THE SCIENCE OF CHECKMATE

The purpose of this article is two fold. One is to assist a programmer currently involved in writing chess programs to reorganize the decision making process for move selection by the machine. The second purpose is to assist a human chess player to have an effective logical process for finding combinations leading to mate. There is a great deal of interest in chess playing computers from the hobby computer enthusiest as well as chess players. The first microcomputer chess tournament was held this year in San Jose. There are several retail chess playing computers programmed by the manufacturer currently on the market. There are sever al available programs written for the home hobby computer which may be purchased from your local game or computer store. Sargon, a program for the Z-80 developed by a husband and wife program team, Kathe and Dan Spracklen will be available this year. Their program finished in first place with a convincing 5 - 0 score against 10 other programs entered in the San Jose tournament. The complexity of programming a computer to play chess is enormous but if a computer were programmed to seek mating moves first in its decision making process before moving on lower level priorities such as exchange values, mobility, development and control of the center, it would play agressive chess and perhaps create threats not possible under current programming techniques. The game of chess is so exact, there must be a correct process of steps leading to mate. Current chess books provide variations of attacks from past games of masters, however different authors may provide conflicting view points explaining why the attack was successful. I will describe a logical process which will find mate at every opportunity where a mating combination exists. There are several elements which lead to mate. In this article I will dwell on the first two key elements. The four elements are: 1. the checking piece 2. moving a piece to support check 3. moving a piece that if captured, check can be given. 4. moving a piece that

denies response to a check. The first priority is the checking piece. There are three responses to a check, they are (A) capturing the checking piece (B) interposing (C) moving the king out of check. The second priority is moving a piece to support check. In the following examples you will see that we will deal only with the pieces involved in the mating combination and will not examine non related pieces on the chess board.

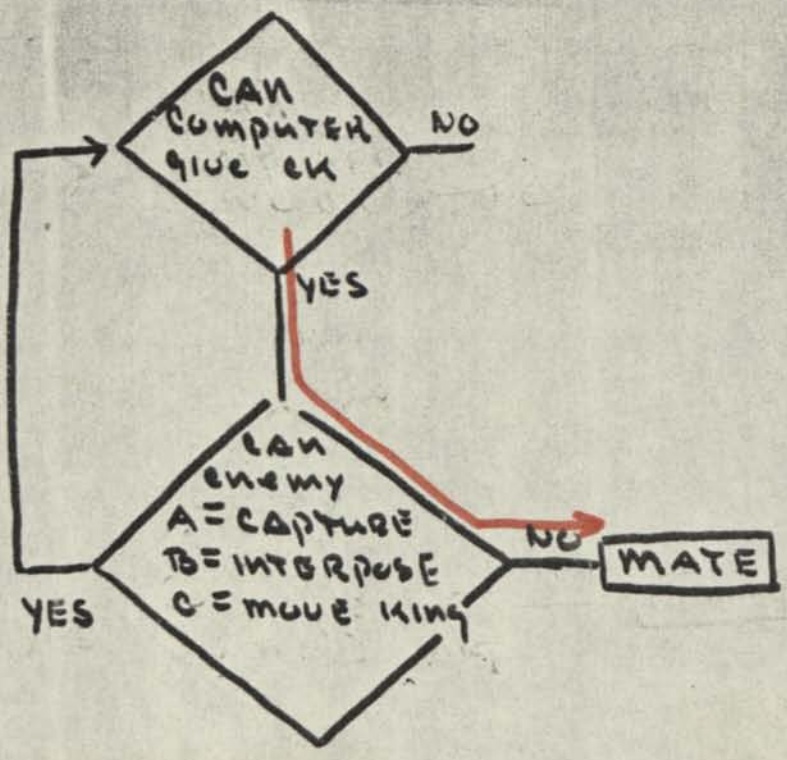
At the present time computer games may develop a mating position but usually as a result of the weakness of the opponent rather than due to the aggressiveness of the computer. For example the computer may make a move based on the value result of an exchange. If the machine has a choice of exchanging a queen for a knight or a pawn for a knight it will weigh the differences of the value of pieces after the exchange and will elect to exchange the pawn for a knight if it has a choice. The reason being the value of a knight is 3 points, the value of a queen is 9, and the value of the pawn is 1 point. By exchanging the pawn for a knight it will have an exchange value of plus 2 and by exchanging a queen for a knight it will have a negative value of minus 6. By comparing these two results it is easy to see that exchanging the pawn for the knight has more value than exchanging a queen for the knight. Why not program the machine to seek mate first and use the lower level decisions such as exchange value second and so on.

A programmer can do this by seeking a checking move first, the opponent can only respond to a check three ways; 1. capturing the checking piece 2. interposing 3. moving the king out of check. If the opponent cannot respond to these conditions the result is mate. (See following example:)

COMPUTER
BISHOP AT B2
QUEEN AT C3
KING AT G1

OPPONENT
KING AT G8
ROOK AT F8
PAWNS AT (F7)
(G7)
(H7)

The flow chart would look like this:



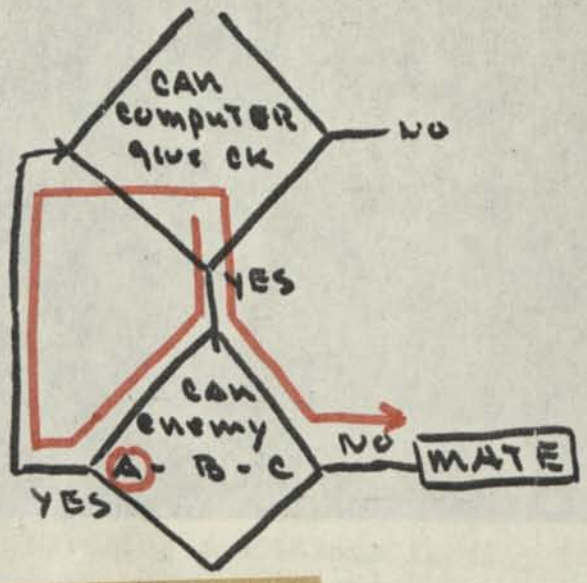
If the opponent can reppond positively to the checking move as in the following position the logic process is as follows: Can computer give check? The answer is yes. QxP+ Can the enemy capture? (yes) BxQ Since that condition is true or yes, following the flow chart can computer give check again is asked and the answer is (yes) R-QE8 The enemy cannot reppond by capturing, interposing, or moving the king. The result is mate.

COMPUTER

Bishop AT B2
Queen AT C3
Rook AT G1
Rook AT E1
King AT H1

OPPONENT

King AT G8
Bishop AT F8
PAWNS AT F7-G7-H7



A = CAPTURE
B = INTERPOSE
C = MOVE KING

and the opponents response to check. The new condition of moving a piece to support check and the opponents response to that support move which can be demonstrated by the following position:

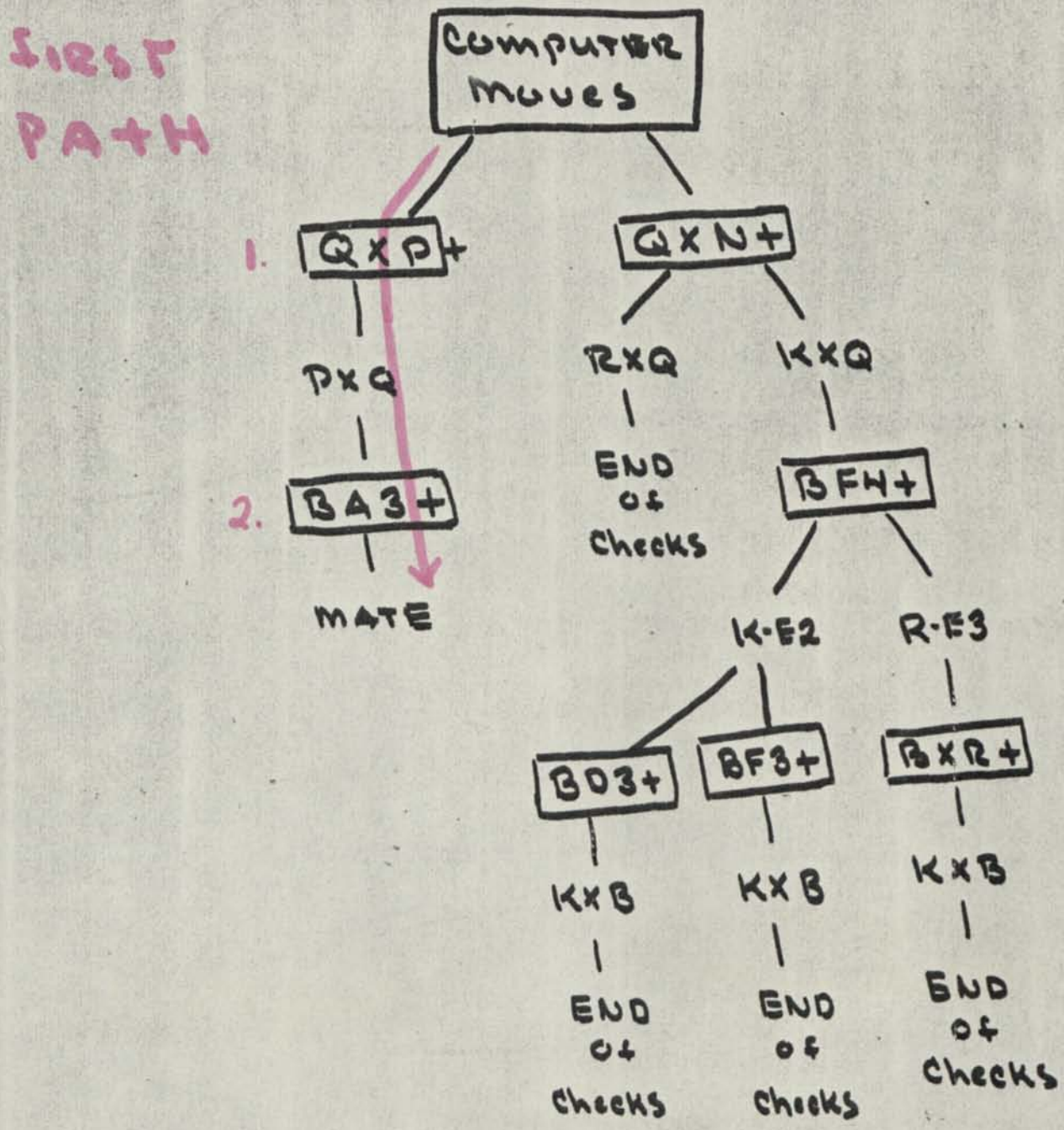
duplicate page 5

The following is a perfect example to examine in detail to see why this process works

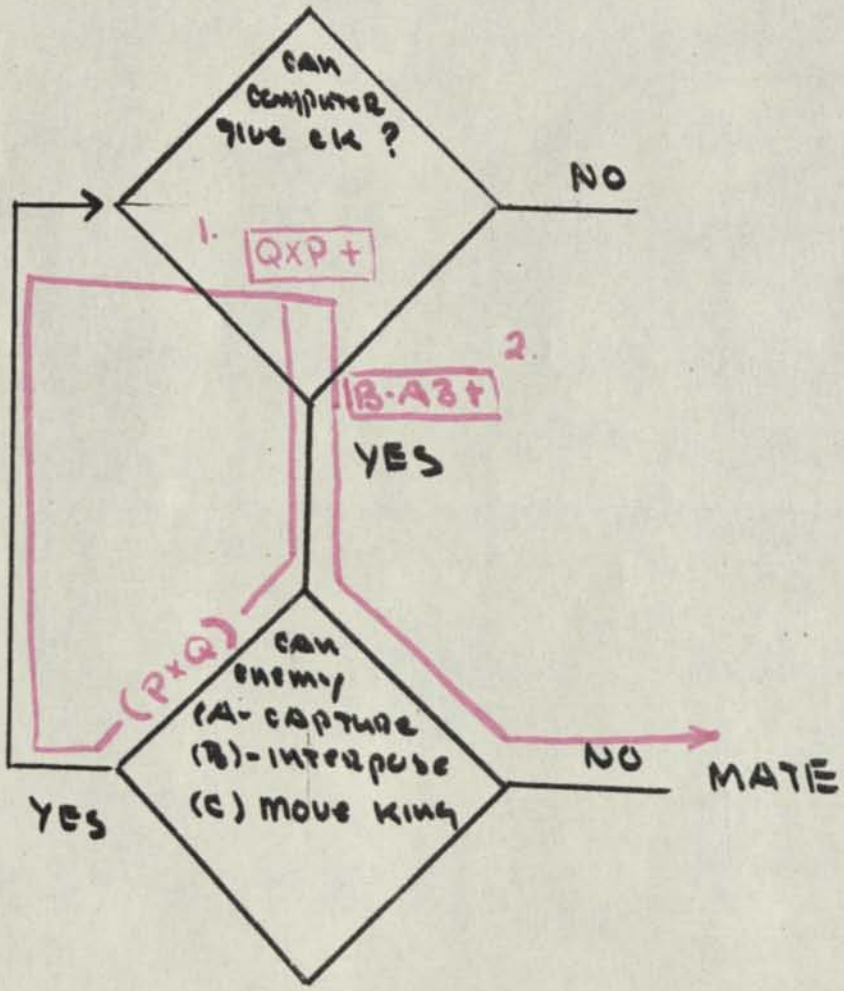
WHITE		Computer	
King	at C1	Queen	at E3
Rooks	at D1 and E1	Bishops	at E4
Knight	at D2		D6
Pawns	at A2-B2-C3		

This simple position is rich in tracking potential so we can do an exhaustive game tree search and compare it with the flow-chart as we go along. The computer is black & has the the move; the computer has a queen and two Bishops other pieces were removed as they are not important to the mating sequence.

You can see the computer has two possible checks; one is QXP ck and the second is QXN ck Lets examine the game tree with the flow-chart

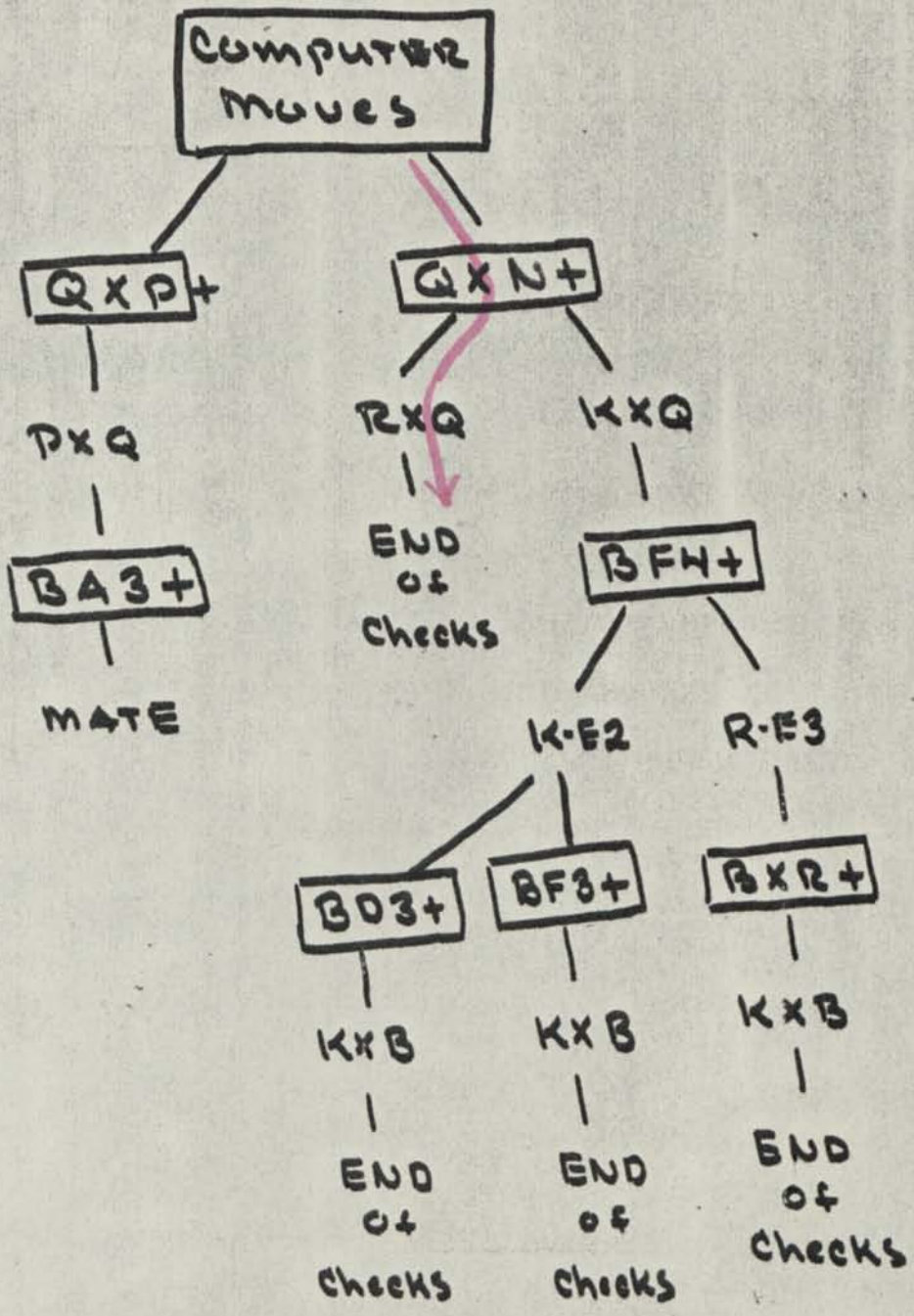


First Path

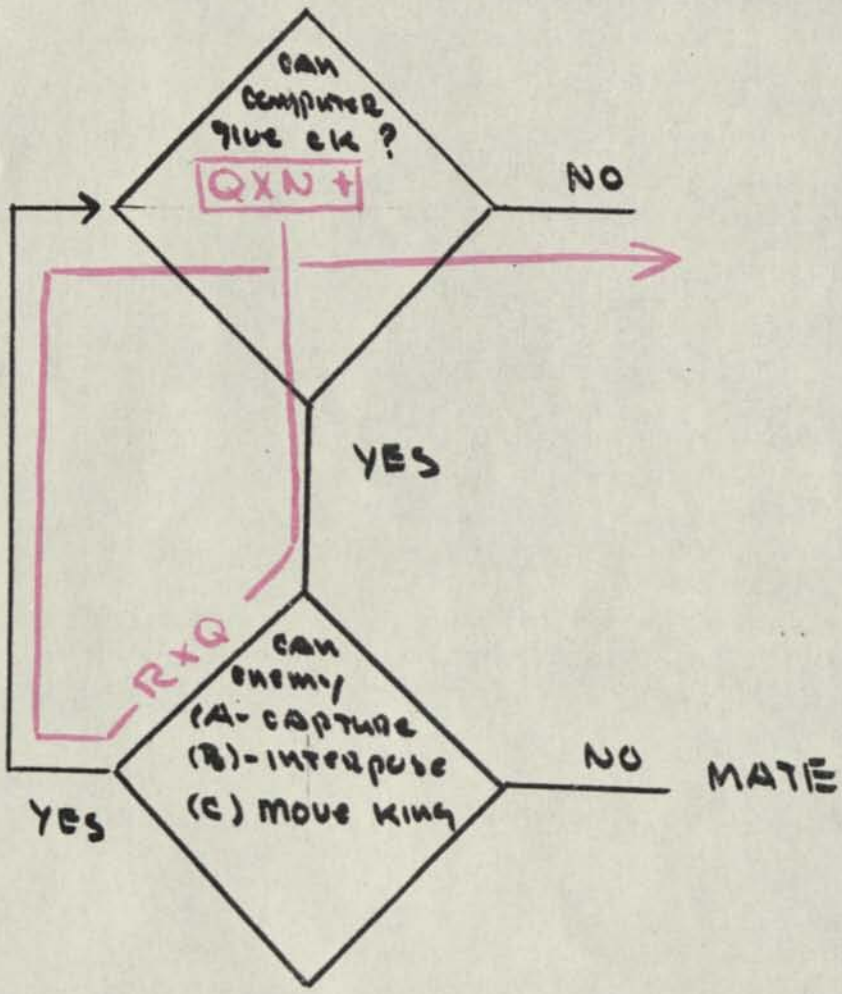

can computer give c/k ?
NO
1.
QxP+
2.
B-A3+
YES
(PxQ)
can enemy
(A- capture
(B)- interpose
(C) move king
NO
MATE
YES

As you can see the computer has two possible checks; one is QXP CK and the second is QXN CK Lets examine the game tree with the flow-chart

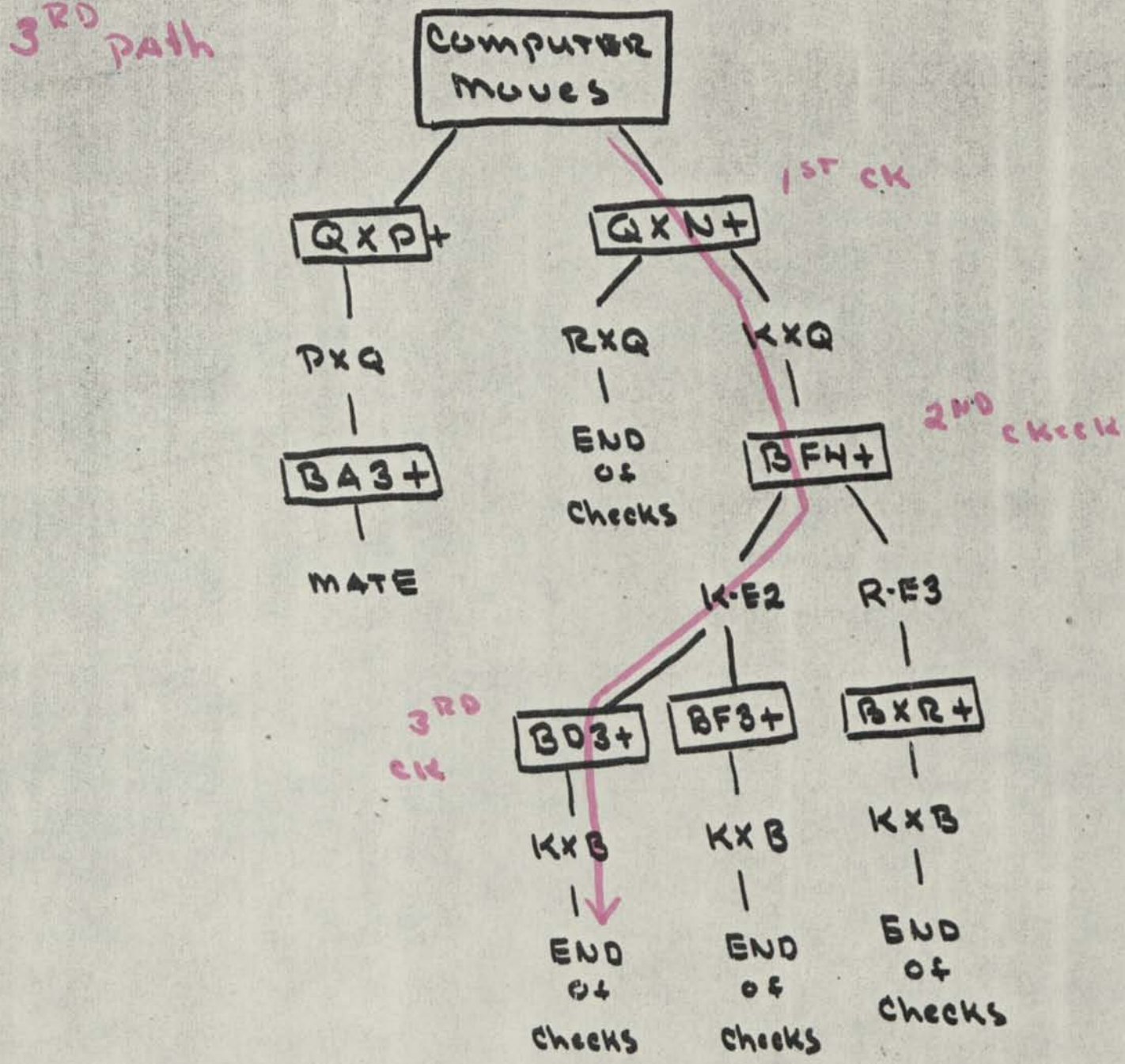
2ND PATH



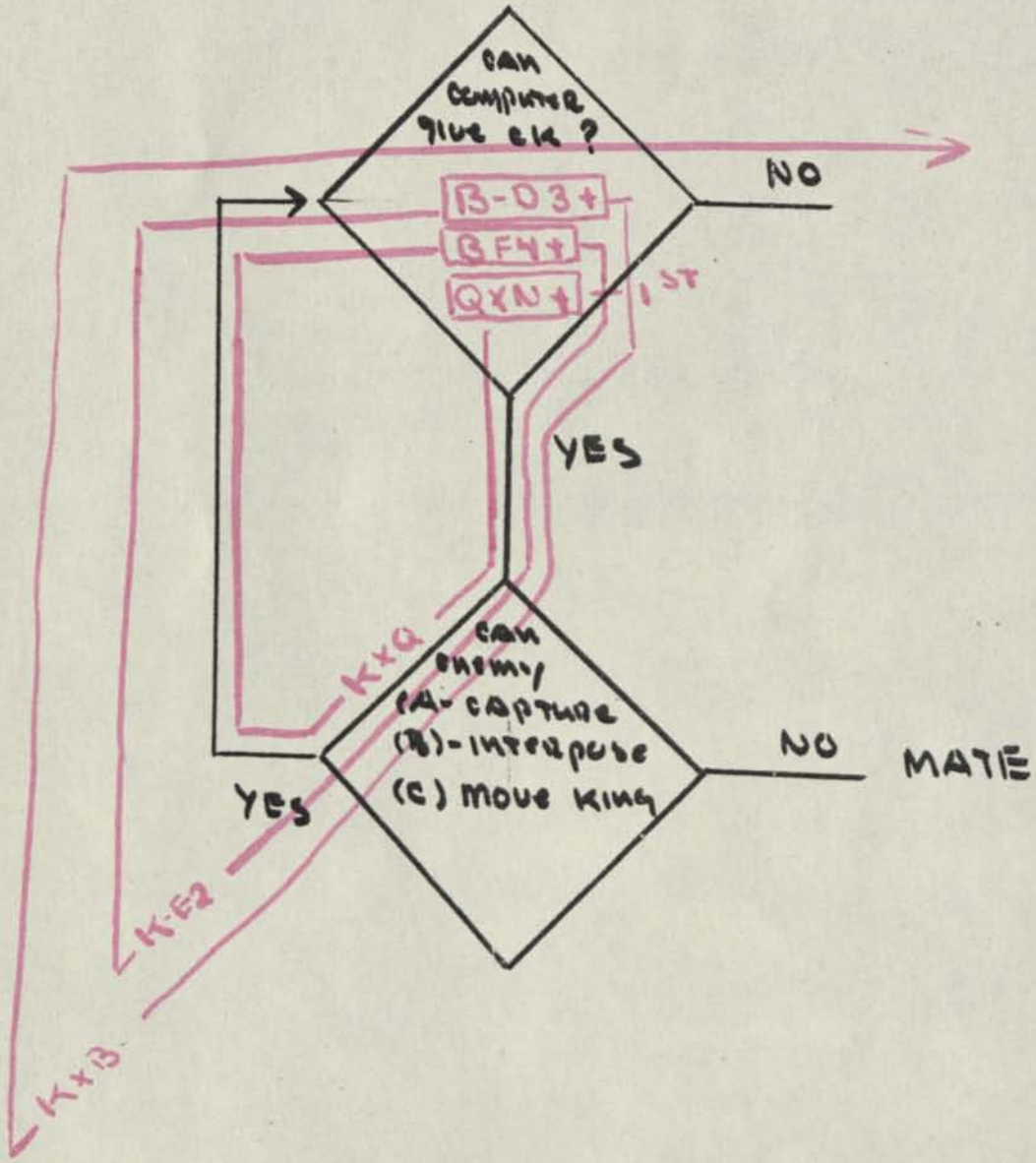
2ND PATH


CAN COMPUTER GIVE CH ?
QXN +
NO
YES
RXQ
CAN ENEMY
(A- CAPTURE
(B)- INTERPOSE
(C) MOVE KING
NO
MATE
YES

As you can see the computer has two possible checks; one is QXP CK and the second is QXN CK Lets examine the game tree with the flow-chart

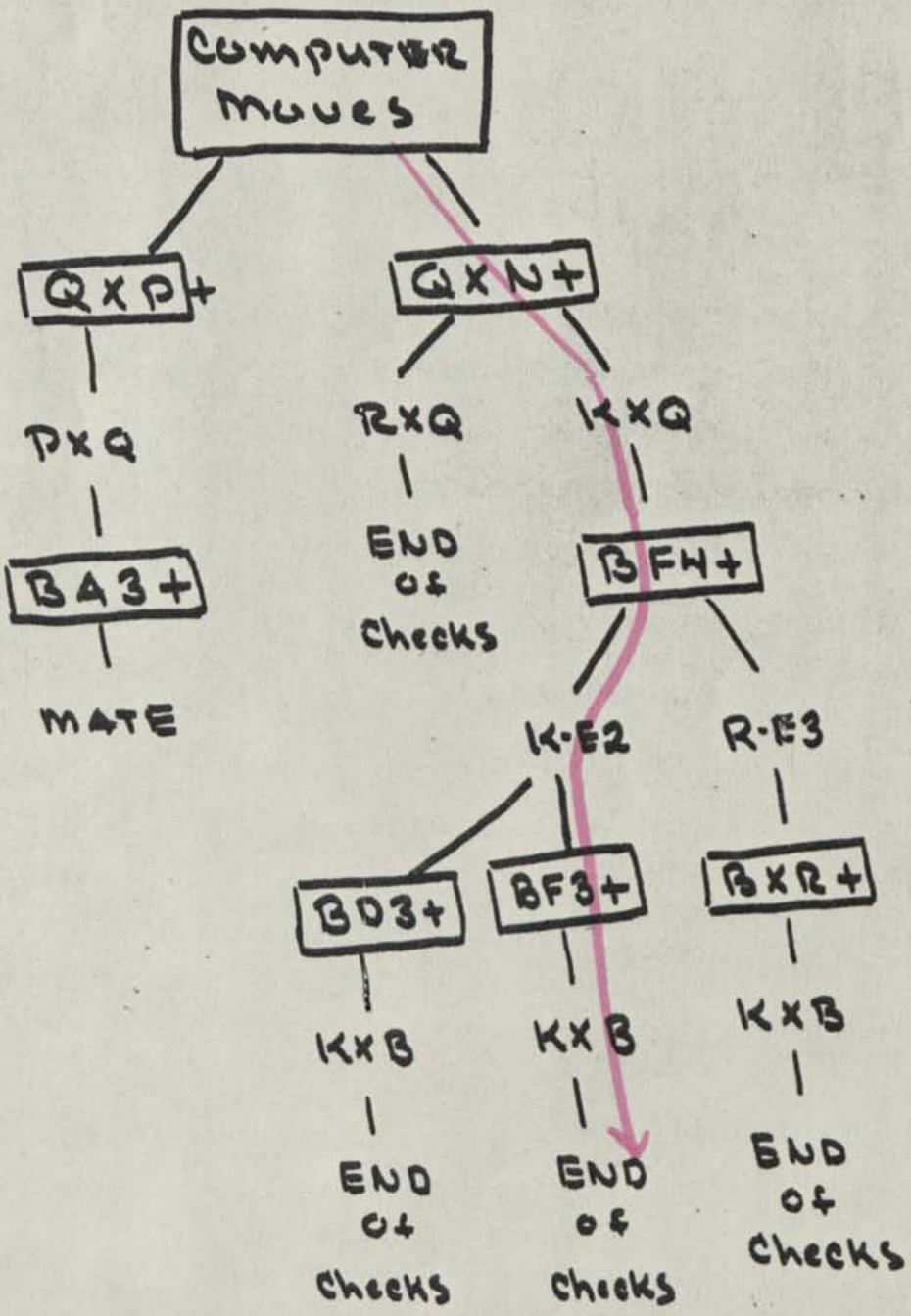


3RD PATH

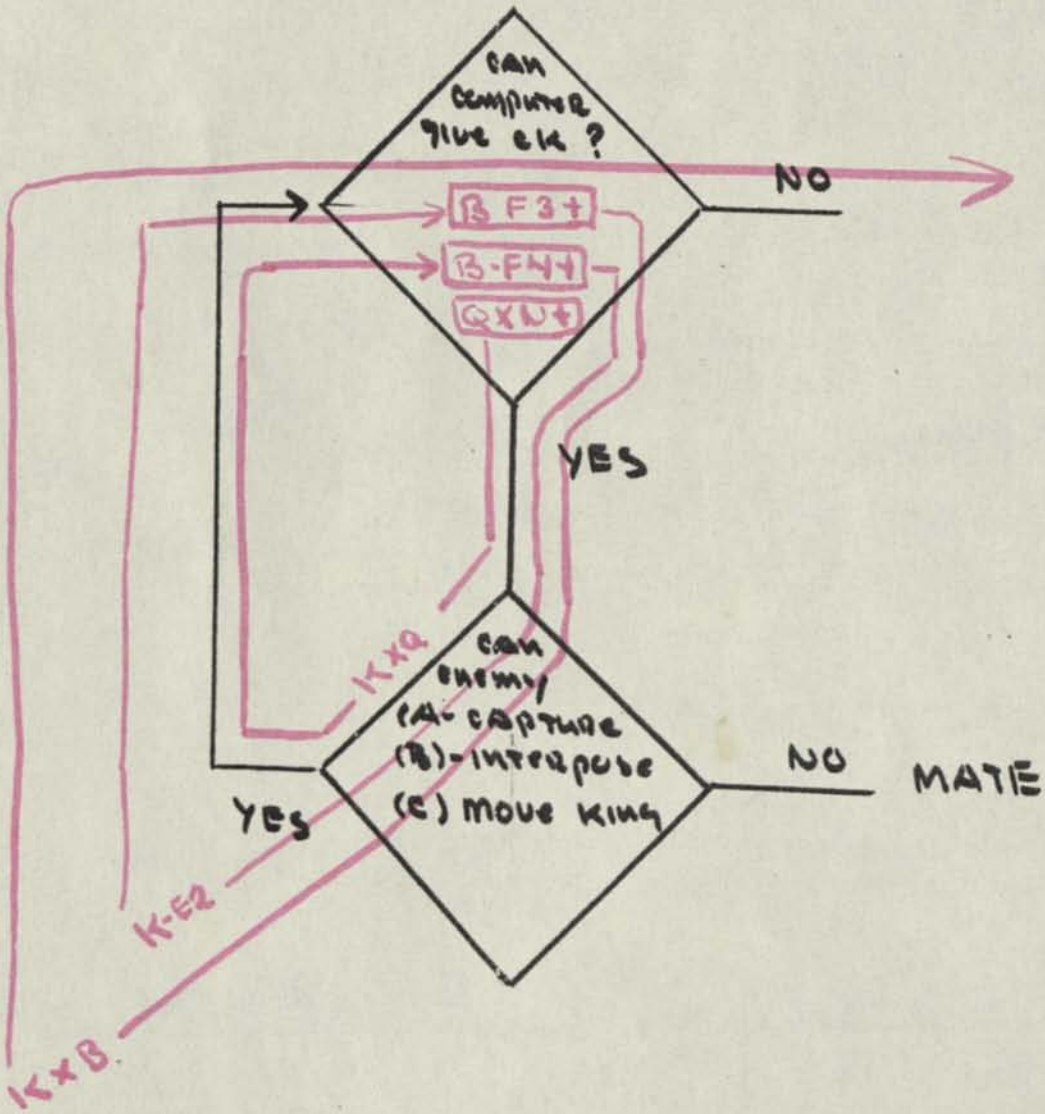

can computer give c/k ?
NO
B-D3+
BFH+
QXN+
1ST
YES
KxQ
can enemy (A- capture (B)- interpose (C) move king
NO
MATE
YES
K-B2
KxB

As you can see the computer has two possible checks; one is QXP CK and the second is QXN CK Lets examine the game tree with the flow-chart

4th path

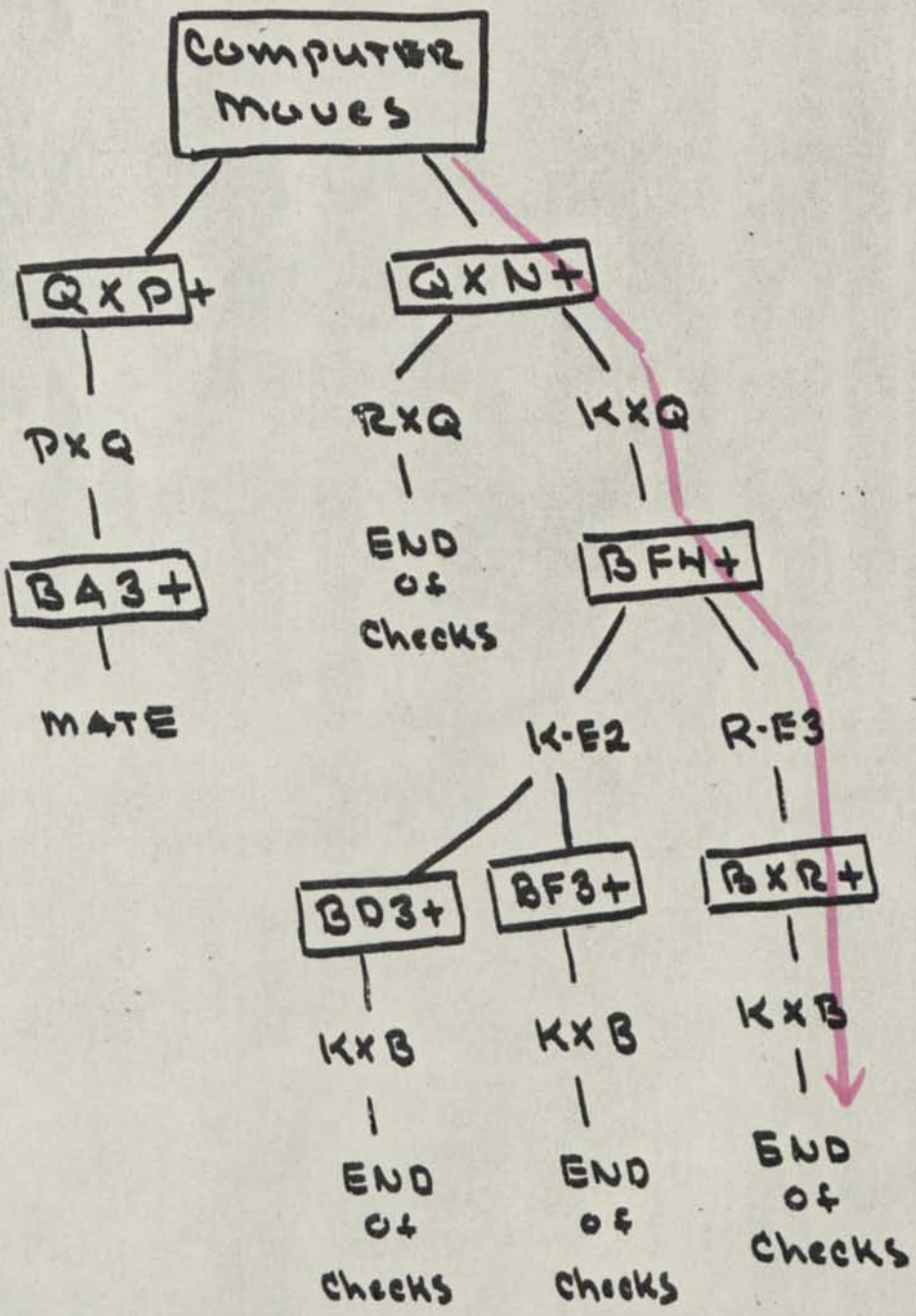


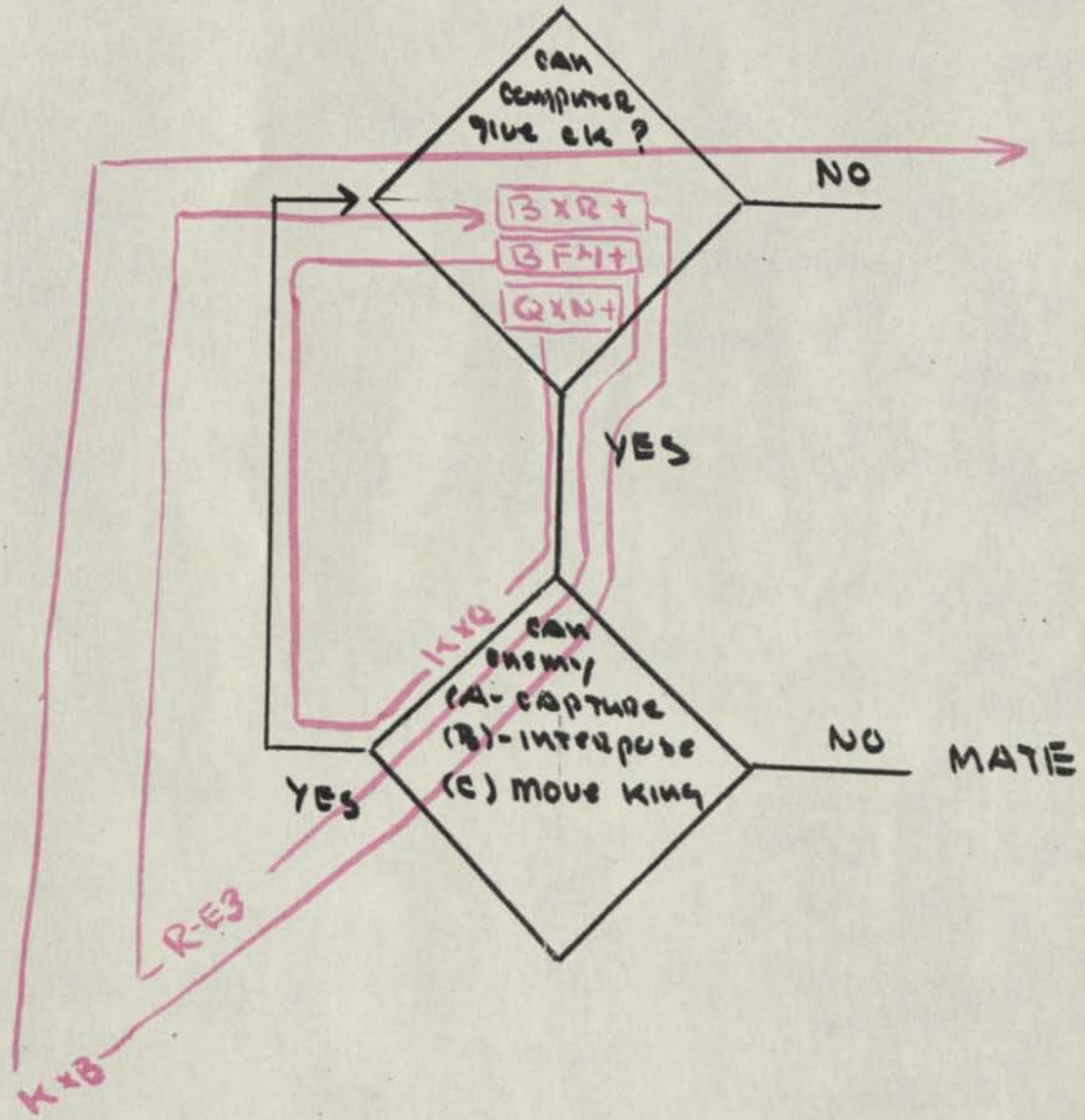
4th Path


can computer give chk ?
NO
B F3+
B-FN4
QXN4
YES
can enemy (A) capture (B)-interpose (C) move king
NO
MATE
YES
KxQ
K-E2
KxB

As you can see the computer has two possible checks; one is QXP ck and the second is QXN ck Lets examine the game tree with the flow-chart

5-th path





the ATTACKING conditions leading to mate ARE A RESULT OF giving CK AND the opponents Response to that check. the new condition of moving A piece to support check can be DEMONSTRATED be the following position

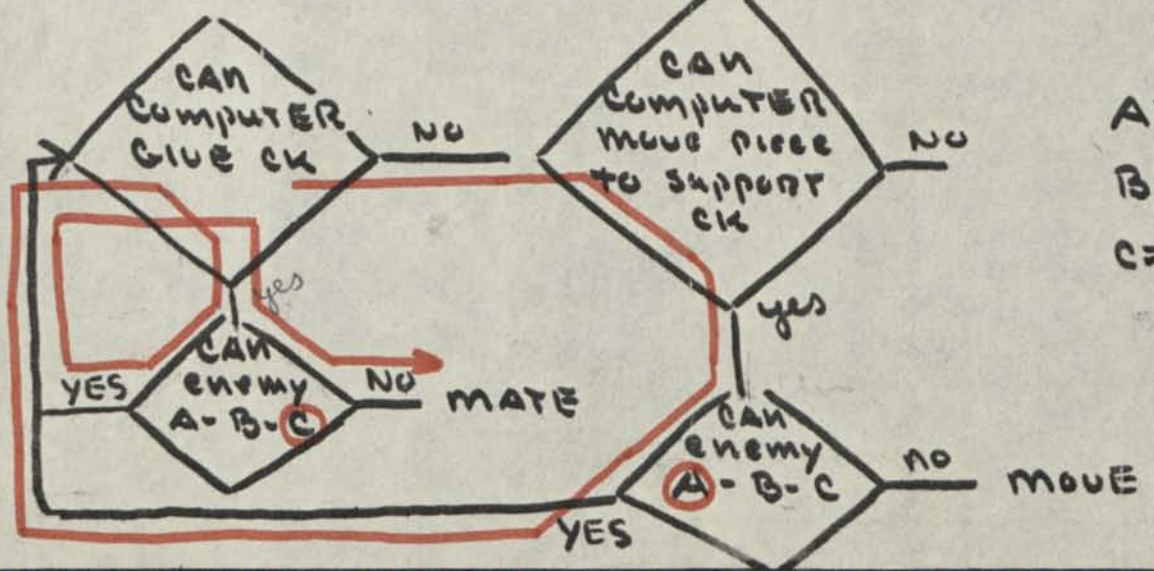
. Computer

King AT H1
Queen - F2
Rook - E1
Bishop - B2

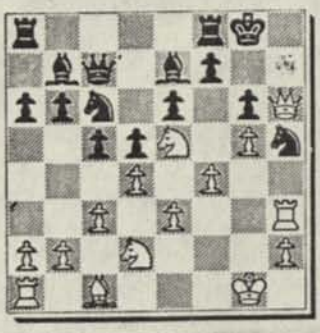
Opponent

King AT G8
Rook AT F8
Knight AT F6
Pawns AT F7-G7-H7

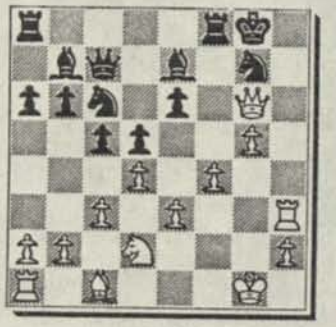
CAN computer give check- (NO) CAN computer move A piece TO support check? (yes) QXN supporting check AT G7. CAN that piece be captured? (yes) PXQ. if that piece is captured. CAN computer give check (yes) R TO G1 + CAN that piece be captured (NO) interposed (NO) move king (yes) K-H8 if condition is true CAN computer give check (yes) CAN opponent capture (NO) interpose (NO). move king (NO) the Result is MATE.



A



B

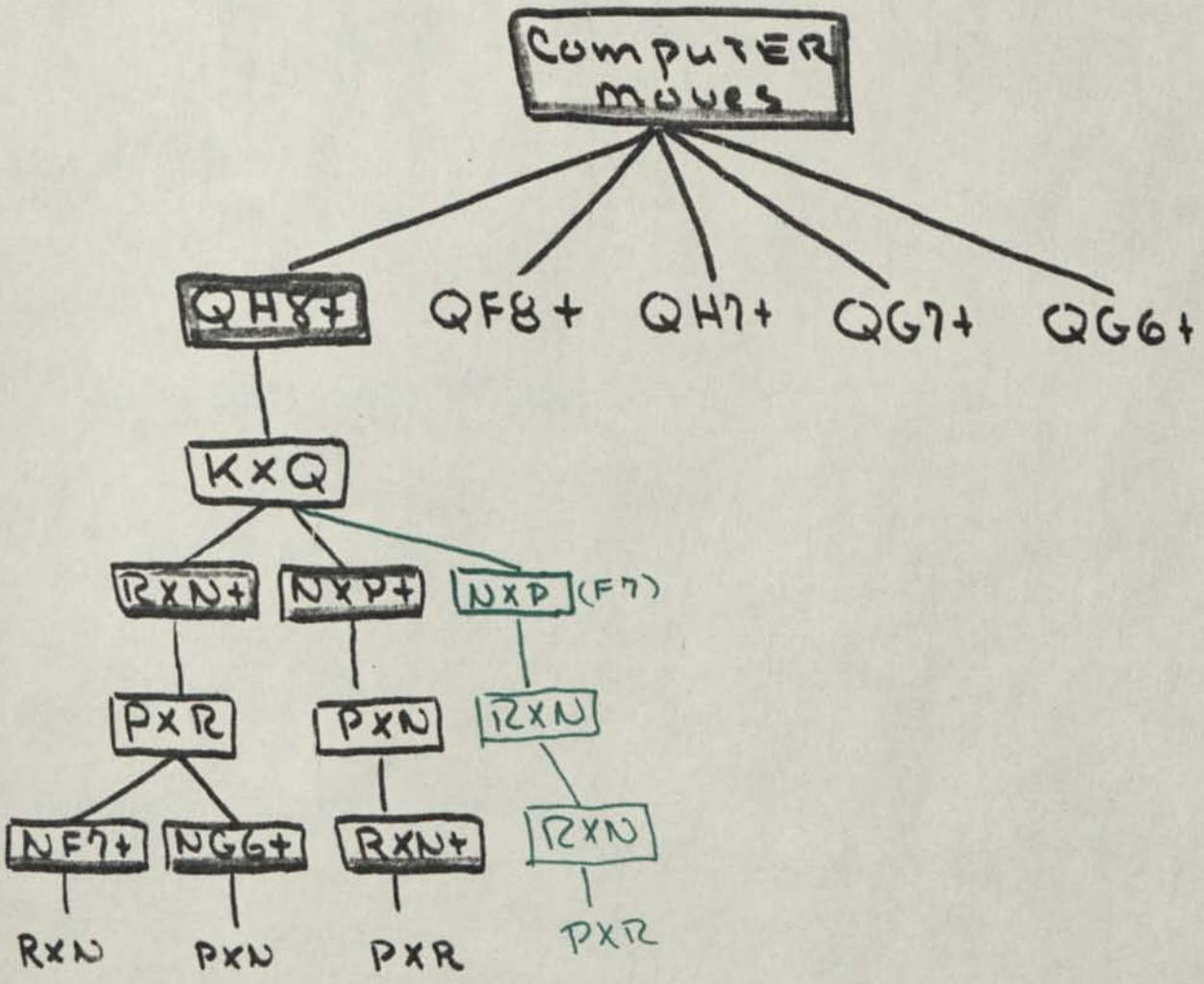


By combining the elements of a check and a move to support check it would work as shown in the above diagrams (A and B).

See diagram A - With white to move, white exhausts all the checking possibilities first and finds he must move to the next level of priority which is moving a piece to support check - which is knight takes pawn at G6 which supports check with the queen at H8. Following the flow chart, the question is asked - can this piece be captutured? Since the answer is yes - again following the flow chart, the question is asked - if this piece is captured - can I give check? The result is yes with QXP check forceing the condition B interposing with the knight which leads to diagram (B). Following the flow chart, white can force a win by giving check with the rook at H8 forceing condition A, king captures rook. If this piece is captured can you give check? The answer is yes - with queen to H6 check, forceing condition C in the flow chart - king moves to G8 which leads to the pawn move of G5 to G6 which is moving a piece to support check and mate follows next move.

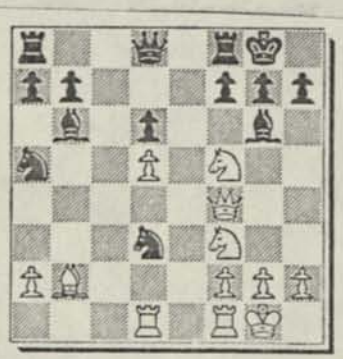
INSERT Showing
ONLY CAPTURES
AND CHECKS

POSITION A PAGE 6A



END of CHECKS

Let us examine the following diagrams: In the diagram on the left with Anderssen as white to move, by consulting the flow chart and exhausting the checking move first you will see that by moving the queen to H6 is moving a piece to support check. It wins the game by force as you can see, the queen can be captured by the pawn. The follow up of course is giving check by knight takes pawn mate.
In the diagram on the right this is the position of a game played in 1920 at Zurich. With white to move Teichmann played rook takes pawn at H6. Can you see why? If not consult the flow chart. There are no checking moves but this move, moves a piece to support check at H8. Black responded with knight takes rook and white moved its queen to G5 which is moving a piece to support check at D8. Black reinforced the checking square by knight to F7 and white moved queen to D8 check. Black captured the queen with the knight and white moved pawn to H6 which is moving a piece to support check. There is no way out for black. He resigned.



AMATEUR

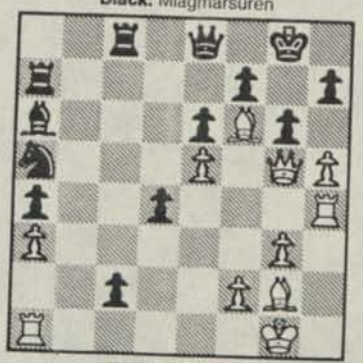
ANDERSSEN

A MAGNIFICENT finish from an odds game by Teichmann, played at Zurich in 1920.

1 R×Pl

examine the following position. fisher moved to support check at G7 after exausting the checking possibilities first. BLACK Reinforced the position with Q-F8 (see flowchart) ASK YOURSELF can you give check (yes) QXP+ (H7) can that piece be captured (yes) condition (A) can you give check if that piece is captured (yes) PXP Dis check can the checking piece be captured meeting condition A (yes) if that piece is capture can you give check (yes) B to E4 + can the checking piece be captured no interposed no, can the king move no the result MATE.

HARD: Sousse 1967
Black: Miagmarsuren



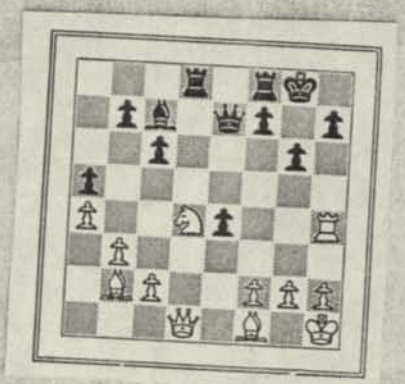
White: Fischer

White to move

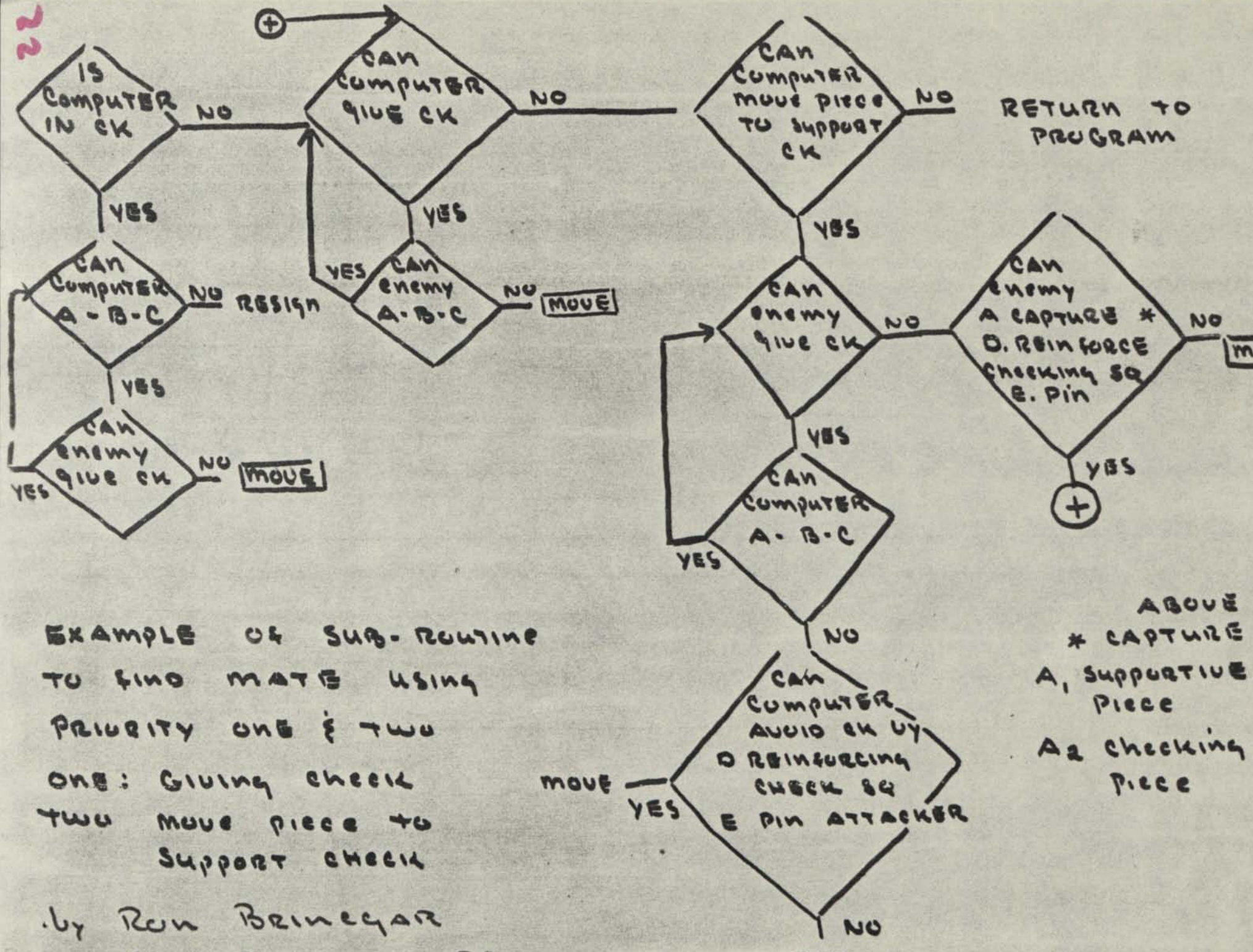
fisher

1. Q-H6 Q-F8
2 QXP + KXQ
3. PXP(dis +) KXP
4. BE4 MATE

Here are two examples of a supportive move which leads to mate as described in the flow chart. The first diagram on the left is a problem composed by Max Ewe in 1920. White moves queen to D6 which is moving a piece to support check. If you follow the flow chart which says that this piece can be captured (condition true). If captured can you give check again? The answer is yes by moving the rook to C1, the result is mate.



The diagram on the right is another example of moving a piece to support check. With white to move, white moved knight to F5 which supports check at E7 and H6. If this piece is captured by the pawn check can be given by moving the rook to G4 which leads to mate. Black responded not with capturing the knight but queen takes rook at H4 which guards against both mates and threatens queen takes pawn at H2 and leaves whites queen vulnerable to attack by black's rook at D8. White responded with the move queen to H5 which again is moving a piece to support check. It supports check with the knight at H6. Please note the queen cannot be captured by either the pawn or the black queen because white would mate with the next move.



EXAMPLE OF SUB-ROUTINE TO FIND MATE USING PRIORITY ONE & TWO

ONE: GIVING CHECK

TWO MOVE PIECE TO SUPPORT CHECK

ABOVE

* CAPTURE

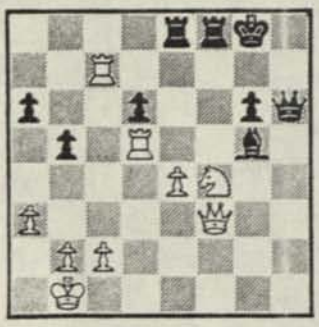
A, SUPPORTIVE PIECE

A2 CHECKING PIECE

by Ron Brinegar
6107 Pettinger Rd
Linden, CA 95236

In the following positions Bobby Fischer forces a win in diagram C with a queen move to B3 which moves a piece to support check by moving the rook next move. In diagram D in the famous Fischer Byrne game for the United States Championship 1963-64, Fischer as black moved queen to D7 which supports check at H3 and Byrne resigned. In diagram E, Fischer again forces a win by moving a piece to support check with bishop to C1 which supports check at F4. In examining diagram F, we look at the famous Morphy sacrifice against Paulsen with queen takes bishop at F3. This is moving a piece to support check, following the flow chart you will see that if that piece is captured check can be given by the rook at G6. Morphy followed up the check with a supportive move to a check by moving the bishop to H3.
One final example of a move which supports check is in the following diagram G. With white to move he first exhausts the checking possibilities and then moves the queen to E5 which supports check at E8. If this piece is captured, check follows with the rook at D8 forcing mate.

C.

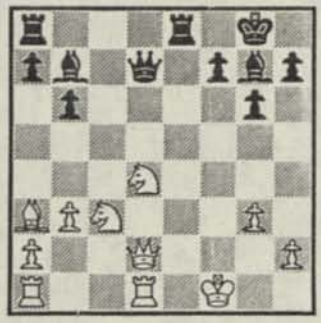


BOLBOCHAN

Position after 34
P×P

FISCHER

D. 21 . . . Q-Q2!



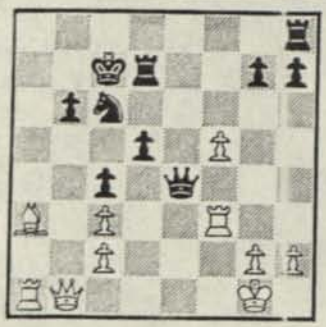
FISCHER

Final Position after
21 . . . Q-Q2!

BYRNE

White resigns

E



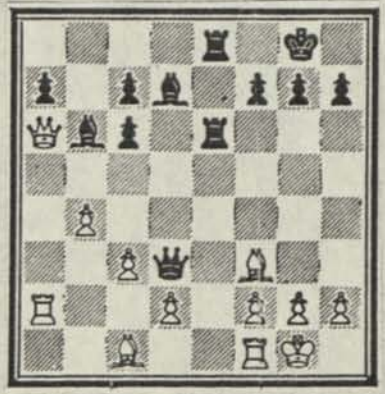
DARGA

Position after 26
K - B2

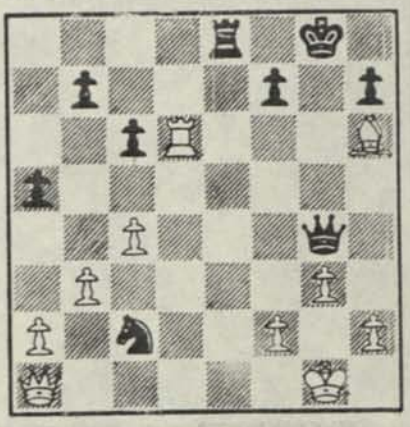
FISCHER

Problem: White to play and win.

F.



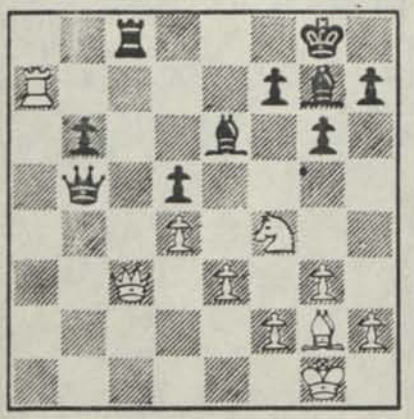
G



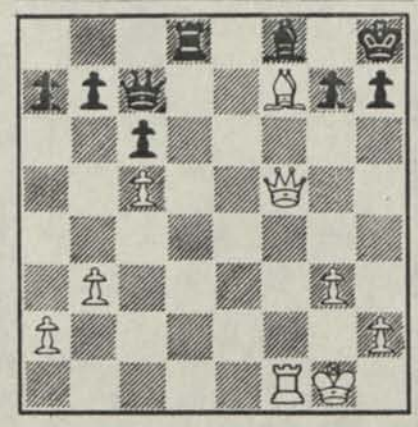
As explained earlier there are other priorities in forceing a checkmate. That is the third priority of moving a piece that if captured check can be given. In diagram H, with Pal Benko with the white pieces to move, captured the bishop with his knight at E6. It leaves his queen open to capture by the enemy rook, however check follows with rook to A8 and mate cannot be stopped. If the knight is captured queen takes rook at C8. Another famous diagram is I with white to move. White moved bishop to E8 which supports a check at F8. If the bishop is captured by the enemy Rook, queen takes bishop check forcing mate, The fourth priority is moving a piece that denies response to a check can be demonstrated in the last diagram J which occured between Larsen and Rogoff in Lone Pine 1978. After rook takes pawn check at A7 if black king captured rook, white responds with queen takes pawn at C6 which denies the black king of condition C when followed up with a checking move of rook F1 to A1.

In summary if a computer seeks mate first by the use of the checking move the enemies response to that check and moving a piece to support check it will deal with the pieces involved only in the mate combination and not every piece on the board. If a mate is there it will find it. If a mate is not there it will then go to the lesser priorities such as mobility, exchange value, developement, control of the center, pawn structure, and so on.

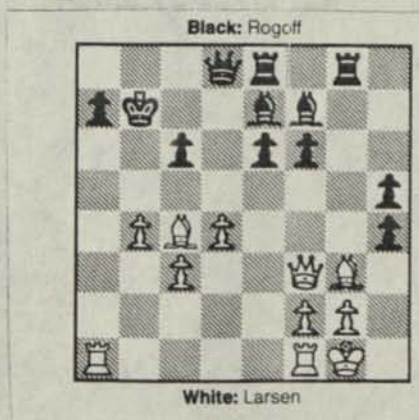
(H)



(I)



(J)



236

The following article is perhaps the seeds of an idea than can be developed into an exacting process for playing the game of chess. Most chess books explain the principles well enough to improve your game. The outline of the idea previously explained is by no means complete. I have mentioned four programmable reasons leading to a mating position. The first is the checking piece, the second is moving a piece to support check, the third is moving a piece that if captured check can be given. The fourth is moving a piece that will deny a response to a check such as capture, interposing, or denying the enemy king the right to move out of check. For every element stated there are others waiting to be found but the level of difficulty increases proportionately. The first two priorities are most important. If you look at game 5 between Spassky and Fischer, Fischer had the black pieces and moved 27 bishop takes pawn at A4. This does not fit with any of the above elements. Perhaps it could be stated that Fischers move was moving a piece that if captured a piece can be moved that threatens check, and so it grows. The idea is there, the moves are there, it is up to someone to write the program on a machine language level. Can you imagine playing an opponent that seeks mate first and foremost before considering other moves.

Sincerely yours

Ron Brinegar

```
ATLAS ASSEMBLER VER 2.0                  LOADR                PAGE 0001

0F00                    0010 ************************************************
0F00                    0020 * BOOTSTRAP LOADER FOR 'MICROCHESS'             *
0F00                    0030 *                                               *
0F00                    0040 *  MICROCHESS IS A PRODUCT OF MICRO-WARE LTD.   *
0F00                    0050 ************************************************
0F00                    0060 * MANUALLY LOAD BOOTSTRAP AT 'F00' HEX.        *
0F00                    0070 ************************************************
0F00                    0080 * INSTRUCTIONS:                                 *
0F00                    0090 * 1.  LOAD TAPE INTO READER ANYWHERE BEFORE     *
0F00                    0100 *     THE DATA STARTS.                          *
0F00                    0110 * 2.  EXECUTE FROM ADDRESS 'F00' HEX.           *
0F00                    0120 * 3.  START THE READER AND ALLOW PAPER TAPE     *
0F00                    0130 *     TO BE READ INTO MEMORY.                   *
0F00                    0140 * 4.  IF THE TAPE READS IN CORRECTLY THE        *
0F00                    0150 *     COMPUTER WILL HALT.                       *
0F00                    0160 * 5.  IF A CHECKSUM ERROR OCCURS TWO THINGS     *
0F00                    0170 *     HAPPEN:                                   *
0F00                    0180 *     A. ON ALTAIR COMPUTERS THE 'INTE' LIGHT   *
0F00                    0190 *        WILL LIGHT.                            *
0F00                    0200 *     B. THE ADDRESS LIGHTS WILL DISPLAY A      *
0F00                    0210 *        STEADY ADDRESS OF '0F33'.              *
0F00                    0220 *     IF THIS OCCURS YOU WILL HAVE TO RELOAD.   *
0F00                    0230 ************************************************
0F00                    0240        ORG  0F00H
0F00 21 00 00           0250 LOADR  LXI  H,0
0F03 45                 0260        MOV  B,L
0F04 4D                 0270        MOV  C,L
0F05 11 00 0E           0280        LXI  D,0E00H
0F08 CD 1F 0F           0290 INTL   CALL INPUT
0F0B B9                 0300        CMP  C
0F0C CA 08 0F           0310        JZ   INTL
0F0F 77                 0320 LOOP   MOV  M,A
0F10 80                 0330        ADD  B
0F11 47                 0340        MOV  B,A
0F12 23                 0350        INX  H
0F13 1B                 0360        DCX  D
0F14 7A                 0370        MOV  A,D
0F15 B3                 0380        ORA  E
0F16 CA 29 0F           0390        JZ   DONE
0F19 CD 1F 0F           0400        CALL INPUT
0F1C C3 0F 0F           0410        JMP  LOOP
0F1F DB 00              0420 INPUT  IN   0
0F21 E6 40              0430        ANI  40H
0F23 CA 1F 0F           0440        JZ   INPUT
0F26 DB 01              0450        IN   1
0F28 C9                 0460        RET
0F29 AF                 0470 DONE   XRA  A
0F2A B8                 0480        CMP  B
0F2B C2 2F 0F           0490        JNZ  ERROR
0F2E 76                 0500        HLT
0F2F FB                 0510 ERROR  EI
0F30 C3 30 0F           0520        JMP  ERROR+1

DUMP F00 F3F
0F00 21 00 00 45 4D 11 00 0E CD 1F 0F B9 CA 08 0F 77
0F10 80 47 23 1B 7A B3 CA 29 0F CD 1F 0F C3 0F 0F DB
0F20 00 E6 40 CA 1F 0F DB 01 C9 AF B8 C2 2F 0F 76 FB
0F30 C3 30 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Service Is Our Specialty - Satisfaction Our Guarantee
SERVICE ON ALL BRANDS

Sales ZENITH Service

GEORGE T.
LYON APPLIANCE
Tuolumne County's Oldest Dealer

Phone (209) 532-4894

Mono Village Center
19515 Village Dr. Sonora, Calif. 95370

```
0F00                    0010 ************************************************
0F00                    0020 * BOOTSTRAP LOADER FOR 'MICROCHESS'           *
0F00                    0030 *                                              *
0F00                    0040 *  MICROCHESS IS A PRODUCT OF MICRO-WARE LTD.  *
0F00                    0050 ************************************************
0F00                    0060 * MANUALLY LOAD BOOTSTRAP AT 'F00' HEX.       *
0F00                    0070 ************************************************
0F00                    0080 * INSTRUCTIONS:                               *
0F00                    0090 * 1.  LOAD TAPE INTO READER ANYWHERE BEFORE    *
0F00                    0100 *     THE DATA STARTS.                         *
0F00                    0110 * 2.  EXECUTE FROM ADDRESS 'F00' HEX.          *
0F00                    0120 * 3.  START THE READER AND ALLOW PAPER TAPE    *
0F00                    0130 *     TO BE READ INTO MEMORY.                  *
0F00                    0140 * 4.  IF THE TAPE READS IN CORRECTLY THE       *
0F00                    0150 *     COMPUTER WILL HALT.                      *
0F00                    0160 * 5.  IF A CHECKSUM ERROR OCCURS TWO THINGS    *
0F00                    0170 *     HAPPEN:                                  *
0F00                    0180 *     A. ON ALTAIR COMPUTERS THE 'INTE' LIGHT  *
0F00                    0190 *        WILL LIGHT.                           *
0F00                    0200 *     B. THE ADDRESS LIGHTS WILL DISPLAY A     *
0F00                    0210 *        STEADY ADDRESS OF '0F33'.             *
0F00                    0220 *     IF THIS OCCURS YOU WILL HAVE TO RELOAD.  *
0F00                    0230 ************************************************
0F00                    0240        ORG   0F00H
0F00 21 00 00           0250 LOADR  LXI   H,0
0F03 45                 0260        MOV   B,L
0F04 4D                 0270        MOV   C,L
0F05 11 00 0E           0280        LXI   D,0E00H
0F08 CD 1F 0F           0290 INTL   CALL  INPUT
0F0B B9                 0300        CMP   C
0F0C CA 08 0F           0310        JZ    INTL
0F0F 77                 0320 LOOP   MOV   M,A
0F10 80                 0330        ADD   B
0F11 47                 0340        MOV   B,A
0F12 23                 0350        INX   H
0F13 1B                 0360        DCX   D
0F14 7A                 0370        MOV   A,D
0F15 B3                 0380        ORA   E
0F16 CA 29 0F           0390        JZ    DONE
0F19 CD 1F 0F           0400        CALL  INPUT
0F1C C3 0F 0F           0410        JMP   LOOP
0F1F DB 00              0420 INPUT  IN    0
0F21 E6 40              0430        ANI   40H
0F23 CA 1F 0F           0440        JZ    INPUT
0F26 DB 01              0450        IN    1
0F28 C9                 0460        RET
0F29 AF                 0470 DONE   XRA   A
0F2A B8                 0480        CMP   B
0F2B C2 2F 0F           0490        JNZ   ERROR
0F2E 76                 0500        HLT
0F2F FB                 0510 ERROR  EI
0F30 C3 30 0F           0520        JMP   ERROR+1

DUMP F00 F3F
0F00 21 00 00 45 4D 11 00 0E CD 1F 0F B9 CA 08 0F 77
0F10 80 47 23 1B 7A B3 CA 29 0F CD 1F 0F C3 0F 0F DB
0F20 00 E6 40 CA 1F 0F DB 01 C9 AF B8 C2 2F 0F 76 FB
0F30 C3 30 0F 00 00 00 00 00 00 00 00 00 00 00 00 00
```