Introduction

OS/360 Algol is the IBM implementation of Algol 60\(^1\) for the System/360\(^2\). Extended Algol\(^3\) for the Burroughs B5500 contains many more capabilities than Algol 60 but is defined analogously. Therefore, the translator from B5500 Algol to OS/360 Algol is a mapping from a superset into a (approximately) proper subset. It is not a very hard translation task for the elements contained in the intersection of the two languages, and is very hard for some of the machine dependent extensions of B5500 Algol. The translator was written to be a simple translator dealing with translating the Algol 60 structures of B5500 Algol into OS/360 Algol.

The translation process handles simple translations which are equivalent to finite state transformations. This level of translation is the simplest possible. It would be harder to deal with Fortran to PL/1 which requires context free transformations, and harder still to go from B5500 Algol to PL/1 which would require dealing with problems of semantic equivalence of similar syntactic form. The translator described here for B5500 Algol to OS/360 Algol mostly does work akin to the following production, 

\[
\texttt{<reserved word>} ::= \texttt{'<quoted reserved word>'}. \]

A Fortran to PL/1\(^4\) translator deals in harder syntactic transformations where, for example, the Fortran \texttt{<do statement>} must be translated with an END inserted at the place of the statement number that ends the \texttt{<do statement>}.

Harder still\(^5\) is to appropriately translate a B5500 Algol \texttt{<for statement>} into a PL/1 \texttt{<do statement>} where the syntax does not differ greatly but the meaning does. The Algol statement constantly
re-evaluates its step size and upper bound, but the PL/1 statement does this only once. For reasonable translation here, it becomes necessary to translate the Algol <for statement> into the corresponding conditional and branching statements, or to check that the Algol statement being executed in the for loop is not affecting the step size or bound.

**Use**

The translator is simple to use, and its output requires little further modification. The translator takes as input a B5500 deck (extended IBM 026 key punch code) and produces an OS/360 Algol translation; this translation is produced as a sequenced deck (IBM 029 key punch code) and for input to the OS/360 Algol compiler. The translator deletes certain constructions which would cause the OS/360 Algol compiler undue syntactic hardship. The translator produces a copy of these deletions referencing the sequence numbers of the outputted deck, and the OS/360 Algol compiler produces its standard syntactic diagnostic messages.

Further translation by hand (see Appendix 1) is required in certain areas, principally input/output. If stream procedures and bit manipulation are extensively used the program should probably not be translated, but reprogrammed and not necessarily in Algol. However, if the B5500 Algol program mainly uses the Algol 60 subset of Extended Algol, then the hand translation will normally be limited to input/output. The (human) translator should be familiar with both B5500 Algol and OS/360 Algol and should be sure to run a few test cases on the translated program.

**Organization**

Paralleling the simple nature of the translation is the simple organization of the translation. Abstractly it is a finite state recognizer with left recursion, needing no backup in its left-to-right scan.
The program is written in OS/360 Algol, which requires that string manipulation be carried out as the manipulation of integer arrays. While this is unnatural (as opposed to the natural handling of strings in PL/1) the experience gained in using the OS/360 Algol system, and the excellent compile time and especially run time diagnostic capability of OS/360 Algol were sufficiently compensatory.

The program translates a card at a time, reading each source card into an integer array. This integer encoding is then scanned for one of four basic elements -- identifier, reserved word, number or special symbol (which includes blanks). Identifiers and numbers are directly passed out in 029 card code form. Reserve words and special symbols require a multiplicity of actions and each is done by a procedure having the various relevant cases.

The special symbols are mapped into one or more output symbols. The most complicated case is the quote mark ('') which initiates a scan for a second quote mark, meanwhile directly emitting all characters encountered. The reserved words are more diverse and complicated in their requirements. The simplest action is to emit them quoted -- so BEGIN becomes 'BEGIN'. Other reserved words initiate simple mappings into special symbols, as is the case with OR, AND and NOT going respectively to |, &, and  . Certain reserved words and the statement containing them are deleted. Label declarations are unnecessary, while input/output statements are intractable, so both are deleted. FINOTHAN is considered a reserved word and its detection causes termination of the translation. The translation is both punched on cards and passed to the OS/360 Algol compiler. The compiler then complains about any text it cannot parse.

The translator is very simple in structure and is easily modified. Extensions could be made in two manners, first as prepasses to this
system or second as more cases in the handling of reserved words. A prepass could be written to handle the expansion of DEFINE'S, FILL'S and the translation of the WHILE and FOR-STEP-WHILE which are not found in OS/360 Algol. These constructions require keeping track of block structure, tasks unsuitable for the present translator. The second possibility, extending the cases in handling the reserve words, could in a more limited way than the above method, add scope to the translator. The extensions could include the deletion of lower bounds lists from array specifications. Nevertheless, the translator as it stands should allow a programmer familiar with both B5500 Algol and OS/360 Algol to translate and debug a 1000 card program in a day or two providing the program is written principally in the Algol 60 subset of B5500 Algol.

Acknowledgements

I would like to thank John Welsch for his critical attention, advice and moral support. Also, would I like to thank Miss Kathleen Maddern for her excellent error detection and typing.
Footnotes and References


5. The key is that semantic equivalence is harder to achieve going from B5500 Algol to PL/1 than it is going from Fortran to PL/1. One is misled by the syntactic differences in judging Fortran the harder translation. For the opposite view see reference 4.

6. The extended 59 character set is used instead of the ugly 48 character set. See appendix 2 of reference 2 (p. 94).

See also: Wayne Wilner, Algol to Fortran Translator, CGTM 15, June 1967
APPENDIX 1  HAND TRANSLATION TIPS

1) 'FOR' V := AE1 'STEP' AE2 'WHILE' BE 'DO' statement;

   Here AE stands for arithmetic expression, BE stands for
   Boolean expression, and V for variable.

   In translating this construction a new variable is declared and used.
   New variables will be represented by Q, Q1, Q2 . . .

   Hand translation:

   Q := AE1;
   'FOR' V := Q 'WHILE' BE 'DO'
   'BEGIN' statement; Q := Q + AE2 'END';

2) 'WHILE' BE 'DO' statement;

   Hand translation:

   'FOR' Q := 0 'WHILE' BE 'DO' statement;

3) FILL A[i1 , i2 , . . . , *] WITH value 1, value 2, . . . , value n;

   Value i are the values in the value list, and in B5500 Algol these
   may be numbers, strings or octal numbers. In the case of strings or
   octal numbers a simple translation may not be possible.

   Hand translation:

   J := lower bound of row;
   'FOR' I := value 1, value 2, . . . , value n 'DO'
   'BEGIN' A(/ i1 , i2 , . . . , J/) := I; J := J + 1 'END';

   Note: A, I, and value 1, value 2, . . . should agree as to type.

4) X MOD Y

   Hand translation:

   X - Y * (SIGN(X/Y) * ENTIER (ABS(X/Y)))

5) CASE AE OF

   BEGIN
   S1;
   S2;
   .
   .
   Sn;
   'END';

   S1 through Sn represent statements. In the following translation
   each CASE statement must have its own switch list.
Hand translation:

'BEGIN'
  'SWITCH' CASE := CASE 1, CASE 2, ..., CASE n;
    'GO TO' CASE (/AE/);
CASE 1: S1; 'GO TO' EXIT;
CASE 2: S2; 'GO TO' EXIT;
  
CASE n: Sn;
  EXIT; 'END';

6) 'IF' BE 'THEN' conditional statement;

Hand translation:
  'IF' BE 'THEN' 'BEGIN' conditional statement 'END';

7) Own variables

These should be moved to the outer block, while consistently changing any conflicts between the scopes of identical identifiers.

8) ALPHA variables

The translator changes these to REAL, but normally this will not suffice. A simple translation may not be possible.

9) Lower bounds in array specifications must be removed.

'PROCEDURE' P(A);
  'ARRAY' A(/O/);

Hand translation:

'PROCEDURE' P(A);
  'ARRAY' A;

10) Double statements

If double precision is required the LONG option should be used when calling the OS/360 Algol compiler. Mixing LONG and SHORT is impossible in OS/360 Algol.

11) Machine dependent constructions - stream procedures, partial words etc.

These create difficult reprogramming problems.

12) Input/output

Read the relevant sections of the Algol manual.

If the above tips do not lead to a painless conversion, try going to Fortran or PL/1 for real difficulty.
'CLMEN'T' A;,3;...E = GIVEN INTEGER VALUE FOR INSYMBOL/OUTSYMBOL
AFTER INITIALIZATION OR RESERVED WORD TABLE H, I, J, K ARE USED
FOR INTEGER VARIABLES AND MAY NOT LONGER BE ASSUMED TO HAVE
SYMBOLIC USE.
AUXRE= COUNTER FOR NUMBER OF CARDS TO BE CUTPLITE PER
CARD INPUT. CURRENT MAXIMUM IS 3.
CARD= CARD BEING PROCESSED.
CARDN= CARD NUMBERING PRODUCED BY SEQUENCE.
ENTRY= ENTRY IN RESERVED WORD TABLE. IT IS USED TO INDEX MP4
WHICH ALLOWS SWITCH TO APPROPRIATE CASE IN WORDS.
ENTRY= SWITCH FOR TRANSLATE ROUTINE. 1 = RESERVED WORD
= IDENTIFIER 3 = NUMBER 4 = SPECIAL SYMBO.
CP= COLUMN POINTED FOR CURRENT CARD
CUR= COLUMN POINTED FOR OUTPUT CARD
)= OUTPUT BUFFER
MK= RESERVED WORD TABLE. IT IS LINEAR AND ORDERED BY LENGTH
OF THE RESERVED WORDS. TABLE[1] GIVES THE START OF THE TABLE
FOR RESERVED WORDS OF LENGTH L. E N T A B [ I ] = END OF TABLE.
PREV=.COLUMN WHICH SAW ANOTHER CURRENT SYMBOL HAS BEEN
TRANSLATED. IF THERE NO SYMBO IS READ,
MAP[1/1]= MAPPING FOR SPECIAL SYMBOLS. MPL= MAPPING FOR RESERVED
WORDS.
**************
INTEG= ALMCY, MCOUNT, MOUNT; B X A Y L 2, HOND 2, SYM, BLANK, LET, DIGIT, LP, CUR,
INTEG= AXL 3, AY, UML, SYM, NTON, DIGIT, GP, CUR,
INTEG= XAY, CYLE, SYM, UMP, CUR,
INTEG= UML, XAY, GP, SYM, CUR,
INTEG= XAY, CYLE, SYM, UMP, CUR,
INTEG= UML, XAY, GP, SYM, CUR,
'CLMEN'T' ALI(SKIPS 1 LINES ON PRINTER
'FREECUT' ALI; 'VALUE', 1; 'INTEG', 1; SYM(14,1);
'COMMENT' SEQUENCE PLACES SEQUENCE NUMBERS ON CARDS.*********;
'PROCEDURE' SEQUENCE;
'BEGIN' 'INTEGER' I; 'INTEGER' 'ARRAY' CRU(1:13);
K:=CRU(I);
EIG(5/6/2)=K;
EIG(2/4/2)=CRU(7);
EIG(2/4/3)=CRU(777);
EIG(2/4/2)=CRU(7777);
EIG(2/4/2)=CRU(77777);
EIG(2/4/2)=CRU(777777);
EIG(2/4/2)=CRU(7777777);
EIG(2/4/2)=CRU(77777777);
EIG(2/4/2)=CRU(777777777);
EIG(2/4/2)=CRU(7777777777);
EIG(2/4/2)=CRU(77777777777);
EIG(2/4/2)=CRU(777777777777);
EIG(2/4/2)=CRU(7777777777777);
EIG(2/4/2)=CRU(77777777777777);
EIG(2/4/2)=CRU(777777777777777);
EIG(2/4/2)=CRU(7777777777777777);
EIG(2/4/2)=CRU(77777777777777777);
EIG(2/4/2)=CRU(777777777777777777);
EIG(2/4/2)=CRU(7777777777777777777);
EIG(2/4/2)=CRU(77777777777777777777);
EIG(2/4/2)=CRU(777777777777777777777);
EIG(2/4/2)=CRU(7777777777777777777777);
EIG(2/4/2)=CRU(77777777777777777777777);
EIG(2/4/2)=CRU(777777777777777777777777);
EIG(2/4/2)=CRU(7777777777777777777777777);
EIG(2/4/2)=CRU(77777777777777777777777777);
EIG(2/4/2)=CRU(777777777777777777777777777);
EIG(2/4/2)=CRU(7777777777777777777777777777);
EIG(2/4/2)=CRU(77777777777777777777777777777);
EIG(2/4/2)=CRU(777777777777777777777777777777);
EIG(2/4/2)=CRU(7777777777777777777777777777777);
EIG(2/4/2)=CRU(77777777777777777777777777777777);
EIG(2/4/2)=CRU(777777777777777777777777777777777);
EIG(2/4/2)=CRU(7777777777777777777777777777777777);
EIG(2/4/2)=CRU(77777777777777777777777777777777777);
EIG(2/4/2)=CRU(777777777777777777777777777777777777);
EIG(2/4/2)=CRU(7777777777777777777777777777777777777);
EIG(2/4/2)=CRU(77777777777777777777777777777777777777);
EIG(2/4/2)=CRU(777777777777777777777777777777777777777);
EIG(2/4/2)=CRU(7777777777777777777777777777777777777777);
EIG(2/4/2)=CRU(77777777777777777777777777777777777777777);
EIG(2/4/2)=CRU(777777777777777777777777777777777777777777);
EIG(2/4/2)=CRU(7777777777777777777777777777777777777777777);
EIG(2/4/2)=CRU(77777777777777777777777777777777777777777777);
EIG(2/4/2)=CRU(777777777777777777777777777777777777777777777);
EIG(2/4/2)=CRU(7777777777777777777777777777777777777777777777);
EIG(2/4/2)=CRU(777777777777777777777777777777777777777777777777);
EIG(2/4/2)=CRU(7777777777777777777777777777777777777777777777777);
EIG(2/4/2)=CRU(777777777777777777777777777777777777777777777777777);
**SOURCE PROGRAM**

```plaintext
LI;
'FLA' K:=1 'STEP' I 'UNTIL' NUMEND 'DO' CUTRD(K);
'FLA' 1:=1 'STEP' 1 'UNTIL' DO 'DO'
'BEGIN'
IASYMP('G', 'G'; (123456789)GHEJKLMNPQRSTUVWXYZ-/# < 
;(_->)*(?:"?[^!])"BUF(/I/ })
'END';
KPCAR:=1;
CALMAT:=CUTCMT+1;
'END' INCAR;
CUTRD
'COMMENT' AXISYM GETS REALSYMBOL FROM INPUT BUFFER BUF .***** ;
'INTEGER' 'PROCEDURE' AXISYM;
'BEGIN'
'IF' CP:=73 'THEN'
'BEGIN' CP:=14 LAGR, 'END';
AXISYM:='LEFT(CL/P); CP:=CP+1;
'END' AXISYM;
'COMMENT' IDLNLK RECOGNIZED IDENTIFIERS .***** ;
'PROCEDURE' IDLNLK(LENGTH,ITEM);
'TRUE' LENGTH; 'INCLUD' 'ARRAY' ITEM;
'BEGIN'
'INTEGER' K;
K:=1;
LENGTH:=1; SYMB:=AXISYM; PREV:="FALSE";
'IF' LENGTH:=LENGTH+1 'AND' SYMB='((SYMB=BLANK) OR (SYMB=LETDIG))'
'BEGIN' ITEM(/LENGTH/) := SYMB;
SYMB:=AXISYM;
K:=LENGTH;
'END';
LENGTH:=K;
'END' INCLUD;
'COMMENT' NUMPAD IS TRUE IF SYMBOL MAY APPEAR IN A NUMBER. IT
IS USED BY INCLUD TO FIND THE END OF A NUMBER. ***** ;
'PROCEDURE' NUMPAD;
'IF' SYMB=LETDIG OR SYMB=' ' THEN' NUMEND='TRUE' 'ELSE'
'IF' (SYMB=39 OR SYMB=49) 'THEN'
NUMAL='TRUE' 'ELSE' NUMEND='FALSE';
'COLUMN' INNUMU RECOGNIZED NUMBERS . ********** ;
PROCEDURE INNUMU(LENGTH,ITEM);
'TRUE' LENGTH; 'INCLUD' 'ARRAY' ITEM;
'BEGIN'
'INTEGER' K;
K:=1;
LENGTH:=1; SYMB:=AXISYM; PREV:="FALSE";
```
SRC STATEMENT

SOURCE PROGRAM

CC165
'FRC' LENGTH:=LENGTH+1 'ARRAY' NUMERAL 'DC'
CC166
'BEGIN' ITEM/LENGTH/J:=SYM; SYMB:=AXISYM; K:=LENGTH 'END';
CC167
LENGTH:=K;
CC168
'END' I[NUMB;
CC171
'CLMENT' GUPTENG OUTPUTS IDENTIFIES EL'T ACTUALLY MAY OUTPUT
CC172
ANY QUANTITY IN ITEM ARRAY. **************
CC173
'FRCCELL' LENGTH ITEM;
CC174
'VALUE' LENGTH; 'INHCl' LENGTH; 'INTEGER' 'ARRAY' ITEM;
CC175
'BEGIN' 'INTEGER' J,J, LAS;
CC176
LAS:=CUR+LENGTH-1;
CC177
J:=C;
CC178
'IF' LAS>76 'THEN'
CC179
'BEGIN'
'FOR' I:=CUR 'STEP' 1 'UNTIL' 72 'DO'
CC180
'BEGIN' J:=J+1; CUR(/NUMCR+I)]:=ITEM(/J) 'ENC';
CC181
NUMCR:=NUMCR+1;
CC182
LAS:=LENGTH+CUR-1; CUR:=1; 'GCIL' RAC;
CC183
'END';
CC184
'END';
CC185
'END' CUR:+1;
CC186
'END' GCIL;
CC187
'CLMENT' GUPTENG OUTPUTS NUMBERS ************
CC188
'FRCCELL' 'DUPTENG (LENGTH ITEM);
CC189
'VALUE' LENGTH; 'INHCl' LENGTH; 'INTEGER' 'ARRAY' ITEM;
CC190
'END' GUPTENG(LENGTH ITEM);
CC191
'CLMENT' UCISYM OUTPUTS SYMBOLS ************
CC192
'FRCCELL' UCISYM(SYM); 'VALUE' SYMB; 'INTEGER' SYMB;
CC193
'BEGIN' 'CLMENT' NEED LARGE NUMBER OF CASES ;
CC194
'IF' CUR>76 'THEN' 'BEGIN' NUMCR:=NUMCR+1; CUR:=1 'END';
CC195
CUR(/NUMCR+CUR/)]:=SYM; CUR:=CUR+1;
CC196
'END' UCISYM;
CC197
'CLMENT' LUTFILT OUTPUTS Item TOPECGCELL ROUTINES*****
CC198
'FRCCELL' LUTFILT LENGTH ITEM;
CC199
'VALUE' LENGTH; 'INTEGER' LENGTH; 'INTEGER' ARRAY ITEM;
CC200
'BEGIN' 'INTEGER' J,J;
CC201
'FOR' I:=1 'STEP' 1 'UNTIL' LENGTH 'DO'
CC202
LUTFILTITEM((0123456789abcdefghijklmnopqrstuvwxyz+/<>&))
CC203
:='E7' ITEM(/I) ));
CC204
'END';
CC205
'END';
CC206
'FRCCELL' LUTFILT ITEM(SYM,ITEM);
CC207
'STRING' MESSAGE; 'INTEGER' NSYM; 'INTEGER' ARRAY ITEM;
'CURRENT' used to print out deleted material. THE DEALTIL
MATERIAL IS NOT LONQ AND IS FOUNO IN ITEM. A MESSAGE IS ALSO
ISSUED.

'ElGIN' 'INTEGER' 1.

'CLITING((1,'**')

'CLITING1',MMENDED); 'OUTSTRING1(''CARD''

'CUTINTEGER1',CALCIT), 'NLET(1);

'FCN' 1:='STEP' 1 'UNTIL' NSYM 'ON'

'CLITSYMBOL(1,''(1:2900/0:ACDEFGHIJKLMNOPRSTUVWXYZ+-*/<>&()')

':;':; 'S&' 'SYMB',ITER(111)

'END' 'DELETE';

'CURRENT' SCANMILL SCANS AND DELETES UNTIL CHARACTER TIL,****;

'PROCEDURE' SCANMILL(TIL), 'VALUE' TIL; 'INTEGER' TIL;

'ElGIN'

'DONTSCAN:

'CLITSYMBOL(1,''(1:2900/0:ACDEFGHIJKLMNOPRSTUVWXYZ+-*/<>&()')

':;':; 'S&' 'SYMB');

'SYMP':='NEXTSYMP';

'IF' SYMP = TIL 'THEN' 'GO TO' DONTSCAN;

'NEXT':='TRUE'; 'CURRENT' DELETION INCLUDES TIL;

'END' 'SCANMILL';

'CURRENT' SCANMILL SCANS UNTIL CHARACTER TIL AND PLAACESYMBOLS
DIRECTLY IN OUTPUT BUFFER. IT IS USE FOR STRING AND COMMENTS.

'PROCEDURE' SCANMILL(TIL); 'VALUE' TIL; 'INTEGER' TIL;

'ElGIN'

'DONTSCAN:

'CLITSYMB(SYMP);

'SYMP':='NEXTSYMP';

'IF' SYMP = TIL 'THEN' 'GO TO' DONTSCAN;

'NEXT':='TRUE';

'END' 'SCANMILL;

'CURRENT' FINDS END, ELSE OR ;

'PROCEDURE' SCANENDORST()

'ElGIN'

'CLITEX:

'LENGTH', 'ITER', 'CODE';

'IF' CODE = 1 'THEN' 'END'; 'IF' ITER(1)/1 = 50 'THEN'

'BEGIN' 'CLITSYMB(50); 'GO TO' 'FOUNDEND'; 'END';

'END';

'ELSE' 'IF' CODE = 1 'THEN' 'BEGIN';

'IF' CODE = 41 'CODE':= 49 'THEN'

'SCITA' 'CARDST LENGTH', 'ITER', 'CODE1'; 'GO TO' 'FOUNDEND'

'END';

'END';

'END';

'LENGTH', 'ITER';
SOURCE PROGRAM

SOURCE STATEMENT

CC161 "GC IC" REJECT;
CC162 EGCHE:
CC163 "END" SCANHECESSAT;
CC164 "CLEMENT" $SWARCH SEARCHES RESERVED WORD TABLE FOR IDENTIFIER
CC165 IN ITEM AND IT RETURNS CODEL THE POSITIKA IN TABLE IF THE
CC166 IDENTIFIER IS A RESERVED WORD. ********
CC167 "ELECT" "PROCEDURE" $SWARCHITEM,LENGTH;
CC168 "VALUE" LENGTH; "INTER" LENGTH; "INTEGER" "ARRAY" ITMP;
CC169 "BEGIN" "INTEGER" J,K,LAST;
CC170 IF" LENGTH<MAXLEN THEN
CC171 "BEGIN"
CC172 "FOR" K:=TABLE1/LENGTH/1 'STEP' LENGTH 'UNTIL'
CC173 "END";
CC174 "BEGIN"
CC175 "END";
CC176 "END";
CC177 "END";
CC178 "END" "$SWARCH$;  
CC179 "CLEMENT" $SPECSYM PERFORMS APPROPRIATE TRANSLATION OF THE
CC180 SPECIAL SYMBS. ******** ;
CC181 "PROCEDURE" $SPECSYM(CHR);  
CC182 "VALUE" CHR; "INTER" CHR;
CC183 "BEGIN"
CC184 "SWITCH" CASE:= CASE1,CASE2,CASE3,CASE4,CASE5,CASE6,CASE7;
CC185 CASES:= CASE1,CASE2,CASE3;
CC186 "GL IC" CASE1/ MAP1(CHR) // 1;
CC187 CASE1: $LTSYM(94)$; $LTSYM(71)$; $GL IC$ $NITEM$; "CLEMENT" $=$ ;
CC188 "CASE2: GTU$UCLETE(94)$% ENCOUNERED REST OF CARD $CLLECTED '$',0,ITEM);
CC189 CP:= 73; $GL IC$ $NITEM$; "COMMENT" PERCENT ENCOUNERED ;
CC190 "CASE2: $LTSYM(49)$; $LTSYM(49)$; $LTSYM(49)$; SYM:= NTSYM;
CC191 SCANTILL(49);
CC192 $LTSYM(49)$; $LTSYM(49)$; $LTSYM(49)$; "$GL IC" $NITEM$;
CC193 "CLEMENT" CASE $SPECSYM$ AND $PASSED$ STRINGS $$$$ ;
CC194 CASES:= $LTSYM(49)$; $LTSYM(49)$; $LTSYM(49)$; "$GL IC" $NITEM$; "CLEMENT" $=$ ;
CC195 "CASE4: $LTSYM(49)$; $LTSYM(49)$; "$GL IC" $NITEM$; "CLEMENT" $<$ ;
CC196 "CASE5: $LTSYM(49)$; $LTSYM(49)$; "$GL IC" $NITEM$; "CLEMENT" $/$ ;
CC197 "CASE6: $LTSYM(49)$; $LTSYM(49)$; "$GL IC" $NITEM$; "CLEMENT" /* ;
CC198 "CASE7: $LTSYM(49)$; $LTSYM(49)$; "$GL IC" $NITEM$; "CLEMENT" (// ;
CC199 "CASE8: $LTSYM(49)$; $LTSYM(49)$; "$GL IC$ $NITEM$; "CLEMENT" STANDARD 1-1 MAP ;
CC200 "CASE9: $LTSYM(49)$; $LTSYM(49)$; "$GL IC$ $NITEM$; "CLEMENT" ** ;
SOURCE STATEMENT

SOURCE PROGRAM

CASES: CASE1 CASE2 CASE3 CASE4 CASE5 CASE6 CASE7 CASE8 CASE9 CASE10 CASE11 CASE12 CASE13 CASE14

AP:*, LENGTH, ITEM; SCARECROW: 'UC UC' WRITER,'CURRENT' DELETION

LET STATEMENTS WHICH CAN NOT BE AUTOMATICALLY TRANSLATED TO
FILL DOUBLE STATEMENTS ARE OF THIS TYPE:************;

MACROS:
LE: MACROS;

'CLAIM' AXIOM GETS REAL ITEM TO TRANSLATE, THEY MAY BE
OF FOUR TYPES: RESERVED WORDS, IDENTIFIERS, NUMBERS, SPECIAL
SYMBOLS ... IT PASSES THE ITEM AND ITS LENGTH AND A CODE
FOR THE ITEM TO THE TRANSLATE ROUTINE: **********;

'PROCEDURE' AXIOM(LENGTH, ITEM, CCU);

'INTEGER' LENGTH, CODE, 'INTEGER' 'ARRAY' ITEM;

'ELSE' 'CLAIM' GETS REAL SYNTACTICAL ITEM;

'IF' PREV THEN SYMB:=XSYM;

'IF' SYMB:=BLANK THEN

'BEGIN' LENGTH:=1; CODE:=4; ITEM(1/1):=SYM; PREV:='TRUE' 'END';

'ELSE' 'IF' SYMB<=10 THEN

'BEGIN' LENGTH:=1; CODE:=4; ITEM(1/1):=SYM; PREV:='TRUE' 'END';

'END' 'BEGIN' ITEM(1/1):=SYM; MACRO(LENGTH, ITEM); CODE:=

'END';

'END' AXIOM;

'CLAIM' TRANSLATE TRANSLATES ITEM TO CUTFIT FORM: **********;

'PROCEDURE' TRANSLATE(LENGTH, ITEM, CODE);

'VALUE' CODE;

'INTEGER' LENGTH, CCU;

'INTEGER' 'ARRAY' ITEM;

'CLAIM' 'WRITE' CASE:= CASE1, CASE2, CASE3, CASE4, CASE5, CASE6;

'CLAIM' CASE(1/CUBE);

CASE1: MACRO(LENGTH, ITEM, CCU);

CASE2: CUM(LENGTH, ITEM), 'GOTO' WRITER;

CASE3: CUM(LENGTH, ITEM), 'GOTO' WRITER;

CASE4: SPECIFY(SYM);

'END' TRANSLATE;

'CLAIM' START OF MAIN PROGRAM **********;

'CLAIM' INITIALIZE DATA SEG**************;
<table>
<thead>
<tr>
<th>SC</th>
<th>SOURCE STATEMENT</th>
<th>SOURCE PROGRAM</th>
<th>PAGE 01C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC40</td>
<td>READ (K) = S;</td>
<td>READ (7) = S;</td>
<td>READ (11) = L;</td>
</tr>
<tr>
<td>CC41</td>
<td>READ (7) = S;</td>
<td>READ (14) = S;</td>
<td>READ (3) = S;</td>
</tr>
<tr>
<td>CC42</td>
<td>READ (1) = S;</td>
<td>READ (8) = S;</td>
<td>READ (5) = S;</td>
</tr>
<tr>
<td>CC43</td>
<td>READ (6) = S;</td>
<td>READ (12) = S;</td>
<td>READ (2) = S;</td>
</tr>
<tr>
<td>CC44</td>
<td>ENFP (44) = 2;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC45</td>
<td>ENFP (53) = 2;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC46</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SC  SOURCE STATEMENT

SOURCE PROGRAM

PAGE 011

OC615  'COMMENT' END OF SYMBOL TABLE FOR RESERVED WORDS**********;
OC615  'COMMENT' THIS RESERVED WORD TABLE SET UP IS BASED ON AN IDEA OF SG;
OC615  NITEM:
OC615  NITEM(LENGTH,ITEM,CODE);
OC616  TRANSLATE(LENGTH,ITEM,CODE);
OC617  'END' NITEM;
OC618  TERMINATE;
OC618  'END'