APPENDICES

APPENDIX A

COMPILER LISTING

```
1  006001 *******  PASS 1: RECOGNIZER AND REDUCED PROGRAM GENERATOR  *******+
2  006001
3  006001 CONST BLANK='  ': ASSERT='ASSERT': ASSERTION='ASSERTION':
4  006001       IDENTIFIER='IDENTIFIER': NUMBER='NUMBER':
5  006001       PR='P': SYM='S': DEFL='L': REFL='R':
6  006001       OGT='9': QUOTE='''': DLR='$': SEMI=';': ENDFILE='EOF':
7  006001       COLEQ=':=': EQL='=': LTRA='A': CMA=',': CLN=':': LP='(':
8  006001       RP=')': DRPR='))': PREQ=')=': ZER='0':
9  006001       SIMPLE='SIMPLE': ARRY='ARRAY': EXITS='EXIT': JUMPTO='JUMPTO':
10 006001       IFS='IF': EXITPT='EXITPOINT': HALTS='HALT': ASSIGNS='ASSIGN':
11 006001       POINTLABEL='POINTLABEL': POINTLBL='LEDWITH': ENTERS='ENTER':
12 006001       CASES='CASE': CASEJOIN1='CASEJOINPO': CASEJOIN2='INT':
13 006001       INITPROC1='INITIALPRO': INITPROC2='CEDURE=':
14 006001       CHARCONST='CHARCONST': MAXARRAY=77777777778;
15 006001       STMAX=100: STKLIM=6030B:
16 006001       MASK=-7777778:
17 006001       NOP=460000B: CODMAX=1000:
18 006001       JMPMAX=778:
19 006001 TYPE IDWORD=PACKED RECORD
20 006001                   STRNG: ALFA:
21 006001                   NUM: 1..60:
22 006001                   TYPEFNCTN: 1..20:
23 006001                   BOUND: -1..MAXARRAY:
24 006001                   PTR: ^ENDSTRNG END:
25 006001   *IDWORD: IDENTIFIERS STORED WITH THIS TYPE
26 006001         STRNG: FIRST 10 CHARS OF IDENT
27 006001         NUM: LENGTH OF IDENT
28 006001         TYPEFNCTN: 2: BOOLEAN
29 006001                    4: CHARACTER
30 006001                    16: INTEGER
31 006001                    19: PROCEDURE
32 006001         BOUND: -1: SIMPLE IDENT
33 006001         20: ARRAY BOUND FOR ARRAY VAR
34 006001             PROCEDURE VIRTUAL ADDR FOR PROCEDURE NAMES
35 006001         PTR: POINTER TO END OF IDENT IF NUM>10+
36 006001       TPTR=^TREENODE:
37 006001       STACKA=ARRAY[1..25] OF TPTR:
38 006001       STACKI=ARRAY[1..25] OF INTEGER:
39 006001       STACKC=ARRAY[1..25] OF INTEGER:
40 006001       CLSTK=ARRAY[1..25,1..2] OF INTEGER:
41 006001       ERRTYPE=SET OF 1..34:
42 006001       OPTYPE=SET OF 'A'..'Z':
43 006001       SCNT=0..25:
44 006001       SHRTINT=-3777778..+3777778:
45 006001        ADDRESS=0..1777778:
46 006001       TREEWORD=PACKED RECORD
47 006001                   FATHER: ALFA:
48 006001                   TY: 0..3:
49 006001                   INX:  SHRTINT:
50 006001                   LSON,RSON: TPTR END:
51 006001 VAR ENDSTRNG: CLASS 1000 OF ARRAY[1..5] OF ALFA:
52 017612         *ENDSTRNG: FOR END OF IDENT>10 CHARS LONG+
53 017612       TREENODE: CLASS 1000 OF TREEWORD:
54 023533         *TREENODE: FOR EXPRESSION TREES AND REDUCED PROGRAM TREES+
55 023533       SYMTAB: ARRAY[1..500] OF IDWORD:
56 025503       CONSTANT: ARRAY[1..250] OF INTEGER: C: INTEGER:
57 026076       VPROG: FILE OF TREEWORD:
58 027100       SENTREE: ARRAY[0..249] OF TREEWORD:
59 030064       REFPROCSET: ARRAY[1..150] OF IDWORD:
```

.NUC 1 .NUC 2 .NUC 3 .NUC 4 .NUC 5 .NUC 6 .NUC 7 .NUC 8 .NUC 9 .NUC 10 .NUC 11 .NUC 12 .NUC 13 .NUC 14 .NUC 15 .NUC 16 .NUC 17 .NUC 18 .NUC 19 .NUC 20 .NUC 21 .NUC 22 .NUC 23 .NUC 24 .NUC 25 .NUC 26 .NUC 27 .NUC 28 .NUC 29 .NUC 30 .NUC 31 .NUC 32 .NUC 33 .NUC 34 .NUC 35 .NUC 36 .NUC 37 .NUC 38 .NUC 39 .NUC 40 .NUC 41 .NUC 42 .NUC 43 .NUC 44 .NUC 45 .NUC 46 .NUC 47 .NUC 48 .NUC 49 .NUC 50 .NUC 51 .NUC 52 .NUC 53 .NUC 54 .NUC 55 .NUC 56 .NUC 57 .NUC 58 .NUC 59

```
 60  030540       DEFCASELAB: ARRAY[1..25] OF CLSTK;                                  .NUC    60
 61  033102     CASEJOIN: ARRAY[1..10] OF STACKC;                                     .NUC    61
 62  033474     DEFLABSET: ARRAY[1..200] OF IDWORD;                                   .NUC    62
 63  034314     REFLABSET: ARRAY[1..100] OF IDWORD;                                   .NUC    63
 64  034624        *TABLES FOR IDENTIFIERS, CONSTANTS, REDUCED PROGRAM TREES          .NUC    64
 65  034624           CASE LABELS, LABELS*                                            .NUC    65
 66  034624     SAVEPTR: *TREENODE;                                                   .NUC    66
 67  034625     LEFTSIDE: TREEWORD;                                                   .NUC    67
 68  034627     LFTARYNAME,IDNAME,TOKENSTRNG,PROCNAME: IDWORD;                        .NUC    68
 69  034637     ARRAYNAME: IDWORD; NUMVAL: INTEGER;                                   .NUC    69
 70  034642     ANDTYPE, BNRYADTYPE, EXPTYPE, MULTYPE, NOTTYPE, PRMRYTYPE, RELTYPE,.NUC    70
 71  034642     UNRYADTYPE, CASEPOINT, IFELSEPNT, IFPOINT, WHILEPOINT: STACKI;        .NUC    71
 72  035316     ANDSTRNG, BNRYADSTNG, CASEXPR, EXPSTRNG, IFEXPR, MULSTRNG,            .NUC    72
 73  035316     NOTSTRNG, PRMRYSTRNG, RELSTRNG, UNRYADSTNG, WHILEXPR: STACKA;         .NUC    73
 74  035741        *TYPE STACKS AND EXPRESSION STACKS: FOR NESTED EXPRESSIONS,      .NUC    74
 75  035741           CASE STATEMENTS, WHILE STATEMENTS, ETC*                         .NUC    75
 76  035741     S,P,RPR,DC,POINT,COLCNT,TYPES,LEFTYPE: INTEGER;                       .NUC    76
 77  035751     TNUM: 1..63;                                                          .NUC    77
 78  035752        *TNUM: HOLDS REFERENCE NUMBER OF TOKEN                             .NUC    78
 79  035752              1-27: INDICATES INDEX OF RESERVED WORD IN RESERVED           .NUC    79
 80  035752                    WORD TABLE                                             .NUC    80
 81  035752              30: IDENTIFIER TOKEN                                         .NUC    81
 82  035752              31: NUMBER TOKEN                                             .NUC    82
 83  035752              32: CHARACTER CONSTANT TOKEN                                 .NUC    83
 84  035752              33: EOF TOKEN                                                .NUC    84
 85  035752              34: = := = TOKEN                                             .NUC    85
 86  035752              37-63: CORRESPONDING DECIMAL EQUIVALENT OF DISPLAY CODE      .NUC    86
 87  035752                     FOR SPECIAL CHARACTERS*                               .NUC    87
 88  035752     IFEPT,IFP,AN,BN,CS,E*,IFE,MUL,NT,PM,REL,UN,WH: 0..25;                 .NUC    88
 89  035767     TOKEN,SAVE: ALFA; D,R: 1..100;                                        .NUC    89
 90  035773     RESWRDS: ARRAY[1..27] OF ALFA; CARD: ARRAY[1..80] OF CHAR;            .NUC    90
 91  036146        *RESERVED WORD TABLE*                                              .NUC    91
 92  036146        *CHARACTER ARRAY TO HOLD CONTENTS OF CARD CURRENTLY                 .NUC    92
 93  036146           BEING SCANNED*                                                  .NUC    93
 94  036146     SIXTEEN, FOUR, TWO: INTEGER;                                          .NUC    94
 95  036151     CHECK: INTEGER;                                                       .NUC    95
 96  036152     PFL,UFL,NFL,BFL,RFL,NFL,AFL,EFL: INTEGER;                             .NUC    96
 97  036162     SCTR: -1..499;                                                        .NUC    97
 98  036163     KEEPRTN: STACKC; CJ,KR: 0..25;                                        .NUC    98
 99  036216     LTEMP:INTEGER;                                                        .NUC    99
100  036217     ERRSET: ERRTYPE;                                                      .NUC   100
101  036220     OPFLAG: OPTYPE;                                                       .NUC   101
102  036221       *OPFLAG: OPTION FLAG.                                               .NUC   102
103  036221          THE APPROPRIATE BIT IS SET FOR A CERTAIN CHARACTER.              .NUC   103
104  036221          THE CHARACTERS CURRENTLY ACTIVATED ARE:                          .NUC   104
105  036221          C,S,P,V,M.  ANY ADDITIONAL CHARACTERS MAY BE USED                .NUC   105
106  036221          AS NEEDED*                                                       .NUC   106
107  036221     ARRAYSIZE: INTEGER;                                                   .NUC   107
108  036222     SENFLAG: 0..1;                                                        .NUC   108
109  036223     SYMSIZE,CODADDR,STSYM,STPROC,PCNTR,ADDR: SHRTINT;                     .NUC   109
110  036231     LT, LE, GT, GE, EQ, NE, NOT, AND, OR: ALFA;                           .NUC   110
111  036242     COLON,SEMI,COMMA,PLUS,MINUS,TIMES,SLASH,EXPO,                         .NUC   111
112  036242     LEFTBRACK,RIGHTBRACK,LEFTPAREN,RIGHTPAREN: ALFA;                      .NUC   112
113  036256     STCON: SHRTINT;                                                       .NUC   113
114  036257     BSRCH,TCODE: ADDRESS;                                                 .NUC   114
115  036261     READCH,WRITECH: ADDRESS;                                              .NUC   115
116  036263     IC: ADDRESS;                                                          .NUC   116
117  036264     PC: INTEGER;                                                          .NUC   117
118  036265     JMPTAB: ARRAY[0..JMPMAX] OF INTEGER;                                  .NUC   118
119  036365     JMPIX: SHRTINT;                                                       .NUC   119
120  036366     CODE: ARRAY[0..CODMAX] OF INTEGER;                                    .NUC   120
121  040337     IT: INTEGER;                                                          .NUC   121
```

```
122  040340      CA,CP: SHRTINT; BUF: INTEGER;                                                    .NUC      122
123  040343      PASCLGO[OUT]: FILE OF INTEGER;                                                   .NUC      123
124  041344      LASTOP,LASTI: SHRTINT;                                                           .NUC      124
125  041346      PRCODE: BOOLEAN;                                                                 .NUC      125
126  041347      JB1,JB2,JX1,JX2,RL,FS,R1: ARRAY[0..7] OF CHAR;                                   .NUC      126
127  041437      SH: ARRAY[12..23] OF CHAR;                                                       .NUC      127
128  041453 VALUE CONSTANT=(0,1,248*0);                                                           .NUC      128
129  041453      C=3; COLCNT=0;                                                                   .NUC      129
130  041453      IFEPT=0; IFP=0; AN=0; BN=0; CS=0; EX=0; IFE=0; MUL=0; NT=0;                      .NUC      130
131  041453      PM=0; REL=0; UN=0; WH=0;                                                         .NUC      131
132  041453      RESWRDS = (ΞARRAYΞ,ΞBOOLEANΞ,ΞCASEΞ,ΞCHARACTERΞ,ΞDOΞ,ΞELIHWΞ,                    .NUC      132
133  041453      ΞELSEΞ,ΞENTERΞ,ΞESACΞ,ΞEXITΞ,ΞFALSEΞ,ΞFIΞ,ΞGOΞ,ΞHALTΞ,ΞIFΞ,                    .NUC      133
134  041453      ΞINTEGERΞ,ΞNOPΞ,ΞOFΞ,ΞPROCEDUREΞ,ΞREADΞ,ΞRETURNΞ,ΞSTARTΞ,ΞTHENΞ,                .NUC      134
135  041453      ΞTOΞ,ΞTRUEΞ,ΞWHILEΞ,ΞWRITEΞ);                                                    .NUC      135
136  041453      STATEFN=16; FOUR=4; TWO=2;                                                       .NUC      136
137  041453      CHECK=0;                                                                         .NUC      137
138  041453      PFL=0; UFL=0; MFL=0; BFL=0; RFL=0; NFL=0; AFL=0;                                 .NUC      138
139  041453      EFL=0; SCTR=-1;                                                                  .NUC      139
140  041453      LTEMP=0;                                                                         .NUC      140
141  041453      ARRAYSIZE=0;                                                                     .NUC      141
142  041453      SENFLAG=0;                                                                       .NUC      142
143  041453      LT=Ξ<          Ξ; LE=Ξ≤          Ξ; GT=Ξ>          Ξ;                          .NUC      143
144  041453      GE=Ξ≥          Ξ; EQ=Ξ=          Ξ; NE=Ξ≠          Ξ;                          .NUC      144
145  041453      NOT=Ξ¬          Ξ; AND=Ξ∧          Ξ; OR=Ξ∨          Ξ;                        .NUC      145
146  041453      COLON=Ξ:          Ξ; SEMI=Ξ;          Ξ;                                     .NUC      146
147  041453      COMMA=Ξ,          Ξ;                                                           .NUC      147
148  041453      PLUS=Ξ+          Ξ; MINUS=Ξ-          Ξ;                                     .NUC      148
149  041453      TIMES=Ξ*          Ξ; SLASH=Ξ/          Ξ; EXPO=Ξ↑          Ξ;               .NUC      149
150  041453      LEFTBRACK=Ξ[          Ξ;  RIGHTBRACK=Ξ]          Ξ;                          .NUC      150
151  041453      LEFTPAREN=Ξ(          Ξ; RIGHTPAREN=Ξ)          Ξ;                          .NUC      151
152  041453      PC=0;                                                                            .NUC      152
153  041453      JH1 = (ΞPΞ,ΞRΞ,ΞJΞ,Ξ Ξ,ΞEΞ,ΞNΞ,ΞGΞ,ΞLΞ);                                      .NUC      153
154  041453      JH2 = (ΞSΞ,ΞJΞ,ΞPΞ,Ξ Ξ,ΞOΞ,ΞEΞ,ΞEΞ,ΞTΞ);                                      .NUC      154
155  041453      JX1 = (ΞZΞ,ΞNΞ,ΞPΞ,ΞNΞ,ΞIΞ,ΞOΞ,ΞDΞ,ΞIΞ);                                      .NUC      155
156  041453      JX2 = (ΞRΞ,ΞZΞ,ΞLΞ,ΞGΞ,ΞRΞ,ΞRΞ,ΞFΞ,ΞDΞ);                                      .NUC      156
157  041453      RL  = (Ξ Ξ,ΞAΞ,ΞVΞ,Ξ-Ξ,Ξ=Ξ,Ξ≠Ξ,Ξ≥Ξ,Ξ<Ξ);                                      .NUC      157
158  041453      FS  = (ΞFΞ,ΞRΞ,ΞDΞ,Ξ Ξ,Ξ Ξ,ΞAΞ,ΞBΞ,ΞXΞ);                                      .NUC      158
159  041453      R1  = (ΞAΞ,ΞBΞ,ΞXΞ,ΞXΞ,ΞAΞ,ΞAΞ,ΞBΞ,ΞBΞ);                                      .NUC      159
160  041453      SH  = (ΞFΞ,ΞDΞ,ΞRΞ,ΞIΞ,ΞLΞ,ΞAΞ,ΞLΞ,ΞAΞ,ΞNΞ,ΞZΞ,ΞUΞ,ΞPΞ);                          .NUC      160
161  041453 PROCEDURE PRIMARY; FORWARD; PROCEDURE BODY; FORWARD;                                   .NUC      161
162  041577 PROCEDURE DUMPSYM(VAR IX1,IX2: INTEGER);                                              .NUC      162
163  041577   *DUMP SYMBOL TABLE WITH VIRTUAL ADDRESSES*                                         .NUC      163
164  041577      VAR I,J,K: INTEGER;                                                            .NUC      164
165  041577      BEGIN WRITE(EOL);                                                              .NUC      165   *1
166  041604         FOR I:=IX1 TO IX2-1 DO BEGIN K:=SYMTAB[I].NUM;                              .NUC      166   *2
167  041617            IF K>10 THEN BEGIN J:=K MOD 10; IF J=0 THEN J:=(K DIV 10)-1              .NUC      167   *3
168  041631               ELSE J:=K DIV 10 END ELSE J:=0;                                      .NUC      168   -3
169  041641            CASE J OF                                                               .NUC      169   *C
170  041646               0: WRITE(BLANK,PCNTR:6,BLANK:10,SYMTAB[I].STRNG,EOL);                .NUC      170
171  041665               1: WRITE(BLANK,PCNTR:6,BLANK:10,SYMTAB[I].STRNG,                     .NUC      171
172  041701                  SYMTAB[I].PTR*[1],EOL);                                           .NUC      172
173  041713               2: WRITE(BLANK,PCNTR:6,BLANK:10,SYMTAB[I].STRNG,                     .NUC      173
174  041727                  SYMTAB[I].PTR*[1],SYMTAB[I].PTR*[2],EOL);                         .NUC      174
175  041750               3: WRITE(BLANK,PCNTR:6,BLANK:10,SYMTAB[I].STRNG,                     .NUC      175
176  041764                  SYMTAB[I].PTR*[1],SYMTAB[I].PTR*[2],                              .NUC      176
177  042002                  SYMTAB[I].PTR*[3],EOL);                                           .NUC      177
178  042014               4: WRITE(BLANK,PCNTR:6,BLANK:10,SYMTAB[I].STRNG,                     .NUC      178
179  042030                  SYMTAB[I].PTR*[1],SYMTAB[I].PTR*[2],                              .NUC      179
180  042046                  SYMTAB[I].PTR*[3],SYMTAB[I].PTR*[4],EOL);                         .NUC      180
181  042067               5: WRITE(BLANK,PCNTR:6,BLANK:10,SYMTAB[I].STRNG,                     .NUC      181
182  042103                  SYMTAB[I].PTR*[1],SYMTAB[I].PTR*[2],                              .NUC      182
183  042121                  SYMTAB[I].PTR*[3],SYMTAB[I].PTR*[4],                              .NUC      183
```

```
184  042137                SYMTAB[I].PTR+[5],EOL)                          .NUC   184
185  042150             END;                                               .NUC   185    -C
186  042157             PCNTR:=PCNTR+1                                     .NUC   186
187  042160          END                                                   .NUC   187    -2
188  042161       END;                                                     .NUC   188    -1
189  042165 PROCEDURE DUMPCONS;                                           .NUC   189
190  042165   *DUMP CONSTANT TABLE WITH VIRTUAL ADDRESSES+                .NUC   190
191  042165       VAR I:INTEGER;                                          .NUC   191
192  042165       BEGIN FOR I:=1 TO C-1 DO BEGIN                          .NUC   192    +1+2
193  042175          WRITE(BLANK,PCNTR:6+CONSTANT[I],EOL);                .NUC   193
194  042211          PCNTR:=PCNTR+1 END                                   .NUC   194    -2
195  042213       END;                                                    .NUC   195    -1
196  042217 PROCEDURE DUMPTREE;                                           .NUC   196
197  042217   *DUMP THE REDUCED PROGRAM.  OPTION FLAG HAS V BIT SET+      .NUC   197
198  042217       CONST L=ΞLΞ; R=ΞRΞ;                                     .NUC   198
199  042217       VAR I: INTEGER;                                         .NUC   199
200  042217       PROCEDURE TRAVERSE(TREE: TPTR; LVL: INTEGER; POS: CHAR);.NUC   200
201  042217          BEGIN IF TREE≠NIL THEN                               .NUC   201    +1
202  042225             BEGIN WITH TREE+ DO BEGIN                         .NUC   202    +2+3
203  042226             WRITE(BLANK:7+POS,BLANK:3*LVL+2,FATHER,EOL);      .NUC   203
204  042243             IF LSON≠NIL THEN BEGIN TRAVERSE(LSON,LVL+1,L);    .NUC   204    +4
205  042255             TRAVERSE(RSON,LVL+1,R) END                        .NUC   205    -4
206  042264             END END                                           .NUC   206    -3-2
207  042264          END;                                                 .NUC   207    -1
208  042266       BEGIN WRITE(ΞIΞ,EOL); I:=0;                             .NUC   208    +1
209  042276          GET(VPROG);                                          .NUC   209
210  042300          REPEAT WRITE(ZER,I:4,BLANK:5,VPROG+.FATHER,EOL);     .NUC   210    +R
211  042313             TRAVERSE(VPROG+.LSON,1,L);                        .NUC   211
212  042321             TRAVERSE(VPROG+.RSON,1,R);                        .NUC   212
213  042327             I:=I+1; GET(VPROG)                                .NUC   213
214  042333          UNTIL EOF(VPROG);                                    .NUC   214    -R
215  042334       END;                                                    .NUC   215    -1
216  042336 FUNCTION MEMBER(VAR ID1: IDWORD;  SETN: CHAR;                .NUC   216
217  042336       VAR INDEX: INTEGER): BOOLEAN;                           .NUC   217
218  042336   *DETERMINE IF AN IDENTIFIER IS A MEMBER OF A SET            .NUC   218
219  042336       SYM: CHECK IN SYMBOL TABLE                              .NUC   219
220  042336       DEFL: CHECK IN DEFINED LABEL TABLE                      .NUC   220
221  042336       REFL: CHECK IN REFERENCED LABEL TABLE+                  .NUC   221
222  042336       VAR I,J,T,TMPLNGTH: INTEGER;                            .NUC   222
223  042336       FUNCTION IDENTEQ(VAR ID2: IDWORD): BOOLEAN;             .NUC   223
224  042336          BEGIN IDENTEQ:=FALSE;                                .NUC   224    +1
225  042342             IF (ID1.STRNG=ID2.STRNG) ∧ (ID1.NUM=ID2.NUM)      .NUC   225
226  042352             THEN BEGIN                                        .NUC   226    +2
227  042354             IF ID1.NUM>10 THEN BEGIN                          .NUC   227    +3
228  042357             T:=ID1.NUM MOD 10;                                .NUC   228
229  042364             IF T=0 THEN T:=(ID1.NUM DIV 10)-1 ELSE            .NUC   229
230  042374             T:=ID1.NUM DIV 10;                                .NUC   230
231  042402             J:=1;                                             .NUC   231
232  042404             WHILE (ID1.PTR+[J]=ID2.PTR+[J]) ∧ (J≤T) DO J:=J+1;.NUC   232
233  042431             IF J>T THEN IDENTEQ:=TRUE END ELSE                .NUC   233    -3
234  042436             IDENTEQ:=TRUE END;                                .NUC   234    -2
235  042437             IF ID2.BOUND>0 THEN TMPLNGTH:=TMPLNGTH+ID2.BOUND; .NUC   235
236  042445             TMPLNGTH:=TMPLNGTH+1;                             .NUC   236
237  042450          END;                                                 .NUC   237    -1
238  042452       BEGIN MEMBER:=FALSE; I:=1; IF INDEX>1 THEN BEGIN        .NUC   238    +1+2
239  042462          TMPLNGTH:=0;                                         .NUC   239
240  042463          CASE SETN OF                                         .NUC   240    +C
241  042467             SYM: REPEAT IF IDENTEQ(SYMTAB[I]) THEN BEGIN      .NUC   241    +R+3
242  042477                MEMBER:=TRUE; ID1.TYPEFNCTN:=SYMTAB[I].TYPEFNCTN;.NUC  242
243  042510                ID1.BOUND:=SYMTAB[I].BOUND;                    .NUC   243
244  042520                ADDR:=TMPLNGTH-1;                              .NUC   244
245  042521                I:=INDEX END ELSE I:=I+1                       .NUC   245    -3
```

```
246  042524              UNTIL I≥INDEX;
247  042530            DEFL: REPEAT IF IDENTEQ(DEFLABSET[I]) THEN BEGIN
248  042540               MEMBER:=TRUE; ID1.BOUND:=DEFLABSET[I].BOUND;
249  042551               I:=INDEX END ELSE I:=I+1
250  042554               UNTIL I≥INDEX;
251  042560            REFL: BEGIN I:=R-1;
252  042562               WHILE I>0 DO
253  042564               BEGIN IF IDENTEQ(REFLABSET[I]) THEN BEGIN
254  042573               MEMBER:=TRUE; ID1.BOUND:=REFLABSET[I].BOUND;
255  042604               PCNTR:=I;
256  042605               I:=0 END ELSE I:=I-1 END END
257  042611         END END
258  042622      END;
259  042624 FUNCTION MEMBER1(VAR INTR: INTEGER): BOOLEAN;
260  042624   *DETERMINE IF CASE LABEL IN CASE LABEL TABLE*
261  042624      VAR I: INTEGER;
262  042624      BEGIN MEMBER1:=FALSE; I:=1;
263  042631         WHILE (INTR≠DEFCASELAB[CS][I+1]) ∧ (I<DC)
264  042647         DO I:=I+1;
265  042653         IF I<DC THEN MEMBER1:=TRUE
266  042655      END;
267  042660 FUNCTION ALFI(PT: INTEGER): ALFA;
268  042660   *CHANGE INTEGER CONSTANT TO ALFA EQUIVALENT*
269  042660      VAR TMP: ARRAY[1..10] OF CHAR; I,J: INTEGER;
270  042660         TMP1: ALFA;
271  042660      BEGIN FOR I:=1 TO 10 DO TMP[I]:=BLANK;
272  042676         J:=PT; I:=10;
273  042701         REPEAT TMP[I]:=CHR((J MOD 10) + 27);
274  042711            J:=J DIV 10; I:=I-1
275  042715         UNTIL J=0;
276  042720         PACK(TMP,1,TMP1);
277  042730         ALFI:=TMP1
278  042730      END;
279  042733 FUNCTION INTI(WD: ALFA): INTEGER;
280  042733   *CHANGE ALFA VALUE TO INTEGER EQUIVALENT*
281  042733      VAR TMPCH: ARRAY[1..10] OF CHAR; I,J: INTEGER;
282  042733      BEGIN UNPACK(WD,TMPCH,1);
283  042746         J:=0; I:=1;
284  042750         WHILE (TMPCH[I]=BLANK) ∧ (I≤10) DO I:=I+1;
285  042765         REPEAT J:=10*J+(INT(TMPCH[I])-27); I:=I+1
286  042775         UNTIL I>10;
287  043001         INTI:=J;
288  043002      END;
289  043004 FUNCTION SUBSET(SET1: CHAR): BOOLEAN;
290  043004   *DETERMINE IF ONE SET SUBSET OF ANOTHER
291  043004      PR: CHECK IF REFERENCED PROCEDURE SET IF SUBSET OF
292  043004          ACTUAL PROCEDURE SET
293  043004      REFL: CHECK IF REFERENCED ALBEL SET SUBSET OF ACTUAL LABEL SET*
294  043004      VAR I,J: INTEGER;
295  043004      BEGIN SUBSET:=TRUE;
296  043010         CASE SET1 OF
297  043015            PR: FOR I:=1 TO RPR-1 DO
298  043022               IF ~MEMBER(REFPROCSET[I],SYM,P) THEN BEGIN
299  043033               SUBSET:=FALSE; I:=RPR END;
300  043041            REFL: FOR I:=1 TO R-1 DO
301  043046               BEGIN ADDR:=REFLABSET[I].BOUND;
302  043053               IF ~MEMBER(REFLABSET[I],DEFL,D) THEN BEGIN
303  043065               SUBSET:=FALSE; I:=R END
304  043067               ELSE BEGIN J:=REFLABSET[I].BOUND;
305  043076               SENTREE[ADDR].LSONF.FATHER:=ALFI(J);
306  043110               END END
307  043110         END
```

```
.NUC    246    -R
.NUC    247    *R*3
.NUC    248
.NUC    249    -3
.NUC    250    -R
.NUC    251    *3
.NUC    252
.NUC    253    *4*5
.NUC    254
.NUC    255
.NUC    256    -5-4-3
.NUC    257    -C-2
.NUC    258    -1
.NUC    259
.NUC    260
.NUC    261
.NUC    262    *1
.NUC    263
.NUC    264
.NUC    265
.NUC    266    -1
.NUC    267
.NUC    268
.NUC    269
.NUC    270
.NUC    271    *1
.NUC    272
.NUC    273    *R
.NUC    274
.NUC    275    -R
.NUC    276
.NUC    277
.NUC    278    -1
.NUC    279
.NUC    280
.NUC    281
.NUC    282    *1
.NUC    283
.NUC    284
.NUC    285    *R
.NUC    286    -R
.NUC    287
.NUC    288    -1
.NUC    289
.NUC    290
.NUC    291
.NUC    292
.NUC    293
.NUC    294
.NUC    295    *1
.NUC    296    *C
.NUC    297
.NUC    298    *2
.NUC    299    -2
.NUC    300
.NUC    301    *2
.NUC    302    *3
.NUC    303    -3
.NUC    304    *3
.NUC    305
.NUC    306    -3-2
.NUC    307    -C
```

```
308  043117      END;
309  043121 PROCEDURE PUSH(VAR STACK:STACKI; INSRT:INTEGER; VAR I:INTEGER);
310  043121   *PUSH TYPE STACKS*
311  043121      BEGIN I:=I+1; IF I>10 THEN BEGIN I:=10;
312  043132         WRITE(E      STACKE,E OVERFLOW E,EOL) END;
313  043140         STACK[I]:=INSRT
314  043145      END;
315  043153 FUNCTION INCONS(NUM:INTEGER):BOOLEAN;
316  043153   *DETERMINE IF CONSTANT IN CONSTANT TABLE*
317  043153      VAR I: INTEGER;
318  043153      BEGIN I:=1; INCONS:=FALSE;
319  043160         WHILE (CONSTANT[I]#NUM) ^ (I<C) DO I:=I+1;
320  043174         IF I<C THEN BEGIN INCONS:=TRUE;
321  043177            ADDR:=I END;
322  043201      END;
323  043203 FUNCTION RESERVED(RWORD: ALFA):BOOLEAN;
324  043203   *DETERMINE IF IDENTIFIER IS A RESERVED WORD*
325  043203      VAR F,L,M: INTEGER;
326  043203      BEGIN F:=1; L:=27;
327  043210         RESERVED:=FALSE;
328  043211         IF TOKENSTRNG.NUM<10 THEN BEGIN
329  043214         WHILE F<L DO BEGIN M:=(F+L) DIV 2;
330  043221            IF RWORD<RESWRDS[M] THEN L:=M-1 ELSE
331  043231            IF RWORD=RESWRDS[M] THEN BEGIN RESERVED:=TRUE; TNUM:=M;
332  043241            F:=L+1 END ELSE F:=M+1 END END
333  043247      END;
334  043251 PROCEDURE SCANNER;
335  043251   *SCAN CARD AND PICK UP NEXT TOKEN FOR PARSER
336  043251      TOKEN: IDENTIFIER, NUMBER,CHARCONST,SPECIAL CHARACTERS
337  043251      TOKENSTRNG: ACTUAL STRING OF CHARACTERS SCANNED*
338  043251      VAR WORD: ARRAY[1..10] OF CHAR;
339  043251         I,J,K,WCNT,WCNT1,TEMP,TEMP1,MFLAG: INTEGER;
340  043251      PROCEDURE GETCHAR;     *GET NEXT CHARACTER*
341  043251         BEGIN IF (COLCNT<0) v (COLCNT>80) THEN BEGIN
342  043261            I:=1; COLCNT:=0; OUTPUT^:=BLANK; PUT(OUTPUT);
343  043267            REPEAT GET(INPUT);
344  043271               IF EOF(INPUT) THEN GOTO 1;
345  043273               IF I>80 THEN I:=80;
346  043277               CARD[I]:=INPUT^;
347  043305               OUTPUT^:=CARD[I]; PUT(OUTPUT); I:=I+1
348  043315               UNTIL INPUT^=EOL;
349  043320            END;
350  043320            COLCNT:=COLCNT+1;
351  043322         1: END;
352  043324      PROCEDURE FILLER;      *SPAN BLANKS*
353  043324         BEGIN WHILE CARD[COLCNT]=BLANK DO GETCHAR;
354  043340            IF CARD[COLCNT]=EOL THEN BEGIN COLCNT:=0; GETCHAR;
355  043347               FILLER END ELSE BEGIN
356  043352            IF CARD[COLCNT]=DLR THEN BEGIN GETCHAR;
357  043361            IF CARD[COLCNT]=QUOTE THEN BEGIN GETCHAR;
358  043370            IF CARD[COLCNT]=E+E THEN BEGIN
359  043376            GETCHAR;
360  043377            WHILE (CARD[COLCNT]>EOL) ^ (CARD[COLCNT]<ZER) DO BEGIN
361  043413            OPFLAG:=OPFLAG v (CARD[COLCNT]); GETCHAR;
362  043423            END END END;
363  043424            WHILE (CARD[COLCNT]#DLR) ^ (~EOF(INPUT)) DO GETCHAR;
364  043437            IF ~EOF(INPUT) THEN BEGIN GETCHAR;
365  043442               FILLER END END END
366  043444         END;
367  043446      BEGIN J:=1; FOR I:=1 TO 10 DO WORD[I]:=BLANK;
368  043465         TOKENSTRNG.PTR:=NIL;
369  043470         IF COLCNT=0 THEN GETCHAR;
```

```
.NUC    308    -1
.NUC    309
.NUC    310
.NUC    311    +1+2
.NUC    312    -2
.NUC    313
.NUC    314    -1
.NUC    315
.NUC    316
.NUC    317
.NUC    318    +1
.NUC    319
.NUC    320    +2
.NUC    321    -2
.NUC    322    -1
.NUC    323
.NUC    324
.NUC    325
.NUC    326    +1
.NUC    327
.NUC    328    +2
.NUC    329    +3
.NUC    330
.NUC    331    +4
.NUC    332    -4-3-2
.NUC    333    -1
.NUC    334
.NUC    335
.NUC    336
.NUC    337
.NUC    338
.NUC    339
.NUC    340
.NUC    341    +1+2
.NUC    342
.NUC    343    +R
.NUC    344
.NUC    345
.NUC    346
.NUC    347
.NUC    348    -R
.NUC    349    -2
.NUC    350
.NUC    351    -1
.NUC    352
.NUC    353    +1
.NUC    354    +2
.NUC    355    -2+2
.NUC    356    +3
.NUC    357    +4
.NUC    358    +5
.NUC    359
.NUC    360    +6
.NUC    361
.NUC    362    -6-5-4
.NUC    363
.NUC    364    +4
.NUC    365    -4-3-2
.NUC    366    -1
.NUC    367    +1
.NUC    368
.NUC    369
```

```
370  043472        FILLER;                                                                  .NUC    370
371  043474        IF EOF(INPUT) THEN BEGIN TOKEN:=ENDFILE; TNUM:=33;                       .NUC    371    +2
372  043500           TOKENSTRNG.STRNG:=TOKEN; TOKENSTRNG.NUM:=3 END ELSE                   .NUC    372    -2
373  043505        IF CARD[COLCNT]≥ZER THEN BEGIN IF CARD[COLCNT]>DGT THEN                  .NUC    373    +2
374  043520  *SPECIAL CHARACTER TOKENS*                                                     .NUC    374
375  043520        BEGIN WORD[J]:=CARD[COLCNT]; MFLAG:=1;                                   .NUC    375    +3
376  043532           IF CARD[COLCNT]=QUOTE THEN                                            .NUC    376
377  043540              BEGIN TOKEN:=CHARCONST; GETCHAR;                                   .NUC    377    +4
378  043542              ADDR:=INT(CARD[COLCNT]); TNUM:=32; IF EOF(INPUT)                   .NUC    378
379  043551                 THEN BEGIN TOKEN:=ENDFILE; TOKENSTRNG.STRNG:=TOKEN;             .NUC    379    +5
380  043554                 TNUM:=33;                                                       .NUC    380
381  043555                 TOKENSTRNG.NUM:=3 END ELSE BEGIN WORD[2]:=CARD[COLCNT];         .NUC    381    -5+5
382  043566                 PACK(WORD,1,TOKENSTRNG.STRNG); TOKENSTRNG.NUM:=2;               .NUC    382
383  043601                 GETCHAR END; MFLAG:=0 END ELSE                                  .NUC    383    -5-4
384  043604           IF CARD[COLCNT]=CLN THEN                                              .NUC    384
385  043612              BEGIN GETCHAR; IF CARD[COLCNT]=EQL THEN BEGIN TNUM:=34;            .NUC    385    +4+5
386  043622                 TOKEN:=COLEQ; TOKENSTRNG.NUM:=2; GETCHAR END ELSE               .NUC    386    -5
387  043630                 BEGIN PACK(WORD,1,TOKEN); TOKENSTRNG.NUM:=1;                    .NUC    387    +5
388  043643                 TNUM:=INT(WORD[1]) END;                                         .NUC    388    -5
389  043644                 TOKENSTRNG.STRNG:=TOKEN; MFLAG:=0 END;                          .NUC    389    -4
390  043647              IF MFLAG≠0 THEN BEGIN PACK(WORD,1,TOKEN);                          .NUC    390    +4
391  043661                 TNUM:=INT(WORD[1]);                                             .NUC    391
392  043662                 TOKENSTRNG.STRNG:=TOKEN; TOKENSTRNG.NUM:=1; GETCHAR             .NUC    392
393  043667              END END ELSE                                                       .NUC    393    -4-3
394  043671  *NUMBER TOKEN*                                                               .NUC    394
395  043671           BEGIN TOKEN:=NUMBER; TNUM:=31; NUMVAL:=0;                          .NUC    395    +3
396  043675              WHILE CARD[COLCNT]=ZER DO GETCHAR;                              .NUC    396
397  043705           IF (CARD[COLCNT]≥ZER) ∧ (CARD[COLCNT]≤DGT) THEN BEGIN              .NUC    397    +4
398  043721              WCNT:=COLCNT; GETCHAR;                                          .NUC    398
399  043724           WHILE (CARD[COLCNT]≥ZER) ∧ (CARD[COLCNT]≤DGT) DO GETCHAR;          .NUC    399
400  043742              TEMP:=COLCNT-WCNT; TOKENSTRNG.NUM:=TEMP;                        .NUC    400
401  043750              IF TEMP>16 THEN BEGIN WRITE(Ξ ERRORΞ,EOL);                      .NUC    401    +5
402  043756                 WCNT1:=WCNT+15 END ELSE WCNT1:=COLCNT-1;                     .NUC    402    -5
403  043763              FOR I:=WCNT TO WCNT1 DO                                         .NUC    403
404  043767                 NUMVAL:=10*NUMVAL+INT(CARD[I])-27 END;                       .NUC    404    -4
405  044000              IF ¬INCONS(NUMVAL) THEN BEGIN CONSTANT[C]:=NUMVAL;              .NUC    405    +4
406  044012              ADDR:=C;                                                        .NUC    406
407  044013              C:=C+1; IF C>250 THEN                                           .NUC    407
408  044017              WRITE(Ξ OVERFLOW ΞΞOF CONSTANΞΞT TABLEΞ,EOL) END;               .NUC    408    -4
409  044027           END END ELSE                                                       .NUC    409    -3-2
410  044030  *IDENTIFIER TOKEN*                                                         .NUC    410
411  044030        BEGIN TOKEN:=IDENTIFIER; TNUM:=30; WCNT:=COLCNT; GETCHAR;            .NUC    411    +2
412  044035           WHILE (CARD[COLCNT]>EOL) ∧ (CARD[COLCNT]≤DGT) DO GETCHAR;         .NUC    412
413  044053           TEMP:=COLCNT-WCNT;                                                .NUC    413
414  044055           IF TEMP>10 THEN BEGIN ALLOC(TOKENSTRNG.PTR); K:=1;                .NUC    414    +3
415  044066              WCNT1:=WCNT+9 END ELSE WCNT1:=COLCNT-1;                        .NUC    415    -3
416  044073           FOR I:=WCNT TO WCNT1 DO BEGIN WORD[J]:=CARD[I]; J:=J+1 END;       .NUC    416    +3-3
417  044114           PACK(WORD,1,TOKENSTRNG.STRNG);                                    .NUC    417
418  044124           WHILE WCNT1<COLCNT-1 DO BEGIN WCNT:=WCNT+10;                      .NUC    418    +3
419  044131              TEMP1:=COLCNT-WCNT; IF TEMP1>10 THEN WCNT1:=WCNT+9 ELSE        .NUC    419
420  044147                 WCNT1:=COLCNT-1; J:=1;                                      .NUC    420
421  044142              FOR I:=WCNT TO WCNT1 DO BEGIN WORD[J]:=CARD[I];                .NUC    421    +4
422  044156                 J:=J+1 END;                                                 .NUC    422    -4
423  044163                 WHILE J≤10 DO BEGIN WORD[J]:=BLANK; J:=J+1 END;             .NUC    423    +4-4
424  044175              PACK(WORD,1,TOKENSTRNG.PTR↑[K]); K:=K+1;                       .NUC    424
425  044214              IF K>5 THEN BEGIN WRITE(Ξ ERRORΞ,EOL); WCNT1:=COLCNT-1 END;    .NUC    425    +4-4
426  044224              END;                                                           .NUC    426    -3
427  044235           TOKENSTRNG.NUM:=TEMP;                                             .NUC    427
428  044241           IF TOKENSTRNG.STRNG≠ASSERT THEN BEGIN IF (TOKENSTRNG.NUM<10)      .NUC    428    +3
429  044234              ∧ (RESERVED(TOKENSTRNG.STRNG)) THEN                            .NUC    429
430  044242                 TOKEN:=TOKENSTRNG.STRNG END ELSE BEGIN                      .NUC    430    -3+3
431  044245                 TOKEN:=ASSERTION;                                           .NUC    431
```

```
432  044247                WHILE (CARD[COLCNT]#SEMI) ^ (~EOF(INPUT)) DO          .NUC    432
433  044257                   GETCHAR; GETCHAR END                               .NUC    433   -3
434  044264             END                                                      .NUC    434   -2
435  044264          END;                                                        .NUC    435   -1
436  044301 PROCEDURE ERROR(NUM: INTEGER);                                       .NUC    436
437  044301   *PRINT ERROR MESSAGE NUMBERS AND MAKE NECESSARY RECOVERY*          .NUC    437
438  044301      CONST STARS='****';                                            .NUC    438
439  044301      BEGIN WRITE(BLANK,STARS:4,NUM:3,BLANK,STARS,EOL);              .NUC    439   +1
440  044320         IF NUM IN [1,4,6,15,22,23,27,28,29,31] THEN                 .NUC    440
441  044323         WHILE (TOKEN#SEMI1) ∨ (EOF(INPUT)) DO SCANNER;              .NUC    441
442  044332         ERRSET:=ERRSET ∨[NUM];                                      .NUC    442
443  044335      END;                                                           .NUC    443   -1
444  044341 PROCEDURE ERRPRINT;                                                  .NUC    444
445  044341   *PRINT ERROR MESSAGES CORRESPONDING TO ERRORS DETECTED*           .NUC    445
446  044341      CONST EXPECTED='EXPECTED'; UNDEF='UNDEFINED'; TY='TYPE';       .NUC    446
447  044341      VAR I: INTEGER;                                                .NUC    447
448  044341      BEGIN WRITE('1',EOL);                                          .NUC    448   +1
449  044350         FOR I:=1 TO 17 DO IF I IN ERRSET THEN                       .NUC    449
450  044356         CASE I OF                                                   .NUC    450   +C
451  044363            1: WRITE(ZER,I:7,COLON1:3,IDENTIFIER:11,EXPECTED,EOL);   .NUC    451
452  044400            2: WRITE(ZER,I:7,COLON1:3,SEMI:2,EXPECTED,EOL);          .NUC    452
453  044415            3: WRITE(ZER,I:7,COLON1:3,LEFTBRACK:2,EXPECTED,EOL);     .NUC    453
454  044432            4: WRITE(ZER,I:7,COLON1:3,NUMBER:7,EXPECTED,EOL);        .NUC    454
455  044447            5: WRITE(ZER,I:7,COLON1:3,RIGHTBRACK:2,EXPECTED,EOL);    .NUC    455
456  044464            6: WRITE(ZER,I:7,COLON1:3,'STATEMENT',EXPECTED,EOL);     .NUC    456
457  044501            7: WRITE(ZER,I:7,COLON1:3,RESWRDS[22]:6,EXPECTED,EOL);   .NUC    457
458  044516            8: WRITE(ZER,I:7,COLON1:3,'LABEL PREV',                  .NUC    458
459  044526               'IOUSLY DEF','INED',EOL);                             .NUC    459
460  044535            9: WRITE(ZER,I:7,COLON1:3,UNDEF,IDENTIFIER,EOL);         .NUC    460
461  044552           10: WRITE(ZER,I:7,COLON1:3,UNDEF,RESWRDS[1],EOL);         .NUC    461
462  044567           11: WRITE(ZER,I:7,COLON1:3,'PREVIOUSLY':11,'DEFINED':8,   .NUC    462
463  044601               IDENTIFIER,EOL);                                      .NUC    463
464  044606           12: WRITE(ZER,I:7,COLON1:3,RESWRDS[19],EXPECTED,EOL);     .NUC    464
465  044623           13: WRITE(ZER,I:7,COLON1:3,RESWRDS[10]:5,EXPECTED,EOL);   .NUC    465
466  044640           14: WRITE(ZER,I:7,COLON1:3,UNDEF,'LABEL',EOL);            .NUC    466
467  044655           15: WRITE(ZER,I:7,COLON1:3,RESWRDS[18]:3,EXPECTED,EOL);   .NUC    467
468  044672           16: WRITE(ZER,I:7,COLON1:3,TY:5,RESWRDS[2]:8,EXPECTED,EOL); .NUC  468
469  044711           17: WRITE(ZER,I:7,COLON1:3,TY:4,'S DO NOT M','ATCH',EOL); .NUC    469
470  044730         END;                                                        .NUC    470   -C
471  044754         FOR I:=18 TO 35 DO IF I IN ERRSET THEN                      .NUC    471
472  044762         CASE I OF                                                   .NUC    472   +C
473  044767           18: WRITE(ZER,I:7,COLON1:3,'UNACCEPTAB','LE RELATIO',     .NUC    473
474  045001               'N TYPE',EOL);                                        .NUC    474
475  045006           19: WRITE(ZER,I:7,COLON1:3,TY:5,RESWRDS[16]:8,EXPECTED,EOL); .NUC  475
476  045025           20: WRITE(ZER,I:7,COLON1:3,LEFTPAREN:2,EXPECTED,EOL);     .NUC    476
477  045042           21: WRITE(ZER,I:7,COLON1:3,RIGHTPAREN:5,EXPECTED,EOL);    .NUC    477
478  045057           22: WRITE(ZER,I:7,COLON1:3,COLEQ:3,EXPECTED,EOL);         .NUC    478
479  045074           23: WRITE(ZER,I:7,COLON1:3,RESWRDS[24]:3,EXPECTED,EOL);   .NUC    479
480  045111           24: WRITE(ZER,I:7,COLON1:3,RESWRDS[6]:6,EXPECTED,EOL);    .NUC    480
481  045126           25: WRITE(ZER,I:7,COLON1:3,RESWRDS[4],RESWRDS[1]:6,       .NUC    481
482  045140               EXPECTED,EOL);                                        .NUC    482
483  045145           26: WRITE(ZER,I:7,COLON1:3,RESWRDS[12]:3,'OR':3,          .NUC    483
484  045157               RESWRDS[7]:5,EXPECTED,EOL);                           .NUC    484
485  045166           27: WRITE(ZER,I:7,COLON1:3,RESWRDS[23]:5,EXPECTED,EOL);   .NUC    485
486  045203           28: WRITE(ZER,I:7,COLON1:3,RESWRDS[5]:3,EXPECTED,EOL);    .NUC    486
487  045220           29: WRITE(ZER,I:7,COLON1:3,'ERROR IN D','ECLARATION',     .NUC    487
488  045232               ' PART',EOL);                                         .NUC    488
489  045237           30: WRITE(ZER,I:7,COLON1:3,COLON1:2,EXPECTED,EOL);        .NUC    489
490  045254           31: WRITE(ZER,I:7,COLON1:3,'UNACCEPTAB','LE PRIMARY':11,  .NUC    490
491  045266               TY,EOL);                                              .NUC    491
492  045273           32: WRITE(ZER,I:7,COLON1:3,UNDEF,RESWRDS[19],EOL);        .NUC    492
493  045310           33: WRITE(ZER,I:7,COLON1:3,RESWRDS[7]:5,'OR':3,           .NUC    493
```

```
494  045322              RESWRDS[9]:5,EXPECTED,EOL)};                                  .NUC   494
495  045331           34: WRITE(ZER,I:7,COLON1:3,RESWRDS[1]:6,'TOO LARGE',EOL)};       .NUC   495
496  045346           35: WRITE(ZER,I:7,COLON1:3,RESWRDS[19],'TOO LONG',EOL)};         .NUC   496
497  045363         END                                                                .NUC   497   -C
498  045405      END};                                                                 .NUC   498   -1
499  045443 PROCEDURE SORT(VAR TABLE: CLSTK; INDEX: INTEGER);                         .NUC   499
500  045443   *SORT ROUTINE.  MAINLY TO SORT CASELABELSET*                            .NUC   500
501  045443      VAR I,TEMP: INTEGER; BOOL: BOOLEAN;                                  .NUC   501
502  045443      BEGIN REPEAT I:=1; BOOL:=TRUE;                                       .NUC   502   +1+R
503  045450            REPEAT IF TABLE[I,1]>TABLE[I+1,1] THEN                         .NUC   503   +R
504  045462              BEGIN TEMP:=TABLE[I,1]; TABLE[I,1]:=TABLE[I+1,1];            .NUC   504   +2
505  045501                    TABLE[I+1,1]:=TEMP; BOOL:=FALSE;                       .NUC   505
506  045506                    TEMP:=TABLE[I,2]; TABLE[I,2]:=TABLE[I+1,2];            .NUC   506
507  045525                    TABLE[I+1,2]:=TEMP; END;                               .NUC   507   -2
508  045532              I:=I+1                                                       .NUC   508
509  045533            UNTIL I>INDEX                                                  .NUC   509   -R
510  045535            UNTIL BOOL;                                                    .NUC   510   -R
511  045537      END};                                                                .NUC   511   -1
512  045541 PROCEDURE FIX(VAR TABLE: STACKC; IX: SCNT);                               .NUC   512
513  045541      *FIX REDUCED PROGRAM TREE FOR RETURN OR END OF CASE*                 .NUC   513
514  045541         VAR I,J: INTEGER; ALF1: ALFA;                                     .NUC   514
515  045541         BEGIN ALF1:=ALF1(PROCNAME.BOUND*POINT);                           .NUC   515   +1
516  045552            FOR I:=1 TO IX DO                                              .NUC   516
517  045556            BEGIN J:=TABLE[I];                                             .NUC   517   +2
518  045564            SENTREE[J].LSON*.FATHER:=ALF1 END                              .NUC   518   -2
519  045571         END;                                                              .NUC   519   -1
520  045575 PROCEDURE GETLENGTH(W:IDWORD;VAR P,I,J,K,L,M,N:INTEGER);                  .NUC   520
521  045575      VAR T: INTEGER;                                                      .NUC   521
522  045575         BEGIN P:=W.NUM; J:=0;K:=0;L:=0;M:=0;N:=0;                         .NUC   522   +1
523  045610            T:=P MOD 10; IF T=0 THEN BEGIN P:=(P DIV 10)-1; T:=10 END      .NUC   523   +2-2
524  045623            ELSE P:=P DIV 10;                                              .NUC   524
525  045627            CASE P OF                                                      .NUC   525   +C
526  045634               0: I:=T;                                                    .NUC   526
527  045636               1: BEGIN I:=10; J:=T END;                                   .NUC   527   +2-2
528  045642               2: BEGIN I:=10; J:=10; K:=T END;                            .NUC   528   +2-2
529  045647               3: BEGIN I:=10; J:=10; K:=10; L:=T END;                     .NUC   529   +2-2
530  045656               4: BEGIN I:=10; J:=10; K:=10; L:=10; M:=T END;              .NUC   530   +2-2
531  045666               5: BEGIN I:=10; J:=10; K:=10; L:=10; M:=10; N:=T END        .NUC   531   +2-2
532  045677            END                                                            .NUC   532   -C
533  045706         END;                                                              .NUC   533   -1
534  045710   *SENTENCE - SENTENCE4                                                   .NUC   534
535  045710      PRINT OUT SENTENCES OF REDUCED PROGRAM AND SET UP TREES              .NUC   535
536  045710      OF REDUCED PROGRAM WHERE APPROPRIATE*                                .NUC   536
537  045710 PROCEDURE SENTENCE(STATENUM:INTEGER; W:IDWORD);                           .NUC   537
538  045710      VAR P,I,J,K,L,M,N,A,B,C,D,E,F:INTEGER;                               .NUC   538
539  045710      BEGIN GETLENGTH(W,P,I,J,K,L,M,N);                                    .NUC   539   +1
540  045725            IF P=0 THEN ALLOC(P,PTR);                                      .NUC   540
541  045735         IF (SENFLAG#0) v (STATENUM IN [3,4,5,6,8,9,11]) THEN              .NUC   541
542  045743         CASE STATENUM OF                                                  .NUC   542   +C
543  045747 *SIMPLE*    1: WRITE(ZER,BLANK:5,                                        .NUC   543
544  045753                  'SIMPLE  :7,TOKENSTRNG.STRNG:I,TOKENSTRNG.PTR*[1]:J,    .NUC   544
545  045762                 TOKENSTRNG.PTR*[2]:K,TOKENSTRNG.PTR*[3]:L,               .NUC   545
546  045770                 TOKENSTRNG.PTR*[4]:M,TOKENSTRNG.PTR*[5]:N,RP,EOL);       .NUC   546
547  046003 *ARRAY*     2:  WRITE(ZER,BLANK:5,'ARRY:6,ARRAYNAME.STRNG:I,             .NUC   547
548  046013                 ARRAYNAME.PTR*[1]:J,'ARRAYNAME.PTR*[2]:K,                .NUC   548
549  046021                 'ARRAYNAME.PTR*[3]:L,'ARRAYNAME.PTR*[4]:M,               .NUC   549
550  046027                 'ARRAYNAME.PTR*[5]:N,' CHA,'ARRAYNAME.BOUND:8,RP,EOL);   .NUC   550
551  046044 *EXIT*      3:  BEGIN                                                    .NUC   551   +2
552  046044                 SCTR:=SCTR+1;                                            .NUC   552
553  046046                 SENTREE[SCTR].FATHER:=RESWRDS[10];                       .NUC   553
554  046052                 ALLOC(SENTREE[SCTR].LSON);                               .NUC   554
555  046064                 ADDR:=PROCNAME.BOUND*POINT;                              .NUC   555
```

```
556  046066                SENTREE[SCTR].LSON↑.FATHER:=ALFI(ADDR)$                                  .NUC    556
557  046077                SENTREE[SCTR].LSON↑.LSON:=NIL$                                           .NUC    557
558  046106                SENTREE[SCTR].RSON:=NIL$                                                 .NUC    558
559  046113                IF SENFLAG≠0 THEN BEGIN                                                  .NUC    559    +3
560  046115                WRITE(ZER,BLANK:5,EXITS:5,PROCNAME.STRNG:I,                              .NUC    560
561  046125                PROCNAME.PTR↑[1]:J,PROCNAME.PTR↑[2]:K,PROCNAME.PTR↑[3]:K,                .NUC    561
562  046136                PROCNAME.PTR↑[4]:M,PROCNAME.PTR↑[5]:N,CLN,POINT:5,RP,EOL)$               .NUC    562
563  046154                WRITE(BLANK:6,EXITPT:9,LP,PROCNAME.STRNG:I,                              .NUC    563
564  046164                PROCNAME.PTR↑[1]:J,PROCNAME.PTR↑[2]:K,PROCNAME.PTR↑[3]:L,                .NUC    564
565  046175                PROCNAME.PTR↑[4]:M,PROCNAME.PTR↑[5]:N,PREQ:2,POINT:5,EOL)$               .NUC    565
566  046211                END                                                                      .NUC    566    -3
567  046211                END$                                                                     .NUC    567    -2
568  046212 *RETURN↓   4$    BEGIN                                                                 .NUC    568    +2
569  046212                SCTR:=SCTR+1$                                                            .NUC    569
570  046214                SENTREE[SCTR].FATHER:=JUMPTO$                                            .NUC    570
571  046220                ALLOC(SENTREE[SCTR].LSON)$                                               .NUC    571
572  046232                SENTREE[SCTR].RSON:=NIL$                                                 .NUC    572
573  046237                SENTREE[SCTR].LSON↑.LSON:=NIL$                                           .NUC    573
574  046246                KR:=KR+1$                                                                .NUC    574
575  046250                KEEPRTN[KR]:=SCTR$                                                       .NUC    575
576  046255                IF SENFLAG≠0 THEN                                                        .NUC    576
577  046256                WRITE(ZER,BLANK:5,                                                       .NUC    577
578  046262                 JUMPTO  :7,PROCNAME.STRNG:I,PROCNAME.PTR↑[1]:J,                         .NUC    578
579  046271                PROCNAME.PTR↑[2]:K,PROCNAME.PTR↑[3]:L,PROCNAME.PTR↑[4]:M,                .NUC    579
580  046302                PROCNAME.PTR↑[5]:N,CLN,POINT:5,CMA,EXITPT:9,LP,                          .NUC    580
581  046317                PROCNAME.STRNG:I,PROCNAME.PTR↑[1]:J,PROCNAME.PTR↑[2]:K,                  .NUC    581
582  046327                PROCNAME.PTR↑[3]:L,PROCNAME.PTR↑[4]:M,PROCNAME.PTR↑[5]:N,                .NUC    582
583  046340                DMPR:2,EOL)$                                                             .NUC    583
584  046344                END$                                                                     .NUC    584    -2
585  046345 *NOP↓      5$    BEGIN                                                                 .NUC    585    +2
586  046345                SCTR:=SCTR+1$                                                            .NUC    586
587  046347                SENTREE[SCTR].FATHER:=JUMPTO$                                            .NUC    587
588  046353                ALLOC(SENTREE[SCTR].LSON)$                                               .NUC    588
589  046365                SENTREE[SCTR].LSON↑.FATHER:=ALFI(PROCNAME.BOUND↑POINT↑1)$                .NUC    589
590  046400                SENTREE[SCTR].LSON↑.LSON:=NIL$                                           .NUC    590
591  046407                SENTREE[SCTR].RSON:=NIL$                                                 .NUC    591
592  046414                IF SENFLAG≠0 THEN                                                        .NUC    592
593  046416                WRITE(ZER,BLANK:5, JUMPTO  :7,PROCNAME.STRNG:I,                          .NUC    593
594  046426                PROCNAME.PTR↑[1]:J,PROCNAME.PTR↑[2]:K,PROCNAME.PTR↑[3]:L,                .NUC    594
595  046437                PROCNAME.PTR↑[4]:M,PROCNAME.PTR↑[5]:N,CLN,POINT:5,CMA,                   .NUC    595
596  046453                POINT↑1:5,RP,EOL)$                                                       .NUC    596
597  046462                END$                                                                     .NUC    597    -2
598  046463 *HALT↓      6$    BEGIN                                                                 .NUC    598    +2
599  046463                SCTR:=SCTR+1$                                                            .NUC    599
600  046465                SENTREE[SCTR].FATHER:=RESWRDS[14]$                                       .NUC    600
601  046471                ALLOC(SENTREE[SCTR].LSON)$                                               .NUC    601
602  046503                ADDR:=PROCNAME.BOUND↑POINT$                                              .NUC    602
603  046505                SENTREE[SCTR].LSON↑.FATHER:=ALFI(ADDR)$                                  .NUC    603
604  046516                SENTREE[SCTR].LSON↑.LSON:=NIL$                                           .NUC    604
605  046525                SENTREE[SCTR].RSON:=NIL$                                                 .NUC    605
606  046532                IF SENFLAG≠0 THEN                                                        .NUC    606
607  046534                WRITE(ZER,                                                               .NUC    607
608  046536                BLANK:5,HALTS:5,PROCNAME.STRNG:I,PROCNAME.PTR↑[1]:J,                     .NUC    608
609  046547                PROCNAME.PTR↑[2]:K,PROCNAME.PTR↑[3]:L,PROCNAME.PTR↑[4]:M,                .NUC    609
610  046560                PROCNAME.PTR↑[5]:N,CLN,POINT:5,RP,EOL)$                                  .NUC    610
611  046573                END$                                                                     .NUC    611    -2
612  046574 *LABEL↓     7$ BEGIN A:=1$ B:=J$ C:=K$ D:=L$ E:=M$ F:=N$                                .NUC    612    +2
613  046605                GETLENGTH(IDNAME,P,I,J,K,L,M,N)$                                         .NUC    613
614  046616                IF P=0 THEN ALLOC(IDNAME.PTR)$                                           .NUC    614
615  046626                IF P≠0 THEN P:=2$ IF N≠0 THEN N:=2$ WRITE(ZER,BLANK:5,                   .NUC    615
616  046649                POINTLABEL,POINTLBL:7,LP,PROCNAME.STRNG:A,                               .NUC    616
617  046663                PROCNAME.PTR↑[1]:B,PROCNAME.PTR↑[2]:C,PROCNAME.PTR↑[3]:D,                .NUC    617
```

```
618  0+6661                PROCNAME.PTR+[4]:E,PROCNAME.PTR+[5]:F,CLN,IDNAME.STRNG:I,   .NUC    618
619  0+6673                IDNAME.PTR+[1]:J,IDNAME.PTR+[2]:K,IDNAME.PTR+[3]:L,        .NUC    619
620  0+6704                IDNAME.PTR+[4]:M,IDNAME.PTR+[5]:N,PREQ:2,POINT:5,EOL);     .NUC    620
621  0+6720                IF P=0 THEN RESET(IDNAME.PTR);                             .NUC    621
622  0+6724                END;                                                       .NUC    622   -2
623  0+6725 *ASSIGN+   8:  BEGIN                                                      .NUC    623   +2
624  0+6725                SCTR:=SCTR+1;                                              .NUC    624
625  0+6727                SENTREE(SCTR).FATHER:=ASSIGNS;                             .NUC    625
626  0+6733                ALLOC(SENTREE(SCTR).LSON);                                 .NUC    626
627  0+6745                ALLOC(SENTREE(SCTR).RSON);                                 .NUC    627
628  0+6756                SENTREE(SCTR).LSON+.FATHER:=LEFTSIDE.FATHER;               .NUC    628
629  0+6763                SENTREE(SCTR).LSON+.INX:=LEFTSIDE.INX;                     .NUC    629
630  0+6773                SENTREE(SCTR).RSON+:=EXPSTRNG(EX)+;                        .NUC    630
631  0+7006                SENTREE(SCTR).LSON+.LSON:=LEFTSIDE.LSON;                   .NUC    631
632  0+7016                SENTREE(SCTR).LSON+.RSON:=LEFTSIDE.RSON;                   .NUC    632
633  0+7025                IF SENFLAG#0 THEN                                          .NUC    633
634  0+7026                WRITE(ZER,BLANK:5,ASSIGNS:7,                               .NUC    634
635  0+7034                PROCNAME.STRNG:1,PROCNAME.PTR+[1]:J,PROCNAME.PTR+[2]:K,    .NUC    635
636  0+7044                PROCNAME.PTR+[3]:L,PROCNAME.PTR+[4]:M,PROCNAME.PTR+[5]:N,  .NUC    636
637  0+7055                CLN,POINT:5,CMA,LEFTSIDE.FATHER,CMA,                       .NUC    637
638  0+7067               EXPSTRNG(EX)+,FATHER,RP,EOL);                               .NUC    638
639  0+7101                END;                                                       .NUC    639   -2
640  0+7102 *WHILE+    9:    BEGIN                                                    .NUC    640   +2
641  0+7102                ADDR:=WHILEPOINT(WH);                                      .NUC    641
642  0+7107                SENTREE(ADDR).FATHER:=RESWRDS(15);                         .NUC    642
643  0+7114                ALLOC(SENTREE(ADDR).LSON);                                 .NUC    643
644  0+7126                ALLOC(SENTREE(ADDR).RSON);                                 .NUC    644
645  0+7137                SENTREE(ADDR).LSON+:=WHILEXPR(WH)+;                        .NUC    645
646  0+7153                SENTREE(ADDR).RSON+.FATHER:=ALFI(ADDR+PROCNAME.BOUND+1);   .NUC    646
647  0+7167                ALLOC(SENTREE(ADDR).RSON+.LSON);                           .NUC    647
648  0+7202                SENTREE(ADDR).RSON+.LSON+.FATHER:=                         .NUC    648
649  0+7207                   ALFI(POINT+PROCNAME.BOUND);                             .NUC    649
650  0+7215                SENTREE(ADDR).RSON+.RSON:=NIL;                             .NUC    650
651  0+7223                SENTREE(ADDR).RSON+.LSON+.LSON:=NIL;                       .NUC    651
652  0+7233                IF SENFLAG#0 THEN                                          .NUC    652
653  0+7235                WRITE(BLANK:6,IFS:3,PROCNAME.STRNG:1,PROCNAME.PTR+[1]:J,   .NUC    653
654  0+7246                PROCNAME.PTR+[2]:K,PROCNAME.PTR+[3]:L,PROCNAME.PTR+[4]:M,  .NUC    654
655  0+7257                PROCNAME.PTR+[5]:N,CLN,WHILEPOINT(WH]:3,CMA,               .NUC    655
656  0+7274                WHILEXPR(WH)+,FATHER,CMA,WHILEPOINT(WH]+1:3,CMA,           .NUC    656
657  0+7315                POINT:5,RP,EOL);                                           .NUC    657
658  0+7323                END;                                                       .NUC    658   -2
659  0+7324 *CASELBL+  10:  WRITE(ZER,                                                .NUC    659
660  0+7336                BLANK:5,POINTLABEL,POINTLBL]:7,LP,PROCNAME.STRNG:1,        .NUC    660
661  0+7350                PROCNAME.PTR+[1]:J,PROCNAME.PTR+[2]:K,PROCNAME.PTR+[3]:L,  .NUC    661
662  0+7361                PROCNAME.PTR+[4]:M,PROCNAME.PTR+[5]:N,CLN,                 .NUC    662
663  0+7361                CASEPOINT[CS]:3,CLN,NUMVAL:TOKENSTRNG.NUM,PREQ:2,          .NUC    663
664  0+7376                POINT:5,EOL);                                              .NUC    664
665  0+7403 *CASEJOIN+ 11:  BEGIN                                                     .NUC    665   +2
666  0+7403                SCTR:=SCTR+1;                                              .NUC    666
667  0+7405                SENTREE(SCTR).FATHER:=JUMPTO;                              .NUC    667
668  0+7411                ALLOC(SENTREE(SCTR).LSON);                                 .NUC    668
669  0+7423                SENTREE(SCTR).RSON:=NIL;                                   .NUC    669
670  0+7430                SENTREE(SCTR).LSON+.LSON:=NIL;                             .NUC    670
671  0+7437                CJ:=CJ+1;                                                  .NUC    671
672  0+7441                CASEJOIN(CS](CJ]:=SCTR;                                    .NUC    672
673  0+7453                IF SENFLAG#0 THEN BEGIN                                    .NUC    673   +3
674  0+7454                IF (N#0) ^ (N>3) THEN N:=3; WRITE(ZER,BLANK:5,             .NUC    674
675  0+7500                 JUMPTO  :7,PROCNAME.STRNG:1,PROCNAME.PTR+[1]:J,           .NUC    675
676  0+7515                PROCNAME.PTR+[2]:K,PROCNAME.PTR+[3]:L,PROCNAME.PTR+[4]:M,  .NUC    676
677  0+7506                PROCNAME.PTR+[5]:N,CLN,POINT:5,CMA,CASEJOIN1,CASEJOIN2:4,  .NUC    677
678  0+7523                PROCNAME.STRNG:1,PROCNAME.PTR+[1]:J,PROCNAME.PTR+[2]:K,    .NUC    678
679  0+7533                PROCNAME.PTR+[3]:L,PROCNAME.PTR+[4]:M,PROCNAME.PTR+[5]:N,  .NUC    679
```

```
                    CLN,CASEPOINT[CS]:3,DBPR:2,EOL)};                                  .NUC    680
                    END END};                                                          .NUC    681    -3-2
               END;                                                                    .NUC    682    -C
                  IF P=0 THEN RESET(W.PTR);                                            .NUC    683
               IF SENFLAG#0 THEN                                                       .NUC    684
               WRITE(BLANK,EOL);                                                       .NUC    685
          END;                                                                         .NUC    686    -1
PROCEDURE SENTENCE(STATENUM:INTEGER; W:IDWORD);                                        .NUC    687
          VAR P,I,J,K,L,M,N,A,B,C,D,E,F:INTEGER;                                       .NUC    688
          BEGIN GETLENGTH(W,P,I,J,K,L,M,N);                                            .NUC    689    +1
               IF P=0 THEN ALLOC(W.PTR);                                               .NUC    690
               SCTR:=SCTR+1;                                                           .NUC    691
               ALLOC(SENTREE[SCTR].LSON);                                              .NUC    692
               SENTREE[SCTR].LSON↑.LSON:=NIL;                                          .NUC    693
               SENTREE[SCTR].RSON:=NIL;                                                .NUC    694
          CASE STATENUM OF                                                             .NUC    695    +C
GOTO      28: BEGIN                                                                    .NUC    696    +2
                 SENTREE[SCTR].FATHER:=JUMPTO;                                         .NUC    697
                 ADDR:=TOKENSTRNG.BOUND;                                               .NUC    698
                 SENTREE[SCTR].LSON↑.FATHER:=ALFI(ADDR);                               .NUC    699
                 IF SENFLAG#0 THEN BEGIN                                               .NUC    700    +3
                 A:=1;                                                                 .NUC    701
                 B:=J; C:=K; D:=L; E:=M;                                               .NUC    702
                 GETLENGTH(TOKENSTRNG,P,I,J,K,L,M,N);                                  .NUC    703
                 IF P=0 THEN ALLOC(TOKENSTRNG.PTR);                                    .NUC    704
                 IF E#0 THEN E:=2;                                                     .NUC    705
                 WRITE(ZER,BLANK:5,JUMPTO:7,PROCNAME.STRNG:A,                          .NUC    706
                 PROCNAME.PTR↑[1]:B,PROCNAME.PTR↑[2]:C, CLN,POINT:5,                   .NUC    707
                 CMA,POINTLABEL,POINTLBL:7,LP,PROCNAME.STRNG:A,                        .NUC    708
                 PROCNAME.PTR↑[1]:B,PROCNAME.PTR↑[2]:C,CLN,                            .NUC    709
                 TOKENSTRNG.STRNG:I,TOKENSTRNG.PTR↑[1]:J,                              .NUC    710
                 TOKENSTRNG.PTR↑[2]:K,DBPR:2,EOL);                                     .NUC    711
                 IF P=0 THEN RESET(TOKENSTRNG.PTR);                                    .NUC    712
                 END                                                                   .NUC    713    -3
                 END;                                                                  .NUC    714    -2
ENTER     29: BEGIN                                                                    .NUC    715    +2
                 SENTREE[SCTR].FATHER:=RESWRDS[8];                                     .NUC    716
                 SENTREE[SCTR].LSON↑.FATHER:=TOKENSTRNG.STRNG;                         .NUC    717
                 IF SENFLAG#0 THEN BEGIN                                               .NUC    718    +3
                 A:=1; B:=J; C:=K; D:=L; E:=M; F:=N;                                   .NUC    719
                 GETLENGTH(TOKENSTRNG,P,I,J,K,L,M,N);                                  .NUC    720
                 IF P=0 THEN ALLOC(TOKENSTRNG.PTR);                                    .NUC    721
                 IF F#0 THEN F:=8; IF N#0 THEN N:=8;                                   .NUC    722
                 WRITE(ZER,BLANK:5,ENTERS:6,PROCNAME.STRNG:A,                          .NUC    723
                 PROCNAME.PTR↑[1]:B,PROCNAME.PTR↑[2]:C,PROCNAME.PTR↑[3]:D,             .NUC    724
                 PROCNAME.PTR↑[4]:E,PROCNAME.PTR↑[5]:F,CLN,POINT:5,CMA,                .NUC    725
                 TOKENSTRNG.STRNG:I,TOKENSTRNG.PTR↑[1]:J,                              .NUC    726
                 TOKENSTRNG.PTR↑[2]:K,TOKENSTRNG.PTR↑[3]:L,                            .NUC    727
                 TOKENSTRNG.PTR↑[4]:M,TOKENSTRNG.PTR↑[5]:N,RP,EOL);                    .NUC    728
                 IF P=0 THEN RESET(TOKENSTRNG.PTR);                                    .NUC    729
                 END                                                                   .NUC    730    -3
                 END;                                                                  .NUC    731    -2
I/O       30: BEGIN                                                                    .NUC    732    +2
                 SENTREE[SCTR].FATHER:=SAVE;                                           .NUC    733
                 SENTREE[SCTR].LSON↑.FATHER:=TOKENSTRNG.STRNG;                         .NUC    734
                 SENTREE[SCTR].LSON↑.INX:=ADDR;                                        .NUC    735
                 ADDR:=TOKENSTRNG.BOUND;                                               .NUC    736
                 ALLOC(SENTREE[SCTR].RSON);                                            .NUC    737
                 SENTREE[SCTR].RSON↑.FATHER:=ALFI(ADDR);                               .NUC    738
                 SENTREE[SCTR].RSON↑.LSON:=NIL;                                        .NUC    739
                 IF SENFLAG#0 THEN BEGIN                                               .NUC    740    +3
                 A:=1; B:=J; C:=K; D:=L; E:=M; F:=N;                                   .NUC    741
```

```
742  050302                GETLENGTH(TOKENSTRNG,P,I,J,K,L,M,N);                                  .NUC      742
743  050313                IF P=0 THEN ALLOC(TOKENSTRNG.PTR);                                    .NUC      743
744  050323                IF (F≠0) ^ (F>8) THEN F:=8; IF (N≠0) ^                                .NUC      744
745  050333                (N>8) THEN N:=8; WRITE(ZER,BLANK:5,SAVE:5,LP,                         .NUC      745
746  050347                PROCNAME.STRNG:A,PROCNAME.PTR↑[1]:B,PROCNAME.PTR↑[2]:C,               .NUC      746
747  050357                PROCNAME.PTR↑[3]:D,PROCNAME.PTR↑[4]:E,PROCNAME.PTR↑[5]:F,             .NUC      747
748  050370                CLN,POINT:5,CMA,TOKENSTRNG.STRNG:I,TOKENSTRNG.PTR↑[1]:J,              .NUC      748
749  050403                TOKENSTRNG.PTR↑[2]:K,TOKENSTRNG.PTR↑[3]:L,                            .NUC      749
750  050411                TOKENSTRNG.PTR↑[4]:M,TOKENSTRNG.PTR↑[5]:N,RP,EOL);                    .NUC      750
751  050423                IF P=0 THEN RESET(TOKENSTRNG.PTR);                                    .NUC      751
752  050427                END                                                                   .NUC      752   -3
753  050427                END;                                                                  .NUC      753   -2
754  050430 ^ELIM↓     31: BEGIN                                                                 .NUC      754   +2
755  050430                SENTREE[SCTR].FATHER:=JUMPTO;                                         .NUC      755
756  050435                ADDR:=PROCNAME.BOUND↓WHILEPOINT[WH];                                  .NUC      756
757  050443                SENTREE[SCTR].LSON↑.FATHER:=ALFI(ADDR);                               .NUC      757
758  050454                IF SENFLAG≠0 THEN                                                     .NUC      758
759  050455                WRITE(ZER,BLANK:5,JUMPTO:7,PROCNAME.STRNG:I,                          .NUC      759
760  050465                PROCNAME.PTR↑[1]:J,PROCNAME.PTR↑[2]:K,PROCNAME.PTR↑[3]:L,             .NUC      760
761  050476                PROCNAME.PTR↑[4]:M,PROCNAME.PTR↑[5]:N,CLN,POINT:5,CMA,                .NUC      761
762  050512                WHILEPOINT[WH]:3,RP,EOL);                                             .NUC      762
763  050524                END;                                                                  .NUC      763   -2
764  050525           END;                                                                       .NUC      764   -C
765  050531              IF P=0 THEN RESET(W.PTR);                                               .NUC      765
766  050535           IF SENFLAG≠0 THEN WRITE(EOL);                                              .NUC      766
767  050541        END;                                                                          .NUC      767   -1
768  050550 PROCEDURE SENTENCE2(STATENUM:INTEGER; W:IDWORD);                                     .NUC      768
769  050550        VAR P,I,J,K,L,M,N: INTEGER;                                                   .NUC      769
770  050550        BEGIN GETLENGTH(W,P,I,J,K,L,M,N);                                             .NUC      770   +1
771  050565              IF P=0 THEN ALLOC(W.PTR);                                               .NUC      771
772  050575              IF (SENFLAG≠0) ∨ (STATENUM IN [21,22,23,24]) THEN                       .NUC      772
773  050603           CASE STATENUM OF                                                           .NUC      773   +C
774  050607 ^IF↓          21: BEGIN                                                             .NUC      774   +2
775  050607                ADDR:=IFPOINT[IFP];                                                   .NUC      775
776  050614                SENTREE[ADDR].FATHER:=RESWRDS[15];                                    .NUC      776
777  050621                ALLOC(SENTREE[ADDR].LSON);                                            .NUC      777
778  050633                ALLOC(SENTREE[ADDR].RSON);                                            .NUC      778
779  050644                SENTREE[ADDR].LSON↑:=IFEXPR[IFE]↑;                                    .NUC      779
780  050660                SENTREE[ADDR].RSON↑.FATHER:=ALFI(PROCNAME.BOUND↓ADDR↓1);              .NUC      780
781  050674                ALLOC(SENTREE[ADDR].RSON↑.LSON);                                      .NUC      781
782  050707                SENTREE[ADDR].RSON↑.LSON↑.FATHER:=                                    .NUC      782
783  050714                   ALFI(PROCNAME.BOUND↓POINT);                                        .NUC      783
784  050722                SENTREE[ADDR].RSON↑.LSON↑.LSON:=NIL;                                  .NUC      784
785  050732                SENTREE[ADDR].RSON↑.RSON:=NIL;                                        .NUC      785
786  050740                IF SENFLAG≠0 THEN                                                     .NUC      786
787  050742                WRITE(ZER,BLANK:5,IFS:3,                                              .NUC      787
788  050750                PROCNAME.STRNG:I,PROCNAME.PTR↑[1]:J,PROCNAME.PTR↑[2]:K,               .NUC      788
789  050760                PROCNAME.PTR↑[3]:L,PROCNAME.PTR↑[4]:M,PROCNAME.PTR↑[5]:N,             .NUC      789
790  050771                CLN,IFPOINT[IFP]:3,CMA,IFEXPR[IFE]↑.FATHER,                           .NUC      790
791  051011                CMA,IFPOINT[IFP]↓1:3,                                                 .NUC      791
792  051022                CMA,POINT:5,RP,EOL);                                                  .NUC      792
793  051032                END;                                                                  .NUC      793   -2
794  051033 ^IFELSE↓       22: BEGIN                                                             .NUC      794   +2
795  051033                ADDR:=IFELSEPNT[IFEPT]-1;                                             .NUC      795
796  051040                SENTREE[ADDR].FATHER:=JUMPTO;                                         .NUC      796
797  051045                ALLOC(SENTREE[ADDR].LSON);                                            .NUC      797
798  051057                SENTREE[ADDR].LSON↑.FATHER:=ALFI(PROCNAME.BOUND↓POINT);               .NUC      798
799  051071                SENTREE[ADDR].LSON↑.LSON:=NIL;                                        .NUC      799
800  051100                SENTREE[ADDR].RSON:=NIL;                                              .NUC      800
801  051105                ADDR:=IFPOINT[IFP];                                                   .NUC      801
802  051112                SENTREE[ADDR].FATHER:=RESWRDS[15];                                    .NUC      802
803  051117                ALLOC(SENTREE[ADDR].LSON);                                            .NUC      803
```

```
804  051131                ALLOC(SENTREE[ADDR].RSON)!                                    .NUC    804
805  051142                SENTREE[ADDR].LSON↑:=IFEXPR[IFE]↑!                            .NUC    805
806  051156                SENTREE[ADDR].RSON↑.FATHER:=ALFI(ADDR+PROCNAME.BOUND+1)!      .NUC    806
807  051172                ALLOC(SENTREE[ADDR].RSON↑.LSON)!                              .NUC    807
808  051205                SENTREE[ADDR].RSON↑.LSON↑.FATHER:=                            .NUC    808
809  051212                   ALFI(PROCNAME.BOUND+IFELSEPNT[IFEPT])!                     .NUC    809
810  051223                SENTREE[ADDR].RSON↑.LSON↑.LSON:=NIL!                          .NUC    810
811  051233                SENTREE[ADDR].RSON↑.RSON:=NIL!                                .NUC    811
812  051241                IF SENFLAG≠0 THEN BEGIN                                       .NUC    812   +3
813  051243                WRITE(ZER,                                                    .NUC    813
814  051245                BLANK:5, JUMPTO  :7,PROCNAME.STRNG:1,PROCNAME.PTR↑[1]:J,      .NUC    814
815  051256                PROCNAME.PTR↑[2]:K,PROCNAME.PTR↑[3]:L,PROCNAME.PTR↑[4]:M,     .NUC    815
816  051267                PROCNAME.PTR↑[5]:N,IFELSEPNT[IFEPT]-1:3,CMA,POINT:5,RP,       .NUC    816
817  051307                EOL)! WRITE(BLANK:6,IFS:3,PROCNAME.STRNG:1,                   .NUC    817
818  051317                PROCNAME.PTR↑[1]:J,PROCNAME.PTR↑[2]:K,PROCNAME.PTR↑[3]:L,     .NUC    818
819  051330                PROCNAME.PTR↑[4]:M,PROCNAME.PTR↑[5]:N,CLN,IFPOINT[IFP]:3,     .NUC    819
820  051346                CMA,IFEXPR[IFE]↑.FATHER,CMA,IFPOINT[IFP]+1:3,CMA,             .NUC    820
821  051371                IFELSEPNT[IFEPT]:3,RP,EOL)!                                   .NUC    821
822  051403                END                                                           .NUC    822   -3
823  051403                END!                                                          .NUC    823   -2
824  051404 *CASESET*  23: BEGIN                                                         .NUC    824   +2
825  051404                   SORT(DEFCASELAB[CP],DC-1)!                                 .NUC    825
826  051415                FOR I:=1 TO DC-1 DO BEGIN                                     .NUC    826   +3
827  051422                SCTR:=SCTR+1!                                                 .NUC    827
828  051423                SENTREE[SCTR].FATHER:=ALFI(DEFCASELAB[CS][I,1])!              .NUC    828
829  051444                ALLOC(SENTREE[SCTR].LSON)!                                    .NUC    829
830  051456                SENTREE[SCTR].LSON↑.FATHER:=ALFI(DEFCASELAB[CS][I,2])!        .NUC    830
831  051477                SENTREE[SCTR].LSON↑.LSON:=NIL!                                .NUC    831
832  051506                SENTREE[SCTR].RSON:=NIL!                                      .NUC    832
833  051513                END!                                                          .NUC    833   -3
834  051516                IF SENFLAG≠0 THEN BEGIN                                       .NUC    834   +3
835  051520                WRITE(ZER,BLANK:5,≡CASELABELS≡,≡ET(≡:3,W.STRNG:1,             .NUC    835
836  051533                W.PTR↑[1]:J,W.PTR↑[2]:K,W.PTR↑[3]:L,                          .NUC    836
837  051547                W.PTR↑[4]:M,W.PTR↑[5]:N,CLN,CASEPOINT[CS]:3,PREQ:2,EOL)!      .NUC    837
838  051573                FOR I:=1 TO DC-1 DO                                           .NUC    838
839  051600                WRITE(BLANK,DEFCASELAB[CS][I,1],BLANK:5,                      .NUC    839
840  051617                DEFCASELAB[CS][I,2],EOL)!                                     .NUC    840
841  051637                END                                                           .NUC    841   -3
842  051637                END!                                                          .NUC    842   -2
843  051640 *ENDCASE*  24:  BEGIN                                                        .NUC    843   +2
844  051640                ADDR:=CASEPOINT[CS]!                                          .NUC    844
845  051645                SENTREE[ADDR].FATHER:=RESWRDS[3]!                             .NUC    845
846  051652                ALLOC(SENTREE[ADDR].LSON)!                                    .NUC    846
847  051664                ALLOC(SENTREE[ADDR].RSON)!                                    .NUC    847
848  051675                SENTREE[ADDR].LSON↑:=CASEXPR[CS]↑!                            .NUC    848
849  051711                SENTREE[ADDR].RSON↑.FATHER:=ALFI(PROCNAME.BOUND+POINT)!       .NUC    849
850  051724                ALLOC(SENTREE[ADDR].RSON↑.LSON)!                              .NUC    850
851  051737                SENTREE[ADDR].RSON↑.LSON↑.FATHER:=ALFI(PCNTR)!                .NUC    851
852  051751                SENTREE[ADDR].RSON↑.LSON↑.LSON:=NIL!                          .NUC    852
853  051761                SENTREE[ADDR].RSON↑.RSON:=NIL!                                .NUC    853
854  051767                IF SENFLAG≠0 THEN                                             .NUC    854
855  051771                WRITE(ZER,BLANK:5,CASES:5,                                    .NUC    855
856  051777                PROCNAME.STRNG:1,PROCNAME.PTR↑[1]:J,PROCNAME.PTR↑[2]:K,       .NUC    856
857  052007                PROCNAME.PTR↑[3]:L,PROCNAME.PTR↑[4]:M,PROCNAME.PTR↑[5]:N,     .NUC    857
858  052020                CLN,CASEPOINT[CS]:3,CMA,CASEXPR[CS]↑.FATHER,                  .NUC    858
859  052040                CMA,POINT:5,RP,EOL)!                                          .NUC    859
860  052050                END!                                                          .NUC    860   -2
861  052051 *CASELSE*  25:  WRITE(ZER,BLANK:5,CASEJOIN1,CASEJOIN2:4,                    .NUC    861
862  052061                PROCNAME.STRNG:1,PROCNAME.PTR↑[1]:J,PROCNAME.PTR↑[2]:K,       .NUC    862
863  052071                PROCNAME.PTR↑[3]:L,PROCNAME.PTR↑[4]:M,PROCNAME.PTR↑[5]:N,     .NUC    863
864  052102                CLN,CASEPOINT[CS]:3,PREQ:2,POINT:5,EOL)!                      .NUC    864
865  052121          END!                                                                .NUC    865   -C
```

```
866  052126          IF P=0 THEN RESET(W.PTR);                                .NUC    866
867  052132        IF SENFLAG#0 THEN                                          .NUC    867
868  052134        WRITE(BLANK,EOL);                                          .NUC    868
869  052140      END;                                                         .NUC    869    -1
870  052153 PROCEDURE UNRYADEXPR;                                            .NUC    870
871  052153   (*RECOGNIZE UNARY ADD EXPRESSION                               .NUC    871
872  052153      UNRYADSTNG: TREE REPRESENTATION OF UNARY ADD EXPRESSION*)   .NUC    872
873  052153      BEGIN IF (TOKEN=PLUS) ∨ (TOKEN=MINUS) THEN BEGIN            .NUC    873    +1+2
874  052165        UN:=UN+1; ALLOC(UNRYADSTNG[UN]);UNRYADSTNG[UN]↑.LSON:=NIL; .NUC    874
875  052206        UNRYADSTNG[UN]↑.FATHER:=TOKEN; SCANNER; PRIMARY;           .NUC    875
876  052215        IF PFL>0 THEN BEGIN UFL:=1; PFL:=0 END ELSE BEGIN          .NUC    876    +3-3+3
877  052221        IF PRMRYTYPE[PM]=16 THEN BEGIN UNRYADTYPE[UN]:=16;         .NUC    877    +4
878  052233          ALLOC(UNRYADSTNG[UN]↑.LSON);                             .NUC    878
879  052246          UNRYADSTNG[UN]↑.LSON↑:=PRMRYSTRNG[PM]↑;                  .NUC    879
880  052262          UNRYADSTNG[UN]↑.RSON:=NIL;                               .NUC    880
881  052271          END ELSE ERROR(19); PM:=PM-1 END END                     .NUC    881    -4-3-2
882  052275          ELSE BEGIN PRIMARY;                                      .NUC    882    +2
883  052277          IF PFL>0 THEN BEGIN UFL:=1; PFL:=0 END ELSE BEGIN        .NUC    883    +3-3+3
884  052303          PUSH(UNRYADTYPE,PRMRYTYPE[PM],UN);                       .NUC    884
885  052313          UNRYADSTNG[UN]:=PRMRYSTRNG[PM]; PM:=PM-1 END END         .NUC    885    -3-2
886  052325      END;                                                         .NUC    886    -1
887  052327 PROCEDURE MULTEXPR;                                              .NUC    887
888  052327   (*RECOGNIZE MULTIPLY EXPRESSION                                .NUC    888
889  052327      MULSTRNG: TREE REPRESENTATION OF MULTIPLY EXPRESSION*)       .NUC    889
890  052327      BEGIN UNRYADEXPR; IF UFL>0 THEN BEGIN MFL:=1; UFL:=0 END ELSE .NUC    890    +1+2-2
891  052337      BEGIN PUSH(MULTYPE,UNRYADTYPE[UN],MUL);                      .NUC    891    +2
892  052347        MULSTRNG[MUL]:=UNRYADSTNG[UN]; UN:=UN-1;                   .NUC    892
893  052361      1: IF ((TOKEN=TIMES) ∨ (TOKEN=SLASH) ∨ (TOKEN=EXPO)) ∧       .NUC    893
894  052373         (MULTYPE[MUL]=16) THEN BEGIN                            .NUC    894    +3
895  052402           SAVEPTR↑.LSON:=MULSTRNG[MUL]↑.LSON;                    .NUC    895
896  052413           ALLOC(MULSTRNG[MUL]↑.LSON);                            .NUC    896
897  052426           MULSTRNG[MUL]↑.LSON↑:=MULSTRNG[MUL]↑;                  .NUC    897
898  052442           MULSTRNG[MUL]↑.LSON↑.LSON:=SAVEPTR↑.LSON;              .NUC    898
899  052454           MULSTRNG[MUL]↑.FATHER:=TOKEN;                          .NUC    899
900  052461           SCANNER; UNRYADEXPR; IF UFL>0 THEN BEGIN MFL:=1; UFL:=0 .NUC    900    +4
901  052466           END ELSE BEGIN                                         .NUC    901    -4+4
902  052467           IF UNRYADTYPE[UN]=16 THEN BEGIN                        .NUC    902    +5
903  052475           ALLOC(MULSTRNG[MUL]↑.RSON);                            .NUC    903
904  052507           MULSTRNG[MUL]↑.RSON↑:=UNRYADSTNG[UN]↑;                 .NUC    904
905  052524           END ELSE ERROR(19);                                    .NUC    905    -5
906  052527           UN:=UN-1; GOTO 1;                                      .NUC    906
907  052531         END END END                                              .NUC    907    -4-3-2
908  052531      END;                                                         .NUC    908    -1
909  052533 PROCEDURE BNRYADEXPR;                                            .NUC    909
910  052533   (*RECOGNIZE BINARY ADD EXPRESSION                              .NUC    910
911  052533      BNRYADSTNG: TREE REPRESENTATION OF BINARY ADD STRING*)       .NUC    911
912  052533      BEGIN MULTEXPR; IF MFL>0 THEN BEGIN BFL:=1; MFL:=0 END ELSE  .NUC    912    +1+2-2
913  052543        BEGIN PUSH(BNRYADTYPE,MULTYPE[MUL],BN);                    .NUC    913    +2
914  052553        BNRYADSTNG[BN]:=MULSTRNG[MUL]; MUL:=MUL-1;                 .NUC    914
915  052565      1: IF ((TOKEN=PLUS) ∨ (TOKEN=MINUS)) ∧ (BNRYADTYPE[BN]=16)   .NUC    915
916  052601         THEN BEGIN                                              .NUC    916    +3
917  052603              SAVEPTR↑.LSON:=BNRYADSTNG[BN]↑.LSON;                .NUC    917
918  052614              ALLOC(BNRYADSTNG[BN]↑.LSON);                        .NUC    918
919  052627              BNRYADSTNG[BN]↑.LSON↑:=BNRYADSTNG[BN]↑;             .NUC    919
920  052643              BNRYADSTNG[BN]↑.LSON↑.LSON:=SAVEPTR↑.LSON;          .NUC    920
921  052655           BNRYADSTNG[BN]↑.FATHER:=TOKEN;                         .NUC    921
922  052662        SCANNER; MULTEXPR; IF MFL>0 THEN BEGIN BFL:=1; MFL:=0 END .NUC    922    +4-4
923  052670           ELSE BEGIN IF MULTYPE[MUL]=16 THEN BEGIN               .NUC    923    +4+5
924  052676              ALLOC(BNRYADSTNG[BN]↑.RSON);                        .NUC    924
925  052710              BNRYADSTNG[BN]↑.RSON↑:=MULSTRNG[MUL]↑;              .NUC    925
926  052725           END ELSE ERROR(19);                                    .NUC    926    -5
927  052730           MUL:=MUL-1; GOTO 1                                     .NUC    927
```

```
928  052732          END END END                                          .NUC    928   -4-3-2
929  052732       END;                                                    .NUC    929   -1
930  052734  PROCEDURE RELEXPR;                                          .NUC    930
931  052734    #RECOGNIZE RELATION EXPRESSION                            .NUC    931
932  052734       RELSTRNG: TREE REPRESENTATION OF RELATION EXPRESSION+   .NUC    932
933  052734       VAR TMPALF: ALFA;                                       .NUC    933
934  052734       BEGIN BNRYADEXPR; IF BFL>0 THEN BEGIN RFL:=1; BFL:=0 END ELSE   .NUC    934   +1+2-2
935  052744          BEGIN PUSH(RELTYPE,BNRYADTYPE[BN],REL);              .NUC    935   +2
936  052754          RELSTRNG[REL]:=BNRYADSTNG[BN]; BN:=BN-1; IF (TOKEN=LT) v   .NUC    936
937  052771          (TOKEN=LE) v (TOKEN=GT) v (TOKEN=GE) v (TOKEN=EQ) v (TOKEN=NE)   .NUC    937
938  053007          THEN BEGIN                                           .NUC    938   +3
939  053011             SAVEPTR+.LSON:=RELSTRNG[REL]+.LSON;               .NUC    939
940  053022             ALLOC(RELSTRNG[REL]+.LSON);                       .NUC    940
941  053035             RELSTRNG[REL]+.LSON+:=RELSTRNG[REL]+;             .NUC    941
942  053051             RELSTRNG[REL]+.LSON+.LSON:=SAVEPTR+.LSON;         .NUC    942
943  053063             RELSTRNG[REL]+.FATHER:=TOKEN;                     .NUC    943
944  053070             SCANNER; BNRYADEXPR; IF BFL>0 THEN BEGIN RFL:=1; BFL:=0 END   .NUC    944   +4-4
945  053076             ELSE BEGIN                                        .NUC    945   +4
946  053076             ALLOC(RELSTRNG[REL]+.RSON);                       .NUC    946
947  053110             IF (BNRYADTYPE[BN]=RELTYPE[REL]) ^ (RELTYPE[REL]#16) THEN   .NUC    947
948  053131             BEGIN                                             .NUC    948   +5
949  053131             SAVEPTR+.LSON:=RELSTRNG[REL]+.LSON+.LSON;         .NUC    949
950  053143             ALLOC(RELSTRNG[REL]+.LSON+.LSON);                 .NUC    950
951  053157             RELSTRNG[REL]+.LSON+.LSON+:=RELSTRNG[REL]+.LSON+; .NUC    951
952  053175             RELSTRNG[REL]+.LSON+.LSON+.LSON:=SAVEPTR+.LSON;   .NUC    952
953  053210             RELSTRNG[REL]+.LSON+.RSON:=NIL;                   .NUC    953
954  053217             CASE RELTYPE[REL] OF                              .NUC    954   +C
955  053227                2: TMPALF:=ΞINTOFBOOLΞ;                        .NUC    955
956  053231                4: TMPALF:=ΞINTOFCHARΞ;                        .NUC    956
957  053233             END;                                              .NUC    957   -C
958  053236             RELSTRNG[REL]+.LSON+.FATHER:=TMPALF;              .NUC    958
959  053244             RELSTRNG[REL]+.RSON+.FATHER:=TMPALF;              .NUC    959
960  053252             ALLOC(RELSTRNG[REL]+.RSON+.LSON);                 .NUC    960
961  053266             RELSTRNG[REL]+.RSON+.LSON+:=BNRYADSTNG[BN]+;      .NUC    961
962  053303             RELSTRNG[REL]+.RSON+.RSON:=NIL;                   .NUC    962
963  053313             END;                                              .NUC    963   -5
964  053313             IF BNRYADTYPE[BN]#RELTYPE[REL] THEN ERROR(17);    .NUC    964
965  053326             IF RELTYPE[REL]=16 THEN                           .NUC    965
966  053334                RELSTRNG[REL]+.RSON+:=BNRYADSTNG[BN]+;         .NUC    966
967  053350             RELTYPE[REL]:=2; BN:=BN-1;                        .NUC    967
968  053356          END END END                                          .NUC    968   -4-3-2
969  053356       END;                                                    .NUC    969   -1
970  053362  PROCEDURE NOTEXPR;                                          .NUC    970
971  053362    #RECOGNIZE NOT EXPRESSION                                 .NUC    971
972  053362       NOTSTRNG: TREE REPRESENTATION OF NOT EXPRESSION+        .NUC    972
973  053362       BEGIN IF TOKEN#NOT THEN BEGIN RELEXPR;                  .NUC    973   +1+2
974  053370          IF RFL>0 THEN BEGIN NFL:=1; RFL:=0 END ELSE BEGIN    .NUC    974   +3-3+3
975  053374          PUSH(NOTTYPE,RELTYPE[REL],NT);                       .NUC    975
976  053404          NOTSTRNG[NT]:=RELSTRNG[REL];                         .NUC    976
977  053415          REL:=REL-1 END END ELSE BEGIN SCANNER; RELEXPR;      .NUC    977   -3-2+2
978  053421          IF RFL>0 THEN BEGIN NFL:=1; RFL:=0 END ELSE BEGIN    .NUC    978   +3-3+3
979  053425          IF RELTYPE[REL]=2 THEN BEGIN                         .NUC    979   +4
980  053433          PUSH(NOTTYPE,TWO,NT);                                .NUC    980
981  053437             NOTSTRNG[NT]+.FATHER:=NOT;                        .NUC    981
982  053444             ALLOC(NOTSTRNG[NT]+.LSON);                        .NUC    982
983  053457             NOTSTRNG[NT]+.LSON+:=RELSTRNG[REL]+;              .NUC    983
984  053473             NOTSTRNG[NT]+.RSON:=NIL;                          .NUC    984
985  053502          END ELSE ERROR(16);                                  .NUC    985   -4
986  053524          REL:=REL-1 END END                                   .NUC    986   -3-2
987  053536       END;                                                    .NUC    987   -1
988  053540  PROCEDURE ANDEXPR;                                          .NUC    988
989  053540    #RECOGNIZE AND EXPRESSION                                 .NUC    989
```

```
990  053510      ANDSTRNG: TREE REPRESENTATION OF AND EXPRESSION*
991  053510      BEGIN NOTEXPR; IF NFL>0 THEN BEGIN AFL:=1; NFL:=0 END ELSE
992  053520         BEGIN PUSH(ANDTYPE,NOTTYPE[NT],AN);
993  053530         ANDSTRNG[AN]:=NOTSTRNG[NT]; NT:=NT-1;
994  053542      1: IF (TOKEN=AND) ^ (ANDTYPE[AN]=2) THEN BEGIN SCANNER;
995  053555         NOTEXPR; IF NFL>0 THEN BEGIN AFL:=1; NFL:=0 END ELSE
996  053562         BEGIN IF NOTTYPE[NT]=2 THEN BEGIN
997  053570            SAVEPTR*.LSON:=ANDSTRNG[AN]*.LSON;
998  053600            ALLOC(ANDSTRNG[AN]*.LSON);
999  053613            ANDSTRNG[AN]*.LSON*:=ANDSTRNG[AN]*;
1000 053627            ANDSTRNG[AN]*.LSON*.LSON:=SAVEPTR*.LSON;
1001 053641            ALLOC(ANDSTRNG[AN]*.RSON);
1002 053653            ANDSTRNG[AN]*.RSON*:=NOTSTRNG[NT]*;
1003 053670            ANDSTRNG[AN]*.FATHER:=AND;
1004 053675         END ELSE ERROR(16);
1005 053700         NT:=NT-1; GOTO 1 END
1006 053702         END END
1007 053702      END;
1008 053704 PROCEDURE EXPRESSION;
1009 053704   *RECOGNIZE EXPRESSION
1010 053704      EXPSTRNG: TREE REPRESENTATION OF EXPRESSION*
1011 053704      BEGIN ANDEXPR; IF AFL>0 THEN BEGIN EFL:=1; AFL:=0 END ELSE
1012 053714         BEGIN PUSH(EXPTYPE,ANDTYPE[AN],EX);
1013 053724         EXPSTRNG[EX]:=ANDSTRNG[AN]; AN:=AN-1;
1014 053736      1: IF (TOKEN=OR) ^ (EXPTYPE[EX]=2) THEN BEGIN SCANNER;
1015 053751         ANDEXPR; IF AFL>0 THEN BEGIN EFL:=1; AFL:=0 END ELSE
1016 053756         BEGIN IF ANDTYPE[AN]=2 THEN BEGIN
1017 053764            SAVEPTR*.LSON:=EXPSTRNG[EX]*.LSON;
1018 053774            ALLOC(EXPSTRNG[EX]*.LSON);
1019 054007            EXPSTRNG[EX]*.LSON*:=EXPSTRNG[EX]*;
1020 054023            EXPSTRNG[EX]*.LSON*.LSON:=SAVEPTR*.LSON;
1021 054035            ALLOC(EXPSTRNG[EX]*.RSON);
1022 054047            EXPSTRNG[EX]*.RSON*:=ANDSTRNG[AN]*;
1023 054064            EXPSTRNG[EX]*.FATHER:=OR;
1024 054071         END ELSE ERROR(16);
1025 054074         AN:=AN-1; GOTO 1;
1026 054076         END END END
1027 054076      END;
1028 054100 PROCEDURE PRIMARY;
1029 054100   *RECOGNIZE PRIMARY
1030 054100      PRMRYSTRNG: TREE REPRESENTATION OF PRIMARY*
1031 054100      VAR PFLAG,PFLAG1: INTEGER;
1032 054100          I: INTEGER;
1033 054100      BEGIN PFLAG:=1;
1034 054104         ALLOC(PRMRYSTRNG[PM+1]);
1035 054114         IF TNUM IN [31,11,25,32,30,41,16,4,2] THEN
1036 054116         CASE TNUM OF
1037 054123 *NUMBER*    31: BEGIN PFLAG:=0; PUSH(PRMRYTYPE,16,PM);
1038 054130                PRMRYSTRNG[PM]*.FATHER:=ALFI(NUMVAL);
1039 054142                PRMRYSTRNG[PM]*.INX:=ADDR;
1040 054151                PRMRYSTRNG[PM]*.TY:=2;
1041 054160                PRMRYSTRNG[PM]*.LSON:=NIL;
1042 054167                SCANNER END;
1043 054171 *T/F*       11,25: BEGIN PFLAG:=0; PUSH(PRMRYTYPE,TWO,PM);
1044 054177                PRMRYSTRNG[PM]*.FATHER:=TOKENSTRNG.STRNG;
1045 054204                PRMRYSTRNG[PM]*.INX:=2;
1046 054213                PRMRYSTRNG[PM]*.TY:=3;
1047 054222                PRMRYSTRNG[PM]*.LSON:=NIL;
1048 054231                SCANNER END;
1049 054233 *CHAR*      32: BEGIN PFLAG:=0; PUSH(PRMRYTYPE,FOUR,PM);
1050 054241                PRMRYSTRNG[PM]*.FATHER:=TOKENSTRNG.STRNG;
1051 054246                PRMRYSTRNG[PM]*.INX:=ADDR;
```

```
.NUC  990
.NUC  991   *1*2-2
.NUC  992   *2
.NUC  993
.NUC  994   *3
.NUC  995   *4-4
.NUC  996   *4*5
.NUC  997
.NUC  998
.NUC  999
.NUC 1000
.NUC 1001
.NUC 1002
.NUC 1003
.NUC 1004   -5
.NUC 1005   -4
.NUC 1006   -3-2
.NUC 1007   -1
.NUC 1008
.NUC 1009
.NUC 1010
.NUC 1011   *1*2-2
.NUC 1012   *2
.NUC 1013
.NUC 1014   *3
.NUC 1015   *4-4
.NUC 1016   *4*5
.NUC 1017
.NUC 1018
.NUC 1019
.NUC 1020
.NUC 1021
.NUC 1022
.NUC 1023
.NUC 1024   -5
.NUC 1025
.NUC 1026   -4-3-2
.NUC 1027   -1
.NUC 1028
.NUC 1029
.NUC 1030
.NUC 1031
.NUC 1032
.NUC 1033   *1
.NUC 1034
.NUC 1035
.NUC 1036   *C
.NUC 1037   *2
.NUC 1038
.NUC 1039
.NUC 1040
.NUC 1041
.NUC 1042   -2
.NUC 1043   *2
.NUC 1044
.NUC 1045
.NUC 1046
.NUC 1047
.NUC 1048   -2
.NUC 1049   *2
.NUC 1050
.NUC 1051
```

```
1052  054256                PRMRYSTRNG[PM]↑.TY:=0;                                  .NUC  1052
1053  054265                PRMRYSTRNG[PM]↑.LSON:=NIL;                              .NUC  1053
1054  054274                SCANNER END;                                            .NUC  1054    -2
1055  054276 ↑IDENT↓     30: BEGIN PFLAG:=0;                                        .NUC  1055    +2
1056  054277                IF MEMBER(TOKENSTRNG.SYM,S) THEN                        .NUC  1056
1057  054304                   PFLAG:=TOKENSTRNG.TYPEFNCTN                          .NUC  1057
1058  054304                   ELSE BEGIN ERROR(9); PFLAG:=3;                       .NUC  1058    +3
1059  054312                   TOKENSTRNG.BOUND:=-1 END;                            .NUC  1059    -3
1060  054316                   PUSH(PRMRYTYPE,PFLAG,PM);                            .NUC  1060
1061  054323                   PRMRYSTRNG[PM]↑.LSON:=NIL;                           .NUC  1061
1062  054332                PRMRYSTRNG[PM]↑.INX:=ADDR;                              .NUC  1062
1063  054342                PRMRYSTRNG[PM]↑.TY:=1;                                  .NUC  1063
1064  054351                   PRMRYSTRNG[PM]↑.FATHER:=TOKENSTRNG.STRNG; SCANNER;   .NUC  1064
1065  054357                   IF TOKENSTRNG.BOUND≥0 THEN                           .NUC  1065
1066  054361                   BEGIN IF TOKEN=LEFTBRACK THEN SCANNER                .NUC  1066    +3
1067  054363                   ELSE ERROR(3);                                       .NUC  1067
1068  054367                   ALLOC(PRMRYSTRNG[PM]↑.RSON);                         .NUC  1068
1069  054401                   I:=TOKENSTRNG.BOUND;                                 .NUC  1069
1070  054403                   PRMRYSTRNG[PM]↑.RSON↑.FATHER:=ALFI(I);               .NUC  1070
1071  054416                   PRMRYSTRNG[PM]↑.RSON↑.TY:=2;                         .NUC  1071
1072  054426                   EXPRESSION; IF EFL>0 THEN BEGIN PFL:=1; EFL:=0 END   .NUC  1072    +4-4
1073  054433                   ELSE BEGIN IF EXPTYPE[EX]=16 THEN BEGIN              .NUC  1073    +4+5
1074  054441                   ALLOC(PRMRYSTRNG[PM]↑.LSON);                         .NUC  1074
1075  054454                   PRMRYSTRNG[PM]↑.LSON↑:=EXPSTRNG[EX]↑;                .NUC  1075
1076  054470                   PRMRYSTRNG[PM]↑.RSON↑.LSON:=NIL;                     .NUC  1076
1077  054500                   END ELSE ERROR(19);                                  .NUC  1077    -5
1078  054503                   EX:=EX-1; IF TOKEN=RIGHTBRACK THEN SCANNER ELSE      .NUC  1078
1079  054511                   ERROR(5); END END END;                               .NUC  1079    -4-3-2
1080  054514 ↑Ξ(Ξ↓       41: BEGIN PFLAG:=0; SCANNER;                             .NUC  1080    +2
1081  054516                RESET(PRMRYSTRNG[PM+1]); EXPRESSION;                    .NUC  1081
1082  054523                IF EFL>0 THEN BEGIN PFL:=1; EFL:=0 END ELSE BEGIN       .NUC  1082    +3-3+3
1083  054527                PUSH(PRMRYTYPE,EXPTYPE[EX],PM);                         .NUC  1083
1084  054537              PRMRYSTRNG[PM]:=EXPSTRNG[EX];                             .NUC  1084
1085  054550                EX:=EX-1; IF TOKEN=RIGHTPAREN THEN SCANNER ELSE         .NUC  1085
1086  054555                ERROR(21); END END;                                     .NUC  1086    -3-2
1087  054560 ↑INTEGER↓   16: BEGIN PFLAG:=0; PUSH(PRMRYTYPE,16,PM);                .NUC  1087    +2
1088  054565                SCANNER; IF TOKEN=LEFTPAREN THEN SCANNER                .NUC  1088
1089  054570                ELSE ERROR(20);                                         .NUC  1089
1090  054574                   EXPRESSION; IF EFL>0 THEN BEGIN PFL:=1; EFL:=0 END   .NUC  1090    +3-3
1091  054601                   ELSE BEGIN                                           .NUC  1091    +3
1092  054601                   CASE EXPTYPE[EX] OF                                  .NUC  1092    +C
1093  054611                      16: PRMRYSTRNG[PM]:=EXPSTRNG[EX];                 .NUC  1093
1094  054622                      2: BEGIN PRMRYSTRNG[PM]↑.FATHER:=ΞINTOFBOOLΞ;     .NUC  1094    +4
1095  054627                         ALLOC(PRMRYSTRNG[PM]↑.LSON);                   .NUC  1095
1096  054642                         PRMRYSTRNG[PM]↑.LSON↑:=EXPSTRNG[EX]↑;          .NUC  1096
1097  054656                         END;                                           .NUC  1097    -4
1098  054657                      4: BEGIN PRMRYSTRNG[PM]↑.FATHER:=ΞINTOFCHARΞ;     .NUC  1098    +4
1099  054664                         ALLOC(PRMRYSTRNG[PM]↑.LSON);                   .NUC  1099
1100  054677                         PRMRYSTRNG[PM]↑.LSON↑:=EXPSTRNG[EX]↑;          .NUC  1100
1101  054713                         END;                                           .NUC  1101    -4
1102  054714                   END;                                                 .NUC  1102    -C
1103  054733                   EX:=EX-1; PRMRYSTRNG[PM]↑.RSON:=NIL;                 .NUC  1103
1104  054743                   IF TOKEN=RIGHTPAREN THEN SCANNER ELSE                .NUC  1104
1105  054747                ERROR(21); END END;                                     .NUC  1105    -3-2
1106  054752 ↑CHAR↓      4: BEGIN PFLAG:=0; PUSH(PRMRYTYPE,FOUR,PM);              .NUC  1106    +2
1107  054760                SCANNER; IF TOKEN=LEFTPAREN THEN SCANNER                .NUC  1107
1108  054763                ELSE ERROR(20);                                         .NUC  1108
1109  054767                   EXPRESSION; IF EFL>0 THEN BEGIN PFL:=1; EFL:=0 END   .NUC  1109    +3-3
1110  054774                   ELSE BEGIN                                           .NUC  1110    +3
1111  054774                   CASE EXPTYPE[EX] OF                                  .NUC  1111    +C
1112  055004                   16: BEGIN PRMRYSTRNG[PM]↑.FATHER:=ΞCHAROFINTΞ;       .NUC  1112    +4
1113  055011                      ALLOC(PRMRYSTRNG[PM]↑.LSON);                      .NUC  1113
```

```
1114  055024                    PRMRYSTRNG[PM]↑.LSON↑:=EXPSTRNG[EX]↑;                .NUC    1114
1115  055040                       END;                                              .NUC    1115   -4
1116  055041                    2: BEGIN PRMRYSTRNG[PM]↑.FATHER:=≡CHAROFBOOL≡;       .NUC    1116   +4
1117  055056                       ALLOC(PRMRYSTRNG[PM]↑.LSON);                      .NUC    1117
1118  055061                       PRMRYSTRNG[PM]↑.LSON↑:=EXPSTRNG[EX]↑;             .NUC    1118
1119  055075                          END;                                           .NUC    1119   -4
1120  055076                    4: PRMRYSTRNG[PM]:=EXPSTRNG[EX];                     .NUC    1120
1121  055107                    END;                                                 .NUC    1121   -C
1122  055126                    EX:=EX-1; PRMRYSTRNG[PM]↑.RSON:=NIL;                 .NUC    1122
1123  055136                    IF TOKEN=RIGHTPAREN THEN SCANNER ELSE                .NUC    1123
1124  055142                 ERROR(21); END END;                                     .NUC    1124   -3-2
1125  055145 ↑BOOLEAN↓  2: BEGIN PFLAG:=0; PUSH(PRMRYTYPE,TWO,PM);                  .NUC    1125   +2
1126  055153                 SCANNER; IF TOKEN=LEFTPAREN THEN SCANNER             .NUC    1126
1127  055156                 ELSE ERROR(20);                                      .NUC    1127
1128  055162                    EXPRESSION; IF EFL>0 THEN BEGIN PFL:=1; EFL:=0 END .NUC    1128   +3-3
1129  055167                    ELSE BEGIN                                        .NUC    1129   +3
1130  055167                    CASE EXPTYPE[EX] OF                               .NUC    1130   +C
1131  055177                    16: BEGIN PRMRYSTRNG[PM]↑.FATHER:=≡BOOLOFINT≡;    .NUC    1131   +4
1132  055204                       ALLOC(PRMRYSTRNG[PM]↑.LSON);                   .NUC    1132
1133  055217                       PRMRYSTRNG[PM]↑.LSON↑:=EXPSTRNG[EX]↑;          .NUC    1133
1134  055233                          END;                                        .NUC    1134   -4
1135  055234                    2: PRMRYSTRNG[PM]:=EXPSTRNG[EX];                  .NUC    1135
1136  055245                    4: BEGIN PRMRYSTRNG[PM]↑.FATHER:=≡BOOLOFCHAR≡;    .NUC    1136   +4
1137  055252                       ALLOC(PRMRYSTRNG[PM]↑.LSON);                   .NUC    1137
1138  055265                       PRMRYSTRNG[PM]↑.LSON↑:=EXPSTRNG[EX]↑;          .NUC    1138
1139  055301                          END;                                        .NUC    1139   -4
1140  055302                    END;                                              .NUC    1140   -C
1141  055321                    EX:=EX-1; PRMRYSTRNG[PM]↑.RSON:=NIL;              .NUC    1141
1142  055331                    IF TOKEN=RIGHTPAREN THEN SCANNER ELSE             .NUC    1142
1143  055335                 ERROR(21); END END;                                  .NUC    1143   -3-2
1144  055340              END;                                                    .NUC    1144   -C
1145  055410           IF PFLAG≠0 THEN BEGIN ERROR(31); PFL:=1 END                .NUC    1145   +2-2
1146  055415        END;                                                          .NUC    1146   -1
1147  055426 PROCEDURE ALTERSEQ;                                                 .NUC    1147
1148  055426   ↑RECOGNIZE ALTERNATIVE SEQUENCE FOR CASE STATEMENTS↓              .NUC    1148
1149  055426        VAR I: INTEGER;                                             .NUC    1149
1150  055426        BEGIN IF (TOKEN=NUMBER) ∧ (¬MEMBER1(NUMVAL)) THEN BEGIN     .NUC    1150   +1+2
1151  055442           DEFCASELAB[CS][DC,1]:=NUMVAL;                            .NUC    1151
1152  055454           DEFCASELAB[CS][DC,2]:=PROCNAME.BOUND+POINT; DC:=DC+1;    .NUC    1152
1153  055471           SENTENCE(10,PROCNAME); SCANNER;                          .NUC    1153
1154  055475           2: IF TOKEN=COLON1 THEN SCANNER ELSE ERROR(30);          .NUC    1154
1155  055503              IF (TOKEN=NUMBER) ∧                                   .NUC    1155
1156  055506              (¬MEMBER1(NUMVAL)) THEN BEGIN                         .NUC    1156   +3
1157  055514                 DEFCASELAB[CS][DC,1]:=NUMVAL;                      .NUC    1157
1158  055526                 DEFCASELAB[CS][DC,2]:=PROCNAME.BOUND+POINT;        .NUC    1158
1159  055541                 DC:=DC+1; SENTENCE(10,PROCNAME); SCANNER;          .NUC    1159
1160  055547                 GOTO 2 END ELSE BEGIN                              .NUC    1160   -3+3
1161  055550                 I:=CJ; BODY; CJ:=I;                                .NUC    1161
1162  055555                 SENTENCE(1,PROCNAME);                              .NUC    1162
1163  055560                 POINT:=POINT+1; ALTERSEQ                           .NUC    1163
1164  055562              END END                                               .NUC    1164   -3-2
1165  055563        END;                                                          .NUC    1165   -1
1166  055566 PROCEDURE STATEMENT;                                                .NUC    1166
1167  055566   ↑RECOGNIZE STATEMENTS↓                                           .NUC    1167
1168  055566        VAR SFLAG, SFLAG1: 0..1;                                    .NUC    1168
1169  055566           I: INTEGER;                                              .NUC    1169
1170  055566           TMPB1: BOOLEAN;                                          .NUC    1170
1171  055566           TMPA1: BOOLEAN;                                          .NUC    1171
1172  055566        BEGIN                                                       .NUC    1172   +1
1173  055571        1: SFLAG:=1;                                                .NUC    1173
1174  055572           IF TNUM IN [30,13,8,20,27,15,3,14,17,21,26] THEN         .NUC    1174
1175  055575           CASE TNUM OF                                             .NUC    1175   +C
```

```
1176  055601 #IDENT+ 30: BEGIN SFLAG:=0; IDNAME:=TOKENSTRNG; SCANNER;                    .NUC   1176  +2
1177  055607          SFLAG1:=1;                                                        .NUC   1177
1178  055610          IF TNUM IN [51,49,34] THEN                                        .NUC   1178
1179  055613          CASE TNUM OF                                                      .NUC   1179  +C
1180  055617    #LABEL+    51: BEGIN SFLAG1:=0; IF(~MEMBER(IDNAME,DEFL,D))             .NUC   1180  +3
1181  055625                  THEN BEGIN DEFLABSET[D]:=IDNAME;                          .NUC   1181  +4
1182  055635                  DEFLABSET[D].BOUND:=PROCNAME.BOUND+POINT;                 .NUC   1182
1183  055646                  IF MEMBER(IDNAME,REFL,R) THEN BEGIN                       .NUC   1183  +5
1184  055653                  LTEMP:=DEFLABSET[D].BOUND;                                .NUC   1184
1185  055661                  I:=REFLABSET[PCNTR].BOUND;                                .NUC   1185
1186  055667                  IF SENTREE[I].LSON≠NIL THEN                               .NUC   1186
1187  055675                  REPEAT                                                    .NUC   1187  +R
1188  055675                     SAVE:=SENTREE[I].LSON↑.FATHER;                         .NUC   1188
1189  055702                     SENTREE[I].LSON↑.FATHER:=ALFI(LTEMP);                  .NUC   1189
1190  055714                  I:=INTI(SAVE) UNTIL SENTREE[I].LSON=NIL;                  .NUC   1190  -R
1191  055725                  ALLOC(SENTREE[I].LSON);                                   .NUC   1191
1192  055737                  SENTREE[I].LSON↑.FATHER:=ALFI(LTEMP);                     .NUC   1192
1193  055750                  SENTREE[I].LSON↑.LSON:=NIL;                               .NUC   1193
1194  055757                  END;                                                      .NUC   1194  -5
1195  055757                  D:=D+1; END ELSE ERROR(8); SCANNER;                       .NUC   1195  -4
1196  055764                  SENTENCE(7,PROCNAME);                                     .NUC   1196
1197  055767                  LTEMP:=5] END;                                            .NUC   1197  -3
1198  055771    #ARRAY+    49: BEGIN SFLAG1:=0; IF MEMBER(IDNAME,SYM,S)                 .NUC   1198  +3
1199  055775                  THEN BEGIN LFTARYNAME:=IDNAME;                            .NUC   1199  +4
1200  056003                     I:=ADDR;                                               .NUC   1200
1201  056005                     LEFTYPE:= LFTARYNAME.TYPEFNCTN;                        .NUC   1201
1202  056007                     END ELSE ERROR(10);                                    .NUC   1202  -4
1203  056011                     SCANNER; EXPRESSION;                                   .NUC   1203
1204  056013                     IF EFL>0 THEN EFL:=0 ELSE                              .NUC   1204
1205  056016                     BEGIN IF EXPTYPE[EX]=16 THEN BEGIN                     .NUC   1205  +4+5
1206  056024                     LEFTSIDE.FATHER:=LFTARYNAME.STRNG;                     .NUC   1206
1207  056025                     LEFTSIDE.INX:=I;                                       .NUC   1207
1208  056031                     ALLOC(LEFTSIDE.LSON);                                  .NUC   1208
1209  056040                     LEFTSIDE.LSON↑:=EXPSTRNG[EX]↑;                         .NUC   1209
1210  056050                     ALLOC(LEFTSIDE.RSON);                                  .NUC   1210
1211  056056                     I:=LFTARYNAME.BOUND;                                   .NUC   1211
1212  056060                     LEFTSIDE.RSON↑.FATHER:=ALFI(I);                        .NUC   1212
1213  056067                     LEFTSIDE.RSON↑.TY:=2;                                  .NUC   1213
1214  056073                     LEFTSIDE.RSON↑.LSON:=NIL;                              .NUC   1214
1215  056077                     END ELSE ERROR(19); EX:=EX-1;                          .NUC   1215  -5
1216  056103                     IF TOKEN=RIGHTBRACK THEN SCANNER                       .NUC   1216
1217  056105                     ELSE ERROR(5);                                         .NUC   1217
1218  056111                     IF TOKEN=COLEQ THEN BEGIN SCANNER; EXPRESSION;         .NUC   1218  +5
1219  056115                     IF EFL>0 THEN EFL:=0 ELSE BEGIN                        .NUC   1219  +6
1220  056120                     IF LEFTYPE=EXPTYPE[EX] THEN BEGIN                      .NUC   1220  +7
1221  056126                     SENTENCE(8,PROCNAME);                                  .NUC   1221
1222  056131                      POINT:=POINT+1;                                       .NUC   1222
1223  056133                     END ELSE ERROR(17);                                    .NUC   1223  -7
1224  056136                     EX:=EX-1;                                              .NUC   1224
1225  056140                  END END ELSE ERROR(22);                                   .NUC   1225  -6-5
1226  056143                  END END;                                                  .NUC   1226  -4-3
1227  056144    #ASSIGN+   34: BEGIN SFLAG1:=0; IF MEMBER(IDNAME,SYM,S) THEN         .NUC   1227  +3
1228  056152                  BEGIN LEFTSIDE.FATHER:=IDNAME.STRNG;                      .NUC   1228  +4
1229  056154                     LEFTSIDE.INX:=ADDR;                                    .NUC   1229
1230  056157                     LEFTSIDE.LSON:=NIL;                                    .NUC   1230
1231  056162                     LEFTSIDE.RSON:=NIL;                                    .NUC   1231
1232  056164                     LEFTYPE:=IDNAME.TYPEFNCTN;                             .NUC   1232
1233  056166                     END ELSE ERROR(9);                                     .NUC   1233  -4
1234  056171                     SCANNER; EXPRESSION;                                   .NUC   1234
1235  056173                     IF EFL>0 THEN EFL:=0 ELSE                              .NUC   1235
1236  056176                     BEGIN IF LEFTYPE=EXPTYPE[EX] THEN                      .NUC   1236  +4
1237  056204                  BEGIN SENTENCE(6,PROCNAME);                               .NUC   1237  +5
```

```
1238  056207                    POINT:=POINT+1 END ELSE ERROR(17);                    .NUC   1238   -5
1239  056214                    EX:=EX-1 END END;                                     .NUC   1239   -4-3
1240  056217                 END;                                                     .NUC   1240   -C
1241  056241                 IF SFLAG1≠0 THEN ERROR(6);                               .NUC   1241
1242  056245                 END;                                                     .NUC   1242   -2
1243  056246 ↑GOTO↓  13: BEGIN SFLAG1:=0; SCANNER; IF TOKEN=RESWRDS[24] THEN         .NUC   1243   +2
1244  056252                 BEGIN SCANNER; IF TOKEN=IDENTIFIER THEN BEGIN            .NUC   1244   +3+4
1245  056255                    TMPBL1:=MEMBER(TOKENSTRNG,DEFL,D);                    .NUC   1245
1246  056263                    IF ¬TMPBL1 THEN                                       .NUC   1246
1247  056264                    BEGIN TMPBL:=MEMBER(TOKENSTRNG,REFL,R);               .NUC   1247   +5
1248  056272                    IF ¬TMPBL THEN                                        .NUC   1248
1249  056273                    BEGIN                                                 .NUC   1249   +6
1250  056273                    TOKENSTRNG.BOUND:=PROCNAME.BOUND+POINT;               .NUC   1250
1251  056300                    REFLABSET[R]:=TOKENSTRNG; R:=R+1 END;                 .NUC   1251   -6
1252  056311                    END;                                                  .NUC   1252   -5
1253  056311                    SENTENCE1(28,PROCNAME);                               .NUC   1253
1254  056314                    IF (¬TMPBL) ∧ (¬TMPBL1) THEN SENTREE[SCTR].LSON:=NIL ELSE .NUC   1254
1255  056326                    IF (TMPBL) ∧ (¬TMPBL1) THEN                           .NUC   1255
1256  056331                    REFLABSET[PCNTR].BOUND:=PROCNAME.BOUND+POINT;         .NUC   1256
1257  056341                    POINT:=POINT+1;                                       .NUC   1257
1258  056343                    SCANNER END ELSE ERROR(1) END                         .NUC   1258   -4-3
1259  056347                    ELSE ERROR(23) END;                                   .NUC   1259   -2
1260  056353 ↑ENTER↓  8: BEGIN SFLAG:=0; SCANNER; IF TOKEN=IDENTIFIER THEN           .NUC   1260   +2
1261  056357                 BEGIN IF MEMBER(TOKENSTRNG,SYM,S) THEN ERROR(11);        .NUC   1261   +3
1262  056366                    SENTENCE1(29,PROCNAME);                               .NUC   1262
1263  056371                    SENTREE[SCTR].LSON↑.INX:=0;                           .NUC   1263
1264  056400                    REFPROCSET[RPR]:=TOKENSTRNG; RPR:=RPR+1; POINT:=POINT+1; .NUC   1264
1265  056413                    SCANNER END                                           .NUC   1265   -3
1266  056414                    ELSE ERROR(1) END;                                    .NUC   1266   -2
1267  056420 ↑I/O↓    20,27: BEGIN SFLAG:=0; SAVE:=TOKEN; SCANNER; IF TOKEN=IDENTIFIER.NUC   1267   +2
1268  056425                 THEN BEGIN IF ¬MEMBER(TOKENSTRNG,SYM,S) THEN ERROR(10);  .NUC   1268   +3
1269  056436                    IF TOKENSTRNG.TYPEFNCTN≠4 THEN ERROR(25);             .NUC   1269
1270  056443                    SENTENCE1(30,PROCNAME);                               .NUC   1270
1271  056446                    POINT:=POINT+1; SCANNER END ELSE                      .NUC   1271   -3
1272  056452                    ERROR(1) END;                                         .NUC   1272   -2
1273  056455 ↑IF↓     15: BEGIN SFLAG:=0; PUSH(IFPOINT,POINT,IFP);                   .NUC   1273   +2
1274  056463                 SCTR:=SCTR+1; SCANNER;                                   .NUC   1274
1275  056466                 EXPRESSION; IF EFL>0 THEN EFL:=0 ELSE BEGIN              .NUC   1275   +3
1276  056472                    IF EXPTYPE[EX]≠2 THEN ERROR(16);                      .NUC   1276
1277  056502                    IFE:=IFE+1;                                           .NUC   1277
1278  056504                    IFEXPR[IFE]:=EXPSTRNG[EX]; EX:=EX-1;                  .NUC   1278
1279  056516                    POINT:=POINT+1;                                       .NUC   1279
1280  056520                    IF TOKEN=RESWRDS[23] THEN BEGIN SCANNER; BODY;        .NUC   1280   +4
1281  056524                    SFLAG1:=1;                                            .NUC   1281
1282  056525                    IF TNUM IN [7,12] THEN                                .NUC   1282
1283  056530                    CASE TNUM OF                                          .NUC   1283   +C
1284  056534              ↑F1↓     12: BEGIN SFLAG1:=0; SENTENCE2(21,PROCNAME);      .NUC   1284   +5
1285  056540                       IFP:=IFP-1; IFE:=IFE-1; SCANNER END;               .NUC   1285   -5
1286  056545              ↑ELSE↓   7: BEGIN SFLAG1:=0; POINT:=POINT+1;               .NUC   1286   +5
1287  056550                       SCTR:=SCTR+1;                                      .NUC   1287
1288  056552                       PUSH(IFELSEPNT,POINT,IFEPT); SCANNER; BODY;        .NUC   1288
1289  056560                       IF TOKEN≠RESWRDS[12] THEN ERROR(26);               .NUC   1289
1290  056564                       SENTENCE2(22,PROCNAME); IFP:=IFP-1;                .NUC   1290
1291  056571                       IFE:=IFE-1; IFEPT:=IFEPT-1; SCANNER END;           .NUC   1291   -5
1292  056576                    END;                                                  .NUC   1292   -C
1293  056604                    IF SFLAG1≠0 THEN ERROR(26) END                        .NUC   1293   -4
1294  056610                    ELSE ERROR(27) END END;                               .NUC   1294   -3-2
1295  056614 ↑CASE↓    3: BEGIN SFLAG:=0; PUSH(CASEPOINT,POINT,CS);                  .NUC   1295   +2
1296  056622                 SCTR:=SCTR+1; SCANNER;                                   .NUC   1296
1297  056625                 EXPRESSION; IF EFL>0 THEN EFL:=0                         .NUC   1297
1298  056630                    ELSE BEGIN IF EXPTYPE[EX]≠16 THEN ERROR(19);          .NUC   1298   +3
1299  056641                    CASEXPR[CS]:=EXPSTRNG[EX]; EX:=EX-1;                  .NUC   1299
```

```
1300  056653                IF TOKEN=RESWRDS[18] THEN BEGIN
1301  056655                DC:=1; CJ:=0; POINT:=POINT+1;
1302  056661                SCANNER; ALTERSEQ;
1303  056663                PCNTR:=SCTR+1;
1304  056665                SENTENCE2(23,PROCNAME);
1305  056670                SFLAG1:=1;
1306  056671                IF TNUM IN [9,7] THEN
1307  056674                CASE TNUM OF
1308  056700          *ESAC*    9: BEGIN SFLAG1:=0; POINT:=POINT+DC-1;
1309  056704                        SENTENCE2(24,PROCNAME);
1310  056707                        FIX(CASEJOIN[CS],CJ);
1311  056717                        SENTENCE2(25,PROCNAME); CS:=CS-1; SCANNER END;
1312  056726          *ELSE*    7: BEGIN SFLAG1:=0; POINT:=POINT+DC-1;
1313  056732                        SENTENCE2(24,PROCNAME);
1314  056735                        FIX(CASEJOIN[CS],CJ);
1315  056745                        SCANNER; BODY;
1316  056747                        IF TOKEN#RESWRDS[9] THEN ERROR(33);
1317  056753                        SENTENCE2(25,PROCNAME); CS:=CS-1;
1318  056760                        SCANNER END;
1319  056762                END;
1320  056765                IF SFLAG1#0 THEN ERROR(33) END
1321  056771                ELSE ERROR(15) END END;
1322  056775 *HALT*   14: BEGIN SFLAG:=0;
1323  056776                SENTENCE(6,PROCNAME);
1324  057001                   POINT:=POINT+1; SCANNER END;
1325  057005 *NOP*    17: BEGIN SFLAG:=0;
1326  057006                SENTENCE(5,PROCNAME);
1327  057011                   POINT:=POINT+1; SCANNER END;
1328  057015 *RTN*    21: BEGIN SFLAG:=0;
1329  057016                SENTENCE(4,PROCNAME);
1330  057021                   POINT:=POINT+1; SCANNER END;
1331  057025 *WHILE*  26: BEGIN SFLAG:=0; PUSH(WHILEPOINT,POINT,WH);
1332  057033                SCTR:=SCTR+1; SCANNER;
1333  057036                EXPRESSION; IF EFL>0 THEN EFL:=0 ELSE
1334  057042                   BEGIN IF EXPTYPE[EX]#2 THEN ERROR(16);
1335  057052                   WHILEXPR[WH]:=EXPSTRNG[EX]; EX:=EX-1;
1336  057064                   POINT:=POINT+1;
1337  057066                   IF TOKEN=RESWRDS[5] THEN BEGIN SCANNER; BODY;
1338  057072                   IF TOKEN#RESWRDS[6] THEN ERROR(24);
1339  057076                   SENTENCE1(3,PROCNAME);
1340  057101                   POINT:=POINT+1; SENTENCE(9,PROCNAME); WH:=WH-1;
1341  057110                   SCANNER END ELSE ERROR(28) END END;
1342  057115             END;
1343  057151             IF LTEMP=5] THEN BEGIN LTEMP:=0; GOTO 1 END;
1344  057154             IF SFLAG#0 THEN ERROR(6);
1345  057160             IF TOKEN#SEMI] THEN ERROR(2)
1346  057163                ELSE SCANNER
1347  057165          END;
1348  057174 PROCEDURE BODY;
1349  057174    *RECOGNIZE BODY*
1350  057174       BEGIN
1351  057177       1: IF ¬(TNUM IN [6,7,9,10,12,31]) THEN BEGIN
1352  057202             IF TOKEN=ASSERTION THEN BEGIN
1353  057204             SCANNER; GOTO 1 END ELSE BEGIN STATEMENT;
1354  057207             GOTO 1 END END
1355  057210          END;
1356  057214 PROCEDURE PROCSEQ;
1357  057214    *RECOGNIZE PROCEDURES*
1358  057214       PROCEDURE OUTPROG;
1359  057214          VAR I: INTEGER;
1360  057214          BEGIN FOR I:=0 TO SCTR DO BEGIN VPROG*:=SENTREE[I];
1361  057232             PUT(VPROG) END;
```

```
.NUC  1300  *4
.NUC  1301
.NUC  1302
.NUC  1303
.NUC  1304
.NUC  1305
.NUC  1306
.NUC  1307  *C
.NUC  1308  *5
.NUC  1309
.NUC  1310
.NUC  1311  -5
.NUC  1312  *5
.NUC  1313
.NUC  1314
.NUC  1315
.NUC  1316
.NUC  1317
.NUC  1318  -5
.NUC  1319  -C
.NUC  1320  -4
.NUC  1321  -3-2
.NUC  1322  *2
.NUC  1323
.NUC  1324  -2
.NUC  1325  *2
.NUC  1326
.NUC  1327  -2
.NUC  1328  *2
.NUC  1329
.NUC  1330  -2
.NUC  1331  *2
.NUC  1332
.NUC  1333
.NUC  1334  *3
.NUC  1335
.NUC  1336
.NUC  1337  *4
.NUC  1338
.NUC  1339
.NUC  1340
.NUC  1341  -4-3-2
.NUC  1342  -C
.NUC  1343  *2-2
.NUC  1344
.NUC  1345
.NUC  1346
.NUC  1347  -1
.NUC  1348
.NUC  1349
.NUC  1350  *1
.NUC  1351  *2
.NUC  1352  *3
.NUC  1353  -3*3
.NUC  1354  -3-2
.NUC  1355  -1
.NUC  1356
.NUC  1357
.NUC  1358
.NUC  1359
.NUC  1360  *1*2
.NUC  1361  -2
```

```
1362  057237          END;                                                                                    .NUC  1362  -1
1363  057241        BEGIN                                                                                     .NUC  1363  +1
1364  057244          IF TOKEN=RESWRDS[19] THEN BEGIN SCANNER;                                                .NUC  1364  +2
1365  057247             TOKENSTRNG.BOUND:=PROCNAME.BOUND+SCTR+1;                                             .NUC  1365
1366  057255             TOKENSTRNG.TYPEFNCTN:=19;                                                            .NUC  1366
1367  057260             SCTR:=-1;                                                                            .NUC  1367
1368  057261             IF (TOKEN=IDENTIFIER) ^ (~MEMBER(TOKENSTRNG,SYM,P)) THEN                             .NUC  1368
1369  057274                BEGIN SYMTAB[P]:=TOKENSTRNG; P:=P+1;                                              .NUC  1369  +3
1370  057305                   PROCNAME:=TOKENSTRNG; POINT:=0; D:=1; R:=1;                                    .NUC  1370
1371  057314                   KR:=0;                                                                         .NUC  1371
1372  057315                   SCANNER; IF TOKEN=SEMI1 THEN SCANNER ELSE ERROR(2);                            .NUC  1372
1373  057324                   BODY;                                                                          .NUC  1373
1374  057325                   IF (TOKEN=RESWRDS[10]) ^ (SUBSET(REFL)) THEN BEGIN                             .NUC  1374  +4
1375  057335                   SENTENCE(3,PROCNAME);                                                          .NUC  1375
1376  057340                   SCANNER; IF TOKEN=SEMI1 THEN BEGIN                                             .NUC  1376  +5
1377  057343                   WRITE(=1=,EOL); SCANNER;                                                       .NUC  1377
1378  057350                   END ELSE ERROR(2);                                                             .NUC  1378  -5
1379  057353                   CHECK:=CHECK+1; FIX(KEEPRTN,KR);                                               .NUC  1379
1380  057361                   OUTPROG;                                                                       .NUC  1380
1381  057363                   PROCSEQ END                                                                    .NUC  1381  -4
1382  057364                   ELSE IF TOKEN#RESWRDS[10] THEN BEGIN ERROR(13);                                .NUC  1382  +4
1383  057371                   OUTPROG;                                                                       .NUC  1383
1384  057373                   PROCSEQ END ELSE BEGIN ERROR(14);                                              .NUC  1384  -4+4
1385  057377                   OUTPROG; PROCSEQ END END                                                       .NUC  1385  -4-3
1386  057402                   ELSE                                                                           .NUC  1386
1387  057403                   IF TOKEN#IDENTIFIER THEN ERROR(1) ELSE ERROR(11) END                           .NUC  1387  -2
1388  057412          ELSE IF CHECK=0 THEN ERROR(12)                                                          .NUC  1388
1389  057416        END;                                                                                      .NUC  1389  -1
1390  057422  PROCEDURE DECSEQ;                                                                             .NUC  1390
1391  057422    *SCAN DECLARATIONS AND SET UP SYMBOL TABLE*                                                   .NUC  1391
1392  057422        VAR DFLAG: 0..1;                                                                          .NUC  1392
1393  057422        BEGIN DFLAG:=1;                                                                           .NUC  1393  +1
1394  057426          IF TNUM IN [2,4,16] THEN BEGIN                                                          .NUC  1394  +2
1395  057431          DFLAG:=0; TYPES:=TNUM; SCANNER; IF TOKEN#RESWRDS[1] THEN BEGIN                          .NUC  1395  +3
1396  057436          REPEAT BEGIN IF DFLAG=0 THEN DFLAG:=1 ELSE SCANNER;                                     .NUC  1396  +R+4
1397  057442             IF (TOKEN=IDENTIFIER) ^ (~MEMBER(TOKENSTRNG,SYM,S))                                  .NUC  1397
1398  057454             THEN BEGIN SENTENCE(1,TOKENSTRNG);                                                   .NUC  1398  +5
1399  057460             SYMTAB[S].STRNG:=TOKENSTRNG.STRNG;                                                   .NUC  1399
1400  057466             SYMTAB[S].NUM:=TOKENSTRNG.NUM;                                                       .NUC  1400
1401  057475             SYMTAB[S].PTR:=TOKENSTRNG.PTR;                                                       .NUC  1401
1402  057503             SYMTAB[S].BOUND:=-1;                                                                 .NUC  1402
1403  057513             SYMTAB[S].TYPEFNCTN:=TYPES; S:=S+1; SCANNER END ELSE                                 .NUC  1403  -5
1404  057526             ERROR(29) END UNTIL TOKEN#COMMA;                                                     .NUC  1404  -4-R
1405  057532             IF TOKEN=SEMI1 THEN BEGIN SCANNER; DECSEQ END                                        .NUC  1405  +4-4
1406  057536             ELSE BEGIN ERROR(2); DECSEQ END                                                      .NUC  1406  +4-4
1407  057542             END ELSE BEGIN                                                                       .NUC  1407  -3+3
1408  057543          REPEAT BEGIN SCANNER; IF (TOKEN=IDENTIFIER) ^                                           .NUC  1408  +R+4
1409  057547             (~MEMBER(TOKENSTRNG,SYM,S)) THEN BEGIN                                               .NUC  1409  +5
1410  057557                ARRAYNAME.STRNG:=TOKENSTRNG.STRNG;                                                .NUC  1410
1411  057560                ARRAYNAME.NUM:=TOKENSTRNG.NUM;                                                    .NUC  1411
1412  057564                ARRAYNAME.PTR:=TOKENSTRNG.PTR; SCANNER;                                           .NUC  1412
1413  057570                IF TOKEN=LEFTBRACK THEN SCANNER ELSE ERROR(3);                                    .NUC  1413
1414  057576                IF TOKEN=NUMBER                                                                   .NUC  1414
1415  057577                THEN BEGIN ARRAYNAME.TYPEFNCTN:=TYPES;                                            .NUC  1415  +6
1416  057604                   IF NUMVAL<=MAXARRAY THEN ARRAYNAME.BOUND:=NUMVAL                               .NUC  1416
1417  057606                   ELSE ERROR(34);                                                                .NUC  1417
1418  057613                   ARRAYSIZE:=ARRAYSIZE+NUMVAL;                                                   .NUC  1418
1419  057615                   SENTENCE(2,ARRAYNAME);                                                         .NUC  1419
1420  057620                   SYMTAB[S]:=ARRAYNAME; S:=S+1; SCANNER;                                         .NUC  1420
1421  057633                   IF TOKEN=RIGHTBRACK THEN SCANNER ELSE                                          .NUC  1421
1422  057637                   ERROR(5); END ELSE                                                             .NUC  1422  -6
1423  057642                BEGIN ERROR(4); DECSEQ END END ELSE                                               .NUC  1423  +6-6-5
```

```
1424  057646              BEGIN ERROR(29)¦ DECSEQ END END
1425  057651            UNTIL TOKEN≠COMMA)¦
1426  057653            IF TOKEN=SEMI) THEN BEGIN SCANNER¦ DECSEQ END ELSE
1427  057660          BEGIN ERROR(2)¦ DECSEQ END END END
1428  057663        END¦
1429  057670 PROCEDURE PASS1¦
1430  057670 BEGIN S¦=1¦ RPR¦=1¦ SCANNER¦
1431  057676       ERRSET¦=[]¦
1432  057677       DECSEQ¦ P1=S¦
1433  057702       PROCNAME.BOUND¦=0¦
1434  057705       RESET(VPROG)¦
1435  057706       ALLOC(SAVEPTR)¦ PROCSEQ¦
1436  057714       PCNTR¦=P+C-2¦
1437  057717       IF SUBSET(PR) THEN BEGIN
1438  057722       IF TOKEN=RESWRDS(22) THEN BEGIN SCANNER¦
1439  057725          IF (TOKEN≠IDENTIFIER) v (¬MEMBER(TOKENSTRNG,SYM,P)) THEN
1440  057740          ERROR(9) END ELSE ERROR(7) END ELSE ERROR(32)¦
1441  057750          RESET(VPROG)¦
1442  057752          IF ERRSET≠[] THEN ERRPRINT¦
1443  057755          PCNTR¦=0¦ COLCNT¦=1¦
1444  057757          IF ΞSΞ IN OPFLAG THEN DUMPSYM(COLCNT,S)¦
1445  057764          IF ΞCΞ IN OPFLAG THEN DUMPCONS¦
1446  057767          IF ΞPΞ IN OPFLAG THEN DUMPSYM(S,P)¦
1447  057774          IF ΞVΞ IN OPFLAG THEN
1448  057776          DUMPTREE¦
1449  057777       WRITE(Ξ1Ξ,EOL)¦
1450  060003 END¦ ¤PASS1¤
1451  060006 BEGIN OPFLAG¦:=[]¦
1452  060016       PASS1¦
1453  060017       OVERLAY(ΞNUC2Ξ,0,1)¦
1454  060022 END.
```

```
.NUC    1424    +5-5-4
.NUC    1425    -R
.NUC    1426    +4-4
.NUC    1427    +4-4-3-2
.NUC    1428    -1
.NUC    1429
.NUC    1430    +1
.NUC    1431
.NUC    1432
.NUC    1433
.NUC    1434
.NUC    1435
.NUC    1436
.NUC    1437    +2
.NUC    1438    +3
.NUC    1439
.NUC    1440    -3-2
.NUC    1441
.NUC    1442
.NUC    1443
.NUC    1444
.NUC    1445
.NUC    1446
.NUC    1447
.NUC    1448
.NUC    1449
.NUC    1450    -1
.NUC    1451    +1
.NUC    1452
.NUC    1453
.NUC    1454    -1
```

```
PASCAL 6600    VERS. APRIL 73                                  19 APR 73 10.20.14.

1  006001 *******   PASS II: CODE GENERATION AND EXECUTION   *******;                .NUC1      1
2  006001                                                                             .NUC1      2
3  006001 CONST BLANK=Ξ Ξ; ASSERT=ΞASSERTΞ; ASSERTION=ΞASSERTIONΞ;                    .NUC1      3
4  006001       IDENTIFIER=ΞIDENTIFIERΞ; NUMBER=ΞNUMBERΞ;                          .NUC1      4
5  006001       PR=ΞPΞ; SYM=ΞSΞ; DEFL=ΞLΞ; REFL=ΞRΞ;                               .NUC1      5
6  006001       DGT=Ξ9Ξ; QUOTE=Ξ'Ξ; DLR=Ξ$Ξ; SEMI=Ξ;Ξ; ENDFILE=ΞEOFΞ;             .NUC1      6
7  006001       COLEQ=Ξ:=Ξ; EQL=Ξ=Ξ; LTPA=ΞAΞ; CMA=Ξ,Ξ; CLN=Ξ:Ξ; LP=Ξ(Ξ;           .NUC1      7
8  006001       RP=Ξ)Ξ; DRPR=Ξ))Ξ; PREQ=Ξ)=Ξ; ZER=Ξ0Ξ;                           .NUC1      8
9  006001       SIMPLE=ΞSIMPLEΞ; ARRY=ΞARRAYΞ; EXITS=ΞEXITΞ; JUMPTO=ΞJUMPTOΞ;      .NUC1      9
10 006001       IFS=ΞIFΞ; EXITPT=ΞEXITPOINTΞ; HALTS=ΞHALTΞ; ASSIGNS=ΞASSIGNΞ;      .NUC1     10
11 006001       POINTLABEL=ΞPOINTLABELΞ; POINTLBL1=ΞLEDWITHΞ; ENTERS=ΞENTERΞ;      .NUC1     11
12 006001       CASES=ΞCASEΞ; CASEJOIN1=ΞCASEJOINPOΞ; CASEJOIN2=ΞINTΞ;             .NUC1     12
13 006001       INITPROC1=ΞINITIALPROΞ; INITPROC2=ΞCEDURE=Ξ;                       .NUC1     13
14 006001       CHARCONST=ΞCHARCONSTΞ; MAXARRAY=777777777B;                         .NUC1     14
15 006001       STMAX=100; STKLIM=6030B;                                           .NUC1     15
16 006001       MASK=-777777B;                                                     .NUC1     16
17 006001       NOP=460008; CODMAX=1000;                                           .NUC1     17
18 006001       JMPMAX=77B;                                                        .NUC1     18
19 006001 TYPE IDWORD=PACKED RECORD                                                .NUC1     19
20 006001                    STRNG: ALFA;                                          .NUC1     20
21 006001                    NUM: 1..60;                                           .NUC1     21
22 006001                    TYPEFNCTN: 1..20;                                     .NUC1     22
23 006001                    BOUND: -1..MAXARRAY;                                  .NUC1     23
24 006001                    PTR: ↑ENDSTRNG END;                                   .NUC1     24
25 006001   *IDWORD: IDENTIFIERS STORED WITH THIS TYPE                             .NUC1     25
26 006001         STRNG: FIRST 10 CHARS OF IDENT                                   .NUC1     26
27 006001         NUM: LENGTH OF IDENT                                             .NUC1     27
28 006001         TYPEFNCTN: 2: BOOLEAN                                            .NUC1     28
29 006001                    4: CHARACTER                                          .NUC1     29
30 006001                   16: INTEGER                                            .NUC1     30
31 006001                   19: PROCEDURE                                          .NUC1     31
32 006001         BOUND: -1: SIMPLE IDENT                                          .NUC1     32
33 006001         ≥0: ARRAY BOUND FOR ARRAY VAR                                    .NUC1     33
34 006001             PROCEDURE VIRTUAL ADDR FOR PROCEDURE NAMES                   .NUC1     34
35 006001         PTR: POINTER TO END OF IDENT IF NUM>10↓                           .NUC1     35
36 006001      TPTR=↑TREENODE;                                                     .NUC1     36
37 006001      STACKA=ARRAY[1..25] OF TPTR;                                        .NUC1     37
38 006001      STACKI=ARRAY[1..25] OF INTEGER;                                     .NUC1     38
39 006001      STACKC=ARRAY[1..25] OF INTEGER;                                     .NUC1     39
40 006001      CLSTK=ARRAY[1..25,1..2] OF INTEGER;                                 .NUC1     40
41 006001      ERRTYPE=SET OF 1..34;                                               .NUC1     41
42 006001      OPTYPE=SET OF ΞAΞ..ΞZΞ;                                             .NUC1     42
43 006001      SCNT=0..25;                                                         .NUC1     43
44 006001      SHRTINT=-377777B..+377777B;                                         .NUC1     44
45 006001       ADDRESS=0..177777B;                                                .NUC1     45
46 006001      TREEWORD=PACKED RECORD                                              .NUC1     46
47 006001                    FATHER: ALFA;                                         .NUC1     47
48 006001                    TY: 0..3;                                             .NUC1     48
49 006001                    INX:  SHRTINT;                                        .NUC1     49
50 006001                    LSON,RSON: TPTR END;                                  .NUC1     50
51 006001 VAR ENDSTRNG: CLASS 1000 OF ARRAY[1..5] OF ALFA;                         .NUC1     51
52 017612         *ENDSTRNG: FOR END OF IDENT>10 CHARS LONG↓                       .NUC1     52
53 017612      TREENODE: CLASS 1000 OF TREEWORD;                                   .NUC1     53
54 023533         *TREENODE: FOR EXPRESSION TREES AND REDUCED PROGRAM TREES↓       .NUC1     54
55 023533      SYNTAB: ARRAY[1..500] OF IDWORD;                                    .NUC1     55
56 025503      CONSTANT: ARRAY[1..250] OF INTEGER; C: INTEGER;                     .NUC1     56
57 026076      VPROC: FILE OF TREEWORD;                                            .NUC1     57
58 027100      SENTREE: ARRAY[0..249] OF TREEWORD;                                 .NUC1     58
59 030064      REFPROCSET: ARRAY[1..150] OF IDWORD;                                .NUC1     59
```

```
 60  030540       DEFCASELAB: ARRAY[1..25] OF CLSTK;                                    .NUC1      60
 61  033102     CASEJOIN: ARRAY[1..10] OF STACKC;                                       .NUC1      61
 62  033474     DEFLABSET: ARRAY[1..200] OF IDWORD;                                     .NUC1      62
 63  034314     REFLABSET: ARRAY[1..100] OF IDWORD;                                     .NUC1      63
 64  034624        *TABLES FOR IDENTIFIERS, CONSTANTS, REDUCED PROGRAM TREES            .NUC1      64
 65  034624           CASE LABELS, LABELS+                                              .NUC1      65
 66  034624     SAVEPTR: ^TREENODE;                                                     .NUC1      66
 67  034625     LEFTSIDE: TREEWORD;                                                     .NUC1      67
 68  034627     LFTARYNAME,IDNAME,TOKENSTRNG,PROCNAME: IDWORD;                          .NUC1      68
 69  034637     ARRAYNAME: IDWORD; NUMVAL: INTEGER;                                     .NUC1      69
 70  034642     ANDTYPE, BNRYADTYPE, EXPTYPE, MULTYPE, NOTTYPE, PRMRYTYPE, RELTYPE,     .NUC1      70
 71  034642     UNRYADTYPE, CASEPOINT, IFELSEPNT, IFPOINT, WHILEPOINT: STACKI;          .NUC1      71
 72  035316     ANDSTRNG, BNRYADSTNG, CASEXPR, EXPSTRNG, IFEXPR, MULSTRNG,              .NUC1      72
 73  035316     NOTSTRNG, PRMRYSTRNG, RELSTRNG, UNRYADSTNG, WHILEXPR: STACKA;           .NUC1      73
 74  035741        *TYPE STACKS AND EXPRESSION STACKS: FOR NESTED EXPRESSIONS,          .NUC1      74
 75  035741           CASE STATEMENTS, WHILE STATEMENTS, ETC+                           .NUC1      75
 76  035741     S,P,RPR,DC,POINT,COLCNT,TYPES,LEFTYPE: INTEGER;                         .NUC1      76
 77  035751     TNUM: 1..63;                                                            .NUC1      77
 78  035752        *TNUM: HOLDS REFERENCE NUMBER OF TOKEN                               .NUC1      78
 79  035752              1-27: INDICATES INDEX OF RESERVED WORD IN RESERVED             .NUC1      79
 80  035752                    WORD TABLE                                               .NUC1      80
 81  035752              30: IDENTIFIER TOKEN                                           .NUC1      81
 82  035752              31: NUMBER TOKEN                                               .NUC1      82
 83  035752              32: CHARACTER CONSTANT TOKEN                                   .NUC1      83
 84  035752              33: EOF TOKEN                                                  .NUC1      84
 85  035752              34: = := = TOKEN                                               .NUC1      85
 86  035752              37-63: CORRESPONDING DECIMAL EQUIVALENT OF DISPLAY CODE        .NUC1      86
 87  035752                     FOR SPECIAL CHARACTERS+                                 .NUC1      87
 88  035752     IFEPT,IFP,AN,BN,CS,EX,IFE,MUL,NT,PM,REL,UN,WH: 0..25;                   .NUC1      88
 89  035767     TOKEN,SAVE: ALFA; D,R: 1..100;                                          .NUC1      89
 90  035773     RESWRDS: ARRAY[1..27] OF ALFA; CARD: ARRAY[1..80] OF CHAR;              .NUC1      90
 91  036146        *RESERVED WORD TABLE+                                                .NUC1      91
 92  036146        *CHARACTER ARRAY TO HOLD CONTENTS OF CARD CURRENTLY                   .NUC1      92
 93  036146           BEING SCANNED+                                                    .NUC1      93
 94  036146     SIXTEEN, FOUR, TWO: INTEGER;                                            .NUC1      94
 95  036151     CHECK: INTEGER;                                                         .NUC1      95
 96  036152     PFL,UFL,MFL,BFL,RFL,NFL,AFL,EFL: INTEGER;                               .NUC1      96
 97  036162     SCTR: -1..499;                                                          .NUC1      97
 98  036163     KEEPRTN: STACKC; CJ,KR: 0..25;                                          .NUC1      98
 99  036216     LTEMP:INTEGER;                                                          .NUC1      99
100  036217     ERRSET: ERRTYPE;                                                        .NUC1     100
101  036220     OPFLAG: OPTYPE;                                                         .NUC1     101
102  036221     ARRAYSIZE: INTEGER;                                                     .NUC1     102
103  036222     SENFLAG: 0..1;                                                          .NUC1     103
104  036223     SYMSIZE,CODADDR,STSYM,STPROC,PCNTR,ADDR: SHRTINT;                       .NUC1     104
105  036231     LT, LE, GT, GE, EQ, NE, NOT, AND, OR: ALFA;                             .NUC1     105
106  036242     COLON1,SEMI1,COMMA1,PLUS,MINUS,TIMES,SLASH,EXPO,                        .NUC1     106
107  036242     LEFTBRACK,RIGHTBRACK,LEFTPAREN,RIGHTPAREN: ALFA;                        .NUC1     107
108  036256     STCOR: SHRTINT;                                                         .NUC1     108
109  036257     BSRCH,TCODE: ADDRESS;                                                   .NUC1     109
110  036261     READCH,*RITECH: ADDRESS;                                                .NUC1     110
111  036263     IC: ADDRESS;                                                            .NUC1     111
112  036264     PC: INTEGER;                                                            .NUC1     112
113  036265     JMPTAB: ARRAY[0..JMPMAX] OF INTEGER;                                    .NUC1     113
114  036365     JMPIX: SHRTINT;                                                         .NUC1     114
115  036366     CODE: ARRAY[0..CODMAX] OF INTEGER;                                      .NUC1     115
116  040337     IT: INTEGER;                                                            .NUC1     116
117  040340     CA,CP: SHRTINT; BUF: INTEGER;                                           .NUC1     117
118  040343     PASCLGO[OUT]: FILE OF INTEGER;                                          .NUC1     118
119  041344     LASTOP,LASTI: SHRTINT;                                                  .NUC1     119
120  041346     PRCODE: BOOLEAN;                                                        .NUC1     120
121  041347     JB1,JB2,JX1,JX2,RL,FS,R1: ARRAY[0..7] OF CHAR;                          .NUC1     121
```

```
122  041437      SH: ARRAY[12..23] OF CHAR;
123  041453 PROCEDURE ERROR(NUM): INTEGER);
124  041577   (*PRINT ERROR MESSAGE NUMBERS AND MAKE NECESSARY RECOVERY*)
125  041577      CONST STARS='*****';
126  041577      BEGIN WRITE(BLANK,STARS:4,NUM:3,BLANK,STARS,EOL);
127  041616         ERRSET:=ERRSET v[NUM];
128  041621      END;
129  041624 FUNCTION ALFI(PT: INTEGER): ALFA;
130  041624   (*CHANGE INTEGER CONSTANT TO ALFA EQUIVALENT*)
131  041624      VAR TMP: ARRAY[1..10] OF CHAR; I,J: INTEGER;
132  041624         TMP1: ALFA;
133  041624      BEGIN FOR I:=1 TO 10 DO TMP[I]:=BLANK;
134  041642         J:=PT; I:=10;
135  041645         REPEAT TMP[I]:=CHR((J MOD 10) + 27);
136  041655            J:=J DIV 10; I:=I-1
137  041661         UNTIL J=0;
138  041664         PACK(TMP,1,TMP1);
139  041674         ALFI:=TMP1
140  041674      END;
141  041677 FUNCTION INTI(WD: ALFA): INTEGER;
142  041677   (*CHANGE ALFA VALUE TO INTEGER EQUIVALENT*)
143  041677      VAR TMPCH: ARRAY[1..10] OF CHAR; I,J: INTEGER;
144  041677      BEGIN UNPACK(WD,TMPCH,1);
145  041712         J:=0; I:=1;
146  041714         WHILE (TMPCH[I]=BLANK) ^ (I≤10) DO I:=I+1;
147  041731         REPEAT J:=10*J+(INT(TMPCH[I])-27); I:=I+1
148  041741         UNTIL I>10;
149  041745         INTI:=J;
150  041746      END;
151  041750    PROCEDURE OUTCH(C:CHAR);
152  041750    BEGIN OUTPUT^ := C; PUT(OUTPUT)  END;
153  041760  (* CODE GENERATING ROUTINES *)
154  041760
155  041760    PROCEDURE NOOP;
156  041760    BEGIN WHILE CP < 4 DO
157  041765       BEGIN APPEND(BUF,15,NOP);
158  041767          CP := CP + 1;
159  041771       END ;
160  041771    END  (* NOOP *)  ;
161  041773
162  041773    PROCEDURE GEN15(OP,I,J,K : SHRTINT);
163  041773    BEGIN LASTOP := OP ; LASTI := I ;
164  042001       IF CP ≠ 4 THEN
165  042003       BEGIN CP := CP + 1; APPEND(BUF,6,OP) END ELSE
166  042007       BEGIN CODE[CA] := BUF;  BUF := OP;  CP := 1;
167  042015          CA := CA + 1;  IC := IC + 1;
168  042021          IF CA > CODMAX THEN
169  042023          BEGIN ERROR(35); CA:=1; BUF:=NOP;
170  042027             GOTO 1;
171  042030          END;
172  042030       END;
173  042030       BUF := ((8*BUF + I)*8 + J)*8 + K ;
174  042035  1:END  (* GEN15 *) ;
175  042037
176  042037    PROCEDURE GEN30(OP,I,J,K : SHRTINT);
177  042037    BEGIN LASTOP := OP ; LASTI := I ;
178  042045       IF CP = 3 THEN (*NOOP*)
179  042047       BEGIN APPEND(BUF,15,NOP);  CP := 4 END;
180  042052       IF CP < 4 THEN
181  042054       BEGIN CP := CP + 2;  APPEND(BUF,6,OP)  END ELSE
182  042060       BEGIN CODE[CA] := BUF;  BUF := OP;  CP := 2;
183  042067          CA := CA + 1;  IC := IC + 1;
```

```
.NUC1    122
.NUC1    123
.NUC1    124
.NUC1    125
.NUC1    126    +1
.NUC1    127
.NUC1    128    -1
.NUC1    129
.NUC1    130
.NUC1    131
.NUC1    132
.NUC1    133    +1
.NUC1    134
.NUC1    135    +R
.NUC1    136
.NUC1    137    -R
.NUC1    138
.NUC1    139
.NUC1    140    -1
.NUC1    141
.NUC1    142
.NUC1    143
.NUC1    144    +1
.NUC1    145
.NUC1    146
.NUC1    147    +R
.NUC1    148    -R
.NUC1    149
.NUC1    150    -1
.NUC1    151
.NUC1    152    +1-1
.NUC1    153
.NUC1    154
.NUC1    155
.NUC1    156    +1
.NUC1    157    +2
.NUC1    158
.NUC1    159    -2
.NUC1    160    -1
.NUC1    161
.NUC1    162
.NUC1    163    +1
.NUC1    164
.NUC1    165    +2-2
.NUC1    166    +2
.NUC1    167
.NUC1    168
.NUC1    169    +3
.NUC1    170
.NUC1    171    -3
.NUC1    172    -2
.NUC1    173
.NUC1    174    -1
.NUC1    175
.NUC1    176
.NUC1    177    +1
.NUC1    178
.NUC1    179    +2-2
.NUC1    180
.NUC1    181    +2-2
.NUC1    182    +2
.NUC1    183
```

```
184  042072          IF CA > CODMAX THEN                                          .NUC1   184
185  042074         BEGIN ERROR(35)$ CA:=1$ BUF:=NOP$                             .NUC1   185    +3
186  042100             CP := 1$  GOTO 1$                                         .NUC1   186
187  042102          END$                                                         .NUC1   187    -3
188  042102        END$                                                           .NUC1   188    -2
189  042102        BUF := (8*BUF + I)*8 + J $                                     .NUC1   189
190  042106        IF K ≥ 0 THEN APPEND(BUF,18,K) ELSE APPEND(BUF,18,7777778+K) $ .NUC1   190
191  042114  1$END  * GEN30 + $                                                   .NUC1   191    -1
192  042117                                                                       .NUC1   192
193  042117     PROCEDURE BXIXJ(I, J : SHRTINT) $                                 .NUC1   193
194  042117     *TO AVOID B XI XJ INSTRUCTIONS WHENEVER POSSIBLE+                 .NUC1   194
195  042117     BEGIN IF (J = LASTI)^(LASTOP ≥ 108)^(LASTOP ≤ 458)                .NUC1   195    +1
196  042131        ^¬(LASTOP IN [208,218,438]) THEN                               .NUC1   196
197  042136        BEGIN IF BUF ≥ 0 THEN BUF := BUF - LASTI*64 + I*64 ELSE        .NUC1   197    +2
198  042142           BEGIN BUF := - BUF $                                        .NUC1   198    +3
199  042144              BUF := BUF - (7 - LASTI)*64 + (7 - I)*64 $               .NUC1   199
200  042151              BUF := - BUF                                             .NUC1   200
201  042151           END $                                                       .NUC1   201    -3
202  042152           LASTI := I                                                  .NUC1   202
203  042152        END ELSE GEN15(108,I,J,0)                                      .NUC1   203    -2
204  042161     END *BXIXJ+ $                                                     .NUC1   204    -1
205  042165                                                                       .NUC1   205
206  042165     PROCEDURE INS(FADR : ADDRESS $ FCP, FCA : SHRTINT) $              .NUC1   206
207  042165     BEGIN IF CA ≠ FCA THEN INSERT(FADR,(4-FCP)*15,CODE(FCA))          .NUC1   207    +1
208  042201                           ELSE INSERT(FADR,(CP-FCP)*15,BUF)           .NUC1   208
209  042205     END $                                                             .NUC1   209    -1
210  042207                                                                       .NUC1   210
211  042207  * ------------------------------------------------------------- +   .NUC1   211
212  042207                                                                       .NUC1   212
213  042207  * ROUTINES TO PRINT CODE AND JUMPTABLE AND TO OUTPUT CODE +          .NUC1   213
214  042207                                                                       .NUC1   214
215  042207     PROCEDURE PRTCOMP$                                                .NUC1   215
216  042207         * GLOBAL ARRAYS : JB1,JB2,JX1,JX2,REL,FS,R1,SH +              .NUC1   216
217  042207        VAR II,JJ,KK : CHAR$                                           .NUC1   217
218  042207           C,CC,S,IT3, OP,I,J,K : SHRTINT$                             .NUC1   218
219  042207           M,IT4 : SHRTINT$                                            .NUC1   219
220  042207                                                                       .NUC1   220
221  042207        PROCEDURE SPACE$                                               .NUC1   221
222  042207        BEGIN FOR IT := 1 TO 3 DO                                      .NUC1   222    +1
223  042216           BEGIN OUTPUT+ := Ξ Ξ$  PUT(OUTPUT)  END$                    .NUC1   223    +2-2
224  042224        END$                                                           .NUC1   224    -1
225  042226                                                                       .NUC1   225
226  042226        BEGIN                                                          .NUC1   226    +1
227  042231           IC := IC - CA - 1$                                          .NUC1   227
228  042234           M := -778$  IT4 := -78$                                     .NUC1   228
229  042237           FOR IT3 := 1 TO CA DO                                       .NUC1   229
230  042243           BEGIN BUF := -CODE(IT3)$  OUTCH(EOL)$                       .NUC1   230    +2
231  042251                WRITE(Ξ Ξ:7,IC+IT3:6 OCT)$ S := 0$                     .NUC1   231
232  042257              REPEAT IT := BUF$  APPEND(IT,S+6,M)$  OP := -IT$         .NUC1   232    +R
233  042266                 C := OP MOD 8$                                        .NUC1   233
234  042270                 IT := BUF$  APPEND(IT,S+9,IT4)$  I := -IT$            .NUC1   234
235  042276                 IT := BUF$  APPEND(IT,S+12,IT4)$  J := -IT$           .NUC1   235
236  042305                 IT := BUF$                                            .NUC1   236
237  042306                 IF (OP < 8) ∨ ((OP ≥ 40) ^ (C ≤ 2)) THEN              .NUC1   237
238  042315                 BEGIN S := S+30$  APPEND(IT,S,MASK)  END ELSE         .NUC1   238    +3-3
239  042322                 BEGIN S := S+15$  APPEND(IT,S,IT4)  END$              .NUC1   239    +3-3
240  042326                 K := -IT$                                             .NUC1   240
241  042330                                                                       .NUC1   241
242  042330           CC := OP DIV 8$                                             .NUC1   242
243  042332           II :=CHR(I+INT(Ξ0Ξ))$ JJ := CHR(J+INT(Ξ0Ξ))$ KK :=CHR(K+INT(Ξ0Ξ))$.NUC1   243
244  042340             WRITE(EOL,Ξ Ξ:14)$                                        .NUC1   244
245  042344           IF CC < 5 THEN                                              .NUC1   245
```

```
246  042346          CASE CC OF
247  042353  0:        IF OP = 3 THEN
248  042355            BEGIN OUTCH(JX1[I]); OUTCH(JX2[I]); OUTCH(Ξ Ξ);
249  042371               SPACE; OUTCH(ΞXΞ); OUTCH(JJ); OUTCH(Ξ+Ξ);
250  042402              WRITE(K:6 OCT);
251  042404            END ELSE
252  042405            BEGIN OUTCH(JB1[OP]); OUTCH(JB2[OP]); OUTCH(Ξ Ξ);
253  042421               SPACE;
254  042423               IF OP > 3 THEN
255  042425            BEGIN OUTCH(ΞBΞ); OUTCH(II); OUTCH(RL[OP]);
256  042437                 OUTCH(ΞAΞ); OUTCH(JJ); OUTCH(Ξ+Ξ); WRITE(K:6 OCT);
257  042450               END ELSE
258  042451               CASE OP OF
259  042456  0:              ;
260  042457  1:              WRITE(K:6 OCT);
261  042462  2:              BEGIN OUTCH(ΞBΞ); OUTCH(II); OUTCH(Ξ+Ξ);
262  042471                 WRITE(K:6 OCT);
263  042473                 END;
264  042474               END;
265  042477            END  * 0 + ;
266  042500  1:        BEGIN OUTCH(ΞBΞ); OUTCH(ΞXΞ); OUTCH(II); SPACE;
267  042511               IF OP > 11 THEN
268  042513               BEGIN OUTCH(Ξ¬Ξ); II := JJ; JJ := KK; KK := II  END;
269  042522               OUTCH(ΞXΞ); OUTCH(JJ);
270  042527               C := OP MOD 4;
271  042531               IF C ≠ 0 THEN
272  042533            BEGIN OUTCH(RL[C]); OUTCH(ΞXΞ); OUTCH(KK) END;
273  042545            END  * 1 + ;
274  042546  2:        BEGIN OUTCH(SH[OP]); OUTCH(ΞXΞ); OUTCH(II); SPACE;
275  042563               IF OP < 18 THEN
276  042565               BEGIN OUTCH(JJ); OUTCH(KK)  END ELSE
277  042574               BEGIN OUTCH(ΞBΞ); OUTCH(JJ); OUTCH(Ξ+Ξ);
278  042603                  OUTCH(ΞXΞ); OUTCH(KK);
279  042610               END;
280  042610            END  * 2 + ;
281  042611  3:        BEGIN OUTCH(SH[OP DIV 2]); OUTCH(ΞXΞ); OUTCH(II);
282  042625               SPACE; OUTCH(ΞXΞ); OUTCH(JJ);
283  042634             IF ODD(OP) THEN OUTCH(Ξ-Ξ) ELSE OUTCH(Ξ+Ξ);
284  042643               OUTCH(ΞXΞ); OUTCH(KK);
285  042650            END  * 3 + ;
286  042651  4:        BEGIN C := OP MOD 4;
287  042653               IF OP = 38 THEN
288  042655               BEGIN OUTCH(ΞNΞ); OUTCH(ΞOΞ)  END ELSE
289  042662               IF C = 3 THEN
290  042664               BEGIN IF OP = 35 THEN OUTCH(ΞMΞ) ELSE OUTCH(ΞCΞ);
291  042673                  OUTCH(ΞXΞ); OUTCH(II); SPACE;
292  042702                  IF OP = 35 THEN OUTCH(JJ) ELSE OUTCH(ΞXΞ) ;
293  042712                  OUTCH(KK);
294  042715               END ELSE
295  042716               BEGIN OUTCH(FS[C]); OUTCH(ΞXΞ); OUTCH(II);
296  042730                  SPACE; OUTCH(ΞXΞ); OUTCH(JJ);
297  042737                  IF OP < 36 THEN OUTCH(Ξ*Ξ) ELSE OUTCH(Ξ/Ξ);
298  042746                  OUTCH(ΞXΞ); OUTCH(KK);
299  042753               END;
300  042753            END  * 4 + ;
301  042754         END  * CASE + ELSE
302  042762         BEGIN
303  042762            OUTCH(ΞSΞ); OUTCH(FS[CC]); OUTCH(II);
304  042774            SPACE; OUTCH(R1[C]); OUTCH(JJ);
305  043006            IF C IN [5,7] THEN OUTCH(Ξ-Ξ) ELSE OUTCH(Ξ+Ξ) ;
306  043016               IF C ≤ 2 THEN WRITE(K: 6 OCT) ELSE
307  043023            BEGIN OUTCH(ΞBΞ); OUTCH(KK)  END;
```

```
.NUC1   246   +C
.NUC1   247
.NUC1   248   +3
.NUC1   249
.NUC1   250
.NUC1   251   -3
.NUC1   252   +3
.NUC1   253
.NUC1   254
.NUC1   255   +4
.NUC1   256
.NUC1   257   -4
.NUC1   258   +C
.NUC1   259
.NUC1   260
.NUC1   261   +4
.NUC1   262
.NUC1   263   -4
.NUC1   264   -C
.NUC1   265   -3
.NUC1   266   +3
.NUC1   267
.NUC1   268   +4-4
.NUC1   269
.NUC1   270
.NUC1   271
.NUC1   272   +4-4
.NUC1   273   -3
.NUC1   274   +3
.NUC1   275
.NUC1   276   +4-4
.NUC1   277   +4
.NUC1   278
.NUC1   279   -4
.NUC1   280   -3
.NUC1   281   +3
.NUC1   282
.NUC1   283
.NUC1   284
.NUC1   285   -3
.NUC1   286   +3
.NUC1   287
.NUC1   288   +4-4
.NUC1   289
.NUC1   290   +4
.NUC1   291
.NUC1   292
.NUC1   293
.NUC1   294   -4
.NUC1   295   +4
.NUC1   296
.NUC1   297
.NUC1   298
.NUC1   299   -4
.NUC1   300   -3
.NUC1   301   -C
.NUC1   302   +3
.NUC1   303
.NUC1   304
.NUC1   305
.NUC1   306
.NUC1   307   +4-4
```

```
308  043030        END;                                                                .NUC1      308    -3
309  043030                                                                            .NUC1      309
310  043030          UNTIL S ≥ 60;                                                     .NUC1      310    -R
311  043032        END;                                                                .NUC1      311    -2
312  043034        IC := IC + CA + 1;  OUTCH(EOL);                                     .NUC1      312
313  043041    END  * PRTCOMP * ;                                                      .NUC1      313    -1
314  043044                                                                            .NUC1      314
315  043044    PROCEDURE PRJMPTAB ;                                                    .NUC1      315
316  043044     * PRINTS OUT JUMPTAB.                                                  .NUC1      316
317  043044      LOADPOINT OF JMPTAB IN MEM[458] *                                     .NUC1      317
318  043044       VAR I : SHRTINT;                                                     .NUC1      318
319  043044       BEGIN IF JMPIX # 0 THEN                                              .NUC1      319    +1
320  043051         BEGIN WRITE(EOL,'0',EOL,' JUMPTABLE'); WRITE(EOL,EOL);             .NUC1      320    +2
321  043065       FOR I := 0 TO JMPIX - 1 DO                                           .NUC1      321
322  043072       WRITE(' ':2,MEM[458]+I:6 OCT,JMPTAB[I]:21 OCT,EOL);                  .NUC1      322
323  043110         END                                                                .NUC1      323    -2
324  043110    END ;                                                                   .NUC1      324    -1
325  043113                                                                            .NUC1      325
326  043113    PROCEDURE WRITOUT ;                                                     .NUC1      326
327  043113     *CALLED AT PROCEDURE END, UPDATING ITS CODE (INSERTING ADDRESSES       .NUC1      327
328  043113      OF CONSTANTS) AND WRITING OUT CODE TOGETHER WITH THE CONSTANTS        .NUC1      328
329  043113      USED IN THIS PROCEDURE*                                               .NUC1      329
330  043113       VAR IT2 : SHRTINT;                                                   .NUC1      330
331  043113    BEGIN                                                                   .NUC1      331    +1
332  043116       NOOP ; CODE[CA] := BUF ;                                             .NUC1      332
333  043123       FOR IT := 1 TO CA DO                                                 .NUC1      333
334  043127       BEGIN PASCLGO* := CODE[IT] ; PUT(PASCLGO) END ;                      .NUC1      334    +2-2
335  043137         IF 'M' IN OPFLAG THEN PRTCOMP; CA:=0; CP:=4;                       .NUC1      335
336  043144    END  * WRITOUT * ;                                                      .NUC1      336    -1
337  043146 PROCEDURE STATCODE; FORWARD;                                               .NUC1      337
338  043146 *FOR THE EXPRESSION CODE GENERATION ROUTINE, ADDCODE, MULCODE,             .NUC1      338
339  043146    RELCODE, BOOLCODE, ARRCODE, TRANSCODE, EXCODE, THE ARGUMENTS           .NUC1      339
340  043146    OF EACH OPERATOR ARE PUSHED ON THE SYSTEM STACK, I.E.,                  .NUC1      340
341  043146    INDEX REGISTER B6 IS INCREMENTED (B6 IS STACK POINTER)*                 .NUC1      341
342  043146 PROCEDURE ADDCODE(ASON: IPTR);                                             .NUC1      342
343  043146   *GENERATE OBJECT CODE FOR ADDOPS.  POP OPERANDS OFF STACK.               .NUC1      343
344  043146    PUT LEFTSIDE IN X1, RIGHTSIDE IN X2, RESULT IN X7.                      .NUC1      344
345  043146    PUSH STACK WITH RESULT*                                                 .NUC1      345
346  043146      VAR OPN: SHRTINT;                                                     .NUC1      346
347  043146      BEGIN GEN15(56B,1,6,0);                                               .NUC1      347    +1
348  043156         IF ASON*.RSON=NIL THEN BEGIN                                       .NUC1      348    +2
349  043161 *UNARY*     IF ASON*.FATHER=MINUS THEN BEGIN                              .NUC1      349    +3
350  043164         GEN15(14B,7,1,1); GEN15(56B,7,6,0); NOOP END END                   .NUC1      350    -3-2
351  043177         ELSE BEGIN                                                         .NUC1      351    +2
352  043200 *BINARY*    IF ASON*.FATHER=PLUS THEN OPN:=36B ELSE OPN:=37B;              .NUC1      352
353  043205         GEN15(67B,6,6,1);                                                  .NUC1      353
354  043212         GEN15(56B,2,6,0);                                                  .NUC1      354
355  043217            GEN15(OPN,7,1,2);                                               .NUC1      355
356  043225         GEN15(56B,7,6,0);                                                  .NUC1      356
357  043232            NOOP;                                                           .NUC1      357
358  043233         END;                                                               .NUC1      358    -2
359  043233      END;                                                                  .NUC1      359    -1
360  043235 PROCEDURE MULCODE(MSON: IPTR);                                             .NUC1      360
361  043235   *GENERATE OBJECT CODE FOR MULOPS.*                                       .NUC1      361
362  043235      VAR OPN: SHRTINT;                                                     .NUC1      362
363  043235      BEGIN GEN15(56B,1,6,0);                                               .NUC1      363    +1
364  043245         GEN15(67B,6,6,1);                                                  .NUC1      364
365  043252         GEN15(56B,2,6,0);                                                  .NUC1      365
366  043257         IF MSON*.FATHER=TIMES THEN BEGIN                                   .NUC1      366    +2
367  043262            GEN15(42B,1,1,2); GEN15(13B,0,0,0);                             .NUC1      367
368  043274            GEN15(36B,7,1,0); GEN15(56B,7,6,0); NOOP; END                   .NUC1      368    -2
369  043307         ELSE IF MSON*.FATHER=SLASH THEN BEGIN                              .NUC1      369    +2
```

```
370  043313            GEN15(27B,1,0,1); GEN15(24B,1,0,1);                    .NUC1   370
371  043325            GEN15(27B,2,0,2); GEN15(44B,1,2,1);                    .NUC1   371
372  043337            GEN15(26B,1,7,1); GEN15(22B,1,7,1);                    .NUC1   372
373  043351            GEN15(13B,0,0,0); GEN15(36B,7,1,0);                    .NUC1   373
374  043363            GEN15(56B,7,6,0); NOOP;                                .NUC1   374
375  043371          END ELSE BEGIN                                           .NUC1   375   -2+2
376  043372          NOOP;                                                    .NUC1   376
377  043373 BMOD+       OPN:=IC;                                              .NUC1   377
378  043375            GEN15(37B,1,1,2);                                      .NUC1   378
379  043402            GEN15(37B,3,1,2);                                      .NUC1   379
380  043407            GEN30(3B,2,3,OPN);                                     .NUC1   380
381  043415            GEN15(22B,7,0,1);                                      .NUC1   381
382  043422          GEN15(56B,7,6,0);                                        .NUC1   382
383  043427            NOOP;                                                  .NUC1   383
384  043430          END                                                      .NUC1   384   -2
385  043430        END;                                                       .NUC1   385   -1
386  043432 PROCEDURE RELCODE(RLSON: TPTR);                                  .NUC1   386
387  043432   BGENERATE CODE FOR RELATIONAL OPERATORS+                       .NUC1   387
388  043432        VAR OPN,I,J: SHRTINT;                                      .NUC1   388
389  043432        BEGIN GEN15(56B,1,6,0);                                    .NUC1   389   +1
390  043442           GEN15(67B,6,6,1);                                       .NUC1   390
391  043447           GEN15(56B,2,6,0);                                       .NUC1   391
392  043454           GEN15(37B,3,1,2);                                       .NUC1   392
393  043461           IF RLSON+.FATHER=EQ THEN OPN:=0 ELSE                    .NUC1   393
394  043465           IF RLSON+.FATHER=NE THEN OPN:=1 ELSE                    .NUC1   394
395  043471           BEGIN IF (RLSON+.FATHER=LE) V (RLSON+.FATHER=GE) THEN   .NUC1   395   +2
396  043500              BEGIN NOOP; GEN30(03B,0,3,IC+2) END ELSE             .NUC1   396   +3-3
397  043510           BEGIN NOOP; GEN30(03B,0,3,IC+1) END;                    .NUC1   397   +3-3
398  043517           IF (RLSON+.FATHER=LT) V (RLSON+.FATHER=LE) THEN         .NUC1   398
399  043526              OPN:=2 ELSE OPN:=3 END;                              .NUC1   399   -2
400  043531           IF (RLSON+.FATHER#EQ) ^ (RLSON+.FATHER#NE) THEN         .NUC1   400
401  043540              I:=1 ELSE I:=2;                                      .NUC1   401
402  043543           GEN30(03B,OPN,3,IC+I); GEN30(71B,7,0,0);                .NUC1   402
403  043557           GEN30(04B,0,0,IC+I); NOOP;                              .NUC1   403
404  043566           GEN30(71B,7,0,1); NOOP;                                 .NUC1   404
405  043574           GEN15(56B,7,6,0);                                       .NUC1   405
406  043601        END;                                                       .NUC1   406   -1
407  043603 PROCEDURE BOOLCODE(BSON: TPTR);                                  .NUC1   407
408  043603   BGENERATE CODE FOR ¬, ^, V+                                    .NUC1   408
409  043603        VAR OPN: SHRTINT;                                          .NUC1   409
410  043603        BEGIN GEN15(56B,1,6,0);                                    .NUC1   410   +1
411  043613 BNOT+     IF BSON+.FATHER=NOT THEN GEN15(14B,7,1,1) ELSE          .NUC1   411
412  043624 BOR,AND+  BEGIN IF BSON+.FATHER=AND THEN OPN:=11B ELSE OPN:=12B;  .NUC1   412   +2
413  043631           GEN15(67B,6,6,1); GEN15(56B,2,6,0);                     .NUC1   413
414  043643              GEN15(OPN,7,1,2);                                    .NUC1   414
415  043651              END;                                                 .NUC1   415   -2
416  043651           GEN15(56B,7,6,0);                                       .NUC1   416
417  043656           NOOP;                                                   .NUC1   417
418  043657        END;                                                       .NUC1   418   -1
419  043661 PROCEDURE ARRCODE(ARSON: TPTR);                                  .NUC1   419
420  043661   BGENERATE CODE FOR ARRAY REFERENCES+                           .NUC1   420
421  043661        VAR K: SHRTINT;                                           .NUC1   421
422  043661        BEGIN GEN15(56B,1,6,0); GEN15(67B,6,6,1);                 .NUC1   422   +1
423  043676           GEN15(56B,2,6,0); GEN15(37B,1,1,2);                    .NUC1   423
424  043710           GEN30(71B,7,0,IC);                                     .NUC1   424
425  043716           GEN30(03B,3,1,171B); GEN15(63B,7,2,0);                 .NUC1   425
426  043730           K:=STSYM+ARSON+.INX-INTI(ARSON+.RSON+.FATHER);         .NUC1   426
427  043741           GEN30(51B,1,7,K);                                      .NUC1   427
428  043747           GEN15(22B,7,0,1); GEN15(56B,7,6,0);                    .NUC1   428
429  043761           NOOP;                                                  .NUC1   429
430  043762        END;                                                       .NUC1   430   -1
431  043764 PROCEDURE TRANSCODE(TSON: TPTR);                                 .NUC1   431
```

```
432  043764    *GENERATE CODE FOR TYPE TRANSFER FUNCTIONS*                          .NUC1   432
433  043764      VAR FLAG: SHRTINT;                                                 .NUC1   433
434  043764      BEGIN WITH TSON* DO BEGIN                                          .NUC1   434    *1*2
435  043771         IF FATHER=ΞINTOFCHARΞ THEN FLAG:=1 ELSE                         .NUC1   435
436  043774         IF FATHER=ΞINTOFBOOLΞ THEN FLAG:=2 ELSE                         .NUC1   436
437  044000         IF FATHER=ΞBOOLOFINTΞ THEN FLAG:=3 ELSE                         .NUC1   437
438  044004         IF FATHER=ΞBOOLOFCHARΞ THEN FLAG:=4 ELSE                        .NUC1   438
439  044010         IF FATHER=ΞCHAROFINTΞ THEN FLAG:=5 ELSE FLAG:=6;                .NUC1   439
440  044015         IF FLAG IN [3,4,5] THEN BEGIN                                   .NUC1   440    *3
441  044020         IF FLAG IN [3,4] THEN FLAG:=1 ELSE FLAG:=77B;                   .NUC1   441
442  044025         NOOP; GEN15(56B,1,6,0); GEN30(71B,2,0,FLAG);                    .NUC1   442
443  044041         GEN15(11B,7,1,2); GEN15(56B,7,6,0);                             .NUC1   443
444  044053         END;                                                            .NUC1   444    -3
445  044053         END;                                                            .NUC1   445    -2
446  044053      END;                                                               .NUC1   446    -1
447  044062 PROCEDURE EXCODE(SON: TPTR);                                           .NUC1   447
448  044062   *GENERATE CODE FOR EXPRESSIONS. CALL ADDCODE,MULCODE, RELCODE,       .NUC1   448
449  044062    BOOLCODE,TRANSCODE,OR ARRCODE WHEN NEEDED.  ACTUAL CODE             .NUC1   449
450  044062    GENERATION IN THIS PROCEDURE IS FOR PRIMARIES*                       .NUC1   450
451  044062      VAR I,J: INTEGER;                                                  .NUC1   451
452  044062      PROCEDURE PRMCODE;                                                 .NUC1   452
453  044062         *CODE FOR PRIMARIES*                                            .NUC1   453
454  044062         VAR K: SHRTINT;                                                 .NUC1   454
455  044062         BEGIN IF J=1 THEN BEGIN GEN30(51B,1,0,STSYM*I);                 .NUC1   455    *1*2
456  044076            GEN15(10B,7,1,1); END                                        .NUC1   456    -2
457  044103            ELSE IF J=2 THEN GEN30(71B,7,0,INTI(SON*.FATHER))            .NUC1   457
458  044117            ELSE IF J=3 THEN BEGIN IF SON*.FATHER=RESWRDS[11] THEN       .NUC1   458    *2
459  044127            I:=0                                                         .NUC1   459
460  044127               ELSE I:=1;                                                .NUC1   460
461  044132               GEN30(71B,7,0,I); END                                     .NUC1   461    -2
462  044140            ELSE BEGIN I:=-INT(SON*.FATHER);                             .NUC1   462    *2
463  044144            APPEND(I,12,-77B);                                           .NUC1   463
464  044147            GEN30(71B,1,0,-I);                                           .NUC1   464
465  044155            GEN30(71B,2,0,77B); GEN15(11B,7,1,2);                        .NUC1   465
466  044167         END;                                                            .NUC1   466    -2
467  044167            GEN15(66B,6,6,1); GEN15(56B,7,6,0);                          .NUC1   467
468  044201            NOOP;                                                        .NUC1   468
469  044202         END;                                                            .NUC1   469    -1
470  044204      BEGIN IF SON≠NIL THEN BEGIN                                        .NUC1   470    *1*2
471  044211         WITH SON* DO                                                    .NUC1   471
472  044213         IF LSON≠NIL THEN BEGIN EXCODE(LSON); EXCODE(RSON);              .NUC1   472    *3
473  044225         IF (FATHER=PLUS) ∨ (FATHER=MINUS) THEN ADDCODE(SON) ELSE        .NUC1   473
474  044240         IF (FATHER=TIMES) ∨ (FATHER=SLASH) ∨ (FATHER=EXPO) THEN         .NUC1   474
475  044252            MULCODE(SON) ELSE                                            .NUC1   475
476  044256         IF (FATHER=LT) ∨ (FATHER=LE) ∨ (FATHER=EQ)                      .NUC1   476
477  044266            ∨ (FATHER=NE) ∨ (FATHER=GE) ∨ (FATHER=GT) THEN               .NUC1   477
478  044301            RELCODE(SON) ELSE                                            .NUC1   478
479  044305         IF (FATHER=ΞINTOFBOOLΞ) ∨ (FATHER=ΞINTOFCHARΞ)                  .NUC1   479
480  044312            ∨ (FATHER=ΞBOOLOFINTΞ) ∨ (FATHER=ΞBOOLOFCHARΞ)               .NUC1   480
481  044320            ∨ (FATHER=ΞCHAROFINTΞ) ∨ (FATHER=ΞCHAROFBOOLΞ) THEN          .NUC1   481
482  044330            TRANSCODE(SON) ELSE                                          .NUC1   482
483  044334            IF (FATHER=AND) ∨ (FATHER=OR) ∨ (FATHER=NOT) THEN            .NUC1   483
484  044346            BOOLCODE(SON) ELSE                                           .NUC1   484
485  044352         ARRCODE(SON);                                                   .NUC1   485
486  044355         END ELSE BEGIN                                                  .NUC1   486    -3*3
487  044356            I:=INX; J:=TY;                                               .NUC1   487
488  044363            PRMCODE;                                                     .NUC1   488
489  044364         END END                                                         .NUC1   489    -3-2
490  044364      END;                                                               .NUC1   490    -1
491  044374 *THE FOLLOWING PROCEDURES ARE THE CODE GENERATION ROUTINES FOR          .NUC1   491
492  044374    THE STATEMENTS*                                                     .NUC1   492
493  044374 PROCEDURE ASCODE;                                                      .NUC1   493
```

```
494  044374    *GENERATE CODE FOR ASSIGNMENT STATEMENT*
495  044374      VAR K: SHRTINT;
496  044374      BEGIN WITH VPROG↑ DO BEGIN
497  044401         EXCODE(RSON);
498  044404         GEN15(66B,7,0,0);
499  044411         K:=0;
500  044412         IF LSON↑.LSON≠NIL THEN BEGIN
501  044416            EXCODE(LSON↑.LSON); EXCODE(LSON↑.RSON);
502  044430            GEN15(56B,1,6,0);
503  044435               GEN15(67B,6,6,1); GEN15(56B,2,6,0);
504  044447            GEN15(67B,6,6,1);
505  044454            GEN30(71B,7,0,IC);
506  044462            GEN15(37B,1,1,2); GEN30(03B,3,1,171B);
507  044474            GEN15(63B,7,2,0);
508  044501            K:=-INTI(LSON↑.RSON↑.FATHER);
509  044510         END;
510  044510            K:=STSYM+LSON↑.INX+K; GEN30(61B,7,7,K);
511  044522            GEN15(56B,1,6,0);
512  044527            GEN15(67B,6,6,1);
513  044534            GEN15(22B,7,0,1); GEN15(56B,7,7,0);
514  044546         END;
515  044546      END;
516  044550 PROCEDURE ENTCODE;
517  044550    *GENERATE CODE FOR ENTER STATEMENT*
518  044550      VAR K: SHRTINT; J: ALFA;
519  044550      BEGIN WITH VPROG↑ DO
520  044555         BEGIN K:=S;
521  044556         WHILE SYMTAB[K].STRNG≠LSON↑.FATHER DO K:=K+1;
522  044567         K:=K+STPROC-S; END;
523  044572         NOOP;
524  044573         GEN30(71B,7,0,IC+2);
525  044601         GEN15(66B,6,6,1); GEN15(56B,7,6,0);
526  044613         GEN30(02B,0,0,K); NOOP;
527  044621      END;
528  044623 PROCEDURE EXTCODE;
529  044623    *GENERATE CODE FOR EXIT STATEMENT*
530  044623      CONST JMP=040000000046000046000B; SHFT=100000000000B;
531  044623      VAR J: ALFA; I: INTEGER;
532  044623      BEGIN GEN15(56B,1,6,0); GEN15(67B,6,6,1);
533  044640         GEN15(63B,7,1,0); GEN30(02B,7,0,0);
534  044652         NOOP;
535  044653         *RITOUT;
536  044654         JMPTAB[JMPIX]:=JMP; INSERT(IC,30,JMPTAB[JMPIX]);
537  044665         JMPIX:=JMPIX+1;
538  044666          PROCNAME.BOUND:=PROCNAME.BOUND+D-1;
539  044674         D:=1; R:=1;
540  044676      END;
541  044701 PROCEDURE HLTCODE;
542  044701    *GENERATE CODE FOR HALT STATEMENT*
543  044701      BEGIN GEN30(51B,1,0,458);
544  044711         GEN15(63B,7,1,1); GEN30(02B,7,0,0);
545  044723      END;
546  044725 PROCEDURE IOCODE;
547  044725    *GENERATE CODE FOR I/O STATEMENTS, I.E., GENERATE CODE
548  044725     FOR JUMP TO SYSTEM I/O ROUTINES.  X1 HAS ADDRESS OF ARRAY.
549  044725     X2 HAS LENGTH OF ARRAY.  X7 HAS RETURN ADDRESS*
550  044725      VAR K,OPN: SHRTINT;
551  044725      BEGIN NOOP;
552  044731         IF VPROG↑.FATHER=RESWRDS[20] THEN
553  044734         OPN:=READCH ELSE OPN:=WRITECH;
554  044740         K:=INTI(VPROG↑.RSON↑.FATHER)+1;
555  044746         GEN30(71B,1,0,VPROG↑.LSON↑.INX+STSYM-K+1);

.NUC1   494
.NUC1   495
.NUC1   496    +1+2
.NUC1   497
.NUC1   498
.NUC1   499
.NUC1   500    +3
.NUC1   501
.NUC1   502
.NUC1   503
.NUC1   504
.NUC1   505
.NUC1   506
.NUC1   507
.NUC1   508
.NUC1   509    -3
.NUC1   510
.NUC1   511
.NUC1   512
.NUC1   513
.NUC1   514    -2
.NUC1   515    -1
.NUC1   516
.NUC1   517
.NUC1   518
.NUC1   519    +1
.NUC1   520    +2
.NUC1   521
.NUC1   522    -2
.NUC1   523
.NUC1   524
.NUC1   525
.NUC1   526
.NUC1   527    -1
.NUC1   528
.NUC1   529
.NUC1   530
.NUC1   531
.NUC1   532    +1
.NUC1   533
.NUC1   534
.NUC1   535
.NUC1   536
.NUC1   537
.NUC1   538
.NUC1   539
.NUC1   540    -1
.NUC1   541
.NUC1   542
.NUC1   543    +1
.NUC1   544
.NUC1   545    -1
.NUC1   546
.NUC1   547
.NUC1   548
.NUC1   549
.NUC1   550
.NUC1   551    +1
.NUC1   552
.NUC1   553
.NUC1   554
.NUC1   555
```

```
556  044760          GEN30(718,2,0,K);                                                    .NUC1   556
557  044766          GEN30(718,7,0,IC+1);                                                 .NUC1   557
558  044774          GEN30(048,0,0,OPN);                                                  .NUC1   558
559  045001       END;                                                                    .NUC1   559     -1
560  045003 FUNCTION MEMLBL(LBL: ALFA): BOOLEAN;                                         .NUC1   560
561  045003       VAR I: INTEGER;                                                         .NUC1   561
562  045003       BEGIN MEMLBL:=FALSE;                                                    .NUC1   562     +1
563  045007          IF R>1 THEN BEGIN I:=1;                                              .NUC1   563     +2
564  045012          REPEAT IF LBL=REFLABSET[I].STRNG THEN                                .NUC1   564     +R
565  045020             BEGIN MEMLBL:=TRUE; LTEMP:=I; I:=R END;                           .NUC1   565     +3-3
566  045024             I:=I+1                                                            .NUC1   566
567  045025          UNTIL I>R;                                                           .NUC1   567     -R
568  045030          END;                                                                 .NUC1   568     -2
569  045030       END;                                                                    .NUC1   569     -1
570  045032 PROCEDURE JMPCODE;                                                           .NUC1   570
571  045032    (*GENERATE CODE FOR JUMPTO STATEMENTS.  SET UP FORWARD                    .NUC1   571
572  045032     REFERENCE CHAIN IF NECESSARY*)                                           .NUC1   572
573  045032     VAR I,L: INTEGER; J: ALFA; K: SHRTINT; TF: BOOLEAN;                      .NUC1   573
574  045032     BEGIN WITH VPROG^ DO BEGIN                                              .NUC1   574     +1+2
575  045037        I:=INTI(LSON^.FATHER)-PROCNAME.BOUND;                                .NUC1   575
576  045045        NOOP;                                                                .NUC1   576
577  045046        IF I<D-1 THEN K:=DEFLABSET[I+1].BOUND ELSE                           .NUC1   577
578  045056        BEGIN J:=LSON^.FATHER;                                               .NUC1   578     +3
579  045061        IF ~MEMLBL(J) THEN BEGIN REFLABSET[R].STRNG:=J;                      .NUC1   579     +4
580  045072        REFLABSET[R].BOUND:=IC; K:=0; R:=R+1; END                            .NUC1   580     -4
581  045103        ELSE BEGIN K:=REFLABSET[LTEMP].BOUND; GEN15(668,7,0,0); END;         .NUC1   581     +4-4
582  045116        END;                                                                 .NUC1   582     -3
583  045116        GEN30(04H,0,0,K);                                                    .NUC1   583
584  045123        NOOP;                                                                .NUC1   584
585  045124        END;                                                                 .NUC1   585     -2
586  045124     END;                                                                    .NUC1   586     -1
587  045126 PROCEDURE CSLBL;                                                             .NUC1   587
588  045126    (*GENERATE CODE FOR CASE LABEL TABLE.  JUMP INSTRUCTION                   .NUC1   588
589  045126     IN UPPER 30 BITS. CASE LABEL VALUE IN LOWER 30 BITS.*)                   .NUC1   589
590  045126     CONST JMP=040000000000000000B;                                           .NUC1   590
591  045126     VAR I,K: INTEGER; J: ALFA;                                               .NUC1   591
592  045126     BEGIN NOOP; CODE[CA]:=BUF;                                              .NUC1   592     +1
593  045136        CA:=CA+1; IC:=IC+1; IF CA>CODMAX THEN ERROR(35);                     .NUC1   593
594  045145        J:=VPROG^.FATHER; I:=INTI(J);                                        .NUC1   594
595  045153        BUF:=JMP; INSERT(I,0,BUF);                                           .NUC1   595
596  045156        K:=PROCNAME.BOUND;                                                   .NUC1   596
597  045160        J:=VPROG^.LSON^.FATHER; I:=INTI(J);                                  .NUC1   597
598  045167        I:=DEFLABSET[I-K].BOUND;                                             .NUC1   598
599  045175        INSERT(I,30,BUF);                                                    .NUC1   599
600  045177     END;                                                                    .NUC1   600     -1
601  045202 PROCEDURE IFCODE;                                                            .NUC1   601
602  045202    (*GENERATE CODE FOR IF STATEMENTS*)                                       .NUC1   602
603  045202     CONST EQJMP=047000000046000460008;                                       .NUC1   603
604  045202     VAR J: ALFA;                                                             .NUC1   604
605  045202     BEGIN WITH VPROG^ DO BEGIN                                              .NUC1   605     +1+2
606  045207        EXCODE(LSON);                                                        .NUC1   606
607  045212        GEN15(568,1,6,0); GEN15(678,6,6,1);                                  .NUC1   607
608  045224        GEN15(638,7,1,0); NOOP;                                              .NUC1   608
609  045232        CODE[CA]:=BUF;                                                       .NUC1   609
610  045235        CA:=CA+1;                                                            .NUC1   610
611  045240        IC:=IC+1;                                                            .NUC1   611
612  045241        CP:=-1;                                                              .NUC1   612
613  045242        IF CA>CODMAX THEN ERROR(35);                                         .NUC1   613
614  045246        BUF:=EQJMP; J:=RSON^.LSON^.FATHER;                                   .NUC1   614
615  045253        IF MEMLBL(J) THEN INSERT(REFLABSET[LTEMP].BOUND,30,BUF)              .NUC1   615
616  045265        ELSE BEGIN REFLABSET[R].STRNG:=J;                                    .NUC1   616     +3
617  045274        REFLABSET[R].BOUND:=IC-1; R:=R+1; END;                               .NUC1   617     -3
```

```
618  045304        END;                                                                  .NUC1   618   -2
619  045304      END;                                                                    .NUC1   619   -1
620  045307 PROCEDURE CSCODE;                                                            .NUC1   620
621  045307    *GENERATE CODE FOR CASE STATEMENTS.  SET UP JUMP TO SYSTEM                .NUC1   621
622  045307     BINARY SEARCH ROUTINE.  X1 HAS CASE EXPRESSION VALUE.                    .NUC1   622
623  045307     B2 HAS ADDRESS OF FAIL POINT OF CASE STATEMENT.                          .NUC1   623
624  045307     B7 HAS ADDRESS OF BEGINNING OF CASE LABEL TABLE+                         .NUC1   624
625  045307      CONST SH2B7=61200000006172000000B;                                      .NUC1   625
626  045307      VAR I: SHRTINT; J: ALFA;                                               .NUC1   626
627  045307      BEGIN WITH VPROG* DO BEGIN                                             .NUC1   627   +1+2
628  045314         EXCODE(LSON);                                                       .NUC1   628
629  045317         GEN15(56B,1,6,0); GEN15(67B,6,6,1);                                 .NUC1   629
630  045331         NOOP; CODE[CA]:=BUF;                                                .NUC1   630
631  045336         CA:=CA+1; IC:=IC+1;                                                 .NUC1   631
632  045341         CP:=4;                                                              .NUC1   632
633  045342         IF CA>CODMAX THEN ERROR(35);                                        .NUC1   633
634  045346         BUF:=SH2B7;                                                         .NUC1   634
635  045350         I:=INTI(RSON*.FATHER)-INTI(RSON*.LSON*.FATHER);                     .NUC1   635
636  045363         INSERT(I,0,BUF); J:=RSON*.FATHER;                                   .NUC1   636
637  045367         IF MEMLBL(J) THEN INSERT(REFLABSET[LTEMP].BOUND,30,BUF)             .NUC1   637
638  045402         ELSE BEGIN REFLABSET[R].STRNG:=J;                                   .NUC1   638   +3
639  045410         REFLABSET[R].BOUND:=IC-1; R:=R+1; END;                              .NUC1   639   -3
640  045420         GEN30(04B,0,0,BSRCH); NOOP;                                         .NUC1   640
641  045426         END;                                                                .NUC1   641   -2
642  045426      END;                                                                   .NUC1   642   -1
643  045431 PROCEDURE STATCODE;                                                          .NUC1   643
644  045431    *DETERMINE WHICH STATEMENT IS BEING RECOGNIZED AND                        .NUC1   644
645  045431     CALL APPROPRIATE CODE GENERATION ROUTINE+                                .NUC1   645
646  045431      VAR J:ALFA; I: INTEGER;                                                 .NUC1   646
647  045431      PROCEDURE BACKCHAIN;                                                    .NUC1   647
648  045431         VAR K: SHRTINT;                                                     .NUC1   648
649  045431         BEGIN J:=DEFLABSET[D].STRNG;                                        .NUC1   649   +1
650  045442            IF MEMLBL(J) THEN BEGIN K:=REFLABSET[LTEMP].BOUND;               .NUC1   650   +2
651  045454            I:=DEFLABSET[I].BOUND;                                           .NUC1   651
652  045456            INSERT(IC,30,CODE[K-I+1]);                                       .NUC1   652
653  045465            END;                                                             .NUC1   653   -2
654  045465         END;                                                                .NUC1   654   -1
655  045467 BEGIN I:=PROCNAME.BOUND+D-1;                                                .NUC1   655   +1
656  045476         DEFLABSET[D].STRNG:=ALFI(I); DEFLABSET[D].BOUND:=IC;                .NUC1   656
657  045517         BACKCHAIN; D:=D+1;                                                  .NUC1   657
658  045522         WITH VPROG* DO                                                      .NUC1   658
659  045523         IF FATHER=ASSIGNS THEN ASCODE ELSE                                  .NUC1   659
660  045530         IF FATHER=RESWRDS[3] THEN CSCODE ELSE                               .NUC1   660
661  045535         IF FATHER=RESWRDS[8] THEN ENTCODE ELSE                              .NUC1   661
662  045542         IF FATHER=RESWRDS[10] THEN EXTCODE ELSE                             .NUC1   662
663  045547         IF FATHER=RESWRDS[14] THEN HLTCODE ELSE                             .NUC1   663
664  045554         IF FATHER=RESWRDS[15] THEN IFCODE ELSE                              .NUC1   664
665  045561         IF FATHER=JUMPTO THEN JMPCODE ELSE                                  .NUC1   665
666  045566       IF (FATHER=RESWRDS[20]) v (FATHER=RESWRDS[27]) THEN                   .NUC1   666
667  045575         IOCODE ELSE CSLBL;                                                  .NUC1   667
668  045600      END;                                                                   .NUC1   668   -1
669  045604 PROCEDURE PASS2;                                                             .NUC1   669
670  045604      CONST JMPSIZE=100B; RLC=54; JMP=040000000046000460008;                 .NUC1   670
671  045604 BEGIN STSYM:=6001B;                                                          .NUC1   671   +1
672  045610      CA:=0; CP:=4;                                                          .NUC1   672
673  045612      JMPIX:=0;                                                              .NUC1   673
674  045613      STPROC:=STSYM+ARRAYSIZE+S-1;                                           .NUC1   674
675  045616      IC:=STPROC+JMPSIZE;                                                    .NUC1   675
676  045620      MEM[43B]:=IC; MEM[45B]:=STPROC;                                        .NUC1   676
677  045623      JMPTAB[JMPIX]:=JMP; INSERT(IC,30,JMPTAB[JMPIX]);                       .NUC1   677
678  045633      JMPIX:=JMPIX+1;                                                        .NUC1   678
679  045635      READCH:=STPROC+RLC; WRITECH:=READCH+1;                                 .NUC1   679
```

```
680  045640       BSRCH:=READCH*2; TCODE:=READCH*3;
681  045643       D:=1; R:=1;
682  045645       PROCNAME.BOUND:=0;
683  045650       RESET(VPROG);
684  045651       GET(VPROG);
685  045653       WHILE ¬EOF(VPROG) DO BEGIN STATCODE; GET(VPROG) END;
686  045660  END;  *PASS2*
687  045663  *MAIN PROGRAM.  WHEN FINISH CODE GENERATION, MUST SET
688  045663   APPROPRIATE WORDS IN LOW CORE TO INVOKE PASCAL OPERATING
689  045663   SYSTEM.
690  045663         LOCATION              CONTENTS
691  045663           41B                  0
692  045663           42B                  1
693  045663           43B                  FIRST IC OF OBJECT CODE
694  045663           44B                  LAST IC
695  045663           45B                  JUMPTABLE ADDRESS
696  045663           46B                  0
697  045663           47B                  ENTRY POINT*
698  045663  BEGIN
699  045672       FOR JMPIX:=0 TO JMPMAX DO JMPTAB[JMPIX]:=0;
700  045702       PASS2;
701  045703       MEM[47B]:=IC;
702  045705       GEN15(56B,7,5,0); GEN15(66B,6,5,1);
703  045717       GEN30(71B,7,0,IC+1);
704  045725       GEN15(56B,7,6,0);
705  045732       PC:=5;
706  045734       WHILE (SYMTAB[PC].STRNG#TOKENSTRNG.STRNG) ^ (PC<P)
707  045745          DO PC:=PC+1;
708  045751       GEN30(04H,0,0,STPROC+PC-5);
709  045760       DOOP;
710  045761       GEN15(56B,1,6,0); GEN15(63B,7,1,0); GEN30(02B,7,0,0);
711  046000       MEM[41B]:=0; MEM[42B]:=1;
712  046002       MEM[44B]:=IC; MEM[46B]:=0;
713  046005       WRITOUT;
714  046006       JMPTAB[66B]:=INT(ΞREADNΞ);
715  046010       JMPTAB[67B]:=INT(ΞWRITENΞ);
716  046011       JMPTAB[70B]:=INT(ΞBINSΞ);
717  046013       FOR PC:=0 TO JMPMAX DO BEGIN
718  046016          PASCLGO*:=JMPTAB[PC]; PUT(PASCLGO); END;
719  046026       RESET(PASCLGO);
720  046030       IF ΞME IN OPFLAG THEN PRJMPTAB;
721  046033       PC:=MEM[145B]; MEM[PC+13]:=0;
722  046041       MEM[PC+11]:=1;
723  046045       MEM[PC+21]:=1625030211160000000008;
724  046052  END.
```

```
.NUC1        680
.NUC1        681
.NUC1        682
.NUC1        683
.NUC1        684
.NUC1        685     +2-2
.NUC1        686     -1
.NUC1        687
.NUC1        688
.NUC1        689
.NUC1        690
.NUC1        691
.NUC1        692
.NUC1        693
.NUC1        694
.NUC1        695
.NUC1        696
.NUC1        697
.NUC1        698     +1
.NUC1        699
.NUC1        700
.NUC1        701
.NUC1        702
.NUC1        703
.NUC1        704
.NUC1        705
.NUC1        706
.NUC1        707
.NUC1        708
.NUC1        709
.NUC1        710
.NUC1        711
.NUC1        712
.NUC1        713
.NUC1        714
.NUC1        715
.NUC1        716
.NUC1        717     +2
.NUC1        718     -2
.NUC1        719
.NUC1        720
.NUC1        721
.NUC1        722
.NUC1        723
.NUC1        724     -1
```

APPENDIX B

CONTROL CARDS AND COMPILER OPTIONS

CONTROL CARDS

The following control cards must be used in order to execute a Nucleus program:

```
READPF,4806,NUCLEUS;NUC2,NUCLIB.
READPF,0997,YPASCAL.
YPASCAL,B=NUCLEUS.
YPASCAL,B=NUCBIN,LIB=NUCLIB.
```

The YPASCAL control cards may contain any of the parameters available to PASCAL users (see Wirth[12]).

COMPILER OPTIONS

Several compiler options are available to Nucleus users. In order to activate these options, the following comment card (or some form of it) must be used:

$↑↓SCPVM$

The letters may appear in any order, and only those letters which activate the desired options are placed in the above comment card. For example, $↑↓VM$ would activate only the V and M options. The following options are available.

Letter	Option
S	Dump the symbol table
C	Dump the constant table
P	Dump the procedure table
V	Dump the virtual program
M	Dump the COMPASS code

Options S, C, P, and V are handled by Pass I, and option M is handled by Pass II.

BIBLIOGRAPHY

1. Burger, W. F., Pascal Manual, University of Texas at Austin Computation Center, To be published.

2. Burstall, R. M., Formal Description of Program Structure and Semantics in First Order Logic, Machine Intelligence 5 (Meltzer and Michie, Ed.), American Elsevier Publishing Co., New York, 1970.

3. Control Data 6400/6500/6600 Computer Systems, COMPASS Reference Manual, Pub. No. 60190900.

4. Dijkstra, E. W., Notes on Structured Programming, EWD 249, Technical U. Eindhoven, The Netherlands, 1969.

5. Good, D. I., Developing Correct Software, In Proceedings of the First Texas Symposium on Computer Systems, 1972.

6. Good, D. I. and Ragland, L. C., Nucleus - A Language of Provable Programs, In Proceedings of an SIGPLAN Symposium on Computer Test Methods, Prentice-Hall, 1972.

7. Knuth, D. E., The Art of Computer Programming, Vol. 1, Addison-Wesley Publishing Company, Mass., 1969.

8. Ragland, L. C., A Verified Program Verifier (Working Title), Ph.D. Thesis, University of Texas at Austin, In preparation.

9. Wang, Y. Y., A Verification Condition Generator for Nucleus Programs (Working Title), Masters Thesis, University of Texas at Austin, In preparation.

10. Wirth, N., The Design of a Pascal Compiler, In International Summer School on Program Structures and Fundamental Concepts of Programming, 1971.

11. Wirth, N., Program Development by Stepwise Refinement, Comm. of ACM, 14, 4, April, 1971.

12. Wirth, N., The Programming Language Pascal, Acta Informatica, 1, 1971.

13. Woods, W. A., Transition Network Grammars for Natural Language Analysis, Comm. of ACM, 13, 10, October, 1970.