DISCLAIMER OF QUALITY

Due to the condition of the original material, there are unavoidable flaws in this reproduction. We have made every effort possible to provide you with the best copy available. If you are dissatisfied with this product and find it unusable, please contact Document Services as soon as possible.

Thank you.

Appendix 1 contains poor quality text.
A CHESS PLAYING PROGRAM FOR
THE IBM 7090 COMPUTER

by
Alan Kotok

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF SCIENCE

at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June, 1962

Signature of Author

Department of Electrical Engineering

Certified by
Thesis Supervisor

Accepted by
Chairman, Departmental Junior Thesis Committee
ABSTRACT

This paper covers the development of a chess playing program. The preliminary planning led to the decision to use a variable depth search, terminating at either an arbitrary maximum, or at a stable position. Two schemes of controlling material balance are discussed. Of major significance is the use of the "alpha-beta" heuristic, a method of pruning the tree of moves. This heuristic makes use of values obtained at previous branches in the tree to eliminate the necessity to search obviously worse branches later.

The program has played four long game fragments in which it played chess comparable to an amateur with about 100 games experience.
ACKNOWLEDGMENT

I wish to thank Michael Lieberman, Charles Niessen, and Robert Wagner, the current members of the MIT chess group, for their invaluable assistance in this project. I also wish to express my appreciation to Elwyn Berlekamp, B. F. Wells and Paul Abrahams, who were previously associated with this project.

Special thanks go to Prof. John McCarthy who has guided the chess program through good days and bad. I wish to acknowledge the cooperation of the MIT Computation Center for providing the computation facilities necessary for this project.

Lastly, I wish to thank Robert Saunders for his help with the programming, and Milton Garber and Robert Fiorenza for giving their time to play against the machine.
# TABLE of CONTENTS

Abstract .................................................. -11-
Acknowledgment ............................................. -111-
Introduction ................................................. -1-
Preliminary Investigation ................................. -1-
Organization of the Chess Program ...................... -4-
Description of Component Sub-Programs ................. -7-
  Administrative Routines ................................. -7-
  Plausible Move Generation .............................. -8-
  Evaluation Routines ...................................... -8-
  Service Routines .......................................... -10-
  Input-Output Routines .................................... -11-
Results ...................................................... -12-
  Figure 1 - representative output .................... -13-

Appendix 1 - Listing of the Chess Program
Appendix 2 - Sample Input to Initia
Appendix 3 - Record of Chess Games
INTRODUCTION

This thesis describes a chess playing program for the IBM 7090 computer. Although chess programs have been previously written, none of these played what could be considered "good chess".

Before commencing work on our chess program, we studied the report published by Newell, Shaw and Simon covering previous attempts, such as the Los Alamos program, and Bernstein's program at IBM.

PRELIMINARY INVESTIGATION

The chess group, consisting of Messrs. Berlekamp, Niessen, Lieberman and Kotok, inherited routines for generating and making legal moves. With these as a basis, we decided to write a three move mate solving program for the purpose of familiarizing ourselves with the existing routines, and to come in contact with many of the problems we would later face in the actual general playing program. The three move mate program was completed in the spring of 1960. It was given problems from actual games, and successfully solved many of them. The three move mate program was written for the IBM 704, which was removed from the MIT Computation Center in the summer of 1960. Due to incompatibility with the incoming 709, the project was dropped at the end of the spring term of 1960.

In the fall of 1960 the chess group, without Mr. Berlekamp, began planning for the general chess program.
It was decided to retain the original McCarthy-Abrams move routines, and to continue coding in FORTRAN and FAP. The program was to be a variable depth search with a "stable position" termination. An evaluation was to be made at the terminal points of the move tree. This evaluation would be a weighted sum of such criteria as material balance, center control, pawn structure, "tempo" advantage, and development.

Moves on each level were to be proposed by "plausible move generators" which would propose moves to fulfill various goals. As the tree was searched, a backing up process would take place, in which the move declared best at each level by the evaluation would have its value brought up to the next higher level.

This procedure, also called mini-max, leads to a "principal variation" which is that set of moves which the machine considers most likely to happen. The evaluation always assumes that a player will always make the best move available to him a given time.

It was, of course, recognized that any evaluation could not be perfect, since chess is a game in which the only way a position can be perfectly evaluated is to look to the end of the game, and see whether it leads to a win, draw, or loss. The only sound basis for an evaluation is that chess masters have, over the years, accumulated knowledge concerning the play of the game. For instance, a position in which a piece is "en prise" is considered
bad, while having rooks on open files is considered good, even though the rules do not state anything about such things.

Since none of the members of the chess group are more than amateurs, we consulted books by masters to find out how much better it is to control the center than to have a strong pawn structure. These books are amazingly elusive on such details. Although many tips were given concerning the play of the game, relative importance of various strategies was uncertain.

We therefore considered having the program play for a while, and adjust the weights of the evaluation criteria to optimize its position. Although such a scheme seemed desirable, it was decided not to include any "learning" in the program due to the unavailability of suitably large amounts of computer time.
ORGANIZATION OF THE CHESS PROGRAM

Work on the chess program itself began in the spring term of 1961. The program is written in subroutine form, using the Fortran Monitor System of linkage. Where possible, programs are written in FORTRAN, and where it becomes too clumsy, or inefficient, FAP is used.

The actual implementation of the above mentioned "plausible move generators" has never been accomplished. Instead, we have a program, called REPLYS, which scans the legal move table, updates, evaluates, and reverts each move and orders them according to a single ply evaluation. (A ply is a half-move, i.e. a move by only one side.) The number of moves actually chosen is a function of the current depth in the tree.

Evaluation functions were written for material balance, center control, and development, since we intended to concentrate our efforts on openings until the program was thoroughly debugged.

The coordinating routine written in the spring of 1961, called TREE, employed the above mentioned mini-max scheme. REPLYS was set to cut the search at a depth of eight plys, or whenever the situation was stable, whichever came first.

The program was tested late in the spring of 1961. The 709 took about 5 to 20 minutes per move, depending on the complexity of the situation. Although the machine did not do too badly, we noted that it was looking at many
irrelevant positions. We therefore attempted to find a method of pruning the move tree, without discarding good as well as bad moves.

Prof. McCarthy proposed a heuristic for this purpose, called "alpha-beta". It operates as follows: Alpha is a number representing the value of the best position which white can reach, using a pessimistic evaluation. Beta represents the best position white can reach, using an optimistic evaluation, due to the fact that black can hold him to this position. Under normal circumstances, alpha starts at $-\infty$, and beta at $+\infty$. At each level, optimistic and pessimistic evaluations are made, and compared to alpha and beta in the following way. If a white move is optimistically less than alpha, it is discarded, since a better alternative exists elsewhere. Likewise, if a white move pessimistically is better than beta, it too is discarded, since black had a better alternative previously, furthermore we revert two levels since no other white moves are worth considering at that position. The reverse strategy is applied for black.

The "alpha-beta" version of TREE was written during the summer of 1961, and was first put to use during the fall of that year. Also, we were joined by Mr. Wagner in the fall term of 1961.

After testing in the fall of 1961, it was decided that the material balance programs were insufficient. We therefore decided to replace the scheme then in use with
a new, updated scheme. The programs then in use, and, as it happens, in use now, completely re-generate the material balance function at each position.

The material balance evaluator consists of two subroutines, SWAP and LTRADE. SWAP's function is to list all attacks and defences on each occupied square. Secondary attackers which reside behind primary attackers (or defenders) are included. The pieces are listed in the order in which they would be played. Lowest valued pieces come first, unless the order is disturbed by the necessity of a higher valued piece to move first due to position. Pieces pinned to the king and queen were not recognized, leading to embarrassing evaluations. Likewise, discovered attacks were not considered.

LTRADE then simulates trade-off of all attacked pieces, and chooses the line most profitable for the side to move. The opponent is given the option of having a given piece taken, or moving the piece away. After all possible trades have been made, the program computes whether it is to the advantage of the machine to initiate an exchange, and if so, what the probable gain would be.

This scheme is both time consuming, and occasionally inaccurate. It was therefore decided to write a new evaluator for the material balance, which kept an updated set of tables, in a list structure format, from which the outcome of a given exchange could be found at a glance.
After a few months of planning and programming, the new list structure program was found to be impractical, due to excessive complication in the update procedure. Furthermore, the values which were to be included in the list were found to be no more accurate than the ones which the above scheme produced. The project was therefore abandoned.

DESCRIPTION OF COMPONENT SUB-PROGRAMS

The chess program is organized into a non-recursive hierarchy of sub-programs. Listings are to be found in appendix 1.

ADMINISTRATIVE ROUTINES

(MAIN) This is the highest level program. The on-line main program has the job of handling input-output, and timing. It determines the opponent's move by looking at the console keys, and picks the appropriate move from the legal move table. It then calls TREE which actually makes the move, after which (MAIN) prints out the machine's reply.

TREE Tree is the second level of control. Tree has the responsibility of constructing the tree of legal moves. It calls REPLYs to generate a list of plausible moves, and enters these in the LISP table, which is the actual tree. The moves are then chosen in order of decreasing value, and
updated. A new list of plausible moves is then generated for the opponent. The optimistic and pessimistic evaluators are called, and the alpha-beta tests are made, as described above. In the event that no replies are generated, due to stability, or excessive depth, a static evaluation is made and assigned to the position. The last move is then reverted, and the search proceeds down the next most likely branch of the tree. When all desired positions have been examined, the "best" move is returned as the answer.

PLAUSIBLE MOVE GENERATION

REPLYS. This program supplies lists of plausible moves to TREE. It updates each of the legal moves, evaluates the position and reverts. The number of moves presented is a function of the present ply. Current values in order of increasing ply are: 4 3 2 2 1 1 1 1 0 0. These are input parameters to the program.

EVALUATION ROUTINES

EVAL Eval is the static evaluation program. Its function is to call all the subsidiary evaluation programs and to apply suitable multipliers, and hence form a weighted sum. Material values are: pawn 1, knight and bishop 3, rook 5, queen 9, and king 1000. These values are normally multiplied by 60 when combined with the other functions. Should one side be ahead at least 4 points, the material multipliers are adjusted to make trading
off advantageous.

LTRADE This program, described in more detail above, provides the projected material gain, considering all attacks and defenses.

ICENTR The center control evaluator gives points for controlling the 16 center squares. Looking from either side, these values are:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The center control points are each worth 1/60 of a pawn. After the game passes the twentieth full move, the center control function is decreased in importance until the 30th move, when it is discarded.

IDVLOP The development function, gives points for each developed piece. These range from 1 point per pawn, to 3 or 4 points for other pieces. Development points are weighted 1/15 of material points. This function is also eliminated as the game progresses.

JPAWNS The pawn structure function, considers the following situations, with approximate point values:

open file +8
isolated pawn -1
backward pawn -5
doubled pawn -3
passed pawn +10

These points are weighted 1/20 of material points.

SERVICE ROUTINES

UPDATE Updates any legal move, and records all relevant information on a push-down list. It then generates all legal replies available to the other side, using the general purpose move routines UPREV and PUTCH.

REVERT Takes back the last updated move. This is actually another option of the the updating routine UPREV.

PUTCH A lower level routine used in making moves. It keeps tables of almost legal moves and piece bearings updated. This table does not include castling, and "en passant" moves.

SWAP Generates the list of all attacks and defenses on occupied squares, listed in the order in which the pieces would be played.

PINS Generates the list of all pieces pinned to Kings and Queens. Includes the pinning direction, so that SWAP will only consider a pinned piece as an attacker or defend-
er along the line of the pin.

**INPUT-OUTPUT ROUTINES**

**PRINT**  The major output routine. It handles most of the printing, both on and off line. It, and its subroutines, print the chess board, legal move table, principal variation, move tree and log of all moves tried, plus other information useful in debugging.

**INITIA**  Reads in any chess board position. Its input language is as follows:

The chess board is scanned, from left to right, starting at white's Queen Rook 1. Digits represent numbers of unoccupied squares. Pieces are represented by the normal chess notation, in its most explicit form, e.g. KBF for King Bishop Pawn. Black pieces are preceded by asterisks. After exactly 64 squares are specified, the character"." (period) signifies the end of the specification and that white is to move. "*" indicates black to move. Additional features include the ability to indicate promoted pawns, by stating the type of piece, followed by the name of the pawn from which it promoted, in parentheses, e.g. Q(KNP). Also, it is possible to indicate that a piece has previously moved (for rooks, kings and pawns) by suffixing (M) to the piece name. Comments must begin and end with slashes.

The input is on IBM cards, punched in columns 1
through 72, taking as many cards as necessary. In case of errors found by INITIA, a comment will be printed, the remaining part of the problem will be skipped, and the next problem will be used.

All tables are initialized, and the program is set to commence with the legal move table generated for the side indicated. An example of an INITIA input will be found in Appendix 2.

RESULTS

As of this date, the machine has not completed any chess games. We have, however, played 4 lengthy fragments of games, and also have investigated many individual positions.

For our first long machine run, we chose an undergraduate student, Milton Garber, who held second place in his dormitory chess tournament. A record of this, and other game fragments is to be found in Appendix 3.

The second game was also played against Mr. Garber. In the record of this game a column indicating the principal variation is included. These are the moves the machine considers most likely to happen in succeeding plays, based on the evaluation and minimax process.

In seventeen moves, the machine guessed correctly only thrice, including only one case where it predicted correctly more than one move ahead.

Figure 1 consists of a set of representative
SET OF TABLES NUMBER 4 MOVE IS *QP - Q4

BLACK

* QR * QN * QB * Q * K * KB * KN * KR *

* * * * * * * * * * * * * * * * * * * * *
* QRP * QNP * QBP * * KP * KBP * KNP * KRP *

* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *

* * * * * * * * * * * * * * * * * * * * *
* QRP * QNP * QBP * * KP * KBP * KNP * KRP *

* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *

WHITE

MAVAIL

K - Q2  QRP-QR3  QRP-QR4  QNP-QN3  QNP-QN4  QBP-QB3  QBP-QB4  KP - K3  KP - K4  KBP-KB3


Q - Q2  QBP-KB4  QB -KN5  QB -KR6  Q - Q2  Q - Q3

FIGURE 1

Representative output
THE MOVE TREE

LEVEL 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 VALUE

*QP - Q4
  QN - QB3
  KN - KB3
  QB - KB4
  *QN - QB3
    QN - QB3
    *KN - KB3
    KN - KB3
    *QB - KB4
    QB - K5
  KN - KB3
  *KN - KB3
    *QB - KB4
    QN - QB3
    QB - K5
  *KN - KB3
  *QN - QB3
  QP - Q5
  *QN - QN5
    KP - K4
    QN - QB3
    *QN - QR4
    KP - K4
    QN - QB3
    KP - K4
    QN - QB3
    *KN - KB3
  QN - QB3
  KN - KB3
  QB - KB4
  *QP - Q3
  KP - K4
  QN - QB3
  KN - KB3

*Fig. 1 (cont)

VALUE -15-
output for a single move. The first page is a printout of
the chess board, and a list of the opponents legal replies,
labeled MAVAIL. The second page contains the principal
variation, beginning with the value of this variation, and
the number of positions examined at the approximate rate of
1100 positions per minute. The principal variation itself
commences with the machine's move.

The following pages contain the actual move tree.
The moves listed therein are moves which were considered
plausible by the reply generator. Moves were considered in
the order top to bottom, however all moves on level one
were generated simultaneously, and all level two replies
to each level one move are generated together, etc. The
"value" column contains a value on each terminating posi-
tion. Values of $+131071$ indicate positions discarded for
alpha-beta cutoff. Terminating positions which have no
values have not even been examined, since the alpha-beta
heuristic found previous moves on that level to be either
too good, or too bad.

A third game fragment was played against an
amateur with little chess experience, in particular, he
knew the game, and had played some before he came to MIT.
The game progressed $3\frac{1}{4}$ moves before time expired, with
the result that the machine was ahead 1 rook, 2 knights
and 2 bishops.

From our analysis of the results, we have found
that in its present state, the program is comparable to
an amateur with about 100 games experience.

Most of the machine's moves are neither brilliant nor stupid. It must be admitted that it occasionally blunders. These blunders can often be traced to wrong multipliers in the evaluation, and occasionally to situations where discovered attacks, forks, etc. cause confusion. It is rare, however, not to find the correct move in the list of plausible moves.

This study is far from complete, but we feel that our efforts are proving fruitful. Hopefully this work will be continued.
APPENDIX 1

* LABEL
* FAP
* COUNT 400
* TREE FUNCTION FOR CHESS WITH ERROR PRINT, MAR. 2, 1962
*
* GIVEN A MOVE AS THE FIRST ARG, IT GENERATES A TREE OF MOVES,
* MINIMAXES, AND ITS VALUE IS THE DESIRED REPLY IN "MOVE" FORMAT.
* THE FORMAT OF THE TABLE IT GENERATES (CALLED LISP) IS AS FOLLOWS-
*
* MOVE    BACK
* VALUE    PLY N
* REPLY(1) POINTER(1)
* REPLY(2) POINTER(2)
* ... ...
* =REPLY(N) POINTER(N)
*
* THE ABOVE IS 1 BLOCK IN THE LISP TABLE. IT IS GENERATED ONLY ONCE
* MOVE IS THE MOVE UNDER CONSIDERATION, IN BITS 3-20. THE SIGN MAY
* BE NEGATIVE IF THERE ARE NO PROPOSED REPLIES.
* BACK IS THE INDEX OF THE FIRST WORD OF THE BLOCK FROM WHENCE
* WE CAME. (NOTE- ALL SUCH INDICES MAY BE OFF BY A CONSTANT.)
* VALUE IS THE VALUE OF THE MOVE AS DETERMINED BY MINIMAXING.
* N IS THE NUMBER OF REPLIES NOT YET CONSIDERED, WHICH IS COUNTED
* DOWN TO ZERO, AT WHICH TIME THE MOVE IS EVALUATED, AND N BECOMES
* THE INDEX OF THE REPLY THAT LED TO THE VALUE CHOSEN.
* SINCE THE ABOVE EXPLANATION IS SO CLEAR, COMMENTS WILL NOT BE
* PROVIDED ADJACENT TO THE PROGRAM, SINCE THESE WILL ONLY SERVE TO
* ADD TO THE ALREADY ABUNDANT CONFUSION. SO HERE IT IS.............
* NEXT FREE REGISTER IN COMMON = 23375
*
* INITIALIZE
* ENTRY TREE
* TREE SXA XR1,1
* SXA XR1+1,2
* SXD XR4,4
* XR4 SYN TREE-2
* AXT 3000,1
* STZ LISP+1,1
* TIX *-1,1,1
* STZ MOVE
* CALL STRTG
* AXT 1,1
* STZ BACK
* STZ PLY
*
* D
* CLA PLY
* ADD =Q200
* STO PLY
* CLA MOVE    HEAD NEW BLOCK
* ADD BACK
* SXA BACK+1
* STO LISP+1,1

*
BEGIN COMPARISON OF A x B 2 BLOCKS HIGHER

FAIL TEST 1 -- REVERT TWICE. (PASS, TRA F01)

(SINGLE REPLY CHAIN -- GO BACK 2 MORE LEVELS)
ORA A
ADD =1
SLW LISP,2
CLA LISP+1,2
TRA FF

* TSX $PESVL,4
SPG TSX $OPTVL,4
TSX $PESVL,4
CAL =0377777000000
TS1 TRA FO1
CAL =0777777000000
TS2 TRA OUT
CAL =0377777000000

* FO1 CLA LISP+1,1
PAX ,2
TXL OUT,2,0
CLA LISP-1,0
TPL +6
CLA LISP+1,2
PAX ,2
TXL OUT,2,0
CLA LISP+1,2
TRA FO1+1

NM XEC SPG+2,4
TXI +2,0,0
PZE TREE-2,0,NM
STO VALUE
LDI =2
STI 1D
LXD MCOL,4
CAS LISP,2
XEC TS2+2,4
TRA +2
XEC TS2+1,4

SLW A
ADD PLY FAIL TEST 2 REVERT (PASS, TRA OUT)
SLW LISP,1
CLA LISP+1,1

FG PAX ,2
CALL REVERT
CLA LISP-1,2
TMI GF SINGLE REPLY EMAIL--GO BACK 2 MORE LEVELS
CLA PLY
SUB =0200
STO PLY
TRA DN

GF CAL LISP,2
ANA =0777777
ORA A
ADD =1
SLW LISP,2
CLA PLY
SUB =0400
STO PLY
CLA LISP+1,2
PAX ,2
CAL LISP,2
ANA =07777777
ORA A
ADD =1
SLW LISP,2
CALL REVERT
CLA LISP+1,2
TRA FG

ORA =037777700000
INT ORA =077777700000
OUT CALL REPLYS
LXD IPE,2
PX A ,2
ADD PLY
SLW LISP,1
TXL B+2,0 (NO REPLYS--POSITION STATIC)
LXD MQOL,4
XEC INT+1,4
SLW LISP,1
SX D AF,2
AX T 1+2
Q CLA IHOP E+1,2
STO LISP-1,1
TX I *+1,1,1
TX I *+1,2,1
AF TX L Q+2,**
TX H ERR,1,3000
CAL =-0
ORS LISP,1
TX I C+1,2
B CAL =-0
ORS LISP+1,1
TXI NOMGVE,1+2

ERR CALL ERROR,FMT
TSX $LDUMP,4
+MT BCI 2,LISP FULL.
SVN -1,7,-1

UPDATE THE NEXT REPLY WITHIN A BLOCK.

C LXA BACK,2
CLA LISP,2
ANA =0177
TZE USEDUP (ALL REPLIES USED UP)
ADD BACK
PAX ,4
SXA G,4
CLA LISP,4
STO RMOVE
SLW MOVE
CALL UPDATE, MOVE
CLA REMOVE SHIFT PROMOTION INFORMATION
LRS 0
STD AA
ALS 15
ANA =0700000
ADD AA
LLS 0
G AXT **, 4
STO LISP +4
SLW MOVE
PXA +1
STA LISP +4
CAL LISP +2
SUB =1
SLW LISP +2
TRA D

* THERE ARE NO ENTRIES IN IHAPER. EVALUATE THE QUOTE
* STATIC UNQUOTE POSITION.

NOMOVE CLA LISP +3, 1
STA BACK
CALL EVAL
ORS LISP +2, 1
CLA LISP +2, 1
LK LXD MCOL +4
LXA BACK +2
CAS LISP +2
XEC BRN +1, 4
TRA +2
XEC BRN +2, 4
STO A
CAL A
STP LISP +2
CLA PLY CHANGE NPLY
ARS 7
PAX +4
CLA LISP +2
ANA =0177
ADD =01
STO NPLY +2, 4
CLA LISP +1, 2
STA A
NIN LXA A +4
TXL OT +4, 0
CLA LISP -1, 4
TPL IN
CLA LISP +1, 4
PAX +4
TXL OT +4, 0
CLA LISP +1, 4
STA A
TRA NIN
IN CLA LISP +2
LXD  MCOL, 4
LXA  A, 2
CAS  LISP, 2
XEC  BNR+1, 4
TRA  *+2
XEC  BNR+2, 4
LXA  BACK, 2
GRA  CALL  LISP, 2
ADD  =1
SLW  LISP, 2
CLA  LISP+1, 2
PAX  , 4
CLA  LISP-1, 4
THI  ARG  (SINGLE REPLY CHAIN)
SXA  BACK, 4
CLA  PLY
SUB  =0400
STO  PLY
CALL  REVERT
CALL  REVERT
TRA  C

* STO  TEST
BNR  TRA  OT
STO  TEST

* ARG  CLA  LISP+1, 4
PAX  , 2
LDQ  TEST
SLQ  LISP, 4
SLQ  LISP, 2
CAL  LISP, 4
ADD  =1
SLW  LISP, 4
CLA  PLY
SUB  =0400
STO  PLY
CALL  REVERT
CALL  REVERT
TRA  GRA
OT  CLA  PLY
SUB  =0200
STO  PLY
CALL  REVERT
TRA  C
STD  LISP, 2
BRN  TRA  OT
STD  LISP, 2

* *
USEDUP  CLA  LISP+1, 2
STA  BACK
CLA  PLY
ARS  7

PASS TEST (FAIL, TRA OT)
DONE  CLA   LISP-1
       ANA   =0177
       SXA   SHMACK,1
       PAX   ,2
       CLA   LISP-1,2
       STD   AA
       ANA   =0700000
       ARS   15
       ADD   AA
       SXA   RX4,4
       TSX   PT11,4
       CLA   AC
       LXA   RX4,4
       XRI   AXT  **,1
       AXT   **,2
       LXD   XR4,4
       STO*  1*,4
       TRA   2*,4

MOVE  PZE
BACK  PZE
AA    PZE
A     PZE
REMOVE PZE
TEST  PZE
VALUE PZE
M     PZE
      BSS    19
NPLY  BSS    1
*     PT     SXA   WW,4
         TSX   PTA,4
         LXA   RX1,1
         LXA   RX2,2
         LXA   WW,4
         TRA   1*,4
         PT11  LDI   =11
         STI   1D
         SXA   WW,4
         TSX   PTA,4
         TSX   PTL,4
         TSX   PTP,4
         TRA   TP

EXIT GLEefully WITH THE BEST MOVE IN THE aC.

REPLACE PROMOTION INFORMATION
ID PZE
AC PZE
WW PZE
RX1 PZE
RX2 PZE
RX4 PZE
*
PTA SXA RX1,1
   SXA RX2,2
   SXA QP,4
   STO AC
   ORS AC
   TSX $SPH,4
   PZE FMTT,,-1
   LDG ID
   STR
   LDG AC
   STR
   LDQ LISP,2
   STR
   LDQ PLY
   STR
   LDQ MCOL
   STR
   LDQ BACK
   STR
   LDQ MOVE
   STR
   LDQ IPE
   STR
   LDQ RX1
   STR
   LDQ RX2
   STR
   LDQ RX4
   STR
   LDQ VALUE
   STR
   TSX $(FIL),4
   QQ AXT **,4
   TRA 1,4
FMTT BCI 6,(144 THIS IS POINT,057(60201))
*
PTL SXA BK,4
   CLA RX1
   ADD =2
   ALS 18
   STD EP1
   TSX $SPH,4
   PZE FOR,,1
   AXT 1,1
   RK PXA ,1
   XCA
   STR
LDQ       LISP+1,1
STR       *+1,1,1
TXI       RK,1,**
EPI       TSX       $ (FIL),4
TXL       BK        AXT       **,4
TSX       TRA       1,4
*          PTP       SXA       GB,4
TSX       $ (SPH),4
PZE       AXT       1,1
FOR       SIK       PXA       1
*          XCA       STR
LDQ       NPLY+1,1
STR       *+1,1,1
TXI       TSX       SIK,1,10
TXL       $ (FIL),4
GA         AXT       **,4
GB         TRA       1,4
FOR       BCI       3,(// (10X,04,020))
*          ZILCH     COMMON  12561
R          COMMON  1
K1         SYN       R+9670
MCOL       SYN       R+9662
IPE         SYN       R+9442
PLY         SYN       R+9441
SHMACK      SYN       R+9440
IHOPe       SYN       R+9439
LISP       SYN       R+9379
END
LABEL
FAP
*SWAP  SOUBROUTINE,  FOR  MATERIAL  BALANCE,  3/5/62
COUNT  250
*
GENERATES  THE  IEXCH  TABLE  WHICH  CONTAINS,  FOR  EACH  PIECE,  ALL
ATTACKERS  AND  DEFENDERS,  LISTED  IN  ORDER  OF  USAGE.  THE  TABLE
IS  ARRANGED  AS  FOLLOWS----
ENTRIES  1  THRU  33  CONTAIN  INFORMATION  ABOUT  EACH  PIECE.
THE  DECREMENT  CONTAINS  THE  INDEX  OF  THE  BEGINNING  OF  ENTRIES
IN  THE  REST  OF  THE  TABLE  FOR  THAT  PIECE,  THE  END  OF  SUCH  ENTRIES
THE  TAG  CONTAINS  THE  NO.  OF  ATTACKERS  AND  THE  PREFIX  HAS  THE  NO.
OF  DEFENDERS.  THE  ADDRESS  CONTAINS  THE  FIRST  USE  OF  THIS
PIECE  AS  AN  ATTACKER  OR  DEFENDER.  THIS  WILL  BE  ZERO  IF  NOT  USED.
THE  REST  OF  THE  TABLE  CONTAINS  THE  LIST  OF  ATTS.  AND  DEFS.
THE  DEC.  OF  A  WORD  WILL  CONTAIN  THE  ATT.  OR  DEF.  PIECE  NUMBER.
THE  TAG  CONTAINS  (IF  THE  SIGN  BIT  IS  1)  THE  INDEX  RELATIVE  TO
THE  BEGINNING  OF  THIS  PARTICULAR  SET  OF  ENTRIES  OF  THE  PIECE
WHICH  MUST  MOVE  FIRST  DUE  TO  MASKING.  THE  ADDRESS  CONTAINS
MORE  OF  THE  CHAIN  OF  USES  OF  THIS  PIECE.
THE  ADDRESS  WILL  BE  ZERO  IF  THIS  IS  THE  LAST  USE  ON  THE  CHAIN.
*
ENTRY  SWAP
SWAP  LDQ  =070707070707070707
SXA  XR1+1
SXA  XR1+1+2
SXD  XR4+4
XR4  SYN  SWAP=2
STI  INDIC
AXT  33,1
PXK  1
STK  CHAIN+1,1
STZ  IEXCH+1,1
TIX  #=-3+1,1
AXT  34,1
SXD  COUNT+1
AXT  1+1
A
SXD  K,1
CLA  LOC+1,1
STZ  ATTACK
TZE  Y
SUB  =1B177
PDX  2
AXT  960,4
AXT  0+1
ZET  IBEAR,6
TXI  C+1,1
D
TIX  #=-2+4,64
TXL  D1+4,0
ZET  IBEAR,2
TXI  D-1+4,64
D1
CLA  COUNT
TXH  ORDER,1,0
E SXD ATTACK, 1
   LX D K+1
   STD IEXCH+1, 1
   CLA ATTACK
   ALS 15
   ST P IEXCH+1, 1
   TX I ++1, 1
   W TX L A+1, 32
   CLA COUNT
   ST O IEXCH-32
   AXT 32, 2
   Z CLA CHAIN+1, 2
   PAX 1
   PX D IEXCH+1, 1
   STA ++2, 1
   XR1 AXT ++1, 1
   AXT ++2, 2
   LXD XR4, 4
   LDI INDIC
   TRA 1, 4

* TX I D+1, -1
   Y CLA COUNT
   TR A E+1
   XR 1* PHASE 1, SET UP INTER WITH ALL BEARERS IN RANDOM ORDER
   C CLA IBEAR, 6
      TMI X
      SX A F, 2
      PDX 2
      PA I
      ZET KPIN+1, 2
      TR A PIND
   C1 STO INTER+1, 1
      II S K
      LFT 1
   H STL ATTACK
   CLA LOC+1, 2
   SUB =1B17
   PDX 2
   CAL IBEAR, 6
   TN Z 6
   A XT ++2, 2
   TX L D+1, 19
   TR A LOSE
   G PDX 2
   LDI KIND+1, 2
   LFT 1
   TR A F
   PA I
   ZET KPIN+1, 2
   TR A PINK
   G1 II S INTER+1, 1
   LFT 1

SET BEG FOR UNATTACKED PIECES
CLOSE OF MAJOR PIECE LOOP
SET LAST BEG (REALLY END FOR PC 32)
ZERO ADDRESSES OF PIECES NOT USED
RESTORE INDEX REGISTERS
RETURN
USED FOR VERTICAL PAWNS
USED FOR PIECE OFF BOARD
PHASE 1, SET UP INTER WITH ALL BEARERS IN RANDOM ORDER
PICK UP BEARER
TRA IF VERT PAWN
SAVE SQUARE
PIECE TO XR2
IS THIS PIECE PINNED
YES
ENTER BEARER IN INTR
YES, SET FLIP-FLOP
NO
XR2 HAS LOC(BEARER)-1
DO WE HAVE MASKED PIECE
YES
NO, RESTORE XR2 TO ORIGINAL PIECE
AND EXIT
CAN PIECE BE MASKED
NO, PAWN KNIGHT OR KING
YES, ARE COLORS THE SAME
IS THIS PIECE PINNED
TRA F
ORA =1
STO INTER,1
TXI H+1,1

* PIND CLA KPIN+1,2
PDX ,2
TMI PIND1
PIND2 PXD ,4
ARS 6
ADD #1B17
STO PINDIR
PXD ,2
CAS PINDIR
TRA ,+2
TRA PIND3
CLA IOPP+1,2
CAS PINDIR
TRA ,+2
TRA PIND3
PIND4 TXH PIND4,2,0
PAX ,2
TRA PIND2
PIND3 PIA ,2
PDX C1
TRA C1
PIND4 LX A F,2
TXI D+1,-1
PINDIR PZE

* PINK CLA KPIN+1,2
PDX ,2
TMI PINK1
PINK2 PXD ,4
ARS 6
ADD #1B17
STO PINDIR
PXD ,2
CAS PINDIR
TRA ,+2
TRA PINK3
CLA IOPP+1,2
CAS PINDIR
TRA ,+2
TRA PINK3
PINK1 TXH F,2,0
PAX ,2
TRA PINK2
PINK3 PIA ,2
PDX G1
TRA G1
PINK4 LX A F,2
TXI D+1,-1

* PHASE 2, COPY INTER INTO TExCH IN ORDER
ORDER NZT ATTACK
TRA E NO ATTACKERS, FLUSH
STZ SIDE ATTACKERS-DEFENDERS FLIP-FLOP
SXD M,1 END TEST FOR INTER TABLE
LXD K,2
STO IEXCH+1,2 SETS SEG OF PIECE
STZ COUNT1 COUNTS TO NUM. ATT. OR DEF.

U CLA =2000817 +INFINITY = VALUE
STO MINVAL
AXT 1,1

P CLA INTER+1,1 SEARCH FOR SMALLEST VALUED PIECE
TMI M-1 PIECE USED
LDI INTER+1,1 SEPARATES ATTACKERS AND DEFENDERS
IIS SIDE ACCORDING TO SIDE
IIS K PIECES AGREEING WITH
LFT 1 SIDE GO TO Q
TRA Q
TXI ++1,1,1

M TXL P,1,** CLOSE INTER SEARCH LOOP
CLA MINVAL
SUB =2000817
TZE NOMORE
LXD COUNT1 INDEX TO IEXCH
IIS INDEX TO INTER

CAND AXT ++,4
CLA INTER+1,4 PICK UP THE BEARER FOR CHAINING
STO IEXCH+1,1 IS THIS PIECE MASKED
PDX 1,2 NO

LBT
TRA SKIP
CAL INTER+2,4 YES, PICK UP COUNT1
STP IEXCH+1,1 MARK MASKED ENTRIES WITH MINUS SIGN
STT IEXCH+1,1 TAG IS INDEX OF MASHER (PREFIX -)

SKIP CLS COUNT1 STORE (-COUNT1)
STO INTER+1,4 INCREMENT COUNT1
SUB =1B20 TOO MANY ATT. OR DEF. ON ONE PIECE
STT COUNT1 SETS THE CONNECTION OF DOUBLE
ADD =8B20 FUNCTION PIECES
TMI LOSE
CLA CHAIN+1,2
PAX +4
PXAX +1
STA IEXCH+1,4 INCREMENTS COUNT
STA CHAIN+1,2
TXI ++1,1,1 MAX SIZE OF IEXCH EXCEEDED
SXD COUNT1
TXL U,1,128

LOSE CALL ERROR,FMT
CALL ERROR,FMT
TSX $LDUMP*4
FMT BCI 5,TABLE SIZE EXCEEDED BY SWAP.
MTH -1,7,-1

# USED IN INTER LOOP
Q PDX 1,2 BEARER IN XR2
CLA KIND+1,2
PDX 4
CLA KVAL+1,4 VALUE OF Bearer
CAS MINVAL
TXI M+1,1
NOP
LDI INTER+1,1
RFT 1
TRA T
R1 STO MINVAL NO, STORE ITS VALUE AND
SXA CAND+1 ITS INTER INDEX.
TXI M+1,1 BACK TO INTER LOOP
T LDI INTER+2,1 HAS MASKER BEEN USED
LNT 400000
TXI M+1,1 NO
TRA R1 YES
* WE HAVE USED ALL ATTACKERS OR DEFENDERS.
NOMORE LXD K+1 ORIGINAL PIECE
CLA COUNT1 NUM ATT. OR DEF.
ZET SIDE
TRA V DEFENDERS
STT IEXCH+1,1 ATTACKERS
CLA =1817
STO SIDE FLIP SIDE
TRA U-1 PICK UP DEFENDERS
V.
ALS 18
STP IEXCH+1,1
TXI W+1,1
*
STORAGE ALLOCATION
COUNT PZE
COUNT1 PZE
SIDE PZE
INDIC PZE
ATTACK PZE
MINVAL PZE
K PZE
COMMON -206 SET TO TOP OF MEMORY
INTER COMMON 20
CHAIN COMMON 32
COMMON 206-20-32+12561 SET TO 29000
R COMMON 1
IBEAR SYN R+12307
LOC SYN R+10971
KIND SYN R+11099
KVAL SYN R+9645
IEXCH SYN R+3374
IOPP SYN R+11277
KPIN SYN R+6372
END
* 
* LABEL
* 
LIST8
SUBROUTINE LTRADE(IW,IB,IND,IARG,IAT)
GIVEN A POSITION, AND UPDATED SWAP TABLES, COMPUTES THE MATERIAL
BALANCE VALUE OF THE POSITION AND SEVERAL STABILITY INDICATORS.
DIMENSION MPVAL(32), N1AT(32)
DIMENSION ITAB(16)
DIMENSION FOO(5000)
DIMENSION LOC(32), NFIRST(22), KPAWNV(8), IEXTD(16), IEXTS(64)
DIMENSION IPIN(32), IOPP(16), KIND(32), MAVAIL(100), KVAL(6)
DIMENSION IHOPE(64), IEXCH(128)
DIMENSION LISP(6000)
COMMON FOO
EQUIVALENCE (FOO(2892), K1), (FOO(1463), KIND), (FOO(2765), MAVAIL)
EQUIVALENCE (FOO(2900), MCOL)
EQUIVALENCE (FOO(2703), IPIN), (FOO(1285), IOPP), (FOO(255), IBEAR)
EQUIVALENCE (FOO(1365), IEXTD), (FOO(1301), IEXTS), (FOO(1527), IQCC)
EQUIVALENCE (FOO(1591), LOC), (FOO(1623), NFIRST), (FOO(3003), KPAWNV)
EQUIVALENCE (FOO(3121), PLY), (FOO(3120), IPE), (FOO(2917), KVAL)
EQUIVALENCE (FOO(3051), MOBW), (FOO(13052), MOBB), (FOO(3123), IHOPE)
EQUIVALENCE (FOO(19188), IEXCH), (FOO(3122), BACK), (FOO(3187), LISP)
EQUIVALENCE (FOO(3053), MATW), (FOO(3054), MATB), (FOO(3119), MLOG)
EQUIVALENCE (FOO(1134), NLOG)
MCOL=MCOL
IARG = 0
IAT=0
IND=0
IW=0
IB=0
IPLY=XSHIFTF(PLY, 11)
DO 5 I=1, 32
MPVAL(I)=0
5 CONTINUE
IF (IAT) 200, 200, 10
20 NAT=XTAGF(IEXCH(I))
IF (NAT) 200, 200, 10
10 NDEF=XPREF(IEXCH(I))
IF (NAT-NDEF) 20, 20, 30
30 K = NAT+NAT+1
GO TO 40
40 K=NDEF+NDEF+2
40 ITAB(I) = I
IATOR=XDECFI(IEXCH(I))
IDEFOR=IATOR+NAT
M=0
J=1-XSHIFTF(XLBITF(I), 1)
IFAT = XDECFI(IEXCH(IATOR))
IDVAL=XGETF(KIND(I), KVAL)-XGETF(KIND(IFAT), KVAL)
IF (IDVAL) 50, 50, 57
50 IF (XLBITF(KIND(IFAT))) 50, 50, 400
400 IAT=IAT+IDVAL*XJ
50 DO 70 L=2, K+2
ITAB(L) = XDECFFXGETF(M+IATOR, IEXCH)
ITAB(L+1) = XDECFFXGETF(M+IDEFOR, IEXCH)
}
70 M=M+1
   ITRA = 0
   DO 80 L=2,K
   JVALUE=XGETF(XGETF(ITAB(L-1),KIND),KVAL)
   ITAB(L-1)=ITRA
   ITRA=ITRA+XSIGNF(JVALUE,J)
   80 J=J
   IF(J)100,100,90
100 IF(K-2)130,130,105
105 ITRA = XMAXOF(ITRA,ITAB(K-1))
   K=K-1
90 IF(K-2)130,130,95
95 ITRA=XMINOF(ITRA,ITAB(K-1))
   K=K-1
   GO TO 100
130 IF(XLBITF(I))160,160,140
140 ITRA = -ITRA
   MPVAL(I) IS THE VALUE OF AN EXCHANGE SQUARE IF THE ATTACKER
   INITIATES THE EXCHANGE WITH HIS LOWEST VALUED PIECE. POSITIVE
   VALUES MEAN THE ATTACKER WINS MATERIAL.
   160 MPVAL(I) = ITRA
   IF(XLBITF(MCOL+1))163,163,161
161 IF(ITRA)165,162,165
   THE MOVER HAS AN EXCHANGE AVAILABLE TO HIM.
162 IND = 1
   GO TO 165
163 IF(ITRA)165,165,164
   THE MOVER HAS A THREATENED PIECE.
164 IARG=4-XMINOF(3,IPLY)
   N1AT(I) IS THE NUMBER OF TIMES THAT PIECE I INITIATES AN EXCHANGE
   SQUARE ATTACK. IF IT IS GREATER THAN 1 WE HAVE A DOUBLE FUNCTION
   PIECE.
165 N1AT(IFAT) = N1AT(IFAT) + 1
200 CONTINUE
   NCVAL = 0
   L1 = 3 - MCOL
   L2 = 30 + L1
   M2 = 30 + MCOL
   DO 300 I = L1, L2, 2
   IF(N1AT(I)-1)300, 300, 240
240 IF(IPLY - 3)250, 255, 255
250 IF(XTAGF(IEXCH(I)))300,300,255
255 DO 280 J9 = MCOL, M2, 2
   NAT = XTAGF(IEXCH(J9))
   IF(NAT)280,280,260
260 IKE = XDECF(XGETF(XDECF(IEXCH(J9)),IEXCH))
   IF(IKE - I)280, 265, 280
265 IF(IPLY - 3)270, 266, 286
266 MPVAL(J9) = 0
   GO TO 310
270 NCVAL = NCVAL + XMAXOF(0, MPVAL(J9))
280 CONTINUE
290 NCVAL=NCVAL + XMINOF(0, MPVAL(I))
   GO TO 310
300 CONTINUE
310 DO 320 I = MCOL, M2, 2
320 NCVAL = XMAXOF(NCVAL, MPVAL(I))
   DO 330 I = 1, 31, 2
      IW = IW + XMAXOF(0, MPVAL(I))
330   IB = IB + XMAXOF(0, MPVAL(I+1))
   IW = -IW
   GO TO (350, 380), MCOL
C +1B OR -1W IS THE AMOUNT AN ATTACKER GAINS ON A BLACK OR WHITE
C EXCHANGE SQUARE, TAKING INTO ACCOUNT THE VALUE OF THE SIDE TO MOVE.
C NCVAL IS THE BUGGER FACTOR WHICH ADJUSTS IB AND IW ACCORDING TO
C THE SIDE TO MOVE.
C NOTE THAT IB+IW IS THE EXPECTED MATERIAL VALUE OF THE POSITION.
350 IW = IW + NCVAL
   GO TO 230
380 IB = IB - NCVAL
230 IAT=XSIGNF1XONEF(IAT),IAT)
RETURN
END
* LABEL
* LIST8
SUBROUTINE PINS
COMMON FOO
DIMENSION IBEAR(64,16), IOCC(64)
DIMENSION FOO(5000)
DIMENSION LOC(32), NFIRST(22), KPAWNV(8), IEXTD(16), IEXTS(64)
DIMENSION IPIN(32), IOPP(16), KIND(32), MAVAIL(100), KVAL(6)
DIMENSION KPIN(32)
EQUIVALENCE (FOO(2992), K1), (FOO(1463), KIND), (FOO(2765), MAVAIL)
EQUIVALENCE (FOO(2900), MCOL)
EQUIVALENCE (FOO(2703), IPIN), (FOO(1285), IOPP), (FOO(255), IBEAR)
EQUIVALENCE (FOO(1365), IEXTD), (FOO(1301), IEXTS), (FOO(1527), IOCC)
EQUIVALENCE (FOO(1591), LOC), (FOO(1623), NFIRST), (FOO(3003), KPAWNV)
EQUIVALENCE (FOO(3121), PLY), (FOO(3120), IPE), (FOO(2917), KVAL)
EQUIVALENCE (FOO(6187), KPIN)
DO 40 J = 1, 32
  40 KPIN(J) = 0
  DO 20 J = 1, 2
    KRAP = 1
  GO TO 7
  20 CONTINUE
  DO 30 J = 31, 32
    KRAP = 2
  GO TO 7
  30 CONTINUE
RETURN
  7 KLOC = LOC(J)
  DO 1 I = 1, 8
    JPIN = LOOK (KLOC, IOPP(I))
    IF (JPIN) 1, 1, 3
  1 IF (XLBITF (IOCC(JPIN)+J)) 1, 4, 1
  3 IF (IFO0 = IBEAR(JPIN,1))
    IF (IFO0) 1, 1, 5
  4 IF (XORF (XLBITF (IFO0+J), XLBITF (KIND (IFO0)+1))) 6, 1, 6
  5 IF (JPIN = IOCC(JPIN))
    GO TO (15, 16), KRAP
  6 KPIN(JPIN) = 1
  GO TO 1
  15 KPIN(JPIN) = -(KPIN(JPIN) + XSHIFTF (1, -18))
CONTINUE
  GO TO (20, 30), KRAP
END
* LABEL
* FAP
*PAWN STRUCTURE FOR CHESS, MARCH 2, 1962
COUNT 150
ENTRY JPAWNS
JPAWNS SXA XR1,1
SX A XR2,2
SX D XR4,4
XR4 SYN JPAWNS-2
* EMTM 
CLA NOP
STO COLOR
STA COLOR1
AXT NP3+1,4
SX A NP3,4
CLA TABLE
LDQ TABLE-8
AXT TABLE-1
AXT COLOR2+1,4
LOOP SXA COLOR2,4
SX A SET1,1
SX A SET2,1
STD TABLE-6
SLQ TABLE-22
STZ PAWNV
AXT 0+1
FILEL STZ ADJAC
STZ PROTEC
STZ NPAWNS
STZ PAST
CLS =2B17
STO OTHER
COLOR1 AXT **,2
RANKL TXL NP1,1,0
CLA IOCC+1,3
PDX *,4
SET1 XEC **,4
NP1 TXH NP2,1,6
CLA IOCC-1,3
PDX *,4
SET2 XEC **,4
NP2 CLA IOCC,3
PDX *,4
TXL NP3,4,6
TXH NP3,4,22
PX A *,4
COLOR HTR *
LBT TRA POSS
CLA NPAWNS
ADD =1B17
STO NPAWNS
CLA ADJAC
STO PROTEC
FOR 7094
SET UP FOR WHITE LOOP
INITIALIZE EXECUTE FOR WHITE
MAJOR LOOP, EXEC. FOR BLACK AND WHITE
FILE IN XR1
ADJACENT PAWN INDICATOR
ANOTHER PAWN PROTECTING INDIC.
INDICATED A PASSED PAWN
RANK IN XR2, O FOR WH., 56 FOR BLK.
ONLY IF A FILE EXISTS TO LEFT
THIS IS AN OPPOSITION PAWN
THIS SAVES THE ABSOLUTE RANK

*+1 FOR WHITE, BLACK FOR BLACK

EVALUATOR

OPEN FILE

NOT DOUBLED PAWN

DOUBLED PAWN

PAST PAWN

THIS HAS BEEN IN AC ALL THIS TIME

*+1 FOR WHITE, DONE FOR BLACK

RE-INITIALIZE FOR BLACK
<table>
<thead>
<tr>
<th>TXI</th>
<th>+1,2,56</th>
</tr>
</thead>
<tbody>
<tr>
<td>SXA</td>
<td>LRANK,2</td>
</tr>
<tr>
<td>TRA</td>
<td>EVL</td>
</tr>
<tr>
<td>DONE</td>
<td>CLA</td>
</tr>
<tr>
<td>SUB</td>
<td>TPAWNV</td>
</tr>
<tr>
<td>*</td>
<td>LMTM</td>
</tr>
<tr>
<td>XR1</td>
<td>**,1</td>
</tr>
<tr>
<td>XR2</td>
<td>**,2</td>
</tr>
<tr>
<td>LXD</td>
<td>XR4,4</td>
</tr>
<tr>
<td>TRA</td>
<td>2,4</td>
</tr>
<tr>
<td>DUP</td>
<td>1,10</td>
</tr>
<tr>
<td>PDX</td>
<td>0</td>
</tr>
<tr>
<td>STL</td>
<td>PAST</td>
</tr>
<tr>
<td>DUP</td>
<td>2,8</td>
</tr>
<tr>
<td>STL</td>
<td>ADJAC</td>
</tr>
<tr>
<td>STL</td>
<td>PAST</td>
</tr>
<tr>
<td>DUP</td>
<td>1,7</td>
</tr>
<tr>
<td>PDX</td>
<td>0</td>
</tr>
<tr>
<td>TABLE</td>
<td>SYN</td>
</tr>
<tr>
<td>PAWNV</td>
<td>PZE</td>
</tr>
<tr>
<td>ADJAC</td>
<td>PZE</td>
</tr>
<tr>
<td>PROTEC</td>
<td>PZE</td>
</tr>
<tr>
<td>NPAWNS</td>
<td>PZE</td>
</tr>
<tr>
<td>PAST</td>
<td>PZE</td>
</tr>
<tr>
<td>OTHER</td>
<td>PZE</td>
</tr>
<tr>
<td>TPAWNV</td>
<td>PZE</td>
</tr>
<tr>
<td>LRANK</td>
<td>PZE</td>
</tr>
</tbody>
</table>

### VALUE TABLES
- **DEC**
  - TXI 7B17,7B17,8B17,8B17,8B17,7B17
  - SXA 7B17
  - TRA -5B17,-1B17,-1B17,-1B17,-1B17,-1B17
  - LXD -5B17
  - XR1 0,-5B17,-5B17,-5B17,-5B17,-5B17,-5B17
  - XR2 -4B17,-4B17,-4B17,-4B17,-4B17,-4B17,-4B17
  - LMTM -4B17
  - ADJAC 0,0,0,0,0,0,0,0
  - NPAWNS 12561
  - PAWNV 1
  - TPAWNV SYN R+11035
* LABEL
* LIST8.
FUNCTION ICENTR(1123)
C COMPUTES THE CENTER CONTROL FUNCTION. LCENSQ IS A TABLE OF CENTER
C SQUARES. KCNVAL IS A TABLE OF RELATIVE WEIGHTS OF THOSE SQUARES.
COMMON FOO
DIMENSION KPIN(32)
DIMENSION IBEAR(64,16), LOC(32), KIND(32), FOO(5000)
DIMENSION LCENSQ(16), KCNVAL(16)
EQUIVALENCE (FOO(9317), NMOVES)
EQUIVALENCE (FOO(6187), KPIN)
EQUIVALENCE(FOO(2892),K1),(FOO(1463),KIND),(FOO(2765),MAVAIL)
EQUIVALENCE(FOO(2703),IPIN),(FOO(1285),IOPP),(FOO(255),IBEAR)
EQUIVALENCE(FOO(1591),LOC),(FOO(1623),NFIRST),(FOO(3003),KPAKNV)
EQUIVALENCE (FOO(3011), LCENSQ), (FOO(3027), KCNVAL)
ICENTR = 0
IF (NMOVES = 30) 102, 101, 101
102 I123 = I123
DO 100 I = 1, 16
   K = LCENSQ(I)
   DO 90 J = 1, 16
      IF (IBEAR(K,J)) 90, 90, 10
   10 KP = IBEAR(K,J)
      IF (KPIN(KP)) 90, 13, 90
   90 CONTINUE
  13 IF (KIND(KP) = 6) 15, 110, 15
  110 IF (XLBITF(KP)) 130, 130, 120
  120 ICENTR = ICENTR + KCNVAL(I)/3
     GO TO 40
  130 ICENTR = ICENTR - XGETF(17-I, KCNVAL)/3
     GO TO 40
  15 IF (XLBITF(KP)) 30, 30, 20
  20 ICENTR = ICENTR + KCNVAL(I)
     GO TO 40
  30 ICENTR = ICENTR - XGETF(17-I, KCNVAL)
  40 LOCKP = LOC(KP)
     IF (IBEAR(LOCKP, J)) 90, 90, 50
  50 KPP = IBEAR(LOCKP, J)
     IF (XLBITF(KPP+KP) + XLBITF(KIND(KP))) 90, 60, 90
  60 KP = KPP
     GO TO 15
  90 CONTINUE
100 CONTINUE
   ICENTR = (ICENTR * XM1NOF(10, 30 - NMOVES))/10
101 RETURN
END
* LABEL
* LIST8
FUNCTION IDVLOP(1123)
C COMPUTES THE STATIC EVALUATION FUNCTION FOR DEVELOPMENT
DIMENSION FOO(6000), LOC(32), NFIRST(22), KPAWNV(8), IEXTD(16)
DIMENSION IEXTS(64), IOCC(64)
COMMON FOO
EQUIVALENCE (FOO(9317), NMOVES)
EQUIVALENCE (FOO(2892), K1), (FOO(1463), KIND), (FOO(2765), MAVAIL)
EQUIVALENCE (FOO(2900), MCOL)
EQUIVALENCE (FOO(2703), IPIN), (FOO(1285), IPOP), (FOO(255), IBEAR)
EQUIVALENCE (FOO(1365), IEXTD), (FOO(1301), IEXTS), (FOO(1527), IOCC)
EQUIVALENCE (FOO(1591), LOC), (FOO(1625), NFIRST), (FOO(3003), KPAWNV)
XBLTCHF(J) = XORF(XGETF(J+ICOLOR, LOC) + XTRANF(XGETF(J+ICOLOR, LOC))
IDVLOP = 0
1123 = 1123
ICOLOR = 0
IF (NMOVES - 15) 69, 100, 100
69 IBARF = IPESS
IPESS = 0
DO 1 I = 7, 21, 2
1 IPESS = IPESS + XN01F(XGETF(I+ICOLOR, NFIRST))
DO 2 I = 13, 15, 2
IPESS = IPESS + XGEF(XBLTCHF(I) + KPAWNV)
IF (XGETF(I+ICOLOR, NFIRST) + XN01F(XGETF(I+ICOLOR, LOC))) 22, 22, 2
22 IDIR = XSHIFTF(ICOLOR+1,1)
NSQ = XN0VF(IEXTD(IDIR) + XGETF(XGETF(ICOLOR+1, LOC), IEXTS))
IF (IOCC(NSQ)+XGETF(XN0VF(IEXTD(IDIR)+1EXTS(NSQ))+IOCC(23, 2, 23
23 IPESS = IPESS - 5
2 CONTINUE
IPESS = IPESS + 5*XN01F(XBLTCHF(I1)-4)
IF (ICOLOR) 40, 40, 50
40 KJ1 = 2
KJ2 = 7
KQ2 = 12
GO TO 60
50 KJ1 = 58
KJ2 = 63
KQ2 = 52
60 IF (IOCC(KJ1)-23-ICOLOR) 62, 61, 62
62 IPESS = IPESS + 4
61 IF (IOCC(KJ2)-25-1COLOR) 64, 63, 64
64 IPESS = IPESS + 4
63 IF (IOCC(KJ1+1)-27-1COLOR) 66, 65, 66
66 IPESS = IPESS + 3
IF (IOCC(KQ2) -23-ICOLOR) 65, 166, 65
166 IPESS = IPESS - 10
65 IF (IOCC(KJ2-1)-29-1COLOR) 68, 67, 68
68 IPESS = IPESS + 3
67 IF (IOCC(KQ2+1) -25-1COLOR) 67, 168, 67
168 IPESS = IPESS - 10
67 IF (IOCC(KJ1+2)-31-ICOLOR) 71, 70, 71
70 IPESS = IPESS + 7
GO TO 75
71 IPESS=4*XORF(LUC(ICOLOR+31)*XNOTF(XRANGEF(XBLTCHF(31),1,3)))+IPESS
75 ICOLOR = ICOLOR + 1
GO TO (69,71) +ICOLOR
711 IDVLOP = IBARF - IPESS
100 RETURN
END
* LABEL
* LISTB

SUBROUTINE REPLY
DIMENSION MPVAL(100)
DIMENSION FO0(3000)
DIMENSION LOC(32), NF1ST(22), & PAWNNV(8), IEXTD(16), IEXTS(64)
DIMENSION IPIN(32), IOPP(16), KINAL(32), MAVAIL(100), KVAL(6)
DIMENSION IHOP(64), IEXCH(128)
DIMENSION LISP(6000)
DIMENSION KPLY(20)
COMMON FO0
EQUIVALENCE (FO0(6219), IWHTM), (FO0(6220), IBLKM)
EQUIVALENCE (FO0(2892), K1), (FO0(1463), KIND), (FO0(2765), MAVAIL)
EQUIVALENCE (FO0(2900), MCOL)
EQUIVALENCE (FO0(2703), IPIN), (FO0(1285), IOPP), (FO0(255), IBEAK)
EQUIVALENCE (FO0(1965), IEXTD), (FO0(1301), IEXTS), (FO0(1527), IOCC)
EQUIVALENCE (FO0(1991), LOC), (FO0(1623), NF1ST), (FO0(3003), PAWNNV)
EQUIVALENCE (FO0(3121), PLY), (FO0(3120), IPE), (FO0(2917), KVAL)
EQUIVALENCE (FO0(3051), MUBW), (FO0(3952), MUBB), (FO0(3123), IHOP)
EQUIVALENCE (FO0(9188), IEXCH), (FO0(3122), BACK), (FO0(3187), LISP)
EQUIVALENCE (FO0(3093), MATN), (FO0(3054), MATB), (FO0(3119), MLUG)
EQUIVALENCE (FO0(134), MLUG)
EQUIVALENCE (FO0(2577), ICHECK)
EQUIVALENCE (KPLY, FO0(9167))

10 IF(K1) 31, 31, 20
20 J = -MCOL-MCOL+3
   IPLY = XSHIFTF(PLY, 11)
   ISTAB = 0
   ID = IDVLOP(1)
21 IPE = XMINOF(KPLY(1PLY), K1)
   IF (IPE) 666, 666, 99
99 IF(PLY-2) 30, 30, 200
30 ISTAB = 1
60 DO 80 M = 1, K1
   NP = XGETF(XMV1F(MAVAIL(M)), IOCC)
   MVR = XMV3F(MAVAIL(M))
   IF(KIND(MVR) = 5) 32, 32, 35
32 IF(XABSF(LOC1MVR) = XMV1F(MAVAIL(M))) 2, 35, 33, 33
33 KS = 28
   GO TO 37
35 KS = 0
37 CALL UPDATE(MAVAIL(M))
869 CALL PINS
   CALL SWAP
   CALL LTRADE(IW, ID, IND, IARG, IAT)
   IDT = IDVLOP(1)
   IF(IDT+J) 62, 62, 60
60 IF(XMAXOF((IDT-ID)+J-20)+XM1UF(XRANGEF(MVR+13, 16))) 62, 62, 61
61 ISTAB = 1
   GO TO 629
62 IAT = 0
629 IF(NP) 50, 50, 40
40 MAVAL = XGETF(KIND(NP), KVAL)
   IKAPT = 6
GO TO 70
50 MVAL = 0
IKAPT = 0
70 IF(J) 555, 556, 556
555 NVAL = MVAL * IWHIM
GO TO 77
556 NVAL = MVAL * IBLKM
77 MPVAL = NVAL + (IWHIM * I + IBLKM * IB + XSHIFTF (IDT, 2) + ICENTR
1(I) + XSHIFTF (IAT, 4) + 3*PAWNS (I)) * J + KS + 24/K1#2 + IKAPT +
2IARG
80 CALL REVERT
 IF (ISTAB) 250, 250, 85
250 IF(XLIBIF(IPLY)) 85, 85, 31
85 DO 120 I=1, IPE
LM=IPE-I+1
MVAL=-5000
DO 110 M=1, K1
IF(MPVAL(M)-MVAL) 110, 110, 90
90 MVAL=MPVAL(M)
K=M
110 CONTINUE
IHOPE(LM)=MAVAIL(K)
120 MPVAL(K)=-5000
GO TO 900
200 CALL SWAP
 CALL LTRADE(IW, IB, IAD, IARG, IAT)
 IF(IPE-I=2) 10, 10, 20
 IF(IPLY-1)=30, 30, 220
 IF(I=1)+XABS(F(IARG)) 100, 100, 222
 IF(IPLY-1)=30, 30, 244
 IF(I=1)+XABS(F(IAT)) 100, 100, 224
 IF(IPLY-1)=30, 30, 244
900 IF(IPLY-2)=905, 905, 950
1000 FORMAT(6HO, IPLY=10, 4X, 14A6)
905 DO 910 M=1, IPE
910 CALL JUNPAK(IHOPE(M), MPVAL(M), MPVAL(M+8))
 WRITE OUTPUT TAPE 100, 1000, IPLY, ((MPVAL(M), MPVAL(M+8)) * M=1, IPE)
 GO TO 950
666 WRITE OUTPUT TAPE 100, 1000, IPLY
31 IPE=0
950 RETURN
END
* LABEL
* LISTB
SUBROUTINE EVAL
DIMENSION FO0(5000)
DIMENSION LOCl32), NFIPST(22), KPAWNW(8), IEZTD(16), IEXTS(64)
DIMENSION IPIN(32), IOPP(16), KIND(32), MAVAIL(100), KVAL(5)
DIMENSION IHQPE(64), IECH(128)
DIMENSION LISP(6000)
DIMENSION NTYPE(20)
COMMON FO0
EQUIVALENCE(FO0(2892), K1), (FO0(1469), KIND), (FO0(2765), MAVAIL)
EQUIVALENCE(FO0(6219), IWHMT), (FO0(6220), IBLKk)
EQUIVALENCE(FO0(2800), MCOL)
EQUIVALENCE(FO0(2703), IPIN), (FO0(1285), IOPP), (FO0(265), IBEAR)
EQUIVALENCE(FO0(1369), IEZTD), (FO0(1301), IEXTS), (FO0(1527), ICCC)
EQUIVALENCE(FO0(1591), LOC), (FO0(1223), NFIRST), (FO0(3003), KPAWNW)
EQUIVALENCE(FO0(3121), PLY), (FO0(3120), IPE), (FO0(2917), KVAL)
EQUIVALENCE(FO0(3051), MO4W), (FO0(3052), 808B), (FO0(3123), IHQPE)
EQUIVALENCE(FO0(19188), IECH), (FO0(3122), BACK), (FO0(3187), LISP)
EQUIVALENCE(FO0(13053), MATW), (FO0(3054), MATB), (FO0(3119), MLOG)
EQUIVALENCE(FO0(134), NLOG)
EQUIVALENCE(1, A)
EQUIVALENCE(FO0(2913), NSPEC), (FO0(2649), NTYPE)
10 I=KXSIGNF(10000, MCOL+MCOL-2)
GO TO 30
15 KS=0
CALL PINS
60 CALL SWAP
CALL LTRADE(IWHMT, I), IARG, IAT)
50 IF(NSPEC)20, 20, 7
7 DO 1 1=1, NSPEC
1 IF(NTYPE(I)+1)8, 1, 1
8 IF(XLBITF(XMV3F(NTYPE(I)))), 4, 4, <
4 KS=KS+28
GO TO 1
5 KS=KS+26
1 CONTINUE
20 I=IWHMT*(MATW+I)+IBLKk*(1D-MATB)+3*JPAWNW(I)+XSHIFTF(IDVLOP(1),
12)+1CENTR(1)
8 30 A=A
RETURN
END
* LABEL
* LIST8
C THE LONG AWAITED STRATEGY PROGRAM. MAY 1, 1962
SUBROUTINE STRTG
COMMON FOO
EQUIVALENCE (FOO(3053),MATW), (FOO(3054),MATB)
EQUIVALENCE (FOO(6219),IWHIM), (FOO(6220),IBLKM)
CALL PINS
CALL SWAP
CALL LTRADE (IW, IB, IND, IAKG, IAT)
ITEM = IW + IB + MATW = MATB
IWHIM = 60
IBLKM = 60
IF (XABSF (ITEM) - 4) 1, 2, 2
2 IWHIM = IWHIM -XSIGNF (10, ITEM)
IBLKM = IBLKM +XSIGNF (10, ITEM)
1 RETURN
END
* LABEL
* FAP
COUNT 31
* ALIAS UPDATE, REVERT, CCOL, SETUP
ENTRY UPDATE
ENTRY REVERT
ENTRY CCOL
ENTRY SETUP
UPDATE SXD UPDATE-2,4
CLA* 1,4
TZE ZERO
STO MIN
CALL UPREV, MIN, ONE
RTN LXD UPDATE-2,4
TRA 2,4
ZERO CALL ERROR, FMT
TRA RTN
FMT BCI 5, UPDATE CALLED WITH ZERO ARG.
MTH -1,7,-1
REVERT SXD UPDATE-2,4
CALL UPREV, ZRO, TWO
RTN1 LXD UPDATE-2,4
TRA 1,4
CCOL SXD UPDATE-2,4
CALL UPREV, ZRO, FOR
TRA RTN1
SETUP SXD UPDATE-2,4
CALL UPREV, MIN3, THR
TRA RTN1
ZRO PZE
ONE PZE,,1
TWO PZE,,2
THR PZE,,3
FOR PZE,,4
MIN3 MZE,,3
MIN PZE
END
* LABEL
* LIST8
C UPREV CHESS SUBROUTINE, 2/26/62, MINOR REVISION
SUBROUTINE UPREV(MIN,M6)
C DIMENSION AND EQUIVALENCE STATEMENTS
DIMENSION IOCC(64), LOC(32), NFIRST(22), NUMB(50),
INTYPE(50), IBEG(32), IEND(32), MOVEP(150), ICAPT(150),
2MOVEFR(150), MOVEMP(150), IBEARK(1024), IBEARK(64,16),
3KIND(32), MSVN(16), IPDIR(32), IEXTD(10), IEXTS(64),
4M64M16(16), NMOV(6), IOPP(16)
DIMENSION JPAWN(8)
DIMENSION MSTO(32)
DIMENSION MAVAIL(100), ITCH(2), ITCID(2), IPIN(32)
DIMENSION NEP(10), MELP(110), MEP2(10)
DIMENSION JROM(4)
DIMENSION LOGG(101)
DIMENSION NZ2Z(120)
DIMENSION KVAL(6), KEFORE(64), KWORDH(64)
C COMMON STATEMENTS
COMMON IPDIR, IOPP, IEXTD, IEXTS, JPAWN, M64M1, MSVN, NMOV, MSTO, JROM,
1IBEARK, JBEARK, KIND, IEND, IBEG, IOCC, LOC, NFIRST, MOVE, IEVNS, MOVEP,
2MOVEFR, ICAPT, NUMB, INTYPE, ITCH, ITCID, IPIN, NEP, MELP, MEP2, LOGG, NZ2Z,
3NZ2Z, NUMTES, MAVAIL, IZ, IY, IX, IU, IT, ISPEC, IR, IQ, JROM, IOPPE, INTER,
4IDIR, ICHECK, IA, IMA, IA, JA, JB, JC, JDIR, JU, JEF, JIN, JU, JROUK, JX, KD,
5K5, LZ, L4, MARET, ICAPT, MCOL, MIN, MOVEMP, MOVENO, MOVEMP, MOVEO, MG, M,
6MVRE, N1, N2, NEWSY, N, NSPEC, NUMT, NTIME, TPRINT, KIN, KVAL, KFORE, KWORDH, MUBW,
7MOBB, MATM, MATB
EQUIVALENCE (IEVNS, IBEG(33)), (NZ2Z, LOGG(101)), (NUMTES, NZ2Z(120)),
1(IBEARK, JBEARK)
DIMENSION NUMBER(64)
COMMON NUMBER
COMMON MLOG
C
C MAIN PROGRAM
C
MCOL = MCOL
GO TO (120, 150, 700, 200), M6
C CHANGE COLOR OR SIDE TO MOVE
200 MCOL = 3 - MCOL
MIN = - MCOL
MOVENO = MOVENO + 1
MOVEP(MOVEMP) = -1
GO TO 700
C MIN IS THE MOVE MADE
120 MOVEO = XMV1F(MIN)
MOVDIR = XMV2F(MIN)
MOVER = XMV3F(MIN)
C C SET UP VARIABLES FOR UPDATE
C 130 MQ = LOC(MOVER)
KD = KIND(MOVER)
MOVENO = MOVENO + 1
MCOL = 1 + XBLITF(MOVER)

BRANCH ON PIECE KIND
GO TO (400,131,134,134,460,134),K0

THIS MAY BE THE FIRST MOVE OF A ROOK
131 IF(NFIRST(MOVER))133,133,134
133 NFIRST(MOVER)=1
NSPEC=NSPEC+1
NUMB(NSPEC)=MOVENO
NTYPE(NSPEC)=1

IS THE MOVE A CAPTURE
134 IF(IOCC(MOVETO))137,137,136
C
CAPTURE
136 ICAPT(MOVENO)=IOCC(MOVETO)
CALL PUTC(IOCC(MOVETO),0)
137 CALL PUTC(MOVER,MOVETO)
139 MOVEFR(MOVENO)=MQ
141 MOVEP(MOVENO)=MOVER

1. CHECKS AND PINS
2. LIST LEGAL MOVES OF KINGS DIRECTLY IN MAVAIL TABLE
3. LIST MOVES OF THE OTHER PIECES IN THE MAVAIL TABLE

INITIALIZE.
700 ICHECK = 0
KLOC = LOC(MCOL)
DO 701 JA=1,32
701 IPIN(JA) = 0
DO 702 I=1,2
ITCH(I)=0
702 ITCHD(I)=0
IR = IEXTS(KLOC)
K1 = 0
M = IBEG(MCOL) - 1
END OF INITIALIZATION

IS THE KING IN CHECK. LIST PINS.
DO 715 K=1,16
C
IS THE KING SUBJECT TO CAPTURE BY THE OTHER SIDE
IF (IBEAR(KLOC,K)) 721,721,718
HAS THE BEARER THE SAME COLOR AS THE KING.
718 IF (XBLITF(IBEAR(KLOC,K)+MCOL)) 730,716,750
C
THE KING IS IN CHECK.
750 ICHECK = ICHECK + 1
ITCH(ICHECK) = IBEAR(KLOC,K)
ITCHD(ICHECK) = IOPP(K)
IF (ICHECK = 2) 400,715,731
C
KNIGHTS CANNOT PIN
721 IF(K = 8) 722,722,715
722 IQ = XGETF(IOPP(K),IEXTD)
IZ = IR

LOOK FOR OCCUPIED SQUARE ALONG LINE FROM KING

IZ = IZ + IQ
NEWSQ = XMOMF(IZ)
IF (NEWSQ - 64) 719, 719, 719
IF (IOMC(NEWSQ)) 728, 728, 727

AN OCCUPIED SQUARE IS FOUND
NEWSQ = XGETF(IBEAR(KLOC,K),LOC)
FIND WHAT IF ANYTHING BEARS FROM OPPOSITE DIRECTION
IU = IBEAR(NEWSQ,K)
IF (IU) 715, 715, 726

IF BEARER IS A LONG RANGE PIECE OF OPPOSITE COLOR WE GET A PIN.
IF(I-XLBITF(IU+MCOL)+XLBITF(KIND(IU))) 715, 732, 715

LIST A PIN
IT = IOMC(NEWSQ)
IPIN(IT) = K
715 CONTINUE

PUT MOVES OF KING'S IN MAVAIL TABLE
FIRST NON-CASTLING MOVES
DO 705 IDIR=1,6
IF(XGETF(M+IDIR,MOVE)) 705, 705, 706
706 NEWSQ = XMOMF(XGETF(M+IDIR,MOVE))

THE KING CANNOT MOVE ALONG THE LINE OF CHECK, UNLESS THE CHECKER IS A PAWN.

IF (ICHECK) 753, 708, 753
DO 751 JA=1,ICHECK
IF (ITCH(JA)=IOPP(IDIR)) 751, 752, 751
IF (XGETF(ITCH(JA),KIND)-1) 705, 751, 705
751 CONTINUE

DO 712 K=1,16
IF (IBEAR(NEWSQ,K)) 712, 712, 713
IF (XLBITF(IBEAR(NEWSQ,K)+MCOL)) 705, 712, 705
712 CONTINUE

K1 = K1+1
MAVAIL(K1) = XGETF(M+IDIR,MOV E)
705 CONTINUE

ARE THERE CASTLING MOVES
NOT IF KING IS IN CHECK OR HAS MOVED
IF (ICHECK+NFIRST(MCOL)) 800, 736, 800

FOR EACH ROOK
DO 737 IDIR=1,3,2

DOES A ROOK WHICH HAS NEVER MOVED BEAR ON THE KING
IF (IBEAR(KLOC,IDIR)) 739, 739, 739

JRook = IBEAR(KLOC, IDIR)
33
IF(KIND(JROOK)-2+NFIRST(JROOK))737,738,737
C
ARE THE INTERMEDIATE SQUARES COVERED BY THE FOE
738 J8=IDIR-2
JD=KLUC
C
FOR EACH SQUARE THE KING MOVES OVER
DO 741 JC=1,2
JD=JD+JB
C
FOR EACH DIRECTION FROM THE INTERMEDIATE SQUARE
DO 742 JDIR=2,16
JE=IBEAR(JD,JDIR)
IF (JE) 742,742,744
744 IF(XLBITF(MCOL + JE))737,742,737
742 CONTINUE
741 CONTINUE
C
CASTLING OK
K1=K1+1
MAVAIL(K1)=JD+XGETF(10PP(IDIR),M64M1)+MSTO(MCOL)
737 CONTINUE
C
C
MOVES OF OTHER PIECES IN MAVAIL TABLE, OMITTING KINGS
800 K=MCOL+2
IF(ICHECK-1)802,824,825
802 DO 803 I=K,32,2
IF(ILOC(I))804,803,804
804 M=IBEG(I)
C
IF A PAWN HAS MOVED, IT CANNOT ADVANCE TWO SQUARES.
IF (XMAXOF(KIND(I))-1,1-NFIRST(I)) 815,816,816
816 N=IEND(I)-1
GO TO 817
815 N=IEND(I)
C
IS PIECE PINNED
817 IF (IPIN(I)) 805,806,805
C
NO PIN
806 DO 807 J=M,N
IF(MOVE(J))807,807,808
808 K1=K1+1
MAVAIL(K1) = MOVE(J)
807 CONTINUE
GO TO 803
C
PINNED
805 IDIR = IPIN(I)
IOPPD = IOPP(IDIR)
809 DO 812 J=M,N
IF(MOVE(J))812,812,813
813 IF(XMINOF(XABSFXMV2F(MOVE(J))-IDIR),XABSFXMV2F(MOVE(J))-IOPPD))
1812,814,812
814 K1=K1+1
MAVAIL(K1) = MOVE(J)
812 CONTINUE
803 CONTINUE
C ADJOIN EN PASSANT MOVES IF ANY
IF(NUMEP)860,143,860
860 IF(NEP(NUMEP)=MOVENO)143,850,143
850 JJ=1
859 GO TO (851,852,143),JJ
851 J1 = MEPl(NUMEP)
GO TO 853
852 J1 = MEP2(NUMEP)
IF(J1)>55,143,853
C IS THE EN PASSANT MOVE PREVENTED BY A PIN
853 IF(XGETF(XMV3F(J1),1PIN))854,855,854
PINNED, WHAT ABOUT THE DIRECTION?
854 IF(XMINOF(XABSF(XGETF(XMV3F(J1),1PIN)-XMV2F(J1)),XABSF(XGETF(XGETF
1(XMV3F(J1),1PIN),1OPP)-XMV2F(J1))))>6,855,856
C NO PIN ON MOVE, WILL REMOVAL OF CAPTURED PAWN PUT US
C IN CHECK.
855 IF (XRANKF(KLOC)-XRANKF(XGETF(XMV3F(J1),1LOC))) 856,857,858
KING ON SAME RANK AS Pawns. REFERENCES TO PUTCH ARE NEEDED
C TO REMOVE PAWNS FROM POSSIBLE LINE OF ACTION.
857 J1OCC=XMV3F(J1)
J1LOC=LOC(J1OCC)
J2=XMV2F(J1)
J3=XMVF(IEXTS(J1LOC)+XGETF(13-J2-J2),IEXTU))
J3OCC=ILOC(J3)
CALL PUTCH(J1OCC,U)
CALL PUTCH(J3OCC,U)
DO 864 K=1,3,2
IF (XBEAR(KLOC,K)) 864,864,861
861 IF (XLBSTF(1XBEAR(KLOC,K)+MCOL)) 854,864,862
864 CONTINUE
J4=0
GO TO 863
863 J4=1
CALL PUTCH(J1OCC,J1LOC)
CALL PUTCH(J3OCC,J3)
IF (J4) 859,859,856
C PUT EN PASSANT MOVE IN MAVAIL.
858 K1 = K1 + 1
MAVAIL(K1) = J1
856 JJ=JJ+1
GO TO 859
C C SINGLE CHECK LEGAL KING MOVES HAVE
C ALREADY BEEN FOUND. LOOK FOR INTERPOSITIONS OR
C CAPTURE OF CHECKER ALONG CHECK LINE.
824 M=XGETF(1ITCHD(J1),IEXTU)
N = IEXTS(KLOC)
C C LOOP WHICH LOOKS ALONG CHECK LINE
C LOOK AT SQUARES IN DIRECTION OF CHECK
834 N = N+M
836 N1 = XMVF(N)
C LOOK AT BEARERS ON SQUARE
DO 826 IDIR = 1,16
IF (XABSF (IBEAR(N1,IDIR)) - 2) 826, 826, 827
827 IF (XLBITF (IBEAR(N1,IDIR) + MCOL)) 826, 826, 826
C SAME COLOR, MAY INTERPOSE OR CAPTURE CHESSER
C IS IT PINNED
828 INTER = IBEAR(N1,IDIR)
IF (IPIN (INTER)) 826, 829, 826
C NOT PINNED
C CONSTRUCT MOVE. THERE ARE PAWN COMPLICATIONS.
829 IF (KIND (INTER) = 1) 830, 831, 830
C A PAWN
831 IF (IDIR = 4) 832, 832, 833
C VERTICAL DIRECTION. OK IF SQUARE IS EMPTY.
832 IF (ILOCAL(N1)) 829, 830, 826
C IS THERE AN INTERVENING OCCUPIED SQUARE
833 IF (XGETF (XMOVF (XGETF (LOCAL (INTER)), IEXTS) + 1EXTD (IDIR)), ILOCAL))
1 826, 830, 826
C DIAGONAL DIRECTION. OK IF THE SQUARE IS OCCUPIED.
834 IF (ILOCAL(N1)) 830, 826, 820
C CONSTRUCT MOVE.
835 K1 = K1 + 1
MAVAIL(K1) = MAVAIL (INTER) + MV + M1(IDIR) + N1
826 CONTINUE
IF (ILOCAL(N1)) 843, 834, 843
C IF THE CHECKER IS A PAWN ANY EN PASSANT MOVES ARE OK
C UNLESS THE MOVER IS PINNED.
843 IF (XGETF (XMOVF (KIND (INTER)) = 1) 825, 840, 825
840 IF (NEP (NUMEP) = MOVENO) 825, 844, 825
844 IF (XGETF (XMV3F (NEP (NUMEP)), IPIN)) 845, 841, 845
841 K1 = K1 + 1
MAVAIL(K1) = NEP (NUMEP)
845 IF (MEP (NUMEP)) 846, 825, 846
846 IF (XGETF (XMV3F (MEP (NUMEP)), IPIN)) 825, 842, 825
842 K1 = K1 + 1
MAVAIL(K1) = MEP (NUMEP)
C IF THERE ARE NO LEGAL MOVES IT IS MATE
825 IF (K1 = 143) 835, 143
835 K1 = -1
143 NLOG = NLOG + 1
MLOG = MLOG + 1
LOGG(NLOG) = MIN
IF (NLOG = 100) 144, 145, 145
145 WRITE TAPE 7, LOGG
NLOG = 0
144 RETURN
C IS MOVE AN ENPASSANT CAPTURE? DOES IT ALLOW ONE, IS IT A PROMOTION
400 IF (NFIRST (MOVETO) = 1) 402, 402, 412
402 NFIRST (MOVETO) = 1
NSPEC = NSPEC + 1
NUMB(NSPEC) = MOVENO
NTYPE(NSPEC) = 1
IF (XTRANKF (MOVETO, MOVER) = 4) 134, 403, 134
C 2ND RANK TO 4TH LOOK TO SIDES
DO 405 J=1,2
IX=XMVFIEXTS(MOVETO' + IEXTU(4*J-1))
IF(IX-64)404,404,405
404 IY=IOCC(IY)
IF(IY)405,405,407
407 IF(KIND(IY)-1)405,408,405
408 IF(XLIBITF(IY+MOVED))405,405,409
C THERE IS AN EN PASSANT TRY
409 IZ = IBEG(IY)+J-1
IF (NEP(NUMEP)-MOVED)420,421,420
420 NUMEP=NUMEP+1
NEP(NUMEP)=MOVED
MEP1(NUMEP) = XABSF(MOVE(IZ))
GO TO 405
421 MEP2(NUMEP) = XABSF(MOVE(IZ))
405 CONTINUE
GO TO 134
C IS THIS MOVE A PROMOTION
412 IF(XADDF(MIN))419,418,419
C NOT A PROMOTION, IS IT AN EN PASSANT CAPTURE
416 IF (MOVEDIR-4)134,134,413
413 IF(IOCC(MOVER))134,416,134
C DIAGONAL MOVE TO EMPTY SQUARE
416 IX=XMVFIEXTS(MQ)+XGETF(I-XABSF(4-MOVEDIR-MOVEDIR)+IEXTU))
NSPEC=NSPEC+1
NUMB(NSPEC)=MOVED
NTYPE(NSPEC)=IX
ICAPT(MOVED)=-IOCC(IX)
CALL PUTC(IOCC(IX),0)
GO TO 134
419 IPROM = XADDF(MIN)
KIND(MOVER)=IPROM
IF (XLIBITF(MOVER))423,423,422
422 MATW=MATW+KVAL(IPROM)+1
GO TO 424
423 MATB=MATB+KVAL(IPROM)+1
424 NSPEC=NSPEC+1
NUMB(NSPEC)=MOVED
NTYPE(NSPEC)=-1
IEND(MOVER)=IBEG(MOVER)+MOVED(IPROM)+1
GO TO 134
C C HANDLES FIRST MOVE OF KING AND
C MAKES CASTLING MOVES
C
460 IF(NFIRST(MOVER))134,462,134
462 NFIRST(MOVER)=1
NSPEC=NSPEC+1
NUMB(NSPEC)=MOVED
NTYPE(NSPEC)=-1
C TEST FOR CASTLING MOVE
IF(XABSF(MOVEO-MQ)-2)134,463,134
463 IF(MOVEO-MQ)464,466,466
C CASTLE QUEENS SIDE
IA = -4 + MQ
JA = -1 + MQ
GO TO 467

C CASTLE KINGS SIDE

IA = 3 + MQ
JA = 1 + MQ
CALL PUTC(MOVER, MOVETO)
IAA = 10CC(IAA)
CALL PUTC(IAA, JA)
NTYPE(NSPEC) = -(IA - 1 + MSTO(IAA))
GO TO 139

C REVERT TAKES BACK MOVES

C

150 IF (MOVEP(MOVENO)) 201, 201, 167
C CHANGE SIDE TO MOVE
201 MCOL = 3 - MCOL
MIN = -0
GO TO 165
C NORMAL REVERSION
167 MOVER = MOVEP(MOVENO)
MOVETO = MOVEXFR(MOVENO)
ISPEC = 0
MIN = 0
C IS THIS A SPECIAL MOVE
151 IF (NUMB(NSPEC) - MOVENO; 152, 152, 152
C SPECIAL MOVE
152 ISPEC = NTYPE(NSPEC)
NUMB(NSPEC) = 0
NTYPE(NSPEC) = 0
NSPEC = NSPEC - 1
C SET UP VARIABLES
153 MQ = LOC(MOVER)
MCAPT = ICAPT(MOVENO)
ICAPT(MOVENO) = 0
MCOL = 2 - XLBITF(MOVER)
KD = KIND(MOVER)
C ORDINARY OR SPECIAL MOVE
154 IF (ISPEC) 155, 156, 154
C SPECIAL CASTLING OR PROMOTION
155 IF (ISPEC + 1) 154, 156, 155
C CASTLING

156 IF (XLBITF(MOVER)) 165, 166, 169
169 MATW = MATW - XGETF(KIND(MOVER), KVAL) + 1
GO TO 170
168 MATB = MATB - XGETF(KIND(MOVER), KVAL) + 1
170 KIND(MOVER)=1

C
C
C WAS IT FIRST MOVE OF K, R, OR P
154 IF (ISPEC=1) 171, 163, 171
C RESTORE NFIRST
163 NFIRST(MOVER)=0
C MOVE PIECE BACK
171 CALL PUCH(MOVER, MOVETO)
C WAS THE MOVE A CAPTURE OR EN PASSANT CAPTURE
IF (MCAPT) 158, 162, 160
C EN PASSANT CAPTURE
158 CALL PUCH(-MCAPT, ISPEC)
GO TO 162
C ORDINARY CAPTURE
160 CALL PUCH(MCAPT, MQ)
C
C IS THERE AN EN PASSANT POSSIBILITY
162 IF (NPEP(NUMEP)-MOVENO) 165, 166, 169
C YES, AT LEAST ONE
166 NUMEP=NUMEP-1
NPEP(NUMEP+1)=0
MEP1(NUMEP+1)=0
MEP2(NUMEP+1)=0
C RESET FUNCTIONS OF MOVENO
169 MOVEP(MOVENO)=0
MOVEFR(MOVENO)=0
ICAPT(MOVENO)=0
MOVENO=MOVENO-1
GO TO 700
END
LABEL
LIST
SUBROUTINE PUTCH (M6,M7)
DEC. 2, 1960, KUTOK, LIEBERMAN AND NIJSSEN.

DIMENSION IOCC(64), LOC(32), NFIRST(12), NUMB(50),
INTYPE(50), IBEG(32), IEND(32), MOVE(304), ICAPT(120),
2MOVEF(150), MOVEP(150), IBEAR(1024), IBEAR(64,16),
3KIND(32), MSVN(16), IPDIR(32), IEXTD(16), IEXTS(64),
4M64(16), NMOV(16), IOPP(16),
DIMENSION JPAWN(8),
DIMENSION MSTO(32),
DIMENSION MAVAIL(100), ITCH(2), ITCHU(2), IPIN(32),
DIMENSION NEP(10), MEP1(10), MEP2(10),
DIMENSION JPRM(4),
DIMENSION LOGG(101),
DIMENSION NZZZ(120),
DIMENSION KVAL(6), KFORCE(64), KWORTH(64),

COMMON STATEMENTS
COMMON IPDIR, IUPP, IEXTS, IEXTD, JPAWN, M64, MSVN, NMOV, MSTO, JPRM,
1IBEAR, JBEAR, KIND, IEND, IBEG, IOCC, LOC, NFIRST, MOVE, IENUS, MOVEP,
2MOVEF, ICAPT, NUMS, INTYPE, ITCH, ITCHU, IPIN, NEP, MEP1, MEP2, LOGG, NLOG,
3NZZZ, NUMTES, MAVAIL, 1Z, 1Y, IX, IX, IT, ISPEC, IR, IQ, JPRM, IOPP, INTER,
4DIR, IDIRECT, VA, IA, JA, JF, JG, JJ, JROOK, J, K, K1, KD,
5K, L2, L3, M4, MAREI, MACT, MCUL, MIN, MOVD, MKENO, MOVER, MOVEO, MOATC, MOE,
6MVR, N1, N2, NNEW, S, N, N, NPRED, NUMEP, NPRINT, KIN, KVAL, KFORCE, KWORTH, MOB1,
7MOB, MATW, MATB,

EQUIVALENCE (IENUS, IBEG(33), (NLOG, LOGG(101)), (NUMTES, NZZZ(120)),
1(IBEAR, JBEAR),
DIMENSION NUMBER(64),
COMMON NUMBER,
COMMON MLOG,

500 MOVES A PIECE FROM ONE SQUARE TO ANOTHER AND UPDATES THE
TABLES IBEAR, MOVE, LOC, IOCC, IBEG, IEND. IT USES 200, 300
AND 600 AS SUBROUNrines.

500 MVR = M6,
MTO = M7,
MOLDSQ = LOC(MVR),
LOC(MVR) = MTO

" IS MOVE FROM OFF BOARD
IF (MOLDSQ) 503 523 503

523 IF (XLBITF(MVR)-1) 530 531 532
532 STOP 532
531 MATW = MATW + XGETF(KIND(MVR), KVAL)
GO TO 516
530 MATB = MATB + XGETF(KIND(MVR), KVAL)

A PIECE COMING FROM OFF THE BOARD MAY NEED MOVE STORAGE

516 IF(IBEG(MVR)) 506 517 506
517 IOCC(MTO) = MVR
K = KIND(MVR)
IF(K-1) 518 519 518
518 MNREQ = NMOV(K)
GO TO 600
519 IF(XTRANKF(MTO,MVR)=7,518,520,518
520 MNREQ = 56
GO TO 600
C      DELETE OLD MOVES AND BEARINGS
503 IOCC(MOLDSQ)=0
M=IBEG(MVR)
N=IEND(MVR)
DO 501 J=M,N
   IF(MOVE(J)=510,501,510
510 K = XDELETE(MOVE(J))
   IF (JBEAR(K+1)=521,521,522
522 L2=XLBITF(MVR)
MOBW=MOBW-L2
MOBB=MOBB+L2-1
521 JBEAR(K+1)=0
MOVE(J)=0
501 CONTINUE
C      IS MOVE TO OFF BOARD
502 IF (MTO) 506,524,506
506 IOCC(MTO)=MVR
   IF(KIND(MVR)=1)512,513,512
C      IS THIS PAWN MOVING TO THE 7TH RANK
513 IF(XTRANKF(MTO,MVR)=7,512,514,512
514 IF (IEND(MVR)-IBEG(MVR)=33) 515,512,512
515 MNREQ=56
GO TO 600
C      UPDATE MOVES OF PIECE IN ALL DIRECTIONS. DATUM IS MTOUP
512 MTUOP = MVR
200 NOLDSQ=LOC(MTUOP)
MSTOP = MSTOP(MTUOP)
K=KIND(MTUOP)
GO TO (210,220,230,240,242,260),K
C      ROOK IN ALL DIRECTIONS
220 ASSIGN 221 TO JRET
DO 221 IDIR=1,4
   L=IBEG(MTUOP)+MNVN(IDIR)-0
GO TO 280
221 CONTINUE
GO TO 201
C
C      BISHOP IN ALL DIRECTIONS
240 ASSIGN 241 TO JRET
DO 241 IDIR=5,8
   L=IBEG(MTUOP)+MNVN(IDIR)-36
GO TO 280
241 CONTINUE
GO TO 201
C
C      QUEEN IN ALL DIRECTIONS
260 ASSIGN 261 TO JRET
DO 261 IDIR=1,8
L=IBEG(MToup)+MSVN(IDIR)-8
GO TO 280

261 CONTINUE
GO TO 201

C
C KING IN ALL DIRECTIONS
222 N1=1
GO TO 232

C N IN ALL DIRECTIONS
230 N1=9
232 N2=N1+7
L3=IBEG(MToup)-N1
DO 271 IDIR=N1,N2
L=L3+IDIR

C N IN GIVEN DIRECTION
DATA ARE MToup, IDIR, NOLDSQ
270 L1=M64M1(IDIR)+MSTOP
NEWSQ=XMOVF(EXT3(NOLDSQ)+EXT4(IDIR))
C IS THE SQUARE ON THE BOARD
273 IF(NEWSQ=64)272,272,271

C ON BOARD
272 IF (IBEAR(NEWSQ,IDIR)) 274,274,268
268 L10=XLB1TF(IBEAR(NEWSQ,IDIR))
MOB=W=MOB-W-L10
MOB=B=MOB-B-1+L10
279 L2=XLB1TF(MToup)
MOB=W=MOB+W+L2
MOB=B=MOB-B-L2+1
269 IBEAR(NEWSQ,IDIR)=MToup
C IS THE SQUARE OCCUPIED
274 IF(IOCC(NEWSQ))275,275,277
279 STOP275

C OCCUPIED. IS THE COLOR THE SAME AS THAT OF THE MOVER
277 IF(XLB1TF(IOCC(NEWSQ)-MToup))276,276,276
276 MOVE(L)=NEWSQ+Li
GO TO 271
278 MOVE(L)=-(NEWSQ+Li)
271 CONTINUE
GO TO 201

C
C UPDATE MOVES OF PAWN IN ALL DIRECTIONS
210-217 AND 320-350
C PURPOSE- TO UPDATE THE MOVES OF A PAWN IN ALL DIRECTIONS.
C ASSIGNS ADDITIONAL STORAGE TO PAWNS REACHING THE 7TH RANK.
C DOES NOT SET UP EN PASSANT MOVES. USES 600, XLB1TF, XMOVF,
C XRANKF, IPD1R, NFIRST, IEXTS, IEXTD, IOCC,
C TABLES AFFECTED- MOVE, IBEG, IEND, IBEAR,
C LOCAL VARIABLES- J,K, L, JREF, MNREG, MNREG, K1 NEWSQ, IDIR, L1,
C AND L2
C DATA SUPPLIED - MToup, NOLDSQ, IENUS(INITIALLY)
210 K=XLBI1TF(MToup)+1
L9 = IBEG(MToup)-1
DO 211 J=1,3
IDIR=IPD1R(J,K)
L = L9+J
ASSIGN 211 TO JARET
GO TO 320
211 CONTINUE
GO TO 201
201 MSQ=MTO
ASSIGN 508 TO MRET
GO TO 300
C IS MOVE FROM ON BOARD
C REMOVE PIECE FROM MATERIAL COUNT
524 IF (XLBITF(MVR)-1) 526,528,527
526 MAT8=MAT8-XGETF(KIND(MVR),KVAL)
GO TO 508
527 STOP 527
528 MATW=MATW-XGETF(KIND(MVR),KVAL)
508 IF (MOLDSQ)511,509,511
511 MSQ=MOLDSQ
ASSIGN 509 TO MRET
GO TO 300
C RETURN
C MOVE STORAGE CONTROL 600 TO 625
C PURPOSE- TO EXPAND AND CONTRACT THE MOVE
C STORAGE ALLOTTED TO PAWNS WHEN THEY
C REACH THE 7TH RANK OR REVERT TO 1T
C TABLES AFFECTED- MOV, IBEG, IEND
C DATA SUPPLIED- - MNREQ, MVR, IENUS(INITIALLY)
C LOCAL VARIABLES M1,M,N,J6,K,M2
C C MOVE STORAGE CONTROL
600 IF (504-IENUS-MNREQ)601,602,603
C STORAGE AVAILABLE AT THE END
602 IF (IBEG(MVR)) 604,606,605
C MOVE THE MOVE INFORMATION
605 M1=IENUS+1
M=IBEG(MVR)
N=IEND(MVR)
DO 606 J6=M*N
MOVE(M)=MOVE(J6)
MOVE(J6)=0
606 M1=M1+1
604 IBEG(MVR)=IENUS+1
IENUS = IENUS + MNREQ
IEND(MVR)=IENUS
GO TO 512
C NOT ENOUGH STORAGE, RESORT
C MAKE SURE CAPTURED PIECES USE NO STORAGE
601 DO 607 J6=1,32
IF (LOC(J6)) 608,607,615
608 IBEG(J6)=0
IEND(J6) = 0
GO TO 607
C PAWNS ON OR BELOW 6TH RANK NEED ONLY 4 MOVES
615 IF (XMINOF(1-KIND(J6),6-XTRANK(LOC(J6),J6))) 607,616,616
616 IEND(J6)=IBEG(J6)+3
607 CONTINUE
M1=1
620 M2=0
DO 609 J6=1,32
IF(M1=IBEG(J6))612,61,609
C HAS J ALREADY BEEN RE-ARRANGED.
612 IF(M2=IBEG(J6))613,617,617
613 IF(M2:617,617+609
617 M2=IBEG(J6)
K=J6
GO TO 609
C NO NEED TO ARRANGE THESE MOVES
611 M1=IEND(J6)+1
GO TO 620
609 CONTINUE
IF(M2:622,622,623
C RE-ARRANGE
623 M=IBEG(K)
N=IEND(K)
IBEG(K)=M1
DO 624 J6=M*N
MOVE(M1)=MOVE(J6)
MOVE(J6)=0
624 M1=M1+1
IEND(K)=M1-1
GO TO 620
C TOTAL STORAGE TOO SMALL AFTER RE-ARRANGEMENT
625 STOP 625
C
C UPDATE ALL PIECES BEARING ON MSQ
300 DO 301 IDIR=1,16
IF (IBEAR(MSQ,IDIR)) 303,301,303
303 MToup=XABSF(IBEAR(MSQ,IDIR))
MSSTOP = MSTOP(MToup)
K=KIND(MToup)
NOLDSQ=LOC(MToup)
ASSIGN 301 TO JRET
GO TO (313,310,314,312,315,310),K
MOV E OF KNIGHT IN GIVEN DIRECTION
314 N1=9
GO TO 317
C MOVE OF KING IN GIVEN DIRECTION
315 N1=1
C CHANGE LEGALITY OF KNIGHT OR KING MOVES
317 IF(MV=MToup) 311,301,311
311 IF(XLBITF(MV=MToup)) 301,316,301
316 L=IBEG(MToup)+IDIR-N1
MOVE(L)=MOVE(L)
301 CONTINUE
GO TO MRET,(508,509)
C
C UPDATE ROOK OR QUEEN IN GIVEN DIRECTION
310 L=IBEG(MTOUP)+MSVN(IDIR)-8
GO TO 280
C UPDATE BISHOP IN GIVEN DIRECTION
312 L=IBEG(MTOUP)+MSVN(IDIR)-36
GO TO 280
313 ASSIGN 301 TO JAREP
J=JPAWN(IDIR)
L=IBEG(MTOUP)+J-1
GO TO 320
C UPDATE Q, B, OR R IN GIVEN DIRECTION
280 L1 = M64M1(IDIR) + MSTOP
L2=XLBITF(MTOUP)
IQ=IEXTD(IDIR)
IR=IEXTS(NOLDSQ)
DO 281 J=1,7
IR=IR+IQ
NEWSQ=XMOVF(IR)
288 IF(NEWSQ-64)284,284,299
284 IF (IBEAR(NEWSQ,1DIR)) 282,282,299
299 L10=XLBITF(IBEAR(NEWSQ,1DIR))
MOBW=MOBW-L10
MOBB=MOBB-1+L10
282 MOBW=MOBW+L2
MOBB=MOBB+L2+1
IBEAR(NEWSQ,1DIR)=MTOUP
J1=L+J
289 IF(IOCC(NEWSQ))285,285,287
285 STOP 2105
281 MOVE(J1)=NEWSQ+L1
C NON EXISTENT SQUARE
283 GO TO JRET,(221,241,261,301)
C SQUARE OCCUPIED
287 IF(XLBITF(IOCC(NEWSQ)-MTOUP))290,291,290
290 MOVE(J1)=NEWSQ+L1
GO TO 292
291 MOVE(J1)=-(NEWSQ+L1)
292 IF (J-6) 252,252,251
252 DO 294 J3=J,6
J1=L+J3+1
293 IF(MOVE(J1))295,296,295
296 GO TO JRET,(221,241,261,301)
295 MOVE(J1)=0
IR=IR+IQ
NEWSQ=XMOVF(IR)
286 IF (XABS(IBEAR(NEWSQ,1DIR))-MTOUP) 294,294,294
298 IBEAR(NEWSQ,1DIR)=0
MOBW=MOBW-L2
MOBB=MOBB+L2-1
294 CONTINUE
251 GO TO JRET,(221,241,261,301)
C 320 UPDATES A PAWN IN A GIVEN DIRECTION, COPIES MOVES OVER FOR A PAWN ON THE 7TH RANK.
C USES-XLBITF, XMOVF, IEXTS, IEXTD, M64M1, IOCC, NFIRST.
TABLES AFFECTED: IBEAR, MOVE.
LOCAL VARIABLES: (NEWSQ, L1, L2, L3, MSTOP, MTUOP, JARET, L, J).

320 NEWSQ=XMOVF(IEXTS(NOLDSQ)+IEXTD(IDIR))
IF(NEWSQ-64) 321, 322, 322
322 GO TO JARET,(211, 301)
321 L1 = M64M1(IDIR) + MSTOP
L3=XLBITF(MTUOP)
IF (IBEAR(NEWSQ, IDIR)) 342, 342, 343
343 L10=XLBITF(IBEAR(NEWSQ, IDIR))
MOBW=MOBW-L10
MOBB=MOBB+L1+L10
342 IBEAR(NEWSQ, IDIR)=MTUOP
MOVE(L)=NEWSQ+L1
L2=IOCC(NEWSQ)
IF(J-3) 330, 323, 323
320 MOBE IS DIAGONAL
MOBW=MOBW+L3
MOBB=MOBB+L3+1
IF (L2) 328, 328, 326
326 IF(XLBITF(L2+MTUOP)) 328, 328, 320
328 MOVE(L)=-MOVE(L)
323 IBEAR(NEWSQ, IDIR)=-XABSF(IBEAR(NEWSQ, IDIR))
IF (L2) 331, 331, 332
331 IF(INFIRST(MTUOP)) 334, 334, 335
335 MOVE(L+1)=0
330 IF (XTRANKF(NOLDSQ,MTUOP)-7) 338, 338, 338
334 NEWSQ=XMOVF(IEXTS(NEWSQ)+IEXTD(IDIR))
IBEAR(NEWSQ, IDIR)=-MTUOP
MOVE(L+1)=-XSIGNF(L1+NEWSQ, IOCC(NEWSQ)-1)
338 GO TO JARET,(211, 301)
322 REMOVE POSSIBLE FALSE BEARING
MOVE(L)=-MOVE(L)
MOVE(L+1)=0
339 NEWSQ=NEWSQ+24-8*IDIR
IF (IBEAR(NEWSQ, IDIR),) 338, 341, 338
341 IBEAR(NEWSQ, IDIR) = 0
GO TO JARET,(211, 301)
353 MOVE(L+4)=MOVE(L)+XSIGNF(JPRUM(2), MOVE(L))
MOVE(L+8)=MOVE(L)+XSIGNF(JPRUM(3), MOVE(L))
MOVE(L+12)=MOVE(L)+XSIGNF(JPRUM(4), MOVE(L))
MOVE(L)=MOVE(L)+XSIGNF(JPRUM(1), MOVE(L))
GO TO JARET,(211, 301)

END
* LABEL
* LIST8

CONLINE CHESS MAIN PROGRAM, FEB. 28, 1962

DIMENSION FOO(5000)
DIMENSION LOC(32), NFIRST(22), KPAWNV(8), IEXTD(16), IEXTS(64)
DIMENSION IPIN(32), IOPP(16), KIND(32), MAVAIL(100), KVAL(6)
DIMENSION IHOPE(64), IEXCH(128)
DIMENSION LISP(6000)

COMMON FOO
EQUIVALENCE (NSPEC, FOO(12913))
EQUIVALENCE (FOO(2892), K1), (FOO(1463), KIND), (FOO(2765), MAVAIL)
EQUIVALENCE (FOO(12900), MCOL)
EQUIVALENCE (FOO(12703), IPIN), (FOO(1285), IOPP), (FOO(255), IBEAR)
EQUIVALENCE (FOO(1365), IEXTD), (FOO(1301), IEXTS), (FOO(1527), IOCC)
EQUIVALENCE (FOO(1591), LOC), (FOO(1623), NFIRST), (FOO(3003), KPAWNV)
EQUIVALENCE (FOO(3121), PLY), (FOO(3120), IPE), (FOO(2917), KVAL)
EQUIVALENCE (FOO(3051), MOBN), (FOO(3052), MOBB), (FOO(3123), IHOPE)
EQUIVALENCE (FOO(9188), IEXCH), (FOO(3122), BACK), (FOO(3187), LISP)
EQUIVALENCE (FOO(3053), MATN), (FOO(3054), MATB), (FOO(3119), MLOG)
EQUIVALENCE (FOO(134), NLOG)
EQUIVALENCE (FOO(2903), MOVENU)
DIMENSION KPLY(20)
EQUIVALENCE (KPLY, FOO(9167))
EQUIVALENCE (FOO(9316), MOVES), (FOO(9317), NMOVES)

CALL BEGIN
READ 101, (KPLY(I), I = 1, 20)

101 FORMAT (20I3)

26 J=1
REWIND 6
NMOVES=0
CALL INITIA(J)
CALL PRINT (-7)
WRITE OUTPUT TAPE 100, 1

FORMAT(59H0 THE MIT CHESS PROGRAM WELCOMES YOU AS ITS WORTHY OPPONENT.
INT. 11TH IF YOU WISH TO PLAY WHITE, KEY IN THE NUMBER OF YOUR MOVE.
2 IN THE DECREMENT OF THE KEYS. IF BLACK, SET KEYS TO ZERO. A 9H IF
3 AT ANY TIME, YOU WISH TO START OVER, SET ADDRESS OF KEYS NON ZERO
4. THEN PRESS START. A 90H KEYS NEGATIVE PRINTS HISTORY. A 1H1)
PAUSE
IF(KEYS(J)) 3,3,2
3 IF(J) 4,4,5
4 WRITE OUTPUT TAPE 100, 7
7 FORMAT(14H0, MACHINE FIRST)
GO TO 10

C
15 CALL REVERT
14 J=1
5 IF(K1-J) 69,8,8
8 J=J
MOVES=MAVAIL(J)
CALL UPDATE(MAVAL(J))
CALL PRINT (-7)
10 WRITE OUTPUT TAPE 100, 9
9 FORMAT(95H0 IF THIS MOVE IS CORRECT, SET KEYS TO ZERO AND PRESS STA
1RT. OTHERWISE SET KEYS TO CORRECT MOVE.../IH1)
1003 PAUSE
  IF(KEYS(J)) 1002, 11, 2
11 IF (I) 12, 12, 13
13 IF (J) 14, 14, 15
12 IF (K1) 16, 16, 18
16 WRITE OUTPUT TAPE 100,19
19 FORMAT(6H0DARN,,/4H1HCARE TO TRY AGAIN... PRESS START IF SO./IH1)
PAUSE
GO TO 2
18 L = XTIMES(L)
CALL TREE (MOVE)
TIME = XLAPESE(L)
CALL UPDATE(MOVE)
8 CALL PRINT (407777000000)
33 IF(K1)20,16,17
20 WRITE OUTPUT TAPE 100, 21
21 FORMAT(16H0WHOOPEE, I WIN./4H1HCARE TO LOSE AGAIN... PRESS START
1 IF SO./IH1)
PAUSE
GO TO 2
17 WRITE OUTPUT TAPE 100, 22, TIME
22 FORMAT (24H0THE PRECEDING MOVE TOOK, -1PF4.1, 9H MINUTES./4HOPLEA
1SE KEY IN YOUR REPLY AND PRESS START.)
RECORD 7
NLOG = 0
MLOG = 0
25 PAUSE
23 IF(KEYS(J)) 69, 23, 2
69 IF(J) 69, 69, 65
C ERROR PSEUDO STOP
69 WRITE OUTPUT TAPE 100, 691
691 FORMAT(25H1ILLEGAL MOVE, TRY AGAIN./IH1)
GO TO 25
C START OVER
2 IF (SENSE SWITCH 3) 709, 1090
7090 BACKSPACE 4
BACKSPACE 4
B709 CALL PRINT (77/40000000)
RECORD 7
MLOG = 0
NLOG = 0
GO TO 26
B1002 CALL PRINT (410000000000)
GO TO 1003
END
* LABEL
* FAP
* COUNT 354
* FUNCTION INITIA, M179 CHESS, APR. 17, 1961
ENTRY INITIA

INITIA SXD XR4$4
SXA XR2$2
SXA XR1$1
STI INDIC
CLA* 1$4
TZE A1342
AXT 32$1

LP32 STZ IBEG+1$1
STZ IEND+1$1
STZ LOC+1$1
STZ IPIN+1$1
STZ LOCIN+1$1
TIX LP32$1$1
AXT 100$1

LP100 STZ MAVAIL+1$1
TIX LP100$1$1
STZ IENUS
AXT 22$1

LP22 STZ NFIRST+1$1
TIX LP22$1$1
AXT 50$1

LP50 STZ NUMB+1$1
STZ NTYPES+1$1
TIX LP50$1$1
AXT 64$1

LP64 STZ IOCC+1$1
PXD ,1
STO NUMBER+1$1
TIX LP64$1$1
AXT 504$1

LP504 STZ MOVE+1$1
TIX LP504$1$1
AXT 150$1

LP150 STZ ICAPT+1$1
STZ MOVEFR+1$1
STZ MOVEP+1$1
TIX LP150$1$1
STZ MATW
STZ MATB
STZ MOBW
STZ MOBB
STZ NUMEP
STZ ISPEC
STZ NSPEC
STZ MOVENO
AXT 1024$1

LP1024 STZ JBEAR+1$1
TIX LP1024$1$1
STZ ITCH

CLEAR TABLES
STZ ITCH-1
STZ ITCHD
STZ ITCHD+1
AXT 10+1
LP10 STZ NEP+1,1
STZ MEP1+1,1
STZ MEP2+1,1
TIX LP10+1,1
CLA =IB17
AXT 7+1
LP722 STO KIND+1,1
TXI +=1,1,1
TXL LP722+1,22
INPUT CLA =IB17 READ PROBLEM
STO LOC1
STO COLOR
AXT INS+1,4
SXA INS,4
STZ LETTER
AXT 0+2
CARD CAL =4B17 READ IN ANOTHER CARD
TSX $HTSH,4
PZE =H(12A6)
AXT 12+1 FORTRAN READ INPUT TAPE 4
STR
STQ TABLE+12,1
TIX -=2+1,1
TSX $6RTN,4
CAL TABLE
LAS =HFORTRA
TRA +=2
TRA B1254
AXT 12+1

B AXT 6+4 WORD COUNT
LDQ TABLE+12,1 CHARACTER COUNT
A SXA CHLOOP,4
PXD
LGL 6
INS TRA *
CAS =H00000
TRA ++2
TRA PERIOD
CAS =H00000 BLANK
TRA ++2
TRA CHLOOP BLANKS IGNORED
CAS =H00000*
TRA ++2
TRA COLOR1 NUMERAL
CAS =H000009
TRA ++3
NOP NOP
TRA NUMBUH
CAS =H000001 OPEN PARENTHESIS
TRA ++2
TRA OPEN
CAS =H00000
TRA ++2
TRA CLOSE
CAS =H00000
TRA ++2
TRA BREAK
CAS =H00000K
TRA ++2
TRA BREAK
CAS =H00000/
TRA ++2
TRA COMMENT
ADD LETTER
SHIFT ALS 6
STO LETTER
TXI CHLOOP,2,1
COLOR1 TSX LOOKUP,4
STZ COLOR
TRA RESETL
B1234 CAL =4B17
TSX $(BST),4
PX D
LXD X4,4
TRA A1342
NUMBUH STO NUM
TSX LOOKUP,4
CLA NUM
ALS 18
ADD LOC1
CAS =65817
TSX ERROR,4
NOP
RESETL AX T0,2
STZ LETTER
AX T INS+1,4
SXA INS,4
CHLOOP AX T **,4
TXI A+4,1
TXI B+1,1
TRA CARD
LOOKUP CLA LETTER
TXL FOUND1,2,0
TXL ONE,2,1
TXL TWO,2,2
TXL THREE,2,3
TSX ERROR,4
THREE ALS 12
TRA PLACE
TWO ALS 18
ORA =H00 000
TRA PLACE
ONE ALS 18

Q OR K BEGINS A NEW PIECE

ANYTHING ELSE ASSUMED LETTER

INCREASE LETTER COUNT

RESET CHARACTER COUNTER

READ ANOTHER CARD

CLOSED SUBROUTINE TO LOOKUP PIECE

NORMALIZE
ORA =H 0 000
PLACE ORA =H000---
ZET COLOR
ORA =H000
AXT 32 * 2
LAS PIECES,2
TRA *+2
TRA FOUND
TIX *-3 * 2 * 1
TSX ERROR * 4
FOUND CLA LOC1
CAS =65B17
NOP
TSX ERROR * 4
ZET LOCIN+1,2
TSX ERROR * 4
STO LOCIN+1,2
ADD =1817
STO LOC1
STL COLOR
TXH FOUND1,2 * 22
CLA LOCBE6+1,2
SUB LOCIN+1,2
TZE * +2
CLA =1817
SSP
STO NFIRST+1,2
FOUND1 TRA 1,4
COMENT AX T COMEN1,4
TRA CHLOOP -1
COMEN1 CAS =H000000
TRA CHLOOP
TRA RESETL
TRA CHLOOP
OPEN CLA LETTER
STO CHANGE
SXA MOVED+1,2
TNZ RESETL
TSX ERROR * 4
CLOSE NZT LETTER
TSX ERROR * 4
CLA LETTER
CAS =H0000M0
TRA * +2
TRA MOVED
TSX LOOKUP * 4
CLOSE1 CLA CHANGE
PROMOTED PIECE HANDLED HERE
RIL 7
CAS =H0000RU
TRA * +2
LDI =2B17
CAS =H0000B0
TRA * +2
LDI =4B17
CAS =H0000U
TRA ++2
LDI =3B17
CAS =H000Q0
TRA ++2
LDI =6B17
LFT 7
TRA CLOSE2
TSX ERROR4
CLOSE2 TXH ++2,2,6
TSX ERROR4
STI KIND+1,2
TRA MOVED1
MOVED CLA CHANGE
AXT **+2
STO LETTER
TSX LOOKUP4
MOVED1 TXL ++2,2,2
TSX ERROR4
CLA =1B17
STO NFIRST+1,2
TRA RESETL
ERROR SXA ERLC4
STL COLOR
LAC ERLC4
ERROR1 TIX ++1,4,INITIA-9
SXA ERLC4
LXD XR4,4
CLA* 1,4
STO J
CAL =106B17
TSX $(MTH),4
PZE ERFOR
LDQ ERLC
STR
LDQ J
STR
LDQ LOC1
STR
AXT 12,2
LDQ TABLE+12,2
STR
TIX *-2,2,1
TSX $(FILES),4
NZT COLOR
TRA A5678
AXT ERROR3,4
TRA CHLOOP-1
ERROR3 CAS =H0000U
TRA CHLOOP
TRA ++2
TRA CHLOOP
A5678 LXD XR4,4
XR4 SYN INITIA-2

(M) MEANS PIECE HAS MOVED
LOOK FOR END OF PROBLEM
CLA* 1,4
SUB  =1B17
STO* 1,4
TNZ LP32-1
TRA XR1-2
BREAK STO KORQ
TSX LOOKUP,4
AXT 0,2
CLA KORQ
TRA SHIFT
ERROR2 STZ COLOR
TRA ERROR1
PERIOD TSX LOOKUP,4
CLA LOC1
SUB =65B17
AXT **+1,4
TNZ ERROR2
CLA =2B17
ZET COLOR
SUB =1B17
STO MCOL
AXT 1,1
PTCH NZT LOCIN+1,1
TRA PTCHLP
SXD JIN,1
PX A LOCIN+1,1
SUB **-1
STA **+3
CALL PUTC,H JIN LOCIN
PTCHLP TXI **+1,1,1
TXL PTCH,1+32
CALL SETUP
LXD XR4*4
CLA* 1,4
A1342 SUB =1B17
STO* 1,4
XR1 AXT **,1
XR2 AXT **,2
LDI INDIC
TRA 2,4
BCI 1,6
BCI 1\textbackslash H012A
BCI 1\textbackslash CARUS/
BCI 1\textbackslash O\textbackslash WING
BCI 1\textbackslash N\textbackslash FULL
BCI 1\textbackslash O\textbackslash UNDU\textbackslash U
BCI 1\textbackslash RROR R
BCI 1\textbackslash 3\textbackslash 2\textbackslash H. E
BCI 1\textbackslash C1=14,
BCI 1\textbackslash 7\textbackslash H. LO
BCI 1\textbackslash J=14,
BCI 1\textbackslash ATIV E
BCI 1\textbackslash 9\textbackslash H REL
BCI 1\textbackslash ON06,1
BCI 1*LOCAT1
BCI 1*JA AT
BCI 1*Y INIT
BCI 1*OUND 6
BCI 1*RROR 7
ERFOR BCI 1*34H4E
ZILCH COMMON 12561
R COMMON 1
*
TEMPORARY STORAGE
J PZE
COLOR PZE
INDIC PZE
ERLOC PZE
CHANGE PZE
LETTER PZE
LOC1 PZE
NUM PZE
KORQ PZE
JIN PZE
TABLE BSS 12
BSS 31
LOCIN BSS 1
ITCH SYN R+9863
ITCHD SYN R+9861
IBEG SYN R+12561
IEND SYN R+11067
LOC SYN R+10971
IPIN SYN R+9859
MAVAIL SYN R+9797
IENUS SYN R+12529
NFIRST SYN R+10939
NUMB SYN R+9963
NTYPE SYN R+9913
IOCC SYN R+11035
NUMBER SYN R+9507
MOVE SYN R+10917
ICAPT SYN R+10113
MOVEFR SYN R+10263
MOVEP SYN R+10413
JBEAR SYN R+12307
NEP SYN R+9827
MEP1 SYN R+9817
MEP2 SYN R+9807
KIND SYN R+11099
MCOL SYN R+9662
PIECES SYN R+9624
LOCBEQ SYN R+9581
MOVENO SYN R+9559
MATB SYN R+9508
MATW SYN R+9509
MOBB SYN R+9510
MOBW SYN R+9511
NUMEP SYN R+9648
ISPEC SYN R+9692
CONTROL WORD BITS ARE IN DECREMENT
1 PRINTS NUMBER, MOVER, MOVE TO ON-LINE, OTHERWISE OFF-LINE.
2 PRINTS BOARD ON-LINE, OTHERWISE OFF-LINE.
4 PRINTS MAVAIL, ON-LINE IF CONTROL WORD IS NEGATIVE.
10 PRINTS MAT, MOB, COLOR, MOVENO, NSPEC, ICHECK, MLOG OFF-LINE.
20 PRINTS LOC, IBEG, IEND, NFIRST, KIND, IPIN OFF-LINE.
40 PRINTS MOVEP, MOVEFR, ICAPT OFF-LINE.
100 PRINTS NUMB, ITCH, ITCHD, NEP. MEP1, MEP2 OFF-LINE.
200 PRINTS LOG OFF-LINE.
400 PRINTS IBEAR OFF-LINE.
1000 PRINTS MOVE TABLE OFF-LINE.
2000 PRINTS PRINCIPAL VARIATION, ONLINE IF NEGATIVE.
4000 PRINTS MOVE TREE OFF LINE
10000 PRINTS HISTORY, ONLINE IF NEGATIVE.
DIMENSION AND EQUIVALENCE STATEMENTS
DIMENSION IOCC(64), LOC(32), NFIRST(22), NUMB(50),
1NTYPE(150), IBEG(33), IEND(32), MOVE(150), ICAPT(150),
1MOVEFR(150), MOVEP(150), JBEAR(1024), IBEAR(164, 16),
3KIND(32), MSVN(16), IPDIR(32), IEXTD(16), IEXTS(64),
4M64M(16), MNMOV(6), IOPP(16)
DIMENSION JPAYN(8),
DIMENSION MSU(32),
DIMENSION MAVAIL(100), ITCH(2), ITCHD(2), IPIN(32),
DIMENSION NEP(10), MEP1(10), MEP2(10),
DIMENSION JROM(4),
DIMENSION LOGG(101),
DIMENSION NZZZ(120),
DIMENSION KEVAL(6), KFORCE(64), KWORD(164)
COMMON STATEMENTS
COMMON IPDIR, IOPP, IEXTS, IEXTD, JPAYN, M64M, MSVN, NMOV, MSTU, JROM,
JBEAR, JBEAR, KIND, IEND, IBEG, IOCC, LOC, NFIRST, MOVE, IENUS, MOVEP,
2MOVEFR, ICAPT, NUMB, NTYPE, ITCH, ITCHD, IPIN, NEP, MEP1, MEP2, LUGG, NLOG,
3NZZZ, NUMTES, MAVAIL, IZ, IY, IX, IU, IT, NSPEC, IK, IQ, JROM, IOPPD, INTEK,
4IPDIR, ICHECK, IA, IA, A, JB, JC, JDIR, JD, JEPF, JIN, JJS, JROOK, JKS, KD,
5K, L2, L4, M4, MAREM, ICAPT, MCOL, MIN, MOVDFR, MOVENO, MOVER, MOVEP, MQ, M9,
6MVDR, N1, N2, NEWSU, NNSPEC, NUMEP, NPRINT, KIN, KEVAL, KFORCE, KWORTH, MOBW,
7MOBB, MATW, MATB
EQUIVALENCE (IENUS, IBEG(33)), (NLOS, LUGG(101)), (NUMTES, NZZZ(120)),
(1IBEAR, JBEAR)
DIMENSION NUMBER(64), IEXCH(128)
COMMON NUMBER
COMMON MLOG
DIMENSION LISP (6000), IHOPE (64)
COMMON IPE, PLY, BACK, IHOPE, LISP, IPRINT
COMMON IEXCH
COMMON MOVES, NMOVES
DIMENSION M1(100), M2(100), A1(100), A2(100)
EQUIVALENCE (M1, A1), (M2, A2)
EQUIVALENCE (1, A1)
C
CODEWD=CODE
IPRINT = IPRINT + 1
C
NUMBER, MOVER, MOVETO
B
IF (CODEWD*000001000000) 6969, 1000, 1001
1000 N=2
GO TO 5
1001 N = 100
5 CALL JUNPAK (MOVETO+MSTO (MOVER) -1, M1 (1), M1 (2))
WRITE OUTPUT TAPE N, 910, IPRINT, M1 (1), M1 (2)
910 FORMAT (21H1SET OF TABLES NUMBER,13,1OH - MOVE IS ,2A6)
C
IOCC
B
IF (CODEWD*000002000000) 6969, 47, 48
47 N = 2
GO TO 49
48 N = 100
49 CALL BOARD (N)
C
MAVAIL
B
IF (CODEWD*000004000000) 6969, 130, 20
50 IF (CODEWD) 51, 6969, 52
51 N = 100
GO TO 54
52 N = 2
54 IF(K1)43,42,44
42 WRITE OUTPUT TAPE N, /0
70 FORMAT (10H STALEMATE )
GO TO 475
45 WRITE OUTPUT TAPE N, /3
73 FORMAT (10H CHECKMATE )
GO TO 475
44 WRITE OUTPUT TAPE N,980
980 FORMAT (7H MAVAIL )
82 DO 17 I=1,K1
17 CALL JUNPAK (MAVAIL (I), M1 (1), M2 (1))
WRITE OUTPUT TAPE N, 1391, (K1(I), M2(I), I=1,K1)
475 WRITE OUTPUT TAPE N, 139
139 FORMAT (1H4)
1391 FORMAT (1H0,20A6)
130 CONTINUE
C
2000 PRINTS PRINCIPAL VARIATION, ONLINE IF NEG.
B
IF (CODEWD*002000000000) 6969,224,223
223 N=2
IF (CODEWD) 221,220,220
221 N=100
220 INT=1
CALL JUNPAK (MOVES*M1(1),M1(1))
I=1
230 INT=XBANDF(XADD(LISP(I+1)),1<7)
IF (INT) .215,215,225
225 INT=1+INT+1
I99=I99+1
M1(I99)=XDECF(LISP(INT))+XSHIFTF(XTAGF(LISP(INT)),-18)
CALL JUNPAK(M1(I99),M1(I99),M1(I99+50))
I=XADDI(LISP(INT))
GO TO 230

215 WRITE TAPE 6,I99,M1
NMoves=NMoves+1
WRITE OUTPUT TAPE N, 222, LISP(I+1), MLOG, (M1(1), M1(1+50), I=2,I99)

222 FORMAT ('1H1PRINCIPAL VARIATION /H VALUE='1/, &
1(1H0,20A6))

224 CONTINUE

C 4000 PRINTS MOVE TREE
B IF(CODEW*004000000000) 6969,270,261
B261 AM1(I)=3
LEVEL=1
PRINT 260,(I;I=1,20)

260 FORMAT (1H1,51X,13THE MOVE TREE/E1H0LEVEL,20D8H VALUE)
I=3

263 IF(LISP(I)) 262,270,264
I=I+1
GO TO 263

B262 AM1(LEVEL)=(AM1(LEVEL)*77777)+A1
CALL WRITE(LISP(I),LEVEL)
I=XADDI(LISP(I))
IF(I) 270,269,271

271 IF(LISP(I)) 268,270,265
I=I+2
LEVEL=LEVEL+1
M1(LEVEL)=XSHIFTF(I,-18)
GO TO 263

268 PRINT 274,LISP(I+1)

274 FORMAT (1H1+,10X,110)
I=XDECF(M1(LEVEL))-1
IF(XADDI(M1(LEVEL))-1) 262,262,267

267 LEVEL=LEVEL-1
IF(LEVEL) 270,270,269

270 CONTINUE

C EVALUATION PARAMETERS
B IF(CODEW*000010000000) 6969,60,134

134 PRINT 133, MATH, MOBW, MATB, MOBB

133 FORMAT ('1H2,8X,1H MATIERIAL MOBILITY/6H WHITE,2110/6H BLACK,2110)
IF(MCOL=1) 6969, 20, 21
B 20 AM2 = 6066303163Q5
GO TO 62

B 21 AM2 = 6022432143Q4
62 PRINT 22, K1, M2 (1), MOVENU, NSPEC, ICHECK, MLOG
22 FORMAT(1H9 NUMBER OF MOVES = 13/8H MCOL IS A6/10H MOVENU = 13/9H N &
1SPEC = 14/9H ICHECK =14/7H MLOG =16)

60 CONTINUE

C PRINT THE OTHER TABLES
C LOC, IBEQ, IEND, NFIRST, KIND, IPIN
B  IF(CODEWD#000020000000) 6969,60,63
63  WRITE OUTPUT TAPE 2,2*(I,1=1,32),(LOC(I),I=1,32),(IBEG(I),I=1,32),
1*(IEND(I),I=1,32),(NFIRST(I),I=1,22),(KIND(I),I=1,32),(IPIN(I),I=1,32)
2 FORMAT (8H0PIECE 3213/8H LOC 3213/8H IBEH 3213/8H IEND
13213/8H NFIRST 2213/8H KIND 3213/8H IPIN 3213)
80  CONTINUE
C  MOVEP, MOVEFR, ICAPT
B  IF(CODEWD#000040000000) 6969,90,9
81  WRITE OUTPUT TAPE 2,8*(MOVEP(I),I=1,MOVEN)
8 FORMAT (6H MOVEP1(16/2016))
8 WRITE OUTPUT TAPE 2,7*(MOVEFR(I),I=1,MOVEN)
7 FORMAT (6H MOVEFR(16/2016))
6 FORMAT (6H ICAPT1(16/2016))
90  CONTINUE
C  NUMS, NTYPE, ITCH, ITCHD, NEP, MEP1, MEP2
B  IF(CODEWD#000100000000) 6969,162,99
95  WRITE OUTPUT TAPE 2,91,(NUMS(I),I=1,NSPEC)
91  FORMAT (6H NUMS 1616/2016))
9 WRITE OUTPUT TAPE 2,92,(NTYPE(I),I=1,NSPEC)
92  FORMAT (6H NTYPE1(16/2016))
9 WRITE OUTPUT TAPE 2,93,(ITCH(I),I=1,2),(ITCHD(I),I=1,2)
93  FORMAT (6H ITCH 213,8H ITCHD 213)
C  SET UP NEP, MEP1, AND MEP2 FOR OUTPUT
153  DO 150 J=1,60
150  M1(J)=0
151  IF (NEP(1)) 151,150,1>1
151  M1(I)+20=XMVF(MEP1(I))
151  M1(I)+40=XMVF(MEP1(I))
151  IF (MEP2(I)) 155,150,159
155  M1(I)+10=XMVF(MEP2(I))
151  M1(I)+30=XMVF(MEP2(I))
151  M1(I)+50=XMVF(MEP2(I))
150  CONTINUE
C  PRINT OUT THE EN PASSANT TABLES
WRITE OUTPUT TAPE 2,154,(NEP(I),I=1,10),(M1(I),I=1,60)
154  FORMAT (4H NEP101,5H MEP11013,5H MEP21013/(142,913,18,913))
162  CONTINUE
C  WRITE THE LOG
B  IF(CODEWD#000200000000) 6969,170,164
164  PRINT 165, MLOG
165  FORMAT (17H1 THE LOG---MLUG=+15///
I1=0
IF (MLOG=100) 160,160,161
161  REWIND 7
166  DO 166 I3=100,MLOG,100
166  I1=13
READ TAPE 7, M1
DO 1640 I = 1, 100
1640 CALL JUNPAK (M1(I), M1(I), M2(I))
166 PRINT 163, (M1(I), M2(I), I = 1, 100)
163 FORMAT (1H0, 2A6, A7, A6, A7, A6, A7, A6, A7, A6, A7, A6, A7, A6, A7, A6, A7, A6, A7, A6)
160 I2 = MLOG-I1
IF (I2) 170, 170, 167
167 DO 169 I = 1, I2
169 CALL JUNPAK (LUGG(I), M1(I), M2(I))
PRINT 163, (M1(I), M2(I), I = 1, I2)
170 CONTINUE
C
C
C
IBEAR
B
IF (CODEW = 000400000000) 6969, 200, 168
168 PRINT 10, ((I, I = 1, 16), (J, J = 1, 16)), (I, (IBEAR(I, J) = 1, 16)),
1 NUMBER (I + 32), (IBEAR(I + 32, J) = 1, 16) I = 1, 32
10 FORMAT (6H1, IBEAR, 1/16, 1/13, 1/16, 15, 13, 13, 13)
200 CONTINUE
C
C
C
MOVE
B
IF (CODEW = 001000000000) 6969, 201, 210
210 PRINT 94
94 FORMAT (12HMOVE, TABLE, *)
DO 11 I = 1, 32
11 IF (LOC(I)) 12, 11, 12
12 M = IBEAR(I)
N = IEND(I)
DO 13 J = M, N
K = J - M + 1
IF (MOVE(J)) 110, 111, 110
111 M1(K) = 0
M2(K) = 0
GO TO 13
110 M1(K) = X SIGNF (XMV1F(MOVE(J)), MOVE(J))
M2(K) = XMVFZF(MOVE(J))
13 CONTINUE
K3 = XMINOF(28, N - M + 1)
WRITE OUTPUT TAPE 2 * 15, I, (M1(L), L = 1, K3)
15 FORMAT (23H MOVE OF PIECE NUMBER, 12/100, 2814))
WRITE OUTPUT TAPE 2 * 16, (M2(L), L = 1, K3)
16 FORMAT (1H0, 2814)
IF (N - M + 1 < 1000000000) 113, 11, 113
113 K3 = N - M + 1
WRITE OUTPUT TAPE 2 * 16, (M1(L), L = 29, K3)
WRITE OUTPUT TAPE 2 * 16, (M2(L), L = 29, K3)
11 CONTINUE
201 CONTINUE
C
C
C
10000 PRINTS HISTORY
B
IF (CODEW = 010000000000) 6969, 350, 310
310 N = 2
IF (CODEW) 311, 312, 311
311 N = 100
312 IF(NMOVES) 350,390,313
313 REWIND 6
WRITE OUTPUT TAPE N, 322
322 FORMAT (31H1LEVEL OPPONENT MACHINE,10X,19HPRINCIPAL VARIATION)
   DO 320 N=1,NMOVES
   WRITE OUTPUT TAPE N, 321, 198, (M1(I), I=1, N+50), (I=1, 99)
320 FORMAT (1H0, 15Z(2X, 2A6), 2X, 14A6/(35X, 14A6))
320 CONTINUE

600 RETURN
6969 PRINT 6970
6970 FORMAT (47HCLOSE. LOGIC OF PROGRAM MAKES THIS IMPOSSIBLE. )
   GO TO 600
END
EN

* LABEL
* FAP
COUNT 55
* FTNBOL BINARY LOADER
* LOADS COLUMN ABSOLUTE FROM TAPE A2.
REM 0056 SYM. CARDS DIST. 535 RCV. 12-03-38CORR. OF DIST. 52711
* WD BTU2, BINARY TAPE UPPER LOADER
*
ENTRY FTNBOL
L TAPENO A2
FTNBOL TFL #+1
SXA TR2+1
SXA TR2+1,2
AXT AXT 1,2
CLEAR CLM
RTBL
RCHL 10CT
LCHL TXH
TFL TR3
LDQ CW
TQP #+2
TR3 CALL EXIT
LGL 6
ALS 3
LGL 6
ARS 3
LGL 12
SLW READ
RCHL READ
STA TR1
PDC LDC READ,1
STQ READ
TRAN TNX TR2+1
CLA CW
LGR 12
TCOL *
TXI #+1,1,1
TR1 ACL **,1
TXH TXH **-2,1
FOLD LDQ EOF
LGR 24
ALS 24
STQ CW
ACL CW
ZET CW
TRA FOLD
TRCL NG
ERA READ
ZET READ
TNZ NG
TRA AXT
NG TIX TR3,2,2
BSRL
READ  PZE      CLEAR,2,1
IOCT IOCT    CW,0,1
EOF   HTR      AXT
CW    PZE
TRZ   AXT **,1
      AXT **,2
      TRA 1,4
      END
* LABEL
* FAP
COUNT 270
*MISPX BUGGERED VERSION OF MISP (SPH), (SPHM), (STH), (STHM), (SCH), AND (SCHM). THIS VERSION RECOGNIZES TAPE 100 AS MEANING
WRITE ON TAPE 2, AND PRINT ON LINE.
ENTRY (SPH)
ENTRY (SPHM)
ENTRY (STH)
ENTRY (STHM)
ENTRY (STHD)
ENTRY (SCH)
ENTRY (SCHM)
REM
(PRCT) EQU 88
(PECT) EQU 89
(ELCT) EQU 90
(LNCT) EQU 97
PUNSW. EQU 4
PRNWS. EQU 5
ON LINE PUNCH SWITCH
ON LINE PRINT SWITCH
(REM
(SPHM) CAL =02000000
(SPHM)=WRITE OUTPUT TAPE 2
(STHM) STL MONSW.
SET SWITCH FOR MONITOR CONTROL
CAS =100017
CHECK FOR TAPE 100
TRA ++2
STZ ONSW
NOT ON LINE SWITCH
(BOTH) BOTH
BOOTH ON AND OFF LINE
PROC SLW UNIT
SAVE LOGICAL TAPE NO.
(LOH) LDQ ++2
LOAD MU WITH OUTPUT SWITCH + RETURN ADDRESS
TRA $ (IOH)
GO TO (IOH)
(REM
BOTH STL ONSW
SET ON LINE SWITCH
CAL #2017
MAKE LIKE TAPE 2
TRA PROC
(REM
(SCHM) STL MONSW.
INDICATE MONITOR CONTROL
SWT PUNSW.
IS ON LINE PUNCH SWITCH DOWN
TRA (STH3) NO; WRITE LOGICAL TAPE 3 (PUNCH TAPE)
(REM
(SCH) CLA M2E2
YES; SET UP TO PUNCH ON LINE ONLY
LDQ ++2
***
TRA* $ (IOH)
***
TRA SCH
OUTPUT SWITCH AND RETURN ADDRESS
(REM
(STH3) CAL =03000000
LOGICAL TAPE NO. FOR PUNCH TAPE
SLW UNIT
INSURE NO ON LINE PRINTING
LDQ ++2
SET UP TO WRITE PUNCH TAPE
TRA* $ (IOH)
***
TRA STH3
OUTPUT SWITCH AND RETURN ADDRESS
(REM
(STHD) LDQ ++2
LOAD MU WITH OUTPUT SWITCH + RETURN ADDRESS
TRA* (IOH)
GO TO (IOH)
TRA STHD
**
CALL FOR PRINTER ONLY (WITHOUT MONITOR)
LOAD MQ WITH OUTPUT SWITCH + RETURN ADDRESS
GO TO (IOH)

SAVE RETURN INDEX TO (IOH)
UPDATE COUNT OF RECORDS ON PUNCH TAPE

ESTIMATED PUNCHED OUTPUT COUNT
TEST FOR PUNCH COUNT EXCEEDED
HERE WHEN PUNCH COUNT ESTIMATE EXCEEDED
MARK (PUCT) FOR SIGN ON

TERM/NATE THIS JOB

SAVE RETURN INDEX TO (IOH)
SAVE INDICATORS
INSURE NO ON LINE PRINTING BLANKS

CHECK THAT LINE IS NON-ZERO AND NON-BLANK OK, WRITE THIS LINE
HERE FOR BLANK OR ZERO LINE
SO SKIP WRITING
COUNTS DEBUG LINES

NORMAL OUTPUT LINE, RETURN FROM (IOH)
IS THIS A MONITOR JOB
NO, SKIP TO WRITE

YES, SO UPDATE TOTAL LINE COUNT
COUNT PROGRAMMER OUTPUT

ESTIMATED PRINTED OUTPUT COUNT
TEST FOR LINE COUNT EXCEEDED
HERE WHEN LINE COUNT ESTIMATE EXCEEDED
MARK (PRCT) FOR SIGN ON

TERM/NATE THIS JOB
REM  TES  TSX $ (WER) * 4  CHECK ANY PREVIOUS WRITE
LXA  STIX * 4  RESTORE CALL INDEX
CAL  1 * 4  CALL = PLE FIRST, *N
ARS  18
ACL  1 * 4
STA  MOVE * 4
STD  STHC * 4  WORD COUNT INTO OUTPUT COMMAND
PDX  0 * 4  AND IR4
TIX * + 1, 4 * OUTPUT
SXA  MOVE * 1, 4
PDX  0 * 4  RESTORE WORD COUNT
MOVE * 1
CAL * + 4  MOVE DATA TO OUTPUT BUFFER
SLW  * + 4  **
TIX  MOVE * 4, 1  **
CAL  TES  SET UP ERROR CHECKING
SLW  * (TES)  **
AXC  STHC * 4  ADDRESS OF I/O COMMAND
PXA  0 * 4  **
STA *  $ (WTC)  SAVE IN CASE OF ERROR
XEC * $ (WRSC)  SELECT OUTPUT TAPE
XEC * $ (RCH)  WRITE OUT THIS RECORD
STHX  AXI * + 4  RESTORE RETURN INDEX
NZT  MONSEW  *  IS THIS A MONITOR JOB
TRA  2 * 4  NO; RETURN TO (I0H)
CLA  UNIT *  IS THIS THE MONITOR STACKED OUTPUT TAPE
SUB  = 02000000  **
TNZ  2 * 4  NO; RETURN TO (I0H)
ZET  ONSW  CHECK TO SEE IF TAPE WAS 100
TRA  * + 3  YES, PRINT ON LINE
SWI  PRNSW  *  IS THE ON LINE PRINT SWITCH ON
TRA  2 * 4  NO; RETURN TO (I0H)
CAL (PRCT)  YES, PRINT THIS ON LINE
ADD  = 01000000  UPDATE ONLINE PRINT COUNT
STD (PRCT)  **
TRA  SPC  GO TO ONLINE PRINT ROUTINE
REM

SCH  NZT  MONSEW *  ON LINE PUNCH ROUTINE
TRA  * + 4  SKIP UPDATE OF (PUCT) IF NOT IN MONITOR
CAL (PUCT)  OTHERWISE UPDATE (PUCT)
ADD  = 01000000  **
STD (PUCT)  **
SXA  NPIR1, 1  SAVE IR1
LDQ  WPUA *  PICK UP ON LINE PUNCH SELECT
CAL  PNP  PICK UP NOP TO AVOID SPACE CONTROL
AXT  12 / 1  PICK UP MAX WORD COUNT FOR ON LINE PUNCH
TRA  PRPUN *  GO TO BCD TO CARD IMAGE CONVERTER
REM

SPH  SXA  NPIR1, 1  ON LINE PRINT ROUTINE
LDQ *  1 *  PICK UP FIRST BCD WORD
PXD
LGL  6  GET FIRST CHARACTER OF LINE
PAX  0, 1  SAVE IT IN IR1
CAL  060  REPLACE WITH A BLANK
LGR:  6
STQ*: 1,4
PXA  0,1
AXT  ESPTB- B$PTB,1
CAS  ESPTB,1
TRA  *+2
TRA  SPFND
TIX  *+3,1,2
CAL  NPNOP
TRA  SPFND+1

SPFND
CAL  ESPTB+1,1
LDO  WPRB
AXT  20,1
REM

PRPUN, SLW  NPSPR
STQ  WRSAM
SXD  TSTCT,1
CAL  1,4
PDX  0,1
TSTCT
TAL  *+2,1,**
LXD  TSTCT+1
PXA  0,1
STA  NPs5V4
ACL  1,4
STA  NPRC3
SXA  NPIR2,2
SXA  NPIR4+4
LXA  NPs5V4,4
TAL  1PASS+4,12
STL  2PswT
REM

1PASS
AXT  24,1
STZ  PBUFF+24,1
TIX  *+1,1,1
AXT  1,2
NPRC1
CAL  COLIND
NPRC2
SLW  PRCOL
SXA  NPs5V4,4
NPRC3
LDO  **4
NPRC4
AXT  6+4
PXD
LGL  6
ALS  1
PAX  0,1
CAL  PRCOL
ARS  6+4
TAL  PDIGIT,1,24
TAL  PNZONE,1,95
TAL  NPRC5+1,96
REM

PNZONE
TXH  PNNMIN,1,62
ORS  PBUFF+23,2
TIX  PULigit,1,32
TRA  NPRC5

FIRST CHARACTER IS CONTROL CHARACTER
LOOK FOR THIS CHARACTER IN TABLE

FOUND, GO TO PICK UP SPRA INST.

NOT FOUND, SET FOR SINGLE SPACE

PICK UP SPRA FOR SPACE CONTROL
PICK UP ON LINE PRINTER SELECT
PICK UP MAX, WORD COUNT FOR ON LINE PRINTER

SET SPACE CONTROL IF ANY
SET ON LINE UNIT SELECT
SET MAX, WORD COUNT
CALL = PCE FIRSTN
WORD COUNT TO 1R1
SKIP IF WORD COUNT OK
WORD COUNT TOO LARGE, SET TO MAX.
SAVE WORD COUNT
FIRSTN
SAVE 1RS

RESTORE WORD COUNT
IS SECOND PASS NEEDED
YES, SET SWITCH FOR 2 PASSES
CLEAR WORKING STORAGE
SET FOR LEFT HALF OF CARD IMAGE
INITIALIZE COLUMN MARKER

SAVE WORD COUNT
PICK UP FIRST OR NEAT BCD WORD
SET CHARACTER COUNT
GET A CHARACTER
DOUBLE IT
INTO 1R1

POSITION COLUMN MARKER
SKIP IF DIGIT ONLY
SKIP IF BLANK
SKIP IF 11 OR O ZONE
UP IN THE 1Z ZONE
REMOVE 1Z PUNCH
SKIP IF + ONLY (NO DIGIT)
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNMIN TXH</td>
<td>PNZER +1.94</td>
</tr>
<tr>
<td>ORS PBUFF+21.2</td>
<td>OR IN THE 11 ZONE</td>
</tr>
<tr>
<td>TIX PDIGIT +1.64</td>
<td>REMOVE II ZONE</td>
</tr>
<tr>
<td>TRA NPRC5</td>
<td>SKIP IN ONLY (NO DIGIT)</td>
</tr>
<tr>
<td>PNZER ORS PBUFF +19.2</td>
<td>OR IN THE 0 ZONE</td>
</tr>
<tr>
<td>TXI PDIGIT +1.76</td>
<td>REMOVE 0 ZONE</td>
</tr>
<tr>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>PNDIG TXL</td>
<td>PN1G +1.18</td>
</tr>
<tr>
<td>ORS PBUFF +3.2</td>
<td>HERE FOR 8-3, 8-4, OR IN THE 8 PUNCH</td>
</tr>
<tr>
<td>TXI +1.1-16</td>
<td>REMOVE THE 8 PUNCH</td>
</tr>
<tr>
<td>NPSVR AXT</td>
<td>**.4</td>
</tr>
<tr>
<td>TIX NPRC4 +4.1</td>
<td>COUNTS CHARACTERS</td>
</tr>
<tr>
<td>ARS</td>
<td>1</td>
</tr>
<tr>
<td>NPSV4 AXT</td>
<td>**.4</td>
</tr>
<tr>
<td>TIX PN1OW +4.1</td>
<td></td>
</tr>
<tr>
<td>TZE PNT1ST</td>
<td></td>
</tr>
<tr>
<td>TRA NPRC2</td>
<td></td>
</tr>
<tr>
<td>PNT1ST TXL</td>
<td>PN1OW +2.0</td>
</tr>
<tr>
<td>AX1 0.*2</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>PNRC1</td>
</tr>
<tr>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>PN1OW TCOA</td>
<td>*</td>
</tr>
<tr>
<td>AX1</td>
<td>24.1</td>
</tr>
<tr>
<td>CAL</td>
<td>PB1UFF +24.1</td>
</tr>
<tr>
<td>SLW</td>
<td>PB1UFF +24.1</td>
</tr>
<tr>
<td>TIX</td>
<td>+2.1+1</td>
</tr>
<tr>
<td>W1RSA WR1SA</td>
<td>**</td>
</tr>
<tr>
<td>RC1HA</td>
<td>NPR1OC</td>
</tr>
<tr>
<td>NPSVR PSE</td>
<td>**</td>
</tr>
<tr>
<td>NZ1 PT1W</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>NP1R1</td>
</tr>
<tr>
<td>STZ</td>
<td>2PSW1T</td>
</tr>
<tr>
<td>CAL</td>
<td>PSPR9</td>
</tr>
<tr>
<td>SLW</td>
<td>NPSVR</td>
</tr>
<tr>
<td>TRA</td>
<td>1PASS</td>
</tr>
<tr>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>NP1R1 AX1</td>
<td>**.1</td>
</tr>
<tr>
<td>NP1R2 AX1</td>
<td>**.2</td>
</tr>
<tr>
<td>NP1R4 AX1</td>
<td>**.4</td>
</tr>
<tr>
<td>TRA</td>
<td>2.*4</td>
</tr>
<tr>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>B1SPTB BCI</td>
<td>1.000000</td>
</tr>
<tr>
<td>SPRA</td>
<td>4</td>
</tr>
<tr>
<td>BCI</td>
<td>1.000001</td>
</tr>
<tr>
<td>SPRA</td>
<td>1</td>
</tr>
<tr>
<td>BCI</td>
<td>1.000002</td>
</tr>
<tr>
<td>SPRA</td>
<td>2</td>
</tr>
<tr>
<td>BCI</td>
<td>1.00000+</td>
</tr>
<tr>
<td>SPRA</td>
<td>5</td>
</tr>
<tr>
<td>ESPTB SYN</td>
<td>*</td>
</tr>
<tr>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>ONSW PZE</td>
<td></td>
</tr>
<tr>
<td>MONSW PZE</td>
<td></td>
</tr>
<tr>
<td>2PSW1T PZE</td>
<td></td>
</tr>
</tbody>
</table>
UNIT.   PZE
SIND.   PZE
PRCOL   PZE
COLIND  MZE
MZE2    MZE  ,,2
MZE3    MZE  ,,3
NPNOP   NOP
PSPR9   SPRA  9
WPRA.   WPRA
WPUA.   WPUA
NPIOC   IOCD  PBUFI   ,,24
STHC    IOST  OUTPUT,,**
OUTPUT  BSS   22
PBUFI.  BSS   24
REM
COMMON  -176
REC      COMMON  76
PBUFF   COMMON  1
END
* LABEL
* LIST8
SUBROUTINE BOARD (ITAPE)
C PRINTS OUT CHESS BOARD IN READABLE FORMAT.
DIMENSION FOO(5000), PIECES(40), TAB1(8), TAB2(8), KIND(32),
   10CC(64)
COMMON FOO
EQUIVALENCE (FOO(2938), PIECES), (FOO(1527), 10CC), (FOO(1463),
   KIND)
WRITE OUTPUT TAPE ITAPE,*6
6 FORMAT (1H,*18X,*5HBLACK/1H,*18X,*5H-----)
   DO 1 1 = 1,57,8
   DO 10 J = 1, 8
   L = XGETF (J + 5/ - 1; 10CC)
   IF (X RANGEF (L, 7, 22)) 7,9,
9 IF (KIND(L) - 1) 6,7,8
8 L = KIND(L) + 2*(XBITF(L)) + 31
B7 TAB1(J) = PIECES (L+1)
B10 TAB2(J) = SHIFTF(PIECES(L+1), 22)
WRITE OUTPUT TAPE ITAPE,*3
3 FORMAT (42H ***************H +H +H +H +H +H +H +H *H)
1 WRITE OUTPUT TAPE ITAPE,*4, TAB1, TAB2
4 FORMAT (1H,*8(2H* ,A3),1H*/1H,*8(2H* ,A3),1H*)
   WRITE OUTPUT TAPE ITAPE,*3
   WRITE OUTPUT TAPE ITAPE,*5
5 FORMAT (1H,*18X,*5HWHITE)
RETURN
END
CARDS ROW

FAP

COUNT 20

MISTOP

FUL

ORG -11

IOCD C,11

TCOA 1

PZE

REM MAIN PROGRAM STARTS HERE

C

AXT *1

A

CAL C,1

ADD B

D

SLW C,1

LGR 37

TOP C

TIX A,1,1

B

HTR 1

REM END OF MAIN PROGRAM

PZE

TXI D,1,C-1

END
* LABLE
* FAP
COUNT 3D
* WRITE FOR PRTREE
ENTRY WRITE
WRITE SXD WRITE-2,4
CLA* 1,4
LGR 18
ALS 15
LGL 3
SLW MOVE
CLA* 2,4
LGR 19
ALS 6
TQP #+2
ADD =3
ORA =HI 00
SLW FMT
CALL JUNPAK,MOVEA,B
TSX $(SPH),4
PZE FMT,=1
LDQ A
STR
LDQ B
STR
TSX $(FIL),4
LKD WRITE-2,4
TRA 3,4
FMT PZE
BCI 1,X,2A6
A
B
MOVE END
* LABEL
* FAP
COUNT 8
* KEYS SETS AC TO ADDRESS OF KEYS (IN DEC.) AND VARIABLE TO DEC.
ENTRY KEYS
KEYS ENK
SLG* 1,4
LLS 35+18+2
TRA 2,4
END
* LABEL
* FAP
*BEGIN INITIALIZING ROUTINE, APR. 19, 1962
COUNT 88
ENTRY BEGIN
ENTRY RECOUP
ENTRY LDUMP
PMRST EQU 63
BEGIN SXA DONE,4
CAL =6B17
TSX $(RWT),4
CAL =7B17
TSX $(RWT),4
CALL FTNBOL
CALL STOMAP
CAL A
SLW PMRST
STZ NLOG
STZ MLOG
STZ IPRINT
STZ MOVES
STZ NMOVES
TSX $(TMLFT),4
TXH AC1
CLA AC1
SUB =900
STO AC1
TSX STIMER,4
TXH AC1
TXH TIMEOUT
DONE AXT **,4
TRA 1+,4
TIMEOUT CAL =100617
TSX $(SHT),4
TSX TIMFMT
TSX $(FIL),4
CALL CLOCK,D2
CALL PRINT,N
LAC 6+,4
SXA PMRST-1,4
CLA $(F2PM)
STA 6
TRA $RSTRTN
A TTR **+1
LTM
SXA XR4,4
AXT FMT,4
C SXA B+,4
STG MQ
SLW AC1
ARS 2
STO AC2
CAL =100617
TSX $(SHT),4
B PZE ** -1
TSX $(FILE)*4
CALL CLOCK,D2
CALL PRINT,N
XR4 AXT **,4
LDO MQ
CLA AC2
ALS 2
ORA AC1
TRA* $(F2PM)
RECOUP SXA XR4*,4
LAC XR4*,4
SXAR PMRST-1*,4
AXT FMT1*,4
TRA C
LDUMP SXA XR4*,4
LAC XR4*,4
SXAR PMRST-1*,4
LXD LDMPF*,4
LDMPF TXI C*,FMT2
N OCT 7777400000
D2 DEC 2817
AC1
AC2
MO
TIMFMT BCI 2,(8H1TIMEOUT)
FMT BCI 6,(28H1PROGRAM MANUALLY RESTARTED.)
FMT1 BCI 4,(16H1RECOUP REACHED.)
FMT2 BCI 4,(15H1LDUMP REACHED.)
COMMON 12561
R COMMON 1
NLOG EQU R+12428
MLOG EQU R+9443
IPRINT EQU R+3375
NMOVES EQU R+3245
MOVES SYN R+3246
END
* * *
* LABEL
* FAP
* FUNCTION LOOK(SQUARE,DIRECTION)
* COUNT 28
* GIVES FIRST OCCUPIED SQUARE IN GIVEN DIRECTION, OR ZERO.
* ENTRY LOOK
LOOK
  SXA XR1,1
  SXA XR1+1,2
  CLA* 2,4
  PDX ,2
  CLA* 1,4
  SUB =1817
  PDX ,1
LOOP
  CLA IEXTD+1,2       FIND NEXT SQUARE
  ADD IEXTS,1
  ANA =020177000000
  PDX ,1
  TXH NOSQ,1,63
  ZET IOC C,1
  TRA FOUND
  TXL LOOP,2,8
FOUND
  TXL NOSQ,1,63
  ZET IOC C,1
  TRA FOUND
  TXL LOOP,2,8
NOSQ
  CLS =1817
  ADD =1817
  XRT **,1
  XRT **,2
  TRA 3,4
  STORAGE ALLOCATION
  COMMON 12961
R COMMON 1
IEXTS SYN R+11201
IEXTD SYN R+11197
IOCC SYN R+11035
END
* LABEL
* FAP
*XTIME WITH INTERVAL TIMER
COUNT 15
ENTRY XTIME
ENTRY XLAPSE
XTIME TRA $RSCLOCK
XLAPSE SXA XIT,4
CALL STOPCL,1
PXD
LDQ I
DVP =360817
XCA
ALS 18
XIT AXT **,4
TRA 1,4
I PZE
END
* LABEL
* FAP
COUNT 80
* JUNPAK TRANSLATES MOVES, XFILE GIVES FILES. FEB 20, 1961
ENTRY JUNPAK
ENTRY XFILE
XFILE SUB = 1b17
ANA = 7b17
ADD = 1b17
TRA 1, 4
JUNPAK SXR 4, 4
CLA* 1, 4
STO T1
TZE ZERO
TMI ZERO
TSX $XMV 3, 4
PDX , 4
ANA = 1b17
STO COLOR
TXL B 4, 6
TXH B 4, 22
CLA KIND + 1, 4
SUB = 1b17
TZE B
PDX , 4
TXI B 4, 32
B
CAL PIECES , 4
ANA = 0777777400000
ARS 12
ZET COLOR
ACL = H040000
ACL = H0*0000
A
SLW ANS
CLA T1
TSX $XMV 1, 4
STO SQUARE
TSX XFILE , 4
PDX , 4
LDQ FILES + 1, 4
CAL ANS
LGL 6
SLW ANS
STQ ANS2
CLA SQUARE
LDQ COLOR
TSX $XTRANK , 4
ALS 6
ORS ANS2
CLA T1
TSX $XADD , 4
TZE PKUP
PDX , 4
CAL PIECES - 31 , 4
ARS 24
ALS       6
ORA       =HOG(00)
ORA       ANS2
XR4       AXT  **,4
SLW*      3*,4
CAL       ANS
SLW*      2*,4
TRA       4*,4
ZERO      PDX  *,4
CAL       SPEC*,4
SLW       ANS
CAL       =H
TRA       XR4
PKUP      CAL  =H00
TRA       XR4-1
T1        SYN  XFILE-2
COLOR     PZE
ANS       PZE
ANSZ      PZE
SQUARE    PZE
BCI       3, SETUP BLACK WHITE
SPEC      BCI  1,REVERT
ZILCH     COMMON 12561
R         COMMON 1
PIECES    SYN  R+9624
KIND      SYN  R+11099
FILES     SYN  R+9519
END
* LABEL
* FAP
COUNT 152
* CHESS ROUTINES IN FAP, RE-ASSEMBLED FOR 709, A. KOTOK
ENTRY XLBIT
ENTRY XMOM
ENTRY XRANK
ENTRY XTRANK
ENTRY XDEL
ENTRY XMV1
ENTRY XMV2
ENTRY XMV3
ENTRY XBAND
ENTRY XBOR
ENTRY XBEOR
ENTRY XBNOT
ENTRY STO
ENTRY XSTO
ENTRY GET
ENTRY XGET
ENTRY XAND
ENTRY XOR
ENTRY XLESS
ENTRY XNOT
ENTRY XONE
ENTRY XRANGE
ENTRY XADD
ENTRY XDEC
ENTRY XPRE
ENTRY XTAG

XLBIT LDQ A1
STQ 0,4
TRA 0,4
A1 ANA =1B17
XMOV ANA M2
ADD =1B17
TRA 1,4
XDEC LDQ A33
TRA XLBIT+1
XPRE XCA
LGL 18
XTAG ALS 3
ANA =7017
TRA 1,4
SUBT SSM
ADD =65B17
TRA XRANK
XTRANK RQL 17
TOP SUBT
XRANK SUB =1B17
ARS 3
ADD =1B17
A33 ANA =077777000000
TRA 1,4
XDEL LDG A2
   TRA XLBIT+1
A2 ANA =0177/000000
XM1 ANA =63B17
   TRA XMOV+1
XM2 ARS 6
   ANA =13B17
   TRA XMOV+1
XM3 ARS 10
   ANA =31B17
   TRA XMOV+1
XADD ANA =077777
   ALS 18
   TRA 1,4
M2 REM 127+8*1024
   TRA
XBAND STQ T1 M
   ANA T1 L&M
   TRA 1,4 EXIT
REM XBANDF(L,M) GIVES L AND M
XBOR STQ T1 M
   ORA T1 L+M
   TRA 1,4 EXIT
REM XBORF(L,M) GIVES L-INCLUSIVE-OR-M
XBEOR STQ T1 M
   ERA T1
   TRA 1,4
XBNOT LDQ A3
   TRA XLBIT+1
A3 ERA =077777000000
   TEMPOORARY STORAGE
T1 REM STOF AND XSTUF
REM STORES X IN A(J) BY CHANGING THE INSTRUCTIONS IN THE PROG
XSTO BSS 0
STO STO T1
   CLA -1,4
   TPL LDQ
REM PREVIOUS INSTRUCTION WAS AN SXD. MOVE IT BACK ONE INSTR
LDQ -1,4
CAL -2,4
STQ -2,4
REM CHANGE LOC(U,4) TO A STO A+1,4
LDQ ANA =077777
   ADD =1
   ANA =077777
   ORA STOR
SLW 0,4
REM CHANGE PREVIOUS INSTRUCTION TO AN LXD -3,4 WHERE J STORED
CAL LXJ
SLW -1,4
   CLA T1
   TRA -1,4
STOR STO 0,4
LXD LXJ A+4
A SYN -3
XAND STO T1
SSP
ADM T1
TRA 1,4
XOR TZE 1,4
XCA
TRA 1,4
XLESS STO T1
SUB T1
TZE 1,4
CHS
LRS 0
PXO
LGK 1
ALS 18
TRA 1,4
XNOT TZE NOSAT
PXO
TRA 1,4
XONE SSP
TZE 1,4
TRA NOSAT
XRANGE TNZ **+2 FIX SIGN OF 0
SSP
STO T1
CAS T1
NOR
TRA **+3 LOWER RANGE IS SATISFIED
NOSAT CLA =1B17
TRA 1,4
TNZ **+2 FIX SIGN OF 0
SSM
CAS -3
TRA NOSAT
NOR
PXO
TRA 1,4
REM GET AND XGET
REM GET ALLOWS USE OF ILLEGAL SUBSCRIPTS IN FORTRAN
XGET BSS 0
GET STO T1
CLA -1,4
TPL LDQA
CLA -2,4
LDQ -1,4
STQ -2,4
LDQA ANA =077777
ADD =1
ANA =077777
ORA CLA
SLW 0,4
CLA PDX
STO -1,4
<table>
<thead>
<tr>
<th>CLA</th>
<th>T1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRA</td>
<td>-1.4</td>
</tr>
<tr>
<td>CLA</td>
<td>0.4</td>
</tr>
<tr>
<td>PDX</td>
<td>0.4</td>
</tr>
<tr>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>
THE FUNCTION VALUE IS +1 IF THE MOVE IS A CHECK, 0 IF THE MOVE CANNOT BE A CHECK, AND -1 IF THE MOVE MAY BE A CHECK.

DIMENSION AND EQUIVALENCE STATEMENTS
COMMON AA
DIMENSION AA(4500), KIND(32), LOC(32), IOCC(64), NEP(10), MEP1(10),
1 MEP2(10), IBEGIN(32), IEND(32), LEGAL(32)
DIMENSION IEXTS(64), IEXTD(15)
EQUIVALENCE (AA(2765), MAVAIL(1)), (AA(2892), K1(1)),
1(AA(1463), KIND(1)), (AA(1591), LOC(1)), (AA(1527), IOCC(1)),
2(AA(2900), MCOL(1)), (AA(2914), NUMEP(1)), (AA(2745), MEP1(1)),
3(AA(2755), MEP2(1)), (AA(1411), IBEGIN(1)), (AA(1495), IEND(1)),
4(AA(2923), LEGAL(1))
EQUIVALENCE (AA(2735), NEP(1))
EQUIVALENCE (AA(1301), IEXTS), (AA(1365), IEXTD)
EQUIVALENCE (AA(2903), MOVENO)

MAIN PROGRAM

M=MV
MVER=XMOVF(M)
MVDIR=XMOVF(M)
MVTO=XMOVF(M)
MVFR=LOC(MVER)
MVKIND=KIND(MVER)
KLOC=XGETF(3, MCOL, LOC)
IOCC(MVFR)=0
I8=IOCC(MVTO)
IOCC(MVTO)=MVER

IS THIS AN EN PASSANT MOVE
IF (NEP(NUMEP)-MOVENO) 8, 9, 8

IS THIS A CASTLING MOVE
8 IF (XANDF(MVKIND-5, MABS(MVFR-MVTO)-1)) 2, 3, 2

MOVE INVOLVES CASTLING. FIND ROOK LOCATION
3 IF (MVDIR=2) 4, 5, 6

I1=XMOVF(IEXTS(MVTO)+IEXTD(1))
GO TO 7

I1=XMOVF(IEXTS(MVTO)+IEXTD(3)+IEXTD(2))

I10CC=I10CC(11)
I2=XMOVF(IEXTS(MVFR)+IEXTD(MVDIR))

11=OLD ROOK SQUARE, 12=NEW ROOK SQUARE

MOVE PIECES
10CC(12)=I10CC
IOCC(11)=0

IS THE KING IN CHECK
ISCHEK=MABLE(I2, KLOC)

RESTORE POSITION OF KING AND ROOK
12 IOCC(11)=I10CC
IOCC(12) = 0
IOCC(MVFR) = MVFR
IOCC(MVTO) = 0
RETURN

C   IS A PAWN PROMOTING
2   IPROM = 0
16   KIND(MVER) = XADD(FM)
15   IPROM = 1

C   SEE IF MOVER IS CHECKING
15   ISCHEK = MABEL(MVTO, KLOC)
IF (ISCHEK) 5, 17, 21

C   IS THERE A DISCOVERED CHECK
17   I4 = LOOK(KLOC, NORIEH(KLOC, MVFR))
IF (I4) 5, 19, 24
24   I5 = IOCC(I4)
IF (XLBITF(MVER)) 5, 25, 19
25   IF (XABS(FKIND(I5) - 3) - 2) 20, 19, 20
20   ISCHEK = MABEL(I4, KLOC)
GO TO 21

C   MOVE IS NOT A CHECK
19   ISCHEK = 0
GO TO 21

C   MOVE MAY BE CHECK
5   ISCHEK = -1
21   IF (IPROM) 23, 23, 22
22   KIND(MVER) = 1
23   IOCC(MVTO) = 18
IOCC(MVFR) = MVFR
RETURN
END
* LABEL
* LIST8
C JAN 14, 1961
FUNCTION MABLE(MSQ1,MSQ2)
C
THE VALUE OF MABLE IS 1 IF THE PIECE AT MSQ1 CAN CAPTURE
A PIECE AT MSQ2, AND 0 OTHERWISE. CHECKS ARE IGNORED.
C
DIMENSION AND EQUIVALENCE STATEMENTS
COMMON AA
DIMENSION AA(4500)
DIMENSION MAVAIL(100),KIND(32),LOC(32),IOCC(64),NEP(10),MEP1(10),
MEP2(10),IBEG(33),IEND(32),LEGAL(5,3)
DIMENSION IOPP(16)
EQUIVALENCE (AA(2765),MAVAIL(1)),(AA(2892),K1(1)),
1(AA(1463),KIND(1)),(AA(1591),LOC(1)),(AA(1527),IOCC(1)),
2(AA(2900),MCOL(1)),(AA(1514),NUMEP(1)),(AA(2745),MEP1(1)),
3(AA(2755),MEP2(1)),(AA(1491),IBEG(1)),(AA(1495),IEND(1)),
4(AA(2923),LEGAL(1))
EQUIVALENCE (AA(2735),NEP(1))
EQUIVALENCE (AA(1285),IOPP)
C
MAIN PROGRAM
1 M1=MSQ1
M2=MSQ2
MP=IOCC(M1)
K=KIND(MP)
IDIR=NORIEN(M1,M2)
C CHECK WHETHER PIECES ARE IN LINE
5 IF (IDIR) 2,2,3
3 IF (LOOK(M2,IOPP(IDIR))-M1) 2,4,2
C PIECES ARE IN LINE
4 IF (K-1) 10,9,10
C CHECK PAWN DIRECTIONS
9 I1=IDIR+IDIR-1
I2=XLBITF(MP)
IF (XMINOF(XABSF(I1+2)-I2,XABSF(I1-2)-I1+2)) 2,11,2
C IS THIS A LEGAL MOVE DIRECTION FOR THE GIVEN PIECE
10 I1=XMINOF((IDIR+3)/4,3)
MABLE=LEGAL(K-1,I1)
IF (MABLE) 7,7,6
6 IF (K-5) 7,8,7
C PAWNS AND KINGS CAN ONLY MOVE ONE SQUARE
11 MABLE=1
8 I2=XABSF(M1-M2)
IF (XMINOF(I2-1,XABSF(I2-5)-1)) 7,7,2
2 MABLE=0
7 RETURN
END
* LABEL
* FAP
* COUNT 100
* NORIEN, RECOMPILED FOR 709 A. KOTOK
REM FUNCTION NORIEN(MFROM>MTO)
REM ROUTINE TO FIND DIRECTION FROM MFROM TO MTO
ENTRY NORIEN
NORIEN CLA* 2*4
SUB* 1*4 IS DIRECTION VERTICAL
TZE 3*4 EXIT IF FROM=TO
STO T1
ANA =7B17
TNZ NOVT IF NOT VERTICAL, TRANSFER
CLA T1
TMI *=3
CLA =2B17 DIRECTION 4
TRA 3*4
CLA =4B17
TRA 3*4
NOVT CLA* 1*4
SUB* 1*4 FILE OF FIRST SQUARE
STO T1
ANA =7B17
STO VF
CLA T1
ANA =56B17
ARS 3
STO HF RANK OF 1ST SQUARE
CLA* 2*4
SUB* 1*4
STO T1
ANA =7B17
STO VT
STO VF
STO WD
CLA T1
ANA =56B17
ARS 3
STO HT RANK OF 2ND SQUARE
SUB HF HORIZONTAL DIFFERENCE
STO HD DIRECTION NOT HORIZONTAL
TNZ NOHOR
CLA WD
TMI *=3
CLA =1B17 DIRECTION 1
TRA 3*4
CLA =3B17 DIRECTION 3
TRA 3*4
NOHOR CLA WD
SSP
SBM HD NOT DIAGONAL
TNZ NODIG FIND WHICH DIAGONAL DIRECTION
PXO VD
LDQ VD
LGL  1
LDQ  HD
LGL  1
SXA  X4,4
PAX  0,4
CLA  DIAGD,4
TRA  X4
NODIG CLA  HD
SSP  VD
ADM  VD
SUB  $B17
TZE  NITE
PXD  3,4
TRA  NITE
LDQ  VD
LGL  1
LDQ  HD
LGL  20
ADM  VD
SXA  X4,4
PDX  0,4
CLA  NITED,4
X4  AXT  **,4
TRA  3,4
REM  STORAGE
   *7
   *6
   *8
DIAGD
   *5
   *13
   *14
   *12
   *11
   *16
   *15
   *9
   *10
NITED BES
T1
VF
HF
VT
VD
HT
HD
END

LOOK UP DIRECTION
CHECK FOR KNIGHT DIRECTION
NO LEGAL MOVE DIRECTION
CHECK WHICH KNIGHT DIRECTION
DIAGONAL DIRECTIONS
KNIGHT DIRECTIONS
* LABEL
* FAP
* CHESS TABLES FOR COMMON STORAGE
ABS
A EQU 1024
REM SYMBOL TABLE FOR ARRAYS
R EQU -1
IPDIR SYN 31284+R
10PP SYN 31278+R
1EXTS SYN 31262+R
1EXTD SYN 31198+R
1ENUS SYN 32530+R
1END SYN 31068+R
1BEG SYN 32562+R
1PAWN SYN 31182+R
1PROM SYN 31104+R
1KIND SYN 31100+R
M64H1 SYN 31174+R
MSTO SYN 31136+R
MSVN SYN 31158+R
NM0V SYN 31142+R
KVAL SYN 29646+R
FILES EQU 29519
KCNVAL EQU 29535
LEGAL EQU 29639
LOCBEG EQU 29581

* LOCBEG AND LEGAL TABLES FOR CHESS, JAN. 31, 1961
ORG LOCBEG-21 LOCBEG TABLE GIVES INITIAL LOCATIONS.
DEC 56B17,16B17,55B17,15B17,54B17,14B17,53B17,13B17
DEC 52B17,12B17,51B17,11B17,50B17,10B17,49B17,9B17
DEC 64B17,8B17,57B17,1B17,61B17,5B17
ORG LEGAL-14 LEGAL GIVES LEGAL MOVE DIR. TO TABLE.
DEC 0,0,0,1B17,0,1B17,1B17,0,0,1B17,1B17,0,0,1B17

* CHESS TABLES—FILES, LCNST?, KCNVAL
ORG KCNVAL-15
DEC 8B17,8817,4B17,4B17,4B17,8817,8B17,4B17
DEC 2B17,4B17,4B17,2B17,1B17,1B17,1B17,1B17
ORG FILES-7
BCI 8*KROOOGKNOOOOKB0000 K0000 W0000Q0000K00000Q0000000Q0000

* KPANV TABLE FOR IDVLOP
REM "MSVN TABLE
ORG MSVN-15
C1 BSS 0
DUP 1,16
0,0,7*16+7*C1-7**
REM IBEQ TABLE
ORG IBEQ-32
0,0,452 IBEQ(33), THE SAME AS IENUS
0,0,225+112+36+4 32
0,0,225+112+4 31
0,0,225+88 30
0,0,285 29
0,0,257 28
<table>
<thead>
<tr>
<th>Number</th>
<th>Piece</th>
</tr>
</thead>
<tbody>
<tr>
<td>000136</td>
<td>QUEEN</td>
</tr>
<tr>
<td>000132</td>
<td></td>
</tr>
<tr>
<td>000104</td>
<td></td>
</tr>
<tr>
<td>000076</td>
<td></td>
</tr>
<tr>
<td>000048</td>
<td></td>
</tr>
<tr>
<td>000020</td>
<td></td>
</tr>
<tr>
<td>000010</td>
<td></td>
</tr>
</tbody>
</table>

**REM KIND TABLE M179**

<table>
<thead>
<tr>
<th>Number</th>
<th>Piece</th>
</tr>
</thead>
<tbody>
<tr>
<td>0006</td>
<td></td>
</tr>
</tbody>
</table>

**ORG KIND-31**

<table>
<thead>
<tr>
<th>Number</th>
<th>Piece</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

**DUP**

<table>
<thead>
<tr>
<th>Number</th>
<th>Piece</th>
</tr>
</thead>
<tbody>
<tr>
<td>004</td>
<td>BISHOP</td>
</tr>
</tbody>
</table>

**DUP**

<table>
<thead>
<tr>
<th>Number</th>
<th>Piece</th>
</tr>
</thead>
<tbody>
<tr>
<td>003</td>
<td>KNIGHT</td>
</tr>
</tbody>
</table>

**DUP**

<table>
<thead>
<tr>
<th>Number</th>
<th>Piece</th>
</tr>
</thead>
<tbody>
<tr>
<td>116</td>
<td>PAWN</td>
</tr>
</tbody>
</table>

**DUP**

<table>
<thead>
<tr>
<th>Number</th>
<th>Piece</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>ROOK</td>
</tr>
</tbody>
</table>

**DUP**

<table>
<thead>
<tr>
<th>Number</th>
<th>Piece</th>
</tr>
</thead>
<tbody>
<tr>
<td>005</td>
<td>KING</td>
</tr>
</tbody>
</table>

**REM IEXTD TABLE M179**

<table>
<thead>
<tr>
<th>Number</th>
<th>Piece</th>
</tr>
</thead>
<tbody>
<tr>
<td>02A+2+15*8</td>
<td></td>
</tr>
<tr>
<td>0A+1+14*8</td>
<td></td>
</tr>
<tr>
<td>015<em>A+7+13</em>8</td>
<td></td>
</tr>
<tr>
<td>014<em>A+6+14</em>8</td>
<td></td>
</tr>
<tr>
<td>001A+6+0</td>
<td></td>
</tr>
<tr>
<td>015*A+7+8</td>
<td></td>
</tr>
<tr>
<td>0A+1+2*8</td>
<td></td>
</tr>
<tr>
<td>02A+2+8</td>
<td></td>
</tr>
<tr>
<td>0A+1+15*8</td>
<td></td>
</tr>
<tr>
<td>015<em>A+1+14</em>8</td>
<td></td>
</tr>
<tr>
<td>015*A+7</td>
<td></td>
</tr>
<tr>
<td>0A+8+1</td>
<td></td>
</tr>
<tr>
<td>015*8</td>
<td></td>
</tr>
<tr>
<td>015<em>A+7+15</em>8</td>
<td></td>
</tr>
<tr>
<td>08</td>
<td></td>
</tr>
<tr>
<td>00A+1</td>
<td></td>
</tr>
</tbody>
</table>

**REM M64M1 TABLE M179**

<table>
<thead>
<tr>
<th>Number</th>
<th>Piece</th>
</tr>
</thead>
<tbody>
<tr>
<td>015*64-1</td>
<td></td>
</tr>
<tr>
<td>014*64-1</td>
<td></td>
</tr>
<tr>
<td>013*64-1</td>
<td></td>
</tr>
<tr>
<td>012*64-1</td>
<td></td>
</tr>
<tr>
<td>011*64-1</td>
<td></td>
</tr>
<tr>
<td>010*64-1</td>
<td></td>
</tr>
<tr>
<td>09*64-1</td>
<td></td>
</tr>
<tr>
<td>08*64-1</td>
<td></td>
</tr>
<tr>
<td>07*64-1</td>
<td></td>
</tr>
<tr>
<td>06*64-1</td>
<td></td>
</tr>
<tr>
<td>05*64-1</td>
<td></td>
</tr>
<tr>
<td>04*64-1</td>
<td></td>
</tr>
<tr>
<td>03*64-1</td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td>Definition</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>MZE</td>
<td>0,2*64-1</td>
</tr>
<tr>
<td>REM</td>
<td>IEXTS TABLE</td>
</tr>
<tr>
<td>ORG</td>
<td>BSS</td>
</tr>
<tr>
<td>K</td>
<td>SYN</td>
</tr>
<tr>
<td></td>
<td>DUP</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>JPawn Table</td>
</tr>
<tr>
<td>ORG</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>IPDIRE Table</td>
</tr>
<tr>
<td>ORG</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>NMOV Table</td>
</tr>
<tr>
<td>ORG</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>IOP Table</td>
</tr>
<tr>
<td>ORG</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
0,0,6
0,0,5
0,0,8
0,0,7
0,0,2
0,0,1
0,0,4
0,0,3

REM MSTO TABLE
ORG MST0-31
DUP 1,32
0,0,MST0*1024-**1024

REM JPRGM TABLE
ORG JPRGM-3
4
2
3
6

ORG KVAL-5
PZE 0,0,10-1
PZE 0,0,1000
PZE 0,0,3
PZE 0,0,3
PZE 0,0,5
PZE 0,0,1
END 96
* LABEL
* FAP
COUNT 20
* FAP
COUNT 8
ABS KPAWNV EQU 29559
LCENSQ EQU 29551
ORG LCENSQ-15
DEC 43817,44B17,45B17,46B17,39B17,3AB17,3B17,38B17
DEC 30B17,29B17,28B17,27B17,22B17,21B17,20B17,19B17
ORG KPAWNV-7
DEC 0,0,4B17,6B17,6B17,4B17,0,0
END
* LABEL
* FAP
COUNT  45
* PIECES  TABLE FOR CHESS BOARD PRINTOUT
ABS
PIECES EQU  29624
ORG PIECES-42
BCI  1, Q1
BCI  1, F00
BCI  1, B
BCI  1, N
BCI  1, R
BCI  1, Q1
BCI  1, F00
BCI  1, B
BCI  1, N
BCI  1, R
BCI  1, Q
BCI  1, Q
BCI  1, KB
BCI  1, KB
BCI  1, QB
BCI  1, QB
BCI  1, KN
BCI  1, KN
BCI  1, QN
BCI  1, QN
BCI  1, KRP
BCI  1, KRP
BCI  1, KNP
BCI  1, KNP
BCI  1, KBP
BCI  1, KBP
BCI  1, KP
BCI  1, KP
BCI  1, QP
BCI  1, QP
BCI  1, QBP
BCI  1, QBP
BCI  1, QNP
BCI  1, QNP
BCI  1, QRP
BCI  1, QRP
BCI  1, KR
BCI  1, KR
BCI  1, QR
BCI  1, QR
BCI  1, K
BCI  1, K
END

TOTAL  4335
SET OF TABLES NUMBER 30 MOVE IS *QN -KB6 BLACK

---

**********************************************************************
* * QR * * Q * KR * * K * *
* * ----- * ----- ----- *
**********************************************************************
* QRP* QNP* QBP* * KBP* KNP* KRP*
* ----- ----- * ----- ----- *
**********************************************************************

* * * * * KN * * *
* * * * * ---- * *
**********************************************************************
* * QR * * QP * * *
* * * * * ----- * *
**********************************************************************
* * * * * Q * * *
* * * * * * * *
**********************************************************************
* QRP* * QNP* * KP * QN * *
* * * * * * * *
**********************************************************************
* * * * * QBP* * KBP* KNP* KRP*
* * * * * * * *
**********************************************************************
* * * * * QB * * K * * KR *
* * * * * * * *

MAVAIL

K -KB1 K -K2 K -Q1 KNP-KB3

WHITE

/SAMPLE INITIA INPUT/ 2 QB 1 K 2 KR 2 CBP 2 KBP KNP KRP QRP
1 QNF 1 KP *QN 5 5 CR 1 *QP 9 *KN 2 *QRP *QNP *QBP 2 *KBP
*KNP *KRP 1 *QR 1 *Q *KR 1 *K 1.
APPENDIX 3

Record of game played 2/24/62. Machine - white, M. Garber - black

<table>
<thead>
<tr>
<th>move</th>
<th>White</th>
<th>Black</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>KP-K4</td>
<td>KP-K4</td>
<td>1.5 min.</td>
</tr>
<tr>
<td>2</td>
<td>QN-QB3</td>
<td>KN-KB3</td>
<td>1.8</td>
</tr>
<tr>
<td>3</td>
<td>KN-KB3</td>
<td>QN-QB3</td>
<td>2.2</td>
</tr>
<tr>
<td>4</td>
<td>QP-Q4</td>
<td>PnP</td>
<td>4.8</td>
</tr>
<tr>
<td>5</td>
<td>NnP</td>
<td>KB-QB4</td>
<td>.8</td>
</tr>
<tr>
<td>6</td>
<td>NxN</td>
<td>QNPxN</td>
<td>3.3</td>
</tr>
<tr>
<td>7</td>
<td>KP-K5</td>
<td>Q-K2</td>
<td>4.4</td>
</tr>
<tr>
<td>8</td>
<td>QB-KB4</td>
<td>QP-Q3</td>
<td>2.2</td>
</tr>
<tr>
<td>10</td>
<td>KPnP(Q6)</td>
<td>KBnP(KB7)ch</td>
<td>1.2</td>
</tr>
<tr>
<td>11</td>
<td>K-Q2</td>
<td>QxQch</td>
<td>.9</td>
</tr>
<tr>
<td>12</td>
<td>KBxQ</td>
<td>QBPxP</td>
<td>1.5</td>
</tr>
<tr>
<td>13</td>
<td>QBxP</td>
<td>KB-K6ch</td>
<td>2.4</td>
</tr>
<tr>
<td>14</td>
<td>K-Q3</td>
<td>QB-QR3ch</td>
<td>1.1</td>
</tr>
<tr>
<td>15</td>
<td>QBP-QB4</td>
<td>O-O-O</td>
<td>1.0</td>
</tr>
<tr>
<td>16</td>
<td>KBxNch</td>
<td>K-QN2</td>
<td>.4</td>
</tr>
<tr>
<td>17</td>
<td>QN-K4</td>
<td>KB-KB5</td>
<td>.4</td>
</tr>
<tr>
<td>18</td>
<td>QN-QB5ch</td>
<td>K-QN3</td>
<td>.9</td>
</tr>
<tr>
<td>19</td>
<td>QN-Q7ch</td>
<td>RxN</td>
<td>.9</td>
</tr>
<tr>
<td>20</td>
<td>KBxR</td>
<td>BxKB</td>
<td>.3</td>
</tr>
<tr>
<td>21</td>
<td>QR-KB1</td>
<td>KBP-KB3</td>
<td>3.4</td>
</tr>
<tr>
<td>22</td>
<td>QR-KB5</td>
<td>KR-Q1</td>
<td>3.3</td>
</tr>
<tr>
<td>23</td>
<td>KB-K6</td>
<td>KNP-KN3</td>
<td>.6</td>
</tr>
<tr>
<td>24</td>
<td>QR-KB1</td>
<td>KBP-KB4</td>
<td>1.5</td>
</tr>
</tbody>
</table>
25  K-Q4  KB-QB4ch  2.2  
26  K-QB3  KB-Q5ch  .8  
27  K-QN3  K-QB4  .8  

average time = 1.8 min./ move
<table>
<thead>
<tr>
<th>move</th>
<th>White</th>
<th>Black</th>
<th>Principal variation</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>KP-K4</td>
<td>KP-K4</td>
<td>KP-K4</td>
<td>1.7</td>
</tr>
<tr>
<td>2</td>
<td>QN-QB3</td>
<td>QN-QB3</td>
<td>QN-QB3</td>
<td>2.8</td>
</tr>
<tr>
<td>3</td>
<td>KN-KB3</td>
<td>KB-QB4</td>
<td>KB-QB4</td>
<td>2.4</td>
</tr>
<tr>
<td>4</td>
<td>KB-QB4</td>
<td>KN-KB3</td>
<td>QP-Q3</td>
<td>5.4</td>
</tr>
<tr>
<td>5</td>
<td>KB-Q5</td>
<td>K-KN1</td>
<td>QP-Q3</td>
<td>6.2</td>
</tr>
<tr>
<td>6</td>
<td>K-KN1</td>
<td>QP-Q3</td>
<td>QP-Q3</td>
<td>5.8</td>
</tr>
<tr>
<td>7</td>
<td>KB-QB6</td>
<td>QNP-QB3</td>
<td>QNP-QB3</td>
<td>3.2</td>
</tr>
<tr>
<td>8</td>
<td>QP-Q3</td>
<td>KN-KN5</td>
<td>QB-KN5</td>
<td>5.5</td>
</tr>
<tr>
<td>9</td>
<td>QB-KN5</td>
<td>KBP-KB3</td>
<td>KBP-KB3</td>
<td>4.0</td>
</tr>
<tr>
<td>10</td>
<td>QB-KR4</td>
<td>QP-Q4</td>
<td>QB-QR3</td>
<td>3.8</td>
</tr>
<tr>
<td>11</td>
<td>KP-Q5</td>
<td>QNP-Q4</td>
<td>QNP-Q4</td>
<td>4.9</td>
</tr>
<tr>
<td>12</td>
<td>QP-Q4</td>
<td>KP-Q5</td>
<td>KN-Q4</td>
<td>5.1</td>
</tr>
<tr>
<td>13</td>
<td>KN-Q4</td>
<td>KB-Q3</td>
<td>KB-Q5</td>
<td>1.5</td>
</tr>
<tr>
<td>14</td>
<td>KBP-KB4</td>
<td>KB-KB4</td>
<td>KB-QB4</td>
<td>3.7</td>
</tr>
<tr>
<td>15</td>
<td>KN-KB5</td>
<td>KNP-KN3</td>
<td>KN-KR3</td>
<td>1.9</td>
</tr>
<tr>
<td>16</td>
<td>Q-KN4</td>
<td>QNP-Q5</td>
<td>KB-QB4</td>
<td>7.7</td>
</tr>
<tr>
<td>17</td>
<td>QN-K4</td>
<td>K-KR1</td>
<td>KB-QN5</td>
<td>3.6</td>
</tr>
<tr>
<td>18</td>
<td>KN-Q4</td>
<td>Q-K1</td>
<td>KB-K2</td>
<td>4.8</td>
</tr>
<tr>
<td>19</td>
<td>Q-KB3</td>
<td>QR-QN1</td>
<td>Q-Q1</td>
<td>4.3</td>
</tr>
<tr>
<td>20</td>
<td>QB-KB6</td>
<td>K-KN1</td>
<td>K-KN1</td>
<td>2.4</td>
</tr>
<tr>
<td>21</td>
<td>QR-QN1</td>
<td>QR-QN5</td>
<td>QN-QN5</td>
<td>5.7</td>
</tr>
<tr>
<td>22</td>
<td>QBP-QB3</td>
<td>QB-QB5</td>
<td>QB-QB5</td>
<td>8.0</td>
</tr>
<tr>
<td>23</td>
<td>QR-K1</td>
<td>Q-Q2</td>
<td>Q-QR1</td>
<td>5.8</td>
</tr>
<tr>
<td>24</td>
<td>Q-Q1</td>
<td>QR-QB4</td>
<td>KB-K2</td>
<td>3.9</td>
</tr>
<tr>
<td>25</td>
<td>KN-KB3</td>
<td>KR-Q1</td>
<td>Q-K3</td>
<td>8.0</td>
</tr>
</tbody>
</table>

Record of game played 4/21/62. Machine - white
R. Fiorenza - black
<table>
<thead>
<tr>
<th>Move</th>
<th>Piece 1</th>
<th>Piece 2</th>
<th>Piece 3</th>
<th>Piece 4</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>QN-QB5</td>
<td>KB-QB4</td>
<td>KB-QB4</td>
<td>K-KR1</td>
<td>4.1</td>
</tr>
<tr>
<td>27</td>
<td>KN-Q4</td>
<td>KB-K2</td>
<td>KR-KB1</td>
<td>QB-K5</td>
<td>3.3</td>
</tr>
<tr>
<td>28</td>
<td>QR-K7</td>
<td>Q-Q3</td>
<td>Q-QB1</td>
<td></td>
<td>3.2</td>
</tr>
<tr>
<td>29</td>
<td>QB-KN5</td>
<td>KR-KB1</td>
<td>KR-Q2</td>
<td>QR-Q7</td>
<td>2.3</td>
</tr>
<tr>
<td>30</td>
<td>QBP-QB4</td>
<td>QBP-QB4</td>
<td>Q-Q1</td>
<td></td>
<td>3.5</td>
</tr>
<tr>
<td>31</td>
<td>KN-K6</td>
<td>Q-Q8</td>
<td>Q-QR3</td>
<td>KN-KB8</td>
<td>2.0</td>
</tr>
<tr>
<td>33</td>
<td>QR-KN7</td>
<td>K-KR1</td>
<td>K-KR1</td>
<td>KR-Q8</td>
<td>1.0</td>
</tr>
<tr>
<td>34</td>
<td>QR-QB7</td>
<td>-</td>
<td>KR-QR1</td>
<td>KR-Q7</td>
<td>3.0</td>
</tr>
</tbody>
</table>

average time = 4.4 min./move
<table>
<thead>
<tr>
<th>LEVEL</th>
<th>OPPONENT</th>
<th>MACHINE</th>
<th>PRINCIPAL VARIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*KN - KB3</td>
<td>QBP - QB4</td>
<td>*KN - K5</td>
</tr>
<tr>
<td>2</td>
<td>*KNB - KN3</td>
<td>QN - QB3</td>
<td>*QP - Q3</td>
</tr>
<tr>
<td>8</td>
<td>*QBP - Q5</td>
<td>QN - QN5</td>
<td>*KP - K4, QB - KN5, *QN - QBP</td>
</tr>
<tr>
<td>12</td>
<td>*KN - KB5</td>
<td>KNP - KB4</td>
<td>*QP - K4, KNP - K5, *QP - KB6</td>
</tr>
<tr>
<td>13</td>
<td>*QP - K4</td>
<td>KNP - KB4, Q - KB6</td>
<td>*KP - K3</td>
</tr>
<tr>
<td>14</td>
<td>*QN - Q2</td>
<td>QN - QB4</td>
<td>*QB - KB6, QB - KB3, *QN - QBP</td>
</tr>
<tr>
<td>15</td>
<td>*QN - QBP</td>
<td>KR - K5</td>
<td>*KB - K4, KB - KB5, *QP - KB6, QN - KB3, Q - QBP, QP - QBP</td>
</tr>
<tr>
<td>16</td>
<td>*KB - K4</td>
<td>KB - K4</td>
<td>*QP - QBP, QP - QBP</td>
</tr>
<tr>
<td>17</td>
<td>*K - KR1</td>
<td>KB - K4</td>
<td>*QP - QBP, QP - QBP</td>
</tr>
</tbody>
</table>