Antworten zu den im Text gestellten Fragen

Auf den folgenden Seiten finden Sie jeweils die Antworten zu den im Text gestellten Frage. Dabei ist immer diejenige Textseite angegeben, auf der die gestellten Fragen stehen.

Wir verweisen hier noch einmal darauf, was bereits auf Seite B8 gesagt wurde: nicht gleich nachschauen, sondern zuerst einmal eine eigenen Antwort zurechtlegen und dann erst mit den hier stehenden Antworten vergleichen. Es sieht und hört ja keiner, wenn Sie völlig daneben lagen.

Zu Seite B8

1) Durch das mehrfache Kopieren des Textblocks wird der Umfang des Textes im Textspeicher so groß, daß der Speicherplatz nicht mehr ausreicht, das Betriebssystem meldet "Textspeicher voll" (FULL). Sie werden noch in diesem Lehrbrief erfahren, was in einem solchen Fall zu tun ist.

2) Der Zeilenüberlauf wird sichtbar gemacht, indem er mit RETURN an den Anfang der nächsten Zeile geholt wird. Die Taste Ret muß gedrückt werden, wenn der Cursor auf dem Pluszeichen (oder auf einem mit ESC M programmierten rechten Rand) steht.

3) Mit CTRL-I lassen sich tabellierte Texte wie Programmlistings ohne viel Mühe mit Leertasten-Abzählen usw. schreiben.

Zu Seite B14

1) Das Kommando PACK 60000 kann das Betriebssystem nicht ausführen, weil es gar nicht so viel Speicherplatz zu vergeben hat. Nach der Eingabe von PACK 100000 erscheint die Fehlermeldung 0, das bedeutet Überlauf (overflow). Das Betriebssystem kann nur 16-Bit-Zahlen verarbeiten!

Zu Seite C2

1. Der Ausdruck "mnemonisch" stammt von dem griechischen Wort für Gedächtnis ab. Im vorliegenden Fall sollen die mnemonischen Abkürzungen eine Gedächtnisstütze dafür sein, welcher Befehl gemeint ist.

2. Die Adresse einer Speicherzelle ist dann für den Programmablauf wichtig, wenn sie z.B. als Sprungadresse oder Startadresse eines Unterprogramms dient.

Zu Seite B15

1. Das Betriebssystem befindet sich im Speicher nicht an festen Adresse. Man weiß also nie, wo die Programme stehen, die man aufrufen möchte.

Zu Seite C6

1. Der Name eines Unterprogramms darf nicht unterteilt sein. Um nicht NEUEZEILE schreiben zu müssen, was gewiß recht unleserlich wäre, kann anstatt des von den Zahlen bekannten Zeichens $ auch der weniger störende Strich unten verwendet werden. Ein richtiger Abstand führt zur Fehlinterpretation bzw. Fehlermeldung.

2. Wir haben als möglichen Fehler angenommen, daß anstelle des ersten LD-Befehls der versehentlich der 8085-Befehl MOVE eingesetzt wurde. Der Assembler stoppt im zweiten Lauf an der Fehlerstelle und schreibt die Zeile mit dem Fehler auf den Bildschirm. Im vorliegenden Fall steht vor der Zeile das Zeichen L, was auf den falschen Opcode hinweist. Ferner gibt er aus: 1 fatal error.

Zu Seite B18

1. Programme mit den normalen Systemfunktionen sind auch auf anderen CP/M-Computern lauffähig, da die gleichen Konventionen wie bei diesem Betriebssystem beachtet werden. Das gilt nicht für die erweiterten Systemfunktionen ab Code 240, die es nur in unserem Betriebssystem gibt.

Zu Seite C10

1 Auf Seite C7 steht, das jedes Pluszeichen für 256 assemblierte Bytes steht. Demnach braucht das eigentliche Programm 256 mal 4 = 1024 Bytes, also 1 KByte Speicherplatz.

2 Alle Eingaben, die vom Assembler in ASCII-Code umgewandelt werden sollen, müssen zwischen Anführungsstriche gesetzt werden.

Zu Seite D8

1 Beim Kommando L wird der Inhalt eines Speicherbereiches in sedezimaler Schreibweise und disassembliert ausgegeben. Adressen und Werte, die als Symbol definiert sind, werden zusätzlich als Symbol ausgegeben. Beim Kommando G hingegen wird das Programm an der angegebenen Stelle gestartet und bei der ebenfalls angegebenen Unterbrechungsstelle (oder am Programmende) erscheint der gleiche Ausdruck wie nach L.

Zu Seite C16

1 An welcher Stelle im Unterprogramm ZEICHEN die beiden Befehle PUSH HL und POP HL stehen, ist dann egal, wenn sie vor und nach dem Befehl stehen, über dessen Einwirkung hinaus der Inhalt des HL-Registers gerettet werden muß. Sie können also auch jeweils an Anfang und Ende des Unterprogramms stehen.

2 Der JR Z-Befehl veranlaßt einen Sprung zum angegebenen Label, wenn das Bedingungsflag Z (zero=null) den Wert 1 hat, d.h. gesetzt ist. Dies geschieht durch den CPIR-Befehl, wenn die Speicherinhalte von A und der Adresse, auf die HL zeigt, gleich sind.

Zu Seite B20

1 Die Antwort zu der Frage nach der Einsprungadresse finden Sie nicht nur als praktische Beispiele in den bisher behandelten Programmen, sondern auch wörtlich auf Seite B15 im letzten Absatz: es ist die Adresse 0005H.

Zu Seite C17

1. Die Assemblierung des Programms wird durch einen Rücksprung ins Betriebssystem abgeschlossen, so daß Sie den Editor aufrufen können. Markieren Sie das gesamte, dort noch stehende Programmlisting als Block und löschen diesen mit ESC-Z. Jetzt ist der Textspeicher leer, ein neuer Text kann eingegeben werden. Eine andere Möglichkeit bietet die Funktion ESC PZ, mit der ab der Cursorposition der gesamte nachfolgende Text gelöscht wird. Aus dem Bild A 2.1 geht hervor, daß die Benutzerpro- programme in der TPA abgelegt werden. Beim Löschen des Textspeichers geschieht dem assemblierten Programm al-so nichts, es kann mir RUN aufgerufen werden.

Zu Seite C20

1 Das neue Hauptprogramm, das für alle Buchstaben des Alphabets die Statistik automatisch durchführt, ist auf Seite F16 aufgelistet.Es holt sich das erste Zeichen, also das A, legt es aber vor der Auswertung mit dem Unterprogramm STATISTIK im Stack ab. Von dort wird es anschliessend wieder geholt und mit dem letzten Zeichen verglichen. Wenn beide übereinstimmen, wird das Programm abgebrochen. Das letzte Zeichen ist nicht z, sondern das zuletzt in der ASCII-Tabelle stehende ß. Wenn dieses Zeichen auf Ihrer Tastatur nicht vorhanden ist, dann müssen Sie dafür seinen ASCII-Code (7E) eingeben, dann aber ohne die Anführungsstriche.

Das jeweilige Hereinholen des nächsten Zeichens durch Inkrementieren des vorangegangenen funktioniert nur deshalb, weil die Zeichen im ASCII-Code, wie schon erwähnt, einfach durchnummeriert sind. Ein kleiner Schönheitsfehler tritt dabei auf: die drei in der Reihenfolge zwischen den Groß- und Kleinbuchstaben stehenden Zeichen werden auch mit erfaßt.

Antworten zu den im Text gestellten Fragen

Zu Seite C24

1. In der Reihenfolge der Wichtigkeit sind folgende Anforderungen an ein Programm für das Life-Game zu stellen:
   a) Eingabe einer Spielfeldsituation mit Tastaturund Bildschirm.
   b) Automatisches Errechnen neuer Generationen und Ausgabe auf dem Bildschirm.
   c) Evtl. verändern einer vorhandenen Spielsituation.
   d) Evtl. ausdrucken einer Spielsituation.
   e) Evtl. abspeichern von Spielsituationen.

Zu Seite A6

1. Wie auf Seite A5 dargelegt, werden Ein/Ausgabe-Operationen und Schreib/Lese-Operationen niemals gleichzeitig vorkommen, deswegen werden die vorhandenen Busleitungen für beide Operationsarten benutzt.

Zu Seite C22

1. Bei allen IN/OUT-Befehlen müssen besondere Steuersignale an den Adreßdecodierer geschickt werden, damit nicht der Arbeitsspeicher, sondern die Ports angesprochen werden (siehe auch Seite A6).

Zu Seite A10

1. Eine Druckerschnittstelle, die nur mit dem Strobe-Signal arbeitet, würde den Drucker von der Geschwindigkeit her "überfahren". Ohne irgendeine Rückmeldung (Busy-Signal) gingen Zeichen verloren.

Zu Seite B22

1. Das Programm zur Demonstration der Konsolstatus-Abfrage sieht bei uns so aus:

```
                TITLE PROGRAMMBEISPIEL SYSTEMFUNKTION 11

                ORG  100H          ;das Programm beginnt bei
                                   ;0100H

SYSTEM          EQU  0005H         ;Adresse für Aufruf von
                                   ;Systemfunktionen
WSTARTF         EQU  0             ;Warmstart
CONEINF         EQU  1             ;Konsoleingabe
CONAUSF         EQU  2             ;Konsolausgabe
STATUS          EQU  11            ;Konsolstatus-Abfrage

CTRL            EQU  00011111B     ;CTRL-C

START:          LD   C,STATUS      ;Funktionscode für Konsol -
                CALL SYSTEM       ;Status ins C-Register

                OR   A             ;Zerobit entspr. Inhalt A
                JR   NZ,SCHIRM     ;wenn Status "Zeichen be-
                                   ;reit", nach SCHIRM

                LD   C,CONAUSF     ;Funktionscode für Konsol-
                                   ;Ausgabe ins C-Register
                LD   E,'.'         ;Auszugebendes Zeichen ins
                CALL SYSTEM        ;E-Register

                JR   START         ;Programmzähler auf Pro-
                                   ;gramm-Anfang

SCHIRM:         LD   C,CONEINF     ;Funktionscode für Konsol-
                CALL SYSTEM        ;eingabe ins C-Register

                CP   'C' AND CTRL  ;ist das eingebene Zei-
                                   ;chen CTRL-C?
                JP   Z,WSTARTF     ;wenn ja, Warmstart

                JR   START         ;Programmzähler auf Pro-
                                   ;gramm-Anfang

ENDE:           END
```

Nachdem Sie das Programm gestartet haben, sehen Sie, wie mit grosser Geschwindigkeit der Bildschirm voller Punkte geschrieben wird. Geben Sie irgendwelche andere Zeichen ein: auch wenn die Tasten ganz schnell nacheinander betätigt werden, ist es schwer, die "Hintergrundarbeit" des Computers (das Punkte-Ausgeben) ganz zu unterbrechen. Damit demonstriert unser Prograemmchen, was beim Gebrauch der Systemfunktionen 1 und 2 durch das Warten auf die Eingabe an Zeit sozusagen vergeudet wird.

Zu Seite B25

1. Mit dem nachstehenden Programm können Sie die Escape-Sequenzen für das Sichtgerät ausprobieren.

```
        ORG     100H                ;das Programm be-
                                    ;ginnt bei 0100H

SYSTEM  EQU     0005H               ;Adresse für Aufruf
                                    ;von Systemfunktion
WSTART  EQU     0000H               ;Warmstart
DIRCONF EQU     6                   ;Direkt-Konsolfunkt.

CONTROL EQU     1FH                 ;CTRL-C

SCHLEIFE:
        LD      E,0FFH              ;für Eingabe
        LD      C,DIRCONF           ;Funktionscode ins
        CALL    SYSTEM              ;C-Register

        OR      A                   ;setzt Null-Flag, falls A=0
        JR      Z,SCHLEIFE          ;falls keine Eingabe
        CP      CONTROL AND 'C'
        JP      Z,WSTART
        LD      E,A                 ;Ausgabe
        LD      C,DIRCONF
        CALL    SYSTEM
        JR      SCHLEIFE

ENDE:   END
```

Zu Seite C29

1. Bei der Verwendung anderer Systemfunktionen wäre eine Drucker-Protokollierung möglich. Diese würde aber zu keinem vernünftigen Ergebnis führen, da der Drucker nicht unbedingt 80 Zeichen pro Zeile hat und die Escape-Sequenzen für das Sichtgerät nicht verarbeiten kann.

Zu Seite C47

1. Das Unterprogramm HOLE_FELD holt sich vom Anfang des Textspeichers beginnend ein Zeichen nach dem anderen und prüft, ob es ein Zwischenraum oder ein Zeichen ist. Nach dieser Prüfung werden die Elemente im internen Feld belegt oder nicht belegt. Im Textspeicher

steht zu Beginn des Programms , also nach RUN, noch das über den Editor eingegebene Programm-Listing. Das gibt die Grundlage für das interne Feld. Vergleichen Sie, wie aus dem Programm-Listing die erste Konfiguration hervorgegangen ist: alle Zeichen (auch das CTRL-I für Tabulator) wurden zu Sternen, alle Zwischenräume bleiben Zwischenräume.

Zu Seite C48

1. Stellen Sie den Cursor an den Anfang des Bildschirms (im Editor) und löschen Sie mit CTRL-PZ den Textspeicher (vorausgesetzt, daß das Programm "Life-Game" assembliert ist und im Arbeitsspeicher steht

Zu Seite E6

1. Wenn der Datenemfänger immer zum Empfang bereit ist, das heißt auch, wenn er in der Lage ist, die Daten ebenso schnell wie der Sender sie ausgibt, zu "verdauen", dann ist ein Betrieb ohne Handshake-Signale oder Protokolle über eine serielle Schnittstelle möglich.

Zu Seite C50

Am Anfang des Druckprogramms ändern:

```
DRUCKE_FELD:
        LD      C,TXTBEGF
        CALL    SYSTEM
        PUSH    HL              ;Textspeicher von HL
        POP     IX              ;nach IX
        LD      HL,(VON_FELD)
        LD      B,ZEILEN
        JR      DRUCKE_NORMAL   ;KEIN CR/LF am Anfang
DRUCKE_CRLF:
```

Am Ende des Druckprogramms statt CALL CR_LF:

```
        LD      E,EOF           ;Textende markieren
        CALL    DRUCKE
```

Bei DRUCKE ändern:

```
DRUCKE:
        LD      (IX),E
        INC     IX
        RET
```

Antworten zu den im Text gestellten Fragen

Zu Seite C 56

1. Der Inhalt des Zählers für ausgelassene Zeichen (Null) wurde mit irgendwelchen Zeichen überschrieben. Das bewirkte, daß beim nächsten Zeichen eventuell der Cursor wieder neu gesetzt wurde, obwohl das gar nicht nötig gewesen wäre. Natürlich kann die dadurch entstandene um einen winzigen Zeitabschnitt längere Programmdauer nicht bemerkt werden.

Zu Seite C 64

1. Der Prozessor arbeitet intern mit binär dargestellten Zahlen. Jeweils vier binär dargestellte Zahlen lassen sich zu einer sedezimal dargestellten Zahl zusammenfassen, wodurch eine bessere Lesbarkeit erreicht wird.

2. Bei der Festkomma-Rechnung wird vorgegangen, als ob ganze Zahlen vorlägen. Das Komma wird lediglich bei den Ein- und Ausgabeoperationen berücksichtigt.
   Bei der Fließkomma-Rechnung werden die Zahlen normalisiert und die Kommastelle in einem getrennten Register mitgeführt.

3. Durch Invertierung aller Bits der binär dargestellten Zahl und anschließender Inkrementierung wird intern das echte Komplement dargestellt.

Zu Seite D 16

1. Von den Grundlagen her wissen Sie sicher noch, daß auch das Dualsystem ein Stellenwert-System ist, also jede Stelle der dargestellten Zahl einen bestimmten Wert hat. Im Dualsystem unterscheiden sich die Werte zweier Stellen (z.B. zweite Stelle zwei hoch zwei und vierte Stelle zwei hoch drei) um den Wert zwei hoch zwei. Eine Verschiebung um zwei Stellen nach links

entspricht also der Erhöhung der Wertigkeit um zwei hoch zwei. Das nachfolgende Beispiel zeigt, daß dies einer Multiplikation mit vier entspricht:

dual:	0	0	0	1	0	1	0	1	
Stellenwert:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
dezimal:	0+	0+	0+	16+	0+	4+	0+	1	= 21

um zwei Stellen nach links geschoben:

	0	1	0	1	0	1	0	0	
	0+	64+	0+	16+	0+	4+	0+	0	= 84 = 21*4

G

10

Zu Seite B 30

1. Die Eingabe wird abgebrochen, wenn die mit mc angegebene maximale Zeichenanzahl erreicht ist. Probieren Sie auch das aus!

2. Da für die Angabe von mc eine 8-Bit-Speicherstelle zur Verfügung steht, können Datenbereiche bis FFH Länge, also bis 255 Zeichen angegeben werden.

Zu Seite C 68

1. Die gleiche Zuweisung läßt sich mit einer Labeldefinition erreichen:

```
        .
        .
       Befehl..
SYMBOL: Befehl..
       Befehl..
        .
        .
```

Zu Seite C 73

1. Ihr System ist ein Entwicklungssystem, auf dem unter Umständen auch Programme entwickelt werden, die auf kleinen, preiswerten Einplatinen-Computern Steuer- und

Regelaufgaben erfüllen sollen. Bei solchen Systemen sind von der Hardware her der Datenbereich (im RAM) und der Programmbereich (im EPROM) getrennt.

2. Die Überprüfung mit dem Tester geht ganz einfach. Schreiben Sie zuerst im Editor die Zuweisungen und gleich noch ein IF-ENDIF-Beispiel:

```
ORG   EQU   100H
FALSE EQU   0
TRUE  EQU   NOT FALSE

CHECK EQU   FALSE

      IF    CHECK
      LD    A,55
      ENDIF
```

Das wird assembliert, danach der Tester aufgerufen und gefragt:

```
*? FALSE
0000
*? TRUE
FFFF
```

Man sieht, was der Assembler gemacht hat. Mal sehen, wie die IF-ENDIF-Geschichte funktioniert. Zuerst den Speicher mit Nullen füllen, damit alles übersichtlich ist:

```
*F 100,200,00
```

Danach nocheinmal assemblieren, also CTRL-C, ASM, und dann in den Tester:

```
*L 100
```

Der Ausdruck enthält lauter Nullen, nichts wurde assembliert, was ja auch richtig ist, denn: CHECK EQU FALSE!

Ändern Sie das im Editor um: CHECK EQU TRUE. Danach die gleiche Prozedur noch einmal und beim List erscheint der nun assemblierte Befehl LD A,55:

```
3E37    LD A,37
```

Wir können nur wiederholen, was schon mehrmals gesagt wurde: probieren Sie, prüfen Sie nach, was im Text behauptet wird, spielen Sie mit den Möglichkeiten des Betriebsprogramms wie im vorliegenden Beispiel, das bringt am meisten!

Zu Seite C 74

1. Im Unterprogramm EINGABE wird die Systemfunktion zum Einlesen eines Strings aufgerufen. Wird in dieser Funktion vom Benutzer als erstes Zeichen ein CTRL-C eingegeben, so erfolgt ein Warmstart (siehe auch Seite B 30).

Zu Seite C 79

1. Gehen Sie, nachdem das Programm assembliert ist, in den Tester. Mit

   G 100,HOLE_ZAHL

   wird bestimmt, wie weit das Programm erst einmal laufen soll (siehe auch Seite D 14). Das danach erscheinende Fragezeichen fordert eine Eingabe an, z.B. wie im Text die Zahl 23. Danach können Sie mit

   D Puffer

   wie auf Seite B 29 gezeigt, die Ablage der Eingabe im Speicher anschauen. Was im weiteren Ablauf des Unterprogramms HOLE_ZAHL in den Schleifen geschieht, läßt sich mit dem Kommando T Schritt für Schritt verfolgen, wobei natürlich auch mit z.B. T 10 das Ergebnis von in diesem Fall zehn Schritten auf einmal angeschaut werden kann.

Zu Seite C 81

1. Der Zeiger HL muß nach dem Aufruf des Unterprogramms CLEAR um 1 dekrementiert und die Speicherstelle, auf die er dann zeigt, inkrementiert werden (siehe Seite F41).

Antworten zu den im Text gestellten Fragen

Zu Seite E 16

1. Bei den mit S und R aufgerufenen Betriebsarten des Modem-Programms wird aus dem Textspeicher gesendet bzw. in den Textspeicher empfangen. Im Textspeicher sind alle Daten in Form von ASCII-Zeichen abgelegt, weil der Textspeicher ja im Editor mit den Ein- und Ausgabe-Geräten Konsole, Drucker usw. korrespondiert.

   Ein assembliertes Programm hingegen steht in sedezimaler (eigentlich binärer) Darstellung im Arbeitsspeicher. Auch wenn man es mit einem Trick in den Textspeicher brächte, hätten die ASCII-Darstellungen des Programms keinen Sinn. Es könnte z.B. das Byte 1AH = Textende mitten im Programm vorkommen.

   Im nachfolgenden Fachgebiet TESTER erfahren Sie, wie so ein Programm direkt aus dem Arbeitsspeicher und mit Protokoll übertragen wird.

Zu Seite D 20

1. Beide angesprochenen Kommando-Paare speichern Daten auf Cassette (bzw. Diskette) oder lesen sie von dort ein. Bei den Editor-Kommandos geht es aber um Daten aus dem Textspeicher, die in ASCII codiert sind.

   Bei den Tester-Kommandos hingegen geht es um Daten aus dem Arbeitsspeicher, die sedezimal codiert sind.

Zu Seite C 95

1. Nehmen Sie als Beispiel die Subtraktion zweier Zahlen,zuerst in der gewohnten dezimalen Darstellung:

```
 52 : 3
-30                (3 einmal nach links geschoben)
 --
 22
 -3
 -3
 -3
 -3
 -3
 -3
 -3
 --
  1   (Rest)
```

Dazu ist nichts zu sagen, es ist die gleiche Rechnung wie auf Seite C 93, nur mit anderen Zahlen. Die Anzahl der Subtraktionschritte (7 mal mit der 3) entspricht dem Ergebnis der Division. Nun zur Rechnung in binärer Darstellung, mit den gleichen Zahlen:

```
  110100  (52D)
-1100000  (3D)  11B um 5 Stellen geschoben
--------
-.......        ergibt Minus, also              0 <-0

  110100
 -110000        11B um 4 Stellen geschoben
 -------
     100        ergibt plus, also              01 <-1

     100
  -11000        11B um 3 Stellen geschoben
  ------
  -.....        ergibt minus, also            010 <-0

     100
   -1100        11B um 2 Stellen geschoben
   -----
   -....        ergibt minus, also           0100 <-0

     100
    -110        11B um 1 Stelle geschoben
    ----
    -...        ergibt minus, also          01000 <-0

     100
     -11        11B
     ---
       1        ergibt plus, also          010001 <-1
```

Man sieht, es wird nicht die Anzahl der Subtraktionsschritte als Ergebnis genommen, sondern das 0 oder 1-Ergebnis jeder Subtraktion nach links in ein Ergebnis-

G

14

Register hineingeschoben. Zum Schluß steht dort 010001B = 17D. Der Rest steht, wie bei der Rechnung mit Dezimalzahlen, als Ergebnis der letzten Substraktion.

Beachten Sie, daß das Ergebnis einer Subtraktion, solange es positiv ist, im Hauptspeicher für die nächste Subtraktion stehenbleibt. Wenn es negativ ist, bleibt es im Zwischenspeicher, der Betrag ist uninteressant (weshalb auch nur Pünktchen dastehen).

Übrigens, das Ergebnis-Bit (ob plus oder Minus), was da in das Ergebnis-Register hineingeschoben wird , ist das Carry-Flag. Es muß allerdings invertiert werden, da es ja bei negativem Subtraktionsergebnis auf 1 gesetzt wird.

Zu Seite C 98

1. Auf den ersten Blick fällt die Antwort einfach aus: es muß nämlich das Eingabe-Unterprogramm des Fakultät-Programms zur Eingabe benutzt werden. Im Einzelnen ergeben sich jedoch eine ganze Reihe von Ergänzungen und Änderungen, die wir hier ohne größere Erklärungen angeben. Versuchen Sie einmal, sie mit Ihren eigenen Ideen zu vergleichen und zu durchschauen.

   Zuerst muß bei den Systemfunktionen auf Seite F 52 eingefügt werden

```
STREINF  EQU 10          ;Stringeingabefunktion
```

   bei den sonstigen Konstanten auf Seite F 53:

```
MAX_LAENGE EQU 10        ;Länge des Eingabepuffers
```

   Ebenfalls auf Seite F 53, nach dem Label START, vor dem LABEL SCHLEIFE0:

```
CALL    EINGABE
CALL    HOLE_ZAHL
JR      C,START
LD      A,H
LD      (PRIM+BIN_BYTE-2),A
LD      A,L
LD      (PRIM+BIN_BYTE-1),A
```

   Nach dem Hauptprogramm auf Seite F 54 müssen die beiden Unterprogramme EINGABE (von Seite F 40) und HO-

LE_ZAHL (bis DJNZ HOL_SCHLEIFE auf Seite F 41) eingesetzt werden:

```
EINGABE_TEXT:
        DB      CR,LF,'? $'

EINGABE:
        LD      DE,EINGABE_TEXT ;? in neuer Zeile ausgeben
        LD      C,STRAUSF
        CALL    SYSTEM

        LD      DE,PUFFER       ;und Eingabezeile einlesen
        LD      C,STREINF
        CALL    SYSTEM
        RET


HOLE_ZAHL:
        LD      HL,PUFFER+1     ;HL auf Länge d. Eingabe
        LD      B,(HL)          ;dieselbe nach B
        INC     HL              ;HL auf Eingabe
        EX      DE,HL           ;DE auf Eingabe und
        LD      HL,0            ;HL := 0

HOL_SCHLEIFE:                   ;wiederhole:
        LD      A,(DE)          ;  Zeichen von Eingabe holen
        INC     DE              ;  und Zeiger weiter

        SUB     '0'             ;  ASCII -> binär wandeln
        RET     C               ;  Fehler falls kleiner 0
        CP      10              ;  oder größer 9
        CCF
        RET     C               ;  (Rücksprung mit Carry bei
                                ;  Fehler)
        PUSH    BC              ;  Schleifenzähler und
        PUSH    DE              ;  Zeiger retten,
        EX      DE,HL           ;  DE := HL
        LD      B,10
        LD      HL,0            ;  HL := 0

HOL_SCHLEIFE1:                  ;  wiederhole 10 mal:
        ADD     HL,DE           ;    HL := HL+DE
        DJNZ    HOL_SCHLEIFE1   ;    (d.h. HL := HL*10)
        LD      E,A
        LD      D,0
        ADD     HL,DE           ;  HL := HL+Ziffer
        POP     DE              ;  Zeiger und
        POP     BC              ;  Schleifenzähler holen

        DJNZ    HOL_SCHLEIFE    ;  Zähler (B) dekrementieren
                                ;bis Zähler=0
        RET
```

Schließlich muß im Datenbereich auf Seite F 60 noch der Puffer für die Zeichen eingefügt werden.

```
;Datenbereich


DEZ_REG:
        DS      DEZ_STELL/2     ;Speicher für Dezimalzahl
BIN_REG:
        DS      BIN_BYTE
PRIM:
        DS      BIN_BYTE        ;Speicher für Binärzahlen

PUFFER:                         ;Puffer f. Eingabezeile
        DB      MAX_LAENGE      ;Wert der maximalen Länge
        DS      1               ;Platz f. tatsächliche Länge
        DS      MAX_LAENGE      ;Platz für Eingabe

TEILER:
        DS      BIN_BYTE
QUOTIENT:
        DS      BIN_BYTE
REST:
        DS      BIN_BYTE

        END     START
```

Zu Seite C 110

1. Beim Wechsel zwischen den beiden Programmen ist wichtig, daß jeweils die Registerinhalte und Programmzählerstände abgespeichert werden. Dies geschieht duch die Unterprogramme TRANSFER_PRIM und TRANSFER_SYS:

2. Das im Hintergrund laufende Primzahlen-Programm könnte durch die Textspeicher-Erweiterung mit PACK überschrieben werden.

3. Der Hinweis auf die Seite D 17, wo die Tester-Kommandos für das Senden und Empfangen von beliebigen Speicherbereichen vorgestellt werden, führt auf die Antwort: im Tester gibt es auch Kommandos, mit denen Dateien abgespeichert und wieder abgerufen werden können Ein Blick auf Seite H 10 zeigt, welche das sind:

SF /ausdruck1/,/ausdruck2/

Mit diesem Kommando wird der Speicherbereich von /ausdruck1/ bis /ausdruck2/ auf Cassette (oder Diskette) abgespeichert. Es erscheint auf dem Bildschirm zuerst die Frage "Filename?". Nachdem Sie einen Namen eingegeben haben, wird der File (die Datei) auf dem externen Speicher angelegt.

Mit dem Kommando

RF /ausdruck/

wird eine Datei ab der Adresse /ausdruck/ oder (wenn nicht angegeben) ab der Adresse 0100H in den Speicher geladen. Auch hier fragt das System zunächst nach dem Filenamen.

Sie können also nach dem Assemblieren eines Programms in den Tester gehen, das Programm für spätere Zeiten mit SF abspeichern und danach, z.B. mit RUN nach Verlassen des Testers, das Programm starten. Wenn es dann irgendwann wieder einmal benötigt wird, läßt es sich fertig assembliert mit RF von der Cassette wieder in den Speicher einlesen.

Antworten zu den im Text gestellten Fragen

Zu Seite B 37

1) Das Betriebssystem CP/M ist, wie erwähnt, das Standard-Betriebssystem der Computer mit den Prozessoren der 80-er Reihe. Durch diese weite Verbreitung gibt es ein sehr umfangreiches Angebot an Programmen und Programmpaketen für alle möglichen Anwendungsgebiete, von dem Sie jetzt auch profitieren können.

Zu Seite B 41

1) PROGRAMM.2 und TESTER sind gültige Dateinamen. LEHRBRIEF.TXT hat einen über 8 Zeichen hinausgehenden Dateinamen und in FRAGE?.BRF ist ein nicht erlaubtes Zeichen (?) enthalten. Daß ein Fragezeichen unter Umständen doch gültig ist, erfahren Sie später noch.

Zu Seite B 48

1) Die Übertragung einer Datei von Diskette auf Cassette ist noch einfacher als umgekehrt. Es müssen aber auch hier wieder die unterschiedlichen Vorschriften für den Dateinamen beachtet werden. Die Prozedur läuft wie folgt ab:

   Rufen Sie zuerst die Diskettenversion von ZEAT auf. Gehen Sie in den Tester und lesen mit RF die Datei in den Speicher. Als Quittung wird die nächste freie Speicherstelle nach der eingelesenen Datei angegeben, beispielsweise:

```
*RF
FILENAME? LIFEGAME.ASM
next:06C9
```

   Nach einem RESET wird das residente ZEAT aufgerufen, also mit "3" aus dem Anfangsmenü. Dann speichern Sie mit SF die Datei auf die Cassette im jetzt angeschlos-

senem Recorder. Dabei muß ein Dateiname angegeben werden, wie Sie ihn vom Cassettenbetrieb her noch kenwerden. Außerdem verlangt SF die Angabe des Speicherraums. Beispiel:

```
*SF 0100,06C9
FILENAME? 1001
```

Schnell noch eine Kontrolle, ob's auch wirklich geklappt hat: steigen Sie mit CTRL-C aus dem Tester aus, mit PACK 20000 wird der Textspeicher vergrößert und danach im Editor mit ESC-A 1001 die Datei von der Cassette geholt.

Zu Seite C 113

1) Das Programm wird geladen und assembliert. Danach wird das zu wandelnde Programm geladen, nicht assembliert und das im Arbeitsspeicher stehende assemblierte Programm mit RUN gestartet. Anschließend wird das gewandelte Programm aus dem Textspeicher wieder auf die Diskette zurückgeschrieben.

Zu Seite B 60

1) Zuerst muß bei den Definitionen der Systemfunktionen eingefügt werden:

```
MAKEFCB   EQU      250
```

weiterhin die Definition des Strings mit der Frage nach dem Dateinamen und des Ausgabe-FCB:

```
AUSG_FRAGE:   DB   'Name der Ausgabe-Datei?$'
AUSGABE_FCB:  DS   36,0
```

und dann:

```
LD      DE,AUSG_FRAGE      ;nach Dateiname fragen
                           ;(Eingabe: Dat.Dat)
LD      C,STRAUSF
CALL    SYSTEM
LD      DE,PUFFER          ;Name in Puffer lesen
LD      C,STREINF
CALL    SYSTEM
```

```
LD      DE,PUFFER+2        ;FCB erzeugen
LD      IY,PUFFER
LD      IX,AUSGABE_FCB
LD      C,MAKEFCB
CALL    SYSTEM
LD      DE,AUSGABE_FCB
LC      C,OPENF
CALL    SYSTEM
.
.
```

2) Der eingegebene Dateiname wird in Großbuchstaben gewandelt.

Zu Seite C 118

1) Die beiden Aufrufe der Systemfunktion "MAKEFCB" werden ersetzt durch einen Unterprogrammaufruf, also statt

```
LD      C,MAKEFCB
CALL    SYSTEM
```

an den beiden Stellen

```
CALL    MAKE_FCB
```

Das Unterprogramm MAKE_FCB kann wie folgt aussehen, wobei bei unserem Vorschlag die Bearbeitung einer vorgestellten Laufwerksangabe (B:....) nicht vorgesehen ist - es wird immer das Standardlaufwerk angesprochen:

```
MAKE_FCB:
   LD      A,(IY+1)          ;Pufferlänge laden
   AND     A
   RET     Z                 ;Länge 0: Fehler
   LD      C,A               ;sonst Länge nach Register C
                             ;zum runterzählen

   LD      (IX),0            ;Standard-Laufwerk (Lauf-
                             ;werksangabe wird von diesem
                             ;Progr. nicht ausgewertet!)
   INC     IX
   INC     C
   LD      B,11              ;= Länge Dateiname im FCB

MAKEFCB_SCHL:                ;wiederhole:
   DEC     C                 ;  Puffer schon leer?
   JR      Z,MAKEFCB_FUELL   ;   ja: Rest FCB auffüllen
   LD      A,(DE)            ;  Zeichen holen
```

G

21

```
   CP      '.'                ;  Punkt ?
   JR      Z,MAKEFCB_TYP      ;   ja: Typ fängt an
   CP      'a'                ;  Kleinbuchstabe?
   JR      C,MAKEFCB_GROSS    ;   nein: ok
   SUB     20H                ;   ja: wandeln
MAKEFCB_GROSS:
   CP      ' '+1              ;  Steuerzeichen?
   JR      C,MAKEFCB_FUELL    ;   ja: Rest auffüllen
   CP      '='                ;  Gleichheitszeichen?
   JR      Z,MAKEFCB_FUELL    ;   ja: auch füllen
   INC1    DE                 ;  Puffer-Zeiger weiter,
   LD      (IX),A             ;  Zeichen in FCB,
   INC     IX                 ;  FCB-Zeiger weiter,
   DJNZ    MAKEFCB_SCHL       ;  Zähler dekrementieren
                              ;bis Zähler Null
   SCF                        ;kein Fehler: Carry=1
   RET                        ; ... fertig

MAKEFCB_FUELL:
   LD      A,B                ;Zähler schon Null?
   OR      A
   SCF                        ;(kein Fehler: Carry=1)
   RET     Z                  ; ja: fertig
   LD      (IX),' '           ; nein: ZWR in FCB
   INC     IX                 ; Zeiger inkrementieren
   DEC     B                  ; Zähler dekrementieren
   JR      MAKEFCB_FUELL

MAKEFCB_TYP:
   INC     DE                 ;Puffer-Zeiger weiter
MAKEFCB_TYPSCHL:
   LD      A,B
   CP      4                  ;Zähler kleiner 4?
   JR      C,MAKEFCB_SCHL     ; ja: Typ kann eingetragen
                              ;     werden
   LD      (IX),' '           ; nein: auffüllen bis
   INC     IX                 ;       zu Dateityp!
   DEC     B
   JR      MAKEFCB_TYPSCHL
```

Zu Seite C 122

1) Auf Seite B 41 sind diese Spezifikationen aufgeführt. Danach besteht die Erweiterung des Dateinamens, mit der der Dateityp angegeben werden kann, aus drei Buchstaben oder Ziffern. Auf Seite B 47 sind die Zeichen aufgelistet, die auf keinen Fall verwendet werden dürfen: < > . , ; : = ? * sowie eckige Klammern.