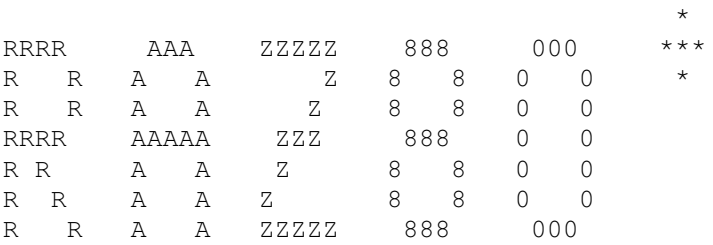

Version 3.3

von

MicroByte & BitResearch

(Software Consulting, Distribution and Production)

Das Original - Handbuch geschrieben von

Hagen D. Frey

Hacker - Software from Thuringia,
produced in G. D. R.

Inhalt:

Vorwort

Sicher ist bei jedem, der sich ernsthaft mit Computer und Software beschäftigt, nach einiger Zeit der Arbeit mit entsprechenden Programmen der Wunsch entstanden, ins Innere dieser Einblick zu nehmen. Die Motivation dazu mag unterschiedlich sein, teils aus Interesse (Neugier, zur Weiterbildung), wie gewisse Aufgabenstellungen realisiert wurden, teils aus dem Wunsch heraus, eigene Ergänzungen oder Änderungen am Programm vorzunehmen, um das Programm auf anderen Systemen zu implementieren oder im schlimmsten Falle, um Fehler im Programm zu beseitigen. Dem Anliegen kommt die Standardisierung der Hard- und Software auf der 8-bit Szene seit Anfang der 80-er Jahre entgegen (Z80 durchgesetzt, Siegeszug von CP/M). Gegen unser Anliegen steht aber, daß normalerweise kein Software-Hersteller seine Quellprogramme, die wir ja lesen, verstehen und modifizieren könnten, verkauft. Als Softwarekunde erhält man im Normalfall fertig übersetzte Programme, mit denen unsere Maschine etwas anfangen kann, wir aber kaum. Aus diesem Grund entstanden Dis- oder Reassembler, Programme also, die aus Maschinenprogrammen etwas lesbares erzeugen.

Die Ansätze von RAZ80 reichen bis 1983 zurück, als ich für ein CP/M-System einen Reassembler brauchte und der alte MEOS-Reassembler den gestiegenen Anforderungen nicht mehr genügte. Ich fand damals in RDK's "Hard- und Software Praxis" einen universellen Disassembler in BASIC, an dem ich etwas herumbastelte. Das Prinzip des Zwei-Pass-Reassemblers wurde von dort übernommen. In einer alten Elcomp fand "Schinki" B. Schinkoethe einen 1K-Disassembler, der recht gut als Zeilendisassembler funktionierte. Leider hatten wir nur eine schlechte Fotokopie, so daß wir weder Heftnummer noch Autor kennen. Er mag uns das verzeihen. Die nächste Version wurde in Pascal MT+ geschrieben und enthielt den 1K-Disassembler als Funktion. Dieser Disassembler konnte schon sinnvoll eingesetzt werden, z.B. für CCP und BDOS. Er war aber immer noch recht hausbacken und hatte an vielen Stellen Ecken und Kanten, vor allem war das Programm zu lang und zu langsam.

Mit der Zeit wurden Anforderungen für einen neuen Reassembler gesammelt. Das er in Assembler geschrieben werden mußte, war klar. Nur in Assembler können kurze Programme geschrieben werden, die leistungsfähig und schnell sind. Leider ist der Programmier- und Wartungsaufwand viel höher als bei vergleichbaren Programmen in einer Hochsprache. Meiner Ansicht nach ist die Assemblerprogrammierung die Krönung der Programmierkunst.

In meinem Urlaub '85 stellte ich das Konzept zusammen und so konnte im September in reichlich einer Woche RAZ80 in seiner jetzigen Form als Version 3.0 geboren werden. Große Verdienste erwarb sich dabei Jonny Jonas, der unermütlich RAZ80 erprobte und immer wieder Anregungen und Kritiken gab. Ihm gilt mein besonderer Dank, aber auch allen Bekannten der Erfurter PC-Szene, die mit ihren Hinweisen RAZ80 zu dem machten, was es jetzt ist. Bei der "Vermarktung" ist mir anfänglich ein Fehler unterlaufen, da ich die letzte Korrektur auf Jonny's Maschine gemacht habe und den Fehler bei mir nur in der Quelle geändert habe. (IX wurde nicht gepush't, so daß das Programm abstürzte, wenn BIOS IX veränderte.) Da ich die Quelle mitvertrieb, brauchte eigentlich jeder Nutzer nur sein RAZ80 nochmal zu übersetzen und alles war Ok.

Anfang '86 kamen noch einige kleine Änderungen dazu. Das Programm HEXDUMP wurde in RAZ80 integriert, um auch vernünftige Hexdumps an beliebigen Adressen zu erstellen. Diese Änderungen wurden mit Version 3.1 und 3.2 verbreitet. Vorliegende Version 3.3 entstand im Juli '86. Neu sind Start und Ende der Ausgabe, Binär- und Dezimalbyte-segmente und Kommentardateien. Die F- und die W-/Y-Optionen entfallen, nach allen Hilfsdateien wird gesucht. Sind sie vorhanden, so werden sie automatisch genutzt. Geplant ist für eine Version 3.4 die Einführung von Segmentmacros.

Erfurt, Juli 1986

1. Einleitung

Willkommen bei der Arbeit mit dem Z80-Reassembler RAZ80 von MicroByte&BitResearch. RAZ80 ist ein Mittelklasse-Reassembler, bei dem weniger Wert auf Intelligenz (dazu sind ja wir da), dafür aber auf Komfort und Geschwindigkeit gelegt wurde. Ein 10 KByte Maschinenprogramm (DDTZ) wird in ca. 90-120 sec reassembliert (Jonny prägte den Begriff "gerazt") und dabei werden 68 KByte Quellprogramm erzeugt.

Folgende Files gehören zum Softwarepaket von RAZ80:

RAZ80.COM	8K	-	das Reassemblerprogramm
RAZ80.MAC	60K	-	der MACRO80-Quellcode
RAZ80.DOK	32K	-	diese Beschreibung
TEST.INP	2K	-	Testprogramm
TEST.RDF	2K	-	Beschreibungsfile dazu
TEST.SYM	2K	-	Symbolfile dazu
TEST.CMT	2K	-	Kommentarfile dazu
COMP.COM	2K	-	Vergleich zweier Programme *)
COMP.MAC	8K	-	Quellprogramm dazu *)

Das Programm COMP wurde von J. Jonas geschrieben und für das RAZ80-Paket zur Verfügung gestellt. RAZ80 wurde für Z80-Programme unter CP/M 2 geschrieben, die mit dem Macroassembler MACRO80 von MicroSoft weiterverarbeitet werden sollen. RAZ80 ist ein Zwei-Pass-Disassembler, der im Pass 1 Marken generiert und im Pass 2 Quellcode erzeugt und diesen auf Console, Printer oder Diskfile ausgibt. Das zu reassemblierende Programm (Inputfile) kann mit Optionen in verschiedene Segmente aufgeteilt werden, um Code von Text, Konstanten (Bytes, Words), Datenbereichen und Markentabellen zu unterscheiden.

2. Starten von RAZ80

Der allgemeine Aufruf von RAZ80 lautet:

```
RAZ80 [d:]infile[.ext] [option[s]]
```

Wird nur RAZ80 eingegeben, so erscheint eine Liste aller Optionen als Helptext, falls Sie eine Option nicht exakt wissen und dieses Papier zuweit von Ihrem PC entfernt ist.

Wird nur ein Inputfile angegeben, so wird ohne Optionen von diesem Filenamen mit der Extension .RAZ ein (Output-) Quellfile reassembliert. Das Inputfile kann auf einem beliebigen Laufwerk sein, das Outputfile wird auf dem aktuellen Laufwerk abgespeichert. Erproben Sie RAZ80, indem Sie

```
RAZ80 TEST.INP
```

eingeben. Bei kleiner Diskettekapazität können Sie sich mit

```
C>A:RAZ80 B:TEST.INP
```

helfen, wenn C: eine Diskette mit hinreichend Platz ist.

Während des Programmlauf wird in Pass 1 nach jeweils 32 neugenerierten Marken ein '+' auf Console ausgegeben. Bei Diskausgabe wird im Pass 2 nach jeweils 128 geschriebenen Zeilen ein '+' auf Console ausgegeben. Der Programmlauf kann mit ctrl-S gestoppt und mit ctrl-C abgebrochen werden.

3. Optionen

3.1 zur Ausgabe

Mit den Optionen L, P, T, U und X wird die Ausgabeform bestimmt. Die Option L definiert die Console als Ausgabegerät, auf der linken Seite werden zusätzlich Adressen und Opcode gelistet, ebenso bei der Druckerausgabe. Die Option P definiert den Drucker als Ausgabe, es werden 70 Zeilen pro Seite gedruckt, dieser Standardwert kann mit P:nn auf nn Zeilen geändert werden. T bewirkt, daß in der Ausgabedatei die Adressen und der Opcode als Kommentar miterzeugt werden. Mit U werden Großbuchstaben erzeugt, sonst nur Kleinbuchstaben. Mit der Option X wird festgelegt, daß statt eines Reassemblings ein Hexdump der Inputfile erzeugt wird.

3.2 zur Begrenzung des Reassemblings

Mit den Optionen S, E und O werden die Grenzen der Reassemblierung festgelegt. Mit S:XXXX wird die Startadresse auf XXXX vereinbart, ohne Angabe ist 100h die Startadresse. E:XXXX gibt die Endadresse, ohne Angabe wird bis zu Ende der Datei (EOF) reassembliert. Mit O:XXXX wird dem ersten Byte in der Inputdatei eine Adresse zugeordnet, ohne Angabe hat das erste Byte die für CP/M-Files übliche Adresse 100h. Die Hexadezimalzahlen XXXX können auch mit weniger als vier Stellen eingegeben werden. Zwischen den einzelnen Optionen brauch kein Trennzeichen sein, es sei den, diese wird von einer Hexzahl gefolgt und danach kommt eine weiter Option A bis E. Dann ist ein Leerzeichen einzufügen, da die Optionen sonst fehlgedeutet werden.

Mit G:XXXX und H:XXXX kann der Ausgabebereich weiter eingeengt werden. Die Option G:XXXX bewirkt eine Ausgabeunterdrückung bei Adressen kleiner XXXX, die Option H:XXXX bewirkt eine Ausgabeunterdrückung bei Adressen größer XXXX.

3.3 Segmente

Mit den Optionen A, B, C, D, I, J, M und N können einzelne Bereiche vereinbart werden. Ohne Angaben dieser Art ist ein Code-Segment ab Dateianfang eingestellt. Die Reihenfolge der Segmentadressen hat in aufsteigender Reihe zu erfolgen. Mit folgenden Optionen sind die entsprechenden Bereichseinstellungen möglich:

Option	Bedeutung	Beispiel
A:xxxx	ASCII-Segment	db 'Text $',0dh,8ah
B:xxxx	Byte-Segment	db 1,0afh,13h,7,0,0,0,0
C:xxxx	Code-Segment	ld hl,m03af
D:xxxx	Daten-Segment	ds 91
I:xxxx	Binary-Byte-Segment	db 01100101b
J:xxxx	Decimal-Byte-Segment	db 4,25,255
M:xxxx	Marken-Segment	dw m0005
N:xxxx	Word-Segment	dw 0dafh,012h,0ffffh

Werden in einem Code-Segment acht aufeinanderfolgende gleiche Bytes gefunden, so wird kein Opcode erzeugt, sondern ein Datensegment mit soviel Bytes, bis der nächste Unterschied festgestellt wird. z.B.:

```
DS   91,0e5h
```

Wird mehr als ein NOP erkannt, so wird analog verfahren.

3.4 Segment - Macros

4. Definitionsdateien

4.1 Optionsdateien (.RDF)

In einer Datei mit dem Namen der Inputdatei und dem Typ .RDF können die zur Reassemblierung nötigen Optionen abgespeichert werden. Diese Datei dient dazu, um bei großen Inputfiles die Bereichsaufteilung nicht jedesmal mit einzugeben. Die .RDF-Datei kann maximal 4KByte lang sein, es können maximal 512 Bereiche verwendet werden. Nachdem die Optionen in der Kommandozeile verarbeitet wurden, wird nach einer eventuell vorhandenen .RDF-Datei gesucht, wenn gefunden, so werden die Optionen daraus interpretiert bzw. gespeichert.

4.2 Symboldateien

Zur Markengenerierung im Pass 1 dienen die Adressen von Unterprogrammaufrufen, absoluten und relativen Sprüngen und von 16-bit-Ladebefehlen, die Marken erhalten ihre Adresse als Name, mit einem "M" vorangestellt. Greifen Marken in Befehle hinein, so werden diese als "hidden labels" bezeichnet. Hidden Labels entstehen durch Programmiertricks, weisen aber auch auf eine noch unsaubere Rückübersetzung hin. Wird im Pass 2 an Stellen, wo eine Marke stehen muß (hinter RET und unbedingten Sprüngen, nach HALT und am Anfang eines neuen Segments), keine gefunden, wird dies als undefined label markiert.

Um auch im Laufe der Rückübersetzung schon sinnvolle Markennamen zu verwenden, werden diese Symbole in einer speziellen Datei ihren Adressen zugeordnet. Wenn vorhanden, wird eine infile.SYM Datei eingelesen und die darin enthaltenen Symbole als Marken des Reassemblers in die Symboltabelle übertragen. Linker wie LINK80 oder LINKMT geben mit den Optionen Y bzw. W Symboltabellen in file.SYM aus, so daß mit RAZ80 effektiv Libraries von Compilern etc. geknackt werden können. Die .SYM-Datei hat folgenden Aufbau:

```
<4Zeichen-Hexadresse><space><Name><tab or crlf>....
.............<EOF>
z.B. 0005 BDOS      005C FCB       0100 TPA
     0080 DEFBUF
```

Der Symbolname kann aus maximal sieben Zeichen bestehen. Mit der Option V können einige CP/M-Standardnamen als Symbole genutzt werden, ohne das eine SYM-Datei erstellt werden muß. Es ist auch möglich, Ziffern als Symbole für Konstanten zu vereinbaren, diese Symbole werden natürlich nicht deklariert. Mit der V-Option werden folgende Symbole vereinbart:

```
0000 0         0003 iobyte    0004 cdrive
0005 bdos      000a 10        000b 11
0050 hour      0053 year      005c fcb
005d fcbnam    0064 100       0065 fcbtyp
0080 defdma    03e8 1000      2710 10000
ffff -1
```

4.3 Kommentardateien

Es hat sich gezeigt, daß es günstiger ist, Symbolnamen mittels Symboldateien einzuführen, als diese nach erfolgtem Rückübersetzen mit einem Texteditor auszutauschen. Man spart erheblich Zeit. Um aber auch schon frühzeitig Kommentare mit ins zu erzeugende Quellprogramm einzufügen, wurden Kommentardateien eingeführt. Diese enthalten Hexadezimaladressen und den zur Adresse gehörenden Kommentar. Dadurch ist es möglich, das vollständige Reassembling erst zu erstellen, wenn das Programm hinreichend verstanden wurde und keine Fehler mehr enthalten sind.

5. Installation

Da das Quellprogramm von RAZ80 mitvertrieben wird, kann jeder Nutzer RAZ80 nach eigenen Bedürfnissen verändern. Hauptsächlich wird sich das auf ein Verändern der globalen Konstanten am Anfang von RAZ80.MAC beschränken. Wenn Sie auf ihrem Drucker mit 12" Papier und 8 LPI arbeiten, so ist eine PAGLEN von 95 sinnvoller, als wenn jedesmal P:95 angegeben wird. Eventuell ist es auch notwendig, das die Länge einiger Tabellen vergrößert wird. Die Einzeichen-I/O wird über BIOS-Call's realisiert in den Routinen CONOUT, LSTOUT, KBHIT und CONIN. Auf Adresse 50h - 55h kann sich das aktuelle Datum befinden, es wird dann mit ins Quellprogramm aufgenommen.

6. Tips zum Reassemblieren

Bevor man mit dem Reassemblieren beginnt, sollte man sich ein Hexdump der Datei erzeugen. Wenn man die Ladeadresse des Programmes kennt, so sollte man den Hexdump mit der S- und O-Option auch dorthin legen. Man sollte den Hexdump auch ausdrucken, bei kürzeren Programmen reicht aber ein Hexdumpfile aus. Am Hexdump kann man schon eine erste Einteilung des Programmes in Segmente vornehmen. Suchen Sie nach Texten und nach großen Konstantenbereichen.

Wenn Sie dann eine .RDF-Datei mit der Segmenteinteilung erstellt haben, können Sie eine erste Rückübersetzung starten. Das erste Stück sehen Sie sich vielleicht erst einmal auf dem Bildschrim an (L-Option). Nach erfolgreicher Erzeugung einer Datei mit dem Typ RAZ sehen Sie sich diese an besten mit WordStar an. Als erstes müßten Sie überprüfen, ob das Programm an der richtigen Ladeadresse ist. Dazu müssen CALL's und absolute Sprünge zu sinnvollen Unterprogrammen und Programmfortsetzungen führen. Weiterhin muß der Programmstartpunkt bekannt sein. Ist dieses erste Reassembling halbwegs okay, macht man davon einen Ausdruck. Danach geht's an das Verstehen des Programms. Man kann vom Startpunkt anfangen, oder man sucht zunächst die Schnittstellen zum Betriebssystem oder zur Hardware (Systemcall's und I/O-Befehle). Alle hidden labels müssen untersucht werden. Die Segmenteinteilung muß im Normalfall mehrfach ergänzt werden. Parallel dazu können Marken mit Symbolnamen und verstandene Routinen mit Kommentaren versehen werden.

Ist ein sauberes Reassembling entstanden, so kann es neu übersetzt werden und mit COMP mit der Urfassung verglichen werden. Stimmen beide überein, so können eigentlich nur noch unerkannte Pointer zu Fehlern führen. Jetzt können Sie daran gehen, das Programm nach ihren Vorstellungen zu ändern.

7. Beispiellauf

Es soll das Programm TEST.INP reassembliert werden. Als erstes wird von dem Programm ein Hexdump gemacht, um eine Bereichsaufteilung vorzunehmen. Mit

```
RAZ80 TEST.INP LX   oder
RAZ80 TEST.INP PX
```

wird folgender Ausdruck erzeugt:

```
Z80 - Reassembler V3.3  (C) 1986 by MicroByte & BitResearch

0100  00 01 00 00  00 00 00 02  03 04 05 06  00 07 08 09  ................
                          .
                          .
                          .

05c0  41 62 20 68  69 65 72 54  65 78 74 0d  0a 24 cd 00  Ab hierText..$K.
05d0  8a 42 69 73  27 68 69 65  72 00 01 02  03 04 05 06  .Bis'hier.......
05e0  07 08 09 0a  00 01 02 04  08 10 20 40  80 ff ff ff  .......... @....
05f0  9e dd cb 00  a6 dd cb 00  ae dd cb 00  b6 dd cb 00  .]K.&]K..]K.6]K.
0600  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
0610  00 00 01 00  0a 00 64 00  e8 03 10 27  ff ff 11 11  ......d.h..'....
0620  11 11 11 11  11 11 11 11  11 11 11 11  11 11 11 00  ................
0630  00 00 00 00  00 00 00 00  c4 05 e4 05  21 06 2e 06  ........D.d.!...
0640  fe 05 24 24  21 00 01 11  01 01 01 ff  04 36 00 ed  ß.$$!........6.m
0650  b0 c9 01 02  04 08 10 20  40 80 55 aa  81 c3 e7 ff  0I..... @.U*.Cg.
0660  01 0a 10 a0  ff 0f 09 44  43 2d 54 65  78 74 a1 45  ... ...DC-Text!E
0670  6e 64 e5 00  00 00 00 45  6e 64 e5 00  00 00 00 00  nde....Ende.....
```

Dieser Hexdump dient beim weiteren Rückübersetzen dem Aufteilen in Bereiche.

Sollten Sie bei der Arbeit mit RAZ80 einen schnellen Überblick über die verschiedenen Optionen benötigen, so geben Sie ein:

```
A>raz80
Z80 - Reassembler V3.3  (C) 1986 by MicroByte & BitResearch

Use:  RAZ80 [d:]infile[.ext] [option[s]]
 Output:        infile.RAZ on default drive
 Options:
  s:xxxx        start address (def.0100)
  o:xxxx        address of first byte (def.0100)
  e:xxxx        end address (def.EOF)
  g:xxxx        start output            h:xxxx  stop output
  a:xxxx        db ascii segment        b:xxxx  db byte segm.
  c:xxxx        code segm.(def.)        d:xxxx  ds segm.
  i:xxxx        db binary segm.         j:xxxx  db decimal
  m:xxxx        dw label segment        n:xxxx  dw word segm.
  v             predefined CP/M-Symbols
  l             list pass2 on console
  p[:nn]        list pass2 on printer [page length]
  t             opcode comment in output
  u             upper case output
  x             hexdump
 xxxx  =        16-bit-hex-address
```

Ohne Einteilung in Bereiche kann nun eine erste Rückübersetzung vorgenommen werden. In folgenden Beispiel wird eine Ausgabe auf den Druckerkanal vorgenommen, die Seitenlänge wurde auf 20 Zeilen pro Seite herabgesetzt.

```
A>raz80 test.inp p:20
Z80 - Reassembler V3.3  (C) 1986 by MicroByte & BitResearch

Pass 1 - Label Generation
25 Label(s) found
Pass 2 - Source Generation
```

Auf dem Drucker erscheint folgendes Listing:

```
raz80 v3.3    reassembling of test.inp    page  1

        title   reass of test.inp
;reass'd with raz80 (c) mb&br 86
;*      reass'd         : 15-jul-86 22:59:59
;*      last update     :
;*      by
        .z80
        aseg
        org     100h
                ;
                ;
                        ext     m0000,m0011,m005c,m00a5,m0a00,m100a,m1111,m2e06
                        ext     me405,mffe7
                ;
0100 00         m0100:  nop
0101 010000     m0101:  ld      bc,m0000
0104 000000             ds      3,0
0107 02                 ld      (bc),a
0108 03                 inc     bc
<formfeed>
```

```
                    .
                    .
                    .

raz80 v3.3      reassembling of test.inp    page  3

0120 16ff               ld      d,0ffh
0122 17                 rla
0123 1880               jr      m00a5
                ;
0125 19         u0125:  add     hl,de
0126 1a                 ld      a,(de)
0127 1b                 dec     de
0128 1c                 in
Break with ^C
<formfeed>
```

Im Programmkopf sollte zur Dokumentation der Name des Bearbeiters eingegeben werden. Mit

```
ASEG
ORG  100H
```

wird das Programm als absolutes Codesegment für den Assembler ab 100h vereinbart. Marken, die außerhalb des Programmes liegen, werden in der ext(ernal)-Liste aufgeführt, externe Symbole werden dagegen mit equ(al) vereinbart. Auf Bildschirm und Drucker erscheint zusätzlich zum reassemblierten Quellcode auf der linken Seite die Adresse und der Objektcode. Die drei NOP's ab Adresse 104 werden zu

```
DS   3,0
```

(Data Storage 3 Byte, Constant Value 0) verkürzt. Zu Adresse 0125 wird eine Marke u<adresse> als undefined label ausgegeben, da der vorhergehende Befehl ein unbedingter Sprung war.

Wenn der Objektcode von CP/M-fremden Programmen kommt oder aus Overlays, so wird mit o:<adresse> eine andere Basisadresse als 100h festgelegt. Der Startpunkt muß dann aber auch entsprechend mit s:<adresse> verändert werden. Folgendes Beispiel zeigt ein solches Vorgehen:

```
A>raz80 test.inp p s:0 o:0 e:16
Z80 - Reassembler V3.3  (C) 1986 by MicroByte & BitResearch

Pass 1 - Label Generation
1 Label(s) found
Pass 2 - Source Generation
```

Auf dem Drucker erscheint folgendes Bild:

```
raz80 v3.3     reassembling of test.inp    page  1

        title   reass of test.inp
;reass'd with raz80 (c) mb&br 86
;*      reass'd         : 15-jul-86 23:05:45
;*      last update     :
;*      by
        .z80
        aseg
        org     100h
                        .phase  0000h
                ;
                ;
                ;
0000 00         m0000:  nop
0001 010000             ld      bc,m0000
0004 000000             ds      3,0
0007 02                 ld      (bc),a
0008 03                 inc     bc
0009 04                 inc     b
000a 05                 dec     b
000b 0600               ld      b,0
000d 07                 rlca
000e 08                 ex      af,af'
000f 09                 add     hl,bc
0010 0a                 ld      a,(bc)
0011 0b                 dec     bc
0012 0c                 inc     c
0013 0d                 dec     c
0014 0e00               ld      c,0
                        .dephase
                ;
                        end

0 hidden Label(s)
0 unused Label(s)
                  36 Lines written
```

Die geänderte Startadresse wird nicht mit

```
ORG  xxxx
```

vereinbart, sondern mit .PHASE und .DEPHASE, um leichter mit M80 und L80 ein neues lauffähiges Programm zu erzeugen.

Wenn Sie nach der ersten Rückübersetzung ihr Programm logisch in Bereiche einteilen können, so können Sie diese Aufteilung im <name>.rdf (reassembler-description-file) abspeichern. Dieses File ist ein normales ASCII-File und kann mit jedem Editor erstellt werden. Für TEST.INP wird folgendes File verwandt:

```
A>TYPE TEST.RDF
c:100
a:5c0 b:5d9 d:5f0n:610
m:638
c:644
i:652
j:660
a:667
c:67b
```

Weiterhin zu Adressen Symbolnamen vergeben werden. Dies ist vor allem bei der Rückübersetzung von Libraries (z.B.: FORLIB.REL) sinnvoll. Dabei kann das SYM-File vom Linker mit

```
L80 FORLIB,FORLIB.SYS/N/E/Y
```

erzeugt werden oder wie folgendes Beispiel mit einem Editor geschrieben werden:

```
A>type test.sym
0000 0          0005 bdos       005C FCB        2424 EXTERN
05C0 Text       05D9 data
0644 Clear      0638 jmptab     0610 words      05f0 area
0645 hidden
```

Zum Einfügen von Kommentaren wird folgende Datei genutzt:

```
A>type test.cmt
Testfile fuer RAZ80-Kommentare
110<TAB>comment at addr 110
118<TAB>comment at addr 118
<TAB>second line at 118
642<TAB>
<TAB>*** Clear 100 - 5FF ***
64D<TAB>fill with zero's
```

Der abschließende Reassemblerlauf liefert folgenden Ausdruck:

```
C>raz80 test.inp pg:599
Z80 - Reassembler V3.3  (C) 1986 by MicroByte & BitResearch

Loading .RDF-File
Loading .SYM-File
Loading .CMT-File
Pass 1 - Label Generation

26 Label(s) found
Pass 2 - Source Generation
raz80 v3.3    reassembling of test.inp    page  1

        title   reass of test.inp
;reass'd with raz80 (c) mb&br 86
;*      reass'd         : 15-jul-86 23:59:59
;*      last update     :
;*      by
        .z80
        aseg
        org     100h
                ;
                        entry   text,data,area,words,jmptab,clear,hidden
                ;
                        ext     m00a5
                ;
boot    equ     0000h
bdos    equ     0005h
fcb     equ     005ch
extern  equ     2424h
                ;
0599 ddcb00f6           set     6,(ix)
059d ddcb00fe           set     7,(ix)
05a1 00000000           ds      4,0
05a5 fd                 db      0fdh
05a6                    ds      11,0ddh
05b1 edff               db      0edh,0ffh
05b3 fd                 db      0fdh
05b4 edff               db      0edh,0ffh
05b6                    ds      10,0
```

```
                ;
05c0 41622068   text:   db      'Ab h'
05c4            m05c4:  db      'ierText',0dh
05cc                    db      0ah,'$',0cbh,0,8ah,'Bis'
05d4                    db      27h,'hier'
                ;
05d9            data:   db      0,1,2,3,4,5,6,7
05e1 08090a             db      8,9,0ah
05e4            m05e4:  db      0,1,2,4,8,10h,20h,40h
05ec 80ffffff           db      80h,0ffh,0ffh,0ffh
                ;
05f0            area:   ds      000eh
05fe            m05fe:  ds      0012h
                ;
0610            words:  dw      0000h,0001h,000ah,0064h
0618                    dw      03e8h,2710h,0ffffh,1111h
0620 1111               dw      1111h
0622                    dw      1111h,1111h,1111h,1111h
062a 11111111           dw      1111h,1111h
062e            m062e:  dw      0011h,0000h,0000h,0000h
0636 0000               dw      0000h
                ;
0638 c405       jmptab: dw      m05c4
063a e405               dw      m05e4
063c 2106               dw      m0621
063e 2e06               dw      m062e
0640 fe05               dw      m05fe
0642 2424               dw      extern       ;
                ; *** clear 100 - 5ff ***
                ;
0644 210001     clear:  ld      hl,m0100
0647 110101             ld      de,m0101
064a 01ff04             ld      bc,m04ff
064d 3600               ld      (hl),0       ; fill with zero's
064f edb0               ldir
0651 c9                 ret
                ;
0652 01         u0652:  db      00000001b
0653 02                 db      00000010b
0654 04                 db      00000100b
0655 08                 db      00001000b
0656 10                 db      00010000b
<formfeed>

raz80 v3.3      reassembling of test.inp    page  2

0657 20                 db      00100000b
0658 40                 db      01000000b
0659 80                 db      10000000b
065a 55                 db      01010101b
065b aa                 db      10101010b
065c 81                 db      10000001b
065d c3                 db      11000011b
065e e7                 db      11100111b
065f ff                 db      11111111b
                ;
0660            u0660:  db      1,10,16,160,255,15,9
                ;
0667            u0667:  dc      'DC-Text!'
066f                    db      'End',0e5h,0,0,0,0
0677 456e64e5           dc      'Ende'
                ;
067b            u067b:  ds      5,0
                ;
                        end
```

```
;hidden labels:
;m012e m04ff m0621 hidden
;unused labels:
;u0652 u0660 u0667 u067b

                4 hidden Label(s)
                4 unused Label(s)
                96 Lines written
```

Bei Ausgabe auf Diskfile erscheint folgende Ausschrift:

```
C>raz80 ddtz.com
Z80 - Reassembler V3.3  (C) 1986 by MicroByte & BitResearch

Pass 1 - Label Generation
++++++++++++++++++++++++++
886 Label(s) found
Pass 2 - Source Generation
++++++++++++++++++++++++++++++++++++
Unused Table Full+++++++
160 hidden Label(s)
256 unused Label(s)
5960 Lines written
```

So, das muß jetzt an Erklärung reichen. Schluß für heute. Dieses Programm sollte jedermann, der es benötigt, zur Verfügung gestellt werden, kostenlos. This program is a public domain! Kommerzielle Vermarktung verboten !! Falls Sie Fehler finden oder Verbesserungsvorschläge haben, lassen Sie es mich wissen.

<<< End of Session >>>

8. Referenzen

CP/M	-	Copyright by Digital Research Inc.
WordStar	-	Copyright by Micropro Inc.
Macro80	-	Copyright by Microsoft Inc.
COMP	-	Copyright by JDE

Anm.: MicroByte & BitResearch ist keine Firma, sondern ein Pseudonym.