auf ueberlappende Speicherbereiche und ist deshalb auch zum Loeschen oder Beschreiben von Speicherbereichen geeignet.

Beispiel COPY 100 101 8 - beschreibt die Speicherplaetze 101H bis 108H mit dem Inhalt der Adresse 100H

- MOVE aaaa bbbb cccc - Verschieben eines Speicherbereiches
  ==========

Parameter - vgl. COPY

Das Umspeichern erfolgt mit Kontrolle auf ueberlappende Speicherbereiche.

Beispiel: MOVE 100 101 8 - Verschieben der Inhalte der Speicherplaetze 100H bis 107H auf die Speicherplaetze 101H bis 108H

- REG a nnnn - Belegen eines Anwenderdoppelregisters (USER-Reg.)
  ==========

  Parameter a  - Registerpaar
         nnnn  - Wert (hexadezimal)

Fuer die Register existieren Speicherplaetze im RAM:

| a | Register | Adresse |
|---|---|---|
| 0 | AF | B7F2H |
| 1 | BC | B7F4H |
| 2 | DE | B7F6H |
| 3 | HL | B7F8H |
| 4 | PC | B7FAH |
| 5 | SP | B7FCH |
| 6 | IY | Register IY direkt |

  Die Flags koennen ueber AF gesetzt werden.

  Beispiel:  REG 4 100 - Setzen des USER-Stack auf Adresse 100H
Neben dem Kommando REG bietet der Testmonitor noch eine weitere Moeglichkeit die Registerinhalte und die Flagwerte festzulegen, was jedoch nicht aus dem TEMO-Menue ersichtlich ist. Hierzu werden die Registernamen bzw. die Flagbezeichnungen als Kleinbuchstaben direkt nach dem Promptzeichen des TEMO-Menues eingegeben und der Registerinhalt nach Leerzeichen als Hexazahl angefuegt.

- (reg) nnn  - Belegen eines Anwenderregisters bzw. Flags.
  =========    Statt '(reg)' ist der Name des Anwenderregisters
               bzw. Flags in Kleinbuchstaben anzugeben. Moeglich
               sind folgende Eingaben:
               fuer Einfachregister: a,b,c,d,e,h,l
               fuer Doppelregister:  bc,de,hl,pc,iy
               fuer Flags:           z,cy

  Parameter  nnnn  - Wert des Registers bzw. Flaginhaltes
                     (hexadezimal).

  Beispiel:  cy 1   - Setzen des Carry-Flags auf 1

- SBRK nnnn  - Vereinbarung eines Haltepunktes
  =========

  Parameter nnnn - Haltepunktadresse

Die Haltepunktadresse zeigt auf das 1. Byte des Befehls. Der Haltepunkt kann auch im ROM/EPROM Bereich liegen. Die Haltepunktsteuerung erfolgt ueber Adressvergleich nach jedem abgearbeiteten Befehl (vgl. 'STEP'- Fortsetzung mit 'Cursor down').

  Beispiel:  SBRK 3155 - Vereinbarung eines Haltepunktes auf
                         Adresse 3155H

- IN aaaa - Anzeige eines Datenbytes vom Eingabekanal
  =======

  Parameter aaaa - 2-Byte-Kanaladresse

  Bildschirmanzeige:  nn- eingelesenes Datenbyte

  Beispiel:  IN  880  -Lesen  des Strukturbytes  des  Moduls  im
                       rechten Schacht

- OUT aaaa nn - Ausgabe eines Datenbytes zu einem Ausgabekanal
  ===========

  Parameter aaaa - 2-Byte-Kanaladresse
              nn - Datenbyte

  Beispiel: OUT C80 0 - Abschalten des Moduls im linken Schacht

- DISPLAY aaaa bbbb cc - Ausgabe HEX/ASCII-Dump
  ====================

  Parameter aaaa  - Anfangsadresse
            bbbb  - Endadresse, es werden 8 Byte in einer Zeile
                    angezeigt
              cc  - Anzahl der zusammenhaengenden Zeilen
                    (wenn kein Wert eingegeben wurde, gilt cc=8)

Die Ausschrift erfolgt im folgenden Format:

Adresse 8 1 -Byte-Hex-Zahlen  8 ASCII-Zeichen

Entsprechend den vorhandenen Zeichenbildtabellen werden im
Bereich der ASCII-Zeichen die den Hex-Codes entsprechen Zeichen
dargestellt.

Nach cc Zeilen erscheint der Cursor und wartet auf eine Eingabe.

Mit der 'BRK'-Taste kann abgebrochen werden.

Durch Betaetigung der 'STOP'-Taste wird in den MODIFY-Modus
uebergegangen.  Dabei wird die dem Hex-Dump folgende Adresse mit
ihrem Inhalt auf der naechsten Bildschirmzeile angezeigt.

Mittels Cursorbewegung koennen nun beliebige Datenbytes im
Hexadezimalteil der Displayanzeige angewaehlt und veraendert
werden.  Die Uebernahme erfolgt in jeder Zeile durch Betaetigung
der 'ENTER'-Taste. Der ASCII-Teil wird nicht  aktualisiert. Alle
weiteren Bedingungen entsprechen dem MODIFY-Modus.

Bei  Betaetigung einer beliebigen anderen Taste wird  die
Ausschrift fortgesetzt.

  Beispiel:  DISPLAY  0 100 - Hex/ASCII-Dump des Bereiches von 0
                              bis 100H

- EXIT - Rueckkehr in das Betriebssystemgrundmenue
  ====

- MODIFY nnnn - ueberpruefen und aendern von Speicherbereichen
  ==========

  Parameter nnnn - Adresse

  entspricht dem Kommando MODIFY des Hauptmenues

- FILL aaaa bbbb nn - Beschreiben (Fuellen) eines
  ========== Speicherbereiches

  Parameter aaaa - Anfangsadresse
  bbbb - Endeadresse +1
  cc - Datenbyte, mit dem der angegebene Bereich beschrieben werden soll

  Beispiel: FILL 200 500 FF - Beschreiben des Bereiches von Adresse 200H bis 500H mit FFH

- CHSUM aaaa bbbb - Berechnung eines Pruefpolynoms (z.B. zum
  ========== Ueberpruefen von EPROM's oder von zu testenden Programmen im RAM auf Veraenderungen

  Parameter aaaa - Anfangsadresse
  bbbb - Laenge des Bereiches, ueber dem das Pruefpolynom errechnet werden soll

  Beispiel: CHSUM C00 200 - Berechnung des Pruefpolynoms ab Adresse C000H in der Laenge 200H
  Anzeige: CE6D

- WORKRAM aa - Verlegen von IX-, Interrupttabellen und Stack in
  ========== anderen RAM-Bereich

  Parameter aa - hoeherwertiger Teil der Zieladresse

  Beispiel: WORKRAM 3F - Verlegen von IX-, Interrupttabelle und Stack an das Ende des RAM's ,auf Adresse 3F00H

Bei Kommandoausfuehrung wird der Bildschirm geloescht, das TEMO-Menue erscheint und der USER-Stack wird neu initialisiert (im Beispiel auf Adresse 3F60H).

## 5.2. Besonderheiten

Der Testmonitor beinhaltet ein Unterprogramm zum Test von Maschinenbefehlsbytes auf Laenge. Gleichzeitig wird getestet, ob es sich um einen absoluten oder relativen Sprungbefehl oder einen Unterprogrammaufruf handelt.

```
Anfangsadresse:    D8AFH
Eingangsparameter: HL -   Adresse des ersten Befehlsbytes
Ausgangsparameter: B  -   Befehlslaenge
                   CY =   Relativsprung
                   Z  =   1 absoluter Ruf/Sprung
veraenderte Register: AF, HL
```

# 6. Druckertreiber-Routinen

## 6.1. Inhalt und Bedienung

### 6.1.1. Allgemeines

Die Treiberprogramme V24K6311 und V24K6313 dienen der Ausgabe von Quellprogrammen, Assemblerprotokollen und zur Protokollierung der Arbeitsschritte im Testmonitor.

Zur Druckerausgabe gehoert der Modul M 003 V24. Hinweise ueber die Handhabung sind dieser Modulbeschreibung zu entnehmen. Sind mehrere V24 Module im Geraet gesteckt, wird jeweils der V24-Modul mit der niedrigsten Schachtnummer vom Programm eingeschaltet. Ausserdem wird der Kanal K1 am Modul fuer die Druckerausgabe genutzt.

Die hier auf dem Softwaremodul zur Verfuegung stehenden Treiberprogramme ermoeglichen die Anwendung der Drucker K6311, K6312, K6313, K6314 sowie verschiedener Schreibmaschinentypen.

Die Programme geben die Daten zeichenweise ueber den USER-Kanal 1 aus. Soll der USER-Kanal 2 genutzt werden, koennen die Programme der V24-Software-Kassette C0171 genutzt werden.Dies gilt auch fuer den Thermodrucker K6303.

### 6.1.2. Uebersicht der Drucker, die mit der vorhandenen Software bedient werden koennen

| Treiberprogramm | Druckgeraet |
|---|---|
| V24K6311 | K6311 |
| | K6312 |
| | S6010 (1200 Bit/s) |
| V24K6313 | K6313 |
| | K6314 |
| | S6005 |
| | S6010 (9600 Bit/s) |

### 6.1.3. Unterschiede der Treiberprogramme

| | K6311 | K6313 |
|---|---|---|
| Datenuebertragungsrate | 1200 Bit/s | 9600 Bit/s |

Weiterhin unterscheiden sich die Steuersequenzen fuer die Einstellung der Formularlaenge und der Formularendezeile.

## 6.2. Grundeinstellung

### 6.2.1. Treiberroutine V24K6311

Datenuebertragungsart: 1200 Bit/s
1 Stoppbit
keine Paritaet
Hardwareprotokoll

Formularlaenge: 72 Zeilen 12 Zoll
Formularendezeile: 66 Zeilen
Heftrand: 10 Zeichen

Der Code 09H wird durch 20H ersetzt.

Zum Empfang der vom Computer gesendeten Daten muss das V24-Interface der Drucker K6311 oder K6313 sowie der Schreibmaschine S6010 auf die oben angegebene Datenuebertragungsart eingestellt werden. Die dazu erforderlichen Manipulationen sind den Beschreibungen der Druckgeraete zu entnehmen.

### 6.2.2. Treiberroutine V24K6313

Datenuebertragungsart 9600 Bit/s
1 Stoppbit
keine Paritaet
Hardwareprotokoll

Formularlaenge: am Drucker einstellbar
Formularende: am Drucker einstellbar
Heftrand: 10 Zeichen

Der Code 09H wird durch 20H ersetzt.

Zum Empfang der vom Computer gesendeten Daten ist die hier angegebene Datenuebertragungsart einzustellen. Alle weiteren Bedingungen sind der Beschreibung des V24-Moduls und der Druckgeraetebeschreibung zu entnehmen.
Fuer die Betriebsbereitschaft der Schreibmaschine S6005 ist keine Einstellung des Interface notwendig.

Das V24-Interface der Schreibmaschine S6010 sollte zur Nutzung des Programms V24K6313 fuer die Datenausgabe nach der Druckgeraetebeschreibung eingestellt werden.

## 6.3. Anschlußbedingungen

Die Ansteuerung Computer- Drucker ist in der M003 V24-Modul-beschreibung enthalten (Bild 5). Ebenfalls gilt dieser Hinweis fuer die Schreibmaschinen.

## 6.4. Initialisierung

Mit dem Einstellen des Cursors auf den jeweiligen Programmnamen (V24K6311 oder V24K6313) und dem Druecken der ENTER-Taste wird der V24-Modul zur Datenuebertragung initialisiert. Das muss auch nach einem System-Reset erfolgen.

Die Datenausgabe erfolgt ueber den Anwenderkanal 1 des Systems (USER-OUT 1).

Fuer den Druckeranschluss ist der Kanal K1 des V24-Moduls, der die niedrigste Schachtnummer im KC-System besitzt, zu nutzen . V24-Module, die eine hoehere Schachtnummer haben, werden nicht genutzt und dem Anwender erst nach dem Ausschalten des untersten V24-Moduls wieder zugaenglich (siehe V24-Modulbeschreibung).

Ist kein V24-Modul gesteckt, erfolgt eine ERROR-Meldung.

Ist der Drucker off-line geschaltet, wartet der Computer, bis der Drucker on-line geschaltet ist (der Cursor wird nicht wieder eingeblendet, daher keine Weiterarbeit moeglich).

## 6.5. Editor und Assembler

Mit Anwahl des Menuewortes PRINT m n wird ein Quellprogramm bei initialisiertem Druckertreiber auf dem Drucker ausgelistet. Die Bedeutung der Parameter:

m Anzahl der Bildschirmzeilen pro Druckzeile
n Anzahl der Druckzeilen pro Druckseite

Diese Parameter werden hexadezimal angegeben. Entfaellt die Parametereingabe, werden die Zeichen auf den Bildschirm ausgegeben. Am Ende der Bildschirm- bzw. der Druckausgabe erfolgt die Abfrage 'Weiter ja/nein'.

Mit Aufruf des Assemblers und Anwahl der Option LP wird bei initialisiertem Druckertreiber ein Assemblerprotokoll ausgedruckt.

## 6.6. Protokollfunktion

Durch die Tastenkombination SHIFT und CLEAR ist es moeglich, den Drucker parallel zu schalten. Es wird also jedes auf dem Bildschirm ausgegebene Zeichen auch auf dem Drucker ausgegeben. Dies ermoeglicht es z.B., ein Disassemblerlisting zu drucken oder den Schrittbetrieb fuer den Testmonitor zu protokollieren.

## 7. Programmbeispiel

**Es wird beschrieben, wie ein Programm mit dem Namen MULT zur Multiplikation zweier Zahlen mit Vorzeichen am KC 85/3 erstellt und getestet wird.**

**Das Produkt Z=X*Y soll berechnet werden. Dabei sind X und Y ganze Zahlen von jeweils 2 Byte Laenge, die in den Registern DE bzw. BC vorgegeben werden. Das Produkt Z ist eine ganze Zahl mit Vorzeichen von 4 Byte Laenge und steht in den Registern HL, BC. Nach dem Aufruf von EDAS und dem Kommando TOP wird das Programm eingegeben.**

**Danach wird der Assembler aufgerufen.**

**Eventuell vorhandene syntaktische Fehler im Quellprogramm sind mit Hilfe des Editors zu beseitigen.**

**Zum Test des Maschinenprogramms ist EDAS mit EXIT zu verlassen.**

**Beispiel:**
**Berechnung des Produktes -5 * 51 = -255**
**(Die Eingabe und Ausgabe der Zahlen erfolgt in hexadezimaler Form.)**

```
%MULT 8005 33

      8005 * 0033 = 800000FF
```

**Auf den naechsten Seiten ist folgendes dargestellt:**

- **Anwendung der EDAS-Kommandos**
  - ***PRINT**
  - ***FIND**
  - ***SAVE**
  - ***ASM**
- **Anwendung der TEMO-Kommandos**
  - **+DISPLAY**
  - **+STEP**
  - **+de**
- **Einsatz des Disassemblers**

```
*PRINT 1 3C
```

```
;*************************************
; BEISPIEL FUER PROGRAMMERSTELLUNG     *
; MIT DEVELOPMENT-MODUL                *
;                                      *
; MULTIPLIKATIONSPROGRAMM              *
; Z = X * Y                            *
; PARAMETERUEBERGABE BEI AUFRUF UEBER  *
; DAS MENU                             *
; MULT X Y 'ENTER'                     *
;                                      *
;*************************************

        DEFW    7F7FH       ;PROLOG TEMO
        DEFM    'MULT'      ;MENU-EINTRAG
        DEFB    1           ;EPILOG
        NOP                 ;EINSPRUNG TEMO
        CALL    0F003H      ;SYSTEMAUFRUF
        DEFB    23H         ;STRINGAUSGABE
        DEFW    0D0AH       ;CRLF
        DEFW    0909H
        DEFB    0           ;STRINGENDE
        CALL    CON
MUL1    CALL    0F003H      ;SYSTEMAUFRUF
        DEFB    23H         ;AUSGABE *
        DEFM    '*'
        DEFB    0
        EX      DE,HL
        CALL    CON
MUL2    EX      DE,HL
        PUSH    HL
        PUSH    DE
        POP     BC
        POP     DE
        LD      H,11H        ;COUNTER
        LD      A,B
        XOR     D
        AND     80H          ;VORZEICHEN Z
        PUSH    AF
        LD      A,B
        LD      B,H
        LD      HL,0         ;Z=0
        RES     7,A          ;Y=|Y|
        RES     7,D          ;X=|X|
LM24    RR      H            ;Z=Z/2
        RR      L
        RRA
        RR      C
        JR      NC,NCM2
        ADD     HL,DE        ;Z=Z+X
NCM2    DJNZ    LM24
        LD      B,A          ;<HLBC>=Z
        POP     AF           ;VORZEICHEN
        OR      H
        LD      H,A
        CALL    0F003H       ;SYSTEMAUFRUF
        DEFB    23H          ;AUSGABE=
        DEFM    '='
```

```
        DEFB    0
        CALL    CON
MUL4    LD      H,B
        LD      L,C
        CALL    CON
MUL5    CALL    0F003H      ;SYSTEMAUFRUF
        DEFB    2CH         ;AUSGABE NEWLINE
        RET
CON     LD      A,H         ;ZIFFERNCONVERT.
        AND     0F0H
        RRA
        RRA
        RRA
        RRA
        CP      0AH
        JR      C,CON1
        ADD     7
CON1    ADD     30H
        CALL    0F003H      ;SYSTEMAUFRUF
        DEFB    24H         ;ZIFFERNAUSGEBE
        LD      A,H
        AND     0FH
        CP      0AH
        JR      C,CON2
        ADD     7
CON2    ADD     30H
        CALL    0F003H      ;SYSTEMAUFRUF
        DEFB    24H         ;ZIFFERNAUSGABE
        LD      A,L
        AND     0F0H
        RRA
        RRA
        RRA
        RRA
        CP      0AH
        JR      C,CON3
        ADD     7
CON3    ADD     30H
        CALL    0F003H      ;SYSTEMAUFRUF
        DEFB    24H         ;ZIFFERNAUSGABE
        LD      A,L
        AND     0FH
        CP      0AH
        JR      C,CON4
        ADD     7
CON4    ADD     30H
        CALL    0F003H      ;SYSTEMAUFRUF
        DEFB    24H         ;ZIFFERNAUSGABE
        RET

*
*FIND
TEXT    :MUL1

MUL1    CALL    0F003H      ;SYSTEMAUFRUF
        DEFB    23H         ;AUSGABE *
        DEFM    '*'
        DEFB    0
```

```
        EX      DE,HL
        CALL    CON
MUL2    EX      DE,HL
        PUSH    HL
        PUSH    DE
        POP     BC
        POP     DE
        LD      H,11H       ;COUNTER
        LD      A,B
        XOR     D
        AND     80H         ;VORZEICHEN Z
        PUSH    AF
        LD      A,B
        LD      B,H
        LD      HL,0        ;Z=0
        RES     7,A         ;Y=|Y|

*
*SAVE
NAME :BEISPIELASM
01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

*
*ASM
OPTIONS(+,2,B,L,O,P,S)?:O

END PASS 1

ERRORS: 0000

*
*ASM
OPTIONS(+,2,B,L,O,P,S)?:S

END PASS 1

ERRORS: 0000

NAME: BEISPIELCOM 3000 309F
01 02
*
*
*ASM
OPTIONS(+,2,B,L,P,S)?: LP

END PASS 1
*
3000        ;**************************************
3000        ; BEISPIEL FUER PROGRAMMERSTELLUNG     *
3000        ; MIT DEVELOPMENT-MODUL                *
3000        ;                                      *
3000        ; MULTIPLIKATIONSPROGRAMM              *
3000        ; Z=X*Y                                *
3000        ; PARAMETERUEBERGABE BEI AUFRUF UEBER  *
3000        ; DAS MENU                             *
3000        ; MULT X Y 'ENTER'                     *
3000        ;                                      *
3000        ;**************************************
```

```
3000          ;
3000 7F7F             DEFW       7F7FH     ;PROLOG FUER
3002 4D554C54         DEFM       'MULT'    ;MENU-EINTRAG
3006 01               DEFB       1
3007 00               NOP                  ;EINSPRUNG TEMO
3008 CD03F0           CALL       0F003H    ;SYSTEMAUFRUF
300B 23               DEFB       23H       ;STRINGAUSGABE
300C 0A0D             DEFW       0D0A      ;CRLF
300E 0909             DEFW       0909H
3010 00               DEFB       0         ;STRINGENDE
3011 CD5A30           CALL       CON
3014 CD03F0 MUL1      CALL       0F003H    ;SYSTEMAUFRUF
3017 23               DEFB       23H       ;AUSGABE *
3018 202A20           DEFM       '*'
301B 00               DEFB       0
301C EB               EX         DE,HL
301D CD5A30           CALL       CON
3020 EB      MUL2     EX         DE,HL
3021 E5               PUSH       HL
3022 D5               PUSH       DE
3023 C1               POP        BC
3024 D1               POP        DE
3025 2611             LD         H,11H     ;COUNTER
3027 78               LD         A,B
3028 AA               XOR        D
3029 E680             AND        80H       ;VORZEICHEN Z
302B F5               PUSH       AF
302C 78               LD         A,B
302D 44               LD         B,H
302E 210000           LD         HL,0      ;Z=0
3031 CBBF             RES        7,A       ;Y=|Y|
3033 CBBA             RES        7,D       ;X=|X|
3035 CB1C    LM24     RR         H         ;Z=Z/2
3037 CB1D             RR         L
3039 1F               RRA
303A CB19             RR         C
303C 3001             JR         NC,NCM2
303E 19               ADD        HL,DE     ;Z=Z+X
303F 10F4    NCM2     DJNZ       LM24
3041 47               LD         B,A       ;<HLBC>=Z
3042 F1               POP        AF        ;VORZEICHEN
3043 B4               OR         H
3044 67               LD         H,A
3045 CD03F0           CALL       0F003H    ;SYSTEMAUFRUF
3048 23               DEFB       23H       ;AUSGABE =
3049 203D20           DEFM       '='
304C 00               DEFB       0
304D CD5A30           CALL       CON
3050 60      MUL4     LD         H,B
3051 69               LD         L,C
3052 CD5A30           CALL       CON
3055 CD03F0 MUL5      CALL       0F003H     ;SYSTEMAUFRUF
3058 2C               DEFB       2CH        ;AUSGABE NEWLINE
3059 C9               RET
305A 7C      CON      LD         A,H        ;ZIFFERNCONVERT.
305B E6F0             AND        0F0H
305D 1F               RRA
305E 1F               RRA
```

```
305F 1F             RRA
3060 1F             RRA
3061 FE0A           CP          0AH
3063 3802           JR          C,CON1
3065 C607           ADD         7
3067 C630     CON1  ADD         30H
3069 CD03F0         CALL        0F003H    ;SYSTEMAUFRUF
306C 24             DEFB        24H       ;ZIFFERNAUSGABE
306D 7C             LD          A,H
306E E60F           AND         0FH
3070 FE0A           CP          0AH
3072 3802           JR          C,CON2
3074 C607           ADD         7
3076 C630     CON2  ADD         30H
3078 CD03F0         CALL        0F003H    ;SYSTEMAUFRUF
307B 24             DEFB        24H       ;ZIFFERNAUSGABE
307C 7D             LD          A,L
307D E6F0           AND         0F0H
307F 1F             RRA
3080 1F             RRA
3081 1F             RRA
3082 1F             RRA
3083 FE0A           CP          0AH
3085 3802           JR          C,CON3
3087 C607           ADD         7
3089 C630     CON3  ADD         30H
308B CD03F0         CALL        0F003H    ;SYSTEMAUFRUF
308E 24             DEFB        24H       ;ZIFFERNAUSGABE
308F 7D             LD          A,L
3090 E60F           AND         0FH
3092 FE0A           CP          0AH
3094 3802           JR          C,CON4
3096 C607           ADD         7
3098 C630     CON4  ADD         30H
309A CD03F0         CALL        0F003H    ;SYSTEMAUFRUF
309D 24             DEFB        24H       ;ZIFFERNAUSGABE
309E C9             RET

ERRORS: 0000

*EXIT

%TEMO
+
+
+STEP 3007
 AF   BC   DE   HL   PC   SP   I
361A 0055 8E16 1116 3008 0130 01
 IX    IY FLAGS OP.CODE
01F0 68BF HN       CD03F0
3008 CALL 0F003H
 AF   BC   DE   HL   PC   SP   I
0054 0055 8E16 1116 3011 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF ZHP     CD5A30
3011 CALL 0305AH
 AF   BC   DE   HL   PC   SP   I
361A 0055 8E16 1116 3014 0130 01
```

```
 IX   IY FLAGS OP.CODE
01F0 68BF HN      CD03F0
3014 CALL 0F003H
*AF   BC   DE   HL   PC   SP   I
0054 0055 8E16 1116 301C 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF ZHP      EB
301C EX DE,HL
 AF   BC   DE   HL   PC   SP   I
0054 0055 1116 8E16 301D 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF ZHP      CD5A30
301D CALL 0305AH
+
+de 55
+bc 8005
+STEP 3007
AF   BC   DE   HL   PC   SP   I
0054 8005 0055 8E16 3008 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF ZHP      CD03F0
3008 CALL 0F003H
AF   BC   DE   HL   PC   SP   I
0054 8005 0055 8E16 3011 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF ZHP      CD5A30
3011 CALL 0305AH
 AF   BC   DE   HL   PC   SP   I
361A 8005 0055 8E16 3014 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF HN       CD03F0
3014 CALL 0F003H
 AF   BC   DE   HL   PC   SP   I
0054 8005 0055 8E16 301C 0130 01
 IX IY FLAGS OP.CODE
01F0 68BF ZHP    EB
301C EX DE,HL
 AF   BC   DE   HL   PC   SP   I
0054 8005 8E16 0055 301D 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF ZHP      CD5A30
301D CALL 0305AH
 AF   BC   DE   HL   PC   SP   I
351A 8005 8E16 0055 3020 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF HN      EB
3020 EX DE,HL
 AF   BC   DE   HL   PC   SP   I
351A 8005 0055 8E16 3021 0130 01
IX   IY FLAGS OP.CODE
01F0 68BF HN     E5
3021 PUSH HL
 AF   BC   DE   HL   PC   SP   I
351A 8005 0055 8E16 3022 012E 01
 IX   IY FLAGS OP.CODE
01F0 68BF HN      D5
3022 PUSH DE
 AF   BC   DE   HL   PC   SP   I
```

```
351A 8005 0055 8E16 3023 012C 01
 IX   IY FLAGS OP.CODE
01F0 68BF HN      C1
3023 POP BC
 AF   BC   DE   HL   PC   SP   I
351A 0055 0055 8E16 3024 012E 01
 IX   IY FLAGS OP.CODE
01F0 68BF HN      D1
3024 POP DE
 AF   BC   DE   HL   PC   SP   I
351A 0055 8E16 8E16 3025 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF HN      2611
3025 LD H,011H
 AF   BC   DE   HL   PC   SP   I
351A 0055 8E16 1116 3027 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF HN      78
3027 LD A,B
 AF   BC   DE   HL   PC   SP   I
001A 0055 8E16 1116 3028 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF HN      AA
3028 XOR D
 AF   BC   DE   HL   PC   SP   I
8E8C 0055 8E16 1116 3029 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF SP      E680
3029 AND 080H
 AF   BC   DE   HL   PC   SP   I
8090 0055 8E16 1116 302B 0130 01
 IX   IY FLAGS OP.CODE
01F0 68BF SH      F5
302B PUSH AF
 AF   BC   DE   HL   PC   SP   I
8090 0055 8E16 1116 302C 012E 01
 IX   IY FLAGS OP.CODE
01F0 68BF SH      78
302C LD A,B
+
+EXIT
%
%
%DISASS 3007 30A0
3007 NOP
3008 CALL 0F003H
300B DEFB 023H
300C DEFB 00AH
300D DEFB 00DH
300E DEFB 009H
300F DEFB 009H
3010 DEFB 000H
3011 CALL 0305AH
3014 CALL 0F003H
3017 DEFB 023H
3018 DEFM ' * '
301B DEFB 000H
301C EX DE,HL
```

```
301D CALL 0305AH
3020 EX DE,HL
3021 PUSH HL
3022 PUSH DE
3023 POP BC
3024 POP DE
3025 LD H,011H
3027 LD A,B
3028 XOR D
3029 AND 080H
302B PUSH AF
302C LD A,B
302D LD B,H
302E LD HL,00000H
3031 RES 7,A
3033 RES 7,D
3035 RR H
3037 RR L
3039 RRA
303A RR C
303C JR NC,0303FH
303E ADD HL,DE
303F DJNZ 03035H
3041 LD B,A
3042 POP AF
3043 OR H
3044 LD H,A
3045 CALL 0F003H
3048 DEFB 023H
3049 DEFM ' = '
304C DEFB 000H
304D CALL 0305AH
3050 LD H,B
3051 LD L,C
3052 CALL 0305AH
3055 CALL 0F003H
3058 DEFB 02CH
3059 RET
305A LD A,H
305B AND 0F0H
305D RRA
305E RRA
305F RRA
3060 RRA
3061 CP 00AH
3063 JR C,03067H
3065 ADD 007H
3067 ADD 030H
3069 CALL 0F003H
306C DEFB 024H
306D LD A,H
306E AND 00FH
3070 CP 00AH
3072 JR C,03076H
3074 ADD 007H
3076 ADD 030H
3078 CALL 0F003H
307B DEFB 024H
```

```
307C LD A,L
307D AND 0F0H
307F RRA
3080 RRA
3081 RRA
3082 RRA
3083 CP 00AH
3085 JR C,03089H
3087 ADD 007H
3089 ADD 030H
308B CALL 0F003H
308E DEFB 024H
308F LD A,L
3090 AND 00FH
3092 CP 00AH
3094 JR C,03098H
3096 ADD 007H
3098 ADD 030H
309A CALL 0F003H
309D DEFB 024H
309E RET
309F NOP
%
```

# 8. Literatur


Classen, L.: Programmierung des Mikroprozessorsystems U880-K1520. Reihe Automatisierungstechnik. VEB Verlag TEchnik, Berlin 1982

Schwarze, W.; Meyer, G.; Eckard, D: Mikrorechner. Wirkungsweise Programmierung, Applikation. VEB Verlag Technik, Berlin 1980

Robotron-Systemmdokumentation MOS K1520. SCPX 1526, Anleitung fuer den Programmierer Teil II. VEB Robotron Buchungsmaschinenwerk, Karl-Marx-Stadt 1984.

Lampe, D.; Jorke, G.; Wengel,N.: Algorithmen der Mikrorechentechni. VEB Verlag Technik, Berlin 1983.

Z80-Assemblersprache. Benutzerhandbuch in deutscher Sprache. ZILOG, 1977.

Klein, M.; Klein, R.-D.: Z80-Applikationsbuch. Franzisverlag GmbH. Muenchen 1983.

Osborn, A.: Einfuehrung in die Mikrocomputer-Technik. te-wi Verlag GmbH, Muenchen 1982.

Systemhandbuch. VEB Mikroelektronik "Wilhelm Pieck" Muehlhausen

Beschreibung zu M003 V24. VEB Mikroelektronik "Wilhelm Pieck" Muehlhausen.

Beschreibung zur Programmkassette C0171 V24-Software. VEB Mikroelektronik "Wilhelm Pieck" Muehlhausen.

Befehlsbeschreibung U880D. VEB Mikroelektronik "Karl-Marx" Erfurt, Stammbetrieb.

# Anhang

## A. Befehlscode-Tabelle

### 8-Bit-Ladebefehle

| | A | B | C | D | E | H | L | (HL) | (BC) | (DE) | (nn) | n |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LD A,. | 7F | 78 | 79 | 7A | 7B | 7C | 7D | 7E | 0A | 1A | 3AXXXX | 3EXX |
| LD B,. | 47 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | | | | 06XX |
| LD C,. | 4F | 48 | 49 | 4A | 4B | 4C | 4D | 4E | | | | 0EXX |
| LD D,. | 57 | 50 | 51 | 52 | 53 | 55 | 55 | 56 | | | | 16XX |
| LD E,. | 5F | 58 | 59 | 5A | 5B | 5C | 5D | 5E | | | | 1EXX |
| LD H,. | 67 | 60 | 61 | 62 | 63 | 66 | 65 | 66 | | | | 26XX |
| LD L,. | 6F | 68 | 69 | 6A | 6B | 6C | 6D | 6E | | | | 2EXX |
| LD (HL),. | 77 | 70 | 71 | 72 | 73 | 77 | 75 | | | | | 36XX |
| LD (BC),. | 02 | | | | | | | | | | | |
| LD (DE),. | 12 | | | | | | | | | | | |
| LD (nn),. | 32XXXX | | | | | | | | | | | |

| | A | B | C | D | E | H | L |
|---|---|---|---|---|---|---|---|
| LD .,(IX+d) | DD7EXX | DD46XX | DD4EXX | DD56XX | DD5EXX | DD66XX | DD6EXX |
| LD .,(IY+d) | FD7EXX | FD46XX | FD4EXX | FD56XX | FD5EXX | FD66XX | FD6EXX |
| LD (IX+d),. | DD77XX | DD70XX | DD71XX | DD72XX | DD73XX | DD74XX | DD75XX |
| LD (IY+d),. | FD77XX | FD70XX | FD71XX | FD72XX | FD73XX | FD74XX | FD75XX |

LD (IX+d),n DD36XXXX LD (IY+d),n FD36XXXX

| | | S | Z | H | P/V | N | C |
|---|---|---|---|---|---|---|---|
| LD A,I | ED57 | * | * | 0 | F | 0 | - |
| LD A,R | ED5F | * | * | 0 | F | 0 | - |
| LD I,A | ED47 | - | - | - | - | - | - |
| LD R,A | ED4F | - | - | - | - | - | - |

### 16-Bit-Ladebefehle

| | BC | DE | HL | SP | IX | IY |
|---|---|---|---|---|---|---|
| LD ..,nn | 01XXXX | 11XXXX | 21XXXX | 31XXXX | DD21XXXX | FD21XXXX |
| LD ..,(nn) | ED4BXXXX | ED5BXXXX | 2AXXXX | ED7BXXXX | DD2AXXXX | FD2AXXXX |
| LD (nn),.. | ED43XXXX | ED53XXXX | 22XXXX | ED73XXXX | DD22XXXX | FD22XXXX |
| LD SP | | | F9 | | DDF9 | FDF9 |

| | BC | DE | HL | AF | IX | IY |
|---|---|---|---|---|---|---|
| PUSH .. | C5 | D5 | E5 | F5 | DDE5 | FDE5 |
| POP .. | C1 | D1 | E1 | F1 | DDE1 | FDE1 |

### Registeraustauschbefehle

EX (SP),HL E3 EX DE,HL EB
EX (SP),IX DDE3 EX AF,AF' 08
EX (SP),IY FDE3 EXX D9 BC-BC' DE-DE' HL-HL'

## Blocktransfer- und -suchbefehle

| | | S | Z | H | P/V | N | C | |
|---|---|---|---|---|---|---|---|---|
| LDI | EDA0 | - | - | 0 | * | 0 | - | LD(DE),(HL);INC HL;INC DE;DEC BC |
| LDIR | EDB0 | - | - | 0 | 0 | 0 | - | wie LDI, wiederholen bis BC=0 |
| LDD | EDA8 | - | - | 0 | * | 0 | - | LD(DE),(HL);DEC HL;DEC DE;DEC BC |
| LDDR | EDB8 | - | - | 0 | 0 | 0 | - | wie LDD, wiederholen bis BC=0 |
| CPI | EDA1 | * | * | * | * | 1 | - | CP A,(HL);INC HL;DEC BC |
| CPIR | EDB1 | * | * | * | * | 1 | - | wie CPI,wiederh.b.BC=0 od.A=(HL) |
| CPD | EDA9 | * | * | * | * | 1 | - | CP A,(HL);DEC HL;DEC BC |
| CPDR | EDB9 | * | * | * | * | 1 | - | wie CPD,wiederh.b.BC=0 od.A=(HL) |

## 8-Bit-Arithmetik- und -Logikbefehle

| | B | C | D | E | H | L | (HL) | A | n | (IX+d) | (IY+d) | S | Z | H | P/V | N | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD . | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | C6XX | DD86XX | FD86XX | * | * | * | V | 0 | * |
| ADC . | 88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F | CEXX | DD8EXX | FD8EXX | * | * | * | V | 0 | * |
| SUB . | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | D6XX | DD96XX | FD96XX | * | * | * | V | 1 | * |
| SBC . | 98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F | DEXX | DD9EXX | FD9EXX | * | * | * | V | 1 | * |
| AND . | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | E6XX | DDA6XX | FDA6XX | * | * | 1 | P | 0 | 0 |
| XOR . | A8 | A9 | AA | AB | AC | AD | AE | AF | EEXX | DDAEXX | FDAEXX | * | * | 1 | P | 0 | 0 |
| OR . | B0 | B1 | B2 | B3 | B4 | B5 | B6 | B7 | F6XX | DDB6XX | FDB6XX | * | * | 1 | P | 0 | 0 |
| CP . | B8 | B9 | BA | BB | BC | BD | BE | BF | FEXX | DDBEXX | FDBEXX | * | * | * | V | 1 | * |
| INC . | 04 | 0C | 14 | 1C | 24 | 2C | 34 | 3C | | DD34XX | FD34XX | * | * | * | V | 0 | - |
| DEC . | 05 | 0D | 15 | 1D | 25 | 2D | 35 | 3D | | DD35XX | FD35XX | * | * | * | V | 1 | - |

| | | S | Z | H | P/V | N | C | |
|---|---|---|---|---|---|---|---|---|
| DAA | 27 | * | * | * | P | - | * | BCD-Korrektur im Akku |
| CPL | 2F | - | - | 1 | - | 1 | - | Komplementiere Akku (1er-Komplement) |
| SCF | 37 | - | - | 0 | - | 0 | 1 | Setze Carry-Flag |
| CCF | 3F | - | - | x | - | 0 | * | Komplementiere Carry-Flag |
| NEG | ED44 | * | * | * | V | 1 | * | Komplementiere Akku (2er-Komplement) |

## 16-Bit-Arithmetikbefehle

| | BC | DE | HL | SP | IX | IY | S | Z | H | P/V | N | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD HL,.. | 09 | 19 | 29 | 39 | | | - | - | x | - | 0 | * |
| ADC HL,.. | ED4A | ED5A | ED6A | ED7A | | | * | * | x | V | 0 | * |
| SBC HL,.. | ED42 | ED52 | ED62 | ED72 | | | * | * | x | V | 1 | * |
| ADD IX,.. | DD09 | DD19 | | DD39 | DD29 | | - | - | x | - | 0 | * |
| ADD IY,.. | FD09 | FD19 | | FD39 | | FD29 | - | - | x | - | 0 | * |
| INC .. | 03 | 13 | 23 | 33 | DD23 | FD23 | - | - | - | - | - | - |
| DEC .. | 0B | 1B | 2B | 3B | DD2B | FD2B | - | - | - | - | - | - |

## Sprung- und Unterprogrammbefehle

| | Z | NZ | C | NC | PE | PO | M | P |
|---|---|---|---|---|---|---|---|---|
| JP | CAXXXX | C2XXXX | DAXXXX | D2XXXX | EAXXXX | E2XXXX | FAXXXX | F2XXXX |
| CALL | CCXXXX | C4XXXX | DCXXXX | D4XXXX | ECXXXX | E4XXXX | FCXXXX | F4XXXX |
| RET | C8 | C0 | D8 | D0 | E8 | E0 | F8 | F0 |
| JR | 28XX | 20XX | 38XX | 30XX | | | | |

| | unbedingt | (HL) | (IX) | (IY) | RST | 00 | 08 | 10 | 18 | 20 | 28 | 30 | 38 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| JP | C3XXXX | E9 | DDE9 | FDE9 | | C7 | CF | D7 | DF | E7 | EF | F7 | FF |
| CALL | CDXXXX | | | | | | | | | | | | |
| JR | 18XX | | | | | | | | | | | | |

| | | |
|---|---|---|
| DJNZ | 10XX | DEC B;JR NZ,nn |
| RETI | ED4D | zurueck vom Interrupt |
| RETN | ED45 | zurueck vom nicht maskierbaren Interrupt |

## Rotations- und Verschiebebefehle

| | | S | Z | H | P/V | N | C | |
|---|---|---|---|---|---|---|---|---|
| RLCA | 07 | - | - | 0 | - | 0 | * | Rotiere Akku links |
| RRCA | 0F | - | - | 0 | - | 0 | * | Rotiere Akku rechts |
| RLA | 17 | - | - | 0 | - | 0 | * | Rotiere Akku links durch Carry |
| RRA | 1F | - | - | 0 | - | 0 | * | Rotiere Akku rechts durch Carry |
| RLD | ED6F | * | * | 0 | P | 0 | - | Rot.Ziffern links zw. Akku und (HL) |
| RRD | ED67 | * | * | 0 | P | 0 | - | Rot.Ziffern rechts zw.Akku und (HL) |

| | B | C | D | E | H | L | (HL) | A | (IX+d) | (IY+d) |
|---|---|---|---|---|---|---|---|---|---|---|
| RLC | CB00 | CB01 | CB02 | CB03 | CB04 | CB05 | CB06 | CB07 | DDCBXX06 | FDCBXX06 |
| RRC | CB08 | CB09 | CB0A | CB0B | CB0C | CB0D | CB0E | CB0F | DDCBXX0E | FDCBXX0E |
| RL | CB10 | CB11 | CB12 | CB13 | CB14 | CB15 | CB16 | CB17 | DDCBXX16 | FDCBXX16 |
| RR | CB18 | CB19 | CB1A | CB1B | CB1C | CB1D | CB1E | CB1F | DDCBXX1E | FDCBXX1E |
| SLA | CB20 | CB21 | CB22 | CB23 | CB24 | CB25 | CB26 | CB27 | DDCBXX26 | FDCBXX26 |
| SRA | CB28 | CB29 | CB2A | CB2B | CB2C | CB2D | CB2E | CB2F | DDCBXX2E | FDCBXX2E |
| SRL | CB38 | CB39 | CB3A | CB3B | CB3C | CB3D | CB3E | CB3F | DDCBXX3E | FDCBXX3E |

| | S | Z | H | P/V | N | C | |
|---|---|---|---|---|---|---|---|
| RLC/RRC | * | * | 0 | P | 0 | * | Rotiere Register links/rechts |
| RL/RR | * | * | 0 | P | 0 | * | Rotiere Register links/rechts durch Carry |
| SLA/SRA | * | * | 0 | P | 0 | * | Schiebe Register links/rechts arithm. |
| SRL | * | * | 0 | P | 0 | * | Schiebe Register rechts logisch |

## Einzelbitbefehle

| | B | C | D | E | H | L | (HL) | A | (IX+d) | (IY+d) |
|---|---|---|---|---|---|---|---|---|---|---|
| BIT 0 | CB40 | CB41 | CB42 | CB43 | CB44 | CB45 | CB46 | CB47 | DDCBXX46 | FDCBXX46 |
| BIT 1 | CB48 | CB49 | CB4A | CB4B | CB4C | CB4D | CB4E | CB4F | DDCBXX4E | FDCBXX4E |
| BIT 2 | CB50 | CB51 | CB52 | CB53 | CB54 | CB55 | CB56 | CB57 | DDCBXX56 | FDCBXX56 |
| BIT 3 | CB58 | CB59 | CB5A | CB5B | CB5C | CB5D | CB5E | CB5F | DDCBXX5E | FDCBXX5E |
| BIT 4 | CB60 | CB61 | CB62 | CB63 | CB64 | CB65 | CB66 | CB67 | DDCBXX66 | FDCBXX66 |
| BIT 5 | CB68 | CB69 | CB6A | CB6B | CB6C | CB6D | CB6E | CB6F | DDCBXX6E | FDCBXX6E |
| BIT 6 | CB70 | CB71 | CB72 | CB73 | CB74 | CB75 | CB76 | CB77 | DDCBXX76 | FDCBXX76 |
| BIT 7 | CB78 | CB79 | CB7A | CB7B | CB7C | CB7D | CB7E | CB7F | DDCBXX7E | FDCBXX7E |
| RES 0 | CB80 | CB81 | CB82 | CB83 | CB84 | CB85 | CB86 | CB87 | DDCBXX86 | FDCBXX86 |
| RES 1 | CB88 | CB89 | CB8A | CB8B | CB8C | CB8D | CB8E | CB8F | DDCBXX8E | FDCBXX8E |
| RES 2 | CB90 | CB91 | CB92 | CB93 | CB94 | CB95 | CB96 | CB97 | DDCBXX96 | FDCBXX96 |
| RES 3 | CB98 | CB99 | CB9A | CB9B | CB9C | CB9D | CB9E | CB9F | DDCBXX9E | FDCBXX9E |
| RES 4 | CBA0 | CBA1 | CBA2 | CBA3 | CBA4 | CBA5 | CBA6 | CBA7 | DDCBXXA6 | FDCBXXA6 |

```
RES 5 CBA8 CBA9 CBAA CBAB CBAC CBAD CBAE CBAF DDCBXXAE FDCBXXAE
RES 6 CBB0 CBB1 CBB2 CBB3 CBB4 CBB5 CBB6 CBB7 DDCBXXB6 FDCBXXB6
RES 7 CBB8 CBB9 CBBA CBBB CBBC CBBD CBBE CBBF DDCBXXBE FDCBXXBE
SET 0 CBC0 CBC1 CBC2 CBC3 CBC4 CBC5 CBC6 CBC7 DDCBXXC6 FDCBXXC6
SET 1 CBC8 CBC9 CBCA CBCB CBCC CBCD CBCE CBCF DDCBXXCE FDCBXXCE
SET 2 CBD0 CBD1 CBD2 CBD3 CBD4 CBD5 CBD6 CBD7 DDCBXXD6 FDCBXXD6
SET 3 CBD8 CBD9 CBDA CBDB CBDC CBDD CBDE CBDF DDCBXXDE FDCBXXDE
SET 4 CBE0 CBE1 CBE2 CBE3 CBE4 CBE5 CBE6 CBE7 DDCBXXE6 FDCBXXE6
SET 5 CBE8 CBE9 CBEA CBEB CBEC CBED CBEE CBEF DDCBXXEE FDCBXXEE
SET 6 CBF0 CBF1 CBF2 CBF3 CBF4 CBF5 CBF6 CBF7 DDCBXXF6 FDCBXXF6
SET 7 CBF8 CBF9 CBFA CBFB CBFC CBFD CBFE CBFF DDCBXXFE FDCBXXFE

Flagbeeinflussung:      S Z H P/V N C
                  BIT   x * 1  x  0 -  Komplement des Bits in Z
                  RES   - - -  -  - -  0 in Bit
                  SET   - - -  -  - -  1 in Bit
```

## CPU-Steuerbefehle

```
            S Z H P/V N C
NOP   00    - - -  -  - -  Leerbefehl
HALT  76    - - -  -  - -  NOP bis RESET oder Interrupt
DI    F3    - - -  -  - -  Interrupts sperren
EI    FB    - - -  -  - -  Interrupts freigeben
IM 0  ED46  - - -  -  - -  Interrupt-Modus 0
IM 1  ED56  - - -  -  - -  Interrupt-Modus 1
IM 2  ED5E  - - -  -  - -  Interrupt-Modus 2
```

Ein- und Ausgabebefehle

```
          A    B    C    D    E    H    L     S Z H P/V N C
IN  .,(C) ED78 ED40 ED48 ED50 ED58 ED60 ED68  * * *  P  0 -
OUT (C),. ED79 ED41 ED49 ED51 ED59 ED61 ED69  - - -  -  - -

               S Z H P/V N C
IN  A,(n) DBXX - - -  -  - -
OUT (n),A D3XX - - -  -  - -
INI       EDA2 x * x  x  1 -  IN (HL),(C);INC HL;DEC B
INIR      EDB2 x 1 x  x  1 -  wie INI,wiederholen solange B<>0
IND       EDAA x * x  x  1 -  IN (HL),(C);DEC HL;DEC B
INDR      EDBA x 1 x  x  1 -  wie IND,wiederholen solange B<>0
OUTI      EDA3 x * x  x  1 -  OUT (C),(HL);INC HL;DEC B
OTIR      EDB3 x 1 x  x  1 -  wie OUTI,wiederholen solange B<>0
OUTD      EDAB x * x  x  1 -  OUT (C),(HL);DEC HL;DEC B
OTDR      EDBB x 1 x  x  1 -  wie OUTD,wiederholen solange B<>0
```

## Flag-Register

```
Bit   7  6  5  4  3  2  1  0
      S  Z  X  H  X P/V N  C

                      gesetzt  nicht gesetzt  wird bei
C Carry-Flag               C  NC             Uebertrag von Bit 7
N Add-/Subtract-Flag                         Subtraktion
P/V Parity-/Overflow-Flag  PE PO             gerader Paritaet
H Half-Carry-Flag                            Uebertrag von Bit 3
Z Zero-Flag                Z  NZ             Ergebnis 0
S Sign-Flag                M  P              negatives Ergebnis
X nicht verwendet
```

```
Beeinflussung:      - unveraendert
                    1 gesetzt
                    0 zurueckgesetzt
                    * entsprechend dem Ergebnis der Operation
                      (gesetzt wenn erfuellt
                      zurueckgesetzt wenn nicht erfuellt)
                    x unbestimmt
                    V Overflow-Funktion
                    P Parity-Funktion
                    F Inhalt des Interrupt-Flip-Flops IFF2
```

## B. Pseudobefehle des Assemblers

```
        ORG     nn      - Adresszaehlerzuordnung
Marke:  EQU     nn      - Wertzuweisung fuer Marke
        DEFB    n       - Legt n auf die naechste Speicherstelle
        DEFW    nn      - Legt nn auf die naechsten zwei Speicher-
                          stellen (dabei zuerst niederwertiger,
                          dann hoeherwertiger Teil)
        DEFM    'Text'  - Legt auf die naechsten Speicherstellen
                          die ASCII-Werte der Zeichen des Textes
```

Dabei sind:

```
        n   -  8-Bit-Konstante
        nn  -  16-Bit-Konstante
```

## C. Übersicht Editor/Assembler

**Editor**

```
MENU-   Anzeige aller Kommandofunktionen
EXIT-   Verlassen von EDAS und Rueckkehr zur CAOS-Systemschleife
CLEAR-  Loeschen des gesamten Textspeichers, Markentabelle und
        Objektprogramm bleiben erhalten
SAVE-   Abspeichern des im Hauptspeicher befindlichen Quelltextes
        auf Kassette
LOAD-   Einlesen eines, mit dem EDAS-SAVE-Kommando
        abgespeicherten ,Quelltextes von Kassette
PRINT-  Ausdruck des Textspeichers
ASM-    Aufruf des Assemblers
TOP-    Cursor auf Textanfang mit Anzeige der 1. Textseite
BOTOM-  Cursor auf die Zeile nach dem Textende mit Anzeige der
        letzten Textseite
EDIT-   Anzeige der zuletzt bearbeiteten Textseite
VERIFY- Ueberpruefen eines auf Magnetband gespeicherten
        Programms auf fehlerfreie Aufzeichnung
```

## Tastenfunktionen

```
F1       - Tabulator setzen
F2       - Wiederholung Textsuche (s. FIND)
BRK      - Ende Editier-Modus,Rueckkehr ins Menue
INS      - Einfuegen eines Leerzeichens links vom Cursor
DEL      - Loeschen eines Zeichens mit Verdichten der Zeile
CLR      - Loeschen eines Zeichens ohne Verdichten der Zeile
HOME     - Cursor zum linken oberen Bildschirmrand
CUU      - Cursor zum Zeilenanfang der vorherigen Zeile
CUD      - Cursor zum Zeilenanfang der folgenden Zeile
CUL      - Cursor ein Zeichen nach links
CUR      - Cursor ein Zeichen nach rechts
ENTER    - Einfuegen einer Leerzeile nach Cursorzeile

SHIFT+HOME  - Loeschen der Cursorseite
SHIFT+DEL   - Loeschen der Cursorzeile
SHIFT+CUL   - Cursor zum Zeilenanfang
SHIFT+CUU   - Eine Seite zurueckblaettern     *1
SHIFT+CUD   - Eine Seite vorwaertsblaettern   *1

*1 Cursor bleibt an der vorherigen Position stehen
```

# Fehlermeldungen

**BRK, SHIFT+CUU, SHIFT+CUD kann die Anzeige >>>NO MEMORY<<< erfolgen - keine vollstaendige Textuebernahme moeglich**

## Assembler

**+ - Hinzufuegen einer neuen Markentabelle zu einer vorhandenen**
**2 - Zweiter Pass wird durchgefuehrt**
**B - Kurzform (Brief) der Bildschirmausgabe**

**L - Listenausgabe eines Assemblerprogramms**
**O - Speichern eines Programms**
**P - Druck einer Assemblerliste**
**S - Kassettenausgabe eines Assemblerprogramms**

## Fehlermitteilungen des Assemblers

**"1" - Es fehlt ein Semikolon in Kommentarzeile**
**"2" - Marke mehrfach definiert**
**"3" - Marke nicht definiert**
**"4" - Falsche Mnemonik (Befehl unbekannt)**
**"5" - Falsches Zahlenformat**
**"6" - Operandenfehler bei JR,IX oder IY (ausserhalb des Bereiches -128,...,+127)**
**"7" - Keine Marke in EQU-Anweisung**
**"8" - Hochkomma fehlt**
**"9" - Operandenfehler bei EX-Befehl**
**"A" - Falsche Flagbedingung (zulaessig sind Z,NZ,C,NC,PE,PO,M,P)**
**"B" - Plus, Minus oder Komma in Operanden fehlt**

## D. Übersicht Disassembler

| Kommando | Funktion |
|---|---|
| DISASS aaaa bbbb (nn) (oooo) | Aufruf des Disassemblers |
| CDISASS aaaa bbbb oooo | Ausgabe des erzeugten Quelltextes auf Kassette |

## E. Übersicht Testmonitor

| Kommando | Funktion |
|---|---|
| TEMO (n) | Start des Testmonitors (Kaltstart)<br>- ohne Parameter - zusaetzliche Aktivierung des Disassemblers<br>- beliebige Parameter - keine Aktivierung des Disassemblers |
| RETEMO | Start des Testmonitors (Warmstart)<br>-USER-Register bleiben unveraendert |
| MENU | Ausschreiben des Testmonitormenues |
| ARITH aaaa bbbb | Anzeige der Summe Differenz und des Sprungoffsets zweier Adressen |
| BREAK nnnn | Vereinbarung eines Unterbrechungspunktes im RAM |
| GO nnnn | Sprung in ein Anwenderprogramm mit Initialisierung des Unterbrechungspunktes |
| STEP (nnnn) | Einzelbefehlsabarbeitung ab nnnn; fehlt nnnn wird der letzte Wert des USER-PC verwendet |
| COPY aaaa bbbb cccc | Kopieren eines Speicherbereiches |
| MOVE aaaa bbbb cccc | Verschieben eines Speicherbereiches |
| REG a nnnn | Belegen eines Anwenderdoppelregisters |
| (reg) nnnn | Belegen von Registern<br>(reg) - Name des Registers in Kleinbuchstaben |
| SBKR nnnn | Vereinbarung eines Haltepunktes |
| IN aaaa | Anzeige eines Datenbytes vom Eingabekanal |
| OUT aaaa nn | Ausgabe eines Datenbytes zum Ausgabekanal |
| DISPLAY aaaa bbbb cc | Ausgabe HEX/ASCII-Dump |

| | |
|---|---|
| **EXIT** | **Rueckkehr in das Betriebsystemgrundmenue** |
| **MODIFY nnnn** | **Anzeige und Veraenderung von Speicherzellen** |
| **FILL aaaa bbbb nn** | **Beschreiben eines Speicherbereiches mit einem Byte** |
| **CHSUM aaaa bbbb** | **Berechnen eines Pruefpolynoms** |
| **WORKRAM aa** | **Verlegen von IX, Interrupttabellen und Stack** |

## F. ASCII-Code-Tabelle ( KC 85/3 )

| HW / LW | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | DUMMY | HOME | SPACE | 0 | @ | P | c | p |
| 1 | CLEAR | PAGE | ! | 1 | A | Q | a | q |
| 2 | ESC | SCROL | " | 2 | B | R | b | r |
| 3 | BREAK | STOP | # | 3 | C | S | c | s |
| 4 | - | CLICK | $ | 4 | D | T | d | t |
| 5 | - | - | % | 5 | E | U | e | u |
| 6 | - | SHIFT L | & | 6 | F | V | f | v |
| 7 | BEEP | - | ' | 7 | G | W | g | w |
| 8 | CUL | CEL | ( | 8 | H | X | h | x |
| 9 | CUR | CCR | ) | 9 | I | Y | i | y |
| A | CUD | INS | * | : | J | Z | j | z |
| B | CUU | - | + | ; | K |  | k | { |
| C | CLS | LIST | , | < | L | \| | l | \| |
| D | CR | RUN | - | = | M |  | m | } |
| E | - | CONT | . | > | N | ^ | n | ~ |
| F | HCOPY | DEL | / | ? | O | _ | o |  |

DUMMY - Fuellzeichen
CLEAR - Loeschen Zeichen
ESC - Loeschen Zeile
BREAK - Programmende *
BEEP - Signaltonausgabe
CUL - Cursor nach links
CUR - Cursor nach rechts
CUD - Cursor nach unten
CUU - Cursor nach oben
CLS - Loeschen Bildschirm
CR - Cursor an den Anfang der naechsten Zeile (ENTER)
HCOPY - Aufruf Sonderprogramm; z.B. Hardcopy
HOME - Cursor nach links oben
PAGE - Umschalten PAGE-Modus
SCROL - Umschalten SCROLLING-Modus
STOP - *
CLICK - Ein- und Ausschalten des Tastenclicks
SHIFT L - SHIFT LOCK *
CEL - Cursor auf Zeilenende *
CCR - Cursor auf Zeilenanfang
INS - Einfuegen Zeichen
LIST - *
RUN - *
CONT - *
DEL - Loeschen Zeichen und verdichten

* - Keine Funktion in der CRT-Routine

## G. Modulhandhabung

**Welcher Steckplatz ?**

**Der Modul ist zur Nutzung in einem Steckplatz des Systems zu kontaktieren. Der Software-Modul kann prinzipiell in jedem Modulsteckplatz betrieben werden. Dies gilt auch fuer Aufsaetze mit weiteren Modulsteckplaetzen. Hierbei ist zu beachten, dass die Modulprioritaetskette geschlossen bleibt.**

**Zur Modulsteuerung im Computer verfuegt der Modul ueber eine Modulprioritaetsschaltung. Sind mehrere Module mit gleicher Anfangsadresse eingeschaltet, so sichert die Prioritaetsschaltung, dass beim Zugriff des Prozessors nur der Speichermodul aktiviert wird, der sich auf der niedrigsten Modulsteckplatzadresse befindet. Soll im weiteren Verlauf der Modul mit der niedrigsten Prioritaet (hoechste Modulsteckplatzadresse) erreicht werden, muessen alle in der Prioritaetskette vorher liegenden Module mit gleicher Anfangsadresse ausgeschaltet sein (zur naeheren Erlaeuterung finden Sie im Punkt Adressierung einige Beispiele).**

**Hieraus ist zu entnehmen, dass erst im Grundgeraet der Steckplatz 08 (rechts), dann der Steckplatz 0C (links) und anschliessend erst weitere Steckplaetze von Erweiterungsaufsaetzen in vorgesehener Reihenfolge zu belegen sind.**

## Kontaktierung

**A C H T U N G !**

**Das Stecken sowie das Entfernen des Moduls aus dem Steckplatz darf nur im ausgeschalteten Zustand des Systems erfolgen !**

**Der Modul ist durch folgende Handgriffe zu stecken:**

1. **Den Computer ausschalten.**

2. **Die Kappe des Modulschachtes ist durch leichten Druck mit Daumen und Zeigefinger auf die Griffflaechen abzunehmen.**

3. **Den Modul bis zum fuehlbaren Einrasten einschieben (hervorstehender Rand des Moduls liegt unmittelbar an der Geraetewand an).**

**Zum Entfernen des Moduls aus dem System sind folgende Schritte notwendig:**

1. **Den Computer ausschalten.**

2. **Den linken und rechten Zeigefinger unter den Modulkopf legen und mit dem Daumen die seitlich am Modul befindlichen Hebel gleichzeitig nach unten druecken. Dabei rastet der Modul aus und wird etwa einen Zentimeter aus dem Geraet herausgeschoben. Nun den Modul aus dem Schacht nehmen.**

3. **Die Kappe auf die Schachtoeffnung stecken.**

## Zuweisung

**Mit der Anweisung**

**SWITCH mm kk**

**wird der Modul auf den Steckplatz mm zugewiesen. kk ist das Steuerbyte.**

**Steckplatzadresse mm**

**Die Steckplatzadresse mm ist fuer jeden Steckplatz eines Geraetes in der dazugehoerigen Beschreibung angegeben. Im Grundgeraet besitzt der rechte Modulschacht die Adresse 08 und der linke die Adresse 0C.**

**Steuerbyte kk**

**Die erste Stelle des Steuerbytes kk legt die Anfangsadresse fuer den Modul fest und die zweite den Betriebszustand.**

**Betriebszustaende**

**Fuer die Module unterscheiden wir zwei Betriebszustaende:**
1. **INAKTIV = 0 (Diode leuchtet nicht. Der Modul ist vom Prozessor getrennt)**
2. **AKTIV = 1 (Diode leuchtet. Der Modul kann gelesen werden)**

**Adressierung**

**Der Speicherbereich des Softwaremoduls ist auf die Anfangsadresse C000H zu legen.**

**Da beim KC 85/3 auf dieser Adresse bereits der BASIC-Interpreter liegt, ist dieser vor der Zuweisung des Moduls INAKTIV zu schalten. Dazu wird die zweite Stelle des Steuerbytes 0 gesetzt. Der interne BASIC-Interpreter besitzt die "Steckplatzadresse" 02.**

**Somit lautet die Anweisung zum INAKTIV-Schalten des BASIC-Interpreters**

**SWITCH 02 C0**

**Steckt der Softwaremodul im Schacht 08, so wird dieser entsprechend mit der Anweisung**

**SWITCH 08 C1**

**zugewiesen.**

**Modulstrukturbyte**

**Jeder Modul besitzt zur Kennung ein zugeordnetes Strukturbyte. Der Software-Modul besitzt das Strukturbyte F7H (bei EPROM-Bestueckung). Ueber die SWITCH-Anweisung wird das Strukturbyte wie folgt angefordert:**

```
SWITCH 08       Eingabe Modulsteckplatzadresse

08 F7 C1        Antwort des Computers auf dem Bildschirm
-- -- --
 |  | ||
 |  | | ------- Betriebszustand (AKTIV) | zuletzt eingegebenes
 |  | |
 |  |  -------- Anfangsadresse (C000H)  | Steuerbyte kk
 |  |
 |   ---------- Strukturbyte
 |
  ------------- Modulsteckplatzadresse
```