```
% ---------------------------------------
% |                                     |
% | Testat II in DVS IF2D               |
% | im Sommersemester 2001              |
% |                                     |
% | von:                                |
% | René Valdéz                         |
% | Robert Stoiber                      |
% | Dominik Wallisch                    |
% | Sebastian Schröder                  |
% |                                     |
% ---------------------------------------

% Programmrahmen f"ur einfache MMIX-Programme.
% ein Aufruf von
% TRIP DrZahl,$n,0
% druckt den Wert aus Register n als vorzeichenbehaftete
% ganze Zahl aus.
% Ferner:
% TRIP LsZahl,$n,0
% liest eine vorzeichenbehaftete ganze Zahl in Reg. n ein.
% In beiden F"allen wird der Inhalt von Register 255 ver"andert!

DrZahl  IS      0
LsZahl  IS      1
        LOC     Data_Segment
        GREG    @
------------------------------------------------------------------------
% AB HIER k"onnen eigene Daten erg"anzt werden
        GREG    @
zweibyte BYTE   0,0

        GREG    @
Buffer  BYTE    0
InSize  IS      4000
        LOC     Buffer+InSize
        GREG    @
Arg     OCTA    Buffer,InSize

        GREG    @
Asize   OCTA    15000
Array   GREG    #3000000000000000
Pointer GREG    0
Heap    GREG    #4000000000000000
Free    GREG    0
Stack   GREG    #2000000000000000
SP      GREG    0

        GREG    @
A1      BYTE    "Hashkey generieren (Testat I, DVS)",#d,#a,#0
A2      BYTE    "---------------------------------",#d,#a,#0
        GREG    @
A3      BYTE    "Im Buffer:             ",#0
A4      BYTE    "gefundenes Wort:       ",#0
A5      BYTE    "nach Konvertierung:    ",#0
A6      BYTE    "erechneter Hashkey:    ",#0
        GREG    @
A7      BYTE    #d,#a,#0
A8      BYTE    "gefundene Woerter              ",#0
A9      BYTE    "-----------------              ",#0
A10     BYTE    "Anzahl der Woerter     ",#0
A11     BYTE    "------------------     ",#0
        GREG    @
A12     BYTE    " ............................. ",#0
A13     BYTE    ".",#0
A14     BYTE    ". ",#0
A15     BYTE    "C",#0
A16     BYTE    "Anzahl Kollisionen: ",#0
A17     BYTE    "       . neues Wort angelegt",#d,#a,#0
A18     BYTE    "               C neues Wort nach Kollision angelegt",#d,#a,#0
A19     BYTE    "Legende:",#0

Foffset IS      16
Counter IS      0
Adresse IS      8

PARAM1  IS      $1      Parameter fuer Funktionsaufrufe
PARAM2  IS      $2      Parameter fuer Funktionsaufrufe
PARAM3  IS      $3      Parameter fuer Funktionsaufrufe

INDEX   IS      $10
BUFFER  IS      $11     Zeiger auf Buffer
```

```
ANFANG  IS      $20     Anfang des Wortes finden
ENDE    IS      $21     Ende des Wortes finden
HASH    IS      $22     Hashkey
WOANF   IS      $23     Wortanfang
WOEND   IS      $24     Wortanfang
ANZAHL  IS      $25     Anzahl der Buchstaben
HPOINT  IS      $26     Heap Pointer
AMAX    IS      $27     Adresse des letzten Arraygliedes
LESEN   IS      $28     Datei lesen

KOLL    IS      $29     Laufvariable
KOLLG   IS      $30     Laufvariable

K       IS      $93     Laufvariable
J       IS      $95     Laufvariable
I       IS      $97     Laufvariable
TEMP    IS      $98     Temporaeres Register
TEMP1   IS      $99     Temporaeres Register
TEMP2   IS      $96     Temporaeres Register
TEMP3   IS      $94     Temporaeres Register

% ENDE eigene Daten
        LOC     0
        JMP     IO      Einsprung f"ur {\tt TRIP}: Verzweigung nach IO
        LOC     #100

Handle  IS      3                               %Handle zum Ansprechen der Datei
Handle_Name     BYTE    "mat.txt",0             %Dateiname
Handle_Args     OCTA    Handle_Name,BinaryRead  %Die nächsten beiden Zeilen
Read_Handle     OCTA    Buffer,InSize   %müssen so sein!!

        GREG    @
-----------------------------------------------------------------------
% AB HIER kann das eigene Programm eingesetzt werden

% Hashkey generiert aus dem Wort einen Hashwert
% Funktionsaufrufe:     keine
% bekommt Parameter:    Wortanfang, Wortende Indizes
% gibt zurück:          HASH Wert

hashkey SUB     SP,SP,8                 Stack allozieren
        STO     $0,SP,0                 RTA ablegen
% Rueckadresse auf Stack abgelegt
% ------------------------------------------------------
        SET     HASH,0
0H      LDB     TEMP1,BUFFER,PARAM1     Zeichen lesen und in TEMP speichern
        CMP     TEMP,PARAM1,PARAM2      Vergeich, ob Wort zuende
        BP      TEMP,1F                 Wenn Wort zuende springe zu Ende8

% ------------------------------------------------------
        %SL     HASH,HASH,2             Der Linksshift und speichern in TEMP
        %XOR    HASH,HASH,TEMP1         XOR zwischen TEMP und HASH, speichern in HASH
% ------------------------------------------------------
        MUL     HASH,HASH,13            neuer Hash Algorithmus
        ADD     HASH,HASH,TEMP1
% ------------------------------------------------------
        ADD     PARAM1,PARAM1,1         auf naechsten Buchstaben zeigen
        JMP     0B                      Schleife erneut ausfuehren
1H      SET     PARAM1,HASH
% ------------------------------------------------------
% Rueckadresse von Stack geholt
        LDO     $0,SP,0
        ADD     SP,SP,8
        GO      $0,$0,0
% ------------------------------------------------------

        GREG    @
% Unterprogramm, das prüft, ob das Zeichen ein Buchstabe ist
% Funktionsaufrufe:     keine
% bekommt Parameter:    Indexnummer
% gibt zurück:          Buchstabe an Indexnummer: ja = 1, nein = 0

islett  SUB     SP,SP,8
        STO     $0,SP,0
% Rueckadresse auf Stack abgelegt
% ------------------------------------------------------
        LDB     TEMP1,BUFFER,PARAM1     Laden des Zeichens

        CMP     TEMP,TEMP1,0            Testen, ob NULL
        BZ      TEMP,2F

        CMP     TEMP,TEMP1,'A'          Testen, ob kleiner A
```

```
        BN      TEMP,1F

        CMP     TEMP,TEMP1,'z'          Testen, ob groesser z
        BP      TEMP,1F

        CMP     TEMP,TEMP1,'Z'          Testen, ob kleiner Z
        BNP     TEMP,0F

        CMP     TEMP,TEMP1,'a'          Testen,ob kleiner a
        BN      TEMP,1F

0H      SET     PARAM1,1                Rückgabewert 1
        JMP     3F

1H      SET     PARAM1,0                Rückgabewert 0
        JMP     3F

2H      SET     PARAM1,0                Rückgabewert -1
        SUB     PARAM1,PARAM1,1
% ------------------------------------------------------
% Rueckadresse von Stack geholt
3H      LDO     $0,SP,0
        ADD     SP,SP,8
        GO      $0,$0,0
% ------------------------------------------------------

% Unterprogramm, das nach dem Wortanfang sucht
% Funktionsaufrufe:     islett
% bekommt Parameter:    Anfangsindex
% gibt zurück:          Anfangsindex des Wortes

strbg   SUB     SP,SP,8
        STO     $0,SP,0
% Rueckadresse auf Stack abgelegt
% ------------------------------------------------------
        SET     ANFANG,PARAM1   Anfangsadresse speichern
0H      GO      $0,islett       nach Buchstabe prüfen
        CMP     TEMP,PARAM1,0
        BN      TEMP,2F
        CMP     TEMP,PARAM1,1   Testen, ob gleich 1
        BZ      TEMP,1F         Falls Buchstabe gefunden Ende1
        ADD     ANFANG,ANFANG,1 Weiter im Index von Buffer
        SET     PARAM1,ANFANG   Aktuelle Stelle über PARAM1 übergeben
        JMP     0B              Schleife
1H      SET     PARAM1,ANFANG   Speichern der Anfangsadresse in PARAM1
% ------------------------------------------------------
% Rueckadresse von Stack geholt
2H      LDO     $0,SP,0
        ADD     SP,SP,8
        GO      $0,$0,0
% ------------------------------------------------------

% Unterprogram, das nach dem Wortende sucht
% Funktionsaufrufe:     islett
% bekommt Parameter:    Anfangsindex des Wortes
% gibt zurück:          Endeindex des Wortes

strend  SUB     SP,SP,8
        STO     $0,SP,0
% Rueckadresse auf Stack abgelegt
% ------------------------------------------------------
        SET     ANFANG,PARAM1   Anfangsadresse des Wortes speichern
0H      GO      $0,islett       nach Buchstabe prüfen
        CMP     TEMP,PARAM1,0
        BN      TEMP,2F
        CMP     TEMP,PARAM1,0   Testen ob gleich 0
        BZ      TEMP,1F         Falls kein Buchstabe gefunden Ende3
        ADD     ANFANG,ANFANG,1 Weiter im Index von Buffer
        SET     PARAM1,ANFANG   Aktuelle Stelle über PARAM1 übergeben
        JMP     0B              Schleife
1H      SUB     ANFANG,ANFANG,1 Um eine Stelle zurücksetzen
        SET     PARAM1,ANFANG   Speichern der Endadresse in PARAM1
% ------------------------------------------------------
% Rueckadresse von Stack geholt
2H      LDO     $0,SP,0
        ADD     SP,SP,8
        GO      $0,$0,0
% ------------------------------------------------------
        GREG    @
% Unterprogramm, das Großbuchstaben in Kleinbuchstaben umwandelt
% Funktionsaufrufe:     keine
% bekommt Parameter:    Index des Buchstabens
```

```
% gibt zurück:          void

lowlett SUB     SP,SP,8
        STO     $0,SP,0
% Rueckadresse auf Stack abgelegt
% ------------------------------------------------------
2H      LDB     TEMP1,BUFFER,PARAM1     Laden des Zeichens
        CMP     TEMP,TEMP1,'a'          Testen, ob NULL
        BNN     TEMP,0F
        ADD     TEMP1,TEMP1,32          ASCII um 32 erhöhen
        STB     TEMP1,BUFFER,PARAM1     Im Buffer abspeichern
0H      ADD     PARAM1,PARAM1,1
        CMP     TEMP,PARAM1,PARAM2
        BNP     TEMP,2B
% ------------------------------------------------------
% Rueckadresse von Stack geholt
        LDO     $0,SP,0
        ADD     SP,SP,8
        GO      $0,$0,0
% ------------------------------------------------------

% Unterprogramm, welches nach einem Wort sucht
% Funktionsaufrufe:     strbg, strend
% bekommt Parameter:    Anfangsindex
% gibt zurück:          Anfangsindex und Endeindex des Wortes

strfind LDO     PARAM1,SP,0     Parameter vom Stack laden
        ADD     SP,SP,8
        SUB     SP,SP,8         Stack allozieren
        STO     $0,SP,0         RTA ablegen
% Rueckadresse auf Stack abgelegt
% ------------------------------------------------------
        GO      $0,strbg        nach Stringanfang suchen
        CMP     TEMP,PARAM1,0
        BN      TEMP,0F
        SUB     SP,SP,8         Stack allozieren
        STO     PARAM1,SP,0     Anfang des Wortes auf Stack speichern
        GO      $0,strend       nach Stringende suchen
        CMP     TEMP,PARAM1,0
        BN      TEMP,0F
        SET     PARAM2,PARAM1   Ende des Wortes in PARAM2 speichern
        LDO     PARAM1,SP,0     Anfang des Wortes aus Stack in PARAM1 speichern
        ADD     SP,SP,8
% ------------------------------------------------------
% Rueckadresse von Stack geholt
0H      LDO     $0,SP,0
        ADD     SP,SP,8
        GO      $0,$0,0
% ------------------------------------------------------
        GREG    @

% Unterprogramm, das Wörter auf dem Heap vergleicht
% Funktionsaufrufe:     keine
% bekommt Parameter:    Pointer, WOANF,WOEND
% gibt zurück:          0 = Wörter sind ungleich,
%                       1 = Wörter sind gleich

strcmp  SUB     SP,SP,8
        STO     $0,SP,0
% Rueckadresse auf Stack abgelegt
% ------------------------------------------------------
        SET     TEMP,0                  Temporäre Variablen
        SET     TEMP1,0                 Temporäre Variablen
        SET     TEMP2,0                 Temporäre Variablen
        SET     TEMP3,0                 Temporäre Variablen
        SET     I,0                     Temporäre Variablen
        SET     J,0                     Temporäre Variablen
        SET     K,0

        SUB     K,PARAM3,PARAM2         Länge des Wortes im Buffer ermitteln
        ADD     K,K,1
        SET     I,PARAM2
        SET     J,0
        LDO     TEMP,PARAM1,Adresse     Adresse auf dem Heap laden
4H      LDB     TEMP1,BUFFER,I          einzelnes Zeichen vom Buffer laden
        LDB     TEMP2,TEMP,J            Zeichen vom Heap laden
        CMP     TEMP3,TEMP1,TEMP2
        BNZ     TEMP3,5F
        ADD     I,I,1
        ADD     J,J,1                   Variable erhöhen
        CMP     TEMP3,J,K
        BNZ     TEMP3,4B                Schleife
```

```
        LDB     TEMP2,TEMP,J            Zeichen vom Heap laden
        BZ      TEMP2,6F
5H      SET     PARAM1,0                Parameter setzen
        JMP     7F
6H      SET     PARAM1,1                Parameter setzen

% ---------------------------------------------------------
% Rueckadresse von Stack geholt
7H      LDO     $0,SP,0
        ADD     SP,SP,8
        GO      $0,$0,0
% ---------------------------------------------------------


% Unterprogramm, das ein neues Arrayfield anlegt
% Funktionsaufrufe:     setheap
% bekommt Parameter:    Adresse, WOANF,WOEND
% gibt zurück:          void

addword SUB     SP,SP,8
        STO     $0,SP,0
% Rueckadresse auf Stack abgelegt
% ---------------------------------------------------------
        SET     ANZAHL,1
        STO     ANZAHL,Pointer,Counter  Counter auf 1 initialisieren
        SET     HPOINT,Free             Pointer merken
        SET     I,PARAM2                I = Anfang des Wortes
0H      LDB     TEMP1,BUFFER,I          einzelnes Zeichen vom Buffer laden
        CMP     TEMP,I,PARAM3           bis Ende des Wortes
        BP      TEMP,1F
        ADD     I,I,1                   Variable erhöhen
        STB     TEMP1,Free,0            Zeichen auf Heap speichern
        ADD     Free,Free,1             Pointer weiterschalten
        JMP     0B                      Schleife
1H      SET     TEMP1,0
        STB     TEMP1,Free,0            Nullbyte setzen (field seperator)
        ADD     Free,Free,1             Pointer weiterschalten
        SET     PARAM1,HPOINT           Pointer zurückgeben
        STO     PARAM1,Pointer,Adresse  Speicher vom Heap ablegen
% ---------------------------------------------------------
% Rueckadresse von Stack geholt
        LDO     $0,SP,0
        ADD     SP,SP,8
        GO      $0,$0,0
% ---------------------------------------------------------

% Unterprogramm, das ein Wort zählt
% Funktionsaufrufe:     keine
% bekommt Parameter:    Hashwert,Wortanfang,Wortende
% gibt zurück:          void

setword SUB     SP,SP,8
        STO     $0,SP,0
% Rueckadresse auf Stack abgelegt
% ---------------------------------------------------------

3H      LDO     TEMP,Asize              Arraygroesse laden
        DIV     HASH,PARAM1,TEMP        Hashwert durch Array teilen
        GET     HASH,rR                 n%hash
        MUL     HASH,HASH,Foffset       *Feldgroesse
        SET     Pointer,Array
        ADD     Pointer,Pointer,HASH    Pointer setzen
        LDO     TEMP,Pointer,Counter    lädt die Anzahl der gefunden Worte
        CMP     TEMP1,TEMP,0            Falls Null, soll neu angelegt werden
        BZ      TEMP1,0F

        SET     PARAM1,Pointer          Übergabeparameter
        SET     PARAM2,WOANF            Übergabeparameter
        SET     PARAM3,WOEND            Übergabeparameter
        GO      $0,strcmp               Vergleich von BUFFER <-> Heap
        CMP     TEMP,PARAM1,1           Falls die Wörter gleich sind
        BZ      TEMP,2F
        CMP     TEMP,PARAM1,0
        BZ      TEMP,4F

0H      SET     PARAM1,Pointer          Übergabeparameter
        SET     PARAM2,WOANF            Übergabeparameter
        SET     PARAM3,WOEND            Übergabeparameter
        GO      $0,addword              Wort hinzufügen

        CMP     TEMP,KOLL,0
        BZ      TEMP,5F
```

```
        LDA     $255,A15
        TRAP    0,Fputs,StdOut
        SET     KOLL,0
        ADD     KOLLG,KOLLG,1
        JMP     1F

5H      LDA     $255,A13
        TRAP    0,Fputs,StdOut

1H      CMP     TEMP,Pointer,AMAX
        BNP     TEMP,9F
        SET     AMAX,Pointer            Die höchste Pointeradresse merken
        JMP     9F

2H      LDO     TEMP,Pointer,Counter    Der Counter wird geladen
        ADD     TEMP,TEMP,1             Der Counter wird um 1 erhöht
        STO     TEMP,Pointer,Counter    Der Counter wird gespeichert
        JMP     9F

4H      ADD     HASH,HASH,Foffset       HASH Wert um 1 erhöhen,
        SET     PARAM1,HASH
        SET     KOLL,1
        JMP     3B                      und erneut versuchen
% ------------------------------------------------------------
% Rueckadresse von Stack geholt
9H      LDO     $0,SP,0
        ADD     SP,SP,8
        GO      $0,$0,0
% ------------------------------------------------------------

% Hauptprogramm
% lokale Variablen:     WOANF,WOEND
% Funktionsparameter:   PARAM1,PARAM2, PARAM3
% Buffer:               BUFFER
% Funktionsaufrufe:     bufout, strfind, lowlett, hashkey,
%                       wordout, setword, setword

Main    SET     SP,Stack        Stack initialisieren
        SET     Free,Heap       Heap initialisieren
        SET     Pointer,Array   Array initialisieren
        SET     AMAX,0          Arraymaximum initialisieren
        SET     KOLLG,0

        GETA    $255,Handle_Args
        TRAP    0,Fopen,Handle                  %öffnen der Datei

        LDA     $255,A7         NL
        TRAP    0,Fputs,StdOut
        LDA     $255,A1         Header ausgeben
        TRAP    0,Fputs,StdOut
        LDA     $255,A2         Header ausgeben
        TRAP    0,Fputs,StdOut

        LDA     $255,A19                Header ausgeben
        TRAP    0,Fputs,StdOut
        LDA     $255,A17                Header ausgeben
        TRAP    0,Fputs,StdOut
        LDA     $255,A18                Header ausgeben
        TRAP    0,Fputs,StdOut

        LDA     $255,A7                 NL
        TRAP    0,Fputs,StdOut

9H      GETA    $255,Read_Handle
        TRAP    0,Fgets,Handle
        SET     LESEN,$255                      %Anzahl gelesener Bytes
        LDA     BUFFER,Buffer
        SET     PARAM1,0        Initialisierung
        SET     PARAM2,0        Initialisierung

0H      SUB     SP,SP,8         Stack allozieren
        STO     PARAM1,SP,0     Ende des letzten Wortes uebergeben
% ------------------------------------------------------------
        GO      $0,strfind      nach Wort suchen
% ------------------------------------------------------------
        CMP     TEMP,PARAM1,0
        BN      TEMP,3F         Falls Nullbyte -> Ausgabe
        SET     WOANF,PARAM1    Abspeichern von Wortanfang
        SET     WOEND,PARAM2    Abspeichern von Wortende
        SUB     SP,SP,8         Stack allozieren
        STO     PARAM1,SP,0     Ende des letzten Wortes uebergeben
```

```
        SUB     SP,SP,8         Stack allozieren
        STO     PARAM2,SP,0     Ende des letzten Wortes uebergeben
% ------------------------------------------------------
        GO      $0,lowlett      Zeichen konvertieren
% ------------------------------------------------------
2H      LDO     PARAM2,SP,0     Parameter vom Stack laden
        ADD     SP,SP,8
        LDO     PARAM1,SP,0     Parameter vom Stack laden
        ADD     SP,SP,8
        GO      $0,hashkey      Hashkey errechnen
        SET     PARAM2,WOANF
        SET     PARAM3,WOEND
% ------------------------------------------------------
        GO      $0,setword
% ------------------------------------------------------
        SET     PARAM1,WOANF
        SET     PARAM2,WOEND

        SET     PARAM1,PARAM2   Anfang und Ende vertauschen
        ADD     PARAM1,PARAM1,1 Index um eins nach rechts verschieben

        JMP     0B                      Schleife
3H      BNN     LESEN,9B
        SET     Pointer,Array           Pointer auf Arrayanfang setzen
        SET     I,0                     Laufvariable initialisieren

        LDA     $255,A8                 Ausgabe
        TRAP    0,Fputs,StdOut
        LDA     $255,A10                Ausgabe
        TRAP    0,Fputs,StdOut
        LDA     $255,A7                 NL
        TRAP    0,Fputs,StdOut
        LDA     $255,A9                 Ausgabe
        TRAP    0,Fputs,StdOut
        LDA     $255,A11                Ausgabe
        TRAP    0,Fputs,StdOut
        LDA     $255,A7                 NL
        TRAP    0,Fputs,StdOut

4H      LDO     TEMP,Pointer,Adresse    Adresse vom Heap laden
        LDB     TEMP2,TEMP,0            erstes Zeichen vom Heap laden
        CMP     TEMP3,TEMP2,'a'         Testen auf Buchstabe
        BNN     TEMP3,5F
        CMP     TEMP1,Pointer,AMAX      Vergleich: ist Ende des Arrays erreicht?
        BP      TEMP1,6F
        ADD     Pointer,Pointer,Foffset Pointer im Array erhöhen
        JMP     4B

5H      ADD     I,I,1
        TRIP    DrZahl,I,0

        LDA     $255,A14                Ausgabe
        TRAP    0,Fputs,StdOut

        LDA     $255,TEMP               Adresse auf Inhalt auf dem Heap laden
        TRAP    0,Fputs,StdOut          Wort ausgeben

        LDO     TEMP,Pointer,Counter    Anzahl der Wörter laden
        LDA     $255,A12                Ausgabe
        TRAP    0,Fputs,StdOut
        TRIP    DrZahl,TEMP,0           Anzahl ausgeben
        LDA     $255,A7         NL      NL
        TRAP    0,Fputs,StdOut

        ADD     Pointer,Pointer,Foffset Pointer im Array erhöhen
        JMP     4B
6H      LDA     $255,A7                 NL
        TRAP    0,Fputs,StdOut
        LDA     $255,A16                Ausgabe
        TRAP    0,Fputs,StdOut
        TRIP    DrZahl,KOLLG,0
        LDA     $255,A7                 NL
        TRAP    0,Fputs,StdOut
        TRAP    0,Halt,0                AUS IS !


% ENDE des eigenen Programms.
% ------------------------------------------------------
% Ausgeben
        PREFIX  Rahmen:
r       GREG    0               Rest
Vorz    GREG    0               Vorzeichen
```

```
Zahl    GREG    0               zu behandelnde Zahl
Stellen IS      23              max. 23 Stellen
BUFFER  OCTA    0
        LOC     (BUFFER+Stellen+4)&-4
Arg     OCTA    BUFFER,Stellen

:IO     GET     $255,:rX X der {\tt TRIP}-instruction extrahieren
        SETML   r,1
        AND     r,r,$255
        BNZ     r,LsZahl

        GET     Zahl,:rY
        GETA    $255,BUFFER+Stellen+1
        SETH    Vorz,#8000
        AND     Vorz,Vorz,Zahl
        PBZ     Vorz,2F
        NOR     Zahl,Zahl,Zahl  2-Kompl. berechnen: invertieren
        INCL    Zahl,1          1 addieren
2H      SUB     $255,$255,1     Zahl ohne VZ ausgeben
        DIVU    Zahl,Zahl,10
        GET     r,:rR
        INCL    r,'0'
        STBU    r,$255,0
        PBNZ    Zahl,2B
        PBZ     Vorz,3F
        SUB     $255,$255,1     neg. VZ schreiben
        SETL    r,'-'
        STBU    r,$255,0
3H      TRAP    0,:Fputs,:StdOut
        RESUME  0

% Einlesen
Digit   GREG    0
Count   GREG    0               Bytes im Eingabepuffer z"ahlen
Str     GREG    0               Adr. Eingegebener String
Num     IS      $255            eing. dez.-Zahl
LsZahl  GETA    $255,Arg                Lesen
        TRAP    0,:Fgets,:StdIn
        GETA    Str,BUFFER
        SET     Count,0                 Vorbelegen
        SET     Num,0
        LDB     Digit,Str,Count         Vorzeichen pr"ufen
        CMP     Vorz,Digit,'-'
        PBNZ    Vorz,1F
        INCL    Count,1
1H      LDB     Digit,Str,Count
        CMP     r,Digit,'0'             Hier kann die Basis
        BN      r,2F                    ver"andert werden
        CMP     r,Digit,'9'
        BP      r,2F
        MUL     Num,Num,10              Mult. mit Basis
        SUB     Digit,Digit,'0'
        ADD     Num,Num,Digit
        INCL    Count,1
        JMP     1B
2H      PBNZ    Vorz,3F
        NOR     Vorz,Vorz,Vorz          Vorz=-1
        MUL     Num,Num,Vorz
3H      SETML   Vorz,#2100  gelesenen Wert bereit stellen:
        INCL    Vorz,#FF00  {\tt ADDI} {\bf \$0,\$255,\$0} nach Vorz
        GET     Zahl,:rX   Zahl$\leftarrow$Zielregister
        SET     Count,#FF00
        AND     Zahl,Zahl,Count
        SL      Zahl,Zahl,8  Als 1. Operand in {\tt ADDI} einbauen
        OR      Vorz,Vorz,Zahl
        PUT     :rX,Vorz
        RESUME  0
```