DudenSkt

Sorting Program for Sanskrit Files

Quick Reference for Programmers

DudenSkt is an extremely fast 80386 assembly language program with a Basic shell designed for sorting mixed Sanskrit and English/German ASCII textfiles with this modified 437 encoding:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
032		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
048	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
064	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
080	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
096	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	□
128	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
144	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	₧	ƒ
160	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
176	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
192	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
208	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	█	▄	▌	▐	▀
224	α	ß	ā	ī	ū	ṛ	ṝ	ḷ	ṅ	ñ	ṇ	ṭ	ḍ	ś	ṣ	ṃ
240	ḥ	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	■	□

Yellow = separators; Red = German diacritics; Green = Sanskrit diacritics

DudenSkt can sort large textfiles (e.g. the entire Mahabharata). The textfiles must consist of lines of text separated by carriage return linefeed (hexadecimal 0D 0A). Each line may have a length of up to 2000 characters, but only the first 100 characters of each line will be sorted (sort depth).

With the help of KOKO (see separate manual) any existing textfile can be converted to the DudenSkt encoding, provided that either the existing textfile does not contain more diacritics than the red and green diacritics shown in the above encoding table or the existing textfile can be reduced without loss of information to the DudenSkt encoding for sorting purposes, e.g. by converting uppercase Sanskrit diacritics to lowercase Sanskrit diacritics via KOKO.

The program DudenSkt is available only to professional programmers who are able to convert textfiles existing in other formats to flawless textfiles in the file format required for DudenSkt and who are able to convert the sorted textfiles back to the original format, if needed. Therefore, no effort has been made to make the program DudenSkt "user-friendly" for ordinary PC users.

Types of Textfiles

DudenSkt can sort the following types of textfiles:

1. Simple Sanskrit Textfiles

Simple files are lists of Sanskrit words only or consist of Sanskrit verse lines only.

Before sorting:
dharmakṣetre kurukṣetre samavetā yuyutsavaḥ /1.1./
māmakāḥ pāṇḍavāś caiva kim akurvata saṃjaya //1.1.//
dṛṣṭvā tu pāṇḍavānīkaṃ vyūḍhaṃ duryodhanas tadā /1.2./
ācāryam upasaṃgamya rājā vacanam abravīt //1.2.//

After sorting:
ācāryam upasaṃgamya rājā vacanam abravīt //1.2.//
dṛṣṭvā tu pāṇḍavānīkaṃ vyūḍhaṃ duryodhanas tadā /1.2./
dharmakṣetre kurukṣetre samavetā yuyutsavaḥ /1.1./
māmakāḥ pāṇḍavāś caiva kim akurvata saṃjaya //1.1.//

Separator "\"

Any string enclosed by a pair of "\" will be ignored during sorting so that intervening remarks or – in the case of verse indexes – prefixed numbers will not disturb the Sanskrit sorting order.
Example of a pada index with prefixed book-chapter-verse numbering:

Before sorting:
\60230011\dharmakṣetre kurukṣetre samavetā yuyutsavaḥ
\60230013\māmakāḥ pāṇḍavāś caiva kim akurvata saṃjaya
\60230021\dṛṣṭvā tu pāṇḍavānīkaṃ vyūḍhaṃ duryodhanas tadā
\60230023\ācāryam upasaṃgamya rājā vacanam abravīt

After sorting:
\60230023\ācāryam upasaṃgamya rājā vacanam abravīt
\60230021\dṛṣṭvā tu pāṇḍavānīkaṃ vyūḍhaṃ duryodhanas tadā
\60230011\dharmakṣetre kurukṣetre samavetā yuyutsavaḥ
\60230013\māmakāḥ pāṇḍavāś caiva kim akurvata saṃjaya

2. Sanskrit Textfiles with Number Subsort

These textfiles consist of lines with the following structure:

Sanskrit text | number

Numbers must be integer numbers in range 1...999999999 (or up to 1999999999), i.e. 9 digits plus – if needed – "1" as prefix digit to allow for zero-padding of topmost partial number, see below.

Before sorting:

dharmakṣetre | 1060230011 i.e. Prefix "1", Book 06, Chapter 023, Verse 001, Pada 1

kurukṣetre | 1060230011

samavetā | 1060230011

kurukṣetre | 1010031441

kurukṣetre | 1010031453

kurukṣetre | 1010950075

After sorting:

kurukṣetre | 1010031441, 1010031453, 1010950075, 1060230011

dharmakṣetre | 1060230011

samavetā | 1060230011

Separator "|"

The sequence after the separator "|" must be an integer number. To guarantee correct sorting, chapter-verse numbers must be zero-padded (except the topmost book number). For instance:

- If the text comprises 20 books, they may be numbered without zero-padding: 1, 2, 3 ... 19, 20
- If each book comprises up to 20 chapters, they must be zero-padded: 01, 02, 03 ...
- If each chapter comprises up to 200 verses, they must be zero-padded: 001, 002, 003 ...

In above case, numbers following "|" would have two different formats: "bccvvv" or "bbccvvv". If you need the topmost number zero-padded, you must prefix the digit "1", format: 1bbccvvv. E.g.: book 2, chapter 3, verse 4 would be encoded 10203004 (1+02+03+004) instead of 203004.

Sort Depth of Number References:

Up to 1000 number references for each single Sanskrit word are subsorted and packed into one large string, whereby the sorted number references are separated by commas.

If there are more than 1000 references to one single word, e.g. in the case of function words like "ca = and" ("ca" occurs more than 30,000 times in the Mahabharata!), only 1000 references will be sorted and packed. As a warning the number 1999999999 will be appended. Example:

ca | 1010010115, 1010010143, 1010010211, 1010010253, 1010010263, ..., 1999999999

If you need all references of "ca" etc., you will have to split a giant file into several smaller files.

3. Sanskrit-English (or Sanskrit-German) Textfiles

These textfiles consist of lines with the following structure:
Sanskrit word=English (or German) word

You may also add number references. In this case the textfile has the enhanced structure:
Sanskrit word=English (or German) word | number

Separator "="

The string before "=" is sorted according to the Devanagari alphabet, and the string after "=" is sorted according to DUDEN: Lower and uppercase letters are treated as if they were the same. In addition, German umlauts are sorted, as if they were non-umlauts, and eszett is sorted as ss.
English words are sorted according to the sequence of the entries in English dictionaries.

Before sorting:
paralokaḥ = Jenseits | 1752
gam, gacchati, 1. = erlangen | 1669
param, Konj. = jedoch | 881
gam, gacchati, 1. = gehen | 2
parama, Superl. = größt | 1859
gam, gacchati, 1. = aufsuchen | 107
param, Adv. = sehr | 1508
parama, Superl. = größt | 1860

After sorting:
gam, gacchati, 1. = aufsuchen | 107
gam, gacchati, 1. = erlangen | 1669
gam, gacchati, 1. = gehen | 2
param, Adv. = sehr | 1508
param, Konj. = jedoch | 881
parama, Superl. = größt | 1859, 1860
paralokaḥ = Jenseits | 1752

Ulrich Stiehl, 8th March, 2003

Excerpt from Program Source

The complete sorting order is shown in every detail in this excerpt from the assembly program.

1. Data Section Duden (German and English)

```
; DATA D01: AS7STRCOMP
; ====================
;
D01A      equ       100                 ;Sortiertiefe: 100 + 00-Endmarker
;
; Vorab geprüfte Sondercodes (in Codetab mit Wert 00)
; ---------------------------------------------------
; trenner: Rightstring nach   ".......|..." wird ignoriert
; flipper: Midstring zwischen "...\...\..." wird ignoriert
; eszett:  "β" wird in "SS" umgewandelt (sofern neustring < maxlen)
;
D01B      equ       124                 ;"|"
D01C      equ       092                 ;"\"
D01D      equ       225                 ;"β"
;
; Neustrings str1$ und str2$ nach Konvertierung
; ---------------------------------------------
;
D01E      db        100 dup (1)         ;100 + 1 = 101 + 1 für Align = 102
          db        2   dup (0)
;
D01F      db        100 dup (1)
          db        2   dup (0)
;
; Mitsortierte Sondercodes
; ------------------------
;
; 032 11 " "  Space bei Bedarf mitsortieren
; 044 12 ","  Komma bei Bedarf mitsortieren
;
; 035 01 "#"  Wörterbuchtrenner
; 061 02 "="  Wortpaar-Trenner
;
; 033 03 "!"  Vorkode
; 036 04 "$"  Vorkode
; 064 05 "@"  Vorkode
;
D01G      dw        9901h                       ;Signatur 01
          db        "aaa>"                      ;Codetab-Startmarker
;
D01H      db        00,00,00,00,00,00,00,00     ;000: Ctrl-Zeichen
          db        00,00,00,00,00,00,00,00     ;008: Ctrl-Zeichen
          db        00,00,00,00,00,00,00,00     ;015: Ctrl-Zeichen
          db        00,00,00,00,00,00,00,00     ;024: Ctrl-Zeichen
;
          db        11,03,00,01,04,00,00,00     ;032: !=03, #=01 $=04
          db        00,00,00,00,12,00,00,00     ;040: "," = 12
          db        48,49,50,51,52,53,54,55     ;048: "0 - 7"
          db        56,57,00,00,00,02,00,00     ;056: "8 - 9", "=" = 02
;
          db        05,65,66,67,68,69,70,71     ;064: @=06, "A - G"
          db        72,73,74,75,76,77,78,79     ;072: "H - O"
          db        80,81,82,83,84,85,86,87     ;080: "P - W"
          db        88,89,90,00,00,00,00,00     ;088: "X - Z"
;
          db        00,65,66,67,68,69,70,71     ;096: "a - g"
          db        72,73,74,75,76,77,78,79     ;104: "h - o"
          db        80,81,82,83,84,85,86,87     ;112: "p - w"
          db        88,89,90,00,00,00,00,00     ;120: "x - z"
;
```

```
          db        67,85,69,65,65,65,65,67       ;128: "Çüéâäàåç"
          db        69,69,69,73,73,73,65,65       ;136: "êëèïîìÄÅ"
          db        69,65,65,79,79,79,85,85       ;144: "ÉæÆôöòûù"
          db        89,79,85,67,76,89,80,70       ;152: "ÿÖÜ¢£¥Pƒ"
;
          db        65,73,79,85,78,78,65,79       ;160: "áíóúñÑªº"
          db        00,00,00,00,00,00,00,00       ;168: Grafikzeichen
          db        00,00,00,00,00,00,00,00       ;176: Grafikzeichen
          db        00,00,00,00,00,00,00,00       ;184: Grafikzeichen
;
          db        00,00,00,00,00,00,00,00       ;192: Grafikzeichen
          db        00,00,00,00,00,00,00,00       ;200: Grafikzeichen
          db        00,00,00,00,00,00,00,00       ;208: Grafikzeichen
          db        00,00,00,00,00,00,00,00       ;216: Grafikzeichen
;
; Sanskrit-Codes mitsortieren, damit gleiche Wörter gleich sind
;
          db        00h,00h,61h,63h,65h,66h,67h,68h  ;224: "αβ" + "āīūṛṝḷ"
          db        73h,78h,7Dh,79h,7Bh,8Ch,8Dh,6Dh  ;232: "ṁñṇṭḍśśṃ"
          db        6Eh,00h,00h,00h,00h,00h,00h,00h  ;240: "ḥ" + Sonderz.
          db        00h,00h,00h,00h,00h,00h,00h,00h  ;248: Sonderzeichen
;
; Für normale deutsche Sortierung diesen Block nehmen
;
;         db        00,00,00,00,00,00,00,00       ;224: 225: "β"
;         db        00,00,00,00,00,00,00,00       ;232: Griechisch
;         db        00,00,00,00,00,00,00,00       ;240: Sonderzeichen
;         db        00,00,00,00,00,00,00,00       ;248: Sonderzeichen
;
          db        "<zzz"                        ;Codetab-Endmarker
```

2. Data Section Sanskrit

```
; DATA D07: AS7SANSCOMP
; =====================
;
D07A      equ       100                 ;Sortiertiefe: 100 + 00-Endmarker
;
; Vorab geprüfte Sondercodes (in Codetab mit Wert 00)
; ---------------------------------------------------
; trenner: Rightstring nach   ".......|..." wird ignoriert
; flipper: Midstring zwischen "...\...\..." wird ignoriert
;
D07B      equ       124                 ;"|"
D07C      equ       092                 ;"\"
;
; Neustrings str1$ und str2$ nach Konvertierung
; ---------------------------------------------
;
D07E      db        100 dup (1)         ;100 + 1 = 101 + 1 für Align = 102
          db        2   dup (0)
;
D07F      db        100 dup (1)
          db        2   dup (0)
;
; Mitsortierte Sondercodes
; ------------------------
; 035 01 "#"
; 044 02 ","
; 063 03 "="
;
D07G      dw        9907h                         ;Signatur 07
          db        "aaa>"                        ;Codetab-Startmarker
;
; a    61-   60
; ā    E2    61 *
; i    69    62 *
; ī    E3    63 *
; u    75    64 *
; ū    E4    65 *
; ṛ    E5    66 *
; ṝ    E6    67 *
; ḷ    E7    68 *
; e    65    69 *
; ai   65=   6A
; o    6F    6B *
; au   65=   6C
; ṃ    EF    6D *
; ḥ    F0    6E *
; k    6B-   6F
; kh   6B=   70
; g    67-   71
; gh   67=   72
; ṅ    E8    73 *
; c    63-   74
; ch   63=   75
; j    6A-   76
; jh   6A=   77
; ñ    E9    78 *
; ṭ    EB-   79
; ṭh   EB=   7A
; ḍ    EC-   7B
; ḍh   EB=   7C
; ṇ    EA    7D *
; t    74-   7E
; th   74=   7F
; d    64-   80
; dh   64=   81
; n    6E    82 *
; p    70-   83
; ph   70=   84
; b    62-   85
```

```
; bh  62=   86
; m   6D    87 *
; y   79    88 *
; r   72    89 *
; l   6C    8A *
; v   76    8B *
; ś   ED    8C *
; ṣ   EE    8D *
; s   73    8E *
; h   68    8F *
;
D07H      db        00,00,00,00,00,00,00,00       ;00: Ctrl-Zeichen
          db        00,00,00,00,00,00,00,00       ;08: Ctrl-Zeichen
          db        00,00,00,00,00,00,00,00       ;10: Ctrl-Zeichen
          db        00,00,00,00,00,00,00,00       ;18: Ctrl-Zeichen
;
          db        20h,00,00,01,00,00,00,00         ;20: " " = 20, "#" = 01
          db        00,00,00,00,02,00h,00,00         ;28: "," = 02, "-" = 00
          db        30h,31h,32h,33h,34h,35h,36h,37h  ;30: 0-7
          db        38h,39h,00,00,00,04,00,00        ;38: 8-9, "=" = 04
;
; Groβbuchstaben zulassen wegen "stichwort, Präp." usw.
;
          db        00,65,66,67,68,69,70,71       ;064: "A - G"
          db        72,73,74,75,76,77,78,79       ;072: "H - O"
          db        80,81,82,83,84,85,86,87       ;080: "P - W"
          db        88,89,90,00,00,00,00,00       ;088: "X - Z"
;
; Groβbuchstaben ggf. Nullcodes
;
;         db        00,00,00,00,00,00,00,00       ;40: "A - G"
;         db        00,00,00,00,00,00,00,00       ;48: "H - O"
;         db        00,00,00,00,00,00,00,00       ;50: "P - W"
;         db        00,00,00,00,00,00,00,00       ;58: "X - Z"
;
; Kleinbuchstaben für Sanskrit Standard
;
          db        00h,60h,85h,74h,80h,69h,00h,71h  ;60: "@" + "a - g"
          db        8Fh,62h,76h,6Fh,8Ah,87h,82h,6Bh  ;68: "h - o"
          db        83h,00h,89h,8Eh,7Eh,64h,8Bh,00h  ;70: "p - w"
          db        00h,88h,00h,00h,00h,00h,00h,00h  ;78: "x - z"
;
          db        00,00,00,00,00,00,00,00       ;80: "Çüéâäàåç"
          db        00,00,00,00,00,00,00,00       ;88: "êëèïîìÄÅ"
          db        00,00,00,00,00,00,00,00       ;90: "ÉæÆôöòûù"
          db        00,00,00,00,00,00,00,00       ;98: "ÿÖÜ¢£¥₧ƒ"
;
          db        00,00,00,00,00,00,00,00       ;A0: "áíóúñÑªº"
          db        00,00,00,00,00,00,00,00       ;A8: Grafikzeichen
          db        00,00,00,00,00,00,00,00       ;B0: Grafikzeichen
          db        00,00,00,00,00,00,00,00       ;B8: Grafikzeichen
;
          db        00,00,00,00,00,00,00,00       ;C0: Grafikzeichen
          db        00,00,00,00,00,00,00,00       ;C8: Grafikzeichen
          db        00,00,00,00,00,00,00,00       ;D0: Grafikzeichen
          db        00,00,00,00,00,00,00,00       ;D8: Grafikzeichen
;
          db        00h,00h,61h,63h,65h,66h,67h,68h  ;E0: "αβ" + "āīūṛṝḷ"
          db        73h,78h,7Dh,79h,7Bh,8Ch,8Dh,6Dh  ;E8: "ṅñṇṭḍśṁ"
          db        6Eh,00h,00h,00h,00h,00h,00h,00h  ;F0: "ḥ"
          db        00h,00h,00h,00h,00h,00h,00h,00h  ;F8:
;
          db        "<zzz"                        ;Codetab-Endmarker
```

2. Code Section

```
; CODE C07: AS7SANSCOMP
; =====================
;
; Zweck:   Vergleich von zwei Sanskrit-Strings
; ======
;
; Aufruf:  Vergleichswert% = AS7SANSCOMP%(string1$ BYVAL, string2$ BYVAL)
; =======
;
;              [bp+16] [bp+14] [bp+12] [bp+10] [bp+8] [bp+6]
;              1.      2.      3.      4.      5.     6.
; AS7SANSCOMP%(s1%,    o1%,    l1%,    s2%,    o2%,   l2%)
;
; Danach:
; =======
;
; Vergleichswert% 0: string1$ = string2$
; Vergleichswert% 1: string1$ < string2$
; Vergleichswert% 2: string1$ > string2$
;
                    PUBLIC    AS7SANSCOMP
AS7SANSCOMP         PROC
;
; Register retten und Frame Pointer setzen
; ----------------------------------------
;
          push      bp                  ;1. Register auf Stack
          mov       bp,sp
          push      di                  ;2. Register auf Stack
          push      si                  ;3. Register auf Stack
          push      es                  ;4. Register auf Stack
;
; Sind Assembler-DATA im MEDIUM-Quickbasic-DATA-Segment?
; Wenn ASM-DS <> QB-DS, dann Endlosschleife
;
C07A:     mov       ax, D07G
          cmp       ax, 9907h           ;Signatur 07
          jne       C07A                ;Fehler: Endlosschleife
;
; Erst-Char von Str1 mit Erst-Char von Str2 vorab vergleichen
; -----------------------------------------------------------
;
; Wenn keine Nullstringlängen vorliegen und die beiden ersten Zeichen
; der beiden Strings einen normalen, UNGLEICHEN Nicht-00-Code haben,
; dann kann bereits hier die Compare-Routine durchgeführt werden, z.B.:
; UNGLEICHE Strings: "A-nton" < "B-erta" oder "D-ora" > "A-nton"
; Dies ist auch bei den Doppelkodes kh, gh usw. möglich.
; Bei den Diphthongen ai und au geht dies jedoch nicht!!!
;
C07B:     mov       ax,0                ;Compare-Null
;
          mov       bx, [bp+12]         ;12: Altstring1-Länge
          cmp       bx, ax              ;Altstring1-Länge null?
          je        C07D                ;wenn ja, Normalroutine
          mov       si, [bp+14]         ;14: Altstring1-Offset
          mov       bx, [bp+16]         ;16: Altstring1-Segment
          mov       es, bx
          mov       bl, es:[si]         ;BL: Erst-Altchar von Altstring1
          mov       bh, ah              ;BH löschen
          mov       cl, D07H[bx]        ;CL: Erst-Neuchar von Altstring1
          cmp       cl, al              ;Null-Code
          je        C07D                ;wenn ja, Normalroutine
          cmp       cl, 60h             ;ai-au-Sonderfall (a = 60h)
          je        C07D                ;!!!
;
          mov       bx, [bp+6]          ;6: Altstring2-Länge
          cmp       bx, ax              ;Altstring2-Länge null?
          je        C07D                ;wenn ja, Normalroutine
          mov       si, [bp+8]          ;8:  Altstring2-Offset
          mov       bx, [bp+10]         ;10: Altstring2-Segment
```

```
          mov       es, bx
          mov       bl, es:[si]         ;BL: Erst-Altchar von Altstring2
          mov       bh, ah              ;BH löschen
          mov       ch, D07H[bx]        ;CH: Erst-Neuchar von Altstring2
          cmp       ch, al              ;Null-Code
          je        C07D                ;wenn ja, Normalroutine
          cmp       ch, 60h             ;ai-au-Sonderfall (a = 60h)
          je        C07D                ;!!!
;
          cmp       cl,ch               ;String1-Char CMP String2-Char
          je        C07D                ;Wenn gleich, Normalroutine
          ja        C07C
          jmp       C07R                ;string1 < string2
C07C:     jmp       C07S                ;string1 > string2
;
; Längen, Offset und Segment BY VALUE vom Stack nehmen
; ---------------------------------------------------
;
; altstring1$
; -----------
;
C07D:     mov       ax, [bp+12]         ;12: Altstring1-Länge
          cmp       ax, D07A            ;cmp len mit maxlen
          jna       C07E                ;okay, da len <= maxlen
          mov       ax, D07A            ;nein, auf maxlen kürzen
C07E:     mov       dx, ax              ;dx: Altstring1-Länge
          mov       si, [bp+14]         ;si: Altstring1-Offset
          mov       bx, [bp+16]         ;es: Altstring1-Segment
          mov       es, bx
          mov       di, OFFSET D07E     ;di: Neustring1-Adresse
;
; Warnung: CALL muß NEAR sein, sonst stürzt Programm ab!!!
;
          call      NEAR PTR C07G
;
; altstring2$
; -----------
;
          mov       ax, [bp+6]          ;12: Altstring2-Länge
          cmp       ax, D07A            ;cmp len mit maxlen
          jna       C07F                ;okay, da len <= maxlen
          mov       ax, D07A            ;nein, auf maxlen kürzen
C07F:     mov       dx, ax              ;dx: Altstring2-Länge
          mov       si, [bp+8]          ;si: Altstring2-Offset
          mov       bx, [bp+10]         ;es: Altstring2-Segment
          mov       es, bx
          mov       di, OFFSET D07F     ;di: Neustring2-Adresse
;
; Warnung: CALL muß NEAR sein, sonst stürzt Programm ab!!!
;
          call      NEAR PTR C07G
          jmp       C07N
;
; Umkodierung der Altstrings in Neustrings
; ----------------------------------------
;
; bx: Codetabellen-Adresse
; si: Altstring-Adresse    ---> es:[si]
; di: Neustring-Adresse
;
; al: Altchar/Neuchar
; cl: Neustring-Länge
; dl: Altstring-Länge
;
; ah: "0"-Endmarker       Beim Compare wird 1 Takt gespart:
; ch: "|"-Trenner         "CMP al,ch" statt "CMP al,trenner"
;
; dh: Next Char "i" für ai, "u" für au, "h" für kh, gh, ch, jh usw.
;
C07G:     mov       ax,0                ;ah: 00-Endmarker
          mov       cx,0                ;cx: löschen
          mov       ch,D07B             ;ch: "|"-Trenner
          mov       bx,OFFSET D07H      ;bx:  Codetab-Adresse
```

```
          cmp       dl,ah               ;Altstring-Länge = 0?
          jne       C07H                ;nein, weiter
          jmp       C07M                ;wenn ja, sofort Ende
;
; Schleife: Adresse zeigt auf momentanes Altzeichen
; -------------------------------------------------
;
C07H:     mov       al, es:[si]         ;Altchar laden
          cmp       al,ch               ;"|"-Trenner?
          je        C07M                ;wenn ja, sofort Ende
          cmp       al,D07C             ;"\"-Flipper
          je        C07L                ;wenn ja, Flipper-Routine
;
          xlat                          ;Vorab auf Nullkode prüfen
          cmp       al,ah               ;Nullkode?
          je        C07J                ;ja, dann sofort weiter
          mov       al, es:[si]         ;Altchar erneut laden
;
; Doppelkode "ai, au" oder "kh, gh, ch, jh, th, dh" usw.?
;
          cmp       al, "a"             ;"a" vorweg prüfen
          je        C07H1
          call      NEAR PTR C07A1      ;"kgcjṭḍtdpb..."-Doppelkode?
          jnc       C07H5               ;nein, dann normal behandeln
;
C07H1:    cmp       dl, 1               ;Noch mindestens 2 Char?
          jna       C07H5               ;nein, dann normal behandeln
;
; Nächstes Zeichen für Doppelkode holen
;
          dec       dl                  ;Altstring-Länge   vermindern
          inc       si                  ;Altstring-Adresse erhöhen
          mov       dh, es:[si]         ;Next Char holen
;
; Diphthonge ai, au?
;
          cmp       al, "a"             ;a-u oder a-u?
          jne       C07H3               ;nein, dann vielleicht x-h
;
          cmp       dh, "i"
          je        C07H2
          cmp       dh, "u"
          jne       C07H4               ;weder ai noch au
          mov       al, 06Ch            ;au
          jmp       C07I
C07H2:    mov       al, 06Ah            ;ai
          jmp       C07I
;
; Aspiraten kh, gh, ch, jh, ṭh, ḍh, th, dh, ph, bh?
;
C07H3:    cmp       dl, "h"             ;Ist nächster Char "h"?
          jne       C07H4               ;Nein, dann normal weiter
;
; Der Aspirat-Kode ist immer um 1 größer als der Nicht-Aspirat-Kode!
;
          xlat                          ;al -> Nicht-Aspirat-Kode, z.B. k
          inc       al                  ;al -> jetzt Aspirat-Kode, z.b. kh
          jmp       C07I
;
; Doppelkode nicht gefunden. Next-Char-Pointer wieder zurücksetzen!
;
C07H4:    inc       dl                  ;Altstring-Länge   normalisieren
          dec       si                  ;Altstring-Adresse normalisieren
;
; Altchar al ergibt xlat-Offset in bx-Codetabelle
;
C07H5:    xlat                          ;al enthält Code nach xlat
          cmp       al,ah               ;ah = al = 0-Code?
          je        C07J                ;wenn ja, Code ignorieren
;
; Neuzeichen speichern und Adresse von Neustring erhöhen
;
C07I:     mov       [di],al             ;wenn nein, Code speichern
```

```
          inc       di                  ;Neustring-Adresse erhöhen
          inc       cl                  ;Neustring-Länge   erhöhen
;
; Wenn Gleichheitszeichen ("=" 04), dann Sprung zu Dudensort
; ==========================================================
;
          cmp       al, 4               ;Diese 2 Zeilen entfernen,
          je        C07B0               ;damit AS7SANS selbständig bleibt
;
; Adresse von Altstring erhöhen
;
C07J:     inc       si                  ;Altstring-Adresse erhöhen
          dec       dl                  ;Altstring-Länge   vermindern
          je        C07M                ;wenn Länge  = 0, dann Exit
          jmp       C07H                ;Weiter mit Programmschleife
;
; Flipper-Spezialroutine
; ----------------------
; Midstring zwischen "\...\" wird ignoriert
; Daher Altstring bis zum nächsten "\" absuchen
;
; Achtung: Wenn zweiter Flipper "\" in Sortdatei vergessen wird,
;          so hat der Flipper dieselbe Wirkung wie der Trenner "|"
;
C07L:     inc       si                  ;Altstring-Adresse erhöhen
          dec       dl                  ;Altstring-Länge vermindern
          je        C07M                ;wenn Länge = 0, dann Exit
          mov       al, es:[si]         ;Altchar
          cmp       al,D07C             ;"\"-Flipper
          jne       C07L                ;wenn nein, nächster Altchar
          je        C07J                ;wenn ja, normal weiter
;
; 0-Endmarker an Neustring anhängen und Ende
;
C07M:     mov       [di],ah             ;ah = 0 = 0-Endmarker
;
; Warnung: RET muß NEAR sein = RETN, sonst stürzt Programm ab!!!
;
          retn
;
; Ab jetzt nach Duden Sortieren
; =============================
;
C07B0:    inc       si                  ;Altstring-Adresse erhöhen
          dec       dl                  ;Altstring-Länge   vermindern
          je        C07M                ;wenn Länge  = 0, dann Exit
          mov       dh,"ß"              ;dh jetzt Eszett
          mov       bx, OFFSET D01H     ;Duden-Tabelle!!!
;
; Adresse zeigt auf momentanes Altzeichen
;
C07B1:    mov       al, es:[si]         ;Altchar laden
          cmp       al,ch               ;Altchar = "|"-Trenner?
          je        C07B6               ;wenn ja, sofort Ende
          cmp       al,dh               ;Altchar = "ß"-Eszett?
          je        C07B4               ;wenn ja, Eszett-Routine
          cmp       al,D07C             ;"\"-Flipper
          je        C07B5               ;wenn ja, Flipper-Routine
;
; Altchar al ergibt xlat-Offset in bx-Codetabelle
;
          xlat                          ;al enthält Code nach xlat
          cmp       al,ah               ;ah = al = 0-Code?
          je        C07B3               ;wenn ja, Code ignorieren
;
; Neuzeichen speichern und Adresse von Neustring erhöhen
;
C07B2:    mov       [di],al             ;wenn nein, Code speichern
          inc       di                  ;Neustring-Adresse erhöhen
          inc       cl                  ;Neustring-Länge   erhöhen
          cmp       cl,D07A             ;cl >= maxlen? (wegen Eszett)
          jnb       C07B6               ;ja, sofort Ende
;
```

```
; Adresse von Altstring erhöhen
;
C07B3:    inc       si                  ;Altstring-Adresse erhöhen
          dec       dl                  ;Altstring-Länge   vermindern
          jne       C07B1               ;wenn Länge <> 0, dann Loop
          je        C07B6               ;wenn Länge  = 0, dann Exit
;
; Eszett-Spezialroutine
; ---------------------
; "β" wird "SS", sofern cl < maxlen
; "β" wird "S",  sofern cl = maxlen
;
C07B4:    mov       al,"S"              ;"SS" für Eszett
          mov       [di],al             ;Erstes "S" speichern
          inc       di                  ;Neustring-Adresse erhöhen
          inc       cl                  ;Neustring-Länge   erhöhen
          cmp       cl,D07A             ;cl < maxlen?
          jb        C07B2               ;Ja:   Zweites "S" speichern
          jnb       C07B6               ;Nein: Exit + Reststring abhacken
;
; Flipper-Spezialroutine
; ----------------------
; Midstring zwischen "\...\" wird ignoriert
; Daher Altstring bis zum nächsten "\" absuchen
;
; Achtung: Wenn zweiter Flipper "\" in Sortdatei vergessen wird,
;          so hat der Flipper dieselbe Wirkung wie der Trenner "|"
;
C07B5:    inc       si                  ;Altstring-Adresse erhöhen
          dec       dl                  ;Altstring-Länge vermindern
          je        C07B6               ;wenn Länge = 0, dann Exit
          mov       al, es:[si]         ;Altchar
          cmp       al,D07C             ;"\"-Flipper
          jne       C07B5               ;wenn nein, nächster Altchar
          je        C07B3               ;wenn ja, normal weiter
;
; 0-Endmarker an Neustring anhängen und Ende
;
C07B6:     jmp      C07M                ;zurück zu ALTEM Exit!!!
;
; Liegt "kgcjṭdpb" (kh, gh, ch, jh, ṭh, ḍh, th, dh, ph, bh) vor?
; ==============================================================
;
; Carry set = gefunden!
;
C07A1:    cmp       al, 06Bh            ;k
          je        C07A2
          cmp       al, 067h            ;g
          je        C07A2
          cmp       al, 063h            ;c
          je        C07A2
          cmp       al, 06Ah            ;j
          je        C07A2
          cmp       al, 0EBh            ;ṭ
          je        C07A2
          cmp       al, 0ECh            ;ḍ
          je        C07A2
          cmp       al, 074h            ;t
          je        C07A2
          cmp       al, 064h            ;d
          je        C07A2
          cmp       al, 070h            ;p
          je        C07A2
          cmp       al, 062h            ;b
          je        C07A2
          clc                           ;nicht gefunden
          jnc       C07A3
C07A2:    stc                           ;gefunden!
C07A3:    retn                          ;ret muβ NEAR sein
;
; Neustrings neustr1 und neustr2 vergleichen
; ------------------------------------------
;
```

```
; si: neustr1-Adresse
; di: neustr2-Adresse
; al: neustr1-char
; dl: neustr2-char
; ah: 0 für CMP
;
C07N:     mov       si, OFFSET D07E
          mov       di, OFFSET D07F
          mov       ah, 0               ;ah = 0 für CMP löschen
;
; Loop
;
C07O:     mov       al, [si]            ;neustr1-char
          mov       dl, [di]            ;neustr2-char
;
; Endmarke bei einem der beiden Strings erreicht?
;
          cmp       al, ah              ;neustr1-Ende erreicht?
          je        C07Q                ;ja!     Wenn nein:
          cmp       dl, ah              ;neustr2-Ende erreicht?
          je        C07S                ;ja, also neustr1 > neustr2
;
; Strings ab Char-Position ungleich?
;
          cmp       al, dl              ;neustr1 ? neustr2
          jc        C07R                ;neustr1 < neustr2
          jne       C07S                ;neustr1 > neustr2
;
; Wenn bei Char-Position gleich, dann Zähler erhöhen
;
          inc       si                  ;neustr1
          inc       di                  ;neustr2
          jne       C07O                ;immer, da maxlen < 255!
;
C07P:     stc                           ;Programmfehler!
          jc        C07P
;
; neustr1-Endmarker erreicht
;
C07Q:     cmp       dl, ah              ;neustr2-Ende ebenfalls erreicht?
          je        C07T                ;ja, also beide Nullstrings
          jne       C07R                ;nein, also neustr1 < neustr2
;
; AX-Wert bei Exit
; ----------------
;
; ax: 0: neustr1 = neustr2
; ax: 1: neustr1 < neustr2
; ax: 2: neustr1 > neustr2
;
; ax <  2 bedeutet neustr1 <= neustr2
; ax >  0 bedeutet neustr1 <> neustr2
; ax <> 1 bedeutet neustr1 >= neustr2
;
C07R:     mov       ax, 1               ;1: neustr1 < neustr2
          jmp       SHORT C07U
C07S:     mov       ax, 2               ;2: neustr1 > neustr2
          jmp       SHORT C07U
C07T:     mov       ax, 0               ;0: neustr1 = neustr2
;
; Endsequenz
; ----------
;
; Gerettete Register in umgekehrter Reihenfolge von Stack holen
;
C07U:     pop       es                  ;4. Register vom Stack
          pop       si                  ;3. Register vom Stack
          pop       di                  ;2. Register vom Stack
          pop       bp                  ;1. Register vom Stack
          ret       12                  ;6  = 2 x 3 Referenzparameter
;
AS7SANSCOMP   ENDP
```