```
                TITLE     'COMAL80 encode'
                PageSize   60
                PageWidth 115

                Dseg    $
                NOLIST
                INCLUDE   RETURNS.CMC
                INCLUDE   C80DCL.A86
                LIST
                EJECT
        ;    FUNCTION encode: line_type;

        ; export
                PUBLIC          E_ENCO          ;   FUNCTION encode

        ; import
                Cseg    $
                EXTRN   CASE:NEAR               ;   PROCEDURE case
                EXTRN   F_DTOB:NEAR             ;   PROCEDURE decbin;
                Dseg    $
                EXTRN   MATERR:WORD             ;   mat_error: integer;
                EXTRN   CORE:WORD               ;   core_adr: addr;
                EXTRN   CORE0:WORD              ;
                EXTRN   NAMEBASE:WORD           ;   name_base: addr;
                EXTRN   STACKTOP:WORD           ;   stacktop: addr;
                EXTRN   DATATOP:WORD            ;   data_top: addr;
                EXTRN   EXCEPT:WORD             ;   exception_code: integer;
                EXTRN   SPMODIF:WORD            ;   spec.modif: integer;
                EXTRN   SPOPR:WORD              ;   spec.opr: integer;
                EXTRN   SP1MODIF:WORD           ;   spec1.modif: integer;
                EXTRN   SP1OPR:WORD             ;   spec1.opr: integer;
                EXTRN   V_OUT:WORD              ;   out: addr;
                EXTRN   ERRPTR:WORD             ;
                EXTRN   IOBUF:WORD              ;   iobuf: addr;
                EXTRN   STACKBASE:WORD          ;   stack_base: addr;
                EXTRN   CPALLOWD:WORD           ;  con_or_proc_allowed:integer;
                EXTRN   ENCODE:WORD             ;   encode: byte;

                Extrn   T_019:WORD,SIZE19:Word  ; (* boolean real ops. *)
                EXTRN   T_021:WORD,SIZE21:Word  ;
                Extrn   T_023:WORD              ; (* Key index table *)
                Extrn   Att_Table:Word

;(* For Nicer Listing and Look:
;               KeyTab, DelimTab, T_Prio, T_ErrTxt , TailTxt , Att_Table
;               Shoud be Abs, but linker fails
;*)
                Extrn   KEYTAB:WORD,DELIMTAB:WORD
                EXTRN   T_PRIO:WORD,T_ERRTXT:WORD,TAILTXT:WORD


;       Create acces to low byte of word's
;
B_EXCEPT        EQU     BYTE PTR ENCODE
B_ENCODE        EQU     BYTE PTR ENCODE
B_MATERR        EQU     BYTE PTR ENCODE

                EJECT
        ; CONST
        ; (*operand types *)
REALTYP         EQU     1       ;   real_type=  1;
STRNGTYP        EQU     2       ;   string_type=2;
BOOLTYP         EQU     3       ;   bool_type=  3;
```

```
        ; CONST


S_FALSE         EQU     0       ;
S_TRUE          EQU     0FFH    ;
MAXDELI         EQU     84      ;   maxdelim = 84;

        ; VAR
SNAME           RS      MAXNAME+1;   name: array(0..maxname+1) of byte;
RECURS          DB      25      ;   recursion: byte;
STACKPTR        DW      0       ;   stack_ptr: addr;
PROGPTR         DW      0       ;   prog_ptr: addr;
ICHAR           DB      20      ;   ichar: byte;
SYMBOL          DW      0       ;   symbol: integer;
SYMVAL          DW      0       ;   sym_value: integer;
OSTKBASE        DW      0       ;   old_stackbase: addr;
DIMNO           DW      0       ;   dim_no,
REFFND          DW      0       ;   ref_found,
ARGNO           DW      0       ;   arg_no,
TYP             DW      0       ;   typ: integer;
SIZE            DW      0       ;   size,
LSTN            DW      0       ;   last: integer;
LNOFND          DW      0       ;   lineno_found : boolean;
REMBUF          DB      0       ;   rembuf: byte;
IOPTR           DW      0       ;   ioptr: addr;
SPSAVE          DW      0       ;   sp_save: addr;
;
;       equ low byte of word
;
B_LNOFND        EQU     BYTE PTR LNOFND
B_SYMVAL        EQU     BYTE PTR SYMVAL
B_SYMBOL        EQU     BYTE PTR SYMBOL

                EJECT
                Cseg    $
        ;
        ; FUNCTION encode: line_type;
        ;
E_ENCO: MOV     SPSAVE,SP       ;BEGIN
        MOV     AL,25           ;
        MOV     RECURS,AL       ;  recursion:=max_recursion;
        MOV     BX,STACKBASE    ;
        MOV     OSTKBASE,BX     ;
        MOV     BX,STACKTOP     ;  stack_ptr:= stack_top;
        INC     BX              ;
        MOV     STACKPTR,BX     ;
        MOV     BX,CORE         ;  prog_ptr:= 1; ioptr:=2;
        INC     BX              ;
        INC     BX              ;
        MOV     PROGPTR,BX      ;
        MOV     BX,OFFSET IOBUF ;
        MOV     AL,[BX]         ;
        MOV     ICHAR,AL        ;  ichar:= iobuf(1);
        INC     BX              ;
        MOV     IOPTR,BX        ;
        MOV     AL,160          ;
        MOV     REMBUF,AL       ;
        CALL    CHKLNO          ;  check_lineno;
        CALL    SCAN            ;  scan;
        MOV     BX,OFFSET L00140;
        PUSH    BX              ;
        MOV     BX,OFFSET L00130;
        MOV     CX,OFFSET L00132 - OFFSET L00130;
        CALL    CASE            ;
        DW      ERRSYN          ; OTHERWISE syntax error
        DW      L00110          ; identyfier
```

```
        DW      L00100          ; string var
        DW      L00090          ; end_of_line
        DW      L00080          ; type 2 key word
        DW      L00070          ; type 1 key word
        DW      L00010          ; Command
L00130  DB      COMMAND         ;
        DB      TYPE1KW         ;
        DB      TYPE2KW         ;
        DB      ENDOFLIN        ;
        DB      STRNGVAR        ;
        DB      IDENTIFY        ;
L00132:                         ;
                EJECT

                                ;  CASE symbol OF
L00010: MOV     AL,KEYBCOMM     ;    command:
        MOV     B_ENCODE,AL     ;      begin
        MOV     BX,SYMVAL       ;        encode:= keyboard_command;
        CALL    STOWHL          ;        store(sym_value);
;       MOV     AL,B_LNOFND     ;        if lineno_found then error(err_syntax)
        OR      AL,AL           ;
        JZ      J0000           ;
        CALL    ERRSYN          ;
J0000:  MOV     AL,B_SYMVAL     ;
        MOV     BX,OFFSET L00065;
        MOV     CX,OFFSET L00020-OFFSET L00065;
        CALL    CASE            ;
                EJECT
        DW      L00060          ; OTHERWISE: system_error
        DW      SCAN            ; size
        DW      SCAN            ; con
        DW      L00045          ; copy
        DW      L00040          ; enter
        DW      L00030          ; load
        DW      L00030          ; save
        DW      L00030          ; run
        DW      L00020          ; dump
        DW      L00020          ; renumber
        DW      L00020          ; list
        DW      L00020          ; edit
        DW      L00020          ; del
        DW      L00020          ; auto
L00065  DB      AUTOCOM         ;
        DB      DELCOM          ;
        DB      EDITCOM         ;
        DB      LISTCOM         ;
        DB      RENUMCOM        ;
        DB      DUMPCOM         ;
        DB      RUNCOM          ;
        DB      SAVECOM         ;
        DB      LOADCOM         ;
        DB      ENTERCOM        ;
        DB      COPYCOM         ;
        DB      CONCOM          ;
        DB      SIZECOM         ;
                EJECT

                                ;        case sym_value of
                                ;          auto_command, del_command,
edit_command,
                                ;          list_command, renum_command,
dump_command:
L00020: CALL    CHKLNO          ;            BEGIN
        CMP     AL,COMMA        ;              check_lineno;
```

```
        CALL    CHKLNO          ;                 get_char;
        JMPS    L00030          ;                 check_line_no;
L00021: MOV     BX,Offset CORE0+6 ;             END
        MOV     CX,[BX]         ;               ELSE
        INC     BX              ;                 store(getcore(3));
                                ;           l1: scan;
        CALL    STOWRD          ;               IF symbol=string_literal THEN
L00030: CALL    SCAN            ;                 scan;
        CMP     AL,STRNGLIT     ;             END;
        JNZ     J0001           ;
        CALL    SCAN            ;
J0001:                          ;           run_command,load_command,
        RET                     ;           save_command:
                                ;             GOTO l1;
                                ;           enter_command:
                                ;             GOTO l2;
                                ;           copy_command:
                                ;           begin
L00045: CALL    L00040          ;             scan;
        CMP     AL,COMMA        ;             if symbol<>string_literal THEN
error(err_syntax);
        MOV     CL,ECOMMA       ;             scan;
        JZ      L00040          ;
        CALL    ERROR           ;             IF symbol<>comma THEN
error(err_comma);
L00040: CALL    SCAN            ;         l2: scan;
        CMP     AL,STRNGLIT     ;             IF symbol<>string_literal THEN
error(err_syntax);
        JZ      J0002           ;
        CALL    ERRSYN          ;             scan;
J0002:  JMP     SCAN            ;           END;
                                ;           con_command,size_command:
                                ;             scan;
L00060: MOV     CL,ESYSTM       ;           otherwise error(err_system);
        JMP     ERROR           ;         END; (*CASE sym_value OF*)
                                ;       END; (* command *)
                EJECT

L00070: MOV     AL,B_LNOFND     ;     type1_keyword:
        OR      AL,AL           ;       BEGIN
        JZ      L00071          ;         IF lineno_found THEN
        MOV     AL,PROGLINE     ;           encode:= program_line
        JMPS    L00072          ;         ELSE
L00071: MOV     AL,LINECOMM     ;           encode:= line_command;
L00072: MOV     B_ENCODE,AL     ;
        MOV     AL,B_SYMVAL     ;         parse_statement(sym_value);
        JMP     PARSTM          ;       END; (* type1_keyword *)
                                ;
                                ;
L00080: MOV     AL,B_LNOFND     ;     type2_keyword:
        OR      AL,AL           ;       BEGIN
        MOV     CL,EILLCM       ;           error(err_illcomm);
        JNZ     J0004           ;
        CALL    ERROR           ;
J0004:                          ;
        MOV     AL,PROGLINE     ;         encode:= program_line;
        MOV     B_ENCODE,AL     ;
        MOV     AL,B_SYMVAL     ;         parse_statement(sym_value);
        JMP     PARSTM          ;       END; (* type2_keyword *)
                                ;
                                ;
                                ;
L00090: MOV     AL,B_LNOFND     ;     end_of_line:
```

```
        JNZ     L00092          ;           encode:= program_line
        MOV     AL,EMPTYLINE    ;         ELSE
L00092: MOV     B_ENCODE,AL     ;           encode:= empty_line;
        MOV     CL,EMPTYSTM     ;         store(empty_stmt);
        JMP     STOWRD          ;       END;
                                ;
                                ;
L00100: MOV     AL,B_LNOFND     ;     string_var:
        OR      AL,AL           ;       BEGIN
        MOV     AL,PROGLINE     ;         IF lineno_found THEN
        JNZ     L00102          ;           encode:= program_line
        MOV     AL,LINECOMM     ;         ELSE
L00102: MOV     B_ENCODE,AL     ;           encode:= line_command;
        MOV     CL,ASSGNSTM     ;         store(assign_stmt);
        CALL    STOWRD          ;
        CALL    VAR             ;         variable(i);
        MOV     AL,STRNGTYP     ;         parse_assignments(string_type);
        JMP     PARASG          ;       END;
                                ;
                                ;     identifier:
L00110: MOV     BX,SYMVAL       ;       BEGIN
        PUSH    BX              ;         i:= sym_value;
        CALL    SCAN            ;         scan;
        CMP     AL,COLON        ;         IF symbol = colon THEN
        JNZ     L00111          ;
        MOV     AL,B_LNOFND     ;         BEGIN  (* label *)
        OR      AL,AL           ;           IF NOT lineno_found THEN
        JNZ     J0005           ;
        CALL    ERRSYN          ;             error(err_syntax);
J0005:  MOV     CL,LABELSTM     ;           store(label_stmt);
        CALL    STOWRD          ;
        MOV     CX,0            ;           store(0);
        CALL    STOWRD          ;
        POP     CX              ;           store(i);
        CALL    STOWRD          ;           encode:= program_line;
        MOV     AL,PROGLINE     ;           scan;
        MOV     B_ENCODE,AL     ;         END
        JMP     SCAN            ;         ELSE
L00111: MOV     AL,B_LNOFND     ;         BEGIN  (* must be assignment *)
        OR      AL,AL           ;           IF lineno_found THEN
        MOV     AL,PROGLINE     ;             encode:= program_line
        JNZ     L00115          ;           ELSE
        MOV     AL,LINECOMM     ;             encode:= line_comand;
L00115: MOV     B_ENCODE,AL     ;
        Pop     Cx              ;
        Push    ProgPtr         ;
        Push    Cx              ;           save:=progptr;
        MOV     CX,ASSGNSTM     ;           store(assign_stmt);
        CALL    STOWRD          ;           scan_subscript_or_parms(typ);
        Call    Scan_Sub_Or_Parm;
        POP     CX              ;           store(i);
        Push    Ax              ;
        CALL    STOWRD          ;
        Pop     Ax              ;
        Pop     Bx              ;           if symbol=assign and typ=false then
        Cmp     Ax,S_False*256+Assign;
        Jne     L00117          ;             parse_assignments(real_type);
        Mov     Al,RealTyp      ;           else
        Jmp     ParAsg          ;             core(save):=Call_Stm;
L00117: Mov     Byte Ptr[Bx],Call_Stm;    END;
        Mov     Ch,ExecSym      ;
        Jmp     StBEox          ;       END;
                                ;     OTHERWISE error(err_syntax);
```

```
                                    ;  END; (* case symbol of *)

                EJECT

L00140: MOV     AL,B_SYMBOL         ;  IF symbol <> end_of_line THEN
        CMP     AL,ENDOFLIN         ;
        JZ      L00142              ;
        CALL    ERRSYN              ;    error(err_syntax);
                EJECT

                                    ;  (* transfer comment if any *)
L00142: MOV     AL,B_SYMVAL         ;  IF sym_value <> comment_op THEN
        CMP     AL,COMMOP           ;    store(eos)
        MOV     CX,EOS              ;
        JZ      L00145              ;
        CALL    STOWRD              ;  ELSE
        JMPS    L00143              ;  BEGIN
L00145: MOV     CH,1                ;    store(comment_eos);
        CALL    STOWRD              ;
        MOV     AL,ICHAR            ;    WHILE ichar <> 0 DO
L00144: OR      AL,AL               ;    BEGIN
        JZ      L00143              ;      sym_value:=ichar;
        MOV     CL,AL               ;      get_char;
        CALL    STOBYT              ;      store(ichar*256+sym_value);
        CALL    GETCHR              ;      get_char;
        JMPS    L00144              ;    END;
L00143: MOV     CL,AL               ;  END;
        CALL    STOBYT              ;
        MOV     BX,PROGPTR          ;  putcore(2, getcore(2) + (prog_ptr-1)*256);
        MOV     DX,CORE             ;  (* set length *)
        OR      AL,AL               ;
        SUB     BX,DX               ;
        MOV     AL,BL               ;
        SHR     AL,1                ;
        DEC     AL                  ;
        MOV     BL,5                ;
        ADD     BX,DX               ;
        MOV     [BX],AL             ;
        JMP     L00150              ;
L00200: MOV     SP,SPSAVE           ;  IF exception_code <> 0 THEN
        MOV     CX,OSTKBASE         ;  BEGIN
        PUSH    CX                  ;    size:=stackbase-old_stackbase;
        MOV     BX,STACKBASE        ;
        PUSH    BX                  ;    count:=stack_top-stackbase;
        SUB     BX,CX               ;    source:=stackbase;
        MOV     STACKBASE,CX        ;    destination:=old_stackbase;
        MOV     CX,BX               ;    stackbase:=old_stackbase;
        MOV     BX,STACKTOP         ;
        PUSH    BX                  ;    stack_top:=stack_top-size;
        SUB     BX,CX               ;    movecore(source,destination,count);
        MOV     STACKTOP,BX         ;
        POP     BX                  ;
        POP     DX                  ;
        SUB     BX,DX               ;
        MOV     CX,BX               ;
        POP     BX                  ;
        XCHG    DX,BX               ;
        INC     CX                  ;
        INC     CX                  ;
        CLD
        MOV     SI,BX
        MOV     DI,DX
        Flush
```

```
        REP     MOVSB
        MOV     DX,DI
        MOV     BX,SI               ;
        MOV     AL,LININERR         ;     encode:= line_in_error;
        MOV     B_ENCODE,AL         ;   END;
                EJECT
L00150: MOV     CX,OSTKBASE         ;
        MOV     BX,STACKBASE        ;   IF stack_base <> old_stackbase THEN
        OR      AL,AL               ;     con_or_proc_allowed:= 0;
        SUB     BX,CX               ;
        JZ      RET_010             ;
        MOV     BX,0                ;
        MOV     CPALLOWD,BX         ;
RET_010:RET                         ;END; (* FUNCTION encode *)

                EJECT
                INCLUDE         ENCODE1.A86
                INCLUDE         ENCODE2.A86
                INCLUDE         ENCODE3.A86
                INCLUDE         ENCODE4.A86
        END
```

```
        EJECT
        ;       ENCODE1: utility procedures

        ; procedure adjust_positive(x: integer);
        ;       CALL    RETURN
        ; BX    x       x+y
        ;
ADJP08: ADD     BX,4
ADJP04: ADD     BX,4
        RET                     ;END; (* procedure adjust_positive *)

        ;
        ; PROCEDURE error(err: integer);
        ;
        ;       call    return
        ; CL    error
        ;
ERR0:   MOV     BX,IOPTR        ;
        MOV     ERRPTR,BX       ;
        JMPS    ERROR           ;
ERRSYN: MOV     CL,ESYNTX       ;
ERROR:  MOV     AL,B_EXCEPT     ; BEGIN
        OR      AL,AL           ;   IF exception_code=0 THEN
        RETNZ                   ;   BEGIN
        MOV     B_EXCEPT,CL     ;     exception_code:=err;
        MOV     ICHAR,0         ;     ichar:=0;
        MOV     AL,ENDOFLIN     ;     symbol:=end_of_line;
        MOV     B_SYMBOL,AL     ;
        MOV     BX,ERRPTR       ;   END;
        MOV     DX,OFFSET IOBUF ;
        SUB     BX,DX           ;
        INC     BX              ;
        MOV     ERRPTR,BX       ;
        JMP     L00200          ; END;
                EJECT
        ;
        ; PROCEDURE getchar;
        ;
        ;       call    return
        ; AL    char
        ;
GETCHR: MOV     AL,ICHAR        ; BEGIN
        OR      AL,AL           ;   IF ichar<>0 THEN
        RETZ                    ;   BEGIN
        MOV     BX,IOPTR        ;     ichar:=iobuf(ioptr);
        MOV     AL,[BX]         ;     ioptr:=ioptr+1;
        MOV     ICHAR,AL        ;
        INC     BX              ;   END;
        MOV     IOPTR,BX        ;
        RET                     ; END;
        ;
        ;
        ;
        ; PROCEDURE get_non_blank;
        ;
        ;       call    return
        ; A             char
        ;
        ;
GETNBL: MOV     AL,ICHAR        ; BEGIN
        CMP     AL,' '          ;   WHILE ichar=space DO get_char;
        JNZ     SAVPTR          ;
        CALL    GETCHR          ;
        JMPS    GETNBL          ; END;
                                ;
```

```
                                    ;
SAVPTR: MOV     BX,IOPTR            ;
        MOV     ERRPTR,BX           ;
        RET                         ;
                EJECT
STOEOX: MOV     CH,0                ;
STBEOX: MOV     CL,EOX              ;
        JMPS    STOWRD              ;
        ;
        ; PROCEDURE store_word(val: integer);
        ;
        ;       call    return
        ; CX    val
        ;
STOWHL: MOV     CX,BX               ;
STOWRD: MOV     BL,AL               ;
        MOV     AL,REMBUF           ;BEGIN
        SUB     AL,2                ;
        MOV     REMBUF,AL           ;  IF prog_ptr < cp1 THEN
        MOV     AL,BL               ;
        JC      L04010              ;  BEGIN
        MOV     BX,PROGPTR          ;    putcore(prog_ptr, val);
        MOV     [BX],CX             ;
        INC     BX                  ;
        INC     BX                  ;    prog_ptr:= prog_ptr+1;
        MOV     PROGPTR,BX          ;  END
        RET                         ;  ELSE
L04010: MOV     CL,ELONG            ;    error(err_longline);
        JMP     ERROR               ;
                                    ;END; (* procedure store_word *)

                EJECT

        ;
        ; PROCEDURE store_byte(val: byte);
        ;
        ;       call    return
        ; CL    val
        ;
STOBYT: MOV     BL,AL               ;
        MOV     AL,REMBUF           ;BEGIN
        SUB     AL,1                ;  IF prog_ptr < cp1 THEN
        MOV     REMBUF,AL           ;  BEGIN
        MOV     AL,BL               ;
        JC      L04010              ;    putcore(progptr, val);
        MOV     BX,PROGPTR          ;    prog_ptr:=prog_ptr+1;
        MOV     [BX],CL             ;  END
        INC     BX                  ;  ELSE
        MOV     PROGPTR,BX          ;    error(err_longline);
        RET                         ;END; (* procedure store_byte *)
                EJECT
        ;
        ; PROCEDURE check_lineno;
        ;
        ;       call    return
        ;
KLNO:   XOR     AL,AL               ;BEGIN
        MOV     B_LNOFND,AL         ;  lineno_found:= false;
        CALL    GETNBL              ;  lineno:= 0;
        MOV     BX,0                ;  get_non_blank;
L07010: CMP     AL,48               ;  WHILE chr(ichar) in digits DO
        JC      L07020              ;
        CMP     AL,58               ;
```

```
        JNC     L07020          ;  BEGIN
        XCHG    DX,BX           ;    IF lineno < 1000 THEN lineno*10+ichar-48
        MOV     BX,1000         ;
        OR      AL,AL           ;
        Cmp     BX,DX           ;    ELSE
        MOV     CL,EILLNM       ;      error(err_illnum);
        JA      L07015          ;
        JMP     ERROR           ;
L07015: MOV     BX,DX           ;
        ADD     BX,BX
        ADD     BX,BX
        ADD     BX,DX
        ADD     BX,BX
        SUB     AL,48           ;
        MOV     DL,AL           ;
        MOV     DH,0            ;
        ADD     BX,DX
                EJECT
        PUSH    BX              ;    get_char;
        CALL    GETCHR          ;
        MOV     BX,OFFSET LNOFND;    lineno_found:= true;
        MOV     BYTE PTR [BX],S_TRUE    ;
        POP     BX              ;
        JMPS    L07010          ;  END;
L07020: MOV     AL,B_LNOFND     ;  IF lineno_found THEN
        OR      AL,AL           ;
        JZ      L07050          ;
        MOV     AL,BL           ;    IF lineno = 0 THEN
        OR      AL,BH           ;
        MOV     CL,EILLNM       ;      error(err_illnum);
        JNZ     L07050          ;
        JMP     ERROR           ;
L07050: CALL    STOWHL          ;  store(lineno);
        JMP     GETNBL          ;  get_non_blank;
                                ;END;(*PROCEDURE check_lineno*)
                EJECT
        ;
        ; PROCEDURE simple_statement;
        ;
SIMSTM: MOV     DX,OFFSET CORE0+6;BEGIN
        MOV     BX,PROGPTR      ;
        DEC     BX              ;  (* regret references *)
        DEC     BX              ;
        DEC     BX              ;
        DEC     BX              ;
        MOV     PROGPTR,BX      ;  prog_ptr:=prog_ptr-2;
        OR      AL,AL           ;  movecore(5,3,prog_ptr-3);
        SUB     BX,DX           ;
        MOV     CX,BX           ;
        MOV     BX,OFFSET CORE0+10      ;
        CLD                     ;
        MOV     SI,BX           ;
        MOV     DI,DX           ;
        Flush                   ;
        REP     MOVSB           ;
        MOV     DX,DI           ;
        MOV     BX,SI           ;
        MOV     AL,B_SYMBOL     ;  CASE symbol OF
        CMP     AL,TYPE1KW      ;    type1_keyword, type2_keyword:
        JZ      L16010          ;
        CMP     AL,TYPE2KW      ;      IF sym_value IN simple_stmts THEN
        JNZ     L16015          ;        parse_statement(sym_value)
L16010: MOV     AL,B_SYMVAL     ;      ELSE
```

```
        Xor     Bh,Bh           ;
        Mov     Bl,Al           ;
        Test    Att_Table[Bx],Att_simpleStm
        JNZ     L16012          ;
        JMP     ERRSYN          ;    identifier:
L16012: JMP     PARSTM          ;      BEGIN
L16015: CMP     AL,IDENTIFY     ;
        JnZ     L16019          ;        save:=progptr;
        Push    ProgPtr         ;
        Mov     Cx,AssgnStm     ;        store(assign_stmt);
        Call    StoWrd          ;
        Push    Symval          ;        save2:=symval;
        Call    Scan            ;
        Call    Scan_Sub_Or_Parm;
        Pop     Cx              ;        scan_subscrift_or_params(typ);
        Push    Ax              ;        store(save2);
        Call    StoWrd          ;        if typ = false and symbol=assign THEN
        Pop     Ax              ;           parse_assigments(real_type)
        Pop     Bx              ;        else
        Cmp     Ax,S_False*256+assign;
        Jne     L16017          ;          putcore(call_stm,save);
        Mov     Al,RealTyp      ;      END;
        Jmp     ParAsg          ;
L16017: Mov     Byte ptr[Bx],Call_Stm
        Mov     Ch,ExecSym      ;
        Jmp     StBEox          ;
                                ;    string_var:
                                ;      BEGIN
                                ;        store(assign_stmt);
                                ;        variable(typ);
L16019: CMP     AL,STRNGVAR     ;        parse_assignments(typ);
        JZ      L16020          ;      END;
        JMP     ERRSYN          ;
L16020: MOV     CX,ASSGNSTM     ;    OTHERWISE
        CALL    STOWRD          ;      error(err_syntax);
        CALL    VAR             ;  END; (*CASE symbol OF*)
        JMP     PARASG          ;END;(*PROCEDURE simple_statement*)

                EJECT

        ; PROCEDURE real_list(nr_reals:Integer);
        ;
        ;       call            return
        ; Cx    nr_reals
        ;
Real_List:                      ; BEGIN
        Push    Cx              ;   FOR i:=1 TO nr_reals DO
        Call    ScArt           ;   BEGIN
        Pop     Cx              ;     scan;
        Dec     Cx              ;     arith_expression;
        JCxZ    Real_List_end   ;
        Push    Cx              ;     IF i < nr_Reals THEN
        Mov     Ch,Comma        ;
        Call    StBEox          ;       Store(Eox+256*comma);
        Pop     Cx              ;
        Cmp     Al,Comma        ;     IF (i<nr_reals) AND (symval<>comma) THEN
        Jz      Real_List       ;       error(comma);
        Mov     Cl,EComma       ;   END;
        Jmp     Error           ;
Real_List_End:                  ;   Store(Eox);
        Jmp     StoEox          ; END;

        ; PROCEDURE real_var_list(eox:al,nr_reals:Integer);
```

```
          ;
          ;         call              return
          ; Al      Eox
          ; Cx      nr_reals
          ;
Real_Var_List:                      ; BEGIN
          Push      Ax              ;
          Push      Cx              ;   FOR i:=1 TO nr_reals DO
          Call      Scan            ;   BEGIN
          Cmp       Al,Identify     ;     scan;
          Jne       Real_Var_Err    ;
          Call      Var             ;     if symbol<>identify then error(IllVar);
          Pop       Cx              ;
          Pop       Ax              ;
          Dec       Cx              ;     Var;
          JCxZ      Real_Var_end    ;
          Cmp       Al,S_True       ;     if i<nr_reals then
          Jne       Real_Var_No_Eox ;     begin
          Push      Ax              ;
          Push      Cx              ;       if eox then store_eox(comma);
          Mov       Ch,Comma        ;
          Call      StbEox          ;
          Pop       Cx              ;       IF symbol<>comma THEN
          Pop       Ax              ;         error(comma);
Real_var_No_Eox:                    ;
          Cmp       B_symbol,Comma  ;     End;
          Jz        Real_Var_List   ;
          Mov       Cl,EComma       ;   END;
          Jmp       Error           ;
Real_Var_End:                       ;   if eox then store(eox);
          Cmp       Al,S_True       ;
          RetNz                     ;
          Jmp       StoEox          ; END;
Real_Var_Err:                       ;
          Mov       Cl,EIllTp       ;
          Cmp       Al,StrngVar     ;
          Je        Real_Go_Err     ;
          Mov       Cl,EIllVr       ;
Real_Go_Err:
          Jmp       Error           ;
          ;
          ; PROCEDURE scan_file_id;
          ;
          ;         call    return
          ;
SCNFID:   CALL      SCART           ;BEGIN
          CMP       AL,COMMA        ;  scan;arith_expression;
          JNZ       L17010          ;  IF symbol = comma THEN
          MOV       CX,CHCOMSYM*256+EOX     ; BEGIN
          CALL      STOWRD          ;    store(ch_comma_eox);
          CALL      SCART           ;    scan;arith_expression;
          MOV       CX,RNDMMDF*256+EOX      ;  store(random_eox);
          CALL      STOWRD          ;  END
          JMPS      L17020          ;  ELSE
L17010:   MOV       CX,CHCOLSYM*256+EOX     ;   store(ch_colon_eox);
          CALL      STOWRD          ;
L17020:   MOV       AL,B_SYMBOL     ;  IF symbol <> colon THEN
          CMP       AL,COLON        ;
          MOV       CL,ECOLON       ;    error(err_colon);
          JZ        J1000           ;
          CALL      ERROR           ;
J1000:    JMP       SCAN            ;  scan;
                                    ;END; (* procedure scan_file_id *)
```

```
                EJECT

        ;
        ; PROCEDURE scan_subscripts(opt: integer);
        ;
        ;       call    return
        ; CL    opt
        ;
        ;
SCNSBC: MOV     AL,B_SYMBOL     ;BEGIN
        CMP     AL,LEFTPAR      ;  IF symbol=left_par THEN
        RETNZ                   ;  BEGIN
SCNSB0: MOV     CH,0            ;    arg_no:= 0;
L19000: PUSH    CX              ;    REPEAT
        CALL    SCART           ;      scan;arith_expression;
        POP     CX              ;
        INC     CH              ;
        CMP     AL,COMMA        ;      arg_no:= arg_no + 1;
        JZ      L19000          ;    UNTIL symbol <> comma;
        PUSH    CX              ;
        CMP     AL,RPAR         ;    IF symbol <> right=par THEN
        MOV     CL,ERPAR        ;      error(err_rpar);
        JZ      J1001           ;
        CALL    ERROR           ;
J1001:                          ;    store(opt+256*arg_no);
        POP     CX              ;    scan;
        JMP     STSCAN          ;  END;
                                ;END;(*PROCEDURE scan_subscripts*)
        ;
        ; PROCEDURE scan_dimensions(var arg_no: integer);
        ;
        ;       call    return
        ; CH            arg_no
        ;
SCNDIM: MOV     CH,0            ;BEGIN
L19020: PUSH    CX              ;  arg_no:= 0;
        CALL    SCART           ;
        CMP     AL,COLON        ;  REPEAT
        JNZ     L19030          ;  scan; arith_expression;
        CALL    SCART           ;  IF symbol = colon THEN
        MOV     CX,BNDSOP       ;  BEGIN
        CALL    STOWRD          ;    scan; artih_expression;
L19030: POP     CX              ;    store(bounds_op);
        INC     CH              ;  END;
        CMP     AL,COMMA        ;  arg_no:= arg_no + 1;
        JZ      L19020          ;  UNTIL symbol <> comma;
        CMP     AL,RPAR         ;
        JZ      L19040          ;
        MOV     CL,ERPAR        ;  IF symbol <> right_par THEN
        JMP     ERROR           ;    error(err_rpar);
L19040: PUSH    CX              ;
        CALL    SCAN            ;  scan;
        POP     CX              ;
        RET                     ;END;
                EJECT
        ; PROCEDURE scan_substring;
        ;
        ;       call    return
        ; AL    symbol
        ; CH            arg_no
        ;
SCNSUB: CMP     AL,LEFTPAR      ;    IF symbol = left_par THEN
        RETNZ                   ;    BEGIN
```

```
          CALL    SCART             ;       scan; arith_expression;
          CMP     AL,COLON          ;       IF symbol <> colon THEN
          MOV     CL,ECOLON         ;         error(err_colon);
          JZ      L19045            ;
          JMP     ERROR             ;       scan;
L19045:   CALL    SCAN              ;       arg_no:=1;
          MOV     CX,256+STRSUBOP   ;       IF symbol<>right_par THEN
          CMP     AL,RPAR           ;       BEGIN
          JZ      L19050            ;         arith_expression;
          CALL    ARTEXP            ;         arg_no:=2;
          CMP     AL,RPAR           ;         IF symbol<>right_par THEN
          MOV     CL,ERPAR          ;           error(err_rpar);
          JZ      J1002             ;
          CALL    ERROR             ;       END;
J1002:    MOV     CX,512+STRSUBOP   ;       store(str_sub_op+256*arg_no);
L19050:   CALL    STSCAN            ;       scan;
          RET                       ;     END;
                  EJECT
          ;
          ; PROCEDURE scan_std1;
          ;
SCNSD1:   MOV     BX,SYMVAL         ;BEGIN
          PUSH    BX                ;  opt:=sym_value;
          CALL    SCAN              ;  scan;
          CMP     AL,LEFTPAR        ;  IF symbol<>left_par THEN
          MOV     CL,ELPAR          ;    error(err_lpar);
          JZ      J1003             ;
          CALL    ERROR             ;  scan;
J1003:    CALL    SCAN              ;  IF opt IN (.len_op,ord_op,val_op.) THEN
          POP     BX                ;    string_expression
          PUSH    BX                ;  ELSE
          MOV     AL,BL             ;
          CMP     AL,LENOP          ;    arith_expression;
          JZ      L20000            ;  IF symbol<>right_par THEN
          CMP     AL,ORDOP          ;    error(err_rpar);
          JZ      L20000            ;  store(opt);
          CMP     AL,VALOP          ;  scan;
          JNZ     L20010            ;
L20000:   CALL    STREXP            ;
          JMPS    L20020            ;
L20010:   CALL    ARTEXP            ;
L20020:   CMP     AL,RPAR           ;
          MOV     CL,ERPAR          ;
          JZ      J1004             ;
          CALL    ERROR             ;
J1004:    POP     CX                ;
          JMP     STSCAN            ;END;
                  EJECT
          ;
          ; PROCEDURE scan_std2;
          ;
SCNSD2:   MOV     BX,SYMVAL         ;BEGIN
          PUSH    BX                ;  opt:=sym_value;
          CALL    SCAN              ;  scan;
          CMP     AL,LEFTPAR        ;  IF symbol<>left_par THEN
          MOV     CL,ELPAR          ;    error(err_lpar);
          JZ      J1005             ;
          CALL    ERROR             ;  scan;
J1005:    POP     CX                ;
SCNS02:   PUSH    CX                ;
          CALL    SCART             ;  arith_expression;
          CMP     AL,COMMA          ;  IF symbol<>comma THEN error(err_comma);
          MOV     CL,ECOMMA         ;  scan;
```

```
        JZ      J1006           ;
        CALL    ERROR           ;  arith_expression;
J1006:  CALL    SCART           ;  IF symbol<>right_par THEN error(err_rpar);
        CMP     AL,RPAR         ;  store(opt);
        MOV     CL,ERPAR        ;  scan;
        JZ      J1007           ;
        CALL    ERROR           ;
J1007:  POP     CX              ;
        MOV     CH,2            ;
        JMP     STSCAN          ;END;

                EJECT

        ;
        ; PROCEDURE parse_assignments(first_type: integer);
        ;
        ;       call    return
        ; AL    first_type
        ;
PARASG: MOV     CX, SYMVAL      ;BEGIN
        PUSH    CX              ; REPEAT
        PUSH    AX              ;  gem assign_operator og typ
        MOV     AL,B_SYMBOL     ;  typ:=first_type;
        CMP     AL,ASSIGN       ;
        MOV     CL,EASSGN       ;    IF symbol<>assign THEN
        JZ      L21005          ;
        JMP     ERROR           ;      error(err_assign)
L21005: CALL    SCAN            ;    ELSE BEGIN
        POP     AX              ;      scan;
        POP     CX              ;
        PUSH    AX              ;      IF typ=real_type THEN
        PUSH    CX              ;
        CMP     AL,REALTYP      ;      BEGIN
        JNZ     L21020          ;        arith_expression;
        CALL    ARTEXP          ;        opt:=assign_op (fra stak);
        POP     CX              ;      END
        JMPS    L21030          ;      ELSE
L21020: POP     AX              ;      BEGIN
                                ;       IF assign-operator<>':='
                                ;
        CMP     AL, RLASSOP     ;       THEN error(err_assign)
        JZ      L21025          ;
        MOV     CL, EASSGN      ;
        JMP     ERROR           ;       ELSE
L21025: CALL    STREXP          ;
        MOV     CX,STRASSOP     ;        string_expression;
L21030: MOV     AL,B_SYMBOL     ;        opt:=string_assign_op;
        CMP     AL,SEMICOLN     ;      END;
        JNZ     L21060          ;      IF symbol=semicolon THEN
        MOV     CH,ASSIGN       ;      BEGIN
        POP     AX              ;        store(assign*256+opt);
        CALL    STSCAN          ;        scan;variable(typ);
        CALL    VAR             ;      END ELSE finis:=true;
        JMPS    PARASG          ;  UNTIL finis OR (exception_code<>0);
L21060: POP     AX              ;  store(opt);
        JMP     STOWRD          ;END;(* parse_assignments *)
                EJECT
        ;
        ; PROCEDURE variable(var var_type: integer);
        ;
        ;       call    return
        ; AL            var_type
        ;
```

```
VAR:    MOV     BX,SYMVAL       ;BEGIN
        PUSH    BX              ;  save_id:=sym_value;
        MOV     AL,B_SYMBOL     ;  CASE symbol OF
NoIfList ; screen var
If      0
        CMP     AL,SYSVAR       ;  system_variable:
        JNZ     L22010          ;  BEGIN
        MOV     AL,ICHAR        ;    IF ichar<>dollar THEN
        CMP     AL,DOLLAR       ;      error(err_syntax);
        JZ      J1008           ;
        CALL    ERRSYN          ;    get_char;
J1008:  CALL    GETCHR          ;    scan; store(save_id);
        CALL    SCAN            ;    scan_substring;
        JMPS    L22020          ;    var_type:=string_type;
                                ;  END;
EndIf
IfList
L22010: CMP     AL,STRNGVAR     ;  string_var:
        JNZ     L22030          ;  BEGIN
        CALL    SCAN            ;    scan;
        CMP     AL,LEFTPAR      ;    IF symbol=left_par THEN
        JNZ     L22020          ;    BEGIN
        MOV     BX,IOPTR        ;      (* look ahead *)
        PUSH    BX              ;      save(ioptr);
        MOV     AL,ICHAR        ;      save(ichar);
        PUSH    AX              ;      save(prog_ptr);
        MOV     BX,PROGPTR      ;      scan;
        PUSH    BX              ;      expression;
        CALL    SCAN            ;
        CALL    EXPRSN          ;
        POP     BX              ;      prog_ptr:= unsave;
        MOV     PROGPTR,BX      ;
        POP     AX              ;      ichar:= unsave;
        MOV     ICHAR,AL        ;
        POP     BX              ;      ioptr:= unsave;
        MOV     IOPTR,BX        ;
        MOV     AL,B_SYMBOL     ;      IF symbol <> colon THEN
        CMP     AL,COLON        ;        scan_params
        MOV     AL,LEFTPAR      ;      ELSE symbol:= left_par;
        JZ      L22020          ;
        CALL    SCNPAR          ;
L22020: POP     CX              ;    END;
        PUSH    AX              ;    store(save_id);
        CALL    STOWRD          ;
        POP     AX              ;    scan_substring;
        CALL    SCNSUB          ;
        MOV     AL,STRNGTYP     ;    var_type:=string_type;
        RET                     ;  END;
                EJECT
L22030: CMP     AL,IDENTIFY     ;  identifier:
        MOV     CL,EILLVR       ;  BEGIN
        JZ      L22035          ;
        JMP     ERROR           ;    scan;
L22035: CALL    SCAN            ;    scan_subscripts(arg_op);
        MOV     CL,ARGOP        ;    store(save_id);
        CALL    SCNSBC          ;    var_type:=real_type;
        POP     CX              ;  END;
        CALL    STOWRD          ;
        MOV     AL,REALTYP      ;  OTHERWISE
        RET                     ;    error(err_illvar);
                                ;END;(*PROCEDURE variable*)
                EJECT
        ;
```

```
        ; PROCEDURE push(elem: integer);
        ;
        ;       call    return
        ; AL    elem
        ;
PUSH:   MOV     BX,STACKPTR     ;BEGIN
        INC     BX              ;  IF stack_ptr=data_top THEN
        MOV     DX,DATATOP      ;    error(err_core)
        OR      AL,AL           ;  ELSE
        SUB     BX,DX           ;  BEGIN
        MOV     CL,ECORE        ;    putcore(stack_ptr,elem);
        JNZ     PUSH10          ;
        JMP     ERROR           ;    stack_ptr:=stack_ptr+1;
PUSH10: ADD     BX,DX           ;  END;
        MOV     STACKPTR,BX     ;
        MOV     [BX],AL         ;
        RET                     ;END;
                EJECT

        Dseg    $
String_seen     Rb      1
        Cseg    $

Scan_Sub_Or_Parm:
        Mov     Ah,S_False
        Mov     Al,b_Symbol
        Cmp     Al,LeftPar
        RetNz
        Mov     String_Seen,Ah
        ;
        ; PROCEDURE scan_params;
        ;
SCNPAR: MOV     CX,ARGOP        ;BEGIN
L09000: PUSH    CX              ;  arg_no:=0;
        CALL    SCAN            ;  REPEAT
        CALL    EXPRSN          ;    scan;
        Cmp     Al,StrngTyp     ;    expression(typ);
        Jne     L09002          ;    if typ = string_type then
        Mov     String_Seen,S_True;    string_seen:=true;
L09002: POP     CX              ;
        INC     CH              ;    arg_no:=arg_no+1;
        MOV     AL,B_SYMBOL     ;  UNTIL symbol<>comma;
        CMP     AL,COMMA        ;  IF symbol<>right_par THEN
        JZ      L09000          ;    error(err_rpar);
        CMP     AL,RPAR         ;  store(arg_op+256*arg_no);
        JNZ     L09005          ;
        Call    STSCAN          ;  scan;
        Mov     Ah,String_Seen  ;
        Ret
L09005: MOV     CL,ERPAR        ;
        JMP     ERROR           ;END;(*PROCEDURE scan_params*)
                EJECT
        ;
        ; PROCEDURE arith_expression;
        ;
SCART:  CALL    SCAN            ;
STEXP:  CALL    EXPRSN          ; BEGIN
        CMP     AL,REALTYP      ;   expression(typ);
        JMPS    STREX0          ;   IF typ<>real_type THEN
                                ;     error(err_illtype);
                                ; END;
        ;
        ; procedure string_expression;
```

```
        ;
SCSTR:  CALL    SCAN            ;
STREXP: CALL    EXPRSN          ; BEGIN
        CMP     AL,STRNGTYP     ;  expression(typ);
STREX0: MOV     CL,EILLTP       ;  IF typ<>string_type THEN
        JZ      LDSYMB          ;
        CALL    ERROR           ;    error(err_illtype);
LDSYMB: MOV     AL,B_SYMBOL     ;
        RET                     ; END;
                EJECT
        ;
        ; PROCEDURE scan_number;
        ;
        ;               Math:   IX => SI
        ;                       IY => DI
        ;
SCNNUM: MOV     AL,REMBUF       ;BEGIN
        SUB     AL,10           ;  IF cp1-prog_ptr<6 THEN
        JNC     LXX000          ;    error(err_longline)
        MOV     CL,ELONG        ;  ELSE
        JMP     ERR0            ;  BEGIN
LXX000: MOV     BX,PROGPTR      ;
        INC     BX              ;    ioptr:=ioptr-1;
        INC     BX
        MOV     SI,BX           ;    IF decbin(iobuf,ioptr,
                                ;              prog_ptr+1)<>0 THEN
        MOV     BX,IOPTR        ;      error(err_illconst)
        DEC     BX              ;    ELSE
        XOR     AL,AL           ;    BEGIN
        MOV     B_MATERR,AL     ;
        CALL    F_DTOB          ;      IF short_const THEN
        MOV     IOPTR,BX        ;        store(-(num(0)+#h70*256+
        MOV     AL,B_MATERR     ;                num(7))
        OR      AL,AL           ;      ELSE
        JZ      LXX010          ;      BEGIN
        MOV     CL,EILLCT       ;
                                ;        store(-#h5004);
        JMP     ERR0            ;
LXX010: TEST    BYTE PTR [SI],080H;      prog_ptr:=prog_ptr+4;
        JNZ     LXX030          ;
        MOV     CH,6            ;        rembuf:=rembuf-4;
        MOV     DH,[SI]         ;
        INC     SI              ;      END;
        XOR     AL,AL           ;
LXX020: CMP     AL,[SI]         ;    getchar;
        JNZ     LXX030          ;
        INC     SI              ;
        DEC     CH              ;
        JNZ     LXX020          ;
        MOV     AL,DH           ;
        OR      AL,0F0H         ;
        MOV     CH,AL           ;
        MOV     CL,[SI]         ;
        CALL    STOWRD          ;
        JMPS    LXX040          ;
                EJECT
X030:   MOV     CX,0D008H       ;
        CALL    STOWRD          ;
        MOV     BX,PROGPTR      ;
        CALL    ADJP08          ;
        MOV     PROGPTR,BX      ;
        MOV     AL,REMBUF       ;
        SUB     AL,8            ;
```

```
        MOV     REMBUF,AL       ;
LXX040: JMP     GETCHR          ;END;
        EJECT
        ;
        ; PROCEDURE scan_proc(VAR proc_type: integer);
        ;       call    return
        ; AL            proc_type
        ;
SCPROC: CALL    STOWRD          ;       BEGIN
        CALL    SCAN            ;         store(sym_value);
        MOV     CH,0            ;         scan; arg_no:= 0;
        PUSH    CX              ;         IF symbol = left_par THEN
        CMP     AL,LEFTPAR      ;         BEGIN
        JNZ     L18212          ;
L18222: CALL    SCAN            ;           REPEAT
        MOV     DL,0            ;             scan;
        CMP     AL,REFSYM       ;             ref_found:= symbol = ref_symbol;
        JNZ     L18213          ;             IF ref_found THEN scan;
        CALL    SCAN            ;
        MOV     DL,1            ;
L18213: CMP     AL,IDENTIFY     ;             IF symbol = identifier THEN
        LAHF                    ;
        AND     DL,0FDH         ; (* RES 1,DL *)
        SAHF                    ;               param_specifier.typ:= 0;
        JZ      L18215          ;             ELSE
        EJECT
        CMP     AL,STRNGVAR     ;             IF symbol = string_var THEN
        MOV     CL,EILLVR       ;               param_specifier.typ:= 2;
        JZ      L18214          ;
        JMP     ERROR           ;             ELSE
L18214: LAHF                    ;
        OR      DL,002H         ; (* SET 1,DL *)
        SAHF                    ;               error(err_illvar;
L18215: MOV     DH,0            ;
        PUSH    DX              ;
        CALL    STSYMV          ;             store(-sym_value);scan;
        CMP     AL,LEFTPAR      ;             dim_no:= 0;
        JNZ     L18217          ;             IF symbol = left_par THEN
L18218: POP     DX              ;             BEGIN
        INC     DH              ;               REPEAT
        PUSH    DX              ;                 dim_no:= dim_no + 1;
        CALL    SCAN            ;                 scan;
        CMP     AL,COMMA        ;               UNTIL symbol <> comma;
        JZ      L18218          ;
        CMP     AL,RPAR         ;               IF symbol <> right_par THEN
        MOV     CL,ERPAR        ;                 error(err_rpar);
        JZ      J1011           ;
        CALL    ERROR           ;               scan;
J1011:  CALL    SCAN            ;             END;
        EJECT
L18217: POP     DX              ;
        MOV     AL,DH           ;
        OR      AL,AL           ;
        JZ      L18221          ;             IF dim_no <> 0 THEN
        TEST    DL,001H         ;               IF NOT ref_found THEN
        MOV     CL,EREF         ;                 error(err_ref);
        JNZ     L18221          ;
        CALL    ERROR           ;
L18221:                         ;             store(dim_no*256 +
        MOV     CX,DX           ;                     typ + ref_found);
        CALL    STOWRD          ;
        POP     CX              ;             arg_no:= arg_no + 1;
        INC     CH              ;
```

```
        PUSH    CX                      ;
        MOV     AL,B_SYMBOL             ;           UNTIL symbol <> comma;
        CMP     AL,COMMA                ;
        JZ      L18222                  ;
        CMP     AL,RPAR                 ;           IF symbol <> right_par THEN
        MOV     CL,ERPAR                ;             error(err_rpar);
        JZ      J1013                   ;
        CALL    ERROR                   ;           scan;
J1013:  CALL    SCAN                    ;         END;
                EJECT
L18212: CMP     AL,CLOSYM               ;         IF symbol = closed_symbol THEN
        JNZ     L18224                  ;         BEGIN
        CALL    SCAN                    ;           scan;
        MOV     AL,CLOSYM               ;           proc_modif:=closed_symbol;
        JMPS    L18235                  ;         END ELSE
L18224: CMP     AL,HANSYM               ;           IF symbol = handler_symbol THEN
        JNZ     L18226                  ;           BEGIN
        CALL    SCAN                    ;
        POP     AX                      ;             scan;
        Or      Ah,Ah                   ;             IF arg_no <> 0 THEN
        PUSH    AX                      ;               error(err_syntax);
        JZ      J1014                   ;
        CALL    ERRSYN                  ;
J1014:  MOV     AL,HANDLMDF             ;             proc_modif:=handler_modif;
        JMPS    L18235                  ;           END ELSE
                EJECT
L18226: CMP     AL,EXTNSYM              ;             IF symbol = external_symbol THEN
        JNZ     L18227                  ;             BEGIN
        MOV     AL,EXTNMDF              ;               proc_modif:=external_modif;
        JMPS    L18235                  ;             END ELSE
L18227: XOR     AL,AL                   ;               proc_modif:= 0;
L18235: POP     CX                      ;
        MOV     BX,OFFSET CORE0+8;
        MOV     [BX],AL                 ;         putcore(4, proc_modif + arg_no*256);
        INC     BX                      ;
        MOV     [BX],CH                 ;         store(proc_modif*256 + eox);
        MOV     CH,AL                   ;
        MOV     CL,EOX                  ;         IF proc_modif = external_modif THEN
        CALL    STOWRD                  ;         BEGIN
        CMP     AL,EXTNMDF              ;           scan;
        PUSH    AX                      ;
        JNZ     J1015                   ;           case symbol of
        Call    Scan                    ;             string_litteral:
        Cmp     Al,StrngLit             ;               scan;
        Jne     L18237                  ;
        Call    Scan                    ;
        Jmps    j1015                   ;             string_var:
L18237: Cmp     Al,StrngVar             ;               var(typ);
        Jne     L18236                  ;
        Call    Var                     ;             otherwise
        Jmps    j1015                   ;               error(ill_type);
L18236: Mov     Al,EIllTp               ;           end;
        Call    Error                   ;         END;
J1015:  POP     AX                      ;
        RET                             ;       END;
```

```
        EJECT
```

```
;*****************************************************************************
;
;
;
;                       ENCODE2: PROCEDURE scan
;
;
;*****************************************************************************
        EJECT
        ;
        ; PROCEDURE scan;
        ;       call    return
        ; AL            symbol
        ; CX            sym_value
        ;
ST00SC: MOV     CX,0                    ;
        CALL    STOWRD                  ;
ST0SC:  MOV     CX,0                    ;
STSCAN: CALL    STOWRD                  ;
SCAN:   MOV     Di,OFFSET SNAME         ; BEGIN
        MOV     CX,MAXNAME+1            ;
        CLD                             ;
        Xor     Al,Al                   ;
        REP     Stos    Al              ;
                EJECT
        CALL    GETNBL                  ;
        OR      AL,AL                   ;   get_non_blank;
        JNZ     L06000                  ;   IF ichar = 0 THEN
        MOV     AL,ENDOFLIN             ;   BEGIN
        MOV     CX,0                    ;     symbol:= end_of_line;
        JMP     L06285                  ;     sym_value:= 0;
L06000: MOV     CL,EILLCH               ;   END
        CMP     AL,127                  ;
        JC      L06001                  ;
L_ERR:  JMP     ERROR                   ;
L06001: CMP     AL,32                   ;
        JC      L_ERR                   ;
        CMP     AL,']'                  ;
        JA      J06180                  ;
        CMP     AL,'a'                  ;   ELSE
        JAE     L06010                  ;
        CMP     AL,'['                  ;   CASE chr(ichar) OF
        JA      J06180                  ;
        CMP     AL,'A'                  ;     'A'..'[', 'a'..']':  (* word *)
        JAE     L06030                  ;       ;
J06180: JMP     L06180                  ;     BEGIN
L06010: SUB     AL,32                   ;             (*read name and pack*)
2;      MOV     ICHAR,AL                ;       IF ichar>96 THEN ichar:=ichar-3
L06030: MOV     BX,OFFSET SNAME         ;       i:= 0;
        MOV     CX,0                    ;       REPEAT
L06040: INC     CX                      ;         IF i < maxname THEN i:= i+1
        INC     BX                      ;         ELSE error(err_toolong);
        CMP     CL,MAXNAME              ;
        JBe     L06060                  ;
        MOV     CL,ETOOLG               ;
        CALL    ERRO                    ;
                EJECT
L06060: MOV     AL,ICHAR                ;
```

```
        MOV     [BX],AL                 ;           name(i):= ichar;
        PUSH    BX                      ;(*name*)
        PUSH    CX                      ;(*i*)
        CALL    GETCHR                  ;           get_char;
        POP     CX                      ;
        POP     BX                      ;
        CMP     AL,')'                  ;           IF namechar(ichar) THEN
        JA      L06066                  ;           BEGIN
        CMP     AL,'a'                  ;             name(i):=name(i)+ichar*256;
        JAE     L06065                  ;             getchar;
        CMP     AL,'_'                  ;           END;
        JZ      L06040                  ;
        CMP     AL,']'                  ;
        JA      L06066                  ;
        CMP     AL,'A'                  ;
        JAE     L06040                  ;
        CMP     AL,'9'                  ;
        JA      L06066                  ;
        CMP     AL,'0'                  ;
        JB      L06066                  ;
        JMPS    L06040                  ;
L06065: SUB     AL,32                   ;
        MOV     ICHAR,AL                ;
        JMPS    L06040                  ;       UNTIL NOT namechar(ichar);
                EJECT
L06066: INC     CX                      ;       name(0):=i;(*length of name*)
        MOV     AL,CL                   ;
        MOV     SNAME,AL                ;       IF exception_code=0 THEN
        MOV     BX,OFFSET T_023         ;       BEGIN
        PUSH    CX                      ;
        Sub     Cx,2                    ;
words*) Shl     CL,1                    ;         (*search in table of reserved
        ADD     BX,CX                   ;         found:=false;  size:=i;
;       MOV     DX,2[BX]                ;         last:=key_index_table(size+1)
        MOV     BX,[Bx]                 ;         next:=key_index_table(size);
                EJECT
        Mov     Ax,Cs           ;
        Mov     Es,Ax           ;
L06070: POP     CX              ;(*size*)
                                ;(*next*) WHILE NOT(found) AND (next<last) DO
        CMP     BX,DX           ;           BEGIN
                                ;             cur:=next;
        JAE     L06110          ;             next:=cur+size+1;
        PUSH    CX              ;(*size*)     IF name(1)=keytable(cur+1) THEN
        PUSH    DX              ;(*last*)
        PUSH    BX              ;(*next*)     BEGIN
        MOV     DI,BX           ;
        ADD     DI,3            ;
        MOV     SI,OFFSET SNAME+1;
        Dec     Cx              ;               WHILE found AND (i<size) DO
N                               ;                 IF name(i)<>keytable(cur+1) THE
        CLD                     ;                   found:=false;
        REPZ    CMPSB           ;                 i:=i+1;
        JZ      L06100          ;
        POP     BX              ;(*cur*)        END;
        POP     DX              ;(*last*)     END;
        POP     CX              ;(*size*)
        PUSH    CX              ;(*size*)
        Inc     Bx              ;(* Sym *)
        Inc     Bx              ;(* Typ *)
        ADD     BX,CX           ;(* Len *)END;
        JMPS    L06070          ;
                EJECT
```

```
L06100: POP     BX              ;(*next*) IF found THEN
        POP     DX              ;(*last*) BEGIN
        POP     CX              ;(*size*)   symbol:=left(keytab(cur));
        MOV     CH,0            ;
        MOV     CL,Es:[BX]      ;           sym_value:=right(keytab(cur));
        MOV     AL,Es:1[BX]     ;         END
        Mov     Bx,Ds           ;
        Mov     Es,Bx           ;         ELSE
        JMP     L06285          ;         BEGIN (*lookup name in nametable*)
L06110: Mov     Ax,Ds           ;
        Mov     Es,Ax           ;
        MOV     BX,NAMEBASE     ;           next:=getaddr(name_base)+4;
        ADD     BX,4            ;
        MOV     CH,0            ;
        PUSH    CX              ;(*size*)   WHILE (NOT found) AND
L06120: XCHG    Si,BX           ;           getbyte(next) <> 0 DO
        PUSH    Si              ;(*next*)   BEGIN;
        Mov     Al,[Si]         ;             cur:=next;
        OR      AL,AL           ;             next:=cur+getbyte(cur)+2;
        JZ      L06150          ;             IF name(0)=getbyte(cur) THEN
        MOV     DI,OFFSET SNAME ;             BEGIN
        MOV     CL,AL           ;               found:=true; i:=1;
L06130:                         ;               WHILE found AND (i<size) DO
        REPZ    CMPSB           ;               BEGIN
EN      JZ      L06170          ;                 IF name(i)<>getcore(cur+i) TH
L06140: POP     BX              ;(*next*)           found:=false;
        MOV     CL,[BX]         ;
        ADD     BX,CX           ;                 i:=i+1;
        INC     BX              ;(* Data Ptr *)  END;
        INC     BX              ;               END;
        JMPS    L06120          ;           END;
L06150: POP     DX              ;(*next*)
        POP     CX              ;
        INC     CX              ;           IF NOT found THEN
        INC     CX              ;(*size*)   BEGIN
        PUSH    CX              ;             (*insert name in name table*)
        MOV     BX,STACKPTR     ;             cur:=next;
                EJECT
        ADD     BX,CX           ;
        MOV     CX,DATATOP      ;             size:=size+2;
N       SUB     BX,CX           ;             IF (stack_ptr+size)>=data_top THE
        MOV     CL,ECORE        ;               error(err_core)
        JC      J2000           ;
        CALL    ERROR           ;             ELSE
J2000:  MOV     BX,STACKPTR     ;             BEGIN
                                ;               movecore(cur-2,cur-2+size,
        SUB     BX,DX           ;                        stack_ptr-cur+3);
        INC     BX              ;
        INC     BX              ;               stack_base:=stack_base+size;
        MOV     CX,BX           ;               stack_top:=stack_top+size;
                                ;               stack_ptr:=stack_ptr+size;
        POP     BX              ;(*size*)
        PUSH    BX              ;(*size*)       FOR i:=0 TO name(0) DO
        PUSH    DX              ;
        DEC     DX              ;(*cur*)          putcore(cur+i,name(i));
        DEC     DX              ;             END;
        ADD     BX,DX           ;           END;
        ADD     BX,CX           ;
        XCHG    DX,BX           ;
        ADD     BX,CX           ;
        INC     CX              ;
        STD                     ;
        MOV     SI,BX           ;
```

```
        MOV     DI,DX           ;
        Flush
        REP     MOVSB           ;
        POP     Di              ;(*cur*)
        POP     CX              ;(*size*)
        PUSH    CX              ;
        PUSH    Di              ;(*cur*)
        Add     StackBase,Cx    ;
        Add     STACKTOP,Cx     ;
        Add     STACKPTR,Cx     ;
        MOV     Si,OFFSET SNAME ;
        DEC     CX              ;
        DEC     CX              ;
        CLD                     ;
        Flush
        REP     MOVSB           ;
                EJECT
L06170: POP     BX              ;(*cur*)
        POP     CX              ;(*size*)
        MOV     DX,NAMEBASE     ;
        SUB     BX,DX           ;
        DEC     BX              ;           sym_value:=cur-2-name_base;
        DEC     BX              ;           IF sym_value > 8191 THEN
        TEST    BH,020H         ;             error(err_nameoverflow);
        MOV     CL,ENAMOVF      ;
        JZ      J2001           ;
        CALL    ERROR           ;           symbol:=identifier;
J2001:  MOV     CX,BX           ;           IF ichar=dollar THEN
        OR      CH,080H         ;           BEGIN
                                ;             symbol:=string_var;
        MOV     AL,ICHAR        ;             sym_value:=sym_value+8192;
        CMP     AL,'$'          ;             getchar;
        MOV     AL,IDENTIFY     ;           END;
        JZ      L06179          ;
        JMP     L06285          ;           sym_value:= -sym_value;
L06179: OR      CH,020H         ;         END;
        PUSH    CX              ;       END;
        CALL    GETCHR          ;     END;
        MOV     AL,STRNGVAR     ;
        POP     CX              ;
        JMP     L06285          ;
                EJECT
L06180: CMP     AL,'9'+1        ;     '0'..'9','.': (*number*)
        JNC     L06210          ;     BEGIN
        CMP     AL,'0'          ;
        JNC     L06185          ;       symbol:=number;
        CMP     AL,'.'          ;
        JNZ     L06210          ;
L06185: CALL    SCNNUM          ;       scan_number;
        MOV     AL,NUMBER       ;
        JMP     L06285          ;     END;
                EJECT
L06210: CMP     AL,'"'          ;   '"':   (*string literal*)
        JNZ     L06240          ;   BEGIN
        Push    PROGPTR         ;     next:=prog_ptr;
        Xor     Cx,Cx           ;     i:=0;
        PUSH    CX              ;     prog_ptr:=prog_ptr+2;
        Add     PROGPTR,2*Word  ;     (* room for length *)
        CALL    GETCHR          ;     get_char;
        Xor     Cl,Cl           ;     previchar := 0;
                                ;     REPEAT
L06220: Cmp     Al,0            ;       WHILE ( (ichar()ord("'") OR
        Jz      L06225          ;                 (PreviChar = Ord('"')) ) AND
```

```
        CMP     AL,'"'          ;               (ichar <> 0) DO
        JnZ     L06222          ;
        CMP     Cl,'"'          ;
        JnZ     L06225          ;         BEGIN
L06222: POP     CX              ;
        INC     CX              ;           i:=i+1;
        PUSH    CX              ;
        MOV     CL,AL           ;           storebyte(ichar);
        CALL    STOBYT          ;
        Call    getchr          ;           getchr;
        Xor     Cl,Cl           ;           previchar := 0;
        JMPS    L06220          ;         END;
L06225: Push    Ax              ;
        Call    GetChr          ;         previchar := ichar;
        Pop     Cx              ;         getchr;
        Cmp     Al,'"'          ;
        jz      L06220          ;       UNTIL ichar <> Ord('"')
L06230: CMP     Cl,'"'          ;
        MOV     CL,EQUOTE       ;       IF previchar <> Ord('"') THEN
        JZ      L06235          ;          Error(err_qoute);
        CALL    ERR0            ;
L06235: POP     CX              ;
        TEST    CL,001H         ;       IF i mod 2 = 1 THEN (* Allign word *)
        JZ      L06236          ;
        PUSH    CX              ;
        MOV     CL,0            ;          StoreByte(0);
        CALL    STOBYT          ;
        POP     CX              ;
L06236: MOV     Dx,PROGPTR      ;
        POP     Bx              ;(*next*)
        SUB     Dx,Bx           ;       putcore(next,-(4*4096+prog_ptr-1-next));
        ADD     Dx,0c000h-2     ;
        MOV     [BX],DX         ;       putcore(next+1,i);
        MOV     2[BX],CX        ;
        MOV     AL,STRNGLIT     ;       symbol:=string_literal;
        JMP     L06285          ;     END;
                EJECT
L06240: MOV     AH,AL
        PUSH    AX              ;     OTHERWISE     (*delimiter*)
        CALL    GETCHR          ;     BEGIN
        POP     CX              ;       delim:=ichar; get_char;
        MOV     BX,OFFSET DELIMTAB;       found:=false;
L06250: INC     BX              ;       cur:=3;
        INC     BX              ;       WHILE (NOT found) AND
        MOV     AL,[BX]         ;             (delimiter_table(cur) <> 0) DO
        OR      AL,AL           ;       BEGIN
        MOV     CL,EILLCH       ;         IF delim=delimiter_table(cur) THEN
        JNZ     L06255          ;
        JMP     ERR0            ;         BEGIN
L06255: INC     BX              ;           found:=true;
        CMP     AL,CH           ;           IF delimiter_table(cur+1)<>0 THEN
        JNZ     L06260          ;           IF delimiter_table(cur+1)<>ichar THEN
        MOV     AL,[BX]         ;             found:=false
        OR      AL,AL           ;
        JZ      L06280          ;           ELSE
        MOV     AL,ICHAR        ;             get_char;
        CMP     AL,[BX]         ;         END;
        JZ      L06270          ;         cur:=cur+4;
L06260: INC     BX              ;       END;
        JMPS    L06250          ;
                EJECT
L06270: PUSH    BX              ;
        CALL    GETCHR          ;
```

```
        POP     BX                  ;
L06280: DEC     BX                  ;        IF found THEN
        DEC     BX                  ;        BEGIN
        MOV     CL,[BX]             ;          symbol:=delimiter_table(cur-6);
        MOV     CH,0                ;          sym_value:=delimiter_table(cur-5);
        DEC     BX                  ;        END
        MOV     AL,[BX]             ;        ELSE error(err_illchar);
L06285: MOV     B_SYMBOL,AL         ;      END;
        MOV     SYMVAL,CX           ;    END;(* CASE ichar OF*)
        RET                         ; END; (*PROCEDURE scan*)
```

```
                EJECT

;*****************************************************************************
;
;
;                       ENCODE3: PROCEDURE expression
;
;
;*****************************************************************************

                EJECT
          ;
          ; PROCEDURE expression(VAR typ: integer);
          ;       call    return
          ; AL            typ
          ;
EXPRSN:   MOV     BX,OFFSET RECURS; BEGIN
          DEC     BYTE PTR [BX]   ;   recursion:=recursion-1;
          MOV     CL,ERECU        ;   IF recursion=0 THEN error(err_recursion)
          JNZ     L10000          ;
          JMP     ERROR           ;   ELSE
L10000:   XOR     AL,AL           ;   BEGIN
          CALL    PUSH            ;   push(0);
          MOV     AL,EOX          ;   push(eox);(*lowest priority*)
          CALL    PUSH            ;
L11000:   MOV     AL,B_SYMBOL     ;   REPEAT
          CMP     AL,ADDOPR       ;     (*operand*)
          JZ      L11010          ;     IF (symbol=add_operator) OR
          CMP     AL,NEGATION     ;        (symbol=negation)     THEN
N         JNZ     L11030          ;     IF getcore(stack_ptr) IN bool_rel_ops THE
L11010:   MOV     BX,STACKPTR     ;     BEGIN (*monadic allowed*)
          MOV     AL,[BX]         ;
          MOV     Di,OFFSET T_019 ;       push(real_type);
          MOV     CX,SIZE19       ;       (* 1. operand dummy *)
          CLD                     ;
          REPNZ   SCASB           ;
          JNZ     L11030          ;
          MOV     AL,REALTYP      ;       IF symbol=add_operator THEN
          CALL    PUSH            ;         sym_value:=sym_value+mon_op;
          MOV     AL,B_SYMBOL     ;
          CMP     AL,ADDOPR       ;       push(sym_value);
          MOV     AL,B_SYMVAL     ;
          JNZ     L11020          ;       scan;
          ADD     AL,MONOP        ;
L11020:   CALL    PUSH            ;
          CALL    SCAN            ;     END;
L11030:   MOV     BX,OFFSET L11140;
          PUSH    BX              ;
          MOV     BX,OFFSET L11040;
          MOV     CX,OFFSET L11042-OFFSET L11040;
          CALL    CASE            ;
                  EJECT
          DW      L11130          ; OTHERWISE Error( Illegal function )
          DW      L11120          ; String function
          DW      L11110          ; standard function
          DW      L11100          ; Left parantes
          DW      VAR             ; String variabel
          DW      L11080          ; Identifyer
```

```
        DW      L11070          ; Logical konstant
        DW      L11060          ; String konstant
        DW      L11050          ; Number
L11040  DB      NUMBER          ;
        DB      STRNGLIT        ;
        DB      LOGCONST        ;
        DB      IDENTIFY        ;
        DB      STRNGVAR        ;
        DB      LEFTPAR         ;
        DB      STDFUNC         ;
        DB      STRFUNC         ;
L11042:                         ;
                EJECT
                                ;     CASE symbol OF
L11050: CALL    SCAN            ;       number:
        MOV     AL,REALTYP      ;       BEGIN
        RET                     ;         typ:=real_type; scan;
                                ;       END;
                                ;
L11060: CALL    SCAN            ;       string_literal:
        MOV     AL,STRNGTYP     ;       BEGIN
        RET                     ;         typ:=string_type; scan;
                                ;       END;
                                ;
L11070: MOV     AL,B_SYMVAL     ;       logical_constant:
        CMP     AL,LOGTRUE      ;       BEGIN
        JNZ     L11071          ;         typ:=real_type;
        MOV     CX,0E001H       ;         IF sym_value=log_true THEN
        JMPS    L11072          ;           store(-#h6001)
L11071: MOV     CX,0E000H       ;         ELSE
L11072: CALL    STSCAN          ;           store(-#h6000);
        MOV     AL,REALTYP      ;         scan;
        RET                     ;       END;
                                ;
                EJECT
L11080: MOV     BX,SYMVAL       ;       identifier:
        PUSH    BX              ;       BEGIN
        CALL    SCAN            ;           typ:=real_type;
        CMP     AL,LEFTPAR      ;         save:=sym_value;
        JNZ     J3000           ;
        CALL    SCNPAR          ;         scan;
J3000:  POP     CX              ;         IF symbol=left_par THEN
        CALL    STOWRD          ;           scan_params;
        MOV     AL,REALTYP      ;         store(save);
        RET                     ;       END;
                                ;
                                ;       string_var:
                                ;         variable(typ);
                                ;
L11100: CALL    SCAN            ;       left_par:
        CALL    EXPRSN          ;       BEGIN
        PUSH    AX              ;         scan;
        MOV     AL,B_SYMBOL     ;         expression(typ);
        CMP     AL,RPAR         ;         IF symbol<>right_par THEN
        MOV     CL,ERPAR        ;           error(err_rpar);
        JZ      J3001           ;
        CALL    ERROR           ;         store(sym_value);
J3001:  MOV     CX,SYMVAL       ;         scan;
        CALL    STSCAN          ;       END;
        POP     AX              ;
        RET                     ;
                EJECT
L11110: MOV     CX,SYMVAL       ;       std_function:
```

```
        MOV     AL,CL           ;
        CMP     AL,RNDOP        ;        BEGIN
        JNZ     L11111          ;          CASE sym_value OF
        PUSH    CX              ;          rnd_op:
        CALL    SCAN            ;            BEGIN
        CMP     AL,LEFTPAR      ;              scan;
        POP     CX              ;              IF symbol = left_par THEN
        JNZ     L11112          ;                scan_std2
        CALL    SCNS02          ;              ELSE
        JMPS    L11115          ;                store(rnd_op);
L11112: CALL    STOWRD          ;
        JMPS    L11115          ;
L11111: CMP     AL,EOP          ;            END;
        JZ      L11113          ;          err_op, eod_op, Pi_Op:
        Cmp     Al,PiOp         ;
        Jz      L11113          ;            BEGIN
        CMP     AL,EODOP        ;
        JNZ     L11114          ;              store(sym_value); scan;
L11113: CALL    STSCAN          ;            END
        JMPS    L11115          ;          OTHERWISE
L11114: CALL    SCNSD1          ;            scan_std1;
L11115: MOV     AL,REALTYP      ;          END;
        RET                     ;          typ:=real_type;
                                ;        END;
                                ;
L11120: MOV     AL,ICHAR        ;        string_function:
        CMP     AL,DOLLAR       ;        BEGIN
        JZ      J3002           ;
        CALL    ERRSYN          ;          IF ichar<>dollar THEN
J3002:  CALL    GETCHR          ;            error(err_syntax);
        MOV     AL,B_SYMVAL     ;          get_char;
        CMP     AL,GETOP        ;
        JNZ     L11121          ;          CASE sym_value OF
        CALL    SCNSD2          ;
        JMPS    L11123          ;
L11121: CMP     AL,KEYOP        ;            get_op: scan_std2;
        Je      x11122          ;
        Cmp     Al,Screen_Op    ;
        Je      x11122          ;            key_op, Screen_Op,
        Cmp     Al,Date_Op      ;            Data_Op,Time_Op:
        Je      x11122          ;
        Cmp     Al,Time_Op      ;
        Je      x11122          ;
        Jmps    L11122          ;            BEGIN
x11122: MOV     CH,0            ;              store(sym_value);
        MOV     CL,AL           ;              scan;
        CALL    STSCAN          ;            END;
        JMPS    L11123          ;            OTHERWISE scan_std1;
L11122: CALL    SCNSD1          ;          END;
L11123: CALL    SCNSUB          ;          scan_substring;
        MOV     AL,STRNGTYP     ;          typ:=string_type;
        RET                     ;        END;
                                ;
L11130: MOV     CL,EILLOP       ;        OTHERWISE
        JMP     ERROR           ;          error(err_illopn);
                                ;
L11140: CALL    PUSH            ;      END;(*CASE symbol OF*)
                                ;      push(typ);
                                ;      (* end operand *)
                EJECT
        MOV     AL,B_SYMBOL     ;
        MOV     Di,OFFSET T_021 ;      IF symbol IN operators THEN
        MOV     CX,SIZE21       ;        opt:=sym_value
```

```
        CLD                     ;
        REPNZ   SCASB           ;       ELSE
        MOV     CX,SYMVAL       ;
        JZ      L11160          ;         opt:=eox;
        MOV     CX,EOX          ;
L11160: PUSH    CX              ;(*opt*)
        MOV     BX,Offset T_PRIO-EXPNTOP;
        ADD     BX,CX           ;       (* operator *)
        MOV     AL,[BX]         ;
L11170: MOV     BX,STACKPTR     ;       prio:= priority(opt);
        DEC     BX              ;       opt1:=getcore(stack_ptr-1);
        MOV     CL,[BX]         ;
        MOV     CH,0            ;
        MOV     BX,Offset T_PRIO-EXPNTOP;
        ADD     BX,CX           ;
        CMP     AL,[BX]         ;       WHILE prio<=priority(opt1) DO
        JBE     L11180          ;       BEGIN
        JMP     L11280          ;
                EJECT
L11180: MOV     AH,AL           ;
        PUSH    AX              ;(*prio*)
        PUSH    CX              ;(*opt1*)
        MOV     AL,CL           ;
        MOV     BX,STACKPTR     ;         (* pop operator *)
        MOV     DL,[BX]         ;         typ2:=getcore(stack_ptr);
        DEC     BX              ;
        DEC     BX              ;
        MOV     DH,[BX]         ;         typ1:=getcore(stack_ptr-2);
        DEC     BX              ;
        MOV     STACKPTR,BX     ;         stack_ptr:=stack_ptr-3;
        MOV     BX,OFFSET L11190;
        MOV     CX,20           ;
        CALL    CASE            ;
                EJECT
        DW      L11260          ; OTHERWISE Error(77);
        DW      L11250          ; Eox
        DW      L11240          ; ↑
        DW      L11240          ; MOD
        DW      L11240          ; DIV
        DW      L11240          ; /
        DW      L11240          ; *
        DW      L11240          ; - (Monadisk)
        DW      L11240          ; + (Monadisk)
        DW      L11240          ; -
        DW      L11230          ; IN
        DW      L11220          ; +
        DW      L11220          ; >
        DW      L11220          ; =>
        DW      L11220          ; <=
        DW      L11220          ; <
        DW      L11220          ; <>
        DW      L11220          ; =
        DW      L11240          ; OR
        DW      L11240          ; AND
        DW      L11200          ; NOT
L11190  DB      NOTOP           ;
        DB      ANDOP           ;
        DB      OROP            ;
        DB      EQOP            ;
        DB      NEOP            ;
        DB      LSOP            ;
        DB      LEOP            ;
        DB      GEOP            ;
```

```
        DB      GROP            ;
        DB      PLUSOP          ;
        DB      INOP            ;
        DB      MINUSOP         ;
        DB      MONPLSOP        ;
        DB      MONMINOP        ;
        DB      MULTOP          ;
        DB      DIVIDEOP        ;
        DB      DIVOP           ;
        DB      MODOP           ;
        DB      EXPNTOP         ;
        DB      EOX             ;
                EJECT
                                ;       CASE opt1 OF (* check types *)
                                ;       not_op:
                                ;       BEGIN
L11200: MOV     AL,DL           ;         IF (typ2<>real_type) THEN
        CMP     AL,REALTYP      ;           error(err_illtype);
        JNZ     L11300          ;         typ2:=real_type;
        JMP     L11270          ;       END;
                                ;
L11220:                         ;       eq_op,not_eq_op,ls_op,ls_eq_op,gr_op,
        MOV     AL,DL           ;       gr_op,plus_op:
        CMP     AL,DH           ;       BEGIN
        JNZ     L11300          ;         IF typ1<>typ2 THEN
        POP     CX              ;           error(err_illtype);
        MOV     AL,PLUSOP       ;         IF opt1<>plus_op THEN
        CMP     AL,CL           ;           typ2:=real_type;
        JZ      L11221          ;         CASE typ1 OF
        MOV     DL,REALTYP      ;           real_type: ;
L11221: MOV     AL,REALTYP      ;           string_type:
        CMP     AL,DH           ;             opt1:=opt1+string_op;
        JZ      L11222          ;           OTHERWISE
        MOV     AL,STRNGTYP     ;             error(err_illtype);
        CMP     AL,DH           ;
        JNZ     L11300          ;         END;
        MOV     AL,STRINGOP     ;
        ADD     AL,CL           ;
        MOV     CL,AL           ;
L11222: PUSH    CX              ;
        JMP     L11270          ;       END;
                                ;
                EJECT
L11230: MOV     AL,DL           ;       in_op:
        MOV     DL,REALTYP      ;       BEGIN
        CMP     AL,STRNGTYP     ;         IF (typ1<>string_type) OR
        JNZ     L11300          ;            (typ2<>strng_type)  THEN
        MOV     AL,DH           ;           error(err_illtype);
        CMP     AL,STRNGTYP     ;         typ2:=real_type;
        JNZ     L11300          ;
        JMP     L11270          ;       END;
                                ;
                                ;       minus_op,mon_plus_op,mon_minus_op,
                                ;       multiply_op,divide_op,div_op,mod_op,
L11240: MOV     AL,REALTYP      ;       exponent_op,and_op,or_op:
        CMP     AL,DL           ;       BEGIN
        JNZ     L11300          ;         IF (typ1<>real_type) OR
        CMP     AL,DH           ;            (typ2<>real_type) THEN
        JNZ     L11300          ;           error(err_illtype);
        JMP     L11270          ;       END;
                                ;
                                ;       eox:
                                ;       BEGIN (*end of expression*)
```

```
                                        ;         GOTO 10;
                                        ;       END;
                                        ;
L11260: MOV     CL,77                   ;       OTHERWISE
        CALL    ERROR                   ;         error(77);
                EJECT
L11270: MOV     AL,DL                   ;    END;(*CASE opt1 OF*)
        CALL    PUSH                    ;    push(typ2);(*resulting type*)
        POP     CX                      ;    store(opt1);
        CALL    STOWRD                  ;    opt1:=getcore(stack_ptr-1);
; May need XCHG AH,AL
        POP     AX                      ;(*prio(opt)*)

        JMP     L11170                  ;    END;
                                        ;
L11280: POP     CX                      ;
        MOV     AL,CL                   ;    push(opt);
        CALL    PUSH                    ;    (* end operator *)
        CALL    SCAN                    ;    scan;
        JMP     L11000                  ; UNTIL false
L11300: MOV     CL,EILLTP               ;
        JMP     ERROR                   ;
                                        ;    10:
L11250: MOV     AL,DL                   ;    typ:=typ2;
        MOV     BX,OFFSET RECURS        ;    recursion:=recursion+1;
        INC     BYTE PTR [BX]           ;
        POP     BX                      ;
        POP     BX                      ;
        POP     BX                      ;
        RET                             ; END; (* PROCEDURE expression *)
```

```
                EJECT

;*****************************************************************************
;
;
;
;               ENCODE4: PROCEDURE parse_statement
;
;
;
;*****************************************************************************
                EJECT
        ;
        ; PROCEDURE parse_statement(stmt_type: integer);
        ;
        ;       call    return
        ; AL    stmt_type
        ;
        ; CONST

THCLSEOX        EQU     THENCLSM*256+EOX        ;
THSTEOX         EQU     THENSTSM*256+EOX        ;
OFEOX           EQU     OFSYM*256+EOX           ;
WHCEOX          EQU     WHCOMSYM*256+EOX        ;
WHEOX           EQU     WHENSYM*256+EOX         ;
DOCLSEOX        EQU     DOCLSSYM*256+EOX        ;
DOSTMEOX        EQU     DOSTMSYM*256+EOX        ;
UNTILEOX        EQU     UNTSYM*256+EOX          ;
EXECEOX         EQU     EXECSYM*256+EOX         ;
FILEEOX         EQU     FILESYM*256+EOX         ;
FCOMEOX         EQU     FLCOMSYM*256+EOX        ;
CHEOX           EQU     CHANSYM*256+EOX         ;
USINGEOX        EQU     USISYM*256+EOX          ;
RTRNEOX         EQU     RETRNMOD*256+EOX        ;

TSYMV:  MOV     BX,SYMVAL               ; BEGIN
        CALL    STOWHL                  ;  store(sym_val); scan;
        JMP     SCAN                    ; END;

                EJECT

PARSTM: MOV     CL,AL                   ;BEGIN
        MOV     CH,0                    ;  store(stmt_type);
        CALL    STOWRD                  ;
        MOV     BX,OFFSET L18000;
        MOV     CX,OFFSET L18010-OFFSET L18000;
        CALL    CASE                    ;
                EJECT
        Dw      L18600                  ; OtherWise SyntaxError( Not Implemented )
        Dw      L18440                  ; GSX
        Dw      L18430                  ; LOCATE
        Dw      L18290                  ; TEXT
        Dw      L18410                  ; WINDOW
        DW      L18400                  ; MOVE
        DW      L18400                  ; MOVETO
        DW      L18400                  ; DRAW
        DW      L18400                  ; DRAWTO
        DW      L18420                  ; CIRCLE
        DW      L18400                  ; PALETTE
        DW      L18280                  ; PENCOLOUR
        DW      SCAN                    ; CLEAR
```

```
        DW      L18390          ; NEW
        DW      L18380          ; CLOSE
        DW      L18330          ; OPEN
        DW      L18320          ; CREATE
        DW      L18310          ; CHAIN
        DW      L18310          ; DELETE
        DW      L18300          ; RENAME
        DW      L18290          ; PREFIX
        DW      L18285          ; DIR
        DW      L18280          ; ZONE
        DW      L18280          ; MARGIN
        DW      L18280          ; MOUNT
        DW      L18270          ; OUT
        DW      L18270          ; SELECET
        DW      SCAN            ; DISABLE
        DW      SCAN            ; CONTINUE
        DW      SCAN            ; RETRY
        DW      SCAN            ; RNDOMIZE
        DW      SCAN            ; ENDIF
        DW      SCAN            ; STOP
        DW      SCAN            ; ENDCASE
        DW      SCAN            ; END
        DW      SCAN            ; BYE
        DW      L18260          ; RETURN
        DW      L18250          ; ENABLE
        DW      L18255          ; ENDFUNC
        DW      L18250          ; ENDPROC
        DW      L18240          ; IMPORT
        DW      L18240          ; GLOBAL
        DW      L18230          ; FUNC
        DW      L18210          ; PROC
        DW      L18200          ; EXEC
        DW      L18190          ; UNTIL
        DW      STOSC           ; ELSE
        DW      STOSC           ; ENDWHILE
        DW      STOSC           ; OTHWISE
        DW      STOSC           ; REPEAT
        DW      L18170          ; WHILE
        DW      L18160          ; GOTO
        DW      L18160          ; NEXT
        DW      L18140          ; FOR
        DW      L18130          ; WHEN
        DW      L18120          ; CASE
        DW      L18110          ; IF
        DW      L18090          ; PRINT
        DW      L18070          ; INPUT
        DW      L18060          ; RESTORE
        DW      L18040          ; DIM
        DW      L18030          ; DATA
        DW      L18020          ; WRITE
        DW      L18010          ; READ
                EJECT
L18000  DB      READSTM
        DB      WRITESTM
        DB      DATASTM
        DB      DIMSTM
        DB      RESTRSTM
        DB      INPUTSTM
        DB      PRINTSTM
        DB      IFSTMT
        DB      CASESTM
        DB      WHENSTM
        DB      FORSTM
```

```
        DB      NEXTSTM
        DB      GOTOSTM
        DB      WHILESTM
        DB      REPSTM
        DB      OTHWSSTM
        DB      ENDWHSTM
        DB      ELSESTM
        DB      UNTSTM
        DB      EXECSTM
        DB      PROCSTM
        DB      FUNCSTM
        DB      GLOBLSTM
        DB      IMPRTSTM
        DB      ENDPRSTM
        DB      ENDFUSTM
        DB      ENABLSTM
        DB      RETRNSTM
        DB      BYESTM
        DB      ENDSTM
        DB      ENDCSSTM
        DB      STOPSTM
        DB      ENDIFSTM
        DB      RNDMZSTM
        DB      RETRYSTM
        DB      CONTNSTM
        DB      DSABLSTM
        DB      SELCTSTM
        DB      OUTSTM
        DB      MOUNTSTM
        DB      MARGSTM
        DB      ZONESTM
        DB      DIRSTM
        DB      PFXSTM
        DB      RENAMSTM
        DB      DELETSTM
        DB      CHAINSTM
        DB      CREATSTM
        DB      OPENSTM
        DB      CLOSTM
        DB      NEWSTM
        DB      Clear_Stm
        DB      PenColour_Stm
        DB      Palette_Stm
        DB      Circle_Stm
        DB      DrawTo_Stm
        DB      Draw_Stm
        DB      MoveTo_Stm
        DB      Move_Stm
        DB      Window_Stm
        Db      Text_Stm
        Db      Locate_Stm
        Db      Gsx_Stm
        EJECT
                                ;  CASE stmt_type OF
                                ;    read_stmt:
L18010: CALL    SCAN            ;      BEGIN
        CMP     AL,FILESYM      ;        scan;
        MOV     CX,READOP+256*COMMA;     IF symbol=file_symbol THEN
        JNZ     L18014          ;        BEGIN
        CALL    SCNFID          ;          scan_file_id;
        MOV     CX,RDFILEOP+256*COMMA;
        JMPS    L18014          ;          opt:= read_file_op;
L18012: PUSH    CX              ;        END
```

```
        CALL    STSCAN          ;          ELSE opt:= read_op;
        POP     CX              ;          variable(typ);
L18014: PUSH    CX              ;          WHILE symbol = comma DO
        CALL    VAR             ;          BEGIN
        POP     CX              ;            store(comma*256+opt);
        MOV     AL,B_SYMBOL     ;            scan; variable(typ);
        CMP     AL,CH           ;          END;
        JZ      L18012          ;          store(opt);
        MOV     CH,0            ;        END;
        JMP     STOWRD          ;
                EJECT
                                ;      write_stmt:
L18020: CALL    SCAN            ;        BEGIN
        CMP     AL,FILESYM      ;          scan;
        JZ      J4000           ;          IF symbol <> file_symbol  THEN
        CALL    ERRSYN          ;            error(err_syntax);
J4000:  CALL    SCNFID          ;          scan_file_id;
        CALL    EXPRSN          ;          expression(typ);
L18022: MOV     AL,B_SYMBOL     ;          WHILE symbol = comma DO
        CMP     AL,COMMA        ;          BEGIN
        MOV     CX,WRFILEOP     ;            store(comma_write_file);
        JZ      L18023          ;
        JMP     STOWRD          ;            scan;
L18023: MOV     CH,COMMA        ;            expression(typ);
        CALL    STSCAN          ;          END;
        CALL    EXPRSN          ;          store(write_file_op);
        JMPS    L18022          ;        END;

                                ;      data_stmt:
L18030: MOV     CX,0            ;        BEGIN
        CALL    STOWRD          ;          store(0); (* chain *)
L18031: CALL    GETNBL          ;          REPEAT
        CMP     AL,'"'          ;            get_non_blank;
        JNZ     L18032          ;            IF ichar=ord('"') THEN
        CALL    SCAN            ;              scan
        JMPS    L18033          ;            ELSE
L18032: CALL    SCNNUM          ;              scan_number;
L18033: CALL    SCAN            ;            scan;
        CMP     AL,COMMA        ;
        JZ      L18031          ;          UNTIL symbol<>comma;
        RET                     ;        END;
                EJECT
                                ;      dim_stmt:
L18040: CALL    SCAN            ;        BEGIN
        PUSH    CX              ;          REPEAT
        CMP     AL,IDENTIFY     ;            scan; save_id:=sym_value;
        JNZ     L18042          ;            IF symbol=identifier THEN
                                ;            BEGIN
        CALL    SCAN            ;              scan;
        CMP     AL,LEFTPAR      ;              IF symbol<>left_par THEN
        MOV     CL,ELPAR        ;                error(err_lpar);
        JZ      J4001           ;
        CALL    ERROR           ;              scan_dimensions(arg_no);
J4001:  CALL    SCNDIM          ;              store(real_dim_op + 256*arg_no);
        MOV     CL,RLDIMOP      ;            END
        JMPS    L18044          ;            ELSE
L18042: CMP     AL,STRNGVAR     ;            IF symbol = string_var THEN
        JZ      J4002           ;
        CALL    ERRSYN          ;            BEGIN
J4002:  CALL    SCAN            ;              scan;
        MOV     CH,0            ;              arg_no:= 0;
        CMP     AL,LEFTPAR      ;              IF symbol = left_par THEN
        JNZ     J4003           ;
```

```
        CALL    SCNDIM          ;                scan_dimensions(arg_no);
J4003:  PUSH    CX              ;
        CMP     AL,OFSYM        ;              IF symbol<>of_symbol THEN
        MOV     CL,EOF          ;                error(err_of);
        JZ      J4004           ;
        CALL    ERROR           ;
J4004:  CALL    SCART           ;              scan; arith_expression;
        POP     CX              ;              store(arg_no*256 + string_dim_op);
        MOV     CL,STRDIMOP     ;
L18044: CALL    STOWRD          ;            END
        POP     CX              ;            ELSE error(err_syntax);
        CALL    STOWRD          ;            store(save_id);
        MOV     CH,COMMA        ;            store(comma_eox);
        CALL    STBEOX          ;
        MOV     AL,B_SYMBOL     ;          UNTIL symbol <> comma;
        CMP     AL,COMMA        ;
        JZ      L18040          ;
        MOV     BX,PROGPTR      ;          putcore(prog_ptr-1, eox);
        DEC     BX              ;
        MOV     BYTE PTR [BX],0 ;
        RET                     ;      END;
                EJECT
                                ;      restore_stmt:
L18060: CALL    SCAN            ;        BEGIN
        CMP     AL,IDENTIFY     ;          scan;
        RETNZ                   ;          IF symbol = identifier THEN
        JMP     STSYMV          ;          BEGIN
                                ;            store(sym_value); scan;
                                ;          END;
                                ;        END;
                                ;
                                ;      input_stmt:
L18070: CALL    SCAN            ;        BEGIN
        CMP     AL,FILESYM      ;          scan;
        JNZ     L18071          ;          IF symbol=file_symbol THEN
        CALL    SCNFID          ;          BEGIN
        MOV     CX,INPFOP       ;            scan_file_id;
        CALL    STOWRD          ;            store(input_file_op);
        JMPS    L18072          ;          END ELSE
                                ;          BEGIN
L18071: CMP     AL,ATSYM        ;            IF (symbol=at_symbol) OR
        JZ      L18074          ;               (symbol=string_literal) THEN
        CMP     AL,STRNGLIT     ;            BEGIN
        JNZ     L18079          ;              IF symbol=string_literal THEN
L18073: MOV     CX,COLON*256+PRMPT1OP ;       BEGIN
        CALL    STSCAN          ;                store(colon_prompt1);
        JMPS    L18076          ;                scan;
L18074: CALL    SCNSD2          ;              END
        CMP     AL,COMMA        ;              ELSE
        JNZ     L18075          ;              BEGIN
        MOV     CX,COMMA*256+EOX;                scan_std2;
        CALL    STSCAN          ;                IF symbol=comma THEN
        CMP     AL,STRNGLIT     ;                BEGIN
        MOV     CL,ECOLON       ;                  store(comma_eox);
        JZ      J4005           ;                  scan;
        CALL    ERROR           ;              IF symbol<>string_literal THEN
J4005:  MOV     CX,COLON*256+PRMPT1OP
        CALL    STSCAN          ;                    error(err_colon);
        JMPS    L18076          ;                store(colon_prompt1);
L18075: MOV     CH,COLON        ;                  scan;
        CALL    STBEOX          ;                END
        MOV     CX,PRMPT2OP     ;                ELSE
        CALL    STOWRD          ;                BEGIN
```

```
L18076: MOV     AL,B_SYMBOL     ;                 store(colon_eox);
        CMP     AL,COLON        ;                 store(prompt2_op);
        MOV     CL,ECOLON       ;               END;
        JZ      J4006           ;             END;
        CALL    ERROR           ;             IF symbol<>colon THEN
J4006:  CALL    SCAN            ;               error(err_colon);
        JMPS    L18072          ;
L18079: MOV     CX,PRMPT2OP     ;             scan;
        CALL    STOWRD          ;           END
                                ;           ELSE store(prompt2_op);
L18072: CALL    VAR             ;         END;
        MOV     AL,B_SYMBOL     ;         variable(typ)
        CMP     AL,COMMA        ;         WHILE symbol=comma DO
        JNZ     L18078          ;         BEGIN
        MOV     CX,COMMA*256+INPOP;         store(comma*256 + input_op);
        CALL    STSCAN          ;         scan; variable(typ);
        JMPS    L18072          ;       END;
                EJECT
L18078: MOV     CX,INPOP        ;       IF symbol = semicolon THEN
        CMP     AL,SEMICOLN     ;       BEGIN
        JZ      JX000           ;
        JMP     STOWRD          ;         store(semicolon*256 + input_op);
JX000:  MOV     CH,SEMICOLN     ;         scan;
        JMP     STSCAN          ;       END ELSE store(input_op);
                                ;     END;
                                ;   print_stmt:
L18090: CALL    SCAN            ;     BEGIN
        CMP     AL,ENDOFLIN     ;       scan;
        JNZ     L18091          ;       IF symbol = end_of_line THEN
        MOV     CX,PRNTNLOP     ;         store(print_nl_op)
        JMP     STOWRD          ;       ELSE
L18091: CMP     AL,FILESYM      ;       BEGIN
        JNZ     L18102          ;         IF symbol=file_symbol THEN
        CALL    SCNFID          ;           scan_file_id;
L18102: CMP     AL,USISYM       ;         IF symbol=using_symbol THEN
        JNZ     L18103          ;         BEGIN
        CALL    SCSTR           ;           scan;string_expression;
        MOV     CH,USISYM       ;           store(using_eox);
        CALL    STBEOX          ;           opt:=using_op;
        MOV     CX,USINGOP      ;           IF symbol<>colon THEN
        PUSH    CX              ;             error(err_colon);
        MOV     AL,B_SYMBOL     ;           scan;
        CMP     AL,COLON        ;         END
        MOV     CL,ECOLON       ;         ELSE
        JZ      J4007           ;
        CALL    ERROR           ;           opt:=print_op;
J4007:  CALL    SCAN            ;
        JMPS    L18093          ;
L18103: MOV     CX,PRINTOP      ;
        PUSH    CX              ;
L18093: CMP     AL,ATSYM        ;         REPEAT
        JNZ     L18094          ;           IF symbol=at_symbol THEN
        CALL    SCNSD2          ;             scan_std2
        JMPS    L18095          ;           ELSE
L18094: CMP     AL,TABSYM       ;             IF symbol=tab_symbol THEN
        JNZ     L18096          ;               scan_std1
        CALL    SCNSD1          ;             ELSE
        JMPS    L18095          ;               expression(typ);
L18096: CALL    EXPRSN          ;
                EJECT
L18095: MOV     AL,B_SYMBOL     ;           IF (symbol=comma) OR
        CMP     AL,COMMA        ;              (symbol=semicolon) THEN
                                ;           BEGIN
```

```
        JZ      L18098          ;                 store(symbol*256+opt);
L18097: CMP     AL,SEMICOLN     ;                 scan;
        JNZ     L18099          ;               END
L18098: POP     CX              ;               ELSE
        PUSH    CX              ;                 IF symbol=end_of_line THEN
        MOV     CH,AL           ;                   store(opt)
        CALL    STSCAN          ;                 ELSE
        CMP     AL,ENDOFLIN     ;
        JNZ     L18093          ;                   error(err_syntax);
        POP     CX              ;
        RET                     ;
rL#8099: CMP    AL,ENDOFLIN     ;             UNTIL symbol=end_of_line; (* OR erro
        POP     CX              ;           END;
        JZ      JX007           ;
        JMP     ERRSYN          ;         END;
JX007:  JMP     STOWRD          ;
                EJECT
                                ;     if_stmt:
L18110: CALL    ST00SC          ;       BEGIN
nces *) CALL    ARTEXP          ;         store(0); store(0); (* room for refere
        CMP     AL,THENSTSM     ;         scan;         (* if_clause only*)
        MOV     CL,ETHEN        ;         arith_expression;
        JZ      J4008           ;
        CALL    ERROR           ;         if symbol <> then_stmt_symbol then
J4008:  CALL    SCAN            ;           error(err_then);
        CMP     AL,ENDOFLIN     ;         scan;
        JNZ     L18111          ;         if symbol = end_of_line then
        MOV     CH,THENCLSM     ;         begin
        CALL    STBEOX          ;           store(then_cl_eox);
        MOV     BX,OFFSET CORE0+4;           putcore(2, if_clause);
                                ;         end
        MOV     BYTE PTR [BX],IFCLS;         else begin
                                ;           store(then_st_eox);
                                ;           simple_statement;
        RET                     ;         end;
                                ;       end;
L18111: MOV     CH,THENSTSM     ;
        CALL    STBEOX          ;
        JMP     SIMSTM          ;
                                ;
                                ;     case_stmt:
L18120: CALL    ST00SC          ;       begin
        CALL    EXPRSN          ;         store(0); (* room for reference *)
        MOV     AL,B_SYMBOL     ;         store(0);scan; expression(typ);
of);    CMP     AL,OFSYM        ;         if symbol <> of_symbol then error(err_
        MOV     CL,EOF          ;         store(of_eox);
        JZ      J4009           ;
        CALL    ERROR           ;         scan;
J4009:  MOV     CX,OFEOX        ;       end;
        JMP     STSCAN          ;
                EJECT
                                ;     when_stmt:
L18130: CALL    ST0SC           ;       BEGIN
        CALL    EXPRSN          ;         store(0); (* room for reference *)
L18131: MOV     AL,B_SYMBOL     ;         scan; expression(typ);
        CMP     AL,COMMA        ;         WHILE symbol=comma DO
        JNZ     L18132          ;         BEGIN
        MOV     CX,WHCEOX       ;           store(when_com_eox);
        CALL    STSCAN          ;           scan; expression(typ);
        CALL    EXPRSN          ;         END;
        JMPS    L18131          ;         store(when_eox);
L18132: MOV     CH,WHENSYM      ;       END;
        JMP     STBEOX          ;
```

```
                EJECT
                                        ;    for_stmt:
L18140: CALL    STOOSC                  ;      begin
                                        ;        store(0);  (* room for references *)
*)                                      ;        store(0);scan;  (* for_clause only
        CMP     AL,IDENTIFY             ;        if symbol <> identifier then
        JZ      J4010                   ;
        CALL    ERRSYN                  ;          error(err_syntax);
J4010:  CALL    STSYMV                  ;        store(sym_value); scan;
ign);   CMP     AL,ASSIGN               ;        if symbol <> assign then error(err_ass
        MOV     CL,EASSGN               ;
        JZ      J4011                   ;
        CALL    ERROR                   ;
J4011:  CALL    SCART                   ;        scan; arith_expression;
        MOV     CX,RLASSOP              ;        store(real_assign_op);
        CALL    STOWRD                  ;
to);    MOV     AL,B_SYMBOL             ;        if symbol <> to_symbol then error(err_
        CMP     AL,TOSYM                ;
        MOV     CL,ETO                  ;
        JZ      J4012                   ;
        CALL    ERROR                   ;
J4012:  CALL    SCART                   ;        scan; arith_expression;
        MOV     CX,TOOP                 ;        store(to_op);
        CALL    STOWRD                  ;
        MOV     AL,B_SYMBOL             ;        if symbol = step_symbol then
        CMP     AL,STEPSYM              ;
        JNZ     L18143                  ;        BEGIN
        CALL    SCART                   ;          scan; arith_expression;
        MOV     CL,STEP1OP              ;          typ:= step1_op;
        PUSH    CX                      ;        END
        JMPS    L18144                  ;        ELSE
                EJECT
L18143: MOV     CL,STEP2OP              ;          typ:= step2op;
        PUSH    CX                      ;
L18144: MOV     AL,B_SYMBOL             ;        if symbol <> do_stmt_symbol then
        CMP     AL,DOSTMSYM             ;
        MOV     CL,EDO                  ;          error(err_do);
        JZ      J4013                   ;
        CALL    ERROR                   ;
J4013:  CALL    SCAN                    ;        scan;
        CMP     AL,ENDOFLIN             ;        if symbol = end_of_line then
        JNZ     L18146                  ;        BEGIN
        POP     CX                      ;          store(do_clause_symbol*256 + typ);
        MOV     CH,DOCLSSYM             ;
        CALL    STOWRD                  ;          putcore(2, for_clause);
        MOV     BX,OFFSET CORE0+4;      ;         END
        MOV     BYTE PTR [BX],FORCLS;   ;         ELSE
        RET                             ;        BEGIN
L18146: POP     CX                      ;          store(do_stmt_symbol*256 + typ);
        MOV     CH,DOSTMSYM             ;          simple_statement;
        CALL    STOWRD                  ;        END;
        JMP     SIMSTM                  ;      END;
                EJECT
                                        ;    next_stmt, goto_stmt:
L18160: CALL    STOSC                   ;      BEGIN
*)                                      ;        store(0); scan;  (* room for reference
d t);   CMP     AL,IDENTIFY             ;        IF symbol<>identifier THEN error(err_i
        MOV     CL,EIDENT               ;        store(sym_value); scan;
        JZ      J4014                   ;
        CALL    ERROR                   ;      END;
J4014:  JMP     STSYMV                  ;

                                        ;    while_stmt:
```

```
L18170: CALL    STOOSC          ;       begin
                                ;         store(0);  (* room for references *)
*)                              ;         store(0);scan;  (* while_clause only
                                ;         scan; arith_expression;
        CALL    ARTEXP          ;         if symbol <> do_stmt_symbol then
                                ;           error(err_do);
        CMP     AL,DOSTMSYM     ;         scan;
        MOV     CL,EDO          ;         if symbol = end_of_line then
        JZ      J4015           ;
        CALL    ERROR           ;         begin
J4015:  CALL    SCAN            ;           store(do_clause_eox);
        CMP     AL,ENDOFLIN     ;           putcore(2, while_clause);
        JNZ     L18171          ;         end
        MOV     CH,DOCLSSYM     ;         else
        CALL    STBEOX          ;         begin
        MOV     BX,OFFSET CORE0+4;           store(do_stmt_eox);
        MOV     BYTE PTR [BX],WHILECLS;           simple_statement;
        RET                     ;         end;
L18171: MOV     CH,DOSTMSYM     ;       end;
        CALL    STBEOX          ;
        JMP     SIMSTM          ;
                EJECT
, else_stmt:                    ;     repeat_stmt, otherwise_stmt, endwhile_stmt
                                ;       BEGIN
*)                              ;         store(0); scan;  (* room for reference
                                ;       END;

                                ;     until_stmt:
L18190: CALL    STOSC           ;       BEGIN
        CALL    ARTEXP          ;         store(0);scan; arith_expression;
        MOV     CH,UNTSYM       ;         store(until_eox);
        JMP     STBEOX          ;       END;

                                ;     exec_stmt:
L18200: CALL    SCAN            ;      BEGIN
        CMP     AL,IDENTIFY     ;         scan;
_ident); MOV    CL,EIDENT       ;         IF symbol <> identifier THEN error(err
        JZ      J4016           ;
        CALL    ERROR           ;         save_id:= sym_value;
J4016:  MOV     BX,SYMVAL       ;         scan;
        PUSH    BX              ;         IF symbol = left_par THEN scan_params;
        CALL    SCAN            ;         store(save_id);
        CMP     AL,LEFTPAR      ;         store(exec_eox);
        JNZ     J4017           ;
        CALL    SCNPAR          ;       END;
J4017:  POP     CX              ;
        CALL    STOWRD          ;
        MOV     CH,EXECSYM      ;
        JMP     STBEOX          ;
                EJECT
                                ;     proc_stmt:
L18210: CALL    STOOSC          ;     BEGIN
        CMP     AL,IDENTIFY     ;       store(0); store(0); scan;
        JZ      JX008           ;
        JMP     ERRSYN          ;       IF symbol <> identifier THEN
JX008:  CALL    SCPROC          ;         error(err_syntax);
        CMP     AL,EXTNMDF      ;       scan_proc(type);
        RETNZ                   ;
        MOV     BX,OFFSET CORE0+4;       IF type = external_modif THEN
;       MOV     BYTE PTR [BX],EXTPRSTM  ;         putcore(2, external_proc_stmt)
        RET                     ;     END;

                                ;     func_stmt:
```

```
L18230: CALL    STOOSC          ;     BEGIN
        CMP     AL,IDENTIFY     ;       store(0); store(0); scan;
        JZ      L18232          ;       IF symbol <> identifier AND
        CMP     AL,STRNGVAR     ;          symbol <> string_var THEN
        JZ      L18232          ;
        JMP     ERRSYN          ;         error(err_syntax);
L18232: CALL    SCPROC          ;
        CMP     AL,HANDLMDF     ;       scan_proc(type);
        JNZ     JX009           ;
yntax); JMP     ERRSYN          ;       IF type = handler_modif THEN error(err_s
 JX009: CMP     AL,EXTNMDF      ;
        RETNZ                   ;       IF type = external_modif THEN
        MOV     BX,OFFSET CORE0+4;         putcore(2, external_func_stmt);
        MOV     BYTE PTR [BX],EXTFUSTM  ;
        RET                     ;     END;
                EJECT
                                ;     global_stmt, import_stmt:
L18240: CALL    SCAN            ;       repeat
        CMP     AL,IDENTIFY     ;         scan;
        JZ      L18241          ;         if (symbol <> identifier) and
        CMP     AL,STRNGVAR     ;            (symbol <> string_var) then
        MOV     CL,EILLVR       ;           error(err_illvar);
        JZ      L18241          ;
        CALL    ERROR           ;         store(-sym_value);
L18241: CALL    STSYMV          ;         scan;
        CMP     AL,COMMA        ;       until symbol <> comma;
        JZ      L18240          ;
        RET                     ;
                EJECT
                                ;     endproc_stmt, enable_stmt:
L18250: CALL    SCAN            ;       BEGIN
        JMPS    L18256          ;         scan;
                                ;         IF symbol <> identifier THEN
                                ;           error(err_ident);
                                ;         store(-sym_value); scan;
                                ;       END;

8255: CALL      SCAN            ;     endfunc_stmt:
        CMP     AL,STRNGVAR     ;       BEGIN
        JNZ     L18256          ;
J18256: JMP     STSYMV          ;         scan;
L18256: CMP     AL,IDENTIFY     ;         IF symbol <> string_var AND
        JZ      J18256          ;            symbol <> identifier THEN
        MOV     CL,EIDENT       ;           error(err_ident);
        JMP     ERROR           ;         store(-sym_value); scan;
                                ;       END;

L18260: CALL    SCAN            ;     return_stmt:
        CMP     AL,ENDOFLIN     ;     BEGIN
        RETZ                    ;       scan;
        CALL    EXPRSN          ;       IF symbol<>end_of_line THEN
        MOV     CH,RETRNMOD     ;       BEGIN
        JMP     STBEOX          ;         expression;
                                ;         store(return_eox);
                                ;       END;
                                ;     END;
                                ;
t,                              ;     new_stmt, end_stmt, endcase_stmt, stop_stm
                                ;     endif_stmt, randomize_stmt,retry_stmt
                                ;     continue_stmt, disable_stmt,
                                ;       scan;

                                ;     select_stmt, out_stmt:
```

```
L18270: CMP     AL,SELCTSTM     ;     BEGIN
        JNZ     L18271          ;       IF stmt_type = select_stmt THEN
        CALL    SCAN            ;       BEGIN
        CMP     AL,OUTPSYM      ;         scan;
        JZ      L18271          ;
        CALL    ERRSYN          ;         IF symbol <> output_symbol THEN
L18271: MOV     BX,PROGPTR      ;           error(err_syntax);
        DEC     BX              ;       END;
        DEC     BX              ;
        MOV     BYTE PTR [BX],SELOPSTM;putcore(prog_ptr-1, select_output_stmt);
        JMPS    L18310          ;       scan; string_expression;
                                ;       store(file_eox);
                                ;     END;

                                ;     Pencolour_stmt,
                                ;     mount_stmt,
                                ;     margin_stmt, zone_stmt:
L18280: CALL    SCART           ;       BEGIN
L18281: JMP     STOEOX          ;         scan;arith_expression;
                                ;         store(eox);
                                ;       END;
                EJECT
                                ;     dir_stmt:
L18285: CALL    SCAN            ;       BEGIN
        CMP     AL,ENDOFLIN     ;         scan;
        RETZ                    ;         IF symbol<>end_of_line THEN
        CALL    ARTEXP          ;         BEGIN
        JMPS    L18281          ;           arith_expression; store(eox);
                                ;         END;
                                ;       END;

                                ;     prefix_stmt:
L18290: CALL    SCSTR           ;       BEGIN
        JMPS    L18281          ;         scan; string_expression;
                                ;         store(eox);
                                ;       END;

                                ;     rename_stmt:
L18300: CALL    L18301          ;       BEGIN
        MOV     CL,ECOMMA       ;         scan; string_expression;
        JZ      J4020           ;
        CALL    ERROR           ;         store(file_comma_eox);
aJ4020: JMPS    L18290          ;         IF symbol <> comma THEN error(err_comm
L18301: CALL    SCSTR           ;
        MOV     CH,FLCOMSYM     ;
        CALL    STBEOX          ;
        MOV     AL,B_SYMBOL     ;         scan; string_expression;
        CMP     AL,COMMA        ;         store(eox);
        RET                     ;       END;

                                ;      delete_stmt,chain_stmt:
L18310: CALL    SCSTR           ;       BEGIN
                                ;         scan; string_expression;
        MOV     CH,FILESYM      ;         store(file_eox);
        JMP     STBEOX          ;       END;

                                ;     create_stmt:
L18320: CALL    L18301          ;       BEGIN
        JZ      L18280          ;
        CALL    ERRSYN          ;         scan; string_expression;
        JMPS    L18280          ;         store(file_comma_eox);
ax);                            ;         if symbol <> comma then error(err_synt
```

```
                                        ;         scan; arith_expression;
                                        ;         store(eox);
                                        ;       END;
                EJECT
                                        ;     open_stmt:
L18330: CALL    SCAN                    ;       BEGIN
        Cmp     Al,Graph_Sym            ;         Scan;
        Jnz     L18335                  ;         IF SymBol = Graphics THEN
        Mov     Bx,ProgPtr              ;         BEGIN
        Mov     Byte Ptr Minus2[Bx],Graph_Open_Stm;
        Call    Scan                    ;           PutCore(ProgPtr-2,Graph_Open_Stm);
        Cmp     Al,EndOfLin             ;           Scan;
        Jz      L18332                  ;           IF Symbol <> EndOfLine THEN
        Call    ArtExp                  ;           BEGIN
        Jmp     StoEox                  ;             ArithExp;
                                        ;             StoreEox;
                                        ;           END
L18332: Mov     Cx,0F000h               ;           ELSE
        Jmp     StoWrd                  ;             StoreWord(0F000h); { 0 }
                                        ;         END
                                        ;         ELSE
L18335: CMP     AL,FILESYM              ;         BEGIN
        JNZ     J4022                   ;
        CALL    SCAN                    ;           IF symbol=file_symbol THEN scan;
J4022:  CALL    ARTEXP                  ;
        MOV     CH,CHCOMSYM             ;           arith_expression;
        CALL    STBEOX                  ;           store(ch_comma_eox);
mma);   MOV     AL,B_SYMBOL             ;           IF symbol <> comma THEN error(err_co
        CMP     AL,COMMA                ;
        MOV     CL,ECOMMA               ;
        JZ      J4023                   ;
        CALL    ERROR                   ;           scan;
J4023:  CALL    SCSTR                   ;           string_expression;
        MOV     CH,FLCOMSYM             ;           store(file_comma_eox);
        CALL    STBEOX                  ;
mma);   MOV     AL,B_SYMBOL             ;           if symbol <> comma then error(err_co
        CMP     AL,COMMA                ;
        MOV     CL,ECOMMA               ;
        JZ      J4024                   ;
        CALL    ERROR                   ;
J4024:
                EJECT
        CALL    SCAN                    ;           scan;
        CMP     AL,APPSYM               ;           case symbol of
        JNZ     L18350                  ;             append_symbol:
        MOV     CX,APPFILMD             ;               begin
        CALL    STOWRD                  ;                 store(append_file_mode);
        JMP     SCAN                    ;                 scan;
                                        ;                 end;
L18350: CMP     AL,RNDSYM               ;             random_symbol:
        JNZ     L18360                  ;               begin
        CALL    SCART                   ;                 scan; arith_expression;
        MOV     CX,RNDSYM               ;                 store(random_symbol);
        JMP     STOWRD                  ;               END;
L18360: CMP     AL,TYPE1KW              ;             OTHERWISE
        JZ      L18361                  ;               IF symbol<>type1_keyword THEN
        JMP     ERRSYN                  ;                 error(err_syntax)
L18361: MOV     AL,CL                   ;               ELSE BEGIN
        CMP     AL,READSTM              ;                 if sym_value = read_stmt then
        JNZ     L18363                  ;                 BEGIN
        MOV     CX,RDFMD                ;                   store(read_file_mode);
        JMPS    L18364                  ;                 END ELSE
                EJECT
```

```
e118363: CMP     AL,WRITESTM     ;                    if sym_value = write_stmt th
         JZ      JX010           ;
         JMP     ERRSYN          ;                    BEGIN
 JX010:  MOV     CX,WRFMD        ;                      store(write_file_mode);
 L18364: CALL    STOWRD          ;                    END ELSE
         JMP     SCAN            ;                      error(err_syntax);
                                 ;                  scan;
                                 ;                end;
                                 ;              end; (* case symbol of *)
                                 ;            end;

                                 ;    close_stmt:
 L18380: CALL    SCAN            ;      BEGIN
         Cmp     Al,Graph_Sym    ;        Scan;
         Jnz     L18385          ;        IF symbol = Graph_sym THEN BEGIN
         Mov     Bx,ProgPtr      ;          PutCore(ProgPtr-2,Graph_Close_Stm)
         Mov     Byte Ptr Minus2[Bx],Graph_Close_Stm;
         Jmp     Scan            ;          Scan; END
                                 ;        ELSE
 L18385: CMP     AL,ENDOFLIN     ;
         JZ      L18389          ;          if symbol <> end_of_line then
         CMP     AL,FILESYM      ;          BEGIN
         JNZ     J4025           ;            IF symbol=file_symbol THEN scan;
         CALL    SCAN            ;            arith_expression;
 J4025:  CALL    ARTEXP          ;            store(channel_eox);
         MOV     CH,CHANSYM      ;          END
         JMP     STBEOX          ;          ELSE
)L18389: MOV     BX,PROGPTR      ;            putcore(prog_ptr-1, close_all_stmt
         MOV     BYTE PTR minus2[BX],CLALLSTM;
         RET                     ;      END;
                                 ;
 L18390: CALL    SCAN            ;    new_stmt:
         CMP     AL,STRNGLIT     ;      begin
         JNZ     J4026           ;
scan;    CALL    SCAN            ;        scan; if symbol = string_literal then
 J4026:  RET                     ;      end;
                                 ;
                                 ;    drawto_stm, draw_stm,
 L18400: Mov     Cx,2            ;    moveto_stm, move_stm, Palette_Stmt :
         Call    Real_List       ;
         Ret                     ;      Real_list(2);
                                 ;
 L18410: Mov     Cx,4            ;    Window:
         Call    Real_List       ;
         Ret                     ;      Real_list(4);
                                 ;
 L18420: Mov     Cx,3            ;    Circle_Stmt:
         Call    Real_List       ;
         Ret                     ;      Real_List(3);
                                 ;
 L18430:                         ;    Locate_Stmt:
         Call    Scan            ;    begin
         Call    Var             ;      scan;
         Push    Ax              ;      var(typ);
         Mov     Ch,Comma        ;
         Call    StbEox          ;      stoeox(comma);
         Pop     Ax              ;
         Cmp     Al,StrngTyp     ;      if typ<>string_type then error(Illvar);
         Mov     Cl,Ellvr        ;
         Jne     L18436          ;
         Cmp     B_Symbol,Comma  ;      if symbol<>comma then
         Je      L18437          ;        error(comma);
         Mov     Cl,EComma       ;
```

```
L18436: Jmp     Error           ;
L18437: Mov     Cx,2            ;      real_var_list(true,2);
        Mov     Al,S_True       ;
        Jmp     Real_Var_List   ;    end;
                                ;
L18440:                         ;    Gsx:

        Mov     Cx,5            ;
Gsx_Parms:                      ;
        Push    Cx              ;   FOR i:=1 TO nr_reals DO
        Call    Scan            ;   BEGIN
        Cmp     Al,Identify     ;     scan;
        Je      Gsx_Real        ;
        Cmp     Al,StrngVar     ;     if symbol<>identify then error(IllVar);
        Jne     Gsx_Ill_Val     ;
Gsx_Real:                       ;     store_word(Symbol_value);
        Call    StScan          ;     Scan;
        Pop     Cx              ;
        Dec     Cx              ;
        JCxZ    Gsx_parms_end   ;
        Cmp     B_symbol,Comma  ;     End;
        Jz      Gsx_Parms       ;
        Mov     Cl,EComma       ;   END;
        Jmp     Error           ;
Gsx_parms_End:                  ;
        Ret                     ;
Gsx_Ill_Val:                    ;
        Mov     Cl,EIllVr       ;
        Jmp     Error           ;
                                ;
L18600: MOV     CL,EIMPL        ;    OTHERWISE
        JMP     ERROR           ;      error(err_notimpl);
                                ;  END;(*CASE stmt_type OF*)
                                ;END; (* procedure parse_statement *)
```