PARALLEL SEARCH OF CHESS GAME TREES

by

Mark T. Jones

A thesis submitted to the faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science.

School of Computer Science
McGill University
805 Sherbrooke Street West
MONTREAL, P.Q.
H3A 2K6
Canada
October 1978

ABSTRACT

Two computers, communicating via a data link, have been programmed to combine when making exhaustive depth first searches of chess game trees. The computers share in the evaluation of nodes near the root of the tree by separately evaluating the successors of these nodes. It has been found that a PDP 11/45, which runs at 1 Mhz., and a PDP11/20, which runs at 0.5 Mhz., combine to produce a distributed processor which has an effective average speed of 1.35 Mhz., which is 0.15 Mhz. less than the maximum combined power of 1.5 Mhz. A final chapter compares control structures suitable for N-processor parallel tree searches.

Resume

On a programmé la combinaison de deux ordinateurs, communiquant entre eux par une ligne de données, pour effectuer une recherche exhaustive "depth first" des arbres au jeu d'échecs. Les ordinateur se partage l'évaluation des somurets près de la racine de l'arbre en evaluant séparément les successeurs de ces sommets. On a trouvé qu'un PDP 11/45, fonctionnant a 1 Mhz., et un PDP 11/20, qui fonctionne à 0.5 Mhz, se combinent pour former une unité centrale distribuée possédant une vitesse effective moyenne de 1.35 Mhz. c.a.d. 0.15 Mhz. de moins que la puissance combinée maximale de 1.5 Mhz. Une chapitre final compare des structures de controle possible pour une exploration de l'arbre en parallèle avec N unités centrales.

ACKNOWLEDGEMENTS

I thank Prof. Newborn, my supervisior, for sussestins that I investisate parallel tree search and for providins research support. I thank Prof. Ratzer for showins me how to structure the messases used for inter-computer communication.

Page

CONTENTS

Chapter 1. INTRODUCTION

"The electronic calculator plays Chess as well as it can be played on mnemonic and arithmetic lines: i.e. not very well. One feeds into the machine the symbols (pieces) and the rules as to their scope (includingsthe lensth of each file, rank, and diasonal): also the relative values, and certain important reactions(such as the effect of a capture, of Check, Mate, etc.). On this basis the machine works out and assesses arithmetic possibilities with sreat accuracy. Given a two move problem, it works out the effect of all possible moves, their permutations and combinations, til a solution is reached. This makes the treatment of a complex position into a vast operation, involvins tremendous programmins. The computer is limited by its instructions and its memory- i.e. the rules with which its circuits are fed. Although with the aid of the binary numeral system, quite an enormous equipment can be 'invested' in it, clearly it must operate mainly throush exhaustive analysis. It spells out the same, so to speak, in sinsle letters. Certain appreciations by way of 'Judsement' misht, with insenuity, be 'fed in' like shapes in shorthand. Subject to this, what it cannot do is 'plan'; and, of course, it can only disregard values under specific instructions, such as check. All the nutrition in the world is not likely to enable it to 'combine'."
(G.Abrahams, footnote in 'The Chess Mind', 1964)

Abrahams consisned computer chess to a footnote in 1964. And his description of how a computer plays by exhaustive analysis is still as true today in 1978. Does computer chess, recipient of this half page lambast still merit only a footnote or does it rather need to be born anew by an approach which enables the computer to plan ?

Paradoxically the most successful chess programs accept the lawyer Abrahams' Baconian prognosis that they can only operate throush exhaustive analysis. They accept this 'sick point', their lack of plannins, and concentrate instead on performins an exhaustive analysis as quickly as possible.

Michie [20] has pointed out that most progress has been made in computer applications by findins the most efficient

algorithms for a computer, rather than by trying to have a computer copy how a human performs the same task. In this thesis, I concentrate on the computer method of making an exhaustive tree search, with my main thrust being on how to make the computer make an exhaustive tree search more quickly by parallel processing.

Having several computers co-operate on this exhaustive analysis is an obvious idea, perhaps more obvious than the other ways currently used to achieve speedup. Levy claims that when Michie made his famous bet with him in August 1968, that no computer could beat him by August 1978, Michie envisioned having Levy face a battery of computers performing parallel search. This 'battery' does not yet exist.

Within I describe how two computers may be programmed to cooperate on the searching of chess game trees. I begin by outlining what a tree search is, give an abstracted program for chess tree searching and then briefly survey the principal methods used for speeding up searches. I also describe the concepts of parallel processing - semaphores and critical areas - which were used in the programming. The second chapter describes the board, piece and move representations used, and how moves are generated.

The third chapter describes how the two machines communicate, and how transmission errors may be detected and corrected.

This thesis attempts to answer the questions:

How can two processors be combined to make a parallel tree search ? (chapter 4)

Is this combination efficient ? (chapter 5)

How can the two processor search be extended to more than two processors ? (Chapter 6)

With only two processors available, it has not been possible to program the general N-processor search.

TREE SEARCHES

The board positions correspond to tree nodes and the moves to the lines joining them. A tree might appear as:

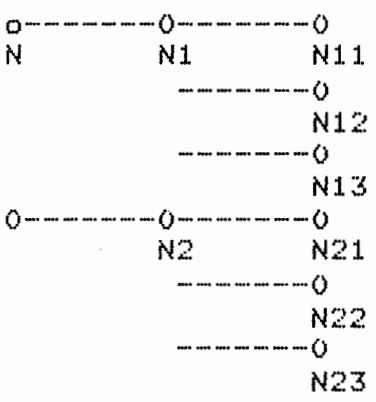


Figure 1.1 A tree of depth two.

The postfixes give the move ordering. For example, N12 is the node reached by taking the first move from the root and then the second move from node N1.

An example of a chess tree search for the position of Figure 1.2 is given below in Figure 1.3. The first column gives the first move, the second the second move. The numbers in brackets are the backed up evaluations of these moves. Search parameters, counts of the number of subroutine calls and the search time are shown below the moves. The time for the same search without the printout of the moves is 10 sec.

Given a chess board, let the task of updating it when a move is made be done by a subroutine UPDATE, and that of restoring it be done by RESTOR. Let SETMVS, or set moves, be the subroutine which generates all the moves from a given position.

```
.DATE 20-OCT-78

.R MARK

*'MAXD  8 MMAXD 10 CD 2 PDEP 2'
*'BLACK KH5 A7 B7 C7'
*'WHITE KH3 A5 B5 C5'
*'BOARD'     ....     ....     ....     ....
    ....     ....     ....     ....
    ....     ....     ....     ....
....     ....     ....     ....
.*P.  *P .*P.     ....     ....
....     ....     ....     ....
    ....     ....     ....     ....
    ....     ....     ....     ....
    ....     ....     ....     ....
....     ....     ....     ....
. P.   P . P.     ....     ....  *K
....     ....     ....     ....
    ....     ....     ....     ....
    ....     ....     ....     ....
    ....     ....     ....     ....
....     ....     ....     ....
....     ....     ....     ....   K
....     ....     ....     ....
    ....     ....     ....     ....
    ....     ....     ....     ....
    ....     ....     ....     ....
....     ....     ....     ....
....     ....     ....     ....
....     ....     ....     ....

  A    B    C    D    E    F    G    H

*
```

Fisure 1.2 Prosram printout of Board position used in search of Fis 1.3.

```
 GO
SINGLE PROC. SEARCH

PC5-C6  PB7XPC6(0)
        PB7-B6(0)
        PA7-A6(0)
        KH5-G5(0)
        KH5-H6(0)
        KH5-G6(0)
PB5-B6  PA7XPB6(600)
        PC7XPB6(600)
        PC7-C6(600)
        PA7-A6(600)
        KH5-G5(600)
        KH5-H6(600)
        KH5-G6(600)
PA5-A6  PB7XPA6(600)
KH3-G3  PC7-C6(800)
        PB7-B6(600)
KH3-H2  PC7-C6(800)
        PB7-B6(600)
KH3-G2  PC7-C6(800)
        PB7-B6(600)
BEST LINE FOUND AND SCORE:
PB5-B6  PA7XPB6 PC5-C6  PB7XPC6 PA5-A6  PC6-C5  PA6-A7  PC7-C6
PA7-A8=Q(600)

ALPHA -1000     BETA 1000       MAXDEPTH 8      MMAXD 10
CDEPTH 1        MOVES FROM CD: 24
UPDATES 31616    SETMVS 8757     CSTMVS 15394   NOMAXD: 12995
NOMMAXD 1039    NBP: 0
UPDATES/SEC: 980        SETMVS/SEC: 271 CSTMVS/SEC: 477 NOMAXD/SEC: 403
NOMMAXD/SEC: 32 NBP/SEC: 0
ELAPSED TIME(SECS): 32  & 16'60 THS SECS
*
```

Figure 1.3 Program printout during search by PDP 11/20 of position shown in Fig 1.2. The program prints moves it searches within 2 plies of the root, the principal continuation, search parameters, and counters of the number of times certain subroutines are called. Also printed is the rate at which they are called. Note that the program makes 980 board updates per. sec. On the PDP 11/45 The program executes at about 2000 board updates per. sec.

A program which uses these subroutines to traverse a tree up to a fixed depth MAXD, would have the following structure.

```
CALL TRAVERSE;
TRAVERSE:PROC(BOARD,D);
/* LET D BE THE GLOBAL DEPTH VARIABLE, INITIALLY 0 */
/* THE SUBROUTINE UPDATE INCREMENTS D, */
/* AND THE SUBROUTINE RESTOR DECREMENTS D */
DCL     MOVE(0:50),
        M; /* MOVE INDEX*/
        SETMVS(BOARD,MOVE);
        M=0;
        WHILE(MOVE(M)/=0) DO;
                CALL UPDATE(BOARD);
                IF D<MAXD THEN CALL TRAVERSE;
                /* VISIT NODE */
                CALL RESTOR(BOARD);
                END;
END TRAVERSE;
```

Figure 1.4 A program to traverse a tree.

This subroutine makes a full width depth first traversal of the tree. Full width searches, sometimes called Shannon Type A strategies [23], consider all successors from each node. Some programs, in an attempt to reduce the search effort, do not update the board for all sucessors of each node at depths below MAXD. Instead they generate all the moves and guess which are likely to be the best. These moves will be updated; the others will not. This is a Shannon Type B strategy.

Now consider how to find the best move. Assume that each player chooses that move which maximizes his score. This assumption defines an algorithm for selecting moves from scores.

Assume that a subroutine EVAL(BOARD) returns an evaluation of the board position for the player whose turn it is to move. The value of this position to the other player is -EVAL(BOARD).

The following program returns the best move for the player whose turn it is to move. I assume that the update subroutine,

when updating from depth d to depth d+1, copies vscor(d-1) into vscor(d+1).

```
DCL
        VSCOR(-1:MAXD),/* VIRTUAL SCORE,OR BACKED UP SCORE OF MOVE */
                       /* AT EACH DEPTH */
        ALPHA, /* ASSUME ROOT PLAYER CAN OBTAIN AT LEAST THIS SCORE */
        BETA,/*ASSUME OTHER PLAYER OBTAINS AT LEAST THIS SCORE*/
        D INIT(0) /* DEPTH */,
        BSTMVS(0:MAXD);/*BEST MOVES */
   VSCOR(-1)=BETA;
   VSCOR(0)=ALPHA;
   BSTMVS=0;
   CALL MAXIMUM;
MAXIMUM:PROC(BOARD,D);
/* THE SUBROUTINE UPDATE INCREMENTS D, */
/* AND THE SUBROUTINE RESTOR DECREMENTS D */
DCL     MOVE(0:50),
        M; /* MOVE INDEX*/
        SETMVS(BOARD,MOVE);
        M=0;
LOOP:   WHILE(MOVE(M)/=0) DO;
                CALL UPDATE(BOARD);
                IF D<MAXD THEN CALL MAXIMUM;
                ELSE CALL EVAL(BOARD);
                CALL RESTOR(BOARD);
                /* SCORE MOVE D+1 */
                VSCOR(D+1)=-VSCOR(D+1); /*CHANGE TO OTHER PLAYER'S
                                          VIEWPOINT */
                IF VSCOR(D+1)>VSCOR(D) THEN DO;
                        /* BETTER MOVE */
                        VSCOR(D)=VSCOR(D+1);
                        /* ALPHA BETA CUTOFF ? */
                        /* I.E. IS THE PRECEDING
                         * MOVE NOW WORSE THAN SOME OTHER MOVE ?
                         */
                        IF -VSCOR(D)<=VSCOR(D-1) THEN LEAVE LOOP;
                        ELSE BSTMVS(D,*)=MOVE(M) AND BSTMVS(D+1,*);
                        END;
                END;
END MAXIMUM;
```

Figure 1.5 A program to traverse and score a tree.

The line 'IF -VSCOR(D)<=VSCOR(D-1) THEN LEAVE LOOP' tests for alpha beta cutoffs. The player from depth d-1 is assumed already to have a move which scores vscor(d-1).

Suppose that the player from depth d has just been scored for a good move. This move is good enough to make the score of

the opposing player's previous move from depth d-1 worse than, or of equal value to, the score which he is assumed to be capable of obtaining already.

Thus the player from depth d-1 will choose not to make the current move from depth d-1. It is no longer necessary to score replies to this move. The remaining replies may be overlooked; this is done by leaving the move loop.

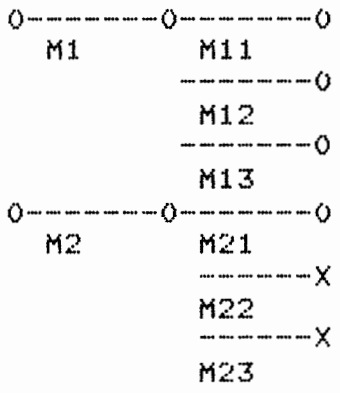


FIGURE 1.6 A search tree. The moves which need to be scored are terminated by a -O and those which do not are indicated by a -X.

In terms of Figure 1.6, M21 is such a good reply to move M2, that move M2 cannot be better than move M1. Thus moves M22 and M23 do not need to be evaluated. Alpha beta cutofffs may be seen by the absence of many replies to white's opening moves as shown in the search tree of Figure 1.4.

Although many positions are not scored, the alpha beta procedure is still considered a full width search since it still returns the best of all possible lines.

The history of the discovery of the alpha beta cutoff is given by Knuth and Moore [17].

EVALUATION OF MOVES AT MAXIMUM DEPTH OF SEARCH

Nodes at the maximum depth of search are scored by minimaxing over all sequences of captures, as described by Gillogly [12]. This quiescent search can take as much as 70% of the search time. This fraction has been reduced by setting MMAXD, the maximum depth of the quiescent search, only 2 plies beyond the maximum depth of the full search. Nodes at depth MMAXD are scored statically.

TREE SIZE AND MOVE ORDERING

The alpha beta cutoff gives significant reductions in the effort required to search a game tree. Let F, the fanout, be the average number of moves in any position. Then there are

F^{maxd} possible positions at depth d. The alpha beta search can score as few as

$2F^{d/2}$ terminal positions for even d,

and $F^{floor(d/2)} + F^{ceiling(d/2)} - 1$ terminal positions at odd depths.

Consider Figure 1.1 again. Suppose that moves M21 and M22 are bad moves and that move M23 is a good sharp reply to M2. Moves M21 and M22 will be unnecessarily evaluated before the cutoff move, M23, is evaluated. Ordering the moves, to increase the likelihood of the cutoff move being evaluated first, would speed up the evaluation of M2.

Three principal methods are used for move ordering: an ordering based upon a knowledge of chess, the killer heuristic, and iterative deepening.

The first uses a plausible move subroutine to statically

order the moves.

A killer move is a move which produces a cutoff. The killer heuristic, Slate [24] notes, is an inexpensive attempt, based on superstition, to find the killer move. A rumour is started during the search as to which ones produce cutoffs. This rumour consists of storing the moves which have produced cutoffs, in the expectation that they will continue to do so in the future. After generating moves these killer moves are ordered so as to be searched first.

Iterative deepening consists of finding the best line in lower depth searches and then examining this line first when making searches at higher depths. An iterative search, deepening at each iteration by two plies, might perform the following sequence of activities:

conduct 1 ply search,
order moves at root using results of 1 ply search,
conduct 3 ply search,
order moves at root using results of 3 ply search,
conduct 5 ply search.

The five ply search is speeded up by the move ordering based on the results of the three ply search, and the three ply search is speeded up by the move ordering made possible by the one ply search. This procedure may be used to order moves other than the root; in practice, this is often the only one so ordered.

Chapter 2. PROGRAM OVERVIEW AND CHESS DATA STRUCTURES

This chapter begins by listing the major sections of the program and then describes the particular data structures used in the chess portions of the program - the board and piece lists, the representation of a move and the data which needs to be stored when the board is updated.

PROGRAM COMPONENTS.

The major program sections are listed in Table 2.1. This will give the reader an idea of the program's capabilities. In Appendix B, I list the input commands which are recognised by the program.

The sections have been programmed by writing abstract code in pseudo PL1, continually refining sections into succesively simpler modules. The titles of these modules are given in the table of contents for the program listing, which is contained in appendix C. In the program listing, each assembly language module clearly shows what it does, its input parameters and the output it produces.

Table 2.1 Program Components.

Input
- Free format string input.
- Interpretation of these strings as commands, moves, pieces or numbers.
- Conversion of moves,pieces, numbers from character to internal form.
- Place a piece on the board.
- Recognise input of moves when playing a game.

Output
- Conversion of moves, pieces to character form.
- Conversion of 16 bit numbers to character form under any base.
- Conversion of 32 bit numbers to decimal character form.
- Formating of numbers and strings.

Double precision arithmetic
- Addition,subtraction.
- Multiplication, division.

Play a game.

Initialise board, update board, restore board.

Generate moves, including castling, queening pawn and nighting pawn.

Machine Communication.
- Buffer control for holding messages.
- Message reception and transmission with capability for recovering from transmission errors.

Tree control for single processor searches.

Parallel Searches.
- Tree control.
- Produce messages for other processor.
- Interpret contents of received messages and produce appropriate replies.

Gather statistics on parallel searches.
- Time individual subroutines.
- Direct parallel searches with varying search parameters.
- Analyse results of parallel search for both machines.
- Produce table of results for parallel searches.

I refer the reader to the program listing for details of program I/O, double precision arithmetic and how the program plays a game. I will now turn to the chess data structures I have used.

PIECE REPRESENTATION

A piece list is necessary so that all the pieces can be passed over when generating moves. Slate confesses [24] that his first programs had no list of pieces; each time the moves for a player were to be generated, the board had to be searched for the pieces.

Pieces are contained in a 32 element structure array:

```
DCL     1 PIECE(0:31),
                2 ON POINTER,
                2 CLR,  /* COLOUR */
                2 TYP,  /*PIECE TYPE NUMBER */
                2 VAL,  /* VALUE */
                2 NXTP POINTER, /* NEXT PIECE */
                2 LSTP POINTER, /* LAST PIECE */
                2 DADD POINTER; /* ADDRESS OF PIECE'S DIRECTIONS */
                                /* OF MOVEMENT */
```

The ON field contains a back pointer to the square on which a piece rests. When the piece moves it is necessary to update this field. The CLR field is 0 for black and 1 for white pieces.

The TYP field gives the piece's type number, and the VAL field gives the material value of the piece. The DADD field points to a list giving the compass directions in which a piece may move. Each direction list is terminated by a zero.

The TYP AND VAL fields and the directions in which a piece may move take on the following values: (Fig. 2.1)

PIECE	TYP	VAL	DADD->directions
knight	4	300	ssw,sse,wsw,ese,wnw,ene,nnw,nne,0
bishop	6	315	nw,ne,sw,se,0
rook	8	500	n,s,e,w,0
queen	10	900	n,s,w,e,nw,ne,sw,se,0
king	12	5000	n,s,w,e,nw,ne,sw,se,0
pawn	14	100	n,nw,ne,0 for white; s,sw,se,0 for black
unmoved rook	16	500	n,s,e,w,0
unmoved king	20	5000	n,s,w,e,nw,ne,sw,se,0
unmoved pawn	22	100	n,nw,ne,0 for white; s,sw,se,0 for black

Figure 2.1 Piece types, values and the directions in which they move.

When generating moves, the piece type field allows a fast table jump to the relevant code for generating that piece's moves.

Unmoved pawns are distinguished from moved ones by their piece type. One does not then need to test which square a pawn is on to see whether it may move forward two squares. And by having a pointer to the directions in which a pawn may move contained in the piece array, one need not test the colour of a pawn to see which direction it may move in.

Unmoved kings and rooks are distinguished so that castling rights may be ascertained.

Unmoved pieces are distinguished by having a TYP field >=16. When an unmoved piece is moved, 8 is subtracted from its TYP

field. It then becomes its moved counterpart.

The active pieces of each colour form two doubly linked chains. The NXTP and LSTP fields are the chain pointers. When updating and restoring the board a piece may need to be removed from or restored to the chain of active pieces. A doubly linked chain makes it easy to find the preceding and succeding pieces.

The doubly linked list for pieces of each colour always commences with the king. The black king is piece(0), and the white king is piece(16). The other black piece indexes are 1 to 15, and those for white 17 to 31.

BOARD REPRESENTATION

Shannon [23] first suggested representing a chess board as a 64 word array which contains the type numbers of the pieces. A white piece would be represented by a positive number, and a black by a negative. For example square(0), or a1, might contain a 4 to indicte a white rook.

My representation is slightly different. The squares form a structure array:

```
DCL
        1 SQUARE (0:63),
                2 PCE POINTER,
                2 CSQ CHAR(2), /* CHARACTER FORM OF SQUARE */
                2 NXTSQ (0:15) POINTER;
```

The PCE field of a square contains a pointer to the piece occupying the square. This is null when the square is empty.

Note that the first element of the piece array contains a pointer to the square. This tight linkage between squares and pieces makes it easy to change a pointer to a piece into a pointer to the piece's square, and vice versa. Suppose R4 points to a piece/square. Then the two cycle machine operation

```
MOV  (R4),R4
```

turns R4 into the corresponding square/piece pointer.

The NEXTSQ subarray contains 16 pointers to the adjoining squares. These 16 squares are the eight squares immediately surrounding any one square whose compass directions are SW, S, SE, W, E, NW, N, NE; and the other eight squares are those attainable by a knight. Their compass directions are SSW, SSE, WSW, ESE, WNW, ENE, NNW, NNE.

A null pointer in an element of NXTSQ indicates that the adjoining square in the direction indexed is off the board. For example, square A1 has no square to the west of it and so SQUARE(0).NXTSQ(W)=0.

This representation of the board and pieces permits common code for generating the moves of many pieces. The moved king and the knight move to any of their 'adjoining' squares. The 'adjoining' squares are those non-null squares in the NXTSQ array in the directions in which the piece is free to move. Queens, bishops and rooks may move as far as they wish in the directions in which they are free to move, provided there is no piece blocking them.

REPRESENTATION OF MOVES

Each move generated occupies 3 words. Its structure is:

```
DCL
        1 MOVE POINTER(*:*),
                2 TYPE,
                2 TO POINTER,
                2 FROM POINTER;
```

The FROM and TO pointers indicate the squares from and to which a piece may move. The move type is generally the type of the moving piece. Exceptional conditions arise in the following four cases:

MOVE	TYPE #
castling	30
en passant capture	32
queening pawn	34
knighting pawn	36

A bishoping pawn and a rooking pawn are omitted. Their moves are a subset of the queening pawn moves.

The high move type value enables a simple test for exceptional conditions:

```
IF MOVE(M).TYPE>=30 THEN DO;
        /* code for exceptional move */
        END;
```

and thus the board may be updated or restored with only one test for exceptional conditions needing to be made.

REPRESENTATION OF UPDATED MOVES

The structure declaration for an updated move is :

```
DCL
        1 UMOVE(*:*) BASED(D),
                2 MOVE
                        3 TYPE POINTER,
                        3 TO POINTER,
                        3 FROM POINTER,
                2 LAST_BASE POINTER, /* POINTS TO BASE(D-1) */
                2 VSCOR,
                2 CPCE POINTER, /* CAPTURED PIECE */
                2 SCOR, /* MATERIAL ON BOARD AFTER UPDATE */
                2 CHK, /* IS THE PLAYER IN CHECK ? */
                2 BMV(*:*) LIKE MOVE; /* BEST MOVES FOUND */
```

LAST_BASE is a pointer to the preceding updated move, at depth D-1. VSCOR, or virtual score, is the current backed up score. Initially vscor(-1)=beta, vscor(0)=alpha. These values are carried down the stack. When updating to depth D, the new VSCOR comes from the second preceding node:

```
VSCOR(D)=VSCOR(D-2)
```

If the move is a capture, the piece captured is stored in CPCE(D), otherwise CPCE(D) is null. SCOR(D) contains the value of the

material on the board after the move has been updated. CHK(D) indicates whether the player to move is in check. This value is not strictly necessary, but it is useful when printing moves to know whether a move has put a player in check.

STORAGE ALLOCATION FOR UPDATED AND GENERATED MOVES.

Updated and generated moves share the same storage area. In a depth first search a stack may be used for storing updated moves and also for storing moves. I have chosen to place them on the same stack. Its use is as follows. The moves are generated; the move at the top of the stack is updated first. When this move has been scored, the board is restored and the updated move is removed from the stack. The next move is now on top of the move stack and is the next one to be updated. The stack might appear as follows:

```
OCTAL
ADDRESS
10000    VSCOR(-1)        1000
 7776    BASE(-1)         0000
 7774    BASE(0)          07776
07772    VSCOR(0)         00000
07770    CPCE(0)          00000
07766    SCOR(0)          00000
07764    CHK(0)           00000
07762    BMV(0)           00014    P
                          05630    E4
                          05110    E2
                          00000
The generated moves immediately following this area:
         ...
         ...
         FROM                0    END MARKER
         TYPE                4    N
         TO              04764    C3
         FROM            03452    B1
         TYPE                4    P
         TO              04640    A3
         FROM            04120    A2
         TYPE               14    P
         TO              05360    A4
         FROM            04120    A2
         TYPE               14    P
         FROM                     D4
         TO                       D3
BASE(1)->                 7774    POINTER TO BASE(0)
         VSCOR(1)
         CPCE(1)
         SCOR(1)
         CHK(1)
         BMV(1)
```

Figure 2.2 An example of how storage might be allocated for storing moves and remembering an updated move.

After the move pd4-d3 is evaluated, the move on top of the stack is pa4-a2.

Note that in PDP11 machine language, stacks grow downwards to lower addresses. Thus the addresses shown are in descending sequence. In what follows I will always represent memory addresses as descending, so that roots of trees, at high addresses, are at the top of, and grow down, the page.

CHAPTER 3. INTERMACHINE COMMUNICATION.

In this Chapter I describe the hardware used to communicate between the two machines, when this link failed to operate correctly, how transmission errors can be detected, and how to recover from them. Consideration of error recovery procedures then leads into sections describing two particular techiques used in programming these procedures: queues and semaphores.

DR11A DEVICE INTERFACE

All inter-machine communication was performed using the asynchronous parallel link installed by W. Striesel in 1976. He installed a modified DR11A. The device most similar to this and described in the DEC Peripherals Handbook [4] is the DR11C. The implementor claims that it transmits data at an average rate of 35 microseconds per 16 bit word.

The parallel link consists of a control and status resister, an output buffer and an input buffer. The addresses of the parallel link resisters are:-

```
PLSR=164000     Parallel link status resister.
PLOB=164002     Parallel link output buffer.
PLIB=164002     Parallel link input buffer.
PLVC=   320     Parallel link interrupt vector.
```

One of these devices is attached to the UNIBUS of each machine. The devices on the two machines are connected by two cables, one for each direction.

The link permits full duplex transmission. A word of data may be placed in the output buffer by an assembly language command such as :

```
MOV  R0,@#OUTBUF
```

When an output operation is begun, the output data is stored in the 16-bit output register. A NEW DATA READY signal is sent to the link interface to indicate that data has been loaded by means of a DATAO or DATOB bus cycle.

The data passes along the cable to the input buffer of the other machine. This is a 16 bit read only register used for transmitting the data to the UNIBUS. This register may then be read by a DATI sequence onto the UNIBUS. When this occurs a pulsed signal, DATA TRANSMITTED is sent to the sender's device and informs it that the transfer has been completed.

The transmitted data is accessible by an assembly language command such as :

```
MOV @#PL1B,R0
```

The control and status register provides four bits which can be used to control and monitor data transfer. (The error bits described in the PDP Handbook are not implemented.) Of the four bits which are implemented, two are set by the device and are not under program control.

One of them, BIT 7, indicates that data has been received. This event is called a REQUEST A. When data has been transmitted, a REQUEST B occurs at the sender and is indicated by the setting of bit 15. This bit remains set until the data has been read by the other machine.

These two bits are used to synchronize transmission. When transmitting, the program waits until REQUEST B is cleared. A macro suitable for sending a word of data is :

```
.MACRO SNDW
TST  @#PLSR
BMI .-4
MOV  R0,@#PLOB
.ENDM
```

When receivins data the prosram waits until REQUEST A is set. A suitable MACRO is :

```
.MACRO RECVW
TST  @#PLSR
BPL .-4
MOV  @#PLIB,R0
.ENDM
```

Data transfer may also be interrupt driven. Two bits in the status resister provide for this. When INTERRUPT ENABLE A (BIT 6) is set , an interrupt occurs upon receivins a word of data. The correspondins interrupt for sendins data, INTERRUPT ENABLE B (BIT 5) was found to be inoperative.

When an interrupt occurs, the prosram stores the current prosram counter and processor status word on the system stack. It then loads a new prosram counter from PLVC and a new processor status word from PLVC+2. Thus PLVC should contain the address of the interrupt service routine.

	ADDRESS	
PLVC	320	Address of ISR.
PLVC+2	322	interrupt status word

Fis 3.1 Parallel link interrupt vectors.

When the machine is turned on, the INIT signal at power up clears the PLSR so that no interrupt will occur unless PLSR is initialised under prosram control.

ASSESSMENT OF THE DR11A.

The DR11A only runs in receive interrupt mode. Each word took on averase of 34 microseconds to transmit. The receiver would attempt to continue processins durins this interval.

It takes some 25 machine cycles to test for the end of a message, save registers and continue processing. Thus little useful work can be done whilst waiting for a complete message to be received.

The link lacks a send interrupt feature. This meant that the sender had to remain in the send subroutine whilst transmitting a message. No other work can be done whilst transmitting a message.

ACTUAL TRANSMISSION ERRORS.

No bit errors, detectable by a checksum inconsistency, have been detected. The link cable is some 15 metres long and connects the two computers directly. Interference is consequently low and this type of error was not expected.

The DR11A sometimes fails to transmit a word of data. Counts of the number of words sent and received confirm that a word has been 'lost', since they differ by one. I believe that this is a hardware problem.

This loss of a word is always preceded by a second receive interrupt occuring before the first has been serviced. Suppression of the receive interrupt by clearing the RECEIVE INTERRUPT ENABLE A bit failed; a word would still be lost.

I believe that clearing the INTERRUPT ENABLE A bit, setting it or allowing a receive interrupt to occur causes the sender to believe that the transmitted data has been read. To provide correctly synchronised transmission, the sender's REQUEST A flag should only be cleared when the data is accessed. When INTERRUPT ENABLE A is accessed, or an interrupt occurs, the sender's REQUEST A flag should not be cleared by hardware.

I now turn my attention from the hardware to the use of it. Successful transmission of messages requires detection and recovery from transmission errors. The procedures used in this project are described in Gray [13]. Errors are detected by sending a checksum with all messages.

ERROR RECOVERY

After an error has been detected, it is necessary to retransmit the original message. The receiver and sender must co-ordinate this retransmission in such a way that no message is lost or duplicated. This is effected by :

1. Numbering messages.
2. Sending an ACK to acknowledge successful receipt of a message, and a NAK when the message has been incorrectly received.

When there may be more than one outstanding message being transmitted, it is necessary to number the ACK's and NAK's so that they may be distinguished. Note that there exists two sequences of message numbers: one for each direction.

MESSAGE FORMAT

All messages have the format shown in Fig 3.1.

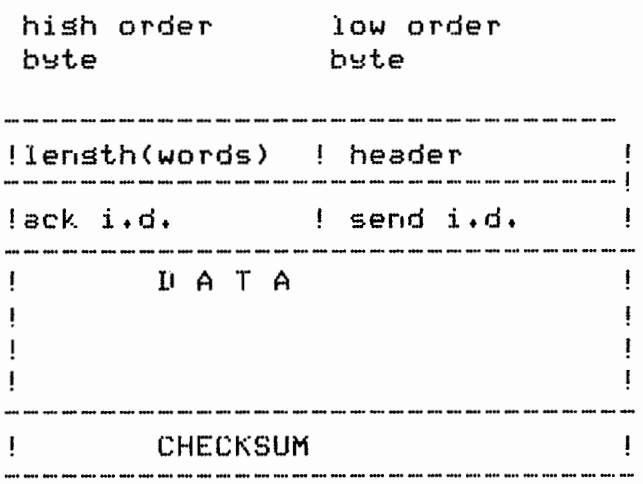


Fig. 3.1 Message format.

The first word contains a message header pattern in the low order byte and the length of the message in the high order byte. The second word contains message IDs. The low order byte contains the sender's identification for the message, and the high order byte contains an acknowledgement ID for any message that the receiver may have sent.

The third and any subsequent words contain data. The last word sent is the checksum, which is such that the sum of all the message words is zero.

Simple acknowledgements are a special case of the above format, for which there is no data. The message is then 3 words long. Negative acknowledgements have a negative acknowledgement ID, the negative of the expected ID.

When SID=0, The message needs no acknowledgement, since it is itself merely an acknowledgement. I decided to send a

checksum with a simple acknowledgement so that all messages would have the same format. This also reduces the probability of an ACK or NAK being incorrectly received, since they then also carry a checksum.

MESSAGE RECEPTION.

This message format permits a message to be received in the following manner .

The receiver is active when it is receiving data, inactive when listening for the beginning of a message. When inactive, the receiver recognises the beginning of a message when a word containing the message header in its low order byte is received. The receiver then becomes active and accepts the rest of the message, the length of which is given in the high order byte of the message header word.

Whilst the receiver is active, it is data insensitive. A receipt of the header pattern would be treated as data. When it is inactive, it can only become activated by the header.

PARALLEL LINK SERVICE ROUTINES.

Receipt of a word causes an interrupt to ISR, the interrupt service routine. It tests to see which of three states the receiver is in.

STATE	ACTION.
1. Expecting header	If header, then set length of message expected and pass to state 2. Otherwise ignore data and remain in state 1.

2. Expecting l.d.	Add data to buffer.
and data.	Check for buffer overflow.
3. Expecting checksum.	Add data to buffer.
	Call INTPT routine.
	Return to state 1.

TESTING MESSAGE VALIDITY AND ERROR RECOVERY.

The following describes the routine INTPT which tests for detectable transmisssion errors.

Every message received may require a reply, so an attempt is made to gain control of the output buffer . If the buffer is not available, the message is not interpreted. This will be done by a different invocation of INTPT from the interrupt service routine.

The checksum is tested. If in error, a NAK is sent. Otherwise the message has a valid format.

The SID is then checked to see if it is the one expected. If so the message is accepted and its contents interpreted. If it is that of a previously accepted message, the message is ignored. If the SID is too high, a NAK of the expected message is sent (this triggers retransmission of the other machine's output buffer).

The AID is tested next. If zero, nothing is done. If negative, signaling a NAK, the output buffer is retransmitted. If AID is greater than zero, all messages upto and including the one acknowledged are freed from the output buffer.

Normal error recovery will detect the loss of a message by the out of sequence SID of the succeding message. It sometimes happens that the sender may only proceed after a reply to a message has been received. A deadlock might occur when a processor is waiting for the reply to a message which has not been successfully received. In these situations, to avoid a deadlock, the sender will wait for a fixed time and then if no reply has been received, retransmit the message.

I close this chapter with a description of how queues and semaphores were programmed.

QUEUES

Queues were required for the holding of messages in the parallel link data buffers. The program generates messages which are queued ready for the subroutine which places the words of data in the parallel link.

An examination of the queueing algorithm in Knuth [16] revealed that his algorithms are inappropriate for implementaton in PDP11 assembly language, in that

1. His buffer pointers are not directly related to the physical address of the data. They point instead to the word before the data.

2. A direct implementation of his algorithms makes impossible the use of PDP11 autoincrement and autodecrement modes of addressing, wherein data may be moved to the address contained in a register, whilst at the same time incrementing or decrementing this register to point at the previous or following address.

If registers point directly to the front or end of a queue,

it is then possible to both write more understandable queuing algorithms and use assembly language autoincrement and autodecrement modes.

Let P1 be a pointer to the front of a queue, and P2 be a pointer to the end of this queue. Then P1 points to the word at the front of the queue and P2 to where another word may be added to the queue.

Use pointers MINP and MAXP to give the limits of the queue storage area. All pointers have the values deducible from examining the following diagram.

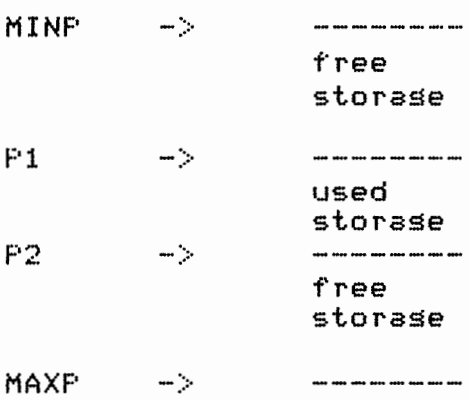


Figure 3.2 The storage area used by a queue.

The following triplets of code permit addition and removal of a word of data from the queue.

```
ADDITION                          REMOVAL

(P2)+=DATA                        IF P1=P2 THEN UNDERFLOW
IF P2=MAXP THEN P2=MINP           DATA=(P1)+
IF P2=P1 THEN OVERFLOW            IF P1=MAXP THEN P1=MINP
```

The operation (P2)+=DATA means that the data to be added to the queue is placed at the address given by P2, and then P2 is incremented to point to the next word.

SEMAPHORES.

These are required when several processors wish to have exclusive use of a shared resource at the same time. Parts of the program code which only one processor may execute at any one time are critical areas.

For example, an ordered sharing is required when two or more messages have been received and, through interrupts, two processors are trying to reply to different messages. The shared resource is the parallel link output buffer.

A second example is the allocation of exclusive access to the critical part of the search tree. The shared resource is the root area of the search tree.

Semaphores are easily implemented on a PDP11 since there are indivisible machine operations:

```
INC  S
DEC  S
```

which either increment or decrement the variable S and also set the zero bit of the program status word in one operation. When a semaphore S has the value one, only one processor may absorb the signal S by a decrement operation and set S to zero at the same time.

Suppose any processor which tries to absorb the signal S and finds it non-zero increments S and then tries again to decrement S to zero. It could go into a busy loop trying to become the one processor which decrements S and simultaneously sets S to zero.

Would this be a logically correct implementation of semaphores? Dijkstra [5] states that it must be impossible for two processors to deadlock in an 'after you' - 'No, after you' conversation where each repeatedly tries to become the processor which decrements

S to zero. The following is an example of how two processors P1 and P2 may become in deadlock.

S	P0	P1	P2
0			
0		DEC S	
-1	INC S		
0			DEC S
-1		INC S	
0		DEC S	
-1			INC S
0			DEC S
-1		INC S	
0		DEC S	
-1			INC S
0			DEC S
-1			

Fig 3.3. Interleaving of semaphor operations by processors P1 and P2 creates a deadlock.

Processors P1 and P2 each want to decrement S to zero. The conflict could go on ad infinitum.

Now suppose instead that a process dies when it is blocked. Thus in the above P1 and P2 would die when they decrement S to less than zero. Processor P0 on exiting from the semaphored process notices that it has incremented S to zero or more. This shows that some other process has been blocked, and indicates that process P0 should not die but repeat execution of the semaphored process. Thus when a processor inside a critical area increments a semaphore to zero or more, it should repeat execution of the critical area on behalf of some blocked process.

SEMAPHORE IMPLEMENTATION ON TWO MACHINES WITHOUT COMMON MEMORY

Where is the semaphore to be stored when two machines do not share memory? The implemented method is to have a copy of the

semaphore on each machine, and when the semaphore is blocked, and an operation is done on the semaphor, a message is sent to the other machine.

I did not find any simple way of having the machines negotiate between themselves which one should have initial value 1 when only one semaphor signal is initially available. Instead the machines are told by the switch registers which one has initial value one. The general solution to this problem is given by Dijsktra [5] - how to have a negotiated access to a critical area without deadlock.

Chapter 4. PARALLEL SEARCH

Parallelism will exist in a tree search program at two levels: within particular subroutines or at a more abstract level wherein different nodes of the tree are examined concurrently. Consider the hierarchical structure of the MINMAX subroutine. Subroutines UPDATE, RESTOR and SETMVS are all called from MINMAX. The parallellism may be sought at a low level, within move generation for example, or at a higher level, in MINMAX. I first examine the former.

PARALLEL SUBROUTINES.

Consider move generation, which is a program bottleneck: The technology chess program was reported to spend 70% of its time generating moves. Thompson has speeded up his chess program BELLE by means of a chess specific microprocessor move generator.

Move generation is highly parallel. The moves of each piece in each direction may be generated independently. Having several processors generate moves in parallel is possible and this speedup in move generation will greatly reduce overall search time.

However this will only speed up one subroutine. If the rest of the program is only executed by one processor, then new bottlenecks will appear in other parts of the program. More significant speedups are potentially available by assigning the many tree nodes to different processors.

PREVIOUS WORK ON PARALLEL TREES.

Marshall, in his Ph. D Thesis [18] and elsewhere [19] describes

how he has simulated the parallel evaluation of decision trees in PL/1. He claims that the parallel evaluation of many nodes at the same depth, which turns the search into a partial width first search, decreases the time taken for complete evaluation of a tree by more than the increase in the number of processors. Unlike a a depth first search, he searches a decision tree iteratively. He attributes the speed increase to the spread of the processors enabling them to stumble on the decision path more quickly.

I would point out that this spreading may be effected by a single processor; that in fact is what his 'simulation' did. He cannot therefore claim that the unexpected extra speedup is due to the parallel algorithm. It is due to a different search strategy.

Several papers [3,8,25] have discussed parallel evaluation of binary trees. They are primarily concerned with balancing them so that they are 'bushy'. This produces many candidate nodes for parallel evaluation, as well as balancing the work load amongst these nodes.

Game trees differ from binary trees in that they cannot be balanced, and in that they are not simple AND trees, but mixed AND/OR trees. An AND node is one for which, in order to score the node, all the successors need to be evaluated; An OR node is one for which there is one successor which will completely score the node.

PARALLEL NATURE OF AND/OR TREES.

Cutoffs introduce OR nodes into the tree, and it is not known beforehand which nodes will have to be examined. If a separate processor is assigned to each of the OR successors of a node, then it may turn out that the 1st OR node is sufficient to score a node. The processors engaged in scoring the other

successors are redundant.

An AND node requires that all its successors be evaluated. One processor may be assigned to each successor without immediate duplication of effort. However the amount of work required to score an AND successor is reduced by narrowing the alpha-beta bandwidth. Thus one processor scores an AND node with less effort than many; the single processor will have a narrow alpha-beta bandwidth when scoring many of the successors.

Suppose many processors commence a search with wide alpha-beta values. Since they will all finish their searches at about the same time, the benefit of a narrow bandwidth is occluded. This disadvantage is partially overcome by passing updated width parameters to effected processors during their search.

An efficient parallel search should therefore aim to:

1 Have only one processor evaluate OR nodes.

2 Quickly achieve narrow alpha-beta windows so that the evaluation of the successors of AND nodes becomes independent and may thus be efficiently performed in parallel.

SEARCH STRATEGY.

I have considered the following two strategies for distributing nodes amongst processors:

Split the processors at a fixed common depth. In the following diagram, the common depth, CD, is 1. The terminal nodes shown are the candidates for parallel evaluation.

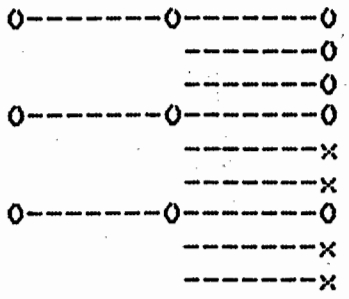


Figure 4.1 Common root area of search tree for CD=1.

The second strategy is to use a variable common depth. In the following CD decreases from 3 to 0 and then is set to 3 again.

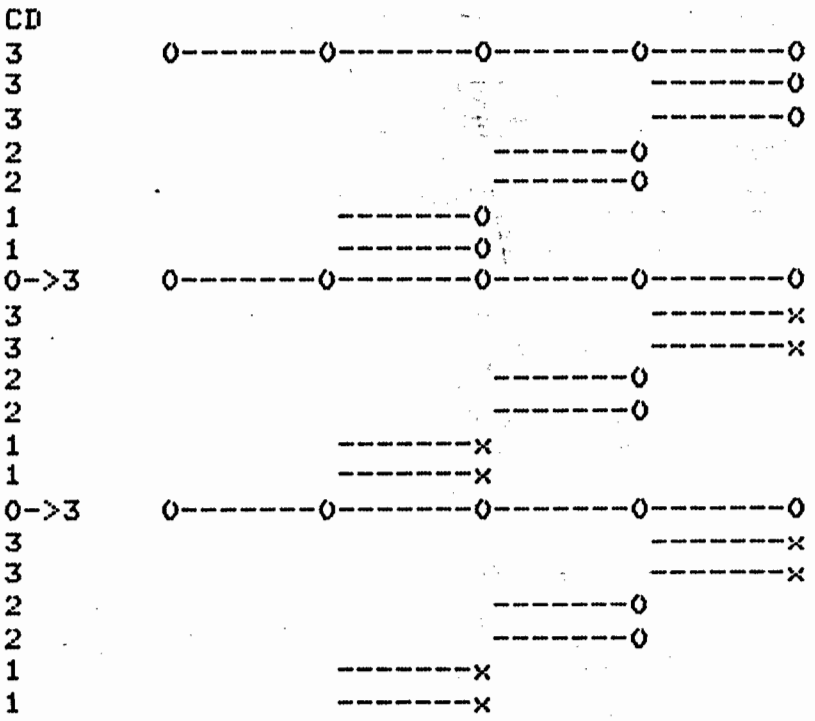


Figure 4.2 Common root area of search tree
for a variable common depth strategy.

The alpha beta search has the property that the first successor of each AND node is an AND node. The moves which might not need to be evaluated are again marked by an x.

EXAMPLE - PROGRAM PRINTOUT.

The position of Fig 1.2 was searched in parallel mode with a fixed common depth of 1. The tree of Fig 4.3 shows which processor searched which node. Program printout for this parallel search is shown in Figs. 4.4 and 4.5. The moves from CD=1 are shared between the two processors.

The move PC5-C6 gives an AND node for which all the

			Processor	
PC5-C6	-------	PB7XPC6	11/45	CUT OFF BY 11/20
	-------	PB7-B6	11/20	
	-------	PA7-A6	11/20	
	-------	KH5-G5	11/45	
	-------	KH5-H6	11/20	
	-------	KH5-G6	11/45	
PB5-B6	-------	PA7XPB6	11/20	CUT OFF BY 11/45
	-------	PC7XPB6	11/45	
	-------	PC7-C6	11/45	
	-------	PA7-A6	11/20	
	-------	KH5-G5	11/45	
	-------	KH5-H6	11/20	
	-------	KH5-G6	11/45	
PA5-A6	-------	PB7XPA6	11/20	
KH3-G3	-------	PC7-C6	11/20	
	-------	PB7-B6	11/45	
KH3-H2	-------	PC7-C6	11/45	CUT OFF BY 11/20
	-------	PB7-B6	11/20	
KH3-G2	-------	PC7-C6	11/20	CUT OFF BY 11/45
	-------	PB7-B6	11/45	

Figure 4.3 Root area of tree searched by both processors, for position of Fig. 1.3.

successors will be searched. The first successor, PB7xPC6 is taken by the 11/45, and the second, PB7-B6 by the 11/20. The 11/20 finishes scoring its position (SCORE=0) first, and continues with the move PA7-A6.

The score of the 20 cuts off the move PB7xPC6 being searched by the 45, as is seen by the move's not being scored (Fig 4.4). The 45 continues with the next move, KH5-G5. The search continues similarly through the moves PB5-B6, with the processors sharing

```
 GO
PARALLEL SEARCH
LEAD SEARCH INITIALLY

PC5-C6  PB7XPC6
        KH5-G5(0)
        KH5-G6(0)
PB5-B6  PC7XPB6(600)
        PC7-C6(600)
        KH5-G5(600)
        KH5-G6(600)
KH3-G3  PB7-B6(600)
KH3-H2  PC7-C6
KH3-G2  PB7-B6(600)
BEST LINE FOUND AND SCORE:
PB5-B6  PC7XPB6 PA5-A6  PB7XPA6 PC5-C6  PB6-B5  PC6-C7  PB5-B4
PC7-C8=Q(600)

ALPHA -1000     BETA 1000       MAXDEPTH 8      MMAXD 10
CDEPTH 1        MOVES FROM CD: 12
UPDATES 17860    SETMVS 4801     CSTMVS 8716    NOMAXD: 7563
NOMMAXD 628     NBP: 0
UPDATES/SEC: 1644       SETMVS/SEC: 442 CSTMVS/SEC: 802 NOMAXD/SEC: 69
NOMMAXD/SEC: 58 NBP/SEC: 0
ELAPSED TIME(SECS): 10  & 52'60 THS SECS
BUSY LOOP TIME(SECS): 0 & 1     '60 THS SECS,%= 0
LINK-RECEIVE INTERRUPTS: 0      WORDS RECEIVED: 360     WORDS SENT: 43
*
```

Figure 4.4 Printout by PDP11/45 during a parallel search of the position shown in Fig. 1.3. The processor searched the subtrees of the moves printed. The BUSY LOOP TIME is the end of search idle time, and is seen to be negligible.

the moves between them. The move PA5-A6 is scored by th 20 before the other processor takes a move in that search tree; the 11/45 does not make any updates for that move. The move KH3-H2 shows a speedup effect due to the parallel search: in fig. 1.3, the single processor search completely scores two successors: PC7-C6(800) and PB7-B6(600). In a parallel search, the score for PB7-B6 is found before that of PC7-C6 and cuts it off (fig. 4.5 has no score for PC7-C6).

```
 'GO' PARALLEL SEARCH

PC5-C6  PB7-B6(0)
        PA7-A6(0)
        KH5-H6(0)
PB5-B6  PA7XPB6
        PA7-A6(600)
        KH5-H6(600)
PA5-A6  PB7XPA6(0)
KH3-G3  PC7-C6
KH3-H2  PB7-B6(600)
KH3-G2  PC7-C6
BEST LINE FOUND AND SCORE:
PB5-B6  PC7XPB6 PA5-A6  PB7XPA6 PC5-C6  PB6-B5  PC6-C7  PB5-B4
PC7-C8=Q(600)

ALPHA -1000     BETA 1000       MAXDEPTH 8      MMAXD 10
CDEPTH 1        MOVES FROM CD: 12
UPDATES 10783    SETMVS 2958     CSTMVS 5199    NOMAXD: 4344
NOMMAXD 411     NBP: 0
UPDATES/SEC: 964        SETMVS/SEC: 265 CSTMVS/SEC: 465 NOMAXD/SEC: 388
NOMMAXD/SEC: 37 NBP/SEC: 0
ELAPSED TIME(SECS): 11  & 11'60 THS SECS
BUSY LOOP TIME(SECS): 0 & 1     '60 THS SECS,%= 0
LINK-RECEIVE INTERRUPTS: 0      WORDS RECEIVED: 430     WORDS SENT: 360
*
```

Figure 4.5 Printout by PDP11/20 during search shown in Fig 4.4 . Note that the 11/20 makes 964 board updates per. sec., whilst the 11/45 makes 1644.

In single processor mode, the 11/20 searches the position at 980 positions/sec, whilst the 11/45 searches the position at about 2000 positions/sec.

CRITICAL AREAS OF THE SEARCH TREE

Both processors share in the evaluation of moves below the common depth(D<=CD) as directed by the program MINMXP. This program directs a tree search which takes note of the best moves, scores and alpha beta cutoffs found by the other processor.

Tree searches above the common depth are done privately by each machine, in a subroutine called MINMX2. This subroutine is called from MINMXP when an update is to be made which will increase the depth beyond the common depth. Control is returned back to MINMXP when the board is restored back to the common depth.

Since both machines share in the evaluation of nodes at and below the common depth, it is possible for the two machines to interfere with each other when scoring and distributing moves between them. In order to eliminate this, only one machine may access nodes at or below the common depth at any one time. Since such operations are all done within MINMXP, this amounts to making MINMXP a critical area. Access to MINMXP is gated by the semaphor CA.

IMPLEMENTATION DATA STORAGE

The program of chapter 1 is implemented as a stack. I felt that it would be easier to keep the stack representation than to use trees. This necessitated having two copies of the root area of the stack, one for each processor. I outline in chapter 6 how to implement a parallel search with a tree representation of updated moves. This would allow one area of memory to hold the root area of the tree used by all processors.

The storage map for both processors is shown in Figure 4.6.

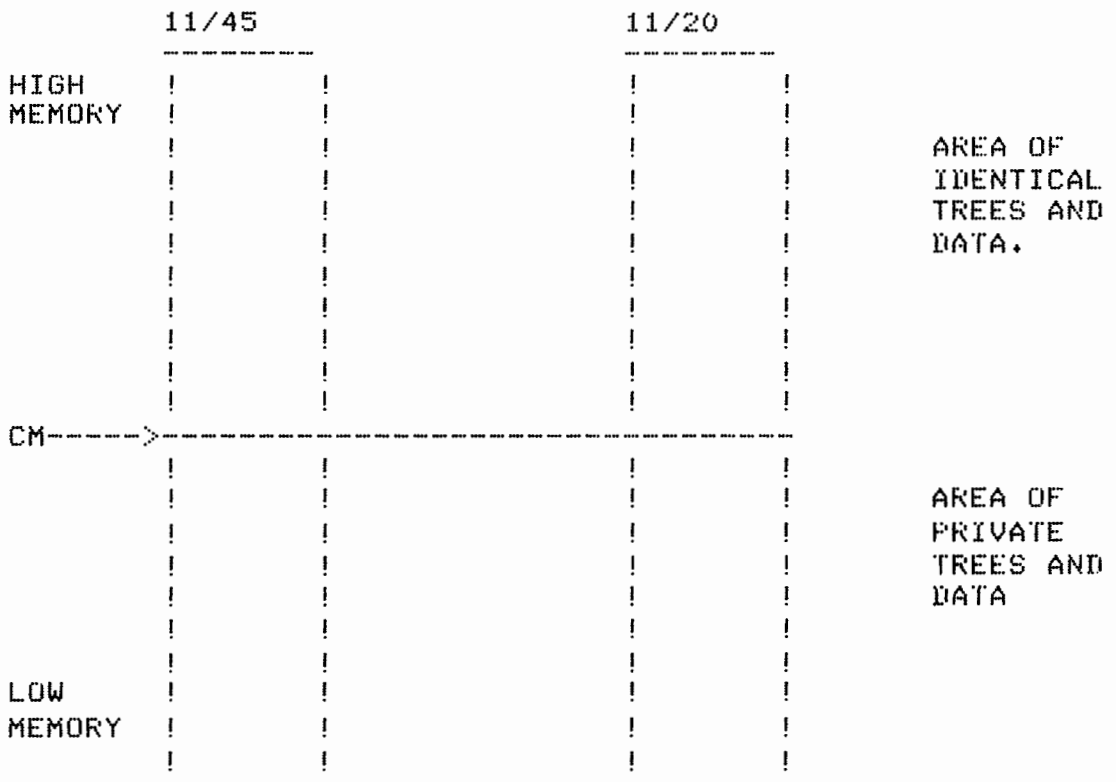


Fig. 4.6 Private and shared portions of move storage area.

I will now discuss the program of Figure 4.7 which is an abstract version of the assembly language program (appendix C pages 63-67).

PROGRAM VARIABLES

The followins variables direct the parallel search.

CD The common depth. Moves from depths <=CD are shared between the two machines.

LEAD A flag set up when this processor is currently leading the other.

CM Common move, the move stack pointer, above which address the processors have the same generated and updated moves tree.

CA Critical area semaphor. It is used to assign exclusive

access to the critical area.

NOIN(D) The number of active processors. At the end of the search, NOIN(0)=0 indicates that all processors have terminated searching.

HANDLING TWO COPIES OF THE ROOT OF THE TREE

The processors must know each other's position in the search; by noting that moves are all stacked, I realized that the move stack pointer was sufficient to indicate how deep in the search tree a processor was. The common stack pointer, the depth in the move stack up to which both processors have an identical search tree, is sufficient for determining when messages need to be sent or requested:

Only the evaluations of shared nodes need to be communicated. The processors share updated and generated moves which lie at address above CM.

When a processor restores its board and clips its search tree, the common search tree must necessarily be clipped by the same amount. Let M be the point to which a processor has restored its board. When M>CM, then M must be updated so that M=CM.

PROGRAM DESCRIPTION

The algorithm of Fig. 4.7 performs a fixed depth alpha beta search. The program contains GOTOs because , from the point of view of any one processor, other processors cause discontinuities in the execution path of that processor.

The program is similar to that given for a single processor, except that there are the following additions:

The processors synchronize the beginning and end of their searches

by sending STRTSRCH and ENDSRCH messages. When receiving such a message, the number of active processors is incremented / decremented.

By setting the switch register 1 on the front panel of one machine, one processor knows that it may lead the search initially. The other processor requests the critical area and awaits an acceptance from the other machine, which will be sent when the other machine leaves the critical area.

FINDING A FREE NODE.

When descending a search line which the other processor has already entered, a processor needs to know which line to take from the current updated node, and the current backed up score for that node. It sends REQM to the other machine which replies with a RELIN message containing the necessary node information.

When one processor is following another (LEAD=0) and arrives at LOOP, it is seeking a move which needs to be evaluated. It must first catch up with the other processor. The other may be above, below or split from this processor at this depth (by exiting from MINMXP and evaluating a move privately).

When CM<BASE(D) then the other is to be reached by ascending. Control passes to RESTOR.

Otherwise the other is below or level with this processor. First skip those moves which the other has already evaluated by setting M=CM.

Then if D<CD the other processor is to be reached by descending.

Otherwise D=CD and the processors have a common tree upto this node and will split at this point. The processor selects the next unevaluated move.

The next section:

```
IF M>CM THEN DO;
        CM=M;   SEND(CM);
        END;
```

ensures that CM is always at least as great as M.

SELECTING A MOVE AT THE COMMON DEPTH.

The contents of FROM(M) are tested. When it is zero or dummy, control passes as with one processor - there is no move to be evaluated. Otherwise the move needs to be evaluated. This is done by a call to MINMX2 if the move is outside the critical area, otherwise it is evaluated by a 'recursive' call to MINMXP which is effected by jumping to the entry point of MINMXP, DESCEND.

The other operations nesting the call to MINMX2 yield access to the critical area and then, after return from the call, request and await access to the critical area.

SCORING NODES.

Node values are only backed up when a final score for the node is obtained. Otherwise evaluation of a node is precluded. Thus a node is scored either when all its successors have been scored or an alpha beta cutoff has been found.

If a move has given an alpha beta cutoff, then the score of other moves from this node are irrelevant. Any other processor in this subtree has been literally cutoff by the cutoff. The node may be scored immediately whether or not another processor is in the subtree of the node scored, viz even when NOIN(D+1)>0.

When OSC=NOIN(D+1)=0 a processor is scoring nodes which have been completely evaluated and when OSC=NOIN(D+1)/=0, it

is scoring nodes because all other processors in that node have been cutoff. When OSC/=NOIN(D+1) the node evaluation is incomplete and cannot be backed up.

When a new node score and best line are found these need to be communicated to another processor whenever they are part of the other processor's search tree. This is done by the call

SEND(SELIN,BASE(D),CM,VSCOR(D),NOIN(D),COLOUR,BSTMVS(D,*));

When an alpha beta cutoff occurs which is on a node which is also part of the other processors search tree(CM>BASE(D)), then the other processor needs to be informed. It sends the message

CUTOFF(CM,BASE(D-1))

which informs the other processor how far to back up before it seeks a new line to evaluate.

FIGURE 4.7 PARALLEL SEARCH ALGORITHM.

```
DCL
        ACTPROC -ACTIVE PROCESSORS - ASSUMED 0 ON ENTRY
        PDPTH   -ASSUMED ON ENTRY TO MINMAX
        CA      -CRITICAL AREA SEMAPHORE
        CM /* COMMON MOVE */ INIT(-1) /* HIGHEST ADDRESS */,
        D /*DEPTH*/ INITIAL (0),
        CD,CD0, /* COMMON DEPTH AND INITIAL COMMON DEPTH */
        OSC, /* NO OF OTHER PROCESSORS THIS PROCESSOR
                IS SCORING */
        NOIN(0:MMAXD), /* NO. OF PROCESOORS IN NODE AT DEPTH D */
        CA SEMAPHORE INIT(1);
;
MINMAX:PROCEDURE;
        D=0; CD=CD0;
        UFLAG=0 /* UPDATE FLAG USED IN PRINTING */;
        LEAD=0;
        IF INITIAL_LEAD THEN DO;
                V(CA); LEAD+;
                END;
        P(CA)   ;/* REQUEST CA AND AWAIT TILL OBTAINED*/
        GOTO START;
DESCEND:
        /* D<CD */
        CALL UPDATE;
```

```
START:
        NOIN(D)=1;
        CALL SETMOVES;
        IF ^LEAD THEN DO;
                SEND(REQM,BASE(D));
                AWAIT REPLY(RELINE,CM,CD,BASE(D),NOIN(D),VSCOR(D),BSTMVS(D,*
                /* OTHER PROCESSOR INCS NOIN(D) & REPLIES NOIN(D) */
                CALL NEWSC; /* FIND IF CUTOFF CAUSED BY REPLY
                               AND PASS NEWSCORE DOWN MOVE STACK
                               */
                M=CM;
                END;
        UFLAG+;
        IF D<PDPTH & D^=0 THEN PRINT_UPDTD_MOVE;
;
;
LOOP:
        OSC=0;/* SCORING FOR NOONE ELSE */
        IF ^LEAD THEN DO;
                IF CM>BASE(D) THEN GOTO BREAK;  /*ABOVE*/
                M=CM;
                IF FROM(M)=0 THEN GOTO BREAK;   /* D=0 */
                IF D<CD THEN GOTO DESCEND;      /*BELOW */
                M+;
                LEAD+;  /*LEVEL */
                END;
        /*M POINTS TO AN UNEVALUATED NODE */
        IF M>CM THEN DO;
                CM=M;
                SEND(CM);       /*OTHER PROCESSOR SETS LEAD=0*/
                END;
        /* LEAD */
        IF FROM(M)=0 THEN GOTO BREAK;
        ELSE IF FROM(M)=DUMMY THEN DO;
                M+;
                VSCOR(D)=SCOR(D);
                END;
        ELSE DO;
                IF D<CD THEN GOTO DESCEND;
                PRX2=0 ;ATTENT=0 ;
                V(CA);  /* FREE CRITICAL AREA */
                CALL MINMX2;
                /* WHEN RETURN IS DUE TO RECEIVED CUTOFF,
                 * ON RETURN, LEAD=0 & OSC=-1
                 */
                P(CA); /* OBTAIN CRITICAL AREA */
                NOIN(D+1)=0;
                IF ATTENT THEN CALL NEWSC;
                /* PRX2=0 */
                END;
;
;
;
RETCALL:
        IF OSC=NOIN(D+1) THEN DO;
                /* FINAL SCORE FOR MOVE TO D+1 */
           IF BLACK THEN DO;
                IF VSCOR(D+1)<VSCOR(D) THEN DO;
```

```
                        VSCOR(D)=VSCOR(D+1);
                        IF VSCOR(D)<=VSCOR(D-1) THEN GOTO CUTOFF;
                        IF  FROM(M)^=DUMMY THEN DO;
                                BSTMVS(D,*)=UPDTD_MOVE(D)&BSTMVS(D+1,*);
                                IF CM<BASE(D) THEN
                                SEND(SELINE,BASE(D),NOIN(D),VSCOR(D),BSTMVS(D,*)
                                END;
                        END;
                END;
           ELSE DO;
                IF VSCOR(D+1)>VSCOR(D) THEN DO;
                        VSCOR(D)=VSCOR(D+1);
                        IF VSCOR(D)<=VSCOR(D-1) THEN GOTO CUTOFF;
                        IF  FROM(M)^=DUMMY THEN DO;
                                BSTMVS(D,*)=UPDTD_MOVE(D)&BSTMVS(D+1,*);
                                IF  CM<BASE(D) THEN
                                SEND(SELINE,BASE(D),NOIN(D),VSCOR(D),BSTMVS(D,*)
                                END;
                        END;
                END;
           END;
        GOTO LOOP;
;
;
;
CUTOFF:
        -NOIN(D);
        IF NOIN(D)>0 /* OTHERS TO NOTIFY */ THEN DO;
                /* SCORED NODE WHICH CAUSED CUTOFF IS COMMON TO BOTH
                 * PROCESSORS
                 */
                OSC=NOIN(D); /* SCORING FOR ALL OTHERS */
                SEND(CUTOFF,BASE(D-1));/*OTHER PROC. RESTORES AND REQUESTS CA*/
                LEAD+;
                END;
        GOTO CONTIN;
BREAK:
        -NOIN(D);
        IF NOIN(D)>0 THEN SEND(EXIT,BASE(D)); /* OTHERS DECREMENT NOIN(D) */
CONTIN:
        IF D=0 THEN GOTO EXIT;
        IF (D<PDPTH &UFLAG^=0 ) OR D=PDPTH THEN DO;
                CALL PRINT_VSCOR;
                UFLAG=0;
                END;
        CALL RESTOR;
        IF D<CD & VARCD THEN DO;
                CD=D; IF D=0 THEN CD=CD0;
                END;
        GOTO RETCALL;
EXIT:
        V(CA);
        AWAIT(NOIN(0)=0);
        CA=0;   /*CLEAR CA IN CASE OTHER PROC. REQUESTS LEAD BEFORE*/
                /* INITIALISATION OF NEXT SEARCH */
        CM=-1;
        RETURN;
END MINMAX;
```

Chapter 5. RESULTS AND DISCUSSION.

This chapter presents the results found and their interpretation. I begin by defining what I mean by overhead, and then distinguish the various types of overhead which can occur.

The analysis used for separating one particular type of overhead, the communication overhead, from the others is then presented. This then leads into the presentation of the results, and is followed by an explanation of the program's printed output. The chapter closes with a discussion of several features of the results.

POWER OF COMBINED PROCESSORS.

Two or more processors may be combined to form a single virtual processor. When the individual processors are combined perfectly, the power of this virtual processor is the sum of the powers of the individual processors. Thus a PDP 11/45 which executes at 1 Mhz and a PDP 11/20 which executes at 0.5 Mhz have a potential combined power of 1.5 Mhz.

Suppose two processors perform a task in times T1 and T2 respectively, when executing singly, and in time T0 when combined. Expressed as times, the maximum power of the combined processor, 1/T0, is the sum of the powers 1/T1 and 1/T2, of the individual processors.

$$\frac{1}{T0} = \frac{1}{T1} + \frac{1}{T2} \qquad (1)$$

I call this minimum time T0 the base time, and compute efficiencies of combined machines by comparing them to the base time.

All time in excess of the base time is overhead.

OVERHEADS - TYPES

Overheads may be caused by:

1 Idle time at the end of a search when a processor is waiting for another to complete its task.

2 Idle time during the search when a processor has completed its task and is waiting to enter the critical area to score its task and select another move (processor lockout).

3 Idle time during a search when a processor wishes to enter the critical area and update its scores to reflect new values received from the other processor (processor lockout).

4 Communication of scores, lines of play and tree status.

The followings overheads are due to extra search effort.

5 Unneccesary full or partial evaluation of a move which will be cutoff by another processor.

6 Move evaluation with unneccesarily large alpha beta search width. Note that for CD=0, the processors cannot cut each other off since the root node is always an AND node. Furthermore, beta values of moves from the root are independent. Only alpha values for the score at D=0 connect the searches.

7 Duplication of the root area - each machine generates the moves in the root area. This is generally negligible since the fanout is so high.

I have not attempted to measure each of these overheads separately, but have instead divided them into two general categories, which are:

1 dsub - the increase in weighted tree size due to unnecessary full or partial evaluation of moves and duplication of the root

area of the tree.

2 dcomm - time spent forming,sending,receiving and processing the messages passed between the two machines.

My calculations separate the effect of these two types of overhead.

Unfortunately, the communication overhead is not easily measured since communication time is dispered over many subroutines. Timing each of these subroutines is too involved.

The time spent communicating is proportional to the number of words sent and received, but this does not give an absolute measure of the time spent communicating. To find communication time, it is necessary to measure the time spent searching the tree, and to call an additional search time communication overhead.

The time spent actually searching the tree needs to be measured. This time is not proportional to any simple measure of the tree size, such as the number of bottom positions in the tree or the number of board updates, as I have found from experience. The only way to measure the time spent searching the tree is to find the amount of time spent executing the subroutines which are called during a tree search. Any other methods are unreliable.

I now present these ideas mathematically, and describe how the ensuing calculations have been programmed.

MEASUREMENT OF SEARCH EFFORT.

The total search time is the weighted sum of the number of calls to each subroutine. There are five principal categories of subroutine calls during a search. These are :

		# calls	Average time of one call
1	Update and restore	U	u
2	Setmvs-move generation	S	s
3	Cstmvs-capturing moves generation.	C	c
4	Minmax-tree control and scoring.	M	m
5	Communication subroutines	W	w per word

The time u is the average time for a board update and restore. For every board update there is a corresponding call to restore.

In a parallel search, one processor spends its time as follows:

$$T=U*u1 \quad +S*s1 \quad +C*c1 \quad +M*m1 \quad +W*w1 \qquad (2)$$

where $W=W_{in} \quad +W_{out}$ is the sum of the words received and sent.

This time is for processor 1. Combined processors will take less time. To convert subroutine timings to that for the combined processor according to (1):

$$u = u1 * T0/T1 \qquad s = s1 * T0/T1 \qquad (3)$$

$$c = c1 * T0/T1 \qquad m = m1 * T0/T1$$

The effort for a distributed parallel search, where the subscripts denote the processor I.D., is:

$$T = (U1 + U2)u +(S1 +S2)s +(C1+C2)c + (M1+M2)m + (W1+W2)w \qquad (4)$$

The communication overhead is (W1+W2)w.

For any board position, u,s and c have been measured by timing a fixed number of calls to each subroutine. I note that during any search s and c will decrease as pieces are captured, so that s and c as measured from the root of the tree will be greater than s and c as measured several plies from the root.

It is only possible to measure m in an actual search. The time for processor 1 to search alone is:

T1= U*u1 +S*s1 + C*c1 + M*m1 (5)

where u1, s1, c1, and m1 are subroutine times as measured by processor 1.

This may be solved for m by setting M=U i.e. by assuming that the amount of work involved in scoring, copying the principal continuation up the tree and selecting moves is proportional to the number of board updates. This gives:

T1 = U*u1 + S*s1 + C*c1 + U*m1

The combined minimum search time is then

T0 = U*u + S*s + C*c + M*m (6)

In practise the number of subroutine calls is increased for a parallel search. Let subscripts denote the number of subroutine calls for each processor during a parallel search.

Then the subroutine time is:

Tsub = (U1 +U2)u + (S1+S2)s +(C1+C2)c + (M1+M2)m (7)

The communication time is defined to be any time in excess of Tsub.

T = Tsub + Tcom (8)

This may be solved for Tcom.

RESULTS

Five endgame positions from Fine [9], Fig 5.1 have been searched mainly with depths MAXD=8 and MAXD=9. Five middlegame positions from Point Count Chess, figure 5.2, by Horowitz and Mott-Smith [15], have been analysed at MAXD=4 and MAXD=5. The maximum depth of the quiescent search, MMAXD, is MAXD+2. The program printout for these position is given in Appendix A

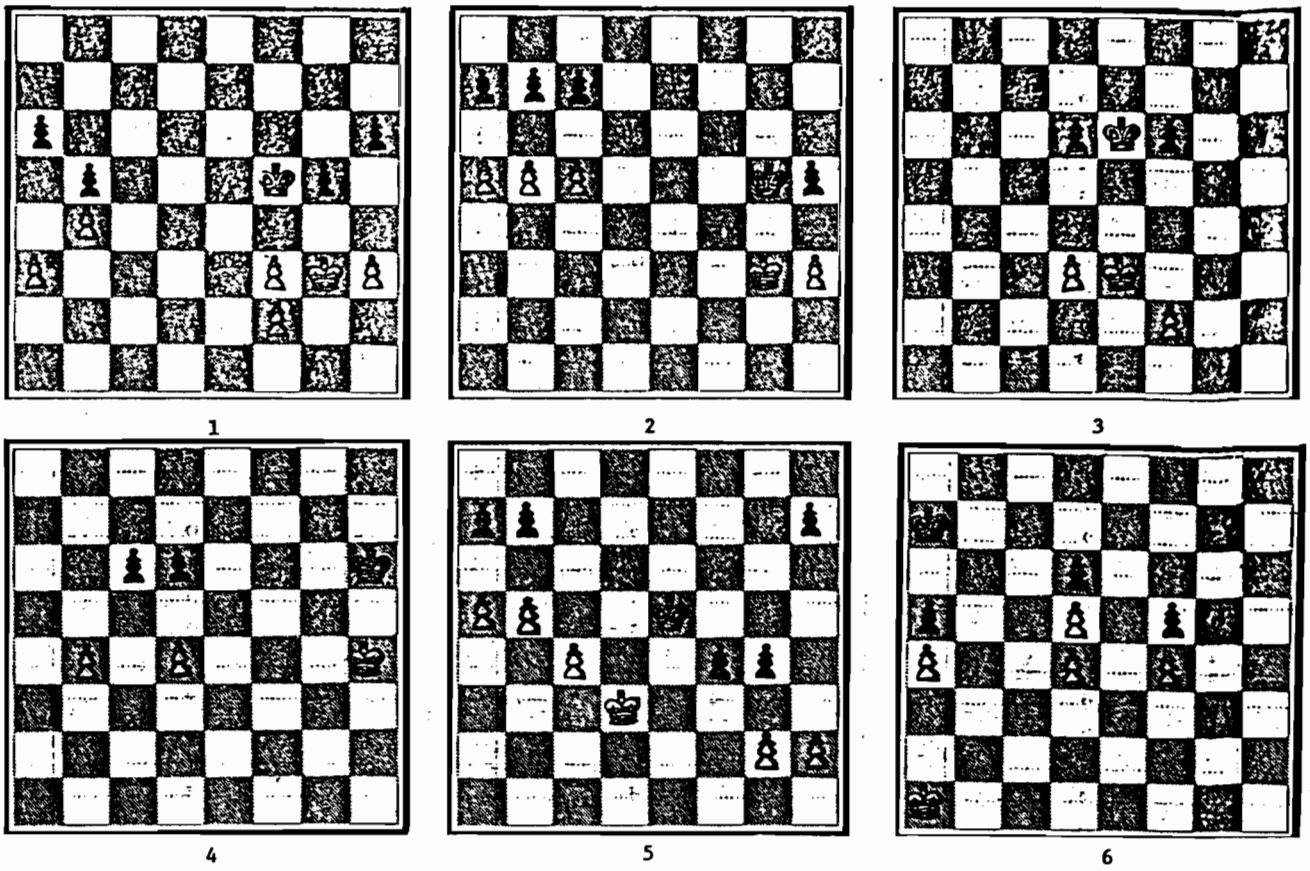
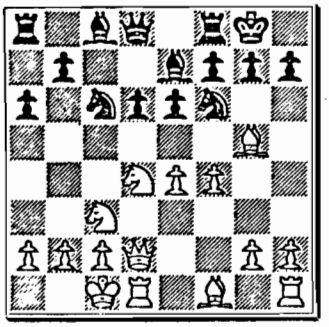


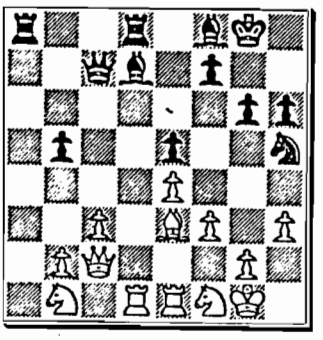
Figure 5.1 Endgame positons which were searched. White is to move. Program printout is found in Appendix A.



Keres v. Szabo 1955

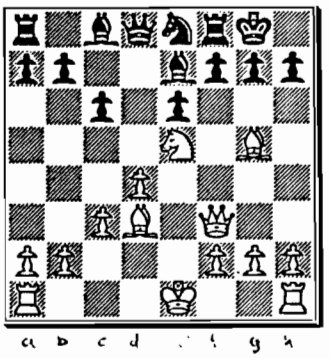
NO. 31. *White to move*

Dus-Chotimirsky v. Capablanca, 1925

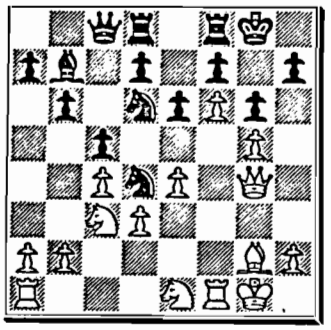


NO. 36. *Black to move*

Capablanca v. Blanco 1913

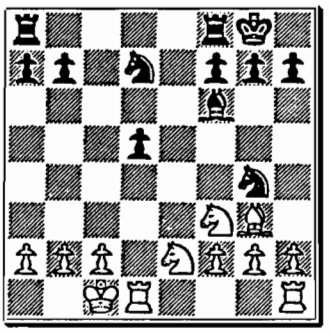


NO. 55A. *White to move*



5	B – B4	Q – B1
6	P – B6	B – Q3
7	BxB	NxB
8	Q – N4 and wins	

NO. 48B. *Black to move*



5	B – R4	B – Q1
6	B – N3	B – B3
7	N – KB3	

NO. 72B. *Black to move*

Figure 5.2 Middle game positions which were searched.

and explained below.

PRINTED OUTPUT - SINGLE PROCESSOR SEARCHES

A single processor search is conducted for each processor separately. The counters

U , S , C , M , T , NOMAXD(the # positions at the maximum depth)

and NOMMAXD (# positions at absolute maximum depth of search)

are printed for both processors.

The line 'BASE' contains the base values used for calculating increases in a parallel search. The base time is found by (1).

The followings lines give the average time of each subroutine call, the total time spent in each subroutine and the % of processor time spent in the respective subroutines.

The results show:

1 The quiescent search, which investigates capturing sequences, takes a large proportion of program time.

2. Move generation typically takes 70% of all search time.

PRINTED OUTPUT - PARALLEL SEARCH.

The calculations shown in the printed output for parallel searches are detailed below.

% increase of subroutine time on base time: $\frac{Tsub-T0 * 100}{T0}$

% increase of communication time on base time: $\frac{Tcomm-T0 * 100}{T0}$

% increase of time on base time: $\frac{T-T0 * 100}{T0}$

Microsecs per word communication time: $\frac{T-Tsub}{Win + Wout}$

Efficiency ; % of base time in total time: $\frac{T0 * 100}{T}$

Efficiency : % of increased subroutine time in total time: $\frac{Tsub-T0 * 100}{T}$

Efficiency : % communication time in total: $\frac{Tcomm * 100}{T}$

MINIMUM SEARCH TIMES - % OVERHEAD AND PARAMETERS FOR MINIMUM

The following table is abstracted from the results given in Appendix A. For the fastest search in each position at each depth, it shows VARCD, CD, the per centage overheads due to increased tree size and communication, and the overall overhead.

Table 5.1 Time overheads for minimum search time
for all positions studied.

position	maxd	varcd	cd	dsub %	dcomm %	% increase of time on base
end game						
1	8	1	2	5	5	10
1	9	1	2	7	2	9
2	9	1	2	12	2	14
2	10	1	2	0	7	9
3	8	0	1	4	1	5
3	9	0	1	7	1	8
4	8	0	1	26	1	27
4	9	0	1	28	0	28
5	7	1	2	7	9	17
5	8	1	2	6	3	9
middle game						
31	4	0	0	1	5	7
31	5	0	0	8	2	10
36	4	0	0	1	1	2
36	5	0	0	9	0	9
48b	4	0	0	13	5	18
48b	5	0	1	4	2	5
55a	4	0	0	0	1	1
55a	5	0	0	-3	1	-3
72b	4	0	0	1	5	5
72b	5	0	0	20	0	20

Most positions studied may be searched with a time overhead of 10% or less. Thus an increase in processor power of 1.5 gives an average speedup of about 1.35.

I will now attempt to distinguish the effects of particular types of overhead.

I attribute the less than expected search time for position 55A to a leapfrog effect whereby the 11/45 evaluates a good move earlier in the search when parallel processing than when searching the same position alone. The score for this good

move then decreases the search effort for the leapfrogged move(s) which the slower 11/20 is processing. This is the only way in which the move ordering may have been altered for the parallel search.

Some of these times are inflated by idle time for one processor at the end of the search. In practise this overhead is only significant for CD=0. It may cause a processor to idle for as much as 40% of the search time (position 4 with MAXD=5, CD=2). For higher values of CD this factor is less than 1 % of the search time.

The following table excludes this idle time and gives time increases for all positions for CD=0 (not necessarily the best time). It may be seen that DSUB generally increases with D.

Table 5.2.
DSUB (% time increase on base time excluding communication time and busy loop time) at CD=0 for all positions studied.

	Endgame				
Position	1	2	3	4	5
d=7					20
d=8	16		4	13	11
d=9	24	6	7	6	
d=10		21			

	Middle game.				
Position	31	36	48b	55a	72b
d=4	1	1	13	0	1
d=5	8	9	14	-3	20

Do new scores found by one processor speed up the other? The program was run with and without the passing of new scores found by one processor into the tree of the other. For the few positions studied, the difference in the time of search

was insignificant. It appears that these new narrower alpha-beta values come too late in the search to cause any appreciable difference to the search time.

OVERHEADS - RELATIVE MAGNITUDES OF COMBINED TREE SIZE AND COMMUNICATIO

In Table 2.1, the communication overhead is generally about of the same magnitude as subroutine overhead. It increases exponentially with CD and makes values of CD>=2 inefficient. In no position is there a minimum search time with CD=3 or 4.

Part of this communication overhead must be due to the simple nature of the link used - all transfers done under CPU control without DMA, a word at a time. I have not been able to isolate processor lockout (time when a machine is waiting to enter the critical area) from the communication time.

INCREASE IN COMMUNICATION OVERHEAD WITH CD.

The average factor by which the communication overhead increases for each unit increase in CD is shown below.

	Middle Game	End Game
Strategy A: variable common depth	2	1.5
Strategy B: fixed common depth	5	3

The higher rate of increase in communication overhead for the middle game is probably due to the higher fanout in these positions.

Strategy A has a lower rate of increase in communication overhead because the common depth averages over time to a low value - it is only high on entry to a node, and is soon decreased.

VARIATION OF DSUB WITH CD.

Dsub is generally lower for even CD (AND nodes) than for odd CD (OR nodes). Aside from this, as CD increases, dsub alters unpredictably.

This is explained by noting that once a principal continuation has been found, a search tree will generally consist of alternating AND then OR nodes. The root node is always an AND node.

All successors of AND nodes need to be evaluated. Only one successor of an OR node may need to be evaluated. When several processors are assigned to evaluating the successors of an AND node, none of them will be cutoff by another; conversely, when several are assigned to an OR node, one may cut off the others; the effort of the cutoff processor is wasted.

COMPARISON OF STRATEGIES A AND B.

The strategy of a variable common depth generally produces lower values of dsub and dcomm than strategy B.

Strategy A is most effective in endgame positions, wherein overhead is generally low for CD=2.

COMPARISON OF PARALLEL SEARCHES FOR MIDDLE GAME AND ENDGAME.

For middle game positions the communication overhead increases so rapidly that high values of CD are inefficient. Fortunately, when CD = 0, dsub is generally low. I attribute this to a bushy tree effect - alpha beta values are fairly constant as the many moves from the root are evaluated, so the successors of the root may be evaluated independently. Most of the program time is spent refuting root moves and not establishing them.

By contrast, endgame positions have a lower fanout and so

a greater proportion of the time is spent in finding the principal continuation and fixing alpha beta values. A greater fraction of the search effort is then spent searching with narrowing alpha beta values and dsub is consequently greater.

SUMMARY.

The two processors combine perfectly when performing a parallel search, except for the following overheads:

increased combined tree size.

end of search idle time

communicatsion time

processor lockout.

For most search positions, ther exists a set of search parameters which produce a total overhead of 10 % or less.

Chapter 6. EXTENSION TO N-PROCESSORS.

The failure of multiprocessors to provide significant speed-up has been attributed to the inadequacy of known control structures to co-ordinate the activities of many processors (Enslow[7]). Enslow observes that throughput in a multiprocessor system is a nonlinear function of the number of processors. Throughput increases by 1.8 for 2 processors, but by only 2.1 for three.

Some have attributed this failure to the fact that many algorithms are simply not suitable for parallel processing. However tree search is highly suited - the number of nodes in the tree and thus the number of tasks available for parallel execution grows exponentially with depth. Each of these nodes requires but little communication compared to the amount of computation involved.

The central design question for N-processor tree searches is the control structure - should there be a central control process or should all processes have independent access to the root area of the tree?

A central control process has the disadvantage that it is itself a potential execution bottleneck. Shared access to the root area of the tree eliminates this bottleneck and moreover eliminates the need for most inter-processor messages. Interference between processors, which may become significant for many processors in a small root area, may be reduced by allowing access to different nodes at the same time.

I begin with a few comments on storage allocation.

STORAGE ALLOCATION FOR A DYNAMICALLY CHANGING TREE

The block structure of the program of figure 2.1 indicates that new storage for placing generated moves is allocated in the prologue of each call to MINMAX, and freed again in the epilogue. This storage use has been implemented as a stack.

However, with many processors, the fact that one processor leaves a block does not imply that the storage declared in that block may be freed; other processors may still be inside this block. Storage is allocated and freed at unpredictable times and cannot be implemented as a stack.

Do any high level languages provide any dynamic storage allocation primitives suitable for this task ? A language such as PL1 assumes that dynamic storage allocation may be implemented as a stack. The programmer is expected not to create holes in the stack by freeing storage out of sequence. PL1 does not, then, provide any storage allocation primitives suitable for a dynamically changing tree. In contrast, Algol 68 uses a heap storage allocation scheme whereby storage may be allocated and freed at random. It provides primitives suitable for this problem.

In the following, I make explicit when storage may be allocated and freed, and also when processes are created and die.

N-PROCESSOR SEARCH WITH CENTRAL CONTROLLING PROCESSES.

The central problem is the coordination of

1 creation of tasks.

2 the evaluation of their results.

3 killing tasks which have been made redundant by the results of another task (cutoffs).

4 the freeing of storage no longer required.

To examine its feasibility , I have written an outline control system which separates these components into subroutines which may be executed by a central processor. Figure 6.1 details the analysis of this system.

The subroutine SPAWN generates the root part of the search tree. It allocates storage for nodes, and creates processes to evaluate them. Processors are assigned to nodes according to the scheme detailed in Chapter 4. The processes created pass the resulting evaluations to the process ASSESS via the pipeline V and then die.

The process ACCESS evaluates the scores it receives along the pipeline and passes maximal scores and lines of play up the tree.

When a node evaluation is completed, either naturally or by a cutoff, it calls TERMINATE, which kills all processes in the subtrees of cutoff nodes and frees the corresponding storage. The process SPAWN uses the system calls:

```
ALLOCATE(UNODE,U)
CREATE(ID,MINMX2(BOARD,ALPHA,BETA,U,M))
```

The former allocates storage. The latter creates a process to execute the task MINMX2 with the given parameters. The call returns the identification ID of the created process.

These calls become blocked either when all storage is allocated, or when all processors are allocated.

ASSESS is the receiver process for node evaluations. These are received on the pipeline V by the system call

```
RECEIVE(V)
```

When the pipeline is empty, receive is blocked. When data is placed in the pipeline, RECEIVE becomes unblocked and ASSESS

continues.

SPAWN and ASSESS are mutually exclusive processes. I presume that when TERMINATE kills processes and frees storage, SPAWN will not be reactivated until ASSESS is again blocked by an empty pipeline.

Figure 6.1. Tree Search with Central Processes.

```
MP:PROC OPTIONS(MAIN);
DCL 1 UMOVE BASED(U),
        2 LSTU PTR,/* BACK POINTER TO PRECEDING NODE */
        2 SCORE,
        2 VSCOR,/* BACKED UP SCORE */
        2 BSTMVS(0:20) LIKE MOVE,
        2 #SUC, /* # SUCESSORS */
        2 MOVESU(0:50),
                3 MOVEU LIKE MOVE,
                3 NXTU PTR,
                3 SUCID; /* ID OF PROCESSOR ASSIGNED TO MOVE */
    CD, /* COMMON DEPTH
    CDO; /* INITIAL COMMON DEPTH */

/* ASSUME THIS PROCESS HAS ID=1 */


CD=CDO;
ALLOCATE UMOVE;CALL IUMOVE;/*INITIALISE UMOVE */
CREATE(J,ASSESS);/*CREATE PROCESS #1 TO EVALUATE NODES */
CALL SPAWNO;

SPAWN:PROC(U,M) RECURSIVE, EXCLUSIVE OF ASSESS;
/* THIS PROCESS CREATES A SEARCH TREE DOWN TO THE COMMON
 DEPTH, CD. THE CALL CREATE IS PRESUMED TO BECOME BLOCKED
 WHEN THERE ARE NO MORE PROCESSORS AVAILABLE.
 CREATE BECOMES UNBLOCKED WHEN AN INVOCATION OF ASSESS
 HAS COMPLETED AND FREED SOME PROCESSOR(S).
 */
DCL
        U PTR,
        M,
        ID;
        UPDATE;
SPAWNO:ENTRY;
        ALLOCATE UMOVE;
        SETMVS;
        M=0;
```

```
        WHILE(FROM(M)/=0) DO;
                IF D<CD THEN CALL SPAWN;
                ELSE IF #SUC>0 /* =-1 FOR CUTOFF NODE */ THEN
                        DO;
                        CREATE(ID,MINMX2(BOARD,A,B,U,M));
                        U->MOVE(M).NEXT=NULL;
                        U->MOVE(M).SUCID=ID;
                        END;
                END;
        RESTORE;
        IF VARCD THEN DO;
                CD=CD-1;
                IF CD=0 THEN CD=CDO;
                END;
END SPAWN;


ASSESS:PROC EXCLUSIVE OF SPAWN;
/* THIS PROCESS BEGINS AND IMMEDIATELY BECOMES BLOCKED
 * SINCE THERE ARE NO EVALUATIONS TO RECEIVE
 * IT BECOMES UNBLOCKED WHENEVER AN EVALUATION IS RECEIVED
 */
DCL
        V PIPELINE OF (U,V,EVAL,LINE)
        U PTR,
        M,
        EVAL,/*RETURNED SCORE */
        LINE(0:20) LIKE MOVE;
RECEIVE(V:U,M,EVAL,LINE)
WHILE(#SUC(M)>0) DO;
        CALL TERMIN(U,M);
        IF EVAL>=VSCOR THEN DO;
                VSCOR=EVAL;
                UO=LSTU;
                IF VSCOR>=UO->VSCOR THEN DO;
                        /* CUTOFF */
                        #SUC(M)=1;
                        END;
                ELSE BSTMVS=MOVE(M) AND LINE;
                END;
        #SUC(M)=#SUC(M)-1;
        END;

TERMIN:PROC(UO,MO) RECURSIVE;
/* KILL ALL PROCESSES IN ALL SUBTREES OF UO,MO
        AND FREE STORE WHICH HAS NO PROCESSORS ROOTED IN IT
 */
DCL UO PTR,M;

IF UO=NULL THEN RETURN;
U=UO->MOVE(M).NXTU
WHILE(MOVE(M).SUCID/=0) DO;
        IF SUCID=1 THEN CALL TERMIN(U,M);/*KILL SUBTREE*/
        ELSE IF SUCID(M)>1 THEN CALL KILL(SUCID(M));/*KILL PROCESS*/
        END;
FREE(U->UMOVE);
UO->UMOVE(MO).NXTU=NULL;
END TERMIN;
```

The declaration contains a storage allocation for the successors of the node, unlike the case heretofore.

The storage area for an updated move has to be allocated and freed dynamically. A processor leaving a node decrements the #SUC count for that node. When it decrements #SUC to zero, then the tree pointed to by base may be freed. The processor should then set #SUC to -1 to indicate that the node has been scored.

N-PROCESSOR SEARCH WITHOUT CENTRAL CONTROL PROCESS.

Assume there are many processors sharing some memory area, and that it contains the tree root area.

I will discuss the case in which each processor may traverse the tree independently of a central controlling process. I will identify each operation for which interference may arise. Each such operation will have a separate semaphore lock.

The condition that the entire root area be a single critical area, so that only one processor may access it at any one time, may be relaxed to allow several processors to traverse the same tree simultaneously. Interference between processors only arises when

1 Scoring nodes & Copying up the principal continuation.

2 Selecting the next move to evaluate from that node.

Let these two types of operation be a separate critical area for each node. Let each be gated by its own semaphore, with initial values one. Then a maximum of two processors may be visiting a node at any one time; a processsor will visit a node from above or below when seeking a node to evaluate and from below when scoring a node.

Consider the following example, wherein N represents nodes and P processors. Suppose processor P1 to be searching for a node to be evaluated. It is not scoring node N. It increments #PROC(N,N1) and enters node N1 simultaneously with processor P2, which is scoring node N1.

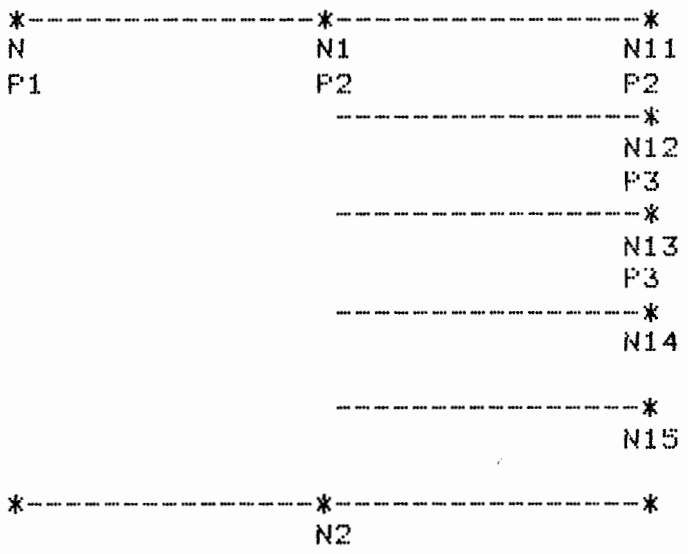


Figure 6.2. A tree in which several processors P are visiting nodes N.

Processor P2 has evaluated N11 and is scoring N1, which requires that it control the evaluation part of node N1, where the score and best line from node N11 are stored. Processors P3 and P4 are scoring nodes N12 and N22.

Processor P1 is about to enter node N1 and score one of its successors. It will either search N14, which is still unevaluated, or else, if P2 reports a cutoff, it will notice this when it visits N1, exit from N1 and try to score a successor to N2 instead.

The declaration of an updated move might be:

```
DCL
        1 UMOVE(*:*),
                2 LAST_BASE PTR /* BACK POINTER UP TREE */,
                2 EVAL_N,
                        3 LOCK SEMAPHOR INIT(1),
                        3 VSCOR,
                        3 BSTMVS(0:MMAXD) LIKE MOVE,
                2 NEXT_MOVE,
                        3 LOCK SEMAPHOR INIT(1),
                        3 M PTR, /* POINTER TO NEXT MOVE TO EVALUATE*.
                2 CUTOFF FLAG INIT(0),
                2 PROCIDS(0:#PROCESSORS) /* IDS OF PROCESSORS IN TREE*
                2 MOVES(0:50),
                        3 TYPE,
                        3 TO PTR,
                        3 FROM PTR,
                        3 #PROC, /* NO OF PROCESSORS IN THIS TREE*/
                        3 BASE PTR /* BASE ADDRESS OF MOVE */
```

Cutoffs must be quickly communicated to other processors. This may be done either by having a pipeline between processors or by periodically testing the cutoff flag.

The program for directing this tree search would be very similar to the one given in Chapter 4. The only parts which would need to be changed are those involving the variables LEAD and CM. With shared memory, a processor will be able to find its own way around a tree, without having to ask any others where they are in the tree, what their score and best line are, or which moves have or have not been evaluated.

The variable LEAD is not required since it only helps find a free move. Instead a processor will traverse the tree (when D<=CD) until it finds a free move - one with #PROC=0.

ESTIMATED SPEED OF MANY PROCESSORS

It is necessary to have at least N nodes available for parallel execution. When there are more processors than moves from the common depth, it will be necessary to increase CD.

Results with CD>0 have shown that the search effort is not significantly increased when many nodes are made available. The optimum value of CD will be the one which balances the narrowed alpha beta values which the processors will inter-communicate against the higher 'communication' (root area traversal) overhead. I cannot give an accurate prediction for the speedup with many processors.

Chapter 7. CONCLUSION.

A compact form of the algorithm used in searching game trees was presented first. This is the basis of the program which has been written to search game trees and play chess. On a PDP11/45 with a cycle time of 1 microsec., the program makes a full width depth first search of any chess position at the rate of up to 2000 positions per second.

An algorithm for parallel search of chess game trees has been devised, and implemented in the case of two distributed processors communicating via a simple data link. The existence of transmission errors has required that protocols be devised and implemented for the detection of, and recovery from, line errors.

It has been found that two computers may be combined to search a game tree, and that the average time overhead is only 10%. Further analysis of this overhead has required that the times spent communicating and searching the tree be measured and distinguished.

A standard method for measuring the size of the search tree, by counting the number of nodes at the bottom of the tree, was found to be an inadequate measure of the time spent searching the tree. A more accurate method is to total the times spent in each subroutine. The time spent in any subroutine is the product of the number of calls to the subroutine and the average time of each call.

A program has been written to find these average times, to conduct tree searches with varying search parameters, to calculate how much of the search time was spent in each subroutine,

and to thereby deduce the communication overhead.

By accounting for where the processor spends its time during the search when parallel processing, insight has been gained as to the relative sizes of the principal overheads in a parallel search. The main overheads are:

end of search idle time,
communication time
and increased combined tree size.

It was found that the end of search idle time is only significant when the common depth is zero, and that for the fastest search in any position, the other two overheads are about equal. Communication overhead increased exponentially with the common depth of the search tree, so that when the common depth is greater than two, the search is always relatively inefficient.

The final chapter considered methods for implementing a parallel search when many processors are available. Two methods were compared:

1. A central processor which has unique access to the root area of the tree, and which controls searches by the other processors, and

2. Independent processors, which have shared access to the root area of the tree.

The latter configuration was then recommended as being the more promising.

BIBLIOGRAPHY

[1]
Abrahams, G.
The Chess Mind
Pensuin 1964

[2]
Adelson-Velskiy,G.M.,'Arlazarov,V.L.,Bitman,A.R.,
Jivotovskiy,A.A., and Uskov, A.V.
Prosrammirovannii Isry Vychislitelnoy Mashiny v Shamaty
Usрehi Mat, Nauk, Vol. 25, No. 2, !970.

[3]
Chans, S.
'Parallel Balancins of Binary Search Trees'
IEEE Transactions on Computers, Vol. 23, p 441, 1974

[4]
DEC
Disital PDP11 Periрherals Handbook
Disital Equiрment Corрoration 1975.

[5]
Dijkstra, E. J.
'Solution of a Problem in Concurrent Prosram Control'
Comm. A.C.M., Vol. 8, 1965, рр 569

[6]
Dijkstra, E.J.
'Cooрeratins Sequential Processes'
in 'Prosrammins Lansuases', F. Genuys (ed.)
Academic Press, New York, 1968,рр. 43

[7]
Enslow P.
'Multiрrocessor Architecture, a Survey'
Procedinss 1975, Sasamore Computer Conference on Parallel Processins,
Aus. 1973

[8]
Evans, D. J.
'Construction of Balanced Binary Trees for Parallel Processins'
Computer Journal, Vol 20, p378, 1977

[9]
Fine R.
Basic Chess Endinss
1941

[10]
Frey R. (ed)
Chess Skill in Man and Machine
Sрrinser-Verlas, N.Y., 1977

[11]
Frey, P.
'An introduction to Computer Chess'
Chess Skill in Man and Machine:
in Frey(1977),pp 54-81

[12]
Gillosly, J. J.
'The Technolosy Chess Prosram'
Artificial Intellisence #3, 1972, pp 145-164

[13]
Gray D.
'Line Control Procedures'
Proc., IEEE Nov 1972

[14]
Hansen Per Brinch
'Concurrent Prosrammins Concepts'
Computer Surveys, Vol. 5, 1978, pp223-245.

[15]
Horowitz I. A. and Mott-Smith G.
Point Count Chess
David McKay Company, Inc., New York

[16]
Knuth, D.E.
The Art of Computer Prosrammins, Vol 1, Fundamental Alsorithms
Addison-Wesley 1973.

[17]
Knuth, D.E. and Moore, R.
'An analysis of Alpha Beta prunins'
Artificial Intellisence, #6, 1975, pp. 293-326

[18]
Marshall D.D.
'The solution of 0-1 Prosrammins Problems:
A parallel Processins Approach'
Ph.D. Thesis, Univ. of Alabama in Huntsville, Alabama, 1977

[19]
Marshall,D.D.
'A Parallel Processor Approach to Searchins Decision Trees.'
Proceedinss 1977 International Conference on Parallel Processins,1978
pp 199-201.

[20]
Michie, D.
'On Machine Intellisence'
Edinbursh Univ. Press 1974.

[21]
Newborn, M.
'PEASANT: An endsame prosram for Kinss and Pawns'
in 'Chess Skill in Man and Machine', Frey(ed)

[22]
Nilsson, N. J.
Problem solving Methods in Artificial Intelligence.
McGraw Hill, 1971

[23]
Shannon, C. E.
'Programming Digital computers for Playing Chess'
Philosophical Magazine, Vol. 41, 1950, pp 256-275

[24]
Slate D. J. and Atkin L. R.
'Chess 4.5 - The Northwestern University Chess Program'
in 'Chess Skills in Man and Machine', Frey (ed), 1977 pp 82 - 118

[25]
Wong, C. K.
'Parallel Generation of Binary Search Trees'
IEEE Transactions on Computers, Vol. 23, p 268, 1974

Appendix A -

This appendix contains program output, the general description of which is found in Chapter 5.

line 1 - The command typed in is 'PDEP 0 SWEEP'
which requests that the print depth be set to zero
(i.e. no moves be printed whilst searching)
and that the subroutine SWEEP be executed.

lines 2 and 3 - The subroutine SWEEP begins by printing
the pieces on the board in Continental Notation.

line 4 - Search parameters are printed: ALPHA, BETA, MAXIMUM
DEPTH, ABSOLUTE MAXIMUM DEPTH.

line 6 - SWEEP commands that each processor search the position
on its own.

line 8 - The number of calls to subroutines UPDATE, SETMVS,
CAPTURING_SET_MOVES, and SCORES are printed, followed
by the search time in milliseconds and then the number
of nodes at the maximum depth of search.
This data is printed both for searches made by 'this',
the printing processor, and searches made by the
'other' processor.

line 12 - BASE# combines the data for the two processors.
TIME-MS gives the search time for perfectly combined
processors.

line 13 - The time of 1 call to each subroutine is shown.

line 14 - The total time spent in each subroutine is shown.

line 15 - The percentage of the total time spent in each
subroutine is also shown.

line 17 - Parallel searches are then conducted with VARCDEPTH=0,1;.
The columns printed show the common depth; the % increase
in time spent for subroutine calls over that predicted by
the BASE time; the % increase in time spent communicating
and processing each word communicated
The following three columns present the efficiency of
the search (the search time is divided between the BASE
time); extra time spent processing subroutines beyond
that allowed for by the BASE time; and the percentage
of time spent communicating between the two processors.

line-bottom - The final lines show the principal continuation.
The score is shown in brackets.

Position 1.

```
 'PDEP 0 SWEEP'
WHITE KG3 PA3 PB4 PF3 PF2 PH3
BLACK KF5 PA6 PB5 PG5 PH6
ALPHA -1000     BETA 1000       MAXDEPTH 8     . MMAXD 10

SINGLE PROCESSOR SEARCHES

         CDEPTH  UPDATES  SETMVS  CSTMVS SCORES  TIME-MS NOMAXD: NOMMAXD
THIS          0   16916    4700    2561   16916   14000   11054      23
OTHER         0   16916    4700    2561   16916    7800   11054      23
-------------------------------------------------------------------------
BASE #        0   16916    4700    2561   16916    5017   11054      23
TIME/SB       0     233    1413     923      62
TOTAL         0    3950    6650    2367    1033
%TIME         0      28      47      17       7

PARALLEL SEARCH

         CDEPTH    %INCR   %INCR   %INCR MICRSEC  EFFNCY  EFFNCY  EFFNCY
                  SUBRTN  COMMUN    TIME   /WORD  %TOTAL   %INCR       %
                    BASE    BASE    BASE  COMMUN    BASE  SUBRTN  COMMUN
-------------------------------------------------------------------------
VARCDEPTH 1

                0      16      20      36    2962      74      12      14

                1      28       1      28      38      78      22       1

                2       5       5      10     158      91       5       4

                3      25       5      30     130      77      19       4

                4      23       7      30     147      77      18       5

-------------------------------------------------------------------------
VARCDEPTH 0

                0      16      19      36    2827      74      12      14

                1      28       1      28      38      78      22       1

                2      31       9      39     188      72      22       6

                3      37      22      59     168      63      23      14

                4      35      62      97     190      51      18      32

BEST LINE FOUND AND SCORE:
PF3-F4  PG5XPF4+        KG3-H4  PH6-H5  KH4XPH5 PF4-F3  PH3-H4  KF5-E5(10

*
```

position 1

'MAXD 9 MMAXD 11 SWEEP'
WHITE KG3 PA3 PB4 PF3 PF2 PH3
BLACK KF5 PA6 PB5 PG5 PH6
ALPHA -1000 BETA 1000 MAXDEPTH 9 MMAXD 11

SINGLE PROCESSOR SEARCHES

	CDEPTH	UPDATES	SETMVS	CSTMVS	SCORES	TIME-MS	NOMAXD:	NOMMAXD
THIS	0	35650	15673	9651	35650	38783	17381	66
OTHER	0	35650	15673	9651	35650	21450	17381	66
BASE #	0	35650	15673	9651	35650	13817	17381	66
TIME/SB	0	233	1410	923	-15			
TOTAL	0	8317	22100	8917	-550			
%TIME	0	21	57	23	-1			

PARALLEL SEARCH

CDEPTH	%INCR SUBRTN BASE	%INCR COMMUN BASE	%INCR TIME BASE	MICRSEC /WORD COMMUN	EFFNCY %TOTAL BASE	EFFNCY %INCR SUBRTN	EFFNCY % COMMUN
VARCDEPTH 1							
0	24	16	40	7173	71	17	12
1	18	3	21	398	83	15	2
2	7	2	9	165	92	6	2
3	21	2	23	153	82	17	2
4	19	3	21	157	82	15	2
VARCDEPTH 0							
0	24	16	40	6077	71	17	12
1	18	2	21	380	83	15	2
2	37	2	39	108	72	27	1
3	24	9	34	195	75	18	7
4	45	21	66	175	60	27	12

BEST LINE FOUND AND SCORE:
PF3-F4 PG5XPF4+ KG3-H4 PH6-H5 KH4XPH5 PF4-F3 PH3-H4 KF5-E5
KH5-G5(100)

*

H5
Position 2

'WHITE KG3 A5 B5 C5 H3 BLACK KG5 A7 B7 C7 H5 WHITE'
*'MAXD 9 MMAXD 11 MAXCD 4 SWEEP'
WHITE KG3 PA5 PB5 PC5 PH3
BLACK KG5 PA7 PB7 PC7 PH5
ALPHA -1000 BETA 1000 MAXDEPTH 9 MMAXD 11

SINGLE PROCESSOR SEARCHES

	CDEPTH	UPDATES	SETMVS	CSTMVS	SCORES	TIME-MS	NOMAXD:	NOMMAXD
THIS	0	49421	15908	4705	49421	40183	32737	2
OTHER	0	49421	15908	4705	49421	22633	32737	2
BASE #	0	49421	15908	4705	49421	14483	32737	2
TIME/SB	0	237	1290	787	87			
TOTAL	0	11700	20517	3700	4267			
%TIME	0	29	51	9	11			

PARALLEL SEARCH

CDEPTH	%INCR SUBRTN BASE	%INCR COMMUN BASE	%INCR TIME BASE	MICRSEC /WORD COMMUN	EFFNCY %TOTAL BASE	EFFNCY %INCR SUBRTN	EFFNCY % COMMUN
VARCDEPTH 1							
0	21	2	23	1145	81	17	2
1	42	0	41	-18	71	29	0
2	12	2	14	142	88	10	2
3	19	2	21	125	83	16	2
4	17	4	21	140	83	14	3
VARCDEPTH 0							
0	21	2	23	1042	81	17	2
1	42	0	41	-18	71	29	0
2	24	2	26	135	79	19	2
3	33	9	42	190	70	23	6
4	26	26	52	188	66	17	17

BEST LINE FOUND AND SCORE:
PH3-H4+ KG5-F5 PB5-B6 PA7XPB6 PA5XPB6 PC7XPB6 PC5XPB6 KF5-E5
KG3-F3(0)

*

```
 'PDEP 0 MAXCD 4 MAXD 10 MMAXD 12'
*'SWEEP'
WHITE KG3 PA5 PB5 PC5 PH3
BLACK KG5 PA7 PB7 PC7 PH5
ALPHA -1000     BETA 1000       MAXDEPTH 10     MMAXD 12

SINGLE PROCESSOR SEARCHES

         CDEPTH  UPDATES  SETMVS  CSTMVS SCORES  TIME-MS NOMAXD: NOMMAXD
THIS          0  354371   91062  171211  354371  373650  148856   13244
OTHER         0  354371   91062  171211  354371  207267  148856   13244
-------------------------------------------------------------------------
BASE #        0  354371   91062  171211  354371  133317  148856   13244
TIME/SB       0     233    1287     790     537
TOTAL         0   67400   32850   83483  189917
%TIME         0      18       9      22      51

PARALLEL SEARCH

         CDEPTH    %INCR   %INCR   %INCR MICRSEC  EFFNCY  EFFNCY  EFFNCY
                  SUBRTN  COMMUN    TIME  /WORD  %TOTAL   %INCR       %
                    BASE    BASE    BASE COMMUN    BASE  SUBRTN  COMMUN
-------------------------------------------------------------------------
VARCDEPTH 1

                0      19      22      41  85722      71      13      16

                1      26      -1      25  -1340      80      21       1

                2       4       5       9   2813      91       4       5

                3      15       2      17    772      85      13       1

                4      15       2      17    658      86      13       2

-------------------------------------------------------------------------
VARCDEPTH 0

                0      19      22      41  79100      71      13      16

                1      26      -1      25  -1340      80      21       1

                2      33      -2      31   -910      76      25       0

                3       0       7       8   1017      93       0       7

                4      31       2      33     70      75      24       1

BEST LINE FOUND AND SCORE:
PB5-B6  PA7XPB6 PH3-H4+ KG5-F5  PC5-C6  PB7XPC6 PA5-A6  PC6-C5
PA6-A7  PC7-C6  PA7-A8=Q(600)

*
```

position 3

```
 'CLEAR BLACK KE6 D6 F6 WHITE KE3 D3 F2'
*' MAXD 8 MMAXD 10 SWEEP'
WHITE KE3 PD3 PF2
BLACK KE6 PD6 PF6
ALPHA -1000     BETA 1000       MAXDEPTH 8      MMAXD 10

SINGLE PROCESSOR SEARCHES

         CDEPTH  UPDATES  SETMVS  CSTMVS SCORES  TIME-MS NOMAXD: NOMMAXD
THIS          0   18037    7825    5708   18037   16500    8855       8
OTHER         0   18037    7825    5708   18037    9167    8855       8
-------------------------------------------------------------------------
BASE #        0   18037    7825    5708   18037    5900    8855       8
TIME/SB       0     240    1010     547      63
TOTAL         0    4333    7900    3117    1150
%TIME         0      26      48      19       7

PARALLEL SEARCH

         CDEPTH    %INCR   %INCR   %INCR MICRSEC  EFFNCY  EFFNCY  EFFNCY
                  SUBRTN  COMMUN    TIME  /WORD  %TOTAL   %INCR       %
                    BASE    BASE    BASE  COMMUN    BASE  SUBRTN  COMMUN
-------------------------------------------------------------------------
VARCDEPTH 1

                  0       4       1       5     158      96       4       1

                  1      45       3      48     170      68      31       2

                  2      14       6      20     172      83      11       5

                  3      16       9      24     182      80      12       7

                  4      12      12      25     177      80      10      10

-------------------------------------------------------------------------
VARCDEPTH 0

                  0       4       1       5     145      96       4       1

                  1      45       3      48     170      68      31       2

                  2      18       8      25     177      80      14       6

                  3      45      24      69     183      59      27      14

                  4      20      73      94     188      52      10      38

BEST LINE FOUND AND SCORE:
PF2-F4  PF6-F5  PD3-D4  PD6-D5  KE3-D3  KE6-D6  KD3-C3  KD6-C6(0)

*
```

— A6 —
position 3

```
 'CLEAR WHITE KE3 D3 F2 BLACK KE6 D6 F6 WHITE'
*'SWEEP'
WHITE KE3 PD3 PF2
BLACK KE6 PD6 PF6
ALPHA -1000     BETA 1000       MAXDEPTH 9      MMAXD 11

SINGLE PROCESSOR SEARCHES

         CDEPTH  UPDATES  SETMVS  CSTMVS SCORES  TIME-MS NOMAXD: NOMMAXD
THIS          0   49575   16872    5510   49575   36033   32172       1
OTHER         0   49575   16872    5510   49575   20350   32172       1
-------------------------------------------------------------------------
BASE #        0   49575   16872    5510   49575   13000   32172       1
TIME/SB       0     243    1007     550      80
TOTAL         0   12067   16983    3033    3950
%TIME         0      33      47       8      11

PARALLEL SEARCH

         CDEPTH    %INCR   %INCR   %INCR MICRSEC  EFFNCY  EFFNCY  EFFNCY
                  SUBRTN  COMMUN    TIME   /WORD  %TOTAL   %INCR       %
                    BASE    BASE    BASE  COMMUN    BASE  SUBRTN  COMMUN
-------------------------------------------------------------------------
VARCDEPTH 1

                  0       7       1       8     262      93       6       1

                  1      44       2      46     355      68      30       2

                  2      13       4      17     205      86      11       3

                  3      12       4      17     202      86      10       4

                  4      12       6      17     190      85      10       5

-------------------------------------------------------------------------
VARCDEPTH 0

                  0       7       1       8     240      93       6       1

                  1      44       3      47     373      68      30       2

                  2      23       4      26     178      79      18       3

                  3      40      12      53     208      65      26       8

                  4      25      33      58     187      63      16      21

BEST LINE FOUND AND SCORE:
PF2-F4  PF6-F5  PD3-D4  PD6-D5  KE3-D3  KE6-D6  KD3-C3  KD6-C6
KC3-B3(0)

*
```

—A7—
position 4

'SWEEP'
WHITE KH4 PB4 PD4
BLACK KH6 PC6 PD6
ALPHA -1000 BETA 1000 MAXDEPTH 8 MMAXD 10

SINGLE PROCESSOR SEARCHES

	CDEPTH	UPDATES	SETMVS	CSTMVS	SCORES	TIME-MS	NOMAXD:	NOMMAXD
THIS	0	22813	8426	9137	22813	20117	11830	4
OTHER	0	22813	8426	9137	22813	11167	11830	4
BASE #	0	22813	8426	9137	22813	7183	11830	4
TIME/SB	0	233	850	530	122			
TOTAL	0	5317	7167	4850	2783			
%TIME	0	26	36	24	14			

PARALLEL SEARCH

CDEPTH	%INCR SUBRTN BASE	%INCR COMMUN BASE	%INCR TIME BASE	MICRSEC /WORD COMMUN	EFFNCY %TOTAL BASE	EFFNCY %INCR SUBRTN	EFFNCY % COMMUN
VARCDEPTH 1							
0	13	21	34	5212	75	10	16
1	26	1	27	55	79	21	1
2	41	3	44	122	70	28	2
3	41	4	45	150	69	28	3
4	39	5	44	160	69	27	4
VARCDEPTH 0							
0	13	21	34	4740	75	10	16
1	26	1	27	72	79	21	1
2	31	4	35	137	74	23	3
3	30	18	48	182	67	20	12
4	16	56	72	187	58	10	32

BEST LINE FOUND AND SCORE:
KH4-G4 KH6-G6 KG4-F4 KG6-F6 KF4-G4 PD6-D5 KG4-F4 KF6-E6(0)

*

```
 'CLEAR WHITE KH4 B4 D4 BLACK KH6 C6 D6 WHITE '
*'SWEEP'
WHITE KH4 PB4 PD4
BLACK KH6 PC6 PD6
ALPHA -1000     BETA 1000       MAXDEPTH 9      MMAXD 11

SINGLE PROCESSOR SEARCHES

        CDEPTH  UPDATES  SETMVS  CSTMVS SCORES  TIME-MS NOMAXD: NOMMAXD
THIS         0    49471   15969    5744   49471   35067   33207      64
OTHER        0    49471   15969    5744   49471   19850   33207      64
-------------------------------------------------------------------------
BASE #       0    49471   15969    5744   49471   12683   33207      64
TIME/SB      0      238     847     530     135
TOTAL        0    11783   13517    3050    6717
%TIME        0       34      39       9      19

PARALLEL SEARCH

        CDEPTH    %INCR   %INCR   %INCR MICRSEC  EFFNCY  EFFNCY  EFFNCY
                SUBRTN  COMMUN    TIME  /WORD  %TOTAL   %INCR       %
                  BASE    BASE    BASE  COMMUN    BASE  SUBRTN  COMMUN
-------------------------------------------------------------------------
VARCDEPTH 1

                0       6      75      81   37467      55       3      41

                1      28       0      28      45      78      22       0

                2      46       2      48     147      68      31       1

                3      45       2      47     145      68      31       1

                4      45       3      48     167      68      30       2

-------------------------------------------------------------------------
VARCDEPTH 0

                0       6      75      81   33628      55       3      41

                1      28       0      28      45      78      22       0

                2      38       3      41     148      71      27       2

                3      35       9      45     183      69      25       6

                4      17      26      43     187      70      12      18

BEST LINE FOUND AND SCORE:
PD4-D5  PC6XPD5 PB4-B5  PD5-D4  PB5-B6  PD4-D3  PB6-B7  PD3-D2
PB7-B8=Q        PD2-D1=Q        QB8XPD6+(0)

*
```

position 5

'MAXD 7 MMAXD 9 SWEEP'
WHITE KD3 PA5 PB5 PC4 PG2 PH2
BLACK KE5 PA7 PB7 PF4 PG4 PH7
ALPHA -1000 BETA 1000 MAXDEPTH 7 MMAXD 9

SINGLE PROCESSOR SEARCHES

	CDEPTH	UPDATES	SETMVS	CSTMVS	SCORES	TIME-MS	NOMAXD:	NOMMAXD
THIS	0	21779	4793	2598	21779	17550	15511	176
OTHER	0	21779	4793	2598	21779	9883	15511	176
BASE #	0	21779	4793	2598	21779	6317	15511	176
TIME/SB	0	243	1543	883	117			
TOTAL	0	5300	7400	2300	2550			
%TIME	0	30	42	13	15			

PARALLEL SEARCH

CDEPTH	%INCR SUBRTN BASE	%INCR COMMUN BASE	%INCR TIME BASE	MICRSEC /WORD COMMUN	EFFNCY %TOTAL BASE	EFFNCY %INCR SUBRTN	EFFNCY % COMMUN
VARCDEPTH 1							
0	20	12	32	2245	76	15	9
1	37	4	41	252	71	26	3
2	7	9	17	190	86	6	8
3	8	11	19	188	84	7	9
4	7	17	24	193	81	6	14
VARCDEPTH 0							
0	20	12	32	2067	76	15	9
1	37	4	41	252	71	26	3
2	22	11	34	190	75	17	8
3	37	36	73	197	58	21	21
4	25	130	156	200	39	10	51

BEST LINE FOUND AND SCORE:
PH2-H4 PG4XPH4E.P. PG2XPH3 PH7-H5 PH3-H4 PF4-F3 PC4-C5(0)

*

A10
position 5

'MAXD 8 MMAXD 10 SWEEP'
WHITE KD3 PA5 PB5 PC4 PG2 PH2
BLACK KE5 PA7 PB7 PF4 PG4 PH7
ALPHA -1000 BETA 1000 MAXDEPTH 8 MMAXD 10

SINGLE PROCESSOR SEARCHES

	CDEPTH	UPDATES	SETMVS	CSTMVS	SCORES	TIME-MS	NOMAXD:	NOMMAXD
THIS	0	56504	20654	14186	56504	62050	29010	440
OTHER	0	56504	20654	14186	56504	34583	29010	440
BASE #	0	56504	20654	14186	56504	22200	29010	440
TIME/SB	0	243	1543	883	68			
TOTAL	0	13750	31883	12533	3883			
%TIME	0	22	51	20	6			

PARALLEL SEARCH

CDEPTH	%INCR SUBRTN BASE	%INCR COMMUN BASE	%INCR TIME BASE	MICRSEC /WORD COMMUN	EFFNCY %TOTAL BASE	EFFNCY %INCR SUBRTN	EFFNCY % COMMUN
VARCDEPTH 1							
0	11	19	31	12710	77	9	15
1	44	0	44	93	69	30	0
2	6	3	9	195	92	5	2
3	8	3	11	180	90	7	3
4	6	5	10	192	91	5	4
VARCDEPTH 0							
0	11	19	31	11703	77	9	15
1	44	0	44	93	69	30	0
2	16	4	20	245	84	13	3
3	45	10	55	200	64	29	7
4	17	36	54	197	65	11	24

BEST LINE FOUND AND SCORE:
PH2-H4 PG4XPH4E.P. PG2XPH3 PH7-H5 PH3-H4 PF4-F3 PC4-C5 PF3-F2(0)

*

position 31

'MAXD 4 MMAXD 6 SWEEP'
WHITE KC1 BG5 ND4 NC3 QD2 RD1 BF1 RH1 PA2 PB2 PC2 PE4 PF4 PG2 PH2
BLACK KG8 RA8 BC8 QD8 RF8 BE7 NC6 NF6 PA6 PB7 PD6 PE6 PF7 PG7 PH7
ALPHA -1000 BETA 1000 MAXDEPTH 4 MMAXD 6

SINGLE PROCESSOR SEARCHES

	CDEPTH	UPDATES	SETMVS	CSTMVS	SCORES	TIME-MS	NOMAXD:	NOMMAXD
THIS	0	12445	1642	2613	12445	19250	2578	1186
OTHER	0	12445	1642	2613	12445	10667	2578	1186
BASE #	0	12445	1642	2613	12445	6867	2578	1186
TIME/SB	0	250	4757	2780	85			
TOTAL	0	3117	7817	7267	1050			
%TIME	0	16	41	38	5			

PARALLEL SEARCH

CDEPTH	%INCR SUBRTN BASE	%INCR COMMUN BASE	%INCR TIME BASE	MICRSEC /WORD COMMUN	EFFNCY %TOTAL BASE	EFFNCY %INCR SUBRTN	EFFNCY % COMMUN
VARCDEPTH 1							
0	1	5	7	482	94	1	5
1	53	9	62	182	62	33	5
2	16	66	82	193	55	9	36
3	19	74	93	200	52	10	38
VARCDEPTH 0							
0	1	5	7	463	94	1	5
1	53	9	62	182	62	33	5
2	8	81	88	195	53	4	43
3	55	333	388	225	20	11	68

BEST LINE FOUND AND SCORE:
BG5XNF6 BE7XBF6 ND4XNC6 PB7XNC6(-15)

*

position 31

```
 'MAXD 5 MMAXD 7'
*'SWEEP'
WHITE KC1 BG5 ND4 NC3 QD2 RD1 BF1 RH1 PA2 PB2 PC2 PE4 PF4 PG2 PH2
BLACK KG8 RA8 BC8 QD8 RF8 BE7 NC6 NF6 PA6 PB7 PD6 PE6 PF7 PG7 PH7
ALPHA -1000     BETA 1000        MAXDEPTH 5      MMAXD 7

SINGLE PROCESSOR SEARCHES

         CDEPTH  UPDATES  SETMVS  CSTMVS SCORES  TIME-MS NOMAXD: NOMMAXD
THIS          0   84200    5347    4392   84200   70233   66476    4684
OTHER         0   84200    5347    4392   84200   39717   66476    4684
-------------------------------------------------------------------------
BASE #        0   84200    5347    4392   84200   25367   66476    4684
TIME/SB       0     252    4753    2780     135
TOTAL         0   21183   25417   12217   11417
%TIME         0      30      36      17      16

PARALLEL SEARCH

         CDEPTH    %INCR   %INCR   %INCR MICRSEC  EFFNCY  EFFNCY  EFFNCY
                  SUBRTN  COMMUN    TIME   /WORD  %TOTAL   %INCR       %
                    BASE    BASE    BASE  COMMUN    BASE  SUBRTN  COMMUN
-------------------------------------------------------------------------
VARCDEPTH 1

                0       8       2      10     663      91       7       2

                1      38       3      41     208      71      27       2

                2      10      19      29     190      78       8      14

                3      11      23      35     195      74       8      17

-------------------------------------------------------------------------
VARCDEPTH 0

                0       8       2      10     620      91       7       2

                1      38       3      41     208      71      27       2

                2      11      22      33     180      75       8      17

                3      31     108     139     192      42      13      45

BEST LINE FOUND AND SCORE:
ND4XNC6 PB7XNC6 PE4-E5  NF6-E8  BG5XBE7 QD8XBE7 PE5XPD6(100)

*
```

A13
— position 36

10 oct

'MAXCD 3 SWEEP'
WHITE KG1 NB1 RD1 RE1 NF1 QC2 BE3 PB2 PC3 PE4 PF3 PG2 PH3
BLACK KG8 RA8 RD8 BF8 QC7 BD7 NH5 PB5 PE5 PF7 PG6 PH6
ALPHA -1000 BETA 1000 MAXDEPTH 4 MMAXD 6

SINGLE PROCESSOR SEARCHES

	CDEPTH	UPDATES	SETMVS	CSTMVS	SCORES	TIME-MS	NOMAXD:	NOMMAXD
THIS	0	34726	4188	8267	34726	52767	12776	3961
OTHER	0	34726	4188	8267	34726	28917	12776	3961
BASE #	0	34726	4188	8267	34726	18683	12776	3961
TIME/SB	0	242	4423	2537	140			
TOTAL	0	8400	18517	20967	4883			
%TIME	0	16	35	40	9			

PARALLEL SEARCH

CDEPTH	%INCR SUBRTN BASE	%INCR COMMUN BASE	%INCR TIME BASE	MICRSEC /WORD COMMUN	EFFNCY %TOTAL BASE	EFFNCY %INCR SUBRTN	EFFNCY % COMMUN
VARCDEPTH 1							
0	1	1	2	230	98	1	1
1	41	7	48	165	67	28	5
2	20	39	58	187	63	12	24
3	20	51	71	193	58	12	30
VARCDEPTH 0							
0	1	1	2	223	98	1	1
1	41	7	48	163	68	28	5
2	8	77	85	192	54	4	41
3	56	392	448	217	18	10	72

BEST LINE FOUND AND SCORE:
BD7-E6 RD1XRD8 RA8XRD8 PH3-H4(85)

*

```
 'MAXD 5 MMAXD 7'
*'SWEEP'
WHITE KG1 NB1 RD1 RE1 NF1 QC2 BE3 PB2 PC3 PE4 PF3 PG2 PH3
BLACK KG8 RA8 RD8 BF8 QC7 BD7 NH5 PB5 PE5 PF7 PG6 PH6
ALPHA -1000     BETA 1000        MAXDEPTH 5      MMAXD 7

SINGLE PROCESSOR SEARCHES

         CDEPTH  UPDATES  SETMVS  CSTMVS SCORES  TIME-MS NOMAXD: NOMMAXD
THIS          0   306553   21874   44245  306553  330217  187837   45183
OTHER         0   306553   21874   44245  306553  183633  187837   45183
-------------------------------------------------------------------------
BASE #        0   306553   21874   44245  306553  118017  187837   45183
TIME/SB       0      245    4423    2533     152
TOTAL         0    75100   96750  112083   46283
%TIME         0       23      29      34      14

PARALLEL SEARCH

         CDEPTH    %INCR    %INCR    %INCR MICRSEC  EFFNCY   EFFNCY   EFFNCY
                  SUBRTN   COMMUN     TIME   /WORD  %TOTAL    %INCR        %
                    BASE     BASE     BASE  COMMUN    BASE  SUBRTN  COMMUN
-------------------------------------------------------------------------
VARCDEPTH 1

                0       9        0        9     -80      91        9        0

                1      16        1       18     188      85       14        1

                2      10        6       16     183      86        9        5

                3       9        8       17     190      85        8        7

-------------------------------------------------------------------------
VARCDEPTH 0

                0       9        0        9     -78      91        9        0

                1      16        1       18     188      85       14        1

                2      29       13       42     178      70       20        9

                3      20       83      103     198      49       10       41

BEST LINE FOUND AND SCORE:
NH5-F6  PH3-H4  RA8-A1  PH4-H5  NF6XPH5(-15)

*
```

```
 'CLEAR'
*'BLACK QC8 RD8 RF8 KG8 BB7 ND6 ND4 A7 B6 C5 D7 E6 F7 G6 H7'
*'WHITE RA1 NC3 NE1 RF1 KG1 BG2 QG4 A2 B2 C4 D3 E4 F6 G5 H2'
*'BOARD'     ....     ....     ....     ....
    .... *Q .*R.     .*R. *K ....
    ....     ....     ....     ....
....     ....     ....     ....
.*P. *B ....  *P ....  *P ....  *P
....     ....     ....     ....
    ....     ....     ....     ....
    .*P.     .*N. *P . P. *P ....
    ....     ....     ....     ....
....     ....     ....     ....
....     .*P.     ....     . P.
....     ....     ....     ....
    ....     ....     ....     ....
    ....   P .*N.   P ....   Q ....
    ....     ....     ....     ....
....     ....     ....     ....
....     . N.   P ....     ....
....     ....     ....     ....
    ....     ....     ....     ....
  P . P.     ....     ....   B . P.
    ....     ....     ....     ....
....     ....     ....     ....
. R.     ....     . N.   R . K.
....     ....     ....     ....

  A    B    C    D    E    F    G    H

*'BLACK MAXD 4 MMAXD 6 CD 1 VARCD 0'
*'GO'PARALLEL SEARCH

BEST LINE FOUND AND SCORE:
PE6-E5  PH2-H4  PA7-A5  PH4-H5(0)

ALPHA -1000     BETA 1000       MAXDEPTH 4      MMAXD 6
CDEPTH 1        MOVES FROM CD: 84
UPDATES 10363    SETMVS 888      CSTMVS 3250    NOMAXD: 3639
NOMMAXD 2006    NBP: 0
UPDATES/SEC: 560        SETMVS/SEC: 48  CSTMVS/SEC: 176 NOMAXD/SEC: 197
NOMMAXD/SEC: 108        NBP/SEC: 0
ELAPSED TIME(SECS): 18  & 30'60 THS SECS
BUSY LOOP TIME(SECS): 0 & 1     '60 THS SECS,%= 0
LINK-RECEIVE INTERRUPTS: 0      WORDS RECEIVED: 3155    WORDS SENT: 2579
*'MAXD 5 MMAXD 7 GO'PARALLEL SEARCH

BEST LINE FOUND AND SCORE:
PE6-E5  PH2-H4  PB6-B5  NC3XPB5 ND6XPE4(0)

ALPHA -1000     BETA 1000       MAXDEPTH 5      MMAXD 7
CDEPTH 1        MOVES FROM CD: 99
UPDATES 68213    SETMVS 5364     CSTMVS 13599   NOMAXD: 35114
NOMMAXD 9955    NBP: 0
UPDATES/SEC: 767        SETMVS/SEC: 60  CSTMVS/SEC: 153 NOMAXD/SEC: 395
NOMMAXD/SEC: 112        NBP/SEC: 0
ELAPSED TIME(SECS): 88  & 58'60 THS SECS
BUSY LOOP TIME(SECS): 0 & 0     '60 THS SECS,%= 0
LINK-RECEIVE INTERRUPTS: 0      WORDS RECEIVED: 3341    WORDS SENT: 2827
*
```

position 485

'MAXD 4 MMAXD 6 SWEEP'
WHITE KG1 RA1 NC3 NE1 RF1 BG2 QG4 PA2 PB2 PC4 PD3 PE4 PF6 PG5 PH2
BLACK KG8 QC8 RD8 RF8 BB7 ND6 ND4 PA7 PB6 PC5 PD7 PE6 PF7 PG6 PH7
ALPHA -1000 BETA 1000 MAXDEPTH 4 MMAXD 6

SINGLE PROCESSOR SEARCHES

	CDEPTH	UPDATES	SE1MVS	CSTMVS	SCORES	TIME-MS	NOMAXD:	NOMMAXD
THIS	0	25162	2121	6680	25162	37717	9572	4174
OTHER	0	25162	2121	6680	25162	20833	9572	4174
BASE #	0	25162	2121	6680	25162	13417	9572	4174
TIME/SB	0	247	4380	2683	172			
TOTAL	0	6200	9283	17917	4317			
%TIME	0	16	25	48	11			

PARALLEL SEARCH

CDEPTH	%INCR SUBRTN BASE	%INCR COMMUN BASE	%INCR TIME BASE	MICRSEC /WORD COMMUN	EFFNCY %TOTAL BASE	EFFNCY %INCR SUBRTN	EFFNCY % COMMUN
VARCDEPTH 1							
0	13	5	18	1055	85	11	5
1	28	8	36	185	74	21	6
2	20	29	50	190	67	14	20
3	24	37	60	197	62	15	23
VARCDEPTH 0							
0	13	5	18	1013	85	11	5
1	28	8	35	183	74	20	6
2	7	59	65	193	60	4	36
3	47	334	381	217	21	10	69

BEST LINE FOUND AND SCORE:
PE6-E5 PH2-H4 PA7-A5 PB2-B4(0)

*

position 48b

```
 'MAXD 5 MMAXD 7'
*'SWEEP'
WHITE KG1 RA1 NC3 NE1 RF1 BG2 QG4 PA2 PB2 PC4 PD3 PE4 PF6 PG5 PH2
BLACK KG8 QC8 RD8 RF8 BB7 ND6 ND4 PA7 PB6 PC5 PD7 PE6 PF7 PG6 PH7
ALPHA -1000     BETA 1000       MAXDEPTH 5      MMAXD 7

SINGLE PROCESSOR SEARCHES

         CDEPTH  UPDATES  SETMVS  CSTMVS SCORES  TIME-MS NOMAXD: NOMMAXD
THIS          0  176439   14537   38344  176439  236667   81799   30427
OTHER         0  176439   14537   38344  176439  131083   81799   30427
-------------------------------------------------------------------------
BASE #        0  176439   14537   38344  176439   84367   81799   30427
TIME/SB       0     247    4380    2683     150
TOTAL         0   43517   63667  102883   26600
%TIME         0      18      27      43      11

PARALLEL SEARCH

         CDEPTH    %INCR    %INCR    %INCR MICRSEC  EFFNCY  EFFNCY  EFFNCY
                  SUBRTN   COMMUN     TIME   /WORD  %TOTAL   %INCR       %
                    BASE     BASE     BASE  COMMUN    BASE  SUBR1N  COMMUN
-------------------------------------------------------------------------
VARCDEPTH 1

                  0      14       4      18    4583      85      12       3

                  1       4       2       5     212      95       4       1

                  2       9       4      13     172      88       8       4

                  3      10       6      16     188      86       8       6

-------------------------------------------------------------------------
VARCDEPTH 0

                  0      14       4      18    4398      85      12       3

                  1       4       2       5     213      95       4       2

                  2      26       9      35     180      74      19       7

                  3      14      73      87     200      53       8      39

BEST LINE FOUND AND SCORE:
PE6-E5  PH2-H4  PB6-B5  NC3XPB5 ND6XPE4(0)

*
```

position 55A

```
 'MAXD 4 MMAXD 6 MAXCD 3 SWEEP'
WHITE KE1 RA1 RH1 QF3 PA2 PB2 PC3 PD4 PF2 PG2 PH2 BD3 BG5 NE5
BLACK KG8 RF8 QD8 RA8 PA7 PB7 PG7 PH7 PF7 BC8 NE8 PC6 PE6 BE7
ALPHA -1000     BETA 1000       MAXDEPTH 4      MMAXD 6

'SINGLE PROCESSOR SEARCHES
```

	CDEPTH	UPDATES	SELMVS	CSTMVS	SCORES	TIME-MS	NOMAXD:	NOMMAXD
THIS	0	62482	6153	17095	62482	93150	12609	6550
OTHER	0	62482	6153	17095	62482	51367	12609	6550
BASE #	0	62482	6153	17095	62482	33117	12609	6550
TIME/SB	0	255	5637	2843	-95			
TOTAL	0	15933	34683	48600	-6065			
%TIME	0	17	37	52	-5			

PARALLEL SEARCH

CDEPTH	%INCR SUBRTN BASE	%INCR COMMUN BASE	%INCR TIME BASE	MICRSEC /WORD COMMUN	EFFNCY %TOTAL BASE	EFFNCY %INCR SUBRTN	EFFNCY % COMMUN
VARCDEPTH 1							
0	0	1	1	322	99	0	1
1	20	4	25	197	80	16	3
2	35	33	68	188	60	21	20
3	34	36	71	193	59	20	21
VARCDEPTH 0							
0	0	1	1	328	99	0	1
1	20	4	25	197	80	16	3
2	7	59	66	190	60	4	36
3	52	304	356	217	22	11	67

```
BEST LINE FOUND AND SCORE:
BG5XBE7 QD8XBE7 NE5-C4  PE6-E5(0)

*
```

position 55A

'MAXD 5 MMAXD 7 SWEEP'
WHITE KE1 RA1 RH1 QF3 PA2 PB2 PC3 PD4 PF2 PG2 PH2 BD3 BG5 NE5
BLACK KG8 RF8 QD8 RA8 PA7 PB7 PG7 PH7 PF7 BC8 NE8 PC6 PE6 BE7
ALPHA -1000 BETA 1000 MAXDEPTH 5 MMAXD 7

SINGLE PROCESSOR SEARCHES

	CDEPTH	UPDATES	SETMVS	CSTMVS	SCORES	TIME-MS	NOMAXD:	NOMMAXD
THIS	0	512990	24035	32983	512990	402733	420555	36264
OTHER	0	512990	24035	32983	512990	228067	420555	36264
BASE #	0	512990	24035	32983	512990	145617	420555	36264
TIME/SB	0	255	5633	2847	83			
TOTAL	0	130817	135400	93883	42633			
%TIME	0	32	34	23	11			

PARALLEL SEARCH

CDEPTH	%INCR SUBRTN BASE	%INCR COMMUN BASE	%INCR TIME BASE	MICRSEC /WORD COMMUN	EFFNCY %TOTAL BASE	EFFNCY %INCR SUBRTN	EFFNCY % COMMUN
VARCDEPTH 1							
0	-3	1	-3	1665	104	-3	1
1	11	0	11	38	90	10	0
2	31	6	37	147	73	23	4
3	33	7	40	152	71	24	5
VARCDEPTH 0							
0	-3	1	-3	1627	104	-3	1
1	11	0	11	40	90	10	0
2	10	14	24	167	81	8	11

```
 'MAXCD 3 MAXD 5 MMAXD 7 SWEEP'
WHITE KE1 RA1 RH1 QF3 PA2 PB2 PC3 PD4 PF2 PG2 PH2 BD3 BG5 NE5
BLACK KG8 RF8 QD8 RA8 PA7 PB7 PG7 PH7 PF7 BC8 NE8 PC6 PE6 BE7
ALPHA -1000     BETA 1000        MAXDEPTH 5      MMAXD 7

SINGLE PROCESSOR SEARCHES

      CDEPTH  UPDATES  SETMVS  CSTMVS SCORES  TIME-MS NOMAXD: NOMMAXD
THIS       0  512990   24035   32983  512990  402333  420555   36264
OTHER      0  512990   24035   32983  512990  227950  420555   36264
-------------------------------------------------------------------------
BASE #     0  512990   24035   32983  512990  145517  420555   36264
TIME/SB    0     255    5627    2843      83
TOTAL      0  130817  135233   93783   42500
%TIME      0      33      34      23      11

PARALLEL SEARCH

        CDEPTH    %INCR   %INCR   %INCR MICRSEC  EFFNCY  EFFNCY  EFFNCY
                  SUBRTN  COMMUN    TIME   /WORD  %TOTAL   %INCR       %
                    BASE    BASE    BASE  COMMUN    BASE  SUBRTN  COMMUN
-------------------------------------------------------------------------
VARCDEPTH 1

                0      -3       1      -3    1625     104      -3       1

                1      11       0      12      43      90      10       0

                2      31       6      37     150      73      22       5

                3      34       7      41     152      71      24       5

-------------------------------------------------------------------------
VARCDEPTH 0

                0      -3       1      -3    1573     104      -3       1

                1      11       0      11      30      90      10       0

                2      10      14      24     167      81       8      11

                3      34      84     119     185      46      16      39

BEST LINE FOUND AND SCORE:
NE5XPC6 PB7XNC6 QF3XPC6 BE7XBG5 QC6XRA8(85)

*
```

```
 'CLEAR'
*'BLACK KG8 RA8 RF8 A7 B7 H7 G7 F7 D5 BF6 ND7 NG4'
*'WHITE KC1 RD1 RH1 A2 B2 H2 G2 F2 C2 BG3 NE2 NF3'
*'BOARD'     ....      ....      ....      ....
 *R ....     ....     .*R.  *K ....
    ....     ....      ....      ....
....     ....      ....      ....
.*P.  *P ....  *N ....  *P .*P.  *P
....     ....      ....      ....
    ....     ....      ....      ....
    ....     ....     .*B.       ....
    ....     ....      ....      ....
....     ....      ....      ....
....     ....  *P  ....      ....
....     ....      ....      ....
    ....     ....      ....      ....
    ....     ....      .... *N ....
    ....     ....      ....      ....
....     ....      ....      ....
....     ....      ....   N . B.
....     ....      ....      ....
    ....     ....      ....      ....
  P . P.   P ....   N . P.   P . P.
    ....     ....      ....      ....
....     ....      ....      ....
....     . K.   R  ....      ....   R
....     ....      ....      ....

  A    B    C    D    E    F    G    H

*'BLACK MAXD 4 MMAXD 6 CD 0 GO'PARALLEL SEARCH

BEST LINE FOUND AND SCORE:
NG4-E5  RD1XPD5 NE5XNF3 RD5XND7(100)

ALPHA -1000      BETA 1000        MAXDEPTH 4       MMAXD 6
CDEPTH 0         MOVES FROM CD: 16
UPDATES 5337      SETMVS 635       CSTMVS 1801     NOMAXD: 1380
NOMMAXD 945      NBP: 0
UPDATES/SEC: 622          SETMVS/SEC: 74  CSTMVS/SEC: 210 NOMAXD/SEC: 161
NOMMAXD/SEC: 110          NBP/SEC: 0
ELAPSED TIME(SECS): 8    & 35'60 THS SECS
BUSY LOOP TIME(SECS): 0 & 1      '60 THS SECS,%= 0
LINK-RECEIVE INTERRUPTS: 0       WORDS RECEIVED: 478      WORDS SENT: 363
*'MAXD 5 MMAXD 7 GO'PARALLEL SEARCH

BEST LINE FOUND AND SCORE:
ND7-C5  RD1XPD5 NC5-E4  BG3-C7  NE4XPF2(0)

ALPHA -1000      BETA 1000        MAXDEPTH 5       MMAXD 7
CDEPTH 0         MOVES FROM CD: 13
UPDATES 35967     SETMVS 3782      CSTMVS 5599     NOMAXD: 24309
NOMMAXD 2819     NBP: 0
UPDATES/SEC: 901          SETMVS/SEC: 95  CSTMVS/SEC: 140 NOMAXD/SEC: 609
NOMMAXD/SEC: 71 NBP/SEC: 0
ELAPSED TIME(SECS): 39  & 56'60 THS SECS
BUSY LOOP TIME(SECS): 0 & 11     '60 THS SECS,%= 0
LINK-RECEIVE INTERRUPTS: 0       WORDS RECEIVED: 440      WORDS SENT: 330
*
```

```
 'MAXD 4 MMAXD 6 MAXCD 3 SWEEP'
WHITE KC1 RD1 RH1 PA2 PB2 PH2 PG2 PF2 PC2 BG3 NE2 NF3
BLACK KG8 RA8 RF8 PA7 PB7 PH7 PG7 PF7 PD5 BF6 ND7 NG4
ALPHA -1000     BETA 1000       MAXDEPTH 4      MMAXD 6

SINGLE PROCESSOR SEARCHES

         CDEPTH  UPDATES  SETMVS  CS1MVS SCORES  TIME-MS NOMAXD: NOMMAXD
THIS          0    15914    1732    5018  15914    23717    5468    2565
OTHER         0    15914    1732    5018  15914    13083    5468    2565
-------------------------------------------------------------------------
BASE #        0    15914    1732    5018  15914     8433    5468    2565
TIME/SB       0      252    3870    2183    130
TOTAL         0     4000    6700   10950   2067
%TIME         0       17      28      46      9

PARALLEL SEARCH

         CDEPTH    %INCR   %INCR   %INCR MICRSEC  EFFNCY  EFFNCY  EFFNCY
                  SUBRTN  COMMUN    TIME   /WORD  %TOTAL   %INCR       %
                    BASE    BASE    BASE  COMMUN    BASE  SUBRTN  COMMUN
-------------------------------------------------------------------------
VARCDEPTH 1

                  0       1       5       5     598      96       1       5

                  1      38       9      47     168      68      26       6

                  2      15      47      62     192      62       9      29

                  3      13      58      71     198      58       8      34

-------------------------------------------------------------------------
VARCDEPTH 0

                  0       1       5       5     597      95       1       5

                  1      38       9      47     168      68      26       6

                  2      13      67      80     195      55       7      37

                  3      44     353     397     220      20       9      71

BEST LINE FOUND AND SCORE:
NG4-E5  RD1XPD5 NE5XNF3 RD5XND7(100)

*
```

'MAXD 5 MMAXD 7 SWEEP'
WHITE KC1 RD1 RH1 PA2 PB2 PH2 PG2 PF2 PC2 BG3 NE2 NF3
BLACK KG8 RA8 RF8 PA7 PB7 PH7 PG7 PF7 PD5 BF6 ND7 NG4
ALPHA -1000 BETA 1000 MAXDEPTH 5 MMAXD 7

SINGLE PROCESSOR SEARCHES

	CDEPTH	UPDATES	SETMVS	CSTMVS	SCORES	TIME-MS	NOMAXD:	NOMMAX
THIS	0	86700	8843	12452	86700	92450	60237	544
OTHER	0	86700	8843	12452	86700	51633	60237	544
BASE #	0	86700	8843	12452	86700	33133	60237	544
TIME/SB	0	252	3870	2180	107			
TOTAL	0	21817	34217	27150	9267			
%TIME	0	24	37	29	10			

PARALLEL SEARCH

CDEPTH	%INCR SUBRTN BASE	%INCR COMMUN BASE	%INCR TIME BASE	MICRSEC /WORD COMMUN	EFFNCY %TOTAL BASE	EFFNCY %INCR SUBRTN	EFFNC COMMU
VARCDEPTH 1							
0	20	0	20	-18	83	17	
1	27	3	30	210	77	21	
2	7	13	20	192	84	6	1
3	5	16	21	193	83	4	1
VARCDEPTH 0							
0	20	0	20	-20	83	17	
1	27	3	30	207	77	21	
2	26	19	45	185	69	18	1
3	25	122	147	202	40	10	4

BEST LINE FOUND AND SCORE:
ND7-B6 NF3-D2 NG4-E5 ND2-B3 NE5-D7(0)

*

APPENDIX B. TELETYPE INPUT COMMANDS RECOGNISED BY PROGRAM.

COMMAND	EXPLANATION
GO	Begin an alpha-beta tree search
SWEEP	Perform parallel searches with varying values of VARCD and CD, and print table of results.
END	End program execution.
CLEAR	Initialise the board
WHITE BLACK	Either 1) the following piece descriptors are assumed to be white/black. or 2) it is white/black's turn to move.
SET PIECE	Follow with the pieces which are to be set on the board.

The following key words are used to set variables.
Follow the word with a space and then the number.

ALPHA BETA	Search width parameters.
MAXDEPTH	Maximum depth to which all moves are searched. Beyond this depth only captures and queening pawn moves are searched.
MMAXDEPTH	Maximum depth of quiescent search. No updates are made beyond this depth. The positiin is evaluated statically.
MOVENO	No of moves from the start.
CDEPTH	Common depth of the tw processors in parallel search mode. Moves at this and lower depths are common to both processors.
VARCD	Variable common depth flag. Non zero when CDEPTH is to be varied during the search.

SETTING THE SWITCH REGISTERS FOR A PARALLEL SEARCH.

BIT #	DESCRIPTION
0	Set up to put machine in parallel mode.
1	Set up on one machine.This machine leads the search initially.
2	Set up for print out of messages sent between the two machines.

APPENDIX C : PARALLEL CHESS. RT-11 MACRO VM02-12 20-NOV-78 09:43:52
TABLE OF CONTENTS

TABLE OF CONTENTS

```
                .TITLE APPENDIX C : PARALLEL CHESS.
                .SBTTL MACROS
                ;
                .MCALL .REGDEF,..V2..,.PRINT,.TTYIN,.EXIT,.GTIM
                .MCALL .TTINR
 000000         .REGDEF
 000000         ..V2..
                        .GLOBL PDUMP,SENDCM,MLOOP,RETCALL,SELIN
                        .GLOBL MINS,MAXS,MINR,MAXR
                        .GLOBL NEWSC,UPSCOR
                        .GLOBL MSTACK
                ;
                ;
                ;
                ;
                        .MACRO SAVREG X0,X1,X2,X3,X4,X5
                        .IRP X,<X0,X1,X2,X3,X4,X5>
                                .IIF NB,X,MOV  R'X,-(SP)
                        .ENDM
                        .ENDM SAVREG
                ;
                        .MACRO RESREG X0,X1,X2,X3,X4,X5
                        .IRP X,<X5,X4,X3,X2,X1,X0>
                                .IIF NB,X,MOV  (SP)+,R'X
                        .ENDM
                        .ENDM RESREG
                ;
                        .MACRO STIME L,H,?L1,?L2,?L3
                        ;R0 LOST
                        .GTIM #L1,#L2
                        BR   L3
                        L1:     .BLKW 2
                        L2:     .BLKW 2
                L3:     MOV  L2,H
                        MOV  L2+2,L
                        .ENDM
                ;
                        .MACRO DCLR X
                        CLR  X
                        CLR  X+2
                        .ENDM DCLR
                ;
                        .MACRO DMOV A,B
                        MOV A,B
                        MOV A+2,B+2
                        .ENDM
                ;
                ;MACRO TO SAVE R0 DURING PRINT
                        .MACRO SPRINT A
                        MOV  R0,-(SP)
                        .PRINT A
                        MOV  (SP)+,R0
                        .ENDM
                ;
                        .MACRO MUL A,B,?L1,?L2
                        MOV  A,-(SP)
                        MOV  B,-(SP)
```

```
                        BR   L2
                        L1:     ADD  (SP),B
                        L2:     DEC  2(SP)
                                BGT  L1
                        CMP  (SP)+,(SP)+
                        .ENDM
                ;
                ;
                        .MACRO DTST L,H,?L1
                        TST  H
                        BNE  L1
                                TST L
                                CLN
                L1:
                        .ENDM
                ;
                        .MACRO DINC L,H
                        ADD  #1,L
                        ADC  H
                        .ENDM
                ;
                        .MACRO DDECR L,H
                        SUB  #1,L
                        SBC  H
                        .ENDM
                ;
                        .MACRO DMOL2,L,H,OFLOW
                        ASL  L
                        ROL  H  ;MOVES C TO LOW BIT
                        BVS  OFLOW
                        .ENDM
                ;
                        .MACRO DSTACK L,H
                        MOV H,-(SP)
                        MOV L,-(SP)
                        .ENDM
                ;
                        .MACRO DUSTACK L,H
                        MOV  (SP)+,L
                        MOV  (SP)+,H
                        .ENDM
                ;
                        .MACRO DNEG L,H
                        NEG H
                        NEG L
                        SBC H
                        .ENDM
                ;
                        .MACRO DADD L1,H1,L2,H2
                        ADD  L1,L2
                        ADC H2
                        ADD  H1,H2
                        .ENDM
                ;
                        .MACRO DSUB L1,H1,L2,H2
                        SUB  L1,L2
                        SBC  H2
```

```
                        SUB  H1,H2
                        .ENDM
                ;
                        .MACRO DCMP L1,H1,L2,H2,LEQ,GTR
                        CMP  H1,H2
                        BGT  GTR
                        BLT  LEQ
                        CMP  L1,L2
                        BHI  GTR
                        BR   LEQ
                        .ENDM
                ;
                        .MACRO DDIV LD,HD,LN,HN,LQ,HQ,LR,HR,?L1,?L2,?L3,?L4
                        CLR  LQ
                        CLR  HQ
                        TST  LD
                        BNE  L3
                                TST  HD
                                                ;DIVISION BY SERO
                        BEQ  L4
                L3:
                        MOV  LN,LR
                        MOV  HN,HR
                        TST  HR
                        BPL  L1
                                DNEG LR,HR
                L1:
                        DSUB LD,HD,LR,HR
                        BLT  L2
                                DINC LQ,HQ
                                BR   L1
                L2:
                        DADD LD,HD,LR,HR
                        TST  HN
                        BPL  L4
                                DNEG LQ,HQ
                L4:
                        .ENDM
                ;
                        .MACRO DMOL A0,A1,B0,B1,C0,C1,?L1,?L2
                        MOV  B0,C0
                        MOV  B1,C1
                        DSTACK A0,A1
                        BR  L2
                        L1:     DADD B0,B1,C0,C1
                        L2:     DDECR (SP),2(SP)
                                DTST (SP),2(SP)
                                BGT  L1
                        CMP  (SP)+,(SP)+
                        .ENDM
                ;
                ;
                ;
```

```
                        .SBTTL VARIABLES
                        .CSECT VARIABLES
                        .=.+30000
030000          MSTACK:
030000          FR:     ;FREE AREA FOR TEMPORARY STORE AT BOTTOM OF STACK AREA
                        .=.+1000
031000          FRONT:
                BL=40
                CR=15
                LF=12
                ;PIECE.
                DUMMY=2.
                NIGHT=4.
                BISHOP=6.
                ROOK=8.
                QUEEN=10.
                KING=12.
                PAWN=14.
                        THRU=15.
                UROOK=16.
                UKING=20.
                        THRUP=21.
                UPAWN=22.
                ;THR.
                THR=29.
                CASTLE=30.
                EPCAP=32.
                QPAWN=34.
                NPAWN=36.
                DT=8.
                NEXTSQ=10.
                SSW=0+NEXTSQ
                SSE=2+NEXTSQ
                WSW=4.+NEXTSQ
                SW=6.+NEXTSQ
                S=8.+NEXTSQ
                SE=10.+NEXTSQ
                ESE=12.+NEXTSQ
                W=14.+NEXTSQ
                E=16.+NEXTSQ
                WNW=18.+NEXTSQ
                NW=20.+NEXTSQ
                N=22.+NEXTSQ
                NE=24.+NEXTSQ
                ENE=26.+NEXTSQ
                NNW=28.+NEXTSQ
                NNE=30.+NEXTSQ
                ;       DCL 1 PIECE(0:31),
                ;               2 ON PTR,
                ;               2 (NXTR,LSTR,REL BIT)
                ;               2 CLR,
                ;               2 (TYP,PTYP BIT(32)),
                ;               2 VAL,
                ;               2 (NXTP,LSTP),
                ;               2 DADD PTR;
                        LENP=24.
031000          PIECE:  .BLKB 32.*LENP
```

```
                ENDP:
                        ON=0.
                        NXTR=2.
                        LSTR=4.
                        REL=6.
                        CLR=8.
                        TYP=10.
                        PTYP=12.
                        VAL=16.
                        NXTP=18.
                        LSTP=20.
                        DADD=22.
                        ;
                        ;
                        ;
                ;       DCL 1 SQUARE(0:63),
                ;       2 PCE PTR,
                ;       2 (NXTR,LSTR,REL BIT),
                ;       2 CSQ CHAR(2),
                ;       2 NXTSQ(0:15) PTR;
                        LENSQ=42.
 32400          SQUARE: .BLKB   64.*LENSQ
 37600          ENDSQ:
                        PCE=0.
                        NXTR=2.
                        LSTR=4.
                        REL=6.
                        CSQ=8.
                        NXTSQ=10.
 37600          DIRS:
 37600                  .WORD   0,SSW,0,SSE,0
 37612                  .WORD   WSW,SW,S,SE,ESE
 37624                  .WORD   0,W,0,E,0
 37636                  .WORD   WNW,NW,N,NE,ENE
 37650                  .WORD   0,NNW,0,NNE,0
 37662          ADIRS:
 37662                  .WORD   0,0,NDIRS,BDIRS,RDIRS,QDIRS,QDIRS,BPDIRS,RDIRS,0
 37712          VALUES:
 37712                  .WORD   0,0,300.,315.,500.,900.,5000.,100.,500.,0,5000.,1C
 37742          PEN:    .WORD 1000.,0   ;WIDTH OF A B SEARCH IS 2*PEN
                                        ;IRRELEVANT MOVE PENALTY FOR GOING OUT
                PCNT=33.                ;BONUS FOR HAVING PIECE ON REL SQ
                TEMPO=17.               ;TEMPO SCORE
                ;USED BY MINMAX ONLY
 7746                   .WORD 0,NNW,WNW,WSW,SSW,SSE,ESE,ENE,NNE
 7770           NDIRS:
 7770                   .WORD 0,SE,SW,NW,NE
 0002           BDIRS:
 0002                   .WORD 0,W,E,S,N
 0014           RDIRS:
 0014                   .WORD 0,W,E,S,N,SE,SW,NW,NE
 0036           QDIRS:
 0036                   .WORD 0,SE,SW,S
 0046           BPDIRS:
 0046                   .WORD 0,NE,NW,N
 0056           WPDIRS:
                ;
```

VARIABLES

1[illegible] ;
1[illegible] ;

```
1                       .SBTTL OUTPUT MESSAGES
2  040056       NLTB:   .ASCIZ <CR><LF><TAB><TERM>
3  040063       M1:     .ASCIZ <CR><LF>/%/<TERM>          ;EXPECTING INPUT
4  040070       M2:     .ASCIZ /NON-NUMERIC INPUT/
5  040112       M3:     .ASCIZ /INVALID PIECE DESCRIPTOR/
6  040143       M4:     .ASCIZ /INVALID SQUARE/
7  040162       M6:     .ASCIZ <CR><LF>/BEST LINE FOUND AND SCORE:/<TERM>
8  040220       M7:     .ASCIZ /UPDATES/
9  040230       M8:     .ASCIZ / SETMVS/
10 40240        M9:     .ASCIZ / CSTMVS/
11 40250        M10:    .ASCIZ /ELAPSED TIME(SECS):/
12 40274        M11:    .ASCIZ  /&/
13 40276        M12:    .ASCIZ /'60 THS SECS/<TERM>
14 40314        M15:    .ASCIZ /ILLEGAL POSITION/
15 40335        M16:    .ASCIZ /MOVE NOT RECOGNISED/
16 40361        M17:    .ASCII /..../<TERM>
17 40366        M18:    .ASCII /./<TERM>
18 40370        M19:    .ASCIZ /DECREASING DEPTH/
19 40411        M20:    .ASCIZ /INCREASING DEPTH/
20 40432        M21:    .ASCIZ /NOMAXD:/
21 40442        M22:    .ASCIZ /NOMMAXD/
22 40452        M23:    .ASCIZ /IRREL MOVE CUTOFFS:/
23 40476        M24:    .ASCIZ /MOVES FROM CD:/
24 40515        M25:    .ASCIZ /UPDATES PER SEC:/
25 40536        M26:    .ASCIZ /ERROR : MESSAGE CONTENTS NOT RECOGNISED/
26 40606        M27:    .ASCIZ /PARALLEL SEARCH/
27 40626        M28:    .ASCIZ /LEAD SEARCH INITIALLY/
28 40654        M29:    .ASCIZ /OUTPUT BUFFER OVERFLOWS/
29 40704        M30:    .ASCIZ /NBP PER SEC:/
30 40721        M31:    .ASCIZ /BUSY LOOP TIME(SECS):/
31 40747        M32:    .ASCIZ /'60 THS SECS,%=/
32 40767        M33:    .ASCIZ /SINGLE PROC. SEARCH/
33 41013        M35:    .ASCIZ /LINK-RECEIVE INTERRUPTS:/
34 41044        M36:    .ASCIZ /WORDS RECEIVED:/
35 41064        M37:    .ASCIZ /WORDS SENT:/
36 41100        M38:    .ASCIZ /NBP:/
37              SL=57   ;=/
38 41105        M40:    .ASCIZ /UPDATES/<SL>/SEC:/
39 41122        M41:    .ASCIZ /SETMVS/<SL>/SEC:/
40 41136        M42:    .ASCIZ /CSTMVS/<SL>/SEC:/
41 41152        M43:    .ASCIZ /NOMAXD/<SL>/SEC:/
42 41166        M44:    .ASCIZ /NOMMAXD/<SL>/SEC:/
43 41203        M45:    .ASCIZ /NBP/<SL>/SEC:/
44 41214        M46:    .ASCIZ <CR><LF>/UNEXPECTED CUTOFF/<TERM>
45 41241        M47:    .ASCIZ /UNEXPECTED CUTOFF IN M1NMX2/<TERM>
46 41276        M48:    .ASCIZ <CR><LF>/SINGLE PROCESSOR SEARCHES/
47 41332        M49:    .ASCIZ <CR><LF>/THIS/<TERM>
48 41342        M50:    .ASCIZ <CR><LF>/OTHER/<TERM>
49 41353        M51:    .ASCIZ <CR><LF>/BASE #/<TERM>
50 41365        M52:    .ASCIZ <CR><LF>/PARALEL/<TERM>
51 41400        M53:    .ASCIZ /TIME-MS/
52 41410        M54:    .ASCIZ /SCORES/
53 41417        M55:    .ASCIZ <CR><LF>/MICROSECONDS PER SUBROUTINE CALL :/<TERM>
54 41465        M56:    .ASCIZ /UPDATES & RESTOR/
55 41506        M57:    .ASCIZ /  %INCR/
56 41516        M58:    .ASCIZ / %TOTAL/
57 41526        M59:    .ASCIZ /   BASE/
```

```
59 41536          M60:    .ASCIZ / SUBRTN/
59 41546          M61:    .ASCIZ / COMMUN/
60 41556          M62:    .ASCIZ /MSEC./
61 41564          M63:    .ASCIZ /   TIME/
62 41574          M64:    .ASCIZ /  /<SL>/WORD/
63 41604          M65:    .ASCIZ /MICRSEC/
64 41614          M66:    .ASCIZ / /
65 41616          M67:    .ASCIZ / EFFNCY/
66 41626          M68:    .ASCIZ /%TIME/
67 41634          M69:    .ASCIZ /      %/
68 41644          M70:    .ASCIZ <CR><LF>/%TIME/<TERM>
69 41655          M71:    .ASCIZ <CR><LF>/TIME/<SL>/SB/<TERM>
70 41670          M72:    .ASCIZ <CR><LF>/TOTAL/<TERM>
71 41701          M73:    .ASCIZ /ACCTD TIME/
72 41714          M74:    .ASCIZ <CR><LF>/WHITE/<TERM>
73 41725          M75:    .ASCIZ <CR><LF>/BLACK/<TERM>
74                        .EVEN
```

```
                .SBTTL VARIABLES
1736    TSTRG:
1736            .WORD STRG
1740            .WORD   0
1742    STRG:
1742            .WORD   0
                .BLKW   377
2742            .WORD   0
2744    STRGO:
                .BLKW 400
3744    STRGS:    .ASCIZ /INPOS/
3752          .ASCIZ /WHITE/
3760          .ASCIZ /BLACK/
3766          .ASCIZ /REL/
3772          .ASCIZ /SETBD/
4000             .ASCIZ /PIECE/
4006             .ASCIZ /SQUARE/
4015    CALPHA:      .ASCIZ /ALPHA/
4023    CBETA:     .ASCIZ /BETA/
4030    CMAXD:     .ASCIZ /MAXDEPTH/
4041    CMAXNIM:     .ASCIZ /MAXNIM/
4050          .ASCIZ /CLEAR/
4056          .ASCIZ /GO/
4061          .ASCIZ /END/
4065             .ASCIZ /BOARD/
4073             .ASCIZ /PDEPTH/
4102             .ASCIZ /PLAY/
4107             .ASCIZ /MOVENO/
4116    CMMAXD: .ASCIZ /MMAXD/
4124    CCDEPTH:         .ASCIZ /CDEPTH/ ;COMMON DEPTH
4133    CVARCD: .ASCIZ /VARCDEPTH/      ;COMMON DEPTH VARIABLE:
                         ;=0 FOR FIXED CDEPTH,1 FOR DECREASING CD
4145    CPEN:   .ASCIZ /PENALTY/        ;IRRELEVANT MOVE PENALTY
4155            .ASCIZ /MAXCD/  ;MAXIMUM COMMON DEPTH
4163            .ASCIZ /SWEEP/  ;PERFORM PARLLEL SEARCHES WITH VARYING DEPTHS
4171            .BYTE 0
                .EVEN
4172    LSTRGS:
4172            .WORD LINPOS,LWHITE,LBLACK,LREL,LSETBD,LPIECE,LSQUARE
4210            .WORD LALPHA,LBETA,LMAXDPTH,LMAXNIM
4220            .WORD LCLEAR,LGO,LEND,LBOARD,LPDPTH
4232            .WORD LPLAY,LMOVENO,LMMAXD,LCD,LVARCD,LPEN,LMAXCD
4250            .WORD LSWEEP
4252            .WORD LREST
4254    SETBD: .WORD 1
4256    SWPIECE:
4256          .WORD 1
4260    PLAY:   .WORD 0
4262    T1:     .BLKW 2 ;INITIALISED BY MINMAX
4266    T3:     .BLKW 2
4272    T4:     .BLKW 2
4276    TU:     .BLKW 2 ;TIME FOR 10000. UPDATES IN 1/60 SEC.
4302    TS:     .BLKW 2
4306    TC:     .BLKW 2
4312    TM:     .BLKW 2 ;TIME IN MINMAX
4316    DTT:    .BLKW 2 ;TOTAL TIME DIFFERENCE FROM THAT EXPECTED
4322    ADT:    .WORD 50.*60.*3,0       ;TIME LOST BEFORE DEPTH IS INCREASED
```

```
324         NADT:   .WORD -50.*60.*9.,-1    ;TIME BEHIND THAT EXPECTED
326         ST:     .BLKW 2 ;TIME IN SECS
336         SPT:    .BLKW 2 ;TIME REMAINDER - 1/50 THS OF A SEC
342         COLOUR:
342               .WORD 1
344         INPOS:
344                 .ASCII /SET PIECE BLACK  PA7 PB7 PC7 PD7 PE7 PF7 PG7 PH7 /
425                 .ASCII  /RA8 NB8 BC8 QD8 KE8 BF8 NG8 RH8 /
465                 .ASCII /WHITE PA2 PB2 PC2 PD2 PE2 PF2 PG2 PH2 /
533                 .ASCIZ  /RA1 NB1 BC1 QD1 KE1 BF1  NG1 RH1 /
                    .EVEN
576         MAXDPTH:.WORD 2
600         MAXNIM: .WORD 2,0
604         MMAXD:  .WORD 8.;ABSOLUTE MAXIMUM DEPTH
606         MOVENO: .WORD 0 ;NO OF PLYS FROM START
610         ALPHA:  .WORD -1000.
612         BETA:   .WORD 1000.
614         TOPPCE:
614                 .WORD   0,0
620         SCORE:
620                 .WORD   0
622         FIRREL:
622                 .WORD   0
624         TOPREL:
624                 .WORD   0
626         RELOFF: .WORD   PRESS+4 ;OFFSET IN MSTACK OF POS OF REL(P) OR REL(SQ)
632         SIX:    .WORD 6,0
634         SIXTY:  .WORD 60.,0
640         TEN:    .WORD 10.,0
644         HUNDRED:.WORD 100.,0
650         THOUSAND:.WORD 1000.,0
654         TENTHO: .WORD 10000.,0
                    .EVEN
            ;
            ;
            ;
```

```
1                       .SBTTL CHARACTER INPUT RECOGNITION
2                       .CSECT INPUT
3  000000       START:
4  000000       MOV  #FRONT,SP
5  000004       MOV  #ISR,@#PLVA
6  000012       MOV  #INTSW,@#PLVA+2
7  000020       MOV  #RINT,@#PLSR
8  000026       JSR  R5,IPTYPCLR
9  000032       JSR  R5,ISQUARE
10 00036        MOV  #1,SETBD
11 00044        MOV  #1,COLOUR
12 00052        MOV  #1,SWPIECE
13 00060        LOOP:
14              ;
15 00060             JSR  R5,SCAN
16                   ; MATCH THIS STRING WITH AN ELEMENT OF STRGS AND GOTO CODE
17 00064             MOV  #LSTRGS,R2
18 00070             MOV  #STRGS,R1
19 00074        1$:  TSTB (R1)
20 00076             BEQ  5$
21 00100                  MOV  #1,R3     ;MATCH=1
22 00104                  MOV  #STRG0,R0
23 00110        2$:       TSTB (R0)
24 00112                  BEQ  3$
25 00114                  TSTB (R1)
26 00116                  BEQ  3$
27 00120                       CMPB (R0)+,(R1)+
28 00122                       BEQ  2$
29 00124                            CLR  R3
30 00126                            BR   2$
31 00130        3$:       TSTB (R1)+     ;NEXT STRG
32 00132                  BNE  3$
33                        ;R1 PTS AFTER A ZERO,R0 PTS AT A ZERO
34 00134                  TST  R3
35 00136                  BEQ  4$
36 00140                       JMP  @(R2)  ;MATCH
37 00144        4$:       TST  (R2)+
38 00146                  BR   1$
39 00150        5$:  ;NO MATCH FOUND
40 00150             JMP  @(R2)
41 00154        LINPOS:
42                        ;STRG0 CONTAINS 'INPOS'
43 00154                  MOV     TSTRG,R1          ;STRG0=STRG
44 00160                  MOV     #STRG0,R2
45 00164        1$:       MOVB    (R1)+,(R2)+
46 00166                  BNE     1$
47 00170                  MOV     #STRG,R1          ;STRG=INPOS
48 00174                  MOV     #INPOS,R2
49 00200        2$:       MOVB    (R2)+,(R1)+
50 00202                  BNE     2$
51 00204                  MOVB    #BL,-(R1)         ;STRG=STRG & ' ' & STRG0
52 00210                  INC     R1
53 00212                  MOV     #STRG0,R2
54 00216        3$:       MOVB    (R2)+,(R1)+
55 00220                  BNE     3$
56 00222                  MOV     #STRG,TSTRG
57 00230                  BR      LOOP
```

```
5  00232        LWHITE: MOV     #1,COLOUR
5  00240                BR      LOOP
60 00242        LBLACK: MOV     #-1,COLOUR
61 00250                BR      LOOP
62 00252        LREL:   CLR     SETBD
63 00256                CLR     PLAY
64 00262                BR      LOOP
65 00264        LSETBD: MOV     #1,SETBD
66 00272                CLR     PLAY
67 00276                BR      LOOP
68 00300        LPIECE: MOV     #1,SWPIECE
69 00306                BR      LOOP
70 00310        LSQUARE:        CLR     SWPIECE
71 00314                BR      LOOP
72 00316        LCLEAR: JSR     R5,IPTYPCLR
73 00322                JSR     R5,ISQUARE
74 00326                CLR     SCORE
75 00332                CLR     TOPREL
76 00336                CLR     FIRREL
77                      ;INITIALISE TIMERS
78 00342                CLR     DTT
79 00346                CLR     DTT+2
80 00352                BR      LOOP
81 00354        LMAXDPTH:
82 00354                JSR     R5,SCANBIN;RETURNS STRGO
83 00360                MOV     R1,MAXDPTH
8  00364                BR      LOOP
85 00366        LMAXNIM:
86 00366                JSR     R5,SCANBIN
87 00372                MOV     R1,MAXNIM
88 00376                BR      LOOP
89 00400        LALPHA: JSR     R5,SCANBIN
90 00404                MOV     R1,ALPHA
91 00410                JMP     LOOP
92 00414        LBETA:  JSR     R5,SCANBIN
93 00420                MOV     R1,BETA
94 00424                JMP     LOOP
95 00430        LGO:
96 00430                TST     PLAY
97 00434                BEQ     1$
98 00436                        JMP   LREST1
99 00442        1$:
100 0442                MOV     @#SWREG,SWITCH
101 0450                JSR     PC,MINMAX
102 0454                JSR     R5,PBMVES;PRINT BEST MOVES
103 0460                JSR     R5,PCOUNTS ;PRINT COUNTERS
104 0464                JMP     LOOP
105 0470        LBOARD: JSR     R5,MAKEBD
106 0474                .PRINT #BOARD
107 0502                JMP     LOOP
108 0506        LPDPTH: JSR     R5,SCANBIN
1  0512                 MOV     R1,PDPTH
1  0516                 JMP     LOOP
111 0522        LPLAY:
112 0522                MOV     #1,PLAY
113 0530                JMP     LOOP
114 0534        LMMAXD:
```

```
1 0534                  JSR   R5,SCANBIN
1 0540                  MOV   R1,MMAXD
117 0544                JMP   LOOP
118 0550        LCD:
119 0550                JSR   R5,SCANBIN
120 0554                MOV   R1,CD0      ;COMMON DEPTH 0 : CD MAY BE ALTERED
121
122 0560                JMP LOOP
123 0564        LVARCD: ;COMMON DEPTH VARIABLE
124 0564                JSR   R5,SCANBIN
125 0570                MOV   R1,VARCD
126 0574                JMP   LOOP
127             ;
128 0600        LMOVENO:
129 0600                JSR   R5,SCANBIN
130 0604                ASL   R1
131 0606                TST   COLOUR
132 0612                BGT   1$
133 0614                   INC   MOVENO
134 0620        1$:
135 0620                MOV   R1,MOVENO
136 0624                JMP   LOOP
137 0630        LPEN:
138 0630                JSR   R5,SCANBIN
139 0634                MOV   R1,PEN
140 0640                JMP   LOOP
1 0644          LMAXCD: ;
142 0644                JSR   R5,SCANBIN
143 0650                MOV   R1,MAXCD
144 0654                JMP   LOOP
145 0660        LSWEEP: ;PARALLEL SEARCHES WITH VARYING CD UP TO MAXCD
146 0660                MOV   @#SWREG,SWITCH
147 0666                JSR   R5,SWEEP
148 0672                JSR   R5,PBMVES
149 0676                JMP   LOOP
150 0702        LEND:
151 0702                BIC   #RINT,@#PLSR
152 0710                .EXIT ;END PROGRAM
153             ;
154             ;
155             ;
156 0712        SCANBIN:
157 0712                JSR   R5,SCAN     ;RETURNS STRGO
158 0716                JSR   R5,BIN      ;BINARY FORM OF NUMBER RETURNED IN R1
159 0722                RTS   R5
160             ;
161             ;
162             ;
163 0724        LREST:  ;CHAR STRING NOT RECOGNISED SO
164 0724                TST   PLAY
165 0730                BNE   1$
1 0732                  JMP   LREST2
1 0736                  1$:
168                        ;LOOK FOR CMVE AND UPDATE BOARD
169                        ;GENERATE ALL LEGAL MOVES AND SEE IF ANY IDENTIFY
170                        ;WITH THE INPUT STRING
171                        ;SET R3,R4
```

```
0736                    MOV  #MSTACK-10.,R3
0742                    MOV  R3,R4
0744                    ADD  #PRESS,R4
0750                    JSR  PC,SETMVS  ;INPUT CHAR STRING SHOULD BE ONE
0754                    TST  KCAP
0760                    BEQ  2$
0762                         CLR  KCAP
0766                         SPRINT #M15     ;ILLEGAL,POSITION
1000                         JMP  LOOP
1004            2$:     ;R4->TOP OF MOVE STACK
1004                    TST  (R4)       ;MOVE ?
1006                    BEQ  8$
1010                         JSR  PC,UPDATE
1014                         JSR  PC,SETMVS  ;IS UPDATED MOVE LEGAL ?
1020                         TST  KCAP
1024                         BEQ  3$
1026                              CLR  KCAP
1032                              BR   6$
1034                    3$:
1034                         INC  REPEAT     ;CHECK ?
1040                         NEG  COLOUR
1044                         JSR  PC,SETMVS
1050                         NEG  COLOUR
1054                         DEC  REPEAT
                             ;UPDATED MOVE LEGAL - NOW FIND CHAR FORM
1060                         MOV  #PBUF,R0   ;RESULT IS IN PBUF
1064                         JSR  R5,CHARMV
1070                         MOV  #STRG0,R1  ;INPUT STRING
1074                    4$:
1074                         TSTB (R1)
1076                         BEQ  10$        ;SUCEED
1100                              TSTB (R0)
1102                              BEQ  6$ ;FAIL
1104                                   CMPB (R0)+,(R1)+
1106                                   BEQ  4$
1110                         6$:     ;FAIL TO MATCH OR ILLEGAL MOVE
1110                                   JSR  PC,RESTOR
1114                                   BR   2$ ;NXTMV
1116                    8$:     ;FAIL TO MATCH ANY MOVE
1116                         SPRINT #M16     ;MOVE NOT RECOGNISED
1130                         JMP  LOOP
1134            10$:    ;MOVE RECOGNISED & BOARD UPDATED AND COLOUR CHANGED
1134                    MOV  SCOR(R3),SCORE
1142                    INC  MOVENO
1146        LREST1: ;ENTRY POINT FOR STARTING WITH MACHINE MOVE
                    ;R3 & R4 WILL BE SET BY MINMAX
1146                    MOV  SCORE,BETA ;SET ALPHA, BETA FOR STRATEGIC SEARCH
1154                    ADD  PEN,BETA
1162                    MOV  SCORE,ALPHA
1170                    SUB  PEN,ALPHA
1176                    JSR  PC,MINMAX
1202                    JSR  R5,PBMVES
1206                    CMP  VSCOR(R3),ALPHA    ;ANY MOVES FOUND ?
1214                    BGT  18$        ;ALPHA<BETA ALWAYS
                             ;SCORE OUTSIDE ALPHA, BETA RANGE, SO TACTICAL
1216                         MOV  ALPHA,BETA
1224                         MOV  #-10000.,ALPHA
```

```
                        BR   22$
                18$:
                        CMP  VSCOR(R1),BETA
                        BLT   23$
                           MOV  BETA,ALPHA
                           MOV  #10000.,BETA
                22$:
                        DSTACK T2,T2+2  ;SAVE TIME OF FIRST CALL
                        JSR  PC,MINMAX
                        JSR  R5,PBMVES
                        DADD (SP)+,(SP)+,T2,T2+2        ;T2 IS NOW
                                                ;SEARCHES
            23$:
                INC  MOVENO
                MOV  MOVENO,R1
                ADD  #2,R1
                ASR  R1
                MOV  #PBUF,R0
                JSR  R5,DEC
                .WORD 10.
                MOVB #TERM,(R0)+
                SPRINT #PBUF
                TST  COLOUR
                BGT  25$
                        SPRINT #M17     ;'.....'
                        BR   ST26$
                25$:
                        SPRINT #M18     ;'.'
            ST26$:
                MOV  R3,R5
                ADD #BMV,R5
                MOV  -(R5),-(R4)
                MOV  -(R5),-(R4)
                MOV  -(R5),-(R4)
                JSR  PC,UPDATE
                NEG  COLOUR
                MOV  #1,REPEAT  ;
                JSR  PC,SETMVS  ;CHK ?
                CLR  REPEAT
                NEG  COLOUR
                MOV  #PBUF,R0
                JSR  R5,CHARMV
                SPRINT #PBUF
                MOV  SCOR(R3),SCORE
                ;ADJUST TIME
                ;CALL TO MINMAX WILL HAVE PUT ELAPSED TIME BEYOND T1 IN T
                DADD T2,T2+2,DTT,DTT+2
                DSUB ADT,ADT+2,DTT,DTT+2
                DCMP DTT,DTT+2,ADT,ADT+2,31$,30$
            30$:    ;TT-ETT GETTING LARGE= DECREASE DEPTH
                        DEC D
                        SPRINT  #M19    ;DECREASE DEPTH
                        BR   33$
            31$:
                DCMP DTT,DTT+2,NADT,NADT+2,32$,33$
            32$:    ;ETT-TT GETTING LARGE - CAN AFFORD TO INCREASE DEPTH
                        CMP  D,#8.      ;DON'T GO BETOND D=8
```

```
1652                                    BLE   33$
1654                                        SPRINT #M20     ;INCREASE DEPTH
1666                                        INC D
1672                    33$:
1672                            JMP   LOOP
1676            LREST2:
                        ; ASSUME INPUT STRING IS A PIECE OR SQAUARE DESCRIPTOR
1676                    TST   SWPIECE
1702                    BEQ   4$
                                ;SET RELEVANT PIECE
                                ;CALL INTERPRET PIECE
                                ;ON CALL TO INTPTP
                                ;R0->    PDESC.TYPE
                                ;2       DESC.ON
                                ;4       STRG0->
1704                            MOV   #STRG0,-(SP)
1710                            SUB   #4,SP
1714                            MOV   SP,R0
1716                            JSR   R5,INTPTP
1722                            TST   (R0)
1724                            BEQ   3$;ERROR
1726                                TST   SETBD
1732                                BEQ 1$
1734                                        JSR   R5,SETPIECE
1740                                        BR    2$
1742                                1$:
1742                                        MOV   @2(R0),2(R0) ; ANY PIECE ?
1750                                        BEQ   3$ ; INVALID
1752                                        ADD   #2,R0
1756                                        JSR   R5,SETRPSQ
1762                                        SUB   #2,R0
1766                            2$:     TST   (R0)
1770                                    BNE   7$
1772                            3$:     MOV   R0,-(SP)
1774                                        .PRINT #M3 ; INVALID PIECE DESCRIP
2002                                    MOV   (SP)+,R0                    OR
2004                            7$:     ADD   #6,SP
2010                                    JMP   LOOP
                                ; SET RELEVANT SQUARE
2014                    4$:     MOV   #STRG0,-(SP)
2020                            SUB   #2,SP
2024                            MOV   SP,R0
2026                            JSR   R5,SQCHAR
2032                            TST   (R0)
2034                            BNE   5$
2036                                MOV   R0,-(SP)
2040                                .PRINT #M4 ; INVALID SQUARE
2046                                MOV   (SP)+,R0
2050                                BR    6$
2052                            5$:     JSR   R5,SETRPSQ
2056                    6$:     ADD   #4,SP
2062                            JMP   LOOP
                ;
                ;
                ;
```

```
                        .SBTTL  INITIALISE PIECE TYPES AND COLOUR
                        .CSECT INITIALISE
                ;
                ;
                ;
000000          INPTYPCLR:
                ;NO ARGUMENTS
                ;DCL BSTRG BIT(32);
                ;PIECE='';BSTRG=1;
                ;DO;TOPPCE(*)=0;
                ; I=31 TO 0 BY -1;
                ;       IF I>=16 THEN PIECE(I).CLR=WHITE;
                ;       ELSE PIECE(I).CLR=BLACK;
                ;       PIECE(I).TYP=BSTRG;
                ;       BSTRG=BSTRG & '0'B;
                ;END
00000                   MOV   #TOPPCE,R1
00004                   CLR   (R1)+
00006                   CLR   (R1)+
00010                   MOV   #PIECE,R1 ; PIECE=''
00014                   MOV   #ENDP,R2
00020           1$:     CLR   -(R2)
00022                   CMP   R2,R1
00024                   BGT   1$
00026                   MOV   #1,R4 ; R4=BSTRG
00032                   MOV   #ENDP,R2
00036           2$:             SUB   #LENP,R2
00042                           MOV   R4,<PTYP+2>(R2)
00046                           INC   CLR(R2)
00052                           ASL   R4
00054                           BNE   2$
                                ;BLACK
00056                           MOV   #1,R4
00062                   3$:     SUB   #LENP,R2
00066                           MOV   R4,PTYP(R2)
00072                           DEC   CLR(R2)     ;BLACK=-1
00076                           ASL   R4
00100                           BNE   3$
00102                   RTS   R5
                ;
                ;
                ;
```

```
                        .SBTTL INITIALISE SQUARES
000104          ISQUARE:
                        ;CLEAR SQUARE
000104                  MOV   #SQUARE,R1 ; SQUARE=0
000110                  MOV   #ENDSQ,R2
000114          1$:     CLR   -(R2)
000116                  CMP   R2,R1
000120                  BNE   1$
000122                  MOV   #7,-(SP)
00126                   2$:     MOV   #7,-(SP)
00132                   3$:     TST   -(SP)
00134                           MOV   SP,R0
00136                           JSR   R5,SQCOLROW
00142                           MOV   (SP),R4 ; R4=SQ0
00144                           JSR   R5,CHARSQ
00150                           MOV   (SP),CSQ(R4)
00154                           MOV   R4,(SP)
00156                           MOV   #4,-(SP) ;DI=4
00162                           4$:     MOV   #4,-(SP) ; DJ=4
00166                           5$:     MOV   2(SP),R1 ; R1=DIRS(DI,DJ)
00172                                   MOL   #5,R1
00214                                   ADD   (SP),R1
00216                                   ASL   R1
00220                                   ADD   #DIRS,R1
00224                                   MOV   (R1),R1 ;R1=D
00226                                   BEQ   6$
                        ;
00230                                   ADD   4(SP),R1 ; R1=->SQUARE(SQ0).NEXTSQ(D)
00234                                   MOV   R1,R4
00236                                   MOV   8.(SP),-(SP) ; I+DI
00242                                   ADD   4(SP),(SP)
00246                                   SUB   #2,(SP)     ;I+DI-2
00252                                   MOV   8.(SP),-(SP) ; J+DJ
00256                                   ADD   4(SP),(SP)
00262                                   SUB   #2,(SP)     ;J+DJ-2
00266                                   TST   -(SP)
00270                                   MOV   SP,R0 ; SQ(1+DI,J+DJ)
00272                                   JSR   R5,SQCOLROW
00276                                   MOV   (SP)+,(R4) ; R1=SQ1
00300                                   CMP   (SP)+,(SP)+
00302                           6$:     DEC   (SP) ; DJ=DJ-1
00304                                   BGE   5$
00306                                   TST   (SP)+ ; POP DJ
00310                                   DEC   (SP)        ;DI=DI-1
00312                                   BGE   4$
00314                           CMP   (SP)+,(SP)+         ;POP DI & SQ0
00316                           DEC   (SP)        ;J=J-1
00320                           BGE   3$
00322                           TST   (SP)+       ;POP J
00324                           DEC   (SP)
00326                           BGE   2$ ;I=I-1
00330                   TST   (SP)+
00332                   RTS R5
                        ;
                        ;
                        ;
```

```
                        .SBTTL SQCOLROW
                ;
000334          SQCOLROW:
                        ;GIVEN R0 POINTING TO SQ, I, J RETURN
                        ; THE ADDRESS OF THE SQUARE IN (R0)
                        ; IF SQUARE IS ILLEGAL RETURN (R1)=0
                        ;(R0)-> SQ
                        ;2      J       COL     COUNTED FROM 0
                        ;4      I       ROW     COUNTED FROM 0
00334                   CLR   (R0)
00336                   MOV   4(R0),R1    ;R1=I=ROW
00342                   BLT   1$
00344                   CMP   R1,#7
00350                   BGT   1$
00352                   MOV   2(R0),R2    ;R2=J
00356                   BLT   1$
00360                   CMP   R2,#7
00364                   BGT   1$
                                ;VALID I & J
00366                           MUL   #8.,R1
00410                           ADD   R2,R1
00412                           MUL   #LENSQ,R1
00434                           ADD   #SQUARE,R1
00440                           MOV   R1,(R0)
00442           1$:     RTS   R5
                ;
                ;
                ;
                        .SBTTL CHARSQ
00444           CHARSQ:
                        ; GIVEN R0 POINTING TO CHAR(2),J,I, RETURN CHAR(2)
                        ;R0->   'A2'    CHAR(2)
                        ;2      J       COL
                        ;4      I       ROW
00444                   MOV   2(R0),R1    ;R1=COL=J
00450                   ADD   #'A,R1
00454                   MOVB  R1,(R0)     ;COL IS LOW ORDER BYTE
00456                   MOV   4(R0),R1    ;R1=ROW=I
00462                   ADD   #'1,R1
00466                   MOVB  R1,1(R0)
00472                   RTS   R5
                ;
                ;
                ;
```

```
                        .SBTTL  INTERPRET PIECE
00474           INTPTP:
                        ;GIVEN ->STRG0, RETURN TYPE AND ON OF PIECE
                        ; IF STRG IS NOT RECOGNISED THEN RETURN TYPE=0
                        ; ON CALL THE STACK IS AS FOLLOWS:
                        ;(SP)-> OLD PC
                        ;(R0)-> TYPE
                        ;2      ON
                        ;4      ->STRG0
                        ;
                        ;
00474                   CLR   (R0)          ; TYPE=0
00476                   MOV   4(R0),R1      ;R1=->STRG0
00502           1$:
00502                   TSTB  (R1)+
00504                   BNE   1$
00506                   DEC   R1 ;R1=->0
00510                   SUB   4(R0),R1      ;R1=LENGTH STRG0
00514                   CMP   R1,#2
00520                   BNE   3$
                ;
00522                       ADD   4(R0),R1
00526                       INC   R1
00530                       MOV   R1,R2
00532                       MOVB  -(R1),(R2) ; MOVE ZERO
00534                   2$:
00534                       MOVB  -(R1),-(R2)
00536                       BNE   2$
00540                       MOVB  #'P,(R2)    ;STRG0='P' & STRG0
00544                       BR    4$
00546           3$:
00546                   CMP   R1,#3
00552                   BEQ   4$
00554                   RTS   R5 ; ERROR
00556           4$:
00556                   MOV   4(R0),R1      ;R1=->STRG0
00562                   CMPB  (R1),#'P
00566                   BNE   8$
                        ;
00570                       TST   COLOUR
00574                       BGT   5$
00576                       CMPB  2(R1),#'7   ;BLACK AND ROW=7
00604                       BEQ   6$
00606                       BR    7$
00610                   5$:
00610                       CMPB  2(R1),#'2   ;WHITE AND ROW=2
00616                       BNE   7$
00620                       6$:
00620                           MOV   #UPAWN,(R0)
00624                           BR    21$
00626                       7$:
00626                           MOV   #PAWN,(R0)
00632                           BR    21$
00634           8$:
00634                   CMPB  (R1),#'N
00640                   BNE   9$
00642                       MOV   #NIGHT,(R0)
```

```
58 00646                        BR   21$
59 00650        9$:
60 00650                CMPB (R1),#'B
61 00654                BNE  10$
62 00656                   MOV  #BISHOP,(R0)
63 00662                   BR   21$
64 00664        10$:
65 00664                CMPB (R1),#'Q
66 00670                BNE  11$
67 00672                   MOV  #QUEEN,(R0)
68 00676                   BR   21$
69 00700        11$:
70 00700                CMPB (R1),#'R
71 00704                BNE  16$
72 00706                   TST  COLOUR
73 00712                   BGT  12$
74 00714                      CMPB 2(R1),#'8
75 00722                      BEQ  13$
76 00724                      BNE  15$
77 00726                   12$:
78 00726                      CMPB 2(R1),#'1
79 00734                      BEQ  13$
80 00736                      BNE  15$
81 00740                   13$:
82 00740                      CMPB 1(R1),#'A
83 00746                      BEQ  14$
84 00750                      CMPB 1(R1),#'H
85 00756                      BNE  15$
86 00760                      14$:
87 00760                         MOV  #UROOK,(R0)
88 00764                         BR   21$
89 00766                      15$:
90 00766                         MOV  #ROOK,(R0)
91 00772                         BR   21$
92 00774        16$:
93 00774                CMPB (R1),#'K
94 01000                BNE  20$
95 01002                   CMPB 1(R1),#'E
96 01010                   BNE  19$
97 01012                   TST  COLOUR
98 01016                   BLE  17$
99 01020                      CMPB 2(R1),#'1
100 1026                      BEQ  18$
101 1030                      BNE  19$
102 1032                   17$:
103 1032                      CMPB 2(R1),#'8
104 1040                      BNE  19$
105 1042                      18$:
106 1042                         MOV  #UKING,(R0)
107 1046                         BR   21$
108 1050                      19$:
109 1050                         MOV  #KING,(R0)
110 1054                         BR   21$
111 1056        20$:
112                     ;CHARACTER STRING NOT RECOGNISED
113 1056                RTS  R5
114 1060        21$:
```

```
115                     ;PREPARE FOR CALL TO SQCHAR
116                     ;(R0)-> ON
117                     ;2      ->STRGO (COL,ROW)
118 1060                INC  4(R0)       ;MOVE PAST PIECE CHARACTER
119 1064                MOV  R0,-(SP)
120 1066                ADD  #2,R0
121 1072                JSR  R5,SQCHAR
122 1076                MOV  (SP)+,R0
123 1100                TST  2(R0)
124 1104                BNE  20$
125 1106                   CLR  (R0)        ;INVALID SQUARE
126 1110                DEC  4(R0)
127 1114                RTS  R5
128             ;
129             ;
130             ;
```

```
                        .SBTTL SETPIECE ON BOARD
01116           SETPIECE:
                ;IF TYPE=KING ^ UKING THEN DO;
                ;       R1=0;
                ;       IF BLACK THEN R2=PIECE(0) ; ELSE R2=PIECE(16);
                ;       IF TOPPCE(COLOUR)^=0 THEN R3=PIECE(P2+1);ELSE R3=0;
                ;       END;
                ;ELSE DO;
                ;       R1=TOPPCE(COLOUR);
                ;       IF R1=0 THEN
                ;               IF BLACK THEN R2=PIECE(1);
                ;                       ELSE R2=PIECE(17);
                ;       ELSE R2=PIECE(P1+1);
                ;       R3=0;
                ;       END;
                ;IFR1^=0 THEN PIECE(P1).NXTP=P2;
                ;PIECE(P2).LSTP=P1;
                ;PIECE(P2).NXTP=P3;
                ;IF P3^=0 THEN PIECE(P3).LSTP=P2;
                ;IF P3=0 THEN TOPPCE(COLOUR)=P2;
                ;       ;OTHERWISE HAVE INSERTED P2 BEFORE P3 & TOPPCE IS UNALTEREI
                ;
                ;
                        ;(R0)-> TYPE
                        ;2      ON
01116                   CMP   (R0),#KING
01122                   BEQ    1$
01124                   CMP   (R0),#UKING
01130                   BNE   5$
01132                   1$:
01132                           CLR   R1
01134                           TST   COLOUR
01140                           BGT   2$
01142                           MOV   #PIECE,R2   ;R2=P2=PIECE(0)
01146                           TST   TOPPCE
01152                           BNE   3$
01154                           BR    4$
01156                   2$:
01156                           MOV   #<PIECE+<16.*LENP>>,R2      ;R2=P2=PIECE(16.)
01162                           TST   TOPPCE+2
01166                           BEQ   4$
01170                           3$:
01170                                   MOV   R2,R3
01172                                   ADD   #LENP,R3
01176                                   BR    11$
01200                           4$:
01200                                   CLR   R3
01202                                   BR    11$
01204                   5$:
01204                           TST   COLOUR
01210                           BGT   6$
01212                           MOV   TOPPCE,R1   ;BLACK
01216                           BNE   9$
01220                           MOV   #1,R2
01224                           BR    8$
01226                   6$:
01226                           MOV   TOPPCE+2,R1
```

```
                BNE   9$
        7$:
                MOV   #17.,R2
                8$:
                        MUL   #LENP,R2
                        ADD   #PIECE,R2
                        BR    10$
                9$:
                        ;ELSE P2=P1+1
                        MOV   R1,R2
                        ADD   #LENP,R2
        10$:
                CLR   R3
11$:
;R1=P1, R2=P2, R3=P3 ARE SET. NOW INSERT PIECE(P2).
        TST   R1
        BEQ   12$
                MOV   R2,NXTP(R1)
12$:
        MOV   R1,LSTP(R2)
        MOV   R3,NXTP(R2)
        TST   R3
        BEQ   50$
                ;INSERTING KING AT FRONT, WHEN THERE ARE OTHER PIECES
                ;LEAVE TOPPCE UNCHANGED
                MOV   R2,LSTP(R3)
                BR    14$
        50$:
                TST   COLOUR
                BGT   13$
                                MOV   R2,TOPPCE
                                BR    14$
                        13$:
                                MOV   R2,TOPPCE+2
        14$:
                ;R2=PIECE 2
        MOV   (R0),TYP(R2)          ;PIECE(P2).TYPE=PDESC.TYPE
        MOV   (R0),R1     ;PIECE(P2).DADDR=ADIRS(PTYPE)
        ADD   #ADIRS,R1
        MOV   (R1),DADD(R2)
        CMP   (R0),#PAWN ;IF PTYPE=PAWN OR UPAWN & COLOUR = WHITE THEN
                                ;PIECE(P2).DADDR=WPDIRS
        BEQ   15$
        CMP   (R0),#UPAWN
        BNE   16$
15$:
        TST   COLOUR
        BLE   16$
        MOV   #WPDIRS,DADD(R2)
16$:
        MOV   (R0),R1     ;IF COLOUR=WHITE THEN PIECE(P2).VALUE=VAL(PTYPE)
                          ;ELSE PIECE(P2).VALUE=-VAL(PT)
        ADD   #VALUES,R1
        MOV   (R1),VAL(R2)
        TST   COLOUR
        BGT   17$
        NEG   VAL(R2)
```

```
1444          17$:
1444                  ADD   VAL(R2),SCORE       ;SCORE=SCORE+PIECE(P2).VALUE
1452                  MOV   2(R0),(R2) ;PIECE(P2).ON=PDESC.ON
1456                  MOV   R2,@2(R0)
                                        ;SQUARE(PDEC.ON).PIECE=P2
1462                  CLR   REL(R2)     ;RP(P)=0
1466                  RTS   R5
                ;
                ;
                ;
```

```
                        .SBTTL SET A RELEVANT PIECE OR SQUARE
001470          SETRPSQ:
                        ;(R0)-> ON/PIECE
001470                  MOV   (R0),R2     ;R2=SQ OR PIECE
001472                  MOV   TOPREL,R1   ;R1=TOPREL
001476                  BNE   1$
                            ;INITIAL WHEN TOPREL=0
001500                      MOV   #PRESS+4,RELOFF
001506                      MOV   R2,FIRREL
01512                       BR    2$
01514                   1$:
01514                       MOV   R2,NXTR(R1)
01520           2$:
01520                   MOV   R1,LSTR(R2)
01524                   MOV   R2,TOPREL
01530                   SUB   #4,RELOFF   ;OBTAIN OFFSET FOR NEXTREL P OR SQ
01536                   MOV   RELOFF,REL(R2)
01544                   CLR   NXTR(R2)
01550                   RTS   R5
                ;
                ;
                ;
                ;
```

```
                        .SBTTL SQUARE FROM CHARACTER CONVERSION
001552          SQCHAR:
                        ;GIVEN COL, ROW FIND ON SQUARE OF PIECE
                        ;RETURN ON=0 IF INVALID
                        ;(R0)-> ON
                        ;2      ->(COL,ROW)     USUALLY ->STRGO
001552                  CLR   (R0)
001554                  MOV   2(R0),R1    ;R1=->COL
001560                  MOVB  1(R1),R2    ;R2=ROW
01564                   SUB   #'1,R2      ;R2=ROW-1
01570                   BLT   3$ ;INVALID ROW CHARACTER
01572                   CMP   R2,#7
01576                   BGT   3$
01600           1$:
01600                   MOL   #8.,R2
01622                   MOVB  (R1),R1     ;R1=COL
01624                   SUB   #'A,R1
01630                   BLT   3$ ;INVALID COLUMN CHARACTER
01632                   CMP   R1,#7
01636                   BGT   3$
01640           2$:
01640                   ADD R1,R2         ;R2=8*(ROW-1)+COL
01642                   MOL   #LENSQ,R2
01664                   ADD   #SQUARE,R2
01670                   MOV   R2,(R0)
01672           3$:
01672                   RTS   R5
                        ;
                ;
                ;
```

```
                .SBTTL SCAN : PUT THE NEXT WORD IN THE BUFFER IN STRGO
        SCAN:
                ;THE TT INPUT BUFFER IS SEARCHED FOR A CHARACTER STRING,
                ;WHICH IS RETURNED IN STRGO, TERMINATED BY A ZERO
                ; IF THE BUFFER IS EMPTY A PROMPTING MESSAGE IS DISPLAYED
                ; AND INPUT IS TAKEN FROM THE TELETYPE
                MOVB #BL,R0
                MOV  TSTRG,R1
                BR   2$
        1$:
                MOV  R0,-(SP)
                .PRINT #M1        ; EXPECTING INPUT
                MOV  (SP)+,R0
                MOV  #STRG,R1
                MOV R1,TSTRG
                JSR  R5,GET
        2$:
                JSR  R5,VERIFY   ;FIND FIRST NON BLANK. RETURNS (R2)->CHAR
                                 ;OR ZERO IF NOT FOUND
                TST  R2
                BEQ  1$
                ;CHARS HAVE BEEN INPUT
                MOV  R2,R1       ;REMOVE LEADING BLANKS
                MOV R2,TSTRG
                JSR R5,INDEX
                TST  R2
                BNE  3$
                    ;NO BLANK FOUND SO STRG IS TERMINATED BY A ZERO
                CLR  R0
        3$:
                    ;COPY STRING TO STRGO UNTIL A DELIMITER(IN (R0) ) IS FOUN
                MOV  #STRGO,R2
        4$:
                CMPB R0,(R1)
                BEQ  5$
                MOVB (R1)+,(R2)+
                BR   4$
        5$:
                CLRB (R2)        ;STRGO IS TERMINATED BY A ZERO
                MOV  R1,TSTRG    ;R1 PTS TO A BLANK OR A ZERO
                RTS  R5
        ;
        ;
        ;
```

```
                .SBTTL VERIFY
VERIFY:
                ;R1 PTS TO A STRG TERMINATED BY A ZERO
                ;R0 CONTAINS THE CHAR WHICH IS TO BE FOUND
                ;ON RETURN R2 PTS TO THE FIRST CHAR WHICH IS NOT THE TEST CHAR.
                ;IF NO COUNTER CHARACTER IS FOUND , (R2)=0
                ;R0 & R1 ARE UNCHANGED
                MOV  R1,-(SP)
                CLR  R2
1$:
                TSTB (R1)
                BEQ  2$
                CMPB (R1)+,R0
                BEQ  1$
                DEC  R1
                MOV  R1,R2
2$:
                MOV  (SP)+,R1
                RTS  R5
                ;
        ;
        ;
        ;
                .SBTTL   INDEX
INDEX:
                ;R0 CONTAINS A CHAR WHICH IS TO BE FOUNDIN THE STRG PTD TO BY R1
                ; ON RETURN R2 PTS TO THE FIRST OCCURRENCE OF THE CHAR
                ; IF IT IS NOT FOUND THEN R2=0
                MOV  R1,-(SP)
                CLR  R2
1$:
                TSTB (R1)
                BEQ  2$
                CMPB (R1)+,R0
                BNE  1$
                DEC  R1
                MOV  R1,R2
2$:
                MOV (SP)+,R1
                RTS  R5
        ;
        ;
        ;
```

```
                        .SBTTL GET
002060          GET:
                        ;ON CALL R1 PTS TO A BUFFER. ON RETURN R1 IS UNCHANGED.
                        ;READ FROM TT INTO BUFFER UNTIL A LF IS FOUND.
                        ;REPLACE THIS LINE FEED BY A ZERO.
                        ; REPLACE ANY PRECEDING CR BY A ZERO
002060                  BIS   #100,@#44  ;SET BIT 6 OF JSW FOR TTINR
002066                  MOV   R0,-(SP)
002070                  MOV   R1,-(SP)
                        ;SEE IF ANY PLINK CHARS INPUT
02072           5$:
02072                   CLRB (R1)       ;R1->RECEIVING BUFFER
02074                   JSR   R5,READPCH
02100                   TSTB (R1)
02102                   BNE   9$
02104                   .TTINR
02106                   BCS   5$
02110                           MOVB  R0,(R1)+
02112                   10$:
02112                           .TTYIN (R1)+
02120                           CMPB #LF,R0
02124                           BNE   10$
                                ;PUT A ZERO AT THE END OF THE STRG
02126                           CLRB -(R1)
02130                           CMPB -(R1),#CR
02134                           BNE   2$
02136                                   CLRB (R1)
02140                                   BR    3$
02142                           2$:
02142                                   INC   R1
02144                   3$:     ;R1 PTS TO 0 BYTE
02144                           CMP   R1,(SP)
02146                           BEQ   5$
02150                           MOV   (SP),R1
02152                           JSR   R5,PSEND   ;PARALLEL SEND
02156           9$:
02156                   MOV   (SP)+,R1
02160                   MOV   (SP)+,R0
02162                   RTS   R5
```

```
                        .SBTTL BINARY FROM DECIMAL CONVERSION
02164           BIN:
                        ;CONVERT STRG0 TO BINARY
                        ;RESULT OBTAINED IN R1
                        ;CARRY SET IF INVALID RESULT IS FOUND
                ;I=0;B=0
                ;       IF STRG(I)='+' THEN DO;I=I+1;CLR MINUS;END
                ;ELSE IF STRG(I)='-' THEN I=I+1;MINUS=1;
                ;DO WHILE (STRG(I)!=0)
                ;       X=CH(I)-'0';
                ;IF X<0 ! X>9 THEN DO; CALL ERROR;END;
                ;       B=(B*10)+X;
                ;       I=I+1;
                ;END;
                ;IF MIN THEN B=-B;
02164                   CLR   R4 ;R4=MINUS
02166                   MOV   #STRG0,R2   ;R2=->STRG0
02172                   CLR   R1
02174                   CMPB  (R2),#'+
02200                   BNE   1$
                ;
02202                   INC   R2
02204                   BR    2$
02206           1$:
02206                   CMPB  (R2),#'-
02212                   BNE   2$
02214                   MOV   #1,R4       ;MINUS=1
02220                   INC   R2
02222           2$:
02222                   TSTB  (R2)
02224                   BEQ   4$
02226                   MOVB  (R2),R3
02230                   SUB   #'0,R3
02234                   BLT   21$
02236                   CMP   R3,#9.
02242                   BLE   3$
02244                   21$:
02244                       MOV   R0,-(SP)
02246                       .PRINT #M2        ;NON NUMERIC INPUT
02254                       MOV   (SP)+,R0
02256                       SEC
02260                       RTS   R5
02262           3$:
                        ;MUL R1,#10
02262                   ADD   R1,R1
02264                   MOV   R1,-(SP)
02266                   ADD   R1,R1
02270                   ADD   R1,R1
02272                   ADD   (SP)+,R1
                        ;
02274                   ADD   R3,R1       ;R1=(R1*10)+(R3)
02276                   INC   R2
02300                   BR    2$
02302           4$:
02302                   TST   R4 ;MINUS ?
02304                   BEQ   5$
02306                   NEG   R1
```

```
59 02310        5$:
5[illegible]02310                CLC
60                               ;       ;RESULT IS IN R1
61 02312                         RTS  R5
```

```
            .SBTTL SWEEP: PERFORM 1 AND 2 PROCESSOR SEARCHES WITH VARYING CD
            .CSECT MAKEBD
SWEEP:
            ;PERFORM 1 PROCESSOR SEARCHES TO ESTABLISH A BASE TIME
            ;AND THEN PARALLEL SEARCHES WITH VARYING CD
            ;PRINT COUNTERS IN COLUMN FORMAT
            ;AND COMPARE TO BASE COUNT
            JSR  R5,PPIECE
                 .WORD M74,WKING,M75,BKING,0
            JSR  R5,PRNOS    ;PRINT SEARCH VARIABLES
                 .WORD 0,10.,CALPHA,ALPHA,CBETA,BETA
                 .WORD CMAXD,MAXDPTH,CMMAXD,MMAXD,0
            JSR  R5,TIMESUB;TIME SUBROUTINES(TU,TS,TC)
            .PRINT  #NEWLINE
            .PRINT #M48
            ;SINGLE PROCESSOR SEARCHES
            CLR  VARCD
            CLR  CD0
            JSR  R5,NLHEAD
                 .WORD CCDEPTH,M7,M8,M9,M54,M53,M21,M22,0
            CLR  SWITCH
            JSR  PC,MINMX1
            MOV  @#SWREG,SWITCH
            JSR  R5,SENDAV
                 .WORD NOUPS,ENDCO,BASE
            JSR  R5,SYNC
            DMOV  BASE+24.,BASE+28.;SAVE OTHER'S TIME
        ;FIND THEORETICAL TIME FOR 2 PROCESSORS
        ;DO BEFORE CONVERSION TO MILLISEC TO AVOID OFLOW
        ;(T2,BASE+24;BASE+24)
        ;TT=T1*T2/(T1+T2)
            JSR  R5,DMOLS
                 .WORD T2,BASE+24.,FR,0
            DMOV BASE+24.,FR+4
            JSR  R5,DADDNOS
                 .WORD T2,FR+4,0
            JSR  R5,DDIVR
                 .WORD FR+4,FR,BASE+24.,0
            JSR  R5,CNVMILL
                 .WORD T2,FR,BASE+28.,FR+4,BASE+24.,FR+8.,0
            .PRINT  #M49     ;THIS MACHINE
            JSR  R5,PRNO
                 .WORD 1,10.,CD0,NOUPS,NOSETMVS,NOCSTMVS,NOUPS
                 .WORD FR,NOMAXD,NOMMAXD,0
            .PRINT  #M50     ;OTHER MACHINE:
            JSR  R5,PRNO
                 .WORD 1,10.,CD0,BASE,BASE+4,BASE+8.
                 .WORD BASE,FR+4,BASE+12.,BASE+16.,0
        ;FIND BASE NOS
            JSR  R5,RULEOFF
            .PRINT  #M51     ;BASE NO.
            JSR  R5,PRNO
                 .WORD 1,10.,CD0,BASE,BASE+4,BASE+8.
                 .WORD BASE,FR+8.,BASE+12.,BASE+16.,0
        ;FIND WORK DONE(TU,TS,TC;BASE+28.)
            JSR  R5,DMOLS
                 .WORD BASE,TU,FR+12.
```

```
5F 0454                         .WORD BASE+4,TS,FR+16.
5[illegible]0462                .WORD BASE+8.,TC,FR+20.,0
60 00472                JSR  R5,ACDDVR  ;MICRO->MILLISECS.
61 00476                        .WORD 3,TENTHO,FR+12.,FR
62 00506                DCLR BASE+28.
63 00516                JSR  R5,DADDNOS
64 00522                        .WORD FR,BASE+28.
65 00526                        .WORD FR+4,BASE+28.
66 00532                        .WORD FR+8.,BASE+28.,0
67                      ;FIND TIME FOR TREE UPDATES PER SEC.
68                      ;(T2-[ACCOUNTED TIME])*10000./NOUPS
69 00540                DMOV  T2,TM
70 00554                DSUB BASE+28.,BASE+30.,TM,TM+2  ;ACCOUNTED TIME
71 00574                DMOV TM,FR+12.  ;MINMAX TIME MILLISECS
72 00610                JSR  R5,DMOLS;MILLI->MICRO
73 00614                        .WORD TENTHO,TM,FR+16.,0
74 00624                JSR  R5,DDIVR
75 00630                        .WORD NOUPS,FR+16.,TM,0
76 00640                .PRINT #M71     ;TIME/SUB
77 00646                JSR  R5,CNVMIC
78 00652                        .WORD TU,FR+16.,TS,FR+20.
79 00662                        .WORD TC,FR+24.,TM,FR+28.,0
80 00674                JSR  R5,PRNO
81 00700                        .WORD 1,10.,CDO,FR+16.,FR+20.,FR+24.,FR+28.,0
82 00720                .PRINT #M72
83 00726                JSR  R5,CNVMIL
[illegible]0732                 .WORD FR,FR+16.,FR+4,FR+20.
85 00742                        .WORD FR+8.,FR+24.,FR+12.,FR+28.,0
86 00754                JSR  R5,PRNO
87 00760                        .WORD 1,10.,CDO,FR+16.,FR+20.,FR+24.,FR+28.,0
88                      ;FIND %TIME FOR SUBROUTINES
89 01000                JSR  R5,ACDMOLS
90 01004                        .WORD 4,HUNDRED,FR,FR+16.
91 01014                JSR  R5,ACDDVR
92 01020                        .WORD 4,T2,FR+16.,FR
93 01030                .PRINT #M70     ;%TIME
94 01036                JSR  R5,PRNO
95 01042                        .WORD 1,10.,CDO,FR,FR+4,FR+8.,FR+12.,0
96 01062                DMOV T2,BASE+28.
```

SWEEP: PERFORM 1 AND 2 PROCESSOR SEARCHES WITH VARYING CD

```
 001076                 INC  VARCD
 001102                 .PRINT #NEWLINE
 001110                 .PRINT #NEWLINE
 001116                 .PRINT #M27
 001124                 JSR  R5,NLHEAD
 001130                 .WORD CCDEPTH,M57,M57,M57,M65,M67,M67,M67,0
 001152                 JSR  R5,NLHEAD
 001156                 .WORD M66,M60,M61,M63,M64,M58,M57,M69,0
 001200                 JSR  R5,NLHEAD
 01204                  .WORD M66,M59,M59,M59,M61,M59,M60,M61,0
 01226          PRCO16$:
 01226          16$:
 01226                  JSR  R5,RULEOFF
 01232                  JSR  R5,PRNOS
 01236                  .WORD 0,10.,CVARCD,VARCD,0
 01250                  CLR  CDO
 01254          17$:
 01254                  .PRINT  #NEWLINE
 01262                  CMP  CDO,MAXCD
 01270                  BGT  50$
 01272                       JSR  R5,SYNC
 01276                       JSR  PC,MINMX1
                             ;JSR  R5,PCOUNTS
 01302                       JSR  R5,SENDAV
 01306                            .WORD NOUPS,ENDCO,PAR
 01314                       JSR  R5,SYNC
 01320                       JSR  R5,PERCENTS
 01324                       INC  CDO
 01330                       BR   17$
 01332          50$:
 01332                  DEC VARCD
 01336                  BGE  16$
 01340          60$:
 01340                  RTS  R5
                ;
                ;
                ;
```

```
                        .SBTTL PERCENTS:FIND % OVERHEADS
001342          PERCENTS:
001342                  JSR  R5,DADDNOS
001346                       .WORD NOUPS,PAR,NOSETMVS,PAR+4,NOCSTMVS,PAR+8.
001362                       .WORD NOMAXD,PAR+12.,NOMMAXD,PAR+16.,NBP,PAR+20.
001376                       .WORD 0
                        ;FIND WORK DONE
001400                  JSR  R5,DMOLS
001404                       .WORD PAR,TU,FR
01412                        .WORD PAR+4,TS,FR+4
01420                        .WORD PAR+8.,TC,FR+8.
01426                        .WORD PAR,TM,FR+12.,0
01436                   JSR  R5,ACDDVR  ;MICRO->MILLISEC.
01442                   .WORD 4,TENTHO,FR,FR+16.
                        ;FIND ACCOUNTED TIME
01452                   DCLR PAR+28.
01462                   JSR  R5,DADDNOS
01466                        .WORD FR+16.,PAR+28.
01472                        .WORD FR+20.,PAR+28.
01476                        .WORD FR+24.,PAR+28.
01502                        .WORD FR+28.,PAR+28.,0
                        ;CONVERT TO COMBINED TIMING
01510                   JSR  R5,DMOLS
01514                        .WORD BASE+24.,PAR+28.,FR,0
01524                   JSR  R5,DDIVR
01530                        .WORD BASE+28.,FR,PAR+28.,0
                        ;JSR  R5,PRNOS
                        ;       .WORD 1,10.,M73,PAR+28.,M63,T2,0
01540                   DMOV  PAR+28.,FR+4      ;TREESIZE
01554                   DMOV  T2,FR+8.  ;COMMUNICATION TIME:TOTAL TIME-TREESIZE TIM
01570                   DSUB  PAR+28.,PAR+30.,FR+8.,FR+10.
01610                   DMOV  T2,FR+12. ;TOTAL TIME
                        ;COPY UP FR.
01624                   DMOV BASE+24.,FR+16.;BASE TIME
01640                   DMOV PAR+28.,FR+20.;EXTRA TREESIZE
01654                   DSUB BASE+24.,BASE+26.,FR+20.,FR+22.
01674                   DMOV FR+8.,FR+24.       ;COMMUN TIME
                        ;COMMUNICATION TIME PER WORD
                        ;MICROSECS/WORD(WDSIN,WDSOUT,COMM TIME;MICROSEC/WORD)
01710                   JSR  R5,DMOLS   ;MICROSECONDS
01714                        .WORD TENTHO,FR+8.,FR+104.,0
01724                   DMOV WDSI,FR+100.
01740                   DADD WDSO,WDSO+2,FR+100.,FR+102.
01760                   JSR  R5,DDIVR
01764                        .WORD FR+100.,FR+104.,FR+108.,0
01774                   JSR  R5,CNVMIC
02000                        .WORD FR+108.,FR+112.,0
                        ;
                        ;FIND % INCREASE ON BASE
02006                   JSR  R5,DSUBNOS ;TIME-BASE
02012                        .WORD BASE+24.,FR+4     ;TREE TIME - BASE TIME
02016                        .WORD BASE+24.,FR+12.,0;T2 - BASE TIME
                        ;FIND % TIME
02024                   JSR  R5,ACDMOLS
02030                        .WORD 8.,HUNDRED,FR,FR+32.
02040                   JSR  R5,ACDDVR
02044                        .WORD 4,BASE+24.,FR+32.,FR
```

```
02054                   JSR   R5,ACDDVR
02060                         .WORD 4,T2,FR+48.,FR+16.
02070                   .PRINT #NEWLINE
02076                   JSR   R5,PRNO
02102                         .WORD 1,10.,CD0,FR+4,FR+8.,FR+12.
02116                         .WORD FR+112.,FR+16.,FR+20.,FR+24.,0
02130                   RTS   R5
                ;
                ;
                ;
```

```
;TIME SUBROUTINE FOR UPDATE,RESTOR,SETMVS & CSETMVS

            .SBTTL TIMESUB:TIME SUBROUTINE FOR UPDATE,RESTOR,SETMVS & CSETMVS
TIMESUB::;TIME SUBROUTINES
            .MACRO TIMEMVS SUBR,TX,?L1,?L2
            JSR   PC,IMVSTACK
            MOV   R4,-(SP)
            MOV   #5000.,-(SP)
            STIME  T1,T1+2
L1:
            MOV   2(SP),R4
            JSR   PC,SUBR
            DEC   (SP)
            BGE   L1
            STIME TX,TX+2
            DSUB T1,T1+2,TX,TX+2
            DMOL2 TX,TX+2,L2
L2:
            CMP(SP)+,(SP)+
            .ENDM
;
            SAVREG 5
            TIMEMVS CSTMVS,TC
TIMV0$:
            TIMEMVS SETMVS,TS
;
            MOV   R4,-(SP)
            MOV   #10000.,-(SP)
            STIME  T1,T1+2
TIMV1$:
10$:
            MOV   #MSTACK-6000,R1
            MOV   2(SP),R0
            MOV   R1,R4
20$:
            MOV   (R0)+,(R1)+
            BEQ   30$
                  MOV   (R0)+,(R1)+
                  MOV   (R0)+,(R1)+
                  BR    20$
            30$:
                  JSR   PC,UPDATE
                  JSR   PC,RESTOR
                  DEC   (SP)
                  BLE   50$
                  TST   (R4)
                  BNE   30$
                  BR    10$
50$:
            STIME TU,TU+2
            DSUB T1,T1+2,TU,TU+2
            CMP   (SP)+,(SP)+
            ;.PRINT #M55;MICROSECS PER SUBROUTINE CALL
            ;JSR   R5,PRNOS
            ;        .WORD 1,10.,M7,TU,M8,TS,M9,TC,0
            RESREG 5
            RTS   R5
;
;
```

TIMESUB:TIME SUBROUTINE FOR UPDATE,RESTOR,SETMVS & CSETMVS

;P ;

```
: PRINT COUNTERS

                .SBTTL PCOUNTS: PRINT COUNTERS
PCOUNTS:
                ;PRINT COUNTERS
                JSR   R5,PRNOS
                .WORD 0,10.,CALPHA,ALPHA,CBETA,BETA,CMAXD,MAXDPTH,CMMAXD,MMAXD,0
                JSR   R5,PRNOS
                .WORD 0,10.,CCDEPTH,CDO,M24,NOMINMX2,0
                TST   FIRREL
                BEQ   10$
                        JSR   R5,PRNOS
                        .WORD 1,10.,CMAXNIM,MAXNIM,M23,NORCUT,CPEN,PEN,0
10$:
                JSR   R5,PRNOS
                .WORD 1,10.,M7,NOUPS,M8,NOSETMVS,M9,NOCSTMVS
                .WORD M21,NOMAXD,M22,NOMMAXD,M38,NBP,0
PCO20$:
                TST   T2
                BNE   20$
                TST   T2+2
                BEQ   50$
                20$:
                        ;PRINT COUNTS PER SEC
                        JSR   R5,ACDMOLS
                        .WORD 6,SIXTY,NOUPS,FR
                        JSR   R5,ACDDVR
                        .WORD 6,T2,FR,FR+30.
                        JSR   R5,PRNOS
                        .WORD 1,10.,M40,FR+30.,M41,FR+34.,M42,FR+38.
                        .WORD M43,FR+42.,M44,FR+46.,M45,FR+50.,0
                        ;PRINT SEARCH TIME
                        JSR   R5,DDIVS
                        .WORD SIXTY,T2,ST,SPT,0
                        JSR   R5,PRNOS
                        .WORD 1,10.,M10,ST,M11,SPT,0
                        SPRINT  #M12
                        BIT   #1,@#SWREG ;PARALLEL MODE ?
                        BEQ   50$
                                ;BUSY TIME
                                JSR   R5,DDIVS
                                .WORD  SIXTY,T3,FR,FR+4.,0
                                ;% BUSY TIME
                                JSR   R5,DMOLS
                                .WORD HUNDRED,T3,FR+8.,0
                                JSR   R5,DDIVS
                                .WORD T2,FR+8.,FR+12.,FR+16.,0
                                JSR   R5,PRNOS
                                .WORD 1,10.,M31,FR,M11,FR+4.,M32,FR+12.,0
50$:
PCO55$:
                BIT   #1,@#SWREG
                BEQ   60$
                        JSR   R5,PRNOS
                        .WORD 1,10.,M35,NORINT,M36,WDSI,M37,WDSO,0
                        DCLR NORINT
60$:
                RTS   R5
;
```

.COUNTS: PRINT COUNTERS

;P ;
; ;

```
                        .SBTTL PBMVES: PRINT BEST MOVES FOUND BY MINIMAX.
003506          PBMVES:
                ;BOARD ASSUMED AT D=0
                ;ASSUME R4=TOP OF MSTACK,
                ;R3=UPM(D=0).
003506                  MOV   #PBUF,R0
003512                  SPRINT #M6        ;BESTLINE
003524                  MOV   R5,-(SP)
003526                  INC   REPEAT      ;CALLS TO SETMVS WILL FIND CHECKS
03532                   CLR   LASTD
03536                   MOV   R3,R5       ;FIND BMV(R3)
03540                   ADD   #BMV,R5
03544           PB1$:   MOV   -(R5),-(R4)
03546                   BEQ   4$
03550                           MOV   -(R5),-(R4)
03552                           MOV   -(R5),-(R4)
                        ;R3=LUM, R4=TOP OF MSTACK(FROM OF MOVE).
03554                           MOV   R5,-(SP)
03556                           JSR   PC,UPDATE
03562                           NEG   COLOUR
03566                           JSR   PC,SETMVS   ;FIND IF CHECK
03572                           NEG   COLOUR
03576                           MOV   (SP)+,R5
03600                           MOV   #PBUF,R0
03604                           JSR   R5,CHARMV
03610                           JSR   R5,PRINTMV
03614                           BR    PB1$
03616           4$:
03616           5$:     TST   D
03622                   BLE   6$
03624                           JSR   PC,RESTOR
03630                           BR    5$
03632           6$:     ;D=0 & BOARD IS RESTORED TO ORIGIN.
03632                   JSR   R5,PVSCOR
03636                   SPRINT #NEWLINE
03650                   CLR   REPEAT
03654                   MOV   (SP)+,R5
03656                   RTS   R5
                        .EVEN
                ;
                ;
                ;
```

```
            .SBTTL MAKEBD: MAKE UP CHARACTER FORM OF BOARD
MAKEBD:
;MAKE A CHARACTER REPRESENTATION OF THE BOARD SUITABLE
;FOR PRINTING.
        MOV   R2,-(SP)
        MOV   R1,-(SP)
        MOV   #7,-(SP)    ;DO I=7 TO 0
1$:             MOV   #7,-(SP)    ;DO J=7 TO 0
        2$:             CLR   -(SP)
                        MOV   SP,R0
                        JSR R5,BSQCOLROW
                        MOV   (R0),-(SP)
                        JSR   R5,SQCOLROW         ;RETURNS (R0)=SQ
                        MOV   (SP)+,R2    ;R2=->CHAR SLOT
                        MOV   @(R0),R1    ;R1=P1
                        BNE   3$
                                ;NO PIECE
                                MOV   R2,R1
                                MOVB -(R1),(R2)+ ;COPY PRECEDING CHAR
                                MOVB (R1),(R2)
                                BR    6$
                        3$:     ;PIECE
                                TST   CLR(R1)
                                BGT   4$
                                        MOVB #'*,(R2)+
                                        BR    5$
                                4$:     MOVB #BL,(R2)+
                        5$:     MOV   TYP(R1),R1
                                ASR   R1
                                ADD   #PTABLE,R1
                                MOVB (R1),(R2)
                    6$:     TST   (SP)+
                    DEC   (SP)        ;J=J-1
                    BGE   2$
                    TST   (SP)+       ;POP J
        DEC   (SP)
        BGE   1$
        TST   (SP)+       ;POP I
        MOV   (SP)+,R1
        MOV   (SP)+,R2
        RTS   R5
;
;
;
```

FIND BOARD OFFSET GIVEN COL AND ROW

```
                        .SBTTL FIND BOARD OFFSET GIVEN COL AND ROW
                        .CSECT BSQCOLROW
000000          BSQCOLROW:
                ;GIVEN THE COL AND ROW OF A SQ THE SUBROUTINE RETURNS
                ;A POINTER TO THE POSITION IN THE CHARACTER BOARD WHEREIN
                ;TWO CHARACTERS DESCRIBING A PIECE SHOULD BE PLACED.
                ;ON CALL         (R0)->  BSQ     RETURNED VALUE
                ;                2       J       COL
                ;                4       I       ROW
                ;OFFSET=BOARD+34*(1+3(7-I))+(1+4*J)
00000                   MOV   R1,-(SP)
00002                   MOV   #BOARD,-(SP)      ;USE (SP) AS ACCUMULATOR
00006                   MOV   #7,R1             ;WANT R1=34*(1+3(7-I))
00012                   SUB   4(R0),R1
00016                   MUL   #3,R1
00040                   INC   R1
00042                   MUL   #34.,R1
00064                   ADD   R1,(SP)
                ;
00066                   MOV   2(R0),R1          ;WANT R1=4*J+1
00072                   MUL   #4,R1
00114                   INC   R1
00116                   ADD   R1,(SP)
00120                   MOV   (SP)+,(R0)
00122                   MOV   (SP)+,R1
00124                   RTS   R5
                ;
                ;
                ;
00126           BOARD:
                        .REPT 4
                                .REPT 3
                                        .REPT 4
                                                .ASCII /    ..../
                                        .ENDR
                                        .ASCII <CR><LF>
                                .ENDR
                                .REPT 3
                                        .REPT 4
                                                .ASCII /....    /
                                        .ENDR
                                        .ASCII <CR><LF>
                                .ENDR
                        .ENDR
01606                   .ASCII <CR><LF>/  A   B   C   D   E   F   G   H /
01650                   .WORD 0
                ;
                ;
                ;
```

```
                        .SBTTL CHARACTER FORM OF UPDATED MOVE
                        .CSECT CHARMV
000000          CHARMV:
                ;PLACE THE CHARACTER FORM OF THE UPDATED MOVE IN THE BUFFER
                ;R0->BUFFER
                ;R3=->UPM
                ;
000000                  MOV   R0,-(SP)    ;SAVE BUFFER ADDRESS
000002                  MOV   MT(R3),R2   ;FIND CHARPIECE(MT)
00006                   ASR   R2
00010                   ADD   #PTABLE,R2
00014                   MOVB  (R2),(R0)+
                        ;FIND CSQ(SQ0)
00016                   MOV   SQ0(R3),R2 ;R1=SQ0
00022                   MOVB  CSQ(R2),(R0)+
00026                   MOVB  <CSQ+1>(R2),(R0)+
                        ;FIND TYPE OF CAPTURED PIECE
00032                   MOV   CPCE(R3),R2         ;R2=P1
00036                   BNE   1$
                            ;NO PIECE CAPTURED
00040                       MOVB  #'-,(R0)+
00044                       BR    2$
00046           1$:         ;PIECE CAPTURED
00046                       MOVB  #'X,(R0)+
00052                       MOV   TYP(R2),R2
00056                       ASR   R2
00060                       ADD   #PTABLE,R2
00064                       MOVB  (R2)+,(R0)+
00066           2$:     ;FIND CSQ(SQ1)
00066                   MOV   SQ1(R3),R2
00072                   MOVB  CSQ(R2),(R0)+
00076                   MOVB  <CSQ+1>(R2),(R0)+
                        ;SPECIAL MOVE ?
00102                   CMP   MT(R3),#THR
00110                   BLT   9$
00112                       CMP   MT(R3),#CASTLE
00120                       BNE   4$
00122                           MOV   (SP),R0     ;CLEAR BUFFER AND RESTART
00124                           CMP   SQ0(R3),SQ1(R3)
00132                           BHI   3$
00134                               MOV   #CKCASTLE,R2
00140                               BR    8$
00142                           3$:     MOV   #CQCASTLE,R2
00146                               BR    8$
00150                   4$:     CMP   MT(R3),#EPCAP
00156                           BNE   5$
00160                               MOV   #CHAREP,R2
00164                               BR    8$
00166                   5$:     CMP   MT(R3),#QPAWN
00174                           BNE   6$
00176                               MOV   #CQPAWN,R2
00202                               BR    8$
00204                   6$:     CMP   MT(R3),#NPAWN
00212                           BNE   7$
00214                               MOV   #CNPAWN,R2
00220                               BR    8$
00222                   7$:     MOV   #M5,R2      ;ERROR ON MT PARAMETER
```

```
                                ;
00226                   8$:     MOVB (R2)+,(R0)+        ;MOVE OVER CHARS
00230                           BNE  8$
00232                           DEC  R0 ;REMOVE 0 BYTE
00234           9$:     TSTB CHK(R3)
00240                   BEQ  10$
00242                           MOVB #'+,(R0)+
00246           10$:
00246                   MOVB #<TERM>,(R0)+      ;MOVE KEEPS SAME LINE
00252                   CLRB (R0)+      ;0 END MARKER
00254                   MOV  (SP)+,R0
00256                   RTS  R5
                ;
                ;
                ;
00260           CKCASTLE:
00260                   .ASCIZ /0-0/
00264           CQCASTLE:
00264                   .ASCIZ /0-0-0/
00272           CHAREP: .ASCIZ /E.P./
00277           CQPAWN: .ASCIZ /=Q/
00302           CNPAWN: .ASCIZ /=N/
00305           M5:     .ASCIZ /ERROR ON MT PARAM./
00330           PTABLE: .ASCII /??NBRQKPR?KP????PPP/
                        .EVEN
                ;
                ;
                ;
```

```
                        .SBTTL PRINT STRING AT TAB(D-1)
                        .CSECT PRINTMV
000000          PRINTMV:
                ;PRINT STRING ON TELETYPE WITH D-1 PRECEDING TABS.
                ;ON CALL R0->STRING TO BE PRINTED
                        TB=11
                        TAB=11
                        TERM=200
000000                  MOV  R1,-(SP)
                        ;TAB TO CORRECT COLUMN
00002                   TST  LASTD      ;FIRST TIME ?
00006                   BEQ  2$
00010                   CMP  D,LASTD
00016                   BLE  2$
00020                           SPRINT #TABC
00032                           INC  NTAB
00036                           CMP  NTAB,#8.
00044                           BLT  4$
00046                                   SPRINT #NEWLINE
00060                           1$:     CLR  NTAB
00064                                   BR   4$
00066                   2$:     MOV  D,R1       ;1=D-1
00072                           SPRINT  #NEWLINE
00104                           CLR  NTAB
00110                           CMP  R1,#8.
00114                           BLT  3$
00116                                   SUB  #8.,R1
00122                   3$:     DEC  R1
00124                           BLE  4$
00126                                   SPRINT #TABC
00140                                   INC NTAB
00144                                   BR   3$
00146           4$:     ;NOW PRINT STRING
00146                   SPRINT R0
00156                   MOV  D,LASTD
00164                   MOV  (SP)+,R1
00166                   RTS  R5
                ;
00170           TABC:   .ASCII <TB><TERM>
00172           NEWLINE:
00172                   .ASCII <CR><LF><TERM>
                        .EVEN
00176           NTAB:   .WORD 0
                ;
                ;
                ;
```

```
                        .SBTTL PRINT VSCOR
                        .CSECT PVSCOR
000000          PVSCOR:
                ;PRINT THE CURRENT VSCOR IN BRACKETS AS A DECIMAL NUMBER
                ;R3=->UPM
                ;
000000                  MOV   R1,-(SP)
000002                  MOV   R0,-(SP)
000004                  MOV   #PBUF,R0
00010                   MOVB  #'(,(R0)+
00014                   MOV   VSCOR(R3),R1
00020                   JSR   R5,DEC
00024                   .WORD 10.
00026           1$:
00026                   TSTB  (R0)+       ;FIND END OF STRING
00030                   BNE   1$
00032                   DEC   R0
00034                   MOVB  #'),(R0)+
00040                   MOVB  #<TERM>,(R0)+       ;SAME LINE
00044                   SPRINT #PBUF
00056                   MOV   (SP)+,R0
00060                   MOV   (SP)+,R1
00062                   RTS   R5
                ;
                ;
                ;
```

```
                        .SBTTL PPIECE: PRINT PIECES
                        ;GROUPS OF TWO ARGUMENTS
                        ;FIRST ARG IS CHAR FORM OF COLOUR
                        ;2ND IS HEAD OF PIECE LIST
000064          PPIECE:
000064                  SAVREG 0,1,2,3
000074                  SUB  #50.,SP
000100          10$:
                        ;FOR WHITE AND BLACK PIECES
00100                   MOV  (R5)+,R0
00102                   BEQ  50$
00104                       .PRINT ;WHITE, BLACK
00106                       MOV  @(R5)+,R2  ;PIECE HEADER
00110                       20$:
                                ;FOR ALL PIECES OF THIS COLOUR DO;
00110                           MOV  SP,R1
00112                           MOVB #BL,(R1)+
00116                           MOV  TYP(R2),R3
00122                           ASR  R3
00124                           MOVB PTABLE(R3),(R1)+
00130                           MOV  (R2),R3
00132                           MOVB CSQ(R3),(R1)+
00136                           MOVB CSQ+1(R3),(R1)+
00142                           MOVB  #TERM,(R1)+
00146                           MOV  SP,R0
00150                           .PRINT
00152                           MOV  NXTP(R2),R2
00156                           BNE  20$
00160                       BR  10$
00162           50$:
00162                   ADD  #50.,SP
00166                   RESREG 0,1,2,3
00176                   RTS  R5
                ;
                ;
                ;
```

```
                        .SBTTL PRNO:PRINT ARRAY OF NUMBERS
000200          PRNO:
                        ;PRINT NOS WITHOUT DESCRIPTORS
                        ;NOS ARE PRECEDED BY A TAB & ARE RIGHT JUSTIFIED TO 7 CHARS
                        ;LARGER NOS ARE NOT TRUNCATED BUT OFLOW 7 CHAR FIELD
000200                  SAVREG 0,1,2,3
000210                  SUB   #50.,SP
000214                  MOV   SP,R2
000216                  ADD   #10.,R2
00222                   MOV   (R5)+,R3    ;BIT INDICATOR FOR SINGLE , DOUBLE PRECISIC
00224                   MOV   (R5)+,25$   ;BASE
00230           10$:
00230                   MOV   (R5)+,R1
00232                   BEQ   50$
00234                           MOV   R2,R0
00236                           TST   R3
00240                           BGT   30$
00242                                   MOV   (R1),R1     ;R1=NO
00244                                   JSR   R5,DEC
00250                           25$:    .WORD 10.
00252                                   BR    35$
00254                           30$:
00254                                   JSR   R5,DDEC     ;R1->NO
00260                   35$:
00260                           MOVB  #TERM,(R0)
00264                           SUB   #7,R0
00270                           MOV   R2,R1
00272                   40$:
00272                           CMP   R1,R0
00274                           BLOS  45$
00276                                   MOVB #BL,-(R1)
00302                                   BR    40$
00304                   45$:
00304                           MOVB  #TAB,-(R0)
00310                           .PRINT
00312                           BR    10$
00314           50$:
00314                   ADD   #50.,SP
00320                   RESREG 0,1,2,3
00330                   RTS   R5
                ;
                ;
                ;
```

```
                        .SBTTL PRHEAD: PRINT ARRAY OF CHARACTERS
000332          NLTBHD:
000332                  SPRINT #NLTB
000344                  BR PRHEAD
000346          NLHEAD:
000346                  SPRINT #NEWLINE
000360          PRHEAD:
                        ;PRINT DESCRIPTORS IN LIST PTD TO BY R5
000360                  SAVREG 0,1
00364           5$:
00364                   MOV   (R5)+,R1
00366                   BEQ   50$
00370                           .PRINT  #TABC
00376                           MOV   R1,R0
00400                           TSTB (R1)+
00402                           BNE   .-2
00404                           MOVB  #TERM,-(R1)
00410                           .PRINT
00412                           CLRB  (R1)
00414                           BR    5$
00416           50$:
00416                   RESREG 0,1
00422                   RTS   R5
                ;
                ;
                ;
```

```
                        .SBTTL DOUBLE PRECISION DIVISION (WITH ROUNDING), + & -
 0424           ADDIVR:
                        ;ARRAY OF DOUBLE PRECISION DIVISION AND ROUNDING OF RESULT
                        ;FIRST ARG IS COUNT, & OTHERS ARE ADDRESSES OF BASES OF ARR
 000424                 SAVREG 0,1
 000430                 MOV  (R5)+,R0
 000432                 MOV  #10$,R1
 000436                 MOV  (R5)+,(R1)+
 000440                 MOV  (R5)+,(R1)+
 00442                  MOV  (R5)+,(R1)+
 00444          5$:
 00444                  DEC  R0
 00446                  BLT  50$
 00450                      JSR  R5,DDIVR
 00454          10$:        .WORD 0,0,0,0    ;3 ARGS AND 0
 00464                      ADD  #4,10$
 00472                      ADD  #4,10$+2
 00500                      ADD  #4,10$+4
 00506                      BR   5$
 00510          50$:
 00510                  RESREG 0,1
 00514                  RTS  R5
                ;
                ;
                ;
 00516          ACDDVR:
                        ;ARRAY OF DOUBLE PRECISION DIVISION BY A CONSTANT
                        ;WITH ROUNDING OF RESULT
                        ;FIRST ARG IS COUNT, & OTHERS ARE ADDRESSES OF BASES OF ARR
 00516                  SAVREG 0,1
 00522                  MOV  (R5)+,R0
 00524                  MOV  #10$,R1
 00530                  MOV  (R5)+,(R1)+
 00532                  MOV  (R5)+,(R1)+
 00534                  MOV  (R5)+,(R1)+
 00536          5$:
 00536                  DEC  R0
 00540                  BLT  50$
 00542                      JSR  R5,DDIVR
 00546          10$:        .WORD 0,0,0,0    ;3 ARGS AND 0
 00556                      ADD  #4,10$+2
 00564                      ADD  #4,10$+4
 00572                      BR   5$
 00574          50$:
 00574                  RESREG 0,1
 00600                  RTS  R5
                ;
                ;
                ;
 00602          DDIVR:
                        ;DIVIDE 2ND BY 1ST TO OBTAIN 3RD
                        ;AND ROUND THIRD TO NEAREST NO.
 0602                   SAVREG 0,1,2
 00610                  CMP  -(SP),-(SP)
 00612          10$:
 00612                  MOV  (R5)+,R0
 00614                  BEQ  50$
```

```
58 00616                        MOV   (R5)+,R1
59 00620                        MOV   (R5)+,R2
60 00622                        DDIV (R0),2(R0),(R1),2(R1),(R2),2(R2),(SP),2(SP)
61 00756                        DMOL2 (SP),2(SP),15$    ;OFLOW COND BRANCH
62 00766                15$:    DCMP (SP),2(SP),(R0),2(R0),10$,20$
63 01006                20$:
64                              ;<=
65 01006                        DINC (R2),2(R2)
66 01016                        BR    10$
67 01020        50$:
68 01020                CMP   (SP)+,(SP)+
69 01022                RESREG 0,1,2
70 01030                RTS   R5
71              ;
72              ;
73              ;
74 01032        ACDMOLS:
75                      ;ARRAY OF DOUBLE PRECISION MULTIPLICATION BY A CONSTANT
76                      ;FIRST ARG IS ARRAY LENGTH,
77                      ;SECOND IS CONSTANT
78                      ;THIRD IS MULTIPLIER,FOURTH IS MULTIPLICAND
79 01032                SAVREG 0,1
80 01036                MOV   (R5)+,R0
81 01040                MOV   #10$,R1
82 01044                MOV   (R5)+,(R1)+
83 01046                MOV   (R5)+,(R1)+
84 01050                MOV   (R5)+,(R1)+
85 01052        5$:
86 01052                DEC   R0
87 01054                BLT   50$
88 01056                    JSR   R5,DMOLS
89 01062                10$:    .WORD 0,0,0,0   ;3 ARGS AND 0
90 01072                        ADD   #4,10$+2
91 01100                        ADD   #4,10$+4
92 01106                        BR    5$
93 01110        50$:
94 01110                RESREG 0,1
95 01114                RTS   R5
96              ;
97              ;
98 01116        DADDNOS:
99                      ;GROUPS OF 2 ARGUMENTS. ADD 1ST TO 2ND.
00 1116                 SAVREG 0,1
01 1122         5$:
02 1122                 MOV   (R5)+,R0
03 1124                 BEQ   50$
04 1126                     MOV   (R5)+,R1
05 1130                     DADD  (R0),2(R0),(R1),2(R1)
06 1144                     BR    5$
07 1146         50$:
08 1146                 RESREG 0,1
09 1152                 RTS   R5
                ;
11              ;
12              ;
13 1154         DSUBNOS:
14                      ;GROUPS OF 2 ARGUMENTS
```

```
                        ;SUB 1ST FROM 2ND.
1154                    SAVREG 0,1
1160            5$:
1160                    MOV(R5)+,R0
1162                    BEQ  50$
1164                            MOV  (R5)+,R1
1166                            DSUB  (R0),2(R0),(R1),2(R1)
1202                            BR   5$
1204            50$:
1204                    RESREG 0,1
1210                    RTS  R5
                ;
                ;
                ;
1212            RULEOFF:
                        ;DRAW A LINE ACROSS PAGE
1212                    SAVREG 0,1
1216                    MOVB #'_,CHARAC
1224                    MOV  #80.,R1
1230                    .PRINT #NEWLINE
1236            10$:
1236                            .PRINT  #CHARAC
1244                            DEC  R1
1246                            BGT  10$
1250                    RESREG 0,1
1254                    RTS  R5
                ;
                ;
                ;
```

CNVMILLL: CONVERT TIME TO MILLISECONDS

```
                        .SBTTL CNVMILLL: CONVERT TIME TO MILLISECONDS
001256          CNVMILL:
                        ;CONVERT FROM 1/60 THS SEC TO MILLISEC
001256                  CMP   -(SP),-(SP)
001260          5$:
001260                  MOV   (R5)+,10$+2
001264                  BEQ   50$
001266                        MOV   SP,10$+4
001272                        MOV   SP,20$+2
001276                        MOV   (R5)+,20$+4
001302                        JSR   R5,DMOLS
001306                  10$:  .WORD HUNDRED,0,99.,0
001316                        JSR   R5,DDIVR
001322                  20$:  .WORD SIX,99.,0,0
001332                        BR    5$
001334          50$:
001334                  CMP   (SP)+,(SP)+
001336                  RTS   R5
                        ;
                        ;
                        ;
                        .SBTTL CNVMIC: CONVERT TIME TO MICROSECONDS
001340          CNVMIC:
                        ;CONVERT FROM 1/10000*1/60 THS SEC TO MICROSEC
001340                  CMP   -(SP),-(SP)
001342          5$:
001342                  MOV   (R5)+,10$+2
001346                  BEQ   50$
001350                        MOV   SP,10$+4
001354                        MOV   SP,20$+2
001360                        MOV   (R5)+,20$+4
001364                        JSR   R5,DMOLS
001370                  10$:  .WORD TEN,0,99.,0
001400                        JSR   R5,DD1VR
001404                  20$:  .WORD SIX,99.,0,0
001414                        BR    5$
001416          50$:
001416                  CMP   (SP)+,(SP)+
001420                  RTS   R5
                        ;
                        ;
```

```
                        .SBTTL DECIMAL FROM 32 BIT BINARY CONVERSION
                ;
                ;CONVERSION OF 32 BIT NO TO DECIMAL
                ;
001422          DDEC:
                ;SUBROUTINE TO CONVERT 32 BIT NO TO DEC
                ;R0->BUFFER WHERE RESULT WILL BE PLACED
                ;R1->WHERE NO IS (LOW ORDER WORD)
                ;ALL REGISTERS RETURNED UNALTERED
                ;       LOW     HIGH
                ;B      R1      R2
                ;D      R3      R4
01422                   SAVREG 4,3,2,1,0
                ;
01434                   MOV  2(R1),R2   ;B=R1,R2;LOW,HIGH
01440                   MOV  (R1),R1
01442                   TST  R2 ;IF B<0 THEN DO;
01444                   BGE  1$
01446                      DNEG R1,R2       ;B=-B
01454                      MOVB #'-,(R0)+
01460           1$:     MOV  #1,R3       ;D=1
01464                   CLR  R4
01466                   CLR  -(SP)
01470                   CLR  -(SP)       ;0 END MARKER
01472           2$:     DCMP R3,R4,R1,R2,3$,4$  ;DO WHILE(D<=B)
01506           3$:        ;D<=B
01506                      DSTACK R3,R4     ;STACK<=D
01512                      DMOL2 R3,R4,31$ ;D=D*10
01520                      DSTACK R3,R4
01524                      DMOL2 R3,R4,31$
01532                      DMOL2 R3,R4,31$
01540                      DADD (SP)+,(SP)+,R3,R4
01546                      BR   2$
01550                      31$:     ;OVERFLOW
01550                      CMP  (SP)+,(SP)+
01552           4$:     ;D>B
01552                   MOV  (SP)+,R3   ;LOWD   ;DO WHILE(D<=STACK^=0)
01554                   BNE  5$
01556                   MOV  (SP)+,R4   ;HIGH D
01560                   BNE  6$
01562                   BR   10$
01564                   5$:     MOV  (SP)+,R4
                                ;D<=B & D^=0
01566                   6$:     MOVB #'0,(R0)   ;ST=ST & CHAR
01572                   7$:     DCMP R3,R4,R1,R2,8$,9$  ;DO WHILE (D<=B)
01606                           8$:     ;D<=B
01606                                   INCB (R0)
01610                                   DSUB R3,R4,R1,R2        ;B=B-D
01616                                   BR   7$
01620                   9$:     INC  R0 ;
01622                           BR   4$
01624           10$:    ;D=0
01624                   CMP  R0,(SP)
01626                   BNE  11$
01630                      MOVB #'0,(R0)+
01634           11$:    CLRB (R0)       ;0 END MARKER
01636                   TST  (SP)+
```

```
;0 01640                 RESREG 4,3,2,1
;1 01650                 RTS  R5
;0                ;
;1                ;
;2                ;
```

```
1                       .SBTTL DECIMAL FROM 16 BIT BINARY CONVERSION
2  001652       DEC:
3               ;SUBROUTINE TO CONVERT 16 BIT NO TO A DEC STRING
4               ;R0=->BUFFER WHERE DECIMAL CHARACTERS RESULT WILL BE PLACED
5               ;R1=BINARY NO & IS RETURNED UNCHANGED
6  001652               SAVREG 3,2,1,0
7                       ;
8  001662               TST  R1 ;IF B<0 THEN DO;
9  001664               BGE  1$
10 01666                   NEG  R1 ;B=-B
11 01670                   MOVB #'-,(R0)+  ;ST=ST&'-'
12 01674       1$:     MOV  #1,R2      ;D=1
13 01700               CLR  -(SP)      ;0 END MARKER
14 01702       2$:     CMP  R2,R1      ;DO WHILE (D<=B);
15 01704               BHI  3$
16 01706                   MOV  R2,-(SP)   ;STACK<=D
17 01710                   MOL  (R5),R2    ;D=D*10.
18 01730                   BR   2$
19 01732       3$:     ;D>B
20 01732       4$:
21 01732               MOV  (SP)+,R2   ;D<=STACK
22 01734               BEQ  7$
23                         ;D<=B
24 01736                   MOVB #'0,R3     ;R3=C='0'
25 01742               5$:
26 01742                   CMP  R1,R2      ;DO WHILE (B>=D);
27 01744                   BLT  6$
28 01746                       INC  R3 ;C=C+1
29 01750                       SUB  R2,R1      ;B=B-D
30 01752                       BR   5$
31 01754               6$:
32 01754                   MOVB R3,(R0)+   ;ST=ST&C
33 01756                   BR   4$
34 01760       7$:
35 01760               CMP  R0,(SP)    ;IF ST='' THEN ST=ST&'0'
36 01762               BNE  8$
37 01764                   MOVB #'0,(R0)+
38 01770       8$:
39 01770               CLRB (R0)       ;0 END MARKER
40 01772               TST  (SP)+
41 01774               RESREG 3,2,1
42 02002               TST  (R5)+
43 02004               RTS  R5
44              ;
45              ;
```

```
                        .SBTTL DMOLS: DOUBLE PRECISION MULTIPLICATION
002006          DMOLS:
                        ;GROUPS OF THREE ARGUMENTS PTD TO BY R5
                        ;MULTIPLY 2ND. BY 1ST. TO GIVE THIRD ARGUMENT
                        ;UNTIL ZERO FIRST ARGUMENT IS REACHED
                        ;ARGUMENT ARE ADDRESSES OF DOUBLE PRECISION WORDS
002006                  SAVREG 0,1,2
002014          5$:
002014                  MOV   (R5)+,R0    ;MULTIPLIER
02016                   BEQ   50$
02020                           MOV   (R5)+,R1
02022                           MOV   (R5)+,R2
02024                           DMOL   (R0),2(R0),(R1),2(R1),(R2),2(R2)
02106                           BR    5$
02110           50$:
02110                   RESREG 0,1,2
02116                   RTS   R5
                ;
                ;
                ;
```

```
                        .SBTTL DDIVS: DOUBLE PRECISION DIVISION
002120          DDIVS:
                        ;ARGUMENTS COME IN GROUPS OF FOUR,
                        ;TERMINATED BY A ZERO
002120                  SAVREG 0,1,2,3
002130          5$:
002130                  MOV   (R5)+,R0
002132                  BEQ   50$
002134                          MOV   (R5)+,R1
02136                           MOV   (R5)+,R2
02140                           MOV   (R5)+,R3
02142                           DDIV  (R0),2(R0),(R1),2(R1),(R2),2(R2),(R3),2(R3)
02276                           BR    5$
02300           50$:
02300                   RESREG 0,1,2,3
02310                   RTS   R5
                ;
                ;
                ;
                        .SBTTL PLINK: MACROS
```

```
                .SBTTL PLINK
                .CSECT DUMP
        .MACRO NXTWD P,MIN,MAX,?L1
        ;ASSUME P IS A REGISTER;
        ;IF P>=MAX THEN P=P-MAX+MIN
                CMP  P,MAX
                BLO  L1
                        SUB  MAX,P
                        ADD MIN,P
        L1:
        .ENDM
        ;
        ;
        ;
        .MACRO LSTWD P,MIN,MAX,?L1
                CMP  P,MIN
                BHI  L1
                        SUB  MIN,P
                        ADD  MAX,P
        L1:
        .ENDM
        ;
        ;
        ;
        .MACRO NXTMSG P,MIN,MAX
        ;FIND NEXT MESSAGE
                MOV  R4,-(SP)
                MOVB 1(P),R4
                ASL  R4 ;WORDS TO BYTES
                ADD  R4,P
                NXTWD P,MIN,MAX
                MOV  (SP)+,R4
        .ENDM
        ;
        ;
        ;
0       PDUMP:
0               BIT  #4,@#SWREG
6               BEQ  PD59$
0               JSR  R5,PDUMP0  ;PRINT MINMXP STATUS
4       PDUMP1: ;ENTRY POINT FOR DUMPING BUFFER POINTERS ONLY
4               JSR  R5,PRNOS
0               .WORD 0,10.,D17,MSGIN,D18,MSGOUT,M36,WDSIN,M37,WDSOUT,0
6               JSR  R5,PRNOS
2               .WORD 0,8.,D10,FIN,D11,RIN,D12,PIN,D13,FOUT,D14,ROUT,D15,POUT,0
0       PD59$:
0               RTS  R5
        ;
        ;
2       PDUMP0: ;PRINT MINMXP STATUS
2               MOV  R3,-(SP)
4               MOV  SP,1$+26.
:0              MOV  R4,-(SP)
:2              MOV  SP,1$+22.
:6              JSR  R5,PRNOS
;2      1$:     .WORD 0,8.,D8,D,D4,LEAD,D7,D3,CM,D6,0,D5,0,0
;6              CMP (SP)+,(SP)+
```

```
58 00170                RTS   R5
5                       ;
```

```
                .SBTTL DEBUGGING MESSAGES
2       D1:     .ASCII <CR><LF><TAB><TAB><TAB><TAB><TAB>/RECMSG:/<TERM>
1       D2:     .ASCII <CR><LF>/SNDMSG:/<TERM>
3       D3:     .ASCIZ /CM=/
7       D4:     .ASCIZ /LEAD=/
5       D5:     .ASCIZ /R3=/
1       D6:     .ASCIZ /R4=/
5       D7:     .ASCIZ /PRX=/
2       D8:     .ASCIZ /D=/
5       D9:     .ASCIZ <CR><LF>/AT MINMX2:/
2       D10:    .ASCIZ /FIN:/
7       D11:    .ASCIZ /RIN:/
4       D12:    .ASCIZ /PIN:/
1       D13:    .ASCIZ /FOUT:/
7       D14:    .ASCIZ /ROUT:/
5       D15:    .ASCIZ /POUT:/
3       D16:    .ASCII /-/<TERM>
5       D17:    .ASCIZ /MSGIN:/
4       D18:    .ASCIZ /MSGOUT:/
4       D19:    .ASCIZ /INVALID CHECKSUM/
5       D20:    .ASCIZ <CR><LF>/OUTBUF BLOCKED - 10 MESSAGES/<TERM>
5       D21:    .ASCIZ /NAK, EXPID=/
1       D22:    .ASCIZ /RECEIVED SID=/
7       D23:    .ASCIZ /AID NOT RECOGNISED/
2       D24:    .ASCIZ /UNEXP.DATA=/
6       D25:    .ASCIZ /BUFFER MESSAGE WITHOUT HEADER/
4       D27:    .ASCIZ <CR><LF>/CONTENTS OF LINK OUTPUT BUFFER ARE:/<TERM>
3       D28:    .ASCIZ <CR><LF>/5 OR MORE MESSAGES OUTSTANDING IN OUTBUF/<TERM>
7       D29:    .ASCIZ <CR><LF>/OUTBUF BLOCKED - SEMAPHORE/<TERM>
5       D30:    .ASCIZ /RECEIVE BUFFER OVERFLOW/
5       D31:    .ASCIZ   /PLSW RECEIVE OR SEND BIT NOT SET ON INTERRUPT/
3       D32:    .ASCIZ <CR><LF>/2ND PLINK INTERRUPT WAITING FOR ATTENTION/
7       D33:    .ASCIZ <CR><LF>/3RD OR MORE PLINK INTS. /
2       D34:    .ASCIZ /PS=/
6       D35:    .ASCIZ /D=/
1       D36:    .ASCIZ /PC=/
5       D37:    .ASCIZ /SEND/
2       D38:    .ASCIZ /LS/
5       D39:    .ASCIZ /LR/
0       D41:    .ASCIZ /VSCOR=/
                .EVEN
        ;
```

```
                .SBTTL PRINT DESCRIPTORS AND VARIABLES
0       PRNOS:
                ;PRINT DESCRIPTORS AND NOS.
                ;ADDRESSES OF DESCS & NOS BEGIN 2 WORDS BEYOND R5, TERMINATED BY(
                ;(R5)=0 FOR 16 BIT, 1 FOR 32 BIT.
                ;2(R5)=NO BASE(10. E.G.)
0               SAVREG 0,1,2,3
0               SUB   #50.,SP     ;PRINT BUFFER
4               CLR   R2 ;R2=TAB COUNTER
6               MOV   (R5)+,R3    ;R3=BIT INDICATOR
0               MOV   (R5)+,25$   ;BASE
4       3$:     TST   (R5)        ;END OF LIST ?
6               BEQ   50$
0                       MOV   SP,R0       ;R0->BUFFER
2                       DEC   R2
4                       BGE   5$
6                               MOVB  #CR,(R0)+
2                               MOVB  #LF,(R0)+
6                               MOV   #3,R2       ;TAB COUNT 4
2                               BR.   10$
4                       5$:
4                               MOVB  #TAB,(R0)+
0               10$:
0                       MOV   (R5)+,R1    ;CHARACTER STRING
2                       BEQ   50$
4               20$:
4                       MOVB  (R1)+,(R0)+
6                       BNE   20$
0                       TSTB -(R0)
2                       MOVB #BL,(R0)+
6                       TST   R3 ;16 OR 32 BIT ?
0                       BGT   30$
2                               MOV   @(R5)+,R1   ;R1=NO
4                               JSR   R5,DEC
0                       25$:    .WORD 10.
2                               BR   35$
4                       30$:
4                               MOV   (R5)+,R1    ;R1->NO
6                               JSR   R5,DDEC
2               35$:
2                       MOVB #TERM,(R0)+
6                       MOV   SP,R0
0                       .PRINT
2                       BR   3$
4       50$:
4               TST   (R5)+
6               ADD   #50.,SP
2               RESREG 0,1,2,3
2               RTS   R5
        ;
        ;
        ;
```

PRINT DESCRIPTORS AND VARIABLES

```
1  001374          PROUTB:
2                          ;PRINT OUTPUT BUFFER AND POINTERS
3  001374                  SAVREG 1,0
4  001400                  JSR  R5,PDUMP1  ;PRINT BUFFER POINTERS
5                          ;PRINT CONTENTS OF OUTBUF
6  001404                  .PRINT  #D27    ;CONTENTS OF LINK OUTBUF ARE
7  001412                  MOV  FOUT,R0
8  001416                  2$:
9  001416                          CMP  R0,ROUT
10 01422                           BEQ  20$
11 01424                           MOV  R0,4$+6
12 01430                           MOV  R0,R1
13 01432                           NXTMSG R1,#MINS,#MAXS
14 01464                           MOV  R1,4$+8.
15 01470                           JSR  R5,PRMSG
16 01474                   4$:
17 01474                           .WORD 1,MINS,MAXS,0,0
18 01506                           MOV  R1,R0
19 01510                           BR   2$
20 01512           20$:
21 01512                   RESREG 1,0
22 01516                   RTS  R5
23                 ;
24                 ;
25                 ;
```

PRMSG:PRINT MESSAGE RECEIVED OR SENT

```
                        .SBTTL PRMSG:PRINT MESSAGE RECEIVED OR SENT
                        .GLOBL PRMSG
001520          PRMSG:
                        ;PRINT MESSAGE RECEIVED OR SENT
                        ;(R5)=0 FOR RECEIVED,=1 FOR SENT
                        ;2(R5)=MIN
                        ;4(R5)=MAX
                        ;6(R5)=FRONT
                        ;8.(R5)=END
01520                   SAVREG 0,1,2,3
01530           PRMSG0:
01530                   SUB  #30.,SP
01534                   MOV  #4,R2      ;TAB COUNT
01540                   MOV  SP,R0
01542                   TST  (R5)
01544                   BGT  10$
01546                       SPRINT #D1      ;RECEIVED
01560                       BR  15$
01562                   10$:
01562                       SPRINT  #D2     ;SEND
01574           15$:
01574                   MOV  6(R5),R3
01600           20$:
01600                   CMP  R3,8.(R5)
01604                   BEQ  50$
01606                       MOV  SP,R0
01610                       DEC  R2
01612                       BGE  25$
01614                           BR   35$
01616                       25$:
01616                           MOVB #TAB,(R0)+
01622                   30$:
01622                       MOVB (R3)+,R1
01624                       NXTWD R3,2(R5),4(R5)
01642                       JSR  R5,DEC
01646                       .WORD 8.
01650                       MOVB #TERM,(R0)+
01654                       MOV  SP,R0
01656                       .PRINT
01660                       BR  20$
01662                   CLR  R2
01664           35$:
01664                   CMP  R3,8.(R5)
01670                   BEQ  50$
01672                       MOV  SP,R0
01674                       DEC  R2
01676                       BGE  40$
01700                           MOVB  #CR,(R0)+
01704                           MOVB #LF,(R0)+
01710                           TST  (R5)
01712                           BGT  37$
01714                               MOV  #5,R2
01720                           36$:
01720                               MOVB  #TAB,(R0)+
01724                               DEC  R2
01726                               BGT  36$
01730                       37$:
```

```
58 01730                                        MOV  #4.,R2
59 01734                                        BR   45$
60 01736                                40$:
61 01736                                        MOVB  #TAB,(R0)+
62 01742                            45$:
63 01742                                MOV  (R3)+,R1
64 01744                                NXTWD R3,2(R5),4(R5)
65 01762                                JSR  R5,DEC
66 01766                                .WORD 8.
67 01770                                MOVB #TERM,(R0)+
68 01774                                MOV  SP,R0
69 01776                                .PRINT
70 02000                                BR  35$
71
72 02002                    50$:
73 02002                            MOV  SP,R0      ;PRINT NEWLINE
74 02004                            CLRB (R0)
75 02006                            .PRINT
76 02010                            ADD  #30.,SP
77 02014                    59$:
78 02014                            ADD #10.,R5
79 02020                            RESREG 0,1,2,3
80 02030                            RTS  R5
81                          ;
82                          ;
83                          ;
```

```
                .SBTTL PARALLEL LINK MESSAGE TRANSFER ROUTINES
                .CSECT PLINK
                NOUTB=10.       ;NO OF MESSAGES IN OUTBUF WHEN IT BECOMES BLOCKEI
                PLVA=320
                PLSW=322
                INTSW=340       ;INTERRUPTS PROCESSED AT PRIORITY 7
                PS=177776
                PLSR=164000
                PLOB=164002
                PLIB=164004
                HEAD=125
                MAXID=177
                RINT=100        ;BIT 6
                SINT=040        ;BIT 5
                RWD=200 ;BIT 7
                INT=140 ;SET BITS 6 & 5 TO ENALBLE RECEIVE & SEND INTERRUPTS
                SWD=100000      ;BIT 15
                NOINT=0
        ;
        ;
        ;
                .MACRO SNDW W
                TST  @#PLSR
                BMI  .-4
                        DINC WDSOUT,WDSOUT+2
                        MOV  W,@#PLOB
                .ENDM
        ;
        ;
        ;
        PIN:    .WORD MINR      ;PARALLEL LINK INPUT RECEIVE PTR
        FIN:    .WORD MINR      ;FRONT OF INPUT BUFFER
        RIN:    .WORD MINR      ;REAR OF INPUT BUFFER
        POUT:   .WORD MINS
        FOUT:   .WORD MINS
        ROUT:   .WORD MINS
        MINR:   .BLKW 200       ;RECEIVE BUFFER
        MAXR:
        MINS:   .BLKW 200       ;SEND BUFFER
        MAXS:
        OFLO:   .WORD 0 ;# OVERFLOWS OF INPUT BUFFER
        MSGIN:  .WORD 0 ;NO OF MESSAGES IN INPUT BUFFER
        MSGOUT: .WORD 0
        EXPID:  .WORD 1 ;EXPECTED I D
        NXTSID: .WORD 1
        OUTBUF: .WORD 1 ;OUTPUT BUFFER ACCESS SEMAPHORE
        LR:     .WORD 0 ;REMAINING LENGTH OF MESSAGE BEING RECEIVED
        LS:     .WORD 0 ;REMAINING LENGTH OF MESSAGE BEING SENT
        SEND:   .WORD 1 ;SEND SEMAPHORE
        RESEND: .WORD 0
        NORINT: .WORD 0,0       ;NO OF RECEIVE INTERRUPTS
        WDSIN:  .WORD 0,0       ;WORDS RECEIVED ON DR11
        WDSOUT: .WORD 0,0       ;WORDS SENT ON DR11
        WDSI:   .WORD 0,0       ;WORDS RECEIVED ON DR11
        WDSO:   .WORD 0,0       ;WORDS SENT ON DR11
        RSEM:   .WORD 1 ;RECEIVE SEMAPHORE TO INDICATE:
                        ;=1 NO DATA IN
```

58 ;<=0 DATA IN

PARALLEL LINK INTERRUPT SERVICE ROUTINE

```
                        .SBTTL PARALLEL LINK INTERRUPT SERVICE ROUTINE
                        .CSECT PLISR
000000          ISR:
                        ;BIC  #RINT,@#PLSR
000000                  SAVREG 0,1
                        ;TST  @#PLSR    ;SEND?
                        ;BMI  1$
                        ;       JSR  R5,SENDC
                        ;       BR   30$
                ;       TSTB @#PLSR     ;RECEIVE
00004           ISR0$:
                ;       BPL 20$
                ;               DINC NORINT,NORINT+2    ;# RECEIVE INTERRUPTS
00004                           DEC RSEM
00010                           BGE  12$
00012                                   CMP  RSEM,#-1   ;2 WORDS OR MORE OF INPUT
00020                                   BEQ  30$
00022                                   BLT  11$
00024                                           JSR  R5,PSTACK
00030                                           BR   30$
00032                                   11$:
00032                                           .PRINT  #D33    ;3 OR MORE INTS.
00040                                           BR   30$
00042                   12$:
00042                           MOV  @#PLIB,R0
00046                           DINC WDSIN,WDSIN+2      ;# WORDS RECEIVED
00060                           MOV  PIN,R1
00064                           MOV  R0,(R1)+
00066                           NXTWD R1,#MINR,#MAXR    ;
00104                           MOV  R1,PIN
                                ;CMP  R1,FIN    ;OVERFLOW?
                                ;BNE  2$
                                ;       INC  OFLO
                                ;       .PRINT #D30     ;RECEIVE BUFFER OVERFLOW
                                ;       BR   3$ ;SHOULD NEVER OCCUR - SENDER'S
                        ;2$:                                    BUFFER BLOCKED.
00110                           DECB LR
00114                           BGT  9$ ;CONTINUE
00116                           BEQ  5$
                                        ;EXPECTING HEADER
                                        ;INCB LR
00120                                   CMPB R0,#HEAD
00124                                   BNE  22$
00126                                           SWAB R0 ;SAVE LENGTH
00130                                           MOVB R0,LR
00134                                           DECB LR
                                                ;BEQ  5$
00140                                           BR   9$
                                        ;21$:   .WORD 0 ;UNEXPECTED DATA WORD
00142                                   22$:
                                                ;UNEXPECTED DATA
                                ;       MOV  R0,21$
                                ;       JSR  R5,PRNOS
                        ;25$:   .WORD 0,8.,D24,21$,D37,SEND,D38,LS,D39,LR,0
00142                           3$:     ;REMOVE WORD FROM QUEUE
00142                                   LSTWD R1,#MINR,#MAXR
00160                                   TST  -(R1)
```

```
                        MOV   R1,PIN
                        BR    9$
                5$:     ;LR=0
                        INC   MSGIN
                ;       BIT   #4,@#SWREG
                ;       BEQ   7$
                ;             MOV   R1,6$+8.    ;END
                ;             MOV   R1N,6$+6    ;FRONT
                ;             JSR   R5,PRMSG
                ;       6$:   .WORD 0,MINR,MAXR,0,0
                ;7$:
                        MOV   R1,RIN      ;END OF MESSAGE
                        INC   RSEM        ;FREE SEM TO ALLOW DATA IN READ
                        BLE   12$
                        BIC   #200,@#PS   ;PRIO 3
                        ;BIS   #RINT,@#PLSR
                        JSR   R5,INTPT    ;INTPT CONTENTS OF RECEIVE BUFFER
                        BR    30$
        9$:
                INC   RSEM        ;ENABLE FURTHER DATA READS
                BLE   12$         ;IF DATA IN THEN TAKE IT
                BR    30$
        20$:
                .PRINT  #D31      ;PLSW REC OR SEND BIT NOT SET ON INTERRUP
30$:
        RESREG 0,1
        ;BIS   #RINT,@#PLSR
        RTI
;
;
;
PSTACK:
        .SBTTL PSTACK:PRINT CONTENTS OF STACK
PSTACK:
        MOV   @#PS,FR
        MOV   (SP),FR+14.
        MOV   2(SP),FR+16.
        MOV   4(SP),FR+2
        MOV   6(SP),FR+4
        MOV   8.(SP),FR+6
        MOV   10.(SP),FR+8.
        MOV   12.(SP),FR+10.
        MOV   14.(SP),FR+12.
        .PRINT  #D32      ;2ND INTERRUPT
        JSR   R5,PRNO
        .WORD 0,8.,FR,FR+14.,FR+16.,FR+2,FR+4,FR+6,FR+8.
        .WORD FR+10.,FR+12.,0
        RTS   R5
;
;
;
```

```
                        .SBTTL SEND LINK DATA
00376           SENDA:
                        .ENABL LSB
                        ;ASSUME R0 IS AVAILABLE WITHOUT SAVING
                        ;TRIGGER SEND IF NO MESSAGE IS BEING SENT, OTHERWISE RETURN
00376                   DEC  SEND       ;INITIALLY SEND=1
00402                   BLT  5$
00404                   2$:
                                ;BISB #SINT,@#PLSR
00404                           TST  RESEND
00410                           BEQ  3$
00412                                   CLR  RESEND
00416                                   MOV  POUT,POUT
00424                   3$:
00424                           CMP  POUT,ROUT
00432                           BEQ  4$
                                ;WAIT FOR LAST MESSAGE TO CLEAR
                                ;AND THEN WASTE SOME TIME
                                ;TST  @#PLSR
                                ;BMI  .-4
                                ;MOV  #50,-(SP)
                                ;DEC (SP)
                                ;BGT  .-2
                                ;TST  (SP)+
00434                                   MOV  POUT,10$+6
00442                                   MOV  ROUT,10$+8.
00450                           BIT  #4,@#SWREG
00456                           BEQ  20$
00460                                   JSR  R5,PRMSG
00464                           10$:    .WORD 1,MINS,MAXS,0,0
00476                   20$:
00476                           MOV  POUT,R0
00502                           MOVB  1(R0),LS  ;LENGTH
00510           SENDC:
00510                           DECB  LS
00514                           BLT  2$
00516                                   MOV  POUT,R0
00522                                   SNDW (R0)+
00546                                   NXTWD R0,#MINS,#MAXS
00564                                   MOV  R0,POUT
00570                                   BR   SENDC      ;INTERRUPT SEND DOESN'T WORK
                                                        ;CONTINUE WITH INTERRUPT
00572                   4$:     ;BICB #SINT,@#PLSR
00572                           INC  SEND
00576                           BLE  2$
00600           5$:
00600                   RTS  R5
                        .DSABL LSB
                ;
                ;
                ;
```

```
                        .SBTTL INTERPRET RECEIVED MESSAGES
                        .CSECT INTPT
                        ;ASSUME R0, R1 ARE AVAILABLE
 000000         INTPT:
 000000                 DEC   OUTBUF
 000004                 BGE   1$
 000006                     INC   OUTBUF
 000012                     RTS   R5
 000014         1$:
0 00014                 CMP   MSGOUT,#NOUTB
1 00022                 BLE   2$
2 00024                     .PRINT #D20
3 00032                     JSR   R5,PROUTB
4 00036         2$:
5 00036                 CMP   FIN,RIN
6 00044                 BNE   32$
7 00046                     INC   OUTBUF
8 00052                     RTS   R5
9 00054         32$:
0                       ;FREE OUTPUT BUFFER.
1 00054                 JSR   R5,FREAIDO ;FREE MESSAGES WITH AID 0 FROM OUTBUF
2                       ;CARRY MAY BE SET BUT CONTINUING WITH INTPT IS ONLY HOPE
3 00060                 MOV   FIN,R0
4 00064                 CMPB  (R0),#HEAD ;ELIMINATE WORDS UNTIL HEADER FOUND
5 00070                 BEQ   3$
6 00072                     TST   (R0)+
7 00074                     NXTWD R0,#MINR,#MAXR
8 00112                     MOV   R0,FIN
9 00116                     .PRINT #D25;BUFFER MESSAGE WITHOUT HEADER
0 00124                     .EXIT
1 00126         3$:
2 00126                 MOV   R0,R1
3 00130                 NXTMSG R1,#MINR,#MAXR
4                       ;R0=FRONT, R1=NEXT MESSAGE
5 00162                 MOV   R2,-(SP)    ;AVAILABLE REGISTER
6 00164                 MOV   R0,-(SP)    ;FRONT OF MESSAGE
7 00166                 JSR   R5,CHKSUM
8 00172         4$:     .WORD MINR,MAXR
9 00176                 TST   R2
0 00200                 BEQ   6$
1                           ;INVALID CHECKSUM
2                           ;SPRINT #D19     ;INVALID CHECKSUM
3 00202                     JSR   R5,SENDACK ;NACK
4 00206                     .WORD 0
5 00210                     BR    30$         ;NEXT MESSAGE
6 00212         6$:
7 00212                 TST   (R0)+
8 00214                 NXTWD R0,#MINR,#MAXR
9 00232                 MOV   R0,-(SP)    ;SAVE PTR TO (SID,AID)
0                       ;SEE IF SID IS VALID
1 00234                 MOVB  (R0),R2
2 00236                 BEQ   28$         ;IF SID=0 THEN MESSAGE IS JUST AN ACK
                                          ;SO IGNORE
4 00240                 CMPB  R2,EXPID
5 00244                 BNE   16$
6                           ;MESSAGE HAS EXPECTED SID
7 00246                     INCB  EXPID
```

```
00252                           BPL   14$
00254                                 MOV   #1,EXPID
00262                   14$:
00262                           JSR   R5,INTCON
00266                           BR    28$
00270                   16$:
00270                           JSR   R5,SNDERR
00274                           BR    12$
00276           28$:
00276                   MOV   (SP),R0     ;R0=->SID
                                ;DOES THE RECEIVED MESSAGE ACK A SENT MESSAGE ?
00300                           MOVB  1(R0),R0    ;R0=AID
00304                           BEQ   12$         ;NO MESSAGE ACK - CONTINUE
00306                           BPL   8$
                                    ;NAK
00310                               INC   RESEND
00314                               JSR   R5,SENDA    ;RESEND DATA
00320                               BR    12$
00322                           8$:
00322                               JSR   R5,FREEOB   ;POSITIVE ACKNOWLEDGEMENT
00326                   12$:
00326                           TST   (SP)+
00330           30$:
00330                   MOV   R1,FIN      ;ELIMINATE MESSAGE INTERPRETED
00334                   MOV   (SP)+,R0
00336                   MOV   (SP)+,R2
00340                   INC   OUTBUF
00344                   DEC   MSGIN
00350                   BGT   INTPT
00352                   RTS   R5
                ;
00354           INTCON: ;SET UP POINTERS FOR INTERPRETING CONTENTS
                        ;OF MESSAGE
00354                   TST   (R0)+
00356                   NXTWD R0,#MINR,#MAXR
00374                   SAVREG 1          ;NEXT MESSAGE
00376                   LSTWD R1,#MINR,#MAXR
00414                   TST   -(R1)
00416                   JSR   R5,INTPTCONT        ;R0 PTS TO BEGINNING OF TEXT
00422                   RESREG 1                  ;R1 TO END &R2=SID
00424                   RTS   R5
                ;
                ;
                ;
```

INTERPRET RECEIVED MESSAGES

```
000426          FREAIDO:
                        ;ELIMINATE MESSAGES AT FRONT OF OUTPUT BUFFER WITH SID=0
                        ;RETURN CARRY SET IF OUTBUF IS BLOCKED ON RETURN
000426          5$:
000426                  CMP   FOUT,ROUT   ;ANY MESSAGES ?
000434                  BEQ   30$
000436                     MOV   FOUT,R0
000442                     TST   (R0)+
000444                     NXTWD R0,#MINS,#MAXS
00462                      TSTB (R0)        ;SID=0?
00464                      BNE   9$
                                  ;SID=0
00466                             MOV   FOUT,R0
00472                             NXTMSG R0,#MINS,#MAXS
00524                             MOV   R0,FOUT
00530                             DEC   MSGOUT
00534                             BR    5$
00536           9$:
00536                   CMP   MSGOUT,#NOUTB;LET FOREGROUND TASK WAIT UNTIL OUTBUF
                                  ;EMPTY
00544                   BLE   30$
00546                      SEC
00550                      RTS   R5
00552           30$:
00552                   CLC
00554                   RTS   R5
                ;
                ;
                ;
00556           FREEOB: ;POSITIVE ACK:FREE OUTPUT BUFFERS
                        ;UPTO AND INCLUDING ACK MESSAGE
00556                   MOV   R2,-(SP)
00560                   MOV   R1,-(SP)
00562                   JSR   R5,SRCHOUT ;GIVEN R0=ID,
                           ;RETURNS R1=PTR TO FOUND MESSAGE
                           ;R2=NO OF MESSAGES PASSED OVER
00566                   TST   R1
00570                   BEQ   9$ ;NOT FOUND - DO NOTHING
                           ;FOUND- DELETE THIS AND PREVIOUS MESSAGE
00572                      NXTMSG R1,#MINS,#MAXS
00624                      MOV   R1,FOUT
00630                      INC   R2
00632                      SUB   R2,MSGOUT
00636                      BR    10$
00640                   9$:
                           ;SPRINT #D23     ;ACKID NOT RECOGNISED.
00640           10$:
00640                   MOV   (SP)+,R1
00642                   MOV   (SP)+,R2
00644                              RTS   R5
                ;
                ;
                ;
```

```
                        .SBTTL SEND ERROR
                        .CSECT SENDERR
 000000         SNDERR: ;SEND ERROR
                        ;
                        ;MESSAGE ID IS NOT THAT EXPECTED-
                        ;1.MESSAGE BECAME A NACK ON LINE,
                        ;2.MESSAGE TRANSMISSION BEGAN BEFORE NAK OF
                        ;       PREVIOUS MESSAGE RECEIVED
                        ;3.HEADER OF PREVIOUS MESSAGE NOT DETECTED.
                        ;STRATEGY:      IF ID IS THAT OF ONE OF LAST
                        ;       TEN MESSAGES, THEN ACK, ELSE NAK
                        ;IF EXPID<NOUTB & (0<=SID<=EXPID OR MAXID+EXPID-NOUTB<=SID<=MAXID)
                        ;       OR (EXPID-NOUTB<=SID<=EXPID) THEN CALL SEND ACK
                        ;ELSE CALL SEND NAK
 00000                  MOVB (R0),R0    ;R0=SID
 00002                  CMPB EXPID,#NOUTB
 00010                  BGT  22$
 00012                      TSTB R0
 00014                      BLT  18$
 00016                      CMPB R0,EXPID
 00022                      BLE  24$        ;SEND ACK
 00024                  18$:
 00024                      MOVB #MAXID,R2
 00030                      ADD  EXPID,R2
 00034                      SUB  #NOUTB,R2
 00040                      CMPB R2,R0
 00042                      BGT  26$        ;SEND NAK
 00044                  20$:
 00044                      CMPB R0,#MAXID
 00050                      BGT  26$        ;SEND NAK
 00052                      BR   24$        ;SEND ACK
 00054                  22$:
                            ;EXPID>=NOUTB
 00054                      MOVB EXPID,R2
 00060                      SUB  #NOUTB,R2
 00064                      CMPB R2,R0
 00066                      BGT  26$        ;SEND NAK
 00070                      CMPB R0,EXPID
 00074                      BGT  26$        ;SEND NAK
 00076                  24$:
                            ;SEND ACK
 00076                      MOV  2(SP),R0   ;R0=FRONT OF MESSAGE
 00102                      JSR  R5,SENDACK
 00106                      .WORD 1
 00110                      BR   28$
 00112                  26$:
                            ;SEND NAK
 00112                      MOV  2(SP),R0
 00116                      JSR  R5,SENDACK
 00122                      .WORD 0
 00124          28$:
 00124                      RTS  R5
                        ;
                        ;
                        ;
```

```
                .SBTTL SEND MESSAGE ACK OR NAK
                .CSECT SENDACK
        SENDACK:
                ;OUTBUF BEING ALREADY OPEN,SEND AN ACK
                ;OF MESSAGE BEING INTERPRETED
                ;ON CALL,       R0=FRONT OF MSG FOR WHICH ACK IS SENT
                ;               THE WORD FOLLOWING (R5)=1 FOR AN ACK, =0 FOR
                                ;A NAK TO BE SENT
                MOV  R1,-(SP)
                MOV  R0,-(SP)   ;FRONT
                JSR  R5,OPNSEA  ;RETURNS R0=OUTPUT PTR
                MOV  (SP),R1    ;R1=->FRONT OF MSG TO ACK
                TST  (R1)+
                NXTWD R1,#MINR,#MAXR
                MOVB (R1),R1    ;R1=SID OF INTPTD MSG
                TST  (R5)+
                BGT  1$
                ;       MOV  R1,22$
                        MOV  EXPID,R1
                ;       MOV  R1,21$
                ;       JSR  R5,PRNOS
                ;10$:   .WORD 0,8.,D21,21$,D22,22$,0    ;NAK,EXPID= ,RECEIVED SID
                ;       JSR  R5,PROUTB
                        NEG  R1
        1$:
                JSR  R5,CLSNDA
                MOV  (SP)+,R0
                MOV  (SP)+,R1
                RTS  R5
        21$:    .WORD 0
        22$:    .WORD 0
                ;
                ;
                ;
```

LOSE OUTPUT BUFFER FOR THE ADDITION OF A MESSAGE

```
        .SBTTL OPEN & CLOSE OUTPUT BUFFER FOR THE ADDITION OF A MESSAGE
        .CSECT OUTBUF
OPNSE:
        ;RETURN R0->AVAILABLE WORD, =0 OTHERWISE
        SAVREG 1
20$:
        JSR  R5,INTPT   ;INTERPRET MESSAGES RECEIVED &
                        ;TRY TO CLEAR OUTPUT BUFFER
        CMP  MSGOUT,#4  ;KEEP NO OF OUTSTANDING MESSAGES LOW
                        ;SO THAT BACKGROUND INTPTATION IS NOT BLOCKED
        BLE  31$
            ;.PRINT #D28    ;5 MESSAGES OUTSTANDING
            BR   20$
31$:
        DEC  OUTBUF
        BGE  32$
            .PRINT #D29     ;OUTBUF BLOCKED
            INC  OUTBUF
            BR   31$        ;TRY AGAIN
32$:
        RESREG 1
;
;
;
OPNSEA: ;ENTRY POINT FOR SKIPPING TESTS: USED BY BACKGROUND TASKS
        MOV  ROUT,R0
        TST  (R0)+      ;FIND DATA POINT, SKIPPING HEADER SPACE
        JSR  R5,ADDOQ   ;TEST QUEUE
        TST  (R0)+      ;SKIP IDS
        JSR  R5,ADDOQ
        RTS  R5
;
CLSND:  ;ENTRY POINT FOR FOREGROUND TASK
        ;ON CALL        R0->END OF TEXT
        ;               R1=AID
        ;ON RETURN      R0=SID OF MESSAGE
        JSR  R5,CLSNDA
        INC  OUTBUF
        TST  MSGIN
        BLE  10$
            SAVREG 0,1
            JSR  R5,INTPT
            RESREG 0,1
10$:
        RTS  R5
CLSNDA:
        ;MESSAGE HAS BEEN ADDED TO OUTBUF SO FIND CHECKSUM &
        ;ON CALL LET R0 BE END OF MESSAGE TEXT, R1 BE THE AID
        MOV  R2,-(SP)
        MOV  R1,-(SP)   ;AID
        BIT  #1,R0      ;ODD BYTE
        BEQ  2$
            CLRB (R0)+
            NXTWD R0,#MINS,#MAXS
2$:
        MOV  R0,-(SP)   ;END OF TEXT, EXCLUDING CHECK SUM.
        MOV  R0,R1      ;R1=REAR, EXC CS.
```

```
CLOSE OUTPUT BUFFER FOR THE ADDITION OF A MESSAGE

                MOV  ROUT,R0     ;R0=FRONT
                SUB  R0,R1       ;WANT R1=LENGTH, EXC CS
                BGT  4$
                   SUB  #MINS,R1    ;IF L<0 THEN L=MAXS-MINS+L
                   ADD  #MAXS,R1
        4$:
                ASR R1   ;BYTES TO WORDS
                INC R1   ;R1=LENGTH, INCLUDING CHECK SUM.
                MOVB  #HEAD,(R0)+        ;HEAD
                MOVB  R1,(R0)+   ;LENGTH
                NXTWD R0,#MINS,#MAXS
                CLR  -(SP)       ;SID OF MSG
                CMP  R1,#3
                BGT  5$
                   CLRB (R0)+       ;NO SID FOR CONTENT FREE MESSAGE
                   BR   6$
                5$:
                   MOVB NXTSID,(SP)         ;SAVE SID OF MSG , TO BE RETURNED
                   MOVB NXTSID,(R0)+        ;SID
                   INCB NXTSID
                   BPL  6$
                      MOV  #1,NXTSID
        6$:
                MOVB 4(SP),(R0)+         ;ACKID
                MOV  ROUT,R0     ;FRONT OF MESSAGE
                MOV  2(SP),R1    ;REAR OF MESSAGE, EXCLUDING CHECK SUM
                JSR  R5,CHKSUM   ;RETURNS R2=CHKSUM
                .WORD MINS,MAXS
                NEG  R2
                MOV  R2,(R1)+
                NXTWD R1,#MINS,#MAXS
                MOV  R1,ROUT     ;NEW REAR OF QUEUE
                INC  MSGOUT      ;NO OF MESSAGES QUEUED FOR OUTPUT
                JSR  R5,SENDA    ;TRIGGER SEND IF NECESSARY, R0 MAY BE ALTERED
                MOVB (SP)+,R0
                TST  (SP)+
                MOV  (SP)+,R1
                MOV  (SP)+,R2
                RTS  R5
        ;
        ;
        ;
```

```
                        .SBTTL SEARCH FOR MESSAGE IN OUTBUF; FIND CHECKSUM
                        .CSECT SRCHOUT
000000          SRCHOUT:
                        ;FIND MESSAGE WITH GIVEN ID IN OUTPUT BUFFER
                        ;ON CALL,       R0=ID
                        ;ON RETURN:     R1->MESSAGE IF FOUND, =0 OTHERWISE
                        ;               R2=# OF MESSAGES PASSED OVER
000000                  MOV  R3,-(SP)
000002                  CLR  R2
00004                   MOV  FOUT,R1
00010           1$:
00010                   CMP  R1,ROUT
00014                   BEQ  2$
00016                       MOV  R1,R3
00020                       TST  (R3)+
00022                       NXTWD R3,#MINS,#MAXS
00040                       CMPB R0,(R3)    ;ID=SID(S)
00042                       BEQ  3$ ;FOUND
00044                       NXTMSG R1,#MINS,#MAXS
00076                       INC  R2
00100                       BR   1$
00102           2$:
00102                   CLR  R1 ;NOT FOUND
00104           3$:
00104                   MOV  (SP)+,R3
00106                   RTS  R5
                ;
                ;
                ;
00110           CHKSUM:
                        ;ON CALL R0-> FRONT OF MSG,R1->REAR
                ;               (R5)=MIN, 2(R5)=MAX ADDRESS OF QUEUE
                        ;RETURNS R2=CHECKSUM
00110                   MOV  R0,-(SP)
00112                   CLR  R2
00114           1$:
00114                   CMP  R0,R1
00116                   BEQ  2$
00120                       ADD  (R0)+,R2
00122                       NXTWD R0,(R5),2(R5)
00136                       BR   1$
00140           2$:
00140                   MOV  (SP)+,R0
00142                   ADD  #4,R5
00146                   RTS  R5
                ;
                ;
                ;
```

```
                        .SBTTL PARALLEL LINK MESSAGE HANDLING MACROS AND VARIABLES
                        .CSECT PINTPT
                        .MACRO SADDB X
                        MOVB X,(R0)+
                        NXTWD R0,#MINS,#MAXS
                        .ENDM
                ;
                        .MACRO SADD X
                        MOV  X,(R0)+
                        JSR  R5,ADDOQ    ;ADD TO OUTPUT QUEUE- CHECK OFLOW
                        .ENDM SADD
                ;
                        .MACRO RADD X
                        MOV  (R1)+,X
                        NXTWD R1,#MINR,#MAXR
                        .ENDM
                ;
                ;
                REQCAR=1
                ACKCA=2
                REQM=3
                SELINE=4
                SETCM=5
                CUTOFF=6
                PCHARS=7
                EXIT=9.
                RDY=11.
                SNDARR=12.
                RELINE=13.      ;RECEIVE LINE
                ;
```

```
                        .SBTTL PINTPT: VARIABLES
000000          PAR:    .BLKW 16.       ;PARALLEL COUNT FROM OTHER PROC.
000040          BASE:   .BLKW 16.       ;1 PROCESSOR COUNTS
000100          BUSY:   .WORD 0 ;BUSY SEMAPHORE FOR SYNC
000102          MAXCD:  .WORD 4 ;MAXIMUM COMMON DEPTH
000104          SWITCH: .WORD 0 ;SURROGATE SWITCH REGISTER
                SWREG=177570
000106          LANOIN: .WORD 0 ;LAST NO IN NODE
000110          HCA:    .WORD 0 ;HAVE CRITICAL AREA
                                ;SEMAPHORE PRIVATE TO THE MAIN PROCESS
                                ;USED AS PIPELINE FROM RECEIVE INTERRUPT
                                ;PROCESS ACKCA TO MAIN PROCESS
00112           CA:     .WORD 0 ;CRITICAL AREA
                                ;=1 WHEN CA UNUSED
                                ;=0 WHEN PROCESSOR IS IN CA
                                ;<0 WHEN REQUESTS FOR CA HAVE BEEN RECEIVED
00114           CM:     .WORD -1;COMMON MOVE
00116           OSC:    .WORD 0 ;NO OF OTHER PROCESSORS TO SCORE FOR
00120           LEAD:   .WORD 0
00122           PRX2:   .WORD 0 ;PROCESSOR CROSSING IN MINMX2
00124           ATTENT: .WORD 0 ;=-1 WHEN COMMUNICATED CUTOFF
                                ;=+VE WHEN SCORED NODE HAS BEEN RECEIVED
00126           FPCH:   .WORD MINPCH
00130           RPCH:   .WORD MINPCH
00132           MINPCH: .BLKW 100
00332           MAXPCH:
000332          QUOTE:  .ASCII /'/<TERM>
000334          CHARAC: .ASCII /0/<TERM>
                ;
                ;
```

```
                        .SBTTL SYNC: SYNCHRONIZE PROCESSORS
000336          SYNC:
                        ;SYNCHRONIZE BOTH PROCESSORS
000336          BIT   #1,SWITCH
000344          BEQ   50$
000346                  INC   BUSY
000352                  SAVREG 0,1,2
000360                  JSR   R5,OPNSE
000364                  SADD #RDY
00374                   CLR   R1
00376                   JSR   R5,CLSND
00402                   JSR   R5,WAITRP
00406                   RESREG 0,1,2
00414                   TST   BUSY
00420                   BNE   .-4
00422           50$:
00422                   RTS   R5
                ;
                ;
                ;
                        .SBTTL SENDAV: SEND AN AREA OF CORE
00424           SENDAV:
                        ;SEND ARRAY VARIABLE
                        ;ARRAY LOWER & UPPER LIMITS ARE (R5), 2(R5)
                        ;RECEIVING ADDRESS FOR OTHER PROCESSOR IS 4(R5)
00424                   SAVREG 0,1,2
00432                   JSR   R5,OPNSE
00436                   MOV   (R5)+,R1    ;FRONT
00440                   MOV   (R5)+,R2
00442                   SADD #SNDARR
00452                   SADD (R5)+        ;RECEIVING ADDRESS
00460           5$:
00460                   CMP   R1,R2
00462                   BHIS 10$
00464                       SADD (R1)+
00472                       BR    5$
00474           10$:
00474                   CLR   R1 ;AID=0
00476                   JSR   R5,CLSND
00502                   JSR   R5,WAITRP
00506                   RESREG 0,1,2
00514                   RTS   R5
                ;
                ;
                ;
```

ADDOQ: UPDATE PLINK OUTPUT BUFFER QUEUE POINTERS

```
                        .SBTTL ADDOQ: UPDATE PLINK OUTPUT BUFFER QUEUE POINTERS
000516          ADDOQ:
                        ;ADD TO OUTPUT QUEUE
                        ;CHECK FOR OVERFLOW
000516                  NXTWD R0,#MINS,#MAXS
000534                  CMP   R0,FOUT
000540                  BNE   10$
000542                          SPRINT  #M29    ;OUTPUT BUFFER OFLOW
000554          10$:
00554                   RTS   R5
                ;
                ;
                        .SBTTL WAIT REPLY
00556           WAITRP:
00556                   SAVREG 2
00560                   TST   -(SP)
00562           10$:
00562                   MOV   #600,(SP)
00566           20$:
00566                   JSR   R5,SRCHOUT
00572                   TST   R1
00574                   BEQ   30$
00576                   DEC   (SP)
00600                   BGT 20$
00602                           INC   RESEND
00606                           SAVREG 0,1
00612                           JSR   R5,SENDA
00616                           RESREG 0,1
00622                           BR    10$
00624           30$:
00624                   TST   (SP)+
00626                   RESREG 2
00630                   RTS   R5
                ;
                ;
                ;
```

```
                        .SBTTL P(CA): REQUEST CRITICAL AREA
 000632         REQCA:  ;LOOP UNTIL CRITICAL AREA IS OBTAINED
 000632         1$:
 000632                 DEC   CA
 000636                 BGE   30$
 000640                     SAVREG 0,1,2
 000646                 2$:
 000646                     JSR   R5,OPNSE    ;RETURNS R0
 000652                     SADD  #REQCAR
 000662                     CLR   R1 ;AID=0
 000664                     JSR   R5,CLSND    ;RETURNS R0=SID
 000670                     JSR   R5,WAITRP   ;WAIT REPLY
                            ;MESSAGE ACKNOWLEDGED
                            ;AWAIT CA>=0,TESTED FOR BY BACKGROUND PROCESS
                            ;AND WHICH SETS HCA PRIVATE FLAG TO 1
 000674                 6$:
 000674                     TST   HCA
 000700                     BLE   6$
                            ;FREE TO PROCEED
 000702                     DEC   HCA
 000706                     RESREG 0,1,2
 000714         30$:
 000714                 RTS   R5
                ;
                ;
                        .SBTTL V(CA): FREE CRITICAL AREA
 000716         FREECA:
                        ;FREE CRITICAL AREA CA+
 000716                 INC   CA
 000722                 BGT   35$
                            ;CRITICAL AREA REQUEST RECEIVED
 000724                     JSR   R5,SENDW
 000730                             .WORD ACKCA,0
 000734         35$:
 000734                 RTS   R5
                ;
                ;
                ;
                ;
```

```
                        .SBTTL REQMUPM: REQUEST UPDATED MOVE DATA
000736          REQMUPM:
                        ;PROCESSOR LAGS OTHER - REQUEST CURRENT MOVE FROM
                        ;BASE(D) AND WAIT (FOR SELIN)
                        ;FORMAT: REQM,UPM
000736                  SAVREG 0,1,2
000744                  JSR  R5,OPNSE
000750                  SADD  #REQM
000760                  SADD  R3        ;UPM(D)
00766                   CLR  R1 ;AID=0
00770                   JSR  R5,CLSND
00774                   JSR  R5,WAITRP
01000                   RESREG 0,1,2
01006                   CMPB NOIN(R3),#2
01014                   BGE  10$
01016                       CLR  @#PLSR
01022                       .PRINT #60$
01030                       .EXIT
01032           10$:
01032                   RTS  R5
01034           60$:    .ASCIZ /1 OR LESS PROCESSORS IN CA NODE/
                        .EVEN
                ;
                ;
                ;
                        .SBTTL SELIN: SEND LINE OF PLAY
01074           SELIN:  ;GIVEN R3=UPM(D), SEND INFORMATION ABOUT NODE:
                        ;SELIN, BASE(D),NOIN(D), VSCOR(D), BSTMVS(D,*)
01074                   SAVREG 0,1
01100                   JSR  R5,OPNSE
01104                   SADD #SELINE
                        ;TST  D ;IF D~=0 THEN BMV(D,*)=0 TEMPORARILY, AVOIDS SENDIN
                        ;BEQ  5$                                    ; MOVES
                        ;       MOV  <BMV-2>(R3),-(SP)
                        ;       CLR  <BMV-2>(R3)
01114           5$:
01114                   JSR  R5,SNODE
                        ;TST  D ;RESTORE BMV
                        ;BEQ  15$
                        ;       MOV  (SP)+,<BMV-2>(R3)
01120           15$:
01120                   CLR  R1 ;AID=0
01122                   JSR  R5,CLSND
01126                   RESREG 0,1
01132                   RTS  R5
                ;
                ;
                ;
```

```
                        .SBTTL SNODE: SEND NODE DATA
001134          SNODE:
                        ;R0->BUFFER
                        ;ADD BASE,NOIN(D),VSCOR(D),BSTMVS(D,*)
                        ; TO SEND BUFFER
                        ;R3->UPM(D)
001134                  SAVREG 3
001136                  SADD R3 ;BASE(D)
001144                  SADD NOIN(R3)
01154                   SADD VSCOR(R3)   ;VSCOR(D)
01164                   ADD  #BMV,R3     ;BSTMV(D,*)
01170           1$:
01170                   SADD -(R3)
01176                   TST  (R3)
01200                   BNE  1$
01202                   RESREG 3
01204                   RTS  R5
                ;
                ;
                        .SBTTL RNODE: RECEIVE NODE DATA
01206           RNODE:
                        ;R0->BUFFER
                        ;RECEIVE BASE,NOIN(D),VSCOR(D),BSTMVS(D,*)
                        ;FROM SEND BUFFER
                        ;R3->UPM(D)
01206                   SAVREG 3
01210                   RADD R3 ;BASE(D)
01230                   RADD NOIN(R3)
01252                   RADD VSCOR(R3)   ;VSCOR(D)
01274                   ADD  #BMV,R3     ;BSTMV(D,*)
01300           1$:
01300                   RADD -(R3)
01320                   TST  (R3)
01322                   BNE  1$
01324                   RESREG 3
01326                   RTS  R5
                ;
                ;
                ;
```

```
 001330         SENDCM:
 001330                 BIT   #1,SWITCH
 001336                 BEQ   10$
 001340                       MOV   CM,1$+2
 001346                       JSR   R5,SENDW
 001352                 1$:   .WORD SETCM,0,0
 001360         10$:
 001360                 RTS   R5
                ;
 01362          SENDCTF:
 01362                  MOV   (R3),1$+2  ;NODE FROM WHICH MOVES CUTOOFF
 01366                  JSR   R5,SENDW
 01372          1$:     .WORD CUTOFF,0,0
 01400                  RTS   R5
                ;
 01402          SENDEXIT:
 01402                  BIT   #1,SWITCH
 01410                  BEQ   10$
 01412                        JSR   R5,OPNSE
 01416                        SADD #EXIT
 01426                        SADD  R3
 01434                        CLR   R1
 01436                        JSR   R5,CLSND
 01442                        JSR   R5,WAITRP
 01446          10$:
 01446                  RTS   R5
                ;
 01450          SENDW:  ;PUT WORDS FOLLOWING R5 INTO OUTBUF
 01450                  SAVREG 0,1
 01454                  JSR   R5,OPNSE
 01460                  BR    2$
 01462                  1$:
 01462                        SADD (R5)+
 01470                  2$:
 01470                        TST   (R5)
 01472                        BNE   1$
 01474                  TST   (R5)+
 01476                  CLR   R1
 01500                  JSR   R5,CLSND
 01504                  RESREG 0,1
 01510                  RTS   R5
                ;
                ;
                ;
```

NTENTS OF RECEIVED MESSAGE

```
        .SBTTL INTPT CONTENTS OF RECEIVED MESSAGE
        .CSECT INTPTCONT
INTPTCONT:
        ;USE R0 AS SADD PTR, R1 AS RADD PTR IN MACROS
        ;INTERPRET CONTENTS OF RECEIVED MESSAGE
        ;ON CALL R0->TEXT, R1->END OF TEXT, R2=AID OF MESSAGE
        SAVREG 0,1,2
        MOV  R1,R2      ;END OF TEXT
        MOV  R0,R1
        JSR  R5,OPNSEA
        CMP  (R1),#REQCAR
        BNE  10$
             DEC  CA
             BLT  6$
                  ;CRITICAL AREA AVAILABLE
                  SADD #ACKCA
             6$:
                  JMP  INTP59$
10$:
        CMP  (R1),#ACKCA
        BNE  15$
             INC  CA
             BLT  13$
                  ;HAVE CRITICAL AREA
                  INC HCA ;PRIVATE SEMAPHORE TO PIPELINE
                          ;TO MAIN PROCESS
        13$:
             JMP  INTP59$
15$:
        CMP  (R1),#REQM
        BNE  18$
             SAVREG 3,4      ;WANT R3->UPM REQUESTED
             RADD -2(SP)     ;SKIP HEADER
             MOV  (R1),R3
             ;REPLY
             ;RELINE,NEWCM,CD,CBASE,NOIN(D),VSCOR(D),BSTMVS(D,*)]
             SADD #RELINE
             MOV  R3,R4      ;FIND NEWCM - THE WORD BELOW UPM(D+1)
             ADD  #PRESS,R4
        17$:
             CMP  -(R4),R3
             BNE  17$
             TST  (R4)+      ;WANT CM
             MOV  R4,CM      ;RESET CM
             SADD CM
             SADD CD
             INCB NOIN(R3)   ;ADD REQUESTING PROC TO COUNT
             JSR  R5,SNODE
             RESREG 3,4
             JMP  INTP59$
18$:
        CMP  (R1),#RELINE
        BNE  INTP20$
             ;THIS IS REPLY TO REQM
             ;RELINE,NEWCM,NEWCD,CBASE(D),NOIN(D),VSCOR(D),BSTMVS(D,*)
             RADD  -2(SP)    ;SKIP HEADER
             RADD CM
```

INTPT CONTENTS OF RECEIVED MESSAGE

```
5[illegible]00272                       RADD  CD
5[illegible]00314                       JSR   R5,RNODE
50 00320                                JMP INTP59$
51 00324        INTP20$:
52 00324                CMP   (R1),#SELINE
53 00330                BNE   INTP25$
54 00332                   RADD  -2(SP)     ;SKIP HEADER
55 00354                   JSR   R5,RNODE
56 00360                   TST   ATTENT     ;DON'T CALL UPSCOR WHEN ALREADY
57 00364                   BLT   22$        ;CUTOFF
58 00366                      INC   ATTENT
59 00372                22$:
70 00372                   JMP   INTP59$
71 00376        INTP25$:
72 00376                CMP   (R1),#SETCM
73 00402                BNE   INTP30$
74 00404                   RADD  -2(SP)
75 00426                   MOV   (R1),CM
76 00432                   CLR   LEAD
77 00436                   JMP   INTP59$
78 00442        INTP30$:
79 00442                CMP   (R1),#CUTOFF
80 00446                BNE   35$
81 00450                   RADD  -2(SP)
82 00472                   MOV   #-1,OSC    ;DISABLES SCORING
83 00500                   MOV   (R1),PRX2  ;NEWSC WILL RESTOR TO PRX2
[illegible]0504                   MOV   #-1,ATTENT
85 00512                   CLR   LEAD
86 00516                   JMP   INTP59$
87 00522        35$:
88 00522                CMP   (R1),#EXIT
89 00526                BNE   40$
90 00530                   RADD -2(SP)
91 00552                   SAVREG 3
92 00554                   RADD R3
93 00574                   DECB NOIN(R3)    ;DECREASE NO OF PROCESSORS IN NODE
94 00600                   RESREG 3
95 00602                   JMP   INTP59$
96 00606        40$:
97 00606        45$:
98 00606                CMP   (R1),#PCHARS       ;PARALLEL LINK CHARACTER TRANSFER
99 00612                BNE   INTP50$    ;MOVE CHARACTERS TO PLINK CHAR BUFFER
00 0614                   SAVREG 0
01 0616                   RADD  -2(SP)
02 0640                   MOV   RPCH,R0
03 0644                41$:
04 0644                   MOVB  (R1)+,(R0)+
05 0646                   BEQ   42$
06 0650                   NXTWD R1,#MINR,#MAXR
07 0666                   NXTWD R0,#MINPCH,#MAXPCH
08 0704                   BR    41$
[illegible] 0706                42$:
[illegible] 0706                   NXTWD R0,#MINPCH,#MAXPCH
11 0724                   MOV   R0,RPCH
12 0730                   RESREG 0
13 0732                   BR    INTP59$
14 0734         INTP50$:
```

```
0734                    CMP   (R1),#RDY
0740                    BNE   54$
0742                          DEC   BUSY
0746                          JMP INTP59$
0752            54$:
0752                    CMP   (R1),#SNDAKR
0756                    BNE   60$
0760                          SAVREG 3
0762                          RADD  R3          ;SKIP HEADER
1002                          RADD  R3          ;ADDRESS FOR PLACING DATA
1022                    56$:
1022                          CMP   R1,R2       ;R2=END OF TEXT
1024                          BEQ   58$
1026                                RADD (R3)+
1046                                BR    56$
1050                    58$:
1050                          RESREG 3
1052                          BR    INTP59$
1054            60$:
1054                    CLR   @#PLSR
1060                    SPRINT #M26       ;PARALLEL LINK MESSAGE NOT RECOGNISED
1072                    .EXIT
1074            INTP59$:
1074                    MOV   (SP),R1     ;AID
1076                    JSR   R5,CLSNDA
1102                    RESREG 0,1,2
1110                    RTS   R5
                ;
                ;
                ;
```

```
                        .SBTTL PARALLEL LINK CHARACTER ECHOING TO OTHER PROCESSOR
001112          READPCH:        ;READ FROM PARALLEL LINK CHR INPUT BUFFER
                        ;R1->RECEIVING BUFFER
001112                  SAVREG 0,1,2
001120          1$:
001120                  CMP  FPCH,RPCH
001126                  BEQ  4$
001130                          MOV  FPCH,R2
001134                          SPRINT #QUOTE
01146                   2$:
01146                           MOVB (R2),CHARAC
01152                           BEQ  5$ ;DON'T PRINT END MARKER
01154                           SPRINT #CHARAC  ;ECHO CHARACTERS
01166                   5$:
01166                           MOVB (R2)+,(R1)+
01170                           BEQ  3$
01172                           NXTWD R2,#MINPCH,#MAXPCH
01210                           BR  2$
01212                   3$:
01212                           SPRINT #QUOTE
                                ;ALL OF MESSAGE READ
01224                           NXTWD R2,#MINPCH,#MAXPCH
01242                           MOV  R2,FPCH
01246                           BR   1$
01250           4$:
01250                   RESREG 0,1,2
01256                   RTS  R5
                        ;
                        ;
                        ;
01260           PSEND:
                        ;PARALLEL SEND OF CHARACTER INPUT
                        ;R1->BEGINNING OF MESG , WHICH IS TERMINATED BY 0 BYTE.
01260                   BIT  #1,@#SWREG ;PARALLEL MODE ?
01266                   BEQ  30$
01270                   SAVREG 0,1
01274                   JSR  R5,OPNSE   ;RETURNS R0->BUFFER
01300                   SADD #PCHARS
01310           1$:
01310                   SADDB (R1)
01330                   TSTB (R1)+
01332                   BNE  1$
01334                   CLR  R1 ;NO AID
01336                   JSR  R5,CLSND
01342                   RESREG 0,1
01346           30$:
01346                   RTS  R5
                        ;
                        ;
                        ;
```

VARIABLES USED IN MINMAX, SETMVS,UPDATE & RESTORE

```
                        .SBTTL VARIABLES USED IN MINMAX, SETMVS,UPDATE & RESTORE
                        .CSECT MINVARS
                ;
                ;OFFSETS FROM BASE OF UPM
                        PRESS=-214.
                        LMV=214.        ;LENGTH OF MOVE AREA EXCLUDING PRESS AREA
                        LBMV=204.       ;LENGTH OF BEST MOVE PART OF MOVE AREA
                        BMV=-10.        ;OFFSET OF BEGINNING OF BESTMOVE AREA
                        CHK=-11.        ;DOES THE MOVE CHECK THE OPPONENT ?
                        NOIN=-10.       ;NO OF PROCESSORS IN NODE
                        SCOR=-8.        ;A SCORE BASED UPON MATERIAL
                        CPCE=-6 ;A POINTER TO THE PIECE CAPTURED BY THE MOVE
                        NIM=-4. ;NO OF IRRELEVANT MOVES
                                ;WHEN<0 GENERATE CAPTURES ONLY
                        VSCOR=-2        ;THE VIRTUAL SCORE -  THE BACKED UP SCORE
                        LUM=0   ;A POINTER TO THE PRECEDING UPDATED MOVE
                        SQ0=2   ;FROM SQUARE OF THE UPDATED MOVE
                        SQ1=4   ;TO SQUARE
                        MT=6    ;TYPE OF THE UPDATED MOVE
                ;
                ;VARIABLES USED IN MINMAX
00000           BOTM:   .WORD   ;BOTTOM OF MSTACK FOR CURRENT SET OF MOVES
00002           TOPCM:  .WORD   ;TOP OF CAPTURES STACK
00004           REPEAT: .WORD
00006           KCAP:   .WORD
00010           D:      .WORD   ;DEPTH
00012           CD:     .WORD 0 ;COMMON DEPTH
00014           CD0:    .WORD 0,0       ;INITIAL COMMON DEPTH FOR SEARCH
00020           VARCD:  .WORD 0 ;WHETHER TO DECREASE CD DURING SEARCH
00022           PBUF:   .BLKB 100
00122           LASTD:  .WORD 0 ;LAST PRINTED UPDATED DEPTH
00124           PDPTH:  .WORD 0 ;DEPTH TO WHICH TO PRINT BOARD
00126           UFLAG:  .WORD 0 ;TEST NO OF UPDATES SINCE LAST MOVE PRINTING
00130                   .WORD 0
00132                   .WORD W,SQUARE+<63.*LENSQ>      ;BKROOK
00136                   .WORD E,SQUARE+<8.*7*LENSQ>     ;BQROOK
                ;VARIABLES USED IN SETMVES
00142           BROOK:
00142                   .WORD 0
00144                   .WORD W,SQUARE+<7*LENSQ>        ;WKROOK
00150                   .WORD E,<SQUARE>        ;WQROOK
00154           WROOK:
00154           CLRKING:        .WORD
00156           OPPKING:        .WORD
00160           UPMOVE: .WORD 0 ;
                        .=.+200.        ;TEMPORARY STACK FOR HOLDING CAPTURES
                                ;AND ENSURING THEY ARE ON TOP OF THE MOVE STACK

00472           CMVE:   .WORD 0
00474           MTYPE:  .WORD
00476           SPEC:   .WORD
                ;VARIABLES USED IN SETMVS BUT ALSO USED ELSEWHERE
00500           BKING:  .WORD PIECE
00502           WKING:  .WORD PIECE+<16.*LENP>
00504           EPSQ:   .WORD
                ;VARIABLE USED IN UPDATE AND RESTORE
00506           RCAST:  .WORD SQUARE+0
```

VARIABLES USED IN MINMAX, SETMVS,UPDATE & RESTORE

```
58 00510                .WORD SQUARE+<3*LENSQ>
59 00512                .WORD SQUARE+<7*LENSQ>
60 00514                .WORD SQUARE+<5*LENSQ>
61 00516                .WORD SQUARE+<56.*LENSQ>
62 00520                .WORD SQUARE+<59.*LENSQ>
63 00522                .WORD SQUARE+<63.*LENSQ>
64 00524                .WORD SQUARE+<61.*LENSQ>
65                ;COUNTER FOR TOTAL # BOARD UPDATES
66 00526          NOUPS:  .WORD 0,0
67 00532          NOSETMVS:       .WORD 0,0
68 00536          NOCSTMVS:       .WORD 0,0
69 00542          NOMAXD: .WORD 0,0
70 00546          NOMMAXD:.WORD 0,0       ;NO OF CALLS TO CSTMVS TERMINATED BY MAXDD
71 00552          NBP:    .WORD 0,0       ;NO OF BOTTOM POSITIONS
72 00556          T2:     .WORD 0,0       ;MINMAX ELAPSED TIME
73 00562          ENDCO:
74 00562          NORCUT: .WORD 0,0
75 00566          NOMINMX2:.WORD 0,0
76                ;
77                ;
78                ;
```

```
                        .SBTTL MINIMAX PARALLEL TREE SEARCH
                        .CSECT MINMAX
 000000         MINMAX:
 000000                 BITB #1,SWITCH  ;PARALLEL SEARCH?
 000006                 BEQ  15$
 000010                     .PRINT #M27      ;PARALLEL SEARCH
 000016                     BIT  #2,SWITCH   ;LEAD ?
 000024                     BEQ  20$
 000026                         .PRINT #M28      ;LEAD SEARCH INITIALLY
0 00034                         BR   20$
1 00036                 15$:
2 00036                     .PRINT  #M33     ;SINGLE PROC SEARCH
3 00044         20$:
4 00044         MINMX1:
5                       ;COLOUR=THAT OF PLAYER TO MOVE
6               ;ON RETURN R3->UPM(0), & R4=TOP OF MSTACK.
7               ;ELAPSED TIME RETURNED IN T2
8               ;INITIALISE VARIABLES NOT ON THE MOVE STACK
9 00044                 STIME T1,T1+2
0 00116                 SAVREG 5
1                       ;SWAP ALPHA,BETA IF NECESSARY
2 00120                 TST  COLOUR
3 00124                 BGT  1$
4 00126                     CMP  ALPHA,BETA ;WANT ALPHA>BETA
5 00134                     BGE  3$
6 00136                     BR   2$ ;SWAP ALPHA,BETA
  00140                 1$:
8 00140                     CMP  ALPHA,BETA ;WANT ALPHA<BETA
9 00146                     BLE  3$
0 00150                 2$:     ;SWAP
1 00150                     MOV  ALPHA,R0
2 00154                     MOV  BETA,ALPHA
3 00162                     MOV  R0,BETA
4 00166         3$:
5                       ;MOV  MAXD,MMAXD        ;ABSOLUTE MAX DEPTH OF SEARCH
6                       ;ADD  #4,MMAXD
7 00166         MO$:
8 00166                 DCLR  NOUPS
9 00176                 DCLR  NOSETMVS
0 00206                 DCLR  NOCSTMVS
1 00216                 DCLR NOMAXD
2 00226                 DCLR NOMMAXD
3 00236                 DCLR NORCUT
4 00246                 DCLR NOMINMX2
5 00256                 DCLR NBP
6 00266                 CLR REPEAT      ;REPEAT=0
7 00272                 CLR  KCAP
8 00276                 CLR D
9 00302                 CLR  LASTD
0 00306                 CLR  NTAB
1 00312                 CLR UFLAG
2 00316                 JSR  PC,INVSTACK
3               ;LENPSQ WILL BE SET BY MINMAX IF NECESSARY
4               ;COLOUR IS THAT OF OPPONENT
5               ;R3=->UPM(0)
6               ;R4=->TOP OF MOVE STACK
7 00322                 CLR LEAD
```

```
58 00326                MOV  CDO,CD     ;INITIALISE CD
59 00334                BITB #1,SWITCH  ;PARALLEL SEARCH?
60 00342                BEQ  15$
61 00344                     BIT  #2,SWITCH  ;LEAD ?
62 00352                     BNE  12$
63                           ;NOT LEAD // SEARCH
64 00354                          JSR  R5,REQCA
65 00360                          BR  20$
66 00362                     12$:
67                                ;CRITICAL AREA IS ENTERED, AND CA=0
68 00362                          INC  LEAD
69 00366                          BR  20$
70 00370                     15$:
71 00370                          INC  LEAD
72 00374             20$:
73 00374                BR   MSTART     ;START
```

```
        ;
        ;
        ;DESCEND
        DESCEND:JSR  PC,UPDATE  ;R0 CONTAINS FROM, R4 POINTS TO IT
        ;START:
        MSTART:
                MOVB  #1,NOIN(R3)        ;NOIN(D)=1
                TST  NIM(R3)     ;IF ALLM(D) THEN DO;
                BLT  41$
                8$:
                        CMP  D,MAXDPTH  ;IF D<MAXDPTH THEN DO;
                        BGE  40$
                        ;R4=->PRESS(D,0)
                        MOV  FIRREL,R0  ;SQPRESS(D,*)=0
                        BNE  11$
                                ;NO NEED FOR RELEVANCE TESTING
                                JSR  PC,SETMVS
                                TST  KCAP
                                BEQ  50$
                                        JSR  R5,KCAPT
                                        CLR  LANOIN;NO PROCS AT D+1
                                        JMP  RETCALL
                        11$:    CLR  -(R4)       ;MAKE SPACE FOR RELEVANCE DATA
                                CLR  -(R4)
                                MOV  NXTR(R0),R0
                                BNE  11$
                12$:    JSR  PC,SETMVS
                        ;R4=->TOP OF MOVE STACK
                        TST  KCAP
                        BEQ  14$
                                TST  D  ;RETURN IF DEPTH=0 SINCE ILLEGAL POSITION
                                BEQ  13$
                                        JSR  R5,KCAPT
                                        CLR  LANOIN;NO PROCS AT D+1
                                        JMP  RETCALL
                                13$:    JMP  M150$       ;D=0, ILLEGAL POSITION
                14$:
                        INC  REPEAT
                        NEG  COLOUR
                        JSR  PC,SETMVS  ;NO MOVES INSERTED, R4 UNCHANGED
                        CLR REPEAT
                        NEG COLOUR       ;COLOUR=-COLOUR
                        TST  D
                        BEQ  50$
                        JSR  PC,EVAL
                        BCC  50$
                        JMP  RETCALL     ;BOARD RESTORED
                40$:
                        DINC NOMAXD,NOMAXD+2
                        MOV  #-1,NIM(R3)
                41$:    ;ELSE DO;
                        DINC NBP,NBP+2
                        CMP  D,MMAXD
                        BLT  43$
                                DINC NOMMAXD,NOMMAXD+2
                                CLR  -(R4)
```

```
00662                                   CLR  LANOIN      ;NOIN(D+1)=0
00666                                   BR   50$
00670                   43$:
00670                           JSR  PC,CSTMVS
00674                           TST  KCAP
00700                           BEQ  49$
00702                                JSR  R5,KCAPT
00706                                     CLR  LANOIN;NO PROCS AT D+1
00712                                JMP  RETCALL
00716                           49$:
00716                                MOV  #DUMMY,-(R4)
00722                                CLR  LANOIN      ;NOIN(D+1)=0
00726           50$:
00726                   TST  LEAD
00732                   BGT  M55$
00734                        JSR  R5,REQMUPM ;RETURN MADE AFTER REPLY RECEIVED
                             ;DOES THE NEW VSCOR CAUSE A CUTOFF ?
00740                        CLR  PRX2       ;PRX2=0
00744                        JSR  R5,UPSCOR  ;LOOKFOR CUTOFFS & UPDATE SCORES
00750                        TST  PRX2
00754                        BEQ  54$
00756                             JMP  M130$      ;GO TO CUTOFF
00762                   54$:
00762                        MOV  CM,R4
00766           M55$:
                ;PRINT UPDATED MOVE
00766                   INC  UFLAG      ;SHOWS UPDATE MADE SINCE LAST PVSCOR
00772                   CMP  D,PDPTH
01000                   BGT  MLOOP
01002                        TST  D
01006                        BEQ  MLOOP
01010                        MOV  #PBUF,R0
01014                        JSR  R5,CHARMV
01020                        JSR  R5,PRINTMV
                ;
                ;
                ;
```

```
 01024          MLOOP:
                ;
 001024                 CLR   OSC          ;OTHER SCORE=0
 001030                 TST   LEAD
 001034                 BGT   10$
 001036                       CMP   CM,R3
 001042                       BLOS  3$
 001044                             JMP M140$
 001050                       3$:
 0 01050                             MOV   CM,R4
 1 01054                             TST   (R4)         ;IN SPECIAL CASE OF CUTOFF
 2 01056                             BNE   4$           ;(R4) will be set.
 3                                        ;0 AT END OF SEARCH
 4 01060                                  JMP   M140$
 5 01064                       4$:
 6 01064                             CMP   D,CD
 7 01072                             BGE   5$
 8 01074                                  JMP   DESCEND
 9 01100                             5$:
 0 01100                                  ADD   #6,R4        ;M+
 1 01104                                  INC   LEAD
 2 01110          10$:
 3 01110                 CMP   R4,CM
 4 01114                 BLOS  20$
 5 01116                       MOV   R4,CM
 6 01122                       JSR   R5,SENDCM
 01126          20$:
 01126                 MOV   (R4),R1      ;IF FROM(M)=0 THEN GOTO BREAK
 9 01130                 BEQ   M140$
 0 01132                 CMP   R1,#DUMMY    ;ELSE IF FROM(M)^=DUMMY THEN GOTO DESCEND
 1 01136                 BNE   31$
 2 01140                       TST   (R4)+        ;MOVE PAST DUMMY TO NEXT MOVE
 3 01142                       CLR   LANOIN       ;NOIN(D+1)=0
 4 01146                       MOV   SCOR(R3),R0          ;VSCOR(D+1)=SCOR(D)
 5 01152                       BR    RETCALL
 6 01154          31$:
 7 01154                 CMP   D,CD
 8 01162                 BGE   33$
 9 01164                 JMP   DESCEND
 0 01170                 33$:
 1 01170                       CLR   PRX2         ;REINITIALISE, MAY BE SET .
 2 01174                       CLR   ATTENT       ;DUE TO SCOR COMING AFTER SEARCH
 3 01200                       DINC  NOMINMX2,NOMINMX2+2
 4 01212                       MOV   R3,-2(R4)    ;SAVE BACK PTR IN CASE OF REQM(D)
 5 01216                       JSR   R5,FREECA
 6 01222                       JSR   PC,MINMX2
 7 01226                       JSR   R5,REQCA
 8 01232                       TST   ATTENT ;COULD HAVE BEEN NEW SCORE WHICH
 9 01236                       BEQ   35$    ;INVALIDATES SCORE.
 0 01240                            JSR   R5,NEWSC
 1 01244                 35$:
 01244                       CLR   LANOIN       ;NOIN(D+1)=0
```

```
        RETCALL:
        ;R0=VSCOR(D+1)
                CMPB OSC,LANOIN ;IF OSC=NOIN(D+1) THEN EVALUATE
                BNE  MLOOP
        39$:
                MOV  (R3),R2
                TST COLOUR
                BGT  38$
                        ;BLACK
                        CMP  VSCOR(R3),VSCOR(R2)
                        BLE  44$        ;UNEXPECTED CUTOFF
                        CMP  R0,VSCOR(R3)       ;VSCOR(D)=VSCOR(D+1)
                        BGE  MLOOP
                                MOV  R0,VSCOR(R3)
                                CMP  R0,VSCOR(R2)       ;IF VSCOR(D)<=VSCOR(D-1)
                                BLE  M130$      ;GOTO CUTOFF
                                CMP  R1,#DUMMY  ;IF R1^=#DUMMY THEN DO;
                                BEQ  MLOOP
                                        JSR  PC,BSTMVS  ;ELSE COPY BESTMOVES
                                        CMP  CM,R3      ; UP STACK.
                                        BHIS MLOOP
                                                JSR  R5,SELIN
                                                BR   MLOOP
                38$:
                        CMP  VSCOR(R3),VSCOR(R2)
                        BGE  44$        ;UNEXPECTED CUTOFF
                        CMP  R0,VSCOR(R3)
                        BLE  MLOOP
                                MOV  R0,VSCOR(R3)
                                CMP  R0,VSCOR(R2)
                                BGE  M130$
                                CMP  R1,#DUMMY  ;IF R1^=#DUMMY THEN DO;
                                BEQ  40$
                                        JSR  PC,BSTMVS
                                        CMP  CM,R3
                                        BHIS 40$
                                                JSR  R5,SELIN
                                        40$:
                                                JMP  MLOOP
                        44$:
                                        .PRINT  #M46    ;UNEXPECTED CUTOFF
                                        JSR  R5,PDUMP0  ;PRINT STATUS
                                        .EXIT
        ;
        ;CUTOFF
        M130$:
                DECB NOIN(R3)
                BLE  M145$
                        MOVB NOIN(R3),OSC
                        JSR  R5,SENDCTF
                        INC  LEAD
                BR   M145$
        ;BREAK:
        M140$:
                DECB NOIN(R3)   ;NOIN(D)=NOIN(D)-1
                BLE  M145$
                        JSR  R5,SENDEXIT
```

```
        M145$:
                MOV   D,R0          ;D=0?
                BEQ   M150$
                        CMP   R0,PDPTH    ;PRINT THE SCORE
                        BGT   42$
                        BEQ   41$
                                TST   UFLAG
                                BEQ   42$
                                41$:    JSR   R5,PVSCOR
                                        CLR   UFLAG
        42$:
                MOVB NOIN(R3),LANOIN      ;LAST NO. IN= NOIN(D)
                JSR   PC,RESTOR   ;RETURNS R0=VSCOR(D+1).
                CMP   D,CD          ;IF D<=CD THEN (CD=D;IF CD= 0 THEN CD=MAXCD;);
                BGT   45$
                        TST   VARCD       ;VARIABLE COMMON DEPTH ?
                        BEQ   45$
                        MOV   D,CD
                        BNE   45$
                                MOV   CD0,CD
        45$:
                BR    RETCALL
        ;
        ;
        M150$:
                JSR   R5,SENDCM   ;GIVES OTHER PROC UP TO DATE CM - THERE MAY BE
                                  ;ILLEGAL MOVES WHICH NEED SKIPPING
                JSR   R5,FREECA
                STIME  T3,T3+2    ;MEASURE BUSY TIME
                JSR   R5,SYNC
                CLR   CA ;CLEAR CA TO EXCLUDE INITIAL PROBLEM NEXT TIME
                MOV   #-1,CM
                DMOV WDSIN,WDSI
                DMOV WDSOUT,WDSO
                DCLR WDSIN
                DCLR WDSOUT
                ;WANT ALPHA<BETA
                CMP   ALPHA,BETA
                BLT   52$
                        ;SWAP ALPHA,BETA
                        MOV   ALPHA,R0
                        MOV   BETA,ALPHA
                        MOV   R0,BETA
        52$:
                STIME T2,T2+2
                DSUB  T2,T2+2,T3,T3+2   ;BUSY TIME      ;BUSY TIME =-(T3-T2)
                DNEG T3,T3+2
                DSUB T1,T1+2,T2,T2+2
                RESREG 5
                RTS   PC ;BEST MOVES ARE IN BSTMVE(0,*)
        ;
        ;
        ;
```

```
                        .SBTTL INITIALISE MOVE STACK
002136          IMVSTACK:
                ;INITIALISE STACK
002136                  MOV    #MSTACK,R4
002142                  CLR   -(R4)        ;LUM(-1)=0
002144                  MOV   BETA,-(R4)   ;VSCOR(-1)=BETA
002150                  CLR   -(R4)        ;MT(D=0)=0
002152                  CLR   -(R4)        ;SQ1(0)=0
002154                  CLR   -(R4)        ;SQ0(0)=0
02156                   MOV   #MSTACK-2,-(R4)     ;LUM(0)=->LUM(-1)
02162                   MOV   R4,R3        ;R3=->UPMOVE
02164                   MOV ALPHA,-(R4)    ;VSCOR(0)=ALPHA
02170                   MOV   MAXNIM,-(R4)         ;NIM(0)=MAXNIM
02174                   CLR   -(R4)        ;CPCE(0)=
02176                   MOV   SCORE,-(R4)          ;SCOR(0)=SCORE
02202                   CLR   -(R4)        ;CHK(0)=0
02204                   CLR   -(R4)        ;BMVS(0,*)=0
02206                   SUB #LBMV-2,R4     ;RESERVE PLACE FOR BESTMOVES(0,*)
02212                   RTS   PC
                ;
                ;
                ;
```

```
                .SBTTL NEWSC: HANDLE COMMUNICATED SCORES AND CUTOFFS
                ;IF A CUTOFF IS FOUND RESTOR BOARD.
                ;ON CALL ATTENT=-1 WHEN CUTOFF RECEIVED,
                ;       & PRX2 IS THEN ALREADY SET IN BACKGROUND
                ;       >0 WHEN ONLY NEW SCORE(S) RECEIVED
        NEWSC:
                TST   ATTENT      ;IF ATTENT>0 THEN CALL UPSCOR
                BLE   20$
                      ;JSR  R5,PRINVS
                      JSR  R5,UPSCOR
                      ;JSR  R5,PRINVS
        20$:
                TST   PRX2        ;RESTORE BOARD ?
                BEQ   50$
                      ;MOV  R3,55$
                      ;SPRINT #NEWLINE
                      ;JSR  R5,PRNOS
                      ;       .WORD 0,8.,56$,55$,61$,ATTENT,57$,PRX2,D8,D,0
                30$:
                      CMP   R3,PRX2     ;WHILE (BASE(D)<PRX2)
                      BHIS 49$
                            JSR   PC,RESTOR  ;CALL RESTOR
                            BR    30$
                      49$:
                            ;MOV  R3,55$
                            ;JSR  R5,PRNOS
                            ;       .WORD 0,8.,60$,D,56$,55$,0
        50$:
                CMP   D,CD        ;IF D<=CD THEN (CD=D;IF CD= 0 THEN CD=MAXCD;);
                BGT   45$
                      TST   VARCD       ;VARIABLE COMMON DEPTH ?
                      BEQ   45$
                      MOV   D,CD
                      BNE   45$
                            MOV   CD0,CD
        45$:
                CLR   PRX2        ;PRX2=0
                CLR   ATTENT
                RTS   R5
        ;55$:.WORD 0
        ;56$:   .ASCIZ /R3=/
        ;57$:   .ASCIZ /PRX2=/
        ;60$:   .ASCIZ /FINAL D=/
        ;61$:   .ASCIZ /ATTENT=/
        ;       .EVEN
        ;
        ;
        ;
                .SBTTL PRINVS: PRINT VSCORES ON STACK
        PRINVS: ;PRINT VSCOR STACK
                TST   MSTACK-2
                BEQ   5$
                      MOV   #NOINT,@#PLSR
                      .PRINT #59$
                      .EXIT
        5$:
                SAVREG 3,0
```

PRINVS: PRINT VSCORES ON STACK

```
58 02346                CLR   -(SP)
59 02350        10$:
60 02350                MOV   R3,-(SP)
61 02352                MOV   (R3),R3
62 02354                BNE   10$
63 02356                .PRINT #NEWLINE
64 02364                .PRINT #D8
65 02372                MOV   SP,R0
66 02374        20$:
67 02374                MOV   (R0)+,R3
68 02376                BEQ   25$
69 02400                   MOV   R3,58$
70 02404                   BIC   #100000,58$          ;ADDRESS NOT NEGATIVE
71 02412                   JSR   R5,PRNO
72 02416                         .WORD 0,8.,58$,0
73 02426                   BR    20$
74 02430        25$:
75 02430                .PRINT #60$
76 02436                MOV   SP,R0
77 02440        30$:
78 02440                MOV   (R0)+,R3
79 02442                BEQ   35$
80 02444                   MOV   VSCOR(R3),58$
81 02452                   JSR   R5,PRNO
82 02456                         .WORD 0,8.,58$,0
83 02466                   BR    30$
84 02470        35$:
85 02470                MOV   R0,SP
86 02472                RESREG 3,0
87 02476                RTS   R5
88 02500        58$:    .WORD 0,0
89 02504        59$:    .ASCIZ /NON-ZERO PTR AT D=-1/
90 02531        60$:    .ASCIZ <CR><LF>/VSCOR=/<TERM>
91                      .EVEN
92                      ;
93                      ;
94                      ;
95                      .SBTTL UPSCOR: PASS COMMUNICATED SCORE DOWN MOVE STACK
96 02544        UPSCOR:
97                      ;PASS COMMUNICATED SCORE DOWN STACK UNTIL
98                      ;EITHER ORIGINAL DEPTH IS REACHED OR A CUTOFF IS FOUND
99 02544                SAVREG 0,1,2,3
100 2554                SUB   #6,SP
101             CLRD=0
102             NEWSC0=2        ;BLACK BETA VALUE
103             NEWSC1=4        ;WHITE ALPHA VALUE
104 2560                MOV   SP,R0       ;BASE R0
105                     ;FIND BASES DOWN TO DEPTH =-1
106 2562                CLR   -(SP)       ;END MARKER
107 2564                MOV   R3,R2
108 2566        10$:                              ;LOOP
109 2566                   MOV   R2,-(SP)    ;        STACK<-BASE(D)
110 2570                   MOV   (R2),R2     ;        D=D-1
111 2572                   BNE   10$         ;UNTIL (D=-2)
112 2574                MOV   (SP)+,R1    ;D'-1=-1
113 2576                MOV   (SP)+,R2    ;D' =0
114 2600                CMP   VSCOR(R1),VSCOR(R2) ;IF VSCOR(-1)<VSCOR(0) THEN DO;
```

```
1[illegible] 2606                    BGT   15$
1[illegible]                                ;BLACK MOVES FROM D'=0
117 2610                                    MOV   #-1,CLRD(R0)          ;CLR'=-1
118 2616                                    MOV   VSCOR(R2),NEWSC0(R0)          ;NEWSC(0)=VSCOR(0)
119 2624                                    MOV   VSCOR(R1),NEWSC1(R0)          ;NEWSC(1)=VSCOR(-1
120 2632                                    BR    20$
121 2634                             15$:   ;WHITE AT D'=0
122 2634                                    MOV   #1,CLRD(R0)           ;CLR'=1
123 2642                                    MOV   VSCOR(R1),NEWSC0(R0)          ;NEWSC(0)=VSCOR(-1
124 2650                                    MOV   VSCOR(R2),NEWSC1(R0)          ;NEWSC(1)=VSCOR(0)
125 2656                     20$:
```

PASS COMMUNICATED SCORE DOWN MOVE STACK

```
        UP20$:
        20$:            ;LOOP
                NEG   CLRD(R0)     ;CLR'=CLR'-1
                MOV   R2,R1        ;D'=D'+1
                MOV   (SP)+,R2     ;IF D'>D THEN LEAVE LOOP
                BEQ   50$
                TST   CLRD(R0)
                BGT   35$
                    ;BLACK
                    CMP   NEWSC0(R0),VSCOR(R2) ;IF NEWSC0<VSCOR(D') THEN DO;
                    BGE   25$
                        MOV   NEWSC0(R0),VSCOR(R2) ;VSCOR(D')=NEWSC(0)
                        CLR   BMV-2(R2)   ;BSTMVS(D',*)=0
                        BR    30$
                    25$:
                        MOV   VSCOR(R2),NEWSC0(R0) ;ELSE NEWSC0=VSCOR(D')
                30$:    ;CUTOFF ?
                    CMP   VSCOR(R2),VSCOR(R1) ;IF VSCOR(D')<=VSCOR(D'-1) THEN
                    BGT   20$                 ;DO
                        MOV   R1,PRX2     ;PRX2=D'-1
                        BR    49$         ;LEAVE A LOOP AND RESTOR STACK
                35$:    ;WHITE
                    CMP   NEWSC1(R0),VSCOR(R2) ;IF NEWSC1>VSCOR(D') THEN DO;
                    BLE   40$                 ;DO
                        MOV   NEWSC1(R0),VSCOR(R2) ;VSCOR(D')=NEWSC1
                        CLR   BMV-2(R2)   ;BMV(D',*)=0
                        BR    45$
                    40$:
                        MOV   VSCOR(R2),NEWSC1(R0);ELSE NEWSC1=VSCOR(D')
                45$:
                    CMP   VSCOR(R2),VSCOR(R1) ;IF VSCOR(D')>=VSCOR(D'-1) THEN
                    BLT   20$                 ;DO
                        MOV   R1,PRX2     ;PRX2=D'-1
                49$:
                    TST   (SP)+
                    BNE   49$
                ;REPEAT
        50$:
                ADD   #6,SP
                RESREG  0,1,2,3
                RTS   R5
        ;
        ;
        ;
```

```
                .SBTTL MINMX2
                .CSECT MINMX2
        ;
        ;
        ;DESCEND
        MINMX2:
                JSR   PC,UPDATE   ;R0 CONTAINS FROM, R4 POINTS TO IT
                TST   NIM(R3)     ;IF ALLM(D) THEN DO;
                BLT   21$
        8$:
                    CMP   D,MAXDPTH   ;IF D<MAXDPTH THEN DO;
                    BGE   20$
                    ;R4=->PRESS(D,0)
                    MOV   FIRREL,R0   ;SQPRESS(D,*)=0
                    BNE   11$
                        ;NO NEED FOR RELEVANCE TESTING
                        JSR   PC,SETMVS
                        TST   KCAP
                        BEQ   25$
                            JSR   R5,KCAPT
                            JMP   RETCL1      ;GOTO RETCALL
                    11$:    CLR   -(R4)       ;MAKE SPACE FOR RELEVANCE DATA
                        CLR   -(R4)
                        MOV   NXTR(R0),R0
                        BNE   11$
                12$:    JSR   PC,SETMVS
                    ;R4=->TOP OF MOVE STACK
                    TST   KCAP
                    BEQ   14$
                        JSR   R5,KCAPT
                        JMP   RETCL1
                14$:
                    INC   REPEAT
                    NEG   COLOUR
                    JSR   PC,SETMVS   ;NO MOVES INSERTED, R4 UNCHANGED
                    CLR REPEAT
                    NEG COLOUR        ;COLOUR=-COLOUR
                    JSR   PC,EVAL
                    BCC   25$
                        JMP   RETCL1      ;BOARD RESTORED- GOTO RETCALL
                20$:
                    DINC  NOMAXD,NOMAXD+2
                    MOV   #-1,NIM(R3)
                21$:    ;ELSE DO;
                    ;DINC NBP,NBP+2
                    CMP   D,MMAXD
                    BLT   23$
                        DINC NOMMAXD,NOMMAXD+2
                        CLR   -(R4)
                        MOV   #DUMMY,-(R4)
                        BR  25$
                23$:
                    TST   FIRREL
                    BEQ   24$
                        JSR   PC,EVAL     ;SEE IF OCCUPATION OF REL SQ
                24$:
                    MOV   #<DUMMY-1>,-(R4)    ;INDICATOR THAT CSETMVS HAS NOT
                                              ; YET BEEN CALLED.
```

```
                                        ;BEEN CALLED
                        MOV  #DUMMY,-(R4)       ;TRY STATIC EVALN FIRST
                        ;JSR  PC,CSTMVS
                        ;TST  KCAP
                        ;BEQ  25$
                        ;       JSR  R5,KCAPT
                        ;       JMP  RETCL1
        25$:
                ;PRINT UPDATED MOVE
                INC  UFLAG      ;SHOWS UPDATE MADE SINCE LAST PVSCOR
                CMP  D,PDPTH
                BGT  M230$
                        MOV  #PBUF,R0
                        JSR  R5,CHARMV
                        JSR  R5,PRINTMV
        M230$:
                .ENABL LSB
        ;
        ;
        ;MLOOP
        ;
        30$:
                MOV  (R4),R1    ;IF FROM(M)=0 THEN GOTO BREAK
                BEQ  40$
                CMP  R1,#DUMMY  ;ELSE IF FROM(M)^=DUMMY THEN GOTO DESCEND
                BHI  MINMX2
                BEQ  31$
                        ;(R4)=<#DUMMY-1>, SO NEED TO GENERATE MOVES.
                        TST  (R4)+      ;REMOVE INDICATOR
                        JSR  PC,CSTMVS
                        TST  KCAP
                        BEQ  30$        ;LEGAL UPDATE
                                JSR  R5,KCAPT
                                JMP  RETCL1
                31$:
                        MOV  SCOR(R3),R0        ;VSCOR(D+1)=SCOR(D)
                        TST  (R4)+
        ;RETCALL:
        RETCL2:
        33$:
        34$:
        ;R0=VSCOR(D+1)
                MOV  (R3),R2
                TST COLOUR
                BGT  35$
                        ;BLACK
                        CMP  VSCOR(R3),VSCOR(R2)
                        BLE  38$
                        CMP  R0,VSCOR(R3)       ;VSCOR(D)=VSCOR(D+1)
                        BGE  30$
                                MOV  R0,VSCOR(R3)
                                CMP  R0,VSCOR(R2)       ;IF VSCOR(D)<=VSCOR(D-1)
                                BLE  40$        ;GOTO BREAK
                                CMP  R1,#DUMMY  ;IF R1^=#DUMMY THEN DO;
                                BEQ  30$
                                        JSR  PC,BSTMVS  ;ELSE COPY BESTMOVES UP
                                BR   30$
```

```
2                35$:
2                       CMP   VSCOR(R3),VSCOR(R2)
0                       BGE   38$
2                       CMP   R0,VSCOR(R3)
6                       BLE   30$
0                           MOV   R0,VSCOR(R3)
4                           MOV   (R3),R2
6                           CMP   R0,VSCOR(R2)
2                           BGE   40$
4                           CMP   R1,#DUMMY  ;IF R1^=#DUMMY THEN DO;
0                           BEQ   30$
2                               JSR   PC,BSTMVS
6                           BR    30$
0                       38$:
0                           TST   ATTENT      ;CUTOFF DUE TO RECEIVED SCORE
4                           BNE   40$         ;ALLOW CUTOFFS AFTER RECEIVED
6                               .PRINT  #M47    ;UNEXPECTED CUTOFF
4                               MOV   #NOINT,@#PLSR
2                               JSR   R5,PRNOS
6                               .WORD 0,10.,D35,D,0
0                               .EXIT
            ;
            ;
            ;BREAK:
2           40$:
2                       CMP   D,PDPTH     ;PRINT THE SCORE
0                       BGT   42$
2                       BEQ   41$
4                           TST   UFLAG
0                           BEQ   42$
2                           41$:    JSR   R5,PVSCOR
6                               CLR   UFLAG
2           42$:
2               JSR   PC,RESTOR  ;RETURNS R0=VSCOR(D+1).
6           RETCL1:
6               TST   ATTENT
2               BEQ   50$
4                   JSR   R5,REQCA
0                   JSR   R5,NEWSC
4                   JSR   R5,FREECA
0           50$:
0               CMP   D,CD
6               BGT   33$
0               RTS   PC ;BEST MOVES ARE IN BSTMVE(CD,*)
                .DSABL LSB
            ;
            ;
            ;
```

VALUATE SCORE AFTER UPDATED MOVE

```
                .SBTTL EVAL: EVALUATE SCORE AFTER UPDATED MOVE
2       EVAL:   ;EVALUATE SCORE AFTER UPDATED MOVE
                ;BY FINDING IF MOVE WAS RELEVANT
                        ;TEST FOR RELEVANCE: CPIECE ?, REL(TO), RP(TO),CHECK
2                       CLR   R0 ;DS=0
4                       MOV   SQ0(R3),R1 ;SQ0 REL ?
0                       TST   REL(R1)
4                       BEQ   18$
                            ;FROM REL SQ
6                           MOV   #-PCNT,R0
2                           MOV   SQ1(R3),R1
6                           TST   REL(R1)
2                           BEQ   50$
4                           BR    20$
6               18$:
6                       MOV   SQ1(R3),R1 ;R1=TO
2                       TST   REL(R1)    ;REL SQ(TO)?
6                       BEQ   22$
0                       20$:
0                           ADD   #PCNT,R0
4                           TST   CPCE(R3)
0                           BEQ   50$
2                               ADD   #PCNT,R0
6                               BR    50$
0               22$:
0                       MOV   (R1),R1    ;RP(TO) ?
2                       BEQ   24$
4                           TST   REL(R1)
0                           BNE   50$
2               24$:
2                       CMP   D,MAXDPTH  ;ONLY EVAL OCCUPATION IF D>MAXD SINCE MO
0                       BGE   50$        ;NOT GENERATED YET
        ;
2                       MOV   FIRREL,R0
6                       MOV   R3,R1      ;IS THE PRESSURE ON ANY RP OR REL SQ
0                       ADD   #PRESS,R1  ;R1=->PRESS(D,0)  DIFFERENT?
4                       MOV   (R3),R2
6                       ADD   #PRESS,R2  ;R2->PRESS(D-1,0)
2                       26$:
2                           MOV   -(R2),R5   ;PRESS(D)& ^PRESS(D-1)=0?
4                           COM   R5
6                           BIT   -(R1),R5
0                           BNE   25$
2                           MOV   -(R2),R5   ;PRESS(D)& ^PRESS(D-1)=0?
4                           COM   R5
6                           BIT   -(R1),R5
0                           BNE   25$
2                       MOV   NXTR(R0),R0        ;R0=->NXTR SQ OR P
6                       BNE   26$
0                       BR    27$
2                           25$:
2                               MOV   #TEMP0,R0  ;IF PRESS(D)&^PRESS(D-1)
4                               BR    50$
0               27$:
0                       CLR   R0
2                       CMP   MT(R3),#THR        ;CASTLE,EPCAP,QPAWN OR NPAWN ?
0                       BGT   50$
```

```
                    TST   CPCE(R3)
                    BNE   50$
                    TSTB  CHK(R3)      ;CHECKS ARE RELEVANT
                    BNE   50$
                    MOV   (R3),R2      ;R2 USED ABOVE BUT LOST
                    TSTB  CHK(R2)      ;REPLIES TO CHECK ARE RELEVANT
                    BNE   50$
                    ;IRRELEVANT MOVE
                    SUB   PEN,R0       ;IRRELEVANT MOVE PENALTY
                    ADD   D,R0         ;DEPTH BONUS
                    DEC   NIM(R3)
                    BGE   50$
                        DINC  NORCUT,NORCUT+2
                        JSR   PC,RESTOR  ;RESTOR BOARD & GOTO NEXT MOVE
                        MOV   SCOR(R3),R0          ;VSCOR(D+1)=SCOR(D)+PENAL
                        TST   COLOUR
                        BGE   28$
                            SUB   D,R0;DEPTH BONUS
                            ADD   PEN,R0
                            BR    29$
                        28$:
                            ADD   D,R0
                            SUB   PEN,R0;IRRELEVANT MOVE PENALTY
                    29$:
                        SEC
                        RTS   PC
            50$:
                    TST   COLOUR
                    BGT   52$
                        ADD   R0,SCOR(R3)
                        BR    55$
                    52$:
                        SUB   R0,SCOR(R3)
            55$:
                    CLC
                    RTS   PC
        ;
        ;
        ;
```

```
                .SBTTL KING CAPTURED
KCAPT:  ;KING IS CAPTURED,
                ;LAST UPDATED MOVE WAS ILLEGAL
                ;SINCE IT MOVED PLAYER INTO CHECK
                JSR   PC,RESTOR
                MOV   SCOR(R3),R0         ;VSCOR(D+1)=SCOR(D)-PIECE(P1).VAL
                MOV   KCAP,R1
                SUB   VAL(R1),R0
                ;ENCOURAGE FAST KING CAPTURES
                TST   COLOUR
                BGT   35$
                        ;BLACK HAS HIS KING CAPTURED:GIVE BLACK DEPTH BON
                        SUB   D,R0
                        BR    36$
                35$:
                        ADD   D,R0
        36$:
                MOV   #DUMMY,R1   ;PREVENTS CALL TO BESTMVS
                CLR   KCAP
                RTS   R5
;
;
;
```

```
                        .SBTTL BESTMOVES: COPY THEM DOWN FROM D+1 TO D
                        .CSECT BSTMVS
000000          BSTMVS:
                ;IF R1=$DUMMY THEN
                ;       BSTMVE(D,*)=0
                ;ELSE
                ;       BSTMVE(D,*)=MOVE(M)& BSTMVE(D+1,*)
                ;R1=TYPE OF LAST MOVE
                ;R3=->UPM
                ;R4=->TYPE+2 OF LASTM (TOP OF MOVE STACK) AT DEPTH D TO D+1
                ;COPY THIS MOVE DOWN UNTIL A PTR TO R3=UPM(D) IS OBTAINED
                ;THEN COPY BSTMVE(D+1)->BSTMVE(D)
00000                   MOV R2,-(SP)
00002                   MOV R3,R2           ;R2=->BSTMVE(D)
00004                   ADD $BMV,R2
00010                           MOV R4,-(SP)    ;ELSE
00012                   2$:     MOV -(R4),-(R2)
00014                           CMP (R4),R3
00016                           BNE 2$
00020                           TST (R2)+       ;DELETE PTR FROM MOVE STACK
                        ;R4=->MOVE(D+1)
00022                           ADD $BMV,R4     ;R4=->BSTMVE(D+1,*)
00026                   3$:     MOV -(R4),-(R2)
00030                           BNE 3$
00032                           MOV (SP)+,R4
00034           4$:     MOV (SP)+,R2
0036                    RTS PC
                ;
                ;
                ;
```

```
                        .SBTTL INSERT MOVE MACROS
                        .CSECT SETMVS
                ;INSMVE
                ;R3=DEPTH
                ;R0=P0, (R0)=SQ0
                ;R1=SQ1
                ;R2=
                ;IF ALLM(D) THEN DO;
                ;       IF REL(SQ1) THEN PRESS(D,REL(SQ1))=&PTYP(P0);
                ;       IF^REPEAT THEN DO;
                ;               M=M-1;
                ;               MOVE(M).TYPE=MT;
                ;               MOVE(M).TO=SQ1;
                ;               MOVE(M).FROM=SQ0;
                ;               END;
                ;       END;
                ;
                .MACRO INSMVE $P0,$SQ1,TYPE,L2,?L1
                        TST  REL($SQ1)  ;REL SQ?
                        BEQ  L1
                                MOV  UPMOVE,R2
                                ADD  REL($SQ1),R2       ;R2=->PRESS(D,SQ1)
                                BIS   PTYP($P0),-(R2)
                                BIS  <PTYP+2>($P0),-(R2)
                L1:
                        TST  REPEAT
                        BNE  L2
                                MOV  TYPE,-(R4)
                                MOV  $SQ1,-(R4)
                                MOV  ($P0),-(R4)
                                BR   L2
                .ENDM
                ;
                ;
                ;
                ;
                ;INSCMVE
                ;IF ALLM(D) THEN DO;
                ;       IF REL(SQ1) THEN PRESS(D,REL(SQ1))=&PTYP(P0);
                ;       IF REL(P1) THEN PRESS(D,REL(P1))=&PTYP(P0);
                ;       END;
                ;IF REPEAT THEN
                ;       IF P1=OPPKING THEN CHK(D-1)=1;
                ;ELSE DO;
                ;       M=M-1;
                ;       MOVE(M).FROM=SQ0;
                ;       MOVE(M),TO=SQ1;
                ;       MOVE(M).TYP=MT;
                ;       IF P1=OPPKING THEN DO;
                ;               GOTO RESET;
                ;               END;
                ;       END;
                ;
                ;
                ;R3=D
                ;R2=P1
                ;R1=SQ1
```

```
                ;R0=P0
                ;
                .MACRO TCMVE $P0,$SQ1,TYPE,$P1,L2,?L1
                        TST REL($SQ1)
                        BEQ  L1
                             MOV  R3,-(SP)
                             MOV  UPMOVE,R3
                             ADD REL($SQ1),R3
                             BIS  PTYP($P0),-(R3)
                             BIS  <PTYP+2>($P0),-(R3)
                             MOV  (SP)+,R3
                L1:     TST REL($P1)
                        BEQ  L2
                             MOV  R3,-(SP)
                             MOV  UPMOVE,R3
                             ADD  REL($P1),R3
                             BIS  PTYP($P0),-(R3)
                             BIS  <PTYP+2>($P0),-(R3)
                             MOV  (SP)+,R3
                .ENDM
                ;
                ;
                ;
                .MACRO INSCMVE $P0,$SQ1,TYPE,$P1,L4,?L2,?L3
                TCMVE $P0,$SQ1,TYPE,$P1,L2
                L2:
                CMP  CLR($P0),CLR($P1)
                BEQ  L4
                TST REPEAT
                BEQ  L3
                     CMP  $P1,OPPKING
                     BNE  L4
                          MOV  R3,-(SP)
                          MOV  UPMOVE,R3
                          INCB CHK(R3)
                          MOV  (SP)+,R3
                          BR   L4
                L3:     MOV  ($P0),-(R5)
                        MOV  $SQ1,-(R5)
                        MOV  TYPE,-(R5)
                        CMP  $P1,OPPKING        ;CAPTURE OPPONENTS KING ?
                        BNE  L4
                             MOV  $P1,KCAP      ;KCAP CONTAINS THE KING
                                                ;WHICH IS CAPTURED
                             JMP  RESET
                .ENDM
                ;
                ;
                ;
```

```
        .SBTTL SETMOVES
SETMVS:
;R5 IS USED AS CMVE PTR FOR INSMVE
;R4 GIVES WHERE MOVES MAY BE PLACED, AND ON RETURN PTS TO TOP OF MSTACK
;R3=->UPM(D) INITIALLY ,IS STORED IN UPM & IS RETURNNED UNCHANGED
;
.MACRO ROOKSQS A,B,?L1,?L2,?L3,?L4
;GIVEN A=TO OF KING, RETURN A PTR TO FROM & TO SQUARES OF CASTLING ROOK
        CMP   A,#SQUARE+<2*LENSQ>
        BNE   L1
              MOV   #RCAST,B
              BR    L4
L1:     CMP   A,#SQUARE+<6*LENSQ>
        BNE   L2
              MOV   #RCAST+4,B
              BR    L4
L2:     CMP   A,#SQUARE+<58.*LENSQ>
        BNE   L3
              MOV   #RCAST+8.,B
              BR    L4
L3:           MOV   #RCAST+12.,B
L4:
.ENDM
;
;
;
        TST COLOUR      ;IF COLOUR =BLACK THEN DO;
        BGT  1$
              MOV BKING,R0     ;SET R0=P0
              MOV  R0,CLRKING
              MOV  WKING,OPPKING
              BR   2$
        1$:
              MOV WKING,R0     ;ELSE DO;
              MOV  R0,CLRKING
              MOV  BKING,OPPKING
2$:
        TST REPEAT      ;IF ^REPEAT THEN DO;
        BNE  5$
              CLR  -(R4)       ;END MARKER
              TST  NIM(R3)     ;IF ^ALLM(D) THEN
              BGE  3$
                   MOV  #DUMMY,-(R4)       ;FROM(M)=DUMMY
        3$:
              CMP  MT(R3),#THRUP
              BLT  5$ ;CHECK LABEL
                   CMP  MT(R3),#UPAWN
                   BNE  4$
                        ;SET EPSQ=MID SQUARE OF LAST MOVE
                        MOV  MT(R3),SPEC        ;SPEC MOVE
                        MOV SQ1(R3),R2  ;TO SQ OF LAST MOVE
                        ADD  SQ0(R3),R2
                        ROR  R2 ;WANT NO NEGATIVE BIT INSERTED
                        MOV  R2,EPSQ    ;SET EN PASSANT SQUARE
                        BR   5$
              4$:       CMP  MT(R3),#CASTLE
                        BNE 5$
```

```
                                        ;PLACE KING ON SQUARES HE PASSED OVER
                                        ;IN CASTLING - IF ANY OF THE SQUARES ARE
                                        ;IN CHECK, KCAP WILL BE FLAGGED
                                        MOV  MT(R3),SPEC        ;SPEC MOVE
                                        ROOKSQS SQ1(R3),R2
                                        MOV  @2(R2),R2  ;R2=P2
                                        MOV  R2,-(SP)   ;STACK THE ROOK POINTER
                                        MOV  CLRKING,@(R2)      ;GHOST KINGS
                                        MOV  CLRKING,@SQ0(R3)
5$:
        MOV  R3,UPMOVE  ;SAVE UPDATED MOVE PTR
        MOV  #CMVE,R5   ;FREE R5, R5=CMOVE AREA. N.B.MAKE SURE 0 AT CMOVE
        BR   S7$
;NPIECE
NPIECE:
;R0=PIECE(P0)
        MOV NXTP(R0),R0
        BEQ  S9$
S7$:    MOV  TYP(R0),R1
        MOV  R1,MTYPE
        JMP  @8$(R1)
8$:     .WORD 0,0,S30$,S10$,S10$,S10$,S30$,S40$,S10$,0,S20$,S40$
S9$:    JMP  RESET
        ;DUMMY,NM,BM,RM,QM,KM,PM,URM,UKM,UPM
;N.B. R3 SHOULD INITIALLY PT ABOVE THE 1ST. DIRECTION
```

```
                        .SBTTL SETMOVES: ROOKS, BISHOPS & QUEENS.
                ;ROOKS, BISHOPS, AND QUEENS
                ;URM:RM:QM:BM:
                ;R0=PIECE(P0)
 000342         S10$:
 000342                 MOV DADD(R0),R3 ;R3=->DIRS
                ;NRDIR:
 000346         11$:
 000346                 TST -(R3)
0 00350                 BEQ NPIECE
1 00352                 MOV (R0),R1     ;R1=SQ0
2               ;SAMRDIR:
3 00354         12$:
4 00354                 ADD (R3),R1     ;(R3)=->ADDR OF NEXTSQ
5 00356                 MOV (R1),R1     ;R1=SQ1
6 00360                 BEQ 11$
7 00362                         MOV (R1),R2     ;R2=PIECE(P1)
8 00364                         BNE 13$ ;IF EMPTY
9 00366                                 INSMVE R0,R1,MTYPE,12$
0 00434                         13$:
1 00434                                 INSCMVE R0,R1,MTYPE,R2,11$
2               ;
```

```
                .SBTTL SETMOVES:CASTLING
        ;UNMOVED KING
        ;CASTLING
        ;UKM:
        S20$:
                TST REPEAT          ;IF ^REPEAT THEN DO;
                BNE S30$
                        TST COLOUR          ;IF COLOUR=BLACK THEN R3=->BROOK
                        BGT 21$
                                MOV #BROOK,R3
                                BR  22$
                        21$:    MOV #WROOK,R3 ;else  R3 -> WRook
        ;NEXTR:
                22$:
                        MOV -(R3),R1        ;R1=SQ2
                        BEQ S30$            ;IF SQ2^=0 THEN DO;
                                TST -(R3)           ;R3=->DIR
                                MOV (R1),R2         ;R2=P2
                                BEQ 22$ ;IF P2^=0 THEN DO;
                                CMP TYP(R2),#UROOK          ;IF PIECE(P2).TYP
                                                            ;=UROOK THEN
                                BNE 22$
                                        CLR -(SP)           ;0 END MARKER
        ;NEXTSQ:
                                23$:    ADD (R3),R1
                                        MOV (R1),R1         ;R1=(SQ4),SQ1,SQ3
                                        CMP R1,(R0)
                                        BNE 24$
                                                ;SQ=SQ0 & CASTLING VALID
                                                TST (SP)+
                                                MOV (SP)+,R1    ;R1=SQ1
                                                INSMVE R0,R1,#CASTLE,26$
                                24$:
                                        MOV R1,-(SP)    ;STACK<=(SQ4),SQ1,SQ3
                                        TST (R1)        ;PIECE ON SQ1 ?
                                        BEQ 23$
                                26$:
                                        TST (SP)+
                                        BNE 26$
                                BR 22$
        ;
```

```
                        .SBTTL SETMOVES: KING AND NIGHT MOVES
                ;KING & NIGHT MOVES
                ;
                ;R0=P0
000760          S30$:
000760                  MOV DADD(R0),R3 ;FIND DIRS
000764          31$:
                ;NEXT N DIR
000764                  MOV -(R3),R1    ;R1=NEXTDIR
000766                  BEQ 33$
000770                  ADD (R0),R1
000772                  MOV (R1),R1     ;R1=SQ1
000774                  BEQ 31$
000776                      MOV (R1),R2     ;PIECE(P1)
001000                      BNE 32$
001002                              INSMVE R0,R1,MTYPE,31$
001050                              BR  31$ ;NNDIR
001052                      32$:
001052                              INSCMVE R0,R1,MTYPE,R2,31$
001226          33$:    JMP  NP1ECE
                ;
```

```
                        .SBTTL SETMOVES: PAWN MOVES
                ;PAWNS & UNMOVED PAWNS
                ;PM:UPM
001232          S40$:
                        ;FORWARD MOVES N OR S
001232                  MOV DADD(R0),R3 ;R3=->DIRS
001236                  MOV (R0),R1
001240                  ADD -(R3),R1
001242                  MOV (R1),R1     ;R1=SQ1, N OR S OF SQ0
                ;
01244                   MOV R1,R2
01246                   ADD (R3),R2
01250                   MOV (R2),R2     ;R2=SQ2
01252                   BNE 41$
                                ;IF SQ2=0 THEN IF MT=PAWN THEN MT=QPAWN
01254                           CMP MTYPE,#PAWN
01262                           BNE 41$
01264                           MOV #QPAWN,MTYPE
                                ;GENERATE QUEENING PAWN MOVES EVEN WHEN ALLM(D)=0
                                ;BY SETTING ALLM(D)=1
                                ;R2=0
                                ;N.B. IF PAWN CAPTURES KING, IMMEDIATE EXIT
                                ;IS TAKEN TO RESET AND ALLM(D) IS WRONG.
                                ;CLR R2 ;IS UNECESSARY
01272           41$:
01272                   TST (R1)        ;PIECE(P1) ?
01274                   BNE S42$
01276                           INSMVE R0,R1,MTYPE,42$  ;R2 LOST
01344                   42$:
01344                           CMP MTYPE,#UPAWN
01352                           BNE S42$
01354                                   ADD  (R3),R1    ;FIND SECOND SQ AGAIN
01356                                   MOV  (R1),R1    ;R1=SQ2
01360                                   TST (R1)        ;IF NO PIECE ON SQ2
01362                                   BNE S42$
01364                                           INSMVE R0,R1,MTYPE,S42$
01432           S42$:
                        ;DIAGONALS
01432                   MOV -(R3),R1
01434                   BEQ S44$
01436                   ADD (R0),R1
01440                   MOV (R1),R1     ;R1=SQ1
01442                   BEQ S42$        ;NPDIR
01444                           MOV (R1),R2     ;R2=P1
01446                           BEQ 43$
01450                                           INSCMVE R0,R1,MTYPE,R2,S42$
01624                           43$:
01624                                   CMP R1,EPSQ
01630                                   BNE S42$
                                                ;R1=SQ1=EPSQ
01632                                           MOV UPMOVE,R1
01636                                           MOV SQ1(R1),R1
01642                                           MOV  (R1),R2    ;R2=P1
01644                                           INSCMVE R0,R1,#EPCAP,R2,S42$
02020           S44$:
02020                   CMP MTYPE,#THR
02026                   BLT 46$
```

```
                        CMP MTYPE,#QPAWN
                        BNE 45$
                                MOV #NPAWN,MTYPE
                                JMP S40$
                        45$:
                                ;MT=NPAWN
        46$:    JMP NPIECE
        ;
        ;RESET BOARD
        ;
        RESET:
                MOV UPMOVE,R3
                TST SPEC
                BEQ 52$
                        CMP SPEC,#CASTLE
                        BNE 51$
                                ;RESET BOARD: REMOVE KING AND REPLACE ROOK
                                CLR @SQ0(R3)
                                MOV  (SP)+,R2   ;R2=P2, THE ROOK
                                MOV  R2,@(R2)   ;SQUARE(SQ3).PIECE=P2
                                CLR SPEC
                                BR 52$
                        51$:
                                CLR EPSQ        ;PREVENT OTHER CALLS FROM
                                CLR SPEC        ; USING THIS EPSQ.
        ;PUT CMOVES ON MOVE STACK
        ;R5=CMOVES, R4=MOVES
        52$:
        53$:
                MOV (R5)+,-(R4)
                BEQ  55$
        54$:    MOV  (R5)+,-(R4)
                MOV  (R5)+,-(R4)
                MOV  (R5)+,-(R4)
                BNE  54$
        55$:
                TST  (R4)+      ;REMOVE 0 FROM TOP OF STACK
                INC  NOSETMVS
                BNE  56$
                        INC  NOSETMVS+2
        56$:
                RTS  PC
        ;
        ;
        ;
```

```
            .SBTTL CSTMVS:CAPTURING MOVES ONLY
            .CSECT CSTMVS
            .MACRO CINSMV $P0,$SQ1,TYPE,$P1,L1
            MOV  TYPE,-(R4)
            MOV  $SQ1,-(R4)
            MOV  ($P0),-(R4)
            CMP  $P1,OPPKING        ;CAPTURE OPPONENTS KING?
            BNE  L1
                MOV  $P1,KCAP   ;KCAP CONTAINS THE KING WHICH IS CAPTURED
                JMP  C50$
            .ENDM
        ;
        ;
        ;
        CSTMVS:
            CLR  -(R4)      ;END MARKER
            TST COLOUR      ;IF COLOUR =BLACK THEN DO;
            BGT  1$
                MOV BKING,R0    ;SET R0=P0
                MOV  WKING,OPPKING
                BR   2$
            1$:
                MOV WKING,R0    ;ELSE DO;
                MOV  BKING,OPPKING
        2$:
        4$:
                        CMP  MT(R3),$UPAWN
                        BNE  C7$
                            ;SET EPSQ=MID SQUARE OF LAST MOVE
                            MOV  MT(R3),SPEC        ;SPEC MOVE
                            MOV SQ1(R3),R2  ;TO SQ OF LAST MOVE
                            ADD  SQ0(R3),R2
                            ROR  R2 ;WANT NO NEGATIVE BIT INSERTED
                            MOV  R2,EPSQ    ;SET EN PASSANT SQUARE
                            BR   C7$
        ;ROOKS, BISHOPS, AND QUEENS
        ;URM:RM:QM:BM:
        ;R0=PIECE(P0)
        C10$:
        ;NRDIR:
        11$:
            TST -(R5)
            BEQ C5$
            MOV (R0),R1     ;R1=SQ0
        ;SAMRDIR:
        12$:
            ADD (R5),R1     ;(R5)=->ADDR OF NEXTSQ
            MOV (R1),R1     ;R1=SQ1
            BEQ 11$
                MOV (R1),R2     ;R2=PIECE(P1)
                BEQ 12$ ;IF EMPTY
                        CMP CLR(R0),CLR(R2)
                        BEQ 11$
                        CINSMV R0,R1,MTYPE,R2,11$
        ;
        ;
        ;
```

```
                ;KING & NIGHT MOVES
                ;
                ;R0=P0
00152           C20$:;UNMOVED KING DOES NOT CASTLE
00152           C30$:
00152           31$:
                ;NEXT N DIR
00152                   MOV -(R5),R1    ;R1=NEXTDIR
00154                   BEQ C5$
00156                   ADD (R0),R1
00160                   MOV (R1),R1     ;R1=SQ1
00162                   BEQ 31$
00164                           MOV (R1),R2     ;PIECE(P1)
00166                           BEQ 31$
00170                                   CMP CLR(R0),CLR(R2)
00176                                   BEQ 31$
00200                                   CINSMV R0,R1,MTYPE,R2,31$
                ;
```

```
                        .SBTTL CSTMVS: PAWN MOVES
                ;NPIECE
000226          C5$:
                ;R0=PIECE(P0)
000226                  MOV NXTP(R0),R0
000232                  BEQ  C50$
000234          C7$:    MOV  TYP(R0),R1
000240                  MOV  DADD(R0),R5        ;GET POINTER TO PIECES' DIRECTIONS
000244                  MOV  R1,MTYPE
00250                   JMP  @8$(R1)
00254           8$:     .WORD 0,0,C30$,C10$,C10$,C10$,C30$,40$,C10$,0,C20$,39$
                        ;DUMMY,NM,BM,RM,QM,KM,PM,URM,UKM,UPM
                ;N.B. R3 SHOULD INITIALLY PT ABOVE THE 1ST. DIRECTION
                ;
                ;
                ;UNMOVED PAWNS
00304           39$:
00304                   TST  -(R5)      ;MOVE PAST N OR S DIRECTIONS
00306                   BR   42$
                        ;PAWNS : SEE IF THEY QUEEN
00310           40$:
                        ;FORWARD MOVES N OR S
00310                   MOV (R0),R1
00312                   ADD -(R5),R1
00314                   MOV (R1),R1     ;R1=SQ1, N OR S OF SQ0
                ;
00316                   MOV R1,R2
00320                   ADD (R5),R2
00322                   MOV (R2),R2     ;R2=SQ2
00324                   BNE 42$ ;WANT NO MOVES FORWARD WHICH DO NOT QUEEN
                                ;IF SQ2=0 THEN IF MT=PAWN THEN MT=QPAWN
00326                           CMP MTYPE,#PAWN
00334                           BNE 41$
00336                           MOV #QPAWN,MTYPE
                                ;GENERATE QUEENING PAWN MOVES
00344           41$:
00344                   TST (R1)        ;PIECE(P1) ?
00346                   BNE 42$
00350                           CINSMV R0,R1,MTYPE,R0,42$       ;NOT A CAPTURE BUT
00376           42$:                                            ;CODE O.K.
                        ;DIAGONALS
00376                   MOV -(R5),R1
00400                   BEQ 44$
00402                   ADD (R0),R1
00404                   MOV (R1),R1     ;R1=SQ1
00406                   BEQ 42$ ;NPDIR
00410                           MOV (R1),R2     ;R2=P1
00412                           BEQ 43$
00414                                   CMP  CLR(R0),CLR(R2)
00422                                   BEQ 42$
00424                                           CINSMV R0,R1,MTYPE,R2,42$
00452                           43$:
00452                                   CMP R1,EPSQ
00456                                   BNE 42$
                                                ;R1=SQ1=EPSQ
00460                                           MOV SQ1(R3),R1
00464                                           MOV  (R3),R2    ;R2=P1
```

```
5F  0466                                                CINSMV R0,R1,#EPCAP,R2,42$
59  0514        44$:
60  0514                CMP MTYPE,#THR
61  0522                BLT C5$
62  0524                    CMP MTYPE,#QPAWN
63  0532                    BNE 45$
64  0534                        MOV #NPAWN,MTYPE
65  0542                        MOV  DADD(R0),R5
66  0546                        BR   40$
67  0550                    45$:
68                              ;MT=NPAWN
69  0550                        BR   C5$
70              ;
71              ;RESET BOARD
72              ;
73  0552        C50$:
74  0552                CLR  EPSQ
75  0556                INC  NOCSTMVS
76  0562                BNE  C51$
77  0564                    INC  NOCSTMVS+2
78  0570        C51$:
79  0570                RTS  PC
80              ;
81              ;
82              ;
```

```
        .SBTTL UPDATE BOARD
        .CSECT UPDATE
UPDATE:
;R3=->LASTUP AND IS RETURNED AS THE NEW ->UPM
;R4=->FROM AND IS RETURNED AS THE TOP OF THE MOVE STACK
        MOV  R4,R5
        MOV  R3,-(R5)    ;MSTACK<=->LUM
        MOV  R5,-(SP)    ;STACK<=->UPM
;R3=->LUM,R4=->FROM, R5=->UPM
        MOV  (R3),R2     ;VSCOR(D+1)=VSCOR(D-1)
        MOV  VSCOR(R2),-(R5)     ;MSTACK<=VSCOR(D+1)
        MOV  NIM(R3),-(R5)       ;NIM(D+1)=NIM(D)
1$:;RESTART
        MOV  @(R4)+,R0   ;R0=P0
        MOV  (R4)+,R1    ;R1=SQ1
        ;R4=->MOVE(M).TYPE
        CLR  @(R0)       ;SQUARE(SQ0).PCE=0
        MOV  R1,(R0)     ;PIECE(P0).ON=SQ1
        MOV  (R1),-(R5) ;MSTACK<=CPIECE
        BEQ  3$
            ;PIECE IS CAPTURED
            MOV  (R1),R1     ;R1=P1
            MOV  SCOR(R3),-(R5)      ;SCOR(D+1)=SCOR(D)-PIECE(P1).VAL
            SUB  VAL(R1),(R5)
            MOV  LSTP(R1),R2         ;R2 ~=0 SINCE THERE IS A KING
            MOV  NXTP(R1),R3
            BEQ  2$
                MOV  R2,LSTP(R3)
        2$:     MOV  R3,NXTP(R2)
;               TST  REL(R1)
;               BEQ  21$
;                   MOV  LSTR(R1),R2
;                   MOV  NXTR(R1),R3
;                   BEQ  20$
;                       MOV  R2,LSTR(R3)
;               20$:
;                   TST  R2
;                   BEQ  21$
;                       MOV  R3,NXTR(R2)
;       21$:
            MOV  (R1),R1     ;R1=SQ1
            MOV  @(SP),R3    ;RESTOR R3=->LUM
            BR   U4$
        3$:
            ;NO PIECE CAPTURED
            MOV  SCOR(R3),-(R5)      ;SCOR(D+1)=SCOR(D)
U4$:
        MOV  R0,(R1)     ;SQUARE(SQ1).PCE=P0
        CMP  (R4),#THRU
        BLT U14$
            CMP  (R4),#THR
            BGT  5$
                SUB  #DT,TYP(R0)
                BR   U14$
        5$:     CMP  (R4),#CASTLE
                BNE  6$
                    SUB  #DT,TYP(R0)
```

```
                        ROOKSQS R1,R2
                        MOV   @(R2)+,R0   ;R0=P0
                        MOV   (R2)+,R1    ;R1=SQ1
                        CLR   @(R0)       ;SQUARE(SQ0).PIECE=0
                        MOV   R0,(R1)     ;SQUARE(SQ1).PIECE=P0
                        MOV   R1,(R0)     ;PIECE(P0).ON=SQ1
                        SUB   #DT,TYP(R0)
                        BR    U14$
        6$:
                CMP   (R4),#EPCAP
                        BNE   8$
                                CLR   (R1)        ;SQUARE(SQ1).PIECE=0
                                TST   COLOUR
                                BGT   7$
                                        ADD   #S,R1       ;BLACK CAPTURES S
                                        BR    17$
                                7$:     ADD   #N,R1
                        17$:
                                MOV   (R1),R1     ;R1=SQ2
                                MOV   R0,(R1)     ;SQUARE(SQ2).PIECE=P0
                                MOV   R1,(R0)     ;PIECE(P0).SQUARE=SQ2
                                BR    U14$
        8$:
                CMP   (R4),#QPAWN
                BNE   11$
                        SUB   VAL(R0),(R5)        ;SCOR(D+1)=- VAL(P0)
                        MOV   #QUEEN,TYP(R0)
                        TST COLOUR
                        BGT   9$
                                MOV   VALUES+QUEEN,VAL(R0)
                                NEG   VAL(R0)
                                BR    10$
                        9$:     MOV   VALUES+QUEEN,VAL(R0)
                10$:    MOV   #QDIRS,DADD(R0)
                        ADD   VAL(R0),(R5)        ;SCOR(D+1)=+ VAL(P0)
                        BR    U14$
        11$:
                CMP   (R4),#NPAWN
                BNE   U14$
                        SUB   VAL(R0),(R5)        ;SCOR(D+1)=- VAL(P0)
                        MOV   #NIGHT,TYP(R0)
                        TST COLOUR
                        BGT   12$
                                MOV   VALUES+NIGHT,VAL(R0)
                                NEG   VAL(R0)
                                BR    13$
                        12$:    MOV   VALUES+NIGHT,VAL(R0)
                13$:    MOV   #NDIRS,DADD(R0)
                        ADD   VAL(R0),(R5)        ;SCOR(D+1)=+ VAL(P0)
U14$:
        CLR   -(R5)             ;CHK=0 & NOIN(D)=0
        CLR   -(R5)             ;BSTMVS(D,*)=0 SAVES TESTING TERMINAL NODES
                                ;FOR #DUMMY
        MOV   (SP)+,R3          ;R3=->UPM
        MOV   R3,R4             ;RETURN R4=->TOP OF MSTACK
        SUB   #LMV,R4
        INC   D   ;D=D+1
```

UPDATE BOARD

```
.16 0472                DINC NOUPS,NOUPS+2
.17 0504                NEG  COLOUR
.17 0510                RTS  PC
.18             ;
.19             ;
.20             ;
```

```
                .SBTTL RESTORE BOARD
                .CSECT RESTOR
        RESTOR:
        ;R3=->UPM
        ;R4 IS UNDEFINED, AND IS RETURNED AS ->FROM OF NEXTMOVE
        ;RETURNS R0=VSCOR(OLD D).
                MOV   R3,R4
                TST   (R4)+         ;R4-->FROM
                DEC   D   ;D=D-1
                NEG   COLOUR
                MOV   (R4)+,R0      ;R0=SQ0
                MOV   @(R4)+,R1     ;R1=P0
                CMP   (R4)+,#THRU
                BLT   R10$
                        CMP   -2(R4),#THR
                        BGT   3$
                                ADD   #DT,TYP(R1)
                                BR    R10$
                3$:             CMP   -2(R4),#CASTLE
                                BNE   4$
                                ;MOVE THE KING & THEN SET R0,R1 SO THAT THE
                                ;BE SET BACK
                                ADD   #DT,TYP(R1)
                                ROOKSQ (R1),R2  ;FIND PTR TO ROOK SQUARES
                                MOV   R1,(R0)       ;SQUARE(SQ0).PIECE=P0
                                CLR   @(R1)         ;SQUARE(SQ1).PIECE=0
                                MOV   R0,(R1)       ;PIECE(P0).ON=SQ0
                                MOV  (R2)+,R0       ;R0=SQ2
                                MOV   @(R2)+,R1     ;R1=P2
                                ADD   #DT,TYP(R1)
                                BR    R10$
                4$:
                        CMP   -2(R4),#EPCAP
                        BNE   7$
                                ;MOVE THE CAPTURING PIECE BACK TO THE SQUARE
                                ; THE CAPTURED PIECE SHOULD BE LEFT.
                                MOV   -4(R4),R2   ;R2=SQ1
                                MOV   R2,R1         ;WANT R1=SQ2
                                TST   COLOUR
                                BGT   5$
                                        ADD   #S,R1
                                        BR    6$
                                5$:     ADD   #N,R1
                        6$:
                                MOV   @(R1),R1    ;R1=P0
                                ;NOW MOVE THE CAPTURING PIECE BACK TO SQ1
                                CLR   @(R1)         ;SQUARE(SQ2).PIECE=0
                                MOV   R1,(R2)       ;SQUARE(SQ1).PIECE=P0
                                MOV   R2,(R1)       ;PIECE(P0).ON=SQ1
                                ;R0=SQ0, R1=P0
                                BR    R10$
                7$:
                                ;QPAWN AND NPAWN
                                MOV   #PAWN,TYP(R1)       ;PIECE(P0).TYP=PAWN
                                MOV   VALUES+PAWN,VAL(R1) ;PIECE(P0).VAL=VALUE(WP
                                TST   COLOUR      ;                              N)
                                BGT   8$
```

SYMBOL TABLE

```
A  IVR  000516R  011   ACDMOL  001032R  011   ACKCA = 000002
A  VR   000424R  011   ADDOQ   000516R  022   ADIRS   037662R  002
ADT     044322R  002   ALPHA   044610R  002   ATTENT  000124R  022
BASE    000040R  022   BDIRS   040002R  002   BETA    044612R  002
BIN     002164R  004   BISHOP= 000006        BKING   000500R  024
BL    = 000040        BMV   = 177766        BOARD   000126R  006
BOTM    000000R  024   BPDIRS  040046R  002   BROOK   000142R  024
BSQCOL  000000R  006   BSTMVS  000000R  027   BUSY    000100R  022
CA      000112R  022   CALPHA  044015R  002   CASTLE= 000036
CBETA   044023R  002   CCDEPT  044124R  002   CD      000012R  024
CDO     000014R  024   CHARAC  000334R  022   CHAREP  000272R  007
CHARMV  000000R  007   CHARSQ  000444R  004   CHK   = 177765
CHKSUM  000110R  021   CKCAST  000260R  007   CLR   = 000010
CLRD  = 000000        CLRKIN  000154R  024   CLSND   000066R  020
CLSNDA  000122R  020   CM      000114R  022   CMAXD   044030R  002
CMAXNI  044041R  002   CMMAXD  044116R  002   CMVE    000472R  024
CNPAWN  000302R  007   CNVMIC  001340R  011   CNVMIL  001256R  011
COLOUR  044342R  002   CPCE  = 177772        CPEN    044145R  002
CQCAST  000264R  007   CQPAWN  000277R  007   CR    = 000015
CSQ   = 000010        CSTMVS  000000R  031   CUTOFF= 000006
CVARCD  044133R  002   C10$    000074R  031   C20$    000152R  031
C30$    000152R  031   C5$     000226R  031   C50$    000552R  031
C51$    000570R  031   C7$     000234R  031   D       000010R  024
DADD  = 000026        DADDNO  001116R  011   DDEC    001422R  011
DDIVR   000602R  011   DDIVS   002120R  011   DEC     001652R  011
DESCEN  000376R  025   DIRS    037600R  002   DMOLS   002006R  011
D  NO   001154R  011   DT    = 000010        DTT     044316R  002
DUMMY = 000002        D1      000172R  012   D10     000272R  012
D11     000277R  012   D12     000304R  012   D13     000311R  012
D14     000317R  012   D15     000325R  012   D16     000333R  012
D17     000335R  012   D18     000344R  012   D19     000354R  012
D2      000211R  012   D20     000375R  012   D21     000435R  012
D22     000451R  012   D23     000467R  012   D24     000512R  012
D25     000526R  012   D27     000564R  012   D28     000633R  012
D29     000707R  012   D3      000223R  012   D30     000745R  012
D31     000775R  012   D32     001053R  012   D33     001127R  012
D34     001162R  012   D35     001166R  012   D36     001171R  012
D37     001175R  012   D38     001202R  012   D39     001205R  012
D4      000227R  012   D41     001210R  012   D5      000235R  012
D6      000241R  012   D7      000245R  012   D8      000252R  012
D9      000255R  012   E     = 000032        ENDCO   000562R  024
ENDP    032400R  002   ENDSQ   037600R  002   ENE   = 000044
EPCAP = 000040        EPSQ    000504R  024   ESE   = 000026
EVAL    000612R  026   EXIT  = 000011        EXPID   001022R  013
FIN     000002R  013   FIRREL  044622R  002   FOUT    000010R  013
FPCH    000126R  022   FR      030000R  002   FREAID  000426R  015
FREECA  000716R  022   FREEOB  000556R  015   FRONT   031000R  002
GET     002060R  004   HCA     000110R  022   HEAD  = 000125
IUNDRE  044644R  002   IMVSTA  002136R  025   INDEX   002034R  004
INPOS   044344R  002   INT   = 000140        INTCON  000354R  015
INTPT   000000R  015   INTPTC  000000R  023   INTPTP  000474R  004
INTP20  000324R  023   INTP25  000376R  023   INTP30  000442R  023
INTP50  000734R  023   INTP59  001074R  023   INTSW = 000340
IPTYPC  000000R  004   ISQUAR  000104R  004   ISR     000000R  014
ISR0$   000004R  014   KCAP    000006R  024   KCAPT   001156R  026
KING  = 000014        LALPHA  000400R  003   LAND1N  000106R  022
LASTD   000122R  024   LBETA   000414R  003   LBLACK  000242R  003
```

SYMBOL TABLE

```
L[illegible]  = 000314          LBOARD  000470R  003   LCD     000550R  003
L[illegible]AR  000316R  003    LEAD    000120R  022   LEND    000702R  003
LENP  = 000030                  LENSQ = 000052        LF    = 000012
LGO     000430R  003    LINPOS  000154R  003   LMAXCD  000644R  003
LMAXDP  000354R  003    LMAXNI  000366R  003   LMMAXD  000534R  003
LMOVEN  000600R  003    LMV   = 000326        LOOP    000060R  003
LPDPTH  000506R  003    LPEN    000630R  003   LPIECE  000300R  003
LPLAY   000522R  003    LR      001030R  013   LREL    000252R  003
LREST   000724R  003    LREST1  001146R  003   LREST2  001676R  003
LS      001032R  013    LSETBD  000264R  003   LSQUAR  000310R  003
LSTP  = 000024          LSTR  = 000004        LSTRGS  044172R  002
LSWEEP  000660R  003    LUM   = 000000        LVARCD  000564R  003
LWHITE  000232R  003    MAKEBD  003660R  005   MAXCD   000102R  022
MAXDPT  044576R  002    MAXID = 000177        MAXNIM  044600R  002
MAXPCH  000332R  022    MAXR    000414RG 013  MAXS    001014RG 013
MINMAX  000000R  025    MINMX1  000044R  025   MINMX2  000000R  026
MINPCH  000132R  022    MINR    000014RG 013  MINS    000414RG 013
MLOOP   001024RG 025    MMAXD   044604R  002   MOVENO  044606R  002
MSGIN   001016R  013    MSGOUT  001020R  013   MSTACK  030000RG 002
MSTART  000402R  025    MT    = 000006        MTYPE   000474R  024
M0$     000166R  025    M1      040063R  002   M10     040250R  002
M11     040274R  002    M12     040276R  002   M130$   001442R  025
M140$   001470R  025    M145$   001502R  025   M15     040314R  002
M150$   001606R  025    M16     040335R  002   M17     040361R  002
M18     040366R  002    M19     040370R  002   M2      040070R  002
M20     040411R  002    M21     040432R  002   M22     040442R  002
M[illegible]    040452R  002    M230$   000270R  026   M24     040476R  002
M[illegible]    040515R  002    M26     040536R  002   M27     040606R  002
M28     040626R  002    M29     040654R  002   M3      040112R  002
M30     040704R  002    M31     040721R  002   M32     040747R  002
M33     040767R  002    M35     041013R  002   M36     041044R  002
M37     041064R  002    M38     041100R  002   M4      040143R  002
M40     041105R  002    M41     041122R  002   M42     041136R  002
M43     041152R  002    M44     041166R  002   M45     041203R  002
M46     041214R  002    M47     041241R  002   M48     041276R  002
M49     041332R  002    M5      000305R  007   M50     041342R  002
M51     041353R  002    M52     041365R  002   M53     041400R  002
M54     041410R  002    M55     041417R  002   M55$    000766R  025
M56     041465R  002    M57     041506R  002   M58     041516R  002
M59     041526R  002    M6      040162R  002   M60     041536R  002
M61     041546R  002    M62     041556R  002   M63     041564R  002
M64     041574R  002    M65     041604R  002   M66     041614R  002
M67     041616R  002    M68     041626R  002   M69     041634R  002
M7      040220R  002    M70     041644R  002   M71     041655R  002
M72     041670R  002    M73     041701R  002   M74     041714R  002
M75     041725R  002    M8      040230R  002   M9      040240R  002
N     = 000040          NADT    044326R  002   NBP     000552R  024
NDIRS   037770R  002    NE    = 000042        NEWLIN  000172R  010
NEWSC   002214RG 025    NEWSC0= 000002        NEWSC1= 000004
NEXTSQ= 000012          NIGHT = 000004        NIM   = 177774
NLHEAD  000346R  011    NLTB    040056R  002   NLTBHD  000332R  011
N[illegible]  = 000050          NNW   = 000046        NOCSTM  000536R  024
N[illegible]  = 177766          NOINT = 000000        NOMAXD  000542R  024
NOMINM  000566R  024    NOMMAX  000546R  024   NORCUT  000562R  024
NORINT  001040R  013    NOSETM  000532R  024   NOUPS   000526R  024
NOUTB = 000012          NPAWN = 000044        NPIECE  000264R  030
NTAB    000176R  010    NW    = 000036        NXTP  = 000022
```

SYMBOL TABLE

```
N[illegible]R  = 000002          NXTSID  001024R  013  NXTSQ  = 000012
O[illegible]D    001014R  013    ON     = 000000       OPNSE    000000R  020
OPNSEA  000044R  020    OPPKIN  000156R  024  OSC      000116R  022
OUTBUF  001026R  013    PAR     000000R  022  PAWN   = 000016
PBMVES  003506R  005    PBUF    000022R  024  PB1$     003544R  005
PC     =%000007         PCE    = 000000       PCHARS=  000007
PCNT   = 000041         PCOUNT  002776R  005  PCO20$   003150R  005
PCO55$  003436R  005    PDPTH   000124R  024  PDUMP    000000RG 012
PDUMP0  000112R  012    PDUMP1  000014R  012  PD59$    000110R  012
PEN     037742R  002    PERCEN  001342R  005  PIECE    031000R  002
PIN     000000R  013    PLAY    044260R  002  PLIB   = 164004
PLOB   = 164002         PLSR   = 164000       PLSW   = 000322
PLVA   = 000320         POUT    000006R  013  PPIECE   000064R  011
PRCO16  001226R  005    PRESS  = 177452       PRHEAD   000360R  011
PRINTM  000000R  010    PRINVS  002316R  025  PRMSG    001520RG 012
PRMSG0  001530R  012    PRNO    000200R  011  PRNOS    001220R  012
PROUTB  001374R  012    PRX2    000122R  022  PS     = 177776
PSEND   001260R  023    PSTACK  000246R  014  PTABLE   000330R  007
PTYP   = 000014         PVSCOR  000000R  011  QDIRS    040036R  002
QPAWN  = 000042         QUEEN  = 000012       QUOTE    000332R  022
RCAST   000506R  024    RDIRS   040014R  002  RDY    = 000013
READPC  001112R  023    REL    = 000006       RELINE=  000015
RELOFF  044626R  002    REPEAT  000004R  024  REQCA    000632R  022
REQCAR= 000001          REQM   = 000003       REQMUP   000736R  022
RESEND  001036R  013    RESET   002056R  030  RESTOR   000000R  033
RETCAL  001250RG 025    RETCL1  000556R  026  RETCL2   000336R  026
R[illegible]    000004R  013    RINT   = 000100       RNODE    001206R  022
R[illegible]K  = 000010         ROUT    000012R  013  RPCH     000130R  022
RSEM    001064R  013    RULEOF  001212R  011  RWD    = 000200
R0     =%000000         R1     =%000001       R10$     000300R  033
R2     =%000002         R3     =%000003       R4     =%000004
R5     =%000005         S      = 000022       SCAN     001674R  004
SCANBI  000712R  003    SCOR   = 177770       SCORE    044620R  002
SE     = 000024         SELIN   001074RG 022  SELINE=  000004
SEND    001034R  013    SENDA   000376R  014  SENDAC   000000R  017
SENDAV  000424R  022    SENDC   000510R  014  SENDCM   001330RG 022
SENDCT  001362R  022    SENDEX  001402R  022  SENDW    001450R  022
SETBD   044254R  002    SETCM  = 000005       SETMVS   000000R  030
SETPIE  001116R  004    SETRPS  001470R  004  SINT   = 000040
SIX     044630R  002    SIXTY   044634R  002  SL     = 000057
SNDARR= 000014          SNDERR  000000R  016  SNODE    001134R  022
SP     =%000006         SPEC    000476R  024  SPT      044336R  002
SQCHAR  001552R  004    SQCOLR  000334R  004  SQUARE   032400R  002
SQ0    = 000002         SQ1    = 000004       SRCHOU   000000R  021
SSE    = 000014         SSW    = 000012       ST       044332R  002
START   000000R  003    STRG    041742R  002  STRGS    043744R  002
STRG0   042744R  002    ST26$   001416R  003  SW     = 000020
SWD    = 100000         SWEEP   000000R  005  SWITCH   000104R  022
SWPIEC  044256R  002    SWREG  = 177570       SYNC     000336R  022
S10$    000342R  030    S20$    000610R  030  S30$     000760R  030
S40$    001232R  030    S42$    001432R  030  S44$     002020R  030
S7[illegible]    000272R  030    S9$     000336R  030  TAB    = 000011
T[illegible]    000170R  010    TB     = 000011       TC       044306R  002
TEMPO  = 000021         TEN     044640R  002  TENTHO   044654R  002
TERM   = 000200         THOUSA  044650R  002  THR    = 000035
THRU   = 000017         THRUP  = 000025       TIMESU   002132R  005
TIMV0$  002342R  005    TIMV1$  002630R  005  TM       044312R  002
```

```
PPENDIX C : PARALLEL CHESS.    RT-11 MACRO VM02-12   20-NOV-78 09:43:52 PAGE 83+
YMBOL TABLE

C  CM    000002R     024  TOPPCE  044614R     002  TOPREL  044624R     002
    *    044302R     002  TSTRG   041736R     002  TU      044276R     002
YP    =  000012           T1      044262R     002  T2      000556R     024
3        044266R     002  T4      044272R     002  UFLAG   000126R     024
KING  =  000024           UPAWN =  000026           UPDATE  000000R     032
PMOVE    000160R     024  UPSCOR  002544RG    025  UP20$   002656R     025
ROOK  =  000020           U14$    000452R     032  U4$     000106R     032
AL    =  000020           VALUES  037712R     002  VARCD   000020R     024
ERIFY    002010R     004  VSCOR =  177776           W     =  000030
AITRP    000556R     022  WDSI    001054R     013  WDSIN   001044R     013
DSO      001060R     013  WDSOUT  001050R     013  WKING   000502R     024
NW    =  000034           WPDIRS  040056R     002  WROOK   000154R     024
SW    =  000016           ...V2 =  000001
  ABS.   000000     000
         000000     001
ARIAB    044660     002
NPUT     002066     003
NITIA    002314     004
AKEBD    004012     005
SQCOL    001652     006
HARMV    000354     007
RINTM    000200     010
VSCOR    002312     011
UMP      002032     012
LINK     001066     013
LISR     000602     014
I  T     000646     015
ENDER    000126     016
ENDAC    000064     017
UTBUF    000364     020
RCHOU    000150     021
INTPT    001512     022
NTPTC    001350     023
INVAR    000572     024
INMAX    003036     025
INMX2    001230     026
STMVS    000040     027
ETMVS    002162     030
STMVS    000572     031
PDATE    000512     032
ESTOR    000360     033
RRORS DETECTED: 0
REE CORE: 10627. WORDS

ARK,DK1:MARK=PRING,MAKEBD,PLINK,PINTPT,MINMXP,MARK3/N:ME:BIN
```