Prof. Dr. H. Völz

Z80 sortiert Texte im Nu

Eine handliche Z80-Routine sortiert Zeichenketten beliebiger und unterschiedlicher Länge in Windeseile. Groß- und Kleinschreibung, auch gemischt, sind zulässig. Selbst Umlaute und „ß" können zusätzlich eingeordnet werden. Ein ideales Hilfsprogramm für Sachwortverzeichnisse und Register.

Das Sortieren von Texten, z. B. Sachwortverzeichnissen, Namens- oder Telefonlisten kommt immer wieder vor. Vielfach wird sogar behauptet, etwa 30 % der Textverarbeitung entfalle auf dieses Problem. Dennoch fehlt in den meisten Textverarbeitungssystemen eine solche Routine. Im wesentlichen werden nur Routinen in Basic, Pascal oder anderen Hochsprachen publiziert, und dies sogar relativ häufig. Beschrieben werden viele Methoden mit sehr unterschiedlicher Effektivität. Sie tauschen meist zwei falsch geordnete Wörter. Versucht man nun eine solche Routine in Maschinensprache zu übertragen, so steht man bei jedem Austausch etwa vor folgendem Problem:

- Die Worte sind meist unterschiedlich lang.
- Sollen sie ausgetauscht werden, muß zusätzlich der zwischen ihnen liegende Teil verschoben werden.
- Dies verlangt zuvor die Auslagerung des längeren Wortes.

Die Anzahl der Berechnungen für die Adressen und die Anzahl der Vertauschungs- und Verschiebeoperation im Speicher sind folglich je Tausch sehr umfangreich. Was nimmt es da Wunder, wenn auf diese Weise die Maschinensprache gegenüber Hochsprachen keinen entscheidenden Gewinn bringt, und das abgesehen vom hohen Programmaufwand.

Im Z80-Befehlsvorrat existieren nun die sehr nützlichen Verschiebe- und Vergleichsbefehle LDDR, LDIR, CPDR und CPIR. Es liegt also nahe, einen speziellen Sortieralgorithmus auf dieser Basis zu entwickeln.

```
;Schnelle Sortierroutine fuer Z-8Ø
;mit Beruecksichtigung von
;Gross- und Kleinschreibung
;
;H.Voelz     3.1.85
;
;HL:=BC, BC:=HL-BC
ADR1    PUSH    BC
        OR      A
        SBC     HL,BC
        PUSH    HL
        POP     BC
        POP     HL
        RET
;
;Hier ist Start
;
SORT    LD      HL,(ENADR)
        LD      (EN1),HL
        DEC     HL
        XOR     A
EN2SU   DEC     HL
        CP      (HL)
        JR      NZ,EN2SU-©
        INC     HL
NWORT   LD      (EN2),HL
        XOR     A
        PUSH    HL
        LD      DE,(STADR)
        OR      A
        SBC     HL,DE
        POP     HL
        RET     Z
        DEC     HL
EN3SU   DEC     HL
        CP      (HL)
        JR      NZ,EN3SU-©
        INC     HL
        LD      (EN3),HL
;Vergleichen
        LD      HL,(EN2)
        LD      DE,(EN3)
VERGL   LD      A,(HL)
        AND     5FH
        LD      B,A
        LD      A,(DE)
        AND     5FH
        CP      B
;Ergebnis (DE)-(HL)
        JR      Z,GLEICH-©
VERGL2  JR      NC,TAUSHE-©  ;(DE)>(HL)
;naechstes Wort
NWOR    LD      HL,(EN2)
        LD      (EN1),HL
        LD      HL,(EN3)
        JR      NWORT-©
;gleiche Worte
GLEICH  CP      Ø
        JR      Z,WEITER-©
WEIT    INC     HL
        INC     DE
        JR      VERGL-©
;Entscheidung bei ØØ bzw. 2Ø
WEITER  LD      A,(DE)
        CP      Ø
        JR      Z,NWOR-©
        CP      (HL)
        JR      Z,WEIT-©
;Umordnen; 1.Auslagern
TAUSHE  LD      HL,(EN2)
        LD      BC,(EN3)
        CALL    ADR1
        PUSH    BC      ;fuer spaeter
        LD      DE,(ENADR)
        LDIR
;EN1 suchen
EN1P    LD      HL,(EN3)
        LD      DE,(EN1)
VERGL1  LD      A,(HL)
        AND     5FH
        LD      B,A
        LD      A,(DE)
        AND     5FH
        CP      B
;Ergebnis (DE)-(HL)
        JR      Z,GLEI2-©
        JR      NC,WORTGE-©  ;(EN1)>(EN3)
;naechstes Wort suchen
NWORT2  LD      HL,(ENADR)
        LD      DE,(EN1)
        OR      A
        SBC     HL,DE
        JR      Z,ALLES-©
        PUSH    HL
        POP     BC      ;Laenge
        XOR     A
        PUSH    DE
        POP     HL
        CPIR
        LD      (EN1),HL
        JR      EN1P-©
;gleiche Worte
GLEI2   CP      Ø
        JR      Z,WEIT2-©
WEI2    INC     HL
        INC     DE
        JR      VERGL1-©
;Entscheidung ob ØØ oder 2Ø
WEIT2   LD      A,(DE)
        CP      Ø
        JR      Z,NWORT2-©
        CP      (HL)
        JR      Z,WEI2-©
;Stelle gefunden, verschieben
WORTGE  LD      HL,(EN1)
        LD      BC,(EN2)
        CALL    ADR1
        LD      DE,(EN3)
        LDIR
;zurueckholen
        POP     BC
        LD      HL,(ENADR)
        JR      NVEKT-©
;alles verschieben
ALLES   LD      HL,(ENADR)
        POP     BC
        ADD     HL,BC
        LD      BC,(EN2)
        CALL    ADR1
        LD      DE,(EN3)
NVEKT   LDIR
        LD      HL,(EN3)
        PUSH    HL
        XOR     A
NVEKTS  CP      (HL)
        INC     HL
        JR      NZ,NVEKTS-©
        LD      (EN1),HL
        POP     HL
        JP      NWORT
;
;Speicherplaetze
;
STADR   DEFW    Ø
;Enthaelt Adresse fuer Filebeginn
;weist auf ØØH
ENADR   DEFW    Ø
;Enthaelt Adresse fuer Fileende
;weist auf Speicherstelle hinter ØØH
EN1     DEFW    Ø
EN2     DEFW    Ø
EN3     DEFW    Ø
;
```

Bild 1. Unglaublich schnell arbeitet diese Sortierroutine in Z80-Maschinensprache

Das in *Bild 1* abgedruckte Programm arbeitet im Gegensatz zu den meisten sonstigen Sortiertechniken nicht mit Austausch-, sondern nur mit Verschiebeoperationen.

Verschobenes Wort paßt genau in die Lücke

Es sei davon ausgegangen, daß die einzelnen Wörter hintereinander nur durch 00 getrennt im Speicher vorliegen. Warum 00 und nicht 0DH (Carriage Return) verwendet wird, ist später ersichtlich.

Ein Beispiel:
00 Wort 00 Wort 200 abends 00 Hut + 00 usw.

Es werden also bewußt Groß-/Kleinschreibung, Leerraum, Zahlen und Satzzeichen zugelassen. Allgemein gilt also die Anordnung von *Bild 2*. Die Adresse des Textbeginns (erstes ASCII-Zeichen) sei an der 2-Byte-Speicherstelle (STADR) abgelegt, das Ende (1 Byte hinter dem letzten 00) in der 2-Byte-Speicherstelle (ENADR). Der Vorgang des Sortierens ist stark vereinfacht in Bild 2 gezeigt. Vom Ende her werden benachbarte Worte (im Bild als Zahlen dargestellt) auf die richtige Reihenfolge getestet. Beim ersten falsch angeordneten (hier die 7) wird eine Marke gesetzt, beim Vergleichswort eine zweite. Jetzt wird rückwärts nach dem nächstliegenden größeren gesucht. Anschließend wird das falsch liegende Wort an das Textende gelegt, der Text zwischen den Marken verschoben und das Wort an die entstandene, genau passende Lücke gelegt.

Von der Marke (ganz links) wird nun der Vergleichsvorgang benachbarter Wörter wiederholt, und falsch angeordnete Wörter werden wieder an die richtige Stelle gelegt.

Wenn dieser Vorgang an der Startadresse anlangt, ist der Text sortiert.

Es sei betont, daß der wirkliche Sortiervorgang erheblich komplizierter ist. Die Wörter müssen ASCII-Zeichen für ASCII-Zeichen verglichen werden. Das Ende 00 muß erkannt werden. Sonderoperationen sind notwendig, damit an den Enden, d. h. bezüglich erster und letzter Wörter nichts passiert usw. Dies sind die Gründe, daß normalerweise nur Textketten genau formatierter Länge sortiert werden.

Sie würden aber zusätzlich oft sehr wertvollen Speicherplatz beanspruchen und sind, wie das beschriebene Programm zeigt, keineswegs notwendig.

Einen lauffähigen Speicherauszug (ab Adresse 0000H zeigt *Bild 3*. Bei der Kürze des Programms kann es auch leicht als Maschinenroutine von Hochsprachen aus mit CALL, USR oder ähnlichen Befehlen aufgerufen werden.

Für die Textsortierung ist das Unterprogramm SORT entscheidend. Es schaltet über die Maske 5FH die Bits für die Großschreibung und die Grafikzeichen zum Vergleich aus. Dadurch werden einmal die Klein- und Großbuchstaben gleichrangig behandelt, zum anderen ist es dadurch möglich, auch die Umlaute und „ß" gleichrangig zu behandeln. Sie sind durch eine kurze Zusatzroutine nur auf die passenden Grafikzeichen zu transformieren und am Ende wieder zurückzuwandeln. Durch diese Maske entstehen aber Probleme, wenn CR = 0DH bestehen bleibt. Es wird deshalb entgegen üblichen Zeichenketten in 00 gewandelt. Dann entsteht nach der Maskierung mit 5FH ein Leerzeichen. Deshalb sind Teilroutinen zur Unterscheidung von 00 = Ende und 20H = Space notwendig.

Der Vorteil dieser Operation ist, daß somit auch alle Satzzeichen und Zahlen optimal angeordnet werden, ja selbst alle Steuerzeichen sind verwendbar.

ADDR	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000:	C5	B7	ED	42	E5	C1	E1	C9	2A	E0	00	22	E2	00	2B	AF
0010:	2B	BE	20	FC	23	22	E4	00	AF	E5	ED	5B	DE	00	B7	ED
0020:	52	E1	C8	2B	2B	BE	20	FC	23	22	E6	00	2A	E4	00	ED
0030:	5B	E6	00	7E	E6	5F	47	1A	E6	5F	B8	28	0D	30	1B	2A
0040:	E4	00	22	E2	00	2A	E6	00	18	CB	FE	00	28	04	23	13
0050:	18	E1	1A	FE	00	28	E8	BE	28	F4	2A	E4	00	ED	4B	E6
0060:	00	CD	00	00	C5	ED	5B	E0	00	ED	B0	2A	E6	00	ED	5B
0070:	E2	00	7E	E6	5F	47	1A	E6	5F	B8	28	1A	30	28	2A	E0
0080:	00	ED	5B	E2	00	B7	ED	52	28	32	E5	C1	AF	D5	E1	ED
0090:	B1	22	E2	00	18	D5	FE	00	28	04	23	13	18	D4	1A	FE
00A0:	00	28	DB	BE	28	F4	2A	E2	00	ED	4B	E4	00	CD	00	00
00B0:	ED	5B	E6	00	ED	B0	C1	2A	E0	00	18	10	2A	E0	00	C1
00C0:	09	ED	4B	E4	00	CD	00	00	ED	5B	E6	00	ED	B0	2A	E6
00D0:	00	E5	AF	BE	23	20	FC	22	E2	00	E1	C3	15	00	00	00
00E0:	00	00	00	00	00	00	00	00								

Bild 3. Lauffähiger Maschinencode mit Startadresse 0000H

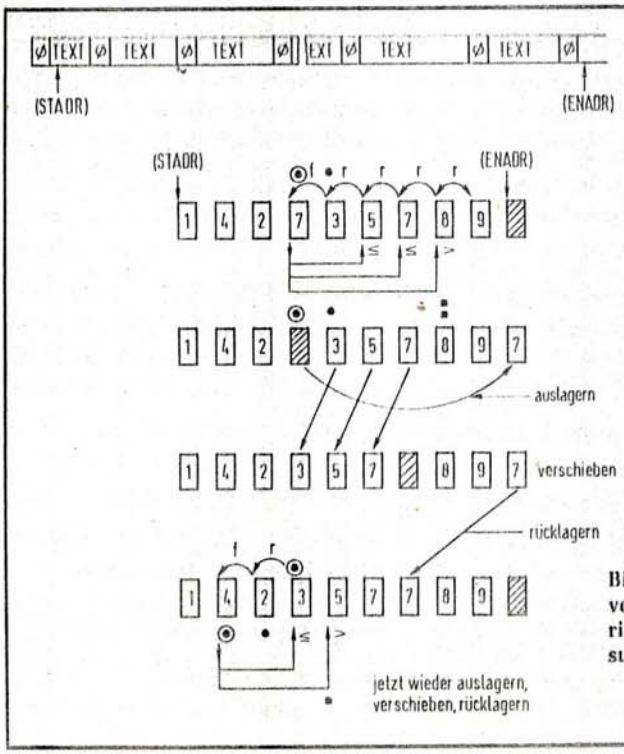

Bild 2. Die Wörter werden vom Textende her auf die richtige Reihenfolge untersucht

Angefügte Wörter schnell eingereiht

Das Programm benötigt zum Sortieren im ungünstigsten Fall n(n-1)/2 Vergleiche und n-1 Verschiebungen. Diese Werte sehen im Vergleich zu den bekannten

Gemessene Sortierzeiten mit einer 2-MHz-CPU

Wörter[1]*	200	500	1000	2000
Zeit/s	3	15	60	200

[1]* Wortlänge max. 10 Zeichen

Wortlänge[2] Zeichen	5	10	20	40
Zeit/s	13	15	22	35

[2] Zahl der Wörter: 500

sonstigen Werten sehr günstig aus. Deshalb wurden gerichtete Zeitmessungen mit Zufallswörtern aller ASCII-Zeichen von 20H bis 7FH und Zufallslängen (1 bis Nmax) durchgeführt. Dabei ergaben sich für eine mit 2 MHz betriebene CPU die in der *Tabelle* dargestellten Werte. Sie sind, verglichen mit den Werten aus der Literatur, beachtlich kurz.

In brauchbarer Näherung gilt für die Sortierzeit bei Zufallswörtern und 2 MHz die folgende Formel:

$$T/s = 2 \cdot 10^{-5} \cdot \sqrt{\text{Wortlänge}} \cdot (\text{Wortanzahl})^2$$

Es überwiegen hiernach also eindeutig die Vergleichsoperationen, und das, obwohl sie bereits zeitoptimal gestaltet wurden.

Es sei aber betont, daß ein bereits sortiertes Feld extrem schnell erkannt wird. Wenn das Feld mit den 2000 Wörtern der maximalen Länge 10 sortiert ist, durchläuft der Algorithmus in 0,8 s (!) das Feld. Dies hat zur Folge, daß auch einzelne hinten (!) angefügte Wörter extrem schnell eingeordnet werden. Im wesentlichen sind also die Werte der Formel bereits Maximalzeiten. Dies bedeutet, daß der Algorithmus eigentlich alle normalen Sortieroperationen in vorbildlicher Weise realisiert.

Der Autor ist Bereichsleiter an der Akademie für Wissenschaften, Ostberlin, DDR.

Literatur

[1] *Wehringner, A.*: Sortieren in Basic. Elektronikschau 11 (1981), Heft 4, Seite 34...39.

[2] *Kollmann, M.*: Sortieren in Sekundenschnelle. Chip 1981, Heft 7, Seite 45...52.

[3] *Turalski, N., Hansohm, J.*: Ein empirischer Vergleich von Sortieralgorithmen auf Microcomputern. Angewandte Informatik 1984, Heft 5, Seite 375...378.