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–Abrahams 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 -infinity, and beta at +infinity. 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 REPLYYS 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:

```
  8   8   4   4
  4   8   8   4
  2   4   4   2
  1   1   1   1
```

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 PUTCHE.

REVERT    Takes back the last updated move. This is actual-
ly another option of 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. KBP 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 *

WHITE

MAVAIL

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


QB - Q2  QB - K3  QB -KB4  QB -KN5  QB -KR6  Q - Q2  Q - Q3

FIGURE 1
Representative output
PRINCIPAL VARIATION

VALUE= 27  EFFORT= 1449

*QP - Q4  KN - KB3

-14-
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
*KN - KB3
KN - KB3
*QB - KB4
QB - K5

KN - KB3
*KN - KB3
QN - QB3
*QB - KB4
QB - K5

*KN - KB3
QN - Q5
QP - Q5
*QN - QN5
KP - K4
QN - QB3
*QN - QR4
QP - Q5
KP - K4
QN - QB3

KP - K4
QN - QB3
*KN - KB3
QN - QB3
KN - KB3
QB - KB4
*QP - Q3
KP - K4
QN - QB3
KN - KB3
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 position. 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 3/4 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............
* DIMENSION 1HOPE(64), LISP(6000)
* NEXT FREE REGISTER IN COMMON = 23375
*
* INITIALIZE
ENTRY TREE
SXA XR1+1
SXA XR1+1,2
SXO XR4+4
XR4 SYX TREE-2
AXT 3000,1
STZ LISP+1,1
TIX %-1,1,1
STZ MOVE
CALL STRTG
AXT 1,1
STZ BACK
STZ PLY
*
* GENERATE A NEW BLOCK.
D CLA PLY
ADD =0200
STO PLY
CLA MOVE HEAD NEW BLOCK
ADD BACK
SXA BACK+1
STO LISP+1,1
BEGIN COMPARISON OF A, B 2 BLOCKS HIGHER

FAIL TEST 1 -- REVERT TWICE (PASS, TRA FO1)

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

* TSX $PESVL,4  FUNCTIONS RETURN IN ALGEBRAIC FORM
SPG TSX $OPTVL,4
TSX $PESVL,4
CAL =0377777000000
TS1 TRA F01
CAL =0777777000000
TS2 TRA OUT
CAL =0377777000000

* F01 CLA LISP+1,1 BEGIN TEST 2--A,3 ONE BLOCK HIGHER
PAX ,2
TXL OUT,2,0
CLA LISP-1,2
TPL ,+6
CLA LISP+1,2
PAX ,2
TXL OUT,2,0
CLA LISP+1,2
TRA F01+1

NM XEC SPG+2,4
TXI ,+2,0,0
PZE TREE-2,0,0
STO VALUE
LDI =2
STI 10
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 CHAIL--GO BACK 2 MORE LEVELS
CLA PLY
SUB =0200
STO PLY
TRA DN

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

ORA =0377777000000
INT ORA =0777777000000
OUT CALL REPLY5
LXD IPE,2
PXA ,2
ADD PLY
SLW LISP,1
TAL B,2,0
LXD MCOL,4
XEC INT+1,4
SLW LISP,1
SXD AF,2
AXT 1,2

Q CLA IHOPE+1,2
STO LISP-1,1
TXI **+1,1,1
TXI **+1,2,1
AF TXL Q,2,**
TXH ERR,1,3000
CAL =0
ORS LISP,1
TXI C+1,2
CAL =0
ORS LISP+1,1
TXI NOMGEV,1,2

* ERR CALL ERROR,FMT
TSX $LDUMP,4
+MT BCI 2*LISP FULL,7
SVN -1,79,1
*
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

LIST PRODUCED BY REPLY5
(NO REPLY5--POSITION STATIC)
CALL UPDATE, MOVE
CLA REMOVE SHIFT PROMOTION INFORMATION
LRS 0
STD AA
ALS 15
ANA 07100000
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 MHOPE. EVALUATE THE QUOTE
* STATIC UNQUOTE POSITION.

NOMOVE CLA LISP+3, 1
STA BACK
CALL EVAL
ORS LISP+2, 1 VALUE RETURNED IN LOGICAL FORM
CLA LISP+2, 1
LK LXD MCOL, 4
LXA BACK+2
CAS LISP, 2 MINIMAX VALUE INTO NEST HIGHER LEVEL
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 A,B TEST
STA A
NIN LXA A, 4
TXL OT*, 0
CLA LISP-1, 4
TPL IN
CLA LISP+1, 4
PAX *, 4
TXL OT*, 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+2,4 PASS TEST (FAIL, TRA OT)
GRA CAL LISP+2
ADD =1
SLW LISP+2
CLA LISP+1,2
PAX ,4
CLA LISP-1,4
TMI ARG (SINGLE REPLY CHAIN)
SXU 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
BNR TRA OT
STD LISP+2
*
*
USEUP CLA LISP+1,2 . ALL REPLYS IN CLOCK USED
STA BACK
CLA PLY
ARS 7
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 PTL1,4
CLA AC
LXA RX4,4

XR1 AXT **,1
AXT **,2
LXD XR4,4
STO* 1,4
TRA 2,4

MOVE PZE
BACK PZE
AA PZE
A PZE
RMOVE PZE
TEST PZE
VALUE PZE
M PZE

NPLY BSS 19

PT SXA WW,4
TP LXA 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
ID   PZE
AC   PZE
WW   PZE
RX1  PZE
RX2  PZE
RX4  PZE

*  
PTA  SXA RX1 1
     SXA RX2 2
     SXA QQ 4
     STO AC
     ORS AC
     TSX $(SPH) 4
     PZE FMTT 1 1
     LDQ ID
     STR
     LDQ 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 (14M4THIS IS POINT 05//6020)

*  
PTL  SXA BK 4
     CLA RX1
     ADD = 2
     ALS 18
     STD EPI
     TSX $(SPH) 4
     PZE FOR 1 1
     AXT 1 1
     RK PXA , 1
     XCA
     STR
LDQ LISP+1,1
STR
TXI *+1,1,1
EPI TXL RK,1,**
TSX $(FIL),4
BK AXT **,4
TRA 1,*4

* PTP
SXA GB,4
TSX $(SPH),4
PZE FOR,,,1
AXT 1,1
SIX PXA ,1
XCA
STR
LDQ NPLY+1,1
STR
TXI *+1,1,1
TXL SIX,1,10
TSX $(FIL),4
GB AXT **,4
TRA 1,*4
FOR BCI 3,//(10X,04,020)

* ZILCH COMMON 12561
R COMMON 1
K I 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+9370
END
* LABEL
* FAP
* SWAP SOUTRINTINE, 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 ATT. AND DEFS.
* THE DECREMENT 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   =0707070707070  MAKE NO LOOK PRETTY
* SXA    XR1,1      SAVE XR5
* SXA    XR1+1,2
* SXD    XR4,4
* XR4   SYN SWAP-2
* STI    INDIC      SAVE INDICATORS
* AXT    33,1      INITIALIZE CHAIN AND IECCH
* PXA    *1
* STL    CHAIN+1,1
* STZ    IEXCH+1,1
* TXI    #0,1,1
* AXT    34,1
* SXD    COUNT+1
* AXT    1,1
* SXD    K,1
* CLA    LOC+1,1
* STZ    ATACK
* TZE    y
* SUB    #1B17
* PDX    #2
* AXT    960,4
* AXT    0,1
* WET    IBEAR,6
* TXI    C,1,1
* D     TIX    #1,4,64
* TXL    D1,4,0
* ZET    IBEAR,2
* TXI    D-1,5,64
* D1    CLA    COUNT
* TXH    ORDER,1,0
*
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

XR2 HAS LOC(BEAVER) = 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

YES, SET FLIP-FLOP

NO
* 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
   PIND1 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 C1
   TRA C1
   PINK4 LX A F*2
   TXI D*1,-1

* PHASE 2: COPY INTER INTO 1EXCH IN ORDER
CLA  KVAL+1,4
CAS  MINVAL
TXI  M+1,1
NOP
LDI  INTER+1,1
RFT  1
TRA  T
R1
STO  MINVAL
SXA  CAND+1
TXI  M+1,1
T
LDI  INTER+2,1
LNT  400000
TXI  M+1,1
TRA  R1
* WE HAVE USED ALL ATTACKERS OR DEFENDERS.
NOMORE
LXD  K+1
CLA  COUNT1
ZET  SIDE
TRA  V
STT  IEXCH+1,1
CLA  =1B17
STO  SIDE
TRA  U-1
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
INTER  COMMON  20
CHAIN  COMMON  32
COMMON  206-20-32+12561
SET TO TOP OF MEMORY
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+6375
END
263
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), KPANV(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), KVAL)
EQUIVALENCE (FOO(1591), LOC), (FOO(1623), NFIRST), (FOO(3003), KPANV)
EQUIVALENCE (FOO(3121), PLY), (FOO(3120), IPE), (FOO(2917), KVAL)
EQUIVALENCE (FOO(3051), MUPW), (FOO(3052), MOBB), (FOO(3123), IHOPE)
EQUIVALENCE (FOO(3188), IEXCH), (FOO(3122), BACK), (FOO(3187), LISP)
EQUIVALENCE (FOO(3053), MATW), (FOO(3054), MATB), (FOO(3119), MLOG)
EQUIVALENCE (FOO(134), NLOG)

MCOL = MCOL
IARG = 0
IAT = 0
IND = 0
IW = 0
IB = 0

PLY = XSHIFTF(PLY, 11)
DO 5 I = 1, 32
    MPVAL(I) = 0
5 N1AT(I) = 0
DO 200 I = 1, 32
    NAT = XTAGF(IEXCH(I))
    IF(NAT) 200, 200, 10
10 NDEF = XPREF(IEXCH(I))
    IF(NAT - NDEF) 20, 20, 30
20 K = NAT + NAT + 1
    GO TO 40
30 K = NDEF + NDEF + 2
40 ITAB(I) = I
    IATOR = XDECF(IEXCH(I))
    IDEFOR = IATOR + NAT
    M = 0
    J = 1 - XSHIFTF(XLBITF(I), 1)
    IFAT = XDECF(IEXCH(IATOR))
    IDVAL = XGETF(KIND(I), KVAL) - XGETF(KIND(IFAT), KVAL)
    IF(IDVAL) 50, 50, 57
57 IF(XLBITF(KIND(IFAT))) 50, 50, 400
400 IAT = IAT + IDVAL * J
50 DO 70 L = 2, K, 2
    ITAB(L) = XDECF(XGETF(M + IATOR + IEXCH))
    ITAB(L + 1) = XDECF(XGETF(M + IDEFOR + IEXCH))
70 CONTINUE
300 CONTINUE
310 DO 320 I = MCOL, M2, 2
320 NCVAL = XMAXOF(NCVAL, MPVAL(I))
   DO 330 I = 1, M1, 2
      IW = IW + XMAXOF(0, MPVAL(I))
   330 IB = IB + XMAXOF(0, MPVAL(I+1))
      IW = -IW
      GO TO (350, 380), MCOL
C +IB OR -IW 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=XSIGNF(XONEF(IAT),IAT)
      RETURN
      END
* LABEL
  * LIST8
  SUBROUTINE PINS
  COMMON FOO
  DIMENSION IBEAR(64,16), IOCC(64)
  DIMENSION FOO(5000)
  DIMENSION LOC(32), NFIRST(22), XPWNV(8), IEXTD(16), IEXTS(64)
  DIMENSION IPIN(32), IOPP(16), KIND(32), MAVAIL(100), KVAL(6)
  DIMENSION KPIN(32)
  EQUIVALENCE (FOO(2892), K1), (FOO(1463), KN1), (FOO(2765), MAVAIL)
  EQUIVALENCE (FOO(2900), MCSV)
  EQUIVALENCE (FOO(2703), IPIN), (FOO(1283), IOPP), (FOO(255), IBEAR)
  EQUIVALENCE (FOO(1369), IEXTD), (FOO(1301), IEXTS), (FOO(1527), IOCC)
  EQUIVALENCE (FOO(1591), LOC), (FOO(1623), NFIRST), (FOO(3003), XPWNV)
  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) I, 1, 3
  3 IF (XLBITF(IOCC(JPIN)+J)) 1, 4, 1
  4 IFOO = IBEAR(JPIN+1)
  IF (IFOO) I, 1, 5
  5 IF (XORFXLBITF(IFOO+J), XLBITF(KIND(IFOO)+1))) 6, 1, 6
  6 JPIN = IOCC(JPIN)
  GO TO (15, 16), KRAP
  15 KPIN(JPIN) = 1
  GO TO 1
  16 KPIN(JPIN) = -(KPIN(JPIN) + XSHIFT(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
SXD 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
PXA ,4
COLOR HTR
LBT
TRA OPPOS
CLA NPAWNS
ADD =1B17
STO NPAWNS
CLA ADJAC
STO PROTEC
FOR 7094
SET UP FOR WHITE LOOP
INITIALIZE XECUTE 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, 0 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
CONVERT INTO TRUE RANK

TXI **1,2,56
SX A LRANK,2
TRA EVL
DONE CLA TPAWNV
SUB PAWNV
*
LMTM
XR1 AXT **,1
XR2 AXT **,2
LXD XR4,4
TRA 2,4
DUP 1,10
PDX ,0
STL PAST
DUP 2,8
STL ADJAC
STL PAST
DUP 1,7
PDX ,0
TABLE SYN
PAWNV PZE
ADJAC PZE
PROTEC PZE
NP A WNS PZE
PAST PZE
OTHER PZE
TPAWNV PZE
LRANK PZE
*
VALUE TABLES
DEC 7B17,7B17,8B17,8B17,8B17,8B17,7B17
IOPEN DEC 7B17
DEC -5B17,-1B17,-1B17,-1B17,-1B17,-1B17,-1B17
ISOLAT DEC -5B17
DEC 0,-5B17,-5B17,-6B17,-6B17,-5B17,-5B17
IBKWD DEC 0
DEC -4B17,-4B17,-2B17,-3B17,-3B17,-2B17,-4B17
IDBLD DEC -4B17
DEC -3B17,0,0,0,0,0,0
KPAST DEC -3B17
ZILCH COMMON 12561
R COMMON 1
IOCC SYN R+11035
END

FOR 7094
* LABEL
* LISTB
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 FO0
DIMENSION KP1N(32)
DIMENSION IBEAR(64,16), LOC(32), KIND(32), FO0(5000)
DIMENSION LCENSQ(16), KCNVAL(16)
EQUIVALENCE (FO0(9317), NMoves)
EQUIVALENCE (FO0(6187), KPIN)
EQUIVALENCE(FO0(2892), K1), (FO0(1463), KIND), (FO0(2765), MAVAIL)
EQUIVALENCE(FO0(2703), 1PIN), (FO0(1285), I0PP), (FO0(255), IBEAR)
EQUIVALENCE(FO0(1591), LOC), (FO0(1623), NFIRST), (FO0(3003), KPAWNV)
EQUIVALENCE (FO0(3011), LCENSQ), (FO0(3027), KCNVAL)
ICENTR = 0
102 IF (NMoves - 30) 102, 101, 101
1123 = 1123
DO 100 I = 1, 16
   K = LCENSQ(I)
   DO 90 J = 1, 16
      IF (IBEAR(K, J)) 90, 90, 10
10   K P = IBEAR(K, J)
      IF (KPIN(KP)) 90, 13, 90
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(17-1, KCNVAL)/3
   GO TO 40
15   IF (XLBITF(KP)) 30, 30, 20
20   ICENTR = ICENTR + KCNVAL(I)
   GO TO 40
30   ICENTR = ICENTR - XGETF(17-1, KCNVAL)
40   LOCKP = LOC(KP)
      IF (IBEAR(LOCKP, J)) 90, 90, 50
50   KPP = IBEAR(LOCKP, J)
      IF (XLBITF(KPP+KP) + XLBITF(KIND(KPP))) 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

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(2692), K1), (FOO(1463), KIND), (FOO(2769), 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)
XBLTCHF(J)=XORF(XGETF(J+ICOLOR,LOC),XTRANKF(XGETF(J+ICOLOR,LOC),
1J+ICOLOR)) + XNOTF(XGETF(J+ICOLOR,LOC))

IDVLOP = 0
I123 = I123
ICOLOR = 0
IF (NMOVES - 15) 69, 100, 100

69 IBARF = IPESS
IPESS = 0
DO 1 I = 7, 21, 2
1 IPESS = IPESS + XNOTF(XGETF(I+ICOLOR, NFIRST))
DO 2 I = 13, 19, 2
IPESS = IPESS + XGETF(XBLTCHF(1), KPAWNV)
IF (XGETF(1+ICOLOR, NFIRST) + XNOTF(XGETF(1+ICOLOR, LOC))) 23, 22, 2

22 IDIR = XSHIFTF(1+ICOLOR+1, 1)
NSQ = XORF(IEXTD(IDIR)+XGETF(XGETF(ICOLOR+1, LOC), IEXTS))
IF (IOCC(NSQ)+XGETF(XMOVF(IEXTD(IDIR)+IEXTS(NSQ)), IOCC)) 23, 2, 23

23 IPESS = IPESS - 5
2 CONTINUE
IPESS = IPESS + 5*XNOTF(XBLTCHF(11)-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 - ICOLOR) 64, 63, 64
64 IPESS = IPESS + 4
63 IF (IOCC(KJ1+1) - 27 - ICOLOR) 66, 65, 66
66 IPESS = IPESS + 3
IF (IOCC(KQ2) - 23 - ICOLOR) 69, 66, 68

166 IPESS = IPESS - 10
65 IF (IOCC(KJ2-1) - 29 - ICOLOR) 68, 67, 68
68 IPESS = IPESS + 3
67 IF (IOCC(KQ2+1) - 25 - ICOLOR) 67, 68, 69

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 (LOC (ICOLOR + 31), XNOTF (XRAANGEF (XBLTCHF (31), 1, 3))) + IPESS
75 ICOLOR = ICOLOR + 1
GO TO (69, 711) + ICOLOR
711 IDVLOP = IBARF - IPESS
100 RETURN
END
* LABEL
* LIST8

SUBROUTINE REPLY8
DIMENSION MPVAL(100)
DIMENSION FOO(5000)
DIMENSION LOC(32),NFIRST(22),PAWNV(8),IEXTD(16),IESTS(64)
DIMENSION IPIN(32),IOPP(16),KIND(32),MAVAIL(100),KVAL(6)
DIMENSION IHOPE(64),IEXCH(128)
DIMENSION LISP(6000)
DIMENSION KPLY(20)
COMMON FOO

EQUIVALENCE (FOO(6219),IWHTM),(FOO(6220),IBLKM)
EQUIVALENCE (FOO(2892),K1),(FOO(1463),KIND),(FOO(2765),MAVAIL)
EQUIVALENCE (FOO(2900),MCOL)
EQUIVALENCE (FOO(2703),IPIN),(FOO(1265),IOPP),(FOO(255),IBEAK)
EQUIVALENCE (FOO(1365),IEXTD),(FOO(1301),IEXTS),(FOO(1527),IUC)
EQUIVALENCE (FOO(1591),LOC),(FOO(1623),NFIRST),(FOO(3003),PAWNV)
EQUIVALENCE (FOO(3121),PLY),(FOO(3120),IPE),(FOO(2917),KVAL)
EQUIVALENCE (FOO(3051),MOBN),(FOO(3052),MOBB),(FOO(3123),IHOPE)
EQUIVALENCE (FOO(9186),IEXCH),(FOO(3122),BACK),(FOO(3187),LISP)
EQUIVALENCE (FOO(3053),MATW),(FOO(3054),MATB),(FOO(3119),MLUG)
EQUIVALENCE (FOO(134),NLOG)
EQUIVALENCE (FOO(2677),ICHECK)
EQUIVALENCE (KPLY,FOO(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(IPLY),K1)
   IF (IPE) 666, 666, 99
99  IF(IPLY-2)30,30,200
30 ISTAB=1
600 DO 80 M=1,K1
   NP=XGETF(XMV1F(MAVERN),IUC)
   MVR=XMV1F(MAVERN)
   IF(KIND(MVR)-5)33,33,35
32   IF(XABSF(LOC(MVR)-XMV1F(MAVERN))-2)33,33,35
33   KS=28
   GO TO 37
35   KS=0
37   CALL UPDATE(MAVERN)
869 CALL PINS
   CALL SWAP
   CALL LTRP(1W,ID,IND,IARK,IAK)
   IDT = IDVLOP(1)
   IF IAT*J162,62,60
60   IF(XMAXOF(1IDT-ID)*J-2,0)+XNUMF(XRANGEF(MVR,13,16)))62,62,61
61   ISTAB = 1
   GO TO 629
62 IAT=0
629 IF(NP)50,50,40
40 MAVN=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 * IBLAM
77  MPVAL = NVAL + (IWHIM * IW + IBLKM * IB + XSHIFTF(IDT, Z) + ICENTR
    1(1) + XSHIFTF(IAT, 4) + 3*JPAWNS(1)) * J + KS + 24/K1**2 + IKAPT + 2*ARG
   CALL REVERT
   IF (ISTAB) 250, 250, 85
250  IF(XLBITF(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
   MVAL=MPVAL(M)
   K=M
110  CONTINUE
   IHOPE(LM)=MVAL(K)
120  MPVAL(K)=-5000
   GO TO 900
200  CALL SWAP
   CALL LTRADE(IW, IB, IND, IARG, IAT)
   IF(IND+IB-IW+IARG) 600, 600, 210
210  IF(IPLY-3) 30, 30, 220
220  IF(IB-IW+XABSF(IARG)) 600, 600, 222
222  IF(IPLY-5) 30, 30, 224
224  IF(IB-IW) 600, 600, 30
900  IF(IPLY-2) 905, 905, 950
1000 FORMAT(6H01PLY=16, 4X, 14A6)
905   DO 910 M=1, IPE
   CALL JUNPAK(M, IHOPE(M), MPVAL(M), MPVAL(M+8))
910   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
  * LIST8
SUBROUTINE EVAL
DIMENSION FOO(5000)
DIMENSION LOC(32), NFIRST(22), KPAWNS(16), IEXTD(16), IEXTS(64)
DIMENSION IPIN(32), IOPP(16), KIND(32), MAVAIL(100), KVAL(6)
DIMENSION IHOPE(64), IEXCH(128)
DIMENSION LISP(6000)
DIMENSION NTYPE(50)
COMMON FOO
EQUIVALENCE (FOO(2892), K1), (FOO(1463), KIND), (FOO(2765), MAVAIL)
EQUIVALENCE (FOO(6219), IWHITE), (FOO(6220), IBLKM)
EQUIVALENCE (FOO(2900), MCOL)
EQUIVALENCE (FOO(2703), IPIN), (FOO(1285), IOPP), (FOO(255), IBEAR)
EQUIVALENCE (FOO(1365), IpEXT), (FOO(1301), IEXTS), (FOO(1527), IOCC)
EQUIVALENCE (FOO(1591), LOC), (FOO(1723), NFIRST), (FOO(3003), KPAWNS)
EQUIVALENCE (FOO(3121), IPLY), (FOO(3120), IPE), (FOO(2917), KVAL)
EQUIVALENCE (FOO(3051), MUBW), (FOO(3052), MUBB), (FOO(3123), IHOPE)
EQUIVALENCE (FOO(9188), IEXCH), (FOO(3122), BACK), (FOO(3187), LISP)
EQUIVALENCE (FOO(3053), MATW), (FOO(3054), MATB), (FOO(3119), MLOG)
EQUIVALENCE (FOO(134), NLOG)
EQUIVALENCE (I*A)
EQUIVALENCE (FOO(2913), NSPEC), (FOO(2649), NTYPE)
   IF(K1) 10, 10, 15
10 I=I+SIGNF(10000, MCOL+MCOL-3)
   GO TO 30
15 KS=0
   CALL PIN8
60 CALL SWAP
   CALL LTRADE(IW, ID, IND, IARG, IAT)
50 IF(NSPEC) 20, 20, 7
7   DO 1 I=1, NSPEC
       IF(NTYPE(I)+1)6,6,1
8       IF(XLBITF(XMV3F(NTYPE(I))))4,4,5
4      KS=KS+26
       GO TO 1
5      KS=KS+26
1   CONTINUE
20 I=IWHITE*(MATW+IW)+IBLKMH*(16-MATD)+4*KPAWNS(I)+XSHIFTF(IDVLOP(I),
   12)+ICENTR(I)
B 30 A=A
   RETURN
END
* LABEL
* LIST8
C THE LONG AWAITED STRATEGY PROGRAM. MAY 1, 1962
SUBROUTINE STRTGY
COMMON FOO
EQUIVALENCE (FOO(3053),MATW), (FOO(3054),MATB)
EQUIVALENCE (FOO(6219),IWHTM),(FOO(6220),IBLKM)
CALL PINS
CALL SWAP
CALL LTRADE (IW,IB,IND,IAKG,IAT)
ITEM = IW + IB + MATW - MATB
IWHTM = 60
IBLKM = 60
IF (XABSF (ITEM) - 4) 1, 2, 2
1 IWHTM = IWHTM -XSIGNF (10, ITEM)
1 BLMKM = BLMKM +XSIGNF (10, ITEM)
RETURN
END
* LABEL
* FAP
 COUNT 31 .
* ALIAS, UPDATE, REVERT, CCOL, SETUP
 ENTRY UPDATE
 ENTRY REVERT
 ENTRY CCOL
 ENTRY SETUP

UPDATE SXD UPDATE-2,4
 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 ""1
 ONE PZE ""2
 TWO PZE ""3
 THR PZE ""4
 FOR PZE ""5
 MIN3 MZE ""3
 MIN PZE END
* LABEL

** LISTB

C UPREV CHESS SUBROUTINE, 2/26/62, MINOR REVISION
SUBROUTINE UPREV(MIN, M6)

C DIMENSION AND EQUIVALENCE STATEMENTS
DIMENSION IOCC(64), LOC(32), NFIRST(32), NUMB(32),
INTYPE(50), IBEG(32), IEND(32), MOVE(50), ICAPT(150),
2MOVEFR(150), MOVEP(150), JBEAR(1024), JBEAR(64,16),
3KIND(32), MSVN(16), IPDIR(132), IEXTD(16), IEXTS(64),
4M64H116, NMNV(6), IUPP(16)
DIMENSION JPAWN(6)
DIMENSION MSTD(32)
DIMENSION MAVAIL(100), ITCH(2), ITCHD(2), IPIN(32)
DIMENSION NEEP(10), MEEP1(10), MEEP2(10)
DIMENSION JPRM(4)
DIMENSION LOGG(101)
DIMENSION NZZZ(120)
DIMENSION KVAIL(64), KFORCE(64), KWORTH(64)
COMMON STATEMENTS
COMMON IPDIR, IUPP, IEXTD, IEXTD, JPAWN, M64H116, MSTD, NMNV, MSTD, JPRM,
1JBEAR, JBEAR, KIND, IEND, IBEG, IOCC, LOC, NFIRST, MOVE, IENUS, MOVEP,
2MOVEFR, I.cpp, NUMO, INTYPE, ITCH, ITCHD, IPIN, NEEP, MEEP1, MEEP2, LOGG, NLOG,
3NZZZ, NUMTES, MAVAIL, 12, 1Y, IX, IJ, IT, ISPEC, IR, IQ, JPRM, IUPPD, INTER,
4IDIR, ICHECK, IAK, IA, JB, J, JDIR, JD, JL, JF, JIN, JJ, JROOK, JK, KD,
5K, L2, L, M, MAREI, ICAPT, MCOL, MIN, MOVDIR, MOVENO, MOVES, MOVETO, M6.
6MV, N1, N2, NEWSU, NSPEC, NUMEP, NPRINT, KIN, KVAIL, KFORCE, KWORTH, MOVES,
7MOBB, MATW, MATB
EQUIVALENCE (IENUS, IBEG(33)), (NLOG, LOGG(101)), (NUMTES, NZZZ(120)),
1(JBEAR, JBEAR)
DIMENSION NUMBER(64)
COMMON NUMBER
COMMON MLOG

C C

C MAIN PROGRAM

C

MCOL = MCOL
GO TO (120, 150, 700, 200), P6
C CHANGE COLOR OF SIDE TO MOVE
200 MCOL = 3 - MCOL
MIN = - MCOL
MOVENO = MOVENO + 1
MOVEP(MOVENO) = - 1
GO TO 700
C MIN IS THE MOVE MADE
120 MOVETO = XMVF(MIN)
MOVDIR = XMVF(MIN)
MOVER = XMVF(MIN)

C C

C SET UP VARIABLES FOR UPDATE

C 130 MO = LOC(MOVER)
KD = KIND(MOVER)
MOVENO = MOVENO + 1
MCOL = 1 + XLBITF(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

ICHECK = 0
KLOC = LOC(MCOL)
DO 701 IA=1,32

IS THE KING IN CHECK. LIST PINS.
DO 719 K=1,16

IS THE KING SUBJECT TO CAPTURE BY THE OTHER SIDE
IF (XBEAR(KLOC,K)) 721,721,718
HAS THE BEARER THE SAME COLOR AS THE KING.
718 IF (XLBITF(XBEAR(KLOC,K)+MCOL)) 750,716,750

THE KING IS IN CHECK.
750 ICHECK = ICHECK + 1
ITCH(ICHECK) = IECC(KLOC,K)
ITCHD(ICHECK) = IPOP(K)
IF(ICHKECK = -1) 719,731,731

KNIGHTS CANNOT PIN
721 IF(K = 8) 722,722,715
722 IQ = XGETF(IPOP(K)+IEXTD)
IZ = IR

LOOK FOR OCCUPIED SQUARE ALONG LINE FROM KING

728  IZ = IZ + IQ

NEWSQ = XMOVF(IZ)

719  IF (NEWSQ = 64) 719, 719, 719

IF (IOCC(NEWSQ)) 728, 728, 728

AN OCCUPIED SQUARE IS FOUND

716  NEWSQ = XGETF(IBEAR(KLOC, K), LOC)

FIND WHAT IF ANYTHING BEARS FROM OPPOSITE DIRECTION

727  IU = IBEAR(NEWSQ, K)

IF (IU) 715, 715, 726

IF BEARER IS A LONG RANGE PIECE OF OPPOSITE COLOR WE GET A

PIN.

726  IF (1 - XLBITF(IU + MCOL) + XLBITF(KIND(IU))) 715, 732, 715

LIST A PIN

732  IT = IOCC(NEWSQ)

IPIN(IT) = K

715  CONTINUE

PUT MOVES OF KINGS IN MAVAIL TABLE

FIRST NON-CASTLING MOVES

731  DO 705 IDIR = 1, 8

IF (XGETF(M + IDIR, MOVE)) 705, 705, 706

706  NEWSQ = XMVIF(XGETF(M + IDIR, MOVE))

THE KING CANNOT MOVE ALONG THE LINE OF CHECK,

UNLESS THE CHECKER IS A PAWN.

IF (ICHECK) 753, 758, 753

753  DO 751 JA = 1, ICHECK

IF (ITCHD(JA) = IOPP(IDIR)) 751, 752, 751

752  IF (XGETF(ITCH(JA), KIND) = -1) 705, 751, 705

751  CONTINUE

708  DO 712 K = 1, 16

IF (IBEAR(NEWSQ, K)) 712, 712, 713

713  IF (XLBITF(IBEAR(NEWSQ, K) + MCOL)) 705, 712, 705

712  CONTINUE

K1 = K1 + 1

MAVAIL(K1) = XGETF(M + IDIR, MOVE)

709  CONTINUE

ARE THERE CASTLING MOVES

NOT IF KING IS IN CHECK OR HAS MOVED

IF (ICHECK + NFIRST(MCOL)) 800, 736, 800

FOR EACH ROOK

736  DO 737 IDIR = 1, 3, 2

DOES A ROOK WHICH HAS NEVER MOVED BEAR ON THE KING

IF (IBEAR(KLOC, IDIR)) 739, 739, 739

739  JROOK = IBEAR(KLOC, IDIR)
IF(KIND(JROOK)-2+NFIRST(JROOK))737,738,737

C ARE THE INTERMEDIATE SQUARES COVERED BY THE FOE

738 JB=IDIR-2
JD=KLOC

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(XLBBTF(MCOL+JE))737,742,737
742 CONTINUE
741 CONTINUE

C CASTLING OK
K1=K1+1
MAVAIL(K1)=JD+XGETF(IOPP(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(LOC(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)))815,816,815
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(XABSF(XMV2F(MOVE(J))-IDIR),XABSF(XMV2F(MOVE(J))-IOPPD)))
1812,814,812
814 K1=K1+1
MAVAIL(K1)=MOVE(J)
812 CONTINUE
CONTINUE

C ADJOIN EN PASSANT MOVES IF ANY

IF(NUMEP)860,143,860
860 IF(INEP(NUMEP)-MOVENO)143,850,143

850 JJ=1
851 J1 = MEP1(NUMEP)
GO TO 853
852 J1 = MEP2(NUMEP)
IF(J1)853,143,853

C IS THE EN PASSANT MOVE PREVENTED BY A PIN
853 IF(XGETF(XMV3F(J1),1,IPIN))854,855,854

C PINNED. WHAT ABOUT THE DIRECTION.
854 IF(XMINOF(XABSF(XGETF(XMV3F(J1),1,IPIN)=XMV2F(J1)),XABSF(XGETF(XGETF
1(XMV3F(J1),1,IPIN),10PP)-XMV2F(J1))))856,855,856

C NO PIN ON MOVE. WILL REMOVAL OF CAPTURED PAWN PUT US

C IN CHECK.
855 IF (XRANKF(KLOC)-XRANKF(XGETF(XMV3F(J1),1,LOC)))856,855,856

C KING ON SAME RANK AS PAWNS. REFERENCES TO PUTCHE ARE NEEDED
C TO REMOVE PAWNS FROM POSSIBLE LINE OF ACTION.

857 J1OCC=XMV3F(J1)
J1LOC=LOC(J1OCC)
J2=XMV2F(J1)
J3=XMOMF(IEXTS(J1LOC)+XGETF(4-XABSF(13-J2-J2),1,EXTD))
J3OCC=I1OCC(J3)
CALL PUTCHE(J1OCC,0)
CALL PUTCHE(J3OCC,0)
DO 864 K=1,3,2
IF (IBEAR(KLOC,K))864,864,861

861 IF (XLBINT(IBEAR(KLOC,K)+MCOL)) 864,864,862
864 CONTINUE

J4=0
GO TO 863

862 J4=1
863 CALL PUTCHE(J1OCC,J1LOC)
CALL PUTCHE(J3OCC,J3)
IF (J4) 856,858,856

C PUT EN PASSANT MOVE IN MAVAIL.
858 K1 = K1 + 1
MAVAIL(K1) = J1
856 JJ=JJ+1
GO TO 859

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(I1CHD(J1),1,EXTD)
N = IEXTS(KLOC)

C LOOP WHICH LOOKS ALONG CHECK LINE
C LOOK AT SQUARES IN DIRECTION OF CHECK

834 N = N+M
836 N1 = XMOMF(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 CHECKER
C     IS IT PINNED
828   INTER = IBEAR(N1,IDIR)
   IF(IPIN(INTER)) 826,826,826
C     NOT PINNED
   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(IOCC(N1)) 826,838,826
C     IS THERE AN INTERVENING OCCUPIED SQUARE
833   IF(IOCC(N1)) 830,826,830
C     CONSTRUCT MOVE.
830   K1 = K1 + 1
   MAVAIL(K1) = MAVAIL(INTER) + MAVAIL(IBEAR(N1,IDIR)) + N1
826   CONTINUE
   IF (IOCC(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(XGETF(LOC(INTER),1) + EXTS) + EXT(IIDIR), IOCC))
840   IF (NEP(NUMEP)-MOVENO) 825,840,825
844   IF (XGETF(XMOVF(NEP1(NUMEP),1) PIN)) 845,840,845
841   K1 = K1 + 1
   MAVAIL(K1) = MPE1(NUMEP)
845   IF (MPE1(NUMEP)) 846,825,846
846   IF (XGETF(XMOVF(MPE2(NUMEP),1) PIN)) 825,842,825
842   K1 = K1 + 1
   MAVAIL(K1) = MPE2(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(MOVER)) 402,402,412
402   NFIRST(MOVER)=1
   NSPEC=NSPEC+1
   NUMB(NSPEC)=MOVENO
   NTYPE(NSPEC)=1
   IF (XTRANKF(MOVER,MOVER)-4) 134,403,134
C     2ND RANK TO 4TH LOOK TO SIDES
DO 405 J=1,2
  IX=XM0VF(IXE1TS(M0VETO)+IEXTD(2*J-1))
  IF(IX=64) GO TO 404
404  IY=IOCC(IY)
    IF(IY=405) GO TO 407
407  IF(KIND(IY)-1)=405 GO TO 408
408  IF(XLB1TF(IY+MOVEM))=405 GO TO 409
C THERE IS AN EN PASSANT TBY
409  IZ = IBEQ(IY)+J-1
    IF(NEP(NUMEP)-MOVNO) GO TO 420
420  NUMEP=NUMEP+1
    N0P(NUMEP)=MOVNO
    MEP1(NUMEP) = XABSF(MOVE(1Z))
    GO TO 405
421  MEP2(NUMEP) = XABSF(MOVE(1Z))
    CONTINUE
    GO TO 134
C IS THIS MOVE A PROMOTION
412  IF(IADDFin(MIN))=19,418,419
C NOT A PROMOTION, IS IT AN EN PASSANT CAPTURE
418  IF(MOVDIR-4) = 13,413,413
413  IF(IOCC(MOVETO)) = 136,413
C DIAGONAL MOVE TO EMPTY SQUARE
416  IX=XM0VF(IXE1TS(MW)+XGETF(1+XABSPT (13-MOVDIR-MOV0IR),1EXTD))
    N8PEC=NSPEC+1
    NUMB(NSPEC)=MOVNO
    N8TYPE(NSPEC)=IX
    ICAPT(MOVNO)=-IOCC(IY)
    CALL PUTCH(IOCC(IY),0)
    GO TO 134
419  IPROM = IADDf(MiN)
    KIND(MOVER)=IPROM
    IF(XLB1TF(MOVER))=423,423,423
422  MATW=MATW+KVAL(IPROM)-1
    GO TO 424
423  MATB=MATB+KVAL(IPROM)-1
424  NSPEC=NSPEC+1
    NUMB(NSPEC)=MOVNO
    N8TYPE(NSPEC)=-1
    IEND(MOVER)=IBEG(MOVER)+NM0V(IPROM)-1
    GO TO 134
C C HANDLES FIRST MOVE OF KING AND
C MAKES CASTLING MOVES
C
460  IF(NFIRST(MOVER))=13,462,134
462  NFIRST(MOVER)=1
    NSPEC=NSPEC+1
    NUMB(NSPEC)=MOV0 NO
    N8TYPE(NSPEC)=1
C TEST FOR CASTLING MOVE
    IF(XABSf(MOVETO-MQ))=2,134,463,134
463  IF(MOVETO-MO)464,466,466
C CASTLE QUEENS SIDE
464   IA=-4+MQ
465   JA=-1+MQ
466   CALL PUTC(MOVER,MVETO)
467   CALL PUTC(IAA,JA)
468   NTYPE(NSPEC)=-(IA-1+MSTO(IAA))
469   GO TO 139

C REVET TAKES BACK MOVES

150   IF (MOVEP(MOVENO)) 201,201,167
151   MCOL=3-MCOL
152   MIN=-0
153   GO TO 165

C NORMAL REVERSION
154   MOVETIMEP(MOVENO)
155   MOVETO=MOVEFR(MOVENO)
156   ISPEC=0
157   MIN = 0
158   C IS THIS A SPECIAL MOVE
159   IF(NUMB(NSPEC)-MOVENO,152,151,152)
160   C SPECIAL MOVE
161   ISPEC=NTYPE(NSPEC)
162   NUMB(NSPEC)=0
163   NTYPE(NSPEC)=0
164   NSPEC=NSPEC-1
165   C SET UP VARIABLES
166   MQ=LOC(MOVER)
167   MCAIPT = ICAPT(MOVENO)
168   ICAPT(MOVENO) = 0
169   MCOL = 2-XLB1TF(MOVER)
170   KD=KIND(MOVER)
171   C ORDINARY OR SPECIAL MOVE
172   IF(ISPEC)153,154,154
173   C SPECIAL CASTLING OR PROMOTION
174   IF(ISPEC+1)155,156,155
175   C CASTLING
176   MVR=XMV3F(ISPEC)
177   NFIRST(MVR)=0
178   NFIRST(MOVER)=0
179   CALL PUTC(MVR,XMV1F(ISPEC))
180   GO TO 154

C PROMOTION
181   IF (XLB1TF(MOVER)) 168,169,169
182   MATW=MATW-XGETF(KIND(MOVER),KVAL)+1
183   GO TO 170
184   MATB=MATB-XGETF(KIND(MOVER),KVAL)+1
170 KIND(MOVER)=1

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 PUTC(MOVER, MOVETO)
C WAS THE MOVE A CAPTURE OR EN PASSANT CAPTURE
IF (MCAPT) 158, 162, 160
C EN PASSANT CAPTURE
158 CALL PUTC(-MCAPT, ISPEC)
GO TO 162
C ORDINARY CAPTURE
160 CALL PUTC(MCAPT, MQ)
C
C IS THERE AN EN PASSANT POSSIBILITY
162 IF (NEP(NUMEP)-MOVENO) 165, 166, 165
C YES, AT LEAST ONE
165 NUMEP=NUMEP-1
NEP(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
LIST8
SUBROUTINE PUTC'H (M6,M7)
DEC. 2, 1960, KOTOK, LIEBERMAN AND NISSLER.

DIMENSION AND EQUIVALENCE STATEMENTS
DIMENSION IOCC(64), LOC(32), NFIRST(22), NUMB(50),
INTYPE(50), IBEG(33), IEND(32), MOVE(504), ICAPT(150),
2MOVEFR(150), MOVEP(150), JBEAR(1024), JBEAR(64,16),
3KIND(32), MSVN(16), IPDIR(32), IEXTD(16), IEXTS(64),
4M64M1(16), NMOV(6,10), IOPP(10)
DIMENSION JPWN(8)
DIMENSION MSTD(32)
DIMENSION MAVAIL(100), ITCH(2), ITCUD(2), IPIN(32)
DIMENSION NEP(10), MEP1(10), MEP2(10)
DIMENSION JPROM(4)
DIMENSION LOGG(101)
DIMENSION NZZZ(120)
DIMENSION KVAL(6), KFORC'E(64), KWORTH(64)

COMMON STATEMENTS
COMMON IPDIR, IOPP, IEXTD, IEXTS, JPWN, M64M1, MSVN, NMOV, MSTD, JPROM,
1IBEAR, JBEAR, KIND, IEND, IBEG, IOCC, LOC, NFIRST, MOVE, IENUS, MOVEP,
2MOVEFR, ICAPT, NUMB, INTYPE, ITCH, ITCUD, IPIN, NEP, MEP1, MEP2, LOGG, NLOG,
3NZZZ, NUMTES, MAVAIL, Z2, Y2, X2, IT, ISPEC, IR, IQ, IPROM, IOPPD, INTER,
4IDIR, ICHECK, IA, IAA, AJ, JA, JB, JC, JDIR, JD, JE, JF, JIN, JJ, JROOK, J, K1, KO,
5K, L2, L4, MAREI, IACAPT, MCOU, MIN, MOVIK, MOVENT, MOVER, MOVENTO, MQ, M,
6MV, N, NZ, NEWS, N, NSPEC, NUMEP, NPRINT, KIN, KVAL, KFORCE, KWORTH, MOBW,
7MOBB, MATB, MATB
EQUIVALENCE (IENUS, IBEAR(33)), (NLOG, LOGG(101)), (NUMTES, NZZZ(120)),
1(IBEAR, JBEAR)
DIMENSION NUMBER (64)
COMMON NUMBER
COMMON ALOG

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

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

IF (MOLDSQ) 503, 523, 503
C ADD NEW PIECE TO MATERIAL COUNT
523 IF (XLBITF(MVR)-1 = 530, 531, 532
532 STOP 532
531 MATH = MATW + XGETF(KIND(MVR), KVAL)
GO TO 516
530 MATH = MATB + XGETF(KIND(MVR), KVAL)
C A PIECE COMING FROM THE BOARD MAY NEED MOVE STORAGE
516 IF(IBEAR(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 = XDELF(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
    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)=53)515,512,512
515 MNREQ=56
    GO TO 600
C   UPDATE MOVES OF PIECE IN ALL DIRECTIONS. DATUM IS MTOP
512 MTOP = MVR
200 NOLDSQ=LOC(MTTOP)
    MSTOP = MSTOP(MTTOP)
    K=KIND(MTTOP)
    GO TO (210,220,230,240,224,260),K
C
C   ROOK IN ALL DIRECTIONS
220 ASSIGN 221 TO JRET
    DO 221 IDIR=1,4
        L=IBEG(MTTOP)+MSVN(IDIR)-6
        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(MTTOP)+MSVN(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
C
  KING IN ALL DIRECTIONS
  N1=1
  GO TO 232

C
  N IN ALL DIRECTIONS
  N1=9
  N2=N1+7
  L3=IBEG(MTOUP)-N1
  DO 271 IDIR=N1,N2
       L=L3+IDIR
       N IN GIVEN DIRECTION
       DATA ARE MTOUP, IDIR, NOLDSQ
       L1=M64M1(IDIR)+MSTOP
       NEWSQ=XMOVF(IEXTS(NOLDSQ)+IEXTD(IDIR))
       IS THE SQUARE ON THE BOARD
       IF(NEWSQ=64)272,272,271
       ON BOARD
       IF (IBEAR(NEWSQ, IDIR)) 279,279,268
       L10=XLBITF(IBEAR(NEWSQ, IDIR))
       MOBW=MOBW-L10
       MOBB=MOBB-1+L10
       L2=XLBITF(MTOUP)
       MOBW=MOBW+L2
       MOBB=MOBB-L2+1
       IBEAR(NEWSQ, IDIR)=MTOUP
       IS THE SQUARE OCCUPIED
       IF(IOCC(NEWSQ))274,274,277
       STOP275
       OCCUPIED, IS THE COLOR THE SAME AS THAT OF THE MOVER
       IF(XLBITF(IOCC(NEWSQ)-MTOUP))276,276,276
       MOVE(L)=NEWSQ+L1
       GO TO 271
       MOVE(L)=-(NEWSQ+L1)
       CONTINUE
       GO TO 201

C
  UPDATE MOVES OF PAWN IN ALL DIRECTIONS
  210-217 AND 320-320
  PURPOSE- TO UPDATE THE MOVES OF A PAWN IN ALL DIRECTION.
  ASSIGNS ADDITIONAL STORAGE TO PAWNS REACHING THE 7TH RANK.
  DOES NOT SET UP EN PASSANT MOVES. USES 600, XLBITF, XMOVF,
  XRANKF, IPDIR, NFIRST, IEXTS, IEXTD, IOCC.
  TABLES AFFECTED- MOVE, IBEG, IEND, IBEAR,
  LOCAL VARIABLES- J,K, L, JRET, MPREQ, MNREQ, K1 NEWSQ, IDIR, L1,
  AND L2
  DATA SUPPLIED - MTOUP, NOLDSQ, IENUS(INITIALLY)
  210 K=XLBITF(MTOUP)+1
  L9 = IBEG(MTOUP)-1
  DO 211 J=1,3
       IDIR=IPDIR(J,K)
L = L9 + J
ASSIGN 211 TO JARET
GO TO 320
211 CONTINUE
GO TO 201
201 MSQ=MTQ
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 MATB=MATB-XGETF(KIND(MVR),KVAL)
GO TO 508
527 STOP 527
528 MATW=MATW-XGETF(KIND(MVR),KVAL)
508 IF (MOLDSQ) 511, 509, 511
511 MSG=MOLDSQ
ASSIGN 509 TO MRET
GO TO 300
509 RETURN
C MOVE STORAGE CONTROL 600 TO 625
C PURPOSE: TO EXPAND AND CONTRACT THE MOVE
C STORAGE ALLLOTTED TO PAWNS WHEN THEY
C REACH THE 7TH RANK OR REVERT TO IT
C TABLES AFFECTED: MOVE, 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, 602
C STORAGE AVAILABLE AT THE END
602 IF (IBEG(MVR)) 604, 604, 605
C MOVE THE MOVE INFORMATION
605 M1=IENUS+1
M=IBEG(MVR)
N=IEND(MVR)
DO 606 J6=M, N
MOVE(M1)=MOVE(J6)
MOVE(K6)=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, 608, 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-XTRANKEF(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)) 612,611,609
C HAS J6 ALREADY BEEN RE-ARRANGED.
612 IF(M2=IBEG(J6)) 613,611,617
613 IF(M2=IBEG(J6)) 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(M)=MOVE(J6)
MOVE(J6)=0
624 M1=M1+1
IEND(K)=M1-1
GO TO 620
C STORAGE COMPLETELY RE-ARRANGED
622 IENUS=M1-1
IF(504-IENUS-MNREQ) 625,602,302
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=XABS(IIBMAR(MSQ,IDIR))
MSTOP = MSTO(MTOUP)
K=KIND(MTOUP)
NOLDMSQ=LOC(MTOUP)
ASSIGN 301 TO JREL
GO TO (313,310,314,312,315,310),K
C MOVE 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 (MVR-MTOUP) 311,301,311
311 IF (XLBITF(MVR-MTOUP)) 301,316,301
316 M=IBEG(MTOUP)+IDIR-N1
MOVE(M)=-MOVE(L)
301 CONTINUE
GO TO MRET,(508,509)
C
UPDATE ROOK OR QUEEN IN GIVEN DIRECTION
310 L=IBEG(MTOUP)+MSVN(IDIR)-6
GO TO 280
C
UPDATE BISHOP IN GIVEN DIRECTION
312 L=IBEG(MTOUP)+MSVN(IDIR)-36
GO TO 280
313 ASSIGN 301 TO JARET
J=JPAWN(IDIR)
L=IBEG(MTOUP)+J-1
GO TO 320
C
UPDATE G, 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)284,284,283
284 IF(1BEAR(NEWSQ,IDIR)=282,282,299
299 L10=XLBITF(1BEAR(NEWSQ,IDIR))
MOBW=MOBW-L10
MOBB=MOBB+L10
MOBB=MOBB+L10
L10=1BEAR(NEWSQ,IDIR)=MTOUP
J1=L+J
289 IF(1OCC(NEWSQ)=285,281,287
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(1OCC(NEWSQ)-MTOUP)=290,291,290
290 MOVE(J1)=NEWSQ+L1
GO TO 292
291 MOVE(J1)=-(NEWSQ+L1)
292 IF(J-6)=292,292,251
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(F1BEAR(NEWSQ,IDIR))-MTOUP)=294,298,294
298 1BEAR(NEWSQ,IDIR)=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
C PAWN ON THE 7TH RANK.
C USES-XLBITF, XMOVF, IEXTS, IEXTD, M64M1, 1OCC, NFIRST.
TABLES AFFECTED = IBEAR, MOVE.
LOCAL VARIABLES = NEWSQ, L1, L2
DATA SUPPLIED = NOLDSQ, IDIR, MSTOP, MTOUP, JARET, L, J.

320 NEWSQ = XMOVF (IEXTS (NOLDSQ) + IEXTD (IDIR))
IF (NEWSQ = 64) 321, 321, 322
322 GO TO JARET (211, 301)
321 L1 = M64M1 (IDIR) + MSTOP
L3 = XLBITF (MTOUP)
IF (IBEAR (NEWSQ, IDIR)) 342, 342, 343
343 L10 = XLBITF (IBEAR (NEWSQ, IDIR))
MOBW = MOBW - L10
MOBB = MOBB - 1 + L10
342 IBEAR (NEWSQ, IDIR) = MTOUP
MOVE (L) = NEWSQ + L1
L2 = IUCG (NEWSQ)
IF (J = 3) 330, 323, 323
C MOVE IS DIAGONAL
330 MOBW = MOBW + L3
MOBB = MOBB - L3 + 1
IF (L2) 328, 328, 326
326 IF (XLBITF (L2 + MTOUP)) 328, 328, 350
328 MOVE (L) = -MOVE (L)
C PROMOTION POSSIBILITIES MAY HAVE BEEN SETUP
IF (XTRANKF (NOLDSQ, MTOUP) - 7) 338, 338, 338
C MOVE IS VERTICAL
323 IBEAR (NEWSQ, IDIR) = -XABSF (IBEAR (NEWSQ, IDIR))
IF (L2) 331, 331, 332
C CAN WE MOVE TWO SQUARES
331 IF (NFIRST (MTOUP)) 334, 334, 335
335 MOVE (L + 1) = 0
350 IF (XTRANKF (NOLDSQ, MTOUP) - 7) 338, 338, 338
C MAY BE ABLE TO MOVE TWO SQUARES
334 NEWSQ = XMOVF (IEXTS (NEWSQ) + IEXTD (IDIR))
IBEAR (NEWSQ, IDIR) = -MTOUP
MOVE (L + 1) = -XSIGNF (L1 + NEWSQ, IUCG (NEWSQ) - 1)
338 GO TO JARET (211, 301)
C REMOVE POSSIBLE FALSE BEARING
332 MOVE (L) = -MOVE (L)
MOVE (L + 1) = 0
IF (NFIRST (MTOUP)) 350, 359, 350
339 NEWSQ = NEWSQ + 24 - 8 + IDIR
IF (IBEAR (NEWSQ, IDIR)) 338, 341, 338
341 IBEAR (NEWSQ, IDIR) = 0
GO TO JARET (211, 301)
C IF ON THE 7TH RANK MOVES MUST BE DUPLICATED
COPY MOVES
353 MOVE (L + 4) = MOVE (L) + XSIGNF (JPRM (2), MOVE (L))
MOVE (L + 8) = MOVE (L) + XSIGNF (JPRM (3), MOVE (L))
MOVE (L + 12) = MOVE (L) + XSIGNF (JPRM (4), MOVE (L))
MOVE (L) = MOVE (L) + XSIGNF (JPRM (1), MOVE (L))
GO TO JARET (211, 301)
C END
CONLINE CHESS MAIN PROGRAM, FEB. 28, 1962

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

COMMON FOO

EQUIVALENCE(NSPEC, FOO(2913))
EQUIVALENCE(FOO(2892), K1, FOO(1463), KIND, FOO(2765), MAVAIL)
EQUIVALENCE(FOO(2900), MCOL)
EQUIVALENCE(FOO(2703), IPIN, FOO(1265), IOPP, FOO(255), IBEAR)
EQUIVALENCE(FOO(1365), IEXTD, FOO(1301), IEXTS, FOO(1527), IOCC)
EQUIVALENCE(FOO(1591), LOC, FOO(1623), NFIRST, FOO(3003), KPAWN)
EQUIVALENCE(FOO(3121), PLY, FOO(3120), IPE, FOO(2917), KVAL)
EQUIVALENCE(FOO(3051), MUBW, FOO(3052), MUBB, FOO(3123), IHOPE)
EQUIVALENCE(FOO(9188), IEXCH, FOO(3122), BACK, FOO(3187), LISP)
EQUIVALENCE(FOO(3053), MATW, FOO(3054), MATB, FOO(3119), MLOG)
EQUIVALENCE(FOO(134), NLOG)
EQUIVALENCE(FOO(2903), MOVENC)
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 (213)
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 OPPONE
1NT. 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. 89H IF
3 AT ANY TIME, YOU WISH TO START OVER, SET ADDRESS OF KEYS N0N ZER
4. THEN PRESS START. 30H KEYS NEGATIVE PRINTS HISTORY. 1H)

PAUSE
IF (KEYS(J)) 3, 3, 2
IF (J) 4, 4, 5

4 WRITE OUTPUT TAPE 100, 7
7 FORMAT (14HUMACHINE 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(MAVAIL(J))
CALL PRINT (-7)

10 WRITE OUTPUT TAPE 100, 9
9 FORMAT (9SH0 IF THIS MOVE IS CORRECT, SET KEYS TO ZERO AND PRESS STA
IRT. OTHERWISE SET KEYS TO CORRECT MOVE. /1H1

PAUSE

IF(KEYS(I)) 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(6HODARN, /4H1HCARE TO TRY AGAIN... PRESS START IF SO./1H1)
PAUSE
GO TO 2

18 L = XTIMEF(L)
CALL TREE (MOVE)
TIME = XLAPSEF(L)
CALL UPDATE(MOVE)

8 CALL PRINT (407777000000)
33 IF(K1)20, 16, 17
20 WRITE OUTPUT TAPE 100, 21
21 FORMAT(16HHOWHOOPPEE, I WIN./43H1HCARE TO LOSE AGAIN... PRESS START
1 IF SO./1H1)
PAUSE
GO TO 2

17 WRITE OUTPUT TAPE 100, 22, TIME
22 FORMAT (24HOTHE PRECEDING MOVE TOOK, -1PF4.1, 9H MINUTES./42HOPLEA
1SE KEY IN YOUR REPLY AND PRESS START.)
RESEND 7
NLOG = 0
MLOG = 0
25 PAUSE

IF(KEYS(J)) 69, 23, 2
23 IF(J) 69, 69, 5
C ERROR PSEUDO STOP
69 WRITE OUTPUT TAPE 100, 691
691 FORMAT(25H1ILLEGAL MOVE, TRY AGAIN./1H1)
GO TO 25
C START OVER
2 IF (SENSE SWITCH 3) 709, 7090
7090 BACKSPACE 4
BACKSPACE 4
8709 CALL PRINT (77/40000000)
RESEND 7
MLOG = 0
NLOG = 0
GO TO 26
S1002 CALL PRINT (410000000000)
GO TO 1003
END
*  LABEL
*  FAP
  COUNT 354
*  FUNCTION INITIAR, M179 CHESS, APR, 17, 1961
ENTRY INITIAR
INITIAR SXD  XR4,4
      SXA  XR2,2
      SXA  XR1,1
      STI  INDIC
      CLA  1,1
      TZE  A1342
      AXT  32,1
LP32  STZ  IBEG+1,1
LP32  STZ  IEND+1,1
LP32  STZ  LOC+1,1
LP32  STZ  IPIN+1,1
LP32  STZ  LOCIN+1,1
                TIX  LP32+1,1
LP32  AXT  100+1
LP100  STZ  MAVAIL+1,1
LP100  TIX  LP100+1,1
LP100  STZ  IENUS
LP100  AXT  22,1
LP22  STZ  NFIRST+1,1
LP22  TIX  LP22+1,1
LP22  AXT  50,1
LP50  STZ  NUMB+1,1
LP50  STZ  NTYPE+1,1
LP50  TIX  LP50+1,1
LP50  AXT  64,1
LP64  STZ  IDOC+1,1
LP64  PXD  1
LP64  STO  NUMBER+1,1
LP64  TIX  LP64+1,1
LP64  AXT  504,1
LP504  STZ  MOVE+1,1
LP504  TIX  LP504+1,1
LP504  AXT  150,1
LP150  STZ  ICAPT+1,1
LP150  STZ  MOVFR+1,1
LP150  STZ  MOVEP+1,1
LP150  TIX  LP150+1,1
LP150  STZ  MATW
LP150  STZ  MATB
LP150  STZ  MOBWWW
LP150  STZ  MOBB
LP150  STZ  NUMEP
LP150  STZ  ISPE\textcircled{C}
LP150  STZ  NSPE\textcircled{C}
LP150  STZ  MOVENO
LP150  AXT  1024,1
LP1024  STZ  JBEA+1,1
LP1024  TIX  LP1024+1,1
LP1024  STZ  ITCH
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 =1B17
AXT 7+1

LP722
STO KIND+1,1
TXI 1+1,1,1
TXL LP722+1,2

INPUT
CLA =1B17
STO LOC1
STO COLOR
AXT INS+1,4
SXA INS,4
STZ LETTER
AXT 0+2

CARD
CAL =4B17
TSX $(TSH),4
PZE =M(12A61)
AXT 12,1

STR
STQ TABLE+12,1
TIX *-2,1,1
TSX $(RTN),4
CAL TABLE
LAS =HFORTA
TRA *+2
TRA B1254
AXT 12,1

FORTRAN READ INPUT TAPE 4

B
AXT 6+4

A
SXA CHLOOP,4

INS

TRA *
CAS =H0000


TRA *+2
TRA PERIOD
CAS =H0000
TRA *+2
TRA CHLOOP
CAS =H0000*
TRA *+2
TRA COLOR1
CAS =H000009
TRA *+3

NOP

TRA NUMBUH
CAS =H000001
TRA *+2

READ PROBLEM
READ IN ANOTHER CARD
FORTRAN READ INPUT TAPE 4
WORD COUNT
CHARACTER COUNT
BLANK
BLANKS IGNORED
NUMERAL
OPEN PARENTHESIS
TRA OPEN
CAS =H000000
TRA ++2
TRA CLOSE
CAS =H000000
TRA ++2
TRA BREAK
CAS =H000000
TRA ++2
TRA BREAK
CAS =H000000
TRA ++2
TRA COMMENT
ADD LETTER
Q OR K BEGINS A NEW PIECE

SHIFT ALS 6
STO LETTER
TXI CLOOP 2,1
COLOR1 TSX LOOKUP 4
STZ COLOR
TRA RESETL
INCREASE LETTER COUNT

81234 CAL =4B17
TSX $(BST) 4
PX D
LXD XR 4,4
TRA A1342
NUMBUH STO NUM
TSX LOOKUP 4
CLA NUM
ALS 18
ADD LOC1
CAS =69B17
TSX ERROR 4
NOP STO LOC1
RESET CHARACTER COUNTER

RESETL AXT 0,2
STZ LETTER
AXT INS 1,4
SXA INS 4
CHLOOP AXT **, 4
TX IX A**, 4
TX IX B**, 1
TRA CARD
READ ANOTHER CARD
LOOKUP CLA LETTER
TXL FOUND 1,2, 0
TXL ONE 2, 1
TXL TWO 2, 2
TXL THREE 2, 3
TSX ERROR 4
CLOSED SUBROUTINE TO LOOKUP PIECE

THREE ALS 12
TRA PLACE
NORMALIZE

TWO ALS 18
ORA =H00 000
TRA PLACE
ONE ALS 18

ORA =H 0 000
PLACE ORA =H000--
ZET COLOR
ORA =H000
AXT 32+2
LAS PIECES,2
TRA ++2
TRA FOUND
TXI *=2,2,1
TSX ERROR,4
FOUND CLA LOCI
CAS =65817
NOP
TSX ERROR,4
ZET LOCIN+1,2
TSX ERROR,4
STO LOCIN+1,2
ADD =1B17
STO LOCI
STL COLOR
TXH FOUND1,2,22
CLA LOCBEG+1,2
SUB LOCIN+1,2
TZE ++2
CLAl =1B17
SSP
STO NFIRST+1,2
FOUND1 TRA 1+4
COMENT AXT COMEN1,4
TRA CHLOOP-1
COMEN1 CAS =H00000/
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
RIL 7
CAS =H0000R0
TRA ++2
LDI =2B17
CAS =H0000B0
TRA ++2
LDI =4B17
INCREMENT LOCATION COUNTER
SET UP NFIRST TABLE
PROMOTED PIECE HANDLED HERE
(M) MEANS PIECE HAS MOVED

LOOK FOR END OF PROBLEM
CLA*  1,4
SUB   =1B17
STO*  1,4
TN2   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  PUTCH,JIN,LOCIN
PTCHLP TXI  **+1,1,1
TXL   PTCH,1,32
CALL  SETUP
LXD   XR4,4
CLA*  1,4
AI342 SUB   =1B17
STO*  1,4
XR1   AXT   **,1
XR2   AXT   **,2
LDI   INDIC
TRA   2,4
BCI   1,6;
BCI   1,1H012A
BCI   1,1CAR012
BCI   1,1OWING
BCI   1,1N FULL
BCI   1,1OUND O
BCI   1,1RROR F
BCI   1,132H, E
BCI   1,1C1=14,
BCI   1,17H, LO
BCI   1,1J=14,
BCI   1,1ATIVE
BCI   1,13H REL
BCI   1,0NO6,1
BCI 1. LOCAT I
BCI 1. IA AT
BCI 1. Y INIT
BCI 1. OUND 0
BCI 1. RROR F
ERFOR BCI 1. 134H4E
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
MFAIL SYN R+9797
IENUS SYN R+12529
NFIRST SYN R+10939
NUMB SYN R+9963
NTYPE SYN R+9913
IOCC SYN R+11039
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+9862
PIECE SYN R+9624
LOCBEG SYN R+9581
MOVENO SYN R+9659
MATB SYN R+9508
MATW SYN R+9309
MOBB SYN R+9510
MOBN SYN R+9511
NUMEP SYN R+9648
ISPEC SYN R+9692
LABEL
LIST8

CHESS PRINT TABLE ROUTINE

SUBROUTINE PRINT (CODE)

CONTROL WORD BITS ARE IN DECREMENT
1 PRINTS NUMBER, MOVER, MOVETO 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 OFFLINE.
20 PRINTS LOC, IBEG, IEND, NFIRST, KIND, IPIN OFF-LINE.
40 PRINTS MOVEP, MOVFR, ICAPT OFF-LINE.
100 PRINTS NUMB, ITCH, ICTDB, 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), NUMP(50),
1NTYPE(50), IBEG(33), IEND(32), MOVE(504), ICAPT(150),
2MOVEFR(150), MOVEP(150), IBEAR(1024), IBEAR(64,16),
3KIND(32), MSVN(16), IPDIR(32), IEYTD(16), IEYTS(64),
4M64M1(16), NMOV(6), IOPP(16)

DIMENSION JPAWN(8)
DIMENSION MSTO(32)
DIMENSION MAVAIL(100), ITCH(2), ICTDB(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, IOPP, IEYTS, IEYTD, JPAWN, M64M1, MSVN, NMOV, MST0, JPROM,
1IBEAR, JBEAR, KIND, IEND, IBEG, IOCC, LOC, NFIRST, MOVE, IENUS, MOVEP,
2MOVEFR, ICAPT, NUMP, 1NTYPE, ITCH, ICTDB, IPIN, NEP, MEP1, MEP2, LOGG, NLOG,
3NZZZ, NUMTES, MAVAIL, 12, 1Y, IX, IU, IT, ISPEC, IK, IQ, IPROM, IOPP, INTEGER,
4IDIR, ICHECK, IA, IAAP, JA, JB, JG, JDIR, JD, JE, JF, JIN, JJ, JROOK, JK, KD,
5K, L2L, M4, MAKEI, ICAPT, MCUL, MIN, MOVDIR, MOVNO, MOVER, MOVETO, MO, M,
6MVR, N1, N2, NEWSW, N, NSPEC, NUMEP, NPRINT, KIN, KVAL, KFORCE, KWORTH, MOBW,
7MOBB, MATW, MATB

EQUIVALENCE (IENUS, IBEG(33)), (NLOG, LOGG(101)), (NUMTES, NZZZ(120)),
1(IBEAR, 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), AM1(100), AM2(100)

EQUIVALENCE (M1, AM1), (M2, AM2)
EQUIVALENCE (I, A1)
CODEWD=CODE
IPRINT = IPRINT+1

NUMBER, MOVER, MOVETO

IF (CODEWD*000001000000 6969, 1000, 1001
1000 N=2
GO TO 5
1001 N = 100
5 CALL JUNPAK(MOVER+MSTU(MOVER) -1, M1 (1), M1 (2))
WRITE OUTPUT TAPE N*910, IPRINT, M1 (1), M1 (2)
910 FORMAT (21H1SET OF TABLES NUMBER,13,10H MOVE IS ,2A6)

IOCC

47 N = 2
GO TO 49
48 N = 100
49 CALL BOARD (N)

MAVAIL

IF (CODEWD*000002000000 6969, 47, 48
50 IF (CODEWD) 51, 6969, 52
51 N = 100
GO TO 54
52 N = 2
54 IF(K1)45,42,44
42 WRITE OUTPUT TAPE N* 70
70 FORMAT (10H STALEMATE )
GO TO 475
45 WRITE OUTPUT TAPE N* 73
73 FORMAT (10H CHECKMATE )
GO TO 475
44 WRITE OUTPUT TAPE N*960
960 FORMAT (7H MAVAIL )

82 DO 17 I=1,K1
17 CALL JUNPAK(MAVAL (1), M1 (1), M2 (1))
WRITE OUTPUT TAPE N*1391, (M1(1), M2(1), I=1,K1)
475 WRITE OUTPUT TAPE N* 139
139 FORMAT (1H4)
1391 FORMAT (1H0,20A6)
130 CONTINUE

2000 PRINTS PRINCIPAL VARIATION, ONLINE IF NEG.

IF (CODEWD*002000000000 6969, 224, 223
223 N=2
224 IF (CODEWD) 221,220,220
221 N=100
220 199=1
CALL JUNPAK (MOVES* M1(1), M1(31))
I=1
230 INT=XANDDF(XADDV(LISP(I+1)),127)
IF(INT) .215 ,215,225
225 INT=1+INT+1
I99=199+1
M1(I99)=XDECF(LISP(INT))+XSHIFTF(XTAFG(LISP(INT)),-18)
CALL JUNPAK(M1(I99),M1(I99),M1(I99+50)),I=2,199
I=XADD(LISP(INT))
GO TO 230
215 WRITE TAPE 6,199,M1
NMOVES=NMOVES+1
WRITE OUTPUT TAPE N,222,LISP(I+1),MULG,(M1(I),M1(I+50),I=2,199)
222 FORMAT(21H1PRINCIPAL VARIATION //1H VALUE=,17,8H EFFORT=,17/
1(1H0,20A6))
224 CONTINUE
C
C 4000 PRINTS MOVE TREE
B IF(CODEWD*004000000000) 6969,270,261
B261 AM1(I)=3
LEVEL=1
PRINT 260,(I,I=1,20)
260 FORMAT(1H1,31X,13THE MOVE TREE/6H hole LEVEL/20D,8H VALUE)
I=3
263 IF(LISP(I)) 262,270,264
264 I=I+1
GOTO 263
B262 AM1(LEVEL)=(AM1(LEVEL)*77777)+AL1
CALL WRITE(LISP(I),LEVEL)
I=XADD(LISP(I))
IF(I) 270,269,271
271 IF(LISP(I)) 268,270,265
265 I=I+2
LEVEL=LEVEL+1
M1(LEVEL)=XSHIFTF(I,-18)
GO TO 263
268 PRINT 274,LISP(I+1)
274 FORMAT(1H+,105X,110)
269 I=XDECF(M1(LEVEL))-1
IF (XADD(M1(LEVEL))-1) 262,262,267
267 LEVEL=LEVEL-1
IF(LEVEL) 270,270,269
270 CONTINUE
C
C EVALUATION PARAMETERS
B IF(CODEWD*000010000000) 6969,60,134
134 PRINT 133, MATW, MOBW, MATB, MOBB
133 FORMAT(1H2+8X,19H MATERIAL MOBILITY/6H WHITE/210/6H BLACK/2110)
IF (MCOL-1) 6959, 20, 21
B 20 AM2 = 606630316329
GO TO 62
B 21 AM2 = 602243214342
62 PRINT 22, K1, M2 (1), MOVENU, NSPEC, ICHECK, MLUG
22 FORMAT(19H NUMBER OF MOVES = 13/8H MCOL IS A6/L MMOVNO = 13/9H N
1SPEC = 14/9H ICHECK =14/7H MLUG =16)
60 CONTINUE
C
C PRINT THE OTHER TABLES
C LOC, IBEG, IEND, NFIRST, KIND, IPIN
B IF(CODEWD=000020000000) 6969,80,63
63 WRITE OUTPUT TAPE 2,2,(I=1,32),(LOC(I),I=1,32),(IBEG(I),I=1,32),
1(IE(1),I=1,32),(NFIRST(I),I=1,22),(KIND(I),I=1,32),(IPIN(I),I=1,32)
2 FORMAT (8H,0,PIECE 3213/8H LOC 3213/8H IBEG 3213/8H IEND
13215/8H NFIRST 3213/8H KIND 3213/8H IPIN 3213)
80 CONTINUE

C MOVEP, MOVEFR, ICAPT
B IF(CODEWD=000040000000) 6969,90,81
81 WRITE OUTPUT TAPE 2,8,(MOVEP(1),I=1,M0VENO)
8 FORMAT (6H MOVEP1916/(2016))
8 WRITE OUTPUT TAPE 2,7,(MOVEFR(1),I=1,M0VENO)
7 FORMAT (6H MOVEFR1916/(2016))
7 WRITE OUTPUT TAPE 2,6,(ICAPT(1),I=1,M0VENO)
6 FORMAT (6H ICAPT1916/(2016))
90 CONTINUE

C NUMB, NTYPE, ITCH, ITCMD, NEP, MEP1, MEP2
B IF(CODEWD=000010000000) 6969,162,95
95 WRITE OUTPUT TAPE 2,91,(NUMB(I),I=1,NSPEC)
91 FORMAT (6H NUMB1516/(2016))
9 WRITE OUTPUT TAPE 2,92,(NTYPE(I),I=1,NSPEC)
9 FORMAT (6H NTYPE1916/(2016))
9 WRITE OUTPUT TAPE 2,93,(ITCH(I),I=1,2),(ITCMD(I),I=1,2)
93 FORMAT (6H ITCH 213,8H ITCMD 213)

C SET UP NEP, MEP1, AND MEP2 FOR OUTPUT
DO 153 J=1,60
153 M1(J)=0
DO 150 I=1,10
150 IF (NEP(I)) 151,150,151
151 M1(I)=XMV3F(MEP1(I))
152 M1(I+20)=XMV2F(MEP1(I))
153 M1(I+40)=XMV1F(MEP1(I))
154 IF (MEP2(I)) 155,150,155
155 M1(I+10)=XMV3F(MEP2(I))
156 M1(I+30)=XMV2F(MEP2(I))
157 M1(I+50)=XMV1F(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 NEP1013,5H MEP11013,5H MEP21013/(142,913,18,913))
162 CONTINUE

C WRITE THE LOG
B IF(CODEWD=000020000000) 6969,170,164
164 PRINT 165, MLOG
165 FORMAT (17H THE LOG --- MLOG=,15,,
11=0
IF (MLOG=100) 160,160,161
161 REWIND 7
DO 166 I3=100,MLOG,100
11=13
READ TAPE 7,M1
DO 1640 I=1,100
1640 CALL JUNPAK (M1 (1), M1 (1), M2 (1))
166 PRINT 163, (M1 (1), M2 (1), I=1, 100)
163 FORMAT (1H0,2A9, A7, A6, A7, A6, A6, A7, A6, A6, A6, A6, A6, A6, A6, A6, A6, A6, 1A7, A6)
160 I2=MLOG-11
IF (I2) 170, 170, 167
167 DO 169 I=1,12
169 CALL JUNPAK (LOGG (1), M1(1), M2 (1))
PRINT 163, (M1 (1), M2 (1), I=1,12)
170 CONTINUE
C
C 1BEAR
B
168 PRINT 10, ((I, I=1, 16), J=1, 2), (I, (1BEAR (I, J), I=1, 16),
1NUMBER (I+32), (1BEAR (I+32, J), J=1, 16) I=1, 32)
10 FORMAT (6H1BEAR/16,1D13, 1D16,1D1513/(1713,113,1613)) -
200 CONTINUE
C
C MOVE
B
210 PRINT 94
94 FORMAT (12HOMOVE TABLE,)
DO 11 I=1,32
IF (LOG(I)) 12, 11, 12
12 M=1BEG(I)
N=1END(I)
DO 13 J=M,N
K=J-M+1
IF (MOVE(J)) 110, 111, 110
111 M(K)=0
M2(K)=0
GO TO 13
110 M1(K)=XSIGNF (XMOV1F (MOVE(J)), MOVE(J))
M2 (K)=XMOV2F (MOVE(J))
13 CONTINUE
K3=XMINOF (28, N-M+1)
WRITE OUTPUT TAPE 2,15,1, (M1(L), L=1,K3)
15 FORMAT (23H MOVES OF PIECE NUMBER, I2/(1H0,2814))
WRITE OUTPUT TAPE 2,16, (M2(L), L=1,L3)
16 FORMAT (1H0,2814)
IF(N-M+1-28) 11, 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 10000 PRINTS HISTORY
B
510 N=2
IF (CODEWD) 311, 312, 312
311 N=100
312 IF(NMOVES) 350, 320, 319
313 REWIND 6
314 WRITE OUTPUT TAPE N, 322
315 FORMAT (31H1LEVEL, OPPONENT, MACHINE, 10X, 19HPRINCIPAL VARIATION)
316 DO 320 198 = 1, NMOVES
317 READ TAPE 6, 199, M1
318 WRITE OUTPUT TAPE N, 321, 198, (M1(I), M1(I+50), I = 1, 199)
319 FORMAT (1IH0, 15, 2(2X, 2A6), 2X, 14A6 / (16X, 14A6))
320 CONTINUE
321 CONTINUE
322 RETURN
323 PRINT 320
324 FORMAT (47HLOSE, LOGIC OR PROGRAM MAKES THIS IMPOSSIBLE.)
325 GO TO 600
326 END
END
* LABEL
* FAP
 COUNT 55
* FTNBL OINARY LOADER
* LOADS COLUMN ABSOLUTE FROM TAPE A2.
 REM 0056 SYM. CARDS DIST. 535 RCV. 12-03-58CORR. OF DIST.52711
* WD BTOU2, BINARY TAPE UPPER LOADER
*  
 ENTRY FTNBL
 L TAPENO A2
 FTNBL TEFL ++1
 SXA TR2,1
 SXA TR2+1,2
 AXT AXT 1,2
 CLEAR CLM
 RTBL
 RCHL IOCT
 LCHL TXH
 TEFL 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
 TRI 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
 TNE NG
 TRA AXT
 NG TIX TR3,2,2
 BSRL
<table>
<thead>
<tr>
<th>Command</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXI</td>
<td>CLEAR^2,1</td>
</tr>
<tr>
<td>READ</td>
<td>PZE</td>
</tr>
<tr>
<td>IOCT</td>
<td>IOCT</td>
</tr>
<tr>
<td>EOF</td>
<td>HTR</td>
</tr>
<tr>
<td>CW</td>
<td>PZE</td>
</tr>
<tr>
<td>TR2</td>
<td>AXT</td>
</tr>
<tr>
<td>AXT</td>
<td>**,1</td>
</tr>
<tr>
<td>AXT</td>
<td>**,2</td>
</tr>
<tr>
<td>TRA</td>
<td>1^4</td>
</tr>
<tr>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>
* LABEL
* FAP
COUNT 270
*MISPX BUGGERED VERSION OF MISPH- (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
(PUCT) EQU 89
(ELCT) EQU 90
(LNCT) EQU 97
PUNSW EQU 4
PRNSW EQU 5
REM
(SPHM) CAL =02000000
(SPHM)=WRITE OUTPUT TAPE 2
(SPHM) CAL =01000000
STE MONSW
SET SWITCH FOR MONITOR CONTROL
CAS =100017
CHECK FOR TAPE 100
TRA ++2
TRA BOTH
BOTH ON AND OFF LINE
STZ ONSW
NOT ON LINE SWITCH
PROC
SLW UNIT
SAVE LOGICAL TAPE NO.
(STH) LDQ ++2
LOAD MQ WITH OUTPUT SWITCH + RETURN ADDRESS
TRAS $(10H)
GO TO $(10H)
TRAS STH
RES
BOTH
CAL =2617
SET ON LINE SWITCH
TRAS PROC
REM
(SCHM) STL MONSW.
SET ON LINE SWITCH
REM
(SCHM) STL PUNSW.
IS ON LINE PUNCH SWITCH DOWN
TRA (STH3)
NO, WRITE LOGICAL TAPE 3 (PUNCH TAPE)
(SCH) CLA MZE2
YES, SET UP TO PUNCH ON LINE ONLY
LDQ ++2
TRAS $(10H)
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
TRAS $(10H)
OUTPUT SWITCH AND RETURN ADDRESS
TRA STH3
REM
(STHD) LDQ ++2
LOAD MQ WITH OUTPUT SWITCH + RETURN ADDRESS
TRAS $(10H)
GO TO $(10H)
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

TERMINATE 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

MAXIMUM LINES OF DEBUG OUTPUT
COUNTS DEBUG LINES

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

YES, SO UPDATE TOTAL LINE COUNT

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

TERMINATE THIS JOB
REM TES $ ( WER ) \times 4
LXA STHX \times 4
CAL 1 \times 4
ARS 18
ACL 1 \times 4
STA MOVE.
STD STHC
PDX 0 \times 4
TXI \# +1,4, OUTPUT
SXA MOVE \times 1,4
PDX 0 \times 4

MOVE.
CAL \# 3 \times 4
SLW \# 3 \times 4
TIX MOVE \times 4, 1
CAL TES
SLW* $ ( TES )
AXC STHC \times 4
PXA 0 \times 4
STA* $ ( WTC )
XEC* $ ( WRS )
XEC* $ ( RCH )

STHX
AXT \# 3 \times 4
NZT MONSW。
TRA 2 \times 4
CLA UNIT.
SUB = 02000000
TNZ 2 \times 4
ZET ONSW
TRA \# 3
XSW PRNSW。
TRA 2 \times 4
CAL ( PUCT )
ADD = 01000000
STD ( PUCT )
TRA SPH
REM

SCH
NZT MONSW。
TRA \# 4
CAL ( PUCT )
ADD = 01000000
STD ( PUCT )
SXA NPIR1,1
LDQ WPUA。
CAL NPNO P
AXT 12 \times 1
TRA PRPUN。
REM

SPH
SXA NPIR1,1
LDQ* 1 \times 4
PXD
LGL 6
PAX 0 \times 1
CAL = 060

CHECK ANY PREVIOUS WRITE
RESTORE CALL INDEX
CALL = PZE FIRSTIN
WORD COUNT INTO OUTPUT COMMAND
AND IR4
RESTORE WORD COUNT
MOVE DATA TO OUTPUT BUFFER
SET UP ERROR CHECKING
ADDRESS OF I/O COMMAND
SAVE IN CASE OF ERROR
SELECT OUTPUT TAPE
WRITE OUT THIS RECORD
RESTORE RETURN INDEX
IS THIS A MONITOR JOB
IS THIS THE MONITOR STACKED OUTPUT TAPE
NO, RETURN TO ( IOH )
CHECK TO SEE IF TAPE WAS 100
YES, PRINT ON LINE
IS THE ON LINE PRINT SWITCH ON
NO, RETURN TO ( IOH )
YES, PRINT THIS ON LINE
UPDATE ONLINE PRINT COUNT
GO TO ON LINE PRINT ROUTINE
ON LINE PUNCH ROUTINE
SKIP UPDATE OF ( PUCT ) IF NOT IN MONITOR
OTHERWISE UPDATE ( PUCT )
SAVE IR1
PICK UP ON LINE PUNCH SELECT
PICK UP NOP TO AVOID SPACE CONTROL
PICK UP MAX. WORD COUNT FOR ON LINE PUNCH
GO TO BCD TO CARD IMAGE CONVERTER
ON LINE PRINT ROUTINE
PICK UP FIRST BCD WORD
GET FIRST CHARACTER OF LINE
SAVE IT IN IR1
REPLACE WITH A BLANK
LGR 6
STQ* 1,4
PXA 0,1
AXT ESPTB-BOSTB.1
CAS ESPTB.1
TRA **+2
TRA SPFND
TIX **--3,1,2
CAL NPNS+1
TRA SPFND+1
SPFND CAL ESPTB.1\,1
LDQ WPRA.
AXT 20,1

PRPUN. SLW NPSPR
STQ WRSA.
SXD TSTCT,1
CAL 1,4
PDX 0,1
TSTCT TAL **+2,1,2
LXD TSTCT,1
PX A 0,1
STA NPSV4
ACL 1,4
STA NPRC3
SXA NPIR2*2
SXA NPIR4*4
LAA NPSV4*4
TXL 1PASS,4,12
STL 2PSW
REM

1PASS AXT 24,1
STZ PBUFF+24,1
TIX **--1,1,1
AXT 1,2
NPRC1 GAI COLIND
NPRC2 SLW PRCOL
SXA NPSV4*4
NP RC3 LDO **+4
AXT 6,4
NPRC4 PXD
LGL 6
ALS 1
PAX 0,1
CAL PRCOL
ARS 6,4
TXL PDIGIT,1,24
TXL PNZONE,1,95
TXL NPRC6,1,96
REM
PNZONE TXH PNMIN,1,62
ORS PBUFF+23,2
TIX PDIGIT,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 = PZE FIRST,\,*N
WORD COUNT TO IR1
SKIP IF WORD COUNT OK
WORD COUNT TOO LARGE, SET TO MAX.
SAVE WORD COUNT
FIRST+\,*N
SAVE IR1
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 NEXT BCD WORD
SET CHARACTER COUNT
GET A CHARACTER
DOUBLE IT
INTO IR1
POSITION COLUMN MARKER
SKIP IF DIGIT ONLY
SKIP IF BLANK
SKIP IF 11 OR 0 ZONE
UP IN THE 12 ZONE
REMOVE 12 PUNCH
SKIP IF + ONLY (NO DIGIT)
PNMIN  TXH  PNZER,1,94  SKIP IF 0 ZONE
ORS   PBUFF+21,2  OR IN THE 11 ZONE
TIX   PDIGIT,1,64  REMOVE 11 ZONE
TRA   NPRC5  SKIP IN - ONLY (NO DIGIT)
PNZER ORS PBUFF+19,2  OR IN THE 0 ZONE
TXI   PDIGIT,1,96  REMOVE 0 ZONE
REM
PDIGIT TXL PDIGIT,1,18  SKIP IF NORMAL DIGIT
ORS   PBUFF+3,2  HERE FOR 8-3, 0-4, OR IN THE 8 PUNCH
TXI   ++1,1,1-16  REMOVE THE 8 PUNCH
PNDIG TXL PBUFF+19,3  OR DIGIT TO CARD IMAGE
NPRC5 TXI NPRC4+4,1  COUNTS CHARACTERS
ARS   1  SET COLUMN MARKER FOR NEXT WORD
NPSV4 AXT **,4  RESTORE BCD WORD COUNT
TNX   PNOW,4,1  SKIP TO END IF DONE
TZE   PNTST  SKIP IF COLUMN MARKER MOVES OUT
TRA   NPRC2
PNTST TXL PNOW,2,0  SKIP TO END WHEN CARD IMAGE COMPLETE
AXT   0,2  OTHERWISE SET UP FOR RIGHT HALF
TRA   NPRC1
REM
PNOW  TCOA *  WAIT UNTIL LAST LINE OR CARD IS OUT
AXT   24,1
CAL   PBUFF+24,1
SLW   PBUFF1+24,1
TIX   **-2,1,1
WRSA  WRS **  SELECT ON LINE I/O UNIT
RCHA  NPIOC  WRITE THIS LINE OR CARD
NPSPR PSE **  SPACE CONTROL IF ANY
NVTN  2PSWT  IS A 2ND PASS NEEDED
TRAP  NPIR1  NO, GO TO EXIT
STZ   2PSWT  YES, RESET SWITCH
CAL   PSPR9  SET SPACE CONTROL FOR 2ND HALF
SLW   NPSPR  **
TRAP  1PASS  GO THROUGH THE WHOLE MESS AGAIN
REM
NPIR1 AXT **,1
NPIR2 AXT **,2
NPIR4 AXT **,4
TRAP  2,4  RETURN TO CALLER
REM
BSPTB BCI 1,000000
SPRA 4
BCI 1,000001
SPRA 1
BCI 1,000002
SPRA 2
BCI 1,00000+
SPRA 5
ESPTB SYN *
REM
ONSSW PZE
MONSSW, PZE
2PSWT PZE
UNIT, PZE
SIND, PZE
PRCOL, PZE
COLIND, MZE
MZE2, MZE, 2
MZE3, MZE, 3
NPNOP, NOP
PSPR9, SPRA, 9
WPRA, WPRA
WPUA, WPUA
NPIOC, IOC0, PBUF1, 24
STHC, IOST, OUTPUT, **
OUTPUT, BSS, 22
PBUF1, BSS, 24
REM
COMMON, -176
REC, COMMON, 76
PBUF, COMMON, 1
END
* LABEL
* LIST8

SUBROUTINE BOARD(ITAPE)
C PRINTS OUT CHESS BOARD IN READABLE FORMAT.
DIMENSION FOO(5000), PIECES(43), TAB1(8), TAB2(8), KIND(32),
10CC(64)
COMMON FOO
EQUIVALENCE (FOO(2938), PIECES), (FOO(1527), IOCC), (FOO(1463),
1KIND)
WRITE OUTPUT TAPE ITAPE,6
6 FORMAT (1H,J18X,5HBLACK/1H,J18X,5H----)
DO 1 J = 1, 8
L = XGETF(J,J+5/1,10CC)
1 IF (XRANGEF(L,7,22)) 7,9
IF (KIND(L) = 1) GOTO 7
8 L = KIND(L) + 5*ALGETF(L) + 31
B7 TAB1(J) = PIECES(L+1)
B10 TAB2(J) = SHIFF(PIECES(L+1),22)
WRITE OUTPUT TAPE ITAPE,3
3 FORMAT (42H**********************************************************************)
1 WRITE OUTPUT TAPE ITAPE,4, TAB1, TAB2
4 FORMAT (1H,B(2H*,A3),1H*,1H,B(2H*,A3),1H*)
WRITE OUTPUT TAPE ITAPE,3
WRITE OUTPUT TAPE ITAPE,5
5 FORMAT (1H,J18X,5HWHITE)
RETURN
END
CARDS ROW
FAP
COUNT 20
MISTOP
FUL
ORG -11
IOCD C,11
TCA 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
* LABEL
  * FAP
  * COUNT 35
  * 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 =5
ORA =H1 00
SLW FMT
CALL JUNPAK,MOVE,A,B
TSX $(SPH),4
PZE FMT,,-1
LDQ A
STR
LDQ B
STR
TSX $(FIL),4
LXD 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

<table>
<thead>
<tr>
<th>KEYS</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENK</td>
<td>1,4</td>
</tr>
<tr>
<td>SLG*</td>
<td>35+10+2</td>
</tr>
<tr>
<td>LLS</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>2,4</td>
</tr>
<tr>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>
* 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 $TIMER,4
TXH AC1
TXH TIMEOUT
DONE AXT **4
TRA 1,4
TIMOUT CAL $100B17
TSX $STH,4
TSX TIMIT
TSX $FIL,4
CALL CLOCK,2
CALL PRINT,4
LAC 6,4
SXA PMRST-1,4
CLA $F2PM
STA 6
TRA $RSTRIN
A TTR **1
LTM SXA XR4,4
AXT FMT,4
C SXA B+4
STQ MQ
SLW AC1
ARS 2
STQ AC2
CAL $100B17
TSX $STH,4
B   PZE       **,-1
TSX   $ (F1L),4
CALL   CLOCK, D2
CALL   PRINT, N
XR4  AXT       **,4
LDQ     MQ
CLA     AC2
ALS     2
ORA     AC1
TRA*    $ (F2PM)
RECOUP  SXA   XR4,4
        LAC   XR4,4
        SXA   PMRST-1,4
        AXT   FMT1,4
        TRA   C
LDUMP  SXA   XR4,4
        LAC   XR4,4
        SXA   PMRST-1,4
        LXD   LDMPF,4
        LDMF   TXI  C*, FMT2
N    OCT    77774000000
D2   DEC    2B17
AC1
AC2
MQ
TIMFMT BCI   2, (8H1TIMEOUT)
FMT BCI    6, (28H1 PROGRAM MANUALLY RESTARTED*)
FMT1 BCI   4, (16H1 RECOUP REACHED*)
FMT2 BCI   4, (15H1LDUMP REACHED*)
COMMON  12561
R COMMON  1
NLOG EQU  R+12428
MLOG EQU  R+9443
IPRINT EQU  R+3373
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 =1B17
PDX +1
LOOP CLA IEXTD+1,2
ADD IEXTS,1
ANA =020177000000
PDX +1
TXH NOSQ,1,63
ZET IOCC,1
TRA FOUND
TXL LOOP+2,8
NOSQ CLS =1B17
FOUND ADD =1B17
XR1 AXT **,1
AXT **,2
TRA 3,4
* STORAGE ALLOCATION
COMMON 12561
R COMMON 1
IEXTS SYN R+11201
IEXTD SYN R+11197
IOCC SYN R+11035
END

FIND NEXT SQUARE
DO XMOVF
OFF BOARD YET
SQUARE OCCUPIED
YES
LOOK AGAIN IF NOT KNIGHT
PICK UP -1 SO RESULT ZERO
MAKE -0 OR ACTUAL SQUARE
RESTORE
RETURN
* LABEL
* FAP
*XTIME WITH INTERVAL TIMER
COUNT 15
ENTRY XTIME
ENTRY XLAPSE
XTIME TRA $RSCLCK
XLAPSE SXA XIT+4
CALL STOPCL+1
PXD
LDQ 1
DVP =360B17
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 =1617
ANA =7617
ADD =1617
TRA 1,4
JUNPAK SXA XR4,4
CLA* 1,4
STO T1
TZE ZERO
TMI ZERO
TSX $XMV3,4
PDX ,4
ANA =1817
STO COLOR
TXL B,4,6
TXH B,4,22
CLA K1ND+1,4
SUB =1817
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 $XMV1,4
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
XR4

SPEC
ZILCH
PIECES

TKUP
COLOR

ABS
PZE

TA
SLM

CAL
CAL

SLM *

COLOR
PZE

ANS2
PZE

ASS

SLM

CAL
CAL

AXT
ORA

CAL
SLM

XR4

HOC (00)

SPEC

FILE12

** 3 SETU BLACK WHITE

R+9622

FILE12
LABEL
FAP
COUNT 152
CHESS ROUTINES IN FAP, RE-ASSEMBLED FOR 709, A. KOTOK
ENTRY XLBIT
ENTRY XMOCV
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
STO 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 =7B17
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  LDQ  A2
    TRA  XLBIT+1
A2  ANA  =01777000000
XMV1 ANA  =63B17
    TRA  XMOM+1
XMV2 ARS  6
    ANA  =15B17
    TRA  XMOM+1
XMV3 ARS  10
    ANA  =31B17
    TRA  XMOM+1
XADD ANA  =077777
    AL5  18
    TRA  1,4
M2  127+3*1024
    REM  XBANDF(L+M) GIVES L AND M
XVND STQ  T1  M
    ANA  T1  L+M
    TRA  1,4  EXIT
    REM  XBORF(L,M) GIVES L-INCLUSIVE-OR-M
XBOR STQ  T1  M
    ORA  T1  L+M
    TRA  1,4  EXIT
    REM  XBEOR(L,M) GIVES L-EXCLUSIVE-OR-M
XBEOR STQ  T1  M
    ERA  T1
    TRA  1,4
XBNOT LDQ  A3
    TRA  XLBIT+1
A3  ERA  =07777700000
T1  TEMPORARY STORAGE
    REM  STOF AND XSTOF
    REM  STORES X IN A(j) BY CHANGING THE INSTRUCTIONS IN THE PROG
XSTO BSS  0
STO STQ  T1
    CLA  -1,4
    TPL  LDQ
    REM  PREVIOUS INSTRUCTION WAS AN SXD. MOVE IT BACK ONE INTR
    LDQ  -1,4
    CAL  -2,4
    STQ  -2,4
    REM  CHANGE LOC(0,4) TO A STO A+1,4
LDQ ANA  =077777
    ADD  =1
    ANA  =077777
    ORA  STOR
    SLW  0,4
    REM  CHANGE PPREV INSTRUCTION TO AN LXD -3,4 WHERE J STORED
    CAL  LXD
    SLW  -1,4
    CLA  T1
    TRA  -1,4
STOR STQ  0,4
LXD  LXD  A,4
A  SYN  -3
XAND  STQ  T1
SSP
ADM  T1
TRA  1.4
XOR  TZE  1.4
XCA
TRA  1.4
XLESS  STQ  T1
SUB  T1
TZE  1.4
CHS
LRS  0
PXD
LGL  1
ALS  18
TRA  1.4
XNOT  TZE  NOSAT
PXD
TRA  1.4
XONE  SSP
TZE  1.4
TRA  NOSAT
XRANGE  TNZ  #+2
SSP
STQ  T1
CAS  T1
NOP
TRA  #+3
NOSAT  CLA  #1817
TRA  1.4
TNZ  #+2
SSM
CAS  -3
TRA  NOSAT
NOP
PXD
TRA  1.4
REM  GET AND XGET
REM  GET ALLOWS USE OF ILLEGAL SUBSCRIPTS IN FORTRAN
XGET  BSS  0
GET  STQ  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>Layer</th>
<th>T1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLA</td>
<td>-1.4</td>
</tr>
<tr>
<td>TRA</td>
<td></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>
* LABEL
* LIST8
C JAN 14, 1961
FUNCTION ISCHEK(MV)
C
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.
C
DIMENSION AND EQUIVALENCE STATEMENTS
COMMON AA
DIMENSION AA(4500)
DIMENSION MAVAIL(100), KIND(32), LOC(32), I0CC(64), NEP(10), MEP1(10),
1MEP2(10), IBEG(33), IEND(32), LEGAL(3, 3)
DIMENSION IEXTS(64), IEXTD(13)
EQUIVALENCE (AA(2765), MAVAIL(1)), (AA(2892), K1(1)),
1(AA(1463), KIND(1)), (AA(1591), LOC(1)), (AA(1527), I0CC(1)),
2(AA(2900), MCOL(1)), (AA(2914), NUMEP(1)), (AA(2745), MEP1(1)),
3(AA(2755), MEP2(1)), (AA(1495), IBEG(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)
C
MAIN PROGRAM
1 M=MV
MVER=XMV3F(M)
MVDIR=XMV2F(M)
MVTO=XMV1F(M)
MVFR=LOC(MVER)
MKIND=KIND(MVER)
KLOC=XGETF(3-MCOL, LOC)
I0CC(MVFR)=0
I8=I0CC(MVTO)
I0CC(MVTO)=MVER
C IS THIS AN EN PASSANT MOVE
9 IF (NEP(NUMEP)-MOVENO) 8, 9, 8
8 IF (XORF(MEP1(NUMEP)+M, MEP2(NUMEP)+M)) 8, 5, 8
C IS THIS A CASTLING MOVE
C MOVE INVOLVES CASTLING. FIND ROOK LOCATION
3 IF (MVDIR-2) 4, 5, 6
4 I1=XMOVF(IEXTS(MVTO)+IEXTD(1))
GO TO 7
6 I1=XMOVF(IEXTS(MVTO)+IEXTD(3)+IEXTD(1))
7 I0CC=I0CC(I1)
I2=XMOVFIEXTS(MVFR)+IEXTD(MVDIR))
C I1=OLD ROOK SQUARE; I2=NEW ROOK SQUARE
C MOVE PIECES
I0CC(I2)=I0CC
I0CC(I1)=0
C IS THE KING IN CHECK
ISCHEK=MABLE(I2, KLOC)
C RESTORE POSITION OF KING AND ROOK
12 I0CC(I1)=I0CC


IOCC(12)=0
IOCC(MVFR)=MVER
IOCC(MVTO)=0
RETURN
C
   IS A PAWN PROMOTING
2
   IPROM=0
IF (XANDF(MVFR\+1, XTRANKF(MVTO, MVER)-8)) 15, 16, 15
16
   KIND(MVER)=XADDF(M);
   IPROM=1
C
   SEE IF MOVER IS CHECKING
15
   ISCHEK=MABLE(MVTO, KLOC);
IF (ISCHEK) 5, 17, 21
C
   IS THERE A DISCOVERED CHECK
17
   I4=LOOK(KLOC, NURIEK(KLOC, MVFR));
IF (I4) 5, 19, 24
24
   I5=IOCC(I4)
IF (XLBIF(I5-MVER)) 5, 25, 19
25
IF (XABS(KIND(I5)-3)-2) 20, 19, 20
20
   ISCHEK=MABLE(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)=MVER
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),
1MEP2(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),MEP1(1)),(AA(2914),NUMEP(1)),(AA(2745),MEP1(1)),
3(AA(2755),MEP2(1)),(AA(1),IBEG(1)),(AA(1495),IEND(1)),
4(AA(2923),LEGAL(1))
EQUIVALENCE (AA(2765),NEP(1))
EQUIVALENCE (AA(1285),IOPP)
C
MAIN PROGRAM
1 M1=MSQ1
2 M2=MSQ2
3 MP=IOCC(M1)
4 K=KIND(MP)
5 IDIR=NORIEN(M1,M2)
6 CHECK WHETHER PIECES ARE IN LINE
7 IF (IDIR) 2,2,3
8 IF (LOOK(M2,IOPP(IDIR))-M1) 2,4,2
C PIECES ARE IN LINE
9 IF (K-1) 10,9,10
C CHECK PAWN DIRECTIONS
10 I1=IDIR+IDIR-13
11 I2=XLABTF(MP)
12 IF (XMINOF(XABSF(I1+2)-12,XABSF(I1-2)-1+12)) 2,11,2
C IS THIS A LEGAL MOVE DIRECTION FOR THE GIVEN PIECE
13 I1=XMINOF((IDIR+3)/4,3)
14 MABLE=LEGAL(K-1,I1)
15 IF (MABLE) 7,7,6
16 IF (K-5) 7,8,7
C PAWNS AND KINGS CAN ONLY MOVE ONE SQUARE
17 MABLE=1
18 I2=XABSF(M1-M2)
19 IF (XMINOF(I2-1,XABSF(I2-8)-1)) 7,7,2
20 MABLE=0
21 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
TZE 3 4
STO T1
ANA = 7B17
TNZ NOVT
CLT T1
TMI ++3
CLA = 2B17
TRA 3 4
CLA = 4B17
TRA 3 4
NOVT CLA* 1 4
SUB = 1B17
STO T1
ANA = 7B17
STO VF
CLA T1
ANA = 56B17
ARS 3
STO HF
CLA* 2 4
SUB = 1B17
STO T1
ANA = 7B17
STO VT
SUB VF
STO VD
CLA T1
ANA = 56B17
ARS 3
STO HT
SUB HF
STO HD
TNZ NOHOR
CLA VD
TMI ++3
CLA = 1B17
TRA 3 4
CLA = 3B17
TRA 3 4
NOHOR CLA VD
SSP
SBM HD
TNZ NODIG
PXD
LDQ VD

I= DIRECTION VERTICAL
EXIT IF FROM=TO

IF NOT VERTICAL, TRANSFER

DIRECTION 4

FILE OF FIRST SQUARE

FILE OF 2ND SQUARE

VERTICAL DIFFERENCE

RANK OF 1ST SQUARE

RANK OF 2ND SQUARE

HORIZONTAL DIFFERENCE

DIRECTION NOT HORIZONTAL

DIRECTION 1

DIRECTION 3

NOT DIAGONAL

FIND WHICH DIAGONAL DIRECTION
LGL 1
LDQ HD
LGL 1
SXA X+4
PAX 0+4
CLA DIAGD+4
TRA X4
NODIG CLA HD
SSP ADM VD
SUB =B17
TZE NITE
PXD
TRA 3+4
NITE LDQ VD
LGL 1
LDQ HD
LGL 20
ADM VD
SXA X+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
* LABEL
* FAP
COUNT 248
* CHESS TABLES FOR COMMON STORAGE
ABS
A EQU 1024
REM SYMBOL TABLE FOR ARRAYS
R EQU -1
IPDIR SYN 31284+R
IOPP SYN 31278+R
IEXTS SYN 31262+R
IEXTD SYN 31196+R
IENUS SYN 32530+R
IEND SYN 31068+R
IBEG SYN 32562+R
JPAWN SYN 31182+R
JPROM SYN 31104+R
KIND SYN 31100+R
M64M1 SYN 31174+R
MSTO SYN 31136+R
MSVN SYN 31158+R
NM0V SYN 31142+R
KVAL SYN 29646+R
FILES EQU 29519
KCNVAL EQU 29533
LEGAL EQU 29639
LOCBEG EQU 29581
* LOCBEG AND LEGAL TABLES FOR CHESS. JAN. 31, 1961
ORG LOCBEG-21
LOCBEG TABLE GIVES INITIAL LOCATIONS.
DEC 56B17,16b17,5Bl7,1Bl7,54B17,14B17,53B17,13B17
DEC 52B17,16317,51B17,11B17,50B17,10B17,49B17,9B17
DEC 64B17,8B17,57B17,18B17,61B17,5B17
ORG LEGAL-14
LEGAL GIVES LEGAL MOVE DIR. TO TABLE.
DEC 0,0,0,1B17,0,1B17,1B17,0,1B17,1B17,0,0,1B17
* CHESS TABLES * FILES, LCENS?, KCNVAL
ORG KCNVAL-15
DEC 8B17,8B17,4B17,4B17,4B17,4B17,8B17,4B17
DEC 2B17,4B17,4B17,2B17,1B17,1B17,1B17,1B17
ORG FILES-7
BCI 84K0000KN000KB0000 K0000 W0000Q0000Q0000QR0000
* KPAWN TABLE FOR IDVLOP
REM * MSVN TABLE
ORG MSVN-15
Cl BSS 0
DUP 1,16
0,0,7*16+7*C1-7**
REM IBEG TABLE
ORG IBEG-32
0,0,452
IBEG(33) IS THE SAME AS IENUS
0,0,225+112+56+4 32
0,0,225+112+4 31
0,0,225+68 30
0,0,285 29
0,0,257 28
<table>
<thead>
<tr>
<th>REM</th>
<th>IEND TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG</td>
<td>IEND-31</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0,0,492</td>
<td>32</td>
</tr>
<tr>
<td>0,0,225+171</td>
<td>31</td>
</tr>
<tr>
<td>0,0,225+115</td>
<td>30</td>
</tr>
<tr>
<td>0,0,225+87</td>
<td>29</td>
</tr>
<tr>
<td>0,0,225+59</td>
<td>28</td>
</tr>
<tr>
<td>0,0,225+31</td>
<td>27</td>
</tr>
<tr>
<td>0,0,228</td>
<td>26</td>
</tr>
<tr>
<td>0,0,220</td>
<td>25</td>
</tr>
<tr>
<td>0,0,212</td>
<td>24</td>
</tr>
<tr>
<td>0,0,204</td>
<td>23</td>
</tr>
<tr>
<td>0,0,196</td>
<td>22</td>
</tr>
<tr>
<td>0,0,192</td>
<td>21</td>
</tr>
<tr>
<td>0,0,188</td>
<td>20</td>
</tr>
<tr>
<td>0,0,184</td>
<td>19</td>
</tr>
<tr>
<td>0,0,180</td>
<td>18</td>
</tr>
<tr>
<td>0,0,176</td>
<td>17</td>
</tr>
<tr>
<td>0,0,172</td>
<td>16</td>
</tr>
<tr>
<td>0,0,168</td>
<td>15</td>
</tr>
<tr>
<td>0,0,164</td>
<td>14</td>
</tr>
<tr>
<td>0,0,160</td>
<td>13</td>
</tr>
<tr>
<td>0,0,156</td>
<td>12</td>
</tr>
<tr>
<td>0,0,152</td>
<td>11</td>
</tr>
<tr>
<td>0,0,148</td>
<td>10</td>
</tr>
<tr>
<td>0,0,144</td>
<td>9</td>
</tr>
<tr>
<td>0,0,140</td>
<td>8</td>
</tr>
<tr>
<td>0,0,229</td>
<td>27</td>
</tr>
<tr>
<td>0,0,193+28</td>
<td>26</td>
</tr>
<tr>
<td>0,0,213</td>
<td>25</td>
</tr>
<tr>
<td>0,0,205</td>
<td>24</td>
</tr>
<tr>
<td>0,0,197</td>
<td>23</td>
</tr>
<tr>
<td>0,0,193</td>
<td>22</td>
</tr>
<tr>
<td>0,0,189</td>
<td>21</td>
</tr>
<tr>
<td>0,0,185</td>
<td>20</td>
</tr>
<tr>
<td>0,0,181</td>
<td>19</td>
</tr>
<tr>
<td>0,0,177</td>
<td>18</td>
</tr>
<tr>
<td>0,0,173</td>
<td>17</td>
</tr>
<tr>
<td>0,0,169</td>
<td>16</td>
</tr>
<tr>
<td>0,0,165</td>
<td>15</td>
</tr>
<tr>
<td>0,0,161</td>
<td>14</td>
</tr>
<tr>
<td>0,0,157</td>
<td>13</td>
</tr>
<tr>
<td>0,0,153</td>
<td>12</td>
</tr>
<tr>
<td>0,0,149</td>
<td>11</td>
</tr>
<tr>
<td>0,0,145</td>
<td>10</td>
</tr>
<tr>
<td>0,0,141</td>
<td>9</td>
</tr>
<tr>
<td>0,0,137</td>
<td>8</td>
</tr>
<tr>
<td>0,0,133</td>
<td>7</td>
</tr>
<tr>
<td>0,0,105</td>
<td>6</td>
</tr>
<tr>
<td>0,0,77</td>
<td>5</td>
</tr>
<tr>
<td>0,0,49</td>
<td>4</td>
</tr>
<tr>
<td>0,0,21</td>
<td>3</td>
</tr>
<tr>
<td>0,0,11</td>
<td>2</td>
</tr>
<tr>
<td>0,0,1</td>
<td>1</td>
</tr>
</tbody>
</table>
0*0,136 7
0*0,132 6
0*0,104 5
0*0,76 4
0*0,48 3
0*0,20 2
0*0,10 1

REM KIND TABLE M179
ORG KIND-31
0*0,6 QUEEN
0*0,6
DUP 1*4 BISHOP
DUP 1*4 K N I T H
DUP 1*16 P A W N
DUP 1*4 R U O K
0*0,5 R U O K
0*0,5 KING

REM IEXTD TABLE M179
ORG IEXTD-15
0*0,2*A+2+15*8 11
0*0,A+1+14*8 10
0*0,15*A+7+13*8 9
0*0,14*A+6+14*8 8
0*0,14*A+6+0 7
0*0,13*A+7+8 6
0*0,A+1+2*8 5
0*0,2*A+2+8 4
0*0,A+1+15*8 3
0*0,15*A+7+14*8 2
0*0,15*A+7 1
0*0,A+8+1 1
0*0,15*8 1
0*0,15*A+7+15*8 1
0*0,8 1
0*0,A+1 1

REM M64M1 TABLE M179
ORG M64M1-15
0*0,15*64-1 11
0*0,14*64-1 10
0*0,13*64-1 9
0*0,12*64-1 8
0*0,11*64-1 7
0*0,10*64-1 6
0*0,9*64-1 5
0*0,8*64-1 4
0*0,7*64-1 3
0*0,6*64-1 2
0*0,5*64-1 1
0*0,4*64-1 1
0*0,3*64-1 1
0,0,2*64-1
0,0,1*64-1
MZE
0,0,1
REM IEXTS TABLE
ORG IEXTS-63
B
BSS 0
K
SYN 63+B
DUP 8,8
0,0,K-**7*A
0,0,K-**6*A
0,0,K-**5*A
0,0,K-**4*A
0,0,K-**3*A
0,0,K-**2*A
0,0,K-**1*A
0,0,K-
REM JPAWN TABLE
ORG JPAWN-7
0,0,2 8
0,0,1 7
0,0,1 6
0,0,2 5
0,0,3 4
0 3
0,0,3 2
0
REM IPDIR TABLE
ORG IPDIR-5
0,0,2
0,0,5
0,0,6
0,0,4
0,0,8
0,0,7
REM NMOV TABLE
REM
ORG NMOV-5
0,0,56
0,0,8
0,0,28
0,0,8
0,0,28
0,0,4
REM IOPP TABLE
REM
ORG IOPP-15
0,0,12
0,0,11
0,0,10
0,0,9
0,0,16
0,0,15
0,0,14
0,0,13
REM MSTO TABLE
ORG MSTO-31
DUP 1,32
  0,0,MSTO*1024-##1024
REM JPROM TABLE
ORG JPROM-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
KPWNV EQU 29559
LCNSQ EQU 29551
ORG LCNSQ-15
DEC 43817,44817,45817,46817,35817,36817,37817,38817
DEC 30817,29817,28817,27817,22817,21817,20817,19817
ORG KPWNV-7
DEC 0,0,4817,6817,6817,4817,0,0
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, K3 ---
BCI 1, KB ---
BCI 1, QB ---
BCI 1, KB ---
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, Q6P ---
BCI 1, QBP ---
BCI 1, QNP ---
BCI 1, QNP ---
BCI 1, QRP ---
BCI 1, QRP ---
BCI 1, KR ---
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**

---

**APPENDIX 2**

---

**MNAVAIL**

<table>
<thead>
<tr>
<th>K -KB1</th>
<th>K - K2</th>
<th>K - Q1</th>
<th>KNP-KB3</th>
</tr>
</thead>
</table>

---

**SAMPLE INITIA INPUT/ 2 QB 1 K 2 KR 2 QBP 2 KBP KNP KRP QRP**

| 1 QNP 1 KP *QN 5 Q 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 m.</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>P×P</td>
<td>4.8</td>
</tr>
<tr>
<td>5</td>
<td>N×P</td>
<td>KB-QB4</td>
<td>.8</td>
</tr>
<tr>
<td>6</td>
<td>N×N</td>
<td>QNP×N</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>KP×P(Q6)</td>
<td>KB×P(KB7)ch</td>
<td>1.2</td>
</tr>
<tr>
<td>11</td>
<td>K-Q2</td>
<td>Q×Qch</td>
<td>.9</td>
</tr>
<tr>
<td>12</td>
<td>KB×Q</td>
<td>QB×P</td>
<td>1.5</td>
</tr>
<tr>
<td>13</td>
<td>QB×P</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>KB×Nch</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>R×N</td>
<td>.9</td>
</tr>
<tr>
<td>20</td>
<td>KB×R</td>
<td>B×KB</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
Record of game played 4/21/62. Machine - white  
R. Fiorenza - black

<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>KP-Q5</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>CB-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</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>QR-QB5</td>
<td>QR-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>
<tr>
<td>Move</td>
<td>Move</td>
<td>Move</td>
<td>Move</td>
<td>Move</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>26</td>
<td>QN-QB5</td>
<td>KB-QB4</td>
<td>KB-QB4</td>
<td>K-KR1</td>
</tr>
<tr>
<td>27</td>
<td>KN-Q4</td>
<td>KB-K2</td>
<td>KR-KB1</td>
<td>QB-K5</td>
</tr>
<tr>
<td>28</td>
<td>QR-K7</td>
<td>Q-Q3</td>
<td>Q-QB1</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>QB-KN5</td>
<td>KR-KB1</td>
<td>KR-Q2</td>
<td>QR-Q7</td>
</tr>
<tr>
<td>30</td>
<td>QBP-QB4</td>
<td>QBP-QB4</td>
<td>Q-Q1</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>KN-K6</td>
<td>Q-Q8</td>
<td>Q-QR3</td>
<td>KN-KB8</td>
</tr>
<tr>
<td>33</td>
<td>QR-KN7</td>
<td>K-KR1</td>
<td>K-KR1</td>
<td>KR-Q8</td>
</tr>
<tr>
<td>34</td>
<td>QR-QB7</td>
<td>-</td>
<td>KR-QR1</td>
<td>KR-Q7</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>*KNP-KN3</td>
<td>QN-QB3</td>
<td>*QP-Q3</td>
</tr>
<tr>
<td>3</td>
<td>*KB-KN2</td>
<td>KP-K4</td>
<td>*QP-Q3 KN-KB3 K-KN1 KP-K5 *QP-K4 QP-K5 *KN-KN5</td>
</tr>
<tr>
<td>4</td>
<td>*QP-Q3</td>
<td>QB-KB4</td>
<td>K-KN1 KN-KB3 QN-QB3 KP-K5 QP-K4 QP-K5 *KN-KN5</td>
</tr>
<tr>
<td>5</td>
<td>*QN-Q2</td>
<td>KN-KB3</td>
<td>K-KN1 KB-Q3 QBP-QB4 KP-K5 *QP-K4 QP-K5 *KN-KN5</td>
</tr>
<tr>
<td>6</td>
<td>*QBP-QB4</td>
<td>KB-Q3</td>
<td>*KN-KR4 QB-K3 KB-Q5 QBP-Q5 KN-K4 *KP-K4</td>
</tr>
<tr>
<td>7</td>
<td>K-KN1</td>
<td>K-KN1</td>
<td>*KN-KR4 QB-K3 KB-Q5 QBP-Q5 KN-K4 *KP-K4</td>
</tr>
<tr>
<td>8</td>
<td>*QBP-Q5</td>
<td>QN-QN5</td>
<td>*KP-K4 QB-KN5 QN-QB4 QB-KB6 *KB-KB3</td>
</tr>
<tr>
<td>9</td>
<td>*QN-QB4</td>
<td>KP-K5</td>
<td>*QP-K4 QB-K5 KB-KB5 *KNP-KB4 KN-Q4 *KP-K3</td>
</tr>
<tr>
<td>11</td>
<td>*QB-KN5</td>
<td>KR-K1</td>
<td>*KN-KB5 KNP-KB4 *QP-K4 KNP-K5 *KB-KB6 Q-KB3 *KP-K3</td>
</tr>
<tr>
<td>12</td>
<td>*KN-KB5</td>
<td>KNP-KB4</td>
<td>*QP-K4 KNP-K5 QB-KB6 Q-KB3 *KP-K3</td>
</tr>
<tr>
<td>13</td>
<td>*QP-K4</td>
<td>KNP-K5</td>
<td>QB-KB6 *KP-K3</td>
</tr>
<tr>
<td>14</td>
<td>*QN-Q2</td>
<td>QN-Q4</td>
<td>*QP-K4 QB-KB6 QBP-QB4 KB-K4 *KP-K3</td>
</tr>
<tr>
<td>15</td>
<td>*QN-K4</td>
<td>KR-K5</td>
<td>KB-K5 KB-KB5 *QP-KB6 QN-KB3 *Q-Q8 QR-Q1 *KB-QN7</td>
</tr>
<tr>
<td>16</td>
<td>KB-K4</td>
<td>KB-K4</td>
<td>Q-Q82 QR-QB1 *KB-Q5 Q-Q4 *KB-KB6 KB-KB3 *KP-K4</td>
</tr>
<tr>
<td>17</td>
<td>K-KR1</td>
<td>KB-K4</td>
<td>Q-Q82 QR-QB1 *KB-Q5 Q-Q4 *KB-KB6 KB-KB3 *KP-K4</td>
</tr>
</tbody>
</table>