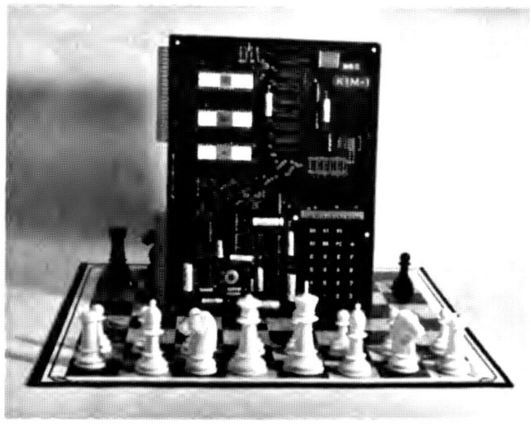
Rick Grossman
4007 Constellation Rd.
Lompoc CA 93436

KIM + Chess = Microchess

Listen to this: A challenging game of chess, for the beginner and intermediate player, can be played in KIM's 1K of memory.

Microchess is a program designed to play chess on a KIM-1 6502 microcomputer system with no additional memory or peripheral equipment. The documentation supplied with the cassette tape consists of a player's manual, a complete source program listing and an excellent programmer's manual, which includes suggestions for expansion and modification of the program.

When I first received the program I was skeptical that a chess program utilizing only 1.1K of memory would be capable of playing a reasonably good game of chess. Peter Jennings, the author of the program, states in the player's manual, "Microchess does not play an expert game of chess." However, I found it does present quite a challenge to the average player and is an excellent teaching aid for the novice. For the intermediate player it is a good way to practice openings and methods of achieving checkmate.



The Program

The design of the program allows modification in the level of play to reduce the computer's response time to suit the ability of the player. The normal response time for the computer to decide its move is 100 seconds. A simple change in data at two locations can reduce this time to either three or ten seconds. Not having played chess in years and also never having been a match for Bobby Fischer, I found this feature beneficial in allowing me to win a few times while sharpening my own game.

Perhaps the one major difference in playing the game with Microchess is the special octal notation used to position the pieces on the chessboard. As shown in Fig. 1, each square is identified by a two-digit octal number. The first digit identifies the rank of the piece, and the second number the file. Moves are made by using this notation. For instance, if you wanted to move the black pawn to king 4, you would move from square 63 to square 43. I found this method easier to get along with by writing the identifying number on each square of the board.

Once the pieces are positioned on the board, the program is loaded, in two blocks of data, from the cassette tape. If you wish to have KIM play a specific opening, select one of the five opening plays, enter the ID of the opening selected, and then load it from the tape. The openings available to the player are: Four Knights, French Defense, Guioco Piano, Ruy Lopez and Queens Indian.

The opening, consisting of nine moves per side, will follow the established lines of play familiar to most chess players. I also found that when trying to get the game started with a specific opening, such as Ruy Lopez, the computer would not make the initial move unless I pressed RS, GO, C and PC, after the program is loaded . . . then the display will indicate the computer's move.

Playing Chess

You are now ready to play chess. The game is initiated by pressing RS and GO on the KIM keyboard. When the PC key is depressed, the computer will analyze the position of the pieces on the board and store it in memory. During this time (approximately 100 seconds), the display will darken and flash until the move has been decided. The computer will then display the move it wishes to make. The computer's moves are displayed as shown in Fig. 2a. The display indicates that king pawn is to be moved from king pawn 2 to king pawn 4 (the computer is playing white).

The player then inserts his move in the same manner by keying in the from-to notation and then depressing F on the KIM keyboard to insert the move into the memory (see Fig. 2b). The player has moved his king pawn from 63 to 43. The FF notation indicates the move has been entered in the memory. To continue the game depress PC, and the computer will then make its move.

There are three moves the computer will not make by itself: Castling, En Passant Capture and Queening Pawns. These will have to be done for the computer by simple keyboard operations that are described in the player's manual. In order to reduce memory requirements, these moves are not included in the basic program.

It should also be noted that the computer does not verify the legality of a move, nor does it warn you if your king is in check. As a matter of fact, it will capture the king by moving one of its pieces to the square occupied by the king, if at all

COMPUTER

00	01	02	03	04	05	06	07
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37
40	41	42	43	44	45	46	47
50	51	52	53	54	55	56	57
60	61	62	63	64	65	66	67
70	71	72	73	74	75	76	77

PLAYER

Fig. 1. Chessboard notation.

0	F	1	3	3	3

Fig. 2a. Computer's move.

F	F	6	3	4	3

Fig. 2b. Player's move.

possible.

As previously mentioned, I am not a proficient player, so I decided to test Microchess against one of the better chess players I know. This young man sat down with KIM and a chessboard for three and one half hours playing a game against the Microchess program. Although he did beat the machine, he was truly amazed at the quality of the game that the computer played against him.

In playing the game, it is almost as though you are playing a person, rather than a computer. Sometimes you get the feeling that a move you've made has scared KIM and the computer proceeds to make an irrational move.

Some fairly good players have tried the Microchess program, and many have stated that the computer plays an aggressive game. Generally, it presses the offensive and tries to put you on the defensive for the remainder of the match. The opinion of most players was that the typical chess-club-level player will beat KIM consistently.

If you want to add a challenge to the game, simply remove one or more of your pieces at the beginning of the game by moving the computer's piece to the square of the man you wish to remove and back again to its original position. You will now be playing with fewer men than the computer and the advantage will be on the computer's side.

At times, you might find the computer is seeing a different placement of pieces on the board than you are. When this situation does occur, it is possible to take a look at the location of each piece as it is internally stored in the memory. The computer's pieces are in location 0050 to 005F; the player's pieces are in location 0060 to 006F.

To get a look at these memory locations, it is possible to exit the Microchess program and return to the KIM monitor to look at the data contained for the piece in question. The display will indicate the address in the first four digits and the rank and file of the square in the other two digits.

This feature is really useful if you enter one of your moves incorrectly or forget to press F to register your move. The error

All you need to play chess with your KIM.

can be corrected by inserting the correct data and then returning to the program by pressing PC and then GO to resume the game.

I found that when I tried this procedure my KIM would not exit the program. To rectify the problem I had to insert the vectors 17FA 00, 17FB 1C, 17FE 00 and 17FF 1C after loading the program at the start of the game.

Summary

This program demonstrates what a small hobby computer, with a minimum amount of memory, can accomplish. I am quite pleased with its chess-playing ability.

If you have a computer with additional memory, a CRT or Teletype terminal, the program can be expanded to include standard notation for the chessboard, graphic display and a much more sophisticated level of play. With the documentation provided in the Microchess program, the only thing to hold you back is your imagination.■

Reference

I. A. Horowitz, *Chess Openings, Theory and Practice,* Simon and Schuster, New York NY.

Microchess is available on KIM cassette for $10 from: Micro-Ware Limited, 27 Firstbrooke Rd., Toronto, Ontario M4E 2L2, Canada.

M9900 CPU-16 BIT MINI for the S-100 BUS with PASCAL

The M9900 CPU brings the most powerful single-chip processor available today—the TI TMS9900—to the S-100 bus and supports it with powerful software. Included with the CPU board are Disc Operating System, BASIC, Assembler, Linking Loader, Text Editor, and Interactive Debug. The powerful Pascal compiler is only $150 more.

Move up to a 16 bit machine and the power of Pascal without losing the economy and selection of the S-100 bus — move up to the M9900 CPU.

Marinchip Systems
M49
16 Saint Jude Road
Mill Valley, Ca. 94941
(415) 383-1545

Kit $550 Assembled $700 Documentation $20 tract or whatever you wish, simply push the desired operation key. The display will show the result in base 10.

7. To convert to the original number base at this point, press the R/S key. After some spurious flashing, the display will show the answer in the correct base. As in the entry mode for bases higher than 10, the answer must be read out in two digit groups.

If only conversion from base 10 is desired, enter the base-10 number, press GTO, 2, 7, R/S and see the answer displayed.

The examples in Fig. 1 will help to clarify any questions you may have.

Two's Complement

For relative jump calculations, the two's complement representation of a negative number is often needed. To find this form, just subtract the number from 1 followed by a number of zeros equal to the digits in the number. The example in Fig. 2 illustrates this technique.

How It Works

The first routine (conversion to base 10) utilizes the fact that a number in any base can be represented as a sum of the digits multiplied by the base raised to the power of the digit position (see Fig. 3). The flowchart shown in Fig. 4 shows how the method was implemented. The requirements of keeping one stack location free and minimizing the size of the stored program lead to a few tricks that should be explained.

The least significant digit of the number to be converted is separated by dividing the number by 10 or 100 (depending on the number of digits required to represent the largest digit in the base system). The integer result is then stored on the stack for the next loop (step 8). R_2 is initialized to 10 or 100 before starting the loop so that its contents represent (10 x

LINE	DISPLAY CODE	KEY ENTRY	X	Y	Z	T
00			N	OLD N		
01	24 07	RCL 7	M	N	OLD N	
02	23 02	STO 2	M	N	OLD N	
03	34	CLX	O	N	OLD N	
04	23 01	STO 1	O	N	OLD N	
05	22	R↓	N	OLD N		
06	24 07	RCL 7	M	N	OLD N	
07	71	÷	N/M	OLD N		
08	14 01	f INT	INT (N/M)	OLD N		
09	14 73	f LASTx	N/M	INT(N/M)	OLD N	
10	15 01	g FRAC	FR(N/M)	INT(N/M)	OLD N	
11	24 02	RCL 2	BN	FR(N/M)	INT(N/M)	OLD N
12	61	×	CN	INT(N/M)	OLD N	
13	23 51 01	STO + 1	CN	INT(N/M)	OLD N	
14	22	R↓	INT(N/M)	OLD N		
15	15 71	g x=0	INT(N/M)	OLD N		
16	13 20	GTO 20	INT(N/M)	OLD N		
17	24 00	RCL 0	B	INT(N/M)	OLD N	
18	23 61 02	STO × 2	B	INT(N/M)	OLD N	
19	13 05	GTO 05	B	INT(N/M)	OLD N	
20	22	R↓	OLD N			
21	24 01	RCL 1	CN	OLD N		
22	74	R/S	CN	OLD N		
23	13 27	GTO 27				
24	13 00					
25	13 00					
26	13 00					
27	23 06	STO 6	N			
28	34	CLX	O			
29	23 01	STO 1	O			
30	01	1	1			
31	23 03	STO 3	1			
32	24 06	RCL 6	N			
33	15 71	g x=0	N			
34	13 49	GTO 49	N			
35	24 00	RCL 0	B	N		
36	71	÷	N/B			
37	14 01	f INT	INT N/B			
38	23 06	STO 6	INT N/B			
39	14 73	f LASTx	N/B	INT N/B		
40	15 01	g FRAC	FRAC N/B	INT N/B		
41	24 00	RCL 0	B	FRAC N/B	INT N/B	
42	61	×	REM	INT N/B		
43	24 03	RCL 3	DP	REM	INT N/B	
44	61	×	CN	INT N/B		
45	23 51 01	STO + 1	CN	INT N/B		
46	24 07	RCL 7	M	CN	INT N/B	
47	23 61 03	STO × 3	M	CN	INT N/B	
48	13 32	GTO 32	M	CN	INT N/B	
49	24 01	RCL 1	CN			

REGISTERS	
R_0	base (B)
R_1	CONV. Num (CN)
R_2	$(\text{base})^n$ x 100 (BN)
R_3	Digit pos. (DP)
R_4	
R_5	
R_6	Temp Stor
R_7	10 (M)

Program listing.

Example: Convert 3C1 from base 16 to base 10

$$\begin{aligned} 3C1 &= (3 \times 16^2) + (12 \times 16^1) + (1 \times 16^0) \\ &= (3 \times 256) + (12 \times 16) + (1 \times 1) \\ &= 768 + 192 + 1 \\ &= 961 \end{aligned}$$

Fig. 3. Basic method for conversion to base 10.

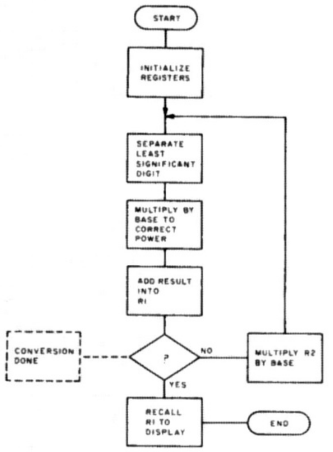


Fig. 4. Flowchart for conversion to base 10.

base^n) or (100 x base^n). This saves having to make the least significant digit (a fraction from step 10) back into an integer before multiplying by the base to the nth power. The sum of the results of the multiplications is accumulated in R_1.

When the loop is complete, R_1 is recalled and placed in the display (x register). Then, when the next number is keyed in, the x register is automatically pushed to the y register. As long as the program does not push too many numbers onto the stack, the contents of the y register are retained. The R↓ in step 20 makes sure this number is in the y register again at the end of the routine. Thus, after two passes, the two numbers converted are ready for whatever arithmetic operation you want to perform.

The second routine converts from base 10 using the successive divide-by-base routine shown by the example in Fig. 5. Fig. 6 is the flowchart.

The programming is fairly straightforward in this routine. After division by the base (step 36), the integer part of the quotient is stored (step 38) for the next loop. The remainder is recovered by multiplying the fractional part of the quotient by the base (step 42). A potential for error exists here because the fractional part will not always be exact. The round-off algorithms in the HP-25 seem to work well, and I have never had any problem with this technique.

Since the least significant digits come out first, the remainders are added into the result at step 45 after multiplication by 1, 10, 100 (or 1, 100, 1000) at step 44. When the quotient from the previous step is 0, conversion is complete, and the program branches to step 49, where it recalls register 1 to the display to show the final answer.

Program Limitations

As designed, the program handles only integer values. Fractions are not allowed. If the 00 after the decimal is not wanted, keying in f FIX 0 will remove it. There is also no error checking for validity of digits. There would be no mistake noted if a 9 were entered in an octal number, so double-check data entries.

Numbers too large to be handled as integers do not cause overflow. The calculator switches automatically to exponential mode. Interpreting the answer, however, would be difficult since digits are lost and the exponent would be a base-10 decimal shift. The system will handle hexadecimal numbers up to FFFFF (1,048,575 base 10).

Program Modifications

For single uses, some modifications could be made to reduce the number of keystrokes required to solve problems. For repeated conversion to base 10, the GTO 27 instruction at location 23 could be changed to GTO 00. This would eliminate continual pressing of the f PRGM keys before each conversion. Similarly, for conversions from base 10 to another base, the RCL 7 instruction at location 01 can be changed to GTO 27.

If, for some reason, you want to convert between two bases, neither of which is 10, use register 4 or 5 for the second base. Change references to register 0 in steps 35 and 41 to 4 or 5. The R/S instruction at 22 can be replaced with a GTO 27. Entering a number and pressing R/S will then cause the calculator to convert the number to base 10 and then the new base without stopping.

I hope you find this set of programs as useful as I have. For those who have other programmable calculators, it would be interesting to see how translating this program would work out. It could, of course, be translated into BASIC. There's a lot to do—have fun.■

Example: Convert 374 to octal using successive division technique.

374 ÷ 8 = 46 remainder 6
46 ÷ 8 = 5 remainder 6
5 ÷ 8 = 0 remainder 5
Therefore $374_{10} = 566_8$

Fig. 5. Conversion from base 10.

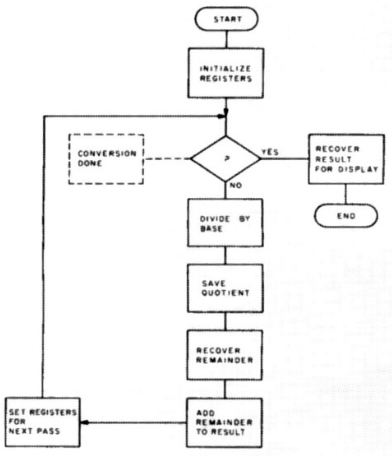


Fig. 6. Flowchart for conversion from base 10.